CG1111A Engineering Principles & Practice I

Introduction to mBot

Objectives:

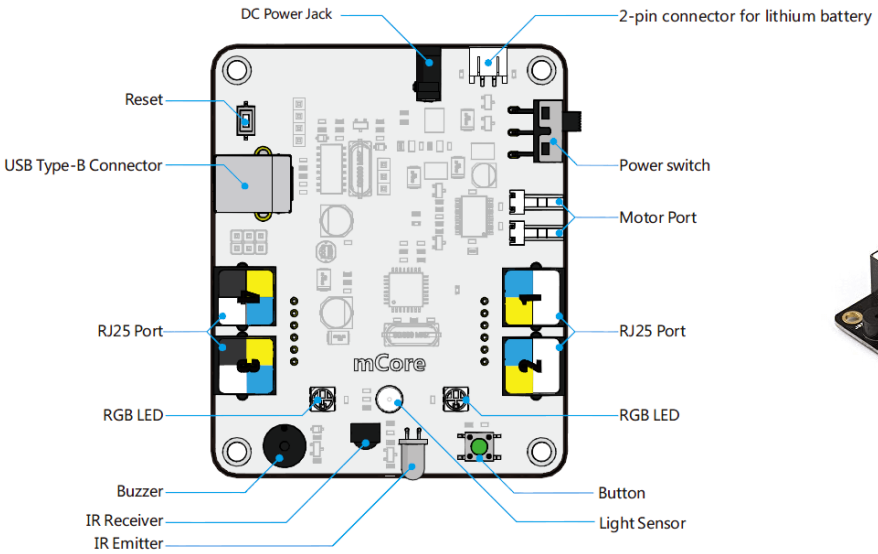
- To get familiarized with the mBot hardware
- To try out some basic functions of the mBot

The mBot Hardware:

- The mBot is a flexible robotic platform that can be customized for a variety of applications. Its basic configuration is a two-wheeled robot with two sensors, namely, a line sensor, and an ultrasonic sensor, as shown in Figure 1 below.

Figure 1: Basic configuration of mBot.

- The mBot's main control board is called the "mCore". Its layout is shown in Figure 2 below.



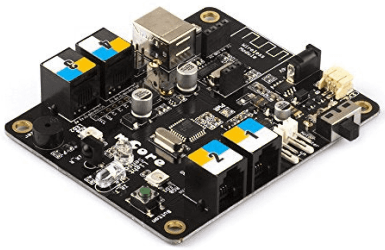


Figure 2: mCore – the mBot's main control board.

- As seen in Figure 2, the mCore has four RJ25 ports (for connecting cables that look like telephone cables). These ports can be used for connecting to other mBot accessories/sensors. We will be using some of these ports for connecting to your other self-built sensors in the project.

- On the mCore, there are also some useful components such as the following:
    - Two RGB LEDs
    - An infrared (IR) emitter and an infrared receiver
    - A light sensor
    - A buzzer

- Figure 3 shows the basic wiring of the mBot. Note that there is **no need** for you to connect the **AA battery holder**, as we have provided you with the Lithium Polymer rechargeable battery. The battery will be recharged whenever you connect your mBot to a computer or a USB charger via the given USB cable.

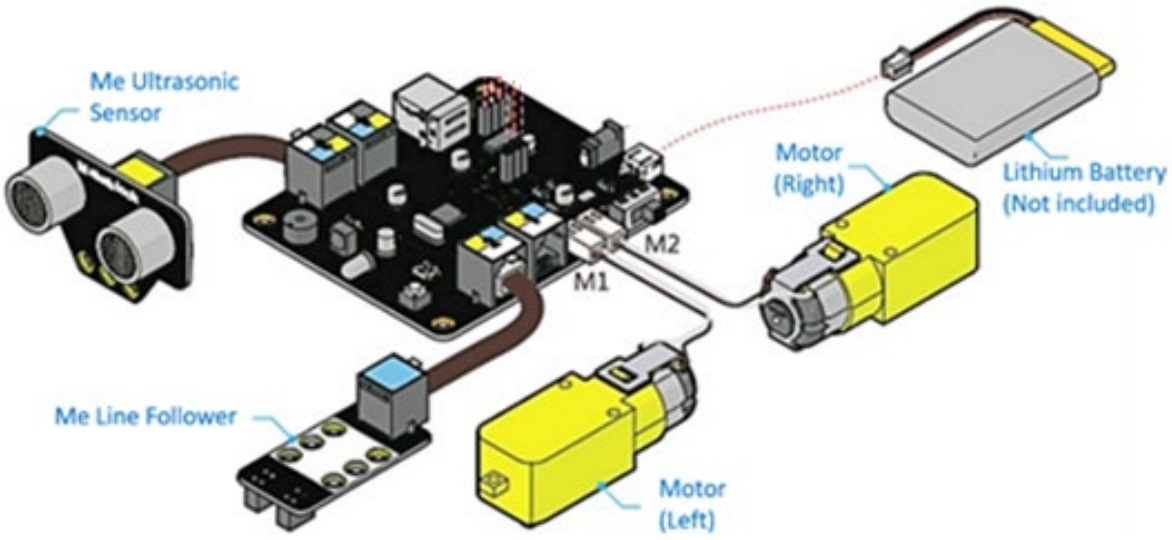


Figure 3: Basic wiring of the mBot.

Programming the mBot with the Arduino IDE:

The mBot can be programmed using the Arduino Integrated Development Environment (IDE). This is the same IDE that is used to programme Arduino Uno. Just for your interest, the mBot actually uses the same microcontroller (ATMega328) as the Arduino Uno.

Before continuing with the rest of this activity, please ensure that you have already installed the **Arduino IDE, mBot Library, and mBot Driver** according to the other document "Installing Arduino IDE, mBot Library & Driver.pdf".

Launch the Arduino IDE. You will see the following interface:

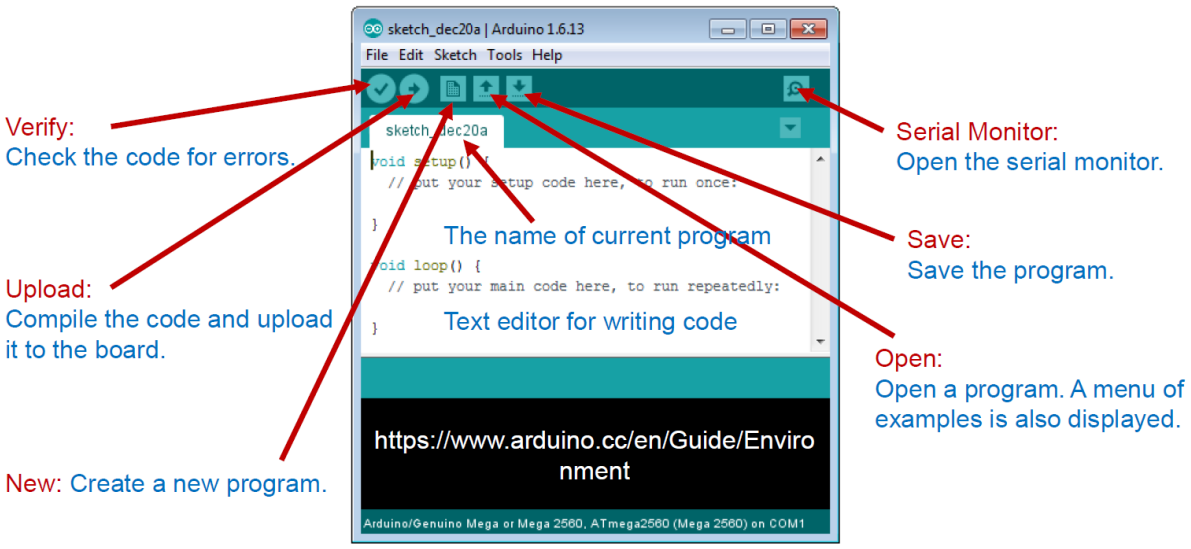


Figure 4: Arduino IDE's interface.

Verify that the board selected in the Arduino IDE is currently "Arduino Uno" as shown in Figure 5:

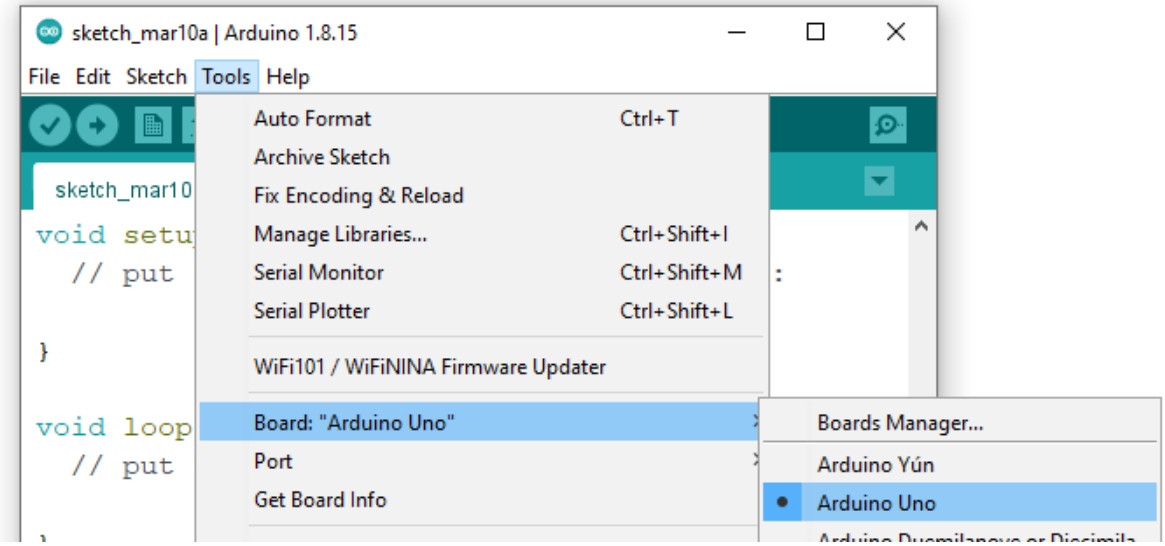


Figure 5: Ensuring that the board selected is "Arduino Uno".

Next, select from the menu: “Tools” -> “Port”, and take note of the list of serial COM ports that are shown (see Figure 6 below). Your list of COM ports may be different from Figure 6, but it does not matter.

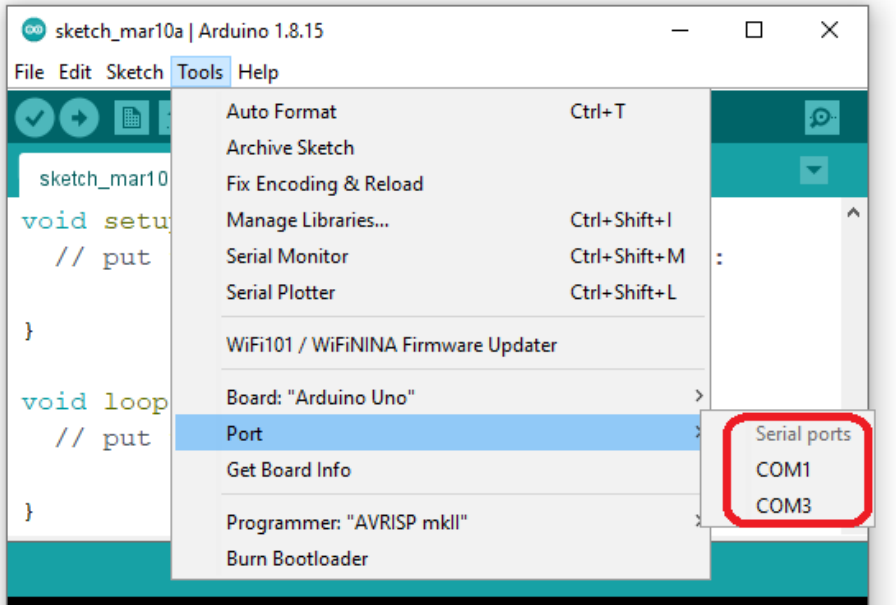


Figure 6: Check the list of COM ports currently active on your computer.

Now, connect the mBot to your computer’s USB port using the USB cable provided in the box. **Turn on the power switch** of the mBot. Check the list of serial COM ports again to observe what is the new COM port that appears (see Figure 7). Make sure that this **new** COM Port is selected.

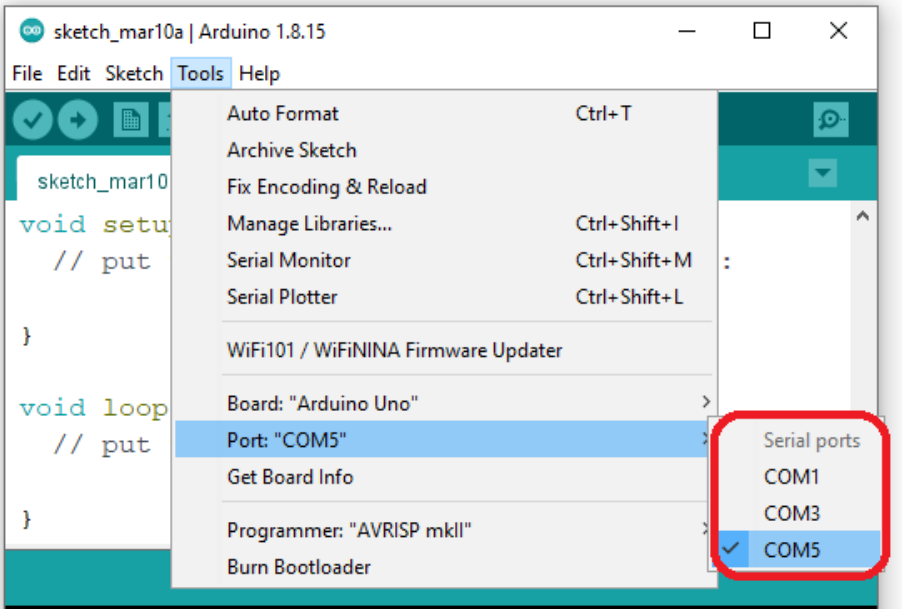


Figure 7: Ensure that the newly appeared COM port is selected.

Copy and paste the following code into your Arduino IDE, and save the file at a convenient location on your computer. Note that programs written using the Arduino IDE are called sketches, and have the file extension ".ino".

```
#include "MeMCore.h"
#define TURNING_TIME_MS 330 // The time duration (ms) for turning

MeDCMotor leftMotor(M1); // assigning leftMotor to port M1
MeDCMotor rightMotor(M2); // assigning RightMotor to port M2

uint8_t motorSpeed = 255;
// Setting motor speed to an integer between 1 and 255
// The larger the number, the faster the speed

void setup()
{
  // Any setup code here runs only once:
  delay(10000); // Do nothing for 10000 ms = 10 seconds
}

void loop()
{
  // The main code here will run repeatedly (i.e., looping):

  // Going forward:
  leftMotor.run(-motorSpeed); // Negative: wheel turns anti-clockwise
  rightMotor.run(motorSpeed); // Positive: wheel turns clockwise
  delay(1000); // Keep going straight for 1000 ms

  leftMotor.stop(); // Stop left motor
  rightMotor.stop(); // Stop right motor
  delay(1000); // Stop for 1000 ms

  // Turning left (on the spot):
  leftMotor.run(motorSpeed); // Positive: wheel turns clockwise
  rightMotor.run(motorSpeed); // Positive: wheel turns clockwise
  delay(TURNING_TIME_MS); // Keep turning left for this time duration

  leftMotor.stop(); // Stop left motor
  rightMotor.stop(); // Stop right motor
  delay(1000); // Stop for 1000 ms
}
```

Figure 8: Sample Arduino program code for testing the mBot's wheel motors.

Click the "Verify" button in the Arduino IDE to compile the code (see Figure 4). If all goes well, the bottom green bar will show "Done compiling". Next, click the "Upload" button to upload the program into the mBot. The bottom green bar shows "Done uploading" once the upload completes. **Important: you have 10 seconds to switch off the mBot's power switch after the uploading, before the mBot starts to move.** Unplug the USB cable from the mBot, and shift the mBot to a suitable flat surface. Turn on the power switch again, and the mBot will start moving after 10 seconds.

What does the program do? You will observe that the mBot does the following repeatedly:

- Moves forward for 1 s
- Pauses for 1 s
- Turns left for a preset time of "TURNING_TIME_MS" (initialized to 330 ms for now) which gives approximately a 90-degree turn
- Pauses for 1 s

Notice that the mBot is able to turn left on the spot when both wheels turn clockwise (when observed from the outside of the wheel). The turning angle may not be exactly 90 degrees; you can finetune the constant "TURNING_TIME_MS" to try to make the mBot turn exactly 90 degrees.

Trying out some other functions of the mBot:

In the following, we provide some sample codes for testing out some other functions of the mBot. In each case, copy the code into your Arduino IDE, and save the file using a different filename from the previous activity. Verify and upload the program into the mBot.

Buzzer

Figure 9 provides a sample code for playing a musical tune through mBot's buzzer repeatedly.

```
#include <MeMCore.h>

MeBuzzer buzzer;    // create the buzzer object

void celebrate() {
  // Each of the following "function calls" plays a single tone.
  // The numbers in the bracket specify the frequency and the duration (ms)
  buzzer.tone(392, 200);
  buzzer.tone(523, 200);
  buzzer.tone(659, 200);
  buzzer.tone(784, 200);
  buzzer.tone(659, 150);
  buzzer.tone(784, 400);
  buzzer.noTone();
}

void setup() {
  // Any setup code here runs only once:
}

void loop() {
  // The main code here will run repeatedly (i.e., looping):
  celebrate(); // play the tune specified in the function celebrate()
  delay(1000); // pauses for 1000 ms before repeating the loop
}
```

Figure 9: Sample Arduino program code for testing the mBot's buzzer.

RGB LEDs

The mBot has two RGB (i.e., red, green, and blue) LEDs on its mCore control board. Figure 10 provides a sample code for displaying some colours using these two RGB LEDs. The function call "led.setColorAt(LED_num, redIntensity, greenIntensity, blueIntensity)" takes in four parameters:

- LED_num: '0' for Right LED, '1' for Left LED
- redIntensity: specifies the light intensity for red colour, ranging from 0-255
- greenIntensity: specifies the light intensity for green colour, ranging from 0-255
- blueIntensity: specifies the light intensity for blue colour, ranging from 0-255

The function call "led.setColor(redIntensity, greenIntensity, blueIntensity)", on the other hand, takes in only three parameters. It synchronizes the displayed colour on both Left and Right LEDs. Check out the following website to see how you can create different colours by varying the individual RGB intensities: https://www.w3schools.com/colors/colors_rgb.asp

```
#include <MeMCore.h>

MeRGBLed led(0,30); // Based on hardware connections on mCore; cannot change

void setup() {
  led.setpin(13);
}

void loop() {
  led.setColor(255, 255, 255); // set both LEDs to white colour
  led.show(); // Must use .show() to make new colour take effect
  delay(500);

  led.setColorAt(0, 255, 0, 0); // set Right LED to Red
  led.setColorAt(1, 0, 0, 255); // set Left LED to Blue
  led.show();
  delay(500);

  led.setColorAt(0, 0, 0, 255); // set Right LED to Blue
  led.setColorAt(1, 255, 0, 0); // set Left LED to Red
  led.show();
  delay(500);
}
```

Figure 10: Sample Arduino code for testing the mBot's two RGB LEDs.

Ultrasonic Sensor

The ultrasonic sensor is able to detect obstacles from a range of **3 cm** to **400 cm**. Figure 11 provides a sample code for testing the ultrasonic sensor. Here, we included a TIMEOUT that limits the time duration the sensor will wait for the reflected ultrasonic pulses. This makes the mBot more responsive while navigating the maze later. You can adjust this TIMEOUT on your own.

To see the output, we need to open up the serial monitor. Refer to Figure 4 to see which button to click to open up the serial monitor.

```
#define TIMEOUT 2000 // Max microseconds to wait; choose according to max distance of wall
#define SPEED_OF_SOUND 340 // Update according to your own experiment

#define ULTRASONIC 12
// If you are using Port 1 of mCore, the ultrasonic sensor uses digital pin 12
// If you are using Port 2 of mCore, the ultrasonic sensor uses digital pin 10

void setup() {
  Serial.begin(9600); // to initialize the serial monitor
}

void loop() {
  pinMode(ULTRASONIC, OUTPUT);

  digitalWrite(ULTRASONIC, LOW);
  delayMicroseconds(2);
  digitalWrite(ULTRASONIC, HIGH);
  delayMicroseconds(10);
  digitalWrite(ULTRASONIC, LOW);

  pinMode(ULTRASONIC, INPUT);
  long duration = pulseIn(ULTRASONIC, HIGH, TIMEOUT);
  if (duration > 0) {
    Serial.print("distance(cm) = ");
    Serial.println(duration / 2.0 / 1000000 * SPEED_OF_SOUND * 100);
  }
  else {
    Serial.println("out of range");
  }
  delay(500);
}
```

Figure 11: Sample Arduino code for testing the mBot's ultrasonic sensor.

Line Follower

The Line Follower consists of two pairs of IR emitter + detector, as shown in Figure 12 below. One pair is called "**Sensor1**", while the other pair is called "**Sensor2**".

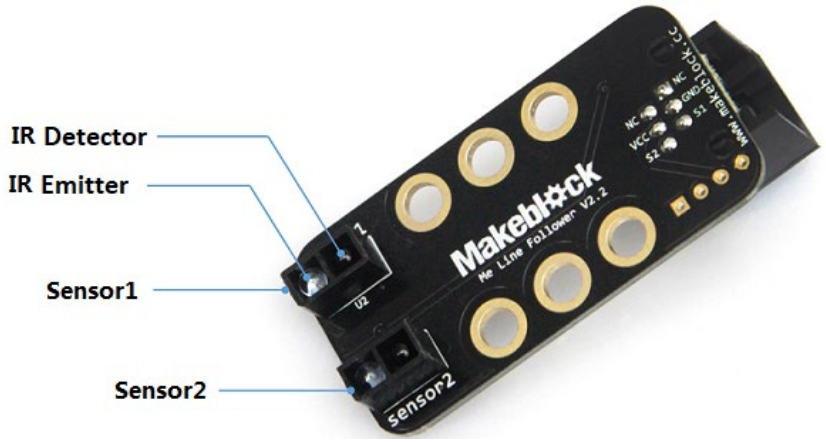


Figure 12: mBot's Line Follower (Picture Credit: http://mBlock.cc).

The Line Follower utilizes the *indirect* incidence operating mode. Each IR emitter continuously shines IR rays onto the surface underneath. When the surface reflects the IR rays, the IR detector will detect them, and regard the surface as non-black; otherwise, it will regard the surface as black.

Figure 13 below shows the four possible situations during the mBot's line following process, while Figure 14 provides a simple code that allows the mBot to perform the line following function. Note that the code is not optimized for performance; a better version can be found within the mBot's factory firmware (see the next section on how to obtain this factory firmware).

Note: The mBot will only start moving after you have pressed the button on the mCore (front left).

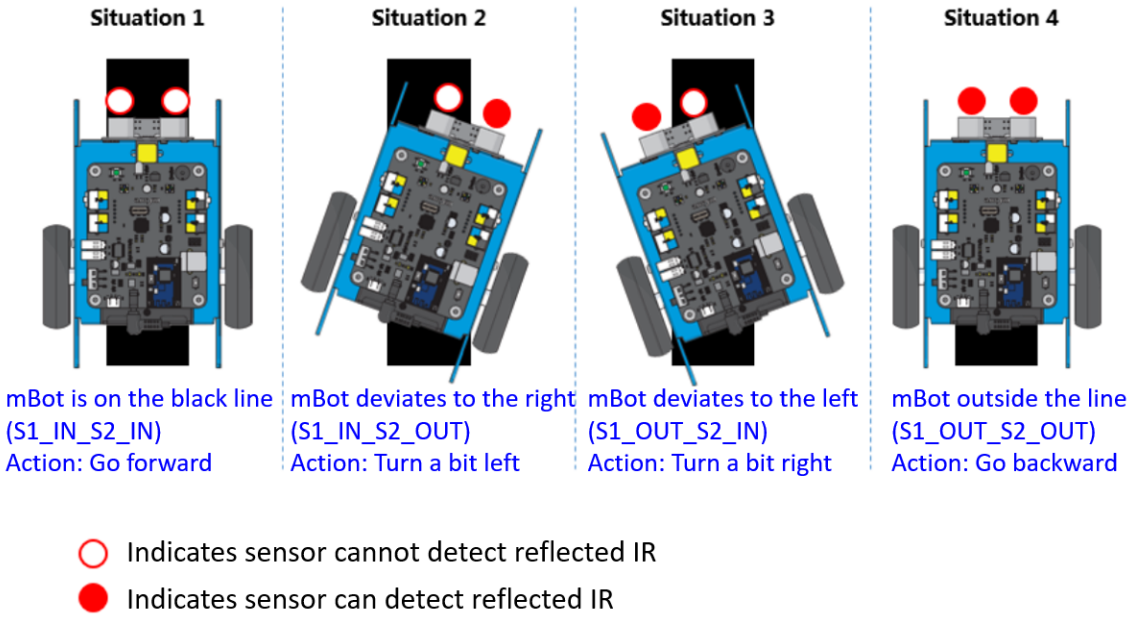


Figure 13: Four possible situations reported by Line Follower (Picture Credit: http://mBlock.cc).

```
#include "MeMCore.h"

MeLineFollower lineFinder(PORT_2); // assigning lineFinder to RJ25 port 2
MeDCMotor leftMotor(M1); // assigning leftMotor to port M1
MeDCMotor rightMotor(M2); // assigning RightMotor to port M2

int status = 0; // global status; 0 = do nothing, 1 = mBot runs

void setup() {
  pinMode(A7, INPUT); // Setup A7 as input for the push button
  Serial.begin(9600); // Setup serial monitor for debugging purpose
}

void loop() {
  if (analogRead(A7) < 100) { // If push button is pushed, the value will be very low
    status = 1 - status; // Toggle status
    delay(500); // Delay 500ms so that a button push won't be counted multiple times.
  }

  if (status == 1) { // run mBot only if status is 1
    int sensorState = lineFinder.readSensors(); // read the line sensor's state
    if (sensorState == S1_IN_S2_IN) { // situation 1
      leftMotor.run(-200); // Left wheel goes forward (anti-clockwise)
      rightMotor.run(200); // Right wheel goes forward (clockwise)
    }
    else if (sensorState == S1_IN_S2_OUT) { // situation 2
      leftMotor.run(0); // Left wheel stops
      rightMotor.run(150); // Right wheel go forward
    }
    else if (sensorState == S1_OUT_S2_IN) { // situation 3
      leftMotor.run(-150); // Left wheel go forward
      rightMotor.run(0); // Right wheel stops
    }
    else if (sensorState == S1_OUT_S2_OUT) { // situation 4
      leftMotor.run(100); // Left wheel reverses (clockwise)
      rightMotor.run(-100); // Right wheel reverses (anti-clockwise)
    }
    delay(20); // decision making interval (in milliseconds)
  }
}
```

Figure 14: Sample Arduino program code for line following.

Light Sensor

As seen in Figure 2 earlier, the mCore has a Light Sensor, which is constructed using a Light Dependent Resistor (LDR). Figure 15 below provides some sample code for reading the value from the Light Sensor. Note that the reading is obtained using a 10-bit analog-to-digital converter, which ranges from 0 to 1023.

```
#define LIGHTSENSOR A6 // internally connected to analog pin A6 in mCore

void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.print("Value = ");
  Serial.println(analogRead(LIGHTSENSOR));
  delay(500);
}
```

Figure 15: Sample Arduino program code for reading the value from the Light Sensor.

mBot's Factory Firmware:

You can learn a lot about how the mBot works by going through its factory firmware, and observing the mBot's behavior when it runs the firmware. You can load the factory firmware into the Arduino IDE by clicking on the "Open" button (see Figure 4) and navigating to the file through the menus shown in Figure 16 below. If you wish to modify some parts of the code for testing, you should save it under a new filename.

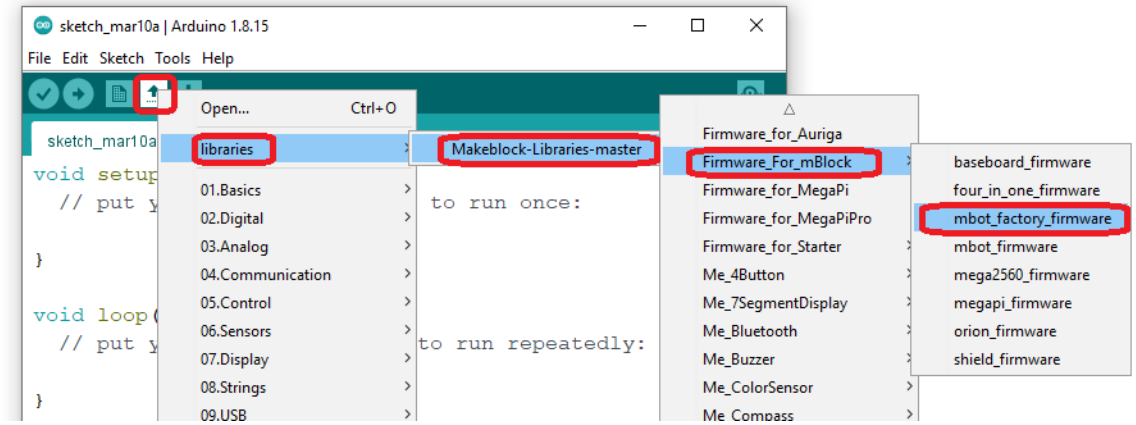


Figure 16: Loading the mBot's factory firmware.

That's all for this introduction to mBot's hardware and some of its basic functions. Have fun trying them out!