Object tracking laser turret system with Raspberry PICO

Project Description

In this project, we'll create a laser turret system that can detect and track the palm process the coordinates and then send the coordinates to the pico board which will use these coordinates to the servo motors and focus the laser on the detected object. All the detection,processing,and coordinate mapping is done by opencv and python.

How to Use servo motors,laser with Raspberry PICO

The servo motors are connected to the Pico board which will provide the X,Y coordinates to the laser and guide it to the detected object. There are two servos used in this project one is for the vertical motion and one is for the horizontal motion[X-Y Axis]. The pico board will receive the XY coordinates; these coordinates are sent to the servo motors which will move the laser pointer accordingly. The laser is powered by the pico board and is attached to the Y axis servo motor.

Things used in this project

1 Hardware

- Wiznet Pico board
- Jumper wires
- 2x SG90 Servo Motors
- Red laser pointer
- Bread Board
- Web Cam

2 Software

- Arduino IDE
- Python
- Libraries Included Arduino : Servo
- Libraries Included Python : cv2, mediapipe, numpy, serial

Story

The project is inspired by laser turret systems placed in battle ships which uses radar and video capturing to detect incoming fighter jets and missiles ,track them and shoot them down with high accuracy. The Israel Iron Dome system is one of the most advanced versions of these systems to exist. This project uses a similar tracking system using AI . The coordinates after tracking are calculated and sent to the PICO board in an encoded format. The pico board will compute the X-Y coordinates and send these to the X-Y servo motors respectively. A laser is attached to the Y servo motor as for the pointing. As the detected object moves the laser pointers will also move according to it in realtime. Serial communication is established between the detection and pico board at 115200 baud rate. The servo will be active only when serial data is received. This eliminates the need of writing the same coordinates to the servo over and over again improving latency of the servo movement.

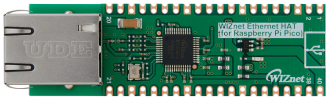
Component Used	Image	Description
Wiznet PICO ethernet hat		Controls and coordinates the pump servo motors and powers the laser beams
Breadboard and Jumper wire		Helps with interconnecting circuit elements with the Raspberry PICO
USB Webcam		Captures plant images to the YOLOV5 Object detection model for analysis
3V Laser		Used for pointing at the detected object

Table 1. Components

Circuit Diagram

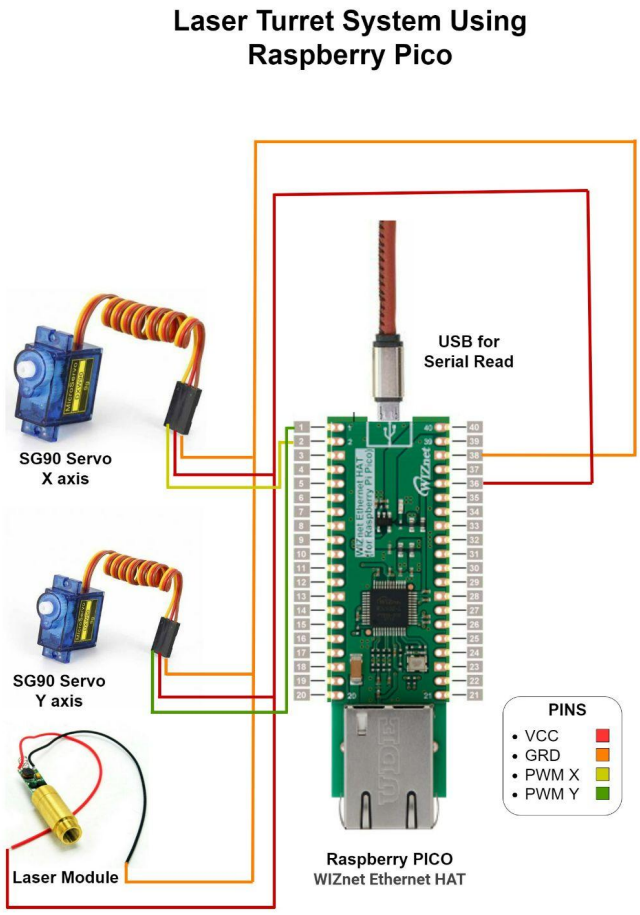


Figure 12. Component Circuit Diagram

AI Integration

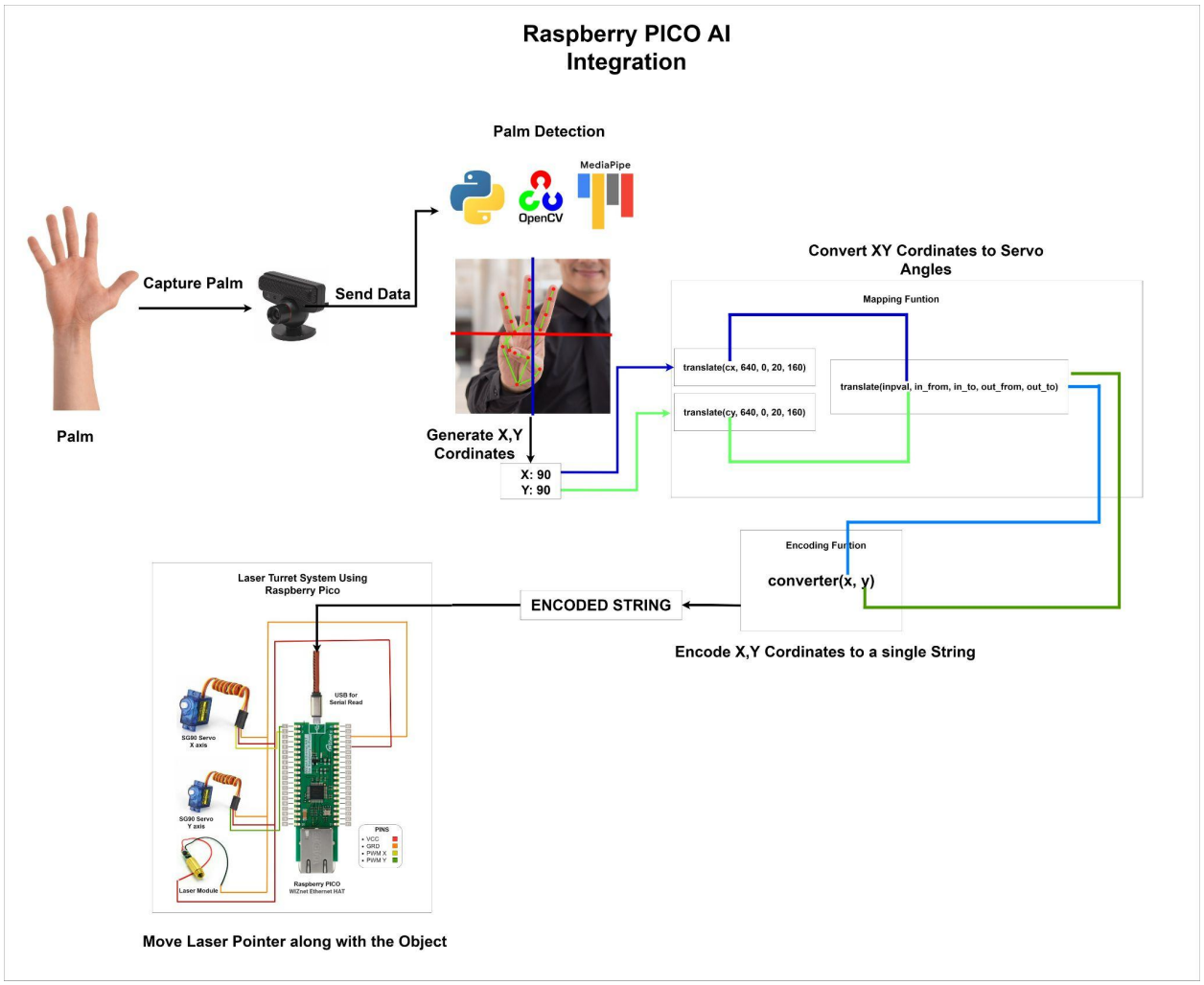


Figure 13. AI Integration in Raspberry Pico

Circuit diagram Explanation

VCC , GND[Pin 36, Pin 38] are connected to the X and Y Servo motors and the laser pointer. PWM of the X-Axis servo is connected to GPIO 0 [Pin 2]. PWM of the X-Axis servo is connected to GPIO 1 [Pin 2]. Serial USB is attached to the Pico to read the incoming coordinate string. The VCC pin supplies 3.5V output which will power the laser pointer.

How to integrate AI ?

In this project OpenCV and Mediapipe are used for palm detection. The X and Y coordinates of the palm of a certain point. These X and Y coordinates are then translated to servo angles using the translate function. Two prevent two writes for X and Y coordinates the data is combined to a single string and is passed to the Serial COM port. This is done for each frames The String is encoded in ‘utf-8’ format.

Python AI Code

```
#Importing Libraries
import cv2
import mediapipe as mp
import time
import numpy as np
import serial

#Setting arduino serial
arduino = serial.Serial(port='COM3', baudrate=115200, timeout=.1)

#Initialize capturing
cap = cv2.VideoCapture(0)

#use hands class
mpHands = mp.solutions.hands
hands = mpHands.Hands() # using default parameters of 'Hands()'
mpDraw = mp.solutions.drawing_utils #to draw landmarks

while True:
 #A frame is captured
 success, img = cap.read()
 #converting colorspace to RGB
 imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
 #process the converted results
 results = hands.process(imgRGB)

 #setup horizontal and diagonal lines
 cv2.line(img, (340, 0), (340, 480), (0, 0, 255), 1)
 cv2.line(img, (0, 240), (640, 240), (0, 255, 255), 1)

 #if palm is detected
 if results.multi_hand_landmarks:
     #for each landmarks
     for handLms in results.multi_hand_landmarks:
         for id, lm in enumerate(handLms.landmark):
             h, w, c = img.shape

```

```
38              #getting x and y coordinates of the palm
39              cx, cy = int(lm.x * w), int(lm.y * h)
40
41              #if finger is detected
42              if id == 7: # finger 1
43                  print("x:"+str(cx)+"y:"+str(cy))
44
45                  #translate function converts coordinates to angle
46                        def translate(inpval, in_from, in_to, out_from,
out_to):
47                      out_range = out_to - out_from
48                      in_range = in_to - in_from
49                      in_val = inpval - in_from
50                      val = (float(in_val) / in_range) * out_range
51                      out_val = out_from + val
52                      return int(out_val)
53
54                     #converter function encodes X,Y coordinates into 1
string
55                  def converter(x, y):
56                      encode = 10000000
57                      xencode = encode + (x * 10000)
58                      yencode = encode + y
59                      return xencode + yencode
60
61                  #to convert X,Y to angles
62                  datax = translate(cx, 640, 0, 20, 160)
63                  datay = translate(cy, 00, 480, 40, 150)
64
65                  #to encode to single string
66                  coordinates = converter(datax,datay)
67                  print(coordinates)
68
69                  #write_read function writes to Serial COM Port
70                  def write_read(x):
71                      arduino.write(bytes(x, 'utf-8'))
72                      time.sleep(0.05)
73                      data = arduino.readline()
74                      return data
75
```

```
                # coordinates = 20900090 => To set LED to midpoint

                #to write to serial
                value = write_read(str(coordinates))
                print(value)
        #draw the lines for palm and fingers
        mpDraw.draw_landmarks(img, handLms, mpHands.HAND_CONNECTIONS)

#display the captured frame
cv2.imshow('Image', np.flip(img,axis=1))
#for exiting the capture
cv2.waitKey(1)
```

Python AI Code Explanation

First we import all the libraries required for capturing,processing and writing to Serial Port,line number 2-6. We initialize the serial communication in line number 9

```
arduino = serial.Serial(port='COM3', baudrate=115200, timeout=.1)
```

Serial port is given as COM3, Baudrate is set to 115200 and a timeout of 0.1 second is given for each writes.

```
11 #Initialize capturing
12 cap = cv2.VideoCapture(0)
13
14 #use hands class
15 mpHands = mp.solutions.hands
16 hands = mpHands.Hands() # using default parameters of 'Hands()'
17 mpDraw = mp.solutions.drawing_utils #to draw landmarks
```

OpenCV video capturing is enabled as set to 0 for internal webcam. We use a hands class from mediapipe for palm detection. Line 19-82 is an infinite loop which captures frames until the program is closed. Line no 28,29 sets up horizontal and vertical lines for the capture screen. The midpoint has to be aligned with the laser pointer.

```
27  #setup horizontal and diagonal lines
28  cv2.line(img, (340, 0), (340, 480), (0, 0, 255), 1)
29  cv2.line(img, (0, 240), (640, 240), (0, 255, 255), 1)
```

Line no 20-25 converts the captured frame to an RGB colorspace and is then given for processing.

```
20  #A frame is captured
21  success, img = cap.read()
22  #converting colorspace to RGB
23  imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
24  #process the converted results
25  results = hands.process(imgRGB)
```

```
31  #if palm is detected
32  if results.multi_hand_landmarks:
33      #for each landmarks
34      for handLms in results.multi_hand_landmarks:
35          for id, lm in enumerate(handLms.landmark):
36              h, w, c = img.shape
37
```

```
38              #getting x and y coordinates of the palm
39              cx, cy = int(lm.x * w), int(lm.y * h)
40
41              #if finger is detected
42              if id == 7: # finger 1
43                  print("x:"+str(cx)+"y:"+str(cy))
```

From line number 31 to 43, if palm is detected the X,Y coordinates of the palm is calculated from by multiplying with screen width and height

```
cx, cy = int(lm.x * w), int(lm.y * h)
```

A translate function is used to convert the XY coordinates to Servo Angles. This done by mapping the X Coordinate which range from 640 to 0 the Servo angle range 20 to 160.Similiarly the Y coordinate is converted from range 0 to 480 to the Servo angle range 40 to 150.

```
datax = translate(cx, 640, 0, 20, 160)
datay = translate(cy, 00, 480, 40, 150)
```

The translate function is then called, the inpval variable is the coordinate

```
46                          def translate(inpval, in_from, in_to, out_from,
out_to):
47                      out_range = out_to - out_from
48                      in_range = in_to - in_from
49                      in_val = inpval - in_from
50                      val = (float(in_val) / in_range) * out_range
51                      out_val = out_from + val
52                      return int(out_val)
```

The translate function has the following arguments inpval which is the input coordinate, in_from and in_to which is the range of the incoming coordinates. out_from and out_to are the angles to which the range has to be converted. Once the conversion is done these values will be stored to variables datax and datay for X and Y coordinates respectively.

Line no 70-74 is to encode the X and Y coordinates to a single string. This is done by the converter function which takes two arguments x,y which are the translated angles.

```
55                  def converter(x, y):
56                      encode = 10000000
57                      xencode = encode + (x * 10000)
58                      yencode = encode + y
59                      return xencode + yencode
```

In line no 70-74 the translated and encoded data is sent to the Serial COM port

```
70                  def write_read(x):
71                      arduino.write(bytes(x, 'utf-8'))
72                      time.sleep(0.05)
73                      data = arduino.readline()
74                      return data
```

In Line number 84 to 87 the processed and landmarked capture is shown .It is inverted in the X axis to show the real image.

Arduino Code

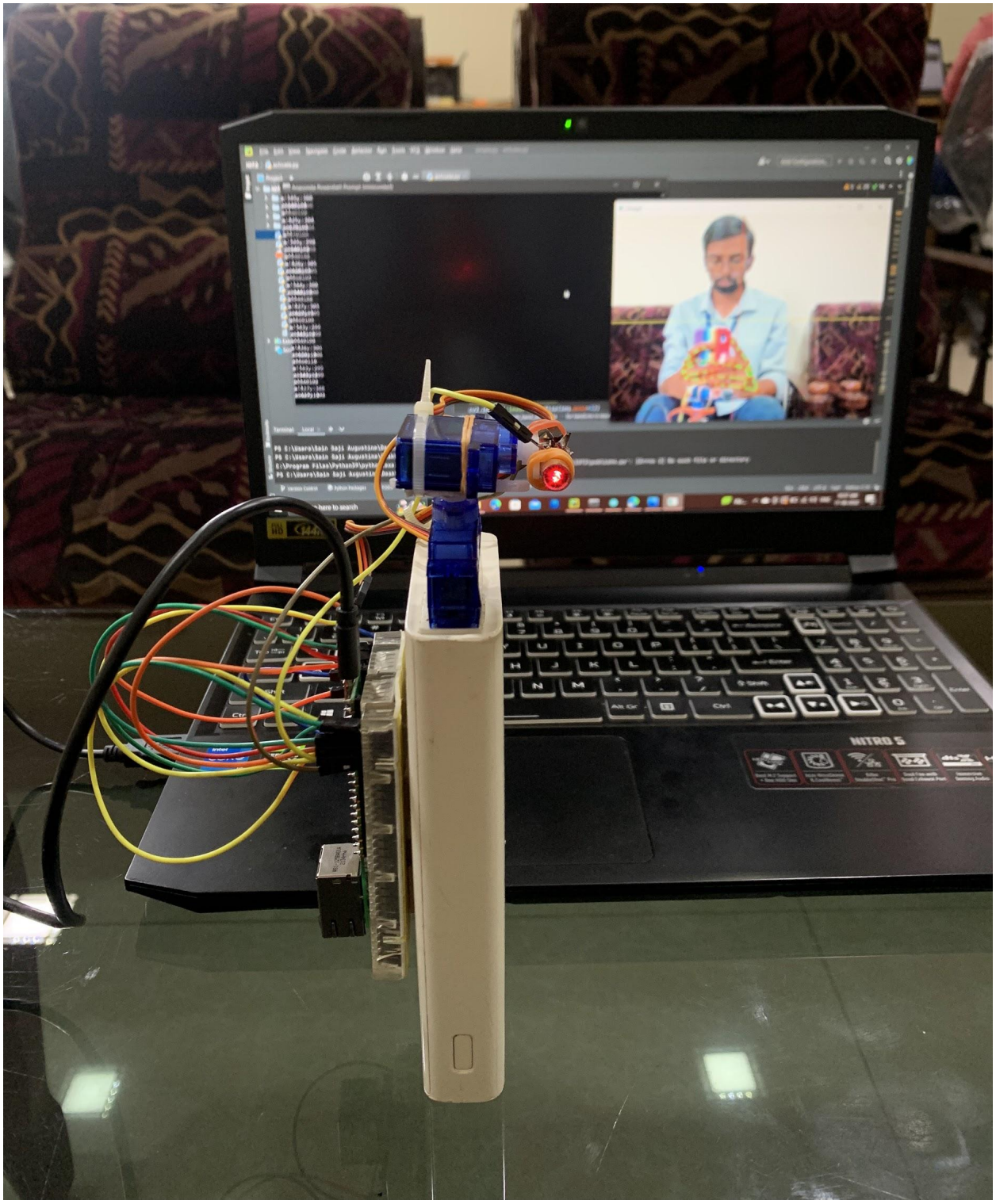
```
1 //Libraries to Include
2 #include <Servo.h>
3
4 //Servo Initialization
5 Servo myservo;
6 Servo myservo2;
7
8 //Variable initialisation
9 String serialdata; //For incoming serial data
10 int midpoint = 90; //For setting the laser pointer to the middle of the
frame
11
12 void setup() {
13   myservo2.attach(1); //Y coordinates servo
14   myservo.attach(2);  //X coordinates servo
15   Serial.begin(115200); //Enabling Serial Communication
16   Serial.setTimeout(10); //Setting a Serial Timeout
17
18   //Initialize XY with 90 to set the midpoint
19   myservo.write(midpoint);    //set X to 90
20   myservo2.write(midpoint);   //Set Y to 90
21 }
22
23 void loop()
24 {
25   //
26 }
27
28
29 int x,y=minpoint; //setting initial coordinates as middle of the frame
30
31 //this function is triggered while receiving a serial data
32 void serialEvent()
33 {
34   serialdata = Serial.readString();     //Reads incoming string serial
data
35    int coordinates = serialdata.toInt();   //converts incoming encoded
string coordinates to integer for calculation
```

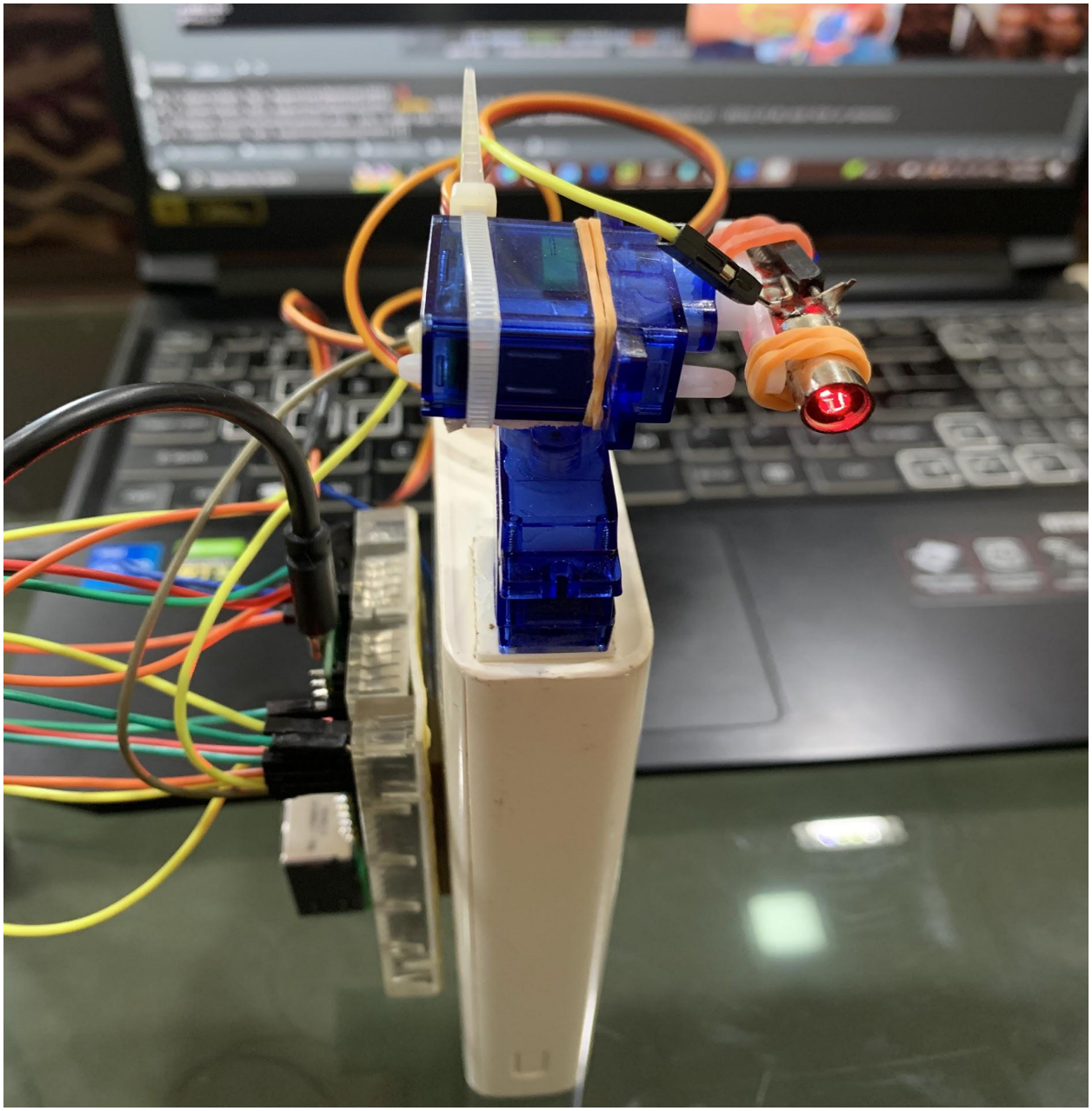
```
36
37   //calculating X-Y coordinates from the encoded string
38   //coordinates will be in the form 2xxx0yyy eg:20390107
39   //where xxx is the X coordinate and yyy will be Y coordinate eg:X=39
and Y=107
40    y = coordinates%1000;           // to isolate the last 3 digits : Y
coordinate eg: CCCCC107
41    coordinates = coordinates-y;     // subtract the Y coordinates from
the original string eg: 20390107 - 107 = 20390000
42   x = (coordinates/10000)-2000;  // divide the coordinates by 100000 to
get the first 4 digits and subtract it from 2000 to get the x coordinates
eg: 2039 - 2000 = 39 -> X coordinates
43
44   myservo2.write(int(y)); //write y coordinates to the Y servo motor
45   myservo.write(int(x));  //write x coordinates to the X servo motor
46 }
```

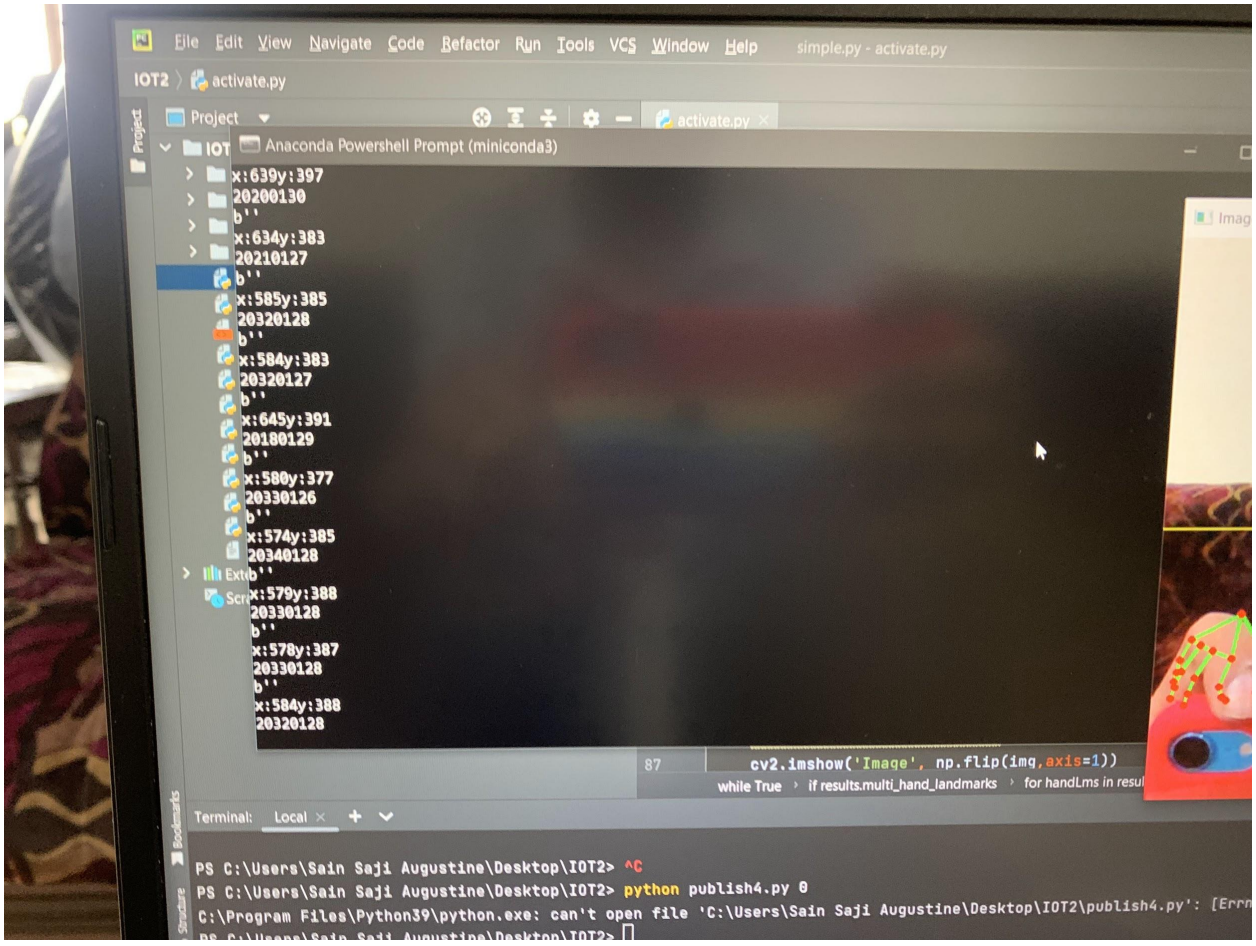
Arduino Code Explanation

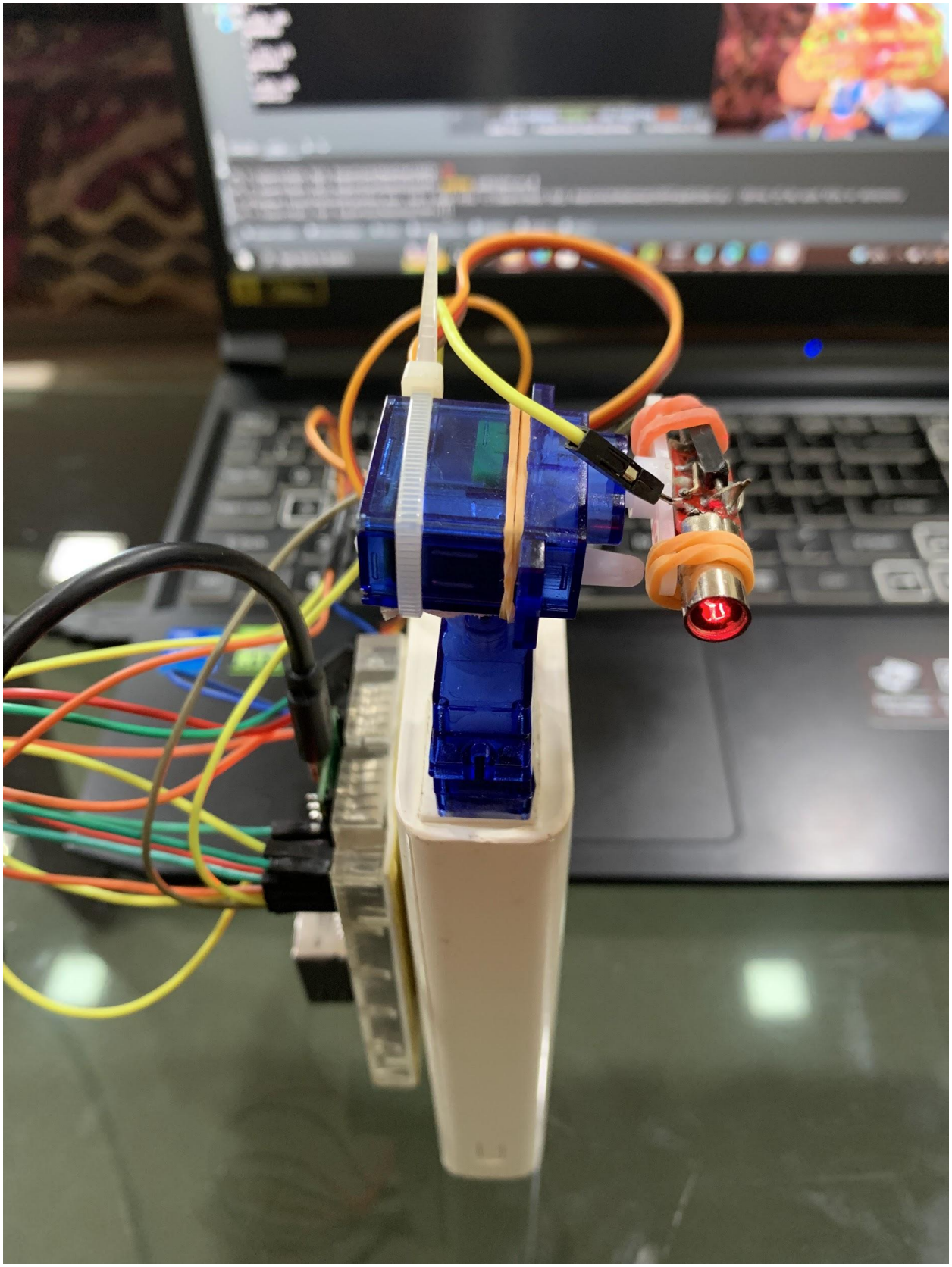
The code starts from line number 2,which includes the Servo library which activates the servo motor. Two servos are initialized in line number 5,6.Two variables are initialized in line number 9,10 in which serialdata is to capture incoming serial string and a variable midpoint which is to set the laser pointer to the middle of the screen. In the void setup() function the Servo motors are attached and Serial is enabled at baudrate = 115200 and a time out of 1 millisecond is given for the incoming serial data. The servos are initialized to an angle of 90,to set the laser to middle in the initial position. Line number 23 to 26 is set as empty as we are not continuously checking for the incoming serial data. A function is written to check for any serial communication in line no 32-46. It reads the incoming serial data in String format; it is then converted into integer format for calculating the X and Y coordinates. The string is in encoded format which has both the X and Y coordinates. Cordinates will be in the form 2xxx0yyy eg:20390107.where xxx is the X coordinate and yyy will be Y coordinate eg:X=39 and Y=107. This has to be extracted and it is done by line number 40-42 which sets two variables X and Y which are then written to the X and Y servo motors in line number 44 and 45.

Images

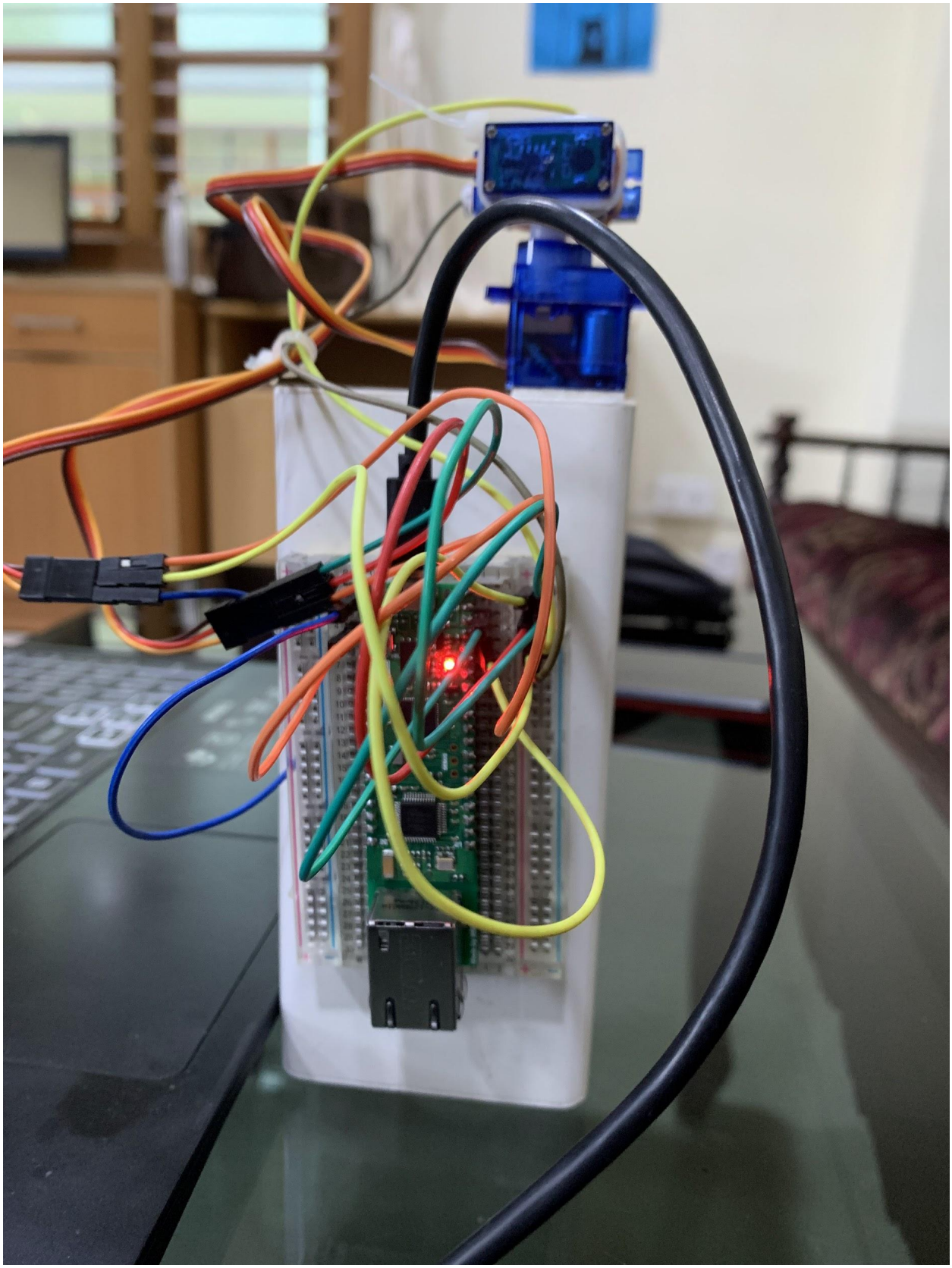

NITRO 5



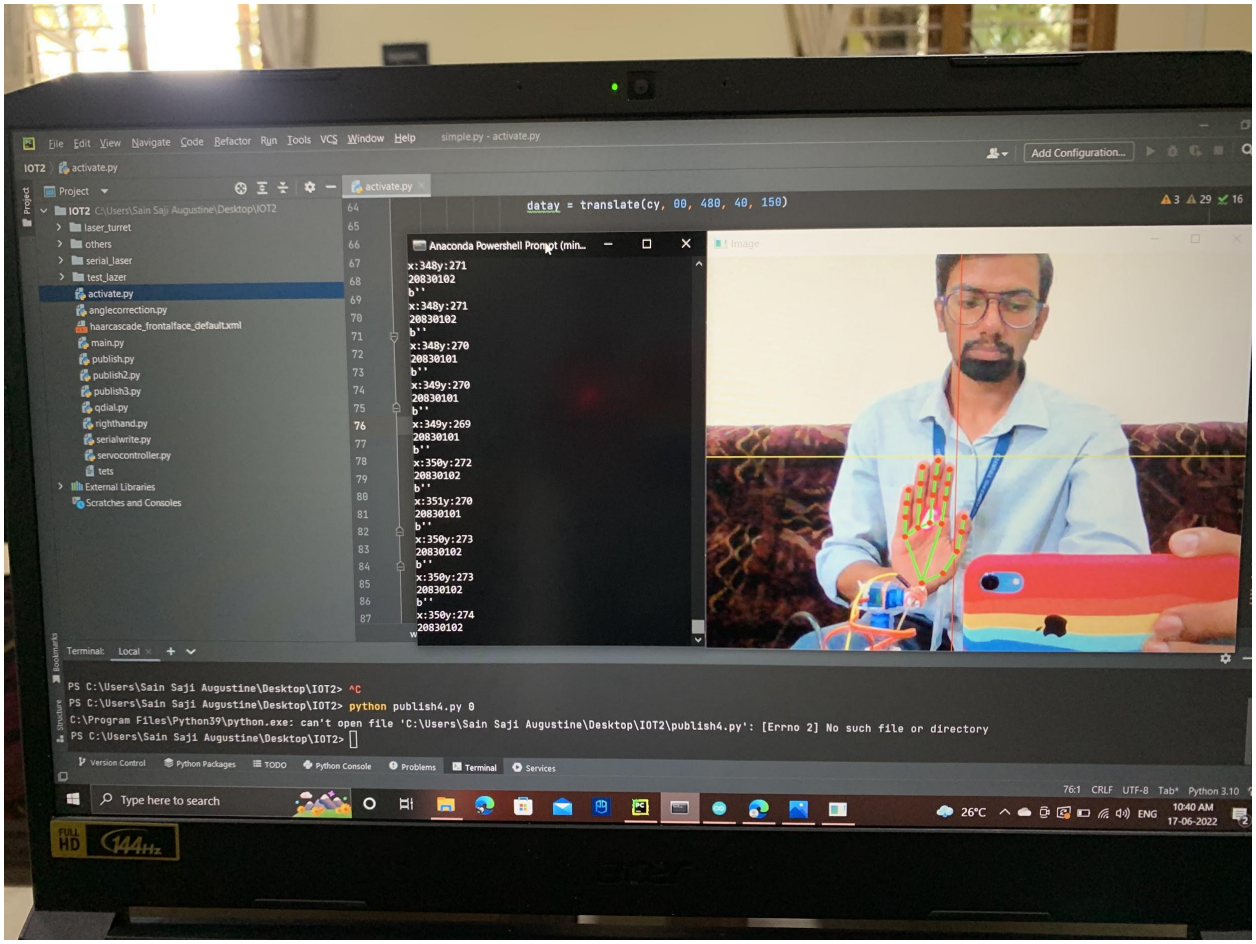

File Edit View Navigate Code Refactor Run Tools VCS Window Help
simple.py - activate.py
IOT2 activate.py
Project
activate.py
Anaconda Powershell Prompt (miniconda3)
x:639y:397
20200130
b''
x:634y:383
20210127
b''
x:585y:385
20320128
b''
x:584y:383
20320127
b''
x:645y:391
20180129
b''
x:580y:377
20330126
b''
x:574y:385
20340128
b''
x:579y:388
20330128
b''
x:578y:387
20330128
b''
x:584y:388
20320128
87
cv2.imshow('Image', np.flip(img,axis=1))
while True
if results.multi_hand_landmarks
for handLms in resul
Terminal: Local
PS C:\Users\Sain Saji Augustine\Desktop\IOT2> ^C
PS C:\Users\Sain Saji Augustine\Desktop\IOT2> python publish4.py 0
C:\Program Files\Python39\python.exe: can't open file 'C:\Users\Sain Saji Augustine\Desktop\IOT2\publish4.py': [Errn



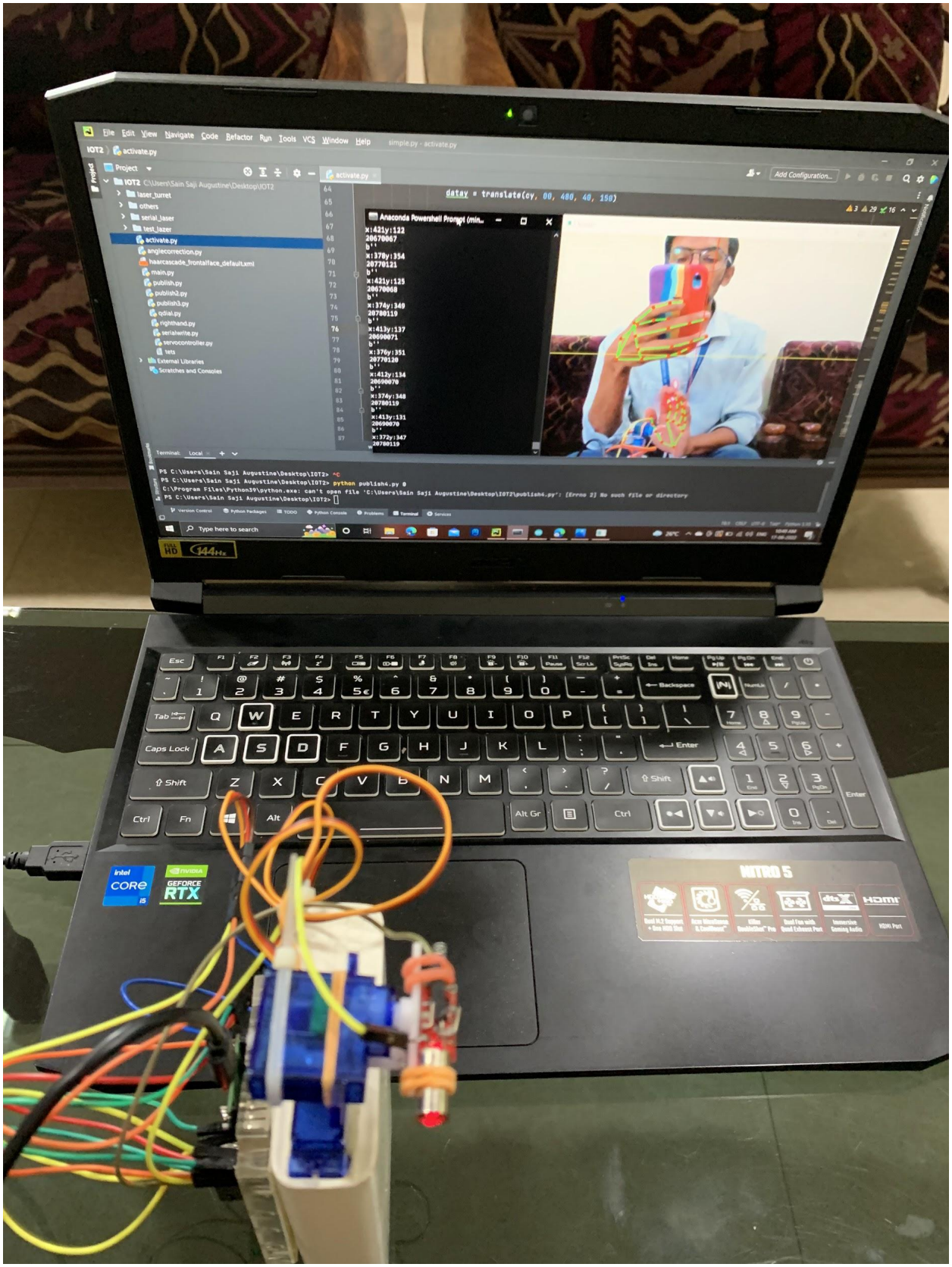
\

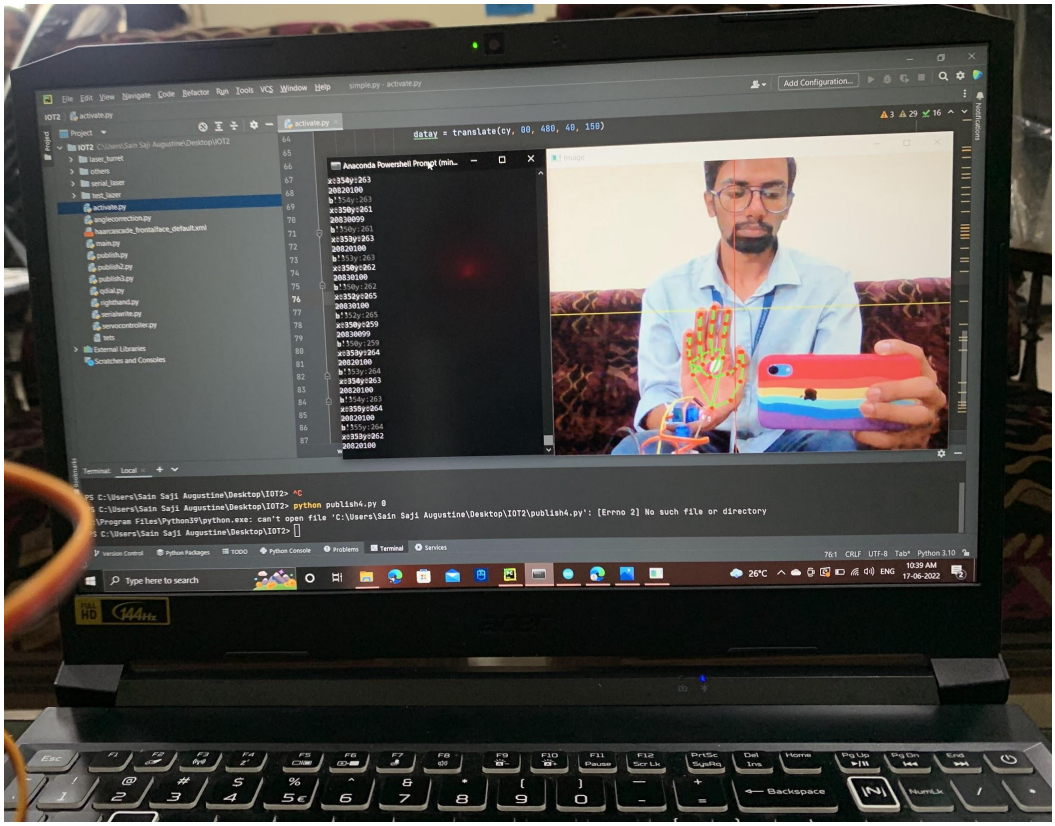



TOWER PRO
Micro Servo
9g
SG90


File Edit View Navigate Code Refactor Run Tools VCS Window Help
simple.py - activate.py
Add Configuration...
IOT2 activate.py
Project
IOT2 C:\Users\Sain Saji Augustine\Desktop\IOT2
laser_turret
others
serial_laser
test_lazer
activate.py
anglecorrection.py
haarcascade_frontalface_default.xml
main.py
publish.py
publish2.py
publish3.py
qdial.py
righthand.py
serialwrite.py
servocontroller.py
tets
External Libraries
Scratches and Consoles
datay = translate(cy, 80, 480, 40, 150)
Anaconda Powershell Prompt (min...
x:348y:271
20830102
b''
x:348y:271
20830102
b''
x:348y:270
20830101
b''
x:349y:270
20830101
b''
x:349y:269
20830101
b''
x:350y:272
20830102
b''
x:351y:270
20830101
b''
x:350y:273
20830102
b''
x:350y:273
20830102
b''
x:350y:274
20830102
Terminal: Local
PS C:\Users\Sain Saji Augustine\Desktop\IOT2> ^C
PS C:\Users\Sain Saji Augustine\Desktop\IOT2> python publish4.py 0
C:\Program Files\Python39\python.exe: can't open file 'C:\Users\Sain Saji Augustine\Desktop\IOT2\publish4.py': [Errno 2] No such file or directory
PS C:\Users\Sain Saji Augustine\Desktop\IOT2>
Version Control
Python Packages
TODO
Python Console
Problems
Terminal
Services
Type here to search
26°C
ENG
10:40 AM
17-06-2022
FULL HD
144Hz


intel
CORE
i5
NVIDIA
RTX


File Edit View Navigate Code Refactor Run Tools VCS Window Help
IOT2 activate.py
Project
IOT2 C:\Users\Sain Saji Augustine\Desktop\IOT2
laser_turret
others
serial_laser
test_lazer
activate.py
angleCorrection.py
haarcascade_frontalface_default.xml
main.py
publish.py
publish2.py
publish3.py
qdial.py
righthand.py
serialwrite.py
servocontroller.py
test
External Libraries
Scratches and Consoles
Add Configuration...
datay = translate(cy, 80, 480, 40, 150)
Anaconda Powershell Prompt (min...
x:421y:122
20670067
b''
x:378y:354
20770121
b''
x:421y:125
20670068
b''
x:374y:349
20780119
b''
x:413y:137
20690071
b''
x:376y:351
20770120
b''
x:412y:134
20690070
b''
x:374y:348
20780119
b''
x:413y:131
20690070
b''
x:372y:347
20780119
Terminal: Local
PS C:\Users\Sain Saji Augustine\Desktop\IOT2> ^C
PS C:\Users\Sain Saji Augustine\Desktop\IOT2> python publish4.py 0
C:\Program Files\Python39\python.exe: can't open file 'C:\Users\Sain Saji Augustine\Desktop\IOT2\publish4.py': [Errno 2] No such file or directory
PS C:\Users\Sain Saji Augustine\Desktop\IOT2>
Type here to search
intel CORE i5
NVIDIA GEFORCE RTX
NITRO 5



Video Link

References

- DIY Laser Turret | Part 1 The Hardware - YouTube
- How to Build a Laser Turret : 5 Steps (with Pictures) - Instructables
- Laser Turret - Arduino Project Hub