Surveillance Tracking System

Internet of Talents
Truong Xuan Nguyen
Fiona Lin
Christopher Beckett

Content

Background

The use of private security has been exploding in Australia, with increased demand comes increased cost. Conventional alarm and surveillance systems are fixed in place, and are not able to adapt to changing environments. This requires the use of private security officers. With the majority of private security officers paid between $600-799/week[1]. Increasing the effective coverage area of each officer stands to drive major savings.

Significance

In order to address the shortcommings of traditional surveillance systems, a IOT surveillance tracking system aims to be:

Highly accurate, allowing less human oversight

Due to unpredictable nature of human resources such, as staff performance, staff turnover and shift work schedule, the accuracy of system is not guaranteed. Using IOT, the surveillance tracking system can monitor and report on Key Performance Indicators (KPI) in an automated manner.

Portable, allowing it to monitor changing locations

A lot of traditional monitoring equipment is cabled heavily and fixed in place. It reduces the monitorable space and location. A single simple relocation of equipment requires registered cablers and possible building modifications.

Without the need for expensive vision systems

A traditional surveillance system can be resource intensive, they typically depend heavily on vision systems, such as expensive low light camera sensors, and are required to be cabled due to the high bandwidth requirements.

Project Scope

- Evaluate the pros and cons in comparison with other traditional fixed active surveillance systems.
- Design and prototype a surveillance system passively track a single human moving through its monitored space (a classroom sized space).
- Notify and present the tracking movements vividly to end users.

System Architecture

Hardware Requirement

Sensor block

Each sensor block contains a passive infrared (PIR) sensor, connected to an Arduino board with a WiFi module installed on it. There will be four sensor blocks deployed to the test environment where each block covers a specific area of the room. Upon detection of a movement (i.e. the PIR sensor is triggered), the sensor block forwards a GET request to the central server updating its current status. The format of the the GET request is as follows:

---

[1] Prenzler T, Earle K & Sarre R. 2009. *Private security in Australia: trends and key characteristics*. Trends & issues in crime and criminal justice No. 374. Canberra: Australian Institute of Criminology. https://aic.gov.au/publications/tandi/tandi374

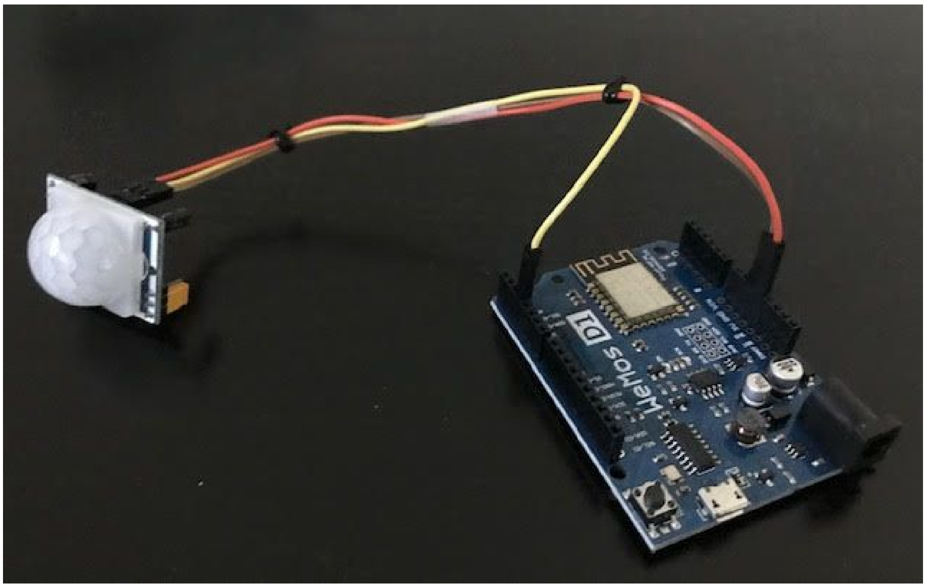
`GET /api/sensors/MAC_ADDR/STATE`
Where MAC Address is the sensor block's unique Wi-Fi NIC MAC address, and State is either 0 for no motion detected or 1 when motion is detected.

Central server
Google App Engine was used to host the central server.

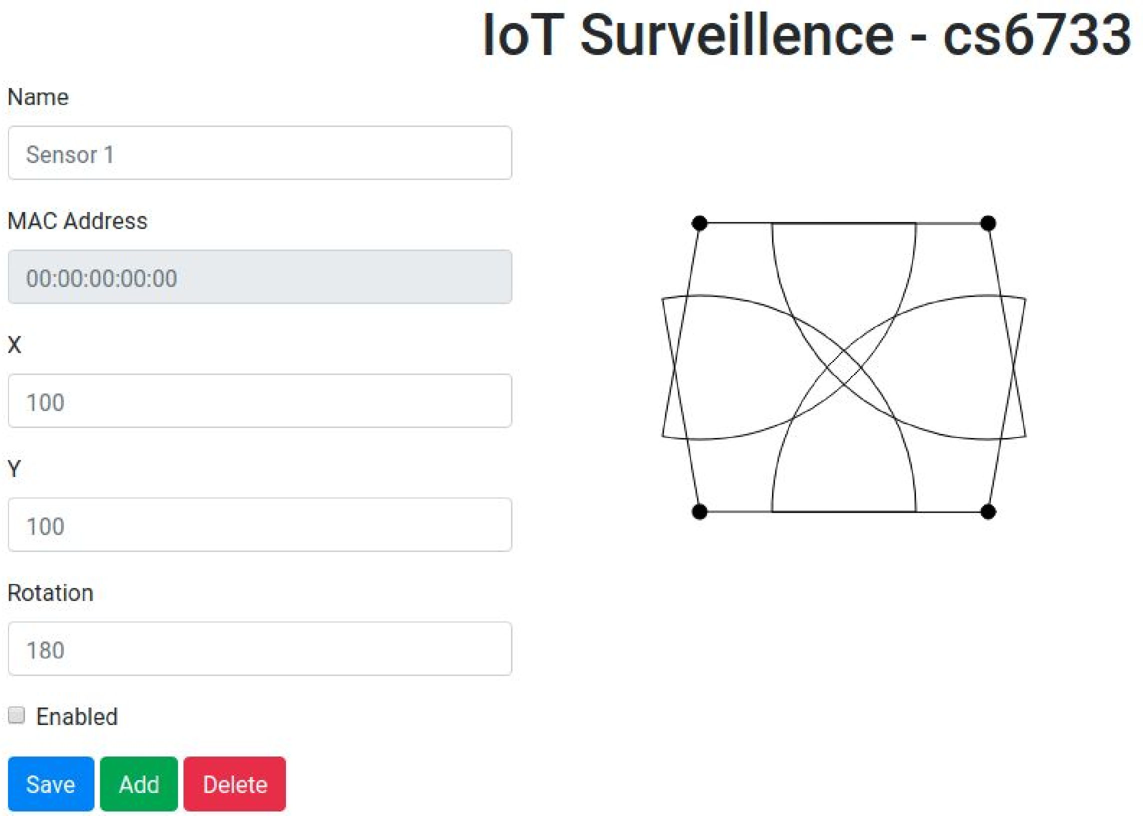
User Device
Any modern device running a web browser can be used to view the output from the sensor network.

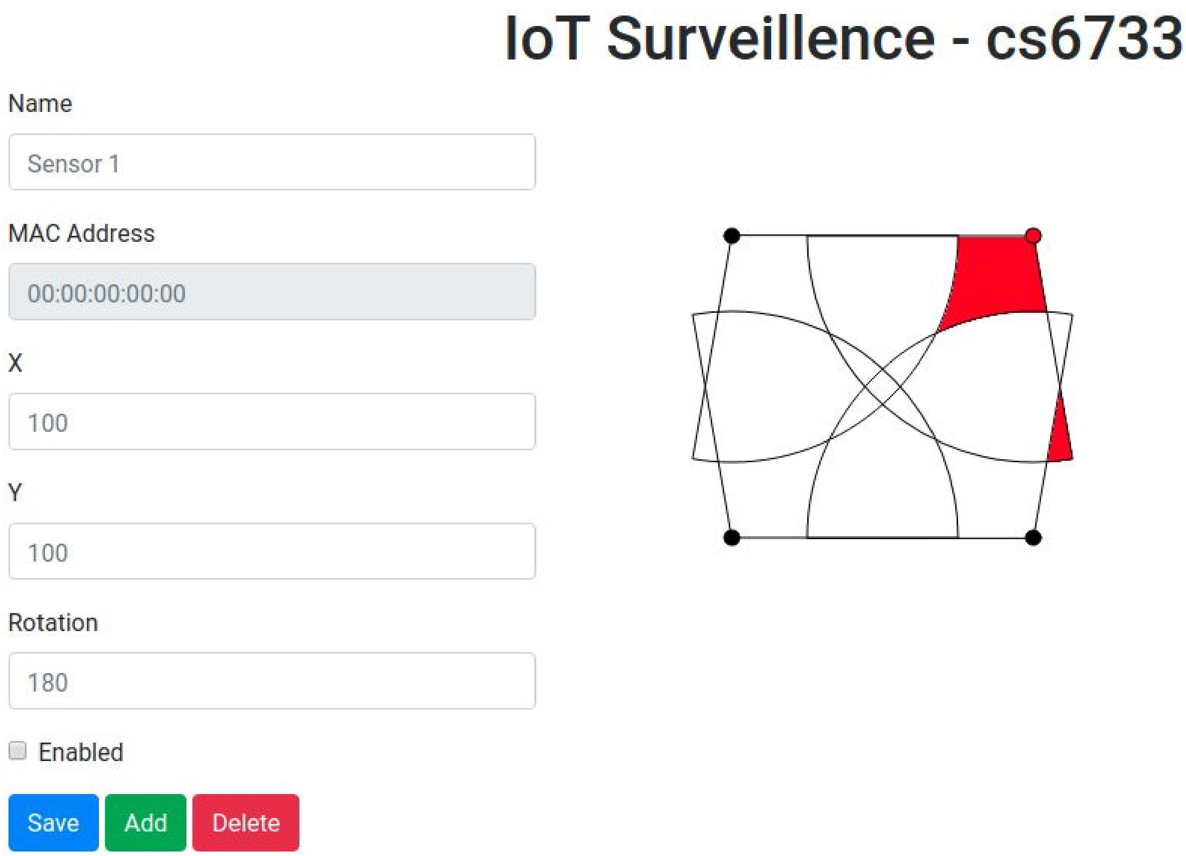
Software Requirement

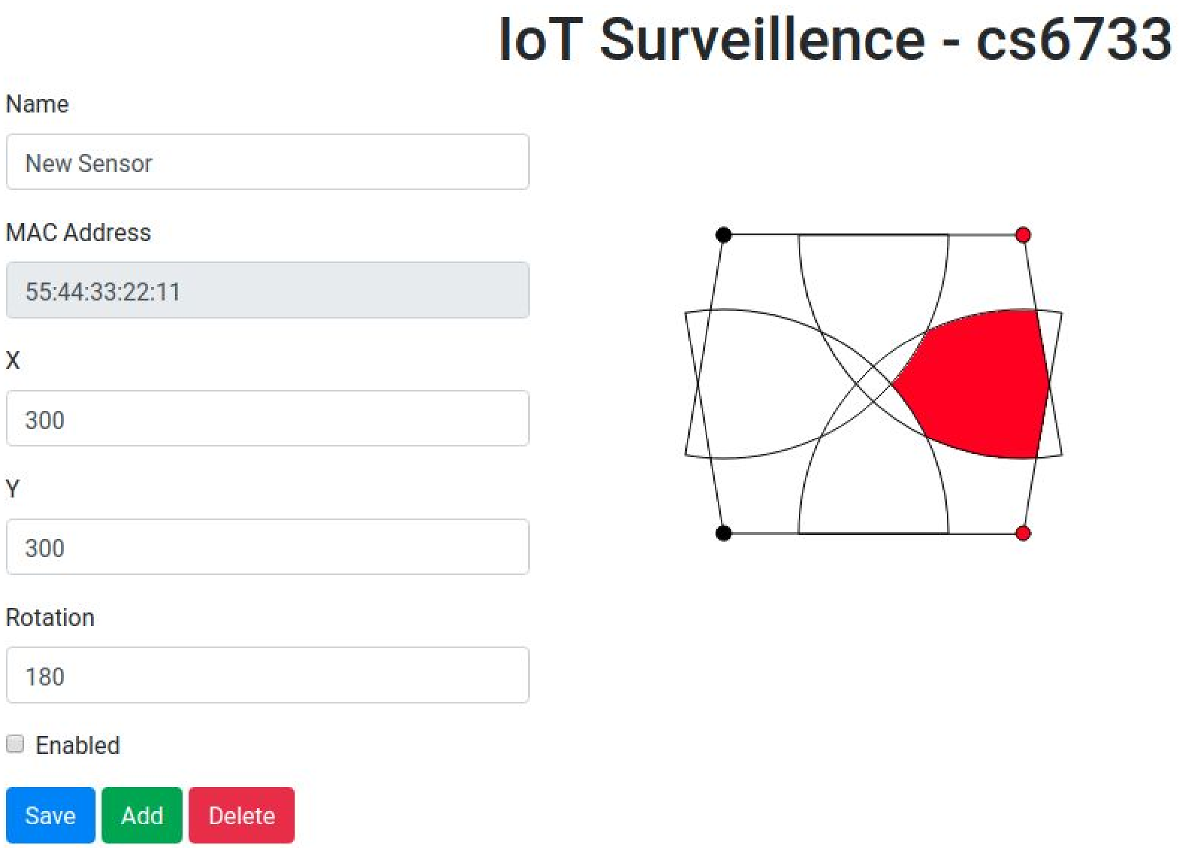
Sensor block
The PIR sensor detects object movements by monitoring the changes in IR radiation. When there is no movements detected, the sensor constantly set its output pin to LOW. When the sensor detects a movement, it would toggle the output pin to HIGH. The Arduino board communicates with the PIR sensor via C-programming language. The MCU repeatedly poll the sensor every one second to retrieve the sensor's latest state. When the MCU recognises a change in the PIR's status, it forwards a GET request to the local server, asking the server to update its database and reflecting the change on the UI.



Central server
A python Google App Engine application is used to present an API for sensor blocks to notify of motion state changes. It was constructed using the Flask web application microframework, and Google Datastore for presisting sensor data. Both the App Engine app, and the storage backend are massively scaleable, such that hundreds of thousands of nodes could easily be serviced without any code changes. A complete API reference has been included in appendix 3.
In addition to the API, the web application serves a web page, which using javascript and AJAX requests displays the sensor data to the end user in a friendly UI. p5.js was used to render the graphics.


IoT Surveillence - cs6733
Name
Sensor 1
MAC Address
00:00:00:00:00
X
100
Y
100
Rotation
180
Enabled
Save
Add
Delete


IoT Surveillence - cs6733
Name
Sensor 1
MAC Address
00:00:00:00:00
X
100
Y
100
Rotation
180
Enabled
Save
Add
Delete

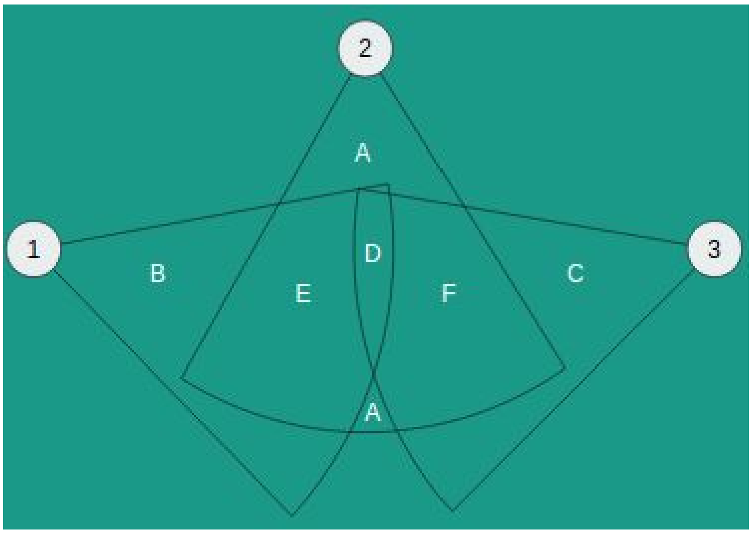


Localisation (detection algorithms)

Once the sensor nodes are set up, the user will configure where they are located in relation to each other, this will allow the software to detect the location of the intruding object by finding the only intersection of sensor coverage and displaying that similar to a Venn diagram on a map of the area.

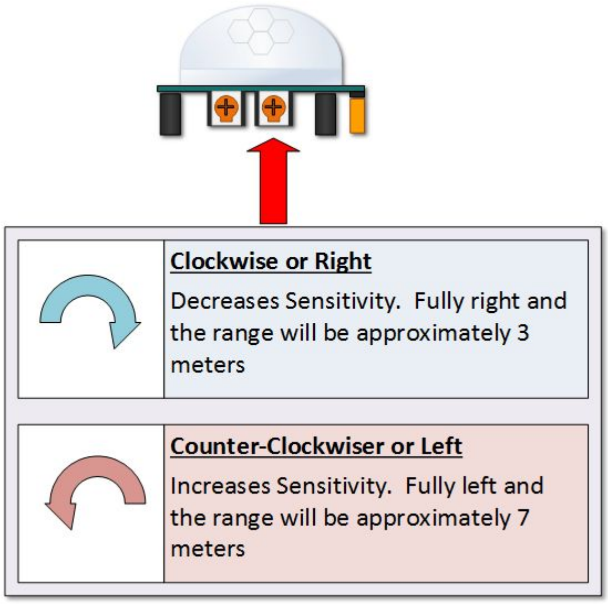
It is being completed on the UI component to enable the processing of motion data to enable the localization of a subject moving through the monitored are

Sensors detecting motion	Where is the motion localized?
2 Only	A
1 Only	B
3 Only	C
1 & 2 & 3	D
1 & 2	E
2 & 3	F

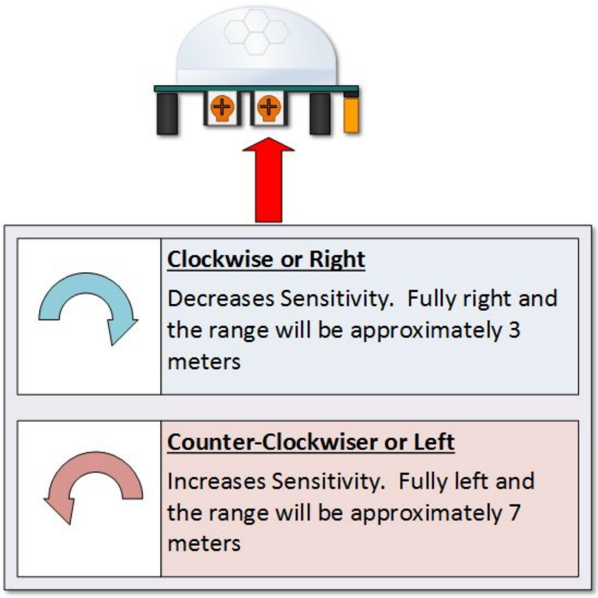


Quality of Analysis

The PIR sensors used in this system have the operating (sensitivity) range of minimum 3m and maximum 7m. This range can be adjusted by turning the right-hand-side variable resistor on the device (refer to Figure x). The angle of detection is fixed to 100° cone.



The signal on the output pin of the sensor, once any movements is detected, remains HIGH for a certain amount of time before resetting itself to LOW and reinitiating the next detection cycle. This time delay can be adjusted by turning the left-hand-side variable resistor on the device (refer to Figure x). The time delay can be set to any value within the range 3 seconds to 5 minutes.



For the purpose of continuous motion monitoring of this project, the trigger mode of the sensors is configured to be “Repeatable Trigger”. There are two trigger methods designed for this type of PIR sensor, namely the “Single Trigger” mode and the “Repeatable Trigger” mode.

For the “Single Trigger” mode, the time delay begins immediately upon detecting motion and the sensor halts until the time delay completes. When timed up, the sensor resets its output signal to LOW and remains LOW for 3 seconds then begins its next detecting phase.

In contrast, the “Repeatable Trigger” mode allows the sensor to continuously monitor the area, detecting whether any motion occurs. In this mode of operation, the sensor resets its time delay everytime motion is detected and will not initialise itself until motion has stopped completely.

Project Budget

Component	Qty	Unit Cost	Sub Total
WeMOS D1 (Rev2) Arduino Compatible	4	8 AUD	32 AUD
PIR Sensor	4	6 AUD	24 AUD
Total Cost			56 AUD

Project Plan Review

The project was carried out smoothly and met the project goals efficiently. Working through three smaller stages of our plan, many problems and setbacks occured during the project. However, they were resolved based on strong collaboration among the team members.

Stage 1: System architecture proposal and project planning.

This stage set out a solid foundation to carry out the development of the whole project. All team members participated the project research and discussions and the brainstormed a draft system blueprint for implementation in the next stage.

It took lots of discussions and team member collaboration to cover the project scope, a potential framework for evaluation of project goals, the required hardware (microcontroller, wireless connection modules, associated equipment, etc.) experiments and selection and software (front-end and back-end technologies, communications protocol, programming language, etc.) analysis.

This stage has spanned from the team has formed to our first proposal report complete (weeks 4 - 7). During this stage, the common understanding of the overall plan formed. It has allowed the team members to to parallelise the subsequent tasks while maintaining a single focus on the end-goal of the project.

Stage 2: Product development.

Within the context of this project, the back-end refers to the design, features and inter-communications of the sensor blocks and the front-end is anything related to the central server and the user interface. Because of the limited timeframe of the project, the team shall be splitted into 2 branches in which one branch accounts for the back-end development and the other branch is in charge of the front-end development.

This is the main phase of the project because it covered the most of the milestones (weeks 8 - 12). Each member not only focused on their allocated subtasks but also cooperated on each component integration. Everyone took ownership of each component and overall system, so as to smooth out all the problems occurred. The thorough understanding of project plan in stage one helped us to alter the original plan efficiently without any further discussions or arguments.

The mainly setbacks during this stage are the 6LoWPAN protocol wifi module and border router and local server. Due to our original plan was over ambitious with regard to project needs and constraints, two extra components of our original plan were removed.

In this stage, Localisation (detection algorithms) was so difficult to develop and implement, that it required more time than we planned. Based on the previous research paper, most of algorithms are too complicated to our project scope. After working through all current the studies, it sums up a solution that is suitable for our project. It was completed finally with our best team collaboration and organisation as well as time management skill. This issue was resolved through time and effort from all team members, everyone was calm and cooperative without slowing down the whole progress.

Stage 3: Prototyping, testing and debug.

This stage covers the rest of the milestone (weeks 13 - 18), and the final prototype and field deployment are the focuses. All the system components developed in the earlier stages were integrated and aggregated at this stage to produce a complete system. Once the final prototype is ready, efforts will be made to deploy the system into the final Demo environment. A test environment has delivered to host the final prototype system. The testing and debugging of the proposed system are also the two main activities of this stage.

Conclusion and Future Work

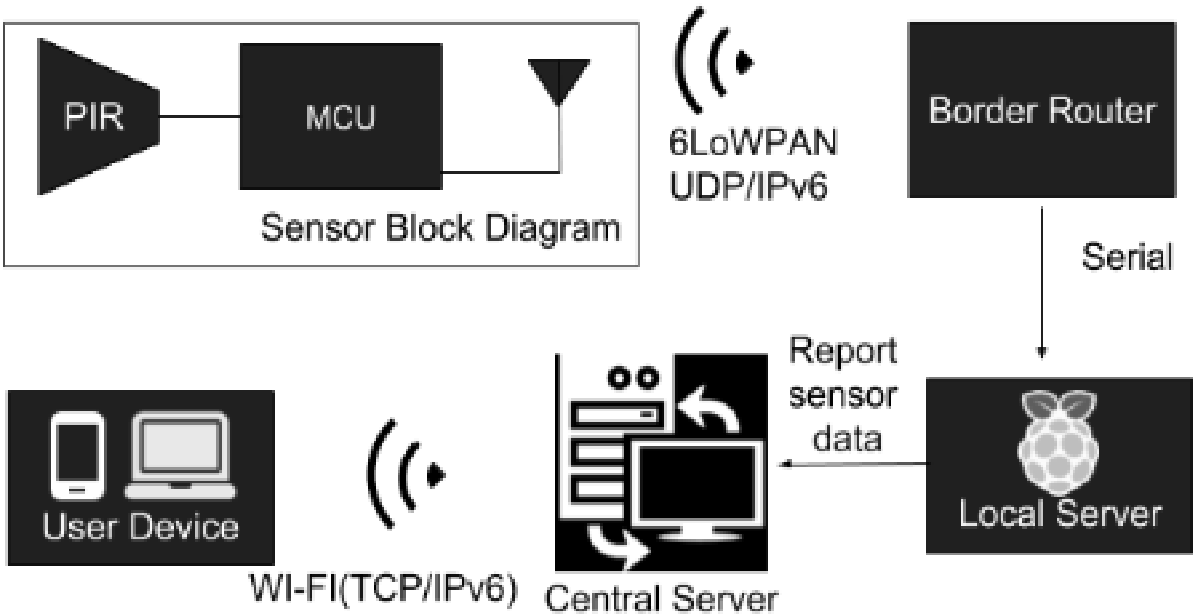
Due to time and hardware constraint, this project has delivered a fundamental working solution using wifi module and PIR sensor. It achieved the aim of a highly accurate, portable and resource independent surveillance tracking system. It lays the foundation for further movement study in machine learning.

This working solution has shown the location of moving object in the monitored space. With AI and machine learning, it is feasible to detect the direction and work out the speed of moving object and estimate the distance of object to PIR sensor.

Another further work is to develop the usage of 6LoWPAN in wifi module. This can reduce the power consumption of wifi module and improve the battery life of sensor device to meet the standard of IoT; also it can reduce the maintenance of battery.

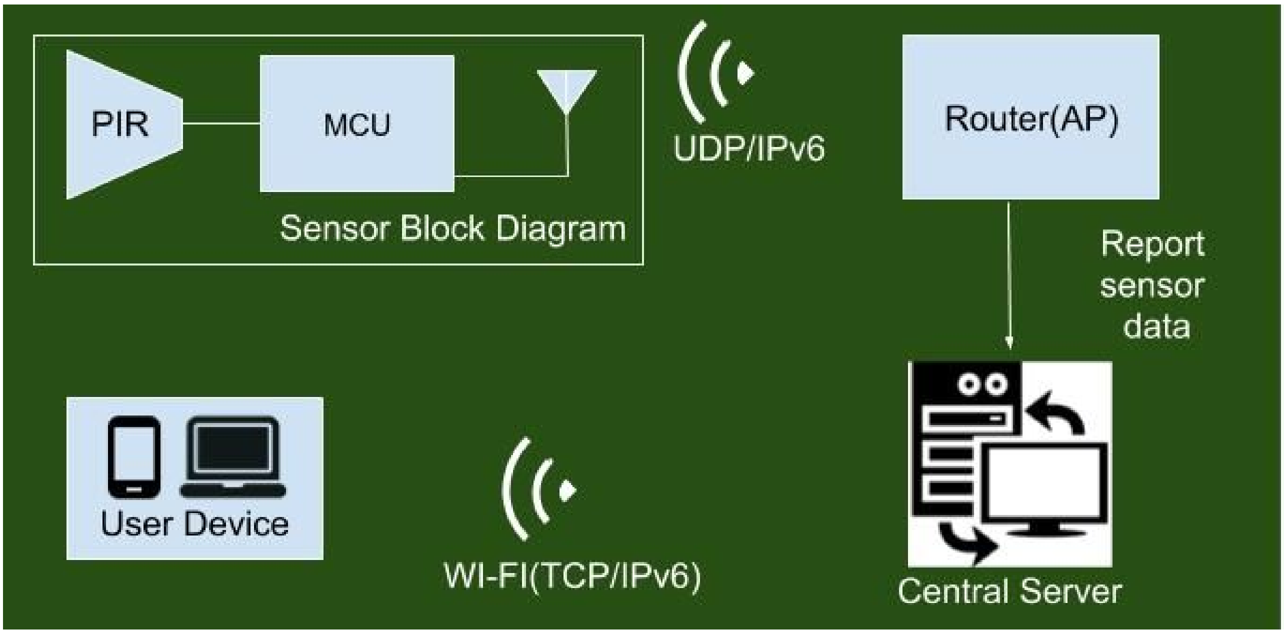
Appendix 1

Original design architecture diagram



Appendix 2

Current design architecture diagram



Appendix 3

Central server REST API

`GET /api/sensors`

Gets a list of all sensors

Sample payload:

```
[
	{
		“name”: “Sensor A”,
		“x”: 123,
		“y”: 321,
		“r”: 0,
		“motion_detected”: 1,
		“enabled”: 1,
		“mac_address”: “AA:AA:AA:AA:AA:AA”
	}
]
```

Example: `curl https://cs6733-220101.appspot.com/api/sensors`

`DELETE /api/sensors/<MAC ADDRESS>`

Delete a sensor.

Example: `curl --request DELETE https://cs6733-220101.appspot.com/api/sensors/00:14:22:01:23:46`

`POST /api/sensors`

Create a new sensor

Example: `curl --header "Content-Type: application/json" --request POST --data '{"name": "Beta", "enabled": true, "r": 180, "motion_detected": false, "mac_address": "00:14:22:01:23:47", "y": 1, "x": 4}' https://cs6733-220101.appspot.com/api/sensors`

`PUT /api/sensors/<MAC ADDRESS>`
Updates sensor record to new state. Send JSON as above
Example: `curl --header "Content-Type: application/json" --request PUT --data '{"name": "Beta", "enabled": true, "r": 180, "motion_detected": false, "mac_address": "00:14:22:01:23:47", "y": 1, "x": 5}' https://cs6733-220101.appspot.com/api/sensors/00:14:22:01:23:47`

`GET /api/sensors/<MAC ADDRESS>/<MOTION VALUE>`
Updates motion record for sensor. Motion value is 0 or 1.
Example: `curl https://cs6733-220101.appspot.com/api/sensors/00:14:22:01:23:47/1`

References

Song, B., Choi, H., & Lee, H.S. (2008). Surveillance Tracking System Using Passive Infrared Motion Sensors in Wireless Sensor Network. *2008 International Conference on Information Networking*, 1-5.

Sathishkumar, M., & Rajini, S.N. (2015). Smart Surveillance System Using PIR Sensor Network and GSM.

Henry's Bench. (2018). *Arduino HC-SR501 Motion Sensor Tutorial*. [online] Available at: http://henrysbench.capnfatz.com/henrys-bench/arduino-sensors-and-input/arduino-hc-sr501-motion-sensor-tutorial/ [Accessed 19 Nov. 2018].