

**MINI PROJECT - II**  
**6<sup>th</sup> Semester (CBGS)**  
Project Report

## **Low Cost Real-time Room Occupancy Indicating System**

*Submitted in partial fulfillment of  
the requirements of the term work for subject MINI PROJECT - II*

Submitted by

---

UCID	Names of Students
------	-------------------

---

2015110048	Chirag Shah
2015110037	Srijal Poojari

---

Under the guidance of  
**Prof. Priya Deshpande**



**Department of Electronics Engineering**  
Bharatiya Vidya Bhavan's  
Sardar Patel Institute of Technology  
Munshi Nagar, Andheri(W), Mumbai-400058  
University of Mumbai  
Academic Year 2017 - 18

## **CERTIFICATE**

This is to certify that this is a bonafide record of the project presented by the students whose names are given below during Semester VI in partial fulfilment of the requirements of the degree of Bachelor of Engineering in Electronics Engineering.

UCID	Names of Students
2015110048	Chirag Shah
2015110037	Srijal Poojari

**External Examiner**

**Internal Examiner**

(signature)

(signature)

Name:

Name:

Date:

Date:

**Seal of the Institute**

## **Abstract**

The objective of this action research based project was to tackle a problem faced by corporate environments. A typical corporate office consists of several meeting rooms with employees requiring frequent access to these meeting rooms, but lack of real-time knowledge of its occupancy leads to inconvenient hassle. The proposed solution consists of a network of motion detection sensors (namely, the PIR sensor) spread out across all meeting rooms, updating the room occupancy status in real-time to a central base-station. Employees using a web application or a smartphone can then check the occupancy of the rooms in real time.

This will lead to optimal utilization of the meeting rooms.

Each sensor node is designed to be Low Cost, Wireless and Battery Powered for seamless integration and have Low Power Consumption and to avoid frequent battery replacement.

## **Acknowledgement**

We would like to take this opportunity to thank Aashish Nehete for his immense help in creating and designing the web-based GUI for this project. We are also grateful to Prof. Kaiser Katchi for his help in 3D printing the project enclosure. Special thanks to our institute, Sardar Patel Institute of Technology, for providing us with 3D printing services and a platform for us to showcase our project.

Chirag Shah  
Srijal Poojari

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Hypothesis</b>	<b>2</b>
<b>3</b>	<b>Market Survey</b>	<b>3</b>
3.1	Workscape . . . . .	3
3.2	OccupEye . . . . .	4
<b>4</b>	<b>Hardware and Software Requirements</b>	<b>5</b>
4.1	Hardware Requirements . . . . .	5
4.2	Software Requirements . . . . .	5
<b>5</b>	<b>Implementation and Build</b>	<b>6</b>
5.1	Designing and Prototyping . . . . .	6
5.1.1	Radio Module Selection . . . . .	7
5.1.2	Low Power Considerations . . . . .	7
5.2	PCB Design . . . . .	8
5.3	3D Modelling and Print . . . . .	9
5.4	Interface with Master Node and Web-GUI . . . . .	10
<b>6</b>	<b>Results</b>	<b>12</b>
<b>7</b>	<b>Next Steps</b>	<b>14</b>
7.1	Testing with increased number of nodes . . . . .	14
7.2	Mobile Application . . . . .	14
7.3	Implementation at Fractal Analytics . . . . .	14
<b>8</b>	<b>Conclusion</b>	<b>15</b>

# Chapter 1

## Introduction

In a typical corporate environment there exists multiple conference/meeting rooms. The site that we studied was the Fractal Analytics Ltd, Goregaon. The office has around 400 employees: 250 employees on 7th floor, 150 employees on 3rd floor. 7th floor has 10 meeting rooms and 3rd floor has 5 meeting rooms. Anyone can book any meeting room for any time (if the room is available) using a mobile app. This is an open office - hence if anyone wants to have a discussion then they need to go to a meeting room. Hence meeting rooms are always in demand.

The problem was that anyone could book a meeting room and then not use it. Or if someone wanted to have a meeting without prebooking the meeting room the he/she would have to go from room to room to check the availability of the rooms. This would create a lot of unnecessary hassle and would lead to unoptimal utilization of the workspace.

To solve these problems we envisioned a solution which would involve placing sensors in every meeting room to monitor the occupancy of the room. This sensor will then relay the real time occupancy information of the room to a central device which will keep track of the occupancy status of all the meeting rooms. This data can then be showed on a web/mobile interface to check the real time occupancy status of the meeting room.

# **Chapter 2**

## **Hypothesis**

If there was a real-time meeting room occupancy monitoring system then users could remotely check in real-time

- If the room is occupied or not
- If a room is booked but not occupied
- If a room is not booked and not occupied
- The mobile app can use the data to cancel bookings if the room is unoccupied

This will lead to optimal utilization of the meeting rooms.

# Chapter 3

## Market Survey

The following products are available which provide similar functionality to our system

### 3.1 Workscape

Product Website: [www.workscape.io/products/manage/smart-sensors/](http://www.workscape.io/products/manage/smart-sensors/)

Device White Paper:

[www.workscape.io/wp-content/uploads/2017/08/ChargeSpot-Whitepaper-Unlocking-Employee-Pr.pdf](http://www.workscape.io/wp-content/uploads/2017/08/ChargeSpot-Whitepaper-Unlocking-Employee-Pr.pdf)

Workscape provides sensing devices similar to ours. It also provides meeting room booking and scheduling applications for mobile devices. Workscape provides the booking and scheduling applications independently or it also provides the applications along with the devices. It also uses the data generated by the sensors to provide analytics on room occupancy status over a period of time. The solutions offered by workscape are similar the ones proposed by us.

Workscape follows a subscription pricing policy as follows

- Basic: 8\$per room per month which includes Calendar sync Room display app, Mobile App, Web booking, Reporting and analytics
- Sensor: 15\$per room per month which includes Everything in Basic + Smart sensors + Room automation + Enhanced reporting and analytics

The "Sensor" solution costs 15\$per room per month which would equal to 2700\$for 15 rooms per year



## 3.2 OccupEye

Product Website: <https://www.occupeye.com/>

Product Whitepaper:

<http://www.occupeye.com/wp-content/themes/OccupEye/brochure/OCC-Brochure.pdf>

Similar to workscape, occupeye can also monitor the occupancy status of a room. In addition occupeye is also intended to serve as a workstation occupancy monitoring sensor. OccupEye is targeted towards improving the workspace utilization. Like our proposed solution all the sensors transmit to a central receiving station which then pushes the data to the Internet. OccupEye analytics is web-based portal that can generate analytics on room/workspace utilization. This can help enable dynamic workspace allocation and serve to promote efficient workspace utilization.



Figure 3.1: Workstation occupancy monitoring: Hot/Cold desk



Figure 3.2: Workstation/room occupancy sensing



Figure 3.3: Base station

m

# **Chapter 4**

## **Hardware and Software Requirements**

### **4.1 Hardware Requirements**

- ATmega328p Microcontroller
- nRF24L01 RF module w/ External Antenna
- OLED Module (for debugging)
- FTDI 232RL USB to Serial converter for programming and debugging
- Low quiescent current voltage regulator
- Other miscellaneous components

### **4.2 Software Requirements**

- Sloeber IDE or the Arduino IDE
- RF24, RF24Network and RF24 Mesh libraries by TMRh20
- Python 2.7
- Flask, SocketIO, Pyserial packages for Python
- HTML, CSS and Javascript for Web-GUI

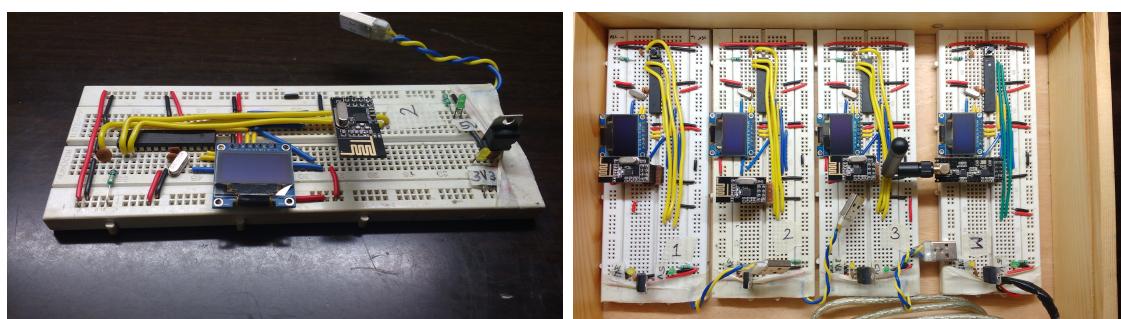
# Chapter 5

## Implementation and Build

### 5.1 Designing and Prototyping

We designed the prototype of the circuit on breadboards which included the microcontroller, radio module and an oled display.

- We used an atmega328p microcontroller as the library we needed for creating the mesh network with NRF24L01+ radio module was easily available for it.
- We added an oled display for testing and debugging purposes and is not intended to be a part of our final product.
- The breadboard circuit also had a 1117 3.3v regulator to supply the NRF radio module and the oled display.
- Programming the ATmega  $\mu$ C was done using FTDI USB to serial converter. The ATmega has an Arduino pro mini 8Mhz bootloader on it.
- We have used an 8Mhz crystal since we planed to reduce the supply voltage and lower clock speeds are required to reduce the supply voltage to the  $\mu$ C.



### 5.1.1 Radio Module Selection

- **ZigBee, Lora and the NRF24L01** were the suitable choice of wireless communication modules/protocols for our project.
- **ZigBee** offered a easy to use mesh network setup, low power consumption and robust communication protocol. But ZigBee was cost prohibitive for creating many inexpensive devices.
- **Lora** offered long range but had a nascent ecosystem and it too was cost prohibitive for creating many inexpensive devices.
- **NRF24L01+** modules offered a decent range had a resonable networking ecosystem developed for it and was very cost effective for creating low cost modules.

### 5.1.2 Low Power Considerations

As we were making a battery powered device the current consumed should be as low as possible to maximize the battery life. We desired a battery life of 3-12 months before the battery needs to be replaced. Hence to reduce the active and standby current of our circuit we took the following steps.

- When the device is not transmitting the microcontroller and the radio module go into a deep sleep state. The current consumption reduces to **80 $\mu$ A**.
  - **50 $\mu$ A** is used by the PIR sensor when motion is not detected. (150 $\mu$ A when motion is detected)
  - **20 $\mu$ A** is the quiescent current of the voltage regulator
  - **9 $\mu$ A** is the current consumption of the ATmega  $\mu$ C
  - **1 $\mu$ A** is taken by the nrf module in the **deep sleep** state
- We have taken various measures to reduce the power consumption of the ATmega  $\mu$ C.
  1. Put the processor to power down mode. This is the most power efficient sleep mode. This disconnects the clock from the CPU, flash memory, IO ports, ADC. It also disconnects the external crystal oscillator and the timer oscillator. BOD is also disabled. The only way to wake up the  $\mu$ C is by WDT or by external interrupts.
  2. Arrange to wake the processor from sleep only when needed based on watchdog timer overflow and interrupt by the PIR sensor
  3. Run the processor at a lower frequency (8Mhz)
  4. Run the processor at a lower voltage (3V) (This will also reduce the number of batteries to be used as the supply voltage reduces)
  5. Turn off unneeded internal modules in software (eg. SPI, I2C, Serial, ADC). This will reduce the power consumption  $\mu$ C is awake. They are disabled by default in the sleep state
  6. Turn off brownout detection

7. Turn off the Analog-to-Digital converter (ADC). This will reduce the power consumption  $\mu$ C is awake. It is disabled by default in the sleep state
  8. Set not required pins to output
- We needed a voltage regulator to reduce and regulate the supply voltage from the battery. Standard voltage regulators like the LM117 consume about 5mA operating current. Moreover the dropout voltage is too high which reduces the efficiency and also increases the voltage needed to be supplied to the device hence increasing the number of batteries. We decided to use **TC1014-3.0V** which is a low dropout voltage, low quiescent current voltage regulator. This enabled our quiescent current of the regulator to be as low as **17 $\mu$ A** and it reduced the dropout voltage to **200mV**. We purchased the voltage regulators online from [tanotis.com](http://tanotis.com).

When motion is detected the device wakes up from its deep sleep state using interrupts and then transmits the required information and goes back to its deep sleep state. During this state the current consumption increases to **18mA** for about 2 seconds before the device goes back to sleep. Out of these 18mA, 14.5mA is used by the NRF module and 3.5mA by the microcontroller.

## 5.2 PCB Design

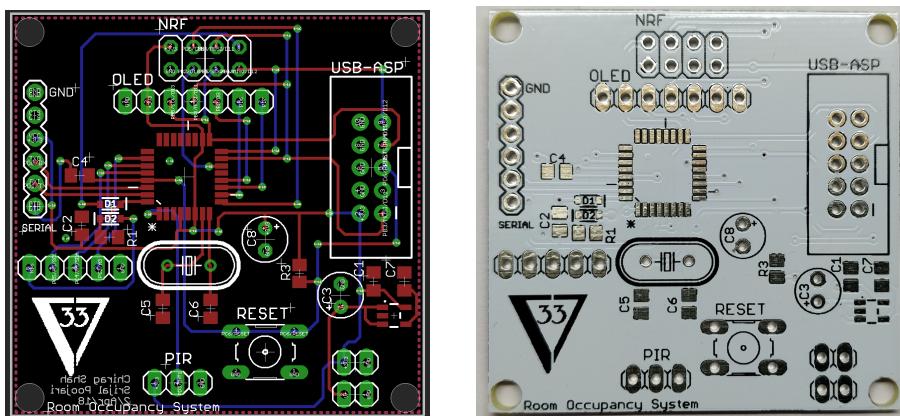
The board has the following features

- Programming interface for both serial (Arduino) as well as ISP programming
- Headers for RTC, NRF radio module and Oled display
- Onboard LDO and ultra low quiescent current voltage regulator for battery supply (TC1014-3.0V)

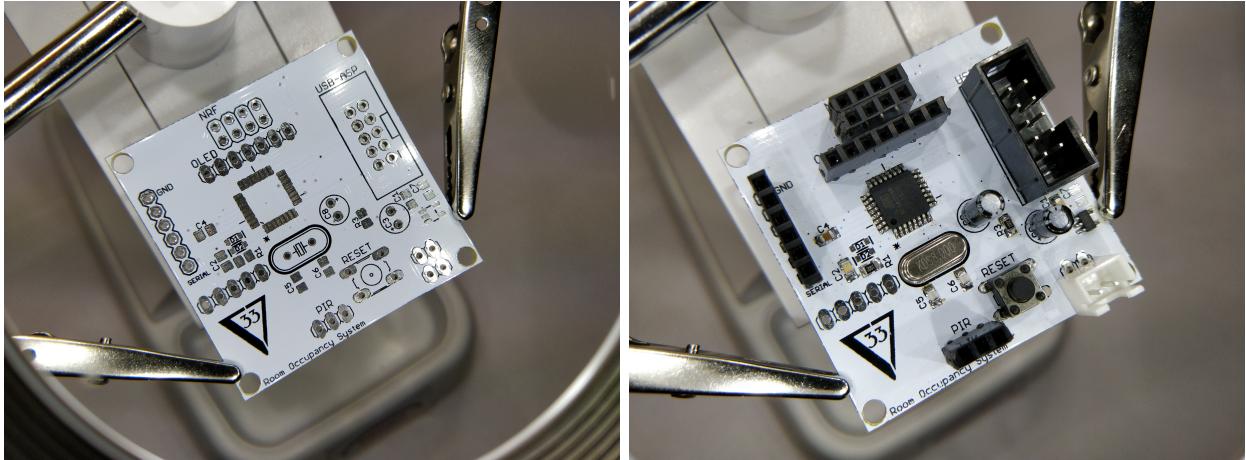
We used the smallest footprint components which was practical to hand assemble so that we could make the device smaller. All the resistors and capacitors are of **0805** package, ATmega is **TQFP** package, the voltage regulator is **SOT-5** package.

We needed to make multiple such boards and they needed to be double sided to support the high package density. Also the lead width is small for TQFP and SOT packages. The lead width is 0.3mm-0.45mm for TQFP. The lead pitch is 0.8mm for TQFP.

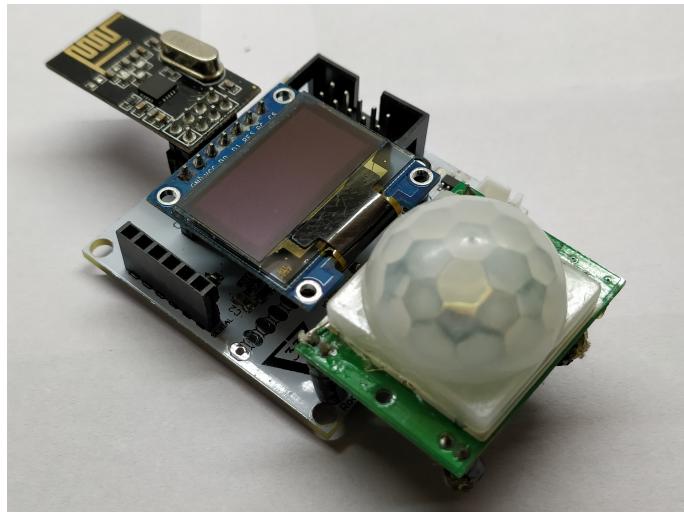
Hence for these reasons we decided to get the PCBs professionally manufactured from a PCB prototyping service ([PCBway.com](http://PCBway.com)).



Eagle CAD Design and Manufactured PCB



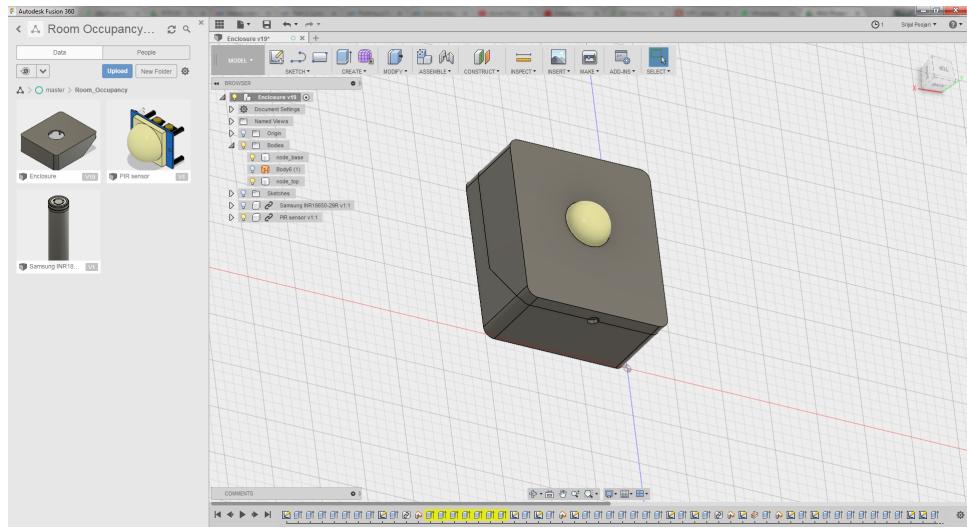
Populating the PCB



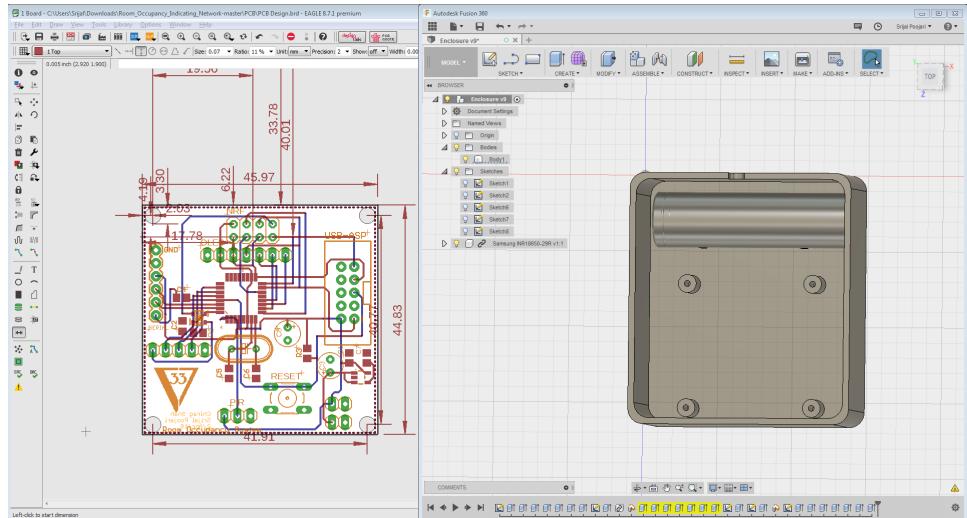
Board with all the components

### 5.3 3D Modelling and Print

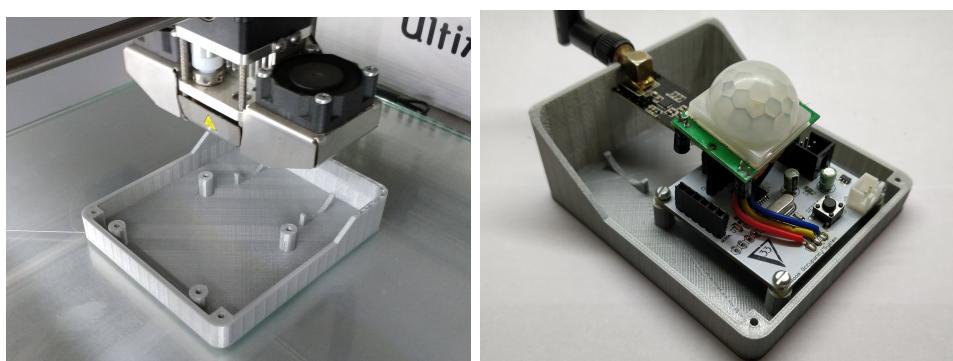
Any electronics project is incomplete without a proper enclosure to go with it. Normally we would have built something manually out of acrylic sheets or cardboard, but recently the institute had purchased a 3D printer, and we had multiple nodes to work with, so 3D printing an enclosure was the best option. We designed the 3D models by mainly taking the PCB, RF module and battery into consideration and tried to fit it a small form-factor. The modelling was done on Autodesk Fusion 360 from ground up. Pre-made models of the battery and PIR sensor was found on grabcad.com, which made the modelling process a bit easier. After the design process, the model was exported as .stl files and sent to the printer for 3D printing. The first print of the base part failed due to a nozzle clog in the 3D printer. The second print was a success and the print turned out well, with all components falling in perfectly. Next we went ahead and printed the top half of the enclosure and it was a success as well. The end result was a neat and aesthetically pleasing which gave the sensor nodes a product-like professional look.



Designing in fusion 360



Modelling the enclosure as per the PCB dimensions

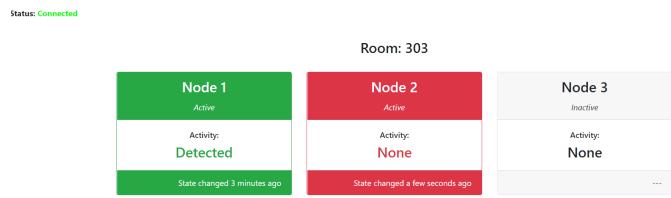
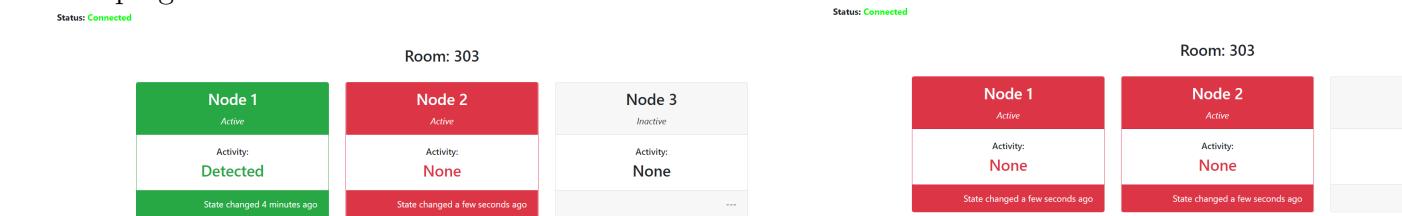


Printing and assembling the device

## 5.4 Interface with Master Node and Web-GUI

All packets sent from each sensor node are eventually routed to the master node, which relays it to the computer maintaining the web server. The interface between the master node and

the computer is done Serially, at a baud rate of 115200 bps. The computer runs a Python script which receives all the sensor data and updates the data on the web server, which displays it to the client (browser). Timestamps are passed between the the web server and client in Unix Epoch format, while a more precise timestamp is logged locally to a text file. Real-time information of all the sensor nodes had to be displayed in a easy to interpret format for the user. In case of OccupEye insert bib reference, the sensor information is displayed on top of the map of the room itself, which makes it incredibly easy to interpret. insert image of occupeye sensor state map Since, this is a demonstration model, and since the number of nodes is less, we do not have a fixed outline of a room available so the sensor information is just displayed on a simple browser based GUI. The webserver was created using the Flask micro web framework, based off Python, and the client end was designed using HTML, CSS and Javascript. High speed dynamic connection between the client and server is implemented using Socket.IO. The MomentJS javascript library was used for timestamping and does the timekeeping on the client end.



# Chapter 6

## Results

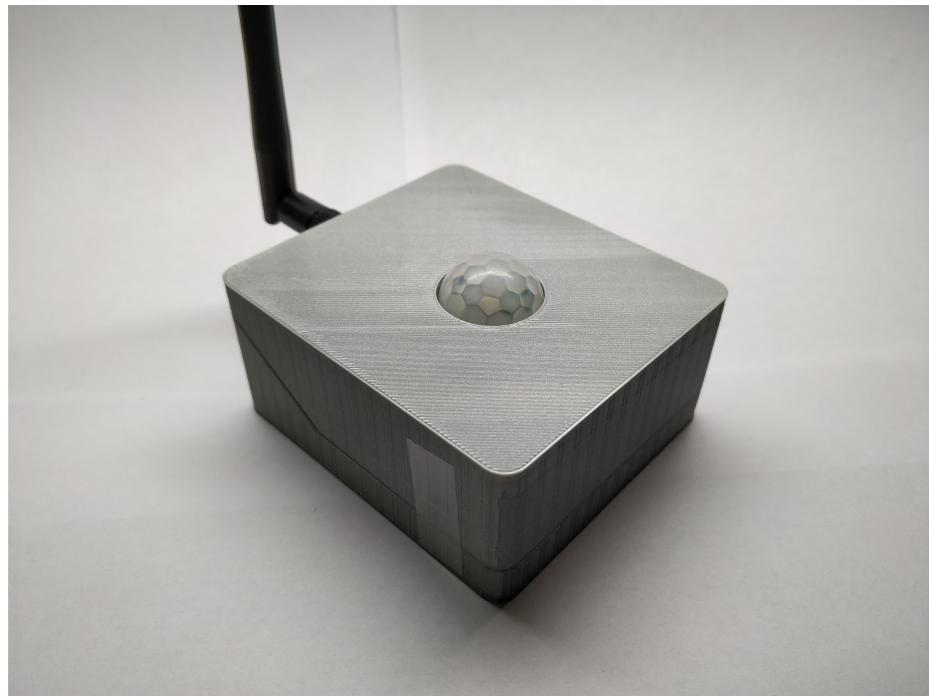
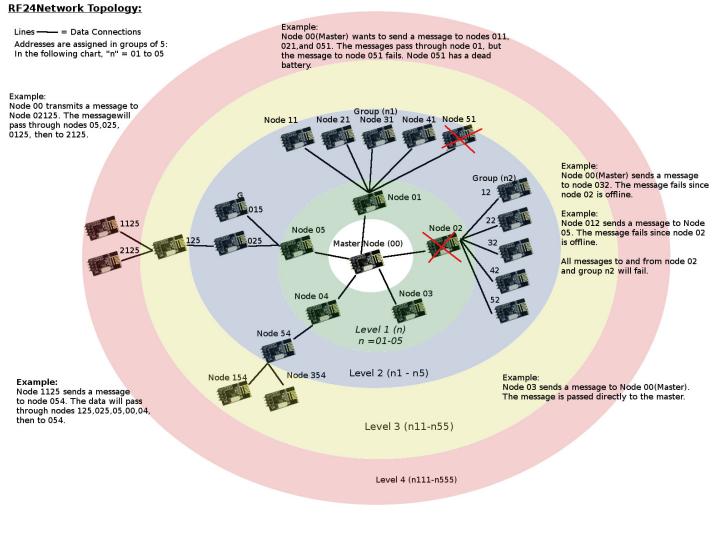


Figure 6.1: Final Assembled Device

The devices have been configured in a tree network setup. All the end nodes stay in a deep sleep state. They wake up only when a change of state in the PIR sensor is detected or when the WDT overflows.

When motion is detected the device wakes up from its deep sleep state using interrupts and then transmits the required information and goes back to its deep sleep state. The devices also periodically wake up and send the state when the WDT overflows. This is so that if a device fails the master can sense that the device hasn't transmitted anything recently and then that device can be marked offline by the user.

The routers are always awake and listening so that they can route information between the devices and the master whenever required. They can be battery powered or can be powered using a mains adapter.



Tree network topology

Status: Connected

Room: 303

Node 1	Node 2	Node 3
Active	Active	Inactive
Activity: <b>Detected</b>	Activity: <b>None</b>	Activity: <b>None</b>
State changed 4 minutes ago	State changed a few seconds ago	...

## Website

Real time occupancy status can be seen

# **Chapter 7**

## **Next Steps**

### **7.1 Testing with increased number of nodes**

Increasing the number of networked devices poses many challenges such as handling the required amount of connections, handling packet losses, planning deep sleep timings, etc and also may create some other unforeseen problems. This has to be tested and our next goal is to run 10 simultaneous devices (which we have available) and go further from there on.

### **7.2 Mobile Application**

Current demonstration runs off a local server with the client (browser) based on the host machine itself. Since the webserver is based off Flask (Python), deploying it over the internet is easy and only requires some paid subscription for server hosting. After this, the same Web GUI will run on any smartphone browser or can even be integrated into a mobile application.

### **7.3 Implementation at Fractal Analytics**

Create a set of 10 physical sensors which can be implemented on the 7th floor in Fractal Analytics Create a simple mobile app page - which will show the occupancy status of individual rooms on a smart phone This will give us an invaluable experience of understanding what it takes to implement a project in a real life live environment

# **Chapter 8**

## **Conclusion**

We were able to create a wireless tree network of low powered battery operated devices which would sense the occupancy of each room and then relay the occupancy status to a central master device. The device then relayed the real time occupancy status to a local server.

A web browser (client) can then dynamically, using websockets, view the status of each node in real-time.

# Bibliography

- [1] Peter M Visser, Marcel A Groothuis and Jan F Broenink, ‘Multi-disciplinary design support using Hardware-in-the loop simulation ,’ *Proceedings of the 5<sup>th</sup> Progress Symposium on Embedded Systems*, October 20, 2004, NBC Nieuwegein, NL pp 206-213.
- [2] Qing Li, and Caroline Yao: ‘Real-time concepts for embedded systems,’ *CMP Books*.
- [3] M.C. Chandorkar, Deepakraj M, ‘Control of Parallel Connected Inverters in Standalone ac Supply Systems,’ *IEEE Transactions Industry Applications*. vol. 29, no. 1, pp. 136-143, March 1993.
- [4] Ram Kelkar, Ronnie A. Wunderlich and Leonard J. Hitchcock,‘Software breadboards for power electronics circuits,’ *IEEE Transactions on Power Electronics* vol. 6, no. 2 pp. 170-179, April 1991.
- [5] Rainer Kraus and Hans Jurgen Mattausch, ‘Status and Trends of power semiconductor device models for circuit simulation,’ *IEEE Transactions on Power Electronics* vol. 13, no. 3, pp. 452-466, May 1998.