Distributed Sound Event Location System - DiSEL

Mark Judy, Drew King, Christopher Santana, Brandon LaGuerre

University of Central Florida, Department of Computer and Electrical Engineering, Orlando, Florida 32816

***Abstract* — The objective of this project was to develop a system that utilized a network of sensors that listen to their surrounding sounds and use this data to determine the origin location of a specific sound event and classify a variety of different sounds. The origin location of a desired sound was achieved by sending GPS data from triggered sensors through Wi-Fi to a main server that determines the sound's approximate location using multilateration. A machine learning algorithm was used to listen to the audio stream and determine if a particular sound event is one of the desired sounds and classify what caused that sound.**

Index Terms* — *multilateration, machine learning, sound classification, GPS., TOA, MFCC

I. Introduction

Sound is a powerful tool that is utilized in many unique ways by humans and animals alike to assist us in different situations from safety and survival, communication, entertainment, and more. This project was envisioned to combine these different elements to a system that can be used in a variety of applications.

The Distributed Sound Event Location System (DiSEL) implements a network of devices around a desired area that listen to their surrounding sounds and in the event a user defined sound event occurs, the system will attempt to classify the sound that occurred and determine the sound's origin location. The resultant information will then be displayed for the user on the DiSEL website.

II. Design Overview

A. System Overview

Multiple sensors will capture audio data, all these sensors will be connected within a centralized server. The sensors will continuously send the server the location of the sensor, the time of arrival (TOA) of all captured audio data, and the audio data itself.

The centralized server will use TOA and multilateration to determine the origin location for single tone sound events. The centralized server will then send latitude, and longitude along with a timestamp of the location of the sound event to a separate web server. The sensors will also record audio data to be used for the classification of bird species, through their bird calls. The centralized server will send the information regarding these events to be analyzed by a separate server running a machine learning algorithm. This machine learning server will grab the audio data from the sensor server, load the sound event, and classify the event based on the twenty-nine trained bird species. The classification result, along with a date and time stamp will then be sent to the web server.

Once either the location event data, or the classification event data are received by the web server, this data will then be saved inside of a database. This stored data will then be accessed by the user when they sign in and access a dashboard within the web application and will then be able to view and interact with the data. The web application will then continuously update the dash with newly saved data at a predetermined interval.

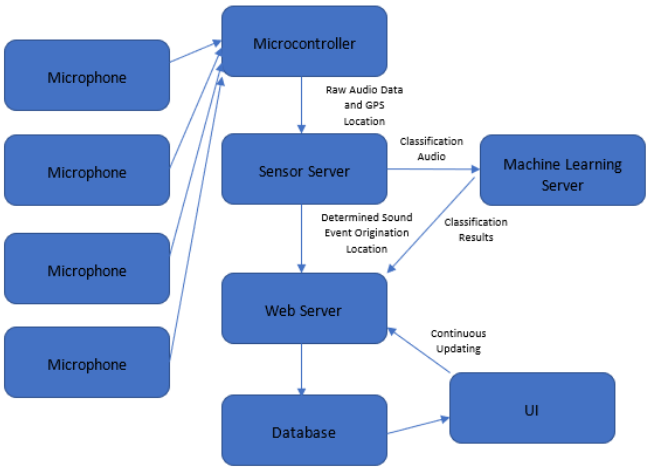

Figure II-1: Design Overview

B. Motivation

Sound plays a major role in communication and making observations about the world around us
Using technology, sound can be converted into useful information even when we are not around to physically hear it ourselves
This information can be used for many things such as: • Increasing safety and security • Research • Conservation of animals • General hobby activities

Table II-1: Motivation

C. Specifications

The array of sensors is connected to a server.
The sensor server analyzes the sound events from multiple sensors.
The sensor server will triangulate the location of the sound event using time of arrival and multilateration.
A separate machine learning server will access audio data from the sensor central server and perform classification predictions.
A web application will give the user access to all information obtained and processed by the system.

Table II-2: Specifications

D. Engineering Requirements

Identify the originating location of sound event within:	5	Meters
Real-World Classify bird calls with greater than	70%	Accuracy
Have a maximum operating distance between sensors greater than:	90	Meters
Classify a bird call within:	10	Seconds
Identify sound event location within:	10	Seconds

Table II-3: Engineering Requirements

E. Goals and Objectives

An array of audio sensors to determine the location of a sound, using a single tone
An audio sensor that can record bird calls that can be sent to be furthered analyzed with machine learning to be classified
An application to allow users to interact with the data that is retrieved and analyzed from the devices
A microcontroller connected to the sensors to analyzed sound events

Table II-4: Goals and Objectives

Figure II-1: Overall objectives for the project design.

III. Research and Theory

A. Subsystems of Machine Learning

There are three main subsystems of machine learning: supervised learning, unsupervised learning and reinforcement learning. Supervised learning provides a dataset to the model that includes labeled data, giving them input and output sets to learn from. Unsupervised learning removes the labels from and only gives the model the input data to train, the final type of is reinforcement learning. This subsystem takes the two ideas of supervised and unsupervised learning and combines these methods, where only a portion of the input data will be labeled, and the rest will be unlabeled. The classifying problem that we aimed to solve required the use of supervised learning. Though there is a great cost of time and the increase in needed computational power, the gains in the overall accuracy from using supervised learning is much greater than those costs.

B. Classification

One main facet of this project is the ability to classify sound events. This is performed with the use of machine learning. This model will be trained on preprocessed data and then perform predictions on data it has never seen before. For this model, a Convolutional Neural Network was chosen. A general structure of a CNN involves several layers stacked together in a deep architecture

1) An input layer
2) Group of convolutional and pooling layers
3) A limited amount of fully connected hidden layers
4) An output (loss) layer

An Activation function is then used to introduce non-linear properties into the network. ReLu being one of the most popular functions, and our problem being a multi-class classification problem was used along with a SoftMax function for the final output.

Deep neural networks in general are highly susceptible to overfitting, where estimated values become much greater than the training cases. This in turn can cause a high accuracy on the training data, but a substantial decrease in accuracy when making predictions on new data presented to the system. Dropout is used to help combat this by randomly selecting connections and killing them, resulting in a decrease in total connections.

Optimization is then used to compile the model. Optimization defines a loss/cost function and then minimizes it using a specified algorithm. The function that was determined to be the best for our problem was the Adam optimization function as it is known to perform well with complex data features. The sparse_categorical_crossentropy loss function was chosen as are problem involves the classification between multiple classes and our class labels are presented as integers.

C. User Interface

The idea for our UI is to utilize a web application. This web application encompasses the user's ability to view and interact with the data collected by the sensors, provided in an easy-to-understand format. The main goal for this interface is to develop a robust and user-friendly interface. Through the comparison of different web stacks, the MEAN stack was chosen for its single page ideology and its functionality and speed from being completely built with JavaScript.

D. Machine Learning Server

There are several functionalities needed for this project that must be performed out of sight from the user and limit the computational strain on the sensor hardware. Due to the nature of the machine learning being computational and resource heavy, a separate server should be implemented to handle the model predictions. Flask, which is a web container written in python was chosen since the machine learning models were also implemented with python, providing a seamless integration of the model onto the server.

E. Multilateration

Multilateration is a technique to determine the location of a subject based on the time of arrival (TOA) of multiple signals that have a known propagation speed. This technique can be used for either navigation or surveillance. If used for navigation, a subject receives signals from multiple known

stations that are synchronized. If used for surveillance, the stations receive a signal from the entity to determine its location based on the relative arrival time at each station. A popular use case of multilateration used for navigation is GPS, and a couple use cases for surveillance include Seismic Event Location, and sound ranging. For this project we will be 44using multilateration in a surveillance configuration to determine the location of a sound source. There are two kinds of multilateration:

a. Determine the time of transmission (TOT) of synchronized signals being emitted by n + 1 stations with known locations. This is used for navigation.
b. Create n time difference of arrivals (TDOAs) of a signal at n + 1 stations to determine n coordinates. This is used for surveillance. As this is a surveillance system, TDOAs must be used because the TOT of the sound is not initially known. Assuming the propagation speed of the signal and the location of each station are known, using one TDOA, a hyperbolic curve of possible locations for the source can be created. Another TDOA can be used to create a second hyperbolic curve, and the points where the two curves intersect are the sources possible location. It may make sense to use a GPS module on the microcontroller in order to maintain up-to-date sensor location information automatically. This would enable the sensors to be mobile and they would require less setup. Sensors could be distributed over an area without keeping track of where each one is. For example, this could be used to deploy a sound event location system over an area by dropping sensors from an airplane. GPS can consume a decent amount of power, and this needs to be understood when choosing the rest of the hardware.

F. Lithium-Ion Battery

Lithium-ion batteries work by passing lithium ions from the anode through the electrolyte and to the cathode. Lithium-ion batteries have several advantages such as a high charge storage per unit mass and a high voltage of 3.7 volts. This means that they can deliver large amounts of current for high power applications. Lithium-ion batteries have no memory effect, meaning that repeated partial discharge of the battery will not reduce the overall capacity of the battery. In addition to this, lithium-ion batteries have a low self-discharge rate of about 1.5% per month. [“Lithium-Ion Battery”] So, when the batteries are not in use, they will hold their charge. Lithium-ion batteries are also very reliable. They require little maintenance and can withstand anywhere from 300 to 500 charge cycles. [“BU-808: How to Prolong”] The main disadvantage of lithium-ion batteries is its safety. The batteries are subject to overheating and sometimes combustion. Another disadvantage is that, like most batteries, they start to fail with age.

G. Solar Panels

Solar panels are made up of different semiconducting materials that are combined to form a photovoltaic cell. A photovoltaic cell consists of an n-type and a p-type semiconductor that create a p-n junction. When light is shines on the p-n junction, the energy from a photon is transferred to an electron. This transfer of energy causes the electron to move from the valance band to the conduction band, creating electricity. This is known as the photovoltaic effect. Photovoltaic cells can only convert about 25% of sunlight into electricity. This is because sunlight consists of many photons, each with various energies. For an electron to jump from the valance band to the conduction band, it needs to receive a specific amount of energy known as the band gap energy. For silicone, the band gap energy is 1.2 electron volts. If a photon has energy less than 1.2 electron volts and it hits an electron, the electron will remain in the valence band and the energy will be absorbed as heat. If a photon has an energy greater than 1.2 electron volts and it hits an electron, the electron will move to the conduction band. However, the excess energy will be absorbed as heat.

Monocrystalline solar cells are made from a single, pure crystal of silicone. In order to achieve this, a solid piece of silicone is cut into wafers. This process causes a lot of waste material. Since the photovoltaic cells are made of a pure silicone crystal, monocrystalline solar panels produce some of the highest efficiencies compared to other types of solar panels. Monocrystalline solar panels have efficiencies of around 20%. Monocrystalline solar cells also appear black because of the way light interacts with the pure silicone crystal. The main advantage of monocrystalline solar panels is their high efficiency. Due to this efficiency, monocrystalline solar panels are preferred when there is not a lot of space. However, achieving such a high efficiency requires an expensive manufacturing process with a lot of waste material.

Polycrystalline solar cells are made from fragments of silicone crystals that are melted together in a mold. This manufacturing process is inexpensive and there is no waste material. Polycrystalline solar panels are blue in color because of the way the different silicone fragments reflect light. Because of the color and the silicone fragments, polycrystalline solar panels are said to be visually unappealing. Polycrystalline solar panels are average when it comes to efficiency. They offer an efficiency of around 15%. Polycrystalline solar panels offer a balance between efficiency and cost.

Thin-film solar panels are made from a variety of materials. The most common material is cadmium telluride (CdTe). In order to make the solar cell, a layer of cadmium telluride is placed between transparent conducting layers. Another material commonly used is amorphous silicon (a-Si). This is a non-crystalline silicone and the cells are not made up of a solid wafer. The last material that can be used is copper indium gallium selenide (CIGS). Thin-film solar panels have a very low efficiency of around 11%. This means that many panels would have to be used to achieve and

desirable energy output. Due to the manufacturing process and depending on what type of material is used, thin-film solar panels are relatively inexpensive. They are also, as the name describes, reality thin. This means that thin-film solar panels are small, lightweight, and flexible.

IV. Hardware Design

This section details many of the different aspects of the hardware design process for this project, including the primary system components and the overall PCB design.

A. Microcontroller

The most important hardware design choice was in selecting the microcontroller. For this project, a MK20DX256VLH7 Cortex-M4 processor was used, which can run at a clock speed of either 72 MHz or 96 MHz. For communication, this ARM processor comes equipped with a USB OTG module, 3 UART modules, 1 SPI module and an I2S module all of which are being utilized by the different components of this project.

The USB module is being used to program the microcontroller and to optionally power the PCB. Serial1 is being used to program the ESP32 from the microcontroller and Serial2 is being used to connect to the GPS module. The I2S module is being used to receive audio data from the audio shield. The SPI module is connected to both the audio module to potentially utilize the onboard SD card and to the ESP32 to send the audio and GPS data to it for transmission to the main server.

This processor was selected primarily due to its compatibility with Teensyduino software, which is a modified version of basic Arduino that allows the use of the Teensy Audio libraries. A bootloader is connected to the Cortex-M4 processor that enables the processor to use the Teensyduino firmware and libraries.

B. Audio Module

In order to ensure compatibility with the Teensyduino software libraries that are available, the Teensy Audio Shield Rev C was used to record, process and store the incoming audio data.

This audio shield is a powerful device that operates at 3.3 V and has audio recording capabilities at 16-bit with a sample rate of 44.1 kHz. This will allow the sensor to capture high-quality audio data that can later be used for processing. This shield communicates with the microcontroller using a combination of protocols: I2S for audio data, I2C for audio control, and SPI for connection to the optional SD card slot.

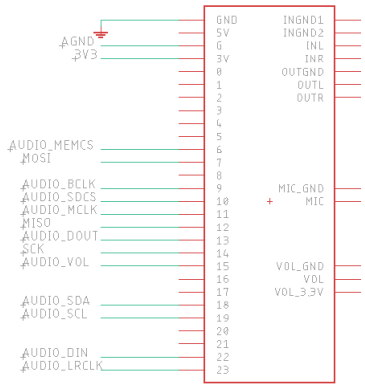

Figure IV-1: Schematic showing the connections from the Audio Shield to the MCU.

C. GPS Module

The GT-U7 GPS module that is used on the circuit board utilizes a NEO-6M GPS receiver. The NEO-6M GPS receiver has a tracking sensitivity of -161 dBm and a time-to-first-fix of approximately 27 seconds. This module communicates with the microcontroller through UART with a default baud rate of 9600. On its TX pin, it sends the GPS coordinate, date and time data to the microcontroller. On its RX pin, it receives programming information from the microcontroller to configure the PPS.

D. Wi-Fi Module

The Wi-Fi chip used for this project was the ESP32-WROOM-32D. Next, to the main microcontroller, the Wi-Fi module is the next most important component to the project as it receives all the audio and GPS data from the microcontroller and transmits them to the main server for analysis.

As for the chip's networking capabilities, the ESP32 utilizes 802.11 b/g/n, has a maximum data rate of 150 Mb/s and an output of 20 dBm.

The ESP32 communicates with the microcontroller through both UART and SPI. The UART connection is used to program the ESP32 and the SPI connection is used to send data from the microcontroller to the ESP32.

While the ESP32 does operate using 3.3V like the other components of the board, it requires more power than what the microcontroller can output. Due to this, a separate LDO voltage regulator is used just for the ESP32 module that is connected to the main 5V line and outputs 3.3V with a much higher output current.

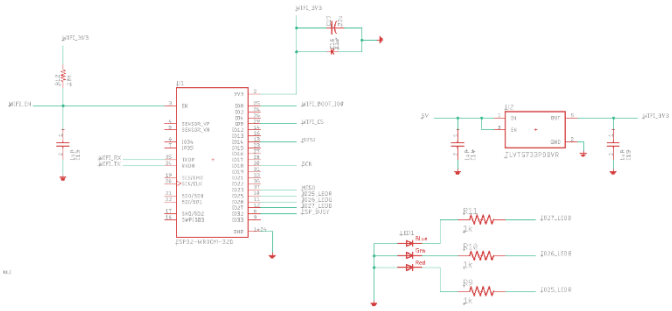

Figure IV-2: Shematic of the ESP32-WROOM-32D with connections to the MCU

.

E. Battery/Solar Power System

DiSEL is equipped with a battery and solar power system. Each PCB is powered using two 3.7 volt, Lithium-Ion, 18650 batteries. Each battery is charged using a TP4056 IC, which is designed specifically for charging Lithium-Ion batteries. Although Lithium-Ion batteries are rated at 3.7 volt, they can actually be charged to maximum of 4.2 volts. All charging circuits take advantage of this fact, and the TP4056 is no different. The TP4056 uses 4.2 volts to charge its Lithium-Ion batteries.

The proper method of charging multiple batteries at once requires a Battery Management System (BMS). When charging a battery, it is dangerous to continue charging it over its operating voltage. A BMS detects when the battery has been fully charged and uses a MOSFET to cut power to the battery. In the case of DiSEL, the battery pack is charged with a solar panel. Without overcharge protection, if the battery pack is fully charged and the solar panel is in direct sunlight, then the battery pack is being overcharged and could be destroyed or possibly combust. The opposite is also true. If there is no sunlight and DiSEL is still consuming power, then the battery pack is at risk of over discharging. This could ruin the battery pack and can cause it to no longer hold a charge.

To add protection for the batteries, a DW01A IC is used in the TP4056 charging circuit. The DW01A IC offers two main features: over-charge protection, over-discharge protection. If the battery voltage goes above 4.3 volts, the DW01A cuts off power to the battery. Once the battery voltage drops below 4.1 volts, the DW01A will restore power to the battery. If the battery voltage drops below 2.5 volts, the DW01A cuts off the battery from the output. The DW01A reconnects the battery to the output once the battery voltage goes above 3 volts.

For DiSEL, two TP4056 charging circuits were used to create a BMS. This allows each battery to be individually monitored, charged, and discharged, ensuring proper cell balancing. The outputs of the TP4056 charging circuits are connected together in series providing a maximum voltage of 8.4 volts.

Since the two TP4056 charging circuits are connected in series at their outputs, the inputs cannot share a ground. If both TP4056 charging circuits are powered by the same voltage source, one charging circuit will get shorted out and bypassed. In order to solve this problem there are two solutions: Use two different power supplies (two solar panels), or isolate the grounds from each other.

For this project, a B0505S-1W was used to isolate the grounds of the charging circuits. The B0505S-1W is a DC-DC converter with isolation. It takes an input of 5 volts and outputs 5 volts. The B0505S-1W is able to isolate the input from the output by using a transformer.

The 18650 Lithium-Ion battery was chosen due to its popularity. The 18650 battery can be found in flashlights, laptops, and power tools. Due to its popularity, the 18560 is well documented, affordable, and has been proven reliable.

The TP4056 charging circuit was also chosen due to its popularity. Many Lithium-Ion projects use the TP4056 with great success. The TP4056 is also designed specifically to charge Lithium-Ion batteries. Not only does it charge the battery, but with the DW01A it also monitors the battery as it discharges. This offers significant protection in a tiny, affordable package.

The solar panel used in this project is a monocrystalline solar panel. This solar panel was chosen because monocrystalline solar panels are the most efficient at producing electricity from sunlight. This means they can produce more electricity while maintaining a smaller footprint. Monocrystalline solar panels also work better in indirect sunlight or cloudy weather.

The solar panel has a built in 5 volt regulator that outputs 5 volts at 2 amps. This output from the solar panel is connected to one charging circuit and the B0505S-1W. From here each circuit powers its own battery and sends the monitored battery output to the PCB.

F. Printed Circuit Board

Initially, the hardware portion of the project was tested using a Teensy 3.2 development board. As with the audio module, the Teensy 3.2 was selected to be the development due to its native compatibility with the Teensyduino libraries.

Once initial testing with the development was completed and proven to be effective and operational, a printed circuit board design was started to finalize and clean up the hardware portion of the project.

The DiSEL printed circuit board (PCB) was designed using Autodesk's EAGLE software. While EAGLE did not come initially equipped with many of the component libraries that were needed for this project, its many tools and the team's experience with the software proved beneficial.

The PCB has two optional sources of power: a USB port or a plug-in port for a battery that is connected to a DC-DC converter to keep the output voltage from the battery at 5V. The two power sources connect to a small schottky diode circuit that is used to select which power source to draw current from. From here, the 5V voltage from the USB or the battery is sent to an LDO voltage regulator to bring the voltage down further to 3.3V, which will be used power the microcontroller.

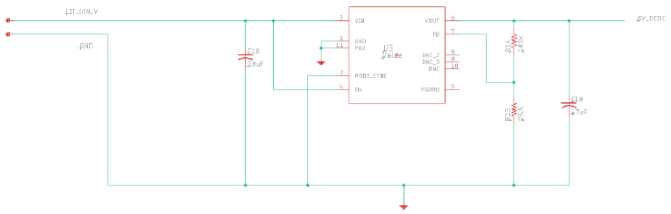

Figure IV-3: Schematic of the DC-DC converter circuit that uses a LMZM236000SILR Step-Down Module to output 5V.

Since three boards were needed for this project, keeping PCB costs to a minimum were crucial. The size of a board accounts for the primary cost of a PCB; with this in mind, the overall PCB size was kept as small as possible, with the final design coming out to 3.80 x 2.59 inches.

Once fully designed, the PCB files were sent to OshPark to be manufactured. OshPark was selected to manufacture the PCBs due to their reputation as a high-quality PCB manufacturer while also maintaining a low price point. Additionally, OshPark also has some plug-ins for EAGLE that allow for easily adjusting the board settings to be compatible with OshPark's manufacturing process and for easy uploading of the board files.

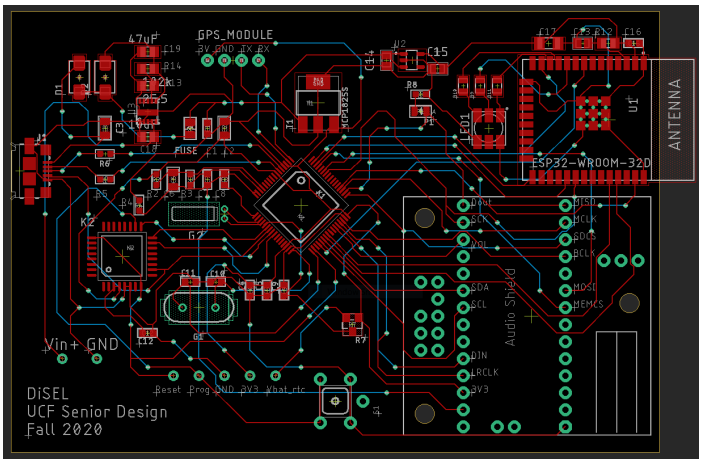

Figure IV-4: DiSEL PCB board layout

V. Hardware Testing

Once the three PCBs were manufactured and all of the components were soldered onto the board, each board was tested in various was to ensure that every section of each board were working as intended.

A. Microcontroller Test

The initial testing for the microcontroller involved using a simple Teensyduino code that would blink the LED connected to the MCU on the PCB. This small piece of code served to test the connections, implementation and components of three different aspects of the board: the microcontroller itself, the bootloader used to upload the firmware needed to run Teensyduino, and the micro-USB port that is used for powering and uploading code to the MCU. After uploading the blinking code, the onboard LED began blinking, showing that the basic MCU and bootloader parts and connections were working as intended.

Another test for the microcontroller was testing the dual Schottky diode circuit at the power section of the board. The goal of this circuit was to allow the board to easily utilize and switch between both USB power and power coming from the battery/DC-DC converter section of the board. This was done by testing whether the PCB worked properly with each power source individually connected and then testing if the PCB worked properly with both power sources connected at the same time.

B. Audio Shield Test

After ensuring that the MCU, bootloader and micro-USB port were working, the next section to test was the audio shield module. This test was done by running two pieces of audio shield example Teensyduino codes. The first test program that was run was a simple "beeping" program that would playback a beeping tone to the user. Once the beeping program proved to be successful, a more advanced test was run which was an audio recorder program that would record a piece of audio from the user, save the data to the onboard SD card and then play the audio back to the user. Once this program ran successfully, we were confident that the audio shield was connected to the MCU properly and the component was working as intended.

C. GPS Module Test

Initial testing of the GPS module without configuring the PPS was rather simple. Once the GPS module ran for long enough to achieve a satellite fix, a simple program was used to read the GPS data through the serial monitor. The received GPS date and time were checked with the current date and time, and the GPS coordinates were inputted into Google Maps to check for accuracy.

D. Wi-Fi Module Testing

Once the data collecting portions of the project were tested and proven to be functional, the next step was to test the ESP32 which is used to send the data over Wi-Fi to the main server.

The microcontroller is intended to send a constant stream of GPS and audio data that are broken up into smaller packets to the ESP32 through SPI. So initial testing for the SPI connection was to send just one set of coordinates and audio data. After configuring the SPI code, the data was sent from the microcontroller to the ESP32 and the serial monitor for both the microcontroller and the ESP32 were opened to see if the data matched. Once the data displayed on both serial monitors finally matched, the SPI transfer was shown to be operating as intended.

The next test for ESP32 was to make sure its Wi-Fi capabilities were fully functional. This began by running a program to scan for active Wi-Fi networks in the area and to

display their names on the serial monitor. Once networks were found, a hotspot was then setup for the ESP32 to try to connect. The ESP32 was successful in both attempts and proved to be working properly so far. Once the main server for the project was functioning, the final test for the ESP32 was to connect to the server and transmit a message to it. After the communication between the ESP32 and the server was established and the message received by the server was accurate, the ESP32 tests were finished.

VI. Software Design

A. CNN Implementation

Our CNN was implemented and trained with the TensorFlow Keras framework. To train the model, the training data required normalization and preprocessing. The bird call data was first pulled from Xeno-Canto, a crowd sourced bird call database and API with the recordings that have creative commons licenses to allow for their use. To normalize the data, twenty-nine bird species were found that contained a minimum of one hundred audio samples that Xeno-Canto considered have A or B quality. One hundred of those species where then extracted to establish an evenly distributed dataset. These samples where normalized further by only loading the first thirty seconds of each sample to then normalize the length of each one. To increase the number of samples to allow for better training, the *2900* samples where split into six equal segments of five seconds each, resulting in a total of *17,400* samples to train the model after all the normalization and preprocessing where completed. This data was then loaded with the use of librosa, a python package for music and audio analysis. Each sample then had a Mel Frequency Cepstral Coefficients (MFCC) features extracted from the audio.

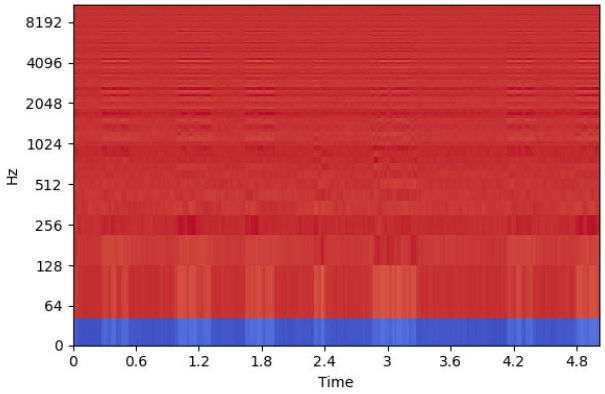

Figure V-1: MFCC

This dataset of labeled MFCC features, was then broken into and 80/20 split to be used for training and testing data. The training data was split further into an 80/20 ratio for the use of training and validation. Using supervised learning, the data was passed through the developed network shown below to train. Once the training of the network was completed, the model was saved as a .h5 file, which saves the models weights, and architecture, to allow for reloading and using the network without the need for retraining. To use the trained model within the Disel system, a Flask server was created to run the model and make predictions, reducing the computational strain on the sensors.

Conv 32
Max Pool
Conv 64
Max Pool
Dropout (.1)
Conv 64
Max Pool
Conv 128
Max Pool
Dropout (.1)
Conv 128
Max Pool
Conv 128
Max Pool
Dropout (.1)
Conv 256
Max Pool
Flatten
Dense 128
Dropout (.4)
Dense 64
Dropout (.4)
Output (29)

Figure V-2: Simplified CNN Build View

B. User Interface Implementation

When implementing the web interface, there are two main components. The login/signup component which is presented first to the user. If they are a returning user, they can simply sign in with their username and password. The form with the entered credentials will then be passed to the web server which will query, and the database will be queried first to see if there is a user with the entered username. If this is returned true, the entered password will then be compared with the hashed password stored in the database that is associated with the found user. If this comparison returns true, then the user is sent to the dashboard. If the user is new, they can complete the signup form, and then continue as a returning user. The second component of the user interface is the dashboard. This component contains all the data captured and processed by the DiSEL system, in an easy to view and interact format. Within this component there contains two sub sections, the location and classification sections. The location section provides an the user with all the location data processed by the system, along with a Google Map shows the current locations of the DiSEL system's sensors, along with the ability for the user to add, and remove markers, showing a visual representation of the latitude and longitude for each of the location sound events. The classification section the user to view all the sound events that are associated with the classification of bird species. Each of these cards contain the classified bird species, along with the time and date of the classification. Both sections are updated every thirty seconds, and the sensor locations are updated every ten minutes.

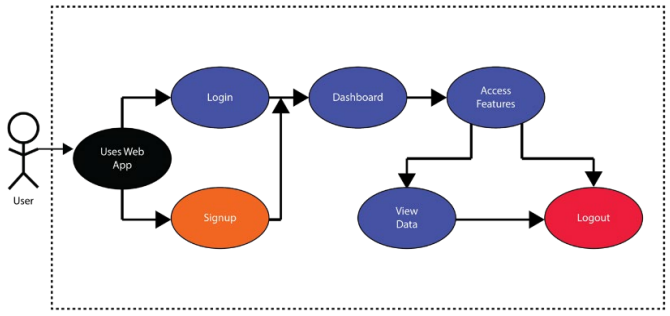

Figure V-3: Use Case

C. Backend Implementation

To process and relay of the data in the different ways necessary for this system, requires a set of different backend applications. For the web server to host the website and perform the database accessing tasks, Nodejs was used. Since Nodejs along with express are part of the MEAN stack ecosystem, the ability to seamlessly transition from the user interface to the database is quite simple. To implement a solution of performing classifications on audio data and being able to relay them to the web server, a Flask container was used, since it is designed in python. To finish the implementation of the backend was to establish a server that could accept the data from the sensors, perform the tasks of multilateration, along with sending the results to the web server, and the classification audio to the machine learning server. To accomplish this, a server written in rust was implemented, with the use of very useful libraries written specifically for helping to solve the equations and algorithms required to perform multilateration.

VII. CONCLUSION

The original goal of this project was to develop a system that could utilize information analyzed from incoming sounds in a variety of different applications. Being able to classify and locate the origin of a particular sound was the clear winner in this front from this information being useful in research, safety, and hobbyist applications.

The solutions used to achieve the project goals allow the DiSEL network to be highly scalable, with a user being able to add additional sensors to the network for larger areas or with the ease of teaching the machine learning software new sounds to be able to classify.

VIII. BIOGRAPHIES

Brandon LaGuerre will be graduating from UCF in December of 2020 with a Bachelor of Science in Electrical Engineering. After graduating, he will transition to a full-time position at L3Harris and in the near future will pursue a Master's degree.

Mark Judy will be graduating from UCF in December of 2020 with a Bachelor of Science in Computer Engineering. After graduating, he is going to transition to a full-time position at IBM.

Chris Santana will be graduating from UCF in December of 2020 with a Bachelor of Science in Electrical Engineering. After graduation, he is going to further pursue his interests in power engineering.

Drew King will be graduating from UCF in December of 2020 with a Bachelor of Science in Computer Engineering. After graduating, his is going to further pursue his interests in machine learning.