makeblock education

Lesson 5: Sightseeing

Subject: STEAM
Grade(s): 5th and up
Duration: 45 minutes
Difficulty: Beginner

Lesson objectives

By the end of this lesson, students will know:

- How color sensors work
- Their applications in everyday life and in robotics

Overview

Imagine being in a foreign city, taking a tour in a coach along the most beautiful sight. Perhaps you've made such a trip yourself, and if not, you probably know the pictures from TV and the Internet. In this lesson the mBot2 is transformed into such a touring car.

Because it would be very boring for a driver to drive the same route every day (and explain every sight over and over again), we are going to program this coach. This way people can enjoy a nice ride without the driver getting bored on repetitive tasks.

Focus

At the end of this lesson, students will be able to:

- Make the mBot2 follow a line
- Have the mBot2 perform actions based on colors

Pre-lesson Checklist

What do you need?

- PC or laptop (with USB output) with the mBlock software installed, the web version (also for Chromebook), or a tablet with the mBlock app installed
- The mBot2 with a CyberPi
- A USB-C cable or Makeblock Bluetooth dongle

Lesson plan

This lesson consists of four steps and takes a total of 45 minutes to complete.

Duration	Contents
5 minutes	1. Warming up • Color sensors in everyday life • How do color sensors work? What is specific about the mBot2's color sensor?
10 minutes	2. Hands-on • Becoming familiar with the different code blocks of the Quad RGB sensor • Recreating and testing programming examples of the Quad RGB sensor • How can the mBot2 follow a line?
25 minutes	3. Trying out • Writing your own program for the robot
5 minutes	4. Wrap-up • Showtime: show what you did with your robot in a fun, short movie for later discussion • If your teacher allows, share the end result on social media with the hashtag #mBot2 • Reflection: What are you most proud of? What would you like to improve about your robot?

makeblock education

Activities

1. Warming up (5 min)

Step 1: Warming up

This step consists of two parts:

1. Color sensors in everyday life
2. How do color sensors work? What is specific about the mBot2's color sensor?

1. Color sensors in everyday life

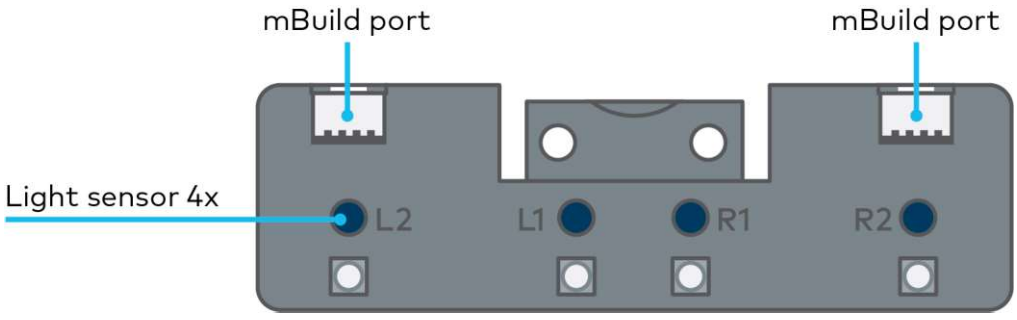
Color sensors can detect light intensity and colors. You will encounter this type of sensor in many different places in everyday life.

For example, a warehouse robot can find the correct route through a warehouse based on different colors on the floor. This kind of sensor is also used if precise color-matching is required, e.g., in vehicle coloring. A measuring device with a color sensor, the color meter, can check if the right color is used in spray-painting a vehicle. Can you and the classmates from your group think of any other examples?

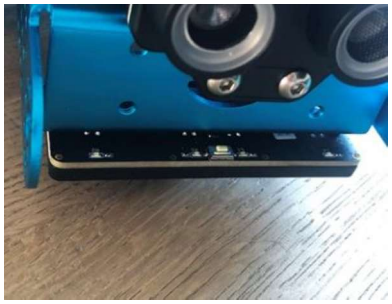
2. How do color sensors work? What is specific about the mBot2's color sensor?

A color sensor consists of three different (internal) sensors that measure the light reflected by an object based on the intensity of the Red, Green and Blue color values. Color sensors are therefore called "RGB-sensors" as well. Any color is decoded into three values by the sensor – so a specific "mix" of these values represents a color.

The mBot2 has four of these RGB-sensors integrated into a single sensor. Take a look at the image below.



This is the Quad RGB sensor that is on the mBot2. The RGB-sensors are named L1, L2, R1, and R2 (L for left side, R for right side). They automatically detect the RGB values of the reflected color and compare the mix of values internally with different pre-sets for colors. This will make coding much easier, because the sensor can then report the color and you don't have to check the RGB-codes yourself. The sensor can detect 6 different colors plus black and white. Instead of color it can report on shades of grey or lines and junctions.
The Quad-RGB-sensors allows the mBot2 to follow a line on the floor and react to different color markers. The Quad RGB sensor is placed on the front of the mBot2. At the bottom face you will see the four sensors. At the topyou can find the names of each one, so you can identify and program the sensors in a specific way. Take a look at the images below.



Quad RGB sensor, front view

Top view

Bottom view

The little button on the top side of the sensor is used for calibration. If you double-click it, the LEDs will start flashing and you can "swipe" the mBot2 with the sensor across the line to be followed. This will calibrate the sensor to differentiate the line from the background. If you are using a black line on a white surface, this is normally not needed.
The map provided with the mBot2 has colors marked inside the track itself to react to the color codes. This might cause the sensor to misunderstand a bright color (like yellow) for the background instead of the track. So as a first exercise, please calibrate the sensor to the yellow color code, it will then automatically recognize reliably this and all "darker" colors as the line.
When you program the mBot2, you will use one or more of these four sensors to check for certain colors on the floor along the way. You can have the mBot2 perform a command when a color is spotted. For example, 'stop' when red and "go left" when green.

makeblock education

2. Hands-on (10 min)

Step 2: Hands-on

This step consists of three parts:

1. Getting acquainted with the different code blocks of the Quad RGB sensor
2. Reproducing and testing some programming examples of the Quad RGB sensor
3. How can the mBot2 follow a line?

1. Getting acquainted with the different code blocks of the Quad RGB sensor

In mBlock, there are several code blocks that you can use to program the Quad RGB sensor. You can find these code blocks in the 'Quad RGB' category of the block field in mBlock. These code blocks are green.

In the table below, you can see some of the code blocks you can use in your program with the Quad RGB sensor. The first four code blocks are for (simple) line-identification. They can be used for simple and advanced line following, but also for detecting crossings and sharp (90°) turns. The calibration of the sensor to different color tracks is briefly mentioned in the section above. The blue LEDs on the top side of the sensor indicate the line/background detection: if they are off, they sense dark color (line), if they are shining blue, they indicate bright (background) color. The status of the LED therefore indicates the reflectivity of the floor or map.

Code blocks:

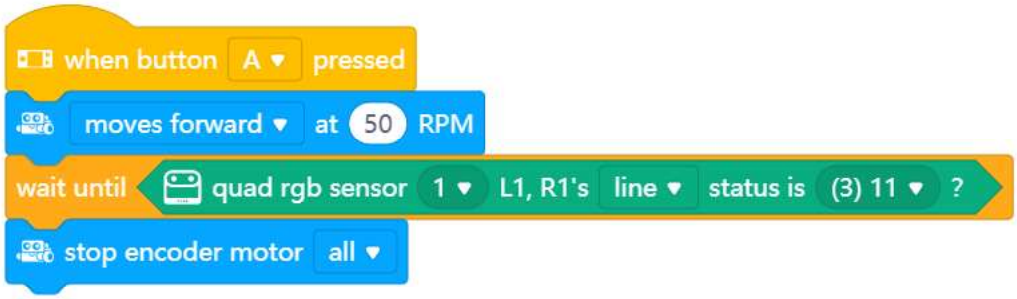
These code blocks take only the inner two sensors L1 and R1 into consideration for line detection. These blocks are intended for simple line-following.

Use the first block in conditional statements. Use the second block if you want to report the value to the screen or save it in a variable.

If the sensor is above a black line, both inner LEDs should be off. Since each sensor (L1, R1) can either detect the line or the background with this block, there are 4 possible combinations: they will be represented by numbers from 0-3 (for reference, the block shows the binary pattern as well):

L1	R1	Meaning	Status (Decimal)
		Sensor on white background, both LEDs are on	0
		R1 detects a black line	1
		L1 detects a black line	2
		Both inner sensors detect the black line	3

Remember, the LEDs show the background state (on or logical "true" = background detected), while this code block detects the line color (logical "true" = line detected).
If you wanted to use more than one Quad-RGB-sensor, you can tell the program which sensor to use with the first number in all the following code blocks. Take a look at the image below. Number 1 indicates the first connected sensor, number 2 the second connected sensor and so on.
The following example will use the line detection function of this block and stop the robot as soon as it arrives at a black line. This example will be changed later to line-following.



Since reacting while driving can be time-critical, it is important to use the Upload mode. You can try by yourself and see the difference if you switch modes.

Code blocks:

Use the first block in conditional statements. Use the second block if you want to report the value to the screen or save it in a variable.

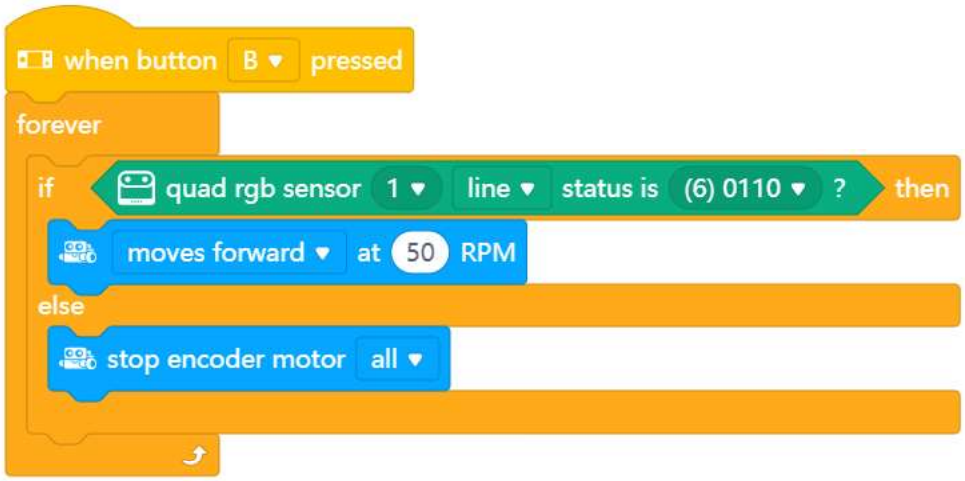
These blocks follow the exact same logic as the previous ones, except they have a larger detection are and can identify crossings while still doing line-following. Remember, the blocks check for the line status, so if one of the four sensors identifies a (dark) line, the corresponding binary digit will be set to "1" (logic true), but the LEDs will go off (no reflection on this sensor from background).

With now 4 individual sensors, the range increases to 16 different statuses. The following table presents the most relevant statuses:

Binary pattern	Meaning	Status (Decimal)
0000	all 4 sensors on white background (no track)	0
0010	only R1 on black line	2
0100	only L1 on black line	4
0110	inner sensors R1 and L1 on black line	6
0111	sharp turn right (L1 and R1 on black line, junction to the right; L-shape junction to the right)	7
1110	sharp turn left (L1 and R1 on black line, junction to the left; L-shape junction to the left)	14
1111	T-shape junction (L1 and R1 on black line, junction to both right and left)	15

An X-shape junction can only be detected by driving over the T-Junction and checking if the middle line continues.

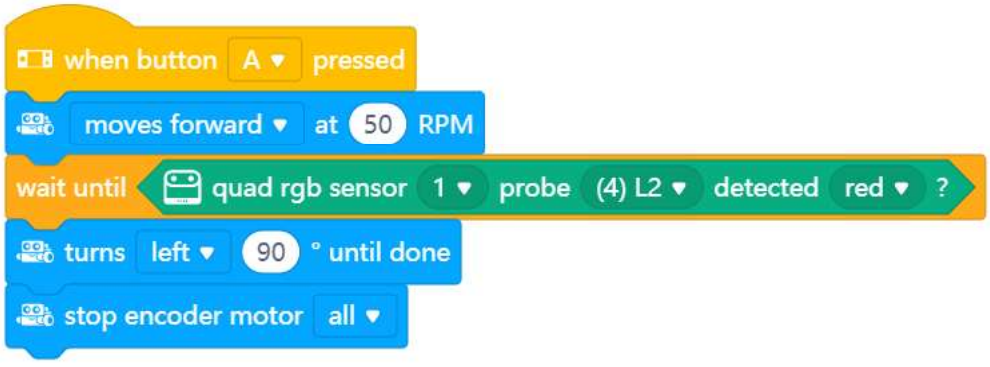
The following example program will make the robot drive as long as it is on a black line with the inner two sensors. It will stop if it goes off track, a junction is detected, or the line suddenly ends:



Code block:

With this code block you select one of the Quad RGB sensors (L2, L1, R1, R2) and determine whether it should recognize a line, background, white, black or any other of the 5 colors (red, green, blue, yellow, cyan, purple). For line-detection and following, the previous blocks are recommended, because they check more than one sensor at the time (faster, more accurate since the robot does not move between readings). But you can use this block to detect for example a red color marker on the left side of the robot – while the robot is following a line.

In the following programming example, if you press the B-button, the mBot2 drives straight (assuming it follows a line) and does a 90° turn when a red marker is detected on the left side of the (imaginary) track.



Code block:

This code block measures how far the robot is off from a black track. You can use it for advanced and smooth line following. The previous line following code block uses a triple distinction: on track, left off or right off (plus added junction detection in some cases). But in everyday life riding a bicycle, you would adjust the steering to the curve more precisely. This block allows a proportional line following: the more the robot is off the track, the more it will steer into the opposite direction. If the offset from the track is 0, the steering is 0 as well (straight).

In step3, we will discuss how we can use this block to follow a line "smoothly" by programming a proportional line follower. The example program below will display the deviation on the screen, so you can try moving the robot on the track by hand first:

2. Reproducing and testing some programming examples of the Quad RGB sensor

In the diagram above, each code block of the Quad RGB sensor has a programming example. You are going to recreate these programming examples in mBlock and test them. For one programming example, come up with your own modification to the code.

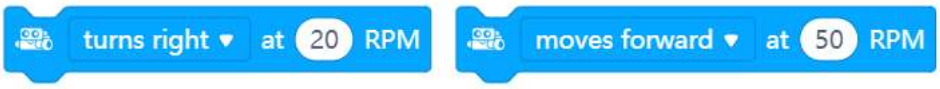
3. How can the mBot2 follow a line?

The basis of the assignment in 'Step 3' of this lesson is that the mBot2 is going to follow the track on a map, being a touring coach. For this purpose, you will need to use the code examples explained and tested so far and modify them accordingly.

First use this coding block:

If the robot is on the track (both L1 and R1 detect a line, code value3), the robot should drive straight. If only one of the two inner sensors identifies a line, then it should turn left or right accordingly.

Use these driving blocks (only turning right is shown):



They just start the motors as needed, and the next code block will be executed immediately. Since you don't know how far the robot is off or when a next curve will occur, the driving commands cannot contain any driving time or distance (same for turning). The next iteration of a program loop will check the new sensor data and select the driving accordingly.

If you are using the provided map, make sure to calibrate the sensor as described to the brightest color (yellow section on the track).

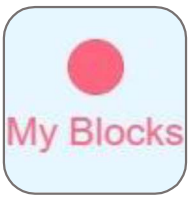
3. Trying out (25 min)

Step 3: Trying it out

You have already learned a lot about the Quad RGB sensor on the mBot2. You are now going to create your own computer program in mBlock using the Quad RGB sensor.

In this assignment you will turn the mBot2 into a touring car. The tour coach takes a tour of landmarks in a city. In the box of the mBot2 there is a map with a track. The coach follows the black line on the map. In the previous part you should have already programmed a simple line following robot, now it is time to add more enhancements.

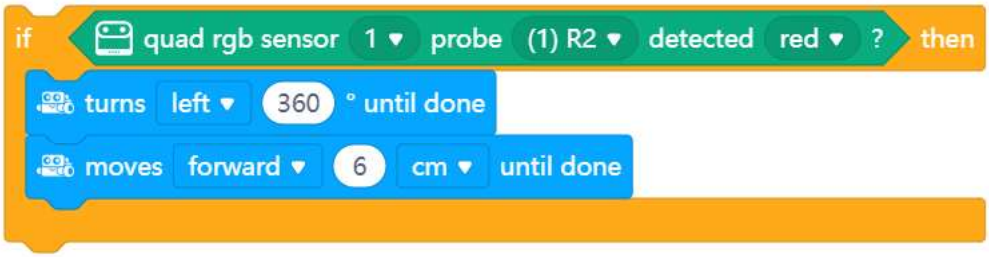
To make the program easier to understand, you can "build" your own coding blocks. The entire line-following part of the program can be added up to one simple command, e.g., "simple_line". The red/pink "My Blocks" in the code menu allows to define a custom block and structure your code in a nice way. The example program below makes use of custom blocks.



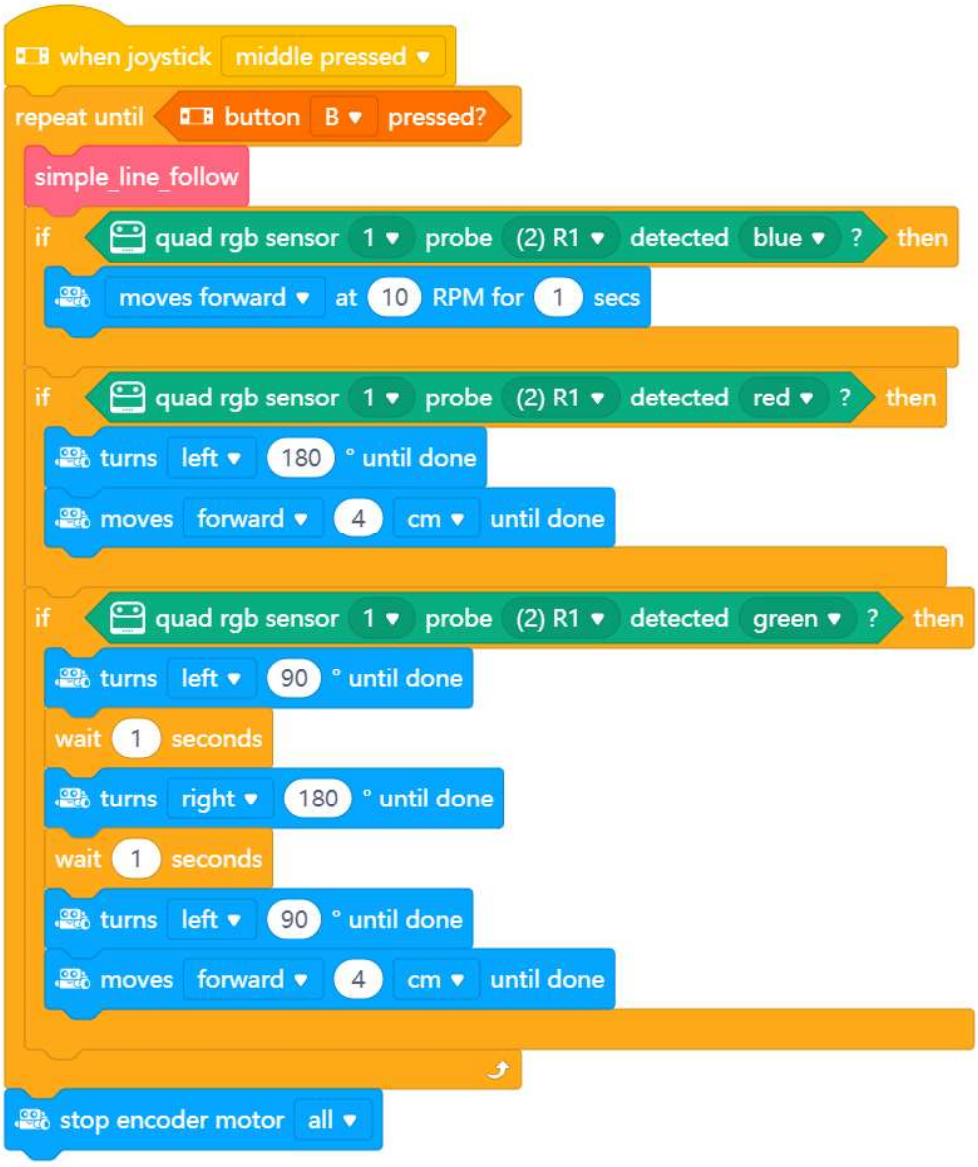
In addition to following the track, the coach performs an action at some of the sights. The sights are indicated by a colored area (red, yellow, green, and blue).

You can think up the actions that the coach should perform. Are you going to have the coach drive around Paris? Then it might be fun to play the French national anthem. Or the bells of Big Ben if the coach is driving through London.

Of course, you can also keep it a little simpler. In the example below, you have the coach turn around at the red marker and do different activities at other colored markers.

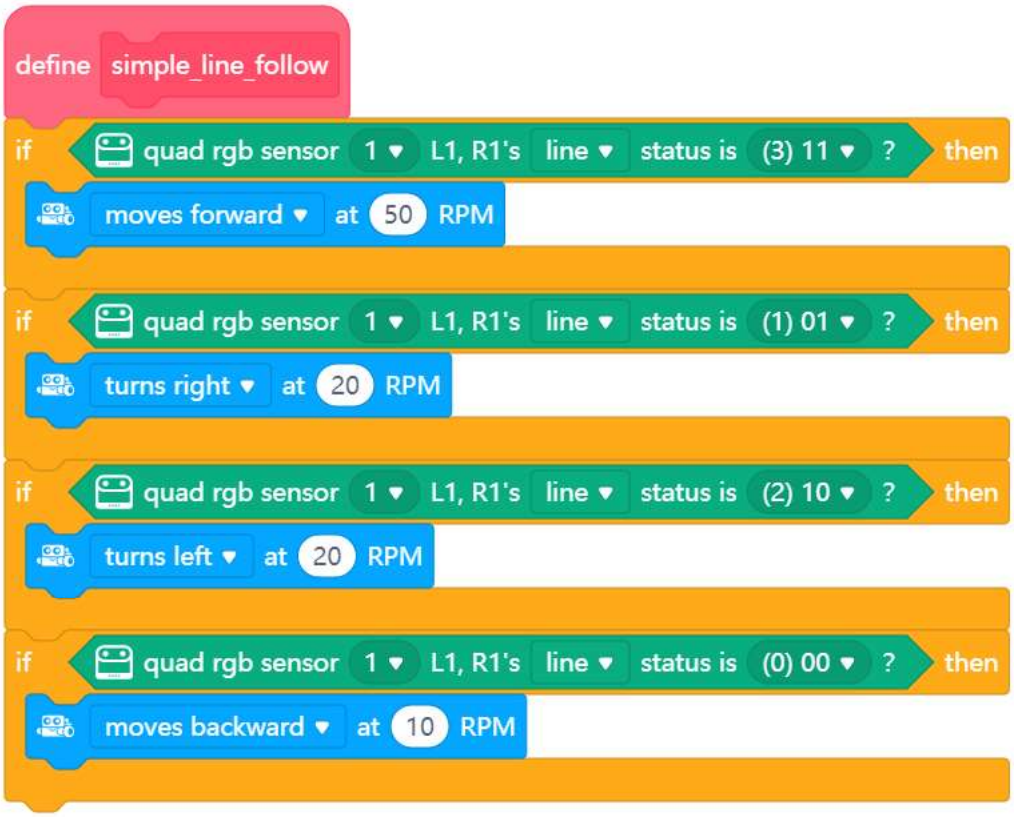


Use the knowledge you gained in 'Step 2' of this lesson. Of course, you can do plenty of experimenting yourself with the different programming possibilities in mBlock. Do you get stuck? Then you can use the programming examples below. You can copy it and adapt it to your own wishes and preferences.



This program starts if you press B and it stops with pressing A. It performs a simple line following using custom "My Blocks"-code and then checks if the R1 sensor detects either blue (drive very slowly) red (turn around), or green (look to each side). To make sure the color does not get recognized again after the activities were done, the robot drives 4cm forward each time.

The "simple_line" block is defined as follows:



Advanced version:

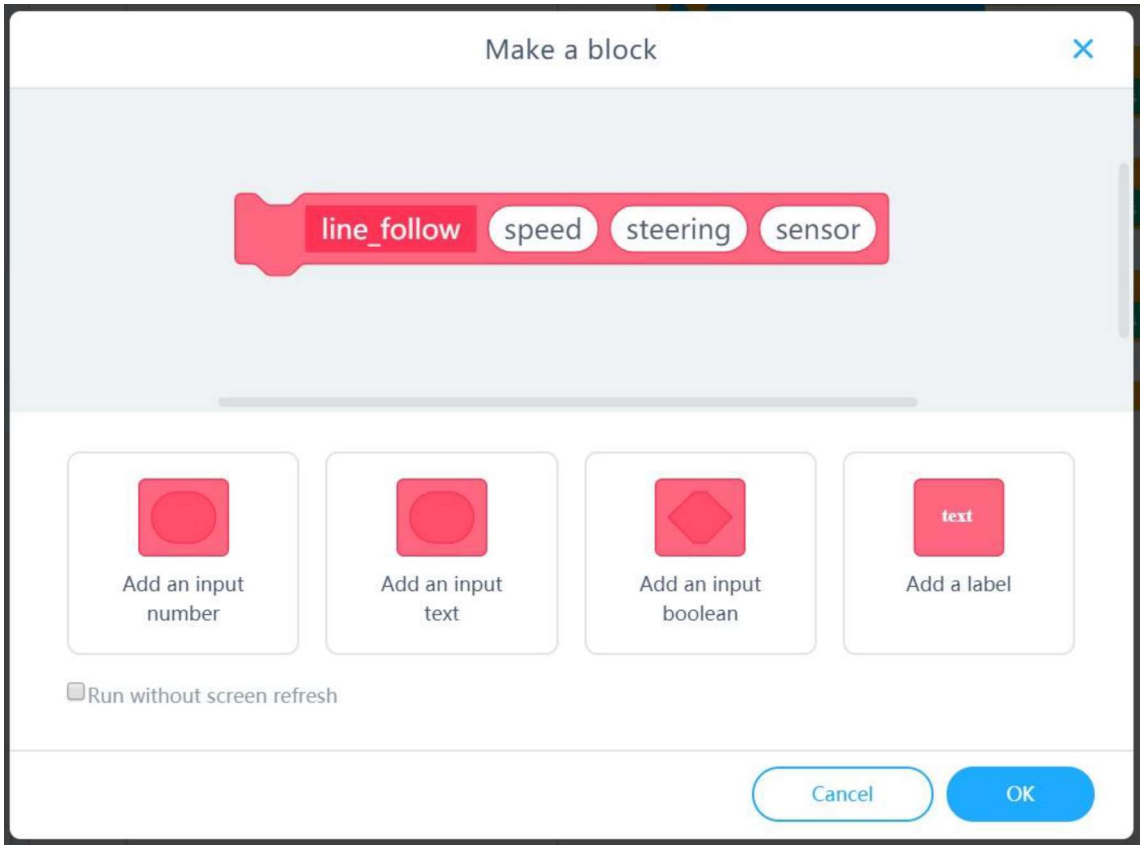
In the previous step, the deviation from the line was introduced. This block reports how far the robot is off track (-100 to +100; 0 for perfect in center) and it can be used to calculate the steering of the robot. This is a proportional line following – the more the robot is off track, the more it needs to steer against this. If the robot is right off track, the deviation is positive (robots needs to drive to the left); for negative values, the robot is left to the track (needs driving right).

For implementing this, you need a basic speed – that is the speed for moving forward on a straight track. For steering left or right, one of the wheels needs to turn faster than the other – the greater the difference, the greater the steering.

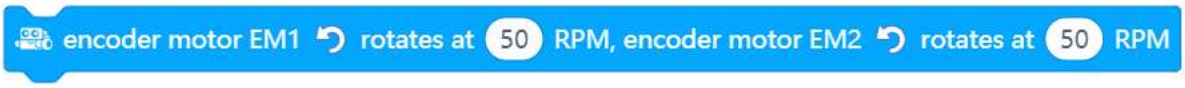
To calculate the steering, we need to add the deviation to one motor's basic speed and subtract it from the other. To allow for a finder adjustment, a factor will be used to adjust how strong the deviation affects the steering, since the range for deviation itself is too big to be used directly for controlling the motors.

But to which motor must the calculated value be added or subtracted? A positive deviation means the robot is off track on the right side and must turn left. This means that the right wheel must run faster – we add the scaled deviation to the right wheel and subtract it from the left therefore.

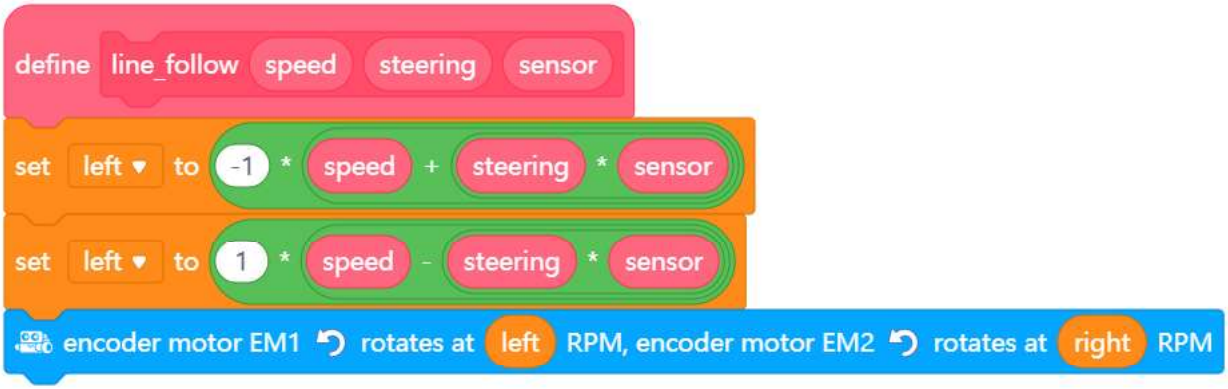
When setting up the "My Blocks", you can add inputs and rename them to a meaningful name:



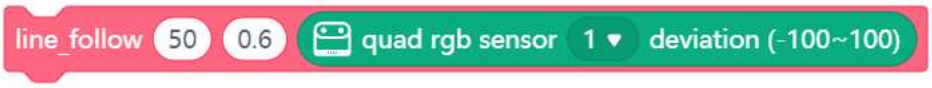
The only thing left to take care of, is that the movement block (shown below) requires negative values for the right motor and positive for the left one to drive forward (this is because they are mounted facing each other).



To finalize the calculation, the values for the right motor need to be multiplied by -1. This is the definition of the proportional line following block:



In the main loop of a program, you simply use this block with one command:



This calls the sub-program with a basic speed of 50 and a steering factor of 0.6 and sends the current deviation from the sensor reading as well. You can experiment with the values a bit – if the robot "oscillates" (rapidly switching from left to right turning etc.) the steering factor is too high; if it goes off track even on weak curves, it is too low.

When working on this assignment, it is helpful to use the following step-by-step plan. Do you have an idea of what you want to make? If so, first discuss with your teacher whether this is feasible.

	Explanation
Step 1: What do you want to do?	• In what way should the mBot2 move?
Step 2: What do you need?	• What do you need in addition to the mBot2?
Step 3: What code blocks do you need to make the mBot2 move?	• How can you make the mBot2 change direction at each corner? • What code blocks do you need to do this? • Make a brief description on how your program works (pseudocode/natural language, flowchart or UML)

	• If you need further explanation, you can discuss with your fellow students, the teacher, or do a research on the topic. There is help available for every coding block in mBlock as well
Step 4: Testing and implementation	• Is the first version ready? Test it! During the testing round, write down points of improvement • Work on the improvement points until your mBot2 runs error-free over your track • Successful? Film the end result and ask your teacher if you can post it on social media with the hashtag #mBot2

4. Wrap-up (5 min)

Step 4: Wrap-up

Did you manage to program the mBot2 as a touring car around the track?

In this lesson, you learned about the Quad RGB sensor and where you might encounter it in everyday life. You know how to program the mBot2's Quad RGB sensor to make the robot follow a line. You also learned how to program the mBot2 to drive a route based on different colors.

It is now time for a brief reflection. Think on your own and discuss with the group:

- What do you think turned out well?
- What could be better?
- Which parts of the lesson did you find easy and which did you find more difficult?
- What would you like more explanation about?
- Who could help you with that?