Multi-Function IoT Button

Design Document
Team 88 - Naveed Jamali, Daryl Lim, Jin Li
ECE 445 — Spring 2017
TA: Eric Clark
Table of Contents

1 Introduction ......................................................................................................................... 3
  1.1 Objective ....................................................................................................................... 3
  1.2 Background .................................................................................................................. 3
  1.3 High Level Requirements ............................................................................................ 4
2 Design ................................................................................................................................ 5
  2.1 Block Diagram ............................................................................................................... 5
  2.2 Circuit Schematic .......................................................................................................... 6
2.3 Block Design ................................................................................................................... 7
  2.3.1 Power Supply ............................................................................................................ 7
    2.3.1.1 Batteries ............................................................................................................. 7
    2.3.1.2 Voltage Regulator ........................................................................................... 8
  2.3.2 Wireless IO ............................................................................................................... 9
    2.3.2.1 Bluetooth ......................................................................................................... 9
  2.3.3 Local Inputs ............................................................................................................. 10
    2.3.3.1 Button .............................................................................................................. 10
    2.3.3.2 Microphone ...................................................................................................... 11
  2.3.4 Local Outputs .......................................................................................................... 12
    2.3.4.1 LCD Display ..................................................................................................... 12
    2.3.4.2 LED ................................................................................................................ 13
    2.3.4.3 Speaker ............................................................................................................ 14
  2.3.5 Control ..................................................................................................................... 15
    2.3.5.1 PCB Microcontroller Board ............................................................................ 15
    2.3.5.2 ATmega328p Chip .......................................................................................... 16
    2.3.5.3 RTC Chip ........................................................................................................ 18
  2.4 Tolerance Analysis ....................................................................................................... 20
3 Cost and Schedule ............................................................................................................ 23
  3.1 Cost Analysis .............................................................................................................. 23
    3.1.1 Labor ..................................................................................................................... 23
    3.1.2 Parts ....................................................................................................................... 23
    3.1.3 Grand Total .......................................................................................................... 24
  3.2 Schedule ....................................................................................................................... 24
4 Ethics and Safety .............................................................................................................. 26
1. Introduction

1.1 Objective

The average person’s life is far more complex than it was a hundred years ago. In this day and age, people have far more hectic schedules and must keep track of more responsibilities than they would have in the past. People are also more aware of the impact that their lifestyle has on their health. With all these new daily tasks and responsibilities, it has become harder to keep track of everything.

Our goal is to better keep track of these daily objectives by creating small, multi-purpose, IoT-enabled buttons with LCD displays. Each button can be placed in the physical location where the task or goal is to be accomplished to act as both a physical reminder and a way to see if the goal has been met. The button can be set to 3 different modes: Counter (keeps count of button presses, ex. press to record each cup of coffee to monitor caffeine intake), Checklist (timely reminder to perform a task, ex. press to show everyone in household that the dog has been fed), or scripted action (assign task to button, ex. place near bed and press to turn off all lights in home). They would connect to a device via Bluetooth to send and receive data through an iOS application, and to detect if a user is home. Our plan is to have all of features of the button to run on its own hardware rather than through the connected bluetooth device.

1.2 Background

Software applications already exist to aid in our remembering, including calendars and checklist applications such as Todoist [1]. However, these applications often fail to be flexible in regards to location and changing schedules, only able to send reminders at set times. To take location into account, most applications will at best react to a geofence around an entire building. However, tasks are often forgotten without a physical reminder in the exact
location they are to be completed. Even if you remember or are reminded to do something in one room, upon walking into the room containing the task, you may get distracted and forget why you are there due to the “Doorway Effect” [2]. This heavily studied phenomenon explains the common memory loss that occurs upon crossing physical barriers, such as when entering another room or getting into a car.

Our button would be an elegant solution to this problem. By putting a button right where the task is to be completed, the button will beep with increasing frequency and intensity as the due time approaches to draw the user into the room to complete the task. This would prevent the doorway effect mentioned above and allow some flexibility in when tasks need to be accomplished. Users could view the data regarding their daily habits collected from the buttons on their smartphones through a bluetooth connection. The buttons could also carry out actions such as sending texts or controlling lights through the smartphone, making them IoT connected. As the inspiration for this project came from the Amazon IoT (Dash) button, we plan for our button to have a similar form factor. The button should also be similarly priced (although the display, microphone, and speaker will add significant cost) and have a comparable battery life.

1.3 High-Level Requirements

- Buttons must be able to connect to an application on the phone or computer wirelessly (through Bluetooth) to allow data collected to be stored on it.
- Buttons must be able to recognize their currently desired function, be able to correctly carry it out independent of all other features (including the connection to the smartphone).
- Buttons must consume as little power as possible so as to prolong battery life, ideally able to last weeks to months.
2. Design

2.1 Block Diagram

Our project can be divided into five parts: Power, Local Inputs, Local Outputs, Control, and Wireless IO. The power supply (battery-powered) powers the button continuously regardless of time of day and should be able to maintain power for weeks to months before requiring changing. The local inputs send the designated data to control to properly handle it. The control then decides the reaction and sends the correct response to local outputs. Our wireless IO of choice (Bluetooth) connects our system to the user device to send and receive data. It allows our buttons to be initialized by an iPhone and detects if the user is away to enter a power-saving mode.
2.2 Circuit Schematic

Figure 2: Circuit Schematic

2.3 Block Design

2.3.1 Power Supply

A power supply is required for each individual button to keep them functioning as long as the user is present. For our buttons, basic batteries should suffice to power it in its entirety.
One of the key intended features of our button is to be able to last for weeks to months before a battery change, as frequent battery changes on buttons would prove tedious.

### 2.3.1.1 Batteries

Our choice for batteries is a pack of four Alkaline AA battery [11], two connected in series, in parallel to two connected in series, which gives a combined output voltage of 3 V, and a capacity of about $2779 \times 2 = 5558 \text{ mAh}$ [6].

![Figure 3: Battery Connection](image)

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output voltage must be $3 \pm 0.3 \text{ V}$</strong></td>
<td>1. Test output voltage across a load.</td>
</tr>
<tr>
<td></td>
<td>2. Ensure that the voltage is within the range of $2.7 - 3.3 \text{ V.}$</td>
</tr>
<tr>
<td><strong>Must power the device for at least 1 month</strong></td>
<td>1. Calculate battery life from capacity of batteries and power required to power product.</td>
</tr>
<tr>
<td></td>
<td>2. Run product on set of batteries for 5 days.</td>
</tr>
<tr>
<td></td>
<td>3. Estimate remaining battery capacity by testing voltage with a multimeter across a load</td>
</tr>
</tbody>
</table>

*Table 1: Requirement and Verification for the Batteries*

The ATmega328P runs at 3.3V, 3.58 mA when active, and at 3.3V, 0.0045 mA when powered down. [3]

Assuming the most power consuming scenario of the ATmega328P being active around the clock, we have power consumption of:
3.58 mA x 24h/day = 85.92 mAh/day

With our initial battery capacity of 5558 mAh, we find that the number of days our batteries should be able to supply power to the ATmega chip should be:

5558 mAh / 85.92 mAh/day = 64.69 days

Not only does this fit our requirement of the battery life lasting from weeks to months, but whenever the user leaves the house or commands the button to be inactive, the ATmega328p would power down, shifting to using 0.0045 mA instead. Given a scenario of a single individual working a standard 8-hour work day, the ATmega328p would power down during that time, giving us a power consumption instead of:

0.0045 mA x 8h/day = 0.036 mAh/day
3.58 mA x 16h/day = 57.28 mAh/day

Or a total of 57.28 + 0.036 mAh/day = 57.316 mAh/day, or a battery lifespan of:
5558 mAh / 57.316 mAh/day = 96.97 days

### 2.3.1.2 Voltage Regulator

The voltage regulator must provide a steady supply of constant voltage to all components. It will step up the voltage from 3V to 3.3V, as all components in our design operate at 3.3V input.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| **Output voltage must be 3.3 ± 0.3 V**           | 1. Use an oscilloscope to read output over a time period of 5 minutes.  
2. Determine the highest and lower output from the regulator. |
| **Must provide a steady voltage output whose error should ≤ 5%** | 1. Use an oscilloscope to read output over a time period of 5 minutes.  
2. Determine the fluctuation error. |

*Table 2: Requirement and Verification for the Voltage Regulator*

In the figure below, we can see that Pololu regulator we chose has good efficiency of (power out)/(power in). It can be observed from the three different curves that the efficiency increases as the input voltage increases. At around 100 mA, when
the input voltage is 2.4V, the efficiency is around 85%. With our input voltage of 3V, we expect the efficiency to be even higher.

![Graph showing regulator efficiency vs output current](image)

**Figure 4: Regulator Efficiency vs Output Current**

Referenced from “Pololu 3.3V Step-Up Voltage Regulator U1V10F3” [7]

### 2.3.2 Wireless IO

#### 2.3.2.1 Bluetooth

A Bluetooth chip (hm11) on the button must be able to communicate effectively with a phone (or whatever other device the user intends to use).

The voltage supply should be 3.3 V, and the chip should communicate with the rest of the PCB board using UART protocol.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| The Bluetooth chip must be able to detect if the user device is within the premises. If not, it must be able to communicate this to the rest of the button to enter a low-power mode. | 1. Implement simple program on ATmega to turn LED on and to turn it off when device leaves Bluetooth radius.  
2. Observe LED as device leaves and verify it responds as intended.  
3. While LED is active, use a multimeter to measure current output of ATmega. |
4. Verify a current output of 3.58mA ± 0.3mA.
5. While LED is inactive, again measure current with a multimeter.
6. Verify a current output of 0.0045mA ± 0.0003mA.

If the button detects a user, it must be able to connect to the device to send and receive data.

1. Set up a simple counter program on the Atmega.
2. While LED is active, press button.
3. Observe on device and verify that it’s registering counter increments accurately.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Current (mA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmit</td>
<td>15</td>
</tr>
<tr>
<td>Receive</td>
<td>8.5</td>
</tr>
<tr>
<td>Deep Sleep</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Table 3: Requirement and Verification for the Bluetooth Module

Table 4: Current Drawn in Different Modes

2.3.3 Local Inputs

2.3.3.1 Button

We intend to adapt from or build on the design of the Amazon dash button. It should register the press from a user, and update relevant information (ex. that the user has taken his/her daily medicine). The button should stay at a logic high when not pressed, and should go to a logic low when pressed.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| The button should stay at a logic high at 3.3 ± 0.3 V, and a logic low when pressed at 0 ± 0.3 V | 1. Supply the button with desired voltage
2. Press down the button                               |
3. Measure the voltage across (should be 0 ± 0.3 V)    |
4. Read the corresponding signal                       |
sent to the ATmega328p chip - check if low

| Button should be properly debounced | 1. Create simple program that increments a counter by one for each button press  
2. Verify that 1 button press corresponds to the counter increasing by exactly one |

Table 5: Requirement and Verification for the Button

For the purpose of debouncing our button, we intend to utilize the following circuit [13], where R1 and the button are the only components in the original non-debounced circuit.

![Debouncing Circuit](image)

**Figure 5: Debouncing Circuit**

### 2.3.3.2 Microphone

The purpose of the microphone is to capture voice commands and transfer analog signal to the ATmega328p chip, where the analog signal will be converted to digital, and processed for word recognition.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| The microphone must be able to accurately capture speech within a 10 foot radius. | 1. Set microphone to receive audio input.  
2. Stand at or within 10 feet from the mic and make audible sound, checking to see if it’s detected.  
3. Repeat five times in five different locations (still at or near microphone) |
The microphone must be able to transmit captured speech signals to Arduino for processing. Verify signal in Arduino

Table 6: Requirement and Verification for the Microphone

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| The LCD display must be able to display the right message with minimal power consumption. | 1. Implement simple program to output desired phrase (ex. Hello World!)  
                                      2. Observe display for desired phrase.  
                                      3. Test three times with three |
4. Observe display: should display task on button press (the next task if set to Toggle)

Table 7: Requirement and Verification for the LCD Display

<table>
<thead>
<tr>
<th>Pin</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Vss</td>
<td>GND</td>
</tr>
<tr>
<td>2</td>
<td>Vdd</td>
<td>Power Supply</td>
</tr>
<tr>
<td>3</td>
<td>E</td>
<td>Enable</td>
</tr>
<tr>
<td>4</td>
<td>DB0 - DB7</td>
<td>Data Bus Software</td>
</tr>
<tr>
<td>5</td>
<td>LED +</td>
<td>LED unit anode</td>
</tr>
<tr>
<td>6</td>
<td>LED -</td>
<td>LED unit cathode</td>
</tr>
</tbody>
</table>

Table 8: LCD Display Key Pin Configurations

2.3.4.2 LED

A visual color display that must be able to react when time indicated for task completion approaches. Since our LCD display has three colors available, we are going to utilize those three color capabilities. Blinks on and off until button is pressed or command is given. The display takes a forward voltage of 2.2 V when color red should be display, which can be taken care of by the PWM.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| LCD display must display green and blue colors when voltage supply is 3.4 ± 0.2 V | 1. Supply the LCD display with voltages ranging from 3.2 to 3.6 V  
2. Visually verify that the two colors are displayed |
| LCD display must display red color when voltage supply is 2.2 ± 0.2 V | 1. Supply the LCD display with voltages ranging from 2.0 to 2.4 V  
2. Visually verify that the red |
2.3.4.3 Speaker

A speaker that must be able to react when time indicated for task completion approaches. Beeps until button is pressed or command is given.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>The speaker must be able to cover at least a frequency range of 1 - 1.5 kHz</td>
<td>1. Perform a frequency sweep</td>
</tr>
<tr>
<td></td>
<td>2. Measure the lowest and highest frequency using FFT</td>
</tr>
<tr>
<td></td>
<td>3. The range of 1 - 1.5 kHz must be covered</td>
</tr>
<tr>
<td>Able to beep with “Encouraging” and “Discouraging” sounds with intensity &gt; 60 dB</td>
<td>1. Verify that there are indeed two distinct sounds</td>
</tr>
<tr>
<td></td>
<td>2. Measure the range of output intensities</td>
</tr>
<tr>
<td></td>
<td>3. The highest intensity should &gt; 60 dB</td>
</tr>
</tbody>
</table>

Table 10: Requirement and Verification for the Speaker

2.3.5 Control

2.3.5.1 PCB Microcontroller Board
In the control unit, we will design a microcontroller using AutoDesk EAGLE, which will comprise of a Real-Time Clock (RTC) timekeeping chip, a ATmega328p processing chip, and circuit elements designed by us.

In the control unit, we will implement one large state machine used in the operation of the button. The state machine will have three main branches for each mode the button is set to. State transitions will react to button inputs as well as the output of the serial Real-Time Clock (RTC) timekeeping chip to be mounted on the PCB.

The PCB microcontroller should be powered by our batteries with the voltage stabilized by the voltage regulator, and should be the center of connection for all functional components in our design: the microphone input, button input, speaker output, LCD display, LED display, and bluetooth connection.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| PCB microcontroller must be programmable to set the button to a certain mode and label | 1. Set up three test cases for the three different modes.  
2. Verify that switching between the three modes yields the correct response when button is pressed. (counter: increments a counter, checklist: resets RTC to go off in 24 hours, toggle: switches to next string of characters on LCD display) |
| PCB microcontroller must be able to correctly read the inputs (button press, bluetooth) | 1. Create simple counter program on the ATmega.  
2. Configure so that each button press increments the counter by exactly one.  
3. Verify that the counter increments on button press.  
4. Code for the LED to light up when a device is in range of... |
the Bluetooth.
5. Verify that when the device leaves the Bluetooth range, the LED turns off.

<table>
<thead>
<tr>
<th>PCB microcontroller must set proper outputs to LCD display, LEDs, Speaker, and iPhone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Set the LED to blink on and off.</td>
</tr>
<tr>
<td>2. Visually confirm that the LED blinks as intended.</td>
</tr>
<tr>
<td>3. Observe data on oscilloscope and confirm the LED blinks (sets voltage to HI 3-5V for 1 second before returning to LOW 0-1.5V).</td>
</tr>
<tr>
<td>4. Hard code a test string of characters</td>
</tr>
<tr>
<td>5. Visually confirm that the LCD display displays the characters as intended.</td>
</tr>
<tr>
<td>6. Set the speaker to output two audible frequencies.</td>
</tr>
<tr>
<td>7. Verify the outputted frequencies are outputted with a 50 kHz tolerance.</td>
</tr>
</tbody>
</table>

Table 11: Requirement and Verification for the PCB board

2.3.5.2 ATmega328p Chip

We intend to mount an Arduino chip (ATmega 328p) onto our PCB, in order to utilize its processing powers to run word recognition algorithms on extracted features of the audio input. Our main approach is to pre-train the weights and biases of a fully-connected neural network on a laptop for specific words, and deploy those weights and biases onto Arduino for real-time recognition. Our alternative approaches can be a CNN, HMM (Hidden Markov Model) for the words to recognize, or even a simple K-Nearest-Neighbor classifier with a pre-recorded database for feature matching.

Below are the requirements and verifications of the control module in our block diagram design.
**Table 12: Requirement and Verification for the ATmega328p Chip**

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arduino chip must be able to retrieve sound from the microphone</td>
<td>1. Input an audio signal into the connected microphone.</td>
</tr>
<tr>
<td></td>
<td>2. Compare the incoming audio signal in the time domain in Arduino to the same audio signal displayed in time domain in MATLAB.</td>
</tr>
<tr>
<td></td>
<td>3. Verify that both signals are the same.</td>
</tr>
<tr>
<td>Arduino chip must be able to carry out algorithmic operations on audio input</td>
<td>1. Input an audio signal into the connected microphone.</td>
</tr>
<tr>
<td></td>
<td>2. Compare the output results of algorithmic operations on input in Arduino chip and MATLAB.</td>
</tr>
</tbody>
</table>

In exploring different methods for word recognition, we compared four different kinds of classifiers for a fully-connected neural network.

Below is a convergence plot for four different kinds of classifiers, on a dataset of consonants from an experiment by Bowon Lee et al., 2004 [8]. As can be seen, a logistic classifier or a SVM classifier is more ideal than a linear classifier or a simple perceptrons classifier in terms of how quickly the classification error plot converges.

However, it is worth noting that the four learning rates used for gradient descent are different from each other. This is because if we use the same learning rate, it might be too big for some classifiers, and the gradient “steps over” the error metric’s local minima. If a chosen learning rate is too small, then convergence is slow; however if a chosen learning rate is too large, then the error rate might not converge at all due to the large steps taken [9]. With that trade-off, we chose learning rates for each classifier accordingly.
The following table shows the error metric (different from the classification error rate) of the four different classifiers we explored, and their corresponding gradient equation for gradient descent.

<table>
<thead>
<tr>
<th>Error Metric</th>
<th>Gradient Equation</th>
<th>Fixed Learning Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>linear</td>
<td>$E = \sum_i ((t_i - y_i)^2) = \sum_i ((t_i - g(w'x_i + b))^2)$</td>
<td>0.00001</td>
</tr>
<tr>
<td></td>
<td>$g(a) = a$</td>
<td></td>
</tr>
<tr>
<td>logisti c</td>
<td>$E = \sum_i ((t_i - y_i)^2) = \sum_i ((t_i - g(w'x_i + b))^2)$</td>
<td>0.0001</td>
</tr>
<tr>
<td></td>
<td>$g(a) = \frac{1}{1+\exp(-a)}$</td>
<td></td>
</tr>
<tr>
<td>Perceptron</td>
<td>$E = \sum_i (\text{max}(0, -(w'x_i + b) \cdot t_i))$</td>
<td>0.000005</td>
</tr>
<tr>
<td></td>
<td>$\frac{dE}{dW_j} = \sum_{\text{errors}} (-x_i \cdot t_i) [16]$</td>
<td></td>
</tr>
</tbody>
</table>
Table 13: Theoretical Basis for Four Classifiers Explored

* t denotes true labels, y denotes predicted labels, g conventionally denotes a non-linearity, w denotes weights, b denotes biases, x denotes input data, and C is a tunable constant.

** Code for carrying out the above procedures and calculations adapted from team member Jin Li’s work in ECE 544 Fall 2016.

2.3.5.3 RTC Chip

The Real-Time Clock (RTC) we chose is the DS1339A chip by Maxim Integrated. The purpose of incorporating this clock chip is to keep track of the time in a day for setting reminders, having a time span reference when counting, etc.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>The RTC must be able to recognize time set by device</td>
<td>Compare time on RTC to a regular clock, error should be &lt;3 seconds</td>
</tr>
<tr>
<td>The RTC must be able to recognize 12:00 AM as a deadline for buttons to reset Counter</td>
<td>Within 5 seconds after 12:00 AM, read the counter, which should be 0</td>
</tr>
</tbody>
</table>

Table 14: Requirement and Verification for the RTC Chip

Figure 9: RTC Chip Block Diagram
This chip will communicate with the rest of the PCB board via I2C serial interface; it takes an input voltage from 1.8 to 5 V.

### 2.4 Tolerance Analysis

A critical feature of our project is the voice recognition feature; when a deadline is approaching and the speaker is making noises to remind us of that deadline, we would like to turn off the noise by uttering a word such as “stop”, which will be picked up by the microphone and processed for word recognition.

This feature produces many risks to the completion of our project. For example, the training process requires a lot of parameter tuning. One of the parameters that needs the most tuning would be the fixed learning rate of gradient descent. As mentioned in the design section of the ATmega328p chip, we need to choose a learning rate not so large that the gradient descent process misses the local minima, and at the same time not so small that it causes the gradient descent to be slow.

The gradient descent is described by the equation:

\[ W^{\text{updated}} = W^{\text{old}} - \gamma \frac{\partial \text{Error Metric}}{\partial W} \]

where \( W \) is the weight vector, \( \gamma \) is the learning rate, and the error metric is specific to the type of classifier.

An observable example of the effect of choosing a learning rate too large can be seen in the convergence figure below, in the linear classifier plot and the perceptron classifier plot.
In this example, we use the same learning rate 0.0001 for all four classifiers instead of the fine tuned values for each of the classifiers (shown in the table below).

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Tuned Fixed Learning Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>0.00001</td>
</tr>
<tr>
<td>Logistic</td>
<td>0.0001</td>
</tr>
<tr>
<td>Perceptron</td>
<td>0.000005</td>
</tr>
<tr>
<td>SVM</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

Table 15: Desirable Learning Rates for Each Classifier

Another potential issue in the training process is that a low Signal-to-Noise-Ratio (SNR) from unfiltered noise could cause the pretrained model on quiet samples...
to fail. If that issue should arise, we will use DSP methods to do more noise filtering.

The overall functionality of the voice recognition feature depends on multiple modules in our design. Assuming that the microphone is well calibrated, the voice will pass from the microphone to the analog-to-digital built-in converter (ADC) on the ATmega328p chip, where we will process each frame of stored input data. The microphone can pick up sounds within the frequency range of 100 ~ 10,000 Hz [14], while for human speech, “Normal voice range is about 500 Hz to 2 kHz” [15], so the speech frequency should be covered by the frequency range the microphone can pick up. Assume the ADC runs at 125 kHz, which is a high enough sampling frequency for speech, we will need to process frames of input sound with length 0.5 second = 500 ms, and a step size of 30 ms between frames. With each input frame, we will process the input data as follows:

![Diagram](image)

*Figure 11: Structure of Our Word Recognition Process*
Where the index of $x$ (1 through $n$) denotes time frame, the corresponding indexed $w$ (1 through $n$) is the weight on that time frame, and $b$ is the bias. We pass the scalar sum, which is

$$X^T W + b$$

* $X$ and $W$ are both $1xN$ vectors here

into one of the four explored classifiers, and yield an output, which in our case should be one of the two possible classes: positive (the frame contains our target word), or negative (the frame doesn’t contain our target word). If the output is positive, we will send a signal from the PCB board to the speaker, and turn off the sound.

3. Cost and Schedule

3.1 Cost Analysis

3.1.1 Labor

If we assume a reasonable salary to be $30/hour, which is common for internships in the Bay Area, then the total cost for developing our design project should be:

$30 \text{ /hour} \times 3 \text{ (team members)} \times 2.5 \times 8 \text{ (hours/member/week)} \times 13 \text{ (weeks)}$

$= \$23,400$

3.1.2 Parts

<table>
<thead>
<tr>
<th>Description</th>
<th>Manufacturer</th>
<th>Part#</th>
<th>Quantity</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical Button + shell</td>
<td>3D Printing it ourselves</td>
<td>N/A</td>
<td>1</td>
<td>$2</td>
</tr>
<tr>
<td>Microphone</td>
<td>Challenge Electronics</td>
<td>CEM-C9745JA D462P2.54R</td>
<td>1</td>
<td>$5.95</td>
</tr>
<tr>
<td>Speaker</td>
<td>CUI</td>
<td>CDMG15008-</td>
<td>1</td>
<td>$2.58</td>
</tr>
</tbody>
</table>
## 3.1.3 Grand Total

**Grand total = Labor + Parts = 23,400 + 41.94 = $23,441.94**

### 3.2 Schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Jin</th>
<th>Daryl</th>
<th>Naveed</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/20/2017</td>
<td>Work out the details of the design document</td>
<td>Work out the details of the design document</td>
<td>Work out the details of the design document</td>
</tr>
<tr>
<td>2/27/2017</td>
<td>Order parts: microphone, speaker, bluetooth HM11, ATmega328p; participate in the button design for 3D printing</td>
<td>Order parts: voltage regulator, AA batteries, LCD display, RTC chip; participate in the button design for 3D printing</td>
<td>Design button for 3D printing</td>
</tr>
<tr>
<td>Date</td>
<td>Task Description</td>
<td>Task Description</td>
<td>Task Description</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>3/6/2017</td>
<td>Connect microphone to the ADC module on the ATmega328p chip; connect speaker into the circuit; start writing code for word recognition</td>
<td>Work on schematic + PCB design in EAGLE</td>
<td>Print button using 3D printing; participate in schematic + PCB design</td>
</tr>
<tr>
<td>3/13/2017</td>
<td>Further develop code for word recognition; test speaker characteristics</td>
<td>Finalize PCB design and submit request to ECE shop</td>
<td></td>
</tr>
<tr>
<td>3/20/2017</td>
<td>Spring Break</td>
<td>Spring Break</td>
<td>Spring Break</td>
</tr>
<tr>
<td>3/27/2017</td>
<td>Test the algorithmic operations on the ATmega328p chip</td>
<td>Work on PCB; test power</td>
<td>Work on PCB</td>
</tr>
<tr>
<td>4/3/2017</td>
<td>Finish all code on word recognition; make sure all connections related to word recognition is functional</td>
<td>Finish all installations; check Bluetooth + PCB connections</td>
<td>Finish all installations; check PCB connections to all other parts</td>
</tr>
<tr>
<td>4/10/2017</td>
<td>Test and debug whole system</td>
<td>Test and debug whole system</td>
<td>Test and debug system</td>
</tr>
</tbody>
</table>
| 4/17/2017  | - Work on final paper  
- Mock demo                                           | - Work on final paper  
- Mock demo                                                                 | - Work on final paper  
- Mock demo                                                                 |
| 4/24/2017  | - Work on final paper  
- Present projects                                      | - Work on final paper  
- Present project                                                                | - Work on final paper  
- Present project                                                                |
| 5/1/2017   | - Turn in lab notebooks  
- Return lab                                             | - Turn in lab notebooks  
- Return lab                                                                              | - Turn in lab notebooks  
- Return lab                                                                              |
4. Ethics and Safety

Our button in and of itself is relatively low risk as it is intended to run on only a single battery as a power supply. The main points of concern would have to be more ethically related than physically - concerns such as users being dangerously reliant on the buttons, such as relying on solely the button to keep track of intake of medicine in which case malfunctions could be deadly. In accordance with the IEEE Code of Ethics, #1: “to accept responsibility in making decisions consistent with the safety, health, and welfare of the public” [4]. Also, if the use of an emergency help application were to malfunction and the user believes help is on the way when it isn’t, this would prove to be fatal as well. As stated in the ACM Code of Ethics and Professional Conduct #1.2: “One way to avoid unintentional harm is to carefully consider potential impacts on all those affected by decisions made during design and implementation.” [5]

Solutions to this issue, on top of making sure the button works as intended as often as possible through heavy testing, is to include cautions to not be overly reliant on the buttons in case malfunctions occur. Additionally, in the case of the emergency help application, we could include a message on the LCD screen assuring that the message ad been sent.
Citations


[2] Z. Lawrence and D. Peterson, "Mentally walking through doorways causes

-low-power-how-to-run-atmega328p-for-a-year-on-coin-cell-battery/. [Accessed:
21-Feb- 2017].

http://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed:
08-Feb- 2017].

[Accessed: 08- Feb- 2017].


