Drone Detector

Senior Design II (CMPE 4374-01)

The University of Texas Rio Grande Valley

College of Engineering and Computer Science

Team Members:

Audrey Sánchez

Edgar Martinez

Jaime Rivera

Advisors:

Dr. Laura Benitez _____

Dr. Junfei Li _____

Dr. Jae Sok Son _____

MAY 2020

Abstract (Non-Technical)

_____The objective of this project is to detect a drone in any house or private property. As long as the device is installed in the desired area, an alert will be sent to the proprietor's

mobile phone when a drone is hovering over the area. The main problem regarding the drones is the fact that it is only strictly illegal for them to fly over airports and military bases, but legal for them to invade a person's property whether it’s being used for recreational or malicious purposes. With the drone detector the proprietor can be out of town and receive an alert to take safe actions as calling the police if desired.

Abstract (Technical)

The objective of this project arose from the need to preserve privacy from the presence of a drone on private property. The problem was not having the necessary elements to take action against a drone even if the drone invades a person’s property. A drone detector, which alerts the proprietor when there is a presence of a drone within a certain range. The device is designed to detect a drone by means of a microphone and a camera controlled by a Raspberry Pi through different coding programs. It captures the sound emitted by a drone and by doing this it activates a camera which takes a photo that is determined through a program if the objective is a drone or a bird (flying object). After these steps, it alerts the proprietor if there is a drone within the desired area and thus achieves the objective of this project.

Table of Contents

Abstract (Non-Technical) **1**

Abstract (Technical) **1**

Table of Contents **2**

List of Figures and Tables **3**

1.0 Introduction **7**

2.0 Problem Formulation **10**
- 2.1 Need Statement 10
- 2.2 Objective Statement 11
- 2.3 Background 11

3.0 Design Plan / Structure **13**
- 3.1 System Proposed Approach 13
- 3.2 System Specification 15
  - 3.2.1 Camera 16
  - 3.2.2 Microphone 16
  - 3.2.3 Software 17
  - 3.2.4 Processor 18
- 3.3 Hardware and Software Requirements 18
  - 3.3.1 Camera 60 FPS 18
  - 3.3.2 Filter Sound from Microphone 18
  - 3.3.3 Training and Testing CNN Model 19
- 3.4 Schedule 19
  - 3.4.1 Senior Design I Gantt Chart 20
  - 3.4.2 Senior Design II Gantt Chart 21
- 3.5 Cost 22
  - 3.5.1 Table Of Component Prices 22
    - 3.5.1.1 Items 22

4.0 Results **23**
- 4.1 Drone Sound Test using Smartphone Microphone 23
- 4.2 Camera 24
  - 4.4.1 ArduCam 2MP oV2640 Mini Module SPI 24
  - 4.4.2 Raspberry Pi Mini Camera 0V5647 29
- 4.3 CNN Model 33
- 4.4 Microphone 48
- 4.5 Drone Detector Cover 51

5.0 Equipment / Fabrication / Software Needs **54**
- 5.1 Cameras 54
  - 5.3.2.1 ArduCam 2MP oV2640 Mini Module SPI[17] 54
  - 5.3.2.2 Raspberry Pi Mini Camera 0V5647[18] 55
- 5.2 Electret Condenser Microphone EM-926[19] 56

5.3 Arduino and Raspberry Pi 57
5.5.1 Arduino Uno[20] 57
5.5.2 Raspberry Pi Model 3B[12] 58
5.5.3 Raspberry Pi 4 Model B[21] 59

6.0 ABET Criteria **61**
6.1 Economic 61
6.2 Environmental 62
6.3 Ethical 64
6.5 Political 69

7.0 Conclusions and Recommendations **71**

8.0 References **74**

Appendices **77**
Arduino Camera Code from the source[4] 77
Train CNN Model Code [30] 92
Test Drone Detector Code[31] 96

List of Figures and Tables

Figure 3.1.1 Block Diagram 13

Figure 3.2.1 Flow Chart 15

Figure 3.4.1.1 Senior Design I Gantt Chart 20

Figure 3.4.2.1 Senior Design II Gantt Chart 21

Table 3.5.1.1.1 Component Prices 22

Figure 4.1.1 Line Chart Sound(dB) vs Distance(ft) 23

FIgure 4.1.2 Close Drone Sound Measurement 24

Figure 4.1.3 Far Drone Sound Measurement 24

Figure 4.2.1.1 Photo taken of the computer processing arducam. This shows the camera taking a photo of the computer screen. 25

Figure 4.2.1.2 Corridor taken with ArduCam. This shows a test subject at the end of the hall and how it can still be recognised. 27

Figure 4.2.1.3 Corridor with person in middle. Shows how the person is getting closer to the camera and how it can be distinguished better. 27

Figure 4.2.1.4 Corridor with a person moving using ArduCam. This shows a moving subject at a short range. 28

Figure 4.2.1.5 ArduCam focuses on an object. The object is not too far away and it can be seen clearly. 29

Figure 4.2.1.6. Photo 6. ArduCam closer focus on object. This is a close water bottle. 29

Figure 4.2.2.1 Field of View of Camera 30

Figure 4.2.2.2 Measure of Camera View 31

Table 4.2.2.3 Measurements of Camera View 31

Figure 4.2.2.4 Drone 2 Taken With Camera 32

Figure 4.2.2.5 Drone 2 Moving Taken With Camera 32

Figure 4.2.2.6 Drone 3 Taken With Camera 33

Figure 4.3.2. Dinosaur. Testing subject for Tensorflow. This is a normal image from google. 36

Figure 4.3.3. Dino Results. The results of Tensor flow with test subject in Figure 4.3.2. This shows how it detected a dinosaur with 97% accuracy. 36

Figure 4.3.4 Drone. Testing subject drone. This is a market picture of a drone and it is the testing subject for the drone example. 37

Figure 4.3.5 Drone results. Results for testing subject drone. Results show with little training 35% accuracy. 37

Figure 4.3.6 Drone training images. Drone images used to train the CNN model. 37

Figure 4.3.7 Bird training images. Bird images used to train the CNN model. 39

Figure 4.3.8 Training process. Process of training CNN model, accuracy and loss are calculated for each image used. 39

Figure 4.3.9 Test result of CNN Model. Image is taken from Testing images 40

Figure 4.3.10 Test result text. Test CNN model code displays answer. 40

Figure 4.3.11 Test result of CNN Model. Image is taken from Testing images 40

Figure 4.3.12 Test result text. Test CNN model code displays answer. 41

Figure 4.3.13 Updated Training process. Process of training CNN model Version 2, accuracy and loss are calculated for each image used 150*150.
43

Figure 4.3.14 Line Chart comparing Training CNN Model of Raspberry Pi 3B(V1) with Raspberry Pi 4(V2)
44

Figure 4.3.15 Results of WAV file

Figure 4.3.16 Graphs of Heavy Rain WAV file. Graphs are Signal vs Time and Frequency vs |F|

Figure 4.3.17 Results of WAV file processing

Figure 4.3.18 Results of WAV file

Figure 4.3.19 Graphs of Labrador Barking WAV file. Graphs are Signal vs Time and Frequency vs |F|

Figure 4.3.20 Results of WAV file processing

Figure 4.3.21 Display of image taken by camera

Figure 4.3.22 Results of frequency

Figure 4.3.23 Results of WAV file

Figure 4.3.24 Graphs of Drone WAV file. Graphs are Signal vs Time and Frequency vs |F|

Figure 4.3.25 Results of WAV file processing

Figure 4.3.26 Display of image taken by camera

Figure 4.3.27 Result of CNN Model

Figure 4.3.28 Display of SMS message

Figure 4.3.29 SMS message

Figure 4.3.30 Display of Email message

Table 3.5.1.1.1 46

Table 4.4.1 46

Figure 7.1 Microphone Circuit 47

Figure 7.2 Microphone PCB 47

Figure 7.3 Drone Detector Case 48

Figure 6.1 World. This image shows a demographic of people who feel threatened by drones. The orange is the color of the demographic and the dark blue is the people who are not. 51

1.0 Introduction

Last year there were “Drone attacks claimed by Yemen’s Houthi rebels struck two key oil installations inside Saudi Arabia on Saturday, damaging facilities that process the vast majority of the country’s crude output and raising the risk of disruption in world oil supplies.”[1]. This attack raised questions as to how low cost drones can be used as potential weapon machines since they are hard to detect, to take down and can cause significant damage. This event gave us the idea to develop a device that can detect drones in a simpler setting like a house since there has been an increase in reports of drones being spotted invading people’s privacy.

With the popularity of drones as toys, they have become a spy tool that can be bought anywhere at an accessible price. Drones can come with a camera that can give the person the opportunity to take great pictures or videos from a big distance resulting in having beautiful shots of landscapes, cities, sunrises, and much more but as enjoyable as they can be, there is also a dark side in the use of the drones where it can be used for malicious purposes. Drones are perfect spying tools for two reasons: The first reason is the high quality of the camera and the second reason is that drones have no limitation in the places it can reach. As a flying object it’s easy to sneak a drone anywhere since it can be hard to detect, this can cause paranoia among people to take drastic measures to take down drones. This scenario happened “Last July, William Merideth, 47, shot down his neighbor’s drone in Louisville, Ky., saying it was spying on his 16-year-old daughter while she was sunbathing by the pool.” [2] . He was taken to court because he was charged with felony charges so he could spend 10 years in jail but at the end he won the case.

More similar cases like Merideth’s can be found throughout the internet since people have never liked being spied on. As a counter measurement, a drone detector could help people know when a drone is being used to invade their privacy instead of being taken by surprise. It is

important to know that drones are protected by the HB2167 law which “permits individuals in certain professions to capture images used in those professions using UAS as long as no individual is identifiable in the image.” [3]. So instead of taking unnecessary aggressive measures against the drone, it is better to take a safer approach such as taking cover from the drone or calling the police.

The HB2167 law can be interpreted in an unethical way and used for purposes other than those established by law, that is to say that any person with a certificate can fly a drone on a person’s property and take photos or videos of their belongings at all times. Also taking into account that it can be difficult to determine if a drone is trespassing when caught in the act, since laws still haven’t been updated to regulate drones correctly, people can have a hard time getting justice so this emphasizes more the need of this project.

“Some countries require that users register their drones before flight. Registration makes it so users can more easily locate their drone if it flies away or be held accountable if they fly irresponsibly. In the United States, anyone with a drone weighing over 0.55 lbs (250 g) is required to register their drone with the FAA. To register, make an account on the FAA’s website and pay the five dollar registration fee.”[4]. This law involves registering a drone at the time of acquiring it and that is where there is a misinterpretation of the same since although it is necessary to register the device, it is not mandatory and as this does not affect the functionality of said device a person with malicious intentions could ignore this law. Since the drone detector is targeted for domestic use, the main goal is to create a drone detector that can be purchased for home use that can be manufactured under $600. This way, drones will be able to keep flying but the owners of property can take the proper measurement in the case of having a drone close by.

The reason why people should consider buying a drone detector from our model is because it's a low cost option and it can hold proof with a picture of the drone that was invading the property.

2.0 Problem Formulation

2.1 Need Statement

Drones have become a popular toy for personal and commercial use so in the last couple of years the market has been growing more and more. The market expects them to reach in 2023 an estimated $141 billion if calculated in the different sectors they are used. For commercial only drones, they have calculated $17 billion. Depending on the price range, they can come with features like a live camera that can be manipulated by a person from far away, or if its a lower priced drone, it can only record in a memory and then be viewed later. With this in mind it is undebatable that cameras can be used for wrong. In December 2017, there was a study made by Pew Research Center about privacy concerns of drones in private property and 54% thought that drones shouldn't be allowed to fly near their property. As drones become more accessible as a toy, the privacy of others are in danger.

The technology available for drone detecting is mostly RF radars. RF (radio frequency) are modules that detect the frequencies around it, so they can detect the specific frequency that a drone generates. These kinds of sensors are built to detect a large area so to put

the area of detection in context for these sensors, the average home for a new single-family is 2,584 square feet. [5]. This sensor detects a mile or 5,280 feet in an angle of 120 degrees.

The drone detectors using RF are high in prince since they are aimed to be used mostly for commercial or military spaces so for example from one company we found the Dedrone RF-100 Drone Defense Countermeasure RF Sensor costs $8,000 and the Dedrone RF-300 Drone Tracking RF Sensor costs $15,000. There are more companies like CRFS, AARTOS, DeTect, and Drone Detector among others that have similar products with prices similar to Dedrone that are too high and too sophisticated to be used for middle or lower class people to be able to afford for their homes since the homes aren't in need of very advanced sensing for security like military or commercial spaces so there should be an affordable option for them to be able to have in their homes.

2.2 Objective Statement

Drone detectors using RF are very expensive for an average house with people that have a limited budget so they can't spend a lot of money on an expensive device to secure their home. Therefore a device can be made with this objective in mind that can be affordable and with the same goal as the expensive devices to detect drones but with lower cost components. It should also be capable of notifying the user if the drone is detected since that is the main purpose to alert the user when a drone is spying through their window.

2.3 Background

The detectors that exist today are too expensive, they mainly use RF sensors and depending on the product, it can have additional sensors like infrared, thermal, audio, and the

range also varies from the different companies. Some products like the Drone Defense from CRFS use RF to geolocate drones, others like the DD610AR Stationary Drone Detector from Drone Detector can be linked with alarms and security response teams and includes audio, RF, GPS, video, thermal, and radar detection. There is also the X3 Laptop, X5 Base, X7 Advanced and X9 Pro from AARTOS that use a IsoLOG 3D DF antenna but have differences in range, capability to detect multiple drones, and mobility in their models. Compared to our drone detector, it only has audio and image detection so it is much simpler since it doesn't need to detect multiple drones at the same time since in an average house there is a higher probability to find only one drone that is spying and not multiple drones since in most cases the offender is the neighbor. If there were a case where two drones did appear in the property of the user to spy, it would have the limitation that it would first process one drone and then the other but this scenario is very unlikely to happen.

3.0 Design Plan / Structure

The Design Plan/ Structure are the various items that were researched for potential development. Why they fail or were proof of how something else could improve the overall project. Different ideas with ultrasonic, waves, camera and microphone. To be able to make a drone detector, the goal was to build a device that had enough range to be able to detect a drone if it appears in any window from a house and find a way to detect effectively the drone and also get some kind of proof so that the person inside the house that is being alerted of the presence of the drone can take actions with the police.

3.1 System Proposed Approach

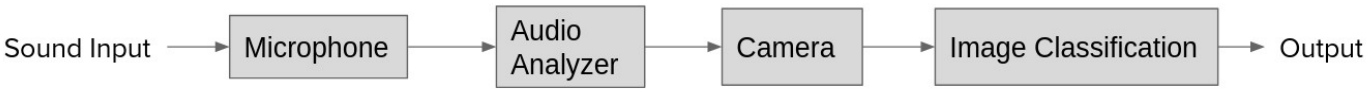


Figure 3.1.1 Block Diagram

The system proposed approach is to have an input and output, having as our input sound, and output would be a notification to the user. The input would have to be consequential from

start to end in order to give the best result and it would have to be audio in order to maintain a low price. From the many methods available to detect drones, image detection is the cheapest on the relationship of accuracy and price. For image detection a camera would be used, but a concern for a constant recording camera is that it can be overheated. Audio detection is lighter on the ram and processor than image detection but the accuracy is low for this it cannot be used as the main input, but it can be used as the input for when to trigger image input. There is no problem in keeping the system running for a long time, because audio detection can function longer without overheating the computer and image detection would be used in small amounts of time. The audio detection works by frequency matches using a microphone. For this a secondary input is necessary to trigger the image input. The design doesn't stop there because any processor cannot be used to process images. A computer that can be on for a long time without being overheated is a necessary because detection cannot have down time. The computer overheating is not a problem if the second input is audio. The frequency detected is created by the blades of the drone. When the frequency is detected then the camera takes pictures. If the picture has nothing then the camera turns off and the microphone is turned on, else a drone is detected, then it can notify. After that it takes a more picture, until no drone is detected.

3.2 System Specification

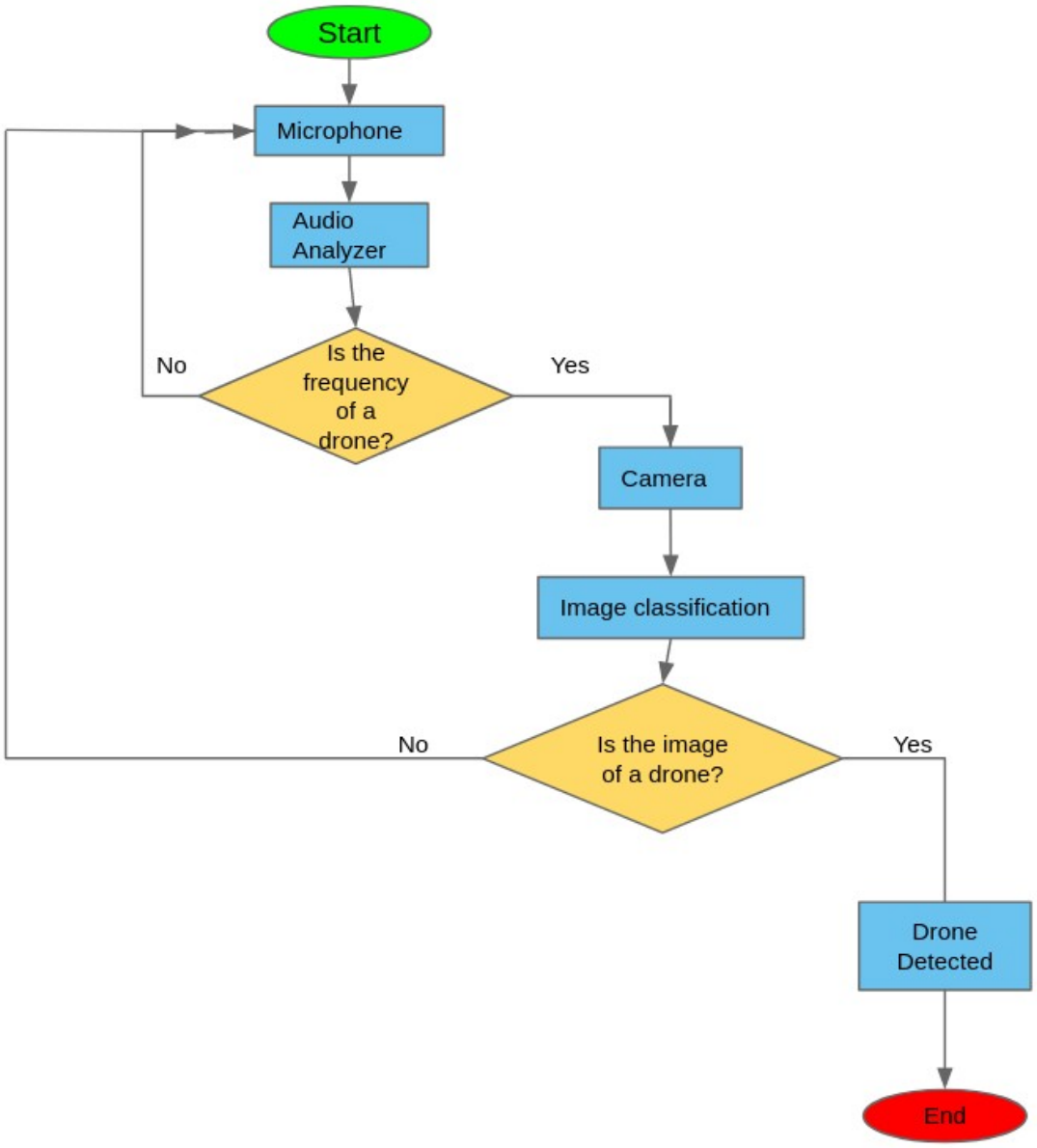


Figure 3.2.1 Flow Chart

3.2.1 Camera

The goal of the system would be to continuously check, if a drone was found then save evidence of it. The camera is the best tool available to be used to detect drones. It has the best relationship of price accuracy. Other more accurate systems would be expensive and would cost too much electricity to run. The main problem with a camera is the heat it created while it runs. The systems specification would require a fan to cold the system down. The problem with this is that a fan would need to be big and have a long life, meaning expensive. For this another input is taken into consideration, audio. With this consequential input of a camera is discarded and overheating problems are solved, but not running the camera continuously also solves the problem that the processor doesn't overheat while running the images.

3.2.2 Microphone

The most affordable option to check for drones is frequency detection. Frequency detection comes in many forms and some of them have high accuracy but it comes with a big price. Frequency detection with a microphone is the most affordable and has the lowest accuracy. The frequency detection with a microphone picks everything with that same frequency as a drone. Too many objects create the frequency of drones, for this audio input is not the best but if used as a first input then it's perfect at triggering the most accurate input, the camera. Using both of them simultaneously would be pointless because they do not complement each other. Using a microphone constantly to take input of frequency doesn't overheat the microphone nor does it require too much from the processor to run audio detection. This doesn't have a high accuracy and doesn't overheat. The way the camera works is that it takes a picture and runs an analysis to decide if there is a drone or not. This has a high accuracy but overheats the system. In the case of

running both programs at the same time the accuracy would not increase and the system would result in an overheating problem. The alternative would be to use the system that doesn't overheat to trigger the one that overheat when it detects something.

3.2.3 Software

Two main softwares would be required to run this system, one for image processing the other the other for the audio processor. Image processing would require the system to handle storing images on the ram then on hard memory. To accomplish the software needs to be pre program outside the main system. This could be done in any computer but the end product should be able to run on the processor. The computer used for this project cannot use a language that makes it difficult to compare images. At the same time a balance between resources use should be maintained because if the language uses too much for its simplicity then it's not good at keeping the cost of the processor low.

3.2.4 Processor

Processors capable of running image detection are not affordable. With keeping the cost low a maintaining fee is introduced, In this case being heat. In order to keep heat low the system must not run extensive tasks for long periods of time. In the case of running extensive tasks for long periods of time cooling methods would be necessary. To avoid heat the extensive task is replaced by a less resourceful task. Audio detection is not as heavy on a processor as image detection. Having audio detection as the task that is running constantly to trigger the camera to

then use image detection. The spikes of processor resources increase when this happens but because it is not constant nor running for too much time the processor has time to cold down.

3.3 Hardware and Software Requirements

3.3.1 Camera 60 FPS

The camera [6] has the requirement that if the system will detect drones then it has to process fast enough for the comparisons to be accurate. Also for the approach of image processing hardware with enough storage and speed to process the code is needed.

3.3.2 Filter Sound from Microphone

The microphone is required to include a high pass filter which allows all frequencies above its cutoff frequency to pass through without attenuation. Frequencies below the cut-off point will be attenuated. As the frequency below the cut-off point decreases, this attenuation, defined in db per octave, increases. It was decided to use a high-pass filter since it focuses specifically on the sound generated by the drone propellers when these are in operation or in the air. For the integration of this filter to our project we decided to choose to use the Audacity audio software due to its functionality and at the same time its practicality when filtering the sounds of any recording, in this case that of our microphone.

3.3.3 Training and Testing CNN Model

Convolutional Neural Network (CNN) “is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.”[7] By using this type of algorithm, the approach was to train the program to recognize an image of a drone and of a bird so that the margin of error is very little when the camera takes a photo. The training CNN Model uses a library called Keras[8] that is required to process the images taken with the camera to be correctly identified as a drone or not drone.

3.4 Schedule

The following schedules consist of assignments with time goals for each task to be completed for Fall 2019 and Spring 2020. Assignments consist of the tasks being done for the drone detector throughout the months to have a consistent advancement in the project that includes research of new topics, building each part of the drone detector, improvements and testing. The time goals are the estimated time that each task will take to complete for each part of the project. Starting Senior Design 2, names were added to each task to make it easier to organize ourselves and to be more informed as a team of what task was assigned to.

3.4.1 Senior Design I Gantt Chart

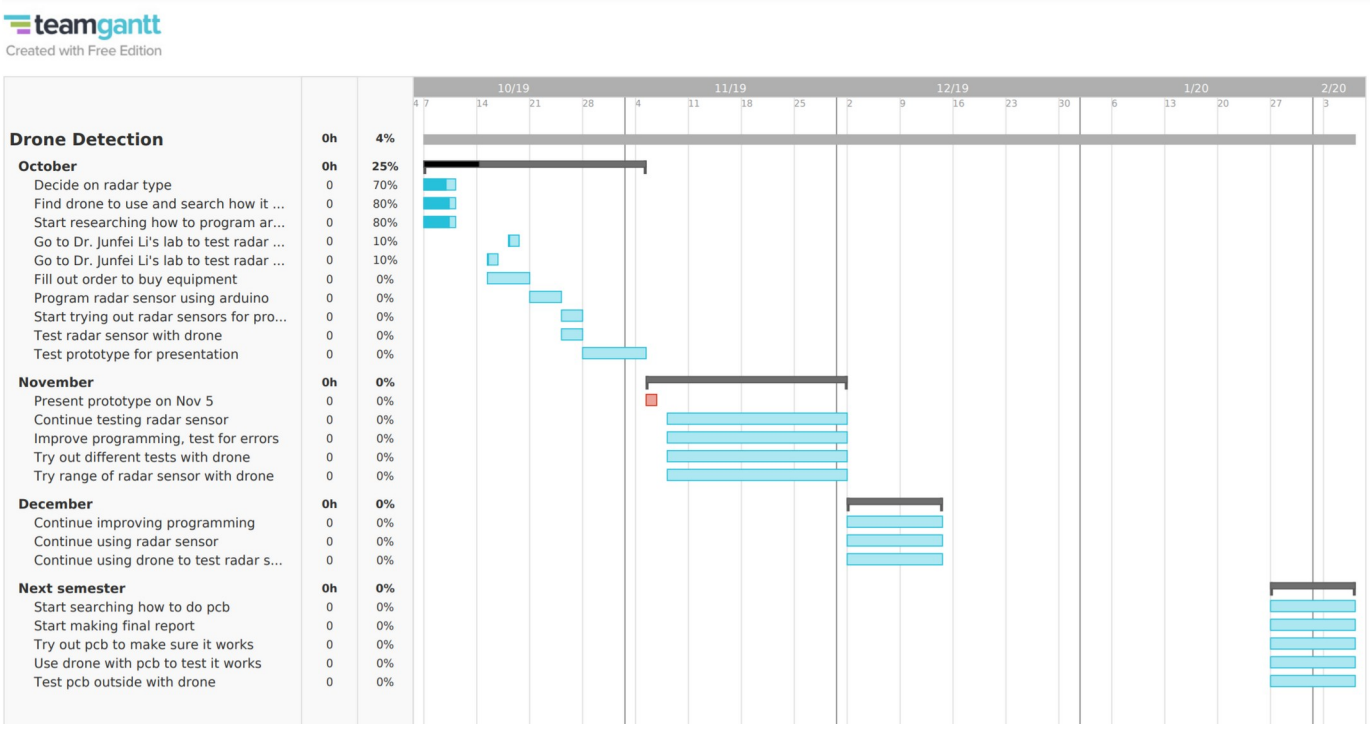


Figure 3.4.1.1 Senior Design I Gantt Chart

3.4.2 Senior Design II Gantt Chart

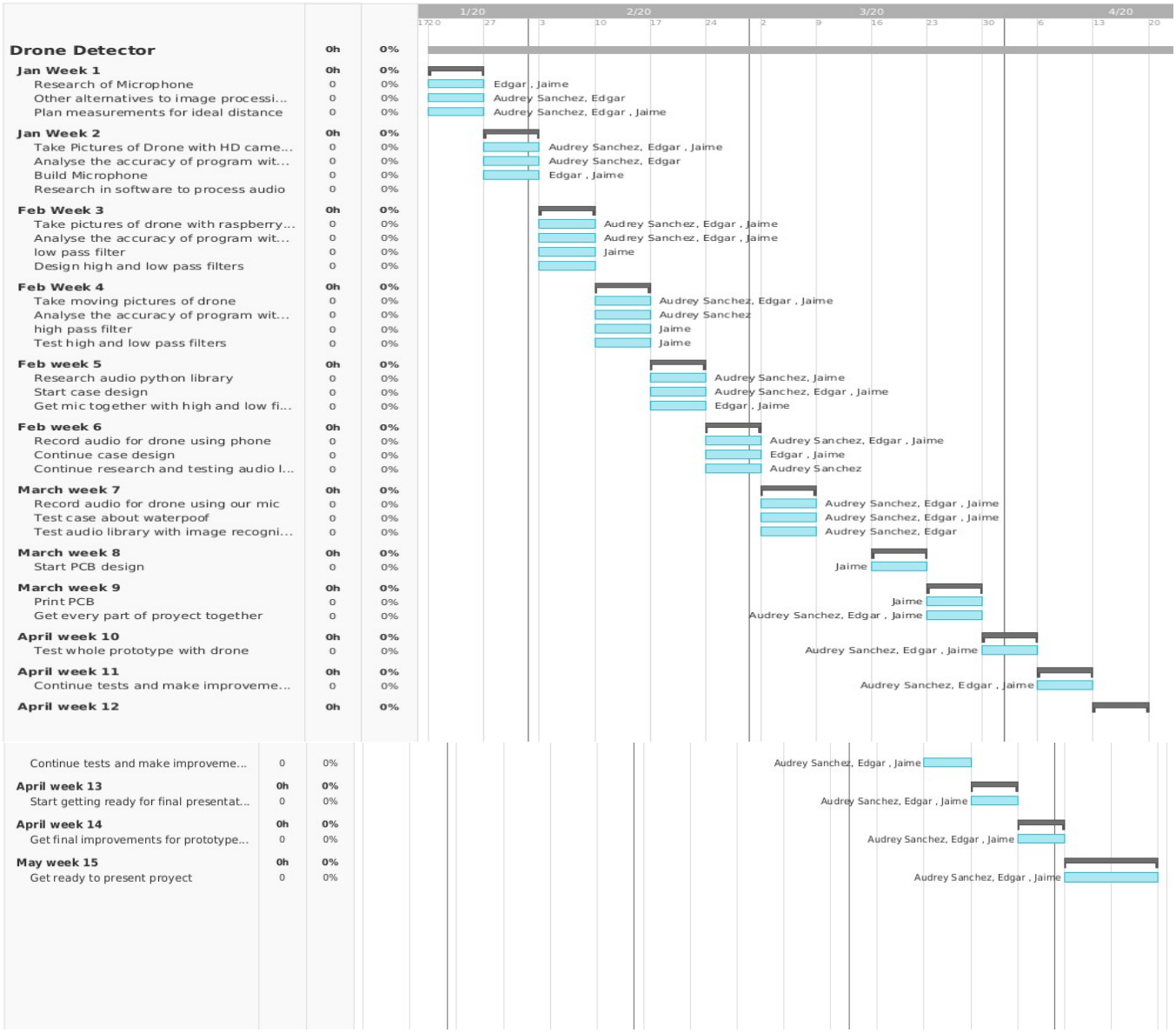


Figure 3.4.2.1 Senior Design II Gantt Chart

3.5 Cost

3.5.1 Table Of Component Prices

3.5.1.1 Items

Table 3.5.1.1.1 Component Prices

Item name	Item amount	Price
Camera 5 mp	1	8.79
Raspberry Pi 4	1	50
Breadboard	1	0
Box	1	25
Total		83.79

The cost of Table 3.5.1.1.1 comes from the items that were used or built during the process of this project.

4.0 Results

The results of different testing and development of components. Focused in supporting the reason why the camera and microphone are the best options after taking into consideration the research done.

4.1 Drone Sound Test using Smartphone Microphone

Inspired by the video[9], it was decided that in order to test a module for a microphone the first thing was to test the noise of a drone in different scenarios. The first being in public with a lot of wind. The video taken with a normal smartphone shows the noise of a DJT drone being stronger than the wind. Measurements were also made based on the sound the drone makes that the phone could detect depending on the distance it was.

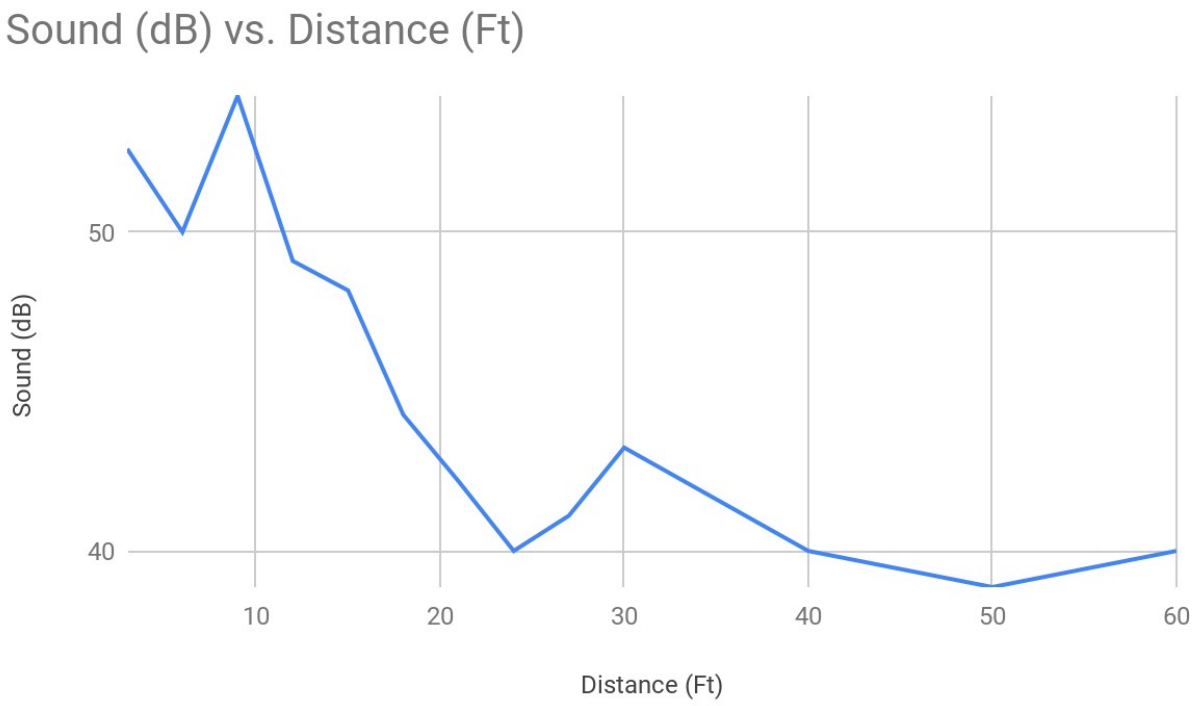


Figure 4.1.1 Line Chart Sound(dB) vs Distance(ft)

Figure 4.1.2 Close Drone Sound Measurement

Figure 4.1.3 Far Drone Sound Measurement

4.2 Camera

4.2.1 ArduCam 2MP oV2640 Mini Module SPI

All test results were made based on the source code[6]. The camera was used to take photos through the arduino to test it's capacity to focus and of quality. In Figure 4.2.1.1, the camera was used to take a picture of the software used, this was to prove that it has enough quality to distinguish different colors and the focus it has. Then the test was made on a corridor, as seen on Figure 4.2.1.2, to test with different lightning and the

results were favorable, the image was clear. Then in Figure 4.2.1.3, the test was made on focus of an object, in this case a person. In Figure 4.2.4 the test was to notice the focus with motion so it can be seen that the silhouette of the person is clear enough to recognize as a person so the goal was met. Next the test was taken into focus on an object relatively close to the camera, so in Figure 4.2.1.5, the object in question is the phone case, it was clearer than the test in Figure 4.2.1.4. Lastly on Figure 4.2.6, the objective was to test the focus on an object that is very close to the camera, and it did result in a clear enough picture but the focus the camera made was to the left side where a monitor appears.

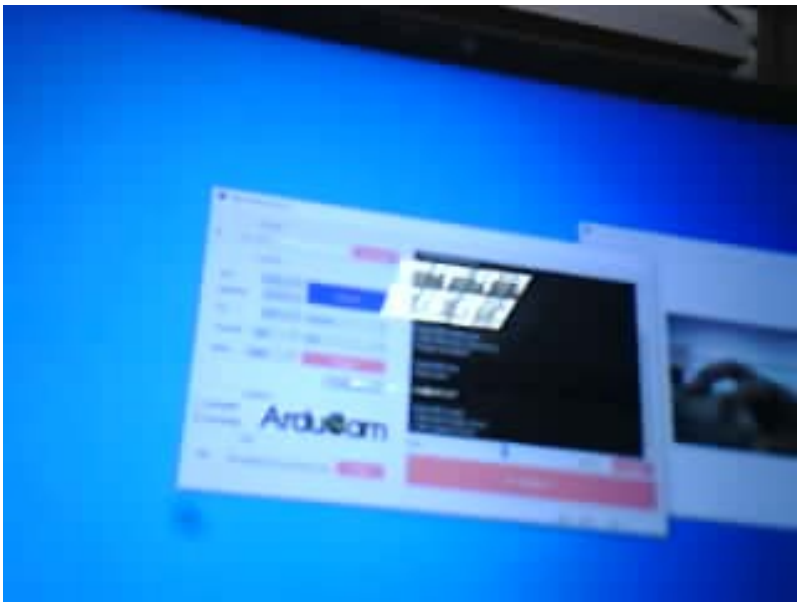


Figure 4.2.1.1 Photo taken of the computer processing arducam. This shows the camera taking a photo of the computer screen.

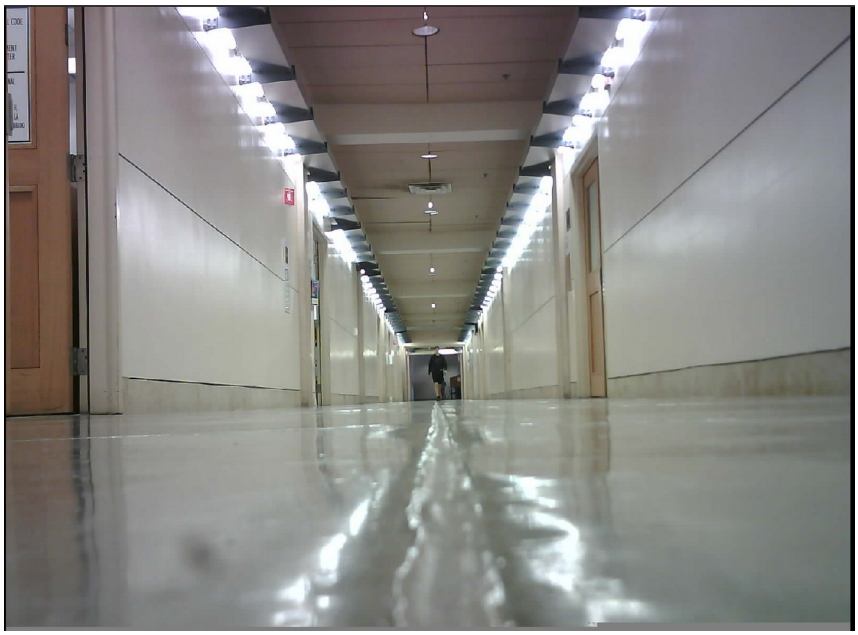


Figure 4.1.1.2 Corridor taken with ArduCam. This shows a test subject at the end of the hall and how it can still be recognised.

Figure 4.1.2.3 Corridor with person in middle. Shows how the person is getting closer to the camera and how it can be distinguished better.

Figure 4.2.1.4 Corridor with a person moving using ArduCam. This shows a moving subject at a short range.

Figure 4.2.1.5 ArduCam focuses on an object. The object is not too far away and it can be seen clearly.

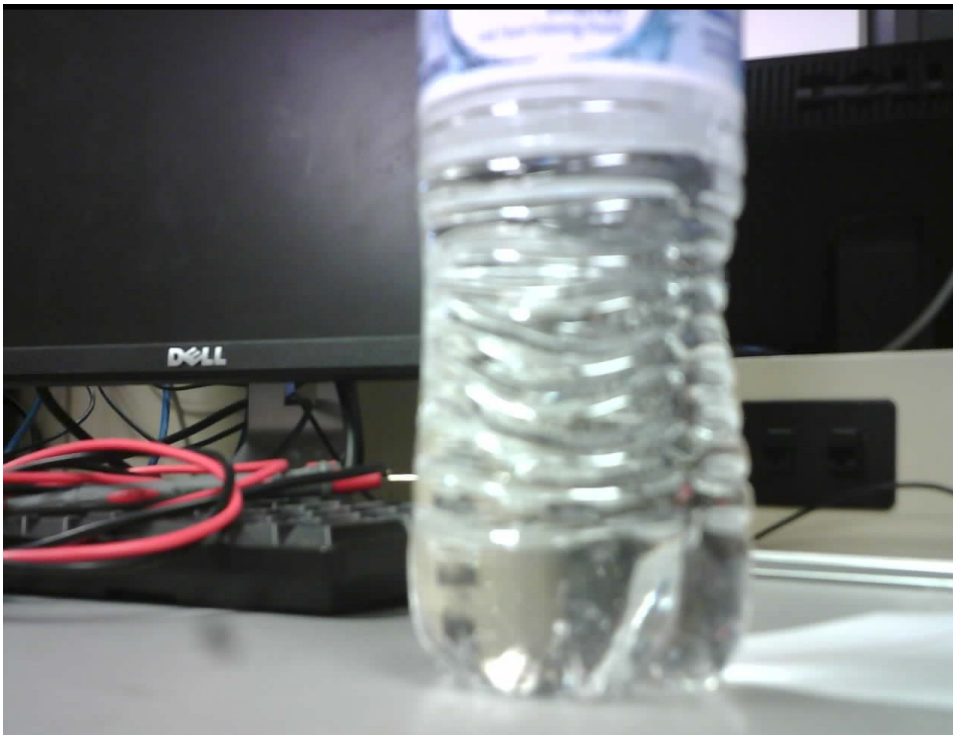


Figure 4.2.1.6. ArduCam closer focus on object. This is a close water bottle.

4.3.2 Raspberry Pi Mini Camera 0V5647

The camera was used to test the field of view it has because it is important to know to be able to position the drone detector, since it is specified in the documentation[10] that it has a horizontal field of view of 54.50+/- 0.13 degrees and a vertical field of view of 41.41 +/- 0.11 degrees, testing the camera can help have a visual guide of the limits the camera may have. The Figure 4.2.2.1 shows the field of view of this particular camera and in Figure 4.2.2.2 measurements were recorded on Table 4.2.2.3 to know based on the distance an object may be from the camera, the reach it can have in base and height.

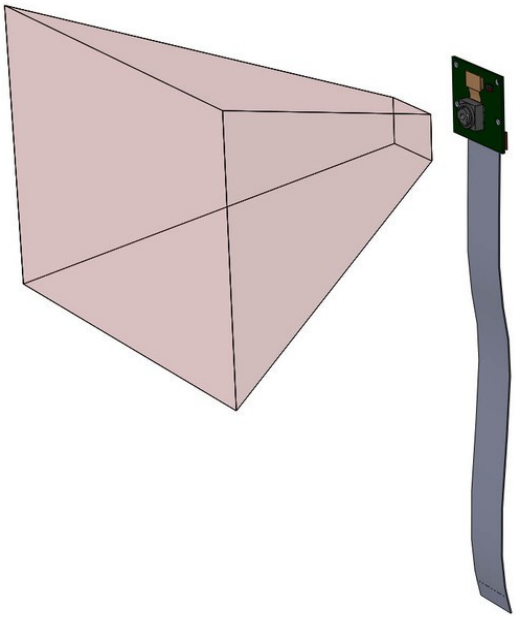


Figure 4.2.2.1 Field of View of Camera[11]

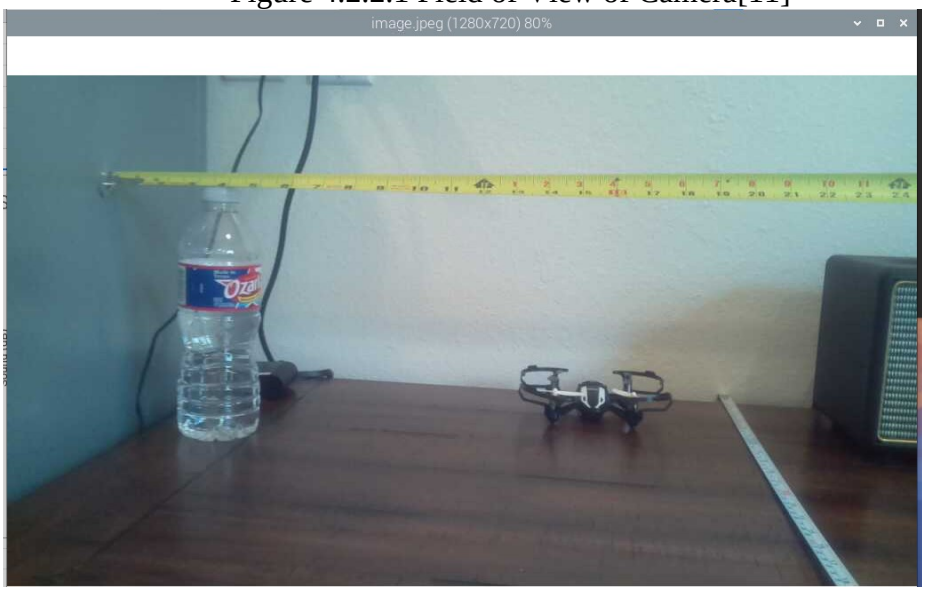


Figure 4.2.2.2 Measure of Camera View

Table 4.2.2.3 Measurements of Camera View

Distance(Ft)	Base(Ft)	High(Ft)
3	2	1
6	4.4	3.7
9	7	4.2
12	9.2	5.2

The camera was also used to visualize the focus it can have on stationary objects like the drones in Figure 4.2.2.4 and Figure 4.2.2.6 that are different models, and on moving objects like Figure 4.2.2.5 and Figure 4.2.2.7. When the drone was moving, the image taken with the camera was blurry, it couldn't focus on it but it can still be identifiable. This is an important detail because the CNN Model has to be able to identify the drone in all types of scenarios that can be when it is night, day, or if the object passed the drone detector, or if it is fixed in a place when flying.

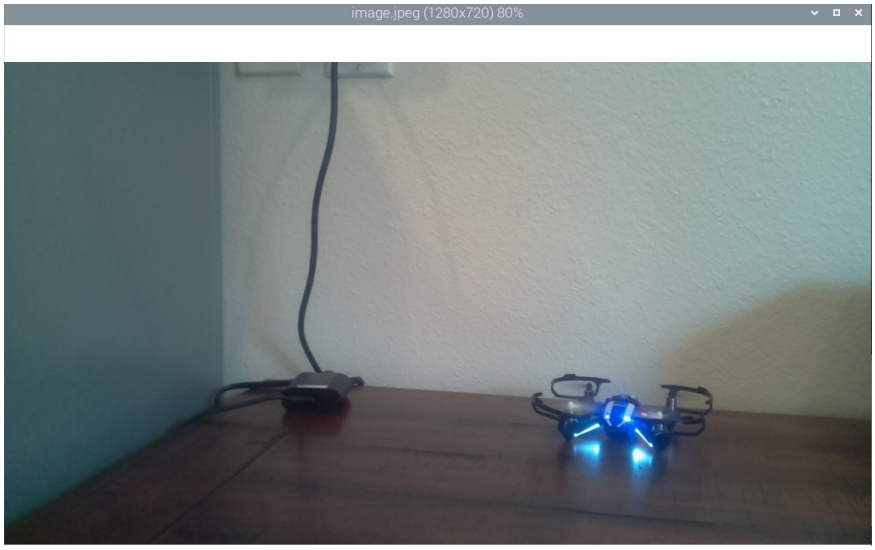


Figure 4.2.2.4 Drone 2 Taken With Camera

Figure 4.2.2.5 Drone 2 Moving Taken With Camera

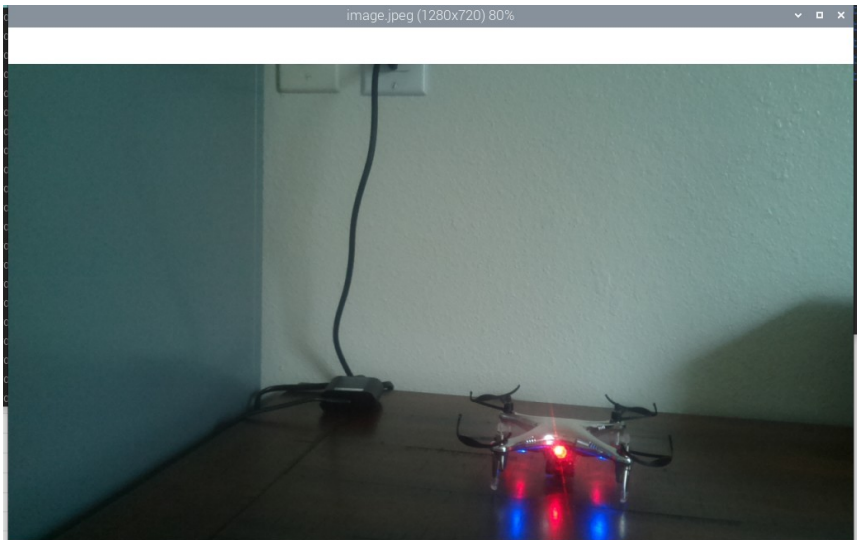


Figure 4.2.2.6 Drone 3 Taken With Camera

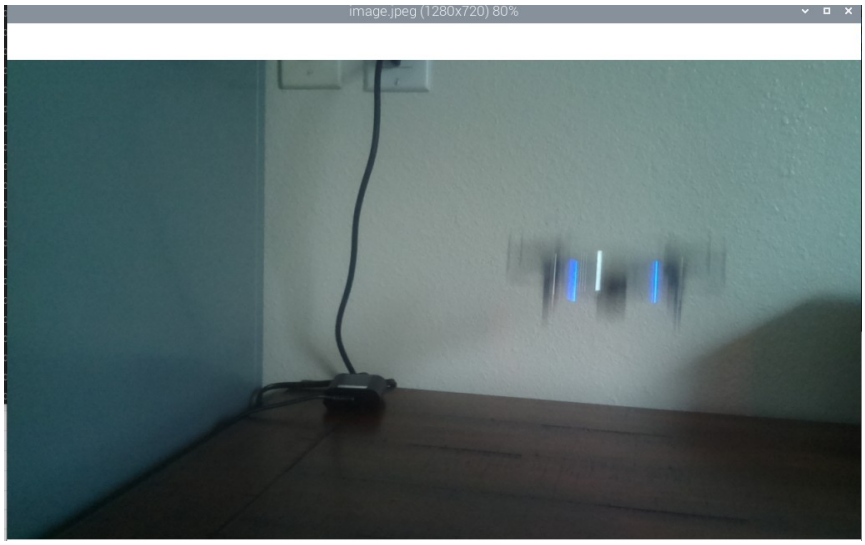


Figure 4.2.2.7 Drone 3 Moving Taken With Camera

4.3 CNN Model

The Raspberry Pi [12] was used to run the library Tensorflow to test the image recognition. The Figure 4.3.1 was a test image to see if it worked, and it successfully recognized a triceratops as seen on Figure 4.3.2. The Figure 4.3.3 was to test if it could recognize a drone and the results weren't the expected as seen in Figure 4.3.4. It didn't recognize it and categorized it as completely different objects. With this, the library has to be used to train the software so that based on the models it has which will be images of different types of drones, it should distinguish between a drone and something else. To accomplish this task the library Keras[8] was used to program a training model that uses Convolutional Neural Network based on a tutorial [13].

Figure 4.3.1. Dinosaur. Testing subject for Tensorflow. This is a normal image from google.

```
triceratops (score = 0.96866)
hippopotamus, hippo, river horse, Hippopotamus amphibius (score = 0.00127)
frilled lizard, Chlamydosaurus kingi (score = 0.00098)
African crocodile, Nile crocodile, Crocodylus niloticus (score = 0.00072)
Indian elephant, Elephas maximus (score = 0.00067)
pi@raspberrypi:~/tf1/models/tutorials/image/imagenet $
```

Figure 4.3.2. Dino Results. The results of Tensor flow with test subject in Figure 4.3.2. This shows how it detected a dinosaur with 97% accuracy.

Figure 4.3.3 Drone. Testing subject drone. This is a market picture of a drone and it is the testing subject for the drone example.

Figure 4.3.4 Drone results. Results for testing subject drone. Results show with little training 35% accuracy.

In Figure 4.3.5 and Figure 4.3.6, the images were taken from two folders which are the dataset meant only to train the CNN Model. They are used to teach the program to identify the images with the class name it was specified on the training code that was based on the code[30] from the tutorial. Figure 4.3.7 displays the training process where it displays in each epoch, cycle going through all the images in the dataset, the accuracy and loss of each image so it can be appreciated how the accuracy grows with each image it analyzes.

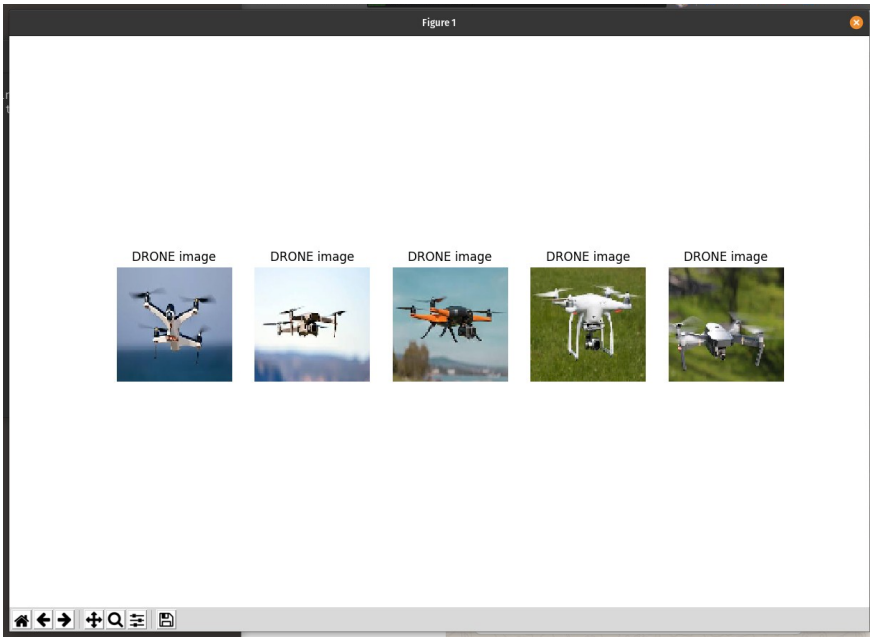


Figure 4.3.5 Drone training images. Drone images used to train the CNN model.

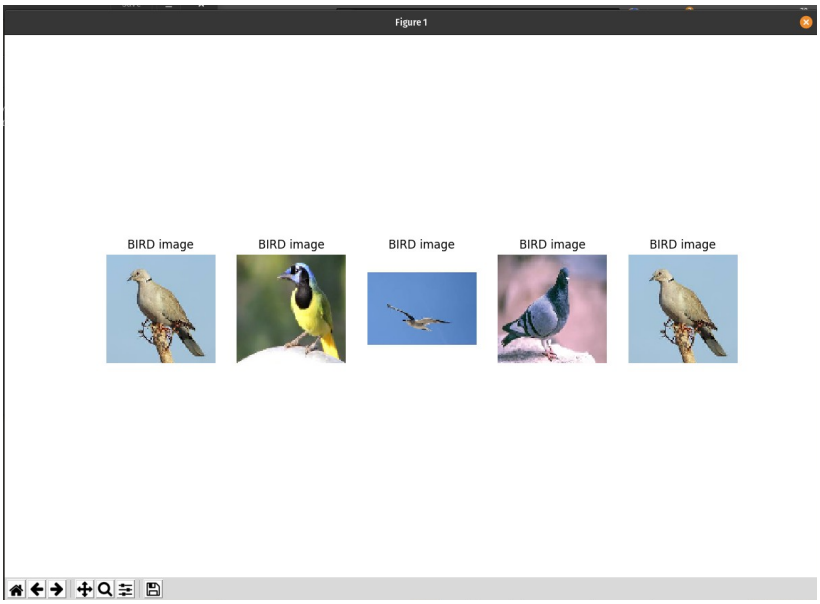


Figure 4.3.6 Bird training images. Bird images used to train the CNN model.

```
Epoch 1/20
72/72 [==============================] - 1s 20ms/step - loss: 5.3980 - accuracy: 0.5694
Epoch 2/20
72/72 [==============================] - 1s 12ms/step - loss: 4.2548 - accuracy: 0.4583
Epoch 3/20
72/72 [==============================] - 1s 12ms/step - loss: 1.3620 - accuracy: 0.4167
Epoch 4/20
72/72 [==============================] - 1s 11ms/step - loss: 0.7814 - accuracy: 0.5000
Epoch 5/20
72/72 [==============================] - 1s 12ms/step - loss: 0.6473 - accuracy: 0.6111
Epoch 6/20
72/72 [==============================] - 1s 11ms/step - loss: 0.7069 - accuracy: 0.4861
Epoch 7/20
72/72 [==============================] - 1s 11ms/step - loss: 0.6849 - accuracy: 0.5833
Epoch 8/20
72/72 [==============================] - 1s 11ms/step - loss: 0.6672 - accuracy: 0.6111
Epoch 9/20
72/72 [==============================] - 1s 13ms/step - loss: 0.6714 - accuracy: 0.6389
Epoch 10/20
72/72 [==============================] - 1s 13ms/step - loss: 0.6610 - accuracy: 0.6667
Epoch 11/20
72/72 [==============================] - 1s 13ms/step - loss: 0.6468 - accuracy: 0.6528
Epoch 12/20
72/72 [==============================] - 1s 12ms/step - loss: 0.6507 - accuracy: 0.6250
Epoch 13/20
72/72 [==============================] - 1s 13ms/step - loss: 0.6052 - accuracy: 0.7222
Epoch 14/20
72/72 [==============================] - 1s 13ms/step - loss: 0.6046 - accuracy: 0.7222
Epoch 15/20
72/72 [==============================] - 1s 12ms/step - loss: 0.5506 - accuracy: 0.7639
Epoch 16/20
72/72 [==============================] - 1s 12ms/step - loss: 0.5021 - accuracy: 0.7778
Epoch 17/20
72/72 [==============================] - 1s 13ms/step - loss: 0.4804 - accuracy: 0.7500
Epoch 18/20
72/72 [==============================] - 1s 13ms/step - loss: 0.5598 - accuracy: 0.6944
Epoch 19/20
72/72 [==============================] - 1s 12ms/step - loss: 0.3886 - accuracy: 0.8333
Epoch 20/20
72/72 [==============================] - 1s 13ms/step - loss: 0.3222 - accuracy: 0.9306
```

Figure 4.3.7 Training process of CNN model Version 1. Accuracy and loss are calculated for each image size 96*96.

Figure 4.3.8 and Figure 4.3.9 are the results of the test code that was based on the code [14] of the tutorial, the images look a little pixelated because of the size given of 96*96, this was a limitation that had to be done because the Raspberry Pi 3 couldn’t run the training code in a higher image size since it only has 1GB of RAM. After it displays the images, on the terminal, It displays the correct answer based on the image of a drone, then Figure 4.3.10 and 4.3.11 also display the correct answer based on the image of a bird.

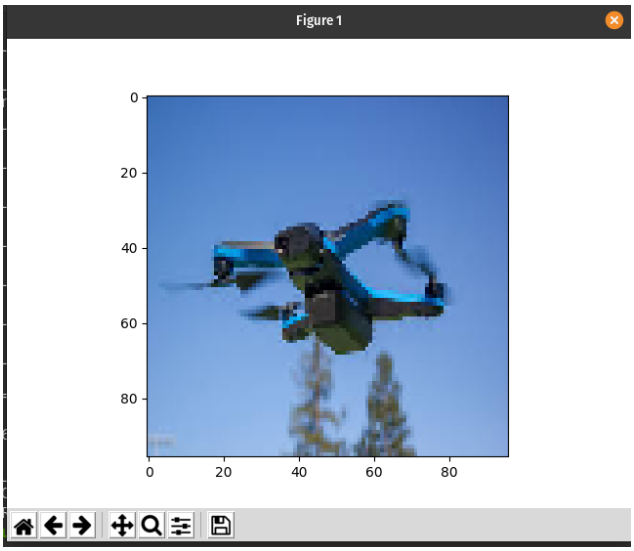


Figure 4.3.8 Test result of CNN Model. Image is taken from Testing images

```
The type predicted is: DRONE
```
Figure 4.3.9 Test result text. Test CNN model code displays answer.

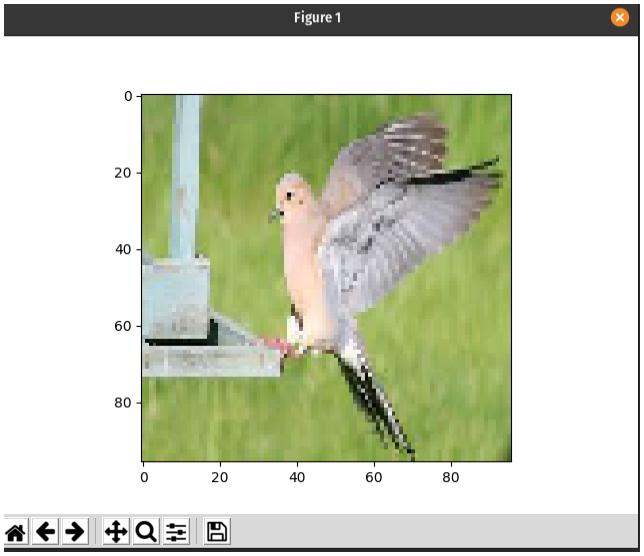


Figure 4.3.10 Test result of CNN Model. Image is taken from Testing images

```
The type predicted is: BIRD
```
Figure 4.3.11 Test result text. Test CNN model code displays answer.

When testing the CNN Model, the objective was to show images of different models of drone that were at different angles and different times of the day, and also of different objects like birds, sky, and lawn mowers among other images. When the CNN Model was tested, those images were tested one by one simulating that the camera had taken it to see the expected result that it would be able to recognize the drone. The testing of the microphone was similar, we had to record the drones that we had, and also look for more recordings of different models of drones because since drones vary in size and quality, the sound can also vary. From the three drones that we had, the sound was recorded first using the same phone and then were saved as a wav file to the folder where the code would grab it to process it and find its frequency. Another important thing that the microphone had to be tested on was of similar sounds that could make the code of the microphone think it was a drone. Having all the recordings saved, the microphone code was tested one by one to have the frequency calculated and have an expected result where it would be able to differentiate between the frequency of drones and from similar sounds like a lawn mower.

Starting Spring 2020, the number of images increased, having originally 43 to 125 drone images and no drone images, previously the folder being named bird, increased 37 to 107 to have a larger dataset. The Raspberry Pi 3B was changed with the newer model 4 to be able to improve the accuracy of the drone detector by having the images used to train the CNN Model to be bigger. Also the image size was changed from being 96*96 to 150*150. The CNN Model had to be retrained as seen in Figure 4.3.12, being Version 2, that greatly resembles Figure 4.3.7 but it now has a more precise accuracy of 0.9955 compared from the previous one that had accuracy of 0.9306 This change greatly improved the CNN Model.

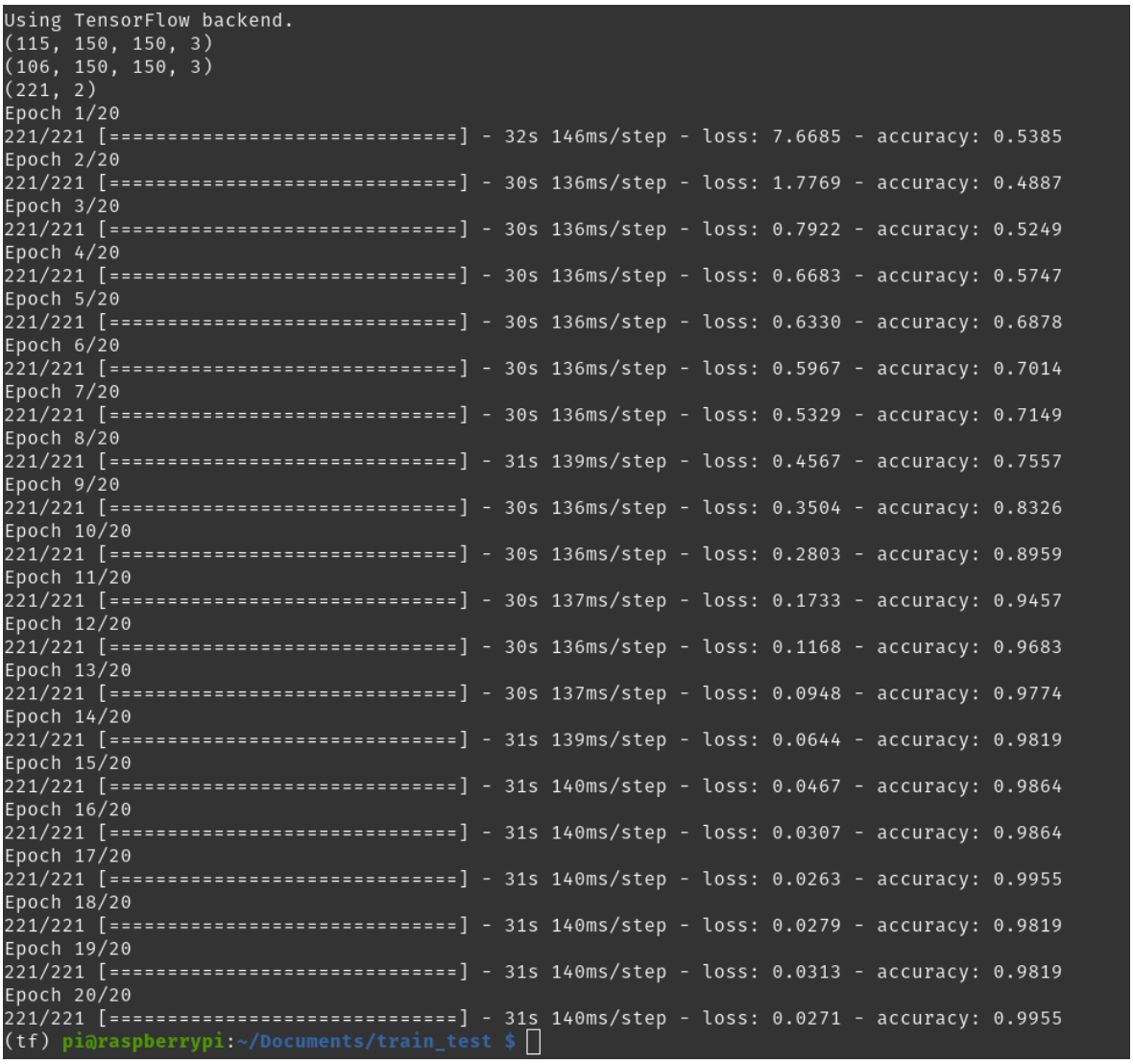


Figure 4.3.12 Updated Training process. Process of training CNN model Version 2, accuracy and loss are calculated for each image used 150*150.

The average accuracy of Version 1, being the Raspberry PI 3B with image size 96*96, was 0.6197 and average loss of 1.0072 , and Version 2, being the Raspberry Pi 4 with image size 150*150, had an average accuracy of 0.7864 and an average loss of 0.6854. As a result, the accuracy improved by 0.1667 and the loss by 0.3218, and the training ended for Version 1 ended with an accuracy of 0.9306, loss of 0.3222 as seen in Figure 4.3.8 and Version 2 with accuracy

of 0.9955, loss 0.0271 seen in Figure 4.3.13. The epoch being 30 and batch size 32 for both versions stayed the same because even though the Raspberry Pi 4 has more RAM, they could be increased having all the images repeated more times in each step but it wasn't needed because having a larger dataset with more variety of drone models and of no drone objects that included the sky, lawn mower, birds, and lawn edgers, with an added benefit of changing the resolution of the images to 150*150 was enough to have at the end an accuracy closer to 100.

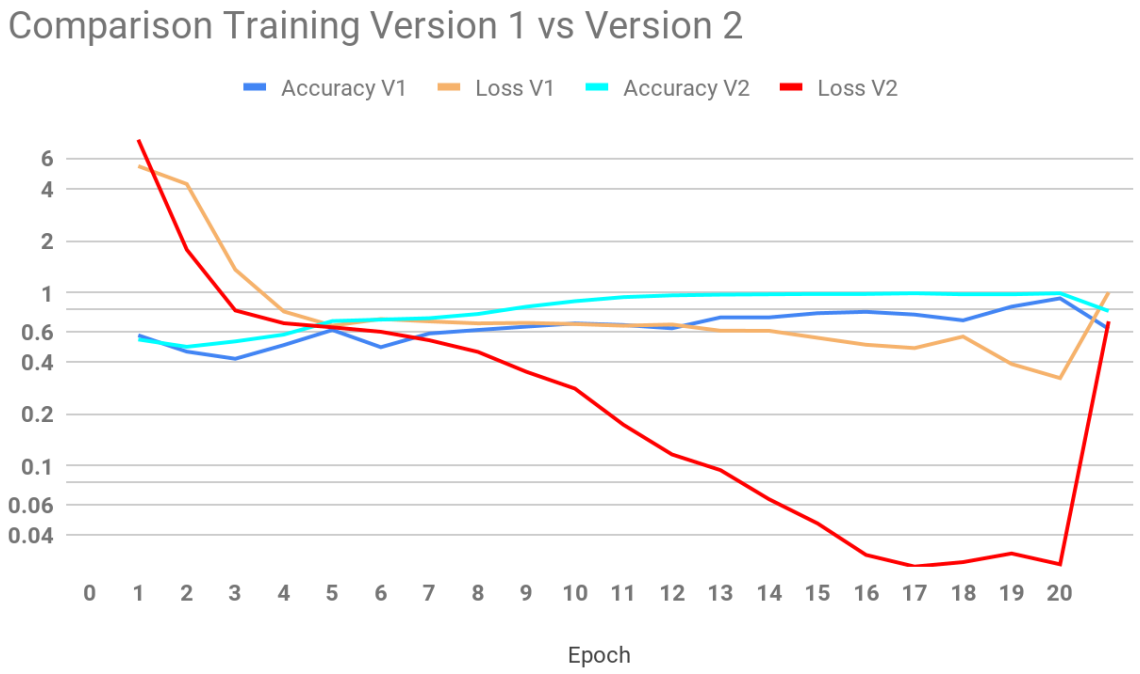


Figure 4.3.13 Line Chart comparing Training CNN Model of Raspberry Pi 3B(V1) with Raspberry Pi 4(V2)

For the testing of the CNN Model code, the only thing changed was the image size to be 150*150 that is the same as the training CNN Model code. The microphone part of the code was redone, using now "sounddevice" and "wavfile" libraries instead of pyaudio because they were much simpler to code for the purpose to record for 15 seconds and save in a WAV file. After the

WAV file is saved in a specific folder, the code grabs it to start processing the file and calculate the frequency that is then compared in an if statement with the values a drone has. If it is within the range of a drone, it passes to the camera part of the code to take a picture, save it in a specific folder so that the testing CNN Model part of the code can grab the image and start the process to decide if it really is a drone.

When the WAV file is processed, the sample rate, data.shape, duration, and frequency are displayed in the command line like in Figure 4.3.14. In the next part of the code, the soundwave of the WAV file is displayed like in Figure 4.3.15 being the first graph of signal vs time(s). Then with the previously calculated data, the Fast Fourier Transform(FFT) function calculates the frequency and display it as a |F| vs frequency(Hz) graph where in Figure 4.3.15, it is the second graph.

```
  sample_rate, data = wav.read('sounds/heavy-rain-daniel_simon.wav' ) #sample_rate= samples/sec
sample rate 44100
data.shape  (3811417, 2)
duration 86.42668934240363
The frequency is 35 Hz
```

Figure 4.3.14 Results of WAV file

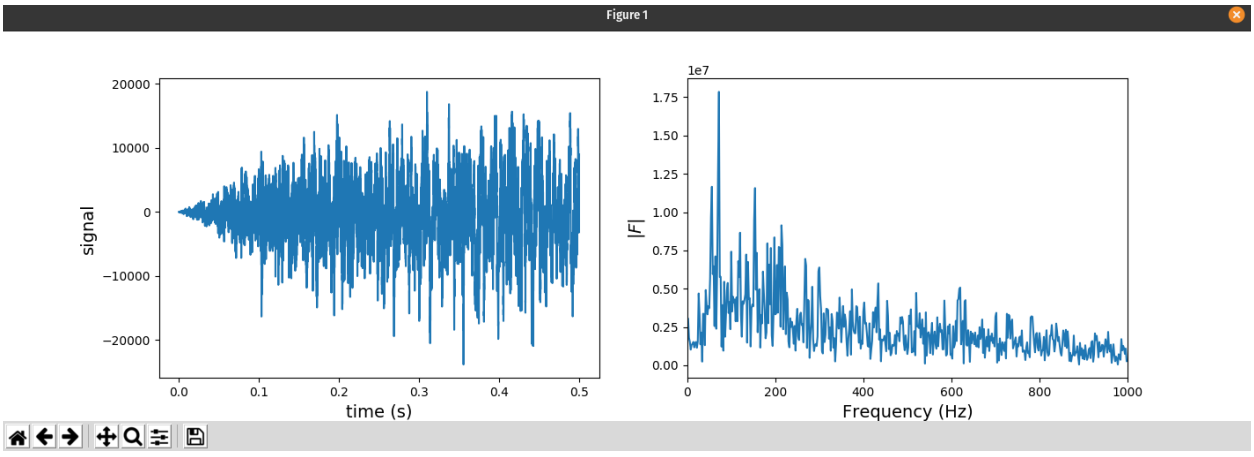


Figure 4.3.15 Graphs of Heavy Rain WAV file. Graphs are Signal vs Time and Frequency vs |F|

Based on the frequency calculated being in this case 35Hz, it is displayed in Figure 4.3.16 that it is not a drone. This is because the frequency range of a drone is between 100 and 500 Hz and also a frequency range 1,000 and 5,000 HZ, depending on the model of the drone. After it displays the results, it stays in its stand-by mode where the microphone is “ON” waiting for a loud sound to be heard.

```
Frequency detected is: NOT DRONE
```

Figure 4.3.16 Results of WAV file processing

The previous results showed where the first case scenario happens if the sound is identified as being not of a drone but in cases of a second scenario where the sound is within the two frequency ranges, like in Figure 4.3.17. The sound in this case is of a labrador barking, the frequency calculated is of 236 Hz so it is within the 100 and 500Hz so it will go through almost all the code. It does the same as scenario one showing both graphs like in Figure 4.3.18, and next it displays the final results of the WAV file in Figure 4.3.19 that it was a drone the microphone detected.

```
  sample_rate, data = wav.read('sounds/labrador-barking-daniel_simon.wav' ) #sample_rate= samples/sec
sample rate 44100
data.shape  (946500, 2)
duration 21.462585034013607
The frequency is 236 Hz
```

Figure 4.3.17 Results of WAV file

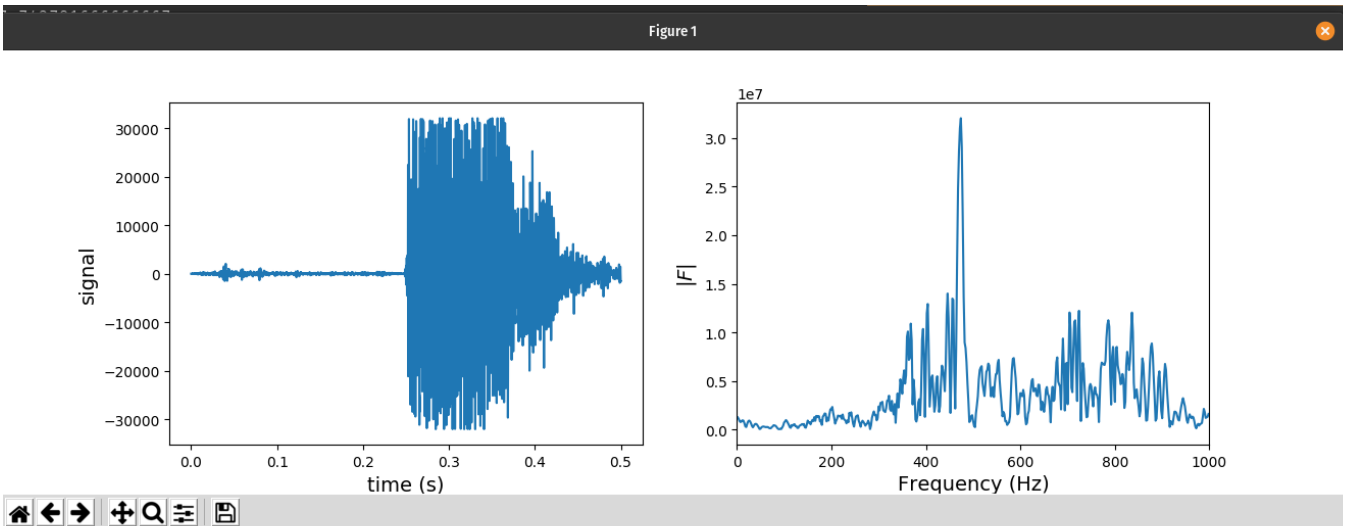


Figure 4.3.18 Graphs of Labrador Barking WAV file. Graphs are Signal vs Time and Frequency vs |F|

```
Frequency detected is: DRONE
```

Figure 4.3.19 Results of WAV file processing

The next of the code that scenario two goes to the camera where it takes the picture, saves it in the specified folder and displays it as in Figure 4.3.20 which in this case was a clear sunny day. Then it goes to the testing of CNN Model part of the code and displays the results in Figure 4.3.21 that it was in fact not a drone so since it was a false alarm it will stop here and return to its stand-by mode.

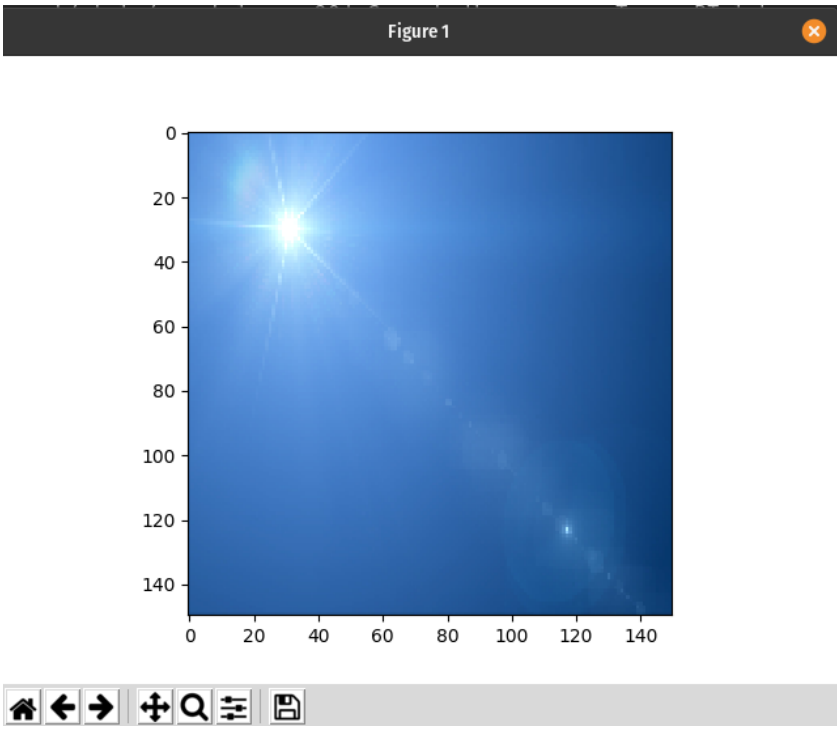


Figure 4.3.20 Display of image taken by camera

```
The type predicted is: NO_DRONE
```
Figure 4.3.21 Results of frequency

The third scenario in which a drone is in fact detected is shown starting in Figure 4.3.22 where the sound being detected is of a drone. The frequency calculated is 282 Hz so based on the frequency, it is within the range between 100 and 500 Hz. It will again as the past two scenarios display the graphs as in Figure 4.3.23, then it will display the answer in Figure 4.3.24 based on the calculations made so it detected the sound of a drone.

```
sample_rate, data = wav.read('sounds/drone2.wav') #sample_rate= samples/sec
sample rate 48000
data.shape  (371702, 2)
duration 7.743791666666667
The frequency is 281 Hz
```
Figure 4.3.23 Results of WAV file

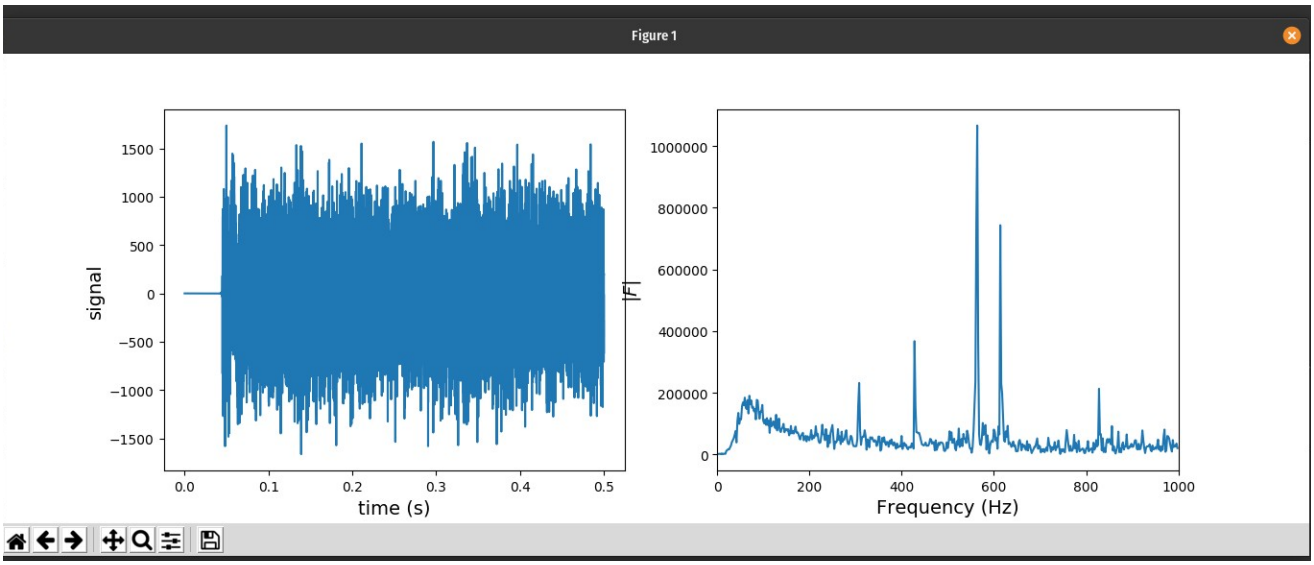


Figure 4.3.23 Graphs of Drone WAV file. Graphs are Signal vs Time and Frequency vs |F|

```
Frequency detected is: DRONE
```
Figure 4.3.24 Results of WAV file processing

Since the sound was detected being of a drone, it goes as in scenario two to the camera, the image taken is saved in the specified folder, then it is displayed as in Figure 4.3.25 in this case being of a drone flying in a cloudy day. Then it displays as in Figure 4.3.26 the answer of the CNN Model that it is in fact a drone that the microphone detected. After this part of the code, it goes to the new addition made in the semester Spring 2020 where it will notify the user that a drone was detected within their property to be able for them to take proper precautions. This is accomplished by sending two types of notifications being an SMS message to the phone and also to their email. Two ways of notifying were decided to do so that in case the user doesn't have their phone with them, there is a second way they can know.

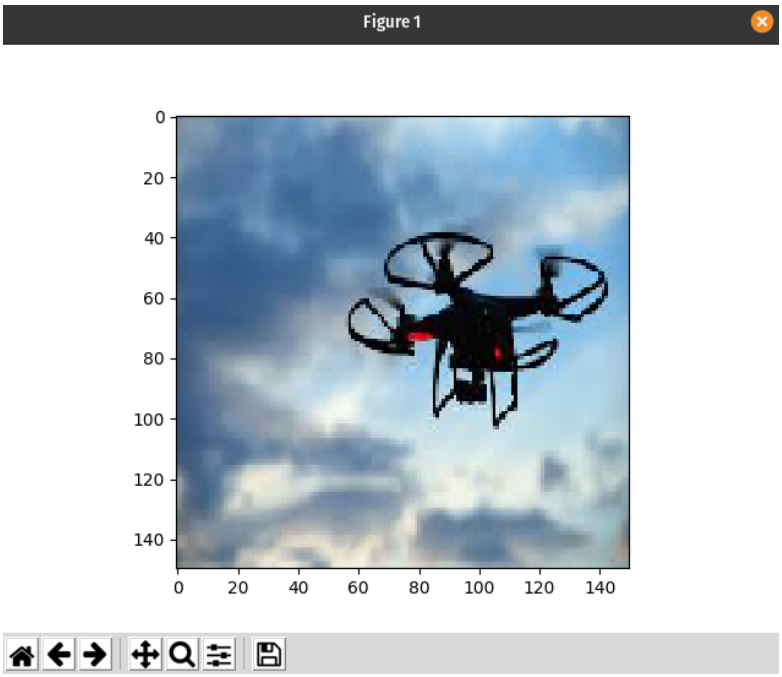


Figure 4.3.25 Display of image taken by camera

Figure 4.3.26 Result of CNN Model

To accomplish these tasks, an email account was created to be able to notify them through email. For the SMS message, a free account was made on the website “Twilio”[15], where they assign a trial phone number with a free balance of $15.50. The notification part of code now starts in scenario three where it will send an SMS message to the user as in Figure 4.3.27 and Figure 4.3.28 with the message “Dear User, A DRONE was detected on your property, please take safe precautions". Then in Figure 4.3.29 the email was also sent with the same message as the SMS, but with a small difference being that an attachment was added with the image of the drone. The reason why only the email has the image is because Twilio was used with a free account so the feature to send an MMS where the image would be also sent to the phone is only available for upgraded accounts. When the drone detector is used commercially,

this feature will be added since the account will have to be upgraded to be able to send to all potential customers.

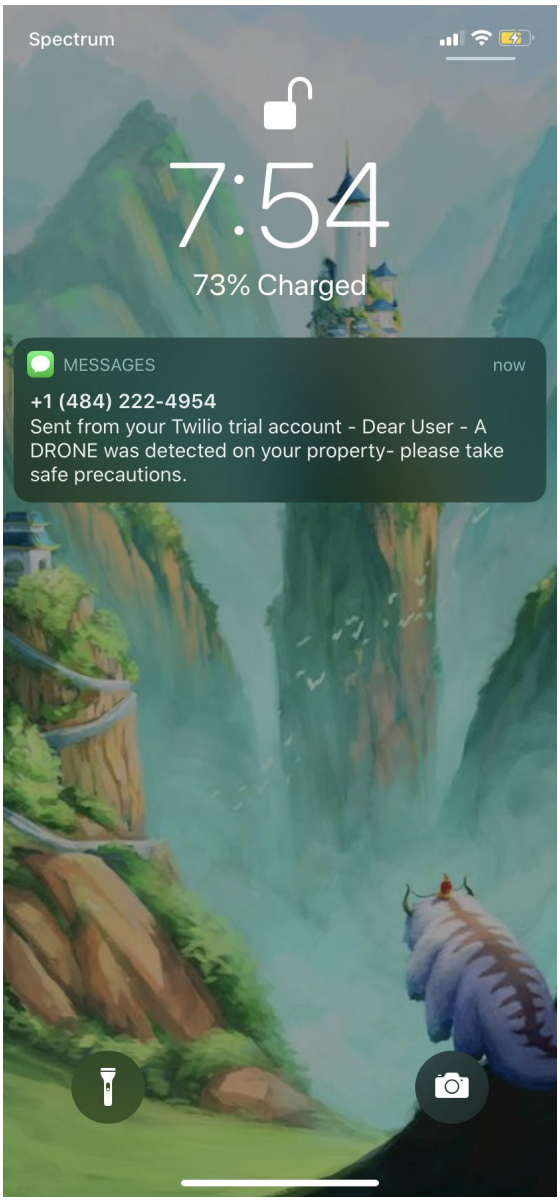


Figure 4.3.27 Display of SMS message

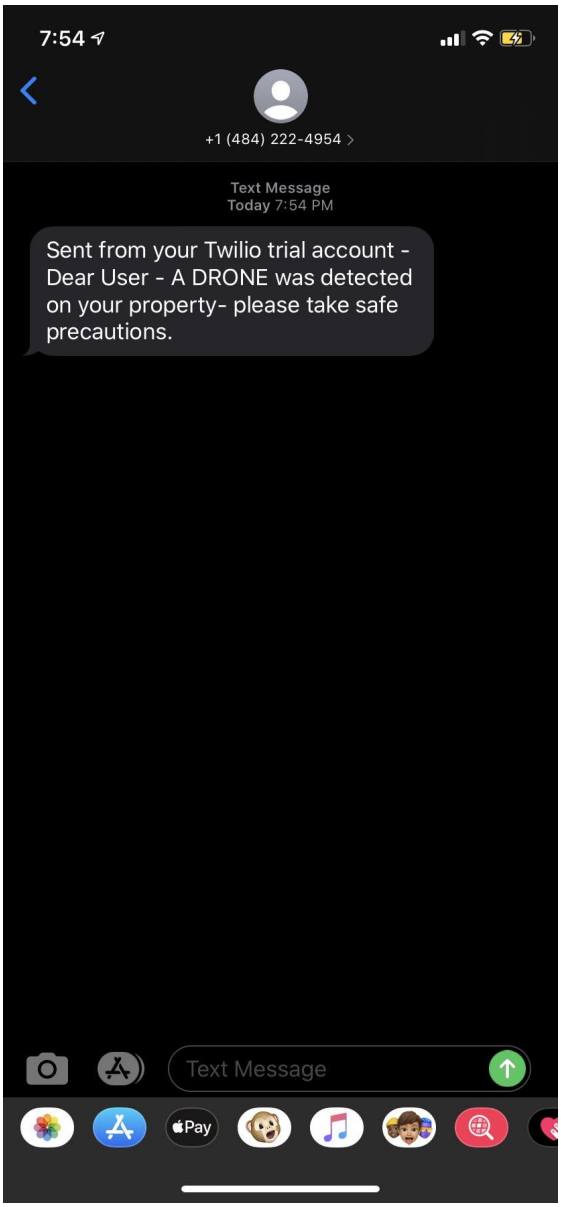


Figure 4.3.28 SMS message

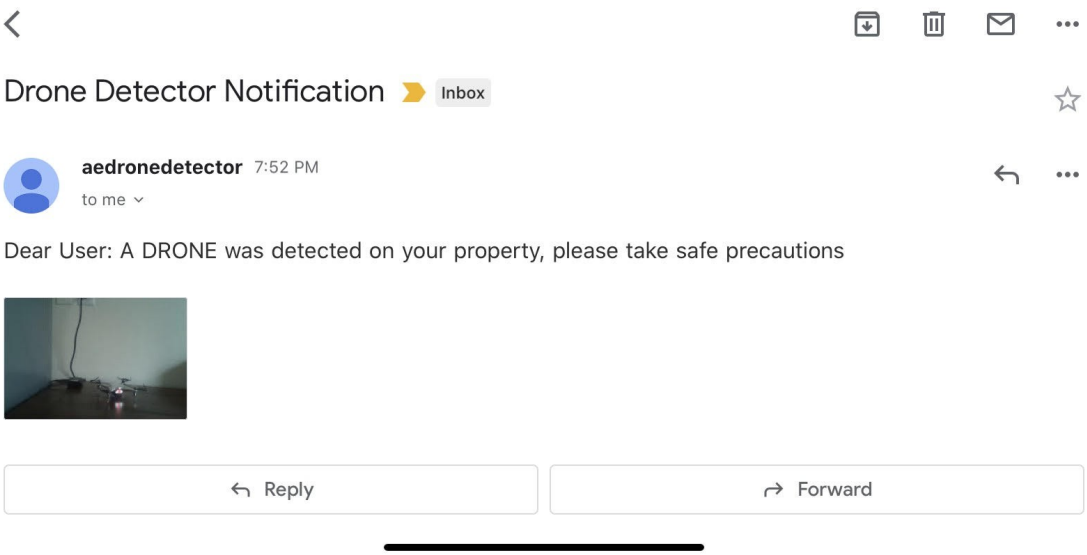
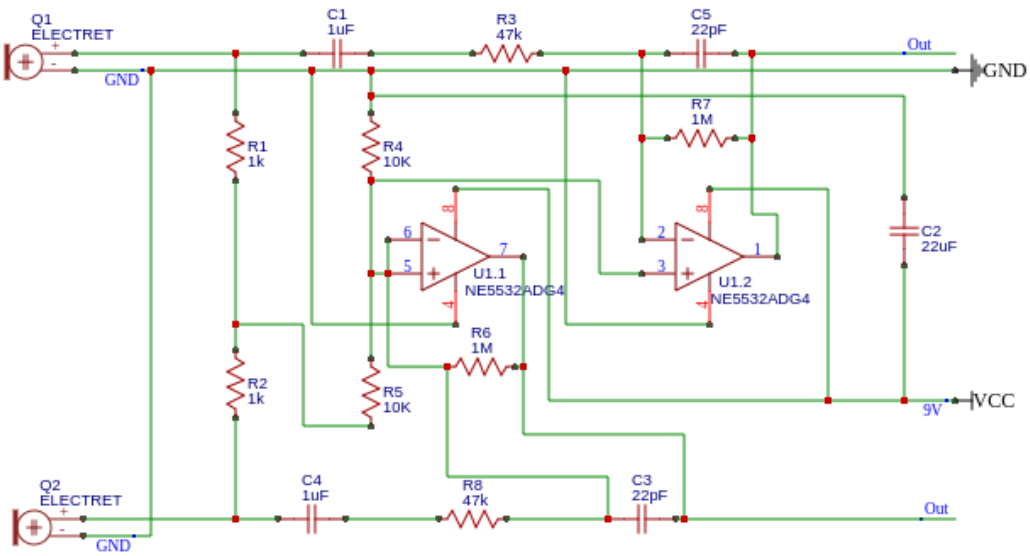


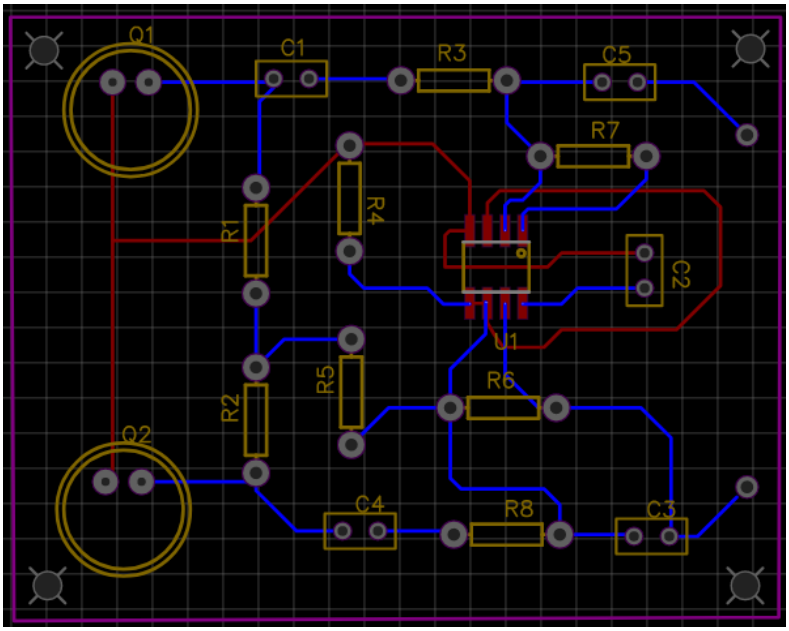
Figure 4.3.29 Display of Email message with Image Attachment of Drone

4.4 Microphone

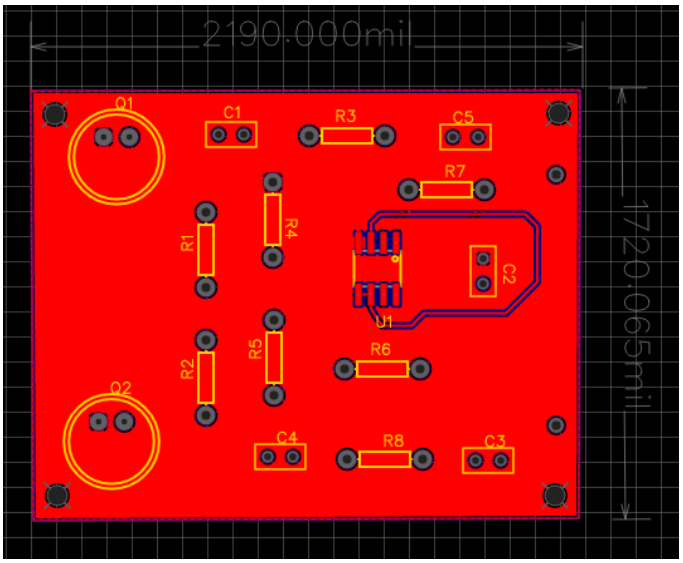
The circuit and PCB design was done using EasyEDA[16], as in Figure 4.4.1 and Figure 4.4.2. The software helped facilitate passing the design into PCB so the only task left was to place each component as desired. On the PCB the top layer being the red one as in Figure 4.4.3, was focused on being only GND connections, the bottom layer being the blue one as in Figure 4.4.4 focused on VCC connections. The measurements of the PCB are 2190mil X 1720.065mil, and the way it would’ve looked if printing had been possible would be as Figure 4.4.5.



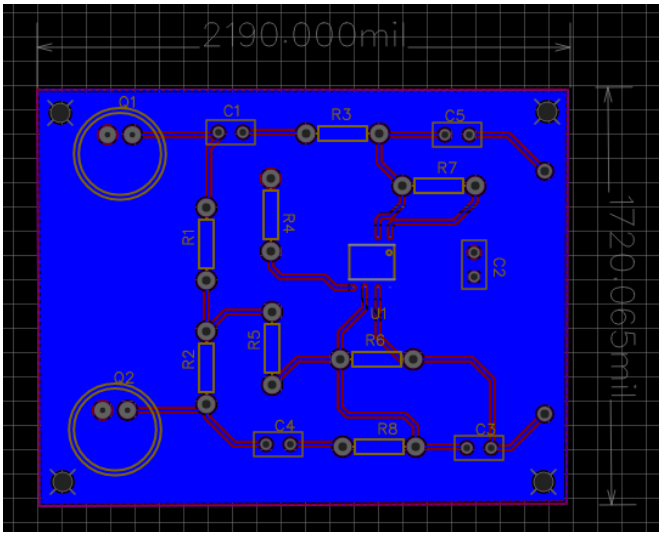
4.4.1 Microphone Circuit



4.4.2 Microphone PCB connections



4.4.3 Top Layer of Microphone PCB



4.4.4 Bottom Layer of Microphone PCB

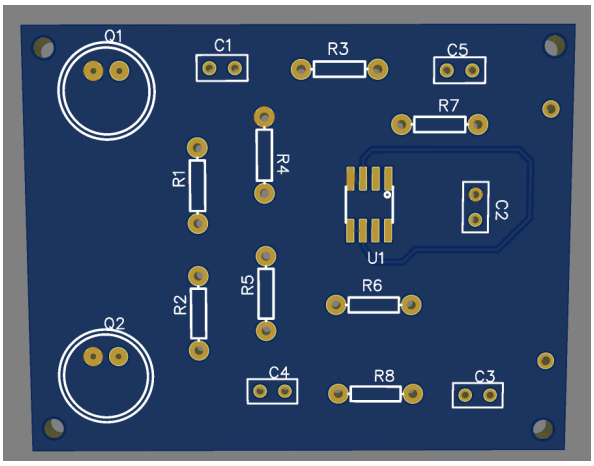


Figure 4.4.5 3D View of MIcrophone PCB

4.5 Drone Detector Cover

The cover that was used for the realization of this project was designed in solidworks (design software). It should be noted that we used the help of a tutorial to understand the basic functions of the software since we did not have the necessary experience to carry it out. We as a team opted for this design due to its practicality and that something so detailed was not needed, we only wanted that all the parts that integrate the drone detector were not mixed with each other but rather that each part had its space for future better or possible modifications.


Figure 4.5.1 3D Case Side View


Figure 4.5.2 3D Case Front View

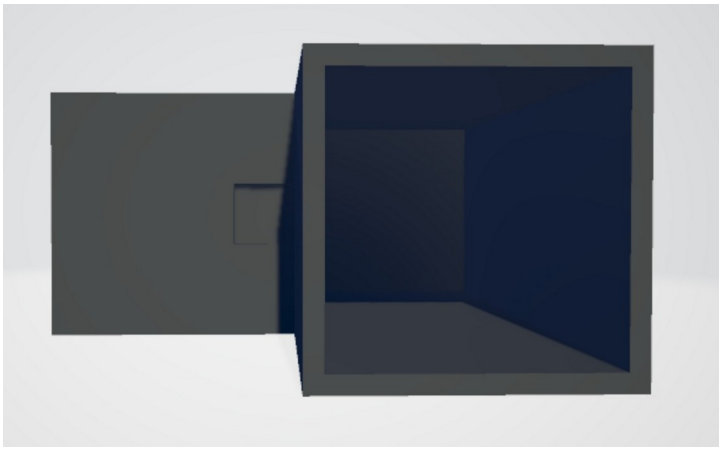


Figure 4.5.3 3D Case Rear View

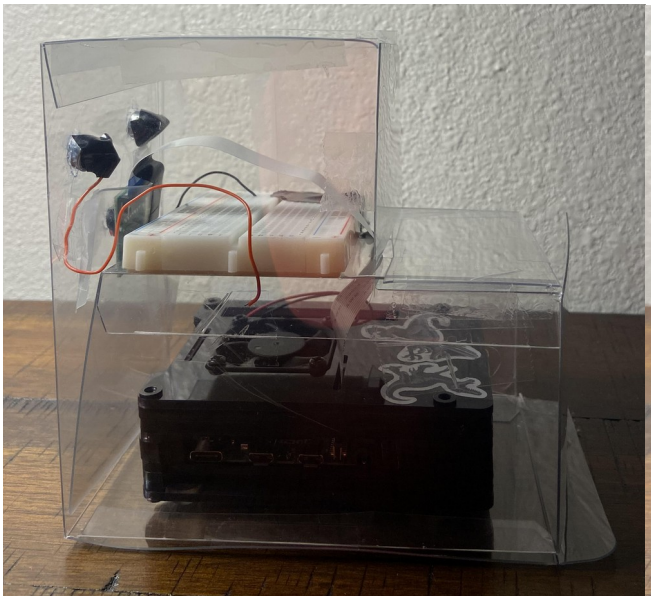


Figure 4.5.4 Prototype Case Side View

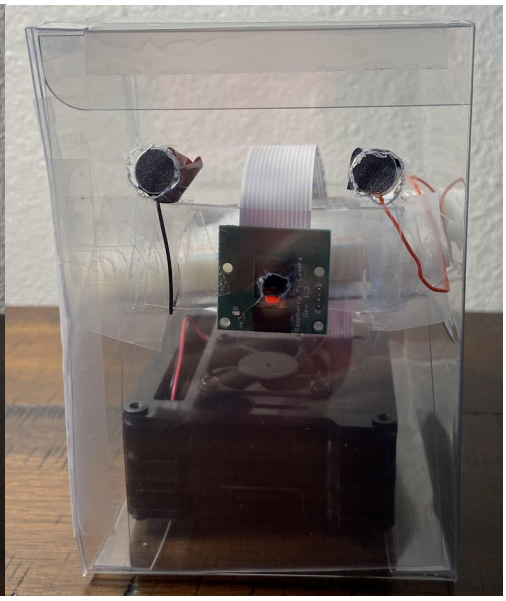


Figure 4.5.5 Prototype Case Front View

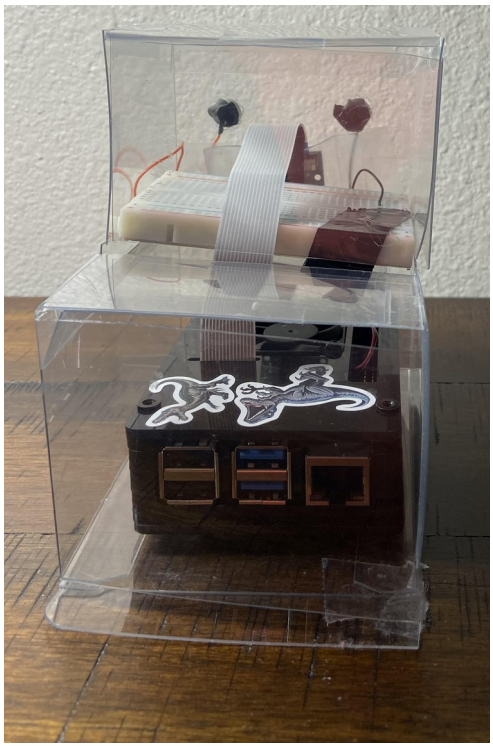


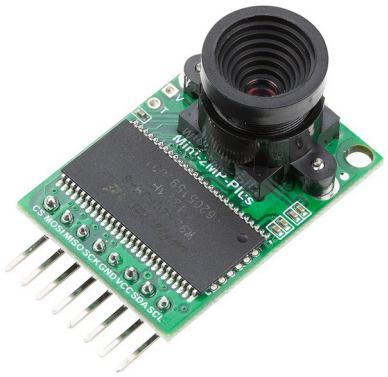
Figure 4.5.6 Prototype Case Rear View

5.0 Equipment / Fabrication / Software Needs

All the components used in this project for testing, and their specification related to the purpose of the detection of drones.

5.1 Cameras

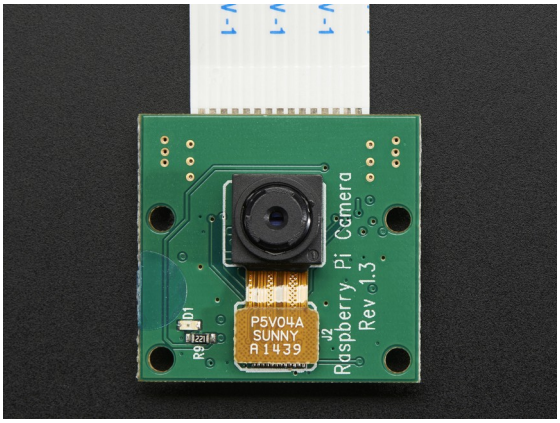
5.3.2.1 ArduCam 2MP oV2640 Mini Module SPI[17]



- 2MP image sensor OV2640
- M12 mount or CS mount lens holder with changeable lens options
- IR sensitive with proper lens combination
- I2C interface for the sensor configuration
- SPI interface for camera commands and data stream
- All IO ports are 5V/3.3V tolerant
- Support JPEG compression mode, single and multiple shoot mode, one time capture multiple read operation, burst read operation, low power mode and etc.
- Well mated with standard Arduino boards
- Provide open source code library for Arduino, STM32, Chipkit, Raspberry Pi,

  BeagleBone Black
- Small form of factor

5.3.2.2 Raspberry Pi Mini Camera 0V5647[18]



- High Definition video camera for Raspberry Pi Model A/B/B+
- 5 Megapixels sensor
- Capable of 2592 x 1944 pixel static image
- Supports 1080 p @ 30 fps, 720 p @ 60 fps and 640 x480 p 60/90 video recording, with 5MP OV5647 1080p webcam sensor.
- Contains Fixed Focus, where the sensor has a native resolution of 5 megapixel with OV5647 sensor in a fixed-focus lens.
- 15 cm flat ribbon cable to 15-pin MIPI Camera Serial Interface (CSI) connector
- CSI Interface, this interface uses the dedicated CSI interface, via the CSI bus, a higher bandwidth link which carries pixel data from the camera back to the processor, high data rates, and it exclusively carries pixel data.
- Integral IR filter
- Angel of View: 54 x 41 degrees
- Field of View: 2.0 x 1.33 m at 2 m
- Full-frame SLR lens equivalent: 35 mm
- Fixed Focus: 1 m to infinity

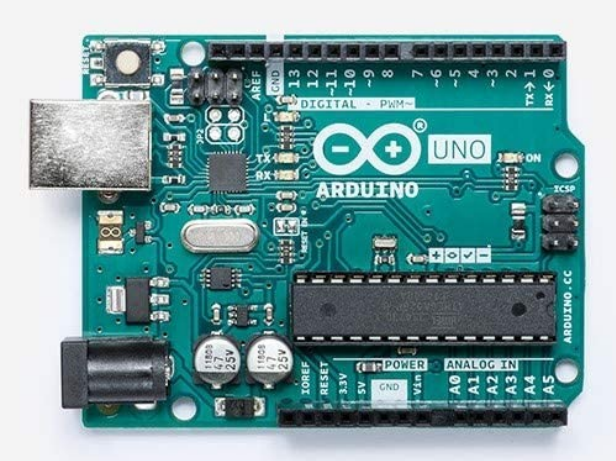
- Still picture resolution: 2592 x 1944
- Max video resolution: 1080p
- Max frame rate: 30fps
- Horizontal field of view: 53.50 +/- 0.13 degrees
- Vertical field of view: 41.41 +/- 0.11 degrees
- Focal ratio (F-Stop): 2.9

5.2 Electret Condenser Microphone EM-926[19]

- Impedance: Low Impedance
- Directivity: Omnidirectional
- Frequency: 30-16.000Hz
- Operation voltage: 1-10V
- Standard operation voltage: 1.5V
- Current consumption: Max 0.3mA
- Sensitivity reduction: Within -3dB at 1.1V
- S/N ratio: More than 40dB

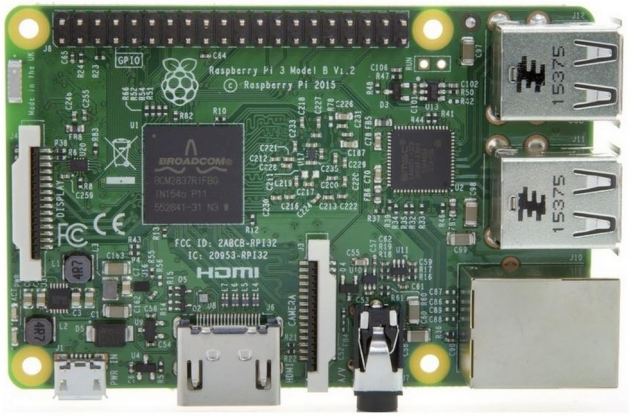
5.3 Arduino and Raspberry Pi

5.5.1 Arduino Uno[20]



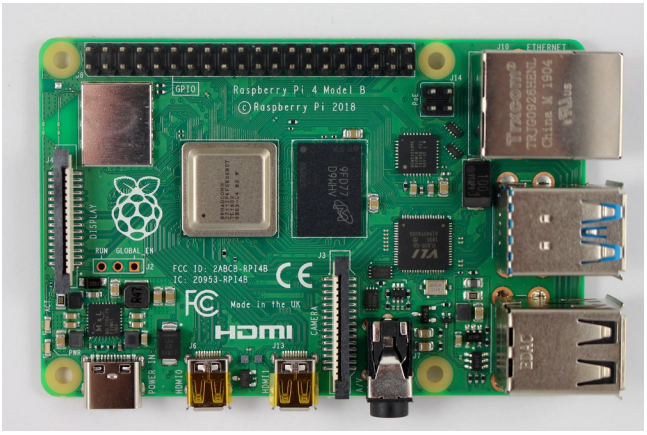
- ATmega328P microcontroller
- Input Voltage: 7-12V
- Digital I/O Pins: 14
- PWM Digital I/O Pins: 6
- Analog Input Pins: 6
- DC Current per I/O Pin 20mA
- DC Current for 3.3V Pin: 50mA
- Flash Memory: 32KB(ATmega328P) of which 0.5KB was used by bootloader)
- SRAM: 2KB(ATmega328P)
- EEPROM: 1KB(ATmega328P)
- Clock Speed: 16MHz

5.5.2 Raspberry Pi Model 3B[12]



- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 100 Base Ethernet
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A

5.5.3 Raspberry Pi 4 Model B[21]



- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz

- 1GB, 2GB or 4GB LPDDR4-3200 SDRAM (depending on model)
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 USB 3.0 ports; 2 USB 2.0 ports.
- Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards)
- 2 × micro-HDMI ports (up to 4kp60 supported)
- 2-lane MIPI DSI display port
- 2-lane MIPI CSI camera port
- 4-pole stereo audio and composite video port
- H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
- OpenGL ES 3.0 graphics
- Micro-SD card slot for loading operating system and data storage
- 5V DC via USB-C connector (minimum 3A*)
- 5V DC via GPIO header (minimum 3A*)
- Power over Ethernet (PoE) enabled (requires separate PoE HAT)
- Operating temperature: 0 – 50 degrees C ambient

6.0 ABET Criteria

This section focuses on the different economical aspects and expenses of the project, the impact of the same at the environmental level, different points of view of ethical and personal character and finally talks about the different laws that exist about the regulations of drones in the political scope.

6.1 Economic

As you can see below in the table, the budget so far is $83.79, including component parts available at UTRGV at no cost to students. Even though the $83.79 is a reasonable cost compared to drone detectors available in the market. Most are only available to large businesses or government agencies. Our Drone Detector could go on the market for an estimated cost of $400, thus taking into consideration the motive which was to protect the users privacy at an extremely affordable price compared to the drone detection systems that are currently available on the market and we could still be making double the profit. We then conclude that it is a reasonable and fair price.

Table 6.1.1

Item name	Item amount	Price
Arduino Camera 5 mp	1	8.79
Raspberry Pi Camera	1	12.99
Raspberry Pi 3	1	50
Raspberry Pi 4	1	62.00
Breadboard	1	0
Box	1	25
Total		83.79

6.2 Environmental

The drone detector uses a Raspberry Pi 4[12]and a Raspberry Pi Mini Camera 0V5647[22] which have ”undergone extensive compliance testing, and meets the following European standards” according to the Raspberry Pi Foundation. It follows the Electromagnetic Compatibility Directive (EMC) 2014/30/EU which prevents electronic devices interference with

other devices like TV sets, radios, washing machines or electrical power lines. The EMC helps in having the device not interfere with these types of electronic devices and, “ensures that electrical and electronic equipment does not generate, or is not affected by, electromagnetic disturbance.”[23]. It also has the Restriction of Hazardous Substances (RoHS) Directive 2011/65/EU which means that the Raspberry Pi and the Mini Camera 0V5647 are free of hazardous substances like lead, mercury and chromium. The RoHS[24] also indicates that the electric devices can’t be recycled, they need to be properly handled as specified with the Waste Electrical and Electronic Equipment Directive (WEEE) 2012/19/EU.

The Raspberry Pi is also authorized by the USA/FCC, having the 15C regulation being followed legally on the Part 15 Spread Spectrum Transmitter with a stated Frequency Range (MHZ) between 2402.0 to 2483.5, and output watts of 0.0037. Also the Digital Transmission System of the Raspberry Pi follow the 15C regulation with Frequency Range(MHZ) between 2402.0 to 2480.0 and output watts of 0.0025, and a second Frequency Range(MHZ) between 2412.0 to 2462.0 and output watts of 0.138. The Raspberry Pi also has a Flammability rating that adheres to UL94-V0 regulation, where UL(Underwriters Laboratories), “is an independent organisation in the United States to control and certificate product safety.”[25]. So UL94 is a test made on the material of the electric device and V0 is a flame rating where the test is done with a vertical burning of the device.

The ArduCam 2MP oV2640 Mini Module SPI Camera[6] follows the LS-4011(S mount) or LS-6018(CS mount) lenses specification. LS-4011 and LS-6018 both are the

components that make up the lens, LS-4011 being the lens itself and LS-6018 being the part that supports the lens and has a plastic component that makes up the zoom.

6.3 Ethical

Drones are becoming a very common toy for people to use for entertainment but as much fun as people can have, it has a negative side. There has been an increase in reports where people inside their homes notice that a drone is spying on them while they are in their bedroom. The problem here is that there are not enough measures to prevent this. Since spying is illegal, spying through a drone should be considered too but the laws vary depending on the state so depending on how high the drone is flying, it can be considered legal or illegal. So a stranger can definitely spy someone in their bedroom if they know the restrictions to be flying their drone legally and worse, the drone could even not be registered with the FAA,the Federal Aviation Administration, as they are supposed to when buying one so the FAA doesn't have enough control over the drones that people own. Buying a drone is a very easy, "The Rossen team even bought a few drones right off the shelf, paid for in cash, making it impossible to trace who owns them."[26] so it's very easy for a person to not be able to be found if their drone was found spying. There was a case in 2016 where a man in Orem, Utah[27] found a drone was spying outside his bathroom window, he then started chasing the drone with his car, caught it and found that it had recorded and taken pictures of other people too, and it had its safety lights covered with tape. Also they can even make the drone quieter so that it's harder to be noticed making the situation worse. As this case, there are more where people can't in most cases find justice because they can't find the owner of the drone, the laws in their state can be more in favor of the culprit because they can be

flying it in a legal way, and also the victims can't even damage the drone if they find it spying on them because it also can have very serious consequences for them and not for the drone's owner.

It is not possible that a person inside their home can't protect themselves if they find a drone spying through their bedroom bathroom window, and with laws varying in state, they can even be the ones at fault instead of the person spying. These people that do these unethical choices in spying on someone in their intimate space that is their bedroom or their bathroom is a very wrong thing. They are violating their intimate space and thus causing them unnecessary fears because it could happen again and they inside their homes can't avoid or prevent this and can't take any action if they don't get any proof of the drone so it could be identified if it's registered with the FAA and if it's not it's very difficult to find the person. So how can a person put trust in their own home when a person could very well be spying on them and be considered legal because of the undated laws that can in most cases protect the person spying more than the victim. A way to solve this ethical problem is through the drone detector that aims to help people feel safe again if they've had these types of situations happen to them or if they want to prevent it from happening to them.

Spying shouldn't be an acceptable thing to permit so if the states aren't moving fast enough updating the laws to protect the people inside their homes, there has to be a way to help them so with the drone detector, the ethical solution would be to aim to get proof of the drone, the detector will sense the drone and when detected it will take a picture. This picture will then be saved and even if the drone escapes, the victims can go to the police with it and an investigation can be started because it could also be happening to more people in their area. There are some ethical implications with the drone detector since it can be a target for hackers as cameras[28] so it has to have constant updates so that it can protect itself if someone tried to

hack it. Also the drone detector has to take into account the privacy of the neighboring houses because if it's protecting the owner's privacy but evading everyone else's, it isn't fulfilling the desired goal so it has to be specially placed. A concern about the drone detector hearing everything is something important to consider because people could think it works similar to popular products as Amazon Echo that is constantly hearing to detect keywords.In the case of the drone detector, it will never save any type of data because it isn't made to record anything since it's function is similar to a motion detector but with sound so it only focuses on a specific sound that comes from a drone. Also the drone detector has a limited amount of memory that will have the raspberry pi, it doesn't save any photos in a cloud or in any website so there shouldn't be a concern that photos could be used for other purposes, the photos it would save would only be temporary and it would be constantly deleting them to keep it from occupying all the memory and also because the photo's purpose is only to recognize if it's a drone or not.

6.4 Health & Safety

Since the availability of drones for personal and commercial use, there have been a lot of safety concerns regarding drones over whether they can be weaponized and in what way if this happened can they be prevented in harming people. In September 2014, there was a drone "attack" aimed to the German Chancellor Angela Merkel, this happened during a Christian Democratic Party and the "attacker" was a member of the German Pirate Party that did this as a "government surveillance protest"[29], and even though the drone wasn't a danger to anyone since no one got hurt, it raised concerns over how a drone could be used to attack people as they currently exist but these type of drones with weapons are used by the military. There have been

other cases where people have been hurt by a drone because it crashed and fell on a man's head as it happened in 2016 to a writer in Cape Town, South Africa [29], and others where multiple people have been hurt very badly like in 2013 where a drone that had been recording a festival in Spain "crashed into the stands, injuring several people in attendance"[29].

Recently there was a drone and missile attack that happened in Saudi Arabia, "The low-flying and relatively cheap drones and cruise missiles purported to have been used in Saturday's attack are a fairly new challenge that many nation states are not in fact prepared to counter."[30]Because of this there are challenges in being able to detect the most amount of drones because if the defense system a country can have can't detect cheap drones, it is a security risk for their citizens safety. In Figure 6.1, it can be noticed the amount of accidents that have been happening this year alone, and most accidents happened in an airport where a drone was detected very close to it, and in some cases a plane almost crashed with a drone.

Figure 6.1 World. This image shows a demographic of people who feel threatened by drones. The orange is the color of the demographic and the dark blue is the people who are not.

The safety of people is starting to be threatened by drones and if business are starting to use drones to make deliveries, there has to be serious measures to prevent drones in flying in restricted areas, and there already exists solutions where there can be jammers near those areas but in other places, if the drone is low price and considered for personal use, even then it can be a safety concern if the owner used if for malicious purposes. The FAA has been trying to remedy this safety problem with drone regulations but the problem is that if a drone can't be detected by a radar, how can these regulations be enforced? The solution is still currently being worked on with recent cases as the one in Saudi Arabia, so in the future there is hope that countries can have their radar system detect any type of drone.

For the drone detector, that uses a radar system, this problem could also be present where a specific type of drone can become undetectable but since it will also have a camera, this issue could be prevented because it would base its action in movement so the drone will have to be detected. The drone detector will be mainly used in houses so a safety concern of a weaponized drone is in a very low probability. The components being used for the drone detector aren't in any way harmful to people in relation to radiation if frequencies were used, and since it will be placed in an ideal place preferably on the roof of a house, it won't come in any way in contact with a person. Also the components that make up the drone detector will be in an insulated box so that if it rains, the components don't get damaged. The drone detector will also be very well secured so that there isn't any risk in it falling and injuring someone. Since it uses a raspberry pi, it doesn't need a high level of electricity since it only uses 5 volts. If in any way the drone detector got damaged, there wouldn't be any risk of a fire or hazard, the only outcome would be that the raspberry pi would stop working so it would have to be then replaced by a specialized person, not by the customer.

6.5 Political

There have been many incidents where drones invade a person's property and are used for illegal purposes. There have been incidents where they use drones to drop homemade bombs without it even being detected and leaving people wondering how that came about. And the best thing here would be to at least detect them and know what's causing the problem.

"Drones pose novel and difficult problems for law enforcement. They are widely available, lightly regulated and can be flown remotely by an operator far away from the crime scene."[31]

It's easy to get a hold of a drone, while purchasing it you are not required to register right there and then, the seller obviously believes you will register the drone on your own time. But there are many criminals out there who use these drones to their advantage such as stealing or stalking. It is hard to realize that there is a drone hovering over your lawn because most likely you won't be able to see it or don't really spend your time looking at the sky.

"In 2017, a Utah couple was charged with voyeurism for using a drone to spy on people in their bedrooms and bathrooms."[31]

"Drones are increasingly being used by criminals across the country, and local law enforcement agencies are often powerless to stop them."[31]

The drone detector based on the current laws, wouldn't be illegal because it will only detect the drone when it is present on the person's property it won't damage it in any way. Currently there are no laws that state specifically that a person can't damage a drone but they can still be charged with "ARSON, CRIMINAL MISCHIEF, AND OTHER PROPERTY DAMAGE OR DESTRUCTION" as stated in the penal code

"Sec. 28.04. RECKLESS DAMAGE OR DESTRUCTION.

(a) A person commits an offense if, without the effective consent of the owner, he recklessly damages or destroys property of the owner.

(b) An offense under this section is a Class C misdemeanor."[32]

7.0 Conclusions and Recommendations

As the number of drones used as a recreational device is increasing based on the increase in popularity, the drone detector was made to counteract the people that may use drones for malicious purposes. Since professional drone detectors are high in cost and are aimed to be used for commercial and military spaces, the drone detector done in this project is made to be at an affordable price aimed at people with an average income. The components used for this project being a camera and a microphone was the best option to use to stay at the price goal to sell the device. Since we first started using the Arduino, the module of the camera available for the Arduino does not go up to 60 fps and the Arduino modules do not have enough ram nor process speed to do live analysis of images. For this a Raspberry Pi 3, being a simple computer, has enough power with 1GB RAM and a dedicated processor, for the tasks needed for the drone detector. In addition it doesn't require much power, just 5v, since it was later throughout the project that the images had to be scaled at a certain size to not overload the Raspberry Pi, the newer model being Raspberry Pi 4 had to be used. With this, the camera, microphone and the image recognition will be used. The microphone to detect drones may lack accuracy because there are many sounds that have the same frequency as a drone blade like a weed eater. It can be a good support for the camera because it doesn't have the speed to instantly capture a drone when it is near the property. The reason why ultrasonic is not a good option is because it has a

maximum range of 12 inches so there isn't enough room to test moving objects, they have to be very close to be detected. Also it can't distinguish between objects detected so this approach wasn't right for the device. The reason why a microphone is better than an antenna is because creating waves is too expensive to run it 24/7 or even in short times. It will be too expensive to get the correct antennas for the project. Also receiving the signal from a drone will be an extensive process of going to the frequency 2.4 Ghz. In the open world it is easy to use this frequency because it is not in nature but in a neighborhood, the wifi signals will interfere with the detection and this may affect the results of detecting drones.

In the future, the drone detector could be improved in the size of the dataset since throughout the year it was started from zero, if there had been more time, from the quantity of pictures it ended being trained, more can still be added as newer models are made so that it can stay updated with the latest shapes, sizes, and colors of the drones. The way the drone detector notifies the user can also be improved because it only sends an sms message and an email but it can be taken a step farther and be made into an app where the user could see the photo taken from the drone detector instead of viewing it through the sms and email. The app could also include a way to view live the drone if spotted but this would need to be available using the cloud. The microphone can also be improved because due to the COVID-19, the PCB couldn't be printed so the circuit stayed in the protoboard. The way the drone detector is mounted to the house can be changed to have the option of moving if a drone is detected because the way we ended up having it was static in a strategic angle to take advantage of the camera used but it could be more precise and have step motors to move a certain amount without damaging any cables that are used for the camera and other parts of the drone detector.

8.0 References

[1] B. Hubbard, P. Karasz, and S. Reed, “Two Major Saudi Oil Installations Hit by Drone Strike, and U.S. Blames Iran,” 14-Sep-2019. [Online]. Available: https://www.nytimes.com/2019/09/14/world/middleeast/saudi-arabia-refineries-drone-attack.html. [Accessed: 03-Apr-2020].

[2] N. Bilton, “When Your Neighbor’s Drone Pays an Unwelcome Visit,” 27-Jan-2016. [Online]. Available: https://www.nytimes.com/2016/01/28/style/neighbors-drones-invade-privacy.html. [Accessed: 03-Apr-2020].

[3] “Drone Laws in Texas (2020) - UAV Coach,” *UAV Coach*. [Online]. Available: https://uavcoach.com/drone-laws-texas/. [Accessed: 03-Apr-2020].

[4] DJI, “Drone Laws and Regulations: Read Now to Avoid Big Fines and Legal Woes,” *DJI Guides*, 23-Oct-2017. [Online]. Available: https://store.dji.com/guides/drone-laws/. [Accessed: 02-Dec-2019].

[5] R. Dietz, “New Single-Family Home Size: First Quarter 2019 Data,” *Eye On Housing*, 21-May-2019. [Online]. Available: http://eyeonhousing.org/2019/05/new-single-family-home-size-first-quarter-2019-data/. [Accessed: 03-Apr-2020].
[6] “ArduCam,” *ArduCam/Arduino*. [Online]. Available: https://github.com/ArduCAM/Arduino. [Accessed: 22-Nov-2019].
[7] S. Saha, “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way,” *Medium*, 15-Dec-2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Accessed: 03-Apr-2020].
[8] “Home - Keras Documentation.” [Online]. Available: https://keras.io. [Accessed: 03-Apr-2020].
[9] *How loud are drones? | Sound Level metering | Testing noise level of DJI’s Mavic and Inspire drone*. 2017.
[10] “Camera Module - Raspberry Pi Documentation.” [Online]. Available: https://www.raspberrypi.org/documentation/hardware/camera/. [Accessed: 07-Apr-2020].
[11] “Free CAD Designs, Files & 3D Models | The GrabCAD Community Library.” [Online]. Available: https://grabcad.com/library/raspberry-pi-camera-1. [Accessed: 07-Apr-2020].
[12] “Raspberry Pi 3 Model B,” *Raspberry Pi*. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. [Accessed: 21-Nov-2019].
[13] *Machine Learning Image Processing using Python, OpenCV, Keras and TensorFlow*. 2018.
[14] S. Basulto, “7-Using our pre-trained model,” *GitHub*. [Online]. Available: https://github.com/rmotr/ml-workshop-image-recognition. [Accessed: 29-Nov-2019].
[15] “Twilio | Login.” [Online]. Available: https://www.twilio.com/login?g=%2Fconsole%3F&t=2b1c98334b25c1a785ef15b6556396290e3c704a9b57fc40687cbccd79c46a8c. [Accessed: 06-Apr-2020].
[16] “EasyEDA - Online PCB design & circuit simulator.” [Online]. Available: https://easyeda.com. [Accessed: 08-Apr-2020].
[17] “Arducam 2MP OV2640 Mini Module SPI Camera for Arduino UNO Mega2560 Board - Arducam,” *Arducam*. [Online]. Available: https://www.arducam.com/product/arducam-2mp-spi-camera-b0067-arduino/. [Accessed: 06-Apr-2020].
[18] “Amazon.com: Raspberry Pi Mini Camera Video Module 5 Megapixels 1080p Sensor OV5647 Webcam for Raspberry Pi Model A/B/A+/B+, Pi 2B and Raspberry Pi 3B, Pi 3 B+, Raspberry Pi 4 B: Home Improvement.” [Online]. Available: https://www.amazon.com/Raspberry-Camera-Module-Megapixels-Sensor/dp/B07L82XBNM/ref =sr_1_3?crid=I5AGRPFVOSK9&keywords=raspberry pi3 camera&qid=1574184225&sprefix=raspberry p,aps,200&sr=8-3. [Accessed: 03-Apr-2020].
[19] “Omnidirectional Electret Condenser Microphone.”
[20] “Arduino Uno Rev3 | Arduino Official Store.” [Online]. Available: https://store.arduino.cc/usa/arduino-uno-rev3. [Accessed: 03-Apr-2020].
[21] “Raspberry Pi 4 Model B specifications – Raspberry Pi.” [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/. [Accessed: 06-Apr-2020].
[22] “[No title].” [Online]. Available: https://www.raspberrypi.org/documentation/hardware/raspberrypi/compliance/rpi_DOC_Camera_CE.pdf. [Accessed: 04-Apr-2020].
[23] Anonymous, “Electromagnetic Compatibility (EMC) Directive - Internal Market, Industry, Entrepreneurship and SMEs - European Commission,” *Internal Market, Industry, Entrepreneurship and SMEs - European Commission*, 05-Jul-2016. [Online]. Available: https://ec.europa.eu/growth/sectors/electrical-engineering/emc-directive_en. [Accessed: 03-Apr-2020].
[24] “EU RoHS 2 (Directive 2011/65/EU).” [Online]. Available: https://www.chemsafetypro.com/Topics/Restriction/EU_RoHS_2_EU_RoHS_Directive.html. [Accessed: 03-Apr-2020].
[25] “What Does The UL94 Flammability Rating Mean? - ETS Cable Components,” *ETS Cable*

Components, 08-May-2014. [Online]. Available: https://www.etscablecomponents.com/2014/05/ul94-flammability-rating-mean/. [Accessed: 03-Apr-2020].
[26] "Peeping drones could be spying on you in your own home," *TODAY.com*, 09-May-2018. [Online]. Available: https://www.today.com/money/peeping-drones-could-be-spying-you-your-own-home-t128590. [Accessed: 03-Apr-2020].
[27] B. D. Wells, "Orem police say they've identified peeping drone operator," *KSTU*, 06-Dec-2016. [Online]. Available: https://www.fox13now.com/2016/12/05/orem-police-say-theyve-identified-peeping-drone-operator. [Accessed: 03-Apr-2020].
[28] "'Security' Cameras Are Dry Powder for Hackers. Here's Why," *Fortune*, 19-Sep-2019. [Online]. Available: https://fortune.com/2019/09/19/security-cameras-are-dry-powder-for-hackers-heres-why/. [Accessed: 03-Apr-2020].
[29] C. Forrest, "17 drone disasters that show why the FAA hates drones," *TechRepublic*, 13-Jun-2018. [Online]. Available: https://www.techrepublic.com/article/12-drone-disasters-that-show-why-the-faa-hates-drones/. [Accessed: 03-Apr-2020].
[30] "Website." [Online]. Available: https://www.cnbc.com/2019/09/19/how-saudi-arabia-failed-to-protect-itself-from-drones-missile -attacks.html. [Accessed: 03-Apr-2020].
[31] V. Swales, "Drones Used in Crime Fly Under the Law's Radar," 03-Nov-2019. [Online]. Available: https://www.nytimes.com/2019/11/03/us/drones-crime.html. [Accessed: 03-Apr-2020].
[32] "PENAL CODE CHAPTER 28. ARSON, CRIMINAL MISCHIEF, AND OTHER PROPERTY DAMAGE OR DESTRUCTION." [Online]. Available: https://statutes.capitol.texas.gov/Docs/PE/htm/PE.28.htm. [Accessed: 03-Apr-2020].

Appendices

Arduino Camera Code from the source[4]

```
// ArduCAM Mini demo (C)2017 Lee
// Web: http://www.ArduCAM.com
// This program is a demo of how to use most of the functions
// of the library with ArduCAM Mini camera, and can run on any Arduino platform.
// This demo was made for ArduCAM_Mini_5MP_Plus.
// It needs to be used in combination with PC software.
// It can test OV2640 functions
// This program requires the ArduCAM V4.0.0 (or later) library and
ArduCAM_Mini_5MP_Plus
// and use Arduino IDE 1.6.8 compiler or above
#include <Wire.h>
#include <ArduCAM.h>
#include <SPI.h>
#include "memorysaver.h"
//This demo can only work on OV2640_MINI_2MP platform.
#if !(defined OV2640_MINI_2MP)
  #error Please select the hardware platform and camera module in the
../libraries/ArduCAM/memorysaver.h file
#endif
#define BMPIMAGEOFFSET 66
const char bmp_header[BMPIMAGEOFFSET] PROGMEM =
{
  0x42, 0x4D, 0x36, 0x58, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x42, 0x00, 0x00, 0x00, 0x28,
0x00,
  0x00, 0x00, 0x40, 0x01, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00, 0x01, 0x00, 0x10, 0x00, 0x03,
0x00,
  0x00, 0x00, 0x00, 0x58, 0x02, 0x00, 0xC4, 0x0E, 0x00, 0x00, 0xC4, 0x0E, 0x00, 0x00,
```

```
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xF8, 0x00, 0x00, 0xE0, 0x07, 0x00, 0x00, 0x1F,
0x00,
  0x00, 0x00
};
// set pin 7 as the slave select for the digital pot:
const int CS = 7;
bool is_header = false;
int mode = 0;
uint8_t start_capture = 0;
#if defined (OV2640_MINI_2MP)
  ArduCAM myCAM( OV2640, CS );
#else
  ArduCAM myCAM( OV5642, CS );
#endif
uint8_t read_fifo_burst(ArduCAM myCAM);
void setup() {
// put your setup code here, to run once:
uint8_t vid, pid;
uint8_t temp;
#if defined(__SAM3X8E__)
  Wire1.begin();
  Serial.begin(115200);
#else
  Wire.begin();
  Serial.begin(921600);
#endif
Serial.println(F("ACK CMD ArduCAM Start! END"));
// set the CS as an output:
pinMode(CS, OUTPUT);
digitalWrite(CS, HIGH);
// initialize SPI:
SPI.begin();
  //Reset the CPLD
myCAM.write_reg(0x07, 0x80);
delay(100);
myCAM.write_reg(0x07, 0x00);
delay(100);
while(1){
  //Check if the ArduCAM SPI bus is OK
```

```
  myCAM.write_reg(ARDUCHIP_TEST1, 0x55);
  temp = myCAM.read_reg(ARDUCHIP_TEST1);
  if (temp != 0x55){
    Serial.println(F("ACK CMD SPI interface Error! END"));
    delay(1000);continue;
  }else{
    Serial.println(F("ACK CMD SPI interface OK. END"));break;
  }
}

#if defined (OV2640_MINI_2MP)
  while(1){
    //Check if the camera module type is OV2640
    myCAM.wrSensorReg8_8(0xff, 0x01);
    myCAM.rdSensorReg8_8(OV2640_CHIPID_HIGH, &vid);
    myCAM.rdSensorReg8_8(OV2640_CHIPID_LOW, &pid);
    if ((vid != 0x26 ) && (( pid != 0x41 ) || ( pid != 0x42 ))){
      Serial.println(F("ACK CMD Can't find OV2640 module! END"));
      delay(1000);continue;
    }
    else{
      Serial.println(F("ACK CMD OV2640 detected. END"));break;
    }
  }
#else
  while(1){
    //Check if the camera module type is OV5642
    myCAM.wrSensorReg16_8(0xff, 0x01);
    myCAM.rdSensorReg16_8(OV5642_CHIPID_HIGH, &vid);
    myCAM.rdSensorReg16_8(OV5642_CHIPID_LOW, &pid);
    if((vid != 0x56) || (pid != 0x42)){
      Serial.println(F("ACK CMD Can't find OV5642 module! END"));
      delay(1000);continue;
    }
    else{
      Serial.println(F("ACK CMD OV5642 detected. END"));break;
    }
  }
#endif
//Change to JPEG capture mode and initialize the OV5642 module
```

```
myCAM.set_format(JPEG);
myCAM.InitCAM();
#if defined (OV2640_MINI_2MP)
  myCAM.OV2640_set_JPEG_size(OV2640_320x240);
#else
  myCAM.write_reg(ARDUCHIP_TIM, VSYNC_LEVEL_MASK);   //VSYNC is active
HIGH
  myCAM.OV5642_set_JPEG_size(OV5642_320x240);
#endif
delay(1000);
myCAM.clear_fifo_flag();
#if !(defined (OV2640_MINI_2MP))
myCAM.write_reg(ARDUCHIP_FRAMES,0x00);
#endif
}
void loop() {
// put your main code here, to run repeatedly:
uint8_t temp = 0xff, temp_last = 0;
bool is_header = false;
if (Serial.available())
{
  temp = Serial.read();
  switch (temp)
  {
    case 0:
      myCAM.OV2640_set_JPEG_size(OV2640_160x120);delay(1000);
      Serial.println(F("ACK CMD switch to OV2640_160x120 END"));
     temp = 0xff;
    break;
    case 1:
      myCAM.OV2640_set_JPEG_size(OV2640_176x144);delay(1000);
      Serial.println(F("ACK CMD switch to OV2640_176x144 END"));
    temp = 0xff;
    break;
    case 2:
      myCAM.OV2640_set_JPEG_size(OV2640_320x240);delay(1000);
      Serial.println(F("ACK CMD switch to OV2640_320x240 END"));
    temp = 0xff;
    break;
    case 3:
```

```
 myCAM.OV2640_set_JPEG_size(OV2640_352x288);delay(1000);
 Serial.println(F("ACK CMD switch to OV2640_352x288 END"));
temp = 0xff;
 break;
 case 4:
   myCAM.OV2640_set_JPEG_size(OV2640_640x480);delay(1000);
   Serial.println(F("ACK CMD switch to OV2640_640x480 END"));
 temp = 0xff;
 break;
 case 5:
 myCAM.OV2640_set_JPEG_size(OV2640_800x600);delay(1000);
 Serial.println(F("ACK CMD switch to OV2640_800x600 END"));
 temp = 0xff;
 break;
 case 6:
  myCAM.OV2640_set_JPEG_size(OV2640_1024x768);delay(1000);
  Serial.println(F("ACK CMD switch to OV2640_1024x768 END"));
 temp = 0xff;
 break;
 case 7:
 myCAM.OV2640_set_JPEG_size(OV2640_1280x1024);delay(1000);
 Serial.println(F("ACK CMD switch to OV2640_1280x1024 END"));
 temp = 0xff;
 break;
 case 8:
 myCAM.OV2640_set_JPEG_size(OV2640_1600x1200);delay(1000);
 Serial.println(F("ACK CMD switch to OV2640_1600x1200 END"));
  temp = 0xff;
 break;
 case 0x10:
 mode = 1;
 temp = 0xff;
 start_capture = 1;
 Serial.println(F("ACK CMD CAM start single shoot. END"));
 break;
 case 0x11:
 temp = 0xff;
 myCAM.set_format(JPEG);
 myCAM.InitCAM();
 #if !(defined (OV2640_MINI_2MP))
```

```
  myCAM.set_bit(ARDUCHIP_TIM, VSYNC_LEVEL_MASK);
  #endif
  break;
  case 0x20:
  mode = 2;
  temp = 0xff;
  start_capture = 2;
  Serial.println(F("ACK CMD CAM start video streaming. END"));
  break;
  case 0x30:
  mode = 3;
  temp = 0xff;
  start_capture = 3;
  Serial.println(F("ACK CMD CAM start single shoot. END"));
  break;
  case 0x31:
  temp = 0xff;
  myCAM.set_format(BMP);
  myCAM.InitCAM();
  #if !(defined (OV2640_MINI_2MP))
  myCAM.clear_bit(ARDUCHIP_TIM, VSYNC_LEVEL_MASK);
  #endif
  myCAM.wrSensorReg16_8(0x3818, 0x81);
  myCAM.wrSensorReg16_8(0x3621, 0xA7);
  break;
  case 0x40:
  myCAM.OV2640_set_Light_Mode(Auto);temp = 0xff;
  Serial.println(F("ACK CMD Set to Auto END"));break;
   case 0x41:
  myCAM.OV2640_set_Light_Mode(Sunny);temp = 0xff;
  Serial.println(F("ACK CMD Set to Sunny END"));break;
   case 0x42:
  myCAM.OV2640_set_Light_Mode(Cloudy);temp = 0xff;
  Serial.println(F("ACK CMD Set to Cloudy END"));break;
   case 0x43:
  myCAM.OV2640_set_Light_Mode(Office);temp = 0xff;
  Serial.println(F("ACK CMD Set to Office END"));break;
 case 0x44:
  myCAM.OV2640_set_Light_Mode(Home);   temp = 0xff;
 Serial.println(F("ACK CMD Set to Home END"));break;
```

```
case 0x50:
 myCAM.OV2640_set_Color_Saturation(Saturation2); temp = 0xff;
  Serial.println(F("ACK CMD Set to Saturation+2 END"));break;
case 0x51:
  myCAM.OV2640_set_Color_Saturation(Saturation1); temp = 0xff;
  Serial.println(F("ACK CMD Set to Saturation+1 END"));break;
case 0x52:
 myCAM.OV2640_set_Color_Saturation(Saturation0); temp = 0xff;
  Serial.println(F("ACK CMD Set to Saturation+0 END"));break;
 case 0x53:
 myCAM. OV2640_set_Color_Saturation(Saturation_1); temp = 0xff;
  Serial.println(F("ACK CMD Set to Saturation-1 END"));break;
 case 0x54:
  myCAM.OV2640_set_Color_Saturation(Saturation_2); temp = 0xff;
  Serial.println(F("ACK CMD Set to Saturation-2 END"));break;
case 0x60:
 myCAM.OV2640_set_Brightness(Brightness2); temp = 0xff;
  Serial.println(F("ACK CMD Set to Brightness+2 END"));break;
case 0x61:
  myCAM.OV2640_set_Brightness(Brightness1); temp = 0xff;
  Serial.println(F("ACK CMD Set to Brightness+1 END"));break;
case 0x62:
 myCAM.OV2640_set_Brightness(Brightness0); temp = 0xff;
  Serial.println(F("ACK CMD Set to Brightness+0 END"));break;
 case 0x63:
 myCAM. OV2640_set_Brightness(Brightness_1); temp = 0xff;
  Serial.println(F("ACK CMD Set to Brightness-1 END"));break;
 case 0x64:
  myCAM.OV2640_set_Brightness(Brightness_2); temp = 0xff;
  Serial.println(F("ACK CMD Set to Brightness-2 END"));break;
 case 0x70:
   myCAM.OV2640_set_Contrast(Contrast2);temp = 0xff;
  Serial.println(F("ACK CMD Set to Contrast+2 END"));break;
 case 0x71:
   myCAM.OV2640_set_Contrast(Contrast1);temp = 0xff;
  Serial.println(F("ACK CMD Set to Contrast+1 END"));break;
  case 0x72:
   myCAM.OV2640_set_Contrast(Contrast0);temp = 0xff;
  Serial.println(F("ACK CMD Set to Contrast+0 END"));break;
 case 0x73:
```

```
      myCAM.OV2640_set_Contrast(Contrast_1);temp = 0xff;
     Serial.println(F("ACK CMD Set to Contrast-1 END"));break;
   case 0x74:
      myCAM.OV2640_set_Contrast(Contrast_2);temp = 0xff;
     Serial.println(F("ACK CMD Set to Contrast-2 END"));break;
   case 0x80:
    myCAM.OV2640_set_Special_effects(Antique);temp = 0xff;
    Serial.println(F("ACK CMD Set to Antique END"));break;
   case 0x81:
    myCAM.OV2640_set_Special_effects(Bluish);temp = 0xff;
    Serial.println(F("ACK CMD Set to Bluish END"));break;
   case 0x82:
    myCAM.OV2640_set_Special_effects(Greenish);temp = 0xff;
    Serial.println(F("ACK CMD Set to Greenish END"));break;
   case 0x83:
    myCAM.OV2640_set_Special_effects(Reddish);temp = 0xff;
    Serial.println(F("ACK CMD Set to Reddish END"));break;
   case 0x84:
    myCAM.OV2640_set_Special_effects(BW);temp = 0xff;
    Serial.println(F("ACK CMD Set to BW END"));break;
  case 0x85:
    myCAM.OV2640_set_Special_effects(Negative);temp = 0xff;
    Serial.println(F("ACK CMD Set to Negative END"));break;
  case 0x86:
    myCAM.OV2640_set_Special_effects(BWnegative);temp = 0xff;
    Serial.println(F("ACK CMD Set to BWnegative END"));break;
   case 0x87:
    myCAM.OV2640_set_Special_effects(Normal);temp = 0xff;
    Serial.println(F("ACK CMD Set to Normal END"));break;
  }
}
if (mode == 1)
{
  if (start_capture == 1)
  {
    myCAM.flush_fifo();
    myCAM.clear_fifo_flag();
    //Start capture
    myCAM.start_capture();
    start_capture = 0;
```

```
  }
  if (myCAM.get_bit(ARDUCHIP_TRIG, CAP_DONE_MASK))
  {
    Serial.println(F("ACK CMD CAM Capture Done. END"));delay(50);
    read_fifo_burst(myCAM);
    //Clear the capture done flag
    myCAM.clear_fifo_flag();
  }
}
else if (mode == 2)
{
  while (1)
  {
    temp = Serial.read();
    if (temp == 0x21)
    {
      start_capture = 0;
      mode = 0;
      Serial.println(F("ACK CMD CAM stop video streaming. END"));
      break;
    }
    switch (temp)
    {
       case 0x40:
    myCAM.OV2640_set_Light_Mode(Auto);temp = 0xff;
    Serial.println(F("ACK CMD Set to Auto END"));break;
     case 0x41:
    myCAM.OV2640_set_Light_Mode(Sunny);temp = 0xff;
    Serial.println(F("ACK CMD Set to Sunny END"));break;
     case 0x42:
    myCAM.OV2640_set_Light_Mode(Cloudy);temp = 0xff;
    Serial.println(F("ACK CMD Set to Cloudy END"));break;
     case 0x43:
    myCAM.OV2640_set_Light_Mode(Office);temp = 0xff;
    Serial.println(F("ACK CMD Set to Office END"));break;
   case 0x44:
    myCAM.OV2640_set_Light_Mode(Home);   temp = 0xff;
   Serial.println(F("ACK CMD Set to Home END"));break;
   case 0x50:
    myCAM.OV2640_set_Color_Saturation(Saturation2); temp = 0xff;
```

```
  Serial.println(F("ACK CMD Set to Saturation+2 END"));break;
case 0x51:
  myCAM.OV2640_set_Color_Saturation(Saturation1); temp = 0xff;
  Serial.println(F("ACK CMD Set to Saturation+1 END"));break;
case 0x52:
 myCAM.OV2640_set_Color_Saturation(Saturation0); temp = 0xff;
  Serial.println(F("ACK CMD Set to Saturation+0 END"));break;
 case 0x53:
 myCAM. OV2640_set_Color_Saturation(Saturation_1); temp = 0xff;
  Serial.println(F("ACK CMD Set to Saturation-1 END"));break;
 case 0x54:
  myCAM.OV2640_set_Color_Saturation(Saturation_2); temp = 0xff;
  Serial.println(F("ACK CMD Set to Saturation-2 END"));break;
case 0x60:
 myCAM.OV2640_set_Brightness(Brightness2); temp = 0xff;
  Serial.println(F("ACK CMD Set to Brightness+2 END"));break;
case 0x61:
  myCAM.OV2640_set_Brightness(Brightness1); temp = 0xff;
  Serial.println(F("ACK CMD Set to Brightness+1 END"));break;
case 0x62:
 myCAM.OV2640_set_Brightness(Brightness0); temp = 0xff;
  Serial.println(F("ACK CMD Set to Brightness+0 END"));break;
 case 0x63:
 myCAM. OV2640_set_Brightness(Brightness_1); temp = 0xff;
  Serial.println(F("ACK CMD Set to Brightness-1 END"));break;
 case 0x64:
  myCAM.OV2640_set_Brightness(Brightness_2); temp = 0xff;
  Serial.println(F("ACK CMD Set to Brightness-2 END"));break;
 case 0x70:
   myCAM.OV2640_set_Contrast(Contrast2);temp = 0xff;
  Serial.println(F("ACK CMD Set to Contrast+2 END"));break;
 case 0x71:
   myCAM.OV2640_set_Contrast(Contrast1);temp = 0xff;
  Serial.println(F("ACK CMD Set to Contrast+1 END"));break;
  case 0x72:
   myCAM.OV2640_set_Contrast(Contrast0);temp = 0xff;
  Serial.println(F("ACK CMD Set to Contrast+0 END"));break;
 case 0x73:
   myCAM.OV2640_set_Contrast(Contrast_1);temp = 0xff;
  Serial.println(F("ACK CMD Set to Contrast-1 END"));break;
```

```
 case 0x74:
   myCAM.OV2640_set_Contrast(Contrast_2);temp = 0xff;
  Serial.println(F("ACK CMD Set to Contrast-2 END"));break;
 case 0x80:
  myCAM.OV2640_set_Special_effects(Antique);temp = 0xff;
  Serial.println(F("ACK CMD Set to Antique END"));break;
 case 0x81:
  myCAM.OV2640_set_Special_effects(Bluish);temp = 0xff;
  Serial.println(F("ACK CMD Set to Bluish END"));break;
 case 0x82:
  myCAM.OV2640_set_Special_effects(Greenish);temp = 0xff;
  Serial.println(F("ACK CMD Set to Greenish END"));break;
 case 0x83:
  myCAM.OV2640_set_Special_effects(Reddish);temp = 0xff;
  Serial.println(F("ACK CMD Set to Reddish END"));break;
 case 0x84:
  myCAM.OV2640_set_Special_effects(BW);temp = 0xff;
  Serial.println(F("ACK CMD Set to BW END"));break;
case 0x85:
  myCAM.OV2640_set_Special_effects(Negative);temp = 0xff;
  Serial.println(F("ACK CMD Set to Negative END"));break;
case 0x86:
  myCAM.OV2640_set_Special_effects(BWnegative);temp = 0xff;
  Serial.println(F("ACK CMD Set to BWnegative END"));break;
 case 0x87:
  myCAM.OV2640_set_Special_effects(Normal);temp = 0xff;
  Serial.println(F("ACK CMD Set to Normal END"));break;
}
  if (start_capture == 2)
  {
    myCAM.flush_fifo();
    myCAM.clear_fifo_flag();
    //Start capture
    myCAM.start_capture();
    start_capture = 0;
  }
  if (myCAM.get_bit(ARDUCHIP_TRIG, CAP_DONE_MASK))
  {
    uint32_t length = 0;
    length = myCAM.read_fifo_length();
```

```
      if ((length >= MAX_FIFO_SIZE) | (length == 0))
      {
        myCAM.clear_fifo_flag();
        start_capture = 2;
        continue;
      }
      myCAM.CS_LOW();
      myCAM.set_fifo_burst();//Set fifo burst mode
      temp =  SPI.transfer(0x00);
      length --;
      while ( length-- )
      {
        temp_last = temp;
        temp =  SPI.transfer(0x00);
        if (is_header == true)
        {
          Serial.write(temp);
        }
        else if ((temp == 0xD8) & (temp_last == 0xFF))
        {
          is_header = true;
          Serial.println(F("ACK IMG END"));
          Serial.write(temp_last);
          Serial.write(temp);
        }
        if ( (temp == 0xD9) && (temp_last == 0xFF) ) //If find the end ,break while,
        break;
        delayMicroseconds(15);
      }
      myCAM.CS_HIGH();
      myCAM.clear_fifo_flag();
      start_capture = 2;
      is_header = false;
    }
  }
}
else if (mode == 3)
{
  if (start_capture == 3)
  {
```

```
  //Flush the FIFO
  myCAM.flush_fifo();
  myCAM.clear_fifo_flag();
  //Start capture
  myCAM.start_capture();
  start_capture = 0;
 }
 if (myCAM.get_bit(ARDUCHIP_TRIG, CAP_DONE_MASK))
 {
  Serial.println(F("ACK CMD CAM Capture Done. END"));delay(50);
  uint8_t temp, temp_last;
  uint32_t length = 0;
  length = myCAM.read_fifo_length();
  if (length >= MAX_FIFO_SIZE )
  {
   Serial.println(F("ACK CMD Over size. END"));
   myCAM.clear_fifo_flag();
   return;
  }
  if (length == 0 ) //0 kb
  {
   Serial.println(F("ACK CMD Size is 0. END"));
   myCAM.clear_fifo_flag();
   return;
  }
  myCAM.CS_LOW();
  myCAM.set_fifo_burst();//Set fifo burst mode

  Serial.write(0xFF);
  Serial.write(0xAA);
  for (temp = 0; temp < BMPIMAGEOFFSET; temp++)
  {
   Serial.write(pgm_read_byte(&bmp_header[temp]));
  }
  //for old version, enable it else disable
 // SPI.transfer(0x00);
  char VH, VL;
  int i = 0, j = 0;
  for (i = 0; i < 240; i++)
  {
```

```
      for (j = 0; j < 320; j++)
      {
        VH = SPI.transfer(0x00);;
        VL = SPI.transfer(0x00);;
        Serial.write(VL);
        delayMicroseconds(12);
        Serial.write(VH);
        delayMicroseconds(12);
      }
    }
    Serial.write(0xBB);
    Serial.write(0xCC);
    myCAM.CS_HIGH();
    //Clear the capture done flag
    myCAM.clear_fifo_flag();
  }
}
}
uint8_t read_fifo_burst(ArduCAM myCAM)
{
  uint8_t temp = 0, temp_last = 0;
  uint32_t length = 0;
  length = myCAM.read_fifo_length();
  Serial.println(length, DEC);
  if (length >= MAX_FIFO_SIZE) //512 kb
  {
    Serial.println(F("ACK CMD Over size. END"));
    return 0;
  }
  if (length == 0 ) //0 kb
  {
    Serial.println(F("ACK CMD Size is 0. END"));
    return 0;
  }
  myCAM.CS_LOW();
  myCAM.set_fifo_burst();//Set fifo burst mode
  temp =  SPI.transfer(0x00);
  length --;
  while ( length-- )
  {
```

```
    temp_last = temp;
    temp =  SPI.transfer(0x00);
    if (is_header == true)
    {
      Serial.write(temp);
    }
    else if ((temp == 0xD8) & (temp_last == 0xFF))
    {
      is_header = true;
      Serial.println(F("ACK IMG END"));
      Serial.write(temp_last);
      Serial.write(temp);
    }
    if ( (temp == 0xD9) && (temp_last == 0xFF) ) //If find the end ,break while,
    break;
    delayMicroseconds(15);
  }
  myCAM.CS_HIGH();
  is_header = false;
  return 1;
}
}
```

Train CNN Model Code [30]

```
import tensorflow as tf
import os
import random
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
from keras import preprocessing
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers.core import Activation, Dropout, Flatten, Dense
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.optimizers import Adam

class_names = ['DRONE', 'NO_DRONE']
```

```
width = 150
height = 150

def load_images(base_path):
    images = []
    path = os.path.join(base_path, '*.jpeg')
    for image_path in glob(path):
        image = preprocessing.image.load_img(image_path, target_size=(width, height))
        x = preprocessing.image.img_to_array(image)
        images.append(x)
    return images

training_drone = load_images('./drone')
training_noDrone = load_images('./NoDrone')

plt.figure(figsize=(12,8))

for i in range(5):
    plt.subplot(1, 5, i+1)
    image = preprocessing.image.array_to_img(random.choice(training_drone))
    plt.imshow(image)

    plt.axis('off')
    plt.title('{} image'.format(class_names[0]))

# show the plot
plt.show()

plt.figure(figsize=(12,8))

for i in range(5):
    plt.subplot(1, 5, i+1)
    image = preprocessing.image.array_to_img(random.choice(training_noDrone))
    plt.imshow(image)

    plt.axis('off')
    plt.title('{} image'.format(class_names[1]))

# show the plot
```

```
plt.show()

#prepare images as tensors

X_drone = np.array(training_drone)
X_noDrone = np.array(training_noDrone)

print(X_drone.shape)
print(X_noDrone.shape)

X = np.concatenate((X_drone, X_noDrone), axis=0)
X = X / 255
X.shape

y_drone = [0 for item in enumerate(X_drone)]
y_noDrone = [1 for item in enumerate(X_noDrone)]


y = np.concatenate((y_drone, y_noDrone), axis=0)

y = to_categorical(y, num_classes=len(class_names))

print(y.shape)

#convolutional neural network configuration

# default parameters
conv_1 = 16
conv_1_drop = 0.2
conv_2 = 32
conv_2_drop = 0.2
dense_1_n = 1024
dense_1_drop = 0.2
dense_2_n = 512
dense_2_drop = 0.2
lr = 0.001

epochs = 30
batch_size = 32
color_channels = 3
```

```
def build_model(conv_1_drop=conv_1_drop, conv_2_drop=conv_2_drop,
                dense_1_n=dense_1_n, dense_1_drop=dense_1_drop,
                dense_2_n=dense_2_n, dense_2_drop=dense_2_drop,
                lr=lr):
    model = Sequential()

    model.add(Convolution2D(conv_1, (3, 3), input_shape=(width, height, color_channels),
                            activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(conv_1_drop))

    model.add(Convolution2D(conv_2, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(conv_2_drop))

    model.add(Flatten())

    model.add(Dense(dense_1_n, activation='relu'))
    model.add(Dropout(dense_1_drop))

    model.add(Dense(dense_2_n, activation='relu'))
    model.add(Dropout(dense_2_drop))

    model.add(Dense(len(class_names), activation='softmax'))

    model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=lr),
metrics=['accuracy'])

    return model


np.random.seed(1) # for reproducibility

# model with base parameters
model = build_model()

epochs = 20

# "X" is the array of images, "y" is the type of the images
```

```
model.fit(X,y, epochs=epochs)  #end training model

model.save('drone_detector.h5') #save training model
```

Test Drone Detector Code[31]

```
import os
import numpy as np
import sounddevice as sd
import matplotlib.pyplot as plt
from scipy import signal
from scipy.io import wavfile as wav
from scipy.fftpack import fft, fftfreq
from scipy.misc import electrocardiogram
from scipy.io.wavfile import write
from scipy.signal import find_peaks
from picamera import PiCamera
from time import sleep
from keras import preprocessing
from keras.models import load_model
from glob import glob
import ezgmail
from twilio.rest import Client
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
tf.get_logger().warning('test')
# WARNING:tensorflow:test
tf.get_logger().setLevel('ERROR')
tf.get_logger().warning('test')
# (silence)

#1) Microphone records, if frequency is of drone, it will pass to 2)

fs=44100
duration = 15  # seconds
myrecording = sd.rec(int(duration * fs), samplerate=fs, channels=2)
print("Recording Audio")
sd.wait()
write('record.wav', fs, myrecording)
```

```
print("Audio recording complete , Play Audio")

sd.play(myrecording, fs)
sd.wait()
print("Play Audio Complete")

sample_rate, data = wav.read('sounds/record.wav' ) #sample_rate= samples/sec
print("sample_rate, data = wav.read('sounds/record.wav') #sample_rate= samples/sec")
print("sample rate", sample_rate)
print("data.shape ",data.shape)
data = data.mean(axis=1)
duration= data.shape[0] / sample_rate # number of samples / sampling rate
print("duration", duration)

N = int(sample_rate/2.0) #halt a second
f = fftfreq(N, 1.0/sample_rate)
t = np.linspace(0,0.5,N)
mask = (f > 0) * (f < 1000)
subdata = data[:N]
F = fft(subdata)

freqmax = abs(F[mask])

hz= np.argmax(freqmax)

print("The frequency is {} Hz".format(hz))


if (hz >=1000 and hz <= 5000 )or (hz >=100 and hz <= 500):
    print("Frequency detected is: DRONE")

    #2) Take picture of drone

    camera = PiCamera()
    camera.rotation=180
    camera.start_preview()
    sleep(5)
    camera.capture('/home/pi/Documents/train_test/test_data/images/image.jpeg')
    camera.stop_preview()
```

```
    sleep(10)

    #3) CNN Model will determine if picture is of a drone

    os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

    class_names = ['DRONE', 'NO_DRONE']
    width = 150
    height = 150

    model = load_model('drone_detector.h5')

    base_path_img = './test_data/images'
    images = []
    path = os.path.join(base_path_img, '*.jpeg')
    for image_path in glob(path):
        img = preprocessing.image.load_img(image_path, target_size=(width, height))

    img_X = np.expand_dims(img, axis=0)

    predictions = model.predict(img_X)
    result = class_names[np.argmax(predictions)]

    print('The type predicted is: {}'.format(result))

    if(result == "DRONE"):
        #4.1)Send email to customer when drone detected
        ezgmail.send('customer123@gmail.com', 'Drone Detector Notification', 'Dear User: A
DRONE was detected on your property, please take safe precautions',
["./test_data/images/image.jpeg"])
        #4.2)Send text message when drone detected
        accountSID = 'AC7f6696f5533bef467a95992df2ed3d1e'
        authToken = '595bc8aa845721d3f57da8d12ddd523d'
        twilioCli = Client(accountSID, authToken)
        myTwilioNumber = '+14842224954'
        cellPhone = '+19562219551'
        message = twilioCli.messages.create(body='Dear User - A DRONE was detected on your
property- please take safe precautions.', from_=myTwilioNumber, to=cellPhone)
        media_url=["./test_data/images/image.jpeg"]
    else:
```

```
        print("No Drone on property")
else:
    print("Frequency detected is: NOT DRONE")
```