Bijlage: Python Software DAQ

Hz02apr’20 Versie 0.2a

Inleiding 1

Globale beschrijving 1

Import van libraries 2

Definitie van variabelen 2

Communicatie met gebruiker 3

Communicatie met het DAQ 4

Verzenden van parameters naar DAQ 4

Ontvangen van conversiedata vanuit DAQ 5

Verwerken van de data 5

Data structuur 5

Demultiplex 5

Plotten van de data 6

# Inleiding

Het prototype Data Acquisitie Device (DAQ) is een op zichzelf staand instrument dat intern draait op programmatuur ontwikkeld in C++ op basis van ESP32 hardware (zie bijlage Arduino-Python DAQ Hardware beschrijving.docx). De communicatie met de buitenwereld loopt via een USB poort, voor zowel de aanlevering van parameters, die de conversie configureren, als voor de uitlezing van het DAQ buffer. Deel van het onderzoek is de inzet van Python, middels Spyder Integrated Development Environment (IDE), voor de aansturing en uitlezing van het DAQ. Hieronder een beschrijving van het Pythonprogramma voor het aansturen en uitlezen van het DAQ. Het is een proof of concept met de nadruk op functionaliteit. De ontwikkeling gaat sneller dan het up-to-date houden van de documentatie. Voor actuele informatie is een github account aangemaakt met daarop de jongst sources: <http://github.com/hazowa/DAQ>

Stroomdiagram Python programmatuur

# Globale beschrijving

Aan de hand van het stroomdiagram hieronder de vijf belangrijkste modulen van dit Python programma:

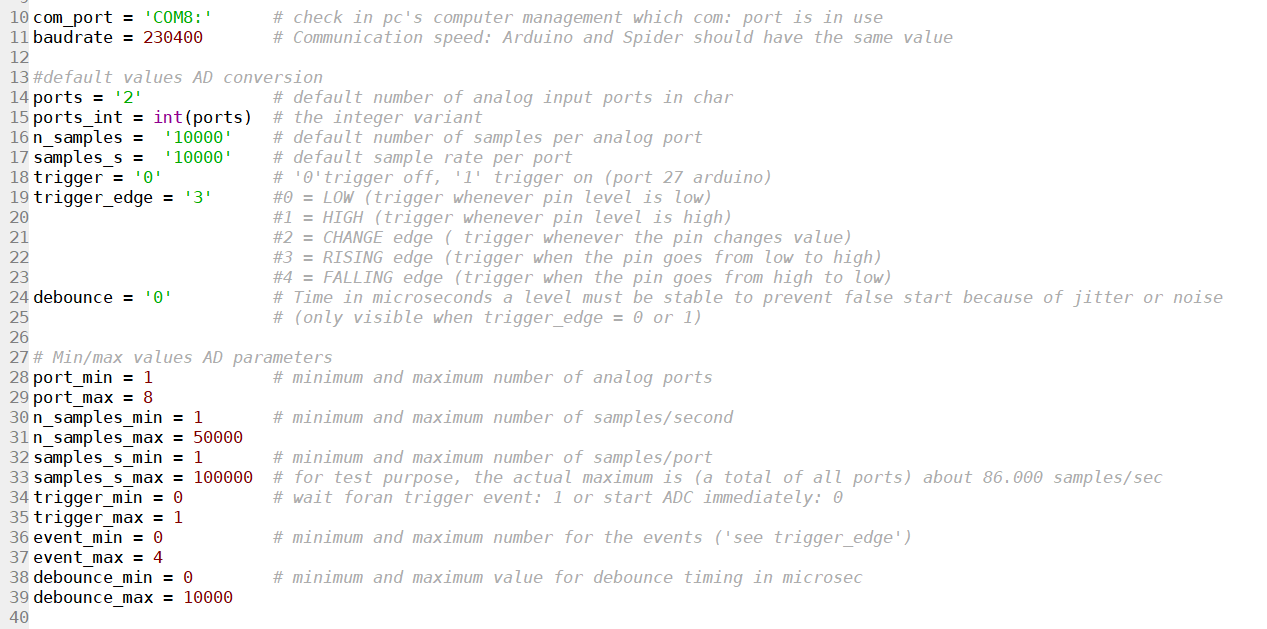
1. Import van libraries
2. Definitie van variabelen
3. Communicatie met gebruiker
4. Communicatie met DAQ
   1. Verzenden van parameters naar DAQ
   2. Ontvangen van conversiedata vanuit DAQ
5. Verwerken van de data met een plot als resultaat.

## Import van libraries

Spyder biedt ondersteuning voor een groot aantal scientific packages. Voor dit programma zijn in gebruik ‘serial’ voor de communicatie met de ESP32, ‘time’ om een delay mogelijk te maken en ‘matplotlib om de vanuit het DAQ ingelezen data weer te geven in een diagram.



## Definitie van variabelen



Het o.s. bepaalt welke COM-poort wordt toegewezen aan het DAQ wanneer deze met een USB kabel wordt gekoppeld aan een pc. De variabele ‘com\_port’ verwijst naar deze poort.

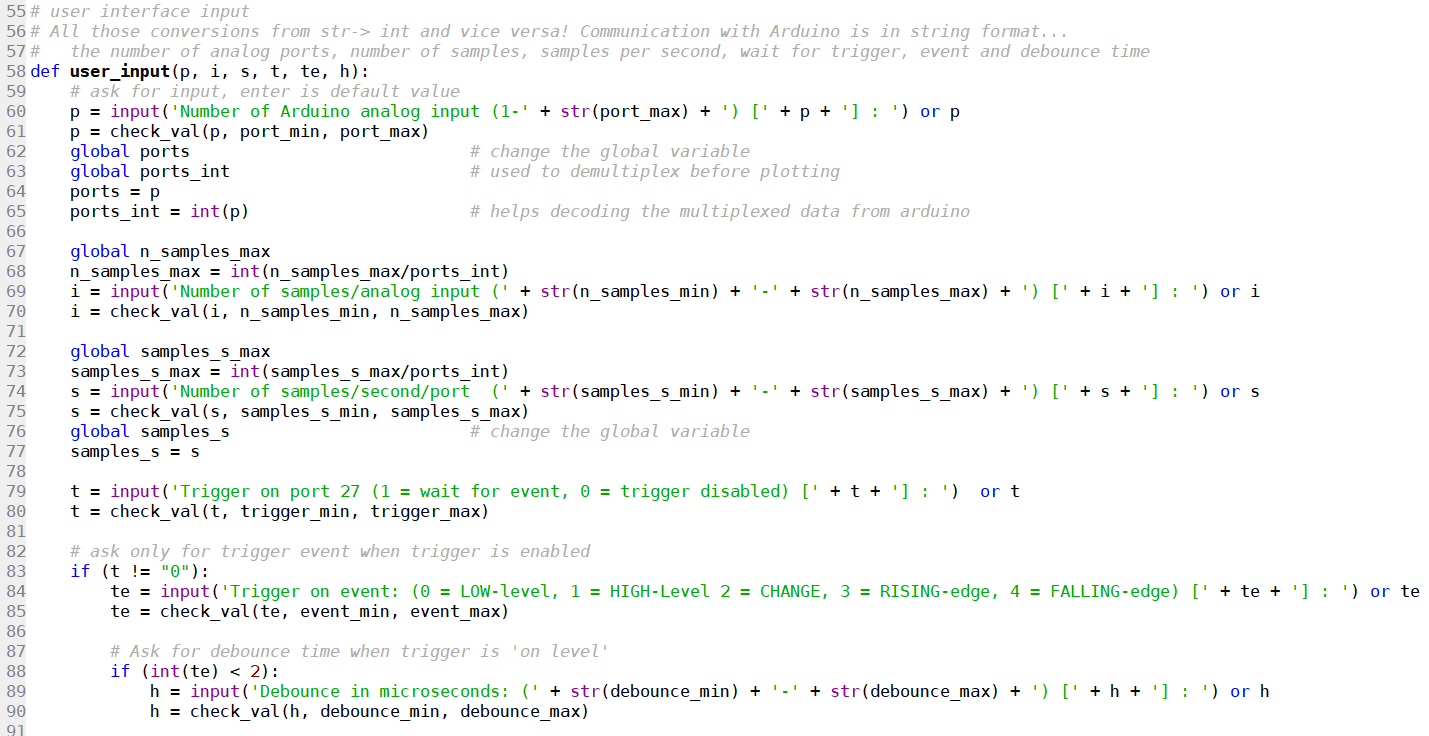
De snelheid waarmee de gebufferde data van het DAQ richting pc wordt gestuurd is afhankelijk van de communicatiesnelheid of baudrate. De maximale baudrate is afhankelijk van een aantal factoren zoals performance en hardware van de pc en kwaliteit van de USB kabel. Met de variabele ‘baudrate’ wordt de communicatiesnelheid ingesteld. Op het display van het DAQ wordt de gedurende enkele seconden na het opstarten de baudrate getoond.

Met een aantal vragen aan de gebruiker worden de conversie parameters bepaald. In het blokje ‘default values AD conversion’ staan de waarden die standaard worden getoond. Onder het kopje ‘Min/max values AD parameters’ de grenswaarden voor de invoer van parameters. De invoer wordt getest op het overschrijden van deze waarden.

## Communicatie met gebruiker

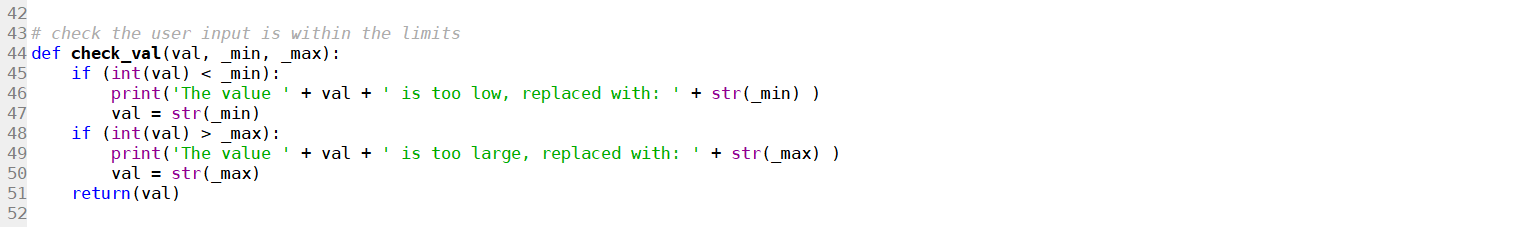
De communicatie met de gebruiker bestaat uit twee onderdelen: het uitvragen van conversieparameters en het controleren van de invoer. De aanroep van deze subroutine vindt plaats vanuit het hoofdprogramma.

In de programmaregels hieronder wordt de gebruiker vragen gesteld over aantallen analoge poorten, het aantal samples per poort en het aantal samples per seconde. Gezien de maximaal haalbare samplefrequentie en de niet onbeperkte buffergrootte van het DAQ bestaat er een relatie tussen het aantal poorten en de maximale waarden van samples per poort en samples per seconde per poort. Na invoer van het aantal poorten vindt een herberekening plaats van de maximale waarden, zodat in de vervolgvraag de juiste grenswaarden worden getoond.

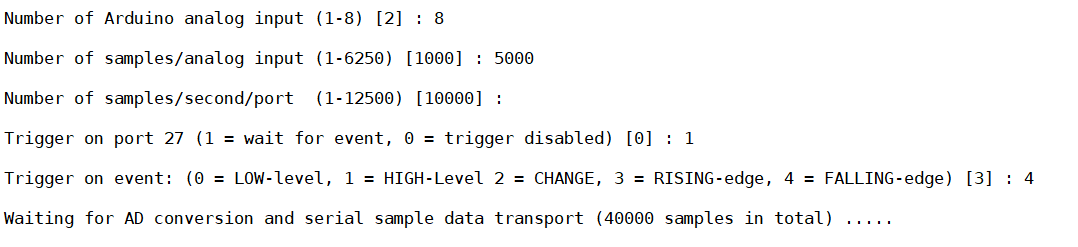


Om resultaten ook buiten deze routine te kunnen gebruiken zijn een aantal variabelen globaal gemaakt.

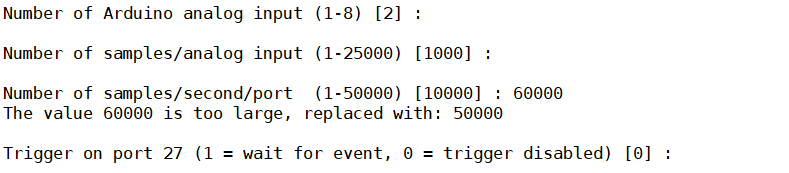
Na elke invoer wordt met onderstaande functie gecontroleerd of de ingevoerde waarde binnen de grenswaarden valt en -indien nodig aangepast- teruggegeven aan het aanroepende programma.



In de schermafdruk hieronder het resultaat van bovenstaande programmaregels:

******

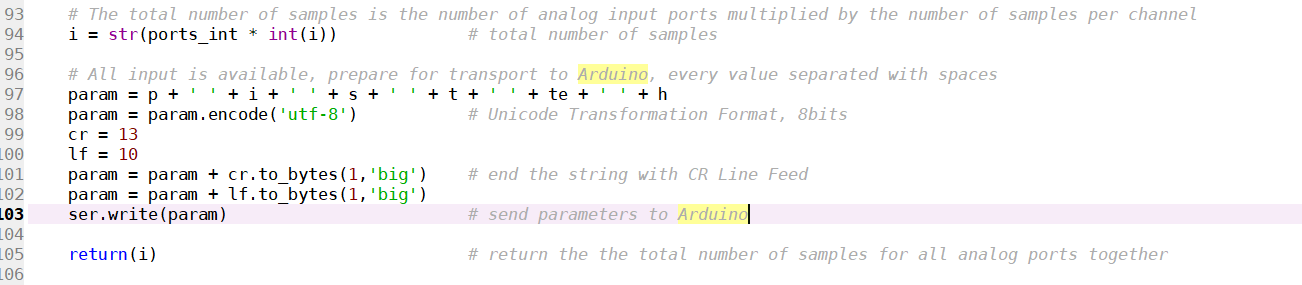
Hieronder en voorbeeld waarbij de invoer de maximale waarde overschrijd en er een correctie optreedt:

******

## Communicatie met het DAQ

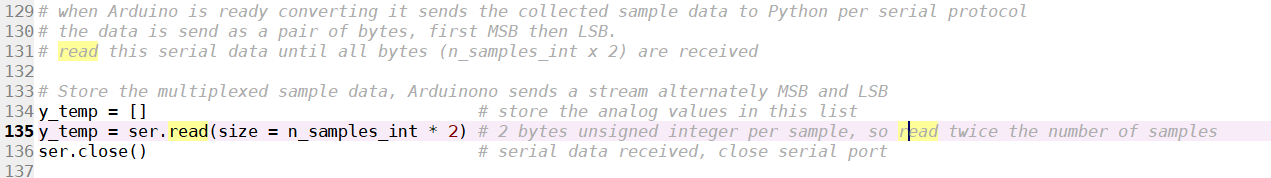
Nadat alle parameters zijn verzameld worden ze gebundeld en voorzien van de juiste codering als een string naar het DAQ gestuurd (regel 103). De functie wordt beëindigd met het retourneren van het totaal aantal in te lezen samples voor verder gebruik in het aanroepend programma.

### Verzenden van parameters naar DAQ



De parameterscheiding is een spatie maar kan door een kleine wijziging in zowel Python als ESP32 programmatuur worden aangepast in bijvoorbeeld een komma.

### Ontvangen van conversiedata vanuit DAQ

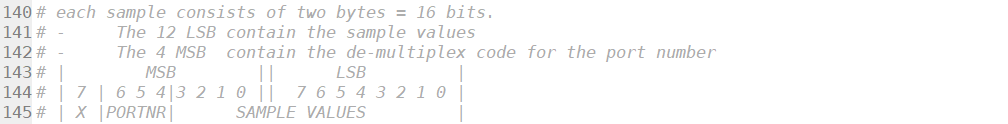


Nadat de conversieparameters door het DAQ zijn ontvangen en de AD-conversie is afgerond wordt het buffer met data aangeboden aan de aangesloten pc. In regel 134 wordt een list geïnitieerd die in regel 135 paarsgewijs wordt gevuld met bytes. Het aantal bytes is 2x het product van het aantal poorten en het aantal samples per poort dat door de gebruiker is opgegeven.

Nadat de data is ingelezen wordt de verbinding met de seriële poort verbroken.

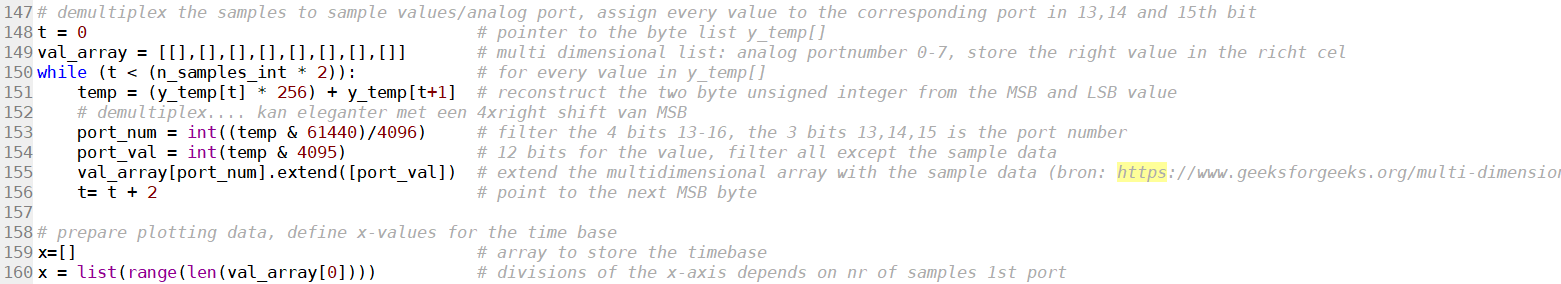
## Verwerken van de data

### Data structuur



De resolutie van de AD converter in de ESP32 is 12 bits. De minimale hoeveelheid data die per seriële poort kan worden verstuurd is een byte ofwel 8 bits. Dit houdt in dat minimaal twee bytes nodig zijn om de 12 bits data te verzenden. We hebben de beschikking over een maximum aantal van 8 analoge poorten die sequentieel worden gesampled. Naast de sample data wordt in de resterende ruimte van de twee bytes het poortnummer genoteerd in het 13,14 en 15 bit. Bit 16 is ongebruikt. Een optie is hierin het triggerlevel (0 of 1) te noteren.

### Demultiplex



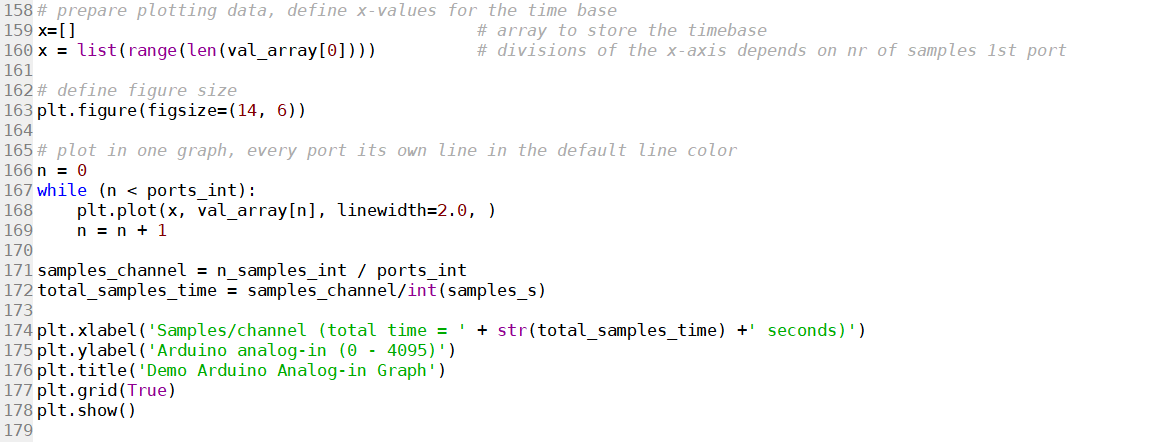
De list ‘y\_temp’ bevat de multiplext sampledata afkomstig van het DAQ. Elk sample bestaat uit twee bytes (=16 bits) waarbij een deel van de 16 bits het adres van de corresponderende analoge poort bevat en een ander deel de sampledata.

Het maximaal aantal beschikbare poorten is vastgesteld op acht. In regel 149 wordt een tweedimensionale list geïnitieerd waarin op elke regel de sampledata van de acht poorten kunnen worden opgeslagen.

In de regels daarna worden, todat het maximaal aantal bytes (n\_samples x 2) is gedemultiplext, de data en poortnummers gescheiden door binair te filteren. De poortnummers vormen een index voor het val\_array bij de opslag van de daarbij behorende data.

### Plotten van de data

Nadat de data gedemultiplext is de data gereed voor analyse. In deze demo wordt de data in een grafiek geplot.



Voor de x-as van de grafiek worden waarden gegenereerd (regel 160) gelijk aan het aantal samples per poort.

In regel 163 wordt de afmeting van de grafiek bepaald waarna in de loop vanaf regel 166 voor elke in gebruik zijnde poort de y-waarde wordt klaargezet.

In regel 171 en 172 wordt voorbereidend werk verricht om een zinnige tekst bij het label voor de x-as te noteren en wordt tenslotte in regel 178 de grafiek getoond.

Hieronder een voorbeeld:

