Automated Volume Adjusting Bluetooth Speakers

Veronica Tu

ECE-499: Capstone Design Project

Advisor: Professor Hedrick

March 26, 2020

Report Summary

Bluetooth is used in many modern devices and in this project, I explore Bluetooth and the many ways in which it can be used in electronic device design. The goal of this project is to ensure that the user hears the audio played from a Bluetooth connected portable device at the same volume regardless of the distance from the speaker by using the signal strength of the wireless connection. The system must be Bluetooth-enabled to connect to a handheld device from a static location for one user in a typical apartment-sized space. It must be as hands-off as possible, besides initial set up and calibration, and with easy installation on a home sound system. There also must be manual input availability of the initial volume, so the user can change the volume at which the user wants to listen. The system must be able to read in the signal strength, use that value in a volume adjustment calculation, and recalibrate the output volume of the audio. The Bluetooth range must work between 0 – 31 meters radius on the system. The volume range should be a minimum of 15% to 100% of the maximum system volume and have a signal strength value input rate of less than one second, to ensure the volume changes in real time. The final design reached most of the requirements, successfully achieving the goal requirements. Using a Raspberry Pi3 with Bluetooth capabilities and connecting it to an amplifier and two large speakers using an AUX cord, the system successfully connects to a portable device using Bluetooth, reads in the RSSI value, takes the average of several iteration, sets the volume on a scale of 15% to 100% of the maximum device volume, calibrates the system volume, and outputs the audio with the calibrated volume. This resulted in the volume of the audio changing according to the distance of the portable device from the speakers.

Acknowledgements

Thanks to Professor Hedrick for advising me on this project. My sincere gratitude would not be enough to express how thankful I am for your teaching and support during these past terms. Thank you for helping me through guiding my research of this project, as well as providing the necessary components and equipment.

Thanks to Xavier Theo Quinn for assisting me throughout this process with new ideas, coding help, and troubleshooting. I would not have been able to finish this project without your help. Thank you for your time and patience throughout this process in addition to your busy schedule.

Thanks to the Electrical and Computer Engineering Department for teaching me core skills that helped me in developing this project and for supporting the process financially.

Table of Contents

List of Figures

List of Tables

Introduction

With the rise of automation and wireless technology, there are so many different possibilities to create potential future technology with these fundamentals. I want to utilize my curiosity and interest in automation and wireless signals, along with my background in audio and music, to create a device that will automatically adjust volume according to received signal strength using Bluetooth by applying my skills and knowledge from my undergraduate education in electrical engineering, as well as nontechnical skills from my liberal arts education.

Bluetooth has become a big part of modern technology, allowing wireless connection between devices. This provides a convenient connection between devices without having to plug in a cord. Anyone with a Bluetooth-enabled device can connect to another device, so long as one or both devices aren't already connected to another device. With wireless technology becoming the norm in modern technology, I have yet to explore the different uses I can have with Bluetooth and its functionality. Some typical examples of Bluetooth include connecting a phone to a wireless headset to take a call while driving on the highway. Another example is when connecting a mobile device to wireless speakers via Bluetooth so that the user can still carry the mobile device around an area and still use the speaker to play music. Though when the user walks around into different rooms or just further away from the speakers, the sound quality and volume is subject to changes. In response, a Bluetooth speaker that improves the auditory experience by optimizing volume for indeterminate listener locations is one solution.

One problem with speakers is that it usually has a fixed volume unless you change it manually via remote or connected device. And generally, the sound is heard at a lower volume the further one is from the sound source. In the situation where the user is walking around an area while playing music on the speakers, the volume will vary depending on where the user is, and the sound quality is probably changing if the user is walking into different rooms. Bluetooth is generally used in short ranges, so if the user goes beyond the radius of the Bluetooth range, the device is no longer connected to the fixed device. Some problems that would be addressed are how to collect and/or measure the distance and position of the speakers and the connected device in real time, as well as automatically controlling the volume using that data.

The goal of this project is to build a Bluetooth-enabled speaker that will use the Bluetooth signal strength of the connected device to the speaker to control the volume of the speaker. The overall resulting goal is having the user listen to the audio at the same volume regardless of the distance from the speaker. Distance and location detection will not be necessary due to the usage of the RSSi value, or the signal strength. So, using the received signal strength, this will determine how far the mobile device is from the speakers. This report will be covering more in depth background information about the inner workings of the project, design requirements of the final product, design alternative for different aspects of the project, the design, some preliminary testing results found over the course of the term, and an implementation schedule for the next term.

Background

Previous Work

Bluetooth is short-range, wireless communication used to send information between devices. The frequency at which Bluetooth operates is around 2.4 GHz, at which frequency hopping is used with the transceiver to reduce interference and fading of the signal. Frequency hopping is when the transceiver rapidly sends the signal among different frequencies. Researchers and engineers have been experimenting with the limits of Bluetooth technology. With much of these articles dating after 2010, these are recent developments. After deciding on my project goal, I researched how to make it happen. Here are some articles of research that has been done exploring the use of Bluetooth received signal strength indicator.

Gowda (Gowda, 2012), of the University of Utah, performed experiments measuring the RSSI values using Bluetooth interface for secret key extraction. Due to the degradation of Wi-Fi, Gowda proposes that Bluetooth would be more efficient in it's adaptive frequency hopping technique of detecting and avoiding bad signals and interferences. Gowda needed to design a protocol to send and received L2CAP packets between Bluetooth-connected devices so that the RSSI value can be measured from each packet. This thesis was published in 2012, so the methods of measuring the RSSI value were different then. The experiment was also performed in both outside and inside environments to understand the difference of the efficiency of

Bluetooth, in which the outdoor setting was more efficient in the resulting secret bit rates than the indoor setting. The thesis concluded that the secret bit rates received with the Bluetooth and Wi-Fi were similar in similar settings and conditions.

Rozum and Sebesta (Rozum & Sebesta, 2019) used the Bluetooth RSSI value as an input for location estimation in Bluetooth Low Energy Indoor Positioning System. The RSSI value was used as a reference for location and the fading method was used to reduce instability in the value so that the reference remains as accurate as possible. By using four antennas to measure difference distances, the back-side reflector was used to obtain the quarter wavelength additional arrangement. They determined that, though with the reflector the antenna gain was greater, without the reflector reception was subpar. So, if the target's location had to be found or estimated through walls, the received signal could not be accurate regardless of how precise the calibration is.

Subhan, Hasbullah, Rozyyev, and Bakhsh (Subhan, Hasbullah, Rozyyev, & Bakhsh, 2011) presented a Bluetooth handover concept that measures and uses the Bluetooth RSSI, Link Quality, Transmitted Power Link, and Received Power values as parameters to estimate location, such as the previous research. Their research includes a lot of information on the date rate, range, and frequency of the Bluetooth network, as well as understanding the specifications within Bluetooth protocols. For their research, due to the issue of handover in Bluetooth, Bluetooth RSSI, Link Quality, Transmitted power Link, and Received Power were used as variables to test different techniques to using Bluetooth in positioning. They concluded that Received Power could be used for distance estimation and indoor positioning.

With the conclusions collected from these research findings, I can gather the necessary information to understand the theory behind my project and how to use these experiences to make sure my project performs as accurately as possible. For example, Gowda's thesis in using Bluetooth instead of Wi-Fi will help me understand the usage of the Bluetooth RSSI value and the fluctuations it has in an indoor setting. While Rozum and Sebesta gives proof that position estimation with RSSI is possible and will need some work into keeping the accuracy through walls. And Subhan, Hasbullah, Rozyyev, and Bakhsh provide vital information in understanding different parameters within a Bluetooth signal that can be used to measure distance.

Impacting Considerations

With these articles in mind, further research was required to understand the limitations of Bluetooth and how to manipulate the thresholds to develop the design requirements for this project. Such as the different classes of Bluetooth and the power at which the device transmits at a maximum range, as shown in Table 1. This is so that the project should be able to have enough power to transmit signals over a large enough range.

Table 1: Bluetooth Power Classes

Device Class	Transmit Power	Intended Range
Class 3	1 mW	Less than 10 meters
Class 2	2.5 mW	10 meters
Class 1	100 mW	100 meters

Another important consideration is how the Bluetooth RSSI value changes according to distance. Though the device is not actually calculating the distance value, the RSSI value is the

main input parameter of the project and does change according to the distance, as shown in Figure 1. The RSSI value in relation to distance can be found using this equation:

$$RSSI = -(10*n*\log10(d) + A)$$

Where *RSSI* is the received RSSI value, *n* is the path-loss exponent, *d* is the distance, and *A* is the RSSI value at the initial distance set (Lau & Chung, 2007). In terms of this project, the setting will be indoor residential area with a path-loss exponent value of about 2.8. The reference value is generally taken as the RSSI value received when the Bluetooth receiver is 1 meter from the transmitter. This will be taken into account during theoretical calculations and testing.

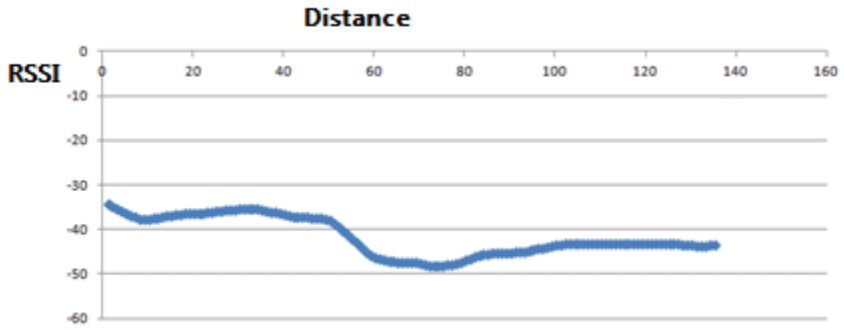

Figure 1: Change in RSSI value by Distance

Other considerations outside of the contents of the articles include usability, marketability, manufacturability, etc. This project could potentially be a marketable product in terms of the setting that it's intended to be used. The project would need to be as hands off as possible to warrant the claim that it's automatic, but accessible in the manual inputs to allow the sound to be customizable. The project should also be compliant to ethical codes and standards, as well as standard policies and regulations to ensure the safety of the user. These considerations would be applied to the range of sound that the product would allow due to the

decibel maximum before sound starts to damage hearing. Using these considerations, I compiled a detailed set of design requirements that will serve as a basis for the project.

Design Requirements

The Automated Volume Adjusting Bluetooth Speaker in a broad sense, is meant to connect to a mobile device via Bluetooth and adjust the volume so that the user will listen to the audio output at a constant volume as if the user was standing next to stationary speakers. So, the device will receive an input of the RSSI value with the code, which will in turn plug it into an equation that will proportionally change the volume value. This volume value will affect the audio amplifier to change the volume of the audio output of the speakers. This volume change will maintain the perceived auditory volume of the user at a constant level.

The intended setting of this device will be between a 5.5 x 7.9-meter room and a 5.9 x 12.5-meter apartment. The location where I have tested this device was in a 5.5 x 7.9-meter room, while the location where I originally intended to use this device was in a 5.9 x 12.5-meter apartment. The size of the room needed to be large enough to test the change in decibels from the stationary speakers. The apartment emulates a typical apartment with three rooms, one bathroom, and an open space with kitchen and living room. With this setting, testing can be done through multiple rooms and spaces for sound quality and volume adjustment with walls.

The Bluetooth range that the device must work within 23 to 31 meters due to the path-loss in a typical home (What is the range of Bluetooth?, 2019). With the Raspberry Pi Zero W,

using BLE 4.1, which achieves an operating range of over 30 meters, this is compliant to the range specification. This ensures that the Bluetooth connection will stay strong and consistent within the intended setting, while having the potential of continuing service outside of the setting.

The initial calibration of the device must be taken 1 meter from the transmitter in order to obtain the received RSSI value at a fixed distance from the speaker. Thus, the user must connect the mobile device to the transmitter via Bluetooth and set the volume at the desired loudness 1 meter from the transmitter. Then the device will use the RSSI value from that distance as a reference RSSI value to keep the desired loudness proportional to the change in RSSI value. This reference RSSI value is subject to reset when the device is shut down so the user will need to recalibrate each time when powering on the device.

The proportion at which the volume value changes in relation to the RSSI value will be logarithmic. During testing the change in decibel readings was logarithmic in relation to distance when the volume was constant, and the distance changed. So, in response to the data, the volume value would change logarithmically according to the change in RSSI value.

Wall detection is necessary in the code due to the intended setting and will be detected with the slight drop or increase in RSSI value when the signal passes through a wall. The typical path-loss exponent in an indoor setting with walls is 2.8, so that will be taken into account within the code. This will allow for the volume value to be boosted or diminished to accommodate the soundproof of the wall or lack thereof.

The device will take into consideration the sound safety limits for hearing. Due to noise danger, there is a decibel limit before the sound can cause hearing loss. Typically, people listen to music and audio from a speaker at around 70 dBA or less (Loud Noise Dangers, 2019). But hearing loss can be caused by listening at 85 dBA for long periods of time and damage accelerates as the noise level increases. So, the device will stop increasing the volume once it reaches 80 dB regardless of the increase in RSSI value from that point.

The expected market price of the device is at least $25 but a manufacturing price of no more than $40. This is due to the use of the Raspberry Pi Zero W, which has a market price of $10, along with power cords and miscellaneous wires and connecters. The speakers and audio amplifiers will be connected to the Raspberry Pi Zero W with equipment resembling an L-pad. This way, the device can be fairly cheap to manufacture and depending on the quality of the speakers, can be marketed at a reasonable price.

The weight of the device itself should be less than a pound, including the casing. This is due to the number of components that would be needed to be connected to the speakers. The size of the device should be no larger than a 10 x 10 x 10 cm cube, the small size is to save space, using only needed space for the device components. The casing should also be small to add aesthetic to the entire sound system, with the use of large speakers to emphasize the amount of power needed to project the sound as needed by the user.

This device will have to comply with core Bluetooth standards. Such standards include IEEE 802.15.1 (IEEE 802.15 WPAN Task Group 1 (TG1), 2019), which includes clauses on SAPs, normative annex providing a Protocol Implementation Conformance Statement proforma, and

informative high level behavioral ITU-T Z.100 specification and description language model for integrated Bluetooth MAC sublayer. Since Bluetooth is a short range RF based connectivity for portable devices, this standard provides an adaptation of the Bluetooth Specification v1.1 Foundation MAC.

Design Alternatives

This project consists of two main issues that are highlighted in the block diagram in Figure 2. One such issue is the input parameter used to measure the distance of the receiver from the transmitter. There are many different alternatives to measuring the physical distance that may require a sensor and others that require receiving a signal. Some examples of each that were considered will be discussed. Another issue is the method of physically controlling the volume. This takes the type of speaker and control panel the device will function with into consideration. For the final design, the controller can alternate between a Raspberry Pi3 or Raspberry Pi Zero W, in which the Pi Zero W would require an external headphone jack dongle. The audio amplifier and output can be any typical home sound system with AUX input. There are advantages, as well as disadvantages, for each solution, but the method that I have chosen for the project will have taken simplicity into account.

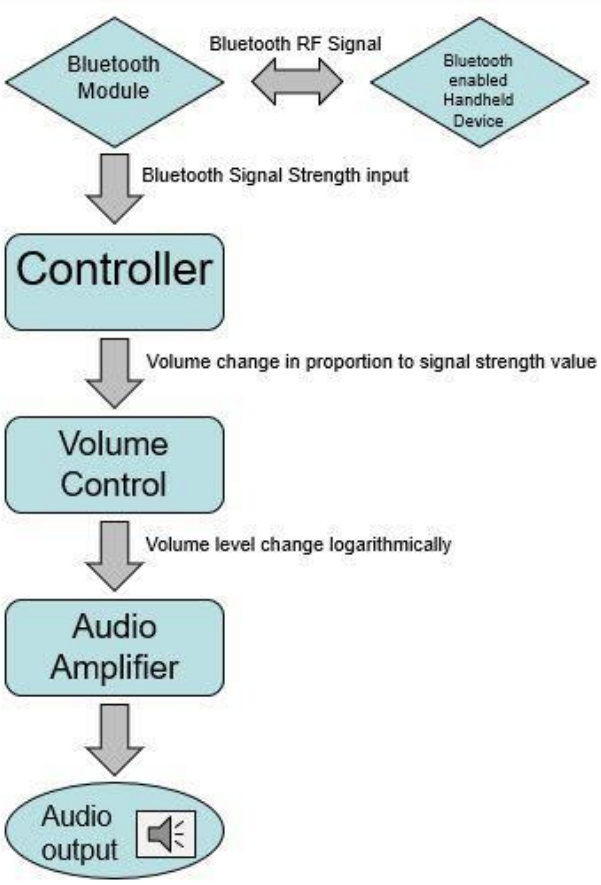

Figure 2: Overall Block Diagram

Input Parameter

There are many ways of measuring distance, one such way is the use of sensors. Though there are many types of sensors, they measure physical attributes to measure the distance from the sensor. Echolocation was the most basic form of distance detection using high frequencies, though would be ineffective with a moving target and obstacles. LED sensors are inexpensive with multiple interface options and good update rate, but it would have been power consuming with low range. LIDAR sensors have a large range and fast update rate but is also very power consuming and expensive. Ultrasonic sensors have low power consumption and multiple interface options but would have a slow update rate and low resolution. And

VCSEL sensors have a small minimum range, large input voltage range, good resolution and is inexpensive, but there is only one interface with a low maximum range (Distance Sensing Overview, 2019).

With Bluetooth connection, there are several parameters that could be taken from the Bluetooth signal alone that could estimate distance without a distance value input. In Subhan, Hasbullah, Rozyyev, and Bakhsh's research (Subhan, Hasbullah, Rozyyev, & Bakhsh, 2011), they tested different signal parameters: Received Signal Strength Indicator (RSSI), Link Quality (LQ), Transmitted Power Link (TPL) and Received Power (RX). The Link Quality and Transmitted Power Link parameters are device and manufacturer specific so the input value would have to be specific to the components used in the project. To allow for component flexibility, these parameters will not be used as input. Received Power is indirectly related to the received RSSI value and would need to be converted using the radio propagation model. But the RSSI value is directly proportional to the distance of the receiver from the transmitter, and with a low update rate, the RSSI value can be received as close to real time as possible.

In RSSI reception, an option for input precision is the Ubertooth, which is an external dongle that can be connected to the Raspberry Pi. This device is a passive receiver for locating wireless devices. With certain functions, this device would be used to receive the RSSI value at a more accurate and quicker rate than other methods, such as command line or python. But due to the lack of information provided about the device and lack of experience of usage, the time limit of this project makes it difficult to implement this device into this project with the low chance of completing it on time. The RSSI input would be later changed to using the Raspberry

Pi capabilities in the program, which does not have the most precise input as the Ubertooth but functions well enough for the purpose of this project.

Volume Control

Methods for volume control is dependent on the type of speaker used during the project. One considered method of volume control is another Bluetooth connection from the device to the speakers, but multiple Bluetooth connections to a single device may result in complications. There is also no physical manual override system for the user in a case of need. A form of volume control would be a direct connection between the controller to the audio amplifiers and speakers by internal wires or hardware. This can potentially be done with an L-pad, as shown in Figure 3. But the L-Pad has a physical dial, much like a potentiometer, so that device may have to include a motor controller and motor to attach to the dial. Adding a motor would require research in how much torque is needed to accurately turn the dial on the L-Pad. This would also be the case if the speakers had physical dials built into the system to control the volume. But seeing as a manual control override system is necessary for usability, having a physical control panel system would be needed. Though the control system would solely be connected to this motor system but would not have the audio output connected. This would not work in accordance to the design requirements of audio output. These concerns were not anticipated before the preliminary design. Another form of volume control is directly using command line functions directly with the Raspberry Pi. These individual functions can be called and do the tasks of receiving the RSSI value, doing calculations, and changing the system output volume, but cannot do these continuously together. The design for volume control that was

chosen for the final design was directly connecting the Raspberry Pi to the speakers using an AUX cord and controlling the volume through the program.

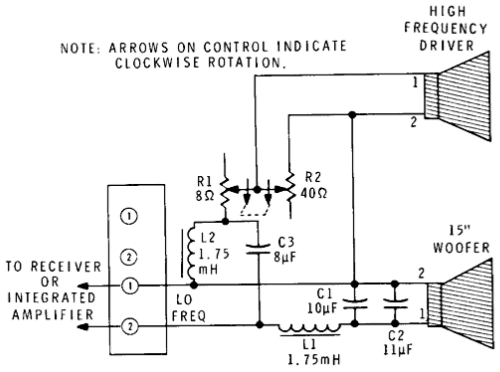

Figure 3: Schematic of Wiring for L-Pad

Preliminary Proposed Design

According to the general block diagram in Figure 2, the first level of the Automatic Volume Adjusting Bluetooth Speaker is the Bluetooth aspect of the system. This is the connection between the controller, the Raspberry Pi Zero W, and a mobile device, most likely a phone with Bluetooth functionality. The Raspberry Pi Zero W uses Bluetooth 4.1/BLE and follows standard Bluetooth protocols, such as using a 2.4 GHz band for Bluetooth and Wi-Fi communication. It also includes the IEEE 802.11n specification, which entails that the Raspberry Pi Zero W can operate at a maximum throughput of at least 100 Mb/s, measured at the MAC data service access point (IEEE 802.11n-2009, 2009). Once the Raspberry Pi Zero W has a Bluetooth connection to the phone, the Pi Zero can use the signal to obtain the RSSI value.

The second level of the device is the software aspect of the system, which relies on functions within the code to calculate the volume value from the RSSI value. Though the full detailed calculation for the volume value in proportion to the RSSI is in progress, below in figure 4 is the generalized pseudocode for the system. There are packages that can be downloaded to Linux to enable Bluetooth capabilities in the Raspberry Pi Zero W. Once the Pi Zero is connected to the phone. There are functions to obtain the RSSI value from the Bluetooth signal. In starting the main program, and to calibrate the initial system settings, one specification is for the user to connect to the system from a distance of 1 meter from the device. Once connected, the program will assume that the device is connected with an initial distance of 1 meter at the start of the program and use the RSSI function to obtain and store the RSSI value from that distance. The user will set the volume of the phone manually according to the volume the user would like

as the constant perceived volume. The control function will perform real time calculation and output the volume value according the RSSI input. The control function consists of a loop that will take the input RSSI value, taking into account if there is significant jump or drop in input stream as wall detection and updating the loss value accordingly. Within the control function is the volume function, which includes the actual calculation of the volume value using the real time RSSI value and loss value adjustment. There will be a maximum volume limit for safety which will contain the output decibel level of the system within safety auditory range to prevent hearing damage. So if the calculated volume value is greater than the maximum volume limit, the maximum volume limit will be the output value to the volume control, otherwise the calculated volume value is outputted. Bluetooth signal disconnection will also be taken into account. If the connection to the mobile device is disconnected, the system will set a delay timer where if the phone is reconnected within the delay time range, the program will continue the control function loop with the same calibrations. Otherwise, if the phone is not reconnected by the end of the delay time range, the system will restart the calibration and remain idle and set another delay timer. If the phone is reconnected during that delay time range, the system has already reset and will need to be recalibrated, but if the phone remained unconnected, the system will shut down after the delay time range.

```
Main() - main program
    Connect to mobile device with Bluetooth
    Calibrate() - initial calibration of RSSI value and volume setting
        RSSI() - Function to obtain RSSI value
        Initial RSSI value = RSSI value at 1 meter from transmitter
        Distance = 1 meter

    Control() - real time calculation of volume value using the RSSI value for volume control
        Loop
            RSSI()
            If: RSSI value has a significant jump/drop
                Then: Add/subtract loss value
            Else: nothing
            Volume()
                Function that takes the RSSI value and loss value to calculate the volume value
            If: volume value is greater than max volume limit
                Then: Output max volume limit
            Else: Output volume value to volume control
            Delay
        End

    If: connection to mobile device is disconnected
        Delay
        If: reconnected
            Then: Control()
        Else: restart system
            Delay
            If: reconnected
                Then: Main()
            Else: System shut down
End
```

Figure 4: General Pseudocode

The third level of the project is the physical components of the system, which is shown in Figure 5. The Raspberry Pi Zero W is connected to the phone wirelessly using Bluetooth signal and is connected to the volume control system. The pin connections of the Pi Zero is found in Figure 6, which includes power connections and output pins. There are input ports for a micro SD card, which this system will be using a 32 GB SD card, and power supply, with a 5.1 V micro USB switching AC power supply. The volume control system will consist of a motor controller connected to a motor that will attach to the L-Pad, which is connected to the amplifier, which is also yet to be decided, and the speaker. This volume control system would either physically change the volume dial or send signals directly to the amplifier to change the audio output of the speakers. The full components list is found in Table 2, this list will be updated over time as official components are selected.

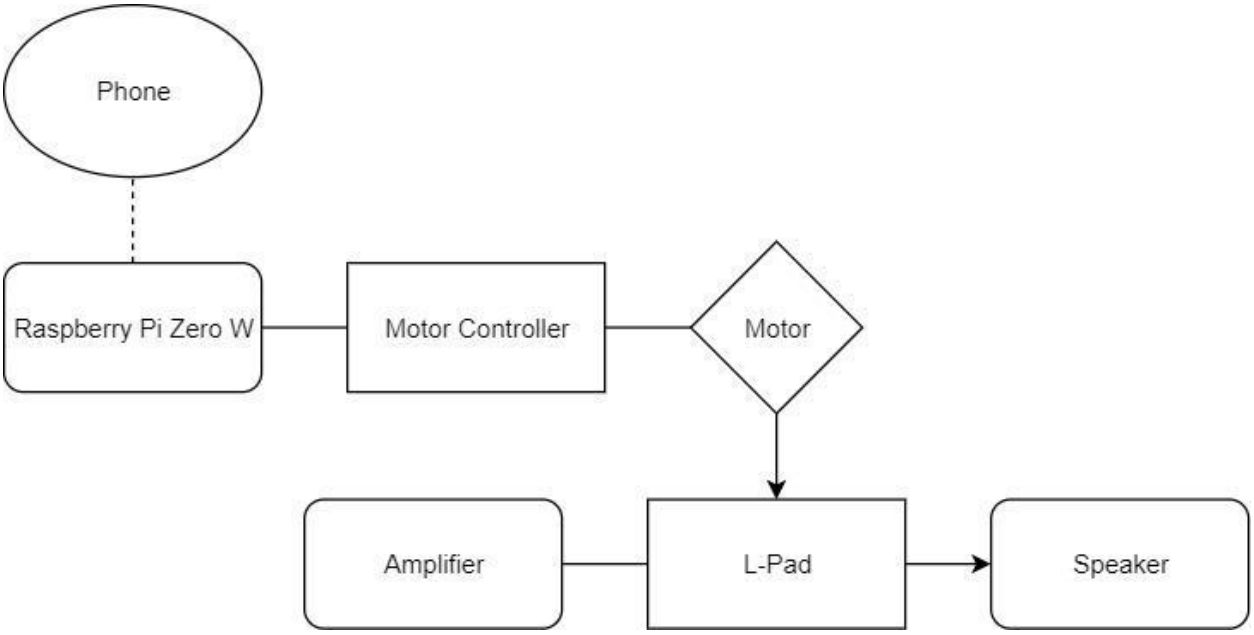

Figure 5: General Physical Block Diagram

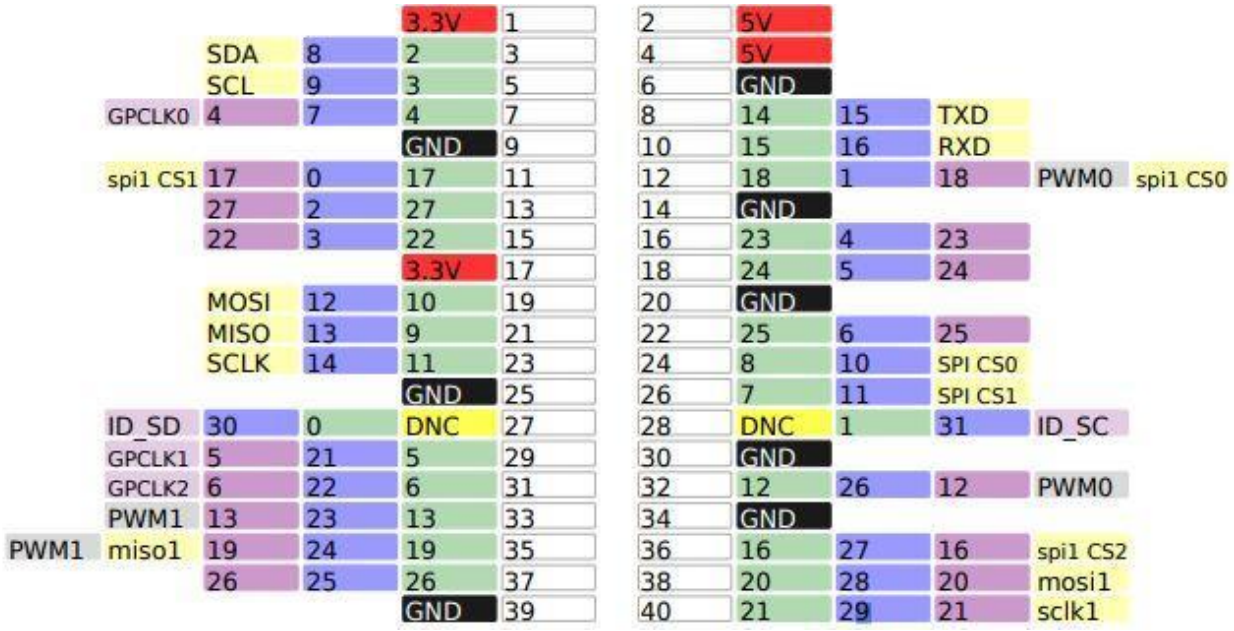

Figure 6: Raspberry Pi Zero W Pin List

Table 2: Preliminary Parts List

Component	Quantity
Raspberry Pi Zero W	1
Amplifier	2

Speaker	2
Micro SD Card – 32 GB	1
Pro-Elec 5.1 V/2.5 A Micro USB Switching AC Power Supply	1
L-Pad	1
Motor	1
Motor Controller	1
Miscellaneous wires	≥ 8

The final level of the system is the audio output of the speakers or the resulting volume level output from the RSSI input value in real time. The perceived auditory output of the system should be constant to the volume set by the user during calibration. With the detection of a wall, the volume should boost slightly to adjust to the sound proofing of the wall. Then detection of straight line space would decrease the volume slightly with the lack of wall.

Final Design and Implementation

Overall System Design

The final product of this project is Bluetooth speakers, that is Bluetooth-enabled by the control system connected to the audio output system, that uses the signal strength of the connected portable device to the speakers to control the volume output. This allows the user to listen to the audio at a constant volume regardless of the distance from the speakers. The final system design, though differed from the preliminary design, kept the core features of the original system block diagram, which is Bluetooth connect, RSSI reception, volume adjustment, and audio output. These core features rely on both the hardware and software of the system in order to function. This section will focus on both of these aspects of the system design and give more detail as to the purpose and function of each section. The final block diagram, in Figure 7, shows the different sections I will explain in more detail. The portable device block, control system block, and the audio output block are the three main sections of the system.

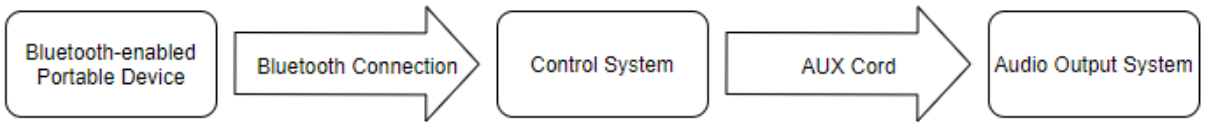

Figure 7: Final System Block Diagram

Hardware System Design

The idea for the hardware design is for the system to be simple for the user to navigate and use in everyday life. As mentioned in the block diagram in Figure 7, the physical system, in

Figure 8, is exactly as follows: a Bluetooth-enabled portable device connected to the control system using Bluetooth and the control system connected to the audio output system using an AUX cord.

Figure 8: Final Hardware System

The first block in the block diagram is the Bluetooth-enabled portable device. This block was designed for any portable devices that has Bluetooth capabilities and are commonly used for playing music. Such examples may include smartphones, tablets, laptops, music players, and smartwatches. For the sake of experimentation, I used a phone due to its availability to me, and its ability to connect to the Raspberry Pi through Bluetooth and play music. Overall, the device used for this block is interchangeable with any of the examples mentioned.

The hardware for the control system block is mainly the Raspberry Pi. The Raspberry Pi is able to use Bluetooth and use functions in different languages. For this project, I chose to use the Raspberry Pi3, due to the headphone jack in the hardware and Bluetooth capabilities. I did

consider the Raspberry Pi Zero W because it was much smaller and seemed visually inconspicuous, which would add to the aesthetics of the overall system. Unfortunately, the system design required an AUX cord to connect to the audio output system directly from the Raspberry Pi and I did not have the time nor funds to procure an external headphone jack, so the Raspberry Pi3 seemed like the more convenient choice due to immediate availability. Much of the control system is the RSSI reception and volume calculation, which is mainly software.

The audio output system block is meant to be a typical home audio system with an amplifier with AUX input and two large, high powered speakers. This makes this design accessible to the normal civilian who has a typical home audio system and so the product the user would purchase would just be the control system itself. As long as the audio system has an AUX input, the project is compatible. The audio output system will output the desired audio from the portable device at the desired volume and will continue to output the audio at the volume adjusted according to the signal strength.

Software System Design

The software system design is entirely within the control system, which follows the main function of the code, in Figure 9. The main function is a continuous while loop, so it would run for as long as the system is powered on. The loop will connect to the desired Bluetooth MAC address of the portable device and receive the RSSI value. The RSSI value that the calculation takes is the running average value of ten iterations of the RSSI value. The volume calculation then takes this running average and put it on a scale between zero and the maximum RSSI,

which then computes the proportional volume value on a scale of a set minimum volume and 100. The calculated volume is then set to the system volume, which is then used in the output.

```
while True:
    try :
        btrssi = BluetoothRSSI(BT_ADDR)
        new_rssi = abs(btrssi.get_rssi())
        av_rssi = running_average(average_list, average_length, new_rssi)
        vol = int(translate(av_rssi, 0, max_RSSI, min_vol, 100))
        m.setvolume(vol) #Set the volume to the calculated vol%
        time.sleep(1)

    except KeyboardInterrupt :
        break
    except :
        print("please connect device")
```

Figure 9: Main Loop

There are many different functions used within the code and many of them need external sources to be imported for them to work, which are Bluetooth RSSI, Bluetooth, time, sys, alsaaudio, math, os, and subprocess.

In order to get the RSSI value of a certain device using Bluetooth, the function 'BluetoothRSSI' requires the Bluetooth MAC address of the desired device. For the purpose of this project, the absolute value of the RSSI value is taken using the function 'abs', and the running average is calculated using the function "running_average', in Figure 10, which uses an input of an average list array, a set average length, and the absolute value of the input RSSI value.

```
#calculates running average provided an averaging list, length of list, and new value
def running_average(arr, length, new_val) :
    #add new value to end
    arr.append(new_val)
    #If the list is at above length
    if len(arr)>length :
        #remove oldest value
        arr.pop(0)
        return sum(arr)/length
    return new_val
```

Figure 10: Running Average Function

The next function 'translate', in Figure 11, uses the average RSSI value and the set minimum and maximum values of the volume and RSSI ranges. So using the input average RSSI value, this function would scale the volume value accordingly. Then it would use the scaled volume value and set it to the system output volume with the function 'm.setvolume'. The system would then wait one second before it repeats this process again.

```
#scales the range to the volume
def translate(value, from_low, from_high, to_low, to_high):
    ratio=float(value)/float((from_high-from_low))
    to_range=to_high-to_low
    scaled_val=ratio*to_range + to_low
    if scaled_val<to_low :
        print "Value too low"
        return to_low
    elif scaled_val>to_high :
        print "Value too high"
        return to_high
    return scaled_val
```

Figure 11: Translate Function

Outside of this loop, by running this program, the script will also run a command that will allow the output of audio from the Bluetooth connected device, as well as open access to the system volume control with the function 'alsaaudio.Mixer'. So overall, the program will include most of the implementation of the project so that the user will not need prior knowledge of programming in order to use the device.

Performance estimates and results

Estimated performance

In the preliminary design, it was expected that the Raspberry Pi would be able to connect to the portable device with an initial volume set at one meter and be able to receive continuous RSSI data. This RSSI stream would act in accordance to the distance between the Raspberry Pi3 and the portable device. Using this RSSI data, the program in the Raspberry Pi3 would be able to linearly calculate the distance in meters, as shown in Figure 1, and in turn, calculate the volume at which the perceived volume is the same as the initial volume set at one meter. This calculated volume is logarithmically proportional to the distance. Then the system will use the calculated volume to set the speaker volume. This project was designed for a standard single person apartment where the speaker system is in a static location. A manual override feature was to be implemented into the system, in which the user would be able to input the rate at which the RSSI value is to be input and the volume at which the system is outputting at any given time. There would also be a volume safety limit to ensure the volume would not increase beyond the maximum volume limit at which would cause hearing damage. The most difficult expected implementation was wall detection and adjustment. This feature would use slight increases or decreases in the RSSI stream that would indicate if the user was or was not behind a wall, so that the system can adjust the volume accordingly. The system would also have an automatic shut off feature that would detect if the user was still using the device or away from the apartment by setting a timer for when audio is not being streamed and if the Bluetooth is disconnected.

Resulting performance

Though not every expected feature was implemented or functioned exactly as designed, the project proved successful in showing concept. The original set of specifications include: functioning in a 5.5 x 12.5-meter space, Bluetooth range working up to 31 meters away from the device, initial calibration from one meter of the transmitter, logarithmic proportional calculation of the volume from the RSSI value, wall detection and recalibration, sound safety consideration, low manufacturing price, and light weight of device. The core ideas of this project, which was the reception of RSSI data and the volume adjustment of the audio output in accordance to the change in RSSI data, was successful. During testing, I added some code in the program to print out the RSSI value and calculated volume in real time, moving my phone throughout the room. Though the RSSI data input was not as precise as I would've liked, there was still a constant data input that did change according to distance. The volume calculation was implemented in the code using a scale function in Python so that the volume would be on a scale of 40 to 100 proportional to the RSSI value from 0 to 30. The output system volume would then be set to the calculated volume. This is shown by connecting the Raspberry Pi3 to a portable device, playing audio, and moving around the area. The volume of the audio becomes louder as the user moves away from the speakers and becomes softer as the user moves closer.

These results may not be as ideal as the estimated performance and the current project may not be able to sell as a product in the market, but it shows proof of concept in that the RSSI value can be used as a distance indicator and wireless signal can be used more than connecting devices. Many of the features in the estimated performance were not implemented due to lack

of time and prioritizing core features. Despite not being able to implement these features, such as wall detection and logarithmic calculation, they may serve as goals for future work. This would require minimal design changes to the basic system block diagram and merely additions to the initial work already done. One such suggestion would be the implementation of the Ubertooth. The Ubertooth would be able to improve the accuracy and precision of RSSI data input. The difficulty in using the Ubertooth is the lack of information publicly available in usage for the requirements of this project. Much of the testing of the Ubertooth was trial and error, most of the time resulting in a dead end. So, with enough time, more research and testing can be done with the Ubertooth to allow a consistent stream of RSSI data of the desired device. Another suggestion that will greatly improve the accuracy of keeping a constant perceived volume is changing the way the volume is calculated. Currently the volume is changing linearly with the RSSI value but the volume should be changing logarithmically. There is more research to be done for the math of the calculation and how sound works in free space and through walls. So, if those two suggestions were successfully implemented into the system, the performance would improve in accuracy into a project I had originally imagined.

Production Schedule

Much of my schedule during the production of my project was experimenting with different methods of implementing the functions of the block diagram. During fall term, I researched background information on the premise of my project in Bluetooth and sound levels. This research and testing contributed to the volume calculation in the code. Most of the

project building occurred during winter term, which did not follow the previous anticipated schedule.

The beginning of my winter term started with reviewing my research over the fall term and extra research over the break. Since I was trying to implement the motor design idea, I ordered and received a stereo L-pad, motor, and motor driver. The short-term goal at the time was trying to get the RSSI values to output on the Raspberry Pi. So, I compiled command line functions and matched them to my pseudocode. Then I set up the Raspberry Pi to enable Bluetooth and connect to my phone, in which I implemented the command line functions to receive RSSI values from the connected phone. The RSSI values were not what I expected and could not be used in the volume formula that I had originally planned. Another function I implemented in command line was to get the audio to play, in which it did. Though, because these functions worked independent from each other and I needed them to work together continuously, I learned Python so that I can just run one continuous script. Later I found out the Raspberry Pi could only be used for moving the motor and not output audio, so I researched an alternative hardware design. I learned that an AUX cable could be connected from the headphone jack of the Raspberry Pi3 and then connected to a speaker system with an AUX input. Towards the middle of the term, I learned of an external device called the Ubertooth, that can output precise RSSI measurements. I ordered the Ubertooth, connected it to the Raspberry Pi, and started researching Python functions and testing it. Though I had made much progress for the next two weeks on learning how to utilize the Ubertooth for my project, the resulting RSSI output was inconsistent and unstable. With the little time I had left in the term, I decided to not rely on the Ubertooth and instead use functions in Python to receive RSSI values,

though they were not as precise as the Ubertooth. I then created a continuously running script in Python that: connects to a device, plays audio from the device, receives the RSSI value, uses the running average of the RSSI value, calculates the volume to scale proportionally to the RSSI value, and changes the system output volume. After more testing for accuracy, the project was complete and ran in accordance to the project description.

Cost Analysis

Table 3: Cost Analysis

QTY	Product	Vendor	Unit Price	Total Price with Shipping
2	Speaker L-Pad Attenuator 50W Mono 1" Shaft 8 Ohm	Parts Express https://www.parts-express.com/parts-express-speaker-l-pad-attenuator-50w-mono-1-shaft-8-ohm--260-255	$11.48	$58.82
2	Raspberry Pi Zero W (with Headers)	Sparkfun https://www.sparkfun.com/products/15470	$14.00	$62.49
1	Wall Adapter Power Supply - 5.1V DC 2.5A (USB Micro-B)	Sparkfun https://www.sparkfun.com/products/13831	$7.95	$7.95
2	microSD Card with Adapter - 32GB (Class 10)	Sparkfun https://www.sparkfun.com/products/14832	$24.95	$49.90
1	Arduino Mega 2560 R3	Sparkfun https://www.sparkfun.com/products/11061	$38.95	$38.95
2	Bluetooth Mate 4.0 - HM-13	SparkFun (https://www.sparkfun.com/products/14839)	$19.95	$56.08
1	JustBoom DAC HAT	Sparkfun https://www.sparkfun.com/products/14319	$39.95	$39.95
1	Audio Cable TRRS - 3ft	Sparkfun https://www.sparkfun.com/products/14164	$2.50	$2.50
			Total:	$316.64

User's Manual

This manual provides information on the set up, operation, and maintenance of the system. All necessary programs and files are already on the Raspberry Pi3.

Set up

1. Connect a keyboard, mouse, ethernet cable, and monitor to Raspberry Pi3
2. Plug the power adapter into an outlet and into the Raspberry Pi3
3. Turn on the monitor and wait for start up to complete
4. Open the files icon and open 'senior_project.py'
5. Input your device's Bluetooth MAC address after 'my_address'
6. Click save
7. Open a command prompt and type 'python senior_project.py'
8. Connect the AUX cable to the Raspberry Pi3's headphone jack and to the amplifier of your sound system

Operation

1. Turn on Bluetooth on your portable device
2. Pair with 'raspberrypi'
3. Play music

Maintenance

1. For troubleshooting, run 'python senior_project.py' in the command prompt
2. Walk to the furthest distance from the sound system and back towards the sound system

3. Make sure the volume is increasing as you walk away
4. If it's not, type 'Ctrl C' to stop the program and the up key and enter to rerun the program

Discussion, Conclusions, and Recommendations

Problem

With my interest in wireless signals and audio engineering, I wanted to find a project that would be both fun and practical. I wanted to be able to explore the limitations of Bluetooth and be able to create a potential product that may sell in the market. From personal experience, walking around an apartment floor while playing music from speakers in my room connected to my phone with Bluetooth, I couldn't hear the sound clearly from far away or in other rooms.

Approach

To address this problem, while creating a challenging but fun project, I decided to find a way to automatically adjust the volume of the speakers to match the location of the user using the signal strength of the Bluetooth connection. To do so, I would need a controller module that would be able to connect to other devices using Bluetooth and is programmable. As a result, I found the Raspberry Pi to be the most compatible with the requirements of my project. The three main components of the hardware design is the portable device, which would connect using Bluetooth and play audio, the controller module, which takes the RSSI value and computes the volume in real time, and the audio output system, which is any typical home audio system with an amplifier and two high powered speakers. After much experimentation with each of these core components, the final design was settled with the Raspberry Pi3 as the controller module which would connect to the audio output system with an AUX cable.

Performance

The final system met the core requirements of connecting to a portable device using Bluetooth, receiving the RSSI value, calculating the new volume according to the RSSI value, and outputting the audio playing from the portable device at the new volume. With minor adjustments to the RSSI value to improve accuracy, which is taking the absolute value and the average of several iterations, the calculation of the volume was a function to scale the volume value according to the RSSI scale. So that when the RSSI value increases, the volume will increase proportionally as well.

Recommendations

Though this system has met the basic requirements to function according to the core requirements, it is far from reaching its full potential of becoming a marketable product. I would consider this project to merely be the first of many prototypes. Improvements on the precision of RSSI reception can be made in the future, perhaps using Ubertooth. The volume calculation can also be changed to be more accurate logarithmically, as how sound changes in accordance to distance, instead of the current linear calculation. Though I did not implement this into my final design, wall detection and volume calibration were difficult features that may be used in future projects.

Lessons Learned

Several lessons were learned during these two terms of working on a single project from scratch. I learned to organize each phase of the project design, researching the most optimal path ahead of time. Much of my time was used on alternatives that were not used in the final design. Another lesson is realizing the time constraint is a consideration in attempting an alternative solution. For example, the Ubertooth was a solution to RSSI accuracy that I could not figure out in time to complete the final product due to the lack of experience and information on how it works, in which many problems arose during the experimentation process. Prior knowledge of the coding languages used during this process would have been helpful as well. The process was slowed due to having to learn how to code in Python and in command line, as well as learning Ubertooth commands later. Staying on task with the end goal was difficult during this process because suggestions for new solutions became prevalent and took too much time to implement and were not used in the end. But after long hours of research and testing, the project was finally complete, functioning as expected with physical results. I was rewarded with a working product, new programming languages, a deeper understanding of the design and testing process, and something to show off to friends and family.

References

Distance Sensing Overview. (2019). Retrieved from Sparkfun: https://www.sparkfun.com/distance_sensing

Gowda, P. L. (2012). *Exploring Bluetooth for received signal strength indicator-based secret key extraction.* School of Computing. Salt Lake City: Univ. Utah. Retrieved from https://pdfs.semanticscholar.org/4a85/489e4f075f2155864177a54b630060850b91.pdf

IEEE 802.11n-2009. (2009). Retrieved from IEEE Standards Association: https://standards.ieee.org/standard/802_11n-2009.html

IEEE 802.15 WPAN Task Group 1 (TG1). (2019). Retrieved from IEEE802.15: http://www.ieee802.org/15/pub/TG1.html

Lau, E.-E.-L., & Chung, W.-Y. (2007). Enhanced RSSI-Based Real-Time User Location Tracking System for Indoor and Outdoor Environment. *2007 International Conference on Convergence Information Technology*, (pp. 1213-1218). Gyeongju. doi:10.1109/ICCIT.2007.253

Loud Noise Dangers. (2019). Retrieved from ASHA American Speech-Language-Hearing Association: https://www.asha.org/public/hearing/Loud-Noise-Dangers/

Rozum, S., & Sebesta, J. (2019). Bluetooth Low Power Portable Indoor Positioning System Using SIMO Approach. *Telecommunications and Signal Processing (TSP) 2019 42nd International Conference*, (pp. 228-231). doi:10.1109/RADIOELEK.2018.8376391

Subhan, F., Hasbullah, H., Rozyyev, A., & Bakhsh, S. T. (2011). Handover in Bluetooth Networks using Signal Parameters. *Information Technology Journal, 10*, 965-973. doi:http://dx.doi.org/10.3923/itj.2011.965.973

What is the range of Bluetooth? (2019). Retrieved from Bluetooth: https://www.bluetooth.com/bluetooth-technology/range/