University of Malaya
Faculty of Computer Science and Information Technology
Department of Software Engineering

WIF3008
Real Time Systems
Lecturer: Dr. Asmiza Binti Abdul Sani

Tello Drone Project Report

Group Members (Tutorial 2 - Group 1)

Name	Matric number
Maher Al Najjar	17098952/1
Shahan Ali Pranto	17078402/1
Zhen Jiaqi	17162497/1
Omar Essam Manaa	17128319/1
Faiyaz Khan (Leader)	17128397/2

Table of Contents

Introduction

In this project, we have been assigned to create a prototype control panel for the Tello EDU drone using Python programming language. Tello EDU is an impressive and programmable drone perfect for education. Tello EDU comes with advanced commands and increased data interfaces.

Figure 1: Tello EDU Drone

We were provided with the pre-planned route of the drone (sweep_route.py), the drone SDK and the standard drone connection (tello.py). By using all these resources, we have created a control panel by which a user can:

- Manually Control the drone.
- Initiate normal perimeter sweep for the drone following a pre-plan route.
- Override the pre-planned route whenever a suspicious situation is detected.
- Display video feed from the drone’s camera to the control panel.

The drone is also able to go back to its original planned route and continue its perimeter sweep if the override functionality is stopped.

Description

Requirements

ID	Description	Priority
FR-01	The system shall allow the user to take off the drone within 7 seconds.	High
FR-02	The system shall allow the user to pause the drone in the sky within 3 seconds.	High
FR-03	The system shall allow the user to land the drone within 3 seconds.	High
FR-04	The system shall allow the user to control the drone to turn left after take off.	High
FR-05	The system shall allow the user to control the drone to turn right after take off.	High
FR-06	The system shall allow the user to control the drone to fly forward after take off.	High
FR-07	The system shall allow the user to control the drone to fly backward after take off.	High
FR-08	The system shall allow the user to control the drone to flip left after take off.	High
FR-09	The system shall allow the user to control the drone to flip right after take off.	High
FR-10	The system shall allow the user to control the drone to flip forward after take off.	High
FR-11	The system shall allow the user to control the drone to flip backward after take off.	High
FR-12	The system shall allow the user to fly the drone at high speed after take off.	High
FR-13	The system shall allow the user to fly the drone at low speed after take off.	High

FR-14	The system shall allow the user to stop the drone from flying using emergency stop.	High
FR-15	The system shall allow the user to control the drone to climb up after take off.	High
FR-16	The system shall allow the user to control the drone to descend after take off.	High
FR-17	The system shall allow the user to turn the drone 30 degree clockwise after take off.	Medium
FR-18	The system shall allow the user to turn the drone 30 degree counterclockwise after take off.	Medium
FR-19	The system shall allow the user to let the drone to perimeter sweep after take off.	High
FR-20	The system shall allow the user to stop perimeter sweep by pressing override route button	High
FR-21	The system shall allow the user to enable video stream	High
FR-22	The system shall allow the user to disable video stream	High

Non Functional Requirement

ID	Description	Priority
NFR-01	The control panel should be easy to use	High
NFR-02	The control panel should have elegant user interface	Medium
NFR-03	The control panel should be able to send commands to the control panel without any issue.	High

Design

Use Case Diagram

i. Diagram:

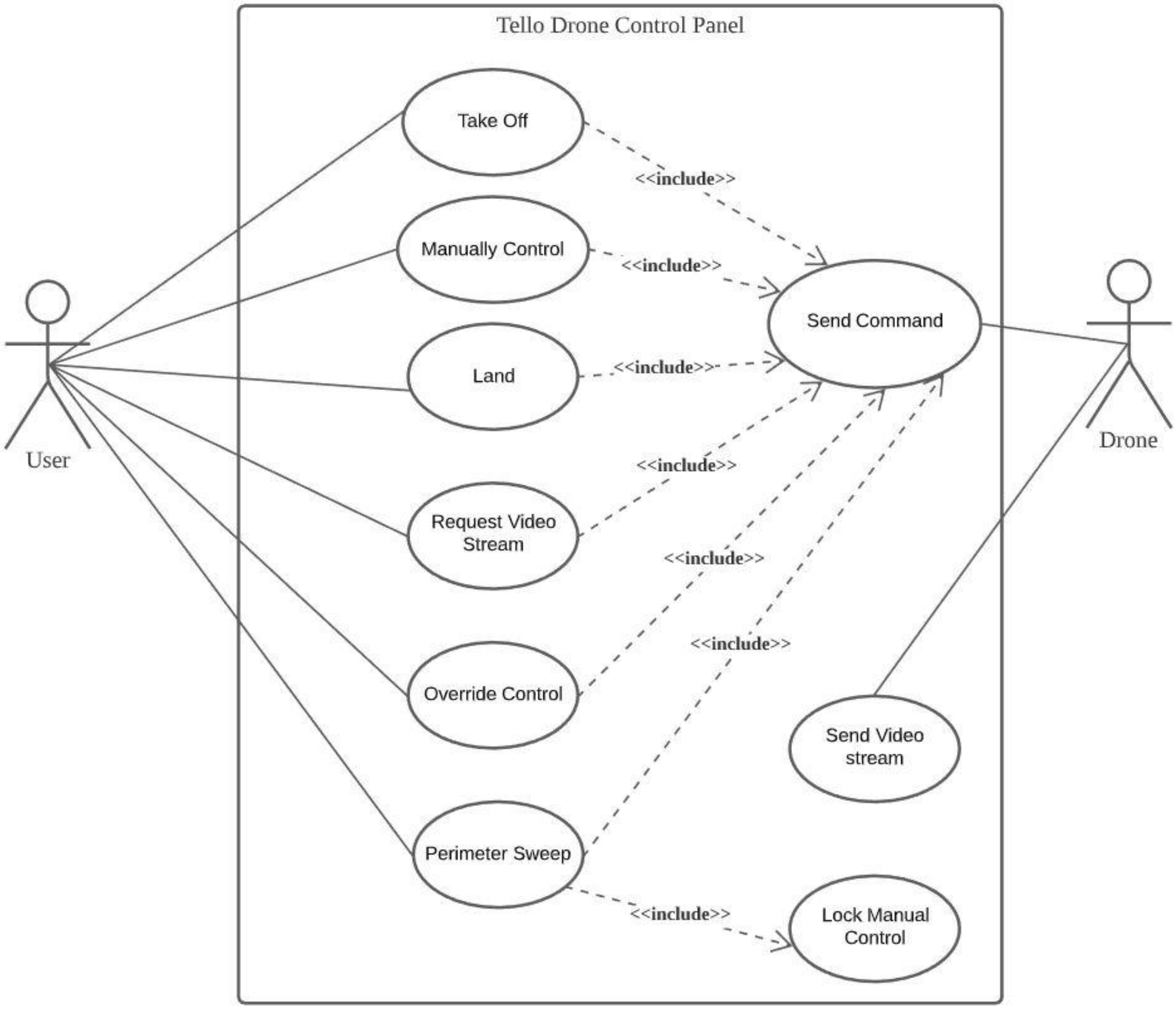


Figure 2: Use Case Diagram

ii. Use Case Descriptions:

1. Take off

Use case name	Take off
Use case ID	01
Actor(s)	User
Brief description	System will allow the user to take-off the drone i.e. from the stationary ground position, to levitated air position. The process takes 7 seconds.
Preconditions	The system is up and running.
Post-conditions	After taking off, the drone can fly in manual or autonomous mode.
Flow of events	1. User clicks the “Take off” button. 2. The drone takes 7 seconds to start the rotors and go to the levitated air position.
Alternate flows and exception	If the drone is in autonomous mode i.e. doing perimeter sweep, pressing this button will not work.
Issues	Drone battery level is not sufficient.

2. Manually Control

Use case name	Manually Control
Use case ID	02
Actor(s)	User
Brief description	The system allows the user to manually control the drone to fly in any directions. The buttons to control the drone manually are : forward, backward, left, right, up, down and the other altitude and speed controls.
Preconditions	The drone is not in autonomous mode.
Post-conditions	Using the manual controls, the user can fly the drone in any direction.
Flow of events	1. User clicks the manual controls individually or in combination. 2. Drone flies according to the input from the user.
Alternate flows and exception	If the drone is in autonomous mode i.e. doing perimeter sweep, pressing these buttons will not work.
Issues	Manually controlling the drone without expertise might result in unexpected collisions.

3. Land

Use case name	Land
Use case ID	03
Actor(s)	User
Brief description	The system shall allow the user to safely land their drone. This landing process takes 3 seconds.
Preconditions	The drone must be already in the air.
Post-conditions	After successful landing, the user can safely shut down the system.
Flow of events	1. User clicks the “Land” button. 2. The system takes 3 second to safely land the drone from the air.
Alternate flows and exception	If the drone is in autonomous mode i.e. doing perimeter sweep, pressing this button will not work.
Issues	There are obstacles for clear landing.

4. Request Video Stream

Use case name	Request Video Stream
Use case ID	04
Actor(s)	User
Brief description	The system allows the user to enable or disable video stream from the drone using the controls “Enable video stream” and “Disable video stream”. The enabling or disabling the stream process takes 3 seconds.
Preconditions	The system is powered on.
Post-conditions	After enabling video stream, users will be able to see from the drone’s perspective.
Flow of events	1. User clicks the “Enable video stream” button. 2. Drone starts transmitting video stream after 3 seconds. 1. User clicks the “Disable video stream” button. 2. Drone stops transmitting the video stream after 3 seconds.
Alternate flows and exception	n/a
Issues	Drone battery level is not sufficient.

5. Override Control

Use case name	Override Control
Use case ID	05
Actor(s)	User
Brief description	While the drone is in autonomous mode, this feature allows the user to take manual control of the drone even while doing perimeter sweep or vice versa.
Preconditions	The drone is up and running.
Post-conditions	From being in the previous autonomous mode, users can now manually control the drone or vice versa.
Flow of events	1. The drone is in autonomous mode. 2. User clicks the “Override route” button. 3. The drone now is in manual control mode.
Alternate flows and exception	If the drone is already doing perimeter sweep and manual control is requested and then, override control is selected, the drone goes to the last position of doing the perimeter sweep and continues from there.
Issues	n/a

6. Perimeter Sweep

Use case name	Perimeter Sweep
Use case ID	06
Actor(s)	User
Brief description	The system allows the user to do a perimeter sweep along the 6 checkpoints including the initial checkpoint.
Preconditions	The system is up and running.
Post-conditions	The drone does a perimeter sweep along the determined checkpoints.
Flow of events	1. User clicks the “Perimeter sweep” button. 2. Drone starts moving along the perimeter sweep route passing one checkpoint after another to complete the sweep.
Alternate flows and exception	If the drone is already in the middle of perimeter sweep, pressing this button will not trigger anything.
Issues	The route for perimeter sweep is not properly defined.

7. Send Command

Use case name	Send Command
Use case ID	07
Actor(s)	Drone
Brief description	The system receives any command from the user control panel and acts on the drone to carry out those instructions.
Preconditions	There is an active connection established in between the control panel and the drone.
Post-conditions	After successful command transfer, the drone carries out the command.
Flow of events	1. The control panel sends a command. 2. Drone receives the command. 3. Drone carries out the command.
Alternate flows and exception	n/a
Issues	The command is not successfully sent to the drone and therefore, malfunctioning.

8. Send Video Stream

Use case name	Send Video Stream
Use case ID	08
Actor(s)	Drone
Brief description	When the user requests a video stream, the drone transmits the video stream into the control panel.
Preconditions	The system is up and running.
Post-conditions	After sending the video transmission, the user can see the video transmission from the drone's perspective.
Flow of events	1. User requests for a video stream from the control panel. 2. Drone transmits video streams.
Alternate flows and exception	n/a
Issues	Connectivity issues leading to unclear or no video stream.

9. Lock Manual Control

Use case name	Lock Manual Control
Use case ID	09
Actor(s)	n/a
Brief description	When the drone has entered autonomous mode to do the perimeter sweep, it locks the manual controls until override control is requested.
Preconditions	The drone enters autonomous mode.
Post-conditions	The manual controls are locked so they can not be used.
Flow of events	1. Drone enters autonomous mode. 2. The manual controls are locked.
Alternate flows and exception	n/a
Issues	System is not acknowledging that it’s in autonomous mode therefore not locking manual controls.

Class Diagram

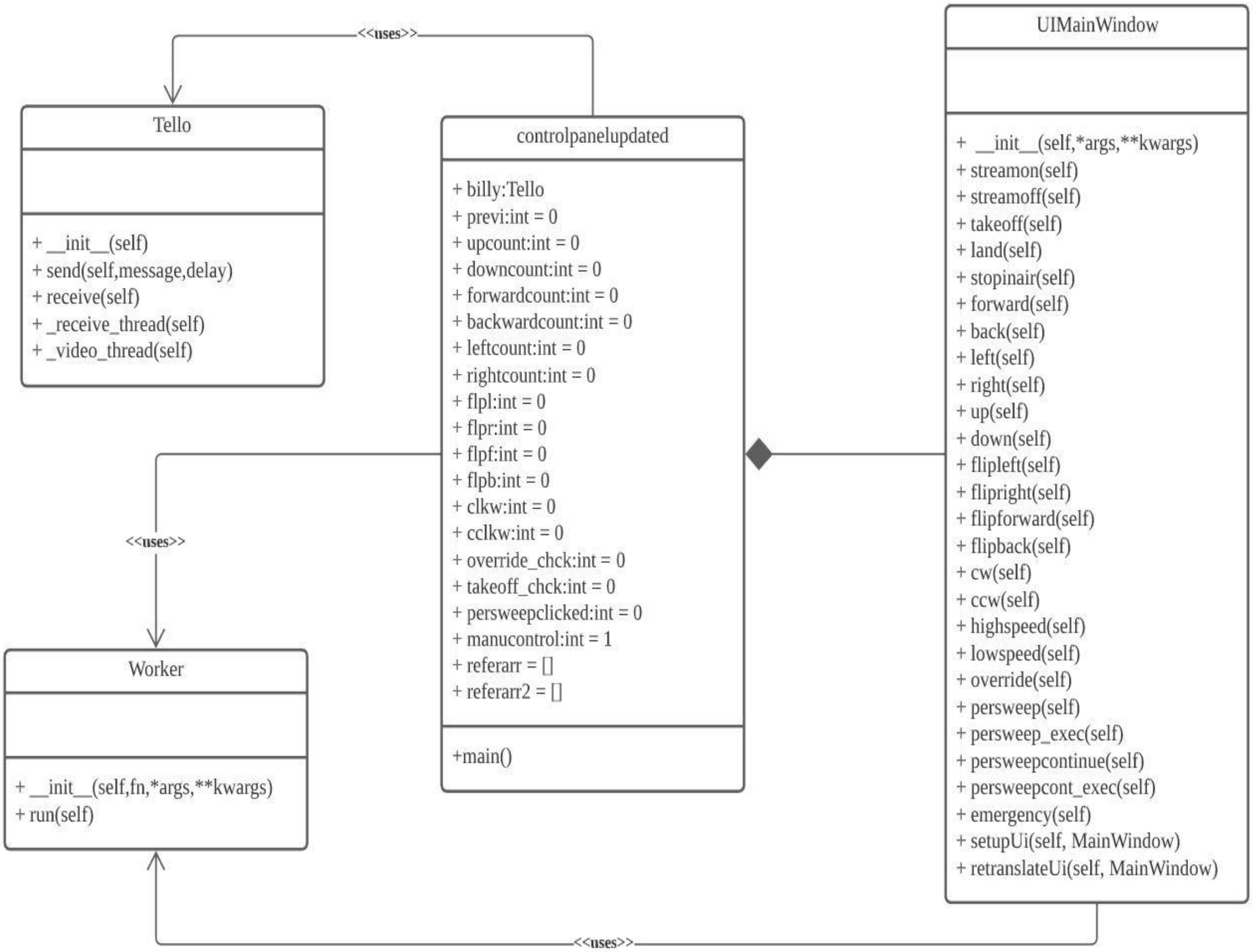


Figure 3: Class Diagram

Activity Diagram

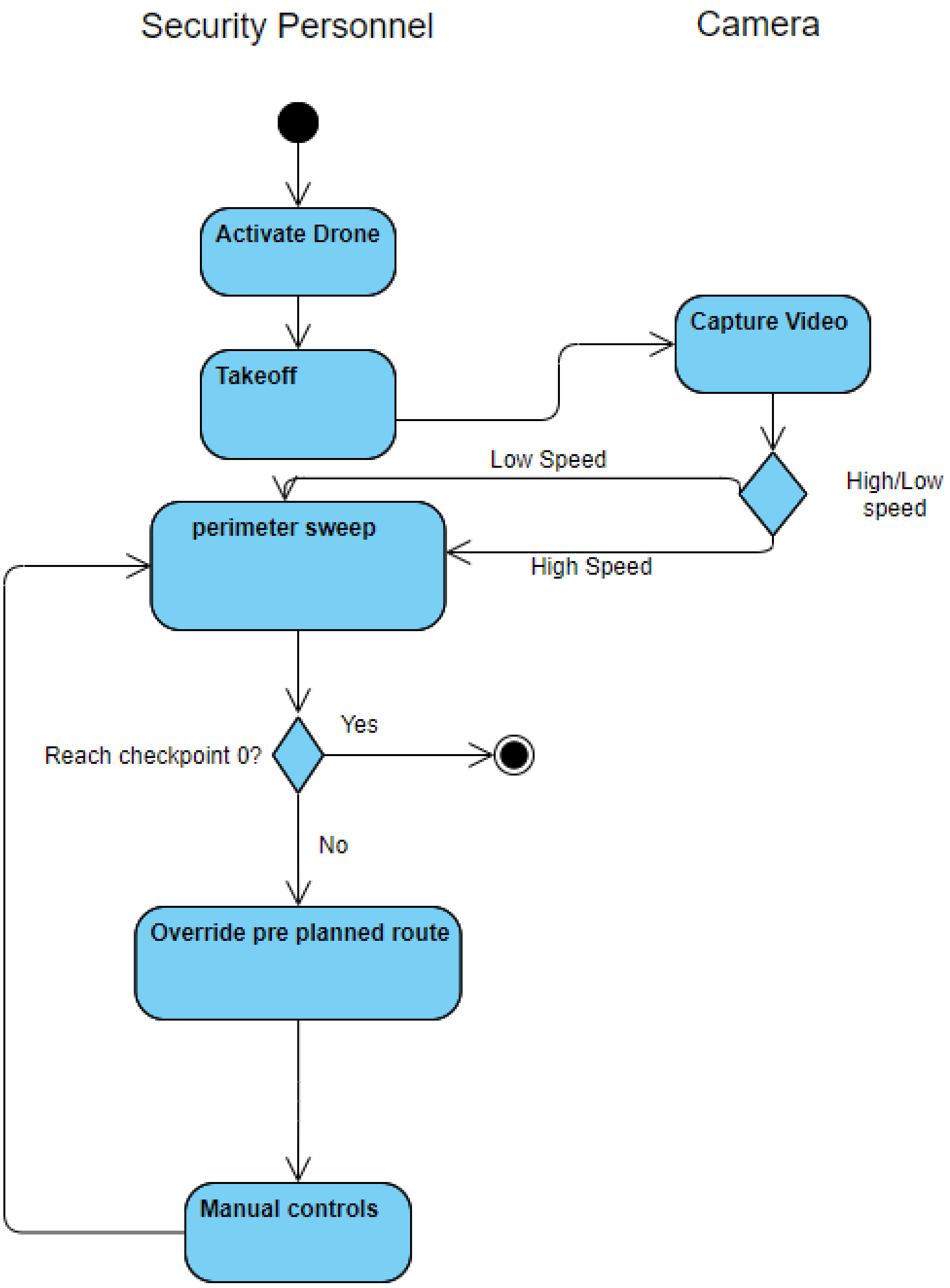


Figure 4: Activity Diagram

Implementation

As we were instructed, we have implemented the control panel using python programming language. We have used some packages of python like PyQT5 and OpenCV.

PyQt5 is a comprehensive set of Python bindings for Qt v5. It is implemented as more than 35 extension modules and enables Python to be used as an alternative application development language to C++ on all supported platforms including iOS and Android. This package was mainly used to make the graphical user interface.

Figure 5: PyQT5

OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection. We have used openCV to add the video streaming feature by which the user can stream video feed from the drone’s camera.

Figure 6: OpenCV

At first we used the standard drone connection (tello.py) to create connection with the drone. Then with the help of the provided SDK guide, we used the commands to create several functions that are assigned with the buttons of the control panel to control the drone.

We used the provided pre-planned route of the drone (sweep_route.py) to create the perimeter sweep function. This function is also assigned to a button by which the user can start the perimeter sweep. We also created the override functionality by which the perimeter sweep can be overridden.

Furthermore, we also created a functionality by which the drone can go back to the point where the override functionality was triggered. By this way, the drone can finish it’s sweep without starting all over again.

Lastly, we have implemented two functions inside the tello class to receive and create the video stream object which will enable the user to see a video stream from the drone’s camera by clicking the video stream button.

Snapshots of Input/Output

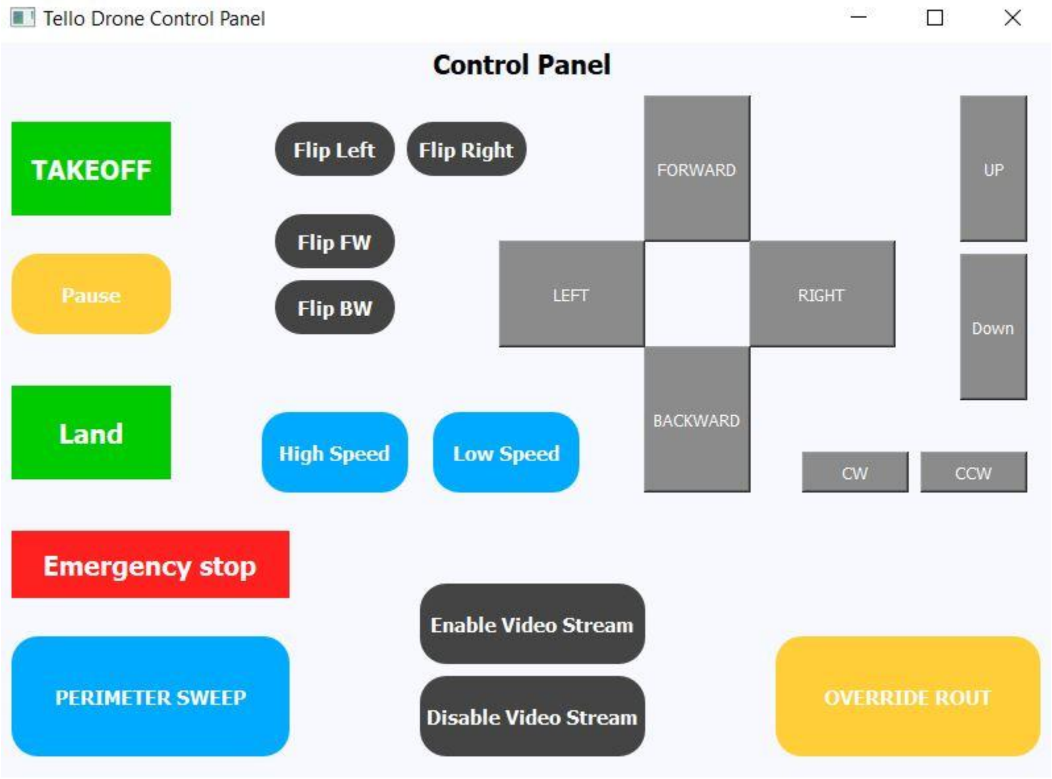
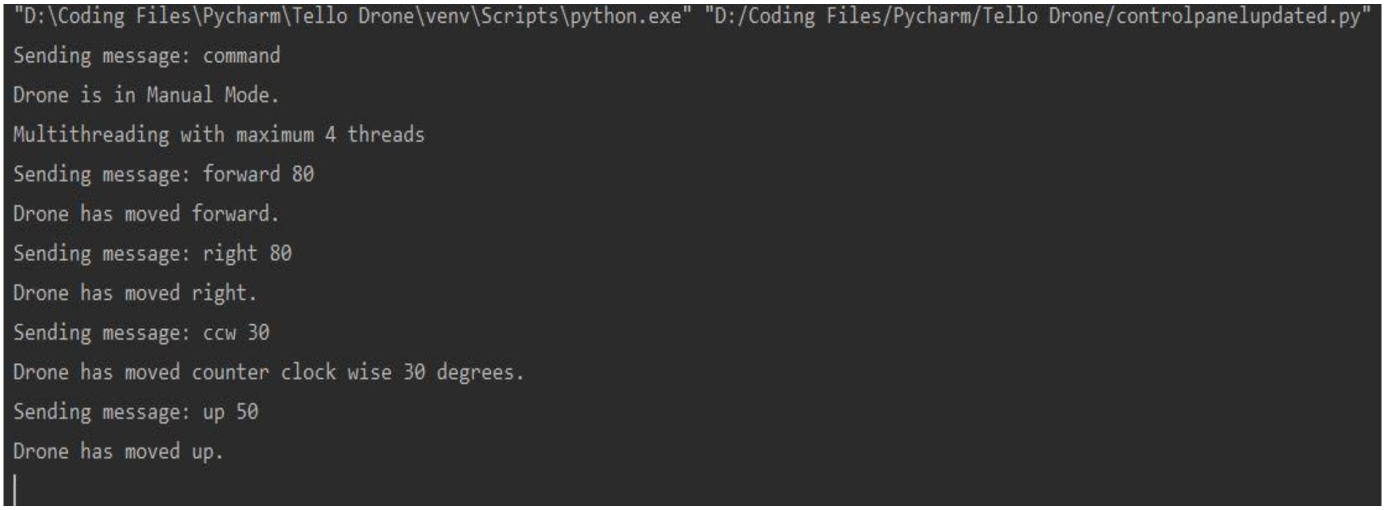
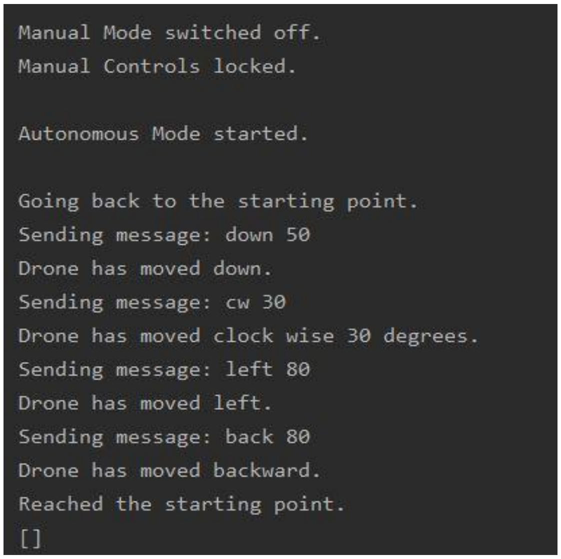


Figure 7: GUI



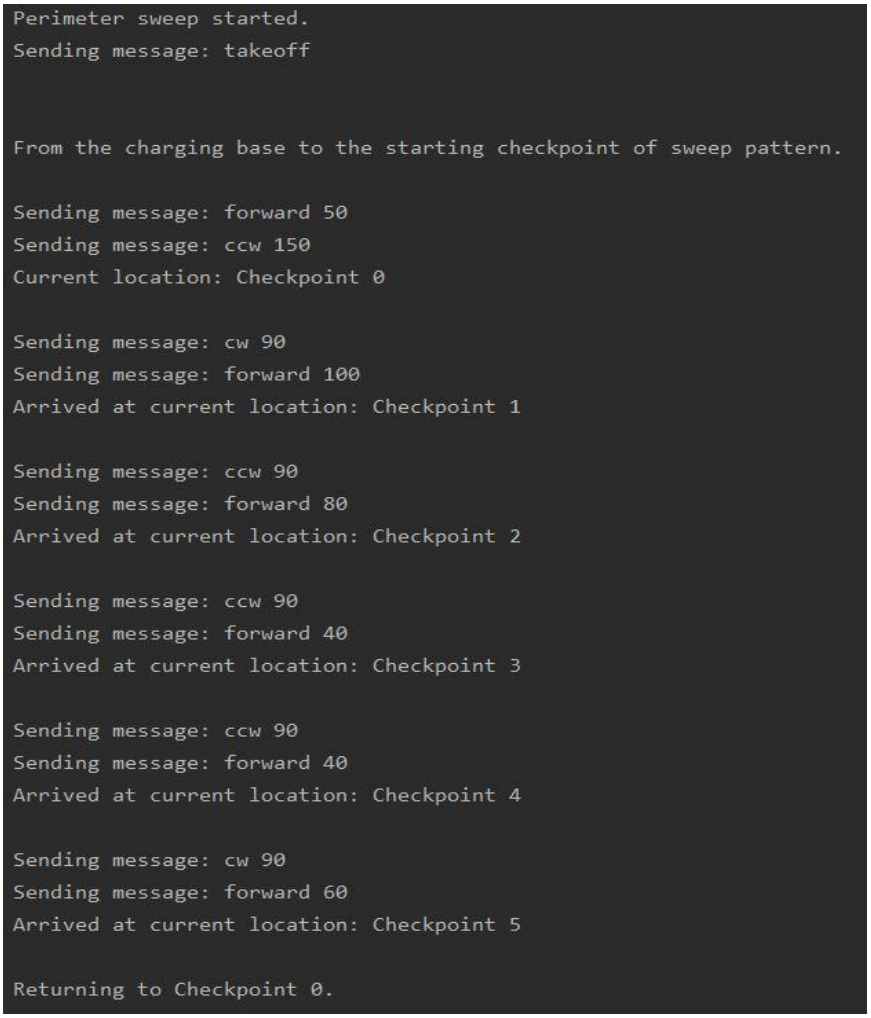
```
"D:\Coding Files\Pycharm\Tello Drone\venv\Scripts\python.exe" "D:/Coding Files/Pycharm/Tello Drone/controlpanelupdated.py"
Sending message: command
Drone is in Manual Mode.
Multithreading with maximum 4 threads
Sending message: forward 80
Drone has moved forward.
Sending message: right 80
Drone has moved right.
Sending message: ccw 30
Drone has moved counter clock wise 30 degrees.
Sending message: up 50
Drone has moved up.
```

Figure 8: Output during manual Controls



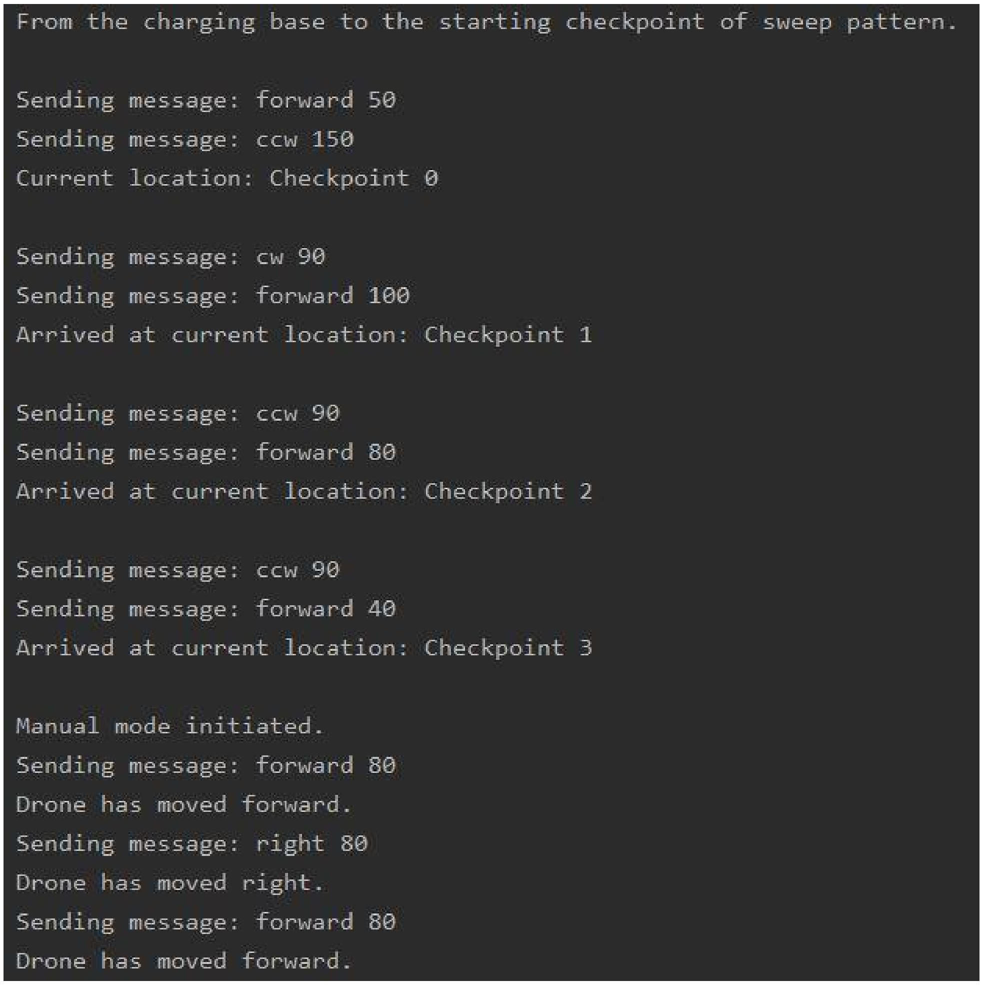
```
Manual Mode switched off.
Manual Controls locked.

Autonomous Mode started.

Going back to the starting point.
Sending message: down 50
Drone has moved down.
Sending message: cw 30
Drone has moved clock wise 30 degrees.
Sending message: left 80
Drone has moved left.
Sending message: back 80
Drone has moved backward.
Reached the starting point.
[]
```

Figure 9: Going back to starting point to start perimeter sweep



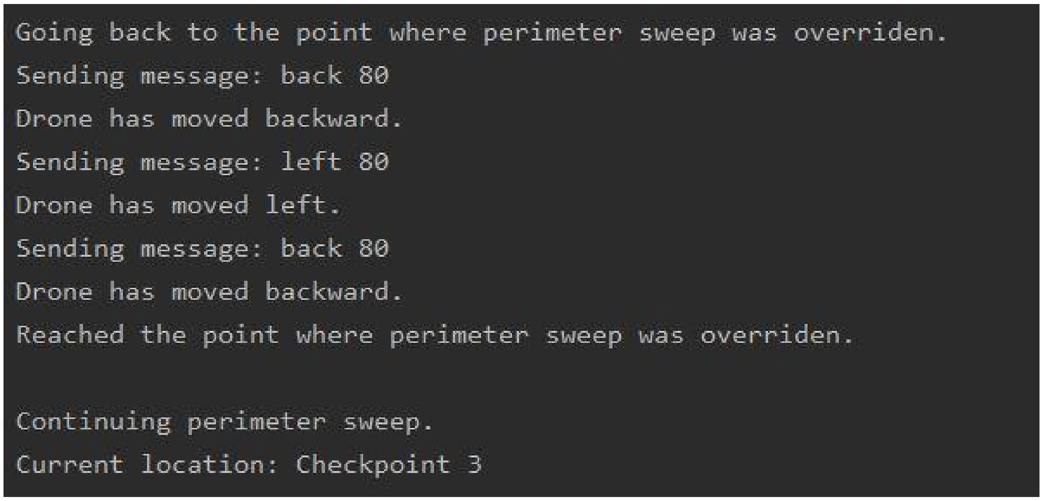
```
Perimeter sweep started.
Sending message: takeoff


From the charging base to the starting checkpoint of sweep pattern.

Sending message: forward 50
Sending message: ccw 150
Current location: Checkpoint 0

Sending message: cw 90
Sending message: forward 100
Arrived at current location: Checkpoint 1

Sending message: ccw 90
Sending message: forward 80
Arrived at current location: Checkpoint 2

Sending message: ccw 90
Sending message: forward 40
Arrived at current location: Checkpoint 3

Sending message: ccw 90
Sending message: forward 40
Arrived at current location: Checkpoint 4

Sending message: cw 90
Sending message: forward 60
Arrived at current location: Checkpoint 5

Returning to Checkpoint 0.
```

Figure 10: Doing Perimeter Sweep



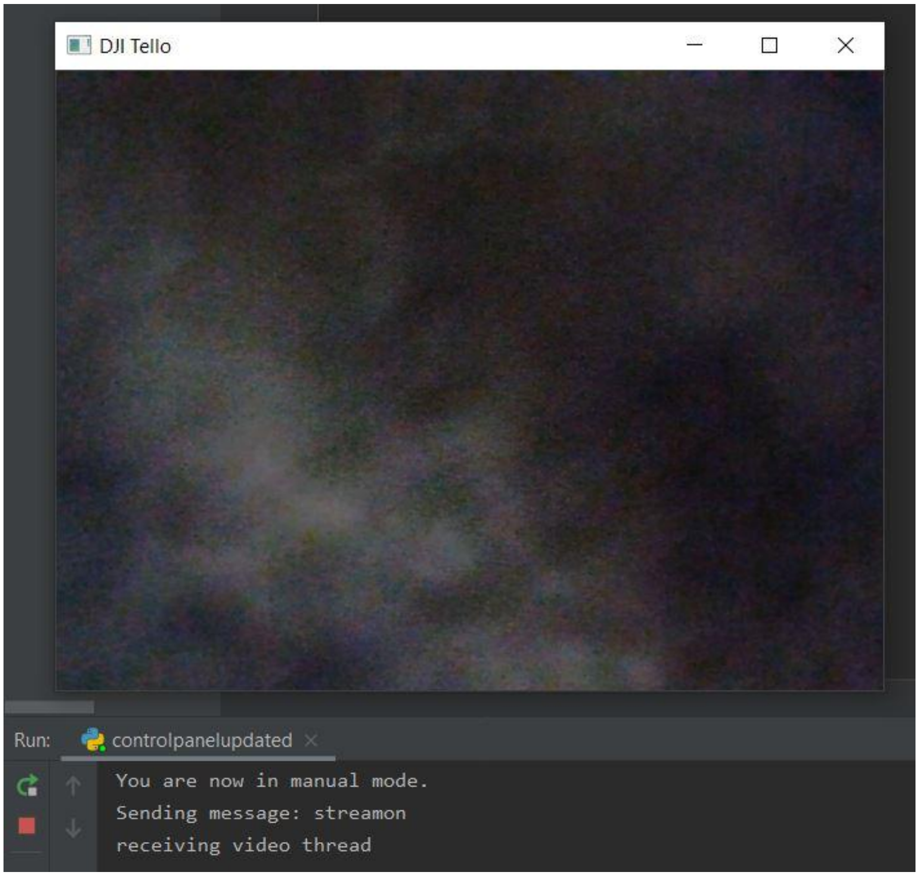
```
From the charging base to the starting checkpoint of sweep pattern.

Sending message: forward 50
Sending message: ccw 150
Current location: Checkpoint 0

Sending message: cw 90
Sending message: forward 100
Arrived at current location: Checkpoint 1

Sending message: ccw 90
Sending message: forward 80
Arrived at current location: Checkpoint 2

Sending message: ccw 90
Sending message: forward 40
Arrived at current location: Checkpoint 3

Manual mode initiated.
Sending message: forward 80
Drone has moved forward.
Sending message: right 80
Drone has moved right.
Sending message: forward 80
Drone has moved forward.
```

Figure 11: Overridden Perimeter Sweep



```
Going back to the point where perimeter sweep was overriden.
Sending message: back 80
Drone has moved backward.
Sending message: left 80
Drone has moved left.
Sending message: back 80
Drone has moved backward.
Reached the point where perimeter sweep was overriden.

Continuing perimeter sweep.
Current location: Checkpoint 3
```

Figure 12: Continuing Perimeter Sweep



```
You are now in manual mode.
Sending message: streamon
receiving video thread
```

Figure 13: Streaming Video

Source Code

tello.py

```
# This code is adopted from
https://learn.droneblocks.io/p/tello-drone-programming-with-python/
# Import the necessary modules
import socket
import threading
import time
import cv2

class Tello():

    def __init__(self):
        # IP and port of Tello
        self.tello_address = ('192.168.10.1', 8889)

        # IP and port of local computer
        self.local_address = ('', 9000)

        # Create a UDP connection that we'll send the command to
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

        # Bind to the local address and port
        self.sock.bind(self.local_address)

        # Create and start a listening thread that runs in the background
        # This utilizes our receive functions and will continuously monitor for incoming
messages
        self.receiveThread = threading.Thread(target=self.receive)
        self.receiveThread.daemon = True
        self.receiveThread.start()

    # Send the message to Tello and allow for a delay in seconds
    def send(self, message, delay):
        # Try to send the message otherwise print the exception
        try:
            self.sock.sendto(message.encode(), self.tello_address)
            print("Sending message: " + message)
    except Exception as e:
        print("Error sending: " + str(e))

    # Delay for a user-defined period of time
    time.sleep(delay)

# Receive the message from Tello
def receive(self):
    # Continuously loop and listen for incoming messages
    while True:
        # Try to receive the message otherwise print the exception
        try:
            response, ip_address = self.sock.recvfrom(128)
            print("Received message: " + response.decode(encoding='utf-8'))
        except Exception as e:
            # If there's an error close the socket and break out of the loop
            self.sock.close()
            print("Error receiving: " + str(e))
        break

def _receive_thread(self):
    while True:
        # Checking for Tello response, throws socket error
        try:
            self.response, ip = self.socket.recvfrom(1024)
            self.log[-1].add_response(self.response)
        except socket.error as exc:
            print('Socket error: {}'.format(exc))

def _video_thread(self):
    print("receiving video thread")
    # Creating stream capture object
    # cap = cv2.VideoCapture('udp://@'+self.tello_ip+':11111')
    cap = cv2.VideoCapture(0)
    # Runs while 'stream_state' is True
    while self.stream_state:
        ret, frame = cap.read()
        cv2.imshow('DJI Tello', frame)

        # Video Stream is closed if escape key is pressed
```

```
            k = cv2.waitKey(1) & 0xFF
            if k == 27:
                break
        cap.release()
        cv2.destroyAllWindows()
```

controlpanelupdated.py

```
import threading

from PyQt5.QtWidgets import QApplication

import tello
import time
import cv2

from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
import traceback, sys

# Create Billy
global billy
billy = tello.Tello()

# Put Tello into command mode
billy.send("command", 3)

# Used for going back
previ = 0

# directional counts
upcount = 0
downcount = 0
forwardcount = 0
backwardcount = 0
leftcount = 0
rightcount = 0
```

```
# Flip counts
flpl = 0
flpr = 0
flpf = 0
flpb = 0

# Rotation counts
clkw = 0
cclkw = 0

# Check if override button is clicked
override_chck = 0

# Check if takeoff button is clicked
takeoff_chck = 0

persweepclicked = 0

# Check if manual control is on or not
manucontrol = 1

referarr = []
referarr2 = []
print("Drone is in Manual Mode.")


class Worker(QRunnable):
    '''
    Worker thread

    Inherits from QRunnable to handler worker thread setup, signals and wrap-up.

    :param callback: The function callback to run on this worker thread. Supplied args
and
                     kwargs will be passed through to the runner.
    :type callback: function
    :param args: Arguments to pass to the callback function
    :param kwargs: Keywords to pass to the callback function
    '''

    def __init__(self, fn, *args, **kwargs):
        super(Worker, self).__init__()
        # Store constructor arguments (re-used for processing)
        self.fn = fn
        self.args = args
        self.kwargs = kwargs

    @pyqtSlot()
    def run(self):
        '''
        Initialise the runner function with passed args, kwargs.
        '''
        self.fn(*self.args, **self.kwargs)


class Ui_MainWindow(object):

    def __init__(self, *args, **kwargs):
        self.threadpool = QThreadPool()
        print("Multithreading with maximum %d threads" %
self.threadpool.maxThreadCount())

    # -------------------------------------------Manual Control
Methods---------------------------------------------------
    def streamon(self):
        billy.send("streamon", 3)
        billy.stream_state = True
        billy.video_thread = threading.Thread(target=billy._video_thread)
        billy.video_thread.daemon = True
        billy.video_thread.start()

    def streamoff(self):
        billy.send("streamoff", 3)

    # takeoff
    def takeoff(self):
        billy.send("takeoff", 7)
        global takeoff_chck
        takeoff_chck = 1
        print("Drone has taken off successfully.")

    # land
    def land(self):
        if manucontrol == 1 or override_chck == 1:
            billy.send("land", 3)
            print("Drone has landed successfully.")
        else:
            print("You are in Autonomous mode.")

        # Close the socket
        # billy.sock.close() [Causes error as it is not connected with real drone]

    # Stop in Air
    def stopinair(self):
        if manucontrol == 1 or override_chck == 1:
            billy.send("stop", 3)
            print("Drone is paused in air.")
        else:
            print("You are in Autonomous mode.")

    # --------------------------------------------Directional
Methods--------------------------------------------
    def forward(self):
        if override_chck == 1:
            global forwardcount
            forwardcount += 1
            referarr.append("f")
            billy.send("forward 80", 5)
            print("Drone has moved forward.")
            return
        elif manucontrol == 1:
            referarr2.append("f")
            billy.send("forward 80", 5)
            print("Drone has moved forward.")
            return
        elif manucontrol == 0:
            print("You are in Autonomous mode.")
            return
```

```
def back(self):
    if override_chck == 1:
        global backwardcount
        backwardcount += 1
        referarr.append("b")
        billy.send("back 80", 5)
        print("Drone has moved backward.")
        return
    elif manucontrol == 1:
        referarr2.append("b")
        billy.send("back 80", 5)
        print("Drone has moved backward.")
        return
    elif manucontrol == 0:
        print("You are in Autonomous mode.")
        return

def left(self):
    if override_chck == 1:
        global leftcount
        leftcount += 1
        referarr.append("l")
        billy.send("left 80", 5)
        print("Drone has moved left.")
        return
    elif manucontrol == 1:
        referarr2.append("l")
        billy.send("left 80", 5)
        print("Drone has moved left.")
        return
    elif manucontrol == 0:
        print("You are in Autonomous mode.")
        return

def right(self):
    if override_chck == 1:
        global rightcount
        referarr.append("r")
        rightcount += 1
```

```
            billy.send("right 80", 5)
            print("Drone has moved right.")
            return
        elif manucontrol == 1:
            referarr2.append("r")
            billy.send("right 80", 5)
            print("Drone has moved right.")
            return
        elif manucontrol == 0:
            print("You are in Autonomous mode.")
            return

    def up(self):
        if override_chck == 1:
            global upcount
            upcount += 1
            referarr.append("u")
            billy.send("up 50", 5)
            print("Drone has moved up.")
            return
        elif manucontrol == 1:
            referarr2.append("u")
            billy.send("up 50", 5)
            print("Drone has moved up.")
            return
        elif manucontrol == 0:
            print("You are in Autonomous mode.")
            return

    def down(self):
        if override_chck == 1:
            global downcount
            downcount += 1
            referarr.append("d")
            billy.send("down 50", 5)
            print("Drone has moved down.")
            return
        elif manucontrol == 1:
            referarr2.append("d")
            billy.send("down 50", 5)
            print("Drone has moved down.")
            return
        elif manucontrol == 0:
            print("You are in Autonomous mode.")
            return

    # ----------------------------------------Flip Methods----------------------------------------------
    def flipleft(self):
        if override_chck == 1:
            global flpl
            flpl += 1
            referarr.append("flpl")
            billy.send("flip l", 5)
            print("Drone has flipped left.")
            return
        elif manucontrol == 1:
            referarr2.append("flpl")
            billy.send("flip l", 5)
            print("Drone has flipped left.")
            return
        elif manucontrol == 0:
            print("You are in Autonomous mode.")
            return

    def flipright(self):
        if override_chck == 1:
            global flpr
            flpr += 1
            referarr.append("flpr")
            billy.send("flip r", 5)
            print("Drone has flipped right.")
            return
        elif manucontrol == 1:
            referarr2.append("flpr")
            billy.send("flip r", 5)
            print("Drone has flipped right.")
            return
        elif manucontrol == 0:
            print("You are in Autonomous mode.")
            return
```

```
def flipforward(self):
    if override_chck == 1:
        global flpf
        flpf += 1
        referarr.append("flpf")
        billy.send("flip f", 5)
        print("Drone has flipped forward.")
        return
    elif manucontrol == 1:
        referarr2.append("flpf")
        billy.send("flip f", 5)
        print("Drone has flipped forward.")
        return
    elif manucontrol == 0:
        print("You are in Autonomous mode.")
        return

def flipback(self):
    if override_chck == 1:
        global flpb
        flpb += 1
        referarr.append("flpb")
        billy.send("flip b", 5)
        print("Drone has flipped back.")
        return
    elif manucontrol == 1:
        referarr2.append("flpb")
        billy.send("flip b", 5)
        print("Drone has flipped back.")
        return
    elif manucontrol == 0:
        print("You are in Autonomous mode.")
        return

# ----------------------------------Rotational Methods------------------------------------
def cw(self):
    if override_chck == 1:
        global clkw
        clkw += 1
        referarr.append("clkw")
        billy.send("cw 30", 5)
        print("Drone has moved clock wise 30 degrees.")
        return
    elif manucontrol == 1:
        referarr2.append("clkw")
        billy.send("cw 30", 5)
        print("Drone has moved clock wise 30 degrees.")
        return
    elif manucontrol == 0:
        print("You are in Autonomous mode.")
        return

def ccw(self):
    if override_chck == 1:
        global cclkw
        cclkw += 1
        referarr.append("cclkw")
        billy.send("ccw 30", 5)
        print("Drone has moved counter clock wise 30 degrees.")
        return
    elif manucontrol == 1:
        referarr2.append("cclkw")
        billy.send("ccw 30", 5)
        print("Drone has moved counter clock wise 30 degrees.")
        return
    elif manucontrol == 0:
        print("You are in Autonomous mode.")
        return

# -----------------------------------Speed Methods------------------------------------------
def highspeed(self):
    if manucontrol == 1 or override_chck == 1:
        billy.send("speed 100", 5)
        print("Speed of drone is set to high.")
    else:
        print("You are in Autonomous mode.")

def lowspeed(self):
    if manucontrol == 1 or override_chck == 1:
        billy.send("speed 30", 5)
        print("Speed of drone is set to low.")
    else:
        print("You are in Autonomous mode.")

# ---------------------------- Overriding perimeter sweep and also going back to the place where drone stopped-----------------------------
def override(self):
    global referarr2
    referarr2=[]
    if persweepclicked == 0:
        print("Nothing to override.")
        return
    global referarr
    global override_chck
    global previ
    if override_chck == 0:
        override_chck = 1
        return
    elif override_chck == 1:
        override_chck = 0
        print()
        #print("i was " + str(previ))
        print("Going back to the point where perimeter sweep was overriden.")
        referarr.reverse()
        for r in range(0, len(referarr)):
            if referarr[r] == "u":
                self.down()
            elif referarr[r] == "d":
                self.up()
            elif referarr[r] == "l":
                self.right()
            elif referarr[r] == "r":
                self.left()
            elif referarr[r] == "f":
                self.back()
            elif referarr[r] == "b":
                self.forward()
            elif referarr[r] == "clkw":
                self.ccw()
            elif referarr[r] == "cclkw":
                self.cw()
            elif referarr[r] == "flpf":
                self.flipback()
            elif referarr[r] == "flpb":
                self.flipforward()
            elif referarr[r] == "flpl":
                self.flipright()
            elif referarr[r] == "flpr":
                self.flipleft()
        print("Reached the point where perimeter sweep was overriden.")
        referarr = []
        print()
        print("Continuing perimeter sweep.")
        self.persweepcont_exec()

# -----------------------------------------------Perimeter
sweep------------------------------------------------------
def persweep(self):
    global persweepclicked
    persweepclicked = 1
    global previ
    global referarr2
    global referarr
    print()
    print("Manual Mode switched off.")
    print("Manual Controls locked.")
    print()
    print("Autonomous Mode started.")
    global manucontrol
    if manucontrol == 1:
        print()
        print("Going back to the starting point.")
        referarr2.reverse()
        for r in range(0, len(referarr2)):
            if referarr2[r] == "u":
                self.down()
            elif referarr2[r] == "d":
                self.up()
            elif referarr2[r] == "l":
```

```
                self.right()
            elif referarr2[r] == "r":
                self.left()
            elif referarr2[r] == "f":
                self.back()
            elif referarr2[r] == "b":
                self.forward()
            elif referarr2[r] == "clkw":
                self.ccw()
            elif referarr2[r] == "cclkw":
                self.cw()
            elif referarr2[r] == "flpf":
                self.flipback()
            elif referarr2[r] == "flpb":
                self.flipforward()
            elif referarr2[r] == "flpl":
                self.flipright()
            elif referarr2[r] == "flpr":
                self.flipleft()
        print("Reached the starting point.")
        referarr2 = []
        print(referarr2)
        print()

    manucontrol -= 1
    # Travel to/from starting checkpoint 0 from/to the charging base
    frombase = ["forward", 50, "ccw", 150]
    tobase = ["ccw", 150, "forward", 50]

    # Flight path to Checkpoint 1 to 5 and back to Checkpoint 0 sequentially
    checkpoint = [[1, "cw", 90, "forward", 100], [2, "ccw", 90, "forward", 80], [3, "ccw",
90, "forward", 40],
                  [4, "ccw", 90, "forward", 40], [5, "cw", 90, "forward", 60], [0, "ccw", 90,
"forward", 40]]

    print("Perimeter sweep started.")

    # Send the takeoff command
    if takeoff_chck == 0:
        billy.send("takeoff", 7)
```

```
print("\n")

# Start at checkpoint 1 and print destination
print("From the charging base to the starting checkpoint of sweep pattern.\n")

billy.send(frombase[0] + " " + str(frombase[1]), 4)
billy.send(frombase[2] + " " + str(frombase[3]), 4)

print("Current location: Checkpoint 0 " + "\n")

# Billy's flight path
for i in range(len(checkpoint)):
    QApplication.processEvents()
    if override_chck == 1:
        print("Manual mode initiated.")
        manucontrol = 1
        return
    if i == len(checkpoint) - 1:
        print("Returning to Checkpoint 0. \n")
        previ = 0

    billy.send(checkpoint[i][1] + " " + str(checkpoint[i][2]), 4)
    billy.send(checkpoint[i][3] + " " + str(checkpoint[i][4]), 4)

    print("Arrived at current location: Checkpoint " + str(checkpoint[i][0]) + "\n")

    previ = i
    time.sleep(4)

# Reach back at Checkpoint 0
print("Complete sweep. Return to charging base.\n")
billy.send(tobase[0] + " " + str(tobase[1]), 4)
billy.send(tobase[2] + " " + str(tobase[3]), 4)

# Turn to original direction before land
print("Turn to original direction before land.\n")
billy.send("cw 180", 4)

# Land
```

```
        billy.send("land", 3)

        # Close the socket
        # billy.sock.close() [Causes error as it is not connected with real drone]
        print("Perimeter sweep completed successfully.")
        print("Autonomous mode switched off.")
        print("You are now in manual mode.")
        manucontrol += 1
        persweepclicked = 0
        referarr2=[]
        referarr=[]

    def persweep_exec(self):
        # Pass the function to execute
        worker = Worker(self.persweep)  # Any other args, kwargs are passed to the run
function

        # Execute
        self.threadpool.start(worker)

    def persweepcontinue(self):
        global referarr
        global referarr2
        global previ
        # Travel to/from starting checkpoint 0 from/to the charging base
        frombase = ["forward", 50, "ccw", 150]
        tobase = ["ccw", 150, "forward", 50]

        # Flight path to Checkpoint 1 to 5 and back to Checkpoint 0 sequentially
        checkpoint = [[1, "cw", 90, "forward", 100], [2, "ccw", 90, "forward", 80], [3, "ccw",
90, "forward", 40],
                      [4, "ccw", 90, "forward", 40], [5, "cw", 90, "forward", 60], [0, "ccw", 90,
"forward", 40]]
        i = previ
        print("Current location: Checkpoint " + str(checkpoint[i][0]) + "\n")
        i += 1
        # Billy's flight path
        while i < len(checkpoint):
            #print("test i" + str(i))
            if i == len(checkpoint) - 1:
                print("Returning to Checkpoint 0. \n")

            billy.send(checkpoint[i][1] + " " + str(checkpoint[i][2]), 4)
            billy.send(checkpoint[i][3] + " " + str(checkpoint[i][4]), 4)

            print("Arrived at current location: Checkpoint " + str(checkpoint[i][0]) + "\n")
            i += 1
            time.sleep(4)

        # Reach back at Checkpoint 0
        print("Complete sweep. Return to charging base.\n")
        billy.send(tobase[0] + " " + str(tobase[1]), 4)
        billy.send(tobase[2] + " " + str(tobase[3]), 4)

        # Turn to original direction before land
        print("Turn to original direction before land.\n")
        billy.send("cw 180", 4)

        # Land
        billy.send("land", 3)

        # Close the socket
        # billy.sock.close() [Causes error as it is not connected with real drone]
        print("Perimeter sweep completed successfully.")
        print("Autonomous mode switched off.")
        print("You are now in manual mode.")
        global persweepclicked
        persweepclicked = 0
        referarr2=[]
        referarr=[]

    def persweepcont_exec(self):
        # Pass the function to execute
        worker = Worker(self.persweepcontinue)  # Any other args, kwargs are passed to
the run function

        # Execute
        self.threadpool.start(worker)

    # ----------------------------------------emergency----------------------------
```

```
def emergency(self):
    if manucontrol == 1 or override_chck == 1:
        # Send the emergency stop command
        billy.send("emergency", 3)
        print("Emergency mode initiated, motor stopped.")
    else:
        print("You are in Autonomous mode.")

def setupUi(self, MainWindow):
    MainWindow.setObjectName("MainWindow")
    MainWindow.resize(800, 600)
    MainWindow.setStyleSheet("background-color:rgb(248, 249, 255)")
    self.centralwidget = QtWidgets.QWidget(MainWindow)
    self.centralwidget.setObjectName("centralwidget")
    self.fbtn = QtWidgets.QPushButton(self.centralwidget)
    self.fbtn.setGeometry(QtCore.QRect(490, 40, 81, 111))
    self.fbtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
    self.fbtn.setStyleSheet("background-color: rgb(139, 139, 139);\n"
                            "color: white;")
    self.fbtn.setObjectName("fbtn")
    self.bcbtn = QtWidgets.QPushButton(self.centralwidget)
    self.bcbtn.setGeometry(QtCore.QRect(490, 230, 81, 111))
    self.bcbtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
    self.bcbtn.setStyleSheet("background-color: rgb(139, 139, 139);\n"
                             "color: white;")
    self.bcbtn.setObjectName("bcbtn")
    self.rbtn = QtWidgets.QPushButton(self.centralwidget)
    self.rbtn.setGeometry(QtCore.QRect(570, 150, 111, 81))
    self.rbtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
    self.rbtn.setStyleSheet("background-color: rgb(139, 139, 139);\n"
                            "color: white;")
    self.rbtn.setObjectName("rbtn")
    self.lbtn = QtWidgets.QPushButton(self.centralwidget)
    self.lbtn.setGeometry(QtCore.QRect(380, 150, 111, 81))
    self.lbtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
    self.lbtn.setStyleSheet("background-color: rgb(139, 139, 139);\n"
                            "color: white;")
    self.lbtn.setObjectName("lbtn")
    self.peribtn = QtWidgets.QPushButton(self.centralwidget)
    self.peribtn.setGeometry(QtCore.QRect(10, 450, 211, 91))
```

```
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.peribtn.setFont(font)
self.peribtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.peribtn.setStyleSheet("background-color: rgb(0, 170, 255);\n"
                           "color: white;\n"
                           "border-radius: 20px;")
self.peribtn.setObjectName("peribtn")
self.overrbtn = QtWidgets.QPushButton(self.centralwidget)
self.overrbtn.setGeometry(QtCore.QRect(590, 450, 201, 91))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.overrbtn.setFont(font)
self.overrbtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.overrbtn.setStyleSheet("background-color: rgb(255, 206, 56);\n"
                            "color: white;\n"
                            "border-radius: 20px;")
self.overrbtn.setObjectName("overrbtn")
self.labelmanu = QtWidgets.QLabel(self.centralwidget)
self.labelmanu.setGeometry(QtCore.QRect(330, -10, 151, 51))
font = QtGui.QFont()
font.setPointSize(12)
font.setBold(True)
font.setWeight(75)
self.labelmanu.setFont(font)
self.labelmanu.setObjectName("labelmanu")
self.envidbtn = QtWidgets.QPushButton(self.centralwidget)
self.envidbtn.setGeometry(QtCore.QRect(320, 410, 171, 61))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.envidbtn.setFont(font)
self.envidbtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.envidbtn.setStyleSheet("background-color: rgb(68, 68, 68);\n"
                            "color: white;\n"
                            "border-radius: 20px;")
self.envidbtn.setObjectName("envidbtn")
self.takeoffbtn = QtWidgets.QPushButton(self.centralwidget)
self.takeoffbtn.setGeometry(QtCore.QRect(10, 60, 121, 71))
font = QtGui.QFont()
font.setPointSize(12)
font.setBold(True)
font.setWeight(75)
self.takeoffbtn.setFont(font)
self.takeoffbtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.takeoffbtn.setStyleSheet("background-color: rgb(0, 202, 0);\n"
                              "color: white;\n"
                              "border-radius: 70px;")
self.takeoffbtn.setObjectName("takeoffbtn")
self.emergstopbtn = QtWidgets.QPushButton(self.centralwidget)
self.emergstopbtn.setGeometry(QtCore.QRect(10, 370, 211, 51))
font = QtGui.QFont()
font.setPointSize(12)
font.setBold(True)
font.setWeight(75)
self.emergstopbtn.setFont(font)
self.emergstopbtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.emergstopbtn.setStyleSheet("background-color: rgb(255, 32, 32);\n"
                                "color: white;\n"
                                "border-radius: 70px;")
self.emergstopbtn.setObjectName("emergstopbtn")
self.upbtn = QtWidgets.QPushButton(self.centralwidget)
self.upbtn.setGeometry(QtCore.QRect(730, 40, 51, 111))
self.upbtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.upbtn.setStyleSheet("background-color: rgb(139, 139, 139);\n"
                         "color: white;")
self.upbtn.setObjectName("upbtn")
self.downbtn = QtWidgets.QPushButton(self.centralwidget)
self.downbtn.setGeometry(QtCore.QRect(730, 160, 51, 111))
self.downbtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.downbtn.setStyleSheet("background-color: rgb(139, 139, 139);\n"
                           "color: white;")
self.downbtn.setObjectName("downbtn")
self.cwbtn = QtWidgets.QPushButton(self.centralwidget)
self.cwbtn.setGeometry(QtCore.QRect(610, 310, 81, 31))
self.cwbtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.cwbtn.setStyleSheet("background-color: rgb(139, 139, 139);\n"
                    "color: white;")
self.cwbtn.setObjectName("cwbtn")
self.ccwbtn = QtWidgets.QPushButton(self.centralwidget)
self.ccwbtn.setGeometry(QtCore.QRect(700, 310, 81, 31))
self.ccwbtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.ccwbtn.setStyleSheet("background-color: rgb(139, 139, 139);\n"
                    "color: white;")
self.ccwbtn.setObjectName("ccwbtn")
self.pauseBtn = QtWidgets.QPushButton(self.centralwidget)
self.pauseBtn.setGeometry(QtCore.QRect(10, 160, 121, 61))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.pauseBtn.setFont(font)
self.pauseBtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.pauseBtn.setStyleSheet("background-color: rgb(255, 206, 56);\n"
                      "color: white;\n"
                      "border-radius: 20px;")
self.pauseBtn.setObjectName("pauseBtn")
self.disvidbtn = QtWidgets.QPushButton(self.centralwidget)
self.disvidbtn.setGeometry(QtCore.QRect(320, 480, 171, 61))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.disvidbtn.setFont(font)
self.disvidbtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.disvidbtn.setStyleSheet("background-color: rgb(68, 68, 68);\n"
                       "color: white;\n"
                       "border-radius: 20px;")
self.disvidbtn.setObjectName("disvidbtn")
self.landbtn = QtWidgets.QPushButton(self.centralwidget)
self.landbtn.setGeometry(QtCore.QRect(10, 260, 121, 71))
font = QtGui.QFont()
font.setPointSize(12)
font.setBold(True)
font.setWeight(75)
```

```
self.landbtn.setFont(font)
self.landbtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.landbtn.setStyleSheet("background-color: rgb(0, 202, 0);\n"
                    "color: white;\n"
                    "border-radius: 70px;")
self.landbtn.setObjectName("landbtn")
self.fliplbtn = QtWidgets.QPushButton(self.centralwidget)
self.fliplbtn.setGeometry(QtCore.QRect(210, 60, 91, 41))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.fliplbtn.setFont(font)
self.fliplbtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.fliplbtn.setStyleSheet("background-color: rgb(68, 68, 68);\n"
                    "color: white;\n"
                    "border-radius: 20px;")
self.fliplbtn.setObjectName("fliplbtn")
self.fliprbtn = QtWidgets.QPushButton(self.centralwidget)
self.fliprbtn.setGeometry(QtCore.QRect(310, 60, 91, 41))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.fliprbtn.setFont(font)
self.fliprbtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.fliprbtn.setStyleSheet("background-color: rgb(68, 68, 68);\n"
                    "color: white;\n"
                    "border-radius: 20px;")
self.fliprbtn.setObjectName("fliprbtn")
self.flipfwbtn = QtWidgets.QPushButton(self.centralwidget)
self.flipfwbtn.setGeometry(QtCore.QRect(210, 130, 91, 41))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.flipfwbtn.setFont(font)
self.flipfwbtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.flipfwbtn.setStyleSheet("background-color: rgb(68, 68, 68);\n"
                     "color: white;\n"
```

```
                        "border-radius: 20px;")
self.flipfwbtn.setObjectName("flipfwbtn")
self.flipbwbtn = QtWidgets.QPushButton(self.centralwidget)
self.flipbwbtn.setGeometry(QtCore.QRect(210, 180, 91, 41))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.flipbwbtn.setFont(font)
self.flipbwbtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.flipbwbtn.setStyleSheet("background-color: rgb(68, 68, 68);\n"
                        "color: white;\n"
                        "border-radius: 20px;")
self.flipbwbtn.setObjectName("flipbwbtn")
self.hspeedbtn = QtWidgets.QPushButton(self.centralwidget)
self.hspeedbtn.setGeometry(QtCore.QRect(200, 280, 111, 61))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.hspeedbtn.setFont(font)
self.hspeedbtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.hspeedbtn.setStyleSheet("background-color: rgb(0, 170, 255);\n"
                        "color: white;\n"
                        "border-radius: 20px;")
self.hspeedbtn.setObjectName("hspeedbtn")
self.lspeedbtn = QtWidgets.QPushButton(self.centralwidget)
self.lspeedbtn.setGeometry(QtCore.QRect(330, 280, 111, 61))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.lspeedbtn.setFont(font)
self.lspeedbtn.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.lspeedbtn.setStyleSheet("background-color: rgb(0, 170, 255);\n"
                        "color: white;\n"
                        "border-radius: 20px;")
self.lspeedbtn.setObjectName("lspeedbtn")
MainWindow.setCentralWidget(self.centralwidget)
self.menubar = QtWidgets.QMenuBar(MainWindow)
```

```
        self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 26))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

        # ------------------------------Connecting takeoff method to takeoff
button---------------------------------
        self.takeoffbtn.clicked.connect(self.takeoff)
        # -------------------------------Connecting land method to land
button-----------------------------------
        self.landbtn.clicked.connect(self.land)
        # ------------------------------Connecting persweep method to persweep
button---------------------------------
        self.peribtn.clicked.connect(self.persweep_exec)
        # ------------------------------Connecting emergency method to emergstop
button-------------------------------
        self.emergstopbtn.clicked.connect(self.emergency)
        # ------------------------------Connecting pause method to pause
button---------------------------------
        self.pauseBtn.clicked.connect(self.stopinair)
        # ------------------------------Connecting directional method to directional
buttons--------------------------
        self.upbtn.clicked.connect(self.up)
        self.downbtn.clicked.connect(self.down)
        self.fbtn.clicked.connect(self.forward)
        self.bcbtn.clicked.connect(self.back)
        self.lbtn.clicked.connect(self.left)
        self.rbtn.clicked.connect(self.right)
        # ------------------------------Connecting flip methods to flip buttons--------------------------
        self.flipfwbtn.clicked.connect(self.flipforward)
        self.flipbwbtn.clicked.connect(self.flipback)
        self.fliprbtn.clicked.connect(self.flipright)
        self.fliplbtn.clicked.connect(self.flipleft)
        # ------------------------------Connecting Rotation methods to rotation
buttons--------------------------
```

```
        self.cwbtn.clicked.connect(self.cw)
        self.ccwbtn.clicked.connect(self.ccw)
        # ------------------------------Connecting Speed methods to speed
buttons-------------------------
        self.hspeedbtn.clicked.connect(self.highspeed)
        self.lspeedbtn.clicked.connect(self.lowspeed)

        # ------------------------------Connecting override method to override
button-------------------------
        self.overrbtn.clicked.connect(self.override)

        # -------------------------------Video
Streaming------------------------------------------------------
        self.envidbtn.clicked.connect(self.streamon)
        self.disvidbtn.clicked.connect(self.streamoff)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Tello Drone Control
Panel"))
        self.fbtn.setText(_translate("MainWindow", "FORWARD"))
        self.bcbtn.setText(_translate("MainWindow", "BACKWARD"))
        self.rbtn.setText(_translate("MainWindow", "RIGHT"))
        self.lbtn.setText(_translate("MainWindow", "LEFT"))
        self.peribtn.setText(_translate("MainWindow", "PERIMETER SWEEP"))
        self.overrbtn.setText(_translate("MainWindow", "OVERRIDE ROUTE"))
        self.labelmanu.setText(_translate("MainWindow", "Control Panel"))
        self.envidbtn.setText(_translate("MainWindow", "Enable Video Stream"))
        self.takeoffbtn.setText(_translate("MainWindow", "TAKEOFF"))
        self.emergstopbtn.setText(_translate("MainWindow", "Stop"))
        self.upbtn.setText(_translate("MainWindow", "UP"))
        self.downbtn.setText(_translate("MainWindow", "Down"))
        self.cwbtn.setText(_translate("MainWindow", "CW"))
        self.ccwbtn.setText(_translate("MainWindow", "CCW"))
        self.pauseBtn.setText(_translate("MainWindow", "Pause"))
        self.disvidbtn.setText(_translate("MainWindow", "Disable Video Stream"))
        self.landbtn.setText(_translate("MainWindow", "Land"))
        self.fliplbtn.setText(_translate("MainWindow", "Flip Left"))
        self.fliprbtn.setText(_translate("MainWindow", "Flip Right"))
        self.flipfwbtn.setText(_translate("MainWindow", "Flip FW"))
        self.flipbwbtn.setText(_translate("MainWindow", "Flip BW"))
        self.hspeedbtn.setText(_translate("MainWindow", "High Speed"))
        self.lspeedbtn.setText(_translate("MainWindow", "Low Speed"))


if __name__ == "__main__":
    import sys

    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

Conclusion

We have successfully completed our Tello EDU drone project. During the project we got to learn how we can use python to control a drone. Although we were unable to implement the vision positioning system as it is embedded inside the drone and we did not get the chance to use the drone because of the MCO, however, we were able to implement other functionalities like video streaming feature, perimeter sweep and override function in the control panel. As a result we now have hands-on experience of working on a real time system.

Copy of Group Contract

References

Olson, E. (2020, March 13). Your next security guard might be a drone. Retrieved January 18, 2021, from https://www.securityinfowatch.com/perimeter-security/robotics/unmanned-aerial-vehicles-drones/article/21127483/your-next-security-guard-might-be-a-drone

Tello Drone User Manuals. (n.d.). Retrieved January 20, 2021, from http://tellohq.com/tello-drone-user-manuals/

UAVLance. (2016, June 25). An Overview Of UAV Hardware Components and Software. Retrieved January 20, 2021, from https://medium.com/@UAVLance/an-overview-of-uav-hardware-components-and-software-2df983222e31