AUTOMATIC VOLUME CONTROL

By

Chris Goulet

Eric Davila

Roland Legrand

Final Report for ECE 445, Senior Design, Spring 2013

TA: Igor Federov

May 2013

Project No. 23

Abstract

This paper describes the methods taken to design, build, and test an automatic volume control system. The system automatically adjusts the volume of multiple sets of speakers based on the distance that a user is from them so that the speakers are louder when the user is further away and quieter when the user is closer. This project was done for ECE 445: Senior Design at the University of Illinois in the spring semester of 2013.

Contents

1. Introduction 1
   1.1 Purpose and Benefits 1
   1.2 Objectives and Function 1
   1.3 Block Diagrams 2
      1.3.1 Central Hub 2
      1.3.2 Speaker System 4
      1.3.3 Transmitter 5
2 Design 6
   2.1 Speakers 6
   2.2 Digital Potentiometers 7
   2.3 Logic Unit 8
   2.4 Microcontroller 10
   2.5 Bluetooth Sensors 13
   2.6 Transmitter 14
   2.7 Power Supply 16
3. Design Verification 17
   3.1 Speaker 17
   3.2 Logic Unit 17
   3.3 Microcontroller 19
   3.4 Transmitter 20
4. Costs 21
   4.1 Parts 21
   4.2 Labor 21
   4.3 Grand Total 22
5. Conclusion 23
   5.1 Achievements 23
   5.2 Ethical considerations 23
   5.3 Safety 24
   5.4 Future work 24

5.5 Special Thanks 25
Appendices 26
Appendix A: References 26
Appendix B: Requirements and Verification 27
B.1 Central Hub 27
B.2 Speaker System 29
B.3 Transmitter 33
Appendix C: Schematics and PCBs 34
C.1 Logic Unit 34
C.2 Bluetooth Sensor 36
Appendix D: Tables and Charts 38
D.1 RSSI to Distance Lookup Table 38
D.2 Bluetooth Configuration 39
Appendix E: Equations 41
Appendix F: Code 42
F.1 MSP430 Code for control unit/digital potentiometer 42
F.2 MSP430 Code for Bluetooth/RSSI 45

1. Introduction

1.1 Purpose and Benefits

A common problem that anyone who loves music runs into is when they are listening to a song on their stereo in one room, but they need to walk to another room for a moment. What do you do to avoid this problem? Some turn up their stereo really loud so that they can hear it from further away, and some grimace and hurry so that they only miss 20 seconds of the song. But what if there was a way to automatically adjust the volume of the stereo based on the distance that the listener is from it? That is exactly what our senior project addresses. With our Automatic Volume Control system, the volume of any sound system can be adjusted simply by the distance that the listener is from it so that no matter what distance they are from it, they hear the same volume level. This system will also be able to function with different speakers in different rooms, so that the closer speakers play the music as you move from one room to the other. By carrying an active device like a smart phone, the listener's position will always be known relative to the speakers in order for this system to function.

1.2 Objectives and Function

This will all be accomplished through a portable active device that the listener would carry, such as a smart phone. The phone will send information using Bluetooth technology to other Bluetooth sensors placed at the speakers so that the distance between the devices can be known. With that information, the volume level of each individual speaker would be adjusted, thanks to a logic circuit and microprocessor which adjusts the control of the amplifier circuit for each speaker.

Benefits

- Move within room while maintaining a constant volume level
- Transition from room to room with a constant volume level

1.2.4 Product Features

- Multiple volume controlled speakers that can be in different rooms
- Any phone that supports Bluetooth can be used with this system

1.3 Block Diagrams

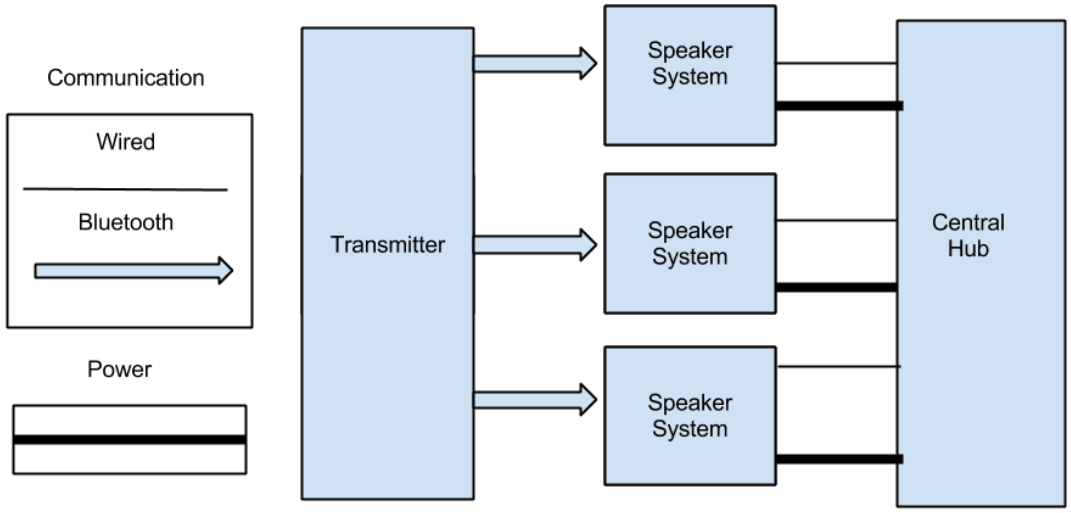


Figure 1

1.3.1 Central Hub

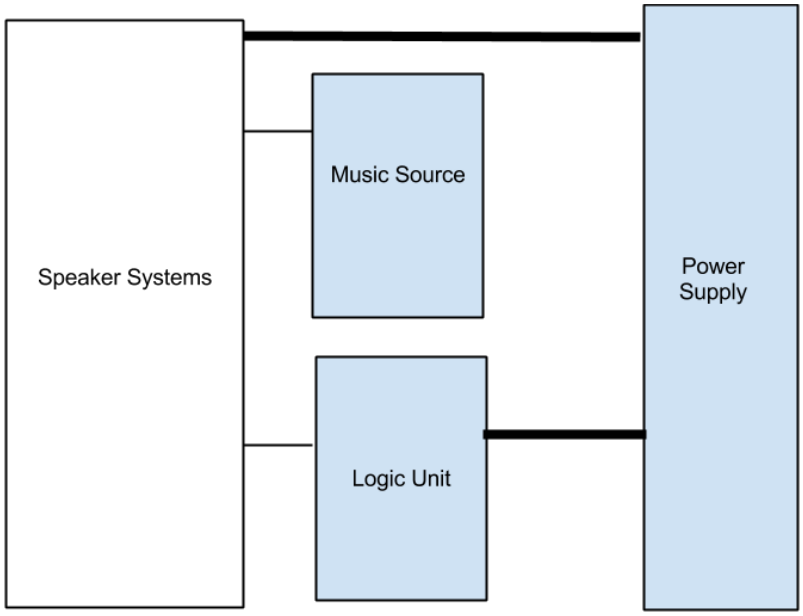


Figure 2

Overall Summary:

The purpose of this component is to dock the music source (i.e. a mp3 player, a cd player, etc.), send the music to the speakers, and to decide which speaker will be on. It receives and sends information from the speaker system. The information it receives from each speaker is the distance the person is from that respective speaker, while the information it sends to each speaker is a binary signal telling each speaker whether it should be on or off. The subsystems of the central hub are the music source and the logic unit that sends out the on/off signals.

Case Design:

The Central Hub will consist of a music source and a logic unit. The music source, such as a portable laptop or stereo, will be powered by a battery charger. The portable logic unit will be powered by the music source as well.

Power Supply:

Input: The input to the power supply will be the 120V, 60Hz AC power supplied by a wall outlet.

Output: The following are the power needs for different devices in our system:

- 3.3V DC for the WT32 Bluetooth sensors
- 5V DC for inverters
- 5V DC for NAND gates
- 5V DC for comparators
- 5V DC @ 1A for digital to analog converter
- 3.3V for microcontroller

Therefore, the power supply will output 3.3V DC and 5V DC, which will cover all of the different components.

Description: The central hub will contain the main power supply, which will be fed to the logic unit in the central hub as well as to the *Speaker Systems*.

Music Source:

Input: The music source will have an attachment for a wall outlet if necessary. It will have music files to play.

Output: The music source will send the music file signal to the microcontroller.

Description: The music source will consist of a laptop or stereo that will take a sound file (MP3, .wav, ITunes, Windows media player) as the desired music to be played and will then send this information to the speakers via audio cables.

Logic Unit:

Input: One analog voltage signal transmitted from each speaker. Each voltage signal will be inversely proportional to the distance between the speaker sending the signal and the user.

Output: One discrete, binary signal to each speaker. They correspond to an on/off signal, and at anytime only one of these signals will correspond to an “on”. This will ensure that only one speaker will be on at a time.

Description: The logic unit is the subsystem of the central hub that will send an on/off signal to each speaker. The input of the logic unit will be analog voltage signals from each speaker system that are inversely proportional to the distance, which means that the speaker closest to the user will input the highest voltage to the logic unit. To generate the on/off signals that will be sent to the speakers, a decoder circuit will be used. To generate a selection signal that the decoder can use additional comparators and TTL logic will also be added to the logic unit.

1.3.2 Speaker System

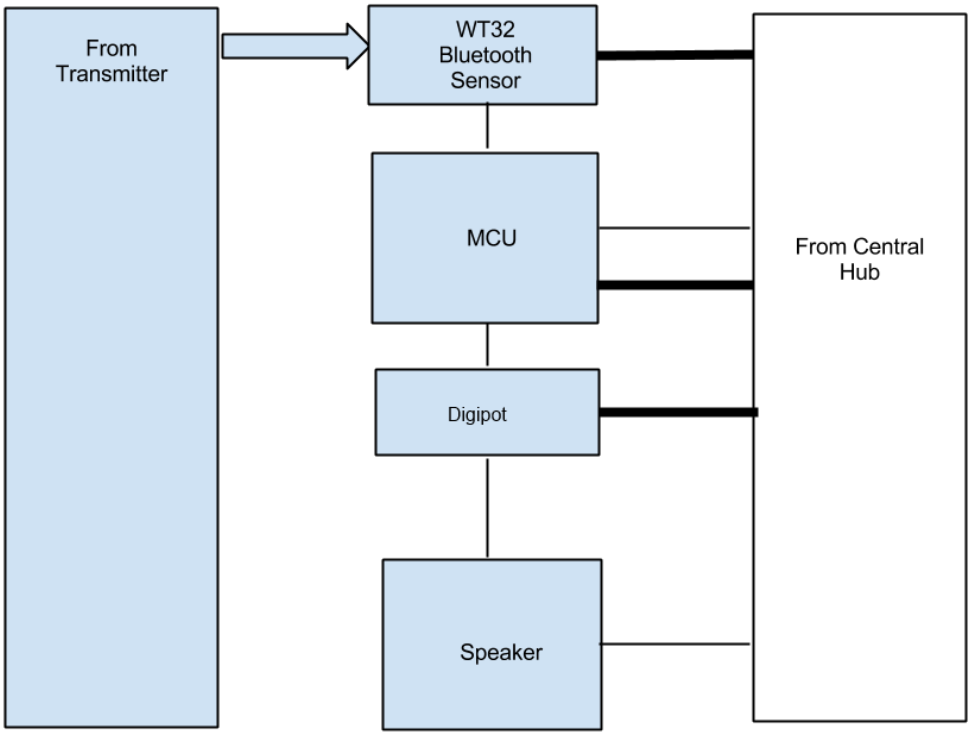


Figure 3

Overall Summary:

The purpose of this component is to play the music at a normal volume. It is composed of four parts: WT32 Bluetooth Sensor, MCU, Amplifier, and Speaker.

Case Design:

This component will consist of a case to be set on top of the speakers. Inside the case will be the Bluetooth sensor, MCU, and digital potentiometer. The case and speakers are powered by cables coming from the *Central Hub* unit. The case will be connected to the speakers such that it will tell the speakers what to play and the volume to be played at.

Microcontroller Unit:

Input: Data received from the WT32 Bluetooth sensor via UART communication. The data received corresponds to RSSI (Received Signal Strength Indicator), which will correspond to the distance the user is from the stereo. The MCU will also receive a binary signal from the Central Hub, which corresponds to whether or not the speaker should be on.

Output: A digital signal that will be sent to a digital-to-analog converter, then amplified so that it can be used by the speaker. This signal will control the volume of the speaker.

Description: The microcontroller unit is what controls the volume of the speaker. Since the objective of the system is to provide constant volume across multiple rooms, the speaker should raise the volume as the user moves away from it. To tell how far away the user is from the microcontroller, the microcontroller will extract the RSSI value from the WT32 Bluetooth sensor via UART communication.

The RSSI (Received Signal Strength indicator) value is a ratio of the power in the received wave signal over the power in the transmitted wave signal. Its units are dB (decibel) with a lower RSSI value

meaning a stronger incoming signal and thus closer in proximity. In our graphs and tables below the absolute value of the RSSI will be displayed since the WT32 will be returning RSSI values within the range -128dB to 20dB, although in practice the range is -100dB to - 28dB.

WT32 Bluetooth Sensor
Input: RSSI data received from the Transmitter
Output: UART communication to the MCU.
Description:

The Bluetooth Sensor design is the Bluegiga#WT32-A-AI4 and is manufactured by Bluegiga Technologies. The Bluetooth sensor is composed of a Bluetooth Audio Module with Internal Antenna and iWRAP 4.0 firmware. The Bluetooth sensor is inside the Speaker System. It will connect directly to the Microcontroller unit (MCU) via the Universal Asynchronous Receiver Transmitter (UART) interface. There are two pins that are connected to the MCU, the UART_RXD (pin 15) and UART_TXD (pin 16).The UART_RXD is a CMOS input with a weak internal pull down, it receives data from the MCU. The UART_TXD is a CMOS input with a weak internal pull up, it sends UART data to the MCU. The MCU can send commands to the Bluetooth using the iWARP 4.0 firmware to receive the RSSI value from the transmitter. The Bluetooth Sensor is powered by the Power supply coming from the Central Hub. Its purpose is to communicate with the transmitter to receive an Received Signal Strength (RSSI) value. The RSSI value will then be sent to the MCU to correlate that value into a distance unit. The RSSI values will range from -128 to 20. We will empirically determine what RSSI values correspond to what distances. See MCU section for more details.

The WT32 Bluetooth sensor will communicate with the transmitter via an antenna using the Simple Secure Path (SSP) protocol and the Hands Free Protocol (HFP). The messaging will be done with iwrap4, an embedded firmware running on the RISC processor. Iwrap4 will be interfaced with Universal Asynchronous Receiver Transmitter (UART) using ASCII commands that Iwrap4 supports.

Digital Potentiometer:

The digital potentiometer receives the signal from the microcontroller and steps up the voltage of the signal at a constant rate before the speaker receives it.

Speakers:

The speakers receive the music signal from the amplifier and convert the electrical signal to sound.

1.3.3 Transmitter

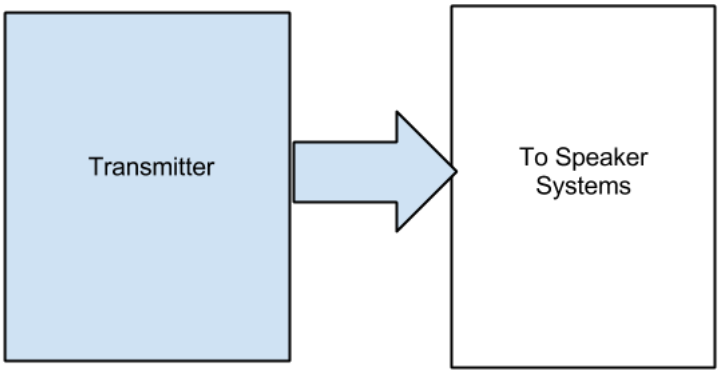


Figure 4

Overall Summary:

The purpose of this component is to identify where the person, who is hearing the audio from the speakers, is in the room in relation to the speaker locations. It will communicate with the other WT32 Bluetooth sensors via Bluetooth using SSP, described in the Bluetooth sensor section of the speaker system.

Case Design:

The Transmitter is a Galaxy S II, Model SPH-D710, Hardware Version D710.10, Android Version 4.0.4. with Bluetooth v3.0 + HS.

Transmitter:

Input: None
Output: RSSI value to WT32 Bluetooth unit
Description:

The Transmitter will be using Bluetooth to communicate with the other WT32 Bluetooth Sensors on the Speaker Systems and Central Hub. The Speaker Systems will request pairing as well as the RSSI value of the transmitter using the messaging protocols found in the *Speaker Systems* section.

2 Design

2.1 Speakers

The speakers used for the system were all the same, GE 2.0 Multimedia speakers. Since they were the same, they all had the same volume range. Another reason to use these speakers is that the mechanical potentiometers already on them were 10 ohms, the same resistance value as the digital potentiometers that were purchased. These speakers were kept mostly intact except for removing the mechanical potentiometers from the speaker circuits so that the digital potentiometer connections could be inserted where they were supposed to go. The speakers had their own power supply already. The original music source input to the speakers was not used because the music would be going through the digital potentiometer instead of through the place where the mechanical potentiometer used to be.

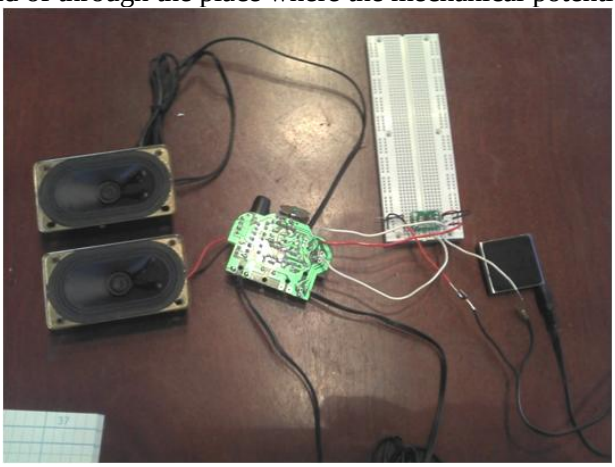


Figure 5

2.2 Digital Potentiometers

The digital potentiometer models used were X9116WS8IZ by Intersil. These were chosen because they have 10 ohms of resistance- a common value for speaker potentiometers- and were relatively cheap. These are easy to control with 8 different pins, as demonstrated in Table 1.

Table 1: Digital Potentiometer Pins

Pin:	Name:	Function:
1	Increment	Signals a wiper change.
2	Up / Down	Tells the wiper which way to move.
3	High Resistance	The high end of the potentiometer.
4	V_{SS}	Chip ground.
5	Wiper Resistance	The wiper end of the potentiometer.
6	Low Resistance	The low end of the potentiometer.
7	Chip Select	Select to store previous value.
8	V_{CC}	Input voltage for chip.

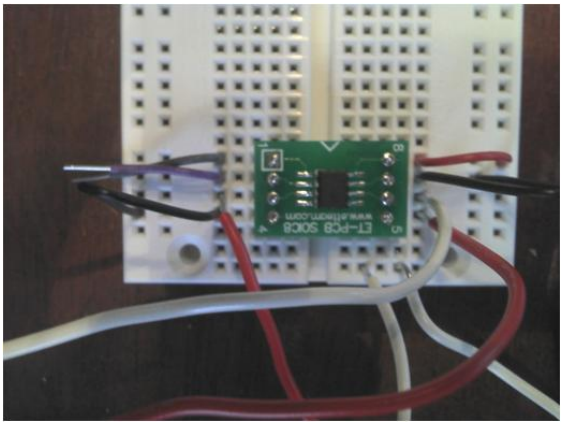


Figure 6: Digital Potentiometer on Breakout Board

2.3 Logic Unit

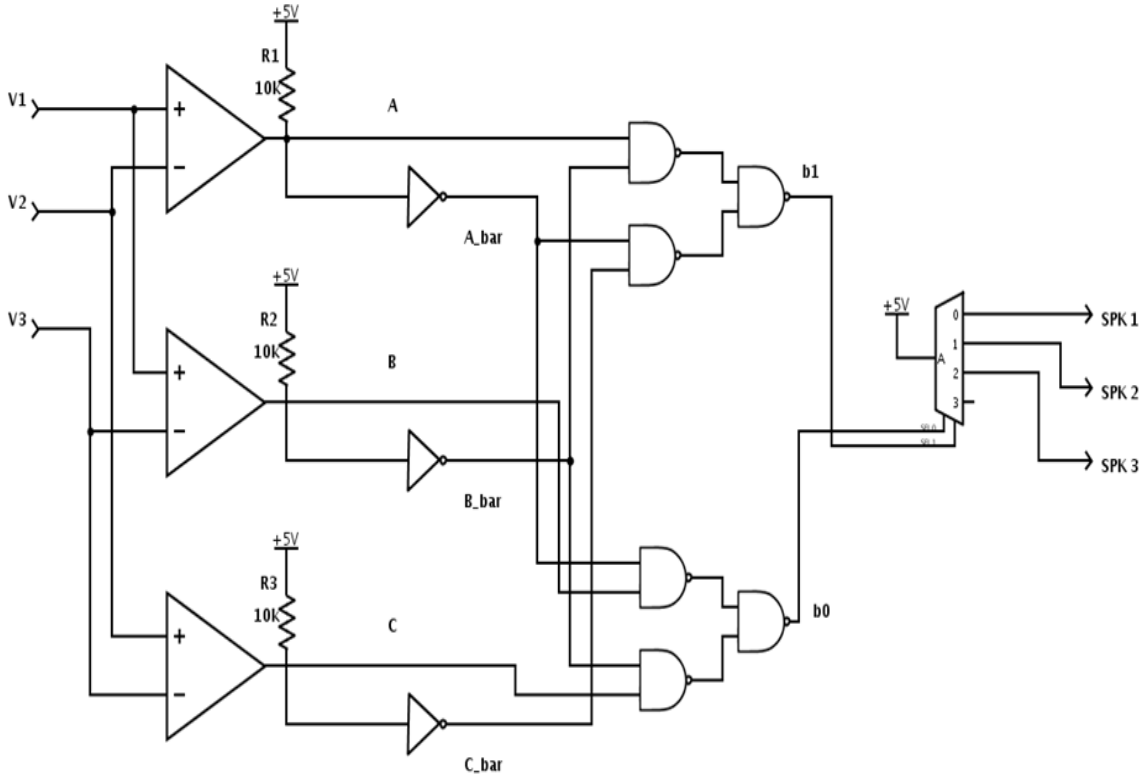


Figure 7

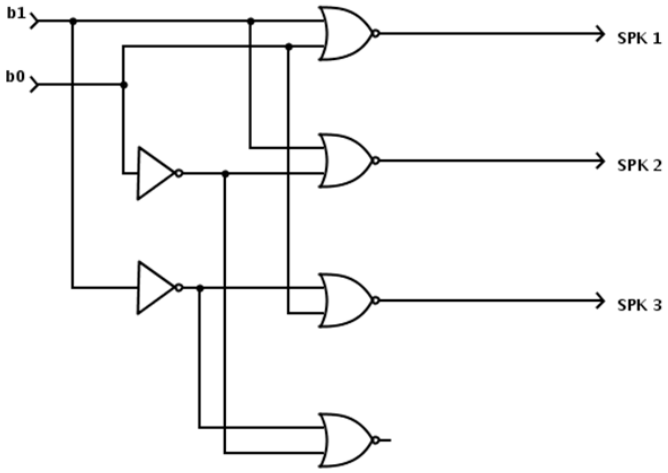


Figure 8

Figure 7 and 8 outline the circuitry that make up the Logic Unit. The Logic Unit is broken up into three blocks: the comparators, the TTL logic, and the decoder. The comparators take the analog signals that come from each of the speakers and outputs binary signals that the TTL logic can use. The job of the

TTL logic is to generate a selection signal that will then feed into the decoder. The decoder will simply output a binary signal to each of the speakers that will tell them whether they should be on or off.
To select the proper pull-up resistor to use with the comparators, voltage drop and power dissipation were taken into consideration. 4.7kΩ and 10kΩ resistors are commonly used as pull-up resistors, so utilizing Ohm's law and KCL, the following values were determined. It is important to note that the input current to the TTL logic is 0.1mA:

$$V_{drop} = 0.0001R$$

$$P_{diss} = \frac{{V_{high}}^2}{R} = \frac{5^2}{R}$$

	V_{drop} [V]	**P_{diss} [mW]**
4.7kΩ	0.47	5.32
10kΩ	1	2.5

The voltage drop across the 10kΩ resistor is not big enough to alter the input state for the TTL logic, and it has half the power dissipation, so the 10kΩ was selected.

To select the proper logic for the TTL logic, a K-map was utilized for the design. This is acceptable since there are not that many inputs and outputs. The truth table for the TTL logic is show below:

Table 2

A (ON if $V_1 > V_2$)	**B (ON if $V_1 > V_3$)**	**C (ON if $V_2 > V_3$)**	**Speaker On**	**Select Signal**
0	0	0	3	10
0	0	1	2	01
0	1	0	X	11
0	1	1	2	01
1	0	0	3	10
1	0	1	X	11
1	1	0	1	00
1	1	1	1	00

From the truth table above and utilizing K-maps, the following logical expressions were derived below;

$$b_0 = B'C + A'B$$

$$b_1 = AB' + A'C'$$

The TTL logic assembly was derived from the logical expressions above.

To design the decoder, a similar approach to the design of the TTL logic was taken. The truth table for the decoder is shown below:

Table 3

b_1	b_0	Output
0	0	1000
0	1	0100
1	0	0010
1	1	0001

From the truth table and utilizing K-maps, we were able to derive the expression for the decoder below:

$$OUT_3 = b_1' * b_0'$$

$$OUT_2 = b_1' * b_0$$

$$OUT_1 = b_1 * b_0'$$

$$OUT_0 = b_1 * b_0$$

2.4 Microcontroller

Mapping the RSSI value to a predetermined relationship, a distance value can be determined. To come up with this relationship we empirically recorded the RSSI values from 0.1m to 8.5m. Our empirical results showed lots of scattering, however by taking the average of the RSSI values we were able to determine a logarithmic relationship.

$$\text{Distance} = e^{\frac{|\text{RSSI}|-67.486}{14.999}}$$

From this equation we created a lookup table and stored these values in our MCU for faster processing time.

We then used a store-10 algorithm that averages the most recent 10 RSSI values and then uses the lookup table to determine the distance. Furthermore, by averaging the RSSI values in a short time window, we found that the averaged RSSI values more closely follow our empirically determined

logarithmic relationship.

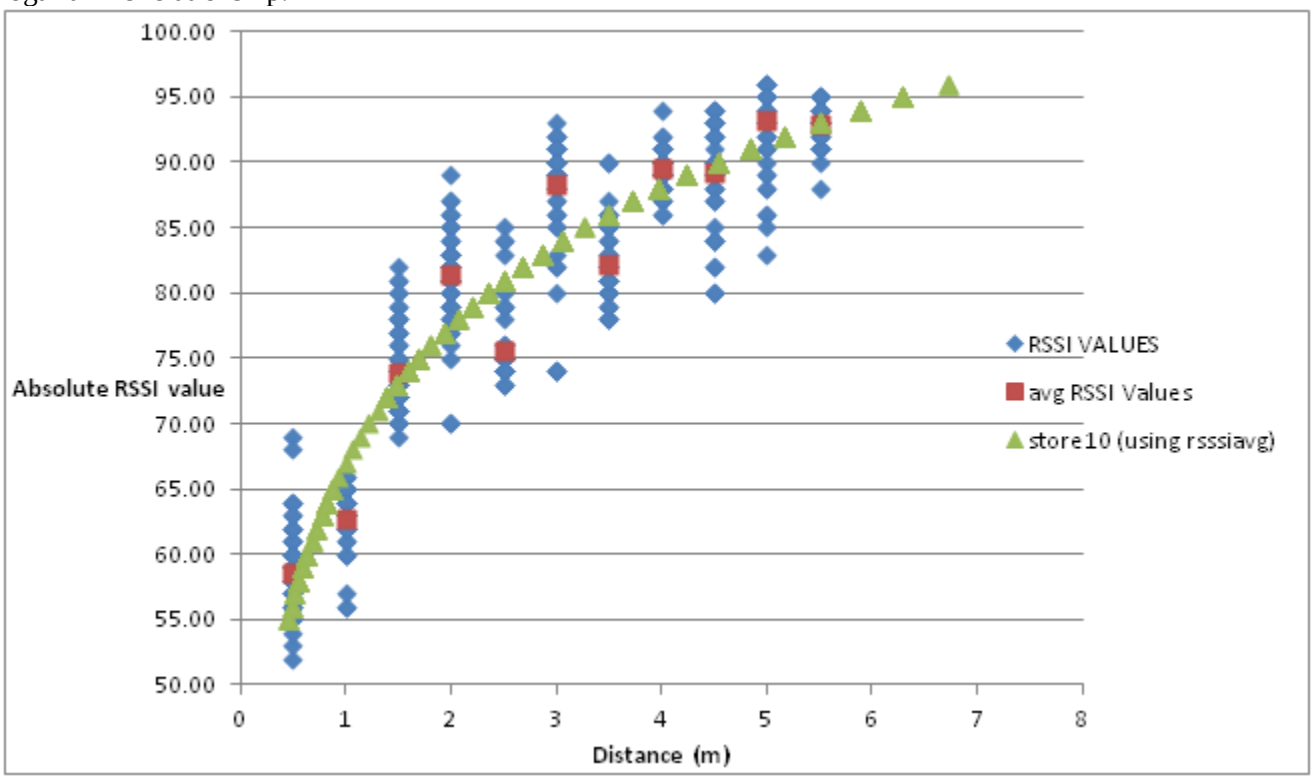


Figure 9

This distance value is valuable since the sound pressure (which determines volume), is related to distance by the distance law:

$$p \propto \frac{1}{r}$$

$$\frac{p_2}{p_1} = \frac{r_1}{r_2}$$

$$p_2 = p_1 \frac{r_1}{r_2}$$

So if the user increases his or her distance by a factor of K, the sound pressure at the sound source (i.e. the speakers) must increase by the same factor K in order to keep the sound pressure the same from the perspective of the user.

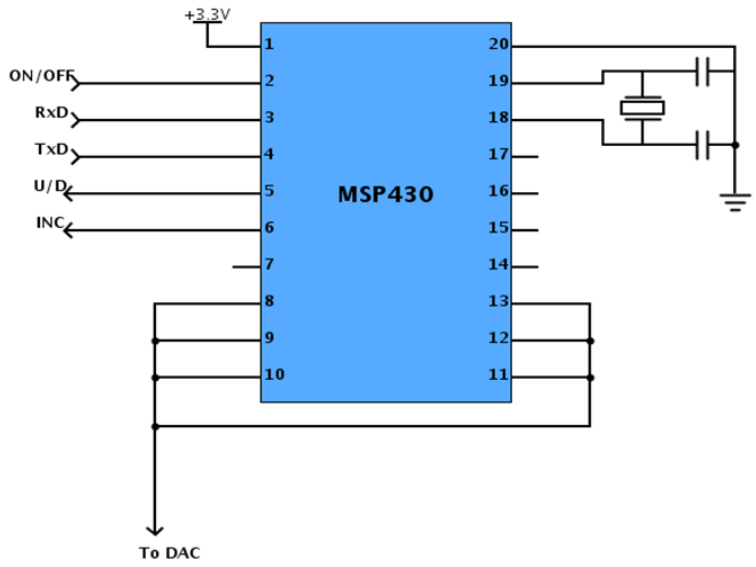


Figure 10

Figure 10 shows the pin out for the MCU. For this project, we utilized an MSP430G2553 microcontroller. There are several inputs and outputs that the MSP430 will interact with. The inputs to the MSP430 are: ON/OFF, RxD, and TxD. The ON/OFF input tells whether or not the speaker should be producing any sound. This signal is produced by the logic unit. The RxD and TxD inputs are how the MSP430 utilizes the UART communication interface. The UART interface is how we connect the microcontroller to the Bluetooth sensor. Through this interface, we can obtain the RSSI values that we need to calculate the distance between the transmitter and the speaker. The outputs of the microcontroller are: U/D, INC, and the 6-bit DAC output. U/D represents the up/down signal that will control what direction the wiper of the digital potentiometer will take when it is made to move. INC, the increment signal, controls the timing of the wiper movement in the digital potentiometer. The digital potentiometer is negative-edge triggered, so whenever the INC signal is changed from a high state to a low state, the digital potentiometer will react by moving the wiper. The 6-bit DAC output will feed into a DAC that will in turn produce an analog signal that will be fed back into the logic unit of the Central Hub.

Baud Rate

To select the baud rate the UCBR and UCBRS buffers must be set. Looking at the list of commonly used baud rates. We selected the BRCLK frequency of 1Mhz with a baud rate of 9600 and UCBR of 109 and UCBRS of 2. with transmission errors of 0.7% and receiving errors of 0.8%

Table 15-4. Commonly Used Baud Rates, Settings, and Errors, UCOS16 = 0

BRCLK Frequency [Hz]	Baud Rate [Baud]	UCBRx	UCBRSx	UCBRFx	Maximum TX Error [%]		Maximum RX Error [%]	
32,768	1200	27	2	0	-2.8	1.4	-5.9	2.0
32,768	2400	13	6	0	-4.8	6.0	-9.7	8.3
32,768	4800	6	7	0	-12.1	5.7	-13.4	19.0
32,768	9600	3	3	0	-21.1	15.2	-44.3	21.3
1,048,576	9600	109	2	0	-0.2	0.7	-1.0	0.8

Figure 11 Possible Baud Rate Settings

2.5 Bluetooth Sensors

The design of the Bluetooth sensor consists of selecting the proper baud rate, connecting the hardware to a host a machine for configuration, and sending the actual messaging between the Bluetooth sensor and the host.

Selecting the Baud Rate

The baud rate is the number of distinct symbol changes made to the transmission per second. The number 0.004096 was empirically found and is used in the WT32 Bluetooth Sensor. The PSKEY UART ABUD RATE is the persist key to be used in UART. The baud rate for the WT32 Bluetooth Sensor is dictated by the equations below:

$$\text{Baud rate} = \frac{\text{PSKEYUARTABUDRATE}}{0.004096} = \frac{39}{0.004096} = 9521.484375$$

and thus the,

$$\text{Percent error} = \text{Percent error} = \frac{9600 - 9521.484375}{9521.484375} = 0.82\%$$

With 39 Persist store values x027 the Baud rate desired is 9600 with 0.82% error which fits inside the parameters of the WT32 described in the table below:

Parameter		Possible values
Baud rate	Minimum	1200 baud (≤2% Error)
		9600 baud (≤1% Error)
	Maximum	3.0Mbaud (≤1% Error)
Flow control		RTS/CTS, none
Parity		None, Odd, Even
Number of stop bits		1 or 2
Bits per channel		8

Figure 12: Possible Baud Rate Settings

Hardware Connection

Connecting the Bluetooth sensor to the MCU consists of crossing the RXD and TXD lines and selecting the correct baud rate from above. Connecting the Bluetooth sensor to a PC however, requires extra circuitry since a USB cable output is 5V and the Bluetooth sensor is 3.3V. To solve this power problem I used a voltage leveler circuit consisting of 4 Resistances 1.477K 3.968K 4.61K 21.95K all ¼ watt power and 2 NPN Transistors 2N2222.

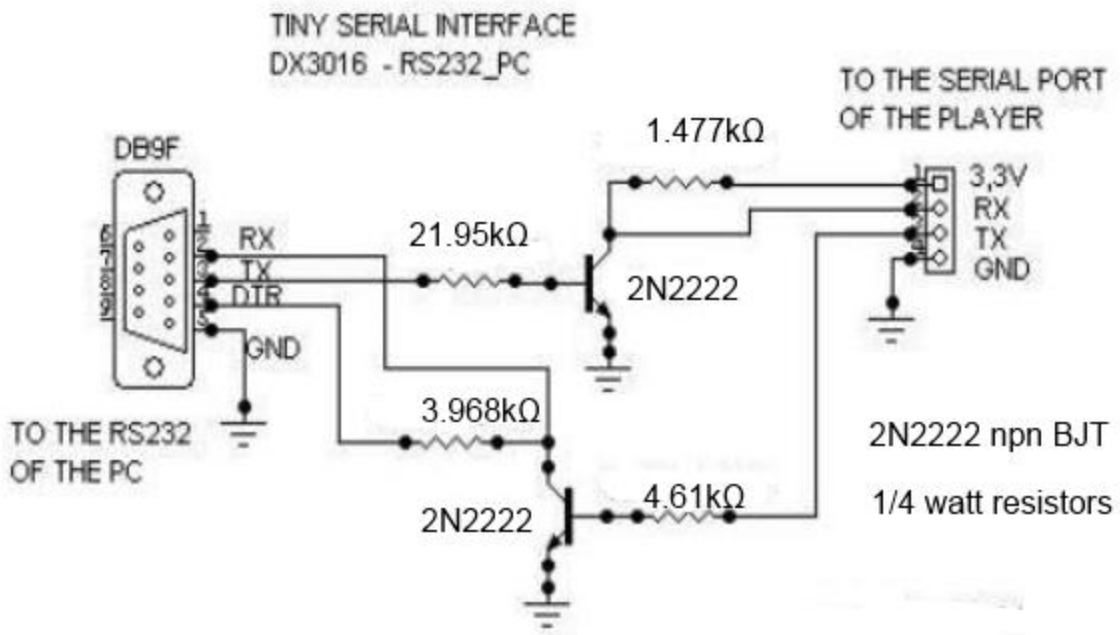


Figure 13 Level Converter

This solution was found using data mining from the following website referenced in the Appendix (Serial Console).

Configuring the WT32 Bluetooth Module

To set up a HFP the following code must be implemented using iwrap4 syntax see Appendix D.2 Bluetooth Configuration for table containing command and response syntax. This configuration could be done with the MSP430 acting as host, but we choose to configure with the PC using the BGTerminal found from Blugiga's support website referenced in the Appendix (BGTerm) because of the use of visual programming.

Modularity

This component is inside the speaker system along with the MCU and speaker. Its entire purpose is to receive signals from the transmitter that indicate the RSSI value and then send it to the MCU for RSSI to distance mapping. The Bluetooth sensor is powered by the power supply in the central hub.

2.6 Transmitter

The transmitter design is the listener's smart phone. Although many different smart phones can be used, the minimum requirement is that the smart phone has Bluetooth v3.0 + HS. This is because we will be using a Galaxy S II, Model SPH-D710, Hardware Version D710.10, Android Version 4.0.4. with Bluetooth v3.0 + HS to conduct all of our testing. In addition, one of our requirements for our transmitter is that it maintains a certain battery life loss of less than 10% ± 1% of total power in 30mins. Using a different smart phone could impact the battery life requirement. In addition, use of the smart phone's other features outside of our project could impact the battery life requirement.

The Transmitter will be sending its Received Signal Strength Indicator (RSSI) value to the Bluetooth sensor of each speaker system. The RSSI value is the ratio of the power of the received signal over a reference power, and has units of decibels.

$$\text{RSSI} = 10\log\left(\frac{\text{Prec}}{\text{Pref}}\right)\text{dB}$$

RSSI is inversely proportional to distance so as the transmitter moves away from the speaker the RSSI value decreases. This means that the theoretical Distance to RSSI value relationship is:

$$\text{RSSI} = \log\left(\frac{1}{\text{Distance}^2}\right)$$

Where K is the slope of the logRSSI vs. log Distance line. D is the Distance and A is an experimentally determined offset.

$$\text{RSSI} = \text{Kln(D)} + \text{A}$$

To determine a more precise relationship between RSSI and distance an empirically test was conducted. This test consisted of setting up a Bluetooth transmitter (Galaxy Nexus) and a Bluetooth receiver (Galaxy S II). The receiver was then moved a set distance of 10cm from 10cm-3m and the RSSI value was recorded. Each distance was measured for 120sec. A second test was done moving the receiver a set distance of 0.5m from 3.5m-8.5m, again for a duration of 120sec. per set distance. The result showed a logarithmic relationship that matched the theoretical formula:

$$\text{RSSI} = -8.0093\text{ln(D)} + 63.226$$

Furthermore, by averaging the samples at each distance a more defined equation was found in which the averaged RSSI values more closely match the empirically determined equation.

$$\text{RSSI} = -7.9224\text{ln(D)} + 63.226$$

Thus it was found that by taking an average of RSSI values the distance from the transmitter to the receiver can be found more accurately. Furthermore, we can expect a tolerance of 15% from 0-2m and 5% when greater than 2m.

Modularity

The transmitter is one of our major components in the high level block diagram. It is self powered by a 3.7 Volt, Lithium Ion, 1800mAh battery inside phone, and thus does not require an external power source. The transmitter communicates to another major component in the high level block diagram: the speaker system. Inside the speaker system the transmitter communicates only to the Bluetooth sensor unit. For the transmitter to work correctly it must be able to send its RSSI value to the Bluetooth sensor.

2.7 Power Supply

The power supply was reused from an old computer, it was a Bestec ATX-300-12E, with DC outputs:

- +12V , 15A
- +5V, 30A
- +3.3V, 28A
- -12V, 0.8A
- +5VSB, 2A

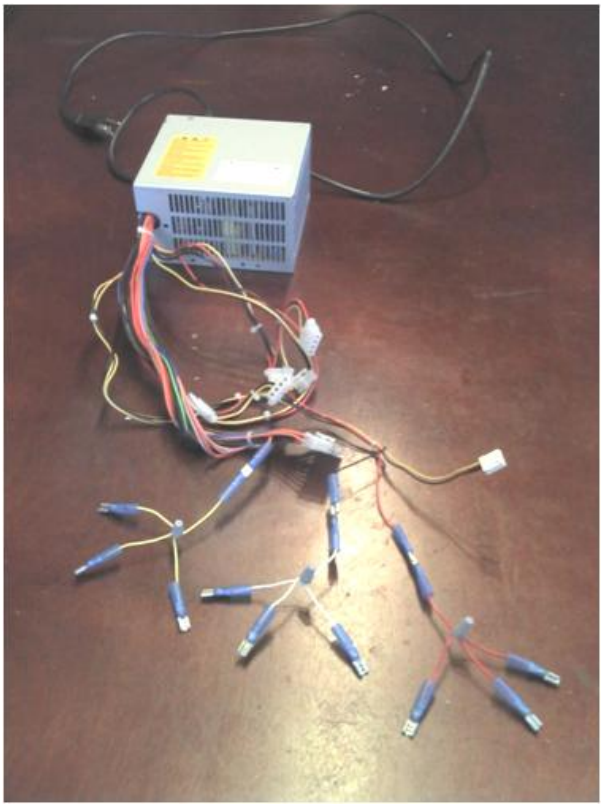


Figure 14: Power Supply with Wire Disconnects

3. Design Verification

3.1 Speaker

The main Requirements and Verification for the speaker consisted of receiving and transducing the music signal.

The speaker needed to provide a steady volume level when the user stands still, and the speakers successfully maintained the same volume in this setting within ±3 decibels, when measured using a smartphone application.

The volume level variance as the user walks around the room could not be measured as the entire system was not completely functional as a whole.

Only one speaker would have been on at a time, satisfying another requirement, because the logic circuit successfully only picked the highest voltage using its algorithm in the demo, using comparators and TTL logic. This also proves that only the closest speaker will be on. The logic truth Table 4 demonstrates this.

The nominal volume of the speakers could be adjusted by adjusting the volume level at the music source. This was demonstrated at the demo.

The volume level of the speakers was not achieved, but in theory it was proved by adjusting the resistance of the digital potentiometer from the increment and up/down signals from the microcontroller. This test demonstrated that the microcontroller could adjust the wiper on the digital potentiometer from 100 ohms to 9.1 k ohms in 15 steps of 600 ohms each.

3.2 Logic Unit

To verify the logic unit, we must first verify that the individual components are functioning properly. The comparators and the TTL logic are to be extensively tested. The requirements and verifications table in Appendix B.1 outline the different performance factors that the components must meet in order for the logic unit to attain satisfactory results. After the individual components are verified, the logic unit must be assembled and the overall functionality of the logic unit must be verified. This means that if the input voltage from speaker 1 is the greatest, the output binary signal must select speaker 1. The same goes for speaker 2 and speaker 3. To set up the test, bias the inputs with different voltages, noting the input with the highest voltage. In our test, we simply grounded two of the inputs and connected the last input to the 5V source. This would ensure that one input had a greater voltage than the other two inputs. We went around and did this for each of the inputs, making sure that each input had the chance to be connected to high. We then tabulated the results, which are listed in Table 4 below.

Table 4

A (V_1 > V_2)	**B (V_1 > V_3)**	**C (V_2 > V_3)**	**b_1**	**b_0**	**SPK 1**	**SPK 2**	**SPK 3**
0	0	0	1	0	0	0	1
0	0	1	0	1	0	1	0
0	1	0	1	1	0	0	0
0	1	1	0	1	0	1	0
1	0	0	1	0	0	0	1

1	0	1	1	1	0	0	0
1	1	0	0	0	1	0	0
1	1	1	0	0	1	0	0

Comparing Table 4 to Table 2 of Section 2.3, we can verify that the logic unit generates the correct outputs that we designed the circuit for.

We also wanted to verify that the logic unit was able to update its output in a reasonable amount of time. In order to measure the time it takes for the logic unit to change the state of its output, we drove one of the inputs with a ramp function ranging from 0V to 5V, using the function generator. The other inputs would either be driven with 2V or be connected to ground. We probed one of the outputs of the logic unit using the oscilloscope and measured the transition time using the oscilloscope's cursors. The results are shown in Figures 15 and 16 below:

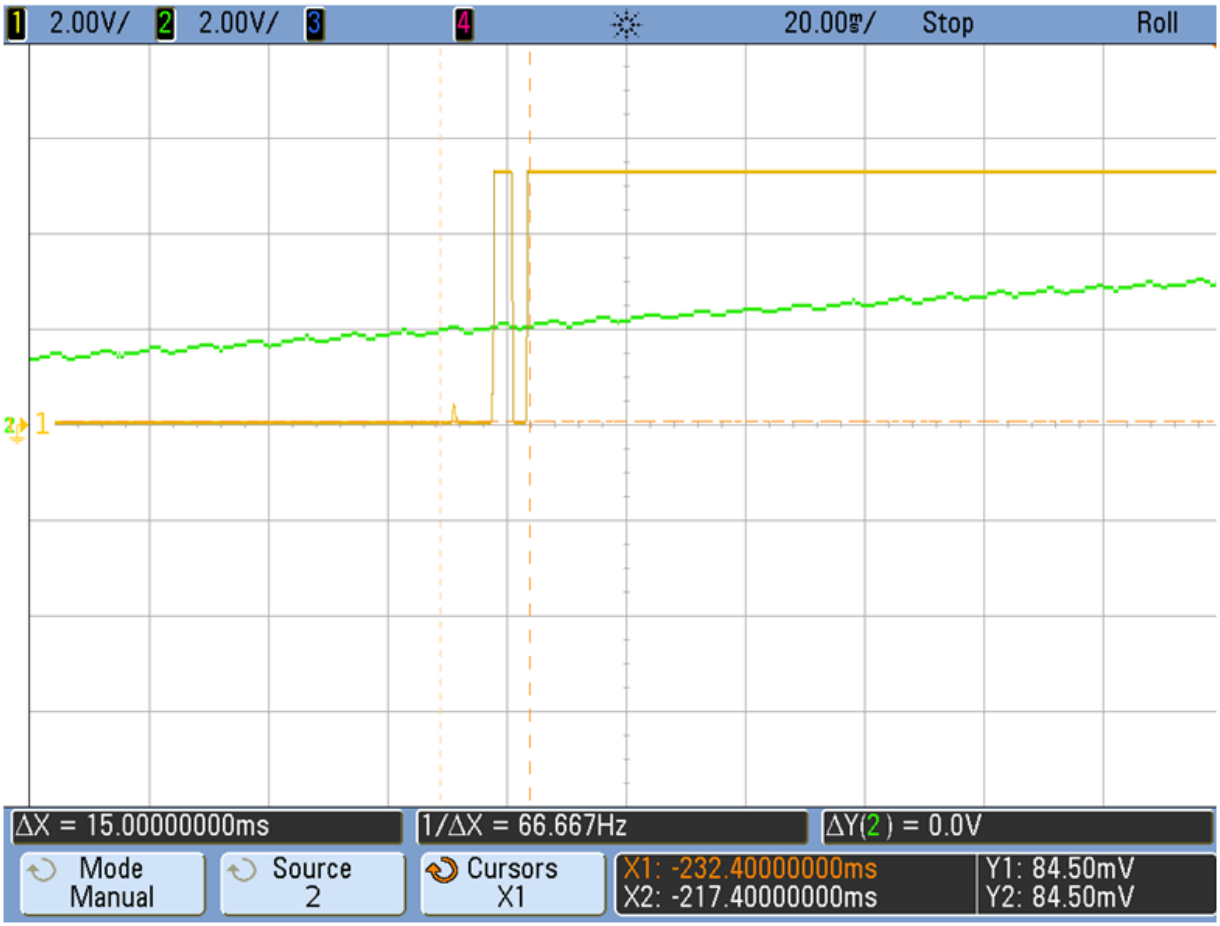


Figure 15

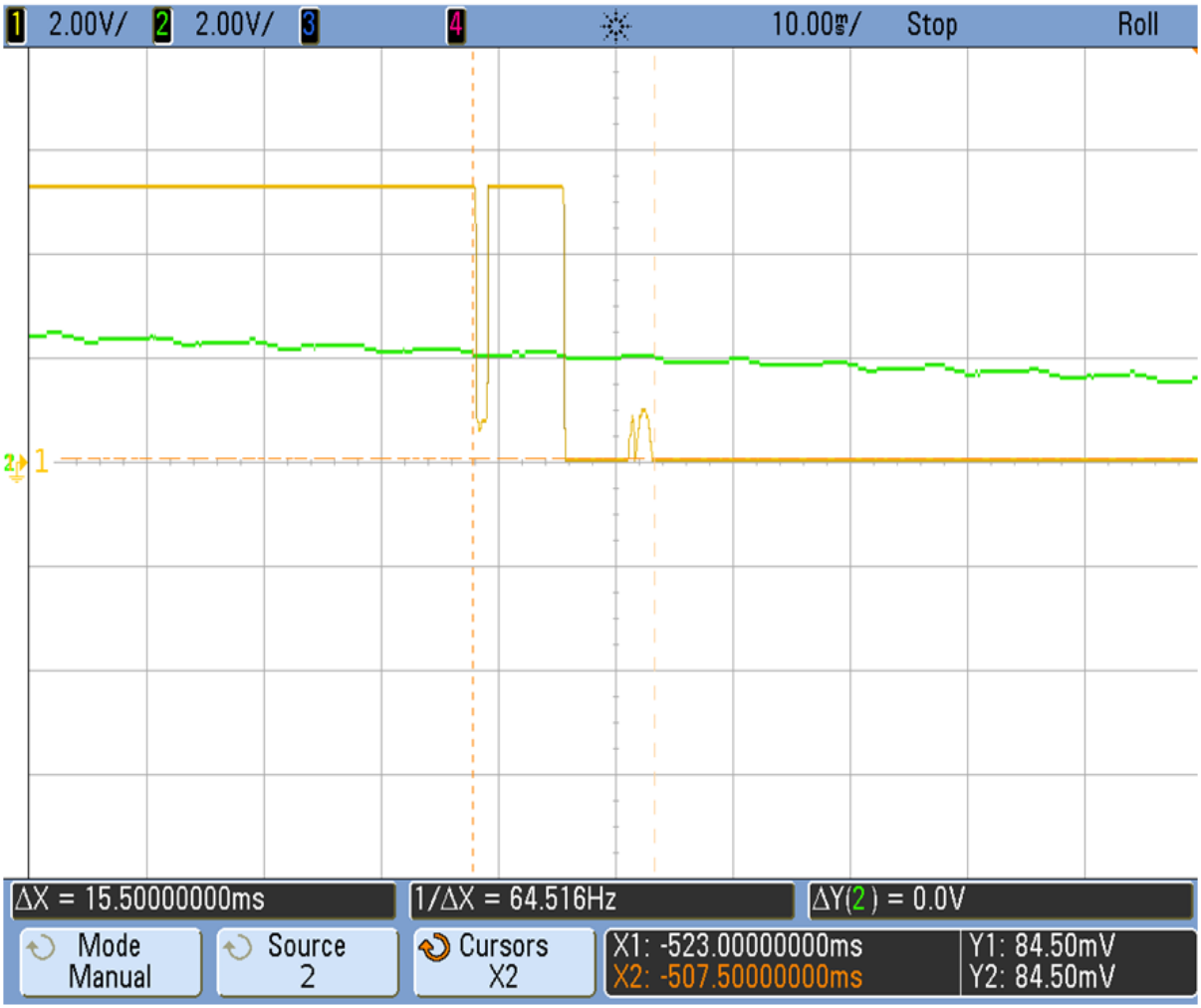


Figure 16

From Figure 15, we see that the rise time is about 15 ms. Similarly, we see from Figure 16 that the fall time is about 15.5 ms. This verifies that the circuit as a whole responds in a timely manner.

3.3 Microcontroller

To verify the microcontroller, the requirements and verifications for the microcontroller must be checked. This is to verify that the MCU is working properly.

We also wanted to verify that the outputs to the digital potentiometer were working properly. To verify this, a distance value was hard coded into the MSP430. We would then probe the U/D and INC outputs from the MSP430 using an oscilloscope. The number of falling edges of the INC output and the state of the U/D output would be noted to verify that the MSP430 would work properly with the digital potentiometer.

The RSSI-to-distance mapping requirement was met using our store-10 algorithm by maintaining an array of the last 10 RSSI values. As each new RSSI value was obtained from the Bluetooth Unit throughout the UART interface, the new value would replace the oldest value and then the average of the

array was computed rounding to the nearest integer. After applying the lookup table the result was an RSSI to distance mapping that was within 5% of the actual distance. This is verified by Figure 17 below:

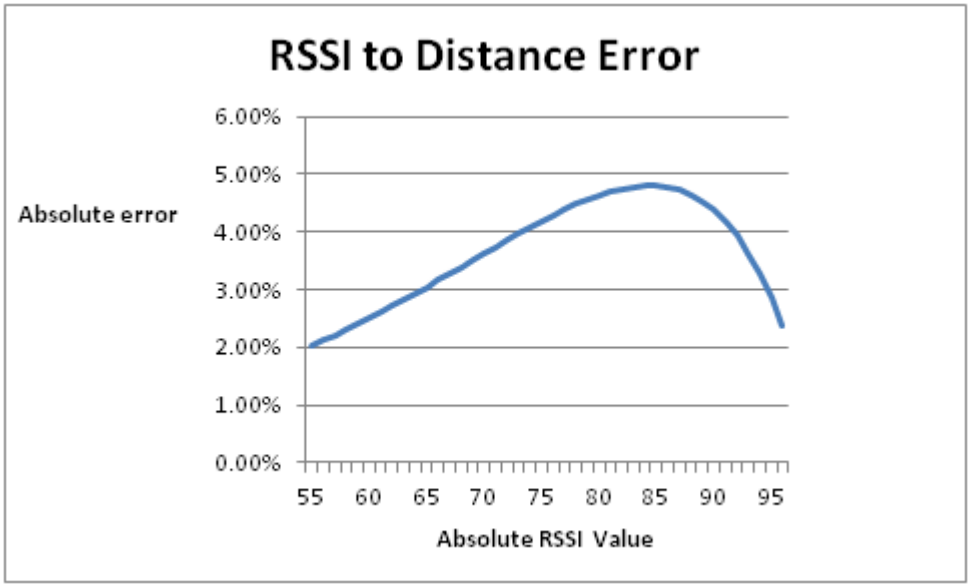


Figure 17

3.4 Transmitter

The Requirements and Verification for the Bluetooth devices consisted of range, latency, and power consumption. The range requirement was to test the connectability range in between the transmitter and the Bluetooth. I was able to successfully connect at a range of 5.5m, which corresponds to 18.0466ft, which is more than the length of several rooms, thus further ranges were not tested.

The latency requirement was to verify we could get RSSI values from the transmitter to the Bluetooth unit in a fast enough time for the MCU to process those values, and send the mapped distance to the logic unit for speaker selection, and finally volume control. Although the Bluetooth unit round trip time was less than 1ms, due to transmission errors related to a low baud rate, the average round trip time for an accurate RSSI value was 138.4ms.

The power consumption requirement consisted of maintaining a certain level of battery power loss with Bluetooth turned on. In fact we exceed this requirement with a loss of only 2% of battery power in 30mins. We tested this with the transmitter turned on and while the Bluetooth unit was requesting RSSI values every 50ms.

4. Costs

4.1 Parts

Name	Hourly Rate ($)	Total Invested Hours	Total = Hourly Rate x 2.5 x Total Hours Invested
Eric Davila	42	150	15750
Chris Goulet	42	150	15750
Roland Legrand	42	150	15750

4.2 Labor

Item	Quantity	Cost/Unit ($)	Total Cost ($)
Wires	1	34.92	34.92
Wire Nuts	2	2.49	4.98
3.5 mm audio adapters	2	5.49	11.98
Heat Shrink Wire Disconnectors	4	5.49	23.96
GE 2.0 Multimedia Speakers	3	11.89	35.67
Intersil X9116 Digital Potentiometers	4	5.72	22.88
WT32-A-AI4 Bluetooth Transmitter/Receiver	3	29.50	88.50
MSP430	4	5	20.00
Resistors	20	0.09	1.80
74AC11004 Hex Inverters	1	2.48	2.48
SN74AHCT02 NOR Gates	1	0.50	0.50
SN7400N NAND Gates	2	1.61	3.02

LM319 Comparators	2	1.08	2.16
NPN BJT	2	1.00	2.00

Total Parts Cost: $254.85

4.3 Grand Total

Section	**Total**
Labor	$47250.00
Parts	$254.85
Total	**$47,504.85**

5. Conclusion

5.1 Achievements

In this project, we were able to build many of the individual components of the system, but we were not able to get the entire system to work as one. There were no glaring issues, however, and we fully expect that this design would be successful if given more time to implement it.

The first achievement of our system was our ability to connect the Bluetooth sensors with the user's transmitter so that the RSSI values could be received. These RSSI values were received and sent to the microcontroller, which would be able to process them.

The next achievement of our system was the logic unit's ability to distinguish which speaker is closest based off of potential RSSI values and only turn on one speaker using that information.

Another achievement was being able to adjust the digital potentiometer using the microcontroller, demonstrating its full range of resistance values.

The power supply was able to successfully power every component necessary, even when the speakers were as far apart as possible.

Finally, we were able to play music through the speakers even though the speakers were taken apart and the mechanical potentiometers were replaced by digital potentiometers.

5.2 Ethical considerations

Specific guidelines from the IEEE code of ethics follow, with our explanation of how we intend to abide by them.

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

Throughout our work we will carefully document every test and simulation done to ensure no false or inaccurate data is reported.

5. *"[To] improve the understanding of technology; its appropriate application, and potential consequences;"*

Learning how volume is changed in a speaker and Bluetooth communication are the important technology aspects of our project. We are learning these technologies, and will be able to explain them to others who are interested in our project.

6. *"[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;"*

Throughout this project we will be improving our creative thinking, research, designing, debugging, and presentation skills. Through research we will learn how each of our components work, their limitations, and the safety hazards involved.

7. *"[To] seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;"*

Our project will go through a comprehensive investigation starting with a project proposal, design review, project demo, and final presentation. The University of Illinois faculty and staff of ECE 445 will check and observe the designs and demos to ensure that to the best of their knowledge, our project is free of major flaws and safety hazards, and that it is unique and properly gives credit to the work of others.

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

Following the guidelines by the FCC in radio transmission we will ensure our communication does not interfere with other objects.

10. *"[To] assist colleagues and co-workers in their professional development and to support them in following this code of ethics."*

By documenting our work through several academic papers and presentations such as a project proposal, design review, presentation, and final paper we will be demonstrating our understanding of this code of ethics. And by reviewing other design reviews as well as critiquing each members work, we will ensure that we are developing and supporting each other in following these standards.

5.3 Safety

This section lists all potential safety considerations when operating our Automatic Volume Control System. Each major component lists potential safety hazards and procedures to limit and respond to those hazards.

Speaker System

The speakers must not emit a volume exceeding 91 decibels, since permanent ear damage can occur within 2 hours at that level.

Transmitter

The transmitter is a Samsung Galaxy S II smart phone. In addition to the normal safety requirements of smart phones, the following safety considerations are relevant for using this device in our system. The transmitted signal from the device must be in FCC's class B for radio frequencies 2400-2483.5 MHz. FCC Section 15.249 also says the fundamental frequency field strength must be in the range of 50 - 500 uV/m. These requirements ensure that the device is not interfering with other objects in the radio spectrum. Since Bluetooth is already configured on the device, the user will not be able to adjust the transmitted signal, thus ensuring the signal stays within the FCC ranges.

Because the device will be running Bluetooth, the device may heat up more often than normal non-Bluetooth use. If the device overheats remove from pocket or away from close contact with skin and wait until device cools before inserting back into pocket.

Control Hub

All electronic components must be contained in a closed container with no loose or exposed wires. Do not expose to water or connect to other circuits other than those specified in our design. All components that need to be grounded must be properly grounded. Never work or tinker on any part of the system if there is live power running through it.

Power Supply

The conductors must not be exposed, there should be in insulator over all conductive materials that could potentially shock someone. All components that need to be grounded must be properly grounded. Never work or tinker on any part of the system if there is live power running through it.

5.4 Future work

Instead of sticking with a digital potentiometer, a variable gain amplifier may have been a more successful way to implement the volume control. Digital potentiometers are flawed in that they only allow a very small amount of current, while a variable gain amplifier has a much greater capacity for current. In addition, a variable gain amplifier would allow the volume to vary across a much greater range.

The power supply was a bit bulky, so constructing a smaller supply for each speaker system would go a long ways in untangling the project and making it easier to transport. The current power supply also supplied many different voltage levels that were not necessary for the system.

The Bluetooth sensor we chose, the WT32, has more features that could be implemented in a future project, such as a wireless music source. Although, a smaller Bluetooth module such as the WT12 is cheaper and has the same functions as the WT32 minus the wireless music source. In addition other

Bluetooth module such as the XBee could be used to more closely interact with a microcontroller such as the Arduino Uno.

5.5 Special Thanks

Our TA Igor Federov, for his weekly support, critique, and encouragement. As well as his thoughtful insights on how to design our speaker system.

Professor Carney for his initial direction and leadership in this course.

The PCB shop guys Mark Smart, Wally Smith, and Skot Wiedmann for the assistance in soldering and making our PCBs and other electrical equipment.

Appendices

Appendix A: References

Electronic Code of Federal Regulations. "§ 15.247 Operation within the bands 902-928 MHz, 2400-2483.5 MHz, and 5725-5850 MHz." http://www.ecfr.gov/cgi-bin/text-idx?c=ecfr&SID=b60f74fb4cb08412a30b436401a5a83d&rgn=div8&view=text&node=47:1.0.1.1.16.3.234.31&idno=47 , 21 Feb 2013. Web. 23 Feb 2013

Electronic Code of Federal Regulations. "§ 15.249 Operation within the bands 902-928 MHz, 2400-2483.5 MHz, and 5725-5850 MHz." http://www.ecfr.gov/cgi-bin/text-idx?c=ecfr&SID=b60f74fb4cb08412a30b436401a5a83d&rgn=div8&view=text&node=47:1.0.1.1.16.3.234.32&idno=47 , 21 Feb 2013. Web. 23 Feb 2013

IEEE.org. "7.8 IEEE Code of Ethics." http://www.ieee.org/about/corporate/governance/p7-8.html, 2013. Web. 23 Feb 2013

Noise Help. "Noise Dose Chart: Noise Exposure Limits." Noise Help, 2013. web. 23 February 2013. http://www.noisehelp.com/noise-dose.html

"Bluetooth Audio Module with Internal Antenna and iWRAP 4.0 firmware." Semiconductorstore.com. web. 24 February 2013. http://www.semiconductorstore.com/Wireless-Bluetooth-Audio-Module-with-Internal-Antenna-and-iWRAP-4-0-firmware-WT32-A-AI4/P/44694

"WT32_Datasheet-1.PDF" Sparkfun.com 3 September 2008 web. 26 February 2013 http://www.sparkfun.com/datasheets/Wireless/Bluetooth/WT32_Datasheet-1.pdf

"iWrap4_UserGuide" Scribd.com 5 May 2010. web. 26 February 2013. http://www.scribd.com/doc/70256674/iWRAP4-User-Guide-1

"Serial Console" wiki.openwrt.org 24 April 2013. web. 30 April 2013 http://wiki.openwrt.org/doc/hardware/port.serial

"BGTerm" techforum.blugiga.com 29 April 2009. web. 30 April 2013 https://techforum.bluegiga.com/protectedstore/38541/8424/127/bluegiga_610/51dd522f18d391efd4841fa38cb26728/BGTerm_manual.pdf

"MSP430 Family Guide" Jan 2012. web. 30 April 2013 http://www.ti.com/lit/ug/slau144i/slau144i.pdf

Appendix B: Requirements and Verification

B.1 Central Hub

Requirements:	Verification:
1.Music Source:	
1. The speaker receives the signal from the music source a) The music source has an amplitude > 0.05 ± 0.01V	1a) Measure the voltage that the speaker receives ● Use voltmeter in AC setting to probe the differential between the received signal and ground ● The signal has an amplitude > 0.05 ± 0.1V
2. Logic Unit	
1. Logic Unit must receive correct power a) Vcc pins in Comparators, TTL logic, and Demux receives 5VDC ± 0.25VDC b) GND pins in Comparators, TTL logic, and Demux receive 0VDC ± 0.25VDC	1a) ● Make sure Comparator Vcc pins are connected to their source voltage ● Power on multimeter and set it to measure DC voltage ● Probe each Vcc pin with the multimeter, and record the reading on the multimeter ● Repeat for TTL logic and Demux 1b) ● Make sure Comparator GND pins are connected to digital ground ● Power on multimeter and set it to measure DC voltage ● Probe each GND pin with the multimeter, and record the reading on the multimeter ● Repeat for TTL logic and Demux
2. Comparators must have correct response to given inputs a) Comparator output must be 5V ± 0.25V when the non-inverting input is higher than inverting input b) Comparator output must be 0V ± 0.25V when the inverting input is higher than the non-inverting input c) Comparator must transition correctly; comparator must correctly output 5V ± 0.25V or 0V ± 0.25V as input difference gets closer to 0.01 V	2a) ● Connect the non-inverting input of the comparator to a 5V source and the inverting input to digital ground ● Power on DMM and set it to measure DC voltage ● Probe the output of the comparator and record the multimeter reading 2b) ● Connect the inverting input of the comparator to a 5V source and the non-inverting input to digital ground ● Power on DMM and set it to measure DC voltage ● Probe the output of the comparator and

3. Logic Unit time delay must be small a) Comparator transition delay must be less than 1 µs b) Time delay for TTL logic in logic unit must be less than 0.1 µs	record the multimeter reading 2c) ● Power on the oscilloscope and connect probes to channel X and Y ● Power on the waveform generator and set it to the triangle waveform (0 - 10V) ● Connect the non-inverting input to the waveform generator and connect the inverting input to a 5V source ● Probe the non-inverting input with channel X and probe the output with channel Y ● Measure the output for two periods, then stop the oscilloscope ● Record the reading in channel Y at the times that the waveform input is at 4.99V and 5.01V 3a) ● Power on the oscilloscope and connect probes to channel X and Y ● Power on the waveform generator and set it to the triangle waveform (0 - 10V) ● Connect the non-inverting input to the waveform generator and connect the inverting input to a 5V source ● Probe the non-inverting input with channel X and probe the output with channel Y ● Measure the output for one period, then stop the oscilloscope ● Observe the time the reading of channel X crosses 5V; the time of the crossing should correspond to a transition in the output read by channel Y ● Measure the transition time 3b) ● Power on the oscilloscope and connect probes to channel X and channel Y ● Probe output 1 (b_0) and output 2 (b_1) of the TTL logic circuit ● Utilizing a 5V source and digital ground, generate an input state of 001; this should generate an output state of 01 ● Change the input state to 101 to generate an output state of 11 ● Find the point of transition in the oscilloscope reading and measure the transition time
3. Power Supply	
1. The Power Supply has output feeds according to the needs of the rest of the system a) Logic Unit receives 3V ± 10% and 5V ± 10% DC	1a) Test the voltage of the output to the logic unit ● Test the voltage on both wires that go from the power supply to the logic unit using a voltmeter in the DC setting, from the wire to

b) WT32 Bluetooth Sensors receive between 1.8-3.6V DC c) Amplifier receives 5V ± 10% DC d) MCU receives 5V ± 10% DC	ground. ● One wire reads 3V ± 10% DC and the other reads 5V ± 10% DC 1b) Test the voltage of the output to the Bluetooth sensors ● Use voltmeter in DC setting to probe the difference in voltage between the output from the power supply to the Bluetooth sensors and ground ● Verify it is at 1.8-3.6V DC 1c) Test the voltage of the output to the amplifier ● Use voltmeter in DC setting to probe the difference in voltage between the Vcc wire that goes to the amplifier from the power supply and ground ● Verify that it is 5V ± 10% DC 1d) Test the voltage of the output ● Use voltmeter in DC setting to probe the difference in voltage between the microprocessor power input and ground ● Verify that it is between 5V ± 10% DC

B.2 Speaker System

Requirements:	Verification:
1. WT32 Bluetooth Sensor	
1. Range a)Bluetooth sensor connects to Transmitter within 0.10m ± 0.01m from speaker. b) Bluetooth sensor connects to the Transmitter within 10m ± 0.01m 2. Latency a)One way Latency between Bluetooth and MCU is 1µs ± 0.5µs	1a) Test close range visibility ● Program MCU to detect nearby devices and display on the SPI bus. ● Use SPI packet watching software to detect information on SPI bus when 0.10m ± 0.01m from speaker. 1b) Test far range visibility ● Program MCU to detect nearby devices and display on the SPI bus. ● Use SPI packet watching software to detect information on SPI bus when 10m ± 0.01m from speaker. 2a) Send data packet from Bluetooth to MCU ● Program MCU to detect packets and display on the SPI bus. ● Analyze SPI packet data and verify packets are arriving within 1µs ± 0.5µs

3. Power Requirements a)Bluetooth sensor has -0.5µA ± 0.1µA when not transmitting b) Bluetooth sensor has -5µA ± 0.1µA when transmitting	3a) Check power from Bluetooth when not transmitting ● Measure output pin 16 (UART TXD) using Multimeter and verify the current is -0.5µA ± 0.1µA when not transmitting 3b) Check power from Bluetooth when transmitting ● Transmit signal from Bluetooth sensor to MCU ● Measure output pin 16 (UART TXD) using Multimeter and verify the current is -5µA ± 0.1µA when transmitting
2. Microcontroller Unit	
1. MCU must receive the correct power a) Vcc pins receives 5VDC ± 0.25VDC b) GND pins receive 0VDC ± 0.25VDC	1a) ● Make sure Vcc pins are connected to their source voltage ● Power on multimeter and set it to measure DC voltage ● Probe each Vcc pin with the multimeter, and record the reading on the multimeter 1b) ● Make sure GND pins are connected to digital ground ● Power on multimeter and set it to measure DC voltage ● Probe each GND pin with the multimeter, and record the reading on the multimeter
2. Incoming RSSI signal should be mapped to the correct distance value a) For an incoming RSSI signal, the distance value that is sent to the Central Hub should match the predetermined mapped value by ±10%	2a) ● Wire the Bluetooth unit to the USART input of the MCU ● Wire the distance output (PB0 - PB7) to the DAC ● Power on the oscilloscope and connect a probe to channel X ● Power on the transmitter and hold the transmitter 1 meter from the speaker system ● Probe the output of the DAC using the oscilloscope ● Run the trial for 10 seconds, and record the oscilloscope reading ● Repeat the above procedures for successive trials, changing the distance between the transmitter and the speaker system to 2m,

3. MCU turns off speaker when the input from the Control Hub is low a) For a Control Hub input of binary 0, output to speaker should be 0 VDC ± 0.01VDC b) Latency of transition from on to off and off to on should be no greater than 1 µs	4m, 8m, and 10m 3a) ● Connect input from Central Hub (PA0) to digital ground ● Connect output to speaker (PC0 - PC7) to DAC ● Power on multimeter and set it to measure DC voltage ● Probe the output of the DAC and record the multimeter reading 3b) ● Connect output to speaker (PC0 - PC7) to DAC ● Connect the USART input (PD0) to the Bluetooth unit ● Power on the transmitter and hold the transmitter 1 m from the speaker system ● Power on the oscilloscope and connect a probe to channel X ● Power on the waveform generator and set it the square wave function (0 - 5 V) ● Connect the waveform generator to the input from the Central Hub (PA0) ● Probe the output of the DAC using the oscilloscope ● Record the oscilloscope reading and measure the time delay between the change of output states
3. Amplifier	
1. Amplifier steps up the signal from the microcontroller by a linear factor a)Amplifier increases input signal from microcontroller by a linear factor independent of the value of the input signal for a voltage level of -10V to 10V	1a) Measure the voltage ratio of input to output by doing a voltage sweep ● Feed a voltage ramp (-10 - 10V) signal into the amplifier ● Measure the input voltage and the derivative of the output voltage from the amplifier using an oscilloscope ● The slope reading from the oscilloscope must be 0 ± 0.3V/s
4. Speaker	
1. Speakers emit a steady volume at the user's location a) Volume level only varies by ± 3 decibels when the user stands still	1a) Measure the volume level at user's location ● Use sound meter app to measure the sound level with the speakers playing music 1ft from the speakers ● The sound varies by no more than ± 3

b) Volume level only varies by ± 3 decibels when measured by a user walking around the room	decibels while remaining still 1b) Measure the volume level at user's location ● Start with phone in one location ● Measure the volume of the stereo using sound meter app ● As you walk around the room at 2 meter intervals, the volume changes by no more than ± 3 dB as read from your location
c) When speaker volume is changed, sound pressure at the transmitter should be within 10% of the desired change.	1c) MCU correctly interfaces with the volume control ● Use Sound Meter app to Measure the sound pressure .3 ± 0.1 meters from the speaker ● Move the transmitter to 2 ± 0.2 meters ● Sound pressure should be within 10% of the desired change
2. The closest speaker is the only one emitting music a) Only one speaker is on at a time	2a) Check switch voltages as user moves around room ● Move to 1 ± .2 meters directly in front of each speaker ● Use a voltmeter in DC settings to read the voltage out of each switch ● Only 1 switch must have a voltage greater than .3V ● Verify by listening and only hearing sound from one speaker
b) The closest speaker is the only speaker on	2b) Check that only the closest speaker is on ● User with phone begins close to one speaker ● Only the closer speaker is playing ● Check that that RSSI value for that speaker is the largest of all three RSSI values ● User walks closer to a different speaker ● When the user is ± 1 meter of the midpoint between the two speakers, the old speaker turns off and the new speaker turns on ● Only the new speaker remains playing ● The current speaker has the largest RSSI value of all of the speakers
3. Adjusting the volume that the user hears is possible by adjusting the volume at the music source, and that volume is constant as the user walks around again a) The volume that the user hears changes as the music source's volume is changed	3a) Monitor the volume level at the user's location ● User with phone 1 ± 0.2 meters from music source remains still ● The volume of the music source is adjusted ● The volume measured by the sound meter on the user's phone changes according to the adjustment of the volume at the music source
b) The volume level of the user still remains within ± 3 decibels of the new level when the volume of the music source is adjusted	3b) Monitor the volume level at the user's location ● The user measures the volume at their location using the sound meter on their phone ● When the user moves around the room again, it remains within ± 3 decibels of the original volume measured before walking around

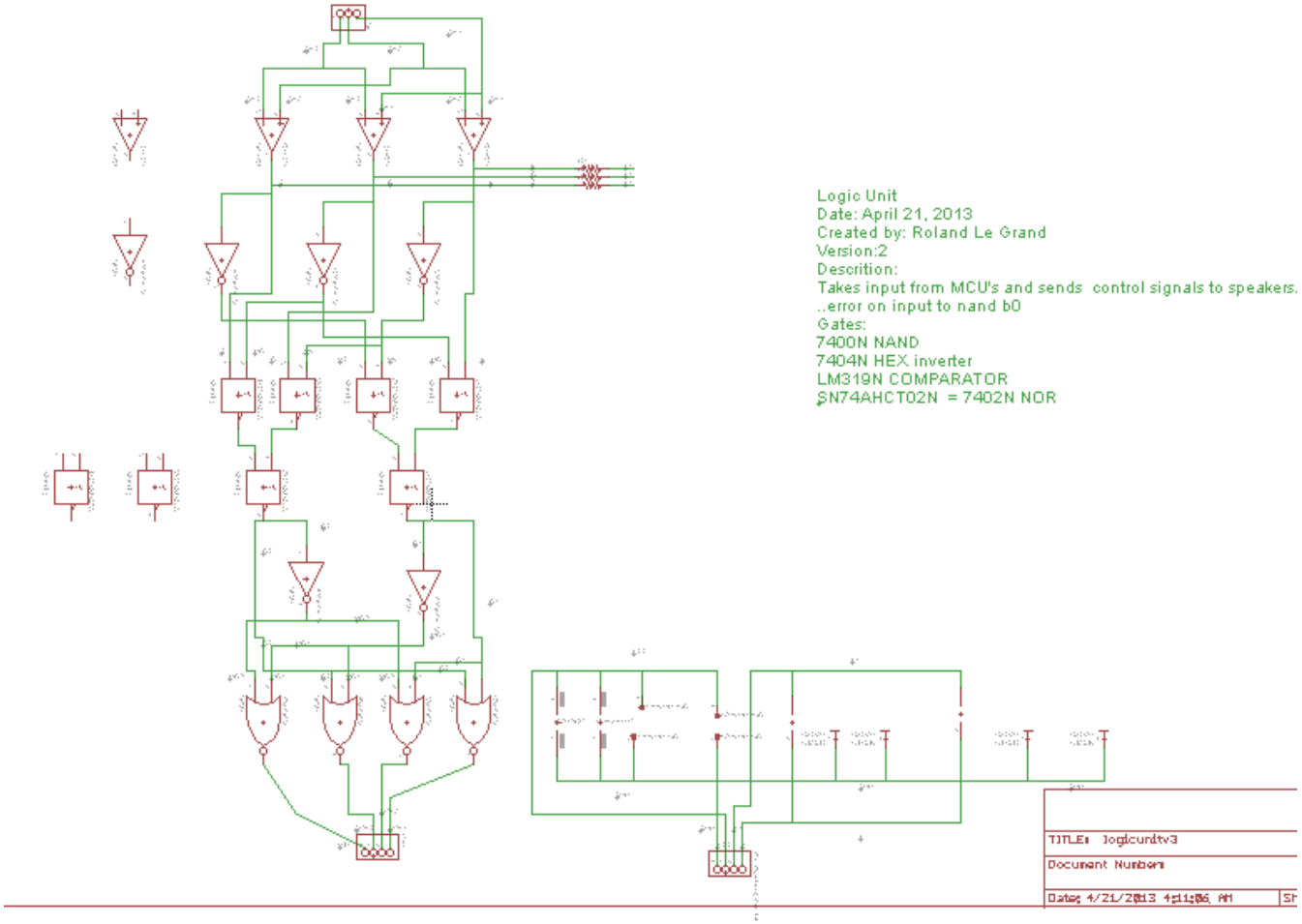
B.3 Transmitter

Requirements:	Verification:
1. Transmitter	
1. Multiple Devices a) Transmitter connects to all three WT32 Bluetooth sensors when within 10m ± 0.01m of an edge speaker 2. Latency a) One way Latency between Bluetooth and transmitter is 10ms ± 5ms 3. Power a)Device must not lose more than 10% ± 1% of power in 30 minutes.	1a). Test Bluetooth visibility ● Program MCU to detect nearby devices and display on the SPI bus. ● Use SPI packet watching software to detect information on SPI bus ● verify all devices have connections when 10m ± 0.01m from an edge speaker. 2a). Send data packet to Bluetooth unit ● Program MCU to detect packets and display on the SPI bus. ● Analyze SPI packet data and verify packets are arriving within 10ms ± 5ms 3a). Check device power ● Use battery trace app to record power level with Bluetooth on. ● Verify battery does not lose more than 10% ± 1% of power in 30mins.

Appendix C: Schematics and PCBs

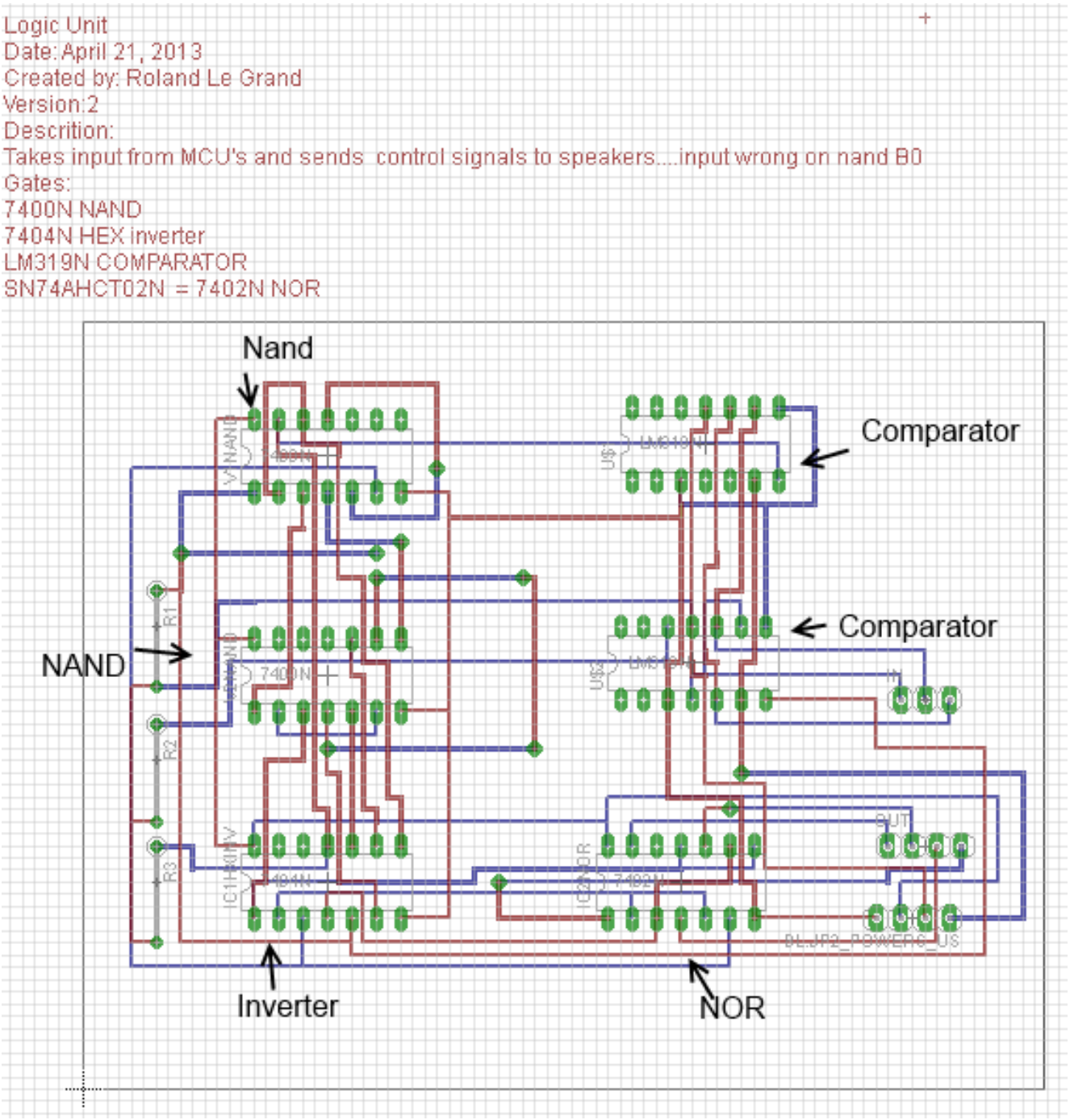
C.1 Logic Unit

Schematic



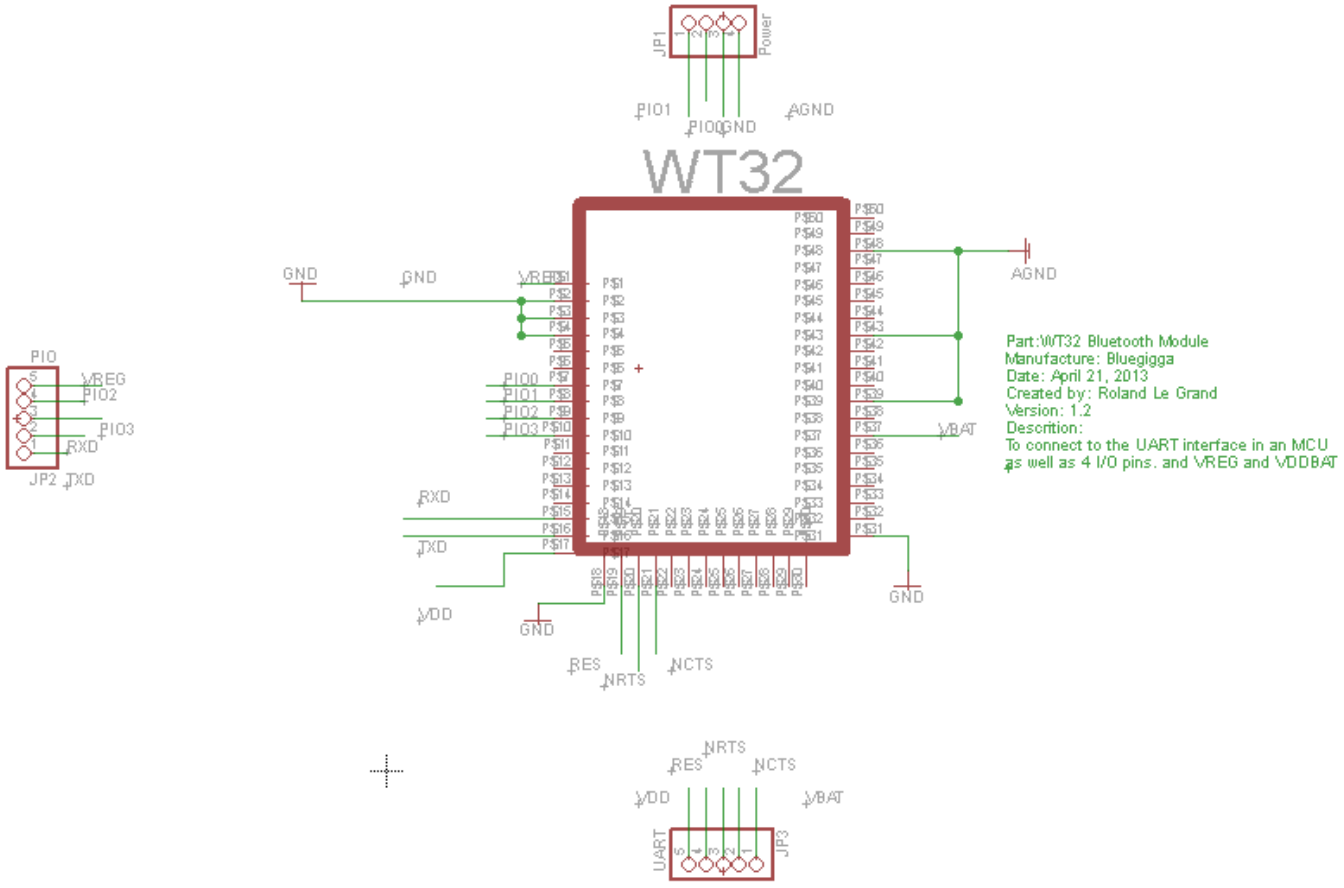
PCB

Logic Unit
Date: April 21, 2013
Created by: Roland Le Grand
Version: 2
Description:
Takes input from MCU's and sends control signals to speakers....input wrong on nand B0
Gates:
7400N NAND
7404N HEX inverter
LM319N COMPARATOR
SN74AHCT02N = 7402N NOR



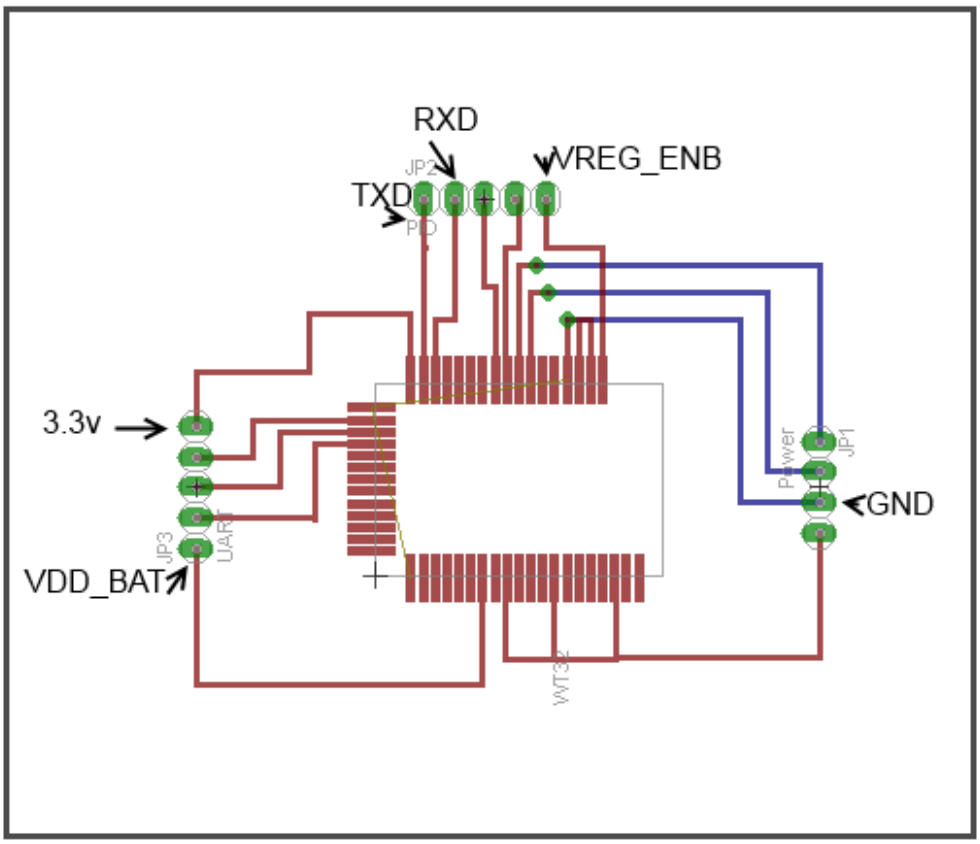
C.2 Bluetooth Sensor

Schematic



PCB

Part:WT32 Bluetooth Module
Manufacture: Bluegigga
Date: April 21, 2013
Created by: Roland Le Grand
Version: 1.2
Descrition:
To connect to the UART interface in an MCU
as well as 4 I/O pins. and VREG and VDDBAT



Appendix D: Tables and Charts

D.1 RSSI to Distance Lookup Table

RSSI	Distance	RSSI	Distance	RSSI	Distance
29	0.082601	53	0.399235	77	1.92962
30	0.088206	54	0.426323	78	2.060544
31	0.09419	55	0.455249	79	2.200351
32	0.100581	56	0.486138	80	2.349644
33	0.107406	57	0.519122	81	2.509066
34	0.114693	58	0.554344	82	2.679306
35	0.122475	59	0.591956	83	2.861096
36	0.130785	60	0.63212	84	3.05522
37	0.139659	61	0.67501	85	3.262515
38	0.149134	62	0.720809	86	3.483876
39	0.159253	63	0.769715	87	3.720256
40	0.170058	64	0.82194	88	3.972674
41	0.181597	65	0.877709	89	4.242218
42	0.193918	66	0.937261	90	4.530051
43	0.207075	67	1.000854	91	4.837414
44	0.221125	68	1.068761	92	5.165631
45	0.236128	69	1.141276	93	5.516117
46	0.25215	70	1.218712	94	5.890383
47	0.269258	71	1.301401	95	6.290044
48	0.287527	72	1.3897	96	6.716821
49	0.307036	73	1.483991	97	7.172555

50	0.327868	74	1.584679	98	7.65921
51	0.350114	75	1.692199	99	8.178885
52	0.373869	76	1.807014	100	8.733819

D.2 Bluetooth Configuration

	Syntax	example
Command	SET PROFILE HFP ON	turns on the hands free profile to place and receive calls on a hands free device
Command	SET BT SSP 3 0	SET BT SSP 3 0 sets up SSP with 3 = none 0 = man in the middle protection disabled.
Command	SET CONTROL CONFIG 0000 0050 0021	sets up control options 5=Bluetooth address will appear with connect command 2&1= store data of discovered devices 0 = rssi value displayed on inquire
Command	SET BT PAGEMODE 3 9000 1	sets up bt options 3 = iwrap is visible in inquiry and answers calls 9000=timeout for connection establishment (ms) convert to decimal and multiply by 0.625 1=iwrap connectable 1.28sec
Command	SET CONTROL BAUD 9600, 8N1	sets the baud rate at 9600 with 8 bit data no parity and 1 stop bit

To initiate an HFP connection the following syntax is used:

	syntax	example
Command	CALL {address} 111F HFP	CALL 00:07:80:80:52:27 111F HFP connects to address of transmitter with HFP option
Response	CALL {linkid}	CALL 0 returns linkid of transmitter
Response	CONNECT {linkid}	CONNECT 0 HFP 3 the transmitter is connected on linkid 0

Once a HFP connection is set up between the WT32 Bluetooth sensor and the transmitter, the WT32 sensor can request the receiver signal strength indicator (RSSI) with the following command:

	syntax	example
Command	RSSI {linkid}	RSSI 0, linkid 0 = transmitter
Response	RSSI {address} {rssi value}	RSSI 00:07:80:80:52:27 -10, this is the RSSI value -10 coming from the transmitter

Alternatively, the Bluetooth sensor can request RSSI values with an inquiry command without connecting to the device

Command	INQUIRY 7	scan nearby devices 7= timeout in sec times 1.28
Response	INQUIRY_PARTIAL{Bluetooth address}{ident} ”” {rssi value}	ex. INQUIRY_PARTIAL 00:14:a4:8b:76:9e 72010c “” -71...an rssi value of -71dB

The RSSI value is a signed 7 bit number ranging from -128 to 20, corresponding to the signal strength coming from the transmitter. This value is then sent to the WT32 sensor on the Central Hub.

Appendix E: Equations

Distance vs. Sound Pressure Calculations:
The sound pressure, is related to distance by the distance law:

$$p \propto \frac{1}{r}$$

$$\frac{p_2}{p_1} = \frac{r_1}{r_2}$$

$$p_2 = p_1 \frac{r_1}{r_2}$$

So if the user increases his or her distance by a factor of K, the sound pressure at the sound source (i.e. the speakers) must increase by the same factor K in order to keep the sound pressure the same from the perspective of the user.

Baud Rate Formulas and Calculations:
The baud rate is the number of distinct symbol changes made to the transmission per second. The number 0.004096 was empirically found and is used in the WT32 Bluetooth Sensor. The PSKEY UART ABUD RATE is the persist key to be used in UART.
The baud rate for the WT32 Bluetooth Sensor is dictated by the equations below.

$$\text{Baud rate} = \frac{\text{PSKEYUARTABUDRATE}}{0.004096} = \frac{39}{0.004096} = 9521.484375$$

$$\text{Percent error} = \frac{9600 - 9521.484375}{9521.484375} = 0.82\%$$

This means some of the messaging between the Bluetooth unit and the MCU will be garbled.

Distance to RSSI Calculations:
Our distance to RSSI calculation was developed empirically by moving the transmitter 0.5m from 0m to 5.5m at 20sec. intervals. We found a logarithmic relationship between Distance and |RSSI| from the equation below.

$$\text{Distance} = e^{\frac{|\text{RSSI}| - 67.486}{14.999}}$$

This means the farther we get away from the speaker the less our |RSSI| value will change.

Appendix F: Code

F.1 MSP430 Code for control unit/digital potentiometer

```
#include <msp430g2553.h>

const char UD_HIGH = 0x08; // P1.3
const char UD_LOW = 0xF7;
const char INC_HIGH = 0x10; // P1.4
const char INC_LOW = 0xEF;

const int MAX_POS = 15;
const int MIN_POS = 0;


/*
 * main.c
 */
int main(void) {

        /*
         * Initialize the MCU
         */

  WDTCTL = WDTPW | WDTHOLD;       // Stop watchdog timer

        // Lock MSP430 if clock is not configured right:
        if (CALBC1_1MHZ == 0xFF || CALDCO_1MHZ == 0xFF)
        {
                while (1);
        }

        // Change clocks to run at 1MHz:
        BCSCTL1 = CALBC1_1MHZ;
        DCOCTL = CALDCO_1MHZ;

        // Set directions of pins:
        P1DIR &= 0; // Set all pins to input before setting output pins
        P1DIR |= 0x58; // U/D, INC
        P2DIR &= 0;
        P2DIR |= 0x3F;

        // Declare all variables:
        int ON;
        float K, distance;
        int position, new_pos, adjust;
        char num;

        int count;


        /*
         * Set increment to high
```

```
 */
P1OUT |= INC_HIGH;

/*
 * Set speaker to standby
 */
P1OUT |= UD_HIGH; //up_down = false
int a;
for(a = 0; a < 15; a++)
{
        P1OUT &= INC_LOW; //increment = false
        P1OUT |= INC_HIGH; //increment = true
}
position = MAX_POS;

while (1)
{
        // Obtain RSSI values:


        // Set distance:
        num = (int)(distance*4);
        P2OUT = num;

        // Evaluate ON/OFF state
        int e;
        count = 0;
        for(e = 0; e < 500; e++)
        {
                count = count + (P1IN & 0x01); //increment if ON/OFF is high
        }

        if (count > 250) // Implement confidence interval
                ON = 1;
        else
                ON = 0;

        // Toggle digipot:
        if (ON==1)
        {
                P1OUT |= INC_HIGH; //increment = true

                // calculate gain K
                K = 15-distance;

                // evaluate correct movement for wiper
                if ((K-(int)K) < 0.5)
                        new_pos = (int)K;
                else
                        new_pos = (int)K + 1;

                adjust = new_pos - position;

                // adjust the wiper
                if (adjust < 0)
```

```
                {
                        P1OUT &= UD_LOW; //up_down = false
                        int b;
                        for (b = 0; b > adjust; b--)
                        {
                                P1OUT &= INC_LOW; //increment = false
                                P1OUT |= INC_HIGH; //increment = true
                        }
                }
                else
                {
                        P1OUT |= UD_HIGH; //up_down = true
                        int c;
                        for (c = 0; c < adjust; c++)
                        {
                                P1OUT &= INC_LOW; //increment = false
                                P1OUT |= INC_HIGH; //increment = true
                        }
                }

                // remember wiper position
                if (position + adjust > MAX_POS)
                {
                        position =  MAX_POS;
                }
                else if (position + adjust < MIN_POS)
                {
                        position =  MIN_POS;
                }
                else
                {
                        position = position + adjust;
                }

        }
        else
        {
                P1OUT |= UD_HIGH; //up_down = false
                int i;
                for(i = position; i <= MAX_POS ; i++)
                {
                        P1OUT &= INC_LOW; //increment = false
                        P1OUT |= INC_HIGH; //increment = true
                        position++;
                }
        }
}
return 0;}
```

F.2 MSP430 Code for Bluetooth/RSSI

```
while(1){                                                    //main while loop
        _delay_cycles(50000);
        TimerA_UART_print("RSSI 0\r\n");
        TimerA_UART_tx(rxBuffer);                                    // Transmit the received data
                /g for rssi value
                        TimerA_UART_print(mystring);         // echo data string
                        buffcnt = 0;                         //reset buffer cnt
                        rxBuffer = '~';                      //reset dataread
                 }
        }
}
void compare_string(unsigned char *string)          //check to make sure it is "{RSSI} {bt addr} {rssi value}"
{
        if ((strncmp(string,"RSSI ",5) == 0) && (strlen(string) >8))
        {
                rssibuff[rssicnt%10] = atoi(string +strlen(string) -5);  // convert rssival from char to int
                P1OUT ^= (BIT0);
                processarray(rssibuff);                              //processrssival to find avg.
        }
}

void processarray(int *rssibuff)
{
        avgrssival = 0;                                                      //get new avgrssival
        for ( i = 0; i < 10; i++)
        {
                avgrssival = avgrssival + rssibuff[i];               //loop through past 10 rssival
        }
        avgrssival = avgrssival / 10;                                //divide by size of rssibuff
}
```