ESP8266 WiFi Robot Car

13.02.2023 r.
Micro-controllers Programming
Repository: ESP8266 WiFi Robot Car
Created by Mateusz Suszczyk
Watch the video demo! -> link

Description

The main purpose of my robot is to drive accordingly depending on what button will be clicked on a website accessible from PC/smartphone. In addition, the robot collects sensor data such as temperature, humidity, motion detection, and noise value.

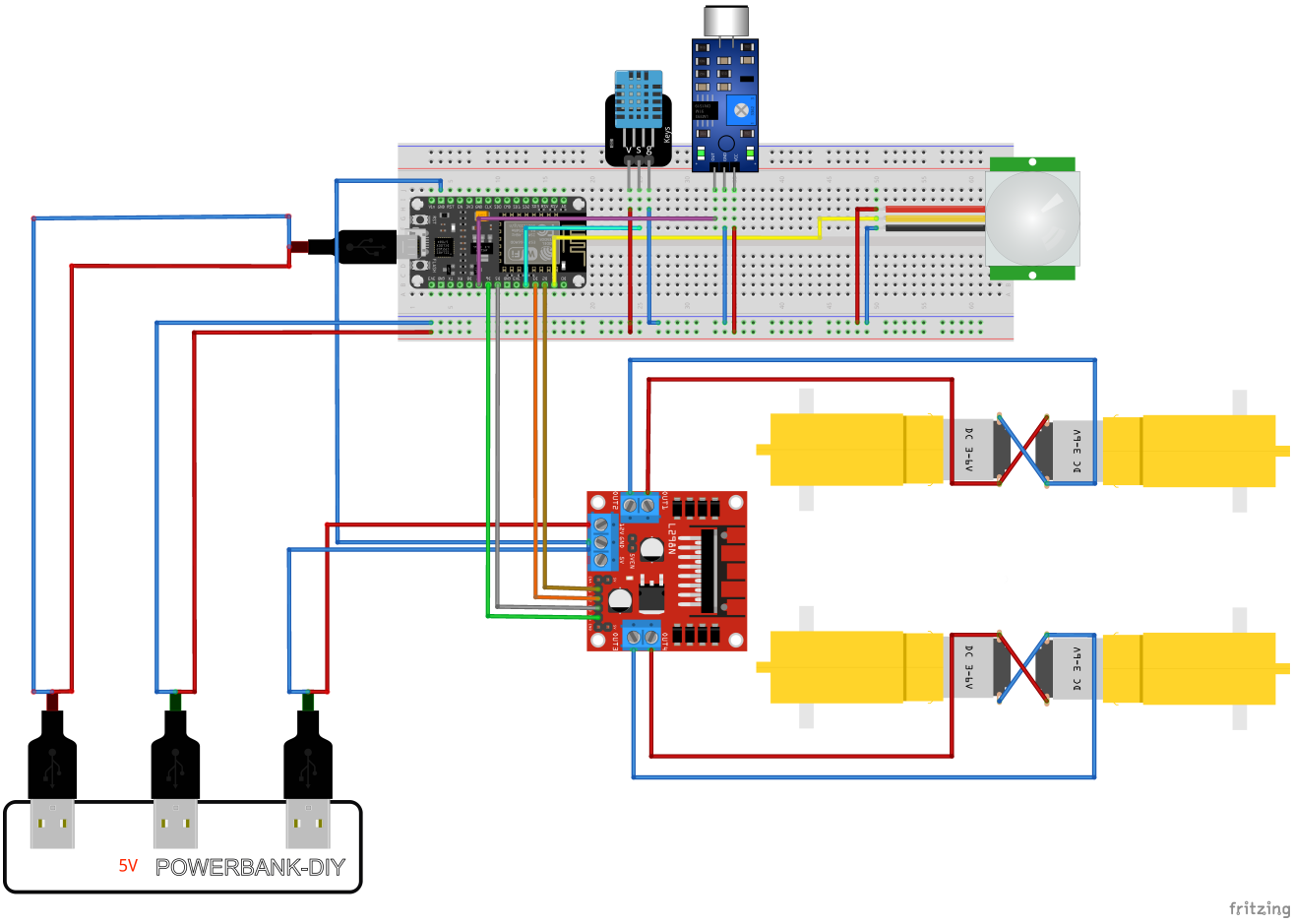
Components used

- NodeMCU v2 WiFi ESP8266
- Robot Chassis + tyres
- Motor Driver LN298N
- DHT 11 - temperature and humidity sensor
- Sound sensor
- HC-SR501 - PIR sensor
- 4 DC motors
- Breadboard
- Wires
- Powerbank 5V
- USB Hub

Encountered problems

- Using a power bank as a source of power for motors is sufficient, but not optimal. To get better performance of motors, I highly encourage to use better source of power (e.g. 2x 18650 batteries).
- DHT11 tends to hang when reading values. To solve this problem try to reset the sensor by removing the cables and then putting it back.
- I had problems with .js file used to properly visualize knobs on site. Automatic refreshing values finally worked after one of those methods:
  - Putting .js on my GitHub and then reach file using cdn.jsdelivr.net -> pureknob.js.
  - Uploading file on internal flash memory using `server.streamFile(file, contentType);`.

Connections



Code

car.ino

```
/*
Created by: Mateusz Suszczyk
Contact: mateusz@suszczyk.pl
Date: 11.02.2023 r.
Language: c++/arduino
Developed for: ESP8266 WIFI development board (NodeMCU v2)
Intended for use with: L298N H-Bridge, 4 DC motors, Temperature sensor DHT 11,
PIR sensor HC-SR501, Sound sensor

// After uploading code to ESP8266, upload index.html to ESP using command
below:
// curl -F "file=@index.html" 192.168.90.200/upload
*/

#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <FS.h> //SPI flash file system library using to load website into flash
memory
#include <WebSocketsServer.h>
#include <ArduinoJson.h>

// Temeperature sensor
#include <DHT.h>
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

// Initialization variables
float humidity = 0.0;
float temperature = 0.0;
float noise = 0.0;

// Assign pin numbers for L298N enable and inputs
int in1 = 4;
int in2 = 0;
int in3 = 14;
int in4 = 12;

// Variables for updating sensors
long sensorUpdateFrequency = 50;
long timeNow = 0;
long timePrev = 0;

// Assign pin numbers for PIR and sound sensor
```

```
int pir_sensor = 5;
int noise_sensor = 13;
int sampleBufferValue = 0; // Variable to compute noise

ESP8266WebServer server;
WebSocketsServer webSocket = WebSocketsServer(81);

// Network credentials
char *ssid = "******";
char *password = "******";
// Hold uploaded file
File fsUploadFile;


void setup()
{
  SPIFFS.begin();
  WiFi.begin(ssid, password);
  Serial.begin(115200);
  dht.begin();

  pinMode(pir_sensor, INPUT);
  pinMode(noise_sensor, INPUT);

  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(500);
  }
  Serial.println("");
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());

  server.on("/", ControlDataFile);

  // List available files
  server.on("/list", HTTP_GET, FileList);

  // Handle file upload
  server.on(
      "/upload", HTTP_POST, []()
      { server.send(200, "text/plain", "{\"success\":1}"); },
      FileUpload);

  server.onNotFound([]()
                    {
    if(!FileRead(server.uri())) {
      server.send(404, "text/plain", "File Not Found!");
    } });
```

```
  server.begin();
  webSocket.begin();
  // Function to be called whenever there is a websocket event
  webSocket.onEvent(webSocketEvent);

  // Motor outputs
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
}

void loop()
{
  // Check the websocket status
  webSocket.loop();
  server.handleClient();
  int test_movement = digitalRead(pir_sensor); // Get PIR sensor value

  // See if it is time to update the sensor values on the website
  timeNow = millis();
  if (digitalRead(noise_sensor) == LOW)
  {
    sampleBufferValue++;
  }

  if (timeNow - timePrev >= sensorUpdateFrequency)
  {
    // Serial.println(sampleBufferValue);
    int noise = sampleBufferValue / 5;
    // Serial.print("Sensor value: "); Serial.println(test_movement);
    // Serial.print("Noise value: "); Serial.println(noise);
    sampleBufferValue = 0;
    timePrev = timeNow;
    // If it is time, call the updateSensors() function
    updateSensors(noise, test_movement);
  }
}

void webSocketEvent(uint8_t num, WStype_t type, uint8_t *payload, size_t length)
{
  if (type == WStype_TEXT)
  {
    // Handle the websocket messages with direction and speed
    // by parsing the parameters from a JSON string
    String payload_str = String((char *)payload);
    // Using the ArduinoJson library
    StaticJsonDocument<200> doc;
```

```
    // Deserialize the data
    DeserializationError error = deserializeJson(doc, payload_str);
    // Parse the parameters we expect to receive (TO-DO: error handling)
    String dir = doc["direction"];
    Serial.println(dir);
    if (dir == "STP")
    {
      // Motors
      digitalWrite(in1, LOW);
      digitalWrite(in2, LOW);
      digitalWrite(in3, LOW);
      digitalWrite(in4, LOW);
      Serial.println("STOP");
    }
    else
    {
      if (dir == "FWD")
      {
        Serial.println("FORWARD");
        digitalWrite(in1, HIGH);
        digitalWrite(in2, LOW);
        digitalWrite(in3, HIGH);
        digitalWrite(in4, LOW);
      }
      else if (dir == "BWD")
      {
        Serial.println("BACKWARD");
        digitalWrite(in1, LOW);
        digitalWrite(in2, HIGH);
        digitalWrite(in3, LOW);
        digitalWrite(in4, HIGH);
      }
      else if (dir == "RGT")
      {
        Serial.println("RIGHT");
        digitalWrite(in1, LOW);
        digitalWrite(in2, HIGH);
        digitalWrite(in3, HIGH);
        digitalWrite(in4, LOW);
      }
      else if (dir == "LFT")
      {
        Serial.println("LEFT");
        digitalWrite(in1, HIGH);
        digitalWrite(in2, LOW);
        digitalWrite(in3, LOW);
        digitalWrite(in4, HIGH);
      }
    }
```

```
  }
}

void FileUpload()
{
  // Deal with data file upload over httpupload
  HTTPUpload &upload = server.upload();
  if (upload.status == UPLOAD_FILE_START)
  {
    String filename = upload.filename;
    if (!filename.startsWith("/"))
    {
      filename = "/" + filename;
    }
    Serial.print("FileUpload Name: ");
    Serial.println(filename);
    fsUploadFile = SPIFFS.open(filename, "w");
  }
  else if (upload.status == UPLOAD_FILE_WRITE)
  {
    if (fsUploadFile)
    {
      fsUploadFile.write(upload.buf, upload.currentSize);
    }
  }
  else if (upload.status == UPLOAD_FILE_END)
  {
    if (fsUploadFile)
    {
      fsUploadFile.close();
    }
    Serial.print("FileUpload Size: ");
    Serial.println(upload.totalSize);
  }
}
void FileList()
{
  // Updates list of html pages for controling the car
  String path = "/";
  // Assuming there are no subdirectories
  Dir dir = SPIFFS.openDir(path);
  String output = "[";
  while (dir.next())
  {
    File entry = dir.openFile("r");
    // Separate by comma if there are multiple files
    if (output != "[")
      output += ",";
    output += String(entry.name()).substring(1);
```

```
    entry.close();
  }
  output += "]";
  server.send(200, "text/plain", output);
}

void ControlDataFile()
{
  // Initiates html page on server to control the car
  File file = SPIFFS.open("/index.html", "r");
  server.streamFile(file, "text/html");
  file.close();
}

String getContentType(String filename)
{
  if (filename.endsWith(".html"))
    return "text/html";
  else if (filename.endsWith(".js"))
    return "text/javascript";
  return "text/plain";
}

bool FileRead(String path)
{
  // Serve index file when top root path is accessed
  if (path.endsWith("/"))
    path += "index.html";
  // Different file types require different actions
  String contentType = getContentType(path);

  if (SPIFFS.exists(path))
  {
    fs::File file = SPIFFS.open(path, "r");
    if (contentType == "text/plain" || "text/javascript")
      server.streamFile(file, contentType);
    // Display the file on the client's browser

    file.close();
    return true;
  }
  return false; // If the file doesn't exist or can't be opened
}

void updateSensors(int noise, int movement)
{
  float humidity = dht.readHumidity();
  // Serial.println(humidity);
  float temperature = dht.readTemperature();
```

```
  // Serial.println(temperature);

  // If any value is isnan (not a number) then there is an error
  if (isnan(humidity) || isnan(temperature))
  {
    Serial.println("Error reading from the DHT11.");
  }
  else
  {
    String json = "{\"temperature\":";
    json += temperature;
    json += ",\"humidity\":";
    json += humidity;
    json += ",\"movement\":";
    json += movement;
    json += ",\"noise\":";
    json += noise;
    json += "}";

    Serial.println(json); // DEBUGGING
    webSocket.broadcastTXT(json.c_str(), json.length());
  }
}
```

index.html file

```
<html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <link rel="preconnect" href="https://fonts.googleapis.com" />
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
  <link href="https://fonts.googleapis.com/css2?
family=Hind+Guntur:wght@300&display=swap" rel="stylesheet" />
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
crossorigin="anonymous" />
  <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.2/css/all.css"
integrity="sha384-
fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr"
crossorigin="anonymous" />

    <!-- Option 1 - File uploaded using server.streamFile(file, contentType); --
>
  <script src="pureknob.js"></script>

  <!-- Option 2. - Host it on GitHub and use service like jsdelivr.net -->
  <!-- <script src="https://cdn.jsdelivr.net/gh/suszczyk/ESP8266-WiFi-
Robot@main/pureknob.js"></script> -->

  <script type="text/javascript">
    // Create meters, 165 x 165 px in size.
    const humidity_meter = pureknob.createKnob(165, 165);
    const temperature_meter = pureknob.createKnob(165, 165);
    const sensor_meter = pureknob.createKnob(165, 165);
    const noise_meter = pureknob.createKnob(165, 165);


    function temperature_knob() {
      // Set properties.
      temperature_meter.setProperty("angleStart", -0.75 * Math.PI);
      temperature_meter.setProperty("angleEnd", 0.75 * Math.PI);
      temperature_meter.setProperty("colorFG", "#ed9d00");
      temperature_meter.setProperty("trackWidth", 0.4);
      temperature_meter.setProperty("valMin", 0);
      temperature_meter.setProperty("valMax", 100);


temperature_meter.setValue(document.getElementById("temperature").innerHTML);
      const node = temperature_meter.node();
```

```
  const elem = document.getElementById("temp");
  elem.appendChild(node);
}

function humidity_knob() {
  // Set properties.
  humidity_meter.setProperty("angleStart", -0.75 * Math.PI);
  humidity_meter.setProperty("angleEnd", 0.75 * Math.PI);
  humidity_meter.setProperty("colorFG", "#04c0f8");
  humidity_meter.setProperty("trackWidth", 0.4);
  humidity_meter.setProperty("valMin", 0);
  humidity_meter.setProperty("valMax", 100);

  humidity_meter.setValue(document.getElementById("humidity").innerHTML);
  const node = humidity_meter.node();
  const elem = document.getElementById("humi");
  elem.appendChild(node);
}

function sensor_knob() {
  // Set properties.
  sensor_meter.setProperty("angleStart", -0.75 * Math.PI);
  sensor_meter.setProperty("angleEnd", 0.75 * Math.PI);
  sensor_meter.setProperty("colorFG", "#24c48e");
  sensor_meter.setProperty("trackWidth", 0.4);
  sensor_meter.setProperty("valMin", 0);
  sensor_meter.setProperty("valMax", 1);

  sensor_meter.setValue(document.getElementById("movement").innerHTML);
  const node = sensor_meter.node();
  const elem = document.getElementById("sens");
  elem.appendChild(node);
}


function noise_knob() {
  // Set properties.
  noise_meter.setProperty("angleStart", -0.75 * Math.PI);
  noise_meter.setProperty("angleEnd", 0.75 * Math.PI);
  noise_meter.setProperty("colorFG", "#d3435c");
  noise_meter.setProperty("trackWidth", 0.4);
  noise_meter.setProperty("valMin", 0);
  noise_meter.setProperty("valMax", 100);

  noise_meter.setValue(document.getElementById("noise").innerHTML);
  const node = noise_meter.node();
  const elem = document.getElementById("nois");
  elem.appendChild(node);
}
```

```
    function ready() {
      humidity_knob();
      temperature_knob();
      sensor_knob();
      noise_knob();
    }
    document.addEventListener("DOMContentLoaded", ready, false);
  </script>
  <style>
    body {
      margin-top: 5%;
      border: 3px;
      background-color: #212227;
      color: white;
      font-family: "Hind Guntur", sans-serif;
      height: 100vh;
    }
    .container {
      position: relative;
      text-align: center;
      font-size: 15px;
    }
    .myrow {
      height: 65px;

    }
    #buttons {
      height: auto;
      padding: 2%;

    }
    .button {
      position: absolute;
      top: 50%;
      left: 50%;
      transform: translate(-50%, -50%);
      background-color: #24c48e;
      border: none;
      color: black;
      padding: 15px 42px;
      text-align: center;
      text-decoration: none;
      display: inline-block;
      font-size: 16px;
      font-weight: bold;
      margin: 4px 2px;
      cursor: pointer;
      border-radius: 30px;
    }
```

```
    #gauges {
      margin-top: 5%;
      pointer-events: none;
    }
    .gauge {
      height: auto;
      width: 50%;

      float: left;

    }
    .gauge-row {
      height: auto;
      width: 100%;
      display: inline-block;

    }
  </style>
</head>

<body onload="javascript:mc_init()">
  <div class="container">
    <div id="buttons">
      <div class="row justify-content-center align-items-center myrow">
        <div class="col-md-4 col-6 mycol">
          <button class="button" id="FWD">FORWARD</button>
        </div>
      </div>

      <div class="row justify-content-around align-items-center myrow">
        <div class="col-md-4 col-5 mycol">
          <button class="button" id="LFT">LEFT</button>
        </div>
        <div class="col-md-4 col-5 mycol">
          <button class="button" id="RGT">RIGHT</button>
        </div>
      </div>

      <div class="row justify-content-center align-items-center myrow">
        <div class="col-md-4 col-6 mycol">
          <button class="button" id="BWD">BACKWARD</button>
        </div>
      </div>
    </div>

    <div id="gauges">
      <div class="gauge-row">
        <!-- Temperature -->
        <div class="gauge">
```

```
          <i class="fas fa-thermometer-half" style="color: #059e8a"></i>
          <span class="dht-labels">&nbspTemperature </span>
          <span id="temperature">0.0</span> 
          <span class="units">&deg;C</span>
          <div id="temp"></div>
        </div>

        <!-- Humidity -->
        <div class="gauge">
          <i class="fas fa-tint" style="color: #00add6"></i>
          <span class="dht-labels">&nbspHumidity </span>
          <span id="humidity">0</span> 
          <span class="units">%</span>
          <div id="humi"></div>
        </div>

      </div>

      <!-- Sensor -->
      <div class="gauge-row">
        <div class="gauge">
          <i class="fa fa-exclamation-triangle" aria-hidden="true" style="color:
#059e8a"></i>
          <span class="dht-labels">&nbspMotion Sensor </span>
          <span id="movement">0</span>
          <div id="sens"></div>
        </div>

        <!-- Noise -->
        <div class="gauge">
          <i class="fas fa-volume-up" style="color: #d3435c"></i>
          <span class="dht-labels">&nbspNoise level </span>
          <span id="noise">0</span> 
          <span class="units">%</span>
          <div id="nois"></div>
        </div>
      </div>
    </div>
  </div>

  <script>
    var socket;
    function mc_init() {
      // Initialiaze the websocket client
      socket = new WebSocket("ws://" + window.location.hostname + ":81/");
      var buttons = document.getElementsByTagName("button");
      for (i = 0; i < buttons.length; i++) {
        buttons[i].addEventListener("mousedown", move, true);
        buttons[i].addEventListener("mouseup", stop, true);
```

```
        buttons[i].addEventListener("touchstart", move, true);
        buttons[i].addEventListener("touchend", stop, true);
      }
      socket.onmessage = function (event) {
        // Receive the color data from the server
        var data = JSON.parse(event.data);
        console.log(data);

        // Get sensor values
        document.getElementById("humidity").innerHTML = data["humidity"];
        document.getElementById("temperature").innerHTML = data["temperature"];
        document.getElementById("movement").innerHTML = data["movement"];
        document.getElementById("noise").innerHTML = data["noise"];

        // Update knobs
        humidity_knob();
        temperature_knob();
        sensor_knob();
        noise_knob();
      };
    }
    function move(e) {
      e.preventDefault(); // Prevent hold to copy-paste menu pop-up on mobile!
      var data = { direction: e.srcElement.id };
      socket.send(JSON.stringify(data));
      return false;
    }
    function stop() {
      var data = { direction: "STP" };
      socket.send(JSON.stringify(data));
      return false;
    }
  </script>
</body>
</html>
```

HTML website

Setup

1. Clone repository.

2. After uploading code from `car.ino` to ESP8266 (remember to change the credentials of the wifi network), upload `index.html` to ESP using the command below:

```
curl -F "file=@index.html" x.x.x.x/upload
```

`x.x.x.x` is IP address of your ESP.

3. Make sure that javascript library is added. For more accurate sensor values I have changed `pureknob.js` line 635:

from

```
value = Math.round(value);
```

to

```
value = Math.round(value * 10) / 10
```

If the website does not work properly, add this line at the beginning of `index.html`: `<script src="https://cdn.jsdelivr.net/gh/suszczyk/ESP8266-WiFi-Robot-Car@main/pureknob.js"></script>` and rerun command from point 2.

4. Connect your phone to the same wifi network as your robot. After reaching `x.x.x.x` you should see the page above.

5. Happy driving :)

Libraries

- ESP8266WiFi
- ESP8266WebServer
- ESP8266WebServer
- FS.h
- WebSocketsServer.h
- ArduinoJson.h

Possible improvements

- Adding LCD screen
- Adding LM393 IR Speed sensor
- Connecting another ESP or Arduino to get more pins

Worth mentioning

datasith projects:

- Robot Car Controlled Using Websockets
- ESP8266 Display JPEGs On NeoPixel Matrix - Store images on ESP8266
- ESP8266 Analog Voltages for Controlling Webpage RGB Colors

Sensors:

- Sound sensor
- Sound sensor 2
- PIR sensor

Other:
ESP8266 and the Arduino IDE - IOT Website
Weather monitoring system using Blynk

UI:

- Dynamic bars: pure-knob
- Blynk colors palette: https://community.blynk.cc/uploads/default/original/2X/2/2d740375e05c5a79124e6d73c0da1c1b26802605.png