Makeblock mBot2

Overview

mBot2 is a next-generation educational robot designed by Makeblock to support Computer Science and STEM learning. It comes equipped with several features, motorized wheel movement, a Cyberpi microcontroller, an LED display, a microphone, a speaker, an ultrasonic sensor and a Quad RGB sensor for colour and line detection.

Appearance

The robot is built on a sturdy aluminium chassis that supports two rubberised wheels, each driven by an encoder motor on either side, along with a mini wheel at the front for balance. An mBot2 Shield, which provides ports for motor and sensor connections, is mounted on top of the chassis, with a CyberPi microcontroller positioned above the shield. The Ultrasonic sensor is installed at the front of the robot, positioned above both the mini wheel and the Quad RGB sensor.

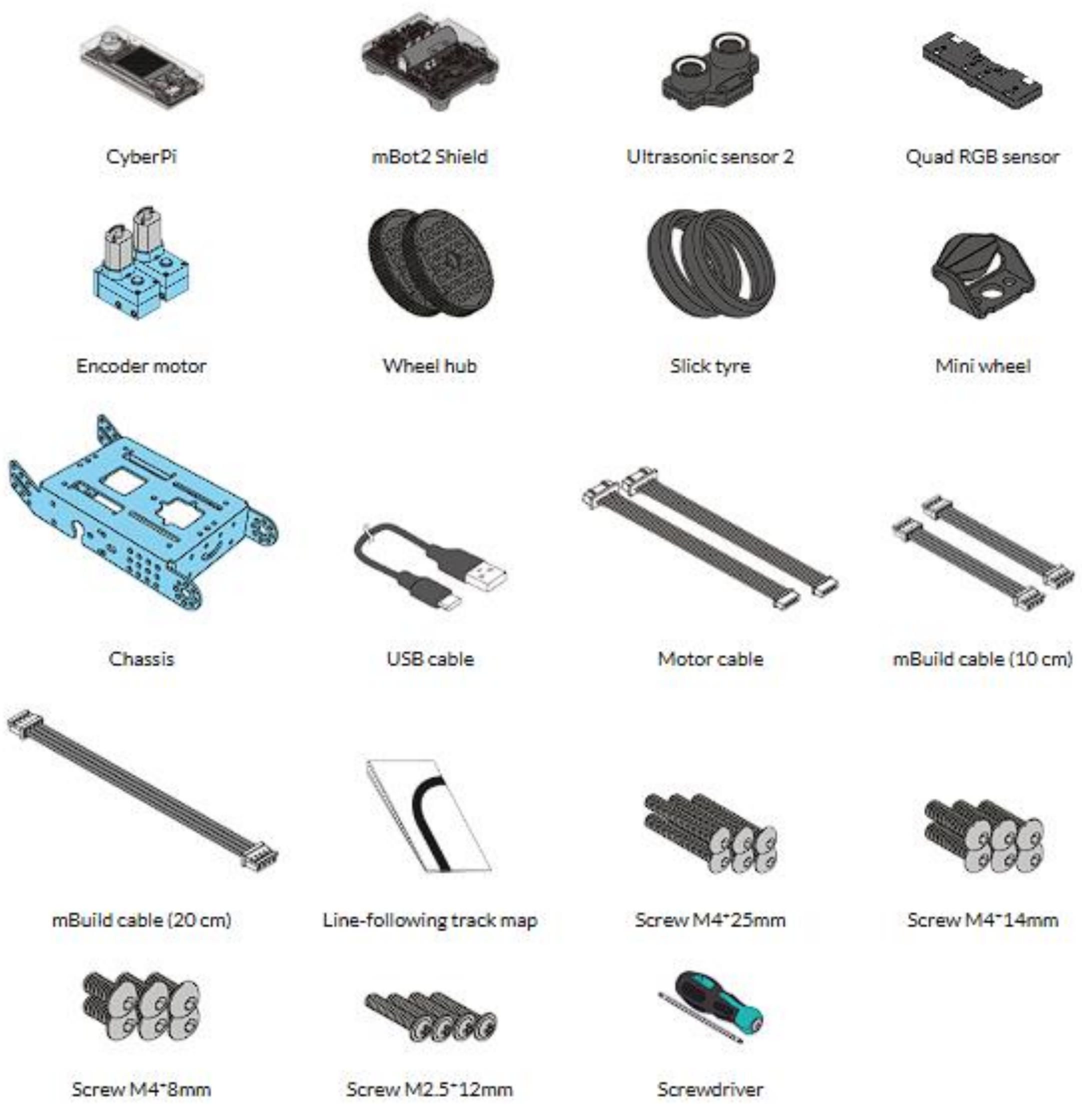
Assembly

Required Parts

To successfully assemble an mBot2 robot, you need the following parts:

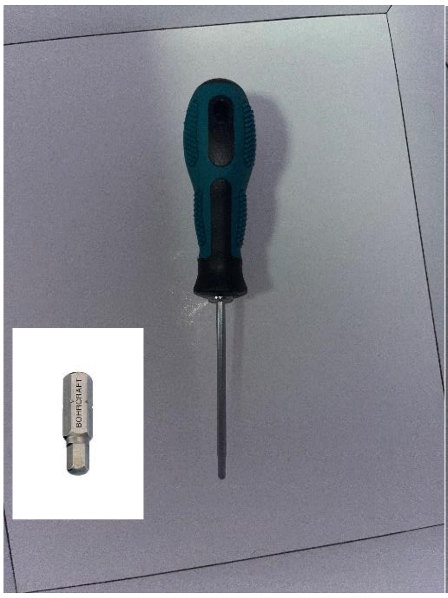
- Screwdriver with two heads
- 1 CyberPi
- 1 mBot2 Shield
- 1 Ultrasonic Sensor 2
- 1 Quad RGB Sensor
- 1 Chassis
- 1 Mini Wheel
- 1 mBuild Cable (20cm)
- 1 Screwdriver
- 1 USB Cable
- 1 Line-following Track Map.
- 2 Encoder Motors, Wheel Hubs
- 2 Slick Tyres
- 2 Motor Cables
- 2 mBuild Cables (10cm)

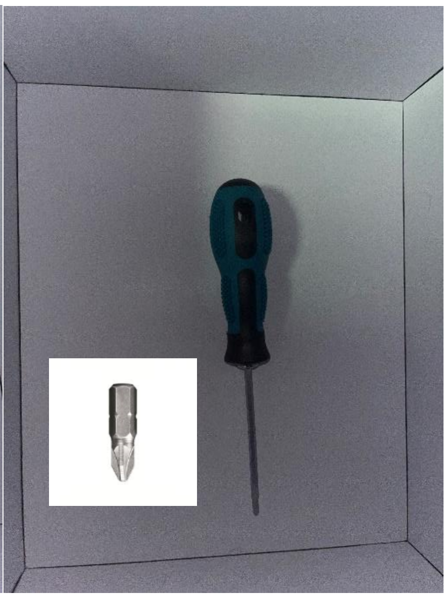
See the picture below for reference



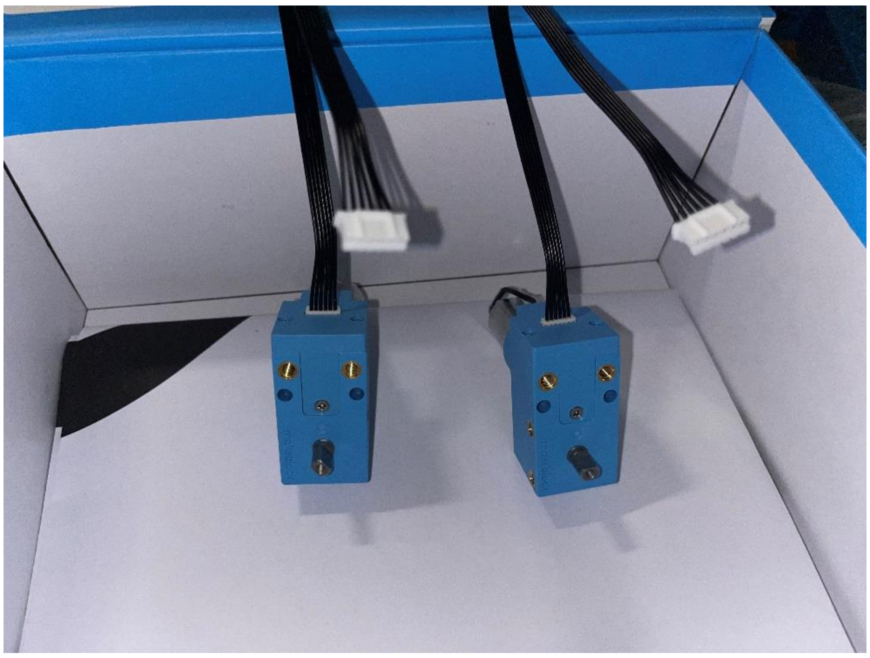
Steps

Step 1: Attach the screwhead to the handle, it can be attached two ways depending on the type of head you need to use.

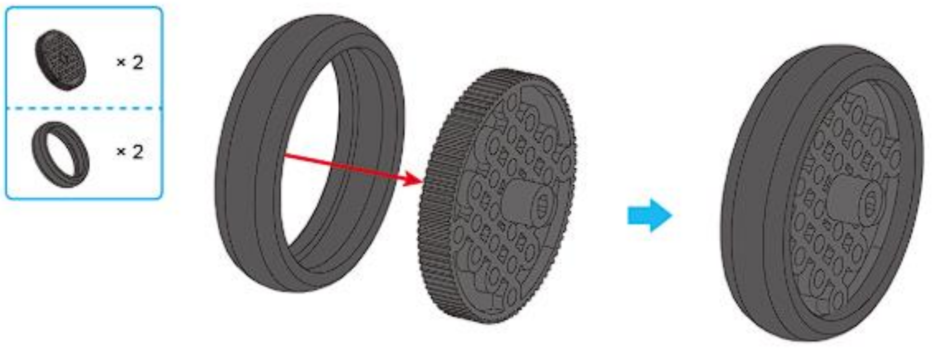




Step 2: Attach the Motor Cables to the Encoder Motors.

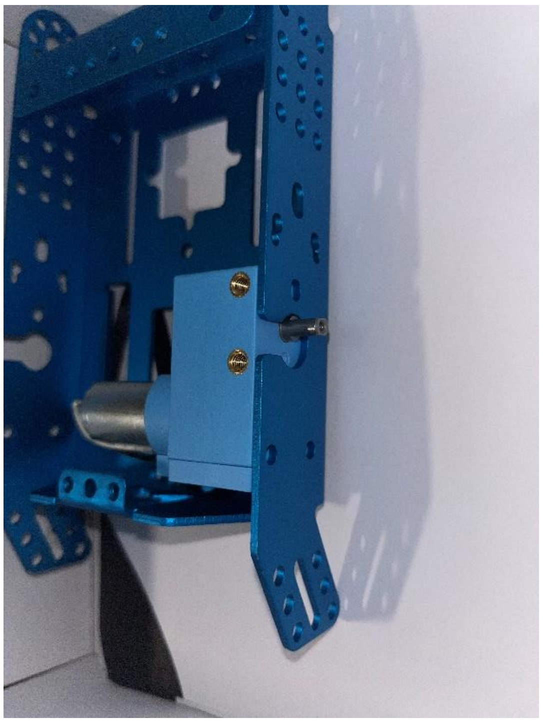


Step 3: Fit the Slick Tyres securely onto the Wheel Hubs.

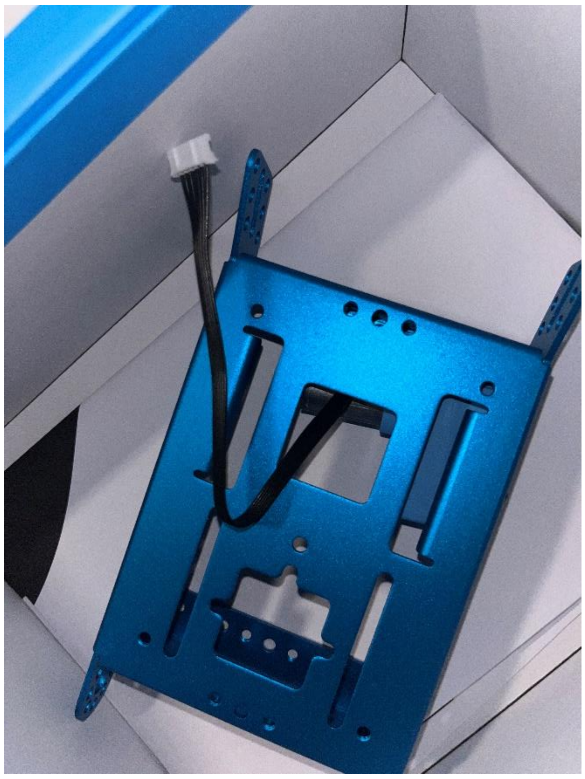


Tips:
Assemble two wheels
in the same way.

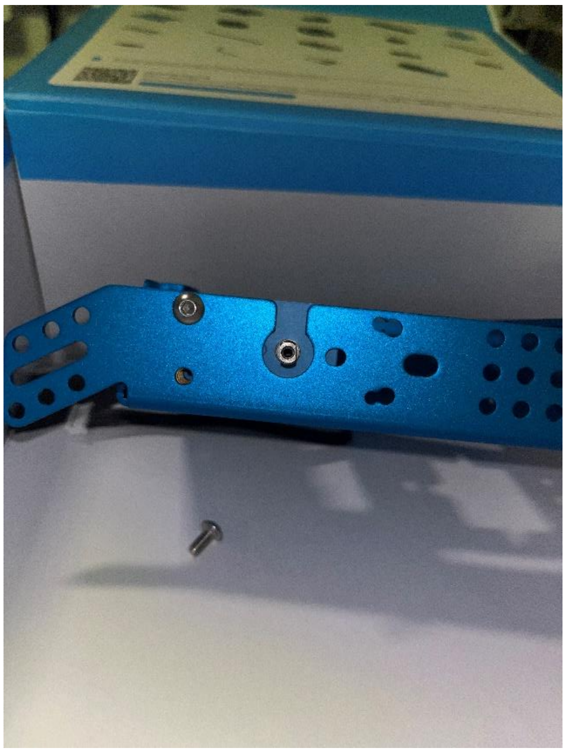
Step 4: Flip the Chassis so the hollow part is facing upwards and place a motor on the Chassis, so the metal head sticks through the circular hole on the side of the Chassis.



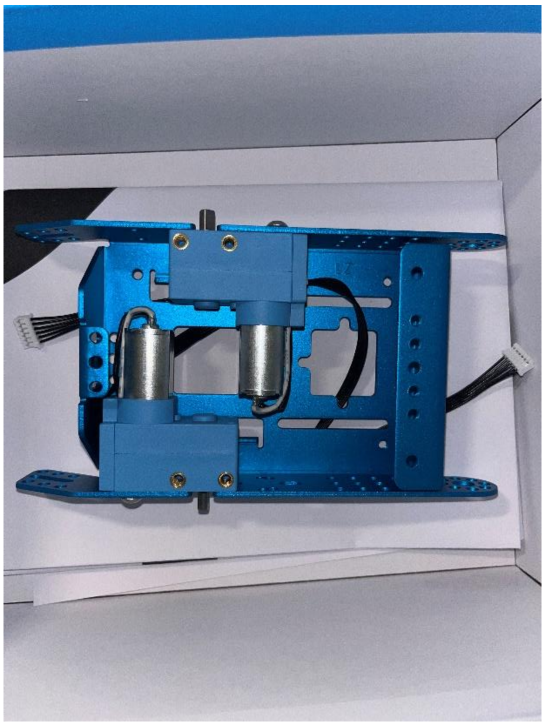
Step 5: Ensure the Motor Cable passes through the square hole underneath.



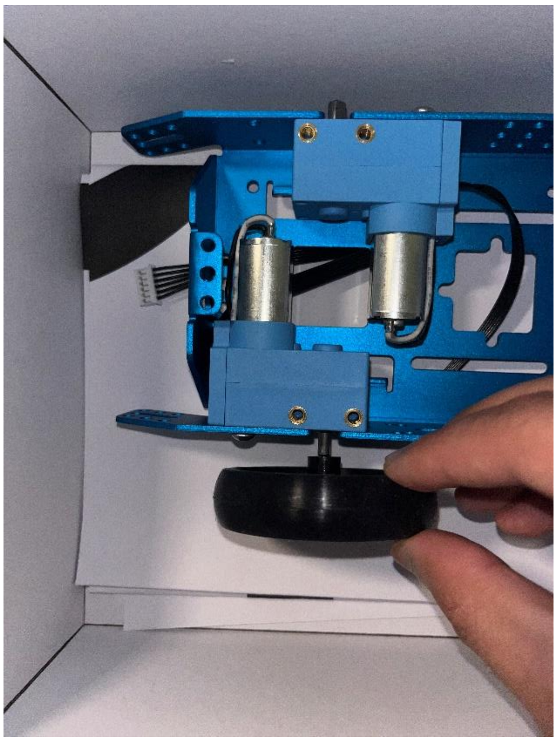
Step 6: Hold the Motor in place by fastening two 8mm screws on the side of the Chassis.



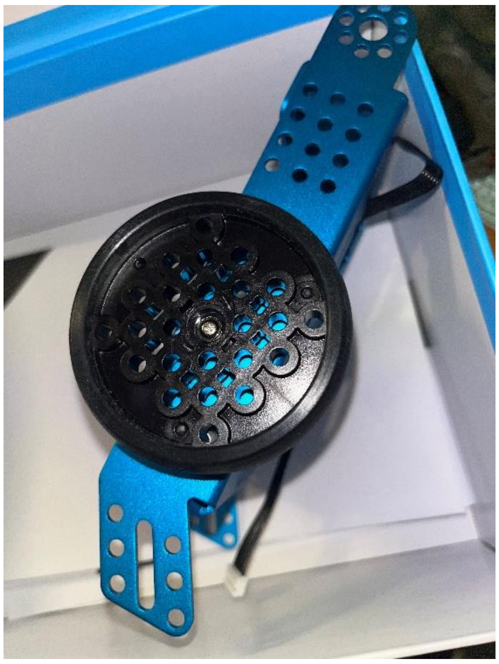
Step 7: Repeat Step 4, 5 and 6 with the second Motor.



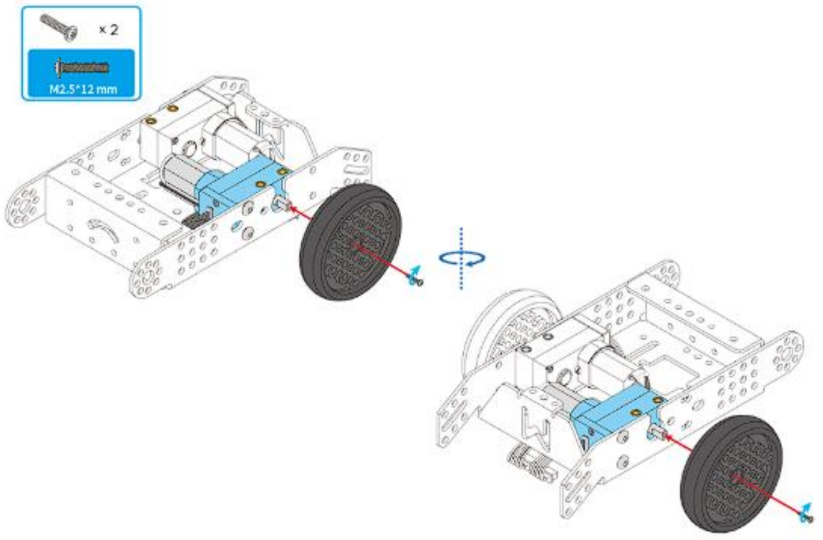
Step 8: Fit the Wheel onto the protruding metal head.



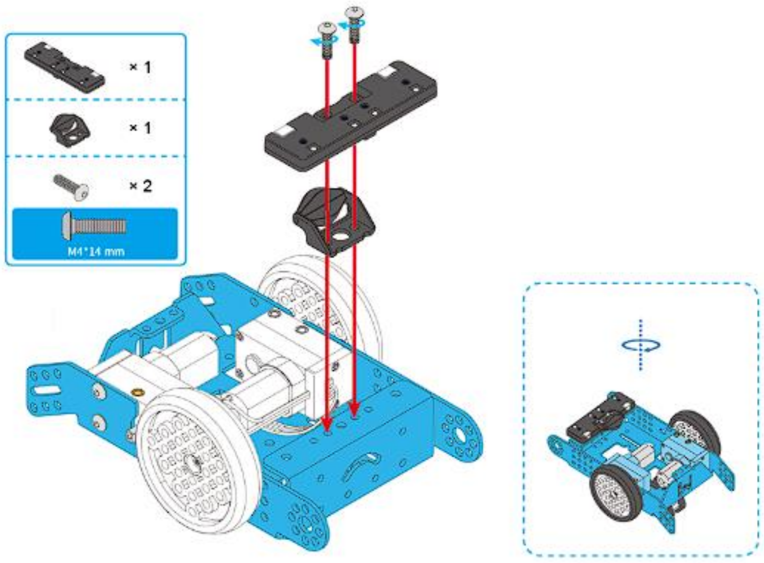
Step 9: Fasten the Wheel with a 12mm screw.



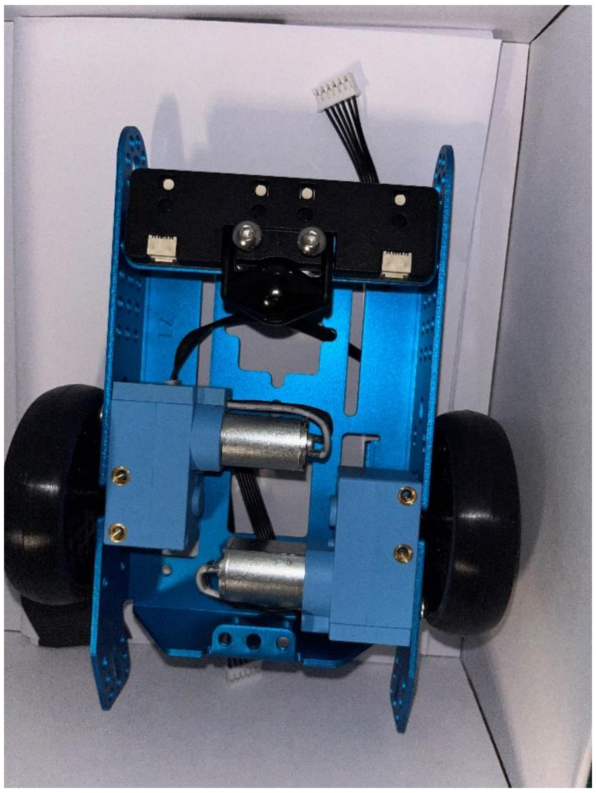
Step 10: Repeat Step 8 and 9 with the second wheel.



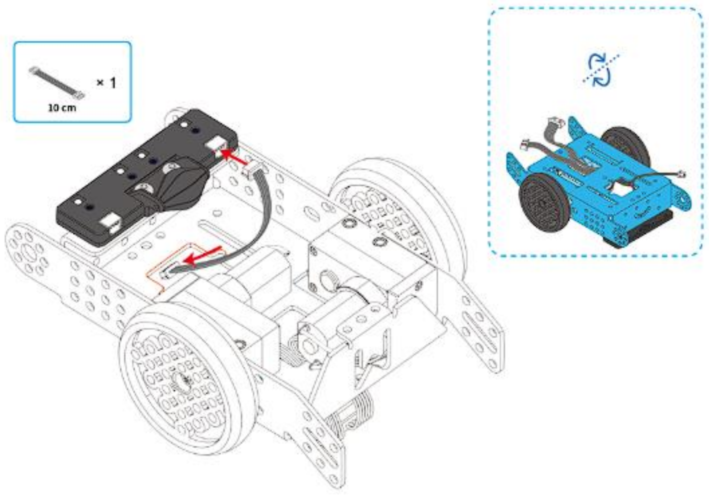
Step 11: Fit the Quad RGB Sensor onto the Mini Wheel.



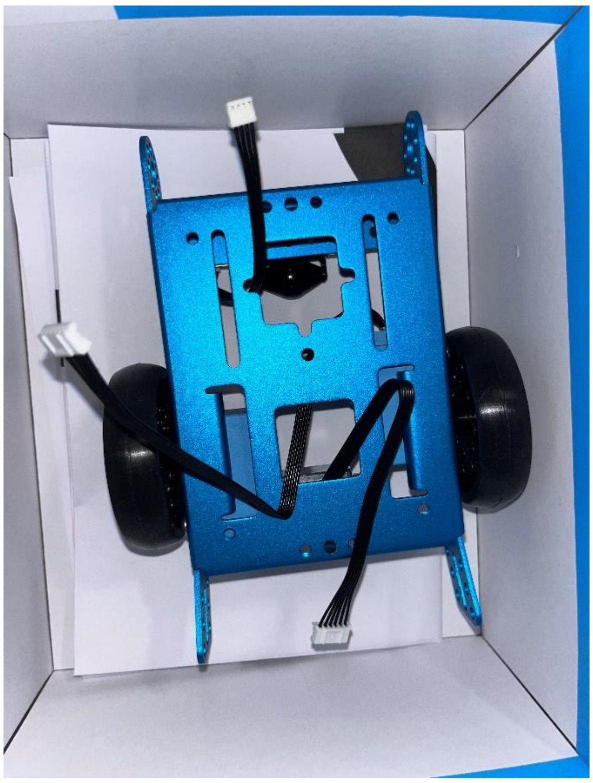
Step 12: Secure the RGB Sensor and the Mini Wheel onto the Chassis with two 14mm screws.



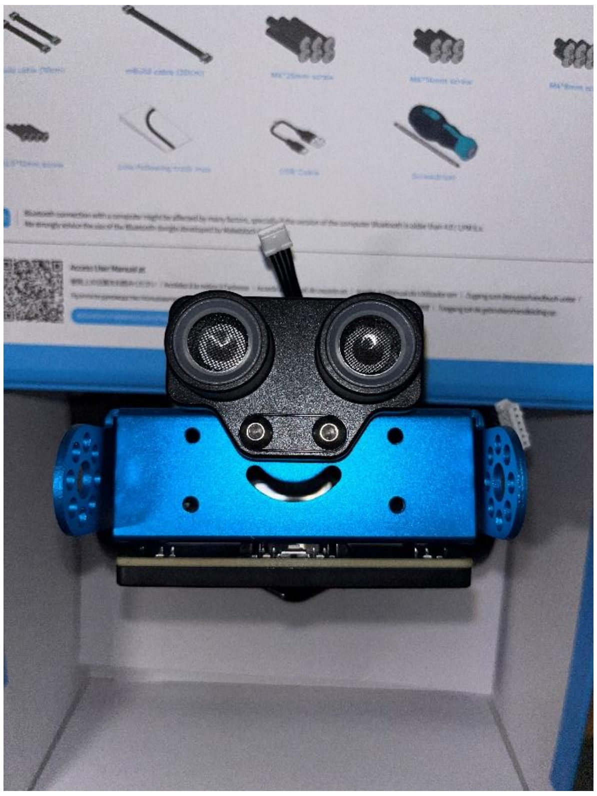
Step 13: Attach a 10cm mBuild Cable to the Quad RGB Sensor.



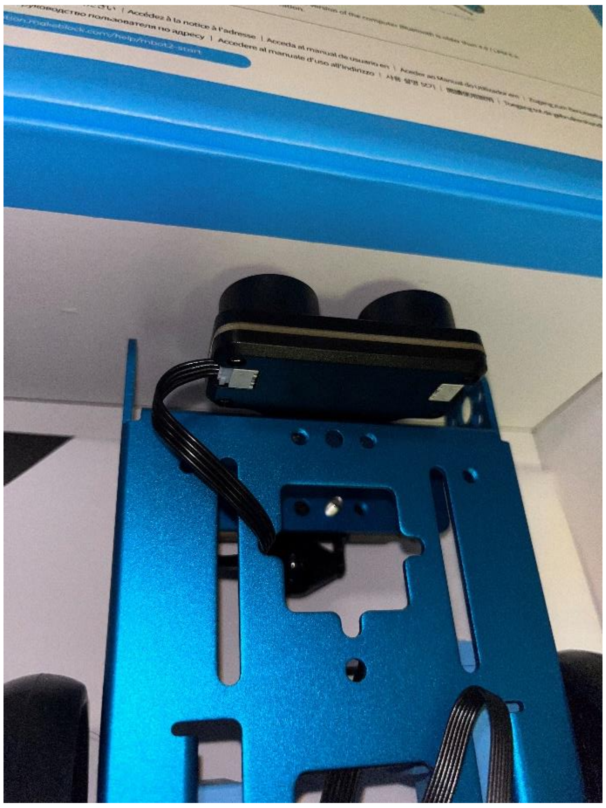
Step 14: Push the other end of the mBuild Cable through the second square hole on the Chassis.



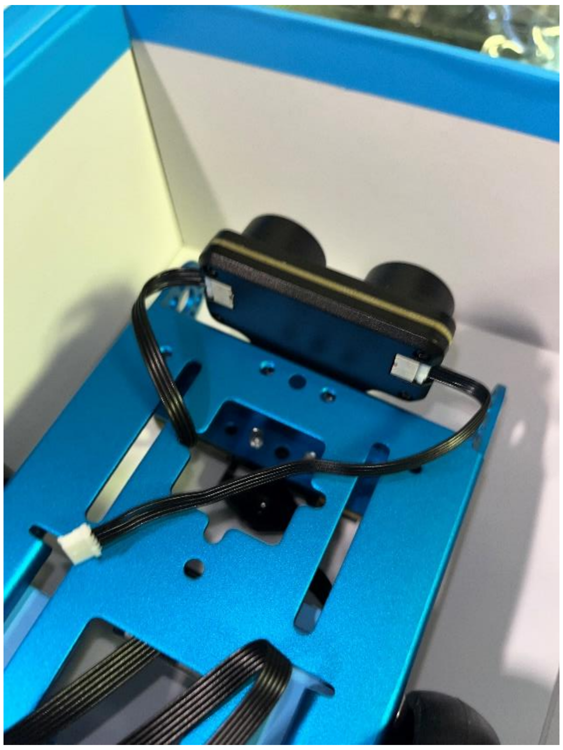
Step 15: Place the Ultrasonic Sensor onto the Chassis and secure it with two 14mm screws.



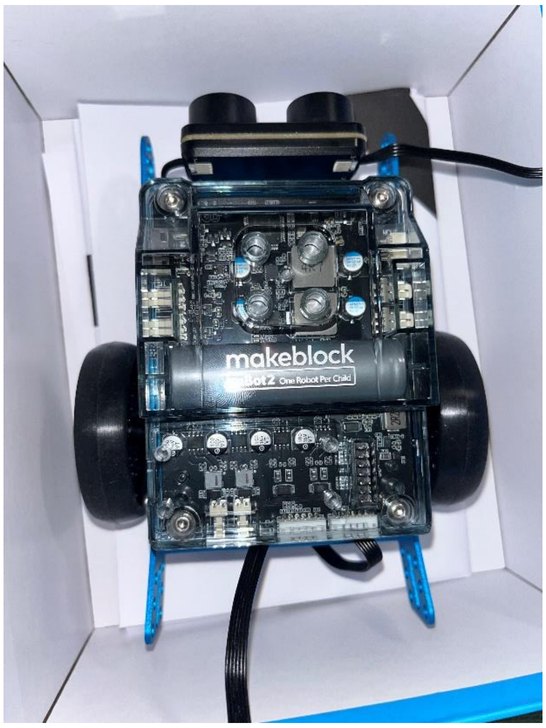
Step 16: Attach the other end of the mBuild cable (the one attached to the Quad RGB Sensor) to the Ultrasonic Sensor.



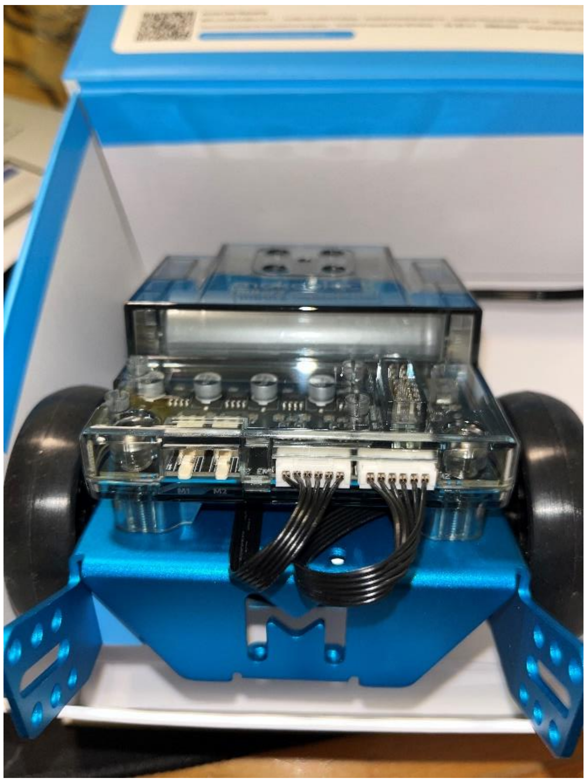
Step 17: Attach the second 10cm mBuild Cable to the other side of the Ultrasonic Sensor.



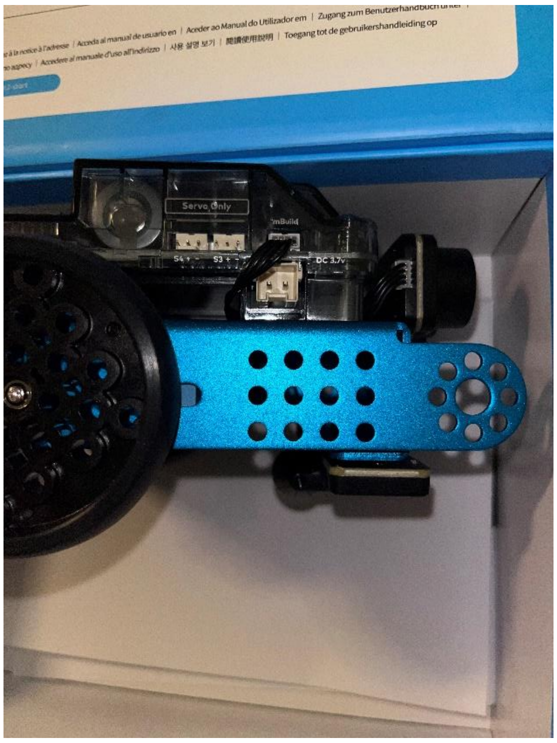
Step 18: Place the mBot Shield on the Chassis and secure it with four 25mm screws.



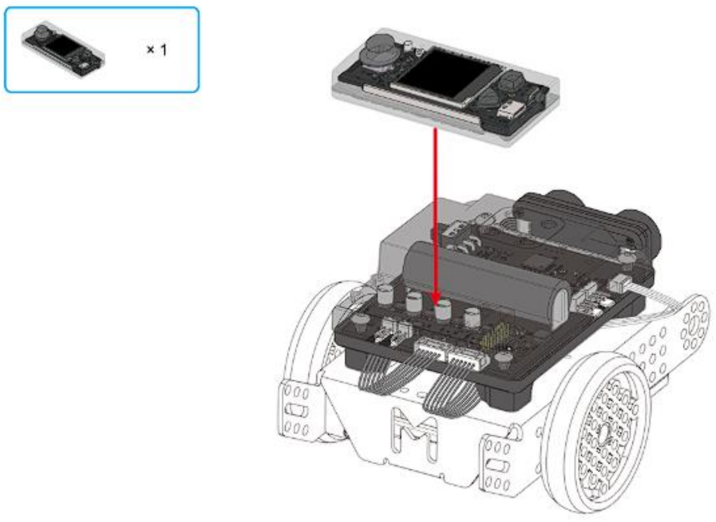
Step 19: Insert the left Motor Cable into the EM1 port and the right Motor Cable into the EM2 port.



Step 20: Insert the other end of the mBuild Cable from the Ultrasonic Sensor to the mBuild port found on the mBot Shield.



Step 21: Attach the CyberPi onto the bottom of the mBot Shield.



Operating mBot2

Requirements

- Computer
- mBot2
- USB Cable
- Line-following Track Map
- mBlock Software | Download link in the Appendix section

Set-up

- Power the mBot on using the power switch on the left side of the mShield.

- Open mBlock on your computer.

- Connect the mBot to your computer using a USB cable.

- On mBlock, add mbot2 as a device from the left most panel.

- Click on the ‘Connect’ button and choose your mBot.

- Switch the mode to ‘Upload’ if you want to upload your own program to the robot for later use (you can run this program on the robot without your computer), or to ‘Live’ mode if you want to test and edit the code live (the robot needs to be connected to your computer to run the program).

- If you have uploaded a code, you can use the CyberPi to access and run it.

Activity: Basic Movements

Code

```
import event, time, cyberpi, mbot2
```

```
2)  @event.is_press('b')
3)  def is_btn_press():
4)          mbot2.forward(50, 2)
5)          time.sleep(2)
6)          mbot2.straight(-50)
7)          time.sleep(2)
8)          mbot2.turn(-90)
9)          mbot2.forward(50, 2)
10)         time.sleep(2)
11)         mbot2.turn(90)
12)         mbot2.forward(50, 2)
```

Explanation

Line 1 imports the libraries required to control the robot

Line 2 creates an event handler that calls the following function when triggered as the 'B' button is pressed on the CyberPi

Line 3 defines the function named is_btn_pressed() which initiates the following commands

Line 4 makes the robot move forward at 50 rpm for 2 seconds

Line 5 makes the robot stop for 2 seconds

Line 6 makes the robot move backwards at 50 rpm

Line 7 makes the robot stop for 2 seconds

Line 8 makes the robot turn 90° left

Line 9 makes the robot move forward at 50 rpm for 2 seconds

Line 10 makes the robot stop for 2 seconds

Line 11 makes the robot turn 90° right

Line 12 makes the robot move forward at 50 rpm for 2 seconds

Activity: Moving in a Rectangle

Code

```
1)  import event, time, cyberpi, mbot2, mbuild


2)  @event.is_press('b')
3)  def is_btn_press():
4)        mbot2.straight(50)
5)        mbot2.turn(90)
6)        mbot2.straight(25)
```

```
7)        mbot2.turn(90)
8)        mbot2.straight(50)
9)        mbot2.turn(90)
10)       mbot2.straight(25)
11)       mbot2.turn(90)
12)       time.sleep(4)
```

Explanation

Line 1 imports the libraries required to control the robot

Line 2 creates an event handler that calls the following function when triggered as the 'B' button is pressed on the CyberPi

Line 3 defines a function called is_btn_pressed() which initiates the following commands

Line 4 makes the robot move straight forward for 50 cm

Line 5 makes the robot turn 90° to the right

Line 6 makes the robot move straight forward for 25 cm

Line 7 makes the robot turn 90° to the right

Line 8 – Line 11 repeats the whole process from Line 4

Line 12 makes the robot stop for 4 seconds

Activity: Avoiding Obstacles

Code

```
1) import event, time, cyberpi, mbuild, mbot2


2) @event.is_press('b')
3) def is_btn_press():
4)     while True:
5)         if mbuild.ultrasonic2.get(1) < 10:
6)             mbot2.turn(-97)
7)         mbot2.forward(50)
```

Explanation

Line 1 imports the libraries required to control the robot

Line 2 creates an event handler that calls the following function when triggered as the 'B' button is pressed on the CyberPi

Line 3 defines a function called is_btn_pressed() which initiates the following commands

Line 4 creates a while loop that runs infinitely

Line 5 checks if the distance between the Ultrasonic sensor and the object infront of it (if there is one) is less than 10 cm

Line 6 makes the robot turn 90° to the left if the previous statement is true

Line 7 makes the robot move forward in a straight line infinitely until an object is detected

Activity: Line-Following

Code

```
1)  import event, time, cyberpi, mbot2, mbuild


2)  motor_power = 0
3)  turn_speed = 0
4)  left_motor_power = 0
5)  right_motor_power = 0


6)  @event.is_press('a')
7)  def is_btn_press():
8)      global motor_power, turn_speed, left_motor_power, right_motor_power
9)      mbot2.drive_power(0, 0)


10) @event.is_press('b')
11) def is_btn_press1():
12)     global motor_power, turn_speed, left_motor_power, right_motor_power
13)     while True:
14)         motor_power = 30
15)         turn_speed = 0.8
16)         while True:
17)             left_motor_power = (motor_power - turn_speed *
                mbuild.quad_rgb_sensor.get_offset_track(1))
18)             right_motor_power = -1 * ((motor_power + turn_speed *
                mbuild.quad_rgb_sensor.get_offset_track(1)))
19)             mbot2.drive_power(left_motor_power, right_motor_power)
```

Explanation

Line 1 imports the libraries required to control the robot

Line 2 – Line 5 defines 4 variables and are all assigned the value 0

Line 6 creates an event handler that calls the is_btn_press function when triggered as the ‘A’ button is pressed on the CyberPi

Line 7 defines the is_btn_press() function

Line 8 accesses the global variables

Line 9 sets the power of both the motors to 0, which stops the robot

Line 10 creates an event handler that calls the is_btn_press1 function when triggered as the ‘B’ button is pressed on the CyberPi

Line 11 defines the is_btn_press1 function

Line 12 accesses the global variables

Line 13 creates a while loop that runs infinitely

Line 14 assigns the value 30 to the motor_power variable

Line 15 assigns the value 0.8 to the turn_speed variable

Line 16 creates another infinite while loop inside the first one

Line 17 reads the values of the track offset from the RGB sensor 1 and adjusts the left motor power by reducing it when the robot is off-center

Line 18 does the same for the right motor and the -1 is there as the motor’s orientation is flipped, so it needs to spin in reverse

Line 19 continuously adjusts the power of both the motors which makes it follow the line by making the necessary turns to stay on track

Activity: Reacting to Colours

Code

```
1)  import event, time, cyberpi, mbuild, mbot2


2)  motor_power = 0
3)  turn_speed = 0
4)  left_motor_power = 0
5)  right_motor_power = 0
6)  offset = 0


7)  def followLine():
8)      offset = mbuild.quad_rgb_sensor.get_offset_track(1)
9)      left_motor_power = (motor_power - turn_speed * offset)
10)     right_motor_power = -1 * ((motor_power + turn_speed * offset))
11)     mbot2.drive_power(left_motor_power, right_motor_power)


12) @event.is_press('b')
13) def is_btn_press():
14)         global motor_power, turn_speed, left_motor_power, right_motor_power, offset
15)         motor_power = 40
16)         turn_speed = 0.8
17)         while True:
18)                 if (mbuild.quad_rgb_sensor.is_color("red","R1",1)):
```

```
19)                 mbot2.EM_stop("ALL")
20)                 time.sleep(3)
21)                 motor_power = 40
22)                 followLine()


23)             elif (mbuild.quad_rgb_sensor.is_color("yellow","R1",1)):
24)                 motor_power = 10
25)                 followLine()


26)             elif (mbuild.quad_rgb_sensor.is_color("green","R1",1)):
27)                 motor_power = 40
28)                 followLine()


29)             followLine()
```

Explanation

Line 1 imports the libraries required to control the robot

Line 2 – Line 6 defines 5 variables and are all assigned the value 0

Line 7 defines the followLine() function

Line 8 assigns the offset value to the variable

Line 9 adjusts the left motor power according to the offset

Line 10 adjusts the right motor power according to the offset

Line 11 uses the motor powers as parameters to steer the robot

Line 12 creates an event handler that calls the is_btn_press function when triggered as the 'B' button is pressed on the CyberPi

Line 13 defines the is_btn_press() function

Line 14 accesses the global variables

Line 15 assigns the value 40 to the motor_power variable

Line 16 assigns the value 0.8 to the turn_speed variable

Line 17 starts an infinite while loop

Line 18 checks if the R1 RGB sensor detects the colour red

Line 19 stops the motors if the colour detected is red

Line 20 pauses the robot for 3 seconds

Line 21 sets the motor_power to 40

Line 22 calls the followLine() function to resume robot movement

Line 23 checks if the R1 RGB sensor detects the colour yellow

Line 24 lowers the motor_power value to 10

Line 25 calls the followLine() function to resume robot movement

Line 26 checks if the R1 RGB sensor detects the colour green

Line 27 sets the motor_power to 40

Line 28 calls the followLine() function to resume robot movement

Line 29 calls the followLine() function outside of the IF loop to keep the robot moving on the track infinitely

Appendix

Links

Assembly Guide: https://support.makeblock.com/hc/en-us/articles/1500006253942-Assemble-mBot-Neo-mBot2

Programming Guides: https://support.makeblock.com/hc/en-us/sections/1500001036301-mBot-Neo-mBot2

mBlock Download Link: https://mblock.cc/pages/downloads