

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

Submitted in partial fulfillment of the requirements  
of the degree of  
Master of Technology

*by*

Cynthia Josephine J

153050053

*Supervisor :*

Prof. Krithi Ramamirtham



Computer Science & Engineering  
Indian Institute of Technology, Bombay

June, 2017

## **Abstract**

Knowing the occupancy details is essential for building automation in order to maintain a robust way of automatic controlling of heating, lighting, ventilation, air conditioning as well as the security of the building under consideration. It is easy to get just the count of people occupying a given room by using a simple sensor-fused-network system but this knowledge of the count is not sufficient as the energy consumption of each occupant varies from another. In order to effectively satisfy each of the occupant's comfort level along with the motive of best achievable energy conservation, we need to recognize the behavior of every individual occupant, ie, the electricity requirement, the consumption pattern through the time of usage and even the exceptional or emergency usage of electricity and also understand the factors influencing them. Thus the first and foremost step is to recognize the individual occupying the room at given time, and then through other means, the electricity consumption pattern could be associated with that individual. The most simple, yet accurate way of identifying a person is through the recognition of the face along with other attributes like height and weight. In order to make this energy efficient, the camera feed should not be continuously processed. Instead, it should be processed only for few seconds when a person enters a particular room through triggering of the camera by simple sensor network which identifies the entry or exit of the person, making it more cost efficient, ultimately making the smart door. The other goal is to manage this smart door system automatically by not requiring user intervention during the system failure and to nail down the causes of the failures through the diagnostics without requiring additional hardware.

Approval sheet

This dissertation entitled **Occupancy Recognition through the Smart Door System - The Design and The Failure Diagnostics** by **Cynthia Josephine J (Roll Number: 153050053)** is approved for the degree of Master of Technology in Computer Science and Engineering from IIT Bombay.

**Examiners**

---

---

---

**Supervisor**

---

---

---

**Chairman**

---

Date: \_\_\_\_\_

Place: \_\_\_\_\_

### Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

---

(Signature)

---

(Name of the student)

---

(Roll No.)

Date: \_\_\_\_\_

# Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor **Prof. Krithi Ramamirtham** for the continuous support & encouragement in my MTech study and research. His guidance helped me all the time in my research and writing of this thesis. I could not have imagined having a better advisor and mentor for my MTech study.

Besides my advisor, I would like to thank **Prof. Kavi Arya** for his insightful comments and queries which has driven me to widen my research from various perspectives and also for letting me to deploy the Smart Door System in ERTS Laboratory, IIT Bombay.

I would also like to thank Mr. Piyush Manavar from ERTS Lab for his kind support during the deployment in the ERTS Laboratory, and Mr. Bhushan Kadam for helping me in clarification of doubts in hardware circuitry and Mr. Akshay Bhandare for helping in soldering the electronic circuits and also other SEIL Laboratory members for their support.

Last but not the least, I would like to thank my family and friends for supporting and motivating me throughout my life.

# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	Introduction . . . . .	7
1.2	Motivation . . . . .	7
1.3	Thesis Organisation . . . . .	7
<b>2</b>	<b>System Design</b>	<b>8</b>
2.1	Hardware Requirements . . . . .	8
2.2	Software Requirements . . . . .	9
2.3	System Design . . . . .	9
<b>3</b>	<b>Sensor Subsystems</b>	<b>12</b>
3.1	Laser Beam Sensors . . . . .	12
3.2	Ultrasonic Sensor . . . . .	14
3.3	Weight Mat Sensor . . . . .	15
<b>4</b>	<b>Sensor Control through Arduinos</b>	<b>17</b>
4.1	Laser Beam - Trigger subsystem . . . . .	17
4.1.1	The Pin Connections . . . . .	17
4.1.2	Working . . . . .	18
4.2	Height Weight Measurement Subsystem . . . . .	19
4.2.1	The Pin Connections . . . . .	19
4.2.2	Working . . . . .	20
<b>5</b>	<b>Raspberry Pi: Client System</b>	<b>21</b>
5.1	Network Settings . . . . .	21
5.1.1	Wifi Interface Setting . . . . .	21
5.1.2	Ethernet(LAN) Interface Setting . . . . .	22
5.2	Storing and Processing Sensor Readings . . . . .	22
5.2.1	Existing Implementation . . . . .	23
5.2.2	Updated Implementation . . . . .	24

<b>6</b>	<b>Image Processing of Camera Feed</b>	<b>26</b>
6.1	Camera Initial Setup . . . . .	26
6.2	Network Parameter Settings . . . . .	26
6.2.1	During the Camera Activation . . . . .	26
6.2.2	Online URL of previously set IP Address . . . . .	27
6.3	Image Parameters Settings . . . . .	28
<b>7</b>	<b>Prediction Processing Server</b>	<b>29</b>
7.1	Prediction of Person based on Height and Weight . . . . .	29
7.2	Prediction of Person based on Face Recognition . . . . .	29
7.3	Integration of two Modules . . . . .	32
7.3.1	Influence of the parameters . . . . .	33
7.4	Displaying prediction result through Web Portal . . . . .	34
7.5	Diagnostics of prediction algorithms' failures . . . . .	34
7.6	Training of the data . . . . .	34
<b>8</b>	<b>Conclusion &amp; Future Work</b>	<b>35</b>
8.1	Conclusion . . . . .	35
8.2	Future Work . . . . .	35
	<b>Appendix A Connection of Sensor to Arduino Color Code Convention</b>	<b>36</b>
	<b>Appendix B Database Schemas</b>	<b>39</b>
B.1	SmartDoor_PeopleEntryExitDetail . . . . .	39
B.2	SmartDoor_PeopleCount . . . . .	39
B.3	SmartDoor_Diagnostics . . . . .	40
B.4	SmartDoor_Face_Identity . . . . .	40
B.5	SmartDoor_FaceDataSetRecordedVideo . . . . .	40
B.6	SmartDoor_FaceDataSetRecentIndices . . . . .	41
B.7	SmartDoor_OnEnterRecordedVideo . . . . .	41
B.8	SmartDoor_Face_PredictionRank . . . . .	41
B.9	SmartDoor_HW_PredictionRank . . . . .	42

# List of Figures

2.1	Smart Door Sketch . . . . .	8
2.2	Smart Door Design . . . . .	10
3.1	Laser-Phototransistor Circuit . . . . .	12
3.2	Signal Conditioning Circuit . . . . .	13
3.3	Ultrasonic - HX711 Module . . . . .	14
3.4	Load Cell - HX711 Circuit . . . . .	15
3.5	Blueprint of the Weight Mat Sensor System . . . . .	16
6.1	Network Parameter Settings. . . . .	27
6.2	Image Parameter Settings. . . . .	28
7.1	Time_Out = 1.5 & Time_Out_Tolerance = 3 . . . . .	31
7.2	Time_Out = 1.5 & Time_Out_Tolerance = 5 . . . . .	32
7.3	Time_Out = 2 & Time_Out_Tolerance = 3 . . . . .	32
7.4	Time_Out = 2 & Time_Out_Tolerance = 5 . . . . .	33
7.5	Time_Out = 1.5 & Time_Out_Tolerance = 7 . . . . .	33
A.1	Laser Detector (Phototransistor) Circuit deployed in ERTS Lab . . . . .	36
A.2	Ultrasonic Sensor module deployed in ERTS Lab . . . . .	37
A.3	Weight Mat Sensor module deployed in ERTS Lab . . . . .	38
B.1	SmartDoor_PeopleEntryExitDetail . . . . .	39
B.2	SmartDoor_PeopleCount . . . . .	39
B.3	SmartDoor_Diagnostics . . . . .	40
B.4	SmartDoor_Face_Identity . . . . .	40
B.5	SmartDoor_FaceDataSetRecordedVideo . . . . .	40
B.6	SmartDoor_FaceDataSetRecentIndices . . . . .	41
B.7	SmartDoor_OnEnterRecordedVideo . . . . .	41
B.8	SmartDoor_Face_PredictionRank . . . . .	41
B.9	SmartDoor_HW_PredictionRank . . . . .	42



# Chapter 1

## Introduction

### 1.1 Introduction

Smart Door[1] is designed to know the identity of the person entering or exiting the room. It uses affordable sensors and uses intelligence to guess the identity through fusing different sensor values without user intervention. It is also essential for maintaining the smart door through implementing a self-diagnostic system without requiring any additional hardware just through the understanding how the modules interact with each other. Through integration of camera, the accuracy of the prediction of the identity has significantly increased.

### 1.2 Motivation

The main motivation of the project is to collect the information about the occupancy of the room along with individual identities so that these details could be further useful for Building Automation System to harness its full potent in conserving the energy along with serving every individual the comfort required.

### 1.3 Thesis Organisation

The chapter 2 depicts the overall design of the smart door, listing the hardware and the software requirements. The chapter 3 outlines the sensor circuits and modules. The chapter 4 explains the working of the hardware (sensor) subsystems and how the arduinos control the sensors to extract the correct values for predicting the identity of the person. The chapter 5 mentions the initial settings that are to be done for robust functioning of the Raspberry Pi and also states the implementation details of how the values are extracted from the arduino and sent to the server for prediction processing. The chapter 6 lists out the network and image processing parameter settings for right functioning of the camera. The chapter 7 describes the prediction algorithms used by the processing server in order to predict the identity of the person entering or exiting the room. The chapter 8 concludes the purpose of the project and drafts the ideas for future work.

## Chapter 2

# System Design

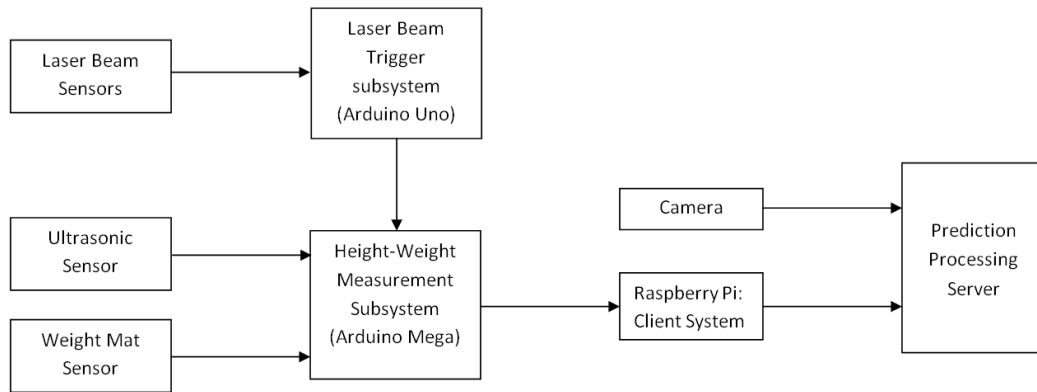


Figure 2.1: Smart Door Sketch

### 2.1 Hardware Requirements

- 650nm 6mm 5V DC 5mW Mini Laser Dot Diode Module - 2 units
- 3DU5C Metal Silicon Phototransistor Transistor - 2 units
- HC-SR04 Ultrasonic Distance Measuring Sensor Module for Arduino - 1 unit
- 50kg Load Cell Weighing Sensor - 4 units
- HX711 Weighing Sensor Dual-Channel 24 Bit Precision A/D weight Pressure Sensor - 1 unit
- HIKVISION DS-2CD1410F-IW(WI-FI) 1MP IR CUBE Camera - 1 unit
- Arduino Uno R3 ATmega328P - 1 unit
- Arduino Mega 2560 R3 Mega2560 REV3 + USB Cable - 1 unit
- Raspberry Pi 2 Model-B+ - 1 unit

## 2.2 Software Requirements

- Python
  - OpenCV
  - Scikit-learn
  - numpy
  - pandas
  - MySQLDb
- Microsoft SQL Database Server
- Arduino Software (IDE)
- SADP (V2.2.3.6) - Hikvision

## 2.3 System Design

The block diagram representing the smart door is shown in the Figure [2.1](#).

### Sensors

- Laser Beam Sensors
- Ultrasonic Sensor
- Weight Mat Sensor
- Camera

### Subsystem controlling Sensors

- Laser Beam - Trigger subsystem
- Height-Weight Measurement Subsystem

### Raspberry Pi: Client System

### Prediction Processing Server

From the previous implementation, a new module of Camera is added and the changes are made to Laser Beam Trigger Subsystem and Height-Weight Measurement Subsystem (adding an interrupt) and a Script in the Raspberry Pi: Client System and a Script in the Prediction Processing Server to integrate the face recognition (Image Processing Module).

The figure 2.2 shows the design of the Smart Door.

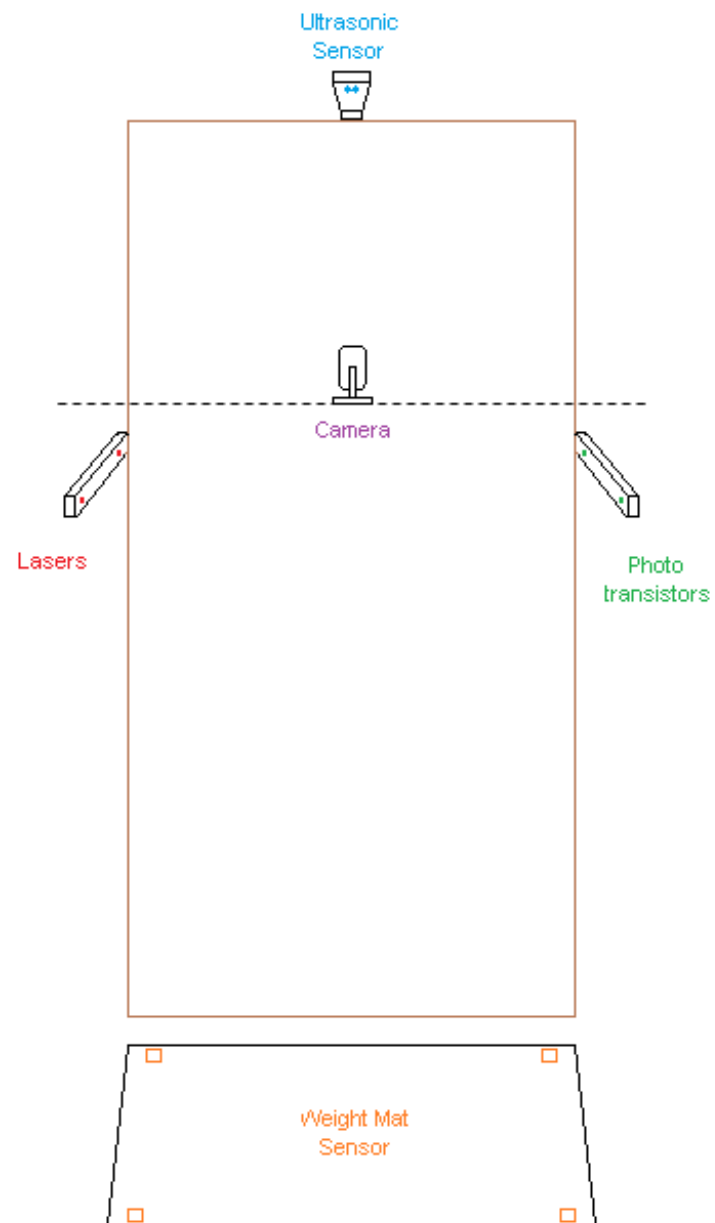


Figure 2.2: Smart Door Design

- When a person enters/ exits the smart door, the laser beam falling on the phototransistor gets obstructed, triggering the other sensors to function
  - On Enter, the laser 1 gets cut first and then the laser 2
  - On Exit, the laser 2 gets cut first and then the laser 1
- The ultrasonic sensor module mounted above the door measures the height of the person passing through the door.
- The weight mat sensor measures the weight of the person stepping on it.
- The camera mounted on the wall(represented by the dotted line) in some distance away from the smart door, records the video of the person entering the door for a short duration not exceeding 4 seconds.
- Using the height and weight values, a probability of person's identity is inferred.
- Using the face recognition algorithm, confidence factor of the person's identity is inferred.
- Combining these results the identity of the person is inferred.
- The person entering or exiting the door is asked to tag his name using the tablets(android) mounted near the door. (This is not shown in the diagram) This is done to self verify the identity.

## Chapter 3

# Sensor Subsystems

### 3.1 Laser Beam Sensors

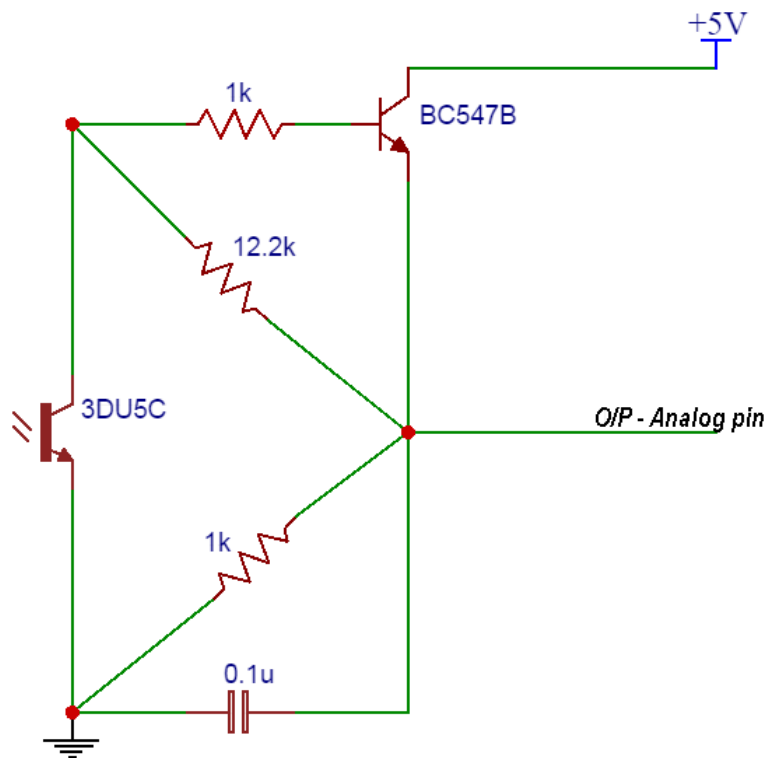


Figure 3.1: Laser-Phototransistor Circuit

- 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

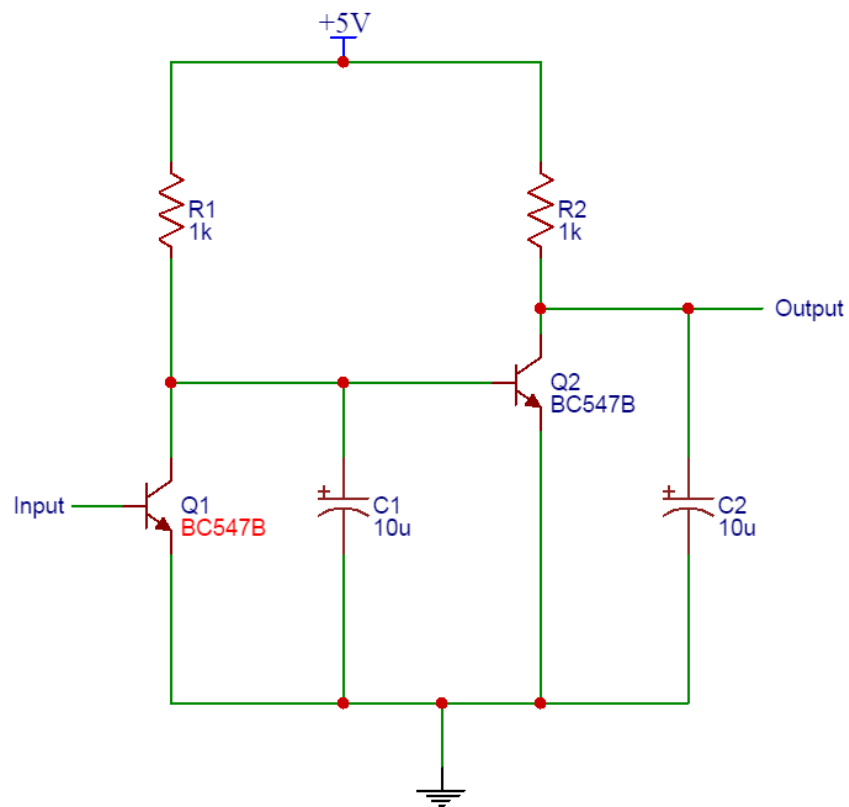


Figure 3.2: Signal Conditioning Circuit

- 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

## 3.2 Ultrasonic Sensor

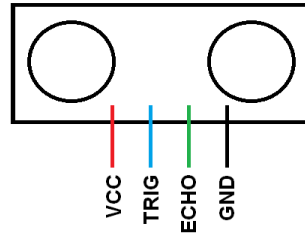


Figure 3.3: Ultrasonic - HX711 Module

The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object. Ranging Distance is from 2cm to 400 cm. It comes complete with ultrasonic transmitter and receiver 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. [2]

To obtain the distance, measure the width (Ton) of Echo pin.

Time = Width of Echo pulse, in uS (micro second)

- Distance in centimeters = Time / 58
- Distance in inches = Time / 148
- Or you can utilize the speed of sound, which is 340ms



### 3.3 Weight Mat Sensor

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.

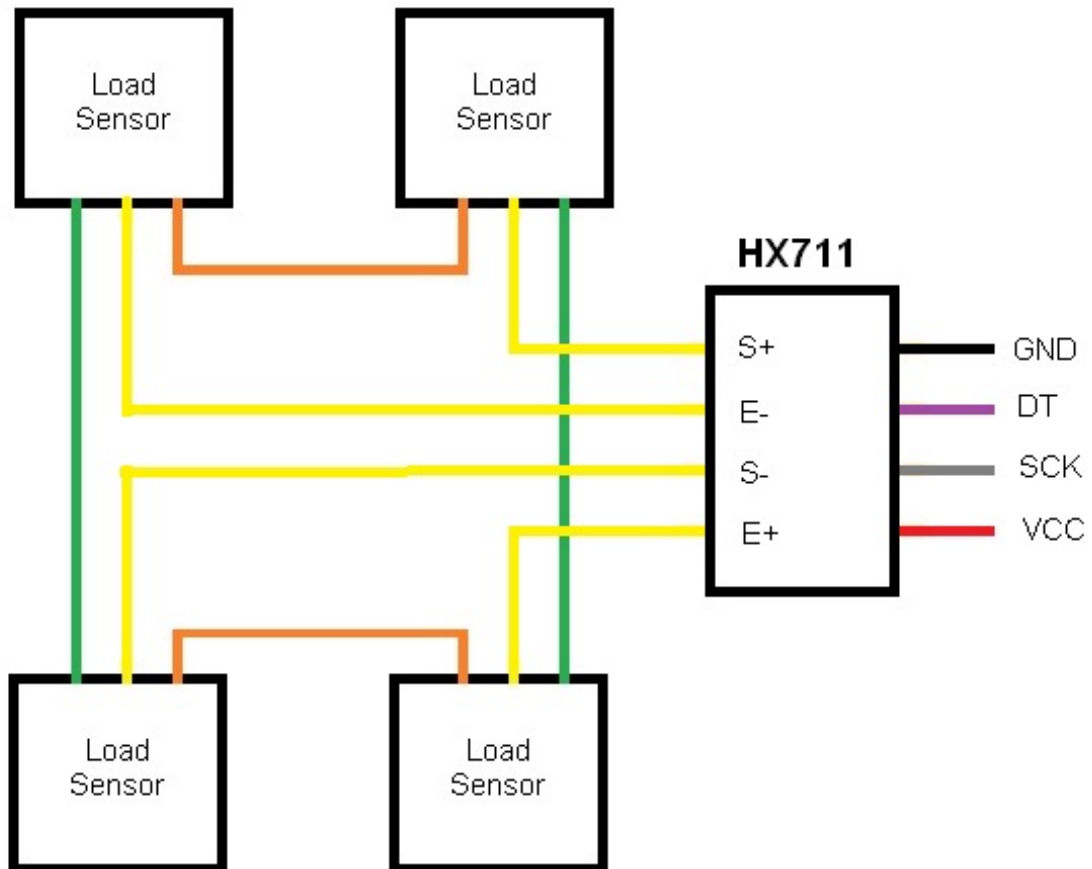


Figure 3.4: Load Cell - HX711 Circuit

#### Color Code of Load Cell

- Green - Negative
- Orange - Positive
- Yellow - Signal

Load cells use a four-wire Wheatstone bridge configuration [3] to connect to the HX711.

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

The figure 3.5 is the blueprint of how the Weight Mat Sensor System is made in the ERTS Lab. The orange boxes represents the placeholder for embedding the load cells.

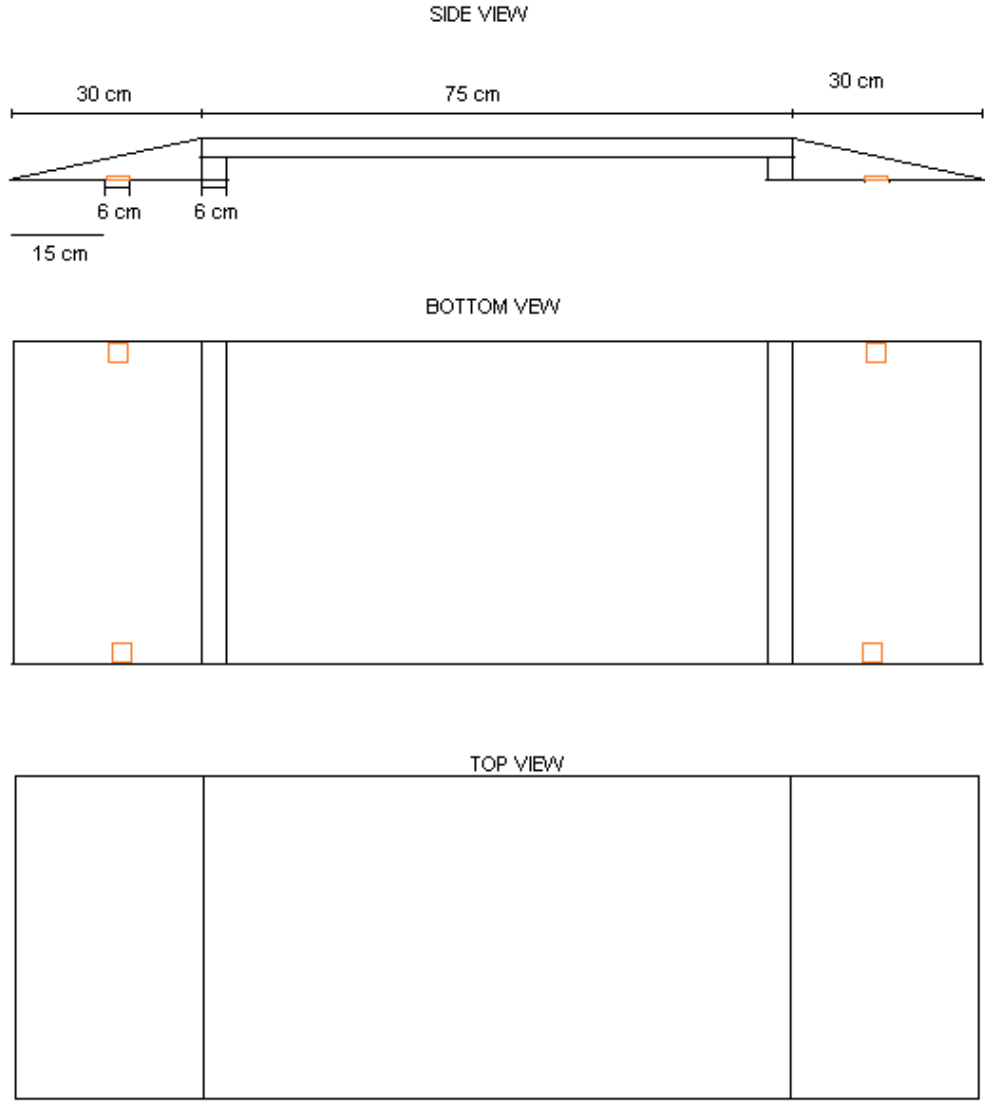


Figure 3.5: Blueprint of the Weight Mat Sensor System

## Chapter 4

# Sensor Control through Arduinos

### 4.1 Laser Beam - Trigger subsystem

Laser Beam Sensors are used to trigger the measuring of the height and the weight through the ultrasonic sensor and the weight mat sensor. This triggering subsystem is implemented by using an Arduino Uno which is microcontroller board based on the ATmega328P.

#### 4.1.1 The Pin Connections

**Input:**

- Laser 1 Pin is A0 (Analog Pin)
- Laser 2 Pin is A1 (Analog Pin)

**Output:**

- Signal to trigger the camera to start recording - Pin 7
- Signal to trigger the Height-Weight Measurement Subsystem to start measuring the height and the weight values - Pin 8
- Signal to trigger the Height-Weight Measurement Subsystem to send the recorded height and weight values - Pin 9
- Signal to trigger the Height-Weight Measurement Subsystem to reset the recorded height and the weight values - Pin 10
- Signal to indicate that a person has entered through the smart door - Pin 11
- Signal to indicate that a person has exited through the smart door - Pin 12
- LED Indicator - Pin 13

### 4.1.2 Working

- L1\_State signifies the state of the laser 1's beam falling on the phototransistor.
- L2\_State signifies the state of the laser 2's beam falling on the phototransistor.
- LX\_State will be HIGH if the laser is obstructed from falling on the phototransistor
- LX\_State will be LOW if the laser is falling on the phototransistor

In every Arduino cycle the following is done:

1. 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
2. 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.

There are 4 cases:

1. **Possible Entry:**

Laser 1 is cut first.

Do the following:

- 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.

2. **Possible Exit:**

Laser 2 is cut first.

Do the following:

- Send a signal to start recording the height and the weight measurements.

3. **Valid Entry:**

Laser 2 is cut first.

Do the following:

- 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

#### 4. Valid Exit:

Laser 1 is cut after Laser 2.

Do the following:

- 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\_IndicatorPin for short duration

This subsystem is connected to the Height Weight Measurement Subsystem.

## 4.2 Height Weight Measurement Subsystem

The ultrasonic sensor and the weight mat sensor are used to measure the height and the weight respectively. This Height Weight Measurement Subsystem is implemented by using an Arduino Mega which is microcontroller board based on the ATmega1280, HC-SR04 Ultrasonic Distance Measuring Sensor Module, four 50kg Load Cell Weighing Sensors and HX711 Weighing Sensor Dual-Channel 24 Bit Precision AD weight Pressure Sensor.

### 4.2.1 The Pin Connections

#### Input:

From HC-SR04 Ultrasonic Sensor Module

- ECHO Pin - Pin 13

From HX711 Weighing Sensor Dual-Channel 24 Bit Precision AD weight Pressure Sensor

- DOUT Pin is Pin A3

From Arduino Uno

- Signal to indicate that a person has entered through the smart door - Pin 3
- Signal to indicate that a person has exited through the smart door - Pin 4
- Signal to trigger the Height-Weight Measurement Subsystem to start measuring the height and the weight values - Pin 18
- Signal to trigger the Height-Weight Measurement Subsystem to send the recorded height and weight values - Pin 19
- Signal to trigger the Height-Weight Measurement Subsystem to reset the recorded height and the weight values - Pin 20
- Signal to trigger the camera to start recording - Pin 21

**Output:**

From HC-SR04 Ultrasonic Sensor Module

- TRIG Pin - Pin 12

From HX711 Weighing Sensor Dual-Channel 24 Bit Precision AD weight Pressure Sensor

- SCK Pin is Pin A2

#### 4.2.2 Working

**Interrupts Attached:**

- Signal to start measuring the height and the weight values - Pin 18 - Interrupt 5
- Signal to send the recorded height and weight values - Pin 19 - Interrupt 4
- Signal to reset the recorded height and the weight values - Pin 20 - Interrupt 3
- Signal to trigger the camera to start recording - Pin 21 - Interrupt 2

**Measurements - Function:**

- 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.

## Chapter 5

# Raspberry Pi: Client System

The Raspberry Pi gets the sensors readings from Arduino Mega via the Serial Port and stores those values in the Server Database and also triggers the Processing Server to fetch those values and to start the prediction of the person entering or exiting the room.

### 5.1 Network Settings

For the robust communication to the Server, the Raspberry Pi's IP address should be made static instead of DHCP services. The network connectivity could be achieved through either the ethernet(LAN) connection or the Wifi dongle.

#### 5.1.1 Wifi Interface Setting

This setup of using Wifi dongle for RPi communication is done in ERTS Lab, IIT Bombay.

The following steps are to be done: [\[4\]](#)

1. Edit **interfaces** file in the directory **/etc/network** using a simple editor called nano.

```
sudo nano /etc/network/interfaces
```

2. To set to the static IP Address, change the line for **wlan0** to

```
iface wlan0 inet static
```

For static IP settings, add lines for address, netmask, broadcast and gateway.

```
iface wlan0 inet static
```

```
    address 10.129.23.213
    netmask 255.255.128.0
    broadcast 10.129.255.255
    gateway 10.129.1.250
    wpa-ssid "SSID-FOR-WIFI"
    wpa-psk "Secret1*Pwd"
```

- The value of the IP address must be unique for the Raspberry Pi on the network.

- The value of the subnet mask must be the same for all devices on the network.
3. Save the changes and exit nano
  4. Reboot the Raspberry Pi.  
**sudo shutdown -r now**

### 5.1.2 Ethernet(LAN) Interface Setting

This setup of Ethernet(LAN) for RPi communication was done in SEIL Lab, IIT Bombay.

The following steps are to be done:

1. Edit **interfaces** file in the directory **/etc/network** using a simple editor called nano.  
**sudo nano /etc/network/interfaces**

2. To set to the static IP Address, change the line for **eth0** to

**iface eth0 inet static**

For static IP settings, add lines for address, netmask, broadcast and gateway.

**iface eth0 inet static**

```
address 10.129.23.214
netmask 255.255.128.0
broadcast 10.129.255.255
gateway 10.129.1.250
```

- The value of the IP address must be unique for the Raspberry Pi on the network.
  - The value of the subnet mask must be the same for all devices on the network.
3. Save the changes and exit nano
  4. Reboot the Raspberry Pi.  
**sudo shutdown -r now**

## 5.2 Storing and Processing Sensor Readings

The Raspberry Pi listens to the Arduino Mega through the serial port waiting for either

- the sensor readings of the height, the weight and the direction (Entry / Exit).
- the signal to start recording the camera feed.

In both the cases, the new session ID is sent to the Server activating it to start the prediction process through fetching of the above information which was stored in the database by a python script running in the Raspberry Pi.



### 5.2.1 Existing Implementation

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

#### Drawbacks

- As the connection to the Arduino Serial Port is done in the beginning, the client script will halt as soon as the Arduino gets physically disconnected from the Raspberry Pi.
- As the connection to the database is established in the beginning, the client script will halt if the database service is stopped or when there is poor internet connectivity.
- As the connection to the Server is established in the beginning, the client script will halt 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.

1. The Server Script needs to be stopped and then restarted again.
2. The client script is run to establish the connection again

Due to the above reasons, the smart door halts working whenever there is loss in internet connectivity and fails to work normally immediately after the network connectivity comes strong again and needs the user to restart the scripts as mentioned above.

### 5.2.2 Updated Implementation

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

#### Added Features

- On entry of a person, the Raspberry Pi instructs the Server to start recording the camera feed and process the video in order to make the prediction of people entering the lab even more accurate.
- The above mentioned drawbacks of previous implementation is rectified.

## Overcoming Drawbacks

- The client script will not halt even after physical disconnection of Arduino Serial Port and the script will keep on trying to reconnect until the connection to Arduino is established
- The client script will not halt when the database service is not running or when the database server is unreachable, rather it will wait until it starts running.
- The client script will not halt when server is unreachable due to temporary poor internet connectivity. The client script starts running normally as soon as the network connectivity is stable, without any user intervention of stopping and restarting the script.

## Chapter 6

# Image Processing of Camera Feed

### 6.1 Camera Initial Setup

The Hikvision camera[5] is being used for the video processing. In order to use the camera, the first and foremost thing is to activate the camera and set the username and password.

1. The camera is connected to the computer through ethernet cable
2. The SADP program scans the lists of IPs of devices connected to that network where the computer is connected.
3. Find the Camera's IP and activate it by giving new password.

### 6.2 Network Parameter Settings

#### 6.2.1 During the Camera Activation

This is simple and we just have to give the values of the following parameters before pressing activate button:

- IP Address
- Port
- Subnet
- Mask
- Gateway

Uncheck the option "Enable DHCP"

### 6.2.2 Online URL of previously set IP Address

1. Click on the "Configuration" in the menu bar.
2. Go the "Network" option in the left pane.
3. Select on the basic settings
4. Uncheck the DHCP checkbox.
5. Give the values of the following parameters:
  - IP Address
  - Port
  - Subnet
  - Mask
  - Gateway

Refer to the following diagram [6.1](#).

The screenshot displays the Hikvision web interface's Configuration section. The top navigation bar includes 'Live View', 'Playback', 'Picture', and 'Configuration' (highlighted in red). The left sidebar shows 'Local', 'System', and 'Network' (selected), with 'Basic Settings' highlighted under 'Network'. The main content area is titled 'TCP/IP' and contains the following settings:

- NIC Type:** Auto (dropdown)
- DHCP:** ☐ (unchecked)
- IPv4 Address:** 10.129.23.250 (text input) with a 'Test' button
- IPv4 Subnet Mask:** 255.255.0.0 (text input)
- IPv4 Default Gateway:** 10.129.1.250 (text input)
- IPv6 Mode:** Route Advertisement (dropdown) with a 'View Route Adv' button
- IPv6 Address:** (empty text input)
- IPv6 Subnet Mask:** (empty text input)
- IPv6 Default Gateway:** :: (text input)
- Mac Address:** a4:14:37:01:85:70 (text input)
- MTU:** 1500 (text input)
- Multicast Address:** (empty text input)
- Enable Multicast Discovery:** ☒ (checked)
- DNS Server:**
  - Preferred DNS Server:** 8.8.8.8 (text input)
  - Alternate DNS Server:** (empty text input)

A red 'Save' button is located at the bottom of the configuration area.

Figure 6.1: Network Parameter Settings.

### 6.3 Image Parameters Settings

The optimal values for effective image processing without network bottleneck is shown in the below figure 6.2:

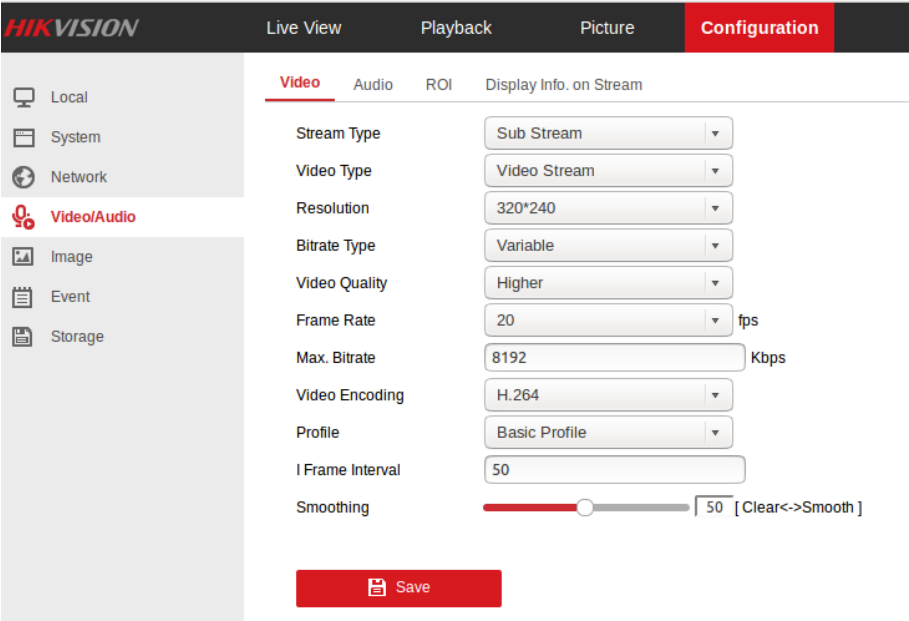


Figure 6.2: Image Parameter Settings.

## Chapter 7

# Prediction Processing Server

There are 2 scripts doing the process of predicting the person entering the room.

1. Prediction of Person based on Height and Weight
2. 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.

### 7.1 Prediction of Person based on Height and Weight

This prediction is already implemented previously. The algorithm used for prediction of the person based on the two parameters (height and weight) is Random Forest Classifier[6].

When a person is exiting the room, the algorithm used is just same as in the previous implementation. 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 for making better results as described in the section [7.3](#)

### 7.2 Prediction of Person based on Face Recognition

This is done only when the person enters the room, as there is only one camera placed facing the door.

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

1. 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).

2. For every frame in the camera video feed, do the following:
  - (a) Detect if there is any face present in that frame.
  - (b) If the face is detected, then recognize[8] the face in that frame using LPBH(Local Binary Patterns Histograms)[7] Algorithm [9].
  - (c) 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
  - (d) 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.
3. 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\*

\* - See the appendix for the Video Processing Algorithm parameters defined.
4. Calculate the total frames with face detected in the Camera Video Feed being recorded.
5. 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. Therefore,the 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) so as to get the optimal number of frames with faces.

This is done by the `Time_Out_Tolerance*` representing the number of loop cycles allowed after timeout.

The optimal value of the above 2 parameters could be found through trial and error such that it gives maximum number of frames for minimum recording time as in figures 7.1 & 7.2 & 7.3 & 7.4 & 7.5

6. 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

7. Store only the top 3 results based on the confidence factor.

```

1 SmartDoorCameraIP = "rtsp://10.129.23.250:554/ISAPI/streaming/channels/102?auth=YvRta46dGhLY2FtQWlLaWw="
2 TrainImageCameraIP = "H_frig_Test_1.avi"
3
4 CanConfig = ("Resolution": (320,240), "FrameRate(fps)": 20) #SubStream
5
6 #####
7 Frame_Door_Top_Left = 80
8 Frame_Door_Top_Right = 280
9 Frame_In_Left = 80
10 Frame_In_Right = 300
11 Max_Face_Frame_Area = 14000 #19500
12 Take_Shot = 0.5
13 Time_Out = 1.5
14 Time_Out_Tolerance = 3
15 #####

cynthia@cynthia-desktop:~/Documents/Sem4/SmartDoorFaceRecognizer/FaceRecognition
cynthia@cynthia-desktop:~/Documents/Sem4/SmartDoorFaceRecognizer/FaceRecognition
FaceRecognizer$ python Test_FR.py
init done
opencv 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$

```

Figure 7.1: `Time_Out = 1.5` & `Time_Out_Tolerance = 3`

Figure 7.2: Time\_Out = 1.5 & Time\_Out\_Tolerance = 5

Figure 7.3: Time\_Out = 2 & Time\_Out\_Tolerance = 3

## 7.3 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

```

1 SmartDoorCameraIP = "http://19.129.23.258:554/ISAP1/streaming/channels/102?auth=VwRtaM46dGhV2FtQWlkaWw="
2 TrainImageCameraIP = "M_Trig_Test_1.avi"
3
4 CanConfig = {"Resolution": (320,240), "FrameRate(fps)": 20} #SubStream
5
6 #####
7 Frame_Door_Top_Left = 60
8 Frame_Door_Top_Right = 280
9 Frame_In_Left = 60
10 Frame_In_Right = 300
11 Max_Face_Frame_Area = 14000 #19500
12 Take_Shot = 0.5
13 Time_Out = 2
14 Time_Out_Tolerance = 5
15 #####

```

```

cynthia@cynthia-desktop:~/Documents/Sem4/SmartDoorFaceRecognizer/FaceRecognition
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$

```

Figure 7.4: Time\_Out = 2 & Time\_Out\_Tolerance = 5

```

1 SmartDoorCameraIP = "http://19.129.23.258:554/ISAP1/streaming/channels/102?auth=VwRtaM46dGhV2FtQWlkaWw="
2 TrainImageCameraIP = "M_Trig_Test_1.avi"
3
4 CanConfig = {"Resolution": (320,240), "FrameRate(fps)": 20} #SubStream
5
6 #####
7 Frame_Door_Top_Left = 60
8 Frame_Door_Top_Right = 280
9 Frame_In_Left = 60
10 Frame_In_Right = 300
11 Max_Face_Frame_Area = 14000 #19500
12 Take_Shot = 0.5
13 Time_Out = 1.5
14 Time_Out_Tolerance = 7
15 #####

```

```

cynthia@cynthia-desktop:~/Documents/Sem4/SmartDoorFaceRecognizer/FaceRecognition
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$

```

Figure 7.5: Time\_Out = 1.5 & Time\_Out\_Tolerance = 7

### 7.3.1 Influence of the parameters

- Higher the probability based on Height and Weight, stronger the identity. Yet there would be many people with almost same height and weight. Hence, this parameter should be given only moderate priority or weightage.
- Higher the Confidence Factor, stronger the identity. This parameter is given the top priority.
- More the number of frames supporting the identity, higher the chance of that identity being correct
- The Minimum Frame Number has slight or indirect influence on the identity. This value represents the first frame in the video supporting this identity.
- The Maximum Frame Number has moderate influence on the identity. This value represents

the last frame in the video supporting this identity. Lower the value, lesser the chance of this identity being correct.

Based on these above parameters, the name of person entering is predicted and stored in the database.

## 7.4 Displaying prediction result through Web Portal

This module is already implemented and the only change that is made to this module is connecting it to the proper field of the record in the table showing the integrated predicted result.

The existing implementation is done by Flask python module and Ajax and simple CSS & HTML language.

The web portal is shown in the Figure ?? and lets the person entering or exiting to tag their identity. The predicted result is displayed to the user through blinking of the button and the result will be confirmed through tagging of the right name.

There is another new field entered in the database, "Scrutinized Name", which signifies the correct person entering or exiting the room. This field could be updated manually after checking the videos of people entering the room at the timestamp and this field is used for training the classifiers.

## 7.5 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.

If any of these fails, the details should be recorded locally (server itself here)

## 7.6 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.

## Chapter 8

# Conclusion & Future Work

### 8.1 Conclusion

Using the simple plug and play framework, it is robust to get the complete occupancy information, thereby aiding the building automation system to work efficiently managing the heating, lighting, ventilation, air conditioning as well as the security of the building under consideration. Adding the diagnostics module is added advantage reducing errors and guarantees automatic recovery in case of software errors.

### 8.2 Future Work

- In the algorithm specified in section 7.2, the parameter Confidence Factor could be given weightage according to the parameter Frame Number in the step 2c instead of considering them as two individual parameters. Say, 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 saves the system from faults caused by user intervention.

## Appendix A

# Connection of Sensor to Arduino Color Code Convention

For easier replication, new color code convention is used during deployment in ERTS Lab.

- Power Supply
  - VCC - Red
  - GND - Black
- Laser Sensor
  - Laser 1 - Yellow
  - Laser 2 - Orange

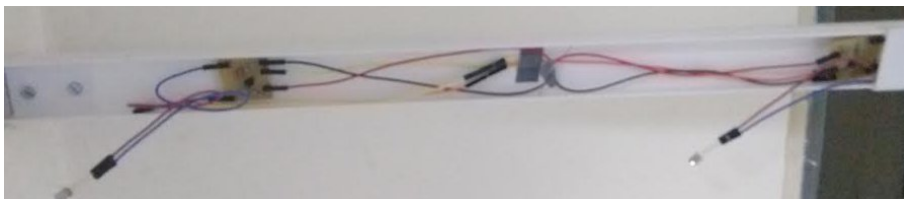


Figure A.1: Laser Detector (Phototransistor) Circuit deployed in ERTS Lab

- Ultrasonic Sensor
  - Echo - Green
  - Trig - Blue



Figure A.2: Ultrasonic Sensor module deployed in ERTS Lab

- Load Sensor
  - Positive(+) - Orange
  - Negative(-) - Green
  - Signal - Yellow
- Weight Mat Sensor
  - DT - Violet
  - SCK - White



Figure A.3: Weight Mat Sensor module deployed in ERTS Lab



## Appendix B

# Database Schemas

### B.1 SmartDoor\_PeopleEntryExitDetail

Name: SmartDoor_PeopleEntryExitDetail		Schema: datapool							
Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
SID	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
TimeOfEntryExit	TIMESTAMP	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CURRENT_TIMESTAMP
RoomID	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Direction	VARCHAR(5)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Height	DOUBLE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Weight	DOUBLE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Predicted_Name	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
Tagged_Name	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
Scrutinized_Name	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Figure B.1: SmartDoor\_PeopleEntryExitDetail

### B.2 SmartDoor\_PeopleCount

Name: SmartDoor_PeopleCount		Schema: datapool							
Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
SID	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
RoomID	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Count	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Figure B.2: SmartDoor\_PeopleCount

### B.3 SmartDoor\_Diagnostics

Name: SmartDoor_Diagnostics		Schema: datapool							
Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
FailureNo	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Failure TimeStamp	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Failure Event	LONGTEXT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Failure Attempts	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Figure B.3: SmartDoor\_Diagnostics

### B.4 SmartDoor\_Face\_Identity

Name: SmartDoor_Face_Identity		Schema: datapool							
Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
Identity	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Room_No	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Name	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	


Figure B.4: SmartDoor\_Face\_Identity

### B.5 SmartDoor\_FaceDataSetRecordedVideo

Name: SmartDoor_FaceDataSetRecordedVideo		Schema: datapool							
Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
RecVideoID	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
PersonID	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
RoomNo	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
VideoNo	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
StartimgIndex	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
EndimgIndex	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Figure B.5: SmartDoor\_FaceDataSetRecordedVideo

## B.6 SmartDoor\_FaceDataSetRecentIndices



Name:
SmartDoor\_FaceDataSetRecentIndices

Schema:
datapool






Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
 PersonID	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 RoomNo	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 RecentRecordedVideoNo	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 LastExtractedImageNo	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Figure B.6: SmartDoor\_FaceDataSetRecentIndices

## B.7 SmartDoor\_OnEnterRecordedVideo



Name:

Schema:










Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
 VideoID	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
 TimeOfEntering	TIMESTAMP	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CURRENT_TIMESTAMP
 VideoFileName	CHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 VideoTimeTaken	DECIMAL(10,0)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 TotalFaceFramesInVideo	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 ImageFileName	CHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 ImgPredictedID	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 ActualID	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Figure B.7: SmartDoor\_OnEnterRecordedVideo

## B.8 SmartDoor\_Face\_PredictionRank



Name:

SmartDoor\_Face\_PredictionRank

Schema:

datapool









Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
 SessionID	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 RankID	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 Name	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 ID	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 Confidence	DOUBLE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 Count	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 MinFrameNumber	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 MaxFrameNumber	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Figure B.8: SmartDoor\_Face\_PredictionRank

# B.9 SmartDoor\_HW\_PredictionRank

Name: SmartDoor_HW_PredictionRank		Schema: datapool							
Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
SessionID	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
RankID	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Name	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Probability	DOUBLE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Figure B.9: SmartDoor\_HW\_PredictionRank

# Bibliography

- [1] Nabeel Nasir, Kartik Palani, Amandeep Chugh, Vivek Chil Prakash, Uddhav Arote, Anand P. Krishnan, and Krithi Ramamritham  
*Fusing Sensors for Occupancy Sensing in Smart Buildings* .  
Department of Computer Science  
Indian Institute of Technology Bombay, Mumbai, India
- [2] Cytron Technologies Sdn. Bhd.  
[https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL\\_pfa39RsB-x2qR4vP8saG73rE/edit](https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL_pfa39RsB-x2qR4vP8saG73rE/edit) .  
Product User's Manual – HC-SR04 Ultrasonic Sensor
- [3] Contributors: Sarah Al-Mutlaq, Alex the Giant  
Load Cell Amplifier HX711 Breakout Hookup Guide  
<https://learn.sparkfun.com/tutorials/load-cell-amplifier-hx711-breakout-hookup-guide>.
- [4] P.Marion; Wireless and eth0 Static IP on Raspberry PI with Wi-Pi  
<http://www.electroschematics.com/9496/static-manual-ip-wireless-raspberry-pi>.
- [5] Hikvision Network Camera User Manual  
<http://www.hikvision.com/UploadFile/image2014101515581083809.pdf>.
- [6] Tin Kam Ho, AT&T Bell Laboratories, NJ, USA *Random Decision Forests* .
- [7] Shireesha Chintalapati, M.V. Raghunadh.  
*Automated Attendance Management System Based On Face Recognition Algorithms* .  
NIT Warangal,INDIA.
- [8] [http://docs.opencv.org/2.4/modules/contrib/doc/face\\_recognizer/tutorial.html#local-binary-patterns-histograms](http://docs.opencv.org/2.4/modules/contrib/doc/face_recognizer/tutorial.html#local-binary-patterns-histograms) .
- [9] Md. Abdur Rahim, Md. Najmul Hossain, Tanzillah Wahid & Md. Shafiul Azam  
*Face Recognition using Local Binary Patterns (LBP)* . Global Journal of Computer Science and Technology Graphics & Vision  
Volume 13 Issue 4 Version 1.0 Year 2013.