SOFTWARE CODE

BATCH 14

# First ESP 32:

#define BLYNK\_TEMPLATE\_ID "TMPL3saN6O6C6"

#define BLYNK\_TEMPLATE\_NAME "MCA"

#define BLYNK\_AUTH\_TOKEN "rRztLQ8SnxPTTElJwlOlDN9CCTvc2W2Q"

#include <Keypad.h>

#include <HardwareSerial.h>

#include <DHT.h>

#include <WiFi.h>

#include <BlynkSimpleEsp32.h>

// WiFi Credentials

char ssid[] = "Naveen"; // Replace with your WiFi SSID

char pass[] = "Naveen0528"; // Replace with your WiFi Password

#define DHTPIN 18

#define PIRPIN 19

#define DHTTYPE DHT11

#define DE\_RE 4

#define BUZZER\_PIN 23

#define ROW\_NUM 4

#define COLUMN\_NUM 4

char keys[ROW\_NUM][COLUMN\_NUM] = {

{'1', '2', '3', 'A'},

{'4', '5', '6', 'B'},

{'7', '8', '9', 'C'},

{'\*', '0', '#', 'D'}

};

byte pin\_rows[ROW\_NUM] = {13, 12, 14, 27};

byte pin\_column[COLUMN\_NUM] = {26, 25, 33, 32};

Keypad keypad = Keypad(makeKeymap(keys), pin\_rows, pin\_column, ROW\_NUM, COLUMN\_NUM);

String enteredPin = "";

const String correctPin = "1234";

bool accessGranted = false;

bool wrongPinEntered = false;

HardwareSerial RS485Serial(2);

DHT dht(DHTPIN, DHTTYPE);

void setup() {

Serial.begin(115200);

delay(10);

// Connect to WiFi

WiFi.begin(ssid, pass);

// Wait for connection

int attempts = 0;

while (WiFi.status() != WL\_CONNECTED && attempts < 20) {

delay(1000);

Serial.print(".");

attempts++;

}

if (WiFi.status() == WL\_CONNECTED) {

Serial.println("\nConnected to WiFi!");

Serial.print("IP Address: ");

Serial.println(WiFi.localIP());

} else {

Serial.println("\nWiFi connection failed. Please check credentials.");

return;

}

// Start Blynk

Blynk.begin(BLYNK\_AUTH\_TOKEN, ssid, pass);

RS485Serial.begin(9600, SERIAL\_8N1, 16, 17);

pinMode(BUZZER\_PIN, OUTPUT);

digitalWrite(BUZZER\_PIN, HIGH); // Ensure buzzer is OFF initially

pinMode(PIRPIN, INPUT);

pinMode(DE\_RE, OUTPUT);

digitalWrite(DE\_RE, LOW); // Start in RX mode

delay(2000); // Give time for DHT sensor to stabilize

dht.begin();

Serial.println("Please enter PIN:");

}

void loop() {

Blynk.run(); // Keep Blynk connected

char key = keypad.getKey();

if (key) {

Serial.print(key);

if (accessGranted) {

return;

}

if (key == '\*') {

enteredPin = "";

Serial.println("\nPlease enter PIN:");

wrongPinEntered = false; // Reset flag

digitalWrite(BUZZER\_PIN, HIGH); // Turn OFF buzzer when reset

} else {

enteredPin += key;

}

if (enteredPin.length() == 4) {

if (enteredPin == correctPin) {

Serial.println("\nAccess Granted");

digitalWrite(BUZZER\_PIN, HIGH); // Turn OFF buzzer

accessGranted = true;

enteredPin = "";

Blynk.logEvent("correct\_pin", "Correct PIN entered, access granted.");

sendSensorData();

} else {

Serial.println("\nAccess Denied");

wrongPinEntered = true; // Set flag

digitalWrite(BUZZER\_PIN, LOW); // Keep buzzer ON

// Send email notification via Blynk

Blynk.logEvent("wrong\_pin", "Wrong PIN entered on Keypad!");

enteredPin = "";

Serial.println("Press '\*' to try again.");

}

}

}

if (wrongPinEntered) {

digitalWrite(BUZZER\_PIN, LOW); // Keep buzzer ON if wrong PIN was entered

}

}

void sendSensorData() {

while (accessGranted) {

bool motion = digitalRead(PIRPIN);

float temperature = dht.readTemperature();

if (isnan(temperature)) {

Serial.println("Error: Failed to read from DHT sensor!");

temperature = -1;

}

Serial.print("Motion: ");

Serial.print(motion ? "Detected" : "Not Detected");

Serial.print(", Temperature: ");

Serial.println(temperature);

digitalWrite(DE\_RE, HIGH);

delay(10);

RS485Serial.print(motion);

RS485Serial.print(",");

RS485Serial.print(temperature);

RS485Serial.print("\n");

delay(10);

digitalWrite(DE\_RE, LOW);

delay(5000); // Send data every 5 seconds

}

}

# SECOND ESP32:

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

#include <HardwareSerial.h>

#include <SPI.h>

#include <MFRC522.h>

#include <ESP32Servo.h>

#define SCREEN\_WIDTH 128

#define SCREEN\_HEIGHT 64

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, -1);

#define RELAY1 26

#define RELAY2 25

#define DE\_RE 4

#define SS\_PIN 5 // RFID SDA pin

#define RST\_PIN 15 // Changed RST pin to avoid conflict with OLED

#define SERVO\_PIN 27 // Servo connected to GPIO 27

MFRC522 mfrc522(SS\_PIN, RST\_PIN);

Servo doorServo;

HardwareSerial RS485Serial(2);

unsigned long lastReceiveTime = 0;

const unsigned long timeoutPeriod = 10000;

bool receivedDataFlag = false;

// RFID card UIDs (Replace with your actual UIDs)

byte entryCardUID[] = {0xF3, 0xF4, 0xAD, 0x30}; // Entry card

byte exitCardUID[] = {0x03, 0x43, 0x9D, 0xFA}; // Exit card

void setup() {

Serial.begin(115200);

RS485Serial.begin(9600, SERIAL\_8N1, 16, 17);

SPI.begin();

mfrc522.PCD\_Init();

doorServo.attach(SERVO\_PIN);

doorServo.write(0); // Start with the door closed

Serial.println("RFID Door Access System Ready");

pinMode(DE\_RE, OUTPUT);

digitalWrite(DE\_RE, LOW);

pinMode(RELAY1, OUTPUT);

pinMode(RELAY2, OUTPUT);

digitalWrite(RELAY1, HIGH);

digitalWrite(RELAY2, HIGH);

if (!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C)) {

Serial.println("OLED init failed");

while (1);

}

display.clearDisplay();

}

void loop() {

if (RS485Serial.available()) {

String receivedData = RS485Serial.readStringUntil('\n');

receivedData.trim();

lastReceiveTime = millis();

receivedDataFlag = true;

int commaIndex = receivedData.indexOf(",");

if (commaIndex > 0) {

int motion = receivedData.substring(0, commaIndex).toInt();

float temperature = receivedData.substring(commaIndex + 1).toFloat();

Serial.print("Motion: ");

Serial.print(motion ? "Detected" : "Not Detected");

Serial.print(", Temperature: ");

Serial.println(temperature);

displayStatus(motion, temperature, "Car Parking Closed");

digitalWrite(RELAY1, motion ? LOW : HIGH);

digitalWrite(RELAY2, (temperature > 28) ? LOW : HIGH);

}

}

if (!receivedDataFlag || (millis() - lastReceiveTime > timeoutPeriod)) {

display.clearDisplay();

display.setTextSize(2);

display.setTextColor(WHITE);

display.setCursor(10, 20);

display.println("SECURITY");

display.println("BREACH!");

display.display();

}

if (!mfrc522.PICC\_IsNewCardPresent() || !mfrc522.PICC\_ReadCardSerial()) {

return;

}

Serial.print("Card UID: ");

for (byte i = 0; i < mfrc522.uid.size; i++) {

Serial.print(mfrc522.uid.uidByte[i], HEX);

Serial.print(" ");

}

Serial.println();

if (checkUID(mfrc522.uid.uidByte, entryCardUID)) {

Serial.println("Access Granted! Opening Door...");

smoothServoOpen();

displayStatus(true, 0.0, "Car Parking Open");

}

else if (checkUID(mfrc522.uid.uidByte, exitCardUID)) {

Serial.println("Exit Detected! Closing Door...");

doorServo.write(0);

displayStatus(false, 0.0, "Car Parking Closed");

delay(5000);

sendResetSignal();

}

else {

Serial.println("Unknown Card!");

}

mfrc522.PICC\_HaltA();

mfrc522.PCD\_StopCrypto1();

}

bool checkUID(byte scannedUID[], byte storedUID[]) {

for (byte i = 0; i < 4; i++) {

if (scannedUID[i] != storedUID[i]) {

return false;

}

}

return true;

}

void smoothServoOpen() {

for (int pos = 0; pos <= 180; pos += 5) {

doorServo.write(pos);

delay(30);

}

}

void displayStatus(bool motion, float temperature, const char\* parkingStatus) {

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(0, 10);

display.print("Presence: ");

display.println(motion ? "Yes" : "No");

display.print("Temperature: ");

display.print(temperature);

display.println(" C");

display.print("Status: ");

display.println(parkingStatus);

display.display();

}

void sendResetSignal() {

digitalWrite(DE\_RE, HIGH);

delay(10);

RS485Serial.println("RESET");

delay(10);

digitalWrite(DE\_RE, LOW);

}

# IOT Automation:

#include "BluetoothSerial.h"

#include <ESP32Servo.h>

BluetoothSerial SerialBT;

const int FAN\_PIN = 25;

const int LIGHT\_PIN = 26;

const int SERVO\_PIN = 27;

Servo garageDoorServo;

void setup() {

Serial.begin(115200);

SerialBT.begin("ESP32\_BT\_Control");

pinMode(FAN\_PIN, OUTPUT);

pinMode(LIGHT\_PIN, OUTPUT);

garageDoorServo.attach(SERVO\_PIN);

garageDoorServo.write(0);

Serial.println("ESP32 Bluetooth Control Ready...");

}

void loop() {

if (SerialBT.available()) {

String command = SerialBT.readStringUntil('\n');

command.trim();

Serial.print("Received Command: ");

Serial.println(command);

if (command.equalsIgnoreCase("ON1") || command.equalsIgnoreCase("fan on")) {

digitalWrite(FAN\_PIN, LOW);

SerialBT.println("Fan is ON");

}

else if (command.equalsIgnoreCase("OFF1") || command.equalsIgnoreCase("fan of")) {

digitalWrite(FAN\_PIN, HIGH);

SerialBT.println("Fan is OFF");

}

else if (command.equalsIgnoreCase("ON2") || command.equalsIgnoreCase("light on")) {

digitalWrite(LIGHT\_PIN, LOW);

SerialBT.println("Light is ON");

}

else if (command.equalsIgnoreCase("OFF2") || command.equalsIgnoreCase("light of")) {

digitalWrite(LIGHT\_PIN,HIGH);

SerialBT.println("Light is OFF");

}

else if (command.equalsIgnoreCase("ON3") || command.equalsIgnoreCase("garage open")) {

garageDoorServo.write(90);

SerialBT.println("Garage door OPENED");

}

else if (command.equalsIgnoreCase("OFF3") || command.equalsIgnoreCase("garage close")) {

garageDoorServo.write(0);

SerialBT.println("Garage door CLOSED");

}

else {

SerialBT.println("Unknown Command");

}

}

}