Build an ESP8266 Mobile Robot

Created by Marc-Olivier Schwartz

https://learn.adafruit.com/build-an-esp8266-mobile-robot

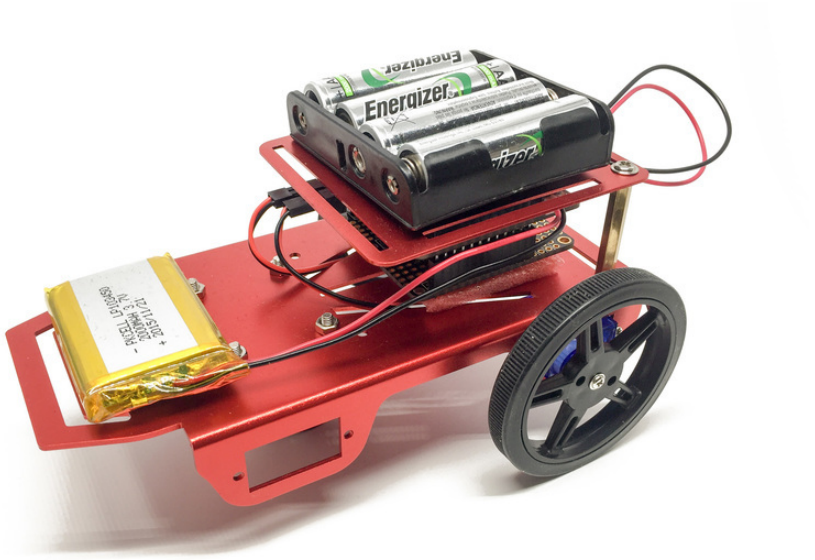
Last updated on 2024-06-03 01:58:04 PM EDT

Table of Contents

Introduction

Building your own mobile robot is becoming easier and easier, thanks to excellent ready-to-use robotic platforms. A good example of such platform is the Adafruit Mini Robot Chassis kit (http://adafru.it/2939), which comes with a nice robot chassis, two servo motors with wheels and a support-wheel. This makes it the perfect base for all your mobile robot projects.

On the other hand, you now can buy powerful & cheap microcontrollers like the ESP8266 WiFi chip, which is not only easy to use but also comes with onboard WiFi connectivity. This is just the perfect chip to control robots remotely from your computer or mobile device.



In this guide, you are going to learn how to build your own mobile robot based on the Adafruit Mini Robot Rover Chassis Kit, and on the ESP8266 WiFi chip. We are first going to see how to assemble the robot, and how to configure it so it can receive commands via WiFi. After that, we'll see how to control it via WiFi from a nice interface that runs in your web browser. Let's start!

Building the Robot

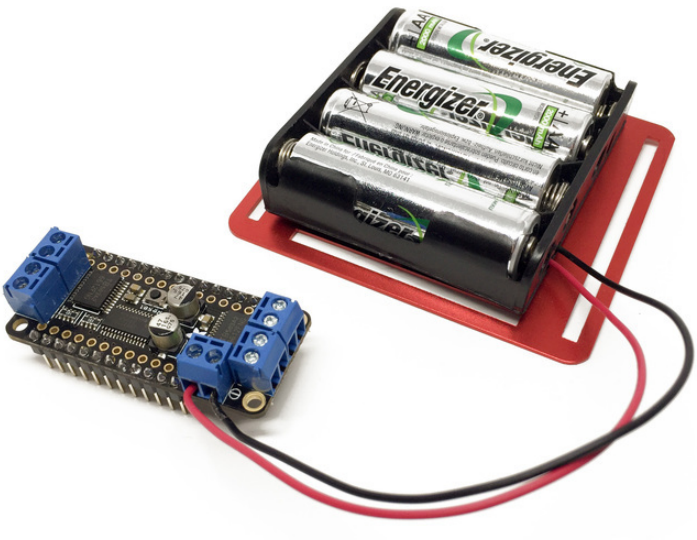
Let's first see how to assemble the robot. The Mini Robot Rover Chassis comes as a kit, so for the basic assembly I will refer you to this excellent guide to assemble the main parts of the robot:

https://learn.adafruit.com/bluefruit-feather-robot/wiring-and-assembly (https://adafru.it/kBl)

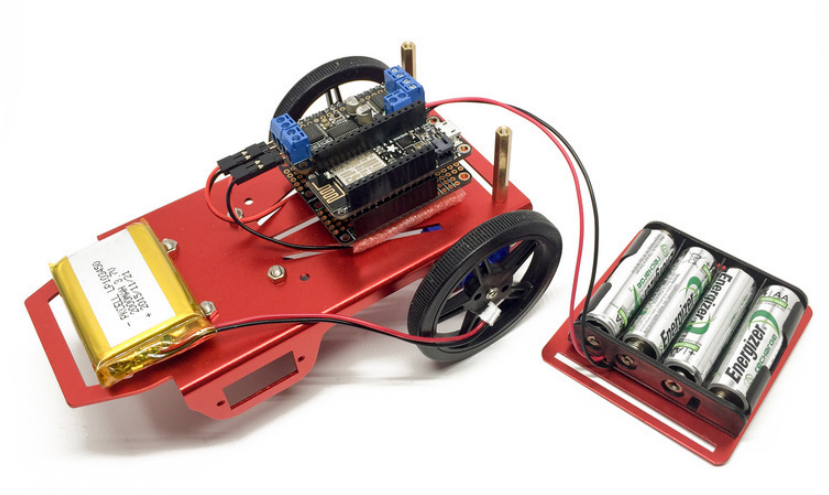
You can basically follow this guide till you obtain a result like on the following picture:

So far, you should have the motors and the wheels assembled on the robot, as well as the 3.7V LiPo battery & the FeatherWing doubler mounted on the robot. We are going to use the battery to power the ESP8266 WiFi chip and the Motor FeatherWing, and we'll use the FeatherWing Doubler to mount all the feather boards on the robot without using a lot of vertical space.

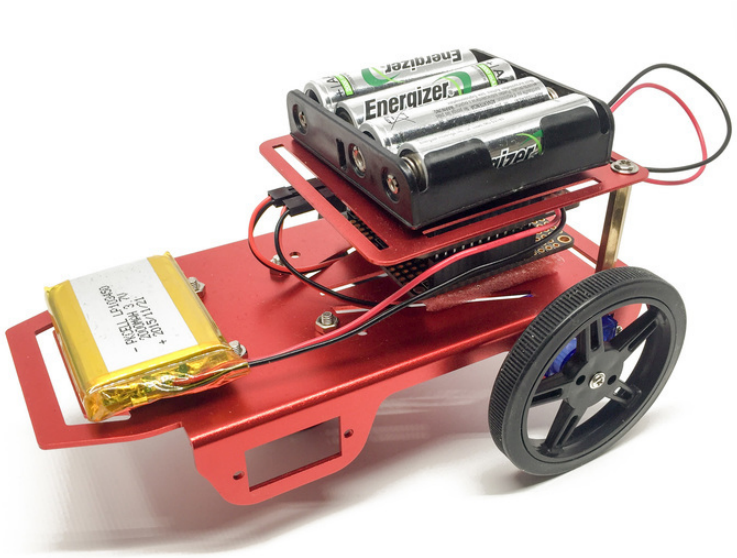
Now, we are going to take care about the motors of the robot. These will be powered by a larger power supply, that can drive the motors faster. First place four batteries (1.2V to 1.5V Alkaline or NimHA AA) into the 4 x AA battery holder, and then connect the battery holder to the motor FeatherWing component just as on the picture:



You can now place this FeatherWing on the FeatherWing doubler, as well as the ESP8266 Feather HUZZAH board. Also connect the two stepper motors to the motor FeatherWing. This should be the result at this stage:



Finally, mount the second stage of the robot using the metallic spacers, and also mount the battery holder on top of the robot:



Note that on the last picture, I already connected the 3.7V LiPo battery to the ESP8266 feather board. However, you can wait until the robot is fully configured to connect the battery.

Congratulations, you just assembled your mobile robot based on the ESP8266 WiFi chip! In the next section, we are going to learn how to configure it so it can receive commands via WiFi.

Configuring Your Mobile Robot

We now need to configure the ESP8266 WiFi chip on our robot so it can receive commands via WiFi. For that, we are going to use the aREST framework, which is a very convenient way to make the ESP8266 be completely controllable via WiFi.

For this section, you will need the latest version of the Arduino IDE, along with the following libraries:

- aREST (https://adafru.it/dis)
- Adafruit_MotorShield (https://adafru.it/ptb)

You can find more information about how to install an Arduino library by following this guide:

https://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use/arduino-libraries (https://adafru.it/dNR)

Let's now have a look at the code for this project. It stars by including the required libraries:

```
#include "ESP8266WiFi.h"
#include &lt;aREST.h&gt;
#include &lt;Wire.h&gt;
#include &lt;Adafruit_MotorShield.h&gt;
```

After that, we create an instance of the Adafruit_MotorShield, and also create instances for the two motors:

```
// Create the motor shield object with the default I2C address
Adafruit_MotorShield AFMS = Adafruit_MotorShield();

// And connect 2 DC motors to port M3 &amp; M4 !
Adafruit_DCMotor *L_MOTOR = AFMS.getMotor(4);
Adafruit_DCMotor *R_MOTOR = AFMS.getMotor(3);
```

We also create an instance of the aREST library:

```
aREST rest = aREST();
```

You will need to enter your WiFi network name & password inside the sketch:

```
const char* ssid = "wifi-name";
const char* password = "wifi-pass";
```

We also declare a set of functions to control the robots:

```
int stop(String message);
int forward(String message);
int right(String message);
int left(String message);
int backward(String message);
```

Inside the setup() function of the sketch, we initialise the Adafruit_MotorShield library:

```
AFMS.begin();
```

We also expose all the control functions to the aREST API, so we can call them via WiFi:

```
rest.function("stop", stop);
rest.function("forward", forward);
rest.function("left", left);
rest.function("right", right);
rest.function("backward", backward);
```

We also connect the ESP8266 WiFi chip to your local WiFi network:

```
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");

// Start the server
server.begin();
Serial.println("Server started");

// Print the IP address
Serial.println(WiFi.localIP());
```

Inside the loop() function of the sketch, we process incoming connections with aREST:

```
// Handle REST calls
WiFiClient client = server.available();
if (!client) {
  return;
}
while(!client.available()){
  delay(1);
}
rest.handle(client);
```

Let's now have a look at the implementation of the functions used to control the robot. For example, this is the function used to make the robot move forward:

```
int forward(String command) {

  // Stop
  L_MOTOR-&gt;setSpeed(200);
  L_MOTOR-&gt;run( FORWARD );

  R_MOTOR-&gt;setSpeed(200);
  R_MOTOR-&gt;run( FORWARD );

}
```

In a similar fashion, here the implementation of the function to make the robot turn right:

```
int right(String command) {

  // Stop
  L_MOTOR-&gt;setSpeed(100);
  L_MOTOR-&gt;run( FORWARD );

  R_MOTOR-&gt;setSpeed(100);
  R_MOTOR-&gt;run( BACKWARD );

}
```

Note that you can find the complete code inside the GitHub repository of the project:

https://github.com/openhardwarerobots/esp8266-robot (https://adafru.it/ptc)

It's now finally time to configure the robot! First, grab all the code from the GitHub repository of the project, and modify it with your own WiFi name and password. Then, upload the code to the ESP8266 feather board. Once that's done, open the Serial monitor, you should see the IP address of the board:

Make sure you copy and paste this IP somewhere, you'll need it in the next section where we'll configure the interface to control the robot.

Controlling the Robot Remotely

We now have a robot that can accept commands via WiFi, but we definitely don't want to have to type any of commands inside a web browser: it would be way to slow to control a robot! That's why we are now going to create a nice graphical interface to control the robot.

This interface will be based on aREST.js, a JavaScript library which is very convenient to control aREST-based projects. It will be automatically imported by the interface we are going to create in a moment, so you don't need to worry about it. If you want to learn more about aREST.js, you can visit the GitHub repository of the library:

https://github.com/marcoschwartz/aREST.js (https://adafru.it/pte)

Let's first have a look at the HTML file of the interface. Inside thetag, we import all the required files for the interface:

```
      &lt;head&gt;
    &lt;meta charset=utf-8 /&gt;
    &lt;title&gt;aREST.js Demo&lt;/title&gt;
    &lt;link rel="stylesheet" type="text/css" href="https://maxcdn.bootstrapcdn.com/
bootstrap/3.3.4/css/bootstrap.min.css"&gt;
  &lt;link rel="stylesheet" type="text/javascript" href="style.css"&gt;
    &lt;script type="text/javascript" src="https://code.jquery.com/
jquery-2.1.4.min.js"&gt;&lt;/script&gt;
    &lt;script type="text/javascript" src="https://cdn.rawgit.com/Foliotek/AjaxQ/
master/ajaxq.js"&gt;&lt;/script&gt;
    &lt;script type="text/javascript" src="https://cdn.rawgit.com/marcoschwartz/
aREST.js/master/aREST.js"&gt;&lt;/script&gt;
    &lt;script type="text/javascript" src="script.js"&gt;&lt;/script&gt;
&lt;/head&gt;
```

Now, inside the same file, we define a button for each of the commands of the robot, for example to make the robot go forward:

```
&lt;div class='row'&gt;

    &lt;div class="col-md-5"&gt;&lt;/div&gt;
  &lt;div class="col-md-2"&gt;
    &lt;button id='forward' class='btn btn-primary btn-block'
type="button"&gt;Forward&lt;/button&gt;
  &lt;/div&gt;
    &lt;div class="col-md-5"&gt;&lt;/div&gt;

&lt;/div&gt;
```

We still need to link the buttons inside the interface to the actual commands of the robot. This will be done in a file called script.js, which will make the link between the graphical interface & the robot.

The file starts by defining the IP address of the robot, and by creating an instance of an aREST device:

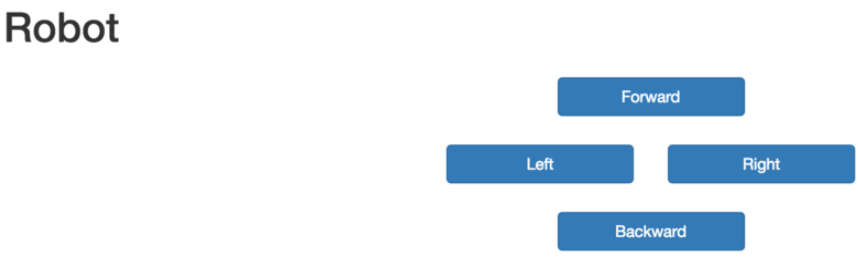
```
var address = "192.168.0.104";
var device = new Device(address);
```

Then, for each of the buttons of the interface, we call the corresponding function on the robot. Also, as we want the buttons to behave like push buttons (meaning whenever the button is released, the robot stops), we also need to call the stop function when the button is released:

```
$('#forward').mousedown(function() {
  device.callFunction("forward");
});
$('#forward').mouseup(function() {
  device.callFunction("stop");
});
```

It's now time to finally test our interface and make our robot move around! For that, make sure to edit the script.js file inside the interface folder, and put the actual IP address of your ESP8266 WiFi chip. If that's not done yet, also disconnect the ESP8266 feather board from USB, and power the ESP8266 using the 3.7V LiPo battery.

Then, open the interface with your favorite web browser. This is what you should see:



As you can see, there is a button for each function of the robot. You can now try it: whenever you press a button (and keep it pressed), your robot should move immediately!

This is an example of my own mobile robot moving around while I was using the interface:

Congratulations, you built your own mobile robot based on the ESP8266 and controlled it via WiFi! Note that you also control the robot using a mobile device, like a smartphone or tablet, using the exact same interface.

How to Go Further

In this guide, we learned how to build a mobile robot based on the ESP8266 WiFi chip, and on the Adafruit Mini Robot Rover Chassis Kit. We first assembled the robot, and configured it so it accepts commands via WiFi. We then controlled the robot using a graphical interface running in your web browser.

There are of course many ways to now improve the project, based on what you learned in this guide. You could for example add a distance sensor in front of the robot, and have the measured distance displayed inside the same interface. You could also couple a gyroscope to the robot, and have more complex functions like making the robot turn from a given angle. The possibilities offered by this excellent chassis & the ESP8266 WiFi chip are endless, so don't hesitate to experiment and have fun!