

















files

master

o to file

.idea

.gitignore

appInsightsSettings.xml

compiler.xml

deploymentTargetSelector.xml

gradle.xml

material\_theme\_project\_new.xml

migrations.xml

misc.xml

other.xml

app

gradle

.gitignore

ESP8266 Code

Code Blame 26 lines (26 loc) · 938 Bytes



Raw



```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project version="4">
3    <component name="AppInsightsSettings">
4      <option name="tabSettings">
5        <map>
6          <entry key="Firebase Crashlytics">
7            <value>
8              <InsightsFilterSettings>
9                <option name="connection">
10                  <ConnectionSetting>
11                    <option name="appId" value="PLACEHOLDER" />
12                    <option name="mobileSdkAppId" value="" />
13                    <option name="projectId" value="" />
14                    <option name="projectNumber" value="" />
15                  </ConnectionSetting>
16                </option>
17                <option name="signal" value="SIGNAL_UNSPECIFIED" />
18                <option name="timeIntervalDays" value="THIRTY_DAYS" />
19                <option name="visibilityType" value="ALL" />
20              </InsightsFilterSettings>
21            </value>
22          </entry>
23        </map>
24      </option>
25    </component>
26  </project>
```



Developer: Initial commit

t

Code

Blame

6 lines (6 loc) · 169 Bytes

```
1    <?xml version="1.0" encoding="UTF-8"?>
2    <project version="4">
3        <component name="CompilerConfiguration">
4            <bytecodeTargetLevel target="17" />
5        </component>
6    </project>
```



```
1  package com.keshav.controller;
2
3  import android.content.Context;
4
5  import androidx.test.platform.app.InstrumentationRegistry;
6  import androidx.test.ext.junit.runners.AndroidJUnit4;
7
8  import org.junit.Test;
9  import org.junit.runner.RunWith;
10
11  import static org.junit.Assert.*;
12
13  /**
14   * Instrumented test, which will execute on an Android device.
15   *
16   * @see <a href="http://d.android.com/tools/testing">Testing documentation</a>
17   */
18  @RunWith(AndroidJUnit4.class)
19  public class ExampleInstrumentedTest {
20      @Test
21      public void useAppContext() {
22          // Context of the app under test.
23          Context appContext = InstrumentationRegistry.getInstrumentation().getTargetContext();
24          assertEquals("com.keshav.controller", appContext.getPackageName());
25      }
26  }
```



Code Blame 26 lines (21 loc) · 756 Bytes

Raw

```
1  package com.keshav.controller;
2
3  import android.content.Context;
4
5  import androidx.test.platform.app.InstrumentationRegistry;
6  import androidx.test.ext.junit.runners.AndroidJUnit4;
7
8  import org.junit.Test;
9  import org.junit.runner.RunWith;
10
11  import static org.junit.Assert.*;
12
13  /**
14   * Instrumented test, which will execute on an Android device.
15   *
16   * @see <a href="http://d.android.com/tools/testing">Testing documentation</a>
17   */
18  @RunWith(AndroidJUnit4.class)
19  public class ExampleInstrumentedTest {
20      @Test
21      public void useAppContext() {
22          // Context of the app under test.
23          Context appContext = InstrumentationRegistry.getInstrumentation().getTargetContext();
24          assertEquals("com.keshav.controller", appContext.getPackageName());
25      }
26  }
```



Code

Blame

222 lines (187 loc) - 9.06 KB



```
1 package com.keshav.controller;
2
3 import android.content.pm.ActivityInfo;
4 import android.os.Bundle;
5 import android.util.Log;
6 import android.view.MotionEvent;
7 import android.view.View;
8 import android.widget.Toast;
9
10 import androidx.activity.EdgeToEdge;
11 import androidx.appcompat.app.AppCompatActivity;
12
13 import soup.neumorphism.NeumorphButton;
14 import soup.neumorphism.NeumorphImageButton;
15 import soup.neumorphism.NeumorphTextView;
16
17 import java.io.BufferedReader;
18 import java.io.InputStreamReader;
19 import java.net.HttpURLConnection;
20 import java.net.URL;
21
22 public class MainActivity extends AppCompatActivity {
```

Code Blame 167 lines (145 loc) - 4.42 KB

Raw Copy Download Edit 49

```
1  #include <ESP8266WiFi.h>
2  #include <ESP8266WebServer.h>
3
4  // Wi-Fi credentials
5  const char *ssid = "your Wi-Fi SSID"; // Replace with your Wi-Fi SSID
6  const char *password = "your Wi-Fi password"; // Replace with your Wi-Fi password
7
8  // Motor Pins
9  int motor1Pin1 = 12; // D6
10 int motor1Pin2 = 14; // D5
11 int enable1Pin = 13; // D7
12
13 int motor2Pin1 = 5; // D1
14 int motor2Pin2 = 4; // D2
15 int enable2Pin = 15; // D8
16
17 // Ultrasonic sensor pins
18 int trigPin = 2; // D4
19 int echoPin = 0; // D3
20
21 // Default speed value
22 int speed = 255;
23 long distance = 0;
24
25 // ESP8266 Web Server
26 ESP8266WebServer server(80);
27
28 // Function to measure distance
29 float getDistance() {
```



```
18 int trigPin = 2; // D4
19 int echoPin = 0; // D3
20
21 // Default speed value
22 int speed = 255;
23 long distance = 0;
24
25 // ESP8266 Web Server
26 ESP8266WebServer server(80);
27
28 // Function to measure distance
29 float getDistance() {
30     digitalWrite(trigPin, LOW);
31     delayMicroseconds(2);
32     digitalWrite(trigPin, HIGH);
33     delayMicroseconds(10);
34     digitalWrite(trigPin, LOW);
35
36     long duration = pulseIn(echoPin, HIGH, 30000); // 30ms timeout
37     if (duration == 0) {
38         return -1; // Indicate out of range
39     }
40     return (duration * 0.034) / 2; // Calculate distance in cm
41 }
```



```
44 void handleDistance() {
45     float distance = getDistance();
46     Serial.print(distance);
47     if (distance != -1) {
48         server.send(200, "text/plain", String(distance) + " cm");
49     } else {
50         server.send(200, "text/plain", "Out of range");
51     }
52 }
53
54 void EmergencyStop(){
55     digitalWrite(motor1Pin1, LOW);
56     digitalWrite(motor1Pin2, LOW);
57     digitalWrite(motor2Pin1, LOW);
58     digitalWrite(motor2Pin2, LOW);
59     analogWrite(enable1Pin, 0);
60     analogWrite(enable2Pin, 0);
61     server.send(200,"text/plain","Brakes Applied!");
62 }
63
64 // Function to handle speed control
65 void handleSetSpeed() {
66     if (server.hasArg("level")) {
67         int level = server.arg("level").toInt();
68         switch (level) {
69             case 1: speed = 550; break;
```



```
64 // Function to handle speed control
65 void handleSetSpeed() {
66     if (server.hasArg("level")) {
67         int level = server.arg("level").toInt();
68         switch (level) {
69             case 1: speed = 550; break;
70             case 2: speed = 650; break;
71             case 3: speed = 800; break;
72             case 4: speed = 900; break;
73             default: speed = 550; // Default to level 1
74         }
75         analogWrite(enable1Pin, speed);
76         analogWrite(enable2Pin, speed);
77         server.send(200, "text/plain", "Speed level set to: " + String(level));
78     } else {
79         server.send(400, "text/plain", "Speed level missing");
80     }
81 }
82
83 // Function to handle motor commands
84 void handleCommand() {
85     String command = server.arg("cmd");
86     Serial.print(command);
87     if (command == "/forward") {
88         digitalWrite(motor1Pin1, LOW);
89         digitalWrite(motor1Pin2, HIGH);
90         digitalWrite(motor2Pin1, HIGH);
```



```
74     }
75     analogWrite(enable1Pin, speed);
76     analogWrite(enable2Pin, speed);
77     server.send(200, "text/plain", "Speed level set to: " + String(level));
78 } else {
79     server.send(400, "text/plain", "Speed level missing");
80 }
81 }
82
83 // Function to handle motor commands
84 void handleCommand() {
85     String command = server.arg("cmd");
86     Serial.print(command);
87     if (command == "/forward") {
88         digitalWrite(motor1Pin1, LOW);
89         digitalWrite(motor1Pin2, HIGH);
90         digitalWrite(motor2Pin1, HIGH);
91         digitalWrite(motor2Pin2, LOW);
92     } else if (command == "/backward") {
93         digitalWrite(motor1Pin1, HIGH);
94         digitalWrite(motor1Pin2, LOW);
95         digitalWrite(motor2Pin1, LOW);
96         digitalWrite(motor2Pin2, HIGH);
97     } else if (command == "/left") {
98         digitalWrite(motor1Pin1, LOW);
99         digitalWrite(motor1Pin2, HIGH);
100        digitalWrite(motor2Pin1, LOW);
```



```
83 // Function to handle motor commands
84 void handleCommand() {
85     String command = server.arg("cmd");
86     Serial.print(command);
87     if (command == "/forward") {
88         digitalWrite(motor1Pin1, LOW);
89         digitalWrite(motor1Pin2, HIGH);
90         digitalWrite(motor2Pin1, HIGH);
91         digitalWrite(motor2Pin2, LOW);
92     } else if (command == "/backward") {
93         digitalWrite(motor1Pin1, HIGH);
94         digitalWrite(motor1Pin2, LOW);
95         digitalWrite(motor2Pin1, LOW);
96         digitalWrite(motor2Pin2, HIGH);
97     } else if (command == "/left") {
98         digitalWrite(motor1Pin1, LOW);
99         digitalWrite(motor1Pin2, HIGH);
100        digitalWrite(motor2Pin1, LOW);
101        digitalWrite(motor2Pin2, HIGH);
102    } else if (command == "/right") {
103        digitalWrite(motor1Pin1, HIGH);
104        digitalWrite(motor1Pin2, LOW);
105        digitalWrite(motor2Pin1, HIGH);
106        digitalWrite(motor2Pin2, LOW);
107    } else if (command == "/stop") {
108        digitalWrite(motor1Pin1, LOW);
109        digitalWrite(motor1Pin2, LOW);
110        digitalWrite(motor2Pin1, LOW);
```



```
109     digitalWrite(motor1Pin2, LOW);
110     digitalWrite(motor2Pin1, LOW);
111     digitalWrite(motor2Pin2, LOW);
112 }
113 server.send(200, "text/plain", "Command executed: " + command);
114 }
115
116 void setup() {
117     // Initialize motor and ultrasonic sensor pins
118     pinMode(motor1Pin1, OUTPUT);
119     pinMode(motor1Pin2, OUTPUT);
120     pinMode(enable1Pin, OUTPUT);
121     pinMode(motor2Pin1, OUTPUT);
122     pinMode(motor2Pin2, OUTPUT);
123     pinMode(enable2Pin, OUTPUT);
124
125     pinMode(trigPin, OUTPUT);
126     pinMode(echoPin, INPUT);
127
128     Serial.begin(115200);
129     WiFi.begin(ssid, password);
130     Serial.println("Connecting to Wi-Fi...");
131
132     while (WiFi.status() != WL_CONNECTED) {
133         delay(1000);
134         Serial.print(".");
135     }
```



```
135 }
136
137 Serial.println("Connected to Wi-Fi");
138 Serial.print("IP Address: ");
139 Serial.println(WiFi.localIP());
140
141 server.on("/status", []() {
142     Serial.print("added");
143     server.send(200, "text/plain", "ESP is online");
144 });
145
146 server.on("/setSpeed", HTTP_GET, handleSetSpeed);
147 server.on("/command", HTTP_GET, handleCommand);
148 server.on("/distance", HTTP_GET, handleDistance);
149 server.on("/EmergencyStop", HTTP_GET, EmergencyStop);
150
151 server.begin();
152 Serial.println("Server started");
153 }
154
155 void loop() {
156     server.handleClient();
157
158     float distance = getDistance();
159     Serial.print(distance);
160     if (distance <= 50 && distance >=0 ) {
161         digitalWrite(motor1Pin1, LOW);
162         digitalWrite(motor1Pin2, LOW);
```



```
144     });
145
146     server.on("/setSpeed", HTTP_GET, handleSetSpeed);
147     server.on("/command", HTTP_GET, handleCommand);
148     server.on("/distance", HTTP_GET, handleDistance);
149     server.on("/EmergencyStop", HTTP_GET, EmergencyStop);
150
151     server.begin();
152     Serial.println("Server started");
153 }
154
155 void loop() {
156     server.handleClient();
157
158     float distance = getDistance();
159     Serial.print(distance);
160     if (distance <= 50 && distance >= 0 ) {
161         digitalWrite(motor1Pin1, LOW);
162         digitalWrite(motor1Pin2, LOW);
163         digitalWrite(motor2Pin1, LOW);
164         digitalWrite(motor2Pin2, LOW);
165         server.send(200, "text/plain", "Stopped Due to Obstacle.");
166     }
167 }
```