Public Safety Alarm

Final Report

By

Swetank Griyage

Sagar Katiyar

Adrian Wells

Final Report for ECE 445, Senior Design, Spring 2021

TA: Dean Biskup

3 May 2021

Project No. 35

Abstract

This report details the design, building and testing process of the Public Safety Alarm. The Public Safety Alarm consists of two major components, the Monitor Device itself and the Supervisor iPhone application. The Monitor consists of a microphone transducer and infrared video camera, allowing it to constantly analyze the scene it's set up in, while protecting the identities and details of the civilians present at the scene. It also consists of two microprocessors, allowing it to perform analysis on the audio input, and alert the supervisor via WiFi communication. The Supervisor Application is specifically designed to allow a supervisor to monitor key areas without always being physically present. In order to accomplish this conveniently, it is created as an iPhone application because a cellphone is something practically all individuals all keep on their person.

While we were unable to finish our system in its entirety, we made significant strides in each of the modular components of the system and describe our endeavor in this report.

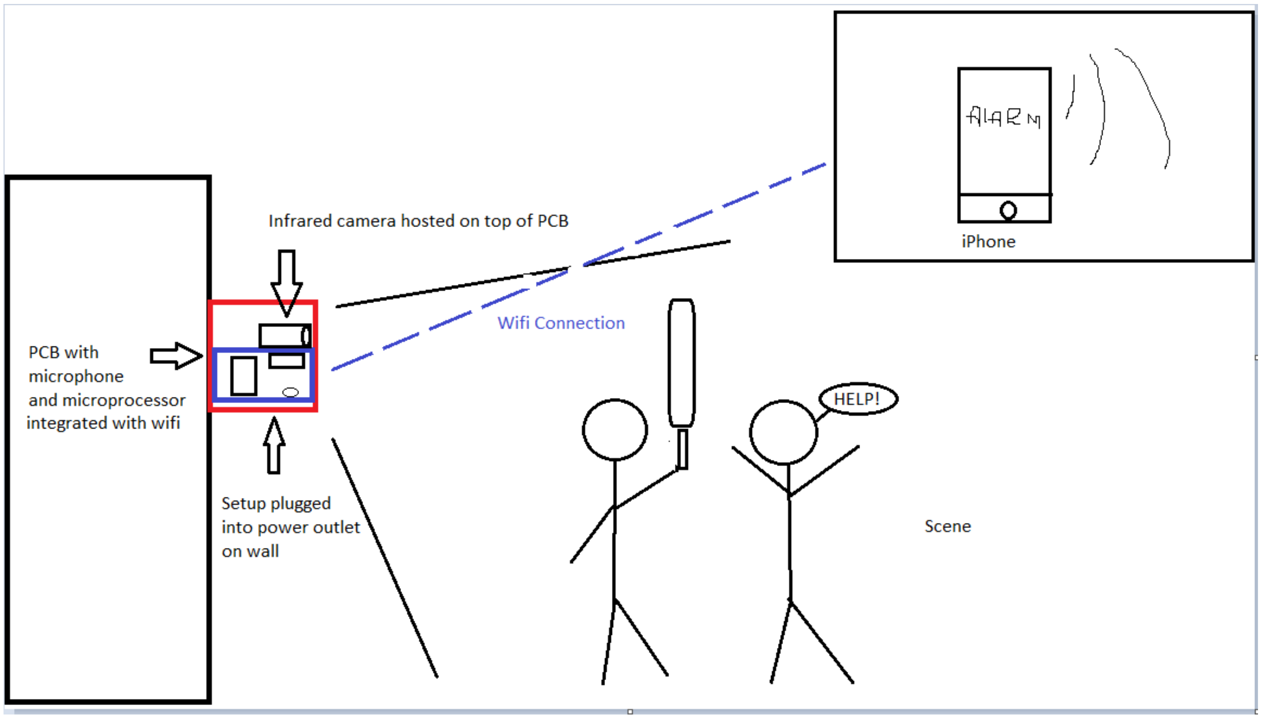
Contents

1. Introduction

In not just the United States but throughout the world, millions of people are victims of horrible crimes. Ranging from bullying to aggravated assault, many of these crimes either go unreported or don't result in conviction due to stigma or lack of evidence [1]. Furthermore, in locations such as school locker rooms or public restrooms, conventional video surveillance can't even be used since it violates privacy laws - resulting in crimes that cannot be prosecuted, let alone prevented. In fact, over 13% of students between the ages of 13 and 18 have reported that they have been bullied in locker rooms and washrooms, according to the National Center for Education Statistics and Bureau of Justice [2]. The objective of our project is to address this problem and monitor these locations that cannot usually be monitored.

The solution we propose consists of 2 main components - a monitoring system and a supervisor system. The monitoring system consists of an infrared camera and a microphone that constantly records the infrared video and sounds of the environment. The audio that is recorded is sent to a microprocessor hosting a decision model that does an energy spectral analysis and is able to differentiate screams from other forms of noise. If a scream is detected, it means that a dangerous situation is detected and a trigger signal along with the infrared footage of the scene is sent via Wi-Fi to the supervisor system. The supervisor system is essentially an iPhone application that receives the trigger signal and infrared footage and alerts the authority that a dangerous situation has been detected. The authority can then take action appropriately. There is also a feature that enables them to confirm whether the alarm was correct or not which would go in as feedback for our decision making model in order to improve it further.

Figure 1. Visual representation of our solution

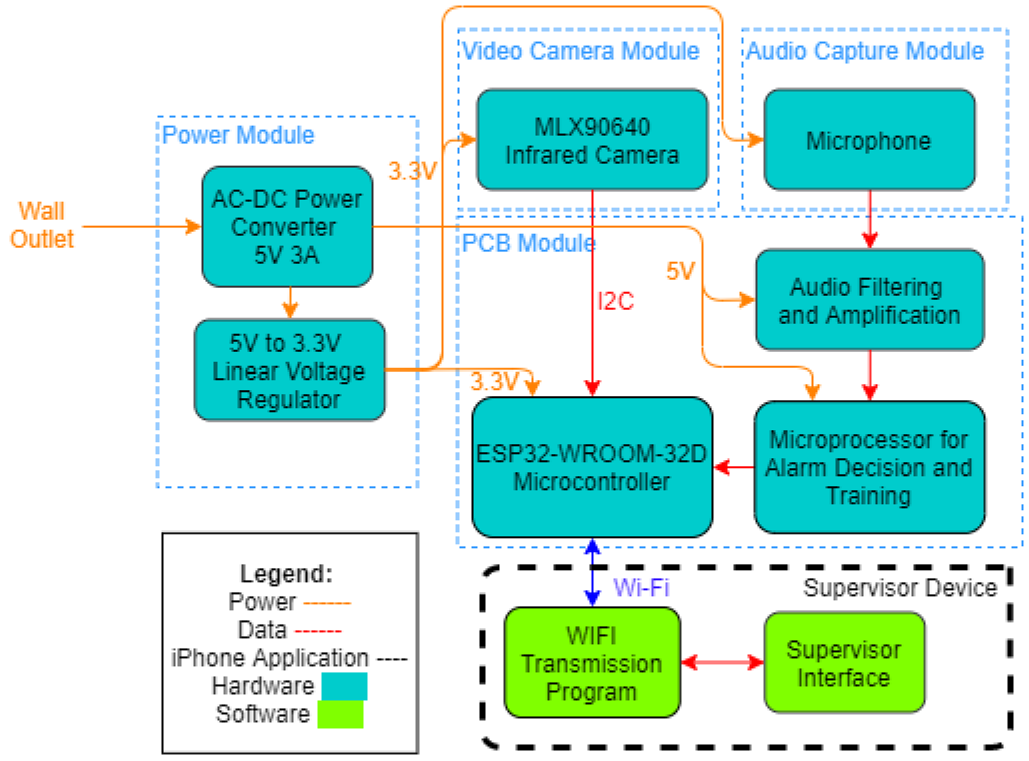

ALARM
iPhone
Infrared camera hosted on top of PCB
Wifi Connection
PCB with microphone and microprocessor integrated with wifi
HELP!
Setup plugged into power outlet on wall
Scene

2 Design

2.1 Block diagram

Figure 2 below represents the block diagram for our project. The power module is essentially responsible for powering all components of our monitoring system with a suitable voltage. The video camera and microphone modules are responsible for capturing the infrared video and sound respectively. While the input from the video camera module directly goes to the ESP32 microcontroller and then to the supervisor app via WiFi, the audio from the microphone first goes through our audio filter and amplification circuit through the microprocessor where our decision model is hosted and then to the ESP32 microcontroller. If the audio is deemed dangerous by the decision model, a trigger signal is also sent to the ESP32 which then sends the audio as well as the infrared video to the supervisor app on the iphone to be displayed. The app also produces a sound once the trigger signal is received to alert the authority present.

Figure 2. Block diagram for the Public Safety Alarm



2.2 Power Subsystem

The monitor's power subsystem is designed to make use of a standard AC outlet in order to drive all processes required by the monitor. This design choice was made because the monitor will be mounted in a stationary location and AC power is readily available in practically all use cases. In order to convert the AC power, we used a Meanwell AC/DC converter which supplies our PCB with 3A at 5V. We also use two 3.3V linear voltage regulators which allow us to drop down 2A from 5V to 3.3V. Using the 5V potential, we are able to drive the audio filtering and amplification circuit along with the M4 Arm microprocessor that houses our decision making model. With the 3.3V we power all digital signals, the ESP32 microcontroller and the infrared camera. Although 3A of power wasn't entirely necessary for our monitor design to date, we wanted excess power so that upgrades such as a better infrared camera could be made to our system without a power supply overhaul. [3]

2.3 Microphone and Amplification Circuit

The microphone and amplification circuit is hosted on the PCB and is connected to the STM32 microprocessors' analog input and ground pins. It is responsible for taking in audio input from the surroundings, filtering it so that only the frequencies of interest remain and amplifying the output so the signal can be easily interpreted by the STM32 that hosts the decision model.

The microphone that was used is the AOM-6738-PR as it is omnidirectional, can be powered using the 3.3 V power supply and can pick up frequencies between 30Hz and 15kHz. [4]

However, we are interested in a narrower frequency range than what can be picked up by the microphone and also needed the microphone to be more sensitive to the changes in the environment. As such a filtration and amplification circuit was used in order to make this possible. Figure 3 shows the filtration and amplification circuit that was constructed using an op-Amp and a combination of resistors and capacitors. [5]

LMV321 (U10) is the OpAmp we use to amplify the input from the microphone whereas the capacitors (C21 and C13) and the resistors (R18 and R17) were used to create our band pass filter.

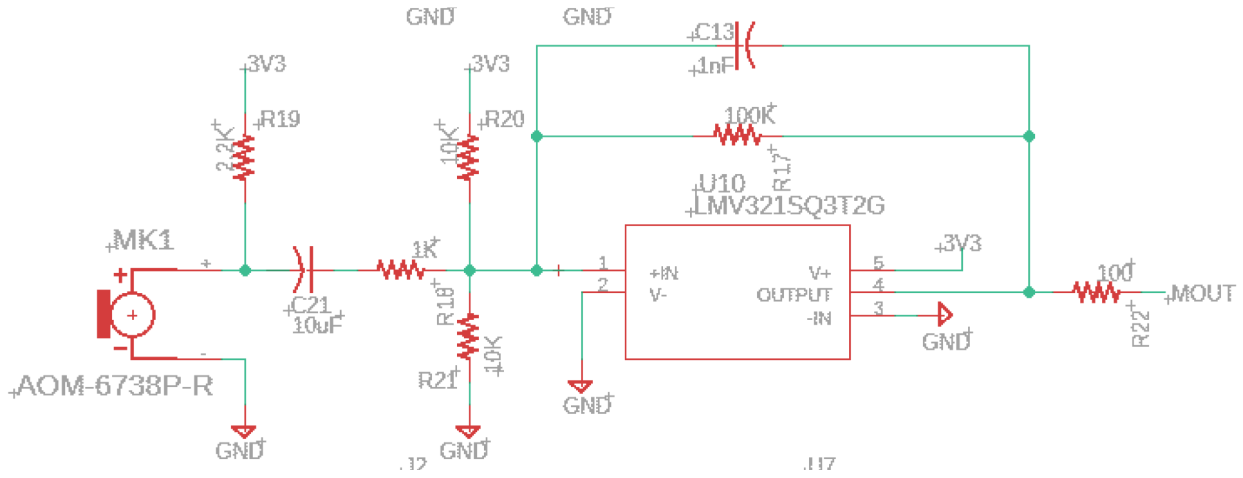
Our lowest frequency is determined by C21 and R18 and highest frequency is determined by C13 and R17 with the formula:

$$F = 1/2\pi RC$$

Causing our filter frequency range to be:

$$70.67\,Hz - 8.8kHz$$

Figure 3. Audio Filtering and Amplification Circuit

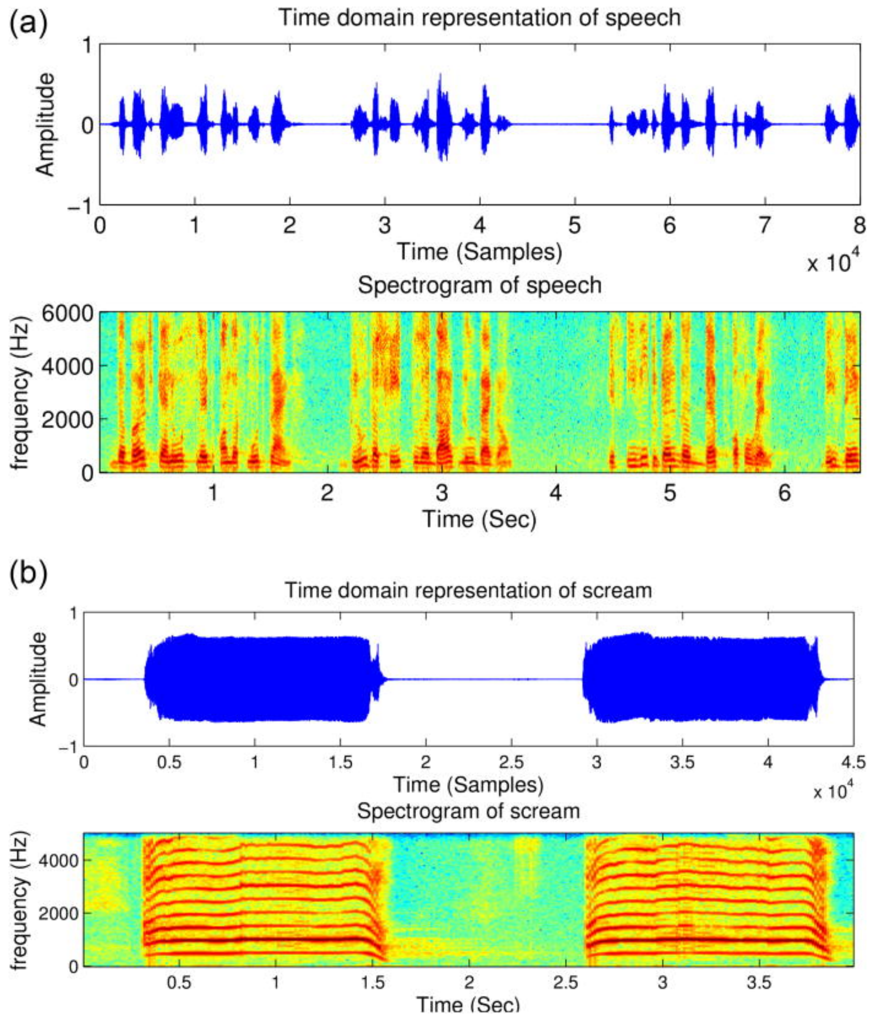


2.4 Emergency Decision Making Model

We base our decision model on the characteristics of the sound input received from the amplification circuit. The key detection here is the presence of a scream-like sound, which can be distinguished from normal speech, crying, or any other sound. At first, we considered implementing the module using a library called PyAudioAnalysis, but due to the heavy reliance on a large amount of labelled data, it was difficult to use it. In addition, while PyAudioAnalysis took .wav files, our microprocessor did not have the space to store it. Despite this constraint, there is a method for detecting screams solely on the basis of the energy vs time graphs and their Fast Fourier Transform. [6][7]

Figure 4

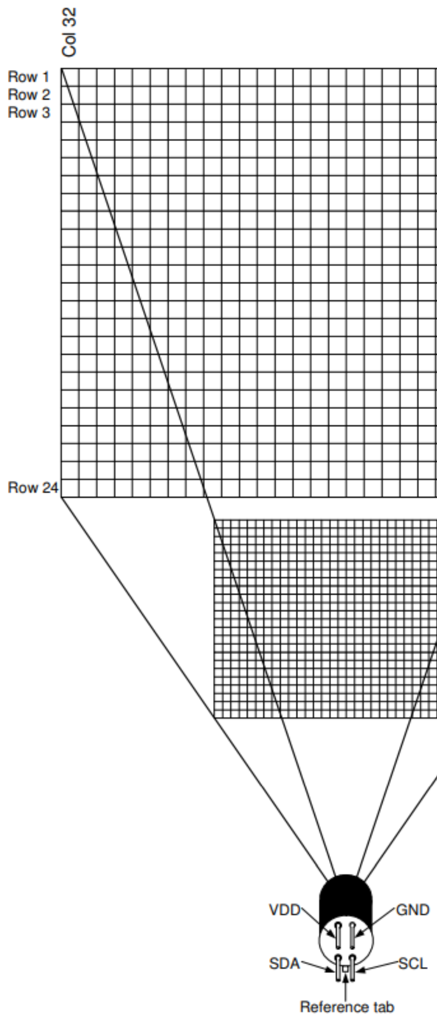
(Color online) Time and F0 domain representation of speech (top) and human scream (bottom)



2.5 Infrared Video Camera

We use the MLX90640 infrared video camera to take thermal images of the emergency scene. This camera provides us with a 32x24 array of floating temperatures ranging across a 110 degree field of vision. The camera operates at approximately 4Hz in our system and successfully detects the shape of a human up to 12 feet away. Our system is designed to be mounted on the wall with the camera lens angling down in the direction of the scene. As can be seen in *figure 5*, the MLX90640 has a wide enough field of view and enough pixels, that a general idea of the scene can be conveyed without revealing specific information about the individuals within view. The camera data is communicated to the ESP32 microcontroller using I^{2}C communication and then processed from there, which we will address in the next section. [8][9]

Figure 5. MLX90640 Field of View and Pixel Positioning [10]



2.6 WiFi Enabled MicroController

We use the ESP-WROOM-32D as the centerpiece of our monitors communication. The ESP32 is in charge of handling communication between the Monitor device and the Supervisor application, which includes relaying the emergency signal, sending the video data upon emergency, and relaying the supervisor feedback after an emergency event. This is performed by hosting a web server over the WiFi network the ESP32 either joins or creates. We chose to use a WiFi connection for communication rather than bluetooth because it allows the Supervisor to range farther away from the monitor while still being in communication. On top of this, WiFi is able to handle the data size at the speed we wanted in order to send large amounts of video data. [11]

2.6.1 Web Server Handling

The ESP32 web server is designed to create an IP address upon startup that can be used to access the various data of the monitor over WiFi. When visiting the main path of this IP address, the ESP32 handler will always respond with the status of the monitor, which can either be “System Monitoring” or “Emergency Detected”. The ESP32 is also designed to respond to a “/videostream” path which will respond with the most recent 20 frames of video as explained in the next section. The ESP32 also handles to “/PFeedback” and “/NFeedback” paths which allow the supervisor to simply touch these paths and signal to the ESP32 that positive or negative feedback needs to be relayed to the emergency decision making model about the most recent emergency.

2.6.2 Infrared Video Data Handling

As explained in the infrared video camera section, the MLX90640 is designed to output an array of 768 floating temperature values with each frame grab. In order to make use of this video data, we designed the ESP32 to do a bit of handling in between grabbing each frame in order to have the data ready to send at all times. As mentioned above, the video data is communicated to the supervisor application in the form of an http string response. After each frame of floating numbers is received from the MLX90640, the ESP32 maps each temperature to a color index between 0 to 255 depending on how warm the temperature is. Temperatures of 20° Celsius map to 0 while temperatures of 36° Celsius map to 255. The ESP32 then converts this index to a string and maintains a running string of the most recent 20 frames of video worth of color indices.

2.7 Supervisor Application

To accomplish our supervisor application needs, we chose to create from scratch an IOS iPhone application that would be deployable on any supervisors pre existing smartphone. We chose this because we wanted the alarm to be something very convenient to the supervisor, and something easily kept on his person at all times. This allows the supervisor to monitor the scene with a monitor set up without any other hardware or much change in their day to day routines. We made sure to integrate the app in a way that the monitor setup is extremely easy and inviting. We also added features that allow the supervisor to actively check the status of any number of monitors connected to his application with just one click of a button.

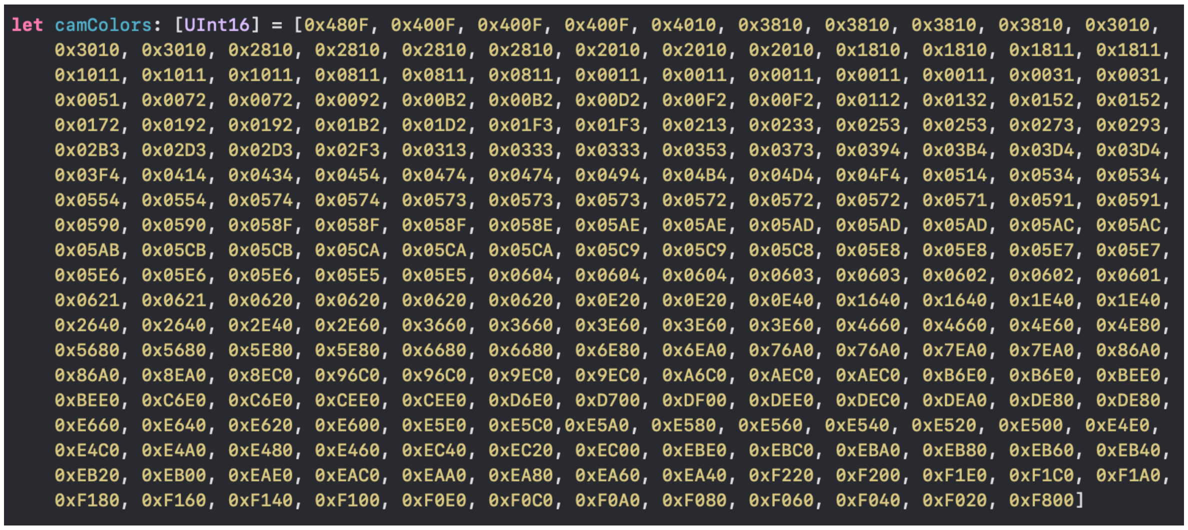
Once an emergency is triggered, the application is designed to automatically redirect the user to a screen displaying the infrared video stream, which is designed to capture 6 seconds before the emergency and 6 seconds afterwards. From here the supervisor can then move to the next screen which is configured to receive feedback from the supervisor on the event that just transpired.

2.7.1 Creating Images from Pixel Data

One of the most challenging aspects of our system was displaying the video data on the supervisor device. Although we originally attempted to set up a mjpeg stream on the web server itself, we decided that this option was easily hackable by malicious attacks and we didn’t want a video stream actively available for anyone with the correct IP address. Because of this, we decided to do the image construction in the back end of our supervisor application. As mentioned in the video camera and ESP32 mcu sections above, the video data is sent to the supervisor application as a string of color palette indices. After decoding the string back into an unsigned integer, our application then plugs each index into the color palette to receive the corresponding RGB pixel. *Figure 6.* shows the color palette array that we used, ranging from deep blue for cold colors to bright red for warm colors.

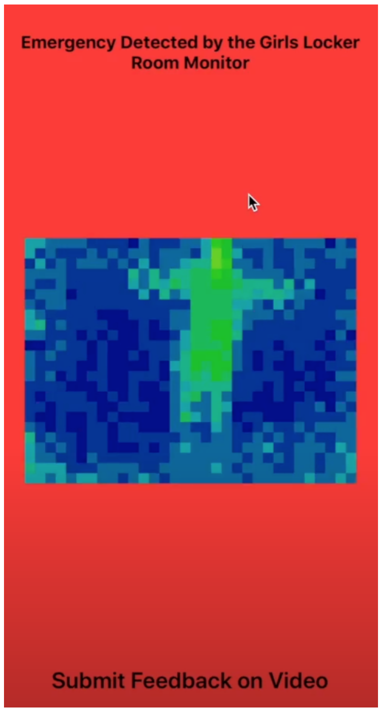
The application is then designed to combine each set of 768 pixels into an image structure. After each image is created, it is added to the application's image array which is then looped through, displaying each image on the screen for 250 ms to create a video stream effect. *Figure 7.* displays a screenshot of the supervisor app in the middle of the video stream with a person on scene.

Figure 6. Infrared Video Color Palette [10]



```
let camColors: [UInt16] = [0x480F, 0x400F, 0x400F, 0x400F, 0x4010, 0x3810, 0x3810, 0x3810, 0x3810, 0x3010,
    0x3010, 0x3010, 0x2810, 0x2810, 0x2810, 0x2810, 0x2010, 0x2010, 0x2010, 0x1810, 0x1810, 0x1811, 0x1811,
    0x1011, 0x1011, 0x1011, 0x0811, 0x0811, 0x0811, 0x0011, 0x0011, 0x0011, 0x0011, 0x0011, 0x0031, 0x0031,
    0x0051, 0x0072, 0x0072, 0x0092, 0x00B2, 0x00B2, 0x00D2, 0x00F2, 0x00F2, 0x0112, 0x0132, 0x0152, 0x0152,
    0x0172, 0x0192, 0x0192, 0x01B2, 0x01D2, 0x01F3, 0x01F3, 0x0213, 0x0233, 0x0253, 0x0253, 0x0273, 0x0293,
    0x02B3, 0x02D3, 0x02D3, 0x02F3, 0x0313, 0x0333, 0x0333, 0x0353, 0x0373, 0x0394, 0x03B4, 0x03D4, 0x03D4,
    0x03F4, 0x0414, 0x0434, 0x0454, 0x0474, 0x0474, 0x0494, 0x04B4, 0x04D4, 0x04F4, 0x0514, 0x0534, 0x0534,
    0x0554, 0x0554, 0x0574, 0x0574, 0x0573, 0x0573, 0x0573, 0x0572, 0x0572, 0x0572, 0x0571, 0x0591, 0x0591,
    0x0590, 0x0590, 0x058F, 0x058F, 0x058F, 0x058E, 0x05AE, 0x05AE, 0x05AD, 0x05AD, 0x05AD, 0x05AC, 0x05AC,
    0x05AB, 0x05CB, 0x05CB, 0x05CA, 0x05CA, 0x05CA, 0x05C9, 0x05C9, 0x05C8, 0x05E8, 0x05E8, 0x05E7, 0x05E7,
    0x05E6, 0x05E6, 0x05E6, 0x05E5, 0x05E5, 0x0604, 0x0604, 0x0604, 0x0603, 0x0603, 0x0602, 0x0602, 0x0601,
    0x0621, 0x0621, 0x0620, 0x0620, 0x0620, 0x0620, 0x0E20, 0x0E20, 0x0E40, 0x1640, 0x1640, 0x1E40, 0x1E40,
    0x2640, 0x2640, 0x2E40, 0x2E60, 0x3660, 0x3660, 0x3E60, 0x3E60, 0x3E60, 0x4660, 0x4660, 0x4E60, 0x4E80,
    0x5680, 0x5680, 0x5E80, 0x5E80, 0x6680, 0x6680, 0x6E80, 0x6EA0, 0x76A0, 0x76A0, 0x7EA0, 0x7EA0, 0x86A0,
    0x86A0, 0x8EA0, 0x8EC0, 0x96C0, 0x96C0, 0x9EC0, 0x9EC0, 0xA6C0, 0xAEC0, 0xAEC0, 0xB6E0, 0xB6E0, 0xBEE0,
    0xBEE0, 0xC6E0, 0xC6E0, 0xCEE0, 0xCEE0, 0xD6E0, 0xD700, 0xDF00, 0xDEE0, 0xDEC0, 0xDEA0, 0xDE80, 0xDE80,
    0xE660, 0xE640, 0xE620, 0xE600, 0xE5E0, 0xE5C0,0xE5A0, 0xE580, 0xE560, 0xE540, 0xE520, 0xE500, 0xE4E0,
    0xE4C0, 0xE4A0, 0xE480, 0xE460, 0xEC40, 0xEC20, 0xEC00, 0xEBE0, 0xEBC0, 0xEBA0, 0xEB80, 0xEB60, 0xEB40,
    0xEB20, 0xEB00, 0xEAE0, 0xEAC0, 0xEAA0, 0xEA80, 0xEA60, 0xEA40, 0xF220, 0xF200, 0xF1E0, 0xF1C0, 0xF1A0,
    0xF180, 0xF160, 0xF140, 0xF100, 0xF0E0, 0xF0C0, 0xF0A0, 0xF080, 0xF060, 0xF040, 0xF020, 0xF800]
```

Figure 7. Screenshot of Supervisor Application displaying Video Stream



3. Design Verification

3.1 Power Subsystem Verification

The power subsystem of our system was entirely successful in powering our monitor and easily verifiable. Because we more or less used an out of the box power converter we never had to second guess the power supply of our system. Upon receiving the power supply we performed the verification tests detailed in Appendix A. and were entirely satisfied.

3.2 Microphone-Amplifier and Emergency Decision Making Model Verification

For the microphone-amplifier, the goal was to have a circuit that would send a voltage output based on the volume. Because we were afraid of sending too high of an input, the sound input was not as pronounced.

While we were able to implement an algorithm for analyzing the sound input, it was not designed for the microphone/amplifier output. The algorithm only worked for the wav files. We were able to view some input values for the microphone-amplified from our chip, but it could not be analyzed the same way our wav-file generated graphs were. This is because the voltage for the circuit had to be clipped to prevent damage to our microprocessor chip.

In addition, while we were able to run an FFT algorithm on sound waves from the wav files, there was no final decision-maker based on the exponential decay curve fit. There was not enough time to implement and test it. Instead, it was simply represented by a button-pressed signal.

3.3 Infrared Video Camera Verification

The infrared video verification was successful and was tested via integration with the ESP32 MCU. We attached the ESP32 to the arduino serial monitor and encoded it to print the infrared video temperatures mapped to different ASCII characters in order to view the different temperatures within the view of the camera. This test successfully verified the infrared video camera to be fully functional and also fully integrated with the ESP32 MCU.

3.4 WiFi Enabled MicroController Verification

In order to verify the functionality of the MCU, we needed to validate the different roles of the ESP32 individually. As mentioned above, our first test was to ensure proper communication between the infrared video camera and the ESP32 which was done by printing to the serial monitor. Our next module to verify was the ESP32 WiFi webserver. To verify this, we connected our laptop or smartphone to the same WiFi network as the ESP32 and used google chrome to reach the ESP32’s designated IP address. We could then type the separate paths we needed to achieve proper communication with the ESP32, as mentioned in the design section. The final module to verify would have been the communication between the ESP32 MCU and the M4 MPU used to house the emergency decision making model. We were not able to verify the communication between the two because we never got them physically connected, but we did replicate the M4 MPU using a tactical button and a few LEDs. The tactical switch allowed us to trigger an emergency and the LEDs were then used to turn on when the supervisor submitted feedback. By doing this, we were able to successfully verify that the ESP32 successfully accomplished all of it’s roles within our systems design.

3.5 Supervisor Application Verification

Verification of the supervisor application by itself was a tall task as it’s sole role was to communicate with the ESP32 MCU. Because of this, we chose to verify the ESP32 MCU prior to beginning work on the supervisor application because we wanted to ensure that we would get to verify any progress made on the supervisor application. Since we already had the ESP32 MCU verified, we then verified communication between the Supervisor application by printing to the arduino serial monitor and by powering on the LEDs connected to the ESP32 when the supervisor application submitted feedback. In order to verify the video stream we triggered an emergency while a person walked in front of the camera, which allowed the video to clearly display the scene on the supervisor application. We were able to successfully accomplish all of the supervisor application needs and in turn verified all of the requirements.

4. Costs

4.1 Parts

Table 1. Breakdown of Individual Part Costs

Part Number	Description	Manufacturer	Quantity	Unit Cost	Total Cost
MLX90640	Infrared Camera	Adafruit	1	$59.95	$59.95
ESP-32-WROOM	WiFi MCU	Espressif Systems	1	$4.35	$4.35
STM32F405RGT6TR	Decision MPU	STMicroelectronics	1	$13.62	$13.62
AOM-6738P-R	Microphone	Projects Unlimited	1	$1.14	$1.14
PJ-202AH	Barrel Jack Female	CUI Devices	1	$0.75	$0.75
CA-2185	Barrel Jack Male	Tensility	1	$2.71	$2.71
BCS-104-L-S-TE	Header Connector	Samtec	1	$0.85	$0.85
LMV321SQ3T2G	Low Power OpAmp	Texas Instruments	1	$0.45	$0.45
3011041F0701	3 Pin AC Cord	GlobTek	1	$3.43	$3.43
1051330001	USB2.0 Port	Molex	1	$1.01	$1.01
CP2102-GMR	USB-UART Bridge	Silicon Labs	1	$3.54	3.54
ECS-80-20-5PX-TR	8MHz Crystal	ECS Inc	1	$0.27	$0.27
LFXTAL003000REEL	32.768 kHz Crystal	IQD Freq.	1	$0.57	$0.57
ECS-120-20-5PX-TR	12MHz Crystal	ECS Inc	1	$0.27	$0.27
LM15-23B05	AC/DC Power Converter	MeanWell	1	$8.22	$8.22
B3F-1052	Tactile Switch	Omron	2	$0.32	$0.64
USB4085-GF-A	USB2.0 C Port	GCT	1	$1.34	$1.34
AZ1117EH-3.3TRG1	Linear Voltage Regulator	Diodes Inc	2	$0.44	$0.88
BAT760-7	Schottky Diode	Diodes Inc	2	$0.58	$1.16
SS8050-G	NPN Transistor	Comchip	2	$0.23	$0.46
----	Capacitor Assortment	Samsung	1	$5.00	$5.00
----	Resistor Assortment	Bourns Inc	1	$5.00	$5.00

Total			**26**		**$115.61**

4.2 Labor

Table 2. Breakdown of Labor Costs

Team Member	Wage	Hours per Week	Weeks	Multiplier	Total
Adrian Wells	$50	30	12	2.5	$45,000
Swetank Griyage	$50	15	12	2.5	$22,500
Sagar Katiyar	$50	15	12	2.5	$22,500
Total	**$50**	**60**	**12**	2.5	**$90,000**

5. Conclusion

5.1 Accomplishments

We were successfully able to get video footage sent to a supervisor app. In addition, we were successful in getting a sound input into our microprocessor chip, which responded to higher volume. Apart from this, most of the decision model was working on my PC. Overall, individual subsystems were showing complete or partial functionality.

5.2 Ethical considerations

The biggest ethical concern in our project was protecting the privacy of the civilians monitored by our system. Data from locker rooms especially is highly sensitive. Since locker rooms and public spaces in general are areas where people might object to being recorded, we took extra care to protect their privacy by opting for a thermal infrared camera instead of the usual cameras. This way, key facial details, clothes, or any other accessories aren't made clear. Another point of concern for some users might be the storing voice data. However, because the chip does not have a capability to store wav files, all analyses must be done in real time.

5.3 Future work

In the future, to improve the quality of the decision model, we consider using infrared video footage rather than rely on sound. This would lower the need for the supervisor's judgement as such a model can detect such incidents reliably. Another area where we could expand our work is improving security. While the demo required us to show that we can connect the supervisor to the device through WiFi by just entering the IP address. However, this allows malicious people who want to avoid being detected to easily hack into the system. To prevent this, security encryption needs to be put into place.

References

[1] B. of J. Statistics, “Victimizations Not Reported to the Police, 2006-2010,” Bureau of Justice Statistics (BJS). [Online]. Available: https://www.bjs.gov/content/pub/press/vnrp0610pr.cfm. [Accessed: 02-Mar-2021].

[2] Wang, Ke, et al. “Indicators of School Crime and Safety.” National Center for Education Statistics, nces.ed.gov/pubs2020/2020063.pdf..

[3] “RS-15-5.” *DigiKey*, www.digikey.com/en/products/detail/mean-well-usa-inc/RS-15-5/7706168.

[4] “AOM-6738P-R.” DigiKey, www.digikey.com/en/products/detail/pui-audio-inc/AOM-6738P-R/3189921.

[5] “Audio Noise Filter Circuit.” ElectroSchematics.com, 8 Jan. 2013, www.electroschematics.com/audio-noise-filter/.

[6] Kong, et al. “Sound Event Detection and Time Frequency Segmentation from Weakly Labelled Data.” ArXiv.org, Cornell University, arxiv.org/pdf/1804.04715.pdf.

[7] Giannakopoulos, Theodoros. “PyAudioAnalysis: An Open-Source Python Library for Audio Signal Analysis.” PLOS ONE, Public Library of Science, journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0144610.

[8] Inspired Engineering, Melexis. “MLX90640 32x24 IR Array Datasheet .” Melexis, 2021.

[9] “Grove - Thermal Imaging Camera / IR Array MLX90640 110 Degree.” Seeed Studio, www.seeedstudio.com/Grove-Thermal-Imaging-Camera-IR-Array-MLX90640-110-degree-p-4334.html.

[10] “Adafruit MLX90640 IR Thermal Camera.” Adafruit, cdn-learn.adafruit.com/downloads/pdf/adafruit-mlx90640-ir-thermal-camera.pdf?timestamp=1581790802.

[11] Systems, Espressif. “ESP-WROOM-32 Datasheet.” Espressif Systems, 2021, www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf.

[12] “Espressif/Esp-Idf-Provisioning-Ios.” GitHub, Apache License, github.com/espressif/esp-idf-provisioning-ios.

[13] “IEEE Code of Ethics,” IEEE. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 02-Mar-2021].

[14] Schirm, Benjy. “Can Retail Stores Videotape Their Dressing Rooms? - Super Lawyers Illinois.” Super Lawyers, Super Lawyers, 7 Feb. 2021,

Appendix A Requirement and Verification Table

Requirements	Verification	Verification status (Y or N)
Infrared Camera must be able to detect a human from 10 feet away.	Mount the IR Camera and mark off 10 feet from the lens. From the marking, have 2 people stand side by side moving slowly. Ensure the video output displays 2 distinct warm figures. To unit test the Infrared Camera and ESP-32, we will be using the arduino serial monitor and printing ASCII characters to represent different heat values.	Yes
Infrared Camera must capture Infrared video at a frame rate of at least 2 Hz	Mount the IR camera and turn on the monitor system. Configure the ESP32 to print the time lapsed to the Serial Monitor after every frame grab. Ensure that 95% of the times remain under 500ms	Yes
The microphone must be able to take sounds ranging from 125 Hz to 8 kHz.	First ensure that the microphone is receiving proper input voltage and current supply. Next use a phone application to sweep the frequency from 125 Hz to 8 kHz. Use an Ohmmeter to ensure that the Microphone detects inputs throughout the entire range. We will be plotting the microphone output using the Putty Terminal	No
The microphone must detect a moderate yell from 10 feet away.	Ensure the microphone is receiving proper input voltage and current supply. Walk 10 feet away from the microphone and use a phone application to sweep a noise at 75% from 125 Hz to 8 kHz. Use an Ohmmeter to ensure that the Microphone detects inputs throughout the entire range from 10 feet away. We will be plotting the microphone output using the Putty Terminal	No
Audio Filter must only allow frequencies between 100Hz and 3Khz to be present in the input sound.	Make some sounds and let them be sampled. Then analyse the frequencies of the sample and see if the noise is being filtered out (consider Audacity software to analyse frequencies).	No
The pre-amp must be able to raise the input volume to the point that sounds from 10m away must be clearly audible.	Sounds must be fed into the microphone from the maximum range and checked to see if they are easily audible and distinguishable.	No

The decision making logic must be able to identify the code words like help.	Test the entire system by using the code words such as “help”, “stop”, “NO!” in a very loud and concerned manner. Check that the alarm at the supervisor interface is triggered.	No
The decision making logic must be able to identify scream waveforms correctly.	Play sample scream noises in front of the microphone along with some non malicious audio samples and check if the voice samples are correctly categorized by the decision making logic.	Yes
The ESP-32 Microcontroller must be able to send the video data and alarm trigger signal to the IOS device.	Without the decision making logic, just program the microcontroller to send the video feed from the infrared camera directly to the IOS device and see if it is being correctly received.	Yes
It must be able to receive the feedback signal from the IOS device.	Send a false alarm and true positive feedback messages from the IOS device and see if the decision making model is updated via the terminal. To test this even more modularly, display the communication signals between esp32 and decision making model using LEDs	Yes
IOS Application is able to receive a trigger signal over WiFi. IOS Application is able to send feedback to the microcontroller	Program the ESP-32 Microcontroller to send a test signal over to the IOS Device. Have the IOS Interface stripped down to merely display the alert was received upon arrival. Program the ESP-32 Microcontroller to send a high voltage to an unused output pin prior to dropping back down to ground when a signal is received. Then, strip down the IOS device to merely press a button to send a signal. Use LEDs to ensure that the MCU Output pin is raised to high voltage.	Yes
IOS Application is able to send an audible notification to alert the user. IOS Application is able to display emergency video upon receiving it. IOS Application provides two options for feedback after the video is done playing.	In order to verify all 3 requirements, we will need all of our system working besides the decision logic and microphone . We can program the ESP-32 Microcontroller to send an alert notification and then follow that notification with video data. We can then turn the notification volume on high and set the phone down. Upon receiving the signal the phone should ring. When we open the app, we should then check for infrared video. After 10 seconds of video, we can then expect the feedback questionnaire.	Yes

Power Supply must output 5V +- 0.1V.	Plug in the Power Supply and allow a few moments for it to reach a stable setting. Probe both terminals using an oscilloscope to ensure the voltage remains between 4.9V and 5.1V.	Yes
Power Supply must output 3A of current.	Attach Power Supply to constant current testing circuit. Adjust the variable resistor to ensure that ~3A of current. This can be measured using a multimeter. In the process, also ensure that the Voltage supplied remains between 4.9V and 5.1V.	Yes
Must Supply 3.3 V +- 0.1V to the Microphone Module.	Plug in the Power Supply and connect the Regulator. Allow a few moments for it to reach a stable setting. Probe both terminals using an oscilloscope to ensure the voltage remains between 3.2V and 3.4V.	Yes
Must Supply 1mA of current to the Microphone Module	Attach Voltage Regulator to constant current testing circuit. Adjust the variable resistor to ensure that 1mA of current. This can be measured using a multimeter. In the process, also ensure that the Voltage supplied remains between 1.4V and 1.6V	Yes