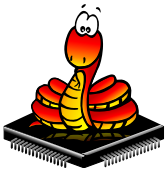
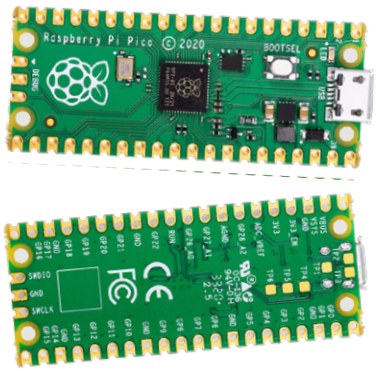
LoRa met de Raspberry Pi Pico

pret met MicroPython



Mathias Claußen (Elektor)

Programmeren in MicroPython op een Raspberry Pi Pico gaat heel gemakkelijk. Met een RFM95 LoRa-module van SeeedStudio en een DS18B20-temperatuursensor kunnen we snel een LoRaWAN-node met MicroPython bouwen. En omdat een schakeling op een breadboard ietwat 'gammel' kan zijn, hebben we ook een suggestie voor een print.



Figuur 1. De Raspberry Pi Pico.

Figuur 2. De RFM95-module.

De Raspberry Pi Pico (**figuur 1**) is een nieuw microcontrollerboard met de RP2040. Deze chip is ontworpen en ontwikkeld door de Raspberry Pi Foundation. Het is hun eerste halfgeleider. Na het zien van al het nieuws rondom die chip, en na het horen van positieve en negatieve geluiden erover, vond ik het tijd om maar eens aan enkele projecten te beginnen. De kaart heeft geen WiFi aan boord, maar we kunnen wel draagbare toepassingen maken met batterijvoeding. Dus besloot ik de Raspberry Pi Pico te combineren met een LoRa-modem. De HOPERF RFM95 (**figuur 2**) is een betaalbare en breed ondersteunde module, die we al eerder hebben toegepast in Elektor-projecten, zoals *Beginnen met LoRaWAN* [1]. In dat artikel werkten we met een STM32 BluePill-module, maar dit keer gebruiken we de Raspberry Pi Pico.

We gaan een eenvoudige temperatuursensor bouwen met een DS18B20 en de data sturen we naar een LoRaWAN-gateway, die de gegevens doorstuurt naar de clouddienst The Things Network. Daar pikken we de data op met een (klassieke) Raspberry Pi, waarop we het (huis) automatiseringsplatform Node-RED draaien.

We bouwen onze temperatuurmetende LoRa-node op een breadboard op, maar als u liever een echte print hebt, hebben we met KiCad iets voor u voorbereid.

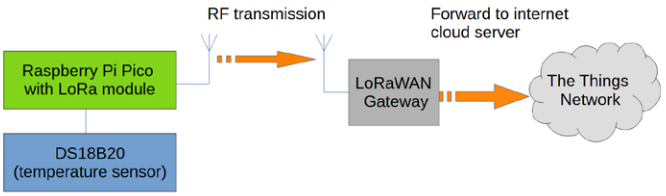
Figuur 3 toont de dataflow van de Raspberry Pi Pico via LoRaWAN naar de server van The Things Network. Zoals u ziet, zit er een gateway in dat pad, die de HF-data vertaalt in iets dat via Internet naar The Things Network kan worden gestuurd. We gaan de data van het The Things Network verwerken, want we zijn geïnteresseerd in de geregistreerde temperaturen. We doen dat door de data op te halen en weer te geven op een kleine webpagina. **Figuur 4** toont de dataflow van The Things Network naar de Raspberry Pi die een webpagina genereert met de huidige temperatuur en een grafiek van eerder gemeten temperatuurwaarden.

We gebruiken Node-RED om de data van The Things Network te halen en een webpagina te genereren. Node-RED is een tool waarmee we grafische dataflows kunnen maken en data verwerken. Dat kan met elke webbrowser. U hoeft alleen maar de Node-RED-server te installeren en een browservenster te openen om aan de slag te gaan. We hebben Node-RED al eerder gebruikt in Elektor-projecten. Het is een

snelle instapmogelijkheid voor dataverwerking en -behandeling. In de ElektorShop is een inleidend boek over Node-RED verkrijgbaar. (Zie **Gerelateerde producten** voor meer informatie)
De ingrediënten voor dit project zijn eenvoudig. Ze zijn te vinden in het kader **Benodigd materiaal**. Natuurlijk moet u wel binnen het bereik van een LoRaWAN-gateway zitten die uw data naar The Things Network kan sturen. Als dat niet het geval is, maar u wilt uw eigen gateway beginnen (en daarmee de dekking van LoRaWAN verbeteren), kijk dan bij [3]. Dat is een kant-en-klare oplossing die u meteen kunt inzetten. Een alternatief is om zelf een gateway op te bouwen rond een Raspberry Pi (zie [4]).

Eerst op een breadboard

Om de werking van de schakeling te demonstreren, bouwen we hem eerst op op een breadboard. Zo kunnen we snel gaan testen, en de opbouw is heel eenvoudig (zie **figuur 5**). Om het LoRa-modem te verbinden met de Raspberry Pi Pico, hebben we vier verbindingen nodig voor de SPI-interface: MOSI, MISO, SCK en CS (data in, data uit, klok en chip select). Daarmee kunnen we data uitwisselen met het LoRa-modem. Daarnaast hebben we RESET en DIO0 nodig om het modem te besturen. DIO1 t/m DIO3 zijn aangesloten. Die zijn nodig voor sommige andere LoRa-bibliotheken, zoals LMIC (als we de Raspberry Pi Pico in C/C++ gaan programmeren). We kunnen één van de SPI-poorten van de Raspberry Pi Pico kiezen en die verbinden met het RFM95 LoRa-modem. Verder hoeven we alleen RESET en DIO0 van de module te verbinden met twee GPIO's van de Pico. Voor een compleet operationeel LoRa-modem moet ook nog een antenne worden aangesloten.
Als antenne gebruiken we gewoon een stuk draad. Een stukje koperdraad van 1 mm dik is ruim voldoende. De lengte is te berekenen als een λ/4-antenne voor 868 MHz, dat is het frequentiegebied waarin de LoRa-module werkt: $\lambda/4 = (c_0/868\ \text{MHz})/4 = (299792458\ \text{m/s})/(868000000\ 1/\text{s}*4) = 0{,}08635\ \text{m} = 8{,}635\ \text{cm}$. Dat zou de lengte zijn als de golf zich in een vacuüm zou voortplanten. Maar in koper is de snelheid van de golf lager en moeten we rekening houden met een verkortingsfactor. Bij een verkortingsfactor van 0,95 komen we uit op een lengte van 8,2 cm voor ons koperdraadje.
Hoewel een pull up-weerstand op de resetlijn niet wordt voorgeschreven, is in de praktijk gebleken dat die wel nodig is. Hij verbetert de stabiliteit, want zonder die weerstand kan de resetlijn storingen oppikken die de module onverwachts zouden resetten. Als u werkt met een eigen LoRa-board, kan het verstandig zijn om de weerstand wel in het printontwerp mee te nemen, maar hem alleen te monteren als blijkt dat hij nodig is.
Voor de temperatuursensor gebruiken we een One-Wire-verbinding. Zoals de naam One-Wire al aangeeft, hebben we maar één GPIO-lijn nodig voor de communicatie. Het protocol kan helemaal in software worden geïmplementeerd, dus elke GPIO-pen is geschikt. Voor One-Wire is een pull up-weerstand noodzakelijk. Afhankelijk van de gebruikte controller kan dat een interne pull up zijn, die wordt



Figuur 3. Dataflow naar The Things Network.

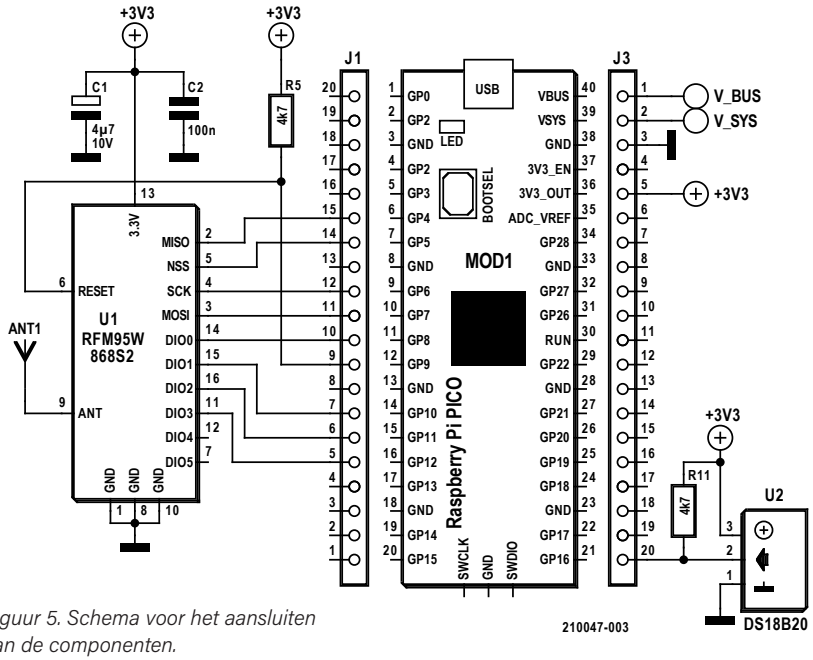
Figuur 4. Dataflow van The Things Network naar een eigen webpagina.

Benodigd materiaal

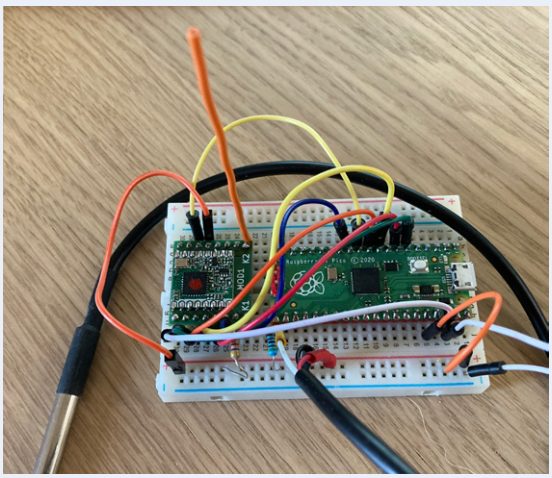
- Raspberry Pi Pico *
- RFM95 *
- Breadboard *
- 9 jumperkabeltjes male/male
- Breakoutboard voor RFM95
- 2x20-polige pinheader + 2x8-polige pinheader
- DS18B20-temperatuursensor
- 10 cm koperdraad (1 mm dik)

Onderdelen met * zijn te bestellen in de Elektor-shop, zie **Gerelateerde Producten**.

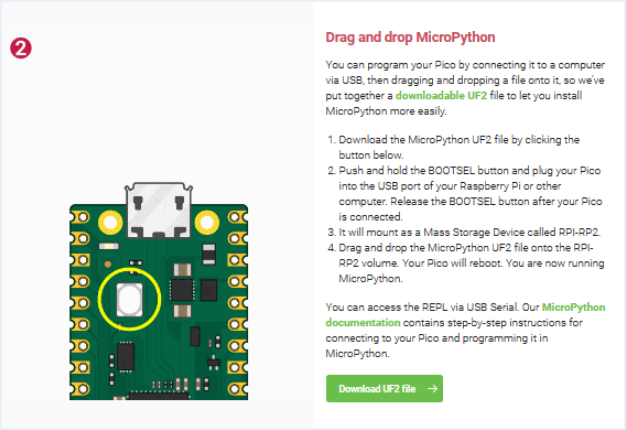
Gerber-files voor het breakout-board zijn te downloaden van [15].



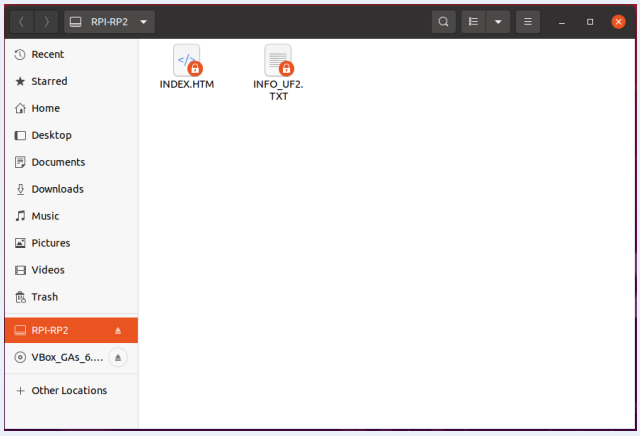
Figuur 5. Schema voor het aansluiten van de componenten.



Figuur 6. Opbouw op een breadboard.



Figuur 7. Download MicroPython voor de Raspberry Pi Pico.



Figuur 8. Raspberry Pi Pico in bootloader-modus.

Figuur 9. Thonny versie 3.3.4 is geïnstalleerd.

verzorgd door het GPIO-blok in de MCU, of een externe. Een externe pull up-weerstand geeft de meest betrouwbare communicatie.
Als alle hardware is aangesloten, ziet die eruit als in **figuur 6**, en kunnen we beginnen met coderen.

MicroPython en LoRa

We hebben in eerdere projecten al laten zien hoe LoRa in C/C++ kan worden geïmplementeerd, maar dit keer gaan we voor MicroPython. De Raspberry Pi Pico is heel geschikt voor educatieve toepassingen en het gebruik van MicroPython maakt alles nog iets gemakkelijker. Als MicroPython op de Raspberry Pi Pico nieuw voor u is, kijk dan eens naar het boek *Get Started with MicroPython on Raspberry Pi Pico* (Raspberry Pi Foundation, 2021) van Gareth Halfacree en Ben Everard. Wilt u een papieren versie, dan kunt u terecht in de onze shop [5]. Bent u tevreden met een e-boek, dan kunt u een gratis Engelstalige versie downloaden van [6].
De implementatie van LoRa met MicroPython is eerder gedaan op andere platforms, maar het brengt wel wat extra uitdagingen met zich mee. MicroPython is een geïnterpreteerde taal, net zoals BASIC dat was op de Commodore 64 en zijn soortgenoten. Dat maakt de code heel toegankelijk, maar er is wel meer CPU-overhead bij het uitvoeren van het programma en dat brengt een paar beperkingen met zich mee. We zullen een aangepaste versie van de library uLoRa van fantasticdonkey gebruiken die op GitHub te vinden is bij [7]. Deze bibliotheek is een afgeleide van de TinyLoRa-bibliotheek voor CircuitPython van Adafruit.
De genoemde beperkingen zijn dat we alleen data kunnen zenden, maar niets kunnen ontvangen van het LoRaWAN. In verband met die beperking moeten we ook gebruik maken van APB (*Activation By Personalization*) voor de authenticatie. Dat wil zeggen dat de netwerk-sessie- en de applicatiesessiesleutels in de code zijn opgeslagen.
Omdat het gaat om een voorbeeld om te laten zien hoe we LoRa met MicroPython op de Raspberry Pi Pico kunnen gebruiken, is het programma niet perfect, maar wel eenvoudig en snel toepasbaar. De aangepaste files zijn te downloaden van de Elektor-pagina op GitHub [12]. Vergeet ook niet dat deze software lijkt te werken, maar dat de stabiliteit wellicht voor verbetering vatbaar is.

Zet MicroPython op de Raspberry Pi Pico

De Raspberry Pi Pico wordt geleverd met een bootloader waarmee de firmware gemakkelijk kan worden vervangen. We willen MicroPython gebruiken, dus we installeren de nieuwste beschikbare versie van [8] zoals in **figuur 7**. Download het UF2-bestand van de internetpagina en zet uw Raspberry Pi Pico in bootloader-modus.
Om in bootloader-modus te komen, koppelen we de Raspberry Pi Pico los, drukken op de BOOTSEL-knop en sluiten hem weer aan op de computer. Er verschijnt dan een nieuw apparaat voor massaopslag op de PC zoals in **figuur 8**. Sleep het gedownloade UF2-bestand daar naartoe om MicroPython te installeren.
Start uw Raspberry Pi Pico opnieuw op, en u kunt aan de slag. Voor het ontwikkelen van software is het handig om een editor of een IDE

te hebben. Daarom gaan we nu Thonny installeren als Python-IDE.

Installeer Thonny

Het installeren van de Thonny-IDE op een recente Ubuntu 20.04 of zelfs op Windows is een kwestie van een paar muisklikken. Een installer voor Windows is te vinden op [9]. Voor Ubuntu gebruikt u wget -q -O - https://github.com/thonny/thonny/releases/download/v3.3.4/thonny-3.3.4.bash om de installer voor versie 3.3.4 met ondersteuning voor de Raspberry Pi Pico te downloaden. Als het bestand is gedownload, kunt u het uitvoeren met bash thonny-3.3.4.bash. Als dat lukt, hebt u nu versie 3.3.4 van de Thonny-IDE geïnstalleerd, zoals in **figuur 9**. We starten de IDE en gaan dan de Raspberry Pi Pico configureren. Dat gaat vanuit het menu met Tools → Options. Er opent dan een dialoogvenster zoals in **figuur 10**. We hebben nu de tools en de hardware op hun plaats; we kunnen gaan programmeren!

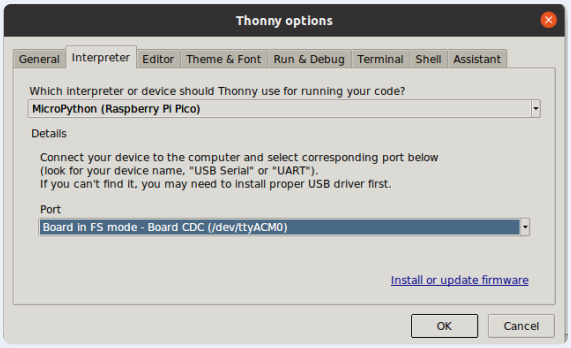
Voer de code in

Het is voor LoRa belangrijk om te weten, waar ter wereld u zich bevindt. We gebruiken de ISM-band, dat kan op 433 MHz, 868 MHz of 915 MHz zijn. In Europa is het over het algemeen 868 MHz. In de VS is het 915 MHz. We ontwikkelen dit project in Europa, dus we gebruiken de 868 MHz-band. Als u de code bouwt, pas hem dan aan voor uw locatie.

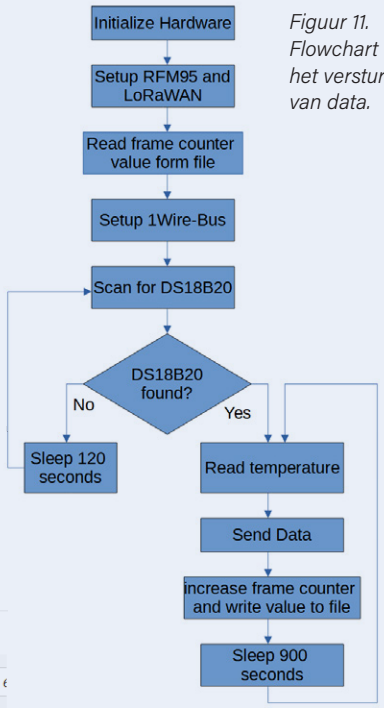
uLoRa is gepatcht voor gebruik op de Raspberry Pi Pico, dus u moet *ulora.py, ttn_eu.py, ulora_encryption.py* en *lora.py* op uw Raspberry Pi Pico zetten met de Thonny-IDE. Als u niet in Europa bent, moet u in plaats van *ttn_eu.py* de juiste *ttn_xx.py* voor uw locatie gebruiken. Dit voorziet uw board van de vereiste bibliotheken en van een basisvoorbeeld dat we gaan bespreken.

In *lora.py* staat het hoofdprogramma voor dit project. De flowchart is te zien in **figuur 11**. Na de initialisatie wordt gezocht naar een temperatuursensor op de One-Wire-bus. Als die niet wordt gevonden, wacht het programma 120 seconden en probeert het dan opnieuw. Als er een sensor is, wordt de temperatuur uitgelezen en daarna verstuurd. De manier waarop de waarde wordt gecodeerd is niet super-efficiënt, maar is wel handig voor deze demo. In **figuur 12** ziet u de verzonden data in de console van The Things Network. Na elk bericht moeten we 900 seconden wachten om te voldoen aan de fair use-policy van The Things Network. Als u wilt dat de code op uw Raspberry Pi Pico automatisch start, sla dan *lora.py* op als *main.py*.

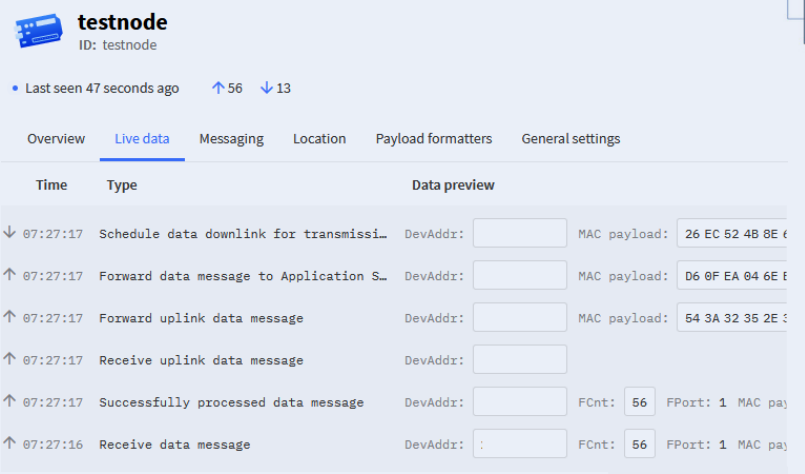
U ziet in de code dat er een waarde wordt geschreven naar *current.txt*. Daar wordt de huidige framecounter opgeslagen die we gebruiken voor de overdracht naar The Things Network. Als we die waarde niet zouden opslaan, zou hij weer nul zijn als we de node opnieuw opstarten, bijvoorbeeld na het verwisselen van de batterij. Als de



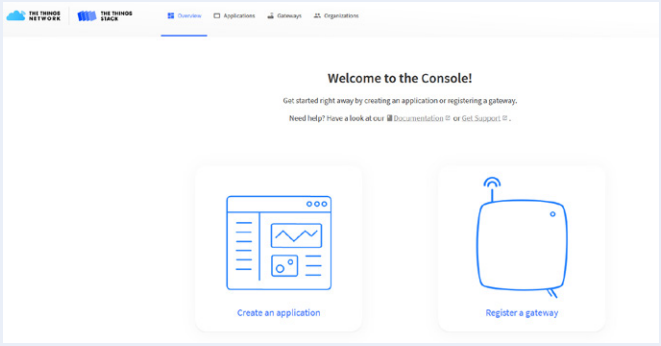
Figuur 10. Verbindingsinstellingen voor de Raspberry Pi Pico.



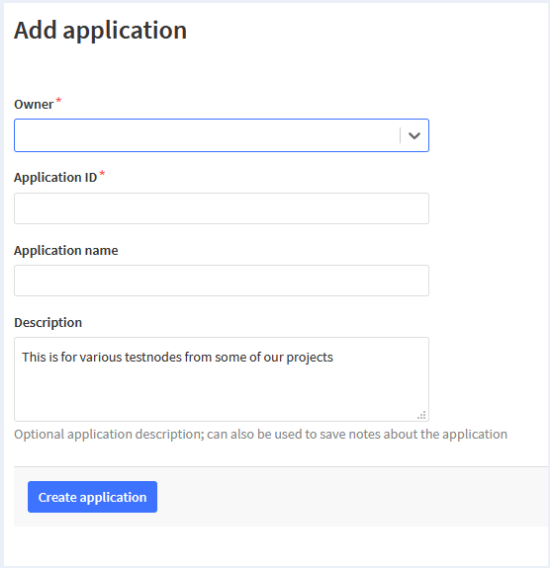
Figuur 11. Flowchart voor het versturen van data.



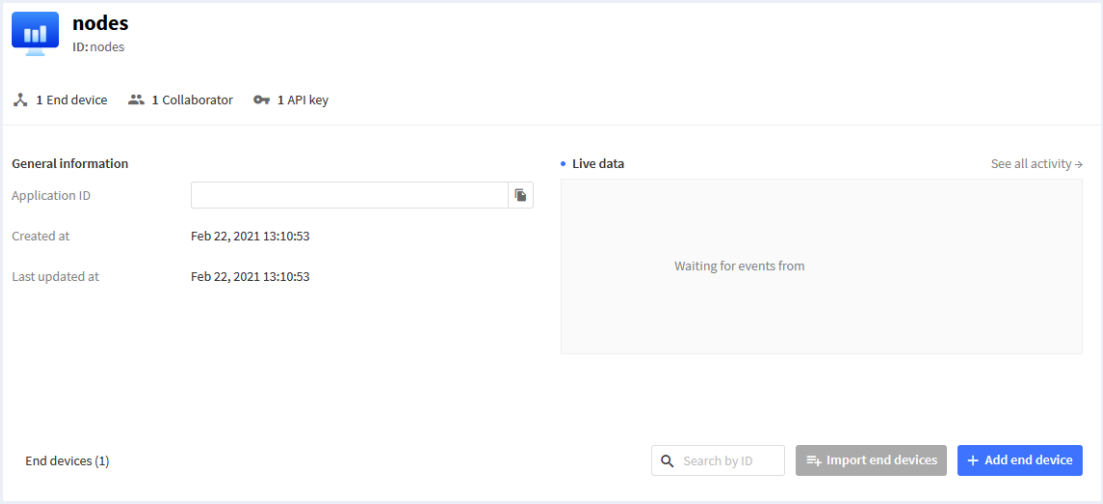
Figuur 12. Ontvangen data in de TTN-console.



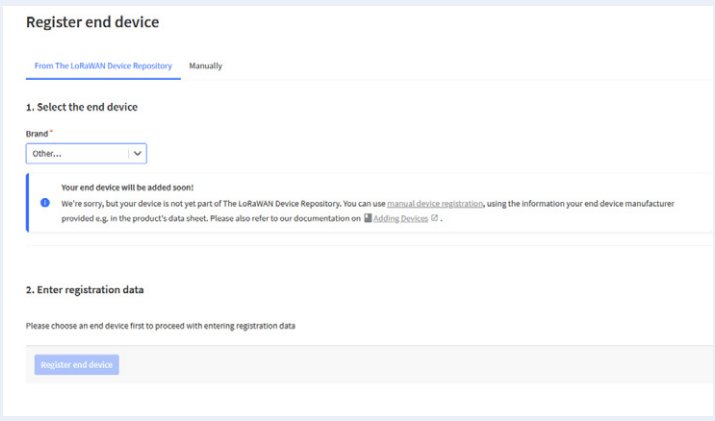
Figuur 13. Maak een nieuwe applicatie.



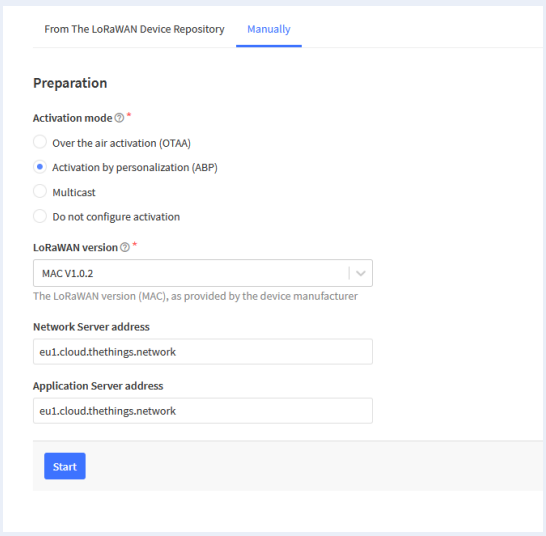
Figuur 14. Wizard voor het toevoegen van een applicatie.



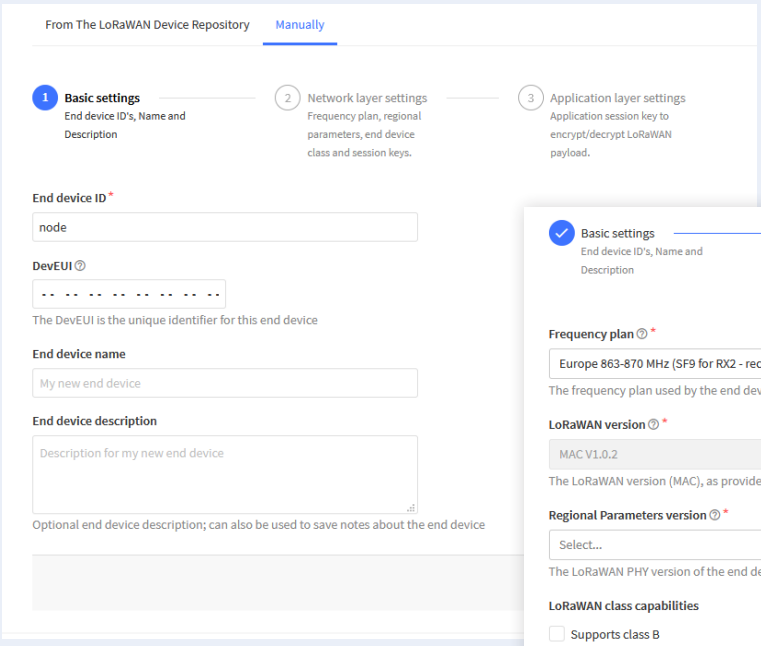
Figuur 15. Een nieuw device toevoegen.



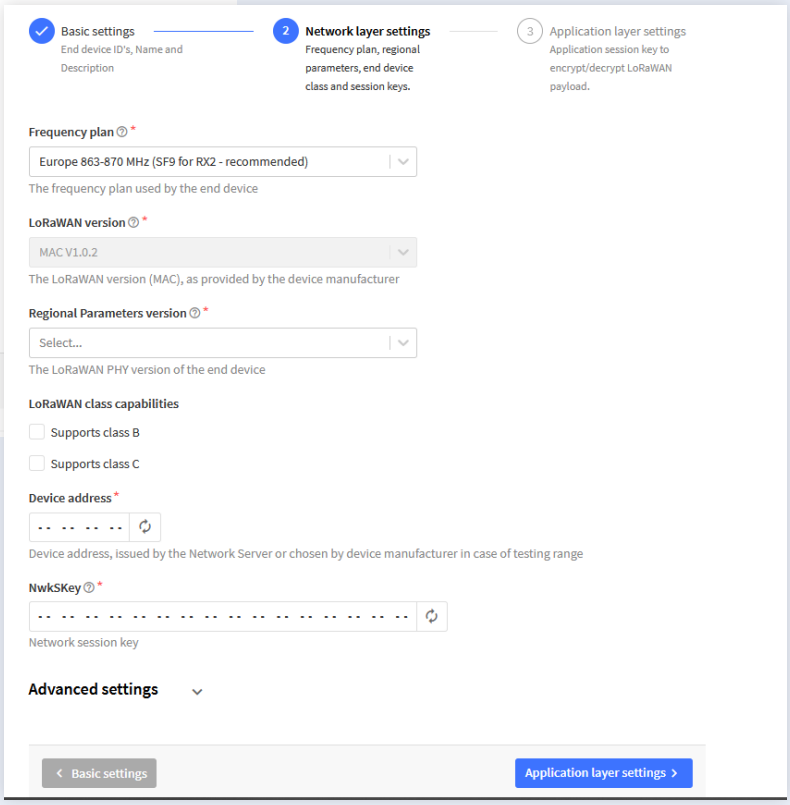
Figuur 16. Wizard voor het toevoegen van een node.



Figuur 17. Instellingen voor ABP-modus.



Figuur 18. Instellingen voor naam en EUI.



Figuur 19. Bandplan-instellingen.

teller telkens zou worden gereset als de node weer wordt opgestart, zou The Things Network de data om veiligheidsredenen negeren. Na elke overdracht wordt het bestand bijgewerkt met de huidige waarde en die wordt bij de volgende start weer gebruikt. Dit is niet ideaal en het verkort de levensduur van het flashgeheugen. Als we elke 15 minuten een nieuwe waarde schrijven, is dat 96 schrijfoperaties per dag, of 35040 per jaar, dat is verre van perfect en moet verbeterd worden.

Naar The Things Network en terug

Het opzetten van een LoRa-node is al in eerdere artikelen besproken, maar ik wil er toch nog iets over kwijt. Het eerdere artikel ging over The Things Network-stack versie 2, maar onlangs is versie 3 aangekondigd. Op sommige plekken kunt u uw toepassingen en gateways al overzetten naar de nieuwe versie 3-stack en -console.

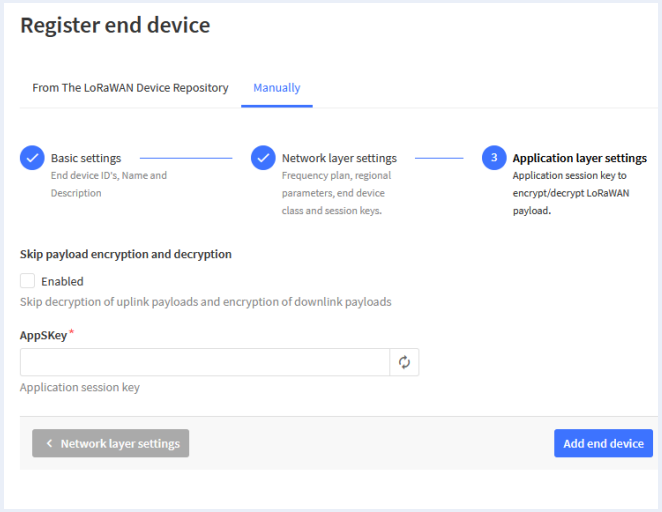
Om data te kunnen verzenden, moet u een aantal belangrijke parameters toevoegen aan de beschikbaar gestelde MicroPython-files. We hebben een account bij The Things Network nodig om de infrastructuur te gebruiken. Als er een account is gemaakt, of als u er een wilt maken, kunt u naar de versie 3-stack via [10]. Log in op uw account en maak een nieuwe node aan. Om dat te doen, moeten we een applicatie opzetten. Zoals u ziet in **figuur 13**, moet u op *Create an application* klikken en de wizard van **figuur 14** volgen. Selecteer de eigenaar van de toepassing, dat is uw account, en vul de Application-ID in. Klik op Create application om het proces af te ronden.

Nu we een nieuwe applicatie hebben opgezet, kunnen we er een nieuwe node aan toevoegen. Deze nieuwe node geeft ons de credentials die we moeten invullen in ons MicroPython-script. Kies in uw nieuwe applicatie *Add end device* om een wizard te starten voor het maken van een nieuwe node, zoals in **figuur 15**.

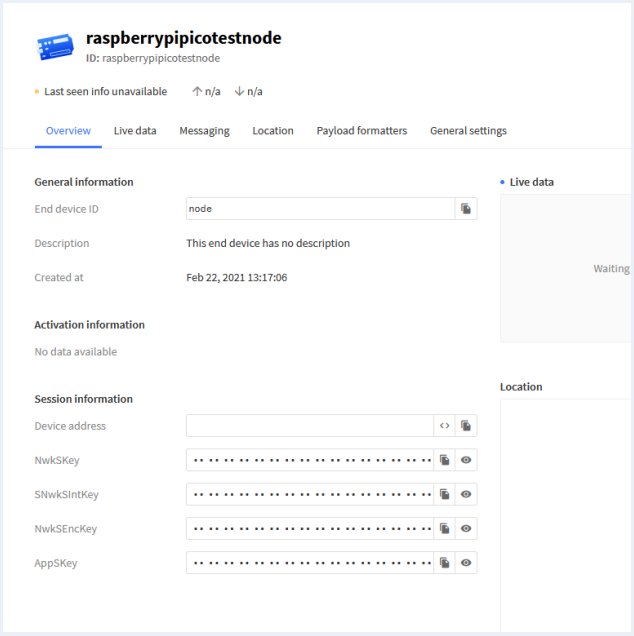
De wizard van **figuur 16** vraagt u een device te kiezen. Omdat ons device niet in de lijst staat moeten we *Manually* kiezen. Kies, zoals in **figuur 17**, voor *Activation by personalization* (ABP) en kies de LoRaWAN-versie voor onze MAC. Hier kunt u MAC V1.0.2 kiezen. Kies *Start* om naar de basisinstellingen te gaan zoals in **figuur 18**.

Voer een *End device ID* in, een unieke ID voor uw node. De velden *End device name* en *End device description* kunt u naar eigen inzicht invullen om later te weten welke nodes u gemaakt hebt. Op de volgende pagina (zie **figuur 19**) moet u een frequency plan kiezen. De invulling is afhankelijk van uw locatie. De node die wij bouwen, kent maar net genoeg LoRaWAN om data te verzenden, we hebben geen ondersteuning voor Class B of Class C. Druk op *Application layer settings* om door te gaan. Genereer een applicatiesessiesleutel zoals in **figuur 20** en sluit af met *Add end device*.

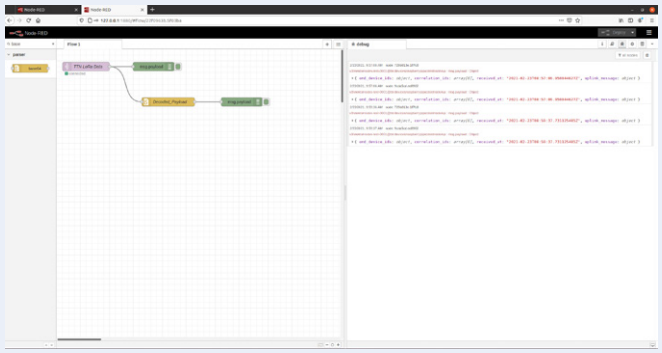
Het zojuist aangemaakte device verschijnt nu zoals in **figuur 21**. Voor ons MicroPython-script hebben we een *Device address*, *NwkSKey* en *AppSKey* nodig. Die zetten we in *lora.py*: `DEVADDR` bevat het deviceadres, `NWKEY` de NwkSKey en `APP` de AppSKey. Als alle gegevens goed



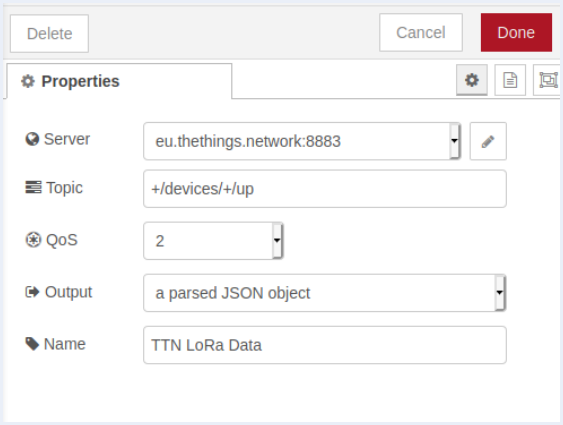
Figuur 20. Genereren van de applicatiesleutel.



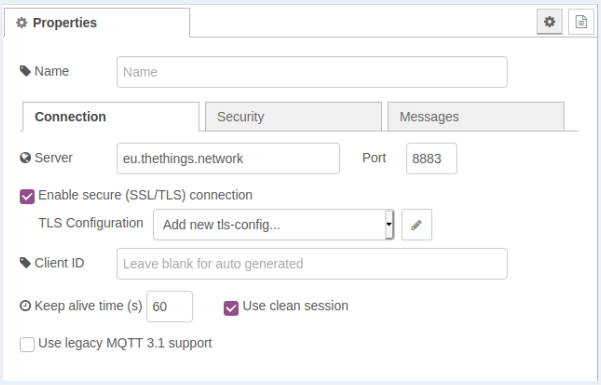
Figuur 21. Device-informatie voor de nieuwe node.



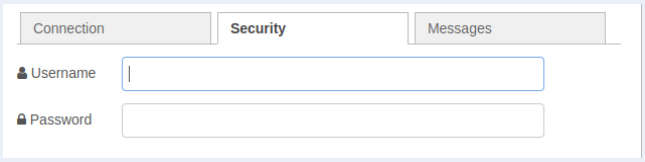
Figuur 22. Node-Red-flow voor LoRaWAN-data.



Figuur 23. Serverinstelling voor de MQTT-node.



Figuur 24. Server- en SSL-instellingen.



Figuur 25. Gebruikersnaam- en wachtwoordinstellingen.

zijn ingevoerd, zal onze data tevoorschijn komen in The Things Network.

Data verzamelen met Node-RED

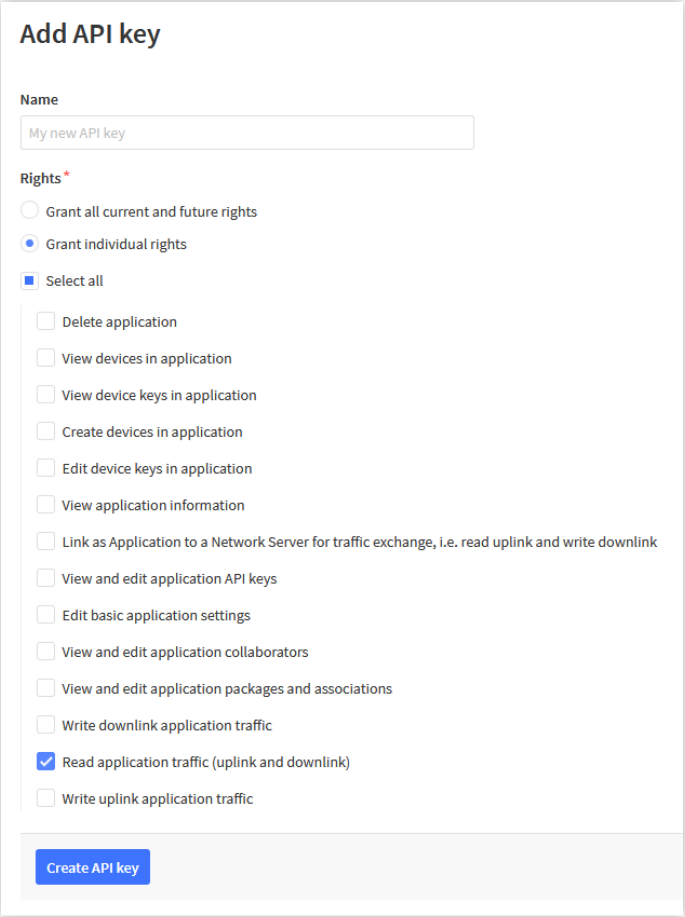
Om Node-RED te installeren op een Raspberry Pi openen we een terminalvenster of een SSH-sessie en geven we het volgende commando:

```
bash <(curl -sL https://raw.githubusercontent.
com/node-red/linux-installers/master/deb/
update-nodejs-and-nodered
```

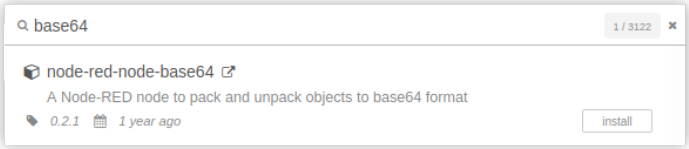
Om Node-RED als een service te laten draaien, moeten we na de installatie ook nog het volgende commando geven:

```
sudo systemctl enable nodered.service
```

De complete installatiehandleiding is te vinden op [11]. Geef het volgende commando om de service te starten: `sudo systemctl start nodered.service`.
Als Node-RED is geïnstalleerd en gestart, kunnen we onze verzonden data gaan ophalen met een flow zoals in **figuur 22**. Ook al hebben we nodes bij The Things Network, op de Raspberry Pi moeten we toch gebruik maken van een generieke MQTT-verbinding. Het gebruik van de nodes voor The Things Network op een Raspberry Pi kan leiden tot fouten en de dataflow laten crashen. Dat kan te maken hebben met compatibiliteitsproblemen, want dezelfde nodes werken wel goed met X86-hardware.
Bij de MQTT-verbinding zijn we geïnteresseerd in de data die onze node gaat versturen, de zogenaamde payload. In **figuur 22** ziet u de complete Node-RED-flow voor het ophalen van de data die we versturen. De configuratie begint met de MQTT-node. Hier moeten we aangeven welke server de data moet leveren en we moeten credentials meegeven.
Figuur 23 toont de *Server* en het *Topic* die we moeten invullen. Een topic is een soort gesprekskanaal over een bepaald onderwerp. Zo krijgen we alleen de berichten waar we in geïnteresseerd zijn. We gebruiken `+/devices/+/up` als topic. Dat betekent, dat we geïnteresseerd zijn in alle berichten van de node die geregistreerd zijn in onze applicatie. Als output verwachten we een JSON-object dat we verder kunnen verwerken. Bij de volgende stap moeten we de instellingen voor de servers van The Things Network instellen. Om een instelling aan te passen, klikken we op het potloodsymbool rechts van de server. Dan verschijnt een nieuwe dialoog zoals in **figuur 24**. Omdat we gebruik maken van de nieuwe V3-stack, kiezen we voor *eu1.cloud.thethings.network* en poort *8883*. Zorg dat *Enable secure (SSL/TLS) connection* is aangevinkt. Onder de tab security voeren we een gebruikersnaam en wachtwoord in (zie **figuur 25**). Gebruik als username de naam van de applicatie die we in The Things Network hebben aangemaakt. Het wachtwoord is wat lastiger te vinden. Ga naar de console van The Things Network en open uw applicatie. Klik in het menu links op *API keys* en voeg een API-key toe met *Add API key*. Er verschijnt dan een nieuwe wizard zoals in **figuur 26**. Kies de benodigde toegangsrechten en sluit de dialoog met *Create API Key*. U krijgt dan een API-key te zien die u kunt gebruiken. Sla deze op op een veilige plaats, want hij is later niet meer op te vragen. Deze key is het wachtwoord dat we moeten gebruiken voor Node-RED.



Figuur 26. Wizard voor een nieuwe API-key.

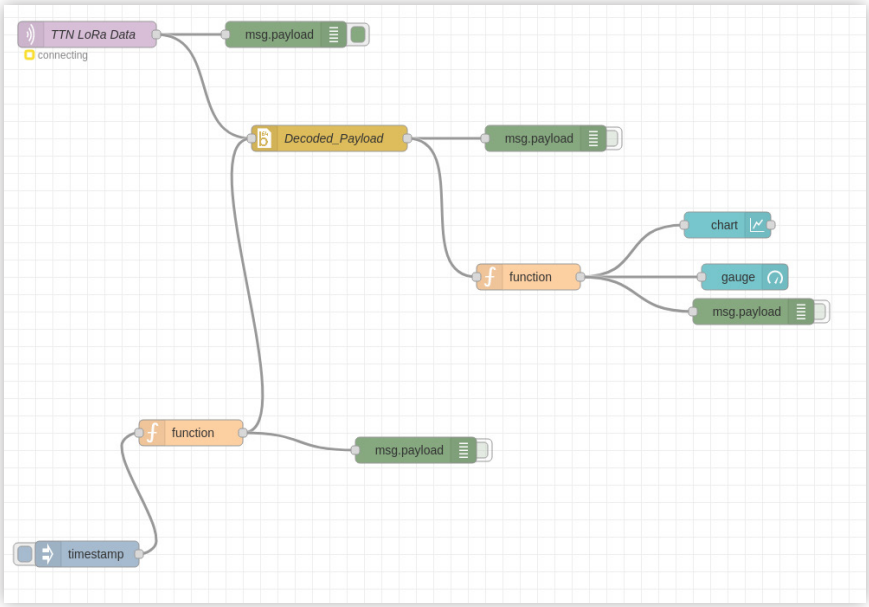


Figuur 27. BASE64-decoderinstellingen in Node-RED.

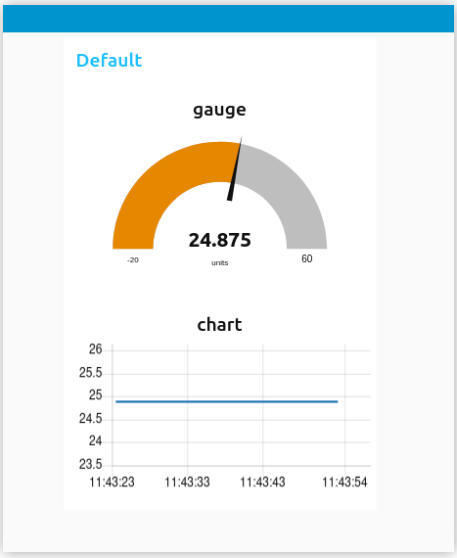
Als alle instellingen zijn gedaan, leggen we ze vast met *Done*. Als u nu de instellingen gebruikt, maakt de MQTT-node een verbinding en wordt nieuwe data weergegeven bij de node. Er is nog wel een klein probleempje. De payload, de data die onze node stuurt, is gecodeerd met BASE64. Om de ruwe bytes terug te krijgen, moeten we een BASE64-decoder toevoegen. Als die wordt geconfigureerd zoals in **figuur 27**, krijgt u de data als byte-arrays.
Voor het gemak is een demo-flow opgezet die er uitziet zoals in **figuur 28**. Die maakt nieuw aangekomen data toegankelijk via een webbrowser. De pagina komt er dan uit te zien zoals in **figuur 29**. Hier hoeft u alleen maar uw credentials in te vullen.

Simpel en geschikt voor beginners

Dit is niet het ingewikkeldste project dat we kunnen maken, maar het is een uitgangspunt om zelf te gaan knutselen met de Raspberry Pi Pico en LoRaWAN. MicroPython kan, vooral voor beginners, een gemakkelijke instap in de wereld van embedded systemen zijn. De



Figuur 28. Voorbeeld van de dataflow in Node-RED.



Figuur 29. Data weergegeven op een webpagina.

benodigde tijd om dit project te bouwen en te draaien maakt het heel geschikt voor (virtuele) lessen en workshops. Maar we hadden u nog een print beloofd! Als u alleen software wilt, kunt u hier stoppen met lezen. Als u de elektronica niet alleen wilt gebruiken, maar ook bouwen, lees dan verder.

Waarschuwing

Meestal zijn de schema's en printen in Elektor door ons gebouwd en getest (dan weet u in elk geval zeker dat ze niet ontploffen). Dat ligt in dit geval anders. Er wordt nog aan de print en de schakeling gewerkt. We verwachten wel dat hij zal werken, maar we hebben hem niet getest. U kunt voor deze schakeling alle KiCad-files downloaden van de Elektor-pagina of van Elektor's GitHub-repository [12], zodat u ze zelf kunt aanpassen en veranderen. Mocht u ontwerpfouten of domme ontwerpkeuzes tegenkomen (behalve de keuze voor een Raspberry Pi Pico), laat het ons dan alstublieft weten, we staan open voor suggesties. Dat hoeven niet alleen fouten te zijn. Als u iets mist dat u graag zou willen toevoegen, doe dan een suggestie. Zoals ik al zei: dit is een *work in progress*. Uw feedback is welkom.

Schema: deel 1

Het schema bestaat uit twee delen, ook al staan alle componenten op één KiCad-sheet. Het eerste deel omvat de verbindingen tussen de RFM95, onze LoRa-transceiver en een DS18B20-temperatuursensor met de Raspberry Pi Pico. Voor de RFM95 hebben we een SPI-verbinding, die bestaat uit *MISO*, *MOSI*, *SCK* en n*CS*. Daarnaast hebben we nog RESET en DIO0 nodig. In figuur 5 ziet u het schema voor de verbinding tussen de RFM95 en de Raspberry Pi Pico. We hebben ook nog twee condensatoren toegevoegd: 100 nF (C2) en 10µF (C1), naast de RFM95. De condensatoren vangen de stroompieken op als de RFM95 aan het zenden of ontvangen is, zoals aanbevolen in de datasheet.

Verder ziet u R5, een weerstand van 4,7 kΩ op de resetlijn. Normaal is die niet nodig, want de RFM95 heeft een interne pull up-weerstand. Wij hebben deze module al gebruikt in een aantal projecten, en we hebben gemerkt dat een externe pull up ongewenste resets voorkomt.

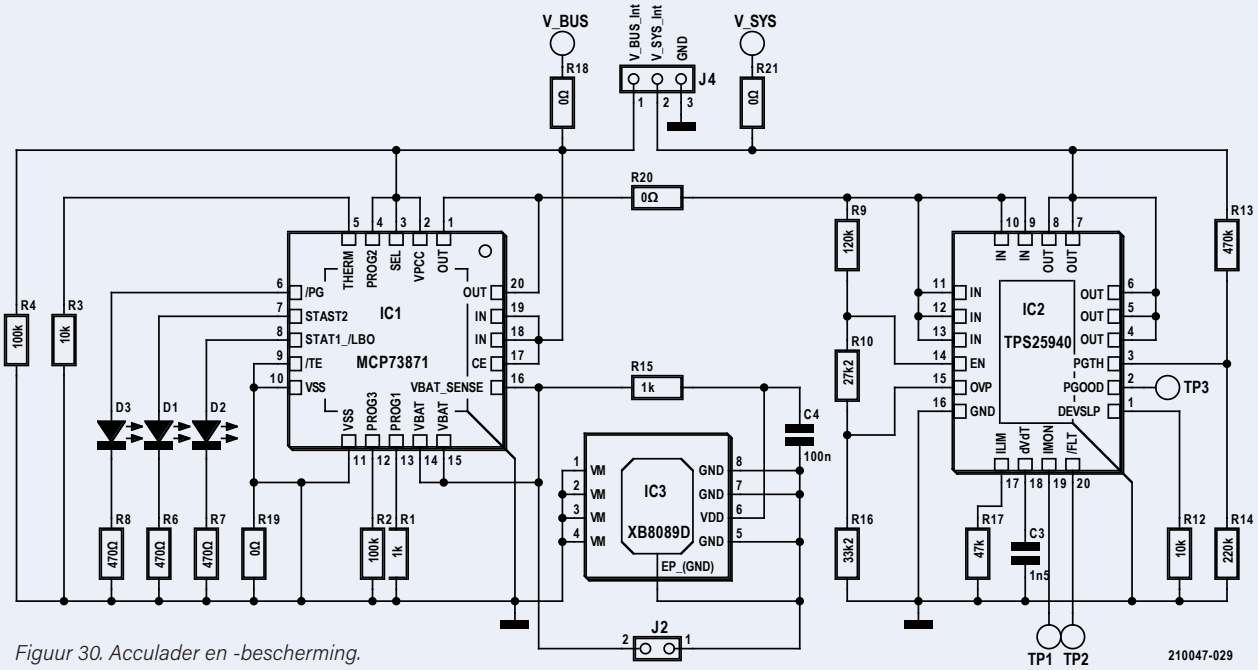
Hoewel de MicroPython-software ze niet gebruikt, zijn ook DIO0, DIO1, DIO2 en DIO3 aanwezig, voor het geval we die later nodig hebben bij C/C++-library's. DIO0 is noodzakelijk om met de module te kunnen werken, de andere DIO-lijnen zijn voor extra functies. Die functies kunnen door andere bibliotheken worden gebruikt.

We voeden de DS18B20 (zie figuur 5) via de VCC-pen. Het is bij One-Wire toegestaan om de sensor niet alleen uit te lezen, maar ook te voeden via de datapen (parasitaire modus), maar dat doen we hier niet. Sommige sensoren werken alleen goed als we ze voeden via de VCC-pen. We gebruiken dus alle drie de aansluitingen, ook de VCC, voor het geval we met een slechte sensor te maken hebben. Dus let op. Als u per ongeluk 'fake' DS18B20-sensoren koopt, werken ze misschien niet goed, of ze werken helemaal niet, als u ze via de datapen probeert te voeden. Voor One-Wire is een pull up-weerstand van 4,7 kΩ nodig. De Raspberry Pi Pico is een module, dus u hoeft niet veel componenten aan te sluiten. Let op de labels V_BUS en V_SYS bij pin 40 en 39. We gebruiken die om een accu aan te sluiten. En dat brengt ons bij deel 2 van het schema.

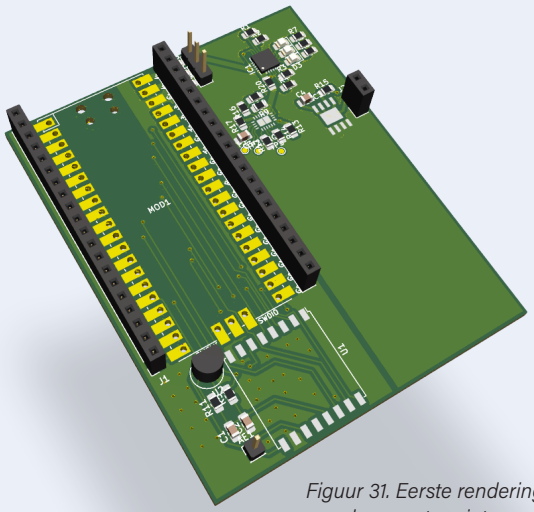
Schema: deel 2

Het tweede deel van het schema is weergegeven in **figuur 30**. Dit bevat het gedeelte voor een oplaadbare batterij. Het bestaat uit een MCP73871-1CC lithiumlader met voedingspad. Als we dus een voeding aansluiten op de input, geeft hij die door naar output, en wordt de accu niet gebruikt. Als de externe voeding wordt afgekoppeld, of als hij te weinig stroom levert, wordt de accu ingezet. De lader is geconfigureerd om de accu te laden met 1000 mA.

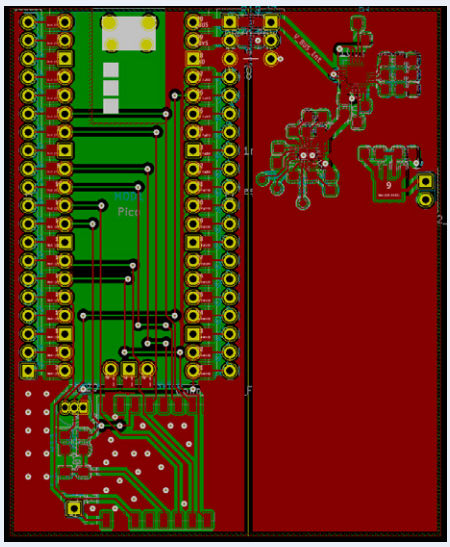
Om de accu te beschermen is er een XB8089D, bekend van de GreatScott! DIY LiPo Supercharger Kit [14]. Die beschermt tegen overladen, te diep ontladen, te grote stroom en verkeerde polariteit. Het laatste onderdeel is een TPS25940 eFuse van Texas Instruments. Die werkt als een ideale diode en voorkomt dat er stroom terug naar de lithiumlader loopt. Ook deze beschermt tegen te grote stroom en te diep ontladen. Overspanning en onderspanning worden bepaald met R9, R10 en R16. De datasheet bevat de formules voor het berekenen van de benodigde waarden. Wij gebruiken 120 kΩ voor R9, 27,2 kΩ voor



Figuur 30. Acculader en -bescherming.



Figuur 31. Eerste rendering van de geroute print.



Figuur 32. Voorbeeld van de layout.

R10 en 33,2 kΩ voor R16 om uit te komen op drempels van 5,37 V voor overspanning en 2,957 V voor onderspanning. Dat zijn veilige waarden, zowel voor de accu als voor de DC/DC-converter op de Raspberry Pi Pico. R17 heeft een waarde van 47 kΩ om de stroom te beperken tot 1,89 A, dat is de maximale stroom die de lithiumlader kan leveren. En waarom gebruiken we geen polyfuse? Lees dit artikel in Elektor [13]. Gebruik van een diode zou een spanningsval van minsten 0,3 V betekenen en dan zouden we energie verspillen.

Iets wat je niet zo vaak ziet op printen zijn R20, R21 en R18. Dat zijn 0Ω-weerstanden, waarmee u bepaalde delen van de lader kunt afkoppelen. Als gedeelten nader onderzocht moeten worden, of helemaal niet werken, kunt u ze afkoppelen en overbruggen, zonder printsporen door te snijden. U hoeft dan alleen maar enkele SMD-componenten te verwijderen.

Snel op de print

Toen het schema klaar was, hebben we snel een voorlopig printontwerp gemaakt. Het verdient geen schoonheidsprijs, maar het is te gebruiken. In **figuur 31** ziet u een eerste #D-aanzicht en **figuur 32** geeft een indruk van de routing. U ziet ook dat de print zó is ontworpen dat het laadgedeelte helemaal kan worden losgenomen en eventueel voor iets anders te gebruiken is. Als het niet werkt, kunt u het ook verwijderen. Eigenlijk zijn het twee printen in één.

En verder...

Hoe het verder gaat, ligt aan de feedback van onze lezers – dat bent u dus. Als dit project u bevalt en u wilt dat we doorgaan en onderdelen toevoegen, of als u suggesties voor verbetering hebt, laat het ons dan weten. Voer een commentaar in, of stuur ons een berichtje. Ook als u andere onderdelen wilt toevoegen, horen we dat graag. U kunt ons ook schrijven over andere interessante projecten.

Tenslotte

Werken met MicroPython op de Raspberry Pi Pico en werken aan een LoRaWAN-project is tamelijk eenvoudig en kan zelfs voor beginners leuk zijn. Maar de stabiliteit van de software kan een probleem vormen. Voor een betrouwbaardere werking verdient C/C++ de voorkeur. Als de MicroPython-scripts werken, vormen ze een snelle en gemakkelijke instap. Wat u wilt meten en verzenden mag u zelf bedenken. En de gebruikte code is eenvoudig genoeg om er met kinderen aan te werken. U hoeft zich niet te beperken tot temperatuurmeting. Denk bijvoorbeeld aan een LoRaWAN-alarmsysteem met wat PIR-detectoren. Of een systeem dat aangeeft wanneer uw planten water nodig hebben. U kunt er al uw creativiteit op loslaten.

Dit is ons eerste project met de Raspberry Pi Pico, maar zeker niet het laatste. Er zit nog meer in de pijplijn. We hopen u te inspireren voor eigen projecten of u wat kennis bij te brengen voor in de toekomst.

210047-03

GERELATEERDE PRODUCTEN

- **Raspberry Pi Pico Microcontroller-Board**
  www.elektor.nl/raspberry-pi-pico-microcontroller-board
- **Breadboard (830 Tie Points)**
  www.elektor.nl/breadboard-830-tie-points
- **Mini Breadboards & Jumper Wires**
  www.elektor.nl/mini-breadboards-jumper-wires
- **SeeedStudio RFM95 Ultra-long LoRa Transceiver Module (868 MHz)**
  www.elektor.nl/rfm95-lora
- **KiCad Like a Pro (2nd Edition)**
  www.elektor.nl/kicad-like-a-pro
- **Dragino LPS8 Indoor LoRaWAN Gateway**
  www.elektor.nl/dragino-lps8-indoor-lorawan-gateway
- **Dragino PG1301 LoRaWAN GPS Concentrator for Raspberry Pi (868 MHz)**
  www.elektor.com/dragino-pg1301
- **Programming with Node-RED (E-book)**
  www.elektor.nl/programming-with-node-red-e-book

Een bijdrage van
Ontwerp en tekst: **Mathias Claußen**
Redactie: **Jens Nickel**
Vertaling: **Evelien Snel**
Layout: **Harmen Heida**

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

WEBLINKS

[1] **Beginnen met LoRaWAN**: www.elektormagazine.nl/magazine/elektor-143/57224
[2] **Projectbestanden op GitHub**: https://github.com/ElektorLabs/210047-LoRa-with-the-Raspberry-Pi-Pico
[3] **Dragino LPS8 Indoor LoRaWAN Gateway**: www.elektor.com/dragino-lps8-indoor-lorawan-gateway
[4] **Dragino PG1301 LoRaWAN GPS Concentrator for Raspberry Pi**: www.elektor.com/dragino-pg1301
[5] **Get Started with MicroPython on Raspberry Pi Pico**: www.elektor.nl/get-started-with-micropython-on-raspberry-pi-pico
[6] **Gratis PDF-download Get Started with MicroPython on Raspberry Pi Pico**: https://hackspace.raspberrypi.org/books/micropython-pico/pdf/download
[7] **GitHub-repository van uLoRa**: https://github.com/fantasticdonkey/uLoRa
[8] **Raspberry Pi Pico MicroPython**: www.raspberrypi.org/documentation/pico/getting-started/
[9] **Thonny IDE download**: https://thonny.org/
[10] **The Things Network v3 Stack**: https://eu1.cloud.thethings.network/console/
[11] **Node-RED-installatiehandleiding**: https://nodered.org/docs/getting-started/raspberrypi
[12] **Elektor GitHub-pagina**: https://github.com/ElektorLabs
[13] **Zelfbouw LiPo-Supercharger-Bundel**: www.elektormagazine.nl/articles/zelfbouw-lipo-superchargerbundel
[14] **GreatScott! DIY LiPo Supercharger Kit**: www.elektormagazine.nl/labs/diy-lipo-supercharger-kit-by-greatscott
[15] **Gerber-bestanden voor RFM95-BOB**: https://github.com/ElektorLabs/191069-RFM95-BOB/