ALERT MESSAGE

// Potentiometer is connected to GPIO 34 (Analog ADC1\_CH6)

const int potPin = 34;

// variable for storing the potentiometer value int

potValue = 0;

void setup() { Serial.begin(115200);

delay(1000);

}

void loop() {

// Reading potentiometer value

potValue = analogRead(potPin); if

(potValue>2000)

{Serial.println("warning");}

Serial.println(potValue); delay(500);

}

PIR SENSOR

// PIR Sensor interfacing with Arduino

int pirPin = 2; // PIR sensor output pin

int ledPin = 13; // LED pin

void setup() {

pinMode(pirPin, INPUT); // Set PIR pin as input

pinMode(ledPin, OUTPUT); // Set LED pin as output

Serial.begin(9600); //Initialize serial communication for debugging

}

void loop() {

int motionState = digitalRead(pirPin); // Read PIR sensor state

if (motionState == HIGH) {

digitalWrite(ledPin, HIGH); // Turn on LED

Serial.println("Motion detected!");

delay(1000); // Delay for one second

} else {

digitalWrite(ledPin, LOW); // Turn off LED

Serial.println("No motion detected.");

}

}

Database

#include <WiFi.h>

#include <WebServer.h>

const char \*ssid = "AndroidAP1F94";

const char \*password = "shilpa19";

const int potPin = 34;

int potValue = 0;

WebServerserver(80);

void handleRoot() {

String html = "<html><body>";

html += "<h1>Potentiometer Value:</h1>";

html += "<p>" + String(potValue) + "</p>";

html += "</body></html>";

server.send(200, "text/html", html);

}

void setup() {

Serial.begin(115200);

pinMode(potPin, INPUT);

// Connect to Wi-Fi

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.println("Connecting to WiFi...");

}

Serial.println("Connected to WiFi");

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

// Handle root URL

server.on("/", HTTP\_GET, handleRoot);

// Start server

server.begin();

}

void loop() {

// Read potentiometer value

potValue = analogRead(potPin);

server.handleClient();// Handle incoming HTTP requests

delay(100); // Adjust delay as needed

}

CPS SYSTEM

// Load Wi-Fi library

#include <WiFi.h>

// Replace with your network credentials

const char\* ssid ="Shilpas";

const char\* password = "shilpa19";

// Set web server port number to 80

WiFiServer server(80);

// Variable to store the HTTP request

String header;

// Auxiliary variables to store the current output state

String output12State = "off";

String output14State = "off";

// Assign output variables to GPIO pins

const int output12 = 12;

const int output14 = 14;

// Current time

unsigned long currentTime = millis();

// Previous time

unsigned long previousTime = 0;

// Define timeout time in milliseconds (example: 2000ms = 2s)

const long timeoutTime = 2000;

void setup() {

Serial.begin(115200);

// Initialize the output variables as outputs

pinMode(output12, OUTPUT);

pinMode(output14, OUTPUT);

// Set outputs to LOW

digitalWrite(output12, LOW);

digitalWrite(output14, LOW);

// Connect to Wi-Fi network with SSID and password

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

// Print local IP address and start web server

Serial.println("");

Serial.println("WiFi connected.");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

server.begin();

}

void loop(){

WiFiClient client = server.available(); // Listen for incoming clients

if (client) { // If a new client connects,

currentTime = millis();

previousTime = currentTime;

Serial.println("New Client."); // print a message out in the serial

port

String currentLine = ""; // make a String to hold incoming

data from the client

while (client.connected() && currentTime - previousTime <= timeoutTime) { //

loop while the client's connected

currentTime = millis();

if (client.available()) { // if there's bytes to read from the

client,

row.

OK)

line:

char c = client.read(); // read a byte, then

Serial.write(c); // print it out the serial monitor

header += c;

if (c == '\n') { // if the byte is a newline character

// if the current line is blank, you got two newline characters in a

// that's the end of the client HTTP request, so send a response:

if (currentLine.length() == 0) {

// HTTP headers always start with a response code (e.g. HTTP/1.1 200

// and a content-type so the client knows what's coming, then a blank

client.println("HTTP/1.1 200 OK");

client.println("Content-type:text/html");

client.println("Connection: close");

client.println();

// turns the GPIOs on and off

if (header.indexOf("GET /12/on") >= 0) {

Serial.println("GPIO 12 on");

output12State = "on";

digitalWrite(output12, HIGH);

} else if (header.indexOf("GET /12/off") >= 0) {

Serial.println("GPIO 12 off");

output12State = "off";

digitalWrite(output12, LOW);

} else if (header.indexOf("GET /14/on") >= 0) {

Serial.println("GPIO 14 on");

output14State = "on";

digitalWrite(output14, HIGH);

} else if (header.indexOf("GET /14/off") >= 0) {

Serial.println("GPIO 14 off");

output14State = "off";

digitalWrite(output14, LOW);

}

// Display the HTML web page

client.println("<!DOCTYPE html><html>");

client.println("<head><meta name=\"viewport\" content=\"width=device-

width, initial-scale=1\">");

client.println("<link rel=\"icon\" href=\"data:,\">");

// CSS to style the on/off buttons

// Feel free to change the background-color and font-size attributes

to fit your preferences

client.println("<style>html { font-family: Helvetica; display:

inline-block; margin: 0px auto; text-align: center;}");

client.println(".button { background-color: #4CAF50; border: none;

color: white; padding: 16px 40px;");

client.println("text-decoration: none; font-size: 30px; margin: 2px;

cursor: pointer;}");

client.println(".button2 {background-color:

#555555;}</style></head>");

// Web Page Heading

client.println("<body><h1>ESP32 Web Server</h1>");

// Display current state, and ON/OFF buttons for GPIO 12

client.println("<p>GPIO 12 - State " + output12State + "</p>");

// If the output12State is off, it displays the ON button

if (output12State=="off") {

client.println("<p><a href=\"/12/on\"><button

class=\"button\">ON</button></a></p>");

} else {

client.println("<p><a href=\"/12/off\"><button class=\"button

button2\">OFF</button></a></p>");

}

// Display current state, and ON/OFF buttons for GPIO 14

client.println("<p>GPIO 14 - State " + output14State + "</p>");

// If the output14State is off, it displays the ON button

if (output14State=="off") {

client.println("<p><a href=\"/14/on\"><button

class=\"button\">ON</button></a></p>");

} else {

client.println("<p><a href=\"/14/off\"><button class=\"button

button2\">OFF</button></a></p>");

}

client.println("</body></html>");

// The HTTP response ends with another blank line

client.println();

// Break out of the while loop

break;

} else { // if you got a newline, then clear currentLine

currentLine = "";

}

} else if (c != '\r') { // if you got anything else but a carriage

return character,

currentLine += c; // add it to the end of the currentLine

}

}

}

// Clear the header variable

header = "";

// Close the connection

client.stop();

Serial.println("Client disconnected.");

Serial.println("");

}

}