Pirate Ship --- Step-by-Step Robot Tutorial

Are you excited about building your first robot? Isn't it cool to make your own robot, which can move forward, move back, turn left and turn right? In this tutorial, you can learn all the basics: how to use the tools, how to assemble your own robot, how to program and how to make your robot move. Before we start building, let's following tools.

Essential Tools

- **Screwdriver**

Screwdriver, a tool for turning screws (driving or removing) is indispensable for building a robot. We suggest that you get yourself a screwdriver set, which is extremely convenient in dealing with screws of different types and sizes.

- **Soldering Pencil**

A soldering pencil is a tool frequently used in electronics projects for soldering and de-soldering work on circuit boards. We'll need it in this project to solder the motors' cables.

NOTE: Once heated, the tip of the soldering pencil will become extremely hot. If you have never used a soldering pencil before, please spend some time familiarizing yourself with how to use the pencil. BE CAREFUL when using the soldering pencil!

- **Needle-nose Pliers**

Needle-nose plierse are often used to cut off excess length from cables and wires. We won't be using the pliers much for this project; however, they're an essential tool in electronics projects.

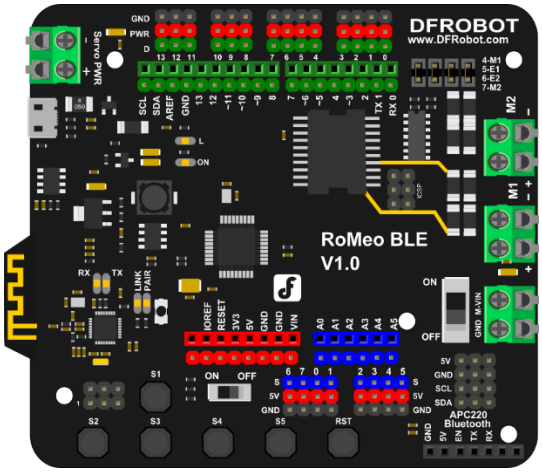
- **Wire Stripper**

A wire stripper is a hand-held tool used to strip the electrical insulation off of wires. The wire stripper can also be replaced with scissors if needed. When using this tool, only the first layer of insulation needs to be stripped, exposing the lead wire inside. This helps make the soldering process easier.

Hardware Parts

- Car Base × 1

- Romeo BLE controller × 1



- Micro USB cable × 1

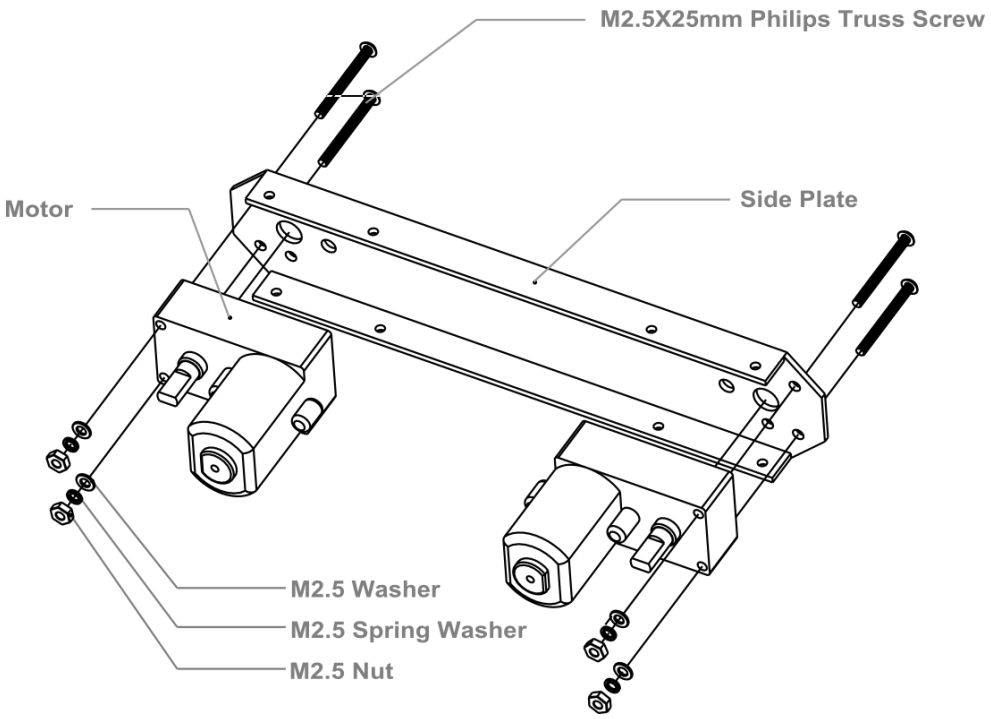
Assembly Instructions

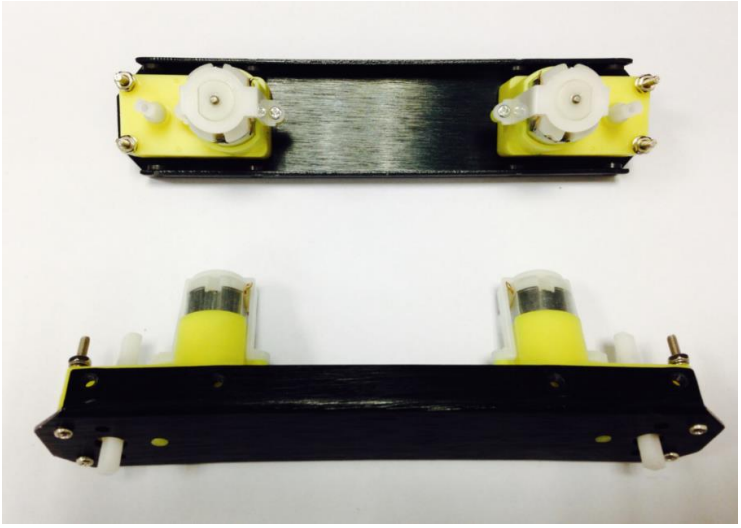
After preparing the necessary tools and hardware pieces, it's time to get started! Assembling the Pirate's mobile platform is straightforward — just follow the below instructions step-by-step.

STEP1: Assemble Your Own Motor

Look in your parts bag for eight long screws. These are used to fix and secure the motors in place. Place the motors in the correct alignment, then screw them into place as shown in the picture below.

Please note that washers and gaskets are also included in the parts bag. Washers can be used to increase friction, which helps fasten the motors into place. The gaskets help prevent the screw nuts from loosening and falling off due to the Pirate's movements and collisions.

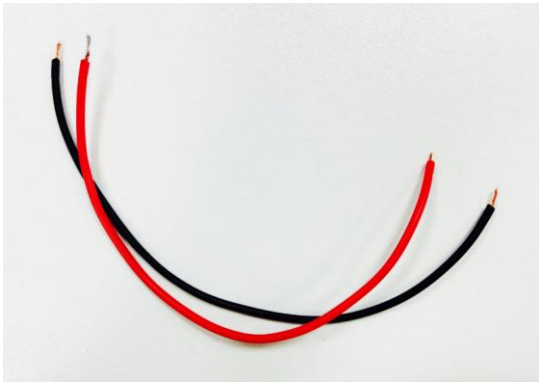


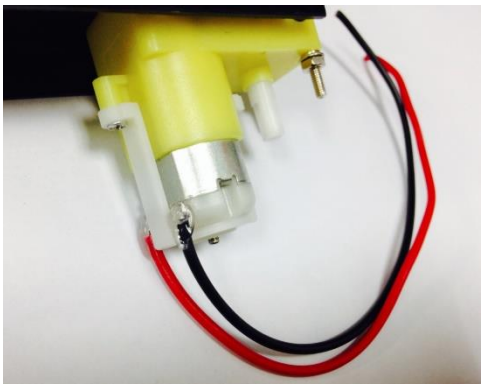


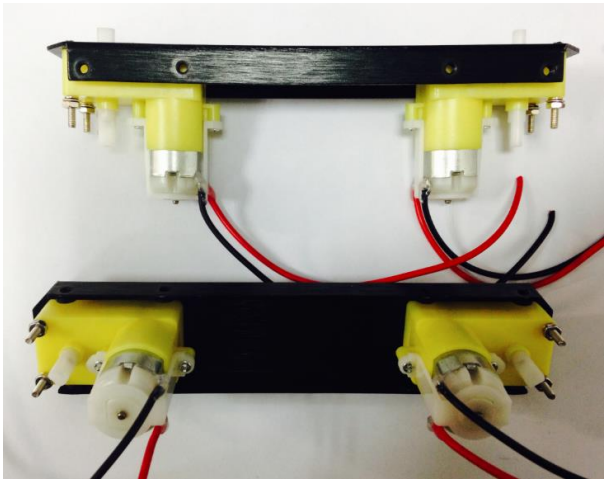
STEP2: Soldering the Cables

Take the black and red wires out of the parts bag. Attach one black and one red cable (15 cm long) to each motor (4 motors in total). Then use your wire stripper to strip the insulation at both ends of the wires (make sure not to strip too much—refer to the pictures below). Next, solder the wires onto the pins affixed to the motors. Repeat the soldering process for all four motors.

NOTE: Pay attention to the correct locations of the red and black wires when soldering. Please consult the following photos for details.

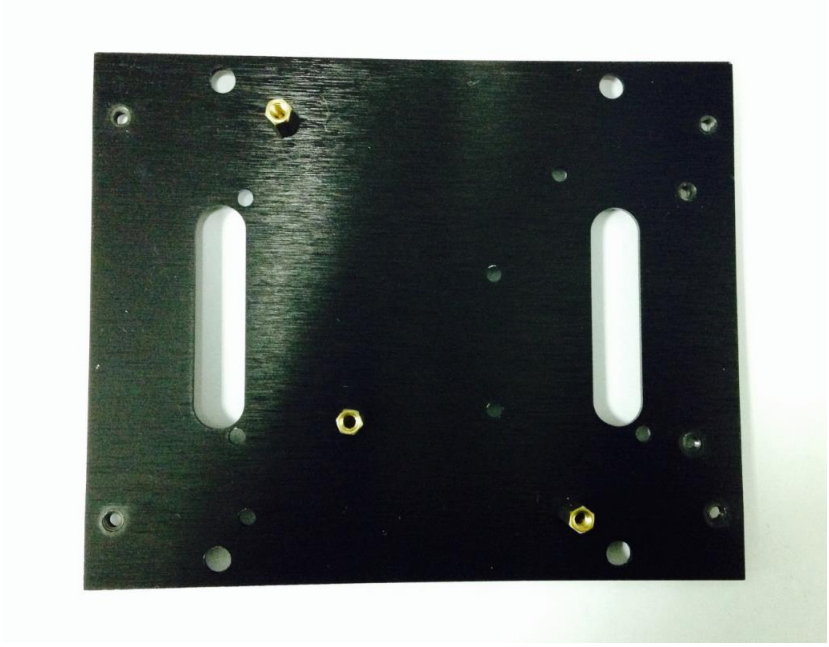


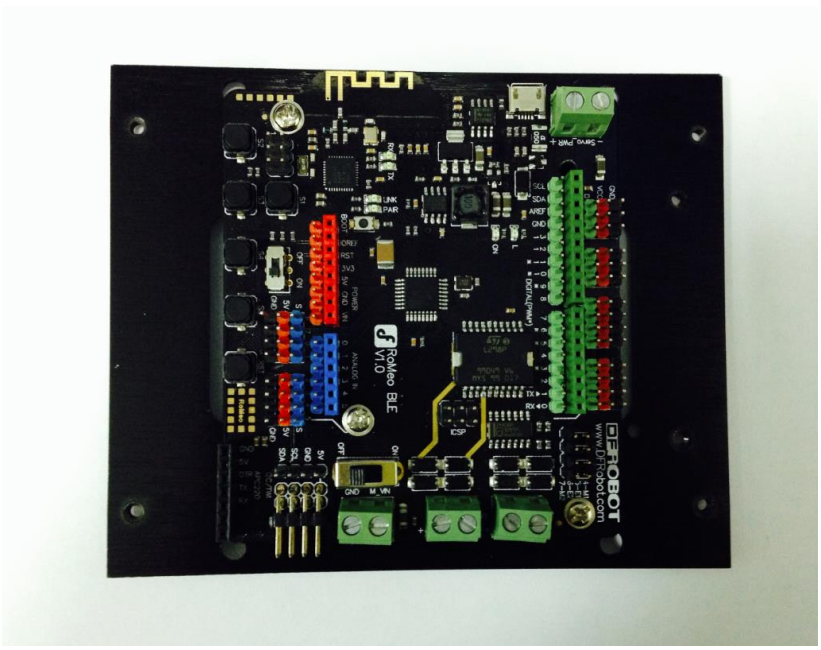




STEP3: Assemble the Romeo BLE controller

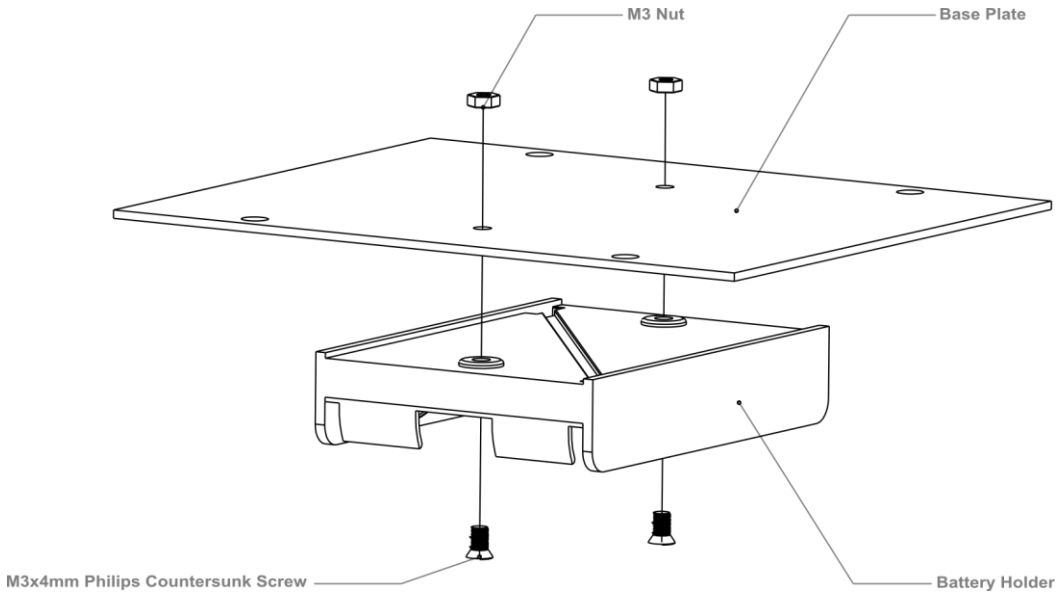
Look in your parts bag for three copper supports. Those 1cm-long supports are used to fasten the Romeo controller board. As shown in the picture below, there are three holes in the controller board. Place the three copper supports into the holes, then fasten them into place with the appropriate screws.

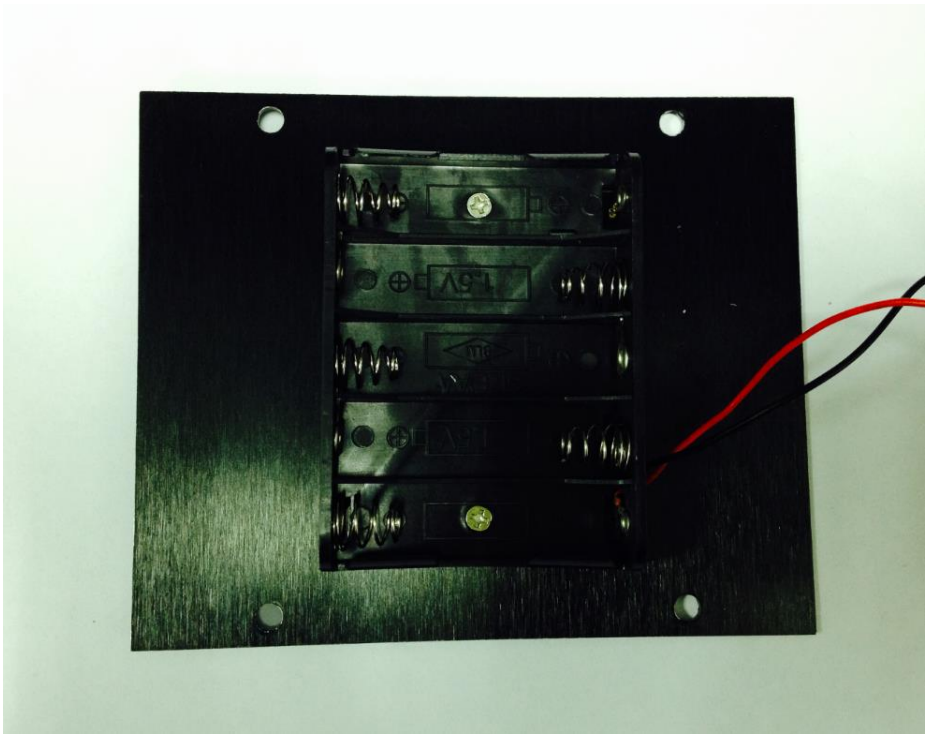




STEP4: Assemble the Battery Box

Take out two countersunk screws (their heads are flat). Then follow the steps shown in the picture below and affix the battery to the car base.

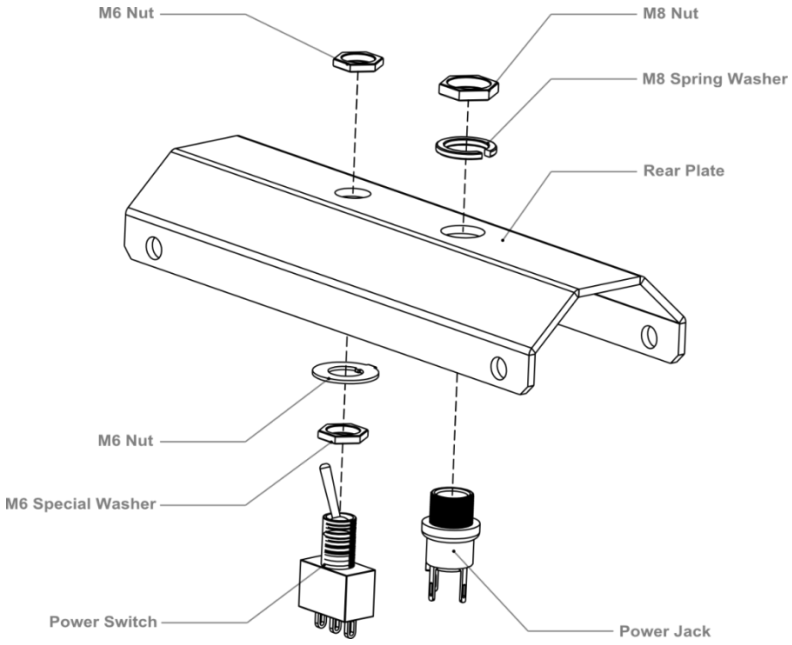




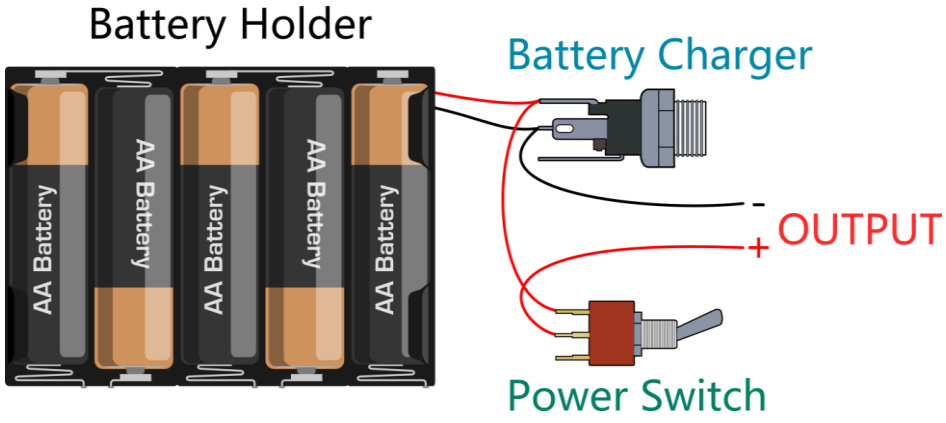
STEP5: Crafting the Power Switch

Batteries are the essential lifeblood of robots. To control power usage, we need to use a power switch: the switch turns off power when not in use, thus preserving electricity and battery life. Refer to the picture below before assembling and installing the power switch.

Please pay attention to the sequence of the gaskets and screw nuts when assembling the switch.

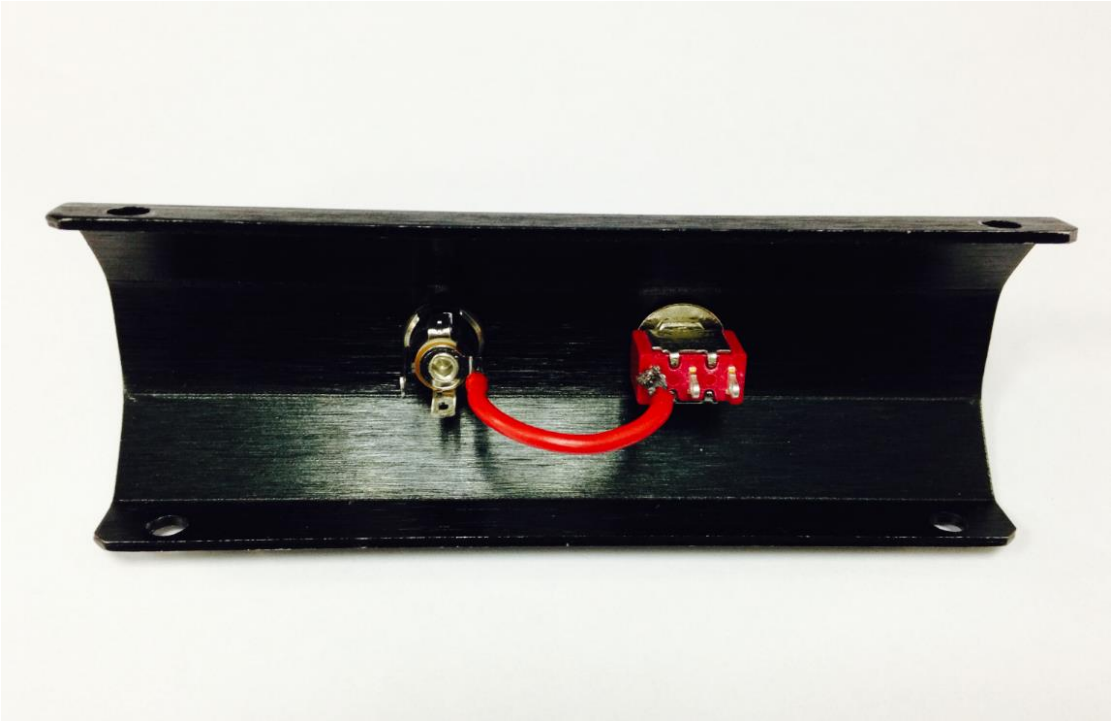


After assembling the switch, we want to start soldering its wires. Take some of the remaining wire leftover from before. Strip the wiring off both ends of the cables so that the inside of the wire is exposed (same process as with the motors before). We want to solder the exposed end of the wires to the pins on the switch. When soldering, it's very important that we note the position of the switch's pins.

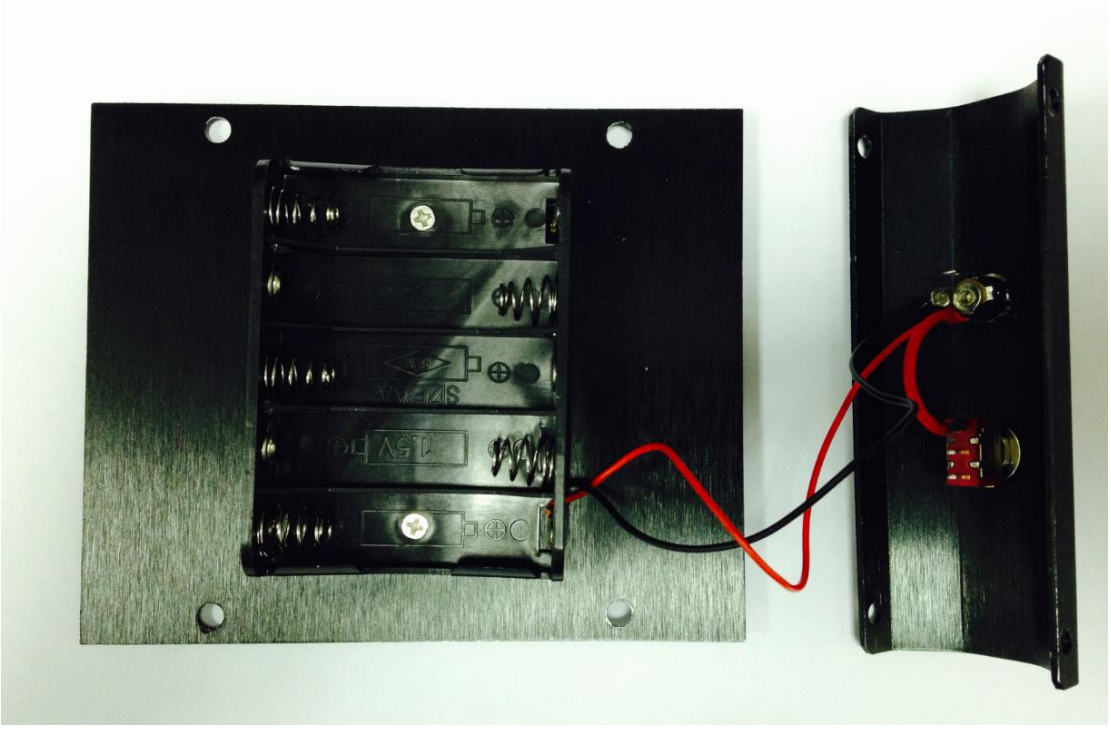


Let’s do this step by step.

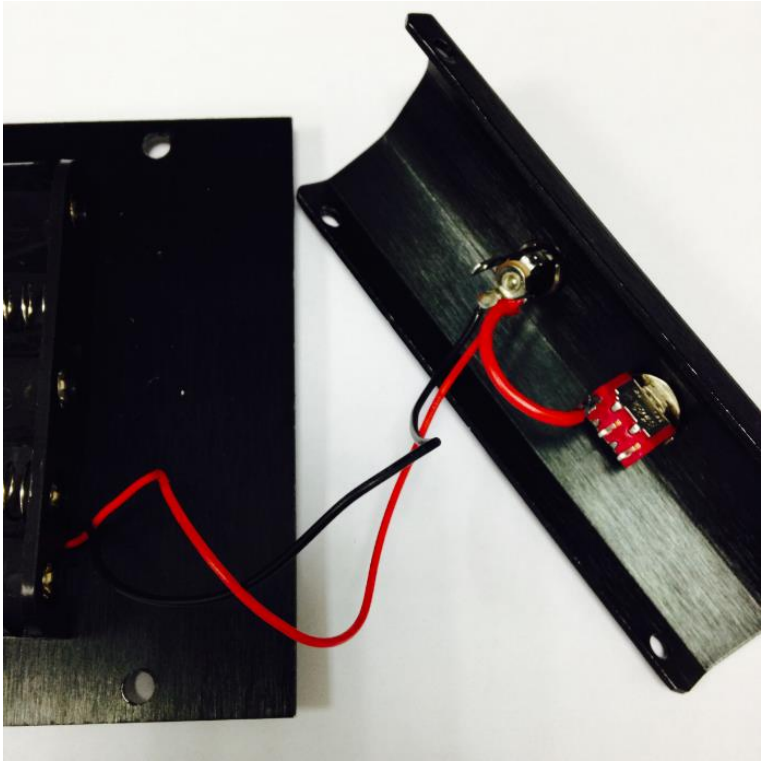
a) Connect the switch to the battery charger. Pay attention to the exact location of both items.



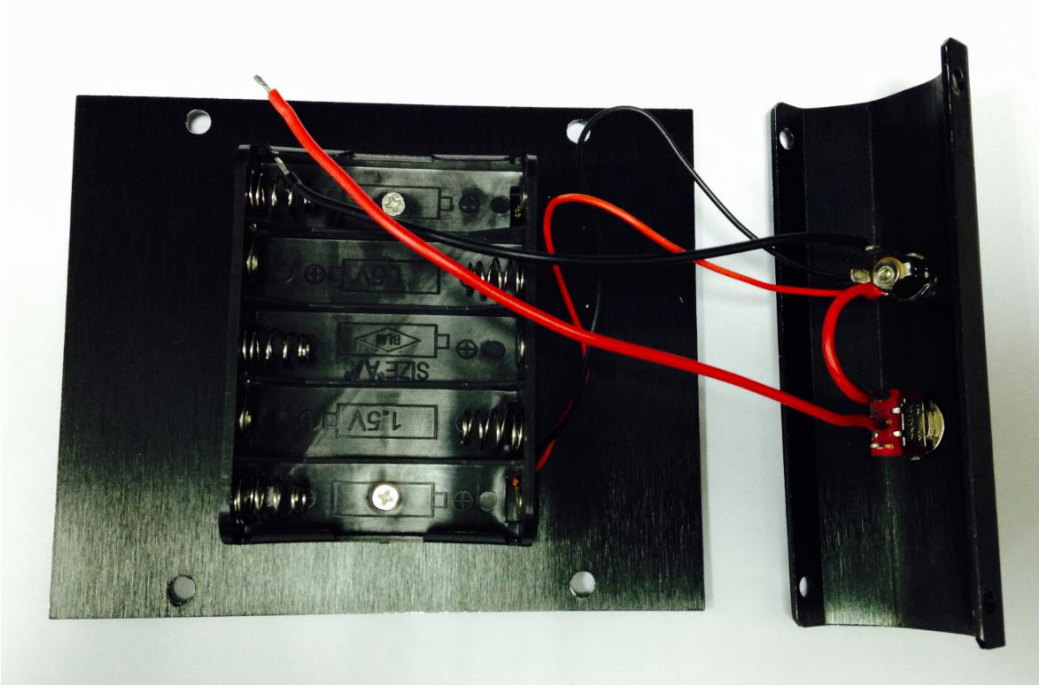
b) Solder the red cables connecting the switch with the battery charger as shown in the picture below.

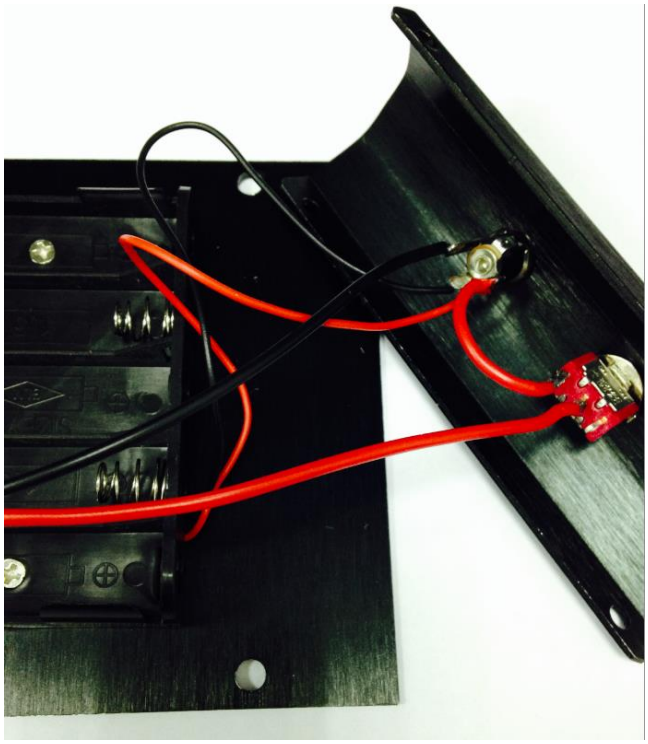


Here’s another picture to make things clearer.



c) Finally, take one red cable and one black cable. Attach one end of one cable to the negative pole of the battery charger and one end of the other cable to the positive pole of the battery charger. Then attach the other ends of both cables to the Romeo BLE controller.



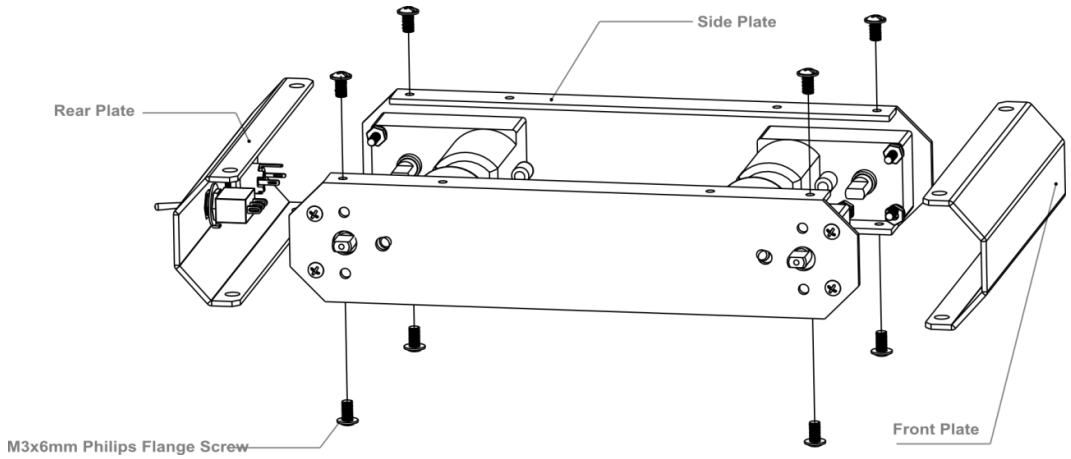


Looking at this enlarged picture should give you a better idea of how the wires should be connected. After soldering, make sure to check and see if your wiring between the battery and Romeo controller is consistent from start to finish and matches with the above pictures.

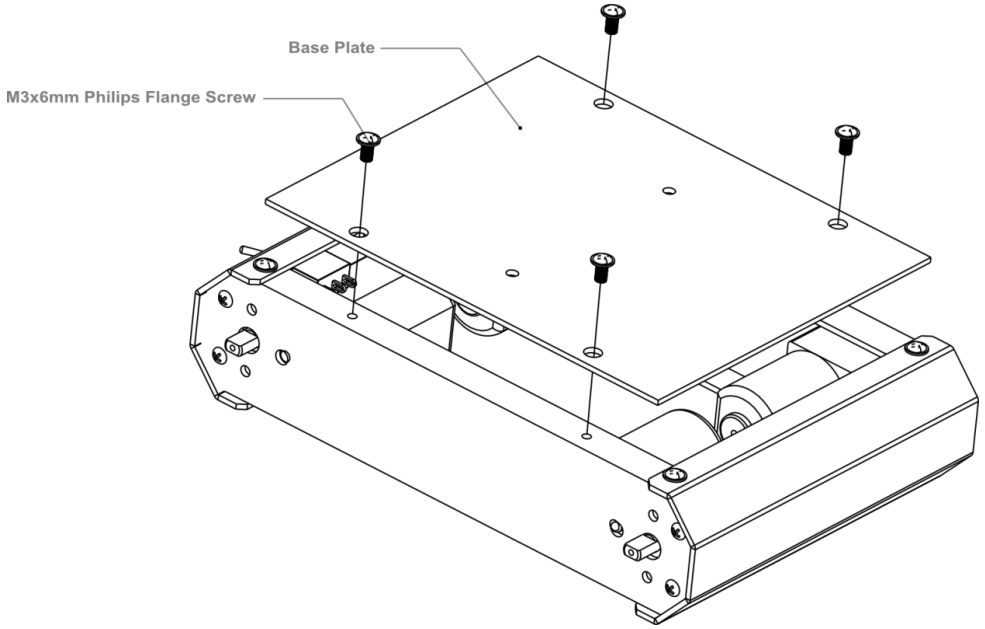
STEP6: Assemble the Car Base

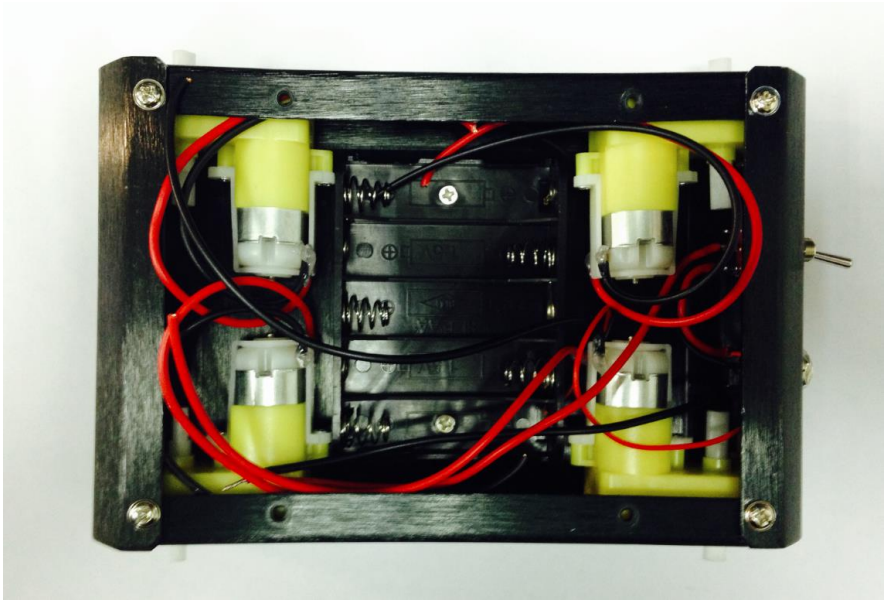
Using eight M3x6mm screws, attach the side plates to the front and back bumper plates as shown by the diagram below.

NOTE: When tightening the screws during this step, make sure not to fully tighten the screws at first — this way, we can easily detach the top board in later steps should we need to make adjustments.



Then, re-attach the base plate to the body of the car as shown in the picture below.



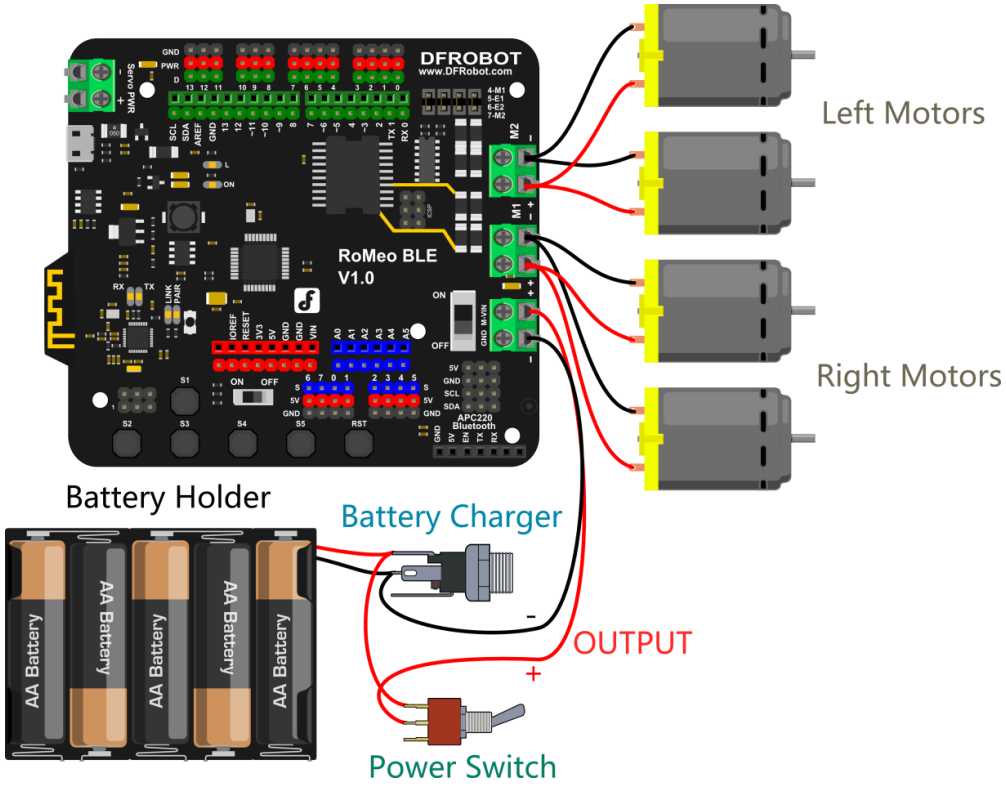


** This is what the car base should like after it's been assembled — remember to install the battery pack!.

STEP7: Connect the Motors with the Microcontroller Board

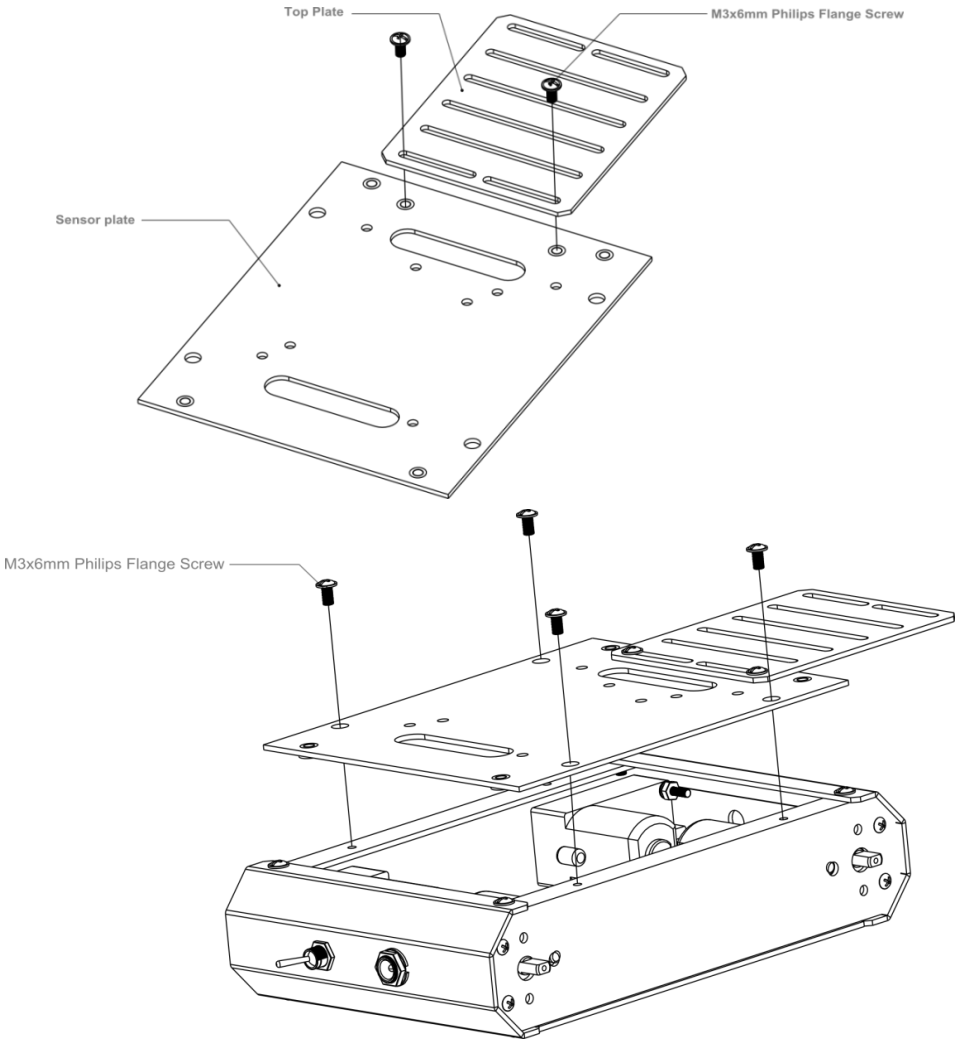
Now we need to the motors with the microcontroller board. Carefully follow the following diagram: the left motor's red and black wires should be soldered into M2; the right motor's red and black wires should be soldered to M1. Pay special attention to the battery pack: the black wire should be soldered into the wire port reading GND, while the red wire should be soldered in the wire port labeled VND. Use your screwdriver to loosen and tighten the wire ports — make sure these ports are fastened well once the wires have been inserted.

NOTE: Make sure the wires from one motor (i.e. the left motor) are soldered into the motor port. (i.e. the M2 port on the diagram below — do **not** solder one motor's wires into two separate ports.)

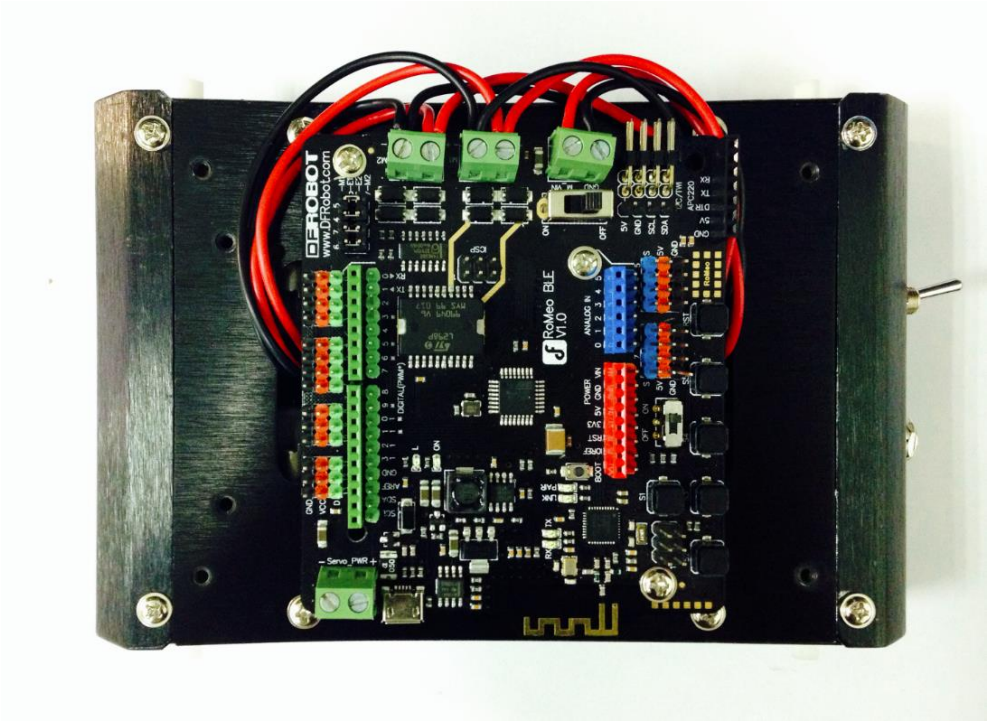


After soldering the motor wires to the microcontroller board, we're ready to attach the top plate to the base of the car.

Before we attach the top plate, you have the option of attaching a sensor plate (see diagram below) — if you don't plan to use sensors just yet, you can skip this extra step.

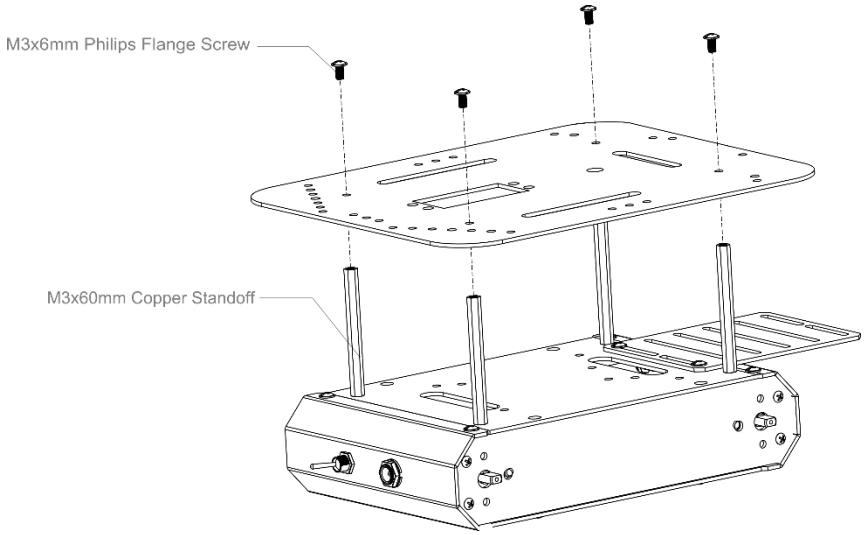


After attaching the top plate, your Pirate should resemble the picture below.

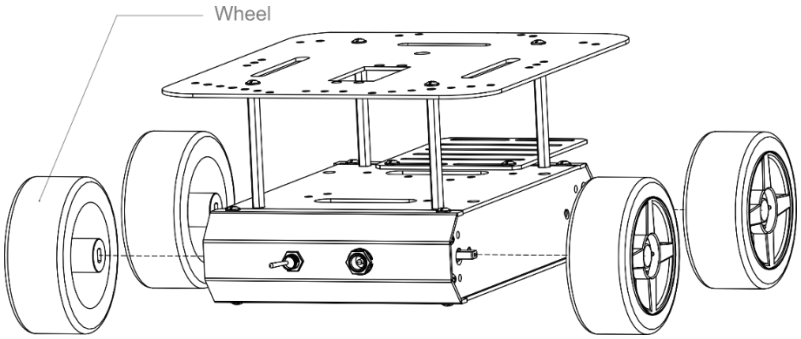


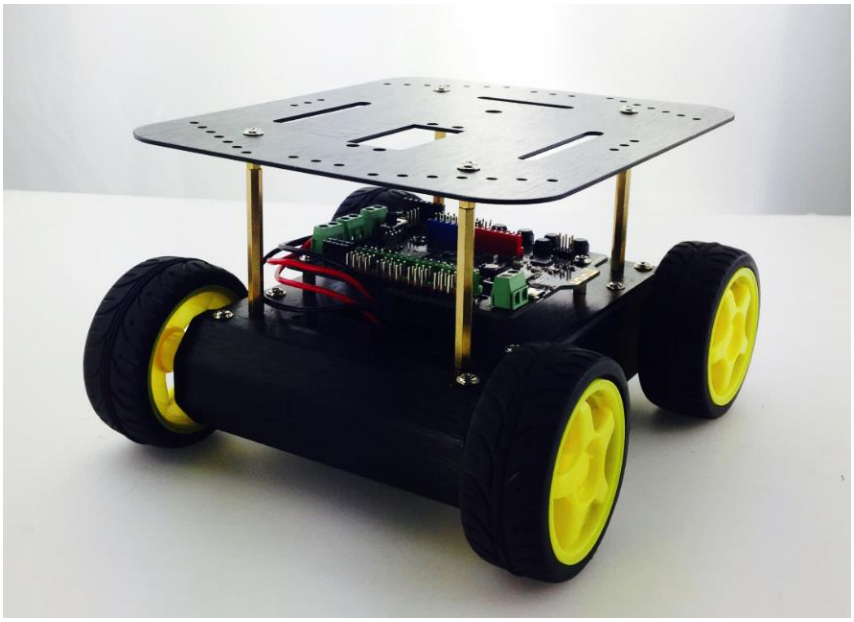
STEP8: Attach an extra level to the Pirate

Find the four holes on the base's top plate. Screw in the four M3x60mm Copper Standoffs, then attach the additional top plate as shown in the diagram below — use M3x6mm screws to affix the plate to the copper standoffs.



Toss some wheels on your Pirate and you're ready to let it whip!





CODING

After assembling, it's time to upload some code onto the microcontroller and get the Pirate moving. The robot has all the components for moving once assembled. Look through the sample codes for the Arduino file titled "MotorTest.ino".

Sample code MotorTest：

```
#include <DFMobile.h>
DFMobile Robot (4,5,7,6);     // initiate the Motor pin

void setup () {
  Robot.Direction (LOW,HIGH);  // initiate the positive direction
}

void loop () {
  Robot.Speed (255,255);      //Forward
  delay (1000);

  Robot.Speed (-255,-255);    //Back
  delay (2000);
}
```

Download the code, then upload it to your microcontroller. The motors and wheels should come alive in a hurry. If not, check to see if your batteries and power switch are properly installed. Once the motors are working, congrats! You've completed a big step — it's almost time to put our rubber to the road.

Then observe your robot car and check if it may move forward within 1 second and move back within 1 second. If that's the case, GOOD LUCK. You don't have to adjust the components. For those who need to make some adjustments to the car base or motors, please find the following information about how the robot moves.

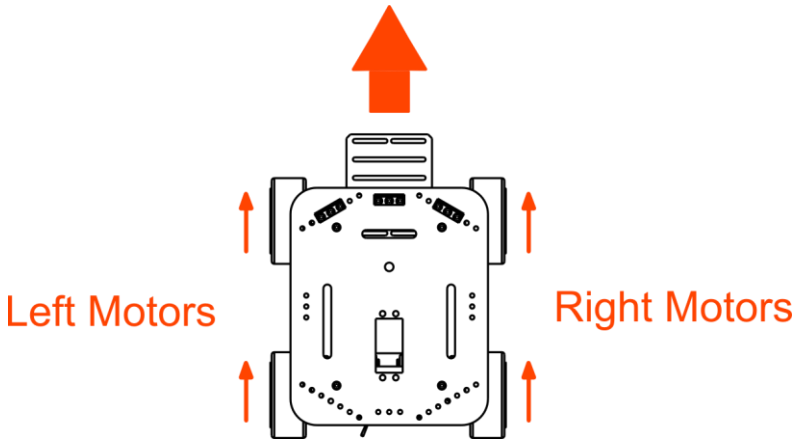
Check to see if your Pirate follows the code shown above: it should move forward for 1 second, then reverse for 1 second. If that's the case, just skim the content below and then you're ready to go!

Most people, however, will need to make adjustments to their motors. Before we do that, let's briefly review how our Pirate's motor function and code works.

How to Make the Robot Move Forward?

In order to understand this question, let's first examine our robot's forward movement.

The diagram below illustrates this forward movement.



The red arrow above represents the direction of the wheels. As it is shown in the map above, the car can move forward only if both the left and right wheels/motors are moving forward. As shown above, the Pirate only moves forward when both the left and right motors and wheels are moving forward.

Code Synopsis

The first line of code is:

```
#include <DFMobile.h> // call library
```

We don't need to think too much about this line. All we're doing is calling upon/employing a set of functions — the DFMobile library — that exist outside of Arduino's basic framework. For more information on Arduino libraries, please check out the Arduino website.

The next line of code is:

```
DFMobile Robot (4,5,7,6);        // initiate the Motor pin
```

This function is taken from the DFMobile library (that is, it's not a universal Arduino function). We use it here to initialize the motor pins (4, 5, 7, 6) on the microcontroller — without this, the motors won't start.

We'll be using this function later on as well.

Take a look at the function below:

```
DFMobile Robot
 (EnLeftPin,LeftSpeedPin,EnRightPin,RightSpeedPin);
```

This function is used to initialize the four motor pins (4, 5, 7, 6) and is divided into four separate parameters:

EnLeftPin： Pin that controls left motor direction
LeftSpeedPin： Pin that controls left motor speed
EnRightPin： Pin that controls right motor direction
RightSpeedPin： Pin that controls right motor speed

Please note: the Pirate's motors won't run without the inclusion of this function. Also, this function must be placed within the void setup() field in your Arduino sketch.

In testing the Pirate's forward motion before, we might've encountered a certain problem: the car will start to drift, change directions and not quite follow the code we've given it. This is due to the motor wires not being soldered to the batteries in the correct manner.

Don't worry — we can correct this through code. By using LOW/HIGH values, we can adjust the direction of the car's turns.

How to Adjust the Straight Direction for the Robot Car?

To adjust the direction of the motors and wheels, we need the following line of code:

```
Robot.Direction (LOW,HIGH);
```

The function is as follows:

```
Robot.Direction (LeftDirection, RightDirection);
```

This function is used to make the motors move in a forward direction. The function is divided into two parameters: LeftDirection & RightDirection, which are written in the Arduino code as either LOW or HIGH.

Earlier, we briefly went over how to make the Pirate move in a forward direction. Here, we'll use LOW/HIGH to correct the Pirate's wayward movement.

For instance, LeftDirection is set as LOW in the sample code. But left wheels of the robot car may rotate backwards instead of rotating forward. Now all you have to do is to change the LeftDirection from LOW to HIGH. The same methods would apply to the right wheels.

For example: in this sample code, LeftDirection is configured as LOW. Suppose your left wheels, rather than moving forward as they should, instead move backwards. In

this case, change the configuration of LeftDirection from LOW to HIGH. Once you change it to HIGH, upload your code once more — you should notice that your left wheel is moving forward now instead of backwards. If this adjustment works, do the same for RightDirection (LOW to HIGH or vice versa).

Once you've successfully adjusted your Pirate's direction, you're set! Congratulations — you can now use all of the Pirate's basic functions. Before finishing up, though, it's worth it to briefly discuss the Robot.Speed() function.

Take a gander at the following function:

```
Robot.Speed (LeftSpeed,RightSpeed);
```

This function with two elements (LeftSpeed and RightSpeed) is used to set the speed of the motor. You may write a number between -255 and 255. 255 is the maximum figure and the minus sign represents the direction.
This function is used to configure the speed of the motors. The function is divided into two parameters: LeftSpeed & RightSpeed. These parameters are written in Arduino code as a value ranging from -255 to 255. 255 is the fastest velocity moving forward; -255 is the fastest velocity moving backwards (that is, reversing).

We already configured the Pirate's speed in the void setup() part of our code. Now, we can use the speed() function to control the car's speed and even forward/backwards direction.

See if you can understand the following two lines:

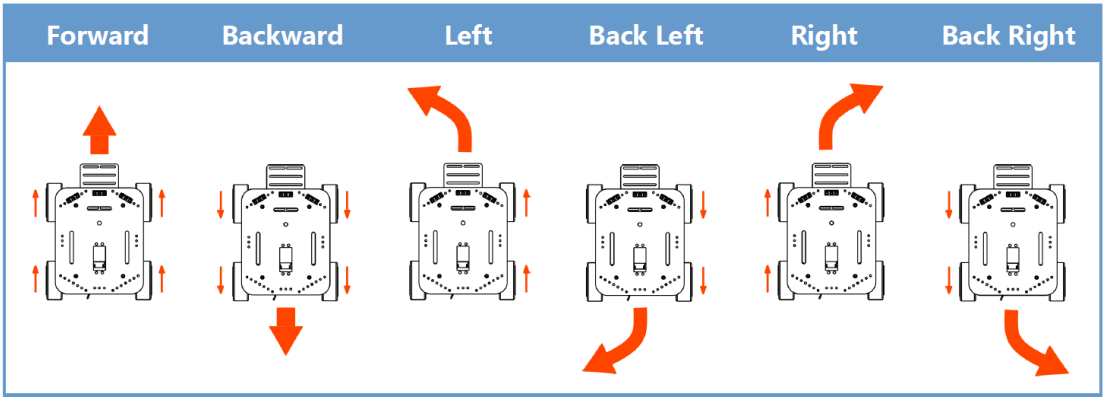
```
Robot.Speed (255,255);
Robot.Speed (-255,-255);
```

The first line shows the car moving forward at full velocity — full speed ahead, if you will (aye aye, captain). The second line shows the car moving backwards (reversing) at full velocity.
In this sense, speed() is an indispensable function. Next, we'll review our last section: the principles behind how the Pirate moves and turns.

How the Pirate moves and turns

The map below shows some regular way of movements for the robot car. For example, in case that speed of the Left Direction is zero, the robot would turn left if you offer the right wheels some force to move forward.

The following diagram displays a number of ways in which the Pirate can move and turn. For instance, if the speed of the left wheels is set to 0, that will cause the right wheels to move forward — thus, the Pirate would turn towards the left.



Something to consider: how can we make the Pirate rotate in circles while stationary?

Lastly: if you want, you can run some more code to test and calibrate your Pirate's movement. Open the "MotorTest2.ino" file. This code should help you better understand and gauge the capabilities of forwards and backwards movement, in addition to left and right turns. With this in mind, put those tires to the road (or carpet) and let'er rip!