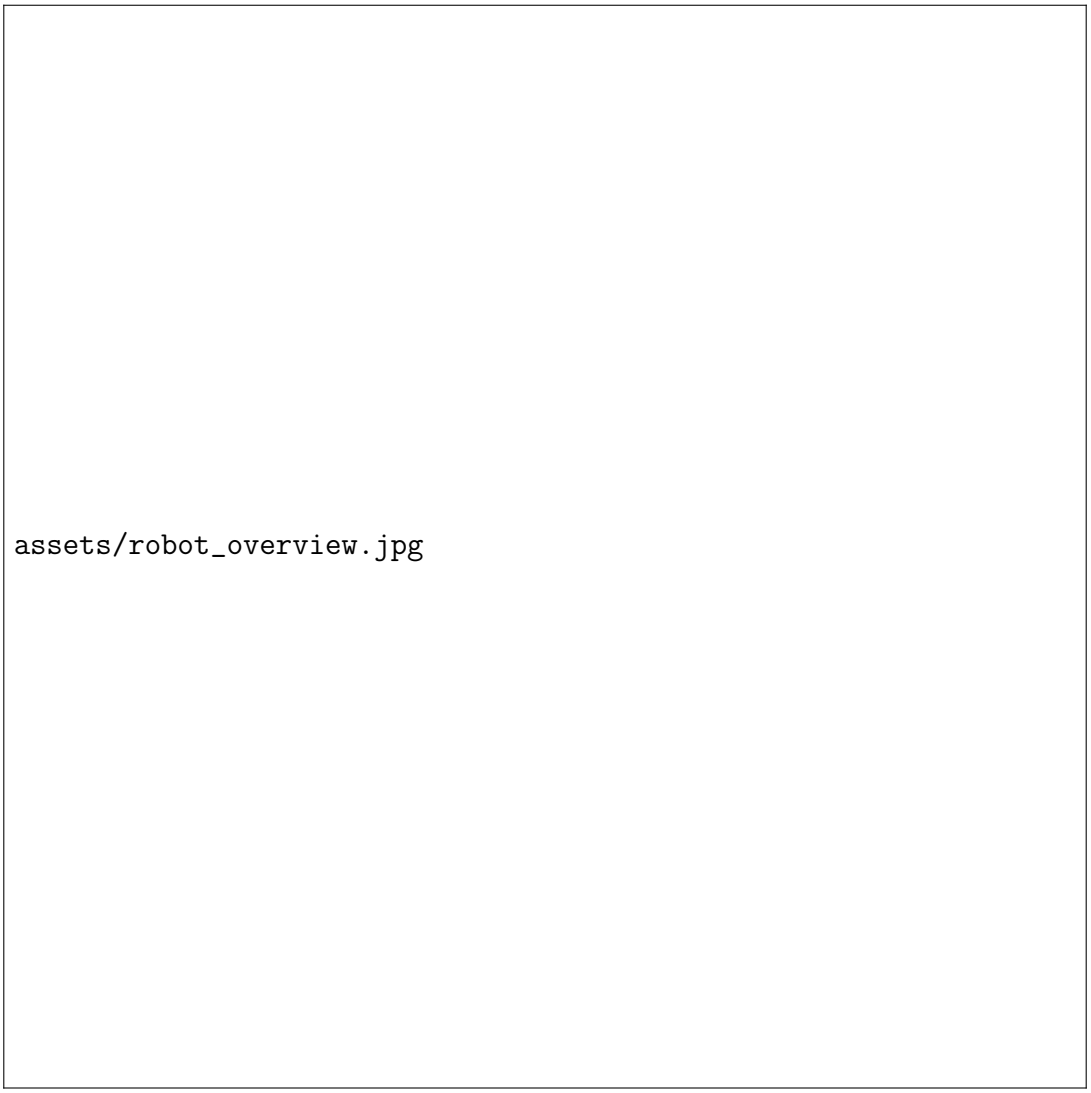
RoboRacers:

Will you reach the top of the leaderboard? UPDATED

Welcome to RoboRacers! Today you will use a LEGO MINDSTORMS EV3 robot to try and race down a racetrack! Let's get going...



Look out for these boxes:

Warning boxes include **very important** information. Read these before moving on to the next section.

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, maybe an extra challenge...

Getting Started

The Robot

The robot is already turned on for you. You should see a green light on the robot. If you do not see a green light, ask your session leader for help. Look at the diagram below to see what the buttons on the robot do.

The `up`, `down`, `left`, and `right` buttons are not used to control the robot. They are used to navigate the `programs` which are stored on the robot. Use the middle button to "click" on whatever is highlighted in black on the robot's screen. We are going to write some programs to control the robot!

You will see the robot has two wheels, and a sensor at the front. There should be a small red light on the sensor - check and see if it's there. If you can't see the red light, ask your session leader for help. Our programs will use the robot's wheels and sensor to move around.

The robot has a speaker which can say things, and a screen which displays information. It also has two motors which turn its wheels, and a sensor to see the world through measuring light reflectance.

Running a program

Look at the small screen on the robot. It should look like picture 1 below. Use the buttons on the robot to first open the `File Browser`, then open the `RoboChallenge` folder, then find the `starting`.py file. This file contains a program, so the next step is to run it!

Make sure the file `starting.py` is highlighted in black, then use the `middle button` on the robot (the enter button) to run it. Listen closely! The robot should speak to you...

assets/run_program_manual.png

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

If you want the robot to stop running a program, press the `back` button at the bottom left of the robot screen to stop the program.

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 `starting.py` program to greet you specifically. You will write your code on the laptop using Visual Studio Code. This IDE (integrated development environment) is used by many students and professionals use 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.

Your Tasks

1. Look for 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. Go to the next page.

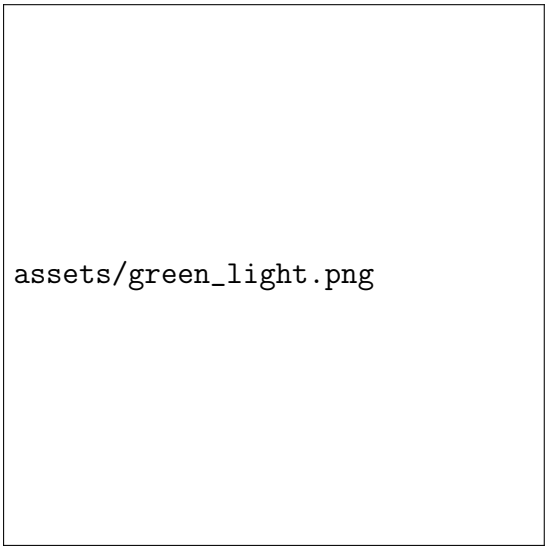
Downloading a program to the robot

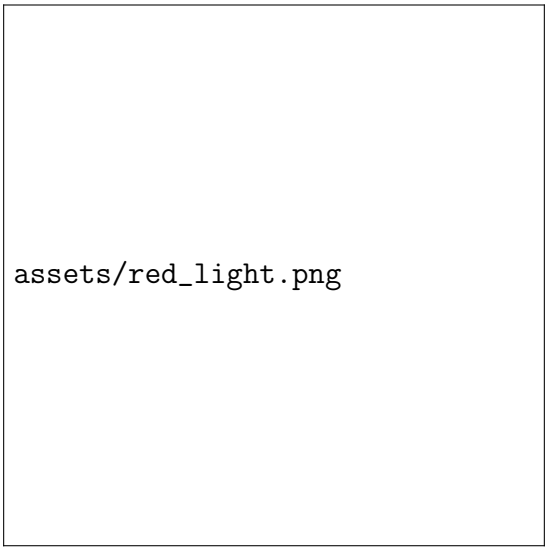
Now we need to download your new, edited program to the robot, and run it. The first step is to make sure the robot is connected to the laptop. You should have a USB cable connecting the robot to the laptop. If you aren't sure where to plug the robot in, or if you don't have a cable, ask your session 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.

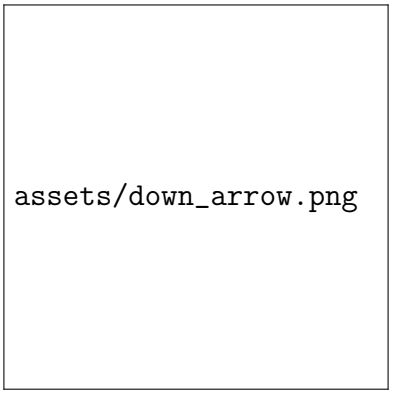
Look for the small circle which is either green, yellow, or red. If it is green, like in picture 1 below, the robot is connected and ready to receive code. If it is yellow like in picture 2, the robot is currently in the process of connecting. If it is red, like in picture 3, the robot is not connected. 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. If it is green, move on to the steps below.

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`.

Challenge: Can you make the robot say something else? Try changing the text in the `ev3.speaker.say()` function.

Getting the robot to move

On the laptop, open the file `moving.py` from the left panel. The goal of this section is to get the robot to move in a few different ways, using a couple of different functions. Your first tasks are below, as well as the functions you will need to complete them...

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 don't actually include the word 'distance'. Instead, you need to replace it with a number to represent the `distance` you want the robot to move. This `distance` is in mm.

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

Look at the bottom of `moving.py` for an example.

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.

Trial run along the racetrack

Using the commands `robot.straight(distance)` and `robot.turn(angle)`, see how far along the racetrack you can get in 5 minutes. If you get to the end, well done! Tell the session leader.

Here are the rules for the tournament!

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.

Using the Reflectance Sensor - Important!

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, rather than us having to tell them every little thing down to the mm.

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. On the robot, find the file `sensing.py`.
2. Put the robot at the start line on the piece of paper on your desk, with the wheels on the line.
3. Run the `sensing.py` file on the robot. 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.

Measuring light reflectance

1. 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.
2. Move the robot around (with your hand) and see how the values change when the sensor is above different coloured surfaces.
3. To stop the program, press the back button (top left button) on the robot.

Before moving on - open `sensing.py` on the laptop and consider: how can you use the numbers you saw the robot's screen to change the code and make the robot stop at the grey line?

Stopping on the grey line

1. If you haven't already, on the laptop, open `sensing.py`.
2. Find the lines in the program which define variables `measured_reflectance` and `stopping_threshold`. Read the `# NOTES FOR STUDENTS #` which discuss the variables and their purpose.
3. Change the program's measured reflectance to make the robot stop at the grey line.
4. Test your new program by sending it to the robot and seeing if your robot stops on the grey line - or if it still keeps going all the way to the black finish line.

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

> Teamwork is the dreamwork!

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 black 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!
Remember the rules of the tournament...

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 `adjusting_threshold` is calculated to be grey, the average of the reflectance values measured for white and black. Suppose the black line is on the left and the white background on the right. When the sensor sees more white than the threshold, the robot neeeds to turn to the left to find the line and the opposite for when it sees too much black. This is why the robot can only follow one side of the line, rather than the middle.

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!