CONTINUOUSLY RECORDING MICROPHONE

By

Brian Song

James Chen

Final Report for ECE 445, Senior Design, Spring 2019

TA: Dong Wei

01 May 2019

Project No. 79

Abstract

Our senior design is to create a device that can continuously record audio from the user's surroundings and save a 15 second clip with a button. In the design, we implement manual and automatic saving as well as cropping the recorded audio. The manual save is done by pressing a button while the automatic save triggers on when the volume gets too loud or a strong downward force was applied to the user. Audio cropping allows the user to cut unwanted parts of audio files, and a speaker on the device gives instant playback to the user on the cropped audio file. For the display, an OLED was used to save space. In our final implementation of the design, we were able to get the manual save and cropping functions to work, while the automatic saves and speaker didn't work.

Contents

1. Introduction 1

1.1 Functionality 1

1.2 Subsystem Overview 1

2 Design 3

2.1 Design Procedure 3

2.2 Design Details 5

3. Design Verification 9

3.1 Power Subsystem 9

3.2 Control Unit 9

3.3 User Interface 9

3.4 Audio Interface 9

3.5 Sensor 10

4. Costs 12

4.1 Parts 12

4.2 Labor 13

5. Conclusion 14

5.1 Accomplishments 14

5.2 Uncertainties 14

5.3 Ethical considerations 14

5.4 Future work 15

References 16

Appendix A Requirement and Verification Table 17

1. Introduction

In the modern day, everyone has smartphones that can record audio whenever they want. However, this is only for pre-planned moments when people know they want to record. Sometimes, there are points in time you'd wish you were recording but missed the chance because you weren't ready or didn't expect to. It's moment like these that are unplanned, spontaneous, or unexpected that our device wants to capture.

Our device's purpose is to allow easy recording of audio without needing to click 'start'. That is, the device will be constantly recording without saving the audio files until told to do so. In addition, the device will have a second feature that allows automatic saving when the user is in danger or in a quarrel for safety purposes. Both features will record clips that would never be recorded otherwise. Finally, the device will be able to crop clips that have already been recorded.

1.1 Functionality

In our original project description, we wanted our device to be small enough to be handheld or wearable as a clip-on accessory. However, due to time constraints and design errors regarding the PCB, we were only able to implement the device on a breadboard and partially on the PCB. For our final design, we were able to implement the manual saving and cropping functions. The user would press a button to save the last 15 seconds of audio captured by the microphone and another button would allow the user to crop a clip. For cropping, the user would begin the cropping process by clicking the crop button. They would then use the knob to choose which clip they would like to crop and press the crop button again to confirm. Afterwards, they would use the knob to crop from the beginning of the clip and the end of the clip, pressing the crop button in between to confirm how much they would like to crop. The device would then save the cropped clip in a separate file for the user to listen to. We were able to get the accelerometer to trigger an autosave; however, we couldn't integrate it with the rest of the design due to RAM limitations on the microchip. Finally, we failed to implement the speaker module along with the audio-based autosave. Due to running out of time, we wanted to focus on the main functionality of the device. As a result, we decided to leave out these functions.

1.2 Subsystem Overview

Figure 1 shows our device has five separate submodules: User Interface, Control Unit, Power, Audio Interface, and Sensory Module.

Our user interface consists of the knob, two buttons, and an OLED. The knob is used to choose a clip to crop and how much to crop of that clip. One button is used to save a clip, and the other is used to control the cropping function. The OLED is used to give the user feedback on which clip they are cropping and how much of it they have cropped. It is not used when during passive operation or while saving a clip since it is unnecessary and saves power.

The control unit consists of the microcontroller, the SRAM buffer, and the microSD module. The microcontroller is used to control the functionality of the device. It controls where the audio data coming from the microphone should go, what should be displayed on the OLED, when to crop, and when to save. The SRAM buffer is used to store the unsaved audio because of the limited amount of RAM on the microcontroller. The microSD card is used to store the clips once the user has decided to save.

The power module consists of a microUSB module, a Li-ion charger, a rechargeable Li-ion battery, and a voltage regulator. The battery charger was chosen such that it would be able to be powered by the 5V output of a microUSB and charges a 3.7V Li-ion battery. Since all of our parts run on 3.3V and the battery outputs a nominal voltage of 3.7V, we needed to use a voltage regulator to step down the voltage from 3.7V to 3.3V.

The audio interface consists of a microphone, a speaker, and a Digital to Analog Converter (DAC). The microphone is used to record audio from the user's surroundings. The microphone outputs an analog signal so we used the microcontroller's internal ADC to sample the audio. In order to play the audio on a speaker, we needed a DAC to convert our digital data into an analog signal. The speaker is used to playback audio while the user is cropping so they know which parts of a clip would be cropped.

The sensory module consists solely of the accelerometer. The accelerometer is used to detect when the user falls as an indication of danger. When a significant downward force is detected from the accelerometer, the past 15 seconds of audio would be saved onto the SD card as a safety measure.

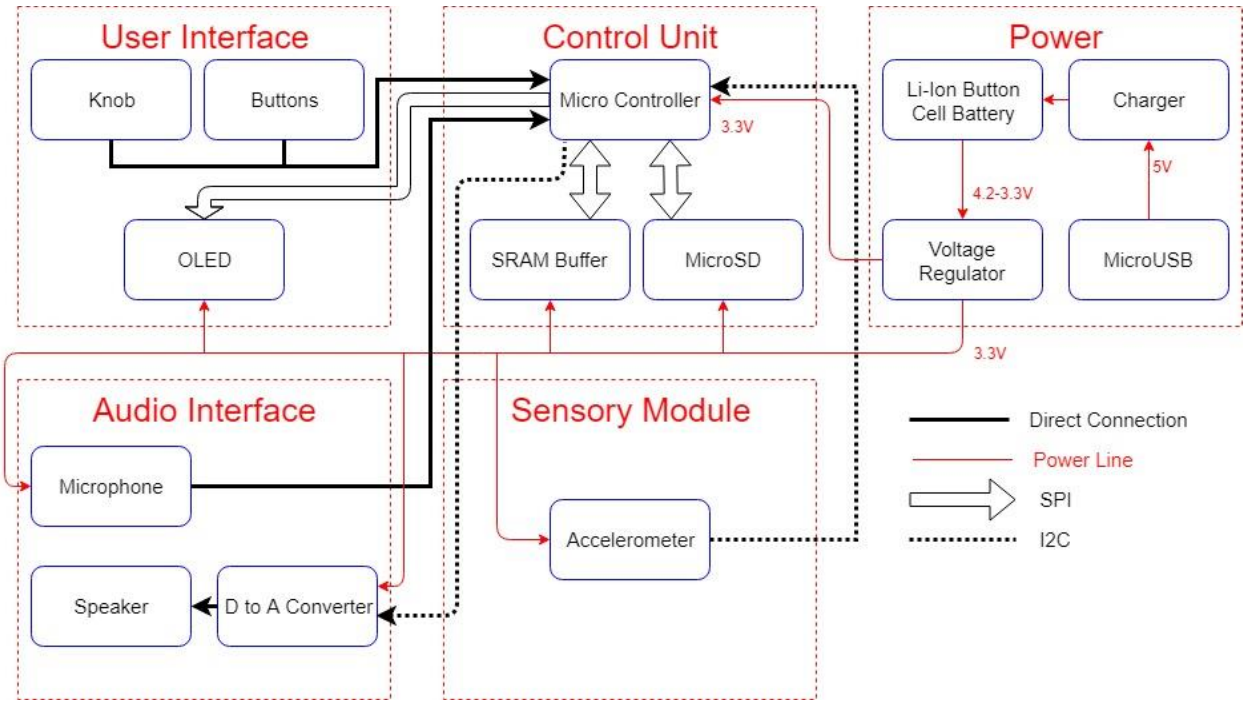


Figure 1 High Level Block Diagram of Device

2 Design

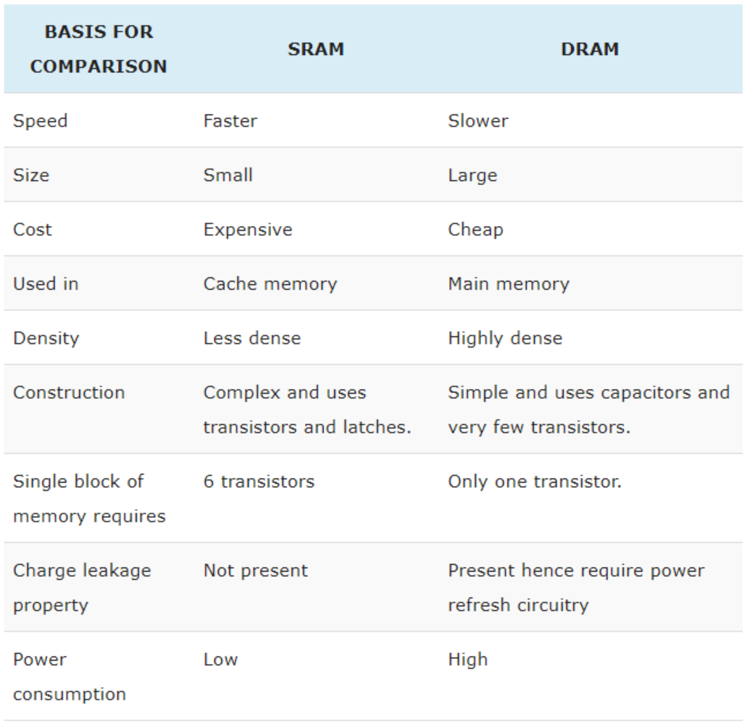
2.1 Design Procedure

For our power module, we wanted to use a rechargeable battery to make it easy for the user to power the device and since we wanted the device to be small, we chose a coin cell battery to save space. Because of this, we decided to use a 300mAh 3.7V Li-ion battery. For an alternative, we could have used replaceable AA batteries, but that would take up a lot more space and would be more inconvenient for the user. The batteries would need to be changed every week or so and there is no control on when the device would suddenly stop due to dead batteries. We chose to charge the battery with a micro USB port since it's something that most people would have to charge their phones. Choosing the charger was just a matter of finding a part that fits the above specifications.

When choosing a microcontroller, we originally went with the ATmega328pb because of its two SPI and I2C buses. We thought that each device needed its own bus to function, but later discovered that the buses can be shared if none of the devices needed to communicate with the microcontroller at the same time. Because of this, we made a lot of errors in design choices that will be stated below. Not only that, but there is no Arduino that uses the ATmega328pb, so there is no bootloader that fully supports the microchip. After discussing this problem with our TA, we decided to change from the ATmega328pb to the ATmega328 since it came on the Arduino Uno and we could simply take it off and put it onto our PCB.

Regardless of what microcontroller we use, we need an external set of memory to store audio and clips. We knew that we wanted to use an SD card, so the user could remove it and transfer the files to their computer. However, we also chose to have an extra set of RAM on board simply because the microSD consumes 100mA while writing. Since we would be constantly writing to it, the battery would run out in less than three hours. Table 1 shows the tradeoffs between using SRAM or DRAM. In the end, we decided to use an SRAM due to the speed being faster and the power consumption being lower. Since we needed to store 15 seconds of audio sampled at 8kHz with a bit depth of 8 bits, the SRAM needed to be at least 120kB large. Regarding the audio bitrate, we chose to use 8kHz x 8 bits after consulting with our professor and researching telephone communications standards [1].

Table 1: SRAM and DRAM Differences [2]



BASIS FOR COMPARISON	SRAM	DRAM
Speed	Faster	Slower
Size	Small	Large
Cost	Expensive	Cheap
Used in	Cache memory	Main memory
Density	Less dense	Highly dense
Construction	Complex and uses transistors and latches.	Simple and uses capacitors and very few transistors.
Single block of memory requires	6 transistors	Only one transistor.
Charge leakage property	Not present	Present hence require power refresh circuitry
Power consumption	Low	High

When designing the user interface, we knew that we needed at least two buttons, at least one knob, and a display of some sort. For the knob, we had a choice between using a potentiometer or a rotary encoder. While a potentiometer has a simpler interface with only one output pin, they also can't rotate infinitely. For our design, we decided it was easier for the user if we had a knob that could rotate indefinitely for cropping and choosing clips. For the display, our original design used a four piece seven-segment display. While a seven-segment display is bulky and relatively hard to interface with, we thought that it was our only option since all of our serial buses were taken. After finding out that multiple devices can use the same bus, we switched our design to use an OLED because it took up less space and could display more information to the user.

For the microphone, we chose to use a MEMS microphone simply because of how small it was. To add to the simplicity, we chose a microphone that would output its data as an analog signal and use the internal ADC on the microcontroller to convert from analog to digital. Both the microphone and the speaker needed to record frequencies up to 4kHz since we were sampling at 8kHz, so that was taken into consideration as well.

Originally, we used a standalone chip as our accelerometer. After doing some preliminary research, we found that the accelerometer would need to record forces up to at least 4g [2]. After spending weeks of trying to get the accelerometer to work on a breadboard, we switched to a premade breakout board and were able to get it working.

2.2 Design Details

Shown in Figure 2 is the schematic of the battery charger. IN is connected to the 5V output of the microUSB breakout and OUT is connected to the voltage regulator. EN1 and EN2 are tied to ground to set the maximum charging rate to 100mA since we didn't want to overheat the battery. Because of this, we were able to leave ILIM floating. The resistor used for TS was set at 10k as specified by the datasheet when not using a thermistor. The resistor used for ISET was calculated using

$$I_{SET} = \frac{K_{SET}}{R_{SET}} \tag{1}$$

where ISET is the charging current, KSET is the current factor, and RSET is the resistor value used to connect ISET to ground. KSET is 870 as stated in the datasheet and ISET is 100mA, which gives a value of 8700 for RSET. Since we didn't have any 8.7k resistors to use on a breadboard, we chose a value of 8.66k as the next best value.

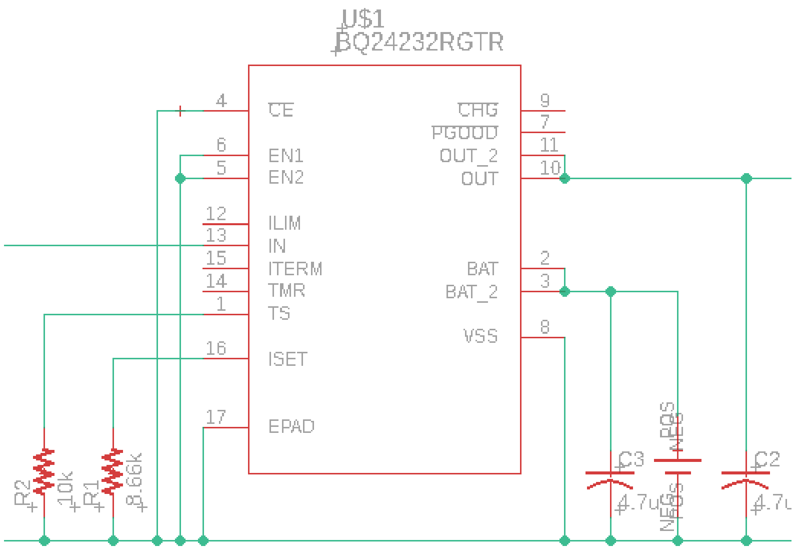


Figure 2 Li-ion Battery Charger

Shown in Figure 3 is the schematic used to interface between the microphone and the microcontroller. This is needed since the microcontroller can't read negative voltage values. Since the peak to peak of the microphone is only around 100mV, we chose to use the microcontroller's internal ADC reference of 1.1 instead of 3.3V. Because of this, the analog signal needs to be centered around 0.55V since that will represent silence in the audio. We do this by using a capacitor to get rid of the DC offset of the microphone. Then, we use a voltage divider to center the AC signal around 0.55V to get audio data that can be read by the microcontroller and played in a wave file. In the simulation, V1 outputs a sine wave with a DC offset of 0.67V, an amplitude of 100mV and a frequency of 1000Hz to represent the output of the microphone. Figure 4 shows the voltage in between R1 and R2. As seen from the figure, the voltage is centered around 0.55V and preserves the AC qualities of V1.

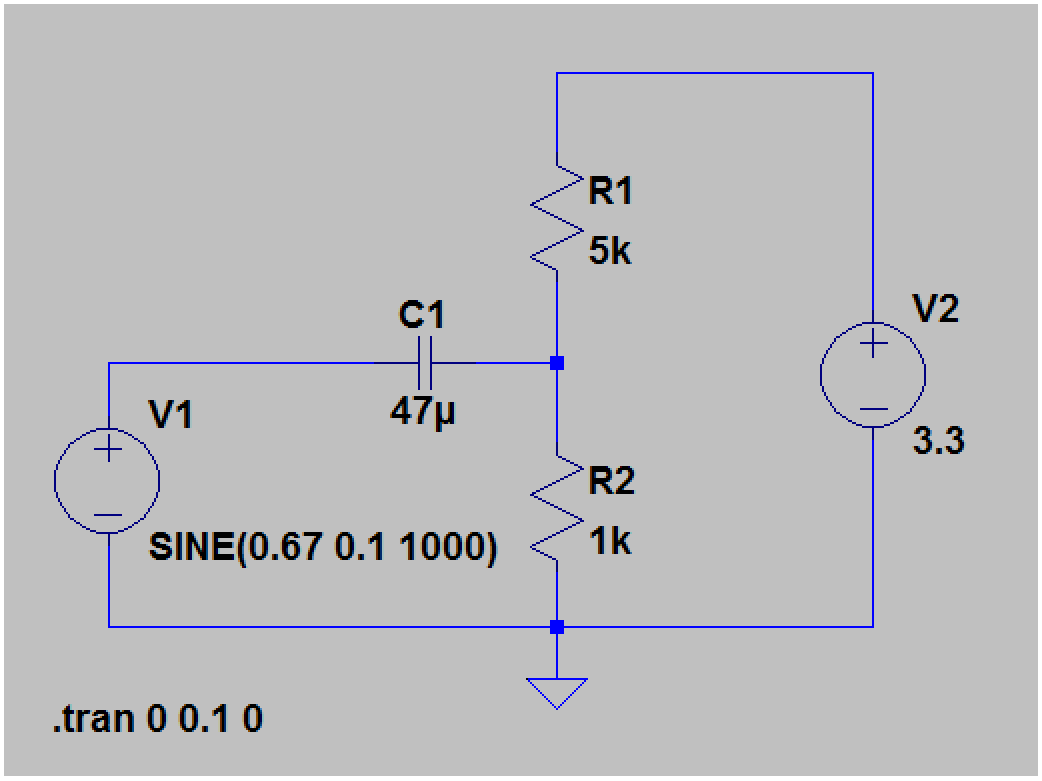


Figure 3 Microphone Interface

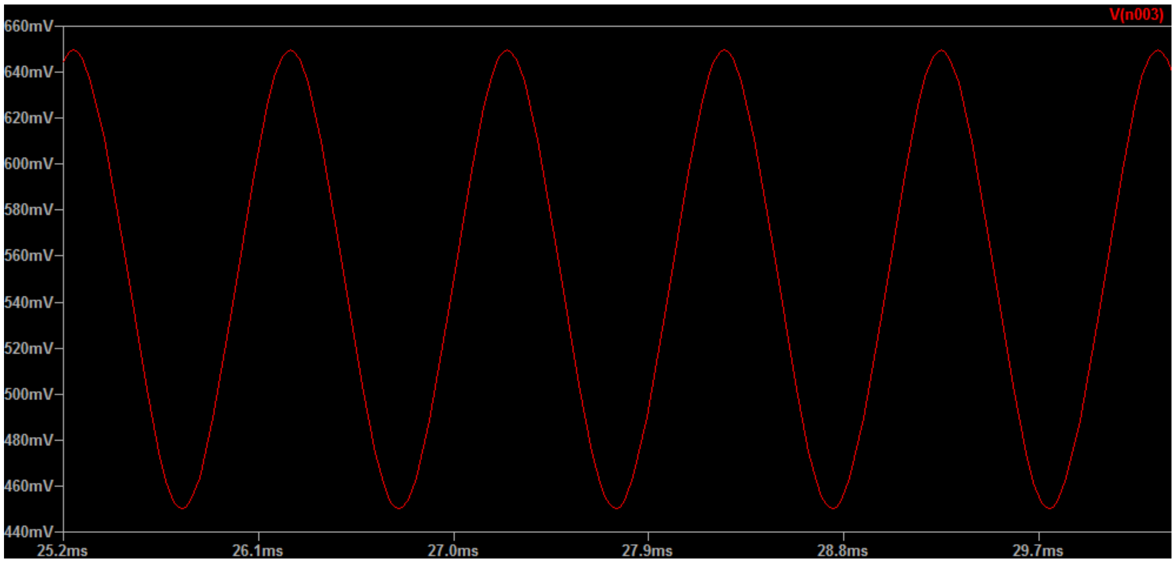


Figure 4 Simulation of Schematic in Figure 3

Shown in Figure 5 is the flowchart of device operation when the device is passively recording audio. If no buttons are pressed, the device simply takes audio from the microphone, puts it into the SRAM in the address specified by Address, increments Address and sets it to 0 if it hits 120000, and loops back to sampling audio. The reason Address needs to reset at 120000 is because 15 seconds of audio sampled at 8kHz gives a total of 120000 samples. After a byte of audio is transferred to the SRAM, the microcontroller then checks if either the save button or the crop button is pressed. Each respective button moves the program into their respective functions.

Shown in Figure 6 is the flowchart of the saving function. Since the buffer we use is 100 bytes large, we need to fill that buffer 1200 times before the entire 15 seconds is transferred. The only caveat we need to be careful of is when the address counter hits 120000. At that point, we need to stop communication with the SRAM and reset the address to 0. The buffer is chosen such that it is as large as it can be such that the device still has enough RAM to perform its other functions. The larger the buffer is, the faster the transfer will be. This is because every time the buffer is filled, we need to stop communication with the SRAM, move the data to the SD card, and resume communication with the SRAM.

Figure 7 shows the flowchart of the cropping function. It is essentially the same as the saving function except it doesn't need to check if address goes above 120000. The front crop cannot go below 0 bytes and the back crop cannot go above 120000 bytes. When the fully cropped clip has been transferred, the function ends, and the device returns to passively recording audio.

While cropping, the OLED display is turned on to give the user some information on what is going on. It shows the user which clip they are cropping and shows how much of the clip they are cropping. When the cropping function has finished, the OLED display turns off again to save power.

It is important to note that the device does not continue recording audio while saving or cropping. This means that the user cannot press the button immediately after saving or cropping because the audio in the SRAM will not represent the past 15 seconds of audio and will be corrupted by the audio data that was in the SRAM previously.

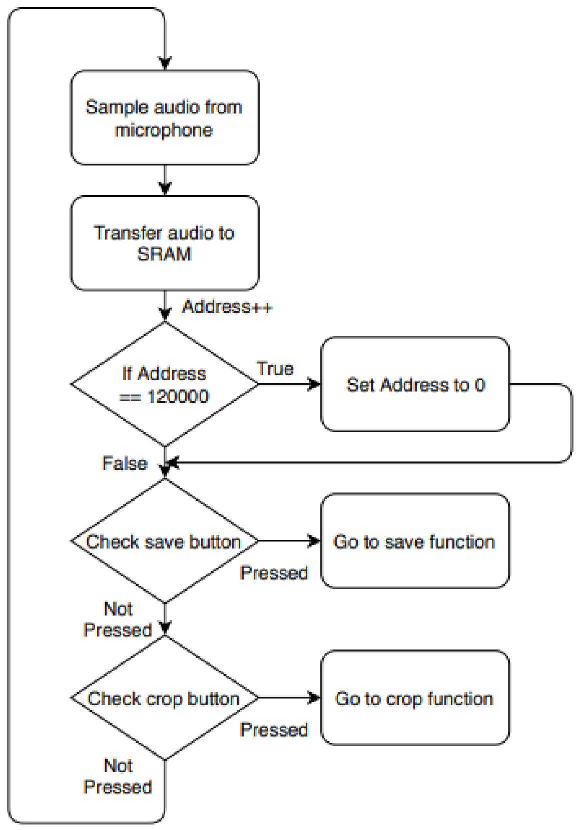


Figure 5 Passive Audio Recording

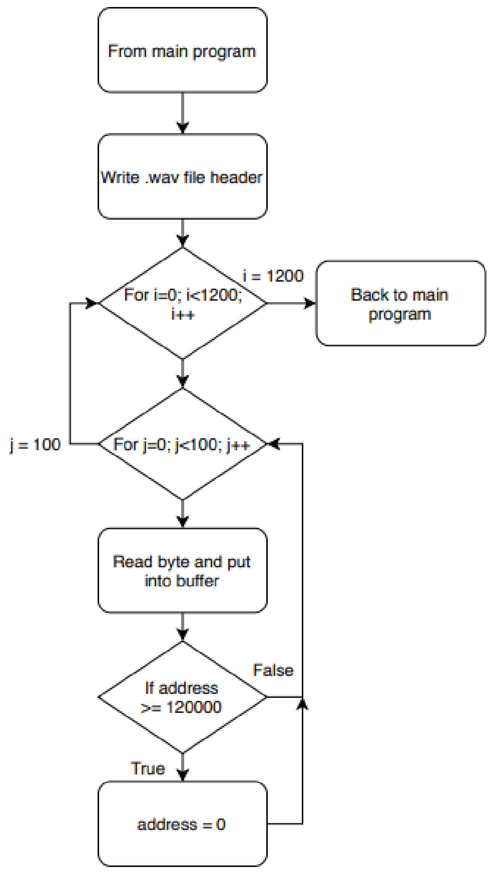


Figure 6 Saving Function

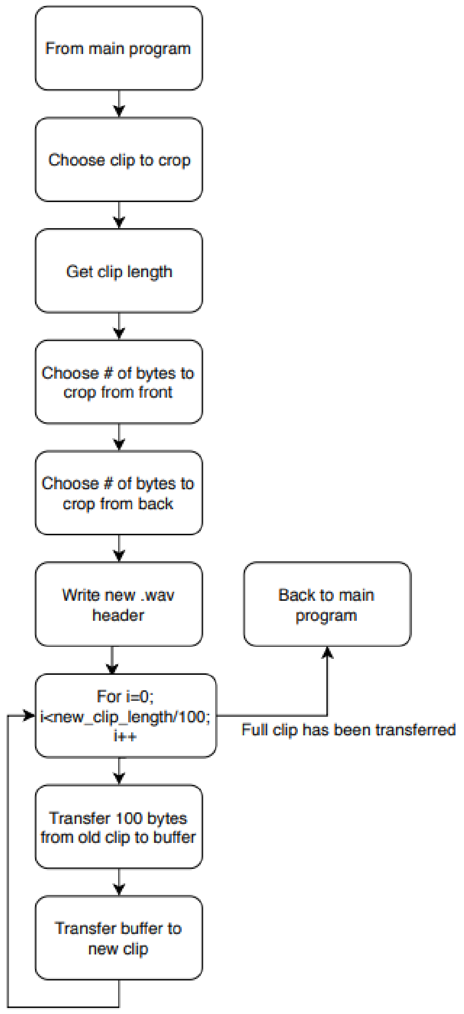

From main program
Choose clip to crop
Get clip length
Choose # of bytes to crop from front
Choose # of bytes to crop from back
Write new .wav header
Back to main program
For i=0; i<new_clip_length/100; i++
Full clip has been transferred
Transfer 100 bytes from old clip to buffer
Transfer buffer to new clip

Figure 7 Cropping Function

3. Design Verification

The final verification of our device was a full test consisting of having the device record 15 seconds of audio and saving it to the microSD. Then, having the device retrieve the file to be cropped and creating a new file that holds the cropped portion. Each component was also tested separately to ensure that each part was fully functional separately.

We were unable to integrate all subsystems together. The power subsystem was not successfully verified and not all submodules that worked could be connected with the control unit. Specifically, the integration of the sensor module could not be integrated with the control unit due to RAM restrictions on the microcontroller. By having the main record, save, and crop functions along with the user interface, the microchip could not handle an extra check from the accelerometer.

3.1 Power Subsystem

We were unable to verify if the power subsystem worked since we did not manage to implement the charger properly into our physical design. Thus, in our final product, there was no power system that was providing the power to the device. A more detailed explanation for each part of the power subsystem can be found in appendix A.

3.2 Control Unit

The control unit contains all the functions that allows the device to record, save and crop audio. The recorded audio is saved onto the microSD and retrieved back to the microchip for cropping.

3.3 User Interface

Each part of the UI was tested separately. Once each part of the UI was verified, this subsystem was tested along with the control unit.

3.4 Audio Interface

The microphone was tested by recording audio with the microphone and checking the output signal. In figure 8, the y axis represents a normalized voltage value where 0 is 0V and 1 is 1.1V. The highest peak to peak shown in the figure was around 120mV (highest at 0.461mV and lowest at 0.352mV). As a result, we didn't implement in a microphone autosave because the peak to peak magnitudes between loud sounds and soft sounds were too small. This resulted in an unreliable autosave mechanic as it was uncertain if a jump in audio was truly loud. The speaker was not implemented in our final design due to incorrect circuit design and lack of time. A more descriptive analysis of the failure is described in appendix A.

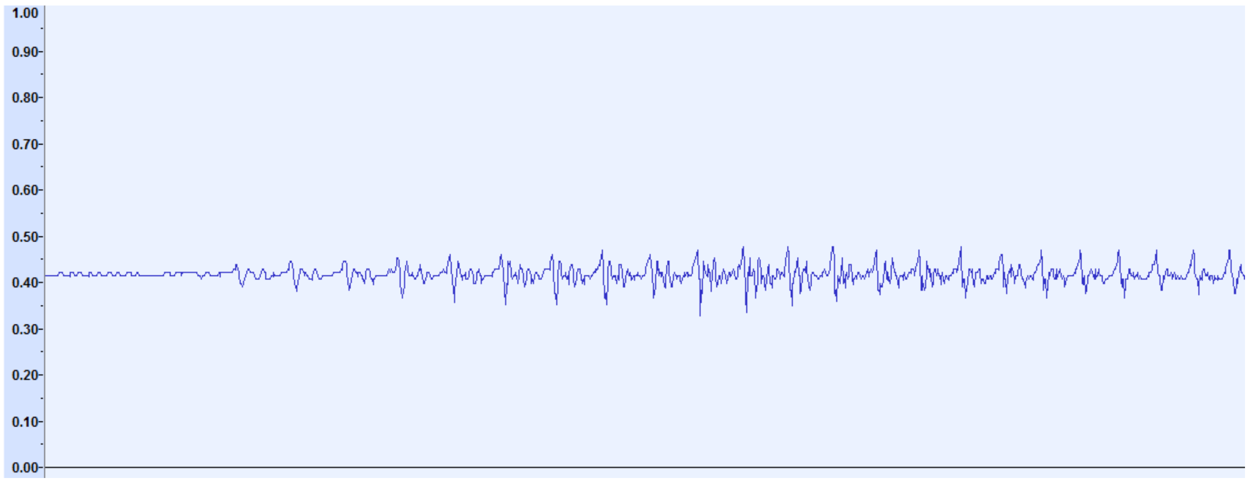


Figure 8 Output of Microphone when Talking Close to Microphone

3.5 Sensor

In order to get the correct results for autosave mechanism, we needed to figure out the peak acceleration needed. To verify this, we tested for the peak acceleration by dropping the accelerometer from waist height (about 80cm). From fig 9, the tested value was around 35m/s^2. In addition, we needed to check the orientation of the accelerometer. This was done by the built-in orientation function of the accelerometer. The verification process was to check that the orientation showed portrait orientation when the accelerometer was lying flat and landscape orientation when the accelerometer is standing up. A depiction of the orientation is shown in fig 10.

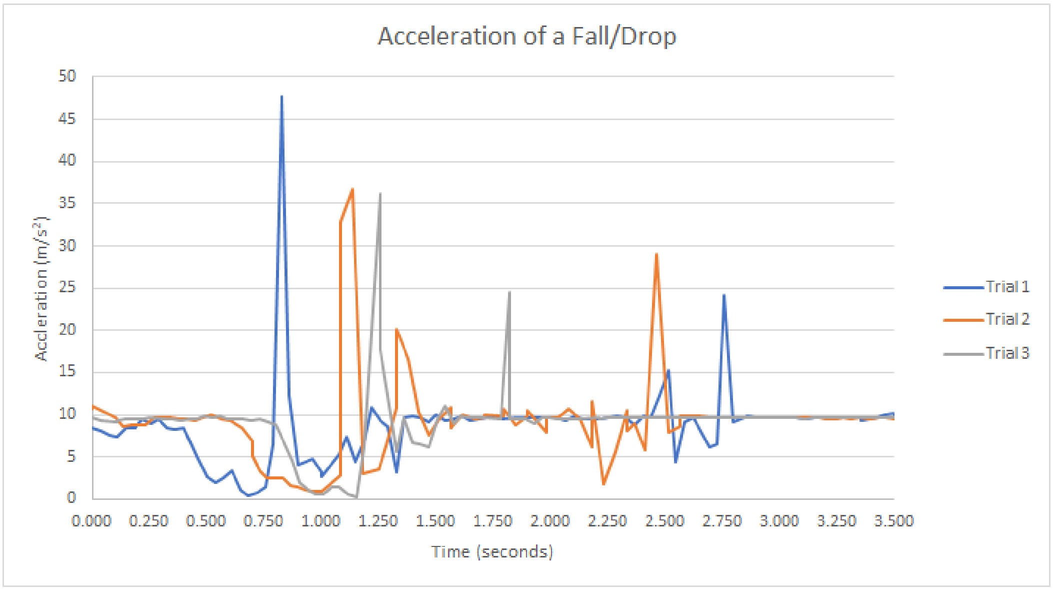


Figure 9 Output of Accelerometer from Freefall Impact

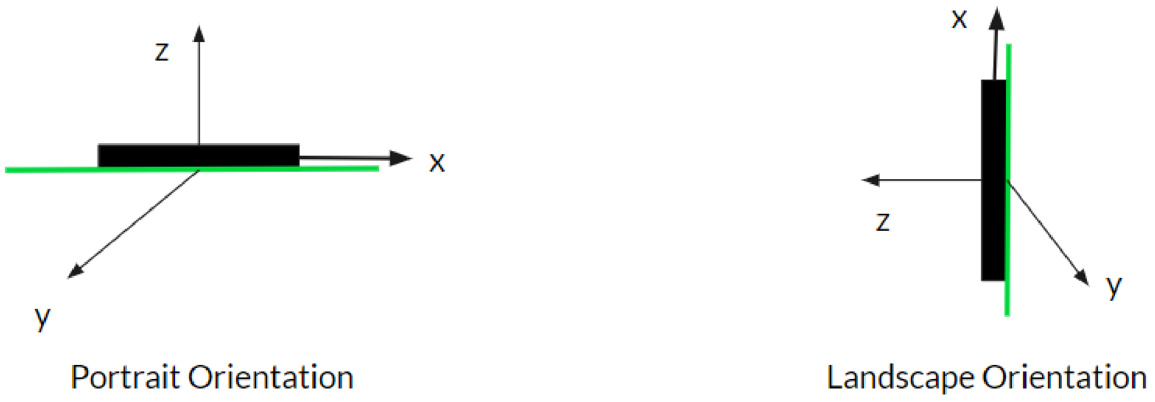


Figure 10 Accelerometer Orientations

4. Costs

While the costs of our parts do not seem realistically plausible as a commercially viable device that could be sold on the market, a lot of the part choices are chosen to make prototyping easier and smoother. For example, the Adafruit and Sparkfun parts can be chosen as their respective parts instead of the breakouts. This would significantly lower the part costs by a good margin. Thus, a final product of the device could probably be generously lowered down to around 30 USD for mass production.

4.1 Parts

Table 2 Parts Costs

Part (Manufacturer Part Number)	Manufacturer	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)
Lithium-Ion Charger (BQ24232RGTR)	Texas Instruments	2.59	1.19	2.59
Coin Cell Battery (RJD3048)	Illinois Capacitor	13.37	7.14	13.37
Battery Holder (36-301-ND)	Keystone Electronics	1.92	0.855	1.92
Linear Voltage Regulator (TS14002-C033DFNR)	Semtech	0.95	0.395	0.95
Microcontroller (ATMEGA328P-PU)	Microchip Technology	2.14	1.78	2.14
SRAM (23LC1024T-I/SN)	Microchip Technology	2.16	2.08	2.16
microSD	SanDisk	5.75	N/A	5.75
microSD Breakout Board (254)	Adafruit	7.50	6.00	7.50
OLED (LCD-13003)	SparkFun	15.95	N/A	15.95
2x Momentary-On Switch (K12SCS1.55NOLFTX)	C&K	2.33	1.86	4.66
SPST Switch (GPTS203211B)	CW Industries	1.53	1.09	1.53
Rotary Encoder (PEC12R-4225F-S0024)	Bourns Inc.	1.28	0.72	1.28
MEMS Microphone (2716)	Adafruit	4.95	3.96	4.95
Speaker (CMS-160925-078SP-67)	CUI	2.24	1.17	2.24
Accelerometer (2019)	Adafruit	7.95	6.36	7.95
microUSB-B Breakout (1833)	Adafruit	1.50	1.20	1.50
D to A Converter (MCP47CVB01-E/UN)	Microchip Technology	0.75	0.566	0.75
Total		**74.86**	**36.37**	**77.19**

4.2 Labor

Since a typical Illinois graduate averages \$71,856 a year, assuming a 40 hour work week, that comes out to around \$35/hour. Assuming we put in 10 hours a week for 16 weeks, our fixed development costs ends up being:

$$2 \times 35 \frac{dollars}{hour} \times 10 \frac{hours}{week} \times 16\, weeks \times 2.5 = \$25{,}000 \tag{2}$$

5. Conclusion

Our device provides on demand audio recording that doesn't require any prior 'start' command. The audio files that are saved on the device can be cropped to remove sections the user doesn't want. In addition, it has autosaving functionalities to defend the user from threats both physically and verbally.

While our device idealistically does all the objectives summarized above, our device we conjured up during this semester cannot handle all the requirements. The major lessons we learned from this project includes the relationship between one master and multiple slaves in I2C and SPI communications, utilizing two rounds of PCB with the first round to print only for testing purposes and not full integration, and exercising caution when connecting parts together that require specific voltages (SPI/I2C communications 5V vs. 3.3V) and setups (speaker & microphone needing amplifiers)

5.1 Accomplishments

For the audio portion of design, our design successfully records audio into the SRAM buffer and rewrites the data inside the SRAM buffer every 15 seconds. The audio data inside the SRAM buffer is also capable of being send to the microSD card for permanent saving. The file saved in the microSD and can be read and written back to the microcontroller to be cropped into a new file.

For the automatic saving functionality, the accelerometer autosave detection works and can be implemented.

5.2 Uncertainties

During our senior design, there were several parts that we didn't get working. The whole power module was not verified and some parts even untested. As a result, we could not test the size constraints and battery lifetime of the device. While our design should pose no issue for the uptime from our power calculations, the same thing can't be said about the size. Currently, the device still remains slightly oversized for a device that is to be worn like a watch. Idealistically, it would be even better to shrink the size of the design to make it more suitable for wearing. There also might be an issue of the device being too heavy as well. However, most of the weight comes from the battery. This is something we didn't get to fully test out and decreasing the size of the battery could end up lowering the uptime of the device significantly.

In addition, we would need to reconsider the designs for audio modules. The current design's microphone cannot record a reasonable distance away. We would like to include some adjustments to the microphone unit. This will be discussed in the future work section of the conclusion. As for the speaker, we never tested the part to be fully functional. This is mainly because we forgot to include an amplifier for the speaker to allow it function properly. Since playback isn't a major part of the design considerations, as long as a general speaker that is small enough and can playback recognizable voice audio would be sufficient for the device.

5.3 Ethical considerations

While the device cannot directly violate any ethical concerns, it is possible for a user to use the device for unethical purposes. The device can potentially violate privacy through misuse such as eavesdropping. According to the ACM code of ethics, our device has the possibility to violate sections 1.2, 1.6 and 1.7 [4]. All three sections deal with issues directly and indirectly due to breaching privacy. In order to combat against such misuse, we implement multiple methods of catching the attention of others when the device is in use.

The first method of grabbing attention is with the use of flashing lights. By lightning up the OLED and the built-in LED from the microSD breakout whenever the device saves a file. The second method if we correctly implement speakers, is to add an audible sound such as a high pitch beep or voice to indicate audio saving. We hope the bright light and sound will catch the attention of others nearby the user.

5.4 Future work

With our current design, the first problem that is in dire need to fix is the low processing power. We propose a solution to this issue by separating the autosave mechanisms and the audio recordings onto two separate microcontrollers. This way, both the audio collection and the autosave calculation can be done at faster rates.

Currently, the MEMS microphone used does not output a clear audible sound. Thus, we propose adding in an amplifier for the microphone to increase the effective distance the microphone can pick up as audible sound.

More advanced algorithms that require extra processing power can be implemented to increase the accuracy and precision of the autosave mechanism. Specifically for the accelerometer, it is possible to increase precision by using the three vectors to calculate the angle of the resulting acceleration from each axis. Then by using the angle to determine when device experiences a fall. As for the microphone's autosave, a second constraint should be included to ensure that sudden changes from silent to normal talking volume doesn't trigger autosave. This second constraint can be something as simple checking if it the volume of the sudden increase is loud enough to be considered a shout or loud voice.

References

[1] *G.711 : Pulse code modulation (PCM) of voice frequencies*. [Online]. Available: https://www.itu.int/rec/T-REC-G.711-198811-I/en. [Accessed: 02-May-2019].

[2] "Difference Between SRAM and DRAM (with Comparison Chart)," *Tech Differences*, 11-Dec-2017. [Online]. Available: https://techdifferences.com/difference-between-sram-and-dram.html. [Accessed: 02-May-2019].

[3] Huynh, Nguyen, L. B., Tran, B. Q., and Uyen D., "Optimization of an Accelerometer and Gyroscope-Based Fall Detection Algorithm," Advances in Decision Sciences, Apr. 09, 2015. [Online]. Available: https://www.hindawi.com/journals/js/2015/452078/. [Accessed: Mar. 01, 2018]

[4] "ACM Code of Ethics and Professional Conduct," *ACM* [Online] Available: https://www.acm.org/code-of-ethics [Accessed: Feb 7, 2019]

Appendix A Requirement and Verification Table

Table 3 Power Subsystem

Requirement	Verification	Verification status (Y or N)
Lithium-Ion Charger 1. Charges the battery to between 4.15-4.20V when supplied with a 5V source from micro USB 2. While charging, the IC does not exceed 50 degrees Celsius	1. a. Discharge the battery to 3V b. Charge the battery while connected to the microUSB c. After the charger indicates that charging is finished, ensure that the battery's voltage is between 4.15-4.20V 2. While charging, measure the temperature of the IC with an IR thermometer to ensure that the IC does not exceed 50 degrees Celsius	**No** We tested the chip several times. The first time the chip was not solder on properly; however, we did not realize this until we tested it the second time. On our second try, it burned up due to a capacitor misconnected resulting in a short circuit from charging pin to ground. By this time, it was already late in semester and we decided to give up on this subsystem to fix the other modules.
Lithium-Ion Battery 1. Supplies at between 4.15-4.20V when fully charged 2. Can power the device for at least 8-9 hours	1. a. Fully charge the battery while connected to the charger b. Measure the voltage of the battery to make sure the voltage is between 4.15-4.20V 2. a. Fully charge the battery b. Connect the battery to a circuit that approximates the current load and impedance of the actual device and leave it running c. Inspect the voltage of the battery after a couple hours to extrapolate how long the device would last	**No** Due to our charger not working, we could not properly test charging the battery. In addition, even if we managed to charge the battery without the chosen charger, we didn't manage to implement the control unit on the PCB to test how long the battery can power the device.

Voltage Regulator 1. Able to step down voltages from between 4.2V-3.0V to 3.3V	1. a. Connect the voltage regulator to a variable power source b. Measure the output of the voltage regulator using an oscilloscope while slowly varying the input voltage from 4.2V to 3.0V	**No** Similar to the battery, we didn't manage to implement the control unit on a PCB. Thus, the power source was straight from an Arduino. As a result, the voltage regulator was not used.

Table 4 Control Unit

Requirement	Verification	Verification status (Y or N)
Microcontroller 1. Able to route data at a rate of at least 64kbps	1. a. Connect the microcontroller to the microphone and the microSD b. Record audio for 15 seconds and store it onto the SD card c. Listen to the audio to verify audio integrity	**Yes**
MicroSD 1. Able to communicate with a microcontroller via SPI	1. a. Insert the SD card into the microSD breakout board and connect it to an arduino b. Try to read and write to the SD card and confirm that it works.	**Yes**
SRAM 1. Able to read and write to the SRAM from the microcontroller	1. a. Connect the SRAM to the microcontroller b. For each address up to 100, write its respective address into that memory space c. Read all of the data from addresses 0-100 to verify that the SRAM has stored the data correctly	**Yes**

Table 5 User Interface

Requirement	Verification	Verification status (Y or N)
OLED Display 2. Able to display text and numbers on the screen	1. a. Connect the display to an arduino via breadboard b. Verify hardware connection by using the test program given by the library c. Verify that we can write our own text to the OLED	**Yes**
Rotary Encoder Knob 3. The encoder is able to send two different signals to the microcontroller when the knob is being turned clockwise or counterclockwise	1. a. Connect the rotary encoder to a breadboard and an Arduino such that the Arduino is connected to the two channels of the encoder b. Rotate the encoder clockwise and counterclockwise and observe the signals received on the Arduino to determine if the encoder sends usable signals	**Yes**

Table 6 Audio Interface

Requirement	Verification	Verification status (Y or N)
Microphone 4. Able to record audio from a human speaker that is recognizable when listened to	1. a. Connect the microphone to the device and start recording to an SD card b. Start speaking into the microphone c. Playback the audio on a computer to verify that the microphone works and can record the frequencies needed to recognize voice	**Yes**

Speaker 5. Able to play sounds from 100Hz up to 4kHz	1. a. Connect the speaker to the device b. Have the device play a sine sweep starting from 100Hz up to 4kHz c. Record the sound played with an external microphone d. Perform a spectrogram on the recorded sound to confirm the speaker can play the lower and higher frequencies	**No** Due to our flawed circuit design, we didn't manage to implement the speaker. In addition to the DAC, we needed an amplifier to allow the speaker to produce sound. By the time we realize our amplifier was faulty, we were already out of time to test a new amplifier.

Table 7 Sensor Module

Requirement	Verification	Verification status (Y or N)
Accelerometer 6. Able to detect changes in position in three dimensions 7. Able to detect at least 5G in force	1. a. Connect the accelerometer to an external device such as an arduino b. Have the device start collecting data the accelerometer is giving c. Confirm from the data that the accelerometer can detect changes in position in three dimensions 2. a. Connect the accelerometer to an arduino b. Start recording data with the accelerometer c. Confirm that the data provided has a wide enough resolution to use in our device	**Yes**