**State change detections using the accelerometers with inbuilt filters at IOT platform**

Manoranjan Kumar ([makuaa@lnu.se](mailto:makuaa@lnu.se))

PhD student

In this IOT (Internet of things) course I decided to study its concept hence performing a small project in the frame of its end-to-end implementations. I choose to use accelerometers hence programming in microcontroller to detect the acceleration changes using filters to avoid noise detections or smoothing of dataset. Further I also like to visualize those changes to the IOT platforms. I am hoping this should not take more than 8 weeks of time for such investigations and implementations.

Contents

[**1.** **Objective** 3](#_Toc87208233)

[**2.** **Material** 3](#_Toc87208234)

[**3.** **Environmental setup** 4](#_Toc87208235)

[**4.** **Putting everything together** 5](#_Toc87208236)

[**5.** **Platforms and infrastructure** 6](#_Toc87208237)

[**6.** **The code** 7](#_Toc87208238)

[**7.** **The physical network layer** 8](#_Toc87208239)

[**8.** **The Visualization and user interface** 8](#_Toc87208240)

[**9.** **Finalizing the design** 10](#_Toc87208241)

[**10.** **References** 11](#_Toc87208242)

# **Objective**

Accelerometer is an important sensor in identifying the state of machines or equipment’s which includes changes in motions. This is also used widely into different domain such as automotive, house safety and health fitness etc. This can capture and identify the states of any equipment’s it is attached depending upon the definitions of state detections. In this project I tried to investigate how I can capture the states changes of the accelerometers depending upon the changes it is experiencing. The detection of states also depends on what kind of filters are applied into the accelerometers original data. Filters are necessary to avoid the noises, smoothing or detecting a particular state of the systems. Hence by doing this project it will enables to deploy filter algorithms and detecting the states due to changes of accelerations. Once the states are detected further insights can be known of the attached equipment’s related to different controls if any.

# **Material**

The following type of materials are brought to perform this project from different places as shown below in Table 1.

Table 1. List of materials in doing this project

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Material** | **Purchased Place** | **Cost** | **Unit** | **Specifications** |
| Expansion Board 3.0 | https://pycom.io/ | 18 | Pound | USB and LiPo battery powered |
| LoPy4 - With Headers | https://pycom.io/ | 39 | Pound | With network protocol such as Bluetooth, Wi-Fi, LoRa and Sigfox |
| Custome Tax | https://pycom.io/ | 428 | SEK |  |
| shipping cost from UK | https://pycom.io/ | 25 | Pound |  |
| ADXL345 from adafruit | [electrokit.se](mailto:info@electrokit.se) | 166 | SEK | I2C and SPI interfaces, 3.3v, sensitivity of +/- 2G to +/- 16G |
| PCboard | [electrokit.se](mailto:info@electrokit.se) |  |
| USB cable | [electrokit.se](mailto:info@electrokit.se) | micro-B of length 1.8m |
| Jumpers | amazon.se | 80 | SEK |  |
| LoRa anteena | Kjell & company | 100 | SEK | 824-960 Mhz (GSM, LoRa etc), 1710-2200 MHz GSM. |

Further the design of the project is shown below in Figure 1 with the details of materials.

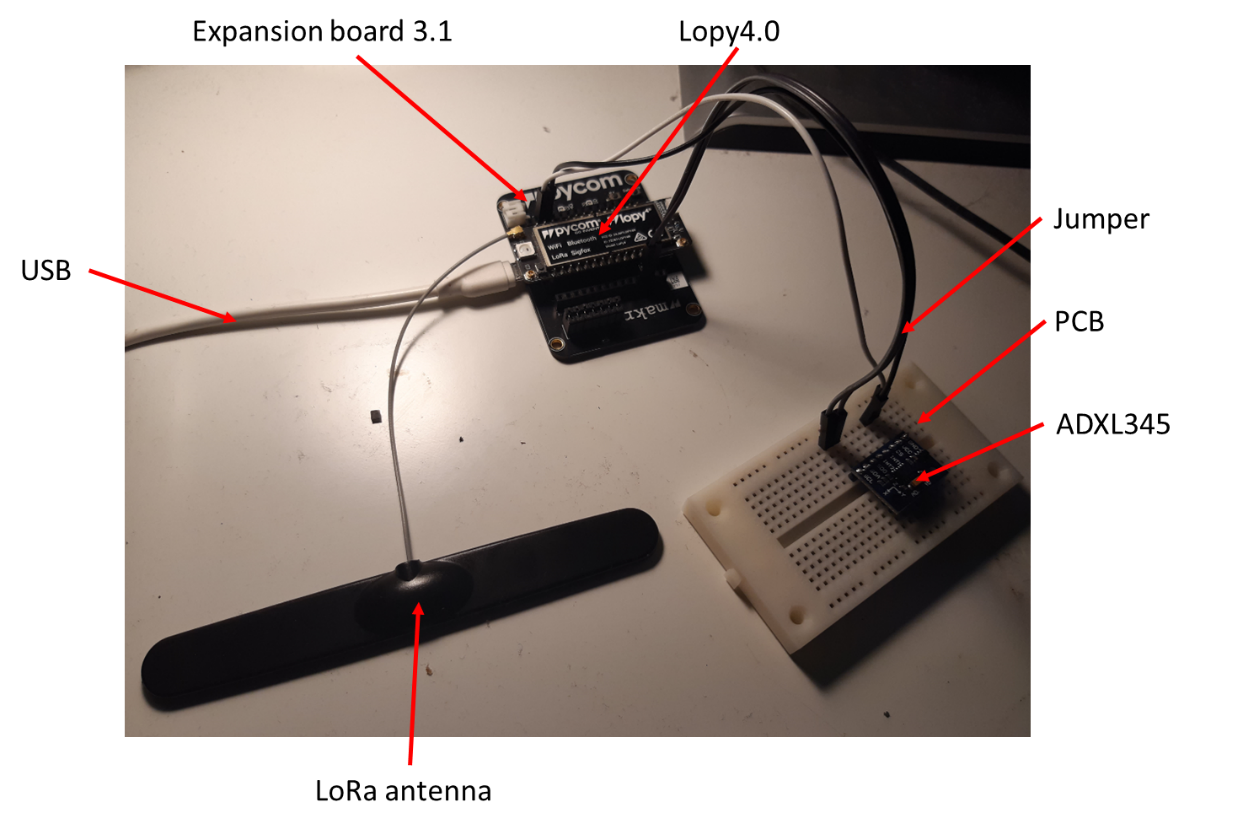


Figure 1. The materials used in designing this project.

# **Environmental setup**

To setup the lopy4.0 the software for e.g. node.js and Atom [1] are installed into the windows environments. Once these programs were installed the lopy4.0 is detected by the atom who gave the specification of the lopy4.0 as shown below Figure 2 .

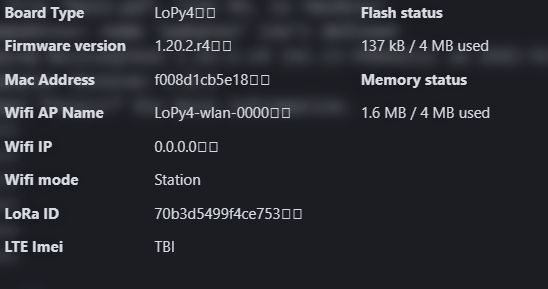


Figure 2. The Lopy4.0 configurations into Atom.

It was not needed to flash the latest firmware but still steps are taken to reflash the latest firmware. The flash has been performed by installing ParticleCLISetup.exe files using the instruction from [2].

Once the lopy4.0 is configured to the system further it is registered to the pybytes [3] for setting up the wi-Fi connections and LoRa for data transfer. This is done under my device sections to the pybytes GUI as shown below in Figure 3.

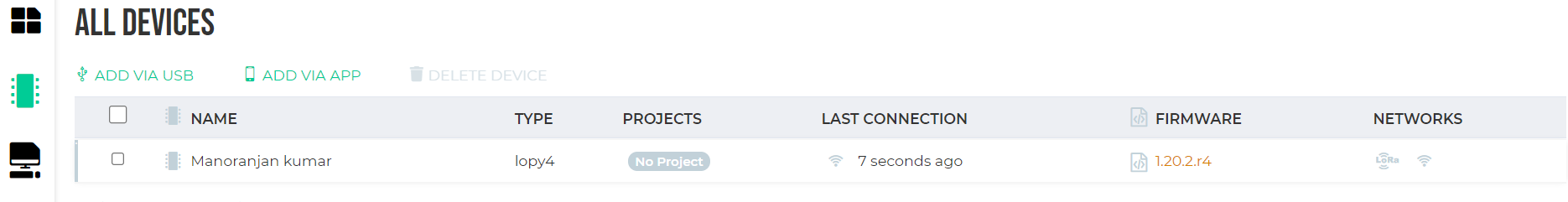


Figure 3. The registration of pycom and setting up the GUI.

# **Putting everything together**

To connect the accelerometer ADXL345 following wiring diagram are used as shown below in Figure 4. As per the definitions of ADXL345 3.3-5 volts are needed while lopy4 has output of 5 volts hence I did not find any problems in connecting directly.

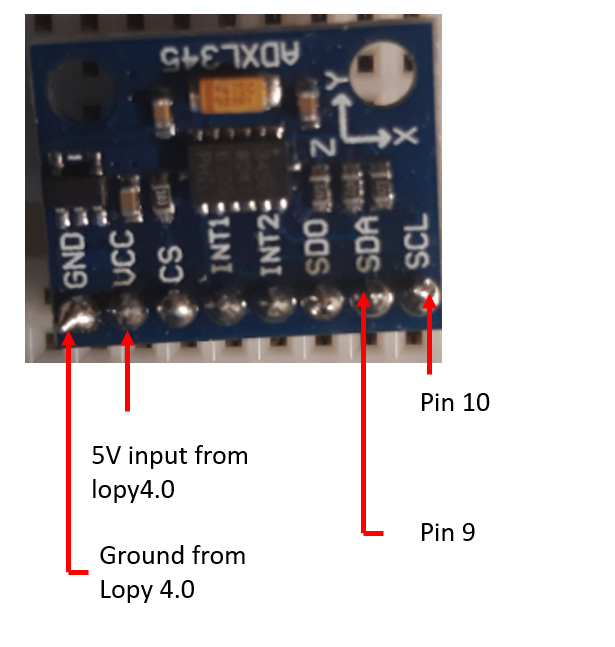


Figure 4. The circuit diagram of ADX345.

This is only the development kit but to scale it for any specific problem can be extended easily.

# **Platforms and infrastructure**

IOT (Internet of things) platforms needs to handle 5 major functionalities as follows [4].

1. connecting to the devices
2. securing the data
3. Managing data
4. Supporting analysis of the data
5. Enabling developers to create code and applications that interact with the IoT system and integrate with broader enterprise systems

Lopy4.0 connects the accelerometer and processed data to the pybytes platform using the Wi-Fi protocol. I investigated data collections at higher sampling frequency which is of 10Hz using the Wi-Fi protocol. It worked quite well. It was not possible to send data at such higher rate using the LoRa protocol. LoRa protocol works at rate of usually 30 seconds.

In the market there are many platform providers such as Azure IoT Hub from Microsoft, AWS IoT Core and the Cloud IoT Core from Google are among the dominant offerings in the market. SAP, Oracle and Siemens also have IoT platforms.

The prices for Azure ranges from 85 SEK to 4300 SEK [5] per devices and size of datasets in basic set up configurations as shown below in Figure 5. The different protocols for messages transfers are HTTP, AMQP and MQTT.

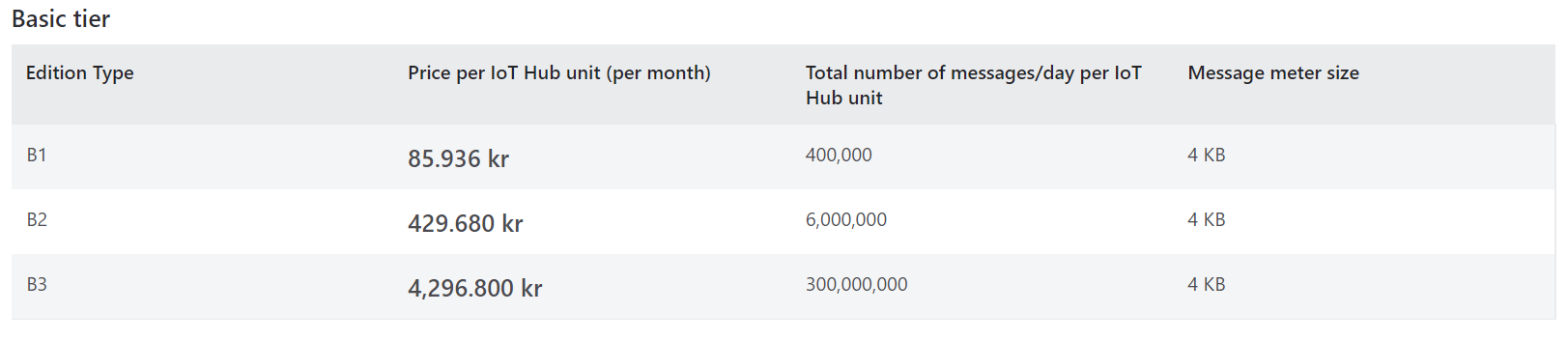


Figure 5. The price ranges for the Azure platform.

The prices from the AWS platform [6] are basically based on the usages calculations for one single device are shown below for collecting data at rate of 10Hz.

Published message count: 36000 message/hour \* 24 hours/day \* 30 days = 25million messages

Published message charges: 25million messages \* $1/25,000,000 messages = $25 ~~ 225 SEK

It is obvious that for each device if data are collected even at higher rate of 10 HZ prices are relatively not much. In case of developing such low code for data collection might requires additional learnings so at this point of time it does not seems a viable option. Whereas readily available options also provide visualization and analytics to identify the insights of the datasets,

I choose to go with pybytes which is free in my case as there exists one year subscription when I bought the plopy4.0.

# **The code**

The code consists of following steps.

1. Importing the needed libraries

* import time
* import math
* import pycom
* import crypto
* from ADXL345 import ADXL345
* from ADXL3452 import ADXL3452
* from machine import I2C

1. ADXL345 library has been imported and converted the data into units of ‘g’, a snippet of code is shown below. The port address for reading is 53 with I2C protocols, other parameters used are shown below.
   * + BW\_RATE\_100HZ = 0x0B
     + POWER\_CTL = 0x2D
     + MEASURE = 0x08
     + DATA\_FORMAT = (0x31)
     + AXES\_DATA = 0x32
     + BW\_RATE = 0x2C
     + RANGE\_2G = 0x00
     + SCALE\_MULTIPLIER = 0.004
     + EARTH\_GRAVITY\_MS2 = 9.80665
     + TO\_READ = 6
2. The filter coefficients are obtained using the Butterworth principle using 3rd order smoothing. The details of poles and zeros are shown below. In general, 4 data points are needed to predict the filtered steps while filtering the raw data.

Poles(a) = [0.0181,0.0543,0.0543,0.0181]

Zeros(b) = [1.000,-1.7600,1.1829,-0.2781]

1. Implementation of filters and data sending code to pybytes platform is shown below in Figure 6.

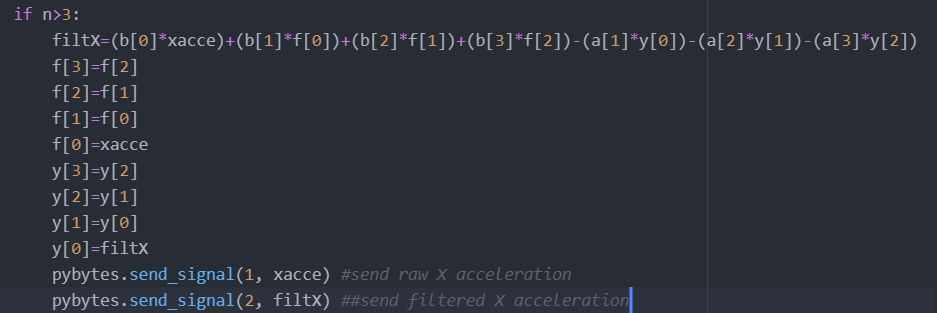


Figure 6. The snippets of filter implementations in code.

Whereas on pybytes platform data signal no. 1 and 2 are mapped accordingly.

# **The physical network layer**

The data is transmitted using the Wi-Fi to the pybytes server. 8 bytes of data are sent each time at sampling rate of 10Hz. The filters are implemented in the lopy4.0 hence raw data and filtered data’s are send together. The filtered signals are verified using the Matlab code [7] as well, shown below in Figure 7.

Graphical user interface

Description automatically generated

Figure 7. The filter comparison accuracy with Matlab code.

Overall implementations of filters into the lopy4.0 works as expected. As it can be seen in above Figure 7 the lopy4.0 filters follows the Matlab inbuilt implementations. At this moment data are not wrapped to send it at lower transferring rate but that can be done later to save the battery usages. It is observed that LoRa protocols cannot be used as that has a requirement of sending data at a rate of 30 seconds. Hence for my case I needed to choose wi-Fi protocols since I am sending data at rate of 0.1 seconds.

# **The Visualization and user interface**

Pybytes interface is chosen to visualize the raw data and filtered data of accelerometers in X and Y directions. The data is stored there for a month. Dashboard consists of three parts mainly.

Part 1 consists of providing details for data received and positions of the sensors as shown below in Figure 8.

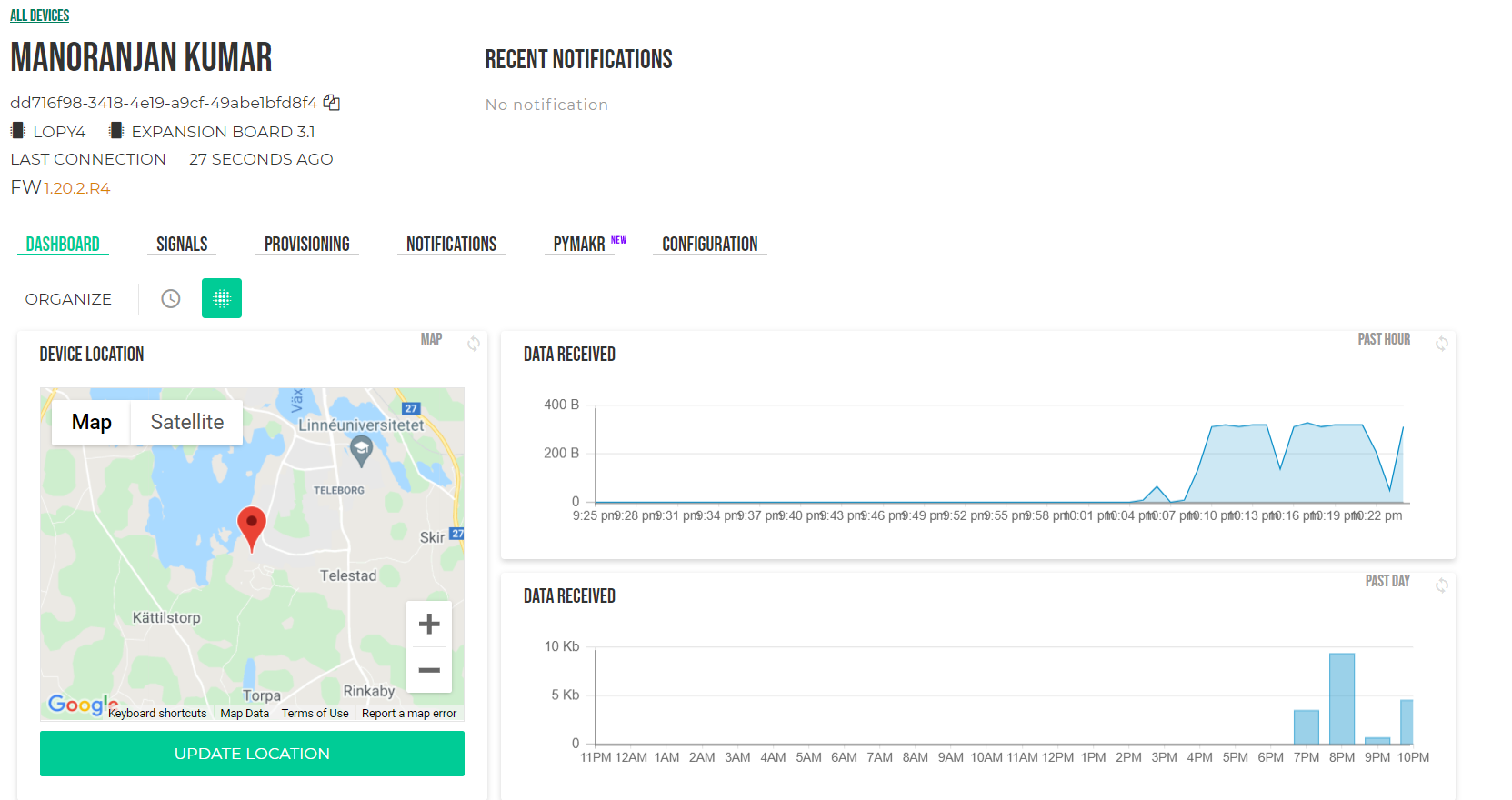


Figure 8. The position of sensor and size of data received.

Part 2 consists of raw data in X and Y accelerations and filtered X and Y accelerations. The data is plotted using the line plots. The dashboard also consists of a table providing value of X accelerations, shown below in Figure 9.



Figure 9. The transferred raw and filtered data to pybytes.

Part 3 consists of mainly counts for exceeding a certain value of X and Y direction over filtered accelerations as shown below in Figure 10.

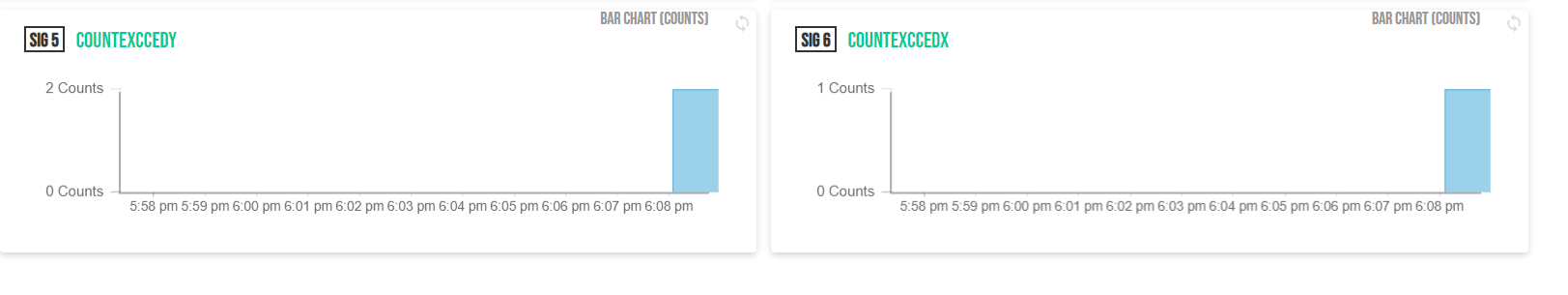


Figure 10. The counts of filtered data in bar plot.

# **Finalizing the design**

The expectations of performing this project were quite clear to understand the concept of IOT and implement the filtering algorithm into the microcontroller.

Using the pycom microcontroller and pybytes dashboard, the filtering algorithm is implemented, and processed data are visualized successfully respectively. The microcontroller also changes color if filtered X or Y acceleration value exceeded set threshold as shown below in Figure 11.

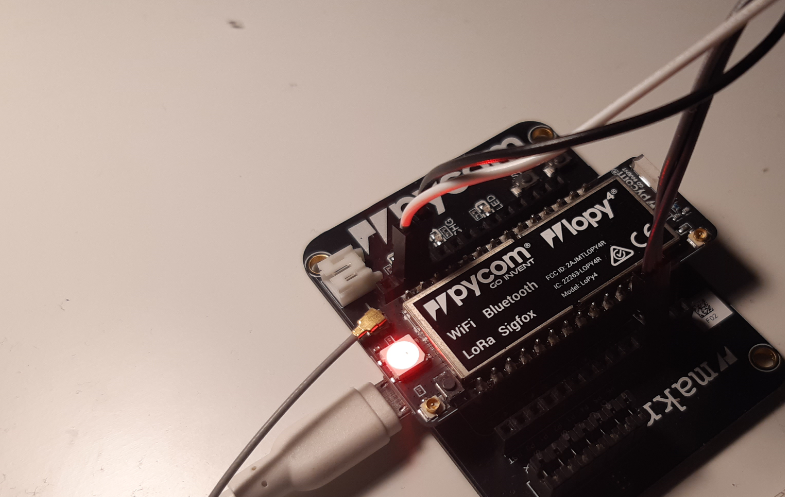


Figure 11. The change in LED display over lopy4.0.

In the end also different counts of those excessive values are also visualized as shown below in Figure 12.

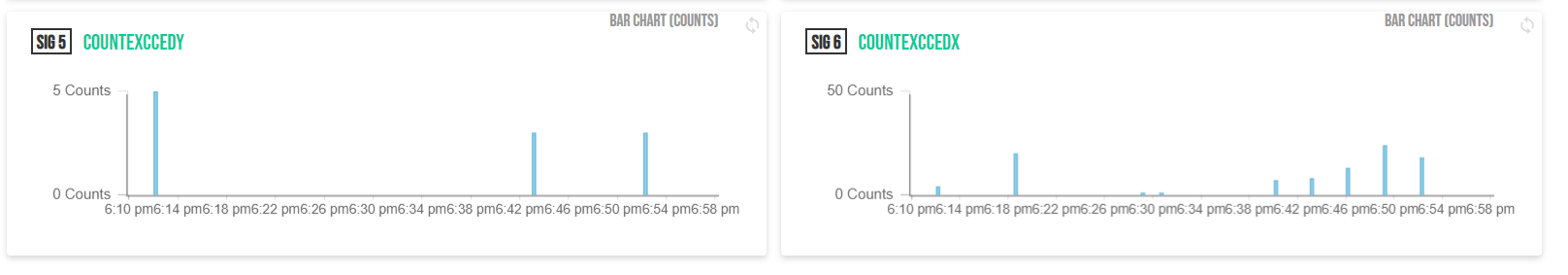


Figure 12. The counts of filtered data in bar plot over time.

I have not had possibilities to investigate other platform such as google sheet to visualize received data. In the coming days I would like to perform that to increase the effectiveness of visualizing dataset with other graphs. I also like to implement few machine learning algorithms into the microcontroller to understand the capability of such processors. So far, I am also not completely convinced with ADXL345 accelerometer sensors hence I also like to change this sensor with other accurate sensors to increase the accuracy of the accelerometers in state predictions.

# **References**

1. <https://atom.io>
2. <https://www.instructables.com/Install-Particle-CLI-Photon-Core/>
3. <https://pybytes.pycom.io/>
4. <https://internetofthingsagenda.techtarget.com/tip/IoT-platforms-How-they-work-and-how-to-choose-one>
5. <https://azure.microsoft.com/en-us/pricing/details/iot-hub/>
6. <https://aws.amazon.com/iot-core/pricing/>
7. <https://se.mathworks.com/products/matlab.html>