Occupancy Recognition through the Smart Door System - The Design and The Failure Diagnostics

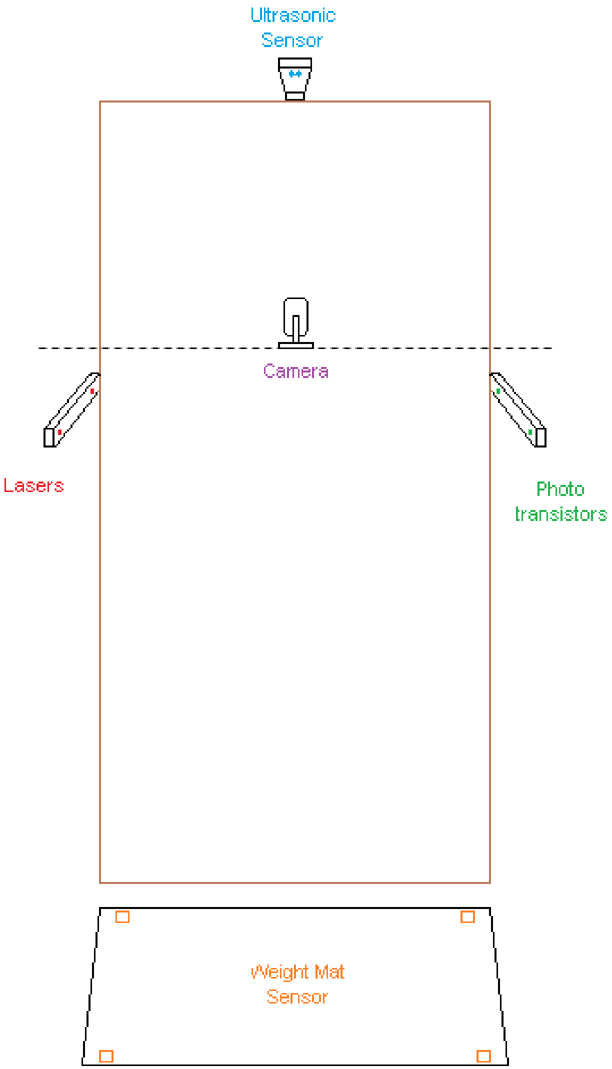
M.Tech Project Thesis of

Cynthia Josephine J

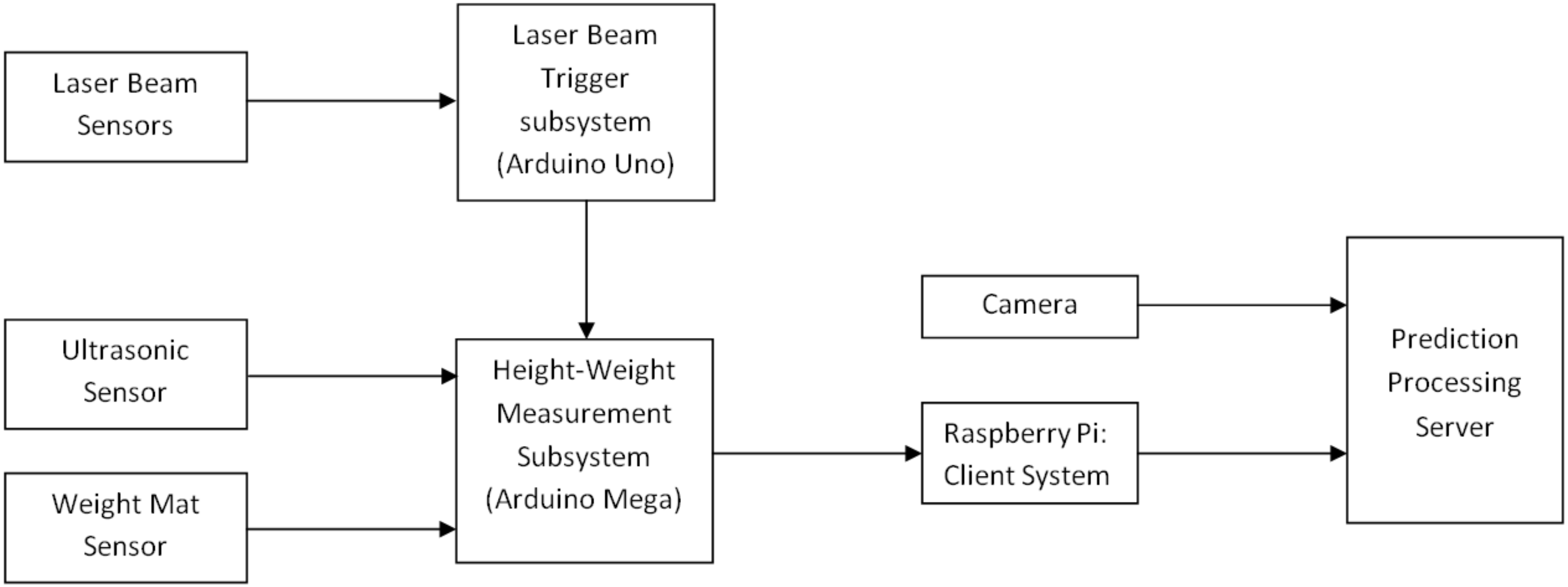
Under the guidance of

Prof. Krithi Ramamritham

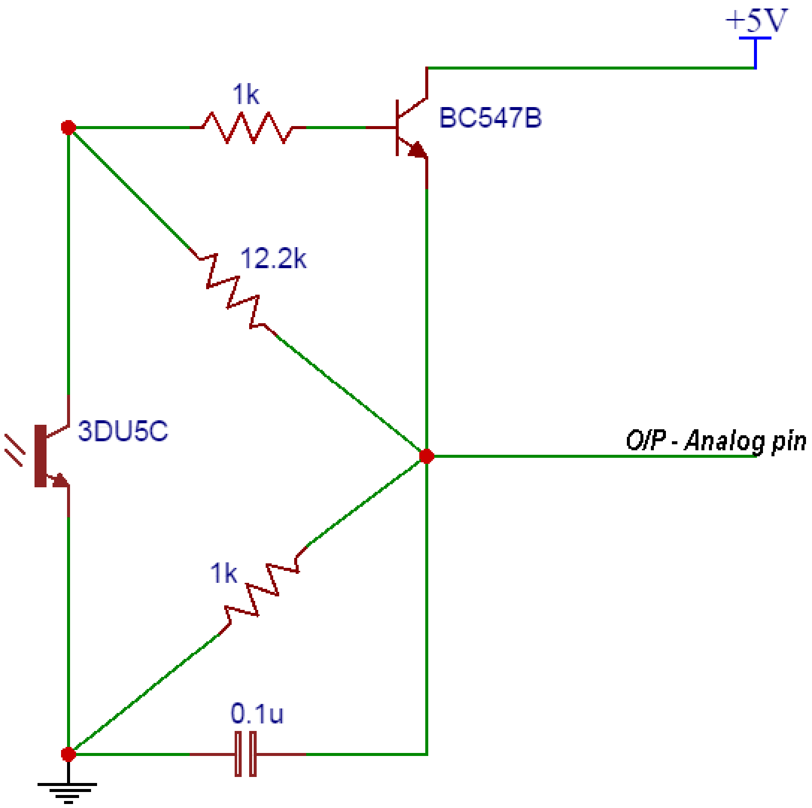
The Smart Door



Smart Door Sketch


Laser Beam Sensors
Laser Beam Trigger subsystem (Arduino Uno)
Ultrasonic Sensor
Weight Mat Sensor
Height-Weight Measurement Subsystem (Arduino Mega)
Camera
Raspberry Pi: Client System
Prediction Processing Server

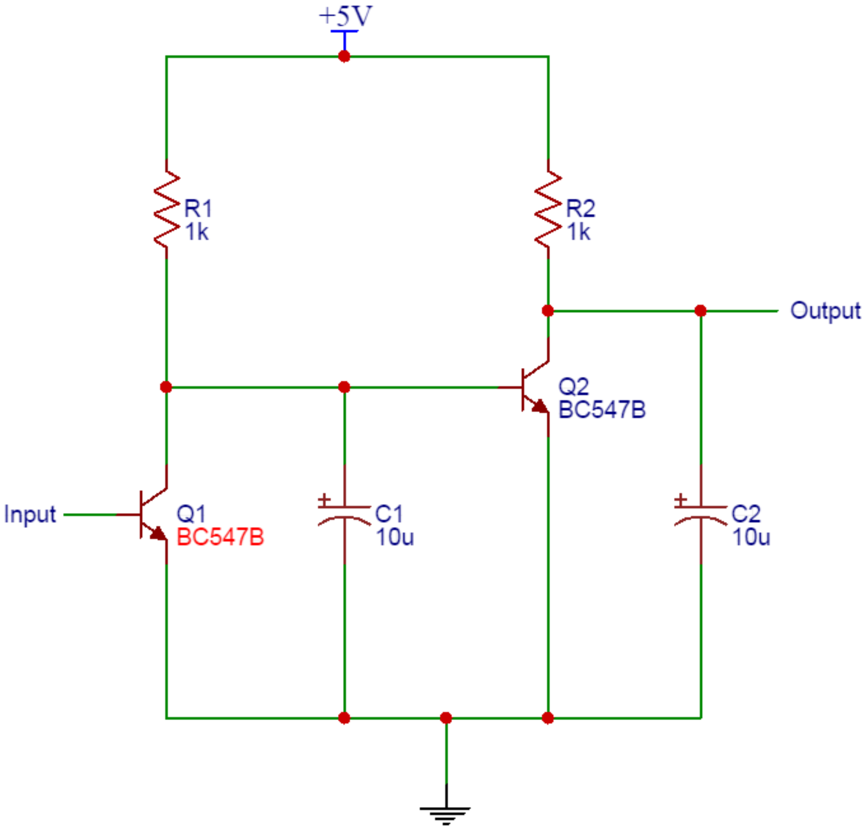
Laser Beam Sensor Circuit



- The output will be of HIGH range value (analog) if the laser is obstructed from falling on the phototransistor

- The output will be of LOW range value (analog) if the laser is falling on the phototransistor

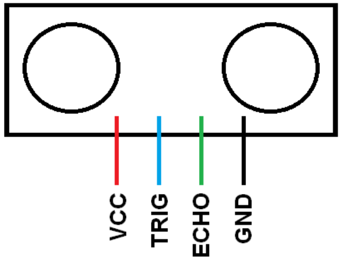
Laser Beam Sensor –
Signal Conditioning Circuit



- The circuit is to condition the input signal protecting it from interference from external voltage fluctuation.

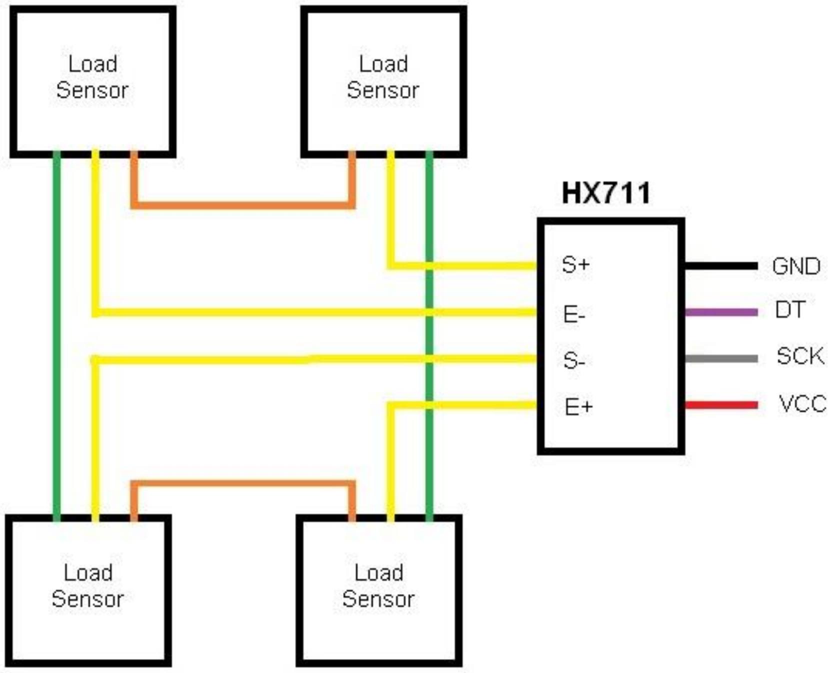
- This is a simple two switch circuit. ie, The output will be of same value (analog) as the input

Ultrasonic Sensor Module



- To start measurement,
  - Trig of SR04 must receive a pulse of high (5V) for at least 10us,
  - this will initiate the sensor will transmit out 8 cycle of ultrasonic burst at 40kHz and
  - wait for the reflected ultrasonic burst.
- When the sensor detected ultrasonic from receiver,
  - it will set the Echo pin to high (5V) and
  - delay for a period (width) which proportion to distance.
- To obtain the distance, measure the width (Ton) of Echo pin.
  - Time = Width of Echo pulse, in uS (micro second)
  - Or you can utilize the speed of sound, which is 340m/s

Weight Mat Sensor Circuit

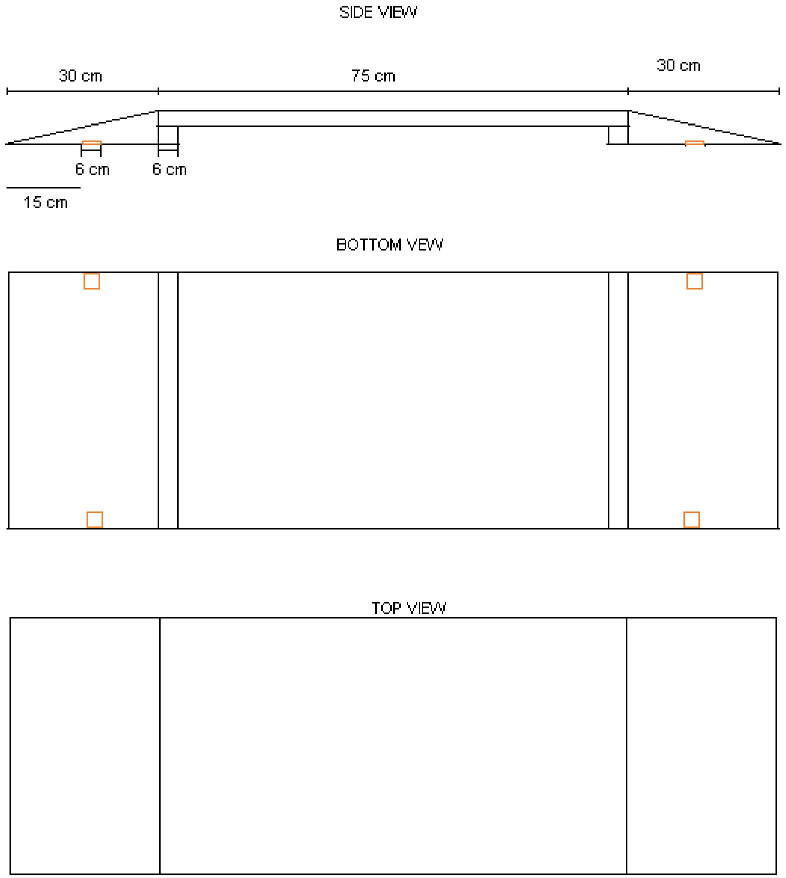


- Load cells use a four-wire Wheatstone bridge configuration to connect to the HX711.

- HX711 is a precision 24-bit analog-to-digital converter (ADC) designed for weigh scales and industrial control applications to interface directly with a bridge sensor.

- The HX711 uses a two-wire interface (Clock and Data) for communication

Weight Mat Sensor



Laser Beam - Trigger subsystem

- In every Arduino cycle the following is done:
  - Check if there is change of state in Laser Beam 1
    - If L1_State is High, then set A = current time
    - If L1_State is Low, then set B = current time
  - Check if there is change of state in Laser Beam 2
    - If L2_State is High, then set C = current time
    - If L2_State is Low, then set D = current time
  - Check if the laser is cut by an obstacle or by a person
    - To verify this, (B - A) and (D-C) should be in certain range of values.

Laser Beam - Trigger subsystem

- **Possible Entry:** Laser 1 is cut first.
  - Send a signal to indicate to ***start*** the recording of the ***camera*** video feed.
  - Send a signal to ***start*** recording the height and the weight measurements.
- **Possible Exit:** Laser 2 is cut first.
  - Send a signal to ***start*** recording the height and the weight measurements.
- **Valid Entry:** Laser 1 is cut before Laser 2
  - Send a signal to ***stop*** recording the height and the weight measurements.
  - Send a signal to ***send*** the recorded the height and the weight measurements.
  - Send a signal to ***reset*** the recorded the height and the weight parameters.
  - Enable ***Entry_Indicator_Pin*** for short duration
- **Valid Exit:** Laser 1 is cut after Laser 2.
  - Send a signal to ***stop*** recording the height and the weight measurements.
  - Send a signal to ***send*** the recorded the height and the weight measurements.
  - Send a signal to ***reset*** the recorded the height and the weight parameters.
  - Enable ***Exit_Indicator_Pin*** for short duration

Height Weight Measurement Subsystem

- When the Signal to ***start*** measuring the height and the weight values is HIGH, start the measuring till it becomes LOW.
  - weight values - Max Function
  - height values - Max Function

- When the Signal to ***send*** the recorded height and weight values is HIGH, send the measured value through Serial Function

- When the Signal to ***reset*** the recorded height and the weight values is HIGH, set the values to zero.

Raspberry Pi: Client System

Existing Implementation:

- The Connection to the Arduino Serial Port is established
- The Connection to the Server is established
- The Connection to the Database is established
- The new session Id is inferred from the old session ID stored in the database.
- For every data got from the Arduino via Serial Port, check if it is of the format ***Direction/Weight/Height***
  - If No, ignore the data received from the Arduino
  - If yes, Continue the below steps:
    - Insert the height, weight, direction, timestamp along with the Session ID in the database
    - Update the occupancy count of the people in the lab based on the entry and exit
    - Increase the session ID by 1

Raspberry Pi: Client System

Existing Implementation - Drawbacks:

- As the connections are done in the beginning, the client script will halt as soon as
  - the Arduino gets physically disconnected from the Raspberry Pi.
  - The Database service is stopped or when there is poor internet connectivity.
  - if the Server is not reachable due to the poor internet connectivity.
- The client cannot reestablish the connection and also the server cannot listen to the data from the Raspberry Pi as soon as the client Script is run again.
  - The Server Script needs to be stopped and then restarted again.
  - The client script is run to establish the connection again.
- The smart door cannot recover its function as soon as the failure conditions are met and the user has to restart the scripts.

Raspberry Pi: Client System

Updated Implementation:

- The Connection to the Arduino Serial Port is established.
- For every data got from the Arduino via Serial Port
  - Check if the data is of the format **R/X.** If yes,
    - Establish connection to the database and get the latest session ID and close the connection
    - Connect to the Server and send a message of format ***SessionID/R.*** This will make the server to start image processing
  - Check if the data is of the format ***Direction/Weight/Height***. If yes,
    - Establish Connection to the Database and do the following and close the connection:
      - Insert the height, weight, direction, timestamp along with the Session ID in the database
      - Update the occupancy count of the people in the lab based on the entry and exit.
    - Connect to the Server and send a message of format ***SessionID/E.*** This will make the server to do prediction based on the Height & the weight
  - If the data is not in the above formats, then ignore the data.
- Check if the established connection to the arduino is not broken. If the connection is broken, reconnect it.

Prediction Processing Server

- There are 2 scripts doing the process of predicting the person entering the room.
  - Prediction of Person based on Height and Weight
  - Prediction of Person based on Face Recognition

- In both the cases, the server establishes the connection with the client (Raspberry Pi) waiting for the signal to any one of the above tasks.

Prediction Processing Server –Height and Weight

- Already implemented previously.

- Random Forest Classifier.

- When the person exits the room, only the prediction based on height and weight happens.

- Whereas, when the person enters the lab, the top three prediction results
  - Identity
  - Probability

  are being stored in the database and then used in the further calculation.

Prediction of Person based on Face Recognition

- Start the camera and record the camera feed video in the grayscale as the video processing of one channel (grayscale) is easier and robust than maintain 3 channels (RGB).
- For every frame in the camera video feed, do the following:
  - Detect if there is any face present in that frame.
  - If the face is detected, then recognize the face in that frame using LPBH(Local Binary Patterns Histograms) Algorithm
  - Get the following parameters of that frame.
    - Area of the face
    - Identity of the person recognized.
    - Confidence of the predicted identity
    - Frame Number in the Camera Video Feed

Prediction of Person based on Face Recognition

- Aggregate the result of each of the image processed frame of the camera video feed being recorded.

  For each identity,

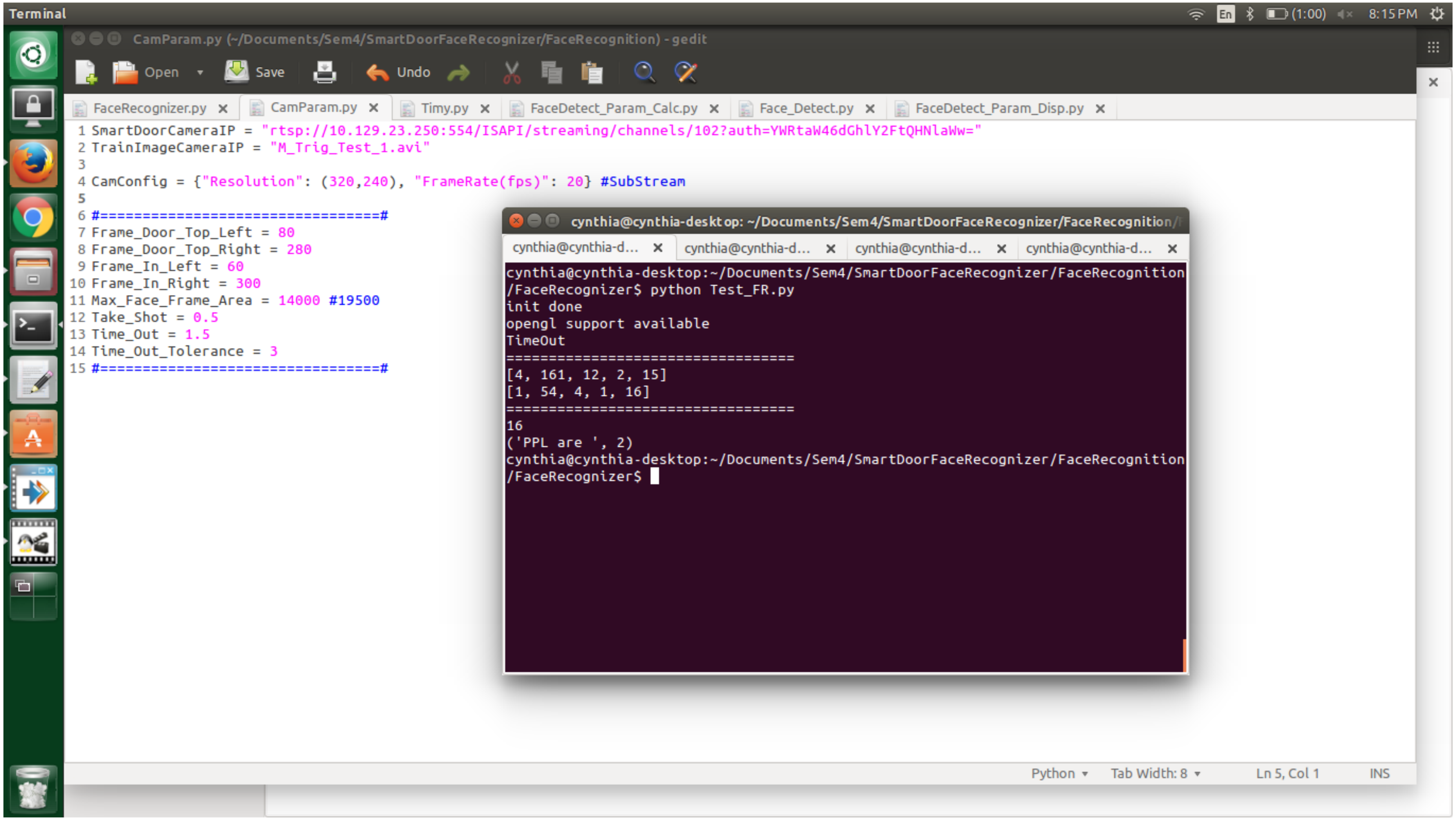
  - Sum up the Confidence Factor
  - Count the number of frames that shows this same identity.
  - The minimum (smallest) frame number which shows this identity.
  - The maximum (largest) frame number which shows this identity.

- Click the picture of dimension specified by
  - Pic_Frame_Top*
  - Pic_Frame_Bottom*
  - Pic_Frame_Left*
  - Pic_Frame_Right*

  once at the time specified by the parameter Take_CamShot*
- Calculate the total frames with face detected in the Camera Video Feed being recorded.

Prediction of Person based on Face Recognition

- Quit recording the video when any one of the following happens:
  - Person moves out of the frame. The following parameters represents the threshold of the Frame in Pixel X direction:
    - Frame_In_Left*
    - Frame_In_Right*
  - Person reaches too close to the camera. Closer the person, Larger the area of the face detected. Area should not exceed the parameter Max_Face_Frame_Area*
  - Time of recording has reached the threshold limit.
    - The parameter Time_Out* gives the threshold limit of recording time.
    - Even after the time of recording has reached its limit, some extra time is given in terms of number of steps(loops) Time_Out_Tolerance* representing the number of loop cycles allowed after timeout

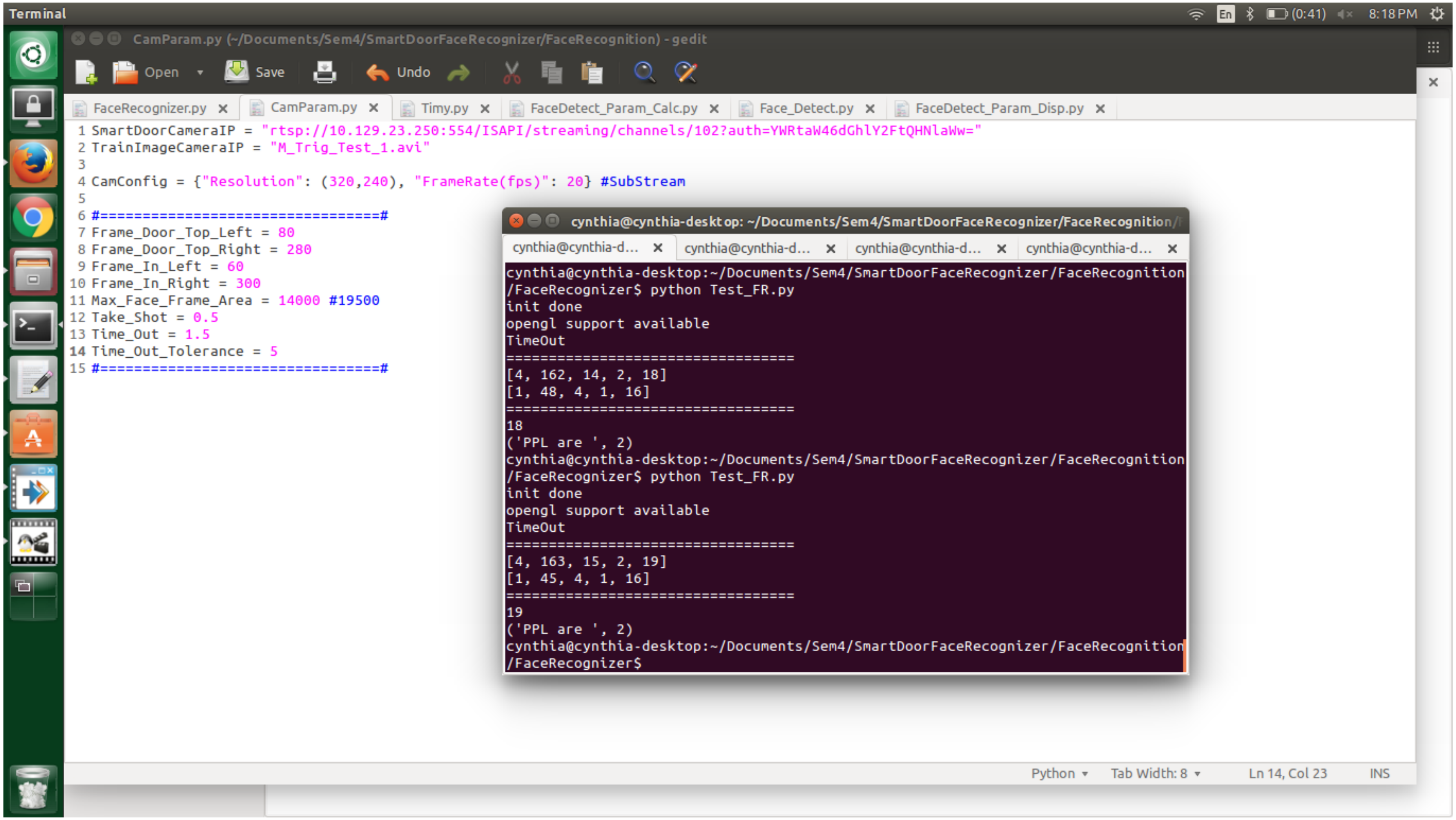


```python
SmartDoorCameraIP = "rtsp://10.129.23.250:554/ISAPI/streaming/channels/102?auth=YWRtaW46dGhlY2FtQHNlaWw="
TrainImageCameraIP = "M_Trig_Test_1.avi"

CamConfig = {"Resolution": (320,240), "FrameRate(fps)": 20} #SubStream

#=================================#
Frame_Door_Top_Left = 80
Frame_Door_Top_Right = 280
Frame_In_Left = 60
Frame_In_Right = 300
Max_Face_Frame_Area = 14000 #19500
Take_Shot = 0.5
Time_Out = 1.5
Time_Out_Tolerance = 3
#=================================#
```

```
cynthia@cynthia-desktop:~/Documents/Sem4/SmartDoorFaceRecognizer/FaceRecognition
/FaceRecognizer$ python Test_FR.py
init done
opengl support available
TimeOut
==================================
[4, 161, 12, 2, 15]
[1, 54, 4, 1, 16]
==================================
16
('PPL are ', 2)
cynthia@cynthia-desktop:~/Documents/Sem4/SmartDoorFaceRecognizer/FaceRecognition
/FaceRecognizer$ 
```

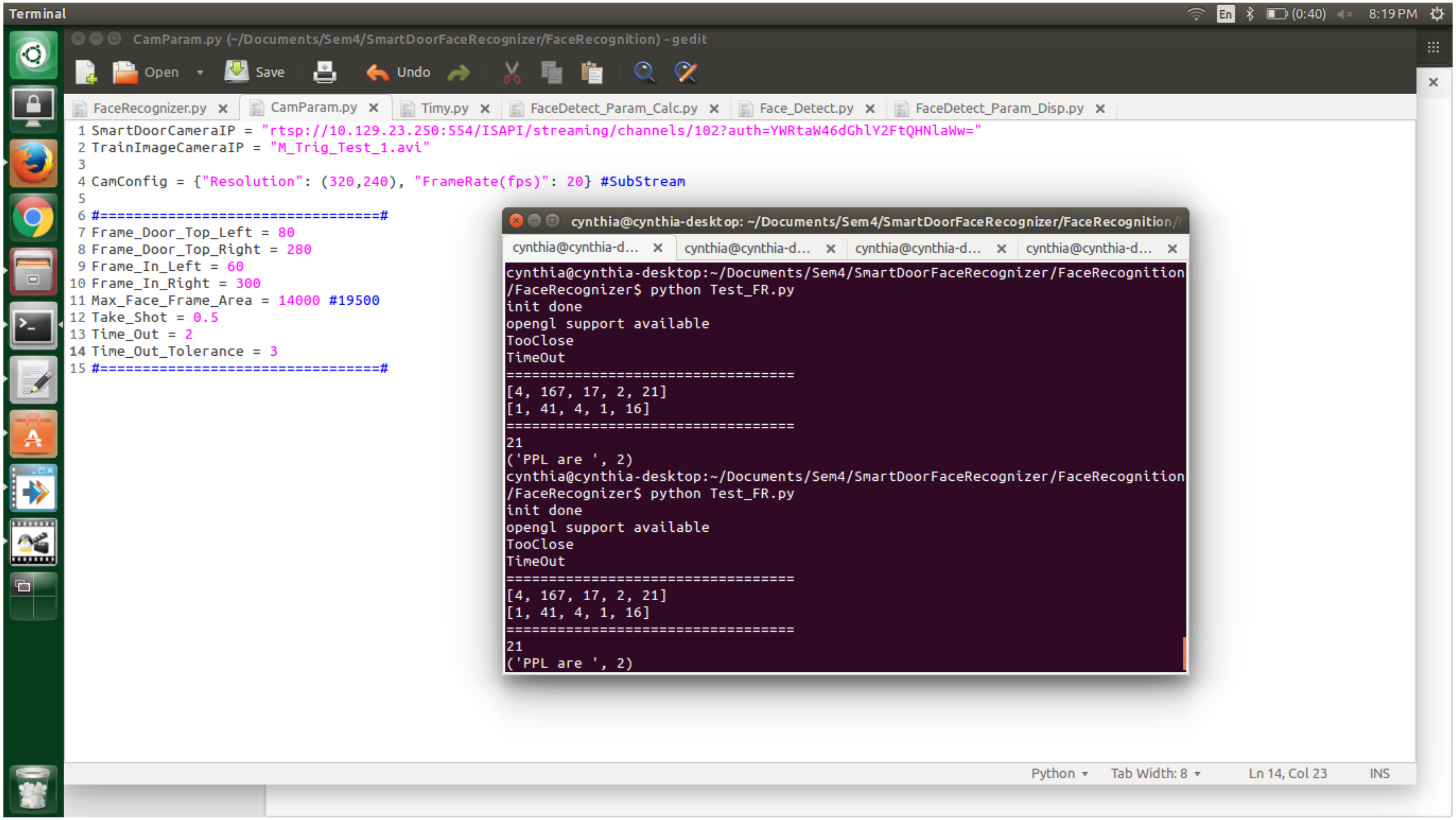


```python
SmartDoorCameraIP = "rtsp://10.129.23.250:554/ISAPI/streaming/channels/102?auth=YWRtaW46dGhlY2FtQHNlaWw="
TrainImageCameraIP = "M_Trig_Test_1.avi"

CamConfig = {"Resolution": (320,240), "FrameRate(fps)": 20} #SubStream

#=================================#
Frame_Door_Top_Left = 80
Frame_Door_Top_Right = 280
Frame_In_Left = 60
Frame_In_Right = 300
Max_Face_Frame_Area = 14000 #19500
Take_Shot = 0.5
Time_Out = 1.5
Time_Out_Tolerance = 5
#=================================#
```

```
cynthia@cynthia-desktop:~/Documents/Sem4/SmartDoorFaceRecognizer/FaceRecognition
/FaceRecognizer$ python Test_FR.py
init done
opengl support available
TimeOut
==================================
[4, 162, 14, 2, 18]
[1, 48, 4, 1, 16]
==================================
18
('PPL are ', 2)
cynthia@cynthia-desktop:~/Documents/Sem4/SmartDoorFaceRecognizer/FaceRecognition
/FaceRecognizer$ python Test_FR.py
init done
opengl support available
TimeOut
==================================
[4, 163, 15, 2, 19]
[1, 45, 4, 1, 16]
==================================
19
('PPL are ', 2)
cynthia@cynthia-desktop:~/Documents/Sem4/SmartDoorFaceRecognizer/FaceRecognition
/FaceRecognizer$
```

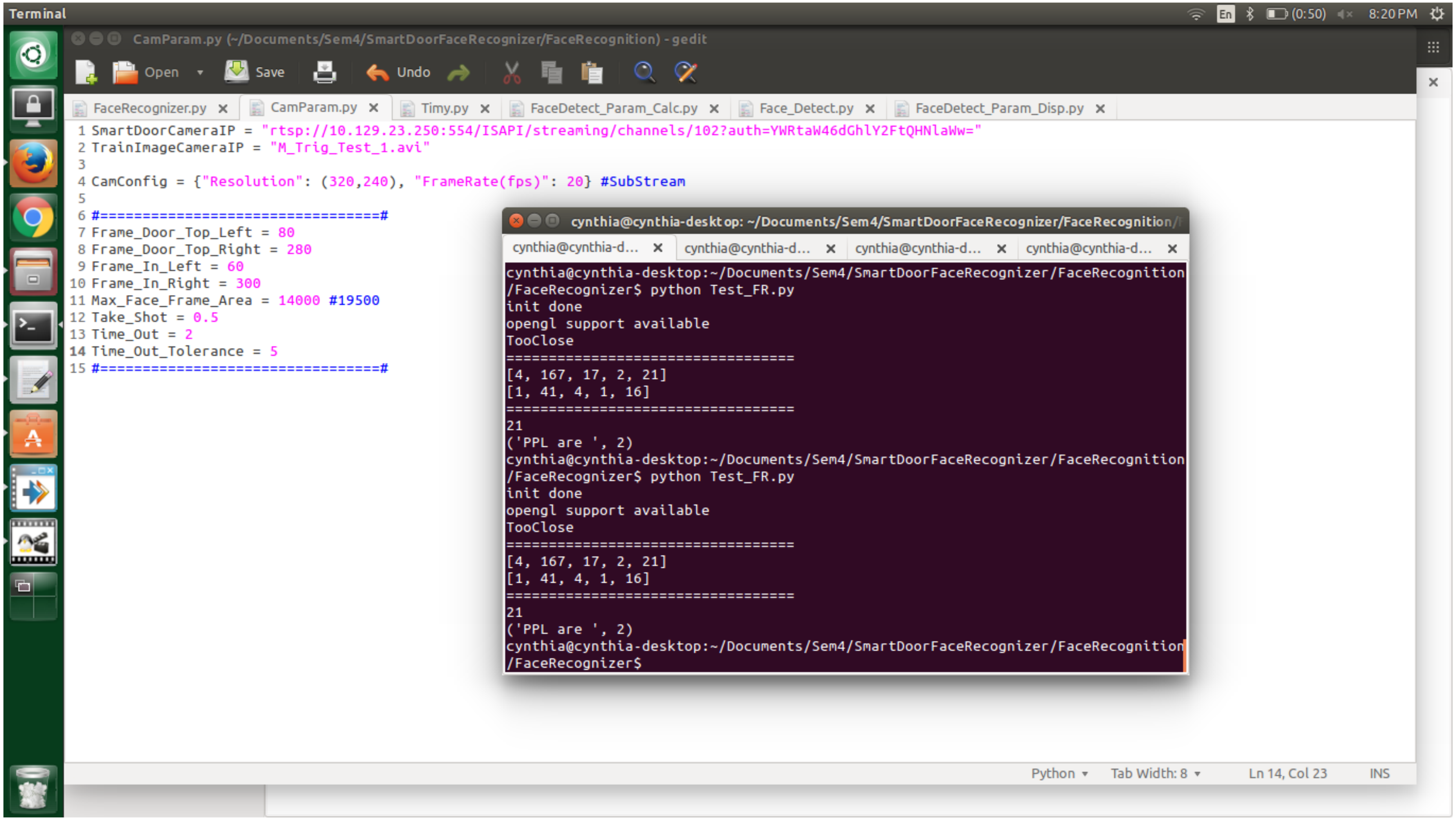


```python
SmartDoorCameraIP = "rtsp://10.129.23.250:554/ISAPI/streaming/channels/102?auth=YWRtaW46dGhlY2FtQHNlaWw="
TrainImageCameraIP = "M_Trig_Test_1.avi"

CamConfig = {"Resolution": (320,240), "FrameRate(fps)": 20} #SubStream

#=================================#
Frame_Door_Top_Left = 80
Frame_Door_Top_Right = 280
Frame_In_Left = 60
Frame_In_Right = 300
Max_Face_Frame_Area = 14000 #19500
Take_Shot = 0.5
Time_Out = 2
Time_Out_Tolerance = 3
#=================================#
```

```
cynthia@cynthia-desktop:~/Documents/Sem4/SmartDoorFaceRecognizer/FaceRecognition
/FaceRecognizer$ python Test_FR.py
init done
opengl support available
TooClose
TimeOut
==================================
[4, 167, 17, 2, 21]
[1, 41, 4, 1, 16]
==================================
21
('PPL are ', 2)
cynthia@cynthia-desktop:~/Documents/Sem4/SmartDoorFaceRecognizer/FaceRecognition
/FaceRecognizer$ python Test_FR.py
init done
opengl support available
TooClose
TimeOut
==================================
[4, 167, 17, 2, 21]
[1, 41, 4, 1, 16]
==================================
21
('PPL are ', 2)
```

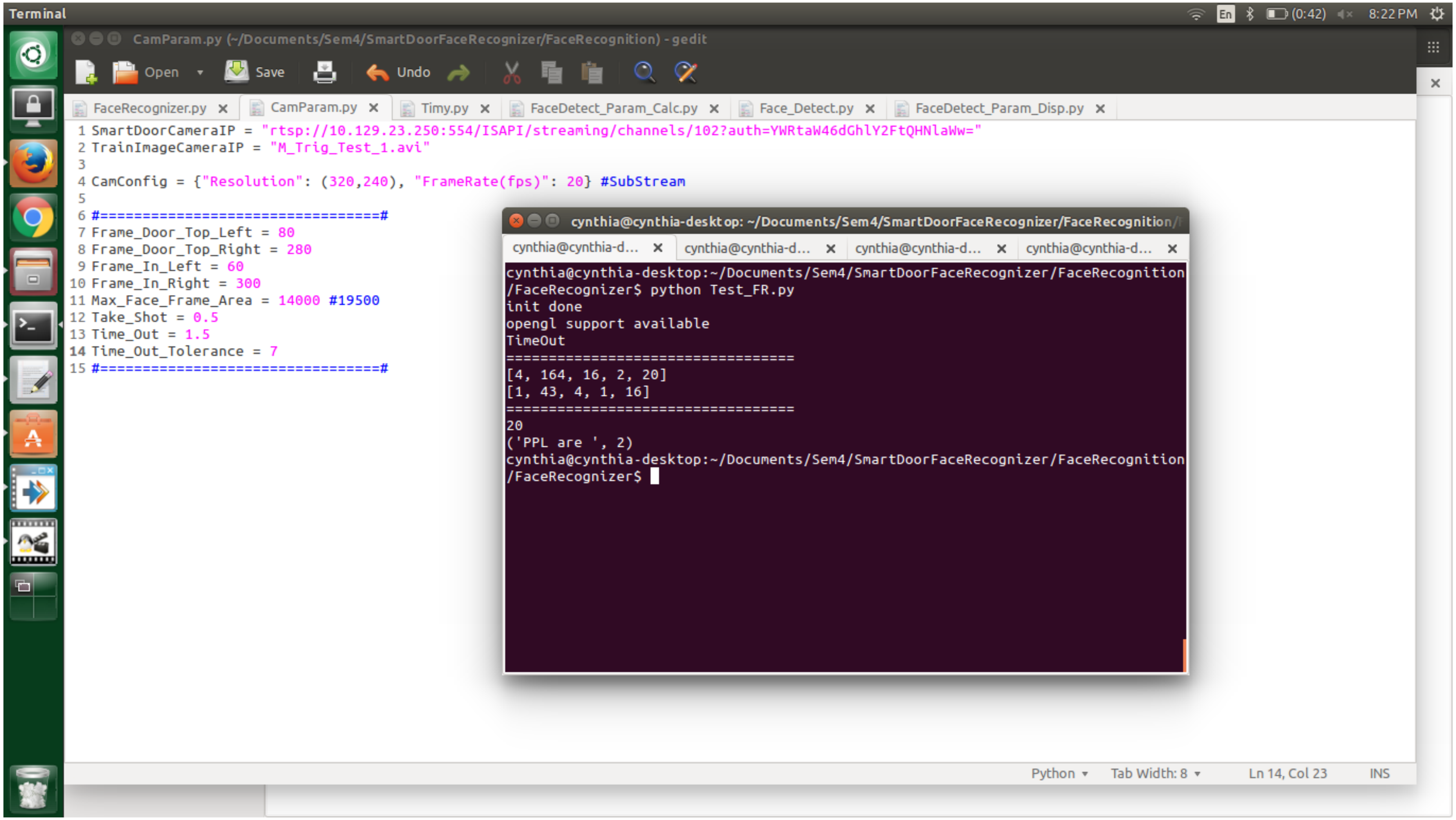


```python
SmartDoorCameraIP = "rtsp://10.129.23.250:554/ISAPI/streaming/channels/102?auth=YWRtaW46dGhlY2FtQHNlaWw="
TrainImageCameraIP = "M_Trig_Test_1.avi"

CamConfig = {"Resolution": (320,240), "FrameRate(fps)": 20} #SubStream

#=================================#
Frame_Door_Top_Left = 80
Frame_Door_Top_Right = 280
Frame_In_Left = 60
Frame_In_Right = 300
Max_Face_Frame_Area = 14000 #19500
Take_Shot = 0.5
Time_Out = 2
Time_Out_Tolerance = 5
#=================================#
```

```
cynthia@cynthia-desktop:~/Documents/Sem4/SmartDoorFaceRecognizer/FaceRecognition
/FaceRecognizer$ python Test_FR.py
init done
opengl support available
TooClose
==================================
[4, 167, 17, 2, 21]
[1, 41, 4, 1, 16]
==================================
21
('PPL are ', 2)
cynthia@cynthia-desktop:~/Documents/Sem4/SmartDoorFaceRecognizer/FaceRecognition
/FaceRecognizer$ python Test_FR.py
init done
opengl support available
TooClose
==================================
[4, 167, 17, 2, 21]
[1, 41, 4, 1, 16]
==================================
21
('PPL are ', 2)
cynthia@cynthia-desktop:~/Documents/Sem4/SmartDoorFaceRecognizer/FaceRecognition
/FaceRecognizer$
```



```
SmartDoorCameraIP = "rtsp://10.129.23.250:554/ISAPI/streaming/channels/102?auth=YWRtaW46dGhlY2FtQHNlaWw="
TrainImageCameraIP = "M_Trig_Test_1.avi"

CamConfig = {"Resolution": (320,240), "FrameRate(fps)": 20} #SubStream

#=================================#
Frame_Door_Top_Left = 80
Frame_Door_Top_Right = 280
Frame_In_Left = 60
Frame_In_Right = 300
Max_Face_Frame_Area = 14000 #19500
Take_Shot = 0.5
Time_Out = 1.5
Time_Out_Tolerance = 7
#=================================#
```

```
cynthia@cynthia-desktop:~/Documents/Sem4/SmartDoorFaceRecognizer/FaceRecognition
/FaceRecognizer$ python Test_FR.py
init done
opengl support available
TimeOut
==================================
[4, 164, 16, 2, 20]
[1, 43, 4, 1, 16]
==================================
20
('PPL are ', 2)
cynthia@cynthia-desktop:~/Documents/Sem4/SmartDoorFaceRecognizer/FaceRecognition
/FaceRecognizer$
```

Prediction of Person based on Face Recognition

- Now calculate and record the following parameters for the entire Camera Video Feed Recorded:
  - Identity
  - Confidence Factor
  - No of Frames
  - Minimum Frame Number
  - Maximum Frame Number
- Store only the top 3 results based on the confidence factor.

Integration of two Modules

- When the person enters the room, the prediction results depends upon the video recorded by camera as well as the height and weight sensed by existing Smart Door.

- The following Parameters are got from the above 2 algorithms
  - Identity
  - Probability based on Height and Weight
  - Confidence Factor
  - No of Frames
  - Minimum Frame Number
  - Maximum Frame Number

Diagnostics of prediction algorithms' failures

The diagnostics of the two server prediction scripts should be done to verify the following situations:

- Check if the database service is up and there is a proper connection to the database.
- Check if the PKL file of the height and weight prediction classifier exists or not.
- Check if the face detector XML exists or not.
- Check if the face recognizer YML file exists or not.
- Check if the Rpi is working and is reachable.

Training of the data

- The training of the height and the weight is done by making the user pass through the door for more number of times.

- The Training of face database is done through extraction of images from the video showing various user expression and head tilts.

Future Works

- ***Confidence Factor*** could be given weightage according to the parameter ***Frame Number*** instead of considering them as two individual parameters.
  - Confidence Factor * Frame Number * Weightage_Factor, making a ***new nonlinear parameter***.
  - This could be done after we collect sufficient number of data of people entering the door.
- There could be an ***automatic training*** of the data (height and weight or Images) once certain conditions are met.
  - These conditions could be as simple as the time interval or
  - could be more hidden factor which could be inferred through the neural network.
  - This ***automatic scheduling*** saves the system from faults caused by user intervention.

Thank You