SABS

Sound Adjustable Bluetooth System

By

PROJECT IDEA PAPER SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Science - Computer & Electrical Engineering

School of Engineering and Computing

NATIONAL UNIVERSITY

Capstone Project Approval Form

SABS (Sound Adjusting Bluetooth System)

Submitted by

A project proposal paper submitted in partial fulfillment of the requirement of

BACHELOR OF SCIENCE

ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY

Approved by:

_____

_

Prof. Peilin Fu, Ph.D.

Lead Faculty, BS Electrical and Computer Engineering

National University

Table of Contents

Abstract

This project designs an audio adjustable system that is used to receive and manipulate frequency ranges of the surrounding environment with the purpose of acquiring a specific waveform. The desired waveform for the output will be specifically audible literate frequency ranges, which will allow the user to have a conversation without removing protective earphones in a dangerously loud work environment. This design can also be used as an instrument for detailed frequency adjustment for individuals with hearing impairment. The design utilizes a microphone, Bluetooth processor, graphical user interface, audio equalizer software, and a speaker output. These components work in unison to receive, process, adjust, and transmit the desired audio output to the user. This device presents an experimental prototype that, upon many modifications was successful testing, meets the requirements, and achieves the design objective.

1. Problem Statement

1.1. Need:

Many industries have a dilemma where loud noises caused by the surrounding environment exist in close proximity to its employees. Protective ear coverings are supplied by the employer, but there is one problem with noise canceling earphones, and that is that sometimes certain incoming sounds are desired, such as conversations with coworkers or alarm systems nearby. Working in a loud environment requires protective ear covering, and with SABS the user will be able to protect their ears while at the same time hear everything they need to hear.

1.2. Objective:

Design an audio frequency adjusting instrument that has the capability of detecting the frequency range of human audible noise in order to attenuate or amplify the incoming audio tone. This instrument will receive the incoming sounds from the surrounding environment and manipulate the outgoing sound that the user receives by means of adjusting the magnitude of the various frequencies. In doing so this will protect the user from external damaging sound frequencies while allowing them to still hear the desired frequencies of the environment around them. This base instrument will also provide multiple sound output options for the user to receive the sound using Bluetooth earphones or wired headphones. This allows for output flexibility and removes the need for the base instrument to require a proprietary set of ear or headphones.

1.3. Background and Technology Survey:

The importance of SABS is to offer an audio equipment instrument to protect employees from loud work environments by allowing the user to keep their Bluetooth headphones on to continue protecting their ears while at the same time still having a conversation with coworkers. This system uses frequency adjustment to increase or filter audible frequencies with the use of an audio graphic equalizer. Some people have hearing loss to audible frequencies, whether it be high or low frequency hearing loss. SABS will act as a hearing aid in that sense, so that the user can keep the protective earphones on while at the same time have no trouble hearing desired frequencies. Similar headphones and earmuffs exist in different industries for various purposes. One example of a need for protective ear covering would be air traffic control on a tarmac. These workers need to protect their ears from damaging frequencies and amplitude of incoming sound signals. There are many choices on the market today for headphones and earphones with Bluetooth capabilities that allow the user to have a digital sound be transmitted into their ear. With SABS this concept is taken a step further by employing multiple technologies so that a user also has the ability to adjust the audio frequencies of the environment around them that they receive to their ears. The technology involved consists of a computer interface with Bluetooth and equalizer software, and a separate Bluetooth module with a microphone for input and speakers for output. The SABS design is also equipped with light indicators that will let the user know when it is picking up sound in the nearby environment.

1.3.1 Operating System, Sound Driver

For our prototype design we are using Microsoft Windows 10 operating system. To use audio in a computer a sound card hardware is required and usually comes installed into the motherboard

or can be external hardware. For audio processing in a computer another important component is the audio driver. The audio drive is the software that helps the operating system communicate with audio devices such as sound cards and speakers. It also manages audio input and output devices along with providing high quality sound playback and recording on the sound system of a computer. Drivers are needed fundamentally just to allow an audio device to even work on the computer.

1.3.2 Bluetooth Technology

Bluetooth technology is a short-range wireless technology that allows wireless communication and data exchange between stationary or mobile electronic devices. Bluetooth technology communication is based on using a Bluetooth transmitter and receiver. Communication is conducted using RF transmission using ISM band centered at 2.4 gigahertz frequency which is similar to that of Wi-Fi and microwaves. Bluetooth devices are managed using an RF topology known as a "star topology." A group of devices synchronized in this fashion forms a piconet, which may contain one transmitter and multiple active receivers. Bluetooth technology can also be conducted in a bidirectional manner wherein two devices can both be capable of transmission and transmitting which is the specific technology need of our design.

1.3.3 Sound

Sound is a complex element and thus is divided into multiple measurements. Two measurements regarding equalizers are Hertz and Decibels. The relationship between Hertz and Decibels allows a listener to measure the frequency and perceived loudness of a sound. The frequency or amount

of air pressure change vibration is measured in Hertz. The resulting change in air pressure created through the vibrating object is measured in decibels. Decibels therefore essentially measure the loudness of a sound and Hertz measures the frequency of the sound. One of the main considerations of sound is to experience it in a way that it is accommodating to the human ear. Furthermore, certain frequencies are louder than others to the human ear, despite having the same or even more energy behind it. The human hearing range is roughly between 20-20,000 Hz, and the closer we approach or exceed these boundaries, the softer and less harmful the sound will be.

1.3.4 Equalizers and DAC

Equalizers are software or hardware filters that adjust the loudness of specific frequencies. As with all sound development and engineering, the basis of an equalizer is to allow a user to amplify or filter specific sound frequencies through attenuation. The equalizer then becomes an instrument that allows the modification of frequencies that are found within human ear hearing ranges. In an equalizer these ranges are called “bands” and are attenuated as desired. For many years equalizers were analog based hardware circuits that consisted of electronic components such as slides, potentiometers, and resistors among other components. Over the years audio has been developed even further and transitioned from analog audio into digital audio using an instrument called a analog-to-digital converter (ADC). ADC is a system that converts an analog signal such as a sound picked up by a microphone into a digital signal. The reverse of this process technology was also created and called a digital-to-analog converter (DAC). DAC is a device that converts digital audio information (composed of a series of 0s and 1s) into analog audio. With the development of systems like the ADC digital equalizer were able to be created using computers and processors. With this advancement, equalizers no longer had to be based on

large hardware and electronic components but could be integrated into a computer and designed as software. Equalizer software can come in many forms such as parametric and graphic, but for our design we will be using the graphic which is a simpler form of frequency adjustment.

1.3.5 Microcontroller

The microcontroller unit is a programmable integrated circuit (IC) that can be used in numerous electronic applications. Microcontrollers are used for controlling aspects of an electronic system. Microcontrollers implement their control off multiple general-purpose input/output pins that can be programmable peripherals. That can even be set to either receive or send data. Furthermore, most microcontrollers feature timer modules and standard communication interfaces.

1.3.6 Needs Identification

1.3.6.1 Marketing Requirements

The system should:

- Have an easy-to-use interface.
- Allow multi-band frequency adjustment.
- Allow microphone audio input.
- Allow adjustment of audio input.
- Allow audio output.
- Be able to store presets.

1.3.6.2. Objective Tree

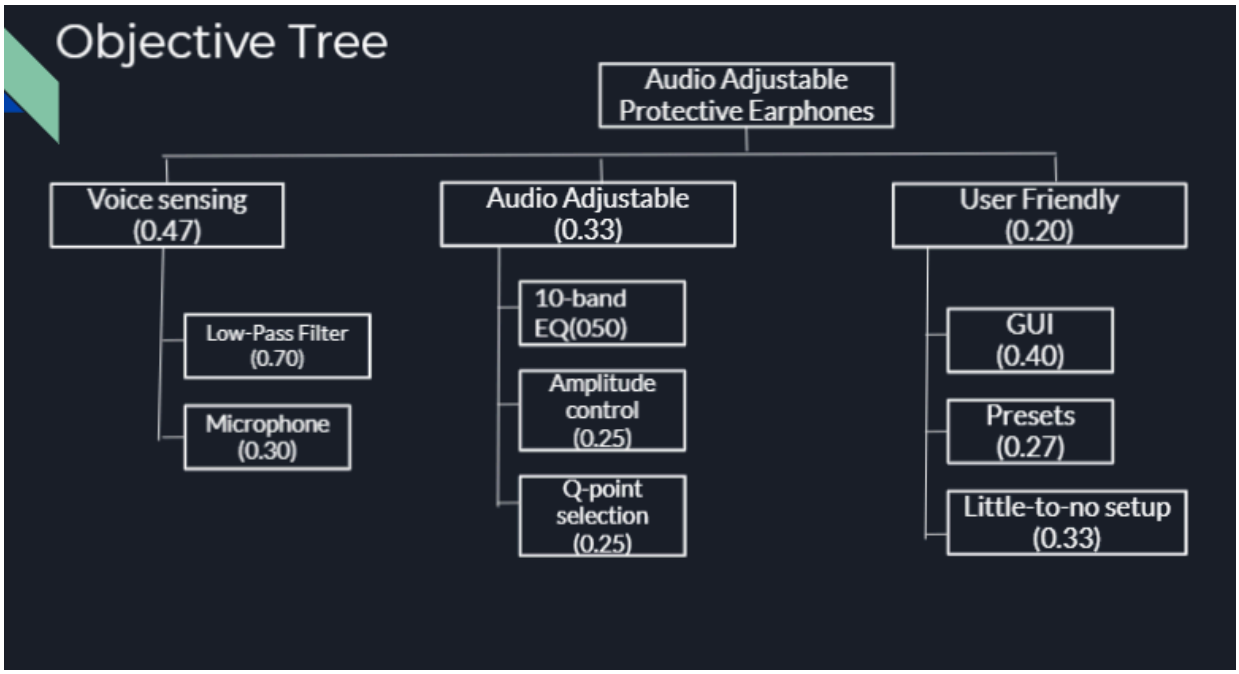


Figure 1: Objective Tree

2. Requirements Specifications

2.1 Requirements

The table below compares the project's engineering requirements with the marketing requirements they cover as well as their justifications.

Marketing	Engineering	Justification
1,2	Microphone must accept all audible incoming frequencies between 20 Hz - 20 kHz	Needs to be able to hear all sounds initially
2	Must be capable of (DSP)	Needs to allow digital and analog conversions
2	Must be capable of audio data two-way communication	Needed for sound processing
3,5	Must provide a user interface	Required for sound modification
3	Must have a passband of approx. 40Hz - 5kHz	This is the literate audible frequency range
3	Must allow modification of frequencies	Required for waveform modification
3	Must have amplitude range of desired output to be between 0-80dB	This is so the volume can be adjusted
3,4,5	The equalizer GUI has presets for different forms of hearing impairment	This is to minimize the user's involvement
5,6	System provides light indication of the presence of sound	

1. The system will receive all incoming audio signals from the surrounding environment.
2. The system will process the incoming audio into Microsoft Windows.
3. The system can manipulate the incoming signals to a desired waveform by audio equalizer.
4. The device will have the desired output sent to the output component.
5. The user interface will be easy to use and configured upon initialization.
6. System has visual indication of the presence of sound.

Figure 2: Requirements Table

2.2 Constraints

2.2.1. Economic:

The price of our device should not exceed more than $150 dollars.

2.2.2. Environmental:

The power consumption should be kept minimal.

2.2.3. Health and Safety:

The device should not be able to injure users or anyone around it.

2.2.4. Social

The device should be able to be simple to set up for anyone with common computer and Bluetooth knowledge.

2.3. Standards

2.3.1. Testing

The device final testing must meet all requirements proposed by the project sponsor.

2.3.2. Documentation

All the design phases should be documented and provided to the company sponsor.

3. Design

3.1 Level Zero Design

Figure 3 below is the Level 0 diagram of our SABS project that displays a general overview of the design. As observed in the diagram, the system receives both a microphone audio input, and a power input. After the system performs its operations the outputs are speaker and the led light system.

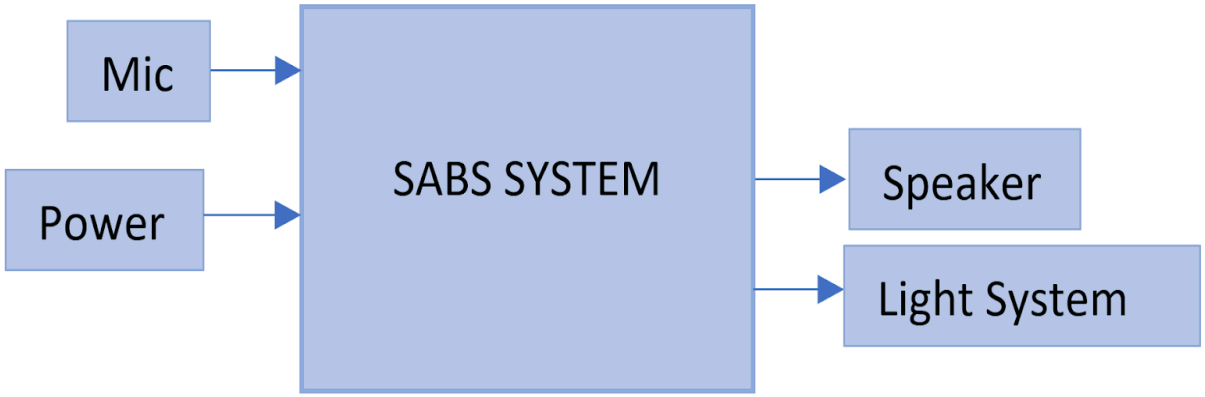


Figure 3: Level 0 Diagram

3.1 Level One Design

In level one design we look deeper into our SABS design configuration and the components relationship with each other. As observed in the diagram below, a 3.6-volt battery is used to power the RN-52 Bluetooth module. In addition, an electret gain amplified microphone is connected to the RN-52 and is used as an input to send analog audio into the Bluetooth module. After converting the analog audio to digital data, the data is then sent to the Microsoft Windows system. The windows system will contain equalizer software and will be used by the user to modify the audio data. The resulting audio data is then sent back to the Bluetooth module which in turn reproduces the audio to the audio output speaker. A USB powered Arduino sound to light indication system is also connected to the system.

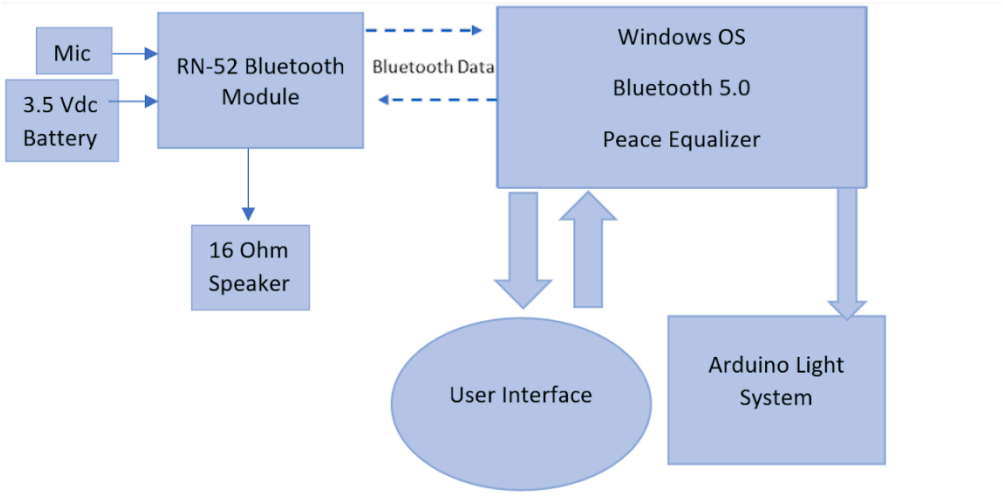
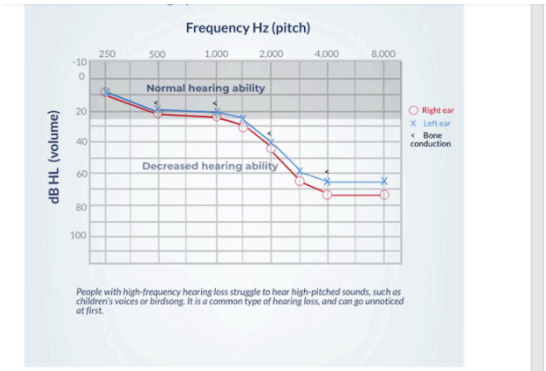


Figure 4: Level 1 Diagram

3.3 Level Two Design

Our SABS system is composed of a hardware-based Bluetooth system along with a separate system that functions as a user interface based on using a computer. The computer that is used has the necessary features of having audio capability, Bluetooth capability, and equalizer software. Below is a step-by-step process of how the overall system works and how the two systems work with each other.



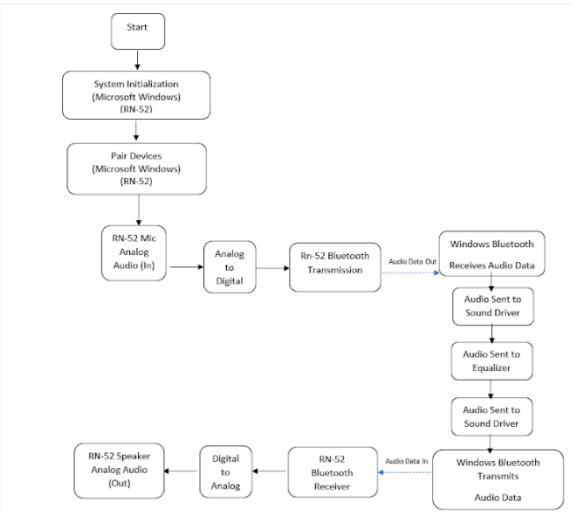


Figure 5 : Level 2 System Block Diagram

Level 2 System Flowchart (Steps)

1. (Initialization) Microsoft Windows and RN-52 are powered on.
2. Pair Bluetooth of Computer and RN-52.
3. RN-52 obtains analog audio from the microphone.
4. Analog audio converted to digital audio (ADC).
5. Audio data transmitted out from RN-52 over Bluetooth.
6. Windows receives Bluetooth Audio Data.
7. Audio Data processed through Audio Driver.
8. Audio driver sends audio through Equalizer.
9. Equalizer adjusts audio according to user adjustments.
10. Audio sent back to Audio Driver.
11. Audio data transmitted out from windows over Bluetooth.
12. RN-52 receives audio data.
13. Audio data converted back to analog (DAC).
14. Analog Audio sent to speaker output.

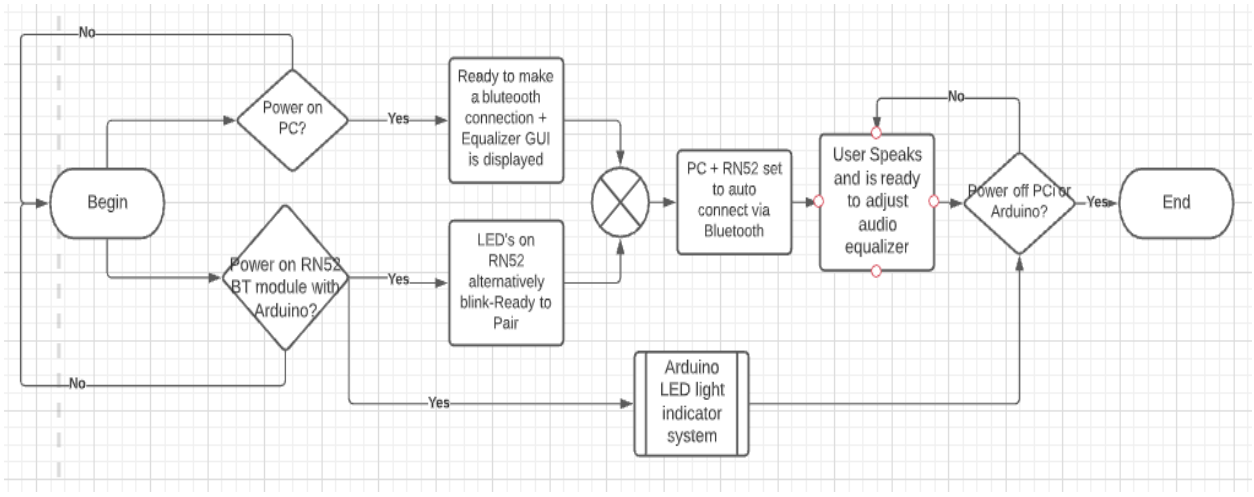


Figure 6: Level 2 Flowchart

3.3.1 Bluetooth System Design

Bluetooth is an essential part of our SABS system and there are two Bluetooth devices used, the RN-52 Bluetooth Module, and the Bluetooth 5.0 technology found within Microsoft Windows computer. The two Bluetooth systems both send and receive Bluetooth audio data back and forth simultaneously. One Bluetooth component is the RN-52 which sends out audio data that is received from its microphone to the other Bluetooth component. Then after equalization the Windows Bluetooth component sends that altered audio data back to the RN-52 Bluetooth module. We will discuss the Windows related Bluetooth 5.0 later in the 3.3.2 SABS Computer interface section and detail its Bluetooth and operation. For now, we will discuss the primary Bluetooth system that is based on the RN-52. First, we will discuss the RN-52 and its configuration procedure to make it compatible with the SABS system and in HFP/HSP profile. Following this we will discuss the circuit design process of the system and its input and output components. Figure 7 below is the diagram of how the Bluetooth system works.

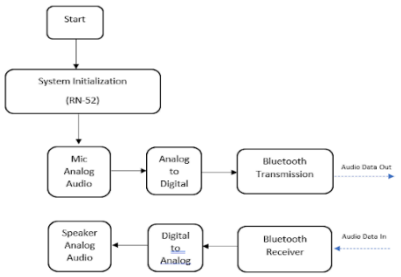


Figure 7: Bluetooth System Flowchart

Bluetooth System Flowchart (Steps)

1) (Initialization) RN-52 powered on.

2) Pair Bluetooth of Computer and RN-52.

3) RN-52 obtains analog audio from the microphone.

4) Analog audio converted to digital audio (ADC).

5) Audio data transmitted out from RN-52 over Bluetooth.

6) RN-52 receives audio data.

7) Audio data converted back to analog (DAC).

8) Analog Audio sent to speaker output.

3.3.1.1 Roving Networks RN-52 Bluetooth Module

The first Bluetooth component being discussed is the Rover Networks RN-52-I/RM model WRL-12849 ROHS Bluetooth module with breakout board. This module uses class 2 Bluetooth

radio, an embedded Audio (DSP) 16-bit Stereo codec digital signal processor that is capable of analog to digital conversion (ADC), along with digital to analog converter (DAC) . The module also contains an amplifier and a version 3.0 Bluetooth module. It is compatible with all Bluetooth v3.0 devices and backwards compatible with all Bluetooth v2.1 + EDR, 1.2, and 1.1 devices. For our Sabs system we use the RN-52 as one of the Bluetooth devices but also as the medium to facilitate the processing and amplification of the input and output components, which in our case are the microphone and speaker. Once configured, the RN-52 can operate in HFP/HSP profile. HFP/HSP allows the audio data in and out process which is required for our Sabs system to work. Figure 7 below reflects the block diagram of the RN-52 module.

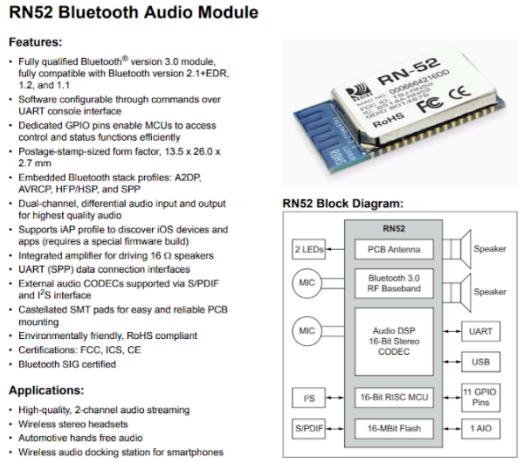


Figure 8: RN-52 Block Diagram

Figure 8 Below represents a general specification table of the RN-52 module

TABLE 1-1: GENERAL SPECIFICATIONS

Specification	Description
Standard	Bluetooth 3.0, class 2
Frequency Band	2.4 ~ 2.48 GHz
Modulation Method	GFSK, PI/4-DQPSK, 8 DPSK
Maximum Data Rate	3 Mbps
RF Input Impedance	50 ohms
Interface	UART, GPIO, AIO, USB, SPI, speaker, microphone
Operation Range	10 meters (33 feet)
Sensitivity	-85 dBm at 0.1 % BER
RF TX Power	4 dBm

TABLE 1-2: WEIGHT & DIMENSIONS

Specification	Description
Dimensions	26.0 mm x 13.5 mm x 2.7 mm
Weight	1.2 g

TABLE 1-3: ELECTRICAL CHARACTERISTICS

Specification	Description
Supply Voltage	3.0 ~ 3.6 V DC
Working current	Depends on profiles, 30 mA typical
Standby current (disconnected)	< 0.5 mA
Temperature	-40°C to +85°C
ESD	JESD22-A224 class 0 product
Humidity	10% ~ 90% non-condensing

Figure 1-1 shows the module's dimensions and Figure 1-2 shows recommended landing pattern and layout.

Figure 9: RN-52 Specifications

The RN-52 supports different configuration profiles as follows:

- **SPP** - Serial Port Profile allows you to configure the device over a UARTserial connection. This can also be used to send commands to the module from a microcontroller.
- **HFP/HSP** - Support of both Hands-Free Profile and Headset Profile mean the module can act as a headset device. This allows the Bluetooth to send audio back and forth, just like a Bluetooth headset. You can send audio input through a microphone and receive audio output through a speaker or headphones.

- **A2DP** - Advanced Audio Distribution Profile sends audio in one direction, but the quality of that audio is better than the quality of HFP and HSP.
- **AVRCP** - A/V Remote Control Profile allows you to control certain features of your audio through the module. Tasks such as Play/Pause, Volume Up, and Volume Down can be controlled by inputs on the module.
- **iAP** - This is the iPod Accessory Protocol, and it allows you to connect the module to any Apple devices such as iPhones, iPods, and Mac laptops and computers.

Range

The RN-52 is a Class 2 Bluetooth device, meaning that the range of the on-board antenna is about **10m**. Thus, you should be able to stream audio to the module from about 32 ft in open air. If you are streaming through walls or windows, that range will diminish. This is important to note as our design configuration can allow the user to leave the PC equalizer user interface in a stationary location and be able to wirelessly receive the audio signal within 32ft before losing Bluetooth connection.

Communication:

Along with Bluetooth connection, the RN-52 has four paths on which it can communicate to outside devices: UART, USB, SPI, and PCM. (UART and USB were used for configuring)

Amplifier:

The RN-52 has a built-in integrated amplifier capable of driving two 16Ω speakers or most standard headphones and outputs approximately 15-20 mW

Data rate: Maximum data over air rate: 3.0Mbps, Baud rate: 115,200 Kbps, Bits: 8

3.3.1.2 Roving Networks RN-52 Configuration Process

To use the RN-52 as a Bluetooth module in the SABS system the RN-52 must first be configured and programmed using an Arduino microcontroller. The components needed for configuring consist of the RN-52 on a breakout board and jumper wires. Along with an Arduino Uno microcontroller with its USB cord for computer connection, and a FTDI to USB Adapter with its USB cord for connecting it to the computer. Figure 9 Below represents the schematic diagram of the configuration that is needed to configure the RN-52.

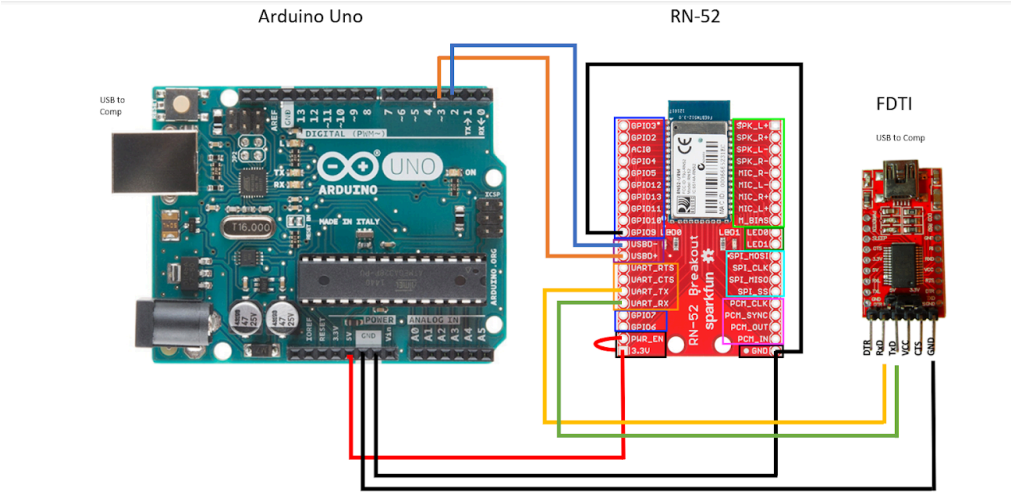


Figure 10: RN-52 Configuration Schematic

RN-52 Configuration Hardware (Steps)

1. Connect Arduino Power output pin 3.3V to RN-52 GPIO pin 3.3V
2. Connect jumper wire from RN-52 GPIO pin 3.3V to RN-52 GPIO pin PWR_EN
3. Connect Arduino Power output pin GND to RN-52 GPIO pin GND.
4. Connect Arduino Digital pin 2 to RN-52 GPIO pin USBD-
5. Connect Arduino Digital pin 3 to RN-52 GPIO pin USBD+

6. Connect RN-52 GPIO pin UART_TX to FTDI pin RX
7. Connect RN-52 GPIO pin UART_RX to FTDI pin TX
8. Connect FTDI pin GND to Arduino Power output pin GND
9. Connect Arduino and FTDI cords to devices, then connect both USB cords to computer

Once Hardware connection is complete now, we use the computer to make our software connection. Using the Arduino IDE application, verify the Arduino board, and verify Arduino and FTDI are on the correct ports. Now the Arduino needs to be flashed with the RN-52 Code, so within the data entry window enter RN-52 configuration Code (Code referenced in Appendix). Verify and Upload Code, Select FTDI port and access serial monitor. Serial Monitor windows should pop up. Finally connect the RN-52 GPIO 9 pin to ground to activate command mode. Figure 9 below illustrates the state diagram of the command mode entry and exit.

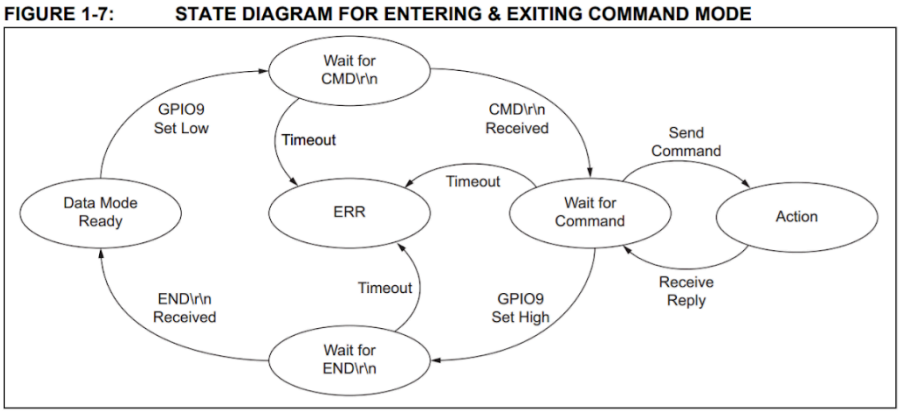


Figure 11: Command Mode State Diagram

Once command mode is entered the serial monitor should display similar to figure 10 below and setting will be displayed once D is entered into the command entry.

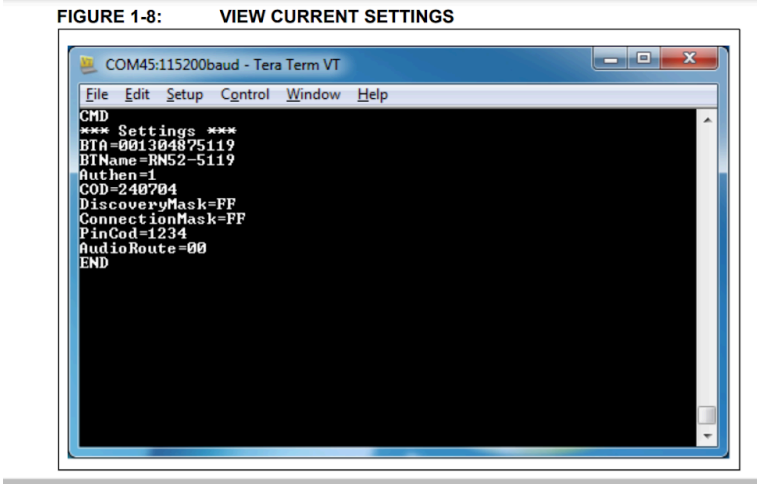


Figure 12: RN-52 Serial Monitor Display

Now that command mode has been successfully entered, the configuration that is needed is to change the Connection Mask and the Discovery mask from the default setting of FF to Connection Mask (HFP) and Discovery Mask (HFP). This can be achieved by entering **(SK,08)** in the command entry line. This will change the Connection Mask from (default = FF) to HFP=08. Once this is successfully done the command line will respond with an AOK to confirm the change. If change was not accepted ERR will be returned. This same process is done to change the Discovery mask using **(SD,08)** which likewise changes the profile to HFP. After confirmation of these modifications command mode can be exited and the RN-52 module is now ready to be installed in the design circuit and used in the SABS system.

3.3.1.3 Full Bluetooth System Circuit Design

Now that the RN-52 has been successfully configured all configuration components can be removed and the RN-52 can be implemented into the Bluetooth system circuit. Figure 13 below illustrates the circuit design schematics of the Bluetooth system.

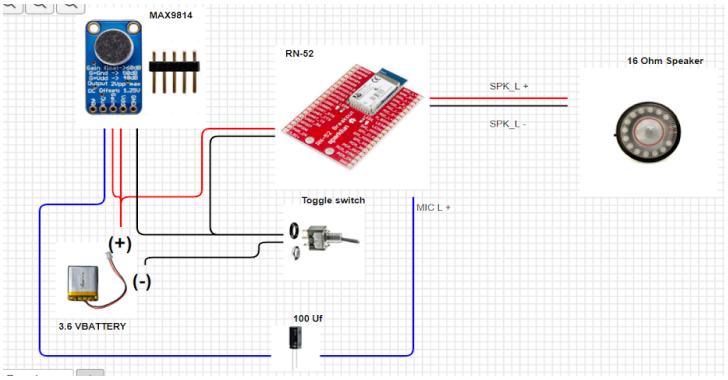


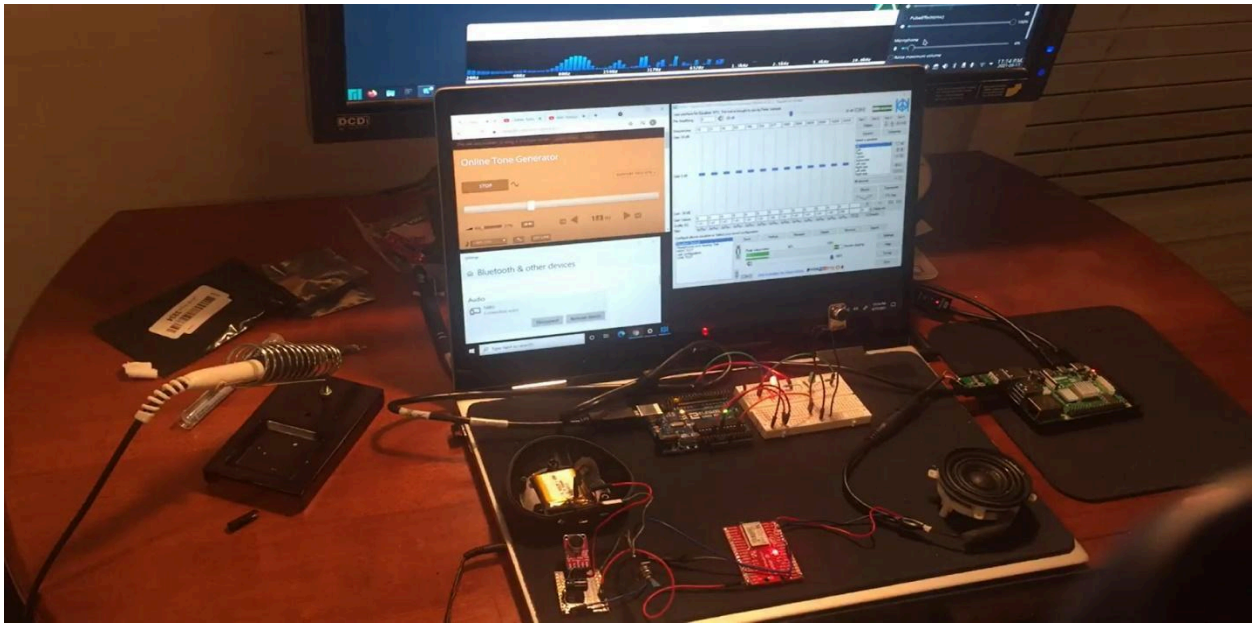
Figure 13: Bluetooth System Schematic

Connection explanation

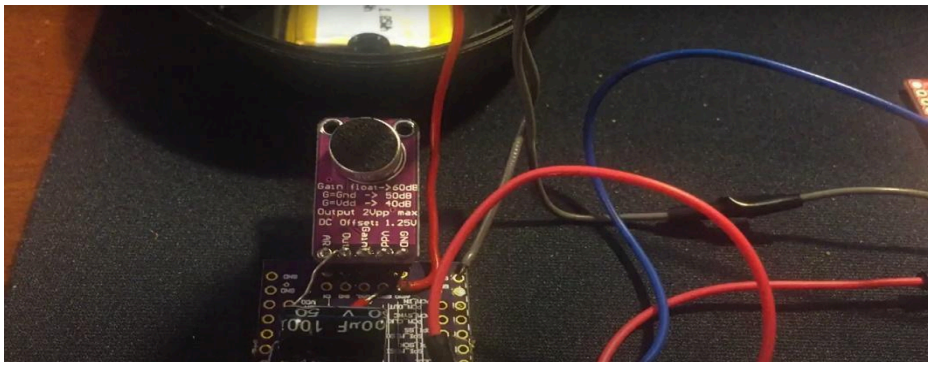
The Bluetooth System consists of the RN-52 Module, a 3.6 VDC Ion Lithium Battery, a toggle switch, microphone, speaker, and connection wire. Connection should start with the input and output components, so first the Microphone OUT post connects to a 100uf Capacitor to the - polarity side. The other side of the capacitor is then connected to the RN-52 GPIO MIC_+. Now the output speaker can be connected by connecting the speaker (+) terminal to RN-52 GPIO SPK L+ and speaker (-) terminal to RN-52 GPIO SPK L-. Now that input output devices are connected, the battery power source can be connected to the RN-52 and the Microphone. The battery operates as a power source for both the RN-52 and the Microphone thus both need to connect to the positive terminal and negative terminal of the battery. The battery + needs to connect to the RN-52 GPIO post 3.3V and a jumper needs to be run from GPIO 3.3V to GPIO PWR_EN which supplies PWR_EN with power. The microphone VCC needs to connect to the positive terminal of the battery. The microphone is amplified and has automatic gain control thus needs a jumper run from the GAIN terminal to the VCC.

RN-52 Operation

Once the RN-52 has been connected to the battery and necessary input and output device the module can be turned on and off with the toggle switch. Once powered on the RN-52 will become active and ready to pair with another Bluetooth device and in this case, pairing is conducted through the computer. Below is a picture of our Bluetooth system.



Microphone (Bluetooth System Input) We used the MAX9814 Electret Microphone



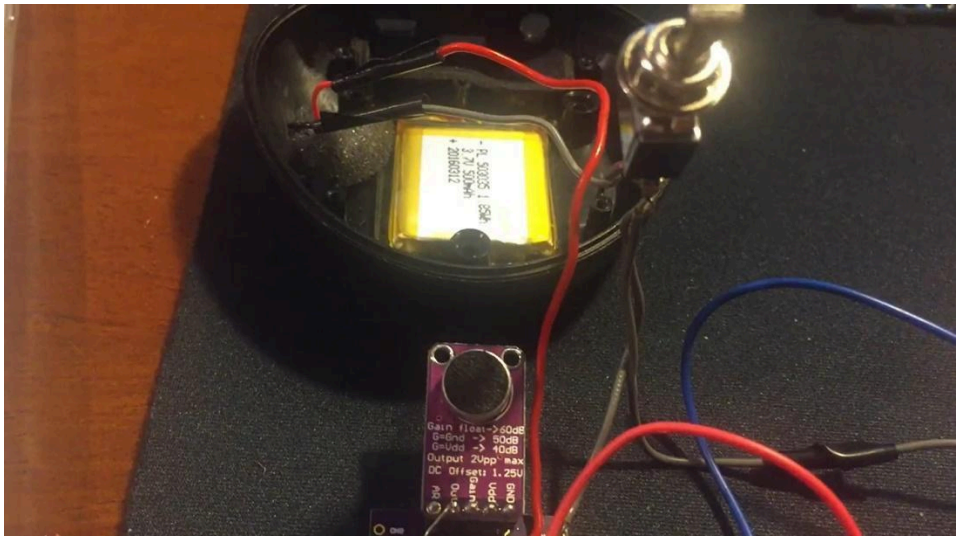
The MAX9814 is a low-cost, high-quality electret microphone amplifier with automatic gain

control (AGC) and low noise microphone bias. The device features a low-noise preamplifier with a fixed 12dB gain, variable gain amplifier (VGA), microphone-bias-voltage generator and AGC control circuitry, while the VGA gain automatically adjusts from 20dB to 0dB and the microphone can receive frequencies between the ranges of 20-20KHz. Supply
The voltage is 2.7v-5.5v and consumes 3mA current with output of 2Vpp on 1.25V bias.

Speaker (Bluetooth System Output)

The RN-52 has a built-in integrated amplifier capable of driving two 16Ω speakers or most standard headphones. For testing we used a general purpose 16-ohm speaker, but for headset configuration two 16-ohm speakers would be recommended such as the Dayton Audio CE Series CE20M-16 3/4" Micro Speaker 16 Ohm which is capable of 0.25 Watt (RMS) power handling.

Battery (Bluetooth System Power)



To power the RN-52 there are three pins necessary to power the board, 3.3V, GND, (PWR_EN). The supply voltage range is 3.0 ~ 3.6 V range. It was important to note that the RN-52 only consumes a maximum of 50mA in the HSP profile and the microphone consumes less than 10ma so a battery of a 3.0-3.6V range with 50-70ma is suitable.

3.3.2 SABS Computer User Interface

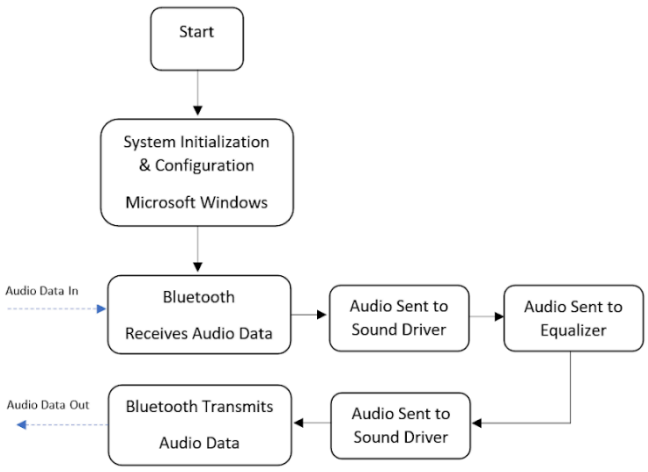


Figure 14: Computer Interface Flowchart

Computer Interface Flowchart (Steps)

1. Initialize Microsoft Windows.
2. Windows receives Bluetooth Audio Data.
3. Audio Data processed through Audio Driver
4. Audio driver sends audio through Equalizer.
5. Equalizer adjusts audio according to user adjustments.
6. Audio sent back to Audio Driver.
7. Audio data transmitted out from windows over Bluetooth.

Along with the Bluetooth system that was just explored, SABS system requires a user interface so that the user can make Bluetooth connections, and use an equalizer to attenuate audio frequencies along with having the ability to select audio input and output devices. After working for a while with the raspberry pi and its operating systems and equalizer we transferred over to windows operating system and its equalizer options. After testing we were successfully able to verify Microsoft Windows, along with its sound driver, Bluetooth technology, and equalizer options was an infrastructure that would work for our design needs. As a processor and interface the computer needs to provide audio device management, sound processing, Bluetooth capability along with an audio driver and support for HSP or HFP Bluetooth profile. For our prototype design we are using Microsoft Windows 10 as our operating system, Realtek High Definition Audio (SST) Audio Driver v.10.0.19041.1 as our Audio driver and Bluetooth 5.0 as our Bluetooth technology.

For our project design we use a computer with a desktop operating system to function as a processor and user interface that will facilitate the equalizer. As a processor and interface the computer needs to provide Bluetooth capability along with an audio driver and support of HSP or HFP Bluetooth profile. In our design the computer will not use any of the analog functionality but will only use data and Bluetooth

3.3.2.1 Computer Software and Technology

Operating System

For our design we are using Microsoft Windows 10 operating system, but many other versions of windows can be used if they can facilitate certain requirements of the system such as Bluetooth profile compatibility and audio driver compatibility.

Audio Driver

Along with the operating system and sound card, the audio driver is also an important aspect of sound processing in our project and the audio driver. The audio driver that comes preinstalled in our designs version of the Microsoft Windows 10 is 10.0.19041.1 Realtek High Definition Audio (SST)) Audio Driver. This Realtek High-Definition Driver is a popular sound driver software that works on top of a computer's sound card. Realtek software comes with several features and functionality such as a six channel Digital to Analog Converter (DAC) that fully supports the 16/20/24-bit Pulse Code Modulation format for 5.1 channel audio. It supports legacy analog input to analog output mixers.

Bluetooth

The second Bluetooth component being used in our system is based on the Microsoft Windows operating system. Bluetooth hardware and software is already installed in our windows operating system along with the audio driver being used. Windows is using Bluetooth, Intel(R) Wireless Bluetooth(R) 20.100.7.1, for driver software to conduct the Bluetooth connection and communication. The Bluetooth driver can sort all current Bluetooth versions and the majority of Bluetooth profiles and uses Bluetooth 5.0 which is backwards compatible with the RN-52's Bluetooth 3.0 technology. It was verified that it is also compatible with making Audio Gateway (AG), and Headset (HS) or Hands-Free (HF) Bluetooth profile connections. In our SABS system we use a Bluetooth profile HSP or HFP Bluetooth profile. Upon testing the Windows driver and Bluetooth we were able to verify that this profile was present in the system as (HFP), and when paired identified the RN-52 as (Sabs Hands-Free AG Audio).

Equalizer Software

After testing multiple equalizers, we selected the two-part equalizer setup that utilized Equalizer APO, and Peace Equalizer Interface (API). Equalizer APO is a free open-source audio graphic equalizer for Microsoft Windows. Equalizer APO allows the equalization parameters to be configured for multiple audio devices and applied to all Windows audio including Bluetooth which is the main need of our design. This equalizer features, virtually unlimited number of filters, works on any number of channels, has very low latency, which makes it suited for interactive application, has low CPU usage, and has a modular graphical user interface.
Peace Equalizer Requirements: - Windows Vista or later (currently only Windows 7, 8, 8.1 and

10 have been tested) - the application must not bypass the system affected infrastructure (APIs like ASIO or WASAPI) exclusive mode cannot be used.

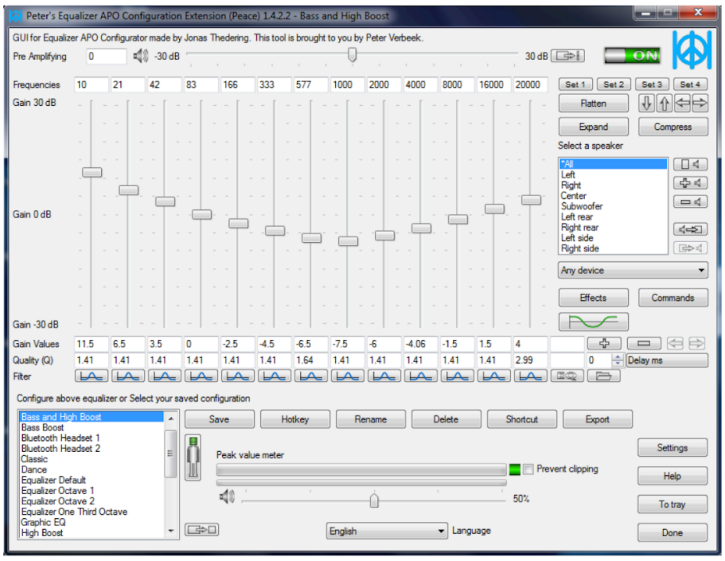


Figure 14: Peace Equalizer Interface

Peace Equalizer is a user interface for Jonas Thedering's Equalizer APO, which must be installed first to be able to run Peace. Peace is your Windows PC equalizer, effects instrument, audio mixer, audio router, bass booster, etc.

Peace Equalizer features:

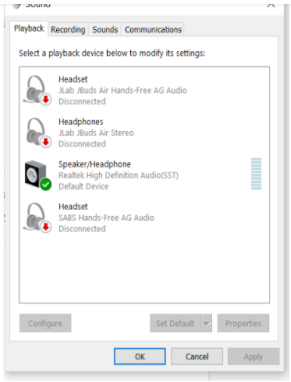
- Up to 31 gain dB Equalizer sliders per channel/speaker (or unlimited if you will)
- 9 speakers support (all, stereo, 5.1, 7.1 or your setup)
- Filter frequencies, dB gains, filter qualities can all be changed per slider
- Filters: peak, low/high pass and shelf, band, notch, all pass and others
- Pre amplifying dB values for volume control (per channel/speaker)
- Graph of your filters (transfer function) per speaker
- Make, save and activate own equalizer configurations (presets)
- Select a device for your equalizer configuration
- Customizable presets
- Effects like down/up mix, cross feed, bass and treble, channel routing
- Control by Windows system tray, hotkeys and desktop shortcuts

- headphones and hearing test interface for creating equalizations based on your headphones/hearing
- automatic activation by program start and device selection

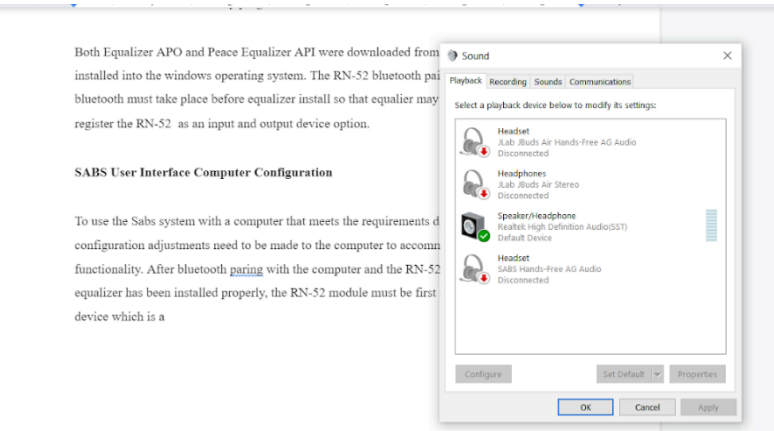
Both Equalizer APO and Peace Equalizer API must be downloaded from online websites and installed into the windows operating system. The RN-52 Bluetooth pairing with the computers Bluetooth must take place before equalizer installation so that the equalizer will be able to identify and register the RN-52 as an input and output device option.

3.3.3 SABS User Interface Computer Configuration

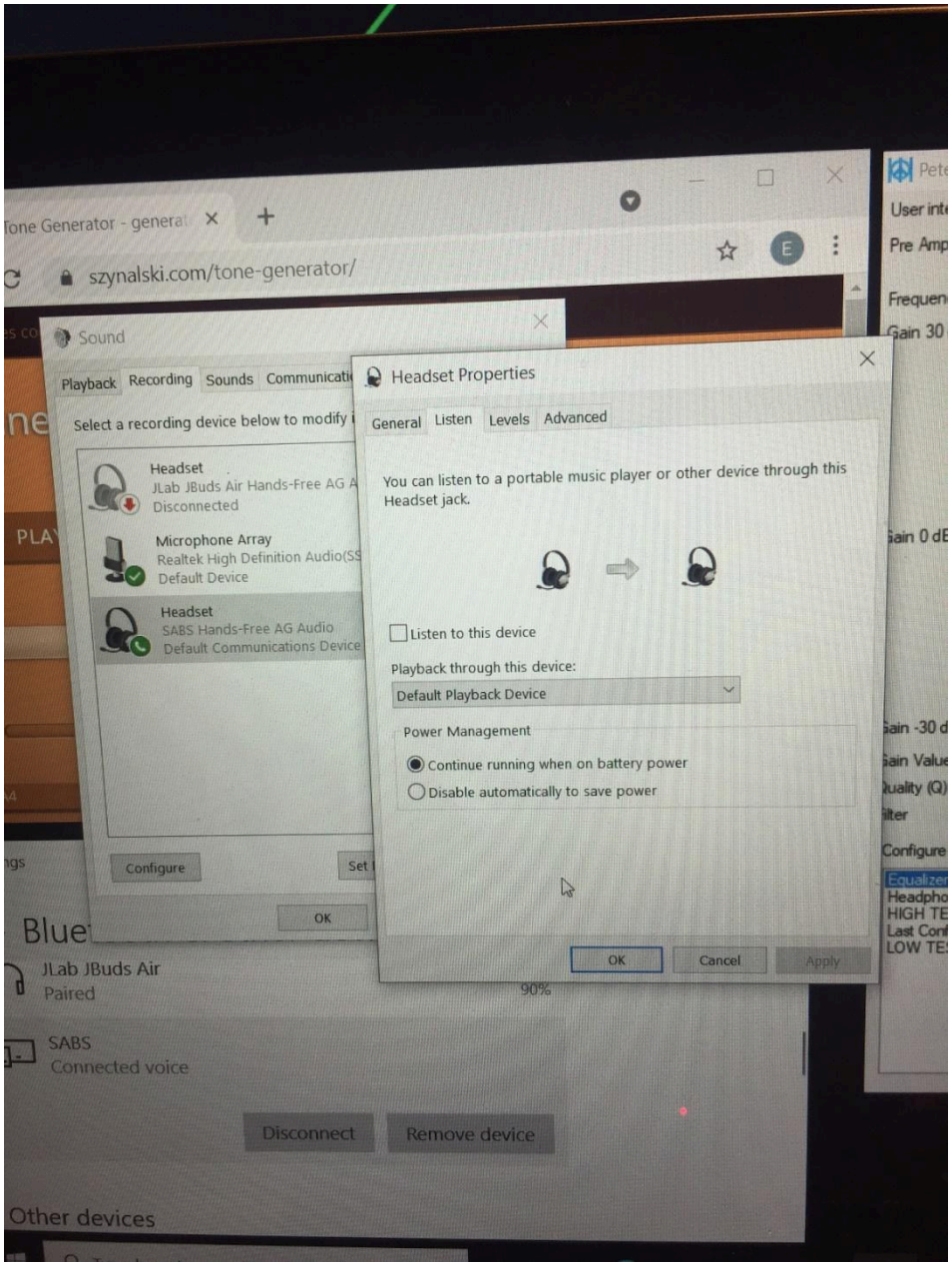
To use the Sabs system with a computer that meets the requirements discussed above a few configuration adjustments need to be made to the computer to accommodate the Sabs functionality. After Bluetooth pairing with the computer and the RN-52 has been made and the equalizer has been installed properly, the RN-52 module must be first selected as the audio output or playback device. This is done by accessing the computer's sounds settings and selecting the RN-52 device to be the playback device. Once this is established any audio going through the computer should be sent to the RN-52 and heard though its speaker output.



After this is successfully done with the output the RN-52's microphone needs to be selected as the computer's input device. This is done by



After this is successfully done with the output the RN-52's microphone needs to be selected as the computer's input device. This is done by also accessing the computer's sound settings but in this case selecting the Recording devices tab.



Once the RN-52's Device is identified and selected (in our case our RN-52 device was renamed SABS) headset properties window will open up. The next step is to select (Listen to this device) and click apply and ok which will save this as the new configuration. Now that this is done the RN-52's mic should be streaming the sound it receives through the computer and its equalizer and back to the RN-52 and heard through its speakers. This will complete the configuration process and the user can access the Peace Equalizer and make audio frequency adjustments that will directly modify the RN-52's microphone input audio and the change should be reflected in the outgoing audio of the speaker.

3.3.3.1 Computer and RN-52 Pairing

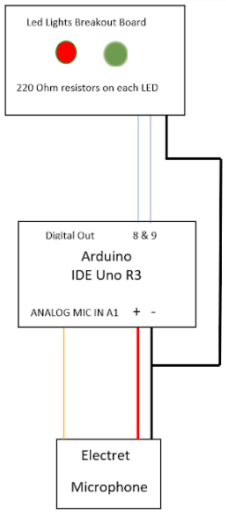
To pair the Windows system with the RN-52 the Bluetooth settings need to be accessed. Once accessed the RN-52 device needs to be added the way a normal Bluetooth device is added. Once the RN-52 is discovered, it can then be connected to which will establish the connection and pair the two Bluetooth devices.

3.3.4 Single LED light system (Arduino)

For this step in our design, we have created an LED light system indicator that will flash green when an incoming voice is within approximately three feet of the user. A red LED will remain on to indicate to the user that the system is on and ready for use. The green light flickers based on an audio response to the incoming serial monitor plotter via the Arduino IDE. The idea is that since all audio frequencies will be equalized and sent to the user's earphones, the Arduino light indication system has a separate microphone that will sense only voices within 3-4 feet from the user and the green LED will only flash when someone is close to the user. Section 3.3.4.1 explains how the Arduino is configured and the code that corresponds to this setup is in the appendix at the end of this paper.

3.3.4.1 Setup

Below is a representation of the system setup schematics.


Led Lights Breakout Board
220 Ohm resistors on each LED
Digital Out
8 & 9
Arduino
IDE Uno R3
ANALOG MIC IN A1
+
-
Electret
Microphone

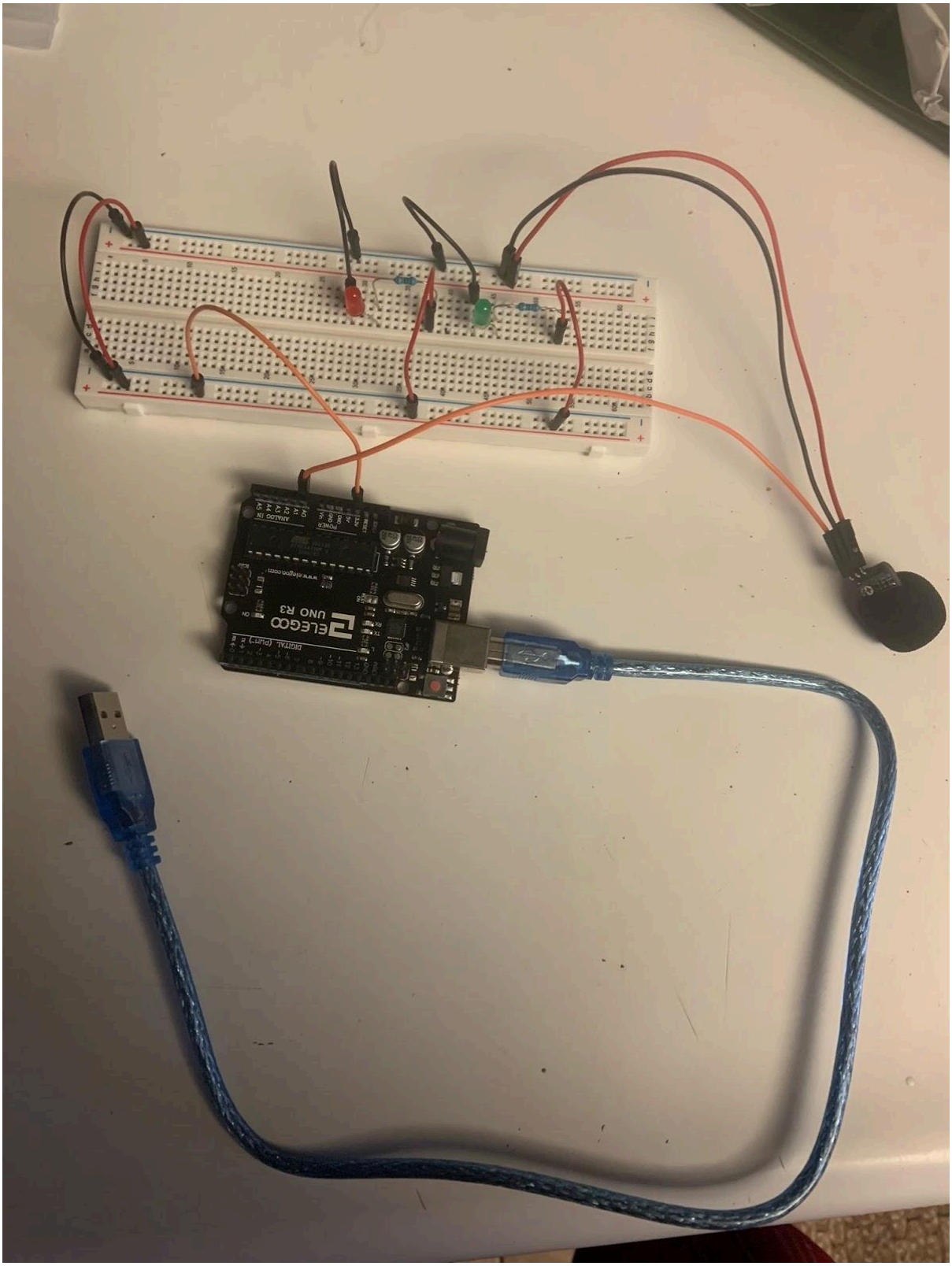


Figure 15: Arduino Light System

As shown in Figure 15, the single LED Arduino light indication system is a single microphone to amplifier that is placed to the “Analog Input 0” connection of the Arduino. A red and a green LED are placed in parallel, and they both are routed to the “Digital Out” pins of the Arduino microcontroller (with a 220 Ohm resistor to protect the Arduino). Lastly, 3.3V is connected to the positive end of the power rail and ground is connected to the negative end.

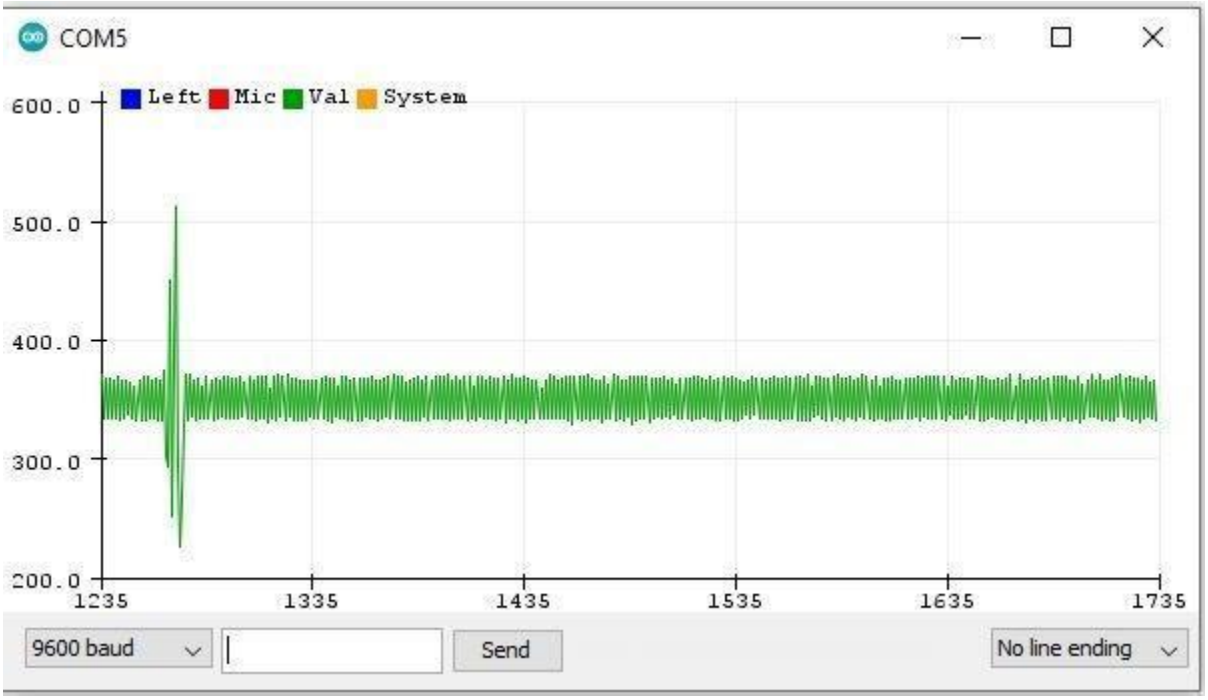


Figure 16: Arduino Serial Plotter

The serial plotter from the Arduino IDE is shown in Figure 15. This serial plotter is used to show the feedback.

Full System Operation (Steps)

1. Turn on RN-52.
2. Turn on the computer.
3. Pair Bluetooth devices through the computer.
4. Set RN-52 as input and output device, (select Listen to Rn-52 input).
5. Access equalizer and adjust frequencies accordingly.

4) Design Verification

4.1 Test Results

Original Designs and Testing

Our design has undergone several evolutions as we worked to implement our design concept into an actual prototype. One of the first obstacles was achieving the objective of having a simple equalizer which is a key feature of the design. Our first design concept was using an Arduino and a Eqic chip, but we discovered this would be extremely complicated to achieve and would require several other components just for that one aspect of achieving an equalizer. With consideration of the time restraint and the projected time needed for research and development, we decided to explore an alternate design route. We then explored options with the Arduino microcontroller but were redirected when discovering the Arduino would not be sufficient to handle the processes and features we were looking to implement. After more research we discovered the Raspberry Pi would be a viable solution for our design. Not only would it allow us to achieve having an equalizer, but it would also have processing capability to incorporate other features such as Bluetooth data communication and a user interface. Furthermore, our design evolved into two separate system components which consist of the Raspberry Pi equalizer user interface, and a unique Bluetooth module. We tested multiple raspberry pi operating systems

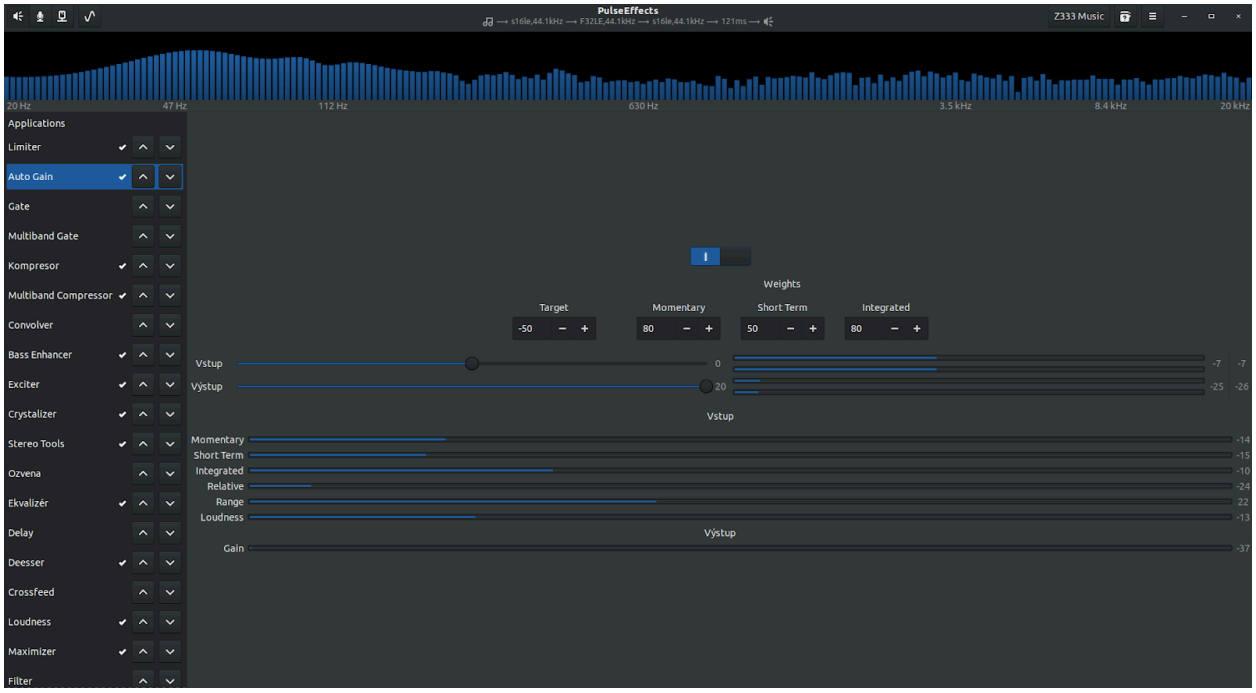
such as Raspberry Pi OS 64-Bit, Ubuntu, Fedora, and various versions of Manjaro. After testing we discovered the Manjaro Arm KDE Plasma was one of the best and most reliable operating systems for audio processing and ease of configurations. With the Raspberry pi-based configuration and the Manjaro KDE operating system being our selected method we then researched equalizer options. After researching equalizer add-on applications that would work well with KDE and we discovered the PulseAudio-Equalizer Ladispa was an excellent choice. It would allow us to accomplish our equalizer objective of offering a simple adjustment user interface that offered presets and multi frequency adjustment. After solidifying this equalizer, we proceeded into the next phase of researching, developing, and testing the Bluetooth connection and audio input and output configuration for our Sabs system design. We then encountered a problem with the RN-52s Bluetooth HSP profile and the Manjaro operating system default sound server Pulseaudio 14.3. When Bluetooth input output communication between the two Bluetooth devices was attempted there was sound distortion and latency in the audio. After further research it was discovered that the Pulseaudio 14.3 server had major issues with the HSP Bluetooth profile due to undeveloped codec that hindered compatibility with the technology of HSP profile Bluetooth. We then researched and tested another audio server named Pipewire-pulse. While this server did resolve the HSP profile issue it was incompatible with Pulseaudio-Equalizer Ladispa because of its dependence on the Pulseaudio 14.3 server. We then explored other equalizers and tried an alternative equalizer called PulseEffects, but dealt with Bluetooth connection issues. This conflict then led us to unfortunately move away from the raspberry pi and the Manjaro operating system and all the work we did with it. We then were encouraged to explore other operating systems that may have better Bluetooth infrastructure so then we explored the windows operating system with its sound drivers and Bluetooth HSP capacities. After

researching and testing we discovered and verified that windows would work properly as it worked with HSP and was equipped with a robust sound driver and had plenty of audio software and hardware capabilities. This lead to us successfully using Windows OS to bring about our SABS design needs.

Final Design & Testing

Microphone Input

For testing our design it was interesting to realize our system works in step by step dependency sequence and essentially if any requirement step is not met the system will not work to expectation. The first step in our systems process is to use a microphone to obtain audio from the surrounding environment. Not only does it need to receive audio, but it has to also be able to receive audio frequency ranges between 20Hz-20kHz, and be a microphone design that is compatible with the RN-52. Due to there being very limited information on which mic is compatible with the RN-52 we tested 5 other microphones which didn't work. We then tested an amplified microphone the MAX9814 which worked and became the mic used for our system. To test and verify the mic could receive 20Hz-20kHz frequencies we used spectrum analyzer. For testing we wired in a 3.5 output jack to the RN-52s Speaker output posts. Then we connected the 3.5 jack into a raspberry pi with a USB audio input module. Within the Raspberry Pi we used the PulseEffects software which has a live graphic spectrum analyzer.



After both speaking into the mic and playing loud music off of a high end home sound system, it was verified that the mic was picking up the desired frequencies between 20HZ-20kHz. For the final product we also used a tone generator and verified these frequencies were being picked up. Lastly, we tested the system was working in normal configuration by audibly listening to the RN-52 speaker and verifying it was reproducing the microphone's audio input.

RN-52

After verifying the microphone, we needed to verify the RN-52 was working correctly. The RN-52 in our system is responsible for receiving analog audio from the microphone, converting the analog to digital and sending it over Bluetooth to a receiving Bluetooth unit. Then the RN-52 also needs to simultaneously receive audio data in from a Bluetooth transmission, convert it back to analog and reproduce it through the attached microphone. The RN-52 also needs to be able to

pair with other devices. As mentioned in section 3.3.1.2 , the RN-52 must be configured to HFP/HSP profile also. To test that this all was working properly we made our power, microphone, and speaker connection. Then we successfully paired the RN-52 to our Windows Interface. After performing the necessary Windows modifications, we were able to verify the module was working properly and as the RN-52's mic was spoken into, the same audio was sent back and played through its speaker output.

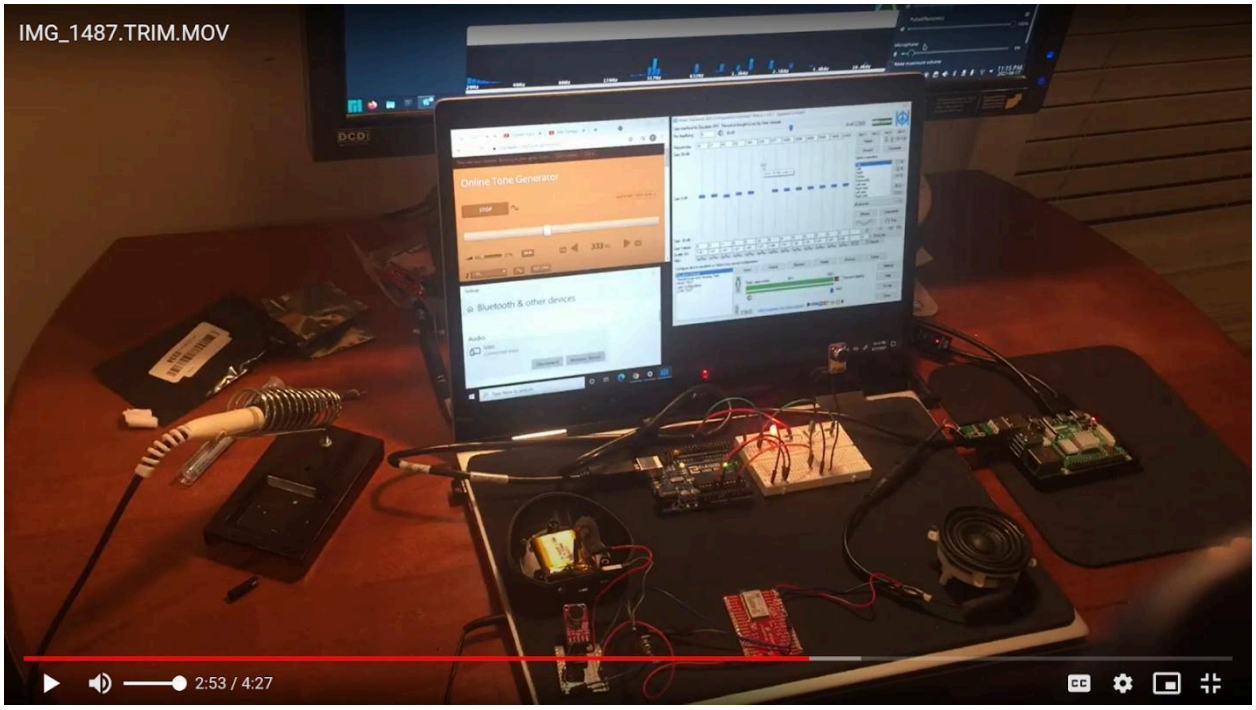
Computer User Interface

The Computer User Interface is the part of our design that allows the user to make the bluetooth connection with the RN-52, designate the RN-52 as the input and output device, and also facilitate the equalizer that the user can access to adjust the audio frequencies. We used a windows operating system to achieve this. One of the big considerations was that the operating system was compatible with the HFP/HSP Bluetooth profile. To test this all was functioning properly we first tested the Bluetooth connection. Pairing was successful and we verified that it was connecting with the RN-52 under the HFP Bluetooth profile. We also verified the RN-52 could be designated as the input and output which allowed the 52's audio to be received, and there to also be audio sent back out to the 52 all under one Bluetooth profile.

Equalizer

The equalizer was an important aspect of our design. We used the Peace Equalizer. The equalizer needed to fulfill multiple requirements for our system to work properly. The requirement was that it would allow a user to adjust the audio frequencies coming from and going to the RN-52. Also, the equalizer had to allow waveform presets to be made and saved. To test the equalizer, we first verified the equalizer worked by testing it on the windows computer using headphones

and playing an audio file. As the audio file was played and the equalizer frequencies were adjusted the output to the headphones reflected the modification. To test the equalizer with our Sabs System we tested using a tone generator from an online website being sent the audio of the computer to the RN-52 which was still connected to the raspberry pi and its spectrum analyzer. We tested multiple tones ranging from 45Hz to 300Hz to 1000Hz and played music all the while adjusting the equalizer and we verified that the equalizer was affecting the audio because the spectrum analyzer displayed the changes to the waveform as the equalizer was adjusted.



Here above is a picture of the tone generating 300Hz and equalizer increasing volume in that range. Furthermore, we validated that presets of specified waveforms could be made and saved.

Speaker Output

This last step in our systems process is that there will be a user frequency adjusted audio output to the RN-52s speaker. We verified this twice over in testing the RN-52 worked properly and that

the computer interface was working properly. So, all in all this final step wherein we received modified audio that originated from the RN-52's microphone verifies our SABS system work properly.

4.2 Requirements Verification

Requirement	Verification
Microphone must accept all audible incoming frequencies between 20 Hz - 20 kHz	This requirement was fulfilled by using the MAX9814 Microphone
Must be capable of (DSP)	This was fulfilled by the RN-52s build in DSP for ADC and DAC conversions
Must be capable of audio data two-way communication	Bluetooth Technology in the RN-52 and Windows conduct audio data communication
Must provide a user interface	Microsoft Windows functions as the user interface
Must have a passband of approx. 40Hz - 5kHz	Equalizer provides bandpass for frequencies
Must allow modification of frequencies	Equalizer allows user to modify frequencies
The equalizer GUI has presets for different forms of hearing impairment	Equalizer allows frequency preset customization
Must have amplitude range of desired output to be between 0-80dB	RN-52 contains amplifier
System provides light indication of the presence of sound	Arduino light indication system fulfills this requirement

Figure 2: Requirements Verification Chart

All the specific engineering and marketing requirements were fulfilled after the final implementation of the design. As mentioned, we underwent several design modifications to achieve our design objective and satisfy our requirements. One of the first challenges we had was that our initial design idea was to use an Eqic which would lead to conflicts with the DSP among other issues, but our final design which used the RN-52 resolved that problem because it incorporated a DSP. So, with the RN-52 and windows design it allowed us to fulfill this requirement. We also had an issue with achieving the two way audio data communication issue.

In the second design configuration where we were using a raspberry pi we had issues with the raspberry pi audio data output in the HSP Bluetooth profile. This caused a major latency and distortion in the audio being received by the RN-52. This concluded in changing to the Windows user interface, and with this design the issue was resolved so that the requirement could be fulfilled. So, all in all we progressively changed our design multiple times to fulfill all the requirements.

5) Summary and Conclusion

5.1 Conclusion

Our Sound Adjustable Audio System (SABS) project design to say the least was a challenging and educational experience. It's interesting to look back and review the whole process from the beginning to the end. As mentioned in chapter three, our design underwent multiple evolution, but we were glad to accomplish the bottom line objective at the end of it all. In review of this project, it almost seems as though we took the longer route in accomplishing our objective. We experienced many hurdles and setbacks, but it was good to work through the problem and obtain solutions which is a vital component of engineering. In working through this project we learned many things regarding the different elements of the project design such as creating the design

idea, research and development, building the design, and lastly documenting the design. One of the most challenging parts for us was research and development of the design. As mentioned prior we encountered multiple design complications such as getting an equalizer that would work with our system, also things like resolving the HSP Bluetooth profile issue wherein some of the technology we had intended to use was incompatible. We also dealt with just funny circumstances like where everything worked with a particular design configuration except for one feature, and then another design configuration fixing the issue with that feature, but then having an issue with a completely different feature. Working on the project also taught us how important certain resources are to have when working on a design. Some of these resources are things like, access to technology information, access to people who are knowledgeable on the technology being used in the design. We must have searched through a hundred forums just looking for information on specific topics like Linux equalizer options, and other topics like HSP compatibility issues with Linux. Large amounts of time were spent on research, and it just shows the value of access to technical information which can save time. Also having resources like knowledgeable people around can be a big help as well. Our team collectively reached out to people we knew that might be familiar with our design's technology such as a person who was knowledgeable with audio and another who was knowledgeable with Linux operating systems. This was a good experience because it gave us more exposure and experience knowledge pertaining to things related to what was used in our system. In reflecting on the project it seems as if we only utilized thirty percent of the overall research we conducted. While that may seem redundant, we can see in retrospect how the research and testing we conducted gave us better experience and understanding of the technology in the market today. We know much more about audio, Linux, Raspberry pi and its operating system distributions among knowing more about

many other things we conducted research and testing on. We even know much more about Bluetooth technology and its various uses in the technology industry of today.

5.2 Further Improvements

As we worked through this project there were several moments where innovation occurred and ideas of alternate approaches and configurations. Due to the limited resources and limited time frame of the project we chose several simpler configurations and elements such as the user interface down to things such as the microphone. There are a few further improvements that could be made if this design idea were continued and advanced upon. As mentioned before, we were exploring a design idea that utilized the raspberry pi with a touchscreen as the basis of the user interface and equalizer. As also mentioned, we encountered several issues as we dealt with the sound server pulseaudio 14.2 and its issues with the HSP and HFP Bluetooth profile issues which are essential for our system. And also dealt with equalizer software issues and their compatibility with specific sound serves. Resolving this issue and obtaining the right sound server, along with the equalizer and HSP/HFP profile compatibility would be a great improvement and allow the raspberry pi configuration method to be utilized. This would allow for much more portability, simplicity, and practicality of the system as it would not need to use a windows computer, but a standard raspberry pi, and even possibly the raspberry pi zero which has an even smaller footprint. Additional improvements can be made regarding the microphone. As found in most audio designs, the microphone is a key element in an audio system and its quality can make a big difference. In our SABS system we tested and researched several

microphones and implemented the best one we found that produced the best audio quality, but this can be further researched and developed which would only give a better experience to the user as they would be able to hear more detail in the audio they receive.

References:

[1] "RN-52 Datasheet" 2013. Sparkfun <https://cdn.sparkfun.com/assets/a/2/a/a/d/5217c61f757b7f55758b456f.pdf>

[2] "RN-52 Hookup Guide" Accessed 6/2021. Sparkfun <https://learn.sparkfun.com/tutorials/rn-52-bluetooth-hookup-guide/all>

[3] "RN-52 Alternative Datasheet" 2021. Microchip <https://ww1.microchip.com/downloads/en/DeviceDoc/RN52-Bluetooth-Audio-Module-DS70005120B.pdf>

[4] "Digital Signal Processing Guide to Digital Signal Processing" Accessed 6/2021 Analog <https://www.analog.com/en/design-center/landing-pages/001/beginners-guide-to-dsp.html>

[5] "MAX9814 Datasheet" Accessed 6/2021. Adafruit <https://cdn-shop.adafruit.com/datasheets/MAX9814.pdf>

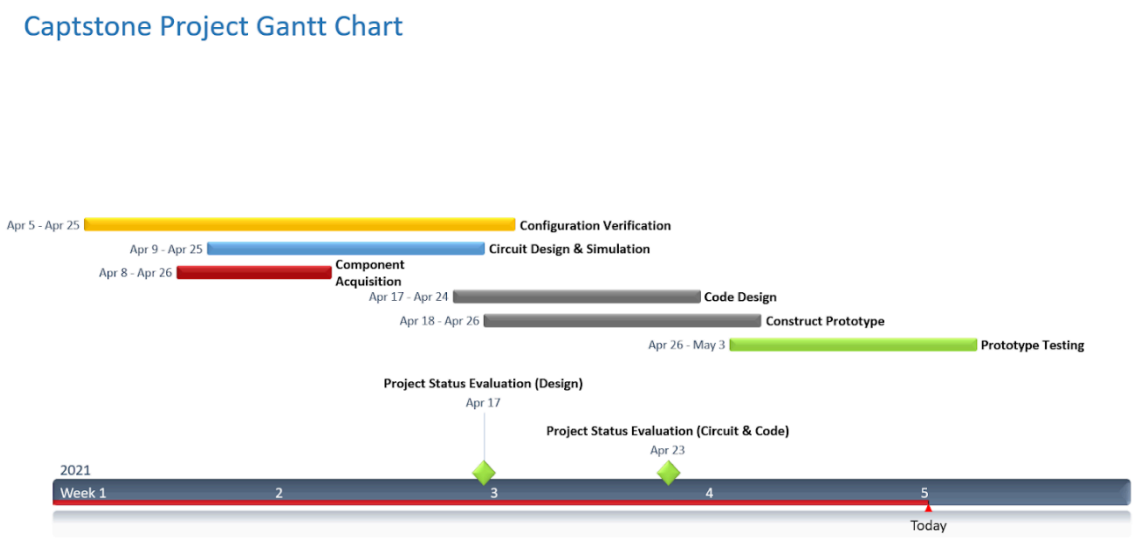
Appendix A: Project Management Plan

Cost List

Tabulated List of Components and Cost

Device	Manufacturer	Part Number	Quantity	Price /ea
Bluetooth Receiver Module RN-52-I/RM	Rover Networks	RN-52-I/RM model WRL-12849 ROHS	1	$54.99
FTDI USB Adapter	IZOKEE		1	$8.55
Arduino IDE Uno R3 Kit	ELEGOO	Uno R3	1	$19.89
8 Ohm speaker	Gikfun	AE1054	1	$5.96
Rechargeable battery lithium-ion 3.7v	EEMB	18650		$9.99
Micro USB port				$4.88
Electret Microphone Amp Module	Comimakr	MAX9814		$5.99
Electret Microphone	HitLetgo	MAX4466		$8.39
Electronic component kit, (switch, capacitor)	Various	Various		$3.00
TOTAL				$121.64

Gantt Chart



Appendix B: Code

Code - RN-52 Bluetooth Module configuration setup using Arduino

```
#include <SoftwareSerial.h>

int bluetoothTx = 3; // TX-O pin of bluetooth mate, Arduino D2

int bluetoothRx = 2; // RX-I pin of bluetooth mate, Arduino D3

int dataFromBt;

SoftwareSerial bluetooth(bluetoothTx, bluetoothRx);

void setup()

{

Serial.begin(9600); // Begin the serial monitor at 9600bps

bluetooth.begin(115200); // The Bluetooth Mate defaults to 115200bps

delay(100); // Short delay, wait for the Mate to send back CMD

bluetooth.println("U,9600,N"); // Temporarily Change the baudrate to 9600, no parity

// 115200 can be too fast at times for NewSoftSerial to relay the data reliably

bluetooth.begin(9600); // Start bluetooth serial at 9600
```

```
bluetooth.print("$"); // Print three times individually

bluetooth.print("$");

bluetooth.print("$"); // Enter command mode

}

Void loop()

{}
```

Code - Single LED light system (Arduino)

```
int LEDRed=9;

int LEDGrn=8;

int micLED=A1;

int val =0;

void setup(){

  Serial.begin (9600);

  Serial.println(F("Initialize System"));

  //Init Microphone

  pinMode(LEDRed, OUTPUT);
```

```
  pinMode(LEDGrn, OUTPUT);
  pinMode(micLED, INPUT);
}
void loop (){
  readMicrophone();}
 void readMicrophone() { /* function readMicrophone */
  val = analogRead(micLED);   ////Test routine for Microphone
  Serial.print("Mic  Output Value:");
  Serial.println(val);
    digitalWrite(LEDRed, HIGH);
    digitalWrite(LEDGrn, LOW);
    if (343< val)
    {
    digitalWrite(LEDGrn, HIGH);
    }
 }
```

Appendix C: Definitions and Abbreviations

SABS - Sound Adjustable Bluetooth System

DSP - Digital Signal Processing

DAC - Digital Analog Converter

ADC - Analog Digital Converter

GUI - Graphic User Interface

HSP – Headset Profile

HFP - Headphone Profile

AGC - Automatic Gain Control

VGA - Variable Gain Amplifier

`