REM Sleep Alarm

Shaun Kelsey
Brendon Soltis

Advisors: Vladimir Prodanov, Franz Kurfess

6/3/11

Contents

List of Figures

List of Tables

Acknowledgements

We would like to acknowledge Professor Vladimir Prodanov for his guidance in the hardware design portion of this project. We would also like to acknowledge Professor Franz Kurfess for helping with the human computer interaction portion of this project and providing examples of user testing frameworks we could implement.

Abstract

This paper explores the REM Sleep Alarm project that deals with the sleep cycle of the average human, and how it affects our ability to function when waking up in the morning.

The purpose of this project is to build a device that has the ability to wake a person up when they will feel the most rested. From research done, it appears that most people feel the most rested when woken up during the REM stage of their sleep cycle. Our goal is to have the device detect when the user is in this stage of sleep, determine how often they enter the stage, and how long it lasts, and to wake them up as close to a preset alarm time as possible (without waking them up later than the set time) when they are in this stage.

Anyone can use this device, but it would perhaps be more useful for people who find waking up and getting out of bed in the morning to be a difficult task. However, waking up feeling more rested is something that anyone can appreciate, and so there is no specific group of intended users.

Introduction

Oftentimes, the hardest part of getting up in the morning is the actual act of getting out of bed. Grogginess from waking up can be compounded with a poor night's sleep or simply not enough sleep to make mornings uncomfortable until coffee or working out can wake us up. The purpose of this project is to alleviate some of this difficulty and give the user a much better experience waking up, while not making it any more difficult for the user to fall or remain asleep.

Research by Yaso et al. has turned up evidence that people tend to feel more refreshed and can get up easier when they are woken up from the Rapid Eye Movement (REM) stage of sleep. (Detection of REM...) This project, then, strives to create a physical system capable of detecting REM sleep in a user and of waking them up near a convenient time set by the user while the user is in REM sleep. We hope to show that this can be done with minimal interaction with the user while they are asleep.

Monitoring REM Sleep

This section covers the REM Sleep research and other similar projects we used in order to validate the project idea. Using this information, we were able to determine the requirements we needed in order to solve our problem.

REM Sleep Research

In their paper Detection of REM Sleep by Heart Rate, Yaso et al. put forth that there is a correlation between REM sleep and how you feel when waking up. They did a study on a group of ten subjects and found that eight out of ten of them felt better or much better when waking up out of REM sleep versus a deeper sleep. This was true even when they had less sleep on the REM night that the non-REM night. This is the basis upon which our project is built – trying to provide users with a better waking up experience.

Similar Projects

There are a number of systems that exist already that attempt to wake a person up while the are in their lightest part of sleep. This includes an iPhone application that monitors the movement of the individual in bed through the phone's camera and notes that the most movement will occur in the lightest stage of sleep.

In addition, a project by Mathieu Mallet, entitled "Morpheus", does direct monitoring of eye movements to determine when the user has entered REM sleep, although the purpose of his project was to try to get the user to enter a state of lucid dreaming through visual and auditory clues, rather than act as an improved alarm clock.

There are a few ways of detecting REM sleep, some more practical than others when it comes to ease of use. The most direct way is though the monitoring of electrical activity in the brain through EEG. There are certain clues available in this electrical activity that can tell you which stage of sleep the user is in. Unfortunately, this method usually requires the application of sensitive electrodes to the scalp, and in some cases requires that the user be shaved for a better reading.

A second method is the one used by the iPhone application as described above. As personally tested, the application only works for people who move minimally in there sleep. For users who consistently roll around in their sleep, the application misreads and wakes the user up at the wrong time. Also, the application is hindered by the surface and material that the iPhone is placed on. If the surface is firmer, the device will have a harder time detecting movement.

A third method is through the monitoring of heart rate. Heart rate tends to slow down during the deeper stages of sleep, and so a system could easily monitor and find when the heart is pumping the fastest and attribute that to REM sleep.

A final method (and the one that we have chosen to use) is the detection of eye movement. REM is an acronym for "Rapid Eye Movement", and during this stage, the user's eyes move back and forth. By shining an infrared signal on the eye and measuring reflection, an oscillation in the signal detected should indicate that the user is in REM sleep.

Requirements

As mentioned above, there are a few physical characteristics of REM sleep that can be user to monitor and determine when a user has entered it. Because we wanted to keep the complexity of detection low, while also keeping any discomfort to the user low, we immediately eliminated EEG monitoring as well as monitoring of physical body movement. We chose instead to monitor the movement of the user's eyes using infrared detectors and an emitter.

To consider this project a success then, we need the project to be able to detect eye movement using IR sensors. Care must be taken to maintain the IR emitter within the maximum safe power levels so that there will be no damage to the user's eyes.
After we have obtained a signal from the user's eye movement, the data must be sent wirelessly to a computer where processing occurs. Wireless transmission is necessary to ensure that no cables are exposed to the user during sleep which could occur in damage to the project, or cause the user to wake up.

The project must be able to consistently determine when the user is in REM sleep based on the data transmitted to the computer. To accomplish this, we will implement a method used by seismologists to determine when an earthquake has occurred based on seismogram readings. This method requires the comparison of two rolling time averages and will be discussed further in later sections of this report.

Finally, the project should implement an alarm feature that can wake the user up when REM sleep has been detected. Because it is unknown beforehand when REM sleep will occur, the project will attempt to activate the alarm as close to the time set by the user without going past it. This will ensure that the project will not allow the user to sleep in past the alarm time if they are not in a state of REM sleep around that time.

We also have a couple of ideas we will try to implement to improve the project, but which we do not consider to be required. These are that the user interface should be easy to use, and that the program save data specific to each user in an attempt to more easily predict when REM sleep will occur based on previous times it has occurred.

Overall Design

The project is split into two main sections, hardware and software. The hardware section includes the detection and amplification of the signal, as well as transmission to a computer for analysis. The software side includes analysis of the signal and alarm clock functionality.

Hardware Design

This section covers the hardware design of the REM Sleep project. It will provide a general overview of the decisions made during the design and construction of the sleep mask detector, including the power supply, signal creation, filtering and amplification, and wireless transmission.

Power Supply

Components of Interest
1.5V AA batteries
LM317
10uF capacitor

The power supply was chosen to be a single-supply 3.3V rail. This decision came primarily from the power supply requirements of the Xbee Series 1 RF module used for wireless transmission, which requires a power rail of 2.8V to 3.4V. Due to the nature of a single-supply rail, it became necessary to add a 1.5V level shift to the signal for the amplification stage to ensure that a full rail-to-rail swing could be achieved with little or no clipping. This process with be described in the amplification design section.

For testing purposes, a voltage regulation stage was added to ensure proper functionality of the Xbee module. Four 1.5V batteries were placed in series and the resulting output was fed in to a voltage regulator to get a power supply rail of 3.3V. A 10µF storage capacitor (C2 in Figure 1) was inserted to reduce sudden rail changes due to the Xbee module drawing variable amounts of current.

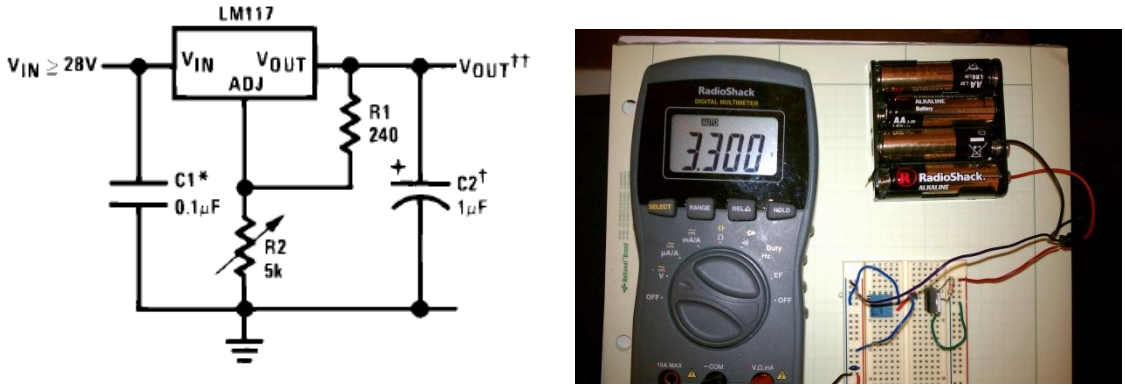

Figure 1: (Left) Voltage Regulation with LM317. (Right) Implemented System

IR Emitter and Detectors

Components of Interest
LTR-301 IR detector
LTR-302 IR emitter

The sleep mask is able to detect eye movement by shining infrared (IR) light on the user's eye and measuring how much light is reflected right versus left via two IR detectors placed on either side of the emitter on the mask. Because the human eye is not perfectly symmetrical, when the user moves their eye to the left or right, as in REM sleep, more IR photons will be deflected one way or the other, creating a difference in output currents between the two detectors. Figure 2 shows the final layout of the emitter, with detectors on either side of it. Note that the user's eye will be approximately underneath the emitter, rather than the center of the sleep mask curvature.

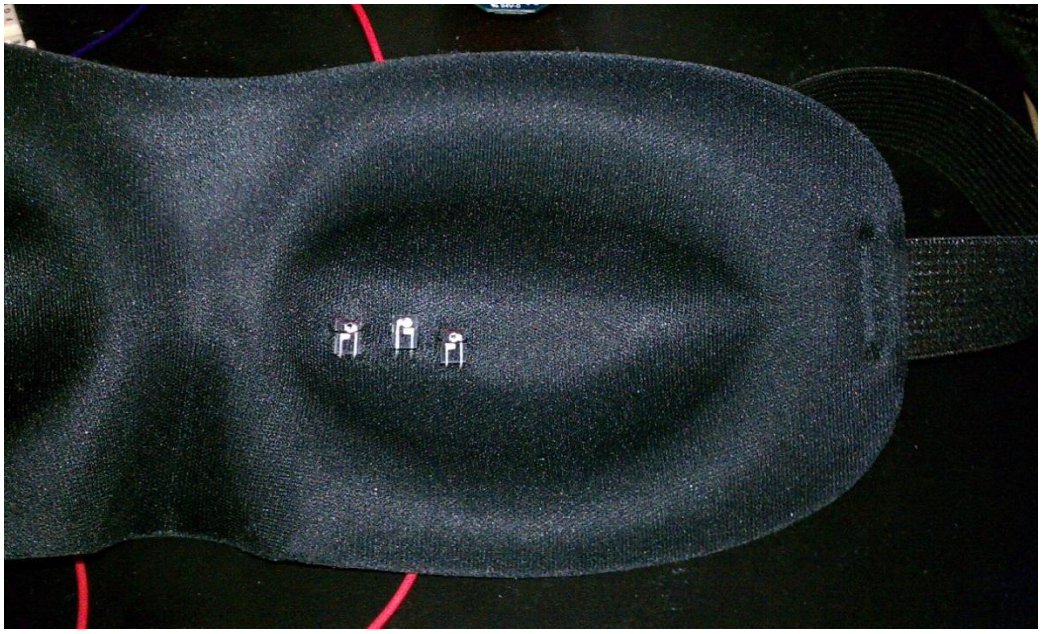

Figure 2: Final layout of emitter (center) and detectors.

The first and foremost goal in using the IR emitter was to ensure that the power level would not cause any damage to the user's eye. From the chart in Figure 3 on the next page, published by Han-Kwang Nienhuys, one may find the Maximum Permissible Exposure (MPE) allowed for a given wavelength and period of time. The MPE is approximately ten percent of the allowed power density that has a fifty percent chance of causing harm to the eye. The LTR-302 IR emitter emits light between 890nm and 990nm, as so would fall somewhere between the pink and orange lines of the aforementioned chart. We decided to leave a healthy margin of error and assume (unrealistically) that the user would keep their eyes open for 1000 seconds, and estimated the worst case limit of the MPE to be approximately 1mW/cm^2. From the datasheet, a forward current of 20mA through the emitter will cause a maximum output power density of 0.288, which is about one quarter of our worst case limit. Later testing proved that a forward current of 5mA is enough to detect a signal.

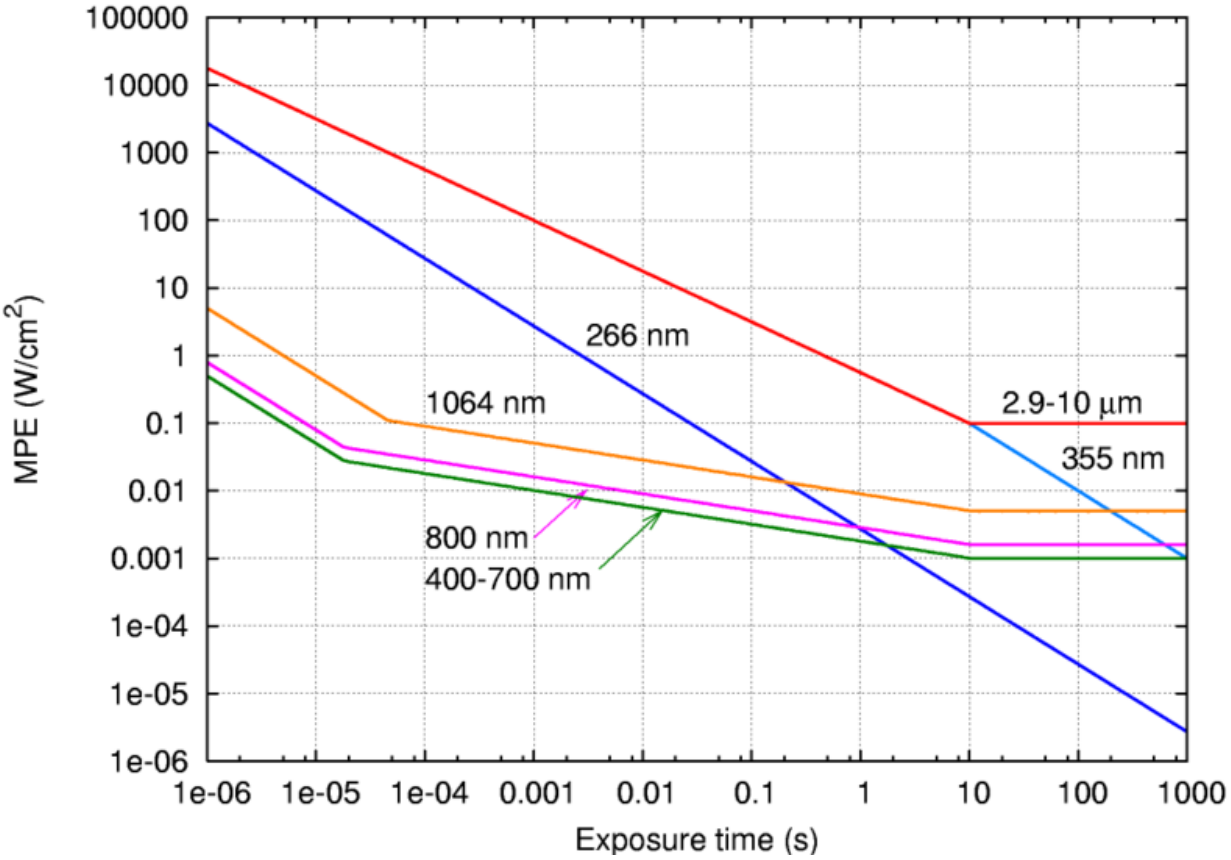

Figure 3: Chart used to calculate worst case MPE

Differencing Stage

The purpose of this stage is to take the two separate current signals from the detectors, subtract one from the other, and then pass this differential signal through a trans-impedance amplifier (TIA) to get a voltage signal.

Components of Interest
LTR-301 IR detector
LTR-302 IR emitter
LMX358 op-amp

The resistors R1 and R2 are selected to be equal so that the voltage across each detector (modeled as npn transistors) is equal. This ensures that for the same number of photons incident on the two detectors, they will each provide the same current. If there is any difference in currents, cause by the user's eye moving, the difference in current will exit towards the op-amp and through the TIA.

Maxim's LMX358 dual op-amp was chosen due to its low quiescent current, its low, single-supply voltage range, and its rail-to-rail output. In practice it works well and draws less power than stated by the datasheet.

R3 was chosen to be large enough to get a decent amplification from the TIA. Although ideally it could be chosen to be infinite, giving rail-to-rail outputs whenever there is any eye movement at all, in practice this cannot be accomplished. This is because there is some

unwanted DC current through R3 due to mismatch in the collector emitter voltages of the detectors, and also due to the placement of the user's eye in the mask (the same number of photons will not necessarily be hitting both detectors when the eye is straight forward. Later testing showed this DC current to be less than 1μA, with signal amplitude to be about a tenth of that. Thus, R3 was selected to be 100kΩ, therefore biasing SignalOut between 0.5V and 2.0V, depending on the DC current. With an ideal signal amplitude of about 10mV, the signal will not be cut off by this biasing range. Care had to be taken to make sure the signal amplitude did not surpass the bias value at SignalOut, or part of the signal would be cut off.

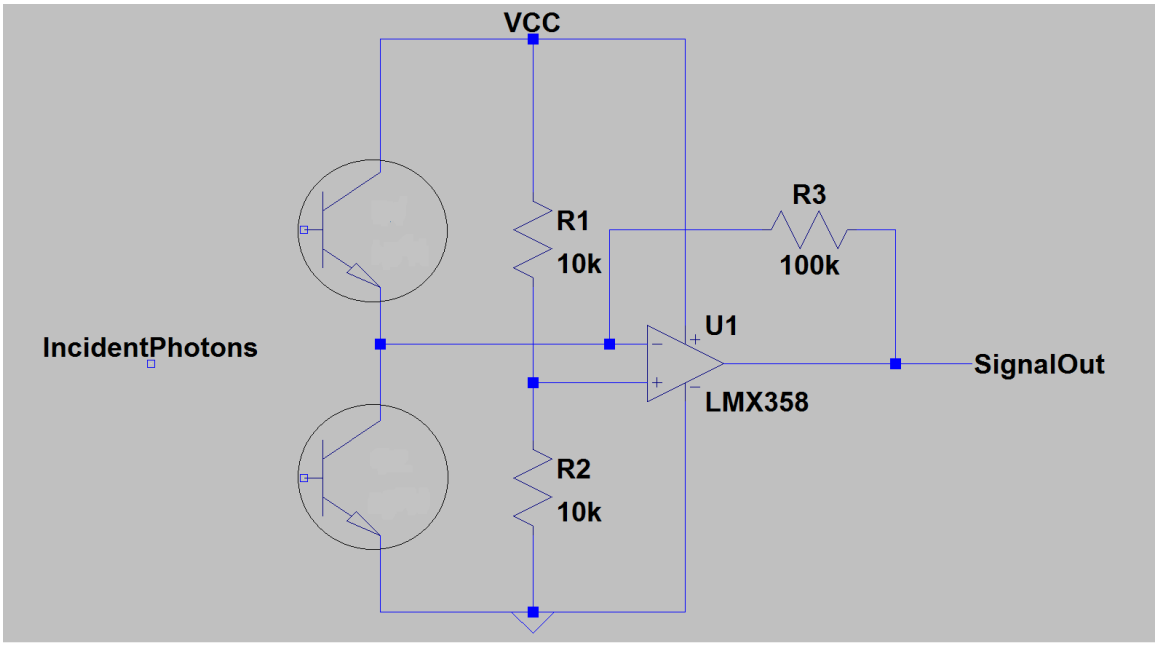

Figure 4: Differencing and TIA stage

Filter and Amplification Stage

Components of Interest
LMX358 dual op-amp
10μF capacitor

In this stage, a high pass filter is used to remove the variable DC biasing created by mismatch in the detectors, as mentioned in the previous section. Because human eye movement is a relatively low frequency signal, the high pass filter required a large RC time constant to ensure that the cutoff frequency would be below that of the signal. Assuming that the eye moves at least once every 5 seconds during REM (an overestimation), the cutoff frequency should then be below 0.2Hz

A 1st order non-inverting high-pass filter op-amp configuration was chosen, and a 10uF capacitor was picked for C1 because we had some extras on hand and it would provide a large value so as not to need an excessively large resistance. To determine the value needed for R3, the classic cutoff frequency formula was used:

$$f_c = \frac{1}{2\pi R_3 C_1}$$

Solving for R3 and inputting 10μF for C1 and 2Hz for fc gives a lower limit of R3 at approximately 80k. Thus, R3 was chosen to be 100k. This ends up setting the cutoff frequency well below the frequency needed, but no harm is done, since we only need to block the DC current signal.

We then chose a large value, 3.3MΩ, for the feedback resistor R4 so that we could get a decent amount of gain out of this stage.

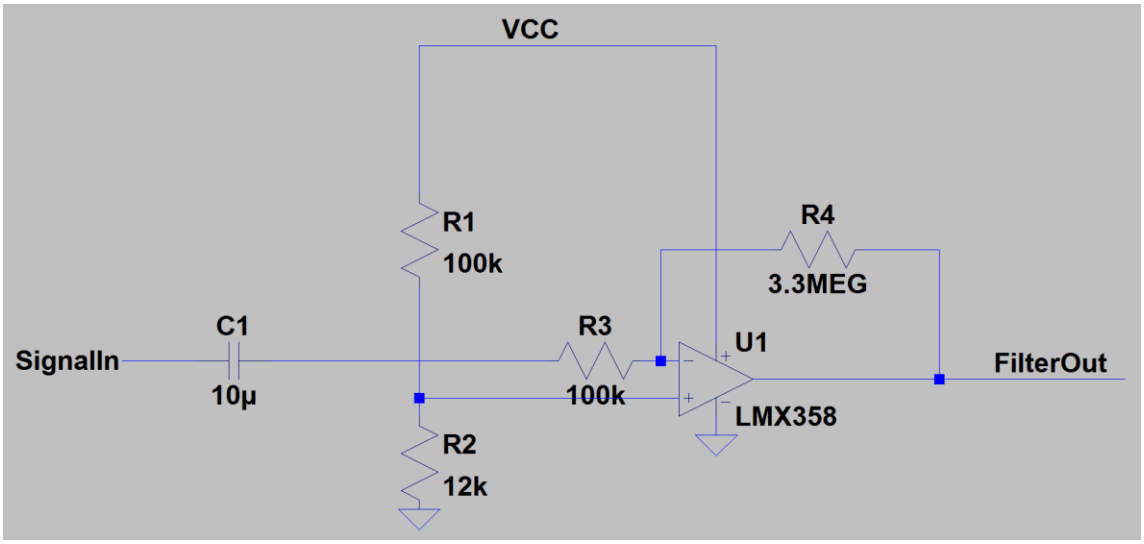

Figure 5: High pass filter with gain

Assuming the initial signal amplitude coming out of the TIA is about 10mV, we still need further gain to get from rail to rail. Instead of making R4 even larger and risk choking current to U1, a non-inverting buffer was added to get the rest of the gain needed. Because rail-to-rail voltage is about 3V, we want a final amplitude of 1.5V with a DC offset of 1.5V. That means a total gain of:

$$\frac{1.5V}{10mV} = 150$$

The gain from the high-pass filter is:

$$\frac{R_4}{R_3} = \frac{3.3Meg}{100k} = 33$$

So we need the buffer to have a gain of about:

$$\frac{150}{33} = 4.54$$

Because the initial estimation of 10mV for the amplitude was an overestimation used to prevent amplitude cutoff in the differencing stage, we chose a gain of 5 without fear of

cutoff here. A 100kΩ resistor was picked, arbitrarily, for R6, and R5 was solved for using the following equation for a non-inverting amplifier:

$$Gain = 1 + \frac{R_5}{R_6}$$

Solving for R5 with gain set to 5 gave a value of 400kΩ.

The values of R1 and R2 used on the high-pass filter were not picked until after the buffer stage was complete. This is because we needed to know the gain for the buffer stage before we could set the DC biasing of the high-pass filter stage. The DC biasing of the filter stage should be:

$$V_{DC\,(filter)} = \frac{V_{DC\,(buffer)}}{Gain_{filter}}$$

This is because the DC biasing of the filter gets amplified going through the buffer, and so must be divided by the filter's gain. Plugging in the output DC bias of 1.5V and gain of 5, we need resistors R1 and R2 to bias at 0.3V. Unfortunately, this causes a problem, because the theoretical amplitude of the signal at the output of the filter is 0.33V, which means the lowest 30mV of the signal would get cut off. Therefore, R1 and R2 were set to 100k and 12k respectively, giving a bias point of about 0.32V, and a 1MΩ potentiometer was inserted for R5 so that the gain could be adjusted for maximum signal amplitude and minimum cutoff.

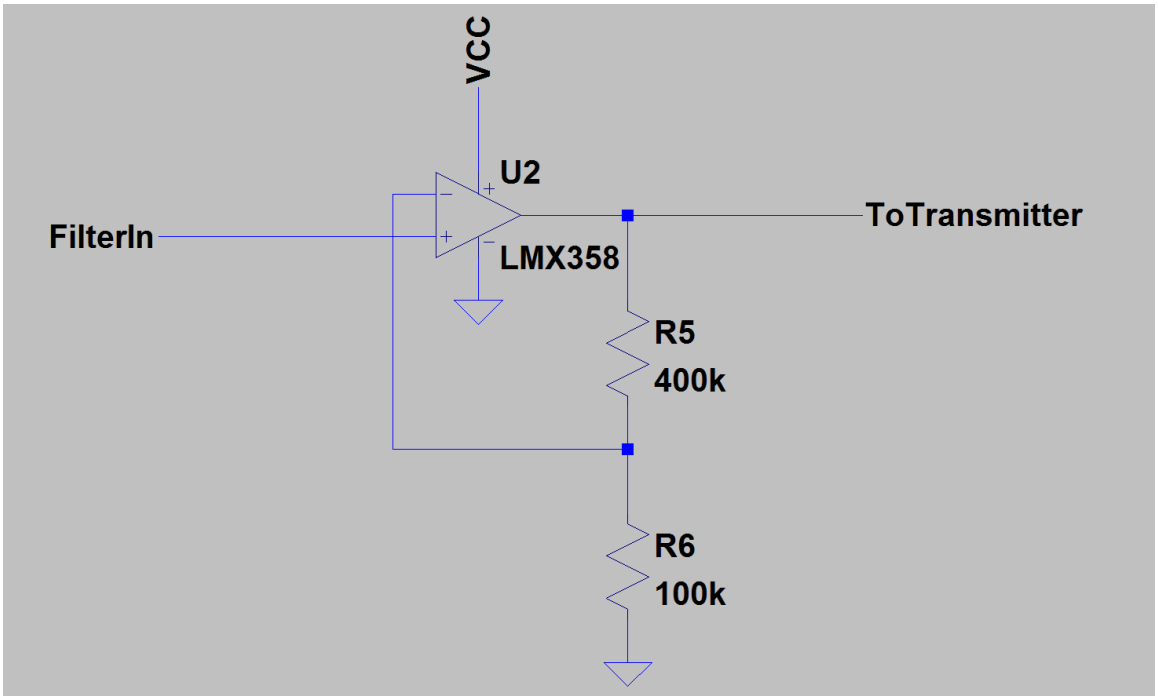

Figure 6: Gain and buffer stage

Wireless Transmission

Components of Interest
LTR-301 IR detector
LTR-302 IR emitter
LMX358 op-amp

Finally, the last stage of the hardware consists of transmitting the signal wirelessly to a computer for further analysis. This transmission is facilitated by the Xbee Series 1 RF module, using the popular ZigBee protocol (802.15.4). This chip comes with two options for antennas, wire and chip. Chip was chosen for this project because of its smaller profile. The chip draws around 50mA when transmitting or receiving, and less than 10μA when sleeping. The chip can be programmed via USB connection or wirelessly, and so it was not necessary to interface it with the sleep mask hardware any more than to provide power, ground, signal, and a voltage reference for the ADC.

Figure 7: Xbee Series 1 RF Module

The module was hooked up to power via breadboard, rather than directly to the sleep mask for ease of testing. Once the power, ground, signal, and voltage reference pins had wires soldered to them, it was trivial to plug the chip up to the necessary pins on the breadboard. A long wire soldered to the output of the sleep mask was hooked up to one of the analog input pins of the RF module, and testing occurred via java code.

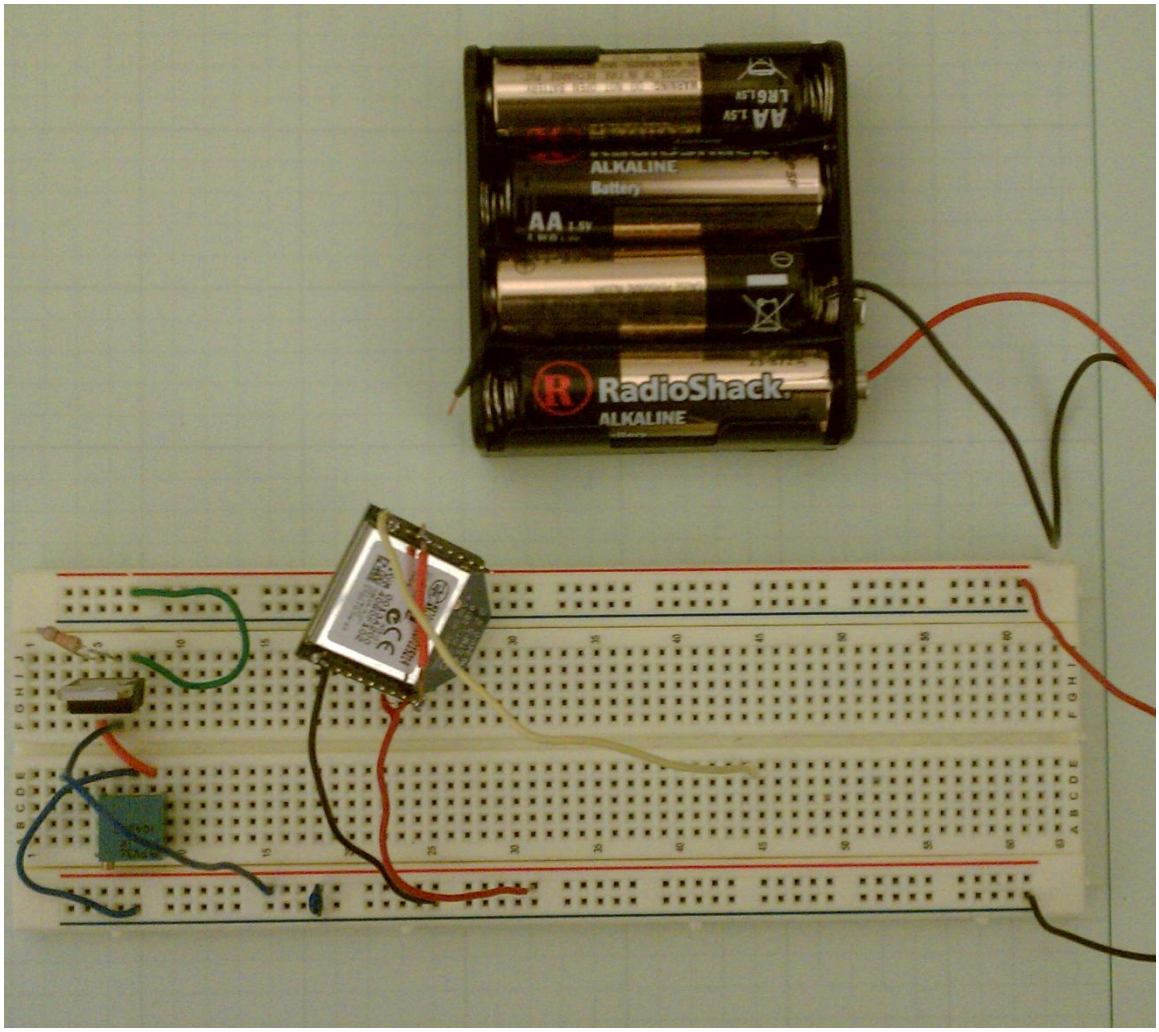

Figure 8: Xbee module hooked up to power

Hardware Design Summary

In summary, the hardware picks up a signal from two separate IR detectors and differences it, giving one signal that changes amplitude when there is eye motion. The signal passes through a high-pass filter and amplification stages, arriving at the Xbee transmitter as (ideally) a 1.5V amplitude signal with a DC bias of 1.5V, giving a nearly rail-to-rail swing.

Unfortunately, one issue the hardware does not cope with is a secondary problem that arises with the detectors not being perfectly matched. That is, they experience different amounts of current gain, as well as the aforementioned DC current. This means that even with the filtering, when the eye moves left versus right, the peak amplitude of the signal being pulled up versus pulled down will be different. Luckily, the software analysis disregards polarity of the signal and only looks at motion relative to the time average, making this less of an issue.

Construction of the circuit on the sleep mask was fairly straightforward, although difficult due to the flexible nature of the sleep mask and the absence of an easily accessible ground

plain. The circuit was soldered together in the "dead bug" style, with components hovering just above the mask itself. They were later secured to the mask via thread and needle, to reduce the chance of accidental damage.

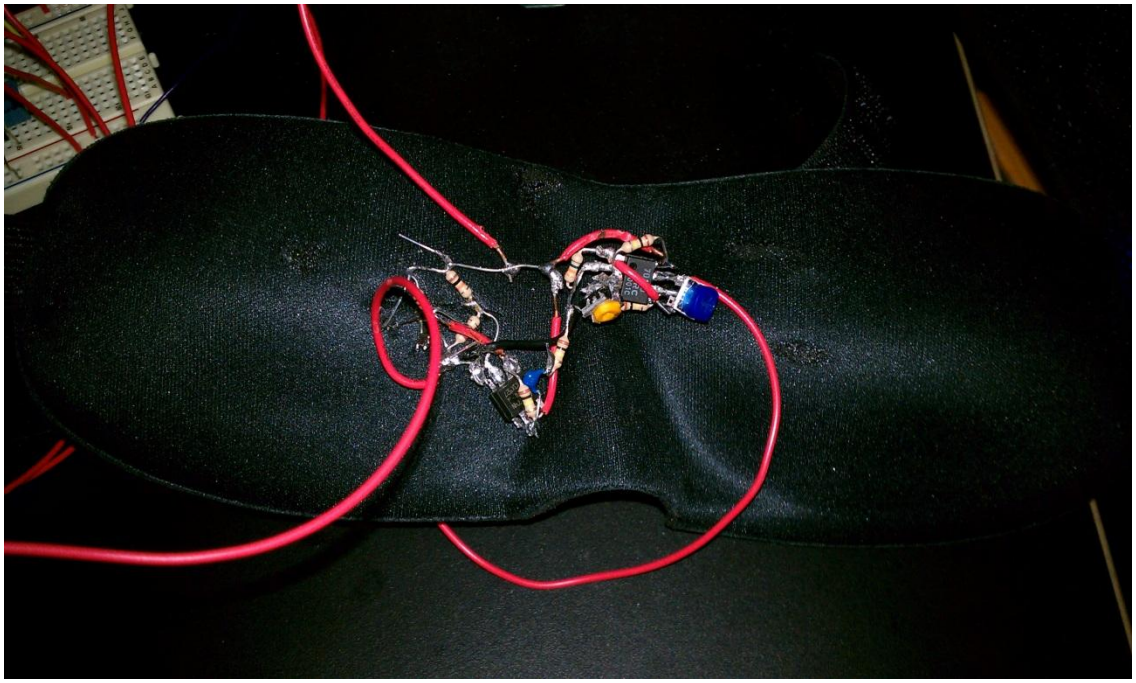

Figure 9: Differencing, filtering, and buffer stages soldered on to sleep mask

Software Design

This section covers the software design of the REM Sleep project. It will provide a general overview of the technical decisions made when designing the user interface, explore use cases, and explain in detail the methods that were implemented.

User Requirements

The user experience in a program that caters to everyday people needs to be simplistic and straightforward to use. Upon starting up the program, the user is presented with a simplistic Home screen as seen in Figure 10 on the next page.

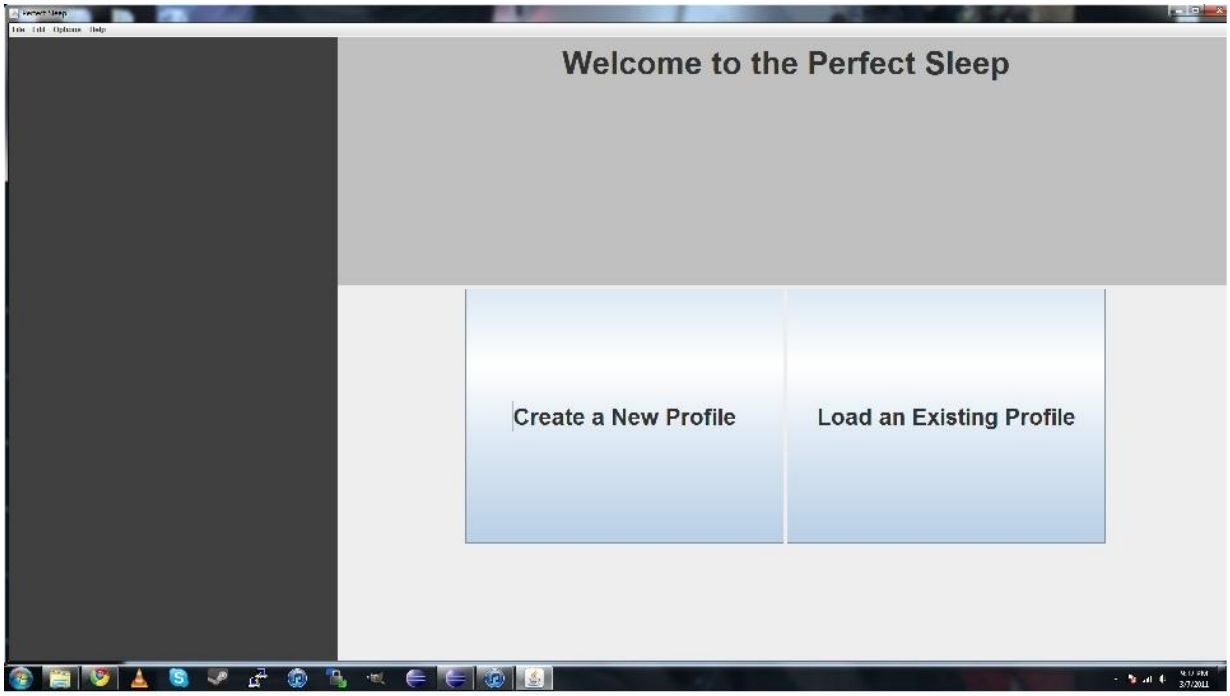

Figure 10: The Home screen of the program.

The user has two choices from here: either create a new profile or load an existing profile. If the user has a profile already created they would click on the *Load an Existing* Profile button. This would bring up a file selection dialog (Figure 11) where the user can choose a .tps file. Upon selecting the file and clicking *Open,* the profile will be loaded into the program.

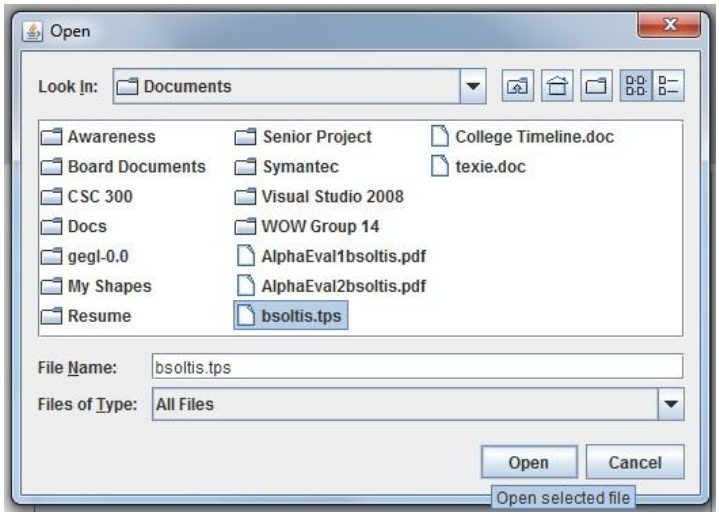

Figure 11: The Open file dialog.

If the user chooses to create a new profile by clicking the *Create a New Profile* button, a form will come up for the user to fill out to create a profile (Figure 12 on the next page).

Figure 12: Empty form for creating a new profile.

The user fills out their name that they want as their user name and their age. In Future Work, we will discuss the possibilities of expanding profile creation to include information that could be pertinent to sleep analysis.

If the user clicks the cancel button, the program will go back to the Home screen. Upon clicking the submit button, a new profile will be created and the user will be taken to the main screen as seen in Figure 13 on the next page.

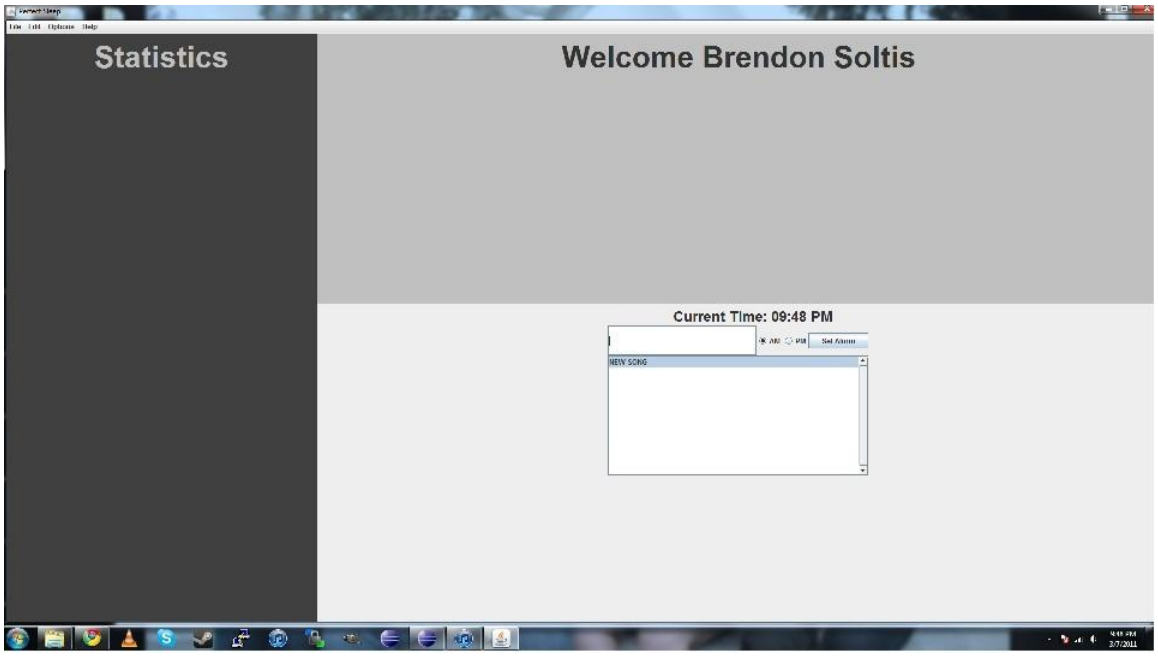

Figure 13: The main screen with a profile.

Here the main screen is divided into three main parts. First is the tool bar at the top. Second is the statistics panel on the left. Third is the main controls in the center of the screen.

The tool bar on the top is divided into four main menus: File, Edit, Options and Help. (Figure 14).

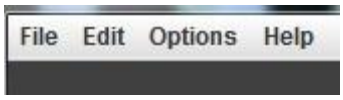

Figure 14: The tool bar.

The user can use the tool bar to access many features in the program. When the user clicks on File, they will see an expanded menu item with New, Open, Save, and Quit (Figure on the next page). New creates a new profile. Open lets the user choose an already made profile to load (Figure 15). Save allows the user to save his or her current profile. Quit exits the program but prompts the user to save their profile.

Figure 15: The File menu item.

The Edit and Options menu items have not been implemented yet. When the user clicks on Help, they will see an expanded menu item with About, How to Use, and Credits.

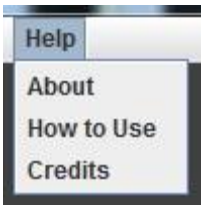

Figure 16: The Help menu item.

If the user clicks on About, they will cause a screen to pop up explaining what the program and the sleep module does in a high level description (Figure 17).

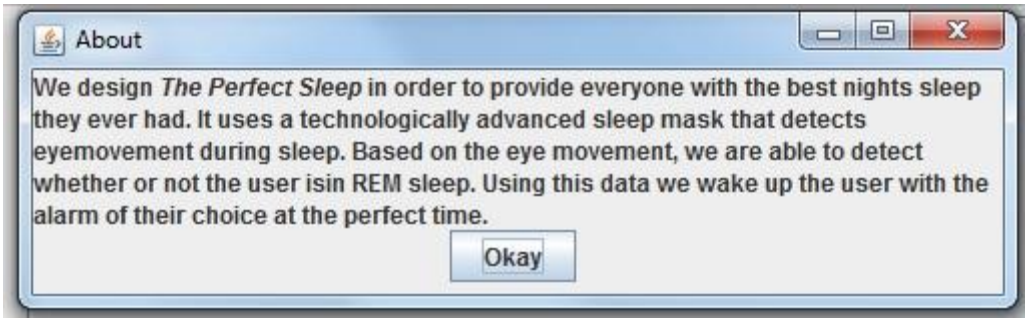

Figure 17: The About dialog.

If the user clicks on How to Use, a screen will come up describing how to use the program. And if the user clicks on Credits, it will display the creators and all credit to the developers (Figure 18).

Figure 18: The Credits dialog.

The Credits list those who contributed to the project as well as a copyright. When the user clicks Okay, the dialog will be closed.

Second, the statistics panel is located on the left hand portion of the screen. It contains information about the user including number of times the device is used, a graph of the current or latest sleep pattern, and an average rating of the sleep quality (Figure 19 on the next page). In Future Work, it will discuss what other useful information a user may want to see in the statistics panel.

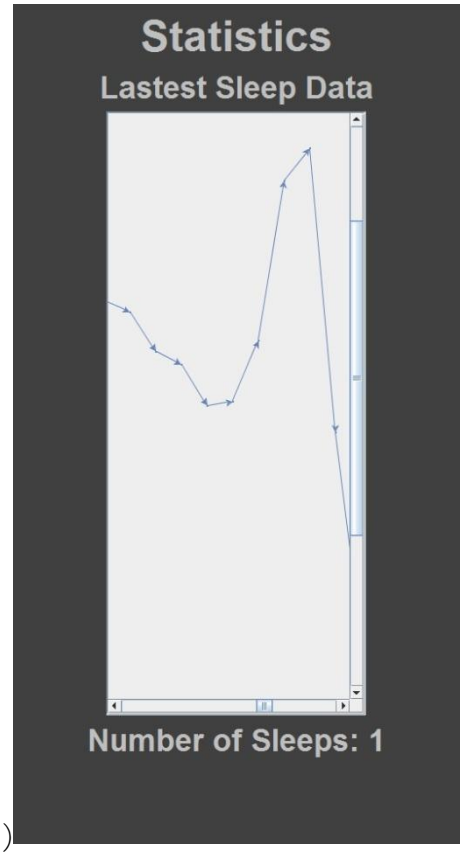

Figure 19: The statistics panel after a user uses the program.

The graph, as shown in Figure 20, provides the user with a more in depth look at their own sleep pattern. The graph is constructed by points sampled from the device every quarter of a second. The graph becomes very large very fast so the user is presented with a panel with a scroll bar to see the entire night.

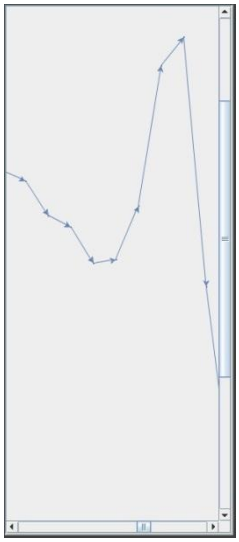

Figure 20: A graph of sleep data.

The final area, which is also to main area the user will interact with is the main alarm controls in the middle of the screen. This includes the alarm controls and the music selection box (Figure 21).

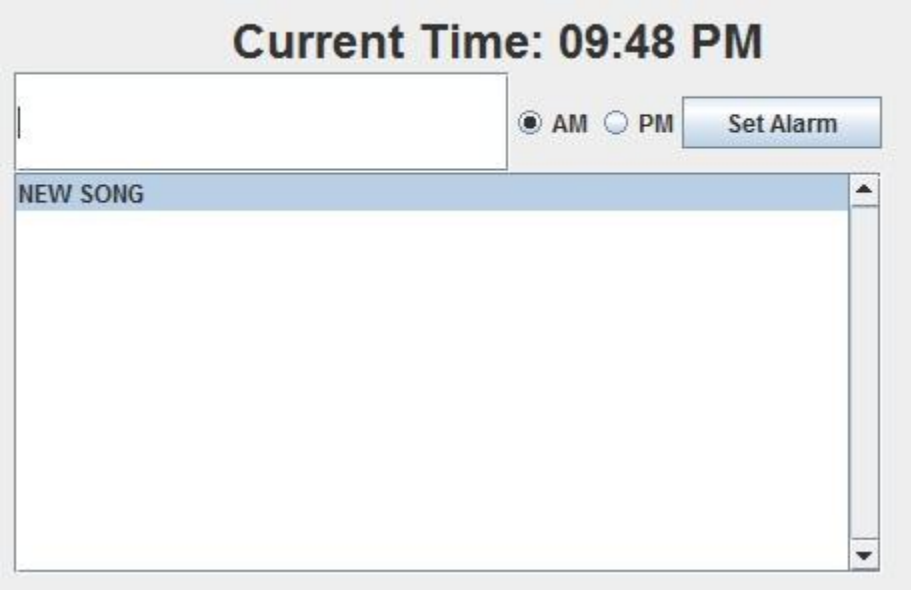

Figure 21: The alarm and music control panel.

The panel displays the current time and has a box for the user to set the desired alarm time. The radio buttons to the right are to indicate AM or PM. Once the user clicks Set Alarm, a dialog will pop up asking the user to select a sound file (Figure 22).

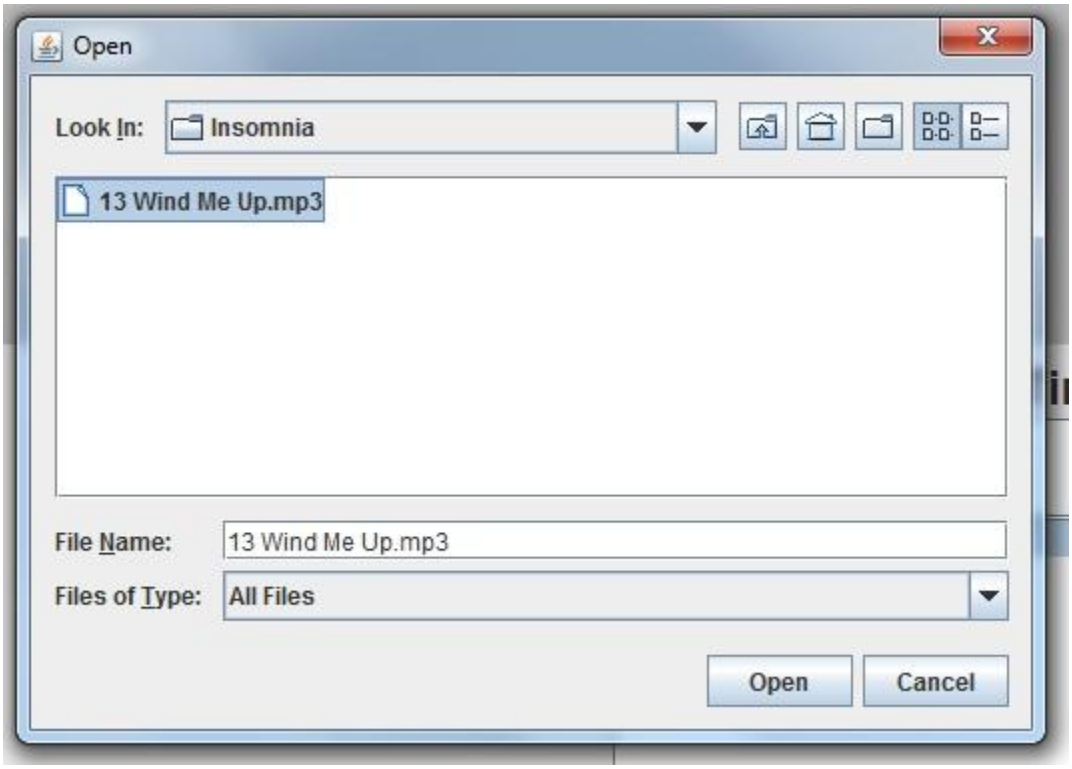

Figure 22: Music selection dialog.

The user can locate their favorite song or sound they want to wake up and open it. When the alarm sounds, it will be the sound file that the user selects. Once the user picks the sound, the alarm is set (Figure 23).

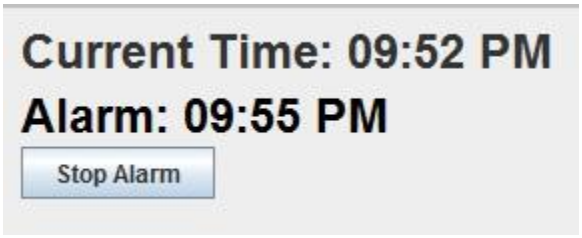

Figure 23: The alarm panel when an alarm is set.

The user sees the current time as before and also the time that the alarm is set. If the user wants to, they can stop the alarm and it will bring them back to the set alarm view. When the alarm goes off, the view changes to Figure 24.

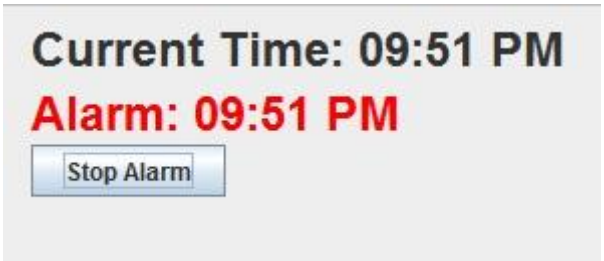

Figure 24: The alarm panel when the alarm sounds.

The alarm goes off and the text turns red to visually indicate the alarm. The user can click Stop Alarm to end the sound and bring them back to the set alarm view. Upon returning to the set alarm view, the song list will be populated. The list will keep track of the last twenty sounds or songs used so the user has fast access to the alarm that he or she wants. When the user saves his or her profile, the song list will also be preserved.

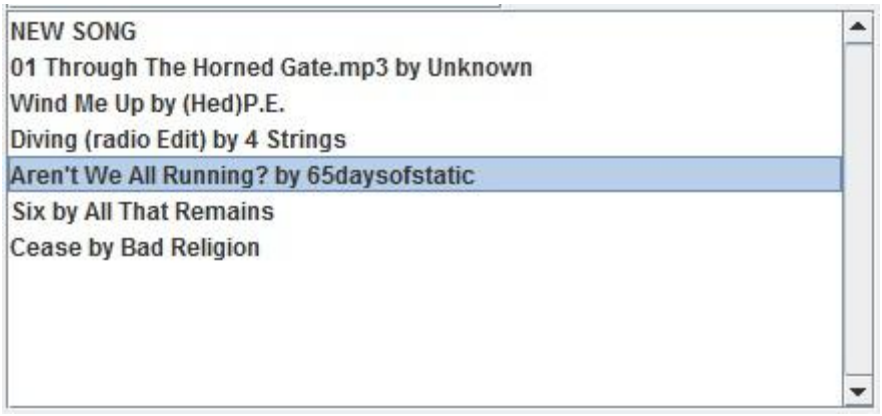

Figure 25: The song list.

That is the functionality of the REM Sleep Alarm, however there are error dialogs that should be known. An error will come up if the user leaves fields empty when creating a new profile (Figure 26). It will simply return them to the create a profile dialog.

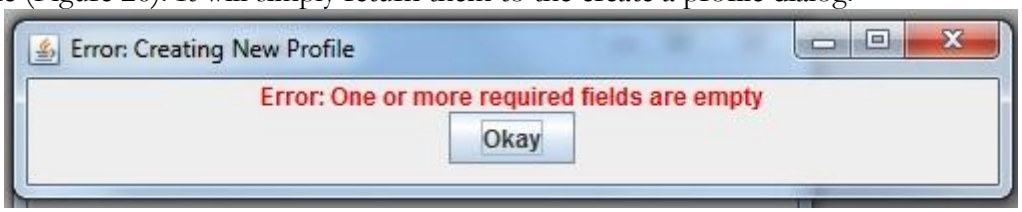

Figure 26: The creating a new profile error dialog.

The next error dialog is when the program cannot open a file. This could either be a profile or a music clip. The error dialog will pop up and return the user back to what screen they came from. If the profile isn't a .TPS file or the sound file a .MP3 file, this error message will appear.

Figure 27: Cannot open file error dialog.

The final dialog is confirming exit of the program (Figure 28). It gives the user the option of saving their profile or cancelling the exit all together.

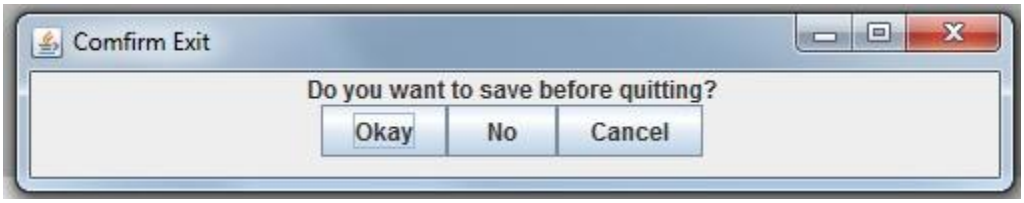

Figure 28: Confirm exit dialog.

By clicking Okay, the save dialog pops up. Clicking No will exit the program, and Cancel will return the user the last screen they were on.

Detailed Methods

This section will cover in more detail the methods and techniques used to accomplish the goals of this project on the software side. It will cover three main parts of the program: the seismic detection, XBee API Configuration, the audio sound library, and graphing.

Seismic Detection

To detect changes in the incoming data from the sleep mask, we are using techniques that are used to detect seismic disturbances. We take a long average of the data that comes in over an hour and at the same time we take in a short average over a period of a minute. By

comparing these averages as a ratio, if the averages spikes over a threshold of 1.2, we can determine that the user's eye movement has spiked. We determined this threshold by testing with real users. We looked at the averages of a person with no eye movement and then a user with a lot of eye movement. The average threshold value between the two states was a 1.2 ratio. By pairing that with the current time and the alarm time, if the user is in REM sleep and the time is within thirty minutes (default), we can set off the alarm to wake up the user.

```
if (compareAve() || (hour == curH && minute == curM && am_pmG.equals(am_pm))) {
        cur.incrNumSleeps();
        isAlarm = false;
        aTime.setForeground(Color.RED);
        playMusic(alarmSound);
        createGraph();
}
```

The method compareAve() takes the two averages that are calculated with each incoming packet and compares them with a small threshold and returns true if the time is in the correct range and there is a spike in the small average. If the user does not enter REM sleep by the time the alarm is set for, we set off the alarm anyways. For the future, we want the user to be able to specify a length of time before or after the alarm time that we can use as a buffer around the alarm time.

```
private boolean compareAve() {

      if (average / shortAverage > DATA_THRESHOLD && diff <=
            TIME_THRESHOLD)
            return true;
      }
      return false;
}
```

XBee API Configuration

To interface with the XBee chips, we need special configurations and initializations to get the wireless communication working. We used a program X-CTU to configure the remote chip as well as some of the preliminary settings on the local chip (Figure 29 on the next page). We were able to set API mode, enable our analog lines, set our sample rate, set the correct address of both local and remote chip, set the PAN ID, set the channel, and many more helpful settings to get the communication working.

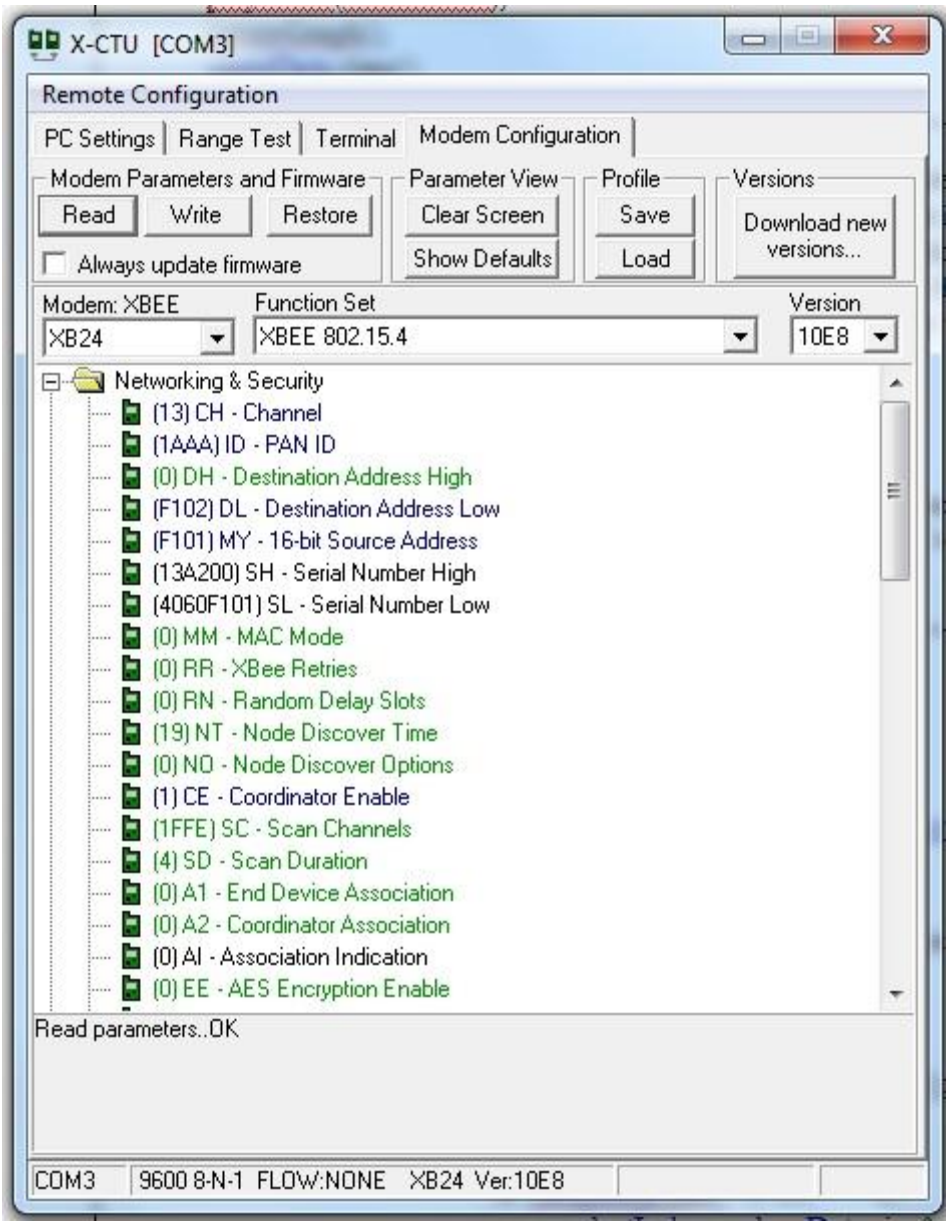

Figure 29: X-CTU interface to configure the XBee chips.

After the chips were properly configured, we could proceed to detect and connect to the chip from the Java program.

```
private void configureXBee() {
        xbee = new XBee();
        count = 0;
        lastIndex = 0;
        int com = 0;
        boolean connected = true;

        while (com < 256) {
                try {
```

```
                        xbee.open("COM" + com, 9600);
                        connected = true;
                } catch (XBeeException e) {
                        System.err.println("XBee chip not connected to COM" + com);
                        connected = false;
                }

                if (connected)
                        break;

                com++;
        }

        if (!connected) {
                System.err.println("XBee chip not connected");
                return;
        }
...
}
```

This code snippet shows how the program connects to the XBee chip. After creating a new XBee object, the program attempts to find a local chip on a COM port. Since there are no specific port that the chip would be connected to, it attempts to connect to all ports up until the max port of 256. If there are multiple chips connected, the chip on the first COM port will be the one connected to. If there is not a XBee connected at the time, the program will throw the error "XBee chip not connected." However, the program will still act like a regular alarm clock.

```
xbee.addPacketListener(new PacketListener() {

        @Override
        public void processResponse(XBeeResponse arg0) {

        if (isAlarm && arg0.getApiId() == ApiId.RX_16_IO_RESPONSE &&
                        !arg0.isError()) {

        RxResponseIoSample ioSample = (RxResponseIoSample)arg0;
        for (IoSample sample: ioSample.getSamples()) {
                if (ioSample.containsAnalog()) {
                        if (sample.getAnalog4() != null) {
                                sleepData.add(sample.getAnalog4());
                                count++;

                                if (sleepData.size() % 120 == 0)
                                        lastIndex = sleepData.size();

                                calculateAve();
```

```
                                calculateShortAve();
                        }
                } else {
                        System.out.println("No analog response");
                        }
                  }
            }
```

This code snippet is the packet listener added to the XBee chip. Every time a packet is sent from the remote chip to the local chip this method is called. First we check to see if the response is the proper packet from the remote chip and if the alarm is set. Then we for loop for the number of ioSamples in the packet received. For each analog sample, we check to see if it contains analog and if it does we get the analog sample off the pin and save it to the current sleep data. Then we calculate the averages on the fly and return.

Audio Alarm Library

Java does not support playing .MP3 files due to licensing issues. To get .MP3 sound files, we had to use a decoding library to decode the file into a format that is playable by Java. We used the Java Zoom media library to accomplish this.

```
import javazoom.jl.player.Player;

private void playMusic(String music) {
        try {
                FileInputStream fis     = new FileInputStream(music);
                BufferedInputStream bis = new BufferedInputStream(fis);
                playMP3 = new Player(bis);
         }catch (Exception e) {
                System.out.println("Problem playing file " + music);
                e.printStackTrace();
         }

         new Thread() {
                public void run() {
                        try { playMP3.play(); }
                        catch (Exception e) { e.printStackTrace(); }
                }
        }.start();
}
```

The method playMusic is utilized to play any sound. It takes an input stream and opens the file and passes that stream into the player. In the player, the stream is decoded. Finally a new thread is created to play the music so the user can still interact with the program while the alarm is sounding.

Graphing

We decided it would be advantageous to the user if we displayed the sleep data visually. For this we use a line graph which is the detected value versus time. For this we use an extended graphing library that extends Java Swing Component.

```
private void createGraph() {
        mxGraph graph = new mxGraph();
        graph.getModel().beginUpdate();
        int MAX = 1200;
        float xpos = 0;

        try {
                Object v1 = graph.insertVertex(graph.getDefaultParent(), null, "", 0,MAX -
                        sleepData.get(0), 1, 1);

                for (int i = 1; i < sleepData.size(); i++) {
                        xpos +=30;
                        Object v2 = graph.insertVertex(graph.getDefaultParent(), null, "",
                                xpos, MAX - sleepData.get(i), 1, 1);
                        Object e1 = graph.insertEdge(graph.getDefaultParent(), null, "", v1,
                                 v2);

                        v1 = v2;
                }
        } finally {
                // Updates the display
                graph.getModel().endUpdate();
        }

        mxGraphComponent graphc = new mxGraphComponent(graph);
        JFrame j = new JFrame();
        j.add(graphc);
        j.setSize(new Dimension(1000, 1200));
        j.setVisible(true);
}
```

In the method createGraph, we use the mxGraph component to insert vertices and edges. In the for loop, the method is taking each of the sleep data points and adding them to the graph. It systematically adds edges in between each adjacent edge. After that it adds the graph to a frame and displays it.

Testing

This section will cover the testing (software and hardware) done in our project.

Hardware Testing

Power

Verification of the separate hardware stages mostly took place as they were being built. It was simple enough to power up the device and use a multimeter to measure specific nodes to double check that the actual values were close to the expected values. Below is a table of expected and actual values for various stages of the final circuit:

Item Measured	Expected Value	Actual Value
TIA DC offset current	<10μA	Varies (<1μA)
TIA output amplitude	10mV	~8mV
HP filter output amplitude	330mV	~240mV
HP filter gain	33	~31
Buffer output amplitude	1.65V	~1.65V
Buffer gain	5	~5

Table 1: Various expected and measured values

The buffer output amplitude is the amplitude of the final signal that is sent to the transmitter, and testing shows that it can reach a full rail-to-rail swing under ideal conditions (using a finger to block one detector or the other), which is desired for maximum resolution.

Once the device was finished, a few test subjects were chosen to see how eye movement would be detected for different people. The main difference between subjects is the amplitude of the final signal; the device gave nearly a full rail-to-rail swing on one subject, whereas with another it could only achieve about a third of that. Luckily, as long as there is an obvious difference in the signal between the times when there is eye movement and there is none, the software analysis should be able to detect eye movement.

Figure 30 on the next page shows the best results obtained from our subjects. Eye movement can be clearly seen when the output spikes down suddenly, and also when it pulls up to VCC. These two directions correspond to the eye moving left versus right. The downward spike are more prominent due to the fact that the detectors are not placed perfectly over the user's eye (thus resulting in the aforementioned DC offset current). This results in different gains for the two detectors, and thus one detector dominates the other in pulling/pushing current.

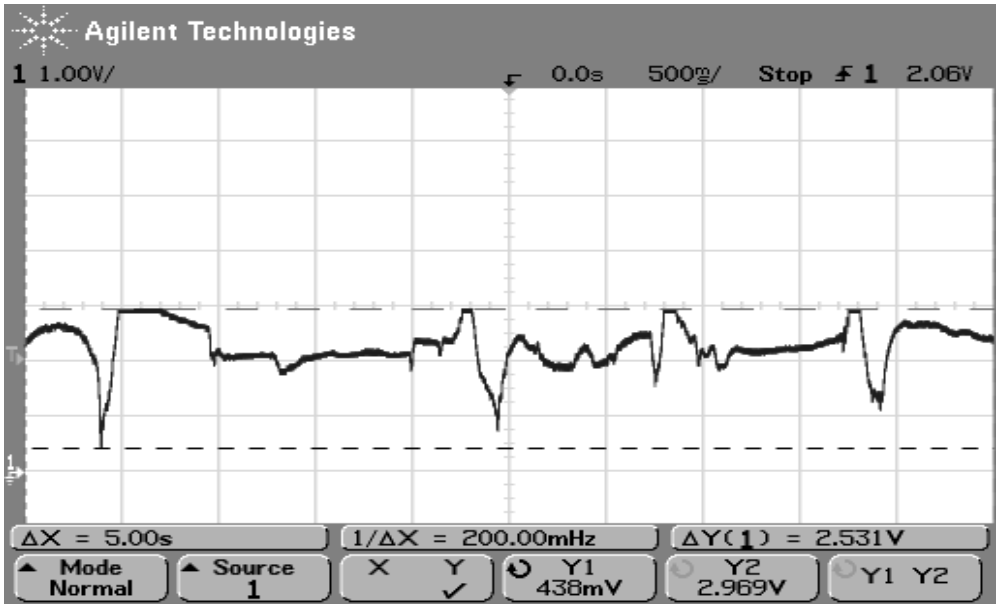

Figure 30: Nearly full rail-to-rail swing is observed (about 2.5 out of 3.3V)

Power wise, the Xbee module draws the largest peak current from the batteries, which is to be expected. The following is an attempt to analytically derive the average amount of current the module draws over one period.

The module is measured to draw between 50-55mA when transmitting, versus a mere 1.8μA when in sleep mode. It transfers data at 250,000bps, and we set the module to gather data ten times per second and transfer 10 data points at a time (once per second). Each data point is 1 byte, and each packet has an overhead of 16 bytes. Therefore, the amount of time the module takes to send one packet is approximately:

$$t = \frac{(overhead + samples)}{transfer\ rate} = \frac{16 + 10}{(\frac{250000}{8})} = 832\mu Sec$$

The duty cycle for the RF module, is therefore:

$$\frac{832\mu Sec}{1Sec} = 0.0832\%$$

This ignores the amount of time it takes the radio to start up, which is likely non-negligible. If we use the model derived by Polastre et al. in their paper on Versatile Low Power Media Access for Wireless Sensor Networks, we can find an estimate of the startup current that should be on the correct order of magnitude.

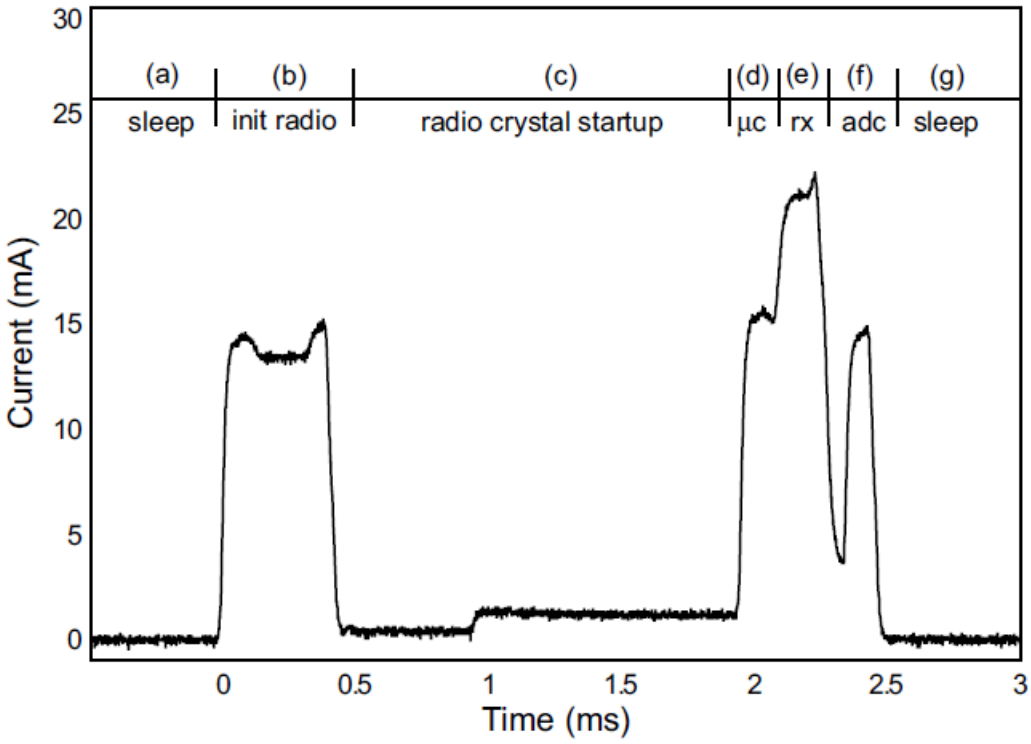

Figure 31: Startup process for Mica2 mote and CC1000 transceiver used in Polastre's project

From the above figure, we can see that the receive/transmit time is actually a small percentage of the overall process. A great deal of the current drawn is during radio initialization, which takes about 0.5ms and draws about 15mA during that time. If we assume the Xbee module draws the same current as this CC1000 transceiver for the same amount of time (again, this is only a order of magnitude approximation), we finally have all the information we need to make an estimation of the average current the Xbee module draws.

By taking the weighted average of the current when the module is on and off throughout one period, we can estimate the average current drawn by the RF module to be:

$$(50mA)(0.000832) + (15mA)(0.005) + (1.8\mu A)(1 - 0.000832 - 0.005) = 118.4\mu A$$

This is a somewhat surprising result, because it means that the RF module actually draws less current than the entire sleep mask circuit by a factor of about 25. Note that this is only the case because it is set to transmit once per second rather than every time it gathers a data point.

The highest use of power then is the emitter, which draws a constant 2.2mA. The sleep mask in its entirely draws only 2.5mA, so the emitter is a large percentage of this. This is good news, because it means battery life can be easily extended by lowering the emitter power as far down as possible while still maintaining a readable signal.

Signal

Testing of the signal from the sleep mask was trivial once the software was up and running, since the software can easily act as a sort of oscilloscope with its graphing functionality. To highlight the difference between the signal derived from different users, we had two subjects perform a simple test: five seconds of no eye movement, five seconds of moving their eyes back and forth, and a further five seconds of no eye movement. The results of the tests are displayed below.

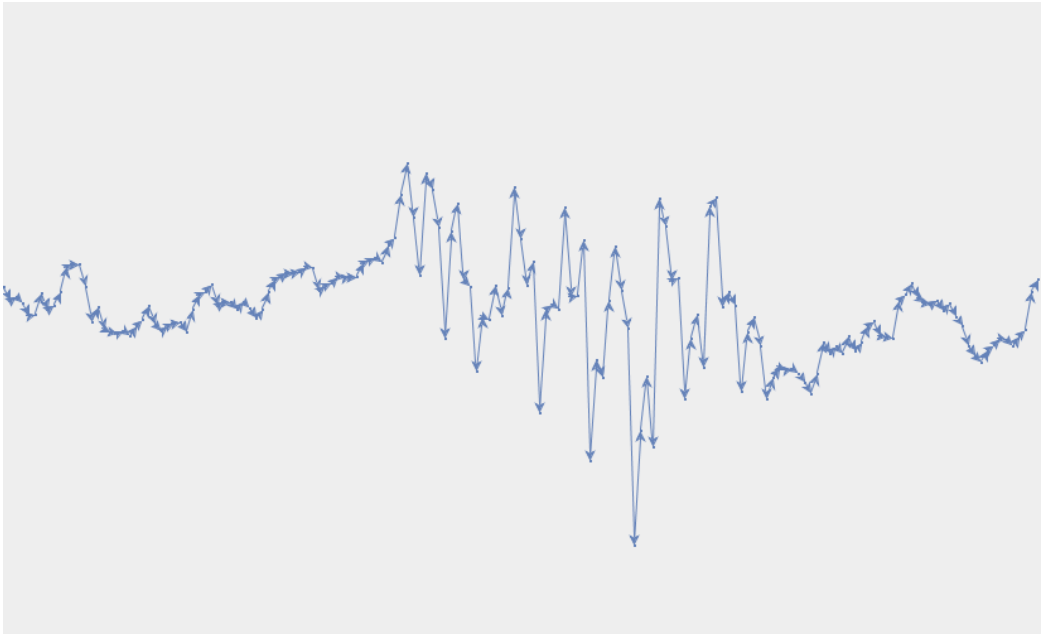

Figure 32: Subject 1 – approximately 15 seconds of user data

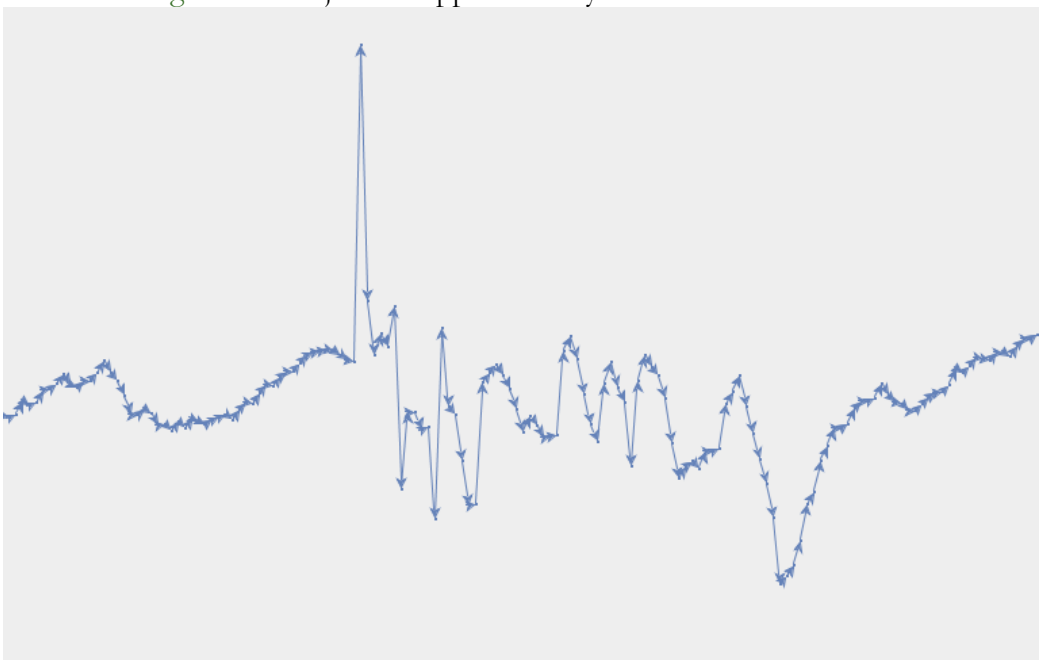

Figure 33: Subject 2 – approximately 15 seconds of user data

The vertical and horizontal axes for figures 32 and 33 are equal. From these figures, we can see that the sleep mask is able to pick up more pronounced movement from subject two, and that the data for subject two is less jagged than that of subject one's. Because the peaks and troughs for subject two are of a larger magnitude, the software will have an easier time detecting REM sleep.

The difference in sleep profiles from the two users can be attributed to many things. Among the most likely are: differing placement of mask over eyes, more or less pronounced eye movement, more or less pronounced bulge of eyelid due to cornea dimension, and finally interference due to eyelashes.

Because each user has a slightly different sleep profile, one of the future plans that we have for the software is to tailor the threshold for REM and time of alarm to specific users. For example, subject one will have a smaller ratio of short-term time average to long-term time average when in REM sleep, so the software should set the threshold for this ratio to be lower for this user. Likewise, if a user has a longer sleep cycle period (resulting in a longer period of time in between two consecutive periods of REM) the software may have to wake them up at an earlier time if it is likely that they will not enter another period of REM before the alarm time. By saving the profile for each user, the software should be able to get better over time as the users use it more.

Software User Testing

We administered a user test to five individuals of varying computer skills. They were all college students but came from different majors. The form we used for the test consisted of the questions below:

Age:

Major (if aaplicable):

Experience/skill with computers:

Task 1: Find the explanation of what this program does, how to work it, and the creators.

Task difficulty: Easy 1 2 3 4 5 6 7 8 9 10 Hard

Why did you rate this task as you did?

Describe your process when completing this task.

Task 2: Create a new profile and set an alarm for one minute past the current time. Wait for the alarm to sound and stop the alarm.

Task difficulty: Easy 1 2 3 4 5 6 7 8 9 10 Hard

Why did you rate this task as you did?

Describe your process when completing this task.

Task 3: Save your newly created profile as "test" and quit the program.

Task difficulty: Easy 1 2 3 4 5 6 7 8 9 10 Hard

Why did you rate this task as you did?

Describe your process when completing this task.

Task 4: Load you "test" profile and set an alarm for one minute past the current time with a song already in the recent songs list. Now reset the alarm with a different sound.

Task difficulty: Easy 1 2 3 4 5 6 7 8 9 10 Hard

Why did you rate this task as you did?

Describe your process when completing this task.

What is your overall experience with the software?

Did you encounter any errors or bugs when using this software?

Do you have any recommendations for a better user experience?

We gave them this form and with no further instructions. they had to complete all four tasks.

Tester 1

Age: 19
Major (if applicable): Liberal Studies
Experience/skill with computers: I own one

Task 1: Find the explanation of what this program does, how to work it, and the creators.

Task difficulty: 1
Why did you rate this task as you did?
Because I was able to accomplish this task quickly.

Describe your process when completing this task.

I selected 'Help' and then each of the things I needed to find were listed under that menu.

Task 2: Create a new profile and set an alarm for one minute past the current time. Wait for the alarm to sound and stop the alarm.

Task difficulty: 5
Why did you rate this task as you did?
It was so hard to figure out why all the random stuff on the computer kept coming up. But then I thought I might choose a song and I did.

Describe your process when completing this task.
1. I typed in the time and chose PM. 2. I clicked set alarm. 3. I chose the music. 4. I listened to the alarm and stopped it.
Task 3: Save your newly created profile as "test" and quit the program.

Task difficulty: 7
Why did you rate this task as you did?
It told me it didn't save it but I did!

Describe your process when completing this task.
1. Selected File -> Save 2. Typed in "test" 3. Clicked save 4. Clicked Quit but it prompted me to save again.

Task 4: Load you "test" profile and set an alarm for one minute past the current time with a song already in the recent songs list. Now reset the alarm with a different sound.

Task difficulty: 1
Why did you rate this task as you did?
Once the program restarted, it was easy to see what I had to do.

Describe your process when completing this task.
1. Load existing profile 2. chose "test"

What is your overall experience with the software?
I don't really know what it does but it's great!
Did you encounter any errors or bugs when using this software?
Some spelling errors on some of the screens.
Do you have any recommendations for a better user experience?
The colors are depressing.

Tester 2

Age: 25
Major (if applicable): Biomedical Engineering
Experience/skill with computers: 7 on a 1 -10 scale

Task 1: Find the explanation of what this program does, how to work it, and the creators.

Task difficulty: 1
Why did you rate this task as you did?
It's intuitive.

Describe your process when completing this task.
Looked for Help

Task 2: Create a new profile and set an alarm for one minute past the current time. Wait for the alarm to sound and stop the alarm.

Task difficulty: 3
Why did you rate this task as you did?
It didn't open to a folder containing mp3's and therefore it was hard to find a MP3 file.

Describe your process when completing this task.
followed on screen instructions.

Task 3: Save your newly created profile as "test" and quit the program.

Task difficulty: 3
Why did you rate this task as you did?
There was no button on the screen for save and I had to go to the menu bar.

Describe your process when completing this task.
I searched for a save function since it was not easily visible.

Task 4: Load you "test" profile and set an alarm for one minute past the current time with a song already in the recent songs list. Now reset the alarm with a different sound.

Task difficulty: 3
Why did you rate this task as you did?
Decently easy...

Describe your process when completing this task.
Did the same thing as task 2, but instead of a new song, used an old one.

What is your overall experience with the software?
decent, no real issues.
Did you encounter any errors or bugs when using this software?

Couldn't use keyboard shortcuts such as tabbing and entering through forms.
Do you have any recommendations for a better user experience?
Implement a Save Profile button and have an example of how the time should be formatted.

Tester 3

Age: 23
Major (if applicable): Civil Engineering
Experience/skill with computers: fairly experienced

Task 1: Find the explanation of what this program does, how to work it, and the creators.

Task difficulty: 1
Why did you rate this task as you did?
It was under help.

Describe your process when completing this task.
Looked for Help and I found it

Task 2: Create a new profile and set an alarm for one minute past the current time. Wait for the alarm to sound and stop the alarm.

Task difficulty: 2
Why did you rate this task as you did?
AM:PM can be a little tricky and missable but it was pretty straightforward.

Describe your process when completing this task.
The giant button that said Create New Profile was a little bit of a giveaway. It was obvious where to set the alarm.

Task 3: Save your newly created profile as "test" and quit the program.

Task difficulty: 1
Why did you rate this task as you did?
file -> save file -> quit
not too hard.

Describe your process when completing this task.
method is above

Task 4: Load you "test" profile and set an alarm for one minute past the current time with a song already in the recent songs list. Now reset the alarm with a different sound.

Task difficulty: 2
Why did you rate this task as you did?

Pretty intuitive

Describe your process when completing this task.
file -> open loaded my old profile and set an alarm with a new song

What is your overall experience with the software?
Very intuitive and usable in its current form. I feel I could get this up and running fairly quickly and skip the manual.
Did you encounter any errors or bugs when using this software?
There is a bug when you Save -> Cancel... you can't quit without saving.
Do you have any recommendations for a better user experience?
Typos and the AM/PM radio buttons could be bigger.

Tester 4

Age: 20
Major (if applicable): Biomedical Engineering
Experience/skill with computers: Pretty good, have written programs

Task 1: Find the explanation of what this program does, how to work it, and the creators.

Task difficulty: 3
Why did you rate this task as you did?
It took some time to find because I did not think this as a regular task.

Describe your process when completing this task.
I looked at the screen to see if it was displayed and then went through the menu bar and quickly found it.

Task 2: Create a new profile and set an alarm for one minute past the current time. Wait for the alarm to sound and stop the alarm.

Task difficulty: 2
Why did you rate this task as you did?
It was pretty simplistic but finding the music file took some time.

Describe your process when completing this task.
I first entered 437 and realized it should have a : . I changed this and went into the file chooser to pick music.

Task 3: Save your newly created profile as "test" and quit the program.

Task difficulty: 1
Why did you rate this task as you did?
Save was under File and very easy to find.

Describe your process when completing this task.
I went to file and clicked on save

Task 4: Load you "test" profile and set an alarm for one minute past the current time with a song already in the recent songs list. Now reset the alarm with a different sound.

Task difficulty: 2
Why did you rate this task as you did?
Took some searching in the file chooser because the amount of files.

Describe your process when completing this task.
I followed the same process as Task 2 but instead I used a recent song. The recent songs were easy to find.

What is your overall experience with the software?
This is my first time using this sort of software.
Did you encounter any errors or bugs when using this software?
The alarm goes off regardless of AM or PM.
Do you have any recommendations for a better user experience?
Include the : for the user and ask if they mean to use that time.

Tester 5

Age: 21
Major (if applicable): Nutrition
Experience/skill with computers: Minimal, good with Word and Excel

Task 1: Find the explanation of what this program does, how to work it, and the creators.

Task difficulty: 3
Why did you rate this task as you did?
The task wasn't difficult itself but I didn't think to look at the menu bar.

Describe your process when completing this task.
Initially I thought the help / about was going to be a main menu button so I created a new profile and was a tad lost until I noticed the menu bar.

Task 2: Create a new profile and set an alarm for one minute past the current time. Wait for the alarm to sound and stop the alarm.

Task difficulty: 1
Why did you rate this task as you did?

I was already half way through this task on accident when I got to it. But it was very straightforward and easy to accomplish.

Describe your process when completing this task.
I already created a profile during step 1 but the buttons and prompts were self explanatory. I typed in a time and clicked set alarm and then chose a song.

Task 3: Save your newly created profile as "test" and quit the program.

Task difficulty: 1
Why did you rate this task as you did?
Since I've found the menu bar, it was simple.

Describe your process when completing this task.
I went to the menu bar and went file -> save

Task 4: Load you "test" profile and set an alarm for one minute past the current time with a song already in the recent songs list. Now reset the alarm with a different sound.

Task difficulty: 1
Why did you rate this task as you did?
It was quite straightforward and prompted easily by the button options.

Describe your process when completing this task.
Open existing profile -> choose test -> set alarm -> turn off alarm -> create new alarm choosing new song -> set alarm.

What is your overall experience with the software?
It was easily navigated and presented the user with the necessary options for the most part.
Did you encounter any errors or bugs when using this software?
No real bugs or errors.
Do you have any recommendations for a better user experience?
An option to return to the main menu.

Software Testing Conclusions

Overall, the user experience was simple and relatively good through all five tests. We learned that some tasks were not intuitive to everyone and we changed some of our layout and forms to make it simpler. Also, a couple of the testers found bugs and errors in the program. This was particularly helpful because it is hard in itself to test a GUI. From their input, we were able to fix a lot of the bugs we would otherwise not catch.

Future Work

This section will cover features that we want to implement but didn't have time in the senior project class. These feature are not necessary for the main functionality of our project but would improve the user experience.

Sleep Mask Upgrades

The sleep mask is not yet at the point where a user would be able to sleep comfortably while wearing it. Two further steps can be taken to rectify this. The first is to attach the Xbee module directly to the sleep mask, rather than have it on a breadboard. Along with this, using a small profile battery that can also be attached directly to the sleep mask will get rid of all wires and make the mask a completely stand-alone device. This is necessary if the device is to be used in real situations. The battery must be able to provide a peak current of approximately 60mA, and be able to last throughout the night while the device is running. This is not a difficult task, as the mask draws less than 3mA on average, and even this can be reduced by lowering the power of the emitter.

Finally, in order to reduce the chance of damaging the electronics on the mask, the front of the mask can be covered by a flexible substance, such as foam. This will also serve to make sure the electronics can't scratch the user or items such as a pillow.

Mobile Application

Every new idea and product appears on the mobile front, both Android and iPhone. The disadvantage of only having a computer based product is the user has to keep his or her computer on all night for the alarm to work. This could be undesirable for a lot of users. If we move the application to a mobile device, the user could be more inclined to use our product. Another advantage is we could keep all sleep data and profiles in a cloud. This will make it so that users can use it with any device they have.

For an Android development, it will be fairly simple to port the program onto the environment since Android development is in Java. All the Swing will have to be taken out and replaced with Android XML to construct the GUI. Instead of using a third party MP3 class to play music, we can utilize the MP3 Player class in Android. The UI will be more spread out and each screen described in the software user requirements will be consolidated or its own screen. The application will most likely be free since they would have to order the XBee chip to run the application.

For an iPhone development, it will be more of a challenge to port the program. The upside is that iPhone has a lot of nice APIs that the Android environment does not have. The iTunes API and others will make it very easy to create the application we want. Also, the iPhone platform makes it very easy to create a layout unlike the Android XML. Again, the application will most likely be free since they would have to order the XBee chip to run the application.

For both we need to come up with an interface to attach the XBee chip to the mobile device. There are a couple of solutions to this problem.

First would be to bypass using the XBee chip all together and use the mobile devices capabilities to get the data. This could happen through two ways: Bluetooth or using a server. Bluetooth could be used to receive, however it tends to be slow and sometimes unreliable. Also, polling at every second or two would drain the battery of the mobile device. The next solution is to use an intermediate server. It would be fast and able to store many data packets as well as profiles. The downside would be the cost to keep the server up and running plus the amount of times a single user will query the server in one night.

Second, we would interface the Xbee chip with the mobile device. The most uniform way of doing this would be to use the audio microphone port to transfer the packet (Figure 32).

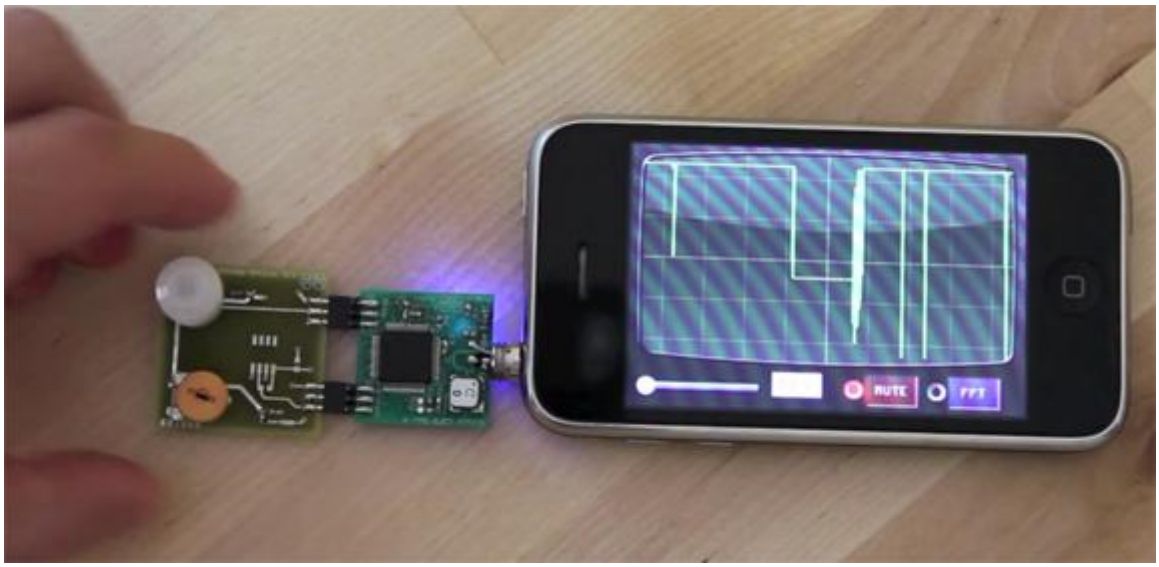

Figure 34: An example of a device using a headphone jack to transfer data.

For a preliminary prototype we would get a USB to audio converter and use the headphone jack. On the software side, we would convert it back form an audio signal into our original data. This would bypass having multiple device connectors to different devices. This also bypasses having to get a license for using Apple's unique plug on the bottom of all their devices.

UI Overhaul

Another aspect of the project that was saved for future work was the UI look and feel. As of now we have a grayscale color scheme and a sectioned set up for our different panels. The menu mar is also simplistic and missing many features.

We need to create custom components and logos for our buttons, text boxes, text, and panels. These components will have a better color scheme and maybe include custom skins that the user could specify such as Google Mail has. Java Swing might not have been the best choice for a full application and we will have to explore other GUI capable languages. As

mentioned in the previous section, the UI will have to be overhauled anyways for a mobile device. We can use this opportunity to lock down our look and feel for an actual release of our software.

Optimizing Wake-up

In the future we plan to have an optimization technique for users who use our product on a regular basis. This would require the software to save every single sleep data and calculate important data values such as average sleep cycle and ask for user feedback on sleep quality. By using those data points, we could tailor the alarm clock to each individual. We were looking into the importance of each of those values but decided to work on implementing that as a future project. Essentially, this program would learn and adapt to it's users for the best user experience possible.

Conclusion

Our initial requirements stated that we should be able to detect eye movement, transmit that data wirelessly, determine if the user is in REM sleep, and wake the user up accordingly with an alarm. All of these requirements were met and we were successful when testing the entire cohesive project, both the software and hardware side.

Bibliography

Digi International. Xbee Datasheet. http://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf

Jarvi, Keane. RXTX. 2003. http://rxtx.qbang.org/wiki/index.php/Main_Page

Java 2 Platform Std. Ed. v1.4.2. Java Swing. 2003. http://download.oracle.com/javase/1.4.2/docs/api/javax/swing/package-summary.html

Javazoom. MP3SPI. 2010. http://www.javazoom.net/mp3spi/mp3spi.html

JGraph Ltd. jgraph. 2011. http://www.jgraph.com/

Lite-On. IR detector datasheet. http://www.sparkfun.com/datasheets/Components/LTR-301.pdf

Lite-On. IR emitter datasheet. http://www.sparkfun.com/datasheets/Components/LTE-302.pdf

National Semiconductor. LM317 datasheet. http://www.national.com/ds/LM/LM117.pdf

Rapp, Andrew. xbee-api. 2011. http://code.google.com/p/xbee-api/

Polastre. Versatile Low Power Media Access for Wireless Sensor Networks. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.68.9138&rep=rep1&type=pdf

Yaso. Detection of REM Sleep by Heart Rate. http://psycho.hes.kyushuu.ac.jp/~lab_miura/Kansei/Workshop/proceedings/P-205.pdf