

IMPLEMENTATION OF AN INTELLIGENT MOTION DETECTOR

Taiwo Samuel Aina¹, Oluwaseun Olanrewaju Akinte², Babatunde Iyaomolere³, Ademola Emmanuel Tosin⁴, Innocent Iriaoghuan Abode⁵, Adetola John Awelewa⁶

¹School of computer Science and Technology, University of Bedfordshire, Vicarage St, Luton, LU1 3JU, United Kingdom

²Department of Energy and Materials Engineering, Rajamangala University of Technology, Thanyaburi, Pathum Thani 12110, Thailand

³Department of Electrical and Electronic Engineering, Olusegun Agagu University of Science and Technology, Nigeria

⁴Department of Computer Science, Harvarde College of Science Business and Management Studies, Nigeria

⁵Department of Electrical Electronic Engineering, Imo State Polytechnic Umuagwo, Nigeria

⁶Department of Computer Science, Harvarde College of Science Business and Management Studies, Nigeria

ABSTRACT: This project is intended to provide a solution to a company's problem of developing a smart motion detection device for home and office security applications. The company intends to deploy 100s of these motion detectors throughout the Greater London Area. As a result, each motion detector will require a low power wide area network (LPWAN) wireless interface. Unlike traditional motion detectors, which detect only motion, the proposed motion detector will be able to detect motion direction (e.g., whether a person is leaving or entering a room) by analyzing signals from multiple motion sensors. The basic idea is that if three motion sensors are placed along the corridor: Sensor 1 - Sensor 2 - Sensor 3, the sequence of sensor activations will provide information about the direction. The first section 1.0 of this report covers the project's general introduction. Section 2.0 provides description of Low Power Wide Area Network technologies (LoRA, SigFox, NB-IoT), including the operating principle, characteristics, hardware, and recommendation for the most appropriate technology. Section 3 describes how to write code for the Raspberry PI to analyse signals from a 4x4 PIR array to detect motion direction (left, right, up, down) and display the results on an LCD screen. The final section discusses the conclusion future work and recommendations.

Keywords: LPWAN, Raspberry PI, motion direction, smart motion detection device, PIR array

I. INTRODUCTION

The prevalence and pervasiveness of insecurity in residential and commercial buildings has skyrocketed in recent time. The advancement of technology has exponentially also increased the techniques used by thieves and other perpetrators to commit their crimes. One major method of combating the problem of insecurity is motion detection [9] whose function is dependent on the type of sensor used and the programming language used for coding of the system. A motion sensor is a security device installed in buildings to detect unauthorized movement in restricted areas during specific times. The device is used in commercial and residential buildings, as well as industrial and military facilities. A motion sensor can be either active or passive. When there is an interruption of radio waves by movement, the detector in active sensors sets off an alarm, whereas in passive sensors, changes in ambient temperature result in alarm sounds from the scanner that detects it. Passive sensors include passive infrared (PIR) [14], microwave (MW) [15], dual technology sensors, ultrasonic sensors, camera-based sensors [16] and vibration sensors, with PIR being the most commonly used sensor for residential monitoring.

There are numerous applications that can be built in recent times, such as smart homes for energy reduction, smart healthcare [17] [18] structural health monitoring [19], mechanical stress and structural issue detection, wildlife tracking for animal migratory patterns [20], intelligent transportation in which smart sensors are embedded into roads and road infrastructure [21], and so on. These applications have several advantages ranging from reduction in cost, consumption of minimal energy, transmission over long distances and data rates that are low. Each of these wireless applications have one or two standards used for the sensors. These sensors can sense some physical changes in the characteristics of environment in which it is operating and produce signals needed to be sent somewhere else which can be achieved through different communication protocols/applications. There are two very well-known standard Wi-Fi and Bluetooth which were not designed for IoT application, they were designed many years ago for different applications. Though Wi-Fi is available in every smartphone, it does not communicate directly and requires a central access point and Bluetooth does not support every IoT need. However, for IoT applications, different standards of communications over long and short ranges have been designed. There have been many various IoT protocols and solutions developed for short-range communication networks. Some of

them, such as ZigBee/IEEE 802.15.4 offered low message throughput and low to no quality of service. As there are many requirements for the applications of Internet of things, a new technology which connects low bandwidth devices powered by batteries to communicate over a range of long distances at low bit data rates was initiated.

LPWAN technology is used for uploading data into cloud where data are gathered from different sensors given gateway device which is then uploaded to the cloud from the device. The application of LPWAN is as wide as the distance it covers and has unique feature of constrained latency, low data rates, reduced connectivity cost, high density of object and reduced cost of hardware. LPWAN also offer many advantages ranging from low bandwidth consumption, very low radio chipsets cost, low radio subscription costs, reduction in power consumption with additional longer base station ranges than cellular network. Nevertheless, capabilities of device hardware, requirements for consuming power at low power, as well as total cost have been major constraints of some IoT technology/applications today. LPWAN ensures battery life and signal coverage while supporting large connections at a low cost [22].

Unlike traditional motion detectors, which detect only motion, the proposed motion detector will be able to detect motion direction (e.g, whether a person is leaving or entering a room) by analyzing signals from multiple motion sensors. The basic idea is that if three motion sensors are placed along the corridor: Sensor 1 - Sensor 2 - Sensor 3, the sequence of sensor activations will provide information about the direction. Compared to prior work [23], the proposed system uses a larger sensor grid for higher accuracy and reliability.

II LORA, SIGFOX and NB-IoT LPWAN TECHNOLOGIES

This section focuses on the comparative study of the aforementioned LPWAN technologies.

A. LoRa

LoRa (Long Range) wireless technology is responsible for long-distance transmission of small data packets ranging from 0.3 to 5.5kbps [6] using a low-energy mechanism to a receiver via a gateway capable of treating hundreds of devices at the same time. A microprocessor for data processing and a radio module with an antenna are the two components of a LoRa gateway. The first version of communication based on LoRa protocol standard is LoRaWAN, a proprietary technology that has been standardized in collaboration with LoRa-Alliance. LoRa operates within bandwidths of 250kHz and 125kHz in the ISM unlicensed bands of 433MHz, 915MHz, and 868MHz in Asia, North America, and Europe respectively [2]. LoRa end devices are classified into three types based on what the downlink transmission demand and energy efficiency are [10]. Class A is bidirectional in nature and used for only downlink communication applications after an uplink transmission has been sent. Class B has scheduled reception slots with beacon time synchronization from the gateway prompting the server about the end device's listening time. Data reception for class C is possible at any time, resulting in this device having the most downlink slots. LoRa can be used in object tracking [24] surveillance system [25], pollution monitoring [26] among others [24]. The LoRa Technology Network Architecture is depicted in figure 1 below.

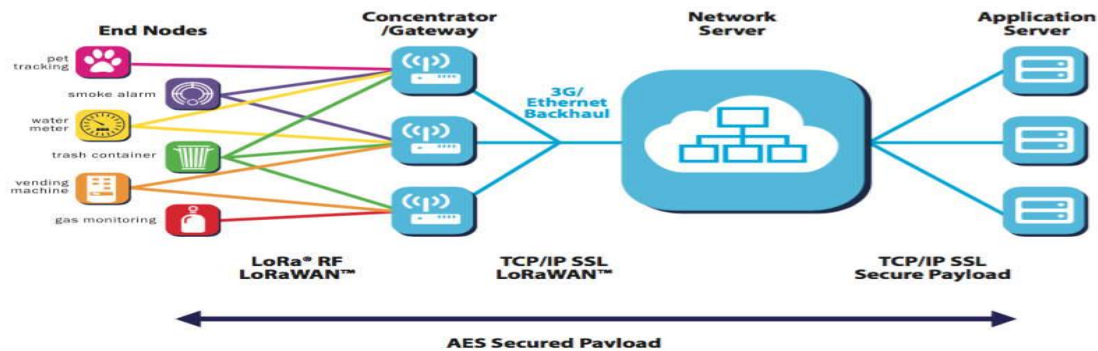


Figure 1: The Network Architecture of LoRa Technology [3]

B. Sigfox

Sigfox is a proprietary technology [11] that has been standardized in collaboration with ETSI in a 10-40km range. The technology makes use of unlicensed bands of 868MHz, 915MHz, and 433MHz in Europe, North America, and Asia respectively. Sigfox provides connectivity to internet of things say from 30 to 50km in rural area with potential secondary connectivity solution of lower battery consumption and better user experience [12]. In urban area its connectivity ranges from 3 to 10km [13] and has characteristics feature of low energy consumption, standardized price and management of data complexity on cloud. With Sigfox, messages are exchange over publicly with 192KHz [6] of the publicly available band and uses a technology Ultra-Narrow band, data rate of 100 or 600 [8] making Sigfox base stations long distances communication possible without being impacted by the noise. Examples of applications of Sigfox include detecting smoke, smart irrigation among others. The Network Architecture of Sigfox Technology is depicted in figure 2 below.

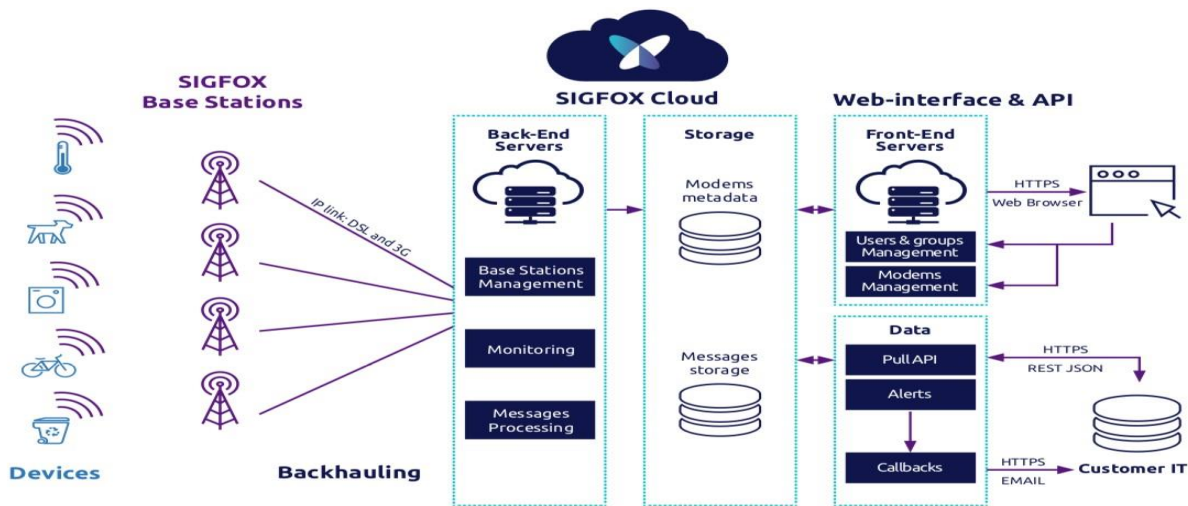


Figure 2: The Network Architecture of Sigfox Technology [5]

C. Narrowband-IOT

This technology was pioneered by Third-Generation Partnership Project [1], is a licensed LPWAN, operates at 200KHz [7] and distinguished by its ability to provide 8-10 years of battery life, low cost, extensive coverage, and high network security. Three modes of operation of NB-IoT are standalone, in-band and guard [7]. This means that stand-alone used one or more existing GSM carriers, while the guard band operation used LTE spectrum resource blocks unused in band operation blocks resource with LTE carrier. Long-distance communication with NB-IoT is possible at low data rates of about 250kps and 20kbps for multitone downlink and single tone uplink communication, respectively. Furthermore, unlike Sigfox and LoRa unlicensed LPWAN technologies, the NB-IoT device offers a reduction in complexity. At the most basic level, NB-IoT end-nodes collect data, with power supplied by rechargeable batteries in addition to having sensors enabled for collecting data from the surrounding environment. NB-IoT applications include smart cities, smart metering, and so on. Because NB-IoT has no duty cycle limitations, it is suitable for providing more frequent communications in IoT applications [27]. The Network Architecture of NB-IoT Technology is depicted in figure 3 below.

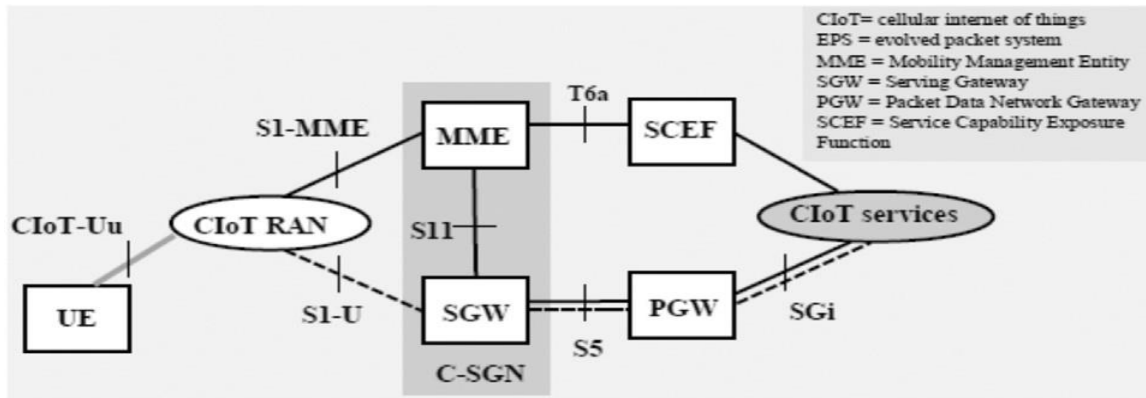


Figure 3: The Network Architecture of NB-IoT Technology [4]

III. IMPLEMENTATION AND TESTING

Implementation

This section focusses on the code and the principle used to detect motion. The programming language for this code is python program. There are different ways the sensors can be programmed but some sets of coding must be completed. The main principle behind code development for Raspberry pi to analyze the signals from a 4x4 PIR array to detect motion direction (left, right, up, down) and display the output on an LCD screen using python programming language approach is Python's ability to support object-oriented languages. The code works by analyzing the timestamp to determine the motion direction. The timestamp is the time when the sensor was activated, and the motion direction is detected by looking at the sequence of the time stamp activated. When the sensor is turned on, the last timestamp in this array will be equal to np.zeros (4,4), and when the sensor is turned off, the last timestamp in the second array will equal to np.zeros (4,4).The implementation is based on open-source tkgpio library that simulates electronic devices connected to the GPIO on Raspberry Pi [28].

Code

```
#!/usr/bin/python3
from json import load
import numpy as np
import time
from tkgpio import TkCircuit
from libas import MyCircuit
from libas import configuration
circuit = MyCircuit(configuration)
@circuit.run
def main():
    # now just write the code you would use in a real Raspberry Pi
    from Adafruit_CharLCD import Adafruit_CharLCD
    from gpiozero import Buzzer, LED, PWMLED, Button, DistanceSensor, LightSensor, MotionSensor
```

```
from lirc import init, nextcode
from py_irsend.irsend import send_once
from time import sleep
lcd = Adafruit_CharLCD(2, 3, 4, 5, 6, 7, 16, 2)
#define motion array
# motion_detected(1,1)
def motion_detected(i, j):
    detected = MotionSensor(8+i*4+j, queue_len = 1, sample_rate=100, threshold=0.001, partial=False)
    return (detected.motion_detected)
motion_last_on = np.zeros((4,4))
motion_last_off = np.zeros((4,4))
def register_on(idx, idy):
    tm = round(time.time())
    motion_last_on[idx, idy] = tm
    print("registering on " + str(idx) + " " + str(idy) + " " + str(tm))
def register_off(idx, idy):
    tm = round(time.time())
    motion_last_off[idx, idy] = tm
    #print("registering off " + str(idx) + " " + str(idy) + " " + str(tm))
# methods to detect movement in each direction
def detect_right():
    # analyse arrays last_on and last_off to detect movement to the right
    if (motion_last_on[0,3] > motion_last_on[0,2] and \
        motion_last_on[0,2] > motion_last_on[0,1] and \
        motion_last_on[0,1] > motion_last_on[0,0]):
        print("motion right detected")
    # clear matrix
    motion_last_on.fill(0)
    pass
```

```
def detect_left():
```

```
    # analyse arrays last_on and last_off to detect movement to the left
```

```
    if (motion_last_on[0,3] < motion_last_on[0,2] and \
```

```
        motion_last_on[0,2] < motion_last_on[0,1] and \
```

```
        motion_last_on[0,1] < motion_last_on[0,0]):
```

```
        message1 = "motion left detected"
```

```
        print(message1)
```

```
            # clear matrix
```

```
            motion_last_on.fill(0)
```

```
    pass
```

```
def detect_down():
```

```
    # analyse arrays last_on and last_off to detect movement in the down direction
```

```
    if (motion_last_on[0,3] > motion_last_on[1,3] and \
```

```
        motion_last_on[1,3] > motion_last_on[2,3] and \
```

```
        motion_last_on[2,3] > motion_last_on[3,3]):
```

```
        print("motion down detected")
```

```
            # clear matrix
```

```
            motion_last_on.fill(0)
```

```
    pass
```

```
def detect_up():
```

```
    # analyse arrays last_on and last_off to detect movement in the up direction
```

```
    if (motion_last_on[0,3] < motion_last_on[1,3] and \
```

```
        motion_last_on[1,3] < motion_last_on[2,3] and \
```

```
        motion_last_on[2,3] < motion_last_on[3,3]):
```

```
        print("motion up detected")
```

```
            # clear matrix
```

```
            motion_last_on.fill(0)
```

```
    pass
```



```
# main loop
while True:
    for i in range(0,4):
        for j in range(0,4):
            if motion_detected(i, j):
                register_on(i,j)
                lcd.clear()
                lcd.message("Motion Detected")
            else:
                register_off(i,j)
                detect_left()
                detect_right()
                detect_up()
                detect_down()
        # print the array content
        for i in range(0,4):
            for j in range(0,4):
                print(motion_last_on[i,j], ' ', end='')
            print()
        print('-----')
        sleep(1)
```

Function definition

1.#!/usr/bin/python3 - This is a special line, and any attempt to remove it will cause the code to fail. It appears to be a command because in Python, commands are used to describe what the programme is doing. This is necessary because many times when you run a code, you may forget what the code is doing and it may take some time to remember. As a result, it's a good idea to provide some commands after you've finished coding. It also indicates that the code is written in Python.

2. from json import load- This command accepts a file object and returns a json object.

3. import numpy as np- It makes the code easier to read.

4. from libas import MyCircuit - Python is programming language with many library. This library is for my circuit operation. The code in this assignment make use of arrays for operations. These arrays are defined and provided by the library.

5. def main()- Def main function ensures that any value inserted in the function will be automatically executed when you run the program.

6. `from gpiozero import` -gpiozero is a special library which importing code for Buzzer, LED, PWMLed, Button, Distance Sensor

7. `lcd = Adafruit_CharLCD(2, 3, 4, 5, 6, 7, 16, 2)` - This is the LCD screen library.

Although parts of the code can be eliminated, the number is not changed in this section of the code.

The reason for this is to output some text on the LCD, hence LCD objects are created.

The numbers indicate which pin on the Raspberry Pi the LCD is attached to.

8. `# motion detected` - In this function two parameters are improvised that is *i* and *j* knowing fully well that sensors are arranged in a matrix form. The way the code work is that you can determine the motion direction by analyzing the timestamp. Timestamp is the time when the sensor was activated, by looking at the sequence of the time stamp you will be able to detect activated motion direction. In this array we are storing the last timestamp when the `motion_last_on = np.zeros (4,4)` sensor was active and second array `motion_last_off = np.zeros (4,4)` when the sensor was turned off.

9. `def register_on` -In a situation whereby you need to modify the array depending on the motion, this function read the current system time which is time time and it will modify the matrix and put the time time inside the array.

10. `#print("registering off " + str(idx) + " " + str(idy) + " " + str(tm))` -This statement help to know whether the code is working or not.

11. `def detect_right()`- It is a function that analyses the sequence of motion sensor activations to detect the motion direction. The current prototype only detects a motion along the top row.

12. `def detect_left()`-It helps to detect motion along the top row only from left without consideration for complex motion .It compares timestamp between the sensor along the direction of consideration.

13. `def detect_down()`-It helps to detect motion along the down column only without consideration for complex motion. It compares timestamp between the sensor along the direction of consideration

14. `def detect_up()`-It helps to detect motion along the up column only without consideration for complex motion. It compares timestamp between the sensor along the direction of consideration

15. `# main loop`-This means that your code will be constantly moving in this region while true means it will be moving forever until you stop the program manually. In range (0,4)the arrays are populated as motion is detected it being registered on (i,j)

16. `detect_left()`

`detect_right()`

`detect_up()`

`detect_down()`

These functions analyze the content of this matrix and detect the motion

17. `# print the array content`- Print the content of the array

18. `for i in range(0,4):`

`for j in range(0,4):`

`print(motion_last_on[i,j], ' ', end="")`

`print()`


```
print('-----')
```

These code help with the debugging process. When the code is run and terminal window is opened then the print out of the content of the array will be seen. It shows current time stamp and whether the code is working or not.

19. *sleep(1)*-Sleeps for 1 seconds

Algorithm

1. Play virtual machine from VM Ware player installed and click terminal to display pi@raspberry dialogue box.
2. At the terminal, type cd Documents and press enter.
3. Type ls in front of the document directory and press enter.
4. Enter cd CIS116-6 as the command.
5. Enter ls in front of the CIS116-6 directory.
6. Run the *python3 motionup.py* and *python3 motiondown.py* scripts. Enter *python3 motionleft.py* and *python3 motionright.py* on the terminal of the CIS116-6 directory.
7. Move the cursor arrow from PIR sensors down to up of, up to down, right to left, and left to right, resulting in terminal
8. Examine and contrast the generated code

Testing

Motion down

The motion down code was implemented and tested by moving the computer's mouse vertically from down along the PIR 14, PIR 24, PIR 34, and PIR 44 axis, which displayed motion up on the LCD, followed by a print to demonstrate the validity of the implemented code. From left to right, the value ranges from 1641410222.0 to 1641410223.0. As evidenced by the result, the motion is downward to upward.

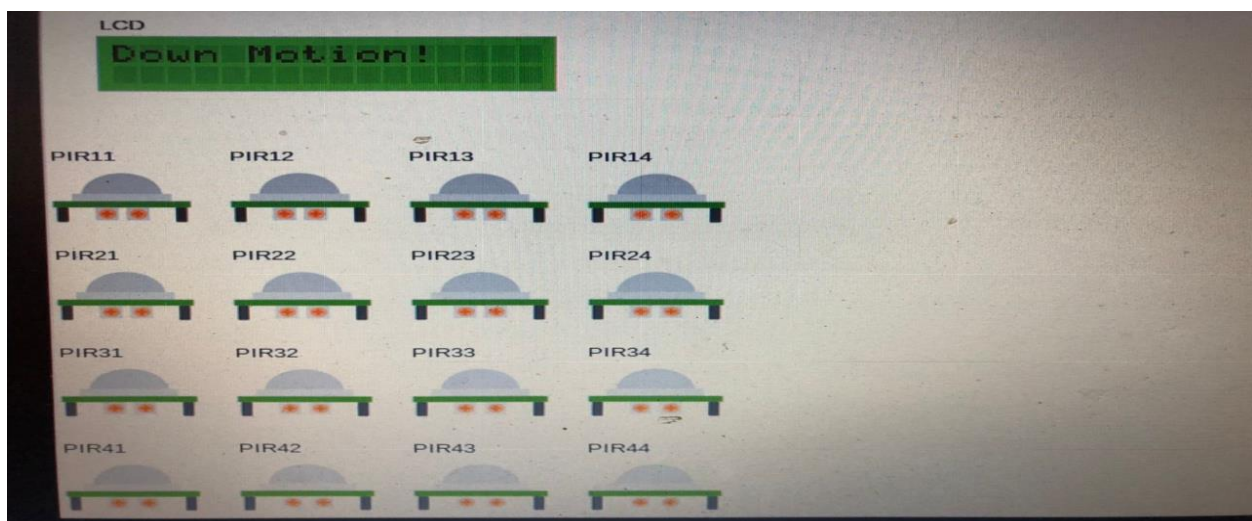
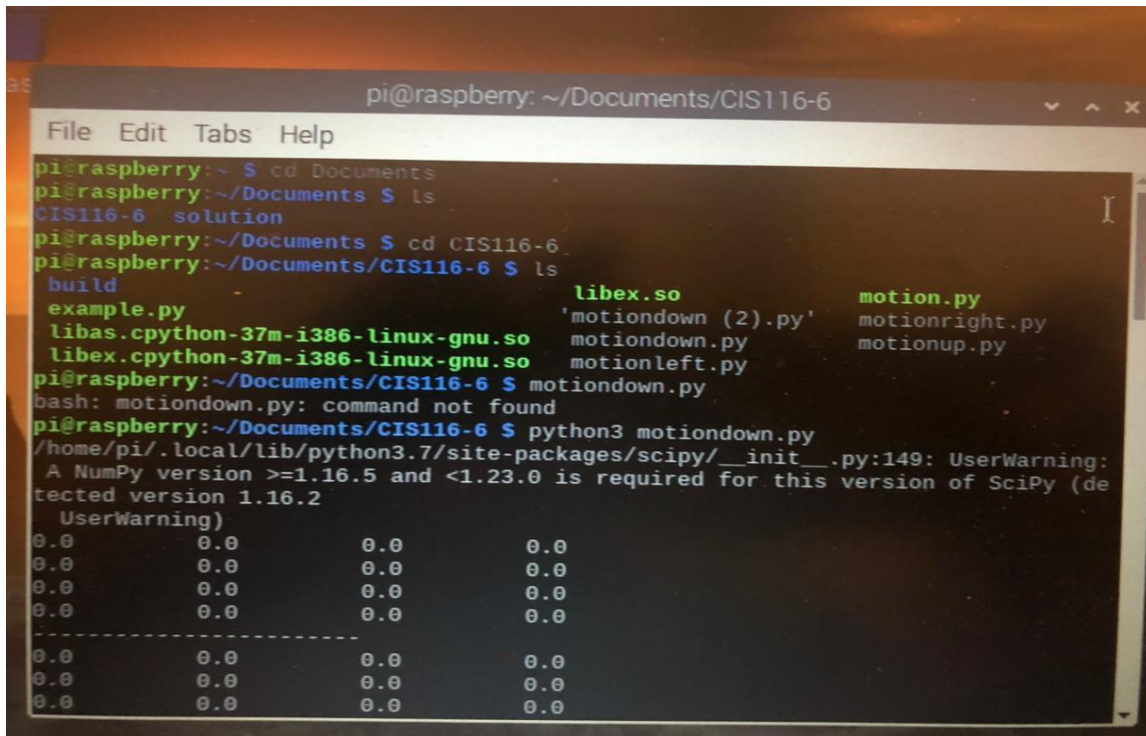
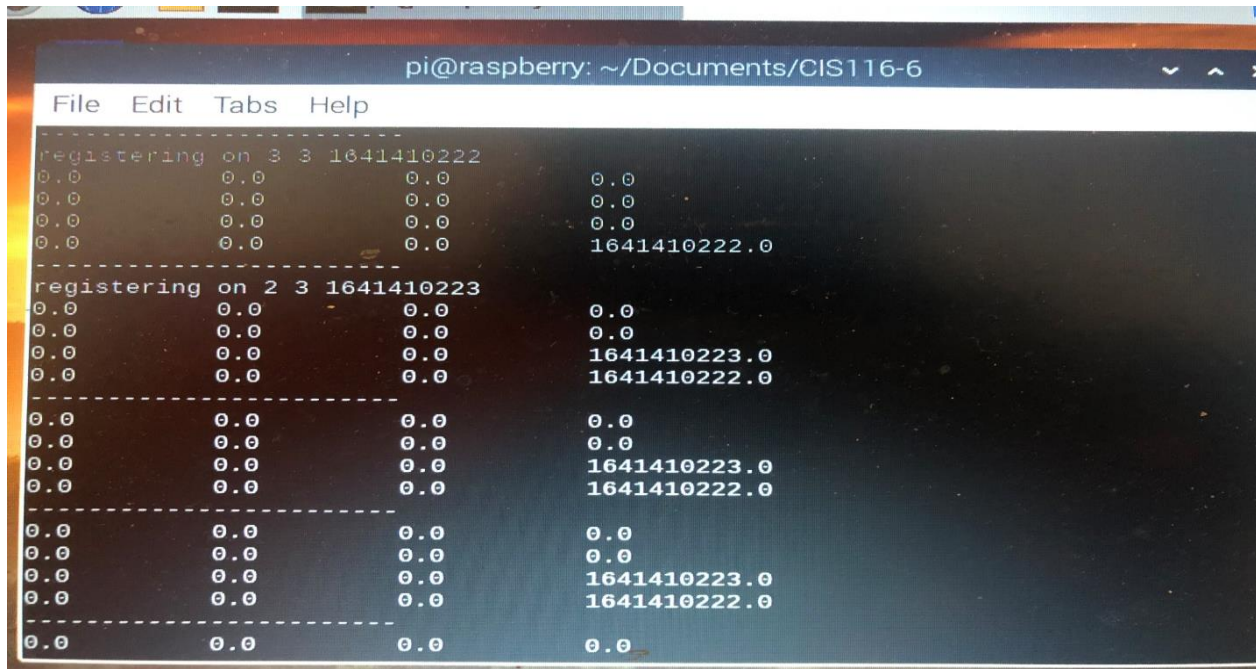


Figure 4: Display of the LCD screen during and after down movement.



```
pi@raspberrypi: ~/Documents/CIS116-6
File Edit Tabs Help
pi@raspberrypi:~$ cd Documents
pi@raspberrypi:~/Documents$ ls
CIS116-6 solution
pi@raspberrypi:~/Documents$ cd CIS116-6
pi@raspberrypi:~/Documents/CIS116-6$ ls
build          libex.so          motion.py
example.py     'motiondown (2).py' motionright.py
libas.cpython-37m-i386-linux-gnu.so motiondown.py     motionup.py
libex.cpython-37m-i386-linux-gnu.so motionleft.py
pi@raspberrypi:~/Documents/CIS116-6$ motiondown.py
bash: motiondown.py: command not found
pi@raspberrypi:~/Documents/CIS116-6$ python3 motiondown.py
/home/pi/.local/lib/python3.7/site-packages/scipy/__init__.py:149: UserWarning:
A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (de
tected version 1.16.2
UserWarning)
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
-----
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
```

Figure 5: Implementation of motion down code on a virtual machine.



```
pi@raspberrypi: ~/Documents/CIS116-6
File Edit Tabs Help
-----
registering on 3 3 1641410222
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
-----
0.0      0.0      0.0      1641410222.0
-----
registering on 2 3 1641410223
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
0.0      0.0      0.0      1641410223.0
0.0      0.0      0.0      1641410222.0
-----
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
0.0      0.0      0.0      1641410223.0
0.0      0.0      0.0      1641410222.0
-----
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
0.0      0.0      0.0      1641410223.0
0.0      0.0      0.0      1641410222.0
-----
0.0      0.0      0.0      0.0
```

Figure 6: Motion down code run result showing increment in figure from down to up.

Motion left

The motion left code was tested and implemented by moving the computer's mouse horizontally from the left -hand side along the PIR 11, PIR 12, PIR 13, and PIR 14 axis, which displayed motion left on the LCD followed by a print to

demonstrate the validity of the implemented code. The column result is 4x4, with the values 0.0, 1641428105.0, 1641428106.0, and 1641428108.0 increasing from left to right, with the value 1641428108.0 indicating that the direction is from left to right.

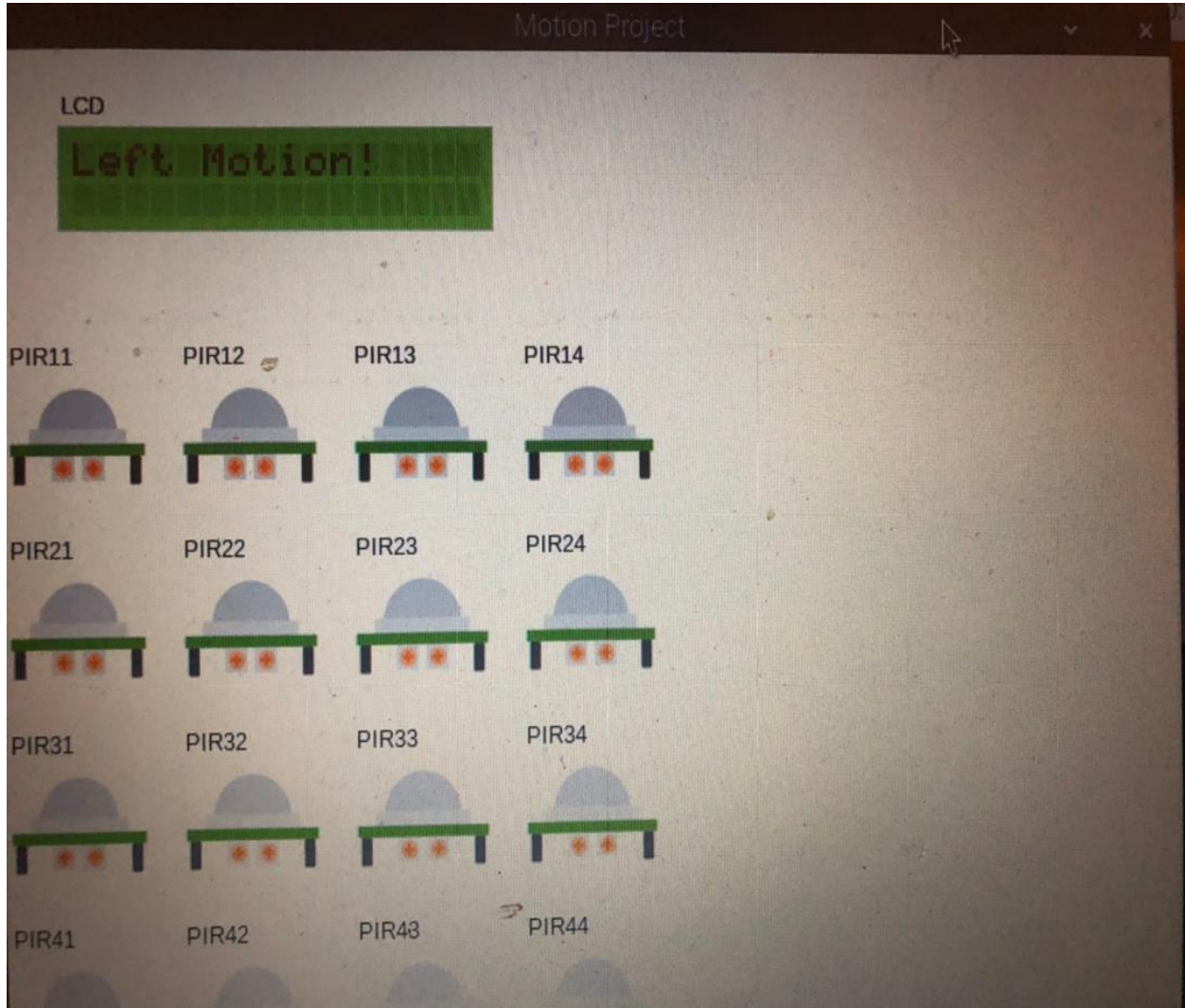
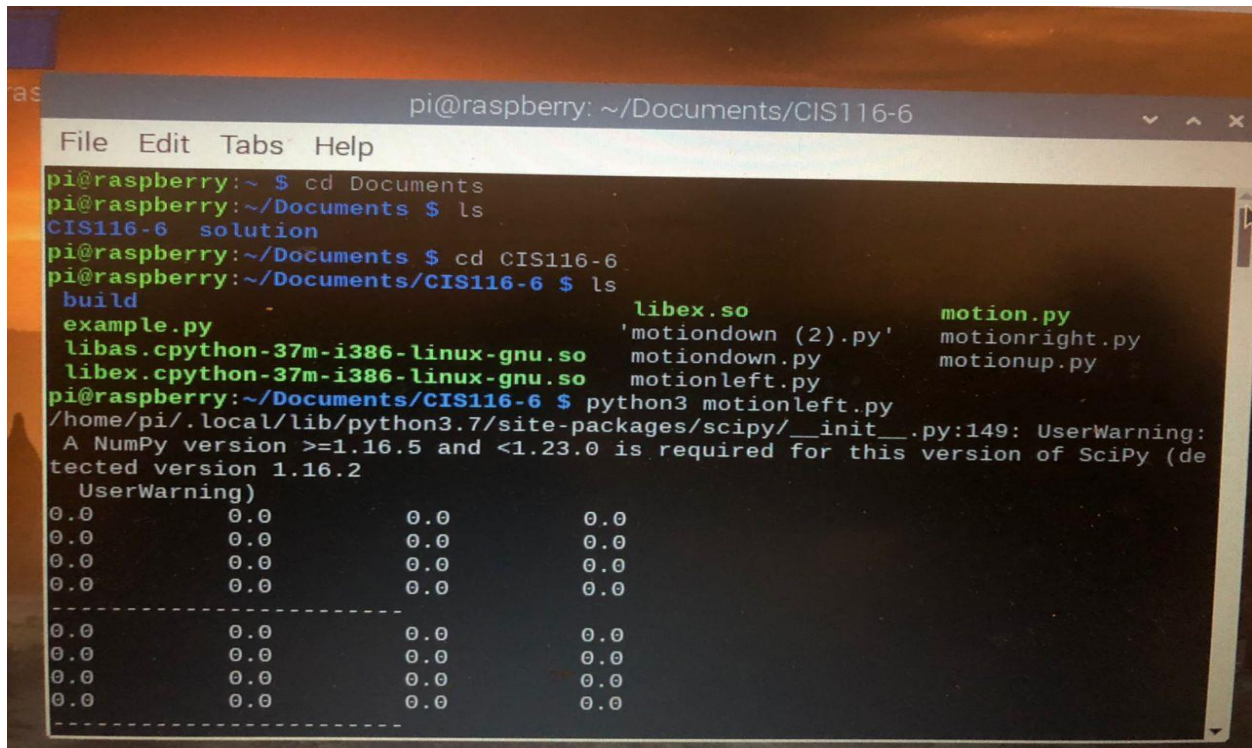
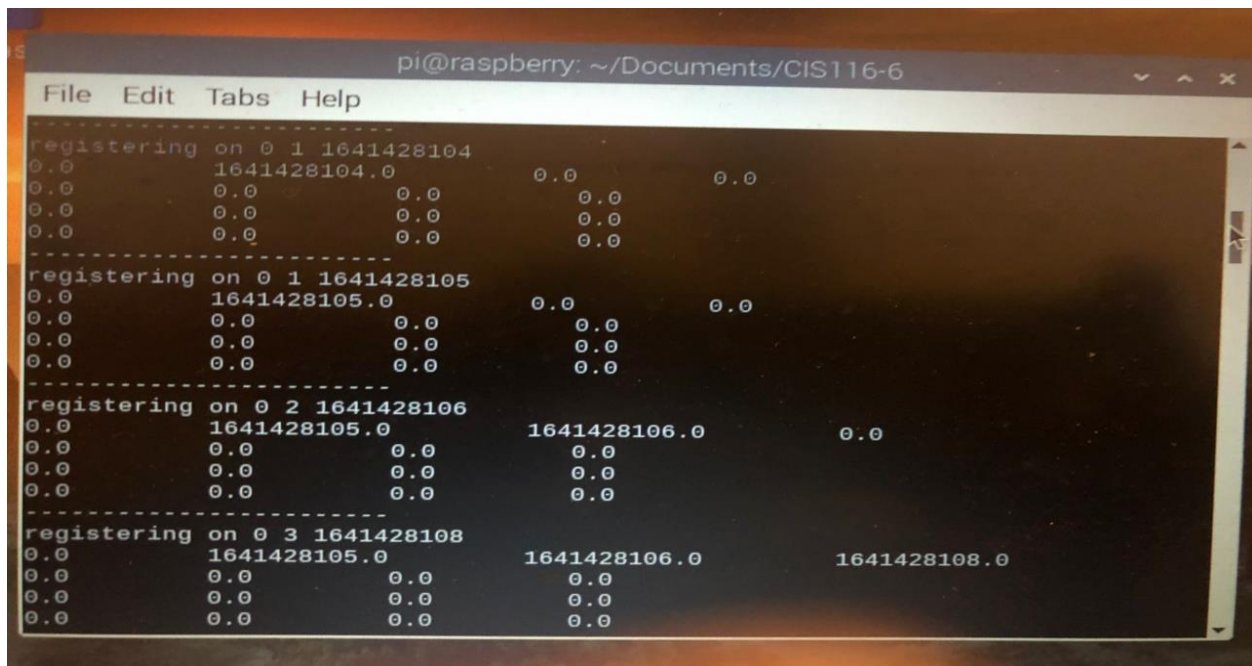


Figure 7: Display of the LCD screen during and after left movement.



```
pi@raspberrypi:~/Documents/CIS116-6
File Edit Tabs Help
pi@raspberrypi:~$ cd Documents
pi@raspberrypi:~/Documents$ ls
CIS116-6  solution
pi@raspberrypi:~/Documents$ cd CIS116-6
pi@raspberrypi:~/Documents/CIS116-6$ ls
build          libex.so          motion.py
example.py     'motiondown (2).py' motionright.py
libas.cpython-37m-i386-linux-gnu.so motiondown.py     motionup.py
libex.cpython-37m-i386-linux-gnu.so motionleft.py
pi@raspberrypi:~/Documents/CIS116-6$ python3 motionleft.py
/home/pi/.local/lib/python3.7/site-packages/scipy/__init__.py:149: UserWarning:
A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (de
tected version 1.16.2
UserWarning)
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
-----
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
-----
```

Figure 8: Implementation of motion left code on a virtual machine.



```
pi@raspberrypi:~/Documents/CIS116-6
File Edit Tabs Help
-----
registering on 0 1 1641428104
0.0      1641428104.0      0.0      0.0
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
-----
registering on 0 1 1641428105
0.0      1641428105.0      0.0      0.0
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
-----
registering on 0 2 1641428106
0.0      1641428105.0      1641428106.0      0.0
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
-----
registering on 0 3 1641428108
0.0      1641428105.0      1641428106.0      1641428108.0
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
0.0      0.0      0.0      0.0
```

Figure 9: Motion left run result showing increment in figure from left to right.

Motion right

When implementing and testing motion right code, the computer's mouse was dragged horizontally from right to left along the PIR 14, PIR 13, PIR 12, and PIR 11 axis, which displayed motion right on the LCD followed by a print to demonstrate the validity of the implemented code. The value obtained increases from right to left, ranging from 16441434375. 0 to 16441434377. 0, indicating that the direction is from the right.

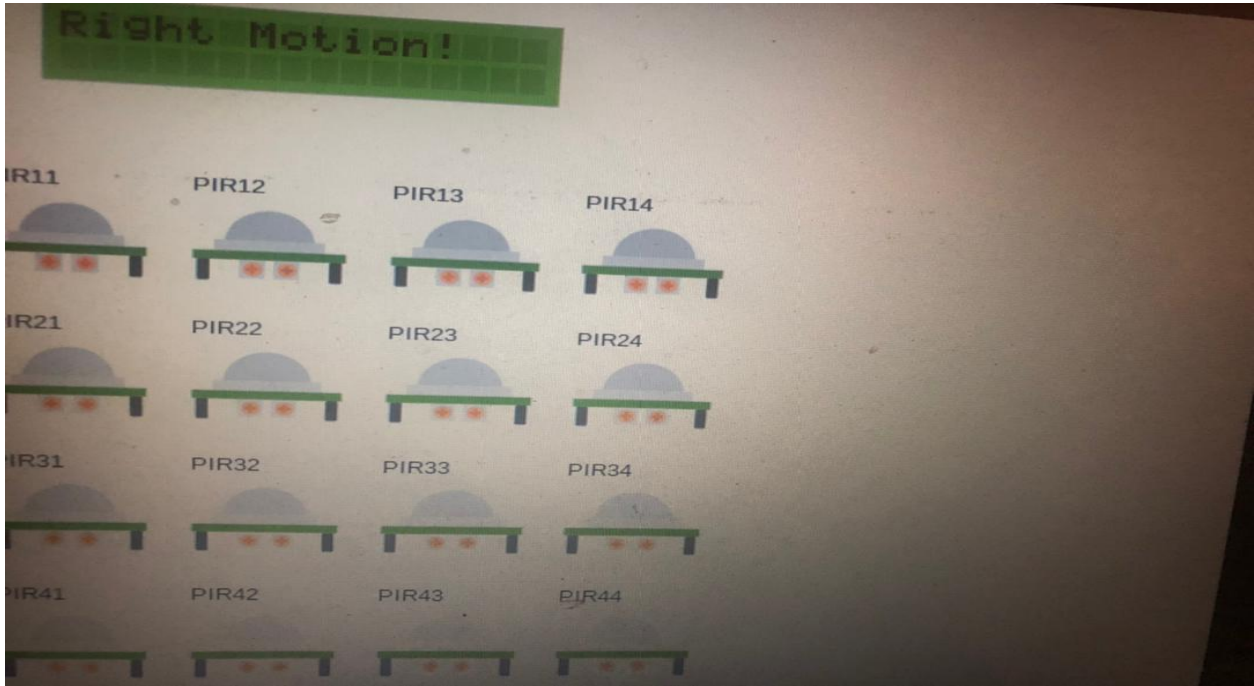


Figure 10: Display of the LCD screen during and after right movement.

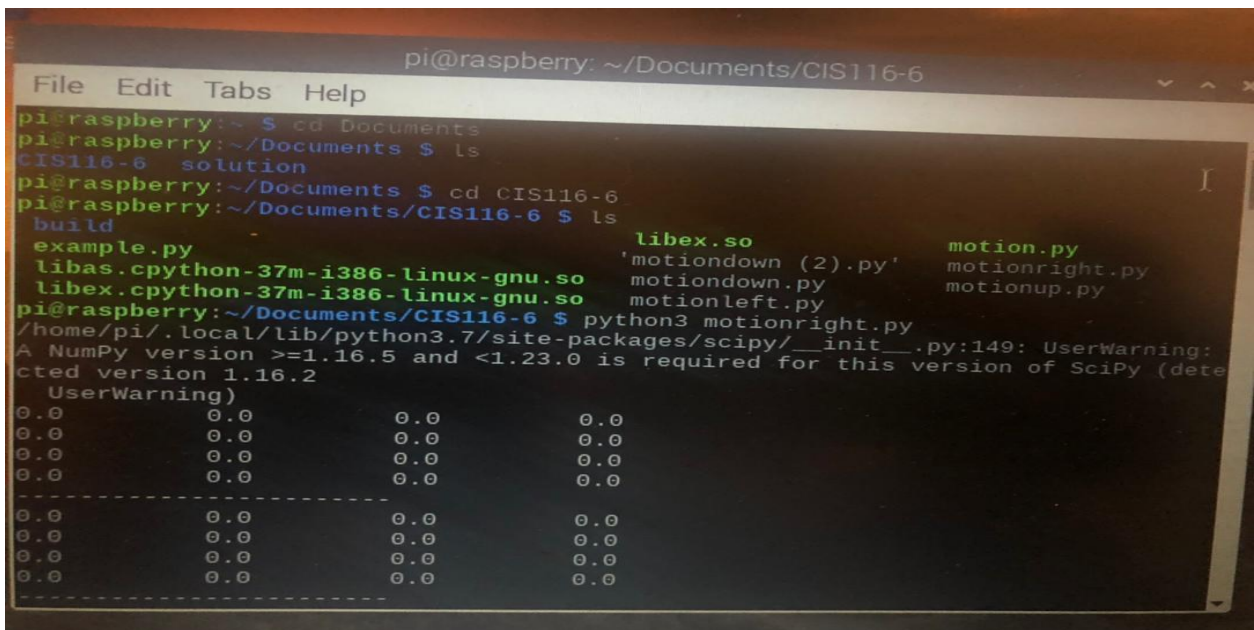


Figure 11: Implementation of motion right code on a virtual machine.

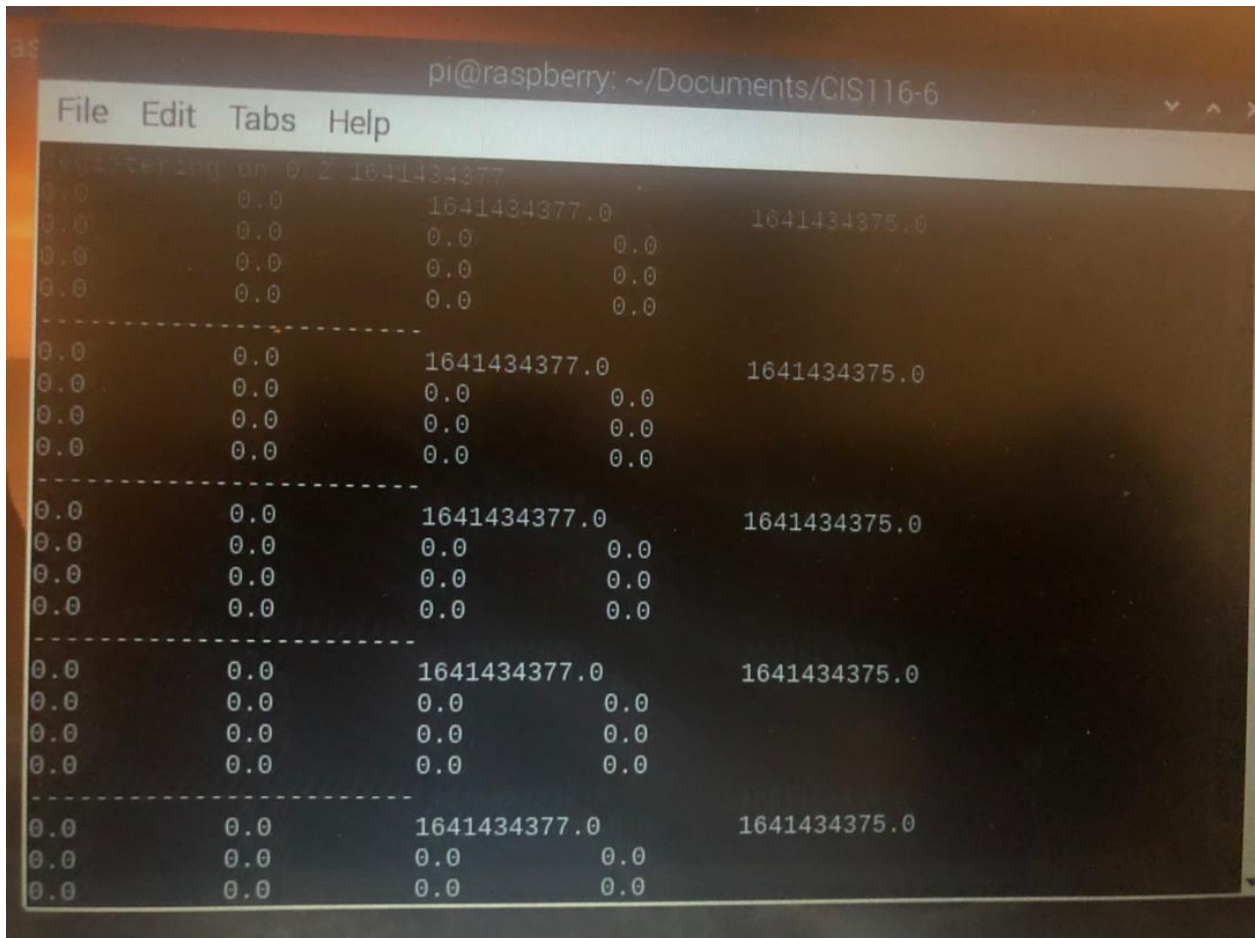


Figure 12: Motion right run result showing increment in figure from right to left.

Motion up

The motion up code was implemented and tested by moving the computer's mouse vertically from top to bottom along the PIR 11, PIR 21, PIR 31, and PIR 41 axes, resulting in motion up on the LCD and a print to demonstrate the implemented code's validity. The value ranges from 1641426922.0 to 1641426926.0, with 1641426922.0 being the lowest and 1641426926.0 being the highest, indicating an upward direction.

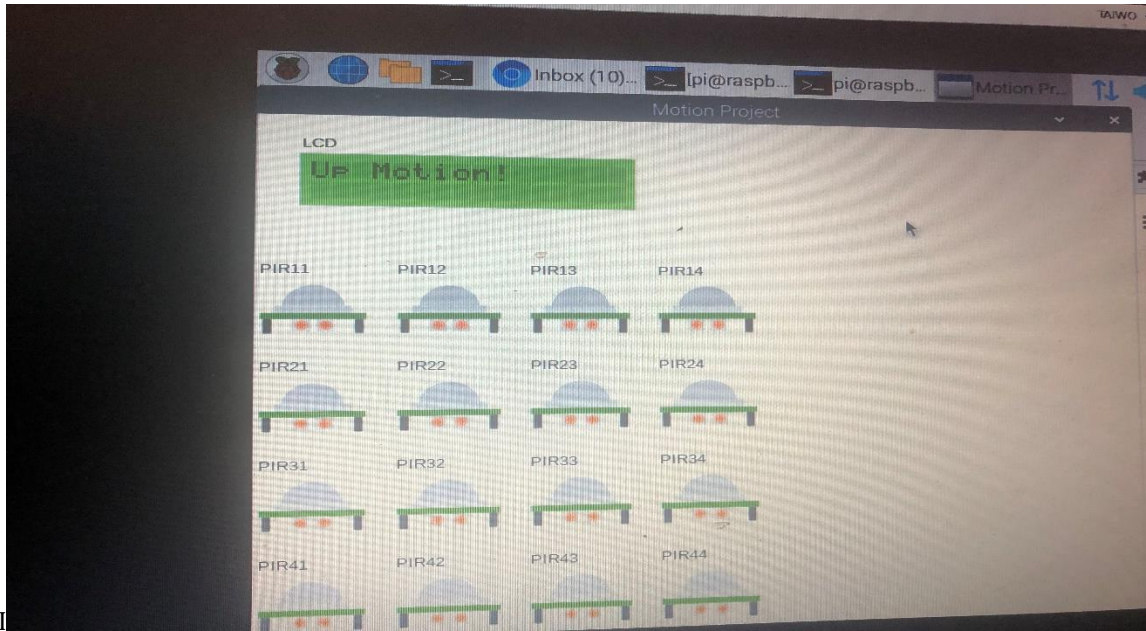


Figure 13: Display of the LCD screen during and after up movement.

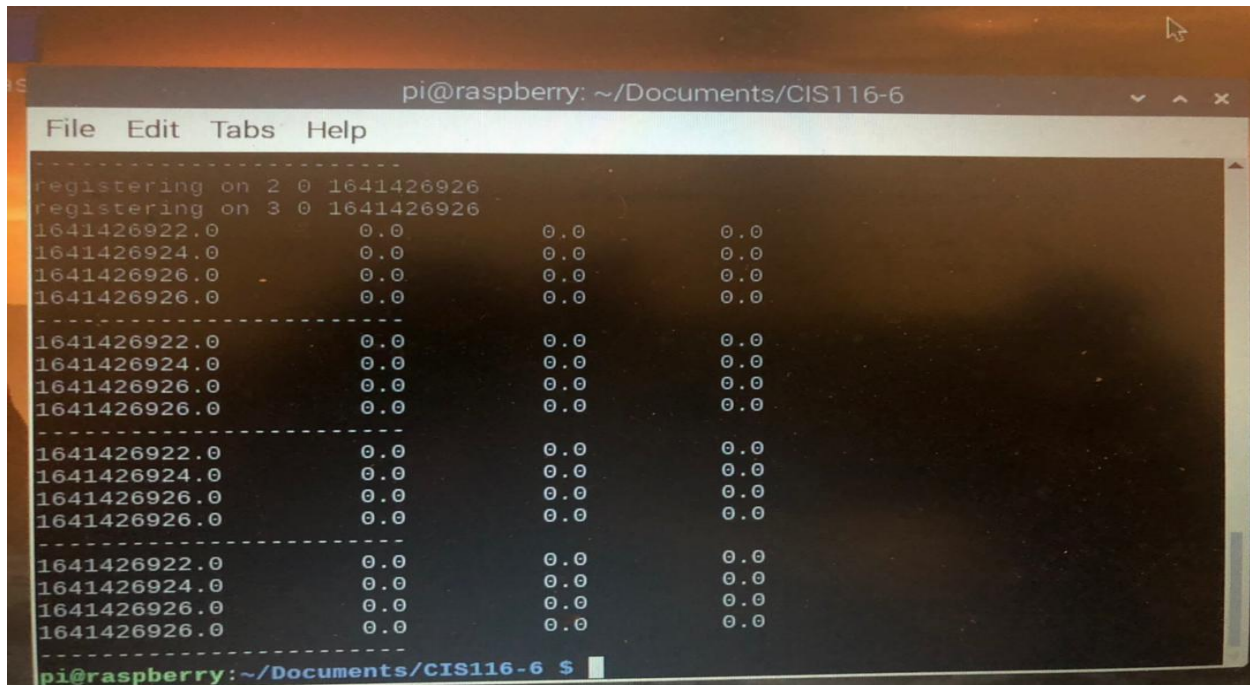
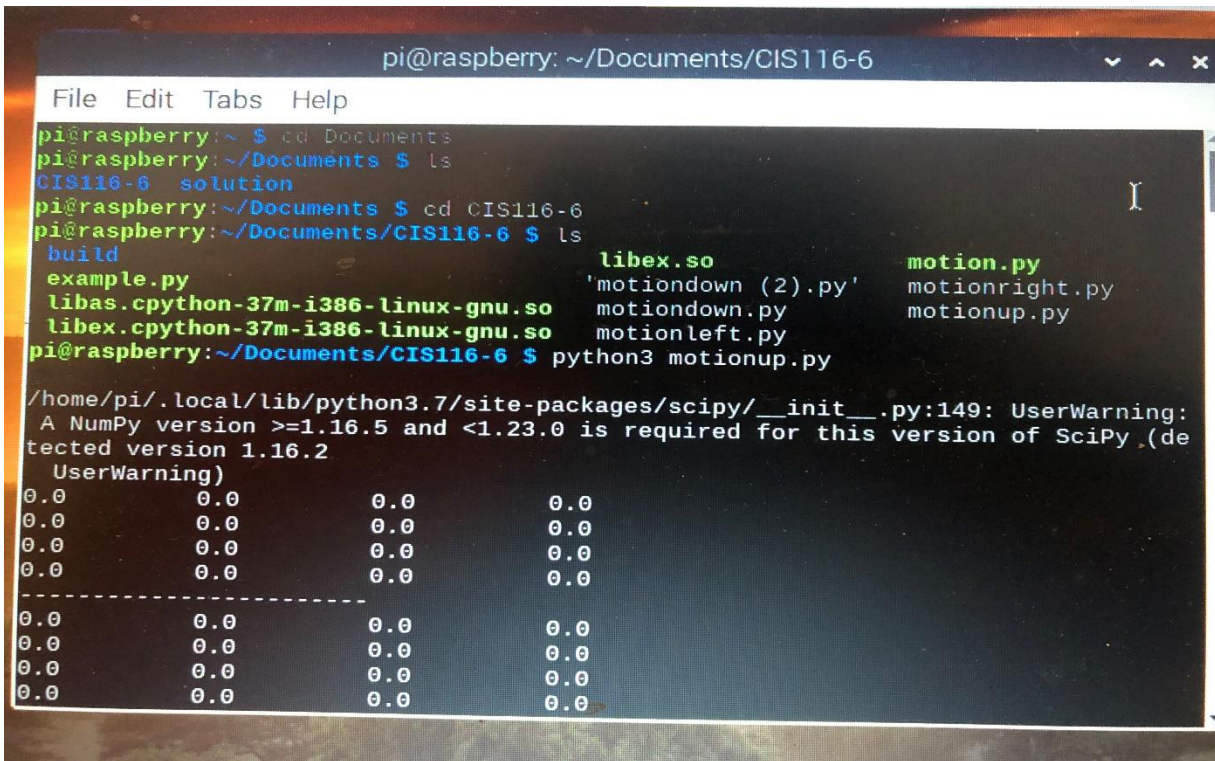


Figure 14: Motion up run result showing increment in figure from up to down.



```
pi@raspberrypi: ~/Documents/CIS116-6
File Edit Tabs Help
pi@raspberrypi: ~ $ cd Documents
pi@raspberrypi: ~/Documents $ ls
CIS116-6 solution
pi@raspberrypi: ~/Documents $ cd CIS116-6
pi@raspberrypi: ~/Documents/CIS116-6 $ ls
build                               libex.so                            motion.py
example.py                          'motiondown (2).py'                motionright.py
libas.cpython-37m-i386-linux-gnu.so  motiondown.py                       motionup.py
libex.cpython-37m-i386-linux-gnu.so  motionleft.py
pi@raspberrypi: ~/Documents/CIS116-6 $ python3 motionup.py

/home/pi/.local/lib/python3.7/site-packages/scipy/__init__.py:149: UserWarning:
A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy. (de
tected version 1.16.2
UserWarning)
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
-----
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
```

Figure 15: Implementation of motion up code on a virtual machine.

Limitations

- The limitation of this code is that the codes can be used to detect motion in all directions but it will show Motion Detected on the LCD. However, if the codes for each motion direction are run individually, the code is able to display direction of motion being tested.
- The code is only compatible with the Python compiler; otherwise, the code will not run.
- Ability to detect diagonal and other complex movements

IV. CONCLUSIONS

This research revealed that each LPWAN technology performs differently based on the principle of operation, characteristics, and hardware. To achieve international coverage, high transmission quality, as well as data throughput and coverage, are required. NB-IoT technology is recommended for the company that plans to deploy 100 motion detectors throughout the Greater London Area. Also, code developed using python programming language from a 4x4 PIR array to detect motion direction (left, right, up, down) and display the output on an LCD screen worked with noisy input.

The author suggests that future research include the following components:

- Implementation of a remote status update function, in which the system sends motion events to the cloud via a wireless interface using the MQTT protocol.
- Options for snapping motion detected on automatic basis should be added.
- The implementation of the hardware using raspberry which make the work more interesting and easier

Recommendation

The project is to deploy 100 motion detectors over Greater London Area in homes and offices. Factors to consider when choices are to be made when choosing an appropriate LPWAN technology are as highlighted below:

- i) QoS: Motion detectors are usually used for applications that require dedicated quality of service.
- ii) Latency: The motion detectors will need to operate on a low latency and consume less energy.
- iii) Coverage: The coverage is basically indoor application which is less than 10 km range.
- iv) Region: The Greater London Area is an urban area and access to LTE is available.

REFERENCES

- [1] B. Ray, "NB-IoT vs. LoRa vs. Sigfox | Blog | Link Labs", Link-labs.com, 2022. [Online]. Available: <https://www.link-labs.com/blog/nb-iot-vs-lora-vs-sigfox>. [Accessed: 04- Jan- 2022].
- [2] K. Mekki, E. Bajic, F. Chaxel and F. Meyer, "A comparative study of LPWAN technologies for large-scale IoT deployment", ICT Express, vol. 5, no. 1, pp. 1-7, 2019. Available: 10.1016/j.icte.2017.12.005.
- [3] L. IoT? and L. Architecture, "LoRa Architecture - LoRaWAN Tutorial", 3GLTEInfo, 2022. [Online]. Available: <https://www.3glteinfo.com/lora/lora-architecture/>. [Accessed: 04- Jan- 2022].
- [4] R. Sinha, Y. Wei and S. Hwang, "A survey on LPWA technology: LoRa and NB-IoT", ICT Express, vol. 3, no. 1, pp. 14-21, 2017. Available: 10.1016/j.icte.2017.03.004.
- [5] "What is SigFox technology", Student Circuit, 2022. [Online]. Available: <https://www.student-circuit.com/learning/year3/iot/what-is-sigfox-technology/>. [Accessed: 05- Jan- 2022].
- [6] J. Rubio-Aparicio, F. Cerdan-Cartagena, J. Suardiaz-Muro and J. Ybarra-Moreno, "Design and Implementation of a Mixed IoT LPWAN Network Architecture", Sensors, vol. 19, no. 3, p. 675, 2019. Available: 10.3390/s19030675.
- [7] J. Blackman, "Three NB-IoT deployment models – and why NB-IoT performance is not what it should be", Enterprise IoT Insights, 2022. [Online]. Available: <https://enterpriseiotinsights.com/20190718/channels/fundamentals/three-nb-iot-deployment-models>. [Accessed: 06- Jan- 2022].
- [8] "SIGFOX.COM", Sigfox.com, 2022. [Online]. Available: <https://www.sigfox.com/en/what-sigfox/technology>. [Accessed: 06- Jan- 2022].
- [9] "Different Types of Motion Sensors and How They Work", ElProCus - Electronic Projects for Engineering Students, 2022. [Online]. Available: <https://www.elprocus.com/working-of-different-types-of-motion-sensors/>. [Accessed: 05- Jan- 2022].
- [10] J. Cotrim and J. Kleinschmidt, "LoRaWAN Mesh Networks: A Review and Classification of Multihop Communication", Sensors, vol. 20, no. 15, p. 4273, 2020. Available: 10.3390/s20154273.
- [11] "IoT Connectivity - Comparing NB-IoT, LTE-M, LoRa, SigFox, and other LPWAN Technologies", IoT For All, 2022. [Online]. Available: <https://www.iotforall.com/iot-connectivity-comparison-lora-sigfox-rpma-lpwan-technologies>. [Accessed: 05- Jan- 2022].
- [12] e. notes, "What is SIGFOX - M2M IoT » Electronics Notes", Electronics-notes.com, 2022. [Online]. Available: <https://www.electronics-notes.com/articles/connectivity/sigfox/what-is-sigfox-basics-m2m-iot.php>. [Accessed: 05- Jan- 2022].
- [13] "11 Internet of Things (IoT) Protocols You Need to Know About", Rs-online.com, 2022. [Online]. Available: <https://www.rs-online.com/designspark/eleven-internet-of-things-iot-protocols-you-need-to-know-about>. [Accessed: 06- Jan- 2022].

- [14] S. Narayana, R. Prasad, V. Rao, T. Prabhakar, S. Kowshik and M. Iyer, "PIR sensors", Proceedings of the 14th International Conference on Information Processing in Sensor Networks, 2015. Available: 10.1145/2737095.2742561 [Accessed 21 January 2022].
- [15] J. Lee, K. Lee and S. Choi, "A Design of Motion Detecting Sensor using Microwave," 2007 Asia-Pacific Microwave Conference, 2007, pp. 1-4, doi: 10.1109/APMC.2007.4554623.
- [16] Weiming Hu, Tieniu Tan, Liang Wang and S. Maybank, "A survey on visual surveillance of object motion and behaviors," in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 34, no. 3, pp. 334-352, Aug. 2004, doi: 10.1109/TSMCC.2004.829274.
- [17] H. Rahaman, V. Dyo, "Counting calories without wearables: Device-free Human Energy Expenditure Estimation". 16th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2020), October 2020.
- [18] H. Rahaman and V. Dyo, "Tracking Human Motion Direction with Commodity Wireless Networks", IEEE Sensors Journal, Volume: 21, Issue: 20, October 15, 2021, DOI: 10.1109/JSEN.2021.3111132
- [19] F. Lamonaca, P. F. Sciammarella, C. Scuro, D. L. Carnì and R. S. Olivito, "Internet of Things for Structural Health Monitoring," 2018 Workshop on Metrology for Industry 4.0 and IoT, 2018, pp. 95-100, doi: 10.1109/METRO14.2018.8439038.
- [20] S. A. Ellwood, C. Newman, R. A. Montgomery, V. Nicosia, C. D. Buesching, A. Markham, C. Mascolo, N. Trigoni, B. Pasztor, V. Dyo, V. Latora, S. E. Baker, D. W. Macdonald. "An active-radio-frequency-identification system capable of identifying co-locations and social-structure: Validation with a wild free-ranging animal". Methods in Ecology and Evolution, John Wiley & Sons. 2017, vol. 8, no.12,
- [21] V. Velisavljevic, E. Cano, V. Dyo, B. Allen, "Wireless Magnetic Sensor Network for Road Traffic Monitoring and Vehicle Classification". Transport and Telecommunication, 2016, vol. 17, no. 4, 274–288
- [22] Foubert, B.; Mitton, N. Long-Range Wireless Radio Technologies: A Survey. Future Internet 2020, 12, 13. <https://doi.org/10.3390/fi12010013>
- [23] J. Yun and M.-H. Song, "Detecting direction of movement using pyroelectric infrared sensors," IEEE Sensors J., vol. 14, no. 5, pp. 1482–1489, May 2014.
- [24] D. H. Kim, J. B. Park, J. H. Shin and J. D. Kim, "Design and implementation of object tracking system based on LoRa," 2017 International Conference on Information Networking (ICOIN), 2017, pp. 463-467, doi: 10.1109/ICOIN.2017.7899535.
- [25] Han Y., Jeong J. (2020) Air Pollution Measurement Platform Based on LoRa and Blockchain for Industrial IoT Applications. In: Gervasi O. et al. (eds) Computational Science and Its Applications – ICCSA 2020. ICCSA 2020. Lecture Notes in Computer Science, vol 12250. Springer, Cham. https://doi.org/10.1007/978-3-030-58802-1_44
- [26] Rashmi Sharan Sinha, Yiqiao Wei, Seung-Hoon Hwang, A survey on LPWA technology: LoRa and NB-IoT, ICT Express, Volume 3, Issue 1, 2017, Pages 14-21, ISSN 2405-9595
- [27] R. Ratasuk, B. Vejlgard, N. Mangalvedhe and A. Ghosh, "NB-IoT system for M2M communication," 2016 IEEE Wireless Communications and Networking Conference, 2016, pp. 1-5, doi: 10.1109/WCNC.2016.7564708.
- [28] <https://github.com/wallysalami/tkgpio>, Accessed 21/1/2022