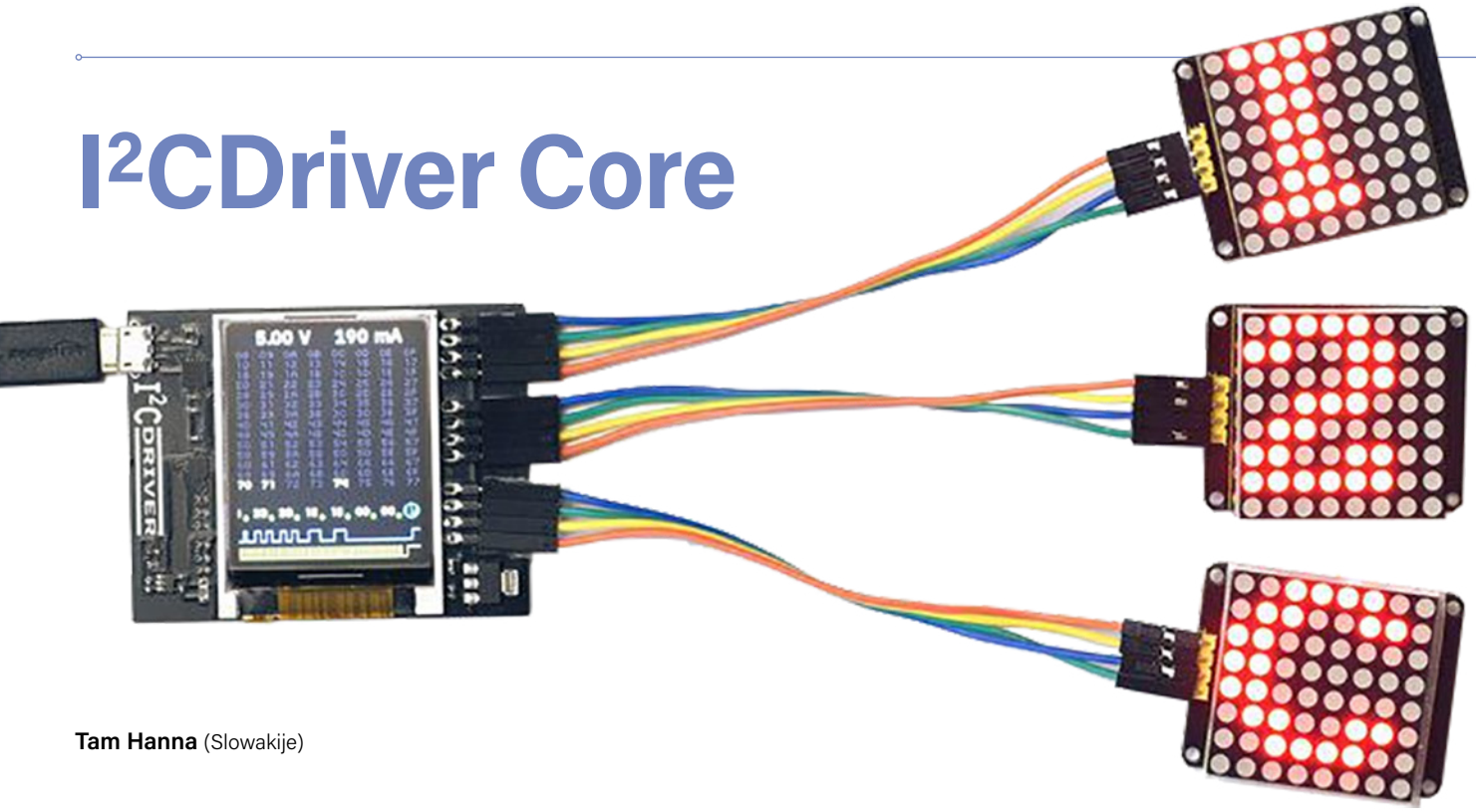
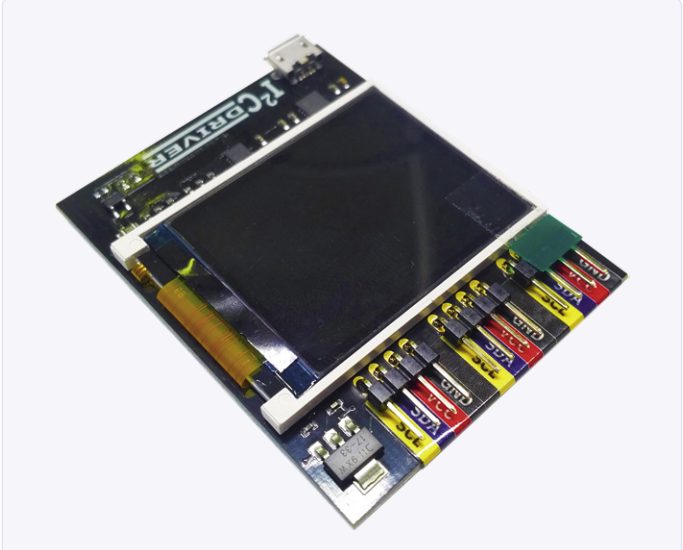
I²CDriver Core



Tam Hanna (Slowakije)

I²C is een veelgebruikte bus die in een grote verscheidenheid aan embedded applicaties wordt toegepast. Voor test- en ontwikkelingsdoeleinden heeft Excamera Labs de I²CDriver ontwikkeld, een board dat als brug tussen PC-software en I²C-hardware (zoals sensoren) fungeert. De I²C-communicatie wordt ook gelogd en duidelijk weergegeven op een kleurendisplay.



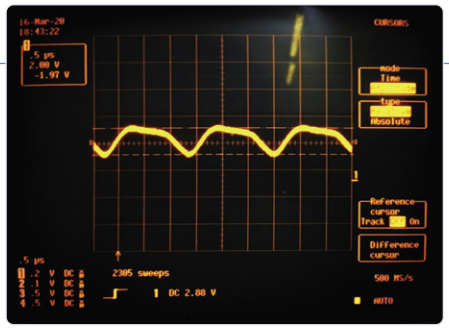
Figuur 1. De I²CDriver is lekker compact.

Na het uitpakken ziet u – zoals in **figuur 1** – een board met een microUSB-connector en drie gelijkwaardige pinheaders met de I²C-signalen. Rubber voetjes aan de onderkant van de print voorkomen dat deze op uw werkbank kan verschuiven.

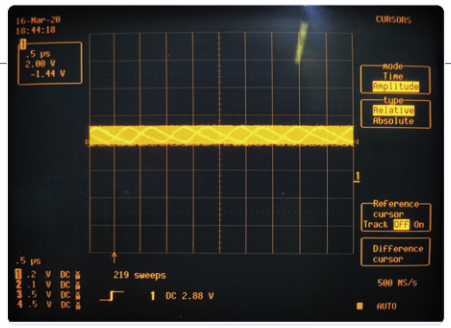
- Excamera Labs voegt bij de I²CDriver Core Kit drie jumperkabelsets bij, die een directe verbinding met de hardware mogelijk maken. De voedingspennen leveren nominaal maximaal 500 mA bij 3,3 V – in **figuren 2, 3 en 4** ziet u hoe het product zich gedraagt wanneer het elektrisch belast wordt.
- Overigens is die 3,3 V afkomstig van de voeding van de computer – het is goed mogelijk dat een deel van de rimpel die zichtbaar is in de oscillogrammen te wijten is aan de USB-bus. Houd er in dit verband ook rekening mee dat de I²CDriver geen galvanische scheiding tussen DUT en computer garandeert.

Test met echte hardware

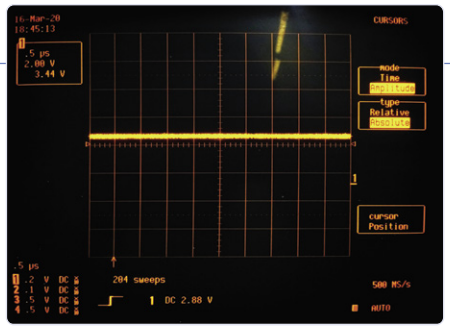
Aangezien in het bedrijf van de auteur momenteel in de vorm van de HygroSage een I²C-sensorsysteem wordt ontwikkeld, ligt het voor de hand dat te gebruiken. Hiervoor sluiten we de I²CDriver tussen de PC en het sensorsysteem aan (zie **figuur 5**). Vanwege het geringe stroomverbruik 'riskeren' we voeding direct vanuit de I²CDriver. HygroSage startte zonder problemen op, ondanks de ietwat avontuurlijke bekabeling, en de weergave van het stroomverbruik aan de bovenin het display werd geactualiseerd. Interessant is dat op het display van de I²C-driver pas iets te zien is na de start van de computersoftware (meer daarover in de volgende paragraaf) die echter soms vastloopt tijdens het opstarten. Bij een succesvolle start ziet u de 'Heatmap' van **figuur 6**, die uitsluitsel geeft over hoe vaak de verschillende apparaten aangesproken worden. Bij het werken met het kleurendisplay valt op dat Excamera de kijkhoek zo heeft gekozen dat de scherminhoud onder een bepaalde hoek zichtbaar is – als u recht op het display kijkt, is met name de Heatmap nauwelijks zichtbaar.



Figuur 2. De I²C-driver bij 500 mA.



Figuur 3. …en bij 200 mA…



Figuur 4. …en onbelast.

Stuur me aan!

Onder [1] vindt u onder het tabblad *Resources* het bestand *i2cdriver-installer.exe*, waarmee u de I²CDriver onder Windows kunt gebruiken. Na het downloaden moet u het met de rechtermuisknop aanklikken en in het configuretievenster aangeven dat het afkomstig is van de lokale computer voordat het besturingssysteem de uitvoering van het installatieprogramma toestaat. Ontwikkelaars die met Linux of Mac OS werken, vinden de corresponderende installatie-instructies op de bovengenoemde website.

Als dit achter de rug is, openen we de map *C:\Programmabestanden (x86)\Excamera Labs\I2CDriver*, waar we zowel een commandoregeltool als een GUI-versie van het product vinden.

Als u de software start met aangesloten I²CDriver en op de knop *Monitor Mode* klikt, krijgt u – zoals te zien in **figuur 7** – informatie over de laatst afgewikkelde transactie.

In de praktijk is het nut van de (visueel aantrekkelijke) analysefunctie echter beperkt, omdat in bijna alle gevallen meer dan één pakket tegelijk onderweg is. In dat geval is een klik op *Capture Mode* aan te raden. De knop komt dan in geactiveerde toestand, zoals in **figuur 8**.

Deze functie – die door de auteur is getest door bewust fouten te veroorzaken en daarna de communicatie te bestuderen – grenst in zoverre aan het geniale dat de communicatie met weinig moeite kan worden vastgelegd. 'Lastige' fouten, die slechts af en toe voorkomen, kunnen op deze manier net als met een digitale geheugen-oscilloscoop aan het licht worden gebracht,

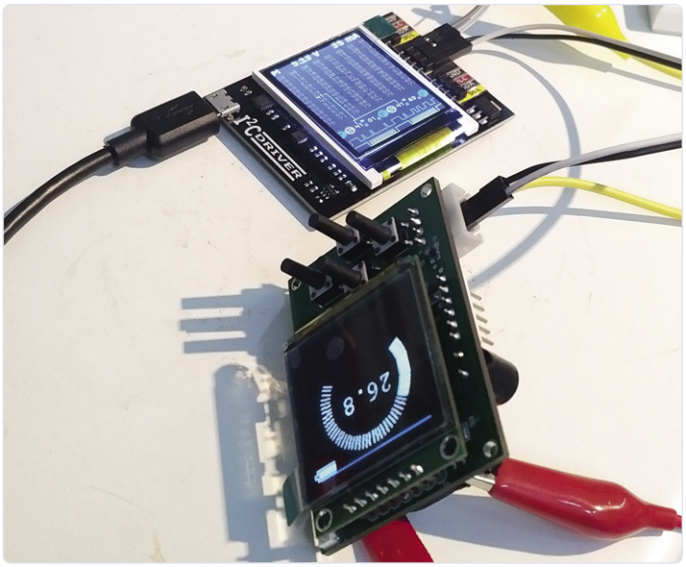
Programmatische interactie

Enige tijd geleden was de auteur bezig met een opdracht die de implementatie van een relatief complex algoritme vereiste. De handigste manier bleek te zijn om het geheel eerst op de PC aan de praat te krijgen, en het vervolgens in de controller te transplanteren. Een soortgelijke procedure is ook geschikt voor de inbedrijfstelling van complexe sensoren. Via de commandoregel is met *i2ccl* een programma beschikbaar waarmee u opdrachten naar de I²CDriver stuurt volgens dit schema:

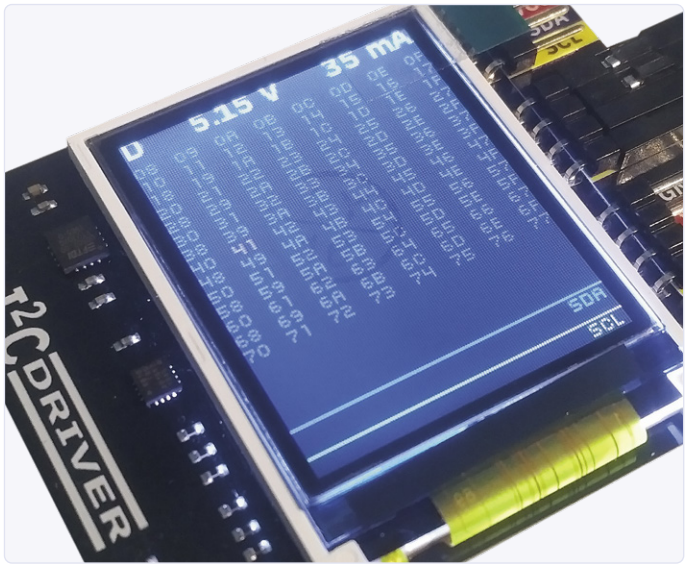
```
C:\Program Files (x86)\Excamera Labs\I2CDriver>i2ccl.exe
Usage: i2ccl <PORTNAME> <commands>
```

Hier is het bijzonder handig dat we informatie kunnen schrijven naar of uitlezen uit specifieke registers van aangesloten apparaten. Dat helpt niet alleen bij de ingebruikname van onbekende sensoren, maar kan ook worden gebruikt om informatie uit te lezen tijdens (geautomatiseerde) testprocedures.

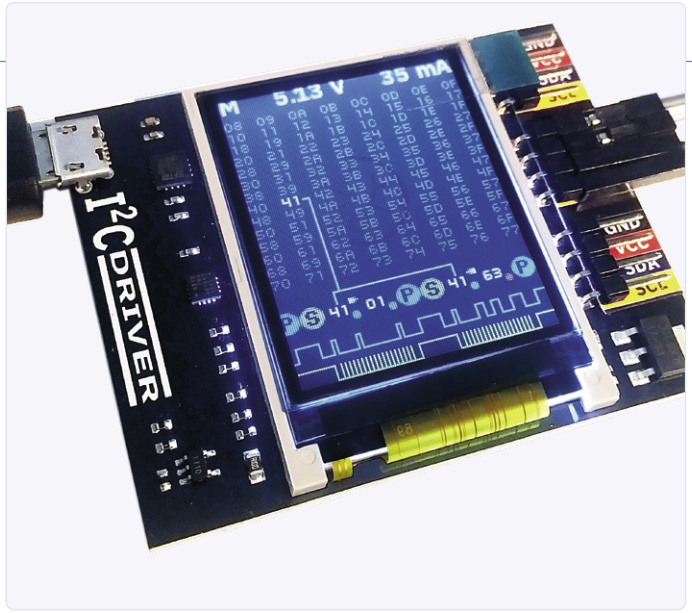
Als u niet in de shell wilt programmeren, kunt u een Python API gebruiken. De fabrikant demonstreert het gebruik ervan in de vorm



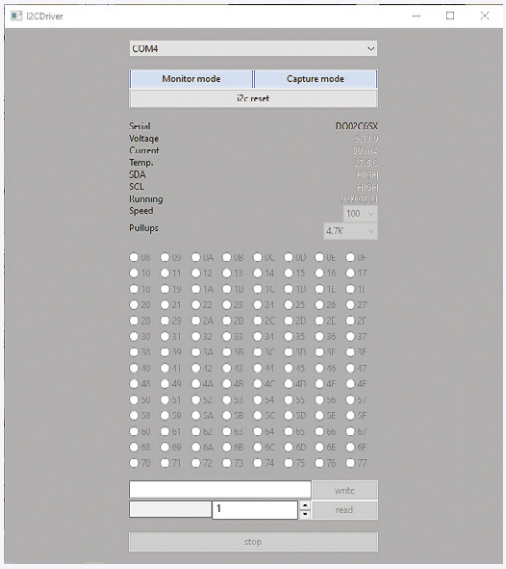
Figuur 5. De I²CDriver in combinatie met een sensorprint van de auteur.



Figuur 6. Met slechts één sensor, zodat slechts één waarde in kleur wordt weergegeven.



Figuur 7. Registeroperaties op het display van de I²CDriver.



Figuur 8. De I²CDriver-desktopsoftware heeft een rustieke charme.

van een reeks kant-en-klare voorbeelddrivers – met de volgende snippet kan bijvoorbeeld een LM75B worden uitgelezen:

```
import i2cdriver
 i2c = i2cdriver.I2CDriver("/dev/ttyUSB0")
d=i2cdriver.EDS.Temp(i2c)
d.read()
17.875
 d.read()
18.0
```

De eigenlijke stuur-API is eenvoudig en kan op GitHub [2] worden bekeken:

```
class LM75B:
    def __init__(self, i2, a = 0x48):
          self.i2 = i2
         self.a = a
```

Excamera Labs implementeert de hardware drivers met behulp van de Python-OOP-API. `self` is een taalspecifieke driver, terwijl `i2` een I²C-driverobject is. Last but not least is `a` het adres waaronder de sensor kan worden geadresseerd.
Registerinformatie wordt dan op deze manier ingelezen:

```
    def reg(self, r):
        return self.i2.regrd(self.a, r, ">h")

    def read(self):
        return (self.reg(0) >> 5) * 0.125
```

Opgemerkt moet worden dat er een scancommando is dat op *i2cdetect* lijkt en dat, wanneer het wordt aangeroepen vanaf de Python-opdrachtregel, de scanfunctie uitvoert die bekend is van OrangePi en consorten.
Ten slotte wordt verwezen naar de documentatie die beschikbaar is onder [3]. Het legt de I²C-API en het fysieke communicatieprotocol uit - als u alleen bent met de FTDI-API, kunt u ook direct toegang krijgen tot de I²CDriver.

Conclusie

De I²CDriver is een van die producten waarvan men het bestaansrecht pas na een tijdje inziet, maar daarna wil men het niet meer missen. Of het nu gaat om een snelle analyse van de activiteit van een I²C-netwerk of om de ingebruikname van een sensor – het board biedt waardevolle ondersteuning. De prijs is acceptabel gezien de tijdwinst die het oplevert – alleen jammer dat er geen standalone-modus is.

200148-04

WEBLINKS

[1] **Software**: https://i2cdriver.com/
[2] **Stuur-API**: https://github.com/jamesbowman/i2cdriver/blob/master/python/lm75b.py
[3] **Documentatie**: https://i2cdriver.com/i2cdriver.pdf