This is just an example to give you an idea of what is expected. Be sure to refer to the assignment document/rubric for the full expectations.

SMART BEDROOM MONITOR SYSTEM

Table of Contents

Table of Figures

Introduction

This year's design challenge is to develop a product for Smart Home applications that utilizes the Internet-of-Things (IoT). The design should count towards all of the following requirements:

- Internet of Things: extend network connectivity to all type of devices
- Home Application: designed for home/indoor usage
- Originality and Usefulness: the product should not have its exact counterpart currently on the market, and it must provide user with satisfaction.
- Low cost: the cost of the prototype cannot exceed $50.

Problem Statement

According to WebMD, all children under the age of twelve should get a minimum of ten hours of sleep each night (Shroff). Unfortunately, children in this age range only receive an average of eight hours of sleep, two hours less than the minimum required amount for this age group (Shroff).

The problem that we decided to address is how to provide more information to a parent about sleeping patterns of their children at night. One core of this solution is to provide parents an avenue to learn about their children's sleep. For example, parents may want to ensure that their children are getting adequate sleep. To help solve this problem, a parent can place our product in a child's bedroom to determine how much and how well their child has slept.

For this challenge, we will develop a bedroom monitoring system that will have the following general requirements:

1. Record information on activity in a bedroom.
2. Provide this information back to a main user through an integrated cell phone application.
3. The devices used to record information must be discrete and universally applicable to any bedroom setup.
4. The device must also be low cost – less than $50 for development.

We first conducted market research to discover what sleep monitor systems are currently on the market. We also looked at what type of information people are interested about their sleep patterns. With the competing factors in mind, we created a survey for parents.

Market Research

One of the first products we looked at was a baby monitoring device. This device is called Infant Optics DXR-8 Video Baby Monitor as seen in Figure 1. It is rated number one on the current baby monitor market according to SafeWise reviewing due to its sound-activated alerts, interchangeable lenses, temperature sensors, pan, zoom, and tilting camera options with a rechargeable battery. This product allows a user to monitor a baby using a real time application that notifies the parent if there is sound activity and displays the view of the situation using a camera.

Figure 1:Infant Optics DXR-8 Video Baby Monitor

This product is relatively expensive pricing $166. This large expense is one of the main problems we see with this product. Another problem with this device is that it only sends a video feed but does not store any data that it sees. For instance, this device could keep track of how long the infant is quiet and sleeping, vs awake and crying. This video feed is great to provide feedback to the parents about their baby, however, younger children might not like the idea of a camera in their room due to privacy purposes, so using a camera to solve this problem is not a viable solution.

Other baby products that we looked into provided similar visual feedback as well as audio. These systems all fell in a price range above $100.

Another interesting baby monitoring product that we looked at was a Smart Sock by Owlet. This product is a sock that slides onto a baby's foot and tracks the infant's heart rate and oxygen levels. This information is recorded and tracked over time. Alerts are provided to parents if oxygen levels or heart rates get too low. This is a great product to ensure safety and health for a baby.

Figure 2:Owlet Smart Sock 2 Baby Monitor

This information might not be as necessary for older children and adults. Maybe a similar product would be useful for elderly. For example, some elderly have heart conditions that cause their heartrate to drop dangerously low at night. This sort of feedback from a sock could be extremely useful for an older person and might even save their life.

Sleep Score Max, shown in Figure 3, is another product that we looked at that uses some interesting non-contact sensors to take detailed information about a user and provide feedback to help improve sleep quality. This product takes use of eco-location sensors that can track breathing rates. The information provided back to the users is quite extensive. Some of the information includes sleep duration, time to fall asleep, type of sleep (light, deep, rem) etc.

Figure 3:SleepScore Max

This product has a few easily identifiable problems. First, it will not record accurate information if there are multiple people sleeping in the same bed. Second, it must be pointed directly towards the person sleeping. Its primary purpose is to help the user change sleeping habits in order to get better sleep, rather than a monitor determining if someone has turned on a light or gotten out of bed while providing instant feedback to an application. It does a great job of recording data overtime; however, it needs a "learning period" in order to accurately track your information better. It is also connected to a larger sleep database to help "enhance the user experience by providing better feedback", however, some people do not like to have their information recorded and saved by someone other than themselves. This product costs $150 which is also relatively expensive for a sleep monitor.

One more interesting product that is used to record sleep data is the Nokia Sleep Tracker. This product is an under-mattress sleep sensor that connects to Amazon's Alexa and IFTTT (If This Then That). This is another non-contact device that tracks breathing, movement, and heartrate, then translates this into data about how well you slept.

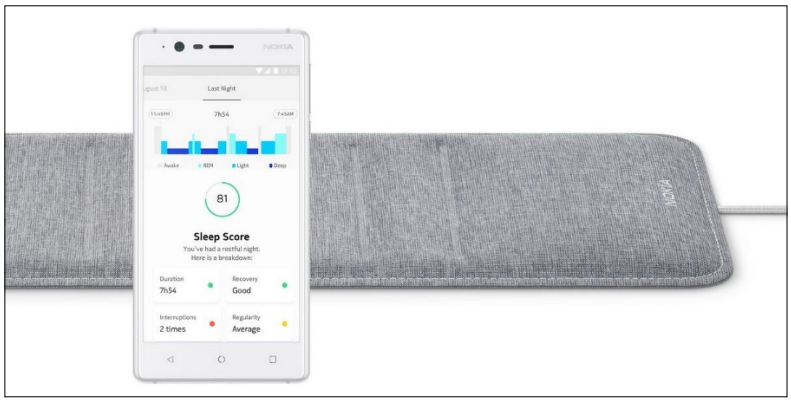


Figure 4: Nokia Sleep Tracker

One additional feature of this product is the Smart Home aspect which allows you to turn off lights, adjust room temperature, mute your phone, and set alarms automatically when you get into bed. This product has a price of $100.

A final product that we looked at is the SevenHugs HuGONE sleep tracker, shown in Figure 5. This device is designed for a family. The sensor itself is placed under the sheets of a bed near your head and transmits data to a base station.

Figure 5: SevenHugs HuGone

The big problem that appears in the reviews of this device is that it is intrusive in that there is a physical sensor that is annoyingly placed near your pillow. Also, it is said to not return super accurate information. It is relatively expensive at $95.

A quick overview of some other products researched are a smart pillow which records similar data as some of the other monitors. Smart blankets and wearables are also used to track data. All of these options are relatively expensive (~ $100) with a primary goal of improving a user's sleep data.

We also conducted a survey. The results of this survey are located below.

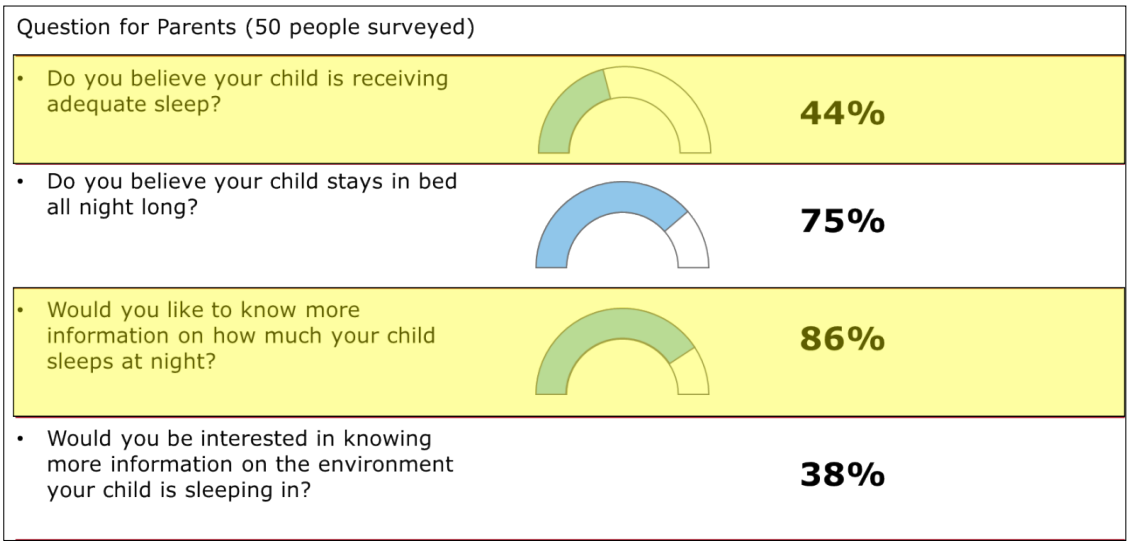


Figure 6: Survey for parents

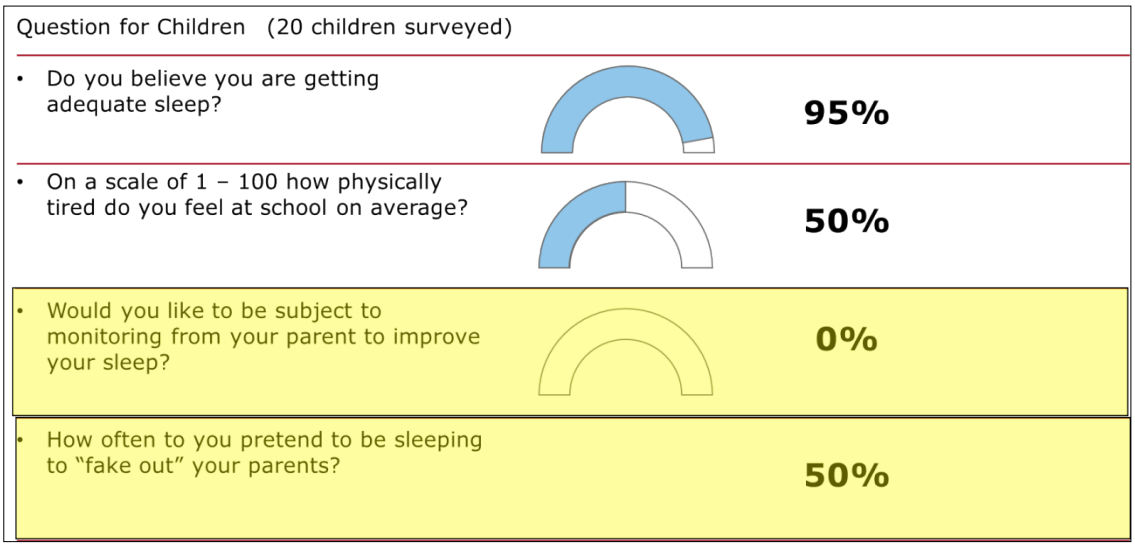


Figure 7: Survey for Children

These results are really interesting because it shows there is a discontinuity between when parents believe their children are going to bed vs when the children are actually going to bed. This is identified because 75% of parents believe their child is staying in bed all night long and 50% of children pretend to be "faking out" their parents. This indicates there is a market for this product.

Another interesting statistic is that 0% of children want to be subject to monitoring form their parent. This will help us define our product specifications.

Product Requirements

Customer Requirements

From the market research we came up with a list of requirements that a consumer appears to be interested in when purchasing a sleep tracking device.

Explicit

1. Physical Device
    a. Non-Contact
    b. Willing to pay above $100
    c. Small and compact for devices placed near bedside
    d. Variety of methods for implementation as long as data recording methods are accurate
    e. User application to easily access recorded data
2. Data Recorded
    a. Total Time slept
    b. Time to fall asleep
    c. Time in light sleep
    d. Time in deep sleep
    e. Time in rem sleep
    f. Time to wake up
    g. Feedback on how to improve sleep quality
3. Use Cases
    a. Individuals interested in learning more about how well they slept
    b. Family members trying to track how well other family members slept.

Implicit

1. Physical Device
    a. The device itself is safe to put near your bed
    b. Non-intrusive
    c. Easy to setup and implement
2. Data Recorded
    a. Accurate and valuable to the consumer
3. Use Cases
    a. The user wants to be tracked!
    b. Different from what we want to track where the user doesn't understand why it's important to get good sleep

There are a few surprises that arose after looking at this research. It appears that contact devices such as Fit-Bit watches and other wearables are not as sought-after as non-contact devices. There appears to be a variety of methods in which sleep data can be tracked accurately. While searching for devices that can be used, we were looking for devices that can be used to track family members discretely and truthfully. However, there were no

devices found that accomplish this. Therefore, the market that we are exploring is similar to sleep tracking devices, but we are really looking for a consumer that is trying to record sleep data on someone that doesn't necessarily wants to be tracked. Therefore, our product will have some modified specifications.

The customer requirements for this project provide good insight into what our product must provide. Since this tracking system is different from normal tracking systems in that it is designed for a parent to track their child, we will have slightly different requirements.

Reliable – This device that we design must provide accurate and valuable information to the consumer. This information's reliability is also dependent upon its functionality as a sensor that cannot be easily tricked. The reason for this is that "faking information" is a common problem with children. This system must be child proof in that a child cannot easily fake the sensors into sending false data to parent. For example, if a child wants to play video games late at night, that child cannot fake the pressure sensors into thinking that he is asleep in bed by placing an item on the bed whereas he is actually playing video games in another room.

Discrete – In order for this device to be successful, it must be discrete in that a child will not see the device as threatening or bothersome. For example, if a camera is mounted right next to the child's bed, he or she might see this device as a "threatening device" that is invading their privacy. However, if the device is shaped like a sticker that can be placed on their headboard, the child will be less suspect to the sensor.

Easy User Interface – The product itself must provide a user interface to interact with and view recorded data. This interface must have nice graphical data representing a day-by-day basis as well as a long-term basis. This data must also be viewable on a laptop or computer.

Diverse User Community – This device is intended for a second user to track a primary user. The primary user for this device must have a variety of age groups from infants, to children, to elderly and even adults that need to be monitored given he or she has poor sleeping habits that might even affect health.

Variety of Sensors to provide a variety of information – This device should be able to provide data on how long someone was in bed sleeping and information on the bedroom environment.

Affordable – Although most sleeping trackers are relatively expensive, to be competitive, this device can be much more inexpensive.

Easily Implementable – This device must be easy to install in any bedroom by any person.

Safety – This device cannot pose any hazards to the primary or secondary users.

Feedback – this device must be able to take feedback from the secondary user in order to provide recommendations to the secondary user about how to change the habits of the primary user.

These are the most important product requirements that must be met in order to consider our product a successful, marketable product.

Product Specifications

Based on the Product Requirements, a general system structure can be defined. The figure below shows a functional block diagram of both the sensor interface and the application interface.

1. Four Sensors
    a. Pressure Sensor
    b. Light Sensor
    c. Microphone
    d. Temperature Sensor
2. Four transmitters, one receiver
3. Backend Database to store information
4. Battery powered sensors and wall powered receiver
5. Most likely going to need five microprocessors (for the transmitters and receivers)
6. User Interface
    a. Main page
    b. Interrupt page
    c. Morning update page
7. Fun (potentially customizable) packaging for light and microphone sensors

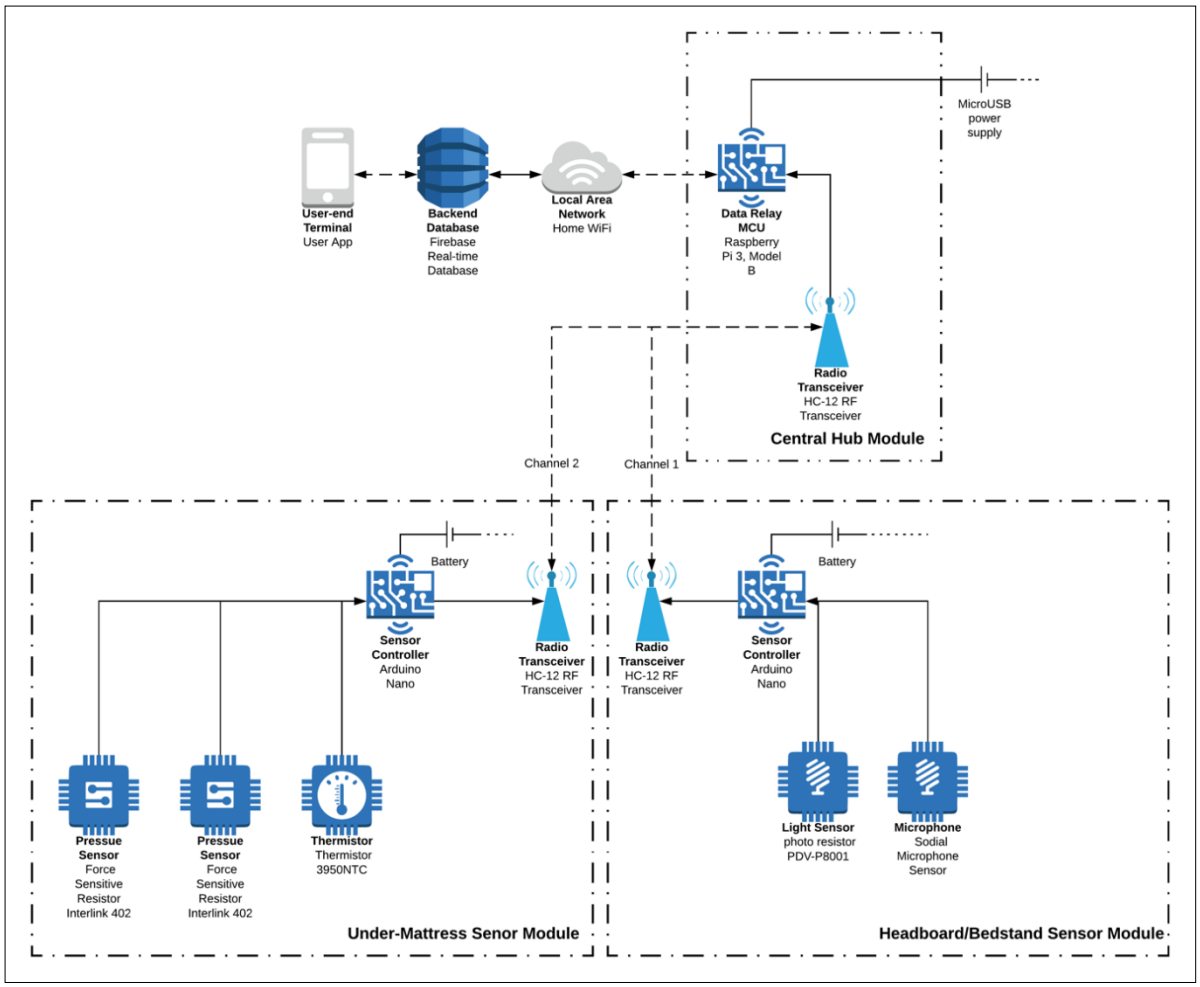


Figure 8: System Block Diagram

Looking at the Sensor Interface of the block diagram, it is clear that there are four different types of sensors that this device must have in order to record valuable information on if someone is in bed or not sleeping. The purpose of the pressure sensors is to determine if someone is actually in bed, as well as any changes in pressure over night. These changes are used to determine restlessness or physically getting out of bed. The light sensor is used to determine if someone is in bed but is actually awake reading, on their phone, or if the room light is on. The microphone is used to determine if the only sound in the room is breathing or snoring rather than a conversation such as a phone call. Finally, the temperature sensor is used to make sure that the system isn't being tricked into thinking that someone is in bed but actually isn't. One way to fake this system is to pile books onto the bed to activate the pressure sensors, turn all the lights off in the room, and leave the room to play games elsewhere. However, with a temperature sensor detecting an actual person in bed, the system is much more difficult to fake.

All of these sensors transmit data to a receiver which will then store this data in a database that can connect to a main laptop and cell phone interface. The cell phone interface main page will be used to select a person. From there, data can be viewed. The data that can be viewed is the current data as in what data for that night is currently being recorded, or you can view old data by providing a range. Finally, you can provide feedback to the system by setting that person's happiness / awareness for the day. Thus, trends can be identified creating a sleep to happiness relationship.

The application also provides an interrupt mode. There are two types of interrupts that can be generated. The first interrupt is if the person has gotten in or out of bed. The second interrupt is if the light in the room is on.

Finally, every morning, the application will provide the user with a summary of each individual's sleep for the night before. All the data will include total time in bed, total time asleep, total time with lights on, number of times gotten out of bed, and what time that person left bed along with maybe a few more valuable pieces of data that future customers might consider useful.

Visually, the sensor interface needs to be packaged correctly so that the person being recorded does not get bothered by the sensors. The pressure sensors and temperature sensors do not need to be very visually interesting because they can be placed underneath a mattress. The light and sound sensors need to be packaged correctly so that they don't bother the person being recorded. Some potential ideas are to package the sensors in a toy shaped 3D printed object.

Competitive Value Analysis

There are a few competitive products that can be compared to our product. These products are not necessarily designed for the same purpose that our “Sleep and Sound” product is designed for, but they can be modified to fit the general purpose. These products are the Infant Optics DXR-8 Video Baby Monitor, Owlet Smart Sock 2 Baby Monitor, SleepScore Max, Nokia Sleep Tracker, and SevenHugs HuGone.

The Infant Optics DXR-8 Video Baby Monitor is essentially a camera that is motion activated and connects to a user cell phone to display a live stream of the view of the camera. This is a competitor to our device because it is system designed for a parent to monitor a child. It also has a user interface that displays valuable information about the child being monitored.

The Owlet Smart Sock 2 Baby Monitor is a competitive device because it is another way to monitor a child. This device is a sock that slides onto a child’s foot that monitors vital signs as well as sleep data. This data is displayed from a phone application.

The SleepScore Max is a device that sits next to a users bed and uses echolocation to motor heart rate, breath rate, and eye motion. This is a very high accuracy device that can generate extremely accurate results, however, it needs to be oriented correctly and can only record data for a single person. This device offers tips the next day to help the user get better sleep.

The Nokia Sleep Tracker is a device that goes on top of a users mattress. It records sleep data, heart rate, breathing rate, and much more. This device is very accurate and also has a user interface to display all this information. It also has a learning mechanism to get better data readings.

Finally, the SevenHugs HuGone device is a series of sensors that can be placed in many rooms to record sleep data for multiple children. This device solves a similar problem to the device we are creating. There are many problems with device such as accuracy of information, how discrete the recording devices are, and the actual user interface.

Table 1: Value Analysis of Competitors

		Infant Optics DXR-8 Video Baby Monitor			Owlet Smart Sock 2 Baby Monitor			SleepScore Max			Nokia Sleep Tracker			SevenHugs HuGone			Our Device		
Criteria	Weight	Value	Rating	Weight	Value	Rating	Weight	Value	Rating	Weight	Value	Rating	Weight	Value	Rating	Weight	Value	Rating	Weight
Non-Contact	8	yes	10	80	no	2	16	yes	9	72	yes	10	80	yes	8	64	yes	10	80
Data Recorded	10	yes	5	50	yes	7	70	yes	10	100	yes	7	70	yes	8	80	yes	8	80
Reliable	7	yes	6	42	yes	10	70	yes	6	42	yes	10	70	no	5	35	yes	8	56
Discrete	7	no	3	21	no	1	7	no	5	35	yes	10	70	no	5	35	yes	8	56
Easy User Interface	6	yes	5	30	yes	7	42	no	6	36	yes	7	42	yes	6	36	yes	8	48
Easily Implemetable	5	yes	8	40	yes	10	50	yes	10	50	yes	7	35	yes	8	40	yes	9	45
Feedback from user	3	no	3	9	no	4	12	no	4	12	no	6	18	no	7	21	yes	8	24
Unhackable	8	no	1	8	no	1	8	no	6	48	no	6	48	no	5	40	yes	8	64
Total Value			41	280		42	275		56	395		63	433		52	351		67	453

Figure 9: Value analysis of competitors.

There is one main area of sensitivity in this value analysis. That area is the data-recorded. Almost none of these devices records all the data that we want to be looking at and this data is weighted significantly higher than all the other criteria.

From this value analysis, the device that we are producing should be superior to all other products in this market based on the customer driven criteria for this problem. The most competitive device with our own is the Nokia Sleep Tracker which is the mattress pad. This product really encompasses a large amount of the product that we are trying to create, however, there are some features that it is missing. For example, it is easily hackable in that if you just take the mattress pad off, then the system no longer works. The largest issue that can be identified with this device is that there is no data recorded on the amount of light in the room or the volume of noise. This data is valuable to a parent as seen from our survey and interviews.

Design Approach

Sensors

Our system design will contain four different types of sensors which are a pressure sensor, a light sensor, a thermistor, and a microphone. These sensors will work together in our system to provide reliable data about the children's sleep quality, whether that is their presence in bed or the environment in which they are sleeping. All the data recorded by the sensors will be read by the Arduino microcontrollers to which they are connected.

Pressure Sensor

Our product will contain two Force Sensitive resistors Interlink 402 as the pressure sensors that are part of the under-mattress sensor module. These will be placed strategically under the mattress, at the normal high-pressure areas of a person in bed, preferably under the head and bottom of the person in bed. The red areas in the figure below show the best positions where the force sensitive resistors will be most effective.

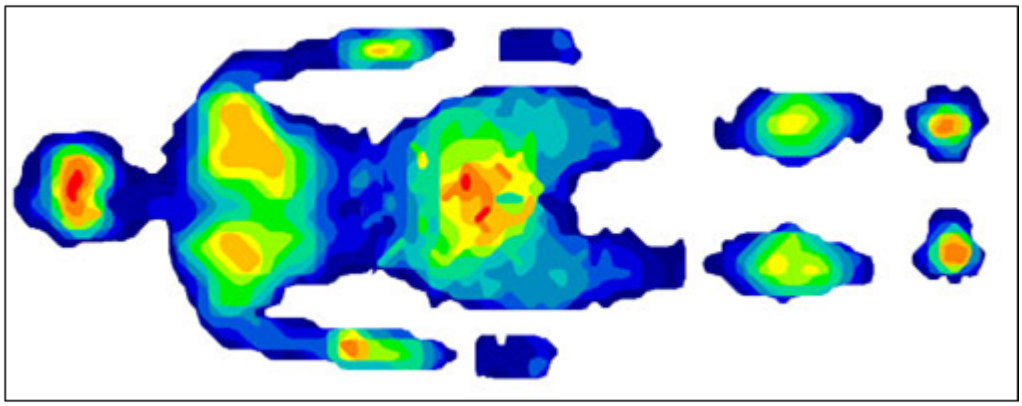


Figure 10: Typical pressure areas of person in bed.

Force Sensitive Resistor Interlink 402	
Functionality	The functionality of the Force Sensitive Resistor Interlink 402 is to detect physical pressure or weight under the mattress when the child is in bed. The FSR is made of 2 layers, the semiconductor part and the interdigitating electrodes, separated by a spacer. When it is pressed, the Active Element dots touch the semiconductor, which makes the resistance go down. Although the force sensitive resistor is not the most accurate to detect exactly how many pounds of weight is on it, our product will still be able to give a good estimate of the weight along with a range of response to track the sleep quality.
Implementation	The pressure sensor will be implemented by connecting one end to 5V power of the Arduino microcontroller. The other end will be connected to a 10kΩ pull-down resistor in a voltage divider configuration to ground, as shown in figure 5. The point between the fixed pull-down resistor and the pressure sensor will be connected to one of the Arduino analog input pins. The pull-down resistor will prevent an undefined state at the input and pull the voltage down to ground (logical zero value). A value of 10kΩ for the pull-down resistor will provide the full range voltage reading and limit current.
Specifications	Power supply: 5V supply, can easily also use 3.3V supply GND: Digital ground Actuation Force: 0.1 Newtons Force Sensitivity Range: 0.1 – 10.0^2 Newtons, 0.022lb – 22.48lb Resistance Range: Infinite(no pressure), 100kΩ(light pressure) to 200Ω (max. pressure) Current: less than 1mA, dependent on pull-down resistor and supply voltage

To test the pressure sensor, we will set it up with the Arduino microcontroller and make sure the serial monitor is reading accurate data from the sensor when it is being pressed. If there is nothing on top of the pressure sensor, then the Arduino serial monitor will read “No pressure detected.” An approximate Newton force will be measured by the sensor and read by the Arduino along with an analog voltage reading which ranges from about 0 to 1023 which maps to a voltage reading of 0V to 5V. This voltage reading is converted to resistance which is finally converted to force. This Newton force reading can then be converted to grams or pounds for a more common unit among product users. The table below shows the approximate analog voltage of the pressure sensor with a 5V supply and 10kΩ pulldown resistor along with an approximate current through the sensor and voltage divider configuration.

Force (lb)	Force (N)	FSR Resistance	(FSR + R) ohm	Current thru FSR+R	Voltage across R
None	None	Infinite	Infinite!	0 mA	0V
0.04 lb	0.2 N	30 Kohm	40 Kohm	0.13 mA	1.3 V
0.22 lb	1 N	6 Kohm	16 Kohm	0.31 mA	3.1 V
2.2 lb	10 N	1 Kohm	11 Kohm	0.45 mA	4.5 V
22 lb	100 N	250 ohm	10.25 Kohm	0.49 mA	4.9 V

Figure 11: Approximate Analog Voltage with 5V supply and 10kOhm Resistor.

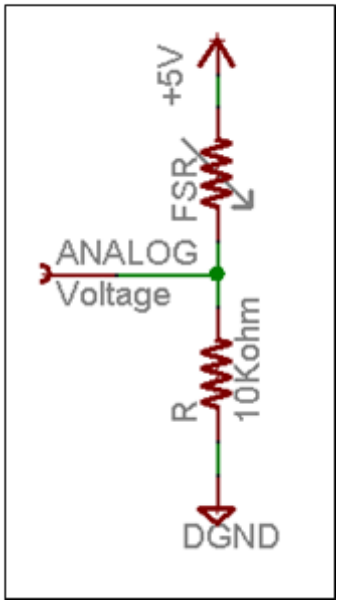


Figure 12: Circuit Diagram for Pressure Sensor with 10kOhm Pull-down Resistor.

Thermistor

Our product will also contain the 3950 NTC 10K Precision Epoxy Thermistor as the temperature sensor that is part of the under-mattress sensor module. The thermistor will be placed under the sheets of the bed to detect whether the bed is occupied or not. This negative coefficient thermistor is a contact sensor that will have its resistance decreased as it detects heat and increased when it gets cooler. Since the length of the thermistor is about 18 inches, the user will have enough bed area covered and can strategically place it around the bed to have more efficient measurements while making it as less intrusive as possible for the person in bed.

3950 NTC 10K Precision Epoxy Thermistor	
Functionality	The main functionality of the 3950 NTC 10K Precision Epoxy Thermistor is to detect when the bed is occupied, especially when a child might try to fool our product by placing objects over the pressure sensors since a temperature sensor is more difficult to fool. The thermistor or thermal resistor is made so that resistance changes drastically with temperature so that it can be 100Ω or more of change per degree. Whenever the thermistor senses an increase in temperature, its resistance will decrease. Conversely, when temperature decreases, the resistance will increase. Our product will just need a reading of a person being in the area measured by the thermistor in bed, so the thermistor temperature range will be more than enough to accurately detect presence in bed by heat from the human body.
Implementation	The thermistor will be implemented by using a 10kΩ resistor like the configuration for the pressure sensor. The reason for this is because our Arduino microcontroller does not have a resistance-meter built in. It only contains a voltage-reader, which is know as the analog-digital-converter. To convert the resistance into a voltage, we use another resistor and connect them in series. While this added resistor is fixed, we can measure the voltage in the middle from the voltage-divider equation. One of the ends of the fixed 10kΩ is connected to 5V power of the Arduino microcontroller while the other end is connected to one pin of the thermistor. The other pin of the thermistor is connected to ground. The point between the fixed resistor and the thermistor will be connected to one of the Arduino analog input pins.
Specifications	Power supply: 5V supply, can easily also use 3.3V supply GND: Digital ground Resistance at 25° C (Room temperature): 10K +- 1% Temperature Coefficient: 3950 +-1% Temperature Range: -55 °C to 125 °C No amplifier needed to read minute voltages Length: 18 inches Resistance range: ~480kΩ(at -55°C) to ~0.3341kΩ(at 125°C)

To test the thermistor, we will set it up with the Arduino microcontroller and make sure the serial monitor is reading accurate data from the sensor whenever the temperature around it changes, specifically by human touch. Similar to the pressure sensor, an analog reading which ranges from about 0 to 1023 and maps to a voltage reading of 0V to 5V will be read by the Arduino microcontroller and then will convert this value into resistance. This resistance is then finally converted to temperature.

Light Sensor

Our product will also contain the photo resistor PDV-P8001 as the light sensor that is part of the headboard/ bedstand sensor module. This will be preferably placed on the headboard or inside an object where it is unnoticeable but still able to detect illumination in the room.

Photo resistor PDV-P8001	
Functionality	The functionality of the Photo resistor PDV-P8001 is to detect whether the light in the room is on or off to make sure the room environment is the ideal for the child to have a good quality of sleep. When the active area of the photo resistor is exposed to more light, the resistance will go down. Conversely, the resistance will increase as darkness increases. The resistance is about 5-10kΩ when there is light while going up to 200kΩ when it is dark. Since we would like to determine basic light changes, the photo resistor is enough to work as the light sensor for our product which will monitor the room environment.
Implementation	The light sensor will be implemented by connecting one end to 5V power of the Arduino microcontroller and the other side to a 10kΩ pull-down resistor in a voltage divider configuration to ground, as shown in figure 8. Then the point between the fixed pull-down resistor and the photocell resistor is connected to one of the Arduino analog input pins. A value of 10kΩ for the pull-down resistor will provide the full range voltage reading necessary to meet our product specifications and customer requirements to monitor a dark or lit room.
Specifications	Power supply: 5V supply, can easily also use 3.3V supply GND: Digital ground Sensitivity Range: Respond to light between 400nm (violet) and 600nm (orange) wavelengths, peaking at about 520nm (green) Resistance Range: 200kΩ(dark) to 10kΩ (10 lux brightness) Current: less than 1mA on average, dependent on power supply voltage Operating and Storage Temperature: -30°C to 75°C

To test the light sensor, we will set it up with the Arduino microcontroller and make sure the serial monitor is reading accurate data from the sensor when illumination in the room increases or decreases. An analog reading which ranges from about 0 to 1023 will be read by the Arduino. The specific analog reading given will then determine the amount of light in a qualitative manner. For example, if the analog reading is less than 10, then the serial monitor will print “dark.” Likewise, if the analog reading is more than 800, then the serial monitor

will print “very bright.” The table below shows the approximate analog voltage of the light sensor with a 5V supply and 10kΩ pulldown resistor along with an approximate current through the sensor and voltage divider configuration.

Ambient light like…	Ambient light (lux)	Photocell resistance (Ω)	LDR + R (Ω)	Current thru LDR +R	Voltage across R
Dim hallway	0.1 lux	600KΩ	610 KΩ	0.008 mA	0.1 V
Moonlit night	1 lux	70 KΩ	80 KΩ	0.07 mA	0.6 V
Dark room	10 lux	10 KΩ	20 KΩ	0.25 mA	2.5 V
Dark overcast day / Bright room	100 lux	1.5 KΩ	11.5 KΩ	0.43 mA	4.3 V
Overcast day	1000 lux	300 Ω	10.03 KΩ	0.5 mA	5V

Figure 13: Approximate Analog Voltage for light sensor with 5V supply and 10kOhm Resistor.

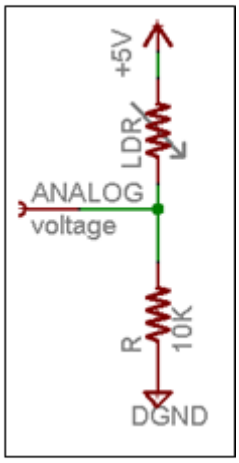


Figure 14: Circuit Diagram for Light Sensor with 10kOhm Pull-down Resistor.

Microphone

Our product will also contain a Sodial Microphone Sensor with High Sensitivity Sound Detection Module as the sound sensor that is part of the headboard/ bedstand sensor module. Similar to the light sensor, it will be preferably placed on the headboard or inside an object where it is unnoticeable and not intrusive but is still able to detect noise in the room.

Sodial Microphone Sensor	
Functionality	The functionality of the microphone sensor is to detect if there are other sounds in the room apart from breathing and snoring that might interfere with the correct room environment for the children to have a good night's sleep. Also, it will be used in our product to determine if the child is staying awake talking on the phone with a friend or listening to music. The sensor will give a measurement of how loud a sound is. The sensor has 3 main components on its circuit board. First, the sensor unit which measures the area physically and sends an analog signal to the second unit, the amplifier. This amplifier amplifies the signal, according to the resistant value of the built-in potentiometer on the circuit board. The signal is then sent to the analog output of the module. The third component is a comparator which switches the digital out if the signal falls under a specific value. Whenever the sensor detects sound intensity above a certain threshold that we will set, a signal will be sent if the measurement of the sound in the room exceeds the threshold. The built-in potentiometer can be used to adjust the sensitivity of the digital output pin.
Implementation	The sound sensor will be implemented by connecting its power supply pin to 5V power of the Arduino microcontroller and its ground pin to ground on the Arduino. The analog and digital pins of the sensor will be connected to the analog and digital pins of the Arduino. The analog out for the sensor will give a direct microphone signal as voltage value, and send a signal whenever the sensor detects sound above a certain threshold. The status of the digital pin will also be sent to see when the threshold has been exceeded or not.
Specifications	Power supply: 5V supply GND: Digital ground Contains built-in power On LED light Contains built-in potentiometer to adjust threshold level Current: Dependent on power supply

To test the sound sensor, we will set it up with the Arduino microcontroller and make sure the serial monitor is reading accurate data and reading a signal from the sensor whenever sound increases in the room and goes above the set normal threshold. We will need to take into account the possible breathing and snoring of the child, so our plan is to play some music or noises that might interrupt someone's sleep and use the potentiometer in order to adjust the

sensitivity of the sound sensor. An analog reading which ranges from about 0 to 1023 will be read by the Arduino. The specific analog reading given will then be converted to voltage and outputted to the serial monitor. Whenever the signal goes above the threshold, the serial monitor will read “reached.” Likewise, whenever the signal is still at a normal threshold, the serial monitor will read “not reached yet.”

Microcontrollers

There are three microcontrollers in our system design. Two microcontrollers are Arduino Nanos. The purpose of the Arduino Nanos is to read in sensor data and control transmitters to pass data through the system. The third microcontroller is a Raspberry Pi which is used to ask for data from the microcontrollers and transmit data to the backend database.

Arduino Nano

This microcontroller is used for controlling input from sensors and wirelessly sending data to the central hub.

Arduino Nano	
Functionality	The main functionalities of the Arduino Nanos are to read sensor data and control RF transmitters to send data to the raspberry pi. There is no data processing going on with this microcontroller. The initial functionality of this module is waiting for a message from the raspberry pi. This message will contain information on what sensor to read data from. Then the microcontroller will read the data from the sensor and transmit that data back to the raspberry pi. Then the microcontroller will continue to wait for another command.
Implementation	This module will be implemented by connecting the controller to a 9V battery for power. Then connecting the sensors using the 5V output and two analog input pins. The transceivers will also be connected to 5V and GND. Three additional digital pins are required to control the transceivers set, tx and rx pins.
Input	Circuit Input: Two or three Analog In pins connected too two sensors 9V input to Vin to power the device Specifications: VIN: 7V – 12V Digital I/O Pins: 22 pins (6 PWM) Analog In Pins: 8

	Current I/O Pins (digital): 40mA (max) Power Consumption: 19mA
Output	Circuit Output: Three digital pins that connect to the transceiver. Supplies transceiver with 5V DC and GND pins. Supplies sensors with 5V DC and GND pins for voltage divider. Specifications: Digital I/O Pins: 22 pins (6 PWM) Current I/O Pins (digital): 40mA (max)
Status	Microcontroller: ATmega328 Operating Voltage: 5V Flash Memory: 32 KB with 2KB for bootloader Clock Speed: 16 MHz

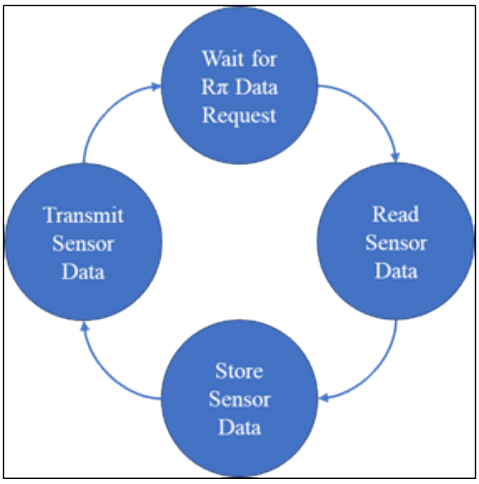


Figure 15: Arduino Microcontroller Flow Diagram.

This module will be tested in a few stages. The first stage is to be able to transmit data between the microcontroller and the raspberry pi. This test will prove that communication is possible. The second stage is to be able to receiver data from the sensors that accurately represent the real world. For example, if the pressure sensor is being tested with the microcontroller, we will place a known weight on the pressure sensor and interpret this data in the microcontroller and display the information on the serial monitor. If the weights match, then we know that the microcontroller is receiving accurate data form the pressure sensor. The final stage of testing will be to transmit the sensor data to the raspberry pi.

Raspberry Pi

This microcontroller is used for receiving data from the sensor modules and wirelessly uploading data to the database.

Raspberry Pi	
Functionality	The main functionality of the raspberry pi is to ask for data from the sensor modules, receive the data, then upload that data to a backend database. There will be a period of waiting used as the rate the raspberry pi asks for data from the sensor modules. This is equivalent to how many datapoints can be recorded in a given period of time. We will write an algorithm that receives lots of data but only uploads certain data to the database so that any inconsistent data is removed before being uploaded.
Implementation	There are two main connections that need to occur with the raspberry pi. The first connection is with a user wifi so that the backend database can communicate with the user application. The second connection is with the transceiver so that data can be asked for and received by the raspberry pi. This connection will require three digital I/O pins, 5V and GND.
Input	Circuit Input: 3 x GPIO for transceiver. Specifications: VIN: 5V / 2.5A DC via micro USB connector GPIO: 28 Input / output pins
Output	Circuit Input: Wireless Bluetooth to upload data to database. 5V and GND to transceiver Specifications: 2.4GHz and 5GHz IEEE 802.11 wireless LAN, Bluetooth Gigabit Ethernet over USB 2.0 (max throughput 300 Mbps) 4x USB 2.0 ports GPIO: 28 Input / output pins
Status	Processor: BCM28370B0, Cortex-A53 64-bit SoC @ 1.4GHz Memory: 1GB LPDDR2 SDRAM Access: Extended 40-pin GPIO header

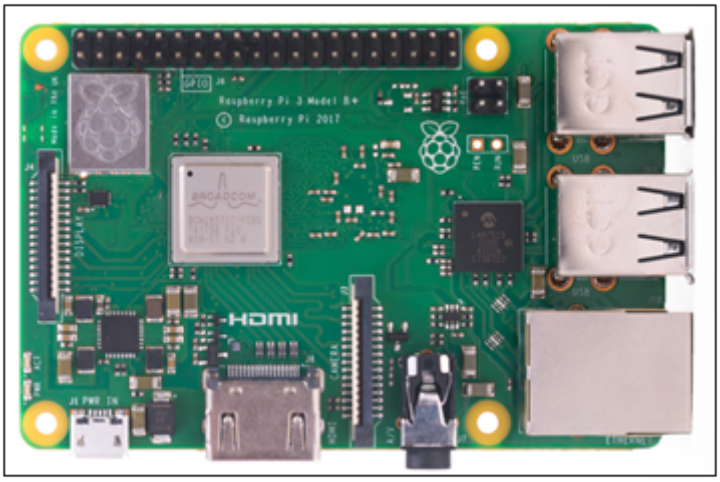


Figure 16: Raspberry Pi image.

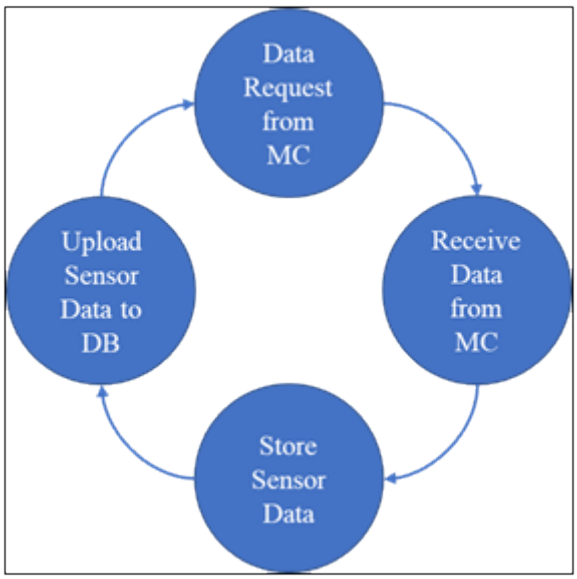


Figure 17: Raspberry Pi Flow Diagram.

This microcontroller will be tested in three stages. The first stage is being able to upload information to a database that is accessible from a user application. This will be implemented first to make sure that the principle of using a raspberry pi to store data is feasible within the time constraints of our project.

The second stage of testing is to communicate with the other microcontrollers through the HC-12 RF transceiver. This is implemented second so that real data can be received.

The third stage is taking the data received from the transceivers and uploading it to the database. There might be an algorithm that takes in a few datapoints at once, and manipulates them to only upload one relevant datapoint rather than many where one might be inaccurate.Page Break

Transceivers

The transceivers are used to wireless transmit data between microcontrollers.

HC-12 RF Transceiver

HC-12 RF Transceiver	
Functionality	The main functionality of the transceiver is to transmit data between microcontrollers. The transceivers at the Arduino Nanos are expecting to receive a message from the Raspberry pi transceiver asking for data from a specific sensor. If that message is received, then the proper receiver will communicate with the Arduino to get this data. Then the transceiver on the Arduino will send this data to the Raspberry pi transceiver which is waiting for a response.
Implementation	The implementation for these devices is simple. There are only three data pins, a power and ground. The data pins are controlled via software and the power and ground are direct connections.
Input / Output	Circuit I/O: VCC: Power supply input DC 3.2-5.5V @ 200mA (minimum) GND: common ground RXD: Input, weak pull-up. UART - 1K internal series resistance. TXD: Output, UART output port 1K internal series resistance. SET: Input, internal 10k pull-up resistance. Antenna: Input / Output @ 433MHz
Specifications	Max Power Output: 100mW (20dBm) ~ 1km. Unlimited bytes transmitted at one time. 100 channels with stepping of 400kHz. Receiving sensitivity is -117dBm at baud rate of 5,000 bps.

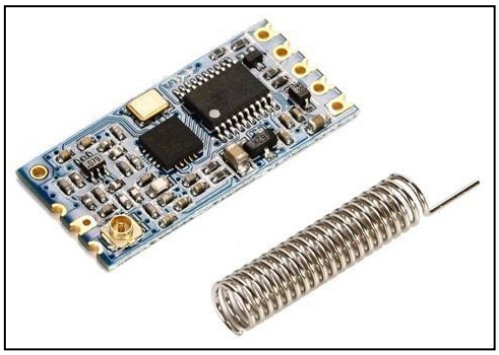


Figure 18: : HC-12 Wireless Transceiver 433MHz.

To test the transceivers, we will first test transmission between Arduino Nanos. The transmission will be a simple string message that will be printed through the serial monitors. Then we will execute some more sophisticated commands that require delays and timing representing an ACK/NACK protocol.

Software

By the nature of IoT product, our system relies heavily on network to achieve intuitive data management and presentation. Most of the networking capability is achieve through the internet to which both the central hub (hardware), backend database (software), and user mobile App (software) connect. In this section, we demonstrate how the software completes the hardware-to-user conversation.

Backend Database

Our application needs to be able to interact with the users through internet, Bluetooth or other cloud technologies. We chose to use the Wi-Fi (internet) as the overall communication interface and implemented Firebase® to serve as the backend database.

Firstly, the functions of the database are to receive, store, sensor readings acquired by the central node from the sensor modules. The MCU has built in Wi-fi module and when a user first setup the application, he or she will be prompted to setup internet connect for the product. Once connected, the module is able to upload data to the Firebase Real-time Database. Moreover, security is significant concern because we are collecting otherwise personal and sensitive data, it is noteworthy that the user must sign up and log in before accessing the database. In other words, one must be authorized in order to read from and write to the online database.

Secondly, as discussed above the database also holds user authorization information. Users can sign up with their email, log in using existing credentials or retrieve password by reset-password email link. All these operations are encrypted, and users' email-password combination are implemented in hash code to enhance security level.

The following figure shows the IO's of the database. The data are mainly stored in the historical database, which interfaces with the real-time database. The real-time database has very low-latency and talks directly with the hardware.

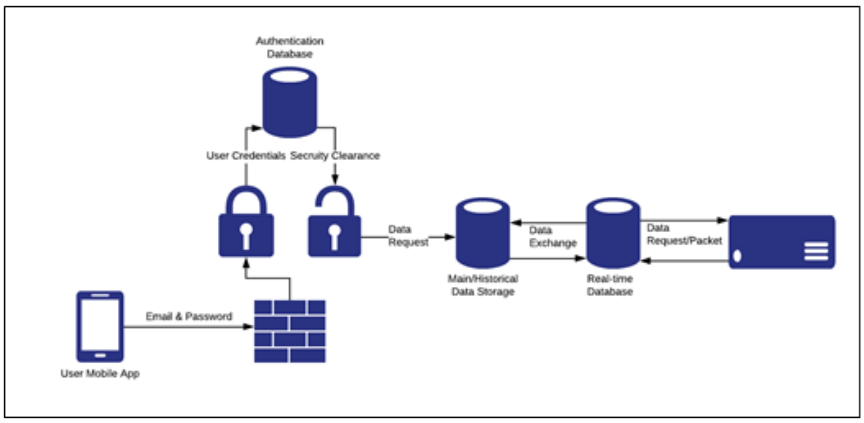


Figure 19: Database Diagram.

The testing of the database is achieved in software by establishing test connections and dummy objects. Overall, the database is considerably robust given reliable internet connect. As the result, database testing is not a huge concern to the product's success.

User Application

We provide the user with a Mobile App for their Android devices as the main interaction window with the rest of our product. The following figure shows the layout of the App.

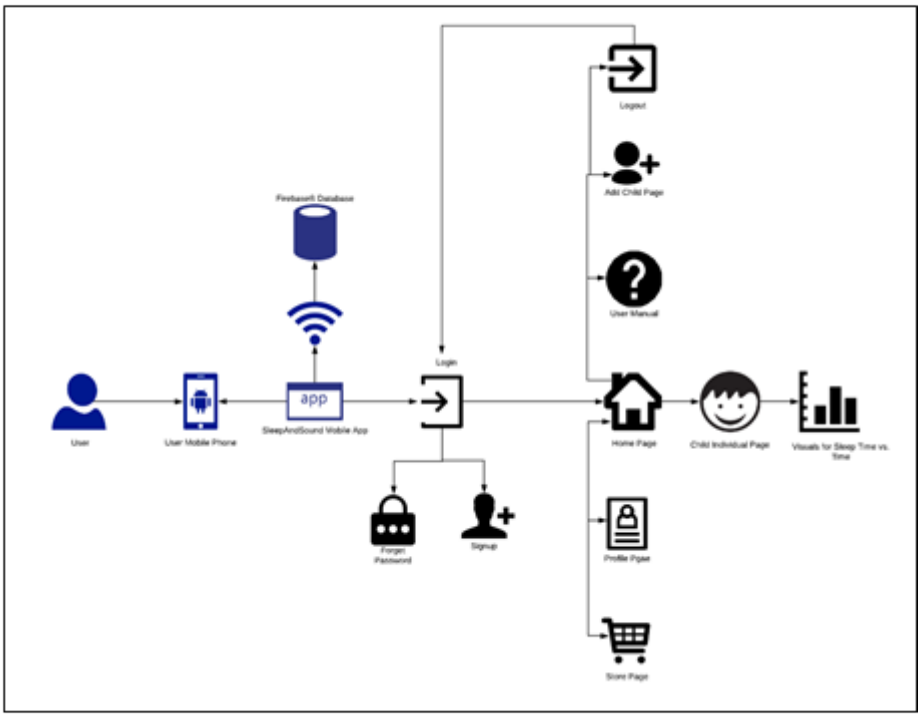


Figure 20: Mobile App Diagram.

When first start up the App, the user will be asked to sign in. He or she can also go to 'forget password' or 'signup' from the login page. Once signed in, the user is brought to the App's home page where the list of registered children is displayed. On this page, we also show user some thumbnail information such as children's name, average sleeping hours, and last-seen-in-bed time. The children list is also colour coded base on the children's current status (in-bed or not, is the bedroom dark or lighted). By clicking on a cardview of the children, the user can view more detail information about the selected child's sleeping patter over time.

The user can navigate the App through the drawer menu, from which they can go to Profile Page, Store Page, User Manual, Add Child Page or logout. The testing of the software is achieved through test classes in software and runs constant as we develop the software.

System Test and Integration

In this section of the report we will discuss how we will integrate each module into our system. This explanation has two sections, the hardware integration, and the software integrations. Each section contains information on how we will integrate each module and how we will test to see if the module is integrated correctly.

Hardware Module Integration

The general integration of the hardware was described on a module level in the module description section of this report. The system level integration is different because it involves integrating all the modules and testing to see if they work together. To do this we will write a software code to test the hardware. This script will mainly be for the central hub to request data from all the sensor modules. We will test each sensor twice. The first time we will let the natural state of the sensor be recorded. The second time we will physically change the environment of each sensor. If the transmitter receives the same data twice then we know that the hardware isn't integrated correctly. If we are able to receive a variety of data from each sensor and record this data into the database, then the hardware integration part of this project is complete.

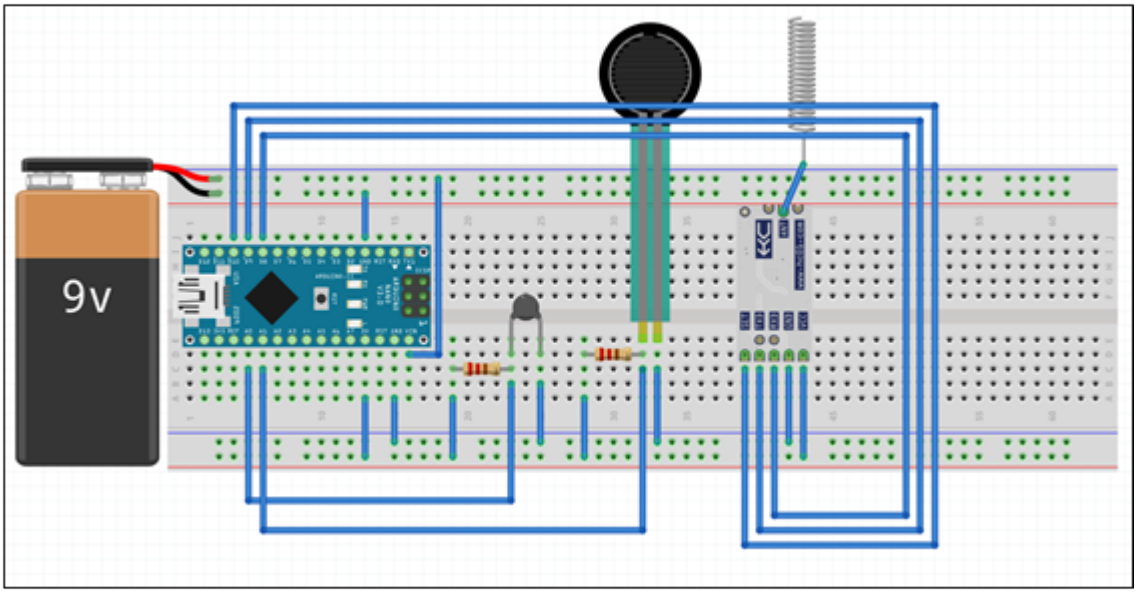


Figure 21: Bed Module circuit diagram.

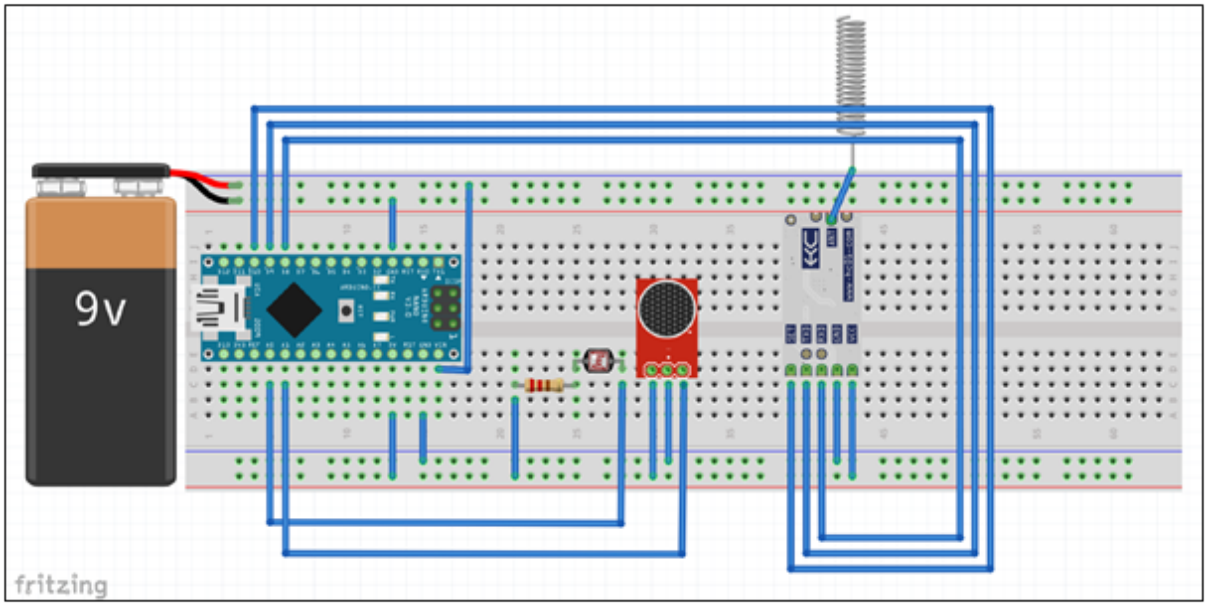


Figure 22: Headboard Module circuit diagram.

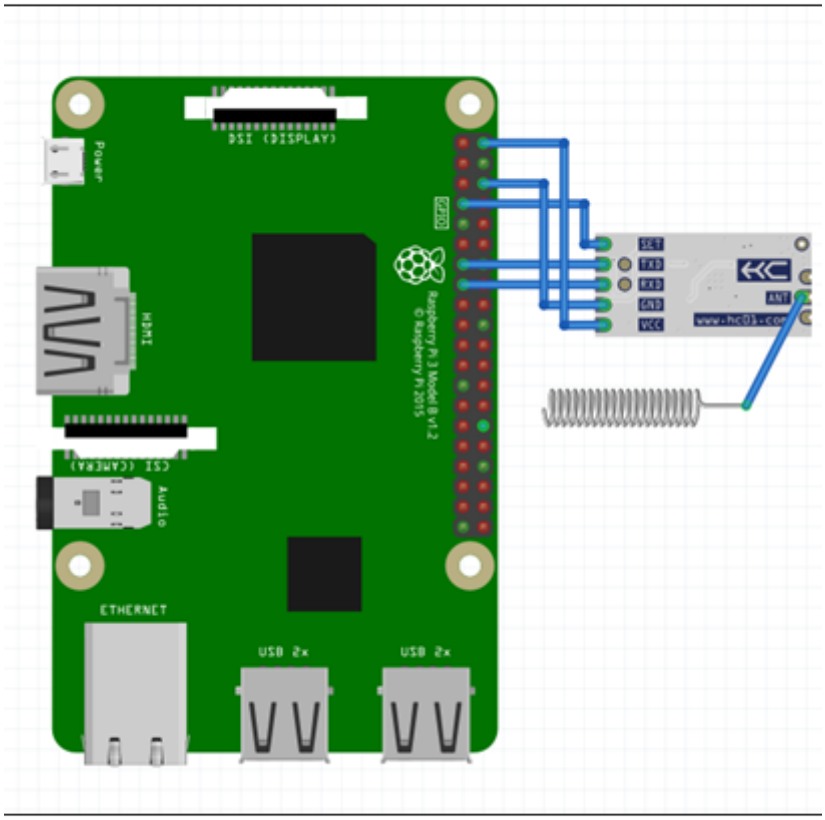


Figure 23: Central Hub circuit diagram.

These figures show how the system will be physically implemented using components, wires and pinouts. The pinouts may change overtime, however, the general idea is described with these images. The schematics are located in the schematics section of this report.

Software Module Integration

The database and the user app are integrated through JavaScript Object Notation (JSON), a commonly used data-interchange format. The data sent to the database by hardware is organized in tables under user_x/children/child_x, then parsed to the user's Mobile App for further visualization.

Product Results

Product Improvement

The headboard/bedstand sensor module from our original design contained both the light sensor and the microphone. The light sensor is for collecting the ambient condition near the bed to know if the room's condition is the best for the children's sleep. The microphone will detect if there are other sounds in the room apart from breathing and snoring that might interfere with the correct room environment for the children to have a good night's sleep. We chose the Sodial Microphone Sensor with High Sensitivity shown in the figure below but have not received it from our provider.

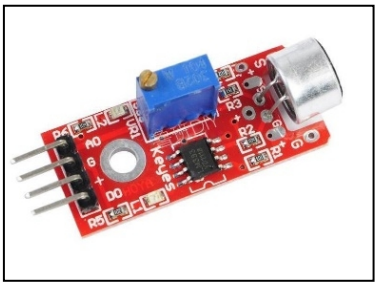


Figure 24: Sodial Microphone Sensor with High Sensitivity Sound Detection Module

Our plan is to implement the microphone sensor if we receive it on time prior to the final demonstration of a working prototype; however, the main ability of our product to track sleep quality of a children has been achieved mainly by the pressure sensors and light sensor which serve as a two-factor validation to check if the human subject is in bed. Including the microphone in our product will serve as a complement to the already existing and working prototype.

One of the customer requirements for our product was to make it as less intrusive as possible for the children being monitored, so he or she might not even notice that they are being monitored by their parents. Since the under-mattress module consisting of the pressure sensors and the temperature sensor go under the mattress and under the sheets respectively, there was no need to worry about these sensors as they will not be in plain sight. The headboard/ bedstand module containing the light sensor and microphone can be placed anywhere in the room where it is still effective, but the child does not notice the sensors. To fulfil this and to make the device more appealing to the eye, we came up with the following 3D printed casing to put the light sensor and microphone along with a special Printed Circuit Board, Arduino microcontroller, and transceiver inside the 3D printed object.

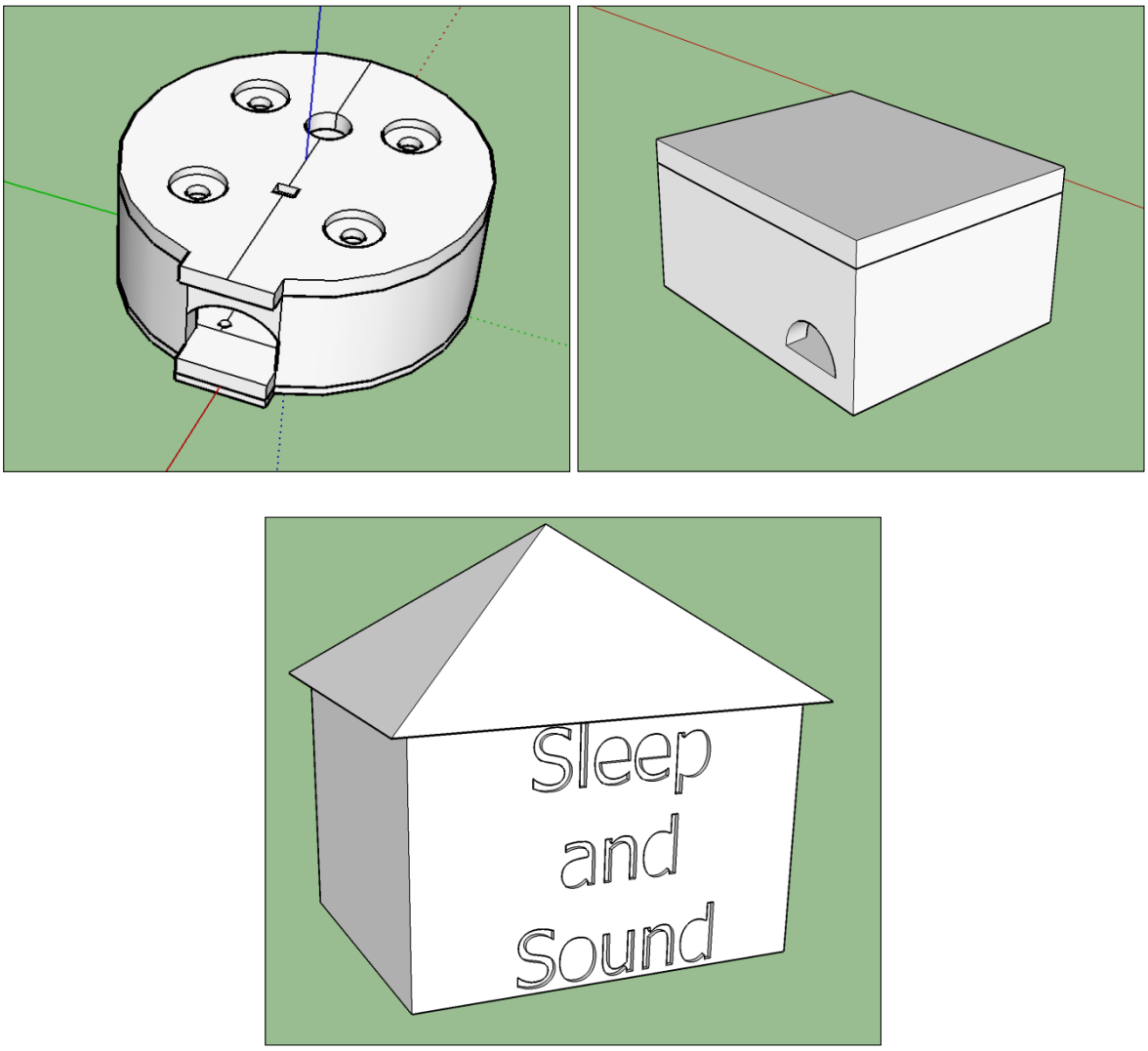


Figure 25: Artist Rendition of 3D Printed Object

Because of its appearance similar to a toy, the object can be placed on the headboard or nightstand without it being intrusive for the person in bed. If placed on the headboard, the light sensor will also be able to detect whenever the child is using electronics in bed along with the room environment to make sure it is ideal for good quality sleep.

In order to get rid of the breadboard containing the headboard/ bedstand module where we had our light sensor connected to the Arduino microcontroller along with a transceiver, we created a Printed Circuit Board that will allow signals and power to be routed between connectors and components and can be modularized with multiple sensors. The purpose for this PCB is to implement our actual product where no breadboard is necessary for the sensor modules. This PCB will also be easier to implement inside the 3D printed object that will hide the headboard/bedstand module. The following picture shows the design of the circuit board we printed.

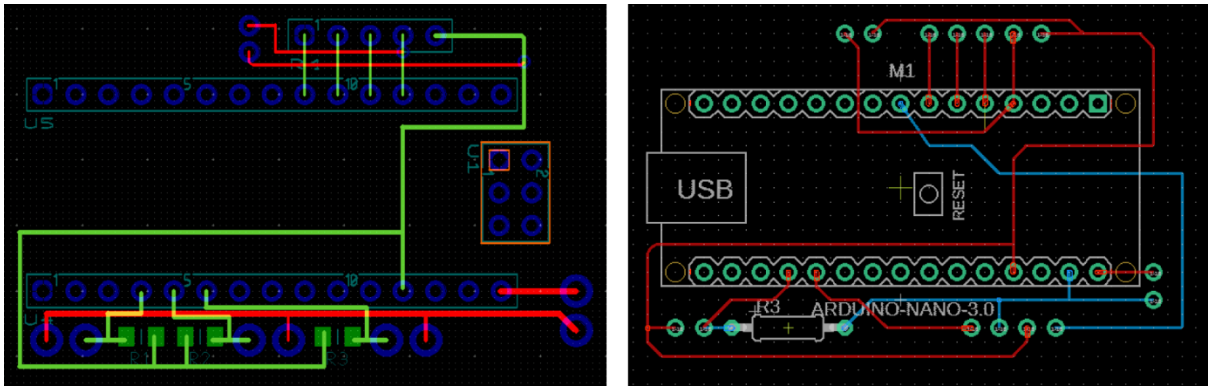


Figure 26: Design of PCB for Sensor Modules

Another improvement that we have planned for our sensor modules if we were to take our device into the competitive market is to break up the modules and use additional circuit boards to test the device using battery power instead of USB power. By using these 9V batteries to power the Arduino microcontrollers, we can guarantee that our product will be able to run for weeks without battery replacement.

Central Hub Module

The central hub module in our product also had many changes and improvements done to it, especially the frontend user application. Similar to the headboard/bedstand module, a 3D printed object will also be needed as a casing for the Raspberry Pi microcontroller. This will guarantee that even this module which does not necessarily have to be placed near the bed will not be intrusive for the person in bed.

Safety is an important topic, especially when talking about data regarding the children of parents. After recommendations from our Teaching Assistants, we implemented a user authentication in the app where the parent will create their profile as a new user. They can sign out of their profile that has the sleep information of their children whenever they would like to sign out. If the user were to forget his or her username and password, retrieval of a new password is possible through the use of a link sent to the email with which the user first created the profile. The following pictures show the Application User Authentication Page.

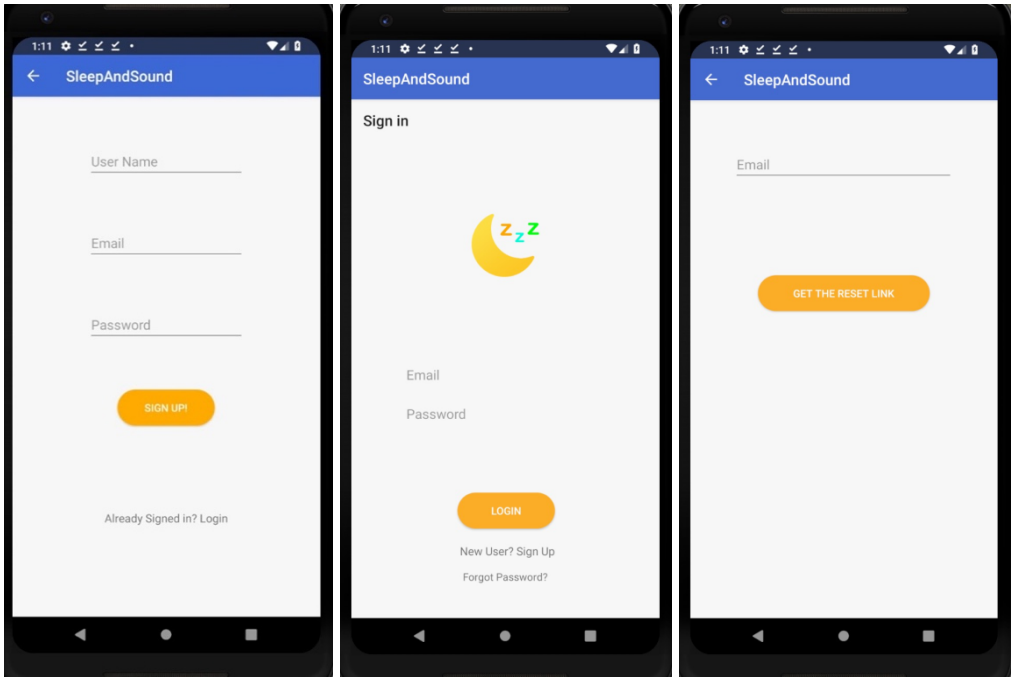


Figure 27: Left: Signup, Middle: Login, Right: Retrieve Password

Our previous approach is, in the central hub's onboard memory, generate a CSV file in which all captured data resides. We have altered this design to provide the customer with a better experience. Instead of having an intermediate CSV file sitting in between the user and the database, the MCU now send data to the database directly. Although we must structure the database architecture more carefully to ensure the data get stored to the right node, this way we can significantly reduce the system overhead and in turn provide a better user experience.

As for the final App, on top of viewing real-time data to see if the child is currently in bed or the lighting and noise condition in the room, the user is able to choose different dates and view that day's data. Moreover, the App provides the user with data visualization so that they may learn from a glance the sleep pattern of the subject under monitoring.

To further the capacity of the design, the team implemented a product registration protocol. When a product is boot up for the first time, it would register itself on the database. However, the device still needs to be associated with a human subject (i.e., a child) to function. Through the App, the user can pull the listed of registered but un-associated hardware and associated with the kid in a one-to-one relation. By this, using a single App, the user can have multiple monitor systems being subjected to various persons. Each child can also their name, age, and picture in the app edited.

The following figures show the various App activities:

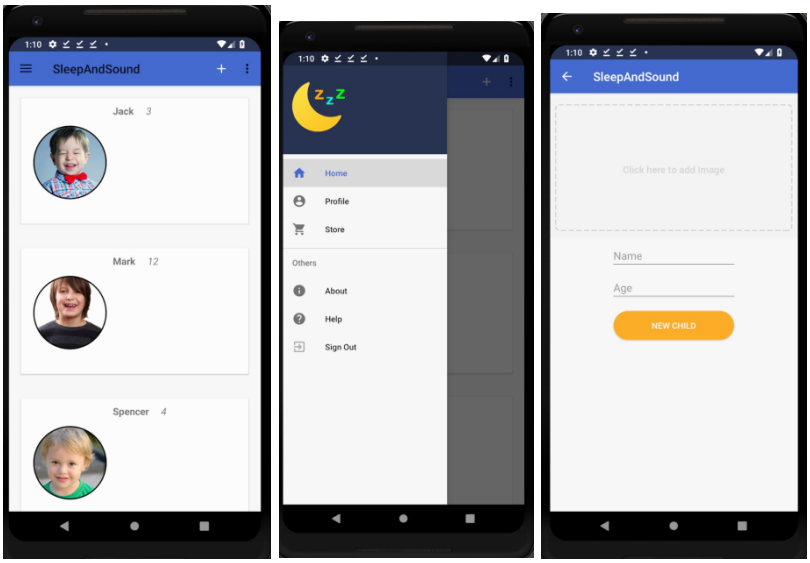


Figure 28: Left: Home View, Middle: Navigation Drawer, Right: Add Child

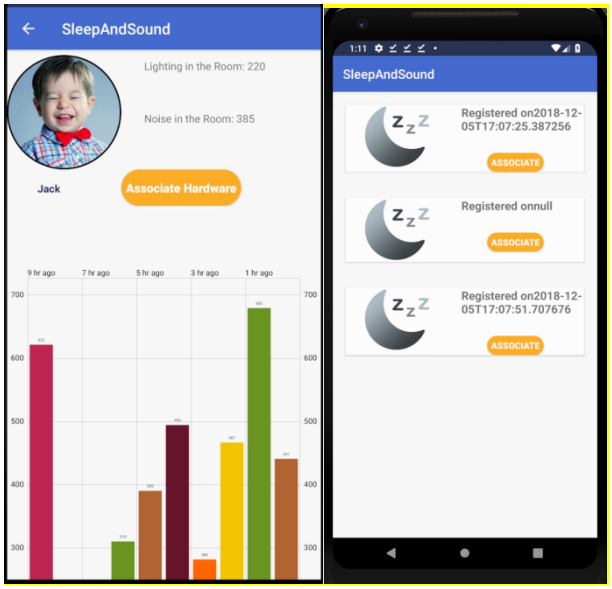


Figure 29: Left: Child View, Right: Associate Hardware

Instead of a bar graph, our plan is to have a line graph for the final demonstration of a working prototype in order to give better and more accurate data to the user. This line graph over time will tell users whenever the child was not in bed according to the pressure sensors and the temperature sensor. Room environment based on the light sensor and the microphone is another feature available for users.

Cost Analysis

Business Analysis

As previously noted, our target market is approximately 16M families in the Untied States that have children under the age of 12. With this we can determine that if we want to productize for this market, we will have an initial fixed cost of $150,000 with a cost per unit for production of $16. This is due to the fact that all our microcontrollers and sensors are very inexpensive, especially if mass produced.

From this we set our MSRP at $50. This gives us a return on investment of $34 per unit sold.

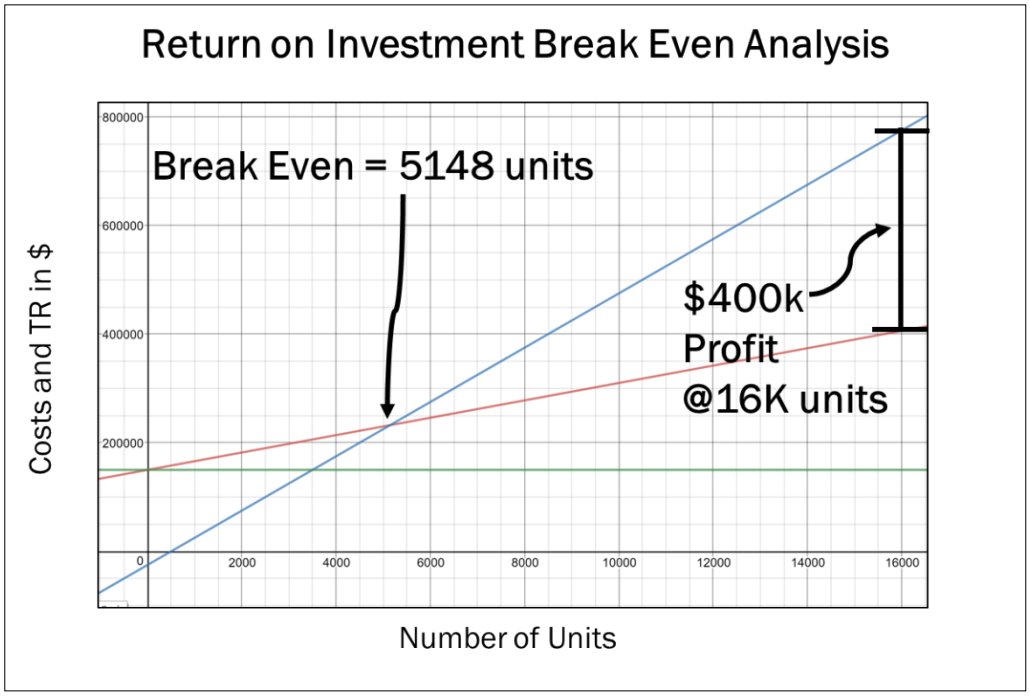


Figure 30: Break even analysis.

Some notable details from this graph.

1. We must sell 5148 units before we will breakeven.
2. At 16000 units, we will have $400k in profit.
3. If we sell to 10% of our market, we will sell 1.6M units, which equates to $53.25 million in profit.
    a. This is reasonable especially considering we are priced at 50% less than all competing devices.

To look at different way of advertising, we are considering selling this product in retail stores such as Bed Bath and Beyond, Best Buy, and mattress companies selling child

mattresses. We also plan on selling online with those retailor as well as with Amazon and QVC. The image below shows a Bed Bath and Beyond online sleep monitor product search.

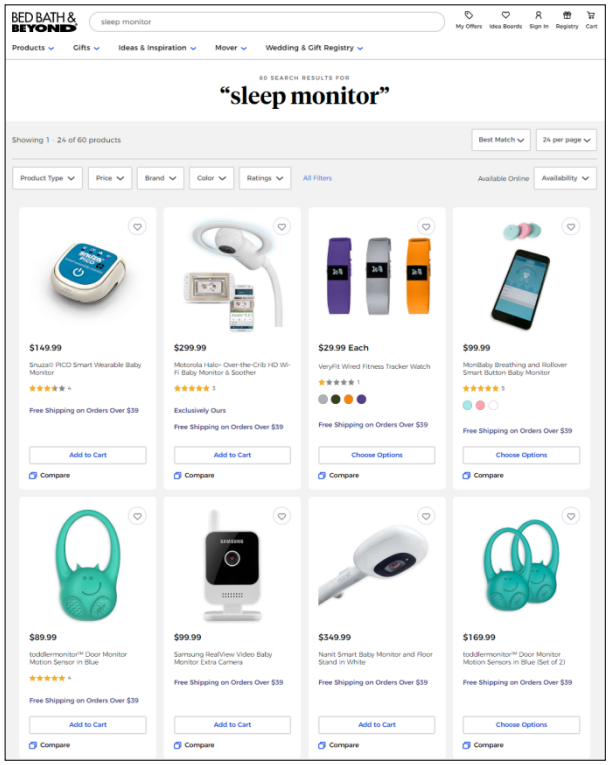


Figure 31: Bed Bath and Beyond Sleep Monitor search.

Clearly, Bed Bath and Beyond already ahs a product base in the same category that we would be selling in, therefore our product will fit in nicely.

We also hope to sell in magazines during the holidays from our distribution channels, in Facebook ads, and potentially family sit-com TV channels.

From a risk factor analysis, we have a few strategic risks that are really true for all products.

1. Minimal customer acceptance
2. Poor Marketing
3. Low percent yield on product production

These risk factors need to be considered, but overall they are negligible towards the product production and selling points.

Conclusions

In the future, we hope to improve our product to be more like a sellable product and not just a prototype. Some things we will improve, are the user interface quality for our application, the customizability of the sensor module objects, and the overall design of how wires are exposed from the sensor modules. We need to have more wire protection so that customers don't actually rip the wires out of our sensor modules. We might further look at power to our sensor modules and make our headboard and under mattress modules battery powered.

Appendix A: Code Repository

Python Script on RP

```
#!/usr/bin/env python3
import random
import time
import datetime
import datetime
import serial
import csv
import uuid;
import os.path
from pathlib import Path
from firebase import firebase

#index string
index = ["hour0", "hour1", "hour2", "hour3", "hour4", "hour5", "hour6", "hour7", "hour8",
"hour9"]
CALIBRATE_REQUESTS = 1
SLEEP_VAL = 1
REQUEST_RATE = 1
modules = ["ps", "ts", "ls", "mp"]

SOUND_STATUS = "normal" #normal, loud
LIGHT_STATUS = "on" #on, off
IN_BED_STATUS = "out of bed" #out of bed, in bed

ps_list = [-1]*100;
ts_list = [-1]*100;
ls_list = [-1]*100;
mp_list = [-1]*100;
list_index = 0;

## establisig serial connection to RF transceiver
print("Establisig serial connection to RF transceiver")
ser = serial.Serial(
        port='/dev/ttyS0',
        baudrate = 9600,
        parity = serial.PARITY_NONE,
        stopbits = serial.STOPBITS_ONE,
        bytesize=serial.EIGHTBITS,
        timeout=1
        )
print("Serial Connection Established")


print("Connecting to S&S server...")
firebase = firebase.FirebaseApplication('https://sleepandsound-d1073.firebaseio.com/', None)
print("Successfully connected to S&S server")
##result = firebase.get('/Users',None)
```

```
##print (result)


# check if there's a product key in local directory
if not os.path.isfile("product_id.txt"):
    file_pid = open("product_id.txt", "w+")
    product_id = str(uuid.uuid4())
    file_pid.write(product_id )
    print ("new S&S-Kid ID generated: ", product_id)
else:
    file_pid = open("product_id.txt", "r")
    product_id = file_pid.read(36)
    print("S&S-Kid ID exists: ", product_id)

# check if the product is registered on the database
print("Connecting to Database")
if (firebase.get('/hardwares', product_id) == None):
    print("S&S-Kid not registered, registering...")
    firebase.put('/hardwares/'+product_id, "pid", product_id)
    firebase.put('/hardwares/'+product_id, "timestamp", datetime.datetime.now())
    print("Setting S&S-Kid status to be 'UNASSOCIATED'...")
    firebase.put('/hardwares/'+product_id, "status", 'UNASSOCIATED')
    print("Successfully set S&S-Kid status to be 'UNASSOCIATED'")
else: print("S&S-Kid already registered")

## update product status
##if(firebase.get('/hardwares', product_id) == None)
#if((firebase.get('/hardwares/'+product_id, "status") == "UNASSOCIATED")):

##if((firebase.get('/hardwares/'+product_id, "status") == "ASSOCIATED")):
##        print("Product assocaited")
##else:
##        print("Product not associated")
##        #exit(0)

ps_none = 0
ts_none = 0
ls_none = 0
mp_none = 0

#take CALIBRATE_REQUESTS samples per sensor and calibrate midpoint of all
def calibrate():
    print("---------------------------------------")
    print("Calibrating Sensor Network")

    global ps_none
    global ts_none
    global ls_none
    global mp_none
    for i in range(CALIBRATE_REQUESTS):
```

```
        for mod in modules:
            ser.write(mod.encode())
            print(mod)
            time.sleep(SLEEP_VAL)
            if(mod == "ps"):
                ps_none = ps_none + int(read_data())
            elif(mod == "ts"):
                ts_none = ls_none + int(read_data())
            elif(mod == "ls"):
                ls_none = ls_none + int(read_data())
            else:
                mp_none = mp_none + int(read_data())
        time.sleep(REQUEST_RATE)
    ps_none = ps_none / CALIBRATE_REQUESTS
    ts_none = ts_none / CALIBRATE_REQUESTS
    ls_none = ls_none / CALIBRATE_REQUESTS
    mp_none = mp_none / CALIBRATE_REQUESTS

#change state of room
def change_state():
    global IN_BED_STATUS
    global LIGHT_STATUS
    global SOUND_STATUS
    ps_average = ps_none; ps_average_new = 0;
    ts_average = ts_none; ts_average_new = 0;
    ls_average = ls_none; ls_average_new = 0;
    mp_average = mp_none; mp_average_new = 0;

    ps_count = 0;
    ts_count = 0;
    ls_count = 0;
    mp_count = 0;

    for i in reversed(range(0, 101)):
        index = i + list_index
        if(index > 99):
            index = index - 100
        #if we want to do more datprocessing this creates full average
        if(i >= 0 and i < 100):
            if(ps_list[index] != -1):
                ps_average = (ps_average + ps_list[index])/2
            if(ts_list[index] != -1):
                ts_average = (ts_average + ts_list[index])/2
            if(ls_list[index] != -1):
                ls_average = (ls_average + ls_list[index])/2
            if(mp_list[index] != -1):
                mp_average = (mp_average + mp_list[index])/2

        if(i > 100-CALIBRATE_REQUESTS):
            if(ps_list[index] != -1):
```

```
            ps_average_new = ps_average_new + ps_list[index]
            ps_count+=1
        if(ts_list[index] != -1):
            ts_average_new = ts_average_new + ts_list[index]
            ts_count+=1
        if(ls_list[index] != -1):
            ls_average_new = ls_average_new + ls_list[index]
            ls_count+=1
        if(mp_list[index] != -1):
            mp_average_new = mp_average_new + mp_list[index]
            mp_count+=1


    if(ps_count > 0):
        ps_average_new /= ps_count
    if(ts_count > 0):
        ts_average_new /= ts_count
    if(ls_count > 0):
        ls_average_new /= ls_count
    if(mp_count > 0):
        mp_average_new /= mp_count

  #change in bed status
  if(ps_average_new - ps_none > 250 and ts_average_new - ts_none > -100):
    IN_BED_STATUS = "in bed"
  else:
    IN_BED_STATUS = "out of bed"

  #change light status
  if(ls_none - ls_average_new > 200):
    LIGHT_STATUS = "off"
  else:
    LIGHT_STATUS = "on"

  #chagne sound status
  if(mp_average_new - mp_none > 5):
    SOUND_STATUS = "loud"
  else:
    SOUND_STATUS = "normal"


def read_data():
  x=ser.readline()
  if len(x) > 1 and ':' in x.decode():
        print("\t" + str(x.decode()))
        temp_val = x.decode().split(":")[1]
        try:
            return int(x.decode().split(':')[1].replace("", ").replace(' ', ").replace('\n', "))
        except:
            return -1;
```

```
    else:
        print("\tno data received")
    return -1

calibrate()

# start data cycle
if not os.path.isfile("dayHead.txt"):
    file_dayHead = open("dayHead.txt", "w+")
    file_dayHead.write(str(0))
    file_dayHead.close()
    firebase.put('/hardwares/'+product_id+'/data', 'historyHourHead', str(0))
    file_hourHead = open("hourHead.txt", "w+")
    file_hourHead.write(str(0))
    file_hourHead.close()
hourFile = open("hourHead.txt", "r")
dayFile = open("dayHead.txt", "r")
hour_of_day = int(hourFile.read(2))
day = int(dayFile.read(2))
hourFile.close()
dayFile.close()
while(1):
    for i_string in index:
        print("----------------------------------------")
        for mod in modules:
            ser.write(mod.encode())
            print(mod)
            time.sleep(SLEEP_VAL)

            if(mod == "ps"):
                ps = read_data()
                ps_list[list_index] = ps
            elif(mod == "ts"):
                ts = read_data()
                ts_list[list_index] = ts
            elif(mod == "ls"):
                ls = read_data()
                ls_list[list_index] = ls
            else:
                mp = read_data()
                mp_list[list_index] = mp

        change_state()

        if list_index == 99:
            list_index = 0
        else:
            list_index = list_index + 1

        print("Status")
```

```
        print("In Bed Status: ", IN_BED_STATUS)
        print("Light Status: ", LIGHT_STATUS)
        print("Noise Volume: ", SOUND_STATUS)
        print(" “)
        ## firebase upload
        if((firebase.get('/hardwares/'+product_id, "status") != "UNASSOCIATED")):
            print("Product assocaited, uploading data")
            #real-time data
            firebase.put('/hardwares/'+product_id+'/data'+'/now', 'bed', IN_BED_STATUS)
            firebase.put('/hardwares/'+product_id+'/data'+'/now', 'light', LIGHT_STATUS)
            firebase.put('/hardwares/'+product_id+'/data'+'/now', 'noise', SOUND_STATUS)


            # HISTORY DATA
            print("Day: " + str(day) + " ; " + "Hour: " + str(hour_of_day))
            firebase.put('/hardwares/'+product_id+'/data', 'historyDayHead', str(day))
            file_dayHead = open("dayHead.txt", "w+")
            file_dayHead.write(str(day))
            file_dayHead.close()
            firebase.put('/hardwares/'+product_id+'/data', 'historyHourHead', str(hour_of_day))
            file_hourHead = open("hourHead.txt", "w+")
            file_hourHead.write(str(hour_of_day))
            file_hourHead.close()
            firebase.put('/hardwares/'+product_id+'/data'+'/history/' + str(day) + '/' +
str(hour_of_day), 'bed', IN_BED_STATUS)
            firebase.put('/hardwares/'+product_id+'/data'+'/history/' + str(day) + '/' +
str(hour_of_day), 'light', LIGHT_STATUS)
            firebase.put('/hardwares/'+product_id+'/data'+'/history/' + str(day) + '/' +
str(hour_of_day), 'noise', SOUND_STATUS)
##            datetime.datetime.now()

            if(hour_of_day < 23):
                hour_of_day = hour_of_day + 1
            else:
                hour_of_day = 0
                if(day < 89):
                    day = day + 1
                else:
                    day = 0
        else:
            print("Product not associated, skipping upload")


##        if ps != -1:
##            print("upload pressure data (averaged): ", ps)
##            firebase.put('/hardwares/'+product_id+'/ptData'+'/'+i_string, 'pres', ps)
##        if ts != -1:
##            print("upload temperature data: ", ts)
##            firebase.put('/hardwares/'+product_id+'/ptData'+'/'+i_string, 'temp', ts)
##        if ls != -1:
```

```
##            print("upload light sensor data: ", ls)
##            firebase.put('/hardwares/'+product_id, "LS_data", ls)
##        if mp != -1:
##            print("upload microphone data: ", mp)
##            firebase.put('/hardwares/'+product_id, "MCP_data", mp)
##
      time.sleep(REQUEST_RATE)
```

Under Mattress Code

```
#include <SoftwareSerial.h>

#include <math.h>


const byte HC12RxdPin = 2;                  // Recieve Pin on HC12

const byte HC12TxdPin = 3;                  // Transmit Pin on HC12

const byte HC12SetPin = 4;                 // Set Pin on HC12


SoftwareSerial HC12(HC12TxdPin, HC12RxdPin); // Create Software Serial Port


int fsr1AnalogPin = 0;

int fsr2AnalogPin = 1;

int tsAnalogPin = 2;


int read_or_write_fsr = 0;

int read_or_write_ts = 0;

int read_or_write = 0; // 0 - read, 1 - write



void setup() {

  Serial.begin(9600);                    // Open serial port to computer

  HC12.begin(9600);                       // Open serial port to HC12

}
```

```
int convert_data(int reading) {
  float fsrResistance;
  float temp;
  float fsrConductance;
  float fsrForce;

  reading = map(reading, 0, 1023, 0, 5000);
  fsrResistance = 5000 - reading;     // fsrVoltage is in millivolts so 5V = 5000mV
  temp = (float)10000 / (float)reading;
  fsrResistance *= temp;                // 10K resistor

  Serial.print(fsrResistance);

  fsrConductance = 1000000;           // we measure in micromhos so
  fsrConductance /= fsrResistance;
  // Use the two FSR guide graphs to approximate the force
  if (fsrConductance <= 1000) {
    fsrForce = fsrConductance / 80;
  }
  else {
    fsrForce = fsrConductance - 1000;
    fsrForce /= 30;
  }
  return (int)fsrForce;
}

int Thermister(int RawADC) {  // Steinhart-Hart equation
  double Temp;
  Temp = log(((10240000 / RawADC) - 10000));
  Temp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * Temp * Temp )) * Temp
);
```

```
  Temp = Temp - 273.15;             // Convert Kelvin to Celsius
  Temp = (Temp * 9.0) / 5.0 + 32.0; // Celsius to Fahrenheit - comment out this line if you need Celsius
  return (int)Temp;
}

void read_data() {
  String HC12ReadBuffer = "";
  while (HC12.available()) {
    HC12ReadBuffer += (char)HC12.read();
    delay(1);
  }
  if (HC12ReadBuffer.length() > 0) {
    Serial.println(HC12ReadBuffer);
    if (HC12ReadBuffer == "ps") {
      read_or_write_fsr = 1;
      read_or_write = 1;
    }
    else if (HC12ReadBuffer == "ts") {
      read_or_write_ts = 1;
      read_or_write = 1;
    }
    else {
      Serial.println("\tUknownCommand");
    }
  }
}

void write_data() {
  long data_reading = 0;
  int i;
```

```
 if (read_or_write_fsr > 0) {
  for(i = 0; i < 50; i++){
   long data_reading_temp = (analogRead(fsr1AnalogPin) + analogRead(fsr2AnalogPin))/2;
   data_reading += data_reading_temp;
  }
  data_reading = data_reading / 50;

  //int data_reading = convert_data(data_reading);
  Serial.print("Pressure Sensor Data: ");
  Serial.println(data_reading);

  read_or_write_fsr = 0;
  char temp [50];
  sprintf(temp, "Pressure Sensor Data: %i \n", data_reading);
  HC12.write(temp);
 }

 if (read_or_write_ts > 0) {
  for(i = 0; i < 25; i++){
   data_reading += analogRead(tsAnalogPin);
  }
  data_reading = data_reading / 25;
  //data_reading = Thermister(data_reading);
  Serial.print("Temperature Sensor Data: ");
  Serial.println(data_reading);
  read_or_write_ts = 0;
  char temp [50];
  sprintf(temp, "Temperature Sensor Data: %i \n", data_reading);
  HC12.write(temp);
 }
```

```
}

void loop() {
  if (read_or_write == 0) { //read
    read_data();
  }
  else {
    write_data();
    read_or_write = 0;
  }
}
```

Headboard Module Code

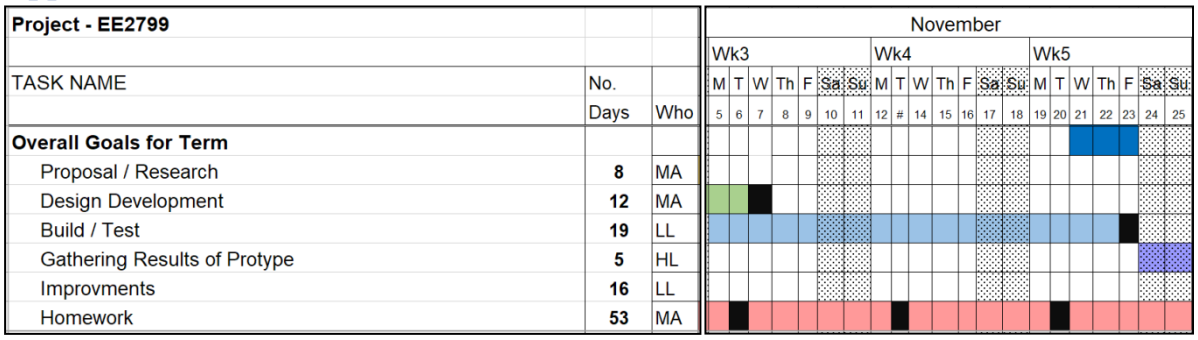
```
#include <SoftwareSerial.h>

const byte HC12RxdPin = 2;                  // Recieve Pin on HC12
const byte HC12TxdPin = 3;                  // Transmit Pin on HC12
const byte HC12SetPin = 4;                  // Set Pin on HC12

SoftwareSerial HC12(HC12TxdPin,HC12RxdPin); // Create Software Serial Port

int lsAnalogPin = 0;
int mpAnalogPin = 1;

int read_or_write_ls = 0;
int read_or_write_mp = 0;
int read_or_write = 0; // 0 - read, 1 - write

void setup() {
  Serial.begin(9600);                       // Open serial port to computer
```

```
  HC12.begin(9600);                   // Open serial port to HC12
}

void read_data(){
  String HC12ReadBuffer = "";
  while(HC12.available()){
    HC12ReadBuffer += (char)HC12.read();
    delay(1);
  }
  if (HC12ReadBuffer.length() > 0){
    Serial.println(HC12ReadBuffer);
    if(HC12ReadBuffer == "ls"){
      read_or_write_ls = 1;
      read_or_write = 1;
    }
    else if(HC12ReadBuffer == "mp"){
      read_or_write_mp = 1;
      read_or_write = 1;
    }
    else{
      Serial.println("\tUknownCommand");
    }
  }
}

void write_data(){
  int data_reading = 0;
  int i;

  if(read_or_write_ls > 0){
```

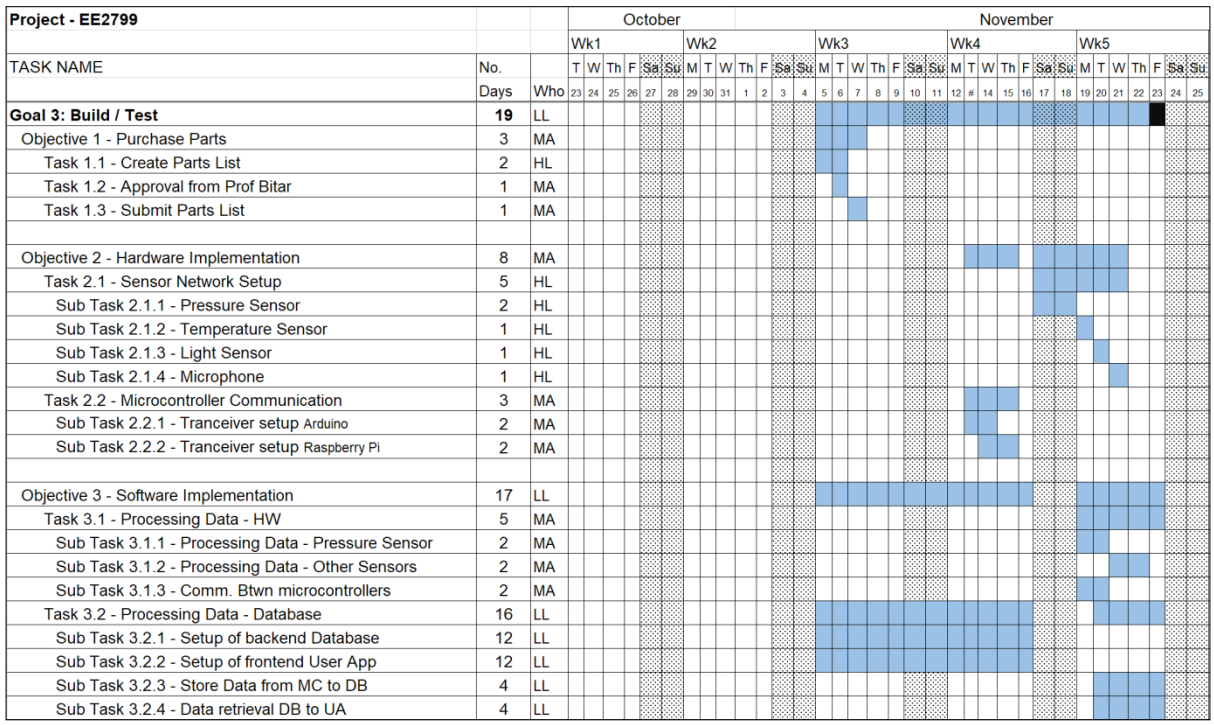
```
  for(i = 0; i < 25; i++){ //average last 25 data readings
    data_reading += analogRead(lsAnalogPin);
  }
  data_reading = data_reading / 25;

  Serial.print("Light Sensor Data: ");
  Serial.println(data_reading);
  read_or_write_ls = 0;

  char temp [50];
  sprintf(temp, "Light Sensor Data: %i \n", data_reading);
  HC12.write(temp);
}

if(read_or_write_mp > 0){
  int max_mp = 0;
  for(i = 0; i < 100; i++){
    data_reading = analogRead(mpAnalogPin);
    if(max_mp < data_reading){
      max_mp = data_reading;
    }
  }
  data_reading = max_mp;

  Serial.print("Microphone Data: ");
  Serial.println(data_reading);
  read_or_write_mp = 0;

  char temp [50];
  sprintf(temp, "Microphone Data: %i \n", data_reading);
```

```
    HC12.write(temp);
  }
}

void loop() {
  if(read_or_write == 0){ //read
    read_data();
  }
  else{
    write_data();
    read_or_write = 0;
  }
}
```

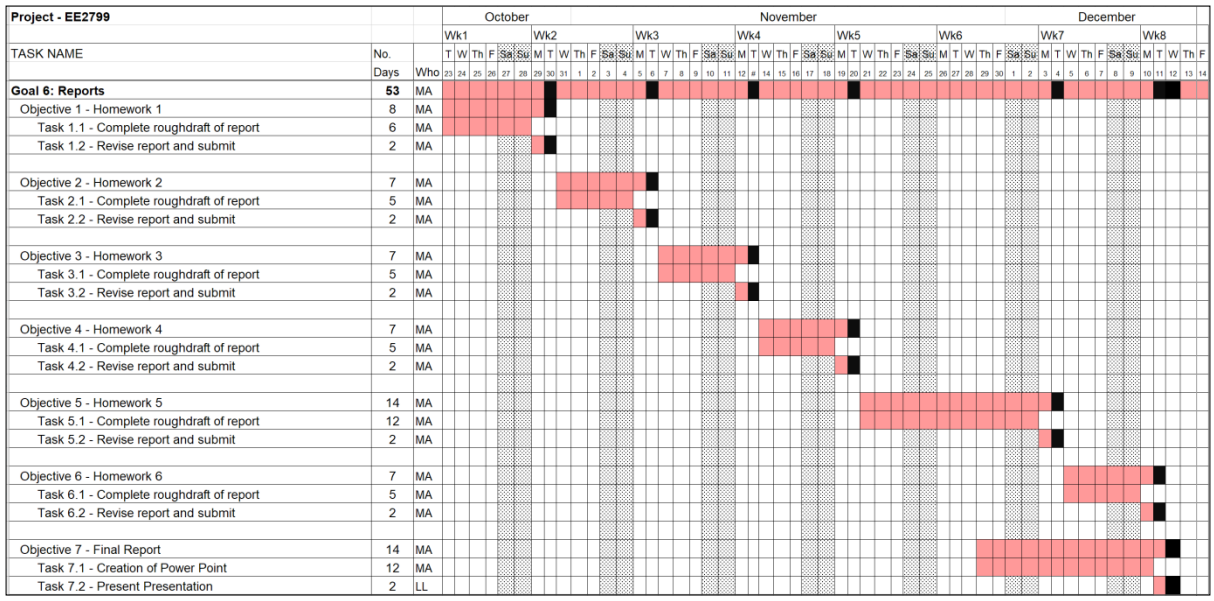
Appendix B: Gantt Chart



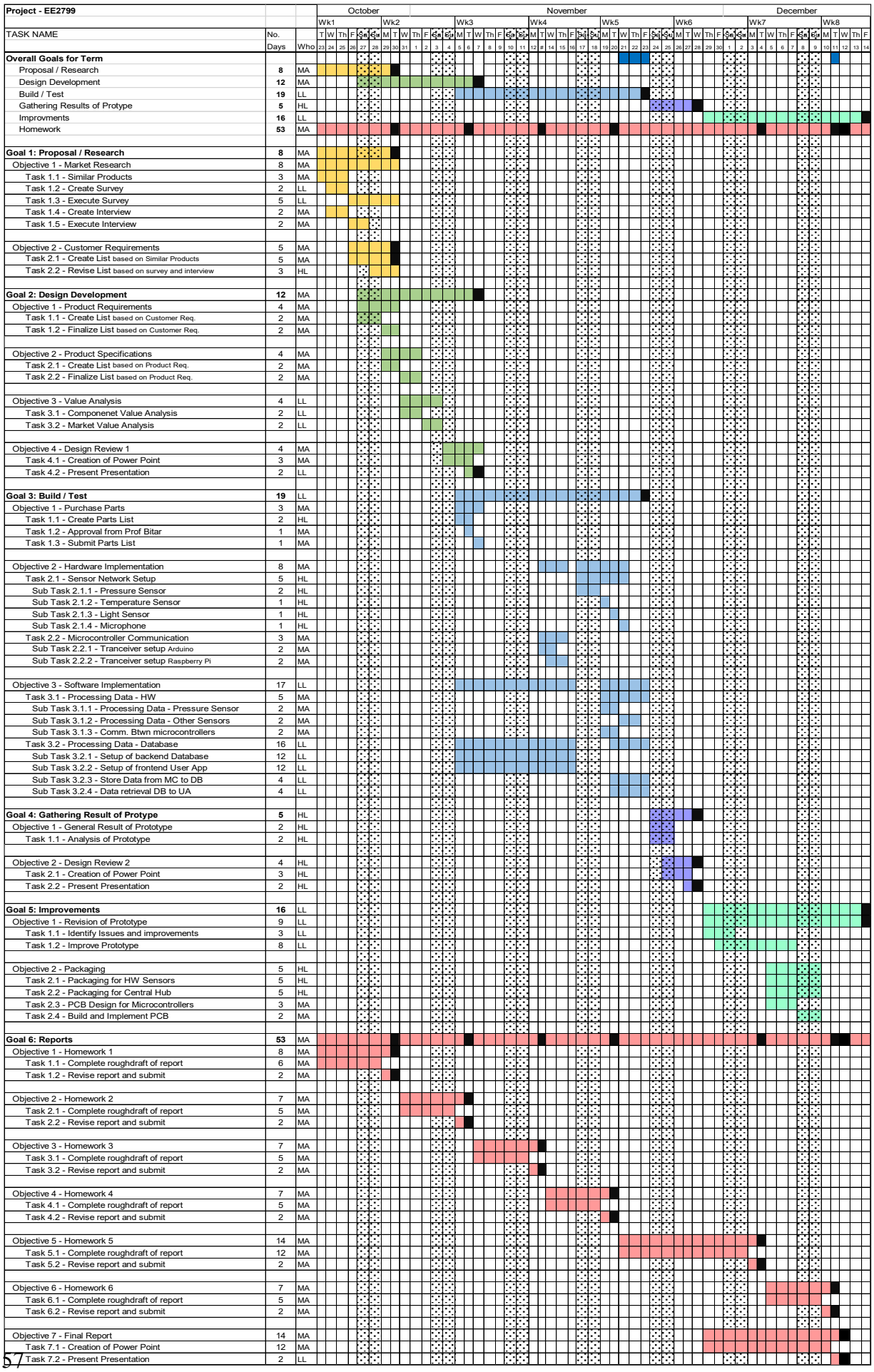
Project - EE2799	No. Days	Who
Overall Goals for Term		
Proposal / Research	**8**	MA
Design Development	**12**	MA
Build / Test	**19**	LL
Gathering Results of Protype	**5**	HL
Improvments	**16**	LL
Homework	**53**	MA



Project - EE2799	No. Days	Who
Goal 3: Build / Test	**19**	LL
Objective 1 - Purchase Parts	3	MA
Task 1.1 - Create Parts List	2	HL
Task 1.2 - Approval from Prof Bitar	1	MA
Task 1.3 - Submit Parts List	1	MA
Objective 2 - Hardware Implementation	8	MA
Task 2.1 - Sensor Network Setup	5	HL
Sub Task 2.1.1 - Pressure Sensor	2	HL
Sub Task 2.1.2 - Temperature Sensor	1	HL
Sub Task 2.1.3 - Light Sensor	1	HL
Sub Task 2.1.4 - Microphone	1	HL
Task 2.2 - Microcontroller Communication	3	MA
Sub Task 2.2.1 - Tranceiver setup Arduino	2	MA
Sub Task 2.2.2 - Tranceiver setup Raspberry Pi	2	MA
Objective 3 - Software Implementation	17	LL
Task 3.1 - Processing Data - HW	5	MA
Sub Task 3.1.1 - Processing Data - Pressure Sensor	2	MA
Sub Task 3.1.2 - Processing Data - Other Sensors	2	MA
Sub Task 3.1.3 - Comm. Btwn microcontrollers	2	MA
Task 3.2 - Processing Data - Database	16	LL
Sub Task 3.2.1 - Setup of backend Database	12	LL
Sub Task 3.2.2 - Setup of frontend User App	12	LL
Sub Task 3.2.3 - Store Data from MC to DB	4	LL
Sub Task 3.2.4 - Data retrieval DB to UA	4	LL



Project - EE2799	No. Days	Who
Goal 6: Reports	**53**	MA
Objective 1 - Homework 1	8	MA
Task 1.1 - Complete roughdraft of report	6	MA
Task 1.2 - Revise report and submit	2	MA
Objective 2 - Homework 2	7	MA
Task 2.1 - Complete roughdraft of report	5	MA
Task 2.2 - Revise report and submit	2	MA
Objective 3 - Homework 3	7	MA
Task 3.1 - Complete roughdraft of report	5	MA
Task 3.2 - Revise report and submit	2	MA
Objective 4 - Homework 4	7	MA
Task 4.1 - Complete roughdraft of report	5	MA
Task 4.2 - Revise report and submit	2	MA
Objective 5 - Homework 5	14	MA
Task 5.1 - Complete roughdraft of report	12	MA
Task 5.2 - Revise report and submit	2	MA
Objective 6 - Homework 6	7	MA
Task 6.1 - Complete roughdraft of report	5	MA
Task 6.2 - Revise report and submit	2	MA
Objective 7 - Final Report	14	MA
Task 7.1 - Creation of Power Point	12	MA
Task 7.2 - Present Presentation	2	LL


Project - EE2799
October
November
December
Wk1
Wk2
Wk3
Wk4
Wk5
Wk6
Wk7
Wk8
TASK NAME
No. Days
Who
Overall Goals for Term
Proposal / Research 8 MA
Design Development 12 MA
Build / Test 19 LL
Gathering Results of Protype 5 HL
Improvments 16 LL
Homework 53 MA
Goal 1: Proposal / Research 8 MA
Objective 1 - Market Research 8 MA
Task 1.1 - Similar Products 3 MA
Task 1.2 - Create Survey 2 LL
Task 1.3 - Execute Survey 5 LL
Task 1.4 - Create Interview 2 MA
Task 1.5 - Execute Interview 2 MA
Objective 2 - Customer Requirements 5 MA
Task 2.1 - Create List based on Similar Products 5 MA
Task 2.2 - Revise List based on survey and interview 3 HL
Goal 2: Design Development 12 MA
Objective 1 - Product Requirements 4 MA
Task 1.1 - Create List based on Customer Req. 2 MA
Task 1.2 - Finalize List based on Customer Req. 2 MA
Objective 2 - Product Specifications 4 MA
Task 2.1 - Create List based on Product Req. 2 MA
Task 2.2 - Finalize List based on Product Req. 2 MA
Objective 3 - Value Analysis 4 LL
Task 3.1 - Componenet Value Analysis 2 LL
Task 3.2 - Market Value Analysis 2 LL
Objective 4 - Design Review 1 4 MA
Task 4.1 - Creation of Power Point 3 MA
Task 4.2 - Present Presentation 2 LL
Goal 3: Build / Test 19 LL
Objective 1 - Purchase Parts 3 MA
Task 1.1 - Create Parts List 2 HL
Task 1.2 - Approval from Prof Bitar 1 MA
Task 1.3 - Submit Parts List 1 MA
Objective 2 - Hardware Implementation 8 MA
Task 2.1 - Sensor Network Setup 5 HL
Sub Task 2.1.1 - Pressure Sensor 2 HL
Sub Task 2.1.2 - Temperature Sensor 1 HL
Sub Task 2.1.3 - Light Sensor 1 HL
Sub Task 2.1.4 - Microphone 1 HL
Task 2.2 - Microcontroller Communication 3 MA
Sub Task 2.2.1 - Tranceiver setup Arduino 2 MA
Sub Task 2.2.2 - Tranceiver setup Raspberry Pi 2 MA
Objective 3 - Software Implementation 17 LL
Task 3.1 - Processing Data - HW 5 MA
Sub Task 3.1.1 - Processing Data - Pressure Sensor 2 MA
Sub Task 3.1.2 - Processing Data - Other Sensors 2 MA
Sub Task 3.1.3 - Comm. Btwn microcontrollers 2 MA
Task 3.2 - Processing Data - Database 16 LL
Sub Task 3.2.1 - Setup of backend Database 12 LL
Sub Task 3.2.2 - Setup of frontend User App 12 LL
Sub Task 3.2.3 - Store Data from MC to DB 4 LL
Sub Task 3.2.4 - Data retrieval DB to UA 4 LL
Goal 4: Gathering Result of Protype 5 HL
Objective 1 - General Result of Prototype 2 HL
Task 1.1 - Analysis of Prototype 2 HL
Objective 2 - Design Review 2 4 HL
Task 2.1 - Creation of Power Point 3 HL
Task 2.2 - Present Presentation 2 HL
Goal 5: Improvements 16 LL
Objective 1 - Revision of Prototype 9 LL
Task 1.1 - Identify Issues and improvements 3 LL
Task 1.2 - Improve Prototype 8 LL
Objective 2 - Packaging 5 HL
Task 2.1 - Packaging for HW Sensors 5 HL
Task 2.2 - Packaging for Central Hub 5 HL
Task 2.3 - PCB Design for Microcontrollers 3 MA
Task 2.4 - Build and Implement PCB 2 MA
Goal 6: Reports 53 MA
Objective 1 - Homework 1 8 MA
Task 1.1 - Complete roughdraft of report 6 MA
Task 1.2 - Revise report and submit 2 MA
Objective 2 - Homework 2 7 MA
Task 2.1 - Complete roughdraft of report 5 MA
Task 2.2 - Revise report and submit 2 MA
Objective 3 - Homework 3 7 MA
Task 3.1 - Complete roughdraft of report 5 MA
Task 3.2 - Revise report and submit 2 MA
Objective 4 - Homework 4 7 MA
Task 4.1 - Complete roughdraft of report 5 MA
Task 4.2 - Revise report and submit 2 MA
Objective 5 - Homework 5 14 MA
Task 5.1 - Complete roughdraft of report 12 MA
Task 5.2 - Revise report and submit 2 MA
Objective 6 - Homework 6 7 MA
Task 6.1 - Complete roughdraft of report 5 MA
Task 6.2 - Revise report and submit 2 MA
Objective 7 - Final Report 14 MA
Task 7.1 - Creation of Power Point 12 MA
Task 7.2 - Present Presentation 2 LL

Appendix C: References

https://learn.adafruit.com/force-sensitive-resistor-fsr/overview

https://cdn-learn.adafruit.com/downloads/pdf/thermistor.pdf

http://www.trossenrobotics.com/productdocs/2010-10-26-datasheet-fsr402-layout2.pdf

https://cdn-shop.adafruit.com/datasheets/103_3950_lookuptable.pdf

https://learn.adafruit.com/photocells/arduino-code

https://cdn-learn.adafruit.com/assets/assets/000/010/127/original/PDV-P8001.pdf

https://randomnerdtutorials.com/guide-for-microphone-sound-sensor-with-arduino/

http://sensorkit.en.joy-it.net/index.php?title=KY-037_Microphone_sensor_module_(high_sensitivity)