

Contents

Exercise details	2
Exercise 1: Blinking an LED	3
Build the circuit	3
Requirements	3
Build the code in Robotnik	4
Build code in JavaScript	4
Going further	6
Exercise 2: Moving a servo	7
Build the circuit	7
Requirements	7
Build the code in Robotnik	8
Build code in JavaScript	8
Going further	10
Exercise 3: Spinning a motor	11
Build the circuit	11
Requirements	11
Build the code in Robotnik	11
Build code in JavaScript	13
Going further	14
Exercise 4: Control a wheeled robot	15
Build the circuit	15
Requirements	15
Build the code in Robotnik	15
Build code in JavaScript	17
Going further	18

Exercise 5: Going bluetooth	19
Build the circuit	19
Requirements	19
Build the code in Robotnik	20
Going further	20

Exercise details

This section holds information about how to build each of the circuits in the exercises and program them

Each circuit is broken out into its own file so you can review it easily with diagrams and code examples.

- [Exercise 1 - Blinking an LED](#)
- [Exercise 2 - Moving a Servo](#)
- [Exercise 3 - Spinning the Motor](#)
- [Exercise 4 - Control a wheeled Robot](#)
- [Exercise 5 - Switching to Bluetooth](#)

Exercise 1: Blinking an LED

An LED is a common method of creating light in a project. They come in a huge variety of colours and sizes.

In this exercise you'll blink an LED under software control. Blinking an LED is an important “Hello World” type example of how to use hardware as the circuit is very simple but being able to control a physical thing by using software illustrates many of the core concepts. Once you can do this then you can control other things in your environment as well.

Build the circuit

Requirements

- 1x Resistor (any size will do but the bigger the less bright it will be)
- 1x LED
- 1x Arduino
- 1x breadboard
- Jumper wires

Build the circuit below, taking care to note that an LED has polarity. This means it only works in one direction. The “long” leg is the positive side and it connects to your source of power through the resistor (which slows the flow of current and stops the LED from burning out). The “short” leg is the negative side and it is connected to ground.

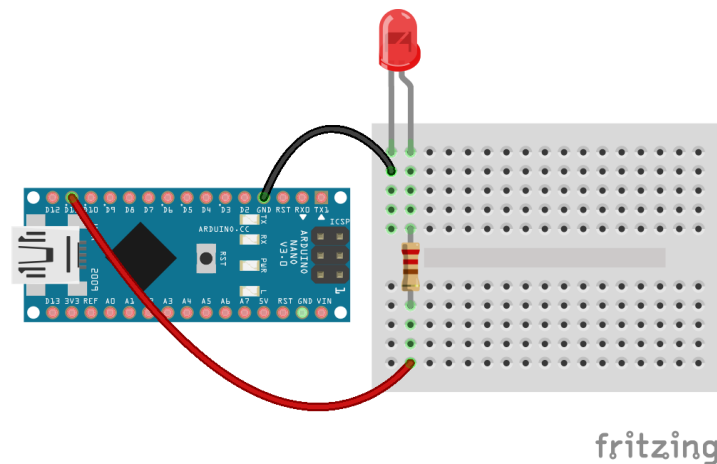


Figure 1: LED Circuit

Make sure the LED is connected to pin 11.

Once you've built the circuit, it's time to program it.

Build the code in Robotnik

Select the **Controller** from the toolbox then grab the controller block and drop it onto the workspace. By default it's wired up to the "red" button however you can change that to another button or the joystick controller if you want.

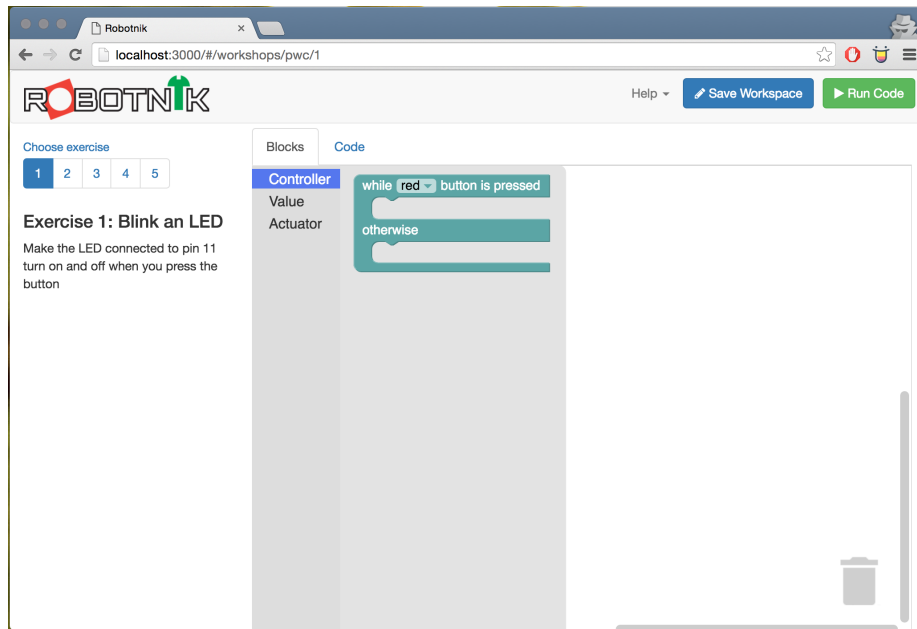


Figure 2: Select controller

Next select **Actuator** from the toolbox and add the LED block onto the workspace and connect it into the top of the controller press event block.

When the button is pressed, this will turn the LED ON which is what we want to happen. After that we add another LED block but this time change it's state to turn off instead.

That's looking pretty good. Save your workspace, run it and you should now be able to press the red button on the controller and it will turn your LED on. When you release the button it should turn it off.

Build code in JavaScript

To build a similar circuit in javascript to run from a console we can use an script such as the following.

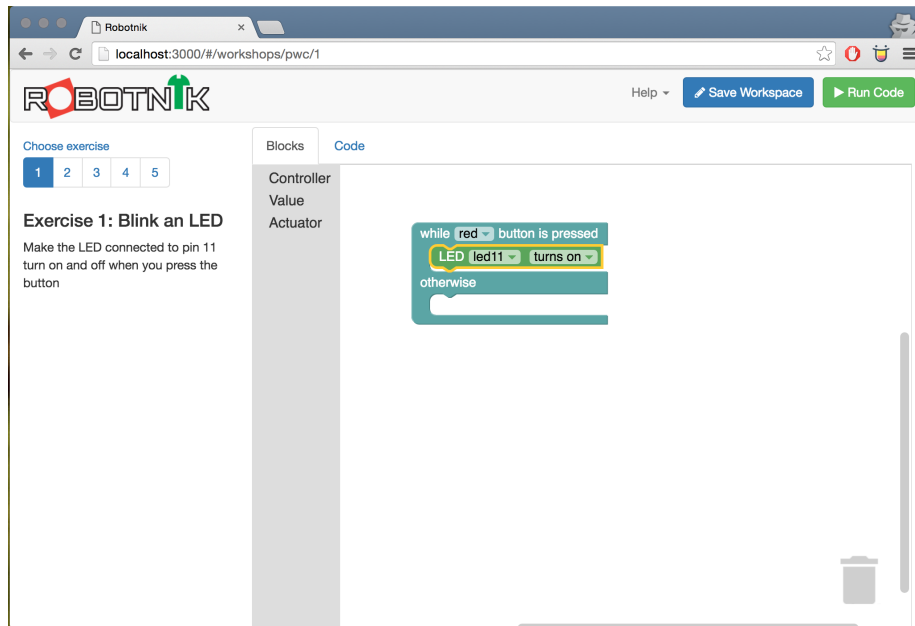


Figure 3: LED

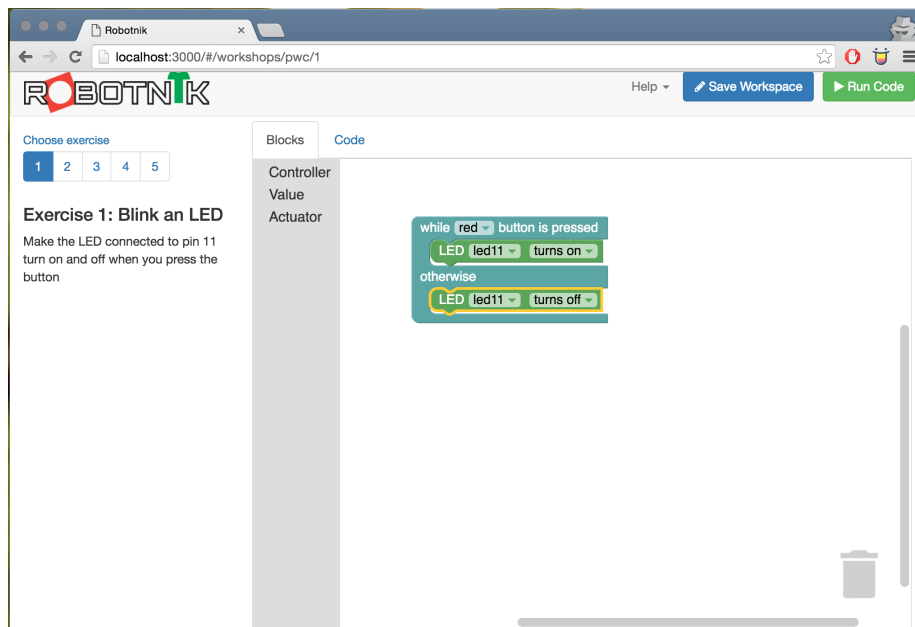


Figure 4: LED off

```
var five = require("johnny-five");

var board = new five.Board();

board.on("ready", function() {

  // Create a standard 'led' component instance
  var led = new five.Led(11);

  led.blink(500);
});
```

Make sure you have the Johnny-Five package installed and you can run it from the command line with

```
node code/led.js
```

Going further

Other things you can do:

- Try out other parts of the Led API such as blinking and stopping
- Pin 11 is a “PWM” pin which means you can use it to simulate analog voltage levels. Try out the fade part of the API.
- Move onto [Exercise 2 - Moving a Servo](#)

Exercise 2: Moving a servo

A servo is a small motor that has feedback on it to determine what position it has rotated to. Servos are used in a lot of robotics due to high torque and reasonable precision versus their cost.

In this exercise you'll move the servo to a position on the press of a button and when it's released it will move back to its starting position.

Build the circuit

Requirements

- 1x Servo
- 1x Arduino
- Jumper wires

A servo has 3 wires, voltage (usually red), ground (usually black or brown) and signal (usually yellow or white). The servo is permanently powered through voltage and ground and then the signal line is connected to a digital pin on the arduino which sends messages to it to tell it where to move to.

Small servos like this one can be run from the arduino when this is the only thing being powered but generally the power and ground would connect to a battery instead.

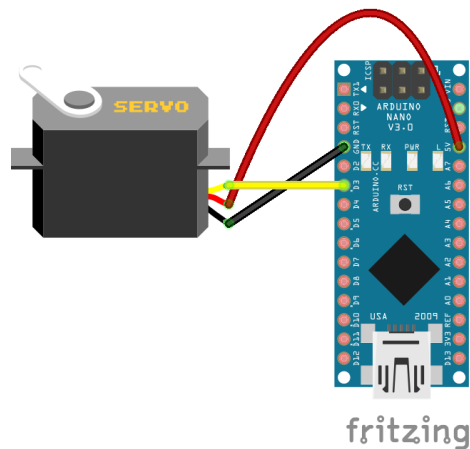


Figure 5: Servo Circuit

Make sure the Servo is connected to pin 3.

Build the code in Robotnik

Select the **Controller** from the toolbox then grab the controller block and drop it onto the workspace. By default it's wired up to the "red" button however you can change that to another button or the joystick controller if you want.

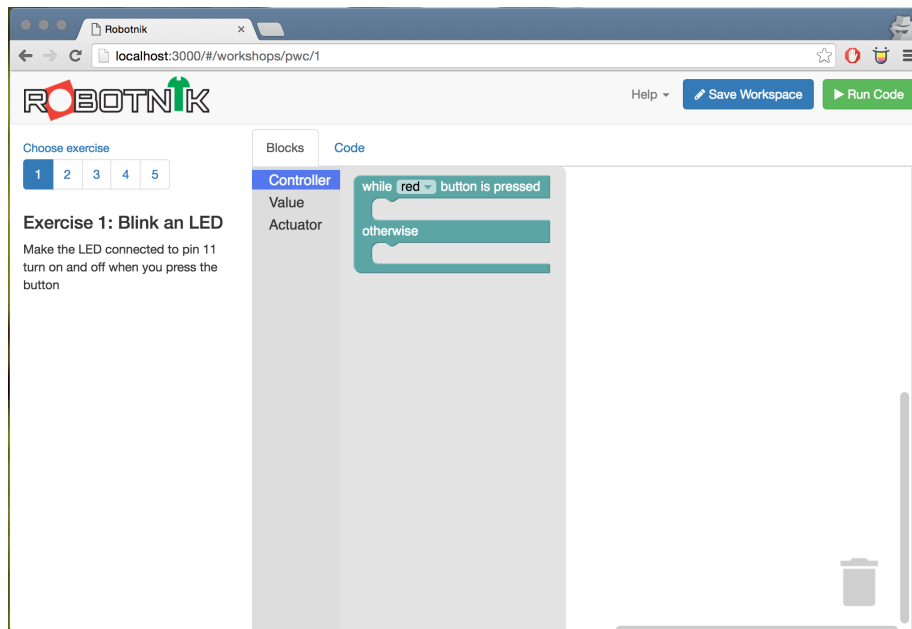


Figure 6: Select controller

Next select **Actuator** from the toolbox and add the Servo block onto the workspace and connect it into the top of the controller press event block.

When the button is pressed, this will move the servo to one extent of it's arc - 180 degrees in this case (most servos can move 180 degrees of a circle). When the button is released we want to move it back to 0 degrees.

When you run this code you should see the servo move through it's whole range and back again when the servo is released.

Build code in JavaScript

To build a similar circuit in javascript to run from a console we can use an script such as the following.

```
// Used from johnny-five.io examples
```

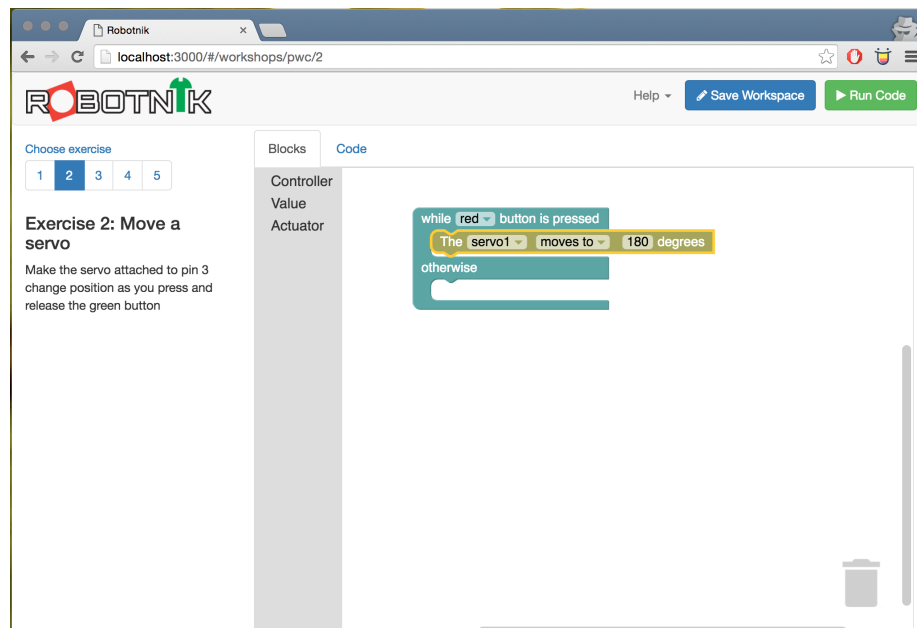



Figure 7: Servo

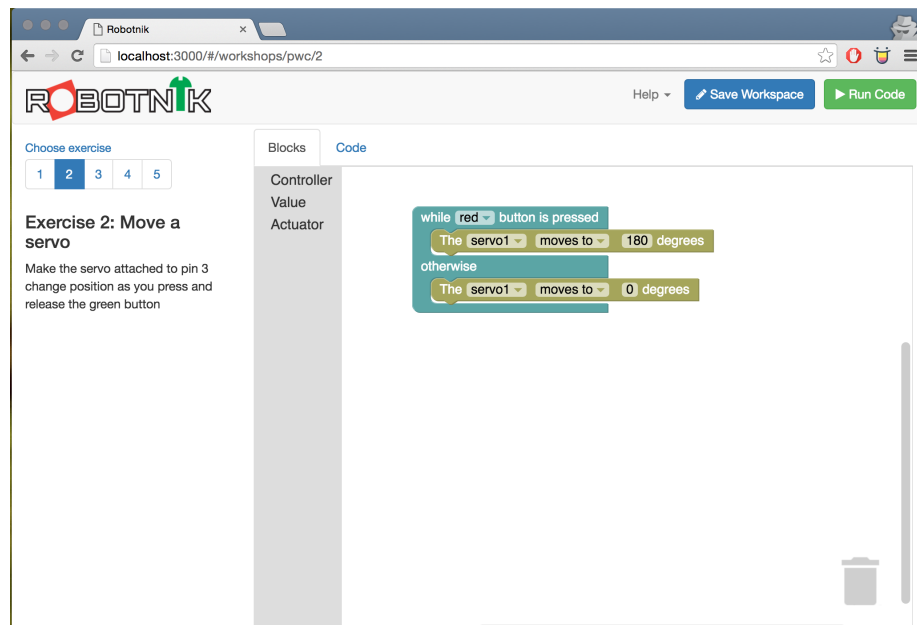


Figure 8: Servo off

```

var five = require("johnny-five");
var readline = require("readline");

var rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

five.Board().on("ready", function() {
  var servo = new five.Servo(3);

  rl.setPrompt("SERVO TEST (0-180)> ");
  rl.prompt();

  rl.on("line", function(line) {
    servo.to(+line.trim());
    rl.prompt();
  }).on("close", function() {
    process.exit(0);
  });
});

```

Make sure you have the Johnny-Five package installed and you can run it from the command line with

```
node code/servo.js
```

Once run, you'll be able to put in a number on the command line and the servo will move to this position.

Going further

Other things you can do:

- Try creating hinges and attachments to the servo arms so you can use a small but rapid movement of the servo and amplify it to make a “flipper”
- Add another servo on pin 4 and see if you can use them together.
- Move onto [Exercise 3 - Spinning a Motor](#)

Exercise 3: Spinning a motor

A normal DC motor is great for turning wheels however they can draw a lot of current. Far too much for an arduino to support. As such a motor controller is used to manage the speed and direction of the motor.

In this exercise, you'll make a motor change directions based on the position of the joystick.

Build the circuit

Requirements

- 1x DC Motor
- 1x Motor Controller
- 1x Battery pack
- 1x Arduino
- Jumper wires

The motor controller has two “channels” which can be used to control 2 different DC motors. Each channel (labelled A and B) then has two pins used to control the speed and direction of the motor. These are labelled “1A” and “1B” so you end up with “A-1A”, “A-1B” to control the A motor and “B-1A”, “B-1B” to control the B motor.

Finally, the motor wires are connected to the screw terminal for the channel you are attaching them to.

Build the circuit below.

Build the code in Robotnik

Select the **Controller** from the toolbox then grab the controller block and drop it onto the workspace. This time we want to select the “Up” event and we want to add another controller where we select the “Down” event.

Next select **Actuator** from the toolbox and add the Motor block onto the workspace and connect it into the bottom of the “Up controller” press event block. Do the same for the “Down controller” event block as well. Set both motor blocks to “Stop”.

This will set the motors to stop when the joystick is returned to the “neutral” or centre position.

Now, add motor blocks to the top part of the condition in each of the “Up” and “Down” controllers that set the direction and speed of the motor. Set the up one

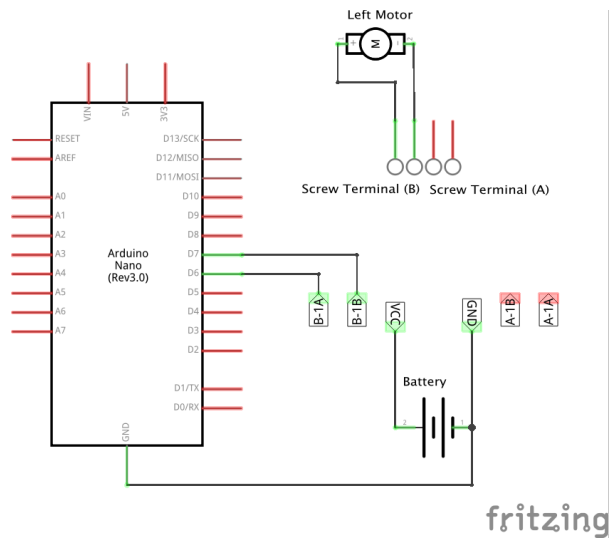


Figure 9: Servo Circuit

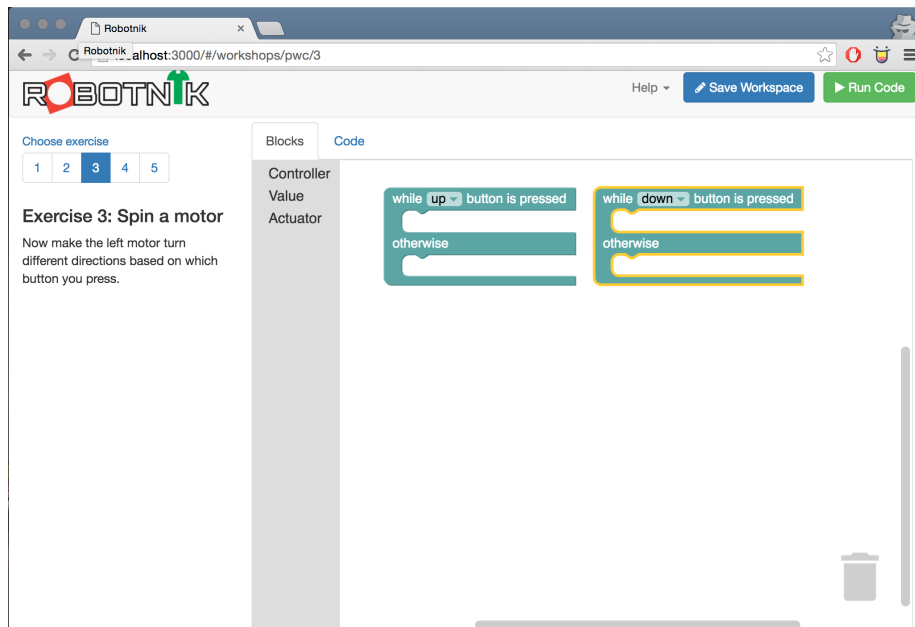


Figure 10: Select controller

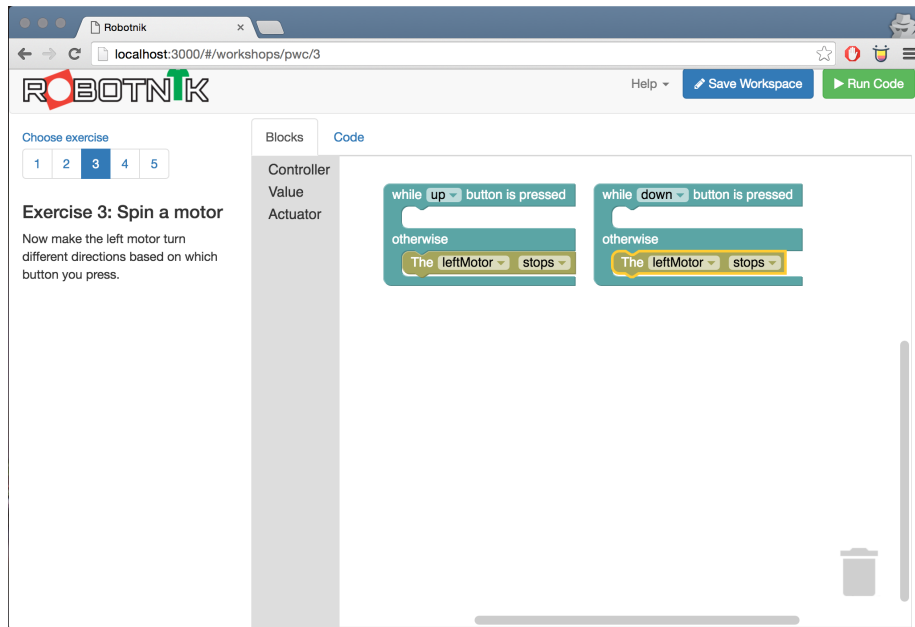


Figure 11: Motor Stop

to turn clockwise at 255 speed and the down one to turn counter-clockwise at 255 speed.

When you run this code you should be able to push the joystick up and down and make the motor turn in different directions.

Build code in JavaScript

To build a similar circuit in javascript to run from a console we can use an script such as the following.

```
// TODO
```

Make sure you have the Johnny-Five package installed and you can run it from the command line with

```
node code/motor.js
```

When you run this code, if you press the up and down arrows you will change the direction of the motor turning.

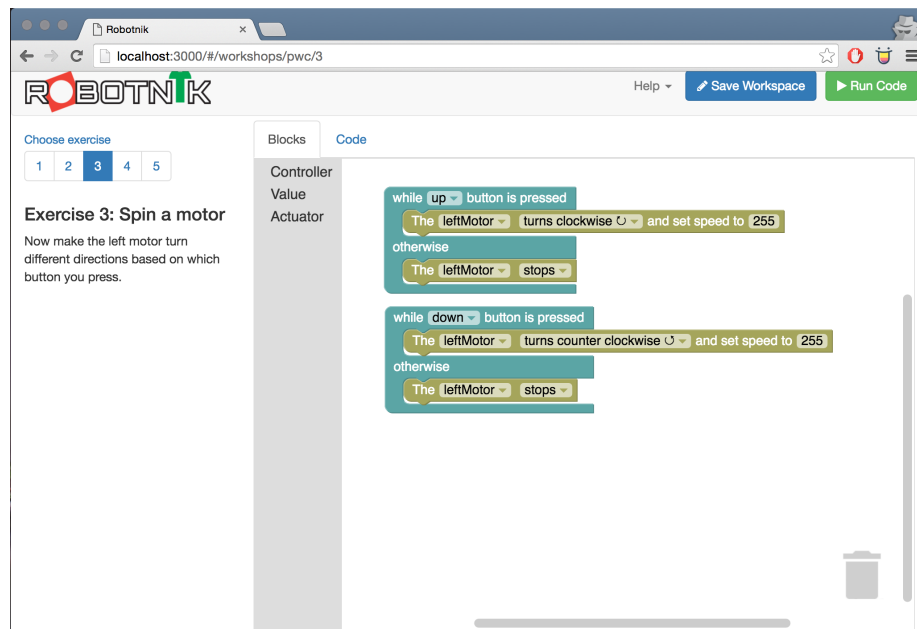


Figure 12: Motor Turn

Going further

Other things you can do:

- Make sure your motor is turning the right direction when you look at the wheel. If it isn't, flip your motor wires around in the screw terminal to get it going as you'd expect it to.
- Move onto [Exercise 4 - Control a wheeled robot](#)

Exercise 4: Control a wheeled robot

Robots can take many different designs for locomotion. The design we'll use is a classic that is well known and has been used in all sorts of robots since the earliest days of robotics. This is known as a 2 Wheeled, Differential design.

How it works is that each of the wheels is connected to an independent motor which can be used to drive its wheel forwards or backwards. This means that by combining the directions of the two motors you can drive forwards, backwards and turn left and right.

The left and right turns are often executed as pivots - eg by turning the left wheel backwards whilst driving the right wheel forwards - however more sophisticated turns can be achieved by slowing the speed of the “inside” wheel whilst keeping the “outside” wheel constant. This is the way you would execute a turn around a corner like you'd drive a car, rather than going to the middle of an intersection and pivoting to the direction of the road you want to turn into.

In this exercise we'll execute a simple pivot which will be enough to illustrate the concepts and you will be able to use the joystick or arrow keys to control the robot.

Build the circuit

Requirements

- 2x DC Motors
- 1x Motor Controller
- 1x Battery pack
- 1x Arduino
- Jumper wires

It is assumed you worked through the [motor exercise](#) so please do that if you haven't already. This circuit is an extension of that one where the right motor uses the “A” channel for control connected to pins 8 and 9.

Build the circuit below.

Build the code in Robotnik

It's assumed you're familiar with the way Robotnik works how so here's some tips on how to build the control.

You need states to detect the up, down, left & right directions. You can also lay them out on the workspace in a way that makes conceptual sense as well.

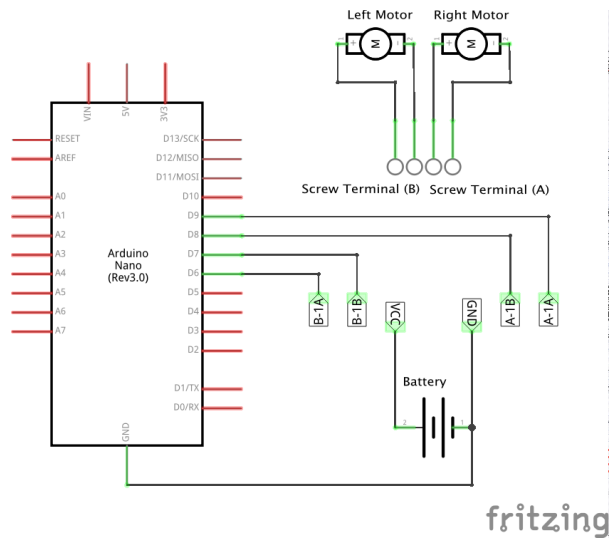


Figure 13: Drive Circuit

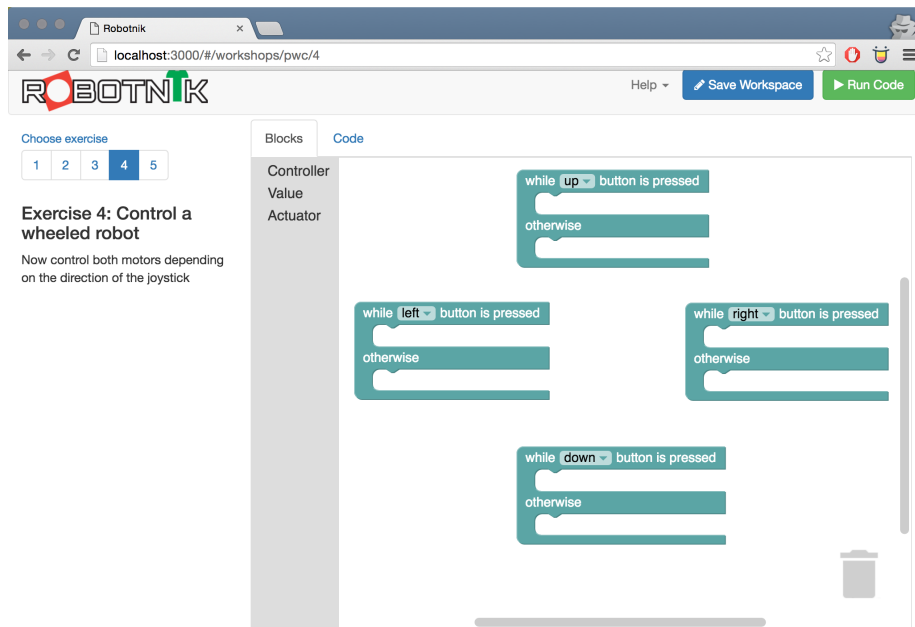


Figure 14: Directions

Before you think about your motors, get the states working. You can write messages out to the console to make sure you're doing the right things at the right time.

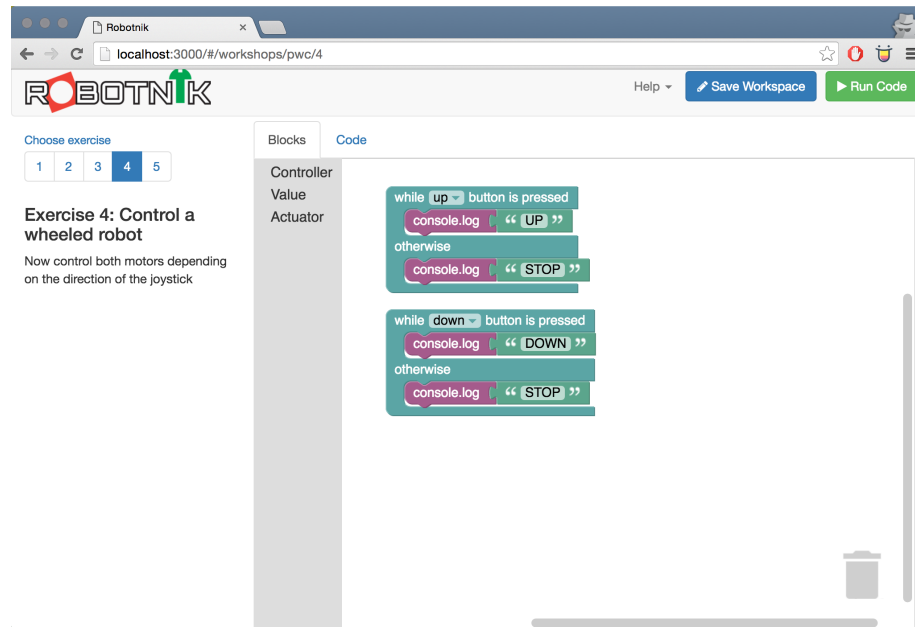


Figure 15: States

If you run this code you'll start to see messages print out to the terminal window when you push the joystick up and down and when you release it. This is a good way to ensure you have your input logic right and you don't even need to wire anything up.

Next you can execute a pivot to the left. For this you turn both motors clockwise. This will make the left side of the robot go backwards and the right side go forwards, thus causing it to pivot towards the left.

Now you can turn left you can build the code to turn right and then to go forwards and backwards using the same logic.

Try running the code after you work on each block so you can assess your work.

It's also easiest to work on things that are similar together eg turning right is just the opposite of going left.

Build code in JavaScript

Rather than repeat the code here, please see the robot.js file in the code examples for a fully worked example.

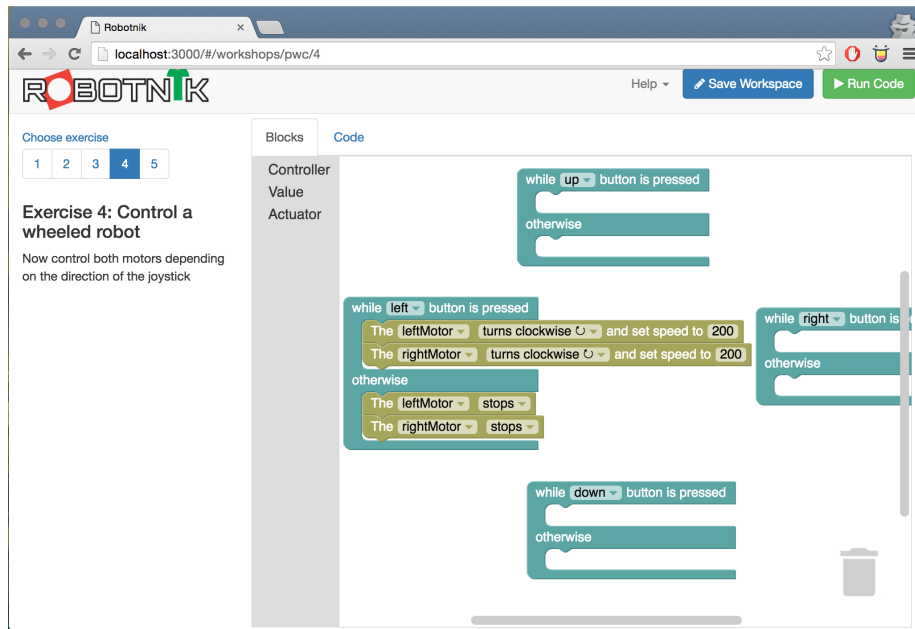


Figure 16: Turn left

Make sure you have the Johnny-Five package installed and you can run it from the command line with

```
node code/robot.js
```

When you run this code, if you use the arrow keys you'll be able to control the direction of the robot.

Going further

Now it's time to take your skills to the ring where you'll battle others.

Think about your design and come up with your own Sumo bot. The aim is to push the other opponent out of the ring or otherwise incapacitate it. You can only make your robot up to 25x25x25cm in size but you can use any component you want inside that volume.

When you're ready and you have your control design sorted, it's time to go wireless and [switch to bluetooth](#) to control your robot.

Pro tip: get your control system fully sorted before you go bluetooth. Going wireless adds another layer of potential bugs so eliminate common problems like your control system first before you introduce complexity.

Exercise 5: Going bluetooth

This exercise assumes you have your robot working and you're simply looking to make it bluetooth enabled. In this case, bluetooth is going to be used to replace the USB cable over what's called Bluetooth Serial.

All the pairing and everything has already been done so all you need to do is wire the bluetooth module up.

Note that it's really important to check your wiring. You can easily wreck your bluetooth module if you wire it back to front

Build the circuit

Requirements

- 1x Arduino
- 1x Bluetooth module
- Jumper wires

Disconnect the USB cable and put it to one side then add the circuit below to your existing design.

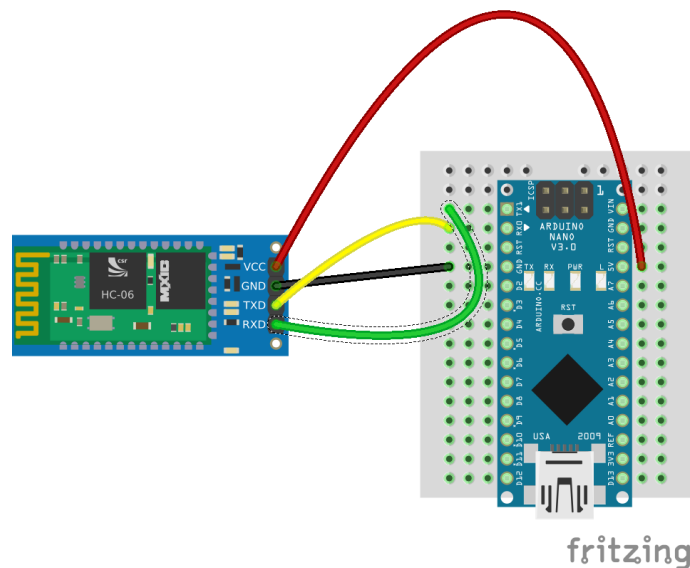
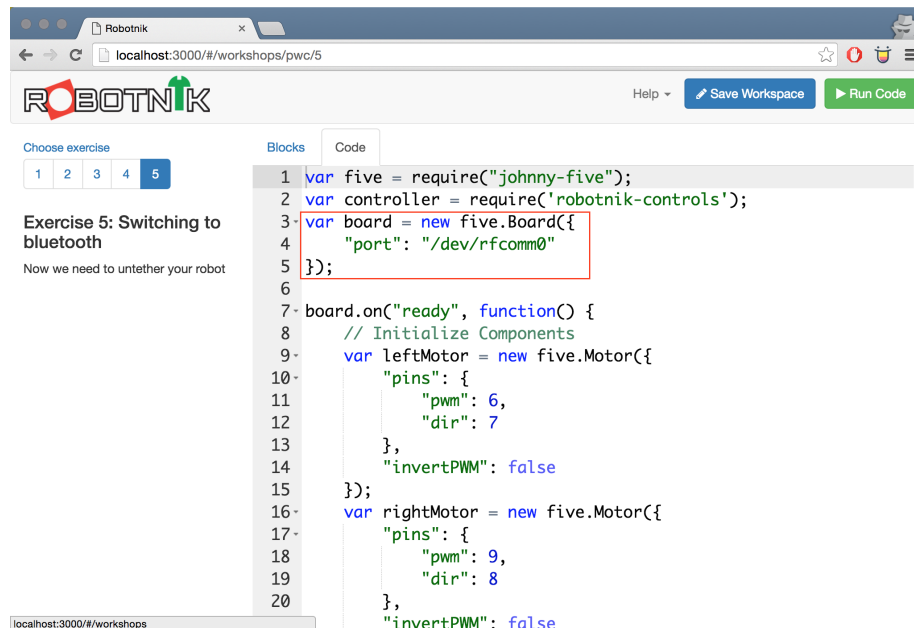


Figure 17: Bluetooth Circuit

Build the code in Robotnik

Using this exercise, it will set the port in the code to use the bluetooth module instead of USB which is the default. We have to tell it which module to use however which is why we need a new exercise.

You can see the code that changes in the image below.



```
1 var five = require("johnny-five");
2 var controller = require('robotnik-controls');
3 var board = new five.Board({
4   "port": "/dev/rfcomm0"
5 });
6
7 board.on("ready", function() {
8   // Initialize Components
9   var leftMotor = new five.Motor({
10     "pins": {
11       "pwm": 6,
12       "dir": 7
13     },
14     "invertPWM": false
15   });
16   var rightMotor = new five.Motor({
17     "pins": {
18       "pwm": 9,
19       "dir": 8
20     },
21     "invertPWM": false
```

Figure 18: Different port

Going further

One of the big things you need to consider is power. If you're having issues then you might need a second battery pack to deal with the current draw.

Consider powering the Bluetooth and arduino from one pack and your motors and servos from another (probably the AA batteries).