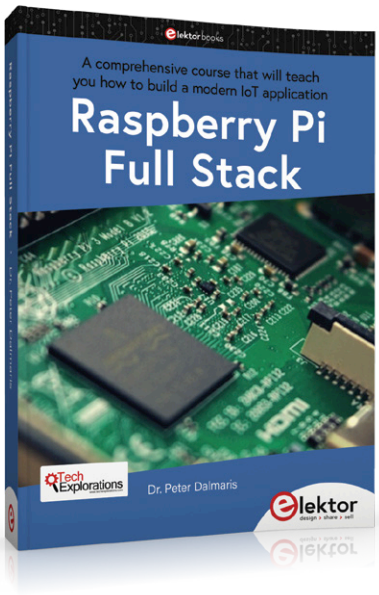
Raspberry Pi Full Stack

RPi en RF24 als hart van een sensornetwerk



Dr. Peter Dalmaris (Australia)

Deze aflevering van *Elektor Boeken* presenteert ongeveer twee hoofdstukken uit het boek *Raspberry Pi Full Stack* van Peter Dalmaris dat onlangs bij Elektor is verschenen. Het boek is bedoeld voor nieuwsgierige RPi-gebruikers die hard- en software willen koppelen terwijl ze 'leren door te doen', en biedt een schat aan educatief materiaal. Hier wordt een 'dagreis' in detail beschreven: de combinatie van een RPi met een RF24-module om waarden van externe sensoren in een klein netwerk uit te lezen.

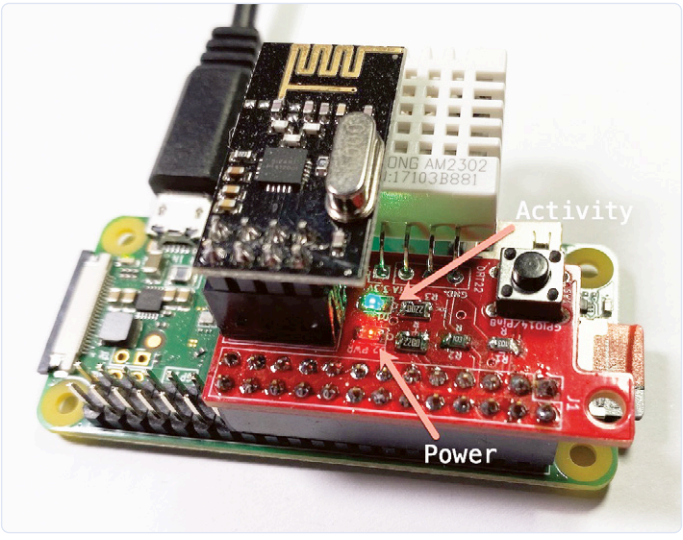
De Raspberry Pi fungeert als de basis-node (knooppunt) van het RF24-netwerk en is verantwoordelijk voor het verwerken van de sensorwaarden die hij ontvangt van het Arduino-knooppunt (of knooppunten, als u er meer dan één hebt). [De Arduino-node is beschreven in eerdere hoofdstukken in het boek die hier niet zijn samengevat – *redactie*.] In dit hoofdstuk laat ik zien hoe u de RF24-module op de Raspberry Pi aansluit. U kunt de bedrading op een breadboard aanbrengen of een speciale HAT gebruiken die ik voor dit doel heb ontworpen.

Ik heb mijn Raspberry Pi RF24-module geïmplementeerd met behulp van de breakout-HAT. U kunt het in actie zien in **figuur 1**.

De HAT biedt plaats aan de nRF24 transceivermodule, de DHT22, een druktoets en twee LED's (aan/uit- en activiteitsindicatie). U kunt de Gerber-bestanden voor de HAT downloaden via de projectpagina (onderdelenlijst) [1]. U kunt de print echter ook direct bij PCBWay [2] bestellen.

Als u er de voorkeur aan geeft deze schakeling op breadboard op te bouwen, raadpleeg dan de tabel met alle verbindingen in **figuur 2** en het schema van de HAT in **figuur 3** . U kunt een high-res versie van het schema downloaden via [3].

Hieronder heb ik alle onderdelen opgesomd die u nodig hebt om de schakeling van figuur 2 op te bouwen. Enkele onderdelen, zoals



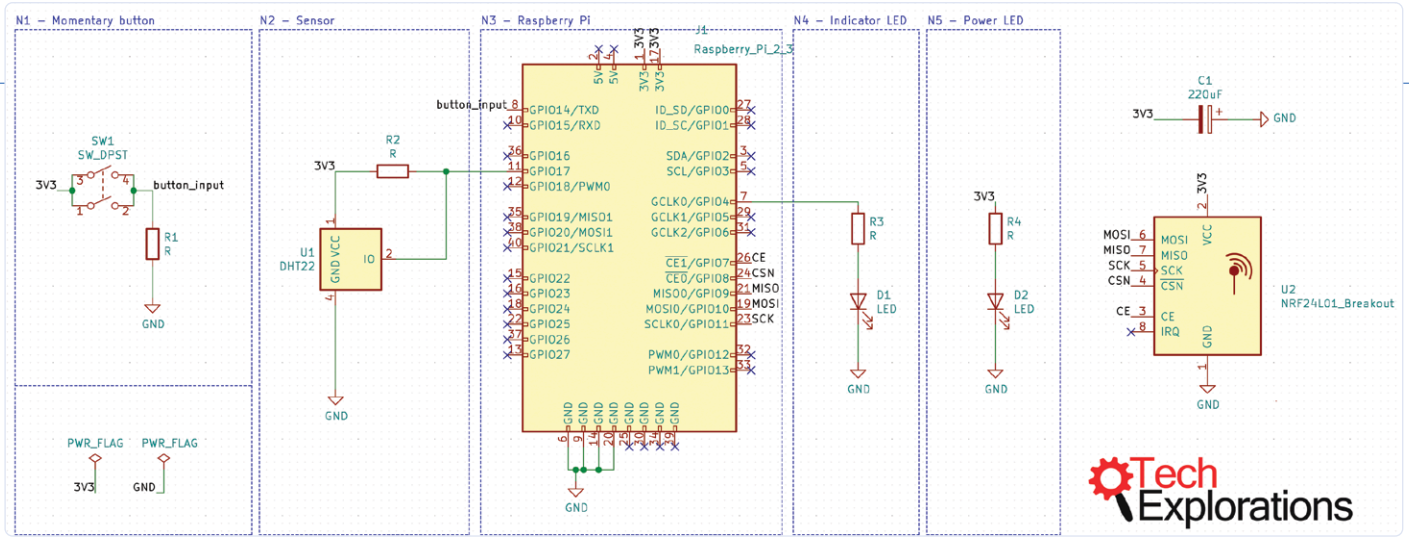
Figuur 1. De Raspberry Pi HAT geïnstalleerd op een Raspberry Pi Zero W.

Raspberry Pi - RF24 - DHT22

Raspberry Pi	nRF24	DHT22	Comments
3.3V	Vcc	Pin 1 - Power	220uF elec. cap pin "+".
GND	GND	Pin 4- GND	220uF elec. cap pin "-".
GPIO 11	SCK		
GPIO 9	MISO		
GPIO 10	MOSI		
GPIO 8	CSN		
GPIO 7	CE		
GPIO4 (Pin 7)		Pin 2 - Data	Add a 10KOhm pull up resistor between Data and GND.

Raspberry Pi Full Stack | Tech Explorations

Figuur 2. De aansluiting van de nRF24- en DHT22-modules.



Figuur 3. Het schema van de Raspberry Pi HAT voor de nRF24- en DHT22-breakoutprint.

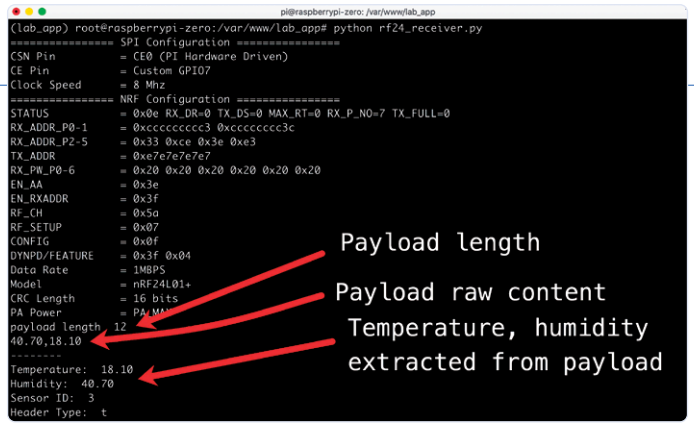
de druktoets en de DHT22-sensor, hebt al bij andere delen van dit project leren kennen. De enige nieuwe componenten zijn eigenlijk de nRF24-transceiver en zijn ontkoppelcondensator.
Benodigde onderdelen:

- 1 DHT22;
- 1 nRF24 breakout-board;
- 2 weerstanden 10 kΩ;
- 2 weerstanden 330 Ω;
- 1 LED rood (aan/uit-indicator);
- 1 LED blauw (activiteitsindicator);
- 1 elektrolytische condensator 220 µF of equivalent.

Het Raspberry Pi nRF24-ontvangerscript

In dit hoofdstuk zal ik de functionaliteit van het Python-ontvangerscript uitleggen dat op de Raspberry Pi draait en de nRF24-communicatie verzorgt. Merk op dat dit script niet 'out of the box' werkt. Het hangt af van de RF24- en RF24Network-drivers in C en de Python-wrappers (in het volgende hoofdstuk laat ik zien hoe u die instelt). Laten we ons voorlopig concentreren op het Python-ontvangerscript. U kunt de volledige broncode van dit script downloaden en bekijken in de projectrepository [4]. Ik heb dit script geschreven zodat het uiteindelijk kan worden uitgevoerd als een achtergrondproces dat wordt bestuurd door `systemd` (vergelijkbaar met de manier waarop u het webtoepassingsscript eerder in dit project al hebt ingesteld). Ik zal u later laten zien hoe u dit moet doen, omdat we er eerst zeker van moeten zijn dat het script correct werkt vanaf de opdrachtregel.
De inhoud van de functie `log_values()` is een bijna perfecte kopie van dezelfde functie in het script env_log.py dat al is geconfigureerd om op basis van een Cron-schema te worden uitgevoerd. Deze functie ontvangt eenvoudig sensorwaarden als parameters en slaat deze op in de lokale database en Google Sheet in de Cloud. Hieronder zal ik een selectie van elementen uit het script opsommen en bespreken, in het bijzonder die welke betrekking hebben op de RPI-nRF24-communicatie.

- Direct na de definitie van de log_values() functie, stelt het script de nRF24 module in. Eerst wordt de variabele radio met behulp van de RF24-constructor geïnitialiseerd. Deze constructor maakt deel uit van de RF24 Python wrapper-bibliotheek waarmee we de RF24 C-driver vanuit ons Python-script kunnen gebruiken. U kunt meer te weten komen over deze functie door de broncode van de driver [5] te bestuderen.
- Zodra het radio-object is gemaakt en geïnitialiseerd, maakt het script het `network`-object aan met behulp van de RF24Network-constructor. Dankzij dit object kan de Raspberry Pi een transmissie ontvangen van de Arduino-node. De enige parameter die nodig is om het netwerkobject te maken, is het radio-object dat we op de regel hiervoor hebben aangemaakt. U kunt meer te weten komen over de RF24Network-constructor in de broncode van RF24Network.h [6], op regel 373.
- Op de volgende regel creëren we met `octlit = lambda n: int (n, 8)` een lambda-functie die we later gebruiken om een decimaal getal om te zetten in een octaal getal. We doen dit omdat de RF24 Network-driver het octale systeem gebruikt voor de knooppuntadressen. In Python is een `lambda`-functie [7] vergelijkbaar met een gewone functie (waar u het `def`-sleutelwoord gebruikt) maar heeft geen naam. Ze zijn handig om dingen te doen zoals het evalueren van een enkele uitdrukking, wat hier het geval is: we geven een decimaal getal door aan `octlit` en de lambda-functie retourneert het octale equivalent met behulp van de Python `int()`-functie [8].
- In `this_node = octlit ("00")` gebruiken we de `octlit`-lambda om het octale equivalent van `00` te krijgen, en slaan we de waarde op in de `this_node`-variabele. Dit is het RF24-netwerkadres van de Raspberry Pi.
- In de volgende zes regels, tot het `while`-blok, zal het script:
  - de RF24-radio starten;
  - 0,1 seconde wachten tot de radio gereed is;
  - het netwerk starten op kanaal 90;
  - de radio- en netwerkconfiguratie naar de console printen;
  - de `packets_sent`-teller op nul resetten;
  - en de `last_sent`-teller op nul zetten.
- Nu de radio en het netwerk klaar zijn, komt het script in een oneindige lus waarin het wacht op een bericht van een Arduino-knooppunt.
- Aan het begin van elke lus roept het de `update()`-functie [9] van het netwerkobject aan. Deze controleert uw berichten.
- Als er geen nieuw bericht is, gaat het script één seconde slapen . Hierna start de lus opnieuw.



Figuur 4. Voorbeeld van de uitvoer van een RF24-ontvangerscript.

- Als er wel een nieuw bericht is, gebruikt het script de functie `read()` [10] om de 12 bytes (de payload) van het bericht te lezen en aan de `payload`-variabele door te geven. De functie haalt ook de header van het bericht op en geeft deze door aan de `header`-variabele. Vergeet niet dat we in de Arduino-sketch een payload van 12 bytes hebben gemaakt die er als volgt uitziet: `51.23,23.09`.
- Op de volgende twee regels gebruik ik `print` om de payload naar de console af te drukken. Dit was handig omdat ik bezig was de payload in getallen te decoderen in het volgende `try`-blok.
- De `payload`-variabele bevat een string die er als volgt uitziet: `51.23,23.09`. Het script gebruikt `decode()` [11] om die om te zetten naar een UTF-8-codering, en de `split()`-functie [12] om de temperatuur- en vochtigheidswaarden in een array van twee subtekenreeksen te schuiven, met een komma als scheidingsteken. De twee waarden worden opgeslagen in de `values`-variabele (een reeks strings).
- In het `try`-blok gebruikt het script de `float()`-functie [13] om de getallen die zijn opgeslagen in de payload-array te extraheren en om te zetten in een drijvende-kommagetal. De functie `float()` ontvangt een tekenreeks en als deze kan worden omgezet in een getal, wordt dat geretourneerd.
- Met `float(values[1][0:5]` neemt het script de eerste vijf karakters van de string die is opgeslagen in index 1 van de `values`-array, en gebruikt de `float()`-functie om die om te zetten in een getal. Dit is de numerieke waarde die is opgeslagen in de `temperature`-variabele.
- Hetzelfde gebeurt met de vochtigheidswaarde met behulp van `float(wvalues[0][0:5])`.
- Als een van deze twee omzettingen mislukt, wordt het `try`-blok afgesloten en wordt er een foutmelding naar de console geprint. Een conversie kan mislukken als de payload-string niet correct is geformatteerd door de Arduino, of als de payload beschadigd raakt tijdens versturen en ontvangen (ten gevolge van een of andere storing).

Neem voldoende tijd om de code te bestuderen, zodat u er vertrouwd mee bent. Als u klaar bent, kopieert u het naar uw toepassingsmap. Gebruik Vim om een nieuw bestand te maken met de naam 'rf24_receiver.py'. Kopieer de code uit de projectrepository [14] in de `Vim`-buffer.
Voordat u de RF24-communicatie kunt testen, moet u de de RF24- en RF24Network C-drivers en Python-wrappers compileren. Dit doet u in het volgende hoofdstuk. Maar eerst wil ik u laten zien hoe het script dat u zojuist hebt bestudeert eruit ziet als het wordt uitgevoerd (**figuur 4**). Ik heb dit screenshot van commentaar voorzien, zodat u de printouts kunt zien die in het script zijn opgenomen. Nu gaan we verder met de setup van de RF24- en RF24Network-drivers. Nadat u de schakeling hebt opgebouwd, wacht het volgende hoofdstuk op u, waar u het Python-script gaat implementeren dat de nRF24-communicatie afhandelt.

Installatie van de Python nRF24-modules op de Raspberry Pi

In dit hoofdstuk laat ik zien hoe u de C-drivers en relevante Python-wrappers voor de nRF24-module compileert en installeert. Er zijn twee modules bij betrokken: de RF24-hoofdmodule en de RF24Network-submodule. Er zijn verschillende *forks* van het RF24-project, maar die welke u gaat gebruiken, wordt onderhouden door TMRh20 [15]. Deze is geoptimaliseerd voor de Raspberry Pi.

U kunt de broncode voor de twee modules ophalen van de betreffende Github-repositories:

- RF24-repository: https://github.com/nRF24/RF24
- RF24Network-repository: https://github.com/nRF24/RF24Network

Voor elke module omvat het installatieproces de volgende stappen:

- download de broncode van de module van Github;
- compileer en installeer de C-driver;
- compileer en installeer de Python-wrapper.

Controleer voordat u begint of de SPI-interface op uw Raspberry Pi is ingeschakeld. De nRF24-module gebruikt SPI om te communiceren met de Raspberry Pi. Om dit te controleren (en SPI in te schakelen als dit nog niet is gebeurd), logt u in op uw Raspberry Pi en typt u op de opdrachtregel:

```
sudo raspi-config
```

Gebruik de pijltjestoetsen en de Enter-toets om naar Interfacing Options te gaan en vervolgens naar SPI. Schakel indien nodig SPI in en sluit raspi-config. Ga verder met het bijwerken van Raspbian:

```
sudo apt-get update
sudo apt-get upgrade
```

U downloadt de broncode van de modules in uw toepassingsmap. U wilt ook de Python-wrappers compileren terwijl de virtuele Python-omgeving van uw applicatie geactiveerd is, zodat ze beschikbaar zijn voor de Python-scripts van uw applicatie.

Bereid uw werk voor met deze opdrachten:

```
$ sudo su
# cd /var/www/lab_app/
# . bin/activate
# mkdir rf24libs
# cd rf24libs
```

U bent nu klaar om de twee modules te compileren en te installeren.

RF24 compilatie en installatie

Maak een kloon van de RF24-broncode, van de git op https://github.com/tmrh20/RF24.

https://github.com/tmrh20/RF24.

```
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs# git clone
https://github.com/tmrh20/RF24.git RF24
Cloning into 'RF24'...
remote: Enumerating objects: 46, done.
remote: Counting objects: 100% (46/46), done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 3545 (delta 16), reused 15 (delta 4),
   pack-reused 3499
Receiving objects: 100% (3545/3545), 1.54 MiB |
   679.00 KiB/s, done.
Resolving deltas: 100% (2119/2119), done.
```

In de broncodemap staat nu een nieuwe map 'RF24'..

```
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs# ls -al
total 12
drwxr-xr-x 3 root root 4096 May 13 01:38 .
drwxr-xr-x 14 root root 4096 May 13 01:39 ..
drwxr-xr-x 9 root root 4096 May 13 01:38 RF24
```

Ga naar de RF24-directory en installeer de driver:

```
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs# cd RF24/
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs/RF24# make install
[Running configure]
[SECTION] Detecting arm compilation environment.
[OK] arm-linux-gnueabihf-gcc detected.
[OK] arm-linux-gnueabihf-g++ detected.
… (omitted output)…
[Installing Libs to /usr/local/lib]
[Installing Headers to /usr/local/include/RF24]
```

De C-driver is nu geïnstalleerd. Nu is de Python-wrapper aan de beurt. Dit kan enkele minuten duren op de tragere Raspberry Pi Zero W.

```
(lab_app) root@rpi4:/var/www/lab_app/rf24libs/RF24/
   pyRF24# cd pyRF24/
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs/RF24/pyRF24#
python setup.py install
running install
running bdist_egg
running egg_info
creating RF24.egg-info
… (omitted output)…
Installed /var/www/lab_app/lib/python3.8/
   site-packages/RF24-1.3.4-py3.8-
linux-armv6l.egg
Processing dependencies for RF24==1.3.4
Finished processing dependencies for RF24==1.3.4
```

De RF24-module is nu geïnstalleerd en de Python-wrapper is klaar voor gebruik. Ga verder met het RF24Network. .

```
RF24Network
```

Ga terug naar de root van de rf23libs-directory en maak een kloon van de RF24Network-broncode (van https://github.com/nRF24/RF24Network.git).

```
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs/RF24/pyRF24# cd
../..
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs# git clone
https://github.com/nRF24/RF24Network.git
Cloning into 'RF24Network'...
remote: Enumerating objects: 2249, done.
remote: Total 2249 (delta 0), reused 0 (delta 0),
   pack-reused 2249
Receiving objects: 100% (2249/2249), 1.60 MiB |
   733.00 KiB/s, done.
Resolving deltas: 100% (1342/1342), done.
```

Door het klonen is een nieuwe directory aangemaakt: 'RF24Network'. Ga naar die nieuwe directory en compileer de broncode:

```
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs# cd RF24Network/
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs/RF24Network# make
install
g++ -Wall -fPIC -c RF24Network.cpp
g++ -shared -Wl,-soname,librf24network.so.1 -o
   librf24network.so.1.0
RF24Network.o -lrf24-bcm
[Install]
[Installing Headers]
```

De driver is geïnstalleerd. We gaan verder met de Python-wrapper voor RF24Network.

```
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs/RF24Network# cd
../RF24/pyRF24/pyRF24Network/
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs/RF24/pyRF24/
pyRF24Network# python setup.py install
running install
running build
running build_ext
building 'RF24Network' extension
… (omitted output)…
running install_egg_info
Removing /var/www/lab_app/lib/python3.8/
   site-packages/
RF24Network-1.0-py3.8.egg-info
Writing /var/www/lab_app/lib/python3.8/site-packages/
RF24Network-1.0-py3.8.egg-info
```

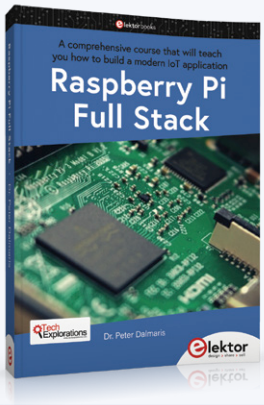
De Python-wrapper RF24Network is nu geïnstalleerd; we kunnen die nu gaan testen.

RASPBERRY PI FULL STACK

Raspberry Pi Full Stack begeleidt de lezer bij het ontwikkelen van full-stack webapplicaties met de Raspberry Pi. Naast het van de grond af bouwen van een applicatie, doet u ervaring en kennis op van verschillende technologieën, zoals:

- het Linux-besturingssysteem en de opdrachtregel;
- de programmeertaal Python;
- de General Purpose Input/Output-pinnen (GPIO's) van de Raspberry Pi;
- de Nginx-webserver;
- het Flask Python webapplicatie-microframework;
- JQuery en CSS voor het maken van gebruikersinterfaces;
- omgang met tijdzones;
- grafieken maken met Plotly en Google Charts;
- datalogging met Google Sheet;
- applets ontwikkelen met IFTTT;
- uw applicatie beveiligen met SSL;
- SMS-meldingen op uw telefoon ontvangen met Twilio.

Dit boek leert u ook hoe u een remote draadloos Arduino-sensorknooppunt bouwt en de gegevens daarvan verzamelt. Uw Raspberry Pi-webtoepassing kan Arduino-node data op dezelfde manier verwerken als data van de ingebouwde sensor.
Raspberry Pi Full Stack leert u veel vaardigheden die essentieel zijn voor het bouwen van web- en IoT-applicaties. De applicatie die u in dit project bouwt, is een platform dat u naar wens kunt uitbreiden. En dat is nog maar het begin van wat u kunt doen met een Raspberry Pi en de software- en hardwarecomponenten waarover u meer leert. Dit boek wordt ondersteund door de auteur via een speciale discussieruimte.



- **Engelstalig boek:** www.elektor.com/raspberry-pi-full-stack
- **Engelstalig e-boek:** www.elektor.com/raspberry-pi-full-stack-e-book

Testen

De eenvoudigste manier om te testen of de RF24- en RF24Network-modules correct werken, is door de Python-voorbeeldprogramma's uit te voeren die met de broncode worden meegeleverd. U vindt deze voorbeelden in de map `examples` van `pyRF24Network`.

Voer het 'rx'-voorbeeld als volgt uit:

```
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs/RF24/pyRF24/
pyRF24Network# cd examples/
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs/RF24/pyRF24/
pyRF24Network/examples# ls
helloworld_rx.py helloworld_tx.py
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs/RF24/pyRF24/
pyRF24Network/examples# python helloworld_rx.py
================ SPI Configuration ================
CSN Pin = CE0 (PI Hardware Driven)
CE Pin = Custom GPIO22
Clock Speed = 8 Mhz
================ NRF Configuration ================
```

WEBLINKS

[1] **Gerber-bestanden voor de RF24 HAT**: https://techexplorations.com/parts/rpifs-parts/
[2] **Kale print van de HAT (PCBWay)**: https://www.pcbway.com/project/shareproject/Raspberry_Pi_Full_Stack_RF24_and_DHT22_HAT.html
[3] **High-res versie van het schema**: https://techexplorations.com/parts/rpifs-parts/
[4] **rf24_receiver.py script-broncode**: https://github.com/futureshocked/RaspberryPiFullStack_Raspbian/blob/master/rf24_receiver.py
[5] **Zie regel 137 van RF24.h**: https://github.com/nRF24/RF24/blob/master/RF24.h#L137
[6] **Zie regel 373 van RF24Network.h**: https://github.com/nRF24/RF24Network/blob/master/RF24Network.h#L373
[7] **Meer informatie over 'lambda-code'**: https://docs.python.org/3/tutorial/controlflow.html#lambda-expressions
[8] **Meer informatie over 'int ()'**: https://docs.python.org/3/library/functions.html#int
[9] **Meer informatie op**: https://github.com/nRF24/RF24Network/blob/master/RF24Network.h#L413
[10] **Meer informatie op**: https://github.com/nRF24/RF24Network/blob/master/RF24Network.h#L467
[11] Meer over 'decode()': https://docs.python.org/3/library/stdtypes.html#bytes.decode
[12] **Meer over 'split()'**: https://docs.python.org/3/library/stdtypes.html#str.split
[13] **Meer over 'float()'**: https://docs.python.org/3/library/functions.html#float
[14] **Nieuwste versie van rf24_receiver.py**: https://github.com/futureshocked/RaspberryPiFullStack_Raspbian/blob/master/rf24_receiver.py
[15] **De 'thuisbasis' van het project**: https://tmrh20.github.io/RF24/

```
STATUS = 0x0e RX_DR=0 TX_DS=0 MAX_RT=0 RX_P_NO=7
   TX_FULL=0
RX_ADDR_P0-1 = 0xcccccccccc3 0xcccccccc3c
RX_ADDR_P2-5 = 0x33 0xce 0x3e 0xe3
TX_ADDR = 0xe7e7e7e7e7
RX_PW_P0-6 = 0x20 0x20 0x20 0x20 0x20 0x20
EN_AA = 0x3e
EN_RXADDR = 0x3f
RF_CH = 0x5a
RF_SETUP = 0x07
CONFIG = 0x0f
DYNPD/FEATURE = 0x3f 0x04
Data Rate = 1MBPS
Model = nRF24L01+
CRC Length = 16 bits
PA Power = PA_MAX
```

Merk op dat de uitvoer statistieken bevat van de RF24-module tijdens het opstarten. Het bevat geldige waarden voor de RX, een adres (dat wil zeggen waarden die ongelijk nul zijn) en de rest van de NRF-configuratie. Als de RF24- en RF24Network-modules niet correct waren geïnstalleerd, of als de Raspberry Pi niet via SPI kon communiceren met de nRF24-module, zou u ofwel een foutmelding krijgen van het voorbeeldprogramma (het zou helemaal niet starten), ofwel de configuratiewaarden zou blanco zijn. Nu de RF24-modules zijn geïnstalleerd, kan het Python-script dat u hebt geïnstalleerd werken.

200517-04

Over de auteur

Dr. Peter Dalmaris is leraar, elektrotechnisch ingenieur en maker; daarnaast produceert hij online videocursussen over doe-het-zelf-elektronica en is hij de auteur van meerdere technische boeken. Sinds 2013 Chief Tech Explorer bij Tech Explorations, het bedrijf dat hij in Sydney, Australië oprichtte, is het zijn missie om technologie te verkennen en de wereld daarover te onderrichten.

Een bijdrage van

Tekst en illustraties: **dr. Peter Dalmaris**
Redactie: **Jan Buiting**
Vertaling: **Eric Bogers**
Layout: **Giel Dols**

Vragen of opmerkingen?

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