Program pro měření a ukladaní dat z magnetometru

Albershteyn Andrey

January 9, 2016

Obsah

1 Description

Tetno program se používá pro zprácování a ukladaní dát z observatorního magnetometru. Cílem tohoto programu je lehký přenasitelní software pro spuštění na jednoduchem počítače Raspberry Pi.

Senzor obsahuje FPGA chip který vytváří datový stream a vysíla ho pomoci seriového portu. Tento stream prijámeme pomoci seriového rozhraní Raspberry Pi.

Program je sestaven ze 3 processu. Jeden přijímá data ze seriového portu, druhy provádí kalibrace dat a třeti ukládá data do souboru. Processy jsou spojeny pomoci Pipeline a jsou nezávisle, jenom při ukončení hlavního processu, který čte data, ukončujou se i vše ostatní.

Tento program obsahuje několik testovacích možnosti. Jednou z ních je vysílaní impulsu při čtení dat. V konfiguračním souboru je nastaven nějaký GPIO pin, který bude se používat pro oznamení čtení dat, tedy před začatkem čtení a po konci. Tento postup je užiteční pro kontrolu jsou-li přijate vše datové vzorky.

Dál je možné zpustit program v Virtualním modu, což známená že budeme testovat jenom kalibrace a ukládaní dat. Tedy nepotřebujeme žádný senzor, protože data se budou randomě generovat přimo v programu. Tento mode je užitoční při nějakých uprávach programu nebo ověření že všechno funguje tak jak má fungovat.

Dalši možnost je použiti loggování celého cyklu vypracování dat. Program obsahuje hlašky s potvrzeními o zpracování dat a nastavajicích chybach, které take jde zapnout.

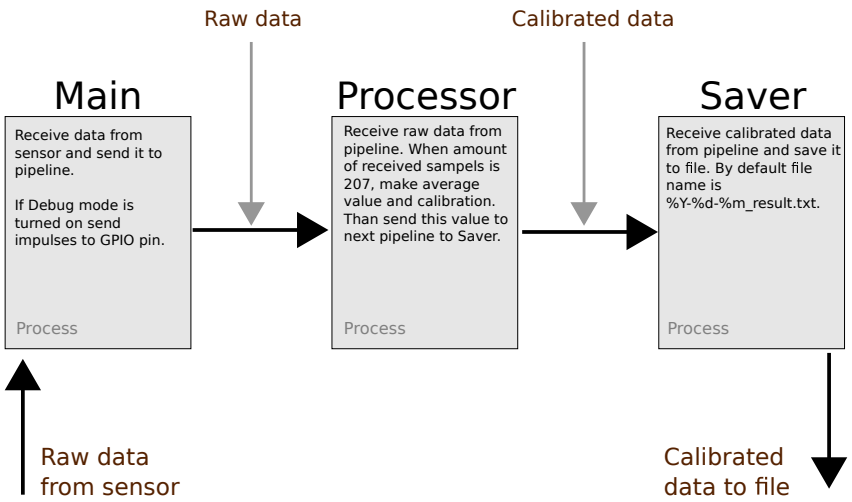
Celý program je napsat jenom s využitim built-in python knihoven. Takže nepotřebuje instalace nestandartních knihoven. Jenom při zapnutem Debug oznamování (použiti nějakého GPIO pinu) je potřeba nainstalovat GPIO knihovnu pro komunikace s piny.

2 Program's structure

V teto sekce máme popis celého programu. Tedy za co jsou odpovidní každý soubor a jaky má funkcional.

Seznam souboru:

1. **main.py** - tento soubor je hlavním vstupem do programu. Běží jako vlastní process a provadí čtení dat ze seriového portu, které pak posílá do pipeline pro kalibrace.
2. **saver.py** - obsahuje jenom jednu funkce, která je spouštěna jako process a provadí ukládaní dat.
3. **calibration.py** - obshauje několik pomocních funkce, které jsou použité v processore pro kalibrace a různé úpravy dat.
4. **processor.py** - obsahuje jedinou funkce, která je spouštená jako process. Provadí kalibrace dat a posilá je dál do ukládacího processu.
5. **reader.py** - tento soubor obsahuje funkce pro nastavnění komunikace a komunikace s zářizenim.
6. **generator.py** - obsahuje třidu, která slouží generatorem nahodných dat používá se jenom v Virtual modě.
7. **settings.py** - obsahuje nastavení programu a parametry kalibrace (například offsety, sensitivity atd.)



3 Configuration settings

V teto sekce máme popis konfigurace programu. Program má dvě konfigurační třídy. Jedna je zodpovídná za nastávení programu (kam ukladat data, používat debug mode atd.), druhá za kalibrace dat (tedy offsety, sensitivity atd.)

Nastavení programu:

1. **samples** - počet přijimaních vzorku za sekundu.
2. **debug** - Zápnout nebo vypnout debug mode. Tedy oznamení každého přijatého vzorku na debug_pinu a loggování průběhu programu.
3. **debug_pin** - GPIO pin na který se budou posilat oznamovácí impulsy o přijatých vzorcích.
4. **path** - addressa složky kam budeme ukladat datové soubury.
5. **port** - port na kterém je připojen senzor. Standardně je nastaven '/dev/ttyAMA0'.
6. **baudrate** - rychlost komunikace s zařízenim. Standardně je nastavená 115 200.
7. **timeout** - timeout pro seriovou komuniakce. Standardně je 3.
8. **start_cmd** - řetězový přikaz, který zasiláme senzoru pro začatek měření. Tedy spouštíme senzor.
9. **stop_cmd** - přikaz, který zasiláme senzoru pro ukončení měření. Tedy zastavujeme senzor.
10. **file_name_format** - řetězec, který definuje v jakém formatu budeme generovat daty pro nazvy souvoru.

Nastavení kalibrace:

1. **comp** - the compensation field
2. **ofs** - offsets
3. **sen** - sensitivity
4. **ort_mat** - the orthogonalization matrix

4 Functions' documentation

Description of functions

calibration.**calibrate**(*data*)

This function calibrate data and return numpy array. (np is numpy)

Parameters **data** – is list with 3 items. For example: np.array([123, 456, 789])

Returns np.array([111, 444, 777]) array with calibrated data

calibration.**find_mean**(*data*, *gauss*)

This function apply gauss filter to data and then calculate mean value. And multiply it by 2.

Parameters **data** – np.array([[H], [Z], [E], [T]]). H, Z, E and T are arrays

Returns np.array([[H], [Z], [E], [T]])

calibration.**parse_data_string**(*string*)

Cut string from sensors to separate values.

Parameters **string** – string in format like this '12345671234567123456712345671234567'

Returns [1234567, 1234567, 1234567, 1234567]

calibration.**save_data**(*data*, *path*, *suffix='*')

Save data to path/%Y-%m-%d_suffix.txt

Parameters

- **data** – some data (For example: 123 123 123)
- **path** – path where to save file (For example: ./data/)
- **suffix** – suffix for filename (For example for suffix 'original' filename will be 2015-12-24_original.txt)

class generator.**Generator**

Bases: object

This class is used only in 'Virtual' mode. In this mode we are don't have connected sensor and just generate random values for test program. So this class implements basic function of serial port.

read(*number_of_byte*)

This function return string in this format: 1234567;1234567;1234567;1234567; Numbers are just random.

Parameters **number_of_byte** – not used

Returns string

readline()

This function return string in this format: 1234567;1234567;1234567;1234567; Numbers are just random.

Returns string

write(*data*)

Just print received data.

Parameters **data** – string

processor.**process_data**(*pipeline*, *samples*, *path='./'*)

Process data from sensor. Accordingly get n samples and calculate average value from these samples. Then use Gauss filter and finally make calibration.

Parameters

- **pipeline** – pipeline where from this function will receive samples
- **samples** – number of samples per second
- **path** – path where we will save our files

reader.**init_communication**(*port='/dev/ttyAMA0'*, *baudrate=115200*, *timeout=None*)

This function create Serail communication object with default parameters:

Parameters

- **port** – serial port. By default '/dev/ttyAMA0'
- **baudrate** – By default 115200
- **timeout** – By default 'None'

Returns pyserial object to communicate with device

reader.**readline**(*inpoint*)

Read one line from sensor.

Parameters **inpoint** – pyserial object

Returns line or False

reader.**start_sensor**(*inpoint*)

This function send start command to sensor ('CN').

Parameters **inpoint** – pyserial object

Returns True if start is successful otherwise False.

reader.**stop_sensor**(*inpoint*)

Send stop command to sensor ('CS'). And return true if writing is successful.

Parameters **inpoint** – pyserial object

Returns True if stop is successful otherwise False.

saver.**data_saver**(*pipeline*, *path='./'*)

This function is run as a process and its save data getted from pipeline.

Parameters

- **pipeline** – pipeline
- **path** – path where to save data