Smart Office Automation System with WIZnet Board

Project Description

In this project, we'll create a Smart Office Automation System that can detect human presence inside a room and the electrical components inside the room can be turned OFF and ON. All the detection, processing, and coordinate mapping is done by opencv and python. The bulbs are connected to the Pico board. The Picoboard receives the status of the room; this response is used by the Picoboard to turn off and on the lights inside the room.

Things used in this project

1 Hardware

- Wiznet Pico board
- Jumper wires
- 1-Channel Relay
- Bread Board
- Web Cam
- Bulb
- Electric Wires

2 Software

- Arduino IDE
- Python
- Libraries Included in Python : cv2, numpy, serial, time

Story

There is a huge wastage of energy in our daily life, which can be saved if we turn off the electronic devices when not in use. Unfortunately we humans are so busy that we forget to turn them off. So to solve this problem, we thought of Automating the electric bulbs inside an office room.

In this project we will make our existing office room/cabin/classroom a smart one. When no one is there, the lights inside the room will be OFF. And when a person is inside the room the light bulbs remain turned ON. The light bulbs are turned ON automatically when the door is opened.

Methodology

The ultrasonic sensor is placed near the door to detect the motion of the door indicating that a person has entered into the room. Also the picoboard sends a message to the 1-channel relay that is turned ON to switch ON the bulb. The picoboard then sends a message to the AI module to start the camera and scan the room. Every alternate minute, the AI module scans the room to check for any human presence. At the end of each scan the end result is passed as a message to the Picoboard.

If human presence was detected in the room, then the bulb remains turned ON. If there was no human found, then the bulb and the Camera is turned OFF and the AI module waits for a signal for infinite times to get a signal from the Ultrasonic sensor. Upon receiving the signal the Camera and the bulb is turned ON again and this process continues.

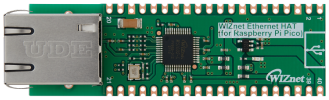
Component Used	Image	Description
Wiznet PICO ethernet hat		Controls the state of the bulb according to the human presence inside the room.
Breadboard and Jumper wire		Helps with interconnecting circuit elements with the Raspberry PICO
USB Webcam		Captures the video/images to the YOLOV3 Object detection model for analysis.
1- Channel relay		Used for connecting the bulb with breadboard and DC

Table 1. Components

Block Diagram

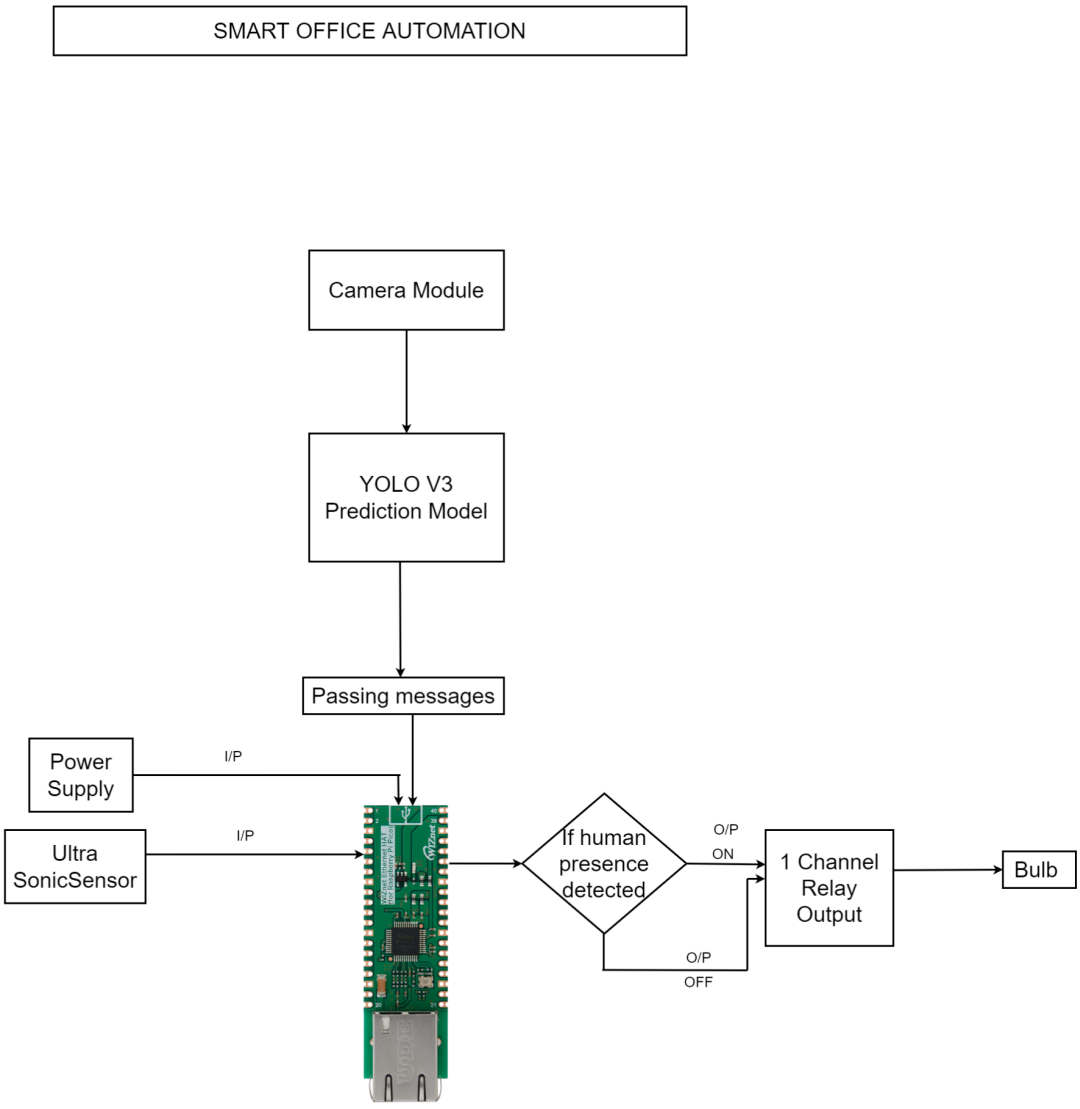


Figure 11. Block Diagram

Block diagram Explanation

This diagram explains the whole project structure. We have used the top-down diagram approach for explaining the project. It consists of the pico board which is the central controller of the Smart Office Automation System. Initially when the door is opened the lights in the room are turned on and the AI module(camera is started) and its scans for the human presence. We have used the YOLO V3 prediction model for prediction. If the model detects the human presence the

particular message is sent to the pico board and the board continues the current state. In the other case the message to turn off the light is sent to the pico board .

Circuit Diagram

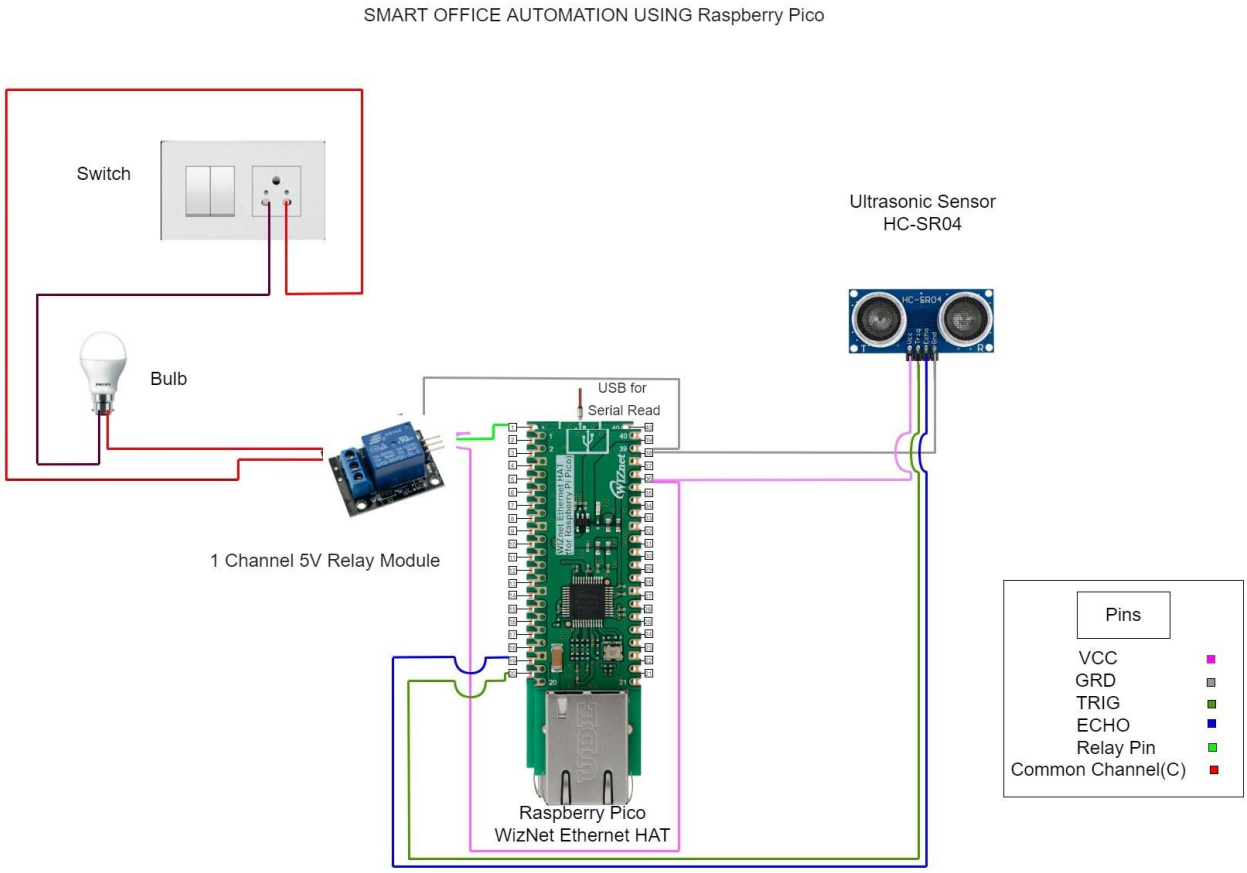


Figure 12. Component Circuit Diagram

Circuit diagram Explanation

VCC , GND pins of the ultrasonic sensor and the 1-channel relay module are connected and coloured by the same colors[pins 36 and 38]. The 1- channel relay module has a connection to the bulb and switch that makes the external AC connection possible. The data pin of the relay module is connected to the pin 1of PicoBoard. The TRIG and ECHO pins of the ultrasonic sensor are connected to [pins 19 and 20].

AI Integration

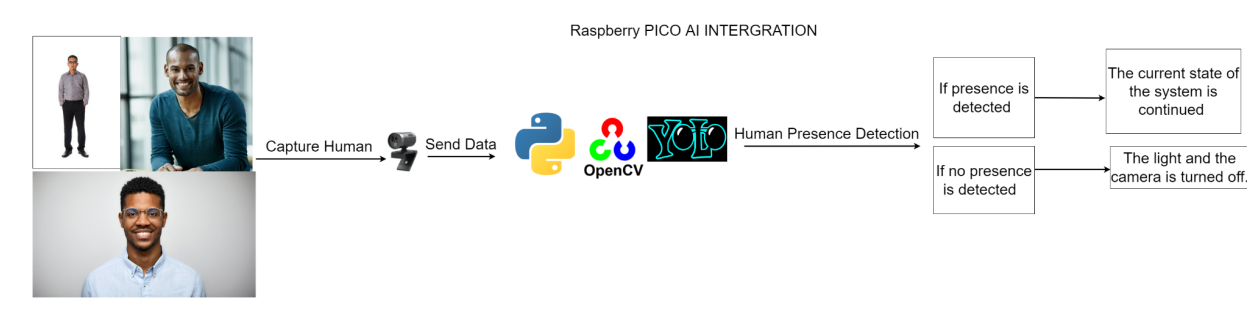


Figure 13. AI Integration in Raspberry Pico

How to integrate AI ?

In this project OpenCV and YOLOV3 were used to detect the human presence inside the cabin room. If the presence of the human is detected then the current state of the system is continued and it scans for checking the human presence in regular intervals of time. If the presence of the human is not detected inside the cabin the AI module will send the turn off message to the PicoBoard and the PicoBoard then turns off the light and also the camera module is turned off. It then waits for the response from the ultrasonic sensor again to turn off the light. The messages that we sent between the systems are in ‘utf-8’ encoded format.

Python AI Code

```
1. #importing libraries
2. import serial
3. import cv2
4. import numpy as np
5. import time
6.
7. # AI function
8. def start_ai():
9.     #
10.     print("Recieved Response!")
11.     net = cv2.dnn.readNet('yolov3.weights', 'yolov3.cfg')
12.     classes = []
13.     with open('coco.names.txt', 'r') as f:
14.         classes = f.read().splitlines()
15.
16.     # video capture
17.     cap = cv2.VideoCapture(1)
18.
19.     while True:
20.         # reinitializing variables for each minute iteration
21.         initial_time = time.time()
22.         final_val = "Human Not Found !!"
23.         bulb_state = "OFF"
24.
25.         while True:
26.
27.             final_time = time.time()
28.             time_diff = final_time - initial_time
29.
30.             # reading frame by frame
31.             _, img = cap.read()
32.             height, width, _ = img.shape
33.
34.             blob = cv2.dnn.blobFromImage(img, 1 / 255, (416, 416), (0,
0, 0), swapRB=True, crop=False)
35.             net.setInput(blob)
36.
37.             output_layers_names = net.getUnconnectedOutLayersNames()
```

```
            layerOutputs = net.forward(output_layers_names)

            val = "No"
            boxes = []
            confidences = []
            class_ids = []

            # detection of human
            for output in layerOutputs:
                for detection in output:
                    scores = detection[5:]
                    class_id = np.argmax(scores)
                    confidence = scores[class_id]

                     # if detected object's confidence is greater than
0.8 then add to list
                    if confidence > 0.8:
                        center_x = int(detection[0] * width)
                        center_y = int(detection[0] * height)
                        w = int(detection[2] * width)
                        h = int(detection[3] * height)

                        x = int(center_x - w / 2)
                        y = int(center_y - h / 2)

                        boxes.append([x, y, w, h])
                        confidences.append((float(confidence)))
                        class_ids.append(class_id)

                        # if human is detected
                        if 0 in class_ids:
                            val = "Yes"
                            final_val = "Human Found !!"
                            bulb_state = "ON"
                    print("Human Presence: ", val)

            indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
            font = cv2.FONT_HERSHEY_PLAIN
            colors = np.random.uniform(0, 255, size=(len(boxes), 3))

```

```
                # check whether human was detected, then show in a
rectangular box
            for i in indexes:
                if class_ids[i] == 0:
                    x, y, w, h = boxes[i]
                    label = str(classes[class_ids[i]])
                    confidence = str(round(confidences[i], 2))
                    color = colors[i]
                     cv2.rectangle(img, (x, y), (x + w, y + h), color,
2)
                    cv2.putText(img, label + " " + confidence, (x, y +
20), font, 2, (255, 255, 255), 2)

            # show the detected output in a window
            cv2.imshow('Observation window for Human Detection', img)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

            # passing message to arduino
            def arduino_write(x):
                arduino.write(bytes(x, 'utf-8'))
                data = arduino.readline()
                return data

            # function to display the observation of the last scan
            def display_observation():
                    print("\n\n\n\nWas there a Human presence in the
Room? : ", final_val)
                      print("What should be the State of BULB : ",
bulb_state)
                time.sleep(5)
                         print("\n\n\nDelay Finished. Scanning the
Room.....\n")

             # print the observation details of the last time interval
             if time_diff > 5:
                 arduino_write(str(bulb_state))
                 display_observation()
                 if bulb_state == "OFF":
                     cap.release()
```

```
                    cv2.destroyAllWindows()
                    return None
                break
                 # here break statement is executed to SET the values of final_val, initial_time and bulb_state


# start point for execution
while True:
          arduino  =  serial.Serial(port="COM23",  baudrate="115200", timeout=.1)
    line = arduino.readline()
    response = line.decode()

    print("Waiting for Response .........")
    print("response: ", response)
    if "start" in response:
        start_ai()
    arduino.close()


```

Python AI Code Explanation

First we import all the libraries required for capturing, processing and writing to Serial Port, line number 2-5. We initialize the serial communication in line number 119.

```
119. arduino = serial.Serial(port="COM23", baudrate="115200", timeout=.1)
```

Serial port is given as COM23, Baudrate is set to 115200 and a timeout of 0.1 second is given for each writes.

```
118. while True:
119.           arduino  =  serial.Serial(port="COM23",  baudrate="115200",
timeout=.1)
120.     line = arduino.readline()
121.     response = line.decode()
122.
123.     print("Waiting for Response .........")
124.     print("response: ", response)
125.     if "start" in response:
126.         start_ai()
127.     arduino.close()
```

Lines 118-127 is an infinite loop which waits for a “start” message from the Ultrasonic sensor. Serial communication is established here. The variable “line” is used to get the message from the ultrasonic sensor. It is then decoded and stored into the variable “response”. When the message is received from the ultrasonic, the “start_ai()” function is called.

```
7. # AI function
8. def start_ai():
9.     #
10.     print("Recieved Response!")
11.     net = cv2.dnn.readNet('yolov3.weights', 'yolov3.cfg')
12.     classes = []
13.     with open('coco.names.txt', 'r') as f:
14.         classes = f.read().splitlines()
15.
16.     # video capture
17.     cap = cv2.VideoCapture(1)
```

We use yolov3 for object detection and the detected class names are stored into the “classes” list.

We use 'yolov3.weights', 'yolov3.cfg' for the detection of humans. The “coco.names.txt” file has all the class names that yolov3 can identify. OpenCV video capturing is enabled and set to 1 for external webcam. Lines 19–113 are infinite loops.

```
19.     while True:
20.         # reinitializing variables for each minute iteration
21.         initial_time = time.time()
22.         final_val = "Human Not Found !!"
23.         bulb_state = "OFF"
24.
25.         while True:
```

Initially, the bulb state is set to OFF and Human presence is not found. Initial time is also captured for calculating the amount of time the code has to be run.

```
52.                      # if detected object's confidence is greater than
0.8 then add to list
53.                     if confidence > 0.8:
54.                         center_x = int(detection[0] * width)
55.                         center_y = int(detection[0] * height)
56.                         w = int(detection[2] * width)
57.                         h = int(detection[3] * height)
58.
59.                         x = int(center_x - w / 2)
60.                         y = int(center_y - h / 2)
61.
62.                         boxes.append([x, y, w, h])
63.                         confidences.append((float(confidence)))
64.                         class_ids.append(class_id)
```

If the detected object’s confidence is greater than 0.8 then we are taking that object into consideration and identifying the positions of the box to be drawn around the human.
“class_ids” will be having the list of the class_id of all the objects detected.

```
66.                         # if human is detected
67.                         if 0 in class_ids:
68.                             val = "Yes"
69.                             final_val = "Human Found !!"
70.                             bulb_state = "ON"
71.                     print("Human Presence: ", val)
```

‘0’ is the class_id for Human, so if it is present inside the class_ids list then it shows that human was identified and values of bulb_state and final_val are changed accordingly.

```
77.                 # check whether human was detected, then show in a
rectangular box
78.             for i in indexes:
79.                 if class_ids[i] == 0:
80.                     x, y, w, h = boxes[i]
81.                     label = str(classes[class_ids[i]])
82.                     confidence = str(round(confidences[i], 2))
83.                     color = colors[i]
84.                      cv2.rectangle(img, (x, y), (x + w, y + h), color,
2)
85.                     cv2.putText(img, label + " " + confidence, (x, y +
20), font, 2, (255, 255, 255), 2)
```

Lines from 77-85 draws the rectangular box around the human. The variables x,y,w,h has the coordinates for the box to be drawn around the human. The color of the rectangular box is set to random so the color of the box changes every second. The Label given to the rectangular box is taken from class_ids.

```
87.             # show the detected output in a window
88.             cv2.imshow('Observation window for Human Detection', img)
```

A window is opened to show the output of the room scanning. The text to be displayed on the window is also given here.

```
# passing message to arduino
93.             def arduino_write(x):
94.                 arduino.write(bytes(x, 'utf-8'))
95.                 data = arduino.readline()
96.                 return data
```

At the end of every 1 minute, the arduino_write() function passes a message to the picoboard, which is either ‘ON’ or 'OFF' according to which the light is Turned ON and OFF.

```
106.             if time_diff > 5:
107.                 arduino_write(str(bulb_state))
108.                 display_observation()
109.                 if bulb_state == "OFF":
110.                     cap.release()
111.                     cv2.destroyAllWindows()
112.                     return None
113.                 Break
```

If the time difference is greater than 5 secs then the arduino_write() function is called and observation for the previous scanning is printed. If bulb_state is ‘OFF’ then scanning is stopped and all windows are closed and Scanning also stops.

Arduino Code

```
1. const int pingPin = 15; // Trigger Pin of Ultrasonic Sensor
2. const int echoPin = 14; // Echo Pin of Ultrasonic Sensor
3.
4. void setup() {
5.    Serial.begin(9600); // Starting Serial Terminal
6.    // initialize digital pin LED_BUILTIN as an output.
7.   pinMode(LED_BUILTIN, OUTPUT);
8.   pinMode(0,OUTPUT);// wait for a second
9.   digitalWrite(LED_BUILTIN,HIGH);
10. }
11.
12. String incomingdata = "";
13.
14. void loop()
15. {
16.   // for motion detection
17.
18.   long duration, inches;
19.   pinMode(pingPin, OUTPUT);
20.   digitalWrite(pingPin, LOW);
21.   delayMicroseconds(2);
22.   digitalWrite(pingPin, HIGH);
23.   delayMicroseconds(10);
24.   digitalWrite(pingPin, LOW);
25.   pinMode(echoPin, INPUT);
26.   duration = pulseIn(echoPin, HIGH);
27.   inches = microsecondsToInches(duration);
28.   // if motion is detected within 5 Inches from the sensor:
29.   if(inches < 4)
30.   {
31.     Serial.println("start");
32.     digitalWrite(0, HIGH);
33.   }
```

```
34.    else
35.    {
36.      Serial.println(inches);
37.    }
38.
39.    // response from AI module to turn ON and OFF lights
40.    incomingdata = Serial.readString();
41.    if(incomingdata=="ON")
42.    {
43.      digitalWrite(0,HIGH);
44.    }
45.    if(incomingdata == "OFF")
46.    {
47.      digitalWrite(0,LOW);
48.    }
49.
50. }
51.
52. long microsecondsToInches(long microseconds) {
53.    return microseconds / 74 / 2;
54. }
```

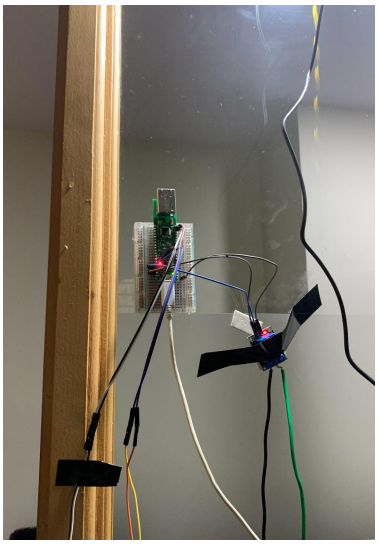
Arduino Code Explanation

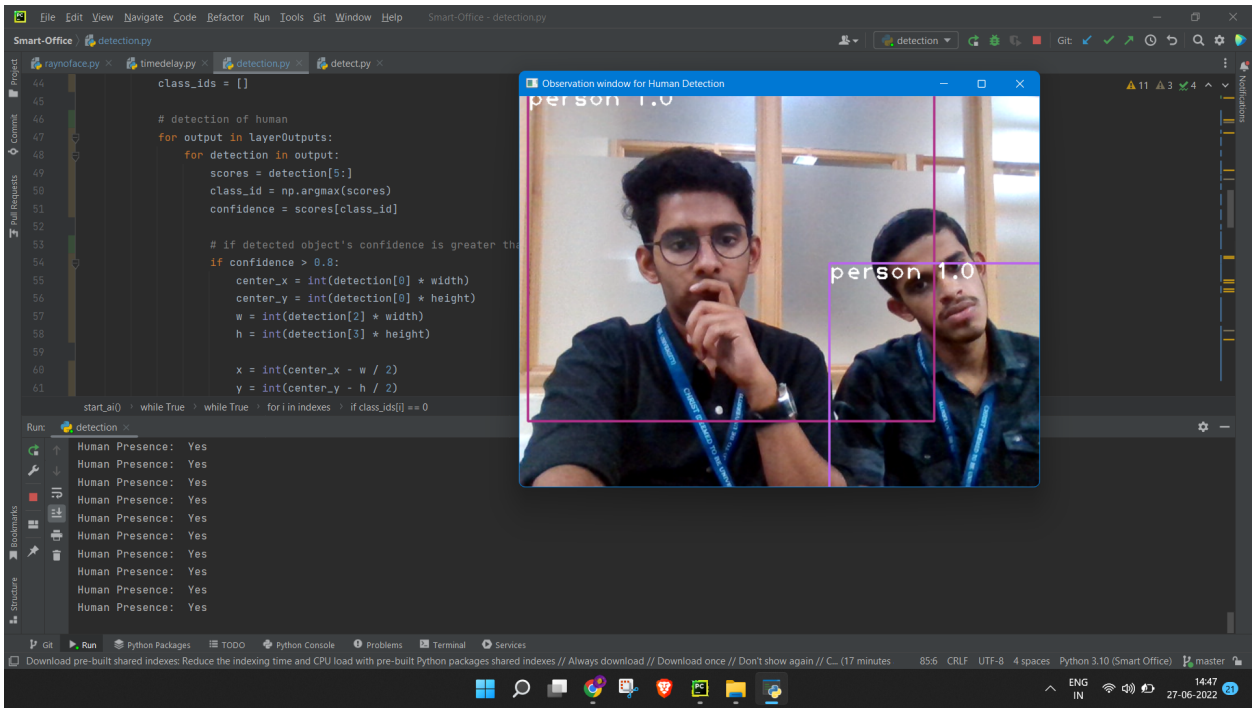
The code starts from line number 1,which assigns the pins number for the TRIG pin and ECHO pin for the ultrasonic sensor; they are specified earlier because the distance is calculated from those pins. After that we start with the void setup() and void loop() functions where the actual code execution takes place.In the setup file the ultrasonic sensors and the bulbs are connected and their pins are specified, Serial is enabled at baudrate = 115200 and a time out of 1 millisecond is given for the incoming serial data. A string variable ‘incomingdata’ is specified so that we can pass the message between the systems. We are calculating the distances between the objects and we have used a condition in which if the inches is less than 4 the lights are turned on.

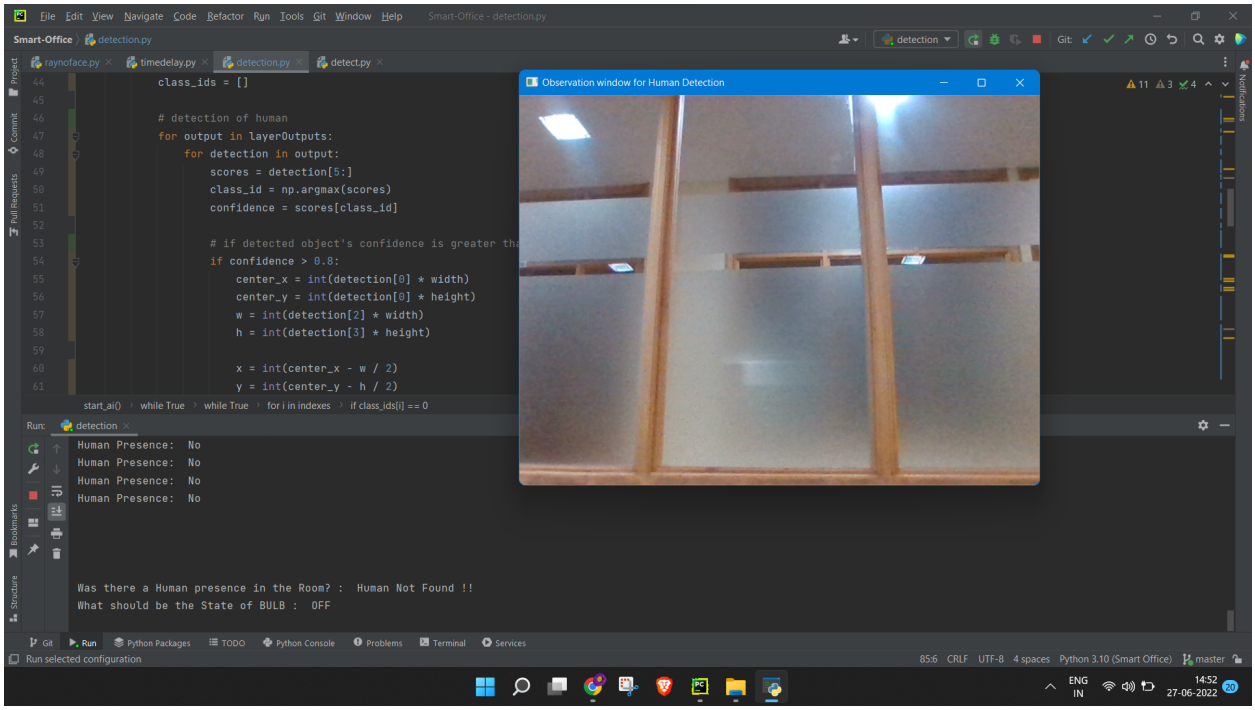
After that condition we have written the code for the AI response code that is to make the bulb continue its state and if no human presence is detected then turn off the lights. The

conditions incomingdata are ON and OFF. The on and off messages are sent and received in the ‘utf-8’ format.The final line of code is a calculation of distance between the object.

Images







Video Link

- https://drive.google.com/file/d/1g1eALu-ZrlJfds5gVNDGmfiTYcNjbFyZ/view?usp=sharing

References

- Python: Real Time Object Detection (Image, Webcam, Video files) with Yolov3 an…
- https://github.com/emasterclassacademy/Single-Multiple-Custom-Object-Detection
- https://www.arrow.com/en/research-and-events/articles/ultrasonic-sensors-how-they-work-and-how-to-use-them-with-arduino#:~:text=Ultrasonic%20sensors%20work%20by%20emitting,return%20after%20hitting%20an%20object.