AUTOMATIC HANDSHAKE CONTACT INFO EXCHANGER

By

AMBIECA SAHA

WILLIAM HANLEY

KUANYSH SAMIGOLLAYEV

Final Report for ECE 445, Senior Design, Spring 2013

TA: Justine Fortier

01 May 2013

Project No. 13

Abstract

We designed and created a wireless contact information exchanger that would do so upon a handshake. The device (henceforth, called '*bracelet*') has a user interface, comprising of two switches (*Receive* and *Send*) and two LEDs to warn the user against low battery and low memory).

This device uses an accelerometer to detect a handshake, a Bluetooth module to send and/or receive business cards, a microcontroller to coordinate this sequence of actions, an external memory to store the received business cards. However, to make the bracelets send data without errors, one had to be set as *Master* and the other *Slave*.

Received business cards can be uploaded to a personal computer or even to a phone (in case the second person involved in handshake does not own a bracelet).

Although our design worked, there are still improvements that can be made by eliminating the master/slave switch, developing a better user interface (that would allow user to automatically convert uploaded received business cards into a contact file), adding a contact picture in the business card and stepping down power consumption when not in use.

Contents

1. Introduction

1.1 Purpose

The goal of this project was to produce a wearable and affordable device that would be capable of transmitting and receiving contact information from other users.

We wanted to use our knowledge of communications and systems design to engineer a device that would improve the way people exchange contact information with each other –

- Instead of taking time to exchange information and wasting paper with business cards, one could automatically receive and send contact information electronically and wirelessly with the shake of a hand.
- This device would eliminate the need to print new business cards whenever information is updated.
- It would allow easy organization and access of contact information (eliminates the need to carry around huge stack of business cards; reduces risk of accidentally losing someone's business card).

This project would be useful for a lot of people in the business world and might be able to revolutionize the process of business card exchange and personal information exchange in general.

1.2 Functionality

The function of our device is to let users exchange business card information after a handshake has occurred. The device automatically transfers contact information wirelessly to the other user. This information can also be sent to a Smartphone. After coming home from a business meeting, conference, etc. the user can attach his/her device to a computer and upload all the contact data he/she received. Once the data is uploaded to the computer, the memory can be cleared and the main code can be reprogrammed to the bracelet.

1.3 Design Subdivisions

The project was broken into many modules, which each perform specific tasks, explained in detail under Section 3. The modules we decided to implement are shown in Figure 1.

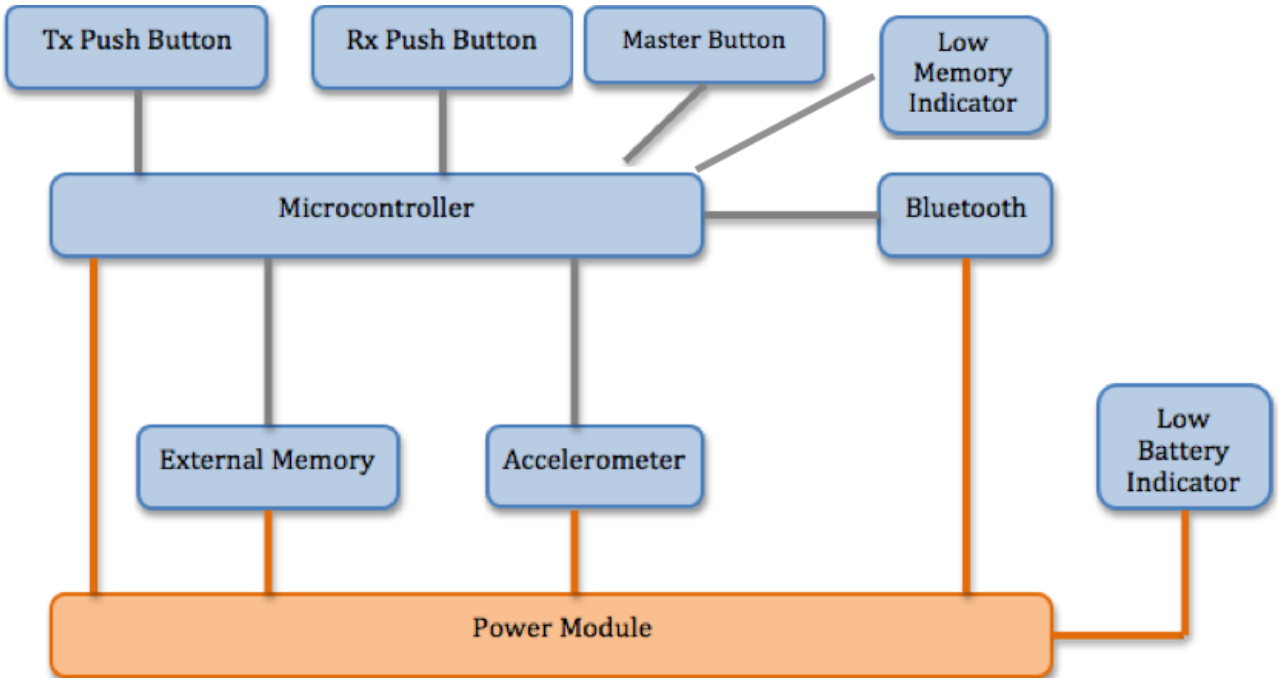


Figure 1: Top Level Block Diagram for Device

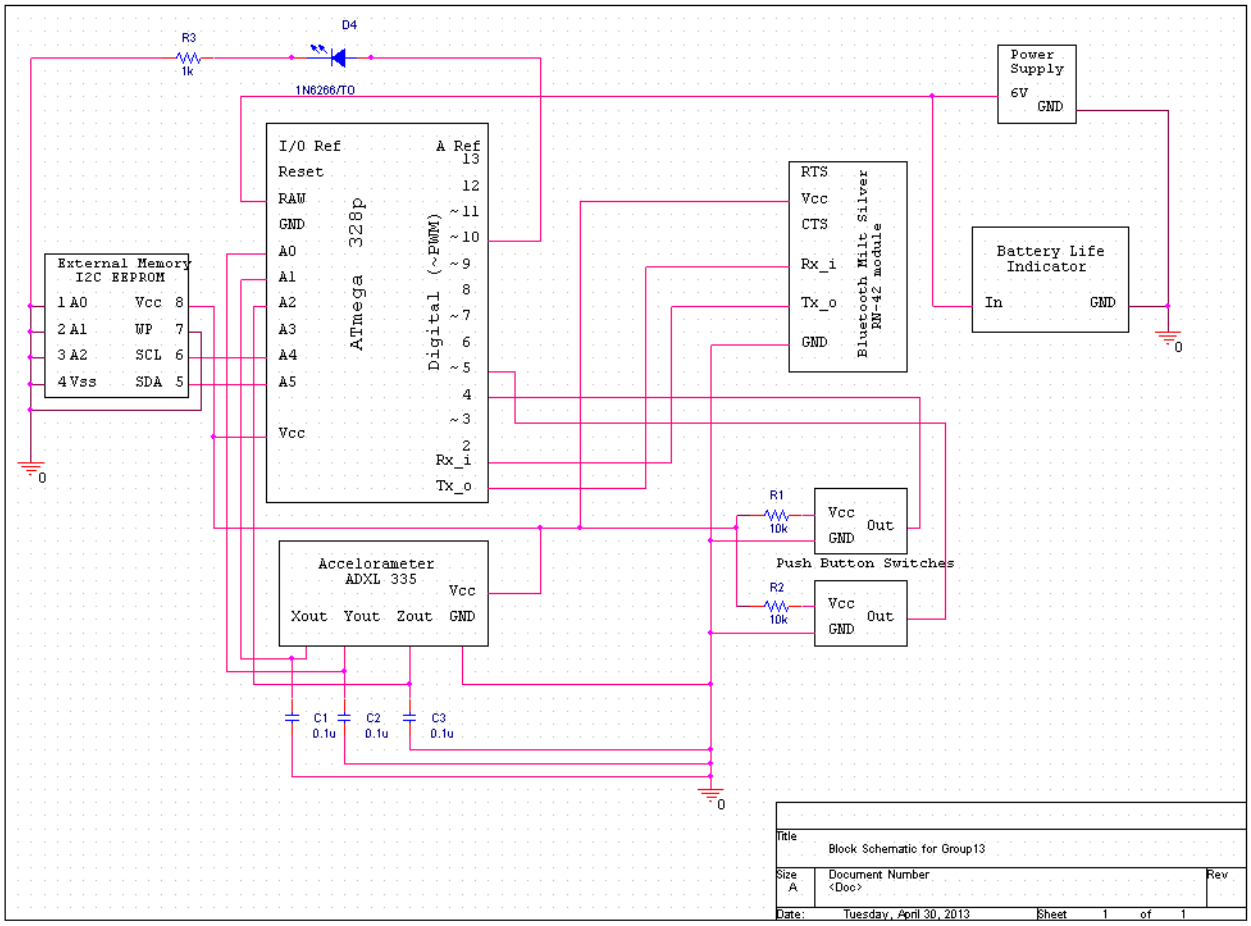


Figure 2: Overall Circuit Schematic for Device

2. Design

2.1 Design Alternatives

One of the biggest problems we faced while working on this project was to coordinate the two Bluetooth modules to transfer data error-free. Upon a handshake, if both Bluetooth modules tried to connect to each other at the same time, the connection attempt would fail and no data would be transferred. So to tackle this problem, in the interest of time, we came up with a crude engineering solution: prior to a handshake, one device would be designated as the *Master* and the other as *Slave*. Upon a handshake, the *Slave* would turn its discoverability on. The *Master* would then be able to connect to the *slave,* initiate the communication and at the end of data transfer, the *Master* would also kill the connection.

However, as mentioned earlier, this was only a crude engineering solution that would allow us to produce a working device. After the demo, we were able to come up with a better solution that would eliminate the need to coordinate who is the *Master* and who is the *Slave* with the other user before every handshake. This can be achieved by sending the Bluetooth modules a command *(GK)* that finds out if they are currently connected or not (elaborated in Section 5.4).

2.2 PCB Design

To maximize the robustness of our design, we soldered all components onto the PCB (Printed Circuit Board). To ensure that our device fit on a user's wrist, we made two small PCBs. One PCB, which contains the power supply and user interface (switches and warning LEDs), is located on the lateral side of the user's wrist (figure 3). The other one is located on the medial side of the user's wrist (figure 4). These two PCBs are connected by 7 wires that wrap around the user's wrist. These wires connect the outputs of the user interface and power supply to the rest of the circuit components. To minimize the size of PCBs, we soldered any low-height components in the area under the accelerometer breakout board and Bluetooth breakout board (figures 5 & 6). The lateral-side PCB with the power supply is 2.23"x1.7'' with 32 mil traces. The dimensions of the medial-side PCB are 2.23"x1.61'' with 16 and 32 mil traces. These trace widths are rated at approximately 1A [1]. The largest current drawn at anytime for a circuit happens when the

Bluetooth is connected and transmitting data and will not exceed 60 mA. The PCB schematics and layouts are located in Appendix B.

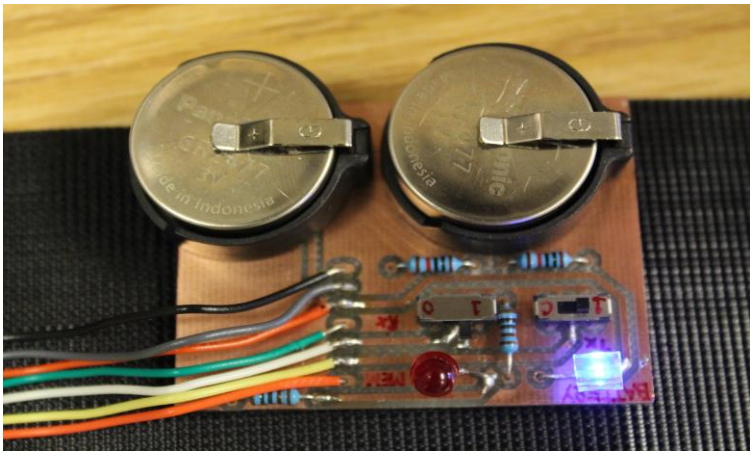


Figure 3: Lateral-side PCB Showing User Interface and Power Supply

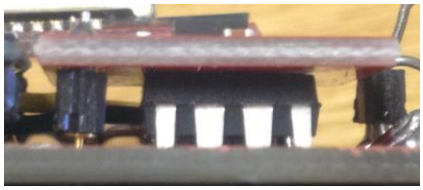


Figure 4: External Memory Under Accelerometer Breakout Board

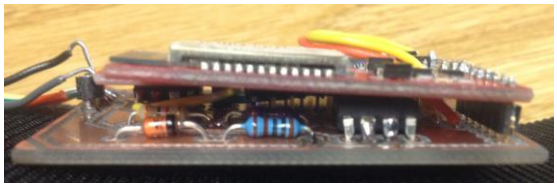


Figure 5: Memory Indicator Circuit Under Bluetooth Breakout Board

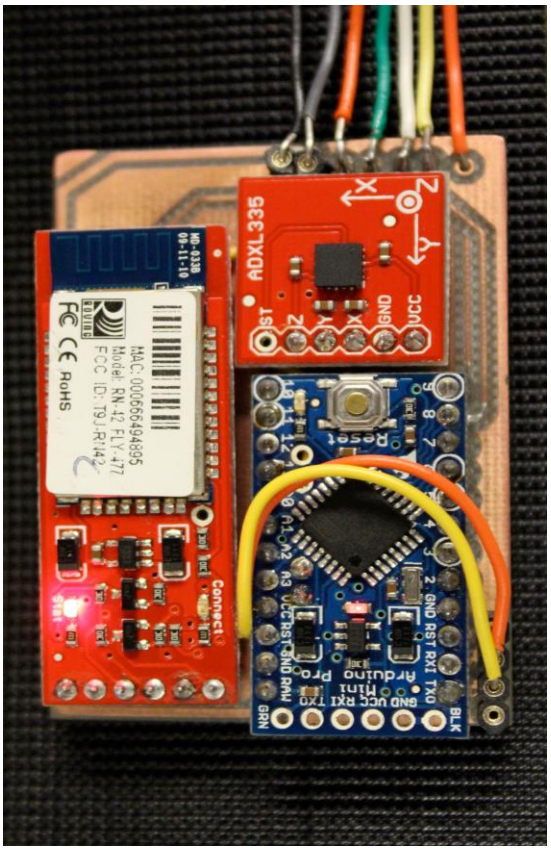


Figure 6: Medial-side PCB

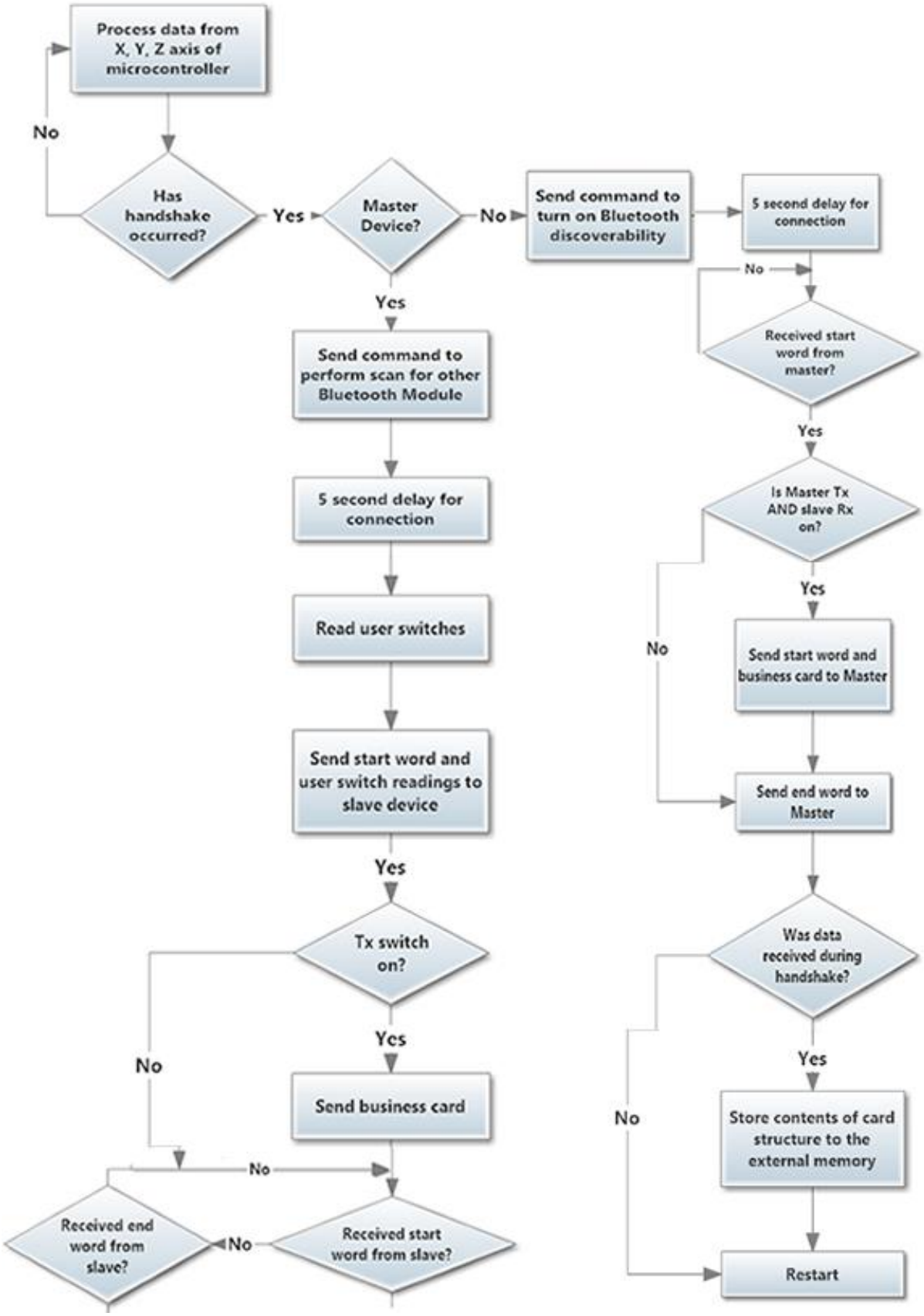
2.3 Design Details

2.3.1. Microcontroller

The microcontroller controls all the components in the circuit. The microcontroller interacts with every component in our device except the low battery indicator circuit. We chose to use an Arduino microcontroller because of its large open source software environment and if needed, we could easily find help with common problems on many online forums. In order to perform initial tests on our components, we used a larger Arduino Uno platform, which uses an ATmega328P microcontroller [2]. After initial testing and verification of the components we moved on to using the 3.3V Arduino Pro Mini microcontroller. The Pro Mini was functionally equivalent to the Arduino Uno except that it has an 8 MHz clock instead of the Uno's 16 MHz [3]. Further, it runs on 3.3V instead of the Uno's 5V power requirement. The Arduino Pro Mini has a RAW input pin that accepts any unregulated voltage between 3.5-12V and regulates it on the board to 3.3V. The biggest advantage of using the Arduino Pro Mini instead of the Arduino Uno was its small size and weight. The dimensions of the Arduino Pro Mini are 0.7"x1.3'' and it weighs less than 2 grams [4]. This makes it ideal to be used for small and lightweight projects such as ours.

However, when moving over from the Uno to the Mini, we encountered the problem that the 3.3V Pro Mini did not support the Software Serial Communication pins that we were using on digital pins 5 and 6 of the Uno microcontroller. Therefore, we had to make changes in both our hardware and software design. The Pro Mini was only able to perform serial communications out of its hardware serial pins 1 and 2. This meant that the device would have to share these pins between the connections to the Bluetooth Module transmitter/receiver and the serial connections necessary to read the contents of the memory onto the computer. Therefore, we had to break-out headers on the medial-side PCB which would allow switching connections to these serial communication pins of the Mini when necessary.

Figure 7 shows a detailed flow chart of the microcontroller program function.


Process data from X, Y, Z axis of microcontroller
No
Has handshake occurred?
Yes
Master Device?
No
Send command to turn on Bluetooth discoverability
5 second delay for connection
No
Received start word from master?
Yes
Is Master Tx AND slave Rx on?
Yes
No
Send start word and business card to Master
Send end word to Master
Was data received during handshake?
Yes
No
Store contents of card structure to the external memory
Restart
Yes
Send command to perform scan for other Bluetooth Module
5 second delay for connection
Read user switches
Send start word and user switch readings to slave device
Yes
Tx switch on?
Yes
No
Send business card
No
Received start word from slave?
No
Received end word from slave?

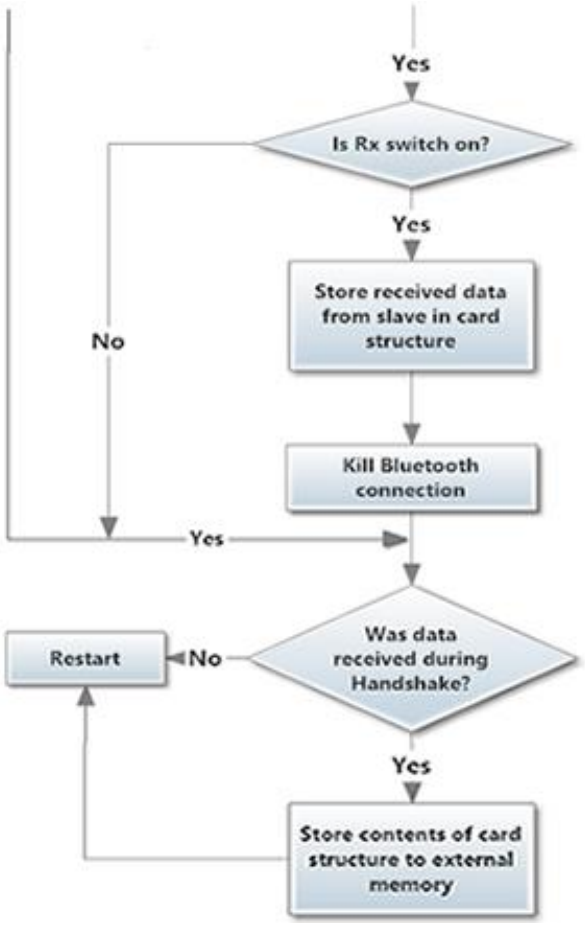


Figure 7: Microcontroller Flow Chart

2.3.2. Bluetooth Module

The Bluetooth module on our device is used for wireless data transfer upon a handshake. Our device uses an RN-42 Bluetooth module on a breakout board from Sparkfun Electronics. We configured the Bluetooth modules (as well as every other component in our circuit) to transfer data with a 9600 *baud rate*. Equation 2.1 below shows how long a business card of 200 bytes will take to transfer at this *baud rate*.

$$9{,}600\ \frac{bits}{second} \Rightarrow \frac{1}{9{,}600}\frac{seconds}{bit} \sim 104.2\ \frac{\mu s}{bit} \qquad (2.1)$$

$$1\ buisness\ card = 200\ bytes * 8\ \frac{bits}{byte} = 1600\ bits$$

$$104.2\ \frac{\mu s}{bit} * 1600\ bits = 166.7\ ms$$

When the user shakes hands, if the device is the *Master*, the microcontroller will send serial data "$$$" to the Bluetooth module. The *Slave* device will make itself temporarily discoverable. When the Bluetooth module reads the "$$$" on its Rx (receiver) pin the module will enter CMD (command) mode. Once in command mode the microcontroller can signal the Bluetooth to perform an inquiry scan to connect the other nearby Bluetooth RN-42 module that is temporarily discoverable. Once it connects to the other device, both modules will automatically enter data mode. In this mode, serial data is transferred between the two modules at their defined *baud rate*. In data mode, the data transfer happens according to the flow chart in the microcontroller section (figure 7). The complete list of commands that were used to set up communication between the modules is listed in the RN-42 User Manual [5] and can be viewed in the microcontroller code in Appendix C.

2.3.3. Accelerometer

The accelerometer on our device outputs analog voltages to the microcontroller for sampling and processing. We chose to use the ADXL335 triple axis accelerometer. We picked this chip because it had low cost and measured acceleration on the three Cartesian axes that were necessary to detect a handshake. The hardware design for this chip solely consisted of attaching the three axis outputs to the analog inputs on the Arduino board. This section explains the software design that was used for handshake detection.

Acceleration from both motion and gravity was used in this process. The data received from the accelerometer was processed by the microcontroller. Our device was designed to detect two different types of handshakes:

a. A handshake starting with the user's initial arm position being by his/her side pointing to the ground
b. A handshake that is continued from a previous handshake, that is the user's initial arm position is already horizontal (finishing up a previous handshake)

Before the microcontroller could process the acceleration data it had to go into the Analog-to-digital converter of the Arduino. This resulted in the analog voltages being sampled and mapped to a value between 0 and 1024. In our design, the accelerometer's is aligned such that the x-axis is pointing towards the user's shoulder and the y-axis points in the vertical direction.

Figure 8 shows a sample output (corresponding to 'a' above) after accelerometer data has been sampled by the Arduino's analog to digital converter. The left column represents the acceleration on the X-axis, middle column the Y-axis, and the right column represents the Z-axis. One can see that the X-axis has a large change in output value. When a large change in the X-output is detected we went on to check for Y-oscillations around its mean value of around 610 when the arm is perpendicular to the body. Only when these two events happen in close succession, the microcontroller signals that a handshake has occurred.

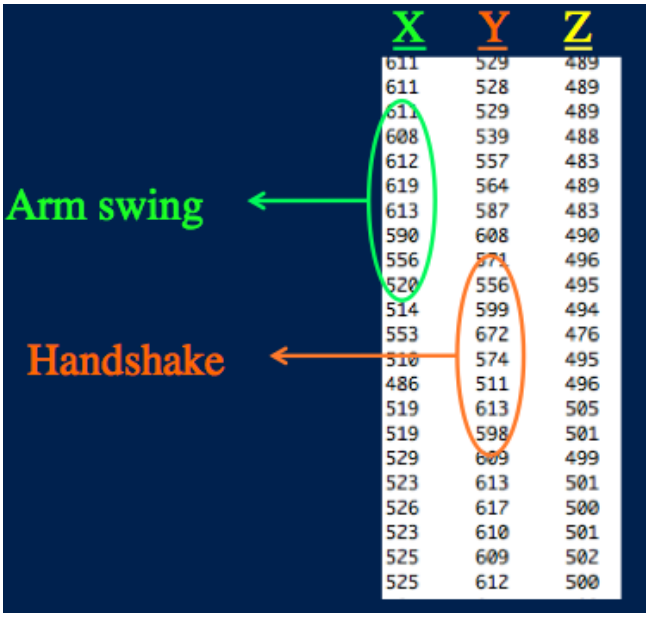


Figure 8: Sample Data From Accelerometer Detecting An Arm Swing And A Consecutive Handshake

A similar design method is used for the horizontal handshake for the bullet point 'b' mentioned above. The only difference is that we are looking for large changes in acceleration on the Z-axis corresponding to the swing, instead of the X-axis.

2.3.4. External Memory

We used the 24LC256 EEPROM for storing received business card data. This was necessary to incorporate into our project because the microcontroller on-board memory was too small to meet our storage needs. The external memory of our circuit was designed to operate with the analog input pins A4 and A5 on the Arduino. These two analog input pins support I^2C communication [6]. Pin A4 is used as the SDA (serial data) line and pin A5 is the SCL (serial clock) line. The SDA line is a bidirectional connection used to transfer addresses and data between the memory and Arduino. The SCL line is used to synchronize data transfer between the two components. The design processes for writing to and reading from this memory is explained below. The Arduino "Wire" library is used to accomplish these processes.

Writing to the memory:

After we receive a business card upon data transfer, we want to store this card into memory. In order to do this the external memory requires a start bit and control byte to be sent to it through the SDA line. The first 4 bits of the control byte are a set control word "1010". The next three bits are device-addressing bits, which are all grounded because we are only using one memory chip. The last bit in the control byte is Read/Write' (R/W'). This bit will be a "0" because we are writing to the chip. Immediately after the control byte is sent we send two bytes representing the address to write to and finally send the byte of data to write and the stop bit. This process is summarized below in Figure 9, taken from the 24LC256 datasheet [7].

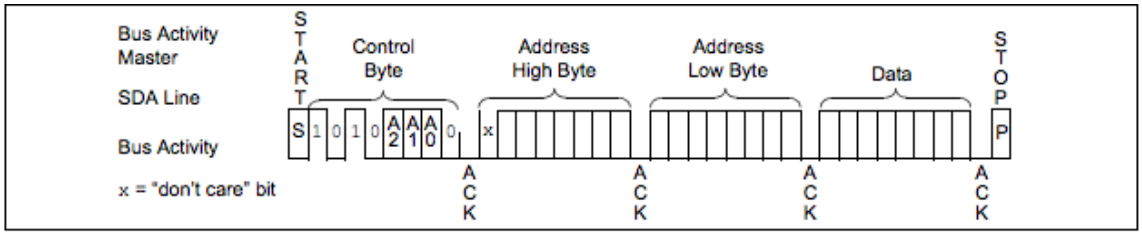


Figure 9: Writing to External Memory

We decided to store the number of business cards currently in the memory in memory address ‘0’. Every time we write another card to memory this number gets incremented and stored back in address ‘0’. This number allows use to know what address we are writing the current business card data to. Equation (2.2) below summarizes what memory address we would write the current card to. The plus one offset is because we are storing the number of business cards in the first memory location so every address needs to be incremented by 1.

$$(Number\ of\ cards) * (size\ of\ card\ in\ bytes)) + 1 = (starting\ address) \qquad (2.2)$$

So this business card will be written in addresses:

(Starting address) to (starting address + 200)

Reading from the memory:

When the user uploads the program to read the stored business cards to their computer we needed to extract every business card that was currently in the memory. The 24LC256 read protocol is pretty similar to writing to it. Except now you don’t need to pass a data byte after the two address bytes. The process is shown in Figure 7, the figure is taken from the 24LC256 datasheet [7].

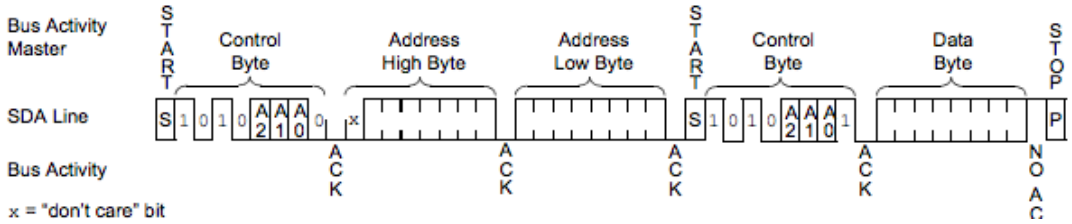


Figure 10: Reading from the External Memory

The read function that we wrote returns the byte of data stored in the specified memory location. We also designed the external memory so that one can clear the memory after reading its contents.

2.3.5. Low Battery Indicator

We decided to design a low battery indicator that would warn the user when their batteries had approximately 15% charge remaining. This would give them sufficient amount of time to replace the device batteries. After referencing the data sheet [8] we decided that when the voltage over each battery hit approximately 2.4V there would be 15% charge remaining. Since we were using two button cell batteries in series we would need to detect when the output of the power supply decreased to under 4.8V (equation 2.3).

$$V_T = 2.4V * 2 = 4.8V \qquad (2.3)$$

In order to detect this threshold, we decided to use an op amp circuit as an analog comparator. The op amp compares the two voltages on its positive and negative input terminals. When being used as a comparator, an op amp operates in open loop and non-linearly. The basic concept is when the non-inverting input (+ve terminal) is at a higher voltage than the inverting input (-ve terminal) then the output saturates to its highest possible output voltage. When the opposite was true and the inverting input is at a higher potential than the non-inverting input then the output saturates to the lowest voltage it can output. Figure 11 below shows the schematic of our design for this component.

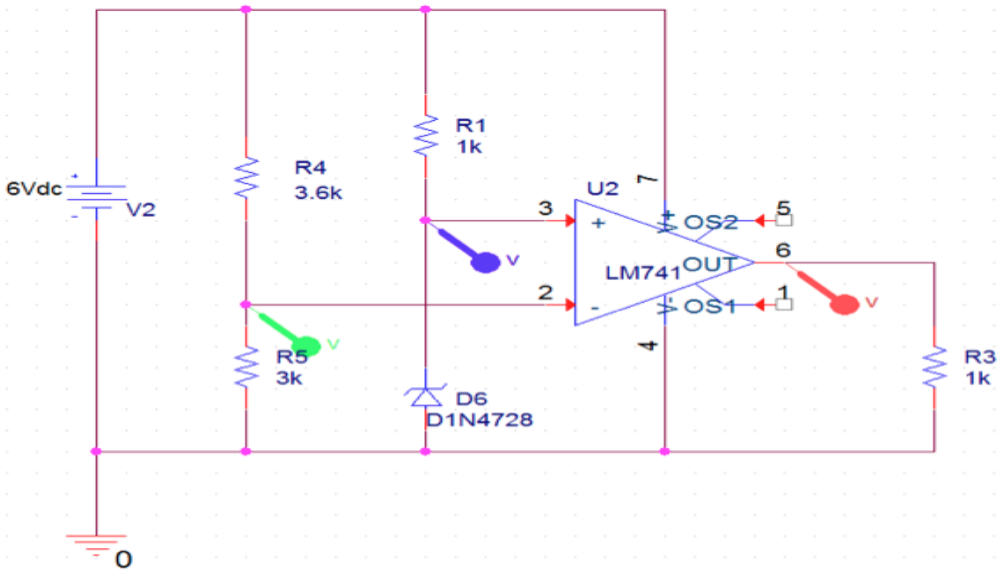


Figure 11: Schematic for Low Battery Indicator

So when the voltage divided battery falls below the voltage at the non-inverting node (set by the Zener diode), the output will become high and light up the blue LED at the output warning the user that the battery life is low.

2.3.6. Low Memory Indicator

The low memory indicator was included in our design to warn the user when the remaining space of the external memory was running low. This component is a red LED that lights up when the memory is running low. In order to do this we attached a red LED to a digital output pin of the Arduino Pro Mini along with a 1kΩ current limiting resistor. The setup of this circuit is pictured below in Figure 12 as a simple resistor in series with an LED going to ground [8].

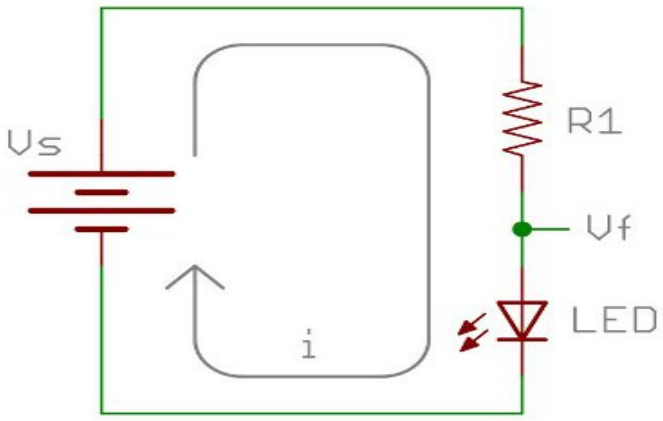


Figure 12: Circuit Schematic for Low Memory Indicator

A KVL loop around this circuit gives:

$$R = \frac{V_s - V_f}{i} \tag{2.4}$$

Where

V_s = Arduino Pro Mini output voltage – 3.3V

V_f = approximate forward voltage for red LED ~ 1.85V [9]

i = current required to light LED to desired brightness

We want ~1.15mA of current to light our LED so we obtain R = 1kΩ.

2.3.7. Power Supply

The design of the power supply is simply two CR2477 button cell batteries in series (figure 13). These are 3V batteries rated at 1000mAh [10], so the output of the power supply will be 6V and 1000mAh. In order to attach these button cells to the PCB we needed to order sockets (BH1000G) for these button cells. The terminals of these sockets were soldered to the PCBs. An advantage of using these sockets is that the button cells can be easily replaced when they die.

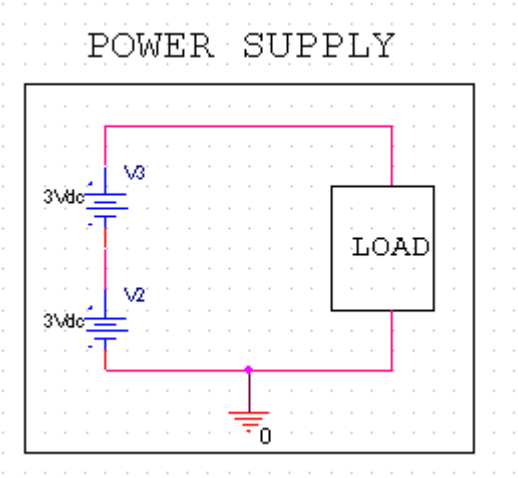


Figure 13: Circuit Schematic for Power Supply Module

2.3.8. Manual Switches

In order for the user to select what mode they want data transfer to occur in (send only, receive only, send & receive, off), we added two switches (Send/Tx and Receive/Rx) to the device's user interface. The design of this module was fairly straightforward. The positive terminal of the switches had a 10kΩ pull up resistor leading to V_{cc} while the negative terminal was fed to the circuit ground. The output of

both these user switches was sent to digital input pins on the microcontroller so that the mode can be read and corresponding data transfer can occur.

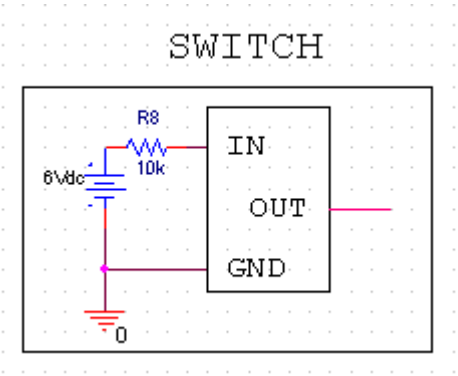


Figure 14: Circuit Schematic for Manual Switches

3. Design Verifications

3.1 Testing Procedures

Functional tests were preformed for all components in our design. The tests performed on the components in our Requirements and Verification will be explained in detail in the next section. The main testing we performed that was not specified in our design review document was the overall device functionality testing that we performed once the device was assembled. This testing was necessary because we wanted to make sure all the functions that we originally proposed worked as desired.

In order to perform the overall device functionality tests we uploaded the handshake code to both devices. After doing this we tested all 16 possible user switch configurations. To verify that each resulted in correct transfer for each device we read the contents of the memory out to the computer screen after every handshake. After some testing and debugging, all the modes worked as desired.

3.2 Test Results

3.2.1. Microcontroller Testing

In order to verify that our microcontroller was working correctly, we tested it with other components of our circuit. After doing so and successfully controlling all components of our device, we concluded that everything was working as desired.

3.2.2. Bluetooth Module Testing

In order for our project to function properly, we required that the Bluetooth modules connect and communicate without error within distances of 0.2m – 5m. Data transfer is required to reach up to 5 meters because firstly, Bluetooth technology usually has a delay of a couple seconds when making a connection and secondly, in case one of the users involved in handshake walks away right after handshake, the data transfer should still be completed without errors. It is also required to perform

the actual data transfer in less than 2 seconds and kill the established connection after data transfer is complete.

To verify these results we built two simple circuits. One was a stationary Bluetooth Module attached to the Arduino Uno. This Uno was connected to a laptop and was able to receive data input from the keyboard. The other circuit was on a smaller breadboard and consisted of another Bluetooth Module attached to an Arduino Pro Mini. This circuit was moved around to different distances from the stationary Bluetooth module and tested from various distances between 0.2m – 5m. The testing procedure was to send data from the keyboard to the stationary Bluetooth module that would connect to the mobile Bluetooth and send data to it. The mobile Bluetooth would receive this data and read it into the Arduino Pro Mini that it was attached to. The Pro Mini would then echo the data back to the Bluetooth and this data would then be sent back to the stationary Bluetooth. This data would be printed back onto the computer screen to verify it was the same data that we sent. Data transfer was successful at all tested distances at a *baud rate* of 9600 bits/second. A sample output and the code for each microcontroller are shown in Figure 15 below.

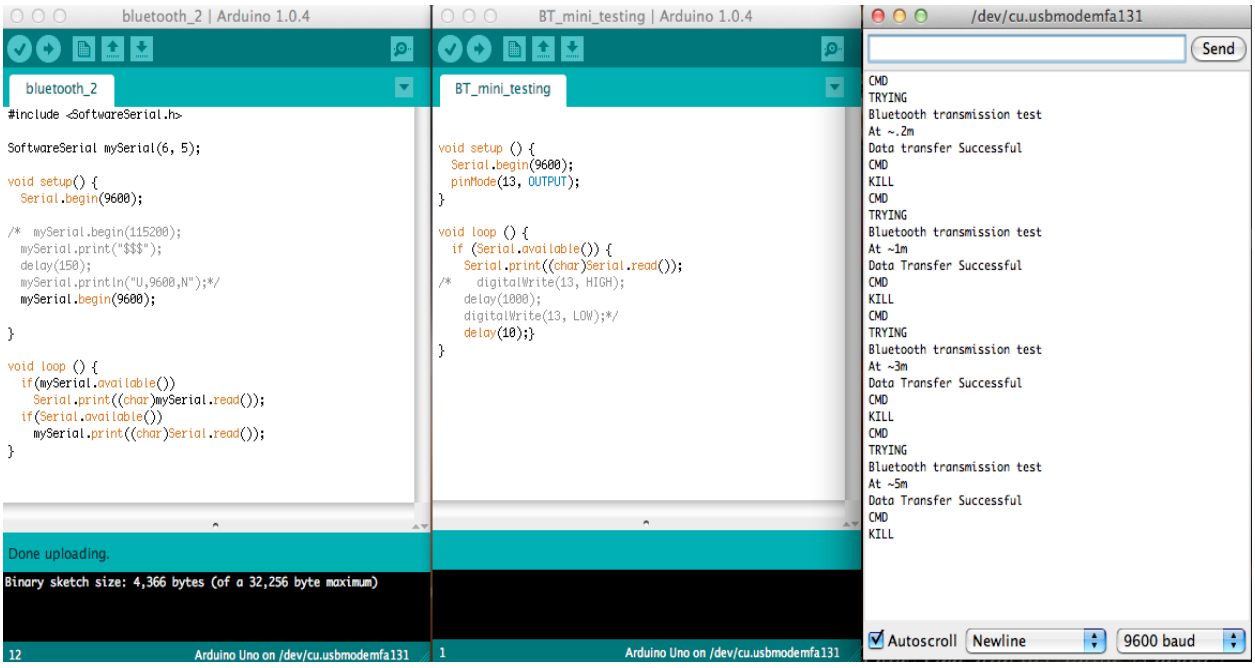


Figure 15: Bluetooth Testing Code and Sample Output

3.2.3. Accelerometer Testing

In order for our device to properly detect a handshake, our accelerometer was required to properly measure acceleration due to both gravity and motion on all 3 axes.

To verify that these requirements were being met, we connected each of the 3 axes' outputs to the oscilloscope independently. We verified that when no acceleration was being measured on the axis we were probing, the voltage on the oscilloscope would read a value around 3.3V. When we tilted the chip to measure acceleration due to positive and/or negative gravity, we verified that the two outputs were equidistant above and below the zero acceleration to due to gravity (0-level) output. After this requirement was verified we also wanted to make sure the accelerometer correctly measured acceleration due to motion (to detect an arm-swing or oscillations during handshake). In order to do this, we orientated the chip such that no acceleration would be measured on the axis of interest. After doing this, we shook the device and verified that corresponding acceleration due to motion was measured on the oscilloscope. This looked like the voltage level oscillating around the 0-level. Each of these tests was performed on all 3 axes.

Figure 16 shows a snapshot of the oscilloscope during the acceleration due to gravity testing. The middle trace on the plot is the 0-level (zero acceleration due to gravity) on the axis of interest. The two equidistant traces represent positive and negative acceleration due to gravity.

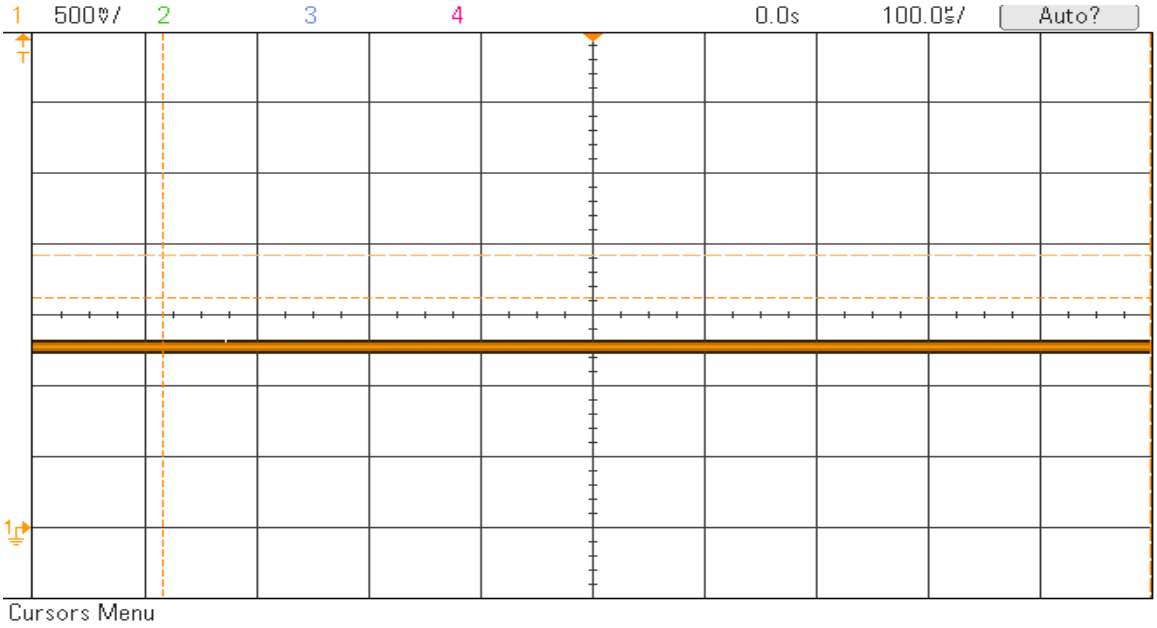


Figure 16: Accelerometer Output Measuring Acceleration Due to Gravity

3.2.4. External Memory Testing

Our external memory requirements were quite straightforward. The memory was required to properly write given data to the address specified by the code. It was also required to read the correct data from the address specified by the code.

In order to test these requirements we wrote 4 different "mock" business cards to subsequent memory locations. After these cards were all written to the memory we uploaded code that would read all of these business cards to the computer screen at one time. The output we viewed on the serial monitor verified that the data was correctly being written and read from the memory.

3.2.5. Low Battery Indicator Testing

The main requirement of the low battery indicator circuit was to turn on the warning LED when approximately 15% of battery charge remained (i.e. when the power supply voltage is equal to 4.8V)

To verify the functionality of this circuit, we input the DC power supply in the lab to the battery terminals of the circuit. We then swept the voltage of the power supply by 0.1V at a time and verified that the blue LED was off for $V_{supply} > 4.8V$ and was on for $V_{supply} < 4.8V$. The op amp output voltage that turns on the led is shown in Figures 17 and 18.

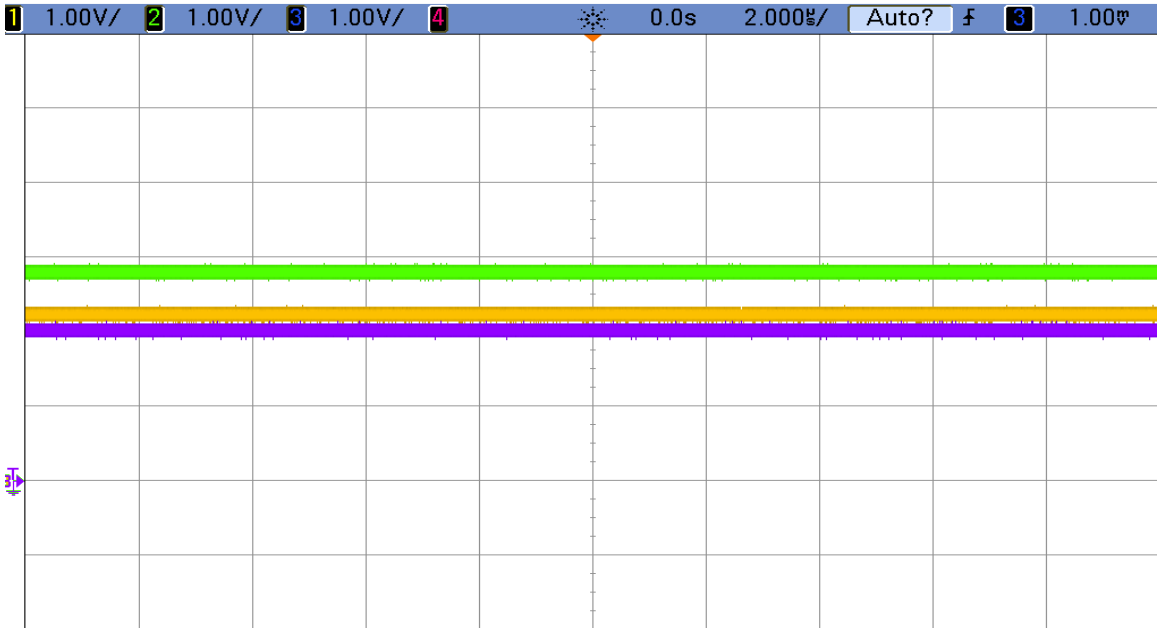


Figure 17: Op amp output (purple) is below turn-on voltage of blue LED (yellow) when power supply (green) is at full charge

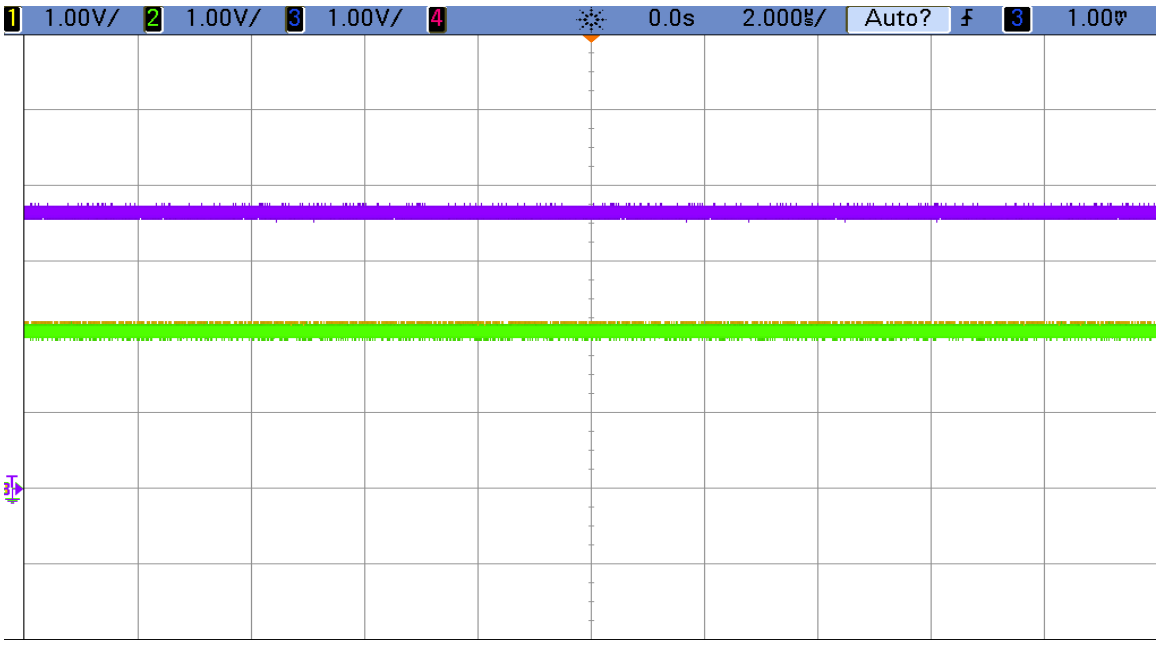


Figure 18: Op amp output (purple) is above turn-on voltage of blue LED (yellow) when battery voltage (green) is below 4.8V

3.2.6. Low Memory Indicator Testing

The low memory indicator LED was required to light up when the number of business cards in memory was greater than the low memory threshold we set up in the code.

In order to verify that this component was working we set the threshold to 2 cards and populated the memory with 2 business cards. We verified that the red LED turned on.

3.2.7. Power Supply Testing

Our power supply was required to supply a regulated 6 ∓ 0.5V to the RAW input to the Arduino Pro Mini.

To verify that the button cells were outputting this voltage we simply probed the ground and positive terminal of the power supply onto the oscilloscope. Figure 19 shows the power supply output.

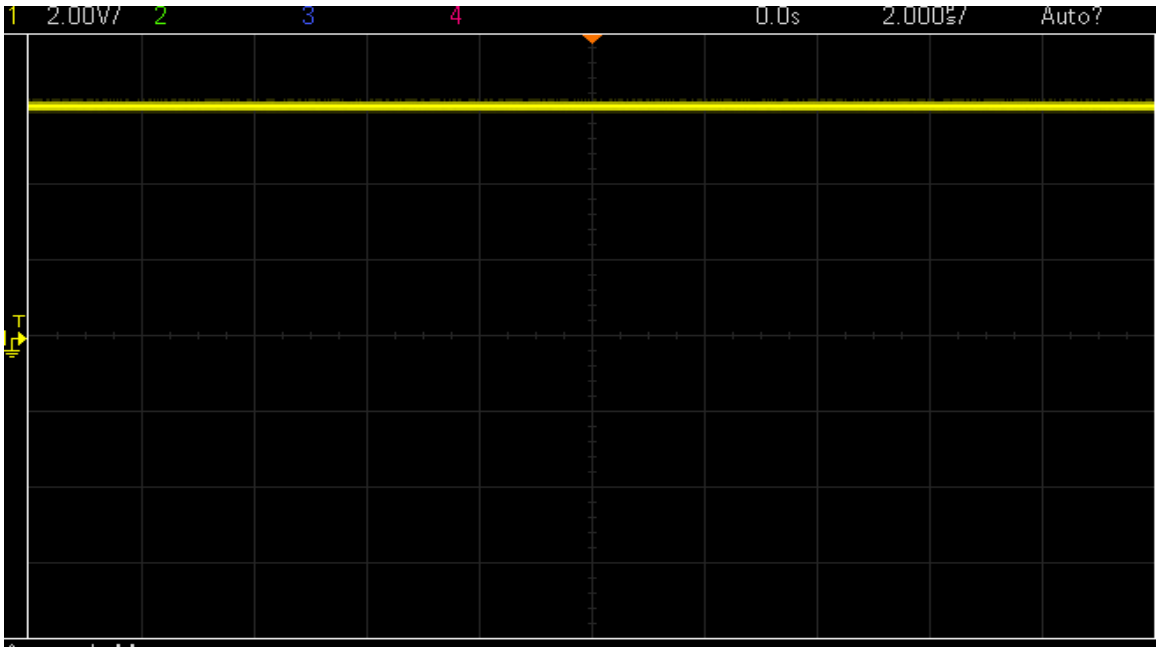


Figure 19: Output of Power Supply Module Supplying 6V ∓ 0.5V

3.2.8. Manual Switch Testing

The manual switches in our circuit were required to output logic high when on and output logic low when off. To test this for each switch we simply probed the output of the switch when logic high and low was on its inputs. We verified that the output toggled between high and low depending on the position of the switch.

3.3 Discussion of Failed Verifications

All of our verifications ended up passing except the power supply module. The button cells were able to supply power to separate components to the circuit but when we attached the power supply module to the entire load, the voltage swung greatly between 6V and under 3V. This might have been because the total load of the device was too large for the button cells and therefore, some components might not have been getting sufficient current.

A solution to this problem could be to step down the voltage of the power supply to approximately 4V (because maximum requirement for any component is 3.5V) before it is connected to the rest of the circuit. This decrease in voltage before the load would allow us to supply a greater current to the device.

4. Costs

Table 1 below summarizes the cost of each component that we used in building this device. It also compares the cost of the device we built in lab with the production cost. It can be seen that if we were to produce this device on a large scale, some of the biggest savings would stem from the accelerometer and Bluetooth. This is because for mass production, we would use the chip without break out boards. These two alone account for a savings of $40.99. On the whole, mass production would be $48.82 cheaper.

4.1 Parts

Table 1: Mass Production Cost for 1 Device

Part	**Quantity**	**Price ($)**		**Production Cost ($)**	
		Unit	Bulk	Unit	Bulk
ADXL335 Accelerometer	1	24.95	7.96	24.95	7.96
Microcontroller Arduino Mini Pro	1	9.95	7.96	9.95	7.96
RN-42 Bluetooth	1	39.95	15.95	39.95	15.95
External Memory	1	1.95	1.56	1.95	1.56
Power Supply (Lithium 3V button cells)	2	2.25	1.12	4.50	4.98
Battery Socket (BH1000G)	2	1.42	0.89	2.84	1.78
LED	2	2.49	0.28	4.98	0.56
Zener Diode	1	0.25	0.20	0.25	0.20
Op Amp	1	0.70	0.23	0.70	0.30
Resistors	8	0.05	0.05	0.40	0.40
Capacitors	3	0.10	0.10	0.30	0.30
Total				**90.77**	**41.95**

4.2 Labor

Table 2: Labor Costs

Name	Rate/hour	Overhead (x 2.5)	Hours *	Total ($)
W. Hanley	35	87.5	180	15,750
K. Samigollayev	35	87.5	180	15,750
A. Saha	35	87.5	180	15,750
Total				**$ 47,250**

Assuming a 15 hour work week for 12 weeks

4.3 Grand Total

Table 3: Total Costs

Section	Total ($)
Parts (bulk price)	41.95
Labor	47,250.00
Total	**$ 47,291.95**

5. Conclusions

5.1 Accomplishments

Our group was successfully able to complete what we wanted to do at the start of our project. We built a bracelet that successfully transfers business card data between two devices upon a handshake. All the user switch configurations ended up transferring data in the way they were designed to do. Not only did we have successful transfer between devices, but we also were able to transfer business card data upon a handshake to a Bluetooth terminal on a Smartphone.

We are also happy that we were able to achieve a reasonable size and weight for our final product. The bracelet has a similar size and weight of a standard wristwatch.

5.2 Uncertainties

Although we were able to complete all basic functionality of our device, we were not able to test it when there were more than two devices in close proximity to each other. If multiple devices are discoverable at the time the Master preforms the inquiry scan, then the Master might not connect to the correct device. This would cause data to be transferred to the incorrect person and raises privacy issues. We believe that during an inquiry scan the Bluetooth would connect to the closest available device it finds, but we are not completely sure about the functionality of the inquiry scan.

5.3 Ethical Considerations

The purpose of the *Automatic Handshake Info Exchanger* is to simplify and enhance the user's ability to exchange personal contact information. We strive to complete this project bearing in mind the IEEE Code of Ethics in Section 7.8 of IEEE Policies. The most relevant policies pertaining to our project are detailed as follows:

3. *To be honest and realistic in stating claims or estimates based on available data;*

Throughout the development of the device, we will follow the 3rd code closely, and only make claims and estimates based on real data acquired from our design. We will be honest and will not falsify the data acquired from our test procedures.

1. *To improve the understanding of technology; its appropriate application, and potential consequences*;
2. *To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations*;

After this project, we will have learned a great deal about various real-world technologies such as Bluetooth, the ATMega328p microcontroller, external Memory, motion sensors, accelerometers, gyroscopes. This will improve our understanding of these technologies and their applications, and also improve our technical competence, as directed in the 5th and 6th codes of the IEEE Code of Ethics.

3. *To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others*;
10. *To assist colleagues and co-workers in their professional development and to support them in following this code of ethics*.

Furthermore, while working on the project, we will build an environment that promotes engineer professionalism, which welcomes constructive and honest criticisms, acknowledges errors, assists peer workers with their professional and academic developments, and credits appropriate contributions, as cited in the 7th and 10th codes of the IEEE Code of Ethics.

9. *To avoid injuring others, their property, reputation, or employment by false or malicious action*;

We will make sure that the data transmitted through this device will offer the users with data that is as accurate as possible, and will not provide them with false information, in order to avoid damaging.

In addition to IEEE Code of Ethics, we will also ensure that privacy of the users is protected and that the device is transmitting the authorized data only.

5.4 Future Work

By the end of the semester, our group was happy that we achieved almost all of our design goals. However, there is always scope for improvements.

Most importantly, we would like to eliminate the Master/Slave switch (in order to eliminate the need to coordinate with the other user who is the *Master* and who is the *Slave* before every handshake). To tackle this, we came up with the solution that both Bluetooths will try to connect to each other at different random times (less than 1000 millisecond) after a handshake. As long as the random delays are staggered, either roe will be able to establish connection first. After they have both made an attempt we would send a command (*GK)* to the Bluetooth on each device to check whether they are connected or not. If they are connected then we can continue with data transfer. However, if they are not connected then try to connect again after another random delay on each module.

Other future work on our project includes

- Developing a user interface for uploading the business cards. This would include making an application for Smartphones and a computer program that can organize and store these uploaded business cards to files.
- We would also like to incorporate a picture to be transferred along with a business card
- Another area we would like to work on would be to reduce power consumption and increase lifetime of the device by using *sleep mode* when not in use.

References

1. UltraCAD Design, Inc. (n.d.). *Ipc trace/temperature charts*. Retrieved from http://www.ultracad.com/ipcchart.htm
2. Atmel. (n.d.). *8-bit avr microcontroller with 4/8/16/32k bytes in-system programmable flash*. Retrieved from http://www.atmel.com/Images/doc8161.pdf
3. Team Arduino. (2008, 08 15). *Original arduino mini design by team arduino*. Retrieved from http://arduino.cc/en/uploads/Main/Arduino-Pro-Mini-schematic.pdf
4. Sparkfun Electronics. (n.d.). *Arduino pro mini 328 - 3.3v/8mhz*. Retrieved from https://www.sparkfun.com/products/9220
5. Roving Networks Wireless For Less. (2009, 11 21). *Roving networks bluetooth product user manual*. Retrieved from https://www.sparkfun.com/datasheets/Wireless/Bluetooth/rn-bluetooth-um.pdf
6. Arduino. (n.d.). *Arduino pro mini*. Retrieved from http://arduino.cc/en/Main/ArduinoBoardProMini
7. Microchip. (n.d.). *256k i2c cmos serial eeprom*. Retrieved from http://ww1.microchip.com/downloads/en/DeviceDoc/20001203T.pdf
8. Sparkfun Electronics. (n.d.). *Led current limiting resistors*. Retrieved from https://www.sparkfun.com/tutorials/219
9. Electronics Club. (n.d.). *Light emitting diodes (leds)*. Retrieved from http://electronicsclub.info/leds.htm
10. MicroBatteryClub. (n.d.). *Lithium manganese dioxide battery cr2477 data sheet*. Retrieved from http://www.sony.net/Products/MicroBattery/cr/pdf/cr2477.pdf
11. Institute of Electrical and Electronics Engineers, Inc. (2012). *IEEE Policies*. Retrieved from http://www.ieee.org/documents/ieee_policies.pdf

Appendices

A. Requirements & Verifications

REQUIREMENTS	VERIFICATIONS	MET?
Power Supply		
1. The button cell batteries need to provide a stable voltage of to the Arduino board a) The batteries must provide the Arduino RAW input pin with a voltage between 3.5-12V, which it can regulate to 3.3V.	1. Measure the voltage across the battery using a multimeter. a) We expect the power supply to input 6 ± 0.5V when the batteries are fully charged.	NO
2. The Arduino's on-board voltage regulator needs to output a stable voltage from its V_{CC} pin a) A regulated 3.3V must be output from the Arduino V_{CC} pin to the other circuit components	2. Measure the voltage out of the Arduino V_{CC} pin using a multimeter. a) We expect the multimeter to read 3.3 ± 0.2V while the input voltage on the Arduino RAW pin is between 3.5-12V.	YES
Manual Switches		
1. Should output a logic LOW (0 ± 0.3V) when not pressed.	1. Connect terminals V_{cc} and GND to 3.3V supply and ground respectively. Using a multimeter, measure the output voltage when switch is not pressed.	YES
2. Should output a logic HIGH of (3.3 ± 0.5V) when pressed.	2. Using a multimeter, measure the output voltage when switch is pressed.	YES

Accelerometer		
1. Functions properly when powered by regulated 3.3V output from Arduino a) The X and Y accelerometer outputs should sense the acceleration due to gravity when the chip is held stationary on their respective axes. b) The X and Y outputs of the accelerometer should sense acceleration due to motion on their respective axes.	1. Use a multimeter to verify that the voltage on the accelerometer's V_{CC} pin is in the required range of 1.8-3.6V a) While powered, connect the X output to the oscilloscope and orient the accelerometer's X-axis parallel to gravity. Verify that the oscilloscope is reading a set value when the accelerometer is held stationary. Do the same for the Y output. b) Move the accelerometer in the X and Y directions while viewing their outputs on the oscilloscope. Their outputs should oscillate around their offset (from gravity) corresponding to motion on their axes.	YES
2. Should correctly output data at a sufficient number of measurements per second (100 Hz), in order to detect a handshake from the use a) The capacitor attached between the accelerometer's X_{OUT} or Y_{OUT} and GND should set the sample frequency to 100 Hz ± 25%. [2]	2. Connect sensor output pin to A_0/A_1 on the Arduino and write software to print the data the Arduino is receiving from the accelerometer on the computer monitor. a) If the frequency of samples is too low then decrease the value of the capacitor between X_{OUT} or Y_{OUT} and GND until the samples are being output at 100 Hz ± 25%.	YES

Microcontroller		
1. The ATmega328P chip should function appropriately when the voltage supply on the Arduino's RAW pin is in the range 3.5V – 12V. a) The logic high output should be 3.3V ± 0.5V with a logic low of 0V ± 0.5V.	1. Supply RAW pin with a voltage of 6V. Verify the Arduino chip with a simple test code setting one digital output HIGH and another one LOW. a) Connect multimeter to the output of the chip and verify that the output high is 3.3V ± 0.5V and output low is 0V ± 0.5V.	YES
2. Required to correctly detect when a handshake has occurred by using mock data from it's Analog Input pins A_O and A_1 a) The Arduino must not detect a "false handshake" and initiate unwanted data transfer from this mock input data	2. Write a software algorithm to process the mock data. Hook up a simple resistor and LED between an Arduino digital output pin and GND. Turn the LED on when the algorithm detects a handshake from the user. a) Test this algorithm by inputting data created to simulate the accelerometer data when a user is wearing it. Verify the LED only turns on during a handshake.	YES
3. Microcontroller should complete communications based on which manual push button switches are pressed a) If the TRANSMIT switch is pressed then data should be sent to the serial input of the Bluetooth	3. Tie the switch inputs to the Arduino to either HIGH or LOW and have the Arduino detect a mock handshake to initiate communication a) When the TRANSMIT switch is pressed view the microcontroller output to the Bluetooth on the computer and verify that it matches the mock data to be sent	YES

b) If the RECEIVE switch is pressed the microcontroller should take data input from the Bluetooth and send it to the external memory	b) When the RECEIVE switch is pressed view the microcontroller output to the external memory on the computer screen and verify that it matches the mock data input from the Bluetooth	
4. After data transfer between two devices is finished end connection between Bluetooth modules.	4. Check that after communication between devices is over the blinking LED on board the Bluetooth module turns off, signaling the connection is over.	YES
Bluetooth		
1. The Bluetooth Mate Silver should function properly when supplied with the regulated 3.3V from the Arduino supply.	1. Use a multimeter to measure the voltage on the Bluetooth's V_{CC} pin and verify that it is between 3.3-6V.	YES
2. The Bluetooth modules should be able to communicate without error when located within a range of 0.2-5m. a) Bluetooth Module can correctly and efficiently (<2s) transmit data from these distances b) Bluetooth Module can correctly and efficiently (<2s) receive data from these distances	2. With Bluetooth Module kept 0.2m, 1m, 3m, and 5m away from the computer's Bluetooth, establish connection between the two modules: a) Transmit 200 bytes of mock data from Bluetooth module to the computer's Bluetooth. Verify that the correct data is transmitted at each distance by viewing the received data on the computer screen. b) Receive 200 bytes of mock data from the computer's Bluetooth to the Bluetooth Module. Verify that the correct data is received at each distance by viewing the received data on the computer screen.	YES

External Memory		
1. The external memory should function properly when powered with regulated 3.3V output from Arduino. a) When performing a write operation the data should be stored in the correct address which is specified by the address byte b) When reading from a memory address the correct data should be printed to the computer screen	1. Use a multimeter to test that the input voltage is within the required range (2.7-5.5V) a) Testing code will send known random data to be written to specified external memory addresses b) Will use software to read data contents from these addresses and display it on the computer monitor to verify that the memory read and write operations are functioning properly.	YES
Battery Life Indicator		
1. The Battery Life Indicator should properly detect when the voltage on its input falls below the reference threshold a) The blue LED on the output of the Battery Life Indicator should turn on when the batteries are low (~15% charge remaining)	1. Provide the Battery Life Indicator circuit with a range of sample input voltages and test the output of the red LED a) Use a multimeter to sweep the battery input voltage from 6V to 2V in increments of 0.1V. Verify that the red LED turns on when the input voltage is about 4.5V.	YES

B. PCB Layouts and Schematics

B.1 Lateral Side PCB

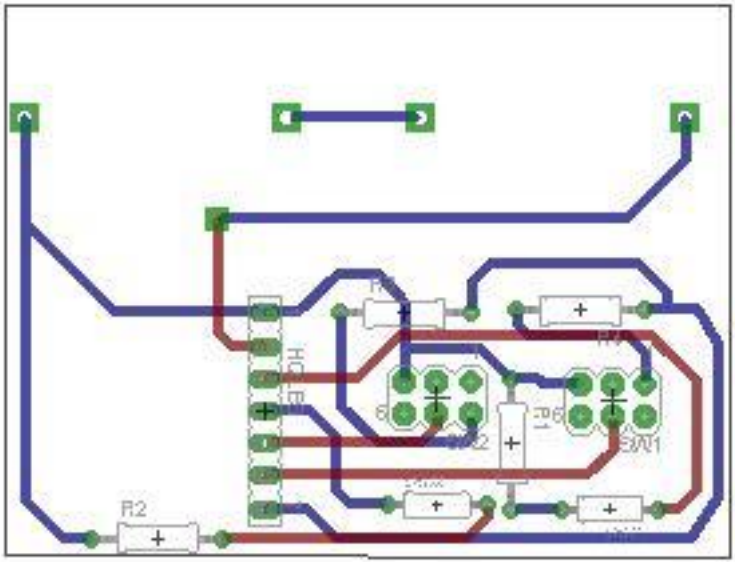


Figure 20: Lateral Side PCB Layout

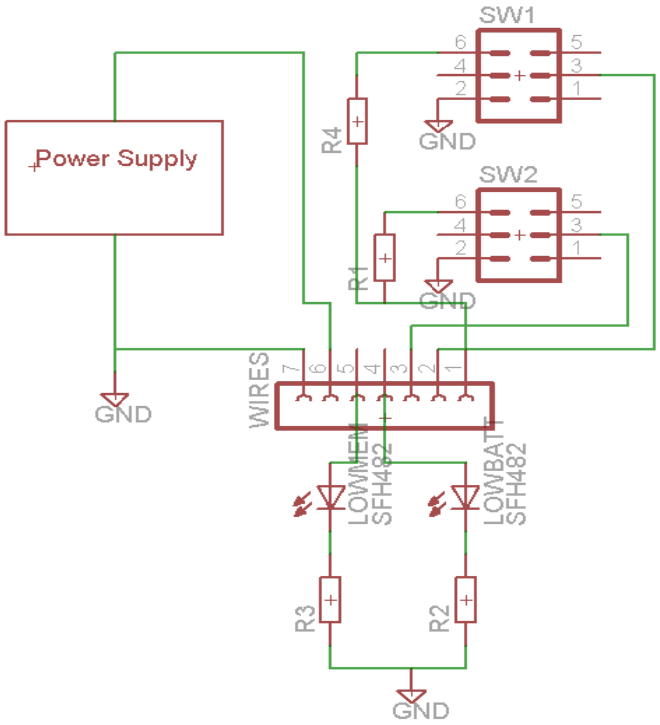


Figure 21: Lateral Side PCB Schematic

B.2 Medial Side PCB

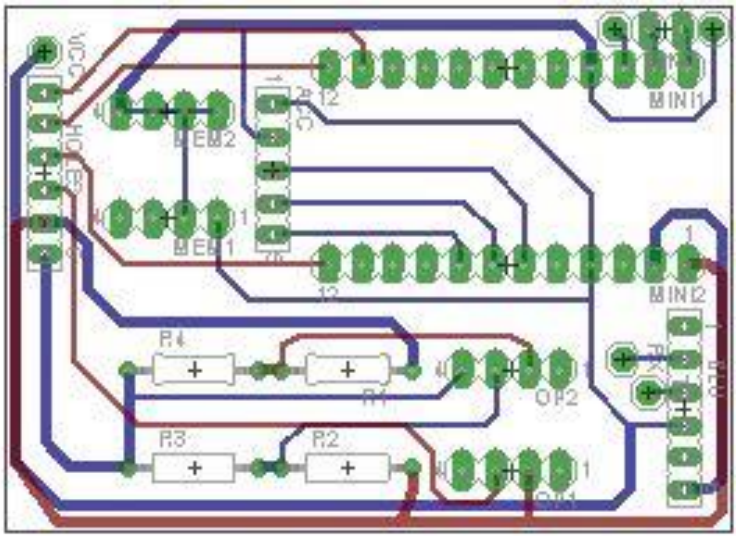


Figure 22: Medial Side PCB Layout

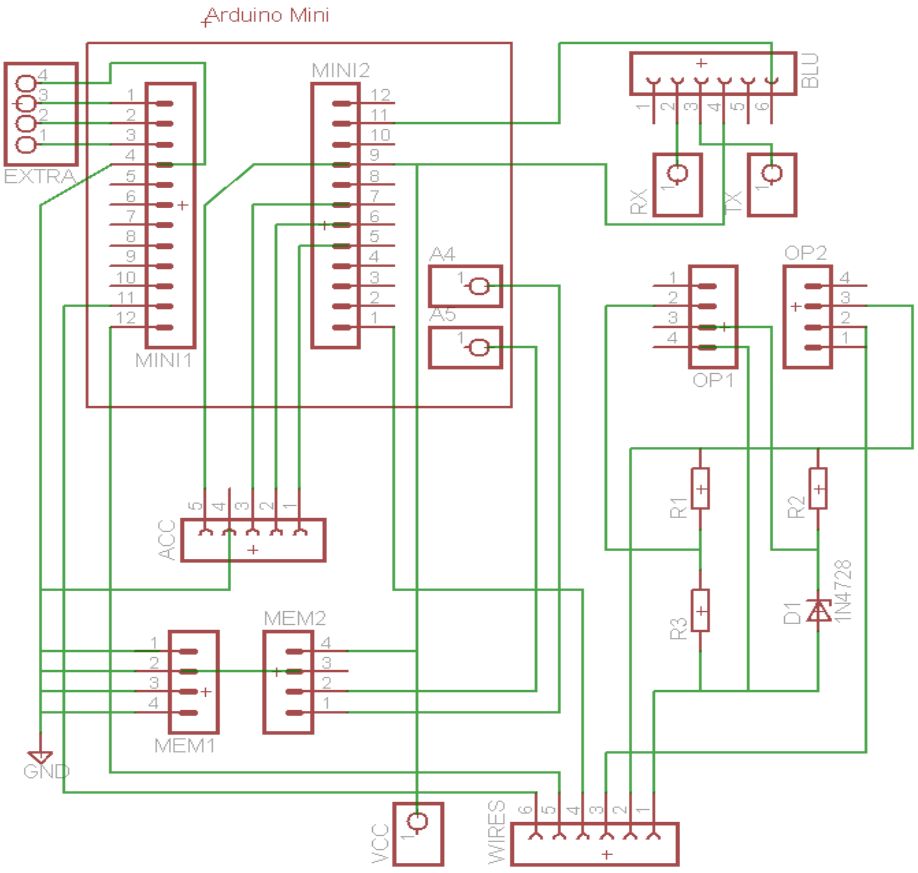


Figure 23: Medial Side PCB Layout

C. Microcontroller Code

In the interest of space, we uploaded all the necessary code to ".txt" files on our project page.

C.1 Main Device Code

C.2 Memory Read Code