



Encouragement Award

Programming, Apps & Robotics

Year 9-10

Caleb Tang

Prince Alfred College



Firefighter

My project is a fully wireless automated firefighting aircraft. The aerial vehicle uses a custom trained machine learning model to completely autonomously operate using Artificial Intelligence.








'Firefighter' utilises an onboard camera to stream the surroundings of the fire, detect fire and follow fire in real time as well as multiple sensors to alert the driver of what is going on around it and to track the conditions of the fire.



My project is designed to make firefighting safer. Over the last decade 65 people have been killed in bushfires, 33 of which were in the 2019-2020 bushfire season. It was reported by the Parliament of Australia that 9 of those 33 deaths from the fires were firemen. This could've easily been prevented using my project. By controlling, monitoring and putting out the fire from a safe distance away it means that there is almost no risk of firemen getting injured – a robot can be replaced but a person can never be brought back alive.



Not only are bushfires dangerous for the firefighters, but they are also extremely harmful to the environment. Approximately 400 million tons of carbon are released from fires in Australia every year, almost as much as Australia's annual human-caused emissions, and soot from bushfires can spread to places like New Zealand's glaciers, causing them to melt even quicker. Millions of animals need to travel over 20-30km to escape the heat and ferocity of the fires, and even if they do survive, they will need to find completely new homes and food sources. This could also lead to a major loss of biodiversity, especially in Australia as many species are endemic to Australia. The use of my project would increase the rate at which fires could be put out, ultimately decreasing the social, economic and environmental impact of the fires.






Because my firefighting drone has a rechargeable and easily replaceable battery it means that refilling the battery is significantly more convenient. Furthermore, the batteries can be charged onsite with solar power and thermoelectricity, two forms of energy that will be substantially abundant next to a fire, reducing its ecological footprint. The drone operates autonomously so that no driver is required, giving it the potential to work 24/7 with no rest, and it can be reproduced substantially quicker than training a group of completely new firemen, maximizing work time and efficiency. My project uses Machine Learning and Computer Vision to classify, detect and follow fire/smoke nearby, and its operator can control the water deployment system from anywhere in the world. I created a 'haar cascade' to recognise sources of fire and I used it with my python program to move the drone in conjunction with the fire. Onboard the drone I also programmed a NodeMCU with Arduino to monitor the conditions of the fire, e.g. temperature, and display it on a webpage with buttons to control the water deployment. I am currently also in the process of developing a mapping function for the operator to keep track of where exactly the drone is.

Although my project isn't fully refined yet and won't put a permanent stop to fires, it will to a large extent make firefighting safer and easier and it is a significant improvement from what is traditionally used to fight fires. Moving forwards, I hope to make this concept of autonomous, air-based firefighting more widespread and build a bigger, more realistic and more professional prototype. In the future I would also like to create a way to prevent fires from starting in the first place. This would be through automatically wetting high fire-risk zones, programming a robot to regularly mow the grass and rake the fields of leaves and tracking and reducing the amount of flammable fuels used in and around high-risk zones.

Components and Supplies	
Tello Drone	Drone base used 
NodeMCU	Used to display statistics webpage 
Jumper Wires (Assorted)	Used to connect parts together 
Breadboard	Used with sensors 
Temperature Sensor	Used to track the temperature of the fire 
LED's	Used to signal water being deployed 
Resistor	Used to reduce current flow 

Necessary Tools and Machines	
Laptop	Used to program components 
Soldering Iron	Used to solder parts together 

Cables (Assorted)	Used to connect/power boards and upload code 
Batteries	Used to power parts 

Apps and Online Services	
Arduino IDE	Used to create webpage 
Python	Language used to control drone 
OpenCV	Used to train haar cascade 
PyCharm	IDE used to write and run code 
Fritzing	Used to create Arduino schematics 

Code and Explanation

This portion of python code tells the drone which direction to move in order to detect and follow the bushfire

```
1 import cv2
2 import numpy as np
3 from djitellopy import tello
4 import time
5
6 me = tello.Tello()
7 me.connect()
8 print(me.get_battery())
9 me.streamon()
10 me.takeoff()
11
12 me.send_rc_control(0, 0, 25, 0)
13 time.sleep(2.2)
14
15 w, h = 360, 240
16 fbRange = [25000, 25500]
17 pid = [0.7, 0.7, 0]
18 pError = 0
19
20
21 def findFire(img):
22     fireCascade = cv2.CascadeClassifier("Resources/fireCascadeNew.xml")
23     imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
24     fires = fireCascade.detectMultiScale(imgGray, 1.2, 8)
25     myFireListC = []
26     myFireListArea = []
27
28     for (x, y, w, h) in fires:
29         cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 255), 2)
30         cx = x + w // 2
31         cy = y + h // 2
32         area = w * h
33         cv2.circle(img, (cx, cy), 5, (0, 255, 0), cv2.FILLED)
34
35         myFireListC.append([cx, cy])
36         myFireListArea.append(area)
37
38     if len(myFireListArea) != 0:
39         i = myFireListArea.index(max(myFireListArea))
40         return img, [myFireListC[i], myFireListArea[i]]
41     else:
42         return img, [[0, 0], 0]
43
44 def trackFire(info, w, pid, pError):
45     area = info[1]
46     x, y = info[0]
47     fb = 0
48     error = x - w // 2
49     speed = pid[0] * error + pid[1] * (error - pError)
50     speed = int(np.clip(speed, -100, 100))
51
52     if area > fbRange[0] and area < fbRange[1]:
53         fb = 0
54
55     elif area > fbRange[1]:
56         fb = -20
57
58     elif area < fbRange[0] and area != 0:
59         fb = 20
60
61     if x == 0:
62         speed = 0
63         error = 0
64     # print(speed, fb)
65
66     me.send_rc_control(0, fb, 0, speed)
67
68     return error
69
70 while True:
71     img = me.get_frame_read().frame
72     img = cv2.resize(img, (w, h))
73     img, info = findFire(img)
74     pError = trackFire(info, w, pid, pError)
75
76     cv2.imshow("Output", img)
77
78     if cv2.waitKey(1) & 0xFF == ord('q'):
79         me.land()
80         break
```

Here I import and install the necessary libraries and packages that I use

Now I utilise the 'djitellopy' library to set up a connection from my computer to the Tello Drone as well as get a reading of its current battery level, start the live camera stream and launch the drone

The first value is the size of the stream, the second is the forward back range, which is how close or far away the fire can be and the third is how far left and right the fire can be from the drone

This refers to the file path of the haar cascade so that that tells the AI whether it sees fire or not. Then I set parameters for the breadth of fires that I want the drone to detect before finding the most prominent fire for the drone to track

Here I tell the AI to draw a red rectangle around the detected fires

This finds the centre of the fire which will be the focus point of the drone and where the rectangle will be drawn around, and draws a green circle there

Now I ensure that the drone does not do anything if no fire is detected

'error = x - w // 2' finds how far left or right the drone is from the centre of the fire and adjusts itself accordingly.

pid slows the drone down gradually so as not to overestimate the distance required to travel

This tells the drone where to move in order to follow the fire. If the area is between the two 'fbRange' values then it won't move at all, if it is too close (it is too big) it will move back, and if it is too far (fire is too small) it will move forwards

Here I am saying that if the fire is smaller the area is also smaller, and if it is larger the area is also consequently larger. This is how the drone will decide whether to move forwards if backwards.

The next script here maps the path of the drone and its' approximate location

```

1 from djitellopy import tello
2 import KeyPressModule2 as kp
3 import numpy as np
4 from time import sleep
5 import cv2
6 import math
7
8 ##### PARAMETERS #####
9 fSpeed = 117/10 # Forward Speed (cm/s) 15cm/s
10 aSpeed = 360/10 # Angular Speed (Degrees/s) 50 degrees/s
11 interval = 0.5
12
13 dInterval = fSpeed*interval
14 aInterval = aSpeed*interval
15 #####
16 x, y = 500, 500
17 a = 0
18 yaw = 0
19
20 kp.init()
21 me = tello.Tello()
22 me.connect()
23 print(me.get_battery())
24
25 points = [(0, 0), (0, 0)]
26
27 def getKeyboardInput():
28     lr, fb, ud, yv = 0, 0, 0, 0
29     speed = 15
30     aspeed = 50
31     global x, y, yaw, a
32     d = 0
33
34     if kp.getKey("LEFT"):
35         lr = -speed
36         d = dInterval
37         a = -180
38
39     elif kp.getKey("RIGHT"):
40         lr = speed
41         d = -dInterval
42         a = 180
43
44     if kp.getKey("UP"):
45         fb = speed
46         d = dInterval
47         a = 270
48
49     elif kp.getKey("DOWN"):
50         fb = -speed
51         d = -dInterval
52         a = -90
53
54     if kp.getKey("w"):
55         ud = speed
56
57     elif kp.getKey("s"):
58         ud = -speed
59
60     if kp.getKey("a"):
61         yv = -aspeed
62         yaw -= aInterval
63
64     elif kp.getKey("d"):
65         yv = aspeed
66         yaw += aInterval
67
68     if kp.getKey("q"): me.land(); sleep(3)
69     if kp.getKey("e"): me.takeoff()
70
71     sleep(interval)
72     a += yaw
73     x += int(d*math.cos(math.radians(a)))
74     y += int(d * math.sin(math.radians(a)))
75
76     return [lr, fb, ud, yv, x, y]
77
78 def drawPoints(img, points):
79     for point in points:
80         cv2.circle(img, point, 5, (0, 0, 255), cv2.FILLED)
81     cv2.circle(img, points[-1], 8, (0, 255, 0), cv2.FILLED)
82     cv2.putText(img, f'({points[-1][0] - 500} / 100), {(points[-1][1] - 500) / 100})m',
83                 (points[-1][0] + 10, points[-1][1] + 30), cv2.FONT_HERSHEY_PLAIN, 1,
84                 (255, 0, 255), 1)
85
86 while True:
87     vals = getKeyboardInput()
88     me.send_rc_control(vals[0], vals[1], vals[2], vals[3])
89
90     img = np.zeros((1000, 1000, 3), np.uint8)
91     if (points[-1][0] != vals[4] or points[-1][1] != vals[5]):
92         points.append((vals[4], vals[5]))
93     drawPoints(img, points)
94     cv2.imshow("Output", img)
95     cv2.waitKey(1)

```

Here I once again import and install the necessary libraries and packages that I use. 'KeyPressModule2' is a separate file that I created previously that allows the drone to be controlled by my laptop keyboard

This allows me to set the speed at which the drone can move forwards/backwards and left/right. The formula below calculates the distance and angle of the drone every unit it travels

Now I initialise the drone by setting its' starting location at the centre of the map as well as connecting to it

As the drone moves either left, right, forwards(up), or backwards(down), it tells the computer where it has been and that is plotted on the map.

This function converts where the drone has travelled according to the keys pressed to a sine and cosine value, allowing the x and y coordinates to be calculated

This draws the points that the drone has travelled to. The first circle is where the drone has been, the second circle is where it currently is, and the text is the current coordinates of the drone

Here I use the numpy library to create a matrix (the map). As points that the drone has travelled to are added they are also plotted on the map at their exact location.

This Arduino sketch creates the webpage for the firefighting drone to display its monitoring system

```
1 #ifndef UNIT_TEST
2 #include <Arduino.h>
3 #include <ESP8266WiFi.h>
4
5 #endif
6 #include <ESP8266WiFi.h>
7
8 const char* ssid = "WiFi-GWW8";
9 const char* password = "abcd1974";
10
11 int ledPin = 13; // GPIO13---D7 of NodeMCU
12 WiFiServer server(80); //the port
13 IPAddress ip(192, 168, 1, 2);
14 IPAddress gateway(192, 168, 43, 1);
15 IPAddress subnet(255, 255, 255, 0);
16 IPAddress dns(192, 168, 43, 1);
17
18 void setup(void) {
19   Serial.begin(115200);
20   delay(10);
21
22   pinMode(ledPin, OUTPUT);
23   digitalWrite(ledPin, LOW);
24
25   // Connect to WiFi network
26   Serial.println();
27   Serial.println();
28   Serial.print("Connecting to ");
29   Serial.println(ssid);
30
31   //Static IP Setup Info Here...
32   WiFi.config(ip, dns, gateway, subnet);
33   WiFi.begin(ssid, password);
34   while (WiFi.status() != WL_CONNECTED) {
35     delay(500);
36   }
37   Serial.print(".");
38 }
39 Serial.println("");
40
41 Serial.println("WiFi connected");
42
43 // Start the server
44 server.begin();
45 Serial.println("Server started");
46 // Print the IP address
47 Serial.print("Use this URL to connect: ");
48 Serial.print("http://");
49 Serial.print(WiFi.localIP());
50 Serial.println("/");
51 }
52
53 void loop()
54 {
55   // Check if a client has connected
56   WiFiClient client = server.available();
57   if (!client) {
58     return;
59   }
60
61   // Wait until the client sends some data
62   Serial.println("new client");
63   while(!client.available()){
64     delay(1);
65   }
66   client.setNoDelay(1);
67
68   // Read the first line of the request
69   String request = client.readStringUntil('\r');
70   Serial.println(request);
71   client.flush();
72
73   // Match the request
74   int value = LOW;
75   if (request.indexOf("/LED=ON") != -1) {
76     digitalWrite(ledPin, HIGH);
77     value = HIGH;
78   }
79   if (request.indexOf("/LED=OFF") != -1) {
80     digitalWrite(ledPin, LOW);
81     value = LOW;
82   }
83
84   // Set ledPin according to the request
85
86
```

Here I import and install the necessary libraries and packages that I use

Now I enter the Wi-Fi details that I want the NodeMCU to connect to

This tells the board that the LED that I want to control is on pin 13, and then I set up the IPAddress for the webpage so that I get the same IPAddress every time

This is where the NodeMCU sets up the Wi-Fi connection and prints out the IPAddress for the webpage

This is where the NodeMCU sets up the Wi-Fi connection and prints out the IPAddress for the webpage

```

87 // Return the response
88 client.println("HTTP/1.1 200 OK");
89 client.println("Content-Type: text/html");
90 client.println(""); // do not forget this one
91 client.println("");
92 client.println("<!DOCTYPE HTML>");
93
94 client.println("<head><meta name='viewport' content='width=device-width, initial-scale=1'>");
95 client.println("<link rel='icon' href='data:;'>");
96 // CSS to style the on/off buttons
97 client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;};");
98 client.println("</style>");
99 client.println("<meta http-equiv='refresh' content='10'>");
100 client.println("<title>Firefighter Control Board</title>");
101 client.println("<center>");
102 client.println("<div style = 'color:white'>Firefighter Control Board<br>Caleb Tang, Year 9, Prince Alfred College<br></div>");
103 client.println("</div>");
104 client.println("</div>");
105 client.println("</div>");
106 client.println("<div style = 'color:white'>Firefighter Control Board<br>Caleb Tang, Year 9, Prince Alfred College<br></div>");
107 client.println("<div style = 'background-color:orange;'>");
108 client.println("<div>");
109
110 float C_temperature = C_tempmonitor();
111 float F_temperature = C_to_F(C_temperature);
112
113 client.println("<div style = 'color:green'>The current temperature of the fire is: <br><br></div>");
114 client.println("<div style = 'color:red'><div style = 'font-size:30px;'><div style = 'font-size:30px;'>");
115 client.println("<div style = 'font-size:30px;'><div style = 'font-size:30px;'>");
116 client.println("<div style = 'font-size:30px;'><div style = 'font-size:30px;'>");
117 client.println("<div style = 'font-size:30px;'><div style = 'font-size:30px;'>");
118 client.println("<div style = 'font-size:30px;'><div style = 'font-size:30px;'>");
119 client.println("<div style = 'font-size:30px;'><div style = 'font-size:30px;'>");
120 client.println("<div style = 'font-size:30px;'><div style = 'font-size:30px;'>");
121 client.println("<div style = 'font-size:30px;'><div style = 'font-size:30px;'>");
122 client.println("<div style = 'font-size:30px;'><div style = 'font-size:30px;'>");
123 client.println("<div style = 'font-size:30px;'><div style = 'font-size:30px;'>");
124 client.println("<div style = 'font-size:30px;'><div style = 'font-size:30px;'>");
125 client.println("<div style = 'font-size:30px;'><div style = 'font-size:30px;'>");
126 client.println("<div style = 'font-size:30px;'><div style = 'font-size:30px;'>");
127 client.println("<div style = 'font-size:30px;'><div style = 'font-size:30px;'>");
128
129 client.println("<div style = 'color:blue'>Water Deployment is: <br><br></div>");
130 client.println("<div style = 'color:blue'>Water Deployment is: <br><br></div>");
131 client.println("<div style = 'color:blue'>Water Deployment is: <br><br></div>");
132
133 if(value == HIGH) {
134   client.print("On");
135 } else {
136   client.print("Off");
137 }
138 client.println("<br><br>");
139 client.println("<a href='\"/LED=ON\"'><button>On </button></a>");
140 client.println("<a href='\"/LED=OFF\"'><button>Off </button></a><br />");
141 client.println("</html>");
142
143 Serial.println("Client disconnected");
144 Serial.println("");
145 delay(500);
146 }
147
148 float C_tempmonitor()
149 {
150
151   int reading = analogRead(A0);
152   float voltage = (reading * 3.15) / 1024;
153   //convert from 10 mV per degree with 500 mV offset
154   //to degrees ((voltage - 500 mV)*100)
155   return (voltage - 0.5) * 100;
156 }
157 float C_to_F(float C_temperature)
158 {
159
160   return (C_temperature * 9/5) + 32;
161 }

```

Here I connect the NodeMCU to the webpage before starting the HTML section

This is where I declare the font that I will use and the size that I want the text to be. I also use a function to automatically refresh the page every 10 seconds so that the temperature updates in live time

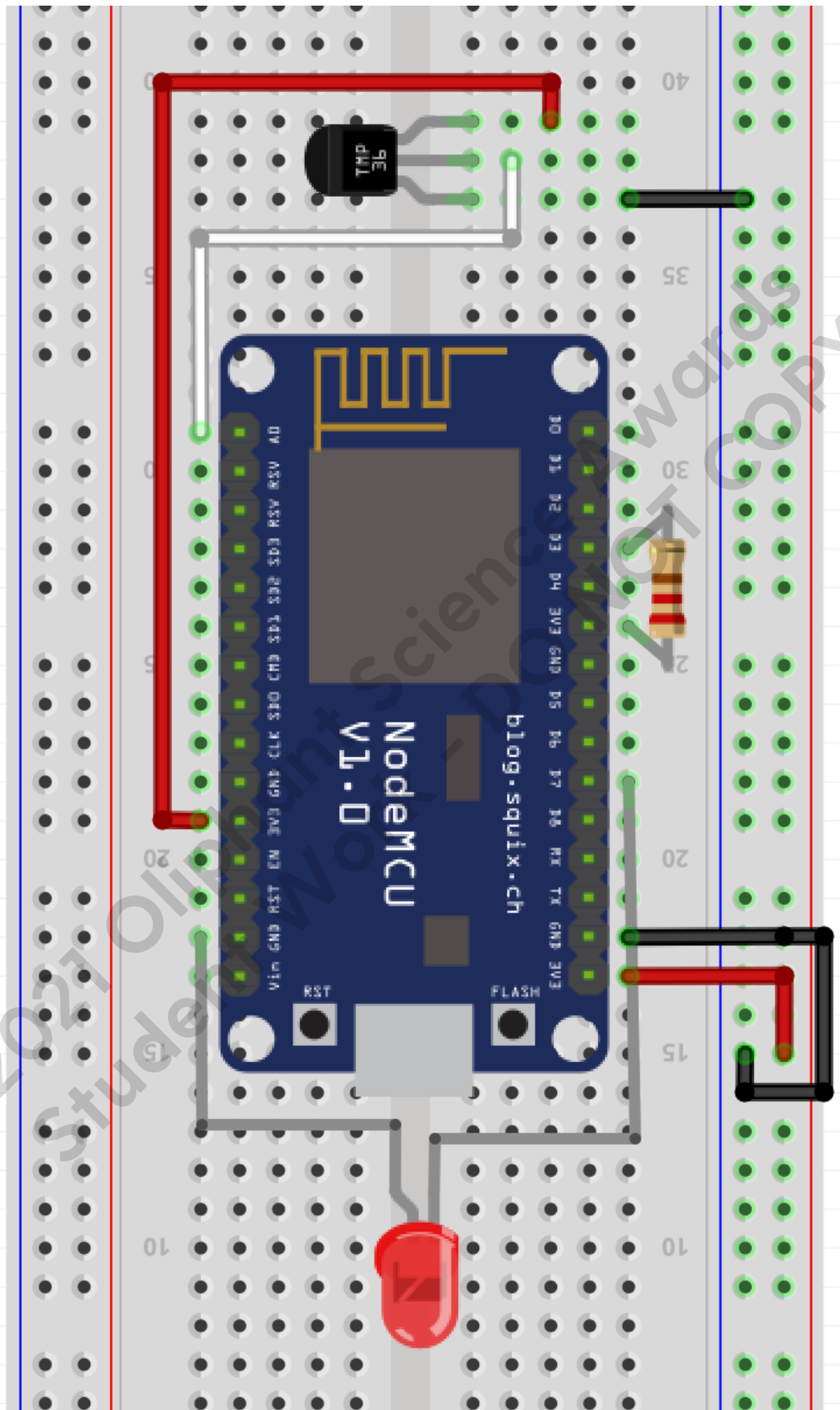
I will use these two 'float functions later to calculate the temperature from the voltage that the TMP32 thermistor receives as well as convert that temperature to degrees Fahrenheit

Here I continue to develop the webpage. 'C_temperature' and 'F_temperature' are variables that are calculated later

This is where I make the buttons on the webpage that turn the LED that signifies water being deployed 'ON' and 'OFF'. I also add either 'LED=ON' or 'LED=OFF' to the end of the IP Address depending on whether the LED is on or off

This is a formula that has been made to calculate the temperature from a TMP32 thermistor and also convert it from degrees Celsius to degrees Fahrenheit

This schematic is a rough idea of what the wiring of the monitoring system should look like.



Acknowledgements

I am extremely grateful for the help that I have received from the several online forums that I have used, including Arduino, GitHub, Stack Overflow, Adafruit, ESP, and Steemit in particular. Medium, Towards Data Science and Python Programming articles have also been highly useful, as well as numerous YouTube Videos. I would also like to acknowledge my parents for their constant and ongoing support throughout my project.

References

- En.wikipedia.org. (2021). *Inductive charging*. [online] Available at: https://en.wikipedia.org/wiki/Inductive_charging [Accessed 12 June. 2021].
- Forum.arduino.cc. (2021). *Arduino Forum - Index*. [online] Available at: <https://forum.arduino.cc/>.
- Question, C., ..., F., t..., C., Production..., H. and promp..., D. (2021). *GitHub Community Forum*. [online] Github.community. Available at: <https://github.community/>.
- US EPA. (2021). *What if Garbage Fumes Powered More of Our Cars, Trucks, and Buses? | US EPA*. [online] Available at: <https://www.epa.gov/greenvehicles/what-if-garbage-fumes-powered-more-our-cars-trucks-and-buses> [Accessed 12 June. 2021].
- Aph.gov.au. 2021. *2019–20 Australian Bushfires—Frequently Asked Questions: A Quick Guide – Parliament Of Australia*. [online] Available at: https://www.aph.gov.au/About_Parliament/Parliamentary_Departments/Parliamentary_Library/pubs/rp/rp1920/Quick_Guides/AustralianBushfires. [Accessed 12 June 2021]
- Duncombe, J., 2021. *Five Environmental Consequences Of Australia'S Fires - Eos*. [online] Eos. Available at: <https://eos.org/articles/five-environmental-consequences-of-australias-fires> [Accessed 18 June 2021].
- Pythonprogramming.net. 2021. *Python Programming Tutorials*. [online] Available at: <https://pythonprogramming.net/haar-cascade-object-detection-python-opencv-tutorial/> [Accessed 22 June 2021].
- Docs.opencv.org. 2021. *OpenCV: Cascade Classifier*. [online] Available at: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html [Accessed 22 June 2021].
- Medium. 2021. *Haar Cascades, Explained*. [online] Available at: <https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d> [Accessed 22 June 2021].
- GitHub. 2021. *GitHub - dji-sdk/Tello-Python: This is a collection of python modules that interact with the Ryze Tello drone..* [online] Available at: <https://github.com/dji-sdk/Tello-Python> [Accessed 22 June 2021].