Heterodyne Bat Detector

By

Kyle Jedryszek

Evan McGowan

Bill Waltz

Final Report for ECE 445, Senior Design, Spring 2026

TA: Gayatri Chandran

6 May 2026

Project No. 3

Abstract

A handheld device acting as a bat detector is designed, tested, and implemented in this report, as a project under Dr. Joy O'Keefe in the Natural Resources and Environmental Sciences department. An analog subsystem captures and converts the ultrasonic frequencies that bats emit as part of echolocation down to audible ranges, allowing users to listen to Central Illinois-native bats' calls and chirps in real time. In addition, newly passionate bat-lovers can cross-reference sounds heard from this device with a multitude of on-board pre-recorded bat calls with reference frequencies. Testing yields a result that is satisfactory for a prototype, but additional work remains to improve the analog aspects of this device.

Contents

1. Introduction..... 1
    1.1 Solution..... 1
    1.2 Requirements for Success..... 2
2. Design..... 3
    2.1 Onboard Bat Call Audio Demo Module..... 4
        2.4.1 Playback Button..... 4
        2.4.2 ATTiny85..... 4
        2.4.3 On-board Flash Memory..... 4
        2.4.4 Digital Switch..... 5
    2.2 Bat Detector Frequency Module..... 5
        2.3.1 Pre-Amplifier & Filter..... 5
        2.3.2 Oscillator..... 6
        2.3.3 Mixer..... 7
        2.3.4 High-Frequency Microphone..... 7
    2.3 Audio Output..... 7
        2.2.1 3.5mm Auxiliary Jack..... 8
        2.2.2 Speaker..... 8
        2.2.3 Audio Amplifier..... 8
    2.4 Power Supply..... 8
        2.1.1 AA battery holder..... 9
        2.1.2 Voltage Regulator and Switch/Power Button..... 9
    2.5 Firmware Design..... 10
        2.5.1 Processing File Loader..... 10
        2.5.2 ATTiny85 Programming..... 11
    2.6. Cost..... 13
3. Testing & Verification..... 14
    3.1 Oscillator Module..... 14
    3.2 Mixer Module..... 15
    3.3 Audio Loading Software..... 17
4. Conclusion..... 19
    4.1 Accomplishments..... 19
    4.2 Contributions..... 19
    4.3 Ethical considerations..... 19
    4.4 Future work..... 19
5. References..... 21
Appendix A..... 22
Appendix B..... 24

1. Introduction

Bat detecting is popular overseas in the United Kingdom and Australia, but lacks the same market share in the United States. As such, many bat detectors used in the United States are imported, incurring additional taxes and higher shipping costs. Domestic options are either too expensive and/or have low audio quality, specifically those that plug into phones and tablets. Our client, Dr. Joy O'Keefe, requires a high-quality, mass-produceable, and domestically made device for the purpose of providing several groups of people with a bat detecting device for Bat Walks at the Central Illinois Bat Festival at a reasonable price.

1.1 Solution

We propose a battery-powered handheld device equipped with a microphone, capable of detecting frequencies between 15 kHz and 100 kHz to be converted down to a human-audible range. The frequency to be mixed with is controlled by a large dial (with illuminated frequency labels) on the front of the device. The sound will then be filtered, amplified, and output via speakers or headphones. This model will also have stored, prerecorded sound bytes that can be played so that first-time users can know what to listen for, a feature our client desires that has not been seen yet. An accessible, inexpensive, and reliable option does not yet exist in the domestic market, and our product fills this niche.

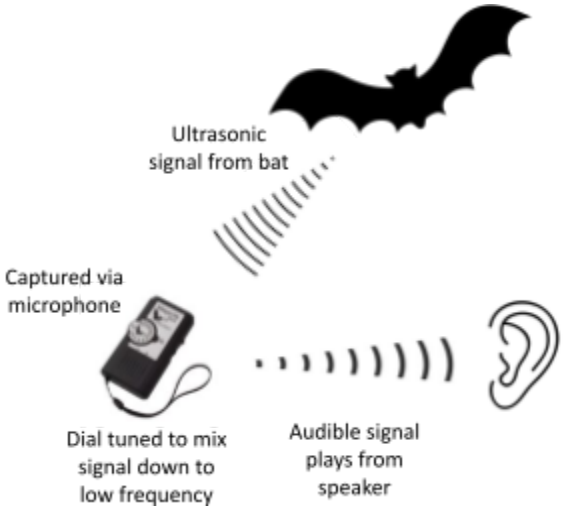

Figure 1: Bat detector use visual aid

1.2 Requirements for Success

- The device must be tunable between 15 kHz and 100 kHz frequencies using the onboard tuner.
- When an ultrasonic source radiates sound, the device must downconvert it to audible frequencies, playing it through the onboard audio output.
    - This is only testable at 40 kHz due to a limitation of our given testing device, the Ultrasonic Calibrator.
- When prompted, digital audio samples (of bat calls or otherwise) must be played through the speaker or headphones.
- Battery life (rechargeable or otherwise) lasts the length of at least one bat walk (1-2 hours)

2. Design

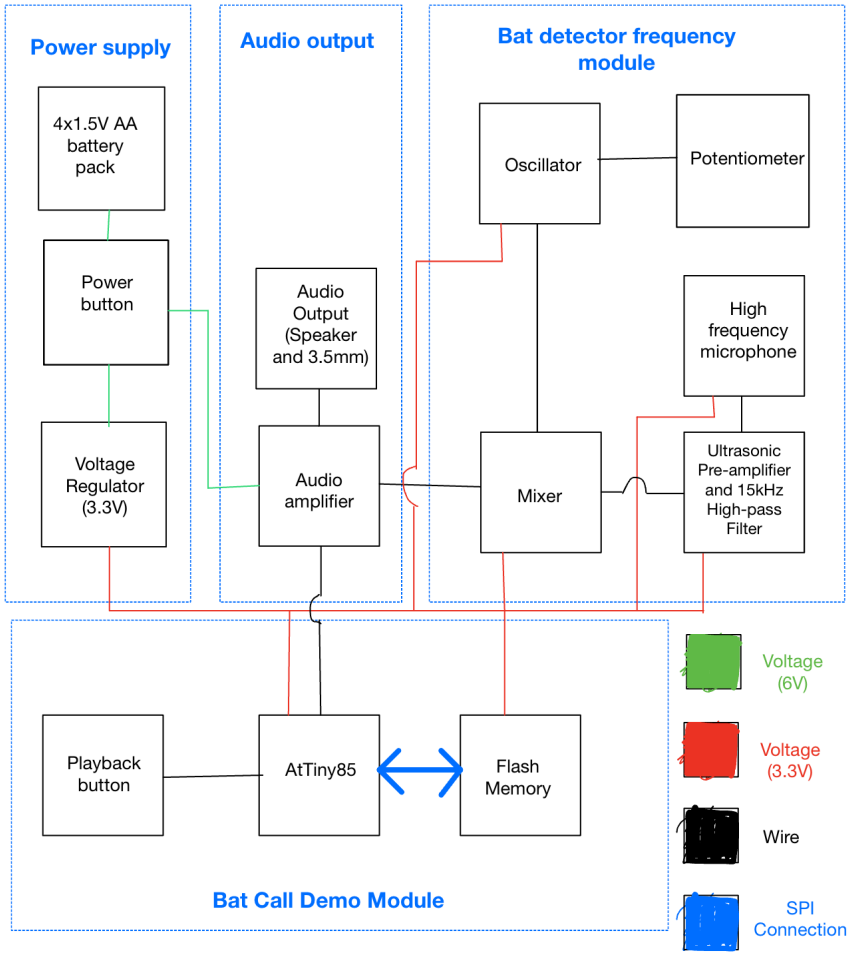

Figure 2: Circuit Design Block Diagram

2.1 Onboard Bat Call Audio Demo Module

- The on-board bat call audio demo will allow the user to listen to pre-recorded (and down-converted) bat sounds in order to be able to identify bats detected by the device when used in detector mode.

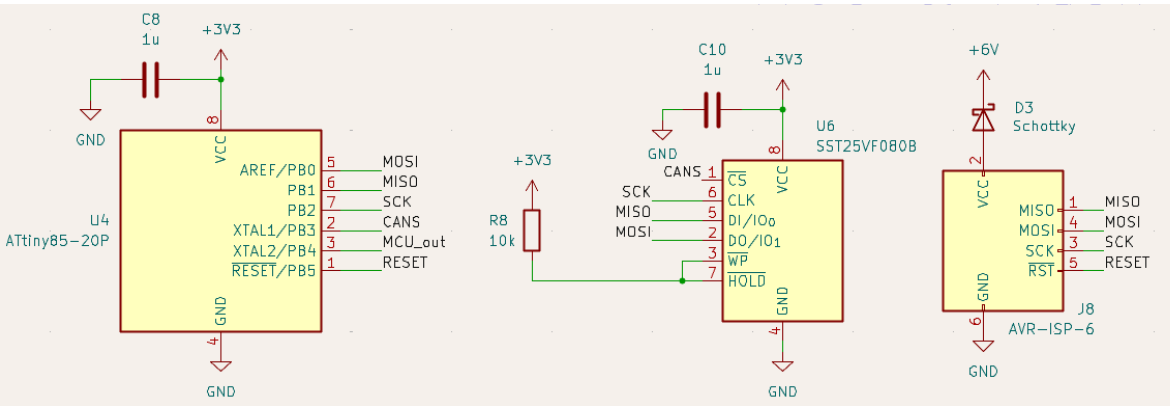

Figure 3: Bat Call Audio Demo Module Circuit Schematic

2.4.1 Playback Button

- The playback button can be pressed by the user to play the pre-recorded audio. When the button is pressed, it will tell the microcontroller to play the audio to the speaker, halting the analog output. See 2.4.4 for more information.

2.4.2 ATTiny85

- The ATTiny85 is our microcontroller of choice, due to its simplicity. An ESP32 or similar would be extremely overkill for this project, as we only need it for SPI communication with the flash memory. When prompted through the button, it will read audio data from the flash memory and output it to the speaker.

2.4.3 On-board Flash Memory

- The on-board flash memory is programmed to store pre-recorded bat sounds, which will be read by the microcontroller for output to the speaker. Compressed audio is flashed at a sample rate of 16 kHz in the SST25VF080B flash memory [1], allowing us to store multiple samples within the 8 Mbit storage.

2.4.4 Digital Switch

- A TS5A3159 SPDT switch [2] regulates which audio input is fed through to the speaker subsystem. This switch is driven by the ATTiny85, such that when the digital bat call demo audio is playing, no analog audio output from the mixer will interfere, and vice versa.

2.2 Bat Detector Frequency Module

- The bat detector frequency module is the main component of the device. It will receive ultrasonic frequencies from the microphone, mix them down based on the value of the potentiometer dial, and send the output to the audio output subsystem.

2.3.1 Pre-Amplifier & Filter

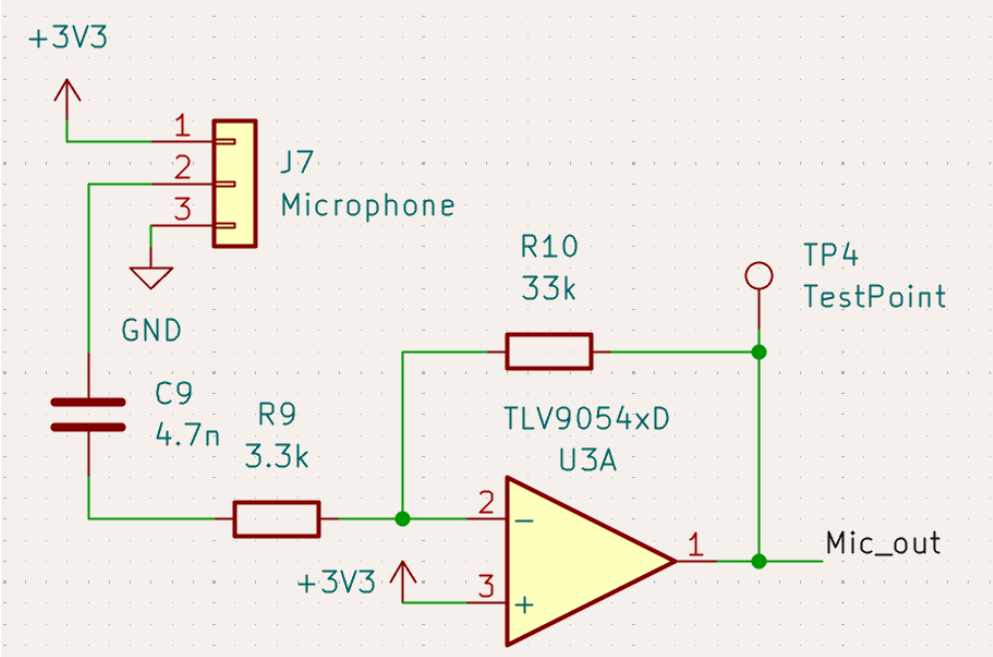

Figure 4: Bat Detector Frequency Module Inverting Pre-Amplifier Circuit Schematic

- The inverting preamp amplifies the input received from the microphone. It is placed before the passive high-pass filter (having a 3 dB bandwidth of 15 kHz) to ensure that most of the low-frequency noise is removed before amplifying.

2.3.2 Oscillator

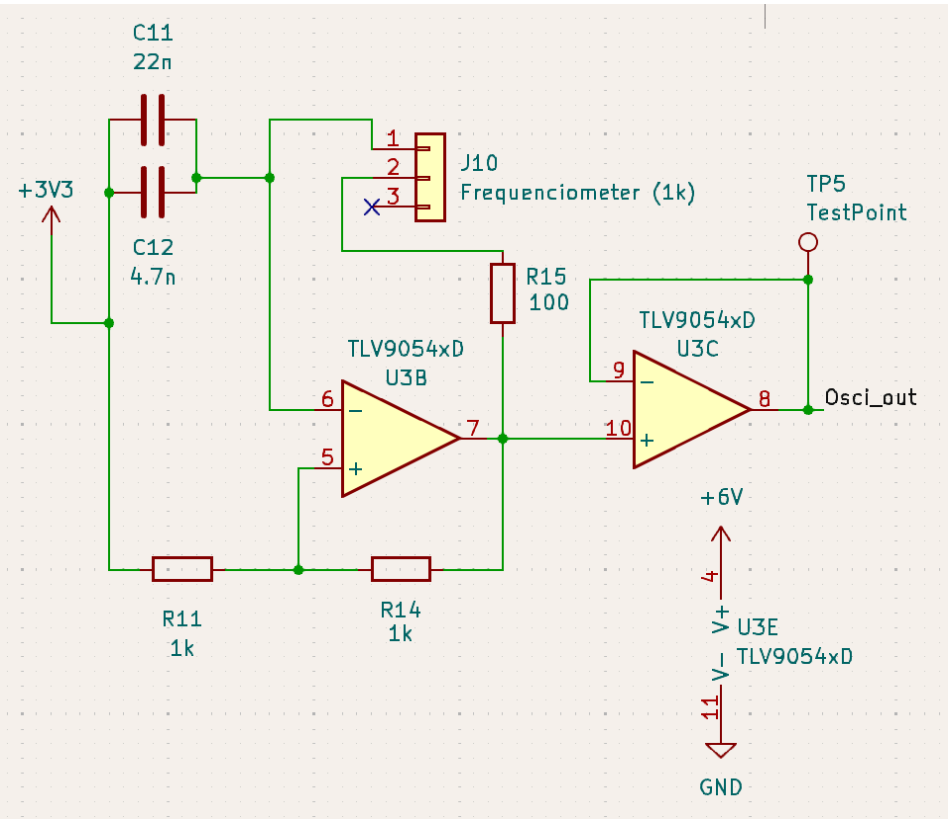

Figure 5: Bat Detector Frequency Module Oscillator Circuit Schematic

- The oscillator will produce a frequency that mixes with the ultrasonic signal, with its frequency being controlled by an onboard potentiometer. A simple relaxation oscillator using the TLV9054 operational amplifier (also known as an op-amp) [3] was chosen due to its high slew rate, which allows for fast-swinging oscillations. A simple RC circuit comprised of C11, C12, J10, and R15 provides feedback to the op-amp, with the potentiometer controlling the capacitors' charge and discharge rate. The structure of this relaxation oscillator is described in [4].

2.3.3 Mixer

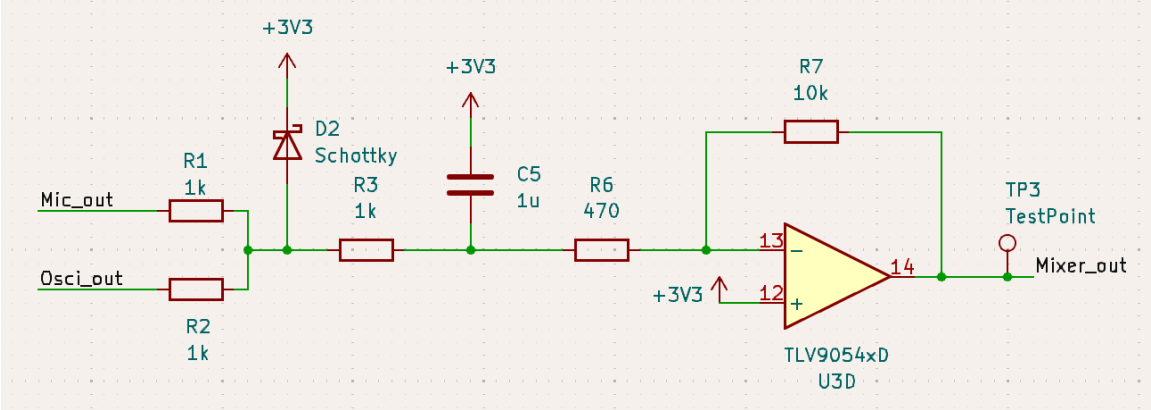

Figure 6: Bat Detector Frequency Module Mixer Circuit Schematic

- The mixer will receive amplified ultrasonic waves and the local oscillator signal, and mix them together to bring them down to a frequency that can be heard by the human ear, which is then output to the audio subsystem. The same frequency deviation is present, but now at an audible frequency.

2.3.4 High-Frequency Microphone

- The high-frequency microphone is the device's input. It will receive ultrasonic audio input from bats to be mixed down by the mixer. Our options for ultrasonic microphones were limited, and we settled on the SPVA1A0LR5H [5] microphone for its high frequency range (0 kHz-80 kHz) and omnidirectional nature.

2.3 Audio Output

- The audio output contains both a 3.5 mm auxiliary jack and a built-in speaker to allow for audible audio output from either of the sources. It will receive the audio output from either the mixer or the microcontroller through the playback feature. The amplifier circuit used here is nearly identical to the one described in the LM386N datasheet [6] for a voltage gain of 200, or 20dB.

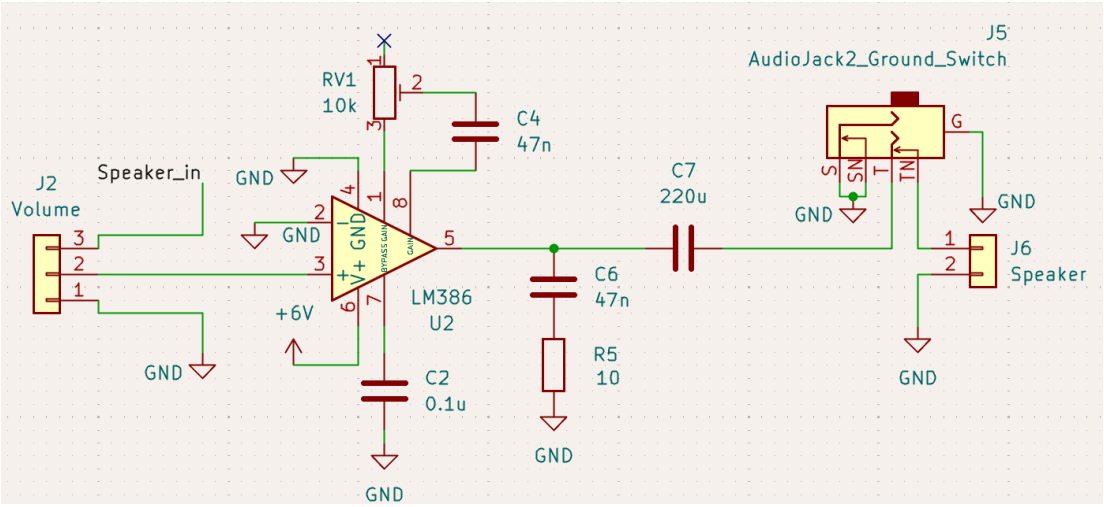

Figure 7: Bat Detector Frequency Module Speaker and Audio Amplifier Circuit Schematic

2.2.1 3.5mm Auxiliary Jack

- The 3.5 mm jack will allow the user to use their own headphones to listen to the device's audio output instead of the built-in speaker, if desired.

2.2.2 Speaker

- The speaker will be built into the detector, allowing the user to hear bat calls through the device through the mixed-down frequency received from the mixer chip.

2.2.3 Audio Amplifier

- The audio amplifier amplifies audio signals received from the mixer and microcontroller into audible frequencies, while filtering out the higher spurious frequencies. It will then send the amplified sound to the speaker.

2.4 Power Supply

- The power supply provides power to the necessary components in the other subsystems. It is powered by four AA batteries with a low-dropout 3.3 V voltage regulator. It is also equipped with a power switch to allow the user to turn the detector on or off.

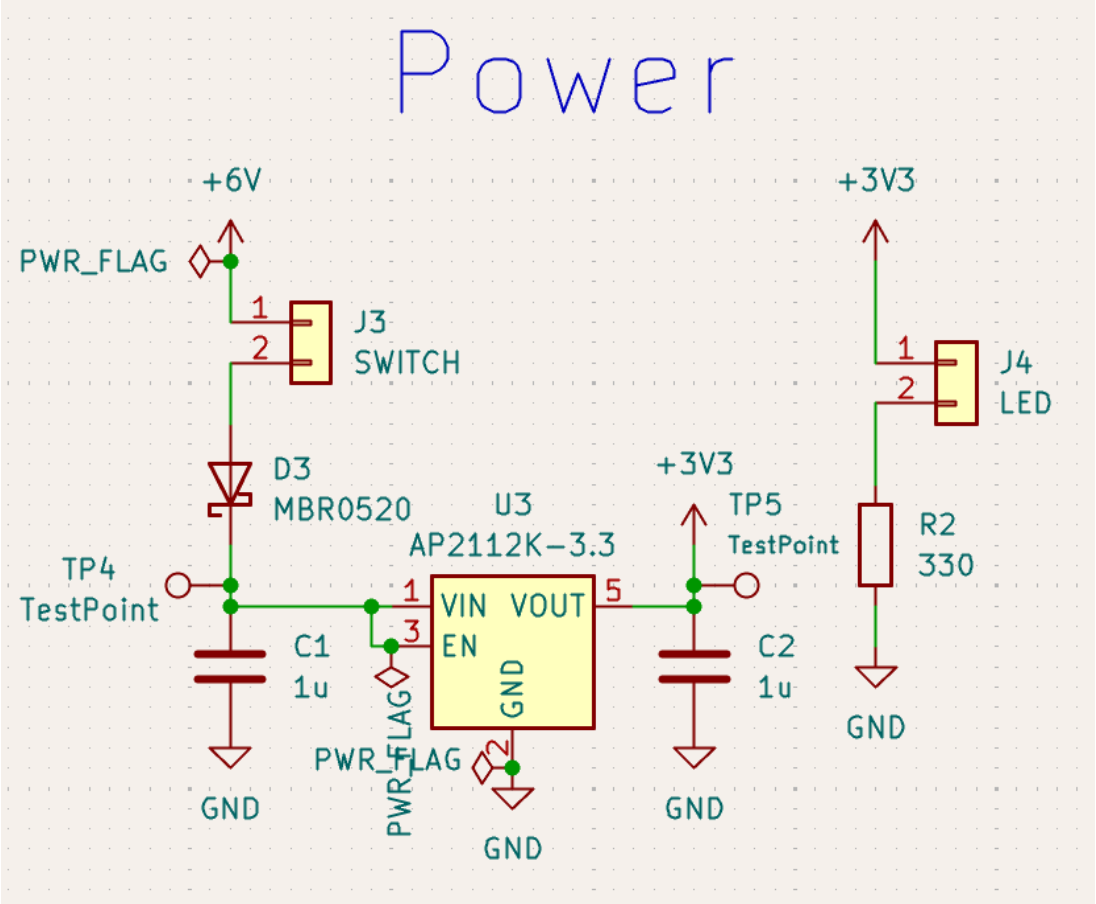

Figure 8: Bat Detector Frequency Module Power Supply Circuit Schematic

2.1.1 AA battery holder

- The battery holder contains four AA batteries to provide voltage to the regulator, amplifiers, and input/output.

2.1.2 Voltage Regulator and Switch/Power Button

- The AP2112K-3.3TRG1 voltage regulator [7] will maintain and distribute a constant voltage to the required components of the circuit. The switch will control when the device is on or off. Its maximum current draw of 600 mA is well within our needs.

2.5 Firmware Design

- The software component of our project is crucial as it is the component that makes our solution unique. In our design, the microcontroller handles audio playback of various pre-recorded bat calls so that the user knows which sounds to listen for while using the device to detect bats. To do so, we modified Adafruit TinyFlash's TrinketPlayer.ino file [8].

2.5.1 Processing File Loader

- Before the MCU can play audio samples saved on our flash memory IC, the audio samples need to be on said flash memory IC. To do so, we modified Adafruit Tinyflash's AudioXfer.pde [9] to accommodate four separate audios.

```
File[] files = {new File("D:/LAST SEMESTER/ECE445-UIUC-Heterodyne/Firmware/WAV Files/FinalFiles/Oskee.wav"),
                new File("D:/LAST SEMESTER/ECE445-UIUC-Heterodyne/Firmware/WAV Files/FinalFiles/EasternRedBat.wav"),
                new File("D:/LAST SEMESTER/ECE445-UIUC-Heterodyne/Firmware/WAV Files/FinalFiles/IndianaBat.wav"),
                new File("D:/LAST SEMESTER/ECE445-UIUC-Heterodyne/Firmware/WAV Files/FinalFiles/BigBrownBat.wav")};

int totalExpected = 0;

byte[][] fileBytes = new byte[files.length][];
for(int i = 0; i<files.length; i++){
  byte[] file = loadBytes(files[i]);
  fileBytes[i] = file;
  totalExpected += file.length;
}

byte[] output = new byte[totalExpected];
int offset = 0;

for (int i = 0; i < fileBytes.length; i++) {
 byte[] file = fileBytes[i];
 int samples = file.length;
 println("file number: ", i, "has mem location: ", offset);
 arrayCopy(file, 0, output, offset, samples);
 offset += samples;
}
println("totalbytes...: ",output.length);
```

Figure 9: File indexer and byte-array accumulator

- In addition to loading these files, we want to know exactly where in the IC they will be stored. So, the Processing 2 script takes our .wav files, turns them into one string of bytes, and then outputs these indexes for us to copy and hardcode into our firmware.

```
  for(int i=0;i<output.length;i++){
//    println(output[i]);
    port.write(output[i]);
    s = readLine();
    if(i % 1000 == 0) print(s);
    if(i % (totalExpected/100) == 0) print(100*i/totalExpected);
  }
```

Figure 10: Byte output to IC via SPI

- By connecting our PCB to an Arduino UNO (or equivalent board), and then connecting that board to a laptop/PC, we can load samples onto our flash memory IC (2.4.3).

2.5.2 ATTiny85 Programming

```
void setup() {
  uint32_t bytes;
  audioNum = 0;
```

Figure 11: Starting with first call

- Upon startup, the device will set the index of the current bat call to the first call stored (index 0). If the button on the device is pressed, the current bat call selected will be played through the speaker.

```
audioIndex[0] = 0;
audioIndex[1] = 134942;
audioIndex[2] = 412466;
audioIndex[3] = 670244;
```

Figure 12: Indexes of each audio value on flash IC

- Since we flash the audio, we know the number of bat calls and the length of each call ahead of time, so these values are hardcoded.

```
PLLCSR |= _BV(PLLE);              // Enable 64 MHz PLL
delayMicroseconds(100);           // Stabilize
while(!(PLLCSR & _BV(PLOCK)));    // Wait for it...
PLLCSR |= _BV(PCKE);              // Timer1 source = PLL

// Set up Timer/Counter1 for PWM output
TIMSK  = 0;                       // Timer interrupts OFF
TCCR1  = _BV(CS10);               // 1:1 prescale
GTCCR  = _BV(PWM1B) | _BV(COM1B1); // PWM B, clear on match
OCR1C  = 255;                     // Full 8-bit PWM cycle
OCR1B  = 127;                     // 50% duty at start
// digitalWrite(3,LOW);


pinMode(4, OUTPUT);               // Enable PWM output pin

TCCR0A = _BV(WGM01) | _BV(WGM00);  // Mode 7 (fast PWM)
if(sample_rate >= 31250) {
  TCCR0B = _BV(WGM02) | _BV(CS00); // 1:1 prescale
  OCR0A  = ((F_CPU + (sample_rate / 2)) / sample_rate) - 1;
} else {                          // Good down to about 3900 Hz
  TCCR0B = _BV(WGM02) | _BV(CS01); // 1:8 prescale
  OCR0A  = (((F_CPU / 8L) + (sample_rate / 2)) / sample_rate) - 1;
}
TIMSK = _BV(OCIE0A); // Enable compare match, disable overflow
pinMode(3, INPUT);
cli();
```

Figure 13: Initialization of MCU to generate PWM of inputted byte data once per interrupt

- When playing audio from the flash, the microcontroller will receive the data on a byte-by-byte basis, with byte reads occurring at every timer interrupt (every 16,000th of a second).

```
ISR(TIMER0_COMPA_vect) {
  if(waiting) return;
  OCR1B = flash.readNextByte();      // Read flash, write PWM reg.
  if(++index >= samples) {           // End of audio data?
    index = 0;
    flash.endRead();
    cli();

    pinMode(3, INPUT);

    waiting = true;
  }
}
```

Figure 14: End of file condition

- Once all bytes have been received and output, the device will disable interrupts and return to the waiting for input state.

```
if(digitalRead(3) == LOW){
  cli();
  waiting = false;
  if(audioNum>3) audioNum = 1;
  samples = audioIndex[audioNum+1] - audioIndex[audioNum];
  pinMode(3, OUTPUT);
  flash.beginRead(audioIndex[audioNum]);

  audioNum++;
  sei();
}
```

Figure 15: Button press condition

- Once the button is pressed, the MCU will enable timer interrupts and begin reading the next batcall. If the last bat call has just been played, the device will circle back to the first index.

2.6. Cost

Delineated in Appendix B is our extensive bill of materials, including all material and labour needs for this project, as well as their cost. We estimate our project will cost $116.76 after labor. If we consider the materials we will be able to retrieve for free from the E-Shop Self Service, our estimated project cost is $103.28 after labor.

3. Testing & Verification

3.1 Oscillator Module

The oscillator module is rather simple to test. Placing an oscilloscope probe on the second op-amp's output allows us to measure a periodic wave. The cursor function provided by the scope (or measurement function) can be used to obtain an accurate measurement of the frequency for a certain in-circuit resistance between 0.1 kΩ-1.1 kΩ.

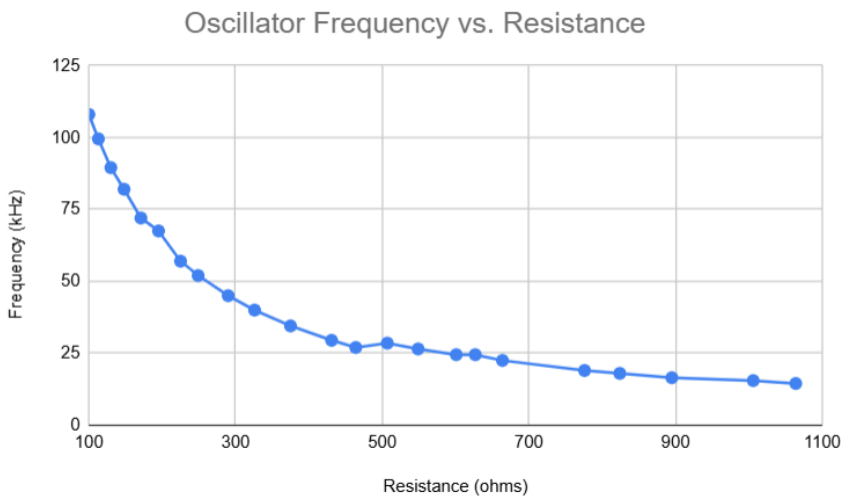

Figure 17: Oscillator Response vs. Resistance

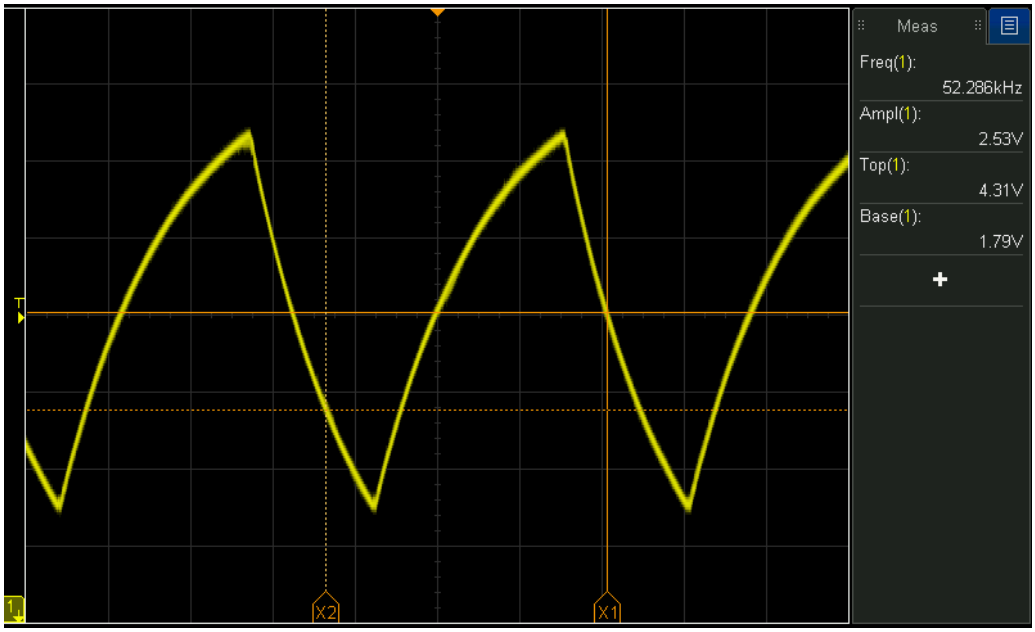

Figure 18: Oscillator Response on Breadboard, Oscilloscope View

Testing on our breadboard, we found a range of 14.5 kHz to 109.5 kHz, which is within our expectations. The amplitude of the oscillations was in the range of 2-3 V, which is more than adequate for the mixing stage that comes next in the circuit. This frequency range was measured while accounting for the resistance and capacitance in the other modules. Our PCB oscillator was found to exhibit a similar response. Its higher slew rate than our breadboard op-amp meant we achieved amplitudes of 6 V for oscillation, outputting a true square wave. The frequency range was found to be 14.7 kHz-117.0 kHz, exceeding our expectations.

3.2 Mixer Module

Detailed testing of this module has been achieved so far by setting the oscillation frequency to a certain value (say, 40 kHz) and piping a low-amplitude signal (<100 mV) of the same frequency through the microphone amplification stage we have on board, using the waveform generator at our bench. This low-amplitude signal method is chosen to replicate the behavior of the ultrasonic microphone. With adequate gain in the microphone amplification stage, we achieve equal magnitudes in the mixer stage, which is ideal for mixer operation. Figure 19 shows the oscillator (set to ~52.3 kHz) mixed with a sinusoid with frequency 52 kHz and amplitude 100 mV, producing a mixed signal with a frequency around 250 Hz.

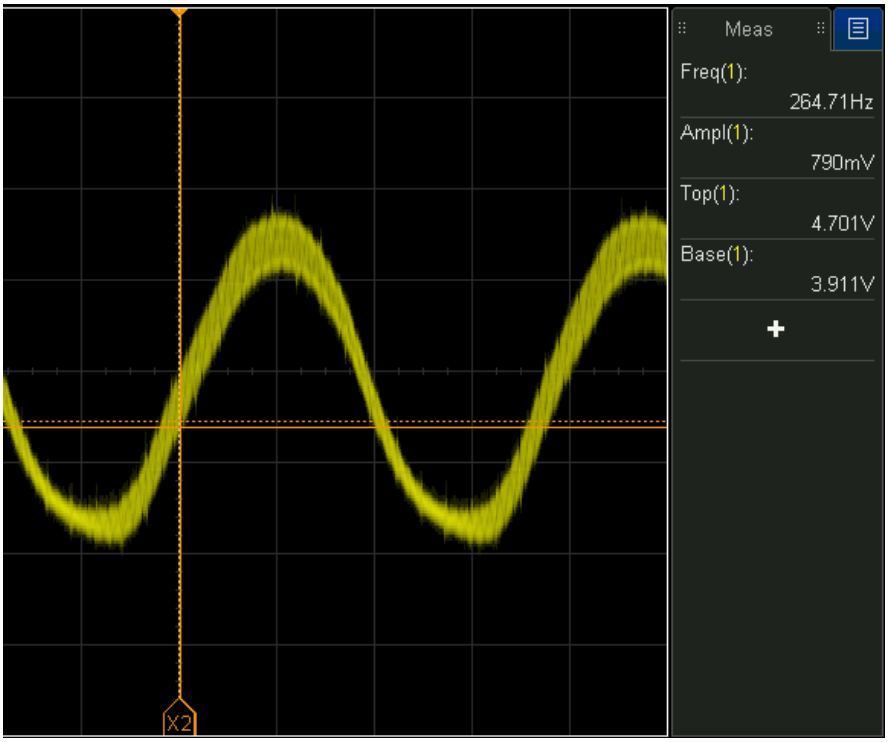

Figure 19: Mixer Response

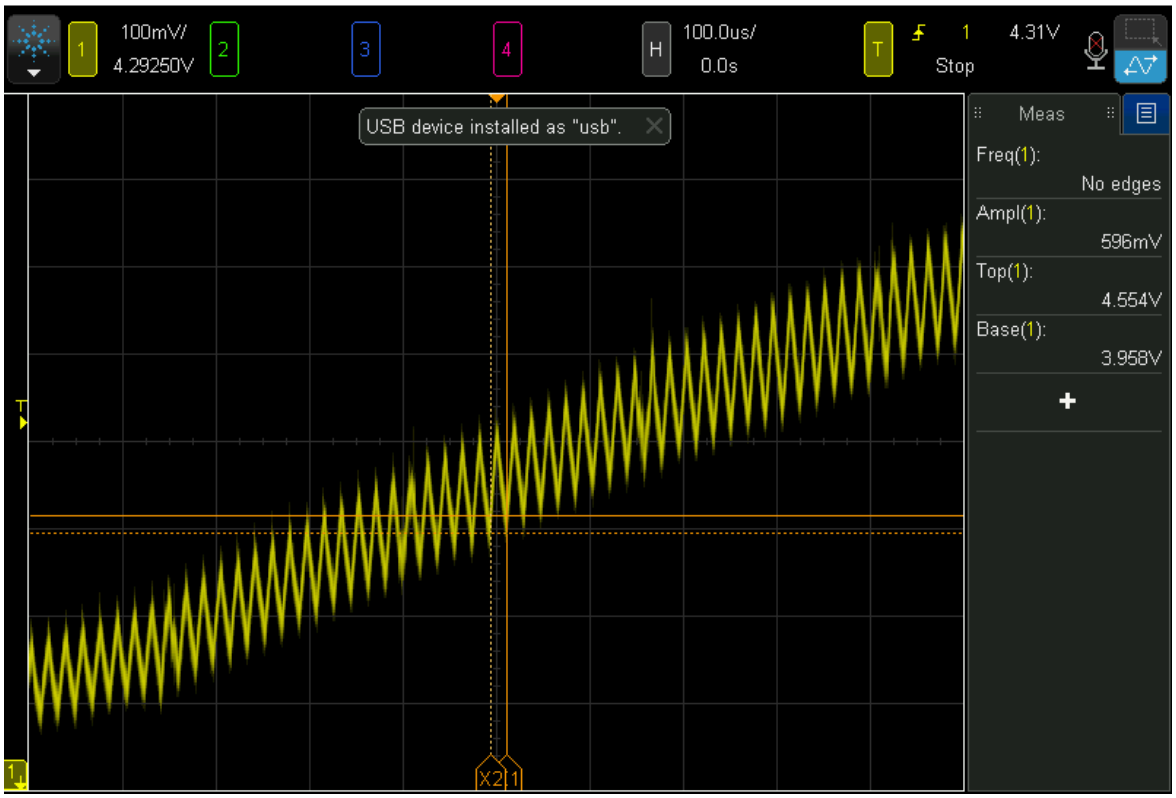

Figure 20: Mixer Ripple Example

On our breadboard, we have achieved results that are very similar to the ones seen in Figures 18 and 19. The high-frequency oscillation is attenuated greatly by the low-pass filter and is inaudible through our chosen speaker, regardless. The amplitude of oscillations after our amplifier stage is low, around 800 mV. According to the LM386N audio amplifier datasheet [6], input waveforms of up to 400 mV are specified for proper operation. Despite the amplification in this stage being low, it is not an issue for our operations. These oscillations being centered around 4-5 V does mean that a direct-current (DC) blocking capacitor is necessary on the mixer output to achieve the appropriate range of 0-1 V for use in the audio amplifier. Everything found here is identical to what we observed on our finished printed circuit board (PCB).

3.3 Audio Loading Software

To ensure that there are no hardware issues during the flashing of the memory IC, the firmware on the Arduino UNO during flashing has a simple, built-in memory-value checker, which we call once every 50 bytes written.

```
uint8_t test = flash.readByte(address);

if(test != data){
  Serial.print(address);
  Serial.println(" FAILED VERIFY");
  return false;
}
return true;
```

Figure 21: Simple byte tester

Then, as a final sanity check, we also print an entire page of values to the serial monitor to ensure that the sound bytes in memory appear as they should.

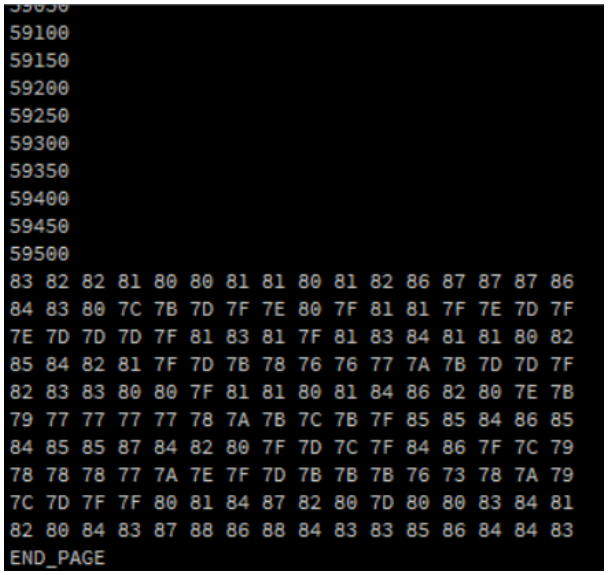

Figure 22: Manual page verification

4. Conclusion

4.1 Accomplishments

The designed and implemented device was able to load and play audio samples, receive and amplify microphone input, generate a range of and amplify oscillating voltages, mix these voltages with the microphone input, and switch between outputting the mixed voltages or the audio samples, all at a range of volumes.

4.2 Contributions

We hope that our project will help to raise awareness of local bat populations in Central Illinois, leading to some people being more courteous toward the environment. It is also our hope that the device will foster more socialization during Central Illinois Bat Festival Bat Walks, result in a better experience for Bat Walkers, and, hopefully, result in a more educational Bat Festival.

4.3 Ethical considerations

Our team believes that our product lacks ethical concerns. The only interaction that our product has with the outside world is in amplifying the high-frequency sounds of bats out in the wild through the use of a microphone. Nothing is sent back out to the bat to communicate with it or otherwise interrupt nature. No recording is enabled by the device, eliminating the possibility of recording without the permission of others. This device only amplifies inaudible sound waves, so it cannot be used to listen in on private conversations. Component choices may be obtained via reverse engineering of existing products, such as the Magenta Bat4, but no circuit or intellectual property is to be copied without permission.

4.4 Future work

While our solution is functional and provides a good baseline, there are improvements that can be made by any future teams who may decide to pick up this project in the future. The primary issue with our design was sound volume and quality, as well as short range. A better microphone could improve both range and sound quality of our device. A higher quality speaker could also improve the sound quality and volume, as it was quite difficult to hear our output through the device's speaker, but it was much easier to hear the output through plugged in

headphones. Further, some additional signal processing could lead to a higher quality output for the device. Future teams can achieve this through improvements to our mixer and oscillator implementations, or through the use of a pre-existing solution such as the SA612A [10]. Finally, we would like to see a more ergonomic chassis for the device. This can be achieved through small improvements, such as additional rounded edges, as well as optimized volume.

5. References

[1] “SST25VF080B Data Sheet”, Microchip Technology, Chandler, AZ, February 2020.

[2] “TS5A3159 1-Ω SPDT Analog Switch datasheet (Rev. D),” Texas Instruments, Dallas, TX, February 2015.

[3] “TLV9054 Op Amp Data Sheet,” Texas Instruments, Dallas, TX, February 2024.

[4] “Relaxation oscillator circuit (Rev. A),” Texas Instruments, Dallas, TX, October 2024.

[5] “SPVA1A0LR5H-1 Data Sheet”, Syntiant, Irvine, CA, November 2024.

[6] “LM386 Low Voltage Audio Power Amplifier Data Sheet,” Texas Instruments, Dallas, TX, August 2023.

[7] “AP2112 Low-Dropout Voltage Regulator Data Sheet,” Diodes Inc., Plano, TX, June 2017.

[8] “TrinketPlayer.ino”, Adafruit Industries, Github, October 2013. [Online]. Available: https://github .com/adafruit/Adafruit_TinyFlash/tree/master/examples/TrinketPlayer

[9] “AudioXfer.pde”, Adafruit Industries, Github, October 2013. [Online]. Available: https://github.com/ adafruit/Adafruit_TinyFlash/tree/master/Processing/AudioXfer

[10] “SA612A Double-balanced mixer and oscillator product data sheet,” NXP Semiconductors N.V., Eindhoven, Netherlands, June 2014.

Appendix A

Table 1: Bat Call Audio Demo Module Requirements and Verification

Requirements	**Verification**
Allow for on-demand user input to hear pre-recorded audio through the microcontroller	• Prompt pre-recorded audio playback through button • Pre-recorded audio should play if and only if the button is pushed, and only for the duration of the sound bytes
Allow the user to cycle between multiple pre-recorded samples to listen to	• Press button to listen to chosen audio • Press button to listen to next sample after audio has ended • Repeat step one: each audio sample should not be played more than once until all other audio samples cycled

Table 2: Bat Detector Frequency Module Requirements and Verification

Requirements	**Verification**
Produce an oscillating signal that can be controlled within the ranges of 15 kHz to 100 kHz	• Probe the oscillator test point with the oscilloscope • Verify the frequency ranges from (at least) 15 kHz to 100 kHz by tuning the potentiometer
Mix down ultrasonic frequencies provided by the microphone to frequencies audible by the human ear	• Use ultrasonic calibrator to send 40 kHz pulses to microphone • Set device frequency dial to same frequency as ultrasound calibrator • Probe mixer test point with oscilloscope, measure signal to verify frequency subtraction

Table 3: Audio Output Requirements and Verification

Requirements	**Verification**
Allow for the user to plug in and use external headphones and speakers to listen to bat calls	● Plug in an external audio output device via the auxiliary jack ● Audio should be heard through plugged in audio output device
Disable use of the speaker while an external output device is plugged into the audio jack	● Plug device into the auxiliary jack ● Audio should no longer be heard through the built-in speaker
Output audible mixed-down ultrasonic signal	● Use ultrasound calibrator to send 40kHz pulses to microphone ● Set device frequency dial to same frequency as ultrasound calibrator ● Output should be a low frequency beep
Amplify audio signal received from the mixer and microcontroller	● Set up two oscilloscope traces, one with the input waveform and the other with the output waveform ● Amplitude of the outgoing signal should be greater than the incoming signal, with the same general shape

Table 4: Power Supply Requirements and Verification

Requirements	**Verification**
Allow the user to control when the device is powered on or off	● Turn the power switch on ● Use the button to play digital/analog audio through speaker ● Turn power switch off, device should only output when power switch is on
Provide 3.3 V to the microcontroller, flash memory, and microphone	● Use a multimeter to ensure that each of the microcontroller, flash memory, and microphone are receiving 3.3 V to their respective power pins
Provide 6 V to audio amplifier and op-amp	● Use a multimeter to ensure that amplifier and op-amp are receiving 6 V to their respective power pins

Appendix B

Table 5: Itemized List of Components, Labor, and Cost

Part/Item	Quantity	Description	Commercial Cost	In Self Service?
GRM2165C1H472JA01D	2	4.7 nF (±5%) SMD Cap	$ 0.28	No
C0805C223G4HACTU	1	22 nF (±1%) SMD Precision Cap	$ 0.30	No
885012207096	2	47 nF (±10%) SMD Cap	$ 0.20	No
CC0805KRX7R9BB104	3	0.1 uF (±10%) SMD Cap	$ 0.24	Yes
C0805C105K3RACTU	5	1 uF (±10%) SMD Cap	$ 0.60	Yes
UWT1C221MCL1GS	1	220 uF (±20%) Electrolytic SMD Cap	$ 0.40	No
RC0805FR-0710RL	1	10 Ω (±1%) SMD Resistor	$ 0.10	No
CRCW0805100RFKEA	1	100 Ω (±1%) SMD Resistor	$ 0.10	No
RMCF0805JT330R	1	330 Ω (±5%) SMD Resistor	$ 0.10	Yes
RMCF0805FT470R	4	470 Ω (±1%) SMD Resistor	$ 0.10	Yes
CRCW08051K00FKEA	7	1 kΩ (±1%) SMD Resistor	$ 0.70	Yes
RC0805FR-073K3L	1	3.3 kΩ (±1%) SMD Resistor	$ 0.10	Yes
CRCW080510K0FKEA	5	10 kΩ (±1%) SMD Resistor	$ 0.40	Yes
RMCF0805FT33K0	1	33 kΩ (±1%) SMD Resistor	$ 0.10	Yes
MBR0520LT1G	4	20 V 0.5 A Schottky Diode	$ 0.69	No
BHR-06-VUA	1	2x3 IDC Connector	$ 0.22	Yes
10129378-903001BLF	2	1x3 Pin Headers	$ 0.34	Yes
0022232021	4	1x2 Molex Male Connectors	$ 0.48	Yes
0022012021	4	1x2 Molex Female Connectors	$ 0.45	Yes
22232031	3	1x3 Molex Male Connectors	$ 0.66	Yes
22012031	3	1x3 Molex Female Connectors	$ 0.54	Yes
0039000038	17	Molex crimps	$ 0.10	Yes
SPVA1A0LR5H-1	1	Ultrasonic Microphone	$ 1.18	No

SPKM.23.8.A	1	Speaker	$ 1.16	No
SJ1-3555NG	1	3.5 mm Auxiliary Audio Barrel Jack	$ 0.98	Yes
PJ-002A	1	Power Barrel Jack	$ 0.47	Yes
PTV09A-4030F-A102	1	1 k Rotary Potentiometer	$ 0.89	No
TC33X-2-103E	1	10 k Trim Potentiometer	$ 0.25	No
PTV09A-4020U-B103	1	10 k Rotary Potentiometer	$ 0.89	No
1825910-6	1	Push button	$ 0.26	Yes
OS102011MS2QN1C	3	SPDT slide buttons	$1.56	Yes
TS5A3159DCK	1	Electrical SPDT	$0.54	No
ATTiny85-20PU	1	Microcontroller	$ 1.66	Yes
TLV9054	1	Op-amp	$ 1.04	No
AP2112K-3.3	1	Voltage regulator (3.3V)	$ 0.22	Yes
LM386N-1	1	Audio amplifier	$ 1.05	Yes
W25X40CLSNIG	1	Flash storage for sounds	$ 0.56	No
BAT-HOLDER-4XAA	1	Battery holder with terminals	$ 0.59	Yes
C503B-GCN-CY0C0792	1	Green LED	$ 0.35	Yes
N/A	6	Stranded copper wires	$ 0.10	Yes
N/A	2	PCBs (Main and Mic)	$ 0.50	No
N/A	~100g	PLA Filament	$ 2.00	No
Machine Shop Labor			$ 0.00	N/A
Labor: Evan		24 minutes of soldering	$ 30.00	N/A
Labor: Kyle		18 minutes of soldering and 6 minutes of connecting auxiliary components and fastening in case	$ 30.00	N/A
Labor: Bill		18 minutes of soldering and 6 minutes of programming MCU and flash memory and testing	$ 30.00	N/A
Total Cost			$ 116.76	$ 103.28