

Comp 311 – Digital system Design

**Smart Door Security System**

by

**Hadi Mostafa|202303776**

**Ahmad Qasem|202303622**

Spring 2025

## 1. Overview

This project is a Smart Door Security System that combines face recognition, ultrasonic sensors, Telegram bot control, and MQTT communication to provide a secure and automated door access system. The system can:

* Detect visitors using an ultrasonic sensor.
* Recognize authorized users (like "Hadi") using face recognition (Python + OpenCV).
* Send real-time alerts to the owner via Telegram.
* Allow remote door control (open/deny access).
* Provide physical button access for recognized users.
* Use RGB LEDs and a buzzer for visual/audio feedback.

## 2. Components Used

### Hardware Components

|  |  |
| --- | --- |
| **Component** | **Usage** |
| ESP8266 (NodeMCU) | Main microcontroller for WiFi & control logic |
| Ultrasonic Sensor (HC-SR04) | Detects visitors near the door |
| Servo Motor (SG90) | Simulates door lock/unlock mechanism |
| RGB LED | Visual feedback (blue=standby, green=access granted, red=denied) |
| Buzzer | Audio alerts for access granted/denied |
| Push Button | Manual override for recognized users |

### Software & Libraries

|  |  |
| --- | --- |
| **Component** | **Purpose** |
| Arduino IDE (C++) | ESP8266 firmware |
| Python (OpenCV, face\_recognition) | Face detection & recognition |
| Telegram Bot API | Remote notifications & control |
| MQTT (HiveMQ Broker) | Communication between Python & ESP8266 |

## 3. How the Project Works

### System Workflow

1. **Visitor Detection (Ultrasonic Sensor)**
   * If someone is within 50cm, the ESP8266 triggers an alert.
   * Sends a Telegram notification: "Visitor detected! Reply 'yes' or 'no'."
   * RGB LED turns orange (waiting for response).
2. **Owner Response (Telegram Bot)**
   * Owner can reply:
     + "yes" → Door opens for 5 seconds (servo moves).
     + "no" → Access denied (RGB turns red, buzzer sounds).
3. **Face Recognition Mode-OWNER MODE (Python Script)**
   * If "Hadi" is detected:
     + Python sends an MQTT "open\_door" command.
     + ESP8266 activates green LED + buzzer and waits for button press.
     + Pressing the button opens the door.
4. **Manual Override (Button Press)**
   * Recognized users can press the button to open the door.
5. **System Reset**
   * If no response within 30 seconds, the system resets.

## 4. The Code

### ESP8266 Code (Arduino C++)

1. **Library Imports and Initialization**

#include <ESP8266WiFi.h>

#include <WiFiClientSecure.h>

#include <UniversalTelegramBot.h>

#include <PubSubClient.h>

#include <Servo.h>

### 

### Hardware Pin Definitions

#define TRIG\_PIN D1

#define ECHO\_PIN D2

#define SERVO\_PIN D3

#define BUTTON\_PIN D8

**3. Network Configuration**

const char\* ssid = "DORMS";

const char\* password = "B@uDorms2024";

#define BOT\_TOKEN "7492296974:AAEdCpH2j0\_R8ZkvWhZW0PqZ1hLY7N7R\_5M"

#define CHAT\_ID "7844226118"

const char\* mqtt\_server = "broker.hivemq.com";

const char\* mqtt\_topic = "hadi\_door\_control";

* **Contains network credentials and service configurations:**
* WiFi SSID and password
* Telegram bot token and chat ID
* MQTT broker server and topic

**4. Object Instantiations**

WiFiClientSecure secured\_client;

UniversalTelegramBot bot(BOT\_TOKEN, secured\_client);

PubSubClient mqttClient(espClient);

Servo doorServo;

* **Creates instances of:**
* Secure WiFi client
* Telegram bot handler
* MQTT client
* Servo controller

**5. State Variables**

bool waitingResponse = false;

bool visitorDetected = false;

bool hadiDetected = false;

* **Tracks system states:**
* Whether waiting for owner response
* Visitor detection status
* Recognized user detection

**6. Setup Function**

void setup() {

Serial.begin(115200);

pinMode(TRIG\_PIN, OUTPUT);

pinMode(ECHO\_PIN, INPUT);

doorServo.attach(SERVO\_PIN);

pinMode(BUTTON\_PIN, INPUT\_PULLUP);

connectToWiFi();

mqttClient.setServer(mqtt\_server, 1883);

mqttClient.setCallback(mqttCallback);

}

* **Initializes:**
* Serial communication
* Pin modes for all components
* WiFi connection
* MQTT client configuration

**7. Main Loop**

void loop() {

mqttClient.loop();

checkUltrasonicSensor();

checkButton();

if (waitingResponse && (millis() - detectionStartTime > 30000)) {

handleTimeout();

}

}

* **Core operational cycle:**
* Processes MQTT messages
* Checks ultrasonic sensor
* Monitors button state
* Handles response timeout

**8. MQTT Callback Function**

void mqttCallback(char\* topic, byte\* payload, unsigned int length) {

String message;

for (int i = 0; i < length; i++) message += (char)payload[i];

if (message == "open\_door") {

hadiDetected = true;

bot.sendMessage(CHAT\_ID, "🟢 Hadi detected! Press the button.", "");

}

}

* **Handles incoming MQTT messages:**
* Receives and parses messages
* Triggers recognized user protocol
* Sends Telegram notification

**9. Door Control Function**

void openDoorForHadi() {

doorServo.write(90);

delay(5000);

doorServo.write(0);

hadiDetected = false;

}

* **Manages door operation:**
  + Opens door (servo to 90°)
  + Holds open for 5 seconds
  + Closes door (servo to 0°)
  + Resets detection flag

### Python Face Recognition Code

import cv2

import face\_recognition

import paho.mqtt.publish as mqtt

* **Library Imports:**
  + cv2 (OpenCV) for camera capture and image processing
  + face\_recognition for facial detection and recognition
  + paho.mqtt.publish for sending MQTT messages to the ESP8266

hadi\_image = face\_recognition.load\_image\_file("hadi.jpg")

hadi\_encoding = face\_recognition.face\_encodings(hadi\_image)[0]

* **Known Face Setup:**
  + Loads reference image ("hadi.jpg") of authorized person
  + Extracts facial features into numerical encoding for comparison

cap = cv2.VideoCapture(0)

* **Camera Initialization:**
  + Sets up video capture from default camera (index 0)

while True:

ret, frame = cap.read()

rgb\_frame = frame[:, :, ::-1]

* **Main Loop:**
  + cap.read() grabs each video frame
  + Converts BGR format (OpenCV default) to RGB (required by face\_recognition)

face\_locations = face\_recognition.face\_locations(rgb\_frame)

face\_encodings = face\_recognition.face\_encodings(rgb\_frame, face\_locations)

* **Face Processing:**
  + Detects all faces in current frame
  + Generates encodings (mathematical representations) for each detected face

for face\_encoding in face\_encodings:

match = face\_recognition.compare\_faces([hadi\_encoding], face\_encoding)

if match[0]:

mqtt.single("hadi\_door\_control", "open\_door", hostname="broker.hivemq.com")

print("Hadi detected! Door unlocked.")

* **Recognition Logic:**
  + Compares each detected face against stored encoding
  + If match found:
  + Sends "open\_door" command via MQTT
  + Prints confirmation message

cv2.imshow('Face Recognition', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

* **Display and Controls:**
  + Shows live video feed with detection results
  + Exits loop when 'q' key is pressed

cap.release()

cv2.destroyAllWindows()

* **Cleanup:**
  + Releases camera resources
  + Closes all OpenCV windows

## 5. Conclusion

This project successfully integrates:

* ✔ Face recognition for secure access.
* ✔ Telegram bot for remote control.
* ✔ MQTT for real-time communication.
* ✔ Ultrasonic sensor for visitor detection.
* **Future improvements:**
* Add multiple user recognition.
* Implement voice control (Alexa/Google Assistant).
* Use IoT cloud logging for access history.