MOTION DETECTION GAME

Objective:

We will develop a **Motion Detection** project based on Squid Game using **ESP32 CAM** & **OpenCV**. With the help of the **Python program** and ESP32 Camera Module, we will develop a **Red Light – Green Light Game**. Here, we will capture the frames of the person moving using ESP32-Cam. If there is any motion detected in the live video stream when the **red light** is turned on then the person is **dead** or game over else **green light** is shown in which the person needs to **move**.

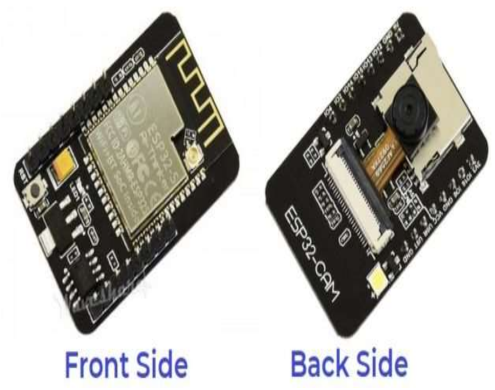
Red Light = No motion & Green Light = Motion

For this we should have a knowledge of **Python**, Image processing, Embedded Systems, and IoT. In this project, we will understand how to detect the motion of a person, and what all requirements are needed to run the python program. **motion detection program** is implemented with the **ESP32 CAM**.

Equipments:

1)ESP 32 CAM AI THINKER MODEL:

The ESP32 Based Camera Module developed by **AI-Thinker**. The controller is based on a **32-bit CPU** & has a combined **Wi-Fi** + **Bluetooth/BLE** Chip. It has a built-in **520 KB SRAM** with an external **4M PSRAM**. Its GPIO Pins have support like **UART, SPI, I2C, PWM, ADC,** and **DAC**.



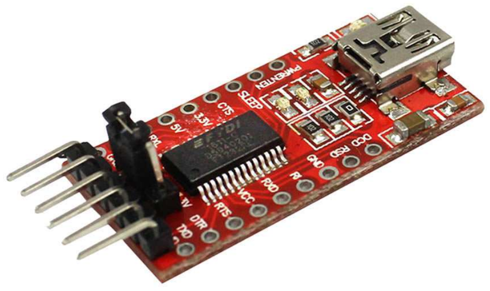
The module combines with the **OV2640 Camera Module** which has the highest Camera Resolution up to **1600 × 1200**. The camera connects to the ESP32 CAM Board using a 24 pins gold plated connector. The board supports an **SD Card** of up to **4GB**. The SD Card stores capture images.

2)OV2640 camera: The board doesn't have a

programmer chip.

3). **FTDI Programmer:**

It is used for uploading the code from Arduino IDE to the ESP32 CAM.



4)Cable:

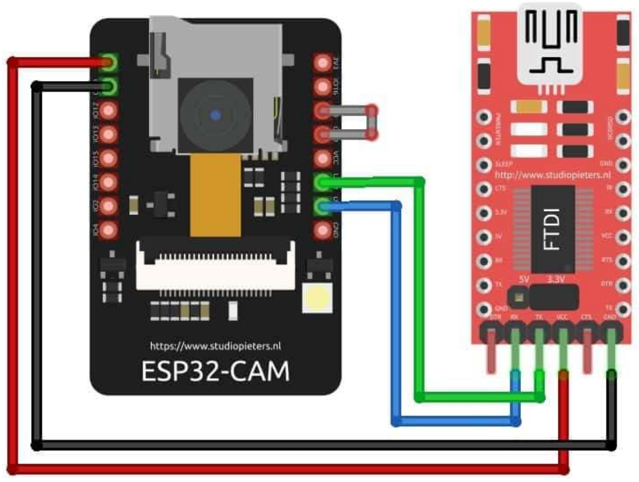
With the help of this we can connect the usb of system to ftdi Programmer

5)Jumper Wire:

It is used to connect the FTDI programmer to the ESP32 CAM.

Connection of ESP32 TO FTDI Programmer:

So In order to program this board, you can use any type of **USB-to-TTL Module**. There are so many **FTDI Module** available based on [CP2102] or [CP2104] Chip or any other chip. Make a following **connection between FTDI Module and ESP32 CAM** module.



.

ESP32-CAM	**FTDI Programmer**
GND	GND
5V	VCC
U0R	TX
U0T	RX
GPIO0	GND

Connect the **5V** & **GND** Pin of ESP32 to 5V & GND of FTDI Module. Similarly, connect the **Rx** to **UOT** and **Tx** to **UOR** Pin. And the most important thing, you need to short the **IO0** and **GND** Pin together. This is to put the device in **programming mode**. Once programming is done you can remove it.

Installing ESP32CAM Library:

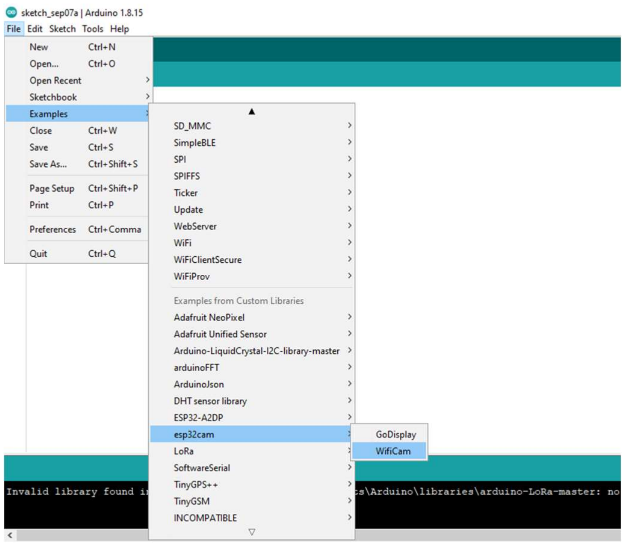
Here we will not use the general **ESP webserver example** rather another streaming process. Therefore we need to add another ESPCAM library. The esp32cam library provides an object oriented API to use **OV2640 camera** on **ESP32 microcontroller**. It is a wrapper of **esp32-camera library**.

Once downloaded add this zip library to Arduino Libray Folder. To do so follow the following steps:

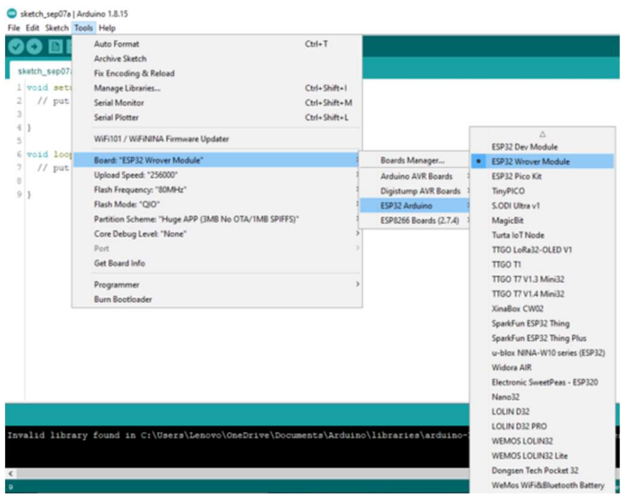
*Open **Arduino** -> **Sketch** -> **Include Library** -> **Add .ZIP Library**-> Navigate to downloaded zip file -> **add***

Source Code/Program for ESP32 CAM Module:

The source code/program ESP32 CAM Motion Detection can found in Library Example. So go to *Files -> Examples -> esp32cam -> WifiCam*.



```
Invalid library found i                              s\Arduino\libraries\arduino-LoRa-master: no
```



```
Invalid library found in C:\Users\Lenovo\OneDrive\Documents\Arduino\libraries\arduino-
```

Before Uploading the code you have to make a small change to the code. Change the **SSID** and **password** variable and in accordance with the WiFi network.

Now **compile** and **upload** it to the ESP32 CAM Board. But during uploading, you have to follow few steps every time.

- Make sure the **IO0 pin** is shorted with the ground when you have pressed the upload button.
- If you see the dots and dashes while uploading press the **reset button** immediately
- Once the code is uploaded, remove the I01 pin shorting with Ground and press the reset button once again.
- If the output is the **Serial monitor** is still not there then press the reset button again.

So setting up ESP32-CAM part is done now. The ESP32-CAM is transmitting the live video, make sure that you copy this IP address displayed.

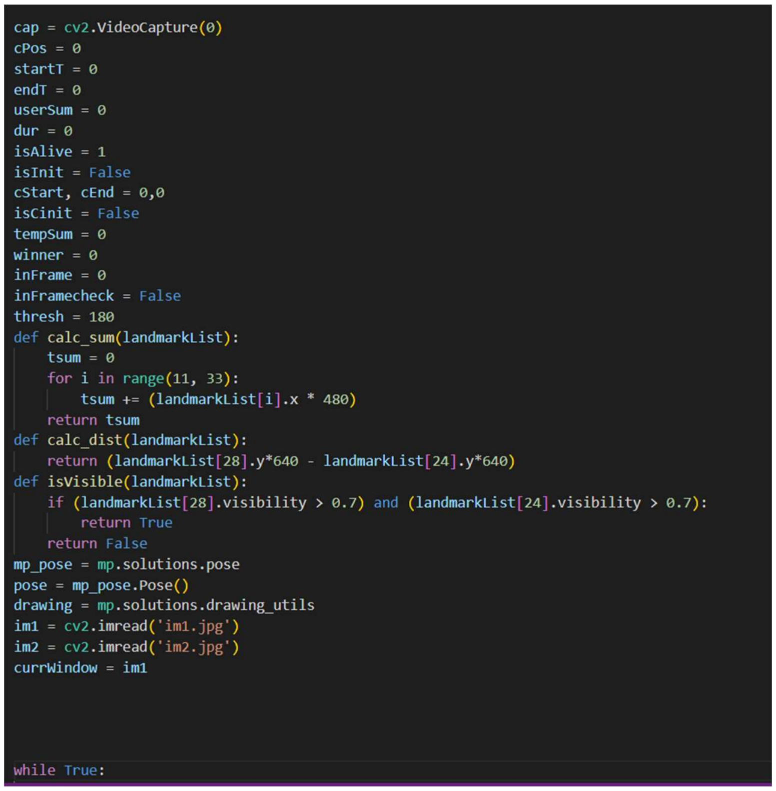
Python code for Motion Detection ESP32 CAM:

Installing Python & Required Libraries

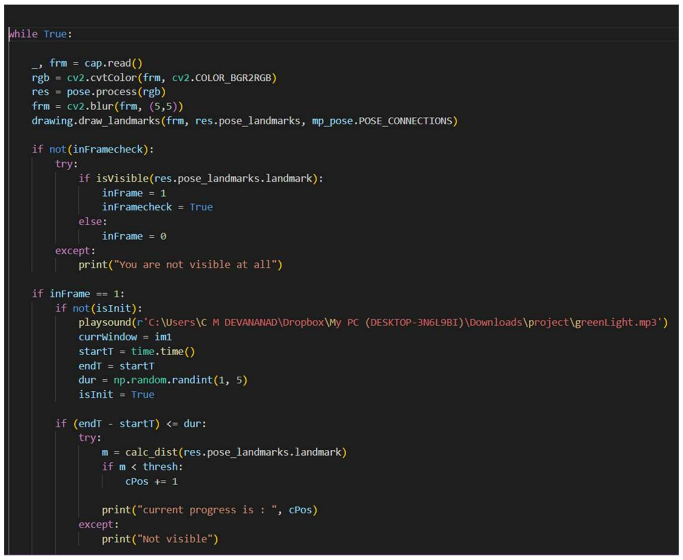
For the live stream of video to be visible on our computer we need to write a **Python script** that will enable us to retrieve the frames of the video. The first step is to **install Python**.

Once downloaded and installed open to the command prompt. Now we have to install a few **libraries**. For that run the following commands below one after another until all the libraries are installed.

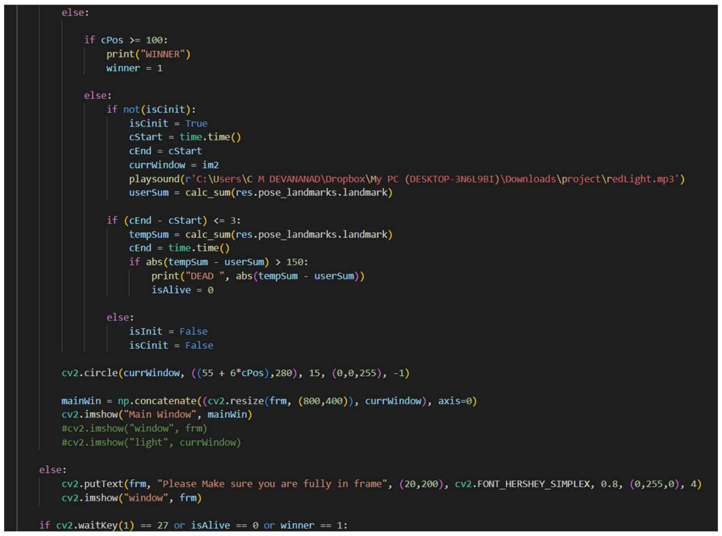
```
pip install numpy
pip install opencv−python
pip install mediapipe
pip install playsound==1.2.2
```



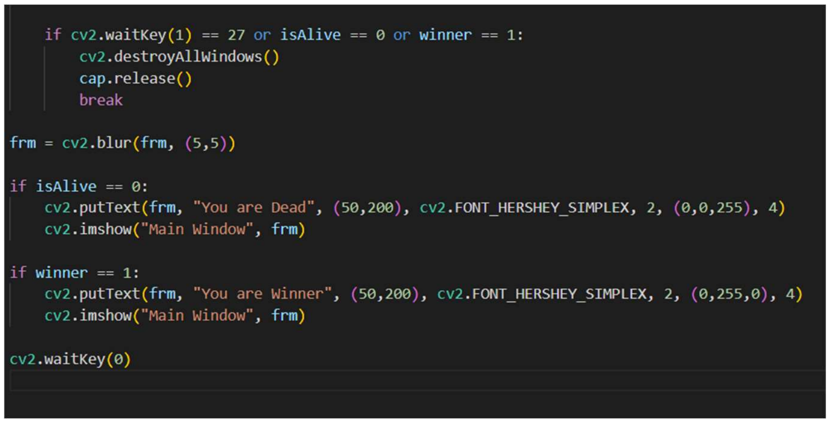
```
cap = cv2.VideoCapture(0)
cPos = 0
startT = 0
endT = 0
userSum = 0
dur = 0
isAlive = 1
isInit = False
cStart, cEnd = 0,0
isCinit = False
tempSum = 0
winner = 0
inFrame = 0
inFramecheck = False
thresh = 180
def calc_sum(landmarkList):
    tsum = 0
    for i in range(11, 33):
        tsum += (landmarkList[i].x * 480)
    return tsum
def calc_dist(landmarkList):
    return (landmarkList[28].y*640 - landmarkList[24].y*640)
def isVisible(landmarkList):
    if (landmarkList[28].visibility > 0.7) and (landmarkList[24].visibility > 0.7):
        return True
    return False
mp_pose = mp.solutions.pose
pose = mp_pose.Pose()
drawing = mp.solutions.drawing_utils
im1 = cv2.imread('im1.jpg')
im2 = cv2.imread('im2.jpg')
currWindow = im1



while True:
```



```python
while True:

    _, frm = cap.read()
    rgb = cv2.cvtColor(frm, cv2.COLOR_BGR2RGB)
    res = pose.process(rgb)
    frm = cv2.blur(frm, (5,5))
    drawing.draw_landmarks(frm, res.pose_landmarks, mp_pose.POSE_CONNECTIONS)

    if not(inFramecheck):
        try:
            if isVisible(res.pose_landmarks.landmark):
                inFrame = 1
                inFramecheck = True
            else:
                inFrame = 0
        except:
            print("You are not visible at all")

    if inFrame == 1:
        if not(isInit):
            playsound(r'C:\Users\C M DEVANANAD\Dropbox\My PC (DESKTOP-3N6L9BI)\Downloads\project\greenLight.mp3')
            currWindow = im1
            startT = time.time()
            endT = startT
            dur = np.random.randint(1, 5)
            isInit = True

        if (endT - startT) <= dur:
            try:
                m = calc_dist(res.pose_landmarks.landmark)
                if m < thresh:
                    cPos += 1

                print("current progress is : ", cPos)
            except:
                print("Not visible")
```

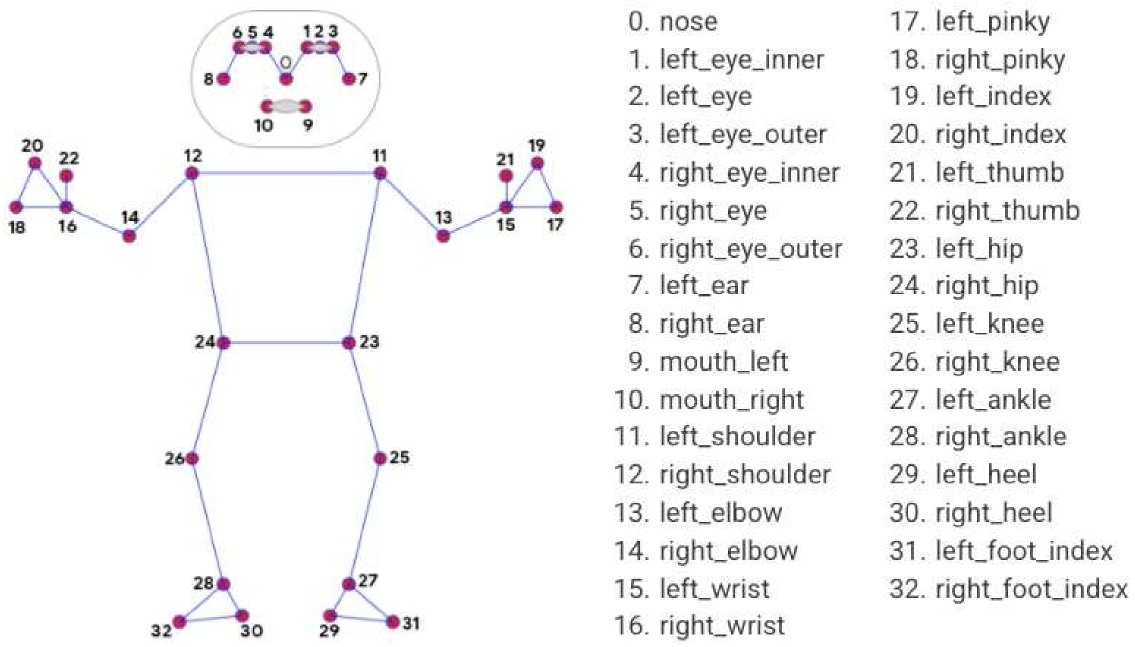


```python
        else:

            if cPos >= 100:
                print("WINNER")
                winner = 1

            else:
                if not(isCinit):
                    isCinit = True
                    cStart = time.time()
                    cEnd = cStart
                    currWindow = im2
                    playsound(r'C:\Users\C M DEVANANAD\Dropbox\My PC (DESKTOP-3N6L9BI)\Downloads\project\redLight.mp3')
                    userSum = calc_sum(res.pose_landmarks.landmark)

                if (cEnd - cStart) <= 3:
                    tempSum = calc_sum(res.pose_landmarks.landmark)
                    cEnd = time.time()
                    if abs(tempSum - userSum) > 150:
                        print("DEAD ", abs(tempSum - userSum))
                        isAlive = 0

                else:
                    isInit = False
                    isCinit = False

        cv2.circle(currWindow, ((55 + 6*cPos),280), 15, (0,0,255), -1)

        mainWin = np.concatenate((cv2.resize(frm, (800,400)), currWindow), axis=0)
        cv2.imshow("Main Window", mainWin)
        #cv2.imshow("window", frm)
        #cv2.imshow("light", currWindow)

    else:
        cv2.putText(frm, "Please Make sure you are fully in frame", (20,200), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,255,0), 4)
        cv2.imshow("window", frm)

    if cv2.waitKey(1) == 27 or isAlive == 0 or winner == 1:
```



```python
    if cv2.waitKey(1) == 27 or isAlive == 0 or winner == 1:
        cv2.destroyAllWindows()
        cap.release()
        break

frm = cv2.blur(frm, (5,5))

if isAlive == 0:
    cv2.putText(frm, "You are Dead", (50,200), cv2.FONT_HERSHEY_SIMPLEX, 2, (0,0,255), 4)
    cv2.imshow("Main Window", frm)

if winner == 1:
    cv2.putText(frm, "You are Winner", (50,200), cv2.FONT_HERSHEY_SIMPLEX, 2, (0,255,0), 4)
    cv2.imshow("Main Window", frm)

cv2.waitKey(0)
```

Motion Detection Algorithm & Testing:
Now to detect motion we are going to use mediapipe library using this we are able to find landmarks of a person standing in front of the camera as in the image below.



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

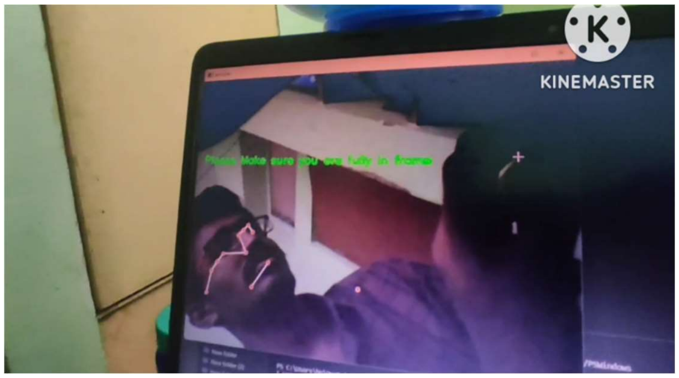
Here to detect motion we are going to use landmark no's 24,23,28, and 27. Because if the person moves then the distance between his hip and alternate foot increases or decreases. For example, if the person moves then the distance between 23 and 28 or 24 and 27 surely changes. Thus, the motion is detected.

Now after connecting your ESP32-Cam module and laptop where we are having our code, with the same local wifi. We run the python code . Thus, our ESP32-Cam-based game is ready.

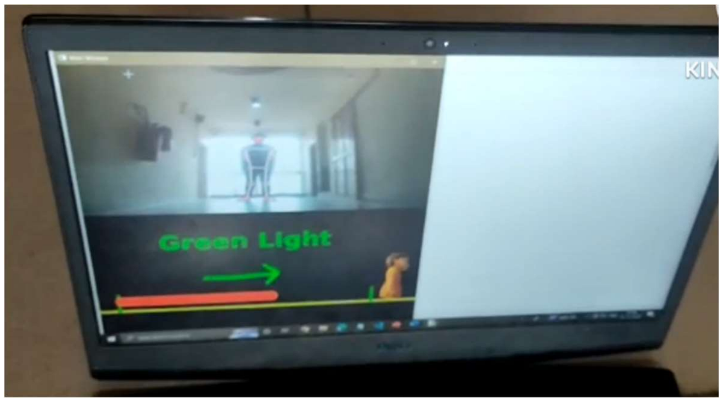
the frames of the person moving are captured using ESP32-CAM and if there is any motion detected in the live video stream when the red light is turned on then the person is dead or game over else green light is shown in which the person needs to move.In this way we can build a Motion Detection-based Squid Game using ESP32 CAM & OpenCV can be developed.

Results of the Project:

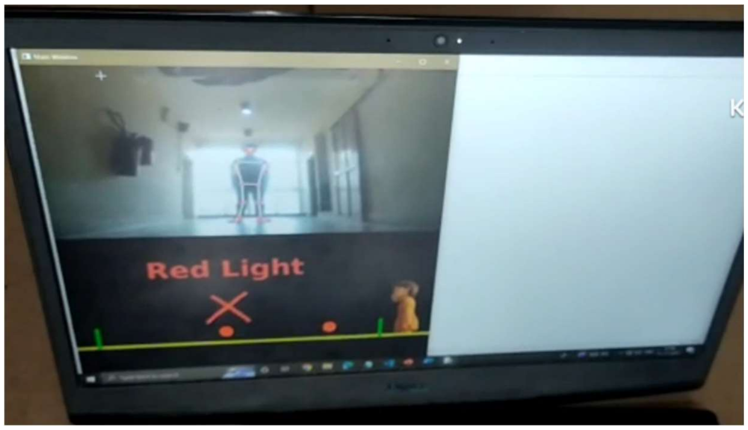
If the whole body of the person is not visible then we get that whole body should visible.



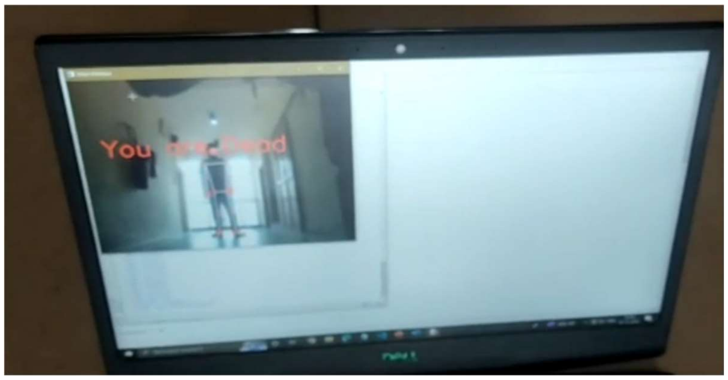
And when the whole body is visible then game will start and from that when green light is shown we can move or stay in the same position.

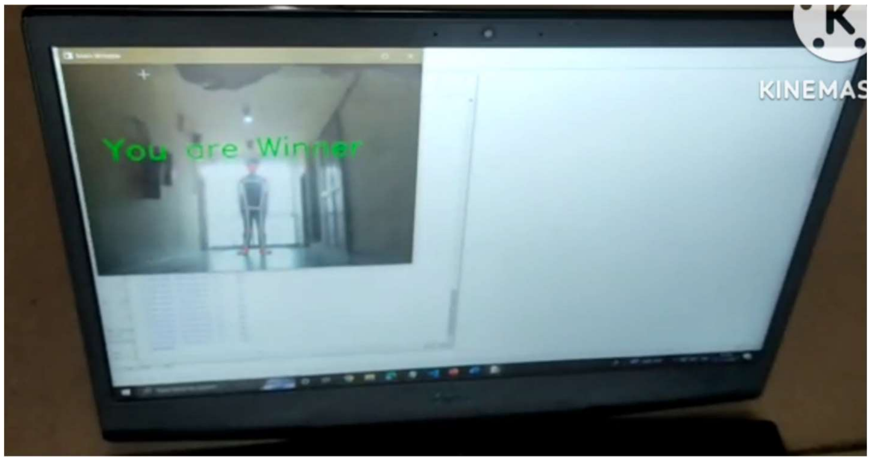


When the red light is shown we should not move from the place.



When we move at the time of red light then we are out of the game and it shows **you are dead.**





If we are alive for after 4 rounds then it will show **you are winner**

These are the results of my demo and the projects.

TEAM MEMBERS:

1. C M DEVANAND 2001CS19

2. B GNANESHWAR 2001CS15