MONTFORT
SECONDARY
A Montfortian Gabrielite Education Institution

2026 Applied Learning Programme (ALP) on
Autonomous Car and Immersive Technologies (AR & VR)

Preface

This programme introduces you to exciting areas of modern technology through hands-on and project-based learning. You will explore Artificial Intelligence, autonomous systems, and immersive technologies by designing, building, and programming real working solutions. Through guided challenges, you will develop problem-solving, computational thinking, and creative skills while learning how technology is used in real-world applications.

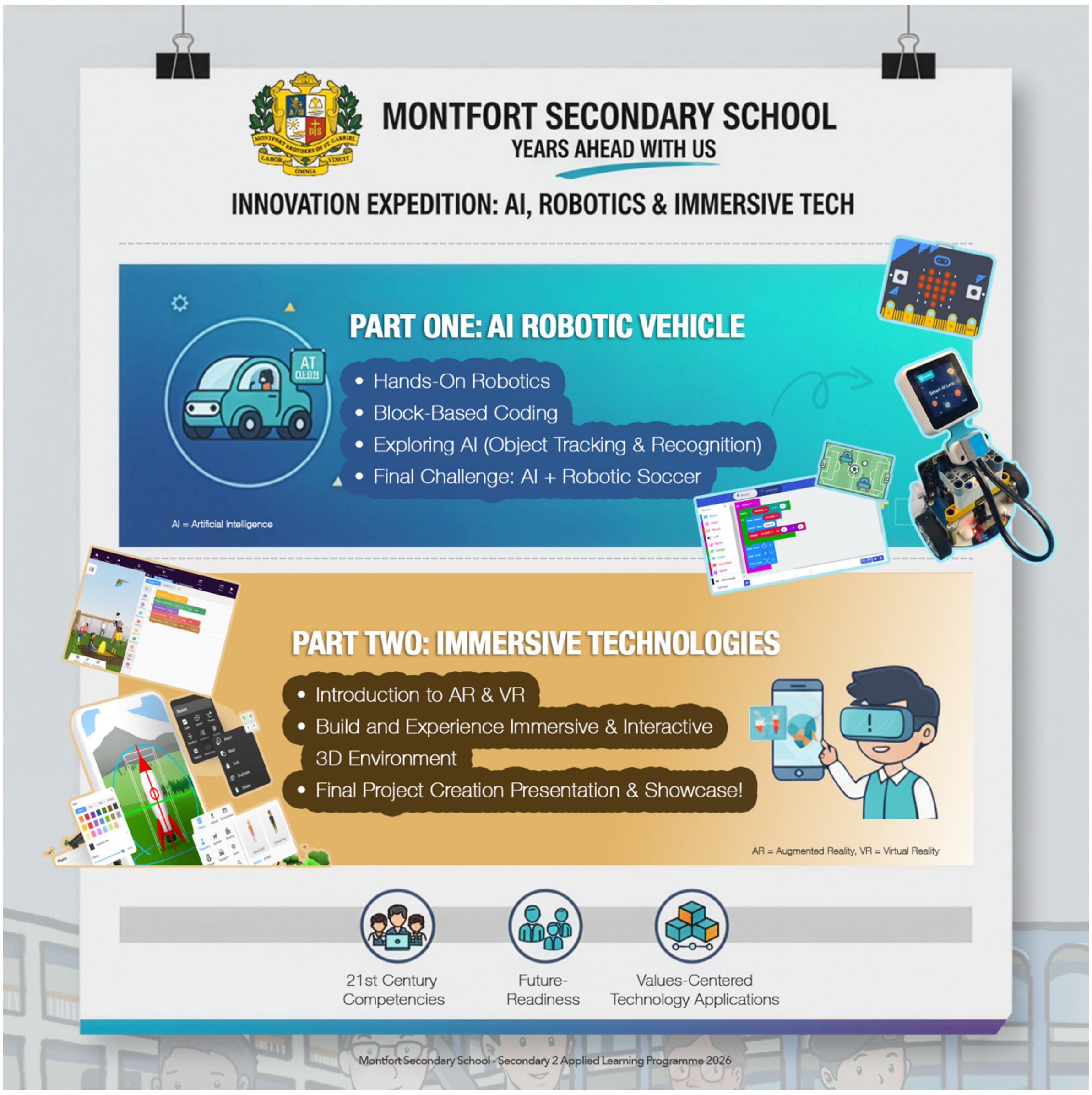
You will begin by creating and programming an autonomous vehicle using AI technology, before moving on to design Augmented Reality (AR) and Virtual Reality (VR) experiences that combine creativity with technical skills. Throughout the programme, you will learn to use technology responsibly and ethically, and apply your learning in a final showcase that demonstrates both your technical ability and creative thinking.

Student Training Guide

Version A1.2

(updated as of 11 January 2026)

Programme Overview



MONTFORT SECONDARY SCHOOL

YEARS AHEAD WITH US

INNOVATION EXPEDITION: AI, ROBOTICS & IMMERSIVE TECH

PART ONE: AI ROBOTIC VEHICLE

- Hands-On Robotics
- Block-Based Coding
- Exploring AI (Object Tracking & Recognition)
- Final Challenge: AI + Robotic Soccer

AI = Artificial Intelligence

PART TWO: IMMERSIVE TECHNOLOGIES

- Introduction to AR & VR
- Build and Experience Immersive & Interactive 3D Environment
- Final Project Creation Presentation & Showcase!

AR = Augmented Reality, VR = Virtual Reality

21st Century Competencies

Future-Readiness

Values-Centered Technology Applications

Montfort Secondary School - Secondary 2 Applied Learning Programme 2026

In this programme, you will dive into two powerful areas of modern technology: Artificial Intelligence & Autonomous Systems, and Augmented & Virtual Reality (AR/VR). Through guided challenges and creative projects, you won't just learn about technology — you will build, develop functionality, and create with it.

🚗 Part 1: AI & Autonomous Vehicles

You will begin by learning how intelligent systems work by building and programming your own autonomous vehicle using the Micro:bit Cutebot with AI lens technology. Step by step, you will take on challenges such as navigating routes, detecting objects, tracking movements, and competing in robotic soccer-style tasks. Along the way, you will apply computational thinking, problem-solving, and basic engineering concepts to make your robot smarter and more reliable.

🌐 Part 2: AR & VR Development

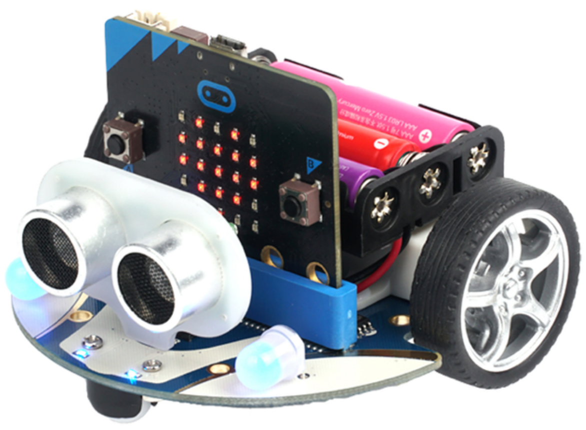
Next, you will enter the world of immersive technologies. Using CoSpaces Edu, you will learn how to create interactive augmented reality experiences and design fully immersive virtual environments. These experiences will challenge you to think creatively about how digital spaces can solve real-world problems and communicate ideas in new ways.

🧠 More Than Just Technical Skills

Throughout the programme, you will develop important 21st Century Competencies, such as critical thinking, creativity, collaboration, and responsible decision-making. You will also explore how technology should be used ethically and responsibly to benefit people and communities.

By the end of this journey, you won’t just be a user of technology — you will be a creator, thinker, and problem-solver, ready to take on future challenges with confidence.

2. Introduction to Robotic Vehicle



Introduction to Cutebot

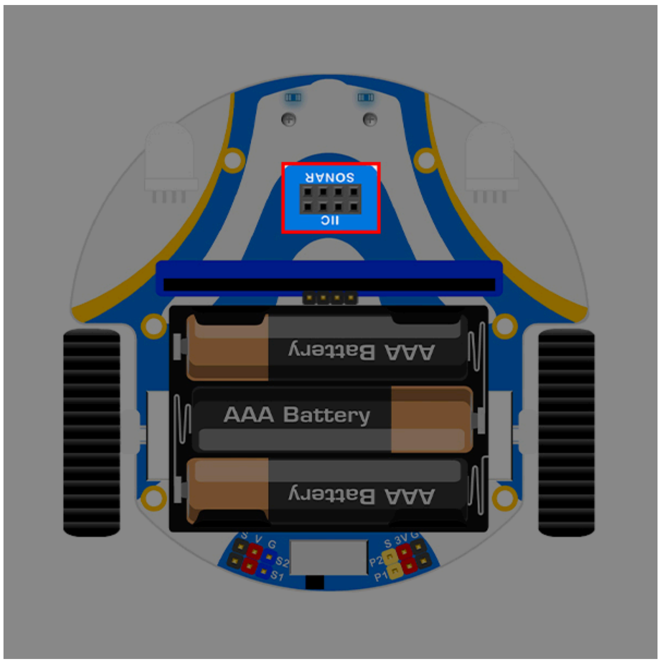
Cutebot is a rear-drive smart car driven by dual high-speed motors.

There are various on-board components on the Cutebot including ultrasonic sensor and distance sensor, two RGB LED headlights and clearance lamps on the bottom, two line-tracking probes, an active buzzer as the horn. It also has the capability to add external components to extend its functionalities.

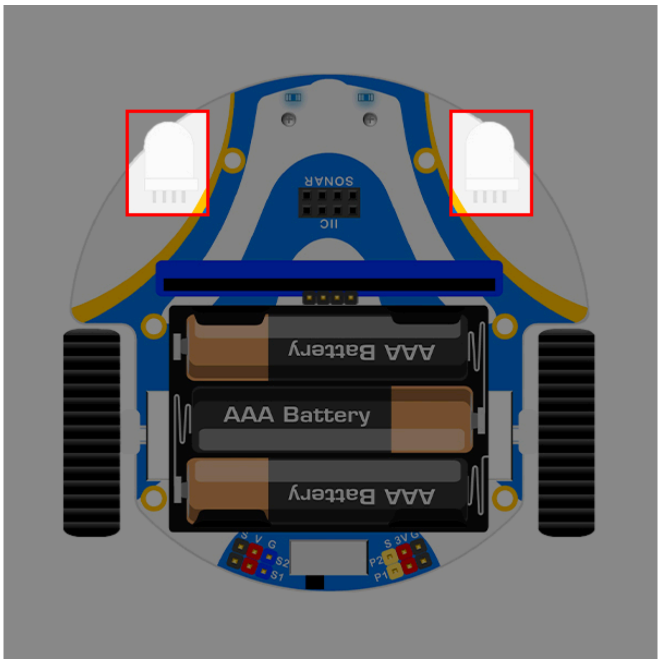
Note: Your hardware might be with an extension that enhances the cutebot. The appearance of the vehicle is different from the basic version.

On-board Components

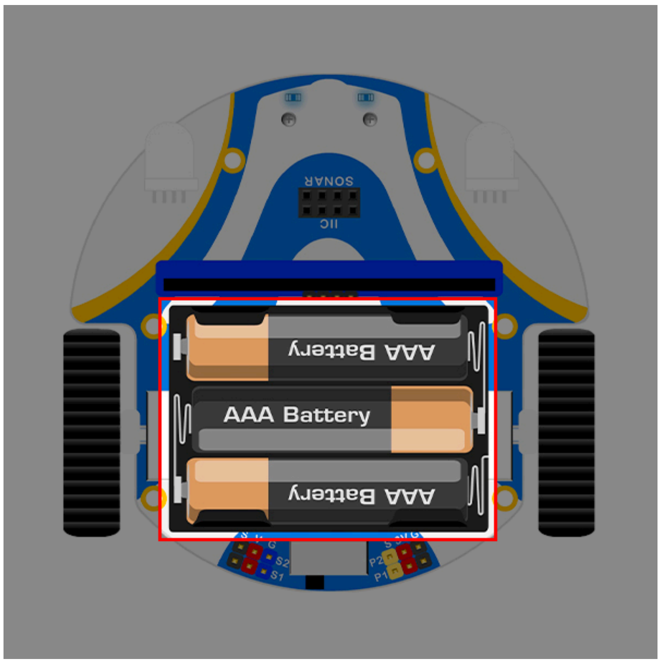
Ultrasonic connection and micro:bit IIC port are placed in the front part of the Cutebot.



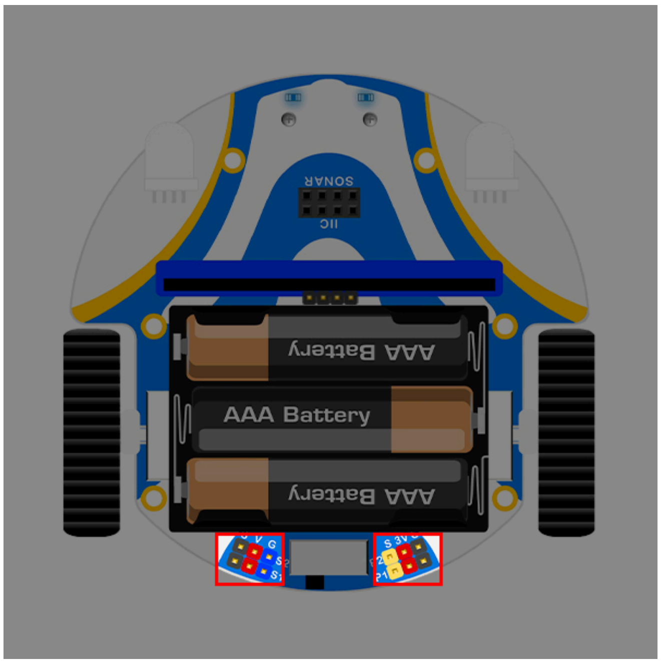
Two full color RGB lights controlled by the expansion board are placed on both side of the front part.



An expansion board for 3x AA batteries is placed in the right above part of the Cutebot.



The IIC port and Servo Port(S1, S2), P1,P2 IO connections are equipped in the battery expansion board.

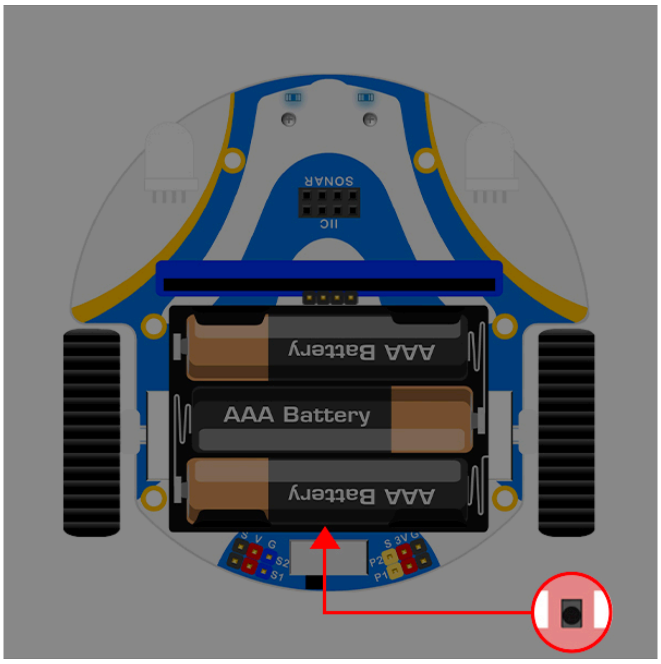


What do you think are the functions of the various pins and ports?

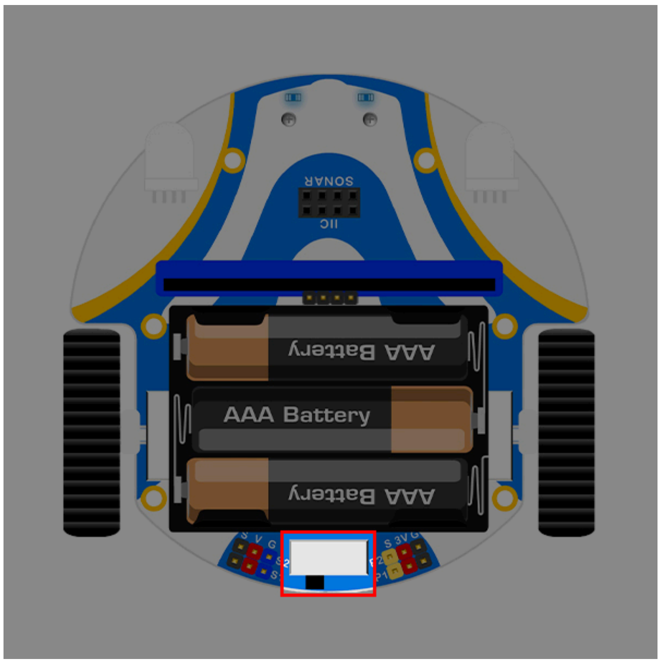
____

____

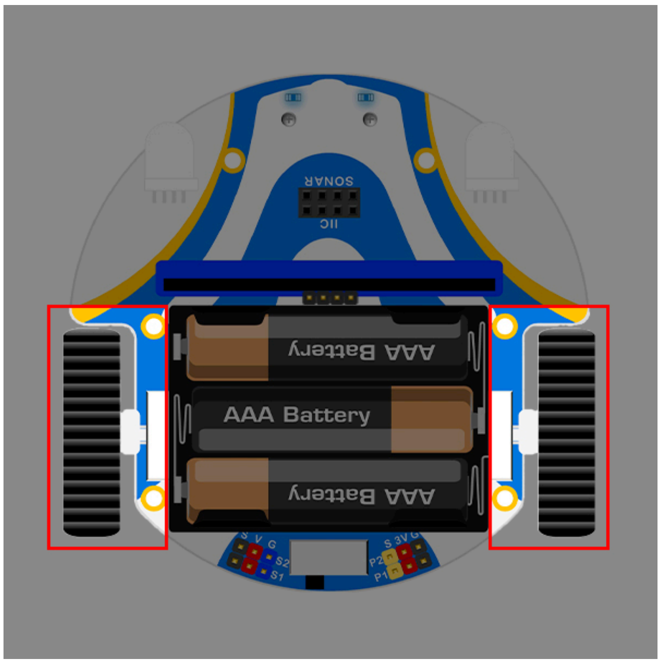
The infrared probes connecting to P16 port of the micro:bit are placed on the tail part of the Cutebot.



The master switch is placed besides the infrared probes and with on/off status showing by the LED.

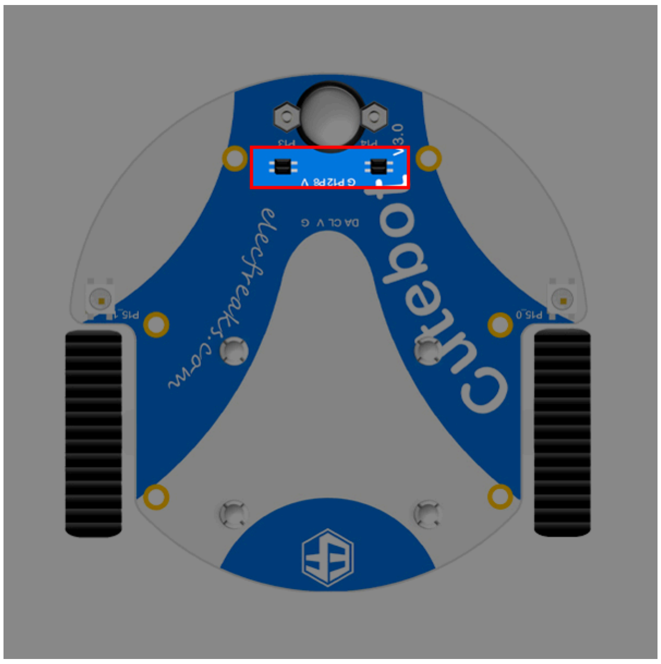


The two wheels on both side are driven by DC micro gear deceleration motors(300 RPM).



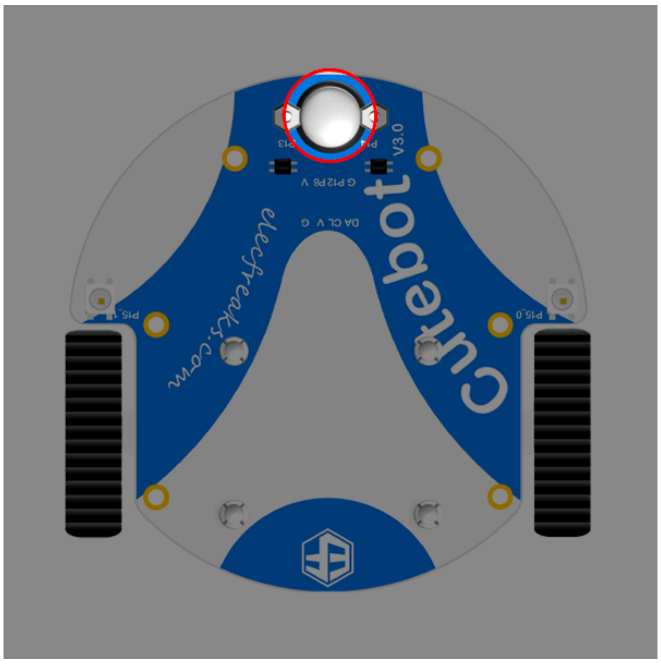
What do you think is highlighted in the image?

____

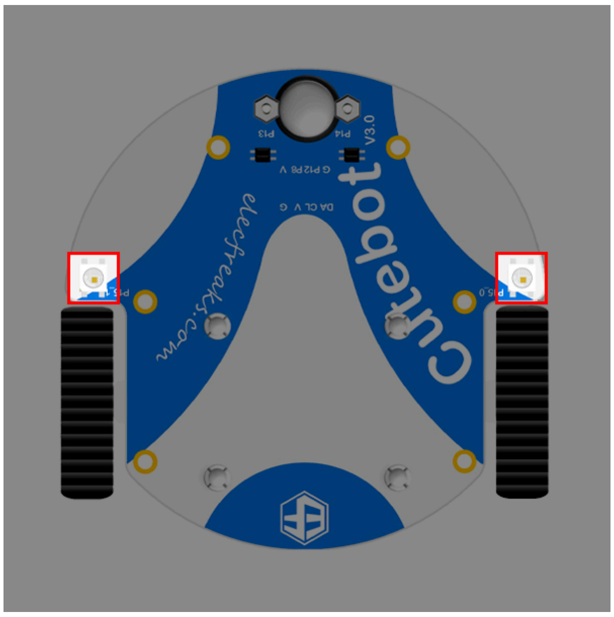


Why do you think an universal wheel is used here?

____



The two full color Rainbow LEDs programmed by Rainbow LED connecting to P15 on the micro:bit is placed on both bottom side of the Cutebot and can be used as the clearance lamps or others.



Note: Check that all the components on your robotic vehicle are correct and seek help from your trainer if any parts are missing or damaged.

Now that you understand the basics of the robotic vehicle, let's move on to adding the extension and AI lens!

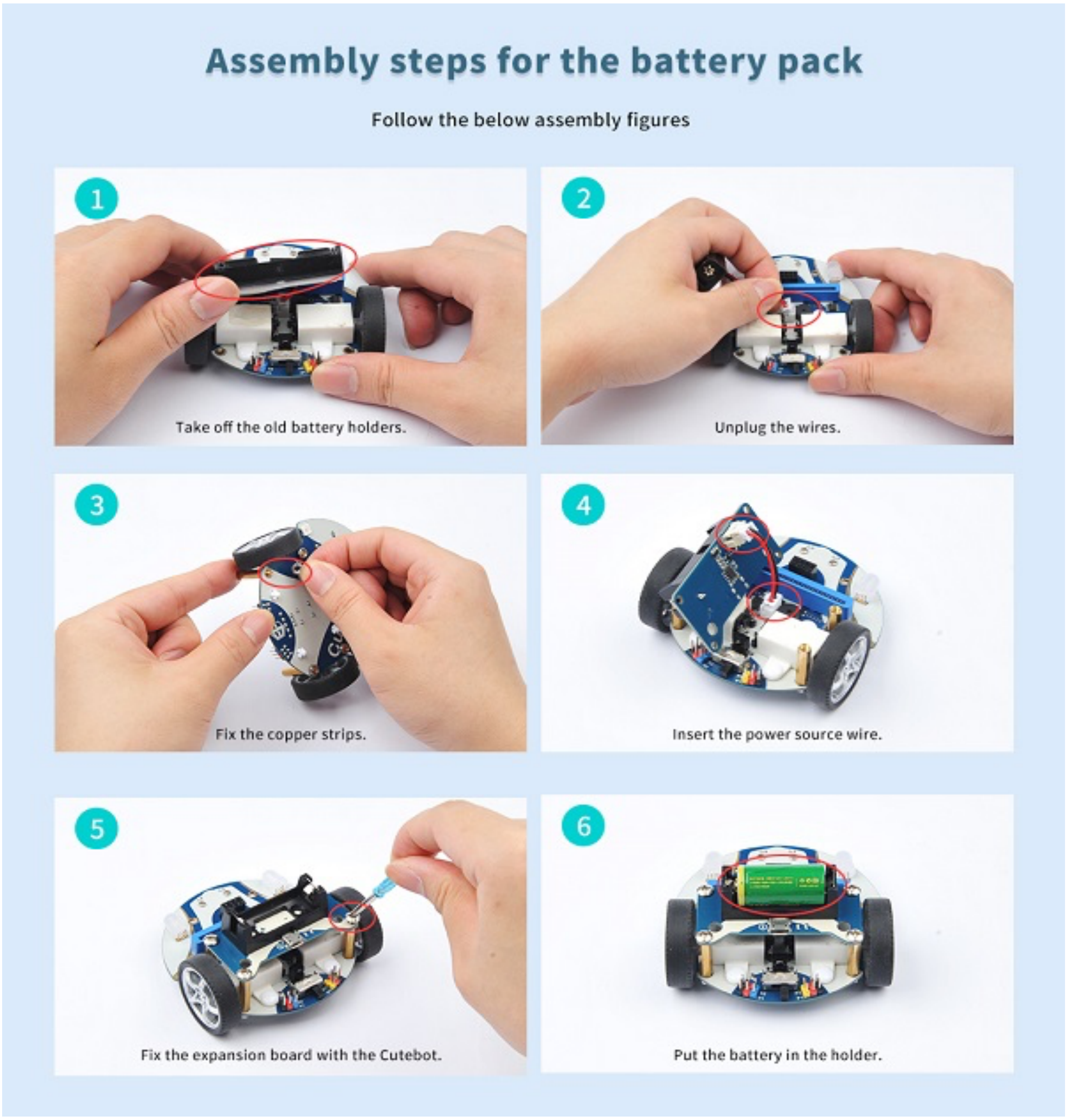
What new functions do you think the Cutebot can perform when the AI lens is added?

____

____

2.1. Adding Extensions

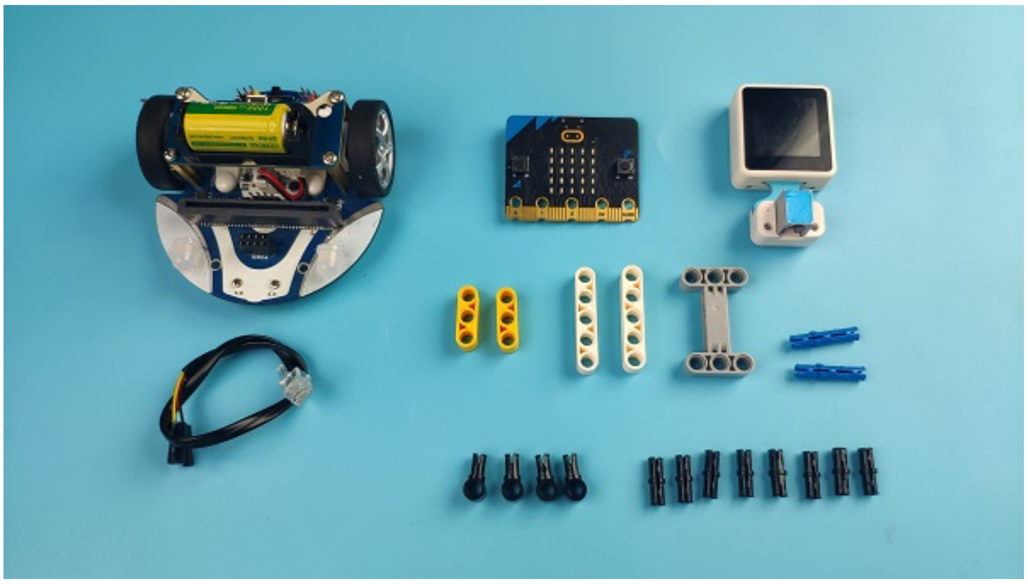
<u>Assembling the battery pack extension</u>


Assembly steps for the battery pack
Follow the below assembly figures
1
Take off the old battery holders.
2
Unplug the wires.
3
Fix the copper strips.
4
Insert the power source wire.
5
Fix the expansion board with the Cutebot.
6
Put the battery in the holder.

Note: Your kit may have already been assembled as shown above. Please refer to Image 6 for the final appearance.

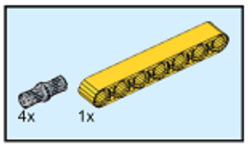
Assembling the AI lens and bricks onto the Cutebot

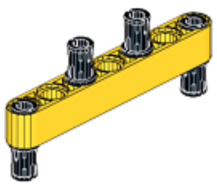
Parts needed. Please ensure you have all the following parts before proceeding.



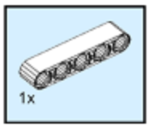
Note: We will complete the assembling first before diving into the learning of microcontroller (micro:bit) and AI lens.

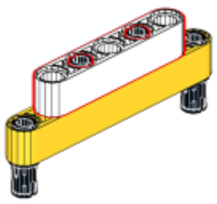
1



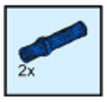


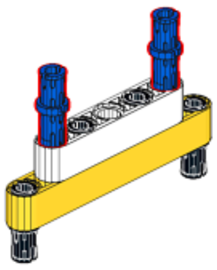
2



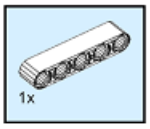


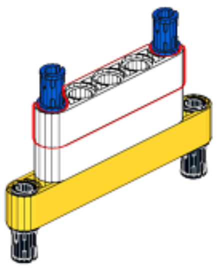
3



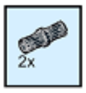


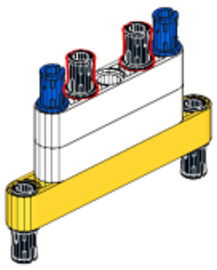
4

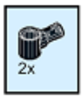


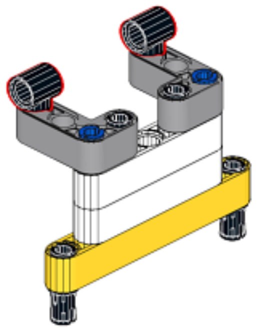


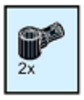
5

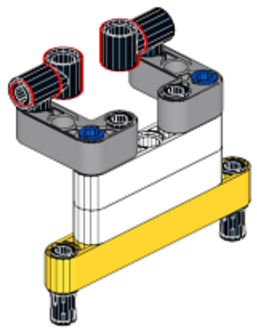




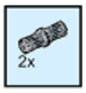
7


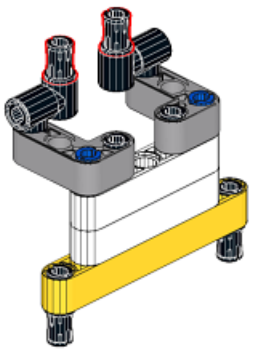


8


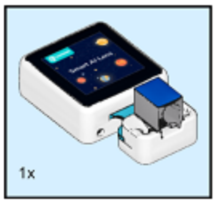


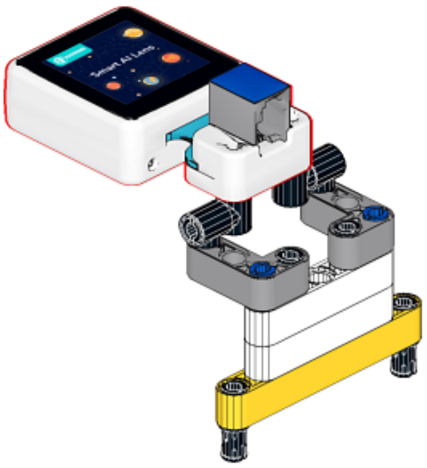
9



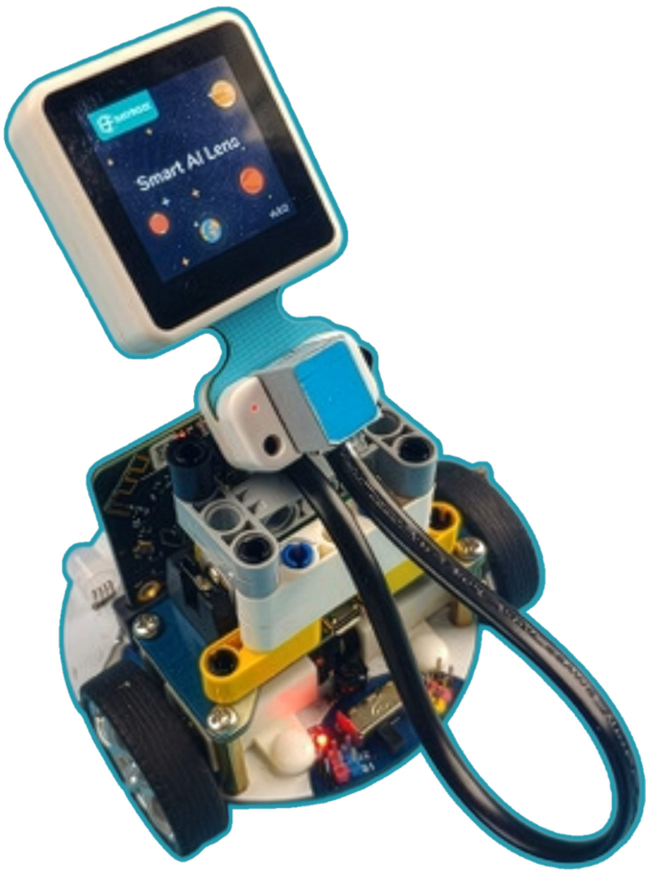


10





Assembly Completion:



Safety notes:

a. Do not use unnecessary force when handling with wires and pins

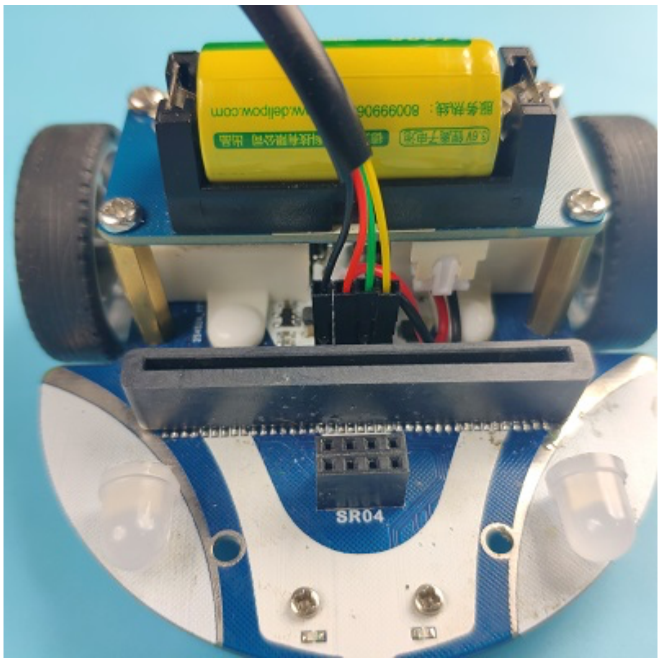
b. Handle the battery with care

c. If there are any missing parts or components, please inform the trainer/teacher in-class

d. The wire of the AI lens may cause the cutebot to go off its designed path due to balancing. Please secure the wire appropriately

Connections of the AI lens to Cutebot

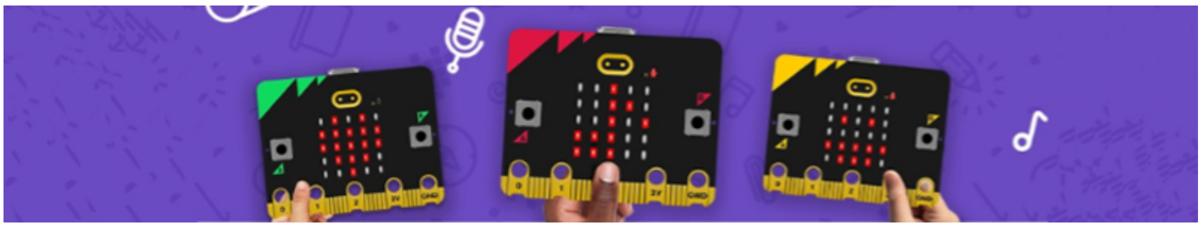
Connect the RJ11 cable with the AI Lens and the other end in Dupont connection to the pins as the picture below (make sure you connect to the right connections; **the wires should be in the same order**).



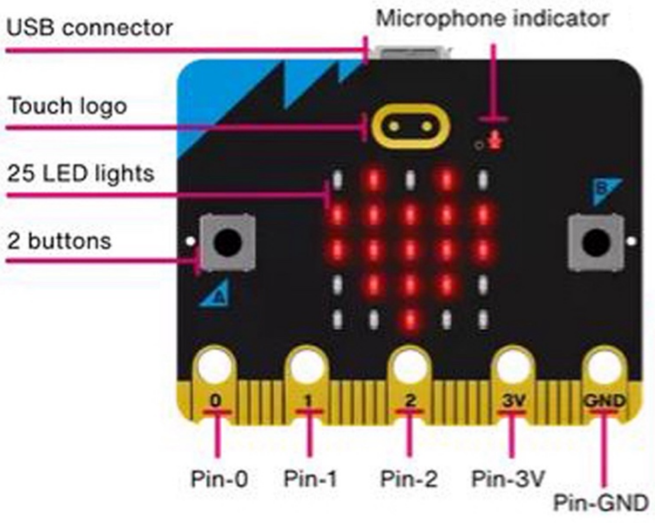
Tips: the bricks holder here is flexible to be adjusted, we may manually adjust the angles of the AI lens to meet the requirements of the functions that you want to achieve.

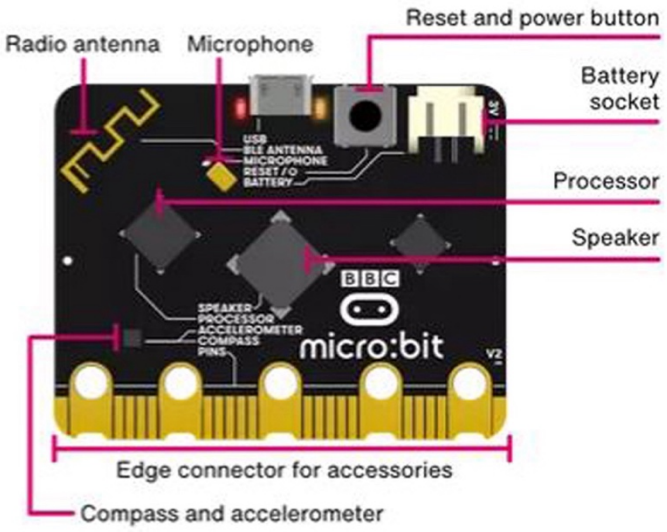
Note: For simple assembling during upcoming sessions, only remove the bricks that are connecting the AI lens to the cutebot when you are keeping it back to the container. You do not need to remove the connections.

2.2. Introduction to Micro:bit



The BBC micro:bit is a pocket-sized computer that introduces you to how software and hardware work together. It has an LED light display, buttons, sensors and many input and output features that, when programmed, let it interact with you and your world.





Display

An LED, or light-emitting diode is an output that gives off light. Your micro:bit has a display of 25 LEDs for you to program.

User Buttons

Buttons are a common input device. Your micro:bit has two buttons you can program, and a reset button.

Accelerometer

An accelerometer is a motion sensor that measures movement. The accelerometer in your BBC micro:bit detects when you tilt it left to right, backwards and forwards and up and down.

Temperature sensor

An input device that measures temperature. Your BBC micro:bit has a temperature sensor inside the processor which can give you an approximation of the air temperature.

Light sensor

A light sensor is an input device that measures light levels. Your BBC micro:bit uses the LEDs to sense the levels of light and lets you

Compass

A digital compass is an input sensor that detects magnetic fields. Your BBC micro:bit has an inbuilt compass that can detect program your micro:bit as a light sensor.

the direction in which it is facing.

Touch logo

The touch logo uses capacitive touch, sensing tiny changes in electrical fields to know when your finger is pressing it - just like your phone or tablet screen.

Speaker

A built-in speaker which makes it easy to add sound to your projects. Make your micro:bit giggle, greet you or let you know when it is sleepy or sad.

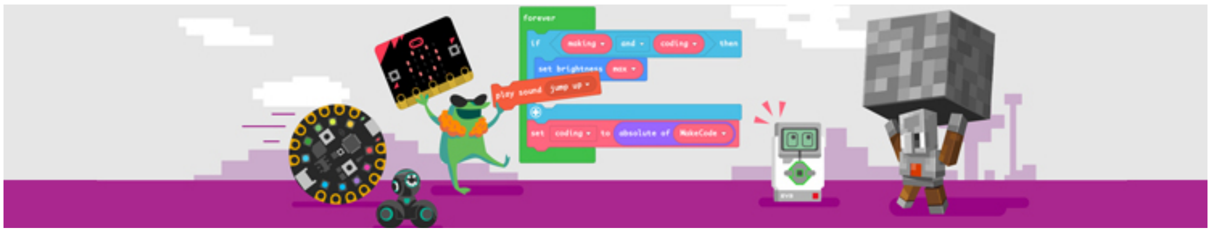
Microphone

Simple input such as make it turn the lights on when you clap. It can also measure the amount of sound, so you can make a noise level meter or disco lights that beat in time with music.

Radio

Radio is a way of sending and receiving messages and BBC micro:bits can use radio waves to communicate with each other.

2.3. Block-based Programming Environment

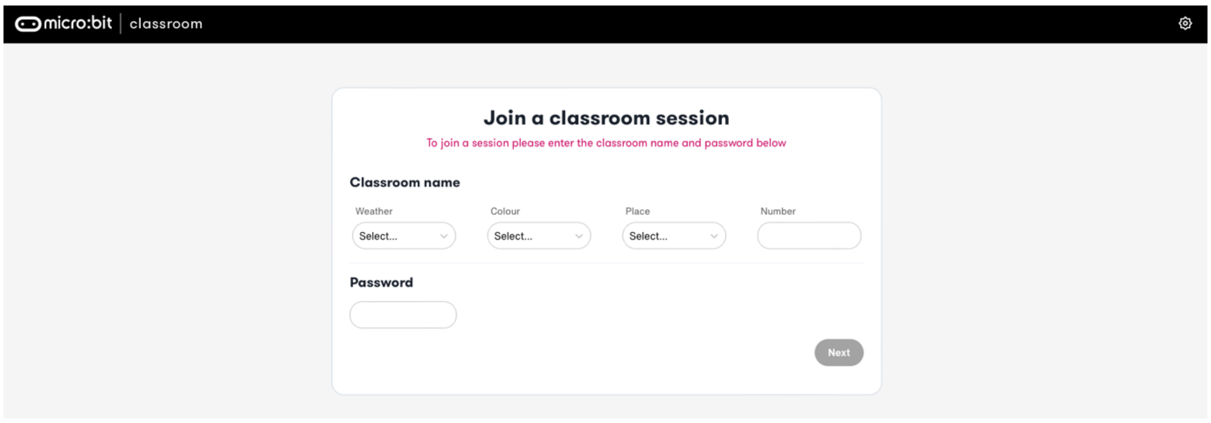


MakeCode Editor for micro:bit is one of the most widely use graphical programming environment from the micro:bit website. It is an open source project developed by Microsoft. Program your micro:bit with the editor and download the code into it view it in action!

In the following instructions, we are using micro:bit classroom to better facilitate the programme and for your learning experience. If you are exploring micro:bit projects yourself and is not in a classroom environment, please use makecode editor instead.

STEP ONE: Open micro:bit classroom

Visit the micro:bit classroom editor page from the following link: https://microbit.org/join



micro:bit | classroom

Join a classroom session

To join a session please enter the classroom name and password below

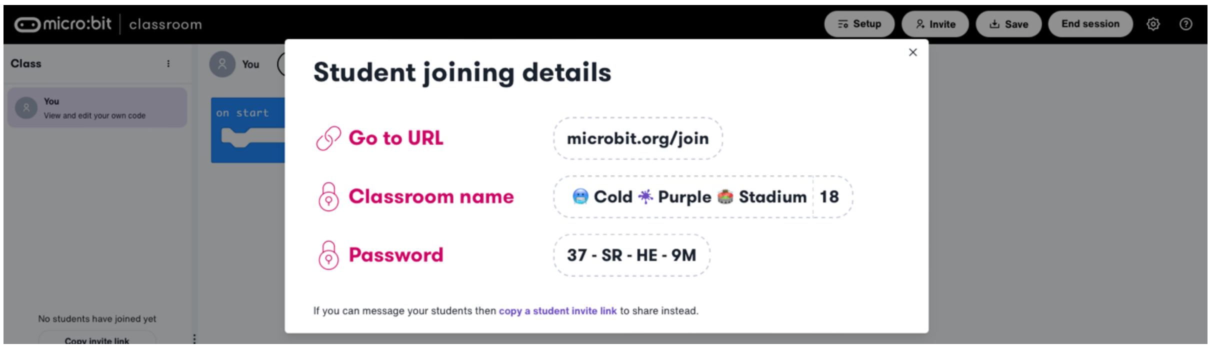
Classroom name

Weather	Colour	Place	Number
Select...	Select...	Select...	

Password

STEP TWO: Join your classroom

Select the information and key in the password to join into your class's micro:bit classroom.



Student joining details

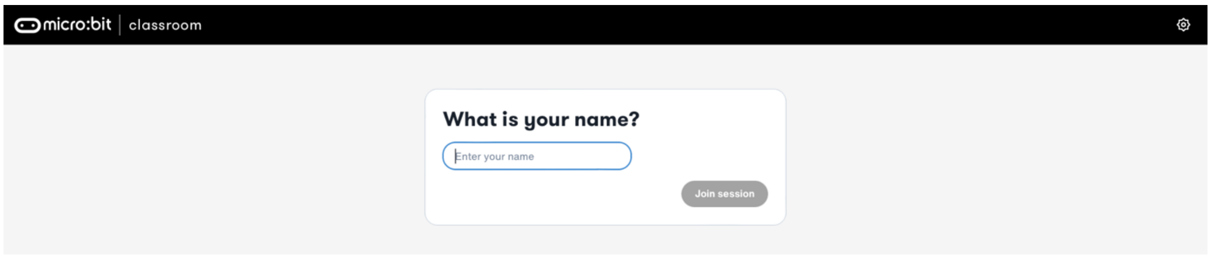
Go to URL	microbit.org/join
Classroom name	Cold Purple Stadium 18
Password	37 - SR - HE - 9M

If you can message your students then copy a student invite link to share instead.

Note: Your classroom name and password will be different as above

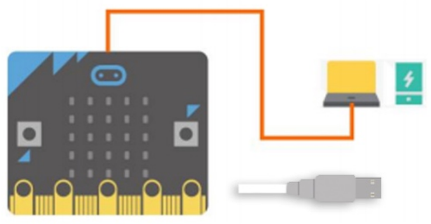
STEP THREE: Key in your name

Provide your full name and click on 'Join session'



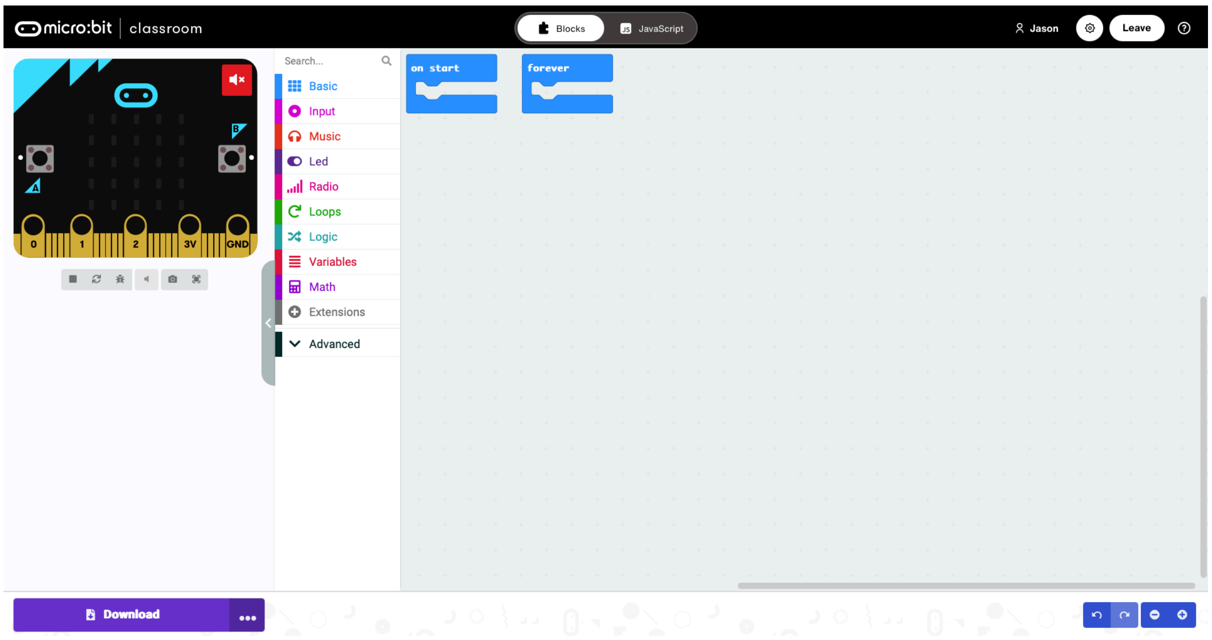
STEP FOUR: Connect micro:bit to computer

Connect the micro:bit to the laptop using the micro-usb cable



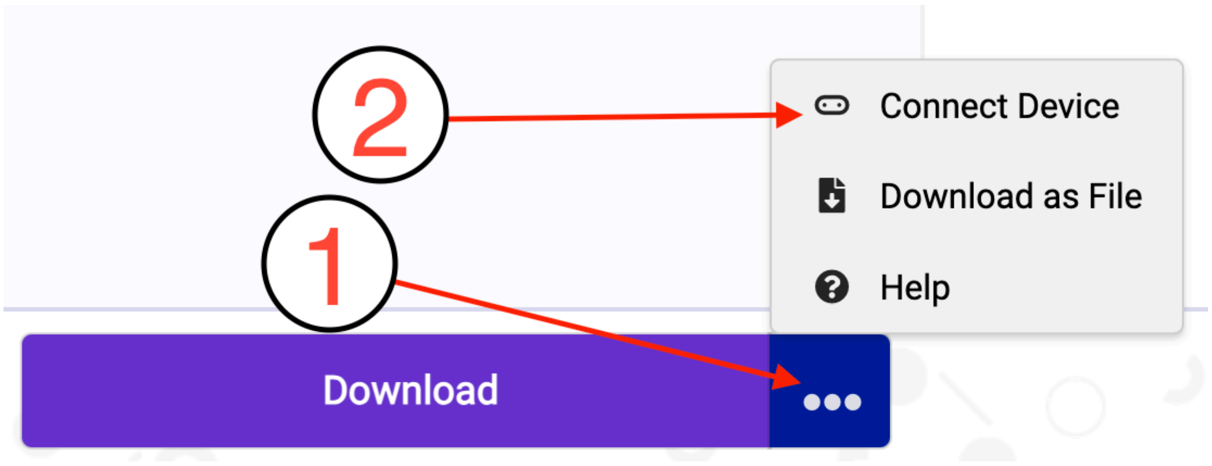
STEP FIVE: Editor interface

Explore and familiarise with the interface

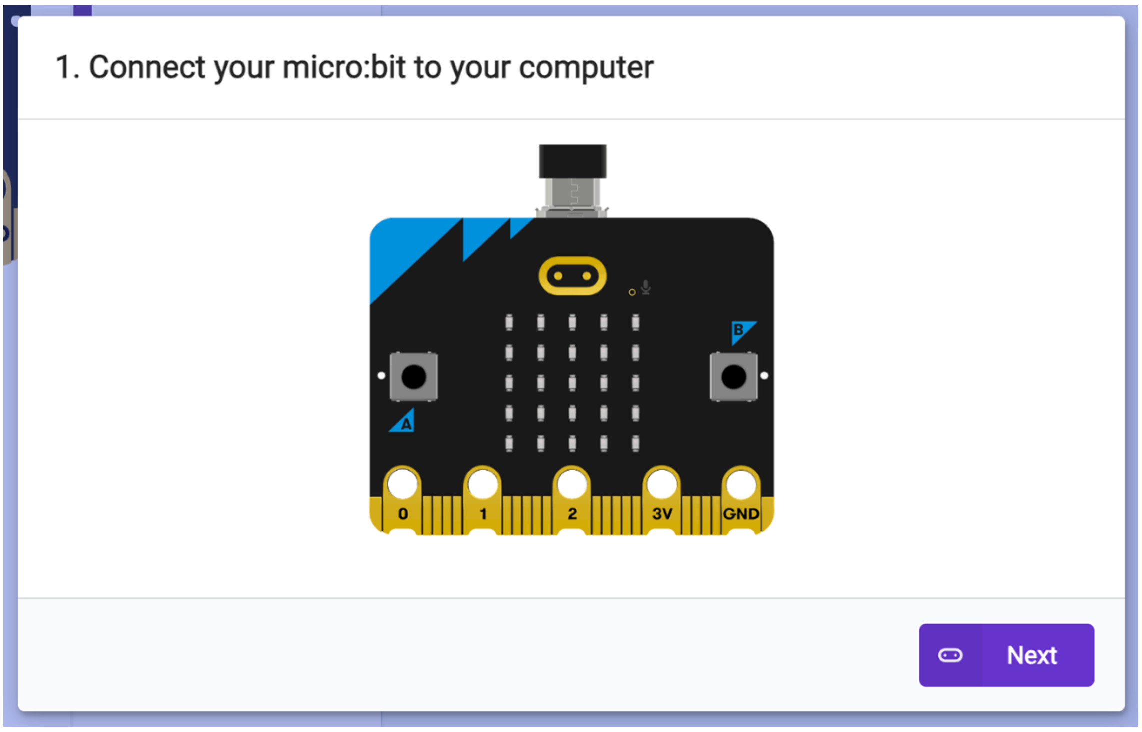


STEP SIX: Download the code

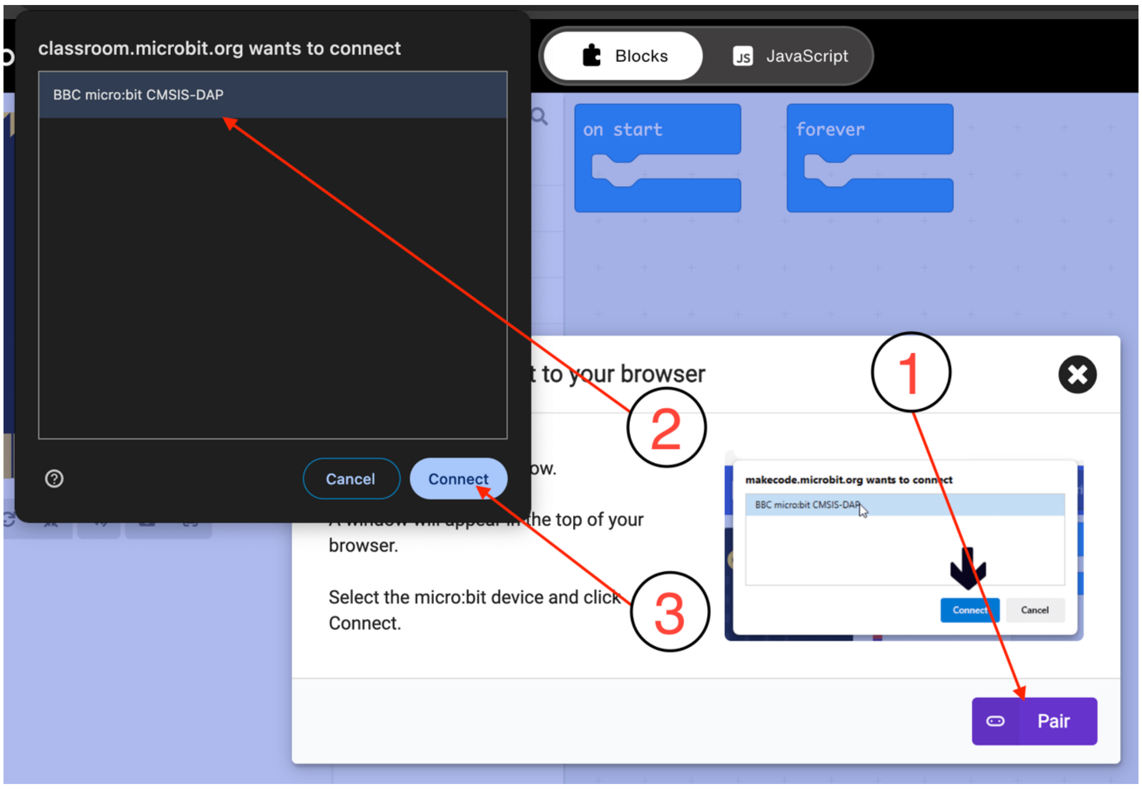
Compiling and downloading program into your micro:bit



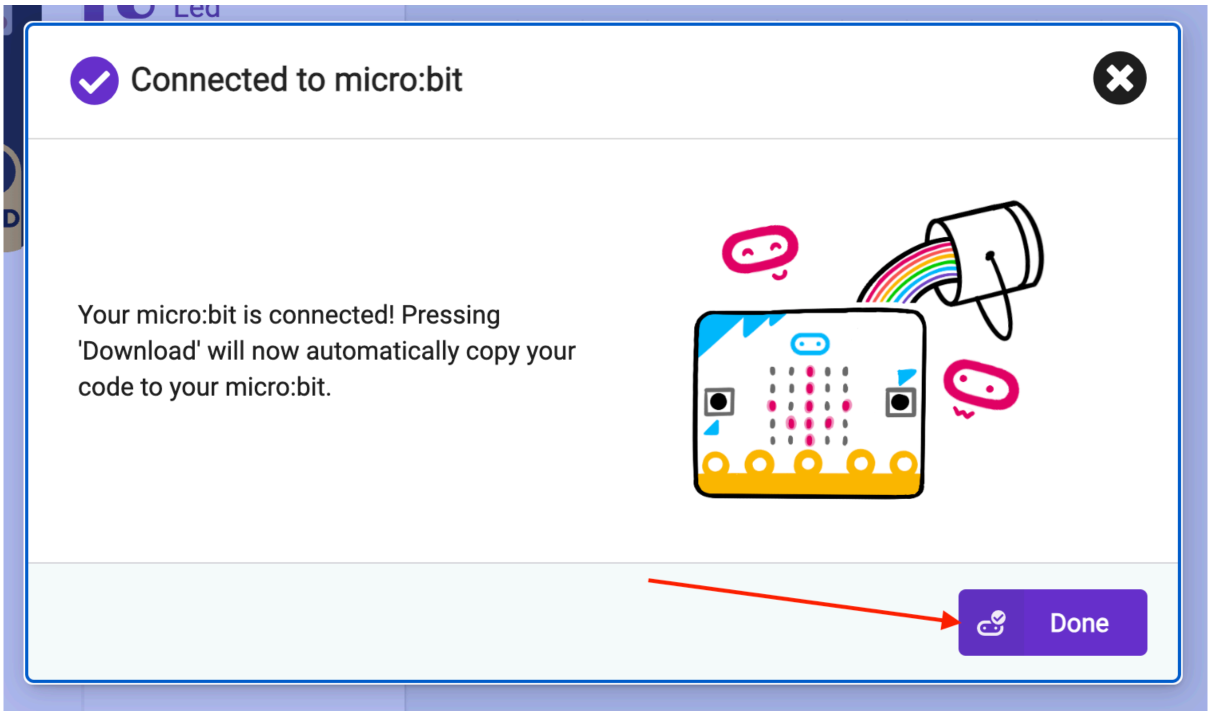
Click on the three dots beside the download button and then connect device



A popup will appear, click next

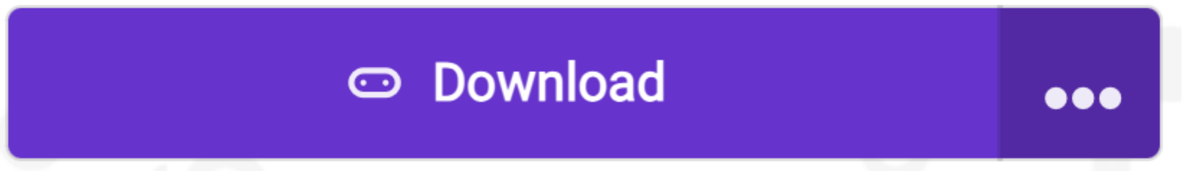

classroom.microbit.org wants to connect
BBC micro:bit CMSIS-DAP
Cancel
Connect
Blocks
JavaScript
on start
forever
t to your browser
ow.
A window will appear at the top of your browser.
Select the micro:bit device and click Connect.
makecode.microbit.org wants to connect
BBC micro:bit CMSIS-DAP
Connect
Cancel
Pair
1
2
3

Click on pair and select the micro:bit on the popup and connect


Connected to micro:bit

Your micro:bit is connected! Pressing 'Download' will now automatically copy your code to your micro:bit.

Click on done after connected successfully



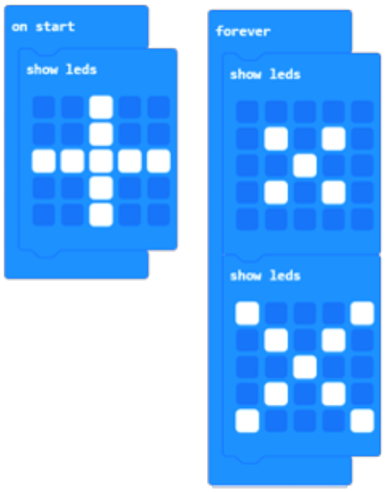
Whenever the **<u>same</u>** micro:bit is connected and paired successfully, there will be a micro:bit icon beside the download label.

Pairing is only needed **<u>once</u>**, after the first successful pairing, your laptop should automatically detect and pair the same micro:bit to the editor.

Troubleshooting: if micro:bit does not automatically pair, please check if it is the same micro:bit that was originally paired. Else please initiate the pairing starting from step four.

2.4. Create your First Code!

Program the micro:bit to display the following LED image.



Viewing Results

1. Look at the [View Results] area and see the results. What do you notice? Can you explain why the results are like this?

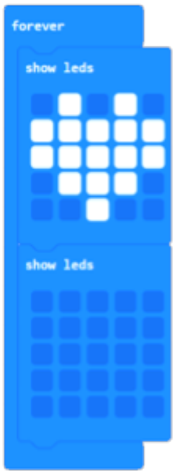
____

____

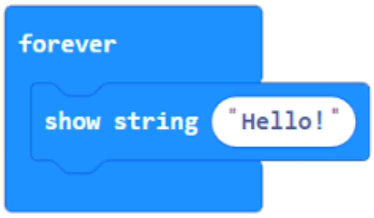
2. Upload the code to the micro:bit. Do you still see the same results?

☐ Yes ☐ No

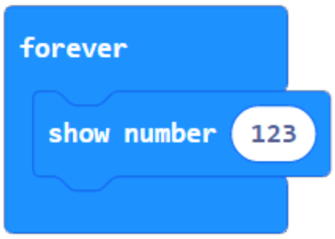
2.5. Blinking, Scrolling Text & Display Number



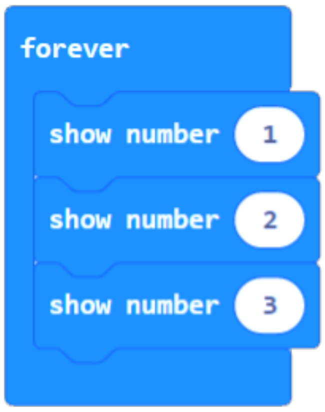
```
forever
    show leds
        . # . # .
        # # # # #
        # # # # #
        . # # # .
        . . # . .
    show leds
        . . . . .
        . . . . .
        . . . . .
        . . . . .
        . . . . .
```



```
forever
    show string "Hello!"
```



```
forever
    show number 123
```



```
forever
    show number 1
    show number 2
    show number 3
```

Create your own unique micro:bit animation.

Try out: Combine blocks

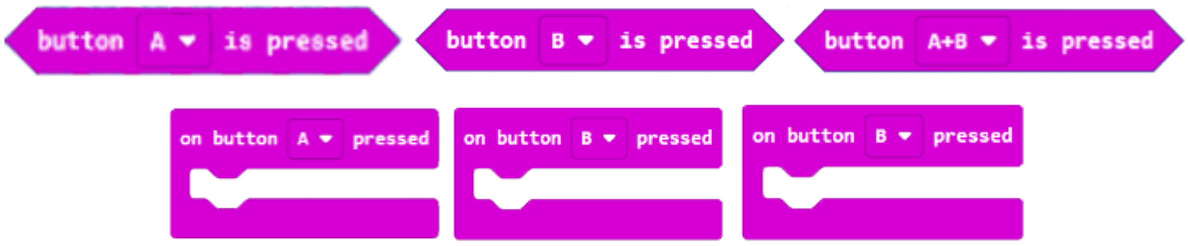
1. Can you combine the different blocks to show icons, text & numbers?

☐ Yes, it will work!

☐ No, it won't work because micro:bit does not know which to block to display first

2.6. Button State and Event

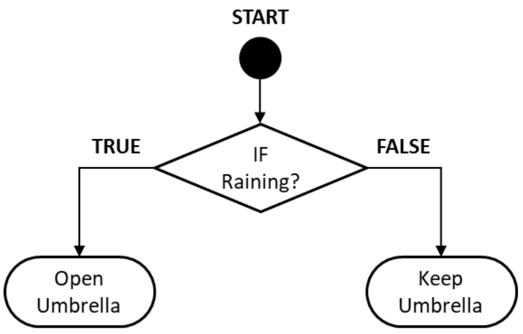
Buttons are a common **input** device. Your micro:bit has two buttons you can program.

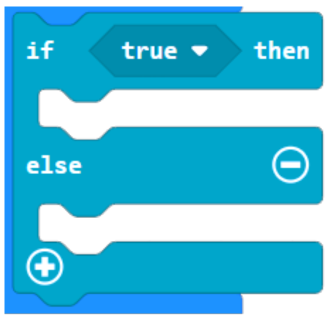


2.7. Selection/Conditional Statements

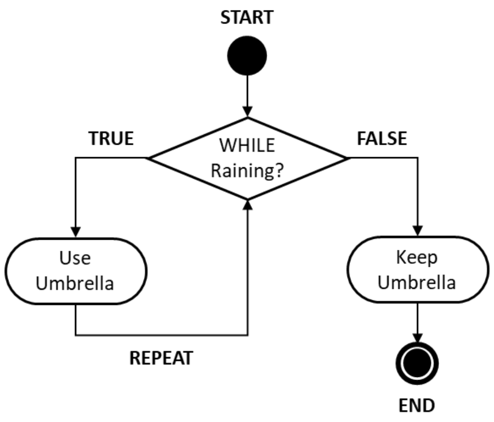
Conditionals are a simple way to tell the micro:bit to perform an **action** upon a **certain condition** being met.

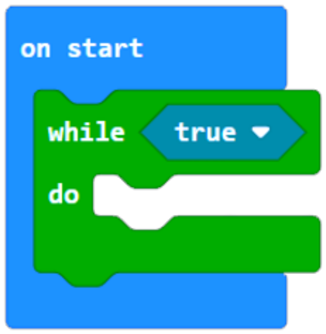
If . . . else Statements





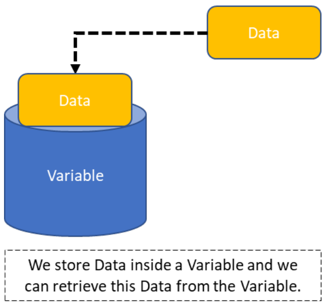
Conditional Loops

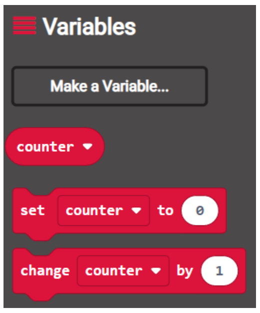




2.8. Variables

Variables are used to **virtually** hold and **manipulate** a numeric value or string stored in the micro:bit.


Data
Data
Variable
We store Data inside a Variable and we can retrieve this Data from the Variable.



Try out: Think!

1. Tick the options below that you think uses variables in real-life application.

☐ A) A digital thermometer displaying the changing temperature throughout the day.

☐ B) A traffic light that switches colours in a fixed cycle without tracking time

☐ C) A score counter in a game that increases each time the player collects a coin.

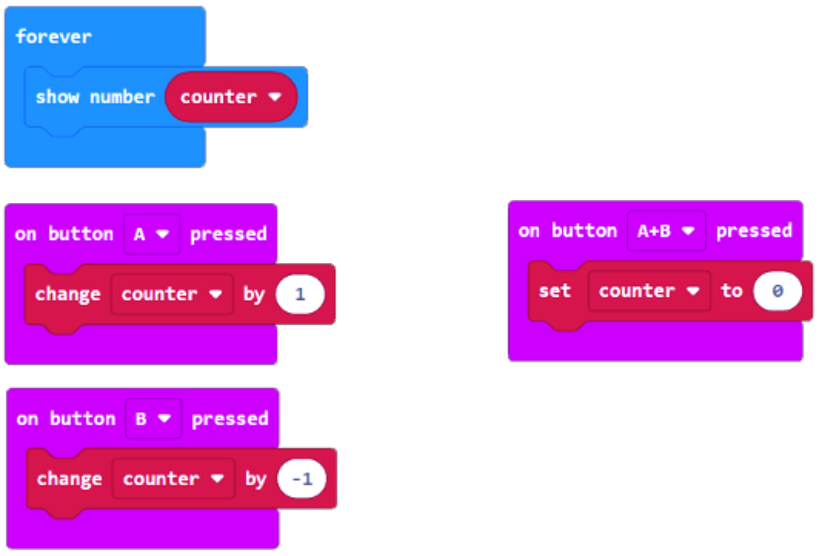
☐ D) A button that always plays the same sound when pressed.

☐ E) An app that asks your name and greets you personally using the name you typed.

☐ F) A quiz app that keeps track of the number of correct answers you have given.

2.9.1. Activity: Counter (Keeping Score)

Follow the guided instructions to create a micro:bit counter for keeping score. The goal is to increase the counter when button A is pressed, decrease the counter when button B is pressed and to reset the counter to 0 when both buttons A+B are pressed.



```
forever
    show number counter

on button A pressed
    change counter by 1

on button A+B pressed
    set counter to 0

on button B pressed
    change counter by -1
```

Download the code into your micro:bit and view the result.

Try out: Think!

1. How can you improve the counter so that the LED will display a message when the counter exceeds 50? Tick the correct answer.

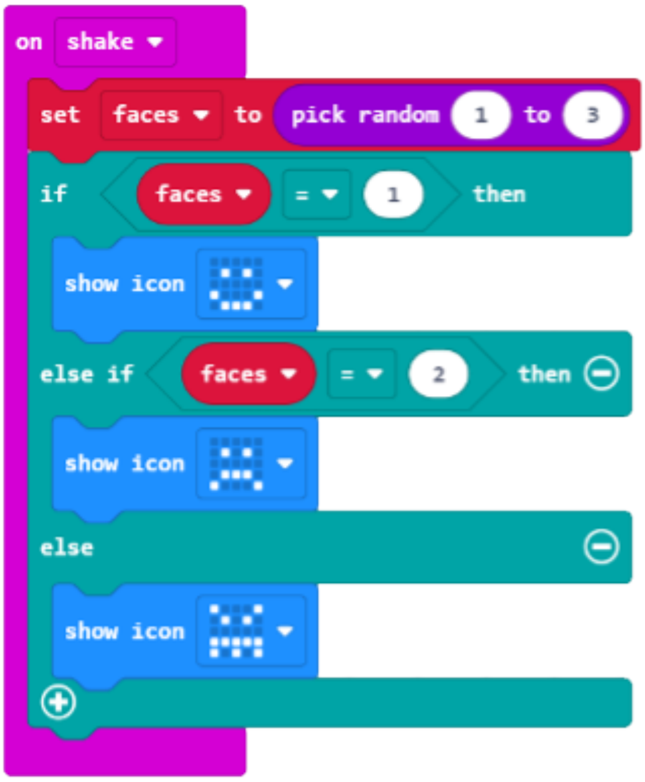
☐ A) Use an 'if' statement to check if the counter is greater than 50 and use show string to display a message on the LED screen.

☐ B) Add more 'show number' blocks until the counter reaches 50, and the message will appear automatically."

2.9.2. Challenge: Happy, Sad and Angry Emoji

Follow the guided instructions to create a random emoji generator.

The goal is to display a random emoji when the micro:bit is shaken.



```
on shake
    set faces to pick random 1 to 3
    if faces = 1 then
        show icon
    else if faces = 2 then
        show icon
    else
        show icon
```

Download the code into your micro:bit and view the result.

Try out: Think!

a) How can I add more face variations?

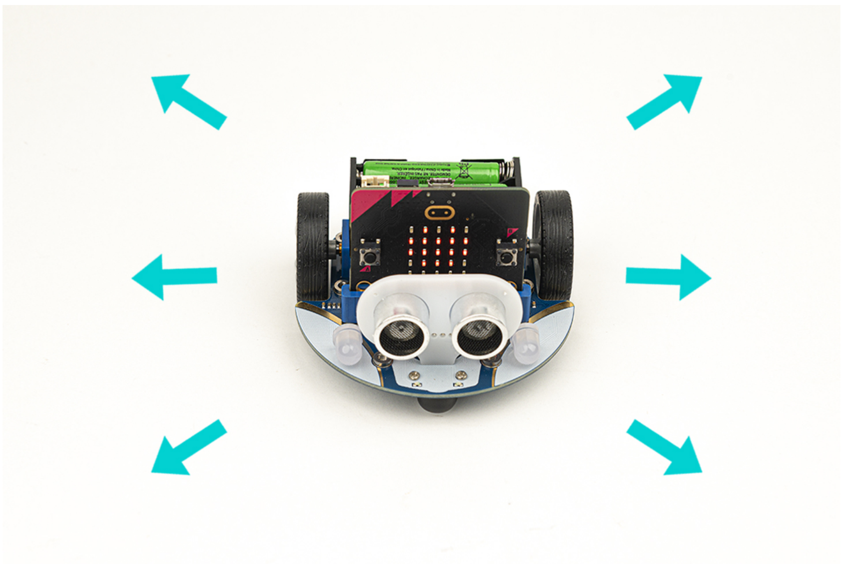
____

____

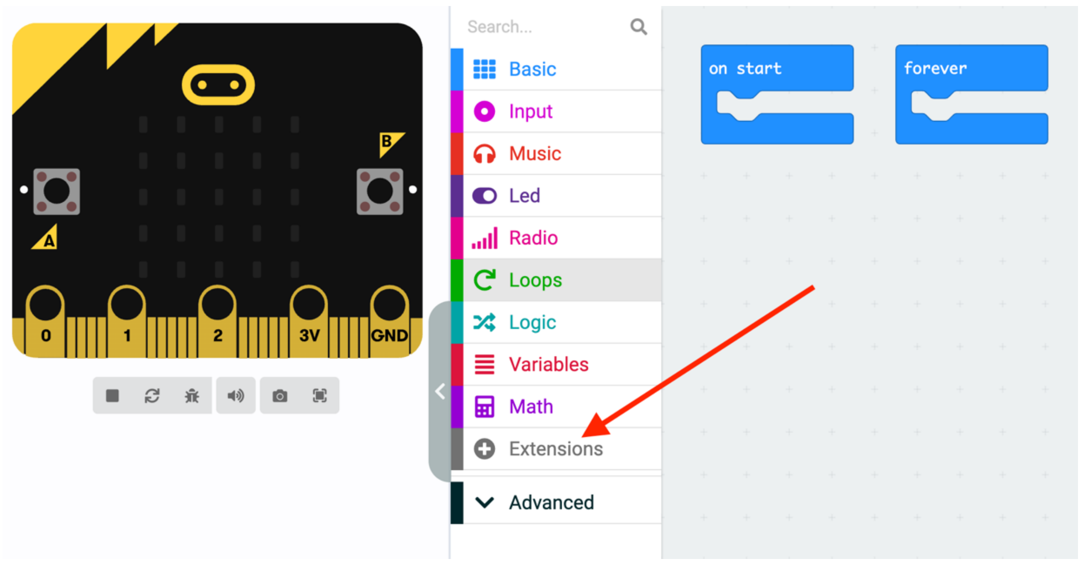
3. Using the Micro:bit with Cutebot

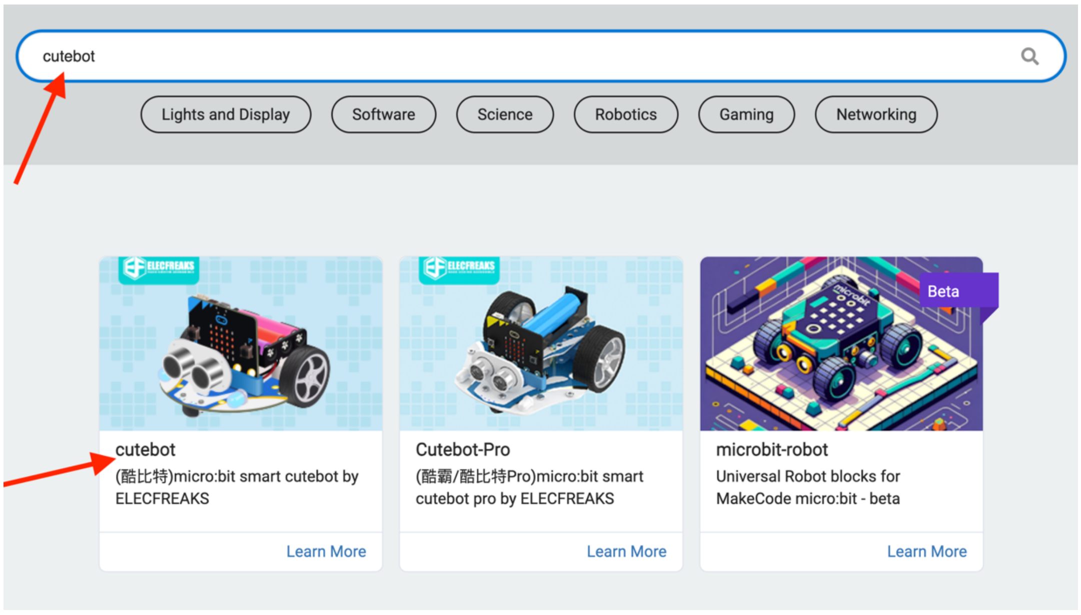
The micro:bit is the brain of the Cutebot; by programming it, you control how the robot thinks, moves, and reacts.

Objective: Make your Cutebot move with programmed code.



a. Visit MakeCode and create a new project. Click on Extensions, search for cutebot and select the first extension





cutebot

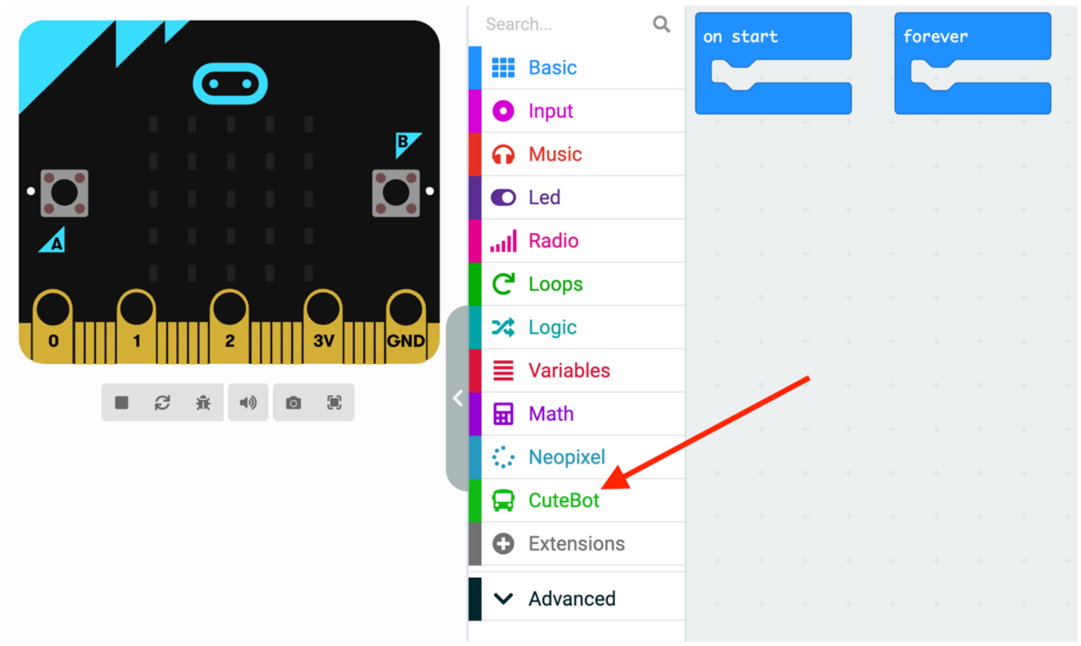
Lights and Display | Software | Science | Robotics | Gaming | Networking

cutebot
(酷比特)micro:bit smart cutebot by ELECFREAKS
Learn More

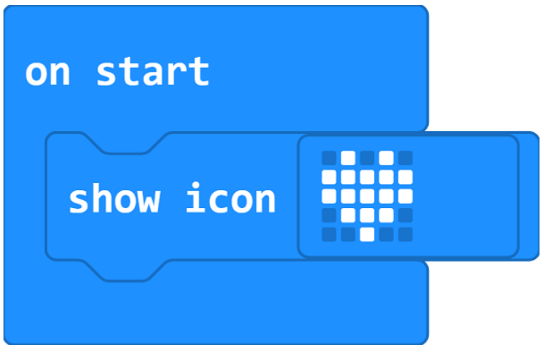
Cutebot-Pro
(酷霸/酷比特Pro)micro:bit smart cutebot pro by ELECFREAKS
Learn More

microbit-robot
Universal Robot blocks for MakeCode micro:bit - beta
Learn More

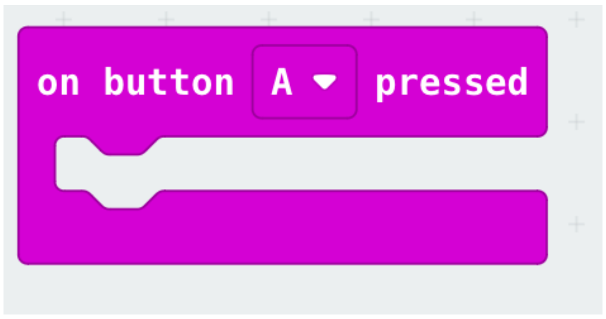
a. A new block category is added



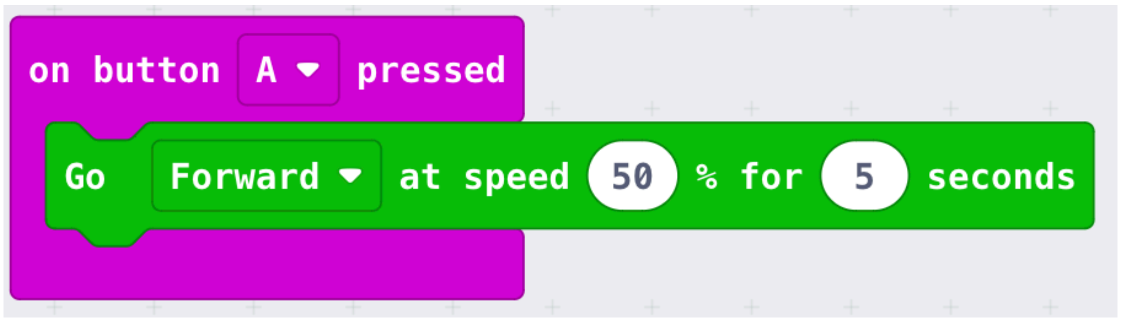
b. Drag [show icon] in to the on-start block



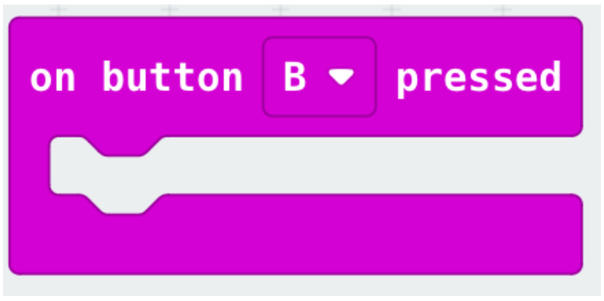
c. Drag the [on button pressed] block on to the working environment and choose ‘A’ as the dropdown selection



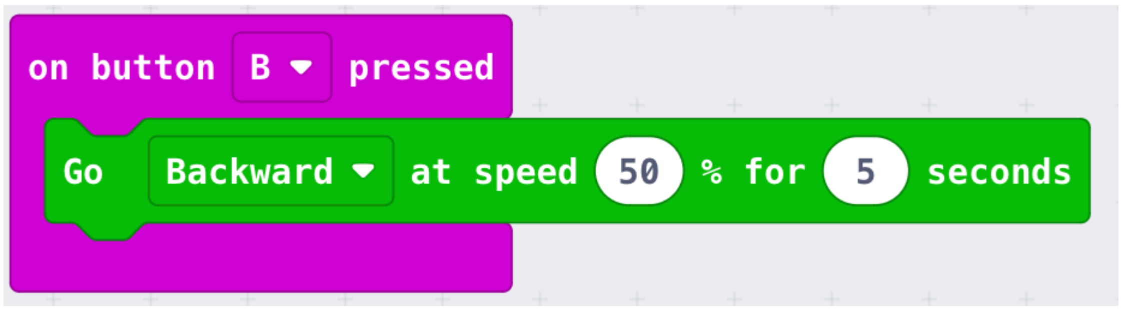
d. Drag the [Go Forward at speed 50% for 5 seconds] block in to the [on button A pressed] block



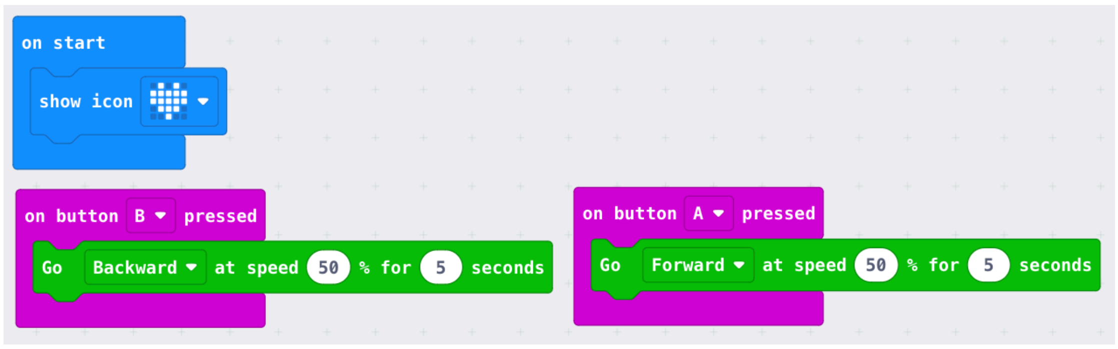
e. Drag the [on button pressed] block on to the working environment and choose ‘B’ as the dropdown selection



f. Drag the [Go Forward at speed 50% for 5 seconds] block in to the [on button A pressed] block and change the selection to ‘Backward’



g. **Final result.** Download the code into your micro:bit and observe the movement on cutebot when respective buttons are pressed.


on start
show icon
on button B pressed
Go Backward at speed 50 % for 5 seconds
on button A pressed
Go Forward at speed 50 % for 5 seconds

Troubleshooting notes:

a. Always **remove** the micro:bit from the cutebot when you are downloading code into it.

b. If the cutebot is unresponsive or AI lens is taking a long time to bootup after the micro:bit is slot in, it might be an error in the code. Please switch off cutebot, remove micro:bit and redownload the code and try again.

c. The wire of the AI lens may cause the cutebot to go off its designed path due to balancing. Please secure the wire appropriately

3.1. Challenge

a) How can you set the speed to be random between 0-80%?

____

____

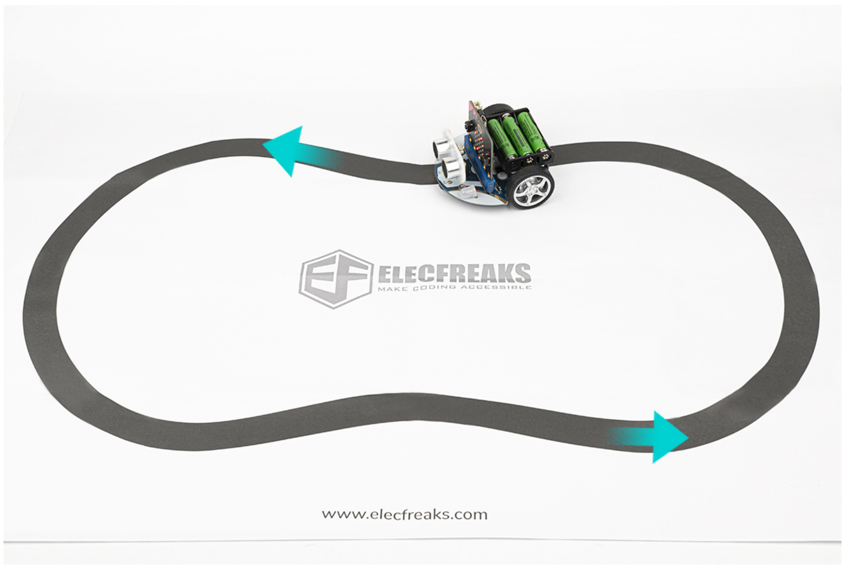
b) How can you trigger the RGB lights to turn on when light level is lower than 50?

____

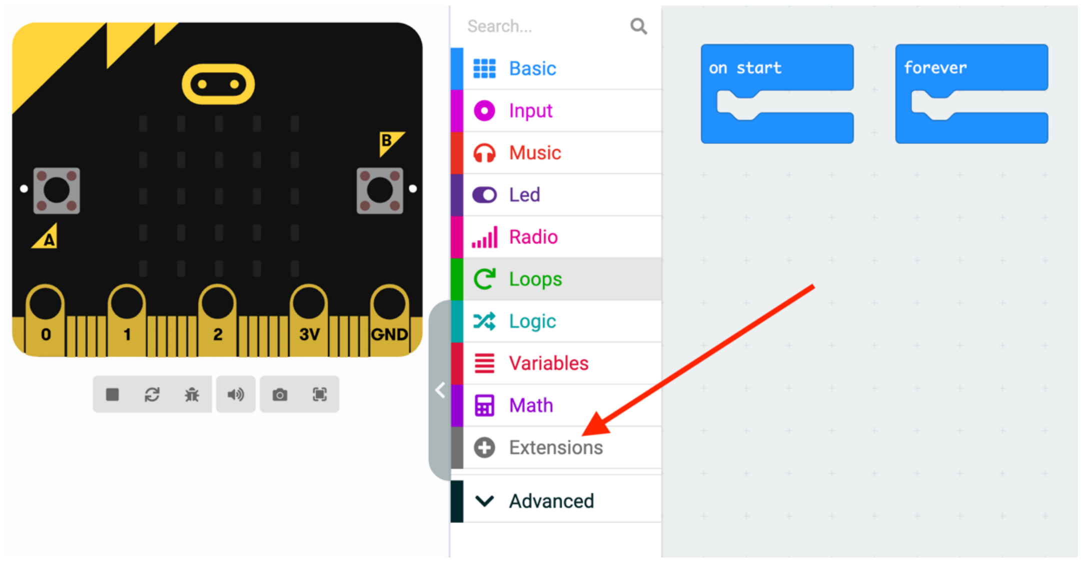
____

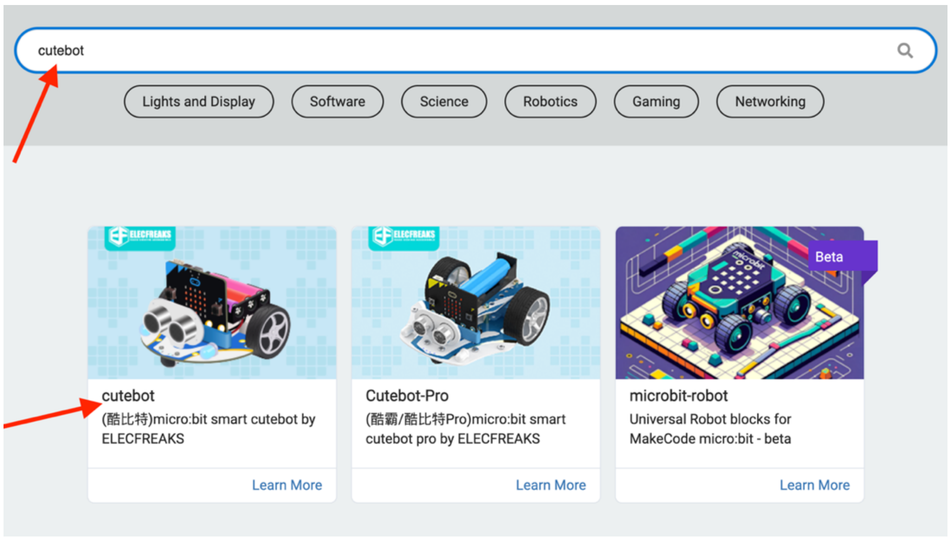
3.2. Run Along the Black Line

Objective: Make your Cutebot move along the black line only with programmed code, using the line-tracking probe.

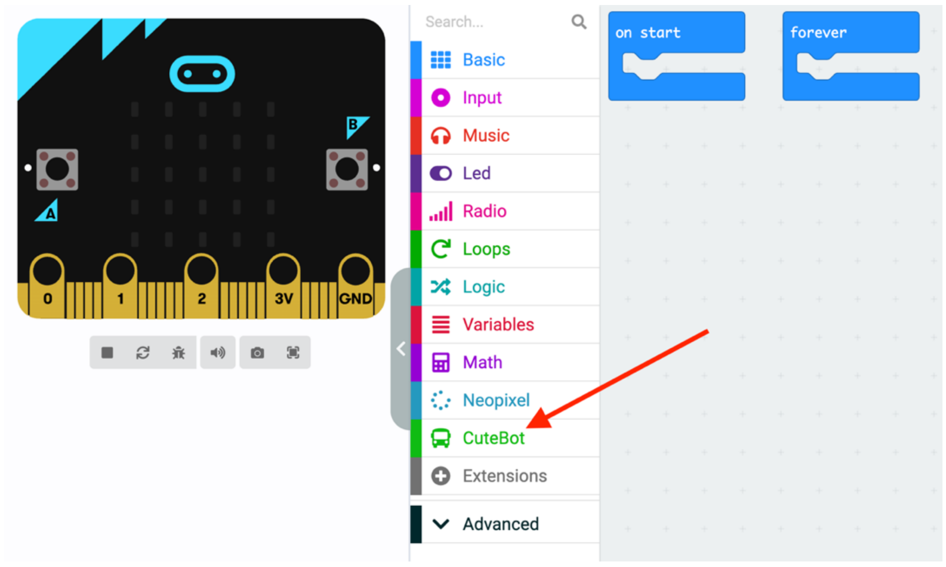


b. Visit MakeCode and create a new project. Click on Extensions, search for cutebot and select the first extension

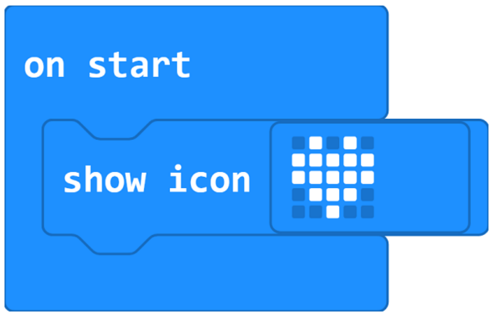



cutebot
Lights and Display
Software
Science
Robotics
Gaming
Networking
cutebot
(酷比特)micro:bit smart cutebot by ELECFREAKS
Learn More
Cutebot-Pro
(酷霸/酷比特Pro)micro:bit smart cutebot pro by ELECFREAKS
Learn More
microbit-robot
Universal Robot blocks for MakeCode micro:bit - beta
Beta
Learn More

c. A new block category is added



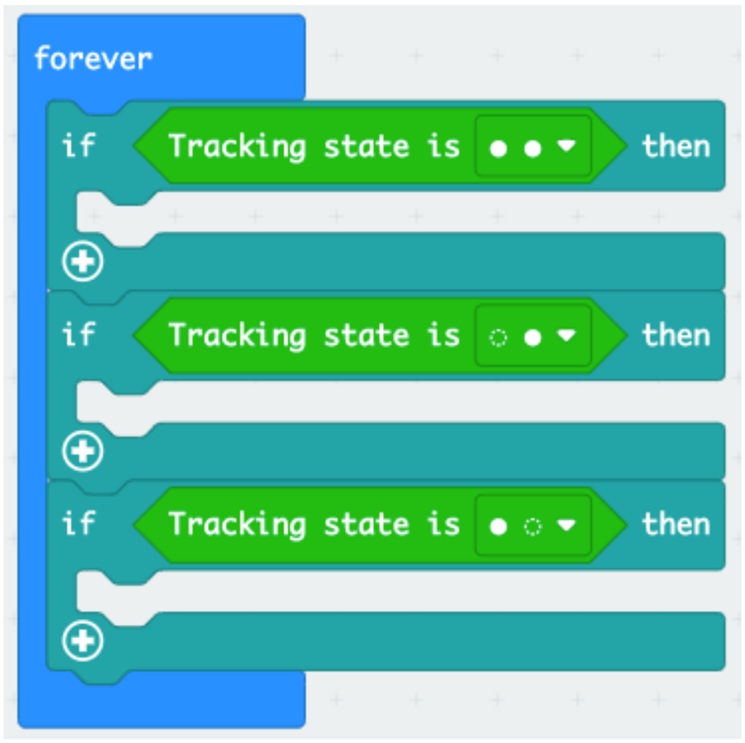
d. Drag [show icon] in to the on-start block



e. Drag three [if then] block in to the forever block

f. Drag the [tracking state is] block and replace the [true] block in the [if then else] block

g. Repeat step two for the other two [if then] blocks, changing the drop down value of the [tracking state is] block to option two and three respectively



```
forever
    if Tracking state is ● ● then
    if Tracking state is ◌ ● then
    if Tracking state is ● ◌ then
```

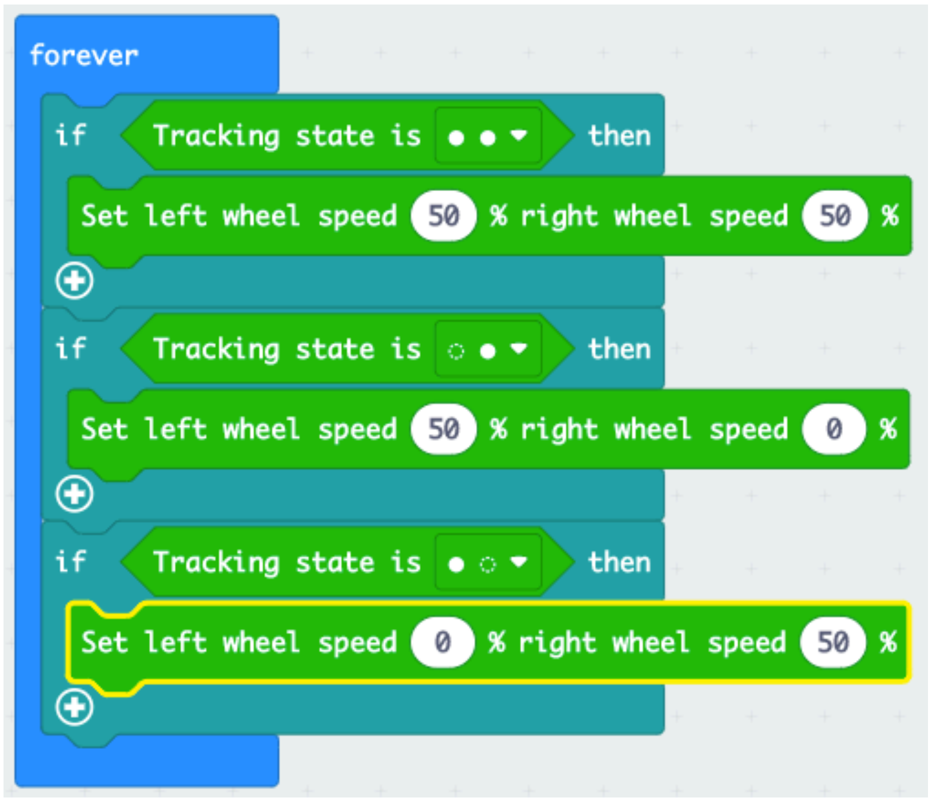
h. Drag the [set left wheel speed] block in to the [if then] block

i. Repeat step one for the other two [if then] blocks

j. Change the value in the first [set left wheel speed] to 50% and 50% respectively

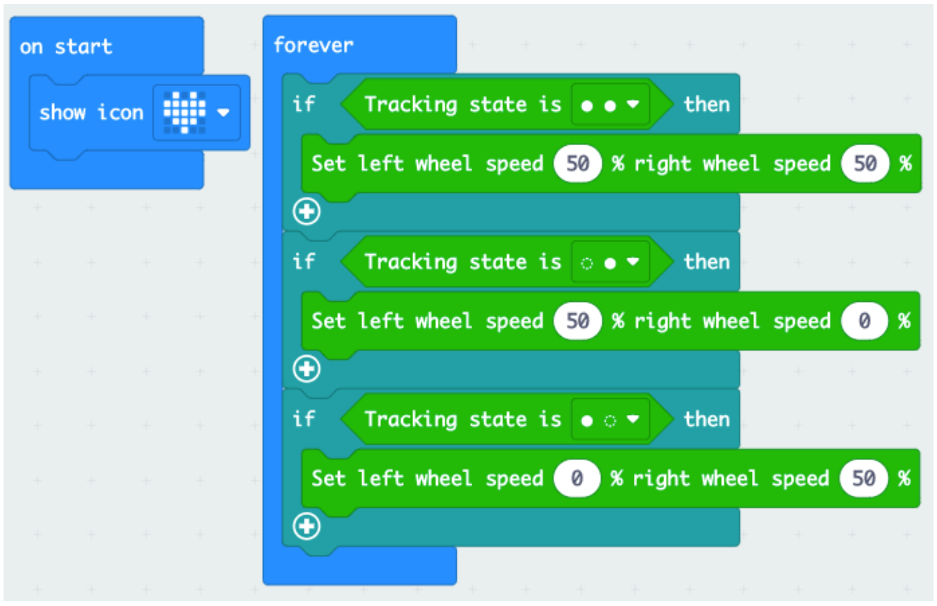
k. Change the value in the second [set left wheel speed] to 50% and 0% respectively

l. Change the value in the third [set left wheel speed] to 0% and 50% respectively



```
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 %
```

m. **Final result.** Download the code into your micro:bit and observe the result on a line-tracing map.



```
on start
  show icon

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 %
```

Troubleshooting notes:

a. Always **<u>remove</u>** the micro:bit from the cutebot when you are downloading code into it.

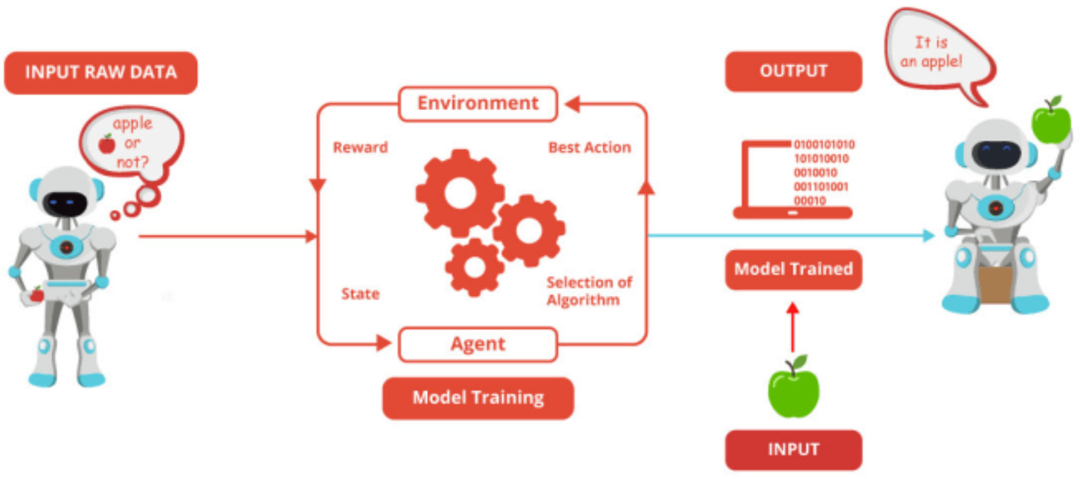
b. If the cutebot is unresponsive or AI lens is taking a long time to bootup after the micro:bit is slot in, it might be an error in the code. Please switch off cutebot, remove micro:bit and redownload the code and try again.

c. The wire of the AI lens may cause the cutebot to go off its designed path due to balancing. Please secure the wire appropriately

How is the movement state when the two motors are rotating at different speeds and directions?

Left Motor		Right Motor		Movement
Speed	Direction	Speed	Direction	
200	Forward	200	Forward	Move forward
200	Forward	50	Forward	
50	Forward	200	Forward	
200	Backward	200	Backward	
200	Backward	200	Forward	

4. Cutebot & AI Lens



Artificial Intelligence in Robotics

Robots are machines that can do tasks, sometimes like humans, sometimes like tools that help humans. But what makes a robot really smart is Artificial Intelligence (AI).

What is AI in simple terms?

AI is like the “brain” of a robot. It allows the robot to learn, think, and make decisions instead of just following pre-set instructions.

- Without AI: A robot just follows a list of steps.
- With AI: A robot can look at its surroundings, recognize objects, and decide what to do next.

Example: A robot car with AI can see a red traffic light and stop automatically, just like a human driver would.

How AI helps robots?

AI can make robots:

- See and understand the world – using cameras and sensors (like the AI lens in your project).
- Learn from experience – improve how they do tasks over time.
- Make decisions – choose what action to take depending on what they see or sense.

Example: In factories, AI robots can sort objects by color or shape automatically. In homes, AI robots can vacuum a room by avoiding furniture.

AI in robotics today

Some cool examples of AI-powered robots:

- Self-driving cars – they detect pedestrians, traffic lights, and obstacles.
- Delivery robots – can carry packages around safely in cities.
- Humanoid robots – like those that can talk, answer questions, or play games.

Why it matters

AI in robotics helps make work safer, faster, and more efficient. It also inspires young inventors (like you!) to design solutions for real-world problems.

In your upcoming activities, AI allows your cutebot to see colours, track objects, and make decisions automatically, just like a mini self-driving car.

AI doesn't replace humans—it helps humans do things smarter and safer and gives you a chance to invent exciting solutions!

Try out: Think!

a) How do you think the AI lens on the Cutebot is similar to technologies used in real-world autonomous vehicles, like self-driving cars or delivery robots?

____

____

b) Can you think of a situation where a robot with an AI lens could help people or solve a problem in daily life? How would it work?

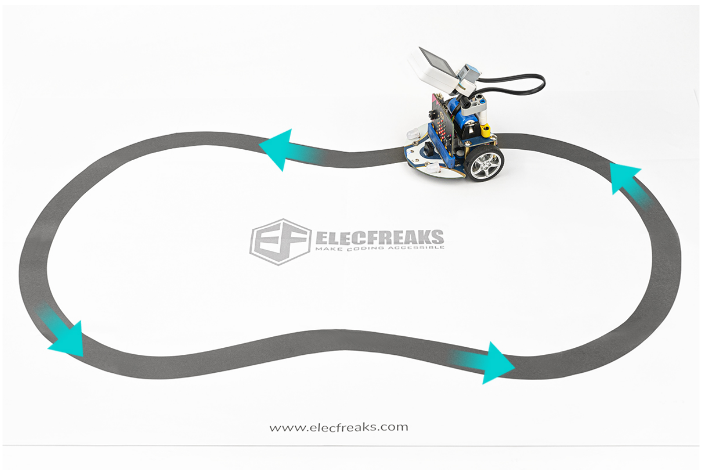
____

____

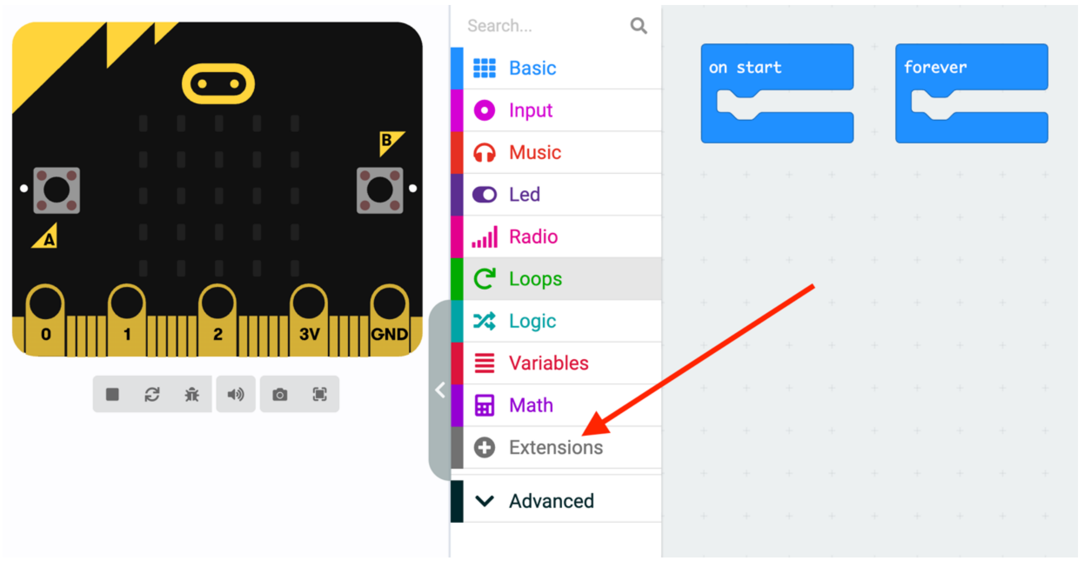
4.1. Line-Tracking with AI Lens

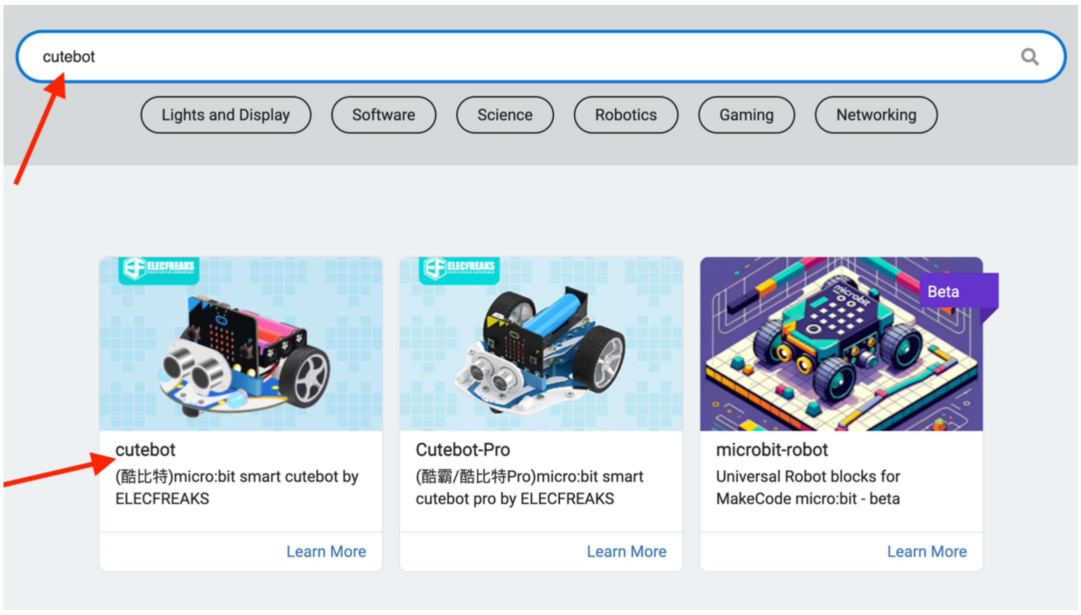
Objective: In your previous activity, you successfully made your Cutebot move along the black line using the line-tracking probe.

In this activity, we will make the cutebot achieve the line-tracking function using the AI lens. When the AI lens detects and recognizes the black line, it will go around it and determine the next movement.

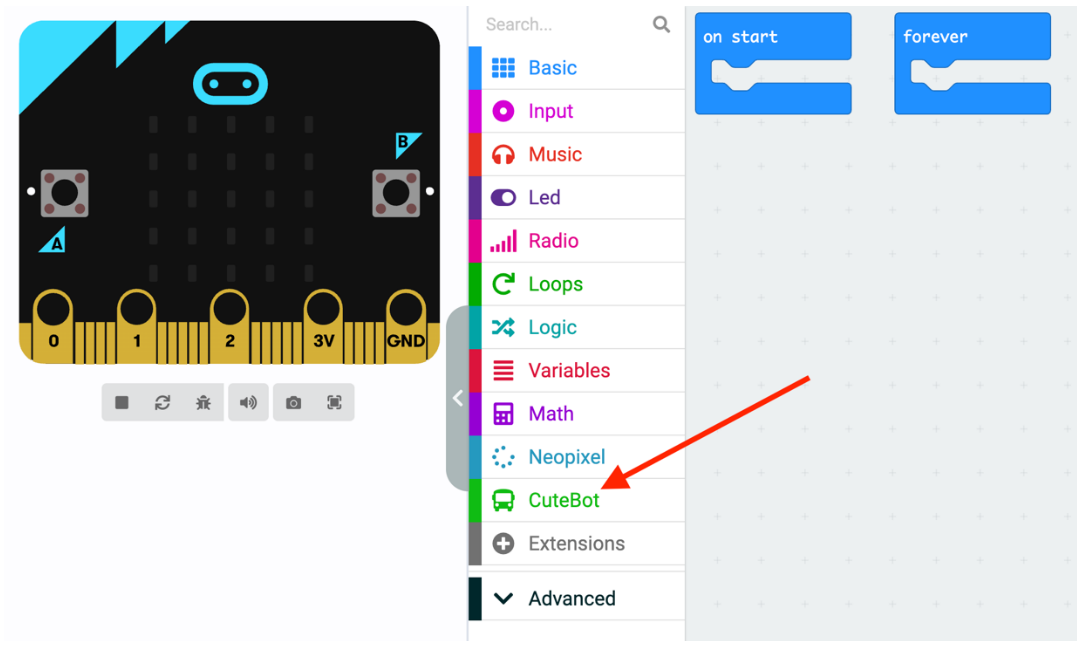


a. Visit MakeCode and create a new project. Click on Extensions, search for cutebot and select the first extension

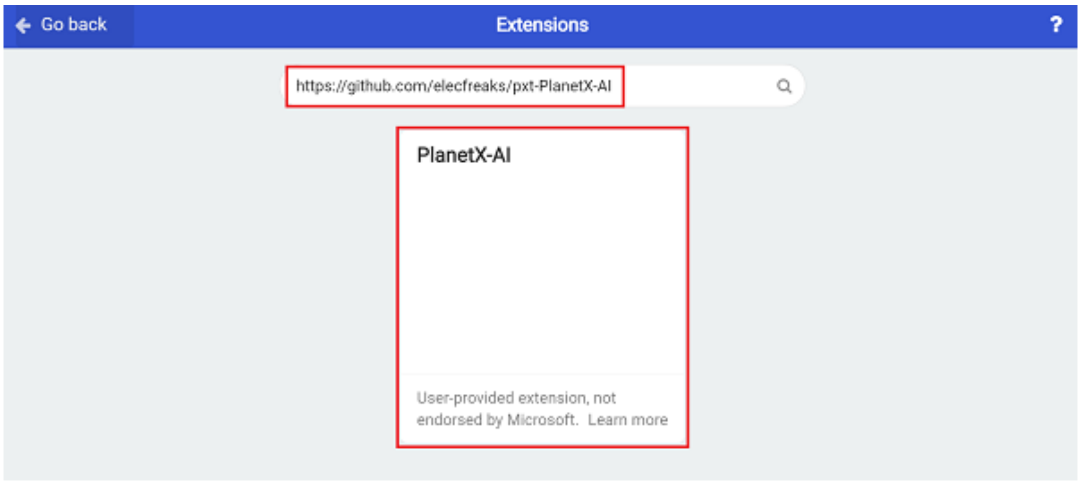



cutebot
Lights and Display
Software
Science
Robotics
Gaming
Networking
ELECFREAKS
ELECFREAKS
Beta
cutebot
(酷比特)micro:bit smart cutebot by ELECFREAKS
Learn More
Cutebot-Pro
(酷霸/酷比特Pro)micro:bit smart cutebot pro by ELECFREAKS
Learn More
microbit-robot
Universal Robot blocks for MakeCode micro:bit - beta
Learn More

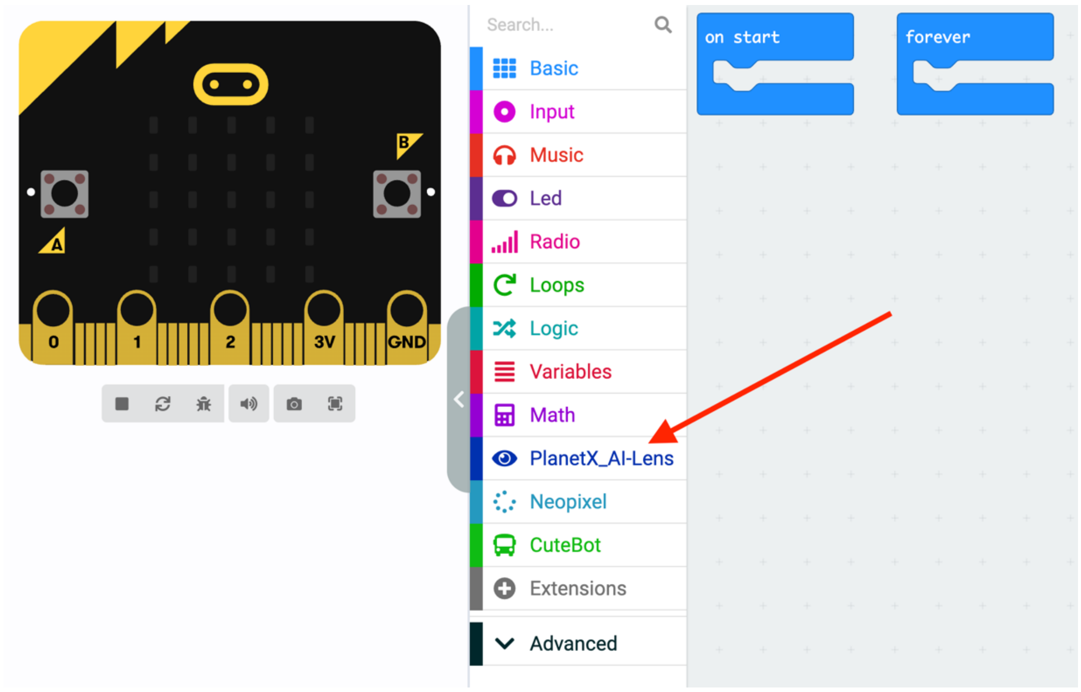
b. A new block category is added



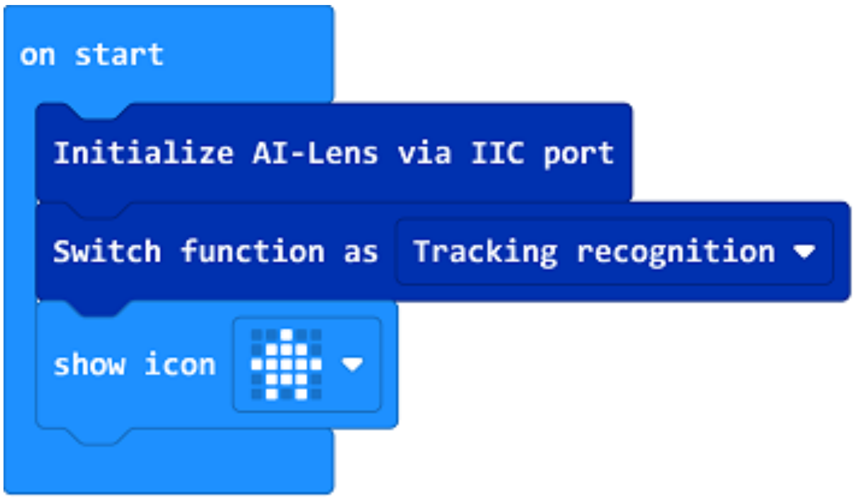
c. Add a 2nd extension. Search for 'PlanetX-AI' in the dialogue box


Go back
Extensions
?
https://github.com/elecfreaks/pxt-PlanetX-AI
PlanetX-AI
User-provided extension, not endorsed by Microsoft. Learn more

d. A new block category is added

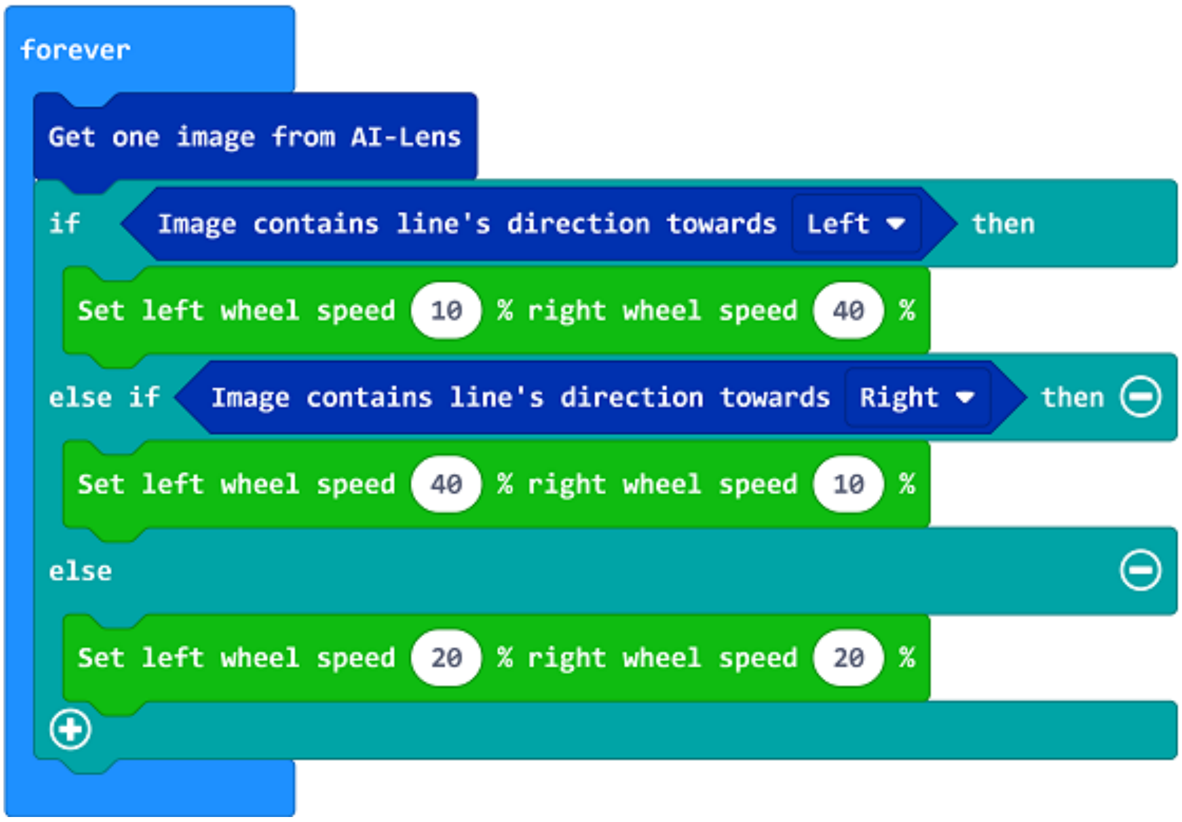


e. In the “on start” block, initialize the AI Lens and switch the function to the line tracking mode, set the micro:bit to display the appointed icon.



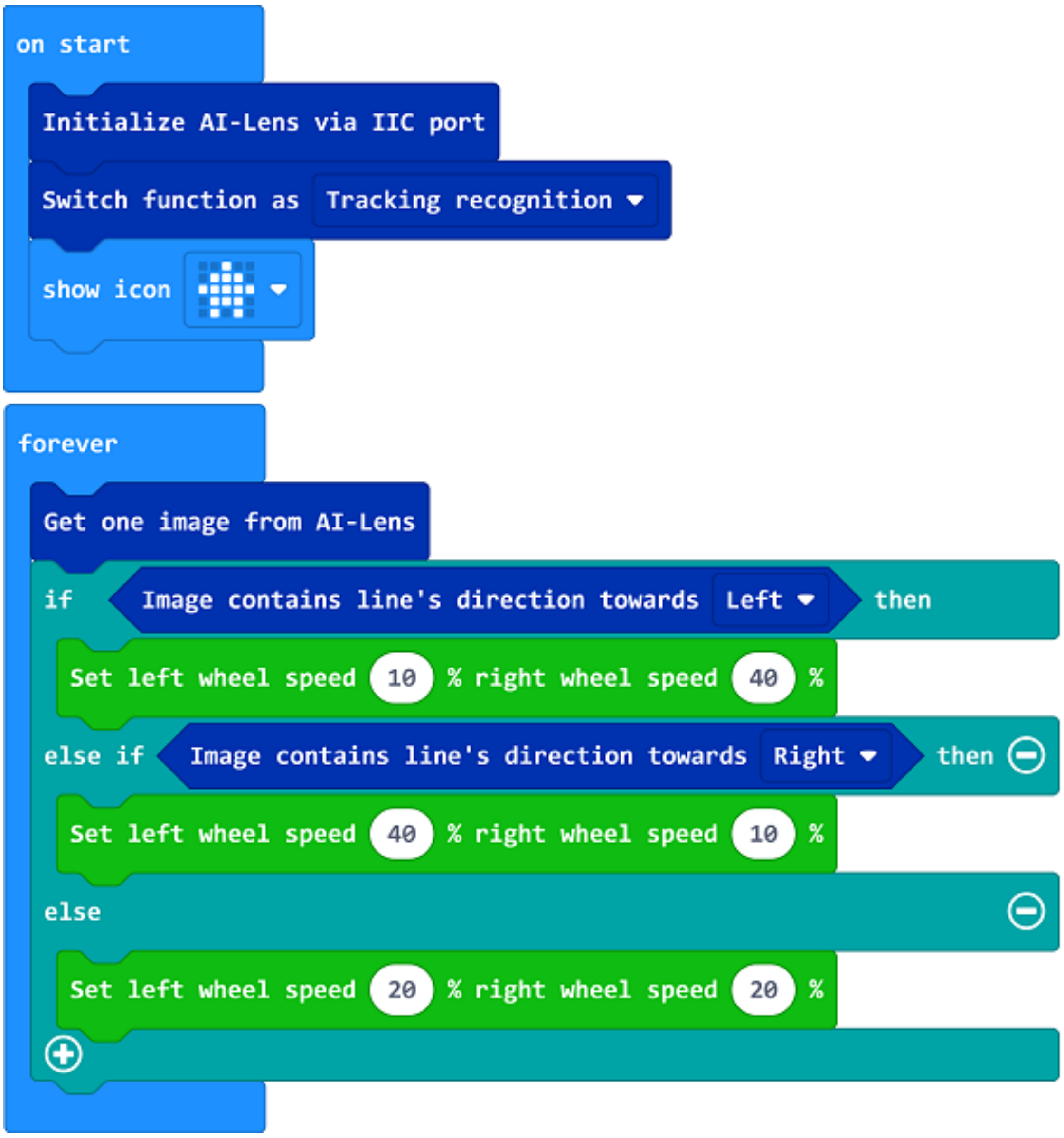
```
on start
    Initialize AI-Lens via IIC port
    Switch function as [Tracking recognition ▾]
    show icon [▾]
```

f. Attempt the following:



```
forever
    Get one image from AI-Lens
    if <Image contains line's direction towards [Left ▾]> then
        Set left wheel speed (10) % right wheel speed (40) %
    else if <Image contains line's direction towards [Right ▾]> then
        Set left wheel speed (40) % right wheel speed (10) %
    else
        Set left wheel speed (20) % right wheel speed (20) %
```

g. **Final result.** Download the code into your micro:bit and observe the result.



```
on start
    Initialize AI-Lens via IIC port
    Switch function as  Tracking recognition ▾
    show icon  ▾
forever
    Get one image from AI-Lens
    if  Image contains line's direction towards  Left ▾  then
        Set left wheel speed 10 % right wheel speed 40 %
    else if  Image contains line's direction towards  Right ▾  then
        Set left wheel speed 40 % right wheel speed 10 %
    else
        Set left wheel speed 20 % right wheel speed 20 %
```

Understanding the code:

In the "forever" block, set to get one image form the AI Lens and judge the deviation direction of the line on the image. If it deviates to the left side, it means the car deviates to the right, we should set the speed of *the left wheel at the speed of 10% and the right at 40% to make the car turn left and go to the right way.*

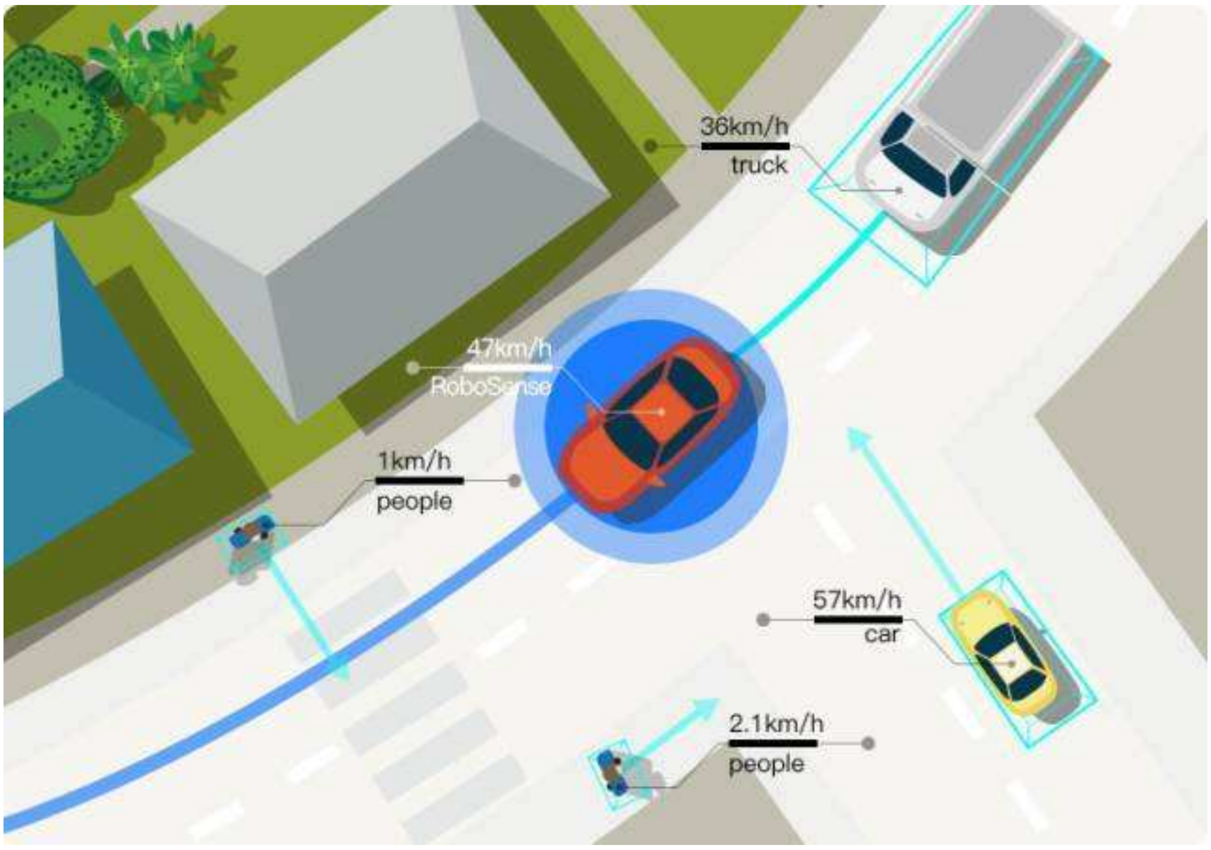
While if the line deviates to the right side, it means the car deviates to the left side, now we should set the speed of the right wheel at the speed of 10% and the left at 40% to make the car turn right and go to the right way; or we may set the speed of both wheels at 20% and the car moves forward with the line.

Troubleshooting notes:

a. Always **remove** the micro:bit from the cutebot when you are downloading code into it.
b. If the cutebot is unresponsive or AI lens is taking a long time to bootup after the micro:bit is slot in, it might be an error in the code. Please switch off cutebot, remove micro:bit and redownload the code and try again.
c. The wire of the AI lens may cause the cutebot to go off its designed path due to balancing. Please secure the wire appropriately

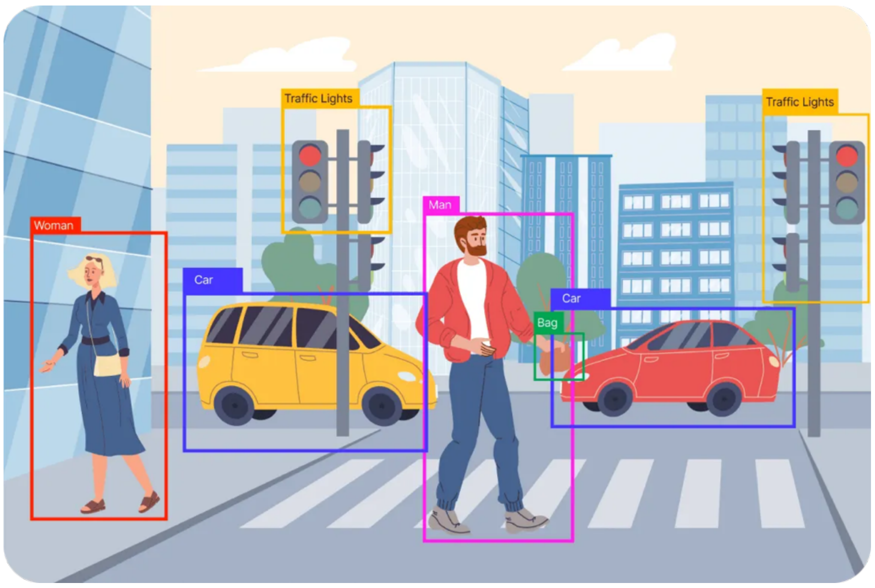
4.2. Object Recognition & Object Tracking

As one of the vital functions of AI visual recognition, object tracking belongs to one type of object recognition. Object tracking is of vital importance in computer vision, referring to the process of making continuous inferences about target status in a video sequence, which can be simply regarded as recognizing and tracking the objects moving within the visual range of the camera.

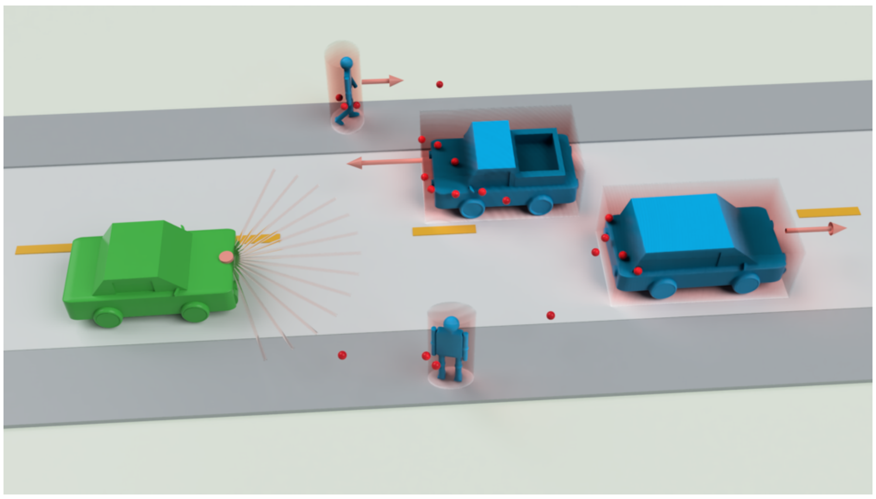


Object recognition and object tracking

Object recognition is to obtain accurate appearance information of the object through some image processing algorithms under a static background, then recognize and mark the shape of the object, as shown in the figure.

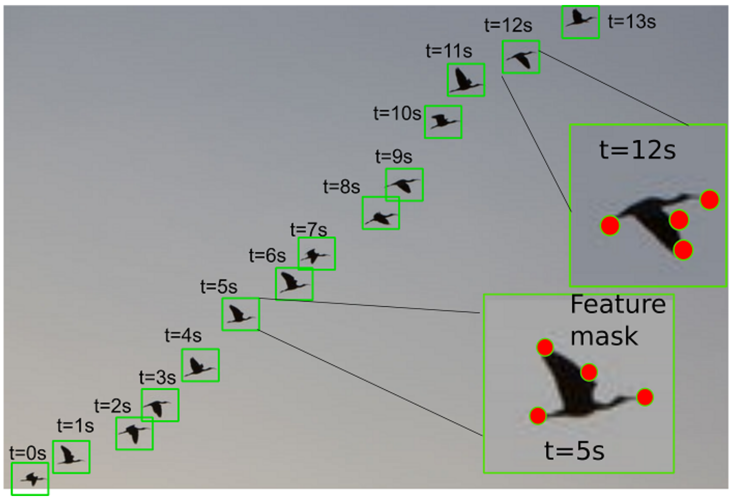


Object tracking refers to tracking the subsequent image sequence through algorithms according to the appearance characteristics of the object obtained from the previous step and carry out more in-depth learning in the subsequent tracking so as to make the tracking more and more accurate.



Object motion prediction

Motion prediction is using an algorithm to predict the image of a moving object in the next frame so that it can optimize the algorithm and improve efficiency. As the picture shows, the following movement path and action can be predicted by the bird's movement trend in the first few seconds.



How do you think object tracking is applied in autonomous transportation?

____

____

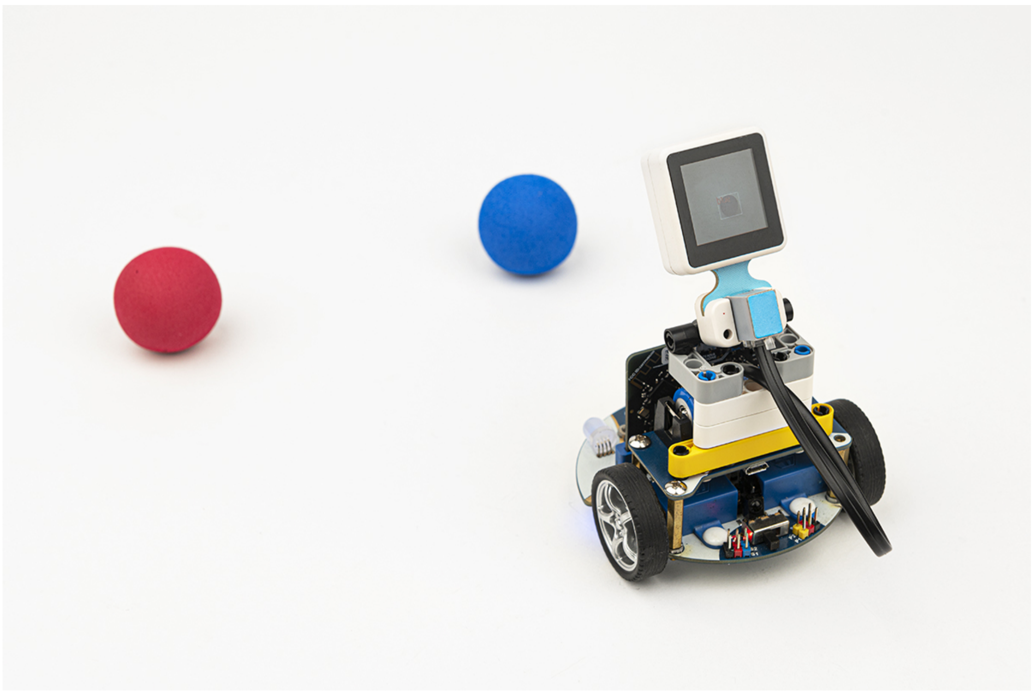
Application of object tracking

Based on motion recognition (human recognition basing on footwork, automatic object detecting), automatic monitoring (monitor the suspicious acts), traffic monitoring (collecting the real-time traffic data to direct the traffic).

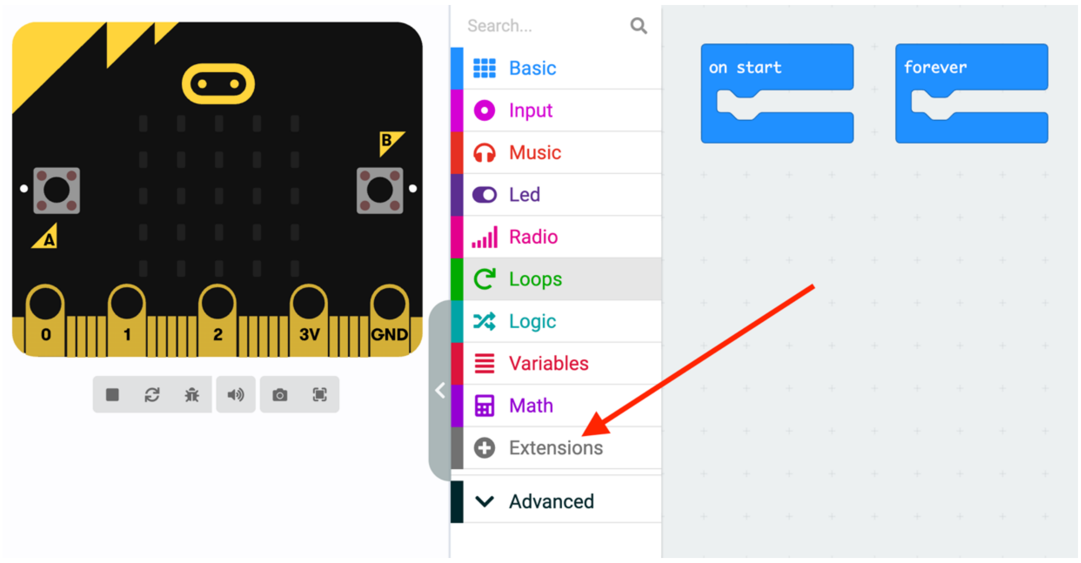
The traditional human-computer interaction is carried out by the keyboard and mouse of the computer. Tracking technology is the key point when a computer needs to be able to recognize and understand the posture, movement, and gesture.

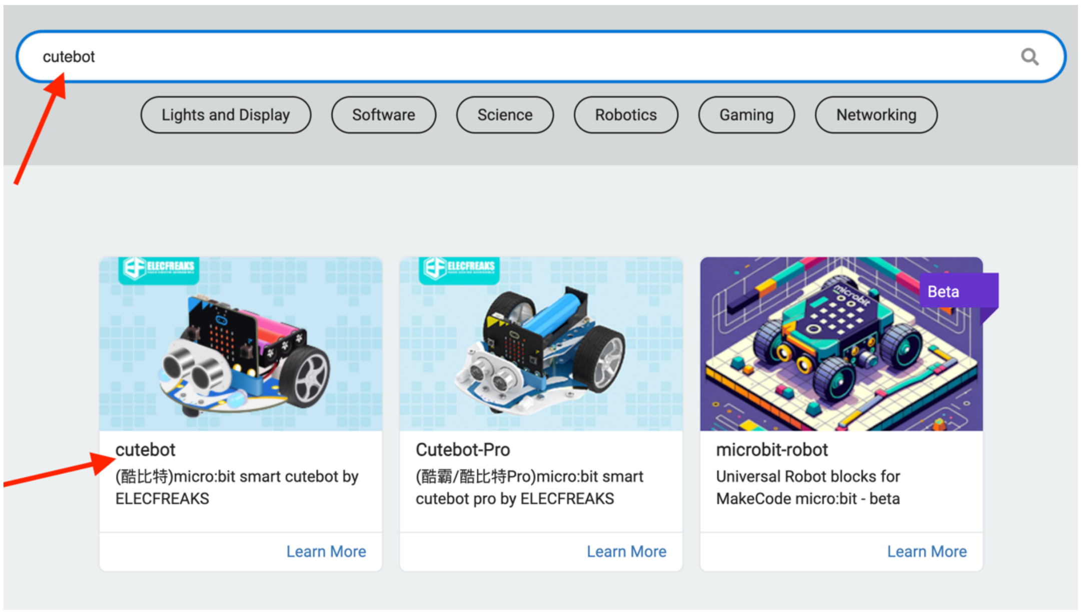
4.3. Cutebot – Object Tracking (Ball)

Objective: In this activity, we will make the cutebot achieve in object tracking on a ball using the AI lens.

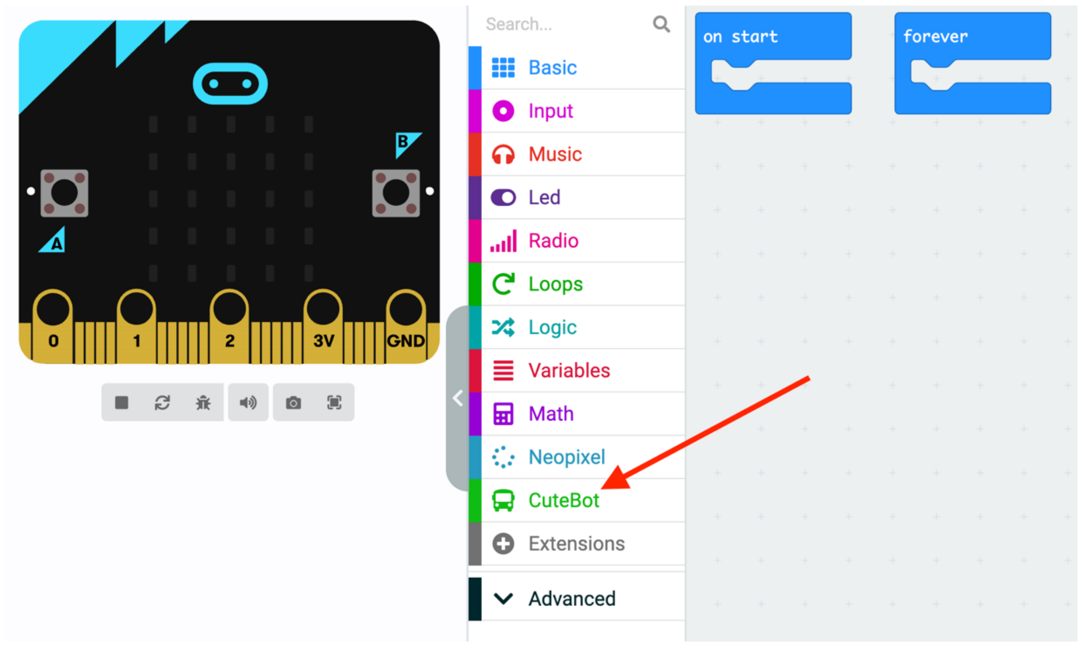


a. Visit MakeCode and create a new project. Click on Extensions, search for cutebot and select the first extension

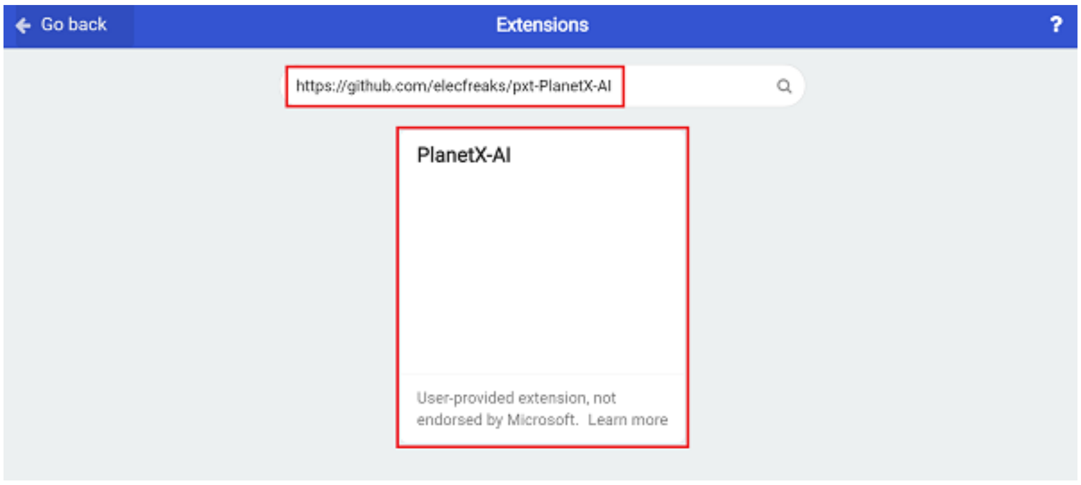



cutebot
Lights and Display
Software
Science
Robotics
Gaming
Networking
cutebot
(酷比特)micro:bit smart cutebot by ELECFREAKS
Learn More
Cutebot-Pro
(酷霸/酷比特Pro)micro:bit smart cutebot pro by ELECFREAKS
Learn More
Beta
microbit-robot
Universal Robot blocks for MakeCode micro:bit - beta
Learn More

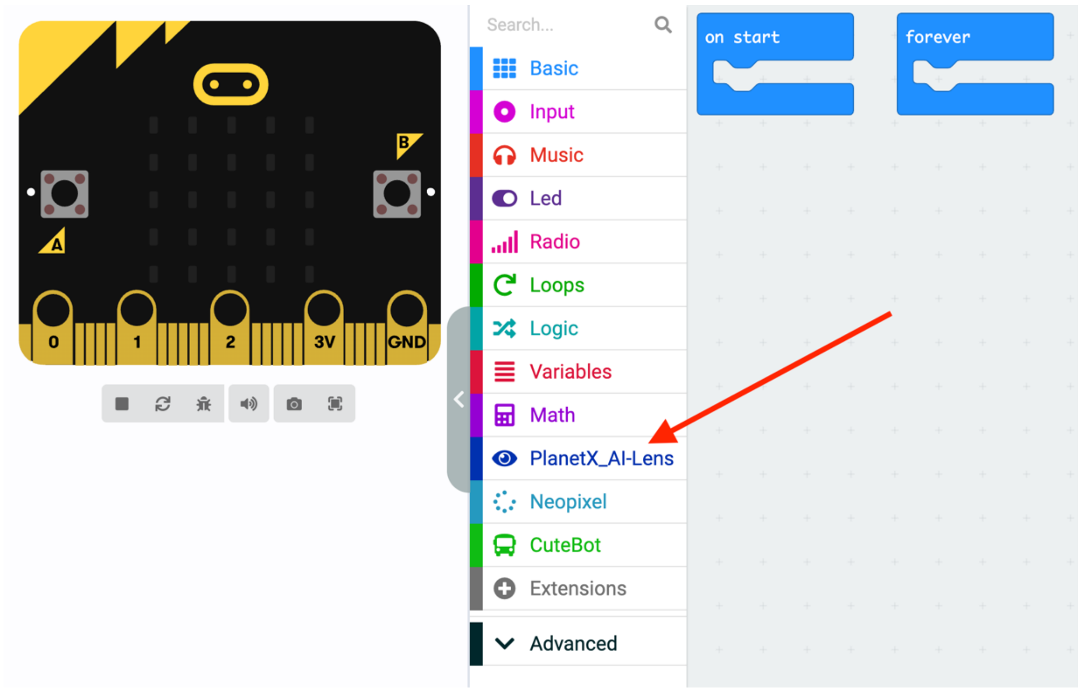
b. A new block category is added



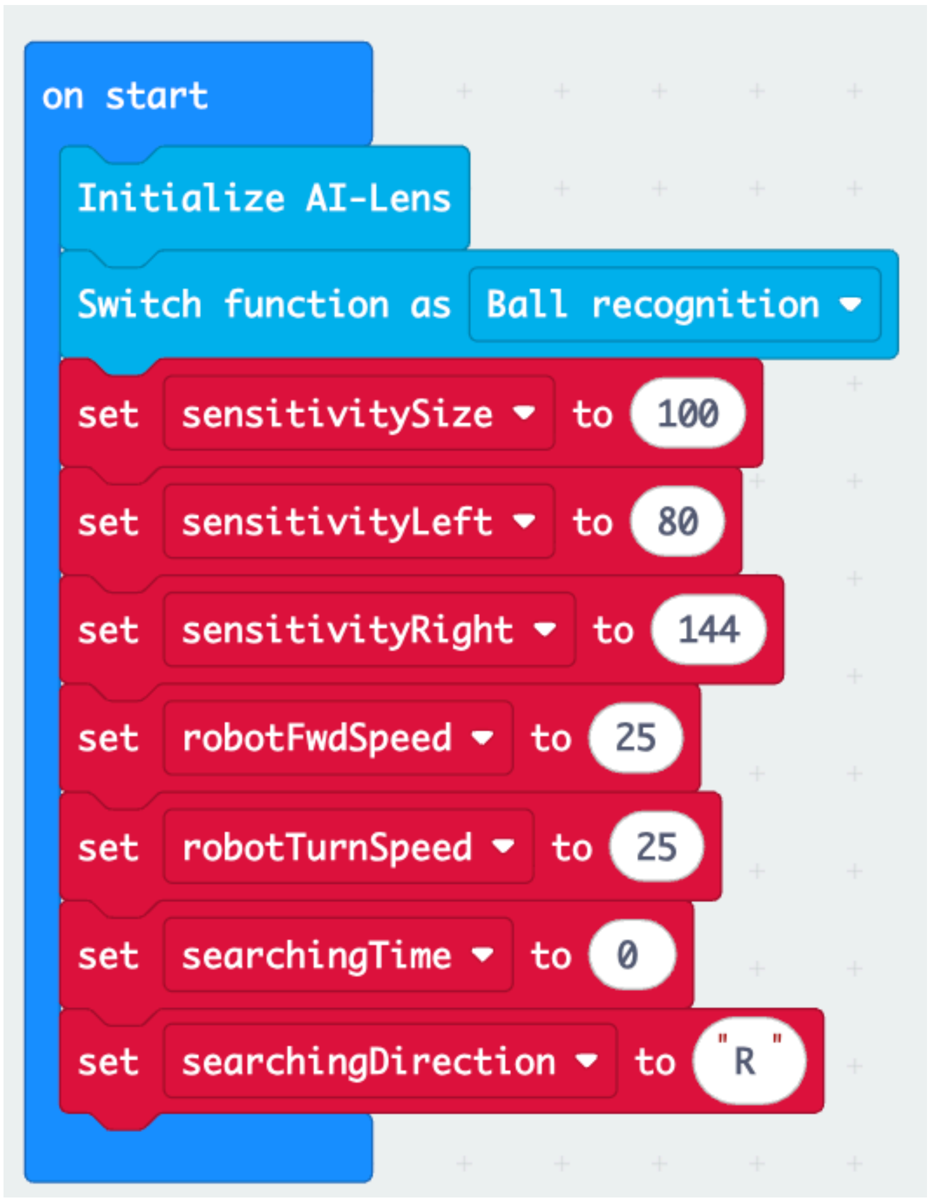
c. Add a 2nd extension. Search for 'PlanetX-AI' in the dialogue box


Go back
Extensions
?
https://github.com/elecfreaks/pxt-PlanetX-AI
PlanetX-AI
User-provided extension, not endorsed by Microsoft. Learn more

d. A new block category is added

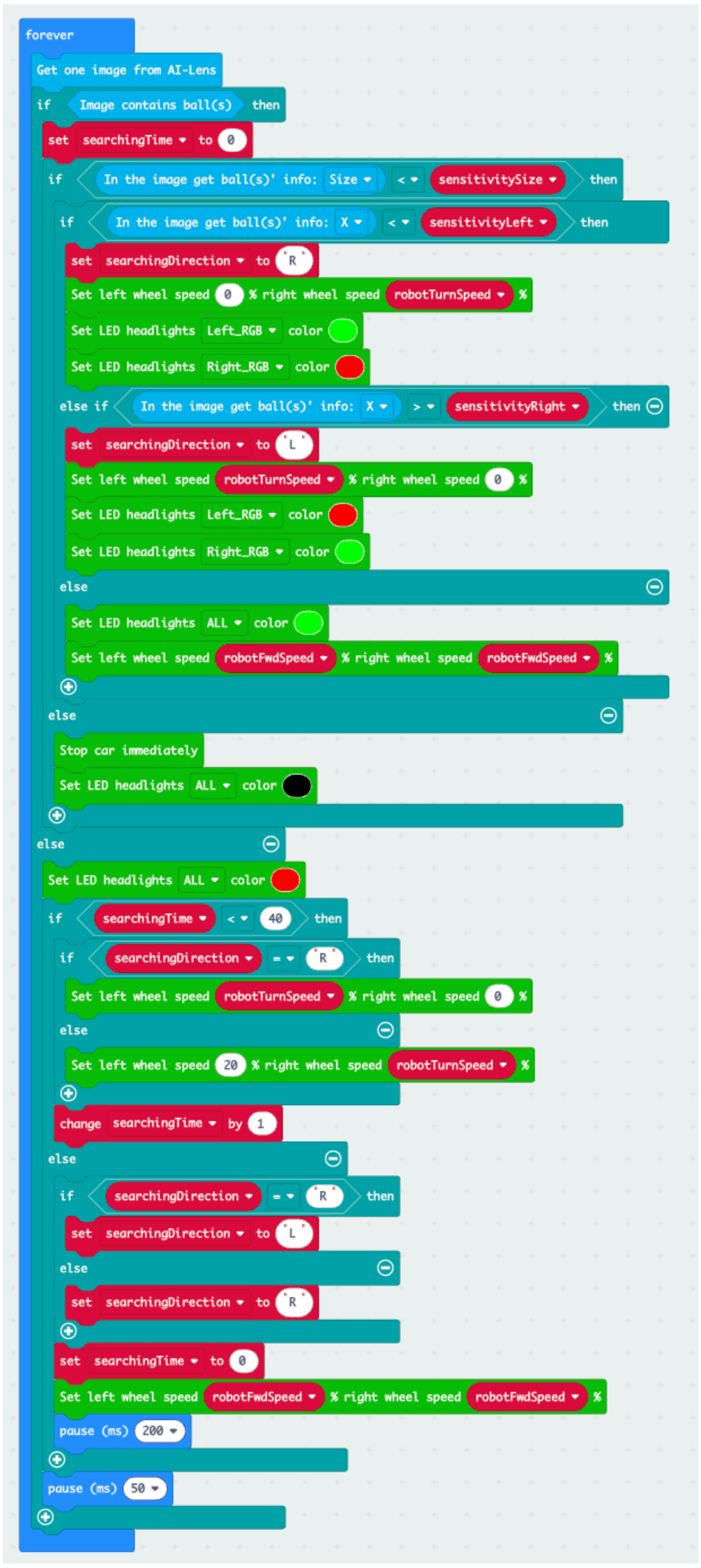


e. In the “on start” block, initialize the AI Lens and switch the function to the ball recognition mode. Set and declare the various variables accordingly.



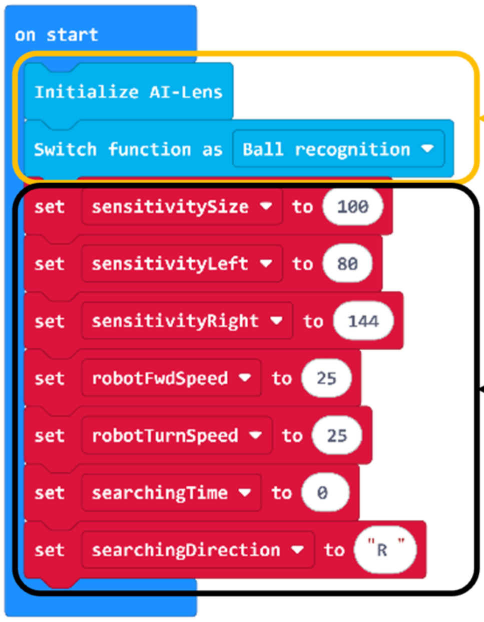
```
on start
    Initialize AI-Lens
    Switch function as Ball recognition ▾
    set sensitivitySize ▾ to 100
    set sensitivityLeft ▾ to 80
    set sensitivityRight ▾ to 144
    set robotFwdSpeed ▾ to 25
    set robotTurnSpeed ▾ to 25
    set searchingTime ▾ to 0
    set searchingDirection ▾ to " R "
```

f. Attempt the following:



```
forever
  Get one image from AI-Lens
  if Image contains ball(s) then
    set searchingTime ▾ to 0
    if In the image get ball(s)' info: Size ▾ < ▾ sensitivitySize ▾ then
      if In the image get ball(s)' info: X ▾ < ▾ sensitivityLeft ▾ then
        set searchingDirection ▾ to "R"
        Set left wheel speed 0 % right wheel speed robotTurnSpeed ▾ %
        Set LED headlights Left_RGB ▾ color 🟢
        Set LED headlights Right_RGB ▾ color 🔴
      else if In the image get ball(s)' info: X ▾ > ▾ sensitivityRight ▾ then ⊖
        set searchingDirection ▾ to "L"
        Set left wheel speed robotTurnSpeed ▾ % right wheel speed 0 %
        Set LED headlights Left_RGB ▾ color 🔴
        Set LED headlights Right_RGB ▾ color 🟢
      else ⊖
        Set LED headlights ALL ▾ color 🟢
        Set left wheel speed robotFwdSpeed ▾ % right wheel speed robotFwdSpeed ▾ %
      ⊕
    else ⊖
      Stop car immediately
      Set LED headlights ALL ▾ color ⚫
    ⊕
  else ⊖
    Set LED headlights ALL ▾ color 🔴
    if searchingTime ▾ < ▾ 40 then
      if searchingDirection ▾ = ▾ "R" then
        Set left wheel speed robotTurnSpeed ▾ % right wheel speed 0 %
      else ⊖
        Set left wheel speed 20 % right wheel speed robotTurnSpeed ▾ %
      ⊕
      change searchingTime ▾ by 1
    else ⊖
      if searchingDirection ▾ = ▾ "R" then
        set searchingDirection ▾ to "L"
      else ⊖
        set searchingDirection ▾ to "R"
      ⊕
      set searchingTime ▾ to 0
      Set left wheel speed robotFwdSpeed ▾ % right wheel speed robotFwdSpeed ▾ %
      pause (ms) 200 ▾
    ⊕
    pause (ms) 50 ▾
  ⊕
```

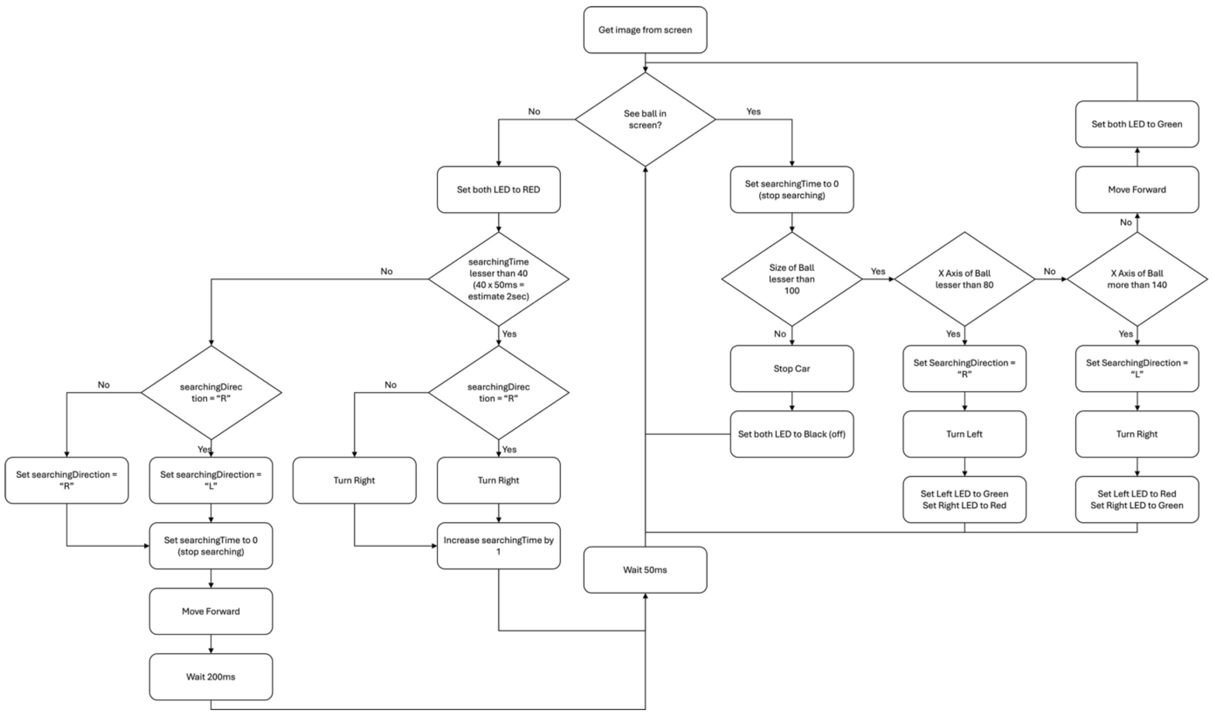
Understanding the code:

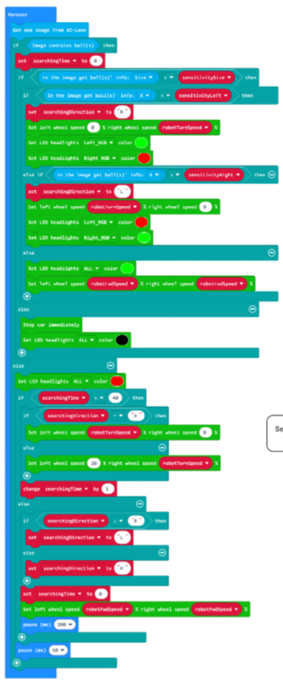


```
on start
  Initialize AI-Lens
  Switch function as [Ball recognition ▾]
  set [sensitivitySize ▾] to (100)
  set [sensitivityLeft ▾] to (80)
  set [sensitivityRight ▾] to (144)
  set [robotFwdSpeed ▾] to (25)
  set [robotTurnSpeed ▾] to (25)
  set [searchingTime ▾] to (0)
  set [searchingDirection ▾] to ("R")
```

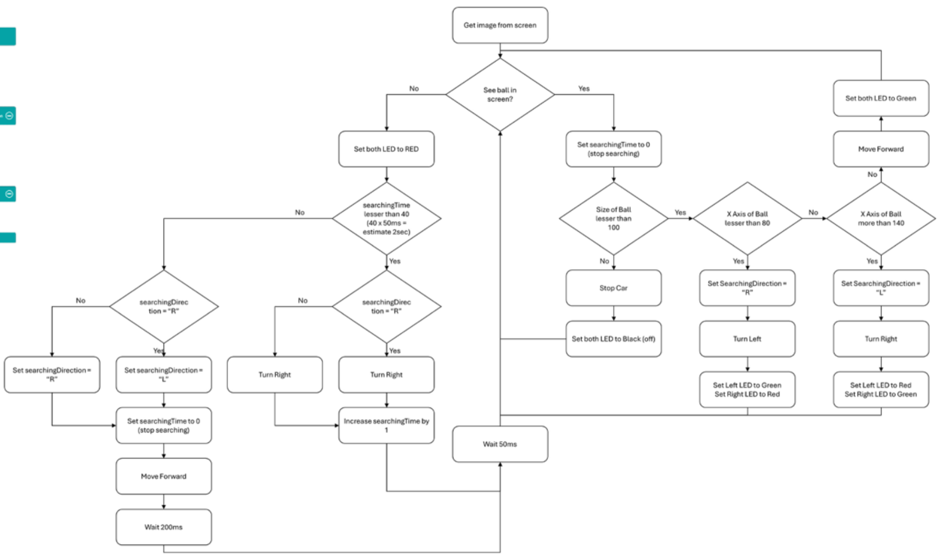
Initialize AI Lens and set to ball recognition

1. sensitivitySize – sets the sensitivity size level
2. sensitivityLeft – sets the sensitity left x asis level
3. sensitivityRight – sets the sensitity right x asis level
4. robotFwdSpeed – sets the robot move forward speed
5. robotTurnSpeed – sets the robot turning speed
6. searchingTime – calculate how long the rotation ball search has gone on before moving forward
7. searchingDirection – which direction to turn when searching; will change after every iteration (to ensure searching is done in both left and right turn direction)


Get image from screen
See ball in screen?
No
Yes
Set both LED to RED
searchingTime lesser than 40 (40 x 50ms = estimate 2sec)
No
Yes
searchingDirection = "R"
No
Yes
Set searchingDirection = "R"
Set searchingDirection = "L"
Set searchingTime to 0 (stop searching)
Move Forward
Wait 200ms
searchingDirection = "R"
No
Yes
Turn Right
Turn Right
Increase searchingTime by 1
Wait 50ms
Set searchingTime to 0 (stop searching)
Size of Ball lesser than 100
Yes
No
Stop Car
Set both LED to Black (off)
X Axis of Ball lesser than 80
Yes
No
Set SearchingDirection = "R"
Turn Left
Set Left LED to Green Set Right LED to Red
X Axis of Ball more than 140
Yes
No
Set SearchingDirection = "L"
Turn Right
Set Left LED to Red Set Right LED to Green
Move Forward
Set both LED to Green



```
forever
  Get one image from AI-Lens
  if Image contains ball(s) then
    set searchingTime to 0
    if In the image get ball(s)' info: Size < sensitivitySize then
      if In the image get ball(s)' info: X < sensitivityLeft then
        set searchingDirection to "R"
        Set left wheel speed 0 % right wheel speed robotTurnSpeed %
        Set LED headlights Left_RGB color ●
        Set LED headlights Right_RGB color ●
      else if In the image get ball(s)' info: X > sensitivityRight then
        set searchingDirection to "L"
        Set left wheel speed robotTurnSpeed % right wheel speed 0 %
        Set LED headlights Left_RGB color ●
        Set LED headlights Right_RGB color ●
      else
        Set LED headlights ALL color ●
        Set left wheel speed robotFwdSpeed % right wheel speed robotFwdSpeed %
    else
      Stop car immediately
      Set LED headlights ALL color ●
  else
    Set LED headlights ALL color ●
    if searchingTime < 40 then
      if searchingDirection = "R" then
        Set left wheel speed robotTurnSpeed % right wheel speed 0 %
      else
        Set left wheel speed 20 % right wheel speed robotTurnSpeed %
      change searchingTime by 1
    else
      if searchingDirection = "R" then
        set searchingDirection to "L"
      else
        set searchingDirection to "R"
      set searchingTime to 0
      Set left wheel speed robotFwdSpeed % right wheel speed robotFwdSpeed %
      pause (ms) 200
  pause (ms) 50
```


Get image from screen
See ball in screen?
No
Yes
Set both LED to RED
searchingTime lesser than 40 (40 x 50ms = estimate 2sec)
No
Yes
searchingDirection = "R"
No
Yes
searchingDirection = "R"
No
Yes
Set searchingDirection = "R"
Set searchingDirection = "L"
Turn Right
Turn Right
Set searchingTime to 0 (stop searching)
Increase searchingTime by 1
Move Forward
Wait 200ms
Wait 50ms
Set searchingTime to 0 (stop searching)
Size of Ball lesser than 100
Yes
No
Stop Car
Set both LED to Black (off)
X Axis of Ball lesser than 80
Yes
No
Set SearchingDirection = "R"
Turn Left
Set Left LED to Green Set Right LED to Red
X Axis of Ball more than 140
Yes
No
Set SearchingDirection = "L"
Turn Right
Set Left LED to Red Set Right LED to Green
Move Forward
Set both LED to Green

View the full code and explanation on the support portal!

Troubleshooting notes:

a. Always **<u>remove</u>** the micro:bit from the cutebot when you are downloading code into it.

b. If the cutebot is unresponsive or AI lens is taking a long time to bootup after the micro:bit is slot in, it might be an error in the code. Please switch off cutebot, remove micro:bit and redownload the code and try again.

c. The wire of the AI lens may cause the cutebot to go off its designed path due to balancing. Please secure the wire appropriately.

5. AI Robotic Soccer Challenge 🏆

In this challenge, your Cutebot will compete in a robotic soccer game using AI ball recognition. The robot must detect the soccer ball using the AI lens, navigate across the field, and push the ball accurately into the goal post on the soccer field map. This challenge brings together everything you have learned about movement control, sensing, and decision-making in autonomous systems.

<u>Important Notes to Take Note Of:</u>

- The AI lens angle and position affect how the ball is detected. A ball that appears larger usually means it is closer, while a smaller ball means it is further away.
- You may need to refine and adjust your code to respond correctly to different ball sizes and distances.
- Lighting conditions and field positioning can also affect detection accuracy, so testing and iteration are important.
- There is no single "perfect" solution — successful teams will test, debug, and improve their programs through multiple trials.