Onderzoek of Arduino/Python inzetbaar zijn voor practicum toepassingen…

Hz, 29apr’20 Versie 0.3a

Opdracht 1

Status 2

De werking in het kort 2

Resultaten tot nu toe 2

Knelpunten 3

Parameterprotocol 3

Voorbeelden parameterprotocol 3

Voorbeeld resultaat in Python (Plot) 4

ToDo 5

Prioriteiten 5

Bronnen 6

# Opdracht

Ten behoeve van een aantal practicum experimenten wordt gebruik gemaakt van een Data Acquisitie Device (DAQ) van National Instruments (NI) in combinatie met Labview. Het DAQ beschikt over een aantal analoge en digitale ingangen en communiceert via een USB interface met Labview.

De opdracht is te onderzoeken of een subset van functionaliteit van het DAQ i.c.m. Labview eveneens bereikt kan worden met een combinatie van een Arduino based single board computer in combinatie met Python zodat deze de NI devices en Labview op termijn kunnen vervangen.

Uit de aard van de experimenten blijkt dat behoefte is aan een DAQ met minimaal de volgende specificaties:

* Twee AD inputs
* Een digitale poort voor triggering
* Samplefrequentie 1 – 20.000 samples/sec
* Aantal samples 1 – 20.000

Daarnaast:

* Aansturing vanuit Python (via USB)
* Uitlezing sample data (via USB)

Wat de hardware betreft:

* Input poorten voorzien van protectie en een instelbare versterker/verzwakker/bias

# Status

Voor dit onderzoek is als Arduino based single board computer gekozen voor een ESP32 DEVKIT V1 module (zie Bronnen). Deze module beschikt over een dual-core CPU en voldoende geheugen, analoge/digitale poorten en interfaces. Is niet al te duur, goed gedocumenteerd en goed verkrijgbaar en wordt geprogrammeerd via een Arduino IDE. De communicatie met de ESP32 verloopt via USB (serieel). Deze module wordt ingezet als een ‘black box’ en regelt de conversie, buffering en het serieel transport.

De hardware en software is modulair ontwikkeld zodat aanpassingen relatief eenvoudig zijn te realiseren. Aanpassingen kunnen echter niet zondermeer door een gebruiker worden uitgevoerd vanwege de complexiteit.

Het user interface is geprogrammeerd in Python-3 en kan naar behoefte worden aangepast door een gebruiker zolang het parameterprotocol voor het verzenden en ontvangen van parameters/data wordt gerespecteerd.

## De werking in het kort

Category ADC: Via Python (of via een console) wordt een opdrachtstring met conversie parameters naar de ESP32 gestuurd (zie Parameterprotocol). De ESP32 analyseert deze string en gaat ermee aan de slag. Wanneer de opdracht is afgerond worden de resultaten teruggestuurd naar Python (of console). De communicatie vindt plaats via het seriële kanaal van het USB interface.

Category TEKST: Via Python wordt een tekst op het display van het DAQ getoond.

Category LEVEL: Schakelt de input level indicator aan of uit.

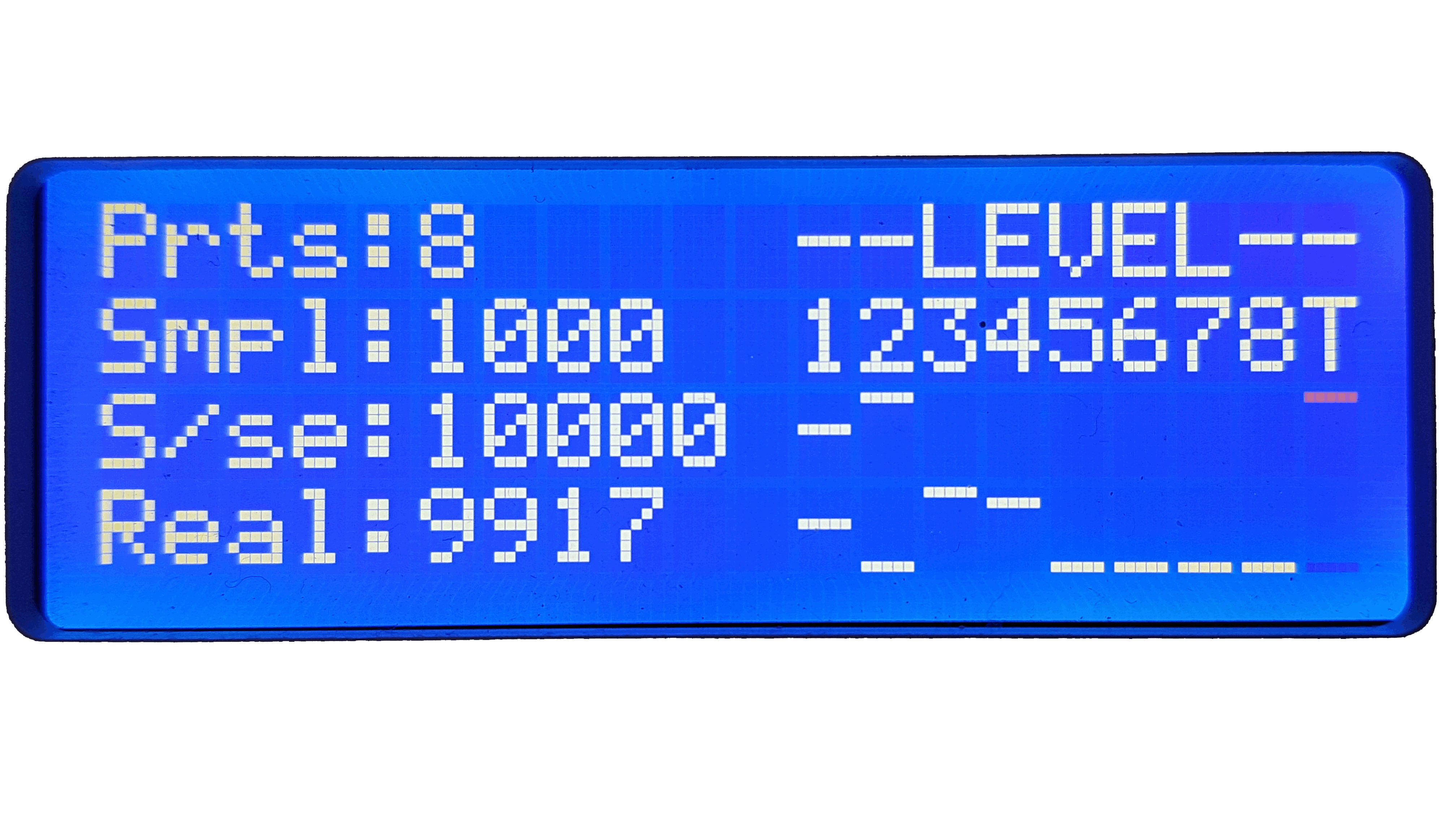
Category DAC (experimenteel): Schakelt de DAC aan of uit.

## Resultaten tot nu toe

Voor de meest actuele situatie: <http://github.com/hazowa/DAQ>)

* Aantal poorten: maximaal aantal van acht met een 12bits resolutie
* Samplefrequentie: *In de praktijk lijkt de grens te liggen op ruim 10.000 smpl/sec/poort wanneer alle acht poorten gelijktijdig worden uitgelezen. Twee poorten gelijktijdig uitlezen is mogelijk met een sample rate van maximaal 45.000 smpl/sec/poort.*
* Samplebuffer: in deze test is een buffer voor *50.000* samples beschikbaar voor alle analoge poorten tezamen. Het maximaal aantal samples per poort is Samplebuffer/Aantal\_poorten.
* Triggerpoort, optioneel: naast de analoge inputs is een digitale poort beschikbaar die kan worden ingezet als trigger voor AD conversie. Is deze triggerpoort geactiveerd dan zal de conversie pas starten als aan een bepaalde voorwaarde van het signaal op deze poort is voldaan: een hoog level, een laag level, een verandering, een opgaande flank of een neergaande flank. In het geval van een hoog of laag trigger level kan als optionele parameter een debounce-tijd worden meegegeven om eventuele problemen door contactdender/spikes/ruis te voorkomen.
* Digital to Analog converter: De inhoud van een array wordt cyclisch naar DAC1 gestuurd. Resulteert in een sinus met een frequentie van 130Hz.

Een functionerend prototype met bovenstaande specificaties is gereed om te worden getest (zie de bijlage met hard- en softwarebeschrijvingen). De gevoeligheid van de analoge ingangen 1 t/m 7 is voorlopig ingesteld op 0-10V (impedantie 50kohm) voor het bereiken van de maximale resolutie van de AD converter. Analoge ingang 0 is voorzien van een differential versterker. Deze versterker heeft een instelbare gain (1,2,5 sequentie tot 1000x), schakelbaar AC/DC filter en een bias regeling. De versterker is inmiddels inzetbaar maar moet nog verder worden doorontwikkeld wanneer de juiste specificaties bekend en componenten beschikbaar zijn.



LCD display 20x4

Het prototype is tevens voorzien van een LCD schermpje voor de weergave van de parameters, de actuele sample rate en een input signal level weergave. Dit display wordt tevens ingezet voor debugging van de software. De foto hierboven toont in het linkerdeel van het scherm het aantal actieve poorten, het aantal samples, het aantal samples per seconde per poort en de werkelijke sample rate (Real). Het rechterdeel van het scherm toont de omhullende waarde (maximum en minimum) van de spanning op de AD poorten en de trigger poort.

## Knelpunten

Op dit moment zijn de knelpunten meer van praktische aard, met name programmatuur om schema’s te tekenen en stroomdiagrammen te maken kosten helaas nogal wat inwerktijd. Verder ontbreken componenten om de differential versterker betrouwbaar te kunnen afbouwen en testen (meerstandenschakelaar voor de gain, afschermen van ruis door CPU).

# Parameterprotocol

De parameterstring die door Python (of via een console) naar de ESP32 wordt gestuurd heeft het format hieronder, waarbij elke parameterwaarde is gescheiden door een spatie of een leesteken (instelbaar in Arduino sketch):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aantal Poorten | Samples/Seconde | Aantal Samples | Trigger | Trigger Event | Debounce Time |
| (1 – 8)\* [1]\*\* | (1 – 20000) [10000] | (1 – *50000*) [1000] | (0 – 1) [0] | (0 – 4) [0] | (0 – 10000 [0] |

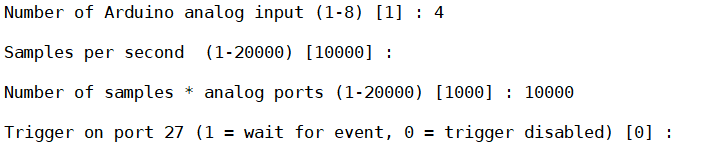
\*( range)  
\*\*[default waarde]

De parameters Aantal Poorten, Samples/Seconde en Aantal Samples zijn verplichte waarden, Trigger Event en Debounce Time zijn optioneel.

## ***Voorbeelden parameterprotocol***

In de screendumps hieronder enkele voorbeelden waarbij Python (als user interface) in een aantal vragen parameters voor conversieslag verzamelt.

Voorbeeld-1:

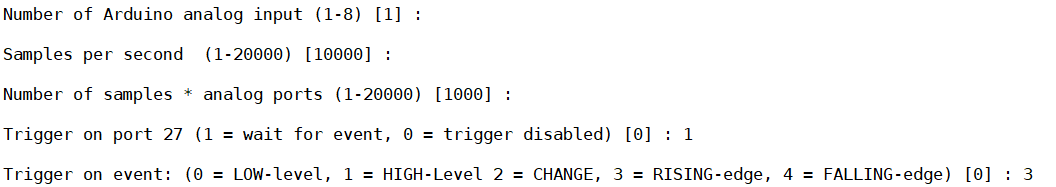
  
Note: Tussen ( ) de grenswaarden en tussen [ ] de default waarde bij het indrukken van Enter.

Bovenstaande invoer resulteert in

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aantal Poorten | Samples/Seconde | Aantal Samples | Trigger | Trigger Event | Debounce Time |
| 4 | 10000 | 10000 | 0 | nvt | nvt |

Na een enter wordt de conversie wordt onmiddellijk gestart.

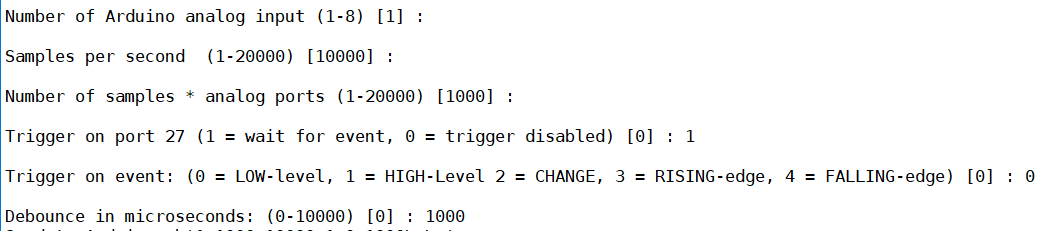
Voorbeeld-2:



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aantal Poorten | Samples/Seconde | Aantal Samples | Trigger | Trigger Event | Debounce Time |
| 1 | 10000 | 1000 | 1 | 3 | nvt |

De conversie wordt gestart bij een opgaande flank van het signaal op trigger ingang.

Voorbeeld-3:



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aantal Poorten | Samples/Seconde | Aantal Samples | Trigger | Trigger Event | Debounce Time |
| 1 | 10000 | 1000 | 1 | 0 | 1000 |

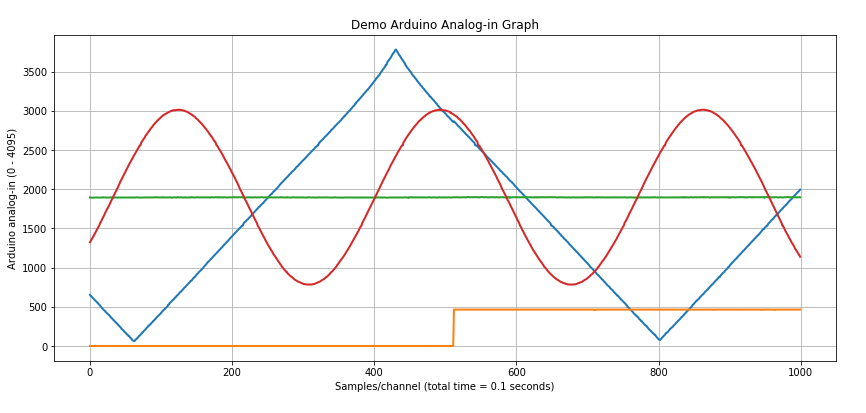
De conversie wordt gestart bij een laag level van het signaal op de trigger ingang maar pas nadat dit level 1000µs stabiel laag is.

## Voorbeeld resultaat in Python (Plot)

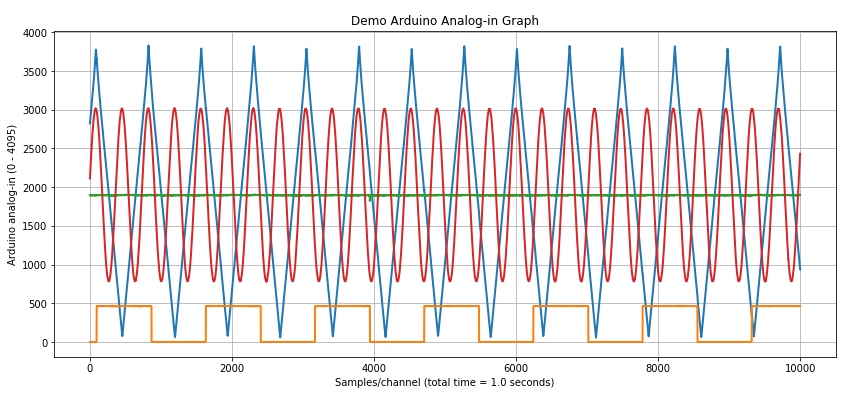
In het Python-plot plaatje hieronder het resultaat van een meting aan een drietal oscillators, een sinus, driekoek en blokgolf. Het vierde signaal is een dc-spanning.

Als parameterstring: 4 poorten, 10.000 samples/sec/poort en 1000 samples/poort

*Dit resulteert in een meting met vier analoge inputs, met een samplefrequentie van 10kHz en een hoeveelheid van 4000 samples, 1000 per poort. (plaatjes zijn geupdate)*



*Wordt als parameterstring: 4 poorten, 10.000 samples/sec en 10.000 samples per poort gestuurd, dan resulteert dit in het plaatje hieronder.*



# ToDo

*Het lijkt er op dat de lastigste software problemen zijn opgelost. De testen zullen moeten uitwijzen wat de maximale acceptabele conversie snelheid kan zijn en wat een voldoende buffergrootte is.*

*Zonder tegenbericht gaat volgende week de aandacht, in volgorde van onderstaande prioriteitenlijst naar hardware en handleiding:*

* *De hardware uitbreiding heeft tot doel aangeboden signalen op een optimale wijze aan te bieden aan de analoge poorten van de ESP32. Om dit te realiseren moet het aangeboden signaal kunnen worden versterkt of juist worden verzwakt en, indien nodig, met behulp van een bias regeling binnen het bereik van de AD converter worden gebracht.*
* *Het doel van de handleiding is dat het Python programma zodanig wordt beschreven dat gebruikers er snel mee aan de slag kunnen en makkelijk kunnen aanpassen aan hun wensen.*

# Prioriteiten

Must haves:

* Bouwen van de hardware voor protectie en versterker/verzwakker/bias van de analoge poorten en trigger poort. Een protype is gereed en kan verder worden ontwikkeld wanneer specificaties bekend zijn en de onderdelen beschikbaar.
* Behuizing prototype (naar voren gehaald, losse breadboards zijn onvoldoende stabiel om te kunnen testen.
* Globale beschrijving hard- en software
* 1e Concept handleiding
* Testen of de opstelling in de praktijk voldoet

Voldoet de test:

* Sample buffer vergroten? (*Is inmiddels vergroot naar 50.000 samples.)*
* Anti-Aliasing Filter (AAF)
* Uitlezing levels op LCD scherm stabiliseren (bijvoorbeeldpeaklevel ipv. momentele waarde)

Nice to have:

* *Toevoegen van een aantal digitale ingangen*
* Pre-triggering, toon ook wat samples voorafgaande aan een triggermoment.
* Display voor weergave van statusinformatie. Is reeds gerealiseerd om debugging te vereenvoudigen.
* Kalibratietool ADC *(prioriteit hangt af van meetresultaat, zie Bronnen voor info en oplossing)*
* ..

# **Bronnen**

Specificaties ESP32 DEVKITV1: <https://docs.zerynth.com/latest/official/board.zerynth.doit_esp32/docs/index.html>

IDE Arduino:  
<https://www.arduino.cc/en/Main/Software>

Arduino Reference:  
<https://www.arduino.cc/reference/en/>

C++ Reference:  
<https://www.cplusplus.com/reference/>

Python IDE SPYDER:  
<https://www.spyder-ide.org/>

Serial API Python:  
<https://pyserial.readthedocs.io/en/latest/pyserial_api.html>

PyFirmata description:  
<https://pypi.org/project/pyFirmata/>

Arduino Firmata:  
<https://github.com/firmata/arduino>

niet-lineariteit AD converter en kalibratie tool  
<https://www.instructables.com/id/Professionals-Know-This/>