Boston University
Electrical & Computer Engineering
EC464 Capstone Senior Design Project

User's Manual

Submitted to:

Prof. Janusz Konrad
8 St. Mary's St Room 443
Boston MA 02215
(617) 353-1246
jkonrad@bu.edu

Prof. Prakash Ishwar
8 St. Mary's St, Room 440
Boston MA 02215
(617) 358-3499
pi@bu.edu

By
Team #11
OccuSense

Alex Bleda ableda@bu.edu
Miguel Cepeda mccepeda@bu.edu
Krystal Kallarackal krystalk@bu.edu
Artem Bidnichenko artbidn@bu.edu
Yerkebulan Nurkatov nyz@bu.edu

Submitted: 4/7/17

OccuSense

Table of Contents

Executive Summary

Current HVAC (Heating Ventilation & Air Conditioning) systems account for 20% to 35% of a building's energy usage. However, they are not responsive or adjustable to the number of people in a room. Air ventilation, by itself, accounts for 7-10% of a building's energy consumption leaving a large opportunity to save on commercial building energy costs. According to the National Grid, efficient HVAC systems are able to save building energy consumption by up to 10%. We aim to create a real-time, accurate and reliable sensor system that determines the occupancy of a room. This system will utilize thermal sensing technology combined with a detection algorithm to track the number of people entering and exiting a doorway. The occupancy data will be pushed and stored to a database housed in a website where the client will be able to view real-time and historical data for a specified room. Finally, this data will be utilized to adjust the room's ventilation system to cater to the number of people occupying the room, lowering building energy consumption and costs.

1 Introduction

The Department of Energy has identified occupancy sensing as a crucial technology in optimizing HVAC and lighting systems within commercial buildings. Most highly accurate occupancy sensing systems to date utilize cameras and, in turn, violate the privacy of people in many different scenarios: bathrooms, conference rooms, classrooms. The goal of this project is to develop an occupancy sensing system that protects the privacy of the people being surveyed and to be able to utilize this system to minimize the energy consumption and costs for the HVAC system of a commercial building.

There is a large opportunity to save on commercial building energy consumption by implementing more efficient or “smarter” HVAC systems. Although Boston University already utilizes occupancy sensing to help monitor and manipulate lighting and HVAC systems these sensors are not sufficient. Most sensors only differentiate between an empty room and an occupied room which translates to only two settings for lighting and HVAC systems for commercial buildings. The purpose of the OccuSense room occupancy sensing system is to evaluate the occupancy of a room to four different levels: empty, low, medium, and high occupancy. Being able to tell the occupancy of a room to this level will cause HVAC systems to be more efficient because the amount of energy consumption will be based on the amount of people in the room and there will be less wasted energy on rooms that have little to no occupancy. By lowering energy consumption the commercial building energy costs will also go down.

The OccuSense system offers a solution to occupancy sensing by creating an entrance/exit counter that keeps track of people entering and leaving a room while also preserving the privacy of anyone it detects. The system is based on the infrared Melexis 90621 low resolution 16x4 pixel thermal imager. This low thermal resolution meets the privacy and cost requirements for the project and achieves impressive accuracy results. The sensor is connected to an arduino for data collection and to a Raspberry Pi for data processing and networking. The OccuSense system also provides a nice user interface hosted in a firebase web server to monitor, customize and control basic operations. The firebase server also includes a database with historical data about room usage that provides clear and useful insights for the prediction of future room utilizations which could translate to further energy cost savings. It is important to note that the project we are presenting here is a prototype for what would signify a much more energy efficient device with the same sensor technology. If we were to deploy this commercially we would plan to develop a custom PCB without having to require the Raspberry Pi or the arduino.

In the following sections of the user’s manual we will cover in detail the overall functionality of the system, including system block diagrams and images of our existing prototype, the installation and setup of the system, information about the operational modes and detailed technical background for every component of the system. We will conclude the document with a cost breakdown of the project and appendices to help visualize some of the technical characteristics of the system explained throughout the document.

2 System Overview and Installation

2.1 Overview block diagram

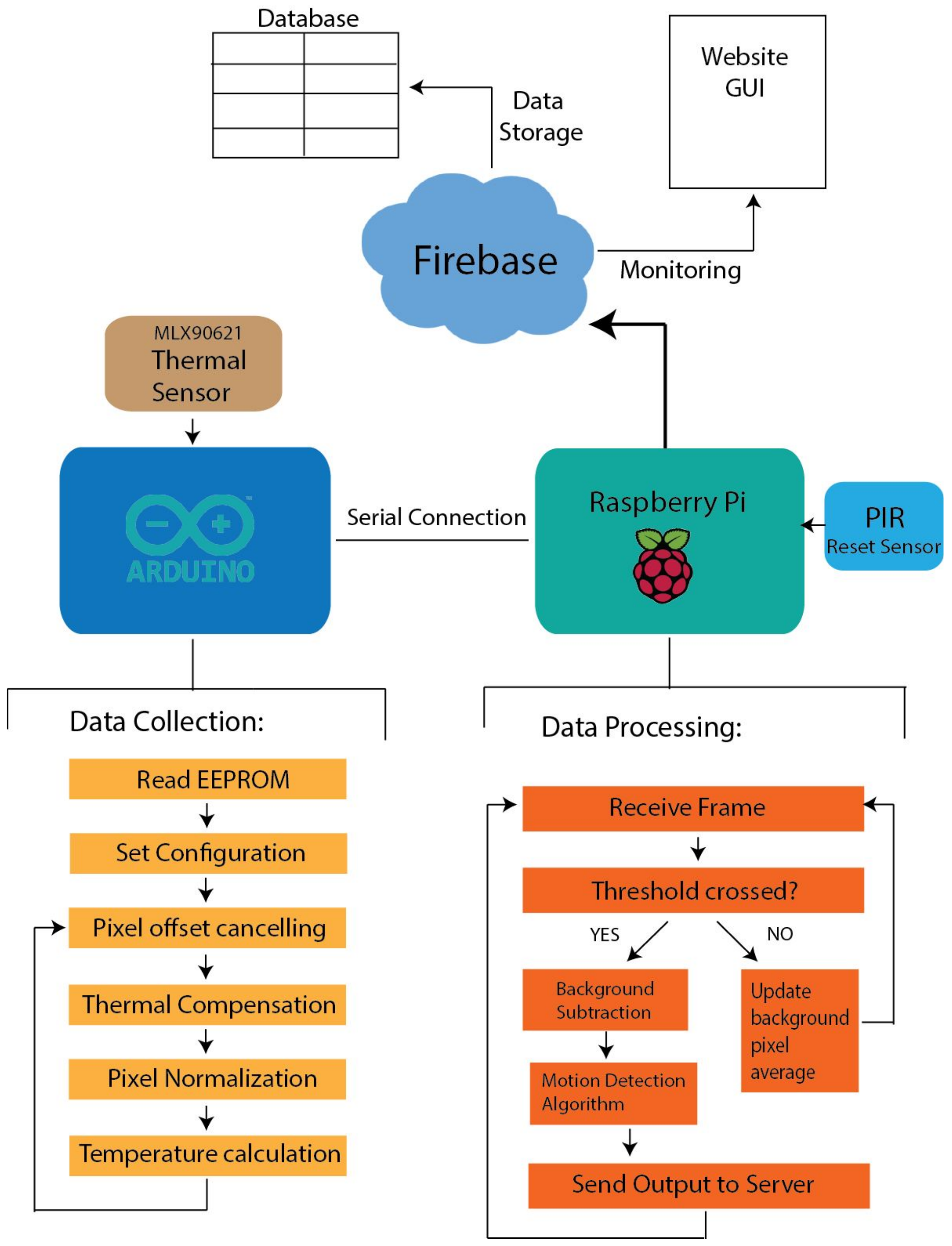


Figure 1: Detailed overall diagram of the functionality of each independent sensor system. The arduino performs data collection from the thermal sensor and the Raspberry Pi processes and sends the result to the server.

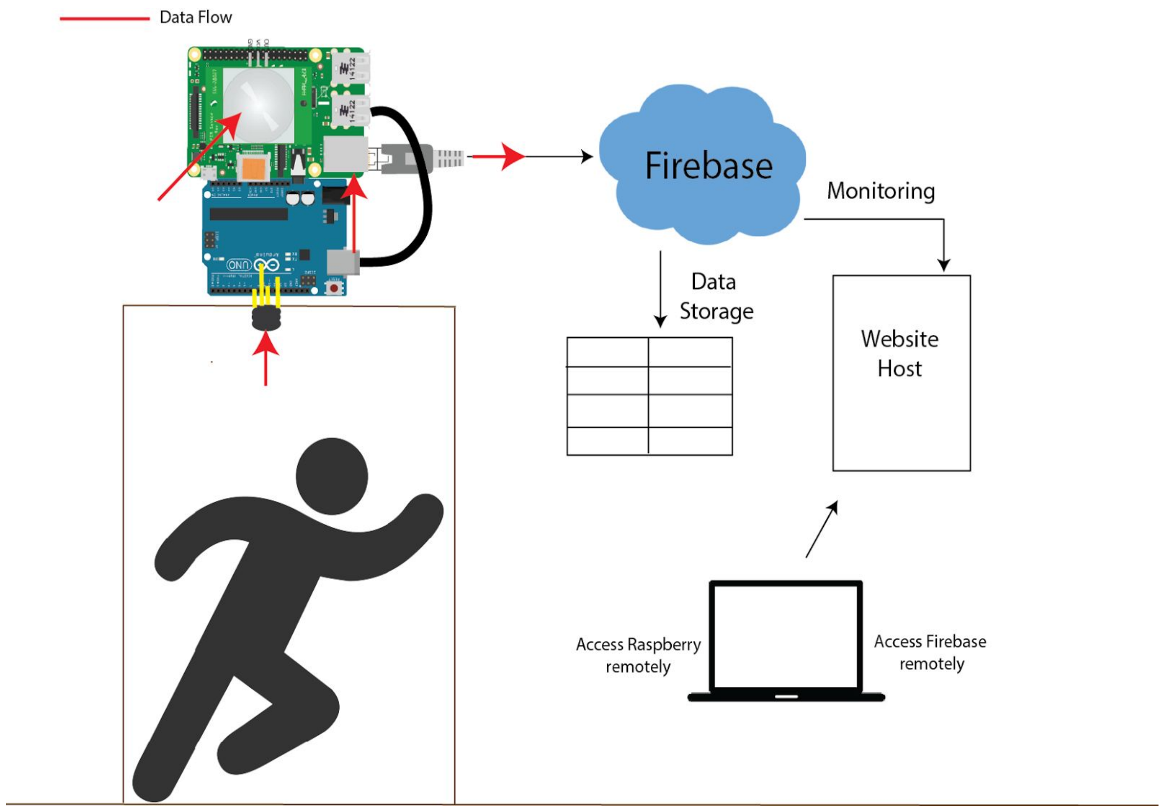


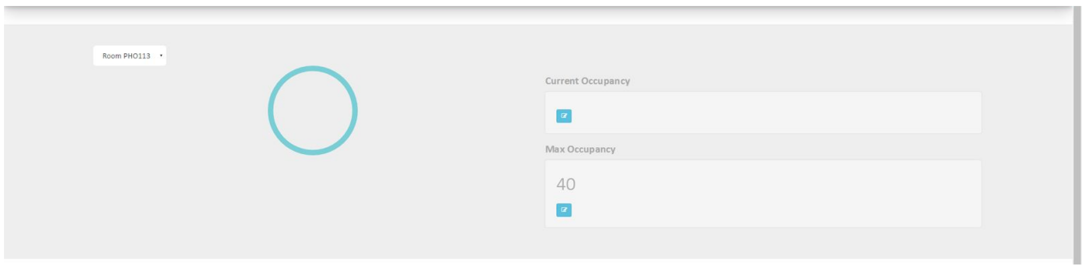
Figure 2: This is the setup used during functional testing which helps visualize the data flow through all the system components. Each sensor system will work independently of each other and combined by the web server.

2.2 User Interface

The web interface that is utilized by the OccuSense system is split into two parts: the database and web user interface. Both of these components are hosted on Google's free web and phone application platform Firebase, that includes a substantial library of infrastructure template and tools.

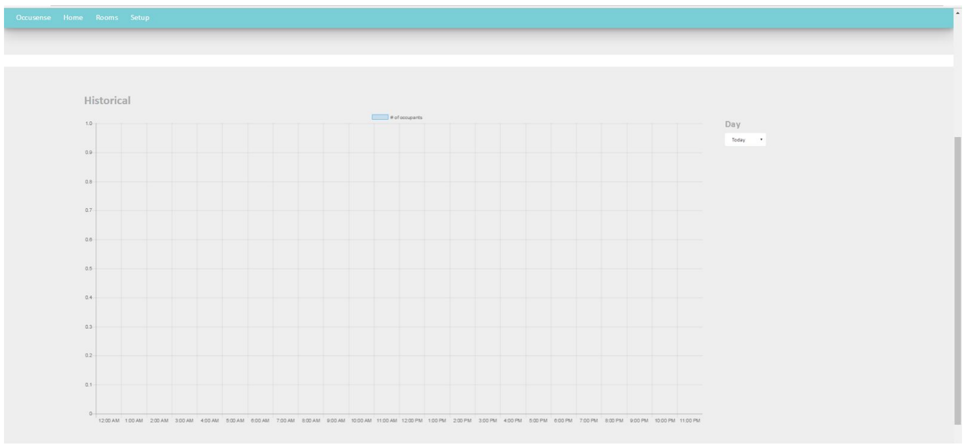
The Firebase web server will house the web user interface for the OccuSense sensor system. Here, the client will be able to login and access the real-time and historical occupancy data of a specified room. This web application is housed on the Firebase web server and will draw the real-time (Figure 3) and historical data (Figure 4) directly from the database also housed on the Firebase web server. From the user interface, the user will be able to add and delete registered sensor systems and toggle between the different occupancy data sets of active sensors.

The Firebase database houses the real-time data collected by the physical OccuSense sensor system. This data is then pushed from the database to the OccuSense website where the real-time and historical occupancy will be updated in the occupancy graphics pictured below. The user does not directly interact with the database, but can start and stop data collection through the "Setup" page on the OccuSense website.



Max Occupancy: 40

Figure 3: Real-time occupancy graphic displayed on the "Rooms" page of the OccuSense website



Day: Today

Figure 4: Historical occupancy graphic displayed on the "Rooms" page of the OccuSense website

2.3 Physical Description

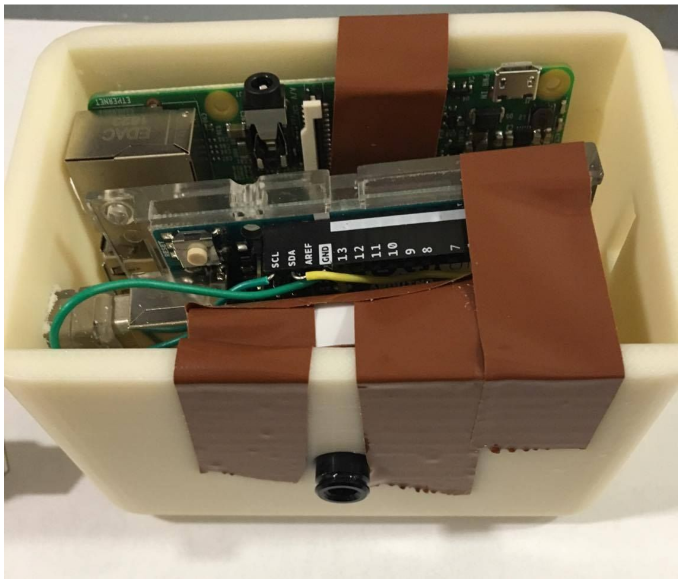


Figure 5: Prototype of OccuSense complete system

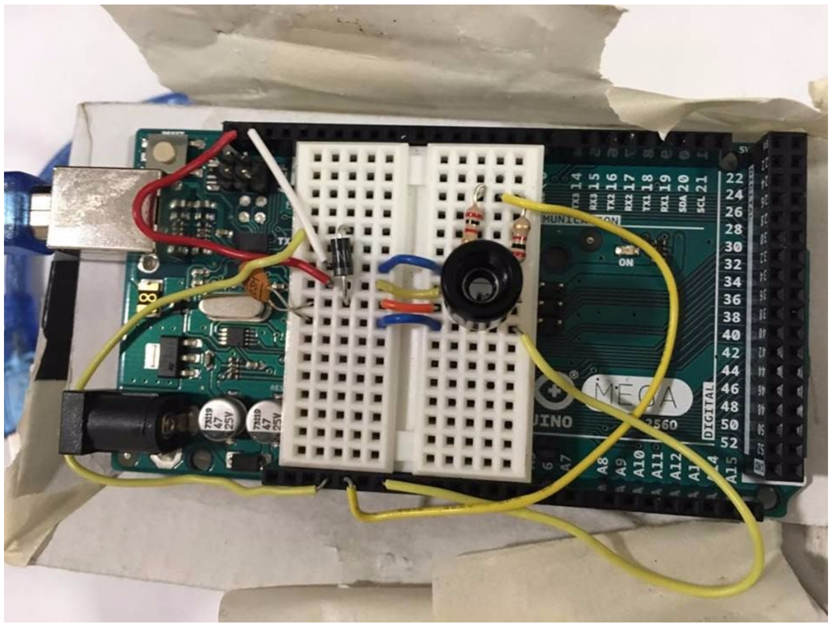


Figure 6: Arduino Mega with MLX90621 sensor circuit used during functional testing

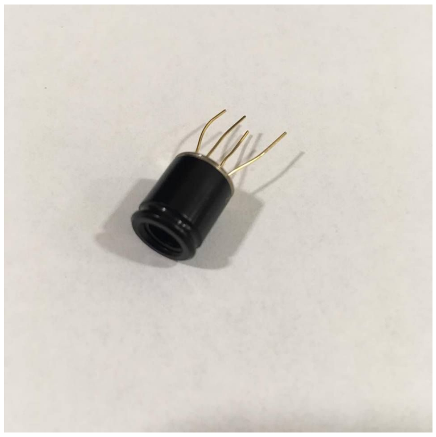


Figure 7: MLX90621 Infrared Sensor

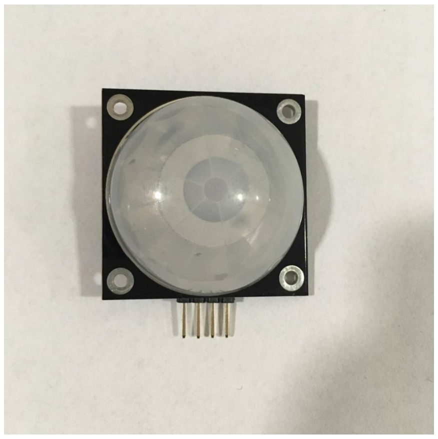


Figure 8: Wide angle PIR sensor

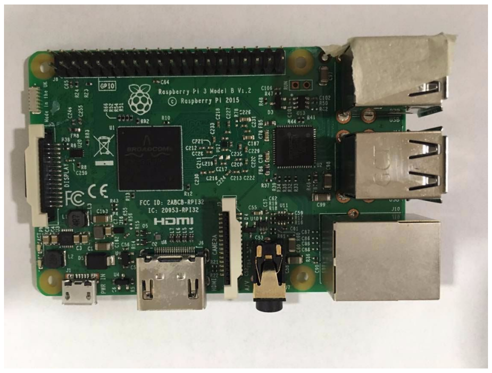


Figure 9: Raspberry PI 3

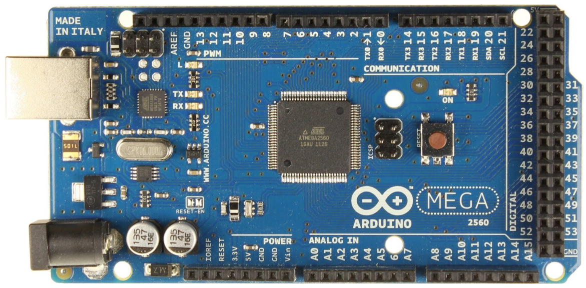


Figure 10: Arduino Mega

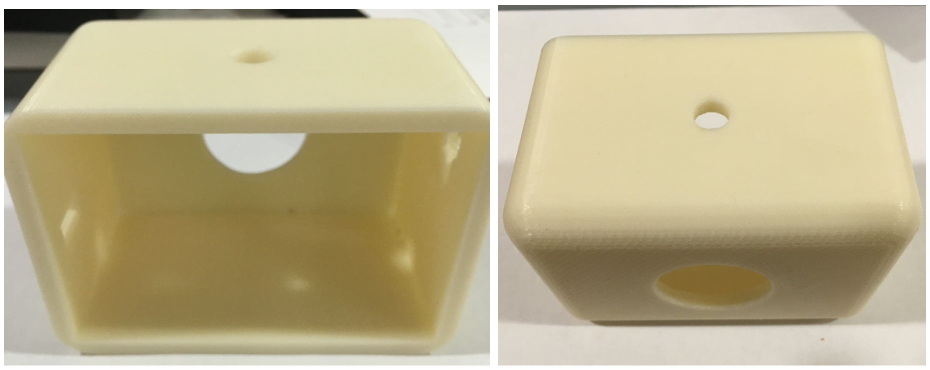


Figure 11: Plastic sensor system case

2.4 Installation and Setup

2.4.1 Installation

The installation of the OccuSense system is designed to be as simple as possible. Each device or as we call it "sensor system" will work independently of each other and will have the same functionality. Thus the installation part only requires mounting the sensor system properly. As seen in Figure 4, the main device comes packaged in a 3d printed case that will be easily mounted into the wall above the frame of the door with mounting tape. The only wire required is the power cord which has to be connected to the Raspberry Pi once the device is mounted. Upon powering up the sensor system will wait to be activated by the server via the user interface website, explained below.

The best place to put the device is 2.25m above the ground, for a 0.90m door width. For a wider door, rise the device 7cm for every 10cm of extra door width. Opposite applies for narrower doors. +/- 5cm of horizontal or vertical error in device placement is tolerable and would not affect its performance.

2.4.2 Setup

2.4.2.1 Through Terminal

Begin by plugging in the Occusense sensor system to the power source of your choosing (outlet, battery, etc.). Once the system is powered on you must SSH into the Raspberry Pi (RPi) using the IP address for the RPi using the following command:

```
ssh pi@XXX.XX.XX.X
```

You will then be prompted for a username and password if you have set one up. Once you are granted access you will have to access the executable programs

Directory: /home/pi/OccuSense

Once you are in the directory you have a choice of running the main script that runs the OccuSense sensor system in conjunction with the web user interface with by running main.py.

```
python main.py
```

Once the main.py script is running you many begin interacting with the OccuSense sensor system using the web user interface. If you do not wish to run through the web user interface and only want to count or collect data separately you need to run count_people.py or collect_data.py depending on your needs

```
python count_people.py/collect_data.py
```

In order to parse the data collected by the collect_data.py program there are a couple options. To produce a video of the heat map of the .txt file you may run the make_video.py program. To produce a histogram containing average temperature/frame we also provide the analyze.py program. Both of these functions are an extension of the collect_data.py function and works with the .txt file that is produced from it.

2.4.2.2 Through UI

The algorithm is set up and started automatically when the device is plugged in. The default parameters include the 12 frames per second data collection rate as well as a given sensor ID that should be changed when configuring the new device to be part of a new room. Use the OccuSense website: https://occuserver.firebaseapp.com/.

Upon opening the OccuSense website you will be prompted to enter login information to access a personal OccuSense sensor system and data. If you are a new user, you will be able to set up account information (username and password) from the homepage. Upon login, you will notice that are three tabs available for the user: Home, Rooms, and Setup. The homepage contains serves as a starting point for the user. From the homepage the "Rooms" and "Setup" page can both be accessed. In order to add devices to the web user interface you must navigate to the Setup page.

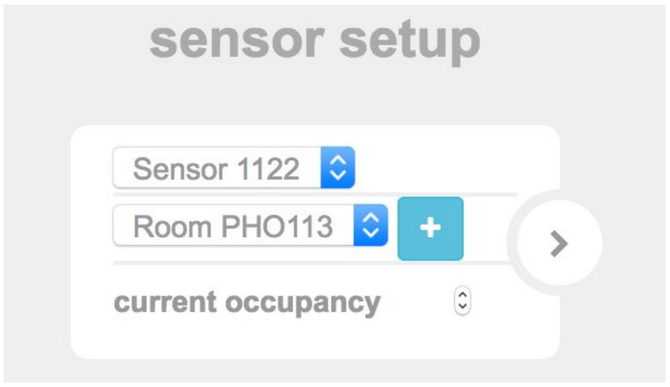


Figure 12: Setup window sample

From the "Setup" page, registered Occupancy systems will appear in the first drop down menu. Select the sensor system that you wish to track from the Rooms page. After selecting the target sensor, enter the name of the room that the sensor will track the occupancy for. After assigning the correct configurations for the sensor press the "+" button and the data collected from that sensor will now be shown on the "Rooms" page. Registered sensors that are meant to be collect data for the same room should be given the same room assignment. If two sensors are assigned identical rooms their data will automatically be cumulatively added together through the web server. Do not add the same room twice under the "Setup" page in order to avoid having inaccurate data collection.

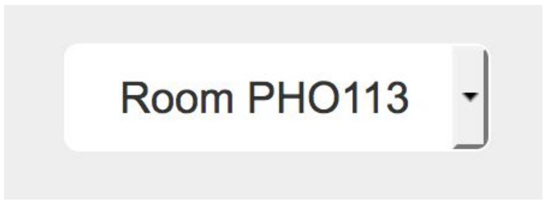


Figure 13: Button to change the room to view data from.

From “Setup” tab you will also be able to start and stop data collection of a specified sensor by click the “Stop Data Collection” or “Start Data Collection” button.

The “Rooms” page contains the real-time and historical occupancy for registered sensor systems. To begin accurately tracking the occupancy of the room, you must first enter the maximum occupancy into the designated field, “Maximum Occupancy”. Also, on this page you can control what sensor occupancy data sets show up on the graphs. You can delete sensors entirely from the “Rooms” page by using the delete function found at the bottom of the page. Finally, the historical data is organized by day so you can toggle through the different days that occupancy data was collected by using the drop down menu to the right of the historical occupancy data graph.

Figure 14: Tabs of the OccuSense website

3 Operation of the Project

3.1 Operating Mode 1: Normal Operation

The OccuSense system normal operation mode is defined as the main mode of operation in which the sensor is actively counting people. It is constantly collecting and processing data and updating in real time the occupancy of every room. Since the system has various components, this section describes how each component in every sensor system operates in the normal operation mode, including notes on how to use the web server interface during the normal operation mode.

3.1.1 Melexis Thermal Sensor

As described in section 2.4 the thermal sensor will always be working in the default parameters unless specifically changed. This means that the thermal sensor will be operating at 12 frames per second with a supply voltage of 2.6V and drawing 9mA. When the device is turned off, the sensor is also turned off. The user should not need to interact with the thermal sensor directly. The data collection technique the system uses to retrieve the data from the sensor is explained more in detail in Section 4.

3.1.2 Raspberry Pi

The Raspberry is the central hub of the sensor system. It hosts the main algorithm that processes and detects people, the PIR sensor program and a simple communication program to the server to push detection results and receive commands, further explained in section 4. The Raspberry Pi will be configured to only start this services when the command from the server is received (default), but also to be activated on boot, all through the user interface. Once activated the OccuSense system is designed to not require any maintenance, it will keep updating the room occupancy automatically. If desired however, these services can also be stopped with the server and in case of server problems the Raspberry Pi could be accessed directly through secure shell. We are planning on including the Raspberry IP address and a sensor system characteristics description in the user interface for further improving troubleshooting and customization. The technical aspects of these services are explained in detail in Section 4.

The Raspberry Pi power consumption would probably never go above 2W and is estimated to be around 1.5W. In order to operate, Raspberry Pi needs a micro SD memory card. Total amount of software used for the project is relatively small. Even a 256Mb would be more than enough to run it properly.

3.1.3 PIR Sensor

PIR sensor is mounted in the hole of the case device, facing the room itself. Its main purpose is global resetting. Because the system might accumulate error by misinterpreting some events over long period of time, PIR is needed to set the people count to 0 if detecting no motion for 2-3 consecutive hours. The actual value is also going to be available to be set through the server. The sensor has 180° detection angle and provides up to 30 feet of detection distance.

3.1.4 Web Server

3.1.4.1 Normal Operations

All user-facing settings can be configured from the OccuSense website. Under normal operations, registered OccuSense sensor systems can be added and tracked on the website form the website’s “Setup Page”. From the “Setup” page, the user be able to assign rooms to registered sensor systems.

Room Assignment:

1. Access first drop down menu to select a registered sensor system. Sensor systems will be labeled according to their sensor ID.
2. Access second drop down menu and select which room the sensor will be monitoring.
3. Press "+" button to add room and assigned sensor to the "Rooms" page, Figure 11
4. Real-time and historical occupancy data for this room/sensor system pairing will now appear on the "Rooms" page

Once devices are added from the "Setup" page, their respective occupancy data can be tracked through the "Rooms" page. On the "Rooms" the following settings can be accessed: Maximum Occupancy, Past Historical Data Sets, Sensor Deletion, and Viewing Options for sensor data.

Assign Maximum Occupancy:

1. Access user input box located to the right of the real-time data graphic
2. Enter desired maximum occupancy logistic for the specified room
3. Real-time occupancy graphic will now accurately display the occupancy in the form of percentage

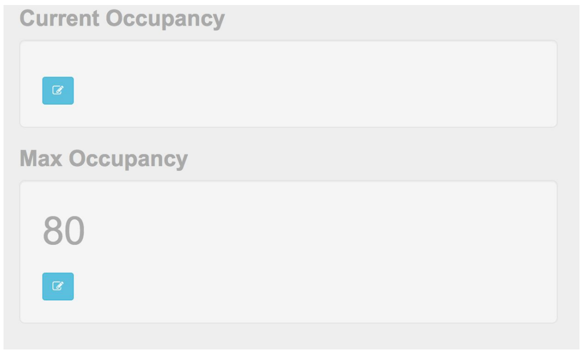


Figure 15: Buttons to set current and max occupancy in the rooms tab

Toggle Between Historical Data Sets:

1. Access drop down menu to the right of the historical occupancy data graphic.
2. Past days that occupancy data was collected will be displayed in the drop down menu
3. Select the intended data set that you want to access
4. The occupancy data for that day will now be viewable on the historical data graphic

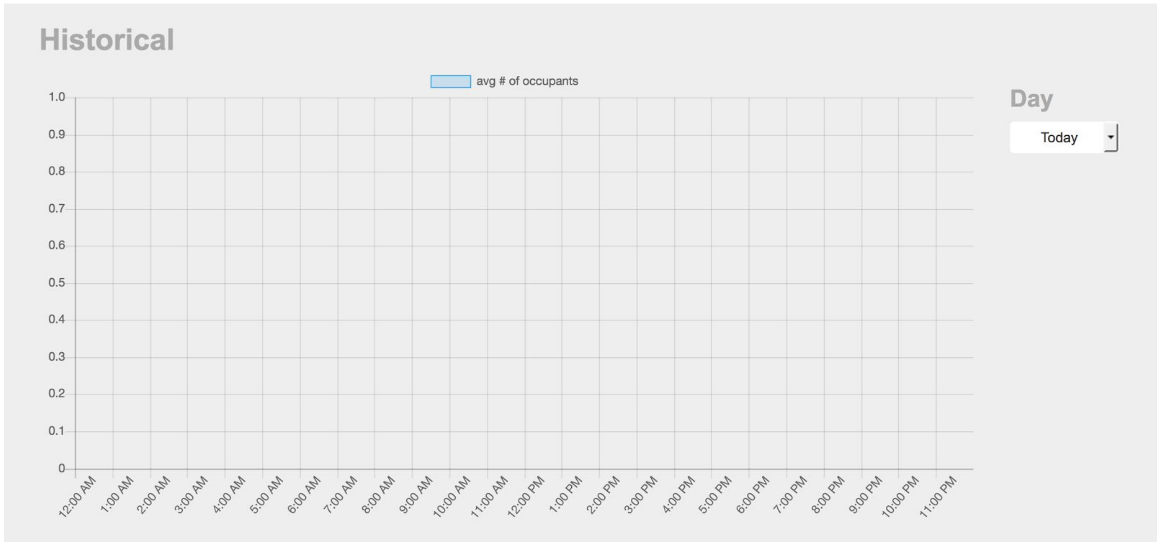


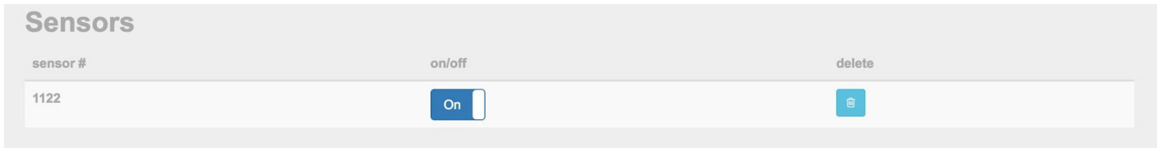
Figure 16: Historical data sample, with button on the right to select day

Viewing Options for Multiple Sensor Systems:

1. Locate "Sensors" subsection on the "Rooms" page
2. Toggle between on and off for the specified sensor
3. This dictates what sensor data sets will be viewable on the occupancy graphics

Sensor Deletion:

1. Locate "Sensors" subsection on the "Rooms" page
2. Click trash icon in the row of the sensor intended for deletion
3. You will prompted to verify that you want to delete sensor, Press OK
4. Access to the deleted sensor will be removed from the "Rooms" page
5. To add the sensor data back to the page, go through "Room Assignment" steps for reading



sensor #	on/off	delete
1122	On	

Figure 16: Sensor section to activate/deactivate and delete sensor

3.1.4.2 Troubleshooting

In the event that the occupancy data sets, stops, graphics are unresponsive or simply want the sensor to be turned off use the following methods. Restarting the device could also fix many of this problems.

1. Navigate to "Setup" page and using dropdown menus select sensor and assigned room of disrupted system
2. Click button labeled "Stop Data Collection"

3. Navigate to “Rooms” page and locate “Sensors” section
4. Validate that the corrupted sensor data is deleted from “Rooms” page
5. If not deleted, press trash icon of the sensor system
6. Read deleted sensor system using the steps from “Room Assignment”
7. If occupancy data is still not being updated correctly check hardware for possible installation flaws.

3.2 Operating Mode 2: Abnormal Operations

The main purpose of the system - approximate the amount of people in the room/building and adjust the HVAC system accordingly. It is not a precise counter. It will make mistakes in counting the exact number of people, and this behaviour is expected. The majority of mistakes might happen when two or more people are present in the door frame at the same time. Second error might happen due to the nature of the sensor we are using. The system will work better for larger differences between background and human temperatures. The system will notice cold coats in winter, that are actually colder than the room temperature, and will work fine a for reasonable room temperatures. But in some limited cases humans are going to be indistinguishable for a sensor. In order to address the issue we implemented a real time adjustment algorithm that reacts to changes in room temperature, thus is flexible, and the system is not supposed to be adjusted to work in different environments (like the outdoors) or changed temperature settings in the room, even though it will adjust to subtle changes in temperature and transitions from winter to summer.

To avoid substantial errors over long period of time, we implemented a reset mechanism, that is triggered by a long term absence of movement in the room. The counter is set back to 0. It may be set to exact time or certain “silence” time. Usually a couple of hours.

There are possible errors that may occur in the Firebase web application that hosts the database and website for the Occusense server system. One possible error may occur when people are detected leaving the room and not accurately detected entering the room. This will lead to a negative occupancy, which is an impossible case. In order to combat this error there is a diagnostic program constantly running to check for a negative count. This will automatically set the occupancy count to 0 rather than allowing it to go negative.

Another error may occur when the json protocol malfunctions and communication between the server and website goes down. If communications between the server and website goes down there the user can refresh the database by pressing the “Stop Data Collection” button on the “Setup page”. This will allow for the people counting algorithm to be shut down and reset. Data collection will begin again after being prompted by the user when the “Start Data Collection” button is pressed on the “Setup” page. If these measures aren’t effective in fixing the problem with the server a hard reset may be done by taking power away from the Occusense sensor system and rebooting all major functions. This can be done by simply unplugging the sensor system from the power supply and replugging it.

3.3 Safety Issues

The OccuSense system is a device mounted on top of the wall that weighs less than a kilogram. There are no physical safety hazards. On the other hand, the possibility of cyberattacks could be substantial. Since this is a privacy protected system, no hacker could use it to identify what people are present in the room, however, they could find out how many people are present in the room. The sensor system is protected by a firewall which is running in the Raspberry Pi and only accepts connections to the server and potentially to the client if they need to access it. The server on the other hand will be protected by SSL and with a login username and password. One of the biggest concerns would be the possible failure of the server, which is responsible for every sensor system. Backups and other mechanisms could be deployed if the server requires as it expands in the number of rooms it manages.

4 Technical Background

4.1 Melexis 90621 Thermopile technology

The MLX90621 contains 64 IR pixels (as a 16x4 array) that detects objects moving in front of the sensor and a PTAT (Proportional to Absolute Temperature) sensor to measure the ambient temperature of the chip. The outputs of these sensors are stored in an internal RAM and can be accessed via I^2C. This process begins by reading the EEPROM calibration data that is used to prep the thermal sensor data points to be processed into ambient temperature (Ta) readings. The measurement data (PTAT and IR readings) stored in the internal ram of the sensor is used to calculate the Ta values of the thermopile array. The OccuSense system uses the arduino to collect this data, with a class with the different parameters and functions to properly read the correct temperature values from the sensor. The melexis 90621 has a wide field of view of 60°x16°, requires 2.6V supply voltage and its current consumption is less than 9mA, with sleep mode consumption being less than 7µA.

4.2 Occupancy Detection

Once the data is collected from the arduino using the approach described above, the Raspberry Pi reads the temperature frames one by one through the serial port at a baud rate of 57600.

The detection algorithm is based on a threshold derived from the data shown in Figures 17, 18. The program uses the average temperature and variance of each frame to determine if there is a person present in the frame. The algorithm receives a frame and determines if it passes the threshold. If it does, it performs background subtraction on the frame and appends it to an array until another frame that does not pass the threshold is received, essentially storing the 4-5 frames of a person walking in or out. (If the array stores more than 5 frames, it only takes the last 5). Those frames are then put through a motion detection algorithm that tracks the time series from each pixel and computes a cross-correlation matrix for each cell with 63 other cells. After a simple error check with the last frame of the array it pushes the value to the server corresponding to a person walking in +1 or a person walking out -1. If the frame does not pass the threshold it performs a gaussian mixture model to update the background average of each pixel, to use in the background subtraction.

Average/Standard Variations of Temperatures Collected

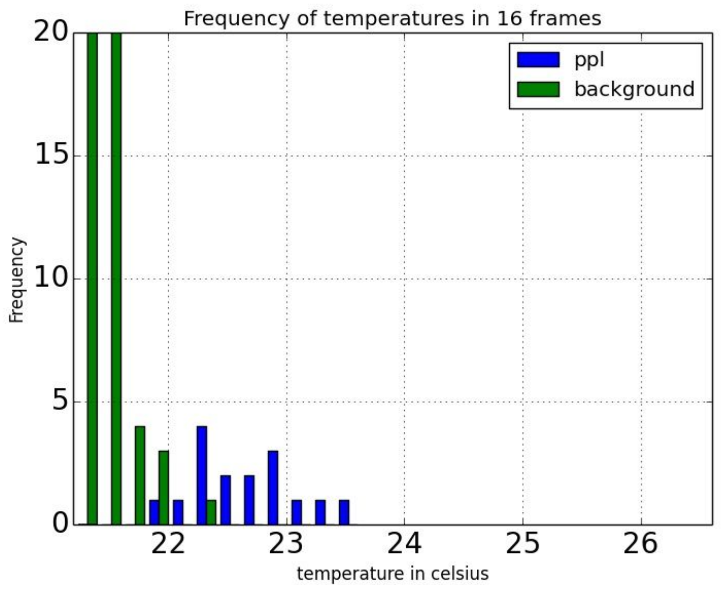


Figure 18: Average of temperatures collected in 16 frames

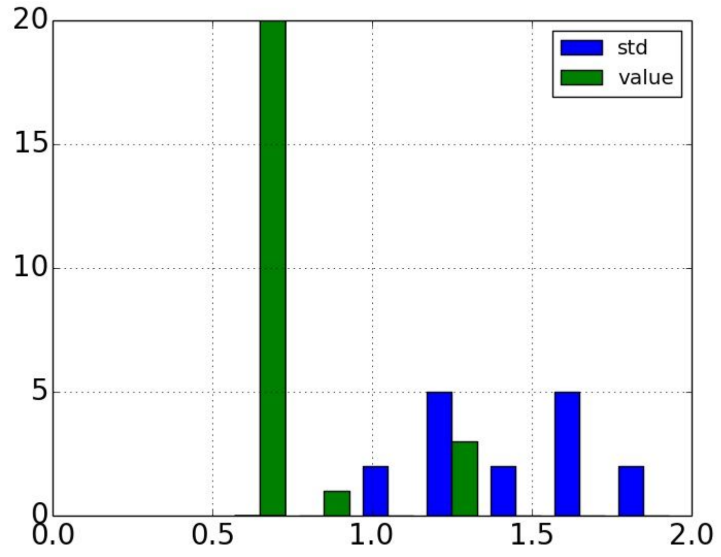


Figure 19: Frequency of standard deviations of all 16 frames, green is background and blue person.

This method has given us good results, as for now we calculated 80% accuracy and we are still in the process of collecting more data to come up with a more robust number. The following figures show graphical representations of what a person would look like in a frame and what a background frame looks like. The temperature scale is the same on both but is only expressed in the second frame.

Figure 20: Thermal array when there is no one in the frame.

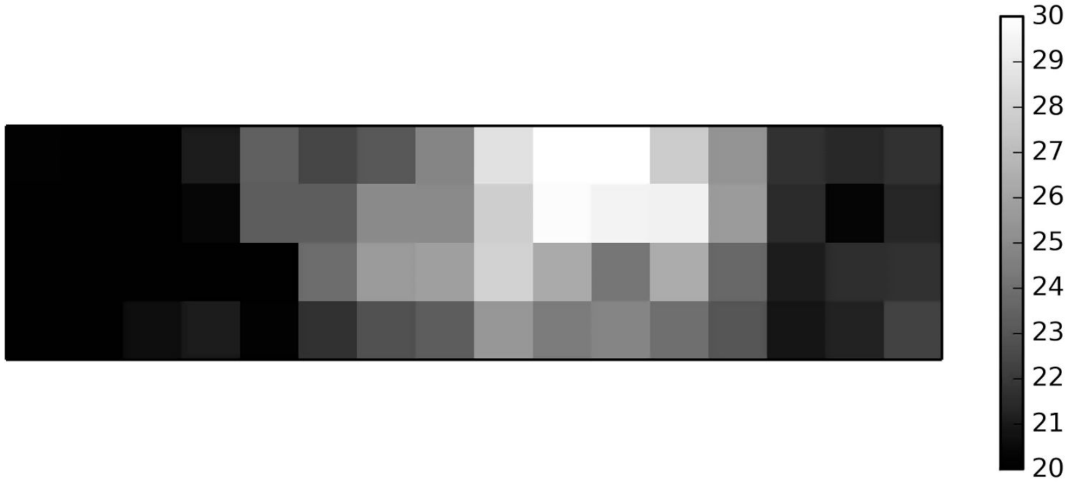


Figure 21: Person in the frame walking out of the room.

The OccuSense system will also include example videos, that have the same format as the pictures in Figures 19 and 20 for the user to visualize the way people are captured by the sensor. The system however will not provide access to the real time frames being captured by the sensor as it should not be needed for operation. If anything in the system fails there are troubleshooting techniques to follow. This videos are constructed using the matplotlib library provided by python and are usually made from sample frames where people one or two people walk through, short minute clips.

4.3 Parallax Wide-Angle Passive Infrared Sensor

PIR sensors allow to sense motion, almost always used to detect whether a human has moved in or out of the sensors range. PIRs are basically made of a pyroelectric sensor, which can detect levels of infrared radiation. Everything emits some low level radiation, and the hotter something is, the more radiation is emitted. The sensor in a motion detector is actually split in two halves. The reason for that is that we are looking to detect motion (change) not average IR levels. The two halves are wired up so that they cancel each other out. If one half sees more or less IR radiation than the other, the output will swing high or low. PIR requires voltage of 3V to 6V and current of 150 μA when idle and 3 mA when active.

4.4 Occuserver Technology

The OccuSense system relies on the "Occuserver" or the project's web-based server to combine all the outputs from the different systems and provide a user friendly environment to check room usage and interact with the sensor systems. The server is based on the free google service Firebase. Firebase stores all its data in MongoDB which offers the built-in capability for apps to scale automatically and gives each piece of data its own unique URL stored in JSON documents.

Communications between the Raspberry Pi and the server are done via json requests and posts. Whenever a person is detected by the MLX90621 thermal sensor three pieces of information are sent to the server: time detected, sensor ID, and sensor value. The time detected is the timestamp of when a person is detected entering or exiting a doorway, it is given in the form hh:mm:ss. The sensor ID represents a means of identifying which sensor is sending data to the web server. The sensor value comes in the form of a "+1" or "-1". When the value is "+1" this means someone was detected entering the room, while "-1" represents somebody exiting. The Raspberry Pi uses a json post via http to push the count (-1 or 1) to the appropriate room by keeping track of sensor IDs. It also keeps a socket open with the server to receive possible commands like reboot, or turn off for debugging and/or practical issues. Once the server sends the updated count to the Occusense website, entries are deleted in order to stay below the maximum capacity for the Firebase web server. As discussed, in the safety issues in section 3.3, the server will have logins with usernames and passwords protected by certificates using https.

4.5 Ownership, Licensing and Maintenance

Ownership of the system will be handed over to the customers leaving them with the responsibilities for proper licensing. Our system manual provides instructions for maintenance of the system if anything fails.

5 Cost Breakdown

Project Costs for Production of Beta Version (Prototype Unit)				
Item	Quantity	Description	Unit Cost	Extended Cost
MLX90621 Thermal Sensor	2	The Melexis MLX90621 Infrared sensor array	$46.00	$92.00
Raspberry PI 3	2	Raspberry PI 3 Model B is the latest version of Raspberry PI single-board computers	$39.13	$78.26
Arduino Mega	1	Arduino Mega microcontroller board	$45.95	$45.95
Arduino Uno	1	Arduino Uno microcontroller board	$16.75	$16.75
Parallax PIR sensor	2	Wide Angle PIR sensor	$12.99	$25.98
SanDisk Ultra 32GB microSDHC	2	Memory card for Raspberry PI 3	$13.45	$26.90
			Beta Version - Total Cost	$285.84

The above table reflects the cost to produce the two fully functioning sensor systems. Each sensor system includes its own MLX90621 Thermal Sensor, Raspberry PI 3 Microcontroller, Arduino Mega, Parallax PIR (Passive Infrared) sensor, and SanDisk Ultra 32GB microSDHC. Other non-cost physical components of the OccuSense sensor system that are not included in the above table include: plastic sensor system case, wiring, circuit board, and circuit materials (pullup resistors, inductors, and capacitors). The remainder of the components include free software tools such as Google's free web application Platform, Firebase. Overall costs may be lowered by making personalized circuit and PCB boards, but for the beta version the above cost is the most accurate estimate of the cost an OccuSense sensor system prototype.

The overall cost for an OccuSense sensor system is dictated by the number of usable entry ways for the intended room. To calculate the rough budget costs for a system with more than two entry ways use the following equation:

Cost = $[142.92 X number of usable entry ways]

6 Appendices

6.1 Appendix A - Specifications

- OccuSense was first assigned to be real-time occupancy-sensing system for a medium-size room capable of estimating at least 4 occupancy levels: empty, low occupancy, medium occupancy, high occupancy

- It was managed to reach higher level of accuracy of detecting, registering up to ~80% of people that came in or left the room.

- Choice of sensor for people count was deeply analyzed by our team and we decided on MLX90621 Far Infrared Sensor as the best choice as it provides privacy protection, low cost and enough resolution to achieve high accuracy.

- Since our system is privacy protected it allows to mount the OccuSense in bathrooms, changing rooms, and corporate conference rooms, etc.

- We refined the system by implementing PIR sensor to avoid any error accumulation in the long run.

- Complete system dimensions for each door: 90mm x 60mm x 36mm

- Power consumption of the whole system - about 3W. Overall yearly cost of running a sensor system < $10.

Appendix B – Team Information

Alex Bleda
(802)917-3275
ableda@bu.edu
Alex is a senior in Electrical Engineering with a minor in Computer Engineering. Interested in embedded systems, digital signal processing and networking. In his spare time he enjoys skiing and the outdoors. Alex's role in this project is to design and implement the sensor system, develop the detection algorithm and help with the interface to the web server.

Yerkebulan Nurkatov
(617) 459-3547
nyz@bu.edu
Yerke is a senior in Computer Engineering expected to graduate in the Spring of 2017. His role in the project includes the following: circuit board and PCB design and implementation.

Artem Bidnichenko
(857)-241-7789
artbidn@bu.edu
Artem is a senior majoring in Electrical Engineering, minoring in a Computer Science. He is interested in Computer Architecture and Machine Learning. His main role in the project is data analysis - visualization of sensor readings and creation of meaningful graphs and diagrams to show the system performance, as well Machine Learning algorithm implementation.

Miguel Cepeda
(513)602-1256
mccepeda@bu.edu
Miguel is a senior majoring in Computer Engineering, minoring in Biomedical Engineering expecting to graduate in May 2017. He is interested in front and back-end web design and networking. He is a member of the Boston University varsity tennis team. His main role in the project is design and implementation of the front-end/back-end of the website and client-server communication protocol

Krystal Kallarackal
(978)496-6282
krystalk@bu.edu
Krystal is a senior pursuing a degree in Computer Engineering at Boston University and expects to graduate in May of 2017. Her coursework focuses primarily on cloud computing, networking, algorithms and business innovation. Krystal's main contribution to this project entails client-server communication, database construction, and front-end/back-end of the website.