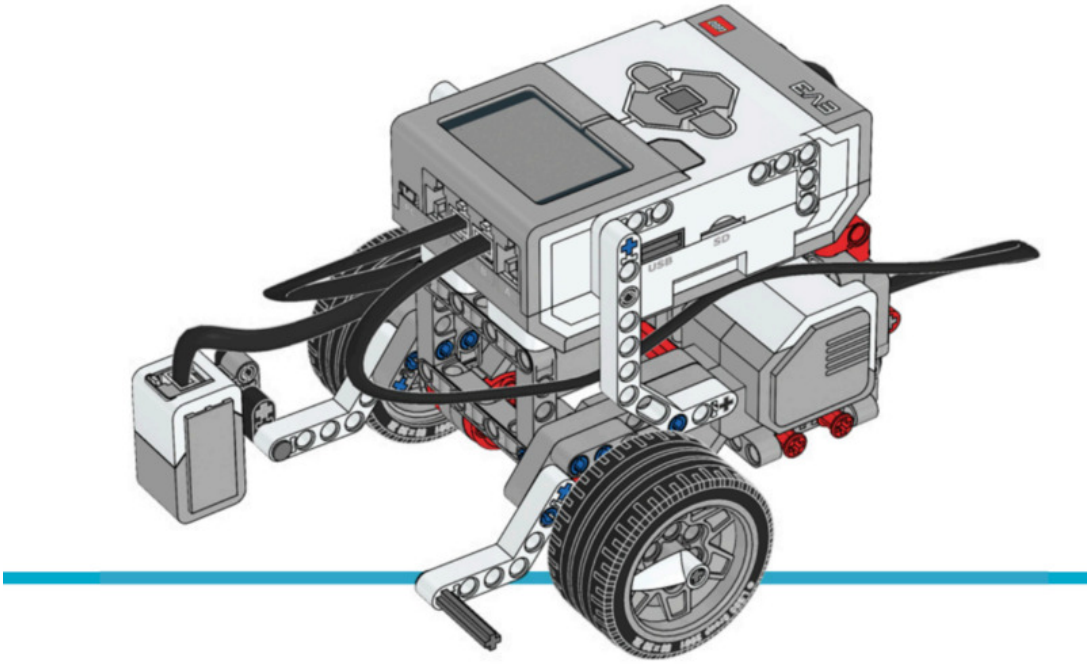
RoboRacers:

Will you be the fastest?

Welcome to RoboRacers! Today you will use a LEGO MINDSTORMS EV3 robot to speed along the racetrack! Let's get going...



Look out for these boxes:

Warning boxes have **very important** information.
Read these before moving on.

Hint boxes give tips to help you complete the tasks.
Read these if you get stuck.

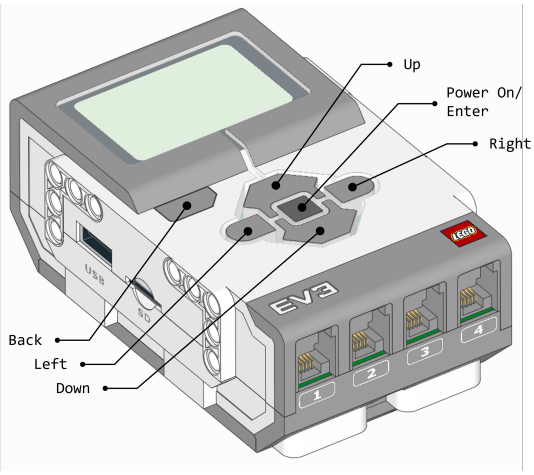
Think boxes give you something to think about and maybe an extra challenge.

This worksheet is for the RoboChallenge session *<session>* *<version>*.
Corresponding programs are available in the repository at `github.com/bennett-j/RoboChallenge`.
© 2025 RoboChallenge Contributors (list in the repository). Licensed under CC BY-SA 4.0.

Getting Started

The Robot

If the robot has a green light on top then it is powered on and waiting for you! If you do not see a green light, ask your session leader for help. The `brick` in the picture below is the brains of the robot - it tells every other part of the robot what to do.

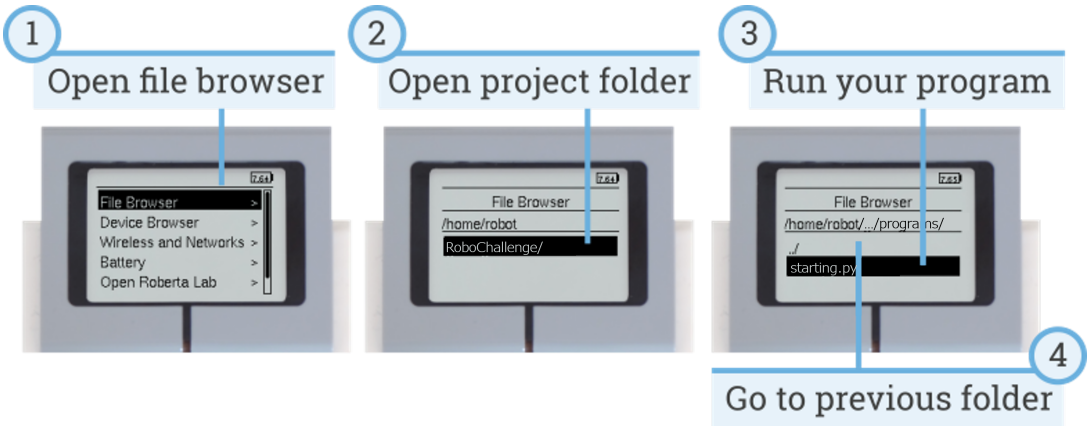


The `up`, `down`, `left`, and `right` buttons are not used to drive the robot. You will use them to select the robot's `programs` from the screen. Let's run a program to test the robot!

Running a program

Look at the small screen on the robot. It should look like the pictures below.

1. Use `up`, `down` and `enter` on the robot to open `File Browser`, then `RoboChallenge`.
2. Find the `starting.py` file and make sure it is highlighted in black.
3. Run the program by pressing the middle `enter` button. Listen closely!



What did the robot say? It should have said "Hello World!". This is a simple program to get you started. If the robot didn't speak or make any noise, ask your session leader for help.

To stop running a program, press the `back` button at the bottom left of the robot screen.

© 2025 RoboChallenge Contributors CC BY-SA 4.0

Editing a program

Programs don't write themselves. It's great that the robot said "Hello World", but that's not your name! So next, we'll change the program to greet you personally. You will write your code on the laptop using Visual Studio Code. This IDE (Integrated Development Environment) is used by many students and professionals to write code.

Look at the laptop in front of you. It should be logged in, with the file `starting.py` open. If you don't think this is the case, then ask your session leader for help. Follow the steps below to change `starting.py` to make the robot greet you.

Any line starting with `#` is a comment which the robot ignores. For example, `# This is a comment`. All other lines tell the robot to do something.

1. Find the line in the program which makes the robot say "Hello World!".
2. Change the program to say"Hello [your names]!" (for example, "Hello Jeremy!").

You will need to change the line which says
`ev3.speaker.say("Hello World!")`

Great, you've written your first program! Now, we need to send the program to the robot so it can run it.

Downloading a program to the robot

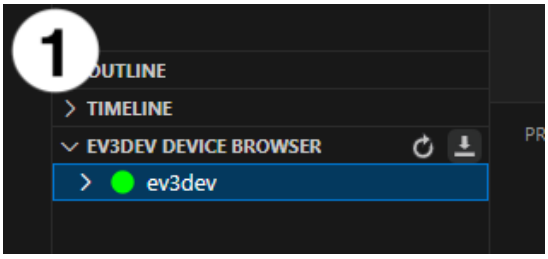
The first step is to make sure the robot is plugged in to the laptop via the USB cable. If you aren't sure where to plug the robot in, or if you don't have a cable, ask your session leader for help.

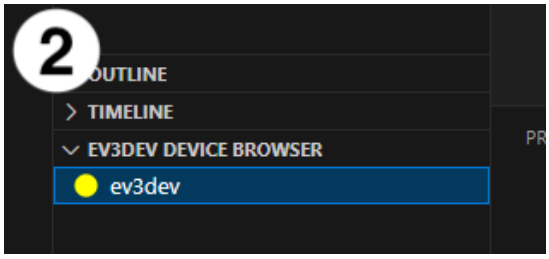
On your screen, you should see the `EV3DEV DEVICE BROWSER` at the bottom of the left panel. You may have to expand it by clicking on the small arrow next to it. If you don't see this, ask your session leader for help.

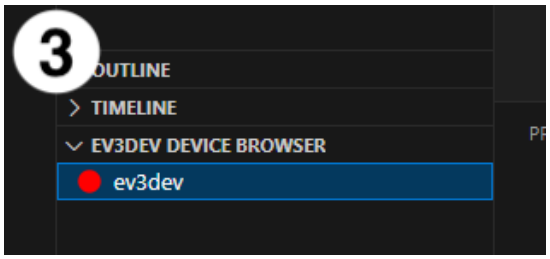
The next steps are very important! Make sure you follow them carefully. If you get stuck, ask your session leader for help.

© 2025 RoboChallenge Contributors CC BY-SA 4.0

Look for the small circle which is either green, yellow, or red. If it is green, like in picture 1 below, the robot is connected and ready to receive code. If it is yellow like in picture 2, the robot is currently in the process of connecting. If it is red, like in picture 3, the robot is not connected. If you are not connected, right click the robot's name, and then click `Reconnect`.

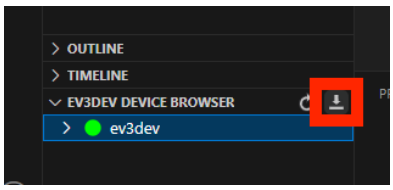




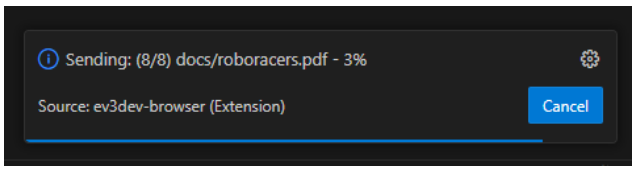


The circle must be green like in picture 1 before you can download your program to the robot. If you can't make it turn green, ask your session leader for help.

1. Hover the cursor to the right of where it says `EV3DEV DEVICE BROWSER` and click on the right hand button which looks like an arrow pointing downwards.



2. Watch in the bottom right hand side of the laptop screen whilst the files and programs are downloaded to the robot. You will see a blue loading bar fill from left to right, and a message saying `Download Complete` when it has finished.



3. Now, unplug the robot, and run the program again by following the steps in the `Running a Program` section above. Listen carefully to make sure the robot says your name!

Now that you have unplugged the robot, the little circle will be red when you plug it back in. This is normal, it just means the robot is not connected to the laptop. You will have to reconnect the robot to the laptop by right clicking on the robot's name and selecting `Reconnect`.

© 2025 RoboChallenge Contributors CC BY-SA 4.0

Getting the robot to move

Now we want to write programs to get the robot to move in different ways. On the laptop, open the file `moving.py` from the left panel. Your first tasks are to edit `moving.py` to make the robot complete the following movements. Read below for the commands you need to make the robot move.

1. Drive forwards 500 mm
2. Drive backwards 500 mm
3. Turn 180° clockwise
4. Turn 90° anti-clockwise
5. Drive in a square

To drive the robot in a straight line, we use:

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

We must replace the *word* `distance` with a *number*. Here, it is the distance in millimeters that we want to move.

To turn the robot on the spot, we use:

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

Here, you replace `angle` with the number of degrees you want the robot to turn.

Hint: Values can be negative.
Bonus hint: You can list commands one after another.

Challenge: Can you write a program to make the robot drive along the racetrack?

© 2025 RoboChallenge Contributors CC BY-SA 4.0

Using the Reflectance Sensor

You may have noticed that it was difficult and time consuming to complete the race track using manual commands. Robots use sensors to help make their own decisions by seeing the world around them.

Imagine trying to walk in a square with your eyes closed. That would be difficult! It would be much easier if you could see. By using the sensor, we are allowing the robot to "see".

The robots you are using have **reflectance sensors**. You should see it sticking out of the front of the robot, shining a red light onto the table. If you can't see this, ask your session leader for help.

The sensor shines a light onto the floor underneath it, and measures how much is reflected back. If the floor is a light colour (for example, white) it will have a **high** reflectance value but if it's dark (for example black) the reflectance will be **low**.

Making sure the sensor is working

1. Put the robot on the piece of paper on your desk, with the wheels on the start line.
2. On the robot, run the `sensing.py` program. It should stop when it reaches the black finish line.

Your task now is to make the robot stop at the grey line in the middle of the paper. Let's measure the reflectance of the grey line.

3. On the robot, run `threshold_measurer.py`. You will see lots of numbers on the robot's screen - these are the *reflectance values* from the sensor.
4. Move the robot around (with your hand) and see how the values change when the sensor is above different coloured surfaces.
5. To stop the program, press the back button (top left button) on the robot.
6. On the laptop, open `sensing.py`.

Before moving on: examine `sensing.py` and figure out what made the robot stop on the black line? How can you make the robot stop at the grey line?

7. Find the lines in the program that define `LINE_REF` and `stop_reflectance`, then read the comments to see what they are used for.
8. Change the program's reflectance value for the line to make the robot stop at the grey line.
9. Download your new program to the robot and run it to test whether your robot stops on the grey line.

Did you manage to make the robot stop on the grey line? If you did, congratulations! If not, try again. Why not ask your class mates for help?

© 2025 RoboChallenge Contributors CC BY-SA 4.0

The Final Round

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

1. Open `racing.py` on the laptop. Have a quick look at it and see if you can figure out how it works.
2. Place your robot on the racetrack with the sensor hovering over the white centre line in the road.
3. Run `racing.py` on the robot and watch what happens!
4. Read the comments that explain how the program works - specifically `FORWARD_SPEED` and `TURN_GAIN`.
5. Adjust `FORWARD_SPEED` and `TURN_GAIN` so that the robot can complete the racetrack in the quickest time! Get your session leader to time your attempts along the race track.

DON'T BREAK THE RULES!

1. Robot must start with wheels behind the start line.
2. Robot must not leave the road or crash into trees.
3. Robot's wheels must cross the finish line for a valid track completion.

The robot follows the edge of a line (not the line itself). The `reflectance_setpoint` is calculated to be grey, the average of the reflectance values measured for the line and road. Suppose the white line is on the left and the dark road background on the right. When the sensor sees more dark road than the setpoint, the robot needs to turn to the left to find the line and the opposite for when it sees too much white line. This is why the robot can only follow one side of the line, rather than the middle.

Congratulations for getting your robot to race along 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!

© 2025 RoboChallenge Contributors CC BY-SA 4.0