12/12/2021

Formal Report

Ben Kennedy, Chase Westlake, Ethan Pyle, Kaden Taylor
CAMOSUN COLLEGE

MEMORANDUM

To: Kimberly Lemieux
CC: Mel Dundas, James Van Oort, Justin Curran
From: Surveilia Team
Date: December 12th, 2021
Re: Surveilia Formal Report

Surveilia is pleased to submit this report as a representation of the long hours spent in producing our drone prototype.

Since its inception in the early weeks of September 2021, Surveilia has been busy producing a prototype that will display the requirements we set out to accomplish:

- Successful detections of humans within the camera frame using artificial intelligence.
- Autonomous aviation through designated flight paths.
- An application to display and chart important information pertaining to the flight and security of property.

Our future goals include fine tuning systems of the drone to work out alpha level bugs. We also plan to integrate a compass to fly for the first time autonomously. The Compute Module 4 (CM4) was unachievable over this semester due to supply chain issues regarding critical components. On the next iteration, the CM4 will be implemented to leverage its small size and super processing power.

We are proud of the range of features Surveilia has to offer and we strongly believe we have reached a point where a future team can take it a step further.

EXECUTIVE SUMMARY

Developed by the Surveilia team during the capstone semester, the Surveilia drone is designed to augment contemporary weakness in the security industry, regarding effectiveness of cameras and guards. By leveraging artificial intelligence, it can detect people and roam autonomously.

Surveilia is built from software and hardware including

- ArduPilot
- Linux
- Raspberry Pi
- Kakute F7
- Fusion 360

Over the fall semester, the Surveilia team built a drone prototype capable of

- Flying steadily
- Being quickly manufactured
- Recognizing humans in the camera frame
- Displaying images of people detected on software
- Displaying technical flight data on software
- Communicating over a network using UDP

Surveilia has been a success and every goal outlined for the project have been achieved.

Table of Contents

BACKGROUND

The security industry is undergoing a period of high demand with a workforce that lacks appropriate resources to provide services. Much of the problem is driven by supply and demand. However, intrinsic limitations exist in the available resources. Guards are capable of intellectual tasks but can become fatigued and face reduced awareness. Cameras can provide ongoing and recorded observation, but only in a static area and require some control measure to review observations.

Surveilia is an autonomous security drone that we designed to supplement the needs of the security industry by providing a tool to security companies or residences to augment their protective capabilities. Security guards and cameras are the two primary services that we are seeking to supplement. For concerns about finances, please see *Appendix A* for details.

We built Surveilia to compensate for the weaknesses in both strategies by autonomously patrolling and observing areas specified by the drone operator. Surveilia can

- Autonomously patrol a site
- Report human activity with artificial intelligence

HARDWARE

FRAME

Our design and hardware team has designed the complete CAD drawings for manufacturing of the drone body, using Fusion 360. The prototype was printed using both polylactic acid (PLA) and Polyethylene terephthalate glycol (PETG) with 50% infill. The intention of the prototype was to ensure that all specifications matched our hardware enclosure requirements and that the body had sufficient mechanical integrity for our components.

Figure 1 Surveilia body

The body consists of; four arms, four stand offs, a top plate, and a bottom plate. The arms attach to the bottom plate and provide a mount for the motors. They also contain legs for landing and take off. The legs have been designed to break safely on significant impact. The bottom plate contains mounts for the flight controller and electronic speed controller. They standoffs pair the top plate to the bottom plate.

Flight Controller

Surveilia flight systems are controlled by the Kakute F7 flight controller (FC). The flight controller is responsible for all backend systems of the drone, including the stability, and outputs to the motors for flight control. The operating firmware is ArduPilot.

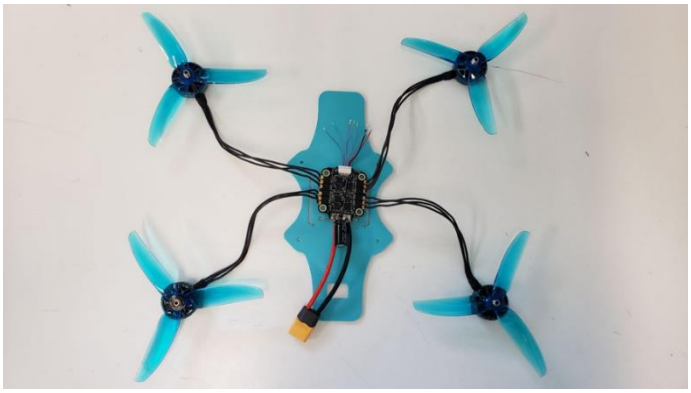

Figure 2 Drive System Assembly

The FC communicates directly to the electronic-speed-controller (ESC). The electronic speed controller processes inputs from the FC to determine how much power is to be delivered to each motor.

Motors

Surveilia uses four T-Motor Velox Veloce V2 motors. These are 1750KV, or RPM per volt. These are lightweight, high-power motors that can be used for a variety of purposes. The amazingly fast speeds of these motors were ideal for use to make a smooth flying drone with plenty of power to lift all our electronics.

GPS

The BN-220 is a small compact GPS module that connects to the flight controller. It provides location feedback to the flight controller relating to the drone's latitude and longitude. The GPS does not work well indoors, this is because the walls are too thick to ping the satellites. Once the drone is outdoors the GPS connects to three to seven satellites.

The GPS is a critical component of fail-safe software and Mission Planner. If the drone were to be under manual control, the GPS can re-home the drone if it were to lose connection. Also, Mission Planner requires location data so it can successfully achieve autonomous patrols.

Raspberry Pi Zero & Daughter Board

The Raspberry Pi Zero (RPi) operates as the Surveilia flight computer. It was selected to replace the Compute Module 4 (CM4), which could not be manufactured due to supply issues.

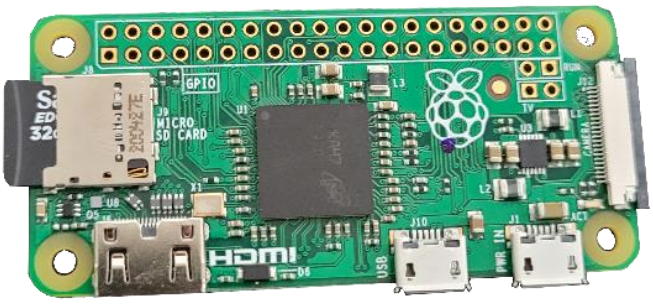

Figure 3 Raspberry Pi Zero Wi-Fi

The RPi was selected due to its small size and low weight. It only holds the required peripherals needed for the drone. It can connect to wireless networks and can take advantage of our human recognition software.

The daughter board was designed such that the RPi could be dropped onto it with soldered pin headers and begin collecting data to feed to the Surveilia ground station. It uses sensors to determine the motion of the drone, and the weather.

SENSORS

The MPU-6050 is a two in one accelerometer and gyroscope on a compact breakout board. It is attached to the flight computer using two I2C lines. The point of the MPU-6050 is to record the movement of the drone as it flies. The data from the sensor is sent to the ground station to be displayed in the home and stats section. The flight controller has its own gyroscope and accelerometer and it's used for the Proportional Integral Derivative (PID) controller to maintain flight stability.

The DHT11 is used to collect environmental data during flight. This two in one module is built onto the flight computer daughter board. It is responsible for acquiring temperature and humidity data from the drone back to the ground station.

FIRMWARE

ARDUPILOT & MISSION PLANNER

ArduPilot is built on to the Kakute F7 flight controller and interfaces with the Mission Planner software. Surveilia was able to accomplish a baseline for automation and is ready for next steps in autonomous flight. For automation to be successful, a compass is required

Figure 4 Mission Planner [1]

Mission Planner is the open-source software we use to automate our drone to fly autonomously through written paths. Follow the following procedures to ensure that you have a successful initial flight.

SOFTWARE

AI

Surveilia collects video using an 8-megapixel camera that is connected to our flight computer. Our program recognizes potential intruders by collecting all the frames from the live video feed and checking each frame for humans. It does this by sending frames into a pre-built algorithm from OpenCV which is a library of recognition functions mainly aimed at real-time computer vision [2]. This algorithm processes each frame individually and uses pre-trained AI (agent) to check each processed frame for a person. If the agent finds a person in the frame, it returns an image of the frame with a green rectangle encompassing them, the total person count, and the agent's confidence of its accuracy.

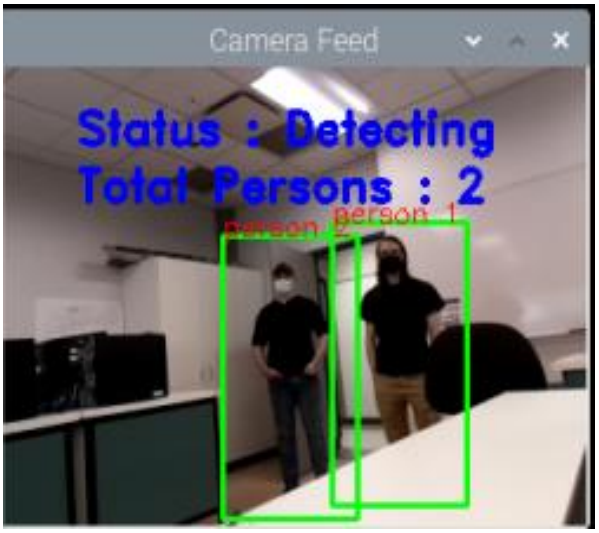

Figure 5 Kaden & Chase being detected by AI

FILE STORAGE & UDP

The file storage algorithm (*See Appendix F for flow chart)* is a subset of the Ground Station software and is run using the UDP server. We needed an algorithm to manage our flight data to be displayed in real time to the ground station. This algorithm ensures that data is not lost during restart of the application. The algorithm stores data returned from the drone for documentation and completes parsing to the Ground Station upon start-up. The algorithm works by retrieving data points every ten seconds and then storing it to a file line-by-line. Each line is counted and stored to the start of same file, so the software knows how many data points to cycle. The storage system is an abstraction of a packet.

GROUND STATION

The Surveilia Ground Station is a graphical user interface (GUI) that displays relevant flight data. Using the UDP connection from the Raspberry Pi Zero to the home computer over a local network, we have been able to successfully display received filtered images and relevant data on our application. The Ground Station is an application for the operator built in Visual Studio 2019, with a MVVM framework structure. The Ground Station offers many features for the users through the four windows.

Each of the four windows has its own task (*See Appendix H for view descriptions*). However, the primary tasks of the Ground Station are contained in the home and stats windows. Home is responsible for rapidly displaying immediate flight data and displaying any recent images that have detected people within the frame.

Stats is responsible for the graphical representation of flight data. It uses a live chart to constantly update an interface to provide a statistical overview of flight motion.

PROBLEMS

COMPUTE MODULE 4

The Compute Module 4 (CM4) was originally our flight computer, responsible for communicating with the ground station and operating inflight software. The CM4 provides the ability to take advantage of the power of the Raspberry Pi 4, while reducing weight and meeting our PCB design requirements.

Figure 6 Compute Module 4 [3].

The original source files contain many components that are unnecessary for our project. The size of the board also extends beyond the capacity of our drone casing. We removed all unused connectors, reducing board size, maintaining power and camera circuits, and adding a Wi-Fi antenna to maximize range of the CM4.

Unfortunately, we determined that parts for the CM4 board would not be available for manufacturing within the timeline of our project. Instead, we designed a daughter board for the Raspberry Pi Zero with the same intention as the CM4.

CONCLUSION

The Surveilia team is proud of our accomplishments. Having completed our scope and deliverables, we have officially achieved what we aimed to do at the beginning of the semester. We have achieved stabilized flight, communication between the onboard computer and our local desktop, developed a starting point for expanded automation of the drone, and have constructed an application for the user to monitor and log the incoming flight data. This all seeks to enhance and augment current shortcomings in the security industry and to provide reliable security for homes and possessions.

We have been given an incredible opportunity to develop our skill sets by working through various challenges and stages of an engineering project. There were many lessons learned and changes we would make in the future. We are thankful for the opportunity to have learned all that we did over the course of the project semester.

We are excited and proud to demonstrate our drone at the upcoming symposium, December 15th.

APPENDICES

APPENDIX A: COST-BENEFIT ANALYSIS

Guards are quite expensive, as most physical security is contractual and fulfilled by third parties. 24-hour security can generate monthly wage costs (before third party fees), approximately between $10,342.08 and $18,144 [4]. Static security cameras, although cost effective relative to guards, cannot provide wide range of observation. Cameras can still be quite expensive and vary in price based on the type of camera installed [4].

Table 1. Table outlining costs of Surveilia relative to other services available.

Camera	**Guard**	**Surveilia**
$125 - $450 per camera $500 - $1600 for 4 CCTV cameras	$10,342 - $18,144 per month	$650 onetime fee

APPENDIX B: AI TROUBLE

The AI is very reliable; however, it sometimes recognizes inappropriate objects as people. The recognition issue is likely caused by the histogram of oriented gradients (HOG) descriptor algorithm miscalculating the x and y gradients direction or the magnitude and orientation for some of the pixel values [5].

The AI has substantially more false positives in dimensionally cluttered and/or dark spaces which would imply that our classification algorithm (a support vector machine or SVM) has a limited model performance [5]. This would make sense as the hardware that our software is running on is a Raspberry Pi CM4 which has very low processing power when compared to something like a typical desktop computer.

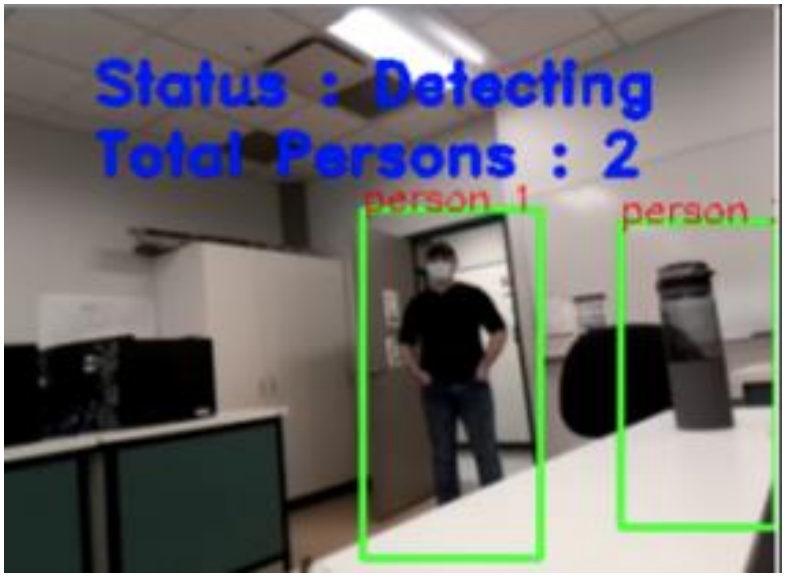

Figure 7 AI Mistaking Water Bottle for a Person.

APPENDIX C: AI CODE

```python
#------------------------------
#Program: human_detection1.0.py
#Author:  Ben Kennedy
#Date:    2021-10-08
#------------------------------

#imports required
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2
import imutils
import numpy as np
import argparse

#This function receives a frame from the camera and then checks the frame for people.
#It does this using a histrogram of oriented gradient (HOG) descriptor algorithm to process
#the image and then uses a pre-trained model (support vector machine or SVM) to check for people.
def detect(frame):
    bounding_box_cordinates, weights = HOGCV.detectMultiScale(frame, winStride = (4,4), padding = (8,8), scale = 1.03)
    person = 1
    for x,y,w,h in bounding_box_cordinates:
        cv2.rectangle(frame, (x,y), (x+w,y+h), (0,255,0), 2)
        cv2.putText(frame, f'person {person}', (x,y), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,255), 1)
        person += 1

    cv2.putText(frame, 'Status : Detecting ', (40,40), cv2.FONT_HERSHEY_DUPLEX, 0.8, (255,0,0), 2)
    cv2.putText(frame, f'Total Persons : {person-1}', (40,70), cv2.FONT_HERSHEY_DUPLEX, 0.8, (255,0,0), 2)

    # Returns the frame with any detected persons bound by a rectangle
    # The frame is also returned including the total person count and SVM's confidence value
    return frame


# This function is required for capturing frames from the PiCamera
def detectByCamera():

    # Initializes the camera
    camera = PiCamera()
    camera.resolution = (320, 240)
    rawCapture = PiRGBArray(camera, size=(320, 240))

    # Tells the camera to capture video continously while storing the current frame
    # From the array into the image variable. It does this by grabbing the raw NumPy array
    # Representing the image, and then initalizing the timestamp
    for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
        image = frame.array

        # Check the current frame for persons
        detect(image)

        # Stream live video feed on a seperate window
        cv2.imshow("Camera Feed", image)
        key = cv2.waitKey(1) & 0xFF

        # Clear the stream in preparation for the next frame
```

Figure 8 Code Written by Surveilia Team for AI.

APPENDIX D: HARDWARE FLOW CHART

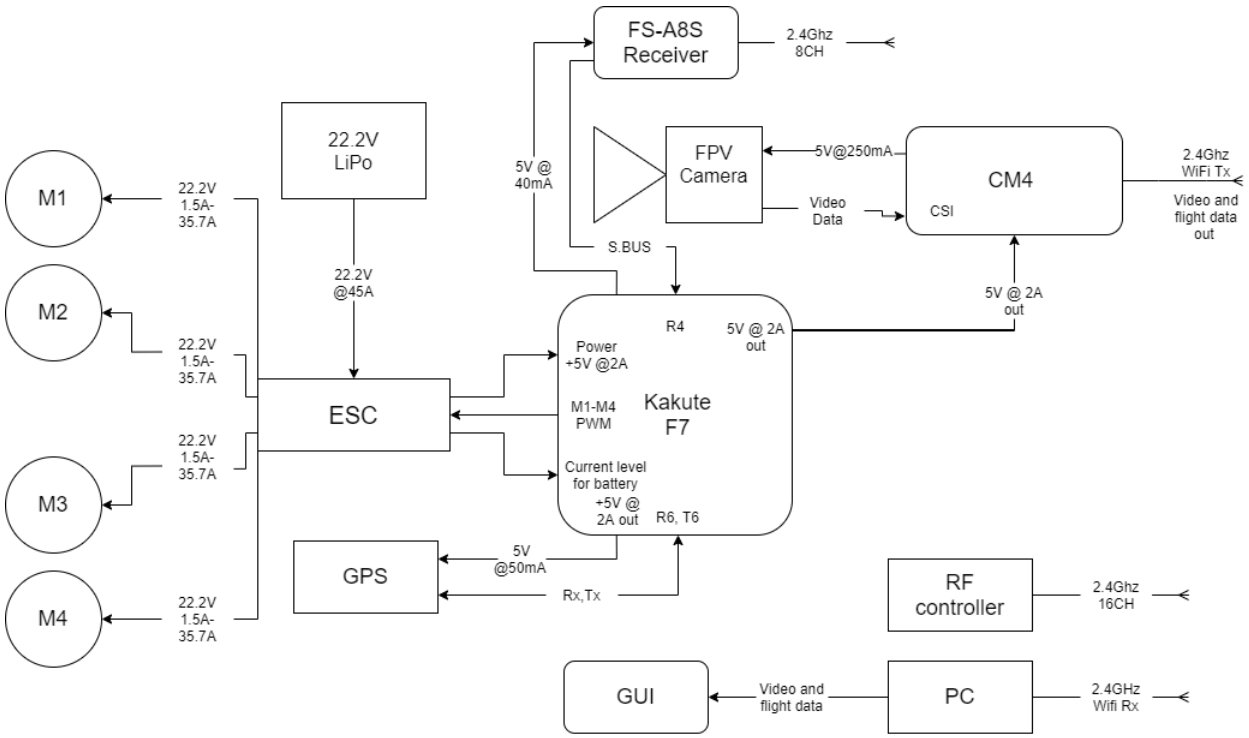

Figure 5 Hardware Flow Chart of Drive and Control Systems.

APPENDIX E: HOG DESCRIPTOR ALGORITHM

We are using a Histogram of Oriented Gradient (HOG) Descriptor algorithm to process each frame and a Support Vector Machine (SVM) to analyse the data for classification.

To understand the algorithm, it is important to understand video file structure. A video file combines a sequence of images to form a moving picture, typically referred to as frames. Once these frames are stored into an array, they can be easily accessed and manipulated [5]. OpenCV [2] has already been implemented in an efficient way to combine the HOG Descriptor algorithm with an SVM, so we are only required to send our array of frames through OpenCV's algorithm.

The first thing our software does is call the pre-model for human detection of OpenCV and then feeds our SVM with it. Whenever a frame is received from the camera, it is passed to OpenCV's algorithm. It then takes the frame and looks for people within the frame. If there are people detected OpenCV returns the frame with the person(s) bound by a green box and provides the person count, and the confidence value of its reliability.

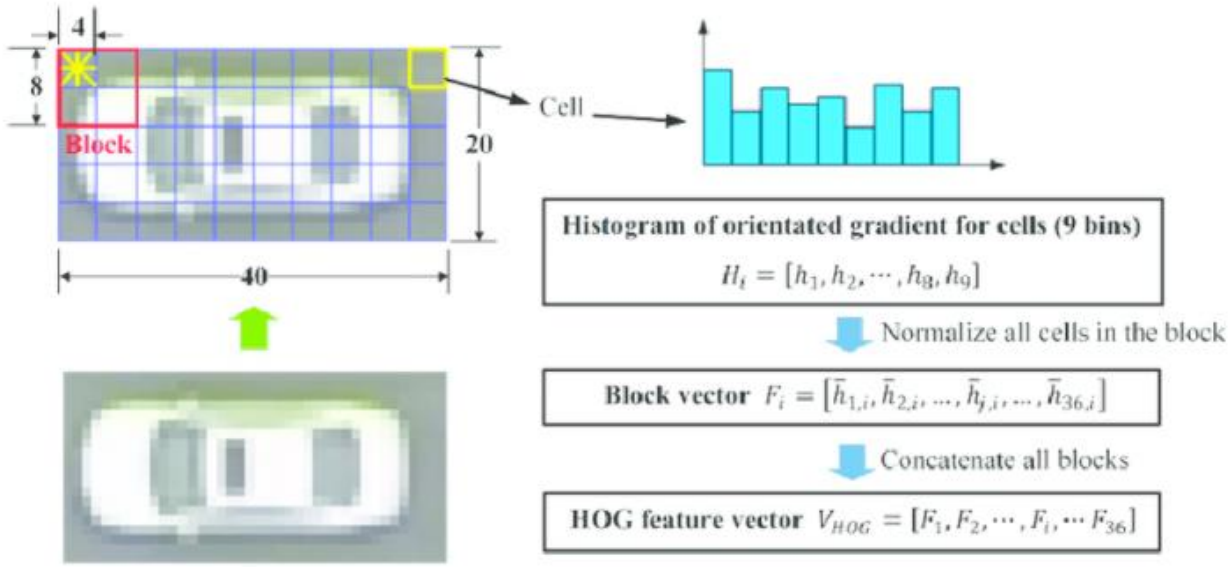

Figure 6 HOG Descriptor Algorithm [5].

APPENDIX F: DATA STORAGE FLOW CHART

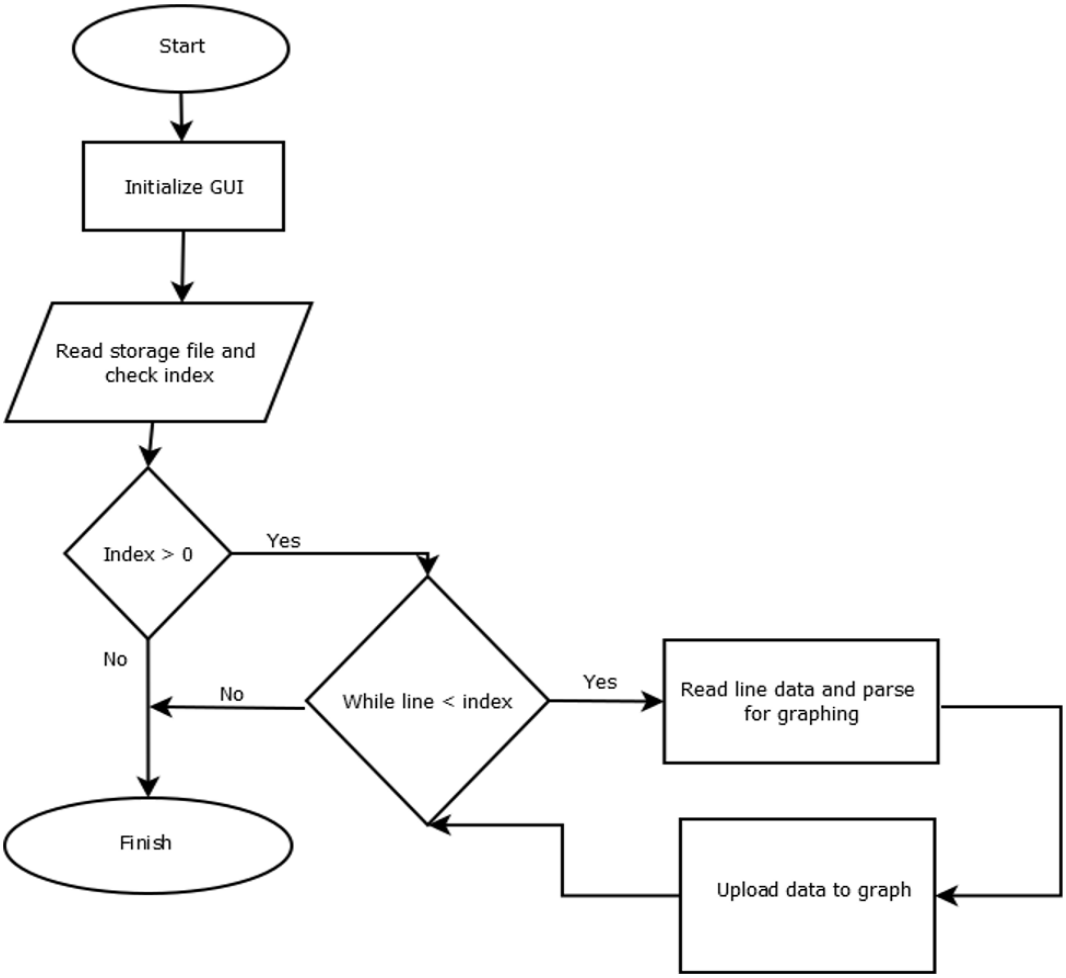

Figure 11 Flow Chart Defines Basic Strategy for Parsing Storage to GUI

APPENDIX G: PI ZERO DAUGHTER BOARD SCHEMATIC

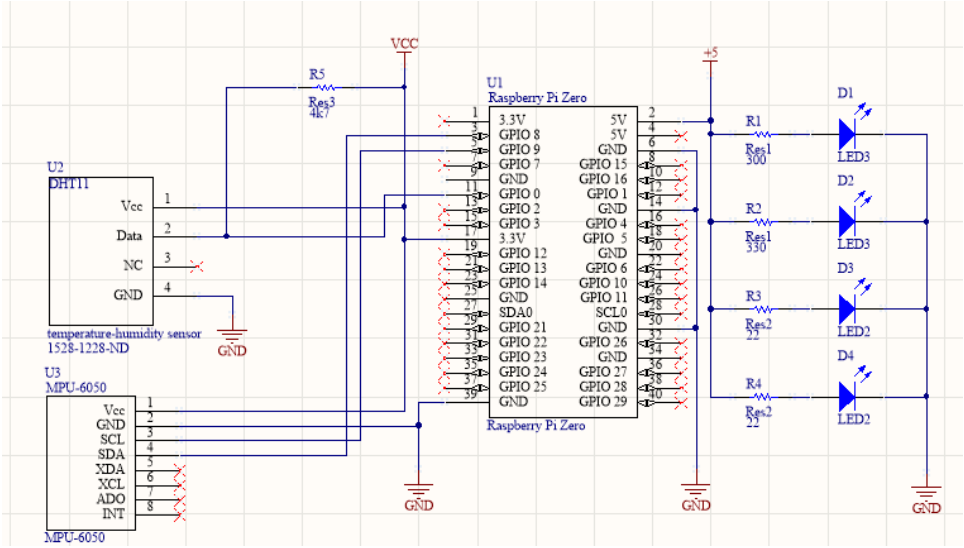

Figure 12 PI Zero Daughter Board Schematic

APPENDIX H: GROUND STATION VIEW FUNCTIONALITY

HOME: On the home screen, you will find an image with people squared out if humans are detected midflight within the camera frame, the Inertial Measurement Units (IMU) from onboard the Raspberry Pi, along with the number of people spotted in the frame, the temperature, and humidity.

USER MENU: The User Menu allows the operator to set their name for logging purposes. Therefore, when the operator chooses to capture the graphing of the data for future demonstration purposes, it has both the time stamp and username for security and insurance purposes.

STATS: Using LiveCharts, an open-source charting project, we have been able to display our transmitted information through text files on the application. LiveCharts is an aesthetically pleasing extension that functions just as well as it looks.

HELP: The Help menu offers simple troubleshooting advice and tips before going out to fly, and includes the link to our website, which contains our contact info and further information including the user manual, which they may find resourceful.

REFERENCES

[1] Mission Planner. (Aug 1, 2021). ArduPilot. Accessed September 20th, 2021. [Software application]. Available: https://firmware.ardupilot.org/tools/missionplanner/

[2] A. Rosebrock, “Install OpenCV 4 on your Raspberry Pi,” 26 Sept, 2018. [Online]. Available https://www.pyimagesearch.com/2018/09/26/install-opencv-4-on-your-raspberry-pi/. [Accessed: Sept. 23, 2021].

[3] Iottrends, “What is Raspberry Pi Compute Module 4 – A Complete Guide,” 16 Jan, 2021 [Online]. Available:https://www.iottrends.tech/blog/raspberry-pi-compute-module-4-everything-you-need-to-know/ [Accessed: Oct 12, 2021].

[4] Paladin Security, “Security Guard Salary in Canada by Province in 2021,” Nov 14,2021. [Online] Available: https://paladinsecurity.com/security-careers/security-guard-salary/. [Accessed: Sept 16, 2021].

[5] Wikipedia, “Histogram of Oriented Gradients,” 01 Feb, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients. [Accessed: September 24, 2021].

[6] Y. Xu, G. Yu, Y, Wang and X. Wu, “A Hybrid Vehicle Detection Method Based on Viola-Jones and HOG + SVM from UAV Images”, ResearchGate, Sensors. 16. 1325., p.6, Aug, 2016. [Online]. Available: https://www.researchgate.net/publication/306333664_A_hybrid_vehicle_detection_method_based_on_viola-jones_and_HOG_SVM_from_UAV_images. [Accessed: Oct 12, 2021].