Game control Using AI and WIZnet EVB PICO Board

Project Description

In this project,we'll create a simple game and control the player movements using gesture detection. Two gestures are taken into consideration which are used to control the player movements and are captured by the gesture detection system. A player moves the basket by using gestures to capture randomly generated apples on the screen. Each capture earns 1 point.We use Socket programming to send the data from the detection code to the Wiznet EVB PICO using ethernet. The game is a time constrained one and once the set time is elapsed the game will stop the detection automatically and display the high score. The gesture detection and data transmission is done by opencv and python.

Things used in this project

1 Hardware

- Wiznet EVB PICO W5100S
- Jumper wires
- SSD1306 OLED Display
- Bread Board
- Web Cam

2 Software

- Arduino IDE
- Python
- Libraries Included Arduino : Adafruit Graphics Library, Ethernet and Ethernet UDP Library
- Libraries Included Python : Sockets Library, OpenCV, Mediapipe

Story

The first thing that comes to our mind listening to the words **Game Control** is the Joystick. By integrating AI as a control system for the gaming environment we can enable immersive gaming without the requirement for electronics. In this project We have made an AI based controller for a 2D game. This enables fine tuning of the movements . AI is used to capture gestures and use these gestures to control the movement of the player. All the game screens and score board are displayed using an SSD1306 OLED display.

Methodology

The webcam will continually capture frames, it will then be analyzed for the presence of gesture and each gesture is numbered. We consider two gestures and this gesture will essentially move the controllable player gestures to and fro the Y axis.A time is also set when the capturing starts. Using gesture controls a player will be able to catch the randomly generated apples on screen. If a collision is detected between basket and apple a score is incremented. Once the time out reaches 60 seconds a “STOP” control signal is sent and the capture is stopped, displaying the “Game Over” screen and the Score is displayed.

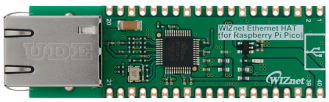
Component Used	Image	Description
Wiznet EVB PICO W5100s		Receives the control signals from the AI and moves the player accordingly.
Breadboard and Jumper wire		Helps with interconnecting circuit elements with the Wiznet EVB PICO W5100S
USB Webcam		Captures the players hand gestures
SSD1306 OLED DISPLAY		Displays the Game Screen and Score Board

Table 1. Components

Block Diagram

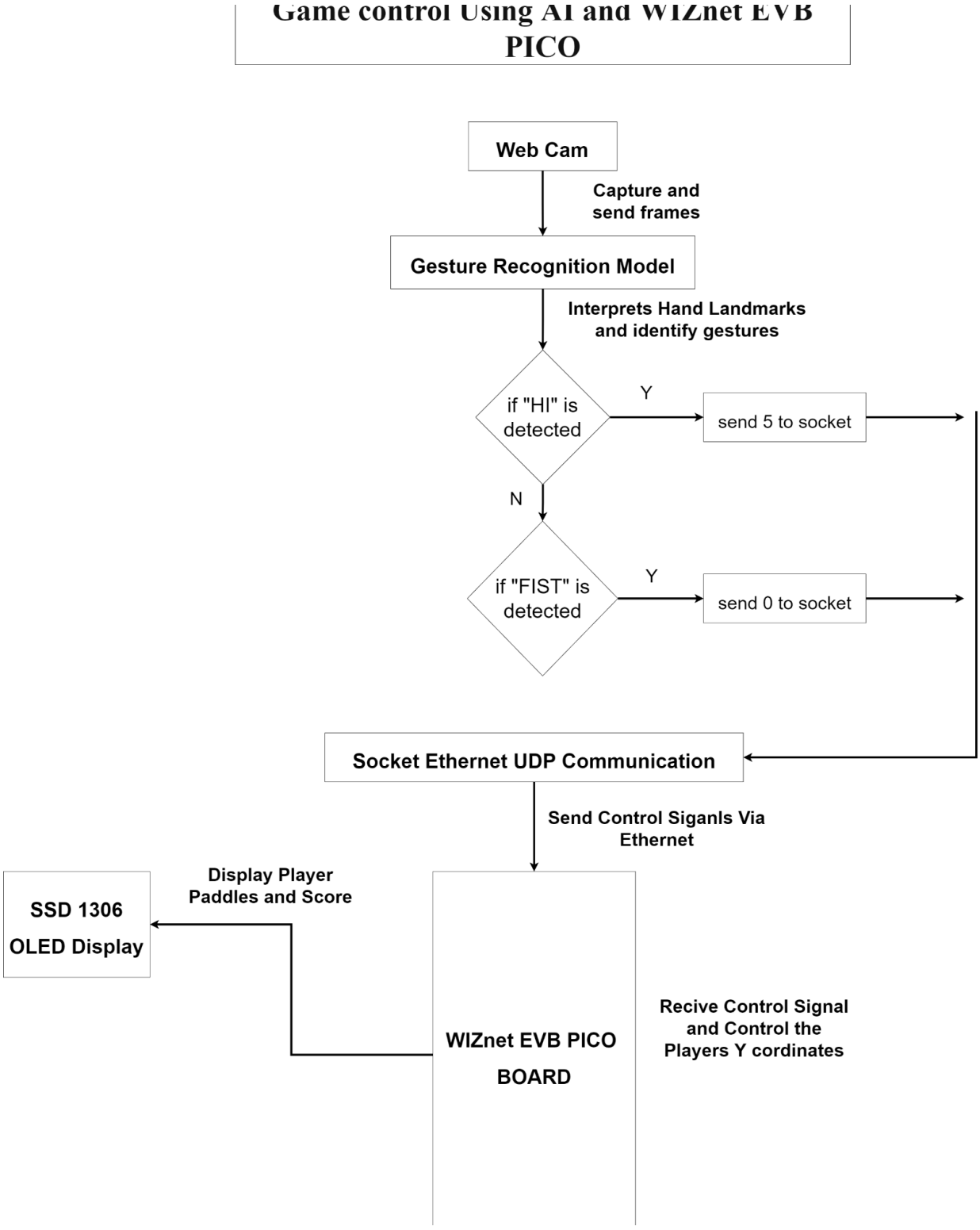


Figure 11. Block Diagram

Block diagram Explanation

The diagram gives an overview of the project, starting with the video capture . Each frame is analyzed for gestures using hand landmarks. If a gesture is detected it is analyzed to see if it's a “HI” gesture or a “FIST” gesture. All the gestures are given a particular number and this number is transmitted via socket connection. If “HI” gesture number 5 is transmitted ,a closed “FIST” gesture number 0 is transmitted. The transmission is done in real time. On receiving the coordinates in the Wiznet EVB PICO W5100s, it checks for the presence of 5 or 0. If 5 is received the player basket is moved downward and if 0 is received downwards. The game is displayed on an OLED SSD1306 display.

Circuit Diagram

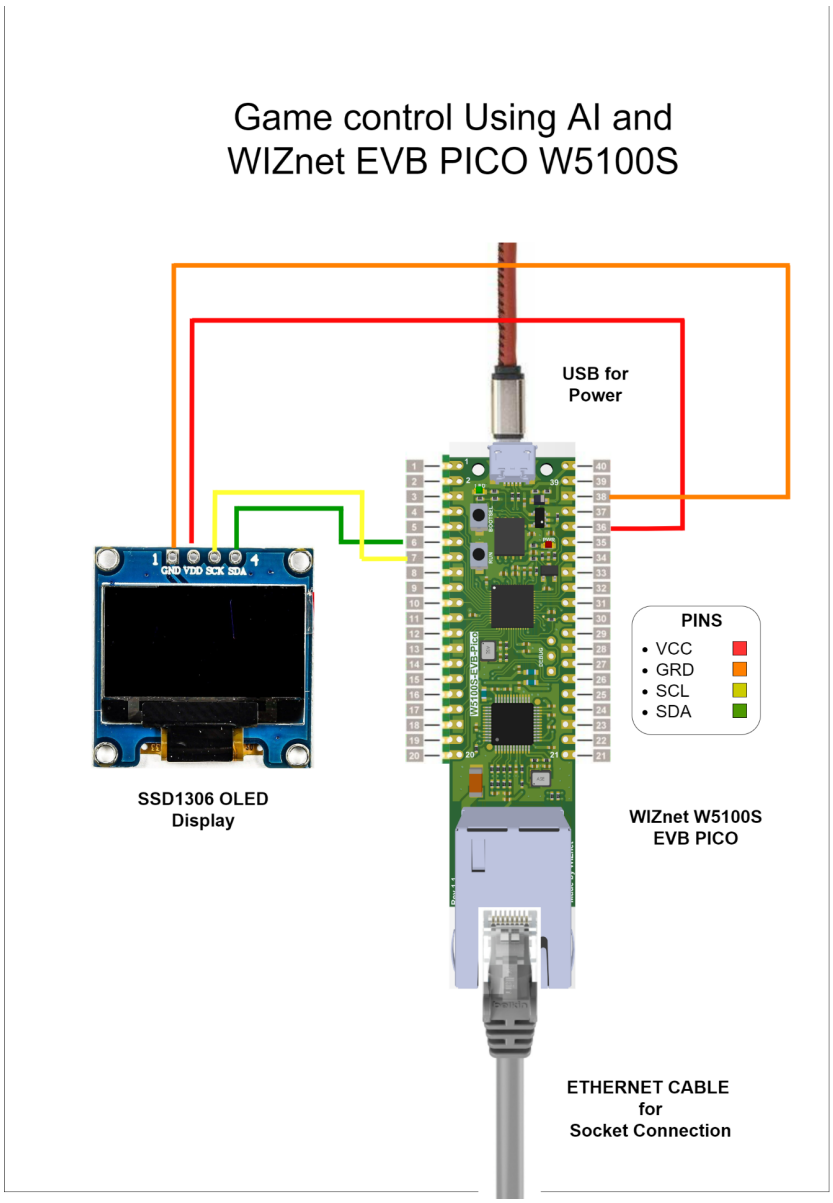


Figure 12. Component Circuit Diagram

Circuit diagram Explanation

VCC , GND[Pin 36, Pin 38] are connected to the VCC and GRD of the SSD1306 OLED Display. The SDA pin of the OLED Display is connected to the GPIO4 pin [Pin 6] which acts as the data pin. The SCL pin of the OLED Display is connected to the GPIO5 pin [Pin 7] which acts as the clock pin. A USB is attached for powering the Wiznet EVB PICO W5100S and the OLED Display. An Ethernet is attached to establish socket Ethernet UDP Communication. The display used is an i2c display which uses 4 pins and will operate in 3v range.SDA pin is used for data and SCL is the clock signal to synchronize the display.

AI Integration

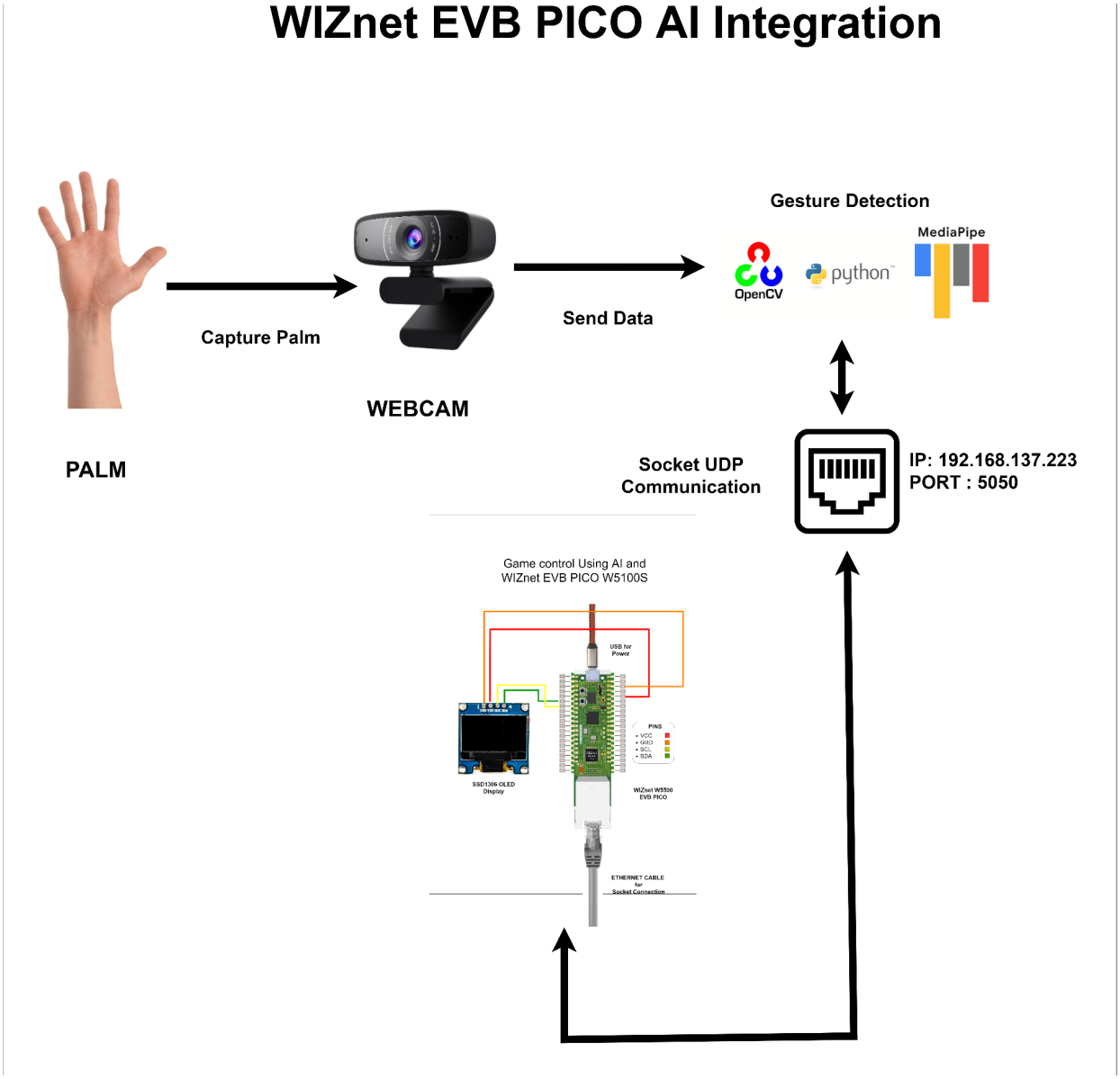


Figure 13. AI Integration in WIZnet EVB PICO W5100s Board

How to integrate AI ?

In this project OpenCV and Mediapipe are used for gesture detection. We use gestures in this project for the player basket movements. The gesture detection code will identify the landmarks of each finger and analyze the position of fingers. The “HI" gesture will move the player basket along the Y axis downward and “CLOSED FIST” will move the player baskets along the Y downwards. The communication between the Wiznet EVB PICO W5100s board and the AI is established by socket UDP connection. On detecting the gestures, each gesture is given a certain number and this number is passed to the socket connection. The game stop control signal “STOP” is also passed through this socket connection.

Python AI Code

```
# -*- coding: utf-8 -*-
"""
@author: sain saji
"""
#importing libraries
import cv2
import os
import time
import handTrackingModule as htm
import sys
from datetime import datetime


# To implements sockets and socket initialization
from socket import *
ip = input("Enter the IP from Serial Monitor")
# '192.168.137.237'
address = (ip, 5000)  # Bind Address and port, Same as Arduino address and port
client_socket = socket(AF_INET, SOCK_DGRAM)  # Set Up the Socket
client_socket.settimeout(0.07)  # only wait 1 second for a resonse


#setting numbers for each gestures
def getNumber(ar):
    s = ""
    for i in ar:
        s += str(ar[i]);

    if(s == "00000"):
        return (0)
    elif(s == "01000"):
        return(1)
    elif(s == "01100"):
        return(2)
    elif(s == "01110"):
        return(3)
    elif(s == "01111"):
```

```
        return(4)
    elif(s == "11111"):
        return(5)
    elif(s == "01001"):
        return(6)
    elif(s == "01011"):
        return(7)


#webcam initialization
wcam, hcam = 640, 480
cap = cv2.VideoCapture(0)
cap.set(3, wcam)
cap.set(4, hcam)
pTime = 0
detector = htm.handDetector(detectionCon=0.75)

#capture current time before starting detection
now = datetime.now()

#start capture and detection
while True:
    # capture time after each frame
    later = datetime.now()
    difference = (later - now).total_seconds()

    #if difference is above 60 seconds stop capture and detection
    if(difference>60):
        data = "STOP"
        client_socket.sendto(data.encode('utf-8'), address)
        break

    #capture cordinates
    success, img = cap.read()
    img = detector.findHands(img, draw=True)
    lmList = detector.findPosition(img, draw=False)
    # print(lmList)
    tipId = [4, 8, 12, 16, 20]
    if(len(lmList) != 0):
```

```
77      fingers = []
78      # thumb
79      if(lmList[tipId[0]][1] > lmList[tipId[0]-1][1]):
80          fingers.append(1)
81      else:
82          fingers.append(0)
83      # 4 fingers
84      for id in range(1, len(tipId)):
85
86          if(lmList[tipId[id]][2] < lmList[tipId[id]-2][2]):
87              fingers.append(1)
88
89          else:
90              fingers.append(0)
91
92      #draw landmarks,text for each gestures
93      cv2.rectangle(img, (20, 255), (170, 425), (0, 255, 0), cv2.FILLED)
94      cv2.putText(img, str(getNumber(fingers)), (45, 375), cv2.FONT_HERSHEY_PLAIN,
95                              10, (255, 0, 0), 20)
96
97      #get the number for gesture
98      print(getNumber(fingers))
99      data = str(getNumber(fingers))  # Set data to Blue Command
100       # send command to arduino
101       client_socket.sendto(data.encode('utf-8'), address)  # send command to arduino
102       try:
103           rec_data, addr = client_socket.recvfrom(2048)  # Read response from arduino
104           print(rec_data)
105       except:
106           pass
107
108
109
110   #to calcualate fps
111   cTime = time.time()
112   fps = 1/(cTime-pTime)
113   pTime = cTime
114
115   #to display fps
```

```
116     cv2.putText(img, f'FPS: {int(fps)}', (400,70),cv2.FONT_HERSHEY_COMPLEX,1,(255,255,0),3)
117     cv2.imshow("image", img)
118     if(cv2.waitKey(1) & 0xFF == ord('q')):
119         break
120
```

Python AI Code Explanation

Line number [5-6] imports all the required libraries for capturing and detecting the gestures.

```
14 # To implements sockets and socket initialization
15 from socket import *
16 ip = input("Enter the IP from Serial Monitor")
17 # '192.168.137.237'
18 address = (ip, 5000)  # Bind Address and port, Same as Arduino address and port
19 client_socket = socket(AF_INET, SOCK_DGRAM)  # Set Up the Socket
20 client_socket.settimeout(0.07)  # only wait 1 second for a resonse
```

Line number [14-20] initializes the binding socket address and ports. This has to be typed manually and has to be typed in the python terminal environment. The port is set as 5000. A timeout of 0.07 seconds has been found optimal for the data transmission speed.

```
23 #setting numbers for each gestures
24 def getNumber(ar):
25     s = ""
26     for i in ar:
27         s += str(ar[i]);
28 
29     if(s == "00000"):
30         return (0)
31     elif(s == "01000"):
32         return(1)
33     elif(s == "01100"):
34         return(2)
35     elif(s == "01110"):
36         return(3)
37     elif(s == "01111"):
38         return(4)
39     elif(s == "11111"):
40         return(5)
41     elif(s == "01001"):
42         return(6)
43     elif(s == "01011"):
44         return(7)
45 
```

In line number [23-45] a function is made to convert the gesture to numbers. Eg: “HI” gesture is given number 5 and closed fist is given 0. This function is called when we have to transmit these numbers as control inputs to the W5100s board.

```
47 #webcam initialization
48 wcam, hcam = 640, 480
49 cap = cv2.VideoCapture(0)
50 cap.set(3, wcam)
51 cap.set(4, hcam)
52 pTime = 0
53 detector = htm.handDetector(detectionCon=0.75)
54
55 #capture current time before starting detection
56 now = datetime.now()
```

Line number [45-56] is to initilaize and setup the internal webcam for capturing frames. These frames will then be exported to the gesture detection module.

```
#draw landmarks,text for each gestures
93        cv2.rectangle(img, (20, 255), (170, 425), (0, 255, 0), cv2.FILLED)
94        cv2.putText(img, str(getNumber(fingers)), (45, 375), cv2.FONT_HERSHEY_PLAIN,
95                                    10, (255, 0, 0), 20)
```

Line number [93-95] sets the landmarks,text for each gestures using the cv2 library

```
#get the number for gesture
98        print(getNumber(fingers))
99        data = str(getNumber(fingers))  # Set data to Blue Command
100         # send command to arduino
101         client_socket.sendto(data.encode('utf-8'), address)  # send command to arduino
102         try:
103             rec_data, addr = client_socket.recvfrom(2048)  # Read response from arduino
104             print(rec_data)
105         except:
106             pass
```

To convert each gesture to corresponding numbers we use the getNumber() function in line number [98] .We use the socket library to send these control numbers to the socket client and the strings are encoded in ‘utf-8’ format. For each transmission the code will check for a response text, if such a text is received it will be printed to the console.

```
56 now = datetime.now()
```

```
57
58 #start capture and detection
59 while True:
60     # capture time after each frame
61     later = datetime.now()
62     difference = (later - now).total_seconds()
```

In line number [56,61] a time capture is enabled. One set before capturing indicating the starting of the game in line number 60 a capturing is enabled to capture time during each frame. A difference is calculated by subtracting two of these time intervals to implement a time out counter.

```
64     #if difference is above 60 seconds stop capture and detection
65     if(difference>60):
66         data = "STOP"
67         client_socket.sendto(data.encode('utf-8'), address)
68         break
```

If the difference is found to be above 60, indicating 1 minute as “STOP” control signal is sent to the Wiznet EVB PICO W5100s via socket, which will put the Wiznet EVB PICO W5100s in an infinite loop which displays the score and game over screen. The break statement will also stop the capture, indicating the end of the game.

Arduino Code

```
1 // Author Sain Saji
2 // BaseCode for SSD1306 128x64 OLED Displays.
3 // GITHUB Link for Complete Tutorial:
4 // https://github.com/sainsaji/SSD1306-Base-Code-for-Raspberry-Pico-Ardunio-Library/edit/main/README.md
5 // Connect Ground PIN of OLED with PICO Ground PIN : PIN: 38
6 // Connect VCC PIN of OLED with the 36th PIN of PICO
7 // Connect SCL [CLOCK] to PIN 7 [GPIO PIN 5]
8 // Connect SDA [DATA] to PIN 6 [GPIO PIN 4]
9
10 //FYI: The sketch won't compile in wowki due to some library issue,Works fine with Arduino IDE
11
12 //Libraries
13 #include <SPI.h>
```

```
14 #include <Wire.h>
15 #include <Adafruit_GFX.h>
16 #include <Adafruit_SSD1306.h>
17 #include <Ethernet.h> //Load Ethernet Library
18 #include <EthernetUdp.h> //Load UDP Library
19 #include "Picopixel.h" //external font library
20
21 //Definitions
22 #define SCREEN_WIDTH 128 // OLED display width, in pixels
23 #define SCREEN_HEIGHT 64 // OLED display height, in pixels
24 #define OLED_RESET     -1 // Make sure this is set to -1 for Pico
25 #define SCREEN_ADDRESS 0x3C //Use 3C with Pico for 128x64 OLED
26 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
27
28 //Ethernet Declarations
29 byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xEE}; //Assign a mac address
30 IPAddress ip(192, 168, 137, 121); //Assign my IP adress
31 unsigned int localPort = 5000; //Assign a Port to talk over
32 char packetBuffer[UDP_TX_PACKET_MAX_SIZE];
33 String datReq; //String for our data
34 int packetSize; //Size of Packet
35 EthernetUDP Udp; //Define UDP Object
36 String localip="192.168.2.1"; //sets a temporary local IP address
37
38 //initialize the control signal
39 int current_input = 10;
40
41 void setup()
42 {
43   //sets and initialize the sockets
44   ethstart();
45   delay(1500); //delay
46
47   //sets and initialize the SSD1306 display
48   display.begin(SSD1306_SWITCHCAPVCC,0x3C);
49   display.fillScreen(0); //0 for filling with black dots. 1 for white
50   display.display();
51
52   //display the IP address to bind on the screen
```

```
53   setip();
54   display.display();
55
56   //sets a delay allowing the user to copy the address to the python console
57   delay(10000);
58
59   //clears the display
60   display.fillScreen(0);
61   display.clearDisplay();
62   display.display();
63 }
64
65
66 void loop()
67 {
68   //starts the game
69   main_menu();
70
71   //clears and updates the display
72   display.display();
73   display.fillScreen(0);
74   display.clearDisplay();
75 }
76
77 //sets the score to zero on start
78 int score = 0;
79
80 //display the game over screen with the score attained
81 int game_over()
82 {
83   while(1)
84   {
85     display.fillScreen(0);
86     display.clearDisplay();
87     display.setCursor(64,32);
88     display.print("Game Over");
89     display.setCursor(80,55);
90     display.print("SCORE:");
91     display.setCursor(115,55);
```

```
92    display.print(score);
93    display.display();
94  }
95 }
96
97 // read the control signals from the python code
98 int read_input()
99 {
100  //initialize the packetSize
101  packetSize = Udp.parsePacket(); //Read theh packetSize
102  if (packetSize > 0)
103  { //Check to see if a request is present
104
105   Udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE); //Reading the data request on the Udp
106   String datReq(packetBuffer); //Convert packetBuffer array to string datReq21
107   current_input = datReq.toInt(); //converts the received control signal to integer value
108
109   //if the recived data is "STOP" the capture
110   if (datReq == "STOP")
111   { //See if Red was requested
112    Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());  //Initialize Packet send
113    Udp.print("GAME OVER"); //Send string back to client
114    Udp.endPacket(); //Packet has been sent
115    //display the end game screen
116    game_over();
117   }
118  }
119  memset(packetBuffer, 0, UDP_TX_PACKET_MAX_SIZE);
120  return 0;
121 }
122
123 //to control the player paddle
124 int y=0;
125 int y_rate=5;
126 int action()
127 {
128  if(current_input==5) //increment if 5 is recived
129  {
130   y+=y_rate;
```

```
131   }
132   if(current_input==0) //decrement 5 if 0 is recived
133   {
134     y-=y_rate;
135   }
136
137   //if player paddle reach max or min y make it stationary
138   if(y>=64-20)
139   {
140     y=64-20;
141   }
142   if(y<=0)
143   {
144     y=0;
145   }
146   return 0;
147 }
148
149 //display the player paddle
150 int move_basket()
151 {
152   display.fillRect(10,y,5,20,WHITE);
153   return 0;
154 }
155
156 //initialize the apple variables.
157 int fire_when=3;
158 int bullet_x = 128-5;
159 int bullet_y = 32;
160 int bullet_rate = 1;
161
162 //fire the apples
163 int enemy_fire()
164 {
165   if(fire_when==3)
166   {
167     bullet_x=bullet_x-bullet_rate;
168     bullet_rate+=1;
169     display.fillRect(bullet_x,bullet_y,3,3,WHITE);
```

```
170    //collition detection
171    if((bullet_y>=y&&bullet_y<=y+20))
172    {
173      if(bullet_x>=0&&bullet_x<=15)
174      {
175      display.setCursor(100,40);
176      display.print("HIT");
177      score+=1;
178      }
179    }
180   }
181   //reset apples and start from a random position
182   if(bullet_x<=5)
183   {
184     bullet_x = 128-5;
185     bullet_y = random(0,63);
186     bullet_rate=1;
187   }
188   return 0;
189 }
190
191 //print the score to the bottom of the screen
192 int printscore()
193 {
194   display.setCursor(80,55);
195   display.print("SCORE:");
196   display.setCursor(115,55);
197   display.print(score);
198   return 0;
199 }
200
201 //starts the game setting player paddle and apples
202 int main_menu()
203 {
204   display.setCursor(80,0);
205   display.setTextSize(1);
206   display.setTextColor(WHITE);
207   read_input();
208   action();
```

```
209   move_basket();
210   enemy_fire();
211   printscore();
212   display.display();
213   return 0;
214 }
216 //initialize the Ethernet Socket UDP connection
218 int ethstart()
219 {
220   Serial.begin(9600); //Turn on Serial Port
221   Ethernet.init(17);
222   Ethernet.begin(mac, ip); //Initialize Ethernet
223   Udp.begin(localPort); //Initialize Udp
224   while (!Serial)
225   {
226     ; // wait for serial port to connect. Needed for native USB port only
227   }
228
229   // start the Ethernet connection:
230   Serial.println("Initialize Ethernet with DHCP:");
231   if (Ethernet.begin(mac) == 0)
232   {
233     Serial.println("Failed to configure Ethernet using DHCP");
234     if (Ethernet.hardwareStatus() == EthernetNoHardware)
235     {
236       Serial.println("Ethernet shield was not found.  Sorry, can't run without hardware. :(");
237     } else if (Ethernet.linkStatus() == LinkOFF)
238     {
239       Serial.println("Ethernet cable is not connected.");
240     }
241     // no point in carrying on, so do nothing forevermore:
242     while (true)
243     {
244       delay(1);
245     }
246   }
247   // print your local IP address:
248   Serial.print("Type this IP to Python: ");
249   Serial.println(Ethernet.localIP());
```

```
250   localip = String(Ethernet.localIP());
251   return 0;
252 }
253
254 //converts the address to printable string
255 String DisplayAddress(IPAddress address)
256   {
257  return String(address[0]) + "." +
258         String(address[1]) + "." +
259         String(address[2]) + "." +
260         String(address[3]);
261   }
262
263 // display the IP address of the pico board
264 int setip()
265 {
266   display.setCursor(0,0);
267   display.setTextSize(1);
268   display.setTextColor(WHITE);
269   display.println("IP Address is:");
270   writeip(DisplayAddress(Ethernet.localIP()));
271   writeip(localip);
272   return 0;
273 }
274
275 //display the IP address to the screen
276 int writeip(String ipaddress)
277 {
278   display.setCursor(0,10);
279   display.setTextSize(1.5);
280   display.setTextColor(WHITE);
281   display.println(ipaddress);
282   display.setCursor(0,20);
283   display.setTextSize(1.5);
284   display.println("Set this IP to Python");
285   display.println("The Screen will exit in 5 Seconds:");
286   return 0;
287 }
288
```

Arduino Code Explanation

All the libraries and variables initialize in line no [13-36]. This initializes the ethernet libraries , the SSD1306 display and the socket binding address and the port. Line number [41-63]. Start the ethernet connection and initialize the screen with zeros. The ethstart() function enables the ethernet port and sets the IP address variable. This will be available for 10000 seconds / 10 seconds . This IP address has to be copied and typed to the python console.

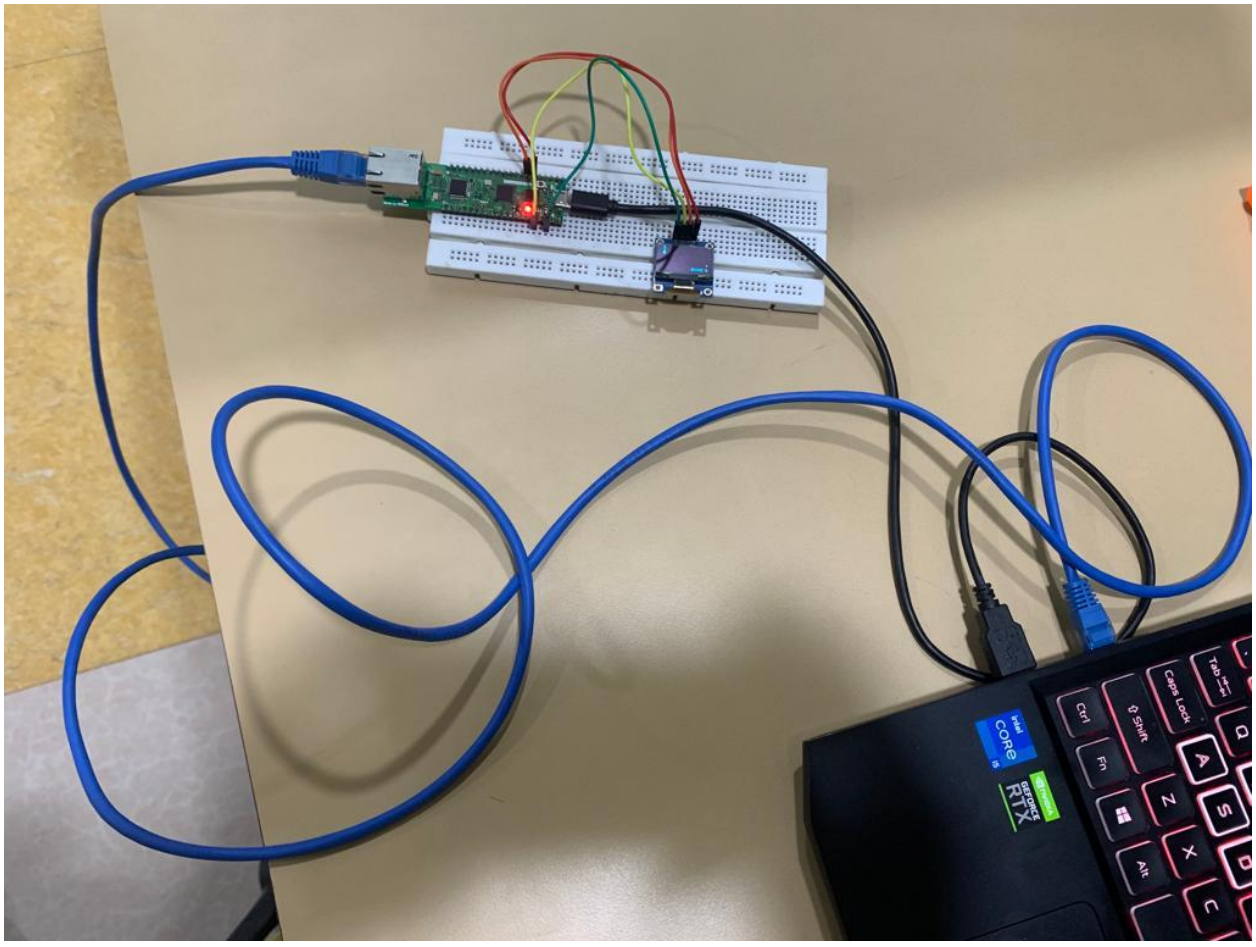
Void loop starts at line number [66-75] which repeatedly calls the main_menu() function and updates the display continuously presenting the changes to the screen. The main_menu() function line number [202-214] which has 5 functions. read_input() function line number [98-121] will read the control signals via signals and set the current_input variable. After the read_input() function call action() function is called. The will sets the player's baskets Y coordinate essentially moving it up and down to catch the incoming apples. The movement is done in real time. move_basket function will display the paddles as per the coordinates. enemy_fire() function line number [163] will fire the apples randomly from a y coordinate. Speed of the apples is increased as it moves along the X coordinate. When a collation happens when the baskets X and the apples X coordinate happens , a score is added and will also display a "HIT" message on the screen. Once it reaches the set X coordinate. The bullet will reset its position to a random Y coordinate. In line number [110] a control signal condition is enabled for the Game Over screen. This is done by receiving the "STOP" signal after 1 minute of gameplay. This will trigger the activation of the game_over() function line number [81]. Which will print a "Game Over" screen along with the score.

The Gameplay

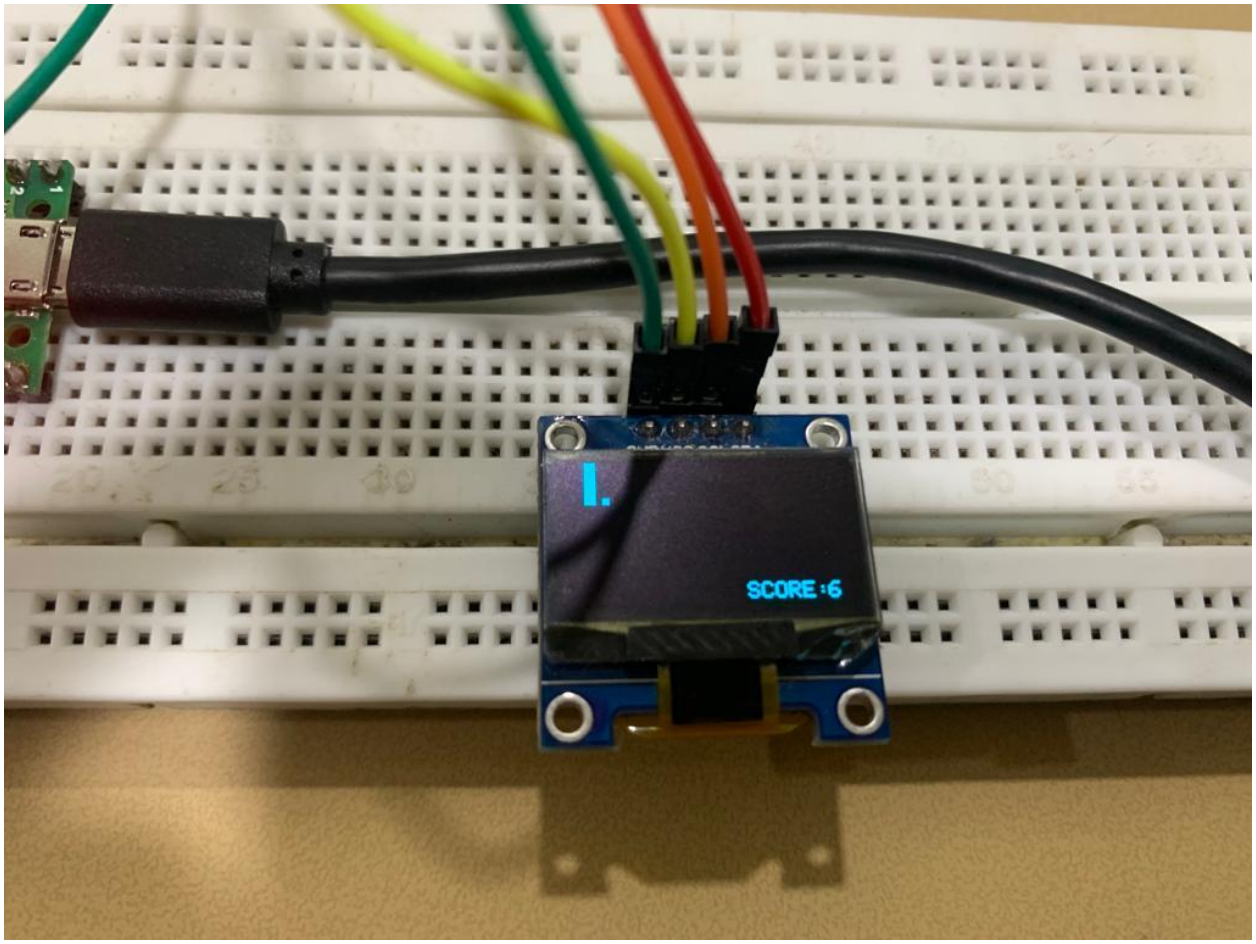
The game is a single player game in which the player has to capture as many apples as they can in under 60 seconds. The apples are randomly generated anywhere on the screen and will move fast every millisecond. The player has to move the basket by using gestures. Each capture of apples will reward 1 score. The basket can be moved only up or down. The player can use the "CLOSED FIST" Gesture to move the basket down and "HI" gesture to move the basket up. By

the end of 60 seconds the capture will stop and a game over screen will appear projecting the score

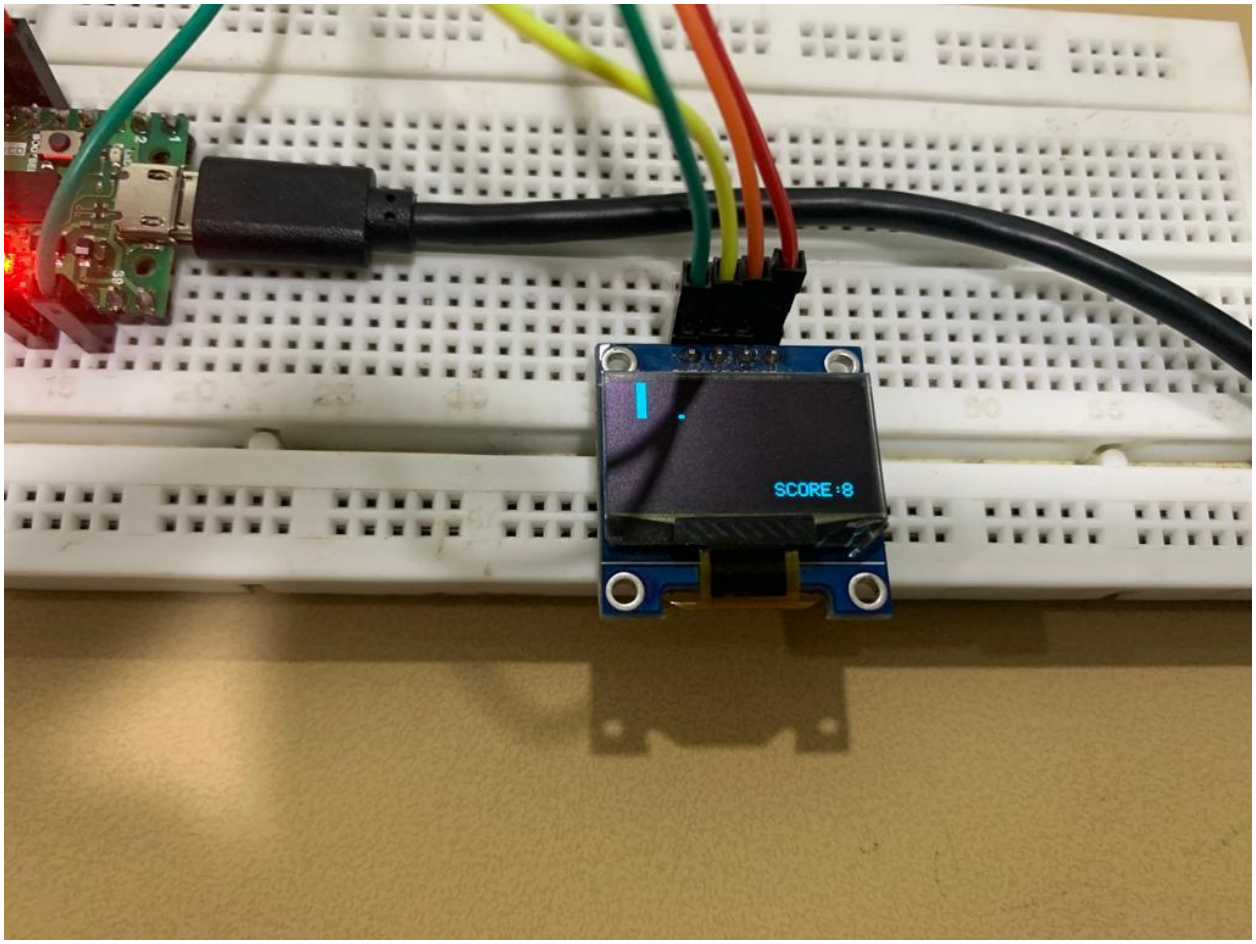
Images



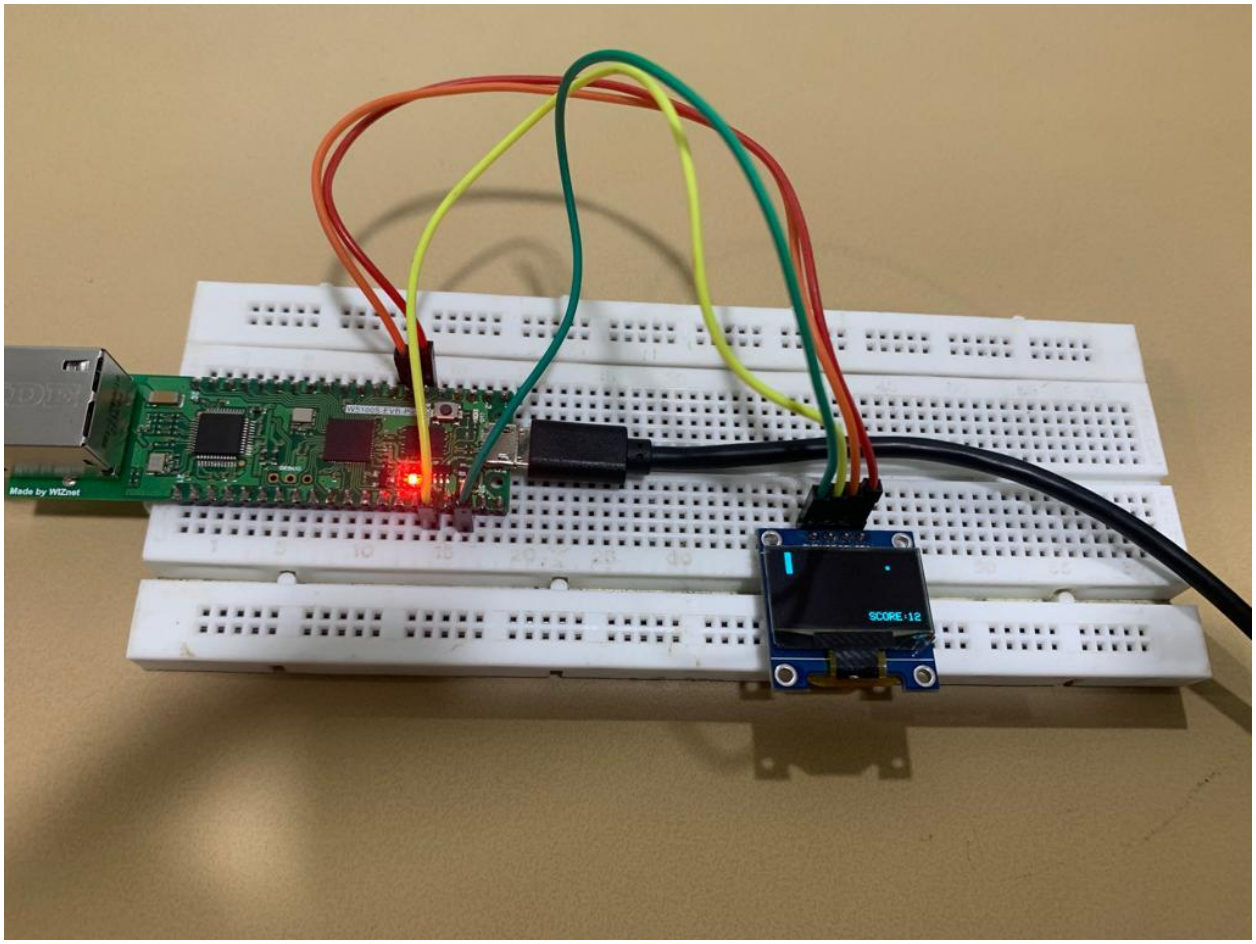
Project setup including Ethernet cable connected directly to the laptop and EVB PICO W5100S . The SSD1306 is displaying the game . The whole setup is powered by a USB cable.



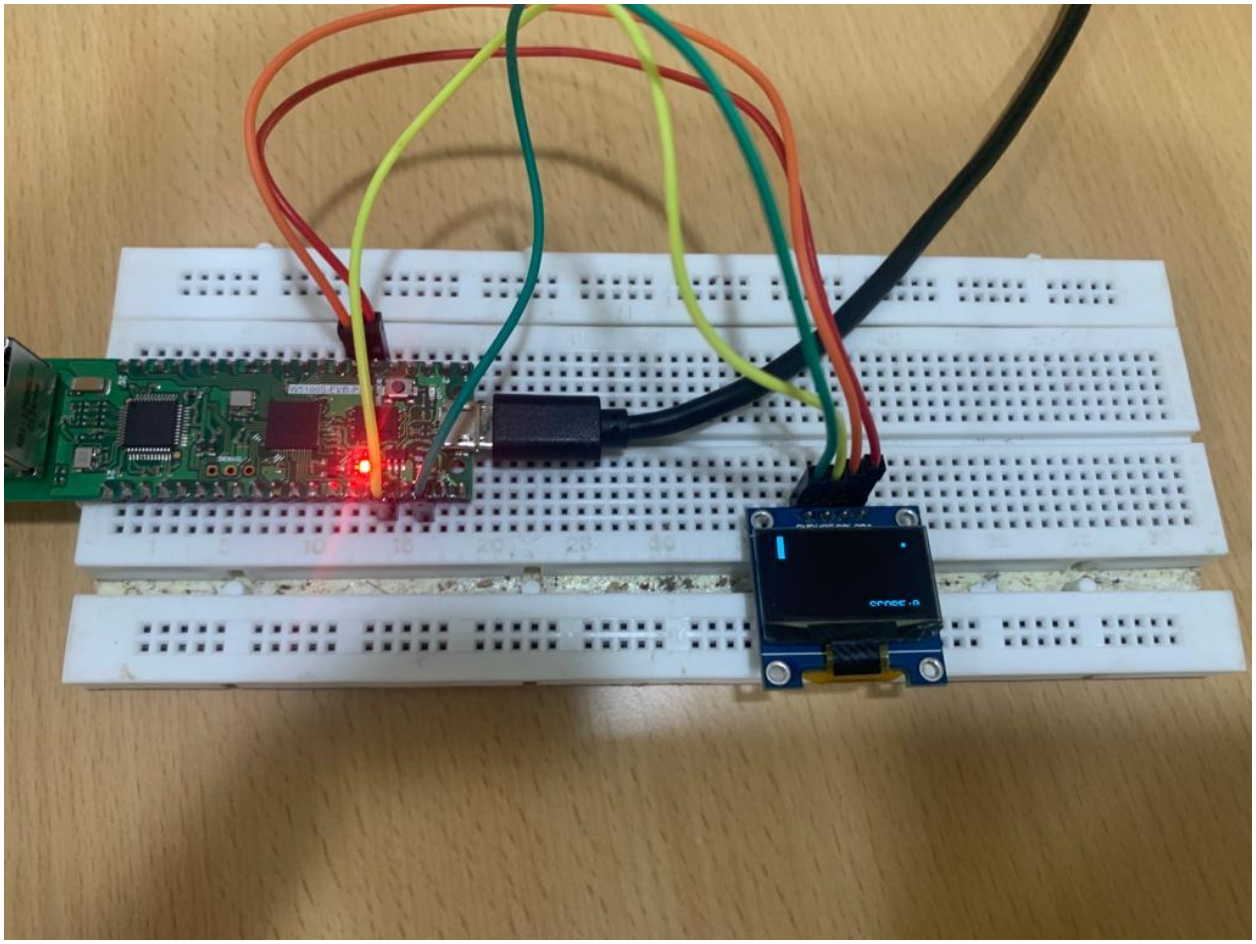
SSD1306 Displaying the game with player basket and apples along with current score before collition



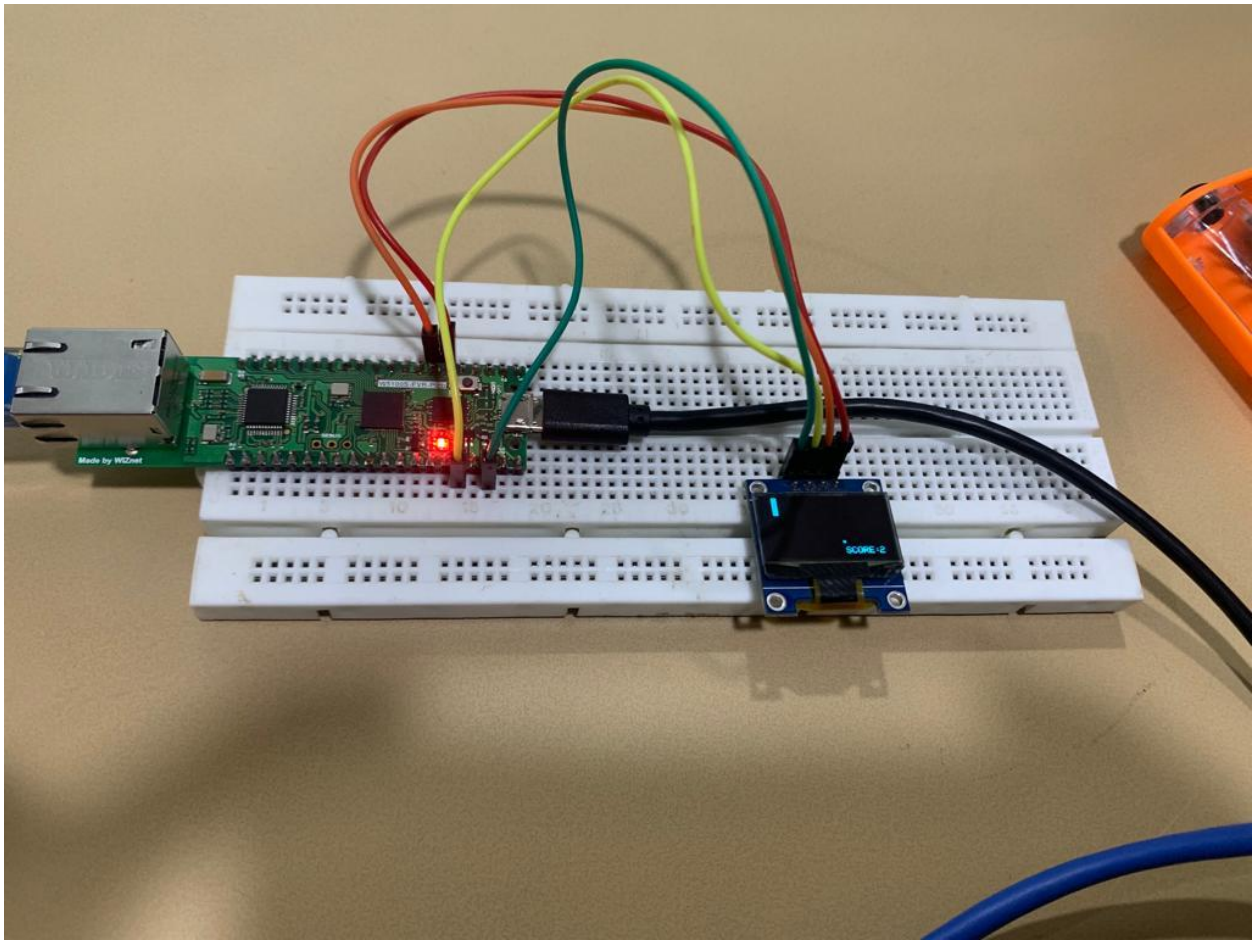
SSD1306 Displaying the game with player basket and apples along with current score, collision will increase the score



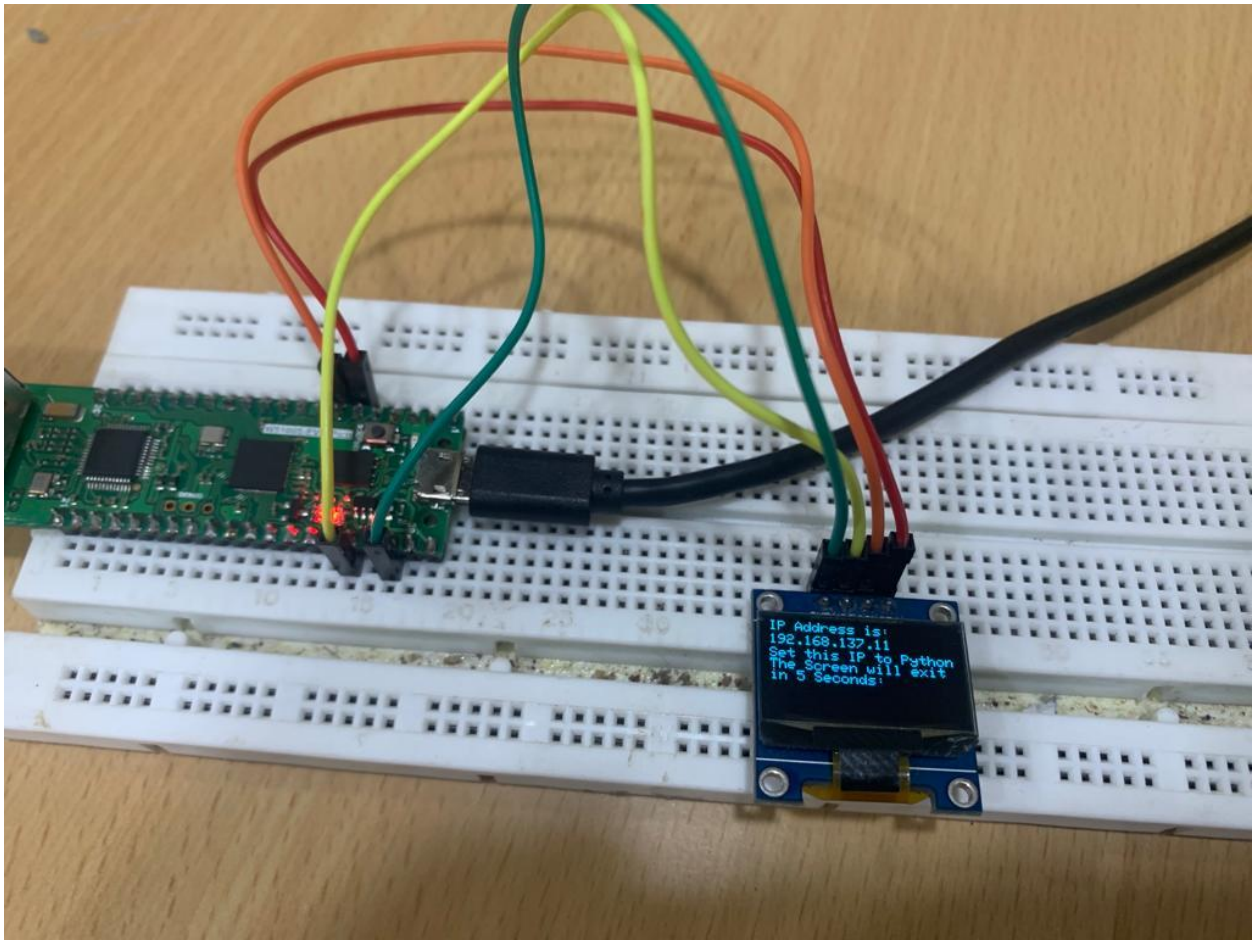
SSD1306 Displaying the game with player basket and apples along with current score



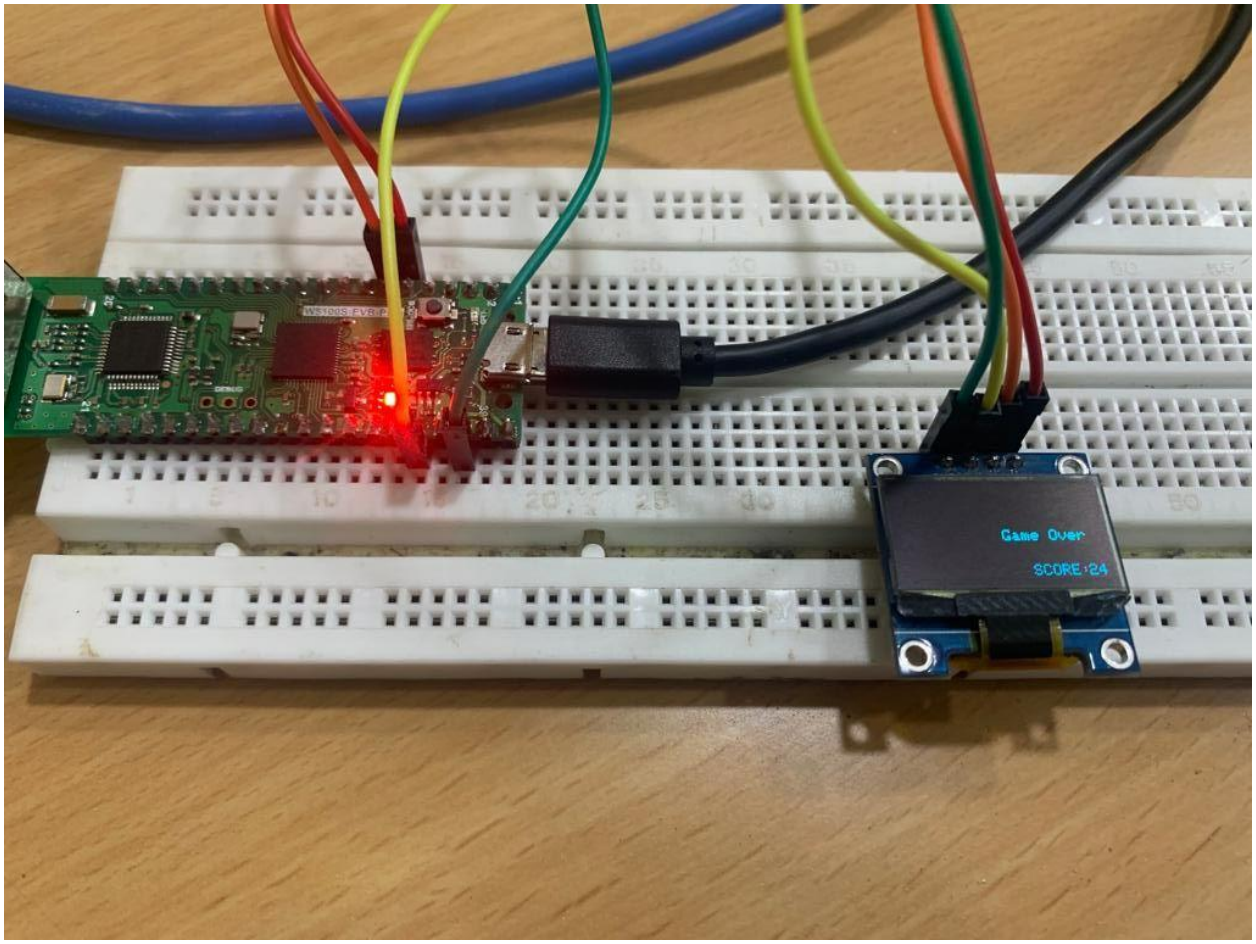
SSD1306 Displaying the game with player basket and apples along with current score



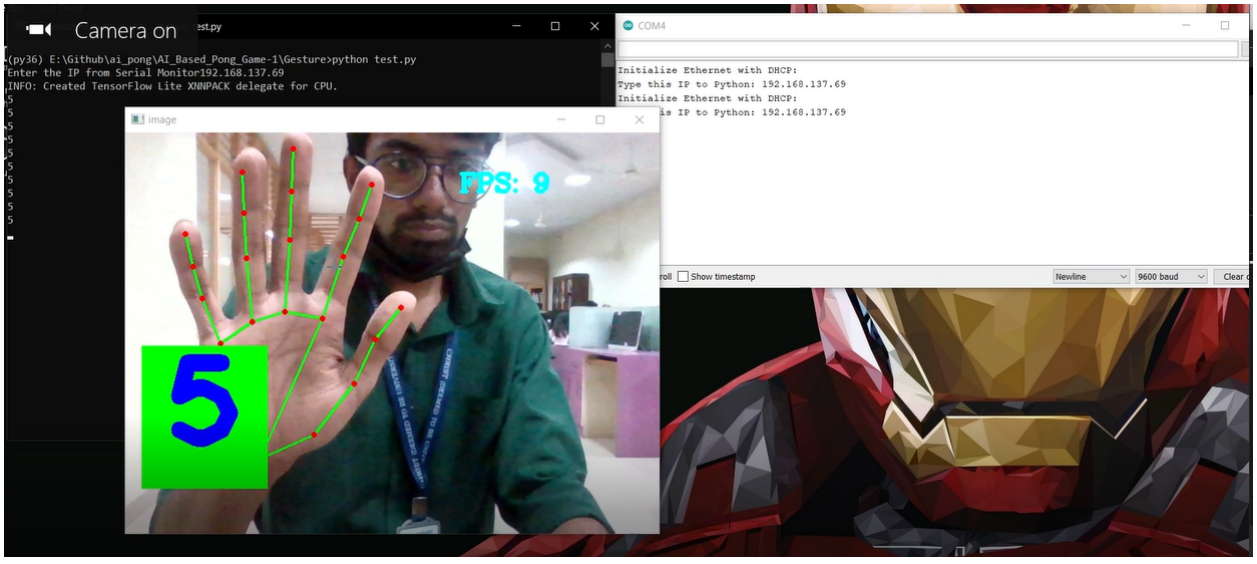
SSD1306 Displaying the game with player basket and apples along with current score



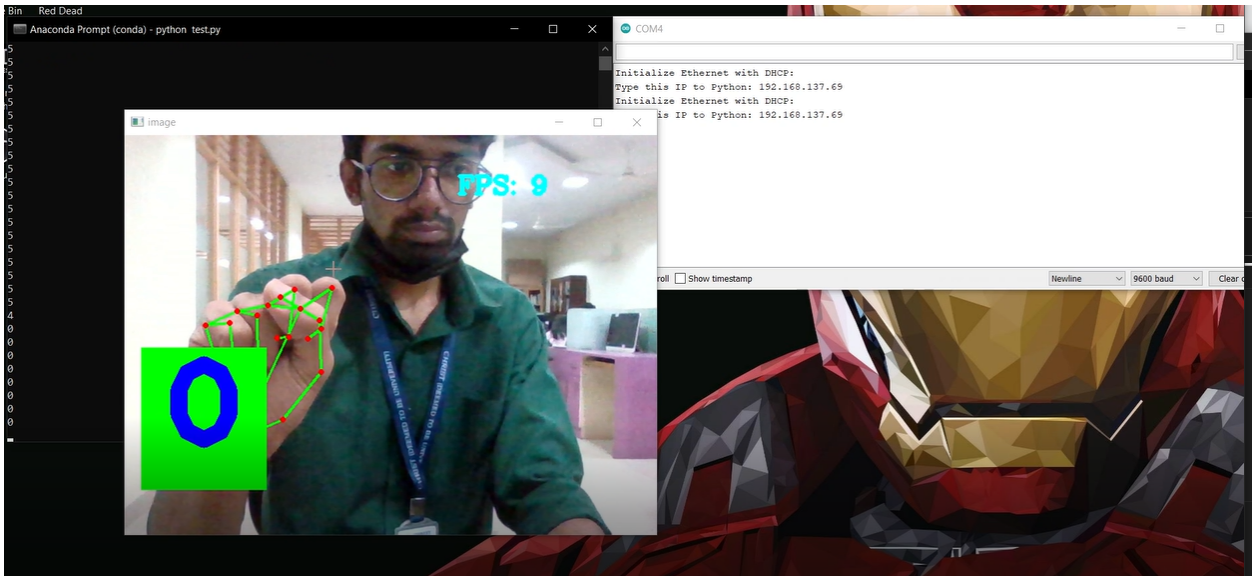
Displaying the EVB PICO W5100S's IP Address which has to be copied to the python console



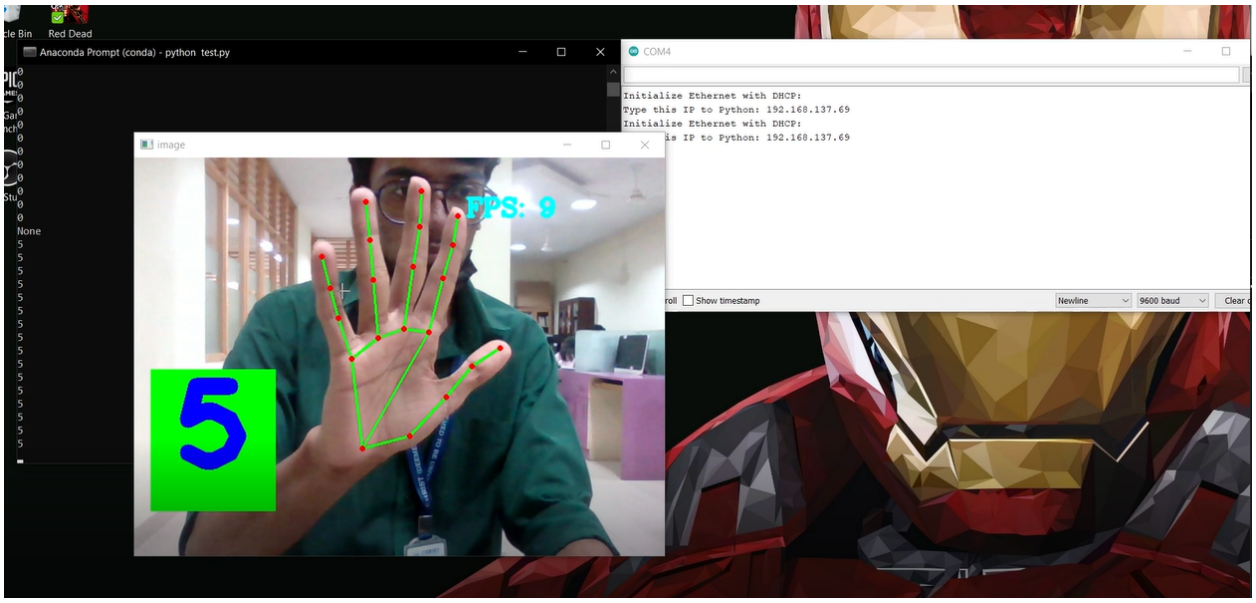
Game Over Screen , displaying the score



Mediapipe hand landmarks identifying the gesture and converting to numbers “HI” gesture will move up the basket



Mediapipe hand landmarks identifying the gesture and converting to numbers "CLOSED FIST" gesture will move up the basket



Mediapipe hand landmarks identifying the gesture and converting to numbers "HI" gesture will move up the basket

Video Link

- https://drive.google.com/drive/folders/1i3i7I2wckqI15laX3dgWvoGH2iArrEdb?usp=sharing

References

- Arduino I2C SSD1306 OLED screen tutorial - The EECS Blog
- Arduino with Python LESSON 17: Transferring Data over Ethernet UDP - YouTube