



# CODERS Summer Workshop



CODERS CS Team

Summer 2024

**Missouri State**  
UNIVERSITY



# What is a Computer?



mouse



keyboard



monitor



ram



motherboard



hard-drive



central processing unit (CPU)

## System software

macOS®, Microsoft Windows,  
Linux®, Android™

## Application software

Microsoft Word, Google Maps™,  
Gmail™

[Commodore 1351 mouse](#) by Boffy, 2005. [Apple keyboard](#) by user Pollyanna1313 [CC BY-SA 3.0](#)

[IBM PC Motherboard](#) by Martinez [CC BY-SA 4.0](#)

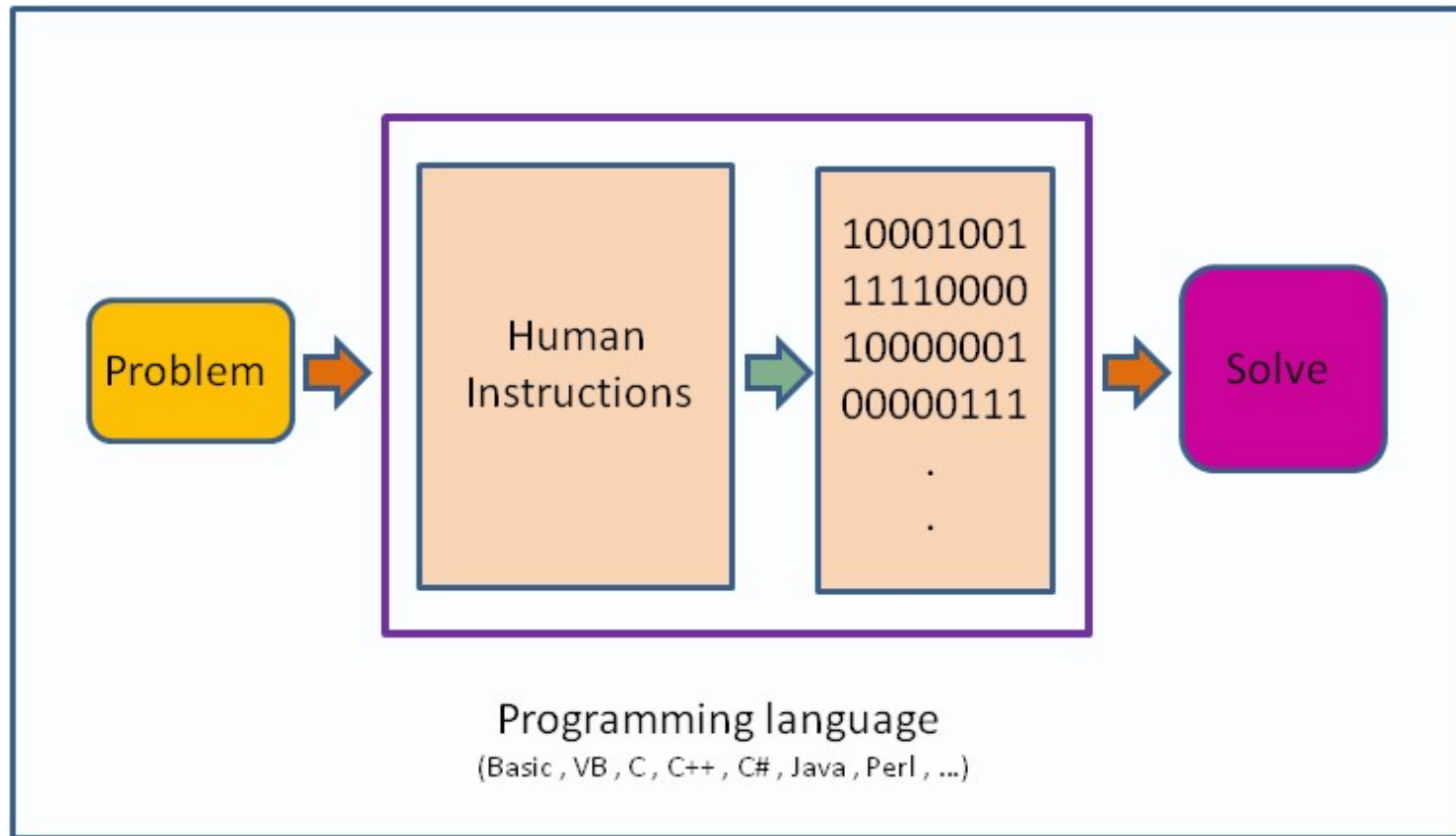
(CPUs, from left) [Intel C80186-6 \(1982\)](#) adapted from Nguyen, 2016. [Intel i7](#) by Gaba, 2018. [Intel i9](#) by Cole. [CC BY-SA 4.0](#)

Missouri State  
UNIVERSITY





# Programming a Computer





# Introduction to Block Coding (Scratch)

<https://scratch.mit.edu/>

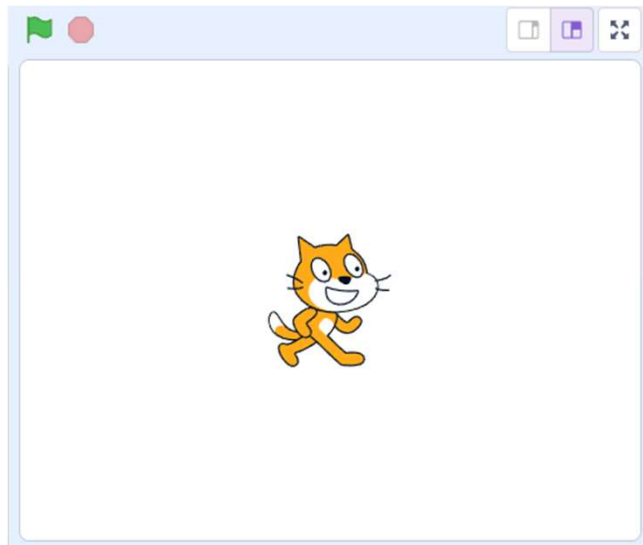
The screenshot shows the Scratch web interface with several key components highlighted by green boxes and red arrows:

- Start & stop:** Points to the green flag and red stop buttons in the top right corner of the workspace.
- Block palette:** Points to the left sidebar containing various coding blocks categorized by Motion, Looks, Sound, Events, Control, Sensing, Operators, Variables, and My Blocks.
- Script pane:** Points to the central workspace where a script is being built. The script starts with a "when green flag clicked" block, followed by an "if" block. The "if" block contains a "pick random 1 to 10 > 5" condition. If true, it says "Hi" for 2 seconds; otherwise, it says "Bye" for 2 seconds.
- Stage:** Points to the right-hand area where the Scratch cat sprite is visible.
- Sprite pane:** Points to the bottom right area showing the selected sprite (Scratch cat) and its properties (Sprite1, Size 100, Direction 90).



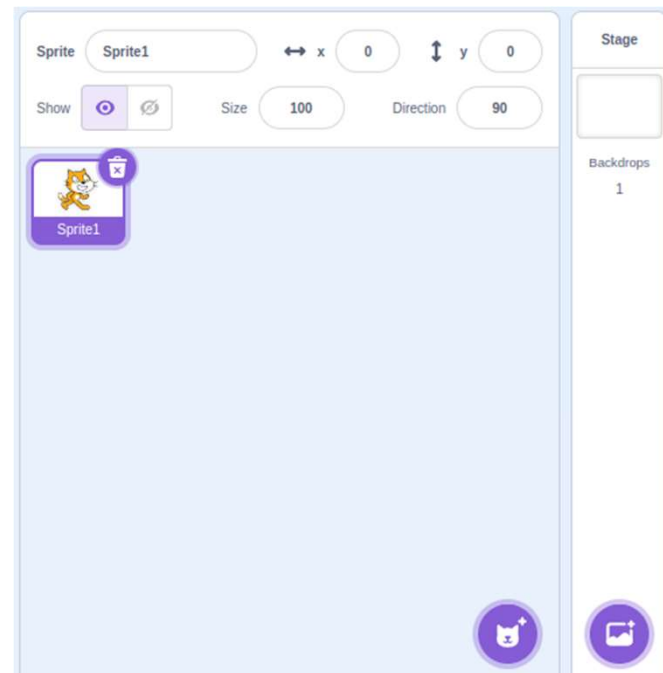
# Stage and Sprite

**Stage** - The stage is the environment where sprites exist and interact.



**Sprite** - A sprite is an image that has its own code (script), costumes, and sounds independent from all other sprites in the Scratch project.

All sprites in your Scratch project can be accessed in the sprite pane.

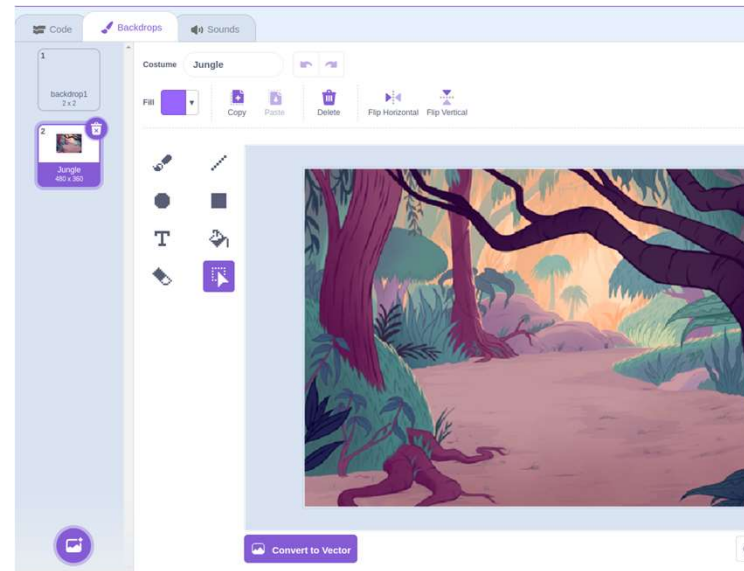
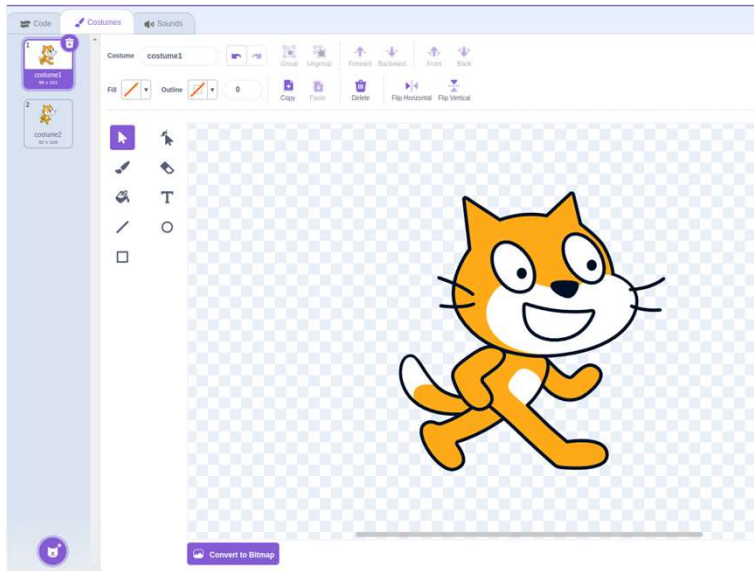




# Costume and Backdrop

**Costumes** – You can edit a sprite

**Backdrops** – You can change the scene appearance

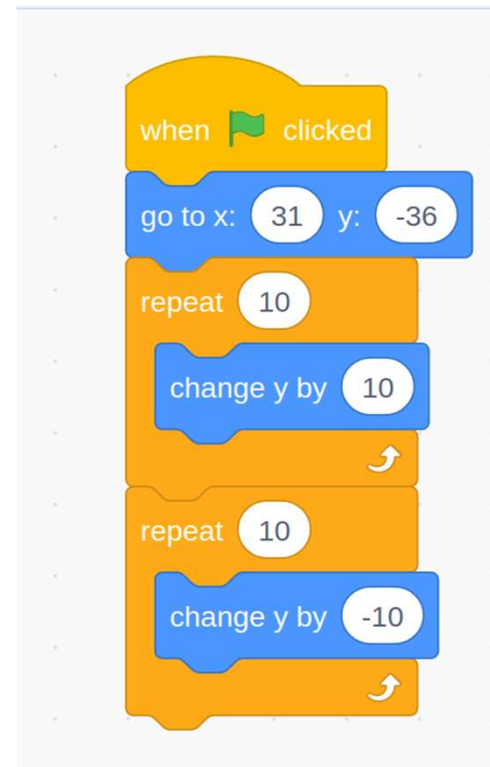
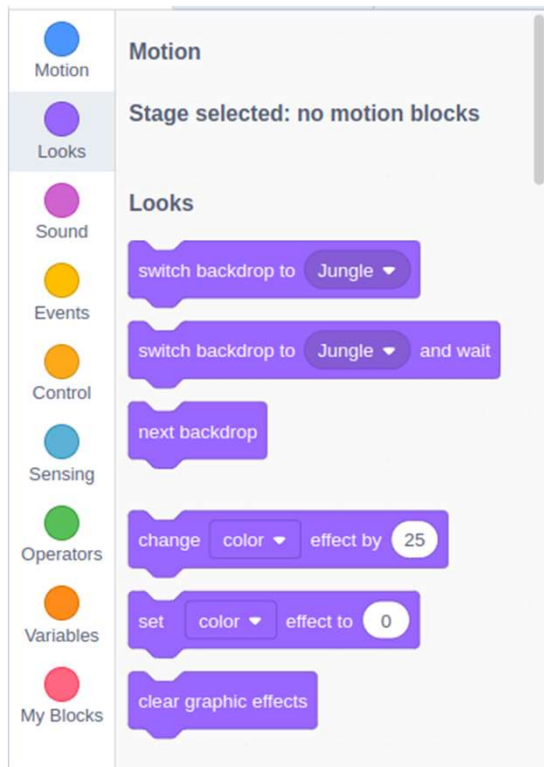




# Blocks and Scripts

**Blocks** are for choosing different actions

**Scripts** are associated with each sprite (and backdrop)





# Variables

- A variable is a placeholder to store and reuse the value.
- The value can change, depending on conditions or information passed to the program.

The screenshot shows the Scratch IDE interface. On the left sidebar, the 'Variables' category is selected. The 'Make a Variable' button is highlighted with a red box. A red arrow points from this button to a 'New Variable' dialog box. The dialog box contains a text input field with the text 'green clicked', two radio buttons labeled 'For all sprites' (selected) and 'For this sprite only', and two buttons labeled 'Cancel' and 'OK'.

Example:

The example shows two Scratch code snippets. The first snippet consists of a 'when green flag clicked' event block followed by a 'change green clicked by 1' block. The second snippet consists of a 'when space key pressed' event block followed by a 'set green clicked to 0' block.

Missouri State  
UNIVERSITY



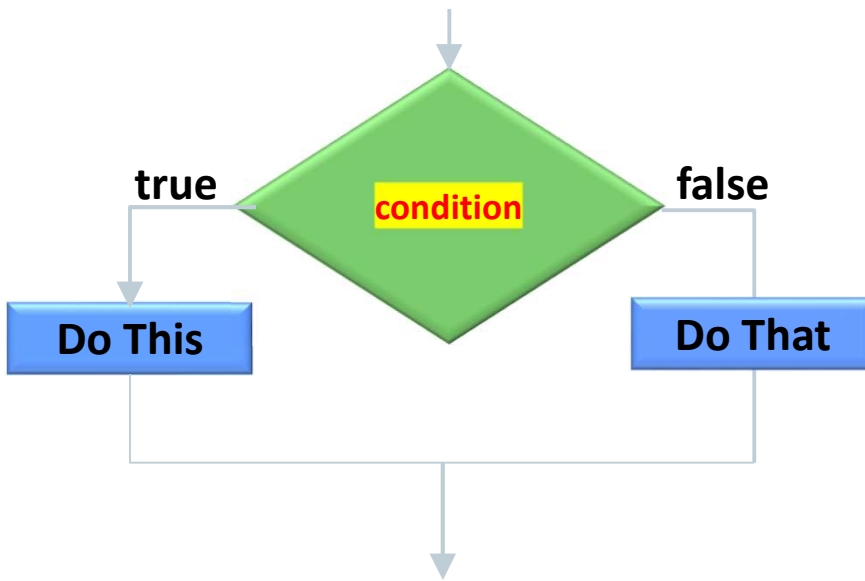




# Branching



- Make the program do something when a condition is True or False.



## Examples:

```
when clicked clicked\nshow\n  go to x: -211 y: -150\n\nwhen space key pressed\n  move 10 steps\n  if touching edge? then\n    hide
```

## IF Block

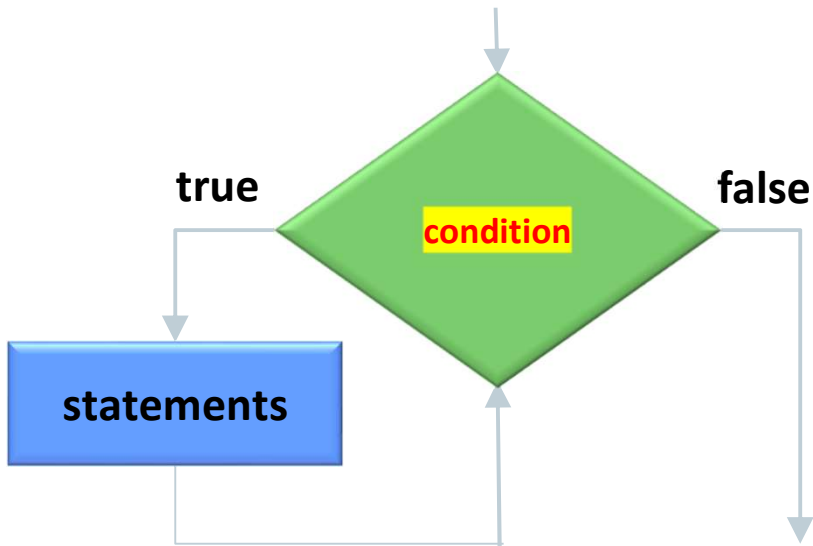
```
when clicked clicked\nshow\n  go to x: -211 y: -150\n\nwhen space key pressed\n  move 10 steps\n  if touching edge? then\n    hide\n  else\n    go to random position
```

## IF-ELSE Block

# Loop



- A **loop** allows repeated execution of a set of statements until a condition is met.



## Example-1:

```
when space key pressed
repeat 50
  turn 25 degrees
  wait 0.1 seconds
```

The code for Example-1 is a Scratch script. It begins with a yellow "when space key pressed" block. This is followed by an orange "repeat" block set to 50 iterations. Inside the repeat loop, there are two blocks: a blue "turn 25 degrees" block and an orange "wait 0.1 seconds" block. The script ends with a curved arrow indicating the end of the code.

## Example-2:

```
when space key pressed
repeat until touching edge ?
  go to random position
  wait 0.25 seconds
```

The code for Example-2 is a Scratch script. It starts with a yellow "when space key pressed" block. This is followed by an orange "repeat until" block with the condition "touching edge ?". Inside the repeat loop, there are two blocks: a blue "go to random position" block and an orange "wait 0.25 seconds" block. The script ends with a curved arrow indicating the end of the code.



# Nested Loop



- A loop can be inside another loop (and so on)

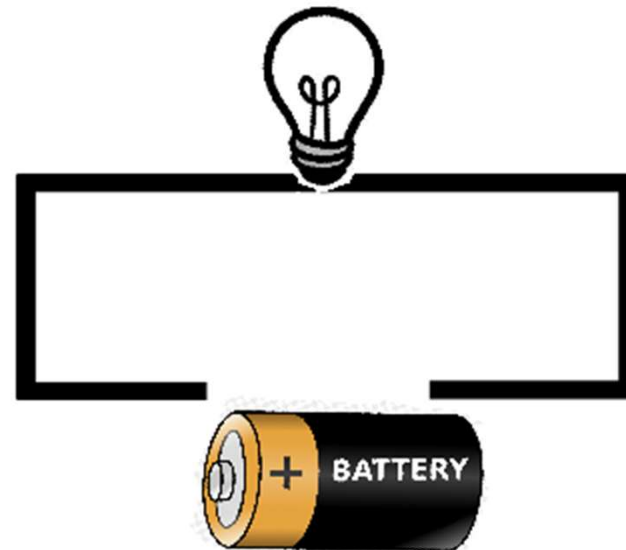
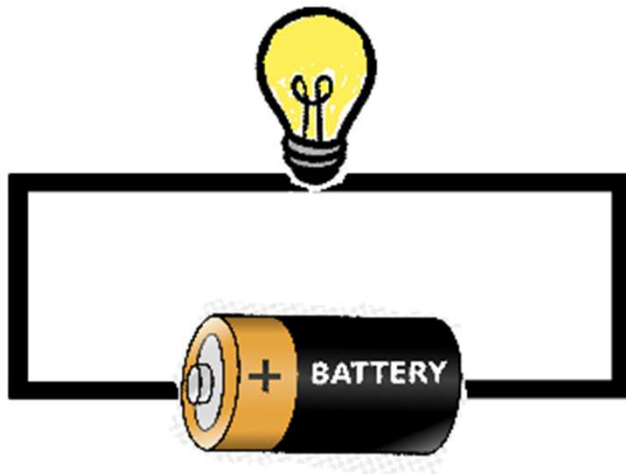
**Example:**





# Activity-1: Light the Bulb!

**Goal:** Simulate a simple circuit in scratch using wires, a bulb, and a battery. The bulb will light up when the circuit is complete.





# Activity: Sample Code

- Code for the battery
- Code for the bulb

```
when clicked clicked
go to x: -184 y: 124
forever
  set on to 0
  if touching wire ? then
    set on to 1
```

```
when clicked clicked
forever
  if on = 1 then
    switch costume to bulb2
  else
    switch costume to bulb1
```



# Introduction to Micro:bit



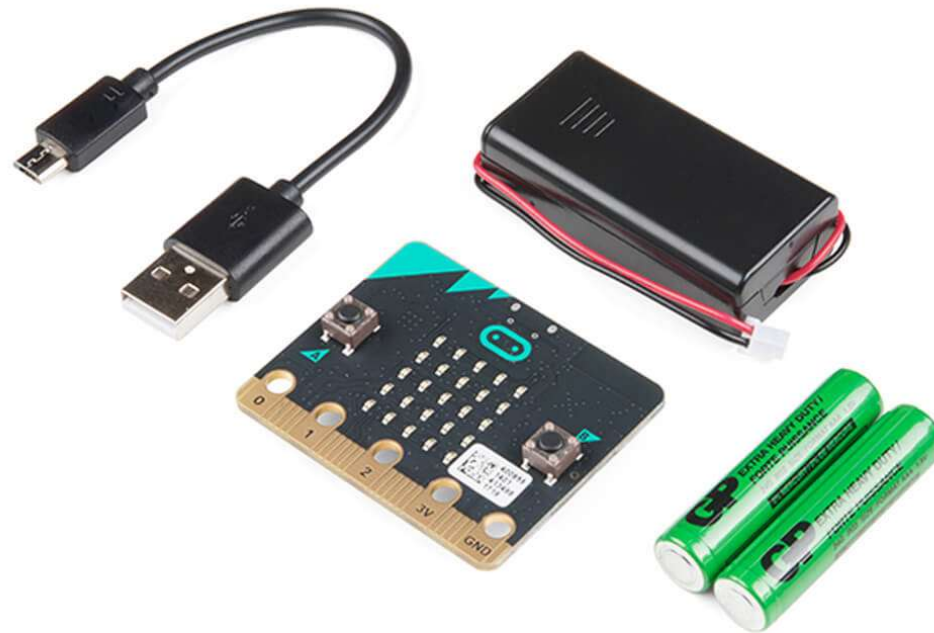
<https://microbit.org/>

Missouri State  
UNIVERSITY





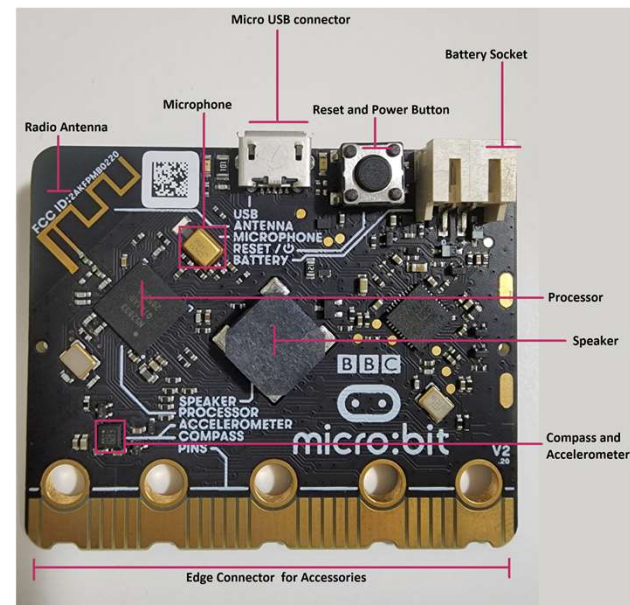
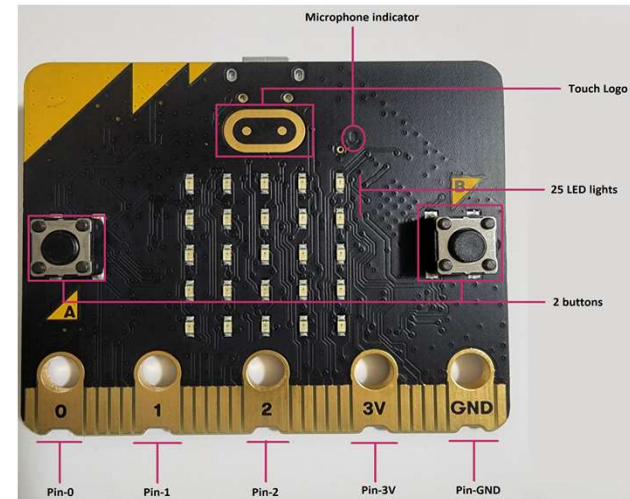
# Unpacking the Micro:bit





# Micro:bit Components and Capabilities

- LED Display
- Speaker
- Buttons
- Microphone
- Accelerometer
- Compass
- Temperature Sensor
- Light Sensor
- Touch Sensor
- Radio



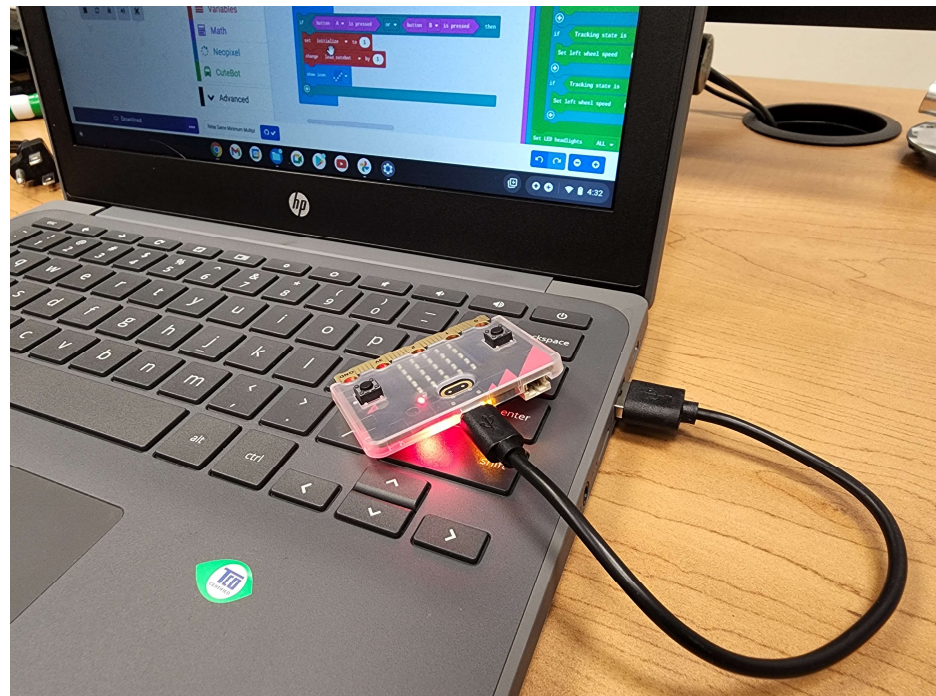




# Connecting to the Micro:bit

Connect micro:bit to a computer directly using a **USB-A to micro-USB** cable.

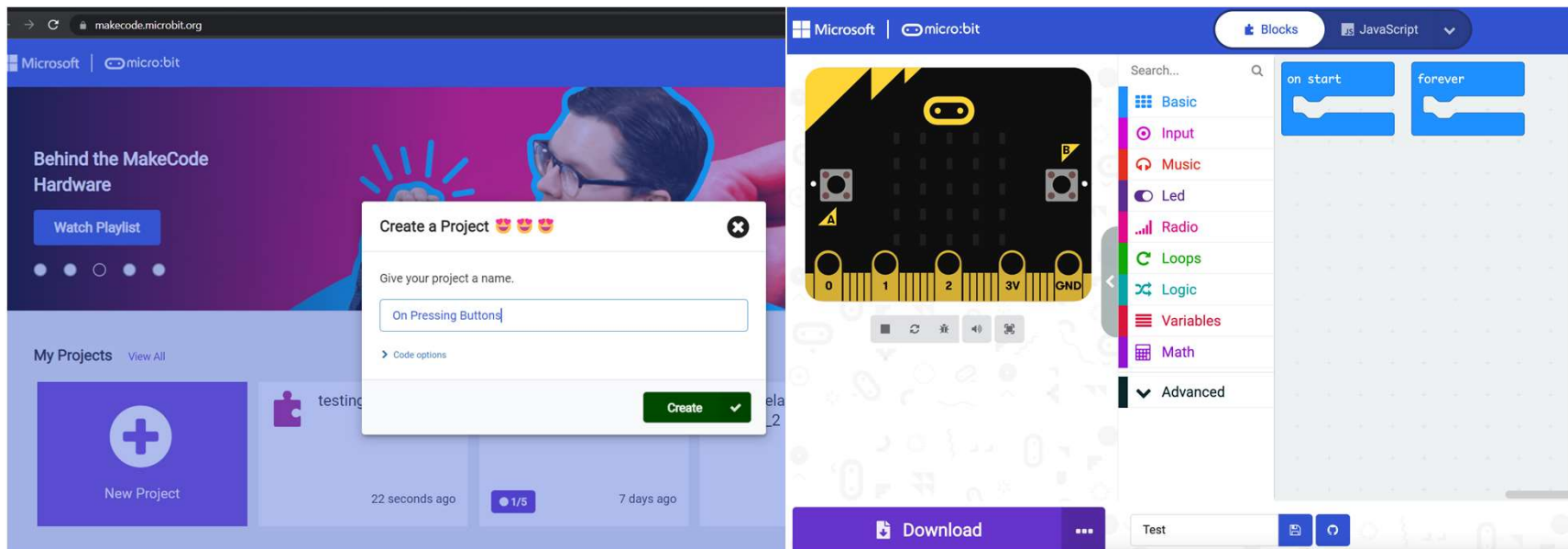
1. Plug small connector of micro-USB cable into the micro:bit USB connector.
2. Plug USB type A connector into an open USB port on the computer.





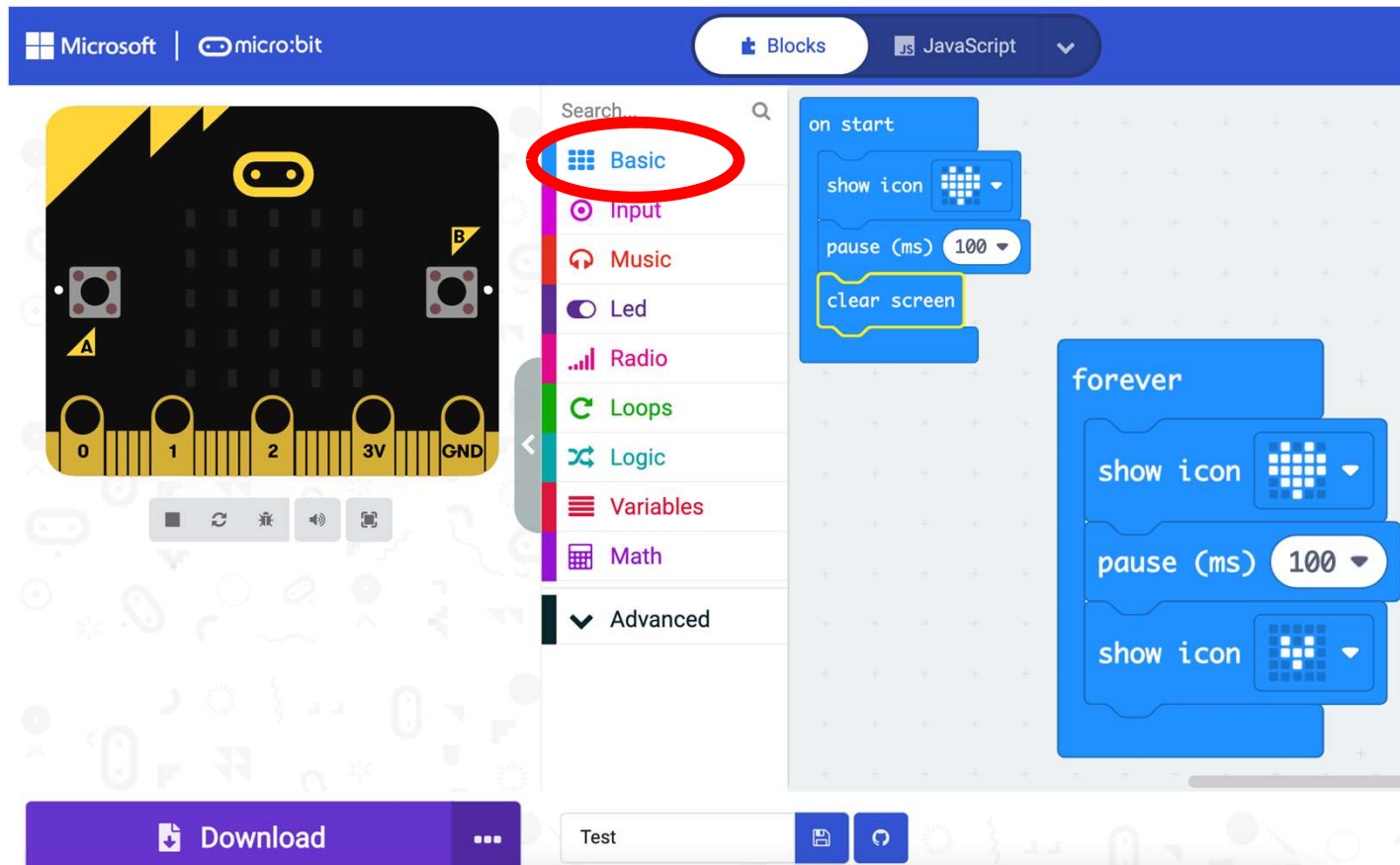
# Let us code!

1. Open **Google Chrome** web-browser
2. Navigate to <https://makecode.microbit.org/>
3. Select "New Project" and type in a name for your project, e.g. **"My First"**.





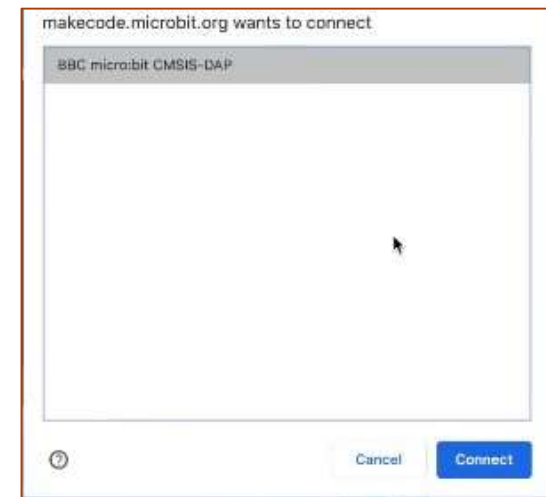
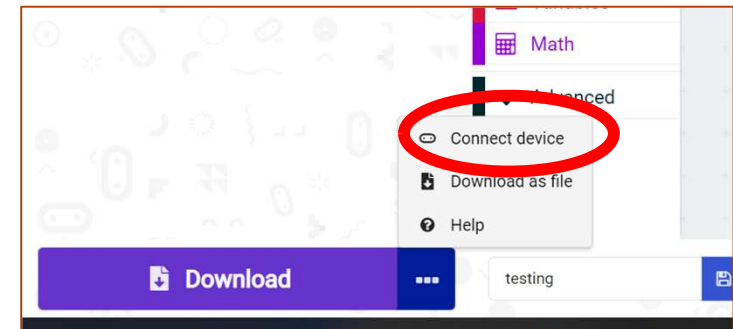
# Micro:bit - Let us code!





# Flashing Program: Direct Flashing

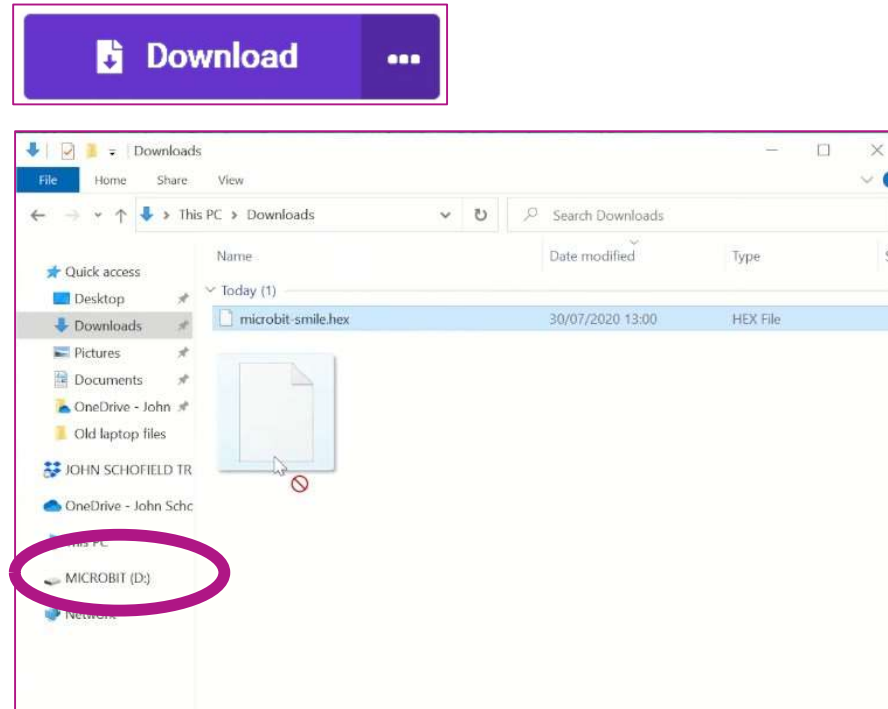
1. From your project window, click on the three dots next to Download and select "Connect Device".
2. Click "next" twice and select your micro:bit.
3. Once connected, click the "Download" button.
4. Your code will be sent to the micro:bit and the amber light on the back of the micro:bit will be flashing while the download is in progress.





# Flashing Program: Drag and Drop

1. Click "Download" and save your code as a **.hex** file in your computer.
2. Drag and drop the .hex file to the micro:bit drive shown in your file explorer.





# Activity-1: Temperature Monitor

Code the micro:bit to display temperature (in Fahrenheit).

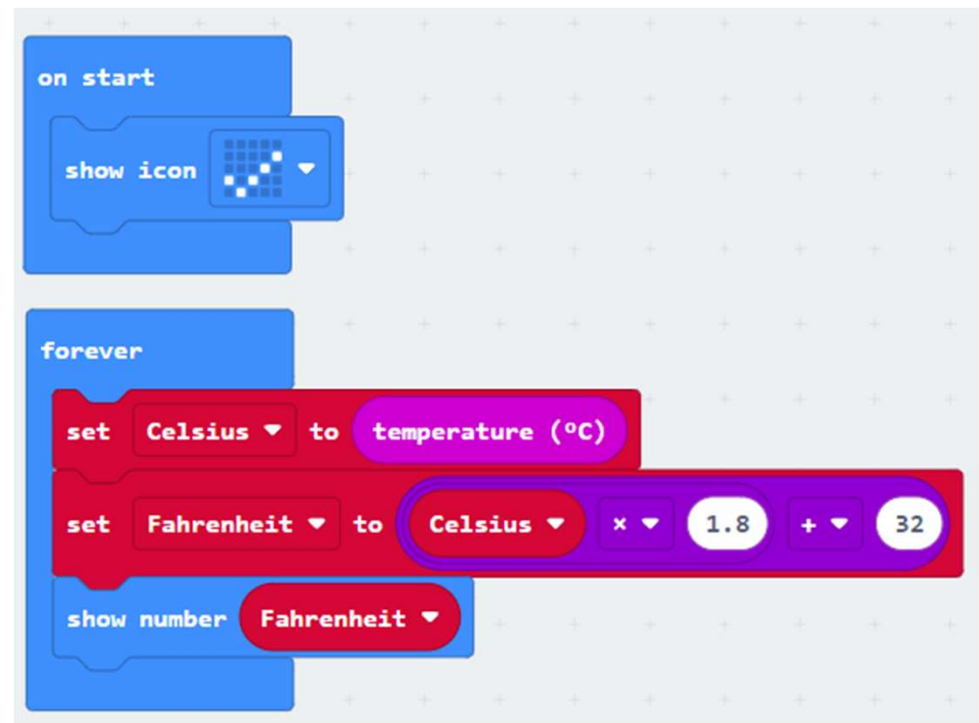
## Input:

Temperature sensor.

## Output:

Display current temperature (in Fahrenheit) on the micro:bit

$$\text{Fahrenheit} = (\text{Celsius} * 1.8) + 32$$





## Activity-2: Exploring Various Inputs

```
on start
  show icon [grid icon]

forever
  if [button A] is pressed then
    show string "Hello!"
  else if [button B] is pressed then
    show string "Bye"

on shake
  set text list to array of "Happy" "Sad" "Angry"
  show string [get random value from text list]
```





## Activity-3: Fortune Teller!

**Goal of this exercise is to create a fortune teller game like magic 8 ball!**

1. From the “Advanced” section create an array of answers.
2. On shake pick a random answer from the array.
3. Display the answer.

**Note:** Arrays are kind of variable that are used to store multiple values.

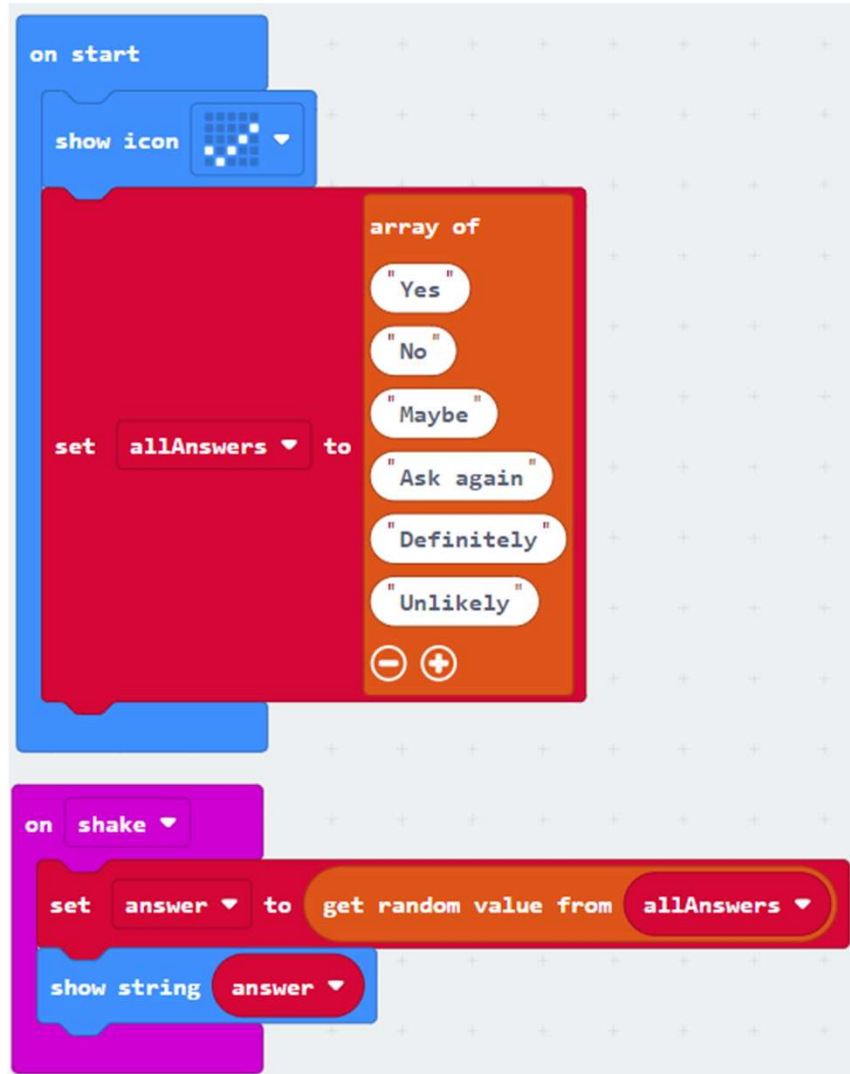
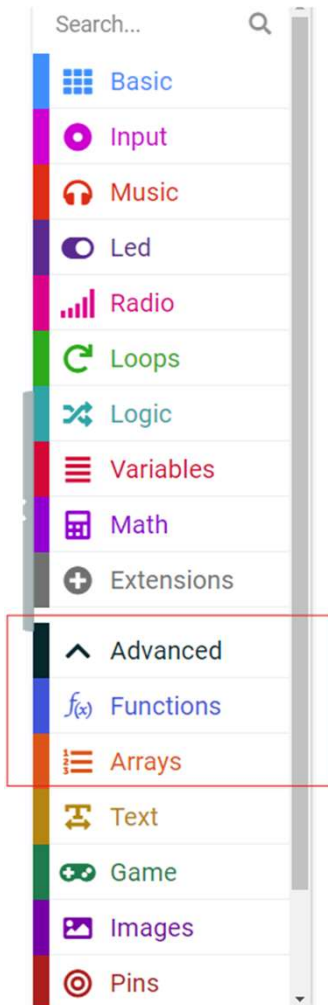
Example,

```
allAnswers = ["Yes", "No", "Maybe", "Ask Again", "Definitely", "Unlikely"]
```





# Activity-3: Code





# Activity-4: Classroom Noise Monitor!

<https://www.mdhearingaid.com/blog/decibel-chart/>

**Goal of this exercise is to create a noise monitor**

1. Sense the sound level using micro:bit sensor.
2. Display message based on the noise level.

Noise Level	Possible Damage
<60	Safe
60-80	Annoying
80-85	Possible hearing damage after 2 hours of exposure
85-95	Possible hearing damage after 50 minutes exposure
95-100	Possible hearing damage after 15 minutes exposure
100-110	Possible hearing damage after 5 minutes exposure
110-150	Possible hearing damage within 1 minute
150+	Damage immediately



# Activity-4: Classroom Noise Monitor!

<https://www.mdhearingaid.com/blog/decibel-chart/>

```
on start
  show icon [grid icon]

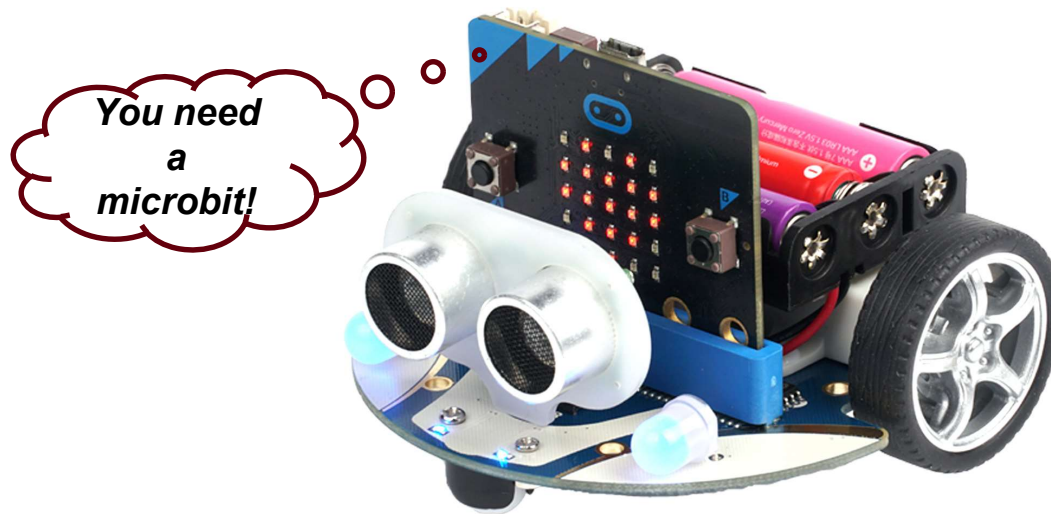
forever
  set noise to sound level
  show number noise
  if noise < 60 then
    show string "Safe!"
  else if noise < 80 then
    show string "Annoying!"
  else if noise < 85 then
    show string "Possible hearing damage after 2 hours exposure"
```

```
else if noise < 95 then
  show string "Possible hearing damage after 50 minutes exposure"
else if noise < 100 then
  show string "Possible hearing damage after 15 minutes exposure"
else if noise < 110 then
  show string "Possible hearing damage after 5 minutes exposure"
else if noise < 150 then
  show string "Possible hearing damage within 1 minute exposure"
else if noise ≥ 150 then
  show string "Pain and ear injury"
```



# Introduction to Cutebot

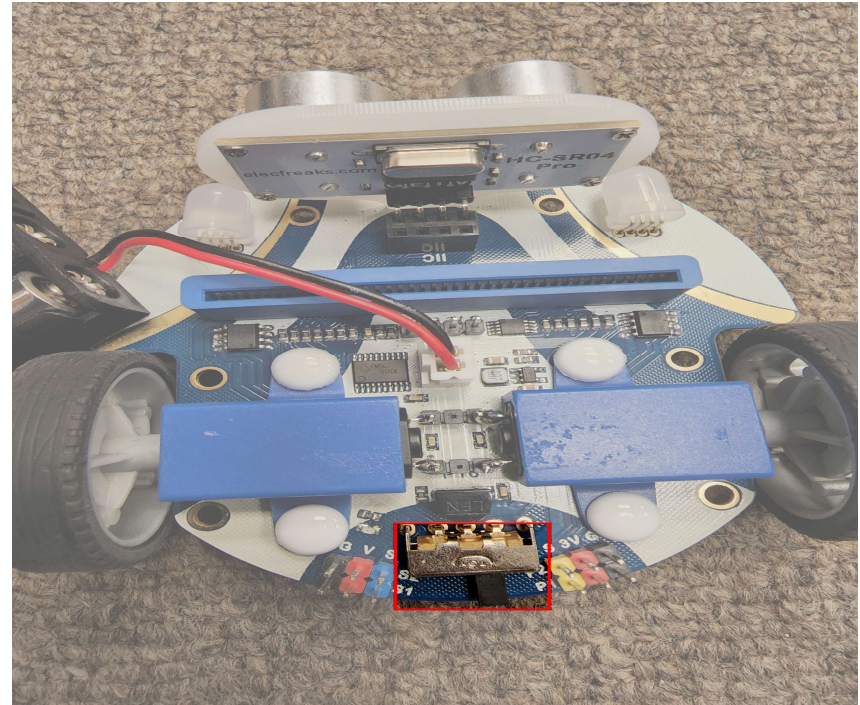
[https://www.electfreaks.com/learn-en/microbitKit/smart\\_cutebot/cutebot\\_car.html](https://www.electfreaks.com/learn-en/microbitKit/smart_cutebot/cutebot_car.html)





# ON/OFF Switch

- There is a **black** ON/OFF switch and a status LED.

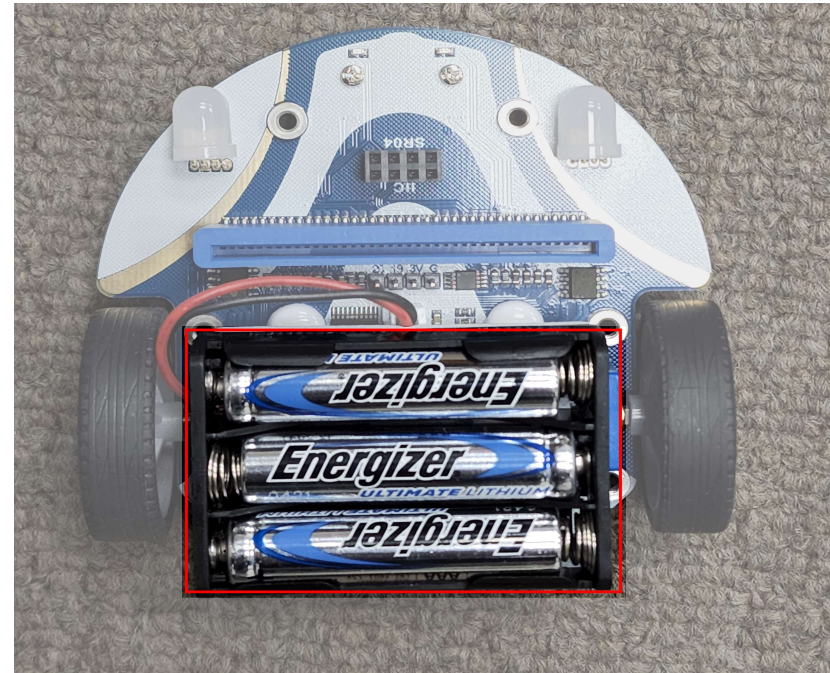






# Battery Placement

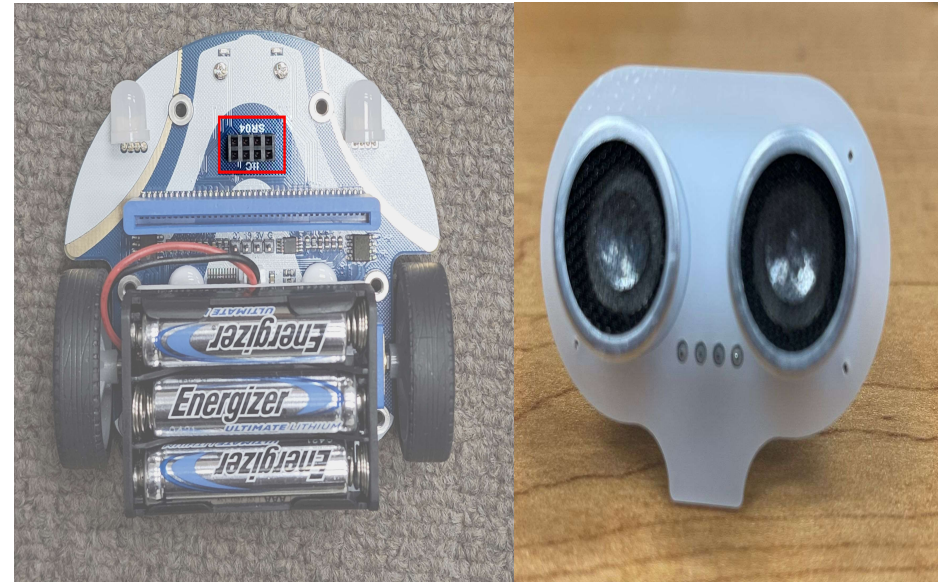
- An expansion board for **3xAAA** batteries is placed on the top of the Cutebot (and behind the micro:bit slot).





# Ultrasonic Sensor

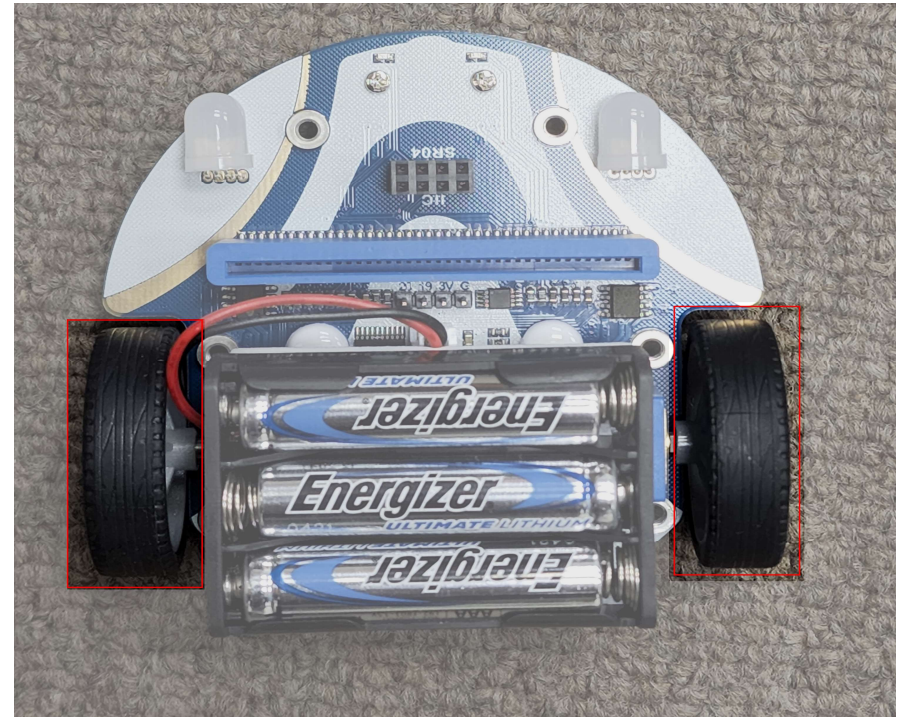
- The red highlighted part of the Cutebot has an **ultrasonic sensor** connection **SR04** in the front top part of the Cutebot to connect ultrasonic sensor.
- The ultrasonic sensor helps the Cutebot to sense object in-front of it and measure the distance of the Cutebot from the object.





# Wheels

- Two wheels on the sides are driven by DC motors (maximum 300 RPM).
- The wheels are responsible for moving the Cutebot forward or backward.

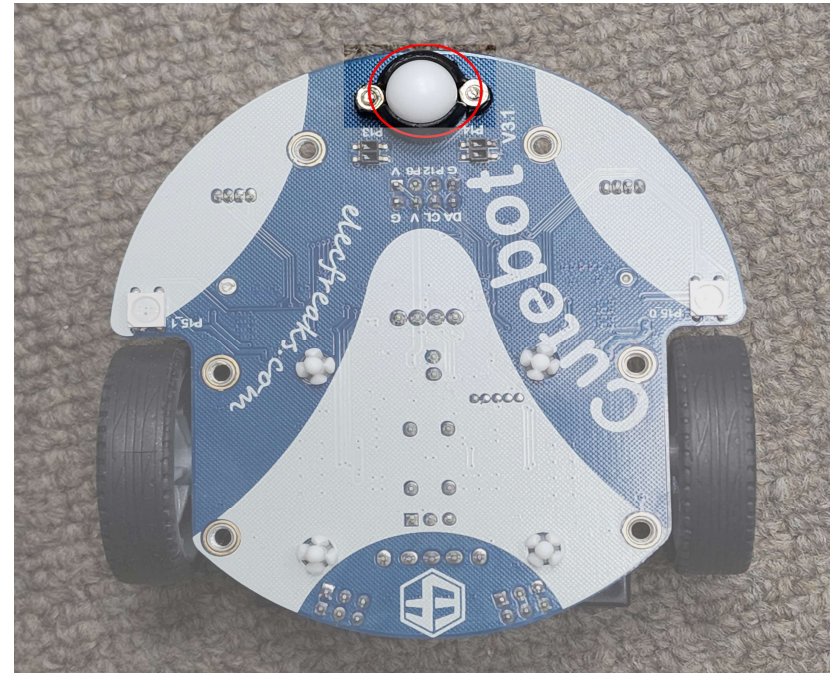






# Universal Wheel

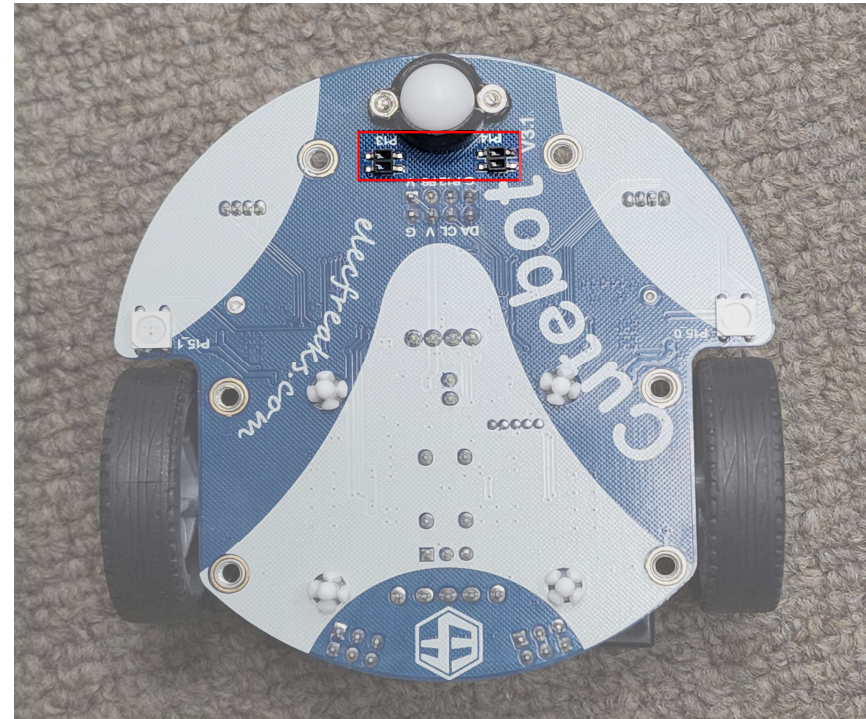
- A white universal wheel is placed in the front bottom of the Cutebot.
- The purpose of this feature is to steer the Cutebot in all directions.





# Line Tracking Sensors

- The highlighted part shows 2-line tracking sensors.
- The purpose of line tracking sensors is to detect broad lines and their edges.

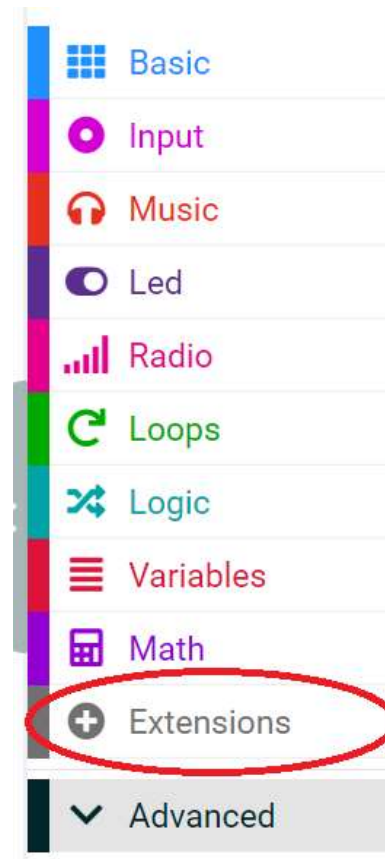




# Coding the Cutebot

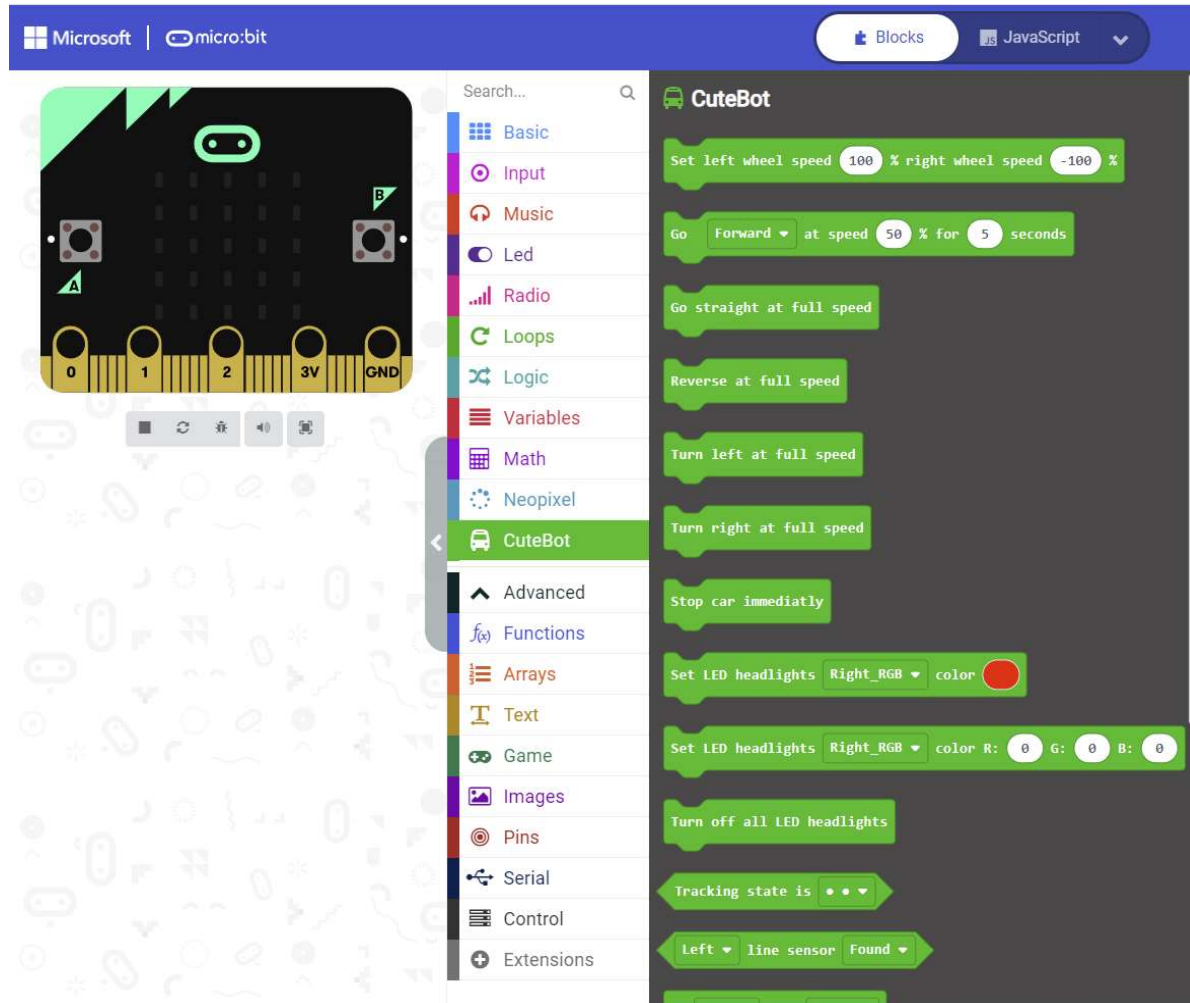
In a Chrome browser, go to: <https://makecode.microbit.org>

1. Create a **new** project and give it a unique name
2. Click "**Extensions**" from the side drawer.
3. Search for "**cutebot**".
4. Select Cutebot to add it to your project.





# Coding the Cutebot





# Activity-1: Don't Crash

**Goal:** The cutebot will move around freely with the headlights on, avoiding the obstacles in front of it.

1. If no obstacle, go forward with green headlights.
2. Else, turn the headlights to red, stop the car, and try to go in a different direction to avoid obstacles.

```
on start
  show icon [grid icon]

forever
  if [HC-SR04 Sonar unit] > 30 then
    show icon [grid icon]
    Set LED headlights ALL color [green]
    Set left wheel speed 40 % right wheel speed 40 %
  else
    show icon [grid icon]
    Set LED headlights ALL color [red]
    Stop car immediatly
    Set left wheel speed [pick random 0 to 100] % right wheel speed [pick random 0 to 100] %
    pause (ms) 500
```





## Activity-2: Line Following

**Goal:** The cutebot will drive along the black line and will adjust to going back to the black line if any deviation happens.

```
forever
  if Tracking state is ●●▼ then
    Set left wheel speed 20 % right wheel speed 20 %
  +
  if Tracking state is ●○▼ then
    Set left wheel speed 10 % right wheel speed 20 %
  +
  if Tracking state is ○●▼ then
    Set left wheel speed 20 % right wheel speed 10 %
  +
```



# Meet the dog!

<https://www.electronicsforu.com/learn-en/microbitKit/microbit-xgo-robot-kit/microbit-xgo-robot-kit-Introduction.html>

- Aluminum alloy shell (NW 500g)
- Built-in battery (120min in one charge)
- Each foot has three servos to allow a flexible and smooth movement
- Built-in actions, e.g., sit down, look for food, etc.



Charger

XGO robot dog

Ring:bit

Screw  
and  
Beam



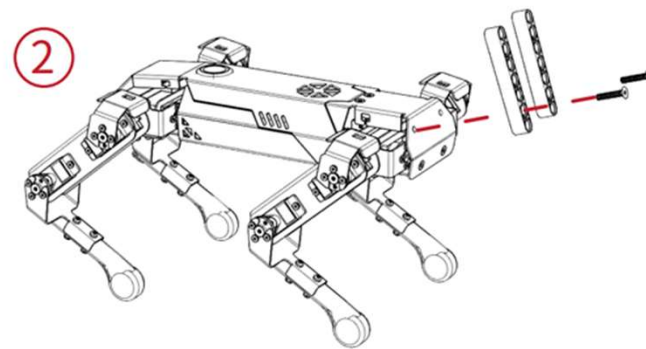
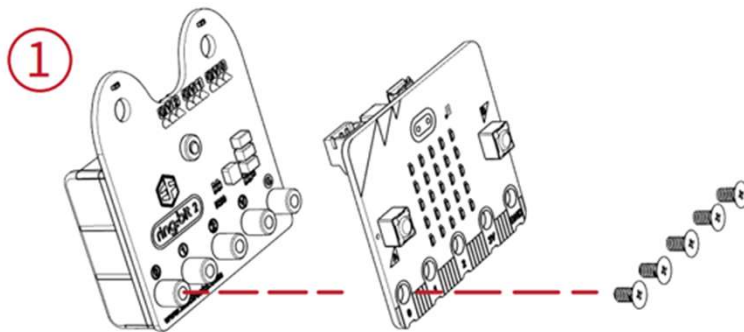
Missouri State  
UNIVERSITY





# Assembling

- **Step-1:** Attach the micro:bit to the ring:bit expansion board using the short flat head screws
- **Step-2:** Use the longer flat head screws to fasten the seven-hole beam to the corresponding screw holes on the XGO

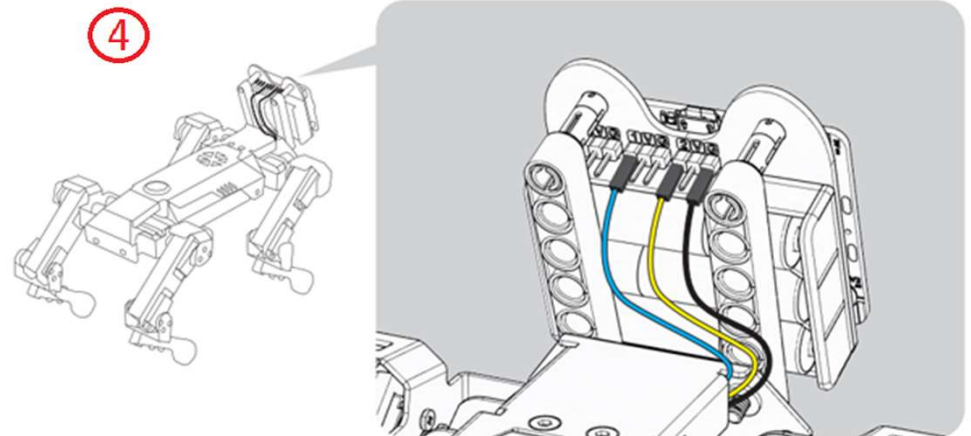
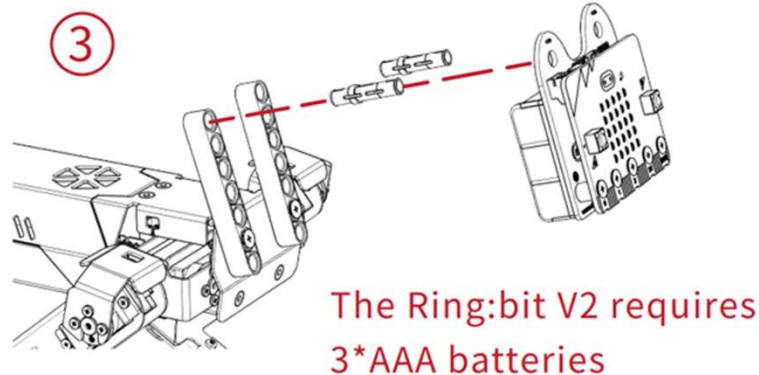






# Assembling Contd.

- **Step-3:** Add 3xAAA batteries and mount the ring:bit expansion board with the micro:bit installed to the seven-hole beam using the long pins
- **Step-4:** Connect the DuPont cables (Yellow, Black, and Blue) from the XGO to the corresponding ports of the ring:bit
- Blue DuPont cable is connected to port 1, yellow DuPont cable is connected to port 2, and black DuPont cable is connected to port G

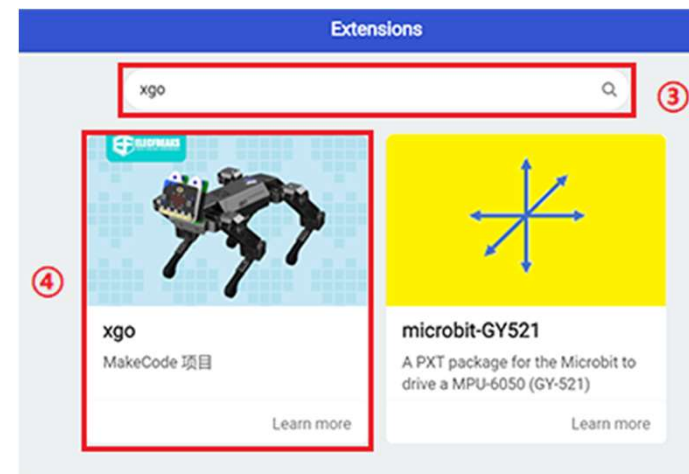
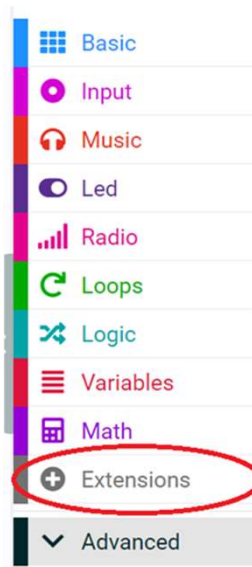
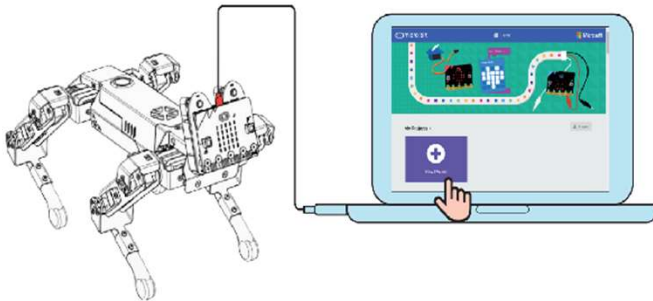




# MakeCode-Project Setup

<https://makecode.microbit.org/>

1. Connect your XGO micro:bit to the computer using USB cable
2. Goto the makecode website from Chrome web browser
3. Create a new project, and give it a unique name
4. Click on **Extensions**
5. Search for **XGO**
6. Add XGO to your project



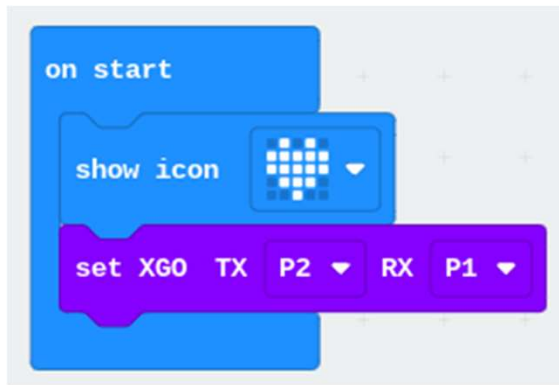
Missouri State  
UNIVERSITY



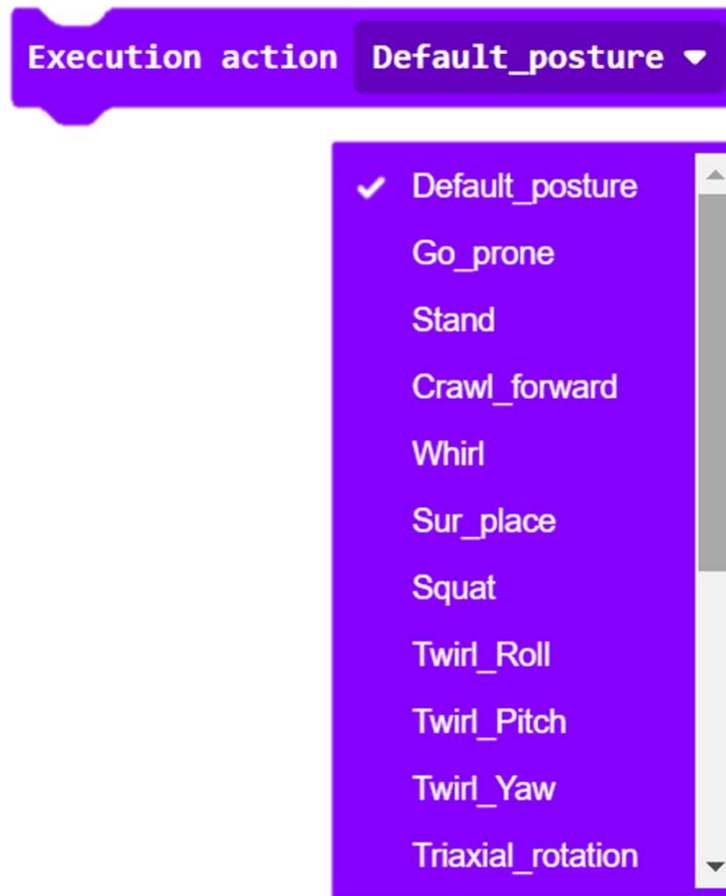


# Programming XGO - Initialization

- Initialize the XGO on start



- Choose a default XGO action from the following list





# Activity-1: Beware of Dog!

**Goal:** The dog starts scaring as soon as the light turns on to indicate that someone has entered the room.

1. If light level  $> 100$ , it means an intruder has entered the house.
2. To scare the intruder, show a scary face and play a mysterious sound in the background while moving like a pendulum.
3. Else, show a happy icon and stand in the default position.

```
on start
  show icon [Scary Face]
  set XG0 TX P2 RX P1

forever
  if light level > 100 then
    show icon [Scary Face]
    play [mysterious] in background
    Execution action Play_pendulum
  else
    show icon [Happy Face]
    Execution action Default_posture
```



# Bulbs and Battery Activity: Sample Code

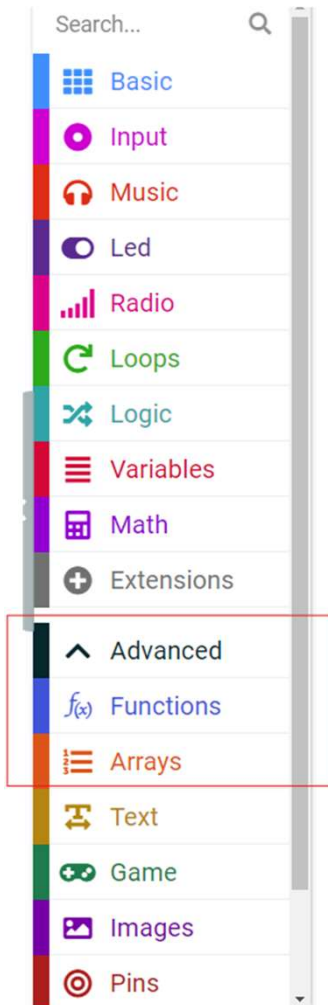
- Code for the battery
- Code for the bulb

```
when green flag clicked
  go to x: -184 y: 124
  forever loop
    set on to 0
    if touching wire? then
      set on to 1
```

```
when green flag clicked
  forever loop
    if on = 1 then
      switch costume to bulb2
    else
      switch costume to bulb1
```



# Fortune Teller: Sample Code



```
on start
  show icon 
  set allAnswers to array of
    "Yes"
    "No"
    "Maybe"
    "Ask again"
    "Definitely"
    "Unlikely"
  on shake
    set answer to get random value from allAnswers
    show string answer
```





## Activity-2: BINGO Bot

**Goal:** The cutebot will generate random numbers while following a black line to complete the bingo.

### Expected functionalities:

1. Create a fixed length array that store some numbers and play the role of bingo card.
2. Generate random number and match with the values of the array.
3. If the generated random number matches with any value of the array, remove that value from the array.
4. Keep repeating the process of point 2 and 3 until the array is empty.
5. While doing above mentioned processes, the cutebot will parallely keep following a black line.
6. When the array becomes empty it will show “BINGO!” on the microbit and the cute cutebot will spin in one place and play a happy music.



## Activity 2: Shake my hand

**Goal:** The dog starts shaking hand when you give your hand forward while holding a micro:bit.

### Expected functionalities:

1. The person will carry a micro:bit in his or her hand which will send a radio signal to the robot.
2. The robot dog will receive the signal.
3. Based on the strength of the received signal, the robot dog will understand if the person has forwarded his/her hand for handshake or not.
4. If the signal strength is strong, the dog will shake hand, else it will stand in the default posture.

**Note:** The range of micro:bit radio signal is from  $-128$  to  $-28$ , where  $-128$  means the signal is weak and  $-28$  means the signal is strong.



# Activity-2: BINGO Bot Sample Code

Code for initializing array

```
on start
  show icon
  set list to [1, 2, 3, 4, 5]
  set bingo to false
```

Code for making the bingo game

```
forever
  set number to pick random 1 to 10
  show number number
  set len to length of array list
  set count to 0
  if len = 0 then
    set bingo to true
    show string "BINGO!"
  else
    while count < len do
      set value to list get value at count
      if value = number then
        list remove value at count
        change count by 1
    end while
  end if
  pause (ms) 200
```

Code for following black line

```
forever
  if Tracking state is ●● then
    Set left wheel speed 50 % right wheel speed 50 %
  +
  if Tracking state is ○● then
    Set left wheel speed 50 % right wheel speed 0 %
  +
  if Tracking state is ●○ then
    Set left wheel speed 0 % right wheel speed 50 %
  +
  if bingo = true then
    Turn right at full speed
    play giggle until done
  +
```



## Activity 2: Shake my hand sample code

```
on start
  show icon [grid icon]
  radio set group 1
  radio set transmit power 7

forever
  radio send string "1"
```

Code for the micro:bit  
in a person's hand

```
on radio received receivedString
  set signal to received packet signal strength

on start
  show icon [grid icon]
  set XG0 TX P2 RX P1
  radio set group 1

forever
  if signal > -28 then
    Execution action Handshake
    pause (ms) 100
  else
    Execution action Default_posture
```

Code for the robot dog to shake its' hand  
based on the received signal



# Thank You

