Project Minuteman

Nathan Cunanan, Nathaniel Dunn, Daniel Vicenti, Eric Watson, Adam Bush

Dept. of Electrical and Computer Engineering
University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — Our project aims to produce an inexpensive, minimal-profile device capable of discerning the direction of gunshots. There are numerous applications, with active shooter situations, military environments, and policing as primary targets. Previous devices have utilized sampling rates in the megahertz and laborious algorithmic approaches to source and analyze sound data for this purpose; we present a machine learning solution using multilateration and a standard audio sample rate as a viable alternative on scaled-down hardware.**

***Index Terms* — Neural networks, machine learning, multilateration, microcontroller, PCB, location, gunfire, recognition, digital signal processing, ADC.**

I. Introduction

The objective of this project was to create an audio sensor system that can be installed in multiple environments: on a building, at ground level, or on a moving vehicle. The Minuteman system can accurately detect gunfire discharges, being able to distinguish them from other sounds in the environment such as engine backfires or glass breaking. It discerns the precise direction of gunfire within the operational area of the sensor and then displays it visually.

This system in its basic form will be affordable for most facilities, such as schools, public buildings, or sites of large public gatherings. Police officers, military organizations, private security, and other authorities will be able to afford and utilize the Minuteman system. Although it is not the primary intention of the project, the device will also be functional for home security and private facilities looking for security. It is a goal of this project to encourage further development in technological security systems against gun violence.

II. Design Overview

A. Initial Design

The initial design for our Minuteman system comprised a series of distinct sensor devices utilizing triangulation. This version of the Minuteman system would monitor for audio signatures of a certain decibel intensity or higher, filtering out low intensity audio. It would have been only capable of securing facilities or specific locations, as locating gunshots would require the network of sensors to cooperatively pinpoint the location, while the central module determined if the audio was a gunshot or not.

These devices were to use a Wi-Fi network that would have allowed other security devices to utilize the data from the system and respond to threats.

B. Current Design

Physically, our device is 1m by 1m by 50cm, with the main PCB and microcontroller at its base. Five microphone arms extend 50cm out, one in each cardinal direction, and one which extends straight up, perpendicular to the ground.

The current design was modified to use multilateration instead of triangulation, since it was found to be more cost-effective and just as accurate to use a single device with multiple sensors, as opposed to using two localized networks of sensors. The task of detection was also passed off onto the program hosted on a computer instead, since any central hub designed would only have microcontroller resources that would be too slow to be useful. Instead it was used to send an audio stream to the computer hosting the Minuteman program via USB.

For detection, it was determined a neural network would cleanest solution to this task, and that instead of looking for audio that exceeded a minimum value, it would instead comb through audio continuously. This was to save resources and account for the fact that long-distance gunshots might fall beneath the volume threshold. Once a gunshot is identified, Minuteman monitors the audio for a signal peak and calculates the difference between the arrival times at the microphones during the same event.

Once multilateration has determined the direction, it sends the data to the program's UI, which will display the device's location using the GPS module, and visualize it over Google Maps in a satellite view of the area. A cone displays the direction of the detected gunshot.

III. Goals and Objectives

The goals of the project are:

- A single device with several microphones to pick up nearby audio.
- The device can be connected to a host computer and utilize a building's or vehicle's power supply.
- A microcontroller to connect the sensors and stream audio to the program.
- A program/app that can be downloaded to a host computer to manage the device and performs the majority of the software functions.

- The device is affordable and broadly effective for the needs of multiple classes of users.

IV. Requirements

The project's requirements are presented in table I.

TABLE I
PROJECT REQUIREMENTS

Identify gunshots with minimum	95%	Accuracy
Identify gunshots within	250	Meters
Identify and display a gunshot within:	2	Seconds
Identify the direction of gunshots to within:	5	Degrees
Maintain a cost of less than:	150	USD

V. Specifications

The specifications for the project are shown in table II.

TABLE II
PROJECT SPECIFICATIONS

The sensors are connected to a microcontroller, which is connected to the host computer program.
The microcontroller streams audio into a byte array on the host computer program.
The program checks the array for the muzzle blast of a gun using a convolutional neural network, searching for specific features in a scaleogram created by a continuous wavelet transformation.
The neural network is trained from a dataset of gunshots and non-gunshot audio files of 80ms length.
The network provides the program with a prediction. If it is a gunshot, it will pass the audio array to multilateration program to determine the direction of the gunshot.
Within 2 seconds of the gunshot, the program will send an alert about the gunshot and its direction from the device's location, as determined by GPS.

VI. Research and Theory

Research for this project explored the topics of multilateration, gunshot sound signatures, neural network architectures, and user interfaces.

A. Multilateration

Multilateration is a method widely used in real-time locating systems to locate a source by using time distance of arrival (TDOA) from receivers that are synchronized at the same clock rate. To implement multilateration for a two-dimensional case, a minimum of three receivers are needed, while the three-dimensional case requires a minimum of four receivers.

B. 2D Multilateration

Multilateration utilizes TDOA, since the time of arrival of the signal is not known. The distance between two receivers can be obtained by having a known propagation speed multiplied by the TDOA as shown in (1). R_i represents the distance, c represents the speed of sound, and τ_i is time the first microphone that received sound (t_0) subtracted from the ith microphone that received sound (t_i). The speed of sound is dependent highly on temperature, so a temperature probe is used to provide the value of T for the speed of sound calculation shown in (2).

$$R_i = c\tau_i = c(t_i - t_0) \tag{1}$$

$$c = 331.3\sqrt{1 + \frac{T}{273.15\ K}} \tag{2}$$

The x and y position of each microphone relative to the origin is known, so the Pythagorean theorem can be applied for R_i. The value of a_0^2 and b_0^2 becomes $(x - x_0)^2$ and $(y - y_0)^2$ for the first microphone, and $(x - x_i)^2$ and $(y - y_i)^2$ for the ith microphone as shown in (3). Taking the square root of these, the final product is shown in (4).

$$a_i = x - x_i \qquad b = y - y_i \tag{3}$$

$$\sqrt{{a_i}^2 + {b_i}^2} - \sqrt{{a_0}^2 + {b_0}^2} = c\tau_i = R_i \tag{4}$$

Given there are two unknowns for the TDOA of the first two microphones, an equation can be derived from the third receiver, which gives the two equations and two unknowns that can be solved, shown in (5) and (6).

$$\sqrt{{a_1}^2 + {b_1}^2} - \sqrt{{a_0}^2 + {b_0}^2} = c\tau_1 = R_1 \tag{5}$$

$$\sqrt{{a_2}^2 + {b_2}^2} - \sqrt{{a_0}^2 + {b_0}^2} = c\tau_2 = R_2 \tag{6}$$

With these two equations, the solution to x and y would be the intersection of two half-hyperbolas. In order to solve this non-linear equation, a guess and check algorithm would be required to solve the unknown variables of x and y, but it is computationally expensive. To alleviate this issue, a fourth microphone can be added to the sensor array and is used to remove the square root of the equation in order to linearize it, which can be solved as a system of linear equations.

C. 3D Multilateration

Using multilateration in the third-dimensional case, a fourth receiver is needed. Based on the equation derived in the two-dimensional case, the z position of the microphone relative to the origin is added, which is represented in (7). Taking the square root, the final product is shown in (8).

$$c_i = z - z_i \tag{7}$$

$$\sqrt{{a_i}^2 + {b_i}^2 + {c_i}^2} - \sqrt{{a_0}^2 + {b_0}^2 + {c_0}^2} = c\tau_i = R_i \tag{8}$$

With three equations, the three unknowns x, y, and z can be solved, as shown in (9), (10), and (11).

$$\sqrt{{a_1}^2 + {b_1}^2 + {c_1}^2} - \sqrt{{a_0}^2 + {b_0}^2 + {c_0}^2} = c\tau_1 = R_1 \tag{9}$$

$$\sqrt{{a_2}^2 + {b_2}^2 + {c_2}^2} - \sqrt{{a_0}^2 + {b_0}^2 + {c_0}^2} = c\tau_2 = R_2 \tag{10}$$

$$\sqrt{{a_3}^2 + {b_3}^2 + {c_3}^2} - \sqrt{{a_0}^2 + {b_0}^2 + {c_0}^2} = c\tau_3 = R_3 \tag{11}$$

With the addition of a fifth microphone, the square root can be removed, and a system of linear equations can be used to solve for the values of x, y, and z. In order to derive this, one of the square roots would be added to the right side of the equation, which can be substituted with the value in (12). Both sides of the equation would then be set to the power of 2, as shown in (13). After performing the FOIL method, the final product can then be rearranged to be set equal to 0 as shown in (14).

$$r_i = \sqrt{{a_i}^2 + {b_i}^2 + {c_i}^2} \tag{12}$$

$${r_i}^2 = (c\tau_i + r_0)^2 \tag{13}$$

$$c\tau_i + 2r_0 + \frac{r_0^2 - {r_i}^2}{c\tau_i} = 0 \tag{15}$$

By setting the equation of the ith microphone in (15) equal to the same equation using the jth microphone, the root term of 2r$_0$ can be removed, which linearizes the equation. The equation can then be written in the format shown in (16). The constant values of A$_i$, B$_i$, and C$_i$ shown in (17) – (19) allow the formation of a 3 x 3 matrix and D$_i$ in (20) allows the formation of a 3 x 1 matrix. Using these two matrices, the inverse matrix formula can be used to solve for the values of x, y, and z.

$$\mathrm{Ax} + \mathrm{By} + \mathrm{Cz} + \mathrm{D} = 0 \tag{16}$$

$$A_i = \frac{2}{c}\left(\frac{x_i}{\tau_i} - \frac{x_j}{\tau_j}\right) \tag{17}$$

$$B_i = \frac{2}{c}\left(\frac{y_i}{\tau_i} - \frac{y_j}{\tau_j}\right) \tag{18}$$

$$C_i = \frac{2}{c}\left(\frac{z_i}{\tau_i} - \frac{z_j}{\tau_j}\right) \tag{19}$$

$$D_i = c(\tau_i - \tau_j) + \frac{1}{c}\left(\frac{x_j^2 + y_j^2 + z_j^2}{\tau_j} - \frac{x_j^2 + y_i^2 + z_i^2}{\tau_i}\right) \tag{20}$$

D. Gunshot Sound Signature

Gunshots have a few notable regions in their sound signature. The first, rarely present, is the shockwave accumulating behind the bullet, if it is travelling both toward the receiver and above the speed of sound; this is unreliable and relatively quiet. The second, our region of interest, is the muzzle blast itself. It features the highest sound pressure level and is the first consistent sound to be noted after a firing. The third, which must be considered, is the reflection and echo of the muzzle blast, both the direct, off-the-ground reflection, and any reflections from any walls or surroundings. This can mirror the appearance of a gunshot, as seen below in Figure 1.

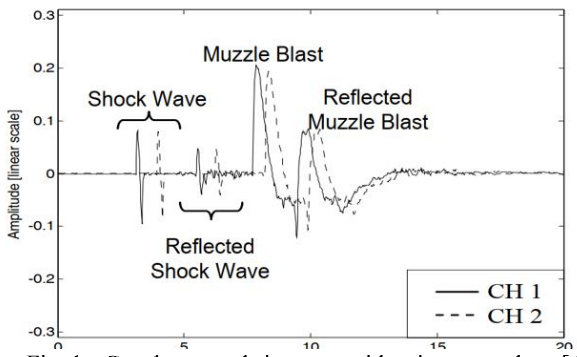

Fig. 1 Gunshot sound signature with primary markers [4]

E. Machine Learning

Machine learning is an important part of this project. The Minuteman system needs to be able to distinguish gunshots from normal audio. In order to do that, an algorithm will be trained from pre-established data to make predictions about the implications of new data. For this project we decided to develop an artificial neural network for classification of audio. ANNs are models simulating a network of neurons which are connected to a number of (or all) the neurons in previous and subsequent layers [1]. Each neuron takes a certain number of input connections with weights, which modifies each input's value. The neuron then sums them together and passes them into an activation function that determines the neuron's output [1]. Training for a classification problem using supervised learning (which means the data has labels which designates the correct class for the data) is done by:

1) Initializing weight values
2) Feeding the data through the layers to produce an output
3) Calculating the error of the output by comparing it to the input's true label
4) Adjusting the weights through backwards propagation, a process of determining the error for each node on the output layer, going to each node connected to the erroneous nodes, and then adjusting their connection weights, through the entire network

A convolutional neural network is an ANN consisting of special convolutional layers. Specifically, they generally consist of an input layer, a series of convolutional and pooling layers, one or more hidden fully-connected layers, and an output layer [3]. A convolutional layer works by creating a number of kernels that cover small areas of the data. These kernels, convolved over the entire input, creating several filters of the data. Unlike a fully-connected layer, there are significantly fewer neurons and connections to adjust, and repeated or similar weights are reduced as a result. The final output of the convolutional layer is a number of feature maps generated by each kernel.

After the convolutional layer, a pooling layer will downsample the data in the feature maps by merging neighboring cells in blocks. Typically, this is done by taking either the average of the data or the maximum value. This is important, as along with receptive fields, it increases the robustness of the analysis against data translation (or the perspective or change of perspective of the data of the sensor) [3].

Activation functions also influence the results of our network. For the Minuteman network, the tanh and sigmoid activation functions were trialed. Tanh is a popular activation function which competes with ReLU. While ReLU is typically preferred, especially for deep neural networks; however, since we are creating a binary classification network, tanh produced a better output alongside sigmoid. Tanh takes the sum through a hyperbolic tangent function. A sigmoid activation function is used for the final output since the output is binary. Sigmoid functions evaluate their inputs and produce an output between zero and one with a hyperbolic curve toward each. The final prediction is rounded to the closest number.

Dropout is used to reduce oversampling. Dropout randomly chooses connections and completely disables them, reducing the total connections and forcing the network to simplify the data as it proceeds.

Flatten is used to reshape the data into a format that can be passed into a fully-connected dense layer. The fully-connected layer is a simple layer where all of the neurons in the layer is connected to all of the possible inputs. This is needed to reduce the input to the final output, which only has a total number of neurons for each class. In this case, since our network is a binary classifier, it only has two.

In order to compile our network, we need an optimization and a loss function. The loss function calculates the 'loss', or the error in the network after it makes a prediction [2]. For Minuteman, the 'binary cross-entropy' function was selected, since that is designed for binary classification systems. The optimization function attempts to minimize the loss function by adjusting the learnable features of the algorithm. Minuteman utilizes the Adam optimization function.

A diagram of the Minuteman CNN architecture is shown in Fig. 2.

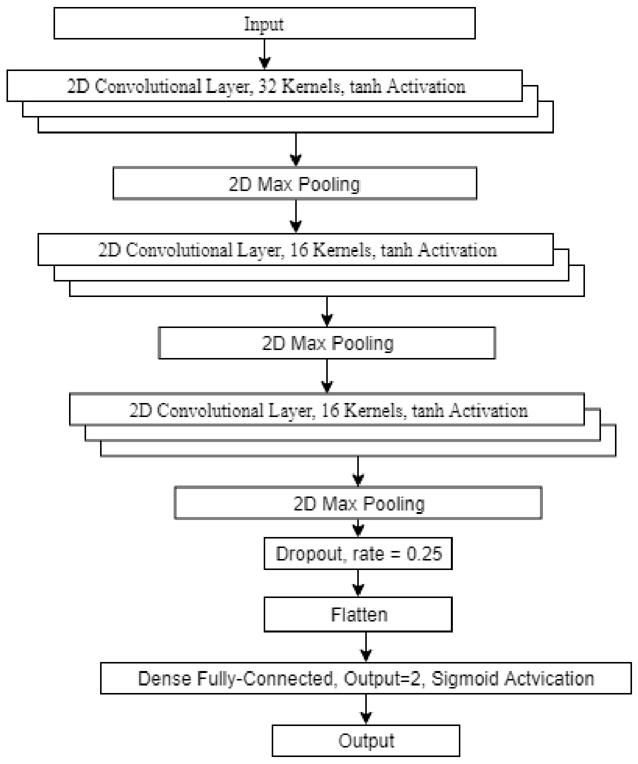
Input
2D Convolutional Layer, 32 Kernels, tanh Activation
2D Max Pooling
2D Convolutional Layer, 16 Kernels, tanh Activation
2D Max Pooling
2D Convolutional Layer, 16 Kernels, tanh Activation
2D Max Pooling
Dropout, rate = 0.25
Flatten
Dense Fully-Connected, Output=2, Sigmoid Actvication
Output

Fig. 2. A simplified view of the architecture for the Minuteman convolutional neural network

C. User Interface

One of the means of displaying the output in our Minuteman project is using a companion software that presents the data acquired by the other parts of the system, which are getting in the gunshots from firearms fired. There are many different options to be considered when efficiently creating a presentable, easy-to-use interface for the project's multiple components.

There are options in line in creating the right interface, ranging from using WPFs to using programming languages like C, C++, or Java. Looking into the right choices, we have concluded that using C#, one of the best programming languages in object-oriented programming, is the best one that can relay in the information coming from the sources of data that the interface will pick up. Microsoft Visual Studio, as researched, is a reliable application in the creation of a complex user-friendly interface such as used for our project. A Windows Forms library was utilized heavily as the primary method of lining up all the components properly, so that the user won't get confused when using buttons or accurately reading the information to be displayed on the application itself.

We found out that the Windows Forms using C# is convenient for this type of project because C# is a useful language in terms of building/designing and interactive software that the person can have control over in his/her own time over any other languages being used.

One of the main things discussed in as group early in the semester is the usage of Google Maps satellite to appropriately show and mark where the microphone setup is located and the general vector direction of where the gunshots are fired. Using some sort of map will properly display what is being asked by the system so that whoever is using the companion software knows what is happening and isn't lost on what the markers mean eventually. We initially planned on just displaying some information about said gunshots, but we have determined it was way more practical to present.

VII. Hardware Design

Our requirements demand an array of audio sources, an orientation sensor, a GPS module, and a microcontroller, with the MCU gathering and forwarding all signals to the host PC. The audio sources themselves will each consist of a microphone and an amplification and filtering stage, and they will all tie into a simultaneous sampling ADC. Primary elements of the system are shown below in Table III.

Table III. Primary hardware elements

Purpose	Device
Microcontroller	ATSAM3X8E
GPS	u-blox NEO-6M
Orientation Sensor	Bosch BNO055
ADC	ADS8586S
Amplifier	TL072
Microphone	CMC-6035-130T

A. System Overview

Each microphone will logically connect to the MCU after the analog handling of the signal, as will the two other sensors. This data will be routed to the host PC, which will first check the audio for the target sound signature. After a sound signature match, the multilateration algorithm will comb through the relevant data, looking for identifying markers – namely the muzzle blast's peak and valley of pressure, as represented in the audio waveform – computing the direction of the source. The direction and location information will be passed to the application's GUI, which will use that information and the statically calculated timestamp to display that data using a Google Maps API. This can be seen in Figure 3 below.

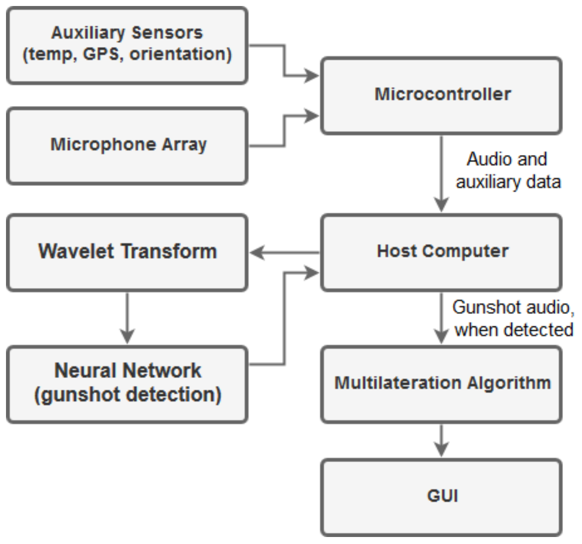
Auxiliary Sensors (temp, GPS, orientation)
Microphone Array
Microcontroller
Audio and auxiliary data
Wavelet Transform
Host Computer
Neural Network (gunshot detection)
Gunshot audio, when detected
Multilateration Algorithm
GUI

Fig. 3. System Block Diagram

B. Microphone

Our microphone model is a small, omnidirectional condenser microphone, which produces a small AC output voltage. It is rated to a maximum input level of 130dB, which is safely operational for nearly all applications – only point-blank firings of high-powered guns threaten to break that threshold, and these are naturally outside of our interest. A sound pressure above that level would merely result in a non-representative span of recording; it would not damage the microphone.

Our device has a typical output range of +/- 8 mV and a measured maximum of +/- 25 mV (at well over 100dB), which is later amplified.

C. ADC

The ADC operates for up to six channels at a maximum of 250k samples per second as a function of its primary timings – 1 microsecond for signal acquisition before sampling and roughly 3 microseconds for the subsequent conversion and data reads. Running at our target rate of 44.1k samples per second, this leaves a period of just over 18 microseconds between each sampling period for other operations to be performed by the MCU, as controlling the ADC is an operation outside the scope of the peripheral controllers and DMA controller of the MCU. It samples all signals on command, holding those values for the conversion process, which is necessary for our time-dependent multilateration algorithm.

D. Absolute Orientation Sensor

The combined gyroscope, accelerometer, and magnetometer module computes the orientation of our device in space, returning the Euler angles of our device, which allows us to align the sensed direction vector and produce a meaningful output. The device also features a temperature monitor, allowing us to find and use an accurate value for the speed of sound, which is temperature-variant.

E. GPS

The simple GPS module was chosen primarily for its relatively fast cold-start time of 32 seconds. The device will be powered on for some time before USB connection is initiated, so this latency is easily hidden. It returns standard longitude and latitude values which easily feed into Google Maps.

F. MCU

The microcontroller receives and centralizes all data before transmitting it to the host PC via USB. Important characteristics are its USB 2.0 Hi-Speed support and its 84MHz clock rate.

G. PCB Layout

The Minuteman project will use the Unite States standard 120V AC at 60 Hz to convert it down using a wall adaptor to 12V DC at 2 Amps. This 12V DC will then go through the dual output LTM4622A and single output LMR23610ADDA to create the 3.3V DC, 5V DC, and the 10V DC. In the final stage, the 10V DC will be inverted by the MAX1044 to -10V DC. These voltages have a high-power efficiency of 89% for 3.3V DC, 91% for 5V DC, and 95.6% for 10V DC.

Both the central module PCB and microphone PCB were designed using the KiCad software because of its large component library and its zero monetary cost. These boards were manufactured by JLCPCB with a central module PCB size of 91mm*114mm and the microphone PCB size of 57mm*38mm. The size of both PCBs designs gave ample room for tracing and correct sectioning without being oversized.

The central PCB was divided into six sections. Two power sections, one for each integrated circuit, so minimal interference would affect our other sections. The ADC section is split between the analog input of the microphone connector and the digital output going into the microcontroller (uC). The uC section being placed in the middle of the board so that the ADC section and the USB section could be separated from the power sections. The USB section was placed close to the uC, because it requires minimal tracing, corners and vias to mitigate interference with high speed data transfer. The final JTAG and sensor section at the bottom of the PCB. The sectioning of the PCB makes it easy to troubleshoot and solder.

The microphone PCB is simpler in its design and layout, with the connector pins to the right providing power and output to the microphone PCB. The center of the PCB is used for audio amplification and the band-pass filter of 72 Hz to 7.6 kHz. The cutoff frequency is ideal for the audio samples that the Minuteman requires. The right section of the PCB contains the CMC-6035-130T microphone and the 5V DC power source it requires.

VIII. Software Design

A. Cyclic MCU Timing Window

While transmitting to the host PC, audio samples for all five microphones are required at the consistent, precise period of 1/44100Hz – every 22.7 microseconds. These can be generated and gathered in roughly 4 microseconds, leaving a window of just over 18 microseconds for communication blocks. UART I/O is channeled through the Peripheral DMA Controller (PDC), using a few of these time periods per second to coordinate and accumulate that data. USB transmissions are performed in blocks of 512 bytes, requiring 512B / 480MBps = roughly 1 microsecond for the data itself to send. This fits with comfortable overhead for the protocol itself and for the host PC to attend to the USB channel.

Both the UART management and the USB signaling will occur infrequently as new timing windows come along, with the UART needs managed by a timer and the USB needs taking priority due to buffer management considerations.

B. USB Packet Format

The need to differentiate sensor data like latitude, longitude, and orientation, each with unique timings, requires a signifier in the data packet to the host PC. One byte of control bits is sufficient for this purpose, and, as each audio sample is two bytes in size, the spare, odd byte is used as a simple incrementing packet number counter, shown below in Figure 4.

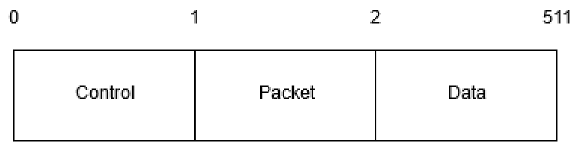

Fig. 4 Example 512-byte USB packet

C. Neural Network Implementation

The Neural Network for Minuteman is trained using Keras, which is an API using a TensorFlow framework. TensorFlow is notable for its development and useful libraries for neural network projects. TensorFlow specifically allows the usage of a GPU to calculate the model, which is better optimized for large models of data. Keras in particular is easy to use and greatly simplifies the process of creating and compiling the network, as well as running it.

Since the audio stream from the USB is constantly updating within a short time-span, the model for the network had to be converted to the programming language used for the UI, to avoid having to implement a pipeline, or other complicated inter-program system that might create an unnecessary delay. For this purpose, a library called TensorFlow Sharp was used to import a graph of the trained model onto the program in C#. Then it can quickly predict the input from the audio stream, informing the system if it needs to run multilateration analysis or if it should continue sending audio streams for gunshot detection.

D. Interface Implementation

It was an easy task managing all the different components in the User Interface. The team decided on making the GUI simple and peaceful to look at. Nothing too out of the ordinary and well balanced so that the one using the program can travel to every page properly. The main duty in creating this program is that it works the right way, which means that it should be as error free as possible. There was no prior experience to C# for the group, so we had to learn another programming language along the way. In that way, we can gain experience for future endeavors and all priorities that might relate to a programming language like this. C# isn’t any different to Java or C because it has similar methods in creating such code.

The plan is to have a sign in/register system when the program is launched. This way, users can have their own accounts to be able to log in/log out with ease whenever they want to. Once the user has logged into the main program, the user enters the main interface that shows buttons like the record/stop, a log for gunshots which state a timestamp of all things captured by the microphone setup and will display an alert if a gunshot is detected or not.

The biggest chunk of the interface is the Google Maps satellite view that we planned on experimenting with early in the semester. What we chose the open source NuGet package called Gmap.NET that illustrates different kinds of maps/satellite for viewing and the ability to use markers and arrows to show direction of mark coordinates on the map itself. In this project, we have illustrated directional arrows coming out of the microphone setup to show the general direction of the audio sources captured by the setup.

IX. Conclusion

Results have been promising, as we met or exceeded all stated requirements, with a final neural network accuracy of 99.17%. This was measured against a test set riddled with strongly misleading null samples such as loud impacts, background highway traffic, and generator whir.

Our system also lends itself to a natural extensibility through multiple networked devices, because it ultimately outputs a unit vector and the receiver’s fixed, known origin. The host code requires only a minor addition to network two or more nearby devices to find the intersection of these produced lines and thus the exact location of the sound source in 3D space. This is rarely a large improvement over a precise direction, but the implementation cost is so low that it seems highly practical if one were to expect to operate multiple units in the same general area.

With an extended training dataset, this setup looks ready to receive and detect even low-volume gunshots, running on minimal-cost hardware and with an easily portable, mobile frame.

References

[1] A. Kaushik, A.K. Soni, Rachna Soni, *A Simple Neural Network Approach to Software Cost Estimation.* Global Journal of Computer Science and Technology: Neural & Artificial Intelligence, Vol. 13, Issue 1, Version 1.0, Year 2013.

[2] J. Brownlee, "Loss and Loss Functions for Training Deep Learning Neural Networks," *Machine Learning Mastery*, 28-Jan-2019. [Online]. Available: https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/. [Accessed: 19-Jul-2019].

[3] Piczak, Karol J. ENVIRONMENTAL SOUND CLASSIFICATION WITH CONVOLUTIONAL ...Institute of Electronic Systems Warsaw University of Technology, http://karol.piczak.com/papers/Piczak2015-ESC-ConvNet.pdf.

[4] R. C. Maher and S. R. Shaw, "Deciphering gunshot recordings," in Proc. Audio Engineering Society 33rd Conf., Audio Forensics—Theory and Practice, Denver, CO, June 2008, pp. 1–8.

Biographies

Nathan Cunanan is a graduating Computer Engineering major from the University of Central Florida. He is planning to apply to Lockheed Martin or Harris Corp. as a software engineer after he graduates from the university.

Nathaniel Dunn is a graduating Electrical Engineering major at the University of Central Florida. He has been working as an intern within his field and plans to continue his professional career post-graduation.

Daniel Vicenti is a Computer Engineering major. He plans on pursuing a software engineering career with an emphasis on neural networks and machine learning.

Eric Watson is a graduating Electrical Engineering and Computer Engineering dual major at the University of Central Florida. He is planning to attend graduate school to further develop his interests within his field.

Adam Bush is a graduating Computer Engineering major at the University of Central Florida. He spent one year working with UCF's Computer Architecture Lab on an NSF Research Experience for Undergraduates grant and plans to work within the lowest abstraction layers of computer architecture.