Indoor noise monitor

ECE 445 Final Report - Spring 2019

William Xu, Ziyao Li and Quan Liu
Team 50
TA:Zhen Qin

4/27/19

Abstract

Our device was created with the intention of controlling noise levels in urban apartments. Studies have shown that noise levels in our living environments greatly affects our health and productivity. Our device aims to solve that problem with a noise sensor that can give dynamic feedback to residents based on their noise level. It also expedites the process of noise complaints by directly sending notifications to landlords or building administrators. Our device is also convenient and portable because it can operate on battery power, charge safely and is small in size. Overall our device was successful because we could measure dB SPL with an accuracy of +-5 dB SPL from 200 Hz to 5000 Hz, and our notification and warning systems worked correctly.

Contents

1. Introduction

1.1 Objective

It is significant to maintain acceptable noise levels in crowded areas such as apartments. Some people prefer quiet environments and are easily disturbed by their neighbors. For instance, when a neighbor is having a party or playing an instrument, it can be very distracting for their neighbors especially in apartments with thin walls. Most apartments don't control the noise well. So we need a product for landlords to maintain a quiet environment so that tenants will have a better living experience. Specifically, we need a device to warn the people indoor when they are being too noisy. If they keep being noisy after several warnings are given, the device will be able to quickly resolve the issue by automatically notifying the device administrator or landlord.

We have created a device that will give feedback to apartment residents to know how loud they are and give them a chance to quiet down. Our device is light, portable and safe because it can run on battery power and has measures to prevent overcharging. The device will also notify the device administrator through WiFi if they ignore the warnings. We have made this device to be very customizable and adjustable by offering the landlord many options in how they control their quiet environment. Our device allows landlords to customize threshold remotely. For example the landlord can change the average decibel value threshold as well as the amount of time that it is acceptable. The administrator can also change the amount of warnings that need to be triggered for the device to start sending phone notifications to indicate that the warnings have been ignored.

1.2 Background

Our inspiration for our project was our realization of how big a problem noise levels could be in urban environments, and how that noise could affect a person's health. Numerous studies have been done which show that urban apartment noise affects both the health and the productivity of its residents. We have found two different institutions in the Graham Research institute and Berkeley Health who have conducted studies on apartment noise and found the impact on health. The study from the Grantham Research Institute highlights the issues of noise, and mentions, "*analyze the health effects of residential noise annoyance using a high quality longitudinal survey of over 5000 adults in the Netherlands between 2007 and 2013."* and finds that "*health effects of residential noise annoyance, with neighbour noise relatively more damaging than street noise*" [1]. Another article from Berkeley points out the specific health effects that residential noise can have such as "*stress which, in turn, can increase your heart rate, cause your blood pressure to rise, slow down digestion*" [2]. Clearly this issue is very widespread especially in urban areas and needs to be addressed.

We have looked at other existing products on the market such as NoiseAware which can sense noise and send property owners alerts if the noise level exceeds a threshold [3]. We designed our product to be unique and to have a number of advantages over NoiseAware. First our product is more portable because our device is capable of running on battery power which makes it easier to move around and adjust positions. Also our product will be cheaper because it will have a one time hardware cost, while the NoiseAware has both a hardware cost and a subscription cost. Our product has other advantages over NoiseAware such as that it can give warnings to the people indoor at first which gives them a chance to fix the noise problem, whereas the competing product only just sends to the device

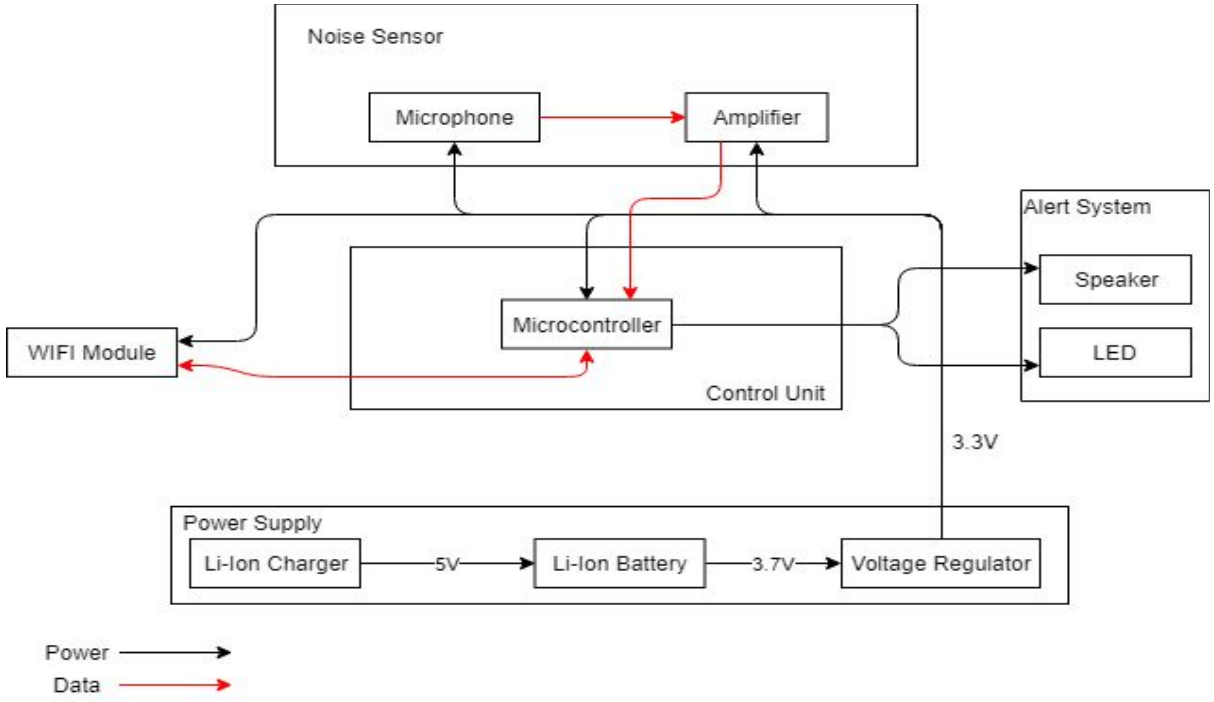
administrator without a warning. Our product is also better than a decibel meter because it can has a multitude of features that a decibel meter cannot provide. It can give notifications and warnings through a wifi connection, adjustable thresholds, and 24 hour monitoring, all features that decibel meter does not have. Thus, it is necessary for us to build this indoor noise monitor to solve residential noise problem effectively.

1.3 High-Level Requirements

- Administrators can change thresholds remotely and the device will output light and sound alarms when threshold is exceeded and will send notifications to the user after multiple warnings are ignored.
- The device should be able to accurately measure sound levels from 50 dB SPL to 85 dB SPL with an accuracy of ±5 dB SPL.
- The device should be able to operate on battery power for 24 hours on a full charge.

2 Design

The device has 5 different subsystems: power supply, noise sensor, control unit, alert system, and wifi module. Power supply powers the rest of the components with 3.3V ±0.05V, and it can be charged by power adaptor. The ATMEGA328 control unit accepts the sound signals and adjustable parameters like threshold from the wifi module and controls the alert system. The alert system is driven by the control unit to give warnings. The wifi module communicates with the control unit to set thresholds and s end phone notification.



Block Diagram

2.1 Power Supply

There are two main requirements for the power circuit design:

1. It should go through a complete fast charging procedure, and will automatically stop charging and maintain constant voltage when charging voltage reaches 4.20V.
2. It should block battery power supply from supplying device when power adaptor is plugged in.

2.1.1 Battery Management and Charging

At the beginning we plan to accomplish the charging procedure by combining current source and voltage source simulated by voltage regulator and MOSFET circuit to deliver the battery fast charging current. However, after testing and measurement, we found that this implementation may not be able to deliver stable charging current to the battery and cannot guarantee charging termination when the battery reaches desired voltage level. Most importantly, this circuit has very high power consumption and generates extremely high amount of heat, which may have serious potential safety risks. Thus, we changed our design and use an MCP73833 IC chip to detect and automatically control the charging process of the battery, and a 100nF thermistor to serves as the safety protection in case of unexpected overcurrent and overheating.

MCP73833 IC Chip **Thermistor**

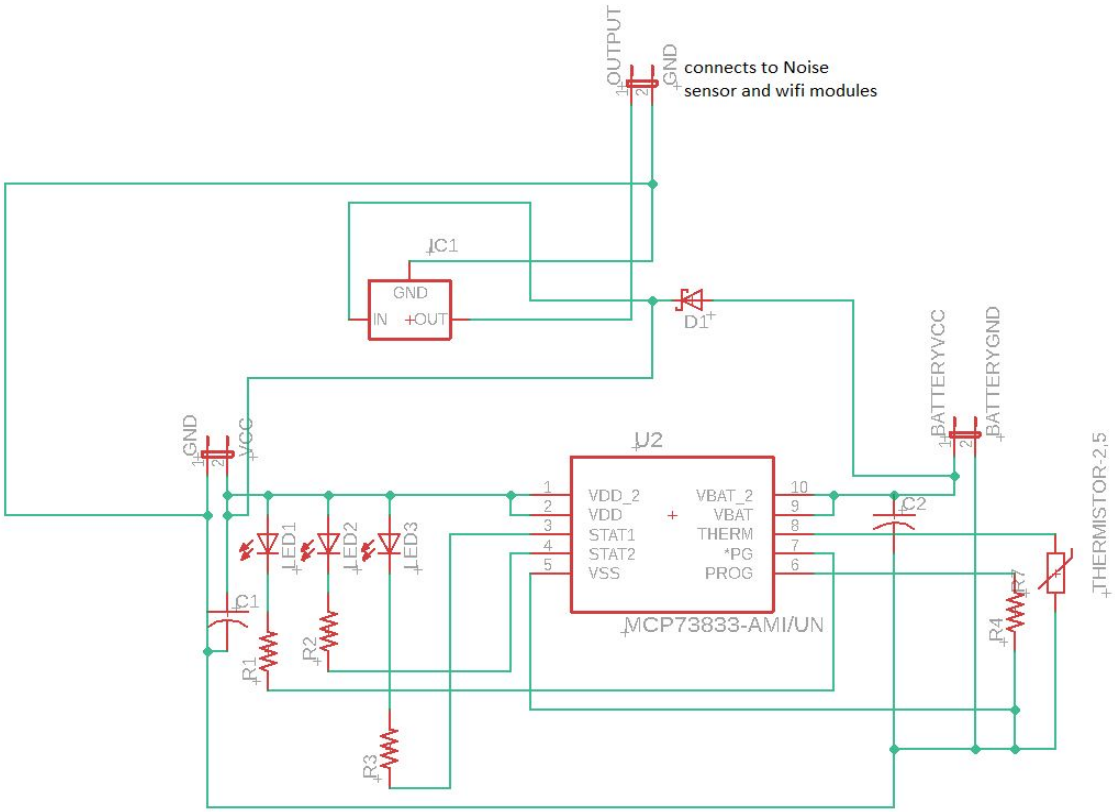
This IC chip can deliver a maximum constant current of 1A and is optimal for charging 2000mah battery or below. Another noticeable point is that not thermistor with any capacitance can fit into the chip's circuit since some capacitance may lead to unexpected operation state of the chip or cannot perform charging correctly, and the thermistor currently using is carefully chosen after tests.

2.1.2 Battery/DC Power Adaptor Circuit Separation and Voltage Control

In order to accomplish the goal of blocking battery power supply when the power adaptor is plugged in, we use a single schottky diode connecting before the voltage regulator in the battery line. This diode serves to create a negative voltage difference and blocks the forward current offered by the battery when the power adaptor is plugged in. Both the battery line as well as the power adaptor line goes to a 3.3V LDO voltage regulator. The voltage regulator will take either 5V from power adaptor or 3.7V from battery as input and output a constant 3.3V to the rest of the device.

3.3V 500mA LDO Voltage Regulator (TC1262) **100V 30A Schottky Diode**

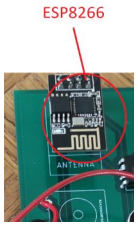
Although there are LDO voltage regulators which can offer smaller drop out voltage, their maximum allowable current is too small and may cause lack of power to the microcontroller chip. Hence we select this voltage regulator which can offer a drop out voltage of less than 0.3V under the maximum 500mA current.

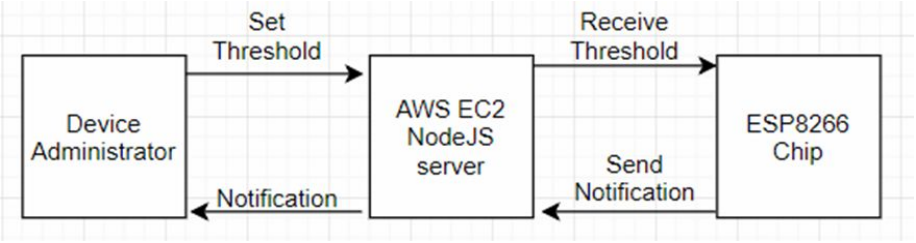


Schematic of the whole power supply circuit

2.2 WIFI MODULE

Our wifi module uses the ESP8266 chip to connect to a wireless network. We have created an AWS EC2 NodeJS server to store the threshold values that can be set remotely by the the device administrator using Postman, a HTTP request sender. ESP8266 will request the threshold values using GET HTTP request, and when the threshold values are exceeded multiple times, the wifi chip will send a request to the AWS server so that a text message notification should be sent to the device administrators phone.





Communication block chart

2.2.1 Connecting to WIFI

One of the most important parts for device to work is to make sure our ESP8266 chip can connect to WIFI. At first we tried to flash the ESP8266 with AT command firmware so that the Wifi connection and HTTP request can be conducted by simple AT command like “AT+CWMODE=1”, “AT+CWJAP="wifiname","password” and AT+CIPSTART=“TCP”,”ip address”,port1, port2. So the way to control ESP8266 is to send a AT command from ATMEGA328P to ESP8266 through serial communication.

However, during the HTTP request, the message response from ESP8266 contained lots of unreadable characters. As a result, this communication method didn’t work.

Design alternatives:

To solve this problem we decided to program the ESP8266 independently. The following code was written to try to connect to the network that is stated in the const ssid and uses the password values to login. We needed to verify that our chip was capable of connecting to wireless networks.

```
#include <ESP8266WiFi.h>
const char* ssid     = "xxx";
const char* password = "1233211234567";
int connected = 0;
void setup() {
  Serial.begin(115200);
  //delay(10);
  //Serial.println();
  //Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  connected = 1;
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
int value = 0;

void loop() {
  if (WiFi.status() != WL_CONNECTED) {
    connected = 0;
  }

  if (connected == 0) {
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED)
    delay(500);
    Serial.println("connecting");
    }
    connected = 1;
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
  }

}
```

ESP8266 code for WiFi access

After successfully connecting to a wireless hotspot that we setup with our phone, we send HTTP GET request to obtain the threshold stored in the server.

```
HTTPClient http;
http.begin("http://ec2-34-211-225-38.us-west-2.compute.amazonaws.com:3000");
int httpCode = http.GET();
if (httpCode > 0) { //Check the returning code
  String response = http.getString();   //Get the request response
  Serial.print(response);
}
http.end();
```

ESP8266 code for HTTP GET request

2.2.2 Sending text message notification

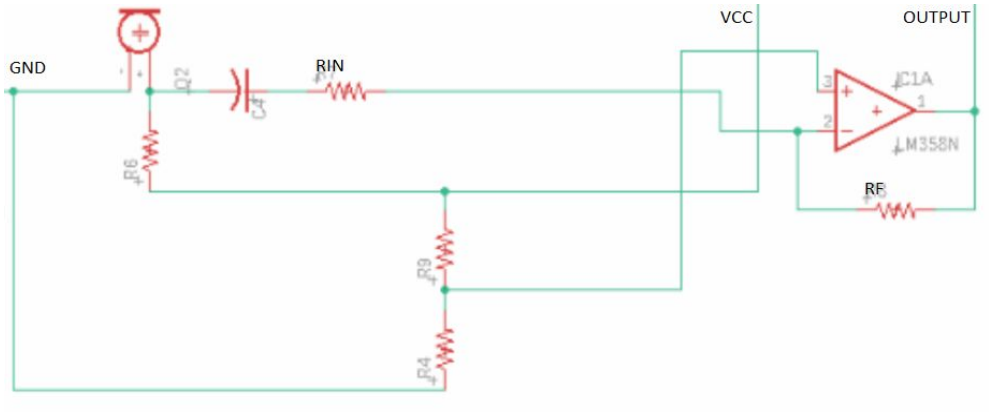
When the threshold is exceeded three times or some other value that the device administrator decides, we have to send a phone text message notification. To do this we programed the ESP8266 to listen for ATmega328P, once ATmega328 sent a signal to ESP8266, ESP8266 sent a HTTP request to the NodeJS server. Then NodeJS server used the NEXMO Communications API to send text message notifications to a phone.

2.3 Noise Sensor

The noise sensor accepts sound signal and outputs amplified analog signals to be processed by the microcontroller. The noise sensor contains a CZN-15E electret microphone and a LM358P amplifier. The purpose of using an amplifier is that the amplitude of microphone's output is not big enough for the analog to digital convertor in the microcontroller to distinguish. For instance, the amplitude of a sound wave of 60 SPL could have a ADC reading of 10 and a sound wave of 80 SPL could have a ADC reading ot 12. As a result, an amplifier is needed to amplify the amplitude so the amplitude can be distinguished by the ADC for further processing.

As shown in the figure below, the microphone in the left accepts sound signal as input and outputs voltages in the right of the capacitor. Then the signal is amplified by the LM358P amplifier by setting the resistor RIN = 1k ohms and RF = 47K ohms. Thus, the amplifier has a gain of 48 as calculated below:

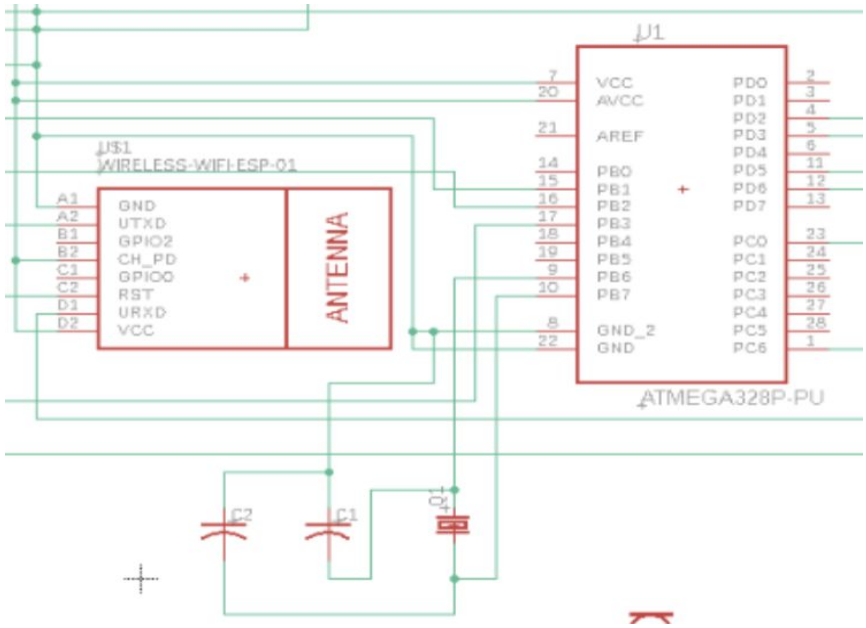
$$A = \frac{V\,out}{V\,in} = 1 + \frac{Rf}{Rin} = 1 + \frac{47k\ ohms}{1k\ ohms} = 48$$



Schematic of microphone and amplifier circuit

2.4 Microcontroller

Microcontroller accepts noise threshold from the WiFi chip, accepts sound signals from the noise sensor, and drives the alert system to send warnings. As shown in the figure below, there is a 16MHz crystal and 2 20pF capacitors as outside clock for the ATMEGA328P chip to operate. It also communicates with the ESP8266 WiFI chip through TX/RX serial communication. It accepts sound signals in its analog port A0.



Schematic of Microcontroller and ESP8266

2.4.1 Increasing Sampling Rate

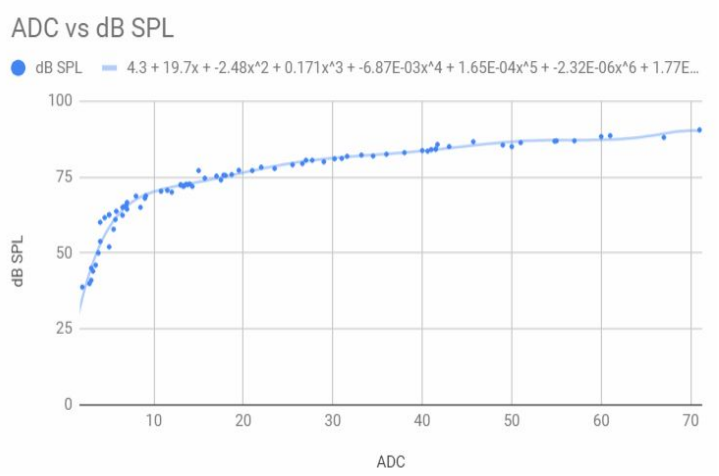
Originally our ATmega328P is only able to sample audio from the analogRead function at 9600Hz. This is not enough to measure the sound waves with higher frequency.

Design alternative: In order to increase the sampling rate to analyze the human hearable sound with frequency range 20 - 20,000Hz, we first disabled the analogRead function by manipulating some registers in ATmega328P, then we directly read the ADC value from analog port during interrupt routines[4]. As a result, the sampling rate was increased to 38.5kHz, which made it possible to analyze the sound with frequency up to 19.25kHz.

```
ISR(ADC_vect) {
  adc = ADCH;
}
```

2.4.2 Converting ADC values to dB SPL

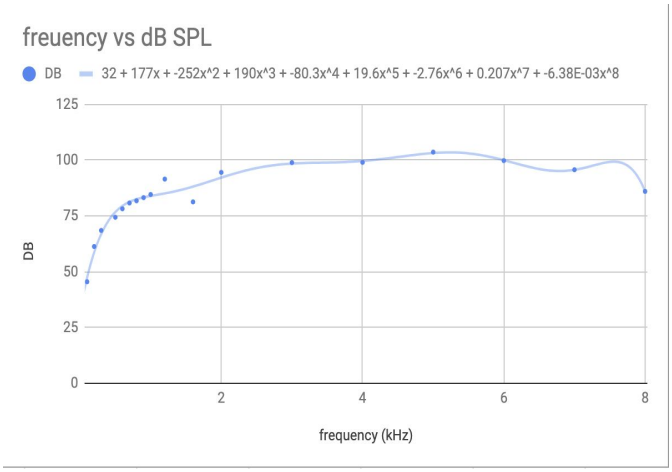
After we received the ADC readings and we converted them to dB SPL. First step was calibration with a real decibel meter. Under a 500 Hz sine wave, we recorded the ADC readings and the corresponding dB SPL values in the decibel meter. Then we used 8-order polynomial fitting to calculate the relationship between ADC and db SPL in the figure below.



Polynomial fitting of ADC vs dB SPL

In addition, since the decibel meter utilized A-weighting and outputed different dB SPL value as the frequency changed, and sensitivity of microphone also changed as different frequency, we need to compensate the calculated dB SPL in different frequencies. So we recorded different frequency and its corresponding dB SPL of decibel meter. Then we did polynomial fitting on it and found that the shape is same for different volumes, meaning that it was a reliable fitting of frequency vs dB SPL. In order to make use of the polynomial fitting below to compensate for different frequencies, first we used the fitting above to convert ADC to dB SPL, then we measure the current frequency and calculate the ratio

of the dB SPL to the dB SPL at 500 Hz, the baseline frequency. After that we multiply the db SPL by the ratio to get the final dB SPL values.



Polynomial fitting of frequency vs dB SPL

2.4.3 Finding the Threshold

Our threshold is a decibel value average that extends over a period of time. An example of this threshold would be 80 dB SPL over 10 seconds which means that the user must not exceed an average of 80 dB SPL over the 10 seconds. We use the Running Average library to calculate this. In our case we created an array of size 100, and since we want 10 seconds, we feed the array with a new value every 1/10 of a second. When the array is full we pop out the oldest value from the array while inserting the new value and we calculate the average to check if it over the dB SPL threshold. This way we can find the average of every possible interval.

2.4.4 Sending warnings & notifications

Everytime the threshold is exceeded, ATmega328P drives LED light and a buzzer. We decided that after a warning is sent, the device would have a cooldown period to give the users time to quiet down. After the cooldown period, it restarted measurements.

Once several warnings have been sent, ATmega328P sends a signal to ESP8266 to send a request for text message notification.

2.5 Alert System

The alert system contains LEDs and a buzzer. At first we used a passive buzzer and drived it with PWM waves, but the voltage is not high enough when powered by the power module.

Design alternative: We replaced the passive buzzer with an active buzzer which output sounds given a steady voltage.

3. Design Verification

3.1 Power Supply

Verification of power supply circuit involves measuring voltage, current, and checking operation state.

3.1.1 Battery Management and Charging

For fast charging a 3.7V 2000mah battery with a constant current of 1A, the estimated charging time should be around:

$$2000mA / 1000mA = 2hrs$$

and the terminated constant voltage level should be at 4.2V in theory. However, after testing we found that the actual charging time is about 4 hours, which is significantly larger than the estimated time. Reasons that may explain this: First, the actual charging current may not be able to always maintain at 1A of full current due to the limitation of chip itself, or the built-in PCM of the li-ion battery which may restrict the current. Second, the charging of li-ion battery may goes through a constant voltage stage which we cannot identify that by observing the display LEDs, and the current during this stage is comparatively small. The terminate voltage to stay constant is measured to be 4.175V.

Requirement	Verification	Verification status (Y or N)
1. Goes through a fast charging procedure and charge the battery to the desired voltage within specific time.	1. i. Connect the adaptor to the charging circuit and connect an empty battery to the output of the IC chip. ii. Connect a voltmeter in parallel with the battery to measure the charging voltage across the battery until the display LED of the IC chip lights up green. Record the time.	1. Y
2. Automatically terminate charging as soon as the charging voltage reaches 4.2V and keep the voltage constant.	2. i. Connect the adaptor to the charging circuit and connect an almost full battery to the output of the IC chip. ii. Connect a voltmeter in parallel with the battery to measure the charging voltage across it until the display LED lights up green. Observe the voltage across it to see whether it stays at the desired constant value.	2. Y

3.1.2 Battery/DC Power Adaptor Circuit Separation and Voltage Control

The power circuit should instantaneously block the power supply from battery once the power adaptor is plugged in. In other words, there should be no current flowing through the battery line once the power adaptor is plugged in. After measurement, there is no forward current flowing through the battery line with a very tiny reverse current of about -1.3uA.

Requirement	Verification	Verification status (Y or N)
1. Block the battery power supply to the rest of the device and no current flowing through the battery line.	1. i. First unplug the power adaptor and only connect the battery to the power circuit. ii. Connect an amp meter in series right after the battery to measure the current flowing through the battery line. Record the current. iii. Next plug in the power adaptor and connect it to the power circuit. Keep the battery connected. iv. Use the amp meter to measure the current flowing through the battery line after the power adaptor is plugged in. Record the current.	1. Y

3.2 Wifi module

The verification of WiFi module involves the individual testing of ESP8266 and integration testing with the AWS EC2 node js server.

3.2.1 WIFI IC

Requirement	Verification	Verification status (Y or N)
1. The wifi IC must be able to receive threshold data from the AWS nodeJS server while connected to the wireless network	1. A. Program the ESP8266 to connect to wifi network. B. Send a HTTP get request to the AWS server. Check whether we receive the request from the ESP8266 in the server C. Check that the correct threshold is received by ESP8266 by printing it to terminal	1. Y

2. The wifi IC's UART_RXD and UART_TXD should function properly.	2. A. Connect ESP8266 to ARDUINO UNO board with its URXD and UTXD port. B. Send AT commands from the ARDUINO UNO board to the wifi IC ESP8266 through UART and see if the ESP8266 sends a response back [5]	2. Y
3. The WIFI IC needs to send a HTTP request to the AWS nodeJS server so that the server responds to send a text message to the user's phone	3. A.Connect ESP8266 to a wireless network B.Send an HTTP set request to the AWS server and see if the server sends a response that it has been received. C. See if the server responds by sending the text message notification to the phone	3. Y

Overall, we were able to verify that the wifi chip could receive the threshold. We used Postman, a HTTP request sender, to add a threshold value to retrieve, then we were able to see through the Arduino console that our device retrieved the same value that we set through POSTMAN. We also saw that our wifi chip was able to send request to our AWS server, and the server successfully sent a text message to our phone through NEXMO communication API.

3.3 Noise Sensor

This module contains microphone and amplifier. The amplifier will amplify the output of microphone and feed it into the microcontroller which contains an ADC. The digital signals will be processed by the control unit to calculate the noise levels.

3.3.1 Microphone

The microphone is a electret microphone(CZN-15E). It has -38 dBV sensitivity (1 kHz, 94 dB SPL) and ± 1 dB sensitivity tolerance.

Requirements	Verification	Verification status (Y or N)
1. The microphone should have -38 dBV sensitivity (1 kHz and 94 dB SPL) and ± 1 dB sensitivity tolerance	1. A. Use signal generator to output 1 kHz sine wave in 94 dB SPL level B. Use oscilloscope to measure the amplitude of the analog output of microphone and calculate difference.	1. Y

After using a signal generator to play a 1 kHz sine wave at 94 dB SPL, we were able to see using an oscilloscope that the analog output of microphone has 0.012 volts (RMS) which means -38.416 dBV sensitivity.

3.3.2 Amplifier

For the amplifier after extensive testing, we decided to go with a gain of 48 using a 47k ohm resistor for Rf and a 1k ohm resistor for Rin. We tested multiple gains for the circuit, but it seemed we needed at least 48 gain to get a reasonable output to process with our polynomial fitting for calculating dB SPL. The output of the amplifier will go to the ADC of microcontroller.

Requirements	Verification	Verification status (Y or N)
1. The amplifier should produce a gain of 48 to produce the desired strength output	1.A.Give the amplifier a sine wave with 0.01 V amplitude B.Use oscilloscope to test if the amplitude increase 0.48 V	1. Y

In our testing, when inputting a sine wave with a 0.01 V amplitude, we were able to observe that the amplitude output on the oscilloscope had an amplitude of 0.49V which is very close to the 48 gain that we were trying to achieve.

3.4 Alert System

Alert system contains a speaker and LED lights. The microcontroller will output analog wave to drive the speaker and LEDs when the sound thresholds are exceeded. The purpose is to make people indoor aware that they are being too loud.

3.4.1 Speaker

The speaker will create some kind of sound alarm and is driven by the microcontroller. We found the speaker to be much louder than we needed (3.3V), so way above 80 dB, but we were able to control the voltage given to the speaker to make it not so loud.

Requirement	Verification	Verification status (Y or N)
1. The speaker should be able to output a sound of 80 ± 5 dB SPL	1. A.Deliver 2.2 V to the speaker from signal generator. B.Measure the dB SPL with a sound meter to verify that it is 80 dB SPL.	1. Y

Originally the output of the buzzer produced a sound with a decibel value of 103 dB SPL, but by lowering the input voltage to 2.2V we were able to lower the sound level to 80 dB SPL.

3.4.2 LED lights

The LED lights, driven by the microcontroller, will light up to a let people indoor know that they are creating too much noise.

Requirements	Verification	Verification status (Y or N)
1. The LEDs should light up given 3.3V	1. A.Connect the LEDs with the output of a signal generator B.Hard code 3.3V analog output from signal generator to see if the LEDs light up	1. Y

3.5 Control Unit

The control unit will handle all the calculations of when to set certain thresholds of sound levels, when to give warnings, and when to notify the device administrator. Verification is shown in the following.

3.5.1 Microcontroller

The microcontroller we will be using is the ATmega328. It calculates the dB SPL with the polynomial fitting formula we give it, and then checks if the noise threshold is exceeded. It also drives the LEDs and speaker with analog wave. In addition, it will use UART to connect and communicate with the wifi module ESP8266 to accept threshold parameters and send notifications.

Requirements	Verification	Verification status (Y or N)
1. The microcontroller should be able to communicate with other device using UART	1. A. Connect ARDUINO UNO board to ATMEGA328 with UART. B. Send commands using UART from the microcontroller(ATMEGA328) to the ARDUINO UNO board and see if the the microcontroller sends a response back [5]	1. Y
2. The microcontroller should be able to output analog output through its PWM port	2. A.Hard code to send analog output through the PWM port B.Use oscilloscope to measure the output signal to check if it match the set parameters	2. Y
3. The microcontroller should be able to sample audio at 38.5 kHz	3. A.Set a timer in arduino to count 1 millisecond. B.Create a counter and add 1 to the counter in the interrupt routine because	3. Y

	that is when the incoming audio is updated to the ADC. C. Check that the counter is around 38500, which would mean 38.5 kHz because a millisecond is 1/1000 of a second	

In our testing we were able to successfully communicate with the ESP8266 using AT Commands. We used the function AnalogWrite(port, duty cycle) and connected the analog port to the oscilloscope and we were able to observe a steady square wave with the specified duty cycle that had a peak value of 3.3V. With our counter check to see if we could sample at 38.5 kHz, we observed the counter going to 38300 which indicates that the ATmega was able to sample the audio at 38.3 kHz.

4. Costs & Schedule

4.1 Parts

Parts Costs

Part	Manufacturer	Actual Cost ($)
WIFI IC ESP8266	Espressif Systems	$6.95
ATMEGA328P	Atmel	$1.96
CZN-15E Microphone	Cylewet	$6.99
LM358P amplifier	Texas Instruments	$0.46
GFORTUN buzzer	Gfortun	$8.29
LED	N/A	$0.81
Battery	AddWires	$11.95
Li-ion charger	Amazon	$19.99
Voltage regulator TC1262	Microchip Technology	$0.83
Battery management IC chip MCP73833	Microchip Technology	$0.87
Schottky diode V30100CI	Vishay Semiconductor	$1.44
Thermistor NKI100NF103C1R1E	Amphenol Advanced Sensors	$5.93
Resistors, Wires, capacitors,etc	N/A	$10.00
PCB	PCBway	$15.00
Total		**$80.20**

4.2 Labor

For our labor costs, we are estimating that each of the 3 people will earn $30 per hour, for 7 hours a week. Our total labor costs will be:

$$3 \times \$30/hr \times 7hrs/wk \times 16weeks = \$10080$$

4.3 Schedule

Date	**William**	**Ziyao**	**Quan**
2/25	Design WIFI and MCU module in eagle including the schematics and PCB layout	Design noise sensor and Alert system module in eagle including the schematics and PCB layout	Design power supply module in eagle including the schematics and PCB layout
3/4	Test wifi signal and speed	Test voltage to dB SPL accuracy from the linear regression	Test power supply requirements of the PCB
3/11	Revise WIFI module schematic and PCB layout	Revise noise module schematic and PCB layout	Revise power module schematic and PCB layout
3/18	Order revised PCB and test	Order revised PCB and test	Order revised PCB and test
3/25	Assemble WIFI module	Assemble noise sensor module	Assemble power supply module
4/1	Assemble and combine all subsystems together	Assemble and combine all subsystems together	Assemble and combine all subsystems together
4/8	Prepare for mock demo	Prepare for mock demo	Prepare for mock demo
4/15	Mock Demo and debug project	Mock Demo and debug project	Mock Demo and debug project
4/22	Begin final report	Begin final report	Begin final report
4/29	Prepare for final presentation	Prepare for final presentation	Prepare for final presentation

5. Conclusion

5.1 Accomplishments

The Noise Sensing device was able to achieve all the big requirements that we set out to create. We were able to give dB SPL values with an accuracy of +- 5 from 50 to 85 dB. Our device also worked across multiple frequencies ranging from 200 Hz to 5000 Hz. Implementing our device to be able to be accurate

across multiple frequencies was definitely a challenge because it also required an accurate measurement of frequency by our device, but we were ultimately able to find an accurate polynomial fitting across the frequencies. Our wifi module also worked very well as we were able to retrieve thresholds and send phone notifications with high speed and little delay. Our power subsystem worked very well too because we were able to implement safe charging by disabling current to the battery when it was full, as well as stopping the battery from supplying current to the device when it was plugged in. It allowed our device to be portable and convenient.

5.2 Uncertainties

Some uncertainties we had were that it was pretty hard to measure dB very accurately across a large range of frequencies. For our project, we were able to get it accurate within the frequency range of 200Hz to 5000Hz, however outside of that, our decibel readings starting getting inaccurate. In frequency greater than 5000Hz, the accuracy increased from +-5 dB SPL to +-10 dB SPL. The reasons were that our microphone was not sensitive enough at really high and low frequencies as well as polynomial fitting not being the ideal way. We discuss how we could improve upon this in our future works section.

Also our decibel meter was only capable of measuring from 8-8000Hz accurately which would have created inaccurate measurements in frequencies outside of those ranges.

5.3 Ethical considerations

There are definitely ethical considerations for our device. Unfortunately there are ways that the device can be abused that would go against certain code of ethics such as IEEE or ACM. A landlord could violate IEEE code of ethics #9 and ACM code 1.2 which states that we should avoid hurting others and their property [6]. Landlords can possibly set threshold levels to extremely low and unfair values so that tenants cannot do any type of activities that involve any types of noise which would violate code #9 and 1.2 because extremely low thresholds could stress them out and prevent them from necessary activities which could hurt their health. To avoid this issue we set a limit to how low of a threshold users can set for this device. Another code that could potentially could be faced with our products would be ACM code 1.6 which states that we should respect privacy [7]. Tenant might think that these devices breach their privacy rights because they will always be listening, just like people nowadays are paranoid of computer webcams spying on them or how social networks are using their data. We have prevented this by making sure our device does not save any audio being fed into the MCU, rather it will just measure the dB SPL levels and the amount of time that it happens.

5.4 Future work

For our future work, we could look at making our device more accurate. Right now our device can measure dB SPL levels at an accuracy of +-5, but this can be improved upon. Further we could also try to increase the frequency range at which our device is accurate, because currently it can measure between 200 and 5000 Hz, but this can be further increased. We could use Fast Fourier transform to easily measure the different voltage rms across frequencies, and then utilized A weighting, which is utilized in the decibel meters, to have a more accurate reading across different frequencies.

References

[1] Graham Research Institute, 'Sick of Noise: the health effects of loud neighbors and urban din' 2015. [Online]. Available: http://www.lse.ac.uk/GranthamInstitute/wp-content/uploads/2015/10/Working-Paper-213-Weinhold.pdf

[2] Berkeley Wellness, 'Sounding off on Noise' 2017. [Online]. Available: http://www.berkeleywellness.com/healthy-community/environmental-health/article/sounding-noise

[3] Noiseaware, 'FAQ' 2019. [Online]. Available: https://noiseaware.io/resources/faqs

[4] Instructables, 'Arduino Audio Input' 2017. [Online]. Available: https://www.instructables.com/id/Arduino-Audio-Input/

[5] Circuit Digest 'Interfacing ESP8266 with PIC16F877A Microcontroller', 2017. [Online] Available: https://circuitdigest.com/microcontroller-projects/interfacing-pic-microcontroller-with-esp8266

[6] Ieee.org, "IEEE IEEE Code of Ethics", 2016. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

[7] Acm, 'ACM Code of Ethics and Professional Conduct' 2018. [Online] Available: https://www.acm.org/code-of-ethics