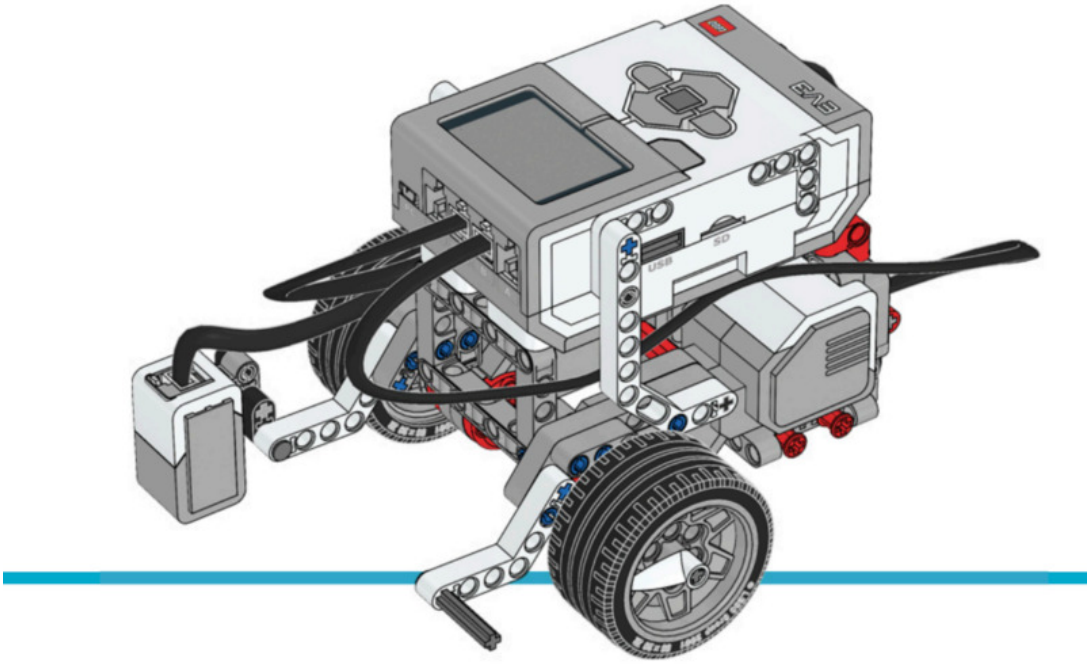
RoboRacers:

Can You Stay On The Track?

Algorithm Development

Welcome to RoboRacers! Today, we're going to use the LEGO MINDSTORMS EV3 robot and try to race it around the racetrack! Before we can start racing, we will need to think about how the robot will see the racetrack. But first we need to master moving our robot around. So let's get going...



Look out for these boxes:

Information boxes give useful information about the robot.

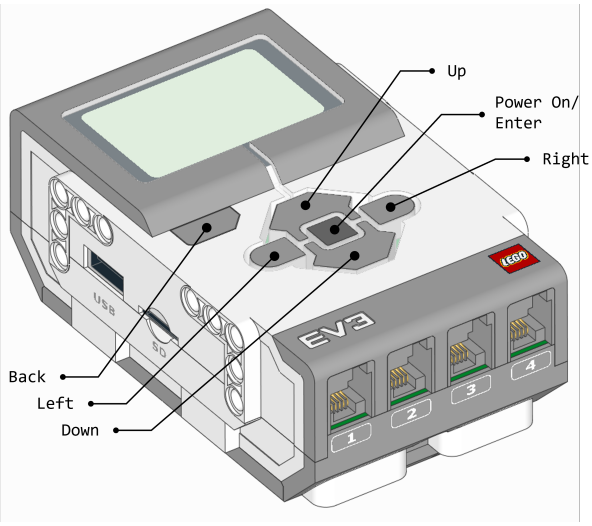
Hint boxes give tips to to help you complete the tasks.

Think boxes give you something to think about and maybe an extra challenge.

Getting Started

The Robot

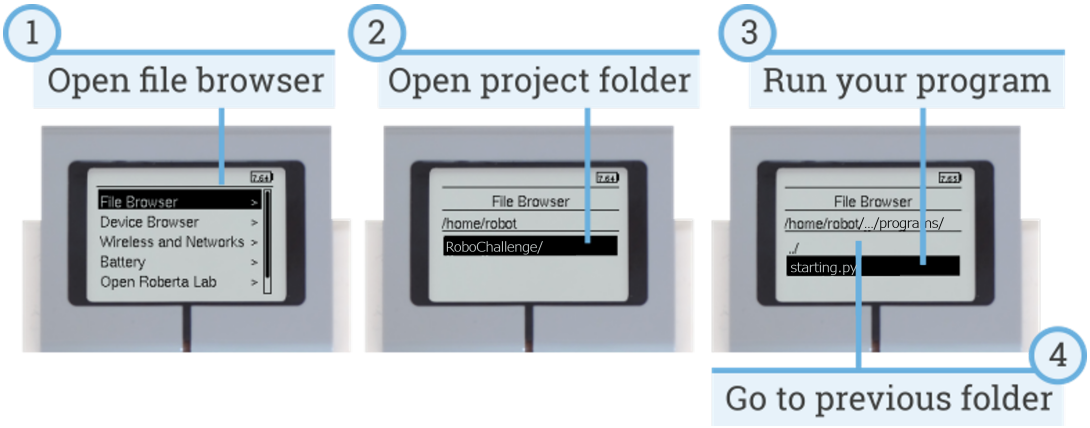
Let's turn the robot on. If the light behind the middle button is green then you're good to go. If there are no lights, press the middle button and wait for it to turn on. This could take a minute, so while you're waiting read on.



You will see the robot has two wheels (that it uses to move) and a range of sensors (we'll get to them later) all held together by lego. At the centre is a big box that we call the `brick`; it tells every other part of the robot what to do. At this stage, robots are just a pile of lego and electronics - not very intelligent - until we tell them what to do. This requires a `program`.

Running a program

Let's run a program.



Use the buttons on the robot to navigate `File browser > RoboChallenge > programs > starting.py`. Clicking `starting.py` will "run" that program. You should hear the robot say "Hello World", and turn a wheel!

Press the `back` button on the robot to stop the program.

Creating a program

Next we want to be able to write our own programs to get the robot to do what we want. You will write the code on the laptop using Visual Studio Code which many students and many professionals use to write code.

1. On the laptop, open the file `starting.py` by clicking on it in the `Explorer` left panel. This was the program we just ran on the robot.
2. Change the program to say "Hello [your names]".
3. Change the program to get the other wheel to turn.

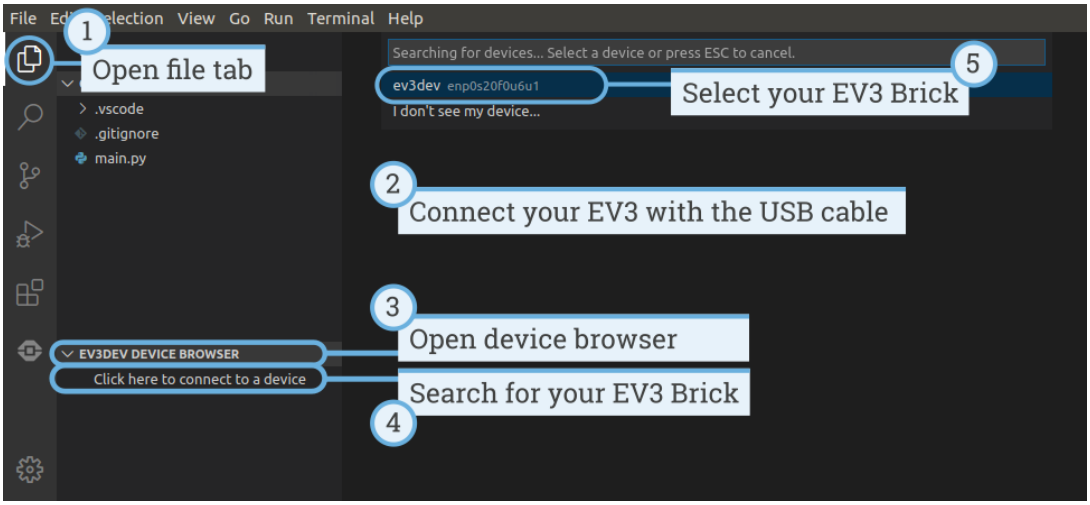
Any line starting with `#` is a comment which the robot ignores. e.g. `# This is a comment`. Every other line tells the robot to do something.

You will need to change the lines `ev3.speaker.say("Hello World!")` and `left_motor.run_time(360,1000)`.

Great, you've written your first program!

Downloading a program to the robot

Now we need to download it to the robot and run it.



1. Open the `Explorer` in the left panel of VSCode (see image).
2. Connect EV3 to PC with a cable. (EV3 must be on and the light lit green)
3. Expand the `EV3DEV DEVICE BROWSER` from the bottom of the panel.
4. Click `Click here to connect to a device`.
5. Select your device (from the top of the screen).
6. Press `F5` (maybe `Fn + F5`) to `download and run current file`. Alternatively open the `Run and Debug` side panel and click the play button.

The robot will start running the program whilst still plugged in. You can uplug it once it has finished downloading and you can re-run the program unplugged as per the earlier instructions to run a program.

Getting the robot to move

Open the file `moving.py`. You will see we have added the line `robot = DriveBase(...)`. It would be inconvenient if we had to control the robot through the motor speeds individually so instead we use `DriveBase` which provides some useful functions to control the robot:

- To drive the robot in a straight line for the given `distance` in mm, use:

```
robot.straight(distance)
```

- To turn the robot on the spot the given `angle` in degrees (°), use:

```
robot.turn(angle)
```

- To combine driving forwards and turning, use this function. The robot will drive forwards at the speed `drive_speed` in mm/s and turn at the speed `turn_rate` in deg/s. This function starts the robot moving and it will continue until another command is given.

```
robot.drive(drive_speed, turn_rate)
```

Modify `moving.py` by to attempt the following movements. You need to select the appropriate function from above (`straight`, `turn` or `drive`) and replace the value (`distance`, `angle`, `drive_speed` and `turn_rate`) with the right number.

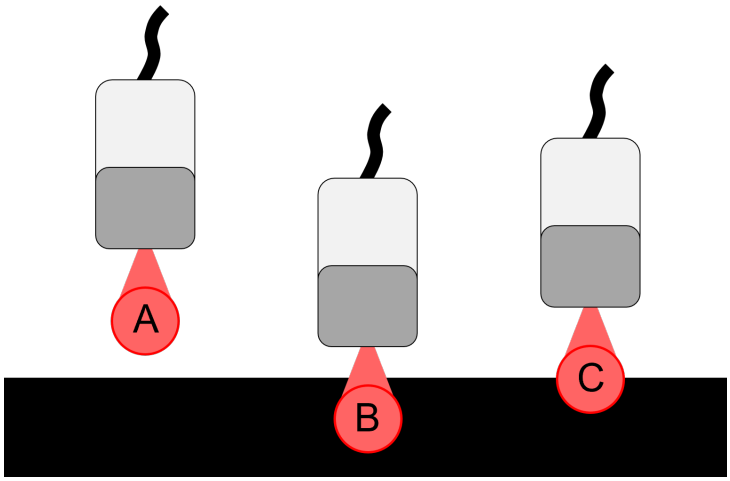
1. Drive forwards 500 mm
2. Drive backwards 500 mm
3. Turn 180° on the spot both ways
4. Drive in a square
5. Drive in a circle

Hint: values can be negative.

Challenge: get the robot to move by controlling the motors without `DriveBase`.

Sensing

Did the robot finish exactly where it started when you drove in a square? The robot has no way to "see" where it is. Just like you trying to walk in a big square whilst blindfolded. To let our robots "see", we use *sensors* on them. In this part, we'll use the reflectance sensor. It shines some light onto the floor beneath it and measures how much is reflected. Let's look at the diagram below with a black line and a white background...



In case **A**, the sensor is shining onto white and the reflectance value will be high (close to 100). However, in case **B**, the sensor is shining onto black and the reflectance value will be low (close to 0).

Think about what the sensor value might be in case **C** where the sensor is half on the black line and half on the white background.

1. Open the script `threshold-test.py`. You will see in the program we have added the sensor with the line

   ```
   line_sensor = ColorSensor(Port.S3)
   ```

   and that we measure the reflectance and show it on the robot's screen with

   ```
   ev3.screen.print(line_sensor.reflection())
   ```

2. Run the program and look at the reflectance values shown on the robot's screen.
3. Move the robot (with your hand) so the sensor is directly above the line and make a note of the reflectance value. Repeat to get a reflectance of the background surface.
4. Now open `sensing.py` and input the values you measured for `LINE_REF` and `BACKGROUND_REF`.

Now the robot can *sense* what is happening in the world, it needs to know how to react to what it *senses*. We put together a set of rules which makes a process we call an *algorithm*.

5. Design an algorithm so that your robot will drive forwards and stop when it reaches the line. Write your algorithm in the program `sensing.py` and test it.

Racing

Now it's time to get our robot to race the track by following the line!

1. Open `racing.py` and input the same values for `LINE_REF` and `BACKGROUND_REF`.
2. Design an algorithm and write the rest of the program so your robot will be the fastest around the racetrack!

Congratulations for getting your robot to race around the racetrack! Was your robot able to stay on the track? We hope you have learnt one way that robots can see the world and that you consider a future in robotics!