University of Southern Denmark

Ubiquitous Computing &
Internet of Things

Project report

Library Usage with Sensors

Authors:
Mohebullah Toofan
Nasib Sarvari
Štefan Töltési

May 25, 2018

Contents

1 Introduction

The cost of being able to connect to the internet, and the cost of devices with Wi-Fi capabilities and sensors built in, is decreasing. This means that an increasing number of devices are connecting to the internet. It is a network of hardware which are uniquely identified within the network [1]. This is called Internet of Things.

1.1 Context

Internet of Things is used in many different areas to connect and exchange data. One of these hardware devices is Pycom LoPy [2] with a Pycom Pysense [3] attached. Pycom LoPy transmits data through Wi-Fi, Bluetooth, Sigfox and more. Pycom Pysense has different sensors such as 3-axis accelerometer, ambient light sensor and many more.

This project's goal is to design and implement an Internet of Things system for collecting data with sensors.

1.2 Problem

The library of the University of Southern Denmark has facilities that are used by students every day. One of these facilities is the 24/7 reading room where students can go to study. The room is directly connected to a heavily used corridor, Gydehytten, as seen in Figure 1.

Currently the library does not have any way of measuring how the room is used. This is an issue, because data about the room could help improve the quality of the room. The room is popular among students and therefore it is important to get a knowledge on how the room is used.

Currently an IC-Meter is installed in the reading room. IC-Meter is a device that collects data with sensors about the indoor climate, such as temperature, humidity and noise levels [5]. This is a stationary device mounted near the main entrance of the room. The IC-Meter does not gather data about the usage of the room, but mainly the indoor climate.

1.3 Related Work

A research about a large-scale Wi-Fi monitoring system performed by Mikkel B. Kjærgaard et al. addresses how data is collected to provide information

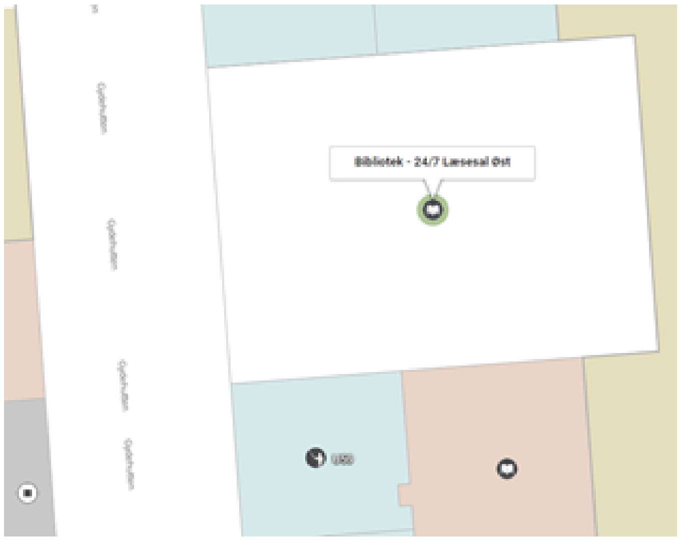


Figure 1: 24/7 Reading room directly connected to Gydehytten [4]

on how a building is used, for example the most used entrances [6]. This is carried out by collecting Wi-Fi traces. In this research the privacy of the user is considered and shows how important it is to include privacy. This research also shows how the data is visualized to make the data readable and understandable.

1.4 Approach

In order to collect data, many things must be planned, such as what, how and when to collect the data.

The reading room has many spots, where relevant data could be gathered, but noise is relevant to measure in this case combined with usage statistics regarding how the room is used. Therefore, the group decided to mount the Pycom on the main entrance door and collect data with the accelerometer.

By using one of the sensors from the Pysense board and combining it with IC-Meter noise level data, the group can gather knowledge on how the noise levels change, as the door opens.

1.5 Report Structure

Here are the different sections of the report described to give the reader a quick overview of the report.

Analysis States the different approaches discussed by the group and the reason behind choosing the approach that the group have chosen.

Design Describes the design of the approach chosen by the group, with little consideration as far as programming language and other technicalities.

Implementation Describes the implementation of the group's approach in detail, with descriptive figures and relevant code snippets.

Evaluation The approach and the results of the data analysis are evaluated in this section.

Discussion Challenges and difficulties throughout the project are discussed in this section.

Future Work Changes and improvements of the Internet of Things system designed by the group are addressed in this section.

Conclusion Group presents what can be learned from the experiment and sums up the whole project work.

2 Analysis

In this section, the approaches discussed by the group will be presented. Each approach will be described with it's pros and cons.

When deciding on which approach the group wanted to take with their solution, the group started by brainstorming different ideas. All the ideas were then written into a table, and the pros and cons of each idea written alongside. Based on the pros and cons the group could then choose a solution that all the members were satisfied with. Table 1 shows the possible approaches with their respective pros and cons.

The group ended up choosing the last approach, as the cons were not concerning the setup of the experiment, but the data analysis. With this in mind the group all agreed that using the accelerometer to measure the usage of the room, and comparing that data with noise data would be the optimal approach.

Approach	**Pros**	**Cons**
Measuring temperature and comparing with noise data	Temperature is easy to configure and measure with Pysense	Temperature does not change very often. Pysense board is inaccurate with temperature as the sensor gets warmer with time.
Measuring ambient light and comparing with noise data	Ambient light output is easy to read and understand.	Too many external factors can affect the light level measured.
Using ambient light sensor by the door to count number of students using the room and comparing with noise data.	Ambient light output is easy to read and understand.	Too many external factors can affect the light level measured.
Using accelerometer on the door to measure usage of room and comparing with noise.	Accelerometer data is very accurate. Setup of the device is simple.	Data analysis is difficult, because the accelerometer provides many different numbers.

Table 1: Ideas and their pros and cons

3 Design

In this section the design of the experiment is described. A deployment diagram, Figure 2, shows the structure of how the group intended the experiment to take place.

3.1 Setup

The group intended to have the structure of the experiment split into three pieces.

One part would be the Pysense board, one part some sort of Server, preferably Python, because Pycom is programmed with Python, and the third part would be the method of visualizing the data.

For the Pysense device to send data to the server, some sort of communication method must be used. The Pysense board included a LoPy board which allowed for the use of Wi-Fi, LoRa, Bluetooth, and Sigfox.

As the group wished to use a webserver, the use of Wi-Fi appeared to be the obvious choice.

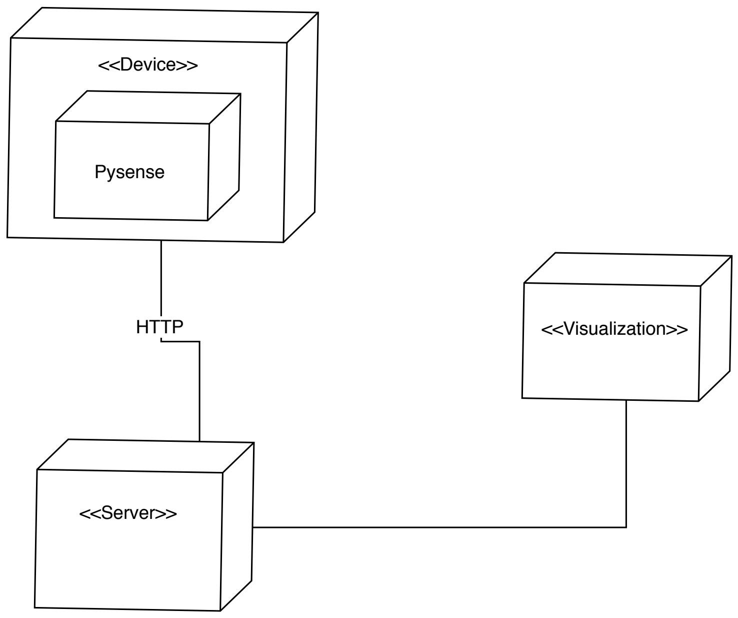


Figure 2: Deployment View

4 Implementation

This section describes which technologies are used and how they are implemented. It covers the implementation of using Google Sheets, using Wi-Fi on the Pysense board, and the Python Server used by the group.

4.1 Technologies

The used technologies are described below.

4.1.1 Google Sheets

The group has used Google Sheets [7] for visualizing the data, as it supports live data insertions within the diagrams created. With Google Sheets the group is also able to generate multiple different diagram types that can display various data in various ways.

4.1.2 Wi-Fi

The LoPy boards supplied to the group have four different built in communication capabilities; LoRa, Wi-Fi, Bluetooth, and SigFox. Wi-Fi was used as the communication technology between the server and the Pysense boards. This was done due the server running on the internet and the use of HTTP methods was needed. The server and boards must both run on the same local network, in order for the IP configuration for either to work.

4.1.3 Python Server

To handle the incoming data from the Pysense boards, a server was needed. Since the group had implemented a simple Python class to insert data into a Google sheet, the group decided, that the server should be implemented in Python.

4.1.4 R

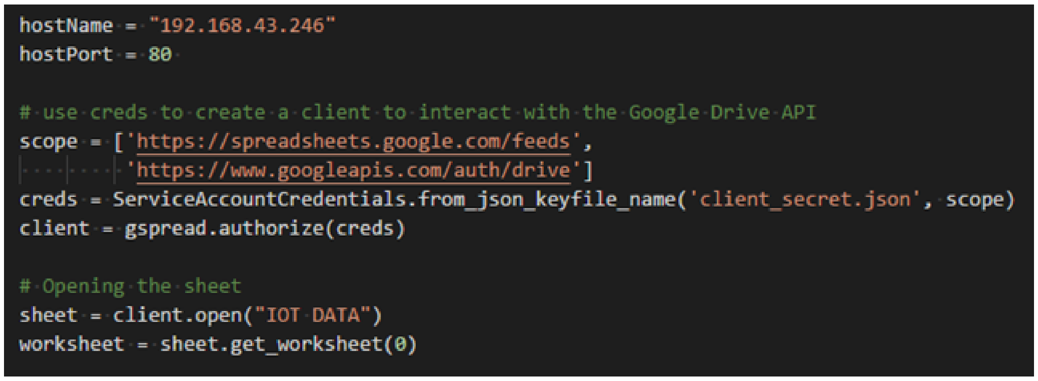
R is a language for statistical computing and graphical visualization of data [8]. R provides a wide variety of statistical tools, with a large open source community. The group used R for analyzing data. Further details can be found in the Evaluation section.

4.2 Server

This section explains the code used for the server. The server is a simple Python HTTP web server which can only handle POST requests.

Figure 3 shows the host IP of the host is set along with the port. When setting the scope of the project, the authentication is set to the Google drive, so that the server has permission to write to the Google Sheet. To gain access to the Google Sheet via the python server, Google Drive API must be enabled through Google's own control center. As seen in Figure 4 the Google Drive API along with the Google Sheets API are enabled. This allows the user to create a service account for these services, which is then used for authentication.

The service account's credentials are stored in the file *client_secret.json*. In the server these credentials are loaded in via the file and used to authorize the user's access to the requested service. The bottom two lines seen in

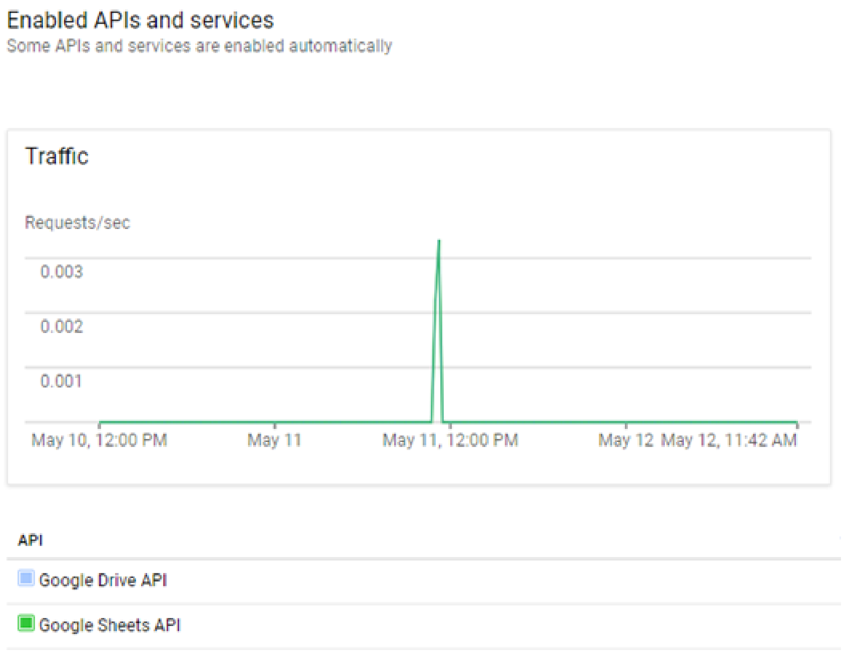


```
hostName = "192.168.43.246"
hostPort = 80

# use creds to create a client to interact with the Google Drive API
scope = ['https://spreadsheets.google.com/feeds',
         'https://www.googleapis.com/auth/drive']
creds = ServiceAccountCredentials.from_json_keyfile_name('client_secret.json', scope)
client = gspread.authorize(creds)

# Opening the sheet
sheet = client.open("IOT DATA")
worksheet = sheet.get_worksheet(0)
```

Figure 3: Server setup

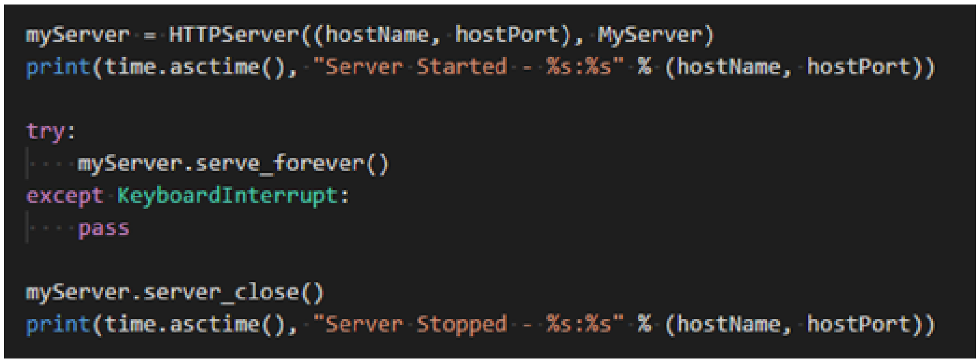


Enabled APIs and services

Some APIs and services are enabled automatically

API
🟦 Google Drive API
🟩 Google Sheets API

Figure 4: Google API Dashboard showing requests sent to the API's

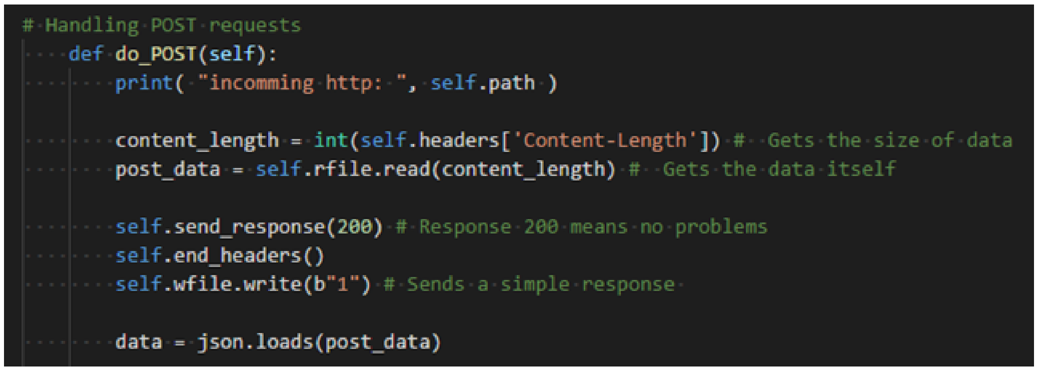


```python
myServer = HTTPServer((hostName, hostPort), MyServer)
print(time.asctime(), "Server Started - %s:%s" % (hostName, hostPort))

try:
    myServer.serve_forever()
except KeyboardInterrupt:
    pass

myServer.server_close()
print(time.asctime(), "Server Stopped - %s:%s" % (hostName, hostPort))
```

Figure 5: Server controls for starting and stopping the server



```python
# Handling POST requests
    def do_POST(self):
        print( "incomming http: ", self.path )

        content_length = int(self.headers['Content-Length']) #  Gets the size of data
        post_data = self.rfile.read(content_length) #  Gets the data itself

        self.send_response(200) # Response 200 means no problems
        self.end_headers()
        self.wfile.write(b"1") # Sends a simple response

        data = json.loads(post_data)
```

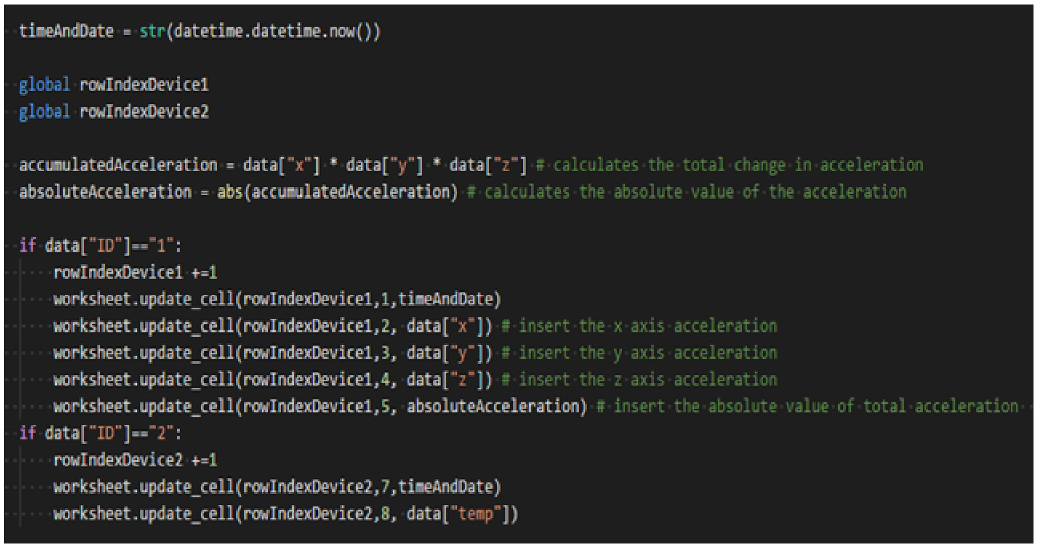
Figure 6: Handling POST request

Figure 3 specify which sheet to open, and which worksheet within that sheet [9].

Figure 5 shows simple Server code for starting and stopping the server. The server is set up with a KeyboardInterrupt, that allows the user to stop the server at any time.

When handling POST requests from the client numerous things are done in the server, before the data can be processed. The server confirms to the client, that the POST request has been successful, by sending back response 200. As the client expects some sort of response package, a simple byte, 1, is sent back to the client. The data sent from the client is then loaded into the data object as seen in the bottom of Figure 6.

Figure 6 is code for handling the data sent from the client to the server. As seen in Figure 7 the data is stored in the data object, which consists of



```
timeAndDate = str(datetime.datetime.now())

global rowIndexDevice1
global rowIndexDevice2

accumulatedAcceleration = data["x"] * data["y"] * data["z"] # calculates the total change in acceleration
absoluteAcceleration = abs(accumulatedAcceleration) # calculates the absolute value of the acceleration

if data["ID"]=="1":
    rowIndexDevice1 +=1
    worksheet.update_cell(rowIndexDevice1,1,timeAndDate)
    worksheet.update_cell(rowIndexDevice1,2, data["x"]) # insert the x axis acceleration
    worksheet.update_cell(rowIndexDevice1,3, data["y"]) # insert the y axis acceleration
    worksheet.update_cell(rowIndexDevice1,4, data["z"]) # insert the z axis acceleration
    worksheet.update_cell(rowIndexDevice1,5, absoluteAcceleration) # insert the absolute value of total acceleration
if data["ID"]=="2":
    rowIndexDevice2 +=1
    worksheet.update_cell(rowIndexDevice2,7,timeAndDate)
    worksheet.update_cell(rowIndexDevice2,8, data["temp"])
```

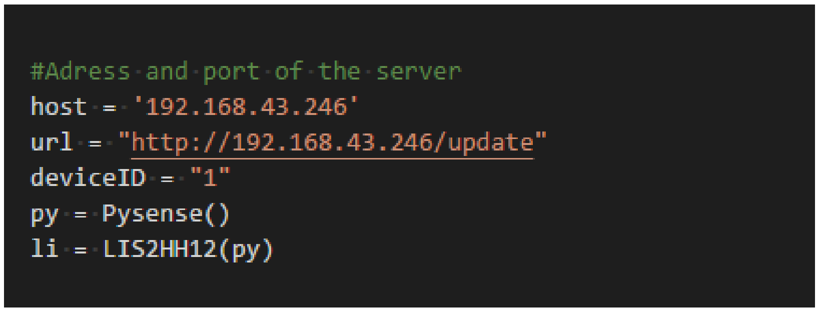
Figure 7: Inserting data to Google Sheets

key-value sets, for various information. As the values of acceleration in the different axis are very low, the product of all three values are calculated. The absolute value of this calculation is then needed, as the acceleration values can be negative as well as positive.

The group wanted to keep all the values positive, as it was easier to show change in value with purely positive values.

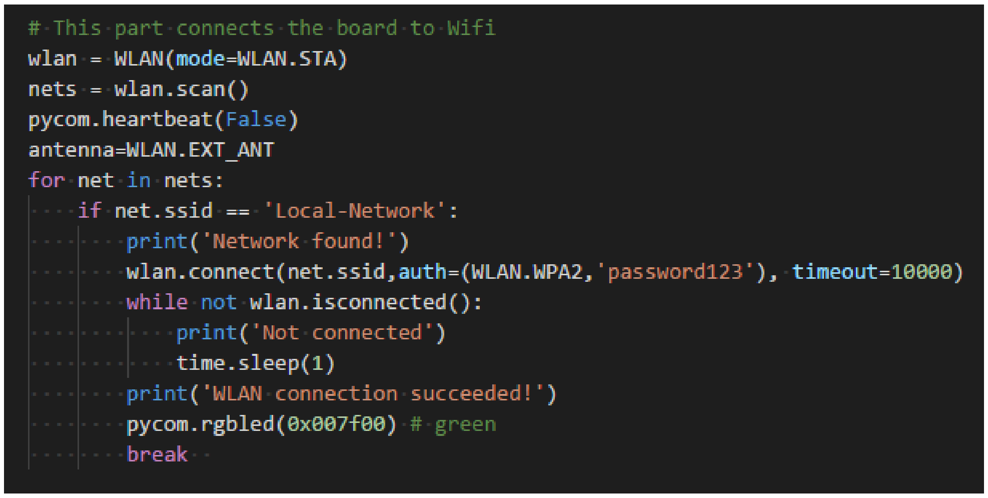
4.3 Extensibility

The two *if-statements* at the end, show how the server can differentiate between the multiple devices/clients, sending data to the server at the same time. This solution requires each device/client to be hard-coded with an ID, and it enables more devices to be connected to the server at once. The connected devices can use different sensors. In this code, it is shown, that the device with the ID 1, collects accelerometer data, and the device with ID 2, temperature. Temperature data was not used by the group, but serves only as an example of how to collect data with multiple devices. For every case, the server first inserts a timestamp into the Google Sheet, and then the data alongside in separate columns.



```
#Adress and port of the server
host = '192.168.43.246'
url = "http://192.168.43.246/update"
deviceID = "1"
py = Pysense()
li = LIS2HH12(py)
```

Figure 8: IP, URL, ID and sensor setup



```
# This part connects the board to Wifi
wlan = WLAN(mode=WLAN.STA)
nets = wlan.scan()
pycom.heartbeat(False)
antenna=WLAN.EXT_ANT
for net in nets:
    if net.ssid == 'Local-Network':
        print('Network found!')
        wlan.connect(net.ssid,auth=(WLAN.WPA2,'password123'), timeout=10000)
        while not wlan.isconnected():
            print('Not connected')
            time.sleep(1)
        print('WLAN connection succeeded!')
        pycom.rgbled(0x007f00) # green
        break
```

Figure 9: Wi-Fi connection setup

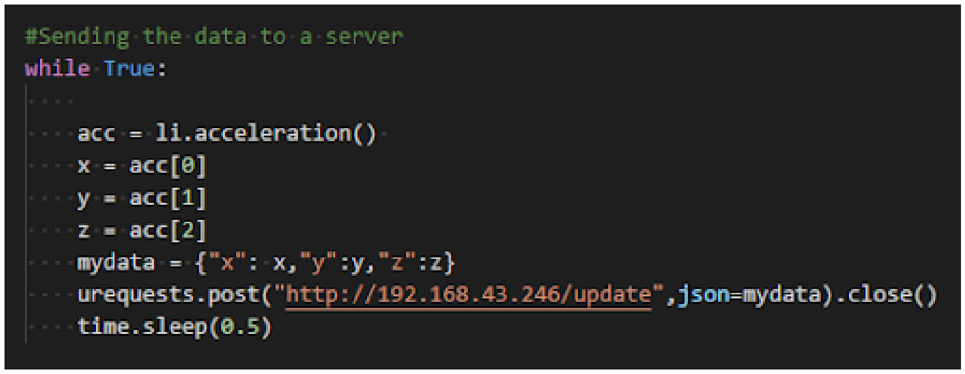
4.4 Client

This section describes the code that is uploaded onto the Pysense boards, using Pymakr. Pymakr was used to connect to the Pysense boards, through the Visual Studio Code IDE [10].

The first thing that is done in the client code is setting up standard values and getting the needed sensors from the libraries included. This is seen in Figure 8, where the server's address/IP is stored along with the URL that the client will use to send POST requests.

LIS2HH12 is the accelerometer on the Pysense board. An instance of this, *li*, is made which is then used later to get accelerometer data [11].

Figure 9 shows the code needed to establish a Wi-Fi connection on the



```
#Sending the data to a server
while True:

    acc = li.acceleration()
    x = acc[0]
    y = acc[1]
    z = acc[2]
    mydata = {"x": x,"y":y,"z":z}
    urequests.post("http://192.168.43.246/update",json=mydata).close()
    time.sleep(0.5)
```

Figure 10: Accelerometer data sent to server

Pysense board. First the WLAN mode of the board must be set to STA to indicate that it is a WLAN station. This can also be set to AP, which then indicates that the board is being used as an access point. The Wi-Fi is then scanned, and every network visible is stored in nets. As the Pysense board the group used came with an external antenna, this was utilized through code [12].

After the visible networks have been saved, a *for-loop* iterates through them, and checks if one of the networks matches the network specified in the code as “Local-Network”. When this network is found the board then attempts to connect to the network. In this case the network was password protected. This password is specified in the *wlan.connect()* method, as *WLAN.WPA2,’password123’*. This method is also given a timeout of 10 seconds, so that the board has enough time to attempt a connection. When the board makes a successful connection to the network, the LED light on the board is set to turn green.

All the print methods in this class are for debugging purposes and will only show up if the board is connected to a pc.

When the board has connected to the Wi-Fi network, it moves on to sending data to the server, this is seen in Figure 10. As sending data is a continuous task, this is all run inside a while-loop. First thing that happens in the *while-loop*, is that the accelerometer data is gathered and stored as *acc*.

Because the accelerometer collects multiple different data, x, y, and z, *acc* becomes an array of numbers. The specific data is then extracted from *acc* by using the different indices, x being stored in the first spot, and so on. All

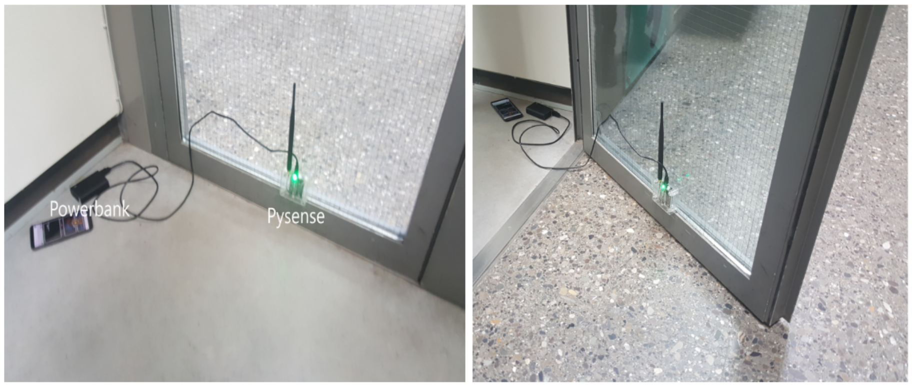


Figure 11: Pysense mounted on the door of the 24/7 reading room

this data is then saved as key-value pairs, as *mydata*. This is done, so that the data can be sent with the HTTP POST request, as a JSON object.

For doing HTTP requests, a library was used called *urequests* [13]. To not have issues with the server, the connection is closed after every request is sent. *Time.sleep(0.5)* indicates that the data must be sent to the server every half a second.

4.5 The Experiment Setup

The setup of the experiment, is described in this section, along with pictures taken of the setup by the group.

Illustrated in Figure 11 is how the group positioned the Pysense board on the door. A Powerbank was needed as there was no access to power outlets near the door. The Pysense board was mounted near the bottom of the door, to minimize the risk of damage, should it accidently fall off.

5 Evaluation

The group collected two datasets for the experiment. First one was accelerometer output that was measured every half a second. The second one was a noise level average that was measured every five minutes from IC-Meter stationed in the library. For data analysis R language was used.

Accelerometer measured data according to a 3-way axis. In the group's case, the y-axis was omitted because the device was mounted on a door in a

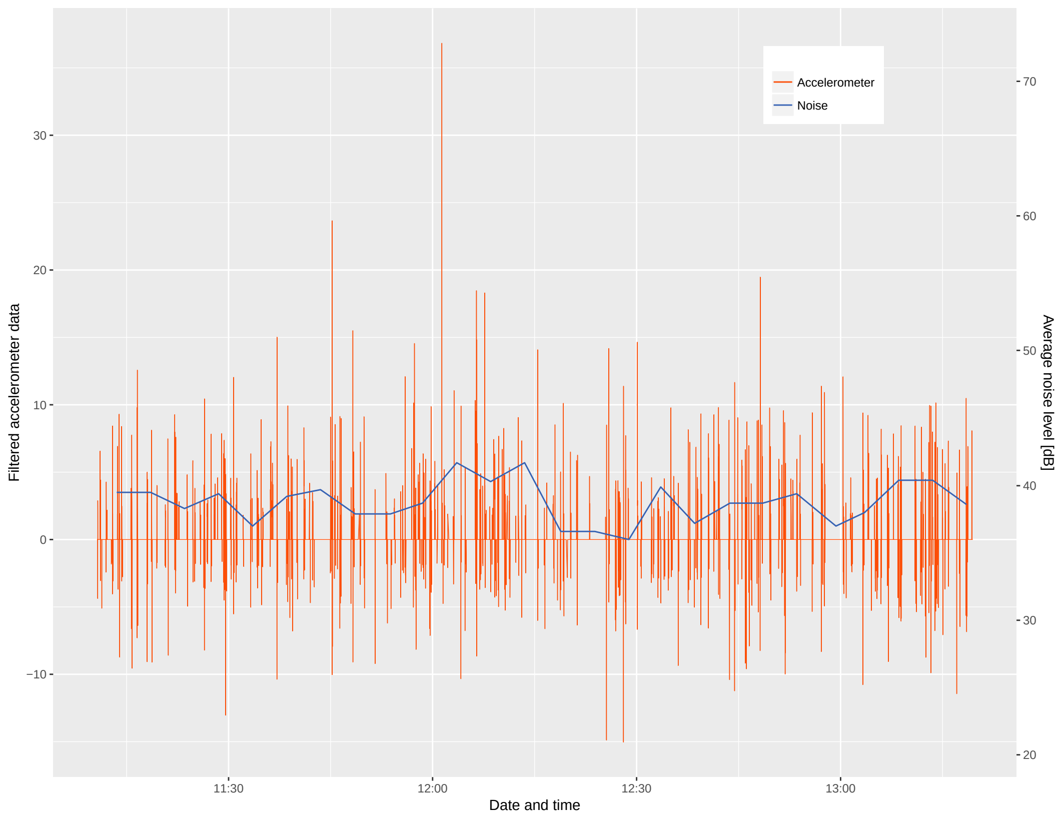


Figure 12: Plot showing filtered accelerometer and noise level data

way where y-axis is perpendicular to a plane for opening doors. Y-axis was measuring only gravitational acceleration and that was not relevant for the group's case.

For visualization of accelerometer data the group scaled the data of the x and z-axis and then added them together.

$$Accel_{unfiltered} = x_{scaled} + z_{scaled}$$

Next step was filtering out noise from the accelerometer data. Simple filter is used for data transformation. Filter checks if a value fulfills a condition and hereby filters the data.

$$|Accel_{unfiltered}| < \sigma_{data}$$

If a value is smaller than standard deviation (σ) then the value is replaced with the mean of all the values . In the group's case mean is equal to 0 and

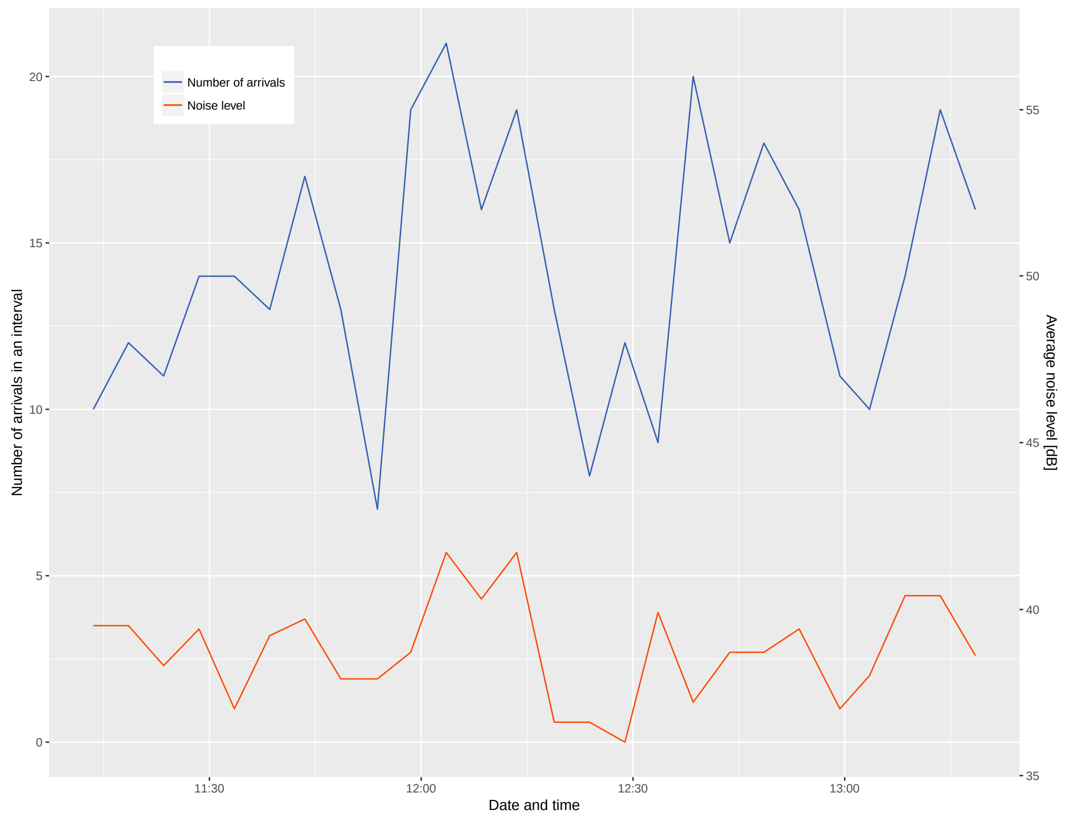


Figure 13: Plot showing how many times door opened and an average noise level in a specific interval

standard deviation is 1 since the values has been scaled. Figure 12 shows that there are visible spikes on a accelerometer data when plotted. Counting how many times door opened in 5 minutes intervals followed. This was done to synchronize the accelerometer data with the average noise level. Open door is classified when there is no activity in previous n seconds. In Figure 13 parameter n is set to two seconds. While having the number of how many times door opened and average noise level we can do a correlation matrix. Pearson correlation [14] can be seen in Table 2. We can see that data are correlated but there should be more data collected to draw more concrete conclusions. There are also other factors that influence noise level. For example people walking around the IC-Meter or using the stairs where IC-Meter is located influences the measurement.

	Arrivals	*Noise Level*
Arrivals	1.0000000	0.4610963
Noise Level	0.4610963	1.0000000

Table 2: Correlation matrix of how many times door opened and an average noise level in a specific interval

6 Discussion

In this section alternatives solutions that the group discussed will be described, along with issues or difficulties with the experiment, the group discovered.

After reviewing the results of the experiment, it was clear to the group that many factors could be changed, to allow for a better overall experiment. When examining the position of the IC-Meter, of which the group used the data, it was clear to the group, that the positioning of the device was not optimal. The IC-Meter was located on the railing of the staircase, which lead up to the upper area of the studying room. This placement meant, that the IC-Meter was susceptible to react to vibrations coming from students walking up and down the stairs, and grabbing the railing. Another risk discovered by the group, was the possibility for the IC-Meter to be touched by students unknowingly, because of its positioning.

Another challenge the group discovered with using the data from the IC-Meter, was the rate of which it collects data. The IC-Meter is set up to collect data, at an unknown frequency, however that data is then averaged every 5 minutes, and only the average and peak values are stored. The data collection rate meant that it was impossible for the group, to see changes in values in real time. Instead the group was forced to wait for 5 minutes between readings, which opens the possibility for the data to be affected drastically in that time frame.

Alternatively, the group discussed using a second device, such as a mobile phone to measure the noise level in the study room, as this would give the group live readings and the desired frequency of data collection. The group found one app that recorded the decibel level using the phone's microphone, however data collected using the app was not useful for the group. The applications data collection rate could not be controlled by the group, and the data was saved with no timestamps, rendering the data unusable for the group.

The Pysense boards provided to the group, had the capability to use different communication methods. Wi-Fi was the group's first choice, as the setup the group wanted, included a web server. The use of the webserver limited the group to using Wi-Fi as the method of communication between the clients and the server. One more option the group discussed was using LoRa to communicate between two Pysense boards, using one of them for sensing, and one as a gateway, which would then send the data to the server. A setup using LoRa would enable the group to have the devices further apart and therefore mitigate the need for the server to be close to the devices.

For visualizing the data collected, the group used Google Sheets as the server already posted the data into a sheet on the service, and it allows for live updating diagrams. Another option for visualizing that the group looked into was using a simple website, hosted using the web server that the group had already implemented. Using a website would allow for either using traditional HTML and CSS [15] elements to visualize the data, or embedding Google Sheets into the website.

The group avoided using private data by collecting the accelerometer data for the main entrance of the 24/7 reading room. The noise level data collection with the phone was done using decibel numbers and no sound recordings were stored. When the data was transmitted to Google Sheets, an authentication was needed to access the data. The whole setup was in a password-protected local network, which the group had control of.

7 Future Work

The main area that the group saw room for improvement in, was the amount of data collected. The group believed for the data to be more insightful, measurements should be taken multiple times during a day, or even for 24 hours. This would ensure that the group could have data that represented the use of the study room for an entire day, and hereby draw conclusions on how much the room was used during different parts of the day. To get a bigger picture of how the room is utilized for an even longer period of time e.g an entire semester, measurements could be taken multiple days of the week.

For future measurements the group discussed the idea of using multiple devices, utilizing different sensors for measuring the environment in the study area, as this was taken into consideration of the future growth in the implementation. This way the group could see how the environment inside the room changes, as time goes on, and if the amount of people entering and leaving the room has any effect.

A final thing the group agreed should be implemented, is that the server and client are able to be on different networks and still communicate. As the implementation section discusses, the server and client must be on the same local network, as the setup does not allow for a remote server. Having the server on a different network would allow the group to measure data in the study room without being present, which significantly increases the practicality of the experiment. This approach of gathering data could then be scaled to other rooms.

8 Conclusion

Based on the results of the experiment done by the group, the group can conclude that the correlation between noise, and the amount of people entering and leaving the study room is noticeable. However the group cannot be certain that this correlation is mainly between students leaving and entering the room and noise, due to the amount of data the group had collected, and external factors being able to affect the data. The group experienced different aspects of working with Internet of Things devices, such as programming a device, and finding the optimal way to collect data with it. As the data was hard to read, the group had to figure out a way to filter out noise from the data, and use the data in a meaningful way, to be able to compare the collected data with IC-Meter data the group was given. When the data had been analysed, the group had to find the optimal way to visualize the data so the key information was being portrayed to the user. This was done using Google Sheets, as it provided a wide range of tools for handling and visualizing the data.

In summary, the group successfully provided usage statistics, both in numbers and with real-time graphs, for the library with an Internet of Things system and sensors. The group made the server extensible for future growth in the implementation as there are many improvements for the system.

References

[1] Dieter Uckelmann et al. An architectural approach towards the future internet of things, 2011.

[2] Pycom. Lopy, Accessed: 2018-03-16. URL `https://docs.pycom.io/chapter/datasheets/development/lopy.html`.

[3] Pycom. Pysense, Accessed: 2018-03-16. URL `https://pycom.io/product/pysense/`.

[4] SDU. Sdu map, 2018. URL `https://www.sdu.dk/da/service/vejviser/odense/sdumap`.

[5] IC-Meter. Ic-meter indoor climate, Accessed: 2018-03-23. URL `http://www.ic-meter.com/`.

[6] Mikkel B. Kjærgaard et al. Analysis methods for extracting knowledge from large-scale wifi monitoring to inform building facility planning, 2014.

[7] Google. Google sheets, Accessed: 2018-04-06. URL `https://www.google.dk/sheets/about/`.

[8] R-project. What is r?, Accessed: 2018-05-18. URL `https://www.r-project.org/about.html`.

[9] Greg Baugues. Google spreadsheets and python, Accessed: 2018-05-04. URL `https://www.twilio.com/blog/2017/02/an-easy-way-to-read-and-write-to-a-google-spreadsheet-in-python.html`.

[10] Visual Studio. Visual studio code, Accessed: 2018-05-04. URL `https://code.visualstudio.com/`.

[11] Pycom. Sensor demos, accelerometer, Accessed: 2018-05-04. URL `https://docs.pycom.io/chapter/tutorials/pysense/`.

[12] Pycom. Wlan, Accessed: 2018-05-04. URL `https://docs.pycom.io/chapter/tutorials/all/wlan.html`.

[13] Pfalcon. Micropython, Accessed: 2018-05-04. URL `https://github.com/micropython/micropython-lib/tree/master/urequests`.

[14] Lærd Statistics. Pearson product-moment correlation, Accessed: 2018-05-18. URL `https://statistics.laerd.com/statistical-guides/pearson-correlation-coefficient-statistical-guide.php`.

[15] W3C. Starting with html + css, Accessed: 2018-05-24. URL `https://www.w3.org/Style/Examples/011/firstcss.en.html`.