Requirements:

Project: „Smart Manufacturing Monitoring and Control System“

Design a Smart Manufacturing Monitoring and Control System that uses an ESP microcontroller to ensure the environmental conditions are within specified ranges before allowing Bottled products to pass through different stages of the manufacturing process. This system incorporates two ultrasonic sensors to detect objects, a servo motor to control a gate, a buzzer for alerts, and a BME sensor to monitor temperature, humidity, altitude, and pressure. Sensor data is sent to a web server for real-time monitoring and analysis.

Components Needed
- ESP32/ESP8266 Microcontroller
- 2 x Ultrasonic Sensors (HC-SR04)
- Servo Motor (SG90)
- Buzzer
- BME280/BME680 Sensor
- Breadboard and Jumper Wires
- Power Supply
- IoT Wi-Fi Module (ESP32/ESP8266)Features

Steps:

1. Environmental Monitoring:
The BME sensor measures temperature, humidity, altitude, and pressure.
Data is continuously monitored and sent to a web server for logging and analysis.

2. Object Detection:
The two ultrasonic sensors detect the presence and position of objects on the production line.
Measure the distance between the object and sensors to ensure accurate detection.

3. Gate Control:
The servo motor acts as a gate, controlling the flow of products based on environmental conditions.
The gate remains closed if conditions are not within specified ranges and opens when conditions are suitable.

4. Alert System:
A buzzer provides audio alerts when environmental conditions are out of range or an object is detected. Alerts can be customized for different events (e.g., unsuitable conditions, object detection).

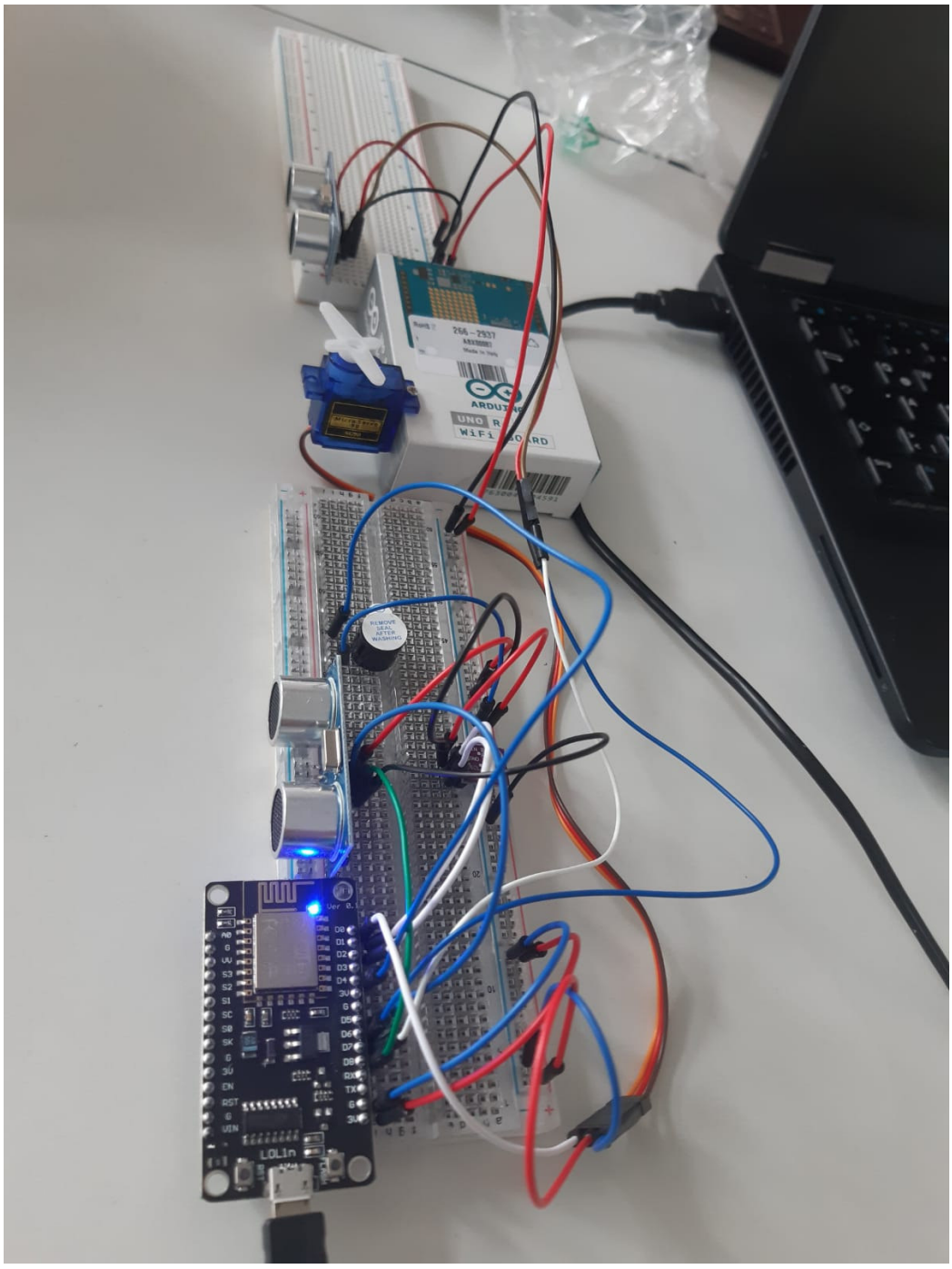
5. Web Server Integration:
Sensor data is sent to a web server for real-time monitoring and analysis.
The web interface displays environmental conditions and system status.
Optionally, remote control and alerts can be implemented via the web server.

How was it? What were the biggest problems in the implementation?

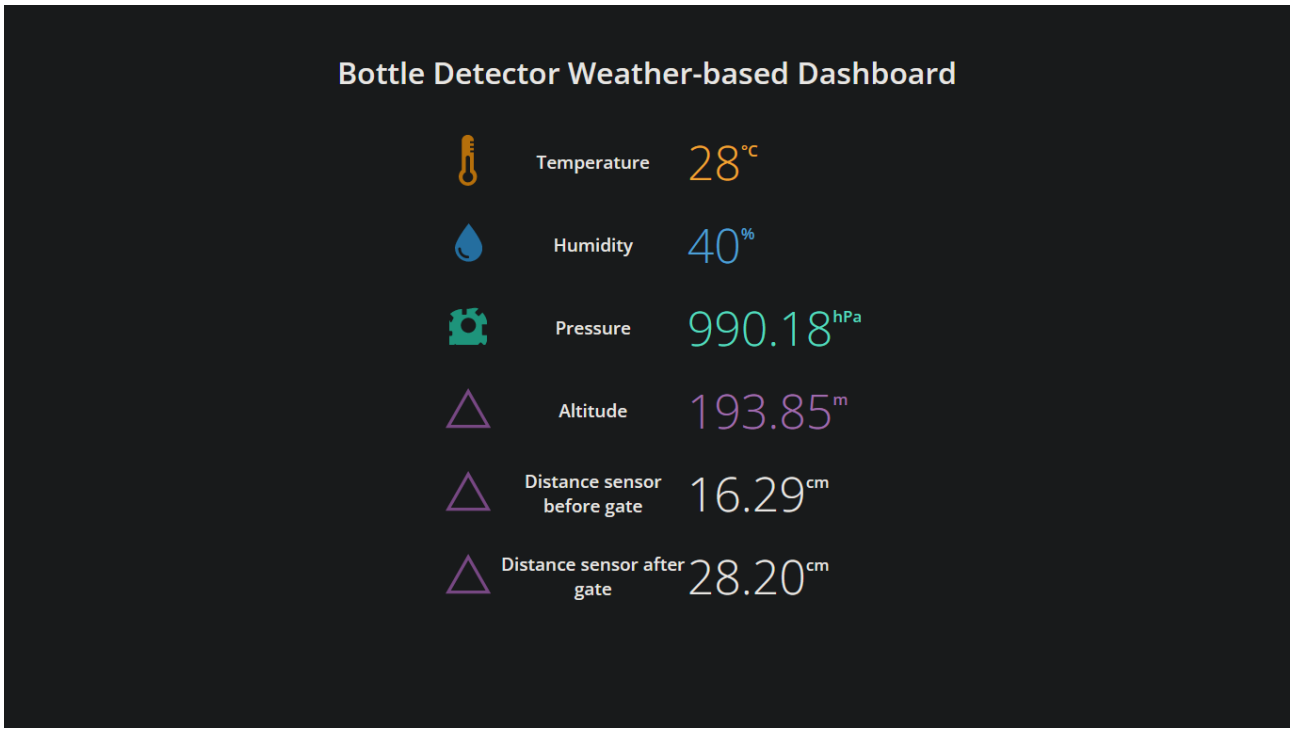
- I2C communication between R3 and R4
- I2C communication between R3 and ESP
- R4 cloud interaction was not possible
- R3 - ESP hirachy slave/master confusion
- CAN bus module was not working alone - a CAN controller is needed

Results:

Construction:



Webserver Dashbord acessable via IP:



C – Code:

```
#include <Arduino.h>

#include <ESP8266WebServer.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <Servo.h>

#define SEALEVELPRESSURE_HPA (1013.25)
Adafruit_BME280 bme;
float temperature, humidity, pressure, altitude, distance;

// Reassign HC-SR04 pins to free up I2C pins
#define TRIGGER_PIN1 12 // D6 (GPIO12)
#define ECHO_PIN1 13    // D7 (GPIO13)

#define TRIGGER_PIN2 14 // D5 (GPIO14)
#define ECHO_PIN2 15    // D8 (GPIO15)

// Maximum distance we want to ping for (in centimeters).
#define MAX_DISTANCE 400
#define MIN_DISTANCE1 10
#define MIN_DISTANCE2 10

// Temperatures set for the gate pass
#define MIN_TEMP 26
#define MAX_TEMP 30
```

```
#define BUZZER_PIN 2 // D4 (GPIO02)
#define SERVO_PIN 16 // D0 (GPIO16)

/*Put your SSID & Password*/
const char *ssid = "Yakha2024";     // Enter SSID here
const char *password = "Yakha2024"; // Enter Password here

float distanceBeforeGate = 0;
float distanceAfterGate = 0;

bool gateOpen = false;

void handle_OnConnect();
void handle_NotFound();
String SendHTML(float temperature, float humidity, float pressure, float altitude, float
distanceBeforeGate, float distanceAfterGate);

ESP8266WebServer server(80);
Servo gateServo;

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

   pinMode(TRIGGER_PIN1, OUTPUT);
   pinMode(ECHO_PIN1, INPUT);

   pinMode(TRIGGER_PIN2, OUTPUT);
   pinMode(ECHO_PIN2, INPUT);

   pinMode(BUZZER_PIN, OUTPUT);
   gateServo.attach(SERVO_PIN);
   gateServo.write(0); // Initialize servo to 0 degrees
   delay(100);
   bme.begin(0x76);
   Serial.println("Connecting to ");
   Serial.println(ssid);
   // connect to your local wi-fi network
   WiFi.begin(ssid, password);
   // check wi-fi is connected to wi-fi network
   while (WiFi.status() != WL_CONNECTED)
   {
      delay(1000);
      Serial.print(".");
   }
   Serial.println("");
   Serial.println("WiFi connected..!");
   Serial.print("Got IP: ");
   Serial.println(WiFi.localIP());
   server.on("/", handle_OnConnect);
   server.onNotFound(handle_NotFound);
```

```
    server.begin();
    Serial.println("HTTP server started");
}
void loop()
{
    temperature = bme.readTemperature();
    humidity = bme.readHumidity();
    pressure = bme.readPressure() / 100.0F;
    altitude = bme.readAltitude(SEALEVELPRESSURE_HPA);

    // Sensor1
    digitalWrite(TRIGGER_PIN1, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIGGER_PIN1, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIGGER_PIN1, LOW);
    float duration1 = pulseIn(ECHO_PIN1, HIGH);
    distanceBeforeGate = duration1 * 0.034 / 2;

    // Sensor2
    digitalWrite(TRIGGER_PIN2, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIGGER_PIN2, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIGGER_PIN2, LOW);
    float duration2 = pulseIn(ECHO_PIN2, HIGH);
    distanceAfterGate = duration2 * 0.034 / 2;

    Serial.print("Temperature = ");
    Serial.print(temperature);
    Serial.println(" *C");

    Serial.print("Distance Sensor 1 = ");
    Serial.print(distanceBeforeGate);
    Serial.println(" cm");

    Serial.print("Distance Sensor 2 = ");
    Serial.print(distanceAfterGate);
    Serial.println(" cm");
    Serial.println();

    // open gate, when something before gate and when temp is above MIN_TEMP
    if (distanceBeforeGate < MIN_DISTANCE1 && temperature >= MIN_TEMP && !gateOpen)
    {
        gateOpen = true;
        tone(BUZZER_PIN, 1000, 400);
        gateServo.write(90); // Turn the servo to 90 degrees
    }
    // keep gate open while passing second sensor and above MIN_TEMP
    else if (distanceAfterGate < MIN_DISTANCE2 && temperature >= MIN_TEMP && gateOpen)
    {
        tone(BUZZER_PIN, 1000, 400);
```

```
      gateServo.write(90); // Turn the servo to 90 degrees
    }
    // keep gate open while passing both sensors and temp above MIN_TEMP
    else if (distanceBeforeGate < MIN_DISTANCE1 && distanceAfterGate < MIN_DISTANCE2
&& temperature >= MIN_TEMP && gateOpen)
    {
      tone(BUZZER_PIN, 1000, 400);
      gateServo.write(90); // Turn the servo to 90 degrees
    }
    // buzz when something before gate, but MIN_TEMP was not reached
    else if (distanceBeforeGate < MIN_DISTANCE1)
    {
      tone(BUZZER_PIN, 1000, 200);
    }
    // close gate and stop buzzing
    else
    {
      gateOpen = false;
      noTone(BUZZER_PIN);
      gateServo.write(0); // Reset the servo to 0 degrees
    }

    server.handleClient();

    delay(500);
}
void handle_OnConnect()
{
    server.send(200, "text/html", SendHTML(temperature, humidity, pressure, altitude,
distanceBeforeGate, distanceAfterGate));
}
void handle_NotFound()
{
    server.send(404, "text/plain", "Not found");
}
String SendHTML(float temperature, float humidity, float pressure, float altitude, float
distanceBeforeGate, float distanceAfterGate)
{
    String ptr = "<!DOCTYPE html>";
    ptr += "<html>";
    ptr += "<head>";
    ptr += "<title>ESP8266 Weather Station</title>";
    ptr += "<meta name='viewport' content='width=device-width, initial-scale=1.0'>";
    ptr += "<link href='https://fonts.googleapis.com/css?family=Open+Sans:300,400,600'
rel='stylesheet'>";
    ptr += "<style>";
    ptr += "html { font-family: 'Open Sans', sans-serif; display: block; margin: 0px auto; text-align:
center; color: #444444;}";
    ptr += "body { margin: 0px; }";
    ptr += "h1 { margin: 50px auto 30px; }";
    ptr += ".side-by-side { display: table-cell; vertical-align: middle; position: relative; }";
    ptr += ".text { font-weight: 600; font-size: 19px; width: 200px; }";
```

```
    ptr += ".reading { font-weight: 300; font-size: 50px; padding-right: 25px; }";
    ptr += ".temperature .reading { color: #F29C1F; }";
    ptr += ".humidity .reading { color: #3B97D3; }";
    ptr += ".pressure .reading { color: #26B99A; }";
    ptr += ".altitude .reading { color: #955BA5; }";
    ptr += ".superscript { font-size: 17px; font-weight: 600; position: absolute; top: 10px; }";
    ptr += ".data { padding: 10px; }";
    ptr += ".container { display: table; margin: 0 auto; }";
    ptr += ".icon { width: 65px }";
    ptr += "</style>";
    ptr += "</head>";
    ptr += "<body>";
    ptr += "<h1>Bottle Detector Weather-based Dashboard</h1>";
    ptr += "<div class='container'>";
    ptr += "<div class='data temperature'>";
    ptr += "<div class='side-by-side icon'>";
    ptr += "<svg enable-background='new 0 0 19.438 54.003' height='54.003px' id='Layer_1'
version='1.1' viewBox='0 0 19.438 54.003' width='19.438px' x='0px' xml:space='preserve'
xmlns='http://www.w3.org/2000/svg' xmlns:xlink='http://www.w3.org/1999/xlink'
y='0px'><g><path d='M11.976,8.82v-
2h4.084V6.063C16.06,2.715,13.345,0,9.996,0H9.313C5.965,0,3.252,2.715,3.252,6.063v30.982
C1.261,38.825,0,41.403,0,44.286c0,5.367,4.351,9.718,9.719,9.718c5.368,0,9.719-4.351,9.719-
9.718 c0-2.943-1.312-5.574-3.378-7.355V18.436h-3.914v-2h3.914v-2.808h-4.084v-
2h4.084V8.82H11.976z M15.302,44.833 c0,3.083-2.5,5.583-5.583,5.583s-5.583-2.5-5.583-
5.583c0-2.279,1.368-4.236,3.326-5.104V24.257C7.462,23.01,8.472,22,9.719,22
s2.257,1.01,2.257,2.257V39.73C13.934,40.597,15.302,42.554,15.302,44.833z'
fill='#F29C21'/></g></svg>";
    ptr += "</div>";
    ptr += "<div class='side-by-side text'>Temperature</div>";
    ptr += "<div class='side-by-side reading'>" + String((int)temperature) + "<span
class='superscript'>&deg;C</span></div>";
    ptr += "</div>";
    ptr += "<div class='data humidity'>";
    ptr += "<div class='side-by-side icon'>";
    ptr += "<svg enable-background='new 0 0 29.235 40.64' height='40.64px' id='Layer_1'
version='1.1' viewBox='0 0 29.235 40.64' width='29.235px' x='0px' xml:space='preserve'
xmlns='http://www.w3.org/2000/svg' xmlns:xlink='http://www.w3.org/1999/xlink' y='0px'><path
d='M14.618,0C14.618,0,0,17.95,0,26.022C0,34.096,6.544,40.64,14.618,40.64s14.617-
6.544,14.617-14.617 C29.235,17.95,14.618,0,14.618,0z M13.667,37.135c-5.604,0-10.162-4.56-
10.162-10.162c0-0.787,0.638-1.426,1.426-1.426
c0.787,0,1.425,0.639,1.425,1.426c0,4.031,3.28,7.312,7.311,7.312c0.787,0,1.425,0.638,1.425,1.425
C15.093,36.497,14.455,37.135,13.667,37.135z' fill='#3C97D3'/></svg>";
    ptr += "</div>";
    ptr += "<div class='side-by-side text'>Humidity</div>";
    ptr += "<div class='side-by-side reading'>" + String((int)humidity) + "<span class='superscript'>
%</span></div>";
    ptr += "</div>";
    ptr += "<div class='data pressure'>";
    ptr += "<div class='side-by-side icon'>";
    ptr += "<svg enable-background='new 0 0 40.542 40.541' height='40.541px' id='Layer_1'
version='1.1' viewBox='0 0 40.542 40.541' width='40.542px' x='0px' xml:space='preserve'
xmlns='http://www.w3.org/2000/svg' xmlns:xlink='http://www.w3.org/1999/xlink'
```

```
y='0px'><g><path d='M34.313,20.271c0-0.552,0.447-1,1-1h5.178c-0.236-4.841-2.163-9.228-
5.214-12.593l-3.425,3.426 c-0.391,0.391-1.023,0.391-1.414,0L27.69,7.455c-0.391-0.391-0.391-
1.023,0-1.414l3.424-3.425C27.749,0.566,23.362,2.639,18.521,2.874 v5.178c0,0.553-0.447,1-1,1h-
3.125c-0.552,0-1-0.447-1-1V2.874c-4.841,0.235-9.228,2.163-12.593,5.214l3.426,3.425
c0.391,0.391,0.391,1.023,0,1.414L2.463,16.62c-0.391,0.391-1.023,0.391-1.414,0L0.317,14.888c-
2.262,3.37-3.348,7.367-3.336,11.383 l5.178,0c0.553,0,1,0.447,1,1v3.125c0,0.553-0.447,1-1,1l-
5.178,0.001c0.236,4.841,2.163,9.228,5.214,12.592l3.425-3.424 c0.391-0.391,1.023-
0.391,1.414,0l1.731,1.731c0.391,0.391,0.391,1.023,0,1.414l-
3.426,3.425c3.37,2.262,7.367,3.348,11.383,3.336 v-5.178c0-0.553,0.448-1,1-
1h3.125c0.553,0,1,0.447,1,1v5.178c4.841-0.236,9.228-2.163,12.593-5.214l-3.426-3.425 c-0.391-
0.391-0.391-1.023,0-1.414l1.731-1.731c0.391-0.391,1.023-0.391,1.414,0l3.425,3.426c2.262-
3.37,3.348-7.367,3.336-11.383 h-5.178c-0.553,0-1-0.447-1-1V20.271z M20.271,28.271c-4.418,0-
8-3.582-8-8c0-4.418,3.582-8,8-8s8,3.582,8,8 C28.271,24.689,24.689,28.271,20.271,28.271z'
fill='#26B99A'/></g></svg>";
    ptr += "</div>";
    ptr += "<div class='side-by-side text'>Pressure</div>";
    ptr += "<div class='side-by-side reading'>" + String(pressure) + "<span
class='superscript'>hPa</span></div>";
    ptr += "</div>";
    ptr += "<div class='data altitude'>";
    ptr += "<div class='side-by-side icon'>";
    ptr += "<svg enable-background='new 0 0 48.226 48.226' height='48.226px' id='Layer_1'
version='1.1' viewBox='0 0 48.226 48.226' width='48.226px' x='0px' xml:space='preserve'
xmlns='http://www.w3.org/2000/svg' xmlns:xlink='http://www.w3.org/1999/xlink'
y='0px'><g><path d='M48.226,45.138H0l24.113-42.05L48.226,45.138z
M42.548,42.05L24.113,9.776L5.679,42.05H42.548z' fill='#955BA5'/></g></svg>";
    ptr += "</div>";
    ptr += "<div class='side-by-side text'>Altitude</div>";
    ptr += "<div class='side-by-side reading'>" + String(altitude) + "<span
class='superscript'>m</span></div>";
    ptr += "</div>";
    ptr += "<div class='data distanceBeforeGate'>";
    ptr += "<div class='side-by-side icon'>";
    ptr += "<svg enable-background='new 0 0 48.226 48.226' height='48.226px' id='Layer_1'
version='1.1' viewBox='0 0 48.226 48.226' width='48.226px' x='0px' xml:space='preserve'
xmlns='http://www.w3.org/2000/svg' xmlns:xlink='http://www.w3.org/1999/xlink'
y='0px'><g><path d='M48.226,45.138H0l24.113-42.05L48.226,45.138z
M42.548,42.05L24.113,9.776L5.679,42.05H42.548z' fill='#955BA5'/></g></svg>";
    ptr += "</div>";
    ptr += "<div class='side-by-side text'>Distance sensor before gate</div>";
    ptr += "<div class='side-by-side reading'>" + String(distanceBeforeGate) + "<span
class='superscript'>cm</span></div>";
    ptr += "</div>";
    ptr += "<div class='data distanceAfterGate'>";
    ptr += "<div class='side-by-side icon'>";
    ptr += "<svg enable-background='new 0 0 48.226 48.226' height='48.226px' id='Layer_1'
version='1.1' viewBox='0 0 48.226 48.226' width='48.226px' x='0px' xml:space='preserve'
xmlns='http://www.w3.org/2000/svg' xmlns:xlink='http://www.w3.org/1999/xlink'
y='0px'><g><path d='M48.226,45.138H0l24.113-42.05L48.226,45.138z
M42.548,42.05L24.113,9.776L5.679,42.05H42.548z' fill='#955BA5'/></g></svg>";
    ptr += "</div>";
    ptr += "<div class='side-by-side text'>Distance sensor after gate</div>";
```

```
    ptr += "<div class='side-by-side reading'>" + String(distanceAfterGate) + "<span
class='superscript'>cm</span></div>";
    ptr += "</div>";
    ptr += "</div>";
    ptr += "<script>";
    ptr += "setInterval(function() {";
    ptr += "var xhttp = new XMLHttpRequest();";
    ptr += "xhttp.onreadystatechange = function() {";
    ptr += "if (this.readyState == 4 && this.status == 200) {";
    ptr += "document.body.innerHTML = this.responseText;";
    ptr += "}";
    ptr += "};";
    ptr += "xhttp.open('GET', '/', true);";
    ptr += "xhttp.send();";
    ptr += "}, 5000);"; // Update the data every 5 seconds
    ptr += "</script>";
    ptr += "</body>";
    ptr += "</html>";
    return ptr;
}
```