Room Occupancy Monitoring

Internet of Things: System, Security, and The Cloud Project

Wei Tat Lee*
s1451142@ed.ac.uk
University of Edinburgh
Edinburgh

Alex Stewart†
s1437078@ed.ac.uk
University of Edinburgh
Edinburgh

ABSTRACT

Using the time of flight, distance sensors to measure distances, we are able to build a room monitoring system. We combine two distance sensors to count the number of a person entering or exiting the door of the room. In this paper, we design and prototype an end to end `IoT` system that consists of the embedded systems, the mobile application and visualization dashboard. We are able to achieve a 81% rate of accuracy in monitoring the occupancy of the room.

KEYWORDS

Internet of Things, Room occupancy monitoring, Embedded Systems, Cloud

1 INTRODUCTION

The increasing quality of lifestyle has caused human to rely on smart electronic devices. This includes small and cheap sensors that can be deployed in almost any environment. Each of these sensors normally collects raw data from the environment. For example, this could be a temperature reading from a temperature sensor. When it can be connected to the internet, it further unlocks the potential of such devices. Raw data is sent to the cloud can be aggregated in the server to apply some simple data analysis. One could do simple useful tasks such as monitoring the temperature or having a gas leak detection mechanism in an environment

The advantages of using such devices are that they are cost-effective and scalable. Internet of Things (`IoT`), have been deployed and used in public and private sectors. `AirBus` partially automates the production of airbuses using `IoT` devices to replace manual labors [1]. `Amazon` manage to reduce 20% operating costs by using `Kiva` robots to locate and fetch the products in warehouses [2]. It [3] is shown that that smart lights are installed in streets and public areas. The smart lights can adjust its brightness according to the weather conditions or can increase the lighting in high crime rate areas.

In this project, we developed a simple full-stack `Internet of Things` (IoT) system that monitors the occupancy of a room. The end to end pipeline consists of the following separate modules:

- A firmware for an embedded system connected with multiple distance sensors.
- A simple `Android` app that acts as a gateway to forward the raw data to the cloud.
- A cloud-based analytics that aggregates the raw data and host a visualisation dashboard using `Firebase`.

The approach that we used to monitor the room is by detecting the motion of people entering or exiting the room. 2 time of flight sensors (TOF) VL503LX, connected to the embedded system (NRF51DK) is attached to the door targeted room to monitor. Time stamps of a person being detected entering and exiting room are then sent to the `Firebase` database server using a mobile phone as a gateway.

There are a few assumptions that are introduced to simplify the design of the system. One such assumption is that there is only one way to enter the targeted room. This assumption could be removed simply by attaching more distance sensors to the other entrances. However, this is not scalable since we have to increase the number of sensors based on the number of entrances. In addition, the mobile phone with an internet connection has to be within reach of the embedded system. There are a few alternatives to replace the mobile phone as a gateway. One of them would be to integrate a `WiFi` module in the embedded system. We can also integrate Low-power wide-area networks (`LPWAN`) to the system to replace the mobile phone. These networks offer low power usage, low cost and enables long distances connections which are all suitable for `IoT` devices. One challenge that were faced in this monitoring system is that the sensor measurements could be noisy and need to be tuned properly.

In **Section 2**, we list out the high level system requirements in detail. In **Section 3** we discuss the design of the system in more detail. We then evaluate our system in **Section 4**.

2 REQUIREMENTS

In this section, the high level system requirements are listed in details.

2.1 Functional Requirements

(1) The system should collect timestamps of a person entering or exiting the room
(2) The system should store the timestamps in a cloud database
(3) The dashboard should calculated and display the occupancy of a targeted room with an hourly rate
(4) Deploy and host a visualization dashboard showing the occupancy of the room

2.2 Non-functional Requirements

(1) The accuracy of the system should be at least 80%
(2) The occupancy information on the dashboard should be updated hourly

*Both authors contributed equally to this research.
†Both authors contributed equally to this research.

3 DESIGN

In this section, we will be discussing the full architecture of the room occupancy monitoring system. The system consists of mainly 3 parts, which are the firmware for the embedded systems, the gateway (Android application) and the cloud server.

3.1 Firmware for Embedded System

A development board is used that have to suit the requirements of the system. This includes having `Bluetooth` connectivity that allows communicating with the Android application. In addition, the development board needs to have standard `GPIO` (General Purposes Input Output) ports to allow interfacing with the sensors that we are using. We were provided with the `Nordic` nRF51 Development Kit. It contains a $32-bit$ ARM Cortex M0 CPU core with $256kB/32kB$ of flash and RAM. The development kit also supports `BLE` (Bluetooth Low Energy). We developed our firmware using the `MBED 5` OS.

The firmware is programmed to have two main tasks. The first task is to detect a person entering or exiting the room and store the timestamps associated with it. To achieve that, we interfaced 2 `VL53L0X` time of flight distance sensors to the board. It acts as a slave using the `I2C` protocol. This allows us to connect multiple[1] distance sensors using only 1 `I2C` channel. We used the library from `MBED` [2] and modified it to interface 2 distance sensors instead of 1.

The distance sensors are attached to the door as shown in **Figure: 1** . Distance is measured in with a period of $100ms$. If the measured distance is within the range $\in [10mm, 750mm]$, it then counts as triggered. **Figure: 2** shows the flow diagram of how a person entering or exiting the room is detected. We used the order of the distance sensor being the trigger to decide on the detected state. The current timestamp relative to the system startup time is then stored in the local buffer.

The second task of the firmware is to send the timestamp data through the mobile phone. We used the `BLE` API from `MBED` to advertise itself as a peripheral. This allows the mobile phone to detect and connect to the development board. The development of the mobile application is further discussed in **Section: 3.2**. The embedded device is programmed to have a custom `GATT` services. It consists of a read array and write `GATT` characteristics. The peripheral device post the available time stamp data into the read array characteristic when the phone connects to the device. The write characteristic is used for the mobile phone to acknowledge the peripheral.

Using the `MBED` library, we are able to get timestamp data relative to the startup time of the board. The returned timestamp is a `Unix` epoch time that is $64-bits$. In order to save memory, we truncated the timestamp into $16-bits$. We also removed the least significant bit of the timestamp to reduce the resolution to be 2 *seconds* instead of 1 *seconds*. We then append the flag as an indication of the detected person entering or exiting the room to the most significant bit. The maximum time of the timestamp would then be $2^{16} - 1 = 65535s \approx 18Hours$. This is fairly enough since we have the assumption that the phone would be periodically polling the data hourly.

[1]We noticed that using `MBED OS 5` with `RTOS` has made the distance sensor to fail frequently. This was fixed by disabling the `RTOS` module in the OS. This can be done by having $.mbedignore$ and ignoring the unused `RTOS` module.
[2]https://os.mbed.com/users/mjarvisal/code/vl53l0x_api/

Figure 1: The distance sensors attached to the door. The bottom figure shows a close up image of the distance sensors.

3.2 Gateway (Android application)

The android application is developed to only act as a gateway to post the data to the `Firebase` cloud server. Hence we modified the `nRF` android blinky application [3] to suit our application. The timestamp stored in the embedded device is relative to the startup time of the device. The android application has to convert the timestamp into actual `UNIX` timestamp. This is done having the embedded device to send the current timestamp on the `BLE` connection being established before sending the actual timestamp data. The mobile phone at the same time associate that timestamp to the current `UNIX` timestamp using the android library `Date()`. The phone could then further calculate the actual timestamps by just using the relative timestamps. It is then pushed to the cloud server.

[3]https://github.com/NordicSemiconductor/Android-nRF-Blinky

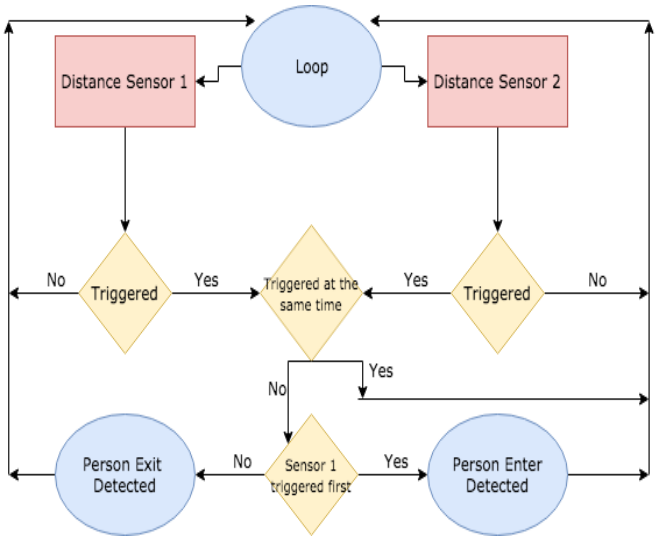

Figure 2: The flow diagram of person being detected.

3.3 Cloud Server

We used `Firebase` from `Google` to store our timestamp data. It is in the form of a key-value pair. The JSON structure of the data being stored is shown below. The high-level key is in the form of `UNIX` timestamp, and underneath it consists of the flag of the person entering or exiting the room. It comes with a unique key generated by `Firebase`, this allows multiple data to store underneath the similar timestamp. Do note that the timestamp uploaded is with a resolution of 1*ms*, but that the actual timestamp resolution is in 2*s*.

```
{
  "1553467127157" : {
    "-LalqrwdwPUFxLKAx9gR" : {
      "EnterRoom" : true
    }
  },
  "1553467276019" : {
    "-Lals-gC_QSjAglgqK-r" : {
      "EnterRoom" : false
    },
    "-Lals-gE1KjcNiaRu7p6" : {
      "EnterRoom" : false
    }
  }
}
```

Other than that, we are using `Firebase Functions` to aggregate the raw data that is being stored in the database. We simply have to create a function in `JavaScript` and deploy it to the `Firebase`. An example of the data being aggregated is shown below. It simply sums up the total number of enters and exits hourly across the day. Data is being aggregated every time a new data is being written to the `Firebase Database`.

```
{
  "24-2-2019" : {
    "0" : {
      "enter" : 0,
      "exit" : 1,
      "occupancy" : -1
    },
    "12" : {
      "enter" : 6,
      "exit" : 4,
      "occupancy" : 2
    },
    .
    .
    .
    "dailyEnter" : 90,
    "dailyExit" : 91,
    "dailyOccupancy" : 1.4545
  }
```

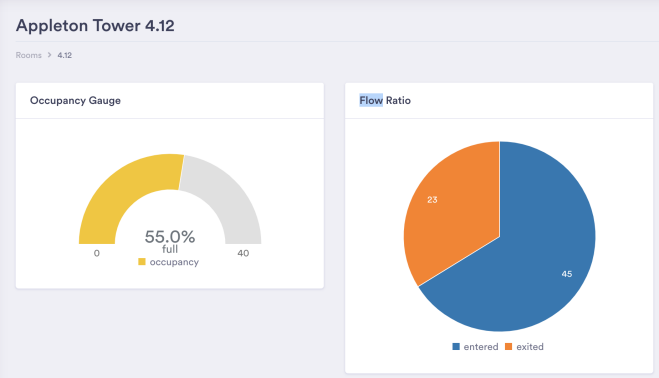

Figure 3: A screenshot of the visualization dashboard.

Figure: 3 shows the screenshot of the visualization dashboard. It shows an occupancy gauge of the current hour which is 55.0%. Furthermore, it also shows the total number of enter and exits at the current hour.

4 EVALUATION

In order to evaluate our system, we conducted an experiment on the occupancy of the targeted room. Our system was set up at the targeted room as shown in **Figure 1**. Before starting the experiment, the total number of a person in the room was 12 and was set up to the initial occupancy for the system on startup. We started and kept the system running for 4 hours. The system would then calculate the occupancy, the total number of enters and the total number of exits across the time. These data are then stored in the cloud `Firebase` and compared later.

Table: 1 shows the overall accuracy of the system. We can see that our system is able to achieve an accuracy of 81%. This can be caused by the number of exits not captured correctly by our system. We can be easily caused by a person going to fast exiting the room.

However, we can see that the deployment of the system is not ideal. This is because a person can easily tamper with the embedded

Type	Our System	Ground Truth	Accuracy
Occupancy	33	41	81%
#Enters	62	66	94%
#Exits	29	37	65%

Table 1: The accuracy of our system evaluated across 4 hours.

device and sensors. Smart deployment of the devices can be done as future work to improve system reliability.

5 CONCLUSION

In conclusion, we prototyped a room monitoring system that uses only distance sensors. We showed how the end to end system architecture was designed and deployed. We also achieved an accuracy of 81% in monitoring the occupancy of the room. This show the capabilities of the monitoring of multiple rooms with only deployment of a few sensors and a board.

REFERENCES

[1] Adrian Bridgwater. 2017. Airbus: engineering the future of intelligent factories. https://internetofbusiness.com/airbus-engineering-intelligent-factories/

[2] Will Knight. 2015. At Amazon Warehouses, Humans and Machines Work in Frenetic Harmony. https://www.technologyreview.com/s/538601/inside-amazons-warehouse-human-robot-symbiosis/

[3] N. Ouerhani, N. Pazos, M. Aeberli, and M. Muller. 2016. IoT-based dynamic street light control for smart cities use cases. In *2016 International Symposium on Networks, Computers and Communications (ISNCC)*. 1–5. https://doi.org/10.1109/ISNCC.2016.7746112