Co-funded by the European Union

MSTEM

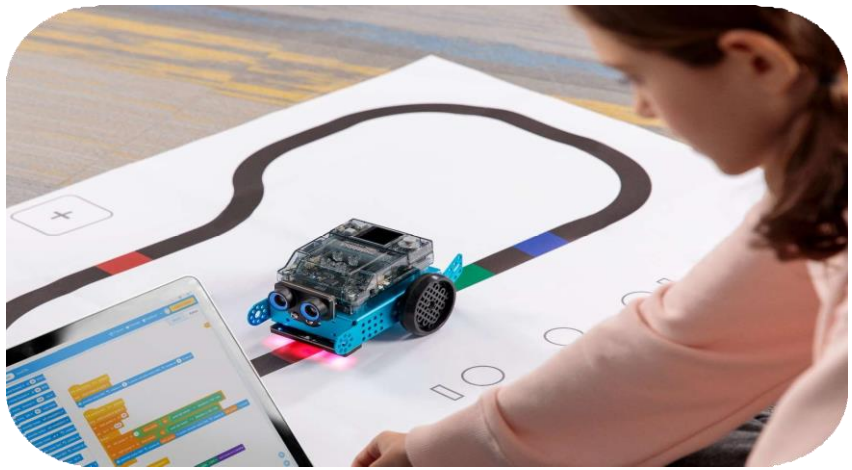
METAVERSE-BASED STEM EDUCATION FOR A SUSTAINABLE AND RESILIENT FUTURE

WORKSHOP mBot2

Co-funded by the European Union

Introduction

- The mBot2 is a state-of-the-art educational robot designed for learning Computer Science and STEAM.
- The mBot2 is designed to enable students to take part in interactive, intelligent lessons that are engaging, fun and reflect real-world applications of cutting-edge technologies.
- Its extensive capabilities make it an ideal solution for primary education, but it can also be used in secondary and higher education.



Co-funded by the European Union

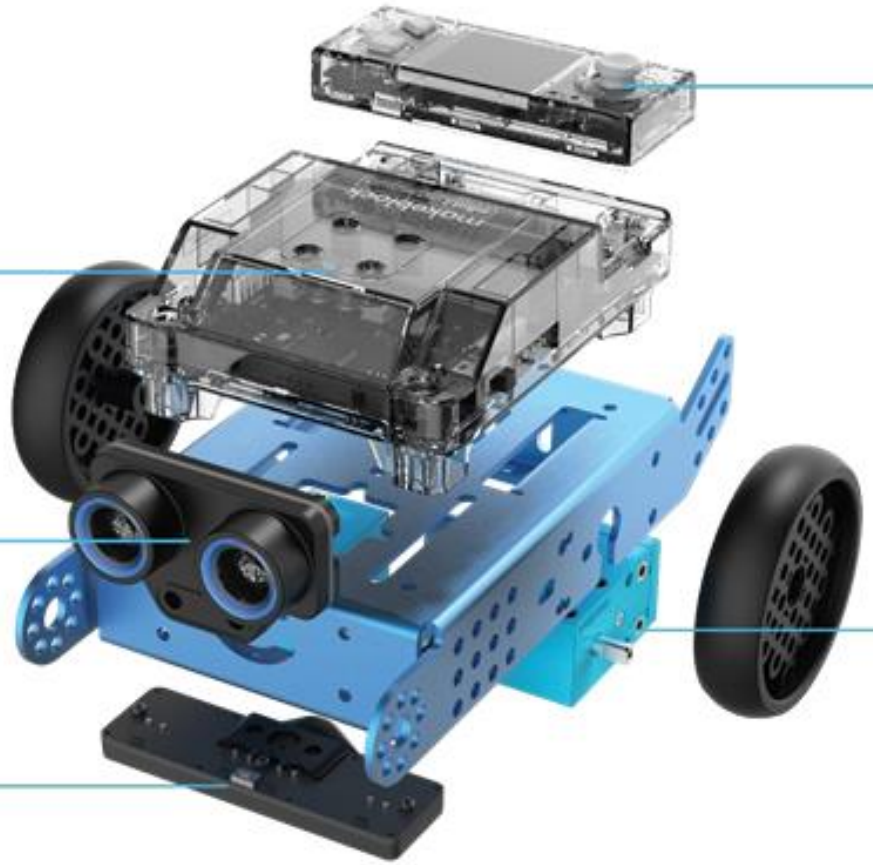
Introduction

- Develop knowledge and skills by combining digital and physical tools that work together to provide a hands-on learning experience.
- Actively engage with the lesson topic to solve a problem or create something new.
- Develop theoretical concepts from practical experience, as well as transfer and apply these concepts back to practical tasks, deepening your understanding.
- Learn from mistakes through a detailed and systematic analysis of the processes involved.

Co-funded by the European Union

mBot2 Components

Here is an overview of the different components thar mBot2 consists



CyberPi

ESP-32 microprocessor for wireless communication. Compatibility with block-based and Python coding

mBot2 shield

Compatible with a wide variety of external components, includes a built-in rechargeable battery

Ultrasonic sensor 2

Object detection is accompanied with 8 programmable LEDs for an enhanced interaction

Quad RGB sensor

4 sensor probes support color recognition, as well as basic and advanced line detection programs

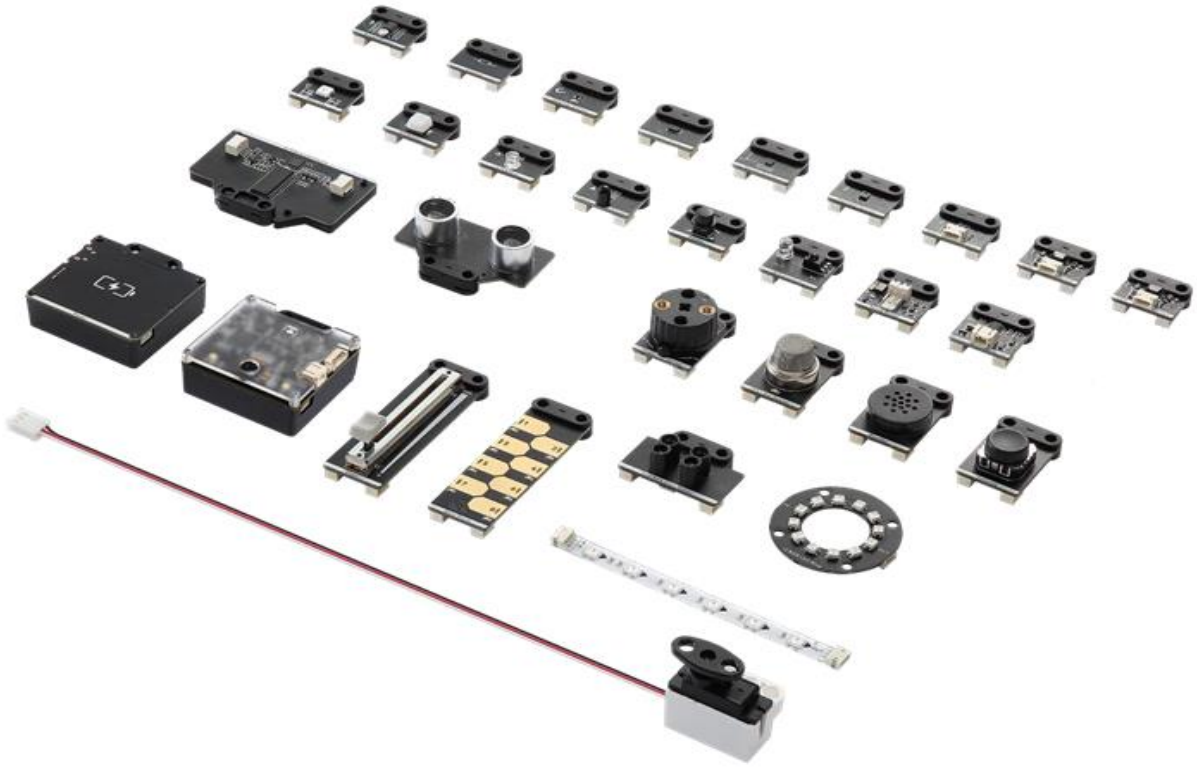
Encoder motors

Measure rotation to 1-degree accuracy, distance travelled and precisely control speed up to 200RPM

Co-funded by the European Union

MBOT2 Components

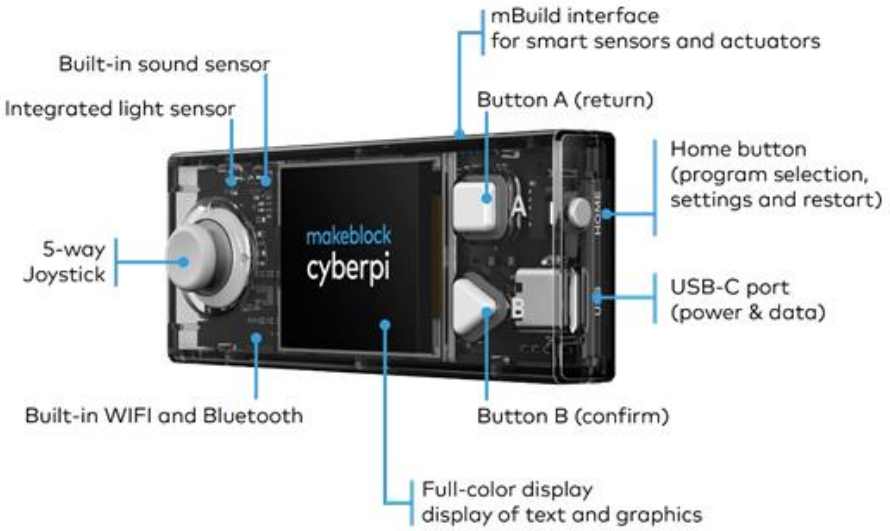
Some examples of sensors and actuators

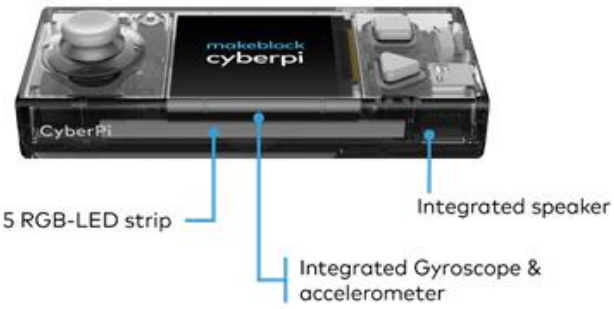


Co-funded by the European Union

MBOT2 Components

The CyberPi has many different functions, such microphone , speaker and a joystick.

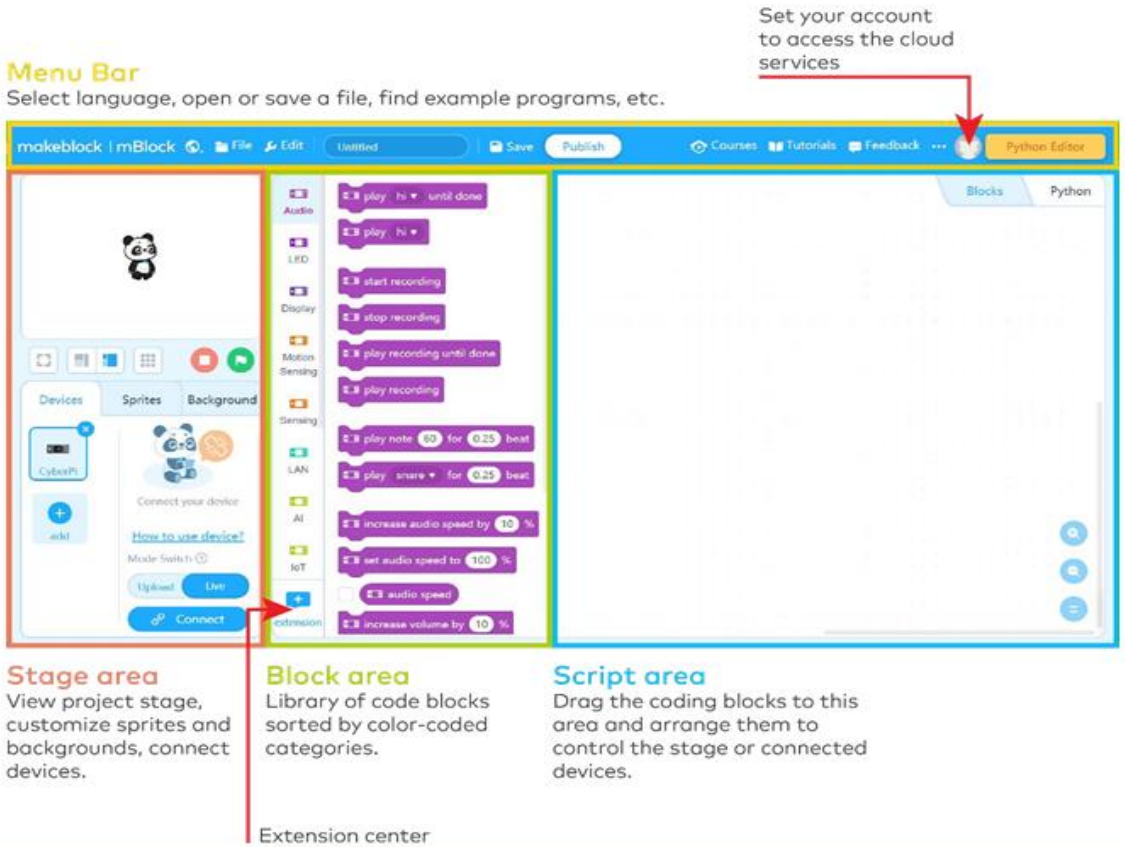




Co-funded by the European Union

Programming mBot2

The mblock interface has the following elements



Co-funded by the European Union

Let´s Move

Description and Lesson Objectives

- ✓ Students discover the mMbot2 and mBlock software and learn to drive the robot with precision
- ✓ Students will also design a simple maze and program the mbot2 to (manually) navigate through it
- ✓ Drive mBot2 with precision

Co-funded by the European Union

Let´s Move

Focus

- ✓ What movement can mBot2 do
- ✓ What different programming blocks can use to make mBot2 move

Co-funded by the European Union

Let´s Move

Activities

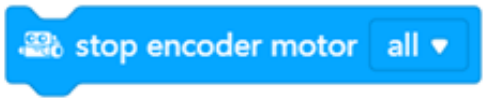
- Get acquainted with the different programming blocks of the mBot2
- Recreating and testing some programming examples to control the mBot2
- There are many different coding blocks you can use to make robot move
- Below you can see some examples of coding blocks needed to make robot move

Co-funded by the European Union

Let´s Move

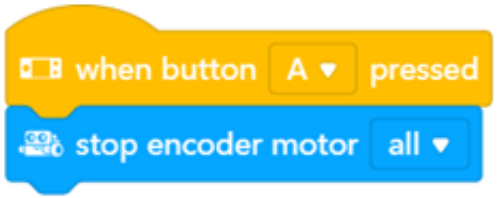
Activities – code block

Code block:



This code block allows you to make your robot stop moving. This can be a useful block during a testing phase. If a program does not work as expected you can use this block to make the robot stop immediately.

For example, you can make the robot stop when the A button of the CyberPi is pressed. To do this, use the programming example below.



Co-funded by the European Union

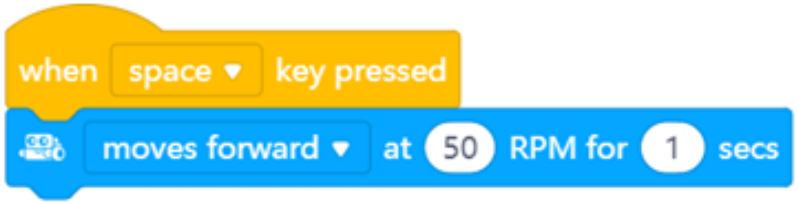
Let´s Move

Activities – code block

Code block:

This code block allows you to move the mBot2 forward, backward, left and right at a specific rotational speed of the wheels and for a number of seconds.

The example below shows how you can make the mBot2 move for two seconds at a speed of 50 rotations per minute. This is useful, for example, when the mBot2 needs to push a load forward.



Co-funded by the European Union

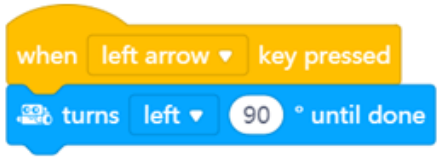
Let´s Move

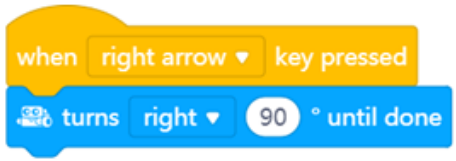
Activities – code block

Code block:

With this code block you can make the mBot2 rotate by a number of degrees, and you can choose whether the rotation should be to the left or to the right.

When the programming example below is set, the mBot2 can be controlled by using the arrow keys. When pressing the right arrow, the mBot2 will turn 90 degrees to the right, when pressing the left arrow, it will turn 90 degrees to the left.





Co-funded by the European Union

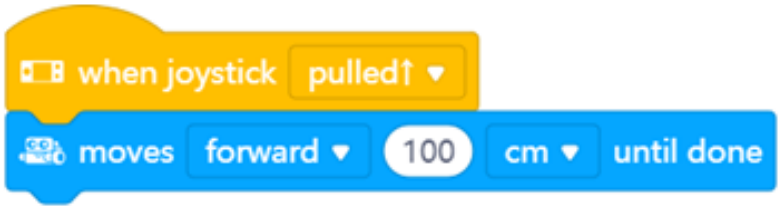
Let´s Move

Activities – code block

Code block:

With this code block you can make the mBot2 move forward or backward a certain distance.

When you use the programming example below, the mBot2 will move 100 cm forward. To start this, move the joystick of the CyberPi upwards.

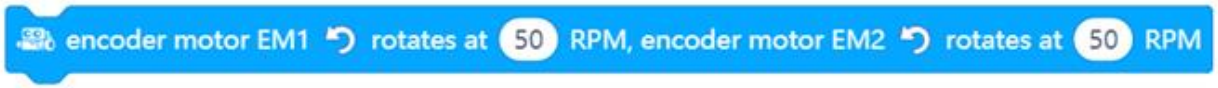


Co-funded by the European Union

Let´s Move

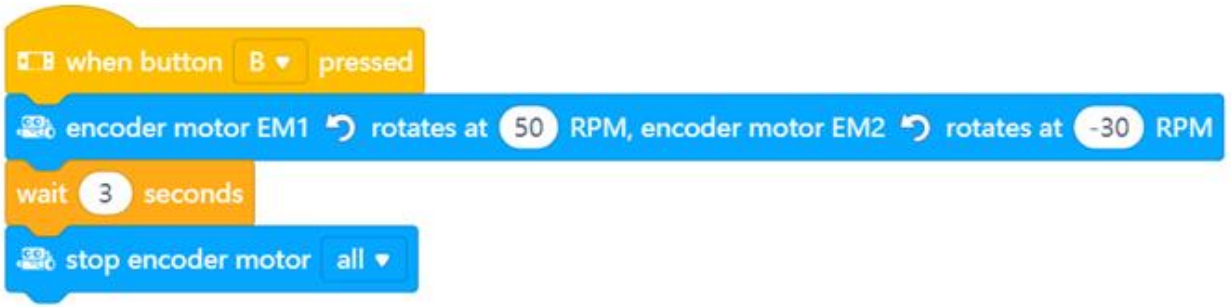
Activities - code block

Code block:



You can also control the motors of the mBot2 independently, and this is one of the blocks which you can use for that.

In the example below, the mBot2 will move in a curve for 3 seconds before it stops. Notice that the value for one of the motors is negative; this is because the motors are mounted in opposite directions, so in order to move the robot in one direction, you need to make one of the motors rotate the opposite way. What would happen if both wheels rotate at 40 RPM?



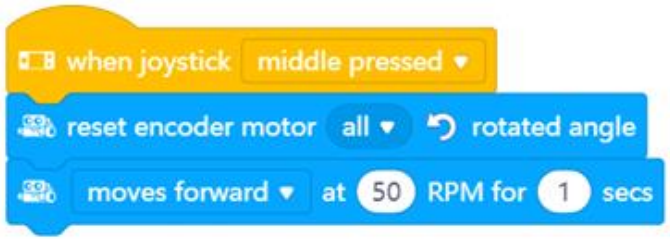
Co-funded by the European Union

Let´s Move

Activities – code block

Code block:

Because the motors of mBot2 can measure their speed and rotation, you can also read these values. The speed measured is the rotational speed in Rounds per Minute (360° = 1 round; in science, it is more common to use "degree per second" (1 RPM = 6° per second)). In mBlock 5, if you tick the box next to this block, you can read the values on the stage above the panda. The programming example below changes the values of the rotation back to zero, and then orders the mBot2 to move forward for one second. What value do you read on the stage after the robot moved?



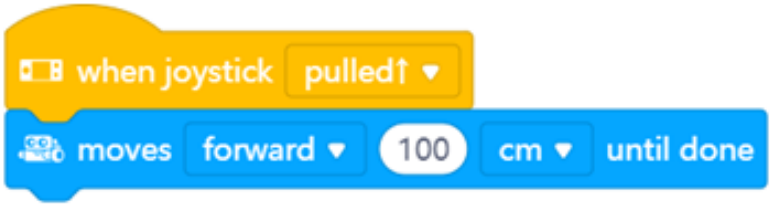
Co-funded by the European Union

Let´s Move

Activities – code block

Code block:

With this code block you can make the mBot2 move forward or backward a certain distance. When you use the programming example below, the mBot2 will move 100 cm forward. To start this, move the joystick of the CyberPi upwards.



Co-funded by the European Union

Let´s Move

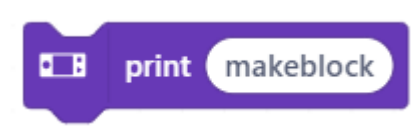
Activities

Below you can see some examples of coding blocks needed to print a label or turn on a led

Audio

LED

Display

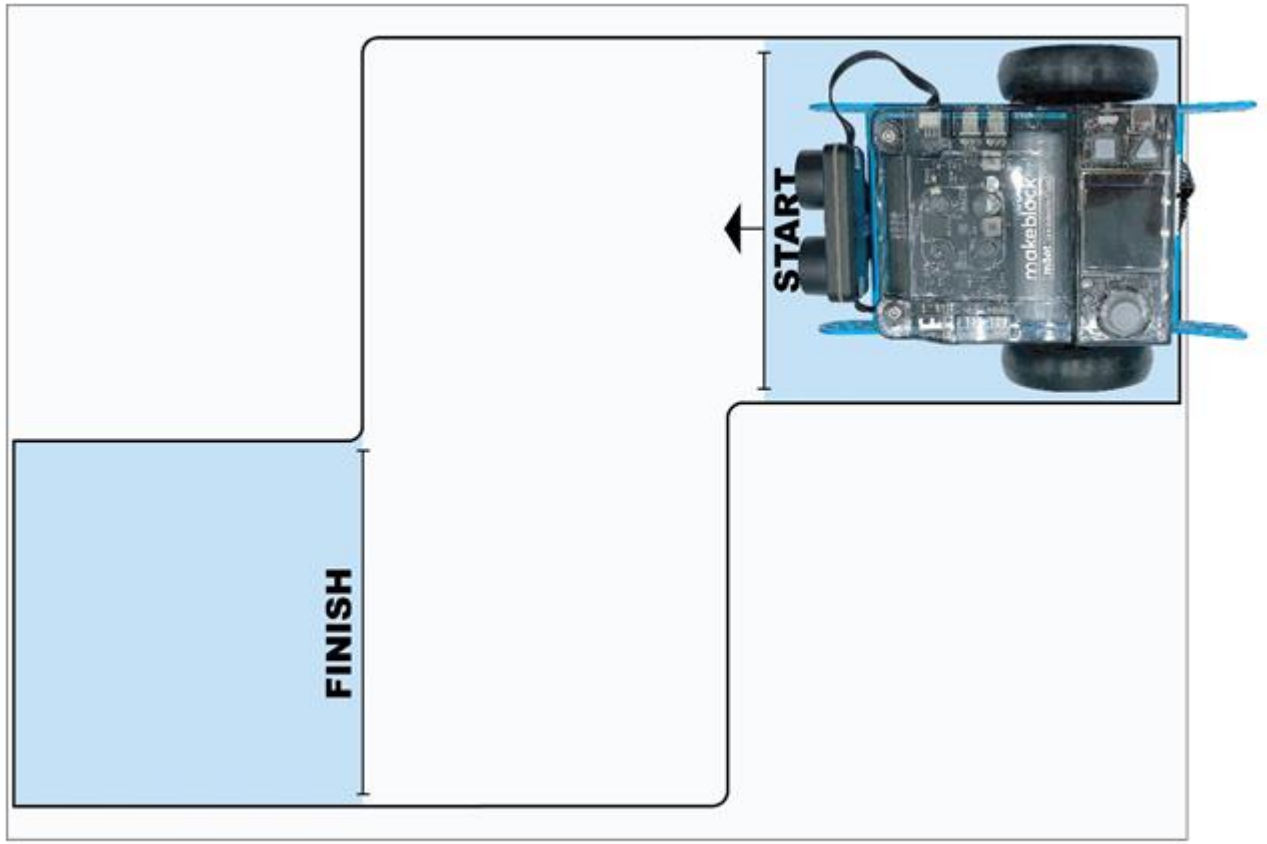


Co-funded by the European Union

Let´s Move

Activities

It is now time to put your mBot2 to work



Co-funded by the European Union

Seeing with Sound

Description and Lesson Objectives

- Students will learn what ultrasound is
- How it is used in a sensor, and they will also create a program to make the mBot2 drive in a loop by turning after detecting obstacles on the road
- Applications of these sensors in everyday live and in robotics

Co-funded by the European Union

Seeing with Sound

Focus

- ✓ Use an ultrasonic sensor for range and obstacles detection
- ✓ Make the mBot2 react to obstacles and avoid them in driving

Co-funded by the European Union

Seeing with Sound

Activities

- Get acquanted with the different programming blocks of the ultrasonic sensor
- Recreating and testing some programming examples of the ultrasonic sensor
- There are many several code blocks for the ultrasonic sensor you can use in the programs

Co-funded by the European Union

Seeing with Sound

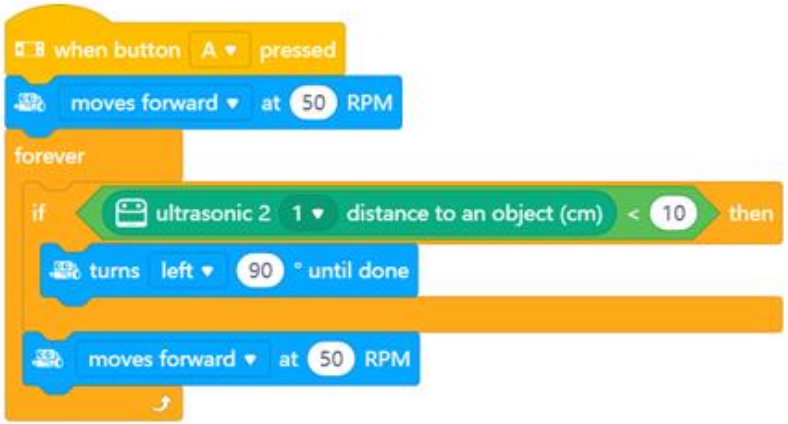
Activities – code block

In the table below, you can see some examples of the code blocks to program ultrasonic sensor

Code block:

With this code block you measure the distance between the sensor and an obstacle. The sensing range is between 3 and 300 cm. You can use the value of the distance to make the robot perform a certain action. For example, you can prevent the mBot2 from colliding with obstacles.

In the programming example below, the mBot2 drives forward. When the mBot2 is less than 10 cm from an obstacle, the robot makes a 90 degree turn to the left. Then the mBot2 drives forward again.



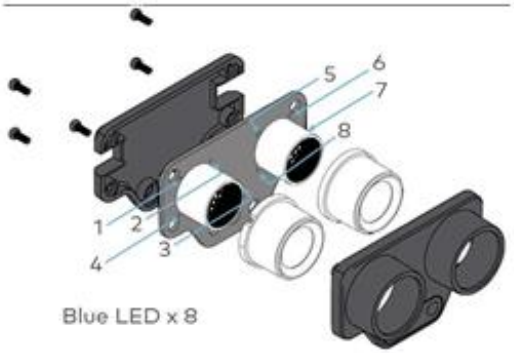
Co-funded by the European Union

Seeing with Sound

Activities – code block

Code block:

The Ultrasonic sensor on mBot2 has eight blue LED lights. You can use these lights to have the mBot2 show a certain "emotion", for example, or as a general visual non-verbal communication. Make the LED lights shine very brightly when the mBot2 is happy and less brightly when it is gloomy. There is a specific code block to show pre-defined "emotions", too. In the image below you can see where exactly the LED lights are located.



Blue LED x 8

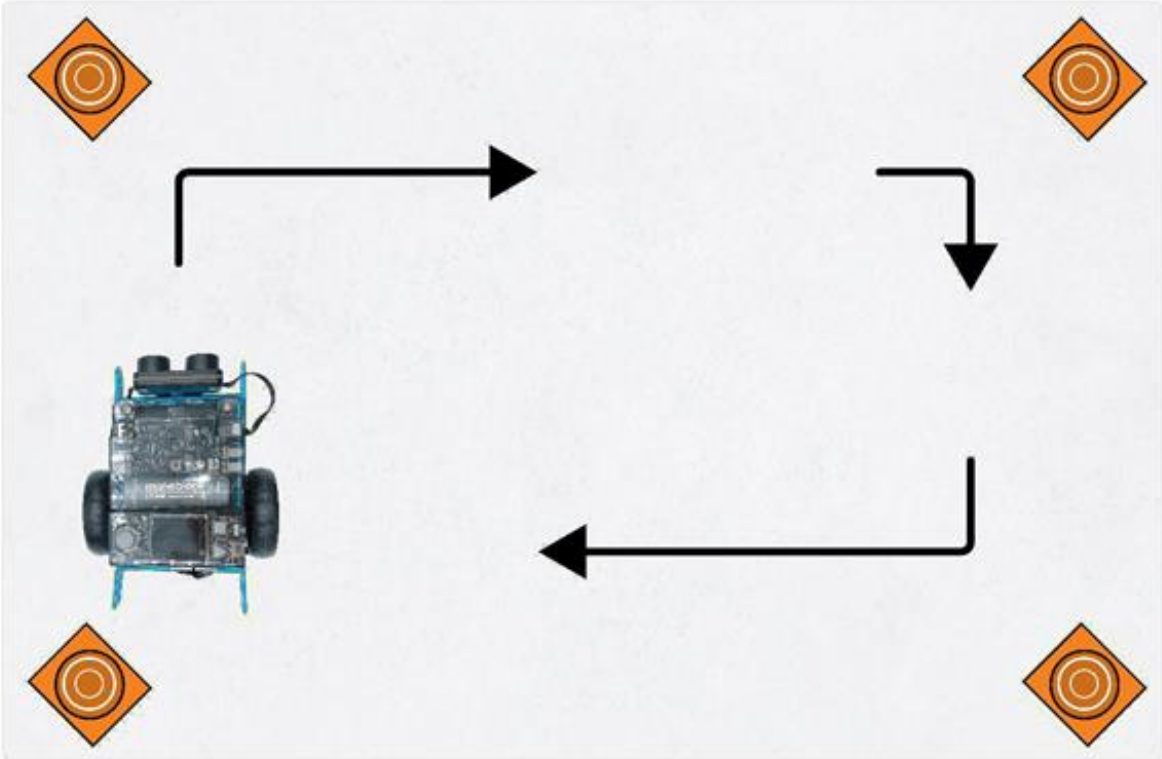
In mBlock there are several code blocks that allow you to program the brightness of the LED lights: you can either set the brightness of the LEDs by decreasing or increasing the brightness by a certain percentage. Or you can set the brightness directly to a specific percentage. The range is 0% to 100%. Both can be done either for a single LED or for all at the same time. In the programming example below, the brightness of the LED lamp 1 is increased by 50%.

Co-funded by the European Union

Seeing with Sound

Activities

You are no going to create you own computer program in mBlock using the ultrasonic sensor



Co-funded by the European Union

Sightseeing

Description and Lesson Objectives

- Students will learn how a color sensor works
- How are they used in a real life , and they will program the mBot2 to become a tour bus that visits different landmarks in a city

Co-funded by the European Union

Sightseeing

Focus

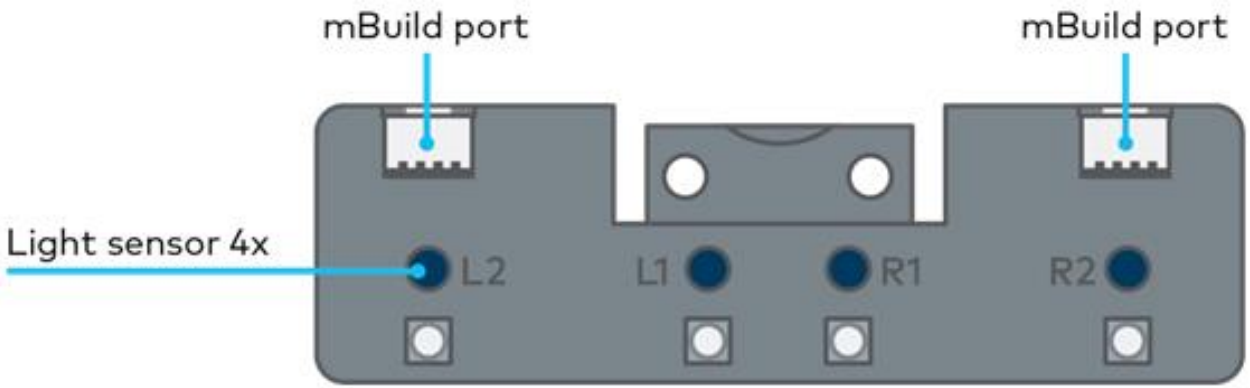
- ✓ Make the mBot2 follow line
- ✓ Have the mBot2 perform actions based in colors

Co-funded by the European Union

Sightseeing

Activities

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



Co-funded by the European Union

Sightseeing

Activities

- ✓ Get acquainted with the different code blocks of Quad RGB sensor
- ✓ Reproducing and testing some programming examples of the Quad RGB sensor
- ✓ How can the mBot2 follow line?
- ✓ In the table below, you can see some of the code blocks you can use in your program with the Quad RGB sensor

Co-funded by the European Union

Sightseeing

Activities - code blocks

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).

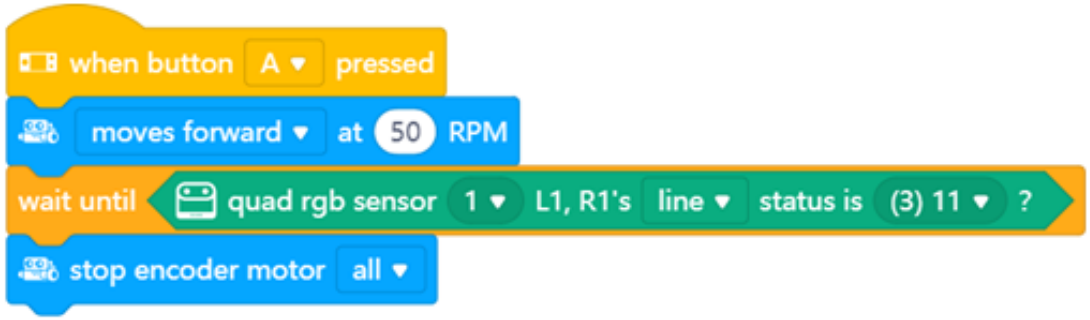
If you wanted to use more than one Quad-RGB-sensor, you can tell the program which sensor to use with the first number in all the following code blocks. Take a look at the image below. Number 1 indicates the first connected sensor, number 2 the second connected sensor and so on.

Co-funded by the European Union

Sightseeing

Activities - code block

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.

Co-funded by the European Union

Sightseeing

Activities - code block

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

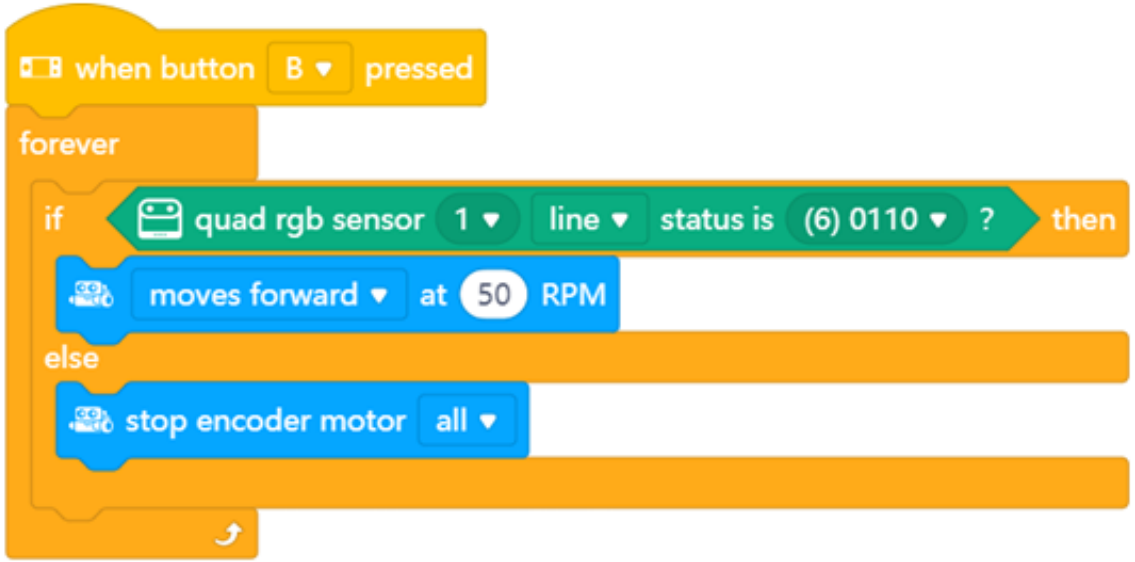
An X-shape junction can only be detected by driving over the T-Junction and checking if the middle line continues.

Co-funded by the European Union

Sightseeing

Activities - code block

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:



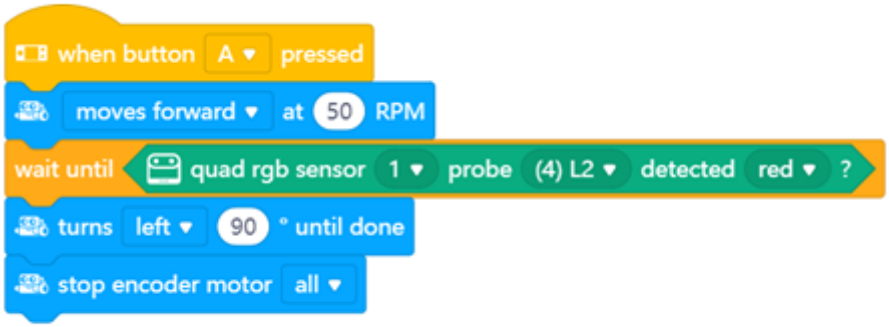
Co-funded by the European Union

Sightseeing

Activities - code block

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.



Co-funded by the European Union

Sightseeing

Activities - code block

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:

Co-funded by the European Union

Sightseeing

Activities

You are no going to create you own computer program in mBlock using the of Quad RGB sensor