

SENIOR PROJECT



AUTONOMOUS UAV

Submitted by:
Koray ÖZYURT

Faculty of Engineering
Antalya Bilim University
Spring 2018

ABSTRACT

This paper serves as the senior project report for computer engineering department. UAV (Unmanned Aircraft Vehicles) are devices that provide a multitude of different services to their users. It is a growing field of technology and is used in a wide variety of tasks such as military based operations, scouting missions and transportation tasks. UAV are slowly reaching a wide variety of users ranging from kids to adults with each using it for different purposes and tasks.

The project's focus is on the use of UAV for transportation, facial recognition and scouting missions. The first phase was mainly about acquiring the needed parts and where to find them, researching what we would need to implement facial recognition and the different designs that could be made for the drone. The second phase focuses on building the drone and structuring it properly, mainly the different parts used in making the drone and the facial recognition software implementation. Finally, it addresses the issues that were faced prior to and after building the drone and how they were overcome. The resulted UAV is a machine capable of fulfilling the role it was made for, facial recognition, transportation and scouting missions, but can also be used for additional tasks or activities depending on the user's creativity and how they wish to use the UAV. The UAV can also be further customized and altered so that it may be used for other missions and activities besides the previously stated tasks the UAV was made for.

Contents

1.Introduction	1
1.1 Project Statement	1
1.2 Goal and Motivation	1
1.2 Proposed Project	1
2. Development.....	2
2.1 Proposed Materials	2
3. Design.....	2
2.1 Circuit Design.....	4
3.2 Drone Code Structure	5
3.3 Telemetry Code Structure	6
3.4 OpenCV.....	9
3.4.1 DataSetCreator.py.....	9
3.4.2 DataSetTrainer.py	10
3.4.3 FaceDetection.py.....	11
4. Components	12
4.1 Drone Components	12
4.1.1 Raspberry Pi.....	13
4.1.2 Raspi-Camera	13
4.1.3 Brushless Motor	13
4.1.4 Electronic Speed Controller.....	16
4.1.5 IMU	16
4.1.6 GPS	19
4.1.7 HCSR-04.....	19
4.1.8 Li-po Module	20
4.1.9 Relay Module	21
4.2 Telemetry Components.....	21
4.2.1 Backend Package Structure.....	21
3.2.2 User Class	22
4.2.3 Tools	22
4.2.4 Network.....	23
4.2.5 Frontend components.....	24

4.2.6 Google Maps	25
4.2.7 Gyroscope.....	27
4.2.8 Camera	28
4.2.9 Motor Offset.....	29
4.2.10 Web Socket JS	30
4.2.11 index.js	31
4.2.12 Search Page	31
4.2.13 Settings Page	31
5. Final	32
6. Constraints	33
7. Conclusion	33
8. References.....	34

1.Introduction

An unmanned aerial vehicle (UAV), commonly known as a drone, is an aircraft without a human pilot aboard. UAVs are a component of an unmanned aircraft system (UAS); which include a UAV, a ground-based controller, and a system of communications between the two. The flight of UAVs may operate with various degrees of autonomy: either under remote control by a human operator or autonomously by onboard computers. UAV (Unmanned Aircraft Vehicles) are devices that provide a multitude of different services to their users. They are slowly changing the way we handle a multitude of different operations ranging from the most basic of missions such as exploring an area to the most complex such as military and police operations.

1.1 Project Statement

UAV are used in a multitude of purposes and tasks with the main aim of reducing risks to the users, reducing the workload or providing the user with the necessary aid in order to complete their goals and accomplish their missions. UAV come in various models , shapes and forms each designed with a set goal or purpose and cannot fulfill more than one need at a time.

1.2 Goal and Motivation

The main goal of this project is to create a drone that focuses on 3 set defined capabilities; Transportation, facial recognition and scouting. In addition to being able to fulfill those attributes it can be further enhanced or altered to the user's liking in order to fulfill a different task or accomplish missions it wasn't made for. Lastly it should be autonomous in order to provide the users with a hassle-free experience but can control the drone manually if they so wish to.

1.2 Proposed Project

The proposed solution is a drone that accomplishes the already set goals as follows:

- Transportation: The drone is able to transport items from one set area to another under a set weight limit.
- Facial Recognition: The drone is able to identify the faces of people whose images have been provided.
- Scouting: The drone is able to scout various locations.
- Telemetry: A telemetry system is provided for the drone in order to fly properly and identify its telemetry in various conditions.
- Upgradability: The drone can be upgraded so that it may fulfill other roles and missions.
- Raspberry Pi: In order for the drone to be able to handle all these operations a Raspberry Pi is used to handle the processing of all of the necessary operations and calculations.
- Autonomous: The drone is able to fly to set destinations and locations.

2. Development

The methods that were used throughout the development process followed the Test-driven development approach. This approach has become very popular for its crucial importance in providing those bits of test verification after each step, which guarantee that the end product is working as it should be. This is split in to two sections, the first being the hardware side of the drone and the second being the software side of the drone. For the hardware side, this was accomplished through buying the necessary parts , then settling on a design and finally implementing it. As for the software side of the equation, it was accomplished by iterations of writing tests of the codes to be implemented, then testing them and later on implementing them, the software in use is broken down into segments as well , the first being the software used in the drone to fly it and the other being the software used to operate the facial recognition software. For the final phase of the project , after the software for the drone is finished and the hardware side of it is working as intended , the facial recognition software is implemented to finalize the project.

2.1 Proposed Materials

These are the main materials that were used in the project are the following:

- OpenCV: A python library that is used for computer vision related operations.
- Raspberry Pi: A single board computer used for processing the different operations and necessary calculations.
- Telemetry System: A system which is used to give details regarding the various status attributes of the drone.
- Electrical Circuit: A circuit used to connect the necessary parts of the drone.

3. Design

In this project, the drone is designed to lift maximum 4 kilograms weigh calculated as equation 3.1 to 3.6.

$$Power = Propeller Constant * rpm^{power factor} \quad (3.1)$$

Where propeller constant of 0.015 and power factor of 3.2 given a rotational speed of 10,000 rpm.

$$T = \frac{\pi}{4} D^2 p v \Delta v \quad (3.2)$$

T = thrust[N]

D = propeller diameter [m]

V = velocity of air at the propeller [m/s]

Δv = velocity of air accelerated by propeller [m/s]

P = density of air [1.225 kg/m³]

$$\text{Velocity of the air at the propeller is } v = \frac{1}{2} \Delta v \quad (3.3)$$

$$T = \frac{\pi}{8} D^2 p (\Delta v)^2$$

According to power equation, Δv can be eliminated, thrust equation can be calculated as;

$$P = \frac{T\Delta v}{2} \rightarrow \Delta v = \frac{2P}{T} \quad (3.4)$$

$$T = \left[\frac{\pi}{2} D^2 p P^2 \right]^{1/3} \quad (3.5)$$

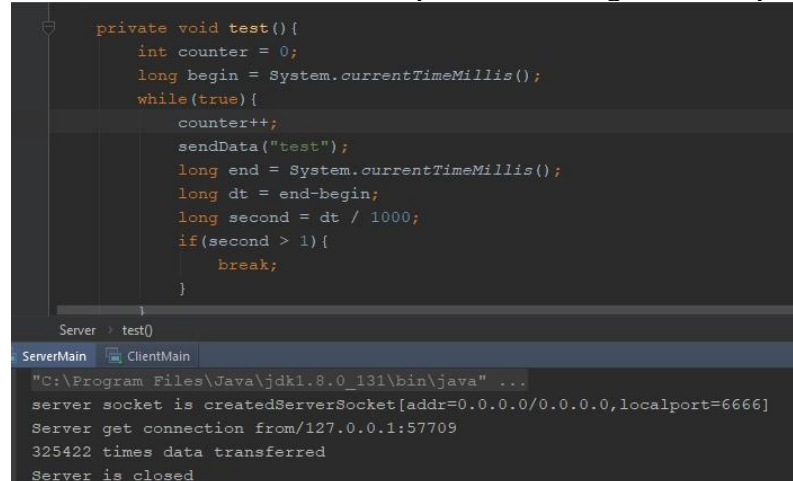
Finally, the Newton's law $F=ma$ formula is known. So;

$$m = \frac{\left[\frac{\pi}{2} D^2 p P^2 \right]^{1/3}}{g} \quad (3.6)$$

Where $g = 9.81 \text{ m/s}^2$

According to these are the calculations, 1400 kV motor and (25.4 x 11.43 cm) propeller could lift 8020.33 grams weigh. This means, drone can get altitude when carries 4 kilogram item. 1400 kV motor needs 2s or 3s li-po battery. Lift equation is calculated by around of 10,750 rpm (according to 2212 Brushless test record). So, 3s li-po battery will be used with 25A electronic speed controller. All the material will be explained in the following report.

The drone is designed as a server to support multiple user handling. Raspberry Pi, collect the all data to control the drone, and send the requested data at the telemetry and in another thread, it waits new command from user. The sending and receiving network threads are different because raspberry pi sends data around of 325,422 times as the test is shown in Figure 3.1 but telemetry does not send data each time. So, the separate threading increases performance.



```

private void test(){
    int counter = 0;
    long begin = System.currentTimeMillis();
    while(true){
        counter++;
        sendData("test");
        long end = System.currentTimeMillis();
        long dt = end-begin;
        long second = dt / 1000;
        if(second > 1){
            break;
        }
    }
}

Server > test()
ServerMain ClientMain
"C:\Program Files\Java\jdk1.8.0_131\bin\java" ...
server socket is createdServerSocket[addr=0.0.0.0/0.0.0.0,localport=6666]
Server get connection from/127.0.0.1:57709
325422 times data transferred
Server is closed

```

Figure 3.1 data transfer counter for in a second

Also, motor there is a motor threading because motor control should not wait any delay. Motor controlling owns most priority. Finally, fourth thread controls camera. Camera get 24 image in a second, then convert them OpenCV objects to do image processing, then convert into bytes to send telemetry screen via wi-fi. Also, the bytes are built as "mjpeg" object. Motion JPEG increase the performance of camera. There will be high fps compared to jpg.

Telemetry uses javaEE, JavaScript technology to display sensor data to user. There is three thread to get data, send command and receive camera frames. Gps location is shown by

google maps, gyroscope data is shown by three-dimensional JavaScript canvas screen, and motor throttle status is shown by JavaScript canvas screen.

2.1 Circuit Design

The materials are described as bellow;

Brushless Motor: High rpm motor. In this project, 1400 kV motor is used.

Electronic Speed Controller: Brushless motor is cannot be connected to microprocessor or microcomputer directly. Electronic speed controller get signal from microprocessor or microcomputer, then control the motor by using li-po battery power. For this project, electronic speed controller must handle 30 amperes and 11.1 voltage.

Li-po battery: Li-po battery can provide high current. 1400 kV motor needs 2s or 3s li-po battery and this project will use 3s li-po battery. 1s means 3.7 voltage and 3s means 11.1 voltage.

Gyroscope: To get -x, -y, -z direction. In this project, MPU-6050 is used. MPU-6050 is an IMU (Inertial Measurement Unit) sensor. MPU-6050 can measure -x, -y, -z gyro values, accelerator values and temperature. It is very cheap but affected by noise.

GPS: Provide us the location in the current world using NASA satellites. GY-NEO6MV2 is used. This sensor has ± 5 -meter error range.

Relay Module: While brushless motors are configuring, li-po battery must be disconnected and after the configuration, li-po battery must be connected again. With relay module, this procedure would be done automatically.

Camera: Any camera can be implemented. For now, raspi-camera is using.

Micro-Computer: Raspberry Pi 3 model B+ is using because of the high performance 4 cores and it has 40 general purpose input/output pins.

All the materials will be explained in more detail. the circuit design is as Bellow in figure 3.1.1 with pin layout in figure 3.1.2, mpu6050 connection in figure 3.1.3 and GPS connection figure 3.1.4 ;

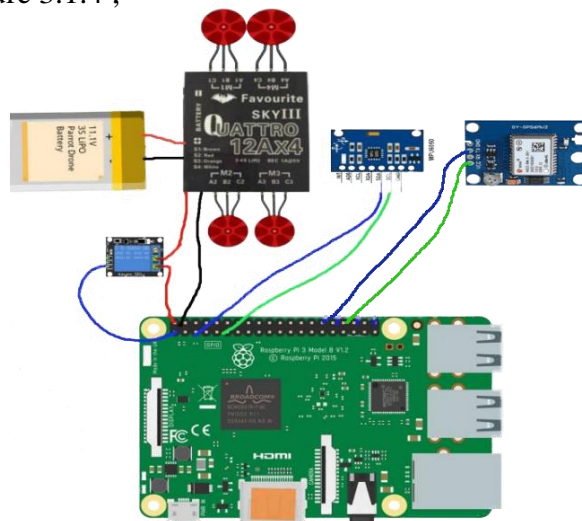


Figure 3.1.1

Pin Layout is:

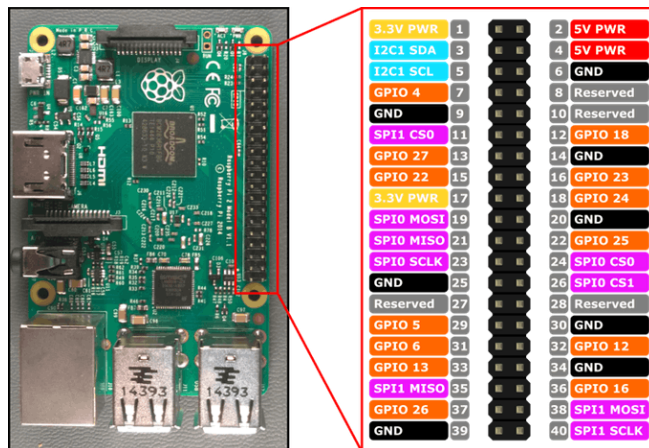


Figure 3.1.2

Relay = GPIO 20

Forward Right Motor = GPIO 12

Forward Left Motor = GPIO 18

Back Right Motor = GPIO 21

Back Left Motor = GPIO 25

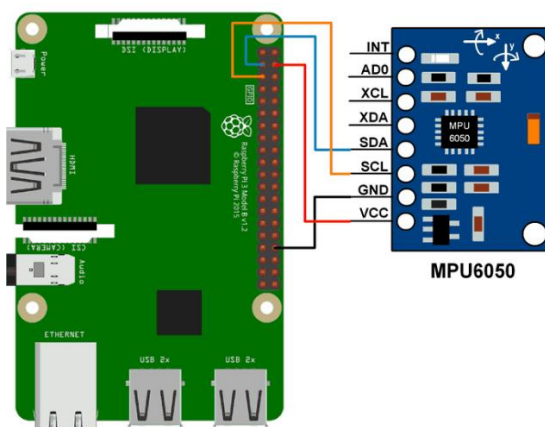


Figure 3.1.3

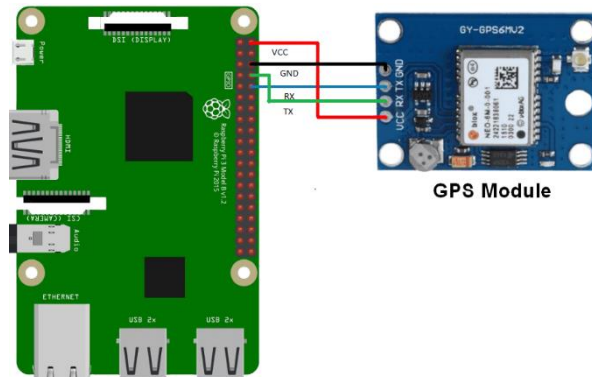


Figure 3.1.4

3.2 Drone Code Structure

Raspberry Pi using four threads. They are motor thread, send data thread, get data thread and camera thread. Send data and get data threads are would been in a single thread, but this structure has more performance because in generally, get data thread is in wait mode but send data thread run around of 355,000 times in a second as tested.

Motor Thread: Controls the motors using MPU-6050 measurements and given command by user. The x and y rotation values will be used to provide pid control algorithm and the command will decide that drone movement.

Camera Thread: Gets 24 capture in a second then convert into OpenCV object to do image processing. Then the thread convert the image into bytes to build “mjpeg” file then send it telemetry.

Server Thread: There will be two socket class. One of them send data and the other one is get data. The send data thread is used more than get data. So, they are separate thread because in generally, get data thread is in wait mode. The threads are named as “UAVServerTX” and “UAVServerRX”.

The class diagram is shown in figure 3.2.1.

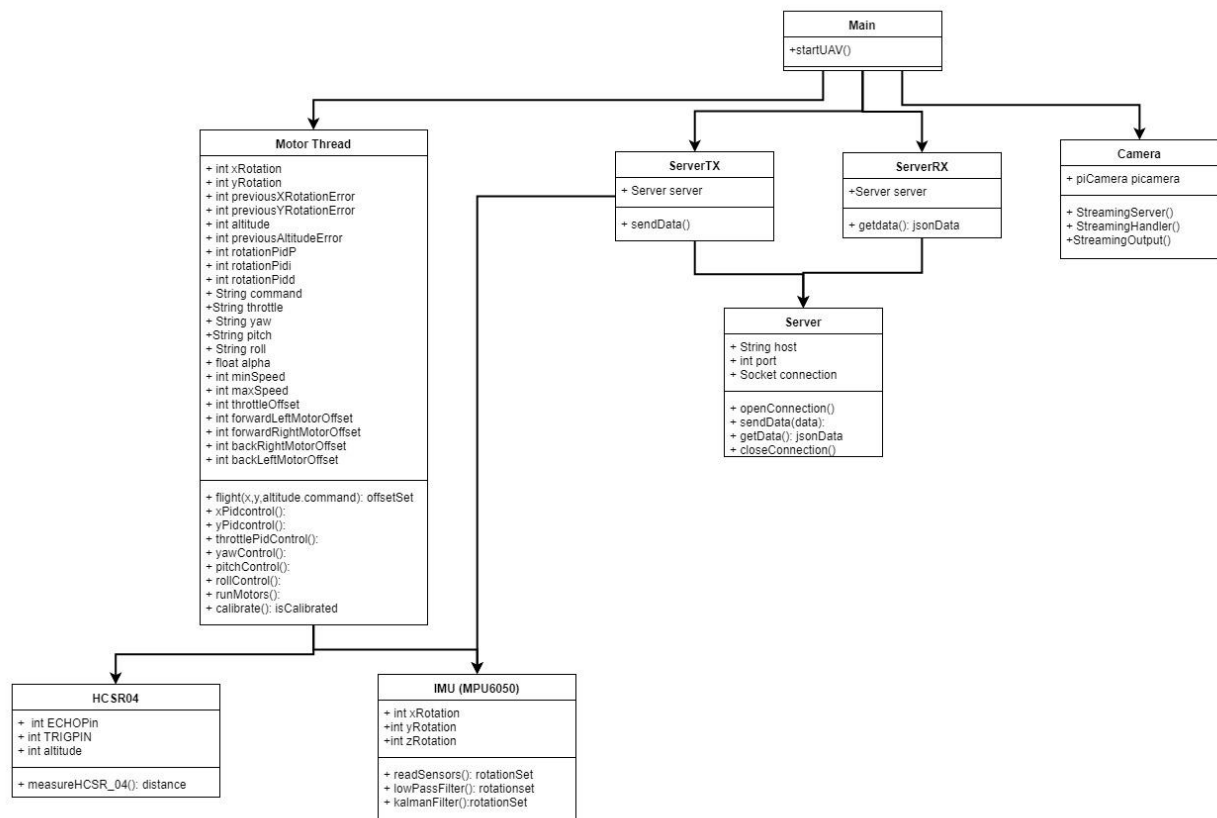


Figure 3.2.1

3.3 Telemetry Code Structure

Telemetry system is based on JavaEE technology. JavaScript, jQuery, Ajax, MySQL, HTML and CSS technologies build the whole system. The telemetry screen use three thread to get camera data, sensor data and to send command. In the next version, depend on the user role user cannot handle the drone but can track. In this version, user can track and handle the system if the user logged the system with username and password. In addition, the database includes log information to detect any problem. The MySQL tables are shown as figure 3.3.1

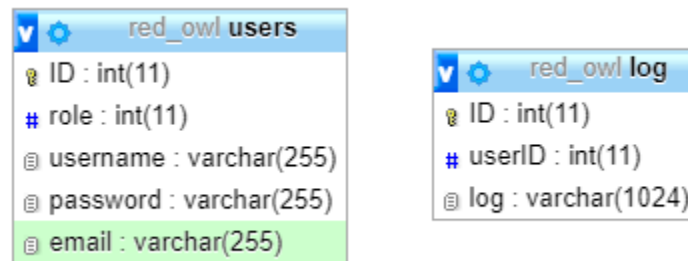


Figure 3.3.1

The important step is designing the servlet structure. There will be two action in “index” page and “uav_index” page. All the drone features will be managed in single page. So, two page is enough for this project because uav_index page will include four tabs. They are; manual control tab, transportation tab, search tab and settings tab. The index uml diagram is described in figure 3.3.2 and activity diagram is described in 3.3.3

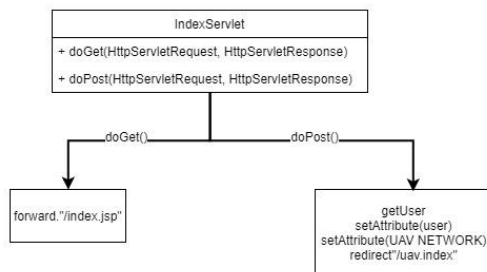


Figure 3.3.2

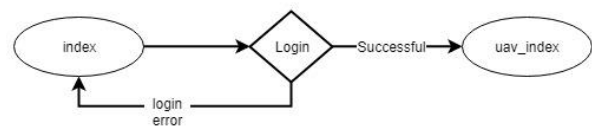


Figure 3.3.3

Uav index page create the socket threads to establish connection between telemetry and drone then redirect to “uav_index” page. User interactions are provided by the page. The servlet diagram is shown by figure 3.3.4 and page components are shown in figure 3.3.5 and implementation in 3.3.6.

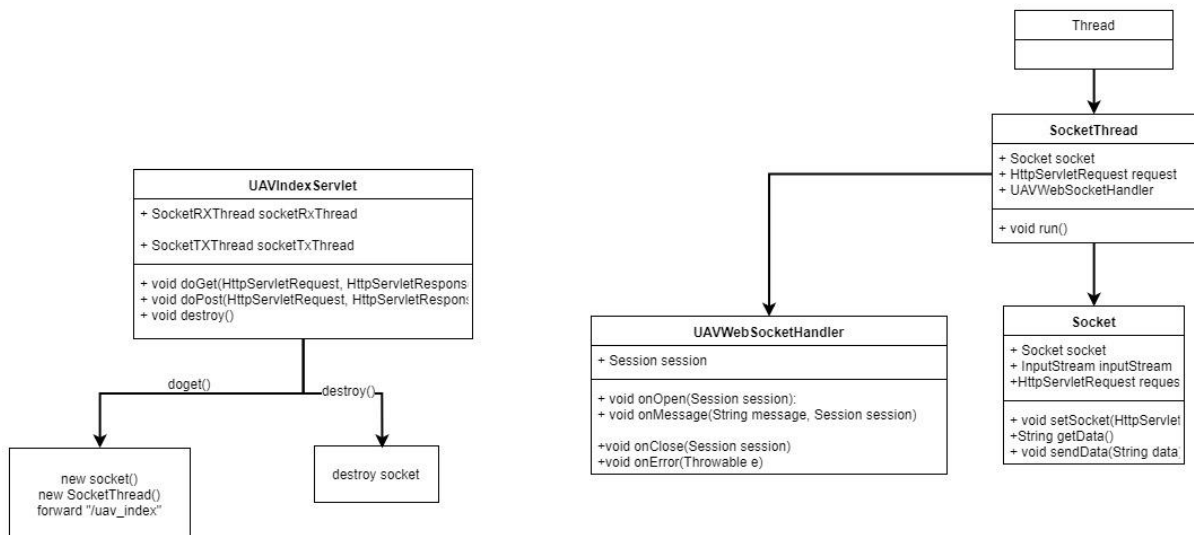


Figure 3.3.4

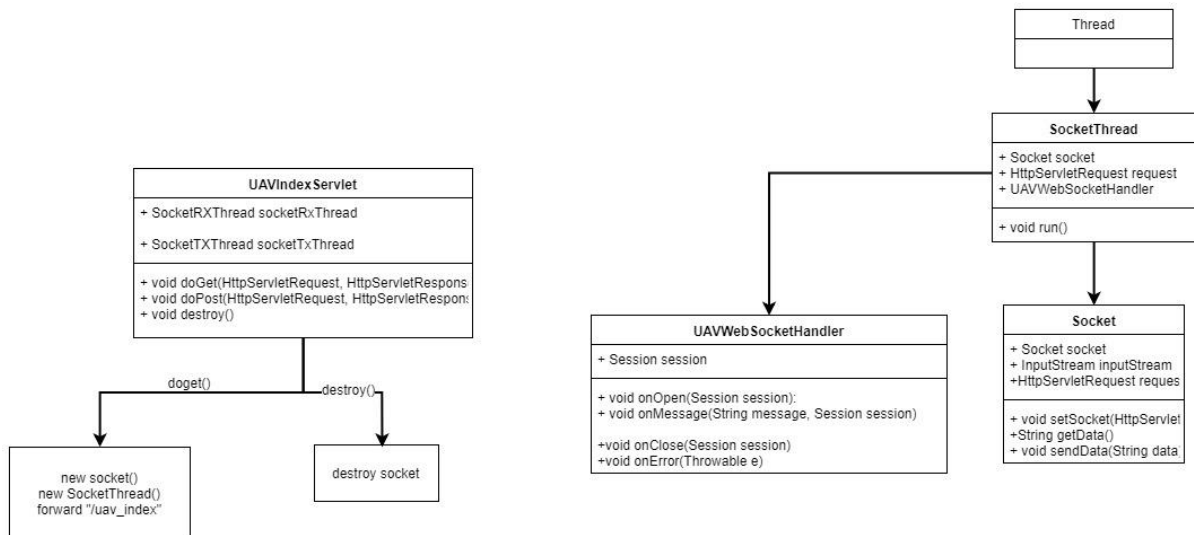


Figure 3.3.5

```

<!-- tab Panes -->
<div class="tab-content">
  <div role="tabpanel" class="tab-pane active" id="ManuelControl">
    <jsp:include page="${pageContext.request.contextPath}/resources/components/manuel_control.jsp"></jsp:include>
  </div>

  <div role="tabpanel" class="tab-pane" id="Transportation">
    <jsp:include page="${pageContext.request.contextPath}/resources/components/transportation.jsp"></jsp:include>
  </div>

  <div role="tabpanel" class="tab-pane " id="Search">
    <jsp:include page="${pageContext.request.contextPath}/resources/components/uav_search.jsp"></jsp:include>
  </div>

  <div role="tabpanel" class="tab-pane " id="Settings">
    <jsp:include page="${pageContext.request.contextPath}/resources/components/settings.jsp"></jsp:include>
  </div>
</div>

```

Figure 3.3.6

3.4 OpenCV

This section is focused on the OpenCV side of drone. As mentioned before OpenCV is an open source library which is used in computer vision, it is used for a wide variety of operations but will be used for the following operations: image manipulation , image processing and facial recognition.

In order to create facial detection software using the OpenCV the needed libraries must first be installed which are the PIL library, the cv2 Library and the numpy Library. After installing the necessary libraries to be used the task at hand is analyzed. In order to create a facial recognition software with identification capabilities then the task can be broken down into smaller tasks. The first is recognizing faces, the second is identifying the faces the camera finds. In order to accomplish the second task we need to break it down further into smaller tasks, the first being taking multiple images of the person in order to identify them, the second is a trainer that will train the drone to identify the people it sees through the camera. So in total we have three files to be made.

3.4.1 DataSetCreator.py

This first file handles the creation of the database of images from which the drone is going to train and recognize faces and identifies them. This python file uses the OpenCV library to access the camera and take pictures of the requested person. Below is the code of the file in figure 3.4.1.1:

```

import cv2
cam = cv2.VideoCapture(0)
detector=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
i=0
offset=50
name=input('enter your id: ')
while True:
    ret, im =cam.read()
    gray=cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
    faces=detector.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=5, minSize=(100, 100), flags=cv2.CASCADE_SCALE_IMAGE)
    for(x,y,w,h) in faces:
        i=i+1
        cv2.imwrite("dataSet/face-"+name+'.'+ str(i) + ".jpg", gray[y-offset:y+h+offset,x-offset:x+w+offset])
        cv2.rectangle(im,(x-50,y-50),(x+w+50,y+h+50),(225,0,0),2)
        cv2.imshow('im',im[y-offset:y+h+offset,x-offset:x+w+offset])
        cv2.waitKey(100)
    if i>30:
        cam.release()
        cv2.destroyAllWindows()
        break

```

Figure 3.4.1.1

In order for the software to recognize faces, a cascade must be set. In this case the haarcascade is used because the haarcascade is known for their calculation speed, because it uses integral images, therefore being able to calculate any haar feature quickly. This python file takes 30 images through accessing the camera with the OpenCV library, the camera takes pictures of only the face of a person to avoid processing unnecessary data.

3.4.2 DataSetTrainer.py

This file handles training the drone so that it may identify people through the camera of the drone. It uses the images and scans the faces and matches the faces to the IDs and prints the IDs when it recognizes the face. The code is shown in figure 3.4.2.1

The algorithm which reads and the images and identifies the faces is the LBPH algorithm , standing for Local Binary Pattern Histogram. This algorithm labels the pixels of the images by thresholding the neighboring pixel and then considering the result as a binary number. This algorithm is composed of 4 parameters:

- Radius: the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.
- Neighbors: the number of sample points to build the circular local binary pattern. Keep in mind: the more sample points you include, the higher the computational cost. It is usually set to 8.
- Grid X: the number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.
- Grid Y: the number of cells in the vertical direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

After the algorithm analyzes the faces and matches them to their respective IDs, a database is produced containing all the necessary information for identification.

```

recognizer = cv2.face.LBPHFaceRecognizer_create()
cascadePath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath);
path = 'dataSet'

def get_images_and_labels(path):
    image_paths = [os.path.join(path, f) for f in os.listdir(path)]
    # images will contains face images
    images = []
    # [label] will contains the label that is assigned to the image
    [label] = []
    for image_path in image_paths:
        # Read the image and convert to grayscale
        image_pil = Image.open(image_path).convert('L')
        # Convert the image format into numpy array
        image = np.array(image_pil, 'uint8')
        # Get the label of the image
        nbr = int(os.path.split(image_path)[1].split(".")[0].replace("face-", ""))
        #nbr=int(''.join(str(ord(c)) for c in nbr))
        print(nbr)
        # Detect the face in the image
        faces = faceCascade.detectMultiScale(image)
        # If face is detected, append the face to images and the label to [label]
        for (x, y, w, h) in faces:
            images.append(image[y: y + h, x: x + w])
            [label].append(nbr)
            cv2.imshow("Adding faces to traning set...", image[y: y + h, x: x + w])
            cv2.waitKey(10)
    # return the images list and [label] list
    return images, [label]

images, [label] = get_images_and_labels(path)
cv2.imshow('test', images[0])
cv2.waitKey(1)

recognizer.train(images, np.array([label]))
recognizer.save('trainer/trainer.yml')
cv2.destroyAllWindows()

```

Figure 3.4.2.1

3.4.3 FaceDetection.py

This file (figure 3.4.3.1) is in charge of recognizing the faces of humans and identifying them. It draws a rectangle on the faces of humans and writes the names of the people on the bottom side of the rectangle in white text.

```

cam = cv2.VideoCapture(0)
font = cv2.FONT_HERSHEY_COMPLEX
while True:
    ret, im = cam.read()
    gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=5, minSize=(100, 100), flags=cv2.CASCADE_SCALE_IMAGE)
    for (x,y,w,h) in faces:
        nbr_predicted, conf = recognizer.predict(gray[y:y+h,x:x+w])
        cv2.rectangle(im, (x-50,y-50), (x+w+50,y+h+50), (255,0,0), 2)
        if (nbr_predicted==1):
            print(nbr_predicted)
            nbr_predicted='Ibo'
        elif (nbr_predicted==2):
            nbr_predicted='Abdo'
        #Draw the text
        cv2.putText(im, str(nbr_predicted), (x,y+h), font, 2, (255,255,255), 2, cv2.LINE_AA)
        cv2.imshow('im', im)
        cv2.waitKey(10)

```

Figure 3.4.3.1

It begins by accessing the camera through the use of the OpenCV library , when the camera finds a human face , a rectangle will be drawn around their face, after that the image is run by the database that was generated and compares the captured images to the database images and according to that it prints the corresponding user ID, from there it prints the user corresponding user name as the in figure 3.4.3.2.

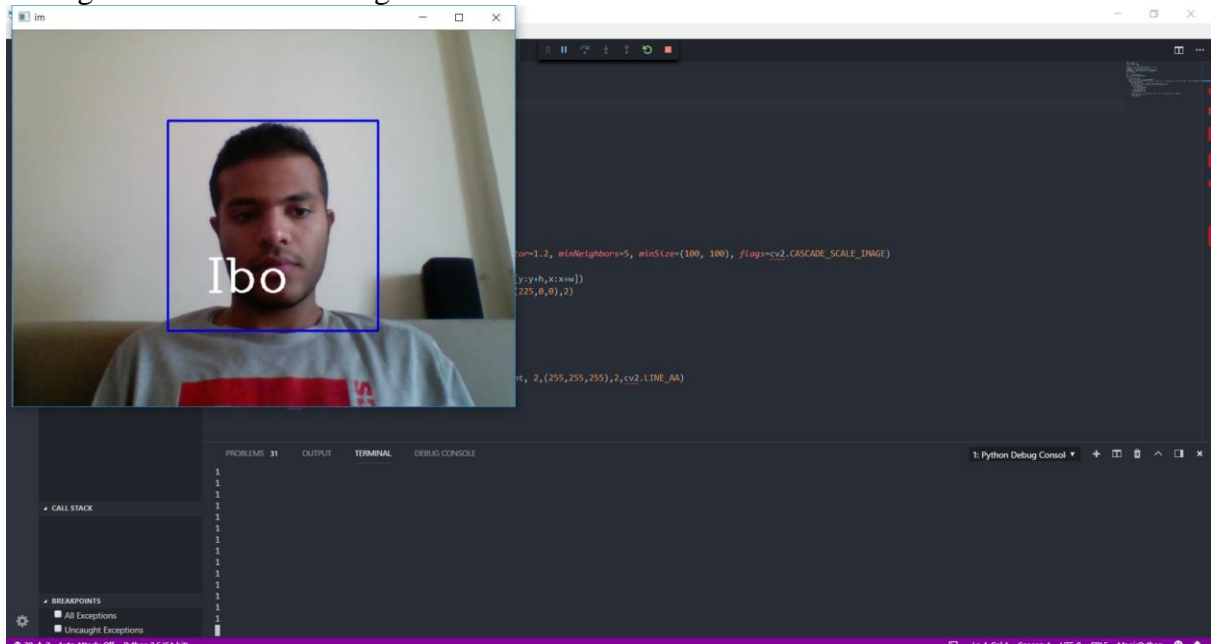


Figure 3.4.3.2

4. Components

In this section, all the components will be explained in more detail including programming implementation.

4.1 Drone Components

The circuit design is explained in section 2.2. In this section, brushless motor, electronic speed controller, gyroscope, gps, raspberry pi, raspi-camera and hcsr-04 will be explained.

4.1.1 Raspberry Pi



Figure 4.1.1.1

Raspberry Pi is a popular microcomputer (Figure 4.1.1.1). because of the high performance and number of input/output pin. Raspberry Pi B+ has Broadcom BCM2837B0 quad-core A53 (ARMv8) 64-bit @ 1.4GHz with 1 GB LPDDR2 SDRAM and 40-pin GPIO header.

Raspberry Pi is supported by cross platform operating systems as windows 10 IoT, ubuntu, noobs operating system, OSMC and etc..

In this project, Linux based, Raspbian operating system will be used.

4.1.2 Raspi-Camera

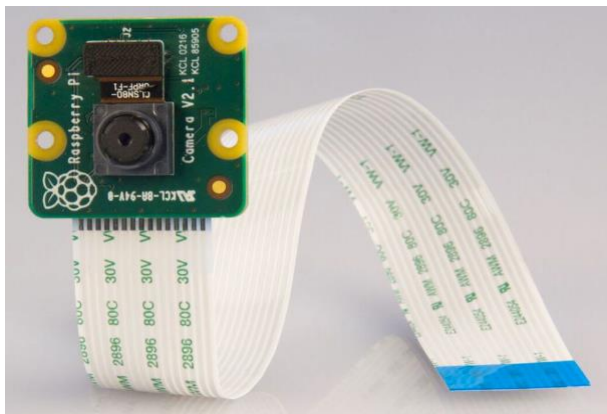


Figure 4.1.2.1

Raspi-camera (Figure 4.1.2.1) is very suitable camera for raspberry pi because raspberry pi has a special CSI connector to connect the raspi-camera. So, there is no need to connect a heavy usb camera at the raspberry pi. Raspi-camera is just 25mm x 23mm and it has 1080p video quality with 3280 x 2464 resolution. When the camera is connected to raspberry Pi, camera interface must be enabled config menu. Config menu would be opened by using “raspi-config” terminal command.

4.1.3 Brushless Motor



Brushless dc motor (Figure 4.1.3.1) is powerful and faster motor compared to brushed motors.

Two key performance parameters of brushless DC motors are the motor constants K_t (torque constant) and K_e (back-EMF constant also known as speed constant $KV = 1/K_e$ explained the 4.1.3.1 and 3.1.3.2 equations).

Figure 3.1.3.1

$$K_t = \frac{\text{NewtonMeter}}{\text{Amp}} = \frac{\text{KilogramMeter}^2}{\text{AmpereSecond}^2} \quad (4.1.3.1)$$

$$K_e = \frac{\text{VoltSecond}}{\text{Radian}} = \frac{\text{KilogramMeter}^2}{\text{AmpereSecond}^2} \quad (4.1.3.2)$$

In this project, 1400 kV brushless motors are used. Brushless motors, must be configured before to fly. The procedure is described as below;

- Disconnect battery
- Set maximum speed to brushless motor
- Wait in a second
- Connect the battery
- Wait in two second
- Set minimum speed for two second
- Set the pulse width 0 for two second
- Then run the motors in minimum speed

The speed can be controlled by pulse width. This method can be implemented as in figure 4.1.3.2;

```
pi.set_servo_pulsewidth(self.forwardRight, self.minThrottle)
pi.set_servo_pulsewidth(self.forwardLeft, self.minThrottle)
pi.set_servo_pulsewidth(self.backleft, self.minThrottle)
pi.set_servo_pulsewidth(self.backRight, self.minThrottle)
```

Figure 4.1.3.2

The motor throttle model controls the drone altitude and pid control or rotations. The model is described in equation 3.1.3.1.

$$Throttle = \begin{bmatrix} minSpeed + throttleOffset * \alpha_T + angleOffset * \alpha_A \\ minSpeed + throttleOffset * \alpha_T + angleOffset * \alpha_A \\ minSpeed + throttleOffset * \alpha_T + angleOffset * \alpha_A \\ minSpeed + throttleOffset * \alpha_T + angleOffset * \alpha_A \end{bmatrix} \quad (4.1.3.1)$$

The “minSpeed” means that drone minimum speed, throttle offset is altitude pid control value times alpha value and angle offset is pid control or command and alpha constant. Pid control will be explained in section 4.1.5. The commands are would be as described below;

throttle : stable, ascending, descending

yaw : stable, rotate left, rotate right

pitch : stable, move forward, move backward

roll: stable, move left, move right

The axis explanation is in figure 4.1.5.2 and drone motion control explanation is in figure 4.1.5.3.

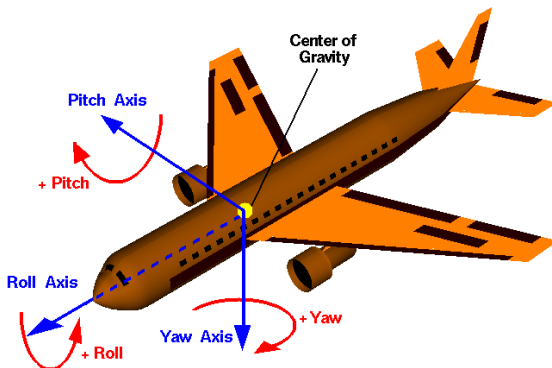


Figure 4.1.5.2

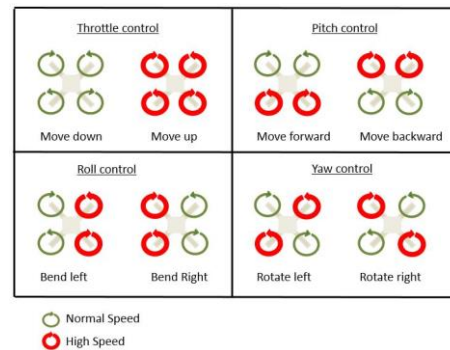


Figure 4.1.5.3

Finally, code implementation is shown in figure 3.1.5.4 for just one motor. The motor pulse width must not less than min speed or must not exceed max speed when the pid is calculated.

```
'''
throttle offset is minSpeed + altitudeSpeed
and motors can be controlled by offsets
'''
def runMotors(self):
    if(self.throttleOffset + self.forwardRightOffset < self.minSpeed):
        self.pi.set_servo_pulsewidth(self.forwardRight, self.minSpeed)
    elif(self.throttleOffset + self.forwardRightOffset > self.maxSpeed):
        self.pi.set_servo_pulsewidth(self.forwardRight, self.maxSpeed)
    else:
        self.pi.set_servo_pulsewidth(self.forwardRight, self.throttleOffset + self.forwardRightOffset)
```

Figure 4.1.5.4

4.1.4 Electronic Speed Controller



Figure 4.1.4.1

In generally, any electronic speed controller can handle one brushless motor (figure .1.4.1). But in this project, Skywalker 25A quattro Electronic Speed Controller (figure 4.1.4.2) will be used. This electronic speed controller can control four brushless motor, so it has less weight and cheaper than four different electronic speed controllers.

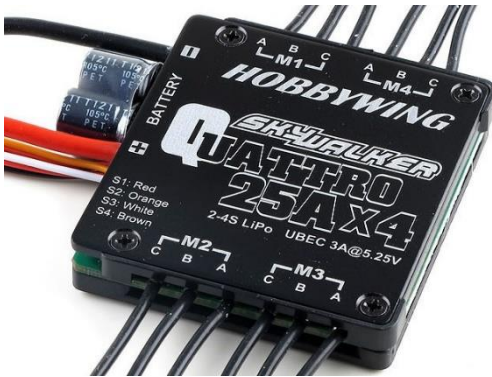


Figure 4.1.4.2

Also, the esc (electronic speed controller) can provide 3A and 5.25 V by using ubec system. This power is enough to powered raspberry pi without using external power source.

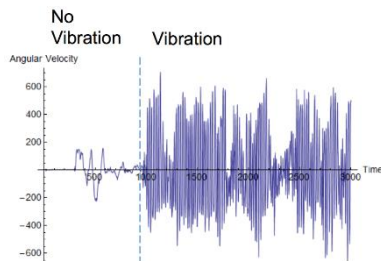
4.1.5 IMU

Inertial Measurement Units (IMUs) is a self-contained system that measures linear and angular motion usually with a triad of gyroscopes and triad of accelerometers. In this project, MPU-6050 (figure 4.1.5.1) will be used because the imu sensor is very cheap but it can very sensible for noise and vibrations.

In fact, each imu sensor is affected by vibrations and it is a big problem for pid control. Because pid control measure the -x and -y rotations. If the sensor values will be wrong, then drone cannot be stable position.



Figure 4.1.5.1



Vibration affect on mpu60-50. Chart is quoted from: [stuka/sensor/fusion](https://stuka.sensors.fusion)

So, the results must be filtered. Low-pass and Kalman filter will be used. The angle equations are shown as equation 4.1.5.1 and 4.1.5.2 and the implementation of low pass filter is as the equation 4.1.5.3 and Kalman filter algorithm is explained in equations between 4.1.5.4 to 4.1.5.6.

$$ac_x = \frac{A_y}{\sqrt{A_x^2 + A_z^2}} \quad (4.1.5.1)$$

$$ac_y = \frac{A_x}{\sqrt{A_y^2 + A_z^2}} \quad (4.1.5.2)$$

Where;

A = {xi} accelerator value set

Ac = {xi} accelerator angle set

Gy = {xi} gyroscope value set

$$angle = \begin{bmatrix} \alpha * (gy_x * dt + gy_{x-1}) + (1 - \alpha) * ac_x \\ \alpha * (gy_y * dt + gy_{y-1}) + (1 - \alpha) * ac_y \end{bmatrix} \quad (4.1.5.3)$$

$$K_i = \left[\frac{\frac{P_{i-1}}{P_{i-1} + noise}}{\frac{P_{i-1}}{P_{i-1} + noise}} \right] \quad (4.1.5.4)$$

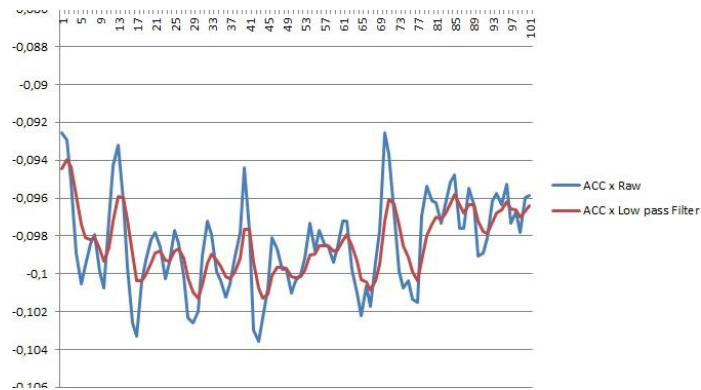
$$P_i = \begin{bmatrix} (1 - K_i) * P_{i-1} \\ (1 - K_i) * P_{i-1} \end{bmatrix} \quad (4.1.5.5)$$

$$angle_i = \begin{bmatrix} angle_{i-1} + K_i * (z_i - angle_{i-1}) \\ angle_{i-1} + K_i * (z_i - angle_{i-1}) \end{bmatrix} \quad (4.1.5.6)$$

Where K_k is the Kalman gain, z_i is the measured value and p_i is the prior error covariance.



Kalman Filter chart from marketcalls/amibroker



Low-pass filter from solerotech1

The result is used for pid control in motor thread. The pid control measure the error than correct the position. Drone estimate that -x angle and -y angle must be 0. If the values are different than 0, then it gives us an error value by using time integral and derivative time. Pid control structure is shown as figure 4.1.5.2 and equation 4.1.5.7.

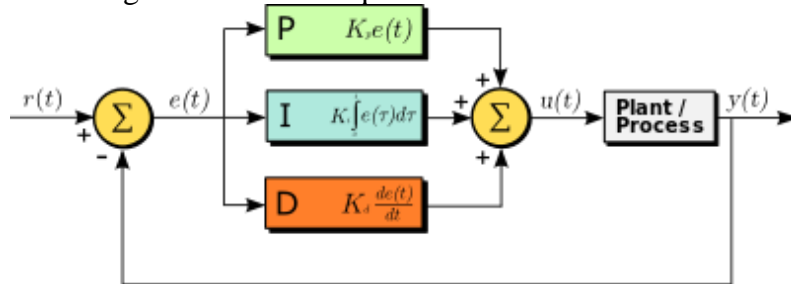


Figure 4.1.5.2

$$u(t) = K_p e(t) + K_i \int_0^t e(t) dt + K_d \frac{de(t)}{dt} \quad (4.1.5.7)$$

The code implementation is described as figure 4.1.5.3

```
def xPidControl(self):
    pidLimit = 150
    """
    expected is 0
    """
    self.resetMovementOffsets()
    error = self.xRotation
    pid = (error * self.rotationPidP) + ((self.previousYError + error) * self.rotationPidI) + (self.previousXError * self.rotationPidD)
    print("x rotation is: ", self.xRotation, "x pid error is: ", pid)
    if(pid > pidLimit):
        pid = pidLimit
    if(pid > 0):
        self.backRightOffset += pid
        self.backLeftOffset += pid

    elif(pid < 0):
        pid = -1 * pid
        self.forwardRightOffset += pid
        self.forwardLeftOffset += pid
    self.previousXError = error
```

Figure 4.1.5.3

4.1.6 GPS



Figure 4.1.6.1

Gps implementation is very easy because the sensor use just rx and tx pins. So, python can read the values using “serial” library. In this project, GY-NEO6MV2 GPS module is used as shown in figure 4.1.6.1 and the code implementation is shown in figure 4.1.6.2.

```
import serial
import time

port = '/dev/ttyAMA0'

ser = serial.Serial(port,baudrate = 9600,timeout = 2)

while True:
    try:
        data = ser.readline()
        print("data is: ", data.decode('UTF-8'))
    except:
        print("loading")
    time.sleep(2)
```

Figure 4.1.6.2

4.1.7 HCSR-04

The gps module has ± 5 -meter error range, and the next version will use barometric sensor but this is also ± 17 cm error range. But distance sensor can provide altitude ± 0.02 cm error range. But it can measure 2-400 centimeter (depend on the sensor quality for hcsr-04 as shown in figure 3.1.7.1). So, the distance sensor is using for measure that how drone get altitude after a specific throttle point. For example throttle needs to 1000 pwm pulse to lift the drone. And 1022 pwm pulse increase the altitude. So when drone wants to get altitude, increase the pwm for 22.



Figure 4.1.7.1

But using of the sensor is optional. The drone will fly at outside and the altitude can be calculated by using mpu-6050 and gps. If the values are mixed, they can provide us small error range altitude.

The code implementation is shown in figure 4.1.7.2;

```
class HCSR_04():

    ECHO = 19
    TRIG = 26

    def __init__(self):
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(self.TRIG,GPIO.OUT)
        GPIO.setup(self.ECHO,GPIO.IN)

    def measureHCSR_04(self):
        GPIO.output(self.TRIG, False)
        time.sleep(0.1)
        GPIO.output(self.TRIG, True)
        time.sleep(0.00001)
        GPIO.output(self.TRIG, False)

        while GPIO.input(self.ECHO)==0:
            pulse_start = time.time()

        while GPIO.input(self.ECHO)==1:
            pulse_end = time.time()

        pulse_duration = pulse_end - pulse_start

        distance = pulse_duration * 17150
        distance = round(distance, 2)
        distance = distance - 0.5

        if distance > 2 and distance < 400:
            return distance
        else:
            return -1
```

Figure 4.1.7.3

4.1.8 Li-po Module

Li-po battery can provide high voltage and high ampere for any electronic device (figure 4.1.8.1). the “mah” meaning the li-po battery can provide x mah in a hour and c means the average current. So, the mah describe the capacity of battery directly. The flight time can be calculated by the equation 4.1.8.1.



Figure 4.1.8.1

$$Flight\ Time = \frac{mah}{1000} * \frac{1}{C} * 60 \quad (4.1.8.1)$$

4.1.9 Relay Module

Relay module (figure 3.1.9.1) controls the li-po battery using gpio(General Purpose Input/Output pins) pins and gpio library. To disconnect or connect the li-po battery, relay module is used because if li-po battery is connected to Raspberry Pi directly, Raspberry Pi will be broken. The code implementation is shown in figure 4.1.9.2.



Figure 4.1.9.1

```
import RPi.GPIO as GPIO
lipoRelayControl = 20
GPIO.output(self.lipoRelayControl, GPIO.HIGH)
print("disconnected")
GPIO.output(self.lipoRelayControl, GPIO.LOW)
print("connected")
```

Figure 4.1.9.2

The connect and disconnect status depend on the physical connection. Not depend on the programming status directly.

4.2 Telemetry Components

The pages and tab structure is described in 3.3. This section explains that each telemetry component in more detail. The telemetry components are classified as java side, JavaScript side and html side.

4.2.1 Backend Package Structure

The package structure is shown as figure 4.2.1.1.

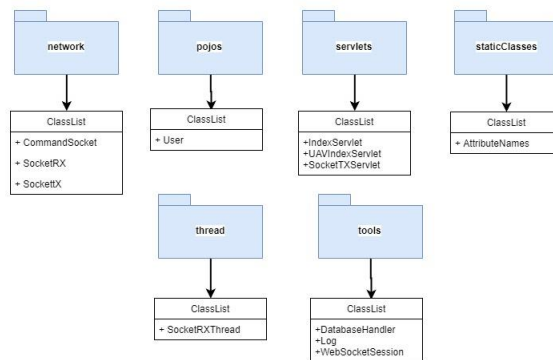


Figure 4.2.1.1

The “WebSocketSession” keeps the web socket session list for web socket session. Not the servlet session. And attribute names keeps the servlet session attribute names in static String as figure 4.2.1.2. All the other classes will be explained.

```
public class AttributeNames {
    public static final String USER = "user";
    public static final String USER_NAME = "username";
    public static final String PASSWORD = "password";
    public static final String DATABASE_HANDLER = "databaseHandler";
    public static final String WARNING_MESSAGE = "warning_message";
    public static final String UAV_HOST_ADDRESS = "uav_host_address";
    public static final String UAV_HOST_RX_PORT_NUMBER = "uav_host_rx_port_number";
    public static final String UAV_HOST_TX_PORT_NUMBER = "uav_host_tx_port_number";
    public static final String UAV_SOCKET_HANDLER = "uav_socket_handler";
    public static final String UAV_SOCKET_TX = "uav_socket_tx";
}
```

Figure 4.2.1.2

3.2.2 User Class

The user class uml diagram is in figure 4.2.2.1. in the index page, user try to login the system. If the login is successful, then the servlet session is set attribute by user to access the user information from anywhere.

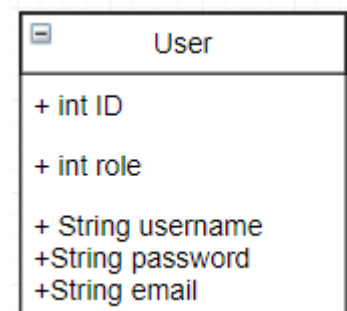


Figure 4.2.2.1

4.2.3 Tools

The tools package includes “DatabaseHandler” , “Log” and “WebSocketSession”. By using database handler program can establish communication between java and MySQL from any class. The log is a static class (figure 4.2.3.1) so, any class can create a log to track the error or track the drone. The log class use the database handler class. And finally, The “WebSocketSession” keeps the web socket session list for web socket session. Not the servlet session (figure 4.2.3.2)

```

/**
 * The Log system helps us to track our system
 */
public class Log {
    /**
     *
     * @param request to access the session to get user.
     * @param log will be recorded the database
     */
    public static void insertLog(HttpServletRequest request, String log) {
        System.out.println(log);
        ((DatabaseHandler) request.getSession().getAttribute(AttributeNames.DATABASE_HANDLER)).
            insertLog((User) request.getSession().getAttribute(AttributeNames.USER), log);
    }
}

```

Figure 4.2.3.1

```

public class WebSocketSession {

    /**
     * TomCat Server cannot provide us session.
     * if multiple users will use this system then convert it to arraylist or another
     * implementation.
     * If just one session is here, then just last user can get the system messages.
     */
    public static ArrayList<Session> sessions = new ArrayList<>();
}

```

Figure 4.2.3.2

4.2.4 Network

The network package include the socket classes as CommandSocket, SocketRX (Figure 4.2.4.1), SocketTX (Figure 4.2.4.2) and UAVWebSocketHandler(Figure 4.2.4.3).

```

public void setSocketRX(HttpServletRequest request) {
    this.request = request;
    System.out.println("host is: " + (String) request.getSession().getAttribute(AttributeNames.UAV_HOST_ADDRESS));
    System.out.println("number is: " + (int) request.getSession().getAttribute(AttributeNames.UAV_HOST_RX_PORT_NUMBER));
    try {
        socket = new Socket((String) request.getSession().getAttribute(AttributeNames.UAV_HOST_ADDRESS),
            (int) request.getSession().getAttribute(AttributeNames.UAV_HOST_RX_PORT_NUMBER));
        inputStream = socket.getInputStream();
    } catch (IOException e) {
        e.printStackTrace();
        Log.insertLog(request, log: getClass().toString() + " SocketRX create ERROR");
    }
}

public String getData() {
    String message = "";
    while (character != -1) {
        try {
            character = inputStream.read();
        } catch (IOException e) {
            System.out.println("SocketRX is dropped while reading something");
            return null;
        } catch (Exception e) {
            System.out.println(getClass().getName().toString() + "has unknown error");
            e.printStackTrace();
            return null;
        } catch (Throwable e) {
            System.out.println(getClass().getName().toString() + "SocketRX is dropped while reading something");
            return null;
        }
        message += (char) character;
        if (character == '\n') { //SPLIT JSON
            message = message.substring(message.indexOf('('), message.length());
            //Log.insertLog(request, message); //close for NOW!!
            return message;
        }
    }
    return null;
}

```

Figure 4.2.4.1

Get Data method gets a json value. In fact, the data come as byte-by-byte but if the method catch the '}' character, then method understand that a json object is complete.

```
public class SocketTX {
    Socket socket = null;

    OutputStream outputStream = null;
    HttpServletRequest request = null;
    public SocketTX() {
    }

    public void setSocketTX(HttpServletRequest request) {...}

    public void sendData(String message) {
        byte buffer[] = message.getBytes();
        try {
            outputStream.write(buffer);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

To send command at drone byte-by-byte. As the RX, the data send as json object and drone can understand when the json object is complete.

Figure 4.2.4.2

```
@ServerEndpoint("/UAVWebSocketEndPoint")
public class UAVWebSocketHandler {

    /**
     * in the on open method, system set the static session as the
     * coming session to send some messages to jsp page continuously.
     * @param session is setted as static session to use all system.
     */
    @OnOpen
    public void onOpen(Session session) {...}

    @OnMessage
    public void onMessage(String message, Session session) {...}

    @OnClose
    public void onClose(Session session) { System.out.println("Session " + session.getId() + " has ended"); }

    @OnError
    public void onError(Throwable e) { e.printStackTrace(); }
}
```

Figure 4.2.4.3

When data is received in telemetry, then the data is send at the web browser by using web socket.

The servlet and static classes were explained.

4.2.5 Frontend components

To provide clean code; footer, header and navbar code are written as a structure (figure 4.2.5.1) and each jsp file include them to fill their own body.

The header includes the css files and title, navbar includes the navbar and footer includes all the jQuery and JavaScript files. Then, <jsp:include page="src"> tag implement the requested file.

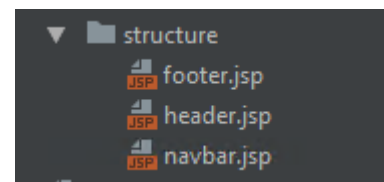


Figure 4.2.5.1

For example, the head and navbar files are implemented as described below in figure 4.2.5.2.

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<html>
<head>
    <jsp:include page="${pageContext.request.contextPath}/resources/structure/header.jsp"></jsp:include>
</head>
<body>
    <jsp:include page="${pageContext.request.contextPath}/resources/structure/navbar.jsp"></jsp:include>
```

Figure 4.2.5.2

The structure follows the css, error, assets and error packages in resources package as shown in figure 4.2.5.3. In generally, the project use bootstrap 4.0 online but for the settings page, a special css file is written. Also, for the gyroscope, three-dimensional display engine is written. This engine needs a three -dimensional drone drawing. 3 dimensional objects include millions of vertex points. These are the points are described as json file. With a texture, the drone would seen beautiful.

In assets, no camera found picture is displayed when the camera is not accessible because of the network trouble. Finally, warning message error is displayed when user cannot login the system because of the incorrect username or password.

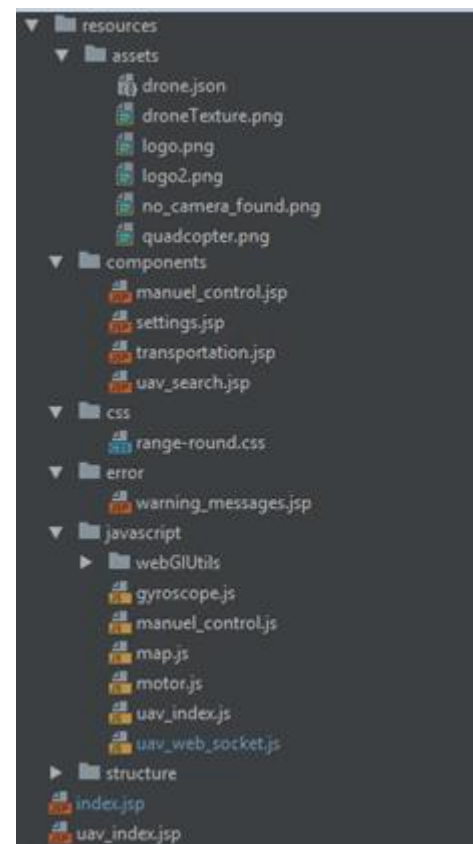


Figure 4.2.5.3

4.2.6 Google Maps

The gps location is tracked and recorded by the map system. The code structure is described in figure 4.2.6.1. The uavMap() function is called when the javascript source is added with google maps api key. Then uavMap call the map options then ihaPathAdder draws polylines as the figure 4.2.6.2 to track the drone.

```

function uavMap() {...}

function flighMapSetter() {...}

function ihaPathAdder(x,y) {
  console.log("Coordinate is: " , x , " " , y);
  var path = polyPath.getPath();
  path.push(new google.maps.LatLng(parseFloat(x),parseFloat(y)));
  polyPath.setPath(path);

  var transportationPath = transportationMapPath.getPath();
  transportationPath.push(new google.maps.LatLng(parseFloat(x),parseFloat(y)));
  transportationMapPath.setPath(transportationPath);

  var searchPath = searchMapPath.getPath();
  searchPath.push(new google.maps.LatLng(parseFloat(x),parseFloat(y)));
  searchMapPath.setPath(searchPath)
}

```

Figure 4.2.6.1

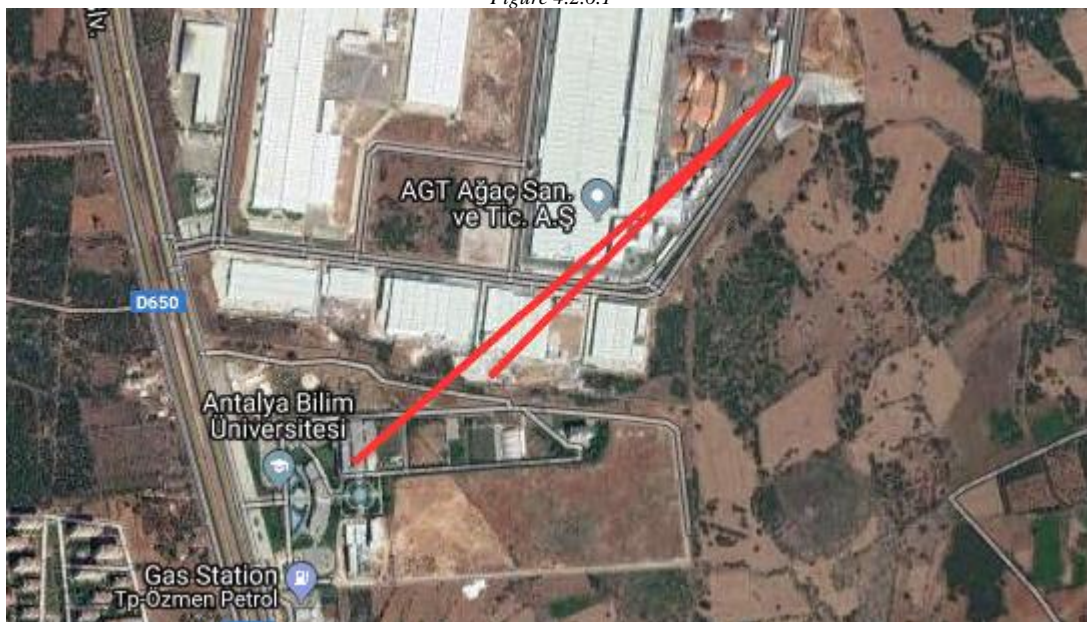


Figure 4.2.6.2

If the telemetry screen is on the transportation mode, then google maps listen for a selected point as figure 4.2.6.3. Then drone will go to selected point autonomously. Try to get a movement, then if the distance is decreased then continue to this way.



Figure 4.2.6.3

4.2.7 Gyroscope

The -x and -y angles are displayed as three-dimensional monitor by using JavaScript 3d canvas. The code structure is shown as figure 4.2.7.1.

```

25  var init = function () {...};
56
57  var run = function (vertexShaderText, fragmentShaderText, droneImage, droneModel) {...};
227
228  var x = 0;
229  var y = 0;
230
231  var rotateX = 0;
232  var rotateY = 0;
233
234  function loop() {...}
262
263  function degToRad (degrees) {...}
266
267  /*
268   the x and y values would be set by setX and setY functions
269  */
270  function setX(xValue) {
271    x = xValue;
272  }
273
274  function setY(yValue) {
275    y = yValue;
276  }
277

```

Figure 4.2.7.1

The init function create the drone object using the drone vertex points from assets folder and paint by using drone texture file. The run function create the three dimensional world for the drone and loop function read and redraw the drone position again and again by using x and y values. Example pictures are bellow in figure 4.2.7.2



Figure 4.2.7.2

4.2.8 Camera

Most important component is the drone camera. But in real life, if wi-fi signal is weak then there may be some package loss occurred or camera would be crashed in network. In this status, the telemetry screen displays a camera error picture as shown in figure 4.2.8.1. In the other hand, if there will be no error, then user can get high quality live camera as shown in figure 4.2.8.2.



No cameras found
on your network.

Figure 4.2.8.1

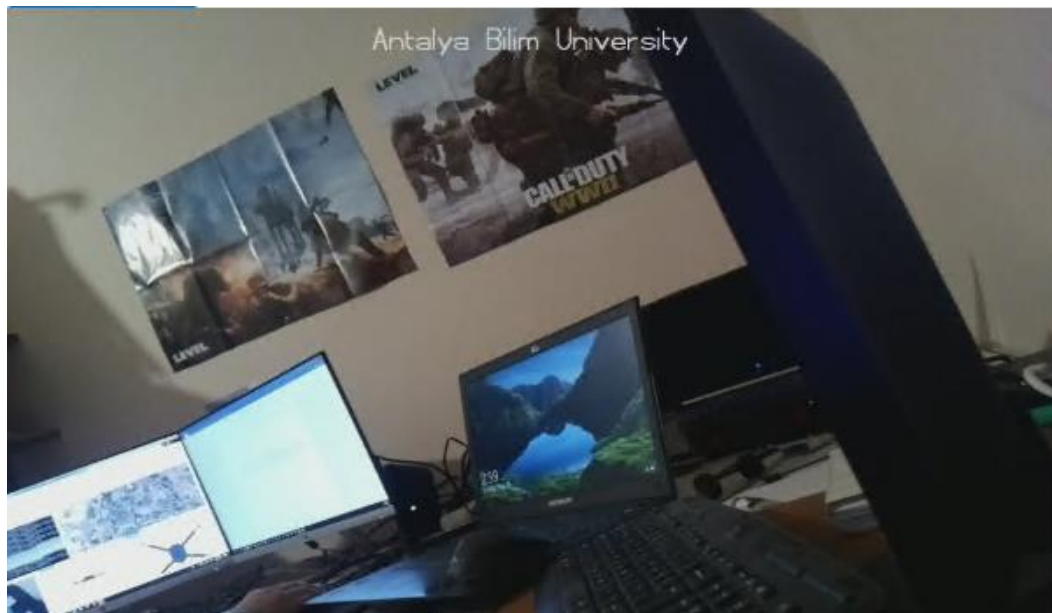


Figure 4.2.8.2

In the drone part, the camera is built as “mjpeg” file and an http header is built by using http and socketserver libraries. So, to access the drone; typing drone ip address with port 8000 is enough as described in figure 4.2.8.3.

```
<div class="col">
  <button onclick="cameraToggle()" class="btn btn-primary">Show/Hide</button>
  
</div>
```

Figure 4.2.8.3

4.2.9 Motor Offset

“Motor.js” file can display the motor status to user. If any motor run more than throttle offset, then the propellers are marked by red circle as the example of figure 4.2.9.1.

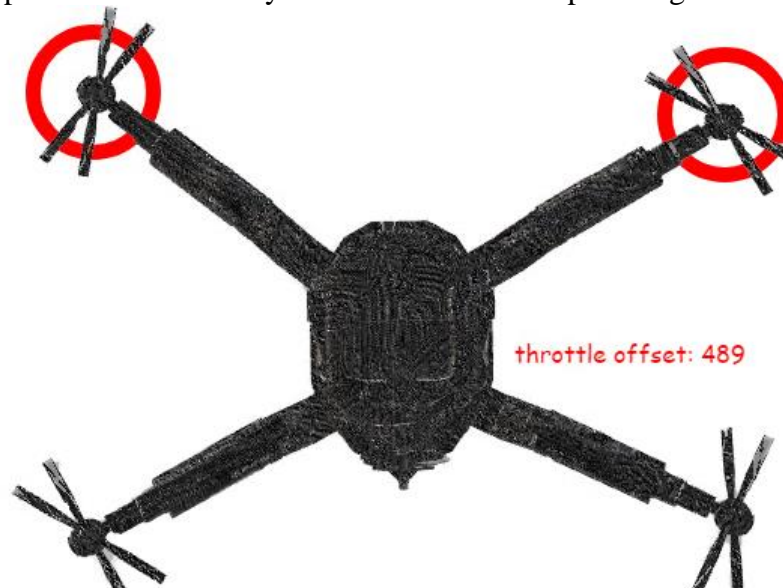


Figure 4.2.9.1

In this example capture, the user can understand that drone goes backward or on the forward direction, drone has a rotation error. So drone try to correct the error. The code structure is shown in figure 4.2.9.2.

```
var motorCanvas;
var motorCanvasContext;

window.onload = function() { ... };

function drawMotor() {
  base_image = new Image();
  base_image.src = "../resources/assets/quadcopter.png";
  motorCanvasContext.drawImage(base_image, 10, 10);
}

function writeThrottleOffset(throttleOffset) {
  motorCanvasContext.font = "16px Comic Sans MS";
  motorCanvasContext.fillStyle = "red";
  motorCanvasContext.textAlign = "center";
  var text = throttleOffset.toString();
  text = "throttle offset: " + text;
  motorCanvasContext.fillText(text, 375, 250);
}

function drawForwardLeft() {
  motorCanvasContext.beginPath();
  motorCanvasContext.arc(95, 70, 40, 0, 2*Math.PI);
  motorCanvasContext.stroke();

  drawMotor();
}
```

Figure 4.2.9.2

4.2.10 Web Socket JS

When drone send any Json data, the SocketRX catch the data then send to uav_web_socket.js on message function then the web socket split the data to send the recommended JavaScript component. The code is shown in figure 4.2.10.1 and the uml diagram is shown in figure 4.2.10.2

```
5 var webSocket = new WebSocket("ws://" + window.location.host + "/UAVWebSocketEndPoint");
6
7
8 $(document).ready(function() {...});
9
10 /*
11  Example of coming data:
12  {'mpu6050_x': 7, 'mpu6050_y': 41, 'gps_x': 39, 'gps_y': 41}
13 */
14
15
16 $(document).ready(function() {
17     writeThrottleOffset(throttleOffset);
18 });
19
20
21 $(document).ready(function () {
22     webSocket.onmessage = function (ev) {
23         var message = ev.data;
24         var messageJson = JSON.parse(message);
25         console.log("message json is: ", messageJson);
26         var xRotation = messageJson.xRotation;
27         var yRotation = messageJson.yRotation;
28         var gps_x = messageJson.gps_x;
29         var gps_y = messageJson.gps_y;
30         /*
31          * Set the x and y rotation
32          */
33         setX(xRotation);
34         setY(yRotation);
35         /**
36          * Set the ihaPath
37          */
38         ihaPathAdder(gps_x, gps_y);
39
40         var throttleOffset = messageJson.throttleOffset;
41         var forwardLeftOffset = messageJson.forwardLeftOffset;
42         var forwardRightOffset = messageJson.forwardRightOffset;
43         var backLeftOffset = messageJson.backLeftOffset;
44         var backRightOffset = messageJson.backRightOffset;
45
46         clearCanvas();
47         if(forwardRightOffset > throttleOffset){...}
48         if(forwardLeftOffset > throttleOffset){...}
49         if(backLeftOffset > throttleOffset){...}
50         if(backRightOffset > throttleOffset){...}
51
52         writeThrottleOffset(throttleOffset);
53
54         console.log("web socket message is: " + message);
55     };
56 });
57
58 //on close document must be already ready
59 webSocket.onclose = function (ev) {
60     console.log("web socket is closed");
61 };
62
63 webSocket.onerror = function (ev) {
64     console.log("web socket error!");
65 };
66
67 //ajax post method access the SocketTXServlet
68 function socketTX(message) {
```

figure 4.2.10.1

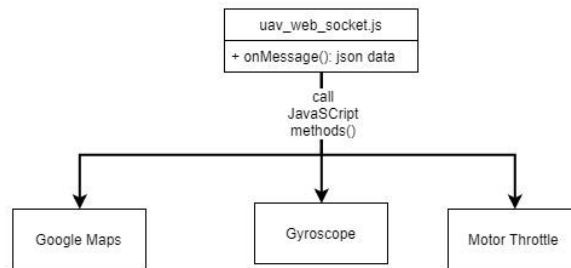


Figure 4.2.10.2

4.2.11 index.js

This javascript file control the toggle buttons. User may not want to see any component of telemetry screen. To open or close any component, the Show/Hide buttons would be used. For example, the motor throttle canvas is closed in figure 4.2.11.1

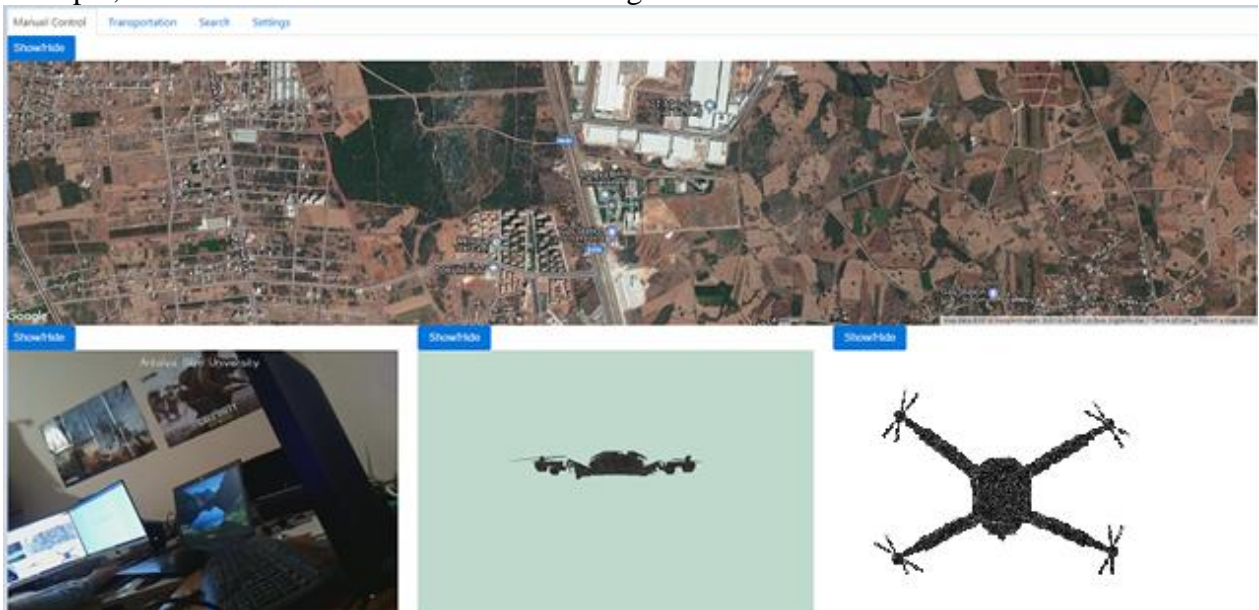


Figure 4.2.11.1

4.2.12 Search Page

This page sends “search” command to drone with requested perso. When the command is received by drone, the drone switch the flight mode as travel and search a person using OpenCV. If the person is found, then a message is send to telemetry screen then user can see the drone location from the maps or log files.

4.2.13 Settings Page

User may want to change the camera gamma, red, green or blue values. This page helps to this process. While user change the rgb values or gamma, then user can see the changes dynamically in the page as in the figure 4.2.13.1.

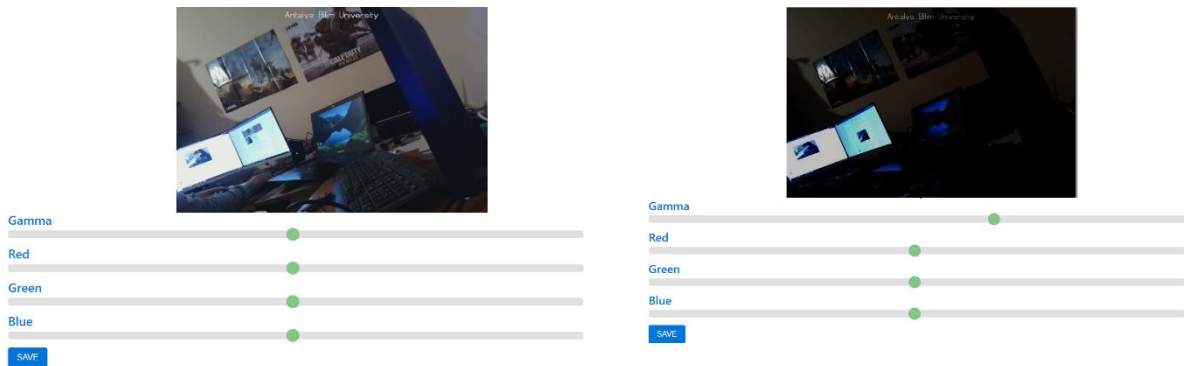
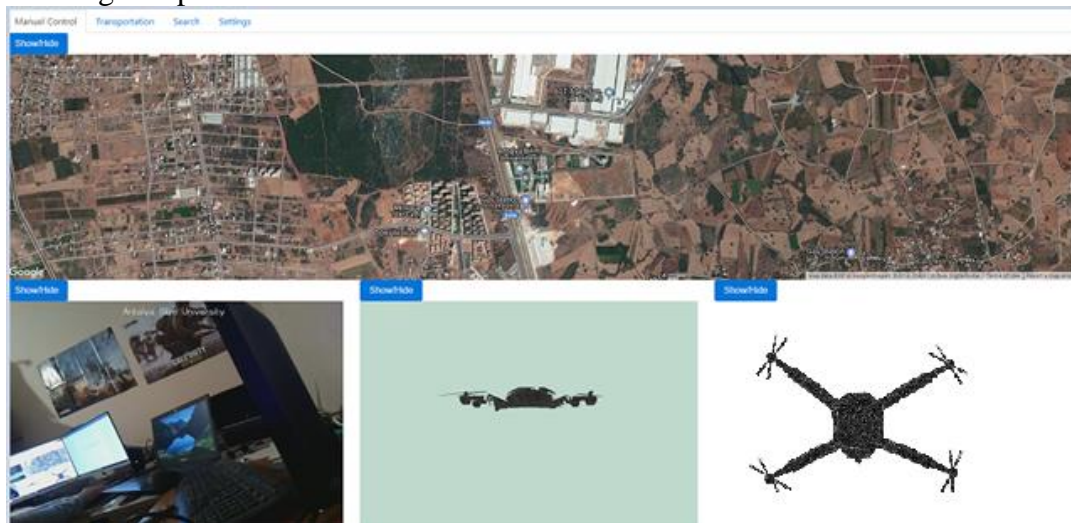


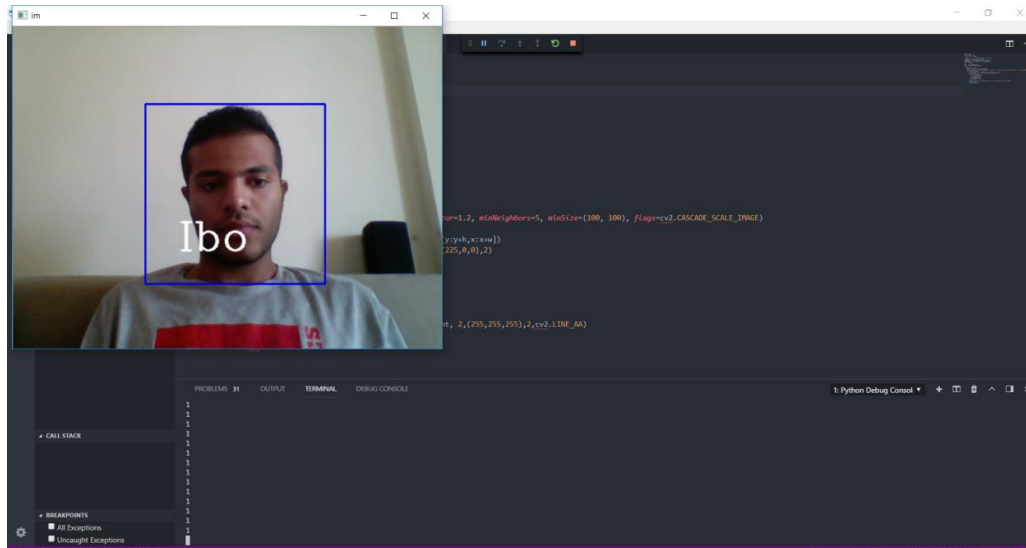
figure 4.2.13.1

5. Final

After months of studying the task at hand , gathering the needed parts, working on the telemetry system and the OpenCV code we finally have a working drone capable of transportation missions, scouting missions and searching and identifying people. In our the drone the user can set the drone to fly to set location autonomously for any of the tasks it was set for while providing complete ease for the user.



Telemetry monitor



OpenCV Demo

6. Constraints

The project is subject to multiple constraints:

- **Technological:** This depends on the user and how well versed they are in terms of basic computer operations , in order to use the drone to its fullest the user must have some computer knowledge.
- **Weather related:** The drone as an UAV may not be able to fly under extreme conditions and as such can only fly in suitable conditions and isn't suited for such as extremely windy weather.
- **Parts:** This constraint depends on the user , depending on how well the user is versed in electronics , they may or may not be able to replace any of the parts of the drone should they break or go defective which hampers the performance of the drone according to the piece.
- **Privacy:** Drones can be used for a multitude of purposes , one of them could be spying in which this kind of action violates the privacy of other people and will end up in stopping the drone from being used and the license being taken from the user.

7. Conclusion

In conclusion we managed to create a drone from scratch, that fulfills all the roles and tasks we intended it to do, capable of face recognition and identification, transportation missions and scouting tasks. We hope with this that the many and various uses of drones have been

highlighted and showcased with the rest being left to the user and their creativity on how to use the drone and further expand upon it or enhance it.

8. References

1. Aron, J and Scott, T, K. 2012., Measurement of Static And Dynamic Performance Characteristics Of Small Electric Propulsion Systems
2. Hugh L. Kennedy “Multidimensional Digital Smoothing Filters for Target Detection” Signal Processing, Volume 114, September 2015
3. Murach, J and Steelman A. Murach’s Java servlets and JSP 2ND Edition.
4. Bozic, S, M, Digital and Kalman Filtering, Edward Arnold, London 1979.
5. Maybeck, P. S. “The Kalman filter: An introduction to concepts.” Autonomous Robot Vehicles.
6. OpenCV documentation, taken from: <https://docs.opencv.org/>
7. Google Maps documentation, taken from: <https://developers.google.com/maps/documentation/>
8. Newton Meter, taken from: https://en.wikipedia.org/wiki/Newton_metre/
9. MPU6050 Accelerometer Gyroscope Interfacing with Raspberry Pi, taken from: <http://www.electronicwings.com/raspberry-pi/mpu6050-accelerometergyroscope-interfacing-with-raspberry-pi>
10. Raspberry Pi documentation, taken from: <https://www.raspberrypi.org/documentation/usage/gpio/>
11. Inertial Measurement Unit, taken from: <https://www.xsens.com/tags/imu/>
12. Getting started with Pi-Camera, taken from: <https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/>
13. Propeller Static Dynamic Thrust Calculation, July 13 2013, taken from: <https://www.flitetest.com/articles/propeller-static-dynamic-thrust-calculation>
14. PID Controller, taken from: https://en.wikipedia.org/wiki/PID_controller
15. jQuery-Ajax documentation: <https://api.jquery.com/>
16. Pi-camera documentation: <https://picamera.readthedocs.io/en/release-1.13/index.html>
17. Python documentation: <https://www.python.org/doc/>
18. Bootstrap 4 documentation: <https://getbootstrap.com/docs/4.1/getting-started/introduction/>
19. Java Oracle documentation, taken from <https://docs.oracle.com/javase/8/docs/technotes/guides/net/index.html>
20. JSON documentation: <https://www.json.org/>
21. JavaScript documentation: <https://documentation.js.org/>
22. Tab Content: https://www.w3schools.com/bootstrap/bootstrap_ref_js_tab.asp
23. Socket Programming: http://www.java2s.com/Tutorial/Java/0360_JSP/CreatingClientServerApplications.htm
24. How to Build a drone, taken from: <http://dronenodes.com/how-to-build-a-drone/>

25. GPS module, Ublox , taken from: <https://www.rlocman.ru/i/File/2011/04/22/1.pdf>
26. Relay Module, taken from:
[http://www.electfreaks.com/wiki/index.php?title=Relay_Module_\(Arduino_Compatible\)](http://www.electfreaks.com/wiki/index.php?title=Relay_Module_(Arduino_Compatible))
27. Brushless Motor, HobbyWing, taken from:
<http://www.hobbywing.com/products/enpdf/SkywalkerV2.pdf>
28. Li-Polymer Battery, MikroElektronika, taken from:
<https://download.mikroe.com/documents/datasheets/battery-datasheet.pdf>