University of Illinois

Department of Electrical and Computer Engineering

Tim Capota and Dave Drake

Friend Finder Armband

Final Report

TA: Jane Tu

05/01/2012

Team #30

Table of Contents

1. Introduction

We developed a pair of armbands that allow the users to easily locate each other. Using RF communication we are able to compare signal strengths and indicate the location and distance intuitively via LEDs and vibration, respectively.

1.1 Motivation: Many times GPS devices can not be used to locate a person in a big crowd. They are not accurate enough to tell you exactly where a person is within close distances, and sometimes the signal is unreliable. The friend finder armband aims to fill this void in the market by creating an intuitive tool to easily find others which works where GPS does not. Our product would be useful in a myriad of situations including concerts, malls, and any other events where large gatherings of people make getting separated easy.

1.2 Performance Requirements

- Indicate location via LEDs from 3ft to 200ft outdoors
- Indicate location via LEDS from 3ft to 100ft indoors
- Accuracy of directionality of 60 +/- 30 degrees
- Indicate close proximity via vibration from 10ft to 50ft
- Armband lasts at least 5 hours

1.3 Block Diagram:

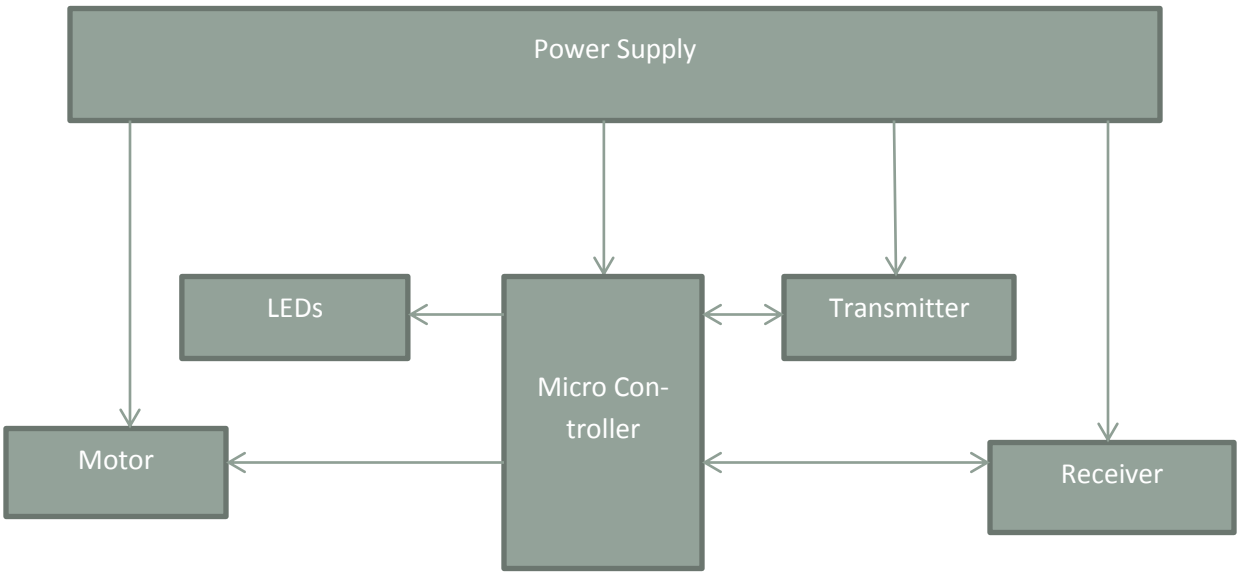

Figure 1.1 Block Diagram

1.4 Subprojects

The design was broken up into 6 modules, each with specific tasks.

1.4.1 Power Supply: The main power will be from four AA batteries. It will supply 3V power to the motor, microcontroller, transmitter, and receiver.

1.4.2 LEDs: This module contains three LEDs that indicate the direction of the paired armband. The LEDs each receive a signal from the microcontroller telling them when to turn on. The LEDs get power from the microcontroller. There is also a switch that is ANDed with the signal from the microcontroller, giving the user the ability to turn off the user interface.

1.4.3 Motor: The motor module contains two dual 4 input multiplexors and a motor. The multiplexors receive select signals from the microcontroller to control what value of current goes to the motor. The motor has eight current levels to control vibration intensity. The motor gets 3V from the power supply.

1.4.4 Microcontroller: This is the main brains of our design. The microcontroller sends data to the transmitter to send. It receives incoming data from the receiver and the RSSI (received signal strength) information. The incoming data from the receiver is line passed and uses the UART ports, not requiring A/D conversion. The UART data uses an API frame that allows us to decode the packet to get RSSI values. It selects which receiver to get incoming data from by controlling two mux select signals. By comparing the strength of the received signals (from the RSSI values) it controls the LEDs to indicate position. Each LED corresponds to the direction of each antenna respectively. It will use the magnitude of the strongest signal to control the motor current by setting the mux selects, such that the motor vibrates more intensely with stronger signal strength. The LED and mux enables are ANDed with a switch giving the user the ability to turn off the user interface. The microcontroller gets 3V from the power supply.

1.4.5 Transmitter: This module will transmit a signal over a single antenna indicating the presence of the friend finder armband. It receives data from the microcontroller to transmit. The transmitter gets 3V from the power supply.

1.4.6 Receiver: This module will use three antennas and receivers to receive the signal transmitted from the other armband. They each output the incoming data (sent from the transmitter) and the RSSI (received signal strength) to the microcontroller using an API frame over a UART connection. All receivers output data to a mux since there is only one hardware UART on the microcontroller. The receiver gets 3V from the power supply.

2 Design

2.1 Design Procedure

2.1.1 Power Supply: All modules run on 3V, so we needed to create a 3V output coming from this block. We also wanted it be wearable. The simplest way to implement this was to use two AA batteries in series. We could have used a larger battery and stepped down the voltage, but this added unnecessary complexity and cost to the design.

2.1.2 LEDs: The LEDs needed to turn on when the microcontroller determined which antenna had the strongest signal. They needed to operate at 3V. We could have used any LEDs for this, but we choose blue ones because they have a similar operating voltage to other components.

2.1.3 Motor: The motor needed to operate at a range of voltages around 3V so we could control the vibration. The range of vibration intensity also had to be large enough to be noticeable to the user. The motor could have been a DC motor with an imbalanced weight, but we used a pancake style vibrating motor because it needed to be compact and wearable in our armband.

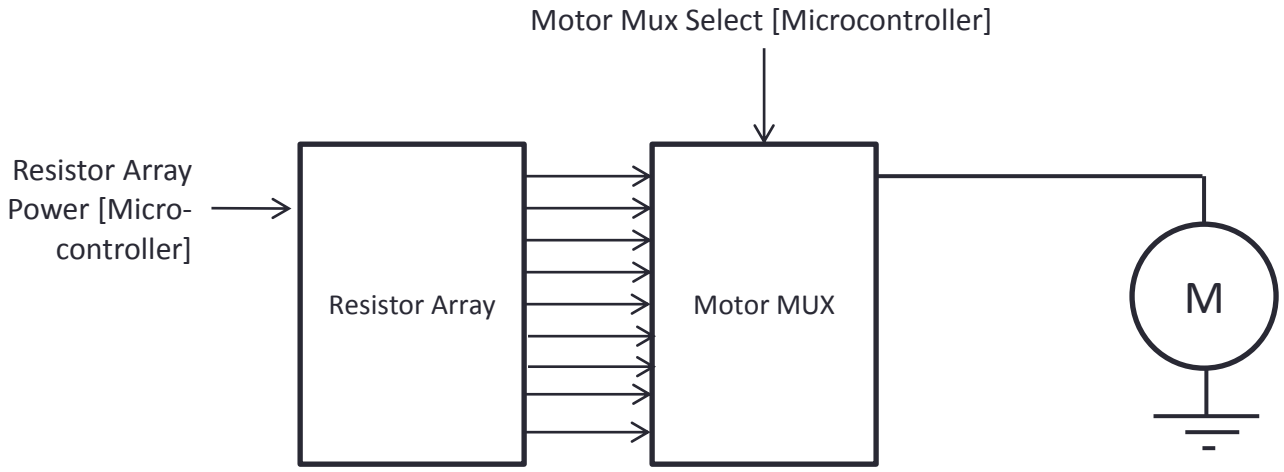

Figure 2.1 Basic Motor Design

2.1.4 Microcontroller: The microcontroller needed to operate at 3V. It also needed to be able to receive and send UART data, and provide outputs for motor selects, receiver selects, and LED enables. We could have used other I/O pins and software UART, but hardware UART is much more reliable so the mux made more sense.

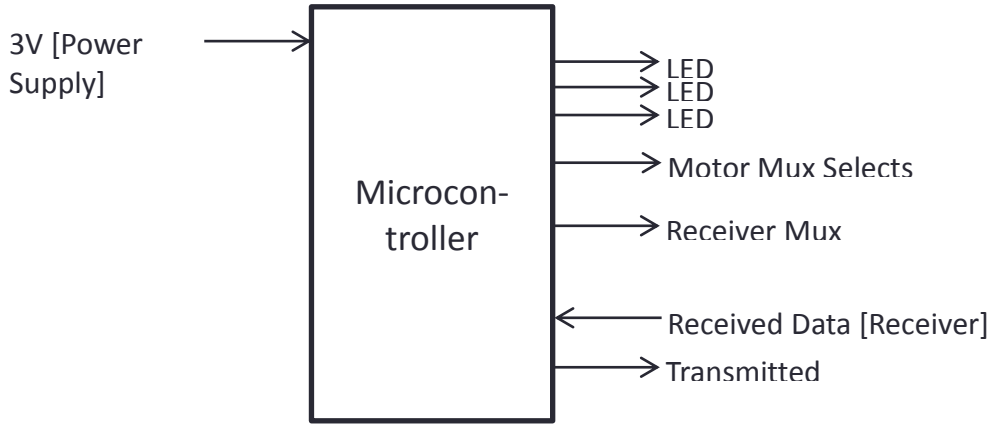

Figure 2.2 Basic Microcontroller Design

2.1.5 Receiver: The receiver needed to operate at 3V. It also needed to get three signals from different antennas, but only pass one to the microcontroller. The best way to do this was to use a mux and have the microcontroller cycle through the different inputs to the mux once each RSSI signal was stored. We needed the receiver to have UART so it was compatible with the microcontroller

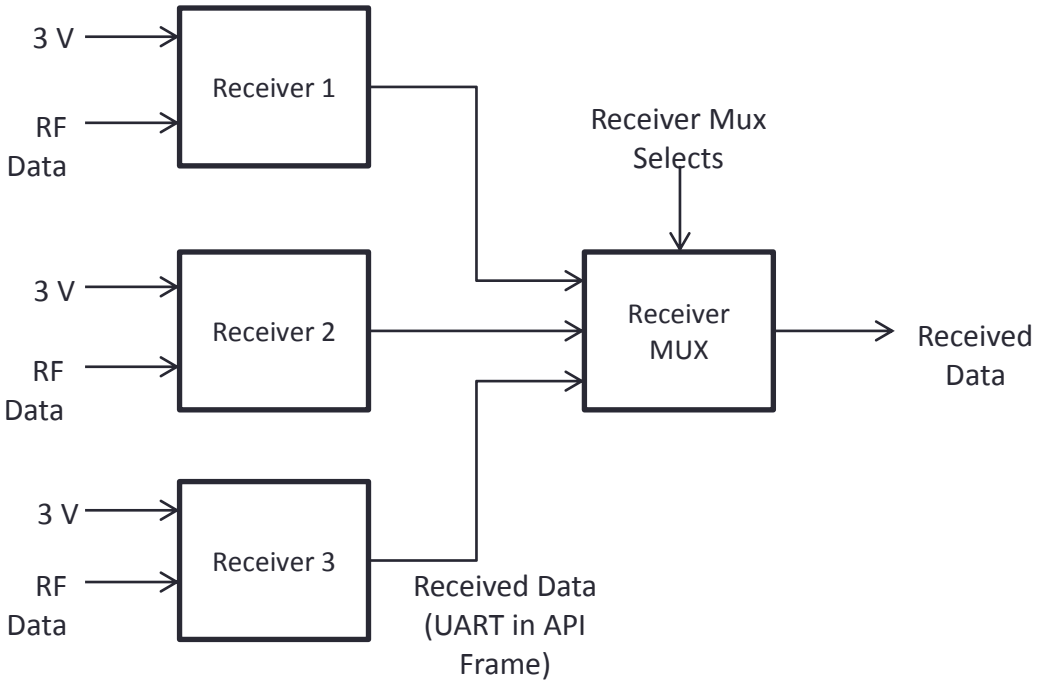

Figure 2.3 Basic Receiver Design

2.1.6 Transmitter: The transmitter needed to operate at 3V. It also needed to communicate via UART with the microcontroller and have a range greater than 100ft indoors and 200ft outdoors.

2.2 Design Details

2.2.1 Power Supply: For the power supply we used four lithium ion AA batteries to get the required lifetime. We calculated the current drawn from our load using specifications from the datasheets [1]-[6]:

- XBee-PRO transmitter: 215mA
- Xbee-PRO receiver: 55mA*3
- LEDs: 30mA*3
- Motor: 53mA
- Microcontroller: 0.22mA
- Multiplexors: 6.5mA*3
- AND gate: .044mA

The total current drawn is 542.8mA. Our batteries have a lifetime of 2900mAh [7]. This gives an armband lifetime of 5.34 hours which meets our specified performance requirements. By connecting two batteries in series we created the necessary three volts for all our modules. The schematic is shown in appendix A.1.

2.2.2 LEDs: We chose blue LEDs because they can have a forward voltage of up to 3.6V. This was perfect because our microcontroller outputs at 3V. We used a pull up resistor with a value of 150 Ohms to limit the current to 20mA through the LED. The schematic is shown in appendix A.2.

2.2.3 Motor: We chose a Pico Vibe 12mm Vibrating Pancake Motor because it operated at 3V, and had amplitude range for different currents as shown below in Figure 2.4. We used two dual four input multiplexors to control the current to the motor. We had a range of resistors attached from the microcontroller to the eight of the inputs on the mux. Our resistors ranged from 1 Ohm to 55 Ohms in increments of roughly 7 Ohms. This gave us currents ranging from 27mA to 53mA, because the motor has an operating impedance of 55Ohms [3]. The lowest resistance is then 56Ohms (when our mux connects the 1 Ohm resistor to the motor), this means:

$$3V/56\ \text{Ohms} = 53mA$$

Similarly, the highest resistance is 110 Ohms when we have the smallest current:

$$3V/\ 110\ \text{Ohms} = 27mA$$

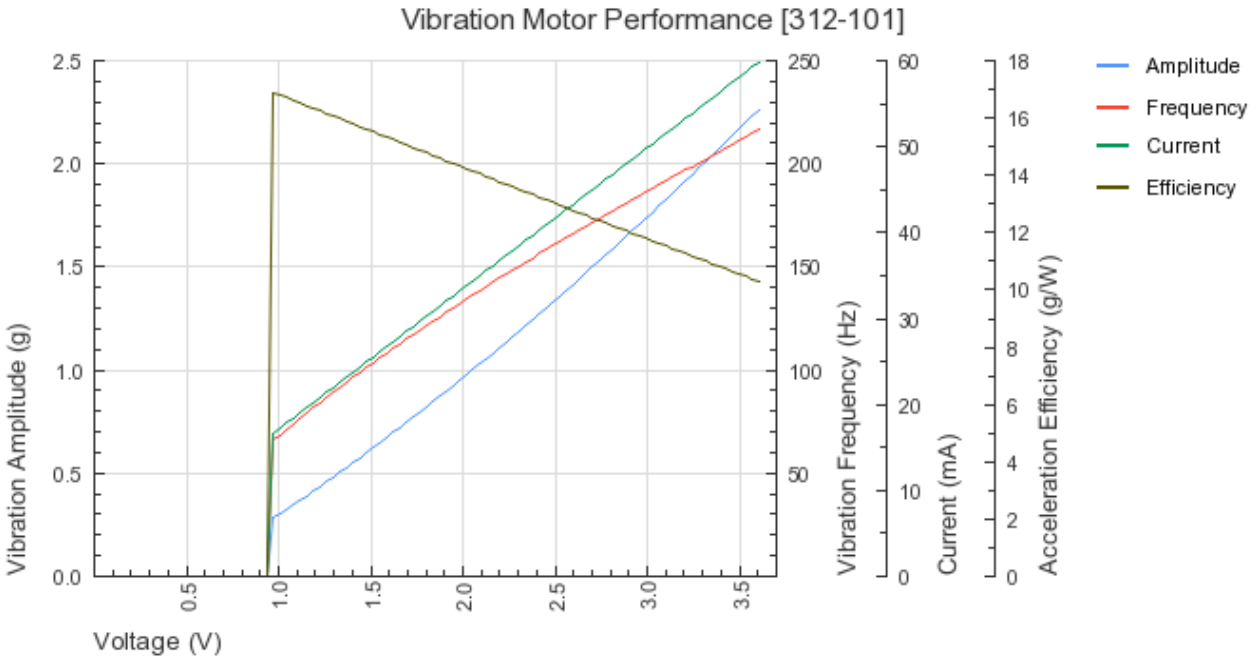

Figure 2.4 Amplitude vs. Voltage/Current [3]

The schematic for the motor is shown in appendix A.3.

2.2.4 Microcontroller: We used a PIC 16F887a as our microcontroller. This worked well because it had hardware UART and a wide operating voltage of 2.2V-5.5V [4]. This worked perfectly with the 3V we were outputting from our power supply. It also had enough digital I/O ports to control our motor resistance network, receiver mux, and LEDs. We programmed it using C following the flowchart shown in appendix B.1. The code is relatively simple; since we know the data packets from the transmitter are sent in an API format, we can look for the address of the transmitter (which we programmed), and then the next byte contains the RSSI data we need (as shown in Figure 2.5). Once an RSSI value is stored, we increment the receiver mux and wait for the next RSSI value. Once we have all three, we compare the signals and store the strongest signal. The LEDs are activated according to the compare code. The strongest signal is then referred to stored reference signals from 10ft to 50ft, which controls motor vibration. The schematic is shown in appendix A.4. The code is shown in appendix B.2.

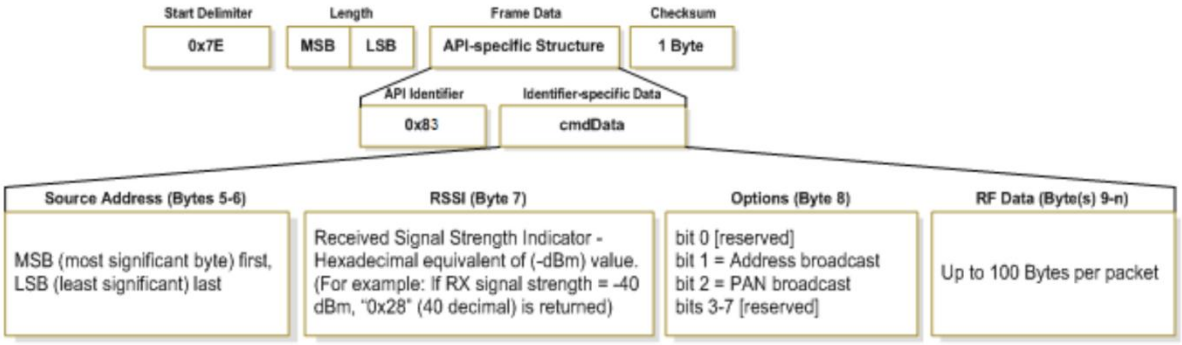

Figure 2.5 API format [1]

2.2.5 Transmitter: We used an Xbee Pro 802.15.4 because it has a range of 300ft indoors, and a mile outdoors [1]. It also uses UART to communicate and can output the RSSI value in an API format. This allowed us to seamlessly send data from the microcontroller. We enabled API format by setting IU=1. We programmed the source address of one armband to 1234, and the other to 2468. We also programmed the destination address to 5678 on one armband, and 1357 on the other. This allowed us to connect one transmitter to the other three receivers for both armbands without interference. The schematic is shown in appendix A.5.

2.2.6 Receiver: Three Xbee Pro 802.15.4 were used for the receiver. These were arranged in a triangular pattern to locate the direction of the transmitter. RF shielding was placed inside this triangle as shown in Figure 2.6 to attempt to create directional antennas. This way we could get directionality of 60 +/- 30 degrees. However, the RF shielding was not effective enough to attenuate the RSSI values. Also due to reflections in an indoor environment the strongest signal did not correspond to the closest antenna consistently. The schematic is shown in appendix A.6

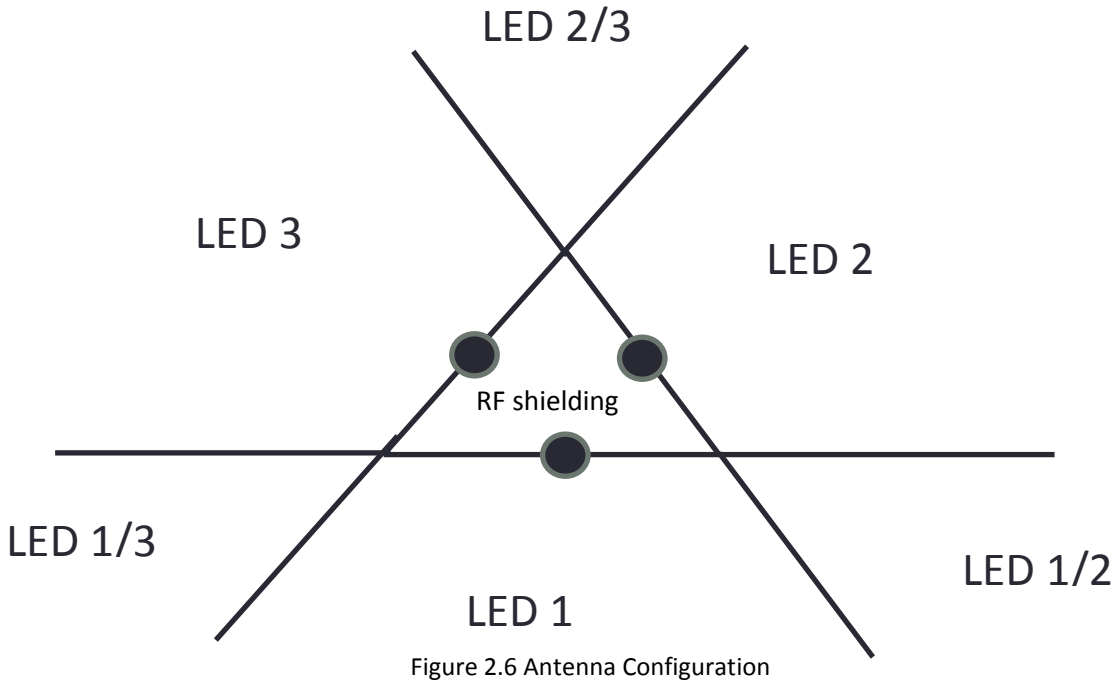

Figure 2.6 Antenna Configuration

3 Verification

Our requirements and verification table are shown in appendix C.

3.1 Power Supply: Power supply was connected to all modules during test. Output was measured using an oscilloscope. Under full load our power supply outputs a clean 3V signal as shown in Figure 3.1

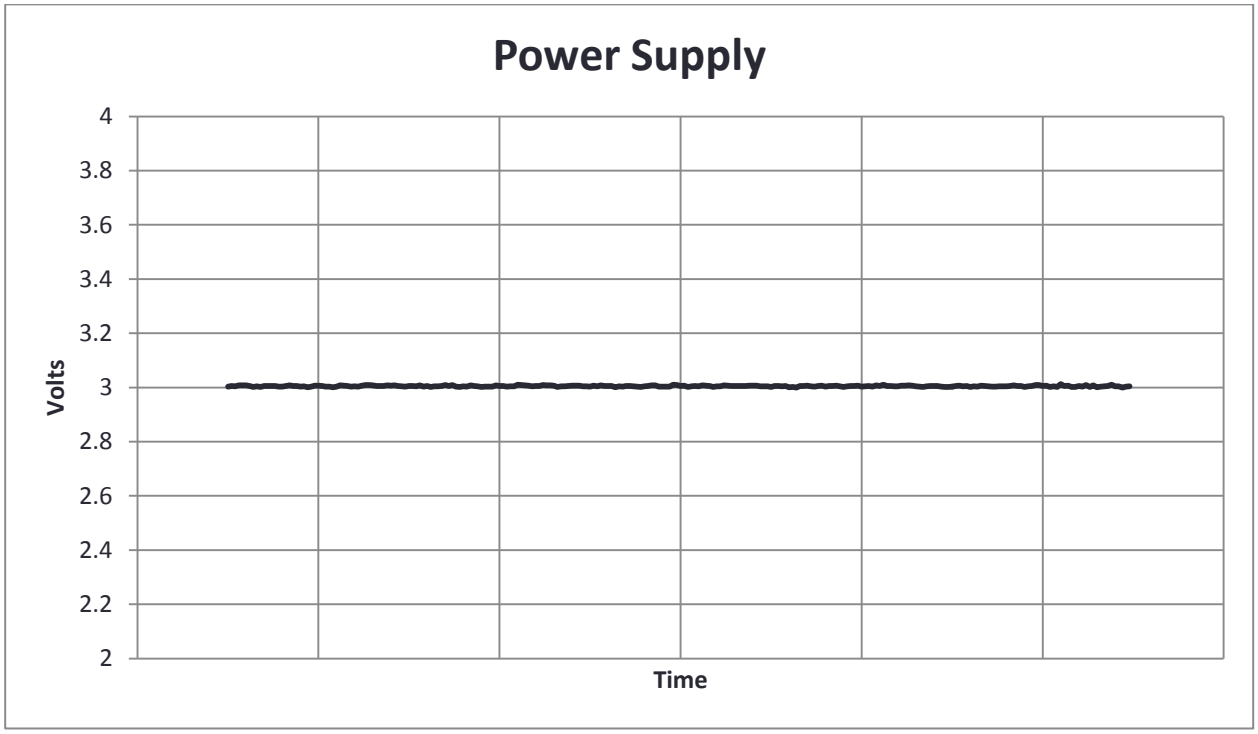

Figure 3.1 Power Supply Voltage Under Full Load

3.2 LEDs: The LEDs were connected to a 3V DC power supply through 150 Ohm resistors. When connected they illuminated as expected.

3.3 Motor: Motor current was measured for two different mux inputs (one corresponding to a close distance, the other to a far distance). At a distance of 10ft our mux selects are 0 and 0, as shown in figure 3.2. We measured the current using a multimeter and got a value of 56.4mA. At a distance of 50ft our mux selects are 1 and 1, also shown in figure 3.2. We measured the current using a multimeter and got a value of 28.3mA. This was what we expected and met our requirements.

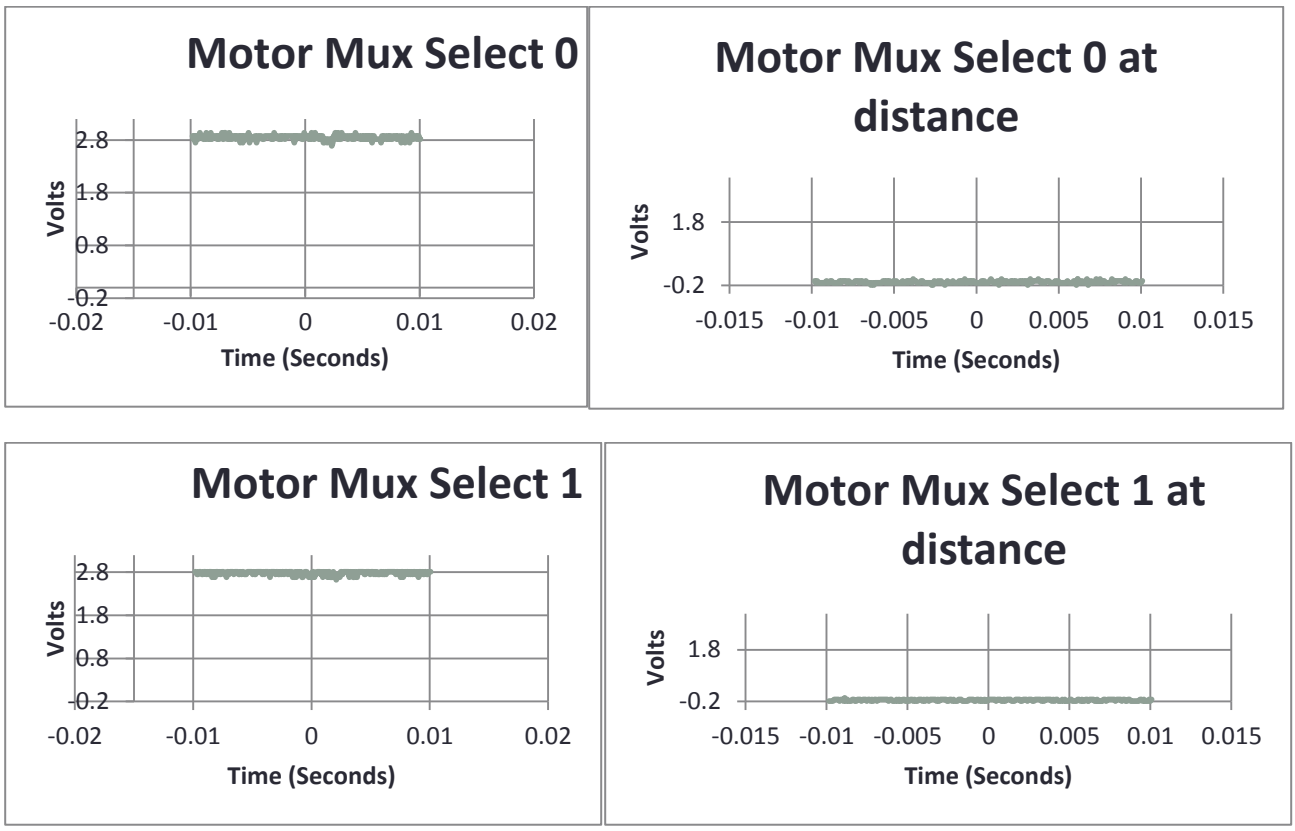

Figure 3.2 Motor Mux Selects at 10ft and 50ft

3.4 Microcontroller: The microcontroller had stored RSSI values to test the compare code. It correctly illuminated the LED corresponding to the strongest signal and the test passed. To test if the microcontroller can receive RSSI values from all three receivers we brought the receivers close together and verified that all three LEDs were illuminated, since they were all equal in strength. As shown in Figure 3.3 the microcontroller correctly cycles the receiver mux to receive all RSSI signals. We programmed the microcontroller to have three signals and verified that the strongest signal activated the right motor control. As shown above in Figure 3.2 the microcontroller correctly controls the mux signals to send the right current to the motor based on distance.

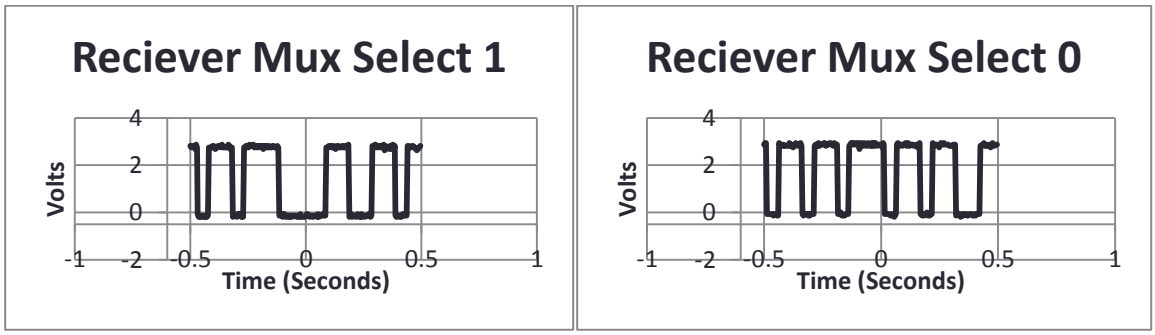

Figure 3.3 Receiver Mux Selects Controlled by the Microcontroller

3.5 Transceiver: We tested that the transmitter and receiver could send data by hooking up the UART ports to an oscilloscope. As shown in Figure 3.4, the transmitter continually sends the character 'a' and the receiver receives this data in the API format as discussed previously. We tested that the transmitter and receiver could send signals omnidirectionally by moving the transmitter completely around the receivers and verifying that the LEDs illuminated. The LED is only illuminated through the microcontroller once data is received.

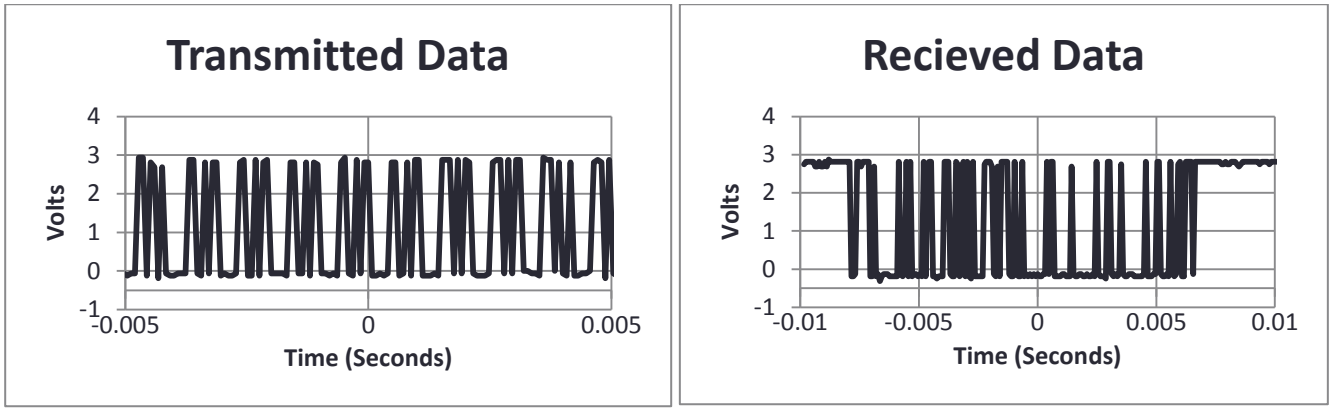

Figure 3.4 Transmitted Data and Received API Packet

4 Cost

4.1 Parts: The main cost of the parts was from the Xbee Pro 802.15.4, with each priced at $38.00, contributing a total cost of $304.00. The total cost for parts was $363.94. These costs were calculated without considering bulk discounts, so this number is an overestimate.

4.2 Labor: Our labor costs were calculated with a pay rate of $40/hr using formula 4.1:

$$\text{Labor Cost} = \text{Ideal Hourly Salary} \times \text{Hours Spent} \times 2.5 \tag{4.1}$$

We estimate the time worked to be approximately 120 hours. Using 4.1 we calculate a labor cost of $12,000 per person. Note that we did not include any shop hours since we did almost all the work ourselves. This combined with our parts cost gives a total cost of $24,363.94 as shown in Appendix D.

5 Conclusion

The friend finder armband met all but one of our design requirements. We were able to send and receive wireless data, get signal strength and run a comparison, and control motor vibration intensity and LEDs according to the specifications we laid out. We were not able to get a directional receiver, so the directionality was unsuccessful. The main problem with this was that the RF shielding did not attenuate the signals enough to make a discernible difference in the RSSI values. Another main issue was all the reflections in an indoor environment caused the strongest signal value to vary independently of direction and distance.

For future improvements we would eliminate the directionality component. This is not necessary to have a good product, because the motor vibration intensity is an intuitive user interface alone. It also greatly reduces the price because we no longer need four antennas per armband. Instead we recommend using the RFM12B-S2 wireless transceiver, which only costs $6.00. We believe we could reduce the cost to around $25 per pair, a marketable cost. The RSSI can still be pulled and compared, so the motor interface and microcontroller would work the same. Another feature we recommend implementing is a user programmable multipoint capability, which would allow more than two armbands to have the same source address.

References

[1] Xbee/Xbee-PRO RF Modules Product Manual v1.Ex– 802.15.4. Digi International Inc. [Online]. Available: http://ftp1.digi.com/support/documentation/90000982_F.pdf

[2] Xbee-Pro 802.15.4 Datasheet. Digi International Inc. [Online]. Available: http://www.digi.com/pdf/ds_xbeemultipointmodules.pdf

[3] Pico Vibe 12mm Vibration Motor – 3.4mm Type Datasheet. Precision Microdrives. [Online] Available: https://catalog.precisionmicrodrives.com/order-parts/product/312-101-12mm-vibration-motor-3-4mm-type

[4] PIC 16F887a Datasheet. Microchip. [Online]. Available: http://www.ti.com/lit/ds/slas639b/slas639b.pdf

[5] TI SN54AC08 Datasheet. Texas Instruments. [Online]. Available: http://www.ti.com/lit/ds/scas536d/scas536d.pdf

[6] TI SN74ALS153N Datasheet. Texas Instruments. [Online]. Available: http://www.ti.com/lit/ds/sdas206a/sdas206a.pdf

[7] L91 Datasheet. Energizer. [Online]. Available: http://data.energizer.com/PDFs/L91.pdf

Appendix A: Schematics

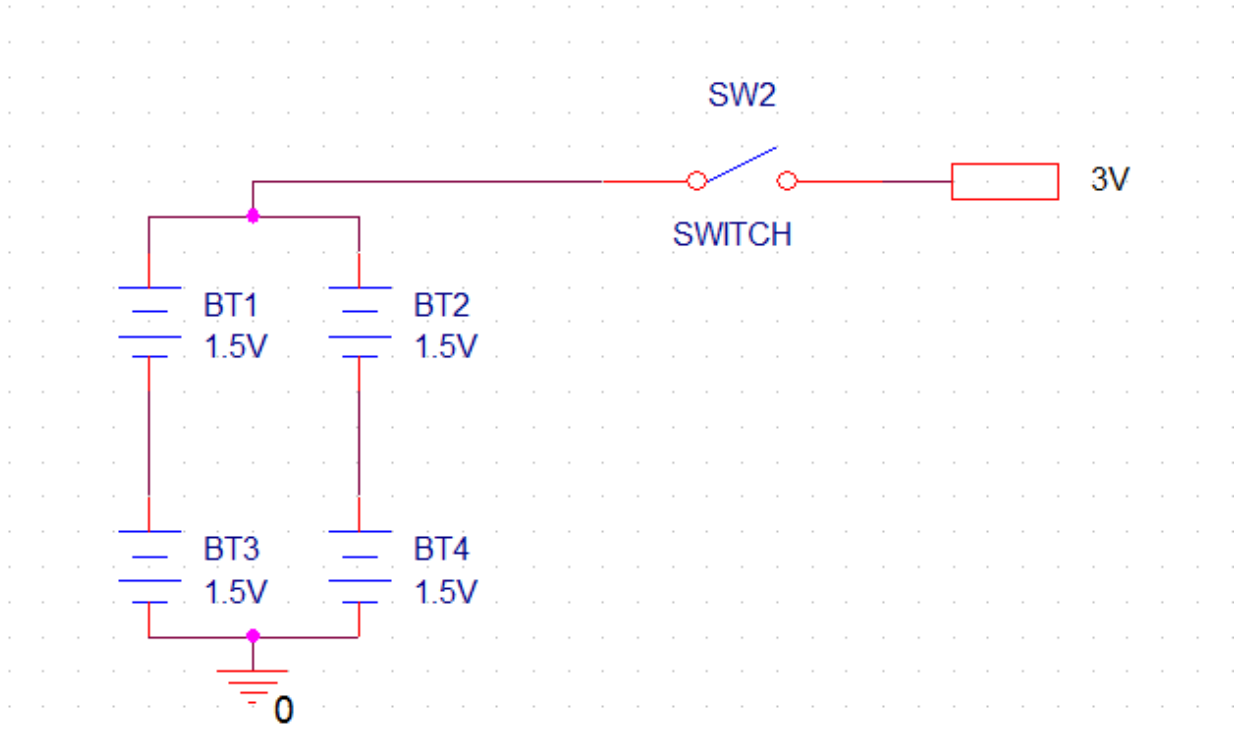

Figure A.1 Power Supply Schematic

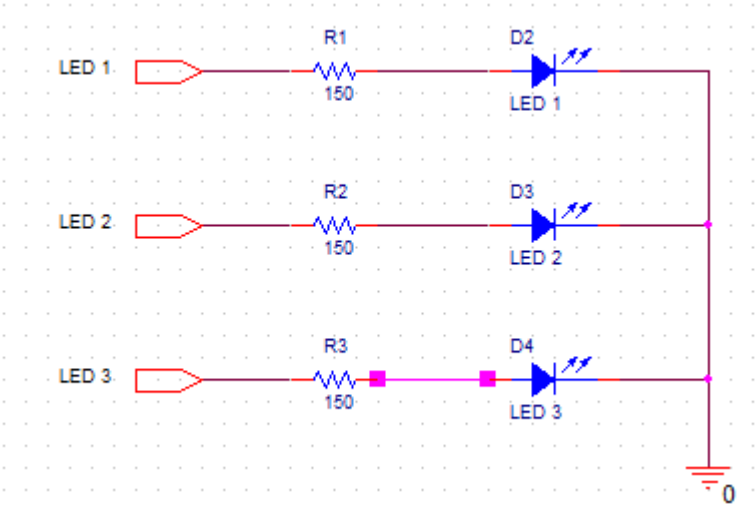

Figure A.2 LED Schematic

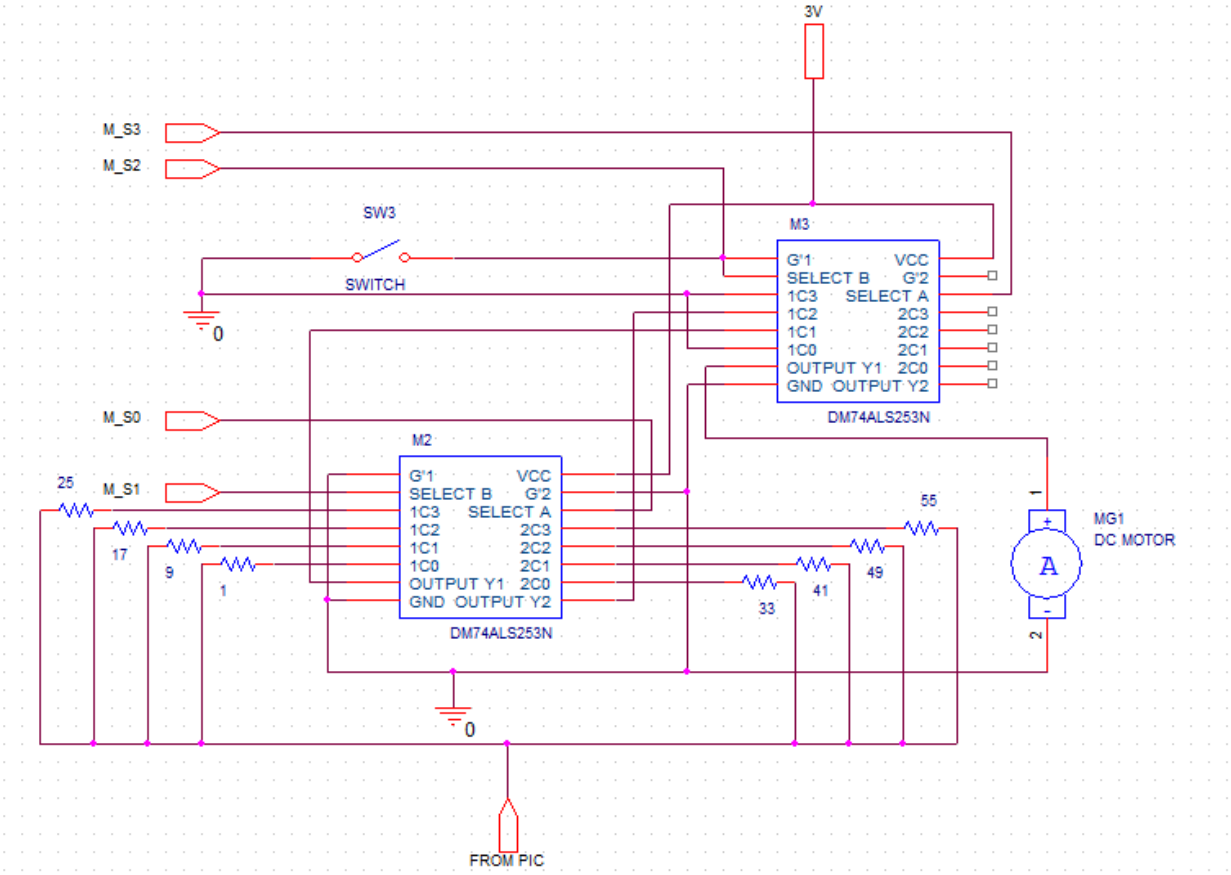

Figure A.3 Motor Schematic

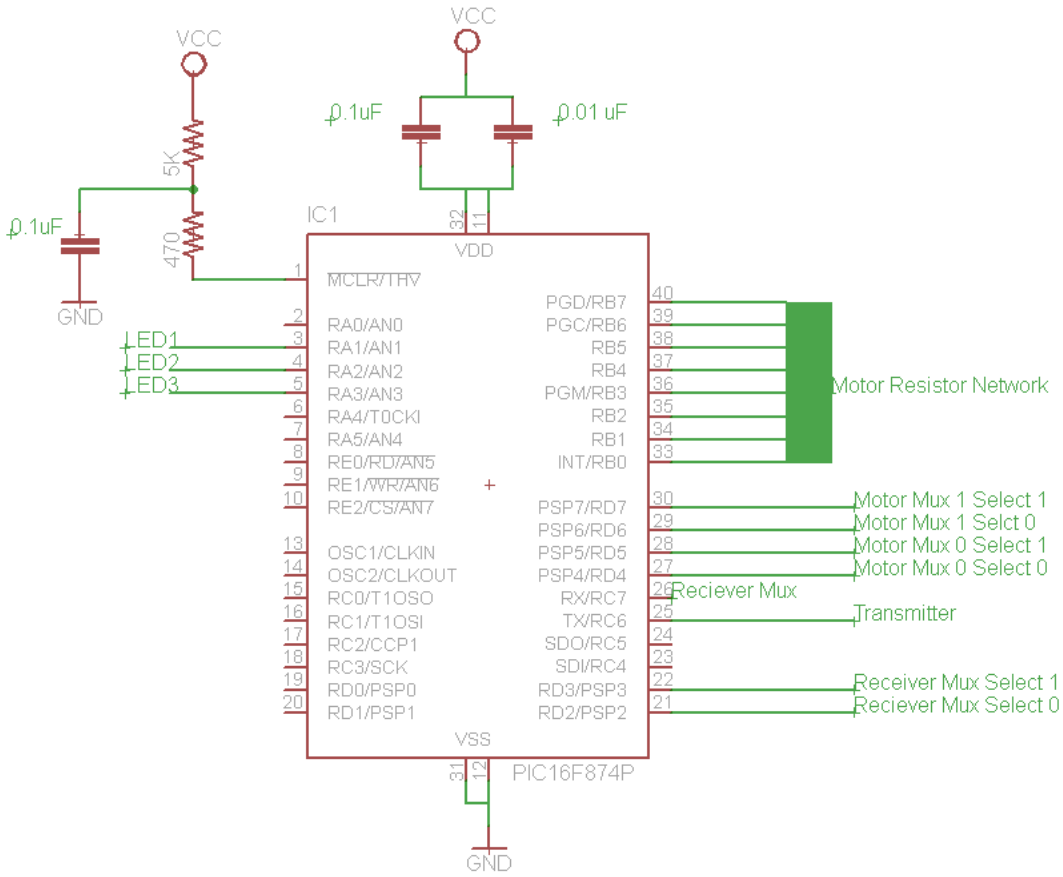

Figure A.4 Microcontroller Schematic

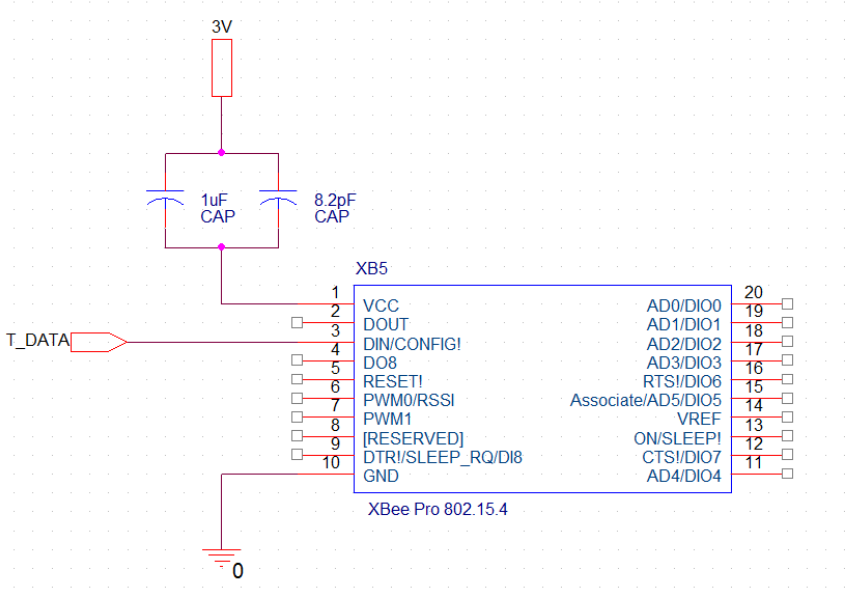

Figure A.5 Transmitter Schematic

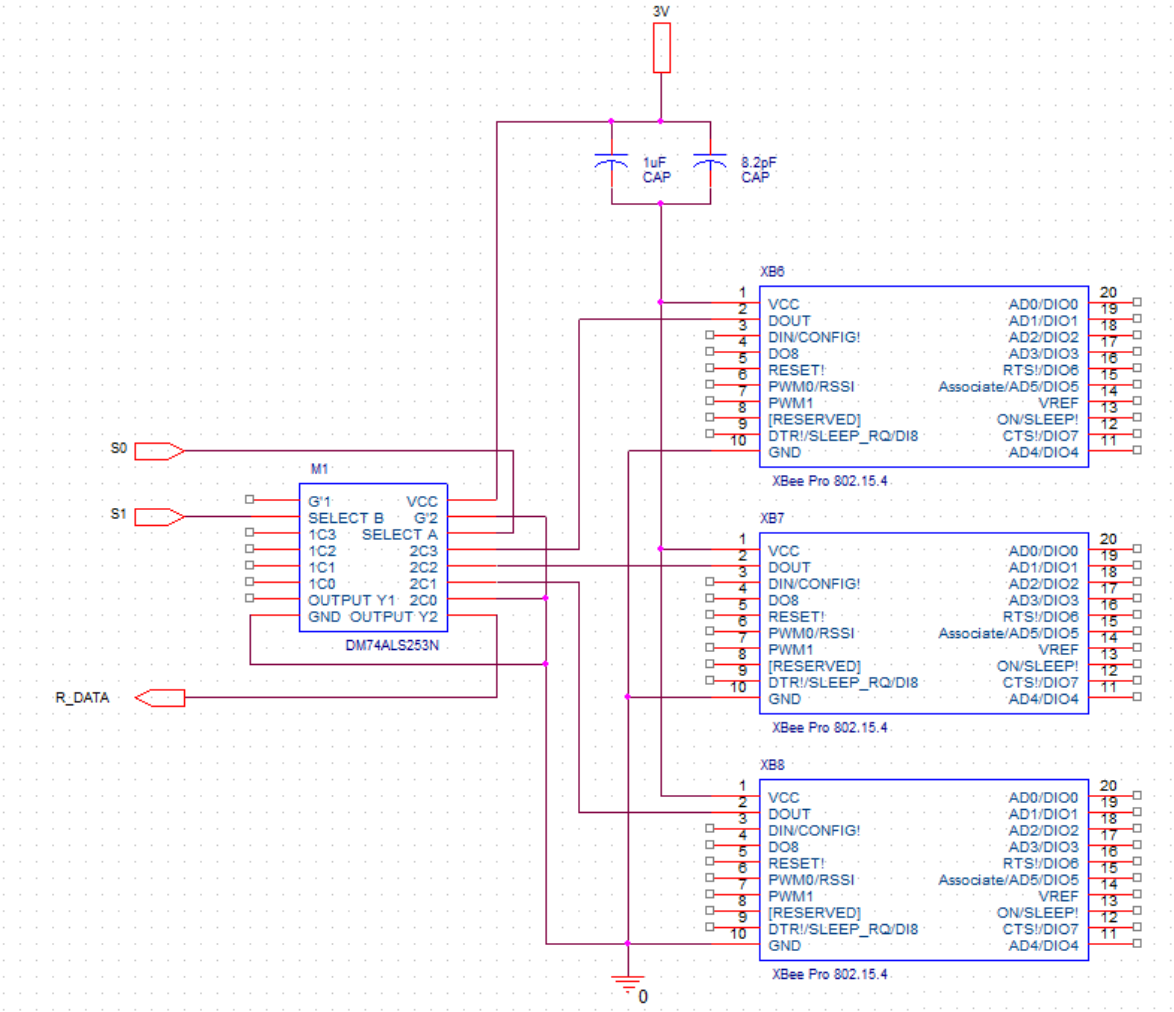

Figure A.6 Receiver Schematic

Appendix B: Microcontroller Flowchart and Code

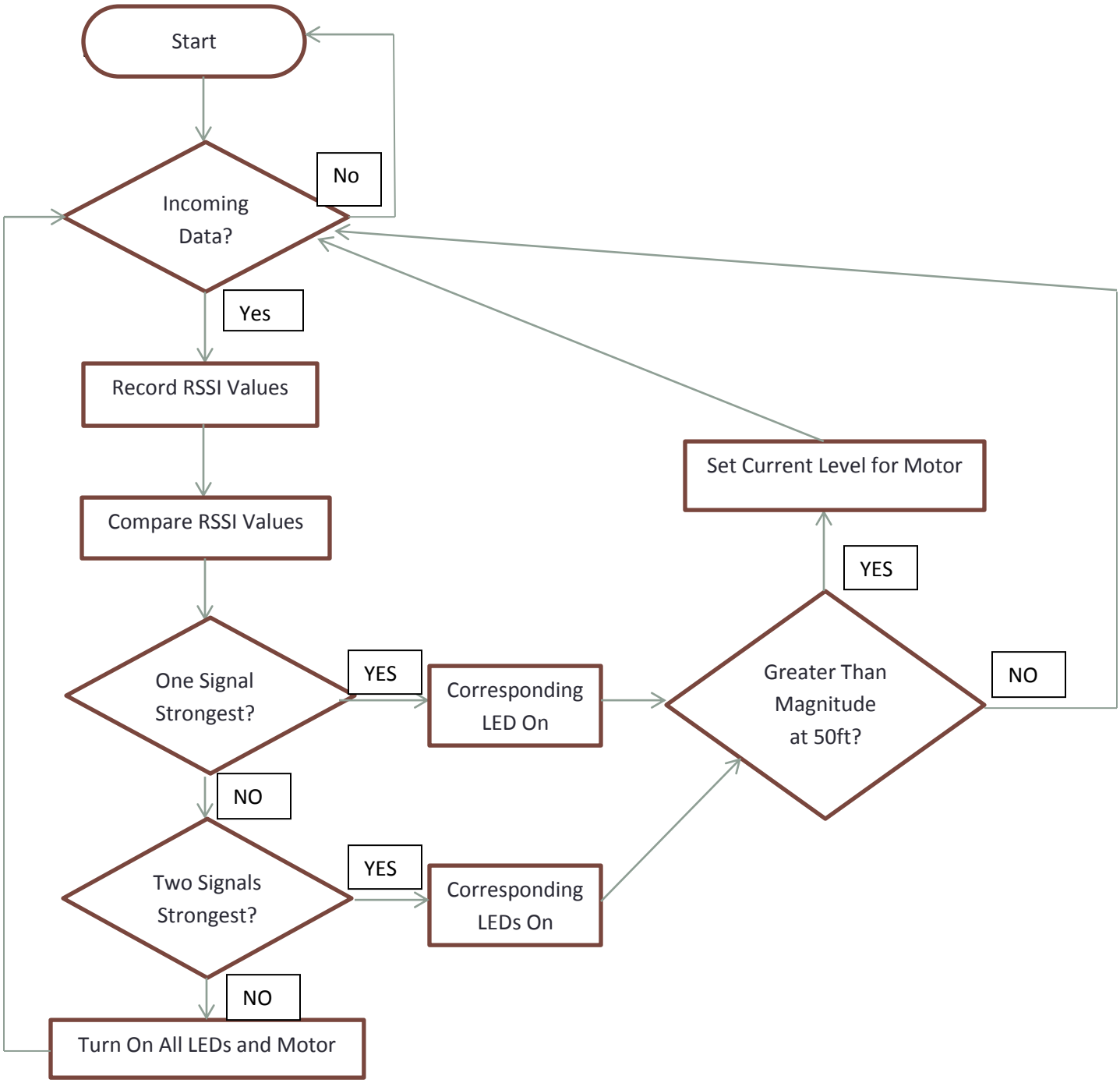

Figure B.1 Microcontroller flowchart

B.2 Code

```
#include <16F887.h>
#include <stdlib.h>
#use delay(clock=4000000)
#use RS232(uart1, baud=9600)
void main() {
//variable declerations
        char strength1;
        char strength2;
        char strength3;
        char max;
        int checkrecieved1=0;
        int checkrecieved2=0;
        int checkrecieved3=0;
        char x,y,z;
        set_tris_b(0x00);
        output_b(0);
        set_tris_a(0xF1);
        output_a(0);
        set_tris_d(0x03);
        output_d(0);
        //check for data
        while (true) {
        output_high(PIN_B7); //set all outputs for motor resistive network high
        output_high(PIN_B6);
        output_high(PIN_B5);
        output_high(PIN_B4);
        output_high(PIN_B3);
        output_high(PIN_B2);
        output_high(PIN_B1);
        output_high(PIN_B0);
        putc('a');
//check for new data
        if (checkrecieved1 == 0) {
                output_high(PIN_D2);  //mux selects reciever 1
                output_low(PIN_D3);
                if (kbhit()) {   //when data is ready store in x
                        x=getc();
                        if (x== 'V') {  //once we are at V we know the next character is x then the RSSI value after
                                while (!kbhit()) {
                                }
                                z=getc();
                                if (z=='x') {
                                while (!kbhit()) {
                                }
                                y=getc();
                                strength1=y;
                                checkrecieved1=1;
                                }
                        }
                }
        }
        //just recieved strength of reciever 1, change mux bits to get reciever 2
        if ((checkrecieved1 == 1) && (checkrecieved2 ==0)) {
                output_low(PIN_D2);
                output_high(PIN_D3);  //mux selects reciever 2
                if (kbhit()) {
```

```
                        x=getc();
                        if (x== 'V') {
                                while (!kbhit()) {
                                }
                                z=getc();
                                if (z=='x') {
                                while (!kbhit()) {
                                }
                                y=getc();
                                strength2=y;
                                checkrecieved2=1;

                                }
                        }
                }
        }
        //got strength2, change mux bits to reciever 3
        if ((checkrecieved2 == 1) && (checkrecieved3 == 0)) {
                output_high(PIN_D2);
                output_high(PIN_D3);  //mux selects reciever 3
                if (kbhit()) {
                        x=getc();
                        if (x== 'V') {
                                while (!kbhit()) {
                                }
                                z=getc();
                                if (z=='x') {
                                while (!kbhit()) {
                                }
                                y=getc();
                                strength3=y;
                                checkrecieved3=1;
                                }
                        }
                }
        }
        //compare strengths and set mux bits to control motor current
if ((checkrecieved1 ==1) && (checkrecieved2 == 1) && (checkrecieved3 == 1)) {
        if (strength1 == strength2) {
                if (strength1 < strength3) { //signal1 and 2 strongest
                        output_high(PIN_A1);
                        output_high(PIN_A2);
                        output_low(PIN_A3);
                        max = strength1;
                }
                 else {                                                 //signal 3 strongest
                        output_high(PIN_A3);                            //turn on LED 3
                        output_low(PIN_A2);
                        output_low(PIN_A1);
                        max = strength3;
                }
         }
         else if ( strength2 == strength3) {
                if (strength2 < strength1) {//signals 2 and 3 strongest
                         output_high(PIN_A3);
                         output_high(PIN_A2);                                   //turn on LED2/3
                        output_low(PIN_A1);
                         max = strength2;
```

```
            }
            else {                                                          //signal 1 strongest
                        output_high(PIN_A1);    //LED 1
                        output_low(PIN_A2);
                        output_low(PIN_A3);
                        max = strength1;
            }
}
 else if (strength1 == strength3) {
            if (strength1 < strength2) { //signals 1 and 3 strongest
                        output_high(PIN_A3);
                        output_high(PIN_A1);                                //LED 1/3
                        output_low(PIN_A2);
                        max = strength1;
            }
            else {                                                  //signal 2 strongest
                        output_high(PIN_A2);          //LED 2
                        output_low(PIN_A3);
                        output_low(PIN_A1);
                        max = strength2;
            }
}
else if ((strength1 < strength2) && (strength1 < strength3)) { //signal 1 strongest
            output_high(PIN_A1); //turn on LED 1
            output_low(PIN_A3);
            output_low(PIN_A2);
            max = strength1;
}
else if ((strength2 < strength1) && (strength2 < strength3)) { //signal 2 strongest
            output_high(PIN_A2); //turn on LED 2\
            output_low(PIN_A3);
            output_low(PIN_A1);
            max = strength2;
}
else if ((strength3 < strength1) && (strength3 < strength2)) {//signal 3 strongest
            output_high(PIN_A3); //turn on LED 3
            output_low(PIN_A2);
            output_low(PIN_A1);
            max = strength3;
}
else {
            output_high(PIN_A3);  //turn all on if all equal
            output_high(PIN_A2);
            output_high(PIN_A1);
            max = strength1;
}
//compare stored max signal to reference signals
if (max <= '(') {
            output_high(PIN_D4); //select bit 0 of first mux
            output_high(PIN_D5); //sets S0 and S1 of first mux to 11
            output_low(PIN_D6);  //
            output_high(PIN_D7); //sets S0 and S1 of second mux to 10
}
else if (max <= '-') {
            output_low(PIN_D4); //select bit 0 of first mux
            output_high(PIN_D5); //sets S0 and S1 of first mux to 10
            output_low(PIN_D6);  //
            output_high(PIN_D7); //sets S0 and S1 of second mux to 10
```

```
	}
	else if (max <= '2') {
		output_high(PIN_D4); //select bit 0 of first mux
		output_low(PIN_D5); //sets S0 and S1 of first mux to 01
		output_low(PIN_D6);  //
		output_high(PIN_D7); //sets S0 and S1 of second mux to 10
	}
	else if (max <= '7') {
		output_low(PIN_D4); //select bit 0 of first mux
		output_low(PIN_D5); //sets S0 and S1 of first mux to 00
		output_low(PIN_D6);  //
		output_high(PIN_D7); //sets S0 and S1 of second mux to 10
	}
	else if (max <= '<') {
		output_high(PIN_D4); //select bit 0 of first mux
		output_high(PIN_D5); //sets S0 and S1 of first mux to 11
		output_high(PIN_D6);  //
		output_low(PIN_D7); //sets S0 and S1 of second mux to 01
	}
	else if (max <= 'A') {
		output_low(PIN_D4); //select bit 0 of first mux
		output_high(PIN_D5); //sets S0 and S1 of first mux to 10
		output_high(PIN_D6);  //
		output_low(PIN_D7); //sets S0 and S1 of second mux to 01
	}
	else if (max <= 'F') {
		output_high(PIN_D4); //select bit 0 of first mux
		output_low(PIN_D5); //sets S0 and S1 of first mux to 01
		output_high(PIN_D6);  //
		output_low(PIN_D7); //sets S0 and S1 of second mux to 01
	}
	else if (max <= 'K') {
		output_low(PIN_D4); //select bit 0 of first mux
		output_low(PIN_D5); //sets S0 and S1 of first mux to 00
		output_high(PIN_D6);  //
		output_low(PIN_D7); //sets S0 and S1 of second mux to 01
	}
	//no matter what reset checkrecieved variables so we dont change outputs without new data
	checkrecieved1=0;
	checkrecieved2=0;
	checkrecieved3=0;
}

	}
}
```

Appendix C: Requirements and Verifications Table and Cost Tables

Table C.1 Requirements and Verification Table

Requirement	Verification	Verified?
Microcontroller		Yes
1. Microcontroller detects differences between incoming signals and sends signals to LEDs. a. Microcontroller compares stored signals to calculate which are greatest and sends correct signals to LEDs.	1. Three different signal strengths will be programmed into the microcontroller. The microcontroller will compare the values and light up the correct LED. a. Correct LED signal is above 2.5V based on highest input from programmed values.	Yes
2. Microcontroller gets RSSI signals from all three antennas.	2. Two armbands will be brought close together so all signal strengths are the same. Each LED corresponds to 1 RSSI signal from a receiver. If all LEDs turn on (the voltage is greater than 2.5V) and the test has passed.	Yes
3. Microcontroller correctly activates and sends correct current control signals. a. Microcontroller finds strongest signal and stores magnitude. b. Microcontroller selects appropriate mux line for the given distance.	3. The microcontroller will be programmed with three signals. The comparing code and setting the magnitude of vibration code will be run. An oscilloscope will be used to check the mux selects for the motor resistor network. a. Have microcontroller turn on the LED which corresponds to the strongest value. b. Measure the select lines on the motor mux with the oscilloscope. The values should be set according to the coded signals.	Yes
Transceiver		
4. Accuracy of directionality of 60 +/- 30 a. Antenna's receive signals only within 180 degrees	3. One receiver will be rotated, while the transmitter will remain in place. The output signal strength will be measured on an oscilloscope.	No

	a. Output signal is at least 20dB higher in the 180 degrees in front of the antenna.	
4. Transmitter sends signal a. Transmitter able to receive serial data from microcontroller b. Transmitter sends serial data Omni directionally	4. The transmitter will receive data from the microcontroller. The data out pin on the receiver will be hooked up to an oscilloscope to check for correct transmission. The receiver will be moved in all directions. a. Oscilloscope has same data as coded. b. Oscilloscope reads a signal greater than 2.5V in all directions.	Yes
5. Receiver receives signal up to 100ft indoors and 200ft outdoors.	5. Reciever receives signal and LED(s) turn(s) on	Yes
User Interface		
6. Motor vibrates more or less intensely a. Motor vibrates more intensely with larger current control signal	6. An multimeter will be used to read the output current from the motor driver. a. As current control bits are incremented, the current at the motor will increase.	Yes
7. LEDs illuminate when active.	7. LEDs will be connected to an outside power supply and resistive network. When 3V (20mA) is supplied to LEDs they will light up.	Yes
Power Supply		
8. Power supply gives correct voltages at outputs.	8. Power supply gives between 2.5 and 3.5 volts.	Yes
Entire Project		
9. Armband lasts at least 5 hours	9. Armbands will be left with all the LEDs on and the motor vibrating for 5 hours. After	Yes

	five hours armbands will be checked.	
10. Microcontroller compares magnitude of stored RSSI signal with that of stored reference signal from 50 ft, sets the correct current at the motor, and illuminates the LEDs.	10. Measure the current at the motor. If it changes for a close and far distance (3ft and 30ft) the test has passed.	Yes

Table C.2 Parts Cost Table

Part	**Quantity**	**Price**	**Total Cost**
PIC 16F887a	2	$2.80	$5.60
Xbee-Pro 802.15.4	8	$38.00	$304.00
Pico Vibe 12mm Vibration Motor – 3.4mm type	2	$6.68	$13.36
LED-117 Ultra Bright Blue	6	$.82	$4.92
Energizer LA522 4 pack	1	$7.50	$7.50
4 Position DIP Switch	2	$1.66	$3.32
TI SN54AC08	2	$2.20	$4.40
74ALS153N	6	$2.14	$12.84
Capacitors	14	$0.20	$2.80
Resistors	26	$0.20	$5.20

Total Cost $363.94

Table C.3 Labor Cost Table

Name	**Rate**	**Hours**	**Total**	**Total * 2.5**
Tim Capota	$40/hr	120	$4,800	$12,000
Dave Drake	$40/hr	120	$4,800	$12,000

Total Labor Cost: $24,000

Total Cost: $24,363.94