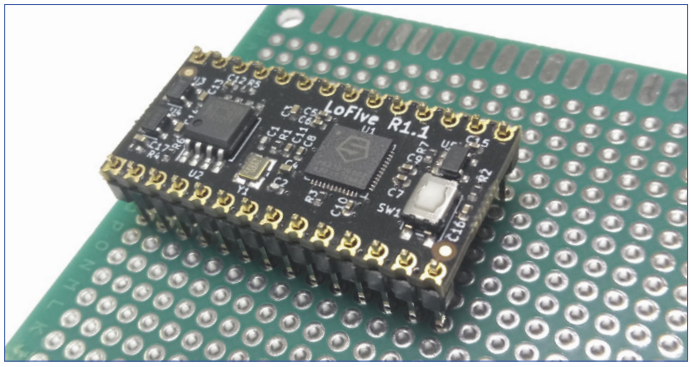
Eerste stappen met RISC-V

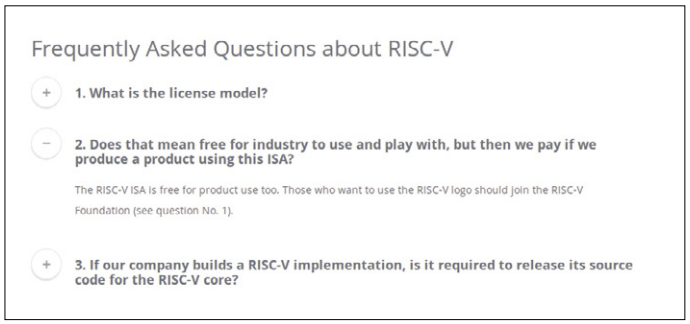
het LoFive-board onder de loep

Tam Hanna (Slowakije)

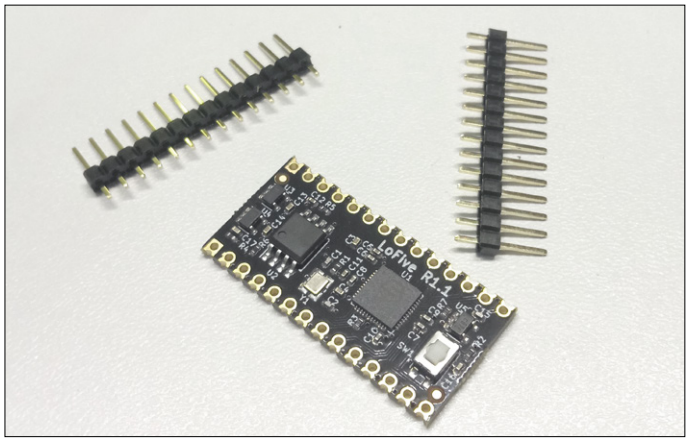
Na ARM zou RISC-V “the next big thing” op de processormarkt kunnen worden. De architectuur van de instructieset, die wordt gepropageerd door een groot consortium [1], is licentievrij en is geschikt voor zowel embedded als computerprocessoren. Op het moment zijn er echter maar een paar microcontrollers en -boards te koop. Elektor-auteur Tam Hanna heeft het goedkope ‘LoFive’-board aan de tand gevoeld.



Terwijl de ARM-architectuur als geheel zwaar gepatenteerd is, zorgt de steeds doorgaande ontwikkeling van de x86-architectuur er voor dat aanbieders van een recente x86-core ook aardig wat voor de licenties moet neertellen.
Ontwikkeld aan de Universiteit van Berkeley, maakt de RISC-V-architectuur expliciet zowel de embedded markt als de markt voor ‘grotere’ processoren tot zijn jachtgebied. Verreweg het belangrijkste argument van de aanbieders is de licentievrijheid – **figuur 1** komt van de website van de RISC-V Foundation [2] en laat zien dat de architectuur gratis gebruikt mag worden.



Figuur 1. De RISC-V-architectuur kan gratis worden gebruikt en brengt geen irritante licentieproblematiek met zich mee.



Figuur 2. Het board kan ook als surface-mount module worden gebruikt.

Terwijl men al jaren aan de RISC-V-architectuur werkt, is tastbare hardware pas kort beschikbaar. Via GroupGets is een goedkoop evaluatieboard met de naam LoFive verkrijgbaar dat een RISC-V-processor Freedom E310 van de fabrikant SiFive bevat [3].
Door de snelle evolutie van de architectuur zijn er inmiddels twee versies van – we gebruiken hier de meer recente variant die R1 of 1.1 heet. Let op: de oudere versie wijkt in technisch opzicht sterk af van de opvolger, en de hier besproken tips zijn daar niet zonder meer bruikbaar.

Voorbereidende schermutselingen

Wie de LoFive uitpakt, ziet – zoals **figuur 2** laat zien – enkele losse headers. De fabrikant gebruikt ‘combinatorische’ pinouts, die naast het insolderen van 2,54mm-headers ook direct solderen van de print op een moederbord mogelijk maken. De auteur gebruikt hierna voor het gemak een klassiek breadboard.
Het LoFive-board is momenteel verkrijgbaar voor ongeveer € 25, dit is onder andere mogelijk omdat de aanbieder geen programmeerchip heeft geïntegreerd. In plaats daarvan moet u een FTDI FT2232H-56Q USB/serieel-adapterprintje van FTDI aanschaffen; de correcte verbindingen zijn samengevat in **tabel 1**.
Iedereen die zich serieus met de LoFive wil bezighouden, zou nu een ‘dragerprintje’ moeten maken. Voor eerste experimenten volstaan jumperkabeltjes echter (**figuur 3**). In het in EMI-opzicht dankzij veel LED-lampen nogal onrustige laboratorium van de auteur gaven kabeltjes van 10 cm meestal geen problemen; en als er iets mis ging was dat bij de volgende doorloop weer verdwenen.

Het ontwikkelen begint

Op het werkstation van de auteur met Ubuntu 18.04 waren de meeste vereiste elementen al voorhanden. Voer de volgende opdracht in om de pakketten te laden:

Tabel 1.
Aansluiting van LoFive en programmeeradapter.

LoFive-pin	FTDI Breakout-pin
+5Vin	VBS
GND	GND
TRSTN	AD5
TCK	AD0
TDO	AD2
TMS	AD3
TDI	AD1
UART0.TX	BD1
UART0.RX	BD0

```
sudo apt-get install autoconf automake libmpc-dev
   libmpfr-dev libgmp-dev gawk bison flex texinfo
   libtool libusb-1.0-0-dev make g ++ pkg-config
   libexpat1-dev zlib1g-dev
```

Overigens staan gebruikers van Windows en Mac OS op dit moment in de kou. Linux is min of meer de standaard geworden in het RISC-V-wereldje; vooral als u met minder gangbare boards werkt, hebt u weinig kans met Windows. De op Eclipse gebaseerde Freedom Studio, ontwikkeld door de processorfabrikant SiFive, werkt met het 'officiële' HiFive-ontwikkelbord. De ondersteuning van de LoFive in deze IDE is op het moment van schrijven geen succes.
In de volgende stap kunt u in elk geval het downloaden van de SDK via de bekende git-client starten:

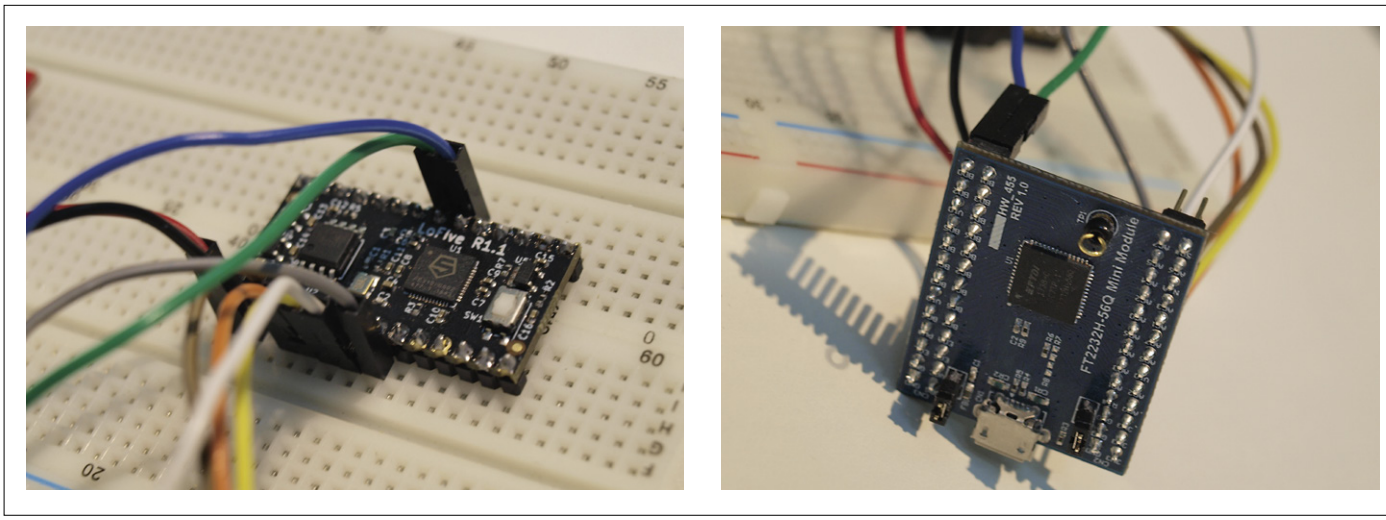
```
tamhan@TAMHAN18:~$ cd riscvneu/
tamhan@TAMHAN18:~/riscvneu$ git clone --recursive
   https://github.com/mwelling/freedom-e-sdk.git
. . .
tamhan@TAMHAN18:~/riscvneu$ cd freedom-e-sdk
tamhan@TAMHAN18:~/riscvneu/freedom-e-sdk$ git
   checkout lofive-r1
M            freedom-devicetree-tools
. . .
```

Nieuw is hier vooral dat we na het downloaden van het hoofdarchief nog een submodule moeten laden. Dat is nodig om de LoFive R1-specifieke modules te verkrijgen.
In de volgende stap moeten we Github opdracht geven om te synchroniseren en vervolgens nog een submodule te downloaden:

```
tamhan@TAMHAN18:~/riscvneu/freedom-e-sdk$ git
   submodule sync
Synchronizing submodule url for 'doc/html'
. . .
tamhan@TAMHAN18:~/riscvneu/freedom-e-sdk$ git
   submodule update --init --recursive
. . .
Submodule path 'freedom-devicetree-tools': checked
   out '4f25a7512696f0b41c17e517d39d097499b931a7'
```

De aanroep van `sync` is niet dwingend vereist – bij tests van de auteur kwam het echter tot foutmeldingen, dus in geval van twijfel is een 'voorzichtiger' benadering geen gek idee.
De RISC V-toolchain gebruikt over het algemeen de GCC-compiler en OpenOCD; enkele andere evaluatieboards gebruiken in plaats daarvan een Segger-programmeerinterface. Omdat het compileren van de toolchain op het werkstation relatief lang duurt, biedt SiFive nu een min of meer gebruiksklaar pakket aan. Open als eerste de URL *https://www.sifive.com/boards* in uw lievelingsbrowser en scroll vervolgens omlaag naar de tekst *Prebuilt RISC-V GCC Toolchain*, zoals te zien in **figuur 4**.
In elk geval hebben we voor de volgende stappen de versies *GNU Embedded Toolchain - v2019.08.0* en *OpenOCD - v2019.08.2* nodig. Klik op de beide links en pak de inhoud vervolgens uit in submappen van de werkmap. De auteur werkt in de volgende stappen met de mapnaam *riscvneu*; de uitvoer van het `dir`-commando ziet er als volgt uit:

```
tamhan@TAMHAN18:~/riscvneu/$ dir
freedom-e-sdk/
riscv64-unknown-elf-gcc-8.3.0-2019.08.0-x86_64-linux-
   ubuntu14/
riscv-openocd-0.10.0-2019.08.2-x86_64-linux-ubuntu14/
```



Figuur 3. Een USB/serieel-module van FTDI wordt gebruikt voor het programmeren.

Voor het gemak gaat de toolchain ervan uit dat compiler en co. via omgevingsvariabelen worden gevonden – voor het installeren van een nieuwe toolchain hoeft dan in de praktijk slechts zo'n omgevingsvariabele iets te worden gewijzigd.

Omdat wijzigingen in de parameter PATH doorgaans veel werk betekenen, werken we bij de volgende stappen met tijdelijk vastgelegde paden:

```
tamhan@TAMHAN18:~/riscvneu/freedom-e-sdk$ export
   RISCV_OPENOCD_PATH=/home/tamhan/riscvneu/riscv-
   openocd-0.10.0-2019.08.2-x86_64-linux-ubuntu14/
tamhan@TAMHAN18:~/riscvneu/freedom-e-sdk$ export
   RISCV_PATH=/home/tamhan/riscvneu/riscv64-unknown-
   elf-gcc-8.3.0-2019.08.0-x86_64-linux-ubuntu14/
```

Merk op dat de export-declaratie alleen geldig is in het terminalvenster waarin hij werd ingevoerd. Als u een ander venster wilt gebruiken of wanneer u het venster per ongeluk sluit, moet u weer vanaf het begin beginnen. Ook merken we voor alle zekerheid op dat de afgedrukte mapnamen en -waarden alleen van toepassing zijn op het werkstation van de auteur - het is niet erg waarschijnlijk dat uw gebruikersnaam ook *tamhan* luidt.

Configuratie van OpenOCD

OpenOCD is een relatief gecompliceerde Linux embedded programming-tool. Het maakt via verschillende ook wel *probes* genaamde systemen verbinding met een doelsysteem om vervolgens via een netwerkpoort op het werkstation opdrachten te ontvangen. Normaal werkt OpenOCD als 'team' samen met de GDB-debugger.

Het probleem met OpenOCD is dat de configuratie in .conf-bestanden erg 'streng' is. Omdat de ontwikkelaar die de toolchain heeft aangepast aan de RISC-V-processor, niet met Ubuntu 18.04, werkt, doen zich soms fouten voor zoals deze:

```
tamhan@TAMHAN18:~/riscvspace/freedom-e-sdk$ sudo make
   upload PROGRAM=led_fade BOARD=freedom-e300-lofive
[sudo] password for tamhan:
. . .

Error: unable to open ftdi device with vid 0403, pid
   6010, description 'FT2232H-56Q MiniModule', serial
   '*' at bus location '*'
```

Wanneer uw systeem deze fotmelding genereet, moet u als eerste bepalen welke *Description*-tekst de kernel-driver toewijst aan de aangesloten FTDI-module. Dit gaat het eenvoudigste met behulp van `lsusb`:

```
tamhan@TAMHAN18:~$ lsusb
. . .
Bus 001 Device 009: ID 0403:6010 Future Technology
   Devices International, Ltd FT2232C Dual USB-UART/
   FIFO IC
```

In de volgende stap moeten we zoeken naar de problematische beschrijving. Dit gaat het gemakkelijkst met de oude vertrouwde commandoregel-opdracht *grep*; de uitvoer ziet er dan zo uit:

```
root@TAMHAN18:~/riscvneu/freedom-e-sdk#  grep -r
   FT2232 *
bsp/lofive-r1-bootloader/openocd.cfg:ftdi_device_desc
   "FT2232H-56Q MiniModule"
bsp/lofive-r1/openocd.cfg:ftdi_device_desc "FT2232H-
   56Q MiniModule"
```

Open de beide bestanden dan met een willekeurige editor en zoek het volgende blok declaraties:

```
interface ftdi
#ftdi_device_desc "Dual RS232-HS"
ftdi_device_desc "FT2232C Dual USB-UART/FIFO IC"
ftdi_vid_pid 0x0403 0x6010
```

Dit informeert OpenOCD in de eerste regel dat het te maken heeft met een FTDI-interface. Daarna volgt, naast de twee numerieke ID's, een description-tekst. Beide moeten het mogelijk maken het doelapparaat te vinden.
Omdat in ons geval de combinatie van VID en PID voldoende is, kunnen we de configuratie vereenvoudigen:

```
interface ftdi
ftdi_vid_pid 0x0403 0x6010
```

Nu begint het

Een probleem waarmee men sinds de introducte van het eerste evaluatieboard mee wordt geconfronteerd, is de gegarandeerd verouderde firmware. Dat ligt ook voor de hand, omdat de boards niet rechtstreeks van de fabrikant op de werktafel van de ontwikkelaar terechtkomen – ze liggen eerst geruime tijd bij een distributeur, waar de meegeleverde software een heleboel stof verzamelt.
Dus onze eerste officiële handeling is het bijwerken van de bootloader. Daartoe moeten we eerst de configuratieomgeving opschonen, omdat daar artefacten van de computer van de leverancier in voorkomen:

```
root@TAMHAN18:~/riscvneu/freedom-e-sdk# make
   PROGRAM=lofive-boot TARGET=lofive-r1-bootloader
   clean
make -C /home/tamhan/riscvneu/freedom-e-sdk/software/
   lofive-boot PORT_DIR= clean
make[1]: Entering directory '/home/tamhan/riscvneu/
   freedom-e-sdk/software/lofive-boot'
. . .
```

Iedereen die via de Freedom SDK met een RISC V-Board werkt, doet dit meestal via het make-tool. Naast de twee variabelen `PROGRAM` en `TARGET`, verwacht het een opdracht ter uitvoering. De daadwerkelijke uitlevering van de bootloader-code vindt dan als volgt plaats:

```
root@TAMHAN18:~/riscvneu/freedom-e-sdk# make
   PROGRAM=lofive-boot TARGET=lofive-r1-bootloader
   upload
cd /home/tamhan/riscvneu/freedom-e-sdk/bsp/lofive-r1-
   bootloader/build/debug/ && \
```

De auteur roept de verschillende make-opdrachten voor het gemak op vanuit een root-shell. Als u dit niet wilt, kunt u de verschillende machtigingen ook handmatig instellen.
Na de succesvolle uitlevering van de bootloader kunnen we ons met een eerste programma bezighouden:

```
root@TAMHAN18:~/riscvneu/freedom-e-sdk# make
   PROGRAM=sifive-welcome TARGET=lofive-r1 upload
```

Nadat deze opdracht zijn doorlopen, ziet u een – weliswaar trage – golfvorm op de PWM-uitgangen. Wanneer die op uw oscilloscoop zichtbaar zijn (rolmodus vereist of aanbevolen!) is de configuratie met goed gevolg afgesloten. Ons voorbeeldprogramma levert ook statusinformatie tijdens de uitvoering; die kunt u bekijken met de screen-opdracht of in een willekeurig ander terminalvenster:

```
tamhan@TAMHAN18:~$ screen /dev/ttyUSB1 115200
[screen is terminating]
tamhan@TAMHAN18:~$
```

Houd er bij het gebruik van deze opdracht rekening mee dat u *screen* beslist moet starten voor het eigenlijke programma. Overigens kan de screen-terminal niet gemakkelijk worden uitgeschakeld – de eenvoudigste manier gaat via *Ctrl-A* en vervolgens *K*. Screen vraagt dan of u het venster echt wilt afsluiten. Het enig juiste antwoord is *Y*.

Waar staat de code?

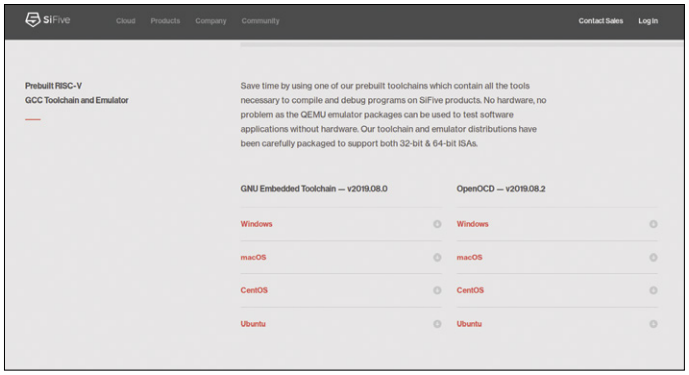
Tot slot willen we in dit artikel kort ingaan op de vraag waar de ‘te verwerken code’ staat. Het antwoord is de software-map; In het geval van ons projectvoorbeeld ziet de inhoud ervan er als volgt uit:

```
root@TAMHAN18:~/riscvneu/freedom-e-sdk/software/
   sifive-welcome# ls
debug  LICENSE  LICENSE.Apache2  LICENSE.MIT
   Makefile  README.md  sifive-welcome.c
```

Het makefile dat verantwoordelijk is voor het besturen van de verschillende compilatieprocessen beperkt zich doorgaans tot het invoegen van een makefile dat door SiFive is voorbereid – en zorgt vervolgens voor de implementatie van de verschillende commando’s zoals upload.
Alleen al om plaatsredenen moeten we onze bespreking hier beëindigen – met de opmerking dat de toolchain op het moment van schrijven nog geen raad weet met C++.

Is het de moeite waard?

Als u vandaag de dag op zoek gaat naar een goedkope microcontroller, vindt u bij ST, Microchip en de verschillende Chinese aanbieders sneller wat u wilt dan bij RISC-V. Want ook een



Figuur 4. De downloadoptie voor de RISC V-toolchain is goed verstopt.

‘open’ architectuur zorgt er niet voor dat het silicium gratis is: vanwege het geringe aantal verkochte exemplaren zorgen momenteel schaaleffecten er voor dat ARM en andere gepatenteerde architecturen per saldo goedkoper zijn.

Het werken met RISC-V is al de moeite waard voor personen die op welke manier dan ook geïnteresseerd zijn in het RISC-V-platform. Vanwege de zeer brede steun van een groot aantal deelnemende bedrijven mag echter worden aangenomen dat het platform in de embedded sector tenminste een heleboel aandacht zal krijgen. ⏮

(191138-04)

Weblinks

[1] RISC-V Foundation: https://riscv.org/
[2] Veelgestelde vragen over RISC V: https://riscv.org/faq/
[3] LoFive bij GroupGets: https://groupgets.com/manufacturers/qwerty-embedded-design/products/lofive-risc-v