Multi-Function IoT Button

Final Report

Team 88 - Naveed Jamali, Daryl Lim, Jin Li

ECE 445 — Spring 2017

TA: Eric Clark

Abstract

We have created a multi-purpose, Internet-of-Things (IoT) button capable of acting as a habit counter, single-task checklist, or as a terminal to carry out an IoT action (such as turning off all “smart” lightbulbs in a home. Our device aims to be an effective tool to remind users to complete vital tasks, such as taking medication, and to monitor good and bad habits such as water/caffeine intake.

Our device uses a push button and a microphone (for speech recognition) as inputs, and a speaker and RGB backlit LCD display as outputs. We used an ATmega328p as our microprocessor, and an HM-11 Bluetooth module to allow data transfer between the button and a host device (smartphone).

Table of Contents

1 Introduction

1.1 Purpose

Our goal is to better keep track of daily objectives and encourage habitual tasks by creating small, multi-purpose, IoT-enabled buttons with LCD displays. The buttons would be mounted in the physical location where the task or goal would take place (ex. above the sink when the task involves the daily intake of water) and act as both a physical reminder and a way to determine if the goal has been met or limit has been reached. The buttons also would be a proposed solution to the “Doorway Effect”, a phenomenon of memory loss occurring upon the crossing physical barriers, often causing one to forget why they crosses the barrier in the first place. [1] Being able to see a physical object again would act as a reminder, especially with tasks you are attempting to drill into habit that you might not be used to remembering to do constantly. It could also be useful in reminding users about daily intake in medicine or being able to quickly dial for help through the action mode, acting as a useful aid - especially for the elderly.

1.2 Features

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). It can connect to a device (smartphone) via Bluetooth to send and receive data through a Bluetooth terminal and detect user presence. When the user is not present, the device enters a low power mode by disabling all notification and high power-consuming features. The button is also designed to produce tone through a speaker as the due time for the button approaches to draw the user into the room to complete the task. Additionally, it has built in voice recognition to be able to determine when the word “off” is uttered with decent accuracy and respond by disabling audio reminders in response.

2 Design

2.1 Block Diagram

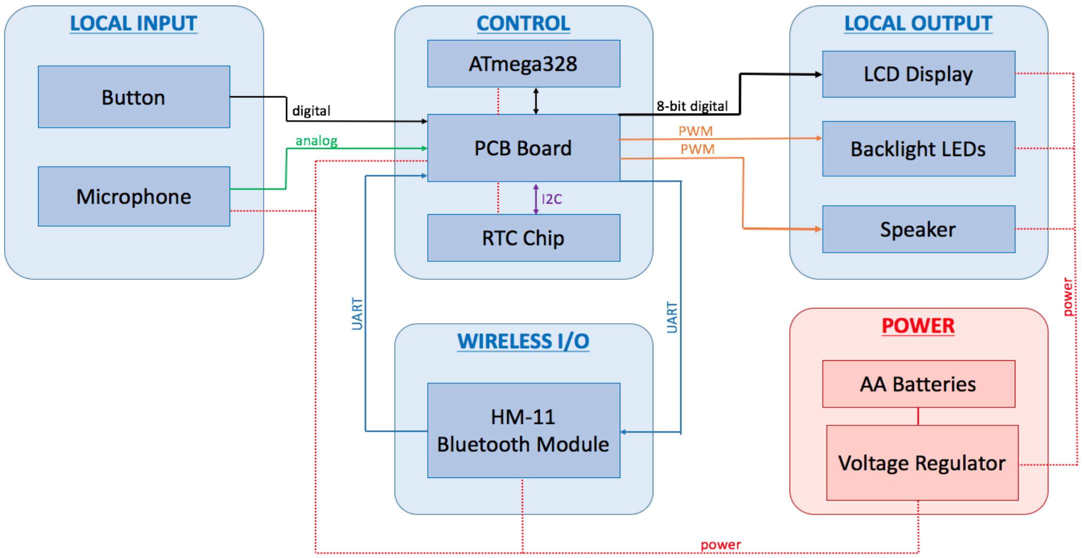

Figure1: Block Diagram

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.

See Appendix D for pictures of the physical layout of our button. Full circuit schematics and PCB layouts can be found in Appendix B and C, respectively.

2.2 Power Supply

2.2.1 Batteries

Our choice for batteries is a pack of four Alkaline AA batteries, two connected in series and two in parallel which gives a combined output voltage of 3V, and a capacity of about 2779 x 2 = 5558 mAh [2].

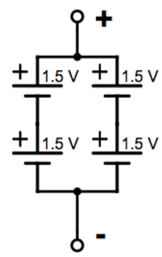

Figure 2: Battery Connection

After calculations of all power consumptions of individual modules and parts, we determined individual power requirements by finding individual current draw of each module and determining their activity required throughout a regular day. We determined these power requirements to be 85.92 mAh/day for the ATmega328P, 31.2 mAh/day for the RTC chip, 0.1111 mAh/day for the LCD, an average of 14.6 mAh/day for the HM-11 Bluetooth chip (see Appendix E for calculations), and 1.08 mAh/day for the speaker. The voltage regulator's quiescent current is negligible, as it is <<1 mA for input voltages close to 3.3V.

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

5558 mAh / (85.92+31.2+14.6+1.08) mAh/day = 41.9 days

Not only does this fit our requirement of the battery life lasting from weeks to months, but whenever the user leaves the house 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

3.58mA 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 +31.2+14.6+1.08) mAh/day   53.3 days

2.2.2 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.

In Figure 3, we can see that Pololu regulator we chose has a very high efficiency with inputs close to 3.3V. 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.

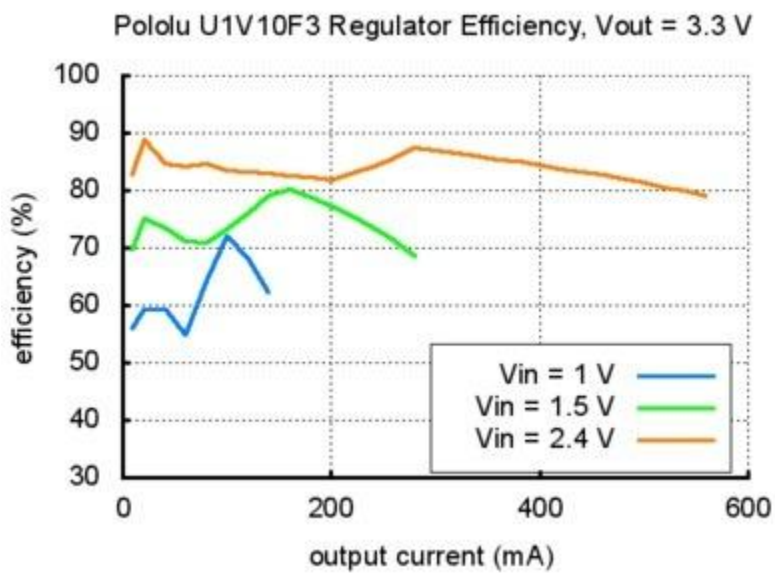

Figure 3: Regulator Efficiency vs Output Current [3]

2.3 Wireless IO

2.3.1 Bluetooth

The purpose of the Bluetooth module is to:

1. Transmit and Receive data from a host device
   a. Receive programming instructions to set time, mode, label, etc.
   b. Transmit data from button presses to be recorded in host device application
2. Detect Presence of User
   . If user is out of range, then enter low-power mode. (LEDs off, Speaker disabled, machine learning voice recognition disabled).

The HM-11 connects to the ATmega328 using the UART protocol. The SPP profile would be most appropriate for our purposes. The input voltage to the chip is the same as the rest of our circuit (3.3V).

The data sheet on the chip states that its maximum range is 30 meters in an open space. For our purposes, we require that the button recognize the user within 15 meters in an open space and 8 meters going through 2 walls. This is reasonable because, as stated above, one of the main purposes of our Bluetooth module is to enter a low power mode if the user is to far away to hear the speaker or issue commands to the microphone. 15 meters is a fairly large distance away from the button where the user probably would not hear it nor want to interact with it. It also logically follows that if walls attenuate sound and decrease audible range, it would be alright if the bluetooth range decreases as well. Therefore, we set the range requirement given two average apartment walls (drywall, not concrete) to be 8 meters, which is the distance from the living room to two rooms down in a typical apartment.

In order to conserve power, we will only activate the bluetooth chip for only a fraction of the the time and leave it in a low-power sleep mode as long as possible. The calculations for how long the HM-11 needs to be active to transfer the required amount of data in the proper amount of time for our design is located in Appendix E. The result of these calculations show that we only need to keep the HM-11 active for 5ms every 5 seconds, and that the resulting average current draw under these conditions would be 0.608mA.

These calculations were made assuming the button received all its programming information at once, which we later changed to having the button get programmed step by step through a series of prompts.

2.4 Local Inputs

2.4.1 Button

The button should register the press from a user, and update relevant information (ex. for a counter it increments, for a checklist it checks off).

2.4.2 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. The microphone can pick up sounds within the frequency range of 100 ~ 10,000 Hz [4], while for human speech, “Normal voice range is about 500 Hz to 2,000 Hz” [5], so the speech frequency should be covered by the frequency range the microphone can pick up.

As the microphone’s natural voltage is small (found to be in mV) and cannot be read easily, we need an amplifier to amplify the voltage to a discernible range of 1V - 3.3 V. Since the sensitivity of the microphone is at -46 dB, we will need an amplifier of 46 dB to amplify the output voltage to 1V, assuming that the mic will be receiving 94 dB-SPL signals. However, we must recognize that the input intensity to the mic will not always be as high as 94 dB-SPL when capturing speech. When a person is shouting at 1 ft away from the microphone, the voice level is 88 dB-SPL; when a person is 12 ft away from the microphone talking at a normal volume, the voice level is 48 dB-SPL [6]. Assuming the voice level can be as low as 48 dB-SPL, we need an additional 94 - 48 = 46 dB. Therefore, we need a total of 46 dB (from voice) + 46 dB (from speaker) = 92 dB for our amplification of sound.

2.5 Local Outputs

2.5.1 LCD Display

An LCD display that must be able to display various pre-programmed instructions (“Enter mode:”, “Enter goal:”), labels (“take meds”), etc. corresponding to its current functionality.

2.5.2 Color Display

A visual color display on LCD must be able to react when time indicated for task completion approaches. Since our LCD display has three backlight colors available, we will utilize two colors to act as our “Encouraging” and “Discouraging” colors. Blinks on and off until button is pressed or command is given. The display takes a forward voltage of 3.3 V and the changing of colors can be taken care of by the PWM.

2.5.3 Speaker

The speaker must be able to remind users when time for task completion approaches. For example, the speaker beeps until button is pressed.

For intensity ranges, we choose a decibel level that surpasses that of human speech (60 dB) to be able to make enough of an impression to spur users into action, and lower than a level that could possibly damage hearing after long exposure (85 dB). We consider this would be at an appropriate level as an alarm clock usually functions at 80 dB.

2.6 Control

2.6.1 Microcontroller

In the control unit, we use a microcontroller put together on AutoDesk EAGLE, which will comprise of a Real-Time Clock (RTC) for timekeeping, a ATmega328p chip for its processing powers, and circuit elements designed by us.

One large state machine is used in the operation of the button. The state machine has three main branches for each mode the button is set to: counter, checklist, and scripted action. State transitions reacts to button inputs as well as the time output of the serial Real-Time Clock (RTC) timekeeping chip mounted on the PCB.

The microcontroller is 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.

The voice recognition process is carried out on the ATmega328p chip.

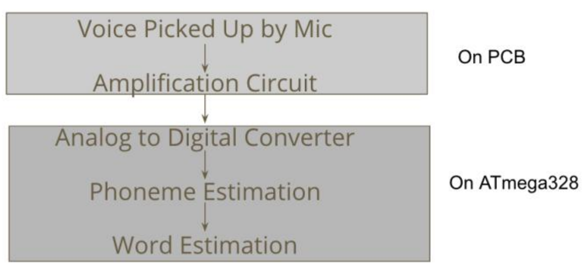

Figure 4: Voice Recognition Flowchart

The estimation method of phonemes makes use of an Arduino library: µspeech [7]. The premise of this method is that the fricative sounds such as /f/, /s/, and /sh/ have an inherent tendency to be more like noise and have higher frequencies, which zigzag a lot and look very chaotic in the time vs air pressure graph; meanwhile, vowels such as /a/ do not have such a tendency. Based on this difference between vowels and fricatives, we can adopt an evaluation metric:

$$C = |df(x)dx| / |f(x)| \qquad \textit{Eqn (1)} \text{ [7]}$$

where f is the amplified air pressure over time, and x is time. Fricatives such as /f/, /s/, and /sh/ should have a high C value, vowels such as /a/ have a low C value, and voiced fricatives such as /v/ have a value in between. Therefore, we just need to find the thresholds that separate the three groups of phon the amplification circuit (op-amp gains, circuit element precision, etc).

By speaking different phoneme groups into the microphone and observing the C values, we can find the thresholds, which depend on the microphone specs and the predetermined silent threshold, an estimation of the single phoneme is added to an initially empty string. When the analog voltage input drops back down below that silent threshold, we use the Levenshtein Distance to compare the accumulated string to each of the preset word patterns [8], one of which indicates the pattern for “off”.

The timing requirements for our inputs/outputs are purposefully lenient since humans react in orders of magnitude slower than a microprocessor's clock cycle, and for the most part we our operations are not computationally intensive.

2.6.2 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.

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. It has an internal oscillatory circuit with a preset load capacitance of 6 pF and is designed to take a standard 32.768kHz Quartz crystal.

3 Design Verification

3.1 Power Supply

3.1.1 Voltage Regulator

Our original requirement was that the voltage regulator could supply enough current at a steady 3.3V when all of the circuit elements were at maximum power consumption at the same time. From our power calculations (2.2.1), this number was estimated to be 194 mA.

However, we learned that although the maximum source current from VCC is 200mA and the maximum sink current to GND is 400mA, the absolute max current per I/O pin is only 40mA. This conflicts greatly with the 173mA current drawn by the speaker with the series trimpot set to the highest setting, and as a result our button was failing to play a sound and brighten the display at the same time. We solved this issue by placing a small delay between the light and sound, but falsely attributed the problem to the voltage regulator. Upon learning about this additional constraint, further testing showed that our regulator is in fact capable of providing a steady voltage at all possible current draws of our device. Future revisions of our designs should therefore power the speaker through a transistor instead or operate at a lower current draw to avoid damage to the ATmega328 and allow the voltage regulator to power the speaker directly.

3.2 Wireless IO

The HM-11 functioned as required once set to the proper mode and role using the AT-commands described in the datasheet. An iPhone 7 was capable of connecting and communicating within 5 seconds of powering on the HM-11, and our distance requirement for an open space was verified in the ECEB lobby. The single wall range requirement was verified using the wall between the Senior Design lab and the TA room.

3.3 Local Inputs

3.3.1 Microphone

In the verification that the amplified output of our microphone amplification circuit stays between 1-3.3 V, we power the microphone circuit, and speak into the microphone at a distance within 50 cm while measuring the AC and DC components of the amplified signal. The DC component stays at 1.6 V, and the AC component is at most 1000 mV Peak-to-Peak, which results in a range of 1.1-2.1 V.

In the verification that the microphone must be able to accurately detect speech within a 2 foot radius, we stand within 2 feet from the mic and make audible sound, checking to see if it’s detected. The waveforms observed are meaningful and visibly different from waveforms of noise on the oscilloscope.

3.4 Local Outputs

3.4.1 LCD Display

For the requirement that the LCD display must be able to display the right message when prompted, we verified by setting the button in different modes and confirming the selected mode is chosen by displaying the names of those modes before prompting the next desired conditions on the LCD display. We tested each mode independently as well as the next desired conditions (label, due time, goal, etc.) and verified that the LCD functioned correctly upon each new insertion.

3.4.2 Color Display

The color backlight requirement for the LCD is that it must be able to be dimmed when not active, light up within 2 sec upon a button press, and change color within 2 seconds when transitioning between “good” and “bad” states. We visually confirmed that the color display functions as intended and then verified it occurred when intended.

3.4.3 Speaker

For the requirement that the speaker must be able to beep with a rising set of tones and a falling set of tones, we downloaded an app that shows the real time frequency of the current audio input and held the phone over the speaker. When the button is programmed to beep with a rising set of tones with each press, we verified with the app that the frequency is indeed rising, as well as with our own hearing perception. The same was done for the falling set of tones.

3.5 Control

3.5.1 Microcontroller

To verify that the microcontroller is able to correctly read button inputs and set proper outputs within 5ms, we connected one digital pin as input to a waveform generator, and another digital pin as output to an oscilloscope. The waveform generator is set at 50% duty cycle and 50 Hz, and is also connected to another channel on the oscilloscope. The two waveforms have the same rising edge and falling edge when the unit measurement is in 5 ms. Therefore, the microcontroller correctly reads digital inputs in way less than 5ms.

To verify that word recognition rate is > 60%, we programmed the ATmega chip to output a specific tone sequence to the speaker whenever the word “off” is recognized. Each of the three of teammates spoke to the microphone 10 times, and the times the speaker outputted the tone sequence was larger than 18.

3.5.2 RTC Chip

In the case of our RTC chip and its requirement, we initially planned to test the timer by simply programming the LCD to display the RTC’s time and comparing it to an internet-connected clock and determine the difference to be no more than a minute off.

However, upon testing we found our RTC to not function as intended, not incrementing properly or being able to keep time. We determined that the fault most probably lies in the oscillator crystal. With the temperature of the part after soldering, we determined that the crystal is most likely getting burned out in the process and wasn't able to function. Through repeated replacements and similar results we were unable to deduce a solution to the issue and ultimately were unable to include the module to our final product.

4 Cost Analysis

4.1 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 /hour x 3 (team members) x 2.5 x 8 (hours/member/week) x 13 (weeks)
= $23,400

4.2 Parts

Description	Manufacturer	Part#	Quantity	Cost
Push Button Switch	Corporate Computer	Corpco-x-6*6*6mmTacSw-10Pk	1	$0.529
Microphone	Challenge Electronics	CEM-C9745JAD462P2.54R	1	$ 5.95
Speaker	CUI	CDMG15008-03A	1	$ 2.58
Bluetooth HM11	Chip: Texas Instrument	SKU 317030001	1	$ 4.72
ATmega328p Chip	ATMEL	ATMEGA328P-PU	1	$ 5.95

Voltage Regulator	Pololu	U1V10F3	1	$ 4.49
AA Batteries	Energizer	N/A	4	$ 1.7
LCD Display with RBG Backlight	adafruit	398	1	$12.95
RTC Chip	Maxim	DS1339A	1	$ 1.60
Crystal Oscillator (for RTC)	MultiComp	56P2811	1	$ 0.72
Switch	ALCOSWITCH Switches	450-1643-ND	1	$ 0.16
Crystal (for ATmega)	TXC CORPORATION	887-2015-ND	1	$ 0.25
Battery Holder (2 AA)	MPD	BC12AAL-ND	1	$ 0.99
Battery Holder (20MM Coin)	Linx Technologies Inc.	BAT-HLD-001-ND	1	$ 0.28

Table 12: All Parts Needed for the Project

Parts Cost = $42.87

4.3 Grand Total

Grand total = Labor + Parts = 23,400 + 42.87 = $23,442.87

5 Conclusion

5.1 Accomplishments

Ultimately, we achieved almost full functionality on our button as per our requirements and goals. In addition to the individual modules working as intended, the logic of the process of selecting mode, goal/limit, due time, and action functioned as intended as well. The LCD, backlit LEDS, and speaker all responded as intended/dictated and the voice recognition worked at a rate exceeding our required expectations. Bluetooth functionality also worked as intended, properly reading and transmitting data to and from a preset terminal in accordance to the button modes and functions.

5.2 Uncertainties

The one requirement we were unable to meet was regarding the RTC (real time clock), which we were unable to get functioning as intended. We believe the main reason for this was our lack of understanding on how to handle the surface mount crystal oscillator. When we mounted the oscillator to our PCB, we found it to be exceedingly hot to the touch as a result of our soldering, even after multiple attempts with different oscillators. Additionally, while we were able to program the clock to the correct time, the timekeeping itself was subpar to nonexistent, which we believe verified our reasoning in why the oscillator burning out was probably the cause of the module's failure.

The lack of a working RTC module also made it impossible to periodically power down the bluetooth chip. It made the main loop unable to determine when the speaker should start outputting reminder sounds, and therefore unable to enable voice recognition, as the voice recognition should only be enabled when the speaker is in reminder mode.

5.3 Future Work/Alternatives

For future work on the button, we definitely would have to pursue the RTC issue further and attempt to have the module fully functioning and integrated into the rest of the circuit. We would also plan on fully implementing the functionality for our Scripted Action to achieve the IoT aspect of our button - currently, it completes the desired requirement of being able to transmit data to the user device that an action needs to be completed but, for our final product, we require the software to be able to execute the action as well. Final work we would like to finish on the product in the future would be further app development to pair with our button, specifically bettering user interface and including the ability the analyze the data acquired through transmission.

5.4 Ethical Considerations

The main points of ethical considerations we had to take in regards to our button were mostly related to how our button would be operated as opposed to the button design itself, such as users being dangerously reliant on the buttons to the point of its malfunction becoming deadly (ex. relying on solely the button to keep track of intake of medicine). 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" [9]. 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." [10]

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. Also in accordance to ACM Code of Ethics and Professional Conduct #1.2, we decided to use an existing Bluetooth model in place of creating our own as our experience in this particular area of expertise was lacking, thus avoiding potentially harmful effects of deciding to design one ourselves.

Citations

[1] Z. Lawrence and D. Peterson, "Mentally walking through doorways causes forgetting: The location updating effect and imagination," Nov. 20, 2014. [Online]. Available: http://www.tandfonline.com/. [Accessed: 10-Feb-2017].

[2] “Battery Myths vs Battery Facts”, 2017. [Online]. Available: http://www.greenbatteries.com/battery-myths-vs-battery-facts-1/ [Accessed: 20-Feb-2017].

[3] “Pololu 3.3V Step-Up Voltage Regulator U1V10F3”, 2017. [Online]. Available: https://www.pololu.com/product/2563. [Accessed: 24-Feb-2017].

[4] “Omni-Directional Foil Electret Condenser Microphone”, 2010. [Online]. Available: http://cdn.sparkfun.com/datasheets/Sensors/Sound/CEM-C9745JAD462P2.54R.pdf. [Accessed: 24-Feb-2017].

[5] “Human Hearing and Voice”, 2001. [Online]. Available: https://users.cs.cf.ac.uk/Dave.Marshall/Multimedia/node271.html. [Accessed: 24-Feb-2017].

[6] “Voice Level at Distance” [Online]. Available: http://www.engineeringtoolbox.com/voice-level-d_938.html [Accessed: 14-Mar-2017].

[7] “uSpeech” [Online]. Available: http://arjo129.github.io/uSpeech [Accessed: 30-Mar-2017].

[8] “Voice Recognition with Arduino” [Online]. Available: http://tiriboy.blogspot.com/2014/08/voice-recognition-with-arduino.html [Accessed: 30-Mar-2017].

[9] Ieee.org, "IEEE Code of Ethics", 2016. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 08-Feb-2017].

[10] acm.org, "ACM Code of Ethics and Professional Conduct", 1992. [Online]. Available:https://www.acm.org/about-acm/acm-code-of-ethics-and-professional -conduct. [Accessed: 08-Feb-2017].

Appendix A: Requirement and Verification Table

Requirement	*Verification*
Output voltage from the ***voltage regulator*** *must be 3.3 0.3 V, while providing a current ranging from the lowest possible (2 mA) to highest possible (233 mA) current draw of our design. [4 pts]*	1. *Set the input to the voltage regulator to be 3.0V (2 AAs in series) using a DC power supply* 2. *Test output voltage across both 1650 Ω (simulating lowest power scenario) and 15 Ω (simulating highest power scenario) resistor load.* 3. *Ensure that the voltage is within the range of 3.0 - 3.6 V.*
Output voltage from the ***voltage regulator*** *must be 3.3 0.3 V given a input voltage ranging from 3.0V to 1.5 V (to simulate the voltage drop of 2 AA batteries in series) [4 pts]*	1. *Repeating the steps above, slowly dial the DC power's output voltage from 3.0 down to 1.5 and verify that the output voltage of the voltage regulator remains within the range of 3.0 - 3.6 V.*
The ***Bluetooth*** *chip must be able to detect if the user device is within 15 meters in open space, and within 8 meters if there is one dry wall (TA room to main lab) in between the button and the user device. [4 pts]*	1. *Hold the user device in one hand and measure the distance between the person and the button, make sure it is 15 meters in open space.* 2. *Pair user device (phone, laptop, or tablet) to the button, verify connection in the user device's built in settings* 3. *Hold the user device in one hand and stand 8 meters away from the button, placing 1 non-concrete wall between the device and the button .*

	4. Pair user device (phone, laptop, or tablet) to the button, verify connection in the user device's built in settings
*The **Bluetooth** chip must be able to detect a user's presence within 5 seconds. [3 pts]*	*1. Set the HM-11 to active and pair it with a host device.* *2. Put the MH-11 to deep sleep and prepare a stopwatch.* *3. Set the HM-11 back to active and time how long it takes to reconnect.* *AND / OR:* *1. Set HM-11 to active and pair it with a device* *2. Turn off the bluetooth on the other device* *3. Turn on Bluetooth on the other device and time with a stopwatch* *Requirement is met if either of these tests succeed.*
*The **microphone**'s maximum output voltage is amplified to 1 to-3.3 Volts. [3 pts]*	*1. Set microphone to receive audio input.* *2. Using a computer or tone generator, set frequency sweep tone from 500 to 2000 Hz, at 70 dB very close to the microphone (within 50 cm)* *3. Use an oscilloscope to measure the voltage after the amplifier, to make sure that the audible part of the signal has peaks in the 1-3.3 Volt range and no peaks are higher than 3.3 Volts.*
*The **microphone** must be able to accurately detect speech within a 2 foot radius. [3 pts]*	*1. Set microphone to receive audio input.* *2. Stand at or within 2 feet from the mic and make audible sound, checking to see if it's detected.* *3. Directly read the voltage across the amplifier, make sure voltage is between 1-3.3V.*
*The **LCD** display must be able to display the right message when prompted. [5 pts]*	*1. Implement simple program to output desired phrase (ex. Hello World!)* *2. Observe display for desired phrase.* *3. Test three times with three modes.*

	4. *Observe display: should display task on button press.*
***LCD** must be able to be dimmed when not active, light up within 2 sec upon a button press, and change color within 2 sec when transitioning between “good” and “bad” states [5 pts]*	1. *Initialize Button to a counter that is under it’s goal/limit* 2. *Press button to increment counter past goal/limit. Ensure that LCD brightens and dims within 2 sec, and that the color changes upon reaching/passing the limit.*
*The **speaker** must be able to beep with a rising set of tones and a falling set of tones [3 pts]*	1. *Program speaker to beep at designated tones.* 2. *Hold phone directly above (within 1cm) the speaker and use app to measure frequency component.* 3. *Verify frequency is rising/falling.* 4. *Repeat steps 1-3 both for “Encouraging” and “Discouraging” frequency ranges.*
*The **speaker** must be able to beep with intensity > 60 dB and < 85 dB when operating as a part of our circuit (powered by 3.3 V ATmega328 output pin, 1 Ω resistor in series). [3 pts]*	1. *Output any tone to speaker.* 2. *Hold phone directly above (within 1cm) the speaker and use app to measure decibel level of speaker sound.* 3. *Verify that decibel level is between 60 dB and 85 dB.*
***Microcontroller** must be able to correctly read button inputs and set proper outputs within 5ms [5 pts]*	1. *Connect the button input of the ATmega328 to a square wave generator set to output 0-3.3V, 50 Hz, at a 50% duty cycle. This simulates the button being pressed for 5ms then released for 5ms.* 2. *Program the ATmega328 to set the output of one of its pins to match the button input.* 3. *Connect oscilloscope probes to both the input and output pins* 4. *Turn on the waveform generator.* 5. *Observe the two waveforms. Ensure that the delay is less than 5ms (output pin is set to high before the input goes low).*

Word recognition rate must be > 60%. [5 pts]	1. *Each of the three teammates will speak the word “OFF” into the microphone, at distance 0 ft three times, 1 ft three times, and 2 ft four times.* 2. *Must correctly recognize command and stop at least 18 of the 30 utterances.* 3. *Repeat the process, speaking words other than “OFF”. Ensure that there are no more than 12 false positives out of 30.*
*The **RTC** must be able to correctly recognize time set by device within 1 minute. [3 pts]*	1. *Display the time read from the RTC onto the LCD.* 2. *Keep an eye on an internet connected clock (phone, computer). Ensure that the time displayed is no more than one minute off.*

Appendix B: Circuit Schematics and Corrections

Below is the circuit schematic used to generate the PCB layout we used to manufacture our board. Corrections made to the board after receiving the PCB are listed underneath followed by a final schematic representing our final design (including changes made with jumper wires and cut traces).

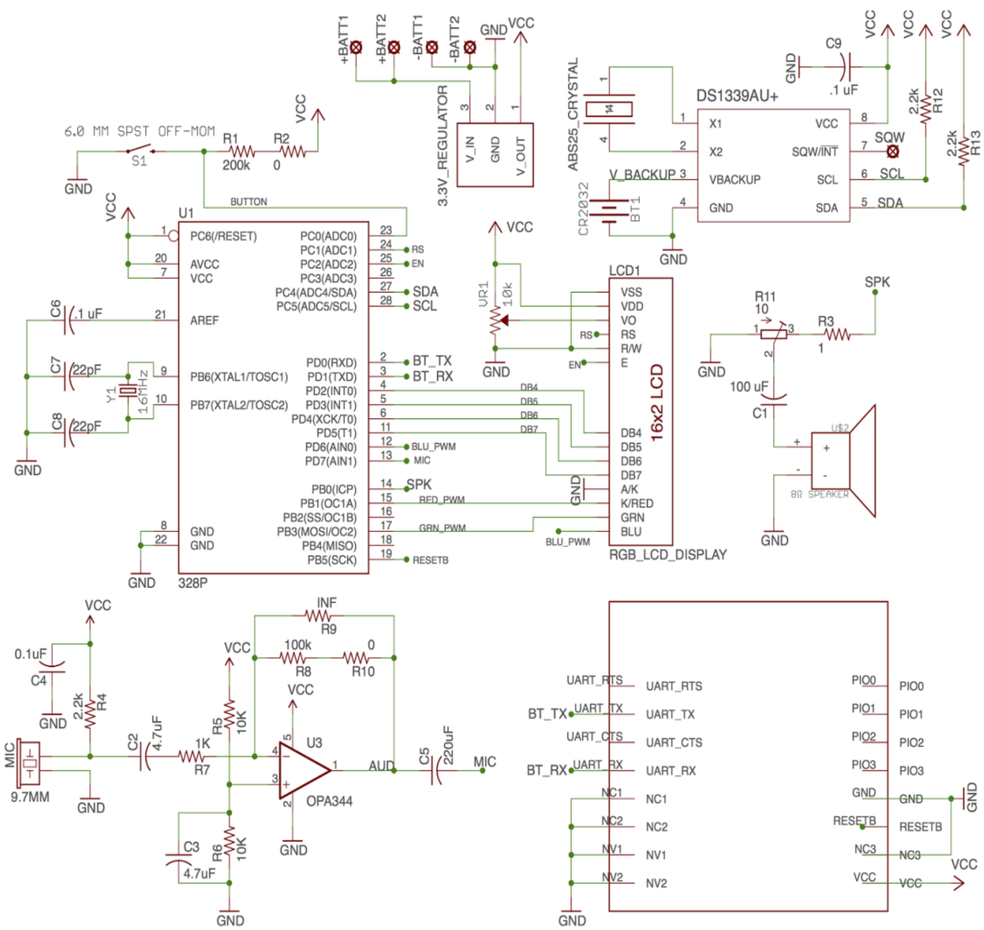

Figure 5: Circuit Schematic Used to Create PCB

Corrections made to above design in the order they were made:

- LCD Display: A/K should be connected to GND, not VCC. LED pins are pulled to ground to light up rather than the other way around. *4/13/17*
- ATmega328: AIN is a comparator, not an analog input. Connect MIC signal to ADC3 instead. *4/13/17*
- C1,C5: These capacitors do not come in a 1206 package and can be excluded. *4/14/17*
- LCD Display: LCD breakout board was designed for 5V despite the LEDs and LCD display themselves being able to run on 3.3V. The contrast circuit requires an input voltage ≈4.5 less than VCC, so we solved this issue by adding a 1.5V coin cell with the positive terminal connected to GND and the negative to what was formerly the VCC end of the trimpot in order to supply the negative voltage necessary to display any text. *4/22/17*
- HM-11: NC3 is internally connected to VCC, so it should not be grounded on the PCB. Must leave unconnected or connect ot VCC. *4/23/17*
- R8, R9, R10: Increase gain of op amp by replacing all of these resistors with 200k chips to change R_{eq} from 100kΩ to 133kΩ. ≈*4/23/17*
- HM-11/ATmega328: Bluetooth presence can be detected through LED pin of HM-11. Connect this to an input of the ATmega328. *4/24/17*

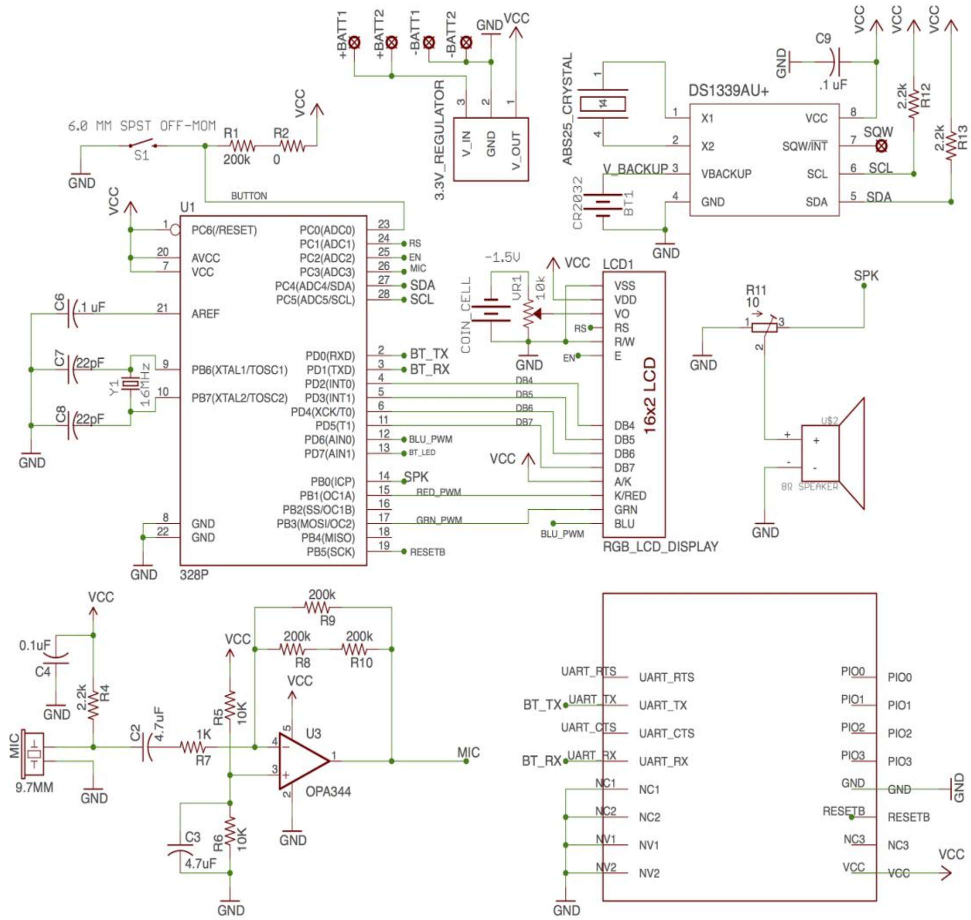

Figure 6: Final Circuit Schematic with Corrections to PCB

Appendix C: PCB Layout

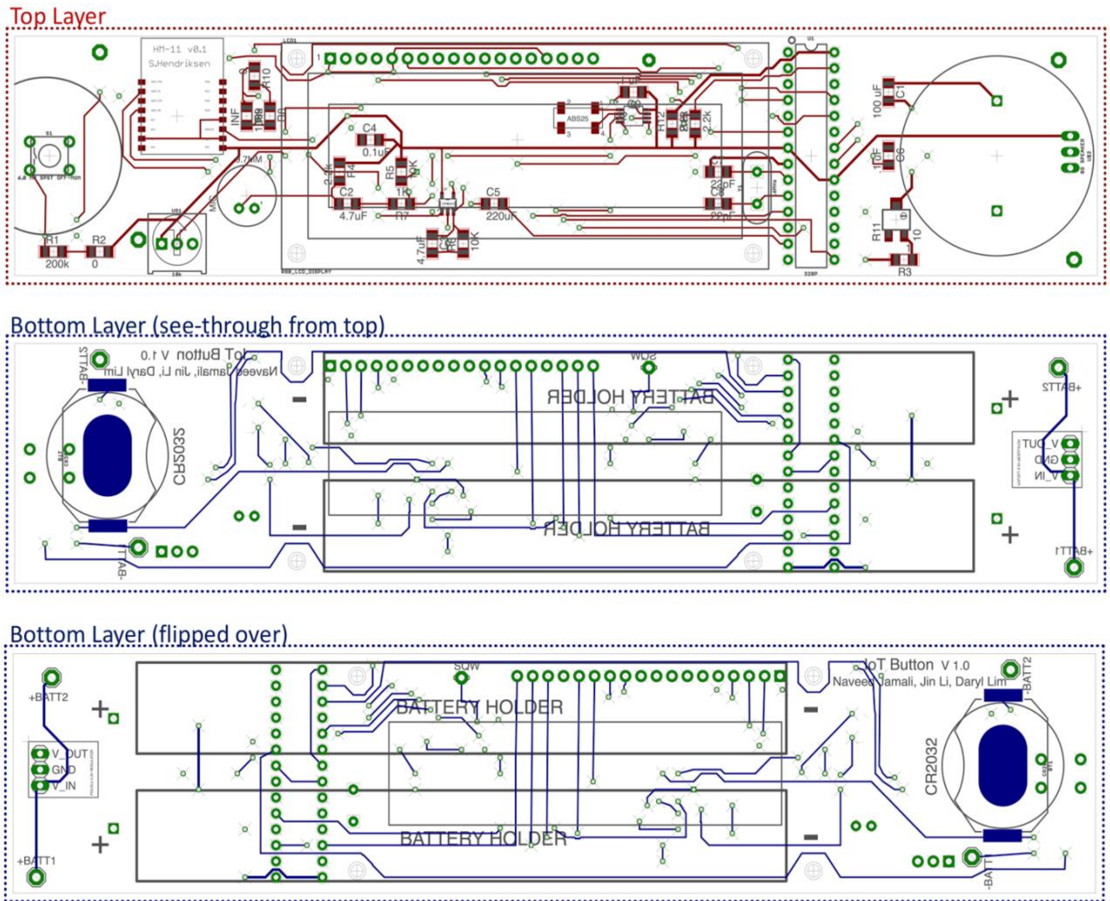

Figure 7: Top and Bottom PCB layouts

NOTE: Adjustments were made to this board by replacing components, cutting traces, and adding jumper wires in our final design. See Appendix B for the schematic used for this board, a full list of corrections, and the final schematic.

Appendix D: Physical Layout

Front:

Figure 8: Physical Layout of Device (Front)

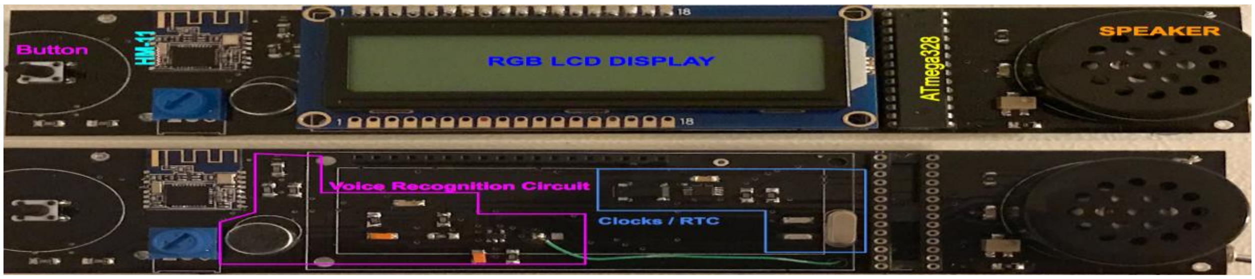

Figure 9: Physical Layout of Device (Front with LCD removed)

Back:

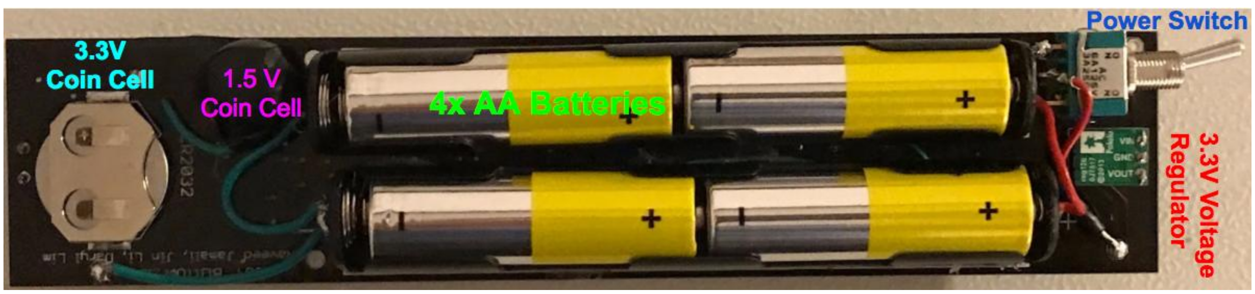

Figure 10: Physical Layout of Device (Back)

Appendix E: Bluetooth Calculations

We want to ensure that if a user were to walk to the button from outside its range, press it, then immediately walk back out of range that the button will have enough time to react and sync data at least once while both approaching and leaving the button. Given the worst case range of 8 meters and the average human walking pace of 1.4 m/s, this implies that the **Bluetooth chip needs to be activated at least once every 5.7 seconds** to meet this requirement.

As for how long the chip needs to be active, the HM-11 advertises a data transfer rate of 115,200 bits/sec. Once data transfer cycle would consist of a send and receive portion, which be estimated as follows:

Sent to button:

Program Enable (1 bit)	Mode (2 bits)	Label (192 bits)	Mode-Specific Data (55 bits)

Program Enable:	True when programming button. Rest of bits are don't cares when false
Mode:	00 - Set Clock 01- Counter 10 - Checklist 11 - Action
Label:	Displayed on LCD, 24 chars x 8 bits per char = 192 bits
Mode - Specific Data:	Worst case is for the checklist. There are 1,440 mins in a day, which requires 11 bits to represent in binary. This means each alarm requires 11 bits, and if we allow up to 5 alarms a day we need 55 bits

Total: 250 Bits

Sent by Button:

The data sent by the button is entirely dependant on the mode it is set to. For the counter, the button will probably not be pressed more than 128 times (probably far less) a day, so 8 bits the hold the count should suffice. The checklist only needs to hold a boolean value for every alarm to represent whether or not the task has been completed. With 5 daily alarms, 5 bits are needed. For the scripted action, we may allow 3 different actions to be carried out depending on how the button is presses. For example, if the button is used to control lights, pressing it briefly can turn the lights on/off, pressing and holding it can dim them, and double pressing it in quick succession can reset the brightness to maximum. We would need 2 bits to represent these 3 different types of button presses. Therefore, the worst case for the number of bits to be sent by the button is for the counter at 8 bits.

Putting these two components together, that means that the minimum amount of data that needs to be sent/received per data transfer cycle is 258 bits. In order to allow large tolerance for error and to account for extra parity bits and bits required to control the Bluetooth protocol, let’s call this 512 bits.

The HM-11 claims to have a data transfer rate of 115,200 bits/sec. With the 512 bits we wish to be able to transfer per cycle, that means that the **chip must be on for 4.44 ms to complete one data transfer cycle.**

Combining our constraints for how often the chip must be be activated with how long it needs to be on, and adding even more of a tolerance to each constraint, it should suffice to **have the chip be set active for 5 ms out of every 5 sec**, and set to deep sleep for the remaining time.

The average current draw with this specification is calculated using the following table and equation:

Mode	Current (mA)

Transmit	15
Receive	8.5
Deep Sleep	0.6

Table 2: Current Drawn in Different Modes (datasheet)

$$i_{avg} = \frac{((.005\ sec)\ ((\frac{250\ bits}{258\ bits})(8.5\ mA_{receive}) + (\frac{8\ bits}{258\ bits})(15\ mA_{transmit}))_{active} + (4.995\ sec)(0.6\ mA_{deep\ sleep})}{5\ sec}$$

Equation 1: Average Current Draw of HM-11

Solving the equation above, we get an **average current draw of 0.608 mA** for the HM-11.

Appendix F: High-Level Software Overview

The following code is a high-level abstraction of the software we wrote for our microprocessor. Internal function calls, preprocessor directives, and many other details have been excluded for clarity.

```
void setup(){

  LCDsetup();
  BTSetup();
  SpeechSetup();

  switch(mode)
  {
    case CHECKLIST: ChecklistSetup();
                    break;

    case COUNTER:   CounterSetup();
                    break;

    case ACTION:    ActionSetup();
                    break;
  }


}

void loop() {

  LCDUpdate();

  switch(mode)
  {
    case CHECKLIST: ChecklistUpdate();
                    break;


    case COUNTER: CounterUpdate();
                    break;

    case ACTION: ActionUpdate();
                    break;
  }

  BTLoop();

  SpeechLoop();
}
```