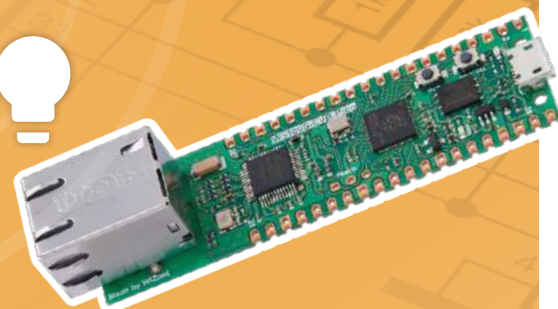


Kiesschijf-afstands-bediening

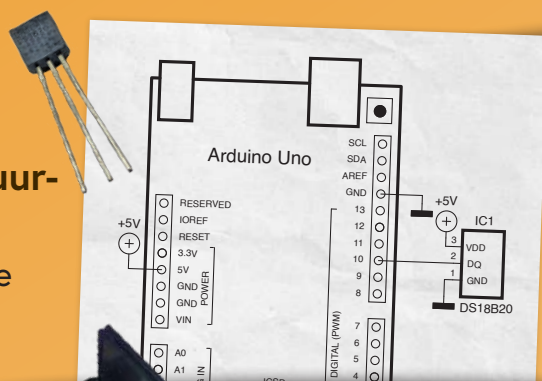
op basis van
de RP2040



FOCUS ON

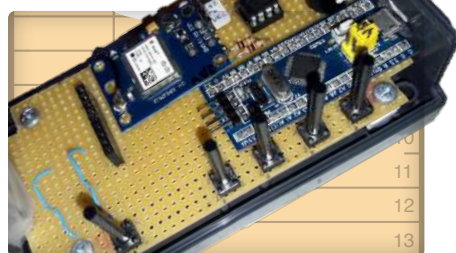
IoT & Sensors

DS18B20
temperatuur-
sensor
meten via de
1-Wire bus



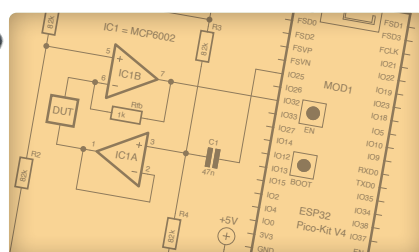
matter

De Matter-standaard
redding voor het Smart Home?



GPS-gebaseerde
snelheidsmeter
nooit meer bekeuringen

p. 11



Impedantie-analyzer
met ESP32
simpel en goedkoop!

p. 30



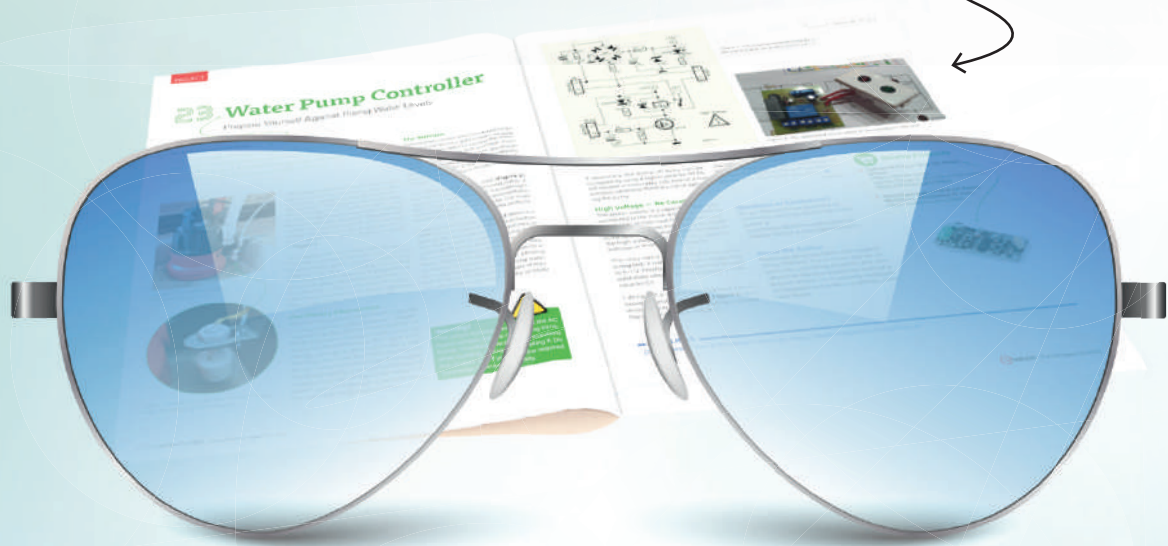
Doppler-bewegingssensor
bewegingen detecteren met
microgolven

p. 90



**BEREID
JE VOOR
OP EEN**

drukke zomer vol projecten!



Oproep aan alle elektronicaliefhebbers en -makers!
De Elektor Circuit Special 2023 verschijnt in augustus en staat weer vol met opmerkelijke schakelingen en projecten. Maak je klaar voor een diepe duik in de boeiende wereld van DIY-projecten en **LAAT JE CREATIVITEIT DE VRIJE LOOP!**

NOG EVEN GEDULD!

Elektor leden ontvangen deze speciale uitgave automatisch.
Niet leden kunnen terecht in de kiosk of de Elektor Store.



www.elektormagazine.nl/circuit-special

Elektor verschijnt acht keer per jaar en is een uitgave van

Elektor International Media B.V.

Postbus 11, 6114 ZG Susteren (Nederland)
Tel.: +31 (0)46 4389444

www.elektor.nl | www.elektormagazine.nl

Voor al uw vragen

service@elektor.nl

Lid worden

www.elektormagazine.nl/abo

Advertenties

Raoul Morreau

Tel. +31 (0)6 4403 9907

raoul.morreau@elektor.com

www.elektormagazine.nl/adverteren

Auteursrecht

© Elektor International Media B.V. - 2023

Niets uit deze uitgave mag vervaelvoudigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm of op welke wijze dan ook, zonder voorafgaande schriftelijke toestemming van de uitgever. De auteursrechtelijke bescherming van Elektor strekt zich mede uit tot de illustraties met inbegrip van de printed circuits, evenals de ontwerpen daarvoor. In verband met artikel 30 van de Rijksoctrooiwet mogen de in Elektor opgenomen schakelingen slechts voor particuliere of wetenschappelijke doeleinden vervaardigd worden en niet in of voor een bedrijf. Het toepassen van de schakelingen geschiedt buiten de verantwoordelijkheid van de uitgever. De uitgever is niet verplicht ongevraagd ingezonden bijdragen, die hij niet voor publicatie aanvaardt, terug te zenden.

Indien de uitgever een ingezonden bijdrage voor publicatie aanvaardt, is hij gerechtigd deze op zijn kosten te (doen) bewerken. De uitgever is tevens gerechtigd een bijdrage te (doen) vertalen en voor haar andere uitgaven en activiteiten te gebruiken tegen de daarvoor bij de uitgever gebruikelijke vergoeding.

Druk

Senefelder Misset, Mercuriusstraat 35,
7006 RK Doetinchem (Nederland)

Distributie

Betapress, Nederland – AMP, België



Jens Nickel

Hoofdredacteur Elektor Magazine



Griezelen en pret met AI

Wat is het tegendeel van Marlon Brando? Weet je het niet? Kijk dan naar de (griezelige) YouTube-video waar onze auteur Ilse Joostens naar linkt in haar artikel op pagina 46. Deze keer gaat de column over kunstmatige intelligentie, en dan vooral over de talloze tools die afbeeldingen, tekst en programmacode genereren. AI maakt nu trouwens ook complete video's, en ook hier zijn de resultaten verbluffend, soms grappig maar vaak ook een beetje beangstigend. Je kunt je afvragen waarom kunstmatige intelligentie zo vaak horror- en apocalyptische beelden genereert – zelfs in onschuldige pizzareclames in de stijl van de jaren '90 (www.youtube.com/watch?v=MpvEXrhnoyw).

Wij natuurlijke intelligenties zijn de situatie echter nog steeds meester. En zolang dat het geval is, moeten we er het beste van maken. Vooral programmeurs en elektronica-ingenieurs hebben hier veel mogelijkheden en daarom zijn we bij Elektor op onderzoek uitgegaan. Ik weet zeker dat je binnenkort met behulp van AI een geschikt artikel kunt zoeken in ons enorme tijdschriftenarchief. Handmatig zou het jaren duren om de ontelbare dwarsverbanden tussen de projecten en basisartikelen op te sporen en het geheel in een complete *body of knowledge* te gieten (als we daar al een werkbaar systeem voor zouden kunnen bedenken).

We zullen ook het automatisch genereren van programmacode behandelen in de komende uitgaven. Ik heb vrienden voor wie ChatGPT de gewoonste zaak van de wereld is om batch-bestanden te schrijven voor het bewerken van audiobestanden. Ik ben benieuwd wat er nog meer in petto is, vooral voor elektronica-ingenieurs. Misschien heb je zelf al geëxperimenteerd? Heb je goede of slechte ervaringen? Schrijf me op redactie@elektor.com!



Deel met Elektor!

Je elektronica-expertise is welkom! Heb je een voorstel voor een artikel of een elektronica-tutorial op video, of een idee voor een boek? Bekijk onze gids voor auteurs en inzendingen op

www.elektormagazine.com/submissions



Elektor Labs: ideeën & projecten

Het Elektor Labs-platform is voor iedereen toegankelijk. Post elektronica-ideeën en -projecten, bespreek technische uitdagingen en werk samen met anderen op

www.elektormagazine.com/labs

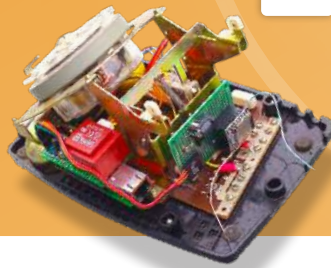
Ons team

Internationaal hoofdredacteur: Jens Nickel | **Content Director:** C.J. Abate | **Internationale redactie:** Asma Adhimi, Roberto Armani, Eric Bogers, Jan Buiting, Stuart Cording, Rolf Gerstendorf (RG), Ton Giesberts, Hedwig Hennekens, Alina Neacsu, dr. Thomas Scherer, Clemens Valens, Brian Tristram Williams | **Vaste medewerkers:** David Ashton, Tam Hanna, Priscilla Haring-Kuipers, Ilse Joostens, prof. dr. Martin Ossmann, Alfred Rosenkränzer | **Vormgeving & Layout:** Harmen Heida, Sylvia Sopamena, Patrick Wielders | **Directeur:** Erik Jansen | **Technische vragen:** redactie@elektor.com

Kiesschijftelefoon als afstandsbediening

voor licht kies 1;
voor koffie kies 2

6



Rubrieken

- 3 Griezelen en pret met AI**
- 23 Alle begin...**
...volgt de emitter
- 26 Developer's Zone**
vrij te kiezen onafhankelijke hystereseniveaus voor comparatoren
- 38 Oost West Lab Best**
zelfbouw moet gestimuleerd!
- 46 Uit het leven gegrepen**
modern luddisme
- 110 Vreemde onderdelen**
microprocessoren voor embedded systemen
- 124 Retrotronica**
transverter voor de 70cm-band
- 126 Ethics in Electronics**
Climate Calling Engineers
- 130 Hexadoku**
puzzelen voor elektronici

Achtergrond en info

- 40 Het MCCAB Arduino Nano Training Board**
alle hardware voor de "Microcontroller Hands-On Course"
- FOCUS**
- 48 Sensor-ABC: de DS18B20 temperatuursensor**
aansluiting op de 1-Wire bus

FOCUS

- 88 Bouw een cool IoT-display**
met de Phambili Newt
- 97 Een gids voor puristisch programmeren (deel 1)**
voor STM32 en andere controllers
- 106 Siglent SDM3045X-multimeter**
- 112 Microcontroller-documentatie verklaard**
deel 3: blokschema's en meer

Industry

FOCUS

- 54 Gaan Matter en Thread het Smart Home redden?**
nieuwe standaarden om het slimme huis te vereenvoudigen

FOCUS

- 58 Een kwestie van samenwerking**
ontwikkelen met het Thing Plus Matter Board en Simplicity Studio

FOCUS

- 62 Elektor infographic**
IoT en sensoren

FOCUS

- 64 Matter, ExpressLink, Rainmaker – waar hebben we het over?**
Q&A met Amey Inamdar van Espressif

FOCUS

- 68 De juiste keuze van microcontroller-ontwikkelkits**
voor IoT- en IIoT-toepassingen
- 74 Condensatoren gedragen zich niet altijd capacitief!**



Bouw een cool IoT-display met de Phambili Newt

88



Low-power LoRa-weerstation

bouw zelf een weerstation met een groot bereik

116

Projecten

FOCUS

- 6 **Kiesschijftelefoon als afstandsbediening**
voor licht kies 1; voor koffie kies 2

FOCUS

- 11 **GPS-gebaseerde snelheidsmeter**
nooit meer bekeuringen

- 16 **RGB-stroboscoop met Arduino**
kleurige toepassing van een nuttig instrument

FOCUS

- 20 **Draadloze nooddrukknop**
meer veiligheid met LoRa

- 30 **Impedantie-analyzer op basis van een ESP32**
simpel, weinig onderdelen en goedkoop!

- 78 **Een NTP-klok met CircuitPython**
waarom zou je deze programmeertaal gebruiken?

FOCUS

- 90 **De doppler-bewegingssensor HB100**
theorie en praktijk

FOCUS

- 116 **Low-power LoRa-weerstation**
bouw zelf een weerstation met een groot bereik

Binnenkort

Elektor Summer Circuits Special (augustus/september 2023)

In de traditie van de Halfgeleidergidsen van weleer zal het volgende nummer extra dik uitvallen, met meer dan 50 DHZ-projecten, retro-schakelingen, tips en trucs en veel meer!

Een greep uit de inhoud:

- > actieve gelijkrichter
- > goedkope frequentiestandaard
- > simpele dynamiekcompressor
- > kleine zonnevoeding
- > THD-generator
- > programmeerbare video-DAC
- > groot RGB-cijferdisplay
- > ChatGPT en Arduino
- > kerst-FM-radiobal op zonne-energie
- > kleine DCF77-simulator

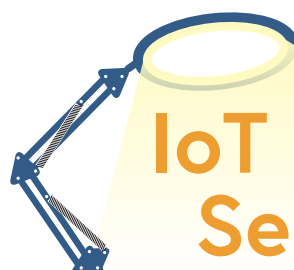
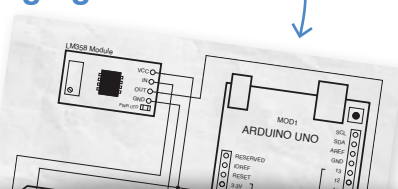
...en nog veel meer!

Aankondigingen onder voorbehoud.

De Elektor Summer Circuits Special 2023 verschijnt omstreeks 9 augustus 2023.

De doppler-bewegingssensor
HB100
theorie en
praktijk

90



FOCUS ON

IoT & Sensors

Kiesschijftelefoon als afstandsbediening

Clemens Valens (Elektor)

Zelfs wanneer verlichting en apparaten worden bestuurd door een domoticsysteem, is het vaak wenselijk een extra mogelijkheid te hebben om een lamp, een ventilator of een ander apparaat in of uit te schakelen. De in dit artikel voorgestelde gewijzigde kiesschijftelefoon maakt dit mogelijk door het nummer van het apparaat te draaien. Tegelijkertijd is het een trendy decoratief object.



In dit project bouwen we een oude analoge telefoon met draai- of kiesschijf om tot een afstandsbediening en alarm voor een domoticsysteem. Behalve als (decoratieve) afstandsbediening kan de omgebouwde telefoon ook worden gebruikt als rekwisiet in (bijvoorbeeld) een escaperoom. En er zijn vast nog veel meer toepassingen denkbaar.

Ontwerp op basis van de RP2040

Het brein voor dit project is een RP2040-microcontroller gemonteerd op een WIZnet W5100S-EVB-Pico-board (figuur 1). Dit is eigenlijk een Raspberry Pi Pico-board met een extra W5100S 'Internet'-chip (en een Ethernet-aansluiting). Daarom is het pincompatibel met het populaire Pico-board. Het enige waar je op moet letten is dat sommige pinnen (GPIO16 tot GPIO21) worden gebruikt om met de W5100S te communiceren.

Bedraad versus draadloos

Hoewel draadloze netwerken vandaag de dag de standaard lijken te zijn, zijn er nog veel toepassingen

voor bekabeld of vast Ethernet. Een voordeel van bekabeld Ethernet is dat de aangesloten knooppunten van stroom kunnen worden voorzien. Deze mogelijkheid wordt in dit project gebruikt (zie verderop). Bovendien hadden kiesschijftelefoons altijd al een kabel naar een wandcontactdoos, en het zou dus vreemd zijn om er een te hebben zonder zo'n kabel. Daarom is het W5100S-EVB-Pico-board een verdedigbare keuze voor deze toepassing.

Afstandsbediening via MQTT

De home automation controller (HAC) die door de kiesschijftelefoon wordt aangestuurd is Home Assistant (HA), een populair automatiseringsplatform dat elke dag meer aanhangers krijgt. Maar omdat de afstandsbediening MQTT spreekt, kan hij ook gemakkelijk worden geïntegreerd in andere domoticsystemen.

De MCU aansluiten op de telefoon

De telefoon die ik gebruikte, een klassiek Frans model (S63), produceert pulsen in een tempo van 10 Hz.

Het kiesmechanisme van de klassieke telefoon (**figuur 2**) kan worden gezien als twee schakelaars: de ene geeft aan dat er gebeld wordt (bezet/in rust), terwijl de andere afhankelijk van het gekozen cijfer een bepaald aantal keren sluit. Een '1' geeft één puls, een '9' negen pulsen en een '0' is tien pulsen.

Het aansluiten van dit mechanisme op het W5100S-EVB-Pico-board is simpel; aansluiting zonder de telefoon aan te passen vereist wat meer denkwerk. Ik wilde de telefoon zo goed mogelijk in de oorspronkelijke staat houden. Door de juiste aansluitpunten in de telefoon te kiezen en de vereiste pull-up weerstanden aan te passen, is dat ook mogelijk (**figuur 3**).

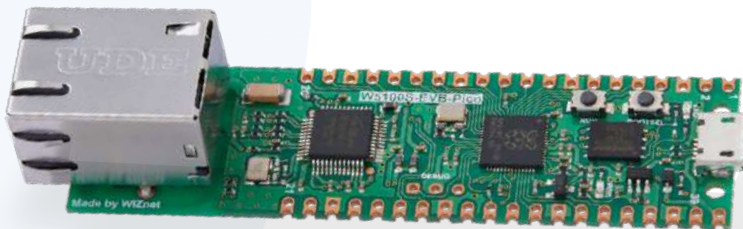
De hoorn rust op een normaal open-schakelaar (NO), dus het opnemen van de hoorn sluit deze schakelaar. Dit verandert de elektrische schakeling van de telefoon en daarmee de impedantie van enkele aansluitpunten van

het kiesmechanisme. Gelukkig kan dit worden opgevangen door relatief lage waarden te kiezen voor sommige pull-up weerstanden.

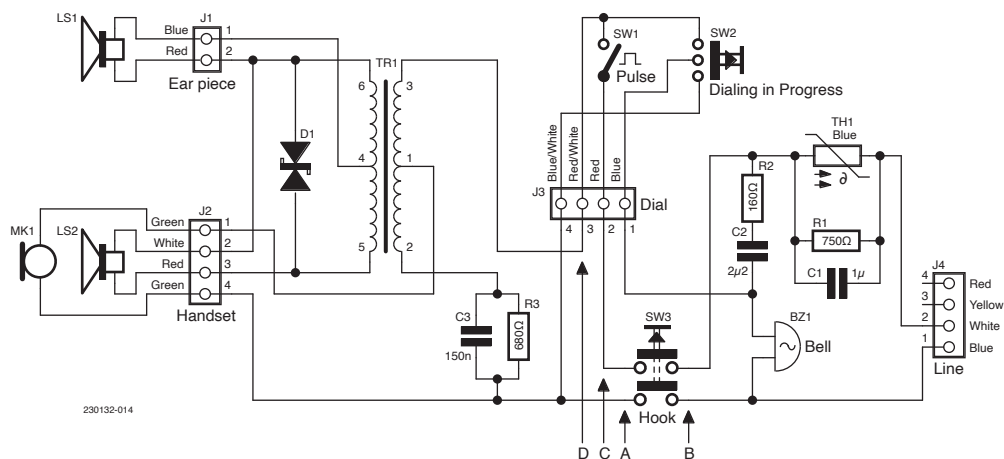
Hoogspannings-belomvormer

De telefoonbel (**figuur 4**) heeft een relatief hoge wisselspanning nodig om te rinkelen, minstens 35 V_{AC} zoals ik gemeten heb. Daarom heb ik een eenvoudige low-power spanningsomvormer gebouwd die door de MCU wordt aangestuurd. De MCU genereert twee 50Hz-blokgolven met een faseverschil van 180° die de secundaire wikkelingen van een kleine 230V_{AC}-nettransformator aansturen. De bel wordt aangesloten op de primaire van die transformator. Een kleine 2×9 V, 1 VA transformator volstaat om een behoorlijk belvolume te krijgen. Het complete schema van de telefoon/microcontroller-interface is getekend in **figuur 5**.

Figuur 2. Het kiesmechanisme van de S63-telefoon produceert pulsen met een frequentie van 10 Hz.



Figuur 1. Het WIZnet W5100S-EVB-Pico-board bezit een RP2040-microcontroller die is verbonden met een W5100S Internet-over-Ethernet IC. Het board heeft dezelfde pinout als het Raspberry Pi Pico-board. Merk echter op dat GPIO16 tot GPIO21 ook verbonden zijn met de W5100S-chip (bron: WIZnet).



Figuur 3. Om de microcontrollerprint op de telefoon aan te sluiten zonder de telefoon te modiëren, heb ik gekozen voor de aansluitpunten A (GND), B (van de haak), C (kiespulsen) en D (kiezen bezig).

Figuur 4. Het belmechanisme van de oude telefoon.



MP3-speler

Ik heb nog een MP3-module toegevoegd voor het afspelen van audiobestanden via de luidspreker in de telefoonhoorn. De MP3-module communiceert met de MCU via een 9.600-baud seriële poort. De HAC kan MQTT-berichten naar de telefoon sturen om het af te spelen MP3-bestand te selecteren, maar de MCU kan hem ook rechtstreeks bedienen. Zo kan wachtmuziek worden afgespeeld, kunnen analoge telefoonlingeluiden worden gereproduceerd, kunnen vooraf opgenomen berichten worden afgespeeld of kan een sprekende klok worden gemaakt. Geluids- en muziekbestanden worden opgeslagen op een microSD-kaart.

Wegens ruimtegebrek is de MP3-speler samen met de voedingsconnector op een tweede (gaatjes)print gemonteerd. De beide printen zijn verbonden via J3 en J4 (figuur 6). De luidspreker in de hoorn is rechtstreeks aangesloten op de uitgang van de MP3-speler.

Voeding

De W5100S-EVB-Pico heeft een $5V_{DC}$ -voeding nodig. De hoogspannings-belomvormer heeft minstens $10V_{DC}$

nodig. Voor de voeding van de telefoon worden de twee vrije aderparen van een normale Ethernet-kabel gebruikt (figuur 7). Mijn vijf meter lange kabel introduceert een merkbaar spanningsverlies, zelfs met twee aders parallel voor elke voedingsaansluiting. Een 12 V AC/DC netadaptor voorziet de telefoon daarom van stroom. In de telefoon zorgt een kleine DC/DC-converter voor de 5 V voor het MCU-board.

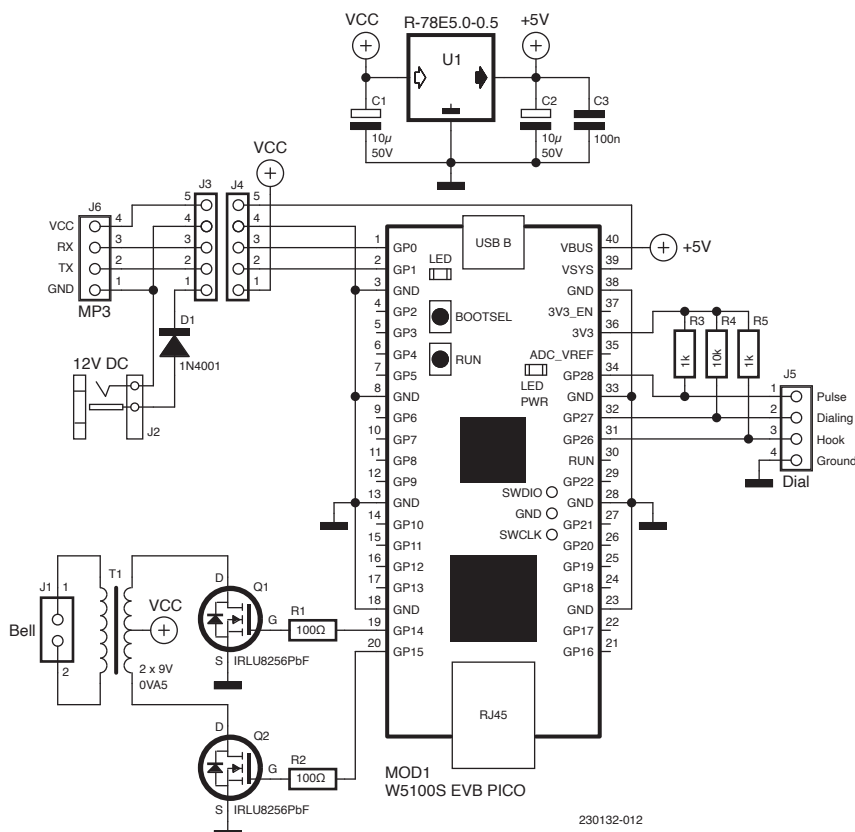
Ondersteunende software

Het W5100S-EVB-Pico-board wordt ondersteund door bibliotheken voor gebruik met de officiële Raspberry Pi RP2040 MCU-ontwikkelomgeving. Ik hou niet zo van deze omgeving, omdat ik het complex en onoverzichtelijk vind met al het Cmake-spul. Gelukkig bestaat er ook een Arduino Boards package voor de RP2040, en dus heb ik de WIZnet-bibliotheken naar deze IDE overgezet. Dit maakt alles veel eenvoudiger en, denk ik, toegankelijker en gemakkelijker te delen. Het complete toepassingsprogramma bestaat nu uit een Arduino-sketch en een eenvoudig te installeren bibliotheek [1].

MQTT

Ik bouwde het programma voor de afstandsbediening op het MQTT-voorbeeld van WIZnet op. Hierdoor kon ik snel aan de slag. Het enige probleem dat ik tegenkwam was een ontbrekende en ongedocumenteerde aanroep

Figuur 5. Er zijn slechts enkele onderdelen nodig om de microcontroller met de telefoon te verbinden. Merk op dat de pull-up weerstanden voor de kiesschakelaars (R3, R4 en R5) niet allemaal dezelfde waarde hebben.



naar `MilliTimer_Handler` (zie het bestand `mqtt_interface.h`). Toen ik die eenmaal had toegevoegd aan mijn millisecondentimer-callback, werkte MQTT prima.

mDNS

Omdat mijn Home Assistant-installatie DHCP gebruikt om verbinding te maken met het netwerk, kan hij elk moment een nieuw IP-adres krijgen. Om zijn randapparatuur te vinden en ermee te communiceren, gebruikt HA multicast-DNS, ook wel mDNS genoemd. Omdat ik geen tijdelijk IP-adres hard in mijn programma wilde coderen om het dan telkens weer te compileren als het adres van de HAC veranderde, heb ik mDNS-ondersteuning toegevoegd. Hiervoor heb ik de WIZnet DNS-bibliotheek aangepast, omdat mDNS sterk lijkt op gewone DNS. Nu stuurt het programma een mDNS-verzoek om het IP-adres van de HAC te verkrijgen voordat het probeert er verbinding mee te maken. Dit maakt het systeem veel flexibeler en betrouwbaarder.

Hoorndetectie en kiezen

Het uitlezen van de schakelaars van het kiesmechanisme vereist ontdendering. Dit wordt softwarematig uitgevoerd. Eenmaal goed ontdenderd is het tellen van de pulsen een fluitje van een cent.

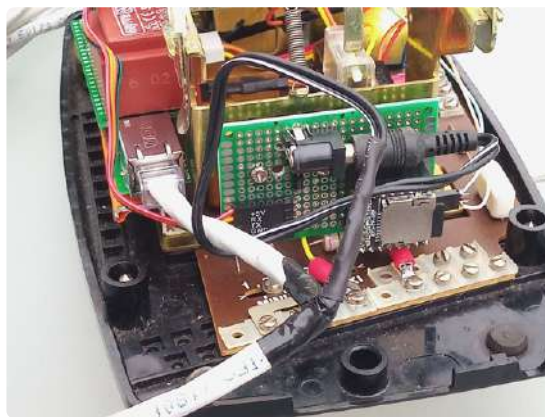
Kiezen met de hoorn op de haak genereert berichten met één cijfer; kiezen met de hoorn van de haak levert een meercijferig nummer op, dat wordt verzonden nadat de hoorn is neergelegd. Het cijfer of nummer wordt verzonden als de payload van een MQTT-packet. Cijfers worden 'min één' verzonden, zodat de 10 waarden in één teken passen. Dus 1 wordt verzonden als 0, 9 als 8, en 0 wordt verzonden als 9.

Tekstberichten

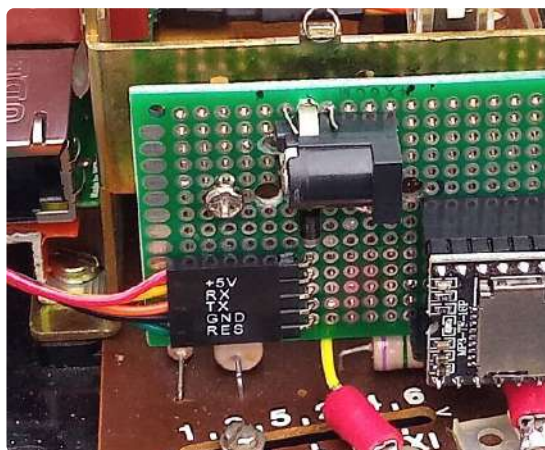
Wanneer de hoorn wordt opgenomen, levert het kiezen naast een meercijferig nummer ook letters op om tekstberichten mee samen te stellen. De 26 letters van het alfabet zijn verdeeld over de cijfers 2 tot en met 9, drie tekens per cijfer, behalve de 6 die er maar twee heeft ('m' en 'n'). Het cijfer 0 heeft er ook twee ('o' en 'q'), het cijfer 1 heeft er geen. Dit is de standaardindeling die op het Franse S63-toestel staat (**figuur 8**). Om een of andere reden is er geen 'z', en dus heb ik die toegevoegd aan het cijfer 0. Om een letter te kiezen, moet het bijbehorende cijfer maximaal drie keer worden gekozen. Bijvoorbeeld: 'a' is 2, 'b' is 22 en 'c' is 222. Om 'aaa' te schrijven is de kiesvolgorde 2-2-2, voor 'abc' is het 2-22-222. Het streepje staat hier voor een pauze van ten minste één seconde. Een pauze van minder dan een seconde tussen twee identieke cijfers selecteert de volgende letter die aan het cijfer wordt toegekend. Deze methode lijkt sterk op die welke vanaf 2000 op mobiele telefoons wordt gebruikt om tekstberichten samen te stellen, behalve dat er geen display is. Je moet je bericht in gedachten bijhouden (goede oefening!). Wanneer de hoorn weer op de haak wordt gelegd, wordt



Figuur 8. Sms'en met deze oude telefoon is gemakkelijk dankzij de letters die rond de kiesschijf is afgedrukt. Zo wordt 'elektor' gekozen als 33-555-33-55-8-0-77, waarbij elk streepje staat voor een pauze van één seconde.



Figuur 7. Power-over-Ethernet op de inefficiënte manier. Het werkt zolang de ingangsspanning hoog genoeg is om de spanningsval over de kabel te compenseren.

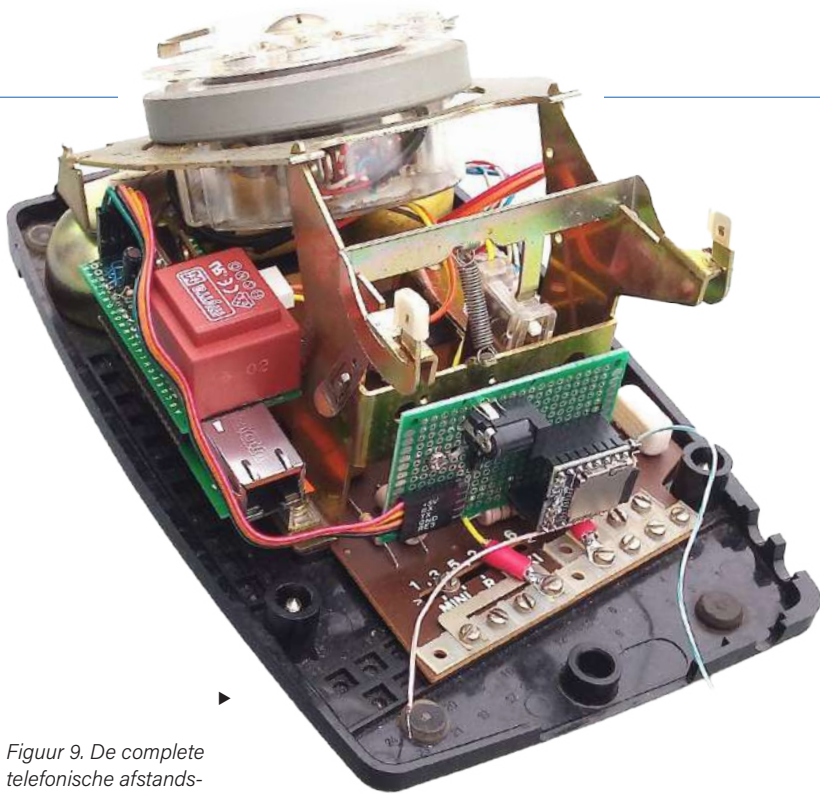


Figuur 6. De MP3-speler module en de voedingsconnector zitten op dezelfde kleine print.

het samengestelde bericht samen met het meercijferige nummer naar de HAC gestuurd als payload van een MQTT-packet. De HAC kan het dan doorsturen naar bijvoorbeeld een sms-dienst, of het op zijn dashboard weergeven.

Misschien wil je wat veranderen?

De in dit artikel voorgestelde interface is ontworpen voor een Franse telefoon van het S63-type en zo gebouwd dat hij er ook in past (**figuur 9**). Mogelijk moet je iets aanpassen voor gebruik met een ander model telefoon. In plaats van Ethernet kun je ook kiezen voor WiFi of een



Figuur 9. De complete telefonische afstandsbediening. Links zie je de transformator van de hoogspanningsomvormer die bovenop de W5100S-EVB-Pico-kaart is geprikt. Een kabel verbindt dit geheel met de MP3-speler module en de voedingsconnector rechts. De aansluitingen naar het kiesmechanisme zijn niet zichtbaar, omdat die zich aan de andere kant van de telefoon bevinden.

andere draadloze communicatiemethode. Dat is uiteraard mogelijk. De software kan gemakkelijk worden aangepast aan andere fysieke layers, aangezien het toepassingsprogramma gebruik maakt van een standaard netwerkdriver-API. De TCP/IP-stack wordt compleet verzorgd door het W5100SIC, en kan zodoende worden vervangen door een andere communicatiemodule.

Door een goede accu (die af en toe moet worden opgeladen) toe te voegen zou een onbekabelde afstandsbediening mogelijk zijn. Als de oorspronkelijke bedrading van de telefoon kan worden verwijderd, dan is er genoeg ruimte voor zo'n accu.

De hoogspanningsomvormer voor de bel kan op verschillende manieren worden gebouwd. Ik heb een transformator gebruikt omdat ik er een bij de hand had, maar een geschikte online verkrijgbare goedkope omvormermodule voldoet waarschijnlijk ook.

VoIP

Momenteel wordt de microfoon van de telefoon niet gebruikt, maar je kan je voorstellen deze te gebruiken voor een voice-over-internet of VoIP-dienst. De RP2040 ondersteunt I²S, zodat standaard microfooninterface-IC's hiervoor kunnen worden gebruikt.

De afstandsbediening bestaat uit een Arduino-sketch en een eenvoudig te installeren Arduino-bibliotheek op basis van de officiële WIZnet-repository. Deze is te vinden op [1]. ◀

230132-03

Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur via clemens.valens@elektor.com of naar de redactie van Elektor via redactie@elektor.com.



Functies van de afstandsbediening

De gewijzigde kiesschijftelefoon heeft de volgende functies:

- > Wanneer de hoorn op de haak ligt en er een nummer wordt gekozen, wordt het bijbehorende apparaat in- of uitgeschakeld, afhankelijk van de momentele toestand.
- > Als de hoorn van de haak wordt genomen, kan de draaischijf worden gebruikt om (moeizaam) een tekstbericht samen te stellen op een manier die vergelijkbaar is met mobiele telefoons van zo'n 20 jaar geleden. Als de hoorn weer op de haak wordt gelegd, wordt het bericht verzonden.
- > De bel van de telefoon is beschikbaar voor de domoticacontroller (HAC) als alarm en kan bijvoorbeeld worden omgeleid naar de deurbel, of dienst doen als keukentimer of (onprettige) wekker.
- > De luidspreker van de telefoon is verbonden met een MP3-speler om vooraf opgenomen berichten of muziek af te spelen. Zo kan bijvoorbeeld een sprekende klok worden geïmplementeerd, zoals in de vorige eeuw gebruikelijk was.

De communicatie tussen de telefoon en de HAC is gebaseerd op MQTT, terwijl mDNS wordt gebruikt om automatisch een verbinding tussen de twee tot stand te brengen.



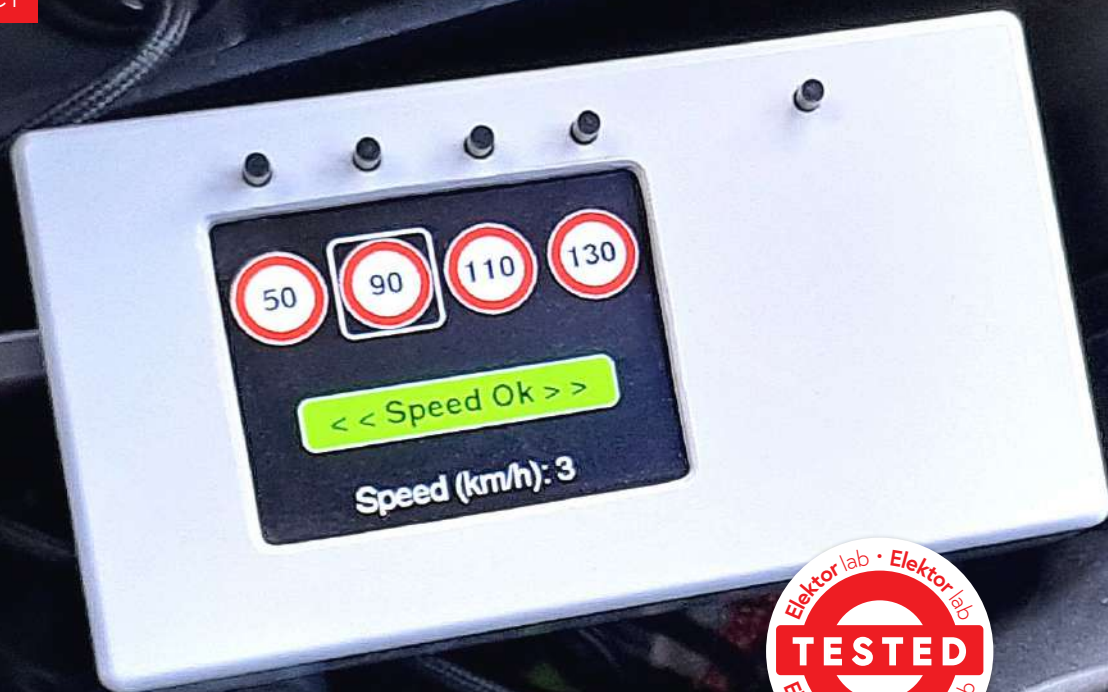
Gerelateerde producten

- > **WIZnet W5100S-EVB-Pico RP2040-based Evaluation Board (SKU 19971)**
www.elektor.nl/19971
- > **WIZnet Ethernet HAT for Raspberry Pi Pico (SKU19970)**
www.elektor.nl/19970
- > **C. Valens, Mastering Microcontrollers Helped by Arduino (3rd Edition) (SKU 17967)**
www.elektor.nl/17967



WEBLINKS

[1] Dit project op GitHub: https://github.com/polyvalens/rotary_dial_remote



GPS-gebaseerde snelheidsmeter

nooit meer bekeuringen

Olivier Croiset (Frankrijk)

Er zijn veel voorbeelden op internet die laten zien hoe je een GPS-module kunt gebruiken in een microcontroller-project. Maar wat kun je met die GPS-coördinaten doen, behalve ze op een kaart in te tekenen? Ik wilde iets bruikbaars maken en bedacht deze GPS-gebaseerde snelheidsmeter. Dit apparaat kan in elk voertuig worden gebruikt en stelt je in staat te rijden zonder op de snelheidsmeter te hoeven letten, en voorkomt onaangename verrassingen achteraf in de brievenbus.

Het hier gepresenteerde apparaat is geen snelheidsbegrenzer, waarvoor aanpassing aan het voertuig nodig zou zijn, maar een alarm dat signaleert wanneer een vooraf ingestelde snelheid wordt bereikt (en overschreden). Om ervoor te zorgen dat je je ogen op de weg

gericht kan houden, laat de zoemer twee lange pieptonen horen als de ingestelde snelheid wordt overschreden, terwijl twee korte pieptonen aangeven dat het voertuig weer een snelheid onder de grenswaarde heeft bereikt.

Vier presets

De snelheidsmonitor heeft vier programmeerbare snelheidslimieten, wat voldoende zou moeten zijn voor de meeste ritten en situaties die je tegenkomt. Bovendien laat het scherm nauwelijks meer dan die vier limieten zien. Onder het rijden moet de bestuurder snel een preset kunnen kiezen, indien mogelijk zonder naar het apparaat te kijken. Ik wilde geen aanraakscherm gebruiken; de bestuurder moet de knopdruk fysiek ervaren. Om ergonomische redenen zijn de knoppen boven de vier snelheidswaarden op het display geplaatst.

Een goed excuus om met 32 bit aan de slag te gaan

Mijn eerste programma's met een 8-bit ATmega328-microcontroller in combinatie met een GPS-module bewezen al snel dat het 32 KB flash-geheugen niet genoeg zou zijn voor een meer geavanceerde toepassing, vooral als de gegevens grafisch op een TFT-scherm moeten worden weergegeven.

Daarom besloot ik, hoewel overtuigd van de vele kwaliteiten van de ATmega328, het hogerop te zoeken en



een STM32 32bit-microcontroller uit te proberen. De meest geschikte uitvoering voor knutselaars zoals ik is het BluePill-board met een STM32F103C8-chip. Dit kleine board vergemakkelijkt het gebruik van deze microcontroller aanzienlijk. Het is eenvoudig te programmeren via de USB-poort met behulp van de Arduino IDE (mits het board vooraf is voorzien van een geschikte bootloader), het is compact en niet duur (slechts een paar euro).

Flash-geheugenformaten

Dit project is mijn eerste poging om de STM32F103C8 te gebruiken. Merk op dat officieel de C8 is uitgerust met 64 KB flash, terwijl de CB-versie 128 KB bezit, maar soms komt de C8 ook met 128 KB (namaak?). Hierdoor kunnen degenen die in de verleiding komen, hun kennis vergroten door andere periferie toe te voegen, zoals een SD-kaart, een SIM-module, draadloze gegevensoverdracht enzovoort.

De GPS-module

De GPS-module die in dit project wordt gebruikt, is een NEO-6-module van u-blox. Het gebruik ervan is eenvoudig, omdat de TinyGPS+ bibliotheek [2] al het decodeerwerk van de NMEA 0183-geformatteerde gegevens-

stroom die door de GPS-module wordt geleverd, voor zijn rekening neemt. Dit zijn de belangrijkste parameters die we op deze manier verkrijgen:

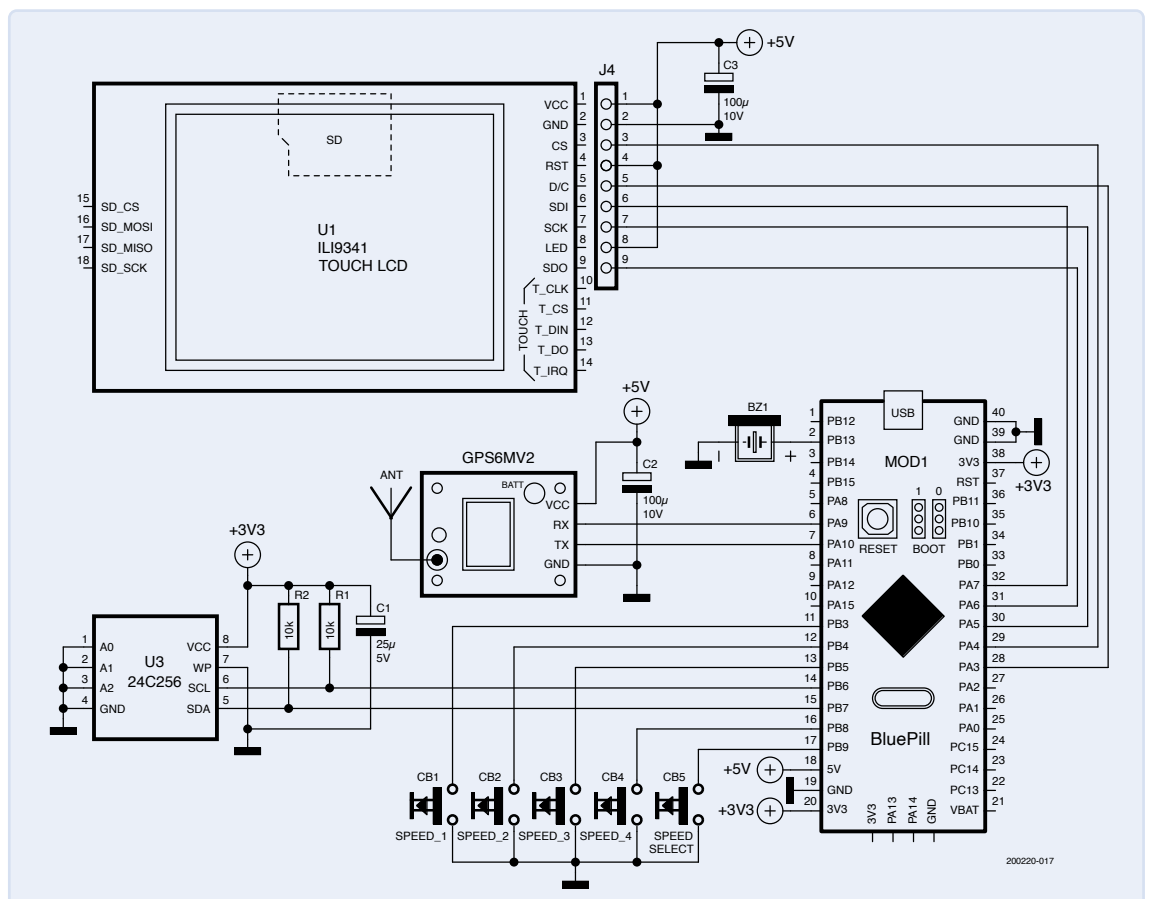
- UTC-datum en -tijd;
- lengte- en breedtegraad (in decimale graden);
- snelheid;
- richting;
- hoogte;
- het aantal zichtbare satellieten;
- de HDOP-waarde.

De *Horizontal Dilution of Precision* ofwel HDOP [2] hangt af van de positie van de satellieten binnen het bereik van de GPS-ontvanger. Hoe lager deze waarde is (dichtbij 1), des te nauwkeuriger de berekende coördinaten. Een HDOP-waarde van 10 geeft aan dat de coördinaten niet erg nauwkeurig of zelfs ongeldig zijn. De snelheidsmonitor gebruikt deze parameter niet.

Het TFT-scherm

Als display gebruikte ik een standaard 2,2" TFT-display-module met een ILI9341-driverchip en SPI-interface. Het heeft een resolutie van 320×240 pixels, wat genoeg is voor de gegevens die we willen weergeven (vier

Figuur 1. De snelheidsmonitor combineert kant-en-klare modules met een paar elektronische componenten.



snelheidslimieten plus een kort bericht dat aangeeft of de limiet is bereikt of niet).

De snelheidsmonitor heeft bij normaal gebruik twee hoofd-weergavemodi. De momentele weergavemodus wordt opgeslagen in het EEPROM en wordt opgeroepen wanneer het systeem opnieuw wordt ingeschakeld. De snelheids-presets worden ook opgeslagen in EEPROM. Van het EEPROM worden slechts zes bytes gebruikt: vier bytes voor de presets, één byte voor het laatst geselecteerde scherm en één byte voor de laatst geselecteerde snelheidslimiet.

Het schema

Het schema van de snelheidsmonitor is gegeven in **figuur 1**. De BluePill-module in de rechterbenedenhoek vormt het hart van de schakeling.

De voeding wordt aangesloten via de USB-connector van de BluePill. Nieuwere voertuigen zijn vaak voorzien van een USB-aansluiting; anders is een 12V-naar-USB-adapter vereist (of gebruik een powerbank). De BluePill-module wordt gevoed met 5 V; hij heeft zijn eigen 3,3V-regelaar. Deze laatste voorziet het EEPROM van 3,3 V. Het TFT-display en de GPS-module worden gevoed met 5 V. Het totale verbruik van de schakeling is ongeveer 200 mA. Het display wordt aangestuurd via een SPI-bus, en de GPS-module wordt aangestuurd via een seriële link. Het EEPROM is aangesloten op de I²C-bus (met zijn twee pull-up weerstanden). Voor mijn prototype gebruikte ik een 32 KB 24C256 EEPROM, maar een 24C01 van 128 byte zou al meer dan groot genoeg zijn.

Twaalf GPIO-poorten blijven vrij. Het is dus mogelijk om meer periferie, features en functionaliteit toe te voegen – zolang het applicatieprogramma maar in het flash-geheugen past. De gecompileerde firmware van dit project neemt 64.020 bytes in beslag en daarmee bijna het volledige flash-geheugen.

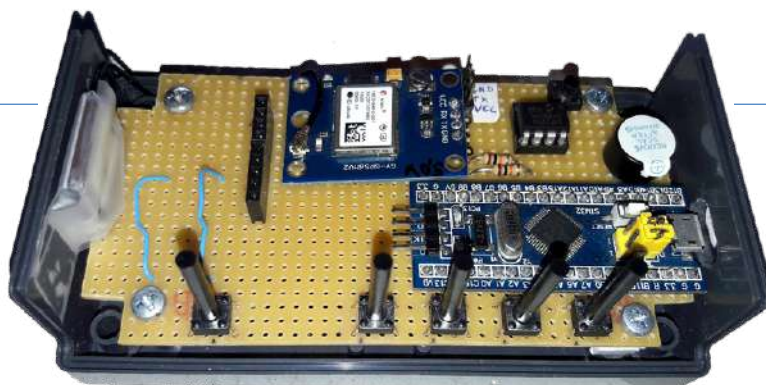
Bouw van de snelheidsmonitor

Het belangrijkste criterium bij de bouw van dit project is ergonomie. De chauffeur moet de knoppen blindelings kunnen vinden. De druktoetsen hebben een toetslengte van 25 mm – de langste die ik kon vinden.

Ik heb mijn prototype opgebouwd op gaatjesprint (**figuur 2**). Een echte print (PCB) zou een compactere uitvoering mogelijk maken in een plattere behuizing. De USB-connector moet toegankelijk blijven, aangezien deze wordt gebruikt om het circuit bij normaal gebruik van stroom te voorzien en om de software te actualiseren. De GPS-antenne moet zo ver mogelijk van de TFT-displayconnector verwijderd blijven (minstens enkele centimeters).

Enkele opmerkingen over de GPS-module

De GPS-module heeft een kleine back-up batterij voor data. Als de GPS een paar dagen achter elkaar niet wordt



Figuur 2: Het prototype op gaatjesprint. De afmetingen van de behuizing zijn 12 x 6,5 x 4 cm.

gebruikt, raakt de batterij leeg en gaan de gegevens verloren. Wanneer je de GPS weer inschakelt, kan het zijn dat je enkele minuten moet wachten om de gegevens te herstellen (en de batterij op te laden). De LED van de GPS-module knippert zodra de satellietgegevens met succes zijn gedecodeerd. Even later wordt het aantal satellieten weergegeven.

De GPS-antenne moet de satellieten 'zien'. De ontvangst van de signalen moet daarom zo min mogelijk worden belemmerd. In stedelijke gebieden, tussen twee gebouwen, kan de ontvangst moeilijk zijn. Desalniettemin werkt de snelheidsmeter in mijn auto naar behoren, ook als die in de voetruimte van de passagier is geplaatst.

Voorbereiding van de snelheidslimieten

Dit mag alleen worden gedaan als het voertuig stilstaat! Bij de eerste keer opstarten van de snelheidsmonitor zijn de snelheidslimieten allemaal ingesteld op 255. Dat komt overeen met FF hexadecimaal, en dat is wat er in een lege EEPROM staat.

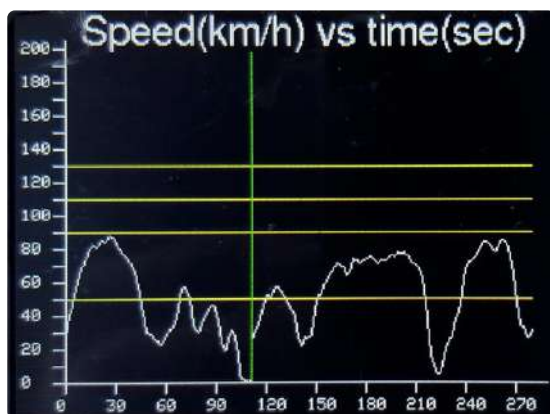


Figuur 3. Scherm 1 toont de snelheidslimieten als ronde verkeersborden met een korte tekst.



Figuur 4. In scherm 2 wordt de werkelijke snelheid weergegeven via een gesimuleerd 7-segment display.

Figuur 5. De curve van de snelheidsgeschiedenis wordt weergegeven in scherm 5. De vier gele horizontale lijnen geven de vier geprogrammeerde limieten weer en de groene verticale lijn geeft de huidige snelheid weer.



In scherm 3 (figuur 5), dat de gegevens van de satellieten toont, druk je op de derde knop om de instelling van de snelheidslimieten te activeren. Gebruik vervolgens de eerste en tweede knop om de gewenste limiet in te stellen. Druk nogmaals op de derde knop om de snelheidslimiet te valideren en door te gaan naar de volgende – en zo vervolgens. Zorg ervoor dat de limieten in een logische volgorde staan, aangezien de software ze niet sorteert.

Gebruik van de snelheidsmonitor

Het gebruik van de snelheidsmonitor is niet moeilijk. Aangezien hij bedoeld is voor gebruik in een rijdend voertuig, moet je ervoor zorgen dat het apparaat en vooral de voedingskabel niet in de weg zit tijdens het rijden. Zodra de monitor in een geschikte positie is geplaatst, selecteer je de gewenste weergavemodus. Scherm 1 (figuur 3) toont de bekende ronde verkeersborden, scherm 2 geeft de momentane snelheid weer met 7-segment cijfers (figuur 4). Selecteer vervolgens de snelheidslimiet die je wilt gebruiken. De zoemer klinkt als je te snel gaat. De passagier kan de snelheidscurve over een periode van 4,5 minuut (270 seconden) bekijken door scherm 5 te selecteren (figuur 5). In deze modus klinkt de zoemer niet. De horizontale lijnen geven de vier maximumsnelheden weer. De verticale lijn die 'nu' aangeeft, toont de huidige snelheid. Een scrollende grafiek was misschien mooier geweest, maar daarvoor moet het scherm voor elk nieuw gegevenspunt compleet worden ververs, en dat is traag. Het verplaatsen van de cursor is veel eenvoudiger.

Figuur 6. GPS-gegevens zijn te zien in scherm 3. Dit is ook het scherm dat toegang geeft tot de snelheidslimiet-presets in scherm 4.



Figuur 7. In scherm 4 kun je de snelheidslimieten instellen. Hier wordt de eerste grenswaarde ('Sp. 1') ingesteld (in het rood, als je goed kijkt).



GPS-gegevensscherm

Scherf 3 (figuur 6) toont de GPS-gegevens en stelt je in staat je locatie te vinden op een kaart waarop coördinaten zijn aangegeven. De nauwkeurigheid is ongeveer 30 meter, wat best goed is voor ons kleine en goedkope apparaatje. In deze weergavemodus klinkt de zoemer niet. Scherm 4 (figuur 7) is hetzelfde als scherm 3, behalve dat je hier de snelheidslimieten kunt instellen.

Tabel 1. De verschillende schermen en hoe je er doorheen navigeert.

	Knop 1	Knop 2	Knop 3	Knop 4	Knop 5
Scherf 1 (ronde verkeersborden)	Limiet 1	Limiet 2	Limiet 3	Limiet 4	Ga naar scherm 2
Scherf 2 (7-segment display)	Limiet 1	Limiet 2	Limiet 3	Limiet 4	Ga naar scherm 3
Scherf 3 (GPS-gegevens)	Limiet 1	Limiet 2	Ga naar scherm 4	Geen functie	Ga naar scherm 1
Scherf 4 (stel limieten in)	Verlaag limiet (-5 km/u)	Verhoog limiet (+5 km/u)	Bevestig en naar volgende limiet, terug naar scherm 3 als je klaar bent	Ga naar scherm 5	Geen functie
Scherf 5 (snelheidscurve)	Geen functie	Geen functie	Geen functie	Ga naar scherm 2	Geen functie

Merk op dat de GPS-datum en -tijd UTC-waarden zijn, dus betrokken op de meridiaan van Greenwich. Als je ze wilt gebruiken, moet je deze waarden omrekenen naar je lokale tijdzone (en eventueel zomer- en wintertijd). Aangezien de snelheidsmonitor geen klok is, is hiervoor niet in een comfortabel menu voorzien.

De software

Het programma kan eenvoudig worden aangepast; de broncode kan worden gedownload van [3]. Het is een Arduino-sketch die het STM32 Boards-package voor Arduino nodig heeft (wij gebruikten v2.3.0) [4]. Het leeuwendeel van de code betreft de grafische gebruikersinterface en de rest zorgt voor de eenvoudige taak van het ontvangen van GPS-gegevens, het extraheren van de snelheid en het vergelijken met een grenswaarde. Zoals gebruikelijk in een Arduino-sketch zorgt de functie `setup()` voor het initialiseren van alle periferie. Als dat gebeurd is, wordt gedurende vijf seconden een welkomstscherm weergegeven.

De `loop()`-functie begint met het inlezen van de drukknoppen voordat de GPS wordt gecontroleerd op nieuwe gegevens (in de `dataDecode()`-functie). Vervolgens worden, afhankelijk van welk scherm actief is, de bijbehorende gegevens weergegeven. Als de momentele snelheid hoger is dan de geselecteerde snelheidslimiet, klinkt een alarm.

Merk op dat debug- en statusinformatie wordt verzonden via de seriële poort (115200,N,8,1).

Aanpassingen

Hier zijn een paar dingen die je misschien wilt toevoegen of veranderen:

- Andere eenheden voor de snelheid. Met de GPS--bibliotheek kun je andere eenheden gebruiken, zoals knopen of mijlen/uur (mph). Hiervoor moeten ook de schermen worden aangepast (bijvoorbeeld km/u vervangen door knopen) en de stapgrootte voor het verhogen van de limieten (in de `SpeedSettings()`-functie). Dit kan handig zijn voor een boot.

Als je BluePill-module 128 KB flash bezit, kunnen extra functies worden toegevoegd zoals:

- Opname op SD-kaart. Aan de achterkant van de TFT-displaymodule bevindt zich een SD-kaartsleuf. Deze kan worden gebruikt om bijvoorbeeld de GPS-gegevens op te slaan voor later gebruik op een PC.
- Voeg een aanraakscherm toe voor nieuwe functies. Zorg in dat geval voor de veiligheid van de gebruiker.
- Mogelijkheid om de lokale tijd in te stellen. 

200220-03

Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de redactie van Elektor via redactie@elektor.com.



Gerelateerde producten

- **OPEN-SMART GPS – Serial GPS Module for Arduino (SKU 18733)**
www.elektor.nl/18733
- **2.2" SPI TFT Display Module ILI9341 (240x320) (SKU 18419)**
www.elektor.nl/18419
- **Majid Pakdel, Advanced Programming with STM32 Microcontrollers (SKU 19520)**
www.elektor.nl/19520

WEBLINKS

- [1] TinyGPS++ bibliotheek: <http://arduinoiana.org/libraries/tinygpsplus/>
- [2] What exactly is HDOP?: [https://en.wikipedia.org/wiki/Dilution_of_precision_\(navigation\)](https://en.wikipedia.org/wiki/Dilution_of_precision_(navigation))
- [3] Projectbestanden op Elektor Labs:
<https://elektormagazine.com/labs/save-money-with-this-speed-monitoring-by-gps>
- [4] STM32 Boards-package voor Arduino:
https://github.com/stm32duino/BoardManagerFiles/raw/main/package_stm32microelectronics_index.json

RGB-stroboscoop met Arduino

kleurige toepassing van een nuttig instrument

Roel Arits (Nederland)

In vroeger dagen, voordat moderne microcontroller-elektronica zijn intrede onder de motorkap deed, gebruikte men een stroboscoop om de ontsteking van de motor af te stellen. Nu is dat zelden meer nodig – bij een niet al te antieke auto wordt de ontsteking door de ECU (Electronic Control Unit) geregeld onder gebruikmaking van tal van sensoren.

Vroeger was alles anders – niet noodzakelijkerwijs beter, maar toch. In een nog niet eens zo grijs verleden hadden we onder de motorkap van een auto ruimte genoeg om daar zelf te kunnen knutselen, repareren en afstellen. Dankzij moderne computerondersteunde ontwerpstechnieken is het mogelijk alle componenten onder de motorkap zo compact en dicht op elkaar te monteren dat zelfs het vervangen van een ordinaire lamp een hels 'genoegen' is. En natuurlijk heeft de computer zelf (in de vorm van een groot aantal onderling verbonden microcontrollers) ook een plaatsje onder diezelfde motorkap gevonden.

Maar terug naar – laten we zeggen – de jaren '70 of '80 van de vorige eeuw. Voor het goede functioneren van een verbrandingsmotor (benzine-stoker) moeten de bougies van elke cilinder op precies het juiste moment een vonk produceren. Daarvoor zorgde de verdeler (een mechanische contraptie die bij bepaalde merken akelig vochtgevoelig was), die dan natuurlijk precies met de motor gesynchroniseerd moest worden. En daar komt dan eindelijk onze stroboscoop om de hoek kijken – of afstellamp zoals die toen ook wel genoemd werd. In zo'n verdeler bevonden zich evenveel contacten (de onderbrekercontacten) als de motor cilinders rijk was; telkens wanneer zo'n onderbrekercontact geopend werd (vandaar de naam) ontladde de in de bobine (een spoel) opgeslagen elektrische energie zich via de bougie in een vonk die het benzine/lucht-mengsel deed ontbranden. En dan ging het vooruit...

De stroboscoop die voor dit afstellen gebruikt werd, had een xenon- of neon-flitsbuis en een triggerkabel. Die werd op de referentie-bougiekabel aangesloten (met een inductieve opnemer); telkens als de referentie-bougie vonkte, werd de flitsbuis ontstoken en leverde deze een heel korte lichtflits. Op de krukasriemschijf was een wit merkteken (streepje) aangebracht; en op een soort hoekijzer daarboven was een tweede (vast) merkteken aangebracht. Wanneer de stroboscoop door de referentiebougie werd getriggerd moesten beide merktekens ten gevolge van het stroboscoop-effect precies tegenover elkaar stil lijken te staan. Als de streep op de bewegende krukasschijf niet tegenover maar links of rechts van de vaste streep stond, was dat een teken dat de verdeler bijgesteld moest worden. In de boven beschreven toepassing werd de stroboscoop gesynchroniseerd met de rotatie die we willen meten door een triggersignaal. Maar het kan ook zonder zo'n triggersignaal!

Triggeren? Nergens voor nodig...

Wanneer we met een niet-gesynchroniseerde stroboscoop een toerental willen meten, moeten we de frequentie van de lichtflitsen zodanig bijregelen dat het merkteken op de schijf (nagenoeg) stil lijkt te staan, en er maar één merkje zichtbaar is. Wanneer er meerdere (nagenoeg) stilstaande merkjes te zien zijn, dan wil dat zeggen dat de flitsfrequentie een veelvoud is van het toerental of dat het toerental een veelvoud is van de flitsfrequentie.

Als het wiel rechtsom draait en het merkje langzaam rechtsom lijkt te bewegen, dan is de flitsfrequentie iets te laag. De flits komt te laat zodat het merkje in de draairichting lijkt te bewegen. En als het wiel rechtsom draait en het merkje langzaam linksom lijkt te bewegen, dan is de flitsfrequentie iets te hoog: elke flits komt iets te vroeg.

Op deze manier kan het toerental worden gemeten door de flitsfrequentie van de stroboscoop zodanig bij te regelen dat er één stilstaand merktekentje duidelijk zichtbaar is. De duur van elke flits moet kort genoeg zijn om een duidelijke reflectie te verkrijgen (dus een duidelijk zichtbaar merkteken); als de flitsduur te lang is ten opzichte van de flitsfrequentie wordt de reflectie 'uitgesmeerd'.

Nu wordt het leuk

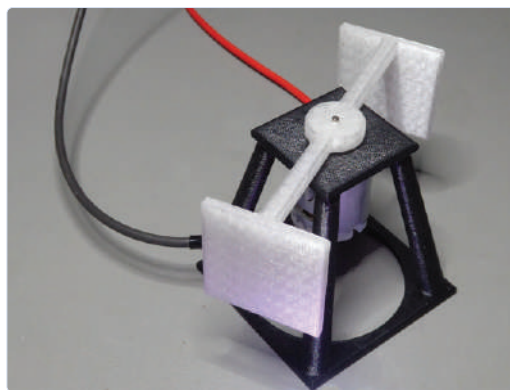
Wat gebeurt er wanneer we meerdere flitsen met dezelfde frequentie genereren, maar dan met verschillende kleuren en met een faseverschuiving tussen de afzonderlijke flitsen? En als we dan ook nog een roterend wit object gebruiken en frequentie, faseverschuiving en breedte van de verschillend gekleurde flitsen variëren?

Zoiets is niet moeilijk te realiseren met behulp van een microcontroller, dus niets weerhoudt ons om aan de slag te gaan. Voor de verschillend gekleurde flitsen kunnen we een RGB-LED gebruiken – beter nog meteen drie stuks om helderder flitsen te verkrijgen. Voor het regelen van frequentie, faseverschuiving en flitsduur gebruiken we een Arduino Pro Mini omdat die genoeg I/O heeft om mee te spelen, en voor ons doel meer dan snel genoeg is. Die LED's worden – hoe kan het ook anders – met pulsbreedte-gemoduleerde signalen (PWM) aangestuurd. We hadden natuurlijk drie speciale PWM-modules van de Arduino Pro Mini kunnen gebruiken, maar deze modules gebruiken drie verschillende timers. Dat maakt het moeilijker om ze te synchroniseren of faseverschuivingen te programmeren. Bovendien hebben we relatief laagfrequente PWM-signalen nodig en een resolutie van 16 bit is overdreven veel. Voor ons doel zijn softwarematig gegenereerde PWM-signalen (softPWM) meer dan goed genoeg. Met softPWM worden de signalen op normale digitale uitgangen uitgevoerd; we zetten en resetten deze aan de hand van een teller; telkens wanneer deze een bepaalde stand bereikt wordt een interrupt gegenereerd. Op deze manier wordt een interval ingesteld dat kort genoeg is om de pulsbreedte of faseverschuiving van het PWM-signaal met voldoende nauwkeurigheid bij te regelen.

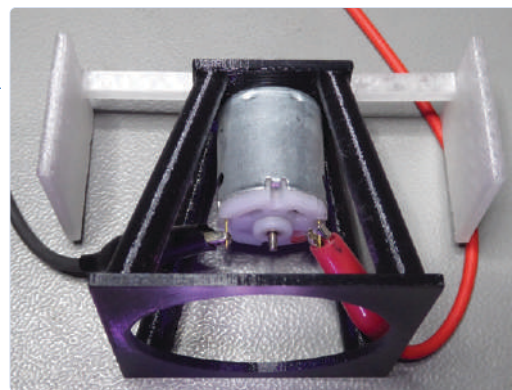
De praktijk

Om het bovenstaande te realiseren, heeft de auteur een 12V-gelijkstroommotortje gebruikt (Velleman MOT3N) en daarvoor een klein frame ontworpen. Dat motortje brengt een 'propeller' met twee verticale vlakken in beweging; die vlakken dienen als reflector. Beide delen zijn met een 3D-printer vervaardigd (de betreffende bestanden maken deel uit van de download bij dit artikel). In **figuur 1** en **figuur 2** ziet u wat de bedoeling is.

Let op: het door de auteur gebruikte motortje draait (onbelast) met zo'n 11500 rpm; dat is veel te veel voor de 3D-geprinte propeller. Bij een eerste experiment vlogen de auteur letterlijk de stukken om de



Figuur 1. Het frame met de propeller die als reflector dienst doet.



Figuur 2. Close-up van de motor.

oren. Dat is niet geheel ongevaarlijk; het is daarom geen gek idee om een veiligheidsbril te dragen. U bent gewaarschuwd!

De motor wordt daarom aangestuurd met een gelijkspanning van ongeveer 3 V; deze spanning is regelbaar om de motorsnelheid met de flitsfrequentie te kunnen synchroniseren. Met 3 V bedraagt het toerental ongeveer 2100 rpm, corresponderend met 35 Hz. Omdat de propeller symmetrisch is geconstrueerd met twee reflecterende vlakken (dat maakt het uitbalanceren wat eenvoudiger), bedraagt de flitsfrequentie het dubbele, dus 70 Hz (de periodeduur bedraagt dan ongeveer 14 ms). Maar nu eerst de elektronica: het schema is getekend in **figuur 3** en is eigenlijk de eenvoud zelf. De drie RGB-LED's worden via bescheiden drivertransistoren op commando van de Arduino Pro Mini in- en uitgeschakeld. De voedingsspanning voor het geheel bedraagt 5 V_{DC} bij een stroomopname van ongeveer 500 mA.

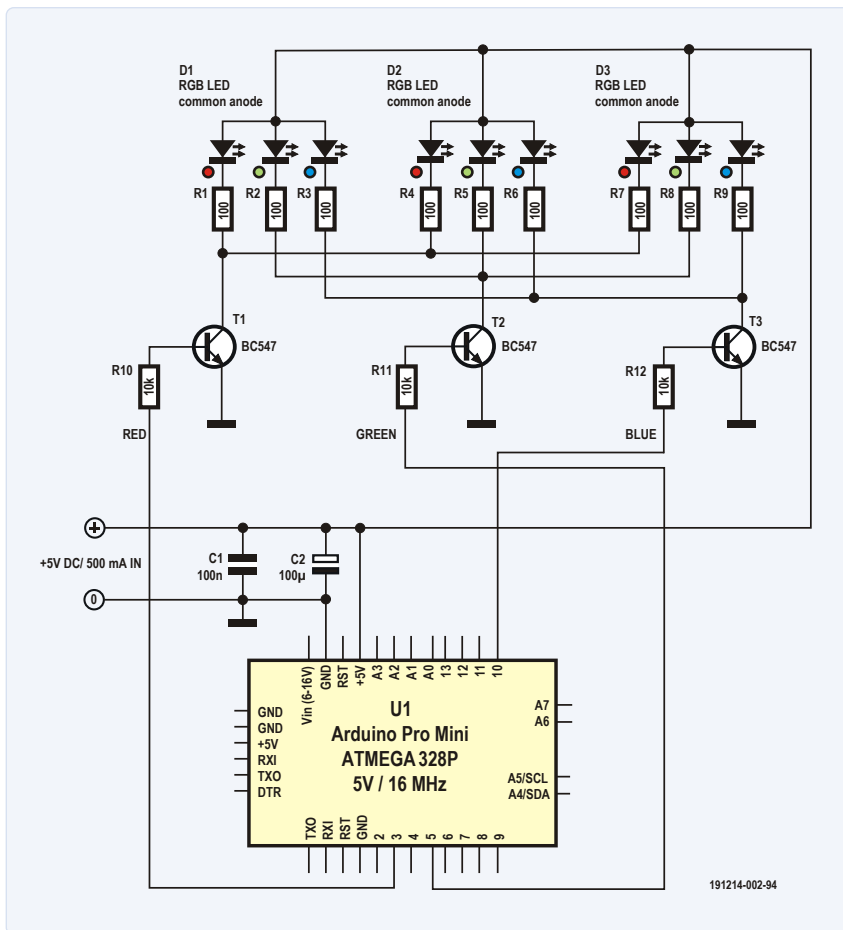
Voor de schakeling is geen print ontworpen, op breadboard zit de boel in een vloek en een zucht in elkaar (**figuur 4** toont de opstelling van de auteur).

De timing van de stroboscoop wordt geregeld op basis van een timer-interrupt die elke 100 µs wordt aangeroepen. Daarmee is dit interval van 0,1 ms de resolutie waarmee de flitsfrequentie kan worden gevarieerd; dit biedt voldoende regelmogelijkheden om het 'stilstaande' beeld van de propeller te verschuiven.

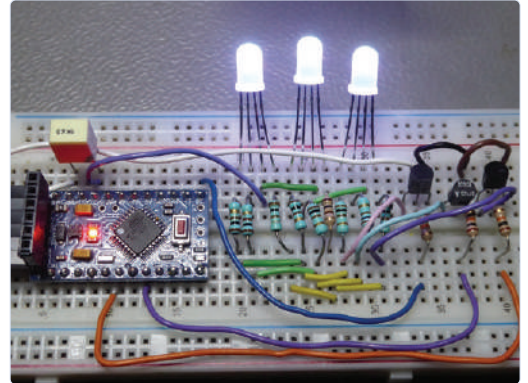
```
...
// Setup 16bit timer1 in normal operation with
// interrupt at 100us
// 16MHz/1 = 6.25ns. So to get 100us we need to let
// the timer count 1600 ticks.
TCCR1A = 0;
TCCR1B = _BV(CS10); //prescaler divide by 1
TCNT1 = 0xFFFF - 1600; // overflow is at 65535 =
// 0xFFFF
TIMSK1 = _BV(TOIE1); // overflow interrupt
TCNT1 = 0;
sei();
}
ISR (TIMER1_OVF_vect)
{
TCNT1 = 0xFFFF - 1600; //100us interrupt
...

```

De flitsperiode is 'hard' in de software gecodeerd en is ingesteld op 144 – dat is het aantal timer-interrupts tussen twee opeenvolgende flitsen; deze periode bedraagt dus $144 \cdot 100 \mu s = 14,4 \text{ ms}$. De flitsduur voor elke LED is vast ingesteld op 8, overeenkomend met



Figuur 3. Het schema is niet bijzonder ingewikkeld.



Figuur 4. De opbouw is niet kritisch; voor eerste experimenten is een breadboard prima geschikt.

$8 \cdot 100 \mu s = 800 \mu s$. De duty cycle (puls/pauze-verhouding) bedraagt zodoende $800 \mu s / 14,4 ms = 5,5\%$.

```
#define STROBE_PERIOD 144
```

```
TSoftPwm Pwm[] = {TSoftPwm(ID_RED, 0, 8,
    STROBE_PERIOD),
    TSoftPwm(ID_GREEN, 0, 8, STROBE_PERIOD),
    TSoftPwm(ID_BLUE, 0, 8, STROBE_PERIOD)};
```

Met deze duty cycle krijgen we (zodra het toerental van de motor optimaal is ingesteld) een mooi 'strak' beeld. De schakeling plus software (die gratis kan worden gedownload van de projectpagina bij dit artikel [4]) is geen 'afgewerkt' project. De auteur bedoelde het veeleer als 'proof of concept' en heeft die daarom niet voorzien van een fraaie gebruikers-interface en dergelijke. De software die beschikbaar is, demonstreert de mogelijkheden door met de parameters flitsfrequentie, flitsduur en faseverschil te spelen. Omdat de software van veel en duidelijk commentaar is voorzien, gaan we er hier niet verder op in. Natuurlijk is een niet-gesynchroniseerde stroboscoop niet echt optimaal. In principe zou het niet zo moeilijk moeten zijn om bij de propeller

een lichtsluisje te monteren waarvan het signaal als trigger voor de stroboscoop kan dienen. Dit laten we echter met een gerust hart aan de geïnteresseerde lezer over.

De auteur heeft twee video's op Youtube gezet waar u de werking van de stroboscoop kunt bewonderen [1] en de onderlinge relatie van de LED-stuursignalen op het scherm van een oscilloscoop kunt bekijken [2]. Het project is ook op Elektor Labs te vinden [3].

191214-01



Related Products

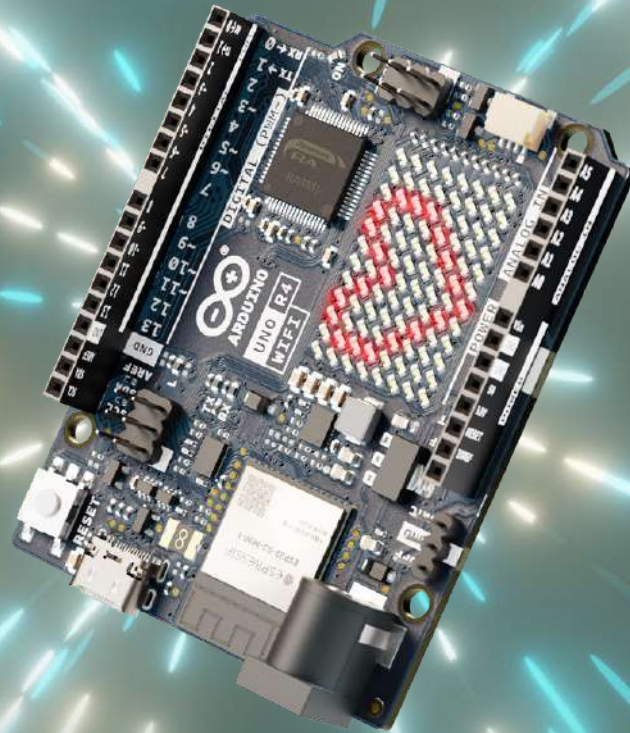
- > **50 Mini microcontroller projecten met ATtiny en Arduino**
www.elektor.nl/
50-mini-microcontroller-projecten-met-attiny-en-arduino
- > **Arduino - projecten voor gevorderden**
www.elektor.nl/arduino-nl
- > **Home Automation Projects with Arduino**
www.elektor.nl/home-automation-projects-with-arduino

WEBLINKS

- [1] Demo-video 1: <https://youtu.be/WHv6DCWM82k>
- [2] Demo-video 2: <https://youtu.be/3tYqUin4-vQ>
- [3] Projectpagina op Elektor Labs: www.elektormagazine.com/labs/arduino-rgb-color-stroboscope
- [4] Projectpagina bij dit artikel: www.elektormagazine.nl/191214-01



UNO R4



The new dimension of making

Take a quantum leap with the new UNO R4, the most advanced and powerful UNO ever!



Faster and larger memory - 48 MHz, 256 kB Flash, 32 kB RAM - perfect to perform more precise calculations and handle more complex projects.



New built-in peripherals - The RA4M1 (Arm® Cortex®-M4 Core with FPU) includes a DAC, CAN-BUS, HID support and OpAMP embedded allowing for more advanced projects.



Expand your projects with UNO shields (5 V compatible) and the huge catalog of *Qwiic nodes.



Connectivity *ESP32-S3 coprocessor - Wi-Fi® and Bluetooth® Low Energy connectivity, leaving the RA4M1 microcontroller free.



Robust by design - with a larger voltage range, 6-24 V, now you can use the UNO R4 in combination with motors, LED strips or other actuators from a single power supply.

UNO R4 WiFi



UNO R4 Minima



*Note - these features are only available on the UNO R4 WiFi model.



Draadloze nooddrukknop

meer veiligheid met LoRa

Somnath Bera (India)

De draadloze nooddrukknop die in dit artikel wordt beschreven, is een voorbeeld van hoe een probleem in een complexe industriële omgeving kan worden opgelost met een paar goedkope elektronische modules en wat Arduino-programmering. Het resultaat is een eenvoudig draadloos systeem dat ook in veel andere situaties van pas kan komen.



De oorspronkelijke reden voor de bouw van de hier gepresenteerde draadloze nooddrukknop (EPB, *emergency push button*) is nogal specifiek en heeft te maken met het nemen van monsters van treinen die kolen leveren aan een elektriciteitscentrale. Er moet van boven een steenkoolmonster van een kolenwagen worden genomen om de calorische waarde ervan te bepalen voordat het mag worden gebruikt. Dit is een belangrijke parameter voor een elektriciteitscentrale.

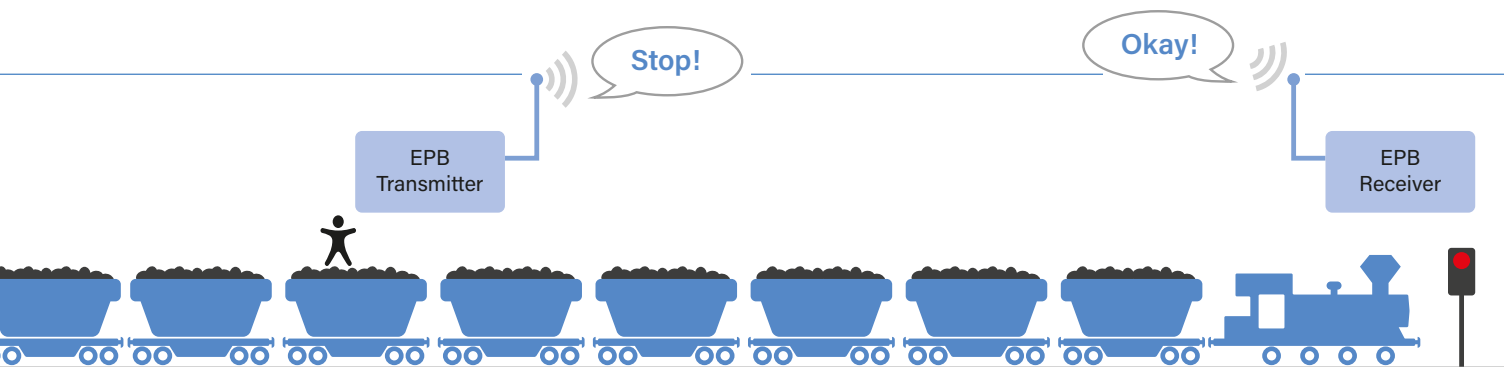
Alvorens een monster te nemen, drukt de betreffende werknemer op de (bestaande) EPB om de machinist van de trein met een sein van zijn aanwezigheid op de hoogte te brengen. Dit zou de trein moeten immobiliseren. Helaas kan de 'monsternemer', afhankelijk van de lengte van de trein, de positie van de wagon en de kromming van het spoor, het betreffende sein niet altijd zien. Bij het verzamelen van monsters is het af en toe voorgekomen dat de trein toch nog doorrolde, waardoor ongelukken

werden veroorzaakt en mensen gewond raakten. Het EPB-systeem was dus voor verbetering vatbaar.

Dingen om op te letten

Die kolentreinen kunnen erg lang zijn, tot 500 meter, en er kunnen er wel vier naast elkaar staan die tegelijkertijd worden bemonsterd en gelost. Voeg daar nog die gekromde spoorbaan aan toe, en je begrijpt dat het lastig is om te zien wat er overal gebeurt. Verder is de omgeving lawaaierig door lossende treinen. Hoorbare feedback is dus ook problematisch.

Het bestaande EPB-systeem is een bekabeld systeem. In een omgeving zoals hierboven beschreven is het echter riskant om lange kabels te gebruiken, omdat ze kapot kunnen gaan zonder dat iemand het merkt. Het vervangen van die kabels, of het verleggen van meerdere kabels ten behoeve van visuele feedback is ingewikkeld en duur. Het hoger maken van de seinmast zodat deze van grote



Figuur 1. Overzicht van het draadloze nooddrukknoopstelsel (EPB).

afstand zichtbaar is, is ook geen praktische oplossing omdat je goede ogen moet hebben om te zien voor welk spoor het sein geldt.

Draadloos met LoRa

De oplossing die we hebben verzonnen, is de draadloze EPB (**figuur 1**). Deze bestaat uit twee eenheden: een EPB-zender en een EPB-ontvanger. Eigenlijk zijn die units bijna identiek en kunnen ze beide zenden en ontvangen, maar we zullen er op deze manier naar verwijzen. Desgewenst kun je het beschouwen als een master/slave- of client/server-systeem.

Alvorens een kolenmonster te nemen, drukt de bemonsteraar op een knop op de EPB-zender om zijn wens om een monster te nemen aan te geven. Wanneer de EPB-ontvanger in de controlecabine dit verzoek ontvangt, activeert deze een relais om de knop van het bestaande EPB-systeem te activeren, en stuurt een bevestiging terug naar de EPB-zenderunit. Nu weet de werknemer dat zijn signaal goed is aangekomen en kan hij veilig een monster nemen. Als hij klaar is, drukt hij op een tweede knop op de EPB-zender om het systeem weer vrij te geven. Dan geeft de EPB-ontvanger de bestaande EPB vrij door een tweede relais te activeren en stuurt hij opnieuw een bevestiging naar de EPB-zender. Het systeem is daarmee klaar voor een nieuw monster.

In het geval dat het systeem niet beschikbaar is, of het EPB-sein voor operationele doeleinden overbrugd wordt, zal de zender nooit feedback krijgen, zodat de bemonsteraar gewaarschuwd is en misverstanden worden voorkomen.

EPB-zender en -ontvanger: de schakelingen

De EPB-zender is een exact gelijk aan de bestaande EPB-unit, maar dan voorzien van een kleine antenne en een dito OLED-display. Het wordt gevoed door een enkele LiPo-cel. Het heeft twee activerings-drukknoppen. Erin zit is een ESP32-module die het OLED-display en een LoRa-transceivermodule aanstuurt (**figuur 2**). Merk op dat de ESP32-module alleen voor het gemak werd gebruikt, niet vanwege de draadloze mogelijkheden. Elke andere goedkope, energiezuinige microcontrollermodule met de juiste interfaces (I²C, UART, tweemaal GPIO) kan in plaats daarvan worden gebruikt.

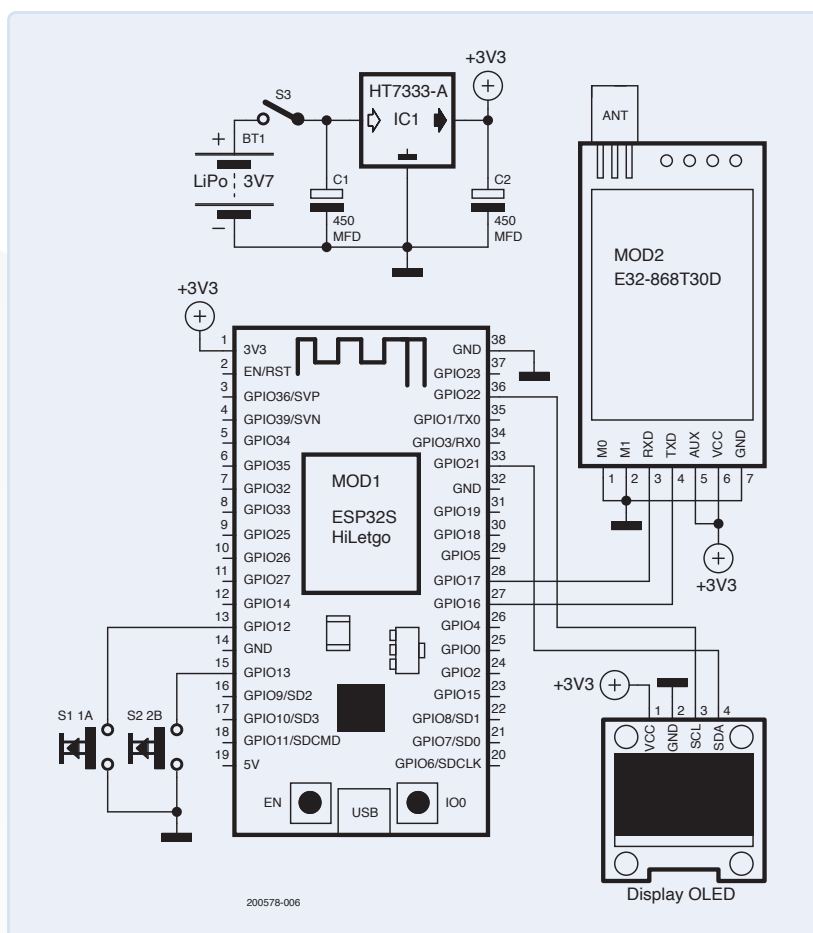
De EPB-ontvanger is vergelijkbaar met de zender, behalve

dat hij twee relais heeft in plaats van twee drukknoppen, en geen display (**figuur 3**). Merk op dat op elk moment slechts één van beide relais actief mag zijn. Daar zorgt de software voor.

Beveiliging

Om het systeem zodanig te beveiligen dat het nooit gestoord kan worden door andere signalen, hebben we gekozen voor low-power LoRa-transceivermodules van Ebyte. Naast het gebruik van storingsbestendige spread-spectrumtechnologie bieden ze nog drie parameters: kanaalfrequentie, air rate en een 4-byte ID. Communicatie is alleen mogelijk als deze drie parameters aan beide zijden van de verbinding gelijk zijn.

Figuur 2. De EPB-zender bestaat uit een OLED-display en twee drukknoppen (naast de LoRa-transceiver).



Alle begin...

...volgt de emitter

Eric Bogers (Elektor)

De laatste keer hebben we heel kort stilgestaan bij de versterkende transistor – in tegenstelling tot de schakelende transistor. Als eerste bekijken we nu de zogenaamde *emittervolger* – zo ongeveer de eenvoudigste transistorschakeling die denkbaar is.

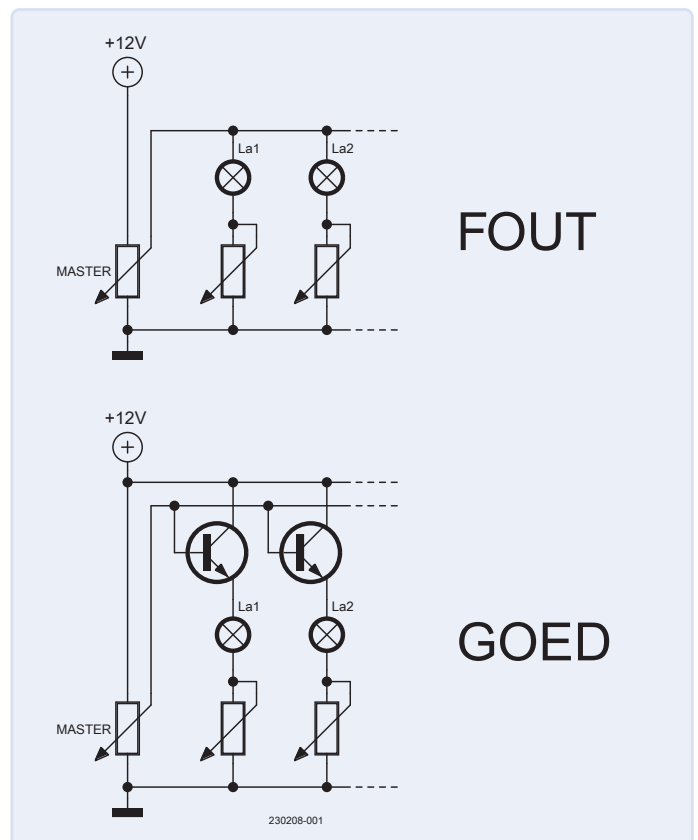
De collectorschakeling

De eerste versterkerschakeling die we zullen bekijken, is de zogenaamde (gemeenschappelijke) collectorschakeling. Deze schakeling wordt zo genoemd omdat de collector van de transistor het gemeenschappelijke referentiepunt is voor in- en uitgangssignaal. Deze benaming is eigenlijk niet bijzonder aanschouwelijk; de schakeling is beter bekend onder de naam *emittervolger* – en dat beschrijft eigenlijk best wel goed wat de schakeling doet: de spanning op de emitter van de transistor ‘volgt’ de spanning op diens basis. Waar dat goed voor is, zullen we in deze aflevering zien. Stel je eens voor dat je een mengpaneel voor een lichtinstallatie wilt bouwen – niets professioneels, gewoon iets ‘voor tussen de schuifdeuren’ en op een lage spanning. Denk bijvoorbeeld aan een aantal lampen (‘kanalen’) die elk afzonderlijk in lichtsterkte geregeld kunnen worden (van 0 V tot 10 V in dit voorbeeld), maar die ook met een master-regelaar allemaal tegelijk gedimd kunnen worden. Niet gehinderd door enige kennis, zou je een master-potentiometer kunnen nemen, en aan de looper daarvan voor elke lamp een kanaal-potmeter vastknopen. We hebben dit geschetst in **figuur 1** (boven).

Dit is echter de verkeerde manier. Al die kanaalregelaars vormen samen een zware belasting die de karakteristiek van de master-regelaar behoorlijk uit het lood zou trekken; bovendien zullen alle potmeters elkaar beïnvloeden zodat er van een bruikbare regeling geen sprake kan zijn.

We moeten de belasting voor de master-regelaar zoveel mogelijk reduceren, en bovendien de afzonderlijke kanaalregelaars van elkaar ontkoppelen. En dat doen we met emittervolgers, zoals geschetst in **figuur 1** (onder). Dat gaat als volgt.

Zolang een transistor geleidt, is de spanning op diens emitter ongeveer 0,7 V lager dan de spanning op diens basis. Dat betekent dat het regelbereik van 0 V tot 11,3 V loopt – dat is iets meer dan de nominale 10 V van dit specifieke voorbeeld, maar het betekent wel dat als de regelaars helemaal ‘open’ worden gezet, de lampjes ook op volle sterkte branden. Als de spanning op de basis onder 0,7 V komt, spert de transistor en is de uitgangsspanning 0 V.



Figuur 1. Een (wel heel erg simpel) lichtmengpaneel – boven op de verkeerde manier, onder op de goede manier.

Tussen deze beide 'uiterste' toestanden volgt de spanning op de emitter braaf de spanning op de basis, maar dan met een *offset* ter waarde van de basis-emitterspanning. Het grote voordeel dat de emittervolger hier biedt is dat de basis slechts een zeer geringe belasting vormt voor de loper van de master-regelaar: de basisstroom is veel kleiner dan de emitterstroom. De verschilfactor is gelijk aan de stroomversterking van de transistor. Dat betekent in de praktijk dat de master-regelaar een belastingsweerstand 'ziet' die een factor h_{FE} kleiner is dan de werkelijke waarde van de kanaal-regelaar.

De collectorschakeling of emittervolger levert dus geen spanningsversterking, maar zorgt wel voor een forse stroomversterking, een hoge ingangsweerstand en een lage uitgangsweerstand. De emittervolger reageert uiterst stabiel op veranderingen in de belasting: als de belastingsweerstand kleiner wordt, dan zouden we 'eigenlijk' verwachten dat de uitgangsspanning afneemt. Maar daardoor neemt U_{BE} echter toe en neemt I_B ook toe. Dit op zijn beurt leidt tot een toename van I_C en dus ook van I_E waardoor de afname van de uitgangsspanning meteen wordt gecompenseerd.

Voor de ingangs- en uitgangsimpedantie van de emittervolger geldt:

$$Z_{in} = (h_{FE} + 1) \cdot Z_{load}$$

$$Z_{out} = \frac{Z_{source}}{h_{FE} + 1}$$

De gestabiliseerde netvoeding

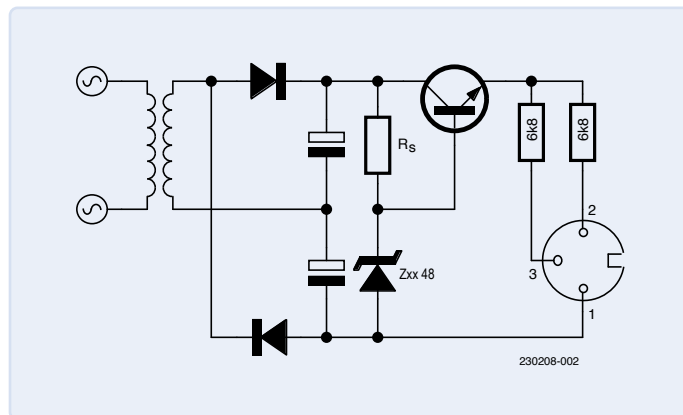
In een eerdere aflevering hebben we na het nodige rekenwerk ontdekt dat de stabilisatie van een spanning met uitsluitend een zenerdiode en een serieweerstand aanzienlijke nadelen met zich meebrengt. Zo'n stabilisatorschakeling fungeert veel en veel beter wanneer we daar ook een emittervolger inbouwen, zoals geschetst in **figuur 2**.

Laten we deze schakeling eens doorrekenen met de waarden uit het eerdere voorbeeld (een zenerspanning van 48 V, en een maximale belastingsstroom van 14,1 mA – het betreft een fantoomvoeding voor een microfoon). Voor de stroom in de basis van de transistor kunnen we dan schrijven:

$$I_B \approx \frac{I_E}{\beta} = \frac{14.1 \text{ mA}}{100} = 0.14 \text{ mA}$$

De stroomversterking geldt voor de collectorstroom; strikt genomen is de emitterstroom gelijk aan I_C plus I_B – maar in kleinsignaalbedrijf, bij een stroomversterking van meer dan 100, mogen we I_C en I_E in goede benadering aan elkaar gelijk stellen. We kennen de exacte waarde van de stroomversterking van de transistor niet; door bij kleinsignaaltransistoren een factor 100 aan te nemen, blijven we vooraleerst aan de veilige kant.

Gedurende een periode van de ingangswisselspanning neemt de spanning over de bufferelco tot ongeveer 60 V af; over de



Figuur 2. Gestabiliseerde netvoeding (fantomvoeding voor een microfoon).

serieweerstand van de zenerdiode valt zodoende 12 V. Dan geldt:

$$R_{series} = \frac{U}{I_B} = \frac{12 \text{ V}}{0.14 \text{ mA}} = 85 \text{ k}\Omega$$

Voor de serieweerstand nemen we de standaardwaarde 82 k Ω .

De dissipatie in de zenerdiode is maximaal wanneer er geen belasting is aangesloten en er dus geen stroom in de basis van de transistor loopt. Er geldt dan:

$$P_{zener} = U \cdot I = 48 \text{ V} \cdot \frac{63 \text{ V} - 48 \text{ V}}{82 \text{ k}\Omega} = 8.78 \text{ mW}$$

Natuurlijk wordt in de transistor ook vermogen gedissipeerd:

$$P_{transistor} = U \cdot I = (63 \text{ V} - 48 \text{ V}) \cdot 14.1 \text{ mA} = 211.5 \text{ mW}$$

De som van de dissipatie in de zenerdiode en die in de transistor is nu al kleiner dan die in de zenerdiode alleen wanneer de transistor wordt weggelaten (dat was namelijk iets meer dan 900 mW). De schakeling heeft echter nog een tweede voordeel: bij de schakeling zonder transistor doet de maximale dissipatie zich voor wanneer geen verbruiker is aangesloten, en de minimale dissipatie doet zich voor bij een belasting van 0 Ω (wat in de praktijk natuurlijk nooit voorkomt). In de schakeling met emittervolger daarentegen wordt het maximale vermogen gedissipeerd bij een belastingsweerstand van 0 Ω , terwijl onder 'normale' bedrijfsomstandigheden de dissipatie geringer is.

De emittervolger als wisselstroomversterker

Tot nog toe hebben we de emittervolger uitsluitend als gelijkstroomversterker gebruikt; nu willen we wisselstromen gaan versterken – zie **figuur 3**.

Wanneer via condensator C1 een wisselspanning wordt aangelegd, dan verandert deze de spanning op de basis (ten opzichte van massa), en de emitterstroom varieert dan zodanig dat de spanning op de emitter steeds ongeveer 0,7 V lager is dan die op de basis. Ook hier hebben we dus niet met spanningsversterking te maken maar met stroomversterking.

Voor een zo groot mogelijk uitsturingsbereik van de schakeling moet de rustspanning van de emitter ongeveer gelijk zijn aan de halve voedingsspanning, in dit voorbeeld dus 6 V. Voor de goede orde: rustspanning en ruststroom zijn grootheden die worden gemeten wanneer er geen signaalspanning (wisselspanning) is aangelegd.

Als we voor de ruststroom een waarde van 1 mA aannemen, komen we voor de waarde van de emitterweerstand uit op:

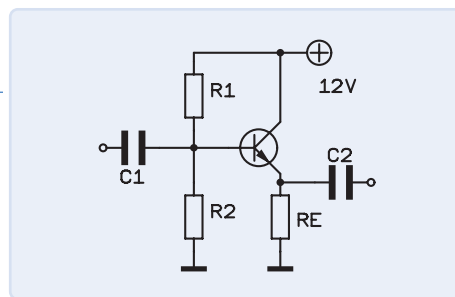
$$R_E = \frac{U_R}{I_R} = \frac{6V}{1mA} = 6k\Omega$$

Die waarde van 6 kΩ ligt tussen twee E12-waarden in. We kiezen 5,6 kΩ. De spanning op de basis van de transistor is 0,7 V hoger dan de emitterspanning, dus 6,7 V ten opzichte van massa. Deze spanning stellen we in door middel van een spanningsdeler. Voor R2 kiezen we arbitrair een waarde van 100 kΩ, en daarmee kunnen we R1 uitrekenen. We laten deze berekening met een gerust hart aan de lezer over; we komen uit op 79,1 kΩ en kiezen een waarde van 82 kΩ. Daarmee vallen basis- en emitterspanning ongeveer 0,1 V lager uit dan eigenlijk de bedoeling was, maar dat is geen enkel probleem.

Nu moeten we de beide condensatoren gaan dimensioneren. Omdat de schakeling over een frequentiebereik van 20 Hz...20 kHz een zo lineair mogelijke overdracht moet vertonen, wordt de onderste kantelfrequentie in beide gevallen bij 10 Hz gelegd. Condensator C1 vormt samen met de parallelschakeling van R1 en R2 en de impedantie van de transistor een hoogdoorlaatfilter; zoals we inmiddels weten hangt de impedantie van de transistor van de belastingsweerstand af. Als we aannemen dat die belastingsweerstand niet kleiner zal zijn dan R_E, dan komt de parallelschakeling van die belastingsweerstand en R_E uit op 3 kΩ; bij een veronderstelde stroomversterking van 100 bedraagt de ingangsimpedantie van de transistor (R_{in}) dan ongeveer 300 kΩ. De parallelschakeling van R1, R2 en R_{in} komt dan uit op 39,2 kΩ (reken dat zelf na!). We dimensioneren C1 dan als volgt:

$$C_1 = \frac{1}{2 \cdot \pi \cdot f \cdot Z} = \frac{1}{2 \cdot 3.14 \cdot 10Hz \cdot 39.2k\Omega} = 0.4 \mu F$$

Met een waarde van 0,47 μF zitten we in elk geval aan de veilige kant.



Figuur 3. Wisselspanningsversterker.

Voor C2 rekenen we met een veronderstelde minimale belastingsimpedantie van 6 kΩ:

$$C_2 = \frac{1}{2 \cdot \pi \cdot f \cdot Z} = \frac{1}{2 \cdot 3.14 \cdot 10Hz \cdot 6k\Omega} = 2.6 \mu F$$

Hier zouden we dan een waarde van 3,3 μF kiezen (of 4,7 μF als die makkelijker verkrijgbaar is).

Nog een opmerking ten behoeve van de Pietjes Precies onder de lezers: bij de berekening van R1 hebben we geen rekening gehouden met het feit dat de gelijkstroomimpedantie van de transistor (die gelijk is aan het product van de stroomversterking en R_E) eigenlijk parallel aan R2 staat.

Als we dit netjes in de berekening willen opnemen, moeten we de stroomversterking van de transistor redelijk nauwkeurig kennen – als we daar ‘om aan de veilige kant te blijven’ een waarde van 100 aannemen, dan zou het werkpunt flink verschuiven; voor kleinsignaaltransistoren is een waarde van 300...500 realistischer. Maar in dat geval mogen we de invloed ervan op de berekening van R1 verwaarlozen...

Tot zover voor deze aflevering; volgende keer wordt het echt spannend want dan gaan we spanningen versterken met de emitter-schakeling (en dat is iets heel anders dan de emittervolger!).

230208-03

De artikelreeks “Alle begin...” is gebaseerd op het boek “Basiscursus elektronica” van Michael Ebner, dat bij Elektor is verschenen.

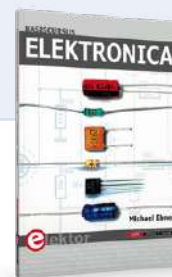
Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de redactie van Elektor via redactie@elektor.com.



Gerelateerde producten

> Michael Ebner, *Basiscursus elektronica* (e-boek, SKU 18232) www.elektor.nl/18232



Vrij te kiezen onafhankelijke hystereseniveaus voor comparatoren

met simulaties, spreadsheets en algebra

Stephen Bernhoeft (Verenigd Koninkrijk)

Bij het configureren van comparatoren willen we vaak beide hystereseniveaus nauwkeurig en onafhankelijk van elkaar kiezen. En dat is heel goed mogelijk. In dit artikel bespreken we hoe je dat voor elkaar krijgt.

In de beschikbare literatuur is meestal weinig te vinden over de keuze van de juiste weerstandswaarden voor de gewenste hystereseniveaus in comparatorschakelingen, behalve voor het geval waar de schakeldrempels op even ver van de voedingsrails moeten komen. In dit artikel leiden we met wat algebra enkele formules af voor populaire schakelingen aan zowel de inverterende als de niet-inverterende ingang. De spreadsheets, afleidingen, en *LTSpice*-schema's zijn beschikbaar als download [1].

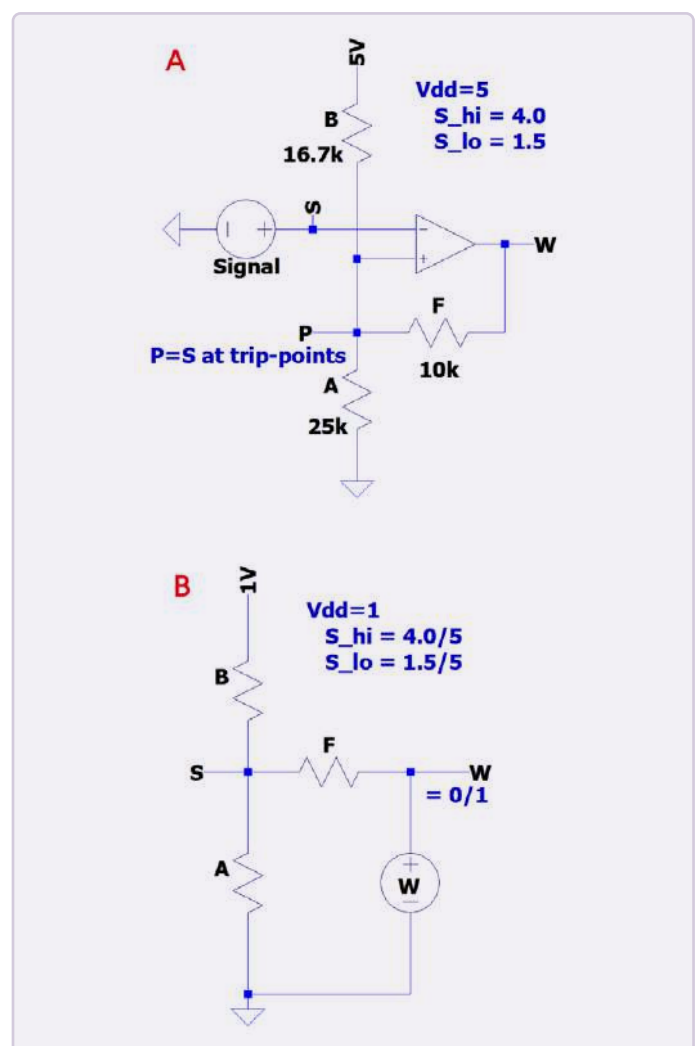
De resultaten zijn geldig voor:

- > comparatoren met rail-to-rail uitgangen
- > comparatoren met open-collector uitgangen en $R_{PULLUP} \ll R_{FEEDBACK}$

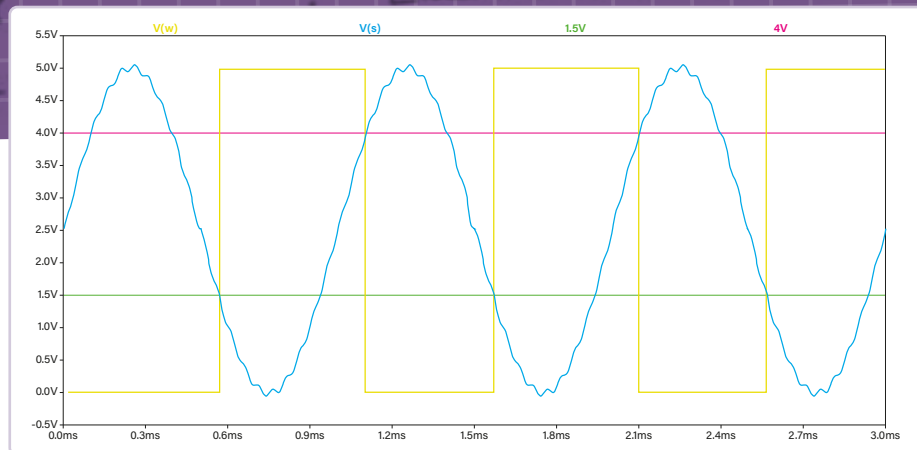
Voor een open-collector ontwerp kies je eerst R_{PULLUP} en maak je dan $R_{FEEDBACK}$ (aangeduid met 'F' in de rest van dit artikel) minstens $10 \times R_{PULLUP}$. De andere weerstanden worden berekend op basis van F.

Parameters

Kies de gewenste schakeldrempels en let op de uitgangsniveaus. De waarden mogen worden geschaald, maar voor open-collector schakelingen moet je op de pullup-waarde letten.



Figuur 1. Boven: de echte schakeling (A); onder: een vereenvoudigde versie voor onze analyse (B).



Figuur 2. De gesimuleerde signalen van schakeling (A) uit figuur 1.

Schema voor inverterende ingang

Zie voor dit geval **figuur 1**. De voordelen van deze configuratie zijn dat er geen aparte referentie nodig is, dat de ingangsimpedantie hoog is en dat weinig componenten nodig zijn. Het inverterende gedrag kan echter lastig zijn, zeker als je het aantal onderdelen wilt minimaliseren. In het vereenvoudigde schema in figuur 1b zien we dat F effectief parallel staat aan B (als $W = 1$) of A (als $W = 0$). De voedingsspanning is 1 V om het rekenwerk te vereenvoudigen. Natuurlijk kan de voedingsspanning worden geschaald. Als de voedingsspanning bijvoorbeeld 5 V is bij een drempelwaarde van 4 V, dan is dat te schalen naar een voedingsspanning van 1 V en een drempelwaarde van $4/5$ V, dus 0,8 V. W schakelt om als $P = S$, dus we kunnen aannemen dat $P = S$ bij beide schakeldrempels. In figuur 1a loopt knooppunt P mee met de uitgang van de comparator. We kunnen P herbenoemen naar S in het schema van figuur 1b omdat P dezelfde waarde heeft als S .

Als we nu definiëren:

- S_{LO} = lage schakeldrempel
- S_{HI} = hoge schakeldrempel

En we stellen:

$$k_{BF} = \frac{S_{HI} - S_{LO}}{S_{LO}} = \frac{B}{F}; \quad k_{AB} = \frac{S_{LO}}{1 - S_{HI}} = \frac{A}{B}$$

dan krijgen we:

geval 1: vaste A

$$B = \frac{A}{k_{AB}}; \quad F = \frac{B}{k_{BF}}$$

geval 2: vaste B

$$A = B \cdot k_{AB}; \quad F = \frac{B}{k_{BF}}$$

geval 3: vaste F

$$B = F \cdot k_{BF}; \quad A = B \cdot k_{AB}$$

We kennen nu de verhoudingen $A : B$ en $B : F$ voor de gekozen schakeldrempels. Als we één weerstand kiezen (A , B of F), kunnen

we de twee andere waarden berekenen. In **figuur 2** zie je de gesimuleerde signalen van schakeling (A).

Ontwerpoverwegingen

Bij de keuze van drempelwaarden spelen verschillende overwegingen een rol:

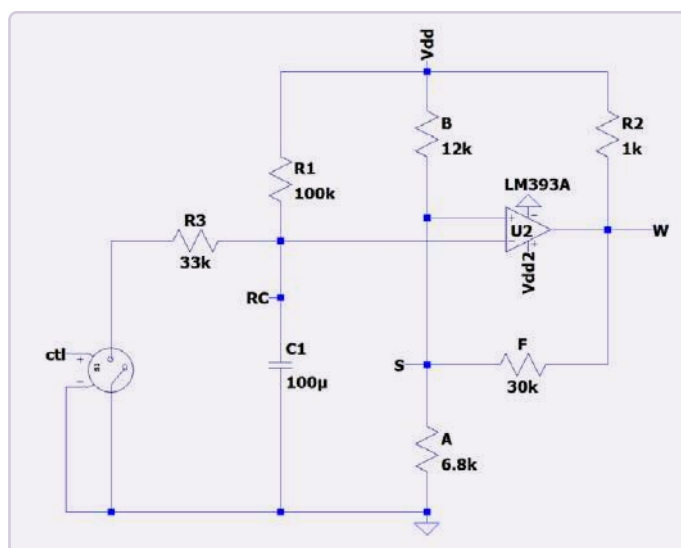
- voor RC-timingschakelingen: vermijd de langzame 'staart' van de RC-curve, want daar heeft een heel kleine verandering in spanning een grote invloed op de tijd;
- drempelwaarden moeten zo veel mogelijk onafhankelijk zijn van de offsetspanningen van de comparator. Daarom moet de laagste drempelspanning veel groter zijn dan de (absolute) waarde van die offsetspanning.

Met behulp van een spreadsheet [1] zien we dat bij een zeer lage waarde voor S_{LO} een groot verschil ontstaat tussen de vereiste weerstandswaarden voor A , B en F . Dat is duidelijk niet ideaal met het oog op de toleranties van de componenten. De beste relatieve nauwkeurigheid wordt bereikt als A , B en F niet te veel in waarde verschillen.

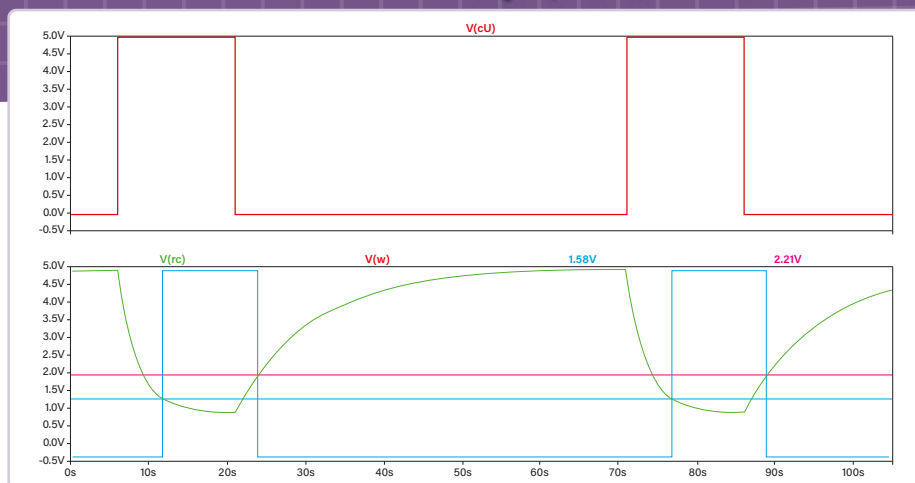
Toepassingen

Een praktijkvoorbeeld: de auteur was betrokken bij het ontwerp van een rookalarmstelsel. In het begin dachten we aan een analoog ontwerp, gebaseerd op een metaaloxidesensor (we durfden zelf geen software te schrijven voor zo'n veiligheidskritische toepassing).

Stel dat we 3 seconden rookdetectie willen voordat de alarmuitgang wordt geactiveerd, maar 6 seconden zonder rook voordat het alarm wordt gestopt. In **figuur 3** zie je een mogelijke schakeling en in **figuur 4** de gesimuleerde uitgangssignalen.



Figuur 3. De ingangstrap van een rookdetector. Als de schakelaar gesloten is, geldt $R_{EQ} = 25 \text{ k}\Omega$ en $V_{EQ} = 1,25 \text{ V}$. Als de schakelaar open is, geldt $R_{EQ} = 100 \text{ k}\Omega$ en $V_{EQ} = 5 \text{ V}$. $\Delta V = (5 - 1,25) = 3,75 \text{ V}$.



Figuur 4. De gesimuleerde uitgangssignalen van de schakeling uit figuur 3.

Als geen rook wordt gedetecteerd, is de schakelaar gesloten en de condensator laadt op tot een niveau van $5/4$ V door een effectieve weerstand ter waarde van $R_{EQ} = R1||R3$. Als wel rook wordt gedetecteerd, opent het signaal ct schakelaar S1, zodat het RC-circuit wordt opgeladen van 1,25 V naar $V_{DD} = 5$ V.

Een spreadsheet geeft de benodigde drempelwaarden, en een ander spreadsheet geeft de omzetting van die spanningen naar weerstandswaarden voor A , B en F .

Schema voor niet-inverterende ingang

We gaan nu kijken naar de configuratie voor een niet-inverterende ingang (zie figuur 5). In dit geval hebben we een ingangsimpedantie van $A+F$. Deze configuratie is wat lastiger te analyseren.

We schrijven:

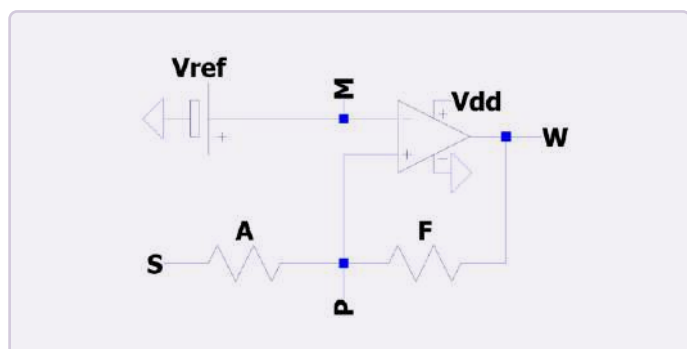
$$\Delta S = S_{HI} - S_{LO}; \quad \Delta W = W_{HI} - W_{LO}$$

$$\Sigma S = S_{HI} + S_{LO}; \quad \Sigma W = W_{HI} + W_{LO}$$

Dan is:

$$\frac{A}{F} = \frac{\Delta S}{\Delta W}$$

$$M = \frac{A \cdot \Sigma W + F \cdot \Sigma S}{2 \cdot (A + F)}$$



Figuur 5. De niet-inverterende configuratie.

M kan ook onafhankelijk van A of F worden berekend:

$$M = \left(\frac{1}{2} \right) \cdot \left(\frac{1}{\Delta W + \Delta S} \right) \cdot (\Delta S \cdot \Sigma W + \Delta W \cdot \Sigma S)$$

Voorbeeld

Stel dat we een lage schakeldrempel willen bij $S_{LO} = 0,5$ en een hoge drempel bij $S_{HI} = 3,0$, met als uitgangsniveaus $W_{LO} = 0$ en $W_{HI} = 5$. Dan geldt: $A/F = (3-0,5)/(5-0) = 0,5$ en $M = 0,5 \times 1/(5+2,5) \times (2,5 \times 5 + 5 \times 3,5) = 2$ V.

De resultaten zijn te zien in figuur 6.

Hysteresis voor stroomingen

Als we weerstand A verbinden met massa, verandert de schakeling van figuur 3 in een comparator met stroomingang, zoals in figuur 7 en figuur 8. De waarde van weerstand F volgt direct uit ΔW en ΔI . Dan kan de deilverhouding β worden bepaald. En tenslotte kunnen we de waarde van A berekenen. Een spreadsheet helpt hier, want we willen zowel β als A zo nauwkeurig mogelijk realiseren met behulp van standaard weerstandswaarden.

$$\beta = \frac{A}{A+F}; \quad R_p = \frac{A \cdot F}{A+F}$$

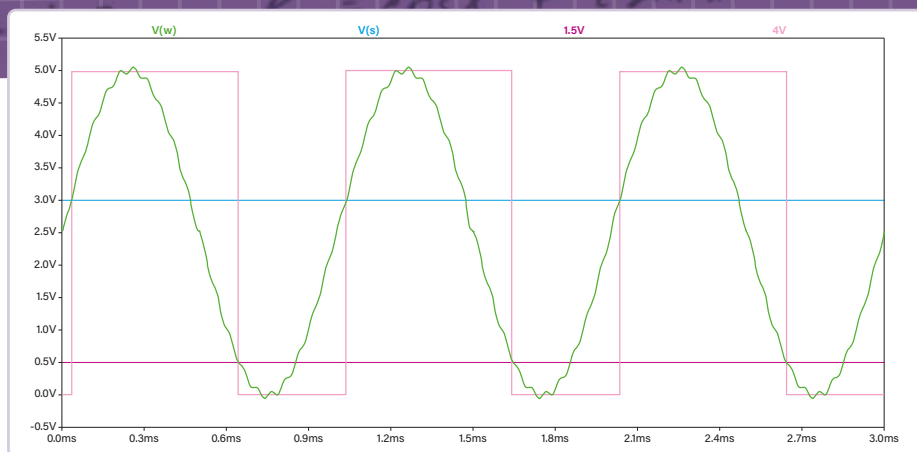
$$P = \beta \cdot W_H + I_L \cdot R_p = \beta \cdot W_L + I_H \cdot R_p$$

$$\beta \cdot \Delta W = R_p \cdot \Delta I; \quad \frac{R_p}{\beta} = F = \frac{\Delta W}{\Delta I}$$

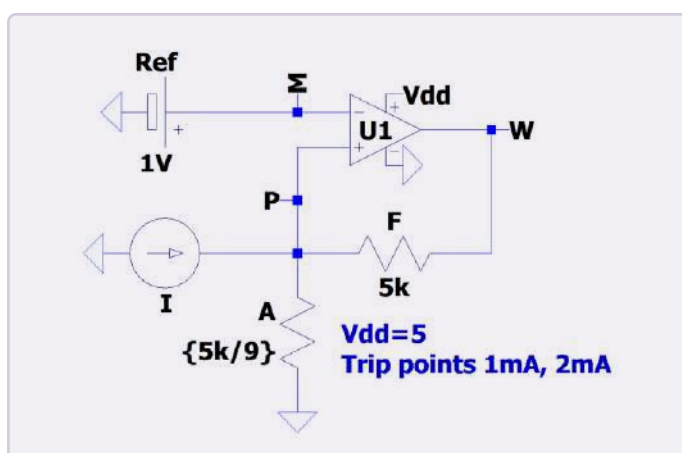
$$W_L = \frac{P - I_H \cdot R_p}{\beta} = \frac{P - I_H \cdot F \cdot \beta}{\beta} = \frac{P}{\beta} - I_H \cdot F$$

$$\frac{P}{\beta} = W_L + I_H \cdot F \rightarrow \beta = \frac{P}{W_L + I_H \cdot F}$$

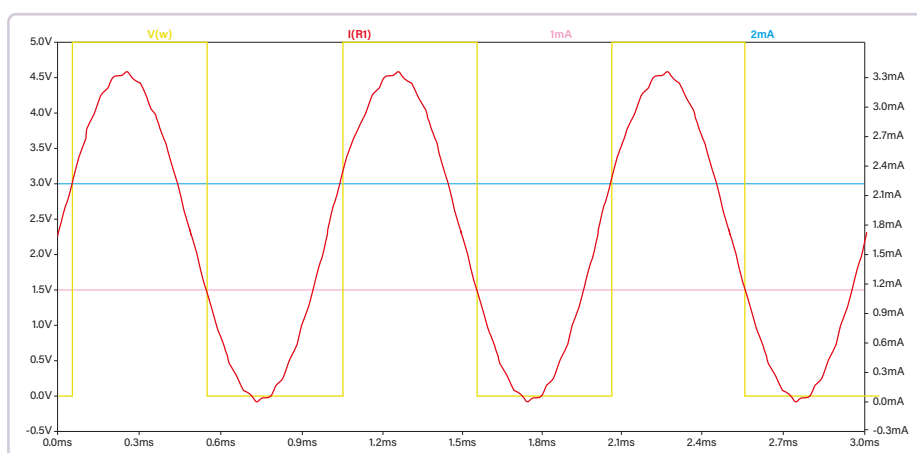
$$A = \frac{F \cdot \beta}{1 - \beta}$$



Figuur 6. De gesimuleerde signalen van de schakeling uit figuur 5.



Figuur 7. De schakeling uit figuur 3 aangepast voor een stroom-input.



Figuur 8. De gesimuleerde uitgangssignalen van de schakeling uit figuur 7.

Simulaties, spreadsheets en afleidingen

De simulaties, spreadsheets en de volledige afleidingen van de formules in dit artikel kunnen worden gedownload van de projectpagina bij dit artikel [1].

vertaling Evelien Snel — 200559-03

Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de redactie van Elektor via redactie@elektor.com.



Gerelateerde producten

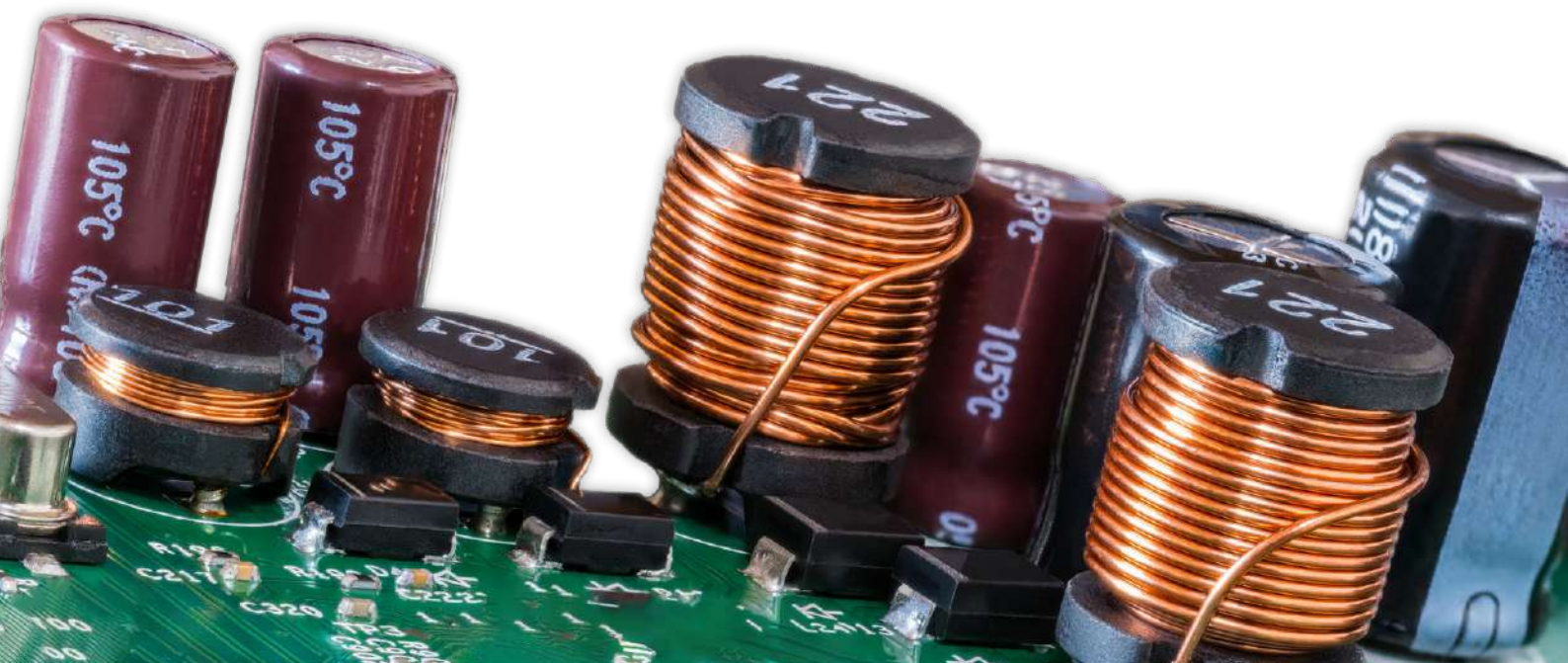
- > Paul Horowitz, Winfield Hill, *The Art of Electronics (3rd Edition)* (SKU 17167) www.elektor.nl/17167
- > Gilles Brocard, *The LTspice XVII Simulator* (SKU 19741) www.elektor.nl/19741

WEBLINK

[1] Projectpagina bij dit artikel: <http://www.elektormagazine.nl/200559-03>

Impedantie-analyzer op basis van een ESP32

simpel, weinig onderdelen en goedkoop!



Volker Ziemann (Zweden)

Een ESP32 als impedantie-analyzer die zelfs de resonantie van een LC-netwerk kan controleren? Ja, dit goedkope controllerboard kan het werk doen, met slechts een handvol externe componenten, en biedt toch een webgebaseerde gebruikersinterface.

We beginnen met wat theorie. De impedantie van een elektronische component $Z = U/I$ is de verhouding tussen de spanning U die over de component valt en de stroom I die er doorheen loopt. De wet van Ohm, geschreven in de vorm $R = U/I$, is een speciaal geval waarbij de weerstand R de eenvoudigste vorm van een impedantie is. Voor een condensator met capaciteit C hangt de impedantie af van de frequentie $\omega = 2\pi f$ en wordt gegeven door $-i/\omega C$; de imaginaire eenheid i is een compacte manier om aan te geven dat de fase van de spanning 90° naait op die van de stroom. Evenzo wordt de impedantie van een spoel met inductie L gegeven door $i\omega L$. Ook hier geeft de imaginaire eenheid i aan dat de spanning nu voorijlt op de stroom. We kunnen dus de impedantie van een component bepalen door de frequentie-afhankelijkheid van zijn impedantie en de faserelatie tussen spanning en stroom te onderzoeken.

Deze metingen worden vooral interessant bij meer dan één component. Als een spoel en een condensator parallel worden geschakeld, heeft de kring de hoogste impedantie bij de resonantiefrequentie waarbij de twee wisselstroomweerstand gelijk zijn. Bij een serieschakeling is de impedantie op dit punt het laagst. Met de Espressif ESP32-impedantie-analyzer kan men nu de frequentie-afhankelijke impedantiekromme



van RLC-netwerken vastleggen en zo hun eigenschappen bepalen. Het enige wat we nodig hebben is een manier om een sinusvormige signaal met constante amplitude te produceren, de frequentie ervan over een liefst zo breed mogelijk bereik te variëren ('sweep'), en dan snel de stroom te meten die door de component(en) loopt – doorgaans aangeduid als *device under test* (DUT). Opmerkelijk genoeg kan de ESP32 microcontroller die ik van Elektor ontving om deel te nemen aan een ontwerpwedstrijd in 2018 de meeste van deze dingen doen. Hij heeft een ingebouwde frequentiegenerator en een DAC die spanningen produceert met frequenties tot een paar honderd kilohertz. Bovendien heeft hij een ADC die spanningen bemonstert met snelheden tot een miljoen keer per seconde, zij het met een beetje valsspelen; maar daar komen we nog op terug. Aangezien de ESP32 slechts één ADC-kanal op hoge snelheid kan inlezen, en de ADC en DAC onafhankelijk van elkaar werken, kunnen we met de ESP32 alleen de grootte van de impedantie bepalen, maar niet de fase. Meting de door de DUT veroorzaakte faseverschuiving vereist speciale hardware, en zal daarom het onderwerp zijn van een apart artikel.

Analoog front-end

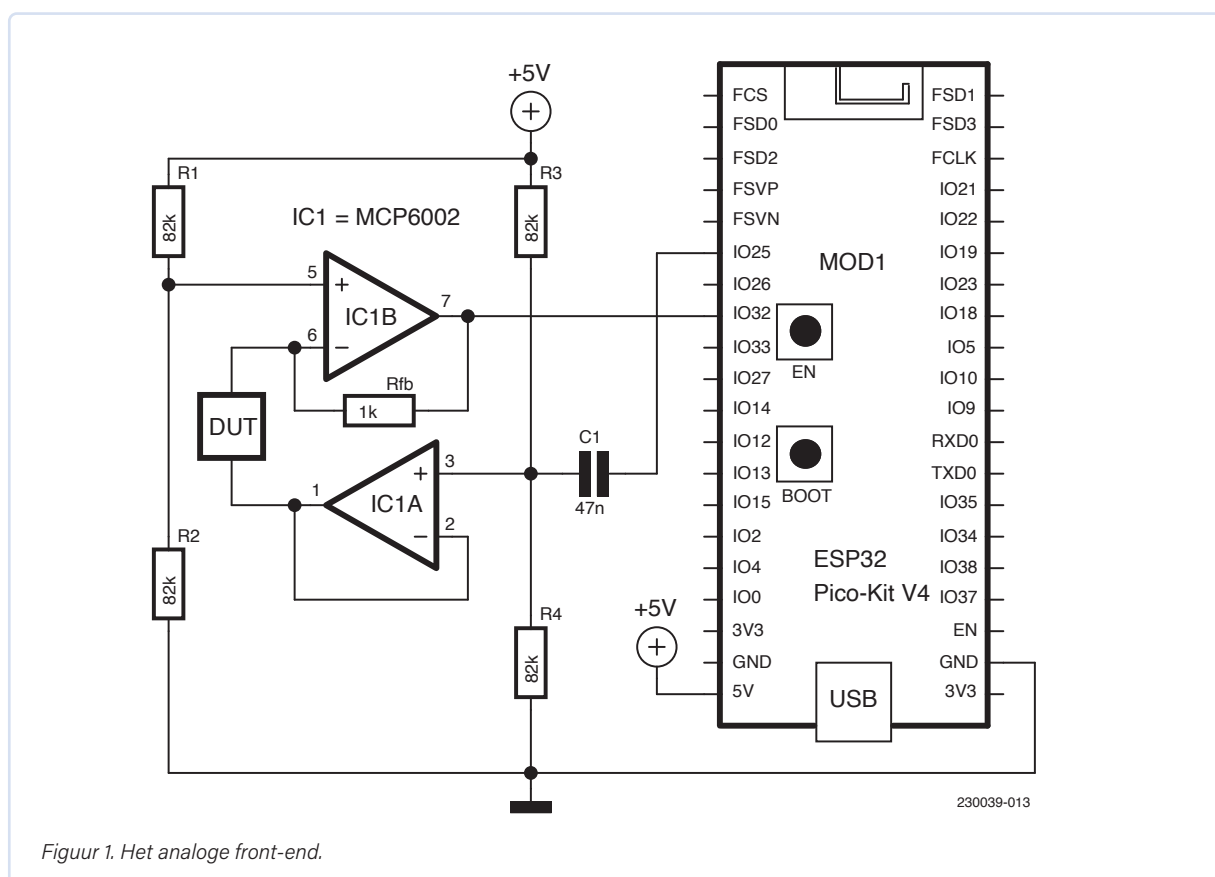
We moeten ervoor zorgen dat de amplitude van de DAC-spanning constant is en niet afhangt van de impedantie van de DUT. Dat is de taak van de onderste opamp in **figuur 1**. Deze dient als eenmaal versterkende buffer voor het signaal van pen 25 van de ESP32. Zijn laagohmige uitgang stuurt de DUT aan. De andere aansluiting van de DUT is verbonden met de min-ingang van een tweede opamp, die als

transimpedantieverstrekker is aangesloten. De stroom in diens min-ingang wordt met behulp van de terugkoppelweerstand R_{fb} omgezet in een spanning aan de uitgang, die naar de ADC-pin 32 van de ESP32 wordt gevoerd. Deze ADC kan alleen positieve spanningen verwerken. Daarom wordt het DC-niveau van het signaal met R1 en R2 verschoven naar de helft van de voedingsspanning. Met deze schakeling ziet de DUT een wisselspanning van constante amplitude terwijl we de stroom meten via de ADC.

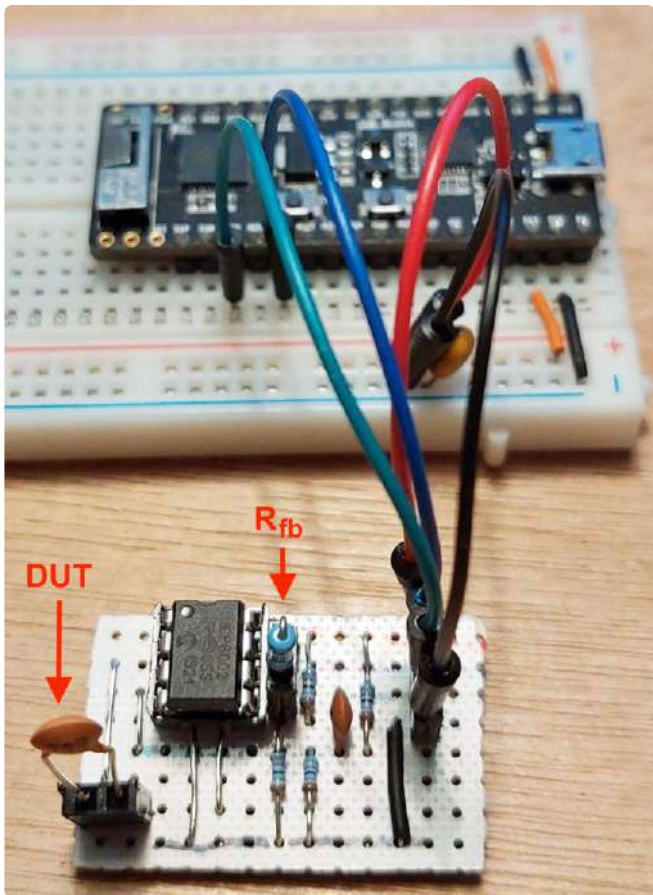
Figuur 2 toont het analoge front-end gemonteerd op een stukje gaatjesprint, met de condensator als DUT links onder. De rechtopstaande weerstand naast de opamp is de eenvoudig te verwisselen R_{fb} . De vier draden die dit printje verbinden met de ESP32 zijn GND (zwart), 3,3 V (rood), DAC pen 25 (groen), en ADC pen 32 (blauw).

Snelle ADC

De snelle modus van de ADC op de ESP32 maakt deel uit van het I2S-subsysteem dat normaliter wordt gebruikt om digitaal geluid te manipuleren. Bovendien ondersteunt de ESP32 een modus waarbij uitgangswaarden van de ADC in een buffer worden gekopieerd met behulp van directe geheugentoegang (DMA) en de CPU dus niet belast wordt. De ADC is vrijlopend, en DMA leest gegevens met een bepaalde snelheid. Zolang deze snelheid lager is dan de maximale conversiesnelheid van de ADC, die ongeveer 250 kS/s bedraagt, zijn alle monsters 'vers.' Als de snelheid daarentegen te hoog is, worden die samples naar de buffer gekopieerd, wat zichtbaar is als 'trapjes' in de data van de bemonsterde golfvorm.



Figuur 1. Het analoge front-end.



Figuur 2. Om met de ESP32 de impedantie van een DUT te meten, hebben we slechts een paar externe componenten nodig.

De ADC-bemonstering wordt ingesteld door de functie `is2sInit()`, die is afgeleid van de sketch `HiFreq_ADC.ino` die deel uitmaakt van het ESP32-supportpakket voor de Arduino IDE. Het grootste deel van de configuratie bestaat uit het vullen van de structuur `is2s_config` met de gewenste werkwijze, de bemonsteringsfrequentie, en een paar vlaggen die na flink wat archeologisch onderzoek in [1] zijn gevonden. Voordat we de functie verlaten, stellen we het dempingsniveau zo in dat de volle schaal van de ADC overeenkomt met de voedingsspanning van 3,3 V, kiezen we het ADC-kanaal, en schakelen we de uitgang in.

Frequentiegenerator

De ESP32 heeft een ingebouwd functieblok dat opeenvolgende waarden van een cosinusfunctie 'uitspuugt' met een instelbare snelheid. De uitvoer van dit blok kan, door een aantal configuratiebits goed in te stellen, naar het ingangsregister van de DAC worden geleid, waardoor een cosinusachtige uitgangsspanning ontstaat. Deze frequentiegenerator wordt aangestuurd door rechtstreeks naar de registers `SENS_SAR_DAC_CTRL1_REG` en `SENS_SAR_DAC_CTRL2_REG` te schrijven, hetgeen beschreven is in [1] en meer gedetailleerd in [2]. Volgens de uitleg uit [2] is de functie `cwDACinit()` voorbereid om deze registers te vullen met waarden om de uitgangsfrequentie en -amplitude te configureren. Na het opnemen van header-bestanden die ons in staat stellen mnemotechnische namen te gebruiken, gebruiken we `SET_PERI_REG_MASK(reg,bits)` om `bits` in register `reg` in te stellen. Bijvoorbeeld, het eerste van de volgende commando's

```
SET_PERI_REG_MASK(SENS_SAR_DAC_CTRL1_REG, SENS_SW_TONE_EN);
SET_PERI_REG_BITS(SENS_SAR_DAC_CTRL1_REG, SENS_SW_FSTEP, frequency_step, SENS_SW_FSTEP_S);
```

schakelt de interne snelle frequentiegenerator in. De tweede bepaalt de uitgangsfrequentie door de integer `frequency_step` op te geven, waarvan de waarde wordt afgeleid van een interne klok. De gewenste uitgangsfrequentie volgt een licht aangepaste vergelijking uit [1].

ESP32-sketch, serieel geregeld

Ik programmeerde de ESP32 met de oude versie 1.8.19 van de Arduino IDE, te vinden bij [3], en volgde de instructies om die te installeren. Op het moment van schrijven bood de nieuwe versie 2 van de IDE nog geen ondersteuning voor een later benodigde add-on. Ik moest ook de ESP32-ondersteuningsfuncties van [4] installeren en de installatie-instructies daarvoor volgen.

De volgende sketches zijn gebaseerd op [5], waar een diepgaande bespreking van vele aspecten te vinden is. De bedoeling is om de frequentieopwekking en de acquisitie via de seriële verbinding te besturen op basis van een eenvoudig protocol waarbij alleen strings heen en weer worden gestuurd tussen de ESP32 en een programma op een PC dat seriële communicatie aankan; hieronder vallen Python, Octave en LabView. Het protocol lijkt op het SCPI-protocol dat wordt gebruikt in oscilloscopen en andere meetapparatuur, waarbij een vraagteken een commando aangeeft dat een antwoord van de ESP32 verwacht. Het verzenden van `STATE?` geeft bijvoorbeeld het antwoord `STATE fmin fmax fstep`, waarbij de drie numerieke waarden het bereik en de stapgrootte van de frequentiesweep aangeven.

Nu de verschillende onderdelen van de sketch. Om te beginnen zijn er twee header-bestanden opgenomen – één voor de zelfgebakken ondersteunende functies voor ADC en DAC, de andere om toegang te krijgen tot een flash-gebaseerd SPIFFS-bestandssysteem op de ESP32. Dit laatste dient om kalibratiegegevens op te slaan. Standaardwaarden voor het bereik van de frequentiesweep en andere relevante variabelen zijn ook gespecificeerd. De `setup()`-functie van de sketch initialiseert de seriële communicatie, stelt de uitgangsfrequentie en -amplitude van de DAC in, en leest kalibratiegegevens uit het SPIFFS-bestandssysteem als dat beschikbaar is.

De functie `loop()` controleert met de aanroep van `Serial.available()` of er een request van de hostcomputer is binnengekomen, leest een regel (beëindigd door een `\n` line feed-karakter) en converteert wat is binnengekomen naar de karakter-array `line`. Nu wordt getest of de regel begint met een specifieke string. Als `line` bijvoorbeeld `FREQ 20000` bevat, wordt de numerieke waarde geëxtraheerd met `atoi(&line[5])`. Met de ingebouwde functie `atoi()` wordt de string die op de vijfde positie in de regel begint, omgezet in een geheel getal. Vervolgens wordt die waarde naar de variabele `dac25freq` gekopieerd en wordt de frequentiegenerator met `cwDACinit()` geïnitieerd. Op dezelfde wijze zijn alle relevante variabelen toegankelijk vanaf de hostcomputer.

Het meest interessante commando is `SWEEP?` Na ontvangst van dit commando en het declareren van de in deze sectie gebruikte variabelen, laat een for-lus de variabele `f` van `fmin` tot `fmax` lopen met een stapgrootte `fstep`. Binnen de lus wordt eerst de frequentie van de DAC op deze frequentie ingesteld, en na een korte vertraging worden de metingen van de ADC opgehaald met `is2_read()`.

```
for (float f=freqmin; f<freqmax;f+=freqstep) {
    dac25freq=f;
    cwDACinit(dac25freq,dac25scale,0);
```



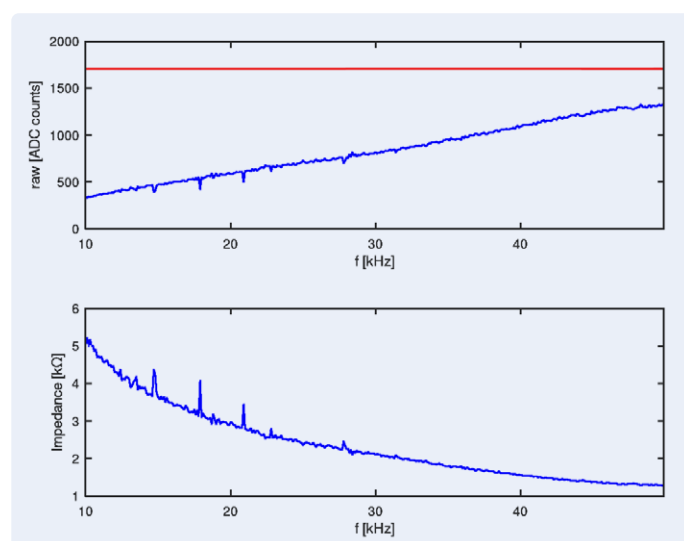
```
delay(50);  
i2s_read(I2S_NUM_0, &buffer, sizeof(buffer), &bytes_  
read, 15);  
...  
}
```

De grootte van het invoerargument `buffer`, gedefinieerd bovenaan de schets, wordt gebruikt om het aantal monsters te bepalen, hier 1024. De functie retourneert ook het aantal `bytes_read`. Aangezien elk monster twee bytes gebruikt, moeten we door twee delen wanneer we de monsters doorlopen om de minimum- en maximumwaarden te bepalen. Aangezien de door de ADC gemeten spanning evenredig is met de stroom door de DUT, gebruiken we deze piek-piek variatie als maat voor de amplitude van de stroom.

Kalibratie

Tijdens het doorlopen van de monsters worden ook de somwaarden `q1` en `q2` opgeteld. Door deze samen met de variabelen `S0`, `S1` en `S2` te gebruiken, krijgen we een rechte lijn op de datapunten, waarvan de parameters nodig zijn voor de kalibratie. De details van het algoritme worden uitgelegd in Appendix B van [5]. Het commando `SAVECALIB` slaat deze parameters op in het permanente SPIFFS-geheugen. Het commando `GETCALIB?` haalt ze op van de PC.

De kalibratieconstanten `calib_slope` en `calib_offset` zijn nodig om verschillende onbekende dempingsfactoren in het systeem in rekening te brengen, bijvoorbeeld ten gevolge van de AC-koppelcondensator aan de ingang van de onderste opamp. Deze dubbelzinnigheid wordt opgelost door het systeem te kalibreren met een weerstand met een bekende waarde als DUT. Alle andere impedanties worden dan bepaald ten opzichte van deze kalibratieweerstand. Idealiter is deze weerstand onafhankelijk van de frequentie, maar met kleine systematische veranderingen wordt rekening gehouden door naast de offset ook de helling als functie van de frequentie te gebruiken. De waarde van de ijkweerstand moet ongeveer dezelfde zijn als die van de terugkoppelweerstand van de transimpedantieverstrekter, om bij het bereik van de ADC te passen.



Figuur 3. Ruwe metingen en impedantie bij een condensator als DUT.

Voordat het systeem wordt gebruikt, moet het SPIFFS-bestandssysteem op de ESP32 worden geïnitialiseerd door een lege map met de naam `data` aan te maken in de directory waar de sketch is opgeslagen. Vervolgens moet de Arduino ESP32 filesystem uploader-plugin van [6] worden geïnstalleerd conform de instructies. Als dat is gebeurd, verschijnt er een nieuw menupunt *ESP32 Sketch Data Upload* in het *Tools*-menu van de Arduino IDE. Nadat je de seriële monitor hebt afgesloten, zal een klik op dat item het SPIFFS-bestandssysteem op de ESP32 aanmaken.

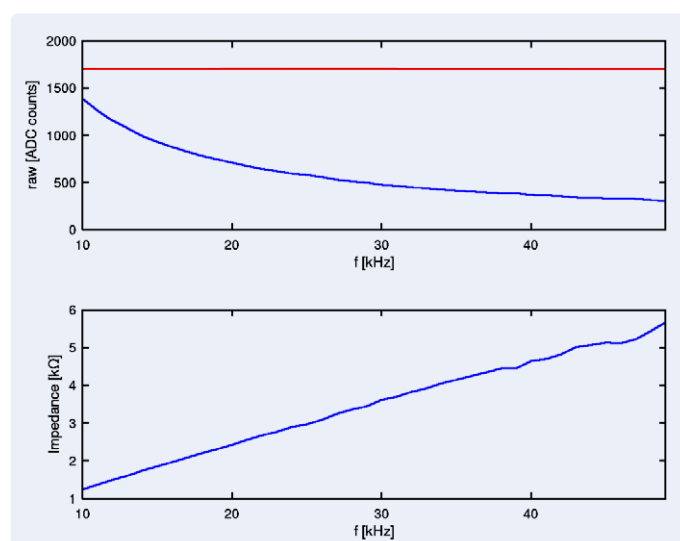
Besturing vanuit Octave of Python

Om de data-acquisitie vanaf de PC te besturen hoeft enkel de seriële poort geopend worden, waarvan de naam verschilt naargelang het besturingssysteem. Deze wordt gewoonlijk `COMn` genoemd onder Windows, `/dev/tty.usbserial-n` op een MAC of `/dev/ttyUSBn` onder Linux. Voor het verzenden van een commando, bijvoorbeeld om de startfrequentie van de sweep in te stellen, hoeft alleen `FMIN 10000` naar de seriële poort te worden geschreven. Om de huidige instellingen te lezen moet `STATE?` worden geschreven, even worden gewacht, en moeten de geretourneerde karakters worden gelezen, tot en met het line feed-karakter `/n`. Het antwoord luidt `STATE fmin fmax fstep`. Het uitvoeren van een frequentiesweep wordt gestart door `SWEEP?` te schrijven en de waarden te ontvangen, steeds één waarde per regel. Zodra alle gegevens beschikbaar zijn, kunnen we de seriële poort sluiten en een plot voorbereiden.

Scripts voor zowel Octave met geïnstalleerde *instrument toolbox* als voor Python met het *python-serial* package zijn opgenomen in het software-archief bij dit artikel. Merk op dat voor de eerste tests zelfs een terminalprogramma zoals Putty kan worden gebruikt.

Gebruik van het systeem

Om het systeem te gebruiken wordt een 1kΩ-kalibratieweerstand als DUT gebruikt en een frequentiesweep uitgevoerd. Door vervolgens het commando `SAVECALIB` uit te voeren, dat standaard is weggecommentarieerd in het begeleidende Octave-script `nwa.m`, worden de kalibratiegegevens op de ESP32 opgeslagen. Als nu het commando `SAVECALIB`



Figuur 4. Ruwe metingen en impedantie bij een spoel als DUT.

weer wordt weggecommentarieerd, de weerstand wordt vervangen door een condensator van 2,2 nF, en `nwa.m` weer wordt uitgevoerd, verschijnt de correct gekalibreerde plot van **figuur 3**. Het toont de verwachte omgekeerde evenredigheid met de frequentie, aan de hand waarvan we de capaciteit berekenen met het Octave-commando:

```
capacitance_nF= mean(1./(2*pi*Zabs*1e3.*xx*1e3))*1e9
```

op basis van de formule:

$$C = 1/(2\pi f Z_{\text{abs}})$$

Het gemiddelde over alle beschikbare datapunten wordt berekend met de ingebouwde functie `mean()`. De machten van tien zijn nodig om rekening te houden met de kilo's in kΩ en kHz en de nano's in nF. Bijvoorbeeld, `1e9` aan het eind converteert de capaciteit van farad naar nanofarad.

Herhaling van de sweep met de condensator vervangen door een spoel van 22 mH levert de grafieken van **figuur 4** op. Zoals verwacht neemt de impedantie van een spoel (onderste grafiek) lineair toe met de frequentie, zodat de zelfinductie kan worden bepaald aan de hand van de formule:

$$L = Z/2\pi f_{\text{abs}}$$

of geschreven in Octave met:

```
inductance_mH= mean(1e3*Zabs./(2*pi*xx*1e3))*1e3
```

die ook het gemiddelde neemt over alle datapunten met `mean()`. Ook hier zijn de machten van tien nodig om rekening te houden met de kilo's in kΩ en kHz en de milli's in mH.

Webgebaseerde gebruikersinterface

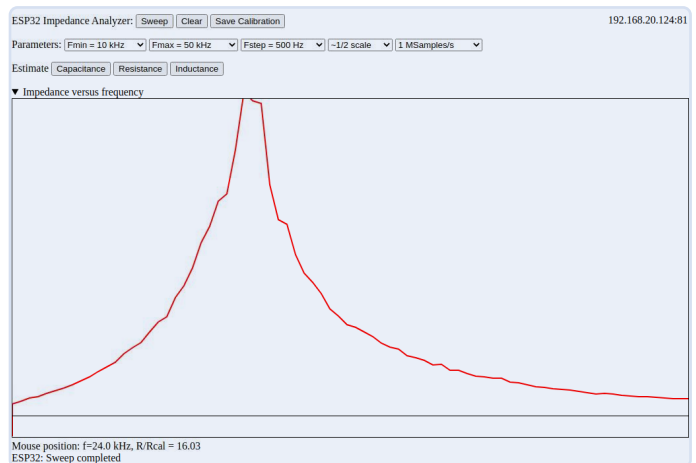
Tot nu toe hebben we de impedantiemeting bestuurd vanuit Octave of Python, maar nu gaan we een webbrowser gebruiken om de metingen te besturen en weer te geven. Daarom wordt de ESP32 geconfigureerd om een webserver te draaien die de webpagina van **figuur 5** produceert. Zodra de browser deze webpagina ontvangt, opent embedded JavaScript-code een tweede communicatiekanaal naar de ESP32, gebaseerd op websockets. Hier vervult een websocket de rol van de seriële verbinding die wordt gebruikt om tekstberichten heen en weer te sturen. Deze berichten worden verstuurd door op de knoppen bovenaan de webpagina te klikken. Daarmee starten we een frequentiesweep, wissen de weergegeven curves en slaan de kalibratiegegevens op. Op de regel daaronder kunnen de parameters die de ESP32 besturen, waaronder het bereik van de sweep en de bemonsteringsnelheid van de ADC, via keuzemenu's worden aangepast. De knoppen op de derde regel berekenen de capaciteit, weerstand en zelfinductie en geven het resultaat weer op de statusregels onder de grafiek. De onderste regel geeft de van de ESP32 ontvangen informatie weer. In de grafiek is de horizontale as de frequentie-as met het bereik dat is opgegeven in de menu's bovenaan de pagina. De verticale as toont Z_{abs} ten opzichte van de kalibratieweerstand, die wordt weergegeven als horizontale blauwe lijn. Een klik op de curve toont de frequentie en Z_{abs} in de statusregel.

Sketch op de ESP32

Na het aanpassen van de WiFi-netwerknnaam (SSID) en het wachtwoord moeten de support-bestanden voor WiFi, websocket en webserver worden opgenomen, waarna we de webserver `server2` configureren om te luisteren op netwerkpoot 80, en de `websocket` om te communiceren via poort 81. De tekstberichten die tussen de browsers worden verzonden zijn JSON-geformatteerd, in de vorm `{"INFO": "Yada yada"}`. Het parsen van deze berichten en het construeren van meer complexe berichten wordt afgehandeld door de `ArduinoJson`-bibliotheek, terwijl ondersteuning voor de ADC en DAC wordt geboden door `ESP32_I2Sconfig.h`, zoals eerder.

```
const char* ssid      = "YOUR_SSID";
const char* password  = "YOUR_PASSWORD";
#include <WiFi.h>
#include <WebSocketsServer.h>
WebSocketsServer webSocket = WebSocketsServer(81);
#include <WebServer.h>
#include <SPIFFS.h>
WebServer server2(80);
#include <ArduinoJson.h>
#include "ESP32_I2Sconfig.h"
```

Na het specificeren van verschillende variabelen worden helperfuncties gedefinieerd, waarvan `websocketEvent()` de belangrijkste is. Deze wordt aangeroepen wanneer een bericht van de browser binnenkomt. Als het een JSON-geformatteerd bericht is, haalt het volgende codefragment het commando `cmd` en de waarde `val` eruit met behulp van functies uit de `ArduinoJson`-bibliotheek. Afhankelijk van het commando worden dan de juiste acties geactiveerd. Als bijvoorbeeld het commando `SWEEP` wordt ontvangen, rapporteert hij terug aan de browser met `sendMSG()` en zet hij de variabele `mmode` op één, die in de hoofdloop wordt geïnterpreteerd. Aangezien `websocketEvent()` asynchroon wordt aangeroepen, onderbreekt hij andere activiteiten en dit moet zo kort mogelijk worden gehouden; daarom wordt de



Figuur 5. De ESP32-geproduceerde webpagina toont een resonantiepiek in de impedantiecurve bij parallelschakeling van een 2,2nF-condensator, een 22mH-spoel en een 33kΩ-weerstand.



sweep uitgesteld tot het hoofdprogramma. Aan de andere kant gebruikt het instellen van de startfrequentie van de sweep `freqmin` niet veel CPU-cycli en wordt dit binnen `WebSocketEvent()` afgehandeld.

```
DynamicJsonDocument root(300);
deserializeJson(root,payload);
const char *cmd = root["cmd"];
const long val = root["val"];
if (strstr(cmd,"SWEEP")) {
    sendMSG("INFO","ESP32: Received Sweep command");
    mmode=1;
} else if (strstr(cmd,"FMIN")) {
    freqmin=val;
    Serial.printf("FreqMin = %g\n",freqmin);
} else
...

```

Een reeks commando's die lijken op de eerder gebruikte wordt op dezelfde manier behandeld.

Het volgende codefragment uit de `setup()`-functie maakt eerst verbinding met het WLAN met de eerder verstrekte referenties. Het drukt puntjes af via de seriële lijn tot de verbinding tot stand gekomen is, en drukt dan het IP-nummer af, start de websocket-communicatie, en registreert `WebSocketEvent` om berichten af te handelen die via de websocket binnenkomen. Merk op dat de seriële lijn niet strikt noodzakelijk is in dit deel, maar het is niettemin geruststellend om te zien wat er gebeurt op de ESP32, vooral tijdens de ontwikkeling van het systeem.

```
WiFi.begin(ssid, password);
while(WiFi.status() != WL_CONNECTED)
Serial.print("\nConnected to ");
Serial.print(ssid);
Serial.print(" with IP address: ");
Serial.println(WiFi.localIP());
WebSocket.begin();
WebSocket.onEvent(WebSocketEvent);

```

Vervolgens controleren we, nog steeds in `setup()`, of het SPIFFS-bestandssysteem bestaat, net als in het eerste deel, en laden we de kalibratiegegevens daaruit. Nu bevat het SPIFFS-bestandssysteem echter ook het bestand `esp32_impedance_analyzer.html`, dat de webpagina beschrijft. Na het starten van `server2` wordt dit HTML-bestand geregistreerd om standaard te worden afgeleverd aan verbonden browsers, zoals aangegeven door het eerste argument in `server2.serveStatic()`:

```
server2.begin();
server2.serveStatic("/",SPIFFS, "/esp32_impedance_analyzer.html");

```

De rest van de `setup()`-functie lijkt grotendeels op de eerdere sketch. In de functie `loop()` wordt eerst alles met betrekking tot de http- en websocket-servers afgehandeld, voordat de variabele `info_available` wordt gecontroleerd. Deze wordt geset door `sendMSG()` om aan te

geven dat de `info_buffer` een JSON-geformatteerd bericht bevat, dat onmiddellijk naar de browser wordt verzonden met een aanroep van `WebSocket.sendTXT()`.

```
server2.handleClient(); // handle http server
WebSocket.loop();       // handle websocket server
if (info_available==1) {
    info_available=0;
    WebSocket.sendTXT(websock_num,
info_buffer,strlen(info_buffer));
}

```

Dan wordt de waarde van `mmode` gecontroleerd en wordt de juiste actie gestart. Zo voert `mmode==1` de frequentiesweep uit met code die sterk lijkt op de eerder gebruikte code. Hier wordt alleen een JSON-geformatteerd bericht met de naam `WF0` (voor golfvorm, waveform) opgebouwd en opgeslagen in de variabele `doc` met:

```
doc["WF0"][nn-1]=floor((512/16)*Z);

```

waarbij de variabele `nn` over de frequentiepunten loopt en `Z` de absolute waarde is van de impedantie bij die frequentie. Merk op dat de variabele wordt geschaald voor 512 pixels die het bereik van nul tot 16 kΩ bestrijken, en dat de waarden worden omgezet in een integer met `floor()`. Nadat de lus is voltooid, wordt de golfvorm naar de browser verzonden met:

```
serializeJson(doc,out);
WebSocket.sendTXT(websock_num,out,strlen(out));
sendMSG("INFO","ESP32: Sweep completed");

```

en kondigt aan dat de sweep klaar is met `sendMSG()`. De andere waarden van `mmode` slaan de kalibratie op en rapporteren de geschatte capaciteit, weerstand of zelfinductie aan de browser. Het bestand `esp32_impedance_analyzer.html` beschrijft de webpagina en bevat de JavaScript-code die hem tot leven brengt.

De webpagina

De meeste interactieve webpagina's gebruiken `<style>`-tags om algemene aspecten te beschrijven van hoe dingen op de pagina verschijnen, gevolgd door HTML-opdrachten die de webpagina beschrijven, en JavaScript-instructies om de pagina interactief te maken.

De volgende stijl-instructie bovenaan `esp32_impedance_analyzer.html` zorgt ervoor dat er een rand rond het displaygebied wordt getekend en dat twee weergegeven curven in rood en zwart worden weergegeven. De laatste instructie zorgt ervoor dat het IP-nummer aan de rechterkant van de webpagina wordt getoond:

```
<style>
#displayarea { border: 1px solid black; }
#trace0 { fill: none; stroke: red; stroke-width: 2px;}
#trace1 { fill: none; stroke: black; stroke-width: 1px;}
#ip
</style>

```

Het belangrijkste deel van de beschrijving van de webpagina staat tussen de tags `<BODY>` en `</BODY>`. Helemaal bovenaan staat de beschrijving van de knoppen. De definitie van de eerste knop staat tussen de `button`-tags en luidt:

```
<button id="sweep" type="button"
onclick="sweep();">Sweep</button>
```

Hiermee wordt `Sweep` op de knop weergegeven, en een klik voert de JavaScript-functie `sweep()` uit. Dergelijke acties gekoppeld aan een gebeurtenis worden gewoonlijk *callback-functies* genoemd. Door aan de knop een ID toe te kennen, in dit geval `Sweep`, kunnen we later de eigenschappen ervan wijzigen, bijvoorbeeld de weergegeven tekst of de te activeren actie. Het definiëren van de andere knoppen gaat op dezelfde manier.

Het keuzemenu op de tweede regel heeft een iets andere syntaxis. De definitie van dit menu staat tussen `SELECT`-tags. Als een van de items is geselecteerd, roept het `setDacFreqMin(this.value)` aan, waarbij `this.value` de waarde is die is opgegeven in de verschillende `OPTION`-tags. De `OPTGROUP`-tags zijn slechts decoratief.

```
<SELECT onchange="setDacFreqMin(this.value);">
<OPTGROUP label="Sweep start frequency">
<OPTION value="1000">Fmin = 1 kHz</OPTION>
<OPTION value="2000">Fmin = 2 kHz</OPTION>
<OPTION value="5000">Fmin = 5 kHz</OPTION>
<OPTION selected="selected" value="10000">
    Fmin = 10 kHz</OPTION>
<OPTION value="20000">Fmin = 20 kHz</OPTION>
<OPTION value="50000">Fmin = 50 kHz</OPTION>
</OPTGROUP>
</SELECT>
```

Ten slotte is het gebied om de impedantie weer te geven een *Scalable Vector Graphic* (SVG) dat een gespecificeerde grootte van 1024 × 512 pixels heeft en twee `path`'s weergeeft met ID `trace0` en `trace1`. Met de `id` kunnen we ze later wijzigen. De eigenschap `d` beschrijft de golfvorm. Initialiseren gebeurt door de cursor te verplaatsen naar pixel (0,200) met `M0 200`. Later wordt `d` opnieuw gedefinieerd met de golfvorm `WF0` die binnenkomt van de ESP32.

```
<svg id="displayarea" width="1024px" height="512px">
<path id="trace0" d="M0 200" />
<path id="trace1" d="M0 200" />
</svg>
```

Ten slotte worden de twee statusregels gedefinieerd met:

```
<div id="status">Status window</div>
<div id="reply">Reply from ESP32</div>
```

Ze worden benoemd via `id`, waarmee later de weergegeven tekst kan worden gewijzigd. Evenzo wordt het gebied rechtsboven voor het IP-nummer `ip` genoemd.

JavaScript

Alle JavaScript-opdrachten worden omsloten door `SCRIPT`-tags. Na de openingstag worden verschillende variabelen gedefinieerd, en wordt het IP-nummer van de ESP32 bepaald met:

```
var ipaddr=location.hostname + ":81";
document.getElementById('ip').innerHTML=ipaddr;
```

Hier wordt het poortnummer van de websocket gekoppeld aan de ESP32 en onmiddellijk wordt de tekst van de tag, gelabeld `ip`, bijgewerkt. De ruimte om tekst weer te geven wordt geadresseerd door de langere constructie in de tweede regel, waarbij `document` verwijst naar de webpagina zelf. Het deel na de punt geeft toegang tot het benoemde element `ip`, en `innerHTML` verwijst naar de weergegeven tekst die wordt gewijzigd om `ipaddr` te tonen. Deze constructie wordt veel gebruikt om toegang te krijgen tot benoemde elementen en om hun eigenschappen te wijzigen. De functies `toStatus()` en `toReply()` volgen dit schema om tekst te kopiëren naar de statusregels onderaan de webpagina.

Nu het adres van de websocket op de ESP32 bekend is, wordt deze geopend met `new WebSocket()` en vervolgens worden callback-functies gekoppeld aan de events `onopen`, `onclose` en diverse andere. Vaak wordt alleen een kort bericht naar de JavaScript-console gestuurd met `console.log()`. De console is toegankelijk via de *Developer tools* van de browser of met `Ctrl-Shift-I` bij Chromium.

```
var websock = new WebSocket('ws://' + ipaddr);
websock.onopen = function(evt)
;
```

De interessantste callback-functie reageert op binnenkomende berichten van de ESP32. De enigszins ingekorte functie `websock.onmessage()` ontleedt eerst het binnenkomende JSON-geformatteerde bericht, dat wordt opgeslagen in `event.data` en plaatst de commando/waarde-paren in de structuur `stuff`. Als `stuff` het commando `INFO` bevat, wordt de bijbehorende waarde opgeslagen in `val` en weergegeven op de webpagina met `toReply()`. Als het commando `WF0` is, bevat het de golfvorm met de impedantiegegevens. Het aantal aankomende datapunten wordt bepaald met `val.length` en gebruikt om de horizontale as zo aan te passen dat alle beschikbare 1024 pixels worden benut. Vervolgens wordt de eigenschap `d` van het pad geïnitieerd, en er worden punten aan toegevoegd, één voor elk item in `val`. Aangezien pixel (0,0) linksboven is, moet de verticale positie worden omgekeerd met behulp van `512-val[i]`. Tenslotte wordt de zwarte referentielijn met de kalibratieweerstand weergegeven.

```
websock.onmessage=function(event) {
var stuff=JSON.parse(event.data);
var val=stuff["INFO"]; // info
if (val != undefined)
var val=stuff["WF0"]; // waveform0
if (val != undefined) {
nstep=Math.floor(0.5+1024/val.length)
pixmax=nstep*val.length;
```



```
var d="M0 511";
for (i=0; i<val.length; i++)

document.getElementById('trace0').setAttribute('d',d);
d="M0 480 L1024 480";
document.getElementById('trace1').setAttribute('d',d);
}
}
```

De rest van het JavaScript bestaat voornamelijk uit callback-functies naar knoppen en menu's. De functie `sweep()`, geactiveerd door de corresponderende knop, luidt:

```
function sweep() {
websocket.send(JSON.stringify({ "cmd" : "SWEEP", "val" :
-1 }));
}
```

De functie `JSON.stringify()` verpakt zijn argument in een correct geformatteerd bericht en `websocket.send()` stuurt het naar de ESP32. Alle andere callback-functies volgen hetzelfde schema.

Net voor het einde van het JavaScript-gedeelte wordt `showCoordinates()` gedefinieerd om de frequentie en impedantie weer te geven die overeenkomen met de pixel op het grafiekgebied waarop is geklikt. Deze functie krijgt een callback van een `mousedown`-event door een `addEventListener()` te koppelen aan `displayarea`:

```
document.getElementById("displayarea")
.addEventListener('mousedown', showCoordinates, false);
```

Deze functie maakt het op een handige manier mogelijk om direct frequenties en de bijbehorende impedanties te bepalen uit de weergegeven curve.

In figuur 5 is te zien dat het resonantiegedrag van een RLC-parallelkring die wij als DUT hebben gebruikt, een resonantiepiek vertoont bij 24 kHz, waarbij de impedantie groter is dan 16 kΩ.

De firmware en andere bestanden kunnen gratis worden gedownload van [7]. Nu is de impedantie-analyzer zelfstandig in die zin dat er geen besturingsprogramma nodig is. Alle communicatie vindt plaats tussen de ESP32 en een browser, zelfs die op een smartphone. ◀

230039-03

Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de redactie van Elektor via redactie@elektor.com.

Over de auteur

De belangstelling van Volker Ziemanns voor elektronica begon met de 40 W Edwin-versterker (Elektor, midden jaren '70), maar hij sloeg een andere weg in, studeerde natuurkunde en werkte sindsdien met deeltjesversnellers – bij SLAC in de VS, CERN in Genève en nu in Uppsala, Zweden. Aangezien elektronica een belangrijke rol speelt bij de besturing en gegevensverwerking in versnellers, kwam zijn vroege belangstelling voor elektronica hem gedurende zijn hele carrière van pas. Hij doceert nu aan de Universiteit van Uppsala. Een van zijn boeken over data-acquisitie met Arduino en Raspberry Pi raakt aan het onderwerp van dit artikel.



Gerelateerde producten

- **ESP32-DevKitC-32D (SKU 18701)**
www.elektor.nl/18701
- **OWON HDS1021M-N 1-ch Oscilloscope (20 MHz) + Multimeter (SKU 18778)**
www.elektor.nl/18778

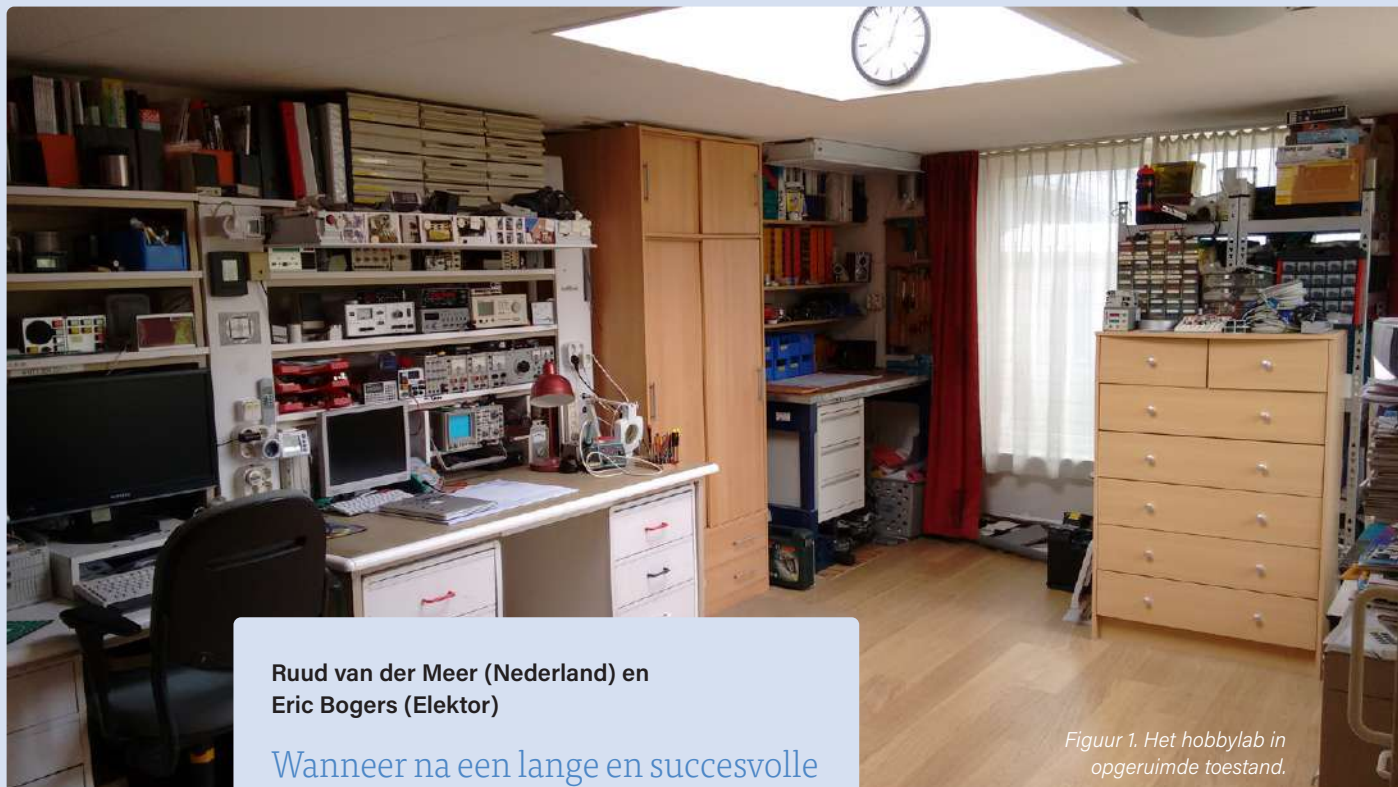
WEBLINKS

- [1] ESP32 Technical Reference Manual (Version 4.7), te vinden op :
https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf
- [2] Uitleg ESP32 cosinus-golfvormgenerator: <https://github.com/krzychb/dac-cosine>
- [3] Arduino-website: <https://www.arduino.cc>
- [4] Arduino Support Functions: <https://github.com/espressif/arduino-esp32>
- [5] V. Ziemann, A Hands-on Course in Sensors Using the Arduino and Raspberry Pi, 2nd ed., CRC Press, Boca Raton, 2023:
- [6] ESP32fs-plugin: <https://github.com/me-no-dev/arduino-esp32fs-plugin>
- [7] Broncode voor dit project op GitHub: <https://tinyurl.com/4awvvec>



Oost West Lab Best

zelfbouw moet gestimuleerd!



Ruud van der Meer (Nederland) en
Eric Bogers (Elektor)

Figuur 1. Het hobbylab in
opgeruimde toestand.

Wanneer na een lange en succesvolle carrière in de elektronica eindelijk de pensioengerechtigde leeftijd bereikt is, gaan velen zich aan iets heel anders wijden, bijvoorbeeld tomaten kweken of cavia's fokken – geeft niet wat als het maar geen elektronica is. Niet zo Ruud van der Meer uit Roelofarendsveen – na meer dan veertig jaar bij Siemens is zijn enthousiasme nog altijd niet te stuiten...



*Er wordt
tegenwoordig
te weinig
elektronica
zelf gebouwd.*

Na de Technische school en een opleiding in de meet- en regeltechniek, plus de opleiding (hoger) elektronicus, ging Ruud in 1970 bij Siemens aan de slag (afdeling meet- en regeltechniek, meetinstrumenten en kalibratie). Ruud vertelt:

“Hier kwamen de kansen. Ik heb daar in korte tijd een aantal producten ontwikkeld – LED-displays, PI-regelaars en meetwaarde-omvormers, en door het ontwikkelen van testapparaten en -systemen heb ik allerlei werkprocessen kunnen verbeteren. Na een aantal jaren ben ik teamleider geworden en heb ik (met de Siemens Hobbycomputer Club) een eigen PC-systeem (SUMO80) ontwikkeld.”

Al snel merkte Ruud dat niet iedereen zonder extra studie de weg naar nieuwe technieken kon vinden; daarom is hij (ook) les gaan geven binnen het bedrijf en op de Regionale Opleidingscentra op het gebied van elektrotechniek en elektronica, digitale technieken, communicatietechnieken en mechatronica.

“Ik was op een gegeven moment aan een nieuwe uitdaging toe, en die kwam toen mij gevraagd werd hoofd van de bedrijfs-vakopleiding (BVO) van Siemens te worden. Deze telde toen ongeveer 60 leerlingen die in de elektrotechniek werden opgeleid. Een van mijn taken (naast het theoretische deel van de opleiding) was het verbeteren van de instroom in het bedrijf, die door de jaren steeds minder was geworden. Ik heb dat gedaan door onder andere de opleiding breder te maken met elektronica en PLC-techniek, communicatietechniek en in het laatste jaar ook mechatronica, in samenwerking met docenten van de TU Delft.”

In 2005 werd de BVO van Siemens gesloten, en moest Ruud op zoek naar een job in een andere richting (maar nog steeds bij Siemens). Dat is uiteindelijk gelukt bij de afdeling voor gebouw-automatisering en brandbeveiliging.

“Om een lang verhaal kort te maken: het leek daar op het gebied van werkprocessen wel het stenen

tijdperk, maar na een tijdje kreeg ik deze afdeling beter gestroomlijnd. En toen ben ik in 2016 met pensioen gegaan.”

Weer werd het tijd voor wat anders. Ruud is de eerste energiecoach van de gemeente Kaag en Braassem geworden, en heeft inmiddels veel adviezen gegeven op het gebied van duurzaamheid. Mede hiervoor heeft hij veel domotica-systemen ontwikkeld op basis van Arduino.

“Na een aantal presentaties over de Arduino bij de Hobby Computer Club is mij gevraagd dit ook aan de LVU (Volksuniversiteit van Leiderdorp) te gaan doen. Dat is inmiddels zes jaar geleden. Ik heb daarna ook Arduino-lessen verzorgd aan de Volksuniversiteiten van Alphen a/d Rijn, Lisse en Hillegom. We hebben inmiddels een grote groep enthousiaste mensen om ons heen verzameld, organiseren maandelijks een Arduino-café (en natuurlijk jaarlijks de World Arduino Day). Ook voor de elektronica-soldeercursus bestaat veel belangstelling.”

“Er wordt tegenwoordig te weinig elektronica zelf gebouwd, en dat terwijl dat dankzij Arduino zo eenvoudig is.”

Thuislab

Voor al die cursussen en lessen heeft Ruud veel lesmateriaal (schema's, printen, software) ontwikkeld. Het meeste hiervan is in zijn thuislab ontstaan (en overigszo ook voor Siemens). **Figuur 1** geeft een impressie van zijn thuislab.

“Ik heb al vanaf 1965 een eigen thuislab (of tenminste een hobbyruimte); de foto toont de ruimte die ik sinds 2005 in gebruik heb. Links is het PC-deel te zien, in het midden het elektronica- en meetgedeelte, en rechts het gedeelte voor mechanisch werk. Hier is bijvoorbeeld mijn mechanische reinigingsrobot ontstaan als vingeroefening in mechatronica (**figuur 2**).”



Figuur 3. Het elektronicagedeelte met meet- en testapparatuur.

“Ik heb veel meet- en testapparaten zelf ontwikkeld; mijn zelfgebouwde buizen-oscilloscoop is helaas niet meer in mijn bezit, maar veel ontwerpen zijn behouden gebleven (en door collega's nagebouwd). **Figuur 3** geeft een impressie.”

“Een recent project is gebaseerd op een bestaande zelfbouw-calculator van KKmoon. Die biedt in zijn oorspronkelijke vorm naar mijn mening te weinig mogelijkheden; ook is hij lastig herprogrammeerbaar. Omdat het echter een leuk hardware-ontwerp is, heb ik hiervoor een nieuw hart ontwikkeld, op basis van (uiteraard) een Arduino. Met een groep Arduino-cursisten bouwen we inmiddels een aantal van deze DIY-calculators. De functionaliteit ervan is (naast de basisfunctie als rekenmachine) gericht op mensen die regelmatig programmeren en/of technisch bezig zijn. **Figuur 4** toont de buitenzijde plus een blik op het innerlijk.”

Deze Arduino-Multicalculator zal in een van de komende nummers van Elektor nog nader aan de orde komen. ◀

230035-03

Vragen of opmerkingen?

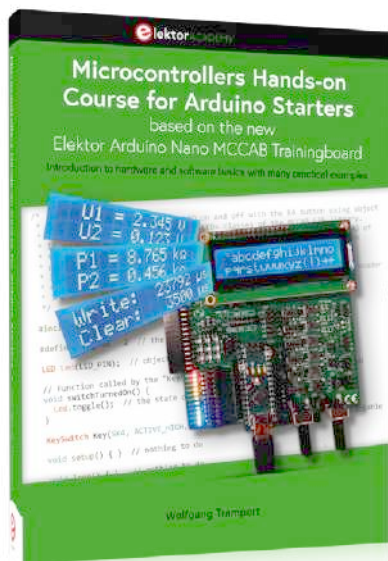
Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur of naar de redactie van Elektor via redactie@elektor.com.



Figuur 2. Reinigingsrobot.



Figuur 4. De Multicalculator van buiten en van binnen.



Het MCCAB Arduino Nano Training Board

alle hardware voor de
"Microcontrollers Hands-On Course"



Wolfgang Trampert (Duitsland) en Jan Buiting (Elektor)

Het in elkaar zetten van kleine Arduino-schakelingen is meestal een kwestie van componenten in een breadboard prikken en verbindingen maken met behulp van jumperkabeltjes. Maar al dat inprikken en puzzelen kost vaak meer tijd dan het schrijven van het programma. En bij de componentenopstelling voor bijvoorbeeld een looplicht met 11 LED's en net zoveel serieweerstanden kan gemakkelijk iets misgaan en dan ben je uren bezig met het debuggen van de hardware! Dan is een speciaal training-board, zoals Elektor's nieuwe "MCCAB", een stuk handiger, zeker als die met een mooi instructieboek wordt geleverd.

Opmerking van de redactie: dit artikel is een gedeelte uit het 455 pagina's tellende Elektor-boek *Microcontrollers Hands-on Course voor Arduino Starters* (Elektor, 2023). Het is enigszins bewerkt en aangepast aan de stijl van Elektor Magazine. De auteur en de redactie hebben hun best gedaan op die bewerking en zijn graag bereid eventuele vragen te beantwoorden. Je vindt hun contactgegevens in het kader **Vragen of opmerkingen?**

De "MCCAB" (Microcontroller Crash Course for Arduino Beginners), een training-board met Arduino Nano, is door de auteur speciaal ontworpen en gefabriceerd voor Elektor als ondersteuning bij de *Microcontrollers Hands-on Course for Arduino Starters*, die wordt geleverd als een groene Elektor-gids. Het MCCAB-board, de gids (in het Engels of Duits) en een Arduino Nano zijn als bundel verkrijgbaar in de Elektor Store [1].

Op het board, die we van nu af aan MCCAB zullen noemen, is een uitgebreid assortiment van componenten al aangesloten op de ingeprikte microcontrollermodule, of kan met eenvoudige jumperkabels worden aangesloten. Dat betreft onder andere schakelaars /drukknoppen, LED's, potentiometers, buzzers, een LCD, interfaces, niveau-omzetters voor seriële verbindingen en drivers voor externe componenten. Ook complete externe modules kunnen via een connector met de MCCAB worden verbonden, of communiceren met de microcontroller op de kaart via de seriële interfaces. Dit bespaart je het moeizaam opbouwen van experimentele schakelingen, zodat je je kunt concentreren op de hoofdzaken: je eigen programma! Dus we laden de programma's van onze oefeningen in de ATmega328P-microcontroller op de MCCAB, waar ze dan in de vorm van programma-code draaien.

Enkele van de 32 pinnen van de ATmega328P-microcontroller worden gebruikt voor de voeding, voor

het aansluiten van een kristal voor de ingebouwde oscillator van de klokgenerator of als RESET-input voor de resetschakelaar op de MCCAB. (Als we die indrukken, wordt de microcontroller gereset en start het programma opnieuw.) Maar de meeste aansluitingen zijn beschikbaar als vrij te programmeren IO-lijnen, die de microcontroller kan gebruiken om met de buitenwereld te communiceren. Op de MCCAB zijn deze pinnen, op enkele uitzonderingen na, beschikbaar op connectorstrips.

In **figuur 1** zie je het schema van de MCCAB met kleur-gecodeerde functieblokken. De handleiding voor de MCCAB, met een uitgebreide beschrijving van alle componenten, kan gratis worden gedownload van de Elektor-website [2].

De functieblokken van de MCCAB

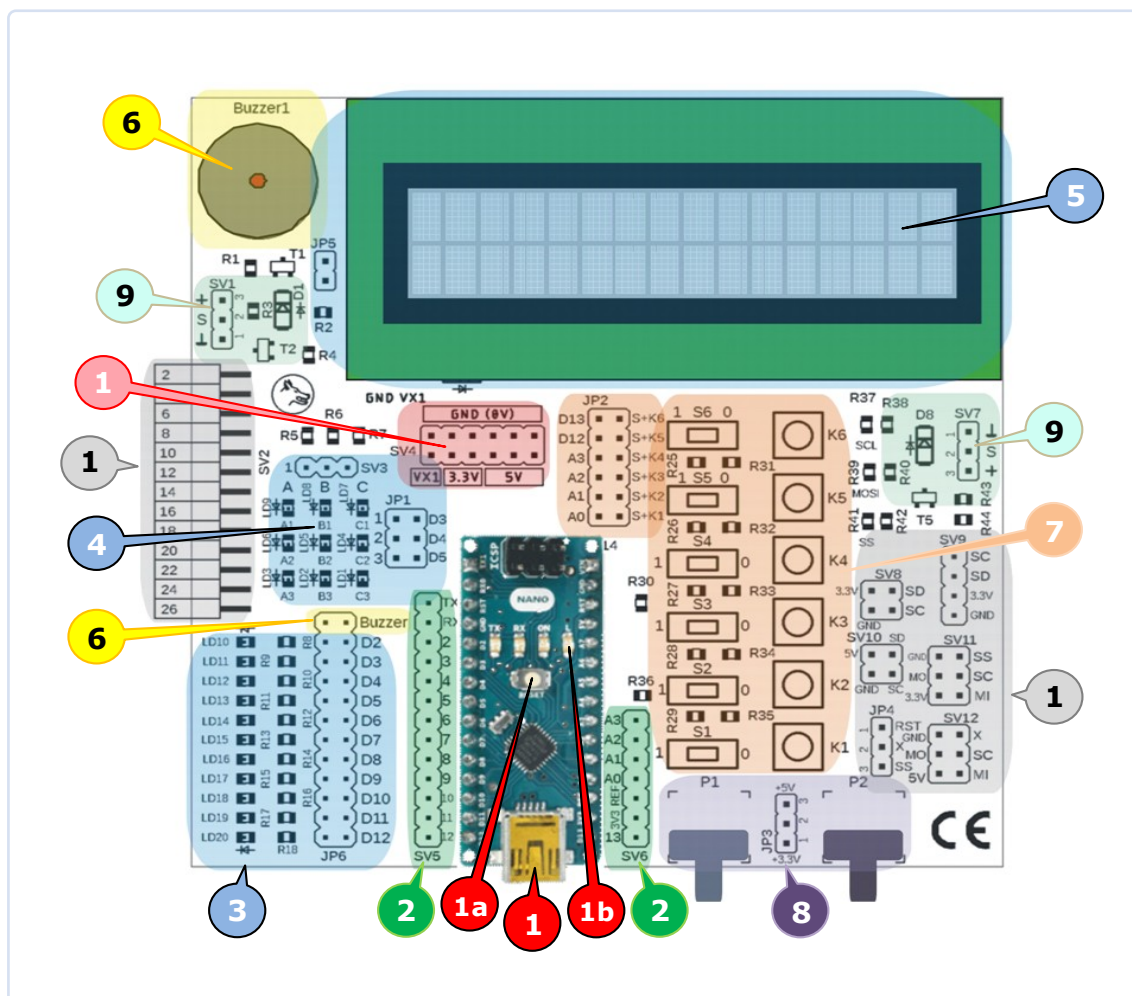
In figuur 1 kunnen we de volgende functieblokken en modules onderscheiden op de MCCAB.

(1) **Arduino Nano-microcontrollermodule** met RESET-knop (pijl 1a), LED L (pijl 1b) en miniUSB-connector voor aansluiting op je PC.

(2) Pinheaders SV5 en SV6 voor de **microcontroller-ingangen en -uitgangen**. Via deze twee pinheaders kunnen de GPIO's (General Purpose Inputs/Outputs) van de microcontroller met behulp van jumperkabels worden verbonden met de interne componenten op de kaart of met externe modules.

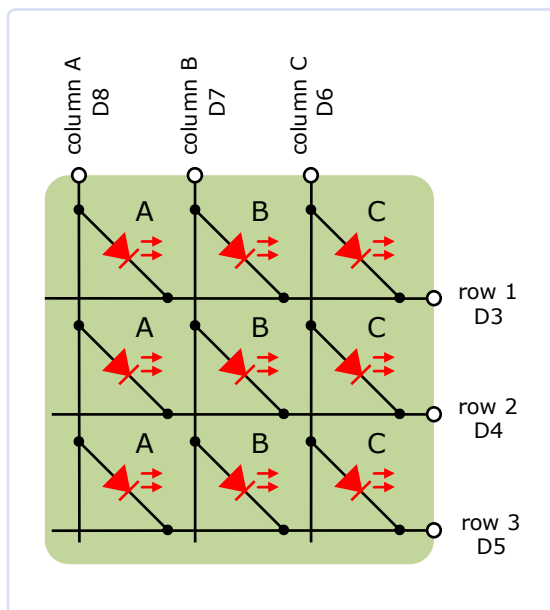
(3) **11 LED's**, LD10...LD20 (statusindicatoren voor de ingangen/uitgangen D2...D12 van de microcontroller). Elke LED kan met de toegewezen GPIO, D2...D12, worden verbonden met een jumper op header JP6.

(4) **3x3 LED-matrix** LD1...LD9 (negen rode LED's). De kolommen zijn permanent verbonden met uitgangen D6, D7 en D8 van de microcontroller, en de rijen kunnen met GPIO's D3, D4 en D5 worden verbonden met behulp van de jumpers op connector JP1.



Figuur 1. De functieblokken van de 'MCCAB' – het Elektor Arduino Nano Training Board.

Figuur 2. 3x3 LED-matrix.



Als de rijen van 3x3 LED-matrix zijn verbonden met de GPIO's D3, D4 en D5 via jumpers op JP1 of via jumperkabels met andere GPIO's, mogen die pinnen, evenals de kolomaansluitingen D6, D7 en D8, nooit worden gebruikt voor andere taken in een sketch. Dubbele toewijzing van de matrix-GPIO's zou kunnen leiden tot storingen of zelfs schade aan de MCCAB!

(5) **LC-display** met 2x16 tekens, via de I²C-bus verbonden met pinnen A4 en A5 van de microcontroller. Door JP5 te verwijderen kan voeding van het LCD worden uitgeschakeld voor oefeningen of experimenten waarbij het display niet wordt gebruikt.

(6) **Piezo-elektrische buzzer** Buzzer1 kan met GPIO D9 worden verbonden via een jumper op de positie 'Buzzer' van pinheader JP6.

(7) **6 schuifschakelaars S1...S6**, parallel geschakeld met **6 drukknoppen K1...K6**. Ze kunnen worden verbonden met de microcontroller-ingangen A0...A3 en D12...D13 via jumpers op pinheader JP2.

(8) **Potentiometers P1 en P2**, waarvan de lopers zijn verbonden met de analoge ingangen A6 en A7 van de microcontroller. De voedingsspanning van 3,3 V of 5 V kan op de potentiometers worden gezet via pinheader JP3.



Let op: de pinnen A6 en A7 van de ATmega328P zijn altijd geconfigureerd als analoge ingangen vanwege de interne architectuur van de chip. Het is niet toegestaan ze met `pinMode()` te configureren. Doe je dat toch, dan kan dat leiden tot een onjuist gedrag van het programma.

(9) Pinheaders SV1 en SV7 zijn **schakeluitgangen voor externe apparaten**.



(10) Pinheaders voor seriële verbinding met **externe SPI- en I²C-modules**.

(11) Connectorstrip SV2 met 2x13 pinnen voor het **aansluiten van externe modules**.

(12) Pinheader SV4 is het **verdeelpunt voor de voedingsspanningen van het board**. De spanningen kunnen met behulp van jumperkabels worden verbonden met interne componenten op het training-board of met externe modules.

De twee wat ingewikkelder functieblokken, de 3x3 LED-matrix (in figuur 1) en het LCD (ook in figuur 1) worden verderop nog in meer detail besproken. Zie voor alle andere de gedetailleerde beschrijving in het document *MCCAB Operating Instructions* [2].



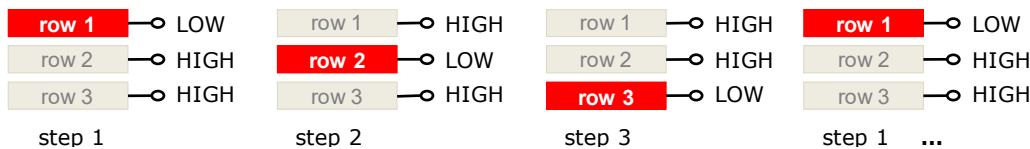
De *MCCAB_Lib*-bibliotheek is beschikbaar voor het aansturen van de 3x3 LED-matrix, de LED's LD10...LD20, de drukknoppen K1...K6, de schakelaars S1...S6 en Buzzer1. Wanneer je een MCCAB bezit, kun je de bibliotheek gratis downloaden en integreren in de Arduino IDE.

Voor het besturen van het LC-display gebruiken we de bibliotheek *LiquidCrystal_I2C*, die gratis kan worden gedownload van het internet en in de Arduino IDE kan worden 'ingevoegd'.

De 3x3 LED-matrix

De MCCAB bevat negen LED's die in een matrix zijn gerangschikt (zie figuur 1). In **figuur 2** zie je het principeschema.

De matrix bestaat uit drie rijen en drie kolommen. De kolommen heten A, B en C en de rijen zijn genummerd 1, 2 en 3. Op elk van de negen kruisingen van rijen en kolommen is een LED aangesloten. De anode is verbonden met een kolomlijn en de kathode met een rijlijn. De namen van de negen LED's corresponderen met hun positie: zo is LED 'B2' verbonden met kolom B en rij 2. Om een LED te laten oplichten, moet zijn kolomlijn logisch 1 zijn (+5 V) en zijn rijlijn logisch 0 (0 V). Volgens de handleiding voor de MCCAB zijn de kolomlijnen permanent verbonden met de GPIO's D6...D8 van de microcontroller. De rijlijnen kunnen met GPIO's D3...D5 worden verbonden via jumpers op header JP1. Als de rijlijnen van de matrix zijn verbonden met de GPIO's D3, D4 en D5 via jumpers op pinheader JP1, of met andere GPIO's van de microcontroller met behulp van jumperkabeltjes, mogen de rijlijnen en kolomlijnen D6...D8 niet worden gebruikt voor andere taken in een sketch. Dat zou kunnen leiden tot storingen of, in het ergste geval, zelfs de MCCAB kunnen beschadigen! **Als de matrix niet wordt gebruikt in een sketch, moeten de jumpers op MCCAB-header JP1 worden verwijderd.**



Figuur 3.
Achtereenvolgens
activeren van de drie
rijen van de matrix.

Voordelen van de matrixopstelling

We kennen LED-matrices bijvoorbeeld van reclameborden in voetbalstadions, waar gekleurde high power-LED's worden gebruikt in een matrixopstelling om schijnbaar bewegende afbeeldingen te genereren. Als we negen LED's individueel zouden aansturen, zijn daar negen GPIO's van de microcontroller voor nodig. Door de LED's in een matrix aan te sluiten, hebben we genoeg aan zes GPIO's. Hoe meer LED's we willen besturen, hoe groter het voordeel van matrixbesturing: Met een matrix van acht rijen en acht kolommen kunnen we 64 LED's aansturen. Bij individueel aanstuurde LED's zouden we 48 GPIO's méér nodig hebben!

De matrix aansturen in multiplexmodus

In figuur 2 zien we dat elk van de van de kolomlijnen en rijlijnen van de MCCAB is verbonden met drie LED's. We kunnen ze dus niet op de normale manier aansturen, want dan zouden er meerdere LED's tegelijk oplichten. In plaats daarvan is er op elk moment slechts één rijlijn actief (zie **figuur 3**), terwijl op dat moment de kolomlijnen actief zijn voor het bitpatroon op die rij. De beide andere rijlijnen moeten op dat moment open zijn, of een logisch-1-niveau hebben, zodat daar geen stroom kan lopen.

Als de drie rijlijnen snel genoeg om beurten worden geactiveerd, zoals in figuur 3, zien we door de traagheid van het menselijke oog, een statisch beeld van alle negen LED's. Het gebruikersprogramma bestuurt de LED-matrix in een eindeloze lus, waarin steeds één van de drie rijen 1, 2 of 3 logisch 0 wordt gemaakt, terwijl de andere twee logisch 1 zijn. De kolomaansluitingen van alle LED's die moeten oplichten in de op dat moment actieve rij worden logisch 1 gemaakt.

De LED's die uit moeten zijn in de actieve rij krijgen een logische 0 op hun kolomaansluiting.

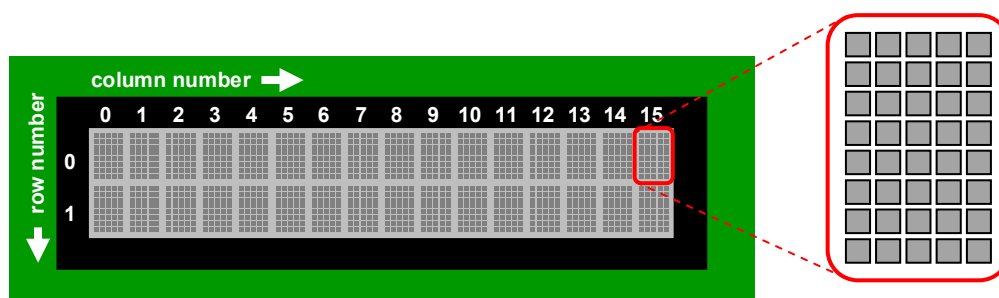
Om bijvoorbeeld de LED's A3 en C3 te laten oplichten moet rij 3 laag zijn en moeten kolommen A en C hoog zijn, terwijl rij 1 en rij 2 hoog zijn en kolom B laag.

Het Liquid Crystal Display

De Elektor MCCAB is uitgerust met een LC-display (LCD) waarmee uitvoer van tekst, numerieke waarden en ook zelf-gedefinieerde bijzondere tekens mogelijk is. Het display heeft twee regels van 16 karakters. Elk karakter wordt gevormd uit de punten van een 5×8-matrix, zoals te zien is in **figuur 4**. De benodigde 5×8-bitpatronen voor elk individueel karakter zijn opgeslagen in de LCD-controller conform de ASCII-tabel.

ASCII is een 7-bits code, terwijl onze ATmega328P microcontroller een '8bit-machine' is. Hij kan dus één byte tegelijk verwerken en slaat zijn data op in byteformaat. Het achtste bit moet altijd 0 zijn bij het opslaan van ASCII-codes. Maar de ontwikkelaars van de HD44780-displaycontroller (de standaard in deze categorie en te vinden op bijna alle LCD-modules) wilden dit achtste bit van het byte niet onbenut laten en dus hebben ze de ASCII-karakterset uitgebreid met 128 extra karaktercodes (waarbij het achtste bit logisch 1 is). Daarom gebruikt ons LCD ook de karaktercodes 128 tot 255 (bijna allemaal).

Naast de 95 afdrukbare tekens (codes 32...126) kent ASCII ook 33 besturingstekens, namelijk 0...31 en 127, die 'niet afdrukbaar' zijn. Omdat deze tekens bedoeld zijn voor besturingsdoeleinden, geeft het display ze niet weer. De HD44780-displaycontroller is verkrijgbaar in twee versies: de ene met de A00 ROM-code en de andere met de A02 ROM-code. Versie A00 geeft



Figuur 4. Opbouw van
het LC-display op de
MCCAB met zijn 16×2
weer te geven tekens.

Tabel 1. Indeling van het karaktergeheugen van de twee HD44780-varianten.

Karaktercode	ROM Code A00	ROM Code A02
0...7	gebruikergedefinieerde tekens	gebruikergedefinieerde tekens
8...15	gebruikergedefinieerde tekens	gebruikergedefinieerde tekens
16...31	(geen weergave)	zichtbare tekens
32...127	zichtbare tekens	zichtbare tekens
128...159	(geen weergave)	zichtbare tekens
160...255	zichtbare tekens	zichtbare tekens

blanco tekens voor de codes 16...31 en 128...159. Versie A02 geeft bijzondere tekens weer voor deze codes (zie **tabel 1**). Welke van de twee ROM-versies is geïnstalleerd in de MCCAB is niet te voorspellen; de fabrikant van de LCD-module maakt deze keuze.

De gebruiker kan maximaal acht bijzondere tekens definiëren en op het LCD weergeven. De ASCII-codes 0 tot 7 zijn toegewezen aan deze acht speciale tekens. Optioneel zijn dezelfde acht tekens ook te adresseren via de ASCII-codes 8...15.

In figuur 4 zie je ook de nummering van de rijen en kolommen van het display. In beide gevallen beginnen we bij 0 te tellen. Met behulp van van deze coördinaten kunnen tekens op een specifieke positie op het display worden geschreven. Het LCD heeft een cursor die aangeeft waar het volgende karakter wordt geschreven. De in het boek beschreven LCD-library bevat methoden voor het positioneren van de cursor. Hoewel het display slechts 16 tekens per regel kan

tonen, zijn er in het geheugen van de displaycontroller 40 geheugenlocaties voor elke regel. Zoals je ziet in **figuur 5** zit er een gat van 24 geheugenplaatsen tussen het laatste displayadres (39) van de eerste regel en het startadres (64) van de tweede regel.

Met behulp van *shift*-functies in de LCD-library kan het in de display zichtbare gedeelte van het geheugengebied als een venster worden verschoven over het gehele displaygeheugen.

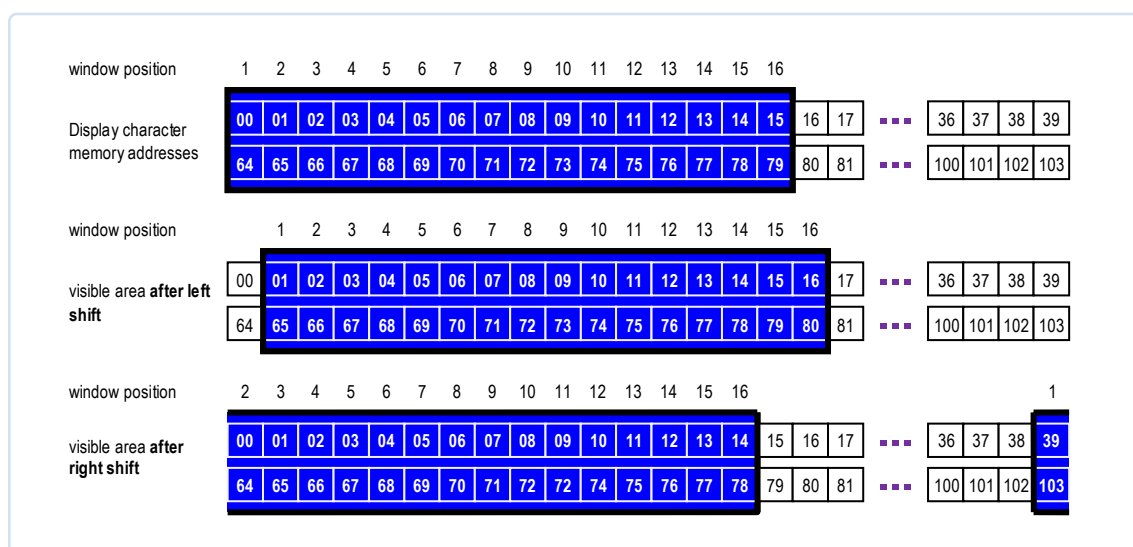
Figuur 5 bestaat uit drie gedeelten:

- boven: het zichtbare venster van het display zonder eerdere schuifoperaties;
- midden: de inhoud van het display na één keer schuiven naar links;
- onder: de inhoud van het display na één keer schuiven naar rechts.

Contrastregeling van het LCD

De MCCAB wordt geleverd met een niet-afgesteld contrast voor het LCD. Daarom en omdat het contrast van het display kan veranderen door omgevingsinvloeden (zoals temperatuur) of veroudering, heeft het LCD een instelpotentiometer voor het instellen van het contrast. Deze zit aan de onderkant van het board en is gemarkeerd met een pijl. Stel het contrast in met een kleine schroevendraaier bij de eerste keer opstarten of wanneer nodig.

i Hint: als er de eerste keer dat je het LCD gebruikt geen tekens zichtbaar zijn op het display hoewel dat wel het geval zou moeten zijn, dan ligt dat waarschijnlijk aan een verkeerde LCD-contrastinstelling!



Figuur 5. De op het display zichtbare inhoud voor en na verschuiving.

Data van de microcontroller naar het display sturen

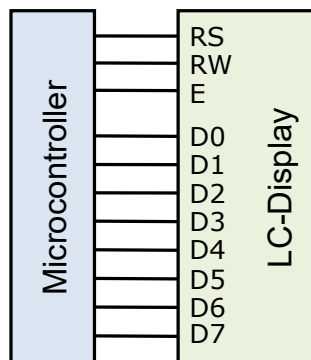
Op elke LCD module zit een HD44780-displaycontroller die de ASCII-codes van de door de microcontroller verzonden tekens ontvangt via een interface, de corresponderende tekens van de eerdergenoemde 5x8-puntmatrix genereert en die weergeeft.

De LCD-controller van het type HD44780 heeft alleen een parallelle interface voor het ontvangen van de weer te geven data. De microcontroller schrijft steeds een databyte bestaande uit acht bits, Do...D7, in het ingangsregister van de displaycontroller met behulp van de drie stuursignalen RS, RW en E (zie **figuur 6**). Deze manier van parallelle data-overdracht zou een groot deel van de resources van de MCCAB in beslag nemen, omdat het LCD alleen al 11 van de 16 beschikbare pinnen van de ATmega328P nodig zou hebben! De tweede manier voor data-overdracht naar de HD44780-controller, waarbij de acht databits worden geschreven in twee pakketjes van elk vier bit, helpt ook niet veel omdat ook in dat geval, nog steeds zeven pinnen van de microcontroller worden gebruikt door het LCD (omdat ook dan de drie stuursignalen, RS, RW en E nodig zijn).

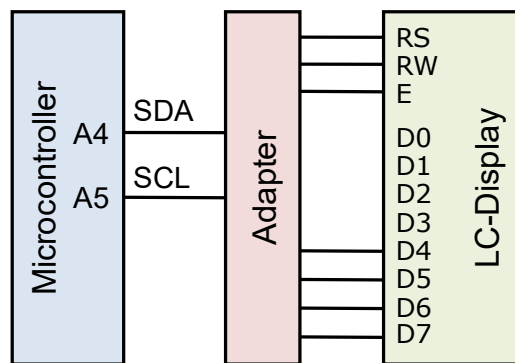
Daarom wordt de LCD-module op de MCCAB aangestuurd via de I²C-bus, een synchrone seriële bus die met slechts één datalijn (SDA) en één kloklijn (SCL) werkt en de data bit voor bit verzendt. Het dataverkeer via de I²C-interface wordt verzorgd door twee lijnen van de ATmega328P-microcontroller: A4 (SDA) en A5 (SCL) (zie **figuur 7**).

Een extra adapter aan de onderzijde van de LCD-module zet de I²C-signalen om in parallelle signalen. Deze adapter gebruikt de eerdergenoemde methode van data-overdracht in twee groepjes van vier bits via de datalijnen D4...D7. Hij stuurt dus eerst de databits D4...D7 en dan de databits Do...D3 via dezelfde lijnen D4...D7. Omdat de lijnen A4 en A5 van de microcontroller op de MCCAB toch al zijn gereserveerd voor de I²C-interface, zijn er geen extra resources nodig voor het aansturen van het display; er kunnen in principe meerdere deelnemers tegelijk worden verbonden met de I²C-bus. Omdat elke deelnemer op de bus een eigen I²C-adres heeft, 'ziet' hij alleen data wanneer die bestemd is voor zijn eigen adres. Het LCD op de MCCAB zit meestal op I²C-adres 0x27. Als de fabrikant een ander adres heeft gekozen, staat dat vermeld op het display. ◀

vertaling Evelien Snel — 230215-03



Figuur 6. LCD-aansturing met acht databits en drie stuurlijnen.



Figuur 7. LCD-aansturing via de I²C-bus in plaats van parallel (vergelijk met figuur 6).

Vragen of opmerkingen?

Hebt u technische vragen naar aanleiding van dit artikel? Stuur een e-mail naar de redactie van Elektor via redactie@elektor.com.



Gerelateerde producten

- **Microcontrollers Hands-on Course for Arduino Starters (Bundle) (SKU 20440)**
 - 1 x Guide: *Microcontrollers Hands-On Course for Arduino Starters*
 - 1 x MCCAB Arduino Training Board
 - 1 x Arduino Nano
- www.elektor.nl/20440

WEBLINKS

- [1] Elektor Arduino Nano Training Board MCCAB en Companion Guide: <http://www.elektor.nl/20440>
- [2] Handleiding voor het Elektor Arduino Nano Training Board MCCAB: <http://www.elektor.nl/20440>



Bron: Shutterstock / Metamorphics

Uit het leven gegrepen

modern luddisme

Ilse Joostens (België)

Het is een feit dat hardware steeds krachtiger wordt, waardoor ook steeds slimmere softwaretoepassingen kunnen worden ontwikkeld. Toen ik zo'n twintig jaar geleden de functie van systeembeheerder uitoefende, was ik gek op nieuwe technologie en was mijn persoonlijke missie om ook de gewone computergebruiker hiervoor te enthousiasmeren, wat uiteraard niet altijd even vlot verliep om het eufemistisch uit te drukken. Nu ik wat ouder en vooral wijzer ben geworden is mijn enthousiasme voor nieuwe technologie wat bekoeld en heb ik toch een ongemakkelijk gevoel bij de recente ontwikkelingen rond AI.

Kunstmatige domheid

Tenzij u de afgelopen maanden – zonder internet – onder een steen hebt geleefd, hebt u vast wel gehoord van de nieuwe technologische (r)evoluties rond artificiële intelligentie en meer bepaald de chatbot *ChatGPT* [1] van OpenAI. De interesse is enorm en met een investering van tien miljard dollar in deze start-up zet Microsoft volop in op generatieve

AI. Naar aloude traditie kan rivaal Google niet achterblijven en wordt er momenteel bij hen druk gesleuteld aan een gelijkaardige bot onder de welluidende naam *Bard*. Zelfs Baidu laat zich niet onbetuigd in deze strijd en die club is gestart met *Ernie*, een eigen Chinese versie van ChatGPT.

Als amateurfotograaf was mijn aandacht al eerder naar *DALL-E* [2], *StabilityAI* [3] en

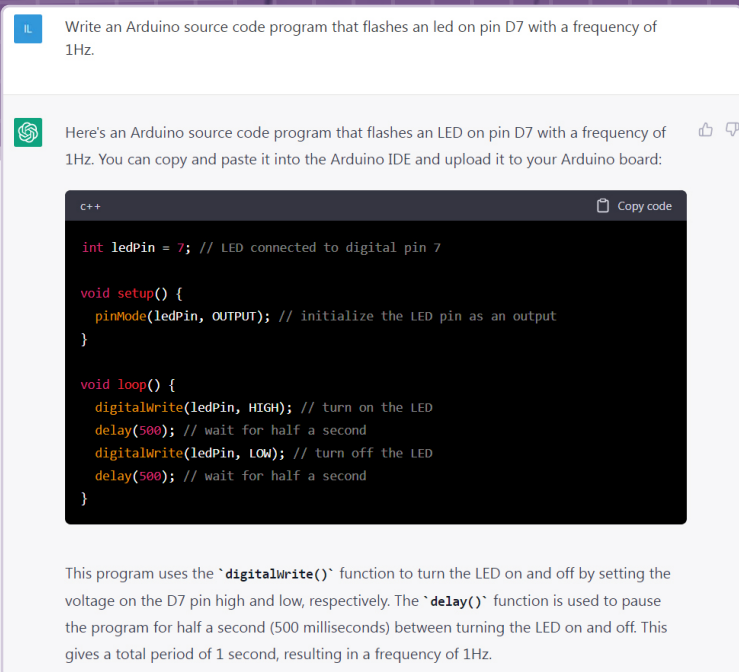
Fotor gegaan. Deze applicaties kunnen tekst omzetten naar afbeeldingen en bestaande afbeeldingen aanpassen of zelfs combineren. Het werd een beetje een verslaving en ik heb ermee zitten experimenteren tot er rook uit mijn toetsenbord kwam (**figuur 1**). De resultaten varieerden van teleurstellend tot verbluffend maar soms ook beangstigend. Wat dat laatste betreft kan mijn voorkeur voor het horrorgenre misschien een onderliggende verklaring bieden. Deze technologie staat ook centraal in de creepypasta rond 'Loab' [4][5], een demonische vrouw die zich diep in de krochten van AI verborgen zou houden. Uiteraard heb ik uit nieuwsgierigheid ook ChatGPT uitgetest en op het eerste zicht deed de bot me denken aan 'Evarist de computer', uit de Vlaamse jeugdreeks *Merlina* [6], aan wie je ook alles kon vragen. Taalkundig waren de antwoorden op mijn vragen in orde maar in tegenstelling tot Evarist sloeg ChatGPT de plank regelmatig mis met soms formidabel hilarische antwoorden in de stijl van Xavier De Baere (*een bekend Vlaams TV-humorist – red.*). Hoewel het aanvankelijk niet de bedoeling was, lijkt het er ook op dat de bot broncode kan genereren. Een eenvoudig Arduino-projectje is geen probleem (**figuur 2**) maar een complexere vraag naar een C-programma voor



Figuur 1. AI-creatie Ilse Joostens.

AVR zonder gebruik te maken van bibliotheken om een Wiznet W5500 ethernet-chip te initialiseren met een vast IP-adres liep keer op keer vast. Misschien dat de betaalde versie met de nieuwste ChatGPT-4 het beter doet maar dat is geen optie voor een applicatie waar ik zelf (nog) geen concrete toepassing voor heb. Ik krijg die chip zélf wel aan de praat en de meer doorsnee vragen des levens, hoe ik bijvoorbeeld een driegangenmenu voor vegetariërs met chocoladedessert in elkaar boks of knutselideeën voor peuters, interesseren mij maar matig.

Net zoals de 'cloud' een modewoord is voor werken en opslaan via het internet en de kreet 'IOT' met het internet verbonden apparaten zoals 'slimme' lampen betekent, is 'artificiële intelligentie' een paraplu begrip voor krachtige algoritmes en modellen gekoppeld aan big data. Voor ChatGPT-3 gaat het om 570 gigabyte aan input en DALL-E 2 moet het met meer dan 650 miljoen afbeeldingen stellen. De term 'intelligentie' is eigenlijk niet relevant en onderzoekers die neurale netwerken trainen om verbanden te zoeken die er niet zijn, creëren zelfs 'kunstmatige domheid', zo u wilt [7].



Figuur 2. ChatGPT schrijft software...

De duistere kant van AI

Ik geef toe dat generatieve AI inspireert en nieuwe inzichten geeft, maar toch ben ik niet zo'n fan aangezien de gegenereerde teksten en afbeeldingen in wezen plagiaat zijn omdat ze gebaseerd zijn op het werk van miljoenen mensen. Daarbij komt nog dat AI die getraind is op basis van historische data, welke onvermijdelijk menselijke vooroordelen bevat, deze overneemt, soms zelfs in versterkte vorm. Ik denk nog niet dat ChatGPT in de toekomst deze column gaat schrijven, maar de online verkoop van foto's en kunstwerken wordt nu wel knap lastig. Na de steeds beter wordende camera's in smartphones is dit een nieuwe klap voor de amateurfotograaf en kunstenaar die een centje wil bijverdienen [8].

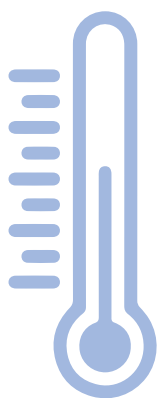
Ook de mogelijkheden tot misbruik zijn legio, gaande van taalkundig perfecte phishing-mails, tot hatespeech en deepfakes. Dankzij ChatGPT is AI nu een hot item maar het bestaat al langer en kan ook ingezet – of misbruikt – worden voor andere doeleinden,

bijvoorbeeld door overheden zoals China met hun sociaal kredietsysteem of de VS die gezichtsherkenningstechnologie inzet om demonstranten te identificeren. Ook in Vlaanderen wil men nu AI loslaten op luchtfoto's om na te gaan of iemand niet een boompje teveel in de tuin gekapt heeft. Ikzelf ben in het verleden meerdere keren slachtoffer geweest van een misdrijf, maar nooit heeft men de moeite gedaan om de daders te zoeken. Om kleine overtredingen door brave burgers vast te stellen zijn daarentegen alle middelen goed, met als triest hoogtepunt een man uit Heusden-Zolder die met zijn gezin, na een restaurantbezoek, onderweg even stopte om een volle luier van zijn ziek dochtertje in een openbare vuilnisbak te gooien. Tot zijn verbazing kreeg hij niet veel later een proces verbaal van twintig (!) bladzijden in de bus, inclusief foto's, wegens sluikstorten [9]. Het moet zijn dat echte criminelen opsluiten niet genoeg geld in het laatje brengt; je zou voor minder al een moderne luddiet worden. ◀

230196-03

WEBLINKS

- [1] ChatGPT: <https://chat.openai.com/>
- [2] Dall-E 2: <https://labs.openai.com/>
- [3] StabilityAI: <https://huggingface.co/spaces/stabilityai/stable-diffusion>
- [4] Loab, een AI-gegenereerde entiteit: <https://loab.ai/>
- [5] YouTube: The Disturbing Art of A.I. (spooky!): <https://youtu.be/i9InAbpM7mU>
- [6] Merlina: Herinneringen aan dé BRT-jeugdserie uit de jaren '80: <https://www.merlina.info/evarest/uitvindingen>
- [7] Wetenschap EOS: Kunstmatige Domheid: <https://www.eoswetenschap.eu/technologie/kunstmatige-domheid>
- [8] Youtube: AI Art Apocalypse: <https://youtu.be/HDbu2rvhCk4>
- [9] Nieuwsblad: Heusdens gezin riskeert GAS-boete voor weggooien van pamber: https://www.nieuwsblad.be/cnt/dmf20220901_96655325



Sensor-ABC: de DS18B20 temperatuursensor

aansluiting op de 1-Wire bus

Mathias Claussen (Duitsland)

De Dallas Semiconductor DS18B20-sensor zit in veel beginner-kits en biedt een gemakkelijke introductie tot het thema temperatuurmeting. Voor je erin duikt, is het handig om enige kennis te hebben van de 1-Wire bus en de aansluiting van de sensor. In dit artikel maken we een korte excursie naar de basis van het gebruik van de DS18B20, zodat je vol vertrouwen temperaturen kunt gaan meten!



Figuur 1. De DS18B20 in TO-92 behuizing.

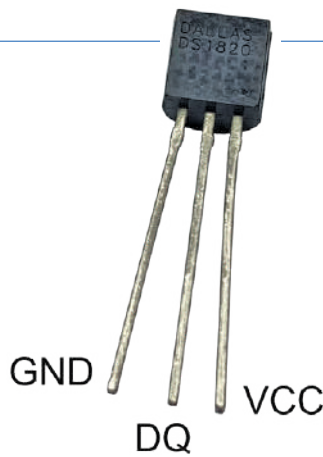


Figuur 2. Waterdichte versie van de DS18B20, met kabel.

Als je temperaturen wilt registreren met een microcontroller, kun je kiezen uit verschillende sensoren en bussystemen. Een populaire keuze is de Dallas Semiconductor DS18B20. Deze heeft een bereik van -5°C tot $+125^{\circ}\text{C}$ (-67°F tot $+275^{\circ}\text{F}$) met een indrukwekkende nauwkeurigheid van $\pm 0,5^{\circ}\text{C}$. Door zijn veelzijdigheid is hij niet alleen geschikt voor het meten van omgevingstemperaturen, maar ook voor het bewaken van vriezers en koelruimten. In dit artikel bekijken we hoe hij samenwerkt met een microcontroller, inclusief voorbeelden met broncode en schema's om te laten zien hoe je hem in je eigen projecten kunt integreren.

De DS18B20

De DS18B20-sensor maakt gebruik van het 1-Wire bussysteem, gepatenteerd door Dallas Semiconductor in 1989, en wat de aansluiting van meerdere sensoren mogelijk maakt. Met een voedingsbereik van 3,0 V tot 5,5 V kan de DS18B20 rechtstreeks worden aangesloten op GPIO's van een reeks apparaten – van de Arduino UNO tot de Raspberry Pi Pico. De 1-Wire bus biedt ook een parasitaire voedingsmodus, waarbij energie wordt onttrokken aan de datalijn om de sensor van stroom te voorzien. Dit betekent dat de 1-Wire bus slechts één pin van een microcontroller (MCU) nodig heeft voor de aansluiting van meerdere sensoren, waardoor het een populaire keuze is, vooral bij kleinere MCU-boards met een beperkt aantal GPIO's. Hoewel er andere 1-Wire sensoren bestaan, zullen we ons hier beperken tot de Dallas-component.

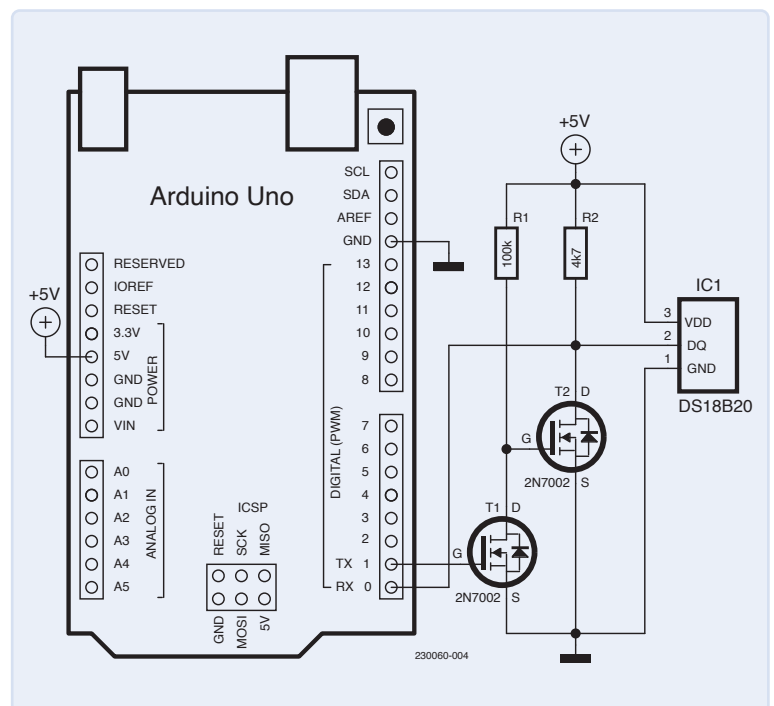


De DS18B20-sensor is verkrijgbaar in verschillende uitvoeringen, zoals te zien in **figuur 1** en **figuur 2**. Dankzij de 1-Wire bus is het bijzonder eenvoudig om meerdere sensoren aan te sluiten. Hoewel het 1-Wire protocol zelf geen grens stelt aan het aantal sensoren dat op de bus kan worden aangesloten, worden die grenzen bepaald door de elektrische eigenschappen van de bus.

Voor de aansluiting van sensoren op de 1-Wire bus zijn drie draden nodig: massa (GND), data (DQ) en voeding (VCC). De 1-Wire bus is bidirectioneel en werkt volgens een controller/target-concept (master en slave), waarbij controller en target gegevens uitwisselen via de datalijn. De pinning van de DS18B20-sensor is geschetst in **figuur 3**.

collector driver; wanneer onverhoopt gegevens botsen (als twee knooppunten per ongeluk tegelijkertijd data proberen te versturen) heeft dat slechts corrupte data tot gevolg. Er kan nooit een hardwareconflict ontstaan waarbij een knooppunt probeert de datalijn hoog te trekken terwijl een ander knooppunt probeert deze laag te trekken. Er is een pull-up weerstand nodig omdat geen van de knooppunten de bus actief naar VCC kan trekken. **Figuur 4** toont de aansluiting van een 1-draads sensor op een Arduino UNO. Naast de elektrische verbinding is een protocol nodig voor het gebruik van 1-Wire componenten. Gelukkig bevatten de meeste microcontrollers een UART die voor dit doel kan worden gebruikt, zonder dat daarvoor speciale hardware nodig is. Een geschikte schakeling is te zien in **figuur 5**.

Wanneer meer dan één sensor op een gemeenschappelijke busdraad is aangesloten, moet elke sensor afzonderlijk kunnen worden geadresseerd, zodat slechts één sensor tegelijk naar de bus kan schrijven en zijn gegevens niet door een andere sensor kunnen worden verstoord of gecorrumpeerd. Daartoe heeft elke



Figuur 5. Schema van de DS18B20 1-Wire bus, aangesloten op de UART.

8-Bit CRC

48-Bit serial number

8-Bit family code (28h)

Figuur 6. Formaat van het DS18B20 UUID-bericht.

deelnemer op de 1-Wire bus een unieke, 64bit-identificatiecode (UUID). Deze bestaat uit een 8bit-code die de sensorfamilie aangeeft, een 48bit-serienummer en een 8bit-checksum (CRC) (figuur 6). Het serienummer van de sensor staat echter nergens op de sensor, maar is intern in ROM opgeslagen in de vorm van 8 bytes. Het kan softwarematig worden achterhaald met behulp van een zoekfunctie op de 1-Wire bus. Het zoekalgoritme is te vinden op de pagina van Analog Devices [2].

Voorbeeldcode voor Arduino

Op basis van het schema in figuur 4 is de opzet zeer eenvoudig gehouden. De sensor gebruikt een 5V-voeding zoals de Arduino UNO. De Arduino UNO GPIO, aangesloten op de data- of DQ-lijn, heeft een 4,7 kΩ pull-up weerstand nodig. De bibliotheek voor de 1-Wire bus die we hier gebruiken is de OneWire-bibliotheek van Paul Stoffregen [3]. De voorbeeldcode van de bibliotheek omvat ook het lezen van gegevens van de DS18B20-sensoren (listing 1), die we nu wat meer in detail kunnen bekijken.

Om de bibliotheek te gebruiken, moet deze eerst worden opgenomen met `#include <OneWire.h>` aan het begin van de sketch. Vervolgens wordt met `OneWire ds(10)` de 1-Wire bus-datalijn toegevoegd voor gebruik met pin 10. Met `ds.search(addr)` wordt de volgende sensor op de bus gezocht. Wanneer alle sensoren zijn

geïdentificeerd, wordt het zoeken beëindigd. De identificatie van elke gevonden sensor wordt opgeslagen in `addr`.

Nu de sensor is geïdentificeerd, kan ermee worden gewerkt. De eerste byte, `addr[0]`, bevat de code die de familie aangeeft. Dit kan worden gebruikt om te identificeren of een DS18B20, DS18S20, of DS1822 temperatuursensor is aangesloten, of dat het een ander type 1-Wire buscomponent is.

In regel 69 en volgende zien we:

```
ds.reset();  
ds.select(addr);  
ds.write(0x44, 1);
```

Met `ds.reset()` worden alle busdeelnemers éénmalig gereset. Daarna wordt met `ds.select(addr)` de ontdekte sensor aangesproken. Om een temperatuurmeting van de sensor op te vragen is het noodzakelijk om eerst een meting te starten. Het betreffende commando is `ds.write(0x44, 1)`, waarbij `0x44` het in de sensor herkende commando is. Met de tweede parameter (hier `1`) wordt de I/O-pin actief hoog gemaakt om sensoren zoals de DS18S20 van parasitaire voeding te voorzien. Het resultaat van de temperatuurmeting is beschikbaar na 750 tot 1000 ms. In dit voorbeeld wordt een vertraging gebruikt om te wachten tot de verwerking voltooid is. Zodra de temperatuurmeting is voltooid en het resultaat klaarstaat, kan het uit de sensor worden opgehaald. De sensor wordt opnieuw aangesproken met `ds.select(addr)` en geïnstrueerd dat nu de in zijn scratchpad-geheugen opgeslagen waarde moet worden uitgelezen (`ds.write(0xBE)`). De negen bytes die hier zijn opgeslagen, inclusief de temperatuurmeting, worden nu uitgevoerd.

Met de DS18B20 is de temperatuur na uitlezing beschikbaar als een 16bit-waarde, bestaande uit twee registerbytes. Indien een sensor van het type DS18S20 wordt gebruikt in plaats van een DS18B20, worden de registerwaarden anders gewogen zodat hun positie in het register moet worden aangepast. Het codevoorbeeld houdt hiermee rekening door de ruwe waarde driemaal naar links te schuiven.

De DS18B20 voert de temperatuur uit met 12bit resolutie en heeft 750 ms nodig voor de conversie. Hij kan indien gewenst 11bit-, 10bit- of 9bit-waarden uitvoeren, wat de conversie navenant versnelt. Bij 9bit-resolutie is de waarde van de lage bits niet gedefinieerd, zodat deze op nul kunnen worden gezet.

De temperatuurmeting van de sensor is altijd in graden celsius; de omzetting naar graden fahrenheit moet je in software doen als dat nodig is.

Samenvattend

Een temperatuursensor kan eenvoudig op een microcontroller worden aangesloten met behulp van de 1-Wire DS18B20. Zoals we zagen, zijn er voor het Arduino-platform een heleboel bibliotheken en broncode-voorbeelden beschikbaar om het proces te stroomlijnen.

DS18B20 – The Clone Wars

De DS18B20 is een populaire sensor die vaak deel uitmaakt van diverse sensorkits en microcontroller-starterkits. Deze sensoren zijn ook erg goedkoop; je kunt er al eentje kopen voor slechts 30 cent op bepaalde online-marktplaatsen. Distributeurs zoals Mouser of Farnell vragen echter meer dan 4 euro voor een enkele DS18B20.

De populariteit van deze sensor heeft het voor sommige malafide chipsbakkers de moeite waard gemaakt om hun eigen versies te ontwikkelen, die lijken op en zich schijnen te gedragen als de originele sensor, maar bij nadere beschouwing sterk kunnen afwijken van het origineel. Erkende distributeurs betrekken hun componenten van gecertificeerde leveranciers.

Om er zeker van te zijn dat je een authentieke sensor hebt, biedt de GitHub-pagina van Chris Petrich [4] Arduino-sketches om je sensor te testen, evenals meer informatie over de verschillende klonen en hun afwijkingen van het oorspronkelijke ontwerp.



Met deze opzet is slechts één pin van de MCU nodig om met de sensor te communiceren, en kunnen meerdere sensoren worden aangesloten, zodat een kleine MCU met één vrije GPIO meerdere sensoren tegelijk kan ondersteunen. ◀

230060-03

Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de redactie van Elektor via redactie@elektor.com.



Gerelateerde producten

- > **Elektor 37-in-1 Sensor Kit (SKU 16843)**
www.elektor.nl/16843
- > **Cytron Maker UNO (SKU 18634)**
www.elektor.nl/18634
- > **MakePython ESP32 Development Kit (SKU 20137)**
www.elektor.nl/20137

WEBLINKS

- [1] Using a UART to implement a 1-wire-bus-master, Analog Devices:
<https://analog.com/en/technical-articles/using-a-uart-to-implement-a-1wire-bus-master.html>
- [2] 1-Wire zoekalgoritme, Analog Devices: <https://analog.com/en/app-notes/1wire-search-algorithm.html>
- [3] OneWireLibrary van Paul Stoffregen op GitHub: <https://github.com/PaulStoffregen/OneWire>
- [4] Chris Petrich, "Your DS18B20 temperature sensor is likely a fake, counterfeit, clone...", GitHub:
https://github.com/cpetrich/counterfeit_DS18B20



Listing 1. DS18B20 Arduino-voorbeeld.

```
001      #include <OneWire.h>
002
003      // OneWire DS18S20, DS18B20, DS1822 Temperature Example
004      //
005      // http://www.pjrc.com/teensy/td_libs_OneWire.html
006      //
007      // The DallasTemperature library can do all this work for you!
008      // https://github.com/milesburton/Arduino-Temperature-Control-Library
009
010      OneWire ds(10); // on pin 10 (a 4k7 resistor is necessary)
011
012      void setup(void) {
013          Serial.begin(9600);
014      }
015
016      void loop(void) {
017
018          byte i;
019          byte present = 0;
020          byte type_s;
021          byte data[9];
022          byte addr[8];
023          float celsius, fahrenheit;
024
025          if (!ds.search(addr)) {
026              Serial.println("No more addresses.");
027              Serial.println();
```

continued overleaf

```

028         ds.reset_search();
029         delay(250);
030         return;
031     }
032
033
034     Serial.print("ROM =");
035     for(i = 0; i < 8; i++) {
036         Serial.write(' ');
037         Serial.print(addr[i], HEX);
038     }
039
040     if (OneWire::crc8(addr, 7) != addr[7]) {
041         Serial.println("CRC is not valid!");
042         return;
043     }
044
045     Serial.println();
046     // the first ROM byte indicates which chip
047     switch (addr[0]) {
048
049         case 0x10:
050             Serial.println("  Chip = DS18S20"); // or old DS1820
051             type_s = 1;
052             break;
053
054         case 0x28:
055             Serial.println("  Chip = DS18B20");
056             type_s = 0;
057             break;
058
059         case 0x22:
060             Serial.println("  Chip = DS1822");
061             type_s = 0;
062             break;
063
064         default:
065             Serial.println("Device is not a DS18x20 family device.");
066             return;
067     }
068
069     ds.reset();
070     ds.select(addr);
071     ds.write(0x44, 1); // start conversion, with parasite power on at the end
072     delay(1000); // maybe 750ms is enough, maybe not
073     // we might do a ds.depower() here, but the reset will take care of it.
074
075     present = ds.reset();
076     ds.select(addr);
077     ds.write(0xBE); // Read Scratchpad
078
079     Serial.print("  Data = ");
080     Serial.print(present, HEX);
081     Serial.print(" ");
082     for (i = 0; i < 9; i++) { // we need 9 bytes
083         data[i] = ds.read();
084         Serial.print(data[i], HEX);
085         Serial.print(" ");
086     }
087     Serial.print(" CRC=");
088     Serial.print(OneWire::crc8(data, 8), HEX);

```




```
089     Serial.println();
090     // Convert the data to actual temperature
091     // because the result is a 16-bit signed integer, it should
092     // be stored to an "int16_t" type, which is always 16 bits,
093     // even when compiled on a 32-bit processor.
094
095     int16_t raw = (data[1] << 8) | data[0];
096     if (type_s) {
097         raw = raw << 3; // 9 bit resolution default
098         if (data[7] == 0x10) {
099             // "count remain" gives full 12 bit resolution
100             raw = (raw & 0xFFF0) + 12 - data[6];
101         }
102     } else {
103         byte cfg = (data[4] & 0x60);
104         // at lower res, the low bits are undefined, so let's zero them
105         if (cfg == 0x00) raw = raw & ~7; // 9 bit resolution, 93.75 ms
106         else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms
107         else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms
108         // default is 12-bit resolution, 750 ms conversion time
109     }
110
111     celsius = (float)raw / 16.0;
112     fahrenheit = celsius * 1.8 + 32.0;
113     Serial.print(" Temperature = ");
114     Serial.print(celsius);
115     Serial.print(" Celsius, ");
116     Serial.print(fahrenheit);
117     Serial.println(" Fahrenheit");
118 }
```

YOUR KEY TO CELLULAR TECHNOLOGY



**WÜRTH
ELEKTRONIK**
MORE THAN
YOU EXPECT

WE are here for you!

Join our free webinars on:
www.we-online.com/webinars

Adrastea-I is a Cellular Module with High Performance, Ultra-Low Power Consumption, Multi-Band LTE-M and NB-IoT Module.

Despite its compact size, the module has integrated GNSS, integrated ARM Cortex M4 and 1MB Flash reserved for user application development. The module is based on the high-performance Sony Altair ALT1250 chipset. The Adrastea-I module, certified by Deutsche Telekom, enables rapid integration into end products without additional industry-specific certification (GCF) or operator approval. Provided that a Deutsche Telekom IoT connectivity (SIM card) is used. For all other operators the module offers the industry-specific certification (GCF) already.

www.we-online.com/gocellular

- Small form factor
- Long range/worldwide coverage
- Security and encryption
- Multi-band support

#GOCELLULAR

Gaan Matter en Thread het Smart Home redden?

nieuwe standaarden om het slimme huis te vereenvoudigen

Stuart Cording (Elektor)

Vegen, drukken, selecteren, terug, selecteren, vegén – ah, de lichten zijn nu aan. Nou ja, de meeste dan. Het Smart Home is niet altijd zo slim als beloofd, dankzij te veel apps, compatibiliteitsproblemen en verschillende standaarden. Maar er is een oplossing – nog meer standaarden! Zullen Thread en Matter het verschil maken?

Wat is er beter dan na een dag hard werken thuiskomen in een huis dat helemaal klaar is om jou te ontvangen? De geluidsinstallatie heeft je favoriete muziek aanstaan, de kamers zijn warm en comfortabel, en je maaltijd staat klaar in de oven. Als je de woonkamer binnenkomt, gaan de gordijnen dicht om het directe zonlicht tegen te houden, en de staande lamp bij je comfortabele fauteuil komt tot leven zodat je verder kunt lezen in je roman op je e-reader. Die is natuurlijk ook tot leven gekomen en toont de pagina waar je gisteravond bent gebleven. Ahh, de droom van het slimme huis. Voor velen blijft het echter een droom. Hoewel er draadloze en bedrade protocollen in overvloed zijn, heeft elk zijn eigen domein, apps en functies. Het idee van huizen die reageren op onze behoeften heeft een lange geschiedenis. Eind jaren negentig

deelde Microsoft een video die veel van de huidige technologie vrij nauwkeurig demonstreerde. In een promotievideo uit die tijd tonen ze elektronische deursloten, internetwinkelen, spraakassistenten en nog veel meer, allemaal onder besturing van Windows XP en CE [1]. En deze ideeën waren ook niet nieuw. Ray Bradbury, de beroemde 20e-eeuwse science fiction-schrijver, bedacht een bijna identiek huis in zijn korte verhaal uit de jaren '50, "There Will Come Soft Rains" [2]. Maar voordat we leren hoe Thread en Matter de hedendaagse complexiteit moeten oplossen, moeten we eerst nagaan hoe we hier zijn beland.

Ontluikende draadloze korte afstand

Toen het Internet of Things (IoT) tot leven kwam, was het duidelijk dat die 'Things' draadloos verbonden moesten worden. Begin jaren 2000 waren er niet veel keuzes. Je had WiFi, een GSM-modem of Bluetooth. De eerste twee stonden niet bekend om hun geringe stroomverbruik, dus waren ze in principe niet geschikt voor batterijgevoede apparaten. Toen bleef alleen Bluetooth over, een technologie die goed aansloot bij de behoeften van de opkomende markt voor mobiele telefoons en een veelbelovend alternatief was voor IrDA, de infraroodverbinding die bij personal digital assistants (PDA's) werd gebruikt om gegevens te synchroniseren.

Vertrouwend op de 100 MHz bandbreedte rond 2,4 GHz die wereldwijd beschikbaar is als onderdeel van de industriële, wetenschappelijke en medische (ISM) band, leek Bluetooth klaar om het IoT over te nemen. In die tijd waren er internet-

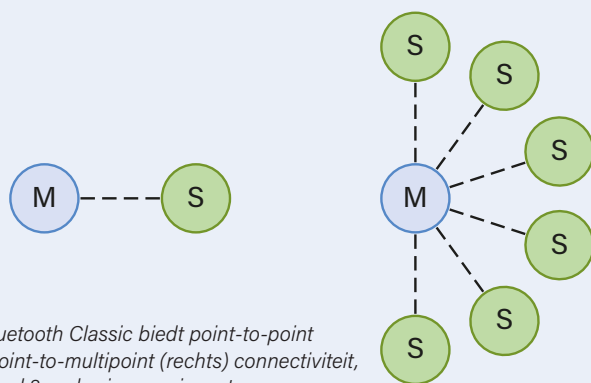
gateways, draadloze seriële verbindingen en zelfs Bluetooth-adapters voor printers. Het werd niet echt ondersteund door Windows, en vereiste software met een dongle om communicatie tussen een PC en een telefoon mogelijk te maken.

Bluetooth blonk uit in audio en bood een robuuste interface voor handsfree bellen met één-oor headsets en auto-accessoires. Autotelefoons en lompe plastic adapters behoorden tot het verleden omdat bestuurders gewoon hun telefoon met de handsfree-kit van de auto koppelden.

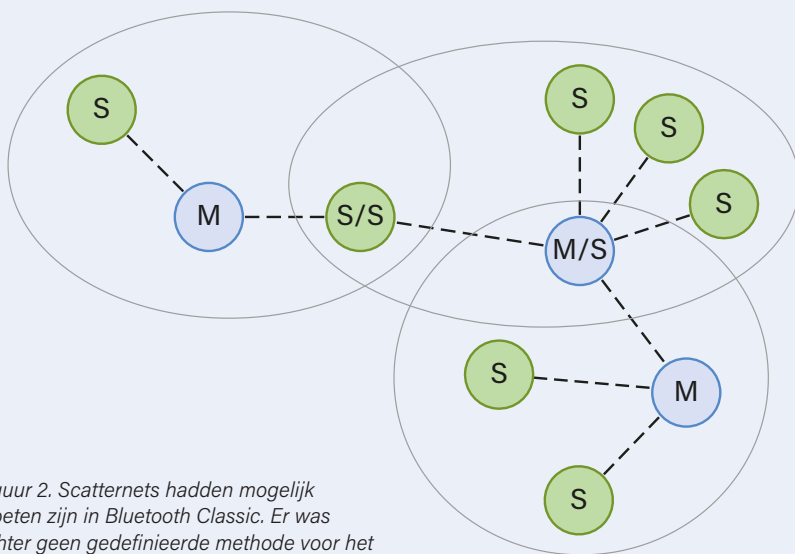
Het protocol is vanaf het begin gedefinieerd als een personal area network (PAN). Volgens de specificatie kunnen maximaal zeven slave-apparaten verbinding maken met een master om een piconet te vormen (**figuur 1**). Er was zelfs voorzien in een scatternet, gevormd door een master van een piconet die een slave is van een ander of een slave die deel uitmaakt van twee piconets (**figuur 2**). Met een destijds beschikbare verwerkingscapaciteit van slechts 720 kb/s nam de datasnelheid echter aanzienlijk af naarmate het netwerk groeide en de master/slave-rolwisseling in aanmerking werd genomen. Bovendien was er geen definitie voor de wijze waarop gegevens tussen piconets zouden worden doorgegeven – dat werd overgelaten aan de uitvoerder. De auteur was betrokken bij verschillende projecten in deze beginperiode, maar scatternets leken in de praktijk nooit mogelijk te zijn.

Stroom sparen

De noodzaak om de transceiver altijd van stroom te voorzien maakte Bluetooth ongeschikt voor batterijgevoede sensoren. Nokia had een 2,4GHz-technologie



Figuur 1. Bluetooth Classic biedt point-to-point (links) en point-to-multipoint (rechts) connectiviteit, met maximaal 8 nodes in een piconet.



Figuur 2. Scatternets hadden mogelijk moeten zijn in Bluetooth Classic. Er was echter geen gedefinieerde methode voor het doorgeven van gegevens tussen piconets.

Draadloze Personal Area Network-alternatieven met laag vermogen

Tegen die tijd waren er al alternatieve draadloze IoT-normen op de markt. Zigbee, een op IEEE 802.15.4 gebaseerde technologie, bestond al sinds 2003, en kwam in 2006 met een herziene specificatie (**figuur 3**). Deze werkte ook in de 2,4GHz-band, maar andere frequenties konden ook worden gebruikt, zoals 784 MHz (China), 868 MHz (Europa) en 915 MHz (Australië en de VS). Er was ook Z-Wave, een sub-gigahertz-technologie ontwikkeld door Zensys die uiteindelijk werd gekocht door Silicon Labs, en die ook rond 2003 van start ging.

Beide technologieën bieden zelfherstellende mesh-netwerken, wat betekent dat zelfs als een knooppunt uitvalt, het netwerkverkeer een alternatief pad vindt tussen het apparaat dat het bericht verzendt en het apparaat dat het moet ontvangen. Dit gebeurt op de achtergrond en de gebruiker van het systeem is er niet bij betrokken. Beide draadloze systemen worden momenteel door duizenden producten gebruikt.

Problemen met laagvermogen-netwerken

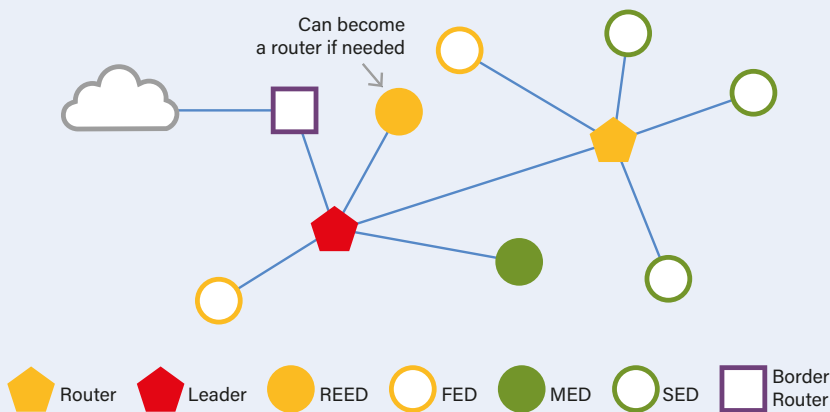
Er zijn echter verschillende problemen met al deze technologieën. Ten eerste maakt geen enkele van nature verbinding met het internet als je een smart home-systeem wilt implementeren. Een Bluetooth-apparaat kan communiceren met cloudservices als er een smartphone in de buurt is, maar anders hebben ze allemaal een hardware-gateway nodig die weer verbinding maakt met een op internet aangesloten router. Ten tweede zijn ze niet van nature interoperabel. Misschien kies je Zigbee als je favoriete smart home-technologie, om er vervolgens achter te komen dat een product dat je wilt gebruiken alleen beschikbaar is met Z-Wave. Het is mogelijk om manieren te ontwikkelen om deze kloof te overbruggen met Node-RED [5], maar dit is voor de meeste gebruikers nauwelijks geschikt. Ten derde kan de controle van de apparaten op het netwerk onpraktisch zijn. Vaak zijn meerdere apps nodig, allemaal met verschillende bedieningsmethoden. Ten slotte kan het koppelen van spraakassistenten aan deze systemen een uitdaging vormen. Amazon Alexa ondersteunt veel meer apparaten dan Google Home, dat op



Figuur 3. Het Ikea TRÅDFRI-assortiment van smart home-apparaten, zoals schakelaars en gloeilampen, maakt gebruik van Zigbee.

ontwikkeld met de naam Wibree [3] om dit aan te pakken. Tegen het midden van de jaren 2000 boden dual-mode chipsets Bluetooth en Wibree naast elkaar, gericht op slimme horloges, sportsensoren en zelfs draadloze toetsenborden. In 2007 werd Wibree overgenomen door de mensen die de Bluetooth-specificatie onderhielden, bekend als de Bluetooth SIG, en werd uiteindelijk Bluetooth Low Energy (LE). Bluetooth

Classic en LE bevinden zich tegenwoordig in dezelfde chipset, delen een antenne en radiozenderontvanger, en hebben overeenkomsten in hun softwarestack. Ze zijn echter niet compatibel met elkaar [4]. Ondanks de low-power prestaties die geschikt zijn voor batterijgevoede IoT-sensoren, was het pas in 2017 dat mesh-netwerkmogelijkheden voor Bluetooth LE beschikbaar kwamen.



Figuur 4. Thread biedt een zelfherstellend netwerk dat, in tegenstelling tot de alternatieven, gebruik maakt van de IPv6-standaard 6LoWPAN.

Figuur 5. Matter-oplossingen zijn beschikbaar bij Espressif, zoals in deze demo op embedded world 2023.



Figuur 6. Silicon Labs demonstreerde ook Matter dat met zowel Zigbee- als Z-Wave-netwerken werkt.



zijn beurt compatibel is met meer apparaten dan Apple HomeKit [6]. Om enkele van deze problemen aan te pakken, hebben de technici van de industrie gedaan wat ze het beste kunnen – een andere technologie introduceren: Thread.

Thread brengt alles samen

De Thread Group Alliance werd opgericht in 2014 en heeft zich geconcentreerd op het bouwen van een low-power draadloze netwerktechnologie met geringe latentie die concurreert met de al genoemde oplossingen. Deze werkt ook op 2,4 GHz, maar onder-

scheidt zich op verschillende manieren. Ten eerste gebruikt het Internet Protocol-communicatie (IP) – meer bepaald een versie van IPv6 genaamd 6LoWPAN. Hiermee wordt IP aangepast aan de behoeften van apparaten met een laag vermogen en de gebruikte IEEE 802.15.4 draadloze netwerken. De toegepaste mechanismen omvatten fragmentatie, reassembly en compressie die in IPv6 niet nodig zouden zijn. Ten tweede is de beveiliging niet alleen robuuster maar ook verplicht. Gegevens worden end-to-end versleuteld, net als in elk ander TCP/IP-netwerk, wat betekent dat

apparaten die als routers op het netwerk fungeren de inhoud van de gegevens die via hen gaan niet kunnen inspecteren. Ten slotte is, in vergelijking met de alternatieven, de mesh-latentie als data door een netwerk 'hopt' ongeveer de helft van die van Zigbee en ongeveer zeven keer lager dan Bluetooth [7].

Thread-knooppunten

De knooppunten van een Thread-netwerk bestaan uit twee soorten apparaten [8]: een Full Thread Device (FTD) of een Minimal Thread Device (MTD). MTD's kunnen alleen End Devices (ED) zijn, waarvoor een Router nodig is om toe te treden tot het Thread-netwerk. Een enkele router kan tot 511 ED's ondersteunen. Er zijn twee soorten MTD's. Het Minimal End Device (MED) heeft een always-on transceiver, terwijl het Sleepy End Device (SED) zijn transceiver periodiek laat controleren op berichten. SED-knooppunten, zoals sensoren, zullen doorgaans op batterijen werken.

FTD's hebben hun transceivers altijd ingeschakeld en zijn er in twee smaken. De eerste is de Router, die de transceiver ingeschakeld laat om nieuwe knooppunten die tot het netwerk toetreden te ondersteunen en pakketten door te sturen. Het eerste knooppunt dat de rol van Router op zich neemt, wordt de Leader genoemd en beheert toekomstige knooppunten die aan het netwerk worden toegevoegd en die de rol van Router op zich nemen. Er kan slechts één Leader zijn en maximaal 32 Routers. Als de Leader faalt of wordt verwijderd, neemt een van de andere Routers de rol over. Indien de rol van Router niet nodig is (omdat er voldoende Routers in het netwerk zijn), kan de FTD een Router Eligible End Device (REED) worden. Hij functioneert als een ED totdat zijn routerfunctie nodig is, waarna hij wordt bevorderd.

Het tweede is het Full End Device (FED). Deze FTD's kunnen alleen functioneren als ED's, maar houden meer gegevens over het netwerk bij dan MED's (zoals multicasting en IPv6-adresmapping). Dit profiel is geschikt voor apparaten met netvoeding. Met deze onderdelen is de Thread-mesh volledig operationeel. Voor de koppeling met clouddiensten is echter nog één element nodig – één of meer Border Routers. Dergelijke apparaten verbinden het Thread-netwerk met Ethernet of WiFi (figuur 4).

Ze kunnen worden geïntegreerd in huishoudelijke apparaten zoals slimme tv's, WiFi-routers en spraakassistenten die al een internetverbinding hebben.

Het belangrijkste voordeel hiervan is dat Thread-apparaten veilig IP spreken, net zoals alle andere Ethernet-, WiFi- en mobiele apparaten dat doen. Ter vergelijking, Zigbee, Z-Wave en Bluetooth moeten alle internet/cloud-communicatie vertalen naar eigen draadloze netwerkprotocollen. Een ander belangrijk verkoopargument is dat de application layer, waar de smartphone-apps en webinterfaces meestal op aansluiten, agnostisch is. Dit betekent dat, in theorie, de interoperabiliteit tussen Thread-geschiedte apparaten en thuisassistenten van verschillende fabrikanten zou moeten verbeteren. Het ondersteunt ook nog een andere nieuwe technologie – Matter.

Interoperabiliteit van apparaten beheren

Matter is al een paar jaar in de maak en stond aanvankelijk bekend als Project Connected Home over IP, of 'CHIP'. Een van de deelnemers is de Connectivity Standards Alliance [9] (CSA), voorheen de Zigbee Alliance, die Matter-producten certificeert naast hun bestaande verantwoordelijkheid voor op Zigbee gebaseerde apparaten. Omdat het op de TCP/IP-transportlayer zit, is het ook klaar om te werken met Thread en elk Ethernet- of WiFi-apparaat in huis.

Matter is een softwarelayer die bepaalt hoe apparaten met elkaar communiceren, waardoor elk smart home-systeem of spraakassistent kan worden gebruikt. Maar het grootste voordeel is misschien wel Multi-Admin, waarmee verschillende apps tegelijkertijd met apparaten kunnen worden gebruikt. Een lamp kan bijvoorbeeld worden bediend door zowel een wandschakelaar als de door de fabrikant geleverde app op je smartphone, maar kan ook worden verbonden met en bediend door de spraakassistent waar je voor hebt gekozen. Dit betekent ook dat een spraakassistent van Google in staat zou moeten zijn om Amazon- of Apple-apparaten te bedienen. Er is echter geen garantie hoe ver de compatibiliteit in werkelijkheid zal reiken. Zoals Wired's Simon Hill opmerkt in zijn artikel over Matter, is er nog steeds de mogelijkheid dat de bediening van geavanceerde functies of instellingen

het gebruik van een apparaat of app uit hetzelfde ecosysteem vereist [10].

Verschillende halfgeleiderfabrikanten hebben demo's van Matter die werken met Thread- en WiFi-apparaten, zoals Espressif [11] en Silicon Labs [12] (**figuur 5** en **figuur 6**). Momenteel ondersteunt de specificatie de meeste Smart Home-basisapparaten, van lampen en stopcontacten tot sloten en thermostaten, en de grote spraakassistent-spelers doen ook mee. Witgoed, robotstofzuigers en camera's staan op de lijst voor de volgende iteratie van de specificatie [13].

Zal het wat uitmaken?

Dus, na een decennium van Smart Home IoT, bevinden we ons nog steeds in een soort VHS-versus-Betamax strijd [14] die steeds complexer wordt in plaats van minder complex. De sleutel lijkt Thread te zijn, dat het aloude probleem met Zigbee, Z-Wave en Bluetooth Mesh oplost, namelijk dat ze een gateway nodig hebben die IP vertaalt naar het protocol dat deze draadloze netwerken gebruiken. Matter levert het smeermiddel die Thread zinvol maakt, zodat gebruikers, en niet de consumentenelektronica-giganten, kunnen beslissen welke apps en spraakassistenten ze gebruiken om hun apparaten te bedienen.

Het biedt ook een manier om ervoor te zorgen dat bestaande installaties niet van de ene op de andere dag verouderd raken. Uiteraard kan het alleen interfacing bieden tot aan de gateways van Zigbee en consorten, niet in de netwerken zelf, maar dit zou al moeten helpen. En omdat Matter WiFi- en Ethernet-apparaten ondersteunt, zou dit de factor kunnen zijn die het tot een succes maakt. Fabrikanten van hoogwaardige, lang houdbare consumptiegoederen, zoals wasmachines en keukenapparaten, weten dat bijna al hun klanten WiFi hebben, zodat ze niet hoeven in te zetten op een van de andere drie. Zal Matter er over tien jaar toe doen? Het lijkt zeker meer gericht op de behoeften van de gebruiker, dus de kans is groot. Maar zoals gewoonlijk zal alleen de tijd het leren. ◀

vertaling Jelle Aarnoudse — 230226-03

Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur via stuart.cording@elektor.com of naar de redactie van Elektor via redactie@elektor.com.

WEBLINKS

- [1] "Microsoft Smart Home," Microsoft, 1999 [YouTube]: <https://bit.ly/3zoDVCv>
- [2] "There Will Come Soft Rains (short story)" [Wikipedia]: <http://bit.ly/3ZR73x7>
- [3] E. Grabianowski, "Is Wibree going to rival Bluetooth?" HowStuffWorks: <http://bit.ly/3nHHTnb>
- [4] M. Afaneh, "Bluetooth vs. Bluetooth Low Energy: What's the Difference?," April 2022: <http://bit.ly/3KvmiaA>
- [5] R. Dullier, "Control a Z-Wave plug using a Zigbee button!," March 2021: <http://bit.ly/3nHRdHB>
- [6] M. Timothy, "Amazon Alexa vs. Google Home vs. Apple HomeKit: What's the Best Smart Home System?," MakeUseOf, March 2023: <http://bit.ly/40C3wDY>
- [7] "Benchmarking Bluetooth Mesh, Thread, and Zigbee Network Performance," Silicon Labs: <http://bit.ly/3Ge74UF>
- [8] "Node Roles and Types," Google, February 2023: <http://bit.ly/3m5ZmVn>
- [9] Connectivity Standards Alliance (CSA) website: <http://bit.ly/3nF6qcm>
- [10] S. Hill, "Here's What the 'Matter' Smart Home Standard Is All About," Wired, October 2022: <http://bit.ly/3zrpWf7>
- [11] Espressif's Solution for Matter: <http://bit.ly/3zsb545>
- [12] Video: Matter over Wi-Fi and Thread Demo - Silicon Labs: <http://bit.ly/3nFZ2gZ>
- [13] J. P. Tuohy, "We're getting our first look at Matter devices today, and here's what's coming next," The Verge, November 2022: <http://bit.ly/410zh9D>
- [14] D. Owen, "The Betamax vs VHS Format War," MediaCollege.com, May 2005: <http://bit.ly/3U78JRD>

Een kwestie van samenwerking

ontwikkelen met het Thing Plus Matter Board en Simplicity Studio

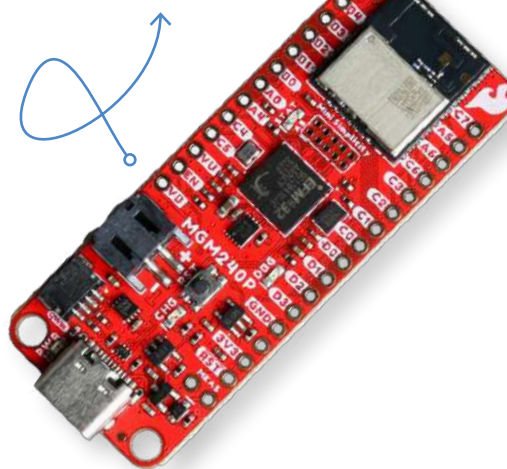
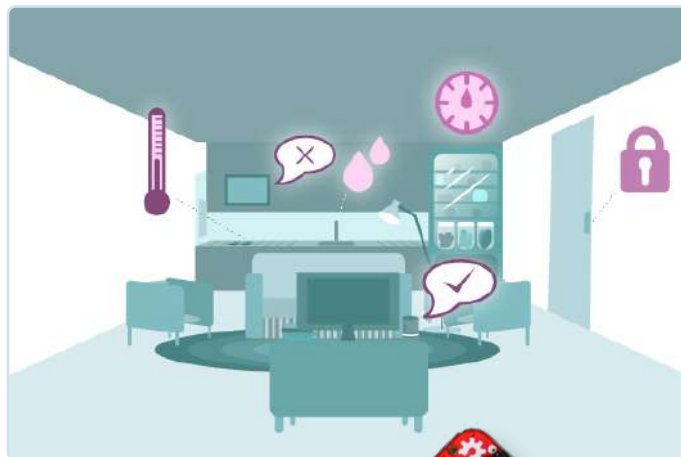
Rob Reynolds (SparkFun)

Als je met domotica wilde beginnen, moest je tot nog toe eerst een ecosysteem kiezen. Dat behoort nu echter tot het verleden, omdat Matter elk IoT-apparaat gemakkelijk wil laten communiceren via dit nieuwe open-source protocol. In dit artikel ontwikkelen we een kleine Matter-compatibele demotoepassing met het nieuwe SparkFun Thing Plus Matter Development Board en de Simplicity Studio IDE van Silicon Labs.

Inmiddels hebben we allemaal van het Internet of Things gehoord (het IoT, zoals het meestal wordt genoemd). Maar het is nog altijd tamelijk gefragmenteerd, met meerdere communicatieprotocollen. Dit dwingt zowel ontwikkelaars als gebruikers een keuze te maken hoe hun apparaten moeten communiceren, en zich vervolgens in die omgeving vast te leggen. Daar komt nu echter een einde aan met de introductie van Matter, een uniforme, open-source application-layer verbingsstandaard die is ontwikkeld om ontwikkelaars en fabrikanten van apparaten in staat te stellen betrouwbare en veilige ecosystemen te bouwen en daar verbinding mee te maken, en de compatibiliteit tussen aangesloten thuisapparatuur te vergroten.

Een korte geschiedenis van Matter

Matter begon in 2019 als Project CHIP, wat staat voor *Connected Home over IP*. Grote (en aanvankelijk concurrerende) spelers zoals

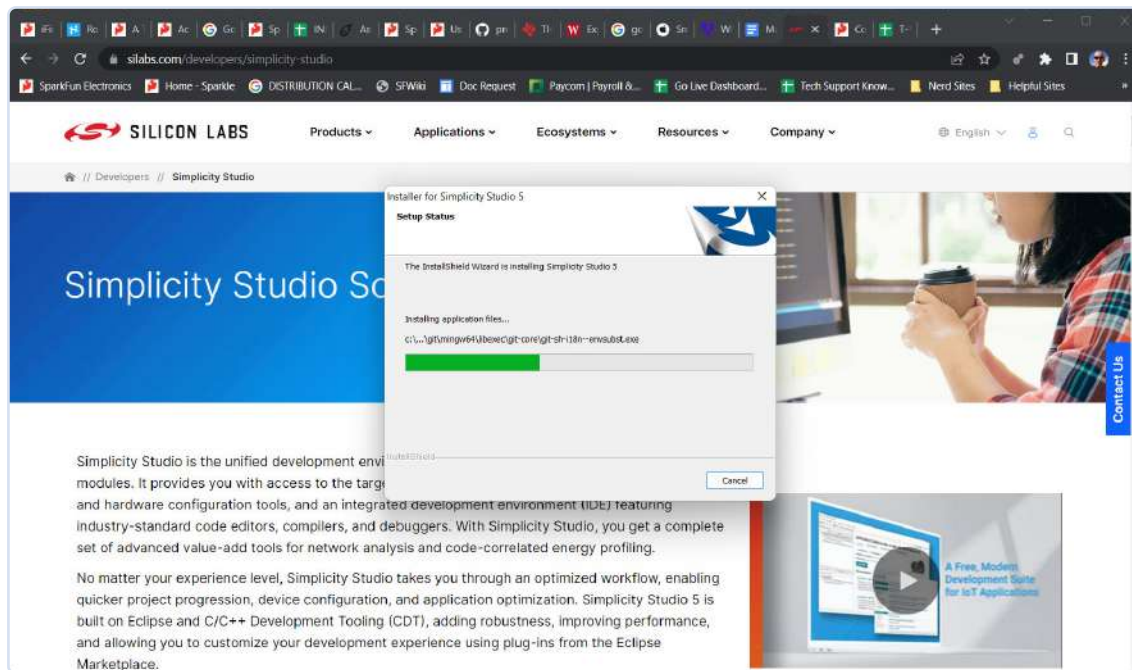


Amazon, Apple, Google deden samen met de Zigbee Alliance en een aantal andere bedrijven zoals Nordic Semiconductor, een poging om één enkel communicatieprotocol te ontwikkelen waarmee het hele Internet of Things geünificeerd zou kunnen worden. Matter is een open-source, royalty-vrij protocol waarmee apparaten kunnen communiceren via WiFi-, Ethernet-, Bluetooth Low Energy- en Thread-netwerken. Dat betekent dat apparaten die Matter-gecertificeerd zijn naadloos met elkaar kunnen communiceren, ongeacht de gebruikte draadloze technologie.

Nu hoeven ontwikkelaars, fabrikanten en consumenten niet langer te kiezen tussen HomeKit van Apple, Alexa van Amazon of Weave van Google. Voor fabrikanten betekent dit eenvoudiger ontwikkeling en voor consumenten meer compatibiliteit.

Een van de belangrijkste voordelen van Matter is dat het de setup en het beheer van smart home-apparaten vereenvoudigt. Met Matter-gecertificeerde apparaten zijn eindgebruikers in staat om hun smart home-systemen snel en gemakkelijk op te zetten, zonder de noodzaak van gespecialiseerde kennis of technische vaardigheden. En omdat veiligheid van het grootste belang is, ondersteunt het protocol ook end-to-end versleuteling, zodat de gegevens die tussen apparaten worden verzonden veilig zijn.

Een ander belangrijk voordeel is voor de meesten van ons dat Matter volledig open-source is. Alle informatie van Matter is beschikbaar in hun GitHub-repository [1], inclusief broncode, documentatie,



Figuur 1. Installatie van Simplicity Studio.

scripts, voorbeelden en alles wat een ontwikkelaar nodig heeft om Matter-compatibele componenten te bouwen.

Dat is natuurlijk heel mooi en prachtig, maar de meesten van ons zijn voorbestemd om de rol van consument spelen, zonder enige mogelijkheid om prototypes van componenten te maken zonder helemaal vanaf nul te beginnen. Gelukkig komt daar verandering in met de recente release van het SparkFun Thing Plus Matter Development Board van SparkFun Electronics [2], dat verkrijgbaar is in de Elektor-shop (zie het kader **Gerelateerde Producten**). Dit is het eerste gemakkelijk toegankelijke board in zijn soort dat Matter en het Qwiic-ecosysteem van SparkFun combineert voor vlotte ontwikkeling en prototyping van Matter-gebaseerde IoT apparaten. De MGM240P draadloze module van Silicon Labs biedt veilige connectiviteit voor zowel 802.15.4 met Mesh-communicatie (Thread) als Bluetooth Low Energy 5.3-protocollen. De module is voorbereid voor integratie in het Matter IoT-protocol voor huisautomatisering van Silicon Labs. SparkFun's Thing Plus ontwikkelboards zijn Feather-compatibel en bezitten een Qwiic-connector voor eenvoudige integratie in het Qwiic Connect System voor soldeerloze I²C-schakelingen.

Aan de slag met Simplicity Studio

Om met Matter onze eerste stappen als IoT-ontwikkelaar te zetten, moeten we beginnen met het aansluiten van het Thing Plus Matter Board op een Google Nest Hub. Maar eerst installeren we Simplicity Studio van Silicon Labs. We geven hier een summiere beschrijving; een veel uitgebreidere tutorial kun je vinden op de website van SparkFun [3].

Als je naar de website van Silicon Labs gaat, kun je de meest recente versie (Simplicity Studio 5 op het moment van schrijven) voor je besturingssysteem downloaden. Als je op de installatieknop voor je

specifieke besturingssysteem klikt, kom je op een inlogpagina. Als je nog geen account hebt, moet je dat nu aanmaken, want daar wordt bij het openen van Simplicity Studio naar gevraagd [4]. Zodra je het programma hebt gedownload, voer je het installatieprogramma uit, zoals getoond in **figuur 1**.

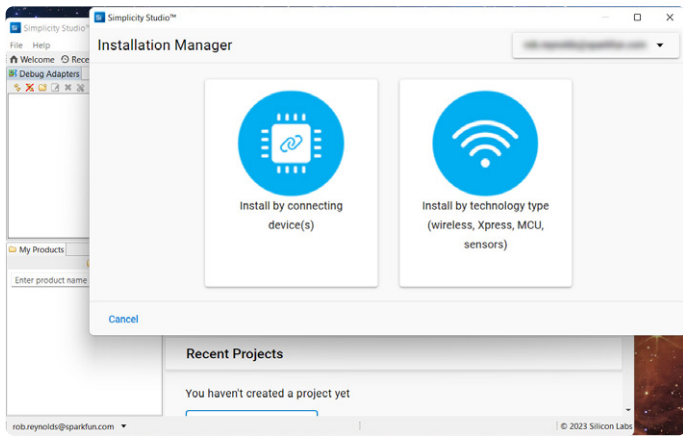
Nadat je Simplicity Studio hebt geïnstalleerd en voor de eerste keer uitvoert, kom je in de Installation Manager, die naar eventuele updates zoekt en deze voor zover nodig voor je opsoemt. Hier kun je gewoon op *Update All* klikken om ervoor te zorgen dat je de laatste versies hebt van alles wat Simplicity Studio nodig heeft om te draaien. Als er geen updates zijn, of je systeem is geconfigureerd voor automatisch bijwerken, zal de Installation Manager meteen naar de volgende stap gaan.

Zodra Simplicity Studio opnieuw opstart (als een herstart nodig was), keer je terug naar de Installation Manager, maar deze keer wordt je gevraagd of je je apparaat/apparaten wilt installeren door ze aan te sluiten, of dat je ze wilt installeren per technologie-type (draadloos, Xpress, MCU, sensoren). We kiezen hier voor *Install by connecting devices* (**figuur 2**) en sluiten we ons SparkFun Thing Plus Matter-board aan via USB.

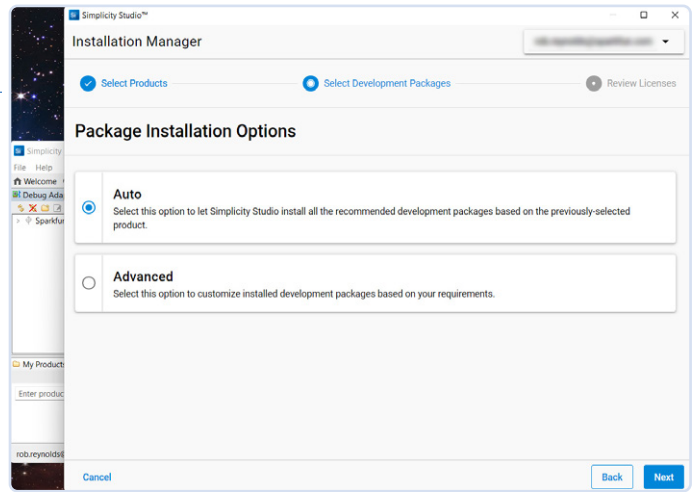
Er wordt gevraagd of je de vereiste packages wilt installeren, wat we natuurlijk doen; klik dus op *Yes*. Daarna zou de Installation Manager "1 Device Found" moeten melden, met een naam als

☒ J-Link (000449050174) (ID: 000449050174)

Vink het vakje aan en klik op *Next*, en je komt bij de Package Installation Options. Door *Auto* te selecteren worden alle benodigde packages geïnstalleerd, met alle toeters en bellen die we willen. Er is ook een *Advanced*-optie waar je precies kunt kiezen welke packages je wel en welke niet wilt installeren, maar als je op dat



Figuur 2. Installeren door apparaten aan te sluiten in Simplicity Studio.



Figuur 3. Package Installation Options.

niveau bezig bent, heb je dit artikel waarschijnlijk niet nodig. Kies daarom Auto en dan Next, zoals in **figuur 3**. Je krijgt een Master Software License Agreement voorgeschoteld die je moet accepteren, en dan begint de installatie. Dit neemt enige tijd in beslag, met aan het eind een herstart, dus misschien is dit een goed moment om wat versnaperingen te halen.

Wanneer je terug bent en Simplicity Studio weer opstart, word je begroet door een andere EULA, en dan, eindelijk, komen we bij het Welcome to Simplicity Studio-scherm! Onder Connected Devices moet je de SparkFun Thing Plus MGM240P zien. Als je op Start klikt, verschijnt de Thing Plus-informatiepagina, waar je een overzicht van het board ziet plus voorbeeldprojecten en demo's, documentatie en compatibele tools. Als je naar het tabblad Example Projects and Demos gaat, kun je in het filterveld het sleutelwoord "Blink" intypen. Dit zou een aantal resources moeten opleveren. Zoek Platform – Blink Bare-metal en klik op de knop Create (**figuur 4**).

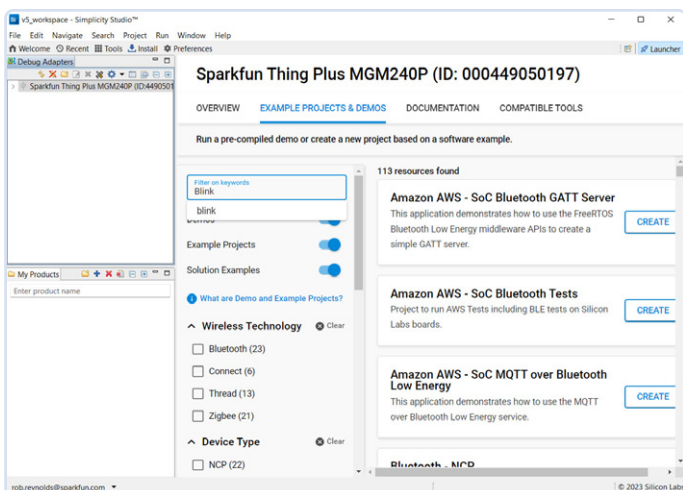
Dit opent een venster waarin je de naam of locatie van het bestand kunt wijzigen. Ik stel voor het bestand een andere naam te geven, zoals FirstBlinkDemo, en vervolgens op Finish te klikken. Zodra het project is gebouwd, zie je aan de linkerkant van je Simplicity Studio-app een Project Explorer-venster. Zoek naar de hoofd-projectmap, die MatterBlinkExample zou moeten heten. Klik daar met de rechter-

muisknop op en navigeer vervolgens naar Run as/1 Silicon Labs ARM Program (zie **figuur 5**).

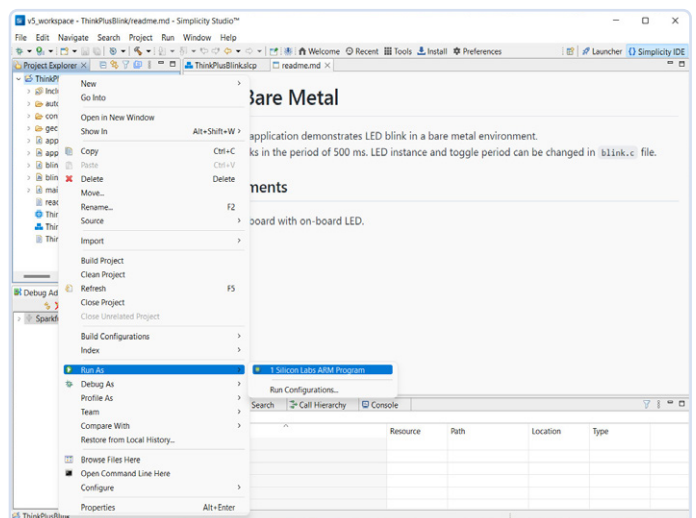
Door daarop te klikken wordt de sketch gecompileerd en naar het Thing Plus Matter-board geflashed. De blauwe LED van het board zou nu met tussenpozen van een halve seconde moeten knipperen. Nu is de kans is groot dat toen je je Thing Plus-board voor het eerst via USB aansloot, die LED al begon te knipperen met tussenpozen van een halve seconde – voordat je ook maar iets had gedaan. Om te controleren of je code daadwerkelijk naar het board wordt geflashed, kun je het knipperinterval wijzigen door naar het bestand `blink.c` in de Project Explorer te gaan en het interval te wijzigen. Ergens in de buurt van regel 31 (**figuur 6**) zou je dit moeten vinden:

```
#define TOGGLE_DELAY_MS 500
```

Verander gewoon de waarde in iets dat duidelijk maakt dat de LED in een ander tempo knippert. Verander de waarde in 100 voor snel knipperen of 3000 voor langzaam knipperen – dan weet je zeker dat het board met jouw code wordt geflashed. Zodra je die waarde hebt veranderd, kun je opnieuw met de rechtermuisknop op de map MatterBlinkExample klikken, naar beneden navigeren naar Run as/1 Silicon Labs ARM Program, dat aanklikken, en dan zien hoe de blauwe LED van het board met een andere frequentie knippert.



Figuur 4. Het tabblad Example Projects and Demos.



Figuur 5. Voer het ARM-programma van Silicon Labs uit.

Aansluiten van Thing Plus Matter op Google Nest Hub (tutorial).
<https://www.elektormagazine.com/portal/ep/article/6428>



Het board verbinden met Google Nest Hub

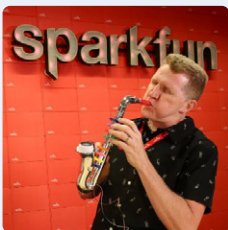
Gefeliciteerd, je communiceert nu met je Thing Plus Matter-board via Matter in combinatie met Silicon Labs Simplicity Studio. Nu wordt het pas echt interessant. Door te communiceren met Google Hub kun je je eigen *custom builds* integreren in je Smart Home, met boards zoals dit SparkFun Thing Plus Matter-board (zie **figuur 7**). Voor wie nu zin heeft gekregen om verder te gaan, heeft SparkFun Engineer Drew samen met Creative Technologist Mariah een video en tutorial samengesteld waarin precies wordt uitgelegd hoe je dat doet [5].

Omdat deze technologie heel erg nieuw is, zijn voorbeelden en tutorials nog vrij beperkt, maar het aantal groeit elke dag. Door nu aan de slag te gaan met Matter krijg je een grote voorsprong op de rest, aangezien dit unificerende protocol alle platforms omvat en helpt de smart home-industrie te verenigen. 

230224-03

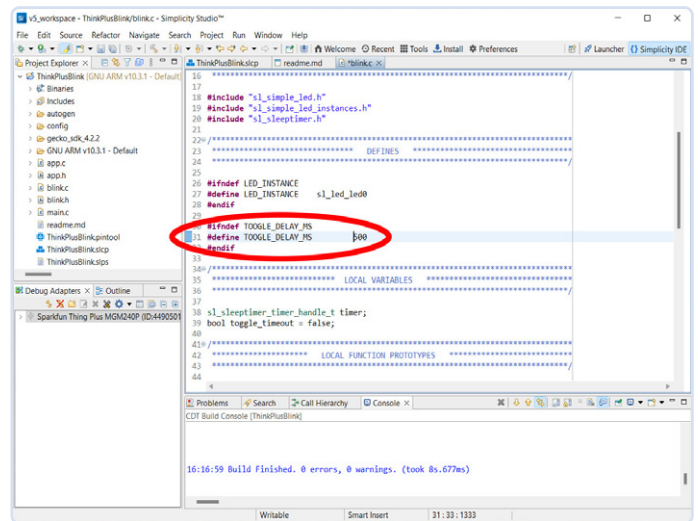
Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar het SparkFun Support-team via support@sparkfun.com of naar de redactie van Elektor via redactie@elektor.com.

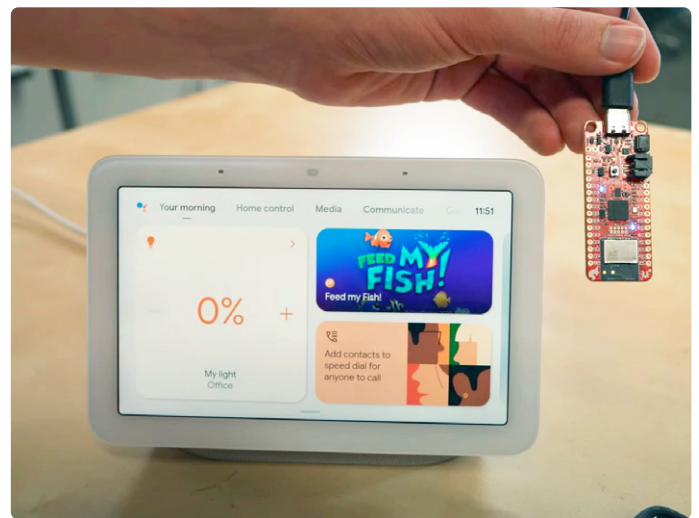


Over de auteur

Rob Reynolds werkt sinds 2015 bij SparkFun, de afgelopen vijf jaar in de rol van Creative Technologist. Met een uitgebreide artistieke achtergrond helpt zijn ervaring hem om projecten, tutorials en video's te maken die meestal even gemakkelijk en amusant als informatief zijn. Je kunt hem op Twitter vinden op [@thingsrobmade](https://twitter.com/thingsrobmade).



Figuur 6. Verander de waarde om de LED in een ander tempo te laten knipperen.



Figuur 7. SparkFun Thing Plus Matter en Google Nest Hub.



Gerelateerde producten

► **SparkFun Thing Plus Matter (MGM240P)**
www.elektor.nl/20442

WEBLINKS

- [1] Matter op GitHub: <https://github.com/project-chip/connectedhomeip>
- [2] SparkFun Thing Plus Matter Development Board: <https://www.sparkfun.com/products/20270>
- [3] MGM240P aansluiten: <https://bit.ly/42NRVll>
- [4] Simplicity Studio: <https://silabs.com/developers/simplicity-studio>
- [5] Thing Plus Matter aansluiten op Google Nest Hub: <https://bit.ly/3VRcQCI>

Het IoT in beeld

Het IoT is allesbehalve oud nieuws. Je zou zelfs kunnen stellen dat IoT-innovatie nog in de kinderschoenen staat. Volgens McKinsey kan de totale waarde van het IoT tegen het jaar 2030 oplopen tot 12,5 biljoen dollar [1]. Sensoren zijn belangrijke onderdelen van het IoT en spelen een rol in talloze toepassingen

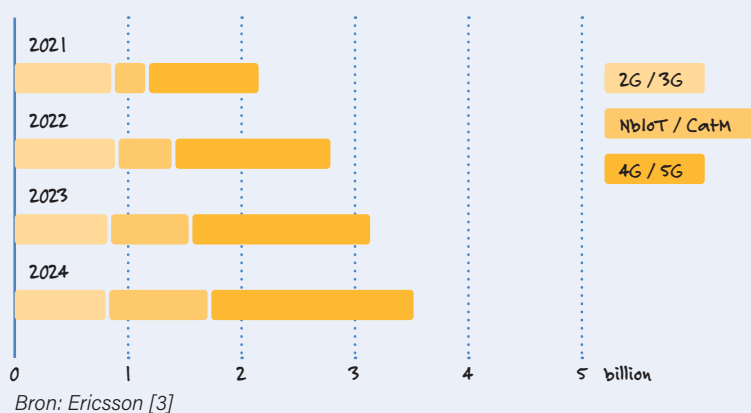
die door leden van de Elektor-community zijn ontworpen en geïmplementeerd. De totale markt van IoT-sensoren zal naar verwachting stijgen van ongeveer 10,9 miljard dollar in 2022 tot 22,1 miljard dollar in 2027[2]. Deze potentiële groei betekent nieuwe kansen voor zowel professionele engineers als makers.



Vijf technologische trends op het gebied van IoT-sensoren

- > slimme sensoren
- > energiezuinige sensoren
- > 'softe' en virtuele sensoren
- > sensor fusion
- > biosensoren

Cellulaire IoT-verbindingen naar segment en technologie



\$12,5 biljoen

McKinsey's schatting van de totale waarde van het Internet of Things tegen 2030.

Het Matter-protocol

Dankzij dit protocol kunnen smart home-apparaten met elkaar communiceren, ongeacht de fabrikant. Het standaardiseert apparaten van verschillende merken. De Connectivity Standards Alliance stelt: "Door voort te bouwen op het Internet Protocol (IP), zal Matter communicatie mogelijk maken tussen smart home-apparaten, mobiele apps en cloud-services, en een specifieke set van IP-gebaseerde netwerktechnologieën definiëren voor apparaat-certificatie" [4]. Apparaten die door Matter worden ondersteund: bridges, controllers, deursloten, HVAC-bediening, verlichting en elektro, media-apparaten, veiligheids- en beveiligingssensoren en jaloezieën en zonwering.

37%

Percentage huishoudens met een internetverbinding die een smart home-apparaat bezitten [6].

68%

Percentage eigenaren (of potentiële eigenaren) van apparaten die Matter-certificering belangrijk vinden [6].

Door Matter ondersteunde apparatuur



HVAC-besturing



jaloezieën
en zonwering



veiligheids- en
beveiligingssensoren



verlichting
en elektro



deursloten



media-
apparaten



controllers en
bridges

Bron: CSA [5]

5G-technologie

5G – de vijfde generatie draadloze cellulaire technologie – biedt tal van voordelen voor bedrijven en consumenten: snellere en veiligere connectiviteit, geringere latentie, langere accu-gebruiksduur en nog veel meer. Sinds de lancering in 2019 is het een van de invloedrijkste technologieën geworden voor de industrie.

50%

Percentage respondenten bij een IEEE-enquête die zeiden dat 5G in 2022 tot de top 5 van belangrijkste technologieën behoorde [7].

\$ 2 biljoen

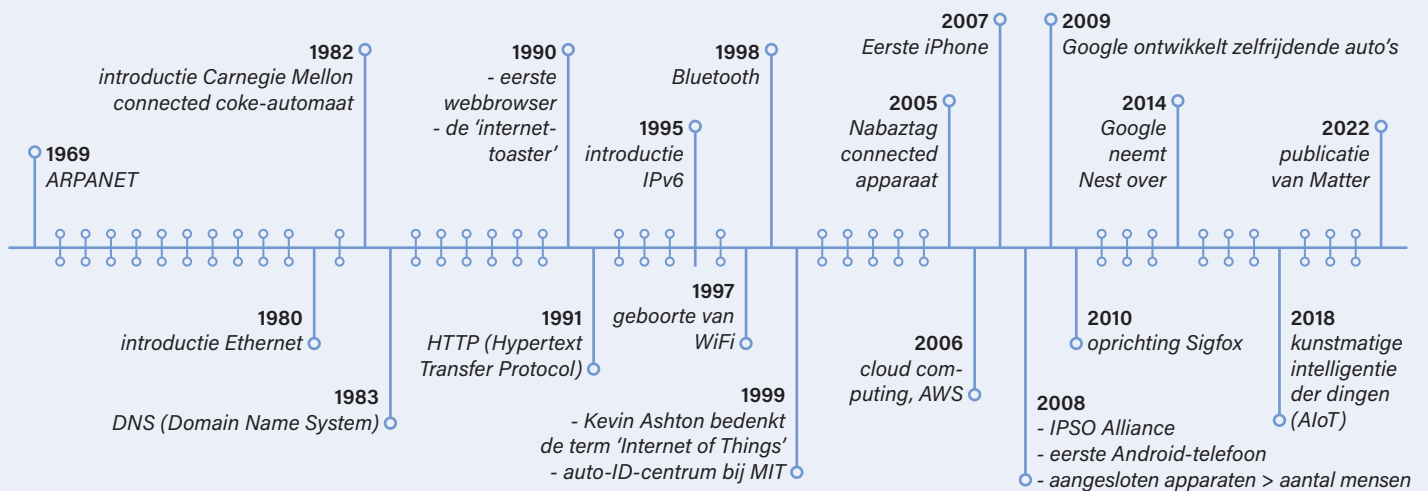
De potentiële boost voor het wereldwijde BBP als 5G wordt ingezet in deze belangrijke commerciële domeinen: retail, productie, gezondheidszorg en mobiliteit [8].

Waar zien IEEE-respondenten [7] 5G een rol spelen?

- leren op afstand
- geneeskunde op afstand
- amusement
- dagelijkse communicatie
- vervoer en verkeersregeling
- productie/assemblage
- energie-efficiëntie

IoT-tijdlijn

Eind jaren negentig bedacht technopionier en auteur Kevin Ashton de term 'Internet of Things'. De onderstaande tijdlijn geeft enkele belangrijke momenten in de geschiedenis van het Internet of Things.



WEBLINKS

- [1] McKinsey & Company, "What is the Internet of Things?" Aug 17, 2022: <https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-the-internet-of-things>
- [2] S. Sinha, "5 IoT sensor technologies to watch," IoT Analytics, Jan 4, 2023: <https://iot-analytics.com/5-iot-sensor-technologies/>
- [3] Ericsson, "IoT Connections Outlook": <https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/iot-connections-outlook>
- [4] CSA, "Matter: The Foundation for Connected Things": <https://csa-iot.org/all-solutions/matter/>
- [5] CSA, "Matter Executive Overview": <https://csa-iot.org/wp-content/uploads/2022/09/22-Matter-Executive-Overview-One-Pager.pdf>
- [6] C. White, "The Wait Is Over and Matter 1.0 Is Here," Parks Associates, Oct 6, 2022: <https://www.parksassociates.com/blog/article/matter-is-here>
- [7] IEEE, "Advancing Connectivity in 2023," IEEE Transmitter, Oct 24, 2022: <https://transmitter.ieee.org/advancing-connectivity-in-2023/>
- [8] McKinsey & Company, "What Is 5G?" Oct 7, 2022: <https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-5g>



Matter, ExpressLink, Rainmaker – waar hebben we het over?

vragen van Tam Hanna (Hongarije) en Jens Nickel (Elektor)

Op embedded world 2023 in Neurenberg liet Espressif, bekend van de beroemde ESP32-microcontroller, een heleboel automatiseringsoplossingen zien die het leven van IoT app-ontwikkelaars (en gebruikers) moeten veraangenamen. Het is echter lastig te begrijpen wat nieuwe begrippen als ExpressLink en RainMaker precies betekenen. In dit vraaggesprek laat Amey Inamdar, Technical Marketing Director bij Espressif, er zijn licht over schijnen. Hij beantwoordt ook algemene vragen over de producten van zijn bedrijf.

Elektor: Vertel ons eens over Espressif vandaag. Waar richten jullie je op? Hoe is het bedrijf veranderd sinds de oprichting in 2008?

Amey Inamdar: Espressif richtte zich op het democratiseren van het IoT-segment met innovatieve, op ontwikkelaars gerichte en betaalbare oplossingen voor WiFi-connectiviteit. We zorgden ervoor dat onze hardware gemakkelijk toegankelijk was en dat onze software beschikbaar was in de open-source gemeenschap. Dit is nog steeds een kernfilosofie van Espressif. Onze focus is echter verbreed door de veranderende eisen van de markt.

Espressif blijft niet alleen de WiFi- en BLE-connectiviteit verbeteren door WiFi 6, dual-band ondersteuning en Bluetooth LE(5) toe te voegen aan zijn portfolio, maar speelt ook in op de groeiende behoeften van de markt door 802.15.4-radio te ondersteunen als basis voor Thread- en ZigBee-protocollen. We hebben op het gebied van connectiviteit het energieverbruik en

de RF-prestaties voortdurend verbeterd. We hebben ook geavanceerde interface-randapparatuur aan ons productpalet toegevoegd.

Daarnaast heeft Espressif baanbrekend werk verricht op het gebied van multi-core MCU's met de ESP32- en ESP32-S3-chips. Ondersteuning voor AI-versnelling in de ESP32-S3 is gunstig voor ML on-the-edge toepassingen. Hardwareversnellers en media-encoders zijn ook toegevoegd aan nieuwere chipsets.

Verder bieden Espressif SoC's beveiligingsfuncties om ervoor te zorgen dat ermee gebouwde apparaten voldoen aan de beveiligingseisen. De meeste chips hebben ook innovatieve beveiligingsperiferie, zoals de digitale handtekening, die geïntegreerde hardware-beveiligde functionaliteit als een soort module biedt. Ook heeft Espressif zich ontwikkeld als leverancier van complete oplossingen, waarbij we pijnpunten van klanten identificeren en deze effectief aanpakken met oplossingen die verder gaan dan hardware en SDK. ESP RainMaker-, ESP Insights- en ESP ZeroCode-modules zijn hier goede voorbeelden van.

Elektor: Op embedded world toonde Espressif domotica-oplossingen met ESP32-boards die verbonden zijn met AWS. We hoorden over zowel 'RainMaker' als 'ExpressLink'. Kun je ons meer vertellen over deze twee? Zijn ze onafhankelijk van elkaar of kunnen ze samenwerken?

Amey Inamdar: ESP RainMaker is een IoT-cloudimplementatie die je kunt inzetten in je eigen AWS-account. Het heeft ook een open-source firmware SDK, telefoontoepassingen en spraakassistent-mogelijkheden. ESP RainMaker is gebaseerd op de serverloze architectuur van AWS en maakt intern gebruik van AWS IoT Core en aanverwante diensten.

ExpressLink is een connectiviteitsmodule die een eenvoudige AT commando-interface biedt naar de host-MCU en naadloze connectiviteit biedt met AWS IoT Core en gerelateerde diensten, zoals OTA.

ExpressLink vermindert de complexiteit aan de apparaatzijde om connected apparaten te bouwen en te beheren.

ESP RainMaker en ExpressLink zijn complementair. Klanten kunnen een van beide – of beide samen – gebruiken om eenvoudig connected apparaten te bouwen.

Elektor: Laten we beginnen met RainMaker [1]. Volgens de documentatie zijn de functies van RainMaker toegankelijk via ESP-IDF. Sinds de herfst is er ook een Arduino-interface beschikbaar. Zou je voor professioneel gebruik nog steeds IDF aanraden?

Amey Inamdar: ESP-IDF is een SDK voor het bouwen van IoT-toepassingen. Het is niet het enige software-framework, maar ESP-IDF is het project waar we als eerste ondersteuning introduceren voor nieuwe producten die we uitbrengen. ESP-IDF is een open source-project, en het is ook een basis voor veel andere Espressif software-frameworks, applicaties en oplossingen.

Arduino biedt een eenvoudige interface en maakt het mogelijk om te profiteren van bestaande bibliotheken en stuurprogramma's voor randapparatuur. ESP-IDF biedt meer flexibiliteit voor klanten die multithreaded applicaties willen ontwikkelen met toegang tot alle native SDK-API's. Klanten kunnen kiezen tussen de Arduino-interface en IDF op basis van hun gebruiksscenario's.

Elektor: Veel gebruikers lijken RainMaker leuk te vinden, maar ze zijn terughoudend om het op de AWS-cloud te hosten. Kan Rainmaker ook zonder AWS worden gebruikt? Is zoiets gepland voor de toekomst?

Amey Inamdar: Er zijn meerdere benaderingen om IoT-cloudplatforms te bouwen. Je zou zoiets kunnen bouwen met Platform as a Service (PaaS) door containers of virtuele machines van cloud-bedrijven te gebruiken. Dergelijke cloud-implementaties vereisen echter technische inspanningen om schaalbaarheid en kosten te handhaven. Er zijn maar weinig klanten die dergelijke DevOps kunnen beheren. Daarom hebben we besloten om ESP RainMaker te baseren op de AWS Serverless architectuur, die een onderhoudsvrije oplossing biedt met pay-as-you-go prijzen. Met deze achtergrond kun je zien dat, door deze bewuste keuzes ten gunste van klanten, het niet eenvoudig is om RainMaker zonder AWS te gebruiken.

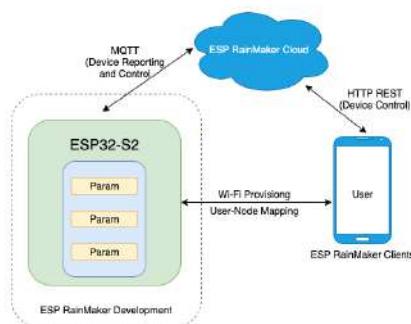
Elektor: De RainMaker-specificaties voorzien in standaard-apparaattypes; de meeste daarvan zijn op het gebied van huisautomatisering (bijvoorbeeld jaloezieën, ventilatoren en inbraakalarmen) [2]. Het systeem zou echter ook voor andere toepassingen kunnen worden gebruikt, zoals studioverlichting en videoapparatuur, om nog maar te zwijgen van typische industriële productie-

Get Started

Note: ESP RainMaker works with all variants of ESP32 like ESP32, ESP32-S2, ESP32-C3 and ESP32-S3. We will just use the name ESP32 to mean all of these, unless explicitly specified otherwise.

Introduction

The ESP RainMaker GitHub project should be used for implementing a "Node" which can then be configured by logged in Users using the clients (like a phone app) and then controlled via the ESP RainMaker Cloud. The examples in this guide are best suited for ESP32-S2-Saola-1, ESP32-C3-DevKitC and ESP32-S3-DevKitC, but the same can be used on other boards by re-configuring Button/LED GPIOs.



apparatuur. Is Espressif van plan om het toepassingsgebied uit te breiden, of zie je RainMaker alleen op het gebied van domotica?

Amey Inamdar: Het ESP RainMaker cloud-backend is volledig agnostisch voor apparaattypes. Het fungeert als een besturings-doorgeefluik en tijd-seriële gegevensopslag. Het kost dus geen moeite om een bepaald apparaattype te ondersteunen. Zelfs de telefoon-apps in de app-stores genereren dynamisch de UI op basis van de beschrijving die door het apparaat wordt gegeven. Als het apparaat bijvoorbeeld zegt dat het een geluidsmixer is die acht verschillende frequentietuners heeft en dat elke frequentie wordt weergegeven als een min/max-schuifregelaar, dan zullen de telefoon-apps automatisch de UI renderen en is er geen verandering nodig aan de cloud-zijde.

Elektor: Zou het geen goed idee zijn om Rainmaker open te stellen voor andere microcontroller-bedrijven zoals Microchip?

Amey Inamdar: Het ESP RainMaker-protocol en apparaat SDK zijn volledig open source, en we vinden het niet erg om ESP RainMaker te gebruiken met MCU's die niet van Espressif zijn.

Elektor: En dan ExpressLink [3]. We kunnen op het internet lezen dat "ExpressLink-compatibele modules een eenvoudige seriële interface bieden waarmee de host-MCU verbinding maakt met AWS IoT-services, waardoor elk offline-product een Cloud-connected product wordt." Maar zoals we zien op een AWS-webpagina [4] van Espressif, lijkt er slechts één compatibel board beschikbaar te zijn.

Amey Inamdar: Dat klopt, maar dat is slechts een ontwikkelboard voor evaluatie en prototyping rond de ExpressLink-compatibele module [wat in feite een ESP32-C3-module is met voorgeprogrammeerde firmware die alle AWS-magie doet – red.]. We hebben gekozen voor een Arduino-compatibele vormfactor voor dit ontwikkelboard. De pin-lay-

Figuur 1. RainMaker verbindt clients (zoals smartphones) en ESP32-gebaseerde apparaten via een cloud-backend dat is gebaseerd op AWS.



De smart home-industrie had standaardisatie nodig.

Figuur 2. RainMaker wordt geleverd met een krachtige SDK en een lange lijst van voorgedefinieerde apparaattypes.

ESP RAINMAKER®						
Smart Home Docs API Help						
Get Started >	Switch	esp.device.switch	Name, Power*	SWITCH	SWITCH	
Develop Firmware	Lightbulb	esp.device.lightbulb	Name, Power*, Brightness, Color Temperature, Hue, Saturation, Intensity, Light Mode	LIGHT	LIGHT	
Basics						
Standard Types						
Specifications >						
Services >	Light	esp.device.light	Name, Power*, Brightness, Color Temperature, Hue, Saturation, Intensity, Light Mode	LIGHT	LIGHT	X
CLI >						
3rd Party Integrations >						
Other Features >						
What's Next? >						
Documentation Feedback >						
	Fan	esp.device.fan	Name, Power*, Speed, Direction	FAN	FAN	
	Temperature Sensor	esp.device.temperature-sensor	Name, Temperature*	X	TEMPERATURE _SENSOR	
	Outlet	esp.device.outlet	Name, Power*	OUTLET	SMARTPLUG	
	Plug	esp.device.plug	Name, Power*	OUTLET	SMARTPLUG	X
	Socket	esp.device.socket	Name, Power*	OUTLET	SMARTPLUG	X
	Lock	esp.device.lock	Name, Lock State*	LOCK	SMARTLOCK	
	Internal Blinds	esp.device.blinds-internal	Name, Blinds Position*	BLINDS	INTERIOR_BLIND	X
	External Blinds	esp.device.blinds-external	Name, Blinds Position*	BLINDS	EXTERIOR_BLIND	X
	Garage Door	esp.device.garage-door	Name, Garage Position*, Lock State	GARAGE	GARAGE_DOOR	X

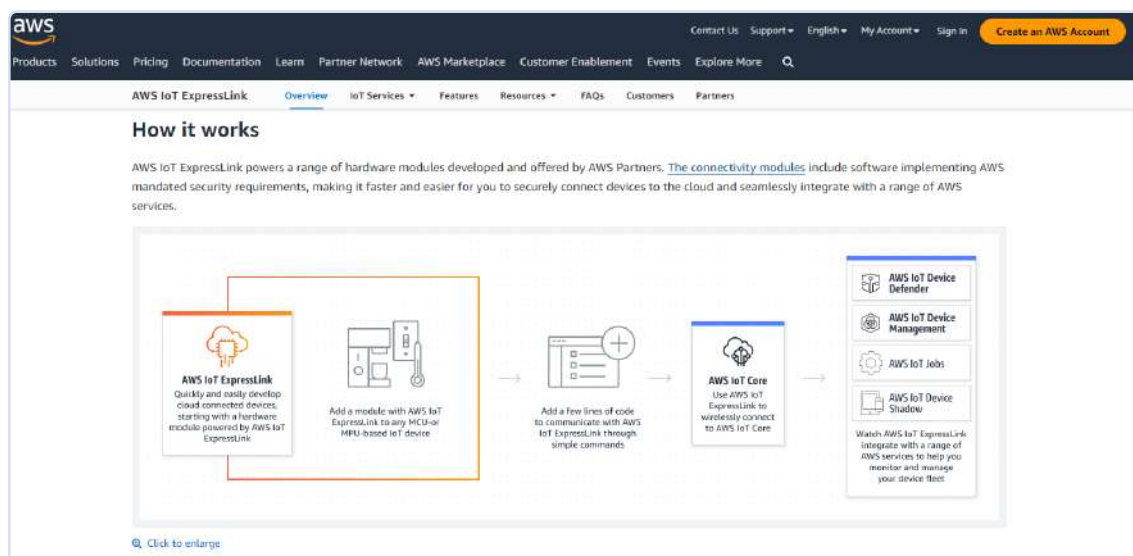
out van deze ESP32-C3-AWS-ExpressLink-DevKit is compatibel met die van het Arduino Zero-board en kan er rechtstreeks op worden aangesloten. Het kan ook eenvoudig worden aangesloten op andere host-MCU's, zoals de Raspberry Pi.

Elektor: Over Matter [5] gesproken, Espressif is een van de eerste en meest prominente aanbieders van oplossingen. Hoe hebben jullie die unieke positie bereikt? Waarom realiseerde Espressif zich de waarde van deze standaard?

Amey Inamdar: De smart home-industrie had standaardisatie nodig – eerdere pogingen hadden geen succes. De 'gefragmenteerde' gebruikersserva-

ring maakte het moeilijk voor consumenten om de aangesloten apparaten te gebruiken en moeilijk voor fabrikanten om ze te bouwen. Dit keer kwamen grotere spelers in het ecosysteem samen onder de paraplu van de Connectivity Standards Alliance en toonden ze zich bereid om het tot een succes te maken. Bovendien hebben de ontwerpoverwegingen van het protocol sterk bijgedragen aan het succes. Opvallende voorbeelden: cryptografische beveiliging voor alle communicatie, ondersteuning van zowel WiFi- als Thread-data-transport, gebruik van blockchain voor authenticatie en veilige OTA, en nog zo het een en ander. Espressif verkeert in een unieke positie om de meest

Figuur 3. ExpressLink is een connectiviteitsmodule die een eenvoudige AT commando-interface naar de host-MCU biedt.



Figuur 4. Momenteel zijn er slechts een paar ExpressLink-ontwikkelboards van Espressif en andere bedrijven beschikbaar.

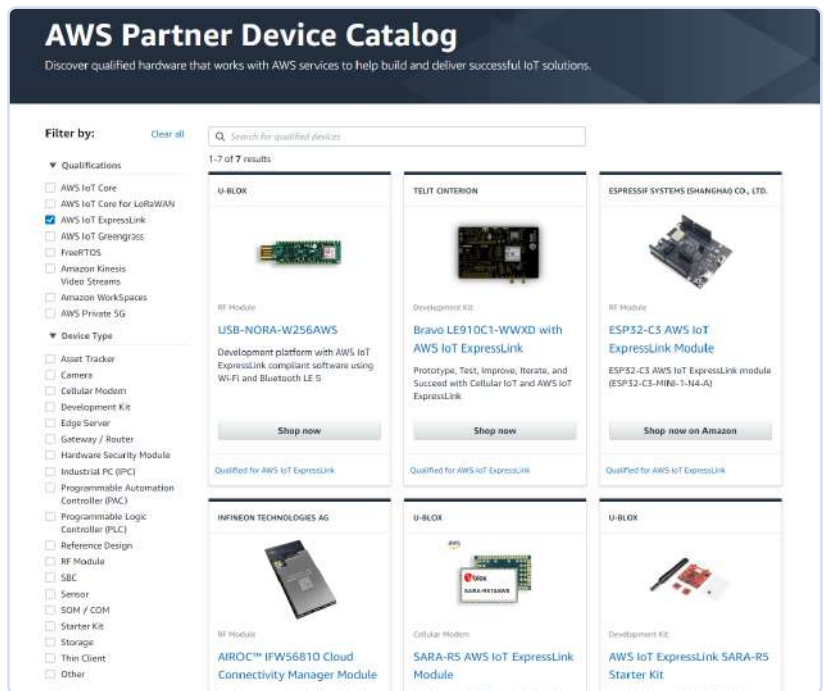
uitgebreide oplossing voor Matter aan te bieden met hardware, software, kant-en-klare oplossingen en services. Espressif biedt chips en modules om WiFi Matter-accessoires, Thread Matter-accessoires, Thread Border Routers en Matter-bridges te bouwen. *esp-matter* is een open-source SDK die tools en voorbeelden biedt om Matter-accessoires te maken. Espressif's DAC-provisioning service biedt modules die veilig zijn voorzien van alle attestaten en certificaten voor apparaten, zodat klanten zich geen zorgen hoeven te maken over veilige en complexe productie. Espressif's ondersteuning helpt klanten om hun Matter-compatibele accessoires gecertificeerd te krijgen. En ESP ZeroCode-modules worden geleverd met voorgecertificeerde firmware voor eenvoudige apparaten, zodat klanten direct Matter-compatibele accessoires kunnen bouwen zonder dat er ontwikkeling van hun kant nodig is.

Elektor: De ESP32-P4 laat zien dat Espressif zich buiten de traditionele markt voor communicatiemodules begeeft. Gezien het feit dat leveranciers zoals ST en Microchip hun klanten geautomatiseerde codegeneratie bieden, wat is dan het unieke verkoopargument voor de P4?

Amey Inamdar: ESP32-P4 heeft heel wat functionaliteit: betere rekenkracht, verbeterde periferie en geheugenarchitectuur. ESP32-P4 kan worden gekoppeld met elke andere Espressif-connectiviteitschip. In die gevallen zal dit een aantal interessante gebruiksscenario's opleveren in het hogere segment van IoT-apparaten. Het belangrijkste verkoopargument van ESP32-P4 is de gestandaardiseerde software-ondersteuning waarbij dezelfde ESP-IDF ESP32-P4 ondersteunt, zodat ontwikkelaars hun leer- en port-toepassingen naadloos kunnen overzetten van andere Espressif SoC's naar de P4. Bovendien blijft de rest van het ecosysteem – met high-level taalinterpreters, RTOS-ondersteuning, SDK's en softwarecomponenten – bruikbaar op de ESP32-P4.

Elektor: Wat zijn jullie plannen, gezien het feit dat steeds meer leveranciers zich op de modulemarkt begeven, om op de langere termijn aan de top van de markt te blijven?

Amey Inamdar: We willen de juiste dingen blijven doen voor onze klanten en we geloven dat dat ons zal helpen om onze leidende positie op de langere termijn te behouden. Ik denk niet dat er één enkele



dimensie is. Een hoog integratieniveau in onze SoC, innovatie en de keuze van de architectuur van het SoC- en communicatiesubstelsysteem, openheid op het vlak van software en informatie, geen concessies doen op het vlak van beveiliging, een efficiënte toeleveringsketen en flexibele productie zijn enkele punten die het verschil uitmaken. ◀

230227-03



Espressif op embedded world 2023
- bekijk de video met Stuart Cording op
het Elektor Industry YouTube-kanaal:
youtu.be/EFnUtAJX2aA



Over Amey Inamdar

Amey werkt als Director of Technical Marketing bij Espressif. Hij heeft 20 jaar ervaring op het gebied van embedded systemen en connected apparaten met functies in engineering, productmanagement en technische marketing. Hij heeft met veel klanten samengewerkt om succesvolle connected apparaten te bouwen op basis van WiFi- en Bluetooth-connectiviteit.

WEBLINKS

- [1] ESP RainMaker-webpagina: <https://rainmaker.espressif.com/>
- [2] ESP RAINMAKER Standard Predefined Types of Devices: <https://rainmaker.espressif.com/docs/standard-types.html>
- [3] AWS IoT ExpressLink-pagina: <https://aws.amazon.com/iot-expresslink/>
- [4] ExpressLink-producten op AWS Partner Device Catalog: <https://devices.amazonaws.com/search?page=1&sv=iotxplnk>
- [5] Espressif's oplossingen voor Matter: <https://espressif.com/en/solutions/device-connectivity/esp-matter-solution>



De juiste keuze van microcontroller-ontwikkelkits

voor IoT- en IIoT-toepassingen

Mark Patrick (Mouser Electronics)

Het Internet of Things (IoT) is overal om ons heen. Embedded ontwikkelaars moeten bij het maken van een nieuw IoT-ontwerp rekening houden met meerdere factoren, zoals energieverbruik, detectiemogelijkheden en draadloze connectiviteit. Dan is er ook nog de marktdruk. IoT-ontwikkelkits bieden een levensvatbaar en handig platform voor het maken van prototypes waarop een ontwerp kan worden gebaseerd. De mogelijkheden van IoT-ontwikkelkits kunnen echter sterk variëren.

Dit vraagt om zorgvuldige aandacht voor de toepassingseisen, de functies van de kit en de capaciteit. Dit artikel belicht enkele van de vele overwegingen die een rol spelen bij de keuze van een IoT-ontwikkelkit voor een nieuw ontwerp.

Het online-tijdperk

Het lijkt geen twijfel dat we in het online-tijdperk leven. Verbonden apparaten zijn overal om ons heen. Sommige dragen we of helpen ons om nauwkeurig ons elektriciteitsverbruik te monitoren, en andere waarschuwen ons als er een bezoeker voor de deur staat. Voor industriële productieprocessen verandert de komst van het Industrial Internet of Things (IIoT) de manier waarop fabrieken werken en helpt het de algehele effectiviteit van de apparatuur te verhogen. In slechts één decennium hebben we de manier waarop we met de wereld omgaan en deze controleren veranderd. We verbazen ons erover hoe we het ooit zonder mobiele telefoons konden stellen nu direct toegang tot informatie over aspecten van ons leven en werk het normaalste van de wereld is.

Ook onze auto's ondergaan radicale veranderingen, met actuele verkeersinformatie die ons waarschuwt voor mogelijke vertragingen. Dankzij met het internet verbonden bewakingsapparatuur in de gezondheidszorg kunnen patiënten thuis genezen onder toezicht en beschikbaarheid van medisch personeel, mocht ingrijpen nodig zijn.

Het IoT werd snel omarmd door de industrie, omdat overheidsinitiatieven, zoals Industry 4.0, de behoefte aan automatisering, verbetering van de procesefficiëntie



en meer gestroomlijnde operaties aanwakkerden. Een leger van sensoren bewaakt en rapporteert nu over de status van elke fase van een proces en stuurt gegevens terug naar het automatische controle- en analysesysteem.

De voordelen van IoT/IIoT-implementaties zijn aanzienlijk, maar vanuit het perspectief van elektrotechniek gaat de ontwikkeling van een IoT-apparaat gepaard met veel uitdagingen.

De vereisten van een IoT-apparaat

IoT-toepassingen variëren aanzienlijk, maar een kernset van functionele eisen blijft hetzelfde, of u nu een druksensor ontwerpt voor een industrieel proces of een aanwezigheidssensor voor een kantoor.

Bij de eerste inventarisatie om de technische specificatie vast te stellen voor een nieuw IoT-apparaat moet rekening worden gehouden met elk van de hieronder genoemde aspecten, aangezien die bepalend zijn voor de functionele architectuur en het ontwerp.

Detectie: sensoren nemen de wereld om ons heen waar, van temperatuur en luchtdruk tot de bewegingen van mensen. Een camera kan bijvoorbeeld gegevens doorsturen naar een machine learning-toepassing voor objectdetectie en bevestigen dat een etiket correct op een fles is aangebracht. Verschillende technische beslissingen hangen af van wat wordt gedetecteerd en hoe vaak. Andere overwegingen zijn kosten, omvang en complexiteit van de sensor. Een thermistor voor het meten van de temperatuur vereist extra componenten voor het analoge domein en een softwareverwerking voor omzetting in digitale vorm. Een andere factor is hoeveel sensoren er nodig zijn en hoe vaak ze moeten worden uitgelezen.

Connectiviteit: hoe communiceert het IoT-apparaat met een host-besturingssysteem? Is er betrouwbare draadloze communicatie beschikbaar in elk use case-scenario, of heeft communicatie via een vaste lijn de voorkeur? Het type sensor bepaalt tevens de hoeveelheid gegevens die moet worden overgedragen en hoe vaak. Draadloze mesh-technologie kan bij een grote inzet een robuustere communicatielink bieden, maar vereist dat alle IoT-appara-

ten op deze manier werken. Voor draadloze communicatie kan worden gekozen voor een discreet ontwerp of een module met typegoedkeuring.

Voedingsbron: wat is het energieverbruikprofiel van uw IoT-apparaat? Sommige toepassingen, communicatiefrequenties en draadloze protocollen vertegenwoordigen een aanzienlijke stroombelasting die de capaciteit van een kleine batterij te boven gaat. Is er voor sommige installatiescenario's een netvoeding beschikbaar? Een recente trend op het gebied van IoT-sensoren maakt gebruik van energy harvesting-technologieën om de batterij geheel te elimineren. In plaats daarvan wordt energie geoogst uit omringende energiebronnen, zoals zonne-energie, trillingen en warmte om een supercondensator op te laden.

Gebruikersinterface: vereist het IoT-apparaat een gebruikersinteractie? Hoe zit het tijdens de installatie en de verbinding met het hostsysteem als het niet in bedrijf is? Zijn er displays, een andere vorm van indicatie of status-LED's nodig?

Toepassingen voor analyse en controle in de cloud: IoT-apparaten maken verbinding met een controlerend hostsysteem. De connectiviteitsmethode en -protocollen bepalen de softwarevereisten van de sensor en hoe deze met de host communiceert. Is er een constante datalink vereist om gegevens te streamen, of kunnen deze met regelmatige tussenpozen als batch worden verzonden?

IoT Dev Kit selectie: hints en tips

Ontwikkelkits bieden embedded ontwikkelaars een handige en snelle manier om een prototype van een ontwerp te maken. In dit deel van het artikel belichten we enkele van de factoren waarmee ontwikkelaars rekening moeten houden bij het selecteren van een geschikte kit. Er is een ruime keuze aan IoT-ontwikkel- en evaluatiekits van toonaangevende leveranciers van microcontrollers. Het is dus het beste om een weloverwogen beslissing te nemen op basis van de hierboven genoemde toepassingsvereisten. Hieronder vindt u een lijst met enkele punten om rekening mee te houden bij de keuze van een dev kit-platform.

Voeding:

- Hoe wordt het board gevoed? Via USB

van een host-werkstation? Met een batterij? Kan het worden gevoed door de daarvoor bedoelde voedingsbron en heeft het een PMIC waar u toegang tot hebt om andere voedingsbronnen te proberen?

- Is het mogelijk om inline een stroomsonde te plaatsen om real-time stroomverbruik te meten voor profileringsdoeleinden? Zo ja, dekt deze alles op het board, eventuele extra modules, sensoren enzovoort?

Sensoren:

- Is het board uitgerust met voor uw toepassing geschikte sensoren?
- Is het mogelijk om extra sensoren toe te voegen? Bijvoorbeeld via een perifere aansluiting of een industrie-standaard add-on formaat, zoals mikroBUS Click?
- Welke perifere interfaces zijn er beschikbaar? I²C, UART, SPI, GPIO?
- Heeft het board of de microcontroller een ADC die u kunt gebruiken en zijn er extra componenten voor signaalconditionering vereist?

Connectiviteit:

- Welke bekabelde/draadloze connectiviteitsopties heeft het board (ethernet, WiFi, LoRa, BLE, ISM enzovoort)?
- Als er geen ingebouwde connectiviteitsopties zijn, kunnen deze dan eenvoudig worden toegevoegd? Adviseert en ondersteunt de fabrikant een geschikte draadloze module of is er een interface van derden aanwezig (mikroBUS Click enzovoort)?
- Is de firmware van het board geschikt voor 'over-the-air' updates?

Rekenkracht:

- Heeft het board een voor u geschikte microcontroller? Heeft u deze al eerder gebruikt en beschikt u al over de juiste programmeertools?
- Is de rekenkracht van het board geschikt om de IoT-toepassing, hostprotocollen en eventuele protocolstacks (connectiviteit) te draaien?
- Als de microcontroller een geïntegreerde draadloze transceiver heeft, kunt u de slaapstanden ervan onafhankelijk regelen om energie te besparen?

Figuur 1. De Microchip EV36W50A IoT-ontwikkelkit (bron: Microchip).

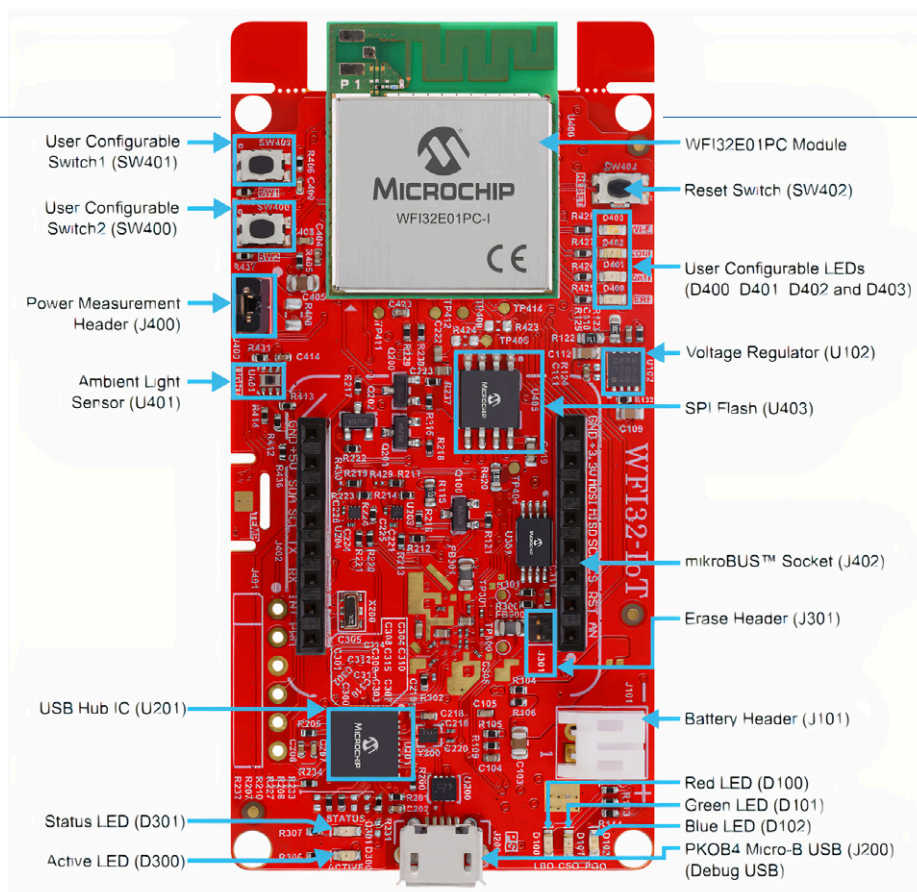
- Welke ingebouwde beveiligingsfuncties heeft de MCU en zijn deze geschikt voor gebruik met uw toepassing?

Besturingselementen:

- Is het board uitgerust met gebruikers-drukknoppen, aanraakgevoelige schuifregelaars of andere besturingshardware?
- Is er een display beschikbaar? Is dit nodig in de uiteindelijke toepassing?
- Heeft u met uw code toegang tot gebruikers-LED's? Zijn er voldoende beschikbaar, of kunt u ze snel toevoegen via een vrije GPIO-poort?

Software-ondersteuning:

- Wat zijn de aanbevolen programmeertools voor dit board? Heeft u die al?
- Is een uitgebreid BSP (board support package) inbegrepen?
- Welke extra stuurprogramma's, bibliotheken en firmware zijn vereist en zijn deze vrij van royalties?
- Check de licentievoorwaarden voor firmware en middleware bij de fabrikant van het board.



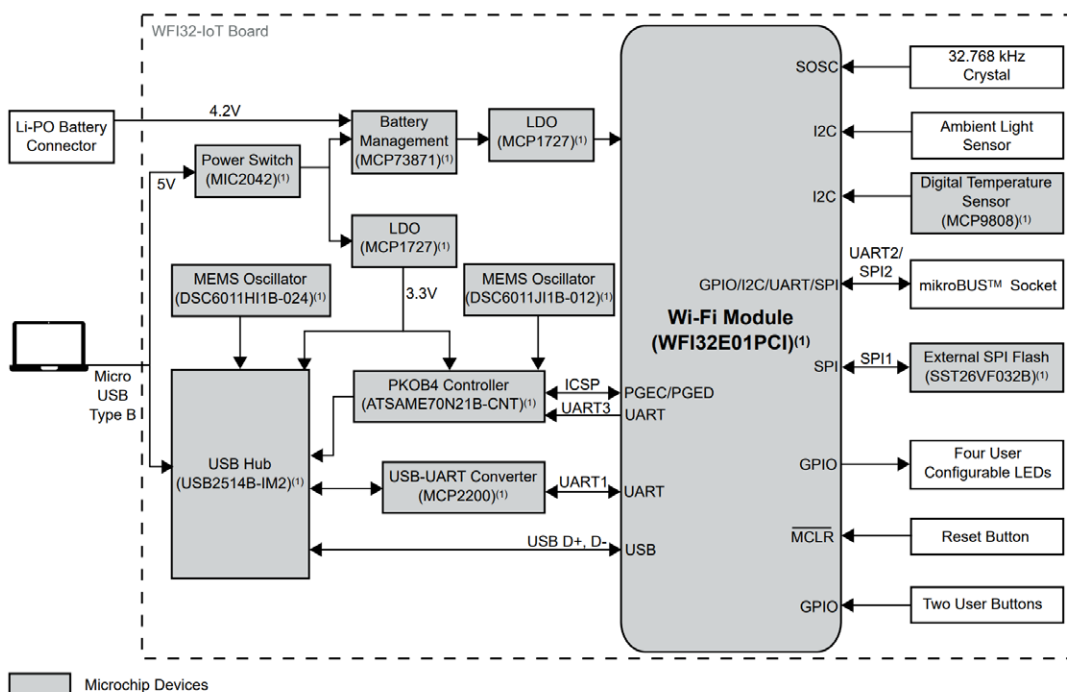
- Wordt het board geleverd met een vooraf geladen demo die de mogelijkheden van het board laat zien? Bevat deze communicatie met populaire service-providers, zoals Microsoft Azure of Amazon AWS?
- Zijn er andere demo's en codevoorbeelden beschikbaar voor het board?

Bestaat er een ecosysteem van bibliotheken en ontwikkelpartners?

IoT-ontwikkelboards in de schijnwerpers

Microchip WFI32 IoT-ontwikkelboard
De Microchip WFI32, onderdeelnummer EV36W50A [1], is een uitgebreid, volledig

Figuur 2. Het functionele blokschema van het Micro WFI32- IoT EV36W50A ontwikkelboard (bron: Microchip).





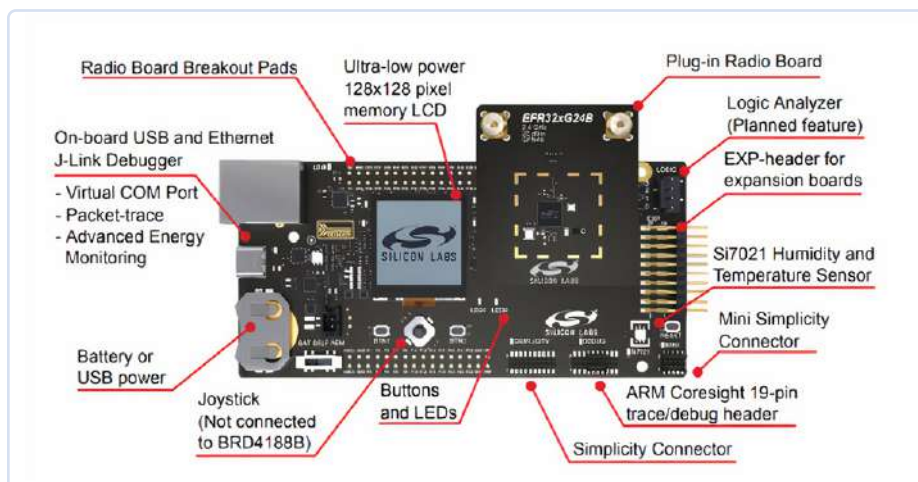
Figuur 3. De STEVAL-ASTRA1B asset tracking ontwikkelkit (bron: ST).

geïntegreerd, standalone IoT-ontwikkelboard (figuur 1). De WFI32-IoT integreert een Microchip WFI32E01PC WiFi 802.11 draadloze module gebaseerd op de PIC-microcontrollerfamilie. De ingebouwde sensoren omvatten een Microchip digitaal I²C temperatuursensor-IC en een digitaal omgevingslichtsensor-IC. Ontwikkelaars kunnen via een mikroBUS-aansluiting extra sensoren of randapparatuur aansluiten. De draadloze MCU-module is tevens uitgerust met een geïntegreerde antenne. Het board kan worden gevoed via een host-werkstation of een LiPo-accu. Een ingebouwde PMIC zorgt voor het opladen van de batterij via de USB-host.

Figuur 2 toont het functionele blokschema van het WFI32-IoT-board en accentueert de in het bord geïntegreerde Microchip-componenten. Het bord is vooraf geladen met een out-of-the-box (OOB) demo-image dat de ingebouwde sensoren uitleest en de gegevens naar de Amazon AWS-cloud stuurt. De democode en de complete instructies zijn beschikbaar via een GitHub-repository [2].

STMicroelectronics STEVAL ASTRA1B multi-connectivity asset tracking reference design

Figuur 3 toont de STEVAL ASTRA1B [3] ontwikkelkit en het referentieontwerp. De kit is speciaal ontworpen voor het bouwen van prototypes en het evalueren van toepassingen met betrekking tot asset



Figuur 4. De Silicon Labs xG24-RB4188A antenne diversity-module gemonteerd op een Silicon Labs Wireless Kit Pro mainboard (bron: Silicon Labs).

tracking en bevat twee draadloze connectiviteitsmodules. Een STM32WB5MMG [4] low-power draadloze 2,4 GHz BLE/ZigBee microcontroller-module en een sub-GHz STM32WL55JC draadloze MCU-module met een groot bereik voor LPWAN-communicatie, zoals LoRa.

De STEVAL ASTRA1B bevat een uitgebreide set sensoren voor het meten van meerdere omgevings- en bewegingsparameters. Een GNSS-module levert gegevens voor positionering van een buitenlocatie. Andere kenmerken van het board zijn een STSAFE-beveiligingselement, een 480 mAh accu en een OOB-demo bestaande uit een asset tracking dashboard en een smartphone-app.

Silicon Labs xG24-RB4188A

De Silicon Labs xG24-RB4188A [5] is een plug-in antenne-diversitymodule voor prototypes van 2,4 GHz draadloze toepassingen (figuur 4). Deze wordt aangesloten op het Silicon Labs BRD4001 draadloze starterboard. De module bevat een Silicon Labs EFR32 Wireless Gecko system-on-chip, een RF-switch, een aanpassingsnetwerk en twee SMA-antenneconnectoren. De RF-uitgang van de EFR32 is +20 dBm.

SEMTECH LR1120 ontwikkelkits

SEMTECH biedt op basis van de SEMTECH LR1120 [6] draadloze microcontroller een reeks aan LR1120-ontwikkelkits



Figuur 5. Een voorbeeld van de SEMTECH LR1120-ontwikkelkits (bron: SEMTECH).



voor het maken van prototypes van LoRa LPWAN-toepassingen [7], zoals getoond in **figuur 5**. De kits zijn beschikbaar in regionale varianten in overeenstemming met het industriële, wetenschappelijke en medische (ISM) sub-GHz spectrum. De LR1120 is geschikt voor multiregionale toepassingen voor het lokaliseren van activa, inventarisbeheer en diefstalpreventie.

Eerder in dit artikel is gewezen op de mogelijkheid om extra sensoren of randapparatuur aan een ontwikkelboard toe te voegen. Zoals al vermeld bij de bespreking van het Microchip-board, is het board uitgerust met een mikroBUS-aansluiting. De mikroBUS, ontwikkeld door Mikroe, heeft zich snel ontwikkeld tot een industriestandaard die door veel leveranciers van halfgeleiders wordt gebruikt voor hun ontwikkel- en evaluatieboards. mikroBUS brengt seriële connectiviteit van SPI, UART en I²C samen met voeding, analoge en PWM-signalen naar een compact connectorformaat.

Mikroe heeft honderden Click-boards [8] die deze handige vormfactor gebruiken.

Een voorbeeld is de Mikroe Ultra-Low Press Click [9]. Deze is ontworpen voor lagedruk pneumatische metingen en heeft een TE Connectivity SM8436 druksensor die communiceert via de I²C-interface (**figuur 6**).

Aan de slag met uw IoT-ontwikkelkit

Het maken van prototypes van een IoT-toepassing is aanzienlijk eenvoudiger geworden dankzij de beschikbaarheid van ontwikkelboards. Dit korte artikel belichtte enkele overwegingen voor embedded ontwikkelaars die op zoek zijn naar een geschikt ontwikkelboard. Naast de genoemde onderwerpen zijn er ook specifieke topics voor de vereiste toepassing die aandacht behoeven.

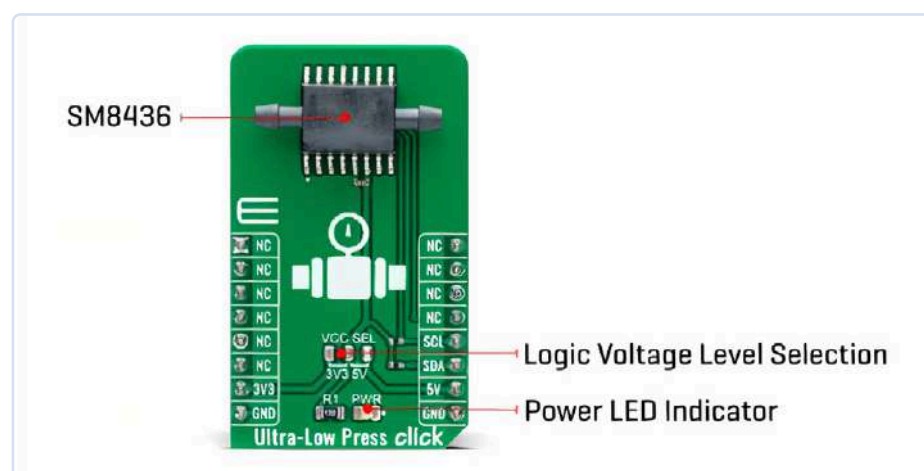
Wat gaat u ontwikkelen? ◀

230338-03



Over de auteur

Als Technical Marketing Manager van Mouser Electronics voor EMEA is Mark Patrick verantwoordelijk voor het creëren en verspreiden van technische content binnen de regio – content die essentieel is voor de strategie van Mouser om het engineeringpubliek te ondersteunen, te informeren en te inspireren. Voordat hij het Technical Marketing-team leidde, maakte Patrick deel uit van het EMEA Supplier Marketing-team en speelde hij een cruciale rol bij het aangaan en ontwikkelen van relaties met belangrijke productiepartners. Naast diverse technische en marketingfuncties heeft Patrick onder andere acht jaar gewerkt bij Texas Instruments in Applications Support en Technical Sales. Hij is een 'hands-on' ingenieur in hart en nieren en heeft een passie voor vintage synthesizers en motorfietsen. Patrick heeft een eerste klas Honours Degree in Electronics Engineering van de universiteit van Coventry.



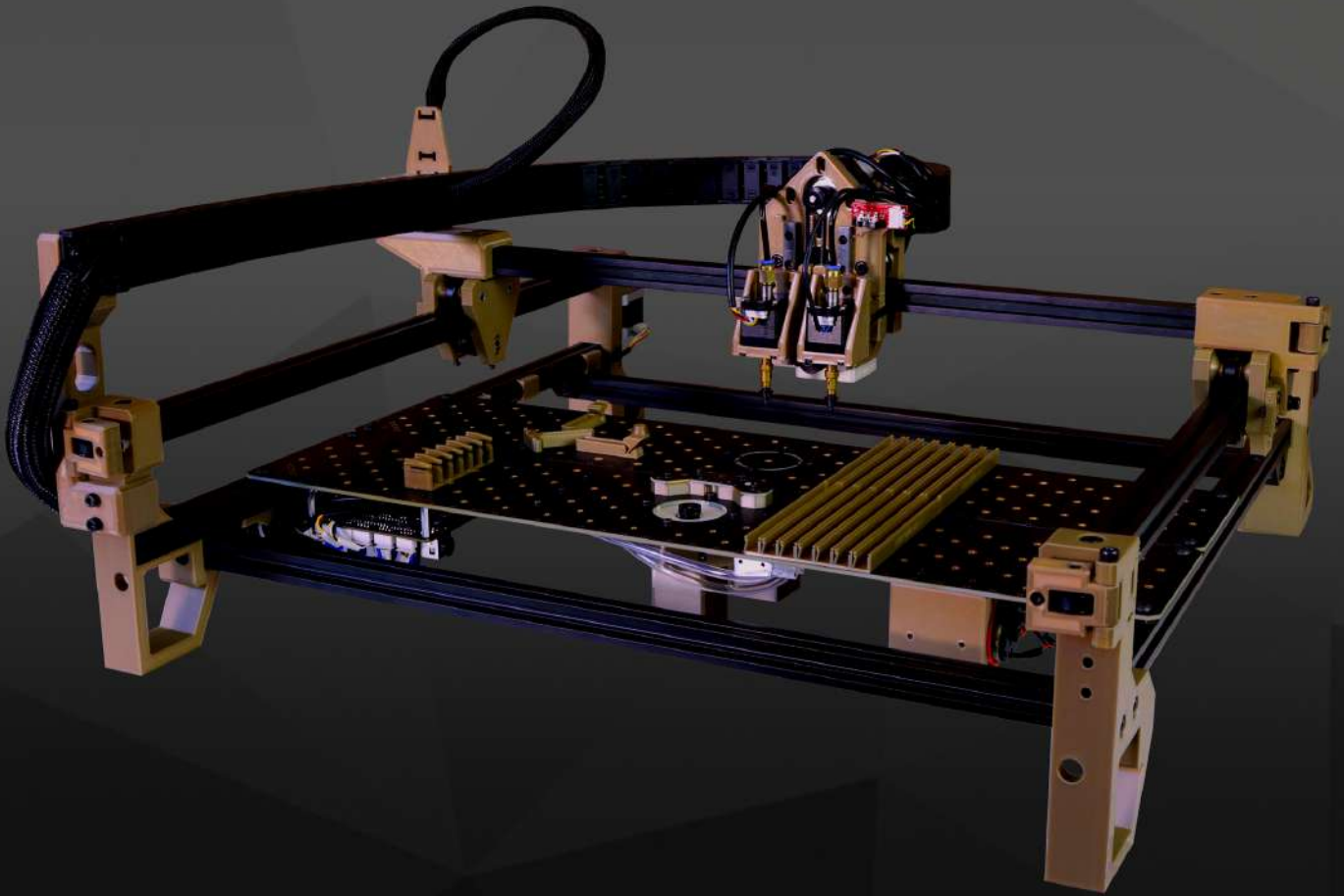
Figuur 6. De Mikroe Ultra-Low Press Click (Bron: Mikroe).

WEBLINKS

- [1] Microchip Technology EV36W50A WFI32-IoT Board : <https://bit.ly/3Vw4L5U>
- [2] WFI32-IoT op GitHub: <https://github.com/MicrochipTech/WFI32-IoT>
- [3] STEVAL-ASTRA1B Asset Tracking Evaluation Board: <https://bit.ly/3LwN4P6>
- [4] STM32WB5MMG 2.4GHz Wireless Module: <https://bit.ly/3NFird0>
- [5] Silicon Labs xG24-RB4188A: <https://bit.ly/3LBRFQ5>
- [6] Semtech LR1120 Wi-Fi/GNSS Scanner + LoRa Transceiver : <https://bit.ly/428oET8>
- [7] LR1120 Development Kits: <https://bit.ly/42rdGaO>
- [8] Click Boards - Mikroe: <https://bit.ly/3LAaH9j>
- [9] Ultra-Low Press Click - Mikroe: <https://bit.ly/3LXr2GH>



versus



Dat is geen strijd.

De LumenPnP Desktop Pick & Place Machine zet uw platen in elkaar, zodat u nooit meer een pincet hoeft te gebruiken.

- Radicaal Open Source
- Aangedreven Feeders
- Dubbele mondstuk
- Plaatst 0402s
- Voordelig



Opulo.io



Condensatoren gedragen zich niet altijd capacitief!

Dr. René Kalbitz (Würth Elektronik eiSos)

Condensatoren gedragen zich capacitief, althans in de theorie van ideale componenten. Dit geldt echter alleen onder bepaalde bedrijfsomstandigheden, en het hangt ook af van het frequentiebereik. Dit artikel belicht de impedantiespectra van verschillende condensatortechnologieën en laat zien wanneer capacitief gedrag te verwachten is en wanneer niet.

Impedantie- en capaciteitsspectra (of S-parameters) zijn gangbare weergaven van de frequentie-afhankelijke elektrische eigenschappen van condensatoren. De interpretatie van dergelijke spectra levert een schat aan elektrochemisch, fysisch en technisch relevante informatie op. Deze moet worden gescheiden van meetartefacten die altijd aanwezig zijn en van parasitaire effecten. Aangezien het niet altijd mogelijk is alle gegevens in de datasheet in kaart te brengen, moeten ingenieurs soms

afgaan op gemeten spectra om de juiste component voor hun eigen schakelingontwerp te selecteren. Om de best mogelijke gegevensbasis te creëren, heeft Würth Elektronik de online tool REDEXPERT [1] geïmplementeerd, waarin niet alleen spectra maar ook andere metingen beschikbaar worden gesteld. Dit artikel beschrijft de kenmerken van dergelijke spectra en bespreekt hoe daaruit elektrische basiseigenschappen kunnen worden afgeleid.

Vervangingsschema voor condensatoren

Met de in **figuur 1** getoonde schakeling is het mogelijk frequentie-afhankelijke impedantiespectra te modelleren voor alle soorten condensatoren, van meerlaagse keramische chipcondensatoren (MLCC) tot supercondensatoren (SC).

C_S staat voor de zuivere capaciteit, die in geen enkele elektrische component op zichzelf bestaat. Elke echte condensator heeft verliezen die het laadproces 'vertragen'. Dit verschijnsel wordt beschreven door de zuiver ohmse equivalente serieweerstand (ESR). De weerstand van de belasting en de draden dragen ook bij tot de ESR. De zuivere verliesvrije capaciteit C_S wordt gedefinieerd door de afgeleide

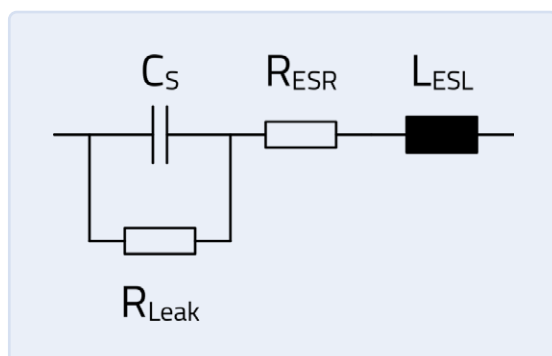
$$C_S = \frac{dQ}{dV}$$

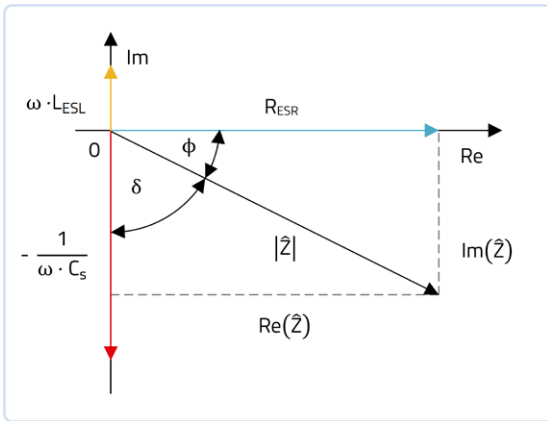
waarbij dQ de ladingsverandering op het condensatoroppervlak is en dV de spanningsverandering over de condensator.

Elke wisselstroom in een metalen geleider induceert een magnetisch veld dat de stroom tegenwerkt. In het beschouwde model (ook LCR-model of standaardmodel genoemd) wordt deze eigenschap beschreven door de equivalente serie-inductantie (ESL), in figuur 1 weergegeven door L_{ESL} .

C_S , R_{ESR} en L_{ESL} zijn de belangrijkste parameters die

Figuur 1. Standaard vervangingsschema voor condensatoren, met capaciteit C_S , equivalente serieweerstand R_{ESR} , equivalente serie-inductantie L_{ESL} en lekweerstand R_{Leak} .





Figuur 2. Vectordiagram van de impedantie in het complexe vlak. R_{Leak} wordt omwille van de eenvoud verwaarloosd.

nodig zijn om de meeste spectra te beschrijven. In de eenvoudigste benadering zijn zij constanten en veranderen zij niet met de frequentie, hetgeen voldoende nauwkeurig is voor de elektrotechniek.

Het ladingsverlies op lange termijn, dat wil zeggen de lekstroom, wordt bij goede benadering beschreven door de zuivere ohmse weerstand R_{Leak} . Gewoonlijk is R_{Leak} orden van grootte groter dan R_{ESR} en kan vaak worden verwaarloosd, dat wil zeggen $R_{Leak} \rightarrow \infty$. Het effect ervan is alleen zichtbaar bij zeer lage frequenties, ver beneden 1 Hz [2].

Impedantie- en capaciteitsspectra

Hieronder worden veelgebruikte termen en grootheden gedefinieerd, zoals capaciteit en impedantie. De bovenstaande schakeling kan worden beschreven als een frequentie-afhankelijke complexe impedantie $\hat{Z}$, capaciteit $\hat{C}$, verstrooiingsparameter (S-parameter) $\hat{S}$, diëlektrische constante ϵ of elke andere meetbare complexe elektrische grootheid. Impedantie $\hat{Z} = \text{Re}(\hat{Z}) + i \times \text{Im}(\hat{Z})$ is een complexe grootheid, met $\text{Re}(\hat{Z})$ en $\text{Im}(\hat{Z})$ als respectievelijk het reële en imaginaire deel. Impedantie wordt vaak uitgedrukt door de absolute waarde $|\hat{Z}|$ en de fasehoek ϕ :

$$\hat{Z} = |\hat{Z}| \cdot e^{i\phi}$$

In een complex vlak zoals in **figuur 2** beschrijft ϕ de hoek tussen $\text{Re}(\hat{Z})$ (abscis) en de complexe vector $\hat{Z}$. Natuurkundig gezien vertegenwoordigt $|\hat{Z}|$ de verhouding tussen de spanningsamplitude en de stroomamplitude, terwijl ϕ het faseverschil tussen spanning en stroom bij een bepaalde frequentie aangeeft. De fasehoek ϕ is als volgt gerelateerd aan de verlieshoek:

$$\arctan\left(\frac{\text{Re}(\hat{Z})}{|\text{Im}(\hat{Z})|}\right) = \delta = \frac{\pi}{2} - \phi$$

In de elektrotechniek is het ook gebruikelijk om de absolute waarde $|\hat{Z}|$ en de equivalente serie weerstand $R_{ESR} = \text{Re}(\hat{Z})$ te gebruiken. In het voorbeeld van figuur 1 is de equivalente serie weerstand het reële deel van de impedantie. Om het verband tussen het model en de complexe magnitude $|\hat{Z}|$ grafisch weer te geven, zijn alle

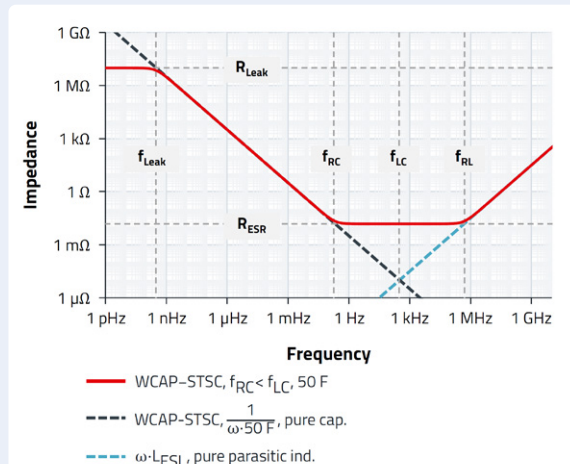
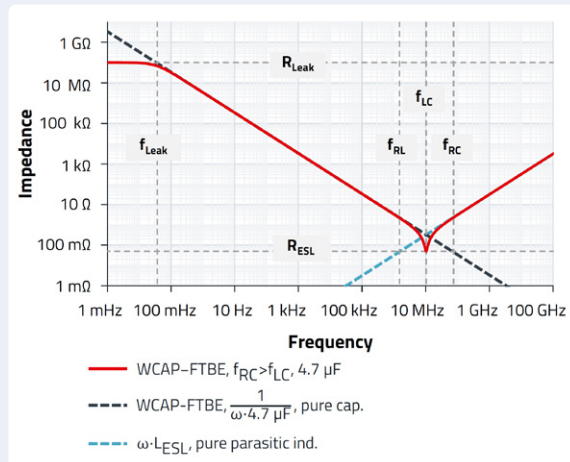
modelparameters (behalve R_{Leak}) ook in figuur 2 aangegeven (de wiskundige beschrijving staat in de appendix van [2]).

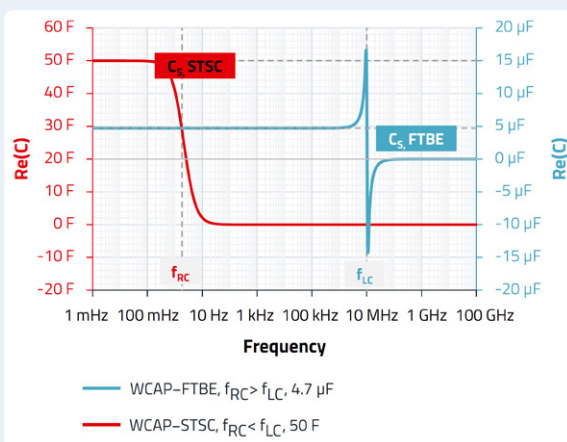
De impedantie kan ook worden herschreven als complexe capaciteit

$$\hat{C} = \frac{1}{i \cdot 2 \cdot \pi \cdot f \cdot \hat{Z}} = \text{Re}(\hat{C}) + i \cdot \text{Im}(\hat{C})$$

Al deze waarden, zoals $\text{Re}(\hat{Z})$, $\text{Im}(\hat{Z})$, $|\hat{Z}|$ of verlieshoek δ , kunnen worden gemeten met impedantie- of netwerk-analysatoren. Elke elektronische component (niet alleen een condensator) kan worden gekarakteriseerd door een reeks frequentie-afhankelijke waarden, zoals $\text{Re}(\hat{Z})$ en $\text{Im}(\hat{Z})$ of $\text{Re}(\hat{C})$ en $\text{Im}(\hat{C})$. Meetresultaten kunnen echter alleen worden geïnterpreteerd aan de hand van vervangingsschema's zoals die van figuur 1. Door aanpassing van C_s , R_{ESR} , L_{ESL} en R_{Leak} kan voor alle condensatoren de basis-frequentierespons worden berekend.

Figuur 3. Impedantiespectra $|\hat{Z}|$ voor filmcondensator WCAP-FTBE (boven) en supercondensator WCAP-STSC (onder) zoals berekend volgens het standaardmodel.





▲
Figuur 4.
Capaciteitsspectra
 $Re(\hat{C})$, zoals berekend
met het standaardmodel.
De grafiek voor
supercondensator
WCAP-STSC (rood) komt
overeen met de linker
verticale as en de plot
voor filmcondensator
WCAP-FTBE (blauw)
komt overeen met de
rechter verticale as.

Dit wordt als voorbeeld getoond voor de impedantie- en capaciteitsspectra van een condensator van 4,7 μF en een condensator van 50 F in respectievelijk **figuur 3** en **figuur 4**. De bijbehorende fase- en verlieshoeken zijn weergegeven in de appendix van [2]. De parameters voor de twee voorbeelden luiden als volgt:

- supercondensator (WCAP-STSC) met $C_S = 50 \text{ F}$,
 $R_{\text{ESR}} = 15 \text{ m}\Omega$, $L_{\text{ESL}} = 5 \text{ nH}$ en $R_{\text{Leak}} = 10 \text{ M}\Omega$
- filmcondensator (WCAP-FTBE) met $C_S = 4,7 \mu\text{F}$,
 $R_{\text{ESR}} = 5 \text{ m}\Omega$, $L_{\text{ESL}} = 5 \text{ nH}$ en $R_{\text{Leak}} = 10 \text{ M}\Omega$

De parameters werden zo gekozen dat ze overeen komen met bestaande Würth Elektronik producten, die te vinden zijn onder de productnamen voor filmcondensatoren WCAP-FTBE (4,7 μF) en supercondensatoren WCAP-STSC (50 F). In deze grafieken werden C_S , R_{ESR} , L_{ESL} en R_{Leak} constant en onafhankelijk van de frequentie verondersteld (**tabel 1**).

Over het algemeen worden de meest prominente delen in de spectra beschreven door vier karakteristieke frequenties:

Karakteristieke frequentie f_{RC} van het R_{ESR} -C element:

$$f_{\text{RC}} = \frac{1}{2 \cdot \pi \cdot R_{\text{ESR}} \cdot C_S}$$

Elektrische parameters	WCAP-FTBE	WCAP-STSC
C_S	4,7 μF	50 F
R_{ESR}	5 m Ω	15 m Ω
L_{ESL}	5 nH	5 nH
R_{Leak}	10 M Ω	10 M Ω

Tabel 1. Elektrische parameters gebruikt voor de berekening van de spectra.

Karakteristieke frequentie f_{LC} van het L-C element:

$$f_{\text{LC}} = \frac{1}{2 \cdot \pi \cdot \sqrt{L_{\text{ESL}} \cdot C_S}}$$

Karakteristieke frequentie f_{Leak} van het R_{Leak} -C element:

$$f_{\text{Leak}} = \frac{1}{2 \cdot \pi \cdot R_{\text{Leak}} \cdot C_S}$$

Karakteristieke frequentie f_{RL} van het R_{ESR} -L element:

$$f_{\text{RL}} = \frac{R_{\text{ESR}}}{2 \cdot \pi \cdot L_{\text{ESL}}}$$

Figuur 3 en figuur 4 tonen twee belangrijke situaties:

- Lorentz-oscillatie: $f_{\text{RC}} > f_{\text{LC}}$ zoals in het geval van $C_S = 4,7 \mu\text{F}$ (blauwe curve)
- Debye-relaxatie: $f_{\text{RC}} < f_{\text{LC}}$ zoals in het geval van $C_S = 50 \text{ F}$ (rode curve)

De zwarte en blauwe stippellijnen in beide grafieken van figuur 3 geven het zuiver capacitieve en inductieve deel aan. f_{RC} , de karakteristieke frequentie van het RC-element, is de frequentie waarmee de condensator kan worden opgeladen en ontladen. Het omgekeerde van de frequentie is in feite de laadtijd bij een ideale constante laadspanning. Boven deze frequentie is de condensator niet meer volledig opgeladen (ten opzichte van de maximale spanning van het signaal).

Bij f_{RC} vertoont het capaciteitsspectrum (figuur 4) van de supercondensator een 'schouder'. Onder deze frequentie kan de capaciteitswaarde uit het diagram worden afgeleid. Boven f_{RC} vertoont het impedantiespectrum in figuur 3 (hieronder) een plateau bij R_{ESR} .

De karakteristieke frequentie van het LC-element f_{LC} is de frequentie waarbij de koppeling van parasitaire inductantie en capaciteit leidt tot resonant gedrag wanneer $f_{\text{RC}} > f_{\text{LC}}$, zoals weergegeven in figuur 3 (boven). Onder deze frequentie gedraagt de condensator zich capacitief, dat wil zeggen dat hij elektrische lading kan opslaan; boven deze frequentie gedraagt de condensator zich inductief. De eigenresonantie leidt tot een scherp minimum in het impedantiespectrum (WCAP-FTBE), zoals weergegeven in figuur 3 (boven). De R_{ESR} -waarde kan worden afgelezen uit het minimum van het impedantiespectrum. Condensatoren horen in toepassingen niet worden gebruikt bij f_{LC} of daarboven.

Het capaciteitsspectrum van de 4,7 μF -condensator FTBE in figuur 4 vertoont één pool. Dit is een echt fysisch verschijnsel en niet slechts een meetartefact. Het meetsysteem, dat bestaat uit de condensator en de parasitaire spoel, gedraagt zich als een oscillator of resonantiekering.

De fysische processen worden in detail uitgelegd in de application note ANP109 [2].

f_{Leak} is de karakteristieke frequentie van het $R_{\text{Leak}}\text{-C}$ element. Onder deze frequentie gedraagt de condensator zich als een weerstand met de waarde R_{Leak} . Normaal is dit effect nauwelijks zichtbaar in de spectra. Het vereist ofwel metingen bij frequenties onder 1 Hz ofwel een vrij lage waarde van R_{Leak} .

De karakteristieke frequentie f_{RL} van het $R_{\text{ESR}}\text{-L}$ element is de frequentie waarboven de condensator zich als een spoel met de waarde L_{ESL} gedraagt (figuur 3 onder). Wanneer $f_{\text{RC}} < f_{\text{LC}}$ markeert dit het begin van de toename van de impedantie bij hoge frequenties.

De twee hier besproken voorbeelden illustreren dat een betrekkelijk eenvoudig model kan worden gebruikt om het gedrag van condensatoren met grote en kleine capaciteit te beschrijven. De berekende spectra vertonen in principe alle kenmerken die ook in de gemeten spectra te vinden zijn. Zeker, deze laatste bevatten meer informatie die een uitbreiding van het model vereist; niettemin stelt het LCR-model ons in staat de parameters te bepalen die van belang zijn voor de techniek en de basisinterpretatie van de spectra. De hier gepresenteerde karakteristieke frequenties vormen in dit verband een belangrijk analytisch hulpmiddel, dat kan worden opgevat als quasi-oriëntatiepunten voor de interpretatie van de

gemeten spectra. Een meer gedetailleerde beschouwing van gemeten spectra van verschillende typen condensatoren wordt gegeven in Application Note ANP109 [2]. De overwegingen omvatten de volgende soorten condensatoren:

- supercondensatoren WCAP-STSC
- aluminium elektrolytische condensatoren WCAP-AIGB
- filmcondensatoren WCAP-FTBE
- meerlaags keramische chipcondensatoren WCAP-CSGP

230318-03



Over de auteur

Dr. René Kalbitz studeerde natuurkunde aan de Universiteit van Potsdam en aan de Universiteit van Southampton (VK). Na het behalen van zijn diploma deed hij onderzoek en promoveerde hij op organische halfgeleiders en isolatoren aan de Universiteit van Potsdam. Hij deed verdere ervaring op in toegepast onderzoek aan het Fraunhofer-instituut voor toegepast polymeeronderzoek. Hij kwam in 2018 bij Würth Elektronik in dienst als productmanager voor supercondensatoren en houdt toezicht op onderzoeks- en ontwikkelingsprojecten op het gebied van condensatoren.



WEBLINKS

[1] REDEXPERT online-simulator: <https://redexpert.we-online.com/redexpert/>

[2] René Kalbitz, Impedance spectra of different capacitor technologies (Würth Elektronik AppNote ANP109): <https://www.we-online.com/en/support/knowledge/application-notes?d=anp109-impedance-spectra-of-different-capacitor-technologies>

Een NTP-klok met CircuitPython

waarom zou je deze programmeertaal gebruiken?



Michael Bottin (Frankrijk)

We nemen een duik in een van de nuttige alternatieven voor Python die licht genoeg zijn om op microcontrollers te draaien: CircuitPython. Hier bouwen we een project rond een Raspberry Pi Pico W, dat de tijd ophaalt van een NTP-server en deze weergeeft op een LCD.

Python is een van de populairste programmeertalen. Omdat het een geïnterpreteerde taal is, maakt het de ontwikkelingsfase veel sneller, in tegenstelling tot talen als C/C++, waar je je programma moet compileren voordat je het kunt uitvoeren. Aan de andere kant betekent dit niet-compileren dat vooraf een uitvoeringsomgeving op het target moet worden geïnstalleerd.

Al meer dan 30 jaar is Python in de voorste gelederen bij computers en cloud computing. Het bestrijkt verschillende gebieden zoals wetenschappelijke berekeningen, webontwikkeling (back-end), systeemscripting, machine learning en big data (gegevensanalyse en visualisatie). Python wordt door veel bedrijven gebruikt, onder andere door Google ("Python waar het kan, C++ waar het moet"), YouTube, Instagram, Spotify, Intel, Facebook, Dropbox, Netflix, Pixar en Reddit.

Het is dan ook geen verrassing dat Python zijn weg heeft gevonden naar de wereld van de embedded elektronica. Maar om dat mogelijk te maken, moest er een lichtgewicht versie komen die op een microcontroller kon draaien. Maar die versie moest

ook hardware-georiënteerd zijn om de verschillende periferie te gebruiken die bij een microcontroller beschikbaar is (GPIO, PWM, I²C, SPI, UART...).

Er zijn twee versies ontwikkeld:

- MicroPython, door Damien George in 2013 [1] gemaakt;
- CircuitPython, gemaakt door Limor Fried (Adafruit CEO), Scott Shawcroft en vele anderen [2].

Deze twee talen delen dezelfde implementatie van de programmeertaal Python (CPython). CircuitPython is een open-source fork van MicroPython. Elke evolutie van MicroPython heeft directe gevolgen voor CircuitPython. Voor zover ik weet zijn er in dit tijdschrift al verschillende artikelen verschenen over het gebruik van MicroPython, maar geen over het gebruik van CircuitPython.

Voordelen van CircuitPython boven MicroPython?

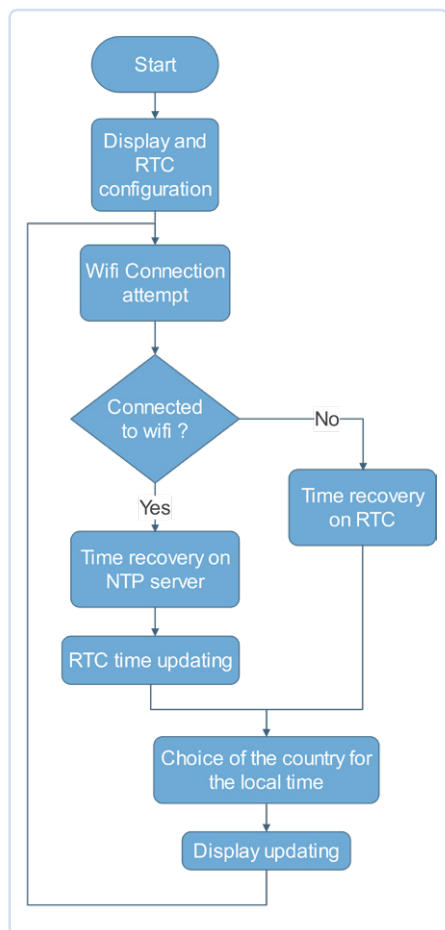
Laten we eerlijk zijn, als je als IT-specialist al bekend bent met Python en je wilt deze

taal gebruiken in de wereld van de embedded elektronica, dan is CircuitPython niet interessant voor je. In dat geval is het eigenlijk veel beter om MicroPython te gebruiken. Ben je echter een maker, student of docent zonder veel kennis van Python, dan kan CircuitPython een interessante en geschikte keuze zijn. CircuitPython biedt namelijk niet alleen een gebruiksvriendelijk ontwikkeltraject, maar ook een extra abstractielaag. Opmerkelijk is ook dat de meeste informatie over deze taal online te vinden is via bibliotheken, tutorials en forums.

Daarom heb ik voor dit artikel CircuitPython gekozen. Ik wilde een eenvoudige aanpak bieden ten behoeve van lezers die Python op microcontrollers willen gaan gebruiken. De CircuitPython-community is bovendien erg actief. Er worden regelmatig nieuwe versies uitgebracht (de huidige versie is 8.x) en nieuwe bibliotheken, en veel op microcontrollers gebaseerde boards ondersteunen deze taal. CircuitPython kan ook worden uitgevoerd op SBC's zoals Raspberry Pi, BeagleBone, Odroid of Jetson met behulp van Blinka API.

Doel van dit artikel

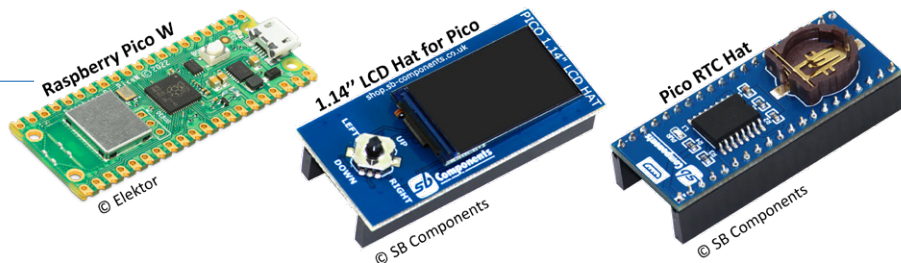
Dit artikel is bedoeld om de lezers een kans te geven CircuitPython te ontdekken en vertrouwd te raken met deze taal door middel van een eenvoudige toepassing. Ik zal je laten zien hoe je met CircuitPython heel eenvoudig een internet-gesynchroniseerde NTP-klok kunt maken. Met dit project kun je ook de lokale tijd kiezen voor meer dan twintig landen over de hele wereld. De klok heeft een WiFi-verbinding nodig (SSID en wachtwoord vooringesteld)



Figuur 1. Algemeen flowdiagram van de NTP-klok.

om de tijd te synchroniseren, en een speciale NTP-server (Network Time Protocol) om de timestamp op te halen.

Deze klok kan de tijd ook bijhouden als er geen netwerk beschikbaar is, dankzij een ingebouwde real-time klok (RTC), die de tijd opslaat als het netwerk beschikbaar is of zelf de tijd aangeeft als er geen netwerkverbin-



Figuur 2. De gebruikte modules.

ding is. Een batterij zorgt ervoor dat de RTC up-to-date blijft wanneer de voeding uitvalt. Het algemene flowdiagram van de klok is getekend in **figuur 1**.

Presentatie van het materiaal

Veel ontwikkelboards (en dus ook microcontrollers) zijn gemakkelijk te programmeren met CircuitPython. Ik heb gekozen voor de Raspberry Pico W omdat het een bekend, goedkoop board is dat in de meeste shops, waaronder die van Elektor [3], verkrijgbaar is en omdat het een WiFi-interface bezit. Daar staat tegenover dat het geen display of een nauwkeurige real-time klok heeft, wat betekent dat extra componenten nodig zijn voor ons project.

In plaats van een eigen board te ontwikkelen, besloot ik voor dit project kant-en-klaar verkrijgbare modules te gebruiken. Drie op elkaar geprikte modules volstaan voor het project (zie **figuur 2**):

- een Raspberry Pico W-module (gebruik in elk geval de W-versie, die de WiFi-interface heeft);
- een displaymodule met een LCD met een resolutie van 240 × 135 en een

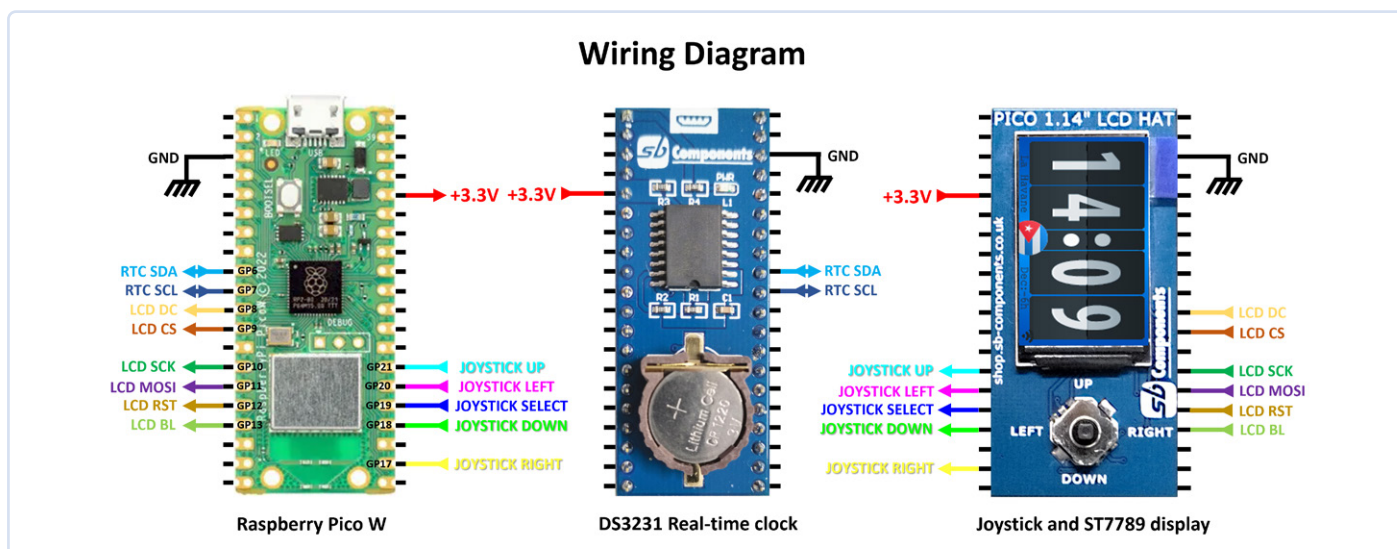
joystick-controller voor de navigatie (SB Components [4]);

- een DS3231 real-time klokmodule (SB Components [5]).

Opmerking: de RP2040-microcontroller heeft ook een interne real-time klok die in plaats van de Pico RTC Hat-module kan worden gebruikt. Maar, zoals gezegd, de RP2040 is niet zo nauwkeurig en heeft geen back-up batterij.

Bedradingsschema

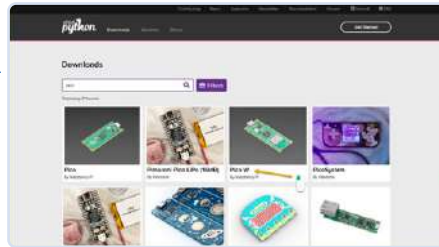
Zelfs als we kant-en-klare modules gebruiken die de hardware-implementatie vereenvoudigen, moeten we de verbindingen tussen de modules kennen om de programmaregels van onze toepassing te kunnen schrijven. **Figuur 3** toont de diverse verbindingen tussen de verschillende modules, evenals de namen die aan deze verbindingen zijn gegeven en die we in het programma opnieuw zullen gebruiken. (Merk op dat de pinout van SB Components die in hun GitHub-documentatie wordt vermeld, op het moment van schrijven onjuist is, maar hun broncode verwijst naar de correcte pinnen in **figuur 3**).



Figuur 3. De verbindingen tussen de modules.



Figuur 4. De CircuitPython-webpagina.



Figuur 5. Selecteer de Pico W uit de CircuitPython-compatible boards.



Figuur 6. Download het .UF2-bestand.

Installatie van CircuitPython

De RP2040-microcontroller van de Raspberry Pi Pico W is niet voorzien van CircuitPython wanneer je de module koopt, dus moet je hem eerst op het board 'flashen'. Net zoals er verschillende C-compiler-versies zijn afhankelijk van het target dat je gebruikt, zijn er evenveel CircuitPython-versies als boards die het ondersteunen. Op dit moment zijn er meer dan 380 CircuitPython-compatible commerciële boards. Veel van de meest recente boards van Adafruit zijn uiteraard compatibel, maar veel andere zijn dat ook, zoals de boards van Pimoroni, Expressif, Seeed-studio, Waveshare, LilyGo, Cytron, DFRobot en Wiznet.

De meeste gebruiken microcontrollers van Microchip (SAMD21/SAMD51), Nordic (nRF52840), Expressif (ESP32) of van de Raspberry Pi Foundation (RP2040). Je moet de passende Raspberry Pico W CircuitPython-versie downloaden:

- ga in je browser naar de website van CircuitPython [6] (figuur 4);
- klik op de link **Downloads** en zoek naar het Pico W-board (figuur 5);
- download het UF2-bestand van de getoonde pagina (figuur 6).

De RP2040-microcontroller van de Raspberry Pi Pico bevat al een bootloader die de installatie van CircuitPython vergemakkelijkt. De onderstaande instructies tonen hoe je de gedownloade CircuitPython-versie kunt installeren.

1. Ontkoppel het Raspberry Pico W-board van je computer.
2. Houd de BOOTSEL-knop van het Raspberry Pico W-board ingedrukt terwijl je het board weer op je computer aansluit via een micro-USB-kabel die geschikt is voor gegevensoverdracht.
3. Er moet nu in de file explorer een nieuw station met de naam RPI-RP2 verschijnen.
4. Laad het gedownloade CircuitPython UF2-bestand in RPI-RP2.

Dat is alles. Zodra CircuitPython met succes op het Raspberry Pico W bord is geflasht, moet er een **CIRCUITPY**-drive verschijnen.

Het gebruik van de CIRCUITPY-drive

Hier volgen enkele belangrijke zaken waar je op moet letten bij het gebruik van het **CIRCUITPY**-station:

- Het station gedraagt zich als een USB-station. Je kunt bestanden/mappen toevoegen of verwijderen vanuit een file explorer. De capaciteit is die van het flash-geheugen dat beschikbaar is op de RP2040-microcontroller om je code en resources (afbeeldingen, audio...) op te slaan.
- Je kunt het Raspberry Pico W-board op uw computer aansluiten zonder bijzondere voorzorgsmaatregelen te nemen. **Aan de andere kant verdient het sterke aanbeveling om het uit te werpen als een USB-station wanneer je het wilt ontkoppelen om geen risico te lopen dat het bestandssysteem beschadigd raakt.**
- Nadat je het bestand *boot.txt* hebt geopend, kun je controleren welke versie van CircuitPython op het Raspberry Pico W-board is geïnstalleerd.
- Zodra het Raspberry Pico W-board via de USB-poort wordt gevoed (door je computer of door een DC-voeding), draait de CircuitPython-code. Maar alleen bestanden met de naam *code.py* of *main.py* worden uitgevoerd; zorg ervoor dat je je bestand voor dit project de naam *code.py* geeft.
- Het is raadzaam (maar niet verplicht) om de afbeeldingen te kopiëren naar een directory *./images*, de bibliotheekbestanden naar een directory *./lib* enzovoort. Dit helpt om je projecten in een overzichtelijke boomstructuur te organiseren.

Gebruik van de IDE

CircuitPython is een geïnterpreteerde taal, dus er hoeft niet gecompileerd te worden. Als er een *code.py*-bestand met CircuitPython-code aanwezig is op de **CIRCUITPY**-drive, dan wordt het automatisch uitgevoerd. Dat betekent dat we dat bestand gewoon in een tekstverwerker kunnen bewerken en opslaan voordat we het uitvoeren, maar vergeet niet: programma's schrijven gaat vaak hand in hand met debuggen. Het gebruik van een IDE geeft je toegang tot hulpmiddelen zoals een console om feedback te krijgen over fouten.

De IDE die we voor dit project zullen gebruiken is Mu Editor [7]. Het is een eenvoudige gratis softwareoplossing die beschikbaar is voor Windows, Mac OSX en Linux. (Je kunt ook Thonny, VS Code, Atom of PyCharm gebruiken als je de CircuitPython-extensie installeert).

Figuur 7 toont de interface van de Mu Editor IDE. De werkbalk is minimalistisch:

- **Mode:** keuze van de programmeertaal. Zorg ervoor dat je de CircuitPython-modus selecteert.
- **New:** aanmaken van een nieuw, leeg bestand.
- **Load:** opent een bestaand bestand. Als de CircuitPython-modus is geselecteerd en je Raspberry Pico W-board correct is aangesloten, zou de **CIRCUITPY**-drive automatisch in het dialoogvenster moeten verschijnen.
- **Save:** slaat het huidige bestand op (het actieve tabblad). Een kleine rode stip naast de bestandsnaam in het tabblad geeft aan dat het bestand is gewijzigd sinds de laatste back-up.
- **Serial:** toont/verbergt de console. Ik raad je aan de console (REPL) altijd weer te geven, omdat deze foutmeldingen toont en wat je wilt afdrukken van de code die op de Pico draait.
- **Plotter:** toont/verbergt een scrollende grafiek met digitale gegevens die je code kan aanmaken.
- **Zoom-in** en **Zoom-out:** maakt het lettertype groter of kleiner.



Figuur 7. De interface van de Mu Editor IDE.

- **Theme**: schakelt tussen drie verschillende displaymodi: *Day* (licht thema), *Night* (donker thema) en *High-Contrast*.
- **Check**: zoekt naar fouten in de code, zelfs als deze geen invloed hebben op de werking ervan.
- **Tidy**: ruimt je code op door bijvoorbeeld nutteloze spaties of lege regels te verwijderen.
- **Help**: Online-helfunctie.
- **Stop**: Sluit de editor.

- Bekijk het resultaat van de uitvoering in de seriële console. Ga terug naar stap 2 als er tijdens de uitvoering een fout is opgetreden.

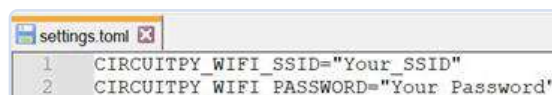
Meer informatie vind je via deze links:

- Begin hier! (over de interface) [8].
- Wat is een REPL? (de console) [9].
- Data plotten met Mu [10].
- Sneltoetsen [11].

Hiermee heb je alles om je project te starten. Dit is waar het nu om gaat: de NTP-klok.

Deze stappen moeten dus worden gevolgd om een programma te ontwikkelen in de IDE:

- Open het `code.py`-bestand van het CIRCUITPY-station.
- Wijzig de code in het editorvenster.
- Sla de wijzigingen in het bestand op.



Figuur 8. SSID en wachtwoord in settings.toml.



Listing 1. Code voor de WiFi-verbinding.

```
1 # CircuitPython's own libraries
2 import os
3 import ipaddress
4 import wifi
5
6 print()
7 print("Connecting to Wi-Fi")
8
9 # connect to your Wi-Fi network with SSID/PASSWORD in 'settings.toml'
10 wifi.radio.connect(os.getenv('CIRCUITPY_WIFI_SSID'), os.getenv('CIRCUITPY_WIFI_PASSWORD'))
11
12 print("Connected to Wi-Fi")
13
14 # pings Google DNS server
15 ipv4 = ipaddress.ip_address("8.8.8.8")
16 print("Ping google.com: %f ms" % (wifi.radio.ping(ipv4)*1000))
```

WiFi-verbindingstest

CircuitPython ondersteunt native WiFi op het Raspberry Pico W-board, dus je hoeft geen extra bibliotheken te installeren. Om verbinding te maken met WiFi moet je de SSID en het wachtwoord van je netwerk opgeven. Om te voorkomen dat deze informatie direct in de code moet worden opgenomen, maakt CircuitPython gebruik van omgevingsvariabelen.

In de root van CIRCUITPY staat een bestand `settings.toml` [12] en deze gegevens blijven 'geheim'. Alle vertrouwelijke gegevens, zoals API-keys, kunnen in dit bestand worden opgeslagen; in ons geval bevat het alleen de SSID en het wachtwoord van het netwerk waarmee je je Raspberry Pico W-board wilt verbinden (zie **figuur 8**). (Opmerking: dit bestand kan niet bewerkt worden in de IDE; je moet een tekstverwerker gebruiken om het te maken en in te vullen.)

De code om verbinding te maken met het netwerk is gegeven in **listing 1**. Je ziet meteen al dat de code erg compact is:

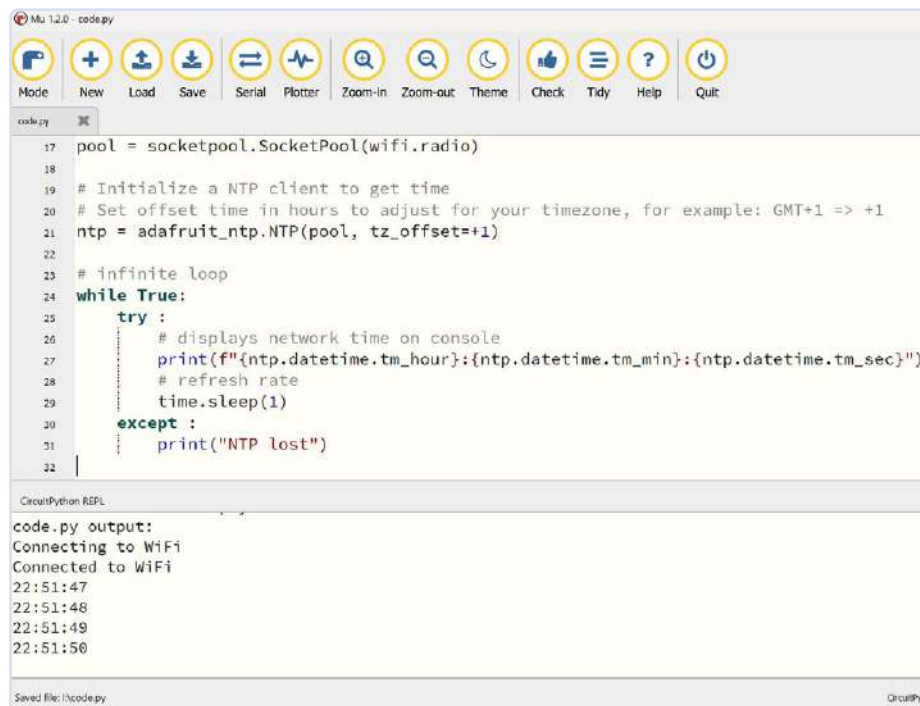
Figuur 9. Downloaden van de bibliotheken (de datum in de bestandsnaam zal natuurlijk anders zijn).

- **Regel 2:** importeren van de `os`-bibliotheek, waarmee de omgevingsvariabelen van het bestand `settings.toml` kunnen worden gelezen.
- **Regel 4:** importeren van de `wifi`-bibliotheek, die de netwerkverbinding mogelijk maakt.
- **Regels 6-7, 12:** deze regels zijn niet noodzakelijk, maar geven de gebruiker informatie via de seriële console.
- **Regel 10:** deze ene regel is voldoende om de fysieke netwerkverbinding tot stand te brengen.
- **Regels 3, 14-16:** deze regels zijn niet noodzakelijk, maar ze stellen je in staat een test (ping) uit te voeren door een server te ondervragen en zo de status van je WiFi-verbinding te controleren, en vervolgens het resultaat uit te voeren naar de seriële console.

Nuttige bibliotheken toevoegen

De meeste basisbibliotheken zijn te vinden op CircuitPython (en niet alle beschikbare bibliotheken, anders zou het flash-geheugen van de microcontroller overbelast worden). Het is dus noodzakelijk, afhankelijk van de projecten waar je aan werkt, om enkele bibliotheken toe te voegen. Het installeren van extra bibliotheken in CircuitPython bestaat gewoon uit het kopiëren van de bijbehorende bestanden naar de CIRCUITPY-drive. Installeer alle bibliotheken die nuttig kunnen zijn voor de uiteindelijke versie van het project, zodat je het maar één keer hoeft te doen. Zo kun je ze installeren:

1. Maak een map `./lib` aan op je CIRCUITPY-drive (als die nog niet bestaat).
2. Alle beschikbare bibliotheken kunnen als één enkel archief worden gedownload van de CircuitPython-website (opmerking: ze zijn ook afzonderlijk beschikbaar op GitHub).
3. Klik op de website op de link *Libraries*.
4. Download de bundel die overeenkomt met jouw CircuitPython-versie: in ons geval versie 8.x (zie **figuur 9**).
5. Pak het uit naar een willekeurige locatie op je computer.
6. Open de map `lib` in de uitgepakte bundel.
7. Selecteer:
 - a. deze mappen:
 - i. `adafruit_register`



Figuur 10. NTP-klok weergegeven in de seriële console.

- ii. `adafruit_imageload`
- iii. `adafruit_display_text`
- b. deze bestanden:
 - i. `adafruit_debouncer.mpy`
 - ii. `adafruit_ds3231.mpy`
 - iii. `adafruit_ntp.mpy`
 - iv. `adafruit_st7789.mpy`
 - v. `adafruit_ticks.mpy`
8. Kopieer de selectie naar de map `./lib` op de CIRCUITPY-drive.

Opmerking: Circup is een tool (geschreven in Python) dat de versie van je bibliotheken kan controleren, ze kan bijwerken en hun dependencies kan installeren [13].

Tijdsweergave in de console

In de bovenstaande code heb je je Raspberry Pico W-board verbonden met het WiFi-netwerk. Nu ga je een speciale NTP-server bevragen zodat deze je de tijd en datum levert. De code waarmee je de tijd kunt ophalen en weergeven in de console (**figuur 10**) is gegeven in **listing 2**. Laten we eens kijken naar de regels die zijn toegevoegd:

- **Regels 5 en 18:** importeren van de `socketpool` native library en aanmaken

van een socket die de client/server-verbinding verzorgt.

- **Regel 22:** instantie van een NTP-cliënt via de socket. De parameter `tz_offset` wordt gebruikt om het tijdsverschil tussen UTC (Coordinated Universal Time [14]) en je eigen tijdzone in te stellen. In Frankrijk gebruiken we CET, wat eigenlijk UTC+1 is, vandaar de waarde `+1`. Je moet deze aanpassen voor je eigen tijdzone. De standaard server is de Adafruit-server (maar dat kan worden aangepast) en de standaard time-out is 10 seconden.
- **Regels 24-32:** in een oneindige lus wordt de tijd in de console weergegeven met de gewenste verversingssnelheid (in ons geval 1 seconde – regel 29). Hiertoe gebruiken we de parameter `ntp.datetime`. Deze parameter retourneert een tuple met naam van de tijds-klasse die de datum- en tijdinformatie van de server bevat. We benaderen elk element van de tuple via zijn naamveld [15]. We gebruiken de `try...except` constructie om een uitzondering af te vangen die betekent dat de gegevens niet zijn ontvangen zodra de time-out is bereikt.



Listing 2. NTP-request code.

```

1 # CircuitPython own's libraries
2 import time
3 import os
4 import wifi
5 import socketpool
6
7 # CircuitPython external libraries
8 import adafruit_ntp
9
10 print("Connecting to WiFi")
11
12 # connect to your WiFi network with SSID/PASSWORD in 'settings.toml'
13 wifi.radio.connect(os.getenv('CIRCUITPY_WIFI_SSID'), os.getenv('CIRCUITPY_WIFI_PASSWORD'))
14
15 print("Connected to WiFi")
16
17 # Setting up a socket
18 pool = socketpool.SocketPool(wifi.radio)
19
20 # Initialize a NTP client to get time
21 # Set offset time in hours to adjust for your timezone, for example: GMT+1 => +1
22 ntp = adafruit_ntp.NTP(pool, tz_offset=+1)
23
24 # infinite loop
25 while True:
26     try :
27         # displays network time on console
28         print(f"::")
29         # refresh rate
30         time.sleep(1)
31     except :
32         print("NTP lost")

```



Figuur 11. NTP-klok weergegeven op het LCD.

Tijdsweergave op LCD-scherm

In deze stap gaan we het display gebruiken in plaats van de console om de tijd weer te geven. CircuitPython beschikt over een native bibliotheek met de naam *displayio*. Deze bibliotheek biedt methoden en gemeenschappelijke attributen voor elk display dat door CircuitPython wordt ondersteund.

De kleuren-LCD-module die we hier gebruiken heeft de volgende features:

- > SPI-protocol
- > resolutie 240 × 35 pixels
- > ST7789-controller

Je kunt de code downloaden via de link aan het eind van dit artikel. Het zou lang duren om alles in detail uit te leggen, maar dit zijn de belangrijkste stappen om een resultaat te bereiken zoals getoond in **figuur 11**. Je moet verschillende bibliotheken importeren:

1. Native bibliotheken van CircuitPython:
 - board*: voor toegang tot de namen van de Raspberry Pico W-pinnen
 - displayio*: voor het beheren van de graphics
 - busio*: voor de SPI-bus communicatie
 - terminalio*: om een lettertype voor het scherm te krijgen
2. Externe bibliotheken:
 - adafruit_st7789*: om het LC-display aan te sturen
 - adafruit_display_text*: om het tekstgebied op het display weer te geven

Vervolgens moet je de display-instellingen configureren:

1. Maak de SPI-communicatiebus *SPI_bus*.
2. Maak de bus *display_bus* om het Raspberry Pi Pico W-board te verbinden met het LCD. Hij bevat de SPI-bus plus twee extra signalen (*Command* en *chip_select*).

3. Maak het scherm *display* door de vorige bus beschikbaar te maken, door de resolutie ervan te definiëren (*width=135*, *height=240*) en door de offsets in x (*rowstart=40*) en y (*colstart=53*) aan te passen. De waarden van deze laatste twee parameters komen voort uit het feit dat de hoogste resolutie van de ST7789-driver 320 x 240 bedraagt, terwijl de resolutie van ons scherm slechts 240 x 135 bedraagt (zie **figuur 12**).
4. In CircuitPython moet elk grafisch element tot een groep behoren. Maak een groep met de naam *display_group* en maak een *time_label* tekstveld. Voeg het tekstveld toe aan de groep. Geef tenslotte de groep weer op het scherm.
5. Om de tijd op het scherm bij te werken, hoef je alleen maar in de oneindige lus het *text* attribuut van ons tekstveld-object *time_label* te wijzigen door de tekenreeks te kopiëren die we gebruiken voor weergave in de console.



Listing 3. RTC-code.

```
1 # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2 # SPDX-License-Identifier: MIT
3
4 # Simple demo of reading and writing the time for the DS3231 real-time clock.
5 # Change the if False to if True below to set the time, otherwise it will just
6 # print the current date and time every second. Notice also comments to adjust
7 # for working with hardware vs. software I2C.
8
9 import time
10 import board
11 import busio
12 import adafruit_ds3231
13
14 i2c = busio.I2C(scl=board.GP7, sda=board.GP6)
15 rtc = adafruit_ds3231.DS3231(i2c)
16
17 # Lookup table for names of days (nicer printing).
18 days = ("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")
19
20 # pylint: disable-msg=using-constant-test
21 if True: # change to True if you want to set the time!
22     # year, mon, date, hour, min, sec, wday, yday, isdst
23     t = time.struct_time((2023, 03, 09, 14, 58, 15, 3, -1, -1))
24     # you must set year, mon, date, hour, min, sec and weekday
25     # yearday is not supported, isdst can be set but we don't do anything with it at this time
26     print("Setting time to:", t) # uncomment for debugging
27     rtc.datetime = t
28     print()
29 # pylint: enable-msg=using-constant-test
30
31 # Main loop:
32 while True:
33     t = rtc.datetime
34     # print(t) # uncomment for debugging
35     print(
36         "The date is {} {}/{} / {}".format(
37             days[int(t.tm_wday)], t.tm_mday, t.tm_mon, t.tm_year
38         )
39     )
40     print("The time is {}:02:02".format(t.tm_hour, t.tm_min, t.tm_sec))
41     time.sleep(1) # wait a second
```

Test van de real-time klok

Onze webgesynchroniseerde klok werkt perfect – houden we het dus voor gezien? Nee, we zijn nog niet klaar, want wat zou er gebeuren als de netwerkverbinding werd onderbroken? Of als de stroom tijdelijk uitvalt?

Om te voorkomen dat de huidige tijd verloren gaat, voegen we een real-time klok toe. Gekoppeld aan een back-up batterij (CR1220) zal deze de tijd jarenlang bijhouden, zelfs als de klok niet gevoed wordt via de micro-USB-poort. Deze RTC gebruikt de bekende DS3231, en die gebruikt de I²C-bus om te

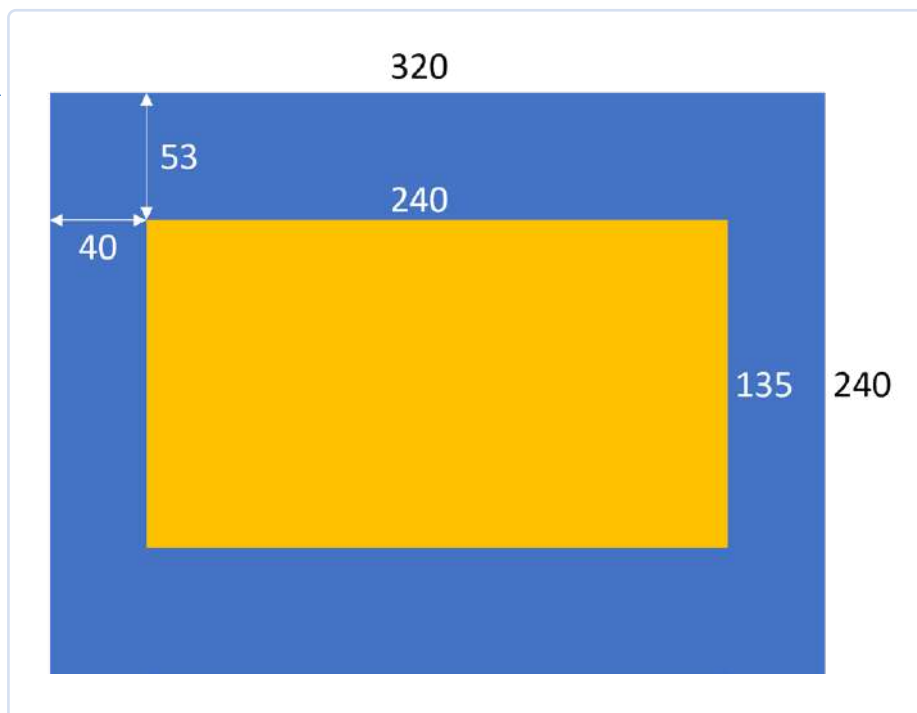
communiceren met de Raspberry Pi Pico W. Alvorens de RTC te combineren met de NTP-server gaan we die separaat testen, en daarvoor heb je de `adafruit_ds3231` bibliotheek nodig. Zoals de meeste CircuitPython-bibliotheken biedt deze online tutoriële en voorbeelden [16].

In de code die in deze tutorial wordt gepresenteerd, hoef je alleen de eerste regels aan te passen zoals getoond in **listing 3**. Veel Raspberry Pi Pico W's ondersteunen het I²C-protocol namelijk niet [17]. Je moet precies weten welke fysiek verbonden zijn met het DS3231 RTC-IC (zie figuur 3). Bij de

uitvoering zou je iets vergelijkbaars moeten krijgen als in **figuur 13**.

Definitieve code

Het primaire doel van ons project is bereikt. De link om de definitieve code te downloaden is te vinden in [24]. **Figuur 14** toont de algemene interface van het display. Je ziet dat het grafische aspect is verbeterd en dat er nieuwe elementen op het scherm zijn verschenen. Nogmaals, een gedetailleerde uitleg zou langdradig worden, maar dit artikel verduidelijkt alle noodzakelijke basisprincipes [18].



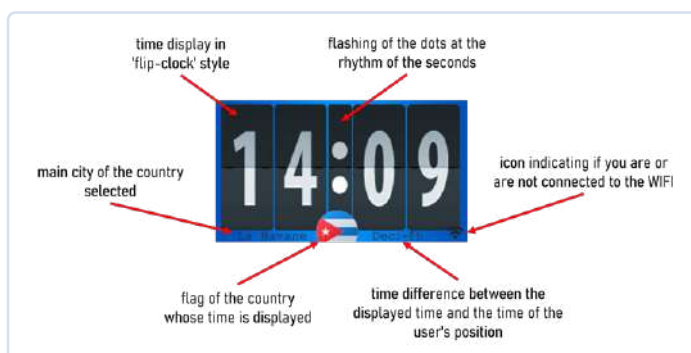
Figuur 12. ST7789 display-offsets.

```

code.py
24 t = time.struct_time((2023, 03, 09, 14, 58, 15, 3, -1, -1))
25 # you must set year, mon, date, hour, min, sec and weekday
26 # year/day is not supported, isdst can be set but we don't do anything with it at this time
27 print("Setting time to:", t) # uncomment for debugging
28 rtc.datetime = t
29 print()
30 # pylint: enable-msg=using-constant-test
31
32 # Main loop:
33 while True:
34     t = rtc.datetime
35     # print(t) # uncomment for debugging
36     print(
37         "The date is {} {}/{}{}".format(
38             days[int(t.tm_wday)], t.tm_mday, t.tm_mon, t.tm_year
39         )
40     )
41     print("The time is {}:02:02".format(t.tm_hour, t.tm_min, t.tm_sec))
42     time.sleep(1) # wait a second
43
CircuitPython REPL
The date is Thursday 9/3/2023
The time is 15:00:11
The date is Thursday 9/3/2023
The time is 15:00:12
The date is Thursday 9/3/2023
The time is 15:00:13
The date is Thursday 9/3/2023
The time is 15:00:14
The date is Thursday 9/3/2023
The time is 15:00:15

```

Figuur 13. RTC-resultaat.



Figuur 14. De definitieve interface.



Figuur 15. Het cijfer-sprite sheet.

Wat zijn de verschillen met de vorige listings?

- > De code om de real-time klok (RTC) te beheren is gecombineerd met de code die het mogelijk maakt de tijd op te halen van de NTP-server.
- > De tijd wordt nu weergegeven met behulp van bitmap-afbeeldingen die eerder in speciale code zijn voorbereid. We moeten de `adafruit_imageload` bibliotheek gebruiken om ze in het geheugen te laden. Om te vermijden dat bij elke cijferwisseling een afbeelding wordt geladen (dat geeft een risico van vertraging), moeten we onze toevlucht nemen tot een 'sprite sheet' zoals ook wordt gebruikt bij animaties (zie **figuur 15**). Vervolgens moeten we bepalen welk deel van de afbeelding weergegeven moet worden. Dezelfde procedure kan worden toegepast op de knipperende punten.
- > Met de joystick op de 1,14" LCD Hat for Pico-module kunnen we een land selecteren en de huidige tijd in dit land en het tijdsverschil met UTC visualiseren. Om contactdender van de joystick te voorkomen, gebruiken we de `adafruit_debouncer` bibliotheek, die op zijn beurt de `adafruit_ticks` bibliotheek nodig heeft.
- > De weergave van de vlag maakt ook gebruik van een bitmap in de vorm van een sprite sheet met 25 landen (zie **figuur 16**).



Figuur 16. Het vlaggen-sprite sheet.



Figuur 17. Boomstructuur van de bibliotheken.

- De teksten die overeenkomen met de stad en het tijdsverschil gebruiken labels, net zoals we eerder deden om de tijd op het LCD weer te geven.

De boomstructuur van de in dit project gebruikte interne en externe bibliotheken is geschetst in **figuur 17**.

Conclusie

Er zijn natuurlijk veel manieren om dit project te verbeteren, zoals het gebruik van een LC-display met een hogere resolutie (zonder echter 320 × 240 te overschrijden, want de Raspberry Pi Pico W moet het wel kunnen aansturen), het toevoegen van hoorbare alarmen of het weergeven van actuele weergegevens voor je locatie... Deze verbeteringen kunnen gemakkelijk worden gecodeerd in CircuitPython dankzij de resources die beschikbaar zijn via online-bibliotheken, of de vele tutorials die te vinden zijn op de Adafruit-website. En waarom probeer je niet je eigen CircuitPython-project te programmeren en aan den lijve te ondervinden hoe gemakkelijk het is om deze taal te gebruiken?

- Download: CircuitPython voor de Raspberry Pi Pico W [18].
- Download: externe bibliotheken [19].
- Documentatie: CircuitPython native bibliotheken [20].
- Documentatie: externe bibliotheken [21].

- CircuitPython Essentials tutorials [22] [23].

Je kunt ook andere resources vinden in een (Franstalig) boek dat ik heb geschreven over CircuitPython versie 5.3 (zie **figuur 18**). Het is gepubliceerd door Elektor in 2020, dus sinds die editie is er natuurlijk veel vernieuwd, maar de basis van de taal blijft hetzelfde en de code kan gemakkelijk worden geport naar andere modules zoals de Raspberry Pi Pico W (wat een van de belangrijkste pluspunten is van CircuitPython en zijn 'unified hardware API!'). En als je niet verder komt, kun je me per e-mail je vragen sturen! ◀

220633-03



Figuur 18. Het boek "Initiation au langage CircuitPython et à la puce nRF52840".

Vragen of opmerkingen?

Als u technische vragen hebt of gewoon nieuwe ideeën voor een artikel wilt aandragen, kunt u per e-mail contact opnemen met de auteur via michael.bottin@univ-rennes.fr, of met de Elektor-redactie via redactie@elektor.com.



Gerelateerde producten

- **Raspberry Pi Pico RP2040 W**
www.elektor.nl/20224
- **Michael Bottin, Initiation au langage CircuitPython et à la puce nRF52840, (Franstalig)**
www.elektor.nl/19523

WEBLINKS

- [1] MicroPython: <https://en.wikipedia.org/wiki/MicroPython>
- [2] CircuitPython: <https://en.wikipedia.org/wiki/CircuitPython>
- [3] Raspberry Pico W bij Elektor: <https://elektor.nl/raspberry-pi-pico-rp2040-w>
- [4] LCD Hat voor Pico: <https://shop.sb-components.co.uk/products/1-14-lcd-hat-for-pico>
- [5] Pico RTC Hat: <https://shop.sb-components.co.uk/products/pico-rtc-hat>
- [6] Download CircuitPython: <https://circuitpython.org/>
- [7] Mu Editor: <https://codewith.mu/en/download>
- [8] Mu Editor Interface: <https://codewith.mu/en/tutorials/1.2/start>
- [9] REPL-console: <https://codewith.mu/en/tutorials/1.2/repl>
- [10] Scrollende grafiek: <https://codewith.mu/en/tutorials/1.2/plotter>
- [11] Sneltoetsen: <https://codewith.mu/en/tutorials/1.2/shortcuts>
- [12] TOML-bestand: <https://en.wikipedia.org/wiki/TOML>
- [13] Circup-tool: <https://learn.adafruit.com/keep-your-circuitpython-libraries-on-devices-up-to-date-with-circup/>
- [14] UTC (Coordinated Universal Time): <https://timeanddate.com/worldclock/timezone/utc>
- [15] Het type van de tijdwaarde: https://docs.python.org/3/library/time.html#time.struct_time
- [16] adafruit_ds3231-bibliotheek tutorials en voorbeelden: <https://learn.adafruit.com/adafruit-ds3231-precision-rtc-breakout/circuitpython>
- [17] Pico W-pinning: <https://datasheets.raspberrypi.com/picow/PicoW-A4-Pinout.pdf>
- [18] CircuitPython-versie voor de Raspberry Pico W: https://circuitpython.org/board/raspberry_pi_pico_w/
- [19] Externe bibliotheken: <https://circuitpython.org/libraries>
- [20] CircuitPython native bibliotheek documentatie: <https://docs.circuitpython.org/en/latest/shared-bindings/index.html#modules>
- [21] Externe bibliotheken documentatie: <https://docs.circuitpython.org/projects/bundle/en/latest/drivers.html>
- [22] CircuitPython Essentials Tutorials 1: <https://learn.adafruit.com/welcome-to-circuitpython/circuitpython-essentials>
- [23] CircuitPython Essentials Tutorials 2: <https://learn.adafruit.com/welcome-to-circuitpython/>
- [24] Software-download: <https://elektormagazine.nl/220633-03>

Word lid van de Elektor Community

- ✓ Een compleet web-archief t/m 1980!
- ✓ 8x Elektor Magazine (Print)
- ✓ 8x digitaal (PDF)
- ✓ 10% korting in onze webshop, en exclusieve aanbiedingen
- ✓ Toegang tot meer dan 5000 Gerberfiles



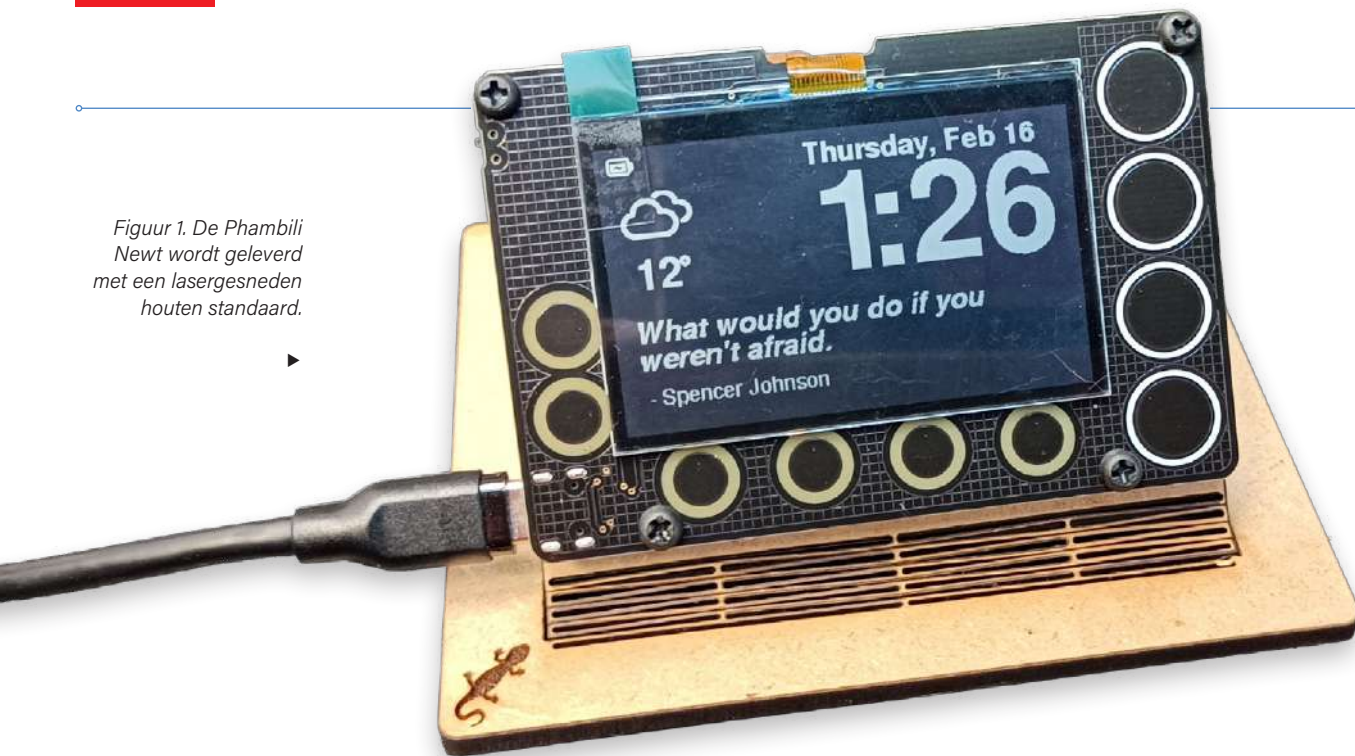
Neem **nu** een
lidmaatschap!



www.elektormagazine.nl/Abonnement



Figuur 1. De Phambili Newt wordt geleverd met een lasergesneden houten standaard.



Bouw een cool IoT-display

met de Phambili Newt

Clemens Valens (Elektor)

Maak kennis met de Phambili Newt, een compacte en configureerbare displaymodule die meer te bieden heeft dan je op het eerste gezicht zou denken. Ontdek de unieke functies, van basisfunctionaliteit tot het opwindende potentieel van door de gebruiker programmeerbare toepassingen, waardoor het een veelbelovend hulpmiddel is voor diegenen die geïnteresseerd zijn in IoT-apparaten.

De Phambili Newt is een batterijgevoed, permanent ingeschakeld display voor wandmontage dat online kan gaan om informatie op het internet te zoeken en weer te geven. De module is iets groter dan een creditcard en heeft 10 touchpads naast een 2,7" e-ink-achtig display van 240x400 pixels. Achter het display zit een ESP32-S2 microcontroller die je kunt programmeren met Arduino, CircuitPython, MicroPython of ESP-IDF. Hoewel het display bedoeld is om altijd ingeschakeld te zijn, heeft het een kleine aan/uit-schakelaar – uitschakelen is dus mogelijk.

Een (oplaadbare) batterij moet de Newt van stroom voorzien, maar die is niet in de kit inbegrepen. Gelukkig kun je hem ook via een USB-C-telefoonoplader laten werken. Volgens de documentatie zou een LiPo-accu van 500 mAh (minimaal) het display tot twee maanden laten werken tussen twee oplaadbeurten. De accu moet een zogenaamde 2-weg JST-connector hebben.

Naast de Newt-module bevat de kit ook een lasergesneden houten standaard en wat bevestigingsmateriaal. Met de standaard kun je het apparaat op bijvoorbeeld je bureau zetten, maar het is een beetje wiebelig bij het aanraken van de touchpads. Een stabielere optie is om de vier magnetische voetjes te monteren en de Newt op een koelkast of een ander metalen oppervlak te bevestigen.

Wat doet de Phambili Newt?

Zo uit de doos doet de Newt niet veel, omdat hij verwacht dat je eerst een WiFi-verbinding configureert. Na het inschakelen van de module geeft hij instructies hoe je hem moet verbinden met je netwerk. Dit is gemakkelijk genoeg, maar gaat een beetje traag, en ik zag verschillende reboots voordat het apparaat verbinding maakte met mijn netwerk. Toen dat uiteindelijk was gelukt, toonde hij de datum en tijd en bleef daarna hangen.

Figuur 2. Met de magnetische voetjes kun je de Newt op een koelkast of ander metalen voorwerp 'plakken' ▶



Ik merkte dat op mijn Newt versie v0.0.11 van de firmware draaide, en dus zocht ik naar een recentere versie. Na het laden van de laatste versie (v1.1.15) werkte de Newt zoals verwacht. Het upgraden van de firmware is eenvoudig: sluit de Newt aan op een computer en kopieer het nieuwe firmware-bestand naar de externe schijf die verschijnt.

Met de juiste firmware maakte de Newt snel verbinding met mijn netwerk en toonde dan tijd, datum en het weer voor mijn locatie. De weergegevens wordt om de drie minuten afgewisseld met een citaat.

Door de touchpad rechtsboven aan te raken, verschijnt een menu onderaan het scherm. Er zijn drie 'pagina's' met dingen als alarm en timer, weer en luchtkwaliteit, een kalender en andere zaken die je misschien nuttig of leuk vindt. Het menu geeft ook toegang tot de instellingen en een upgrade van de firmware. Helaas zit de bevestigingsschroef een beetje in de weg van de menuknop (net als de schroef bij de knop linksonder).

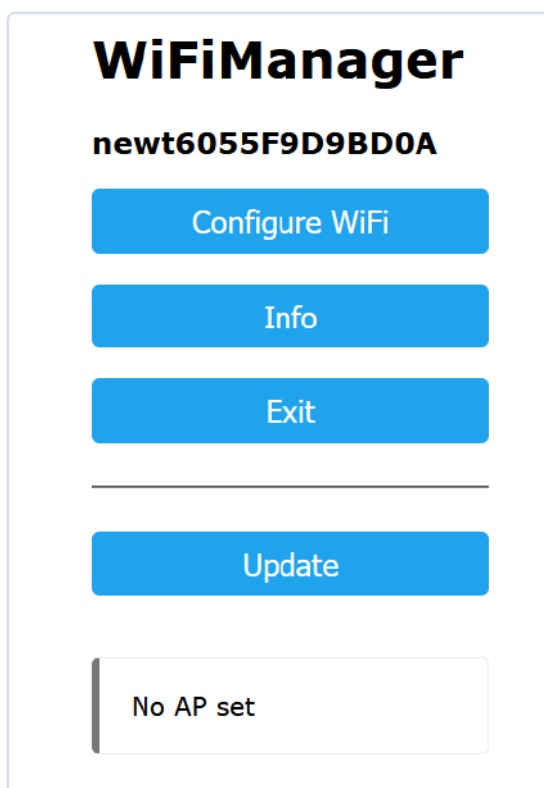
Schrijf je eigen toepassingen voor de Phambili Newt

Hoewel de basisfunctionaliteit van de Newt mooi is, is dat waarschijnlijk niet de reden waarom je er een zou willen hebben. Zijn echte kracht ligt in het feit dat hij als het ware gehackt kan worden. De broncode van de firmware is beschikbaar op GitHub, samen met instructies voor het opzetten van de Arduino IDE voor het schrijven van je eigen Newt-applicaties. Met een I²C-connector in Qwiic-formaat (SparkFun) kun je sensoren en andere uitbreidingen op de Newt aansluiten, waardoor het een echt IoT-apparaat wordt in plaats van alleen maar een klok.

Conclusie

De ESP32-S2 is een krachtige microcontroller die over WiFi beschikt maar niet over Bluetooth. Het zwart-wit display ziet er erg mooi uit en is snel, in tegenstelling tot normale e-ink displays. De beeldopbouw op het display gebeurt vrijwel ogenblikkelijk. De combinatie van de twee zorgt voor een coole module met veel toepassingsmogelijkheden, vooral in het low-power IoT domein. De I²C-uitbreidingsconnector biedt nog verdere mogelijkheden. ◀

230345-03



Figuur 3. Na de eerste keer inschakelen zijn dit je mogelijkheden. ◀

Vragen of opmerkingen?

Hebt u vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur via clemens.valens@elektor.com of naar de redactie van Elektor via redactie@elektor.com.



Gerelateerde producten

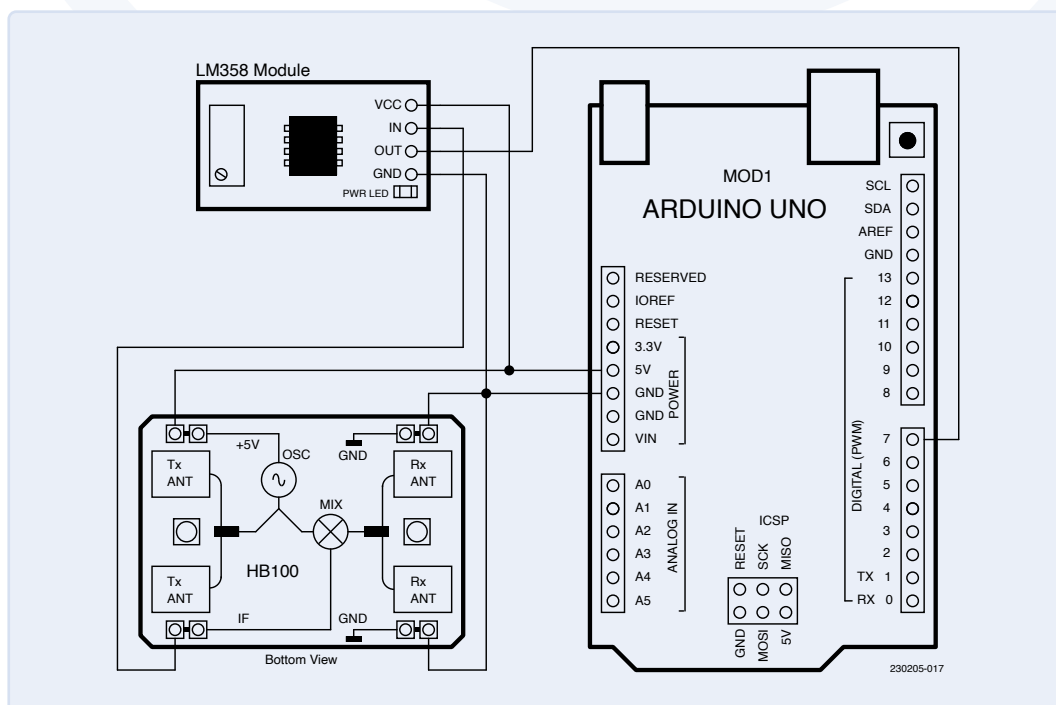
- ▶ **Phambili Newt 2.7" IoT Display (powered by ESP32-S2)**
www.elektor.nl/20230
- ▶ **SparkFun Environmental Combo Breakout - CCS811/BME280 (Qwiic)**
www.elektor.nl/19580

WEBLINKS

- [1] Günter Spanner, "MicroPython voor de ESP32 en consorten (deel 1)":
<http://www.elektormagazine.nl/articles/micropython-voor-de-esp32-en-consorten>
- [2] Nieuwste versie van de firmware voor de Newt:
https://phambili-pub.s3.amazonaws.com/Newt.ino_latest.bin
- [3] Inrichten van de Arduino IDE: https://github.com/Phambili-Tech/Newt_Display/wiki/Arduino-Setup

De doppler- bewegingssensor HB100

theorie en praktijk



Stefano Lovati (Italië)

Om de beweging van mensen, andere dieren of allerhande objecten te detecteren, zijn geschikte bewegingssensoren nodig. Dit artikel gaat in op een specifiek type sensor, dat bekend staat om zijn unieke mogelijkheden en educatieve betekenis: de microgolfsensor. Deze bewegingsdetector werkt volgens het dopplereffect, hetzelfde principe als in moderne radarsystemen wordt gebruikt.

In tegenstelling tot infrarood-bewegingsdetectoren, detecteren microgolfsensoren de beweging van een voorwerp door het analyseren van microgolven die door het voorwerp worden gereflecteerd. Vergeleken met andere soorten sensoren blijven de soorten objecten die kunnen worden gedetecteerd niet beperkt tot levende wezens. Bovendien wordt de microgolfsensor niet beïnvloed door de omgevingstemperatuur, heeft hij een groot meetbereik en een dito gevoeligheid. Vanwege zijn eigenschappen wordt hij veel gebruikt in industriële apparatuur, transport, automatische deuren, parkeersensoren en als snelheidsmeter. Omdat hij verschillende soorten objecten kan detecteren, wordt deze sensor in veel praktische toepassingen gecombineerd met een ander type sensor voor een gerichte en foutloze detectie. De microgolfsensor kan bijvoorbeeld worden gecombineerd met een infrarood-aanwezigheidssensor (PIR) om de doorgang of aanwezigheid van een persoon effectiever te bepalen en mogelijke storingsbronnen te elimineren.



Over het dopplereffect

Het dopplereffect heeft betrekking op de verandering in frequentie, of toonhoogte, die optreedt wanneer een geluidsbron de waarnemer nadert of zich van hem verwijderd, of omgekeerd, wanneer de waarnemer de geluidsbron nadert of zich ervan verwijderd. Ieder van ons heeft dit ongetwijfeld al eens bemerkt aan het geluid van een sirene (waarvan we gemakshalve aannemen dat die monotoon is en een vaste frequentie heeft) dat verandert wanneer de bron ons nadert of zich van ons verwijderd. Dit principe, beschreven door de Oostenrijkse natuurkundige Christian Doppler, geldt niet alleen voor geluidsgolven, maar ook voor radiogolven (elektromagnetische straling). Deze schijnbare verandering in frequentie tussen de bron van een golf en de ontvanger wordt exact bepaald door de relatieve beweging van bron en ontvanger ten opzichte van elkaar.

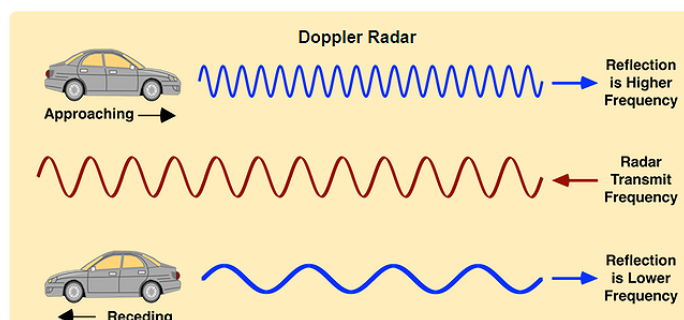
Om het dopplereffect beter te begrijpen, nemen we aan dat de frequentie en de golflengte van een geluid van een bron constant blijven. Wanneer zowel de bron als de ontvanger stilstaan, zal de ontvanger dezelfde geluidsfrequentie horen als door de bron wordt geproduceerd. Dit komt doordat de ontvanger hetzelfde aantal door de bron geproduceerde golven per seconde waarneemt. Als daarentegen de bron, de ontvanger of beide naar elkaar toe bewegen, zal de ontvanger een geluid met een hogere frequentie waarnemen. Dit komt doordat de ontvanger meer geluidsgolven per seconde waarneemt en dit interpreteert als geluid met een hogere frequentie. Als de bron en de ontvanger daarentegen van elkaar af bewegen, zal de ontvanger minder geluidsgolven per seconde waarnemen en een geluid met een lagere frequentie waarnemen.

Figuur 1 geeft een grafische voorstelling van het zojuist beschreven verschijnsel. Wanneer het voertuig de bron (de zender) nadert, neemt de frequentie van de gereflecteerde golf toe; wanneer het voertuig zich er echter van verwijderd, neemt de frequentie van de gereflecteerde golf af ten opzichte van de oorspronkelijke golf.

Een beetje rekenwerk

Voor een radarsysteem is de waarde van de frequentieverschuiving f_D in hertz, de zogenaamde dopplerfrequentie, te berekenen met de formule

$$f_D = \frac{2 \cdot v}{\lambda} \quad [1]$$



Figuur 1. Voorbeeld van de toepassing van het dopplereffect (bron: physicsopenlab.org, CC BY 4.0).

waarin v de snelheid van het gedetecteerde object is in m/s en λ de golflengte in meters. Deze formule is geldig wanneer de snelheid v uitsluitend in de lengterichting ligt en dus geen dwarscomponenten heeft, zoals het geval is bij zijwaartse beweging van het te detecteren object (wat het signaal reflecteert) en de bron (de zender). In het meest algemene geval is de bovenstaande vergelijking als volgt opgebouwd

$$f_D = \frac{2 \cdot v}{\lambda} \cdot \cos \alpha \quad [2]$$

waarbij α de hoek is tussen de richting van het uitgezonden/gereflecteerde signaal en de bewegingsrichting van het te detecteren voorwerp. Opmerking: wanneer het te detecteren object zich alleen haaks op de richting van het uitgezonden signaal beweegt, dan is er geen dopplereffect ($f_D = 0$). Dit verklaart waarom in *Top Gun*-achtige luchtgevechten, waar geavanceerde radars worden gebruikt die juist van het dopplereffect gebruik maken, piloten in netelige situaties altijd proberen haaks op de richting van de tegenstander te vliegen, wat de radardetectie en het volgen van positie en snelheid bemoeilijkt.

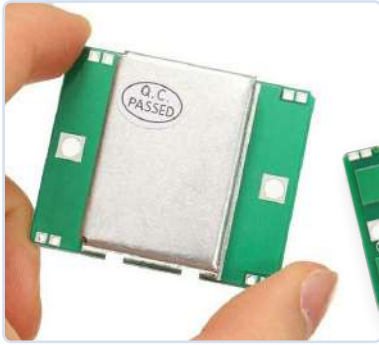
In het specifieke geval van een radar die elektromagnetische golven uitzendt (zoals in de voor dit artikel gebruikte sensor), luidt de formule die de modulus (absolute waarde) van de dopplerfrequentie f_D geeft, als volgt:

$$|f_D| = \frac{2 \cdot v_r}{\lambda} = \frac{2 \cdot v_r \cdot f_{TX}}{c_0} \quad [3]$$

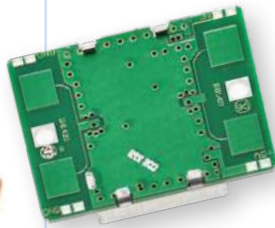
waarbij v_r de radiale snelheid is, f_{TX} de frequentie van het uitgezonden signaal en c_0 de lichtsnelheid.

De HB100-module

In dit artikel bekijken wij de eigenschappen van een van de goedkoopste aanwezigheidssensoren gebaseerd op het dopplereffect: de HB100. Het is in feite een geïntegreerde module met een X-band microgolvenzender, een ontvangerschakeling, een mixer en het hele RF-gedeelte, gemonteerd op een uiterst compacte print. Ter geruststelling van de lezer merk ik op dat, hoewel de betreffende frequenties vrij hoog zijn (ongeveer 10 GHz), er geen gezondheidsrisico's bestaan omdat het zendvermogen zeer gering is. Het werkingsprincipe van deze module, die voor een paar euro gemakkelijk verkrijgbaar is bij grote distributeurs van elektronische componenten, is niet alleen zeer interessant maar ook belangrijk vanuit een educatief oogpunt. De door de module gebruikte superheterodyne schakeling vormt namelijk de basis van moderne radarsystemen; het betreft een klassieke opzet die in veel RF-ontvangers wordt aangetroffen. **Figuur 2** toont de buitenkant van de uiterst compacte module, met een metalen behuizing die dient als afscherming van het RF-gedeelte. **Figuur 3** toont ter vergelijking de achterkant van de module, die in feite als zendantenne dienst doet. De HB100-bewegingssensor in miniatuurformaat is een dopplerezend/ontvangermodule die werkt in de X-band op een frequentie van 10,525 GHz. Binnenin bevindt zich een oscillator op basis van een diëlektrische resonator (DRO) en een paar antennes, die rechtstreeks op



Figuur 2. Bovenaanzicht van de HB100-module (bron: [6]).



Figuur 3. Onderaanzicht van de HB100-module (bron: [7]).

de print zijn geëtst. Deze module is een ideale oplossing om het aantal valse alarmen in inbraakdetectiesystemen te verminderen, vooral in combinatie met een conventionele passief infrarood-sensor (PIR). De sensor kan ook worden gebruikt in automatische deuropeningssystemen en snelheidsmeters voor voertuigen. De belangrijkste voordelen van de HB100-module zijn:

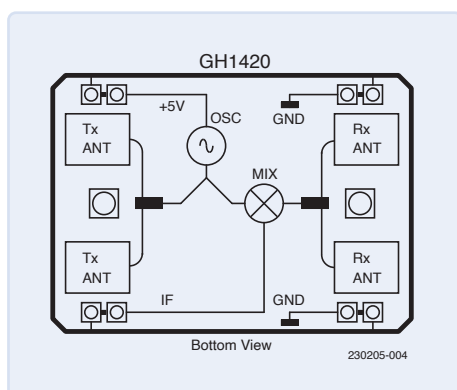
- > contactloze bewegingsdetectie;
- > de meting wordt niet beïnvloed door temperatuur, luchtvochtigheid, geluid, lucht, stof; de sensor is ook geschikt voor bijzonder ruwe omgevingsomstandigheden;
- > uitstekende immuniteit voor radio-interferentie;
- > gering uitgangsvermogen: de module vormt geen gevaar voor mensen en voldoet aan de voorschriften van de Federal Communication Commissions (FCC);
- > grote detectieafstand (tot 20 meter);
- > mogelijkheid om de beweging van verschillende objecten te detecteren, niet alleen van personen;
- > sterk gebundelde radiogolven;
- > gering stroomverbruik;
- > kan zowel in CW-modus (Continuous Wave, dat wil zeggen continue transmissie van het radiosignaal) als puls-modus (transmissie van korte opeenvolgende pulsen) werken;
- > compacte afmetingen en zeer dun.

In **figuur 4** zien we het blokschema van de module, wat we nu in detail zullen doornemen. Laten we beginnen met de al genoemde oscillator, een DRO die is afgestemd op de frequentie van 10,525 GHz. Een keramische schijf, gewoonlijk gemaakt van bariumtitaanaat ($\text{Ba}_2\text{Ti}_9\text{O}_{20}$) fungeert als resonantieholte voor de RF-energie. De resonantiefrequentie is

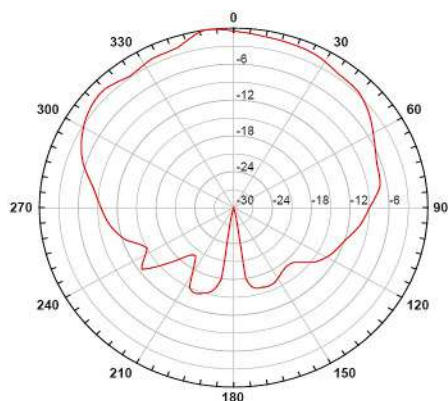
afhankelijk van de grootte en de vorm van het materiaal en de resonantieholte. Dit is precies de frequentie van het signaal dat door de module wordt uitgezonden via de twee rechthoekige antennes (geëtst op de print) die links in figuur 4 zichtbaar zijn. Elk obstakel in de richting van de uitgezonden golfbundel zal een gereflecteerde golf veroorzaken, waarvan de frequentie varieert afhankelijk van de beweging van dat voorwerp ten opzichte van de zendmodule. In ieder geval zal de gereflecteerde golf een frequentie hebben die enigszins afwijkt van de uitgezonden golf. De omgang met een RF-signaal bij deze frequenties is voor iedereen een moeilijke opdracht, niet alleen op hardwareniveau maar vooral ook op softwareniveau. De oplossing voor dit probleem wordt gevormd door de superheterodyne schakeling, om precies te zijn de RF-mixer die we in figuur 4 zien. De functie daarvan is het 'combineren' van het uitgezonden signaal met het signaal dat wordt ontvangen via de twee rechthoekige antennes rechts in figuur 4. Over het algemeen heeft een mixer twee ingangssignalen en produceert hij twee uitgangssignalen die respectievelijk de som en het verschil van de beide ingangssignalen zijn. In ons geval worden dus twee signalen gegenereerd: één met een frequentie in de orde van grootte van 20 GHz (som van het uitgezonden signaal en het ontvangen signaal) en één met een frequentie van enkele Hertz (verschil tussen het uitgezonden en het ontvangen signaal). Dit mechanisme biedt een elegante oplossing voor ons probleem, aangezien het probleem van het meten van een X-bandsignaal (tientallen GHz) wordt omgezet in het meten van een laagfrequent signaal (tot enkele honderden Hz, afhankelijk van de snelheid van het bewegende voorwerp), dat gemakkelijk kan worden verwerkt door een gewone en goedkope microcontroller zoals een Arduino.

Het IF-uitgangssignaal van de mixer is precies dat verschilsignaal. Dit proces, dat in veel RF-schakelingen voorkomt, wordt ook wel 'omlaagconversie' wordt genoemd: het doel ervan is in feite een hoge frequentie om te zetten naar een lagere frequentie om de verwerking ervan te vereenvoudigen. De aldus verkregen frequentie wordt de 'intermediate frequency' (IF) of ook wel middenfrequentie genoemd om deze te onderscheiden van de oorspronkelijke of 'basisband'-frequentie. We kijken nog even verder naar figuur 4 en zien niet meer dan 4 aansluitingen om de module op een microcontroller of meetschakeling aan te sluiten: 2 massa-aansluitingen, 1 pin voor de positieve 5V-voeding en 1 pin voor het laagfrequente IF-uitgangssignaal. In wezen hebben we hier een signaal met een frequentie van slechts enkele Hertz ter beschikking, waarbij de 'moeilijke' verwerking van het microgolfsignaal wordt overgelaten aan de hardware die in de HB100-module is ingebouwd. Er resteert nog het probleem van de vrij kleine amplitude van het signaal op de sensoruitgang; we zullen later zien hoe dit kan worden aangepakt.

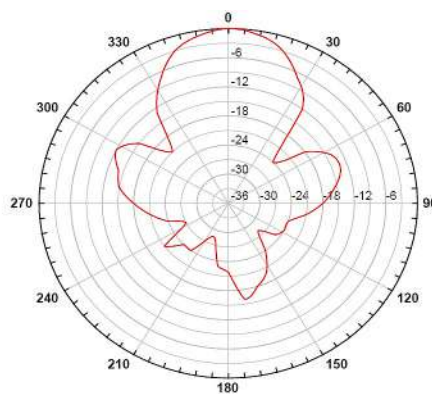
Voor praktisch gebruik moet de module worden gemonteerd met de antennes (zie figuur 3) gericht op het gebied dat men met het afgestraalde uitgangssignaal wil bestrijken, en wel zo dat de beste dekking wordt verkregen. De stralingsdiagrammen van de antennes (in het horizontale en verticale vlak) zijn getekend in **figuur 5**.



Figuur 4. Blokschema van de sensor (bron: [5]).



Azimuth



Elevation

*Figuur 5.
Stralingsdiagrammen
van de antenne
(bron: [1]).*

Uitgangssignaal

De IF-pin levert het uitgangssignaal van de module, dat correspondeert met de frequentieverschuiving die wordt veroorzaakt door de grootte van de beweging van een door de antenne aangestraald voorwerp. De amplitude van deze frequentieverschuiving ten gevolge van het dopplereffect is evenredig met de gereflecteerde energie en ligt in de orde van enkele microvolt (μV). Daarom wordt gewoonlijk een laagfrequent-versterker met hoge versterkingsfactor op de IF-pin aangesloten om het uitgangssignaal op te voeren. Meer precies is de frequentie van de dopplerverschuiving evenredig met de bewegings-snelheid. Om je een indruk te geven: een normaal lopend mens veroorzaakt een frequentieverschuiving van minder dan 100 Hz. De ontvangen signaalsterkte (RSS) komt overeen met de spanning op de IF-uitgang. De dopplerfrequentie kan worden berekend met behulp van de eerder gegeven vergelijkingen. Wanneer het door de sensor aangestraalde voorwerp alleen in de radiale richting beweegt, hetzij naderend of zich verwijderend, kan de vorige formule worden vereenvoudigd en krijgt deze de volgende vorm, waarbij v_r de radiale snelheid is, uitgedrukt in km/h:

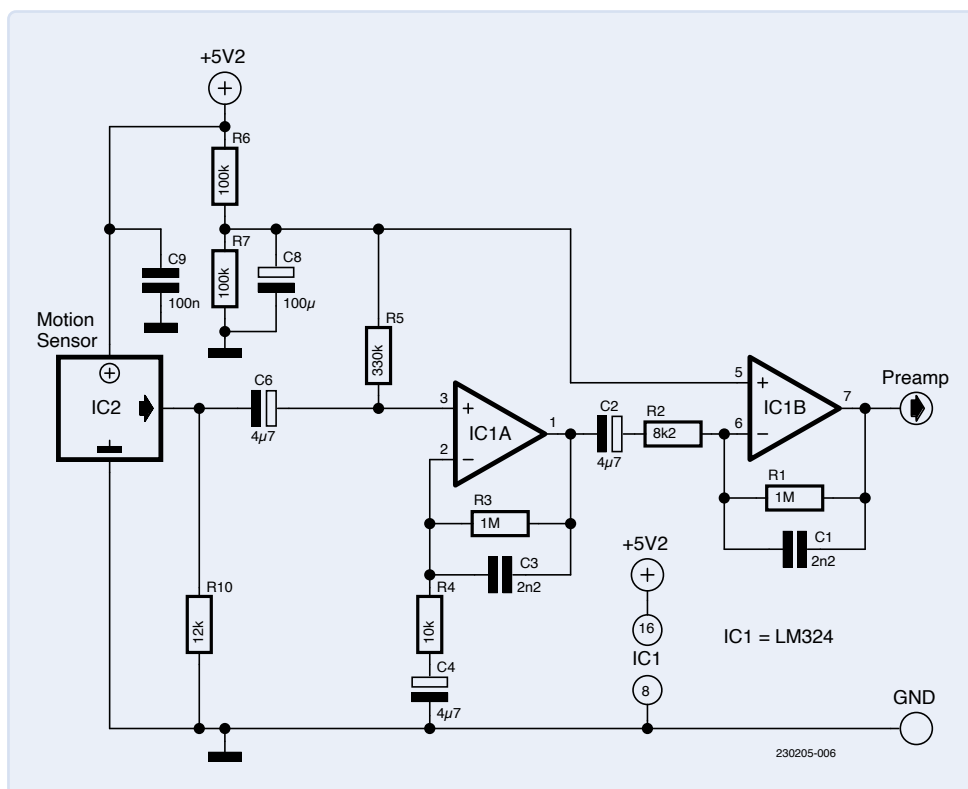
$$f_D = 19.49 \cdot v_r \quad [4]$$

De technische specificaties van de module kunnen als volgt worden samengevat:

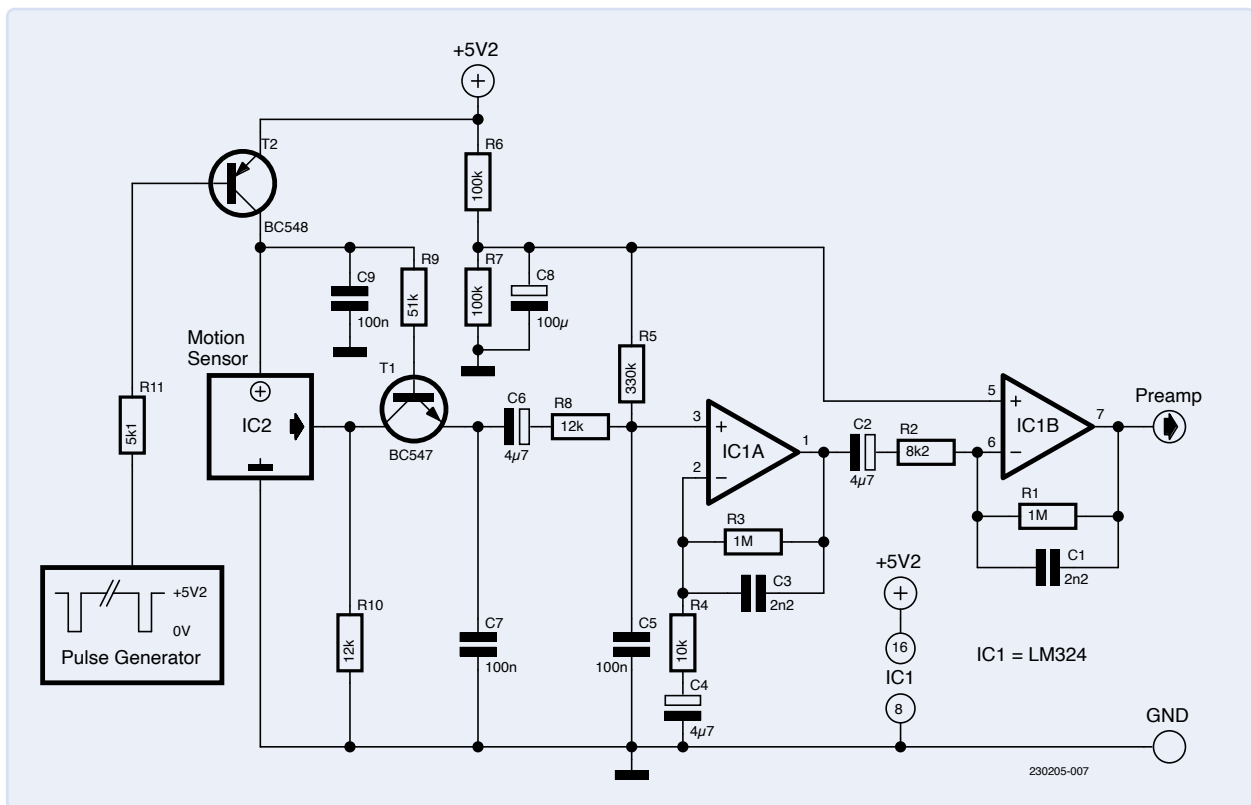
- > voedingsspanning: $5\text{ V} \pm 0,25\text{ V}$;
- > stroomopname (in CW-modus): 60 mA maximaal, 37 mA typisch;
- > afmetingen: $61,2 \times 61,2\text{ mm}$;
- > detectieafstand: tussen 2 m en 16 m;
- > zendfrequentie: 10,525 GHz;
- > frequentie nauwkeurigheid: 3 MHz;
- > minimaal uitgangsvermogen: 13 dBm EIRP;
- > harmonische emissie: $< -10\text{ dBm}$;
- > gemiddelde stroom: 2 mA typisch;
- > minimale pulsbreedte (zenden): 5 μs ;
- > minimale duty cycle: 1%.

Signaalconditionering

De uitgang van de HB100-sensor, beschikbaar op de IF-pin, is een laagfrequent signaal dat evenredig is met de frequentieverschuiving die door het ontvangen signaal wordt veroorzaakt. Het probleem is dat het signaal op deze uitgang niet versterkt is en een zeer kleine amplitude (enkele μV) heeft; daarom is in praktische toepassingen een forse versterking nodig, bij voorkeur in combinatie met een laagdoorlaat-filter om alle ongewenste frequenties boven een paar honderd hertz te elimineren. Zowel de datasheet [1] als de application note [2] van de sensor zijn in dit opzicht zeer nuttig. Met name die laatste toont twee mogelijke schakelingen voor de signaalconditionering. De eerste, te zien in **figuur 6**, is van toepassing op de CW-modus en is gebaseerd



Figuur 6. Signaalconditionering voor de CW-modus (bron: [5]).



Figuur 7. Signaalconditionering voor de puls-modus (bron: [5]).

op de operationele versterker LM324 van Texas Instruments, die een forse versterking biedt.

De tweede schakeling, te zien in **figuur 7**, wordt gebruikt in de puls-modus, waarbij de zender een reeks pulsen uitzendt met een bepaalde frequentie (bekend als PRF, kort voor Pulse Repetition Frequency) en met een bekende duty cycle. De fabrikant van de HB100-module adviseert $PRF = 2 \text{ kHz}$ en $duty \text{ cycle} = 4\%$.

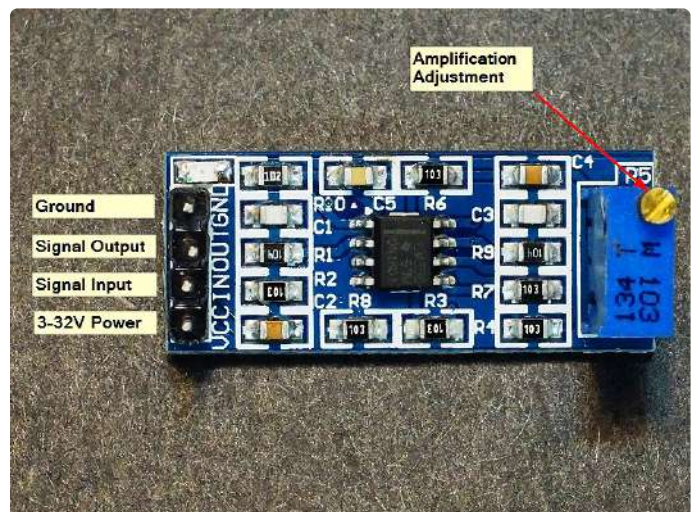
Deze schakeling heeft, naast de operationele versterker LM324 met hoge versterking, twee transistoren. Zoals te zien in de linkerbenedenhoek van **figuur 7**, kan de pulsmodus worden toegepast door middel van een pulsvormige voedingsspanning voor de module, en wel zodanig dat de door de fabrikant genoemde aanbevolen PRF en duty cycle worden aangehouden. Met andere woorden, in de pulsmodus schakelt de extra serie-schakeltransistor elke 0,5 ms (2 kHz) de module in met een duty cycle van 4% (4% van de tijd AAN).

Om de tests met de sensor te vereenvoudigen, is echter gekozen voor een universele opamp die al op een print is gemonteerd, compleet met pinheader en potmeter voor het instellen van de versterking. De module, die zeer goedkoop en gemakkelijk verkrijgbaar is, is gebaseerd op de LM358 en is afgebeeld in **figuur 8**. De LM358 zorgt voor een eenkanaals-versterking met een 1x en 100x instelbare versterking: door de potmeter op het printje rechtsom te draaien wordt de versterking verkleind en wanneer deze linksom wordt gedraaid wordt de versterking groter. Verder bevat de module een LED die aangeeft dat de voedingsspanning aanwezig is. De LM358 opamp kan signalen verwerken met frequenties tot 700 kHz (dus ruim boven het bereik waarin wij geïnteresseerd zijn) en kan gevoed worden met spanningen tussen 3 V en 32 V. **Figuur 8** toont de aansluitingen van de module: V_{CC} , GND, ingang en output.

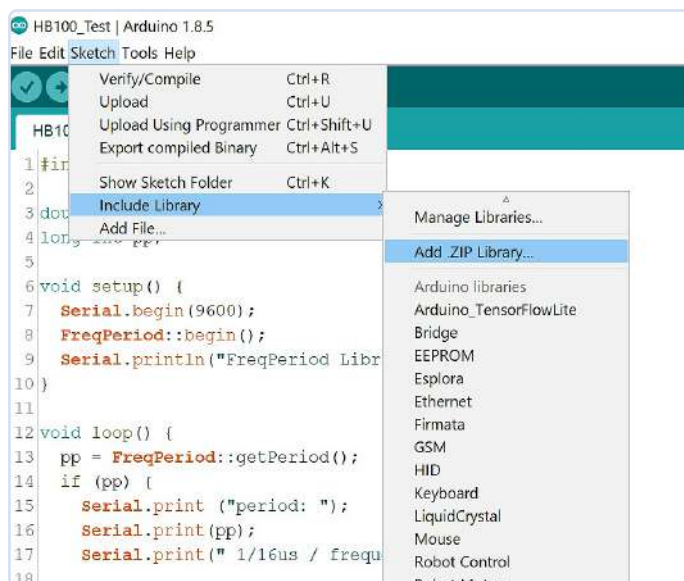
Test met de Arduino

Om de HB100-dopplersensor te testen, kunnen we gebruik maken van een gewoon Arduino UNO-board en de LM358-versterker-

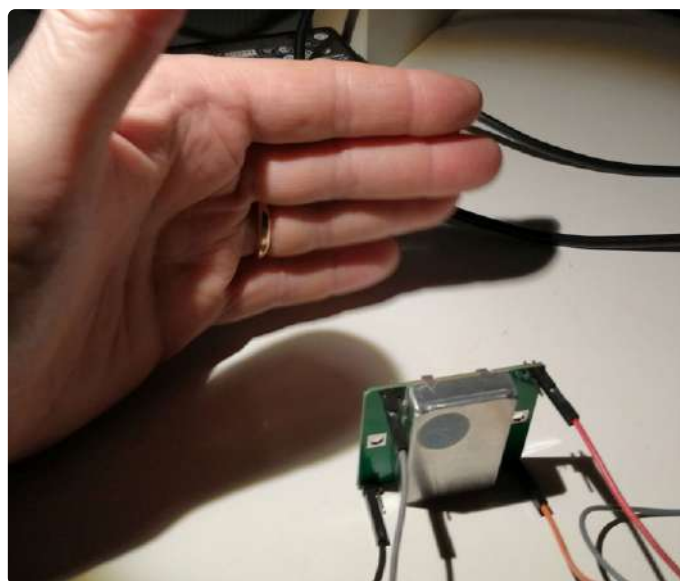
module – of als alternatief, een andere eenkanaal-versterker met hoge versterking, weinig ruis en met een enkele voedingsspanning. Op het Arduino UNO-board draait een speciale sketch die de frequentie van het signaal van de module kan detecteren, en daaruit eventueel de snelheid kan afleiden waarmee het potentiële 'doelwit' beweegt. Deze sketch maakt gebruik van een bibliotheek die speciaal is ontwikkeld voor nauwkeurige frequentiemetingen. Voor het meten van frequenties in de audioband of lager moet namelijk de periodetijd van hetingangssignaal zeer nauwkeurig worden bepaald. Hiervoor gebruikt de bibliotheek een uiterst nauwkeurige 'Counter and Capture'-module die beschikbaar is in de hardware-architectuur van de ATmega-microcontroller. De bibliotheek laat de tijdsduur zien, uitgedrukt als een



Figuur 8. De LM358-versterkermodule.



Figuur 9. Installatie van de FreqPeriod-bibliotheek.



Figuur 10. Snelle handbewegingen voor de sensorantenne.

geheel getal met een resolutie van $1/16 \mu\text{s}$. Om de frequentie af te leiden deel je eenvoudigweg de klokfrequentie door de waarde die de bibliotheek retourneert. In ons geval wordt de klokfrequentie ingesteld op 16.000.400 om eventuele onnauwkeurigheden van het Arduino-board te compenseren. De bibliotheek is momenteel niet opgenomen in de officiële Arduino-bibliotheek, maar kan gemakkelijk worden geïnstalleerd in de IDE na het downloaden van het .zip-bestand dat beschikbaar is op [3]. Open hiertoe de Arduino IDE, selecteer het menuonderdeel *Sketch -> Include Library -> Add .ZIP Library...*, zoals getoond in **figuur 9**. Selecteer vervolgens het .zip-bestand van de bibliotheek die je hebt gedownload. Sluit de IDE af en herstart die weer om de nieuwe bibliotheek beschikbaar te maken in de Arduino-omgeving. De sketch is te zien in **listing 1**. Hij is zeer compact en relatief eenvoudig. In de `setup()`-functie worden de seriële verbinding (9600 bps) en de *FreqPeriod*-bibliotheek geïnitieerd via de `begin()`-methode. In de `loop()`-functie wordt eerst herhaaldelijk de tijdsduur van het ingangssignaal gemeten via het aanroepen van de `getPeriod()`-methode. Vervolgens worden de bijbehorende waarden van de frequentie (in hertz) en de radiale snelheid (in km/h) berekend en weergegeven. Om de werking van de sketch te controleren moeten we de HB100-sensor en de LM358-versterkermodule aansluiten op het Arduino-board. De aansluitingen zijn te zien op de kopillustratie van dit artikel. Vervolgens schakelen we het geheel in en activeren de seriële monitor van Arduino IDE om de geproduceerde logs te bekijken. Door een voorwerp, bijvoorbeeld je hand (zie **figuur 10**), voor de HB100-sensorantenne te bewegen (die, zoals bekend, aan de achterkant van de print zit, aan de andere kant van de metalen behuizing), zouden we een verandering moeten zien in de snelheid die de sketch berekent. Een voorbeeld van de uitvoer via de seriële interface is te zien in **figuur 11**.

Op basis van deze snelheidswaarden kunnen we dan zelfs zeer complexe algoritmen construeren met toepassingen als deurbediening



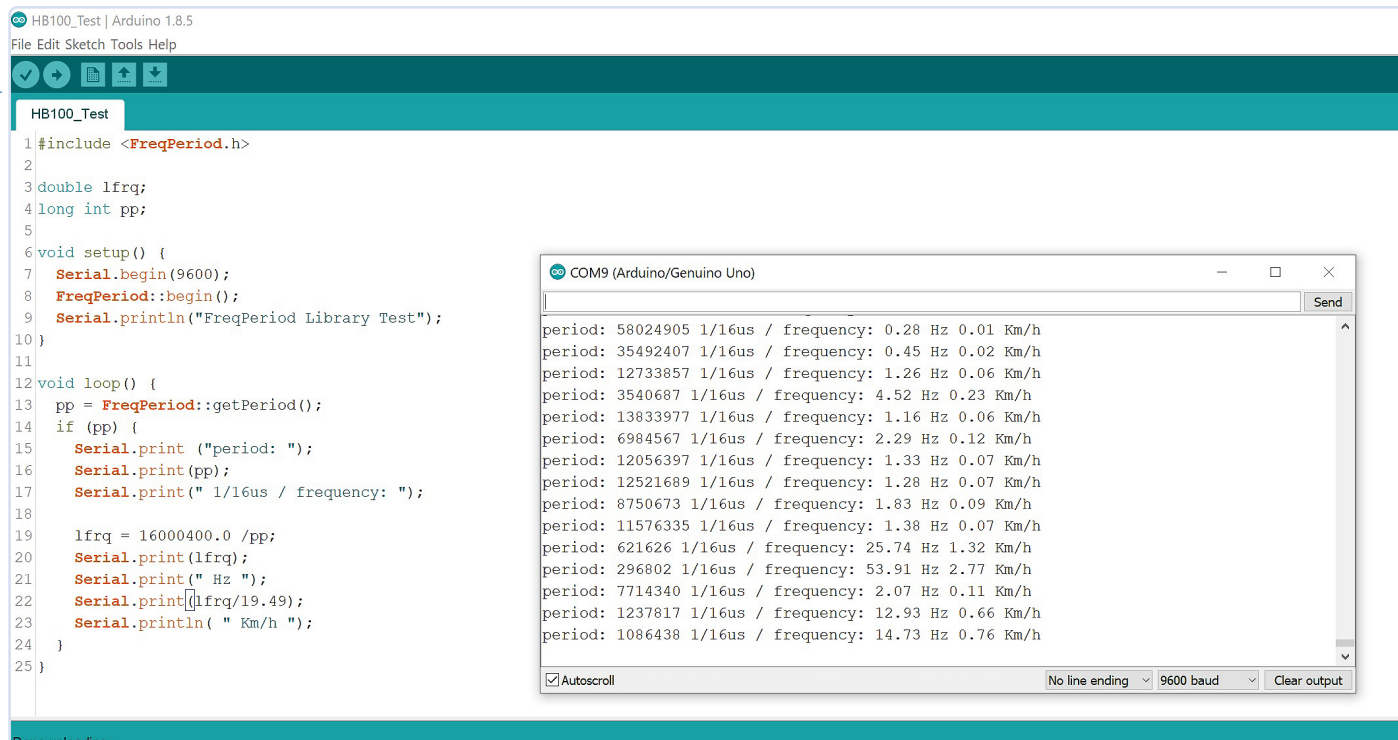
Listing 1. Arduino-sketch.

```
#include <FreqPeriod.h>
double lfrq;
long int pp;

void setup() {
    Serial.begin(9600);
    FreqPeriod::begin();
    Serial.println("FreqPeriod Library Test");
}

void loop() {
    pp = FreqPeriod::getPeriod();

    if (pp) {
        Serial.print("period: ");
        Serial.print(pp);
        Serial.print(" 1/16us / frequency: ");
        lfrq = 16000400.0 / pp;
        Serial.print(lfrq);
        Serial.print(" Hz ");
        Serial.print(lfrq/19.49);
        Serial.println(" km/h ");
    }
}
```

Figuur 11. Weergave in de seriële monitor.

(inclusief schuifdeuren), automatische lichtsakelaars in trappenhuizen, het openen van toegangspoorten voor voetgangers, inbraakdetectiesystemen, videobewaking en meer. De gevoeligheid van de module zelf is goed genoeg, mits hij binnen zijn maximale bereik van ongeveer 20 meter wordt gebruikt. De prestaties kunnen worden verbeterd door een versterkerschakeling te gebruiken met een hogere versterking dan die van de LM358 (100x).

Conclusie

Het doel van dit artikel was de introductie van een erg interessante en niet erg bekende HB100 doppler-bewegingssensor. De HB100 is goedkoop en gemakkelijk aan te sluiten op een microcontroller en biedt de mogelijkheid om te experimenteren met belangrijke concepten en technieken voor signaaldetectie die ten grondslag liggen aan geavanceerde en dure commerciële radars. Aan toepassingen voor de sensor is geen gebrek, ongetwijfeld denken veel makers al na over verdere, toekomstige projecten die ermee gerealiseerd kunnen worden. 

vertaling: Hans Adams — 230205-03

Vragen of opmerkingen?

Hebt u technische vragen naar aanleiding van dit artikel? Stuur een e-mail naar de redactie van Elektor via redactie@elektor.com.

Over de auteur

Na zijn studie Elektrotechniek aan de Politecnico di Milano begon Stefano zijn carrière als firmware- en softwareontwikkelaar. In de loop der jaren heeft hij de transformatie van de embedded wereld meegemaakt, van de eerste 8bit-microprocessoren – alleen programmeerbaar in assembler – tot de meest recente SoC, FPGA, DSP en programmeerbare logica met uitstekende prestaties en functies. Hij heeft een brede belangstelling voor technologie en elektronica en wijdt een deel van zijn vrije tijd aan het bestuderen van nieuwe componenten en het realiseren van kleine projecten.



Gerelateerde producten

- > **YDLIDAR X2 Lidar – 360-degree Laser Range Scanner (8 m)**
www.elektor.nl/18941
- > **Arduino Uno Rev3**
www.elektor.nl/15877
- > **Arduino Uno Mini (Limited Edition)**
www.elektor.nl/20098

WEBLINKS

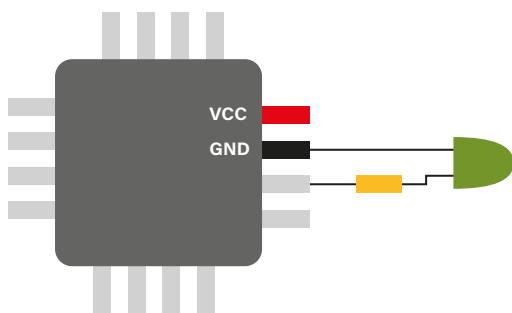
- [1] Datasheet HB100-module: https://limpkin.fr/public/HB100/HB100_Microwave_Sensor_Module_Datasheet.pdf
- [2] Application Note HB100: https://limpkin.fr/public/HB100/HB100_Microwave_Sensor_Application_Note.pdf
- [3] Bibliotheek frequentiemeting: <https://github.com/Jorge-Mendes/Agro-Shield/tree/master/OtherRequiredLibraries/FreqPeriod>
- [4] Software op de webpagina bij dit project: <https://elektormagazine.com/230205-01>
- [5] Schema's: https://mantech.co.za/Datasheets/Products/MSAN-001_AGILSENSE.pdf
- [6] Foto bovenzijde HB100: https://mantech.co.za/Datasheets/Products/HB100_RADAR.pdf
- [7] Foto onderzijde HB100: <https://kuongshun-ks.com/uno/uno-sensor/hb100-microwave-doppler-radar-wireless-module-moti.html>

Een gids voor puristisch programmeren (deel 1)

voor STM32 en andere controllers

Sergey Lyubka (Ierland)

Wil je microcontrollers programmeren op een no-nonsense-niveau, waarbij je bovendien inzicht krijgt in hoe ze eigenlijk werken? Deze gids, geschreven voor ontwikkelaars, helpt je op weg met niet meer dan de GCC-compiler en een referentiehandleiding. De principes die je hier leert, helpen je beter te begrijpen hoe frameworks als Cube, Keil en Arduino functioneren. In deze tweedelige gids gebruiken we de STM32F429-controller op een Nucleo-F429ZI-board, maar wat je hier leert kan gemakkelijk worden aangepast aan andere microcontrollers.



Figuur 1. Firmwarecode kan een hoge of lage spanning op een signaalpin zetten, waardoor een LED gaat knipperen.

Een microcontroller (μ C, of MCU) is een kleine computer. Doorgaans heeft hij een CPU, RAM, flash-geheugen voor de firmware, en een heleboel aansluitpinnen. Sommige daarvan worden gebruikt om de MCU te voeden (meestal aangeduid met GND (ground) en VCC). Andere pinen worden gebruikt om met de MCU te communiceren door middel van hoge of lage spanningen die op die pinen worden gezet. Een van de eenvoudigste communicatiemiddelen is een LED die op een pin is aangesloten: één pootje van de LED is verbonden met de massapin (GND) en de andere via een stroombegrenzende weerstand met een signaalpin. Firmware kan een hoge of lage spanning op die signaalpin zetten, waardoor de LED aan of uit gaat (**figuur 1**).

Benodigde hardware en tools

In deze handleiding gebruiken we een Nucleo-F429ZI-ontwikkelboard (verkrijgbaar bij Mouser en andere leveranciers). Om deze handleiding te volgen, moet je de STM32F429 MCU referentiehandleiding [1] downloaden en vervolgens de handleiding voor het ontwikkelboard [2].

Daarna heb je de volgende tools nodig:

ARM GCC, <https://launchpad.net/gcc-arm-embedded> – voor compileren en linken

GNU make, <https://gnu.org/software/make> – voor build-automatisering

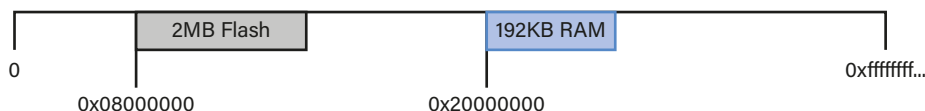
ST link, <https://github.com/stlink-org/stlink> – voor het flashen

Dit zijn de installatie-instructies voor Linux (Ubuntu):

Start een terminal en voer dit uit:

```
$ sudo apt -y update
$ sudo apt -y install gcc-arm-none-eabi make
stlink-tools
```

Zie [3] om de tools op een Mac of Windows PC te installeren.



Figuur 2. Locatie van STM32F429 flash- en RAM-bereiken.

Geheugen en registers

De adresruimte van een 32bit-MCU, bijvoorbeeld de STM32F429 van STMicroelectronics, is in bereiken verdeeld. Een geheugenbereik (dat op een specifiek adres begint) is bijvoorbeeld toegewezen aan het interne flashgeheugen van de MCU. Instructies in de firmware worden gelezen en uitgevoerd door uit dat bereik te lezen. Een ander bereik is RAM, dat ook op een specifiek adres begint. In het RAM-geheugen kunnen willekeurige waarden worden gelezen en geschreven.

In de STM32F429-referentiehandleiding [1] lezen we in paragraaf 2.3.1 dat het RAM-bereik begint op adres 0x20000000 en een grootte heeft van 192 KB. In paragraaf 2.4 zien we dat het flash-geheugen vanaf adres 0x08000000 is gedefinieerd. Onze MCU heeft 2 MB flash, dus de flash- en RAM-bereiken kunnen zoals in **figuur 2** in kaart worden gebracht.

De handleiding weet ons ook te vertellen dat er nog veel meer geheugenbereiken zijn. De corresponderende adresbereiken zijn gegeven in paragraaf 2.3, 'Memory map'. Er is bijvoorbeeld een 'GPIOA'-bereik dat begint bij 0x40020000 en een lengte heeft van 1 KB.

Deze geheugenbereiken komen overeen met verschillende 'periferie' in de MCU – een plukje siliciumhardware dat ervoor zorgt dat bepaalde pinnen zich op een specifieke manier gedragen. Een perifeer geheugenbereik is een verzameling 32bit-registers. Elk register is een bereik van 4 bytes op een bepaald adres in het geheugen, dat overeenkomt met een bepaalde functie van

het betreffende randapparaat. Door waarden in een register te schrijven – met andere woorden, door een 32bit-waarde naar een bepaald geheugenadres te schrijven – kunnen we bepalen hoe een bepaald randapparaat zich moet gedragen. Door deze registers uit te lezen, kunnen we de gegevens of configuratie van een randapparaat ophalen/teruglezen.

Er bestaat veel verschillende periferie. Een van de eenvoudigste is GPIO (*General Purpose Input Output*), waarmee de gebruiker MCU-pinnen in 'uitgangsmodus' kan zetten ze een hoge of lage spanning kan laten voeren. Of pinnen in 'ingangsmodus' zetten en er spanningswaarden mee inlezen. Er is een UART-randapparaat dat seriële gegevens kan verzenden en ontvangen via twee pinnen met behulp van een serieel protocol. En er bestaat nog veel meer randapparatuur.

Vaak zijn er meerdere 'instanties' van hetzelfde randapparaat, bijvoorbeeld *GPIOA*, *GPIOB* enzovoort, die verschillende groepen MCU-pinnen aansturen. Evenzo kunnen er *UART1*, *UART2* enzovoort zijn, die de implementatie van meerdere UART-kanalen mogelijk maken. Op de STM32F429 zijn er diverse GPIO- en UART-randapparaten.

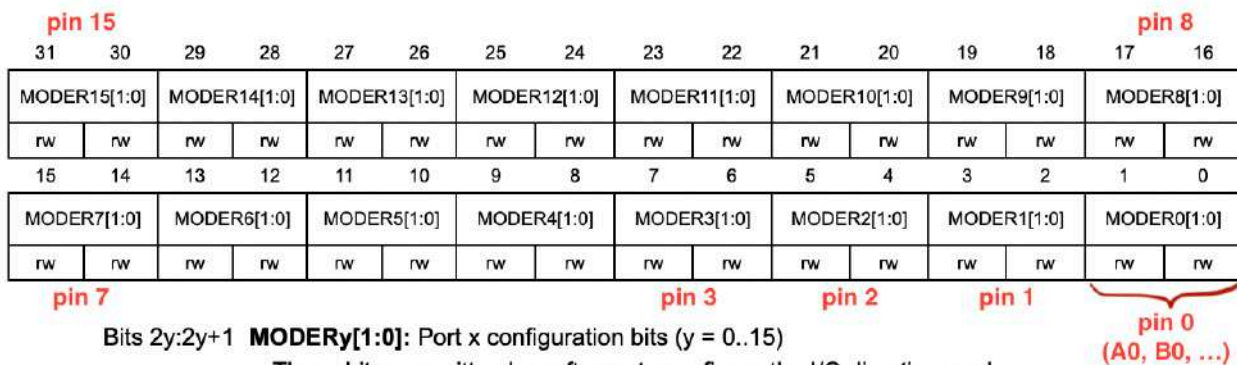
De GPIOA-periferie begint bijvoorbeeld op 0x40020000, en we kunnen de beschrijving van het GPIO-register vinden in paragraaf 8.4 [1]. De referentiehandleiding zegt dat het *GPIOA_MODER*-register een offset 0 heeft, wat betekent dat het adres $0x40020000 + 0$ is. In **figuur 3** zien we het formaat van het register.

8.4.1 GPIO port mode register (GPIOx_MODER) (x = A..I/J/K)

Address offset: 0x00

Reset values:

- 0xA800 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports



Figuur 3. De beschrijving van de GPIO-registers staat in de referentiehandleiding. Eén MODER-register stuurt 16 fysieke pinnen aan (bron: [1]).

De handleiding laat zien dat het 32-bit *MODER*-register een verzameling van 2bit-waarden is; 16 in totaal. Daarom bestuurt één *MODER*-register 16 fysieke pinnen: bits 0...1 besturen pin 0, bits 2...3 besturen pin 1 enzovoort. De 2bit-waarde codeert de pinmodus: 0 betekent invoer, 1 betekent uitvoer, 2 betekent 'alternatieve functie' – een specifiek gedrag dat elders wordt beschreven – en 3 betekent analoog. Aangezien de naam *GPIOA* is, worden de pinnen 'A0', 'A1' enzovoort genoemd. Voor het randapparaat *GPIOB* zouden de pinnen 'B0', 'B1' ... heten.

Als we de 32bit-waarde 0 in het *MODER*-register schrijven, zetten we alle 16 pinnen, van A0 tot en met A15, in de ingangsmodus:

```
* (volatile uint32_t *) (0x40020000 + 0) = 0;
// Set A0...A15 to input mode
```

Let op de 'qualifier' *volatile*. De betekenis daarvan komt later aan de orde. Door afzonderlijke bits in te stellen, kunnen we selectief specifieke pinnen in een gewenste modus zetten. Dit codefragment bijvoorbeeld zet pin A3 in de uitgangsmodus:

```
* (volatile uint32_t *) (0x40020000 + 0) &= ~(3 << 6);
// Clear bit range 6...7
* (volatile uint32_t *) (0x40020000 + 0) |= 1 << 6;
// Set bit range 6...7 to 1
```

Laat me die bitbewerkingen uitleggen. We willen bits 6...7, die verantwoordelijk zijn voor pin 3 van de *GPIOA*-periferie, een bepaalde waarde geven (1 in dit geval). Dat gebeurt in twee stappen. Eerst moeten we de huidige waarde van bits 6...7 kennen, omdat ze mogelijk al een waarde bevatten. Daarna moeten we de relevante bits van 6...7 instellen om de waarde te krijgen die we willen.

Eerst moeten we dus bitbereik 6...7 (twee bits op positie 6) op 0 zetten. Hoe zetten we een aantal bits op 0? De vier stappen zijn in **tabel 1** opgesomd.

Merk op dat de laatste bewerking, een logisch AND-opeatie, N bits op positie X op 0 zet (omdat ze met een 0 worden ge-AND), maar de waarde van alle andere bits behoudt (omdat die met een 1 worden ge-AND). Het behouden van de bestaande waarde is belangrijk,

Tabel 1. Specifieke bits 0 maken.

Actie	Uitdrukking	Bits (eerste 12 van 32)
Een getal met N aaneengesloten bits instellen: $2^N - 1$, hier $N=2$	3	000000000011
Verschuif dat getal X posities naar links	$(3 << 6)$	000011000000
Inverteer het getal: nullen worden enen, en enen worden nullen	$\sim(3 << 6)$	111100111111
Logische AND met bestaande waarde	$VAL \&= \sim(3 << 6)$	xxxx00xxxxxx

omdat we de instellingen in andere bitbereiken niet willen veranderen. Dus kunnen we in het algemeen, als we N bits op positie X willen wissen, het volgende schrijven:

```
REGISTER &= ~((2^N - 1) << X);
```

En tenslotte willen we een bepaald bitbereik instellen op de waarde die we willen. We verschuiven die waarde X posities naar links, en OR-ren met de huidige waarde van het hele register (om de waarden van andere bits te behouden):

```
REGISTER |= VALUE << X;
```

Periferie-programmering die voor mensen leesbaar is

In de vorige paragraaf hebben we gezien dat we een perifeer register kunnen lezen en schrijven door rechtstreeks bepaalde geheugenadressen aan te spreken. Laten we eens kijken naar het codefragment dat pin A3 in uitgangsmodus zet:

```
* (volatile uint32_t *) (0x40020000 + 0) &= ~(3 << 6);
// Clear bit range 6...7
* (volatile uint32_t *) (0x40020000 + 0) |= 1 << 6;
// Set bit range 6...7 to 1
```

Dat is tamelijk cryptisch. Zonder uitgebreid commentaar is zulke code moeilijk te begrijpen. We kunnen deze code echter herschrijven in een veel leesbaarder vorm. Het idee is om het hele randapparaat voor te stellen als een structuur met 32bit-velden. Laten we eens kijken welke registers er bestaan voor de *GPIO*-periferie in paragraaf 8.4 van de referentiehandleiding. Het zijn *MODER*, *OTYPER*, *OSPEEDR*, *PUPDR*, *IDR*, *ODR*, *BSRR*, *LCKR* en *AFR*. Hun offsets zijn 0, 4, 8 enzovoort. Dat betekent dat we ze kunnen voorstellen als een structuur met 32bits-velden, en een *#define* kunnen opstellen voor *GPIOA*:

```
struct gpio {
    volatile uint32_t MODER, OTYPER, OSPEEDR, PUPDR,
    IDR, ODR, BSRR, LCKR, AFR[2];
};
#define GPIOA ((struct gpio *) 0x40020000)
```

Dan kunnen we voor het instellen van de *GPIO*-pinmodus een functie definiëren:

```
// Enum values are per reference manual: 0, 1, 2, 3
enum ;

static inline void gpio_set_mode (struct gpio *gpio,
uint8_t pin, uint8_t mode) {
    gpio->MODER &= ~(3U << (pin * 2)); // Clear existing
setting
    gpio->MODER |= (mode & 3) << (pin * 2); // Set new
mode
}
```

Nu kunnen we het codefragment voor A3 als volgt herschrijven:

```
gpio_set_mode(GPIOA, 3 /* pin */, GPIO_MODE_OUTPUT);  
// Set A3 to output
```

Onze MCU heeft diverse GPIO-periferie (ook wel ‘banken’ genoemd): A, B, C...K. Uit paragraaf 2.3 zien we dat ze 1 KB uit elkaar liggen: GPIOA is te vinden op adres 0x40020000, GPIOB op 0x40020400, enzovoort:

```
#define GPIO(bank) ((struct gpio *) (0x40020000 + 0x400  
* (bank)))
```

We kunnen een pinnummering definiëren die de bank en het pinnummer omvat. Daarvoor gebruiken we een 2-byte `uint16_t` waarde, waarbij de bovenste byte de GPIO-bank aangeeft en de onderste byte het pinnummer:

```
#define PIN(bank, num) (((bank) - 'A') << 8) | (num))  
#define PINNO(pin) (pin & 255)  
#define PINBANK(pin) (pin >> 8)
```

Zo kunnen we pinnen specificeren voor elke GPIO-bank:

```
uint16_t pin1 = PIN('A', 3); // A3 - GPIOA pin 3  
uint16_t pin2 = PIN('G', 11); // G11 - GPIOG pin 11
```

Laten we de `gpio_set_mode()`-functie herschrijven op basis van onze pinspecificatie:

```
static inline void gpio_set_mode(uint16_t pin, uint8_t  
mode) {  
    struct gpio *gpio = GPIO(PINBANK(pin)); // GPIO  
bank  
    uint8_t n = PINNO(pin); // Pin number  
    gpio->MODER &= ~(3U << (n * 2)); // Clear existing  
setting  
    gpio->MODER |= (mode & 3) << (n * 2);  
    // Set new mode  
}
```

De code voor A3 spreekt nu voor zich:

```
uint16_t pin = PIN('A', 3); // Pin A3  
gpio_set_mode(pin, GPIO_MODE_OUTPUT); // Set to output
```

Merk op dat we zo een nuttige initiële API hebben gemaakt voor de GPIO-periferie. Andere periferie, zoals UART (seriële communicatie) en dergelijke, kan op soortgelijke wijze worden geïmplementeerd. Dit is een goede programmeerpraktijk, omdat het code zelfverklarend en voor mensen leesbaar maakt.

MCU boot- en vectortabel

Wanneer een ARM-MCU opstart, leest hij een zogenaamde ‘vectortabel’ uit het begin van het flash-geheugen. Een vectortabel is een concept dat alle ARM-MCU's gemeen hebben: een array met

32bit-adressen van interrupt handlers. De eerste 16 ingangen zijn gereserveerd door ARM en zijn gemeenschappelijk voor alle ARM-MCU's. De rest van de interrupt handlers zijn specifiek voor de betreffende MCU – dit zijn interrupt handlers voor periferie. Eenvoudigere MCU's met weinig periferie hebben weinig interrupt handlers en complexere MCU's hebben er veel.

De vectortabel voor de STM32F429 is gedocumenteerd in tabel 62 van de referentiehandleiding [1]. Daaruit blijkt dat er, naast de 16 standaard handlers, 91 perifere handlers zijn.

Elke ingang in de vectortabel is een adres van een functie die de MCU uitvoert wanneer een hardware-interrupt (IRQ) wordt getriggert. De uitzonderingen zijn de eerste twee ingangen, die een sleutelrol spelen in het opstartproces van de MCU. Deze twee eerste waarden zijn een initiële stackpointer en het adres van de bootfunctie (het beginpunt voor de uitvoering van firmware).

We weten nu dus dat we ervoor moeten zorgen dat onze firmware zo is samengesteld dat de tweede 32bit-waarde in het flash-geheugen het adres van onze bootfunctie bevat. Wanneer de MCU opstart, zal hij dat adres uit het flash-geheugen lezen en naar onze bootfunctie springen.

Minimale firmware

Laten we een bestand maken, *main.c*, en onze bootfunctie specificeren, die aanvankelijk niets doet (in een eindeloze lus terechtkomt), en daarnaast een vectortabel opstellen die 16 standaard-ingangen en 91 STM32-ingangen bevat. Maak en open het bestand *main.c* in een editor naar keuze en tik het volgende in:

```
// Startup code  
__attribute__((naked, noreturn)) void _reset(void) {  
    for (;;) (void) 0; // Infinite loop  
}  
  
extern void _estack(void); // Defined in link.ld  
  
// 16 standard and 91 STM32-specific handlers  
__attribute__((section(".vectors")))  
void (*tab[16 + 91])(void) = {_estack, _reset};
```

Voor de functie `_reset()` hebben we GCC-specifieke attributen `naked` en `noreturn` gebruikt – ze betekenen dat de standaard functie-proloog en -epiloog niet door de compiler moeten worden aangemaakt, en dat de functie niet terugkeert.

De uitdrukking `void (*tab[16 + 91])(void)` betekent: definieer een array van 16 + 91 POINTERS naar functies, die niets retourneren (`void`), en twee argumenten nemen. Elk van die functies is een IRQ-handler (Interrupt ReQuest-handler). Een array van die handlers wordt een vectortabel genoemd.

De vectortabel `tab` zetten we in een aparte sectie met de naam `.vectors` – die hebben we later nodig om de linker te vertellen dat die sectie helemaal aan het begin van de gegenereerde firmware moet staan, in die volgorde, aan het begin van het flash-geheugen. De eerste twee ingangen zijn de waarde van het stackpointer-register en het beginpunt van de firmware. De rest van de vectortabel laten we gevuld met nullen.

Compileren

Laten we onze code compileren. Start een terminal (of een opdracht-regel in Windows) en voer dit uit:

```
$ arm-none-eabi-gcc -mcpu=cortex-m4 main.c -c
```

Het werkt! De compilatie leverde een bestand *main.o* op dat onze minimale firmware bevat die niets doet. Het *main.o*-bestand is in ELF binair formaat, dat verschillende secties bevat. We bekijken ze in **listing 1**.

Merk op dat de VMA/LMA sectie-adressen op 0 staan, wat betekent dat *main.o* nog geen complete firmware is omdat het niet de informatie bevat over waar die secties in de adresruimte moeten worden geladen. We moeten een linker gebruiken om het firmwarebestand, *firmware.elf*, te completeren vanuit *main.o*.

De *.text*-sectie bevat firmwarecode, in ons geval slechts een *_reset()*-functie van twee bytes lang – een jump-instructie naar zijn eigen adres. Er is een lege *.data*-sectie en een lege *.bss*-sectie [8] (voor niet-geïnitieerde, maar gedeclareerde variabelen wordt deze sectie meestal gevuld met 0). Onze firmware zal worden gekopieerd naar het flash-bereik op offset 0x8000000, maar onze datasectie moet zich in RAM bevinden – daarom moet onze *_reset()*-functie de inhoud van de *.data*-sectie naar RAM kopiëren. Ook moet hij nullen schrijven naar de gehele *.bss*-sectie. In ons geval zijn de *.data*- en *.bss*-secties leeg, maar laten we onze *_reset()*-functie desondanks aanpassen om ze correct te behandelen.

Daartoe moeten we weten waar de stack begint, en waar de data- en bss-secties beginnen. Dit kunnen we aangeven in het linker-script, een bestand met instructies aan de linker over waar de verschillende secties in de adresruimte moeten komen, en welke symbolen moeten worden aangemaakt.

Linker-script

Maak een bestand met de naam *link.ld*, en kopieer/plak daar de inhoud uit [4]. Hieronder volgt de uitleg:

```
ENTRY(_reset);
```

Deze regel vertelt de linker de waarde van het 'entry point'-attribuut in de gegenereerde ELF-header – dit is dus een duplicaat van wat een vectortabel heeft. Dit is een hulpmiddel voor een debugger (zoals Ozone, beschreven in het tweede artikel van deze reeks) dat ons helpt een breakpoint in te stellen aan het begin van de firmware. Een debugger kent geen vectortabel, dus vertrouwt hij op de ELF-header.

```
MEMORY {
flash(rx) : ORIGIN = 0x08000000, LENGTH = 2048k
sram(rwx) : ORIGIN = 0x20000000, LENGTH = 192k /*
remaining 64k in a separate address space */
}
```

Dit vertelt de linker dat we twee geheugenbereiken in de adresruimte hebben, plus hun adressen en omvang.

```
_estack = ORIGIN(sram) + LENGTH(sram); /* stack points
to end of SRAM */
```

Dit vertelt de linker om een symbool, *_estack*, aan te maken dat een waarde bevat helemaal aan het einde van het RAM-geheugen-bereik. Dit zal onze initiële stackwaarde zijn!

```
.vectors : { KEEP(*(.vectors)) } > flash
.text : { *(.text*) } > flash
.rodata : { *(.rodata*) } > flash
```

Deze regels vertellen de linker de vectortabel eerst in flash te plaatsen, gevolgd door de *.text*-sectie (firmwarecode), gevolgd door alleen-lezen data *.rodata*.

Dan is de *.data*-sectie aan de beurt:

```
.data : {
_sdata = .; /* .data section start */
*(.first_data)
*(.data SORT(.data.*))
```



Listing 1. Compilatie van main.o.

```
$ arm-none-eabi-objdump -h main.o
```

Idx	Name	Size	VMA	LMA	File off	Algn
0	.text	00000002	00000000	00000000	00000034	2**1
	CONTENTS, ALLOC, LOAD, READONLY, CODE					
1	.data	00000000	00000000	00000000	00000036	2**0
	CONTENTS, ALLOC, LOAD, DATA					
2	.bss	00000000	00000000	00000000	00000036	2**0
	ALLOC					
3	.vectors	000001ac	00000000	00000000	00000038	2**2
	CONTENTS, ALLOC, LOAD, RELOC, DATA					
...						

Listing 2. Opstartcode.

```
int main(void) {
    return 0; // Do nothing so far
}

// Startup code
__attribute__((naked, noreturn)) void _reset(void) {
    // memset .bss to zero, and copy .data section to RAM region
    extern long _sbss, _ebss, _sdata, _edata, _sidata;
    for (long *src = &_amp;sbss; src < &_amp;ebss; src++) *src = 0;
    for (long *src = &_amp;sdata, *dst = &_amp;sidata; src < &_amp;edata;) *src++ = *dst++;

    main(); // Call main()
    for (;;) (void) 0; // Infinite loop in the case if main() returns
}
```

```
_edata = .; /* .data section end */
} > sram AT > flash
_sidata = LOADADDR(.data);
```

Merk op dat we de linker opdracht geven om `_sdata`- en `_edata`-symbolen aan te maken. We zullen ze gebruiken om de gegevenssectie naar het RAM te kopiëren in de `_reset()`-functie. Hetzelfde geldt voor de `.bss`-sectie:

```
.bss : {
    _sbss = .; /* .bss section start */
    *(.bss SORT(.bss.*) COMMON)
    _ebss = .; /* .bss section end */
} > sram
```

Opstartcode

Nu kunnen we onze `_reset()`-functie bijwerken. We kopiëren de `.data`-sectie naar RAM, en initialiseren de `.bss`-sectie met nullen. Dan roepen we de `main()`-functie aan – en komen in een eindeloze lus terecht wanneer `main()` retourneert; zie **listing 2**.

Het diagram van **figuur 4** illustreert hoe `_reset()` `.data` en `.bss` initialiseert.

Het bestand `firmware.bin` is gewoon een aaneenrijging van de drie secties `.vectors` (IRQ-vectortabel), `.text` (code) en `.data` (gegevens). Deze secties zijn opgebouwd volgens het linker-script: `.vectors` staat helemaal aan het begin van het flash-bereik, `.text` volgt onmiddellijk daarna en `.data` ligt ver daarboven. Adressen

in `.text` liggen in het flash-geheugenbereik, en adressen in `.data` liggen in het RAM-bereik. Als een functie een adres heeft, bijvoorbeeld `0x8000100`, dan bevindt deze zich precies op dat adres in flash. Maar als de code een variabele in de `.data`-sectie benadert met zijn adres, bijvoorbeeld `0x20000200`, dan is er niets op dat adres, omdat bij het opstarten de `.data`-sectie in het bestand `firmware.bin` zich in flash bevindt! Daarom moet de opstartcode de `.data`-sectie verplaatsen van het flash-geheugenbereik naar het RAM-bereik. Nu zijn we klaar om het volledige firmware-bestand, `firmware.elf`, te produceren:

```
$ arm-none-eabi-gcc -T link.ld -nostdlib main.o -o
firmware.elf
```

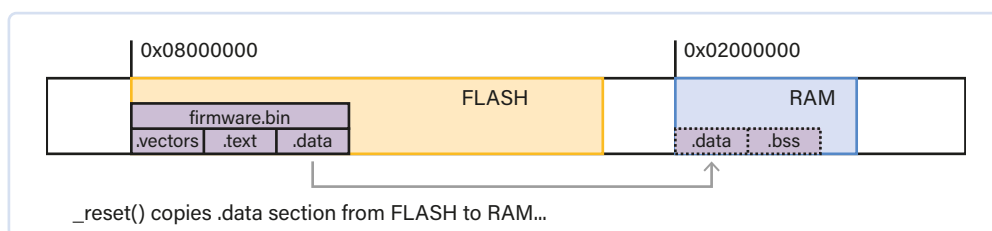
Laten we secties in `firmware.elf` nu eens nader bekijken – zie **listing 3**.

Nu zien we dat de `.vectors`-sectie helemaal aan het begin van het flash-geheugen staat, op adres `0x80000000`, en de `.text`-sectie direct daarna, op `0x80001ac`. Onze code maakt geen variabelen aan, dus er is geen datasectie.

De firmware flashen

We zijn klaar om deze firmware te flashen! Extraheer eerst de secties van `firmware.elf` naar een enkele, aaneengesloten portie binaire code:

```
$ arm-none-eabi-objcopy -O binary firmware.elf firmware.bin
```



Figuur 4. De figuur maakt inzichtelijk hoe `_reset()` `.data` en `.bss` initialiseert.



Listing 3. De secties in *firmware.elf*.

000011000000

```
$ arm-none-eabi-objdump -h firmware.elf
...
Idx Name           Size      VMA       LMA       File off  Algn
  0 .vectors        000001ac  08000000  08000000  00010000  2**2
CONTENTS, ALLOC, LOAD, DATA
  1 .text           00000058  080001ac  080001ac  000101ac  2**2
CONTENTS, ALLOC, LOAD, READONLY, CODE
...
```

Gebruik dan het hulpprogramma *st-link* om *firmware.bin* te flashen. Verbind je board via USB en voer dit uit:

```
$ st-flash --reset write firmware.bin 0x80000000
```

Klaar! Hiermee hebben we firmware geflashed die niets doet.

Makefile: build-automatisering

In plaats van al die compilatie-, link- en flashcommando's in te typen, kunnen we het *make*-hulpprogramma gebruiken om het hele proces te automatiseren. Het *make*-hulpprogramma gebruikt een configuratiebestand met de naam *Makefile*, waarin het instructies leest voor het uitvoeren van acties. Deze automatisering is een geweldige uitvinding omdat het ook het proces van het bouwen van firmware, de gebruikte compilatievlaggen enzovoort documenteert. Er bestaat een heel goede *Makefile*-tutorial [5]. Degenen die onbekend zijn met *make*, raad ik aan deze door te kijken. Hieronder hebben we de belangrijkste concepten opgesomd die je moet kennen om onze simpele en puristische *Makefile* te begrijpen. Wie al bekend is met *make*, kan dit gedeelte overslaan.

Het *Makefile*-formaat is eenvoudig:

```
action1:
    command ... # Comments can go after hash symbol
    command .... # IMPORTANT: command must be preceded
with the TAB character
action2:
    command ... # Don't forget about TAB. Spaces won't
work!
```

Nu kunnen we *make* aanroepen met de naam van de actie (ook wel *target* genoemd) om de corresponderende actie uit te voeren:

```
$ make action1
```

Het is mogelijk om variabelen te definiëren en in commando's te gebruiken. Ook kunnen acties de namen zijn van bestanden die moeten worden aangemaakt:

```
firmware.elf:
    COMPILATION COMMAND .....
```

Ook kan elke actie een lijst van afhankelijkheden hebben. *firmware.elf* is bijvoorbeeld afhankelijk van ons bronbestand *main.c*. Als het bestand *main.c* verandert, bouwt het *make build*-commando *firmware.elf* opnieuw:

```
build: firmware.elf
firmware.elf: main.c
    COMPILATION COMMAND
```

Nu zijn we klaar om een *Makefile* te schrijven voor onze firmware. We definiëren een *build* actie/target:

```
CFLAGS ?= -W -Wall -Wextra -Werror -Wundef -Wshadow
-Wdouble-promotion \ -Wformat-truncation -fno-common
-Wconversion \ -g3 -Os -ffunction-sections -fdata-
sections -I. \ -mcpu=cortex-m4 -mthumb -mfloat-abi=hard
-mfpu=fpv4-sp-d16 $(EXTRA_CFLAGS)
LDFLAGS ?= -Tlink.ld -nostartfiles -nostdlib --specs nano.
specs -lc -lgcc -Wl,--gc-sections -Wl,-Map=$@.map
SOURCES = main.c
build: firmware.elf
firmware.elf: $(SOURCES)
    arm-none-eabi-gcc $(SOURCES) $(CFLAGS) $(LDFLAGS)
-o $@
```

Hier definiëren we compilatievlaggen. De *?* staat voor een standaardwaarde; we kunnen ze op deze manier vanaf de opdrachtregel overschrijven:

```
$ make build CFLAGS="-O2 ...."
```

We specificeren *CFLAGS*-, *LDFLAGS*- en *SOURCES*-variabelen. Dan vertellen we *make*: als je de opdracht krijgt om te bouwen, maak dan een *firmware.elf*-bestand. Het is afhankelijk van het bestand *main.c* en om het te maken moet je de *arm-none-eabi-gcc* compiler starten met de gegeven flags. De speciale variabele *\$@* wordt geëxpandeerd tot een target-naam – in ons geval *firmware.elf*.

Laten we *make* aanroepen:

```
$ make build arm-none-eabi-gcc main.c -W -Wall -Wextra
-Werror -Wundef -Wshadow -Wdouble-promotion -Wformat-
truncation -fno-common -Wconversion -g3 -Os -ffunction-
sections -fdata-sections -I. -mcpu=cortex-m4 -mthumb
-mfpu=fpv4-sp-d16 -Tlink.ld -nostartfiles -nostdlib
--specs nano.specs -lc -lgcc -Wl,--gc-sections
-Wl,-Map=firmware.elf.map -o firmware.elf
```

Als we het opnieuw uitvoeren:

```
$ make build
make: Nothing to be done for 'build'.
```

Listing 4. Blinky LED-sectie van het main.c bestand.

```
#include <inttypes.h>
#include <stdbool.h>
#define BIT(x) (1UL << (x))
#define PIN(bank, num) (((bank) - 'A') << 8) | (num))
#define PINNO(pin) (pin & 255)
#define PINBANK(pin) (pin >> 8)

struct gpio {
    volatile uint32_t MODER, OTYPER, OSPEEDR, PUPDR, IDR, ODR, BSRR, LCKR, AFR[2];
};
#define GPIO(bank) ((struct gpio *) (0x40020000 + 0x400 * (bank)))

// Enum values are per datasheet: 0, 1, 2, 3
enum { GPIO_MODE_INPUT, GPIO_MODE_OUTPUT, GPIO_MODE_AF, GPIO_MODE_ANALOG };

static inline void gpio_set_mode(uint16_t pin, uint8_t mode) {
    struct gpio *gpio = GPIO(PINBANK(pin)); // GPIO bank
    int n = PINNO(pin); // Pin number
    gpio->MODER &= ~(3U << (n * 2)); // Clear existing setting
    gpio->MODER |= (mode & 3) << (n * 2); // Set new mode
}
```

De `make`-utility onderzoekt de modificatietijden voor de `main.c`-dependency en `firmware.elf` – en doet niets als `firmware.elf` actueel is. Maar als we `main.c` veranderen, zal de volgende `make build` opnieuw een compilatie uitvoeren:

```
$ touch main.c # Simulate changes in main.c
$ make build
```

Wat overblijft is het `flash`-target:

```
firmware.bin: firmware.elf
$(DOCKER) $(CROSS)-objcopy -O binary $< $@ flash:
firmware.bin
st-flash --reset write $(TARGET).bin 0x8000000
```

Dat is alles! Nu maakt het `make flash`-terminalcommando een `firmware.bin`-bestand aan en flasht het naar het board. Het zal de firmware opnieuw compileren als `main.c` verandert, omdat `firmware.bin` afhankelijk is van `firmware.elf`, en dat is op zijn beurt weer afhankelijk van `main.c`. Dus nu zou de ontwikkelingscyclus uit deze twee acties in een lus bestaan:

```
# Develop code in main.c
$ make flash
```

Het is een goed idee om een schoon target toe te voegen om build-artefacten te verwijderen:

```
clean:
    rm -rf firmware.*
```

De volledige broncode van het project staat in de map *Step 0 minimal* [6].

Knipper-LED

Nu we de hele build-/flash -infrastructuur hebben opgezet, is het tijd om onze firmware te leren iets nuttigs te doen. Zoals het laten knipperen van een LED. Een Nucleo-F429ZI-board heeft drie LED's aan boord. In de gebruikershandleiding van het Nucleo-board [2] kunnen we in paragraaf 6.5 zien op welke pinnen deze LED's zijn aangesloten:

- PBo: groene LED
- PB7: blauwe LED
- PB14: rode LED

Laten we het bestand `main.c` aanpassen en onze definities voor `PIN`, `gpio_set_mode()` toevoegen. In de `main()`-functie zetten we de blauwe LED op uitgangsmodus en starten we een oneindige lus. Eerst kopiëren we de definities voor de pinnen en GPIO die we eerder hebben besproken. Merk op dat we ook voor het gemak macro toevoegen, `BIT(x)` – zie **listing 4**.

Bij sommige microcontrollers wordt alle periferie automatisch ingeschakeld en van stroom voorzien. Bij STM32-MCU's is de periferie echter standaard uitgeschakeld om energie te sparen. Om een GPIO-randapparaat in te schakelen, moet het expliciet worden ingeschakeld (geklakt) via de RCC-unit (Reset en Clock Control). In paragraaf 7.3.10 van de referentiehandleiding vinden we dat het `AHB1ENR` (AHB1 peripheral clock enable-register) verantwoordelijk is voor het in- of uitschakelen van GPIO-banken. Eerst voegen we een definitie toe voor de complete RCC-unit:

```
struct rcc {
    volatile uint32_t CR, PLLCFGR, CFGR, CIR, AHB1RSTR,
    AHB2RSTR, AHB3RSTR, RESERVED0, APB1RSTR, APB2RSTR,
    RESERVED1[2], AHB1ENR, AHB2ENR, AHB3ENR, RESERVED2,
    APB1ENR, APB2ENR, RESERVED3[2], AHB1LPENR, AHB2LPENR,
```



```
AHB3LPENR, RESERVED4, APB1LPENR, APB2LPENR, RESERVED5[2],
BDCR, CSR, RESERVED6[2], SSCGR, PLLI2SCFGR;
};
#define RCC ((struct rcc *) 0x40023800)
```

In de documentatie van het AHB1ENR-register zien we dat bits 0 tot en met 8 de klok instellen voor GPIO-banken GPIOA...GPIOE:

```
int main(void) {
    uint16_t led = PIN('B', 7); // Blue LED
    RCC->AHB1ENR |= BIT(PINBANK(led));
    // Enable GPIO clock for LED
    gpio_set_mode(led, GPIO_MODE_OUTPUT);
    // Set blue LED to output mode
    for (;;) asm volatile("nop"); // Infinite loop
    return 0;
}
```

Nu moeten we nog uitzoeken hoe we een GPIO-pin aan of uit kunnen zetten, en vervolgens de hoofdlus aanpassen om een LED-pin in te schakelen, even te wachten, uit te schakelen en weer even te wachten. Als we een blik werpen op paragraaf 8.4.7 van de referentiehandleiding, zien we dat register BSRR verantwoordelijk is voor het hoog of laag maken van een pin. De lage 16 bits worden gebruikt om het ODR-register in te stellen (dat wil zeggen pin hoog te maken), en de hoge 16 bits worden gebruikt om het ODR-register te resetten (dus een pin laag te maken). Laten we daarvoor een API-functie definiëren:

```
static inline void gpio_write(uint16_t pin, bool val) {
    struct gpio *gpio = GPIO(PINBANK(pin));
    gpio->BSRR |= (1U << PINNO(pin)) << (val ? 0 : 16);
}
```

Vervolgens moeten we een vertragsingsfunctie implementeren. We hebben nu geen nauwkeurige vertraging nodig, dus laten we een functie definiëren met de naam `spin()` die gewoon een bepaald aantal keren een NOP-instructie uitvoert:

```
static inline void spin(volatile uint32_t count) {
    while (count--) asm("nop");
}
```

Nu zijn we klaar om onze hoofdlus aan te passen om een LED te laten knipperen:

```
for (;;) {
    gpio_write(pin, true);
    spin(999999);
    gpio_write(pin, false);
    spin(999999);
}
```

De volledige broncode van het project is te vinden in de map *Step 1 blinky* [7]. Voer `make flash` uit en geniet van de knipperende blauwe LED!

In het tweede deel van deze reeks komen zaken als UART-uitvoer, debuggen, een webserver-implementatie, automatische tests en meer aan de orde. Tot dan! ◀

220665-03

Over de auteur

Sergey Lyubka is ingenieur en ondernemer. Hij heeft een MSc in natuurkunde van de staatsuniversiteit van Kiev, Oekraïne. Sergey is directeur en medeoprichter van Cesanta, een technologiebedrijf gevestigd in Dublin, Ierland (Embedded Web Server voor elektronische apparaten: <https://mongoose.ws>). Zijn passie is puristische embedded netwerkprogrammering.

Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur via sergey.lyubka@cesanta.com of naar de redactie van Elektor via redactie@elektor.com.



Gerelateerde producten

- **Dogan Ibrahim, Nucleo Boards Programming with the STM32CubeIDE, Elektor**
www.elektor.nl/19530
- **Dogan Ibrahim, Programming with STM32 Nucleo Boards, Elektor**
www.elektor.nl/18585

WEBLINKS

- [1] Referentiehandleiding RM0090 voor STM32F429: <https://bit.ly/3neE7S7>
- [2] Nucleo-144 Board gebruikershandleiding (UM1974): <https://bit.ly/3oIBXKZ>
- [3] Dit artikel op GitHub: <https://github.com/cpq/bare-metal-programming-guide>
- [4] Inhoud van het link.ld-bestand: <https://github.com/cpq/bare-metal-programming-guide/blob/main/step-0-minimal/link.ld>
- [5] Makefile-tutorial: <https://makefiletutorial.com/>
- [6] Step 0 minimal demoprogramma: <https://github.com/cpq/bare-metal-programming-guide/blob/main/step-0-minimal>
- [7] Step 1 blinky demoprogramma: <https://github.com/cpq/bare-metal-programming-guide/blob/main/step-1-blinky>
- [8] .bss [Wikipedia]: <https://en.wikipedia.org/wiki/.bss>



Siglent SDM3045X-multimeter

Philippe Demerliac (Frankrijk)

Een gewone handheld multimeter doet zijn werk goed en is altijd handig om onder handbereik te hebben. Een tafelmultimeter biedt daarentegen veel meer mogelijkheden die het leven in het lab veel aangenamer maken. De SDM3045X-tafelmultimeter van Siglent is zo'n apparaat; ik zal hem hier bespreken.

Ik ben al in het bezit van verschillende Siglent-apparaten, die ik erg goed vind. Ik miste echter een tafelmultimeter uit hun uitgebreide assortiment. Nu ik deze leemte heb opgevuld met de SDM3045X, kan ik mijn indrukken delen.

Siglent heeft drie multimeterfamilies in de aanbieding: SDM3045X, SDM3055 en SDM3065X. Deze modellen hebben vergelijkbare functionaliteit maar verschillen in resolutie. De 3045 heeft 4½ digit (60.000 counts), de 3055 heeft 5½ digit (240.000 counts) en de 3065 heeft 6½ digit (2.200.000 counts). De 3055 en 3065 bieden ook betere prestaties in de lage meetbereiken: 200 mV/200 µA tegen 600 mV/600 µA voor de 3045. Bovendien bieden de 3055 en 3065 de SC-optie, die meerdere programmeerbare meetingangen mogelijk maakt. Deze modellen hebben dezelfde behuizing en ergonomie. De prijzen variëren uiteraard met de resolutie.

Resolutie en nauwkeurigheid

Laten we het eerst eens hebben over resolutie. Deze wordt uitgedrukt als een aantal significante cijfers of counts. Het aantal counts geeft het aantal

verschillende waarden aan dat kan worden weergegeven binnen een bepaald bereik. Bij 600 mV bijvoorbeeld, het bereik met de hoogste resolutie, kan de 3045 stapjes van 10 µV/count weergeven (0,6 V/60.000 counts). Elk hoger bereik – 6 V, 60 V, 600 V, 1000 V – brengt een verlaging van de resolutie met zich mee, waarbij het 1000V-bereik 100 mV/count heeft. (Merk op dat het hoogste bereik om veiligheidsredenen 1000 V is en geen 6000 V).

Bij modellen met een hogere resolutie zijn er meer significante cijfers. Verwar resolutie niet met nauwkeurigheid. Als we een meting van 1,0000 V aflezen op het display, weten we dan zeker dat de spanning 1 V is met een nauwkeurigheid van 100 µV? Nee, want zelfs bij goede kalibratie noemen de specificaties een maximale fout van 0,06% ± 8 digits. De werkelijke spanning kan dus elke waarde hebben tussen 0,9986 V en 1,0014 V, waarbij deze grenzen maxima zijn.

Betekent dit dat die laatste cijfers nutteloos zijn? Nee, want ze maken het mogelijk om metingen te vergelijken en hun verloop in de tijd te observeren. Terugkomend op onze SDM3045X, is 60.000 counts genoeg of moeten we een ander model kiezen? Het antwoord op deze vraag hangt natuurlijk af van het gebruik. Wat ik kan zeggen is dat deze resolutie voor een amateur of zelfs een professional meer dan genoeg is, vaak zelfs meer dan nodig. Ik heb multimeters met lagere resoluties gebruikt en gebruik ze nog steeds zonder dat dit (in 90% van de gevallen) een probleem is.

Je kunt de datasheet van het model eenvoudig downloaden van Siglent [1] en de maximale resolutie en fout zien voor elk type meting en, voor elk bereik, de maximale resolutie en fout die door de fabrikant worden gegarandeerd. Over nauwkeurigheid gesproken, de multimeters worden geleverd met een kalibratiecertificaat dat bewijst dat het apparaat binnen de aangegeven limieten blijft, en vaak zelfs beter presteert.



Siglent zegt in dit verband:

"SIGLENT heeft vastgesteld dat de fabriekskalibratie van dit instrument niet significant wordt beïnvloed door opslag tot 180 dagen voor het eerste gebruik. Het kalibratie-interval moet beginnen wanneer het apparaat in gebruik wordt genomen of 180 dagen na de 'kalibratiedatum' op het certificaat dat bij het apparaat is geleverd."

Moeten deze multimeters periodiek opnieuw worden gekalibreerd? In een professionele omgeving is het raadzaam om dit te controleren als je aan bepaalde normen wilt voldoen, maar voor een amateur is dit optioneel. De ervaring leert dat moderne materialen niet veel verlopen met de tijd. Als je daartoe in staat bent, is het aan te raden om om de nauwkeurigheid van het apparaat zo af en toe te controleren.

Er zijn erkende meetcentra die deze kalibraties kunnen uitvoeren, maar dat is een relatief kostbare grap vergeleken met de prijs van het apparaat. Zelfs als Siglent niet al te veel vertelt over de kalibratie, zijn er hacks op het internet om het zelf te doen, maar dan moet je nog steeds de juiste referenties ter beschikking hebben. Samenvattend kunnen we stellen dat dit meestal geen probleem is en je multimeter je vele jaren trouw van dienst zal zijn.

Eerste kennismaking met de SDM3045X

De SDM3045X is een tafelmultimeter en is dus, hoewel verplaatsbaar, niet echt bedoeld voor mobiel gebruik in het veld. Hij is al niet autonoom en heeft een netvoeding nodig, dan moet hij op een tafel worden geplaatst of op zijn minst op een stabiel oppervlak om hem te kunnen gebruiken. In vergelijking met handheld modellen bieden tafelmultimeters over het algemeen betere en extra functies, zoals connectiviteit, 4-draads metingen, een informatiever display enzovoort. Je kunt ze ook veilig recht voor je plaatsen, wat het gebruik duidelijk comfortabeler maakt, en de netvoeding maakt het gehannes met batterijen of periodiek opladen overbodig.

De SDM3045X kan (RMS) DC- en AC-spanningen en -stromen meten tot 100 kHz (perfect voor LF!), weerstanden (tweedraads of vierdraads voor zeer kleine weerstandswaarden), doorgangstest, diode-test, condensatoren en temperatuur met verschillende sensoren (hij heeft koudebrug-compensatie voor thermokoppels). Dit dekt de meeste gangbare behoeften. Het is ook mogelijk om de afgelezen waarden in een geheugen op te slaan, statistieken te maken en alarmen te zien in het geval van grenswaarde-overschrijdingen.

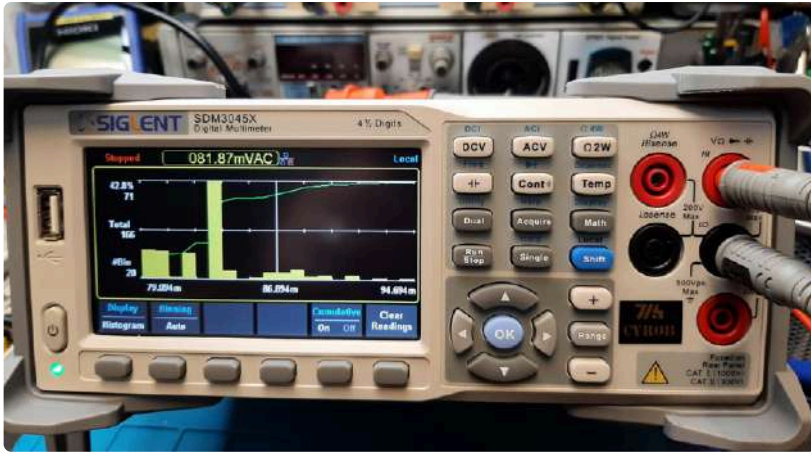
Net als de andere Siglent-apparaten wordt de multimeter geleverd in een goed ontworpen verpakking die hem perfect beschermt tegen schokken.

Ik raad je aan om die verpakking goed te bewaren. Het apparaat wordt geleverd met een netsnoer, twee flexibele meetsnoeren van uitstekende kwaliteit, een USB A/B-kabel voor aansluiting op een PC, een kalibratiecertificaat en een basis-gebruikershandleiding in het Engels. Gedetailleerde handleidingen, ook in andere talen, kunnen worden gedownload van de Siglent-website.

Het toestel straalt kwaliteit uit en de afwerking is onberispelijk. De zachte toetsen voelen prettig aan. De aan/uit-knop is geen 'echte' schakelaar, dus het apparaat staat altijd op stand-by. Persoonlijk geef ik de voorkeur aan een echte ontkoppeling van het lichtnet, en schakel ik alles uit als ik deze apparaten niet gebruik, via een stopcontact met een hoofdschakelaar. Het display is goed leesbaar. Bij het opstarten duurt het initialiseren enkele tientallen seconden. In ieder geval is het aan te raden om deze apparaten niet voor korte periodes aan of uit te zetten, omdat de maximale nauwkeurigheid pas gegarandeerd is na een opwarmperiode van 10...30 min. Maar we kunnen ze natuurlijk ook eerder gebruiken. In tegenstelling tot de superieure modellen met hun ietwat luidruchtige ventilatoren heeft de SDM3045X geen ventilatoren en is daarom stil, afgezien van een paar discreet klikkende relais. Het achterpaneel (**figuur 1**) heeft twee BNC-connectoren, een ingang om de metingen te triggeren via een extern signaal en een uitgang die aangeeft dat de meting voltooid is. Deze aansluitingen zijn vooral handig voor geautomatiseerde meetopstellingen, maar worden zelden gebruikt in de dagelijkse praktijk. Er is ook een RJ45 LAN-connector en een USB-B plug om het apparaat op een PC aan te sluiten. De SDM3045X is ook voorzien van een Kensington-beveiligingssleuf voor het bevestigen van een anti-diefstalkabel. De 10A-zekering

Figuur 1.
Het achterpaneel.





Figuur 2. De histogram-modus.

van de ampèremeter kan ook gemakkelijk worden vervangen in het geval van een overstroomgebeurtenis. In eerste instantie raad ik je aan om te controleren of de firmware up-to-date is. Siglent werkt zijn apparaten regelmatig bij en de updates zijn eenvoudig te downloaden op hun site. Updates worden uitgevoerd via een USB-stick die in het frontpaneel wordt gestoken.

Basisgebruik

Het basisgebruik is eenvoudig en intuïtief. Je kunt eenvoudig van de ene meetmodus naar de andere overschakelen via de knoppen aan de voorkant. Het is mogelijk om de meetsnelheid te wijzigen, wat de temporele resolutie beïnvloedt. In veel gevallen zal de snelle modus goed genoeg zijn, met een bijna onmiddellijke reactie. De automatische verandering van bereik vermindert ook de benodigde handelingen aanzienlijk. Het bereik kan echter op elk moment eenvoudig handmatig worden geselecteerd.

Voor AC spanning- en stroommeting biedt hij 100 kHz bandbreedte en true RMS-meting. Zoals bij de meeste multimeters moet de piekfactor echter handmatig worden gecorrigeerd voor asymmetrische signalen; de handleiding geeft hierover alle details.

In de doorgangs- of diodetestmodus kunnen we de drempel van de pieptoon instellen, evenals het volume, wat een pluspunt is. Bij alle metingen kunnen we een 'relatieve' modus instellen om de metingen ten opzichte van een vooraf gedefinieerde waarde uit te voeren. Je kunt de acquisitie ook op elk moment handmatig stoppen en de laatste meting in het geheugen opslaan.

Interessante functies

De SDM3045X gedraagt zich als een normale multimeter, maar biedt extra functies die het leven gemakkelijker kunnen maken. Dit zijn de belangrijkste:

- Verschillende weergavemodi: met een staafdiagram kun je de orde van grootte van een meting en de variatie ervan visualiseren, de histogrammodus (**figuur 2**) toont de statistische verdeling van de metingen grafisch, zodat je meteen de meest frequente waarden kunt zien, en een curvemodus toont het verloop van de meting in de tijd, zelfs voor perioden zo kort als een seconde. Dit is erg handig voor het observeren van de verandering

van een waarde in de loop van de tijd. In al deze modi kunnen schalen en limieten automatisch worden berekend of handmatig worden ingesteld.

- Een statistische modus toont het gemiddelde, min, max, standaardafwijking enzovoort van de metingen.
- Met het dubbele display (**figuur 3**) kun je twee parameters tegelijk controleren, zoals spanning en frequentie.
- Een dB/dBm-modus maakt relatieve metingen in dB of vermogensmetingen in dBm mogelijk voor een instelbare belastingsimpedantie. (LET OP: deze instelling verandert de ingangsimpedantie van de multimeter niet; je moet de gewenste belasting extern aansluiten).
- De limiet-modus (**figuur 4**) geeft visueel aan of de gemeten waarde tussen twee instelbare grenswaarden ligt, wat erg handig is voor herhaalde instellingen zonder de waarden te hoeven interpreteren.
- De Probe Hold-modus (**figuur 5**), een van mijn favorieten, onthoudt automatisch opeenvolgende stabiele metingen; dat komt van pas om componenten te sorteren.
- Je kunt de acquisitiemodus, de samplesnelheid en of het automatisch, handmatig of getriggerd wordt door een extern triggersignaal met een definieerbare waarde en polariteit nauwkeurig afstellen.



Figuur 3. Dubbel display.



Figuur 4. De limietmodus.



Figuur 5. De Probe Hold-modus.

- Meetwaarden en/of de huidige instellingen kunnen worden opgeslagen op een USB-stick zodat ze eenvoudig kunnen worden opgeroepen.
- Voor temperatuurmetingen accepteert het instrument verschillende types gewone thermokoppels (met koudebrug-compensatie) en resistieve sensoren. (Opmerking: er wordt geen temperatuursonde meegeleverd met het apparaat).
- Een ingebouwde hulpfunctie herinnert je aan het juiste gebruik en de juiste aansluitingen voor metingen (in het Engels).

Gebruik met een PC

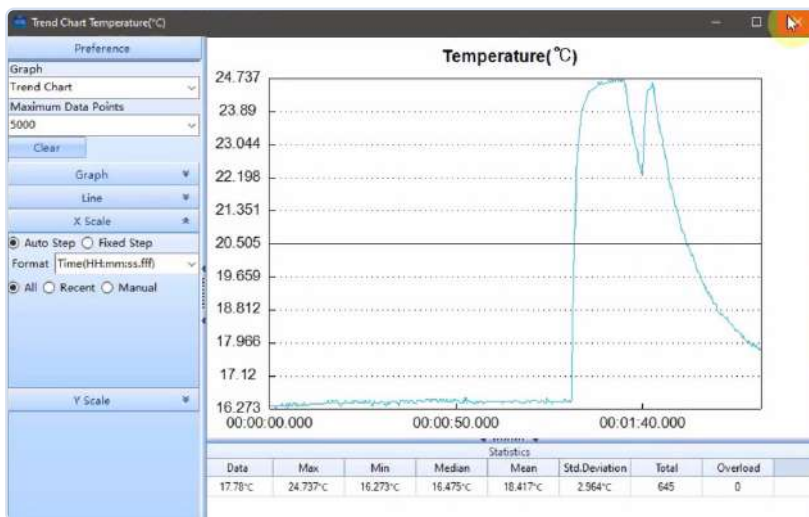
Het apparaat kan worden aangestuurd via SPI, USB of LAN Ethernet. LabVIEW stuurprogramma's zijn beschikbaar op de website van Siglent. Voor de bediening van het apparaat wordt protocoldocumentatie meegeleverd. Daarnaast biedt Siglent de gratis Windows EasyDMM-applicatie (**figuur 6**), waarmee alle Siglent-multimeters eenvoudig kunnen worden aangestuurd, metingen worden bekeken en naar CSV worden geëxporteerd. Deze applicatie werkt, ondanks zijn wat verouderde uiterlijk, erg goed.

Samenvattend

Ik ben nog nooit teleurgesteld door meetapparatuur van Siglent, en deze multimeter heeft over het algemeen ook een goede indruk op me gemaakt. Hij is handig in het gebruik en dekt waarschijnlijk de meeste behoeften van een service- of R&D-lab. De prijs is iets hoger dan van Aziatische instapmodellen, maar de kwaliteit van het product rechtvaardigt dat. De prijs/kwaliteit-verhouding is uitstekend, en als je wilt investeren in een praktische en efficiënte tafel-multimeter, dan raad ik je deze zeker aan.

Ik heb vooral hiervan genoten:

- gebruiksgemak;
- algehele kwaliteit en zeer goed leesbaar scherm;
- de uitgebreide functionaliteit;
- geruisloze werking (geen ventilator);
- waar voor je geld;
- gedetailleerde Franstalige documentatie;
- gemakkelijk beschikbare updates.



Figuur 6. EasyDMM Windows-applicatie.

Ik was minder gecharmeerd van:

- soft aan/uit-schakelaar die alleen naar de slaapstand schakelt;
- de servicehandleiding: het ontbreekt aan duidelijke informatie of bruikbare instructies voor kalibratie – er is een procedure met PHP-scripts, maar die is erg complex. Er zijn instructies om het apparaat te openen en spanningen te controleren, maar geen schema. Bij problemen moet je contact opnemen met Siglent. ◀

230126-03

Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur via info@cyrob.org of naar de redactie van Elektor via redactie@elektor.com.



Over de auteur

Philippe Demerliac, geboren in 1962, is ontwerper van elektronische schakelingen voor wetenschappelijke doeleinden, met een levenslange passie voor fijnmechanica, meetapparatuur, metaalbewerking en wetenschap in het algemeen. Hoewel zijn professionele leven een wending nam in de richting van software, blijft hij toegewijd aan de elektronica. In een poging om iets terug te geven aan de community, creëerde Philippe de cyrob.org-website en, later, een Frans YouTube-kanaal: youtube.com/@Cyrob-org.

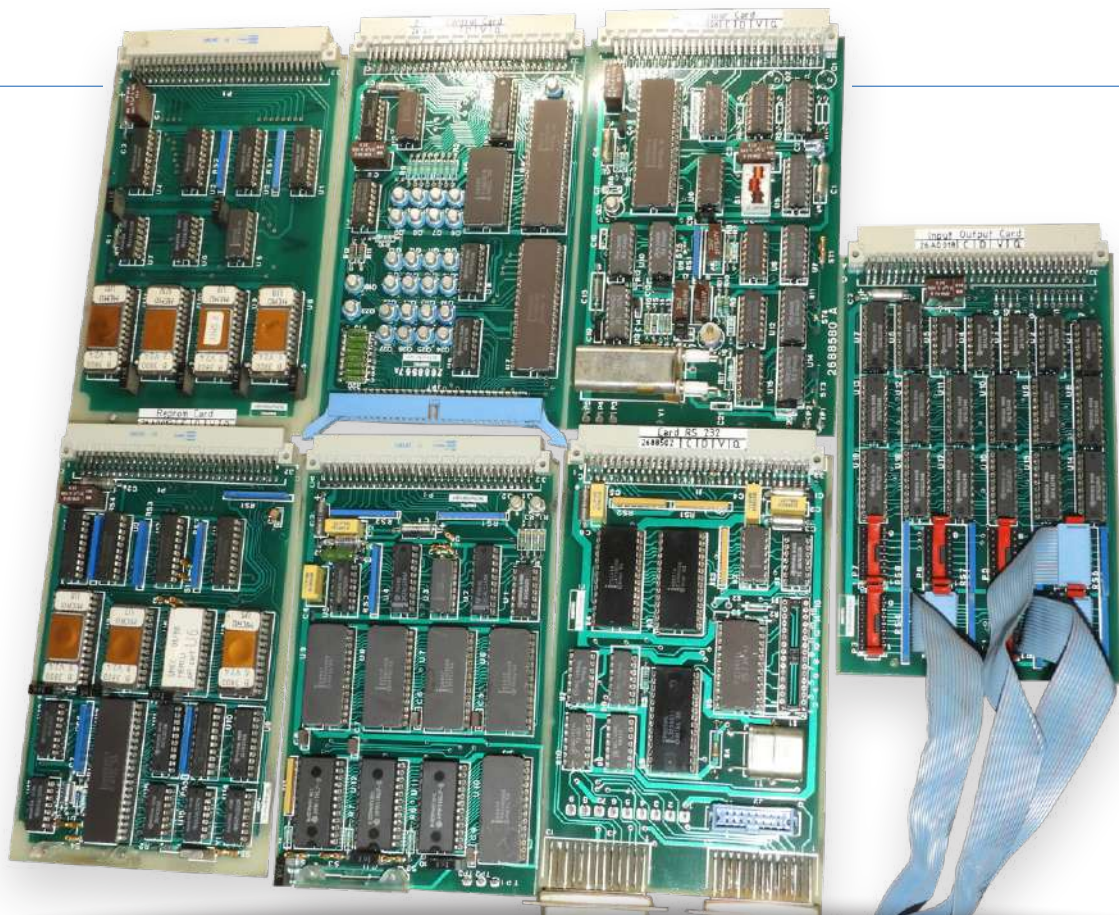


Gerelateerde producten

- Siglent SDM3045X Digital Multimeter
www.elektor.nl/17892

WEBLINKS

[1] Siglent SDM3045X-datasheet: <https://bit.ly/4oiCh00>



Microprocessoren voor embedded systemen

vreemde onderdelen

David Ashton (Australië)

Embedded engineers en hobbyisten hebben tegenwoordig keuze te over als het gaat om rekenkracht, dankzij het enorme aanbod aan microcontrollers. Maar dat was niet altijd zo – laten we ons daarom eens verdiepen in de embedded systemen van het verleden...

Tegenwoordig hebben zelfs low-end microcontrollers zoals de 8-pins AVR ATtiny tot 8 KB flashgeheugen aan boord en tot 512 bytes EEPROM en SRAM. Plus nog de nodige periferie: twee timers/counters, seriële interface, vierkanaals ADC, watchdog-timer en een analoge comparator. De chip voert je code uit met 10 MHz en ondersteunt een breed ingangsspanningsbereik.

Vergeleken daarmee was het leven van vroegere ontwikkelaars van embedded systemen niet gemakkelijk. Ze moesten het stellen met microprocessoren zonder enig on-chip geheugen (afgezien van de registers) of periferie. In plaats daarvan moest al dat lekkers separaat worden toegevoegd om een compleet systeem te bouwen. Dat betekende dat er andere componenten

op de bus van de microprocessor moesten worden aangesloten – heel eigenaardig. Normaal gesproken heb je 16 bit adresruimte, genoeg voor 64 KB geheugen, en 8 bit voor gegevens. Daarnaast zijn er verschillende besturingssignalen, zoals lees/schrijf- en timingsignalen. Alle perifere chips die je hebt aangesloten, hebben meestal één enkele functie. Als geheugen zou je een ROM (voorgeprogrammeerd of EPROM) en RAM aansluiten, en dat laatste moest dynamisch zijn, als je tenminste een fatsoenlijke hoeveelheid wilde hebben. Dit vereiste een duivels gecompliceerde refresh-timing, waarvoor vaak weer een aparte chip nodig was. Periferie werd ook separaat toegevoegd. Timers, counters, in- en uitgangspoorten, UART's voor seriële



Figuur 1. Dit board dateert uit het midden van de jaren '80 en bevat een 8085-microprocessor. Samen met zes andere boards (zie de kopfoto) was het net zo uitgerust als huidige microcontrollers!

communicatie, ADC's, CRT-controllers (die uitgangen hadden om beeldbuizen aan te sturen) – en ga maar door. Elke chip was een 24...40-pins dual inline-monster, meestal op maat gemaakt voor de microprocessoren van dezelfde fabrikant, bij een maximale klokfrequentie van 4 MHz. Maar daar hield de pret nog niet mee op. Vervolgens kwamen de bekende logische IC's voor decodering en buffering, bekend als *glue logic*. Programmeerbare logische chips, zoals GAL's, PAL's en ULA's, werden graag hiervoor ingezet, maar soms stopten fabrikanten die functionaliteit in speciaal op maat gesneden chips. Intel bracht de bal aan het rollen in 1974 met de eerbiedwaardige 8080, waarvan we zouden kunnen zeggen dat deze het werkelijke begin van het microprocessor-tijdperk inluidde. Hij had twee ondersteunende chips nodig, de 8224-klokgenerator en de 8228-buscontroller. Ongebruikelijk voor hedendaagse engineers had hij twee voedingsspanningen nodig: zowel ± 5 V als $+12$ V. Later ontwikkelde exemplaren, zoals de 8085 (figuur 1), had nog slechts een enkele voedingsspanning van $+5$ V nodig. Hij was echter ingewikkelder vanwege het vreemde bus-multiplexschema. Dit vereiste extra glue logic, hoewel er een speciale reeks perifere chips voor de 8085 beschikbaar was. Programma's werden tijdens de ontwikkeling opgeslagen in UV-EPROM's – programmeerbare alleen-lezen geheugens die met ultraviolet gewist konden worden. Dankzij een klein venster boven het halfgeleiderkristal kon de inhoud worden gewist met een EPROM-gum – een kastje met een UV-lamp, een timer en een uitschuifbare lade voor de EPROM. Het UV-licht had de inhoud in ongeveer 20...30 minuten gewist. Daarna kon de chip opnieuw worden geprogrammeerd met behulp van een EPROM-programmer die was aangesloten op de seriële of parallelle poort van je PC. Hoe dit werd gedaan vóór de komst van de PC weet ik niet! Ik herinner me echter dat ik programmeurs met hex-toetsenborden heb gezien die

gegevens in elke geheugenlocatie typten... Programmeren gebeurde in machinecode, waarbij de geheugenlocaties met hexadecimale waarden werden gevuld. De andere optie was assembleertaal die geheugenteuntjes gebruikte voor de bewerkingen en labels voor variabelen en codesecties. Een instructie zoals "ADC B" betekende "voeg de inhoud van register B toe aan register A, met carry." Talen zoals C waren vooralsnog slechts een droom die nog bewaarheid moest worden, maar als je geluk had kon je misschien een BASIC-interpreter te pakken krijgen. Als je programma niet werkte of herschreven moest worden, moest je de EPROM wissen en opnieuw programmeren. Als je eenmaal het definitieve programma had, kon je dat in ROM laten branden. Hoewel zo'n ROM-IC per stuk goedkoper was, kon je die niet meer wissen!

Andere fabrikanten wilden niet achterblijven. Motorola had zijn 6800, MOS Technology de 6502 en National Semiconductor de SC/MP. Enkele voormalige Intel-ontwerpers richtten Zilog op, dat de Z80 aanbood (figuur 2), een soort luxe-versie van de 8080. De meeste van deze chips werden gebruikt in de eerste homecomputers, zoals de Sinclair ZX80/81 en Spectrum (Z80), de Commodore 64 (6502-variant) en vele andere. Het SC/MP-project van Elektor uit 1978 is een ander goed voorbeeld [1].

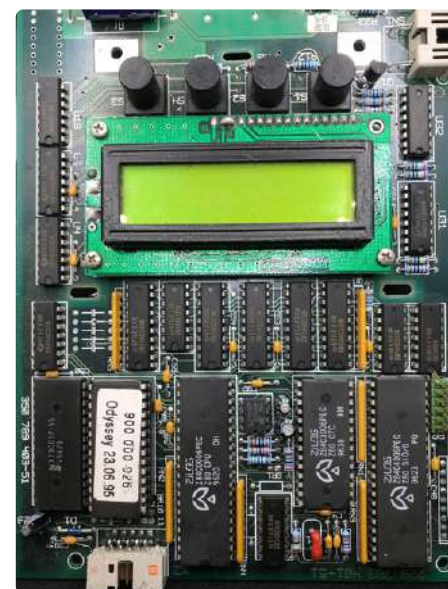
In 1980 introduceerde Intel de 8051. Deze had een kleine hoeveelheid ROM (soms EPROM) en RAM aan boord, maar ook vier 8-bit poorten, een UART en twee teller/timers. Met alles in één apparaat zou je kunnen zeggen dat hier het tijdperk van de microcontrollers is begonnen. En de rest, zoals de dichter zegt, is geschiedenis. De 8051-architectuur wordt nog steeds gebruikt, maar heeft veel meer perifere functies dan zijn voorouders, en maakt een einde aan de bussen en interconnectielogica uit het verleden.

De 8051 en de 68xx-chips van Motorola brachten de rijke en veelzijdige reeks microcontrollers voort die we tegenwoordig kennen en liefhebben: AVR's, PIC's en een groot aantal op Arm gebaseerde typen,

die op hun beurt boards zoals Arduino en Raspberry Pi mogelijk maakten. Intel waagde de gok om de 8085 op te schalen naar de 16-bit 8086, toegepast in de eerste IBM PC XT, en vervolgens de 80286, 80386, 80486 en de Pentiums die nog steeds in PC's worden gebruikt. Opgemerkt moet worden dat op microprocessoren gebaseerde enkelkaartcomputers (SBC's), in plaats van microcontrollers, tot de jaren negentig nog steeds gebruikelijk waren.

Toen de 8080 uitkwam, was ik pas 18 en ik herinner me dat ik erover las in de hobby-elektronictijdschriften van die tijd. Ik had het geluk om aan het begin van mijn carrière aan microprocessor-apparatuur te werken – en dat was best wel opwindend. Mijn werkzaamheden kwamen neer op een beetje programmeren en veel foutzoeken. Ik heb het geluk gehad deze technologie gedurende mijn hele carrière te volgen, en die heeft me altijd gefascineerd. Vergeleken met de huidige microcontrollers waren de oude microprocessoren beslist 'vreemd', maar dat maakte ze niet minder leuk om mee te werken. ◀

230047-03



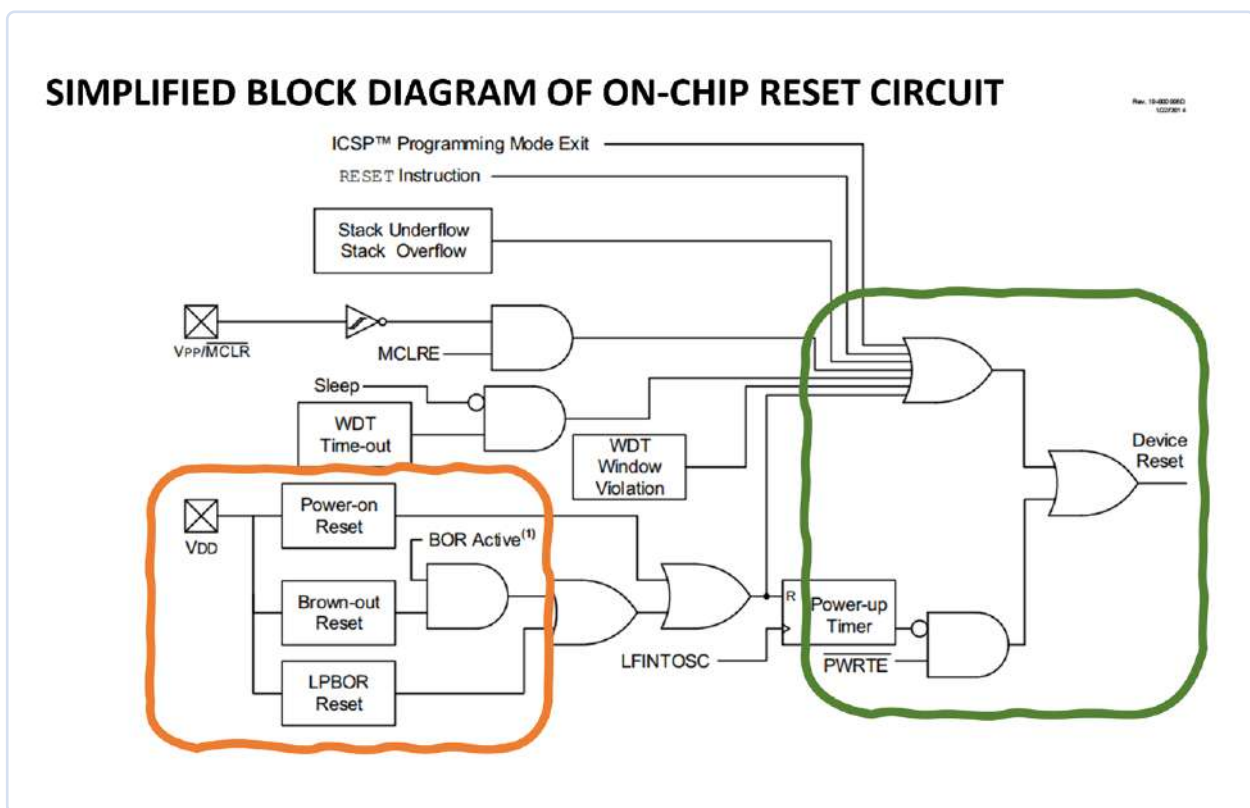
Figuur 2. Een ander werkpaard van vroege embedded systemen was de Z80 van Zilog. Hier gaat hij vergezeld van een seriële I/O-controller (SIO) en een counter-/timerschakeling (CTC).

WEB LINKS

[1] J. Buiting, "De Elektor SC/MP-computer", Elektor Summer Circuits 2022: <http://www.elektormagazine.nl/magazine/elektor-263/60836>

Microcontroller-documentatie verklaard

deel 3: blokschema's en meer



Figuur 1. De resetschakeling biedt zeer weinig configuratiemogelijkheden, en sommige reset-mogelijkheden zijn gekoppeld aan de V_{DD} -voedingspin (bron: Microchip Technology).

Stuart Cording (Elektor)

In de vorige delen van deze serie [1] hebben we de registerfunctionaliteit behandeld en het blokschema van de klok bekeken. We geven hier meer blokschema's en kijken waar we de rest van de benodigde documentatie kunnen vinden.

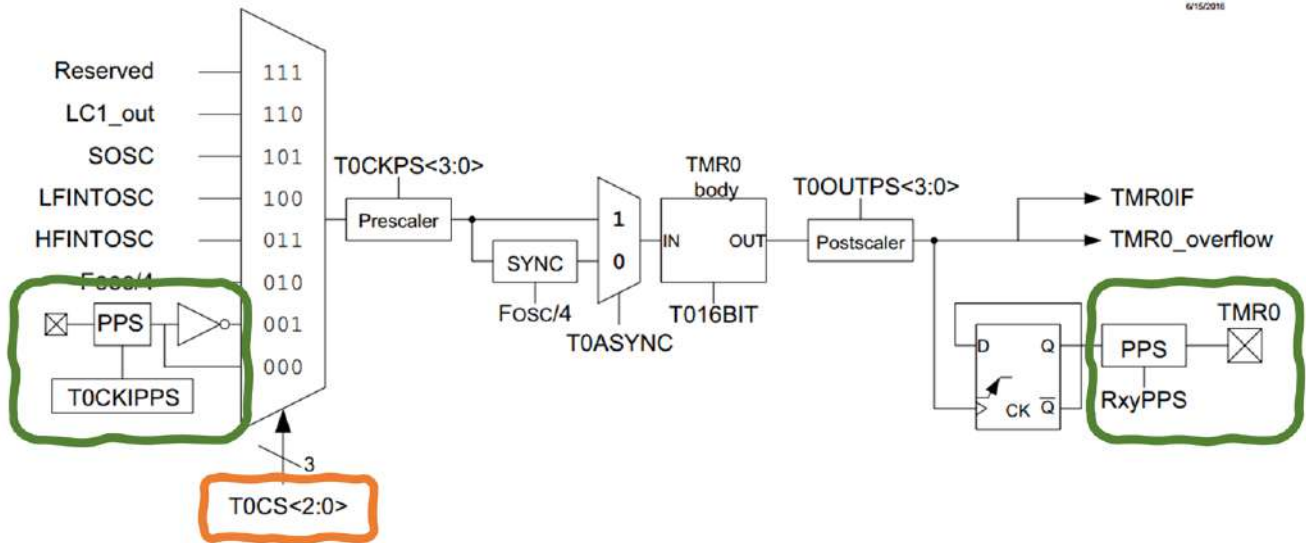
We blijven bij de 8-bit PIC16F18877-microcontroller van Microchip Technology als voorbeeld, en het is raadzaam om de datasheet [2] te downloaden om deze uitleg te volgen.

Herstarten met een reset

Als de klok-periferie het belangrijkste onderdeel van een microcontroller is, dan is de implementatie van de resetschakeling

BLOCK DIAGRAM OF TIMER0

Rev. 10-000017E
6/15/2016



Figuur 2. Het blokschema voor **TIMER0** laat zien dat er een aantal bronnen voor het kloksignaal is, en dat hij zelfs een uitgangssignaal kan genereren naast interne interruptsignalen (bron: Microchip Technology).

het op één na belangrijkste onderdeel. Net als de klok hoeft deze maar één keer geconfigureerd te worden (als er al configuratiemogelijkheden zijn), maar het bespaart later een hoop ongemak als je goed begrijpt waardoor een reset kan worden veroorzaakt. In het voorbeeld op pagina 100 (figuur 1) zien we dat twee pinnen (gemarkeerd als vierkantjes met een kruisje, aan de linkerkant) een reset kunnen genereren. Dit als aanvulling op allerlei andere interne manieren. Eén daarvan is de **RESET**-instructie, twee komen van de stack under- en overflow, terwijl andere gekoppeld zijn aan de watchdog-timer.

We zien een aantal reset-mechanismen (oranje gemarkeerd) die zijn gekoppeld aan de voedingsaansluiting van de microcontroller, zoals de power-on en brown-out detectie. Bij deze controller zijn er hier weinig mogelijkheden om in te grijpen, waarbij in principe alle interruptbronnen potentiële resetsignalen zijn (groen gemarkeerd). Zoals we eerder zagen bij het **STATUS**-register, kan de oorzaak van een vorige reset in sommige gevallen worden achterhaald.

Blokschema's van de periferie

De blokschema's voor de periferie zijn even divers als complex. Hier hebben we een deel van een eenvoudige timer (**TIMER0**, op pagina 396) gekozen (figuur 2). Aangegeven in oranje zien we opnieuw het gebruik van een groep bits in (vermoedelijk één) register om te kiezen tussen een aantal ingangsbronnen voor deze **TIMER0**-periferie. In groen gemarkeerd zien we ook dat één bron een IC-pin kan zijn, terwijl het blok ook een signaal naar een pin kan sturen. Mogelijk herken je de D flip-flop [3] aan de rechterkant die verbonden is met de uitgangspin, maar veel van de blokken in het schema zijn weinig meer dan vierkanten. Een paar daarvan, zoals de pre- en postscaler, spreken voor zich. Het **SYNC**-blok lijkt vanzelfsprekend, maar een ontwerper zal waarschijnlijk de begeleidende tekst nauwkeurig moeten lezen om de functie ervan helemaal te begrijpen. Schema's en tekst gaan dus vaak hand in hand in datasheets. Soms is het nodig wat testcode te schrijven om echt te begrijpen hoe sommige periferie werkt.

Het bouwen van een board

Op een gegeven moment moet het ontwerp op een print worden gezet als je van plan bent je ontwerp op grotere schaal te produceren. De meeste computerondersteunde ontwerpsoftware (CAD) zal de schemasymbolen voor je processor hebben, evenals de footprint ('land pattern') voor verschillende behuizingen voor het ontwerp van de print. Maar als die ontbreken, bevat de datasheet waarschijnlijk technische tekeningen voor elke behuizing en de bijbehorende footprint. In dit geval zijn ze te vinden op pagina 639.

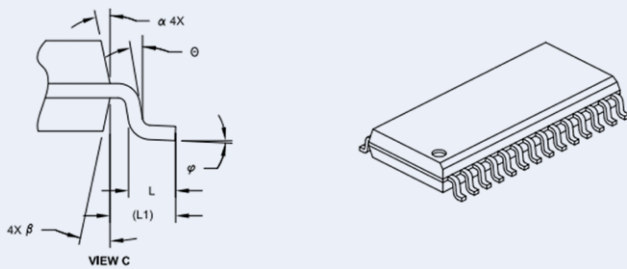
De SOIC-behuizing in figuur 3 is te vinden op pagina's 645 en 646. Voor de ontwerper van de print is de hoogte van de behuizing tegenwoordig misschien wel het belangrijkste gegeven als het ontwerp zo compact mogelijk moet blijven (de breedte en lengte zijn al bekend). Veel behuizingen hebben tegenwoordig ook een 'heat dissipation pad', dat is een apart vlak dat op de print gesoldeerd moet worden om de warmteafvoer te bevorderen. Dat is hier niet het geval, maar als dat wel het geval was, moeten er aanwijzingen te vinden zijn hoe dit gedaan moet worden. Misschien moet je ook controleren of dit pad met de massa van de chip is verbonden, of wellicht met een andere pin.

Wat staat er maar zelden in de datasheet van een microcontroller?

Zoals je inmiddels weet, staat er veel informatie in een datasheet. Er zijn echter een paar dingen die misschien zijn weggelaten de datasheet anders te dik wordt, of omdat de informatie bij veel onderdelen voorkomt en afzonderlijke documentatie verdient. Bijvoorbeeld:

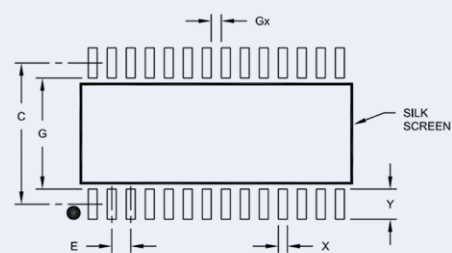
- > de instructieset van de processor: als het aantal instructies beperkt is (minder dan 60...70), staan ze misschien allemaal met uitleg in de datasheet. Zo niet, dan is er waarschijnlijk een afzonderlijk document waarin elke instructie in detail wordt uitgelegd;
- > codevoorbeelden: deze worden vaker afgedrukt als de microcontroller nog gewoon in assembler wordt geprogrammeerd.

28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]



Dimension	Limits	MILLIMETERS		
		MIN	NOM	MAX
Number of Pins	N	28		
Pitch	e	1.27 BSC		
Overall Height	A	-	-	2.65
Molded Package Thickness	A2	2.05	-	-
Standoff	§	0.10	-	0.30
Overall Width	E	10.30 BSC		
Molded Package Width	E1	7.50 BSC		
Overall Length	D	17.90 BSC		
Chamfer (Optional)	h	0.25	-	0.75
Foot Length	L	0.40	-	1.27
Footprint	L1	1.40 REF		
Lead Angle	Ø	0°	-	-
Foot Angle	φ	0°	-	8°
Lead Thickness	c	0.18	-	0.33
Lead Width	b	0.31	-	0.51
Mold Draft Angle Top	α	5°	-	15°
Mold Draft Angle Bottom	β	5°	-	15°

28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]



Dimension	Limits	MILLIMETERS		
		MIN	NOM	MAX
Contact Pitch	E	1.27 BSC		
Contact Pad Spacing	C	9.40		
Contact Pad Width (X28)	X	0.60		
Contact Pad Length (X28)	Y	2.00		
Distance Between Pads	Gx	0.67	-	-
Distance Between Pads	G	7.40	-	-

Figuur 3. De tekeningen van de behuizing geven alle afmetingen (links) samen met het aanbevolen 'land pattern' voor de print (rechts) (bron: Microchip Technology).

Voorbeelden in hogere talen zoals C hebben weinig zin, omdat de compiler de precieze instructies bepaalt die gebruikt worden;

- schemavoorbeelden: deze komen het meest voor met betrekking tot functionaliteit die uniek is voor de betreffende microcontroller, zoals de oscillator of de voeding, of complexere interfaces die specifieke eisen stellen de belasting van signalen, zoals USB. Het is echter waarschijnlijker dat deze in meer detail worden behandeld in een 'application note' over dat onderwerp.

Wat staat er niet in de datasheet van een microcontroller?

De volgende onderwerpen worden gewoonlijk niet in een datasheet vermeld, maar in andere documentatie. Dit is meestal omdat het een onderwerp betreft dat een groot aantal microcontrollers betreft.

- het flashen van het geheugen van de microcontroller: dit wordt normaal gesproken apart behandeld als onderwerp voor degenen die zich richten op het programmeren van microcontrollers voor massaproductie;
- het gebruik van ontwikkeltools: de compiler, het gebruik van een Integrated Development Environment (IDE) en debug-tools hebben hun eigen documentatie;
- een gedetailleerde uitleg van de processorkern: dit wordt vaak in een apart document behandeld, vooral voor 16bit- en 32bit-kernen;
- een gedetailleerde uitleg van complexe periferie: USB, grafische interfaces en Ethernet-randapparatuur worden meestal in aparte documenten behandeld, omdat de omvang van de datasheet van de microcontroller daardoor kan verdubbelen;
- errata: eventuele fouten in de documentatie, of werkwijzen om fouten in de silicium-wafer op te lossen die niet door de fabrikant worden verholpen, worden hier behandeld.

Welke andere microcontroller-documentatie is beschikbaar?

Naast de datasheet worden gewoonlijk de volgende documenten beschikbaar gesteld:

- "Family Reference Manual": terwijl de datasheet je precies vertelt wat één specifieke microcontroller kan, geven dergelijke documenten een overzicht op hoger niveau van wat alle microcontrollers van een bepaalde controllerfamilie kunnen;
- Application notes: deze gaan veel dieper in op het gebruik van specifieke onderdelen, of een groep onderdelen, om een toepassing of interface te realiseren. De datasheet kan uitleggen hoe de CAN-periferie (Controller Area Network) gebruikt moet worden, maar de application note legt in detail uit hoe ze ingezet worden als onderdeel van een CAN-netwerk, met richtlijnen voor softwareprotocollen op hoger niveau en de keuze van geschikte transceivers;
- specificatie van de programmering: voor programmering in massaproductie-omgevingen worden hier in detail de vereiste spanningen en timingdiagrammen gegeven, en het protocol dat gebruikt wordt voor de programmeerinterface.

Hoe kun je een grote hoeveelheid documentatie in je opnemen?

Helaas is er geen manier om snel alle gegevens van een microcontroller en al zijn hulpmiddelen in je op te nemen. Voor beginners is het waarschijnlijk het beste om de datasheet te lezen in combinatie met een boek of artikel over de microcontroller die je gaat gebruiken. Elektor heeft niet alleen boeken voor wie geïnteresseerd is in PIC-microcontrollers; er zijn ook beginnersboeken voor de STM32 [4] en ARM-gebaseerde microcontrollers [5] in het algemeen. Of waarom probeer je niet gewoon een eenvoudig MSP430-project [6]? De combinatie van praktische voorbeelden in combinatie met twee soorten documentatie (een boek/artikel en de datasheet) zal helpen.

Als je al enigszins gevorderd bent, kun je je het beste richten op de hoofdstukken over de processorkern, de klokstructuur en het reset-blok, gevolgd door de periferie die je wilt gebruiken. Daarna moet je je vertrouwd maken met de documentatie van de compiler-toolchain. Maak ook veel gebruik van softwarebibliotheken en voorbeelden om meer begrip te krijgen, en stel vragen op fora als je meer hulp nodig hebt.

Datasheets kunnen lastig zijn om te lezen en te begrijpen, en vormen niet de meest boeiende vorm van literatuur. Ze zijn echter (meestal) nauwkeurig en vormen een technische beschrijving van de functionaliteit (de errata kunnen worden gebruikt om in te schatten in hoeverre je de datasheet kunt vertrouwen). Geef niet te snel op en na verloop van tijd zul je een goed begrip ontwikkelen van hoe datasheets zijn opgebouwd en geformuleerd. ◀

vertaling: Hans Adams — 230286-03

Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur via stuart.cording@elektor.com.



Gerelateerde producten

- **T. Hanna, *Microcontroller Basics with PIC*, Elektor 2020**
www.elektor.nl/19188
- **A. Pratt, *Programming the Finite State Machine*, Elektor 2020**
www.elektor.nl/19327

WEBLINKS

- [1] Microcontroller-documentatie serie: <http://www.elektormagazine.nl/magazine/elektor-295/61525>
- [2] Microchip Technologies PIC16F18877 Datasheet: <https://microchip.com/en-us/product/PIC16F18877>
- [3] De D-flipflop: https://en.wikipedia.org/wiki/Flip-flop_%28electronics%29#D_flip-flop
- [4] Beginnersboeken voor de STM32: <https://elektor.com/programming-with-stm32-nucleo-boards-e-book>
- [5] ARM-gebaseerde microcontrollers: <https://elektor.com/embedded-in-embedded>
- [6] Simpel MSP430-project: <https://elektormagazine.com/labs/elektorpost-no-1-led-earring>



Farnell
AN AVNET COMPANY

SOFTWAREOPLOSSINGEN VOOR TESTEN EN METEN

Wij leveren nu je digitale softwareoplossing bij je hardware.

Kies je T&M software en ontvang deze binnen 72 uur na aankoop per e-mail.





Low-power LoRa-weerstation

bouw zelf een weerstation met een groot bereik

Edward Ringel (USA)

Toen mijn oude draadloze temperatuur- en luchtvochtigheidssensor het na een vruchtbaar leven begaf, verving ik die door een systeem van eigen ontwerp: een buitensensor en een binnendisplay die op batterijen werken en met elkaar verbonden zijn via LoRa. Het ontwikkelen van een betrouwbaar systeem dat bestand is tegen de winter in het noordoosten van de VS was niet eenvoudig! Lees er hier alles over...



Figuur 1. Ondergesneeuwde sensor die bij onze oprit hangt.

De minimaal te ontwikkelen systeemfunctionaliteit was een draadloze, op batterijen werkende buitensensor (**figuur 1**) en een binnendisplay, eveneens voorzien van een sensor. Naarmate het project zich ontwikkelde, voegde ik een heel eenvoudige webserver toe.

Hiervoor wilde ik de Arduino-omgeving met zijn vele hoogwaardige bibliotheken gebruiken. Om het project eenvoudig te houden, wilde ik kant en klare componenten kiezen die handmatig gesoldeerd konden worden. Tenslotte was het minimaliseren van het stroomverbruik een prioriteit. Vrij onverwacht bleek het effectief weerbestendig maken van de buitensensor een verrassende uitdaging.

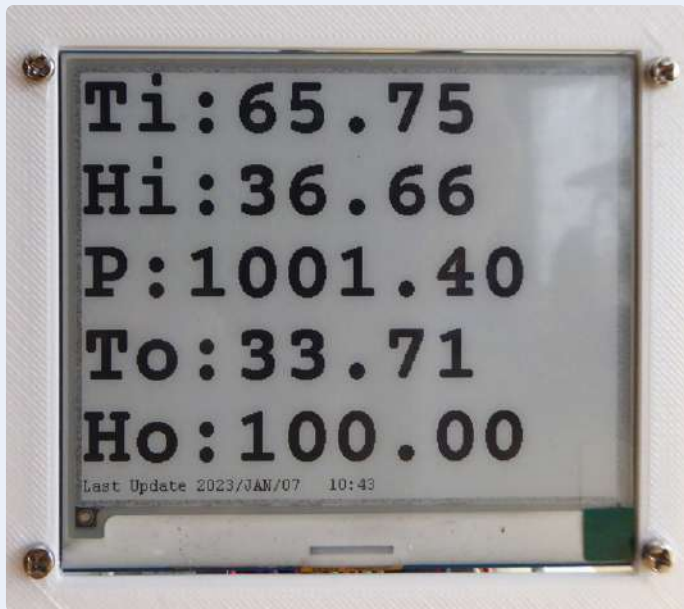
Aan de hand van algemene overwegingen kon ik bepalen welke hardware en boards gebruikt moeten worden – lees er hieronder meer over en over de moeilijkheden die ik tegenkwam. Allereerst zijn een buitenstation, een binnenstation en een basisstation nodig. Dit laatste verzamelt de gegevens van de twee meetstations en geeft ze door aan een computer. Daarom hebben alle

drie minstens een microcontroller-board en een radio-unit nodig om met elkaar te communiceren. De twee meetstations voor buiten en binnen hebben ook sensoren en een timer nodig, waarover later meer. Hieronder volgt een kort overzicht van de centrale bouwstenen van de drie stations:

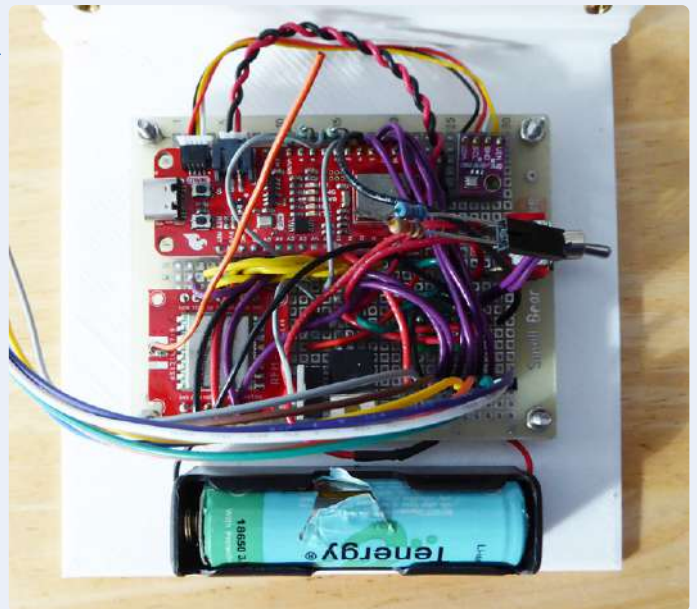
- buitenstation: Artemis Nano, LoRa-radiomodule, timer, sensoren
- binnenstation: Artemis Thing Plus, LoRa-radiomodule, timer, sensoren, display
- basis- of thuisstation: Raspberry Pi Pico, Arduino Mini, LoRa-radiomodule

Over het stroomverbruik gesproken

Metingen van temperatuur, luchtvochtigheid en barometerdruk vinden periodiek plaats. Het stroomverbruik wordt geminimaliseerd door het draadloze meetapparaat in een low-power of sleep-modus te brengen, te laten ontwaken, metingen te verrichten en te verzenden, en terug te laten keren naar de low-power modus. De duty cycle van het stroomverbruik wordt



Figuur 2. Mijn ePaper display toont de temperatuur (in °F), derelatieve luchtvochtigheid en de ongecorrigeerde/niet genormaliseerde barometrische druk. De aanduidingen "i" en "o" verwijzen naar binnen en buiten. Alleen de luchtdruk buiten wordt getoond.



Figuur 3. Een vroege versie van de display-print en de displayscheakeling, vóór de aanpassing van het Artemis-board en de toevoeging van de TPL5111. Optocouplers, MOSFET's, TPL5110, schakelaars voor het opladen – jakkes!

weergegeven door T_{ON}/T_{TOT} . Het effectieve stroomverbruik wordt bepaald door:

- het stroomverbruik van het station in bedrijf terwijl het instructies of gegevens verwerkt;
- de periode dat het station in bedrijf is om gegevens te verzamelen en te verzenden (of weer te geven) – het minimaliseren van de duty cycle beïnvloedt het totale stroomverbruik aanzienlijk;
- het stroomverbruik van het station tussen de metingen door. Ideaal zou het zijn als er dat geen stroom wordt verbruikt, maar dat is geen realistische verwachting.

Ik heb het stroomverbruik van meerdere MCU-boards getest door het board op te starten zonder programma's te draaien of periferie aan te sluiten. Zie **tabel 1** voor de resultaten (deze geven mogelijk niet het stroomverbruik weer na fijnregeling zoals het uitschakelen van periferie, het vertragen van de klok enzovoort). Op grond van deze bevindingen besloot ik de Artemis-boards van Sparkfun te gebruiken.

Het stroomverbruik was ook een factor bij het ontwerpen van het displaystation voor binnen, maar de keuze voor een display was eenvoudig: ePaper heeft het laagste stroomverbruik. Deze technologie verbruikt alleen stroom als het scherm wordt ververs

(daarom kan de met ePaper uitgeruste Kindle een afbeelding weergeven terwijl hij niet actief is en geen stroom verbruikt). WaveShare heeft een mooi 4,2" ePaper-display. De instructies waren moeilijk te volgen, en ik moest wat lettertypes genereren, maar zoals **figuur 2** bewijst werkte het display uiteindelijk prima.

Voor de metingen heb ik Bosch BME280-sensoren gebruikt. Deze modules meten temperatuur, luchtvochtigheid en barometrische druk (T/H/P), en er zijn robuuste I²C-bibliotheken voor. Het stroomverbruik is te verwaarlozen.

Een timermechanisme regelde de duty cycle. De MCU, radio en sensor moesten worden in- en uitgeschakeld – het zou onzin zijn om een efficiënte MCU te gebruiken die in de sleep-modus 20 μ A verbruikt terwijl de radio 20 mA trekt. In plaats van de sleep-modus van de MCU en de radio te coderen, schakelde ik het complete station in en uit.

De 3,3V-regelaar op het Artemis-board levert een maximale stroom van 600 mA, genoeg om alle componenten van het binnendisplay en van de buitensensor/radio van stroom te voorzien. Omdat ik de stroom naar de boards moest kunnen uitschakelen, was de beste oplossing een ultra-low power timer-chip. Deze IC's hebben een intern timercircuit, waarvan het interval wordt ingesteld met een enkele externe weerstand,

en worden gebruikt met een MCU. Als de timer inschakelt ('aan'), wordt de stroom ingeschakeld. Als de MCU zijn taak heeft voltooid, stuurt hij een DONE-bericht naar de timer, en wordt de stroomtoevoer uitgeschakeld. Ik heb uitgebreid geëxperimenteerd met de TPL5110-chip op een zelfgemaakt board en op het Adafruit-breakout-board, beide in combinatie met een eerdere versie van het displayboard (**figuur 3**). Mijn mislukte avonturen met dit kleine onderdeel zijn gedocumenteerd in [1].

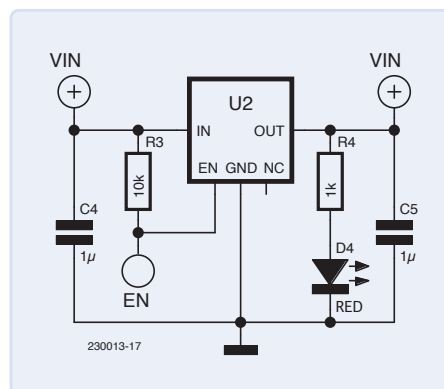
Tabel 1. Stroomverbruik van de boards.

Board	Stroomverbruik (mA)
Teensy 3.5	74,5
Raspberry Pi Pico	20,4
Teensy 4.1	92,6
ESP32 Adafruit Huzzah Feather	125,0
Artemis Thing Plus	8,5
Generic ESP32-WROOM	70,3

Het stroomverbruik van verschillende operationele MCU's, met 5 V op V_{in} , maar zonder draaiende programma's. Het stroomverbruik wordt niet alleen bepaald door de MCU, maar ook door andere componenten op het board (LED's, spanningsregelaars, extern geheugen, draadloze configuratie enzovoort).



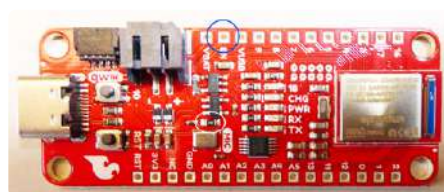
Ik heb uiteindelijk een TPL5111 gebruikt die is aangesloten op de enable-pin van de 3,3V-regelaar op de Artemis-boards (**figuur 4**). Hierbij was het noodzakelijk om de SMD pull-up weerstand R1 op de Artemis Nano (**figuur 5**) te verwijderen, of R3 op de Artemis Thing Plus (**figuur 6**); dit stelde me in staat de gehele stroom-



Figuur 4. Artemis-spanningsregelaar. De ENABLE-pin wordt hoog getrokken door R3 (R1 op de Artemis Nano). Deze weerstand moet worden verwijderd. De ENABLE-pin is voor de spanningsregelaar van het board, niet voor de MCU.



Figuur 5. Het Artemis Nano-board. R1 is wit omcirkeld. In tegenstelling tot de Thing Plus is er geen EN-connector maar een 'PSWC'-connector. TPL5111 ENABLE moet worden aangesloten op de groen omcirkelde connector nadat R1 is verwijderd. Let op: R1 zit aan de andere kant van de print, en moet worden verwijderd voordat deze wordt gemonteerd.



Figuur 6. Het Artemis Thing Plus-board. R3 is wit omcirkeld. TPL5111 ENABLE kan rechtstreeks worden aangesloten op de EN-pin (blauw omcirkeld) nadat de weerstand is verwijderd.

voorziening op het board te regelen. Deze weerstanden maken de enable-pin van de spanningsregelaar 'hoog', tenzij de pin aan massa ligt. Uiteindelijk bleken de aangepaste boards een handige oplossing, waarbij de hele schakeling slechts 20 μ A trok in de sleep-modus.

Zowel de TPL5110 als de TPL5111 zijn interessante chips die het bestuderen waard zijn, vooral bij draadloze IoT-sensorapplicaties. De Adafruit-boards zijn geweldig en bieden uitstekende ontwerpflexibiliteit.

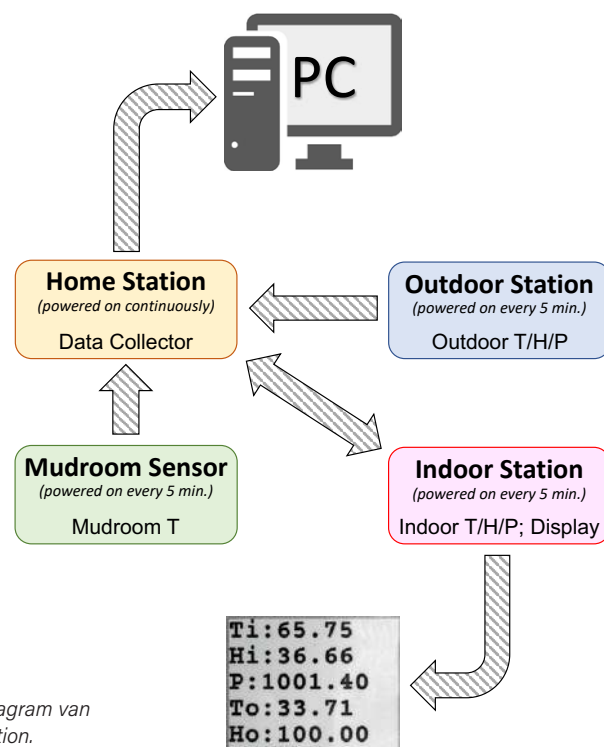
Datatransmissie

Ik koos voor HopeRF LoRa-boards op 915 MHz voor de datatransmissie (andere landen kunnen een afwijkende frequentie voorschrijven). Er zijn meerdere bibliotheken beschikbaar voor de Arduino-omgeving die de implementatie vereenvoudigen. De sub-gigahertz draaggolffrequentie maakt het mogelijk beter door muren en gebouwen heen te dringen en grotere afstanden te overbruggen. De transmissieprotocollen zijn geoptimaliseerd voor kleine hoeveelheden data. De protocollen

vereisen geen tijdrovende handshakes en beveiligingsoverhead. Dit legt echter de last van betrouwbare overdracht van gegevens bij de programmeur in plaats van bij de ingebouwde transmissieprotocollen.

Nu kun je je afvragen hoe de sensor- en displaystations met elkaar communiceren. De T_{ON} -tijden van beide stations overlappen elkaar immers maar zelden. Als de sensor ontwaakt en gegevens verstuurt, is het displaystation waarschijnlijk in slaap verzonken. De oplossing was een derde, constant ingeschakeld station op basis van een Raspberry Pi Pico, dat geen sensoren bevat maar functioneert als data-concentrator en doorgeefluik en dat gekoppeld is aan een computer. Ik zal dit het thuisstation noemen. De enige externe onderdelen zijn de radio en een paar LED's voor het debuggen. Zijn spanningsregelaar kan een paar honderd milliampère stroom leveren, en ik heb de radio daarmee gevoed. De Pico wordt gevoed via de USB-poort van een PC en communiceert serieel met een Processing-programma. Slechts een paar honderd regels code zijn voldoende om

Data Flow Diagram



Figuur 7. Data-flowdiagram van het weerstation.



Figuur 8. Sensorprint ingegoten in epoxy. Let op het plastic vierkantje in de linkerbovenhoek dat voorkomt dat de sensor door de epoxy wordt bedekt. Let ook op de haakse USB-C adapter (midden links) die de print toegankelijk maakt voor opladen en software-updates.



Figuur 9. Close-up van de sensorbehuizing. Alle openingen zijn met gaas afgeschermd om ongewenste gasten buiten de deur te houden.

de barometrische druk te plotten en een tabel met metingen op een webpagina te zetten. **Figuur 7** illustreert de gegevensstroom als volgt:

1. De buitensensor ontwaakt elke vijf minuten en stuurt temperatuur-, luchtvochtigheids- en drukwaarden (T/H/P) naar het thuisstation. Er is geen communicatie van het thuisstation terug naar de buitensensor. De buitensensor schakelt na verzenden uit zonder ontvangstbevestiging door het thuisstation.
2. Het thuisstation slaat de meest recente metingen van de buitensensor op in het RAM-geheugen. De buitenmetingen worden ook naar de computer gestuurd voor verdere verwerking.
3. De binnensensor en de display-unit ontwaken elke vijf minuten. Het meet de T/H/P-waarden binnenshuis, stuurt die informatie naar het thuisstation, en bewaart een kopie van de metingen in het RAM-geheugen.
4. Als het thuisstation een update-verzoek ontvangt van de binnensensor en -display, stuurt het deze informatie naar de computer. Vervolgens stuurt het de meest recente buitenwaarnemingen naar het binnenstation. Daarna actualiseert het binnenstation het display en schakelt het uit.

In de loop der tijd merkte ik dat het thuisstation herhaaldelijk vastliep, met als gevolg

dat het display en het Processing-programma niet werden bijgewerkt. Berichten op diverse ondersteunende fora suggereerden dat de LoRa-transceiverchip SX1276 kon vastlopen en dit probleem kon veroorzaken, waarbij zowel de radio als de MCU vastliepen als ze langere tijd achtereen aan stonden. Dit probleem werd nooit vastgesteld bij de display- of sensorstations, omdat die modules vaak worden in- en uitgeschakeld. De oplossing voor de Pico was het gebruik van een tweede MCU, een Arduino Mini, als slimme timer. De uitgang van de Mini werd verbonden met de enable-pin op de Pico, waardoor de spanningsregelaar elk uur gedurende 500 ms werd uitgeschakeld. Eén uitgang van de Pico werd verbonden met de Mini om aan te geven wanneer de datastroom het minst onderhevig zou zijn aan de gevolgen van een reset: de harde reset zou pas plaatsvinden nadat er een gegevensupdate was geweest en terwijl de Pico geen nieuwe informatie ontving.

Hoewel dit een overdreven toepassing van een MCU lijkt, zijn de onderdelen zo goedkoop dat het gerechtvaardigd is. Ik moedigde de Elektor-community aan om mee te denken over hoe dit probleem eleganter opgelost kan worden. Er zijn twee versies van de software voor het thuisstation [2], één voor gebruik met het Processing-programma, en één die onafhankelijk is van een computeraansluiting en alleen een voeding nodig heeft, zoals een oude telefoonlader

met een passende USB-aansluiting. De Processing-code is nog in ontwikkeling en mag niet worden beschouwd als een afgeronde en foutloze toepassing.

Weerbestendig maken

Eerder viel mijn buitensensor sporadisch uit bij snelle temperatuur- en luchtvochtigheidsveranderingen; ik vermoedde condensvocht als oorzaak. Nadat ik al mijn soldeerverbindingen en bedrading had onderzocht en geen fouten had gevonden, besloot ik de buitenprint in te gieten. Ik printte een behuizing in 3D voor de print en een afscherming rond de sensor zodat die niet ingegoten zou worden (**figuur 8**). Ik sloot een haakse USB-C adapter aan, waardoor de USB-C connector van het Artemis-board toegankelijk bleef.

Ik gebruikte goedkope transparante hobby-epoxy, verkrijgbaar bij Amazon. Hobby-epoxy is zo samengesteld dat het heel langzaam uithardt en de gebruiker een langere periode heeft om het materiaal te bewerken. Omdat ik niet streefde naar een perfect esthetisch resultaat en niet wilde dat mijn katten onder de epoxy kwamen te zitten, zette ik de bak op een elektrische warmhoudplaatje bij ongeveer 65...66 °C. De epoxy was binnen een uur kattenbestendig. Daarna liet ik het geheel een nacht lang bij kamertemperatuur (18...20°C) uitharden. De print zit in een wit geschilderde kunststof pot (**figuur 9**) en de ventilatie-openingen



Onderdelenlijst

Weerstanden:

R1 = zie tekst

R2...R4 = 1 M, 1/4 W

R5 = 680 Ω, 1/4 W

R6...R8 = 360 Ω, 1/4 W

Halfgeleiders:

D1 = LED blauw

D2 = LED rood

D3 = LED geel

D4 = LED groen

Boards:

2× TPL5111 breakout-board (Adafruit Industries)

1× Raspberry Pi Pico

1× Artemis Thing Plus (Sparkfun)

1× Artemis Nano (Sparkfun)

2× BME280 breakout-board (ASIN B08DHTGNHR *)

1× FTDI basic breakout 3,3 V (Sparkfun)

3× HopeRF RFM95CW 915 MHz FCC Certified LoRa Transceiver (Anarduino.com)

1× Arduino Pro Mini, 3,3 V 8 MHz (zie tekst)

Diversen:

2× ST 4-polige connector (ASIN B01DUC1M2O *; zie Over de bouw)

1 4,2" ePaper-display (WaveShare; ASIN B074NR1SW2 *)

3× dubbelzijdige FR4-gaatjesprint 7x9 cm (ASIN B08F7X8JHV *)

2× 18560 LiPo-accu

2× batterijhouder voor 18650 -accu

Epoxyhars-kit (ASIN B07TVWTG829 *)

USB-C haakse adapter (ASIN B0BBVWF54L *)

Headers en bedrading waar nodig

* ASIN-codes verwijzen naar zoekbare Amazon-voorraanummers.

zijn afgeschermd met gaas om de sensor tegen insecten te beschermen.

Voordat ik de schakeling ingoot, heb ik getest dat de epoxy geen kortsluiting tussen de blanke draden veroorzaakt of elektrische verbindingen stoort. Sinds ik de print heb ingegoten, hebben zich geen sporadische storingen meer voorgedaan.

Software en data

Alle drie stations gebruiken in hun software LoRa-bibliotheken en WaveShare-code. Voor het binnen- en buitenstation zijn bibliotheken voor de BME280-sensorchip nodig. Je kunt de complete pakketten gratis downloaden van [2]. De moeite van het coderen voor de TPL5111 is onbeduidend. Ik heb eenvoudige codering en een kortere T_{ON} ingeruild voor gegarandeerde gegevensoverdracht. Na de meting wordt een korte string samengesteld die de meetresultaten en de identificatie van het zendstation

bevat; deze string wordt tweemaal verzonden (zie verderop). De transmissies van het thuisstation naar het displaystation bevatten ook tijdgegevens, zodat het display de tijd van de laatste update weergeeft.

De code lijkt wellicht onvolledig. De uitvoering van de code stopt echter als de spanningsregelaar wordt uitgeschakeld. Hiertoe stuurt de MCU een *DONE*-signaal naar de timer. Hierdoor worden eventuele hiaten in de software gedicht. Behalve de WaveShare-code heb ik geen bibliotheken opgenomen in de software-download. Deze kunnen eenvoudig door de Arduino IDE aan je omgeving worden toegevoegd. Ik heb de Sparkfun BME280-bibliotheek gebruikt omdat er geen andere code-afhankelijkheden zijn dan *Wire.h*. De Sandeep Mistry LoRa-bibliotheek bood alle noodzakelijke functionaliteit en had geen andere code-afhankelijkheden dan *SPI.h*.

Omdat het thuisstation de transmissies

tussen de sensoren en het displaystation niet kan coördineren, zijn gelijktijdige transmissies met als gevolg dataverlies onvermijdelijk. De datastring wordt tweemaal verzonden met daartussen een vertraging om het risico van dataverlies te verkleinen. Als er uiteindelijk een paar datapunten per dag verloren gaan, is dat acceptabel voor een niet-kritische toepassing.

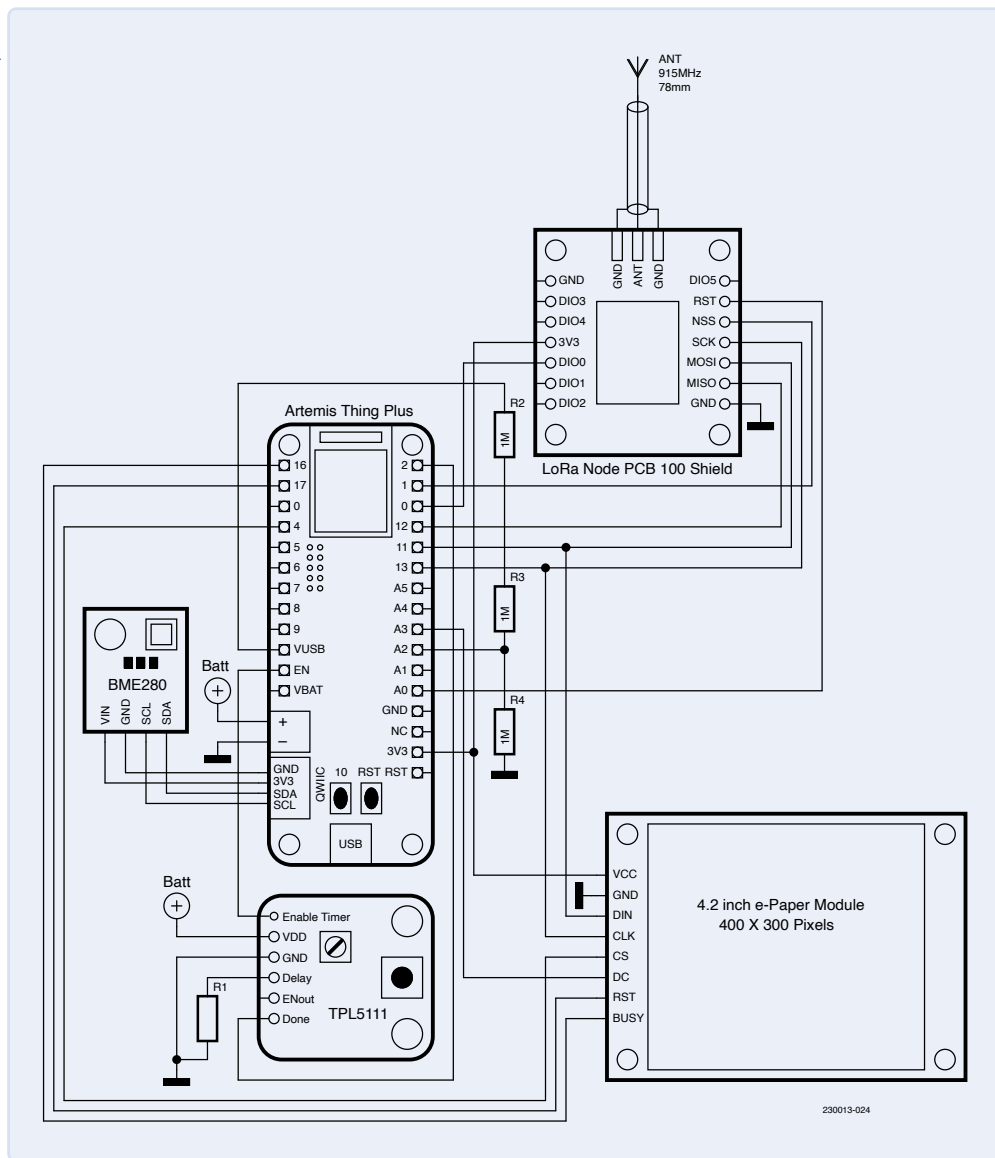
Tijdens de ontwikkeling realiseerde ik me dat het thuisstation zou kunnen communiceren met een computer en de meetresultaten naar een database, een webpagina of iets dergelijks zou kunnen sturen. Dit is nog in ontwikkeling. Ik heb een ruwe webserver geschreven in Processing, een van Java afgeleide taal met een uitgebreide gebruikersbasis en veel goede bibliotheken. Zoals gezegd wordt de Processing-code meegeleverd, evenals de Arduino-sketches. Hoewel er enkele oppervlakkige overeenkomsten kunnen zijn, is dit geen LoRaWan-project, en is het thuisstation geen LoRa-gateway. Mijn configuratie registreert de omstandigheden buiten, in mijn keuken (waar het display staat), en een derde sensor in onze bijkeuken, waar het 's winters vriest. De data van de sensor in de bijkeuken verschijnen alleen op de webpagina. Er zijn bovengrenzen aan het aantal sensoren en displays dat in deze opstelling wordt ondersteund. Naarmate het netwerkverkeer toeneemt, zal dataverlies door gelijktijdige transmissies uiteindelijk onaanvaardbaar worden. Hierop hebben veel factoren invloed, waaronder de nauwkeurigheid van de timingweerstand voor de TPL5111, omgevingsfactoren en, het belangrijkste, de frequentie van de sampling- en/of update-requests. Er is geen bovengrens aan het aantal displays die geen data opvragen. Extra stations die gegevens opvragen dragen bij aan het radioverkeer. Het totale radioverkeer, en niet de transmissies van sensormetingen, beperkt het netwerk. Het berekenen van de theoretische grenzen van verschillende configuraties is een aardige oefening voor de lezer.

Praktische opmerkingen

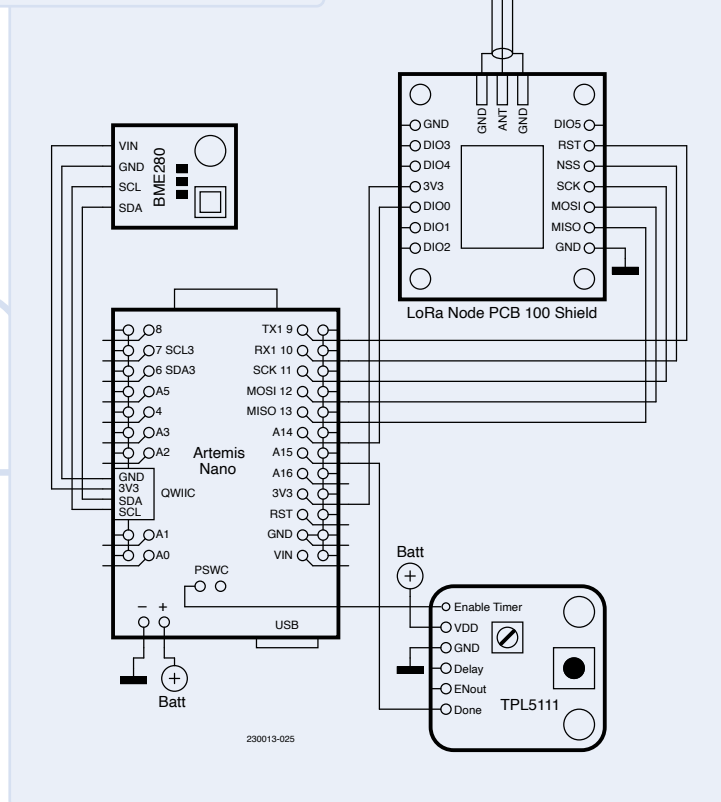
De hardware is met de hand bedraad. Lees het kader **Over de bouw** zorgvuldig. De sensoren communiceren via I²C. Ik heb het Sparkfun Qwiic-systeem gebruikt, dat erg handig is. Het is eenvoudig om

Over de bouw

1. **Verwijderen van R1 en R3:** zie de figuren om de van Nano en Thing te verwijderen weerstanden R1 en R3 te lokaliseren. Anders werken de schakelingen niet.
2. **ENABLE-aansluiting op Artemis Nano:** zie figuur 5 voor de ENABLE-aansluiting op de Nano. Ik heb een foto gemaakt van de onderkant van de printplaat. Gebruik de massa-pin NIET!
3. **Hoofdprint:** elke soort gaatjesprint kan worden gebruikt, maar de in de onderdelenlijst genoemde printen zijn voorvertind, FR4, zijn groot genoeg en bestand tegen herhaald solderen.
4. **Radiofrequentie:** koop een radio-module die in jouw land gebruikt mag worden.
5. **Aansluiten op het Qwiic-systeem:** de genoemde JST-connector past perfect in het QWIIC-systeem, maar de kleuren komen NIET overeen met de Sparkfun-conventie. Voor deze connectoren geldt: wit = GND, geel = 3,3 V, zwart = SDA en rood = SCL.
6. **BME280-boards:** pas op bij het kopen van BME-boards. Sommige Amazon-verkopers vervangen (al of niet bewust) de BMP280- door BME280-chips. De BMP-versie is goedkoper en ziet er bijna hetzelfde uit, maar de BME-bibliotheken werken niet en je krijgt geen luchtvochtigheidsdata.
7. **Over de WaveShare-verbinding gesproken:** de WaveShare-unit wordt geleverd met een gebruiksvriendelijke kabel. Paars = BUSY, wit = RESET, groen = DC, oranje = CS, geel = CLK, blauw = DIN (MOSI), bruin = GND, grijs = Vcc. De unit heeft geen MISO aansluiting.
8. **Tijdsinterval van de TPL5111:** het tijdsinterval van de TPL5111 wordt geprogrammeerd met behulp van een on-board potmeter of een externe weerstand. De Adafruit-website en de datasheet van het IC geven een tabel met weerstandswaarden voor verschillende tijdsintervallen. Als je een externe weerstand gebruikt, moet je een spoor op de achterkant van de print doorsnijden. Als het stroomverbruik een probleem is, kan de activiteits-LED ook worden uitgeschakeld door een spoor door te snijden.
9. **Aansluiting van de accu op Artemis-boards:** de accu wordt op het board aangesloten met een goed verkrijgbare twee-aderige JST-kabel.
10. **Radio breakout-board:** ik gebruikte een breakout-board voor de radio van Diycon.nl (LoRa Node PCB 100 Shield Only voor RFM92/RFM95). Daar kan gemakkelijk een draadantenne of een antenneconnector aan worden bevestigd.
11. **Radio-antenne:** ik heb een eenvoudige kwart-lambda draadantenne gebruikt. De juiste draadlengte voor 915 MHz is 78 mm, gesoldeerd aan de middelste connector op het radio breakout-board.
12. **Epoxy:** een heel geklieder. Handschoenen, wegwerpbakjes om te meten en te mengen, wegwerpmengstokjes en een doek om druppels op te vangen zijn onmisbaar. Zorg ervoor dat het board perfect werkt voordat je verder gaat. De haakse USB-C adapter moet geplaatst zijn, en de timer moet de juiste weerstand hebben. De ondiepe behuizing moet iets groter zijn dan het board en de accu. Dompel de BME280 niet onder in epoxy. Ik heb een afscheiding rond de sensor aangebracht (de epoxy komt door de perforaties naar boven, dus dek ook de onderzijde af). Of sluit de sensor via draden aan op de QWIIC-connector en houd de sensorprint boven de epoxy terwijl die uithardt. Zachtjes verwarmen van de epoxy versnelt het uitharden aanzienlijk. De drie belangrijkste punten zijn dat de sensor niet is bedekt met epoxy, dat de USB-C connector toegankelijk blijft via de adapter, en dat met uitzondering van de sensorprint alle onderdelen op de print zijn bedekt.
13. **Behuizing:** er zijn verschillende aandachtspunten bij het ontwerp ervan. Ventilatie-openingen moeten zodanig worden aangebracht en beschermd dat water alleen 'tegen de zwaartekracht in' de behuizing kan binnendringen. Ten tweede moet de behuizing reflecterend zijn of tenminste wit geschilderd om opwarming van het inwendige te minimaliseren. Ten derde moet de behuizing licht zijn. De thermische traagheid van een grote, zware behuizing maakt je metingen traag en onnauwkeurig. Tenslotte moet de behuizing op de een of andere manier tegen insecten beschermd zijn.
14. **18560-accu's:** bij Amazon en eBay zijn er in overvloed accu's met een onwaarschijnlijke energiedichtheid tegen een absurd lage prijs te vinden. Wees hier gezond wantrouwig. Merkcellen met 2500...3500 mAh zijn de veiligste keuze!



Figuur 10. Schema van het displaystation.



Figuur 11. Schema van de buitensensor.





een Qwiic-compatibele connector aan te sluiten op de kleine BME280-boards die verkrijgbaar zijn op Amazon en eBay. Let erop dat je een BME-versie koopt in plaats van een BMP280 als je de luchtvochtigheid wilt meten. De HopeRF-module heeft een raster van 2 mm in plaats van de 'normale' 2,54 mm, wat een breakout-board of zorgvuldig soldeerwerk nodig maakt. Ik gebruik een eenvoudige kwart-lambda draadantenne.

Ik gebruikte FontEdit om een 36-punt lettertype te maken voor het ePaper-display. De instructies van WaveShare zijn moeilijk te begrijpen, maar uiteindelijk ontwikkelde ik werkende code. Als je een andere MCU dan de Artemis gebruikt, denk er dan aan dat de fonttabellen en de bitmap-buffer veel RAM-geheugen in beslag nemen.

Ik gebruikte een oplaadbare 18560 lithium-accu voor de stroomvoorziening. Zie **Over de bouw** voor opmerkingen over de verkrijgbaarheid van de 18560. Andere voedingsopties zijn mogelijk, maar alkaline-batterijen worden nadrukkelijk niet aanbevolen voor temperaturen onder nul (0 °F, -18 °C).

De schema's (figuren 10...12) dienen als suggestie. De populaire nieuwe 32bit-boards zijn flexibel, met veel aansluitingen die interrupts ondersteunen en met alternatieve SPI-, I²C- en UART-interfaces. In dit verband zijn sommige verbindingen gericht op een optimale fysieke layout van het geheel.

Veel plezier! ◀

vertaling: Hans Adams — 230013-03

Over de auteur

Ed Ringel is een halfgepensioneerde long- en intensive care-arts. Hij geniet met zijn vrouw van het landleven in Maine, schrijft science fiction, maakt dingen met zijn 3D-printer en ontwikkelt elektronica-projecten met MCU's.

Figuur 12. Schema van het thuisstation.

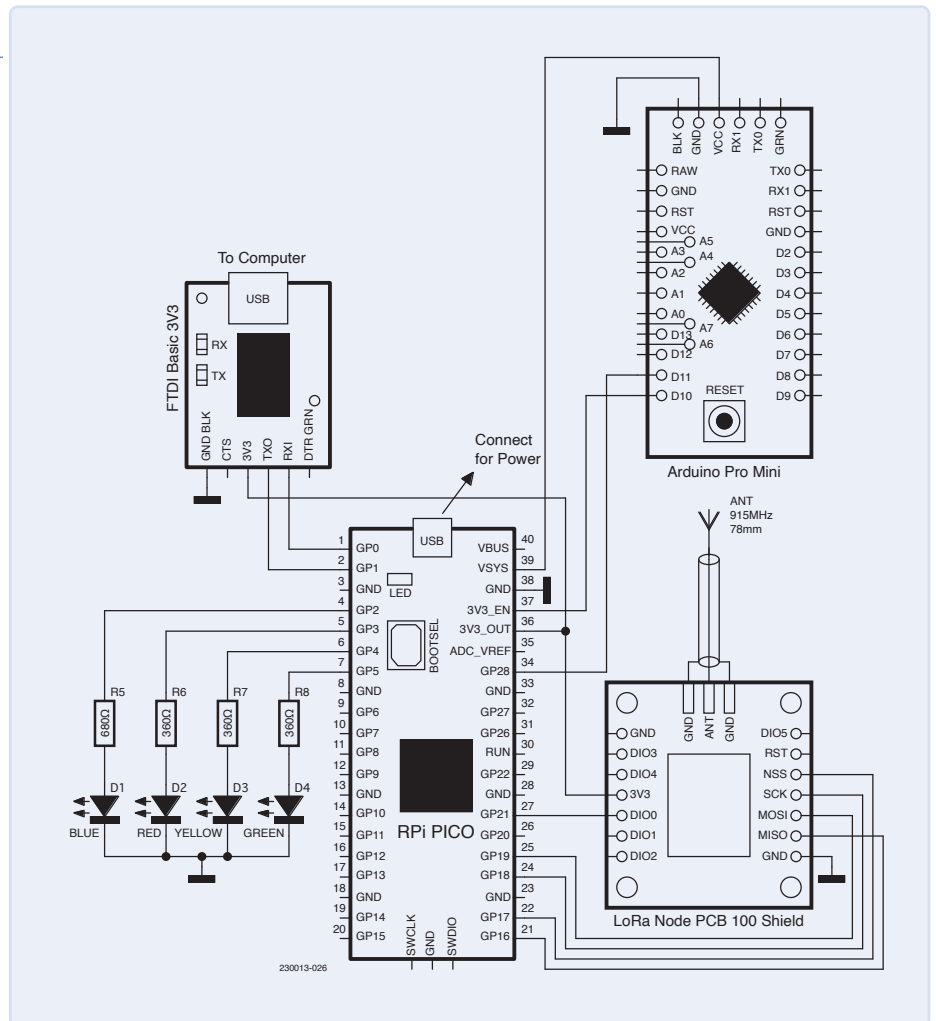
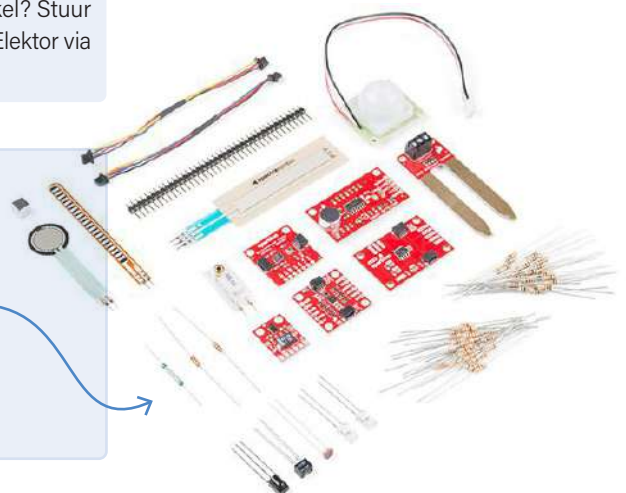
Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de redactie van Elektor via redactie@elektor.com.



Gerelateerde producten

- **SparkFun Sensor Kit**
www.elektor.nl/19620
- **Raspberry Pi Pico RP2040**
www.elektor.nl/19562



WEBLINKS

- [1] Avontuurlijke mislukkingen: <https://forums.adafruit.com/viewtopic.php?p=927526>
- [2] Dit project op Elektor Labs: <https://elektormagazine.com/labs/low-power-lora-weather-station>

Transverter voor de 70cm-band

Jan Buiting PE1CSI (Elektor)

In 1981 verblijdde Elektuur de radioamateur-gemeenschap met een prachtig ontworpen zend/ontvang-converter (transverter) voor de 70cm-band, toen nog met een weinig gebruikt deel van het radiospectrum dat door echte experimenteerlustigen werd gebruikt om via de ether te communiceren zonder mobiele telefoons – kun je je dat voorstellen?

De 70cm-transverter (430...440 MHz) waaraan ik hier herinneringen ophaal, is in twee delen gepubliceerd in Elektuur mei en juni 1981 [1][2]. Het is een mooi voorbeeld van een publicatie voor op radioamateurs die niet bereid waren de (in die tijd) exorbitante prijzen van commerciële apparatuur te betalen. Zij wilden gewoon SSB (single side band, enkelzijband) op 70 cm, zoals ze dat al vele jaren op de kortegolf en op 2 m (144...146 MHz) konden genieten. In tegenstelling tot FM is SSB een lineaire modus die een perfecte lineariteit vereist van alle zender-trappen tot en met de antenne-aansluiting.

Pas op: zendamateurs aan het werk

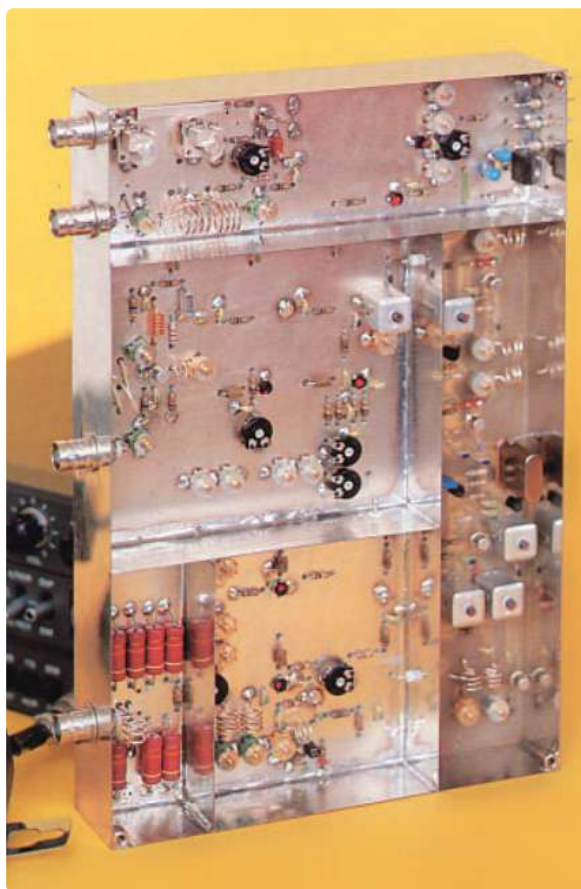
Het oorspronkelijke ontwerp voor de transverter is van de hand van van J. de Winter PE0PJW. Bij Elektor Labs hebben redacteur Gerrit Dam PA0HKD en stagiair Ed Warnier (PE1CJP, nu PA1AW) het ontwerp zodanig aangepast en ge-elekturiseerd dat het repro-

duceerbaar was voor Elektuur-lezers en voldeed aan de wettelijke eisen wat betreft ongewenste emissies en harmonischen. Ed herinnerde zich dat het ontwerpen van een print voor een UHF-ontwerp (ultra-hoge frequentie) "een uitdaging" was, omdat hij in de clinch lag met de printontwerpers van Elektuur die niet vertrouwd waren met de nukken en kuren van sub-pF strooicapaciteiten en spoelen in het nH-bereik, en zich alleen comfortabel voelden met 'DC-spul' zoals audio, microcontrollers en PSU's. Uitein-

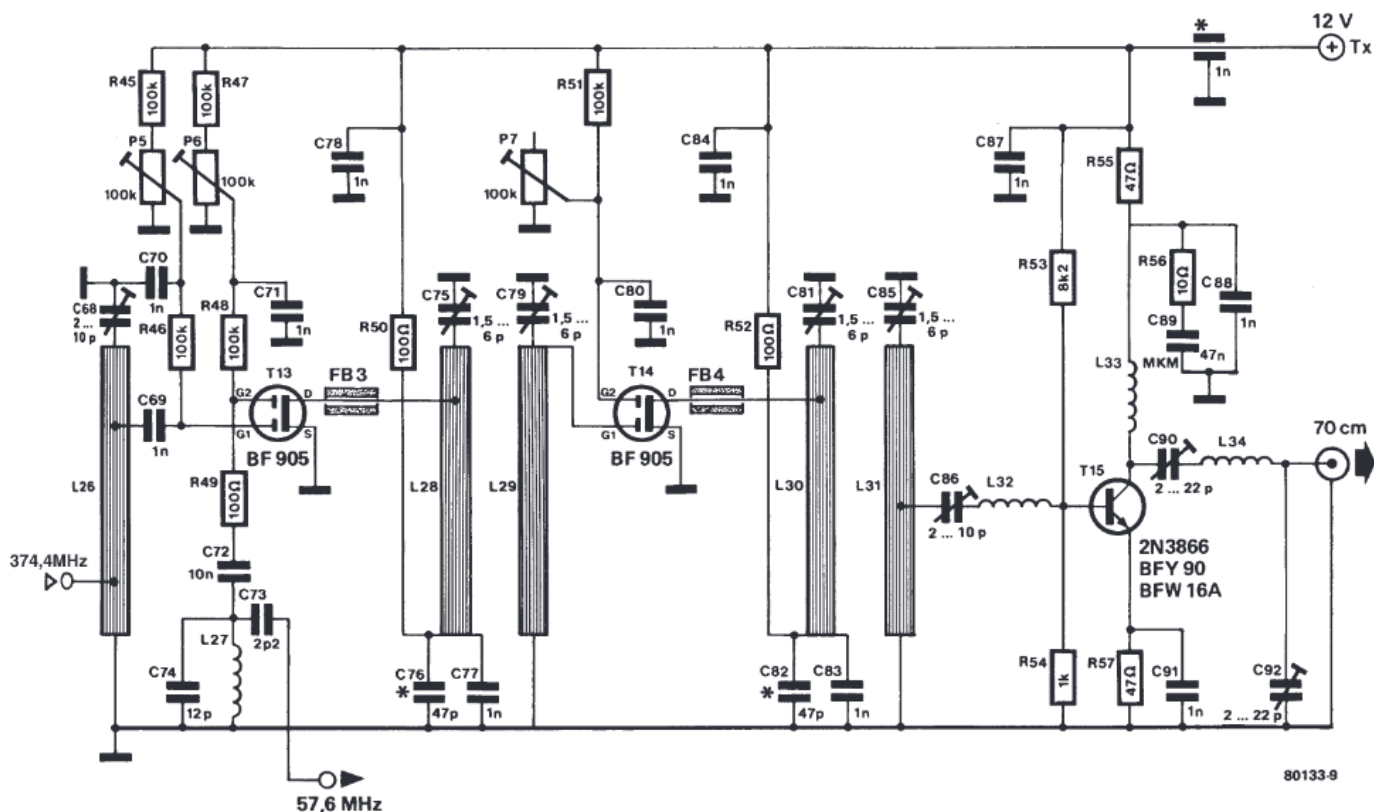
delijk werd een uitstekende oplossing gevonden in de vorm van van op de print geëtste microstriplijn-spoelen voor de UHF-delen van de schakeling, en niet te vergeten een verzilverde print en natuurlijk vertinde blikken 'wanden' op de print om ongewenste emissie tot een minimum te beperken.

Luid en duidelijk

In het begin van de jaren tachtig had de 70cm-band een bijzondere aantrekkingskracht, niet alleen als ontmoetingsplaats



Prototype van de 70cm-transverter gebouwd door Gerrit (PA0HKD) en Ed (PE1CJP) voor Elektuur in 1981. De transverter wordt aangestuurd door een Icom IC211 2 meter all-mode transceiver.

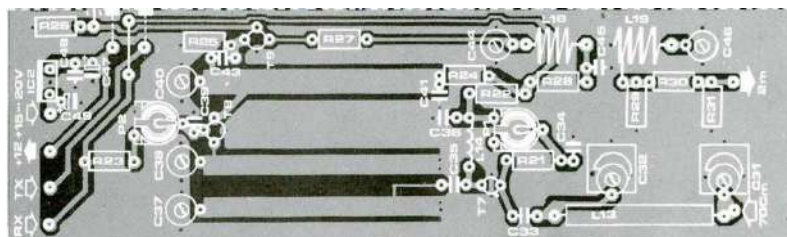


Schema van het UHF-gedeelte van de transverter. Voor die 'microstriplijn-spoelen' moest in 1981 door de toenmalige tekenafdeling van Elektor een nieuw symbool worden gemaakt en een woord worden toegevoegd aan het technisch woordenboek van de redactie.

voor zendamateurs met 100% zelfgebouwde apparatuur (waaronder amateurtelevisie, ATV), maar ook voor satellietcommunicatie die continentoverschrijdende QSO's in CW en SSB mogelijk maakte, met een relatief gering zendvermogens en sterk bundelende antennes.

De 70cm-transverter van Elektuur haalde een hoge Q-factor bij de zendamateurs, kwam op het juiste moment en was precies afgestemd op hun wensen. Het project was briljant uitgewerkt en glorieus gedocumenteerd in twee tijdschriftartikelen, resulterend in een UFB-publicatie (Ultra-Fine Business). 

220214-03



Een deel van de print van de omvormer, overgenomen uit de uitgave van juni 1981. Die rechthoekige gebieden tussen de dikke zwarte lijnen zijn geen massieve kortsluitingen, maar spoelen in 'microstrip'-stijl-uitvoering, afgestemd op ongeveer 430 MHz. Ze zijn niet iets wat je kunt slepen, neerzetten en autorouteren!

Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur via jan.buiting@elektor.com.

WEB LINKS

- [1] J. de Winter, "Transverter voor 70 cm (1)", Elektuur mei 1981:
<http://www.elektormagazine.nl/magazine/elektor-198105/42938>
- [2] J. de Winter, "Transverter voor 70 cm (2)", Elektor juni 1981:
<https://www.elektormagazine.nl/magazine/elektor-198106/42952>

Climate Calling Engineers

Move Fast and Fix Things

By Priscilla Haring-Kuipers

The sixth synthesis report by the Intergovernmental Panel on Climate Change (IPCC) is clear: “Human activities, principally through emissions of greenhouse gases, have unequivocally caused global warming. What are we going to do about it?”

The sixth synthesis report by the Intergovernmental Panel on Climate Change (IPCC) is the collective scientific wisdom on climate change and how to fix it. [1] They inform the UN. The main message of their latest report has hope: “There are multiple, feasible and effective options to reduce greenhouse gas emissions and adapt to human-caused climate change, and they are available now,” [2] but currently, we are not applying the technical solutions we have with enough vigour, scale or speed.

Calling all engineers! There is a lot to do, so let me give you some highlights:

Current Climate

We are now at 1.1°C global warming and will likely reach 1.5°C in the early 2030s and shoot up to 3.5°C this century if we don't change drastically. “There is a rapidly closing window of opportunity to secure a liveable and sustainable future for all.” [1] We have already caused a lot of damage across ecosystems. More than climate scientists estimated earlier. We have lost many species, nearly 50% of coastal wetlands, and we are impacting ecosystems in ways that are not reversible. Cities have become hotter and the air we breathe more polluted.

Our supply chain has already been impacted by the more frequent occurring extreme weather, making factories freeze or catch fire. Water is fast becoming a contested resource, and factories should look into either recycling or using seawater.

We have not done nothing. Agreements made at Kyoto and Paris have helped. Social movements have accelerated climate action. We can still save ourselves with climate resilient development based on science, indigenous knowledge and local context. High-tech and low-tech solutions working together.

“Individuals with high socio-economic status contribute disproportionately to emissions, and have the highest potential for emissions reductions, e.g., as citizens, investors, consumers, role models, and professionals.” [1] That means us. What we do and what we demand of our governance makes a big difference. What you choose to work on as an engineer will either contribute to a liveable world or to further heating up the place. When we support developing regions with our technological development, they can leapfrog to low-emissions solutions with us.

Stop That

If we are ever to stay under 2°C of global warming, a lot of fossil fuels are going to have to stay in the ground. Today, new fossil fuel developments are still being funded, and the fossil fuel industry receives more money in private investments, public subsidies and tax breaks than developments tackling climate adaptation and mitigation. [1] Simply ending fossil fuel subsidies would lower greenhouse gas emissions with 10% by 2030, while improving public revenue that could be redirected to our necessary transition. If your work or your pension funds are connected to the fossil fuel industry, you might want to start looking for a way to decouple before the well is shut down. Our electronics are heavy on petrochemicals and will be looking to shift to bio-based alternatives. Opportunities abound.





By Priscilla Haring-Kuipers, made with DALL-E: Electronics engineer soldering a product to achieve a future save from climate change.

Carbon pricing such as carbon taxes or emission trading have led to some low-cost emission reduction measures but have not been very successful to promote the higher-cost measures that are necessary to shift an industry. We need more. Luckily, climate laws are increasing, and they are helping to fight climate change causes and effects. Climate-related litigation is growing and has already had an effect on the “outcome and ambition of climate governance.” [1] It is likely that climate law will grow in the near future, both internationally and on regional levels. The WEEE regulations and the Supply Chain Act are early versions of what is coming. Your efforts and your company can be ahead of the curve, riding the green wave, or you can be dragged along by legislation, but everyone is coming eventually.

Technology for the Win

“If all technically available options were used, global emissions could be at least halved by 2030, at manageable costs.” [1] We need tons of engineers to roll out, scale up, improve and adapt to local circumstances many of the already available and proven solutions.

In the last decade, the cost of solar energy has dropped by 85%, wind energy by 55% and lithium-ion batteries by 85%. Meanwhile, deployment has increased over tenfold for solar and over a hundred-fold for electric vehicles. In some areas and industries, keeping the old is becoming more expensive than changing to the new. Work on whatever you can to push, develop and spread this development.

Green energy will not only curb our emission, but the economic benefit in air quality alone would offset the cost of the transition. Co-development of energy efficiency and renewable energy will create a happy feedback loop of improvement. Work on big renewables and small-scale nets, smart-grids, transmission and capacity is very much needed.

Cities are critical in this transition. We can build or retrofit to match our new low-emission lifestyles and make space for cycling and walking, teleworking and electric public transport. More plants and water in cities would help cool during heatwaves, process heavy rainfall slower and keep moist during droughts while benefiting the health and well-being of all who live there. Engineers should work on building materials and practices, sustainable urban planning and maintenance, digital communities and smart transport solutions. Many cities have already announced a net-zero emissions target. My city started a green jobs market for all the technical roles we are going to need to develop, install and maintain this bright new future. Your city might have a similar initiative.

Carbon capture is most reliably done by reforestation, improved forest management, soil carbon sequestration, peatland restoration and coastal blue carbon management. Protecting high-carbon ecosystems would have an immediate impact. Globally, we need to protect 30-50% of our land and water to maintain a resilient biodiversity. Throw your skill set behind any project that supports conservation and restoration.

Your time is now. ◀

230265-01

No Geo-Engineering

Blocking the sun with solar shields or sulfur is a no-go. Short-term and local cooling effects are likely, but the amount of green house gases would still grow and the acidification of our oceans would continue. We don't know enough about the effects on the targeted region nor our global ecology. Once it is up and running, turning it off could cause “rapid climate change.” The risks are too great, and the reward is too uncertain.

WEB LINKS

- [1] IPCC, “AR6 Synthesis Report: Climate Change 2023,” 2023: <https://www.ipcc.ch/report/ar6/syr/>
- [2] IPCC, “Urgent climate action can secure a liveable future for all,” March 20, 2023: <https://www.ipcc.ch/2023/03/20/press-release-ar6-synthesis-report/>

De Elektor Store

nooit duur, altijd verrassend!

De Elektor-store heeft zich ontwikkeld van de community-shop voor de eigen producten van Elektor (boeken, tijdschriften, kits en modules) tot een volwassen webshop die veel waardevolle elektronica-aanbiedingen heeft.

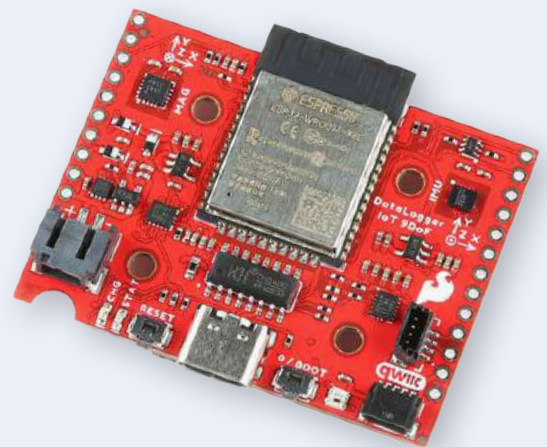
We bieden hier producten aan waar we zelf enthousiast over zijn of die we gewoon willen uitproberen. Suggesties zijn altijd welkom (sale@elektor.nl).

SparkFun DataLogger IoT (9DoF)

De SparkFun DataLogger IoT (9DoF) is een datalogger die al is voorgeprogrammeerd om automatisch IMU, GPS en verschillende druk-, vochtigheids- en afstandssensoren te loggen. Allemaal zonder ook maar één regel code te hoeven schrijven! De DataLogger detecteert, configureert en logt automatisch Qwiic sensoren. Hij is speciaal ontworpen voor gebruikers die alleen veel data dienen vast te leggen in een CSV- of JSON-bestand om dan verder te kunnen gaan met hun grotere project.

Prijs: € 94,95

Ledenprijs: € 85,46



 www.elektor.nl/20487

Miniware MDP-XP Digitale Voedingsset (MDP-M01 + MDP-P906)



Het MDP (Mini Digital Power System) is een modulair systeem van programmeerbare lineaire DC-voedingen, waarbij meerdere modules op maat kunnen worden gecombineerd. De MDP-XP bestaat uit een display control module (MDP-M01) en een digitale power module (MDP-P906).

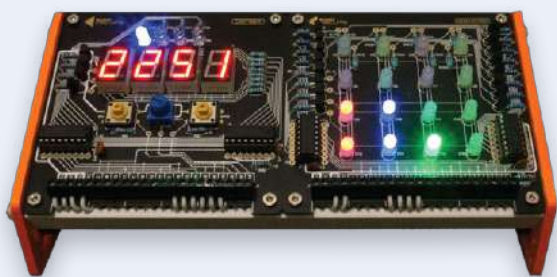
Prijs: € 269,00

Ledenprijs: € 242,10

 www.elektor.nl/20458



Short Circuits: The 4-Pack (Arduino-compatible elektronica- platform)

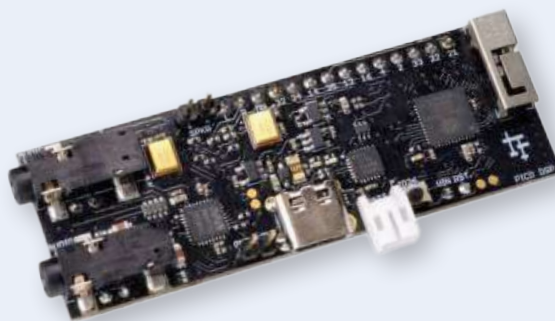


Prijs: € 99,95

Ledenprijs: € 89,96

www.elektor.nl/20474

PÚCA DSP ESP32 Development Board

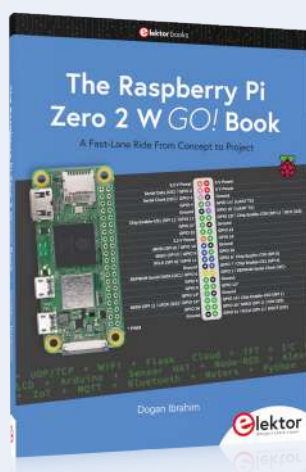


Prijs: € 69,95

Ledenprijs: € 62,96

www.elektor.nl/20504

The Raspberry Pi Zero 2 W GO! Book



Prijs: € 34,95

Ledenprijs: € 31,46

www.elektor.nl/20445

The Elektor Power Supply Collection (USB-stick)



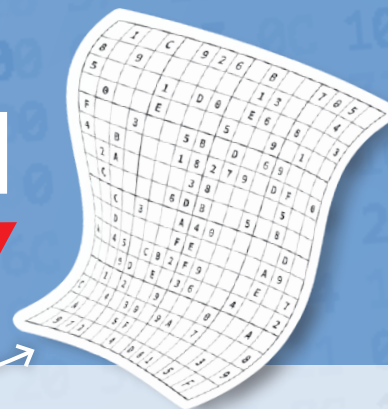
Prijs: ~~€ 49,95~~

Actieprijs: € 34,95

www.elektor.nl/20451

Hexadoku

puzzelen voor elektronici



PC, oscilloscoop en soldeerbout kunnen weer even op adem komen terwijl u uw hersenen pijnigt met onze Hexadoku. De instructies voor deze puzzel zijn heel eenvoudig. De Hexadoku werkt met de hexadecimale cijfers 0 t/m F, helemaal in de stijl van elektronici en programmeurs.

Vul het diagram van 16 x 16 hokjes zodanig in dat **alle** hexadecimale cijfers van 0 t/m F (dus 0...9 en A...F) precies éénmaal voorkomen in elke rij, in elke kolom en in elk vak van 4 x 4 hokjes (gemarkeerd door de dikkere zwarte lijnen). Een aantal cijfers is al aangegeven en deze bepalen de uitgangssituatie voor de puzzel.

Onder de inzenders met de goede oplossing verloten we vijf waardebonnen. Om mee te dingen naar een van deze prijzen dient u **de cijfers in de grijze hokjes** naar ons op te sturen.



Doe mee en win!

Onder de internationale inzenders met het juiste antwoord verloten we vijf Elektor-waardebonnen, elk ter waarde van **50 Euro**.

Het is dus zeker de moeite waard om mee te doen!

Stuur uw antwoord (de getallen in de grijze hokjes) vóór **15 augustus 2023** naar:

hexadoku@elektor.nl

DE PRIJSWINNAARS

De juiste oplossing van de Hexadoku uit het mei/juni-nummer 2023 is: **EBoC8**.

Oplossingen die ons vóór 15 juni 2022 hebben bereikt, deden mee aan de trekking van vijf Elektor-waardebonnen.

De winnaars zijn bekendgemaakt op www.elektormagazine.com/hexadoku.

Allemaal van harte gefeliciteerd!

D		1	6	0		2	7		A						B
	3								0	1	8	2	7		
5		F					4	3	B		D			0	A
A					3	5					F			4	
3				4			1	9	C			E			
			1		B			6						C	8
6			5			9	0	A	8					D	F
0		A		C		D				1		4	5	6	7
	D	6		8	4	F			E			2			
2	E	8		5		0		7	4			6	1		
	F						6			2	B			5	
	5	0	4							9		C		E	8
	6			9			3	C	7		0				
					5	4	2		9						
		5	2		0	E	A			D	8			B	
8		9					F				4				6

E	2	5	A	0	3	7	B	1	F	8	C	6	4	9	D
6	C	D	F	1	5	8	9	3	2	4	0	E	B	A	7
B	0	1	7	6	E	4	F	D	9	5	A	C	8	3	2
8	3	4	9	2	A	C	D	6	7	B	E	F	0	1	5
D	5	F	8	4	B	A	7	C	1	6	3	0	E	2	9
7	4	2	3	E	0	1	C	F	5	9	B	D	6	8	A
9	6	B	0	3	D	F	8	2	A	E	7	5	1	4	C
A	1	C	E	5	2	9	6	4	8	0	D	7	3	B	F
C	7	8	2	A	9	D	3	0	E	1	6	B	F	5	4
F	D	A	1	7	6	0	2	5	B	3	4	8	9	C	E
3	E	9	6	B	8	5	4	A	C	7	F	1	2	D	0
4	B	0	5	C	F	E	1	8	D	2	9	3	A	7	6
0	9	E	B	D	1	2	A	7	3	F	5	4	C	6	8
1	A	7	D	9	4	6	E	B	0	C	8	2	5	F	3
2	8	6	C	F	7	3	5	E	4	A	1	9	D	0	B
5	F	3	4	8	C	B	0	9	6	D	2	A	7	E	1

Medewerkers van Elektor International Media en hun familieleden zijn van deelname uitgesloten.



Algemene Inlichtingen- en
Veiligheidsdienst
Ministerie van Binnenlandse Zaken en
Koninkrijksrelaties

Elektronicus AIVD

Gebruik jouw kennis van elektronica voor het aanpassen en ontwikkelen van middelen die operationeel worden ingezet. Kom met bewezen technieken tot snelle en praktische oplossingen. Hiermee ondersteun je de collega's bij het uitvoeren van de operationele activiteiten van de AIVD.

Wat ga je doen?

Bij de elektronische werkplaats worden er bestaande (hulp)middelen aangepast, die worden gebruikt tijdens de operationele activiteiten van de Algemene Inlichtingen- en Veiligheidsdienst (AIVD). Dit gebeurt in afstemming met de collega's die de middelen inzetten in het veld. Wanneer aanpassen niet mogelijk is worden deze middelen zelf ontwikkeld, in de vorm van embedded systemen. Hierbij wordt een combinatie gebruikt van elektronica (hardware) en embedded software (bare metal programming).

Als specialist in embedded systemen maak je gebruik van jouw uitgebreide kennis op het gebied van elektronica. Deze kennis weet je creatief en praktisch toe te passen, waardoor je elke keer weer opnieuw vlot met een werkbare oplossing weet te komen.

Kwaliteit en betrouwbaarheid zijn hierbij erg belangrijk, want je kunt niet zelf het veld in om het een en ander bij te werken. Omdat van de aangeleverde systemen niet altijd precies bekend is hoe deze functioneren, ben je in staat dit te onderzoeken: werken met een multimeter, oscilloscoop of logic analyzer is geen probleem voor je. Ook zet je gevonden oplossingen om naar een printplaat die je zelf ontwerpt en daarna laat produceren. Hierna laad je de door jou geschreven software in deze printplaat en test je het geheel. Tijdens dit gehele proces werk je nauw samen met andere afdelingen binnen de directie Operatiën. Je blijft hierbij praktijkgericht toewerken naar een zo eenvoudig mogelijke oplossing.

Wat krijg je terug?

In deze rol krijg je grote vrijheid om zelf oplossingen te verzinnen, zolang de interne klant uiteindelijk maar tevreden is. Je krijgt ruimte om te experimenteren en mee te denken over de inzet van verschillende soorten gereedschap. De creativiteit die we van je vragen wordt gewaardeerd: je kan deze gebruiken om met eigen manieren tot een werkend eindresultaat te komen. Ook ben je nauw betrokken bij de operationele inzet van de AIVD, wat zorgt voor een interessante en veelzijdige inhoud van het werk. Aangezien geen operationele situatie hetzelfde is gaat het tenslotte om maatwerk.

Functie-eisen

Je bent creatief, nieuwsgierig en leergierig. Je houdt de ontwikkelingen in het werkveld goed bij en je vindt het leuk om nieuwe dingen te onderzoeken, om te verkennen of ze toepasbaar zijn in je werk. Je stelt je dienstverlenend op naar interne klanten, waarbij je vooral goed luistert en analyseert hoe jij kunt helpen met jouw kennis en producten.

Hiernaast:

- Beschik je over een mbo4 diploma elektrotechniek, specialisatie elektronica;
- Heb je parate kennis van elektriciteitsleer en praktische kennis van analoge en digitale elektronica;
- Heb je vaardigheid in het lezen van technische datasheets;
- Heb je vaardigheid in het ontwerpen van printplaten, met behulp van software zoals KiCad of Altium;
- Kan je kleuren goed onderscheiden en beschik je over goede fijne motoriek, zodat het solderen van microscopisch kleine onderdelen geen probleem vormt.

Voor meer informatie over deze vacature:

Wervingsbureau (op werkdagen vanaf

08:00 tot 15:30 uur)

079-3205373

www.werkenvoornederland.nl/vacatures/elektronicus-AIVD-2023-0071#0

Grootste keuze aan elektronische componenten™

Op voorraad en klaar om te verzenden



nl.mouser.com

