Sensory Awareness Device for Bars and Restaurants

Megan Heinhold, Evan Lindquist, Carl Wolff

meganjh3@illinois.edu, evanl3@illinois.edu, cwolff2@illinois.edu

Final Report for ECE 445, Senior Design, Spring 2022

TA: Amr Ghoname

May 5, 2022

Project No. 17

Abstract

Sensory disabilities are underrepresented and unrecognized in society. We have successfully created and implemented a device which allows those with sensory disabilities to understand the light, sound, and temperature levels of the environment they will be entering without having to be in the vicinity of the establishment. The device we have created is small, requires little to no maintenance, and can be installed easily. Currently, there is no similar product to this, and owing to the simplicity and efficiency of our device, it can be mass produced and easily integrated into any establishment wishing to make themselves more accessible.

Contents

1. Introduction

Many factors play a role when individuals and groups decide where to spend their time and money. Today, many of these factors relating to locations including bars and restaurants can be easily found and compared online with services such as Google and Yelp. These services might include information such as hours, location, ratings, prices, and even busiest hours. Factors that are left out of these online resources include what makes up the general ambiance of a location: sound levels, lighting, and room temperature. These factors are dynamic and constantly changing but play a particularly important role in how much a patron might enjoy themselves, especially if said patron is affected by a Sensory Processing Disorder or another condition which makes it easy to become overwhelmed by one's environment.

Much like the services mentioned above, our solution seeks to provide information about a location's current noise, light, and temperature levels to individuals before they arrive at a location. In addition, our solution will identify conditions that pose specific threats to one's health or safety with regards to flashing lights or dangerous sound levels. We accomplished this goal using a physical device that is installed within a restaurant or bar of interest equipped with three sensors. These sensors are continuously polled, and each minute the average values are computed and sent over Wi-Fi to a Google Sheet. These raw sensor values are then converted into buckets corresponding to understandable levels of light and sound, and actual temperature, respectively. These understandable levels and values are displayed on a web application that can be accessed by patrons.

The success of our project was defined by these high-level requirements:

1. The device must be able to be plugged into a standard 110 V wall outlet and fit within a 10 inch by 10 inch by 2 inch footprint. It should not require maintenance more than once a month.
2. The device must accurately characterize both ambient levels and safety alerts. Temperature should be recorded to within +/- 1 degree Fahrenheit. Safety alerts should be identified for sounds above 110 dB and flashing light frequencies between 10-12 Hz.
3. The transmission of data to the user must be "real-time," so collection, transmission, and interpretation of data cannot take more than 10 minutes. Safety alerts regarding flashing lights and dangerous sound levels should be communicated to the web app in no more than 2 minutes.

1.1 Block Diagram and Subsystems

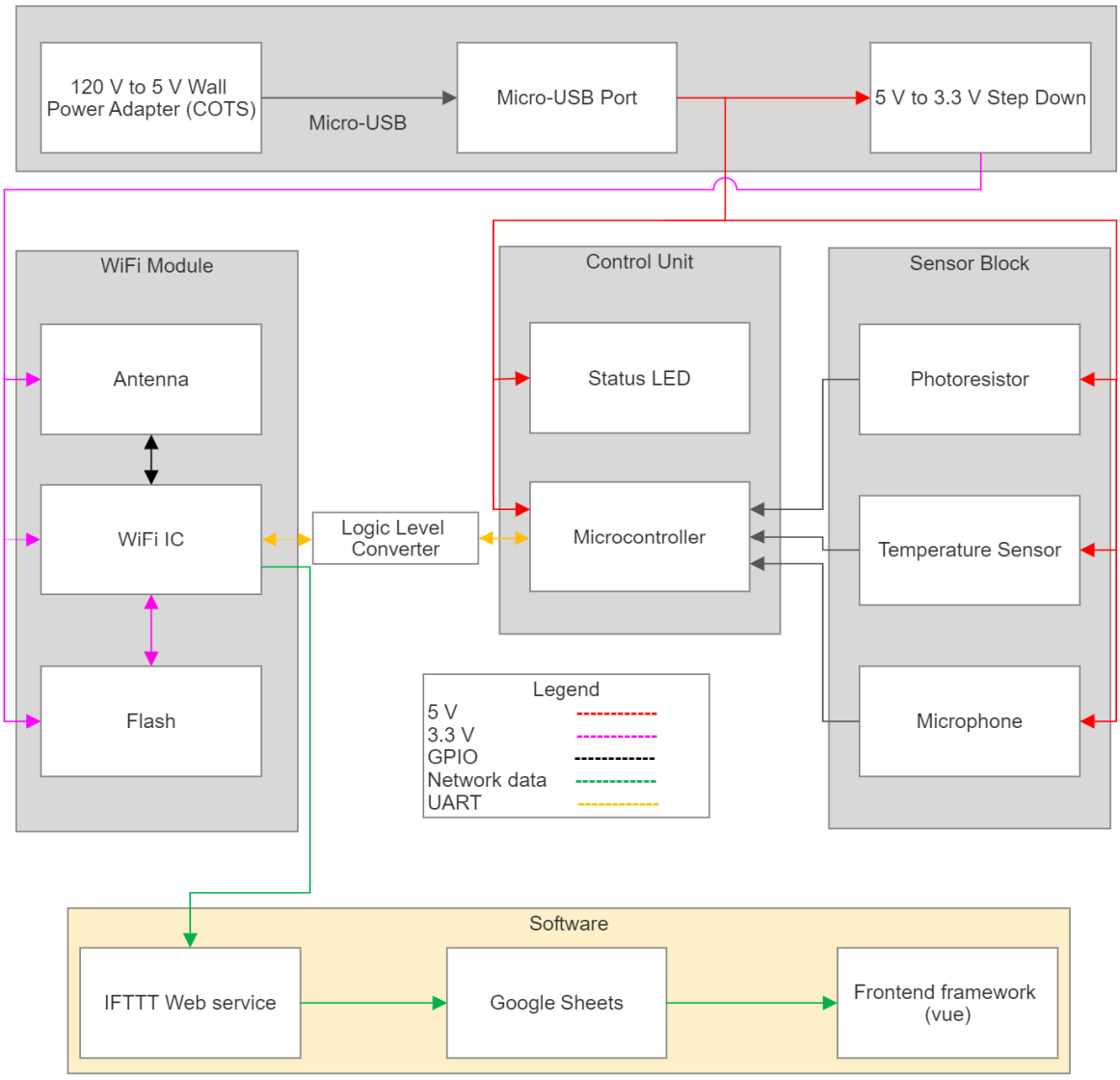

Figure 1. Final block diagram.

Figure 1 shows the final version of our block diagram, composed of five subsystems: the power supply, Wi-Fi module, control unit, sensor block, and high-level software. The entire system is powered via the power supply, which supplies steady 5 V and 3.3 V levels. The status LED shows the device owner that it is powered properly. Over the span of a minute, samples are taken with all three sensors. The microcontroller averages these samples over that minute in time and sends a packet containing three corresponding variables to the Wi-Fi integrated circuit. The Wi-Fi module submits an HTTP POST request to a web service called IFTTT, which then stores the packet as a new line with a timestamp in a Google Sheet. Our front-end web

application pulls this most recent data and refines it by identifying safety alerts over the span of the last ten minutes and sorting raw light and sound levels into our calibrated buckets. This interpreted information, along with the room temperature, is displayed to the end user.

2. Design

All subsystem schematics can be found in Appendix A.

2.1 Power Supply

A key criterion driving many of our design decisions was simplicity for the end users, the business owner, and patrons. This came into play in our first design decision, which was how to power our device. Since most of our components use a steady 5 V power, USB to micro-USB was ideal. This allows the owner of the device to use existing power bricks, such as the ones used to charge phones, or even the port on a laptop to power the device. The only other voltage level required for any of our planned components was 3.3 V, and due to the small difference with regards to the original 5 V we used a low dropout regulator with the recommended capacitors [1].

2.2 Wi-Fi Module

The Wi-Fi module's main purpose is to receive data from the control unit and upload that data to a JSON file over a Wi-Fi network. The Wi-Fi chip we used was the ESP8266 because of its availability, favorable reviews, and number of guides available on how to use it properly in projects. To program this chip, we utilize a breadboard setup with the Arduino bootloader as opposed to programming it directly on the PCB. To program this chip, we utilized a breadboard setup with the Arduino bootloader as opposed to programming it directly on the PCB. Figure 2 shows the schematic for this module.

2.3 Control Unit

The control unit is primarily comprised of the microcontroller which is an ATMEGA88A device and a port to connect an external programmer to program the microcontroller. The necessary features for our microcontroller are programmability using the Arduino IDE, running on 5 V, several analog I/O pins, the ability for every I/O pin to sink or source approximately 20 mA of current, at least 8 kB of program space (based on approximations of program size), and the ability to utilize UART communication. The ATMEGA88A was the most cost-efficient, mainstream microcontroller that met those specifications. The capacitor between the power pins on the microcontroller and ground are recommended by Microchip, who makes the device [2].

2.4 Sensor Block

There were many criteria to consider when choosing which sensors to utilize. First, we wanted the total power consumption to be a minimum, as the device is intended to be left

plugged in for extended periods of time. Additionally, we wanted to be able to classify multiple levels of sound and light. Lastly, we needed these components to be consistent when given an unchanging environment. Most of the design alternatives came when selecting the type of sensor to use for light. Photoresistors, photodiodes, and phototransistors would all allow us to identify changes in light and some basic light levels. However, using a photoresistor allowed us an analog reading of the light level which was ideal for our purposes. The photoresistor we chose minimized power consumption.

2.5 Software

An important part of our overall project was to allow the data collected from our devices to be accessible to the patrons interested in it before they even step foot in a new location. To make this a reality, we utilized our Wi-Fi module and a web application to display data in an easily digestible fashion. In our initial proposal, we suggested accomplishing this using network JSON files. However, due to the unforeseen complexity of directly modifying such files, and the additional insecurity the files themselves could pose, we transitioned to using a service called If-This-Then-That (IFTTT). The new flow of data is shown in Figure 3. This service receives our POST request containing a packet of sensor data and writes it to a Google Sheet that we have designated. The front-end web application then pulls this data. It places the light sensor data into one of five buckets: “very dark,” “dark,” “ambient lighting,” “bright,” and “very bright.” It does the same with the sound data, placing it into one of five buckets: “little background noise,” “moderate background noise,” “significant background noise or light music,” “significant background noise or loud music,” and “very loud.” The appropriate levels along with temperature and a timestamp are all displayed to the user under the location (or device) name on the web application. An example of how this might look is shown in Figure 4.

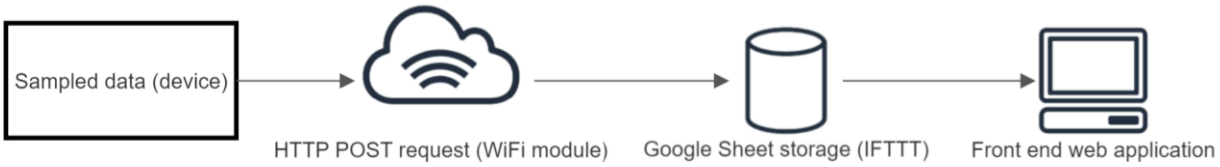

Figure 2. Flowchart of data path through software.

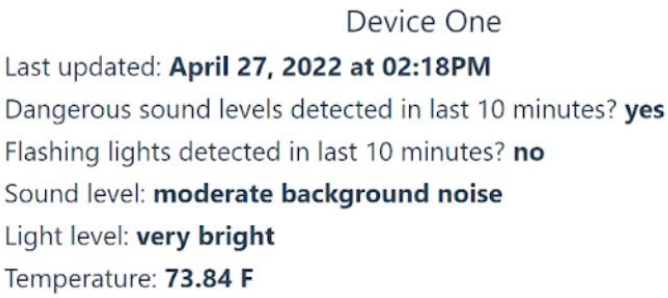

Figure 3. Screenshot of web application displaying interpreted data.

2.6 PCB Design

Our PCB measures about 3 inches by 4 inches and contains every component necessary for our device to function. Our original design is extremely similar to the final design, the differences being part footprint changes, an additional 2-connector pins for testing and backup plans, an additional through-hole micro-USB for better structural stability, and an additional capacitor-resistor circuit to increase stability in our temperature sensor. The final design is shown in Figure 5.

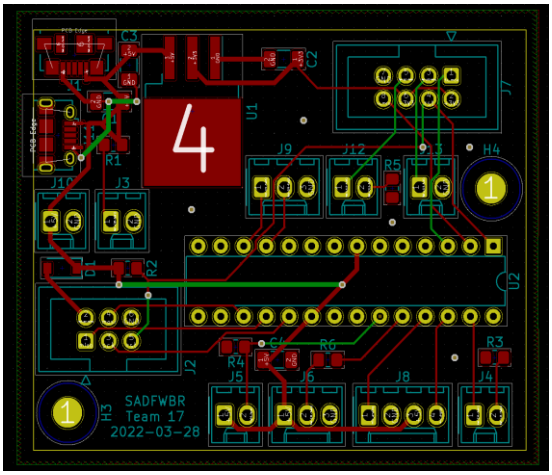

Figure 4. PCB Layout.

3. Design Verification

3.1 Sensor Characterization and Testing

All initial tests and characterizations described in the following section were done by connecting the applicable sensor to an Arduino as well as any necessary passives (resistors, capacitors). Specific voltage levels were recorded using the Serial Monitor function within the Arduino IDE.

3.1.1 Photoresistor

To test the photoresistors, we connected a GL5549 photoresistor in series with a 1 MΩ resistor between the 5 V and GND provided by the Arduino. We connected the node between the photoresistor and traditional resistor to an analog pin of the Arduino. We then read the value of that analog pin under different lighting conditions. During a bright day, we illuminated the photoresistor with a phone flashlight to gauge the resistance under intense light. We then waited and took another reading during the night in a pitch-black room with an additional cover over the sensor. The results of these tests and their comparison with the nominal values can be found in Table 1. In summary, the photoresistor was well within the factory specification.

Table 1. Photoresistor test results.

	v_dark (V)	dark resistance (kΩ)	v_light (V)	avg light resistance (kΩ)
nominal	x	10000	x	90
measured	0.45	10111	4.5	111
% diff	x	1.11%	x	23.46%

3.1.2 Temperature Sensor

To test the temperature sensor, we connected the TMP36 sensor's power and ground pins to the Arduino's power and ground and then connected the data pin to an analog pin on the Arduino. We then set a thermostat to a temperature, placed the device next to it and gave it a few minutes to reach a steady state. As seen in Table 2, the temperature result was not adequately accurate. According to the sensor's datasheet, a small resistor could be added to the data pin and a capacitor could be added across the power and ground terminals to increase accuracy, so we took both of those steps and retook the reading with much better accuracy. To confirm it was working, we again adjusted a group member's apartment thermostat to a new

temperature and took another reading. The temperature sensor reading was within a sufficient range for our device.

Table 2. Temperature test results.

	v_out (V)	calculated temperature (°F)	thermostat temperature (°F)	% difference
just TMP36	0.69	66.2	71	6.76%
	0.66	60.8	68	10.59%
TMP36 + passives	0.72	71.6	71	0.85%
	0.7	68	68	0.00%

3.1.3 Microphone

To test the microphone, we connected the microphone board to the power and ground of the Arduino, connected the digital output to a digital pin on the Arduino, and connected the analog output to an analog pin on the Arduino. To test the analog input, we simply had the Arduino IDE's Serial Monitor display the analog reading every few milliseconds. We watched the sensor stay remarkably consistent when there was minimal noise present. We then began doing activities like talking which created slight changes in the analog reading and then finally played music and saw large perturbations in the analog reading. We used the Serial Plot function on Arduino and saw that the result loosely resembled the song's waveform in an audio program.

We began calibrating the digital cutoff threshold by using the onboard potentiometer. To do this, we set the microphone directly next to a phone speaker and played the loudest song we could think of on repeat. The max sound output from an iPhone is 102 dB, which is right at the threshold for when sound can cause hearing loss [3]. We turned the potentiometer so that the digital reading would be a logical low when the song was playing a slightly quieter part or when there was just background noise and so that the reading would be a logical high during the loudest parts of the song. Doing this took a long time but resulted in a digital output that would trigger when the sound energy was above approximately 100 dB.

3.2 Microcontroller and PCB Testing

The next test was to see if we could program our microcontroller on our soldered PCB. To accomplish this, we connected the external programmer to the port on the PCB and connected the network status LED to the microcontroller. We then programmed the

microcontroller with a program to blink the LED on and off. We followed this by connecting the sensors in our sensor block to the PCB and uploaded multiple different test codes that would illuminate the LED based on the input from each sensor. Each sensor was working flawlessly with the microcontroller. We were unable to connect the Wi-Fi chip to the PCB at this time due to an issue with our PCB's schematic, but we fixed that issue for the second run.

3.3 Wi-Fi Chip Testing

To test the Wi-Fi chip, we first had to figure out how to program it. We combined information from a few different online tutorials to get a setup that would allow me to program the ESP8266 using the Arduino IDE and an Arduino board. To test if we could successfully upload a program to the board, we again created a blinky program and connected an LED to one of the two GPIO pins on the board which worked well. We then attempted to connect the board to a Wi-Fi network using the ESP8266 Arduino Library which worked flawlessly.

There were two further tests that needed to be done: one to test if the Wi-Fi chip could successfully edit a Google Sheet and another to test if the Wi-Fi chip could communicate with the microcontroller. To test the Wi-Fi connection, we programmed the Wi-Fi chip to upload dummy data into a test Google Sheet, which it was able to do repeatedly without fail.

To test the serial communication, we first programmed the Wi-Fi chip with a code to illuminate an LED if the sequence "abcdefghijklmnopqrstuvwxyz" was seen on the serial feed. Then we serially connected an Arduino to the Wi-Fi chip and programmed it to send some garbage sequences before sending the desired sequence. Unfortunately, the two devices were unable to communicate flawlessly. To fix that issue, we had to introduce a logic level converter to go between the 5 V of the microcontroller and the 3.3 V of the Wi-Fi chip. This allowed the two devices to communicate without error.

4. Costs

We will break down costs and theoretical anticipated costs in the following two subsections. All costs regarding mass-production are variable with regards to Bulk Purchase quantity.

4.1 Parts

Table 3. Parts Costs

Part	Manufacturer	Retail Cost ($)	Bulk Purchase Cost ($/unit)	Actual Cost ($)
LM1086 – Low Dropout Voltage Regulator	TI	0.90	30.30/10	30.30
ATMEGA328p – Microcontroller	Microchip Technology	2.73	13.65/5	13.65
LEDs – Power Block Light	Generic	sourced from leftover parts	sourced from leftover parts	0
4459s – Photoresistor	XINGYHENG	0.48	11.99/25	11.99
TMP36s – Temperature Sensor	KOOKYE	2.20	10.99/5	10.99
KY-038 – Microphone	DEVMO	4.66	13.99/3	13.99
ESP8226-12E – Wi-Fi Module	Esspressif Systems	3.25	12.99/4	12.99
Micro-USB Port – Power/Micro Controller Flashing	Generic	~0.80	7.97/10	7.97
Total	-	15.02	101.88	101.88

4.2 Labor

We estimated worker labor at around $40 per hour to approximate a realistic starting salary for a recently graduated ECE student with a bachelor's degree. We have also approximated this project to take around 5 hours of our time each week, leaving us with a total worker's compensation of $500.

Manual labor required to build a fully functioning device will cost around $103.125 for a 3-hour soldering/building process, flashing the device for each establishment's specific internet connection, and ensuring functionality.

In total, for each device with the parts we bought, and the time and effort put in to each one with fair compensation, our total cost comes out to $118.127 per device. This can be lowered by purchasing more of each part, lowering the worker's salary, and automating many processes that would otherwise be tedious for the worker.

5. Conclusions

5.1 Accomplishments

We exceeded many of the requirements set for ourselves. Our size requirement was lowered from a top profile of 100 inches^2 to 24 inches^2, and our volume was lowered from 200 inches^3 to 48 inches^3. Our "real-time" transmission requirement was set so that all measurements and alerts are updated every minute, instead of 10 minutes for measurements and 2 minutes for alerts. Through stress-testing, we found that our device can send measurements in increments of 5 seconds if required to, although this would greatly increase the power consumption of our device. All other requirements (aside from one) were met as shown in the requirements and verification tables in Appendix B.

5.2 Uncertainties

The single requirement not specifically met requires the device to be reliable, such that it does not need to be maintained "more than once a month." During testing, it would have been extremely difficult to test the durability of each component. As such, we have not verified the device's ability to run for one month continuously. However, from the research conducted when selecting components, we have no reason to believe that anything would fail without outside interference. We have no moving parts, nothing that is continuously running at dangerous voltages or currents, and no required maintenance such that human error could destroy or harm our device. Therefore, we feel confident in asserting that our device can run for more than a month without maintenance.

5.3 Ethical considerations

Since the purpose of our project seeks to improve accessibility of social gatherings in restaurants and bars, we have ensured that the development of our project respects all persons and does not discriminate against anyone especially against those with disabilities that our project may be useful for as outlined in [6, Principle 1.4] and [5, Sec. II]. Use of our device by establishments is completely optional, and furthermore continued use of our device and corresponding app is optional as well. At any time, users of our device and app can discontinue use, especially if undue harm to their customers or business occurs from the use of our project [6, Principle 1.2]. To respect privacy and honor confidentiality, our project does not collect any data that can be traced back to individuals. Our sensors do not transmit audio recordings, only signals corresponding to audio levels, and our web app does not collect private information [6, Principle 1.6]. The individual components of our project should pose no serious safety concerns because we intend to use them all within the manufacturer's guidelines.

5.4 Future work

If given more time, we would make changes to certain parts of our project and add more. Firstly, we would use more powerful components. Our ESP8266-12E is the second iteration of our Wi-Fi chip, since it has more RAM and plays nicer with our components. This also allows us to remove the current microcontroller, and put our program onto the new Wi-Fi chip, since it is completely integrated. We would also clean up our webpage and add additional features such as location information (menu, hours, address etc.), and a feedback form to improve our service at later dates. Finally, we would add music genre identification. Our main concern regarding music genre identification is with regards to privacy. This should not be a problem, as it will not be able to identify individuals or collect data that would identify individuals without their own consent.

References

[1] “LM1086 1.50-A Low Dropout Positive Voltage Regulator.” *Texas Instruments.* April 2015. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm1086.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1648685653577

[2] “ATmega48A/PA/88A/PA/168A/PA/328/P.” *Microchip.* 2020. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf

[3] J. Lapook, *When using headphones to listen to music, how loud is too loud for kids?*, CBS News, December 22, 2016. [Online]. Available: https://www.cbsnews.com/news/when-using-headphones-to-listen-to-music-how-loud-it-too-loud-for-kids/#:~:text=The%20top%20volume%20on%20an,for%20eight%20hours%20a%20day.

[4] I. Lopez, *Program the ESP8266 with the Arduino IDE in 3 simple steps,* Ubidots, June 13, 2016. [Online]. Available: https://help.ubidots.com/en/articles/928408-program-the-esp8266-with-the-arduino-ide-in-3-simple-steps

[5] “IEEE Code of Ethics”. IEEE Website. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed Feb. 7, 2022]

[6] “ACM Code of Ethics and Professional Conduct”. Association for Computing Machinery Website. [Online]. Available: https://www.acm.org/code-of-ethics [Accessed Feb. 7, 2022].

Appendix A: Subsystem Schematics

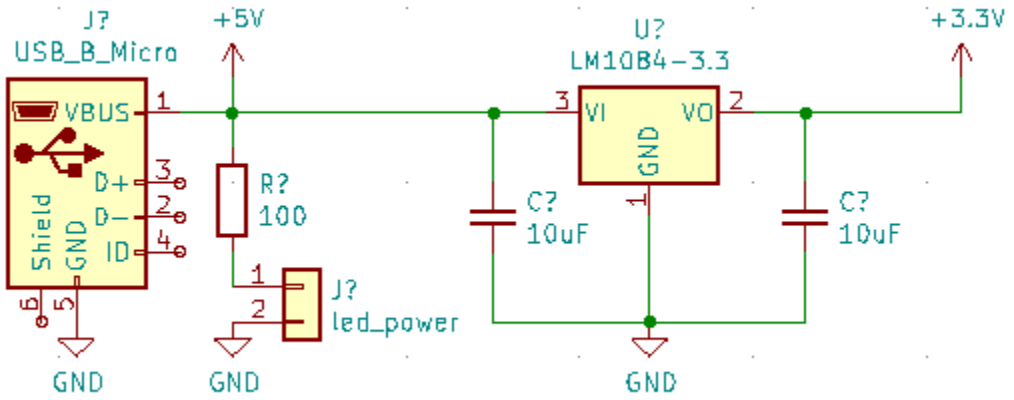

Figure 5. Circuit Power Supply Schematic.

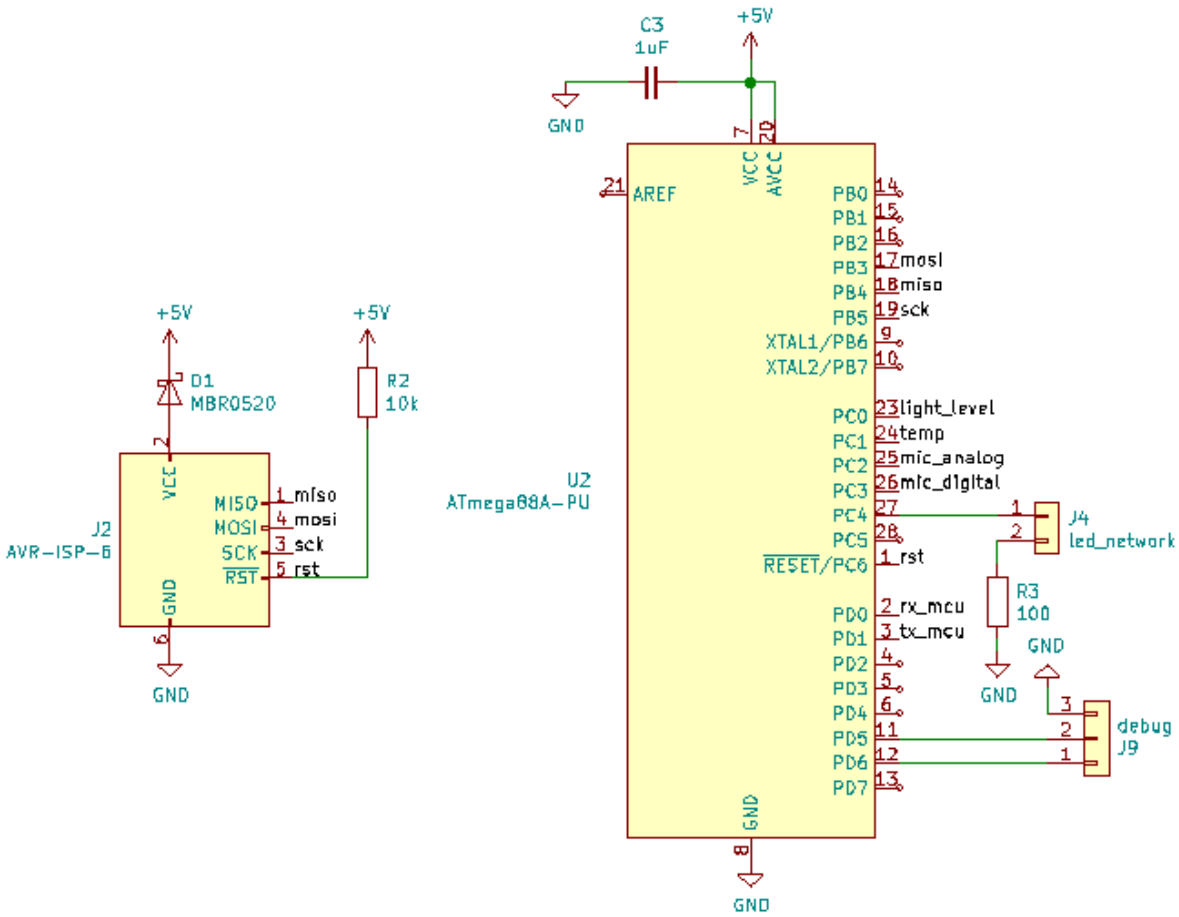

Figure 6. Microcontroller Schematic.

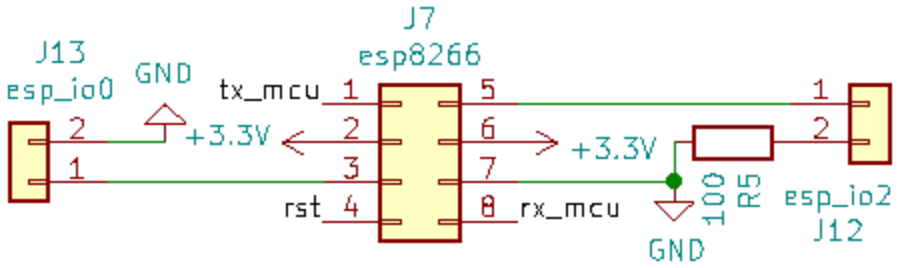

Figure 7. Wi-Fi Module

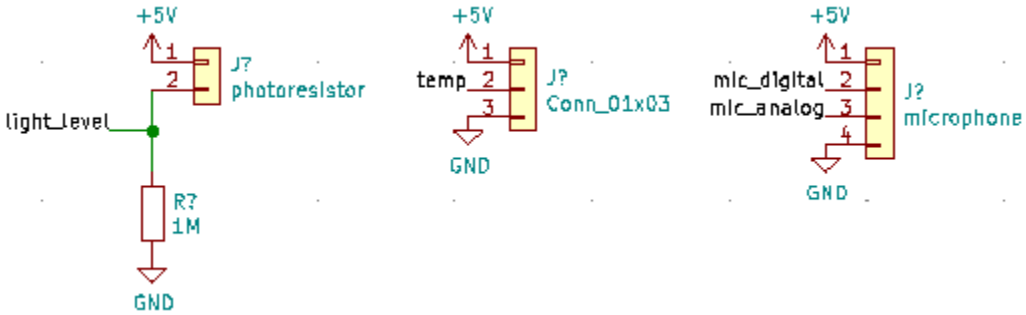

Figure 8. Sensor Block

Appendix B: Requirement and Verification Table

Table 4. Power Supply Requirements and Verifications

Requirements	Verification	Verification Status (Y/N)
1. The power supply provides 5 V +/- 0.5% from a wall power adaptor.	1A. Measure the output voltage from the wall power adaptor using an oscilloscope, ensuring that the output voltage stays within 0.5% of 5 V.	Y
2. The power supply provides a 3.3 V +/- 0.5% from a low dropout regulator driven by the 5V mentioned above.	2A. Measure the output voltage from the regulator using an oscilloscope, ensuring that the output voltage stays within 0.5% of 3.3 V.	Y
3. The power supply can operate within 0-1 A from the 5 V source and able to operate within 0-0.1 A from the 3.3 V source.	3A. For the 5 V source, connect the 5 V line to multiple resistors together such that the total resistance of the network is 5 Ω +/- 0.5% but no resistors are dissipating more power than they can handle. Measure the voltage network across the resistive network using an oscilloscope to confirm it is 5 V+/- 0.5%. 3B. For the 3.3 V source, connect the 3.3 V line to multiple resistors together such that the total resistance of the network is 3 Ω +/- 0.5% but no resistors are dissipating more power than they can handle. Measure the voltage network across the resistive network using an oscilloscope to confirm it is 3.3 V +/- 0.5%.	Y

Table 5. Control Unit Requirements and Verifications

Requirements	Verification	Verification Status (Y/N)
1. There must be a total of six GPIO pins that can appropriately handle signals between 0 V and 5 V +/- 0.5% while sinking/sourcing at least 20 mA +/- 0.5% of current per pin.	1A. Connect an I/O pin of the MCU to a series combination of an LED with a forward voltage of 3.2 V +/- 0.1 V and a resistor of 90 Ω +/- 0.5%. 1B. Using an ISP programmer, program the board with a test program that blinks the LED on and off.	Y
2. The RX pin must be able to interpret 3.3 V +/- 0.5% signals (from WiFi chip) as logical HIGHs.	2A. Use the same LED setup as 1A while also connecting the WiFi chip to the MCU. 2B. Using an ISP programmer, program the board with a test program that pings the WiFi chip and illuminates an LED upon a successful response.	Y
3. There must be an A/D converter with at least ten bits of resolution and three available channels which can be read sequentially.	3A. Use the same LED setup as 1A while also connecting the TMP36 to a different I/O pin while the sensor is in a temperature-controlled environment. 3B. Using an ISP programmer, program the board with a program that reads the temperature and illuminates the LED if the temperature is different from the initial reading. 3C. After the initial reading is complete, adjust the temperature of the environment by a small amount (~1°F) to ensure that the LED illuminates.	Y

Table 6. Wi-Fi Module Requirements & Verification

Requirements	Verification	Verification Status (Y/N)
1. The chip must be able to take commands via UART from an external microcontroller.	1A. Connect the chip to an Arduino (or similar microcontroller) using a breadboard. 1B. Upload a test program to the Arduino that attempts to ping the Wi-Fi chip and indicates using the serial monitor if a response was received.	Y
2. The RX pin must be able to handle a 5 V +/- 0.5% signal (from the microcontroller) without breaking the chip.	2A. Connect the RX pin to the microcontroller, using the same program as in Table 2 1A. Verify that the LED can illuminate, indicating a successful ping.	Y
3. The chip must be able to connect to a network and send 4 kB (max size) of data supplied by the microcontroller once every ten minutes +/- thirty seconds.	3A. After completing 2A, use the same hardware setup but change the software on the MCU using an ISP programmer to have the WiFi chip connect to a network and upload a dummy data packet to a JSON file once every ten minutes. 3B. Verify on a separate computer that the JSON file is modified once every ten minutes.	Y

Table 7. Sensor Block Requirements & Verification

Requirements	Verification	Verification Status (Y/N)
1. The output of the photoresistor must be within 0 V and 5 V +/- 0.5% with a current of no more than 20 mA +/- 0.5% in various light levels.	1A. On a breadboard, connect the photoresistor and a 1 M Ω +/- 0.5% resistor in series. Connect one lead of the photoresistor to 5 V using an Arduino (or similar microcontroller) and ground the circuit. 1B. Connect the second lead of the photoresistor to an analog I/O pin on the Arduino. 1C. Run a test program on the Arduino that records the voltage level of this pin. 1D. Use an oscilloscope to measure the current running through the resistors. 1E. Verify that the voltage stays between 0 - 5 V +/- 0.5% for different sound levels.	Y
2. The microphone output must be within 0 V and 5 V +/- 0.5% with a current of no more than 20 mA +/- 0.5% depending on sound levels.	2A. Power and ground the microphone chip using an Arduino (or similar microcontroller) 5 V output. Connect the analog output of the microphone to an analog I/O pin on the Arduino. 2B. Run a test program on the Arduino that records the voltage level of this pin. 2C. Use an oscilloscope to measure the current running through the microphone. 2D. Verify that the voltage stays between 0 - 5 V +/- 0.5% for different sound levels.	Y
3. The temperature sensor output must be within 0 V and 5 V +/- 0.5% with a current of no more than 20 mA +/- 0.5% depending on temperature.	3A. Using the same setup as in 2A - 2C, connect the output of the temperature sensor to the Arduino and run the same program. 3B. Verify that the voltage stays between 0 - 5 V +/- 0.5% and current less than 20 mA +/- 0.5% for different temperature levels.	Y

Table 8. Web Application Requirements & Verification

Requirements	Verification	Verification Status (Y/N)
1. The C++ program must take raw input data from 3 sensors over the span of 10 minutes and output averaged levels in terms of temperature, light, and sound.	1A. Using an ISP programmer, program the microcontroller with the C++ program. 1B. Connect the outputs of the 3 sensors to the input pins of the microcontroller. 1C. After 10 minutes, verify that the output values are reasonable for the environment in terms of temperature, light, and sound.	Y
2. Data from the microcontroller must be able to be transmitted via the WiFi module to the JSON file.	2A. Write a test program that can update a single value in a JSON file. 2B. Using an ISP programmer, program the microcontroller with the test program. 2C. Run the program and verify that the JSON file is updated.	Y
3. The web application must display updated data within 1 minute +/- 30 seconds.	3A. Edit the JSON file from which the web application reads its data. 3B. Refresh the web application. Continue to refresh the page every 2 seconds. Verify that the updated changes appear within 1 minute +/- 30 seconds.	Y