DENİZLİ
İL MİLLÎ EĞİTİM MÜDÜRLÜĞÜ

FUTURE LANGUAGE IS ROBOTIC CODING

2019-1TR01-KA201-077586

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY

Avrupa Birliği tarafından finanse edilmektedir.

CHAPTER I

mBot

A. Introduction

With a screwdriver and step-by-step instructions, children can build their own robot from scratch and enjoy the fun of hands-on creation. The building process provides a perfect opportunity to introduce kids to the basics of robotic machinery and electronic parts. They can easily get started with block-based programming to play with mBot.

a. Use mBlock 5

Use mBlock 5 to play with your mBot. Graphical programming is ideal to show kids the magic of programming. As they progress, they can even further delve into more complicated Arduino C programming.

You can get mBlock 5 on your PC, phones and tablets, or use mBlock 5 in a web browser.

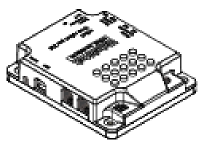
For PC, please visit: *http://www.mblock.cc/mblock-software/*

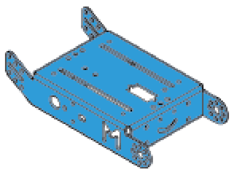
For Android and iOS, please search "mblock" in any application store to download

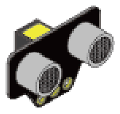
For web browser, please visit: *https://ide.makeblock.com/*

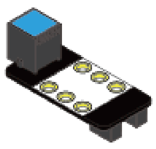
B. Building mBot

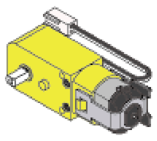
a. Parts List


Mainboard mCore (1)

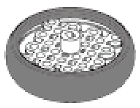

Chassis (1)

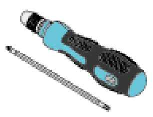

Ultrasonic Sensor (1)

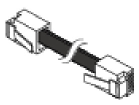

Line-follower Sensor (1)

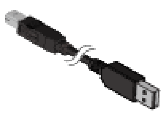

Motor (2)

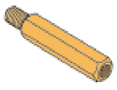
Mini Wheel (1)

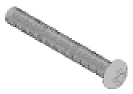

Wheel (2)

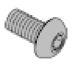

Screwdriver (1)

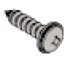

RJ25 Cable (2)


USB Cable (1)


M4*25mm Brass Stud (4)

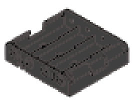

M3*25mm Screw (6)

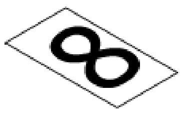

M4*8mm Screw (15)

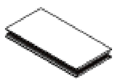

M2.2*9mm Self-drilling Screw (4)

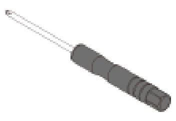
M3 Nut (8)

Infrared Remote Controller (1)

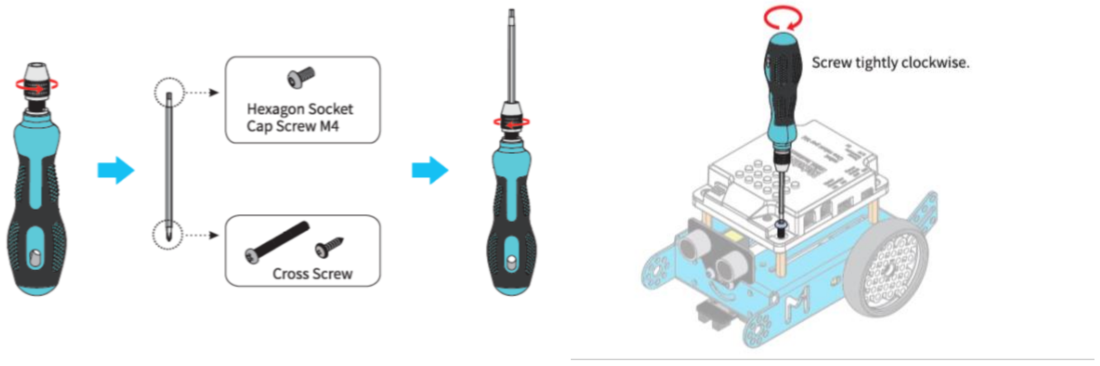

AA Battery Holder (1)


Line-follower Map (1)

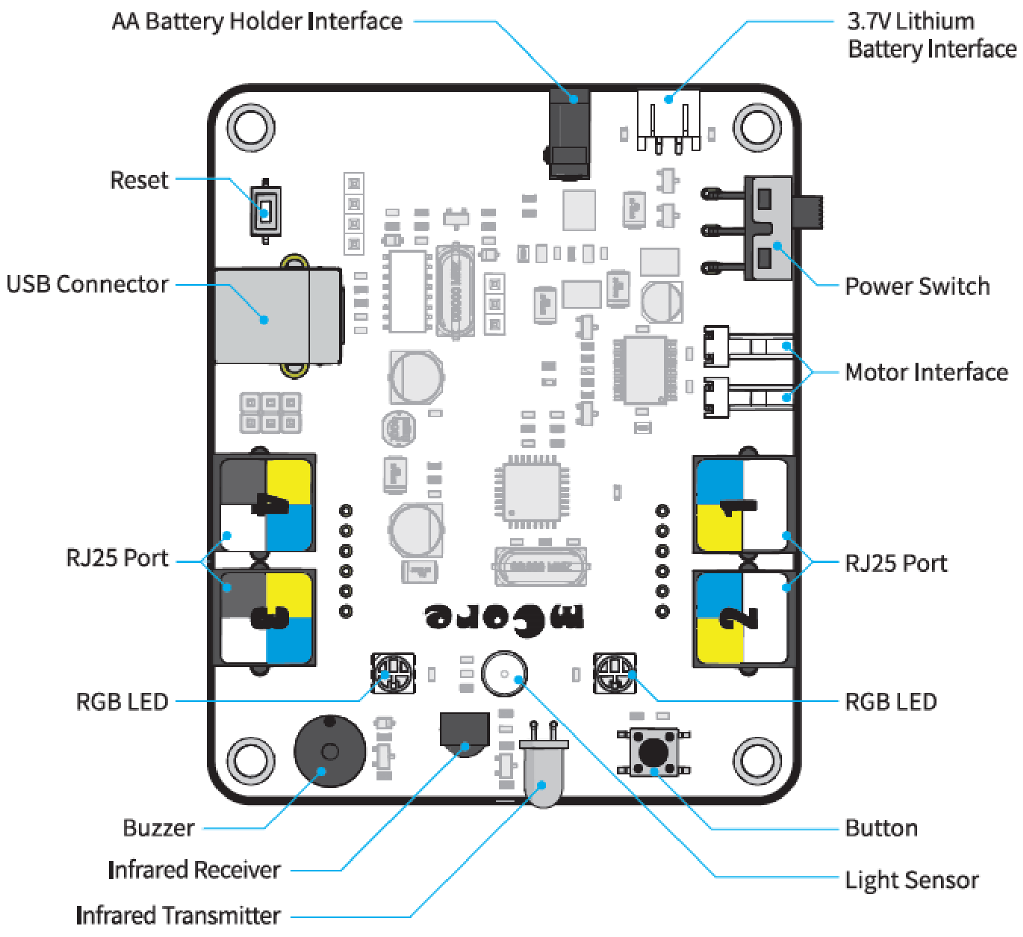

Velcro sticker pad (2)


Mini Screwdriver (1)

b. Use of screwdriver

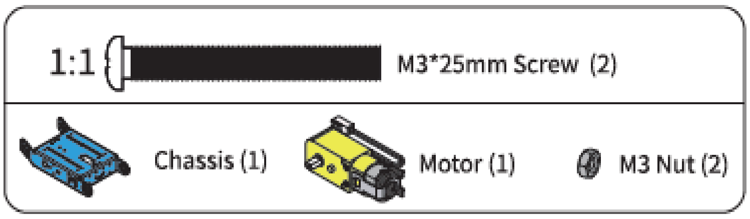


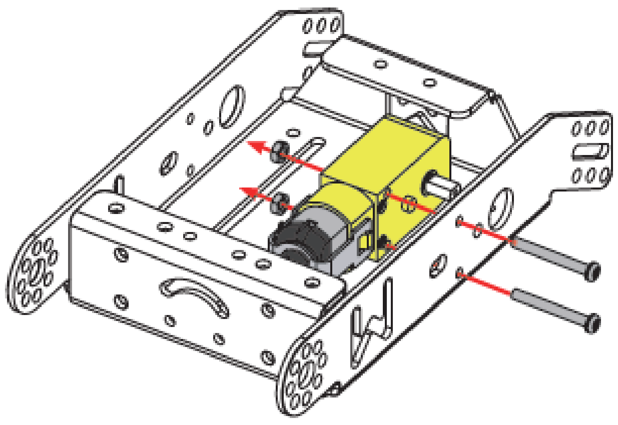
c. Main board



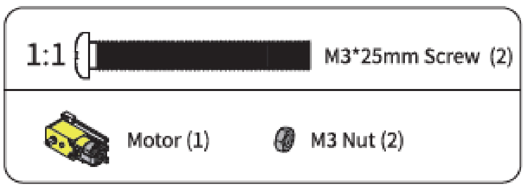
d. Building instructions

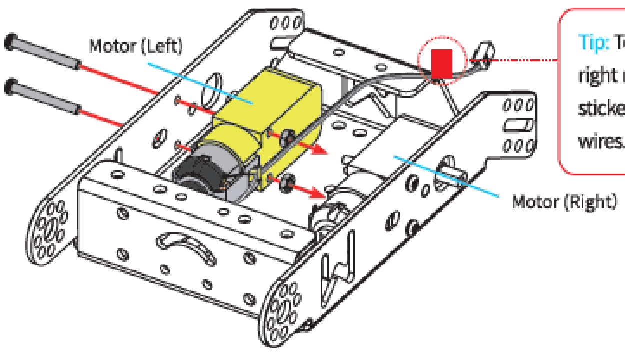
Step 1





Step 2



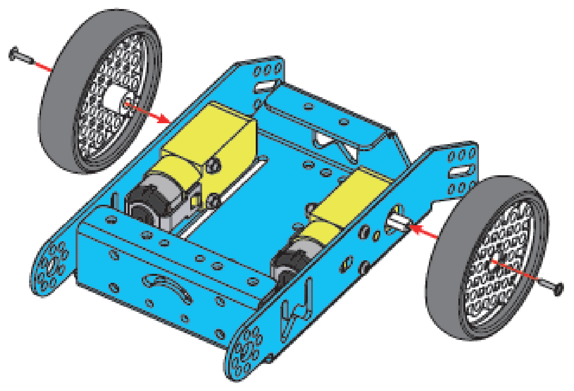


Tip: To easily distinguish between the left and right motor cable, it is advised for you to use stickers, thin string, or pen to mark the motor wires.

Step 3

1:1 M2.2x9mm Self-drilling Screw (2)

Wheel (2)



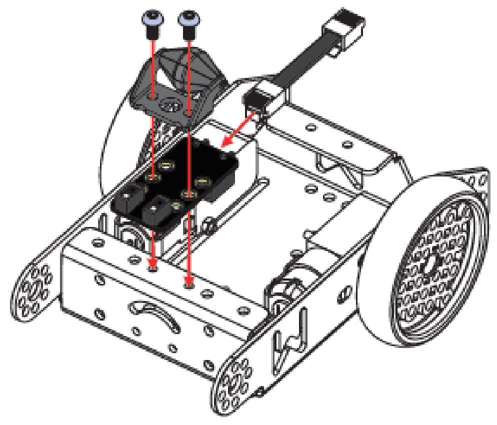
Step 4

1:1 M4*8mm Screw (2)

Line-follower Sensor (1)

Mini Wheel (1)

RJ25 Cable (1)

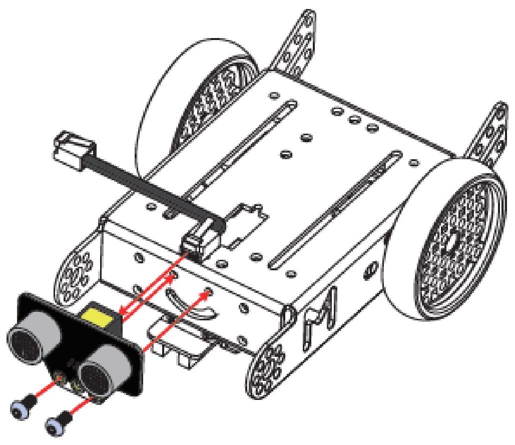


Step 5

1:1 M4*8mm Screw (2)

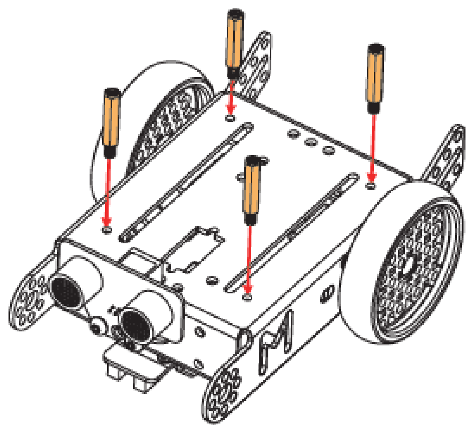
Ultrasonic Sensor (1)

RJ25 Cable (1)



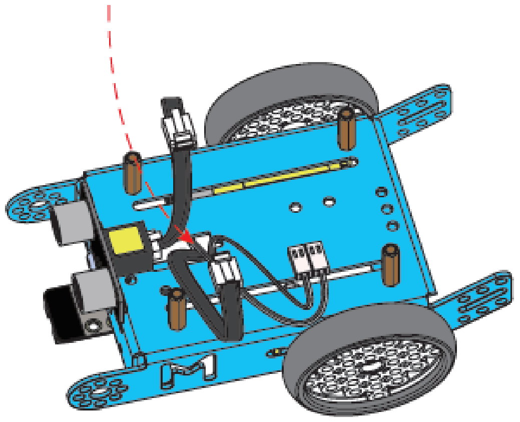
Step 6

M4*25mm Brass Stud (4)



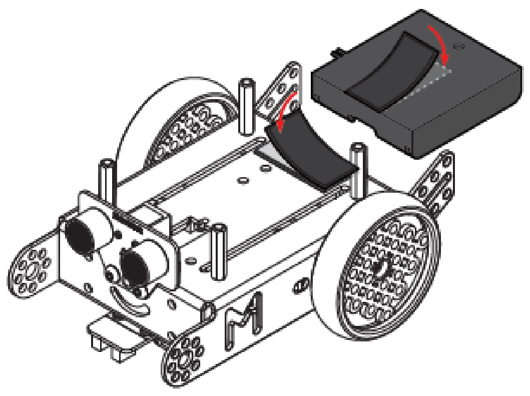
Step 7

Warning: be sure to place the RJ25 connecting line and the motor cable in the direction shown in the picture.



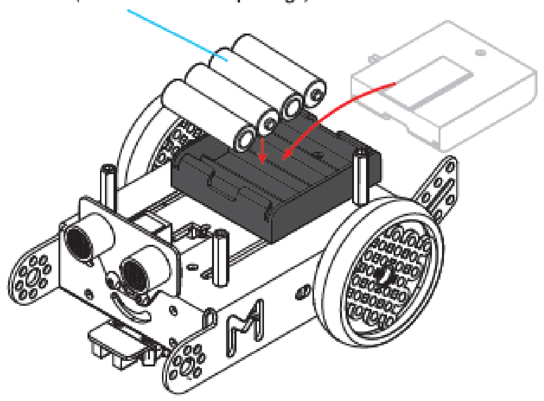
Step 8

AA Battery Holder (1)	
Velcro sticker pad (1)	



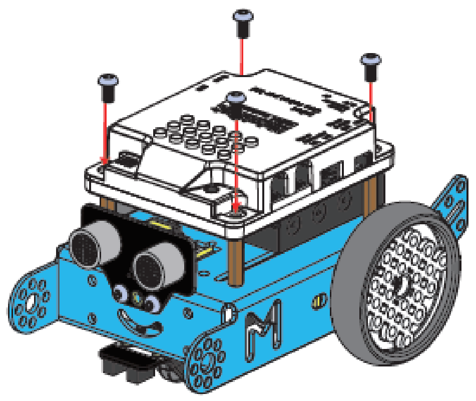
Tip: You can also power your mBot using a 3.7 V lithium battey (not included in the package).

Step 9



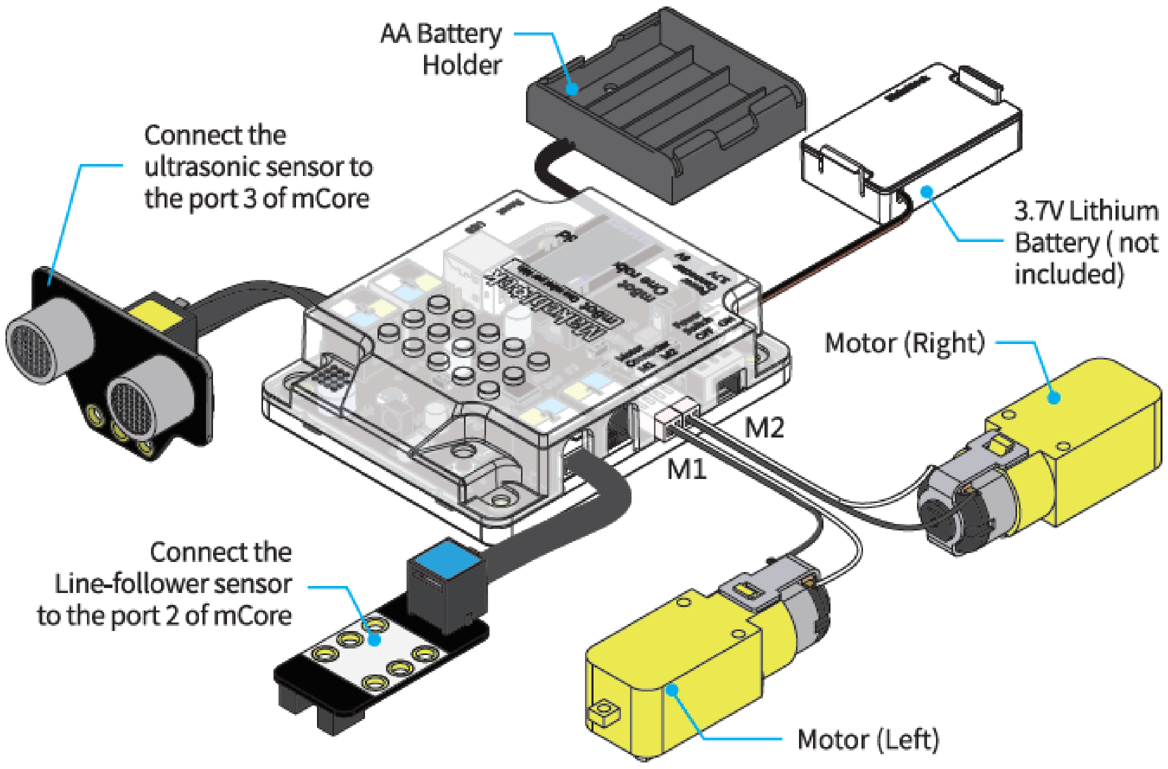
Step 10

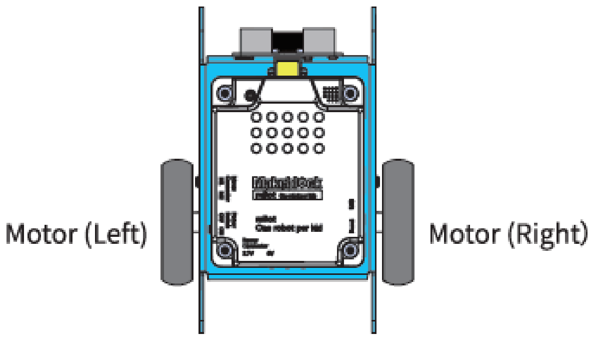
1:1	M4*8mm Screw (4)
	Mainboard mCore (1)



Step 11

Please consult the graphic below to finish wiring.


AA Battery Holder
Connect the ultrasonic sensor to the port 3 of mCore
3.7V Lithium Battery (not included)
Motor (Right)
M2
M1
Connect the Line-follower sensor to the port 2 of mCore
Motor (Left)

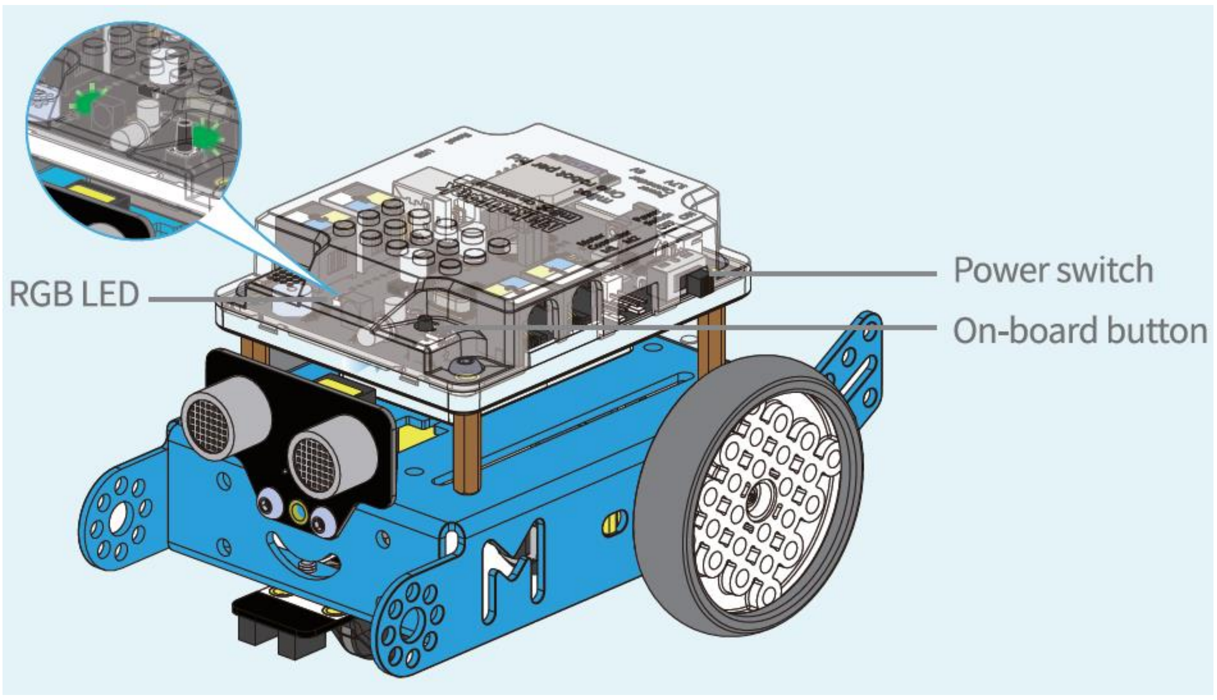


C. Preset Modes

There are three preset modes of mBot:

- Obstacle avoidance
- Infrared remote control
- line-following

Switch modes using *the infrared remote controller or the on-board button.*



The color of the LED on the main board indicates the current mode:

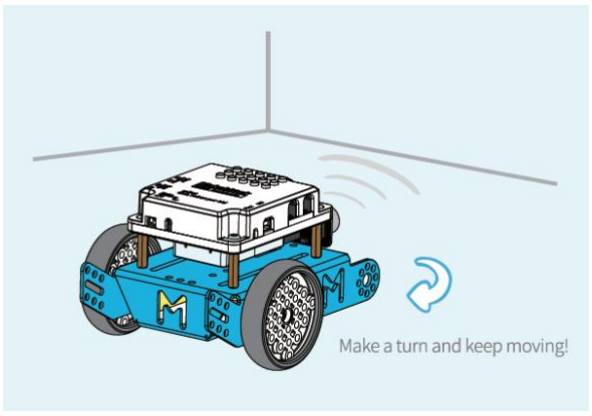
○ White: infrared remote control mode

● Green: obstacle avoidace mode

● Blue: line-following mode

a. Obstacle avoidance mode

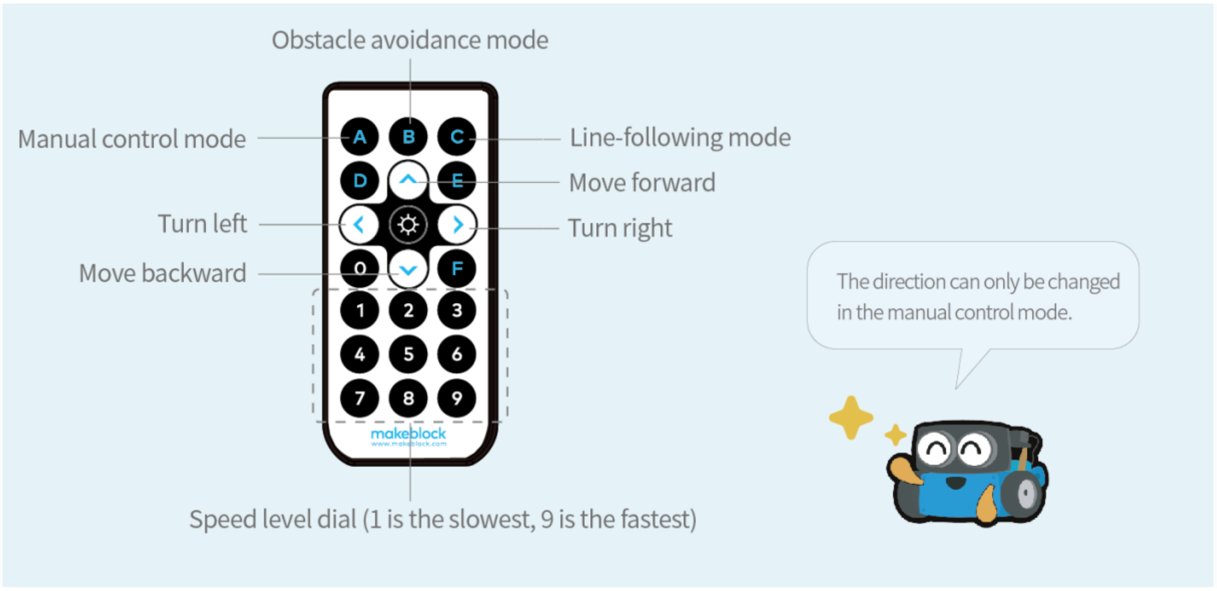
In obstacle avoidance mode, mBot moves and avoids obstacles autonomously.



b. Infrared remote control mode

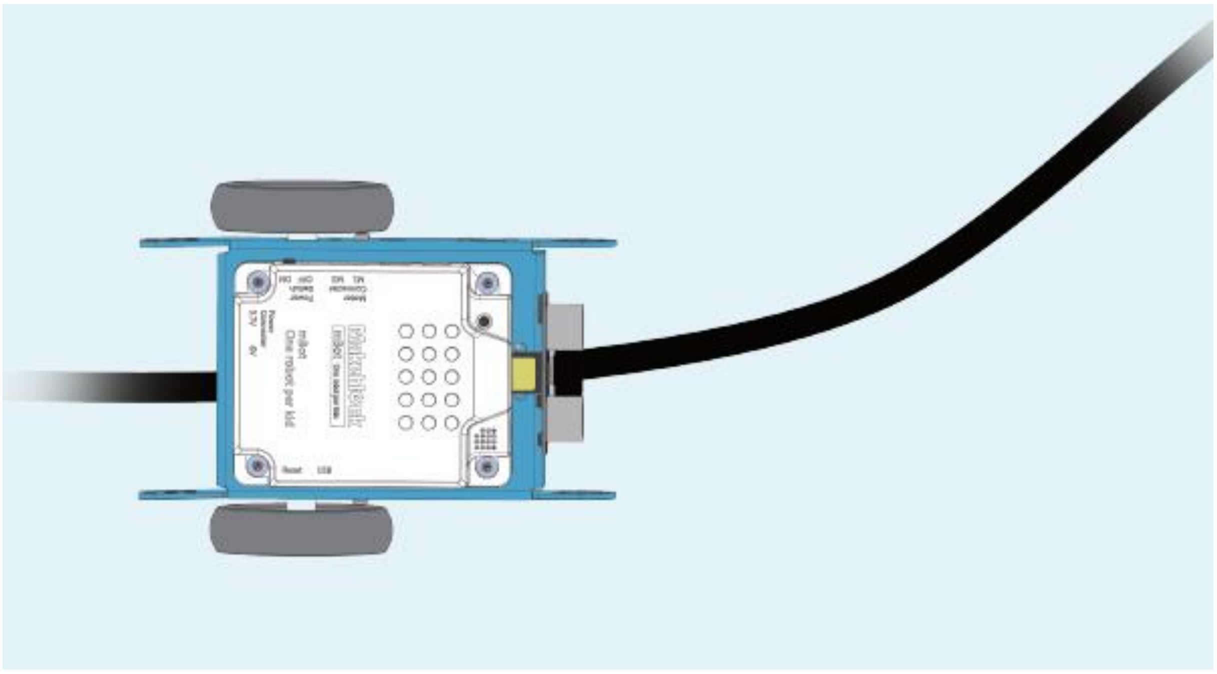
A special infrared remote controller is included in the package. You can use the controller to control mBot, such as speed, direction, and others.

Note: place your mBot on a piece of smooth and flat ground.


Obstacle avoidance mode
Manual control mode
Line-following mode
Move forward
Turn left
Turn right
Move backward
The direction can only be changed in the manual control mode.
Speed level dial (1 is the slowest, 9 is the fastest)

c. Line-following mode

In line-following mode, mBot moves autonomously along the black lines on the map.



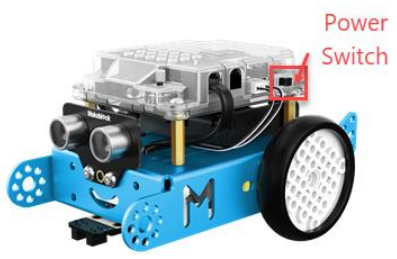
D. Connect mBot

You can connect mBot to mBlock 5 through a USB cable, Bluetooth 4.0, or the 2.4G module.

- Method 1: using a USB cable
- Method 2: using Bluetooth 4.0
- Method 3: using the 2.4G module

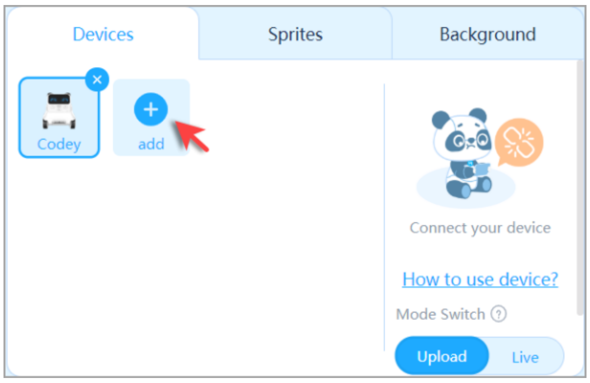
a. Method 1: using a USB cable

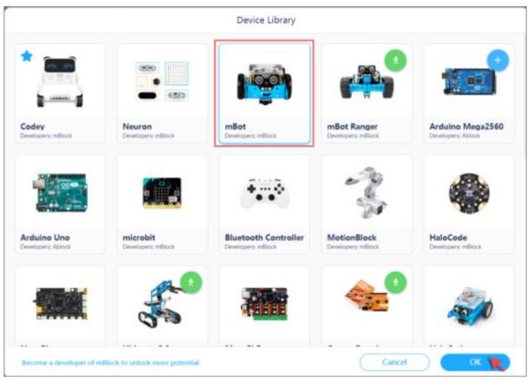
1. Use the USB Cable that came with mBot to connect your mBot to a USB port on you computer.



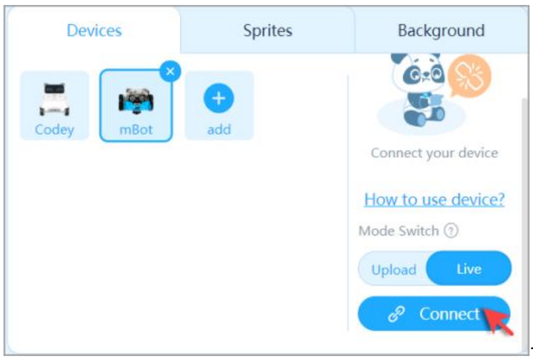
2. Power on your mBot.

3. Under **Devices**, click **add**, choose **mBot** from the pop-up **Device Library** window, and click **OK**.

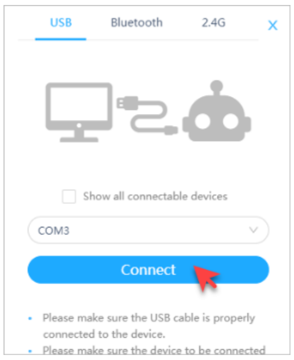




4. Click **Connect**, and from the pop-up device connection window, click **Connect** on the **USB** tab.



→



- Please make sure the USB cable is properly connected to the device.
- Please make sure the device to be connected

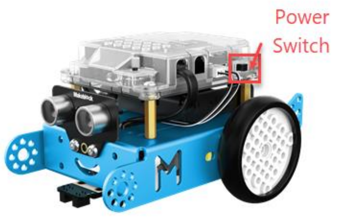
b. Method 2: using Bluetooth 4.0

System requirements:

Windows: the version of Bluetooth must be 4.0; for other Bluetooth versions, a Bluetooth 4.0 adapter is recommended (refer to Bluetooth 4.0 Adapter for detailed instructions)

Mac OS: support most Mac OS models

1. Power on your mBot.

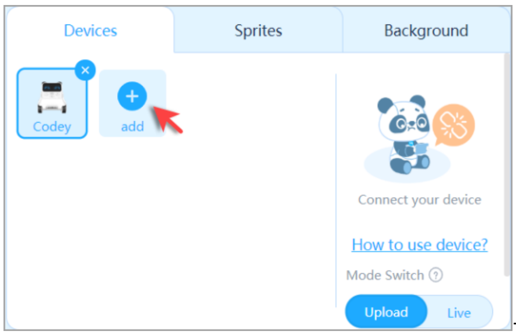


2. Turn on the Bluetooth on your PC.

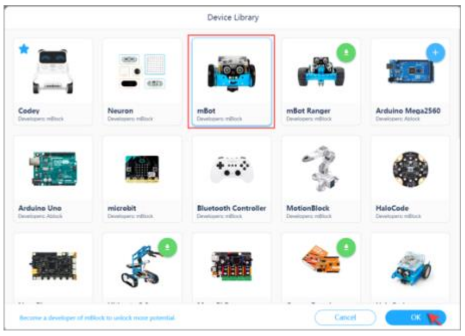
Windows: on the task bar, select **action center** > **Bluetooth**

Mac OS: choose **Apple menu** > **System Preferences**, then click **Bluetooth**

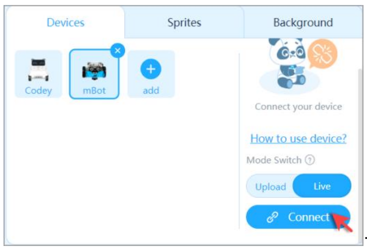
3. Under **Devices**, click **add**, choose **mBot** from the pop-up **Device Library** window, and click **OK**.



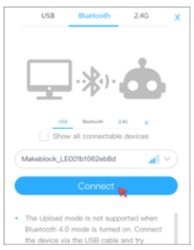
→



4. Click **Connect**, and from the pop-up device connection window, click **Connect** on the **Bluetooth 4.0** tab.



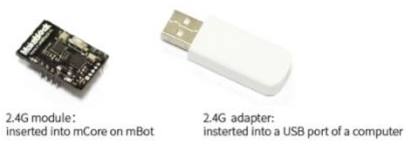
→



c. Method 3: using the 2.4G module

Computers with the following systems are recommended:

Windows: Windows 7 or later
Mac OS: macOS Sierra 10.12 or later
To connect mBot in this way, you need to use the 2.4G module and adapter.

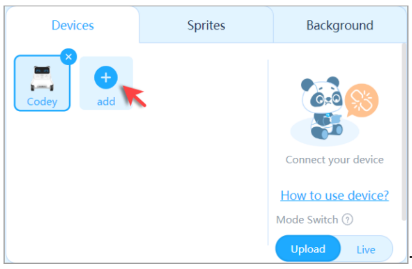

2.4G module:
inserted into mCore on mBot
2.4G adapter:
insterted into a USB port of a computer

You can use them simply by plugging them. No drivers or pairing is required.

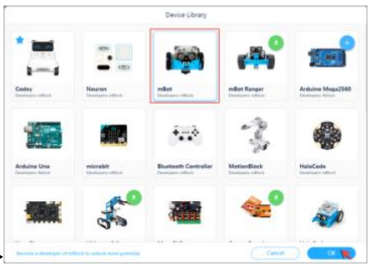
1. Insert the 2.4G module into mCore on mBot and insert the adapter into a USB port of a computer.

2. Power on your mBot.

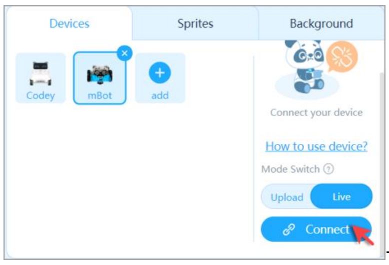
3. Under **Devices**, click **add**, choose **mBot** from the pop-up **Device Library** window, and click **OK**.



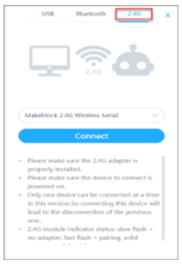
→



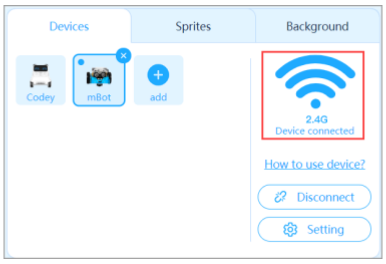
4. Click **Connect**, and from the pop-up device connection window, click **Connect** on the **2.4G** tab.



→



- Please make sure the 2.4G adapter is properly installed.
- Please make sure the device to connect is powered on.
- Only one device can be connected at a time in this version.So connecting this device will lead to the disconnection of the previous one.
- 2.4G module indicator status: slow flash = no adapter, fast flash = pairing, solid



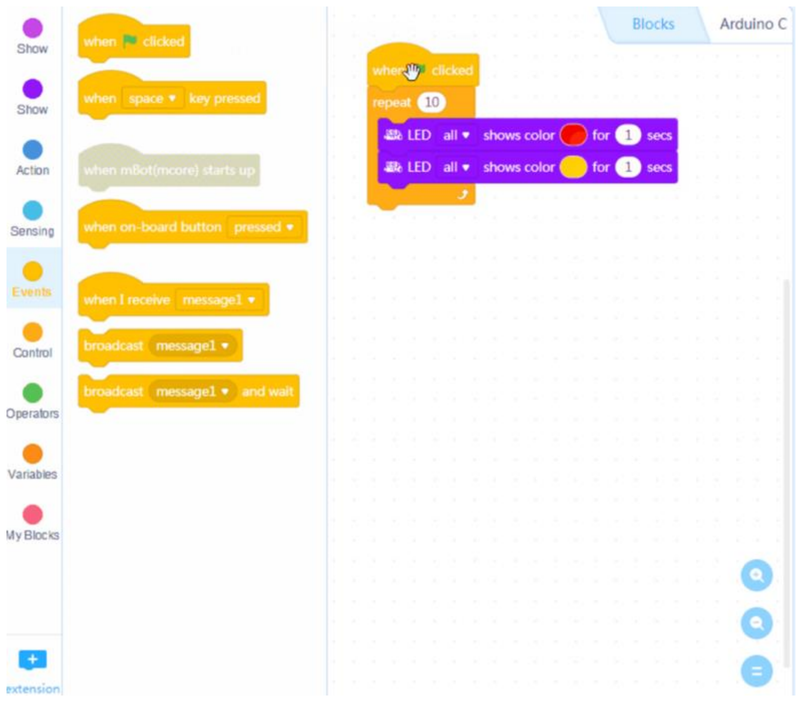
After the successful connection, mBlock 5 displays the following information indicating the 2.4G connection.

E. Get Started

Let's get started with a simple project. The LEDs of mBot will switch between red and yellow.

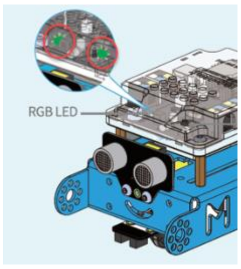
Connect mBot

1. Click "Connect" to connect mBot.
2. Drag two Show blocks LED(all) shows color () for () secs to the Scripts area. Change the color of the second block to yellow.
3. Add a Control block repeat (10) to wrap up the two Show blocks.
4. Add an Events block when green flag clicked and snap it on top of other blocks.
5. Click the green flag under the stag and see what happens.



F. Block Reference

a. Show

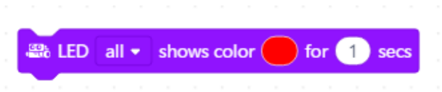


LEDs of mBot

There are two on-board RGB LEDs, as shown in the image above.

1. LED (all) shows color () for () secs

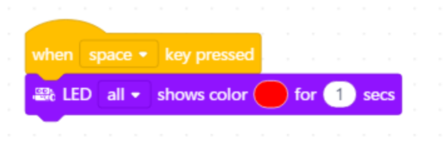
Lights up the specified LED/LEDs the specified color for the specified amount of seconds. There are three options: all LEDs, left LED, or right LED.



```
LED all ▾ shows color 🟥 for 1 secs
```

Example:

When the space key is pressed, both LEDs will light up red for 1 second.

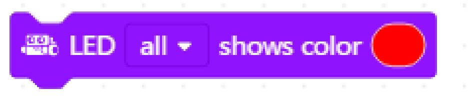


```
when space ▾ key pressed
LED all ▾ shows color 🟥 for 1 secs
```

---

2. LED (all) shows color ()

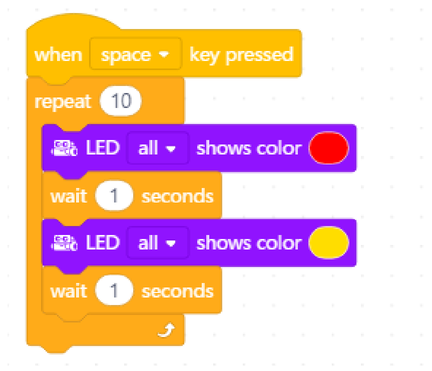
Lights up the specified LED/LEDs the specified color. There are three options: all LEDs, left LED, or right LED.



```
LED all ▾ shows color 🟥
```

Example:

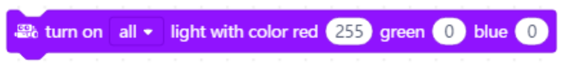
When the space key is pressed, both LEDs will switch between red and yellow for 10 times, with the time gap of 1 second.



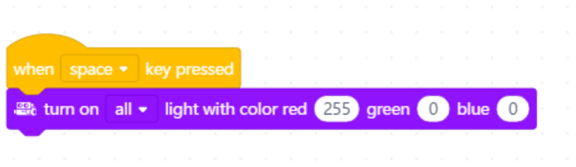
```
when space ▾ key pressed
repeat 10
    LED all ▾ shows color 🟥
    wait 1 seconds
    LED all ▾ shows color 🟨
    wait 1 seconds
```

3. turn on (all) light with color red() green() blue()

Lights up the specified LED/LEDs the specified color, mixed by specified RGB color values. There are three options: all LEDs, left LED, or right LED.


turn on all ▾ light with color red 255 green 0 blue 0

Example:

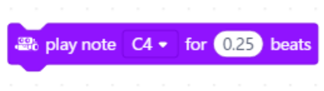

when space ▾ key pressed
turn on all ▾ light with color red 255 green 0 blue 0

When the space key is pressed, both LEDs will light up red.

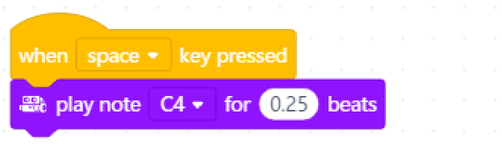
---

4. play note (C4) for () beats

Plays the specified note for the specified number of beats.


play note C4 ▾ for 0.25 beats

Example:

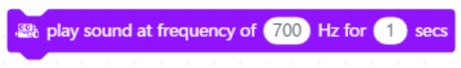

when space ▾ key pressed
play note C4 ▾ for 0.25 beats

Whe n the space key is pressed, the note C4 will be played for 0.25 beat.

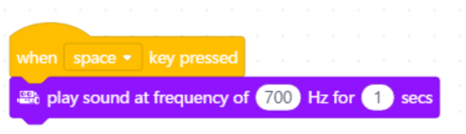
---

5. play sound at frequency of () Hz for () secs

Plays sound at the specified frequency for the specified amount of time.


play sound at frequency of 700 Hz for 1 secs

Example:

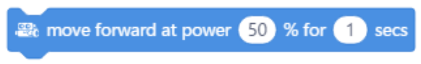

when space ▾ key pressed
play sound at frequency of 700 Hz for 1 secs

When the space key is pressed, the sound at frequency 700 Hz will be played for 1 second.

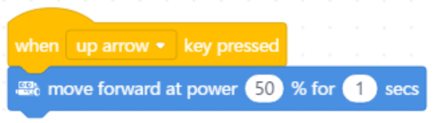
b. Action

1. move forward at power (50)% for () secs

Moves mBot forward at the specified power for the specified amount of time.



Example:

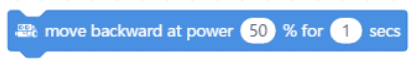

when up arrow key pressed
move forward at power 50 % for 1 secs

When the "↑" key is pressed, mBot will move forward at 50% power for 1 second.

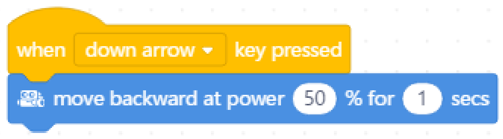
---

2. move backward at power (50)% for () secs

Moves mBot backward at the specified power for the specified amount of time.



Example:

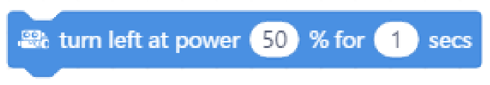

when down arrow key pressed
move backward at power 50 % for 1 secs

When the "↓" key is pressed, mBot will move backward at 50% power for 1 second.

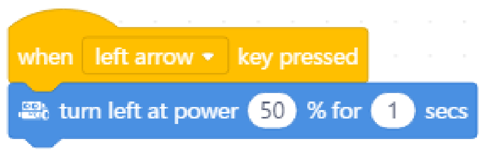
---

3. turn left at power (50)% for () secs

Turns mBot left at the specified power for the specified amount of time.



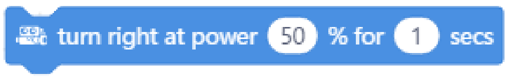
Example:


when left arrow key pressed
turn left at power 50 % for 1 secs

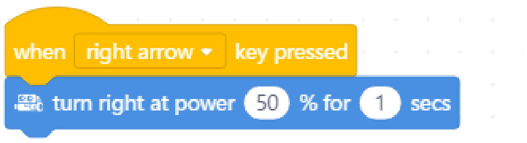
When the "←" key is pressed, mBot will turn left at 50% power for 1 second.

4. turn right at power (50)% for () secs

Turns mBot right at the specified power for the specified amount of time.



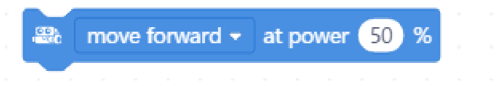
Example:


when right arrow key pressed
turn right at power 50 % for 1 secs

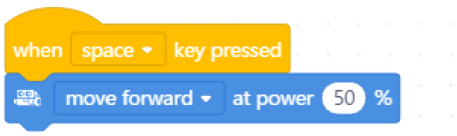
When the "→" key is pressed, mBot will turn right at 50% power for 1 second.

5. (move forward) at power (50)%

Moves mBot to the specified direction at the specified power. There are four options: "move forward", "move backward", "turn left", or "turn right".



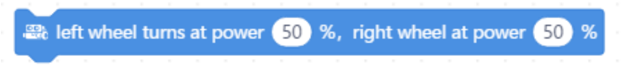
Example:



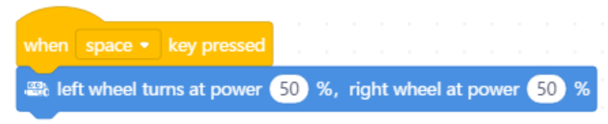
When the space key is pressed, mBot will keep moving forward at 50% power.

6. left wheel turns at power (50)%, right wheel at power (50)%

Turns the left wheel and right wheel at the specified power.



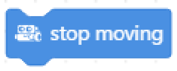
Example:



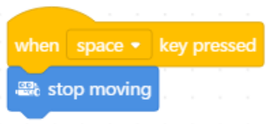
When the space key is pressed, both the left wheel and the right wheel will move at 50% power.

7. stop moving

Makes mBot stop moving.



Example:

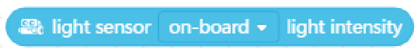


When the space key is pressed, mBot will stop moving.

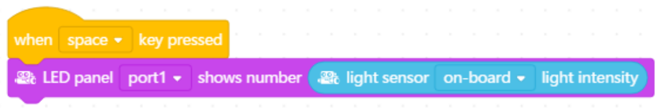
c. Sensing

1. light sensor (on-board) light intensity

Reports light intensity by the specified light sensor.



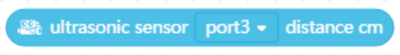
Example:


when space key pressed
LED panel port1 shows number light sensor on-board light intensity

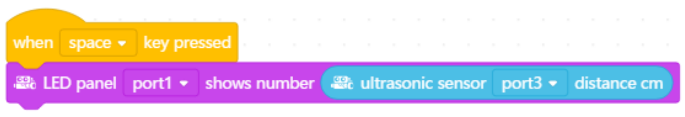
When the space key is pressed, the light intensity value will be displayed on the external LED panel.

2. ultrasonic sensor (port3) distance cm

Reports the distance of obstacles detected by the ultrasonic sensor that is connected to the specified port.



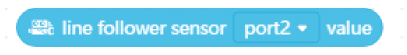
Example:


when space key pressed
LED panel port1 shows number ultrasonic sensor port3 distance cm

When the space key is pressed, the distance of obstacles detected by the ultrasonic sensor will be displayed on the external LED panel.

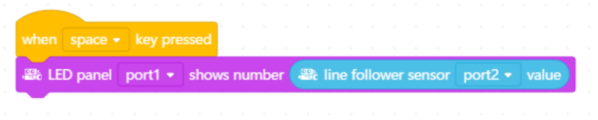
3. line follower sensor (port2) value

Reports the value detected by the specified line follower sensor.



```
line follower sensor (port2 ▾) value
```

Example:

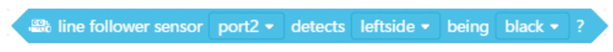


```
when (space ▾) key pressed
LED panel (port1 ▾) shows number (line follower sensor (port2 ▾) value)
```

When the space key is pressed, the value detected by the line follower sensor will be displayed on the external LED panel.

4. line follower sensor (port2) detects (leftside) being(black)?

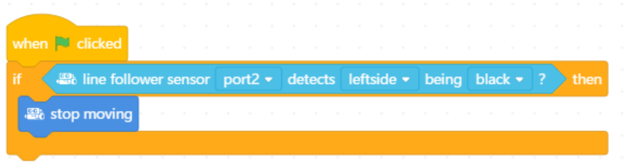
If the color detected by the specified line follower sensor on the specified side is the specified color, the report condition is met.



```
line follower sensor (port2 ▾) detects (leftside ▾) being (black ▾) ?
```

Example:

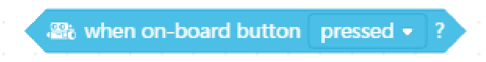
When the green flag is clicked, if the line follower sensor detects black obstacles on the left side, mBot will stop moving.



```
when 🏁 clicked
if <line follower sensor (port2 ▾) detects (leftside ▾) being (black ▾) ?> then
    stop moving
```

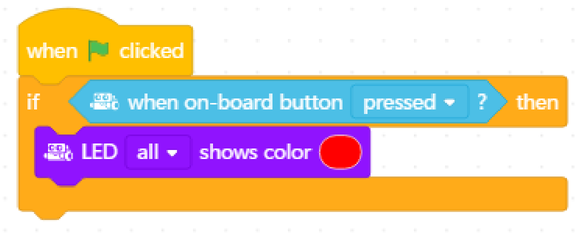
5. when on-board button (pressed)?

If the on-board button is pressed or released, the report condition is met.



```
when on-board button (pressed ▾) ?
```

Example:

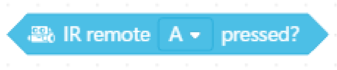


When the green flag is clicked, if the on-board button is pressed, the LEDs of mBot will light up red.

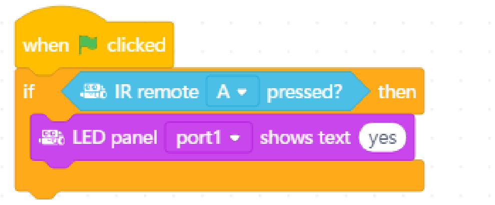
---

6. IR remote (A) pressed?

If the specified button of the IR remote is pressed, the report condition is met.



Example:

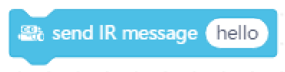


When the green flag is clicked, if button A of the IR remote is pressed, "yes" will be displayed on the external LED panel.

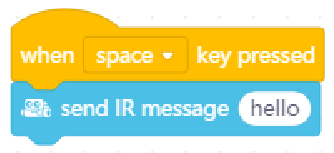
---

7. send IR message (hello)

Sends the specified IR message.



Example:

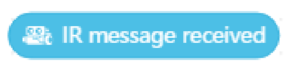


When the space key is pressed, IR message "hello" will be sent out.

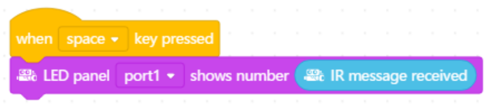
---

8. IR message received

Reports the received IR message.



Example:

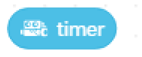


When the space key is pressed, the IR message received will be displayed on the external LED panel.

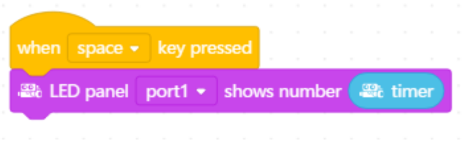
---

9. timer

Reports the value of the timer.



Example:

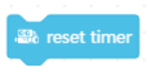


When the space key is pressed, the value of the timer will be displayed on the external LED panel.

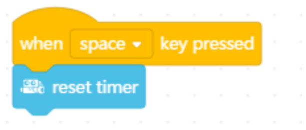
---

10. reset timer

Resets the timer.



Example:

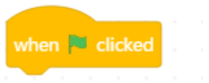


When the space key is pressed, the timer will be reset.

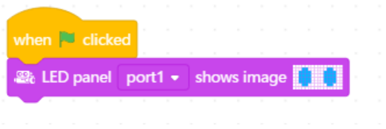
d. Events

1. when green flag clicked

When the green flag is clicked, run the script.



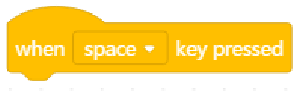
Example:



When the green flag is clicked, the specified image will be displayed on the external LED panel.

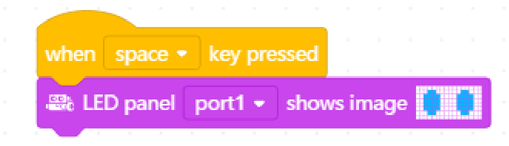
---

2. when (space) key pressed



When the specified key is pressed, run the script.

Example:

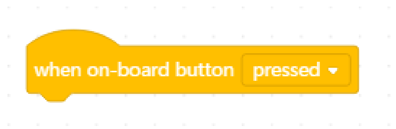


When the space key is pressed, the specified image will be displayed on the external LED panel.

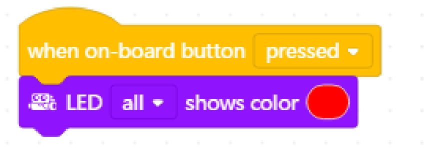
---

3. when on-board button (pressed)

When the on-board button is pressed or released, run the script.



Example:

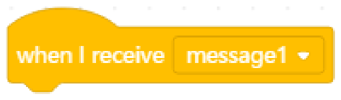


When the on-board button is pressed, all LEDs will light up red.

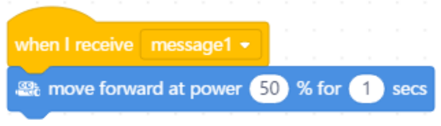
---

4. when I receive (message1)

When the specified message is received, run the script.



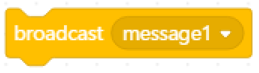
Example:



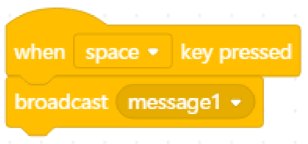
When "message1" is received, mBot will move forward at 50% power for 1 second.

5. broadcast (message1)

Broadcasts the specified message.



Example:

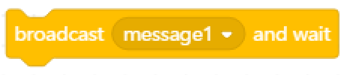


When the space key is pressed, "message1" will be broadcast.

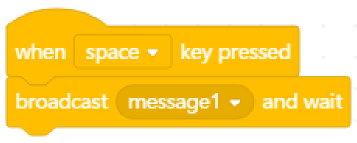
---

6. broadcast (message1) and wait

Broadcasts the specified message and waits until all the scripts activated by this broadcast end.



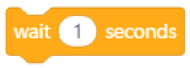
Example:



When the space key is pressed, "message1" will be broadcast and the next script will be executed when all the scripts activated by this broadcast end.

e. Control

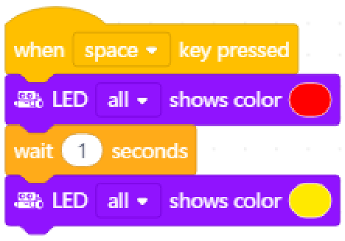
1. wait () seconds



```
wait (1) seconds
```

Waits for a specified period of time to run the script.

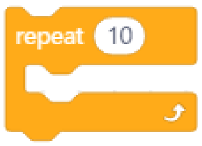
Example:



```
when [space] key pressed
LED [all] shows color 🟥
wait (1) seconds
LED [all] shows color 🟨
```

When the space key is pressed, all LEDs will light up red, and 1 second later, switch to yellow.

2. repeat ()

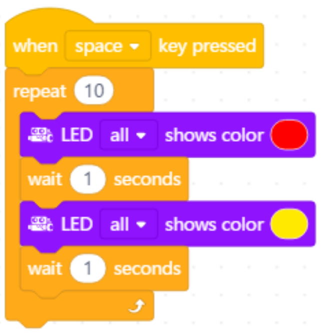


```
repeat (10)
```

Runs the script for the specified number of times.

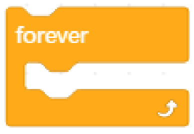
Example:

When the space key is pressed, all LEDs will switch between red and yellow for 10 times.



```
when [space] key pressed
repeat (10)
    LED [all] shows color 🟥
    wait (1) seconds
    LED [all] shows color 🟨
    wait (1) seconds
```

3. forever

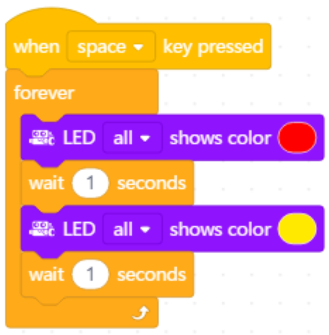


```
forever
```

Runs the script repeatedly.

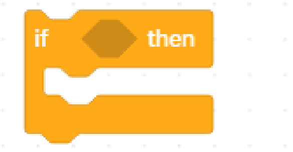
Example:

When the space key is pressed, all LEDs will switch between red and yellow.



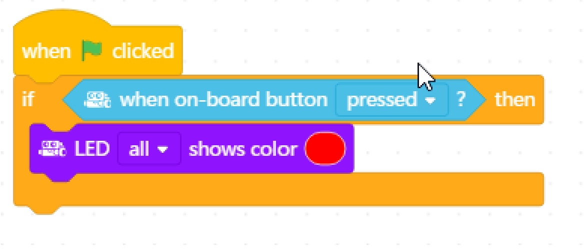
```
when [space] key pressed
forever
    LED [all] shows color 🟥
    wait (1) seconds
    LED [all] shows color 🟨
    wait (1) seconds
```

4. if () then ()



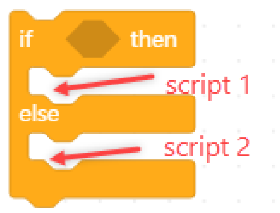
If the report condition is met, run the script.

Example:



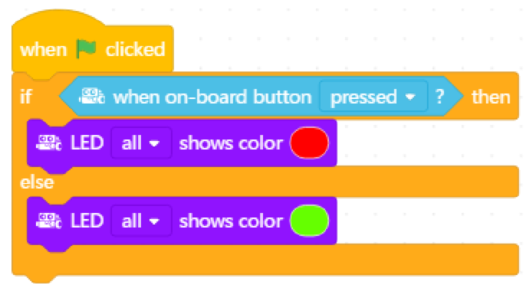
When the green flag is clicked, if the on-board button is pressed, all LEDs will light up red.

5. if () then () else ()



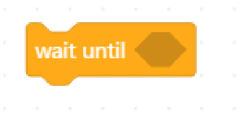
If the report condition is met, run script 1. If not, run script 2.

Example:



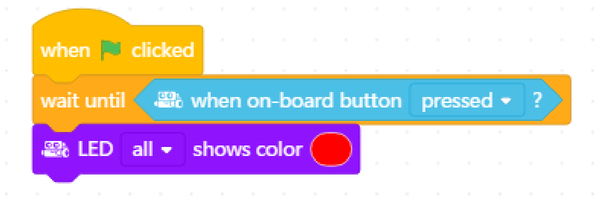
When the space key is pressed, if the on-board button is pressed, all LEDs will light up red, else, green.

6. wait ()



Wait until the report condition is met. Then run the script.

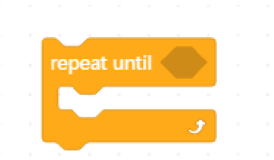
Example:



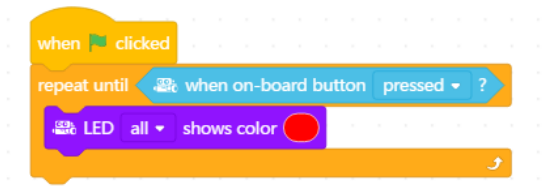
When the green flag is clicked, if the on-board button is pressed, all LEDs will light up red.

7. repeat until ()

Run the script repeatedly until the report condition is met.



Example:

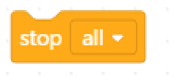


When the green flag is clicked, all LEDs will light up red until the on-board button is pressed.

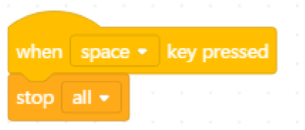
---

8. stop ()

Stop the specified script or scripts. There are three options: all, this script, or other scripts in sprite.



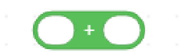
Example:



When the space key is pressed, all scripts will stop.

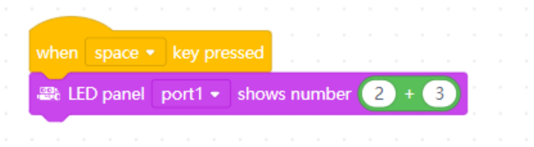
f. Operators

1. () + ()



Performs mathematical addition.

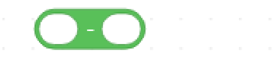
Example:



When the space key is pressed, the external LED panel will display the result of "2 + 3".

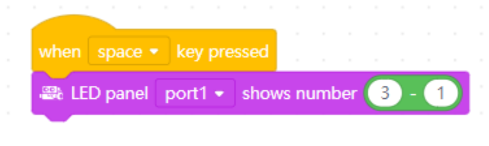
---

2. () − ()



Performs mathematical subtraction.

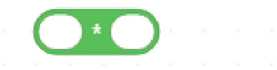
Example:



When the space key is pressed, the external LED panel will display the result of "3 - 1".

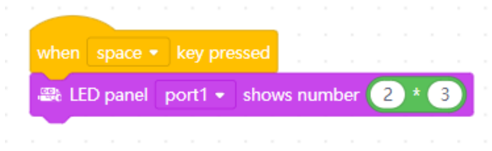
---

3. () * ()



Performs mathematical multiplication.

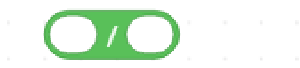
Example:



When the space key is pressed, the external LED panel will display the result of "2 × 3".

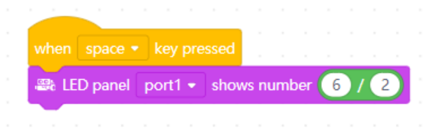
---

4. () / ()



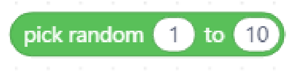
Performs mathematical division.

Example:



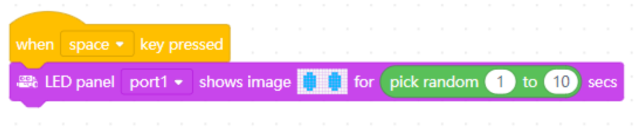
When the space key is pressed, the external LED panel will display the result of "6 ÷ 2".

5. pick random () to ()



Picks a random number from the specified range.

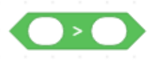
Example:



When the space key is pressed, the external LED panel will display the image for a random number of seconds within the range of 1 to 10.

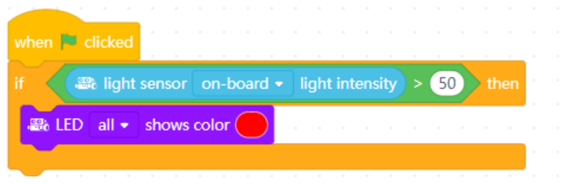
---

6. () > ()



If the value of the specified parameter is greater than the specified value, the report condition is met.

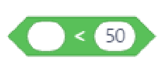
Example:



When the green flag is clicked, if the light intensity is greater than 50, all LEDs will light up red.

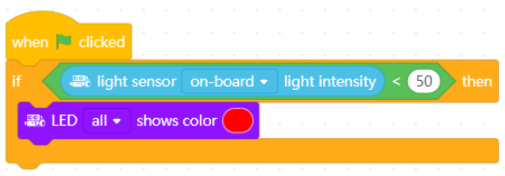
---

7. () < ()



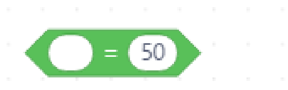
If the value of the specified parameter is less than the specified value, the report condition is met.

Example:



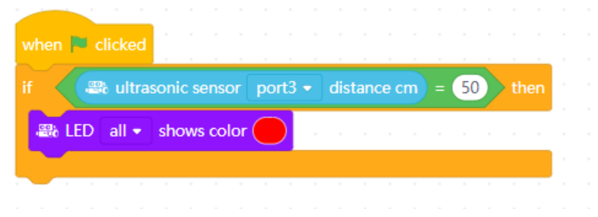
When the green flag is clicked, if the light intensity is smaller than 50, all LEDs will light up red.

8. () = ()



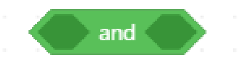
If the value of the specified parameter equals the specified value, the report condition is met.

Example:



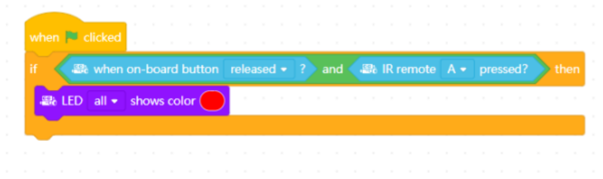
When the greed flag is clicked, if the distance of obstacle detected by the ultrasonic sensor equals 50 cm, all LEDs will light up red.

9. () and ()



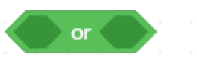
If both the conditions are met, the report condition is met.

Example:



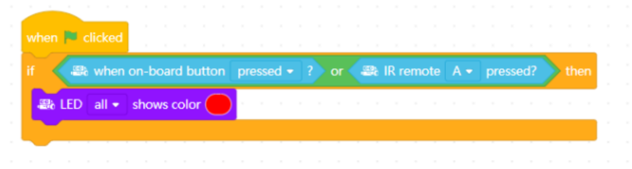
When the greed flag is clicked, if the on-board button is released, and button A of the IR remote is pressed, all LEDs will light up red.

10. () or ()



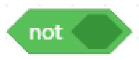
If either one of the two conditions is met, the report condition is met.

Example:



When the greed flag is clicked, if either the on-board button is pressed, or button A of the IR remote is pressed, all LEDs will light up red.

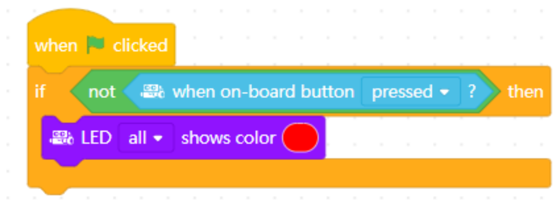
11. not ()



```
not < >
```

The report condition is met when the specified condition is not met.

Example:

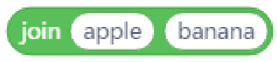


```
when 🏴 clicked
if not < when on-board button (pressed ▾) ? > then
    LED (all ▾) shows color ●
```

When the greed flag is clicked, if the on-board button is not pressed, all LEDs will light up red.

---

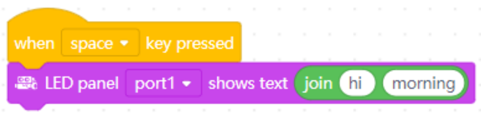
12. join () ()



```
join (apple) (banana)
```

Join two specified character strings.

Example:

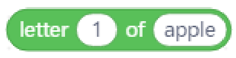


```
when (space ▾) key pressed
LED panel (port1 ▾) shows text (join (hi) (morning))
```

When the space key is pressed, the external LED panel will display "hi" and "morning" together.

---

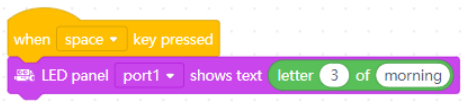
13. letter () of ()



```
letter (1) of (apple)
```

Report the letter at specified position of a character string.

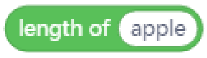
Example:



```
when (space ▾) key pressed
LED panel (port1 ▾) shows text (letter (3) of (morning))
```

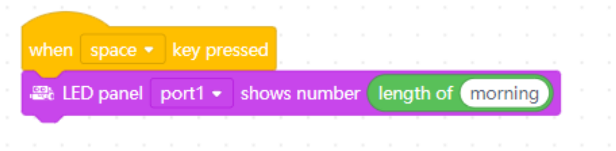
When the space key is pressed, the external LED panel will display the third letter of "morning".

14. length of ()



Report the length of a specified character string.

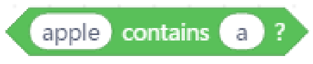
Example:



When the space key is pressed, the external LED panel will display the length of "morning."

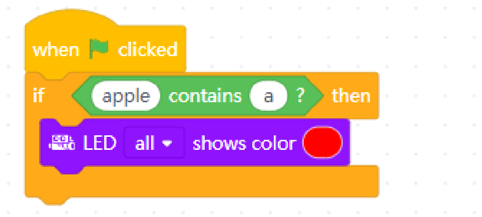
---

15. () contains ()?



If the specified character string contains the other specified character string, the report condition is met.

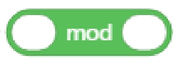
Example:



When the green flag is clicked, if "apple" contains "a", all LEDs will light up red.

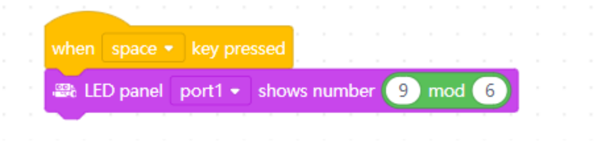
---

16. () mod ()



Calculate the remainder (modular) of two specified numbers.

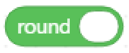
Example:



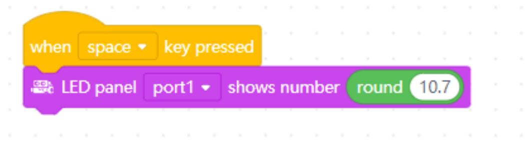
When the space key is pressed, the external LED panel will display the remainder of "9 ÷ 6".

17. round ()

Round the specified number to nearest integer.



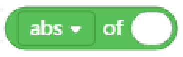
Example:



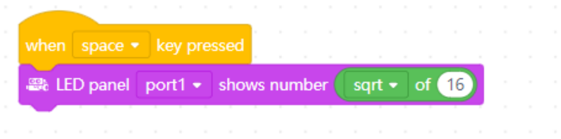
When the space key is pressed, the external LED panel will display the rounded value of 10.7.

18. (abs) ()

Perform specific mathematical operation on the specified number. Mathematical operations include: abs (absolute value), floor, ceiling, sqrt (square root), sin, sos, tan, asin, atan, acos, ln, log, e^, and 10^.



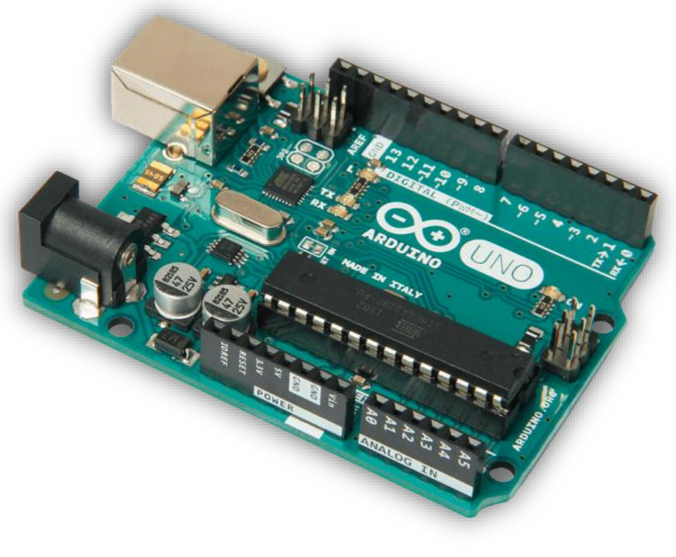
Example:



When the space key is pressed, the external LED panel will display the square root of 16.

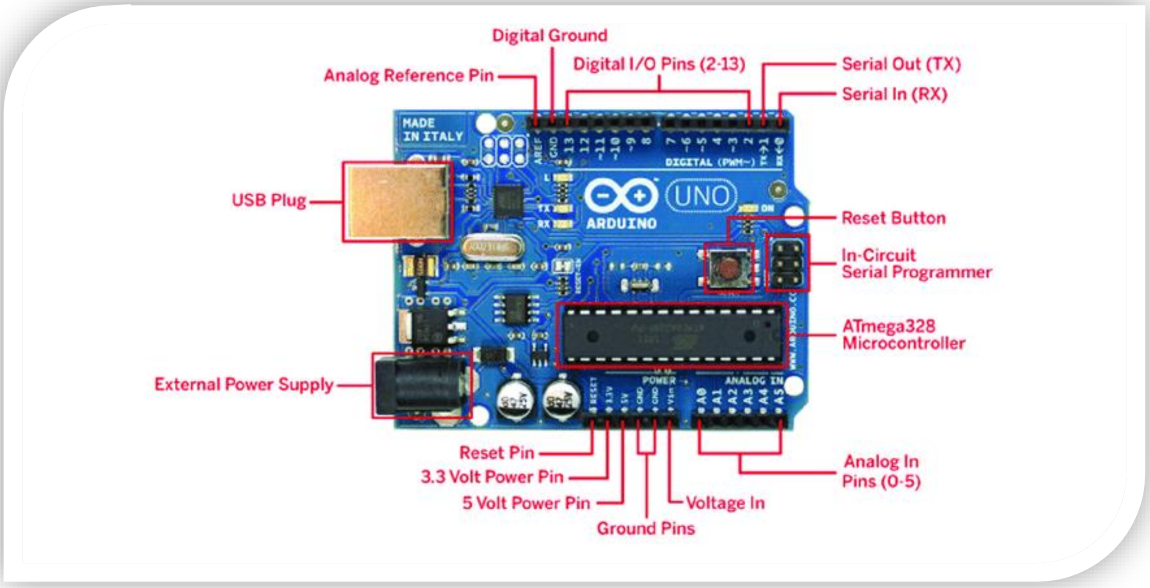
CHAPTER II

ARDUINO PROGRAMMING



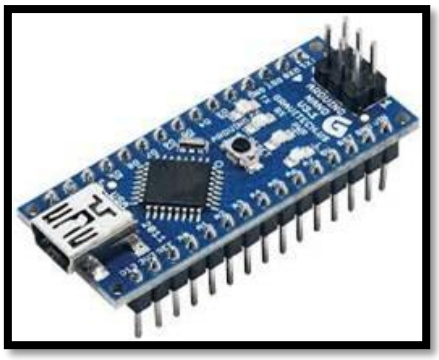
A. SOME ARDUINO MODELS

a. Arduino Uno (most used model)



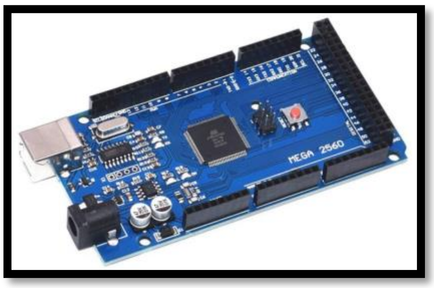
b. Arduino Nano

The preferred model in sensitive projects. (like drone projects)



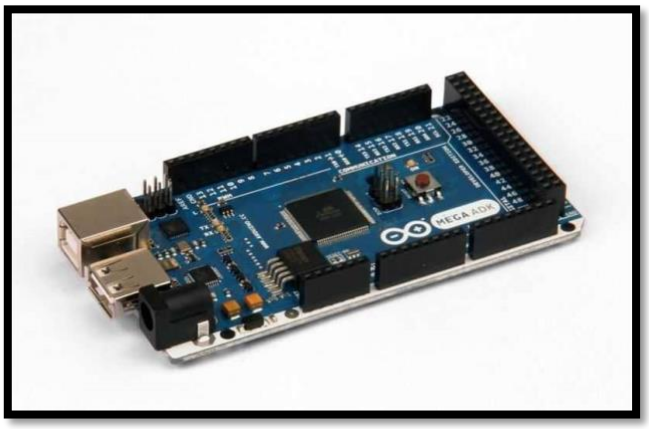
c. Arduino Mega

Model similar to Arduino UNO but used in larger projects



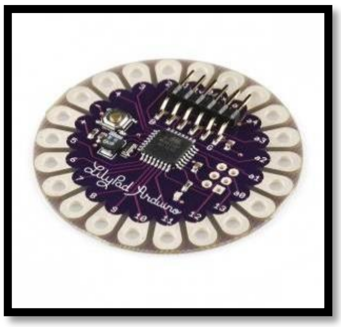
d. Arduino Mega ADK

Model used in programming Android devices

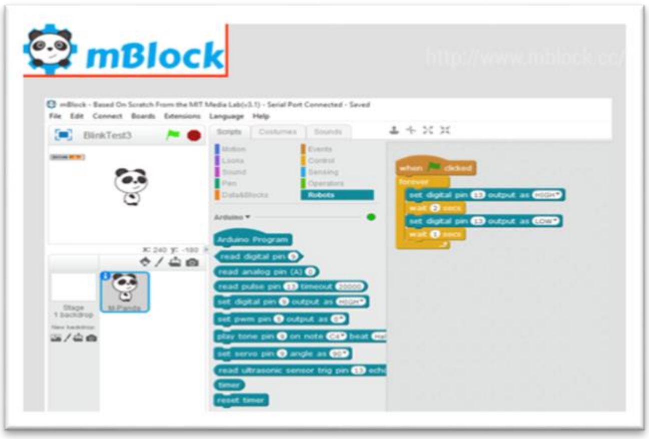


e. Arduino Lilypad

The model used in wearable Technologies



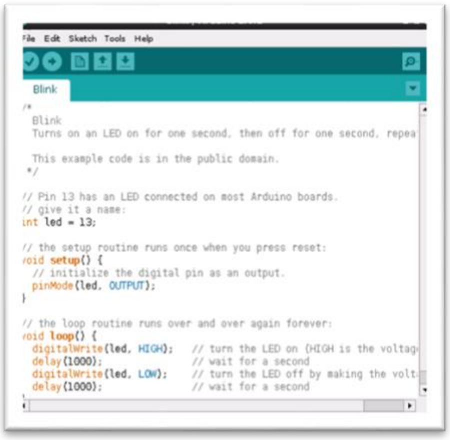
B. PROGRAMMING INTERFACES


mBlock
http://www.mblock.cc/
mBlock - Based On Scratch From the MIT Media Lab(v3.1) - Serial Port Connected - Saved
File Edit Connect Boards Extensions Language Help
BlinkTest3
Scripts Costumes Sounds
Motion Looks Sound Pen Data&Blocks
Events Control Sensing Operators Robots
Arduino
Arduino Program
read digital pin 9
read analog pin (A) 0
read pulse pin 13 timeout 20000
set digital pin 9 output as HIGH
set pwm pin 5 output as 0
play tone pin 9 on note C4 beat
set servo pin 9 angle as 90
read ultrasonic sensor trig pin 13
timer
reset timer
when clicked
forever
set digital pin 13 output as HIGH
wait 2 secs
set digital pin 13 output as LOW
wait 1 secs
Stage
1 backdrop
New backdrop:
M-Panda

https://mblock.makeblock.com/

AN OPEN PROJECT WRITTEN, DEBUGGED, AND SUPPORTED BY ARDUINO.CC AND THE ARDUINO COMMUNITY WORLDWIDE

LEARN MORE ABOUT THE CONTRIBUTORS OF ARDUINO.CC on arduino.cc/credits



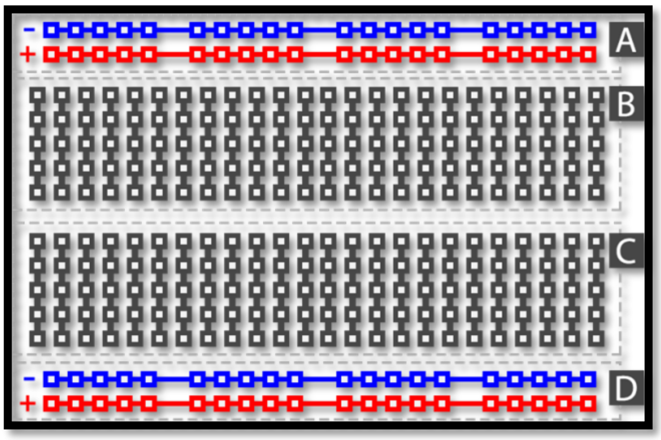
```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repea

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
oid setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
oid loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltag
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the volt
  delay(1000);               // wait for a second
```

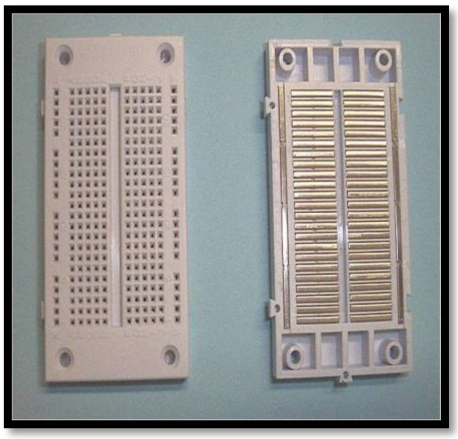
https://www.arduino.cc/

C. CIRCUIT ELEMENTS AND OTHER EQUIPMENT THAT CAN BE USED IN CIRCUIT DESIGN

a. Breadboard And Its Internal

We can make our projects work by placing electronic components on the breadboard that contains interconnected lines.





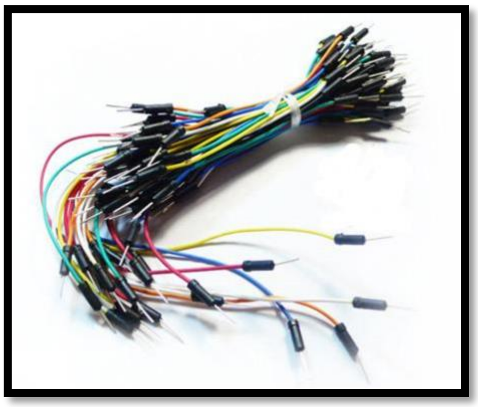
• Red and blue lines running horizontally above and below are generally used for voltage connections.

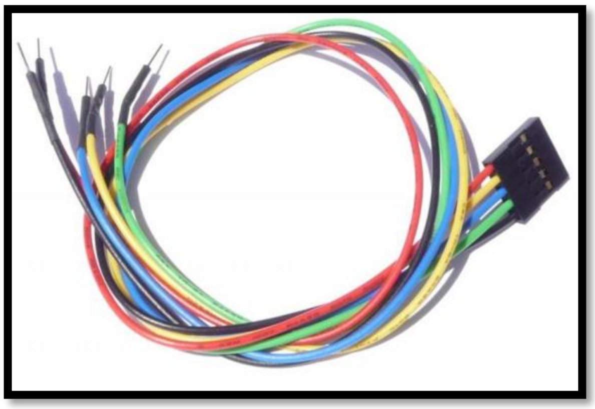
• You can connect the + line to the red line and the ground line to the blue line and then access the voltages (power) through these lines in other parts of your circuit.

• Each of the 5-hole groups in the middle section is connected within itself.

b. Jumper Cables

Cables that we can use in various applications you will make with breadboard.





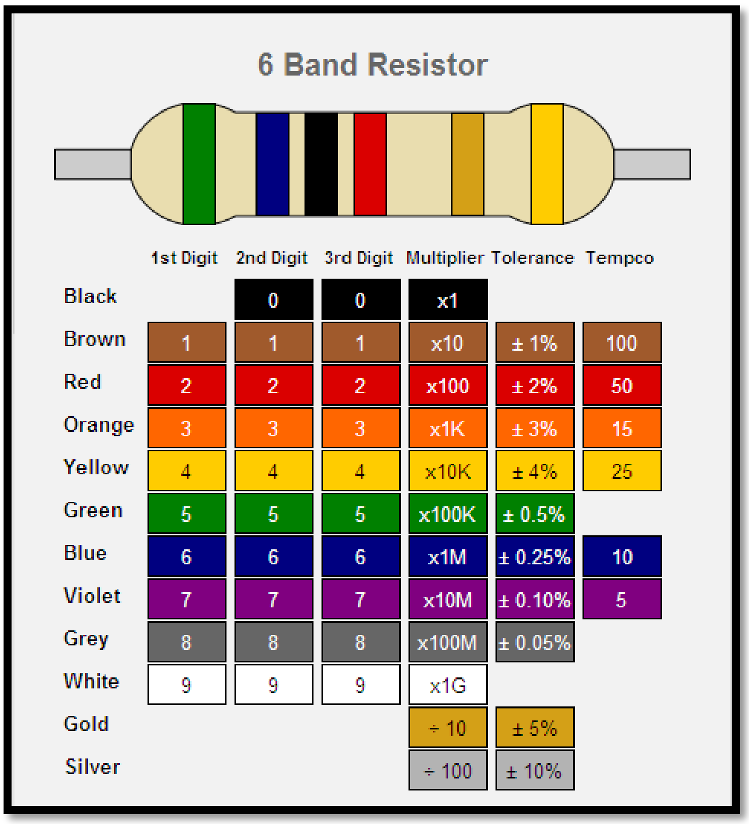
c. LED (Light Emitting Diode, Light Emitting Diode)

LEDs are one of the materials we use the most because of their low energy need, simplicity and nice look.

Operating voltages are between approximately 1.5 and 3.5 Volts and the value changes according to their colors.

d. Resistance

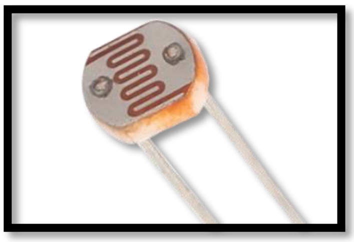
They are used to limit the current in electrical circuits and keep it at a certain value. It is also used to divide the supply voltage and current.



6 Band Resistor

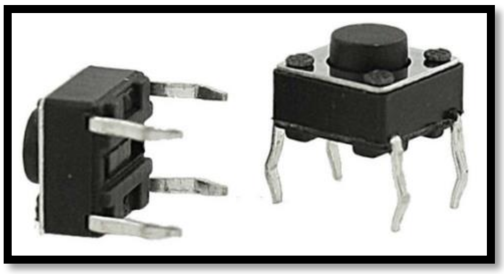
	1st Digit	2nd Digit	3rd Digit	Multiplier	Tolerance	Tempco
Black		0	0	x1		
Brown	1	1	1	x10	± 1%	100
Red	2	2	2	x100	± 2%	50
Orange	3	3	3	x1K	± 3%	15
Yellow	4	4	4	x10K	± 4%	25
Green	5	5	5	x100K	± 0.5%	
Blue	6	6	6	x1M	± 0.25%	10
Violet	7	7	7	x10M	± 0.10%	5
Grey	8	8	8	x100M	± 0.05%	
White	9	9	9	x1G		
Gold				÷ 10	± 5%	
Silver				÷ 100	± 10%	

e. LDR (Light Dependent Resistor, Light Dependent Resistor)



As the intensity of light falling on it increases, its resistance value decreases, and as the light intensity decreases, its resistance value increases.

f. Push Buton



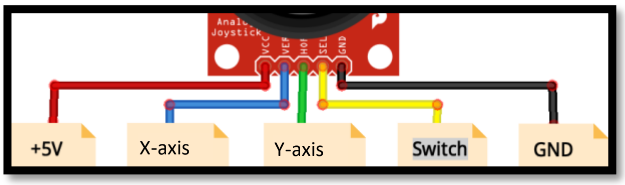
Push buttons are circuit elements that are ON only when pressed and OFF when released. One push button can do the job of two keys alone. When the button is pressed, pins 1 and 3 are connected to each other. Similarly, pins 2 and 4 are connected to each other and the circuit is completed.

g. Buzzer

Buzzer is an equipment used to produce sound in Arduino circuits.

h. Joystick

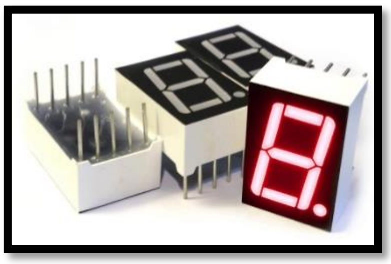
The joystick is a module that contains two potentiometers that allow movement in two axes and acts as a button in its vertical movement.



i. RGB Led

RGB (Red-Green-Blue) LED is a type of LED that contains red, green and blue colors. It is frequently used especially in animation and lighting systems. RGB LEDs have specific ranges for each color. Thanks to these ranges, it is possible to obtain many colors.

j. 7 Segment Display



Each of the 7 LEDs in the structure of the element is called a segment. The 7 spatially partitioned regions are labeled with 7 letters (a,b,c,d,e,f,g) between the letters "a" and "g". There are two types, Common Anode and Common Cathode.

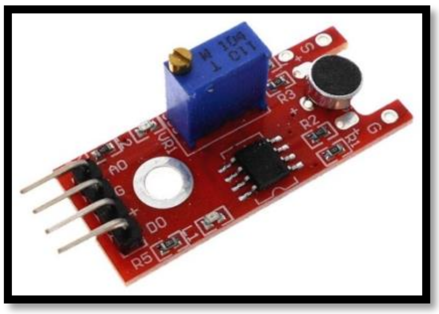
k. Potentiometer

A potentiometer is a type of resistor. But the biggest feature that distinguishes it from other resistor types is that the resistor value can be changed.

l. Sensors

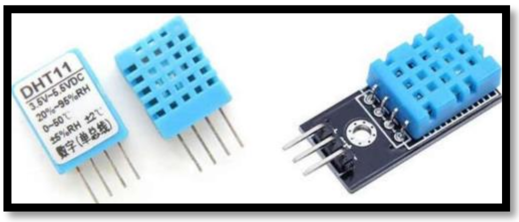
There are many sensors and modules that can be used when developing projects with Arduino. Here we will touch on commonly used sensors.

i. Sound Sensor Module (Ky-038)



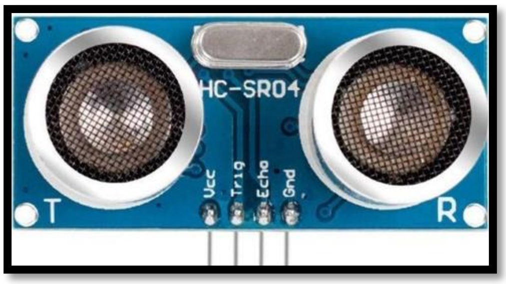
Sound Sensor; It is a sensor that can detect the sound in the environment and give both analog and digital output. It detects sounds with the microphone on it and can be adjusted with the trimpot on the card..

ii. Heat And Humidity Sensor (DHT 11)



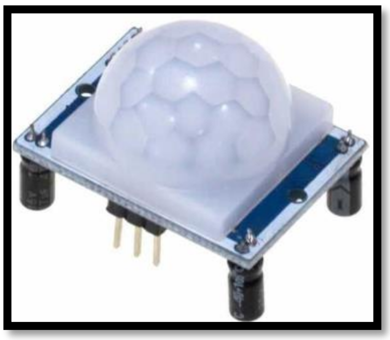
The DHT 11 sensor is an effective sensor with a digital temperature and humidity sensor. It is an advanced sensor that outputs a calibrated digital signal.

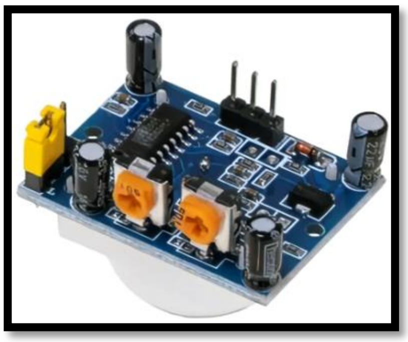
iii. Ultrasonic Distance Sensor (HC-SR04)



The HC-SR04 Ultrasonic Sensor is a source that calculates the distance to the opposite object using sonar communication. The system we call sonar helps us calculate the distance of the object using sound waves.

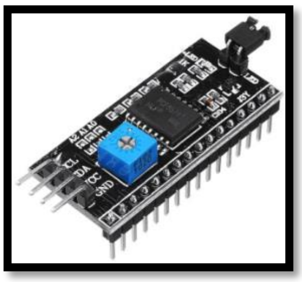
iv. Pir Motion Sensor (HC-SR501)





The PIR sensor works by measuring the IR (Infra Red) light emitted by the people and warm-blooded creatures around it. If it sees movement, it outputs 1. If it doesn't, it stays 0. It has two adjustment potentiometers on it. Delay time and measuring range are adjusted with these potentiometers.

m. LCD Liquid Crystal Displays

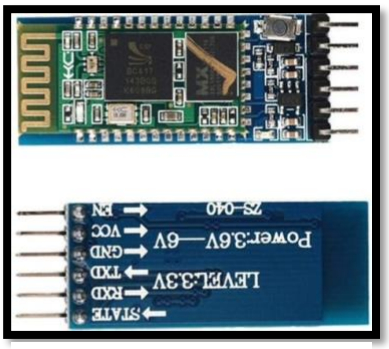


The Liquid Crystal display, which is called 1602 lcd and consists of 16 columns and 2 lines, has a green or blue backlight. It has 16 pins on it. Also available with I2C card type. This type has 4 pins. These are SDA, SCL, GND and VCC.

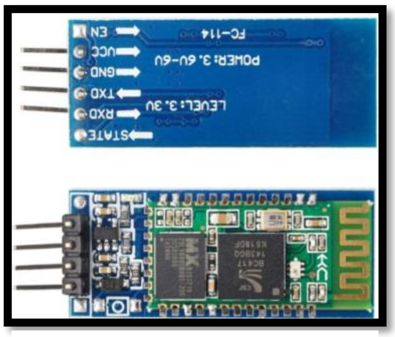
n. PWM Controlled Servo Motor (SG 90)

In the SG90, we can send data from a single channel to get it to the angle we want, and we can rotate the angle strongly and precisely whenever we want. The PWM signal is a type of signal created by quickly cutting and opening the electrical signal. Servo motor has 3 terminals. Two of them provide energy to the motor, while the other end reports the position of the motor shaft to the input unit of the system.

o. Bluetooth Module (HC-05 and HC-06)

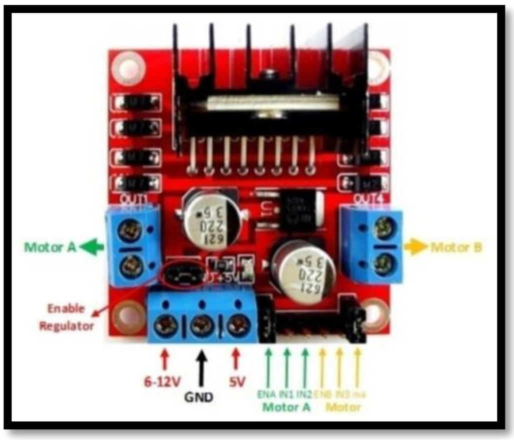

HC-06

Bluetooth Module is a wireless communication module that provides wireless communication and uses the Bluetooth protocol for wireless communication.


HC-05

The only difference between them in terms of features is that the **HC05** module can simultaneously connect to other bluetooth devices while responding to incoming requests. On the other hand, **HC06** can only accept incoming requests, but cannot send a connection request to another bluetooth card..

p. L298N Motor Driver



This motor driver board, which is prepared to drive motors up to 24V, has two channels and gives 2A current per channel. L298N motor driver IC is used on the board.

It can control two separate motors independently of each other.

It can give 2A current per channel. It has a built-in regulator.

It has high temperature and short circuit protection. There are leds that light up according to the motor rotation direction. There is a built-in cooler on the board..

D. APPS

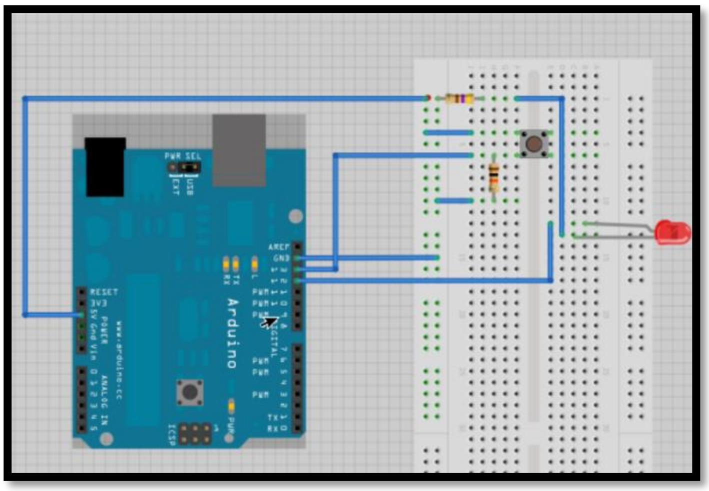
a. Turn On Led With Button

Description: Connect the 5V and gnd terminals on the Arduino to the board. Connect the anode lead of the led to the pin 12 of the arduino and the cathode lead to the gnd with a 470 Ω resistor.

Materials Used in the Circuit

- 1 x 470 Ω
- 1 x 10 KΩ
- 1 x push button
- 1 x led
- 1 x Arduino UNO

i. Circuit diagram:



ii. Circuit Software:

```
const int butonPin = 13;
const int ledPin = 12;
int buttonDurumu = 0;
void setup() {

 pinMode(butonPin, INPUT);

  pinMode(ledPin, OUTPUT);
}
 void loop() {

         buttonDurumu = digitalRead(butonPin);



     if (buttonDurumu == HIGH) {
     digitalWrite(ledPin, HIGH);
  }
  else {
    digitalWrite(ledPin, LOW);
  }
}
```

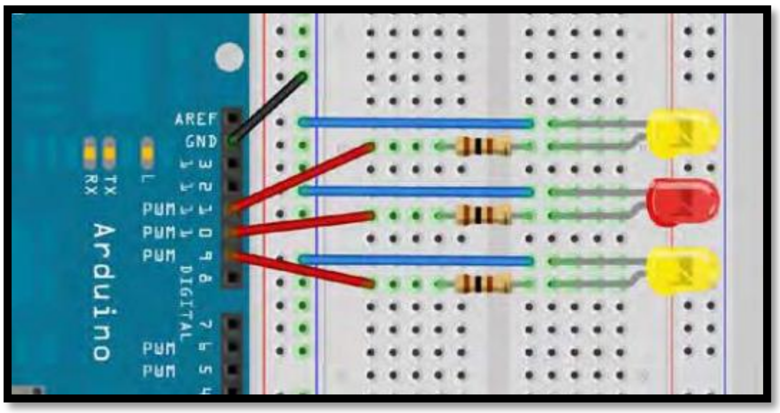
b. Flame Effect

Explanation: Via PWM, we will give the LEDs a random flickering light effect. For example, in a decorative fireplace in your home, movies, stage plays, model railways, etc. You can use this

Materials Used in the Circuit

- 1 X Arduino UNO
- 1 X Kırmızı LED
- 2 X Sarı LED

i. Circuit diagram:



ii. Circuit Software:

```
int ledPin1 = 9;
int ledPin2 = 10;
int ledPin3 = 11;
void setup()
{
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
  pinMode(ledPin3, OUTPUT);
  }
  void loop()
  {

   analogWrite(ledPin1, random(120)+135);
   analogWrite(ledPin2, random(120)+135);
   analogWrite(ledPin3, random(120)+135);
   delay(random(100));
   }
```

c. Light Intensity Control with PWM

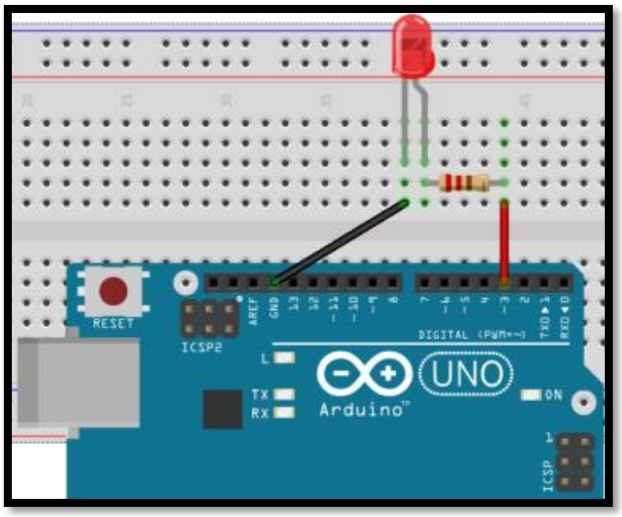
Explanation: The pin to which the LED will be connected is PWM pin 3. With the for loop in the loop method, the counter value i starts from 0 and counts up to 255. The LED continues this process by

Materials Used in the Circuit

- 1 X Arduino UNO
- 1 X

led

i. Circuit diagram:

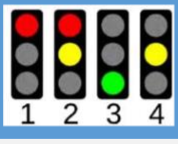


ii. Circuit Software:

```
const int led=3;
void setup()
{
  pinMode(led,OUTPUT);
  }
  void loop()
  {
    for (int i=0;i<=255;i++)
    {
      analogWrite(led,i);
      delay(10);
     }
   }
```

d. Traffic Light

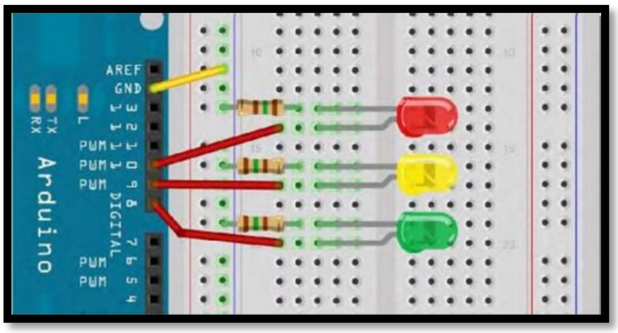
Explanation: The flashing time of the yellow led is 2 seconds. You can change the LED durations as you wish by typing in the "delay(.....)" command. The LEDs work in the following order.



Materials Used in the Circuit

- 1 X Arduino UNO
- 3 X led (Red, Yellow, Green)
- 3 x 150 Ω

i. Circuit diagram:



ii. Circuit Software:

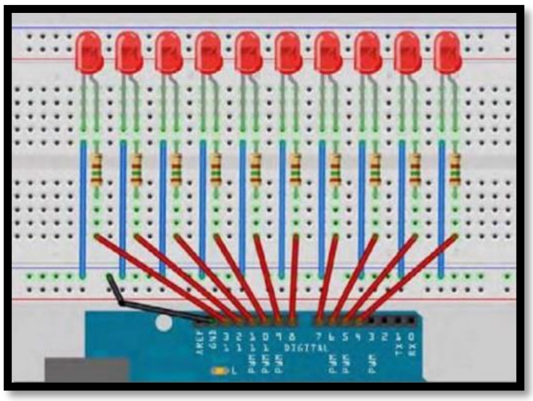
```
int ledDelay = 10000;
int kirmiziPin = 10;
int sariPin = 9;
int yesilPin = 8;

void setup() {
  pinMode(kirmiziPin, OUTPUT);
  pinMode(sariPin, OUTPUT);
  pinMode(yesilPin, OUTPUT);
  }
void loop() {
  digitalWrite(kirmiziPin, HIGH);
  delay(ledDelay);
  digitalWrite(sariPin, HIGH);
  delay(2000);

 digitalWrite(yesilPin, HIGH);
 digitalWrite(kirmiziPin, LOW);
 digitalWrite(sariPin, LOW);
 delay(ledDelay);


digitalWrite(sariPin, HIGH);
digitalWrite(yesilPin, LOW);
delay(2000);
digitalWrite(sariPin, LOW);
 }
```

e. Follower 10 LEDs

Materials Used in the Circuit

- 1 X Arduino UNO
- 10 X led
- 10 X 150 Ω

i. Circuit diagram:



ii. Circuit Software:

```
byte ledPin[] = {4, 5, 6, 7, 8, 9, 10, 11, 12, 13};


int ledDelay(65);
int yon = 1;
int currentLED = 0;
unsigned long degismeZamani;

void setup() {
for (int x=0; x<10; x++) {
pinMode(ledPin[x], OUTPUT); }
degismeZamani = millis();
}
void loop() {
if ((millis() - degismeZamani) > ledDelay) {
changeLED();
degismeZamani = millis();
}
}
void changeLED() {
for (int x=0; x<10; x++) {
digitalWrite(ledPin[x], LOW);
}
digitalWrite(ledPin[currentLED], HIGH);
currentLED += yon;
if (currentLED == 9) {yon = -1;}
if (currentLED == 0) {yon = 1;}
}
```

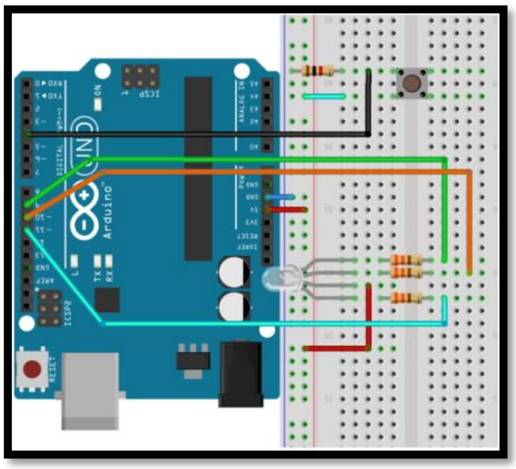
f. Rgb Led Applications

Explanation: The RGB LED anodes we use are the 3 LEDs connected together in a single case. It is possible to digitally control the light intensity of three colors. In addition, the desired colors can be obtained by using the PWM technique.

Materials Used in the Circuit

- 1 X Arduino uno
- 3 X 330 Ω
- 1 X 1 KΩ
- 1 X RGB led

i. Circuit diagram:



ii. Circuit Software:

```
const int MaviLed=11;
const int YesilLed=10;
const int KirmiziLed=9;


void setup()
{
pinMode(MaviLed,OUTPUT);
pinMode(YesilLed,OUTPUT);
pinMode(KirmiziLed,OUTPUT);

}


void loop()
{


digitalWrite(MaviLed,HIGH);
digitalWrite(YesilLed,HIGH);
digitalWrite(KirmiziLed,LOW);
  delay(1000);
```

```
digitalWrite(MaviLed,HIGH);
digitalWrite(YesilLed,LOW);
digitalWrite(KirmiziLed,HIGH);
  delay(1000);


digitalWrite(MaviLed,LOW);
digitalWrite(YesilLed,HIGH);
digitalWrite(KirmiziLed,HIGH);
delay(1000);


digitalWrite(MaviLed,LOW);
digitalWrite(YesilLed,LOW);
digitalWrite(KirmiziLed,LOW);
delay(1000);
}
```

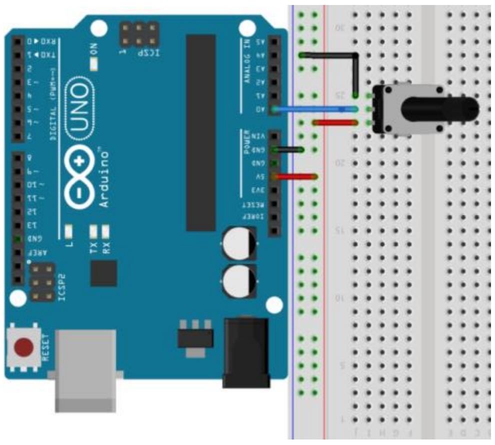
g. Reading Value From Potentiometer

Açıklama: The potentiometer works as a voltage divider in this application with varying resistance value. We will use the analogRead() command to convert the analog values received from the setting tip to digital. We will transfer every value of the pot to the PC environment and observe it on the serial monitor. Since the value read is 10 bits, it takes 210 = 1024 different values. When the input voltage is 5V, the value of the generated digital data is 1023. The value read every 1 second is sent to the PC. To view these values, simply click the " Seri Port Ekranı " button in the Arduino IDE program.

Materials Used in the Circuit

- 1 X Arduino UNO
- 1 X 100 KΩ POT

i. Circuit diagram:



ii. Circuit Software:

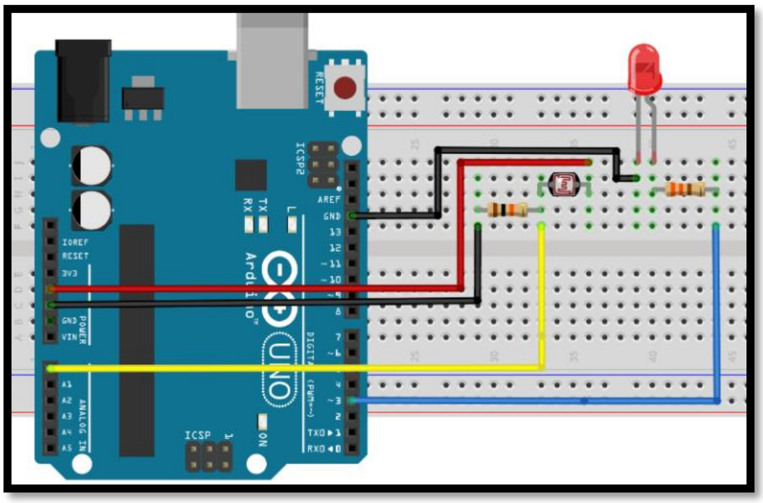
```
const int POT=0;
int deger=0;
void setup()
{
   Serial.begin(9600);
}

void loop()
{
   deger=analogRead(POT);


   Serial.println(deger);


   delay(1000);
}
```

h. LDR Circuit with LED in the Dark

Materials Used in the Circuit

- Arduino UNO
- 1 x 1KΩ Resistor
- 1 x 220Ω Resistor
- LDR
- 1 x LED

i. Circuit diagram:



ii. Circuit Software:

```
#define led 3
void setup() {
pinMode(led, OUTPUT);
Serial.begin(9600);
}
void loop() {
int isik = analogRead(A0);
Serial.println(isik);

delay(50);
if (isik > 900) {
  digitalWrite(led,LOW);
  }

if (isik < 850) {
  digitalWrite(led,HIGH);
  }
}
```

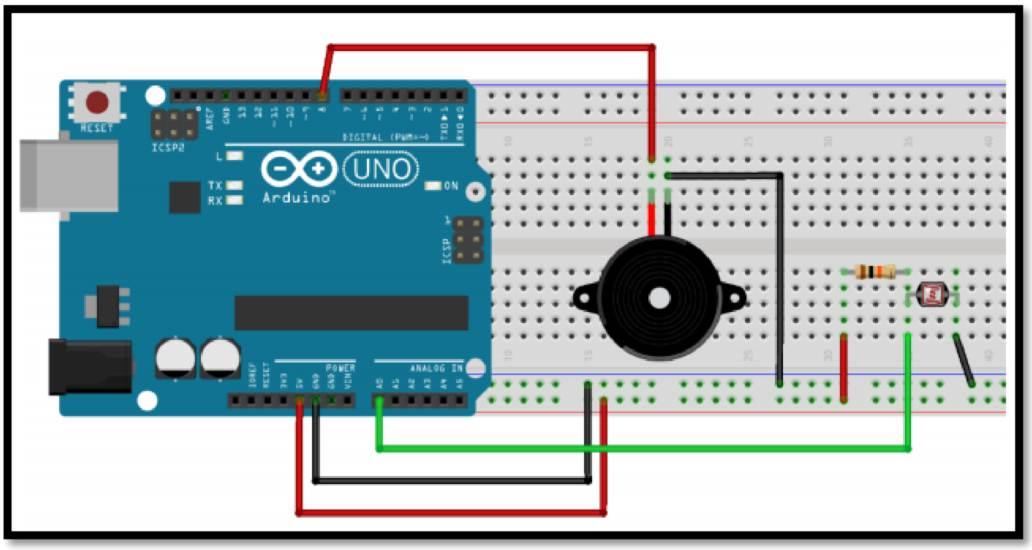
i. Audio Feedback with Light Effect

Explanation: In this application we will use LDR to detect light and piezo sounder to get audible feedback based on the amount of light detected.

Materials Used in the Circuit

- 1 X Arduino UNO
- 1 X Piezo Sounder or Buzzer
- 1 X LDR (Light Dependent Resistor)

i. Circuit diagram:



ii. Circuit Software:

```
int piezoPin = 8;
int ldrPin = 0;
int ldrValue = 0;
void setup() {
}
void loop() {
ldrValue = analogRead(ldrPin);
tone(piezoPin,1000);
delay(25);
noTone(piezoPin);
delay(ldrValue);
}
```

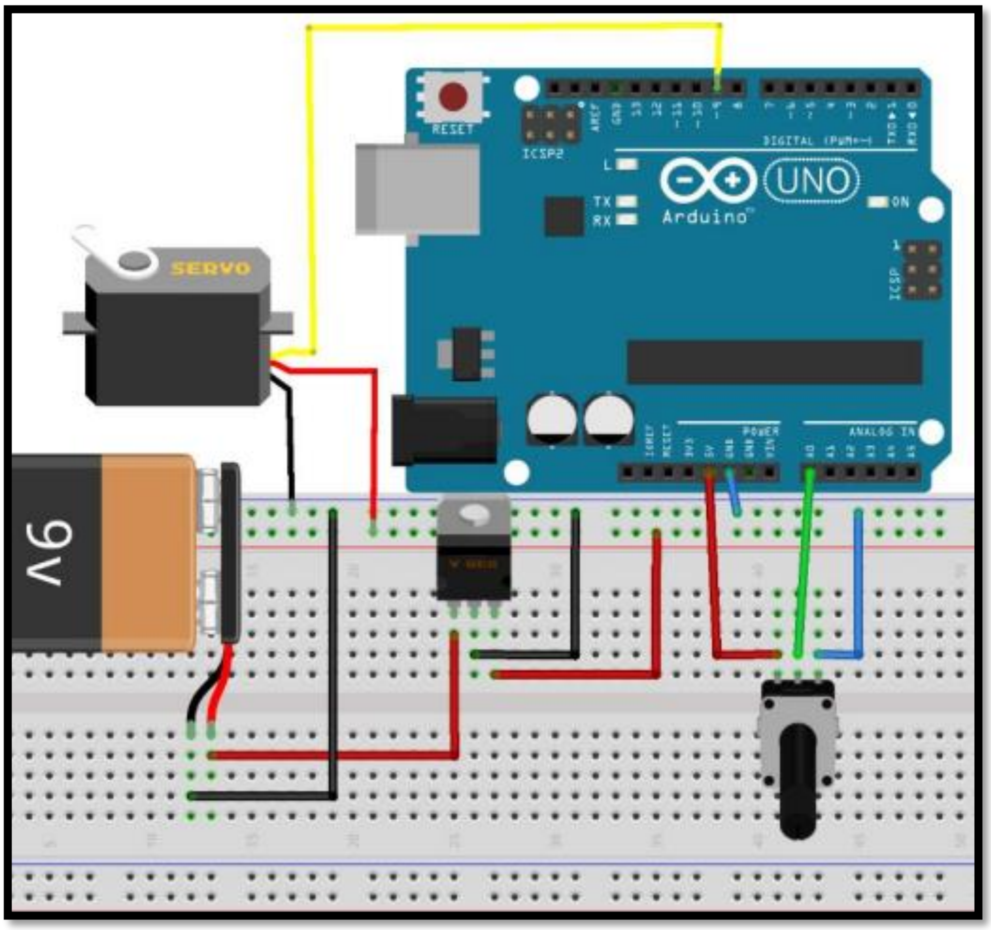
j. Servo Motor Applications

Explanation: The type of motor we use in our application moves angularly according to the given command and can stay in the position we want. They have 3 connecting pins. These pins are GND (brown, black), 5V (red) and signal terminal (white or orange). We will use the 7805 regulator IC since we will supply the motor from an external power source in our circuit. In the circuit below, the values corresponding to the analog data between 0 and 1023 read from the potentiometer are limited between 0 and 179 degrees with the map method. In this way, as the value of the potentiometer changes, the servo motor moves between 0 - 180 degrees.

Materials Used in the Circuit

- 1 X Arduino UNO
- 1 X 7805
- 9 Volt battery
- Servo engine

i. Circuit diagram:



ii. Circuit Software:

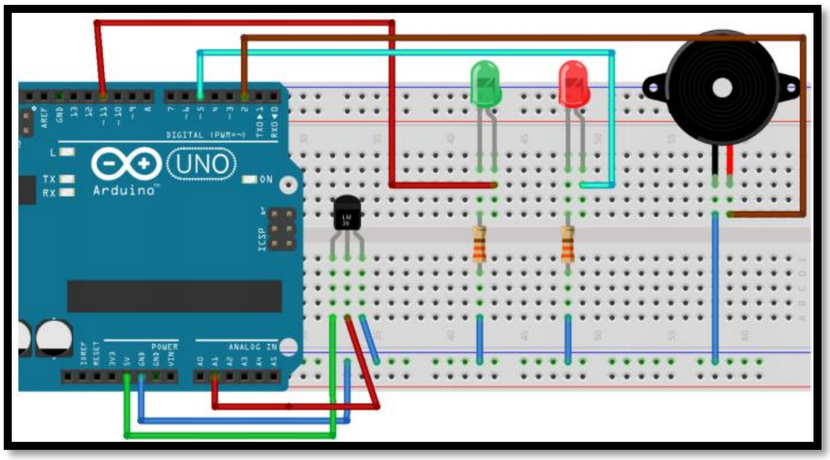
```
#include <Servo.h>
const int ServoPals = 9;
const int Pot=0;
Servo servoKontrol;
int deger=0;
void setup()
{
  servoKontrol.attach(ServoPals);

}
void loop()
{
  deger=analogRead(Pot);
  deger=map(deger,0,1023,0,179);

  servoKontrol.write(deger);

  delay(15);
}
```

k. Fire Alarm Application

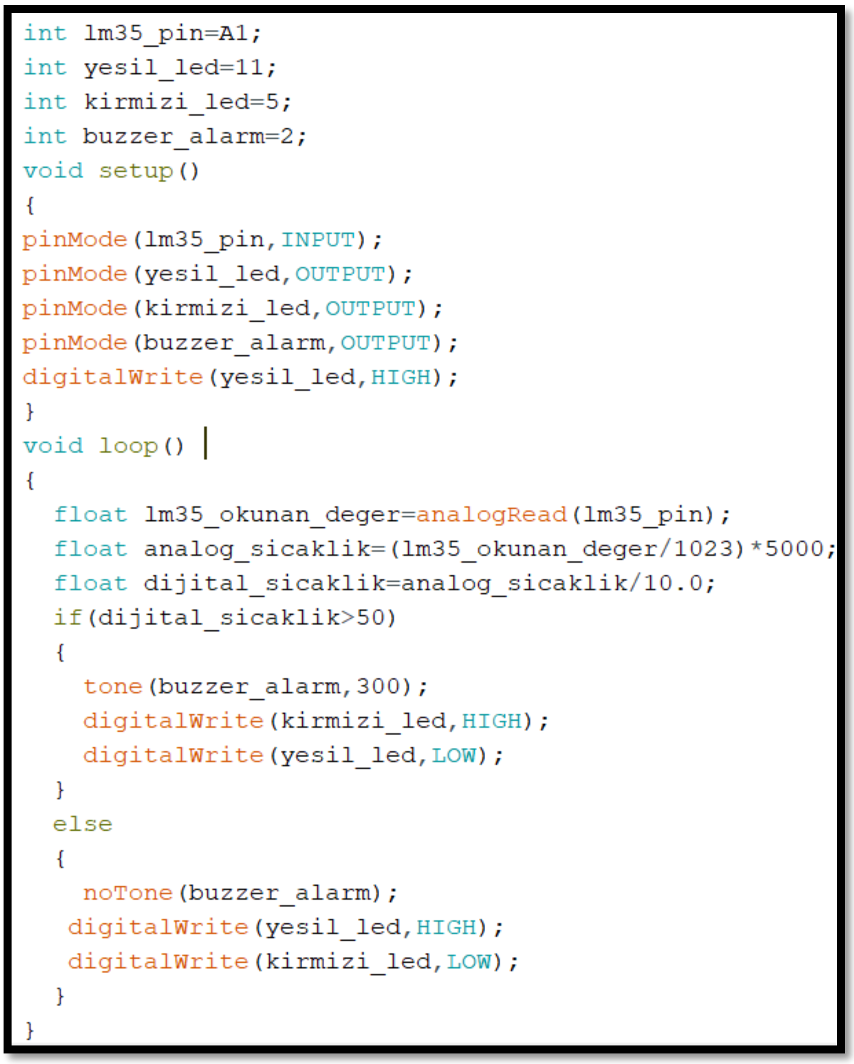
Materials Used in the Circuit

- 1 X Arduino UNO
- 1 X LM35
- 2 X Led (Red, Green)
- 2 X 330 Ω Resistance

i. Circuit diagram:



ii. Circuit Software:



```
int lm35_pin=A1;
int yesil_led=11;
int kirmizi_led=5;
int buzzer_alarm=2;
void setup()
{
pinMode(lm35_pin,INPUT);
pinMode(yesil_led,OUTPUT);
pinMode(kirmizi_led,OUTPUT);
pinMode(buzzer_alarm,OUTPUT);
digitalWrite(yesil_led,HIGH);
}
void loop()
{
  float lm35_okunan_deger=analogRead(lm35_pin);
  float analog_sicaklik=(lm35_okunan_deger/1023)*5000;
  float dijital_sicaklik=analog_sicaklik/10.0;
  if(dijital_sicaklik>50)
  {
    tone(buzzer_alarm,300);
    digitalWrite(kirmizi_led,HIGH);
    digitalWrite(yesil_led,LOW);
  }
  else
  {
    noTone(buzzer_alarm);
   digitalWrite(yesil_led,HIGH);
   digitalWrite(kirmizi_led,LOW);
  }
}
```

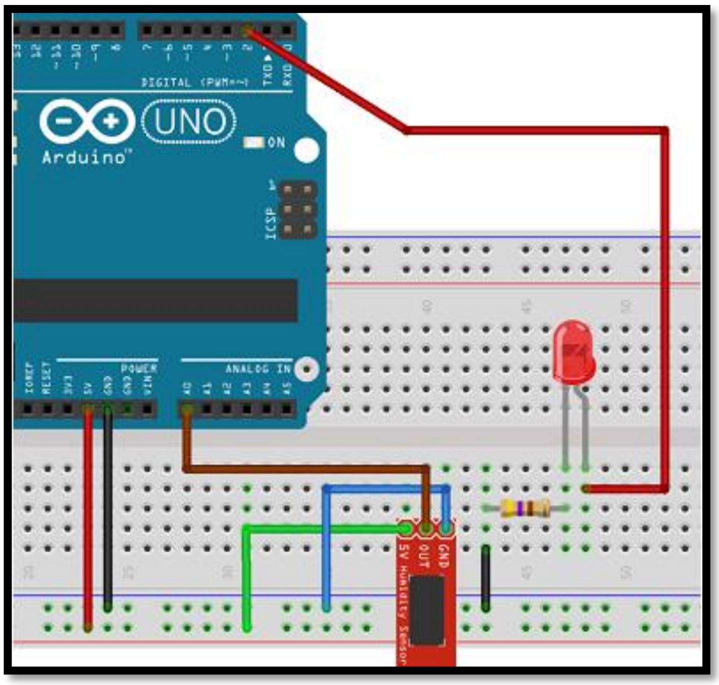
l. Water Level Measurement Application

Explanation: The working logic of the water level sensor; While there is no electrical connection between the copper plates under normal conditions, when it starts to work in water, the electric current passing over the copper plates increases according to the water level between the copper plates, so an analog voltage varying according to the water level is obtained from the output end. This voltage varies between 0– 3.3 Volts.

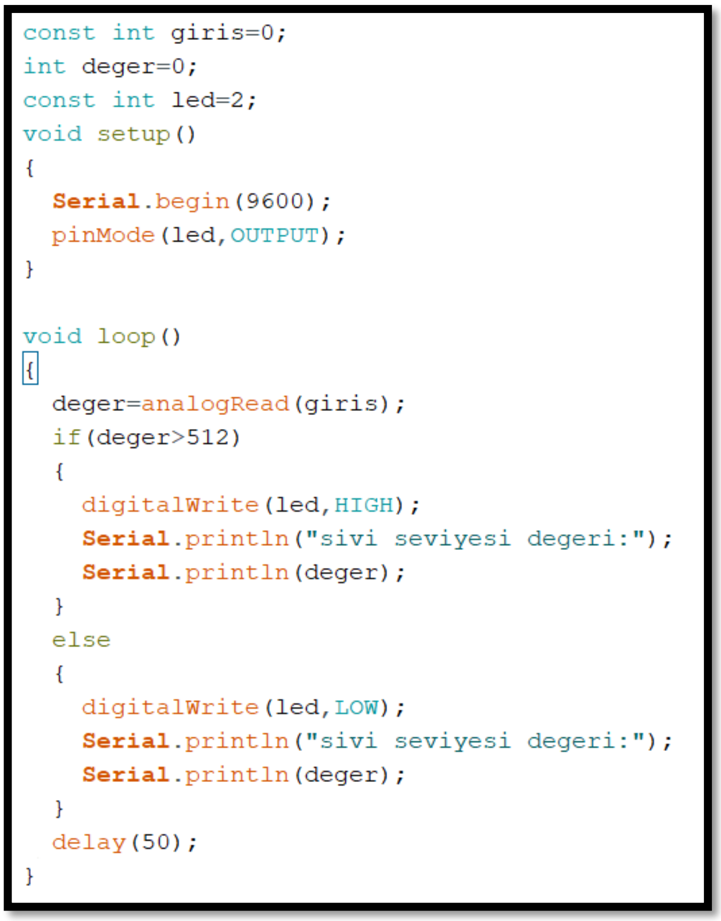
Materials Used in the Circuit

▪ 1 X Arduino UNO ▪ 1 X Led

i. Circuit diagram:



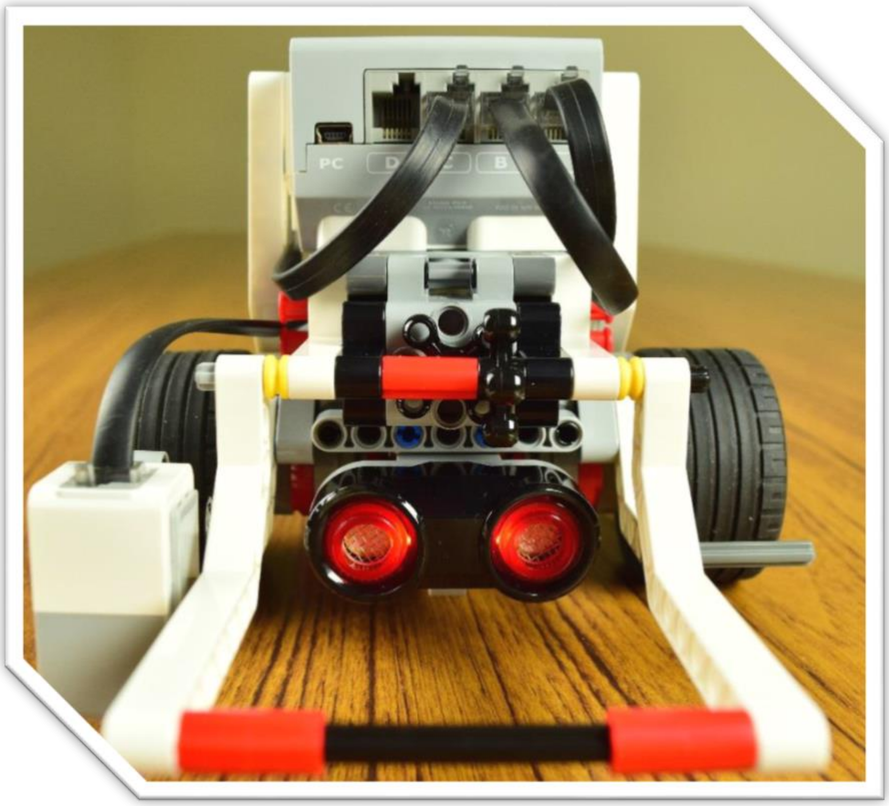
ii. Circuit Software:



```
const int giris=0;
int deger=0;
const int led=2;
void setup()
{
  Serial.begin(9600);
  pinMode(led,OUTPUT);
}

void loop()
{
  deger=analogRead(giris);
  if(deger>512)
  {
    digitalWrite(led,HIGH);
    Serial.println("sivi seviyesi degeri:");
    Serial.println(deger);
  }
  else
  {
    digitalWrite(led,LOW);
    Serial.println("sivi seviyesi degeri:");
    Serial.println(deger);
  }
  delay(50);
}
```

CHAPTER II

LEGO Mindstorms EV3 Programming Basics



The Lego Mindstorms Robotics system, which includes the EV3 Programming Software, can be as advanced or as basic as you'd like it to be. But for our purposes, we'll just be covering the essentials of the system to demystify some programming concepts and set the foundation for building the imagination and creativity that are fundamental to Science, Technology, Engineering, Arts, and Math; also known as STEAM.

Through this introduction into programming and robotics, you will learn the thought process behind creating a program, basic programming functions, and how they relate to robotic actions and reactions.

A. **You will also need:**

1. 1. A flat and smooth surface area of at least 3’ X 3’ (or close to it) for the robot to run its programs freely.
    a. Most of the exercises can be confined to a smaller area and some can even run while connected to the computer, so the average portable table top could work.
    b. If floor space is your only option, a smooth floor is best, but a very low and tight-threaded carpet will suffice for most exercises.
        i. Carpet will not work very well for any exercises using the Color Sensor.
    c. Having both floor types available is a great way to test/illustrate movement variables.
2. Electrical tape (or something similar) of at least one basic color that will contrast with the surface area.
    a. Tan masking tape will NOT work (inconsistent color readings), but the blue or green should.
    b. Having multiple colors is a great way to test variables and possibilities.
3. To charge the battery (does not apply if using AA’s).
    a. In theory, the battery is already attached to the “Brick,” (the robot’s computer/control center) so it’s just a matter of plugging the charger into it.
    b. A fully charged battery will display one green light while still plugged into an outlet. During charging there *may* be just a red light to start, but normally it’s both a green and a red light. A full charge could take a few hours depending on its current state.

B. The main LEGO Mindstorm components used for our robot:

1. The computer/control center…a.k.a. the "Brick"

- **a.** Programs are executed from the Brick.
- **b.** Programs are downloaded to, or created from the Brick.
  NOTE: directly programming on the brick using its interface is possible, but a bit cumbersome, and will not be covered in this tutorial. There are examples of this function in the EV3 instruction manual for your reference.
- **c.** Sends programmed information to motors.
- **d.** Receives information from sensors to initiate programmed parameters.
- **e.** Must be connected to Motors and Sensors via cabling in order to function.
- **f.** Motors MUST be connected to the front alpha ports.
  NOTE: alpha ports send information out.
- **g.** Sensors MUST be connected to the rear numeric ports.
  NOTE: numeric ports receive information in.
- **h.** Plays programmed sounds and displays.
- **i.** Connects to the computer via PC Port that's located near the Alpha Ports.
- **j.** See "Brick Overview" on page 8 for more information.

2. Large Motor

- **a.** Receives programmed instructions from the Brick.
- **b.** Heavy duty with lower gearing for mobility requirements.

3. **Medium Motor**
    a. Receives programmed instructions from the Brick.
    b. Used primarily for moving parts of a robot.
    c. Light duty with higher gearing for quick response; **NOT:** suitable for mobility.

4. **Color Sensor**
    a. Detects color differences and sends that data to the Brick for possible action according to programmed parameters.
    b. Emits a set of color wavelengths. The color wavelength that is reflected back, and not absorbed, determines the color of the surface.
    NOTE: sometimes, what appears to be a certain color to our eyes, may reflect differently to the sensor and cause unexpected results.

5. **Ultrasonic Sensor**
    a. Detects distance to an object and sends that data to the Brick for possible action according to programmed parameters.
    b. Using the same principle as Bats and submarines, it uses echo location by emitting an ultrasonic wave that is received back after bouncing off an object. The time it takes to be received back determines the distance to that object.

C. A few other LEGO Mindstorm components…NOT used for our robot:

1. Touch Sensor

- **a.** Sends the state of either being pressed, not pressed, or bumped (pressed, then not pressed) to the Brick for possible action according to programmed parameters.
- **b.** Often used to determine if the robot has physically bumped into something.

2. IR (infrared) Sensor *and* IR Beacon

- **a.** The IR Sensor detects proximity to other objects much like the Ultrasonic Sensor and sends that data to the Brick for possible action according to programmed parameters.

- **b.** Also detects IR signals from the Beacon.
- **c.** Beacon can be used as a hand-held remote control.

3. Gyro Sensor

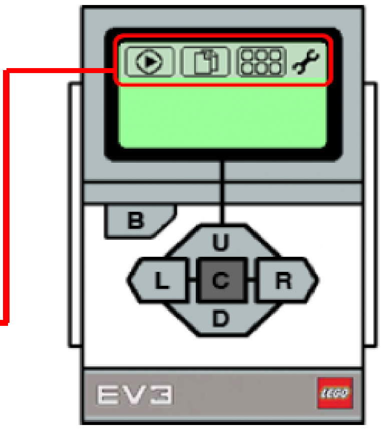
Detects the robots orientation and rotational motion and sends that data to the Brick for possible action according to programmed parameters.

D. Brick Overview:

As mentioned earlier, the Brick is the control center of your robot's functionality. All your programs are downloaded to it and run from it. All your connected Motors are sent commands from it, and all your connected Sensors provide information to it for strict interpretation as applied to the program being run. Also mentioned is that Motors MUST be connected to the Alpha ports, and Sensors MUST be connected to the Numeric ports for them to function. That's the basics of what the Brick does, so now let's learn how to turn it on, and navigate the interface.

NOTE: Button lettering is informational only; they are not actually on the Brick buttons.

The Buttons:

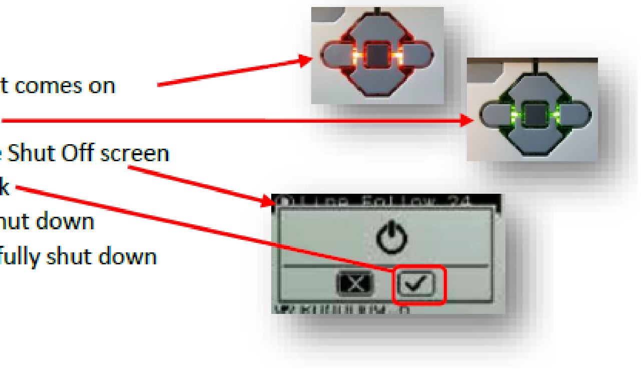


- [B]ack | Cancels Action, Aborts Running Program (important to remember), Navigates to previous screens and Shut Down screen.
- [C]enter | Turns the Brick On, Selects highlighted option
- [U]p | Navigates Up a list
- [D]own | Navigates Down a list
- [L]eft | Navigates Left across the Menu Tabs
- [R]ight | Navigates Right across the Menu Tabs

Menu Tab Icons:

- **Run Recent** | Lists all programs recently run
- **File Navigation** | Lists all Project Folders and the Programs within them that are loaded on the EV3
- **Brick Applications** | Provides access to native apps such as Port View (displays connected Motors a Sensors) and the Brick Program (enables programming directly using the interface) among other options.
- **Settings** | General configurations such as Volume, Sleep, and Wireless settings are made here

ON and OFF:



- **ON** | Press the [C]enter button until a red light comes on
  - Changes to green when ready for use
- **Off** | Press the [B]ack button until you see the Shut Off screen
  - Tab [R]ight to highlight the check mark
  - Press the [C]enter button to initiate shut down
  - The light will turn red, then off when fully shut down

E. Programming Workflow:

Basic Programming Workflow Model
1. Create Program (initial instructions / connected programming blocks)
2. Test/Run the Program
3. Experience the eternal question: “Why isn’t it working?”
4. Modify Program
5. Rinse and Repeat steps 2 thru 4 as necessary

The “Basic Programming Workflow Model” (as outlined by Evelyn Lindberg) is the very essence of this tutorial. Though we will be taking things very slowly at first, the approach is the same throughout. Plus, the Final Challenge will put this workflow to good use, so I’d like to break it down a bit and apply it to Lego Mindstorms EV3 Programming.

1. **Create Program:** In EV3 terms, this means connecting Programming Blocks together that *we hope* work in concert with each other for a desired outcome. But part of that creation process is planning, or what might be considered as pseudo programming. For us, as we think about programming our robot to do specific tasks, it might look something like this.
    a. Follow a line to an obstacle.
    b. Stop at the obstacle and grab it.
    c. Backup in an arc then stop.
    d. Release the obstacle and make a victory noise.

   With that bit of planning applied to EV3 programming, we could figure out what our robot might need in terms of Motors and Sensors, how it’s constructed, as well as what type of Programming Blocks should be used.

2. **Test/Run the Program:** As with all things that have function, programs need to be tested. What’s nice about the EV3 programming method is the ease in which you can test programs and portions of programs. This is due to how EV3 programs execute; starting with the first block, each block executes, then hands control over to the next block in line…and so on. Visually, this provides a means to “see” blocks being executed; on the programs canvas (if your bot is connected), or by the specific moves of your robot that correlates with each block. There are slight exceptions to this run-first scenario, but for the most part, it holds true.

3. **Experience the eternal question: “Why isn’t it working”:** It’s fairly rare that a program of any length runs correctly the very first time. It may *effectively* run correctly with all the correct robotic moves, but not necessarily in ways intended, or even expected. A myriad of things can be slightly off to make your robot be, well, slightly off. Any result other than exactly what you had in mind, needs debugging, or tweaking.

4. This is something you come to expect and in many ways, it can be the fun part…or not. But as mentioned earlier, the EV3 programming provides several clues as to where that undesirable outcome occurred in the programming. It’s just a matter of recognizing the signs and modifying where needed.

5. **Rinse and Repeat steps 2 thru 4 as necessary:** Eventually, victory will reign and you’ll be able to skip this step #5. But until your robots programming works exactly as intended, Rinse and Repeat is the name of the game.

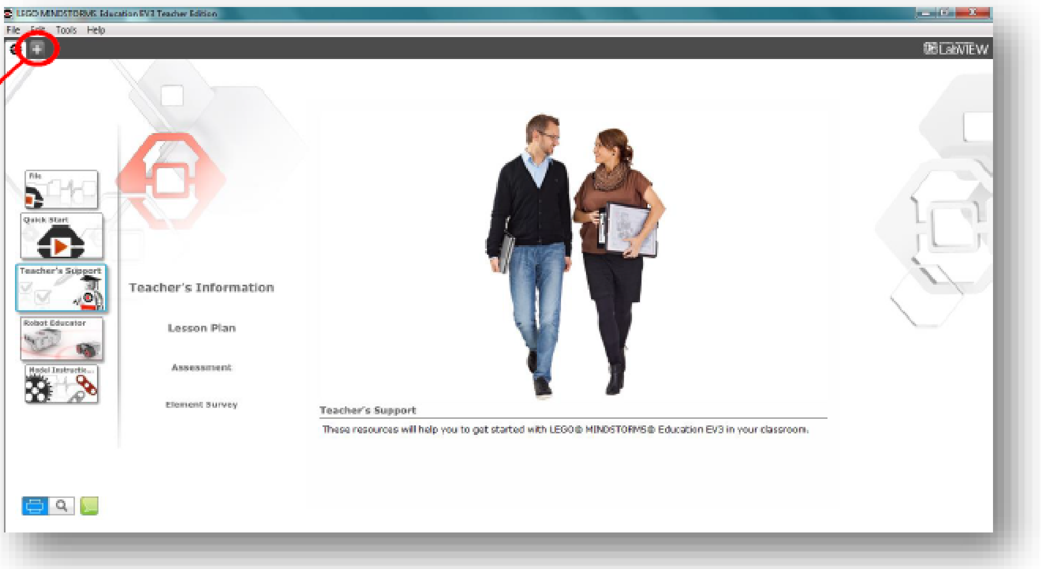
NOTE: I encourage you to explore block configurations to gain a better feel for the variables that are possible. Experimentation’s a great way to learn.

F. Launching The EV3 Program:

On your desktop, find the EV3 icon - . Double click it to launch into the EV3 Lobby Page.

There's a lot to be discussed about the Lobby and all that it offers. So much so, that I'll cover it in the video rather than attempt it here. But one thing I would like to show here is the quick way to get to the Programming Page.

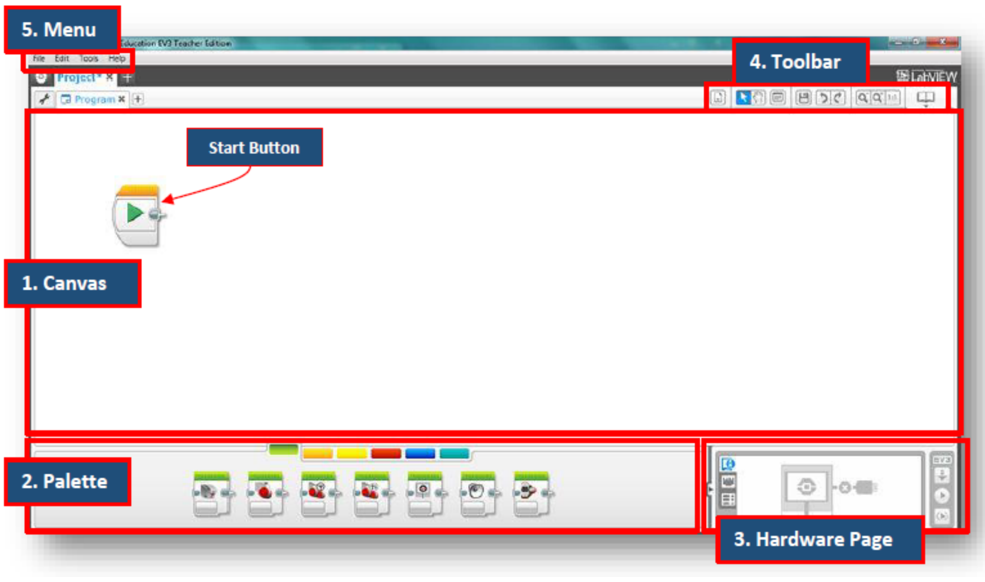
The easiest way to get to where everything happens, is to click the plus (+) sign in the upper left corner.



This will open the Programming Page that's discussed in the next section "EV3 Programming Screen Overview".

Besides several other items of interest, the video portion will discuss other ways to open new and existing Projects, as well as explore key "Helps" areas.

G. EV3 Programming Screen Overview:



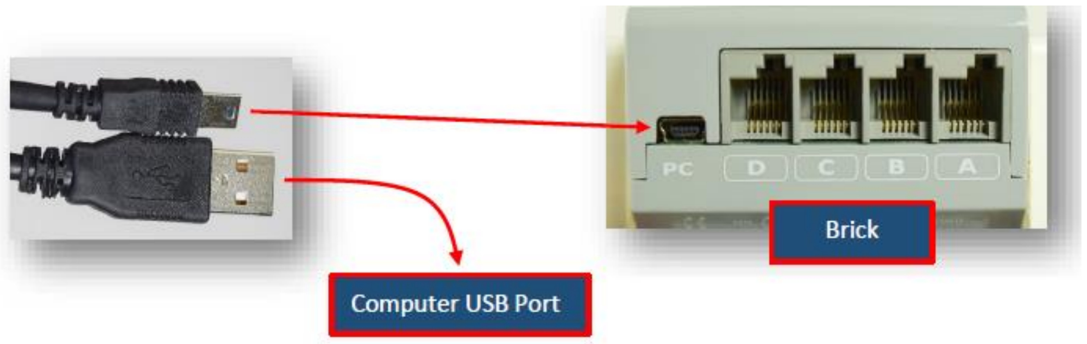
1. **Canvas**
    a. Programming area
    b. Create your program here by connecting Programming Blocks from the **Palette**
    c. By default, a Flow Block Start Button is placed in the **Canvas** area
2. **Palette**
    a. Connectable Programming "Blocks" are located here *(hovering over blocks reveals type)*
    b. Each color has a function, but we will only focus on the Green and Orange in this tutorial.
        i. Green: Action
        ii. Orange: Flow Control
        iii. Yellow: Sensors
        iv. Red: Data Operations
        v. Blue: Advanced
        vi. Turquois: My Blocks
    c. Green Action Blocks
        i. Enables programmed movement, sound, and display capabilities
    d. Orange Flow Control Blocks *(I'll refer to as just Flow Blocks)*
        i. Modifies, or enhances the capabilities of other blocks
        ii. Cannot be used solo
        iii. For our purposes, we'll start with incorporating Green Action Blocks
        iv. As we progress, we will also combine Flow Blocks
3. **Hardware Page**
    a. Brick information displayed here
    b. Download center
        i. See "Downloading to the Brick / Hardware Page Details" (page 12) for more information
4. **Toolbar**
    a. **Canvas** area controls

5. **Menu**
    a. Very similar to feel and function of a Microsoft Menu Bar

b. The “Show EV3 Help” (quick launch = F1) from the “Help” dropdown is excellent. It’s one of the best Help links I’ve ever seen. Strongly suggest exploring this feature

H. Connecting Your Robot:

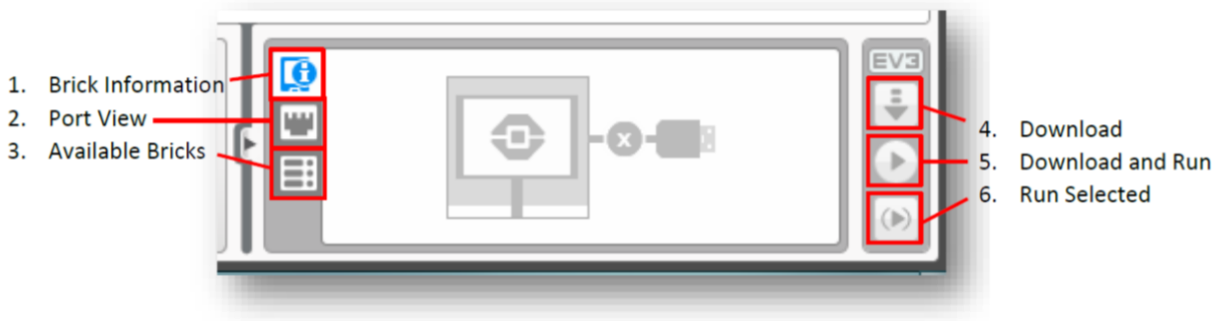
Besides the Alpha and Numeric Ports mentioned earlier, there are **PC**, USB, and SD Ports on the Brick as well. The SD Port is for expanding the memory capability and the USB port is



actually for connecting Bricks together in sequence (daisy chaining), not for connecting to the Brick to the computer; the **PC Port** is how we do that. Using the USB cable that came with your Core Set, connect the small mini B end to the Brick’s **PC Port**, and the standard end to your computer as illustrated.
NOTE: In order to run your robot (run your program) untethered, you must first download the program completely onto the Brick. The Hardware Page is where you do this, but the Brick MUST be turned on and connected (see “Connecting Your Robot” above) to the computer as well.

On the left side of the Hardware Page are the tab selections “Brick Information,” “Port View,” and “Available Bricks.” The “Available Bricks” tab is mostly for advanced use (daisy chaining) that we won’t be covering. The “Brick Information” will be covered more in-depth further down the road as it pertains (for our use) to viewing and clearing program information. For the most part, “Port View” is where we will hang out when working with our connected robots.
On the right side of the Hardware Page are three buttons that provide “Download”, “Download and Run”, or “Run Selected” operations. Depending on where you are with your programming will determine which of these buttons is most useful at the time.

Here’s all the previously mentioned tabs and buttons plus a little more Hardware Page info:



1. **Brick Information:** With the Brick turned on and connected, selecting this tab provides general information such as Firmware Version (basically the Operating System), Battery Level, Name (yes, you can name your bot here), as well as several options to manage settings and program information. Later, in the Brick Maintenance section, we will cover some of these options since by then we’ll have some programs downloaded to view and talk about.
2. **Port View:** With the Brick turned on and connected, selecting this tab provides real-time information on the status of Motors and Sensors that are connected to the Brick. When creating programs, this is an invaluable tab to have displayed. Primarily, for verifying Port assignments and checking Sensor accuracy.
3. **Available Bricks:** For advanced use that we won’t be covering in this basics tutorial.
4. **Download:** With the Brick turned on and connected, clicking this button will download your program to the Brick for non-connected operation. More on how to access and run a downloaded program in the following section “Running Untethered.”
5. **Download and Run:** With the Brick turned on and connected, clicking this button will both download your program to the Brick, and run it at the same time. So be sure to have the space for your robot to do its thing, otherwise it may run off the table or knock your coffee over.
6. **Run Selected:** With the Brick turned on and connected, clicking this button will run a specified (highlighted) block, or string of blocks, as well as download to the Brick. This is a very useful tool. Basically, there will come a time that you want to trouble shoot, or simply test a certain outcome of a few blocks. By highlighting a set of connected blocks (even if they are part of a

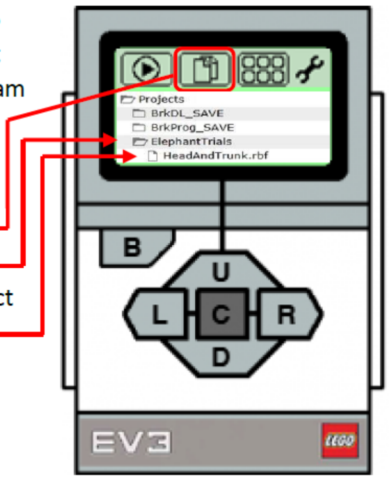
larger string), then clicking the “Run Selected” button, only those highlighted blocks will actually be run. This is also great when dealing with the sometimes troublesome “Sounds” or “Displays.” When using the Sound or Display Blocks, occasionally your first attempt to download and run will produce no results on your Brick. Highlighting the Block and clicking the “Run Selected” button will oftentimes fix this anomaly.

I. Running Untethered:

Once you’ve downloaded a program to your Brick and have unplugged the USB cable, you need to find, select, and run the program directly from the Brick; a.k.a. run untethered.

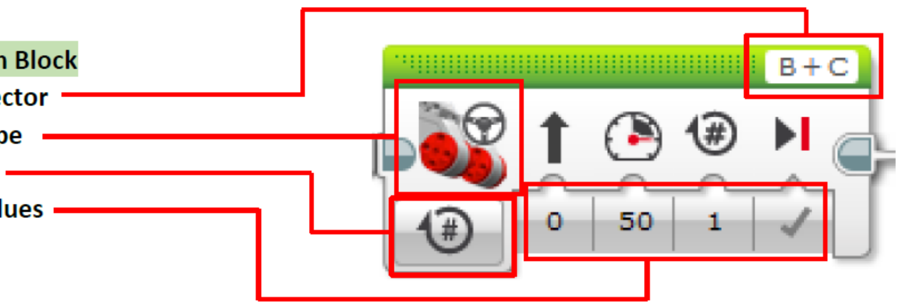
In a nutshell, you navigate to “File Navigation” on the menu tab and drill down to your Project, then Program, and select it. More specifically, using the example at right, we will navigate to the Project “ElephantTrials” to select and run the Program “HeadAndTrunk”…

1. Depending on your starting point, press the [L]eft or [R]ight buttons until you’re on the File Navigation Tab
2. Press the [D]own button to highlight your Project
3. Press the [C]enter button to select and reveal the Programs in that Project
4. Press the [D]own button to highlight your Program
5. Press the [C]enter button to select and launch your robot into action



NOTE: Each block has unique characteristics, so the following focuses only on some common components, which themselves will vary. The intention is to gain a foundational understanding of the Action Blocks we will be using.

- **The Green Action Block**
  1. **Port Selector**
  2. **Block Type**
  3. **Mode**
  4. **Input Values**



J. In the above example

1. The **Port Selector** indicates what port that particular block has been assigned. By default, every motor block has an alpha port assigned. This can be changed as needed by selecting the field and choosing the actual alpha port a particular motor is physically connected to on the Brick. It will also change automatically (auto detect) if you have your robot turned on and connected to the computer prior to connecting blocks on the canvas.
2. Keep in mind that only Motor and Sensor Blocks have **Port Selector** fields since they represent a physical Motor or Sensor that needs to communicate with the Brick via cabling.
   a. On the Brick,
      i. Alpha ports send information to Motors
      ii. Numeric Ports receive information from Sensors
   b. Also, rather than a **Port Selector** field, some blocks have a **File Name Input** field in the same location.

IMPORTANT: Though you'll not likely have a port selection issue during our exercises, sooner or later…YOU WILL! Port selection is critical for successful operation of Motors and Sensors. Always verify that the ports assigned on your program block matches the physical ports your Motors *and Sensors* are connected to on the Brick itself. And remember, Motors must be physically connected to Alpha (send) ports, and Sensors to Numeric (receive) ports.

3. The **Block Type** displays an icon that indicates the action component being configured by that block. In this example, a Steering Block (*a.k.a. – Move Steering Block*) is being used picturing two Large Motors and a steering wheel as the icon.
    a. Similar to the Steering Block, is the Tank Block. Both provide simultaneous control of two Large Motors, but the approach differs. Our exercises focus on Steering Blocks.
    b. In place of the **Direction** field (see bullet #4), a Tank Block has a second **Power** field (see bullet #4) for individual power control of each Large Motor, which for our robot equates to individual forward/backward control of each wheel.
4. Selecting a **Mode** sets the type of **Input Values** that will be available to set parameters for that block. In this example, the default **Mode** of "Number of Rotations" is selected. Notice the number (#) inside the rotation (circular arrow) symbol that makes up the icon.
5. The available **Input Values** will vary according to which **Mode** type has been selected. In this example, the **Mode** is set to "Number of Rotations." This sets the **Input Values** (from left to right) to be **Direction**, **Power**, **Rotations**, and **Brake**. Changing (or leaving) the values in those fields is what determines the parameters of influence that block has. For our robot, it's how the Large Motors are programmed and thus affect how the robot will move according to wheel rotation. More specifically…
    a. **Direction**: By entering a plus, or minus value of up to 100 in this field, the robot will either turn left, or right, and the icon (arrow) above it will also change to reflect those values. In this example, with a value of zero (neither plus, or minus), the robot will travel straight forward as indicated by the icon (arrow) above It.
    b. **Power**: By entering a plus, or minus value of up to 100 in this field, the robot will either travel forward, or backward, and the icon (speedometer) above it will also change to reflect those values. For our robot, should the value entered be a minus, the robot will move backward, plus the arrow icon above the **Direction** field will change to pointing downward indicating that the robot will move backwards.

c. **Rotations**: By entering a plus, or minus value in this field, the robot will either travel forward, or backward. But for this field, no icons will change to reflect the field entry.

   i. Also to note, should the value entered be a minus, the robot will go backward, but the arrow icon above the **Direction** field WILL NOT change to reflect backward travel as it does for a minus value in the **Power** field.

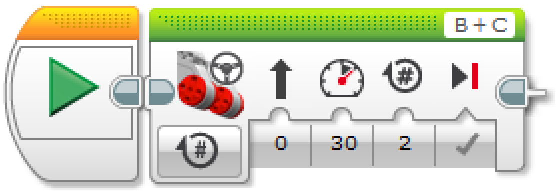
d. **Brake**: This field is strictly a select field. You either select a “Stop” or a “Coast” action for when the previous fields have run their programmed parameters. In this example, the Stop action is selected for a quicker stop, whereas the Coast action allows the Large Motors to continue turning with inertia.

IMPORTANT: A key thing to remember is that the **Mode** and **Input Values** are common to nearly every block (though not necessarily in the same place), and critical to proper EV3 programming. Also, taking note of icons that are above a field, or are part of a selection field (**Mode** as an example), will help you understand each blocks function and configuration properties.

K. Exercises – Using the Action Block's:

NOTE: For all the following exercises where the robots speed (Power) is concerned, for safety's sake, we'll slow things down and operate at a setting of 30, from the default of 50. Depending on the exercise, this setting could be a -30. To run your programs, click the Start button, or the "Download and Run" button on the Hardware Page.

1. Move Straight



a. Use a Steering Block to move 2 rotations forward.

b. Example at right shows the **Mode** set to its default "Number of Rotations," but two of the **Input Values** have changed from their default.

i. **Power** set to 30 from 50.

ii. **Rotations** set to 2 from 1.

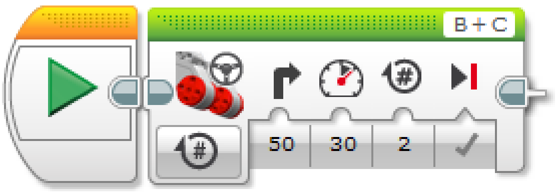
c. Use a Steering Block to move 2 rotations backward.

i. Same as above, except change the **Power** setting to -30.

ii. Note the change in the Speedometer and the Direction icons.

d. Run your programs and note wheel rotations.

2. Make 90° Turns



a. Use a Steering Block to make a 90° right turn.

Basically, the same settings as "Move Straight," but with a **Direction** setting change.

i. **Direction** set to 50 from 0.

ii. Note the icon change.

b. Use a Steering Block to make a 90° left turn.

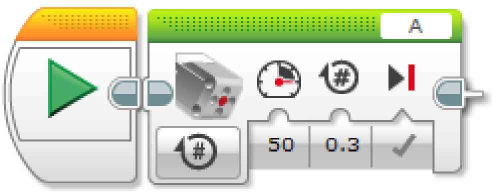
Basically, the same as above, but with a Direction setting change.

i. **Direction** set to -50 from 0.

ii. Note the icon change.

c. Run your programs and note accuracy of 90° and tweak if needed.

3. Move The Arm Up And Down



a. Use the Medium Motor Block to move the arm up.

i. Set up: Set arm down in a horizontal position.

ii. Change only the **Input Value** of number of rotations from 1 to .3 (will show as 0.3).

b. Reset the **Input Value** from 0.3 to -.3

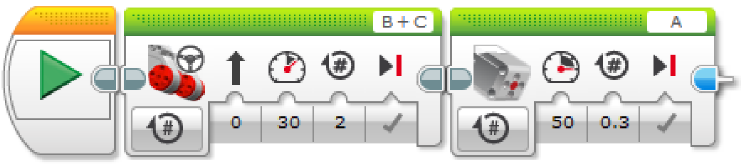
c. Run your program and note accuracy of movement and tweak if needed.

d. What's needed to move the arm up, then down as one program?

i. Hint: can't do it with just one.

4. Combine a Move and Arm Action

a. Connect a Steering Block and a Medium Motor Block to move straight forward, then move the arm up.



b. Using what you've learned, set your Steering Block to move straight at a Power of 30 for 2 rotations and then lift the arm from a horizontal, to a vertical position.

c. Run your program.

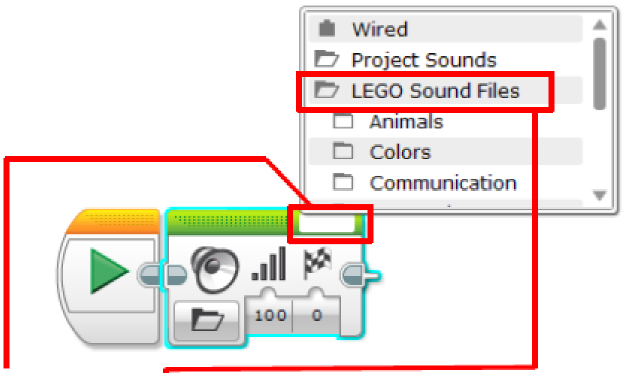
d. Now add a vertical to horizontal arm move.

5. Play with sounds

a. Set up: Make sure the sound on your laptop is on and turned up enough to hear.

b. Using the Sound Block, select a sound to play.

c. By default, the **Mode** is set to Play File; leave that *and* the **Input Values** as is.



d. Now select a sound file to play clicking the File Name field.

e. From the pop-up menu, select anything under the LEGO Sound Files (there's a lot to choose from).

f. When selected, your sound should play on your computer.

g. To hear it on your robot, run the program.

h. Have some fun and try out several of the sounds.

NOTE: The Sound Block files (and Display Block files) sometimes don't play on the Brick as they should. Normally, selecting a different sound file, running it (likely no sound still), then going back and running your original selection will usually do the trick. Another way of fixing the issue is also outlined in bullet #6 of the "Downloading to the Brick / Hardware Page Details"

“MAKE YOUR OWN ROBOT”

FIELD	INFORMATION TECHNOLOGY
DEPARTMENT	ROBOTIK CODING
MODULE NAME	MAKE YOUR OWN ROBOT
AIM	To prepare your own robot by understanding the tasks of the equipment to be used for the robot and placing these equipment at the most suitable points according to their tasks.
EQUIPMENTS	LEGO Mindstorm EV3 Core Set
PRECONDITION	• Essential LEGO Mindstorm Home Coding Platform • Loop concept • Switch case concept
TECHNIQUES	•Team work •Trial and error •Research •Argument
IMPLEMENTATION STEPS	Speed Bot robot build is downloaded from the link: https://drive.google.com/file/d/1KIiqgd24m7oQfaAqyM88udiMBLdgsVRX/view?usp=sharing and a simple robot is made by following the instructions. Then, the touch sensor, color sensor, ultrasonic sensor, gyro sensor and medium motor are assembled respectively, by discussing the functions and features of the equipment listed below. It is then tested. It is tested whether each equipment produces correct data and works.
QUANTIFICATION AND CONSIDERATION	After the group tests and finalizes their robot, they are asked to do the following task and are scored. • A black/white (changeable) line is drawn. • An object is placed 50 cm ahead. • It is ensured that the robot moves towards the object.(15 p) • The robot is stopped 2 cm before the object. (20 p) • It is provided to grasp this object by means of medium motor. (15p) • After the process is finished, the robot comes back. (15p) • When it sees the black/white (changeable) line, it is stopped. (20 p) • After stopping, the medium motor releases the object. (15p)