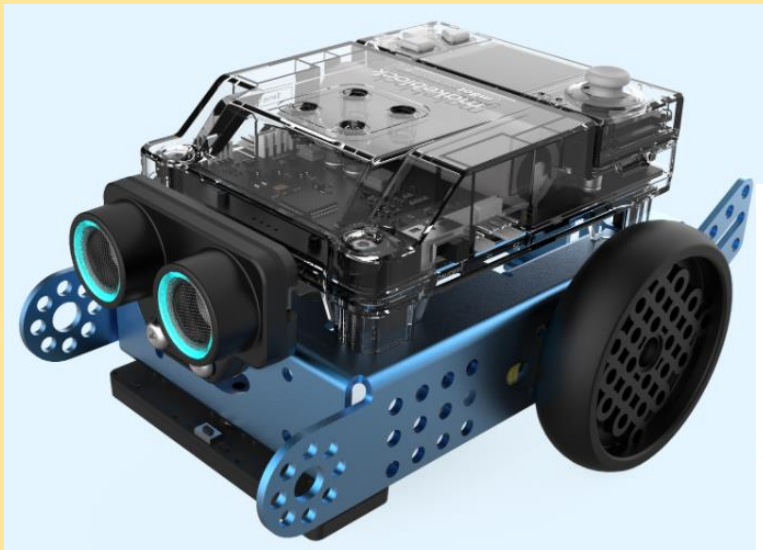


Digital Technology

MakeBlock mBot2 / CyberPi Python Code



Version 2.2
Feb 2022

Barry Butler
bbutl58@eq.edu.au

Content and Challenges

Section	Content
A	The mBot2 Vehicle
B	Introduction and Setup
C	Our First Program – Hello
D	Push the Buttons
E	Run the Motors
F	Avoid or Seek
G	Detect and Follow a Line
H	SumoBot
I	Connect Servos, Sensors and Motors
Appendix 1	CyberPi Extras

Documentation

<https://www.yuque.com/makeblock-help-center-en/mcode/mblock-python>

CyberPi Python <https://www.yuque.com/makeblock-help-center-en/mcode/cyberpi-api>
(including Pocket Shield, mBot2 Shield and mBuild Modules)

Firmware Update

To update the CyberPi firmware:

1. Open the online ide at <https://ide.mblock.cc/#/>
2. Click on **Devices** and add the CyberPi device to the list, if it is not there already
3. Click **Connect** and connect the CyberPi (download and install the device driver, if asked)
4. Click on **Settings** and select Firmware Update

A. The mBot2 Vehicle

Documentation

MBot2 Introduction

<https://education.makeblock.com/help/cyberpi-series/cyberpi-series-cyberpi-series-packages-and-extensions/mbot2-introduction/>

Operational Guide

<https://education.makeblock.com/help/cyberpi-series/cyberpi-series-cyberpi-series-packages-and-extensions/mbot2-operational-guide/>

Python Reference

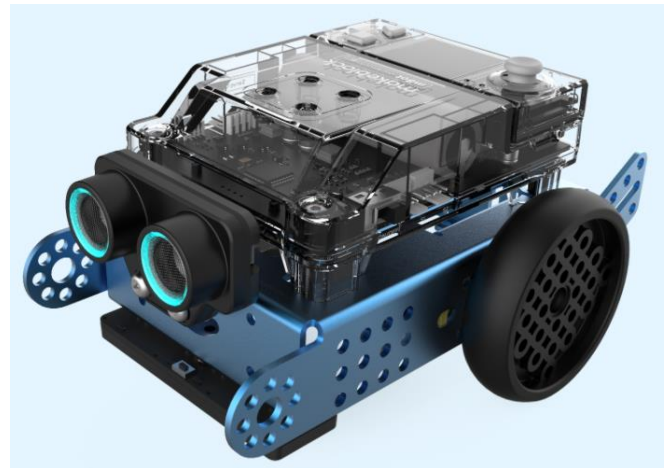
<https://www.yuque.com/makeblock-help-center-en/mcode/cyberpi-api-shields#9eo89>

mBuild Modules (Ultrasonic Sensor 2, Quad RGB Sensor)

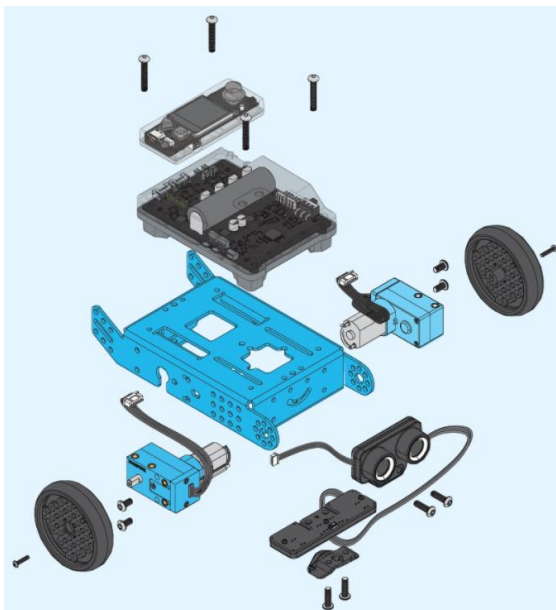
<https://www.yuque.com/makeblock-help-center-en/mcode/cyberpi-api-mbuild>

or

<https://education.makeblock.com/help/mblock-python/mblock-python-editor-python-api-documentation-for-devices/mblock-python-editor-python-api-documentation-for-cyberpi/mblock-python-editor-apis-for-mbuild-modules/>

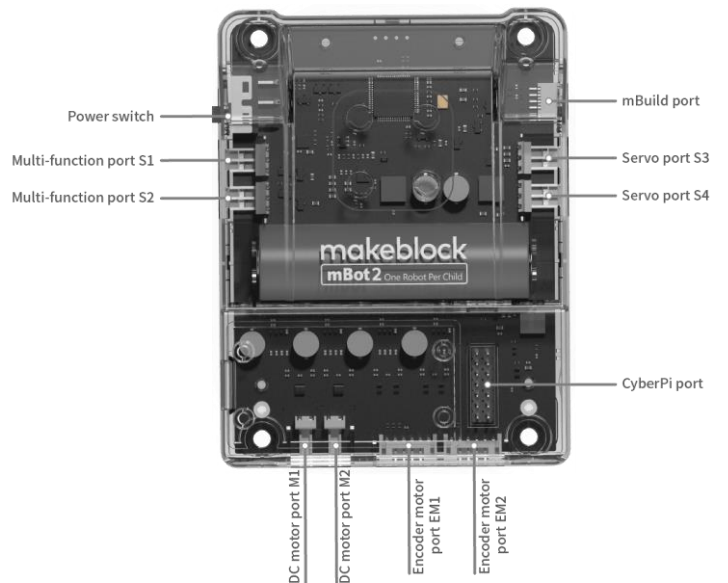


The Build



The Connections

(Ultrasonic into the mBuild port, motors to EM1/EM2)



The Power Switch must be turned on before the you can upload code

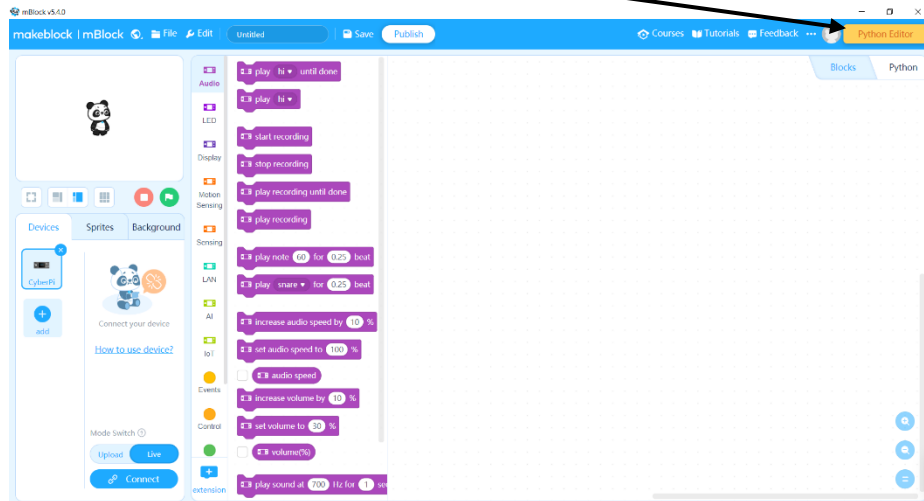
B. Introduction and Setup

Download and Install the Software

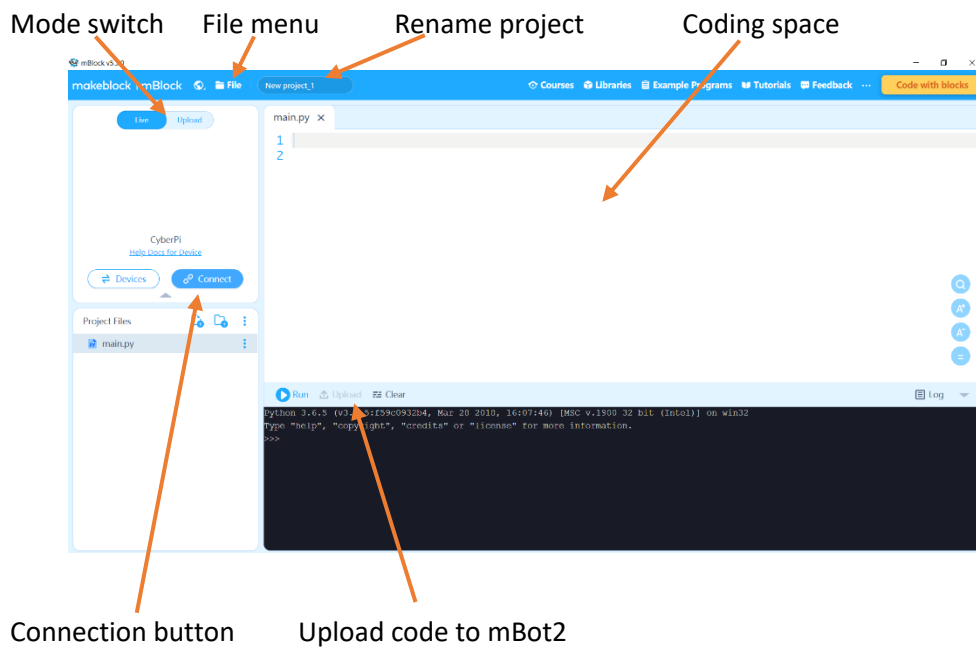
Download and install the mBlock Windows or Mac software from <https://mblock.makeblock.com/en-us/download/>

(The PC software seems to be more stable than the web version located at <https://python.mblock.cc/>)

1. Run the software and select the Python Editor.



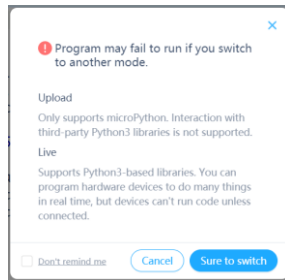
The Python Editor program will open. The block editor will stay open, but you can close it at any time.



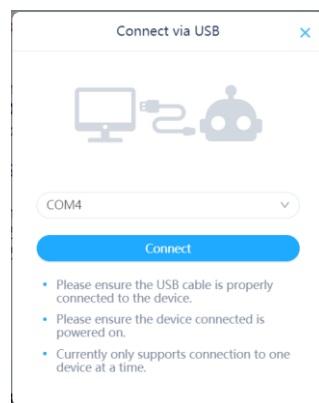
2. TURN ON THE MBOT2 USING THE SWITCH ON THE SIDE

The lights on both the ultrasonic sensor and the line follower sensor should turn on. If they are don't, the wiring is incorrect or unplugged, and needs to be fixed.

3. Select **Upload** mode. If a message appears, tick “Don't remind me” and then click *Sure to switch*.



4. Plug the mBot2 into a USB port and click the **Connect** button .



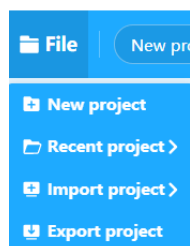
Find Your Port

You can easily find your device by first leaving your mBot **unplugged**. Click *Connect* and look at the list of USB ports.

Close the connect window, then **plug in** your mBot2. Click *Connect* again and look for the port that has just been added.

Select your USB port from the list and click **Connect**.

5. Click on the **File** menu and select **New Project**.



6. Start coding

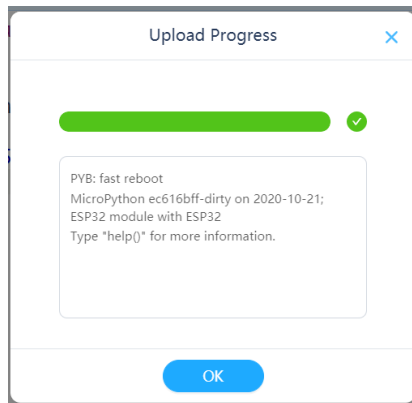
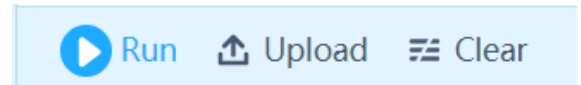
C. Our First Program – Hello

Our first program will write 'hello' on the console, say it on the audio speaker and turn all LED's to green for 2 seconds.

```
import cyberpi as cpi
import time

cpi.console.print("hello")
cpi.audio.play('hello')
cpi.led.on(0,255,0)           #red, green, blue values from 0 to 255
time.sleep(2)                #time delay in seconds
cpi.led.off()
cpi.console.clear()
```

Click the **Upload** button to send your code to the mBot2.



The code will start executing immediately it is uploaded.

Unsuccessful Upload

If the upload is unsuccessful check three things:

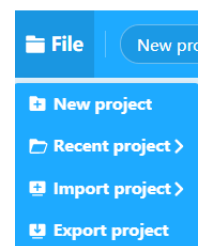
1. The **mBot2 is turned on** (the power switch on the left side).
2. The cable is plugged in and a **connection established** (see section B4).

Save the Project and Upload to the CyberPi

Save the project to your computer by clicking on the **File** menu and choosing **Export project**.

It is a good idea to create a folder to contain all your projects.

Make sure you type in a descriptive name for your file.



Coding Errors and Feedback from the CyberPi

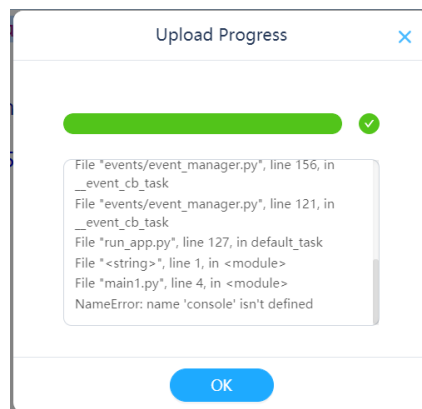
When you write code, errors show up with an explanation mark symbol.

In the example shown here there are a couple of errors:

- the statement in line 1 is ***import cyberpi as cpu*** rather than ***import cyberpi as cpi*** causing errors in all the other lines.
- line 4 is missing ***cpi***.

```
1 import cyberpi as cpu
2 import time
3
4 console.print("hello")
5 cpi.audio.play('hello')
6 cpi.led.on(0,255,0)
7 time.sleep(2)
8 cpi.led.off()
9
```

If you were to upload this code it would not run and the upload window will show you the first error. Scroll to the bottom of the text to see the error message.

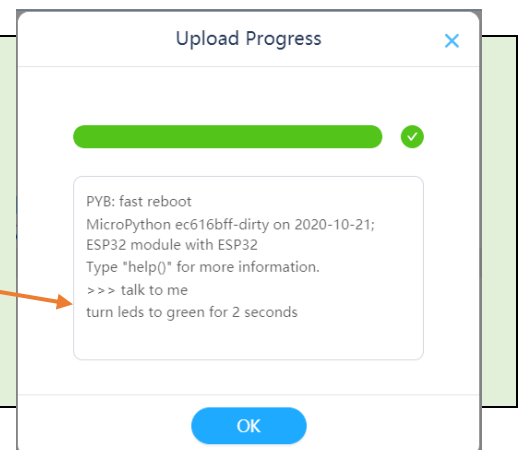


Program Feedback

You can also give yourself feedback in the code you write by using the **print()** function. This is *different* to the `cpi.console.print()` function. Try this:

```
import cyberpi as cpi
import time

cpi.console.print("hello")
print('talk to me')
cpi.audio.play('hello')
print('turn leds to green for 2 seconds')
cpi.led.on(0,255,0)
time.sleep(2)
cpi.led.off()
cpi.console.clear()
```

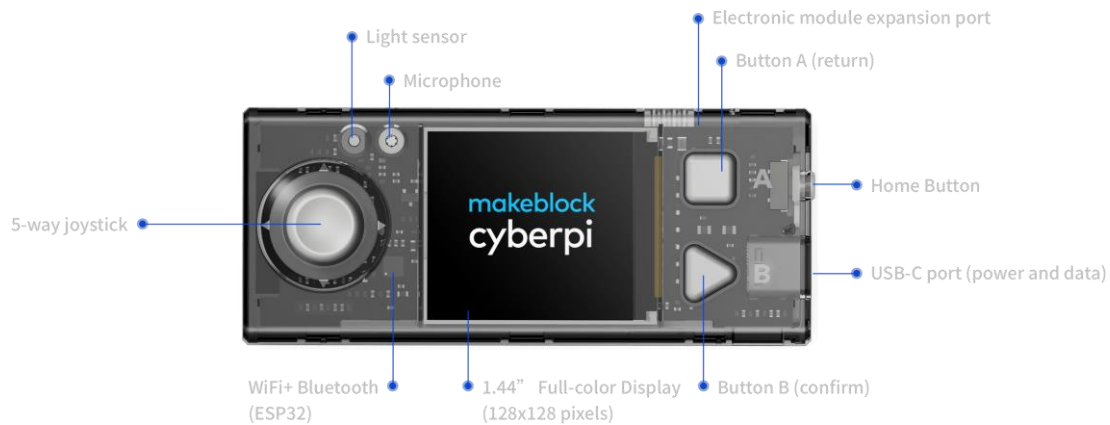


Comments and turning on/off code statements

Put a **#** in front of any line to create comments or to turn code statements into comments so they are not executed.

D. Buttons

The mBot2 is controlled by a module called **cyberpi**. This has a joystick, a home button and two push buttons (A and B). We can use the joystick and buttons in our code. It also has a light sensor and microphone that we can use.



Instead of the code running automatically when it is uploaded, let's turn on the display when we press button A. To do this we use a **while** loop, that does nothing but turn on the red lights and wait for the button to be pressed.

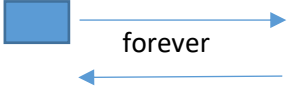
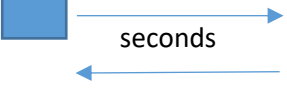
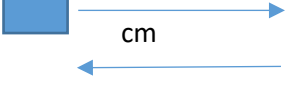
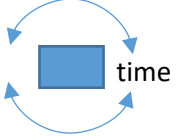
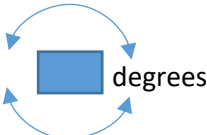
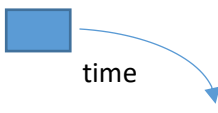
```
import cyberpi as cpi
import time

while not cpi.controller.is_press('a'):    #while button A is not pressed
    cpi.led.on(255,0,0)
    cpi.led.on(0,255,0)

cpi.console.print("hello")
cpi.audio.play('hello')
cpi.led.on(0,255,0)
time.sleep(2)
cpi.led.off()
cpi.console.clear()
```


E. Run the Motors

There are a number of ways we may want to move the mBot2. Forward motor speeds are between 0 and 100. Backward motor speeds are between 0 and -100. Movement still occurs at speeds close to zero.

Movement		Commands
Forward or backward forever. (Should only be used when the ultrasonic sensor or colour sensors are used to control when the motors should stop)		<pre> cpi.mbot2.forward(speed = 50) cpi.mbot2.backward(speed = 50) cpi.mbot2.forward(speed = -50) cpi.mbot2.EM_stop(port = "all") </pre>
Forward or backward for a length of time		<pre> cpi.mbot2.forward(speed = 50, run_time = 1) cpi.mbot2.backward(speed = 50, run_time = 1) cpi.mbot2.forward(speed = -50, run_time = 1) </pre>
Forward or backward for a fixed distance		<pre> cpi.mbot2.straight(40, speed = 50) cpi.mbot2.straight(-40, speed = 50) </pre>
Turn on the spot for a length of time (wheels turning in different directions)		<pre> cpi.mbot2.turn_left(speed = 50, run_time = 1) cpi.mbot2.turn_right(speed = 50, run_time = 1) </pre>
Turn for a number of degrees of heading		<pre> cpi.mbot2.turn(90, speed = 50) </pre>
Gradual turn for a length of time (wheels turning in the same direction or one wheel stopped)		<pre> cpi.mbot2.drive_power(60, -40) #left +, right - time.sleep(2) cpi.mbot2.EM_stop(port = "all") </pre>
Stop motors		<pre> cpi.mbot2.EM_stop(port = "all") </pre>

Code Templates

There are two basic code templates we use when running motors. In both cases, we use button A to turn on the mBot2 to start the actions.

Separating code into sections makes it much easier to understand the code and make changes to it. Later, we will add more sections as we require them.

1. **Single Actions.** Use this when the mBot2 actions should only occur once.

```
#IMPORTS-----
import cyberpi as cpi
import time

#WAIT TO START-----
cpi.console.println('Press A')
while not cpi.controller.is_press('a'):
    cpi.led.on(255,0,0)
    cpi.led.on(0,255,0)

#ROBOT ACTIONS-----
cpi.mbot2.forward(speed = 50, run_time = 2)      #Example commands.
cpi.mbot2.backward(speed = 50, run_time = 2)     #Replace with your own!

cpi.led.off()
```

If we have actions that are repeated, we can use a **for loop**. For example, to move in a square:

```
#IMPORTS-----
import cyberpi as cpi
import time

#WAIT TO START-----
cpi.console.println('Press A')
while not cpi.controller.is_press('a'):
    cpi.led.on(255,0,0)
    cpi.led.on(0,255,0)

#ROBOT ACTIONS-----
for i in range(4):
    cpi.mbot2.straight(40, speed = 50)           #cm
    cpi.mbot2.turn(90, speed = 50)              #degrees

cpi.led.off()
```

CHALLENGES

1. Place one or more large objects on the floor. Navigate the mBot2 through and/or around them.
2. One of the RoboRAVE competitions is AMAZE-ing. It consists of a series of boards that make up a maze. You do not know the shape of the maze until the competition. The person who keeps the robot on the boards and has the fastest time wins.

2. **Forever Actions.** This code has a **while True** loop that repeats the actions forever – or until you press the **home button** next to the USB connection.

```
#IMPORTS-----
import cyberpi as cpi
import time

#WAIT TO START-----
cpi.console.println('Press A')
while not cpi.controller.is_press('a'):
    cpi.led.on(255,0,0)
    cpi.led.on(0,255,0)

#MAIN LOOP-----
while True:
    cpi.mbot2.forward(speed = 50, run_time = 2)    #Example commands.
    cpi.mbot2.backward(speed = 50, run_time = 2)   #Replace with your own!
```

This code is mainly used in conjunction with the joystick and buttons, or the ultrasonic and line follower sensors, where the mBot2 will respond to changes in sensor values.

CHALLENGES

3. Place two small objects on the floor at least 1m apart. Drive around these multiple times in a figure of 8. When you turn use the led's to indicate your turns.
4. Place a large object on the floor and turn around the object 3 times in a large, smooth circle. *(Use the `cpi.mbot2.drive_power()` function)*

F. Avoid or Seek

The **Ultrasonic Sensor** is used to measure the distance between the mBot2 and anything in front of it (up to about 200cm). It can be used to avoid obstacles or seek out an object and move toward it.

The minimum distance detected is 4cm. Smaller distances give a reading of 300.

Test your Ultrasonic Sensor with this code. Putting all the sensor reading code into a function unclutters the main loop.

```
#IMPORTS-----
import cyberpi as cpi
import time

#GLOBAL VARIABLES-----
distance = 300

#FUNCTIONS-----
def get_all_values(output=True):
    global distance

    distance = cpi.ultrasonic2.get(index=1)
    if output:
        cpi.console.println( str(distance) )
        time.sleep(0.1)

#WAIT TO START-----
cpi.console.println('Press A')
while not cpi.controller.is_press('a'):
    cpi.led.on(255,0,0)
cpi.led.on(0,255,0)

#MAIN LOOP-----
while True:
    get_all_values(output=True)
```

Obstacle Avoidance

```
#MAIN LOOP-----
while True:
    get_all_values(output=False)

    if distance < 10:
        cpi.mbot2.EM_stop(port = "all")
        cpi.mbot2.straight(-5, speed = 50)
        cpi.mbot2.turn(135, speed = 50)
    else:
        cpi.mbot2.forward(speed = 50)

#collision test
#stop
#move back 5cm
#turn 135 degrees
#forward
```

Slow Down when Close to a Collision

```
#MAIN LOOP-----
while True:
    get_all_values(output=False)

    if distance < 10:
        cpi.mbot2.EM_stop(port = "all")
        cpi.mbot2.straight(-5, speed = 50)
        cpi.mbot2.turn(135, speed = 50)
        #collision test
        #stop
        #move back 5cm
        #turn 135 degrees

    elif distance < 30:
        new_speed = round(50 * (distance - 10)/20)
        cpi.mbot2.forward(speed = new_speed)
        #ratio of speed required
        #forward at reduced speed

    else:
        cpi.mbot2.forward(speed = 50)
        #forward
```

Seek Objects and Move Toward Them

Rotate to detect an object closer than 80cm, then move toward the object.

```
#MAIN LOOP-----
while True:
    get_all_values(output=False)

    if distance > 80:
        cpi.mbot2.turn_left(speed = 50)
        #rotate to locate

    else:
        cpi.mbot2.EM_stop(port = "all")
        cpi.mbot2.forward(speed = 100)
        #object detected
        #stop
        #forward full speed
```

CHALLENGES

5. Place 4 objects at the corners of a square. Find one of them and stop before you hit it. Turn and find the next object, until you have found all four.
6. Find your way autonomously through a simple maze (sides are 10cm high)

G. Detect and Follow a Line

The **Quad RGB Sensor** (color sensor) enables us to detect and follow lines, and detect colours and respond to the colours in different ways.

Test the Sensor using this code, by passing the mBot2 over a black line on a white background.

```
import cyberpi as cpi
import time

#GLOBAL VARIABLES-----
distance = 300
L1 = 0
L2 = 0
R1 = 0
R2 = 0
any_line = 0

#FUNCTIONS-----
def get_all_values(output=True, black_line=True):
    global distance, L1, L2, R1, R2, any_line

    distance = cpi.ultrasonic2.get(index=1)
    L2 = cpi.quad_rgb_sensor.get_gray('l2', index = 1)
    L1 = cpi.quad_rgb_sensor.get_gray('l1', index = 1)
    R1 = cpi.quad_rgb_sensor.get_gray('r1', index = 1)
    R2 = cpi.quad_rgb_sensor.get_gray('r2', index = 1)
    if black_line:
        any_line = (L2 < 50) or (L1 < 50) or (R1 < 50) or (R2 < 50)
    else:
        any_line = (L2 > 50) or (L1 > 50) or (R1 > 50) or (R2 > 50)

    if output:
        #cpi.console.println(str(distance) )
        cpi.console.println(str(L2)+' '+str(L1)+' '+str(R1)+' '+str(R2) )

#WAIT TO START-----
cpi.console.println('Press A')
while not cpi.controller.is_press('a'):
    cpi.led.on(255,0,0)
cpi.led.on(0,255,0)

#MAIN LOOP-----
while True:
    get_all_values(output=True, black_line=True)
    time.sleep(0.1)
```

We can use the color sensor values to test whether the color sensor is on or off a black line.

- On a line will give a low reflectance value or off a line will give a high value.
- Assume for a start that if the reflected light value is less than 50% if we are on or near a black line.
- Place the mBot2 on the middle of the black line
- If both sensors L1 and R1 are on black – go straight ahead
- If only sensor L1 is on black – turn to the left
- If only sensor R1 is on black – turn to the right

First, test the code below without the motors driving. Then take off the comment # and try with the motors running.

```
#MAIN LOOP-----  
cpi.mbot2.drive_power(50, -50)           #forward  
  
while True:  
    get_all_values(output=False, black_line=True)  
  
    if L1 < 50 and R1 < 50:  
        cpi.mbot2.drive_power(20, -20)   #straight ahead  
        cpi.led.on(255,0,0,id=2)  
        cpi.led.on(255,0,0,id=4)  
    elif L1 < 50:  
        cpi.mbot2.drive_power(5, -20)    #turn left  
        cpi.led.on(255,0,0,id=2)  
    elif R1 < 50:  
        cpi.mbot2.drive_power(20, -5)    #turn right  
        cpi.led.on(255,0,0,id=4)  
    else:  
        cpi.led.on(0,255,0)
```

To follow the line faster, you might need the change:

- The power to the left and right wheels
- The difference in power between the left and right wheels
- How you interpret the percentage color sensor values
- Use the L2 and R2 sensors as well

CHALLENGES

5. Oval Race. Follow an oval line from start to finish. Time the run. The robot that does the quickest time wins.
6. RoboRAVE Line Follower Race. Be the fastest robot to get from home to the box.

H. SumoBot

SumoBots use the ultrasonic sensor to seek and destroy another robot vehicle in the Sumo ring, while using the color sensor to sense the white border and avoid falling off the edge.

H1. Basic Sumo Code

The basic actions of a SumoBot are:

- A three second wait before doing anything
- Move forward from the edge 20cm
- Rotate until the ultrasonic sensor locates the other vehicle (less than 80cm away)
- Drive full speed toward the other vehicle
- If the white edge is detected (high reflectance value) then stop, back up and rotate to locate the other vehicle

```
#MAIN LOOP-----
found = False
cpi.mbot2.straight(20, speed = 40)

while True:
    get_all_values(output=False, black_line=False)

    if any_line:                                #white line detected
        found = False
        cpi.led.on(0,255,0)
        cpi.mbot2.EM_stop(port = "all")         #stop, back and rotate
        cpi.mbot2.straight(-10, speed = 40)

    if distance < 80 or found:                  #other robot detected
        found = True
        cpi.led.on(0,0,255)
        cpi.mbot2.forward(speed = 40)          #forward full speed
    else:
        cpi.mbot2.turn_left(speed = 5)         #rotate to locate
```

H2. Enhancements

- Don't waste time moving forward at the start before starting to find the other vehicle
- Only scan left and right up to 90 degrees the first time
- Stop every 10 degrees when scanning to make sure scan detects vehicle (moving too fast doesn't work)
- Use movement sensor to detect a collision or the bot lifted off the ground (pitch or roll) and respond to that (see Appendix 1)
- If motion is stopped for x seconds, use a series of rapid wheel movements (e.g. back and forth) to try and get free
- Use a different strategy:
 - Follow white line around the outside (use L2 or R2)
 - Drive to a random place
 - Drive forward until white line and turn and randomly go somewhere else until white line
- Use more than one ultrasonic sensor at different angles

I. Connect Servos, Sensors and Motors

Servos

Up to 4 servos can be plugged in the servo ports on the right-hand side (S3 and S4), or the general IO ports on the left (S1 and S2).

```
import cyberpi as cpi
import time

while True:
    cpi.mbot2.servo_set(90, 'S1')
    time.sleep(1)
    cpi.mbot2.servo_set(140, 'S1')
    time.sleep(2)
    cpi.mbot2.servo_set(40, 'S1')
    time.sleep(2)
```

Read Analog Sensors

Read analog sensors (such as potentiometers or soil moisture sensors) using ports S1 and S2

```
cpi.mbot2.read_analog(port)          #returns 0 - 5V
```

Read and Write Digital Sensors

```
cpi.mbot2.write_digital(val, port)   #val = True, False, 0, 1
cpi.mbot2.read_digital(port)         #returns True, False
```

Run DC motors

Additional motors can be run from the M1 and M2 ports.

```
cpi.mbot2.motor_set(power, port)     #power is -100 to 100
cpi.mbot2.motor_stop(port)

cpi.mbot2.motor_drive(power1, power2) #set the power to M1 and M2
```

More Loops

e.g. `for i in range(4,6): cpi.led.on(0,255,0, id = i)`

```
import cyberpi as cpi
import time

while True:
    if cpi.controller.is_press('a'):
        for i in range(1,6,2): cpi.led.on(0,255,0, id = i)    #or id = 1, value 1-5
        cpi.console.print('green\n')
    elif cpi.controller.is_press('b'):
        cpi.led.off()
        for i in range(2,5,2): cpi.led.on(0,0,255, id = i)
        cpi.console.print('blue\n')
    time.sleep(0.1)
```

Appendix 1 CyberPi Extras

Ultrasonic, slider (potentiometer) and multi-touch

```
import cyberpi as cpi
import time

while True:
    distance = cpi.ultrasonic2.get(index=1)
    pot = cpi.slider.get()
    touch = cpi.multi_touch.is_touch(ch = 1)    #1-8 or ch = "any"
    print(distance, pot, touch)
    time.sleep(0.1)
```

Light sensor

```
light = cpi.get_bri()
```

Sound sensor

```
volume = cpi.get_loudness(mode = "maximum")
```

Audio Commands

```
cpi.audio.play_tone(freq, t)
cpi.audio.add_vol(val)          #-100 - 100
```

Accelerometer/Gyro Commands

```
forward = cpi.is_tiltforward()
backward = cpi.is_tiltback()
left = cpi.is_tiltleft()
right = cpi.is_tiltright()
```

```
cpi.is_shake()
cpi.get_shakeval()           #0-100

cpi.get_pitch()             #pitch angle
cpi.get_roll()              #roll angle

cpi.get_yaw()                #yaw angle
cpi.reset_yaw()
```