MINI PROJECT - II
6^{th} **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 Semister 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. We also thank our project guide and mentor, Prof. Priya Deshpande for her support and insights. 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

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. The would create a lot of unnecessary hassle and would lead to unoptitmal utilization of the workspace.

To solve these problem 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 fuctionality to our system

3.1 Workscape

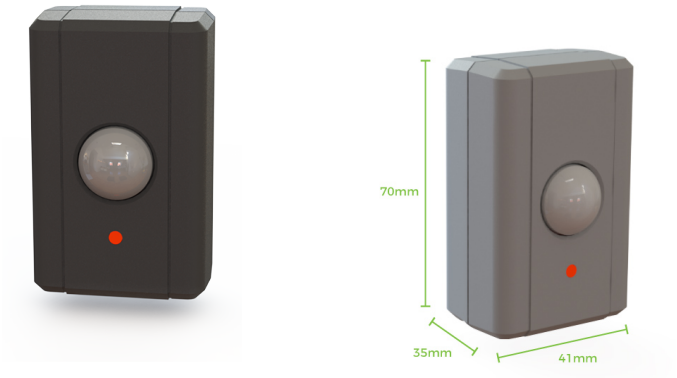
Product Website: www.workscape.io/products/manage/smart-sensors/
Device White Paper: https://goo.gl/K2D8hQ

Workscape[6] 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 indepedantly 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 subscricption 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: https://goo.gl/PpC2pw

Similar to workscape, occupeye can also moniter the occupancy status of a room. In addition occupeye is also intended to serve as a workstation occupancy monitoring sensor. OccupEye[5] 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

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

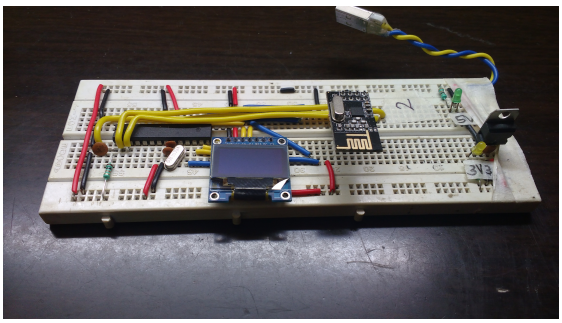
Chapter 5

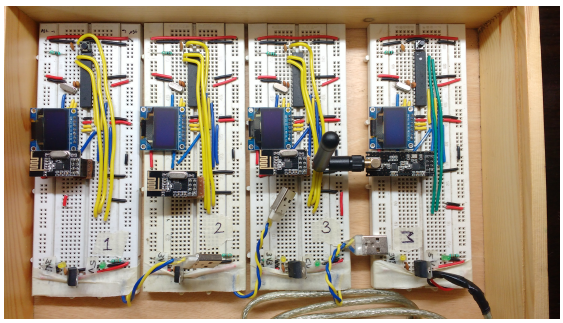
Implementation and Build

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 μC 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 μC.





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 **80μA**.
  - **50μA** is used by the PIR sensor when motion is not detected. (150μA when motion is detected)
  - **20μA** is the quiescent current of the voltage regulator
  - **9μA** is the current consumption of the ATmega μC
  - **1μA** is taken by the nrf module in the **deep sleep** state
- We have taken various measures to reduce the power consumption of the ATmega μC.
  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 μC 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 μC 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 μC 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 quiecent current voltage regulator. This enabled our quiescent current of the regulator to be as low as **17μA** and it reduced the dropout voltage to **200mV**. We purchaced the voltage regulators online from tanotis.com.

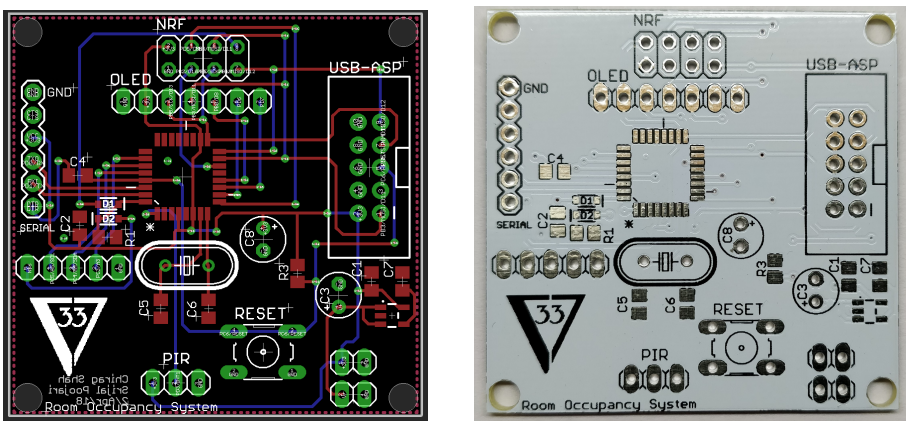
  When motion is detected the device wakes up from its deep sleep state using interrupts and then transmits the required information and goes back to its deep sleep state. During this state the current consumption increases to **18mA** for about 2 seconds before the device goes back to sleep. out of these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

5.2 PCB Design

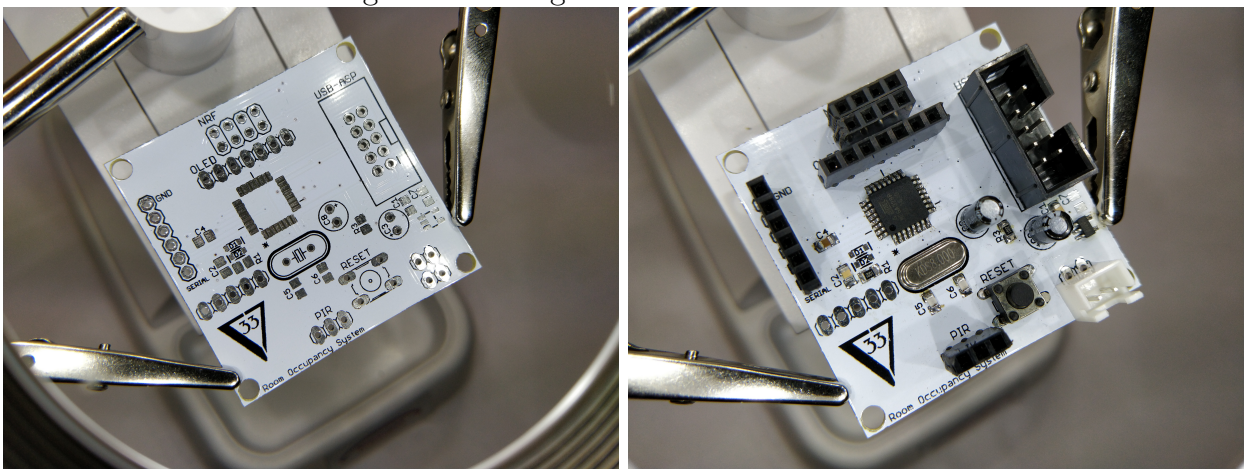
The board has the following features

- 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)

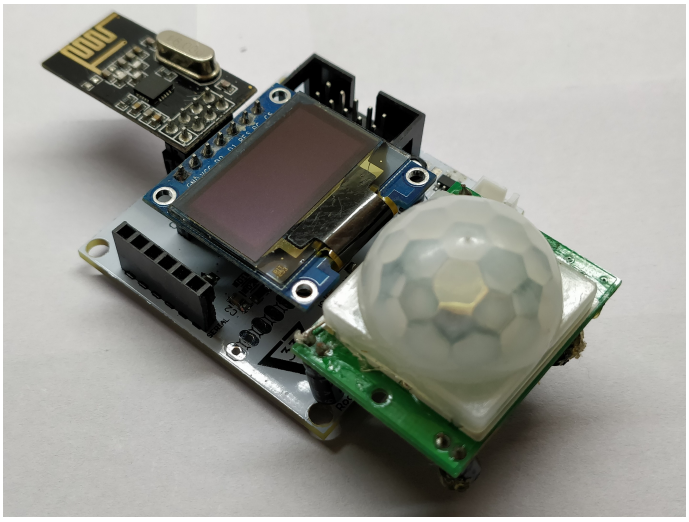
We used the smallest footprint components which was practical to hand assemble so that we could make the device smaller. All the resistors and capacitors are of **0805** package, ATmega is **TQFP** package, the voltage regulator is **SOT-5** package.
We needed to make multiple such boards and they needed 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)[4].



Eagle CAD Design and Manufactured PCB



Populating the PCB

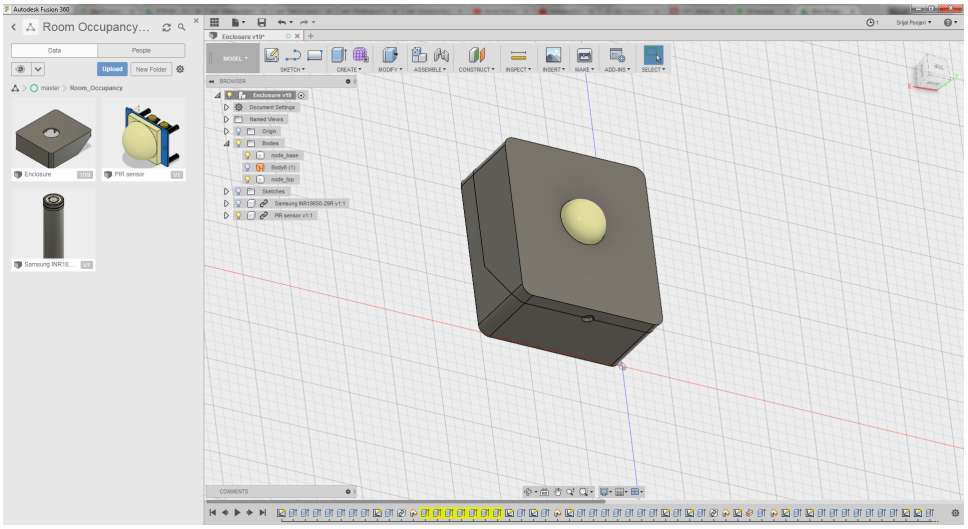


Board with all the components

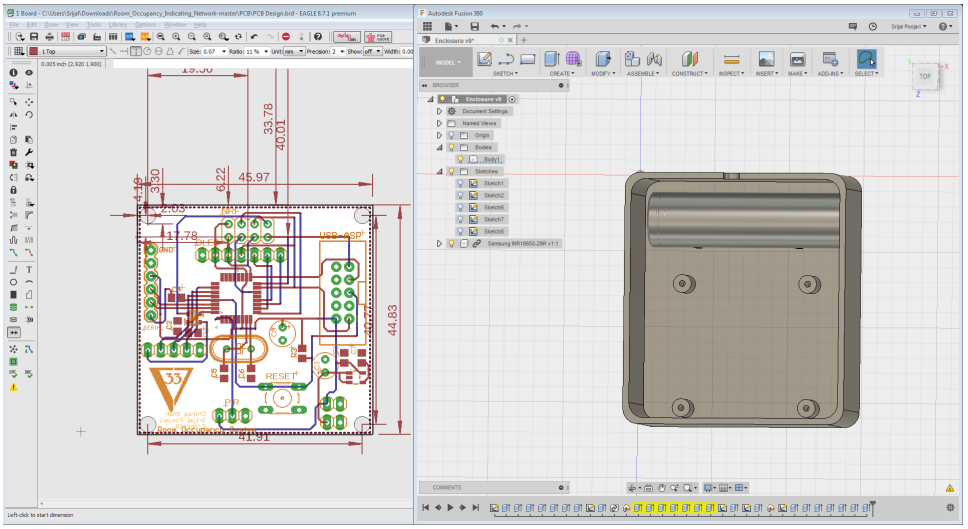
5.3 3D Modelling and Print

The product enslosure has been 3D printed for a neat and professional look. The modelling was done on Autodesk Fusion 360.

We designed the 3D models by carefully taking the PCB, RF module and battery into consideration and tried to fit it a small form-factor. The battery is placed under the radio module by slightly raising the PCB standoffs to achieve a much shorter enclosure length than the slight increase in thickness. Pre-made models of the battery and PIR sensor was found on grabcad.com[3], which made the modelling process a bit easier.



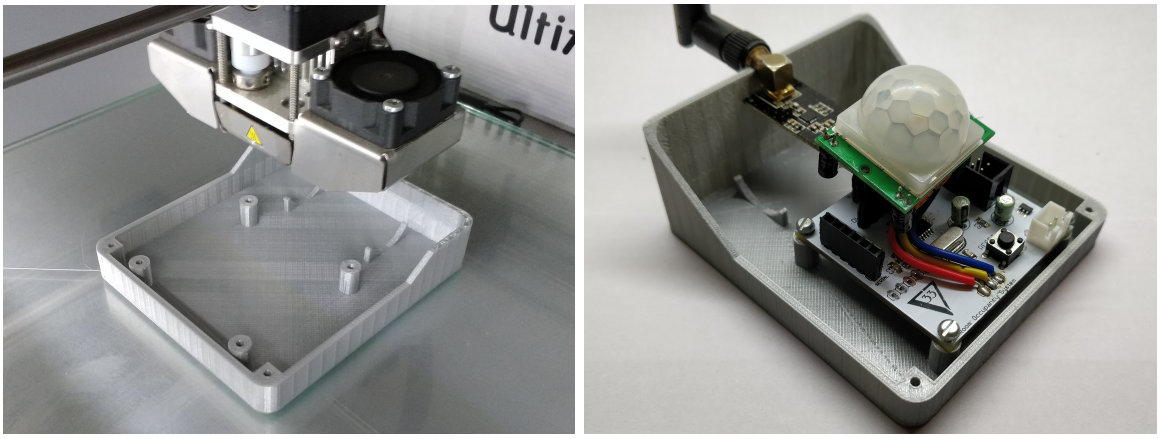
Designing in fusion 360



Modelling the enclosure as per the PCB dimensions

After the design process, the model was exported as .stl files and sent to the printer for 3D printing. The 3D printer used was the Ultimaker 2+ recently purchased and made available by the Institute. Material used is Polylactic Acid (PLA). The first print of the base half 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.

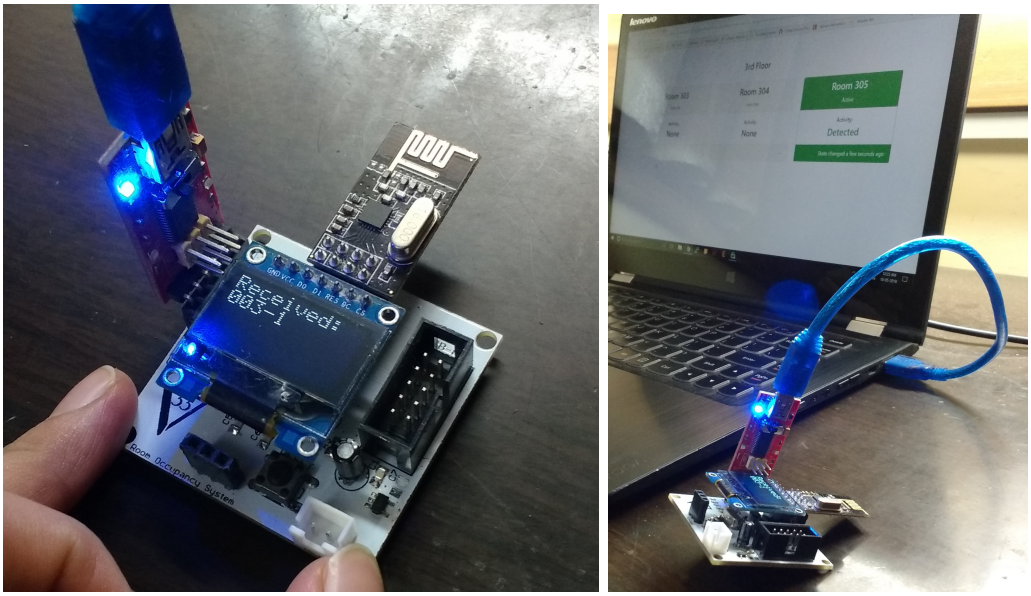
The end result was a neat and aesthetically pleasing which gave the sensor nodes a product-like professional look.



Printing and assembling the device

5.4 Interface with Master Node and Web-GUI

All packets (sensor information) 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 Master Node connected to a Computer via a USB to Serial Converter.

To handle this serial communication, 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). The Pyserial Python module is used for serial interfacing. The Python Flask module is used for creating the webserver.

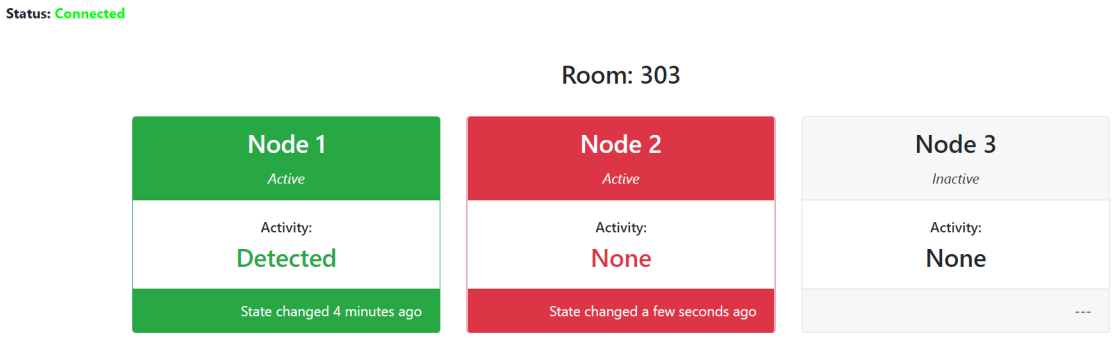
The client side programming is done using Javascript and HTML. For dynamic and high speed update of the GUI, socketIO module is used on Javascript(client) and Python(webserver).

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[5], the sensor information is displayed on top of the map of the room itself, which

makes it incredibly easy to interpret. This could be easily done with our solution as well, provided it is implemented on such a scale.

Sensor information for a complete floor, Occupeye[5]

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 various possible GUI states.

Chapter 6

Software

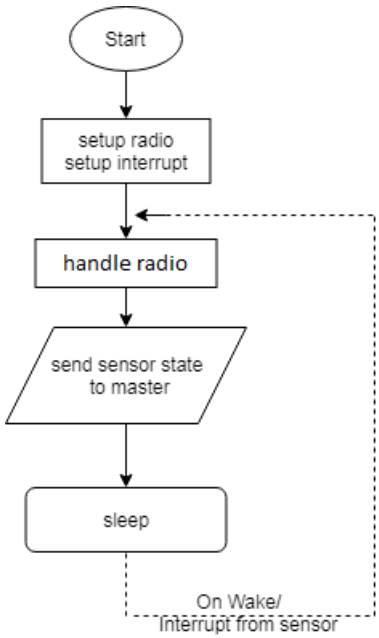
All the programs developed, including the Arduino codes, Python script for master interface and template for Web GUI are available on our GitHub repository[7].

6.1 Master and Node Codes

The codes for the master and nodes are written and implemented using Arduino as the networking libraries for the NRF24 are well developed and documented for Arduino based systems.

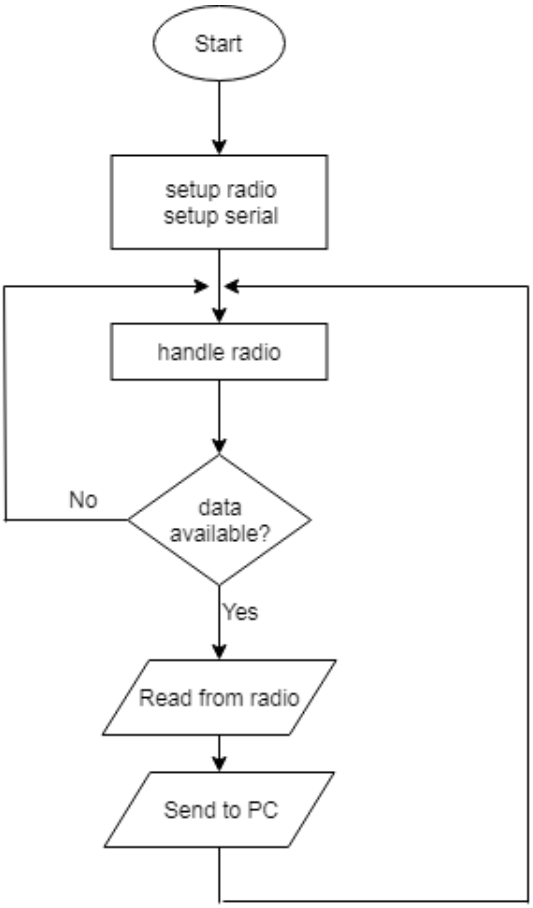
The program flow is can be depicted by the simple flowcharts as follows.

6.1.1 Node



Program flow for a sensor node

6.1.2 Master



Program flow for a the master node

Chapter 7

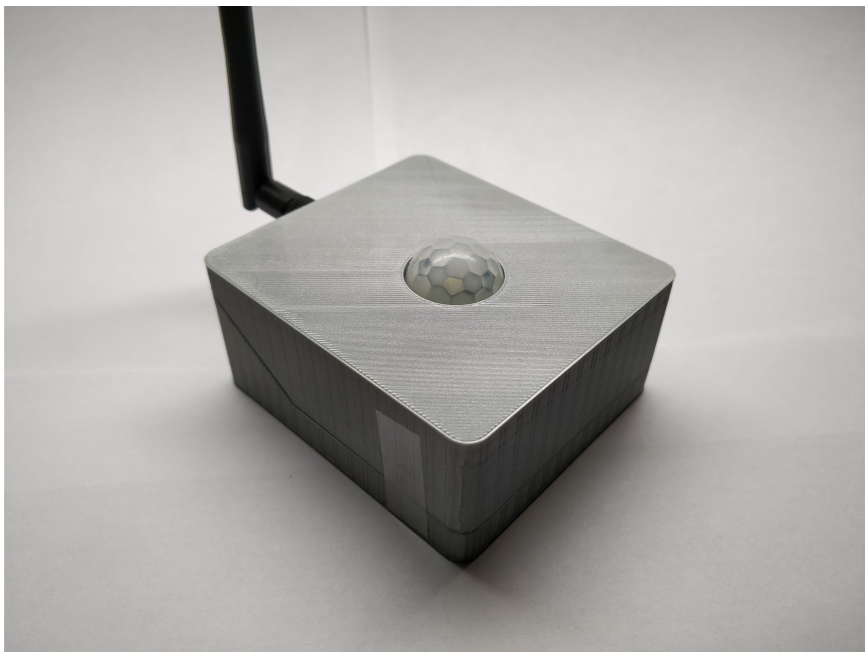
Results

7.1 Breadboard Testing

The intial testing aimed to completely lay out the network configuration as the sending the sensor data was the relatively easy part. The USB connector attached to the breadboard would supply power to all of the devices with the native 5V and regulated 3V3 (using the LM1117-3V3 Regulator) supply voltages with a wall charging adapter or even a power bank which increased portability and made testing easier. Various networking libraries for the NRF24 were tested out. We also conducted range tests with varying transmission bit rates and signal strength and found that the NRF24 satisfied our requirements.

This was the foundation for our project as these results confirmed our concept and so we moved on to making the PCBs while simultaneously conducting further tests and code development on these breadboard prototypes.

7.2 Final Device



Final Assembled Device

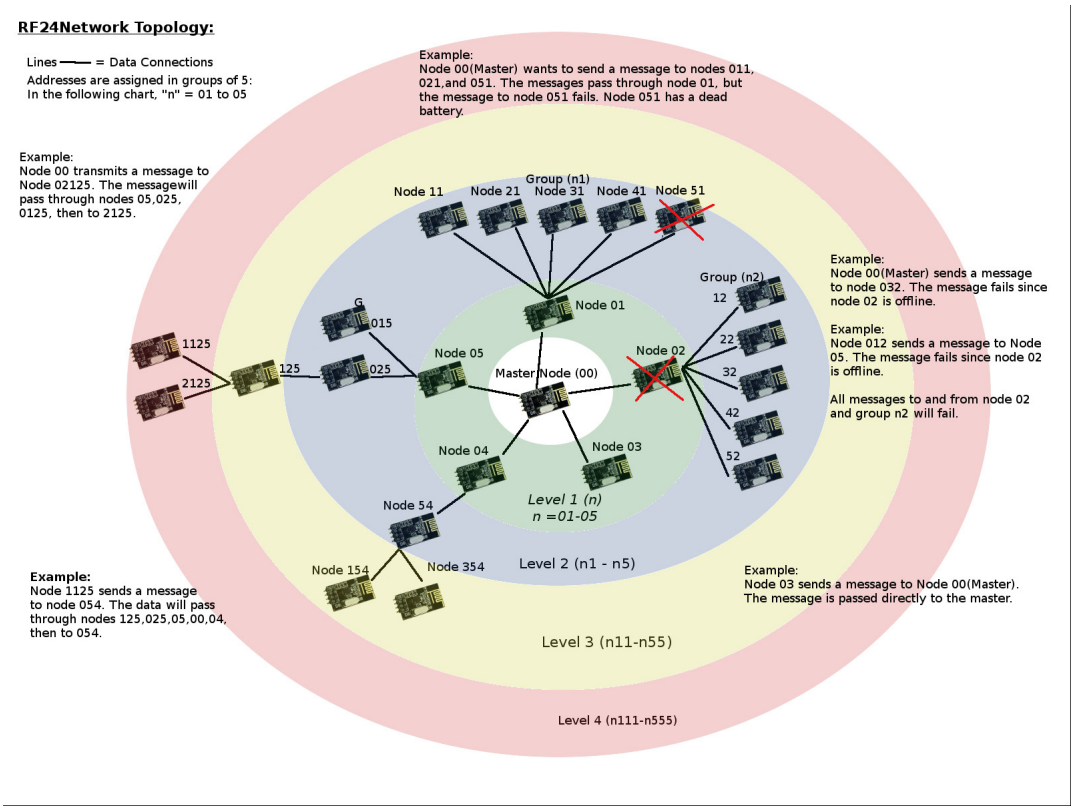
All the end nodes stay in a deep sleep state. They wake up only when a change of state in the PIR sensor is detected or when the WDT overflows.

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.

The current of the end nodes is 18mA when transmitting but it is only 80μA when it is in standby mode. This will significantly improve the battery life of the devices.

All the devices are configured in a tree network configuration to increase the range of the network. Each NRF24 Module can support upto 6 slave devices simultaneously, so the network has to be designed appropriately while setting it up in any real environment.



Tree network topology

7.3 Cost per Device

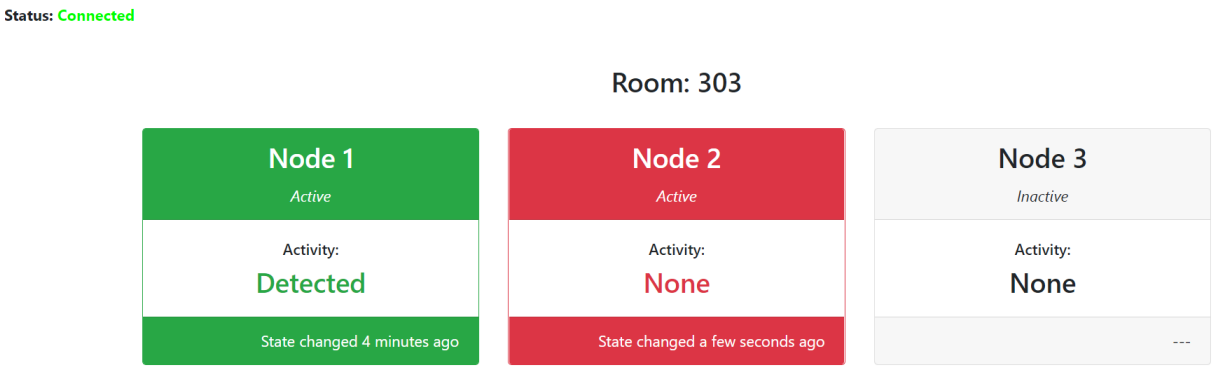
The final cost per device is about 900Rs. This is subject to change as per the source of the components. Also this is the cost of building the prototype. Large volume production cost will be much lower. The cost distribution per device is as follows

Part	Cost
NRF24l01+ radio transceiver module	250
PCB	120
ATmega328P-PU	120
PIR Sensor	100
Voltage Regulator	40
Components	20
Li-ion Battery	200
3D Printed Enclosure	50
Total Cost	**900**

This is much lower than other such sensing systems such as Workscape[6] and Occupeye[5], which typically have a pricing of 15$ per device, per month; as stated earlier.

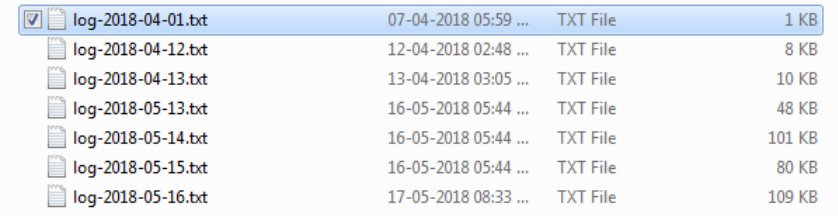
7.4 Web GUI

The occupancy status of each room can be seen on the webpage created.



Web GUI

Real time occupancy status can be seen on any web browser. The colours indicate state of each node. Green and Red indicate that the node is ON and actively sensing, and their sensor state is represented. The time since the current state has occured is also shown. Greyed out nodes are not active and are either switched OFF or their connection is broken.



Log files generated

log-2018-05-13.txt

```
64  Room: 303, Node number: 2, [2, '303', 1, 1526218052.169, datetime.datetime(2018, 5, 13, 18, 57, 32, 169000)]
65  Room: 303, Node number: 2, [2, '303', 0, 1526218054.953, datetime.datetime(2018, 5, 13, 18, 57, 34, 953000)]
66  Room: 303, Node number: 3, [3, '303', 0, 1526218055.322, datetime.datetime(2018, 5, 13, 18, 57, 35, 322000)]
67  Room: 303, Node number: 3, [3, '303', 1, 1526218060.57, datetime.datetime(2018, 5, 13, 18, 57, 40, 570000)]
68  Room: 303, Node number: 2, [2, '303', 1, 1526218062.13, datetime.datetime(2018, 5, 13, 18, 57, 42, 130000)]
69  Room: 303, Node number: 3, [3, '303', 0, 1526218064.445, datetime.datetime(2018, 5, 13, 18, 57, 44, 445000)]
70  Room: 303, Node number: 2, [2, '303', 0, 1526218064.915, datetime.datetime(2018, 5, 13, 18, 57, 44, 915000)]
71  Room: 303, Node number: 2, [2, '303', 1, 1526218071.02, datetime.datetime(2018, 5, 13, 18, 57, 51, 20000)]
72  Room: 303, Node number: 3, [3, '303', 1, 1526218072.855, datetime.datetime(2018, 5, 13, 18, 57, 52, 855000)]
73  Room: 303, Node number: 2, [2, '303', 0, 1526218073.81, datetime.datetime(2018, 5, 13, 18, 57, 53, 810000)]
74  Room: 303, Node number: 3, [3, '303', 0, 1526218076.711, datetime.datetime(2018, 5, 13, 18, 57, 56, 711000)]
75  Room: 303, Node number: 2, [2, '303', 1, 1526218079.247, datetime.datetime(2018, 5, 13, 18, 57, 59, 247000)]
76  Room: 303, Node number: 2, [2, '303', 0, 1526218082.045, datetime.datetime(2018, 5, 13, 18, 58, 2, 45000)]
77  Room: 303, Node number: 3, [3, '303', 1, 1526218083.356, datetime.datetime(2018, 5, 13, 18, 58, 3, 356000)]
78  Room: 303, Node number: 3, [3, '303', 0, 1526218087.21, datetime.datetime(2018, 5, 13, 18, 58, 7, 210000)]
79  Room: 303, Node number: 2, [2, '303', 1, 1526218087.912, datetime.datetime(2018, 5, 13, 18, 58, 7, 912000)]
80  Room: 303, Node number: 2, [2, '303', 0, 1526218090.705, datetime.datetime(2018, 5, 13, 18, 58, 10, 705000)]
81  Room: 303, Node number: 3, [3, '303', 1, 1526218110.432, datetime.datetime(2018, 5, 13, 18, 58, 30, 432000)]
82  Room: 303, Node number: 3, [3, '303', 0, 1526218114.313, datetime.datetime(2018, 5, 13, 18, 58, 34, 313000)]
83  Room: 303, Node number: 3, [3, '303', 1, 1526218124.35, datetime.datetime(2018, 5, 13, 18, 58, 44, 350000)]
84  Room: 303, Node number: 3, [3, '303', 0, 1526218128.224, datetime.datetime(2018, 5, 13, 18, 58, 48, 224000)]
85  Room: 303, Node number: 3, [3, '303', 1, 1526218143.426, datetime.datetime(2018, 5, 13, 18, 59, 3, 426000)]
86  Room: 303, Node number: 3, [3, '303', 0, 1526218147.306, datetime.datetime(2018, 5, 13, 18, 59, 7, 306000)]
87  Room: 303, Node number: 2, [2, '303', 1, 1526218153.153, datetime.datetime(2018, 5, 13, 18, 59, 13, 153000)]
88  Room: 303, Node number: 3, [3, '303', 1, 1526218153.306, datetime.datetime(2018, 5, 13, 18, 59, 13, 306000)]
89  Room: 303, Node number: 2, [2, '303', 0, 1526218155.942, datetime.datetime(2018, 5, 13, 18, 59, 15, 942000)]
90  Room: 303, Node number: 3, [3, '303', 0, 1526218157.19, datetime.datetime(2018, 5, 13, 18, 59, 17, 190000)]
91  Room: 303, Node number: 2, [2, '303', 1, 1526218161.968, datetime.datetime(2018, 5, 13, 18, 59, 21, 968000)]
92  Room: 303, Node number: 2, [2, '303', 0, 1526218164.795, datetime.datetime(2018, 5, 13, 18, 59, 24, 795000)]
93  Room: 303, Node number: 3, [3, '303', 1, 1526218165.698, datetime.datetime(2018, 5, 13, 18, 59, 25, 698000)]
```

A sample log file

The status, location and a precise timestamp of each node gets stored in a log file which can be used to debug or check past sensor states. New log files are generated every day and are organised by date.

Chapter 8

Next Steps

8.1 Testing the Battery Life of the Devices

The devices currently have a 3000mAH li-ion battery. The battery life can only be truly measured by running the devices for a long period of time in a real life environment. We plan to properly install the devices in a real life setting and monitor the battery voltage change in the devices over a period of time. This will give us a true idea of the battery life of the devices.

8.2 Testing with increased number of nodes

Increasing the number of networked devices poses many challenges such as handling the require 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 and go further from there on.

8.3 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.

8.4 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 9

Conclusion

We were able to create a wireless tree network of low powered battery operated devices which 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.

The cost per device (900 INR) is much less than the current available commercial alternatives such as Workscape[6] and Occupeye[5], which typically are priced at 15$ per device per month, as stated earlier. Hence the goal of making a low cost device has been achieved.

The software can be greatly improved further to make the system robust and easily scalable, but the system at its current state is a lays a strong foundation for future development.

There are also a few hardware changes that could be made such as rearranging the components on the PCB for a smaller form-factor and utilizing the RTC port for such as alarm interrupt for sending heartbeat packets, and get more power savings by a complete shutdown after office hours, holidays etc.

Summarizing, we conclude that the Minproject has been completed as per our expectations and the targets set by the Hypothesis have been met.

Bibliography

[1] Sparkfun.com, 'nRF24L01+ Transceiver Hookup Guide,' [Online]. Available:https://learn.sparkfun.com/tutorials/nrf24l01-transceiver-hookup-guide [Accessed: 10-Feb-2018]

[2] geekstips.com, 'Internet of Things Project, Communication between ESP8266 modules,' [Online]. Available: https://www.geekstips.com/two-esp8266-communication-talk-each-other/ [Accessed: 18-Feb-2018]

[3] GrabCAD, 'Free cloud-based collaboration solutionto view and share CAD files,' [Online]. Available:https://grabcad.com/ [Accessed: 20-Mar-2018]

[4] PCBWay 'PCB Manufacturing service' [Online]. Available:https://pcbway.com/ [Accessed: 12-Mar-2018]

[5] Occupeye 'OccupEye workspace utilisation sensors' [Online]. Available:https://occupeye.com/ [Accessed: 11-Feb-2018]

[6] Workscape 'Workscape smart sensors' [Online]. Available:https://workscape.io/ [Accessed: 11-Feb-2018]

[7] ROIS on Github, 'Room Occupancy Indicating Network,' [Online]. Available:https://github.com/Nabla33/Room_Occupancy_Indicating_Network [Accessed: 18-May-2018]