

**Woof**

**Team 2**

Nafis Abeer [nafis@bu.edu](mailto:nafis@bu.edu)

Justin Lam [jlam8080@bu.edu](mailto:jlam8080@bu.edu)

Chase Maival [chase1@bu.edu](mailto:chase1@bu.edu)

Rajiv Ramroop [raj19@bu.edu](mailto:raj19@bu.edu)

Daniel Shimon [dshimon@bu.edu](mailto:dshimon@bu.edu)

**Required Materials:**

Software:

* React Native + JavaScript Project for User Interface
  + Expo for compiling code to show on Physical Phone
* MapBox SDK integration with React Native for Xcode
  + Xcode for running the iOS application on either a...
    - Simulator for any iOS software
    - Physical iPhone
  + MapBox account to access API token
* Arduino IDE for visualizing accelerometer data and inferences
* Python Notebook
  + Audio classification for dog bark detection
    - Librosa for gathering and normalizing Audio channels, Sample rate and Bit depth. MFCC for audio spectrogram to run CNN
    - Tensorflow, Keras, Numpy and Pandas
    - UltraSound8K database for retraining. Weights have previously been saved

Hardware:

* Arduino Uno
  + Adafruit Bluefruit LE UART Friend - Bluetooth Low Energy (BLE)
* ESP32 Microcontroller
  + Accelerometer - HiLetgo GY-521 MPU-6050 MPU6050

**Set Up:**

The setup has 3 components.

One component is to train a local model to detect if a certain audio file is of a dog barking.

* A computer acts as a makeshift server until we deploy this model on an actual server. It needs to come preinstalled with Anaconda and Jupyter. The UrbanSound8K dataset from 2014 should be installed. The model should be pretrained on this dataset. Post training, we will download audio files from arbitrary online sources and test with models to display effectiveness.

Second component is to display MapBox API working on an Xcode project, which will work on an iPhone, and UI working on EXPO. This UI on EXPO is integrated with firebase.

* This setup materials for the MapBox API include running a bare react native project, and furthermore integrating the API to work with this bare project, as EXPO integration is not yet allowed with MapBox.
* The User Interface will be displayed on a separate phone, with Email SignUp/LogIn capability that connects to Firebase.

Third component is to display accelerometer data on ESP32 and BLE detection from a phone.

* The setup materials include the following hardware components: an Arduino Uno, an ESP32 microcontroller, an accelerometer, a BLE sensor, and a microphone.
  + The BLE setup should display signal strength, in RSSI (Received Strength Signal Indicator). The closer this value is to 0, the stronger the signal strength is, which means the closer the object of interest is to the BLE. The closer this value is to -100, the weaker the strength, and the farther the object of interest is to the BLE. In our case, a value of -50 or greater indicates that the dog (represented by our phone, in this prototype), is close enough to the other dog (represented by the BLE, in this prototype), in which case we can start collecting and processing data. The BLE was wired and powered by the Arduino UNO, as depicted in the Hardware pinout below.
  + The accelerometer data is collected by a 3 axis accelerometer, which measures the speed in meters per second squared in the x, y and z directions. The accelerometer data currently can detect if the dog is running or sitting. This was wired and powered by the ESP32, as depicted in the Hardware Pinout below.

**Pre-testing Setup Procedure:**

* Run notebook.py on Nafis’s Macbook. It will train the model to classify audio files
* Run commands to get maven running, and furthermore build and deploy the bare react native project to show the API implementation, as well as User Location, is included and running.
* Run npm start for Expo UI to demonstrate working SignUp/Login capabilities
* Plug in accelerometer according to the Hardware pinouts below, then upload code serial\_test\_accelerometer to Arduino IDE. Compile the code, then upload it. Then, click on Tools ----> Serial Monitor to view the live results.
* Plug in BLE according to the Hardware pinouts below, then download the Bluefruit Connect Application on your mobile device, or any similar app. Then, connect your mobile device to the BLE.

**Measurable Criteria:**

Model:

* Download audio files of dog barking from the internet. We will be able to display the model working on these files. The ratio of number of files correctly identified vs. incorrect classifications are our metrics for testing accuracy of this model.

Hardware:

* We will display the accelerometer data in meters per second squared for the x axis, y axis, and z axis.
* We will display the BLE data from the Bluefruit connect application which outputs the RSSI.

Software:

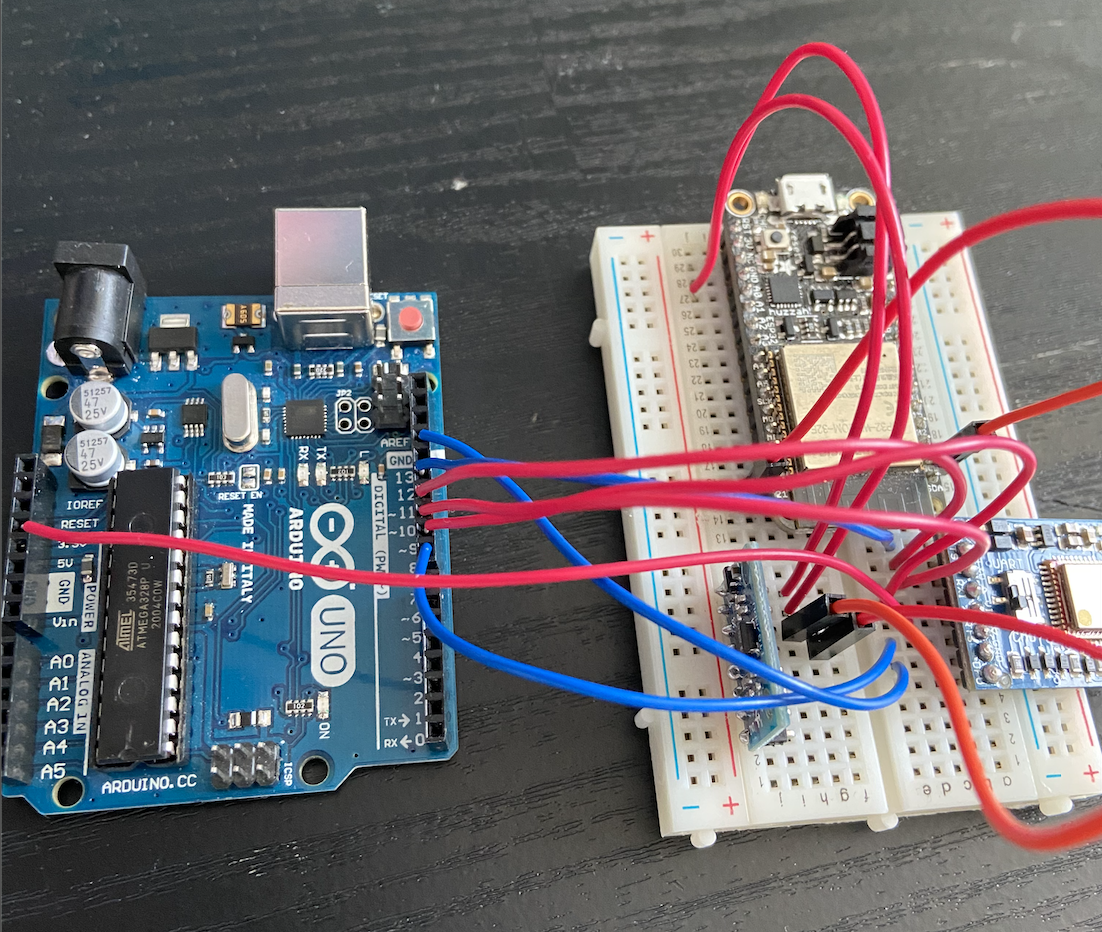
* We are able to display a working UI with SignUp/Login capabilities
* MapBox will be up and displaying the phone's current location. Along with examples of how drop-down pins should work.

**Hardware Pinout (Arduino UNO to Adafruit BLE :**

| PIN12 | MOD |
| --- | --- |
| Pin 11 | CTS |
| Pin 10 | TXO |
| Pin 9 | RXI |
| 5V | VIN |
| Pin 8 | RTS |
| GND | GND |

**Hardware Pinout: ESP32 to MPU6050**

| 3V | VCC |
| --- | --- |
| GND | GND |
| 27 | SCL |
| 21 | SDA |



**In-Lab Testing:**

Our setup was successful for each component. We had a model running on a Jupyter notebook, and it was trained just before entering the lab. We had the BLE and Accelerometer pre-wired to the Arduino and ESP32. These devices accurately displayed real time data on the console. We had our EXPO project running to display the UI component and the instructor was able to see our various navigation screens. We also had our mapbox UI up and running, and it accurately displayed the phone’s current location.

A lot of testing is more about displaying functionality of components rather than metric data. Our focus for the prototype was to identify and familiarize ourselves with each required component of the project. These components are displayed to be working in its preliminary stages.

The model which was meant to be at its preliminary stage of identifying noises and classifying it into distinct categories performed perfectly on every type of audio file.

* We downloaded 3 arbitrary files off the internet to display that the model works on any kind of audio file despite the source.
* We tested the model on Dog-Barks as well as other classifiers such as “gun-shots” and “children playing” and it accurately identified all of them

The accelerometer data was accurately measured in real time. We had hard-coded conditions set-up to simulate how acceleration data could be an indicator of real-time movement in dogs.

if(abs(X)>abs(Y) && abs(X)>abs(Z) && abs(Y)<1 && abs(Z) < 1){

Serial.print(" Dog is Sitting ");Serial.print(" \n ");

}

if((abs(Y)>2 || abs(Z)>2) ) {

Serial.print("Dog is Running"); Serial.print(" \n ");

}

if(abs(X)>10 && abs(Y)<3 && abs(Z) < 3){

Serial.print("Dog is Jumping"); Serial.print("\n");

}

* When the accelerometer measured the absolute value of x (abs x) to be greater than the absolute value of y (abs y) , and abs x was greater than the absolute value of z (abs z), and abs y was less than 1(m/s^2) and abs z was less than 1, we concluded that based on the movements of the sensors, the dog is predicted to be sitting. The dog was predicted to be running if abs y was greater than 2 or abs z was greater than 2. Finally, the dog was predicted to be jumping if abs x was greater than 10 and abs y was less than 3 and abs Z was less than three.
* G force was converted to m/s^2 through dividing X, Y, and Z data by the common factor 1692 to use gravity at 9.81 m/s^2
* The BLE range was also displayed using the Bluefruit Connect app. The range of RSSI is from 0 to -100, with 0 indicating a strong signal and -100 indicating a weak signal. We found that based on our estimates, a range of -30 to -55 is an appropriate range for dogs to be interacting in, and thus that is the range that will be used as an indicator for when our sensors start collecting data to feed the recommender system.

The UI was set up to display signup and login screens through React’s Navigation container to have multiple view windows. Each user id was logged to Firebase, and the login will be integrated with Google Cloud’s client IDs for Android/IOS third-party Google authentication support with the Spark plan -- which supports authenticating 10,000 users per month.

The Mapbox component accurately displayed user location (and pins add locations). This was used to show that we are able to integrate with the MapBox SDK, which was a concern before as we are developing our application using React Native. MapBox allows us to have 20,000 users locations before requiring to be a part of a paying subscription plan, which is more than enough for our needs. Furthermore, the idea behind showing user location was for one of the features of the social media app in the future, where we would be able to display user, dog’s, as well as friends locations all on an interactive map.

**Results/Conclusions:**

1. The performance of the model indicates that we can easily train neural networks to differentiate between distinct noises so long as we have a consistent dataset. We can normalize audio files to have a unified sample rate, bit depth and audio channel and it performs well on identifying noises by turning it into a sort of spectrogram. This finding is important because we can now train our model on just audio files of dogs barking where the labels are classifying the type of bark. For the future, we will not only be able to detect dog barks from incoming audio streams, but also classify the dog-bark into friendly vs. Hostile
2. Having an accurate UI running was essential to have a centralized source to display our progress. We can now focus on the social media component and adding friends since we can authenticate individual users. We can now also explore ways to add instant messaging and add friend functionalities such that authenticated users can interact within the application.
3. The Mapbox UI being used to display accurate location information for user’s current phones. In the future we will for one integrate the dog’s location on the map so that the user will always be able to track their dog’s location. Furthermore the user will be able to see where their friend’s dogs are as well, with permission of course. This is all with the plan of making this application as user friendly as possible, allowing owners to interact with other dog owners, as well as keep track of their dog. Another small feature being developed is allowing users to drop pins for their favorite locations, and interact with these locations as well, which led to the idea of being able to show this functionality during the demo.
4. The sensors working will now allow us to send RSSI and accelerometer data through the serial lines to print the real-time signal strength or acceleration. As a part of the dog-tag we can use this data in the future to make predictions and recommendations based on where the dog has been, and based on its behavior in some distinct locations.