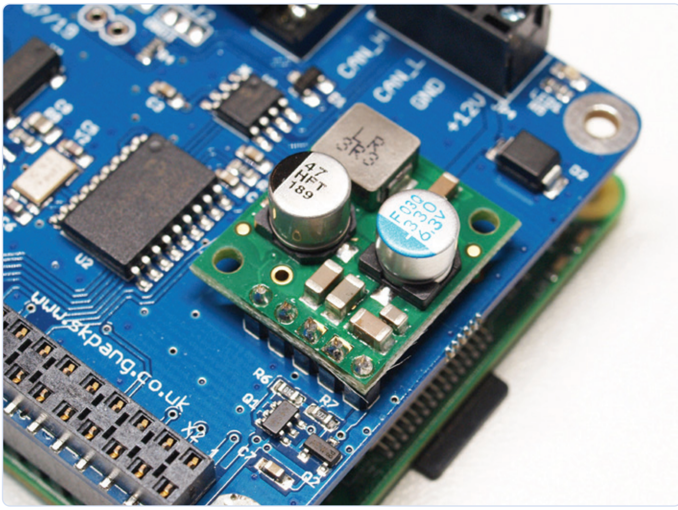
Yes We CAN met PiCAN 3

Een CAN Bus HAT voor de Raspberry Pi 4

door **Tam Hanna** (Slowakije)

Een mobiele CAN-bus unit is handig om communicatie te analyseren, problemen te diagnosticeren en te controleren. De PiCAN HAT die hier wordt besproken, kan deze klus doen en hij kan worden aangesloten op de Raspberry Pi 4. Laten we het eens nader bekijken.



Figuur 1: Vijf pinnen met een onderlinge afstand van 0,1 inch verbinden de DC/DC converter board (groen) met het mainboard.

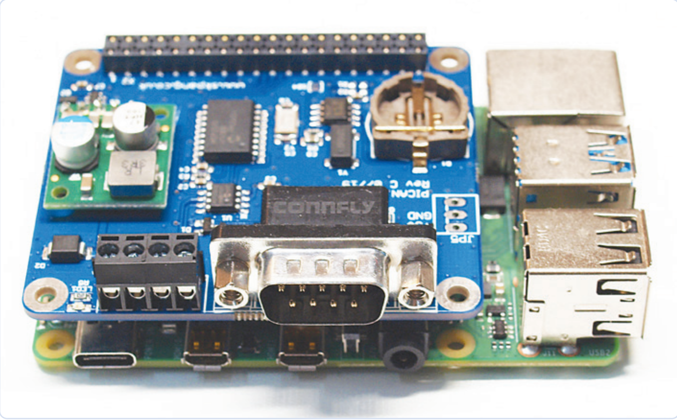
Sinds Bosch in 1986 de Controller Area Network (CAN) busspecificatie registreerde en Mercedes Benz deze voor het eerst gebruikte in zijn S-Klasse-model in 1991, is er veel gebeurd op dit gebied. Inmiddels is CAN een integraal onderdeel geworden van ontwikkeling en ontwerp van auto's. De CAN-bus is robuust en daardoor bruikbaar voor vele andere toepassingsgebieden. Een mobiele CAN-bus unit is handig om communicatie te analyseren, problemen te diagnosticeren en te controleren. De PiCAN HAT die hier wordt besproken, kan deze klus klaren, en hij kan worden aangesloten op de nieuwste versie van de Raspberry Pi. Laten we het eens nader bekijken. Elektrische systemen van voertuigen zijn vijandige omgevingen voor kwetsbare elektronische apparaten. Ik herinner me veel van mijn pogingen om fouten op te sporen, waarbij ik 40 m lange kabel legde om een oscilloscoop van stroom te voorzien om apparatuur in een auto te testen. Als u puur geïnteresseerd bent in het analyseren van CAN-busgegevens, biedt de PiCAN 3 een aantrekkelijk en draagbaar diagnostisch alternatief.

Waarom een Raspberry Pi gebruiken?

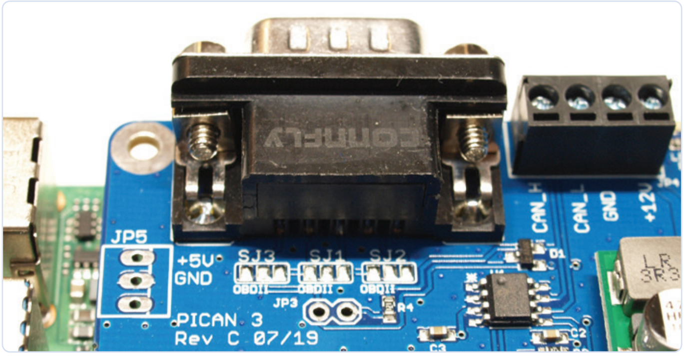
Wanneer je experimenteert met communicatie via bussen zoals I²C, SPI, enz., kunnen er problemen optreden die niet het gevolg zijn van een slechte signaalkwaliteit op de fysieke laag, maar bij het organiseren en rangschikken van de verzonden gegevens zelf. Als je bijvoorbeeld verkeerd samengestelde data-pakketten verstuurt of in de war raakt hoe de bytes moeten worden gerangschikt, moet je je niet verbazen als de motor of ander apparaat dat je hoopte te besturen, weigert mee te werken.
Tools zoals PiCAN 3 in combinatie met een Raspberry Pi maken mobiele metingen metingen. Idealiter is een Raspberry Pi 4 met een kleine monitor alles dat nodig is om een draagbare testopstelling te bouwen. Als u de ontwikkelomgeving van het CAN-systeem integreert in de ARMbian-distributie van het systeem, bent u klaar om aan de gang te gaan.

De hardware-functies

De elektrische omgeving in een auto kan behoorlijk agressief zijn voor elektronische circuits. De fabrikant SK Pang heeft hier rekening mee gehouden in het ontwerp van het board door een DC/DC converter voeding op te nemen (**Figuur 1**) met een breed ingangsspanningsbereik van 6 VDC tot 20 VDC die via een vijf-pins header op het board wordt aangesloten. De PiCAN HAT zou beschikbaar moeten zijn zonder de DC/DC converter voeding, maar momenteel is dit niet het geval en is de ingebouwde DC/DC converter standaard.
Dit nieuwste CAN-shield wordt beschreven als het "PiCAN 3 - CAN-Bus Board voor Raspberry Pi 4 met 3 A DC/DC converter & RTC" om het te onderscheiden van zijn voorgangers. Deze nieuwste versie bevat een 3 A DC/DC converter voeding om de energieverslindende Raspberry Pi 4 te laten werken. Het shield kan ook rechtstreeks vanaf de Raspberry Pi worden gebruikt wanneer het wordt gevoed via de USB-C poort. Het schema kan worden bekeken op [1] en laat zien dat in het hele ontwerp standaard IC's worden gebruikt. De gebruikershandleiding is beschikbaar op [2].
De CAN-signaalbusinterface wordt gevormd met de MCP2515, die via SPI en een GPIO-interruptpin met de Raspberry Pi communiceert. De fysieke interface met de bus wordt geleverd via het MCP2562 CAN-bus-transceiver IC. Het PiCAN 3 schakelschema laat zien dat de 3,3 V-voeding die door sommige componenten op het board wordt gebruikt, afkomstig is van een regelaar op de Raspberry Pi, waardoor een extra regelaar wordt bespaard.
Het board heeft twee connectoren (**Figuur 2**) om de PiCAN 3 aan te sluiten op een externe CAN-bus. De eerste is een vierpolig



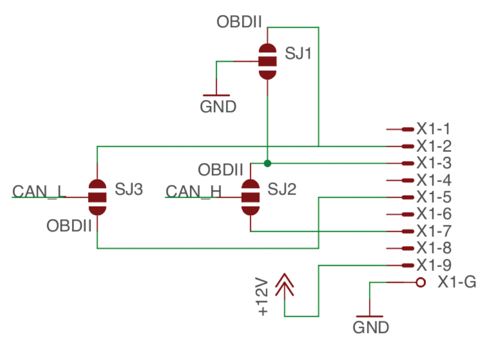
Figuur 2: Het PiCAN 3-board biedt twee manieren om verbinding te maken met een CAN-bus.



Figuur 3: Grote soldeervlakken zorgen ervoor dat het niet al te lastig is om de signalen te routen.

aansluitblok met signaalidentificatie op het board gedrukt. De tweede methode maakt gebruik van een 9-pins sub-D-connector, die veel lezers nog zullen herkennen als de connector die wordt gebruikt voor standaard RS232-communicatie.

De 9-pins sub-D-connector accepteert een standaard OBD-II naar DB9-kabel om verbinding te maken met een OBD-systeem. Helaas is er geen standaard toewijzing van de connectorpinnen en CAN-signalen. Om te zorgen voor verschillende bedradings-mogelijkheden, moeten pads op het board (**Figuur 3**) worden overbrugd met soldeer. Een toewijzingsplan (**Figuur 4**) helpt u te beslissen welke pads u wilt koppelen, zodat de connector compatibel is met het systeem waarmee u verbinding maakt. Deze pads zijn bij levering volledig open en zonder de soldeerbruggen worden de signalen niet naar de juiste pinnen geleid.



	CAN ANALYSER	OBDII	SOLDER-JOINT
CAN H	7	3	SJ2
CAN L	2	5	SJ3
GND	3	2	SJ1

Figuur 4: De tabel laat zien hoe de soldeerpads moeten worden overbrugd.

Automotive toepassingen vereisen vaak nauwkeurige datum- en tijdinformatie. Hiervoor wordt in de PiCAN 3 een PCF8523 real-time clock chip gebruikt, die communiceert met de Raspberry Pi via een I²C-interface. Op de print is een houder voor een knoopcelbatterij, die geschikt voor een CR1220-type knoopcel om de real-time klok van stroom te voorzien.

Het CAN-ecosysteem

Het Linux-besturingssysteem heeft een zekere relevantie in de automobielsector, in feite is er een heel ecosysteem in de Linux-wereld om CAN-bustoepassingen te ondersteunen. De beschikbare hulpprogramma's variëren van kernel-drivers tot command-line programma's, samen met vele andere handige tools.

Om te beginnen met communiceren met de PiCAN 3, moeten we enkele aanpassingen maken aan de */boot/config.txt* file van een nieuwe Raspbian-installatie. De SPI-bus moet eerst worden geactiveerd door het volgende blok (let op de extra Overlay-call):

```
dtparam=spi=on
dtoverlay=mcp2515-can0,oscillator=16000000,interrupt=25
dtoverlay=spi-bcm2835-overlay
```

De volgende instructies zijn vereist om de real-time klok te gebruiken:

```
dtparam=i2c_arm=on
dtoverlay=i2c-rtc,pcf8523
```

De volgende stap is het downloaden van kernel-modules en enkele andere hulpprogramma's. Gelukkig is er ook een compleet, kant-en-klaar pakket beschikbaar:

```
sudo apt-get install can-utils
```

Na de inzet van de CAN-hulpprogramma's moet de interface worden geregistreerd bij het besturingssysteem. Hiervoor is het voldoende om een nieuwe interface te genereren volgens onderstaand schema. De waarde 500.000 geeft hier de maximale databitrate aan die door de hardware wordt ondersteund:

```
sudo /sbin/ip link set can0 up type can bitrate 500000
```

Zodra de interface gereed is, kun je deze op dezelfde manier gebruiken als vergelijkbare producten van andere fabrikanten die kunnen worden gebruikt om te communiceren via de CAN-bus. Een klassieke toepassing is het gebruik van *candump*. Deze tool kan op de command-line worden geactiveerd door de volgende opdracht in te voeren:

```
candump
```

Eenmaal actief, toont het automatisch en continu alle CAN-informatie die zichtbaar is voor het shield. Dit is met name handig voor reverse-engineering van onbekende motorbesturingen in bestaande systemen.

Er is ook een Python CAN API die kan worden geïnstalleerd:

```
git clone https://github.com/hardbyte/python-can
cd python-can
sudo python3 setup.py install
```

De real-time clock toevoegen

Het Linux-besturingssysteem heeft al eeuwenlang real-time klokken op laptops en pc's geïmplementeerd. Systemen die niet zijn uitgerust met een hardware-RTC (vanwege de kosten) maken gebruik van de *fake-hwclock* emulatormodule in het besturingssysteem om informatie over de tijd te geven; dit is ook het geval met de Raspberry Pi.
Om de hardware real-time klok op het CAN-board te kunnen gebruiken, moeten we eerst deze "nep" hwclock uitschakelen om te voorkomen dat deze de tijd verstoort die wordt geleverd door de "echte" hwclock:

```
sudo apt-get -y remove fake-hwclock
sudo update-rc.d -f fake-hwclock remove
sudo systemctl disable fake-hwclock
```

Vervolgens moeten we het bestand */lib/udev/hwclock-set* openen (hiervoor heb je normaal gesproken Superuser-rechten nodig) en de regels code in deze twee blokken uitschakelen:

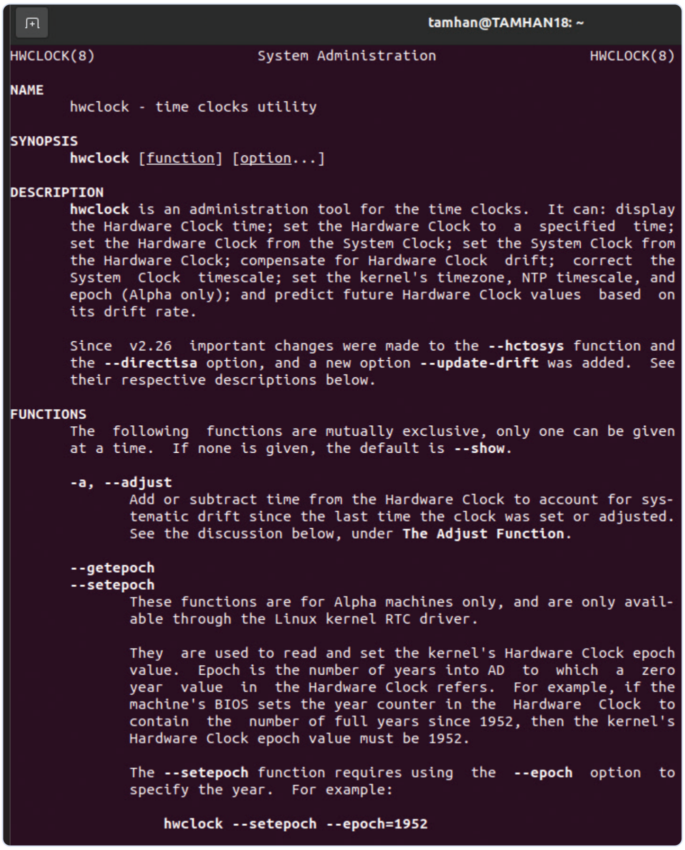
```
#if [ -e /run/systemd/system ] ; then
# exit 0
#fi

#/sbin/hwclock --rtc=$dev --systz --badyear
#/sbin/hwclock --rtc=$dev --systz
```

Tijdinformatie van de hardware-klok kan nu worden uitgelezen met *hwclock*. De kenmerken van deze beheertool voor de hardware-klok zijn weergegeven in **figuur 5**.

CAN-buscommunicatie en -besturing

De PiCAN 3 HAT van SK Pang met zijn on-board 3 A DC/DC converter is nu geschikt voor gebruik met een Raspberry Pi 4 om CAN-bus communicatie- en besturingsmogelijkheden te bieden. Samen met de Raspberry Pi vormt het een compact en relatief goedkoop experimenteel mobiel platform, compleet met de benodigde interface om direct op een CAN-bus aan te sluiten. De hardware is open-source, zodat het systeem in uw eigen ontwerpen kan worden geïntegreerd zodra het noodzakelijke evaluatieproces is voltooid.

210303-03



```
tamhan@TAMHAN18: ~
HWCLOCK(8)                 System Administration                 HWCLOCK(8)

NAME
       hwclock - time clocks utility

SYNOPSIS
       hwclock [function] [option...]

DESCRIPTION
       hwclock is an administration tool for the time clocks.  It can: display
       the Hardware Clock time; set the Hardware Clock to  a  specified  time;
       set the Hardware Clock from the System Clock; set the System Clock from
       the Hardware Clock; compensate for Hardware Clock  drift;  correct  the
       System  Clock  timescale; set the kernel's timezone, NTP timescale, and
       epoch (Alpha only); and predict future Hardware Clock values  based  on
       its drift rate.

       Since  v2.26  important changes were made to the --hctosys function and
       the --directisa option, and a new option --update-drift was added.  See
       their respective descriptions below.

FUNCTIONS
       The  following  functions are mutually exclusive, only one can be given
       at a time.  If none is given, the default is --show.

       -a, --adjust
              Add or subtract time from the Hardware Clock to account for sys-
              tematic drift since the last time the clock was set or adjusted.
              See the discussion below, under The Adjust Function.

       --getepoch
       --setepoch
              These functions are for Alpha machines only, and are only avail-
              able through the Linux kernel RTC driver.

              They  are used to read and set the kernel's Hardware Clock epoch
              value.  Epoch is the number of years into AD  to  which  a  zero
              year  value  in  the Hardware Clock refers.  For example, if the
              machine's BIOS sets the year counter in the  Hardware  Clock  to
              contain  the  number of full years since 1952, then the kernel's
              Hardware Clock epoch value must be 1952.

              The --setepoch function requires using  the  --epoch  option  to
              specify the year.  For example:

                  hwclock --setepoch --epoch=1952
```

Figuur 5: Gebruik het `man` (afkorting voor handmatig) commando voor meer informatie over toegang via de command-line tot de hardware-RTC. Dit is het resultaat voor `hwclock`, die informatie geeft om de RTC te besturen die door het CAN-Board wordt gebruikt.

Vragen of opmerkingen?
Als je technische vragen of opmerkingen hebt over dit artikel, neem dan contact op met de auteur via:
tamhan@tamoggemon.com of
het Elektor-team bij editor@elektor.com.

Een bijdrage van:
Tekst en afbeeldingen: **Tam Hanna** Layout: **Giel Dols**
Redacteur: **Dr. Thomas Scherer** Vertaling: **Hans Adams**

WEBLINKS

[1] **PiCAN-3 schema printplaat:** https://cdn.shopify.com/s/files/1/0563/2029/5107/files/pican3_rev_C.pdf?v=1619981690

[2] **PiCAN3 Gebruikershandleiding:** https://cdn.shopify.com/s/files/1/0563/2029/5107/files/PICAN3_UGA_10.pdf?v=1619981615