BerryBot

pico bricks | IDE

MicroBlocks

MicroPython

ARDUINO

Application

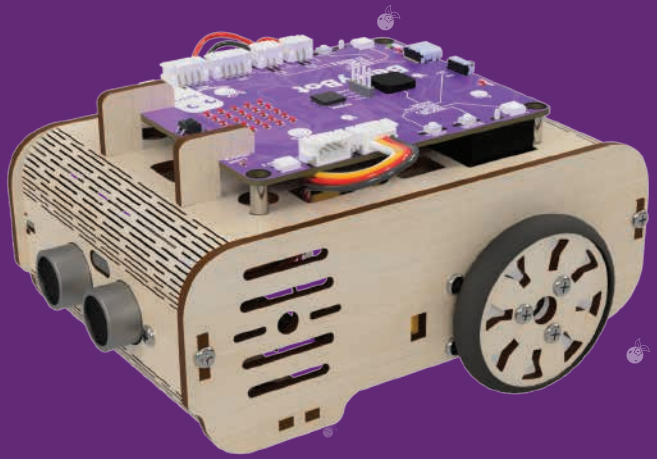
click

GO!

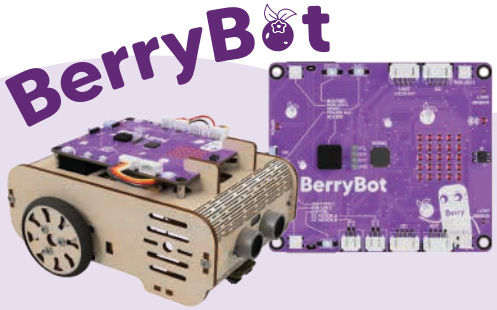
coding

light tracker mode

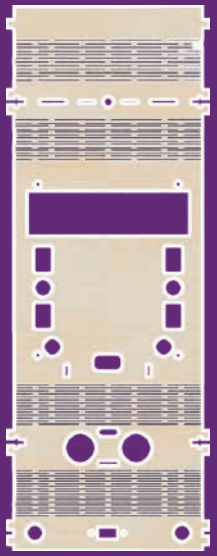
line tracker mode



CONTENTS



Wooden and Plexiglass Piece



1

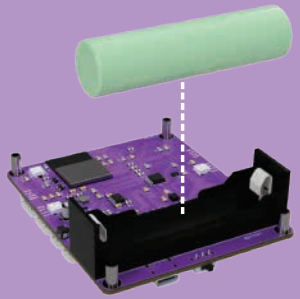
2

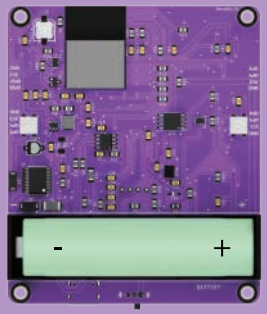
3 2pcs.

4 4pcs.

5 2pcs.
(Plexi piece)

6 2pcs.

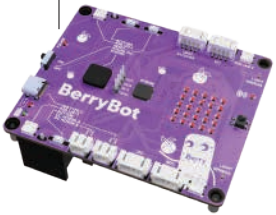




Pay attention to the battery orientation.

Components of BerryBot

HC-SR04
Distance Sensor



BerryBot
Main Board

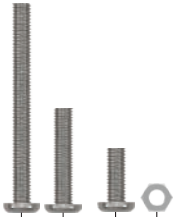
Line Tracer Sensor

DC Motor 2pcs.

DC Motor
Connection Bracket 2pcs.

Remote Controller

Mini Drunk Wheel
2pcs.



M3 Nut 16pcs.
M3 x 8mm Screw 10pcs.
M3 x 15mm Screw 8pcs.
M3 x 30mm Screw 4pcs.

Connecting Cables 4pcs.

Wheel Gasket 2pcs.

What is BerryBot?

BerryBot is a robotics programming kit aimed at developing children's skills in robotics infrastructure, basic programming, hand-eye coordination, problem-solving and more. Thanks to the main board included in the kit, motor connections that allow the robot to move physically can be made quickly. With its easy-to-assemble wooden body, the mechanical and electronic setup of the robot can be completed in a short time.

Once you have completed the setup of BerryBot, you can improve your programming skills by programming it using the PicoBricks IDE either in a block-based or text-based manner.

You can also control the robot from your phone or tablet via PicoBricks GO mobile application, making it a fun STEM toy.

With the tracks included in the kit, you can quickly carry out popular projects such as line following, sumo, light following, and obstacle-avoiding robots using the mobile application. You can also learn how to develop the algorithms for these projects and write the code using the PicoBricks IDE. Additionally, with the other track included in the kit, you can carry out the tasks in the booklet or the tasks you create based on your imagination, organizing fun activities with your friends.

Programming Environments

BerryBot can be programmed online with the PicoBricks IDE using block-based and text-based coding. It can also be programmed offline using text-based coding with Thonny IDE and Arduino IDE.

RGB LEDs

Horn

Obstacle Avoidance Mode

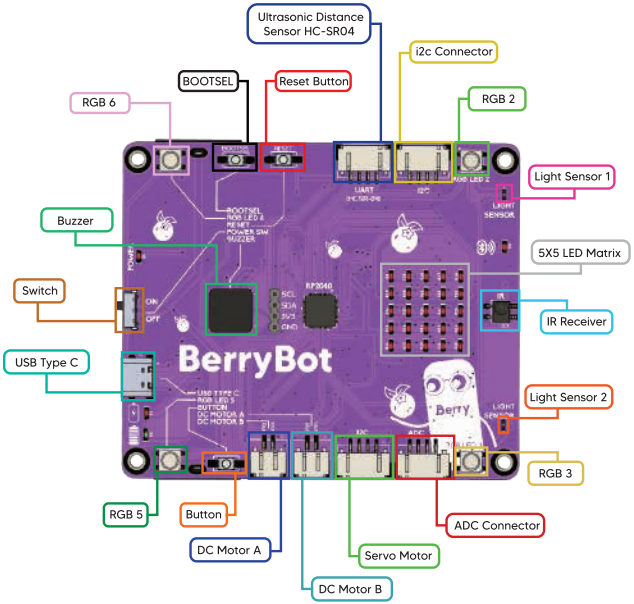
Line Tracker Mode

Light Tracker Mode

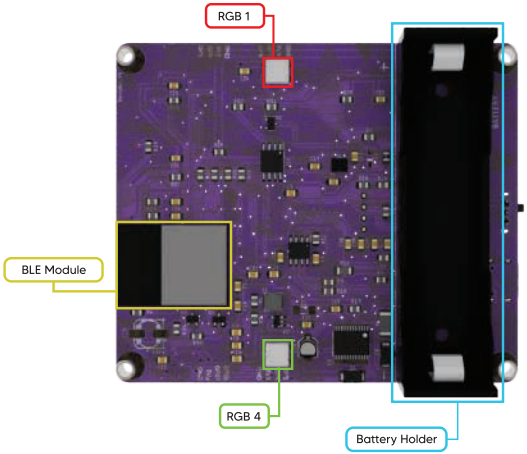
Sumo Robot

Components

Front of the Motherboard



Back of the Motherboard



BerryBot features over 25 input and output sensors on its motherboard. With its components, BerryBot becomes a project development robot that you can customize by developing your own code.

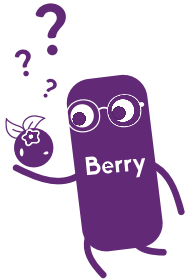
Programming Platforms

BerryBot supports both block-based and text-based programming editors. BerryBot can be programmed online with block-based and text-based coding using the PicoBricks IDE. Additionally, you can develop projects in MicroPython and C programming languages with text-based coding using editors like Thonny IDE and Arduino IDE, which you can download to your computer.

What is Block Programming?

Block-based programming is a type of visual programming designed for beginners. In block-based programming, the desired algorithm is created by dragging and dropping blocks in the editor, either stacking them vertically or nesting them. Block-based programming eliminates syntax (punctuation) errors commonly encountered in programming languages and allows for the quick and error-free creation of the desired algorithm's code.

BerryBot can be programmed using block-based programming with the PicoBricks IDE.

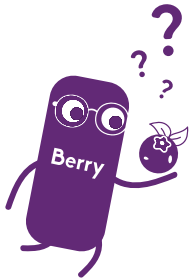


What is Text-Based Programming?

Text-based programming involves creating programs using command structures written in words. Each programming language has its own syntax rules. **C**, **C++**, and **Python** are examples of **text-based** programming languages.
BerryBot can be programmed using **text-based** programming languages with **PicoBricks IDE** - **PicoPy (MicroPython)**, **Arduino IDE (C)**, and **Thonny IDE (MicroPython)**.

How to Use BerryBot?

BerryBot can be controlled remotely with PicoBricks Go! with the modes available in PicoBricks Go!, you can run pre-made projects or control BerryBot using a joystick.



Controlling BerryBot with PicoBricks Go!

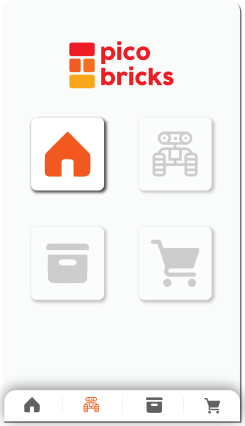
PicoBricks GO! is a mobile application that allows you to control BerryBot The REX 8 in 1 Robot Kit and PicoBricks.

Download and open PicoBricks Go! from your app store according to your phone's operating system.

rbt.ist/picobricksgomac

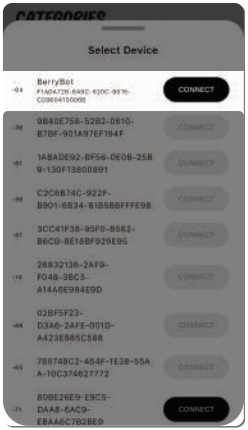
rbt.ist/picobricksgoandroid.svg

1. When you first open the app, click on the home icon in the main menu that appears.

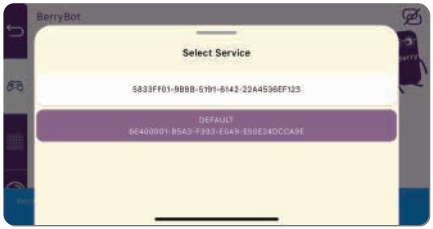


2. Select BerryBot from the categories.

3. In the "Select Device" menu, choose BerryBot. This process will enable your BerryBot to communicate with the app via Bluetooth.



4. In the "Select Service" menu, select the second default option. This will allow you to control BerryBot remotely via Bluetooth.



5. After completing these steps, you can use the gamepad screen that opens to control BerryBot or switch to the pre-made modes.

PicoBricks Go! Menu Buttons

Allows you to return to the previous screen.

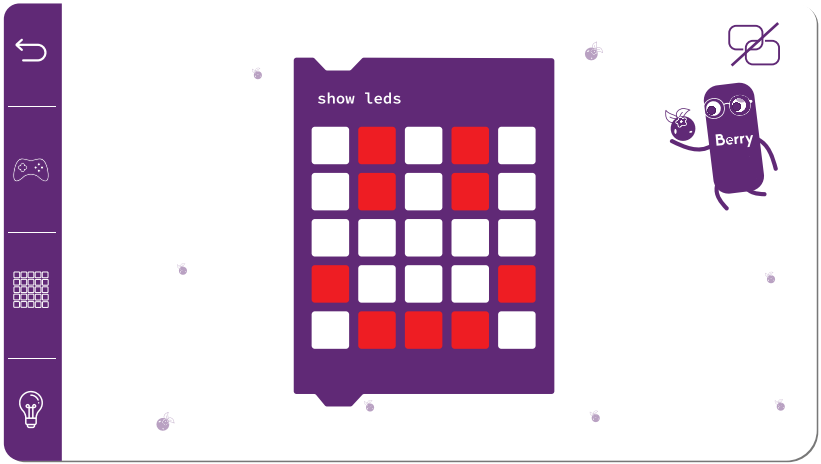
Switches to the BerryBot gamepad screen.

Switches to the screen where you can control BerryBot's matrix LEDs.

Switches to the screen where you can control BerryBot's RGB Leds.

Creating a Smiley Face Emoji on Matrix LEDs

Switch to the matrix LED screen using the menu buttons and create a smiley face by tapping on the LEDs screen as shown below.



Block Programming

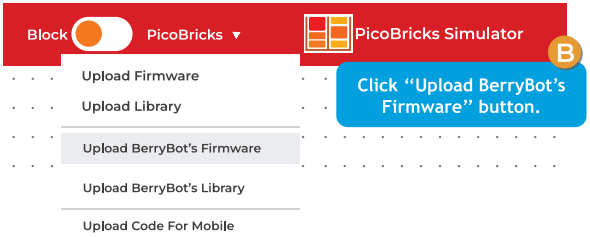
Programming with PicoBricks IDE

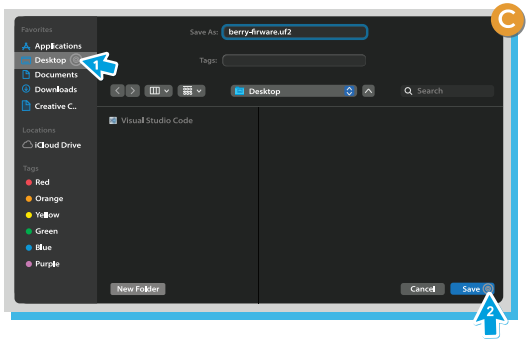
PicoBricks IDE is an online programming editor that you can use from your browser on a computer with an internet connection, without installing any software on your computer. Using the editors within PicoBricks, you can create block-based programs or text-based programs in the MicroPython language.

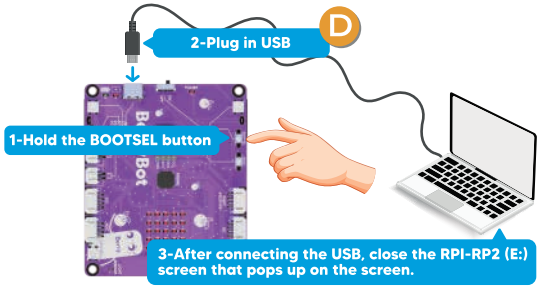
PicoBlockly

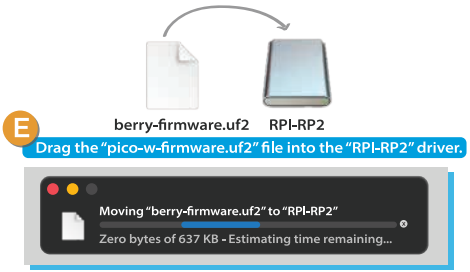

Educational + Be Our Ambassador
IDE
PicoJR
A simple and visual-based coding platform for kids who are new to robotic coding
Go
PicoBlockly
Easily code PicoBricks with block coding and see python code at the same time.
Go
PicoPY
Do high-level coding with text-based python.
Go
Simulator
With PicoBricks Simulator you can experience using PicoBricks.
Go
C
Click "PicoBlockly" button.

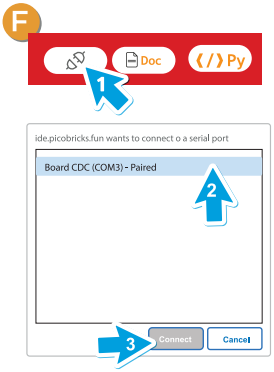

Block
PicoBricks
PicoBricks Simulator
A

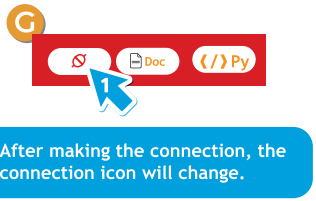

Block
PicoBricks
PicoBricks Simulator
Upload Firmware
Upload Library
Upload BerryBot's Firmware
Upload BerryBot's Library
Upload Code For Mobile
B
Click "Upload BerryBot's Firmware" button.


C
Favorites
Applications
Desktop
Documents
Downloads
Creative C..
Locations
iCloud Drive
Tags
Red
Orange
Yellow
Green
Blue
Purple
Save As: berry-firware.uf2
Tags:
Desktop
Search
Visual Studio Code
New Folder
Cancel
Save
2


D
2-Plug in USB
1-Hold the BOOTSEL button
3-After connecting the USB, close the RPI-RP2 (E:) screen that pops up on the screen.

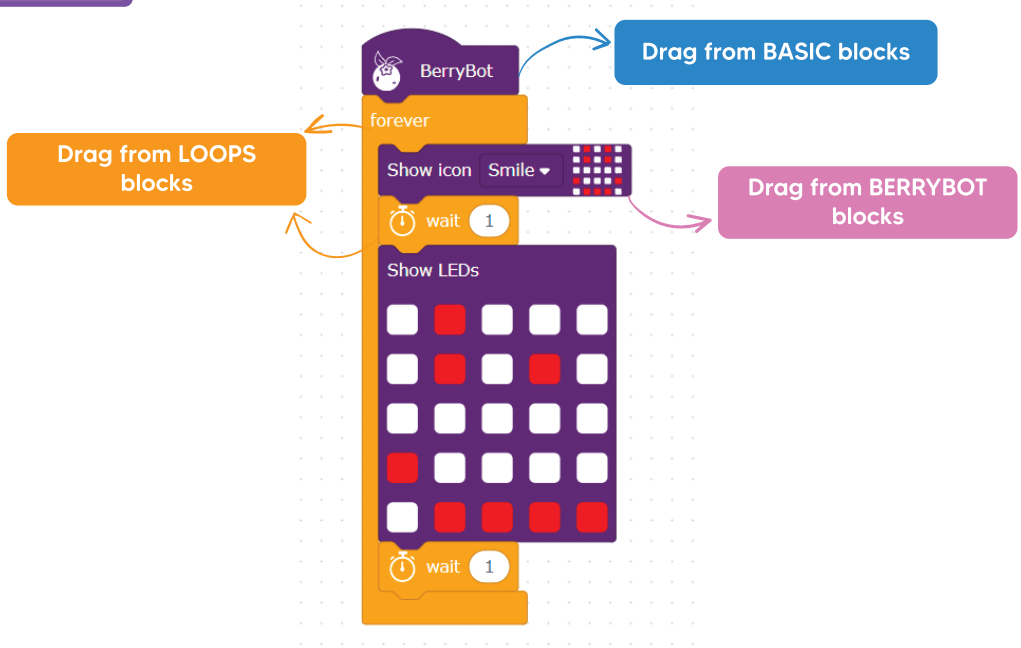

berry-firmware.uf2
RPI-RP2
E
Drag the "pico-w-firmware.uf2" file into the "RPI-RP2" driver.
Moving "berry-firmware.uf2" to "RPI-RP2"
Zero bytes of 637 KB - Estimating time remaining...


F
Doc
Py
1
ide.picobricks.fun wants to connect o a serial port
Board CDC (COM3) - Paired
2
3
Connect
Cancel


G
Doc
Py
1
After making the connection, the connection icon will change.

Lest's Start Coding With PicoBricks IDE

Blink Activity



Text-Based Programming

Programming with Thonny IDE

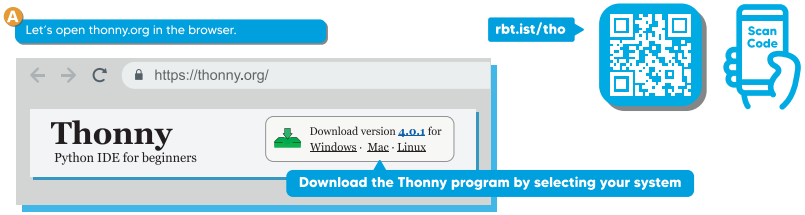
Thonny IDE is a text-based programming platform designed for the Python programming language. Using Thonny IDE, you can upload code to microcontrollers using the MicroPython programming language.

What is MicroPython?

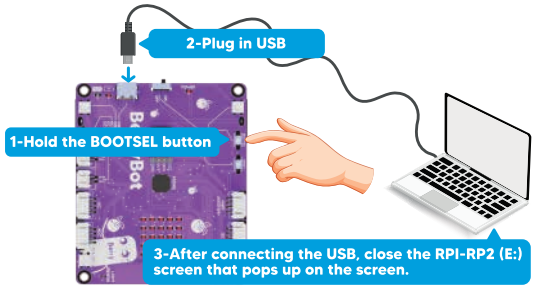
MicroPython is a software implementation designed for developing embedded software in the Python programming language. With MicroPython, you can upload code to microcontrollers using the Python programming language.

The syntax structure of MicroPython is the same as the Python programming language. Using editors like Thonny IDE, the prepared lines of code can be easily uploaded to the microcontrollers supported by the editors.

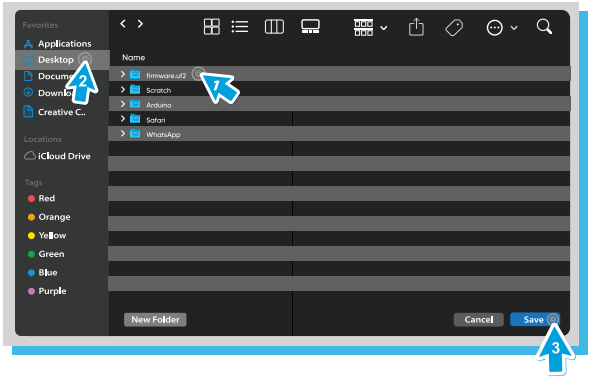
How to Use Thonny IDE

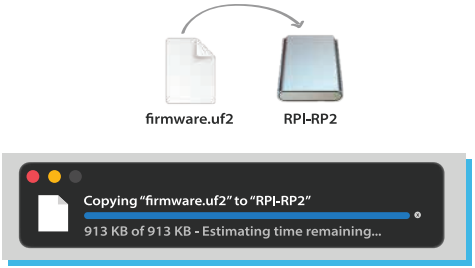


1. To put your device into boot mode, press the "BOOTSEL" button on the board and connect the cable.



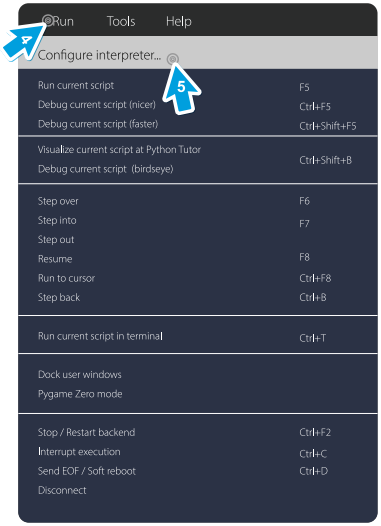
2. Go to the BerryBot GitHub page https://github.com/Robotistan/BerryBot/tree/main/Bootloader and download the file named "**firmware.uf2**".





3. Follow the steps below to upload the downloaded file to your processor.

4. Open Thonny IDE and click on the run option from the top menu.

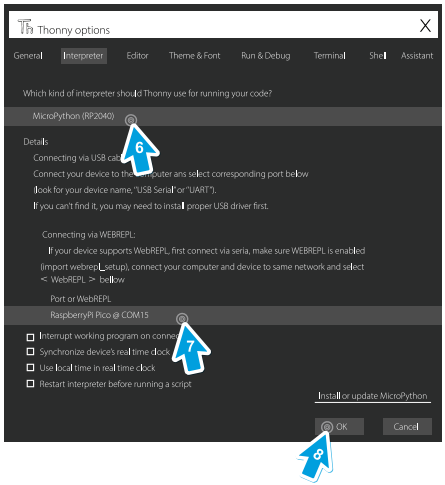


5. From the window that opens, select the **"Configure interpreter..."** option.

6. Firstly select MicroPython (RP240) from the window that opens

7. Then select the port to which your board is connected.

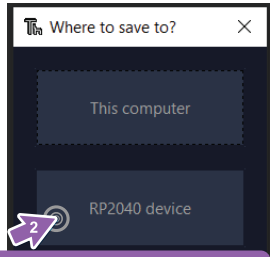
8. Finally, click the **"OK"** button.



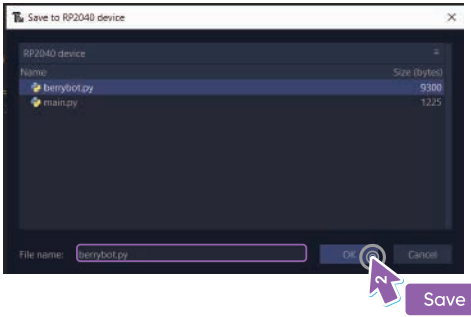
9. After the connection is complete, write the following lines of code to run your first project.

Loading the BerryBot Library

Before writing the project code, load the **"berrybot.py"** library onto the BerryBot motherboard. If you do not do this, you will get an error in your code.



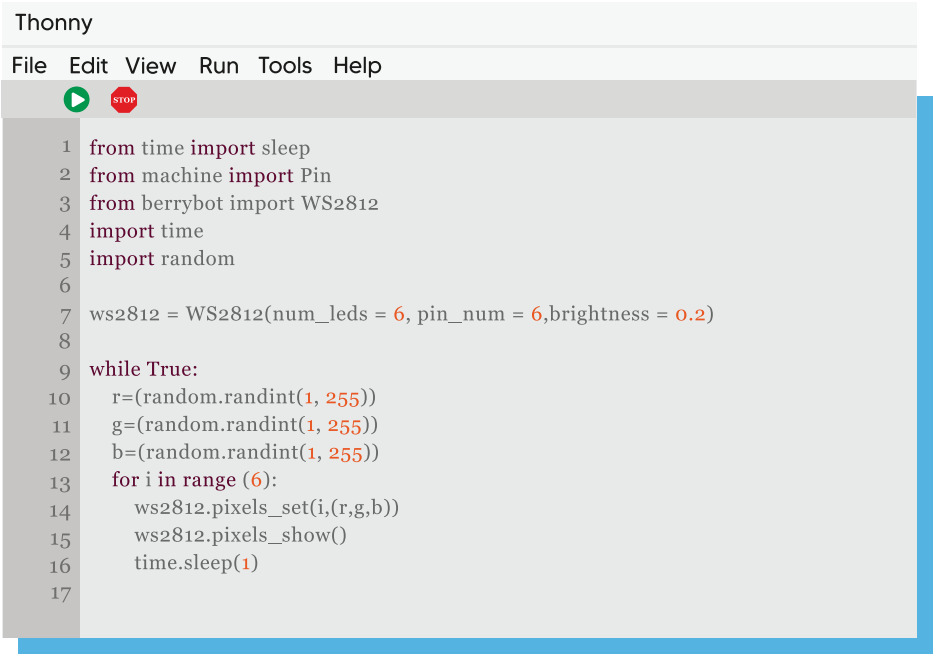
Press the Ctrl+Shift+S key combination and then select RP2040 device option.



Save it as 'berrybot.py'.

My First Project with BerryBot

By writing the following lines of code in Thonny IDE, make the BerryBot RGB LEDs light up in a randomly selected color one by one.

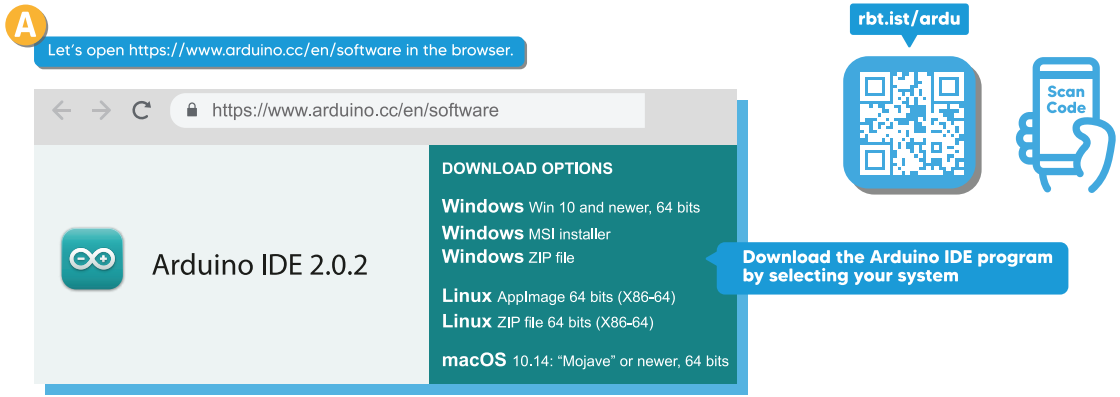


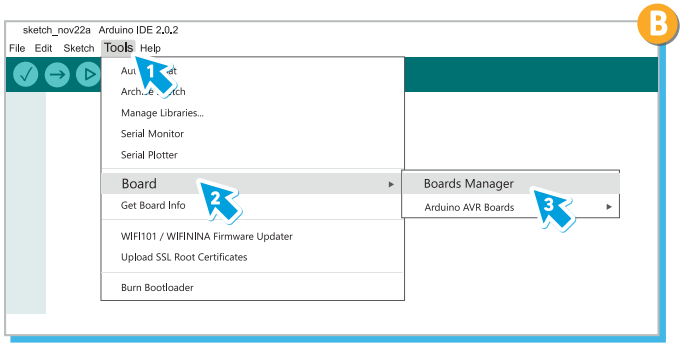

```
1  from time import sleep
2  from machine import Pin
3  from berrybot import WS2812
4  import time
5  import random
6
7  ws2812 = WS2812(num_leds = 6, pin_num = 6,brightness = 0.2)
8
9  while True:
10   r=(random.randint(1, 255))
11   g=(random.randint(1, 255))
12   b=(random.randint(1, 255))
13   for i in range (6):
14      ws2812.pixels_set(i,(r,g,b))
15      ws2812.pixels_show()
16      time.sleep(1)
17
```

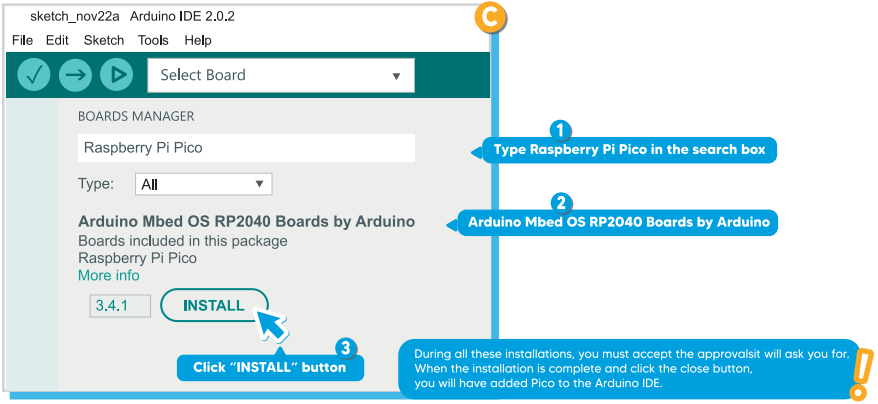
What is Arduino?

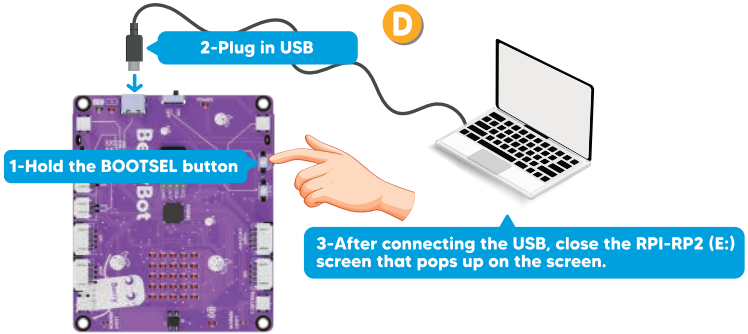
The Integrated Development Environment (IDE) for Arduino is a cross-platform application written in C and C++ languages (for Linux, macOS, Windows). It is used to write and upload programs to Arduino compatible boards, but can also be used in 3rd party cores and vendor development boards.

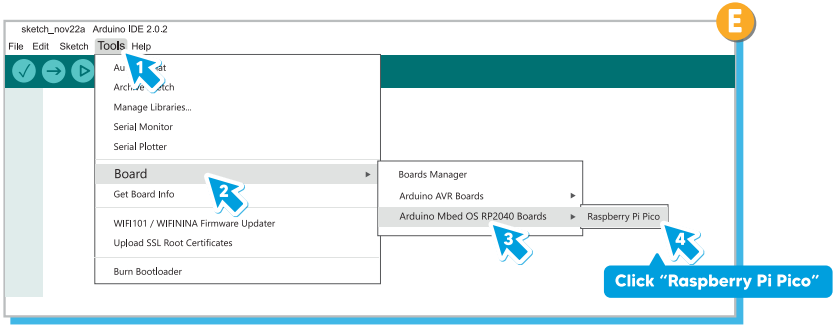
How to Use Arduino IDE?




B
sketch_nov22a Arduino IDE 2.0.2
File Edit Sketch Tools Help
Manage Libraries...
Serial Monitor
Serial Plotter
Board
Get Board Info
Boards Manager
Arduino AVR Boards
WIFI101 / WIFININA Firmware Updater
Upload SSL Root Certificates
Burn Bootloader
2
3


C
sketch_nov22a Arduino IDE 2.0.2
File Edit Sketch Tools Help
Select Board
BOARDS MANAGER
Raspberry Pi Pico
Type: All
Arduino Mbed OS RP2040 Boards by Arduino
Boards included in this package
Raspberry Pi Pico
More info
3.4.1
INSTALL
1
Type Raspberry Pi Pico in the search box
2
Arduino Mbed OS RP2040 Boards by Arduino
3
Click "INSTALL" button
During all these installations, you must accept the approvalsit will ask you for.
When the installation is complete and click the close button,
you will have added Pico to the Arduino IDE.





You are ready now. Let's start coding with Arduino

Do your first coding with Arduino IDE

Activating the Buzzer with the BerryBot Horn Activity

Arduino IDE

Fil

2

Click the "Upload" button after typing the code

Raspberry Pi Pico

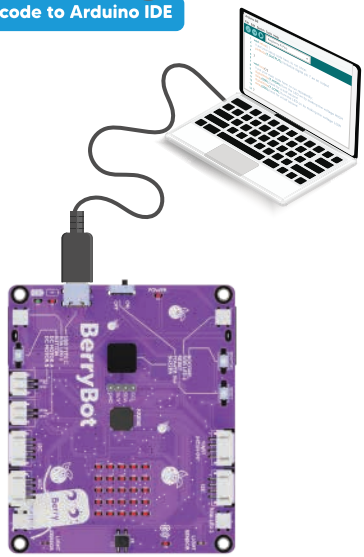
```
#include <Wire.h>

#define MODE_BUTTON 10
#define BUZZER_PIN 14

int buttonState=0;

void setup() {
  Serial.begin(115200);
  Wire.begin();
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(MODE_BUTTON, OUTPUT);

}

void loop() {
  buttonState = digitalRead(MODE_BUTTON);
  delay(100);
  if (buttonState==1){
    tone(BUZZER_PIN, 500);
  }
  else{
    noTone(BUZZER_PIN);
  }
}
```

Texts starting with the // sign are comments.
You do not need to write.

rbt.ist/blink

1

Type the code to Arduino IDE

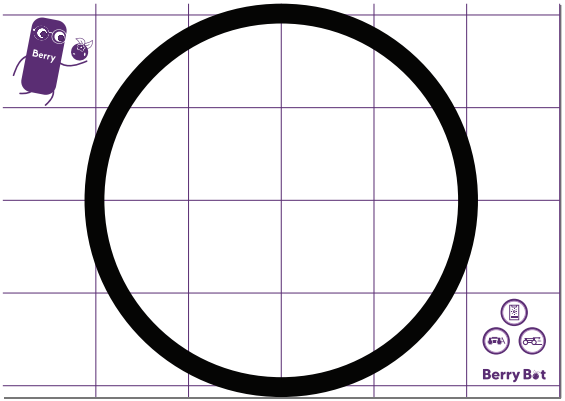


BerryBot Modes

BerryBot comes with 4 pre-made modes: obstacle avoidance, sumo robot, line following, and light following. You can quickly run these modes and perform tasks on the track using the buttons in the PicoBricks Go! application or by using the ready-made PicoBricks IDE and Thonny IDE (MicroPython) codes.

Obstacle Avoidance

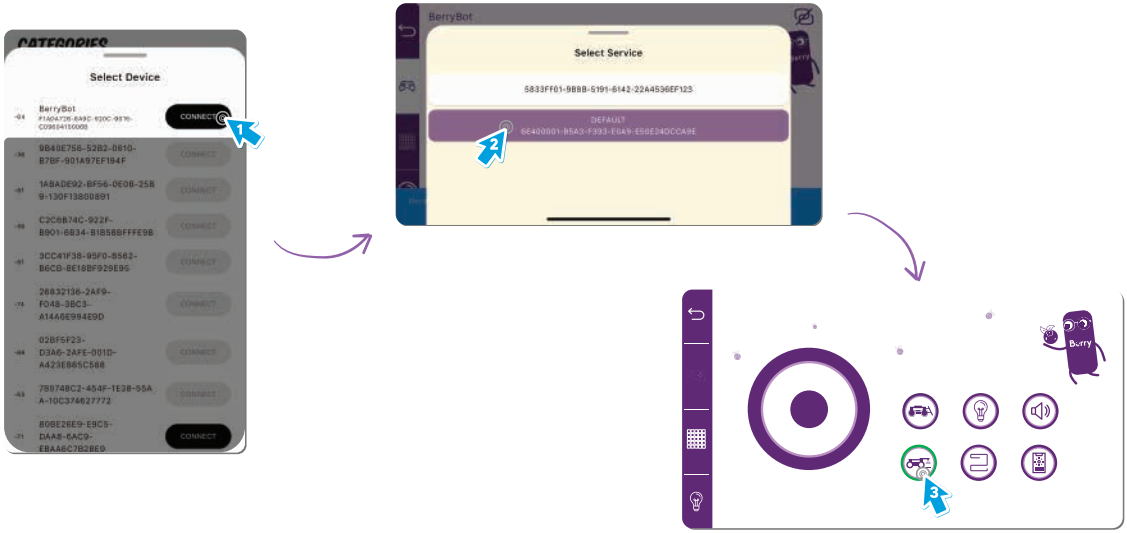
In obstacle avoidance mode, BerryBot detects objects within 12 cm in front of it using its front ultrasonic distance sensor and avoids them.



You can create a maze for BerryBot to navigate by using the checkered areas on the track.

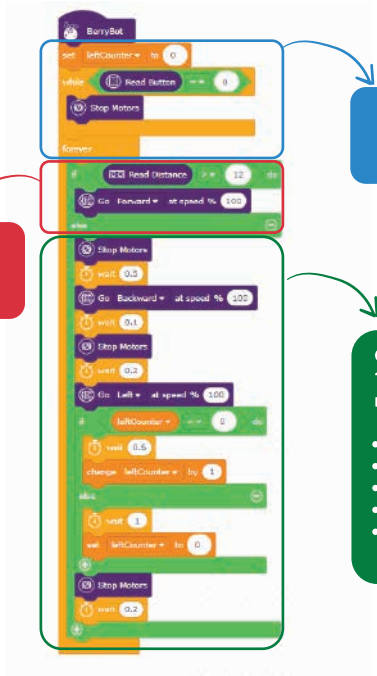
PicoBricks Go!

To run the obstacle avoidance mode in the PicoBricks Go! mobile application, you can follow the steps below.



pico bricks IDE

The code blocks for the obstacle avoidance mode prepared using PicoBricks IDE are given in the image below.



After the button is pressed, continuously repeat the following steps.

If there is no object within 12 cm in front of BerryBot, move forward at 100% power.

Otherwise, if there is an object within 12 cm in front of BerryBot, perform the following movement combination:

- Stop for 0.5 seconds
- Move backward for 0.3 seconds
- Stop for 0.2 seconds
- Turn left for 0.6 or 1 seconds
- Stop for 0.2 seconds

Thonny IDE (MicroPython)

berrybot_sonic_mode.py ×

```
##########Libraries##########
from machine import Pin,
from time import sleep
from berrybot import HCSR04, TB6612
import time
##########Pin Defination##########
MOTOR_A1_PIN = 25
MOTOR_A2_PIN = 24
MOTOR_B1_PIN = 22
MOTOR_B2_PIN = 23
MOTOR_PWM_A_PIN = 15
MOTOR_PWM_B_PIN = 21
MODE_BUTTON = 10
##########Pin Initialization##########
motor = TB6612(MOTOR_A1_PIN, MOTOR_A2_PIN, MOTOR_B1_PIN, MOTOR_B2_PIN, MOTOR_PWM_A_PIN, MOTOR_PWM_B_PIN)
sensor = HCSR04(trigger_pin=8, echo_pin=9, echo_timeout_us=10000)
push_button = Pin(MODE_BUTTON,Pin.IN,Pin.PULL_DOWN)
counter=0
right_counter=0
left_counter=0

while (push_button.value()==0):
    motor.stop()

while True:
    print(sensor.distance_cm())

    if (sensor.distance_cm()) > (12):
        motor.forward(100 * 650)
    else:
        motor.stop()
        time.sleep((0.5))
        motor.backward(100 * 650)
        time.sleep((0.1))
        motor.stop()
        time.sleep((0.2))
```

rbt.ist/obstacleavoidancemp

Scan the QR code to access the code.

Arduino IDE

sonic_mode_arduino.ino

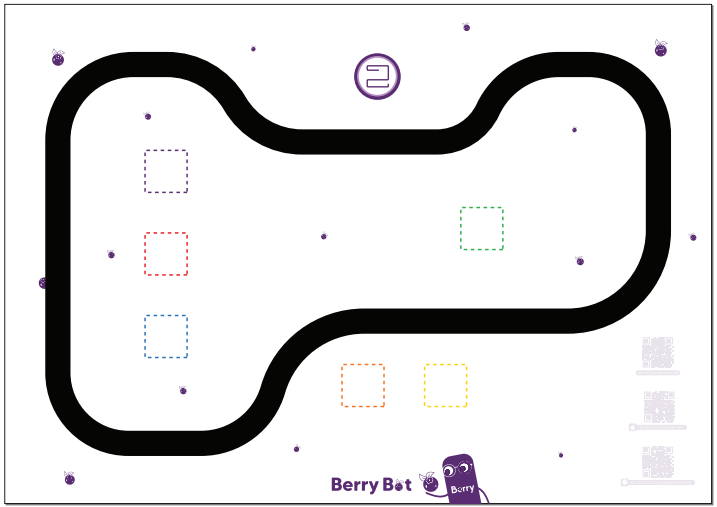
```
#include <Wire.h>
#include <stdio.h>

// Pin Defination
#define TX_PIN 0
#define RX_PIN 1
#define NEOPIXEL_PIN 6
#define TRIG_PIN 8
#define ECHO_PIN 9
#define MODE_BUTTON 10
#define BUZZER_PIN 14
#define PWM_A 15
#define IR_PIN 20
#define PWM_B 21
#define INPUT_B1 22
#define INPUT_B2 23
#define INPUT_A2 24
#define INPUT_A1 25
#define LEFT_SENSOR 26
#define RIGHT_SENSOR 27
#define LDR_R_PIN 28
#define LDR_L_PIN 29

// Variable
long duration;
int distance;
int left_counter=0;

void attachMotor()
{
  pinMode(INPUT_A1, OUTPUT);
  pinMode(INPUT_A2, OUTPUT);
  pinMode(INPUT_B1, OUTPUT);
  pinMode(INPUT_B2, OUTPUT);
  pinMode(PWM_A, OUTPUT);
  pinMode(PWM_B, OUTPUT);
}
```

rbt.ist/obstacleavoidancearduino

Scan the QR code to access the code.

Line Tracker

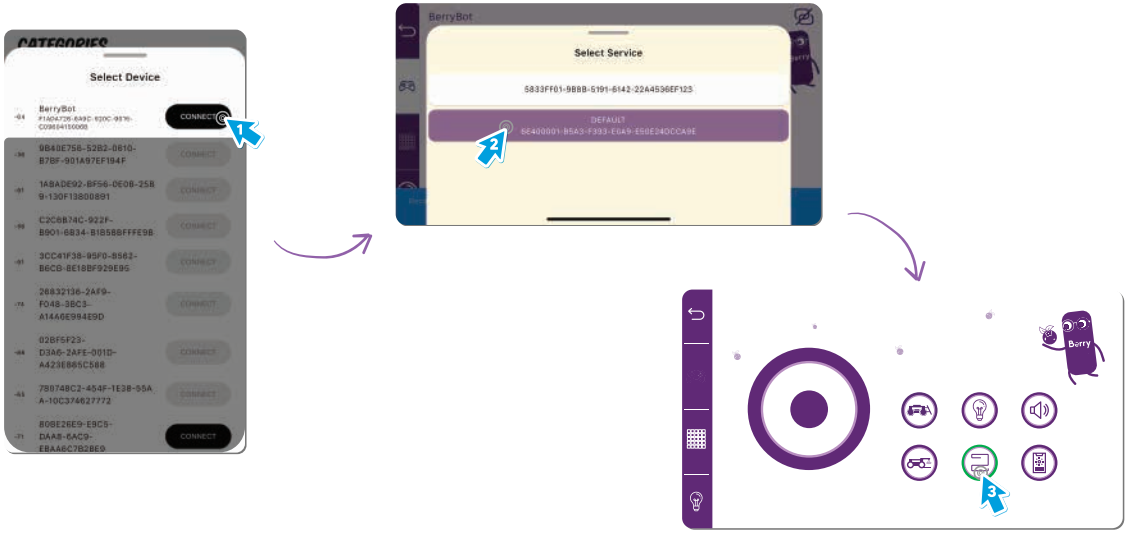
In line tracker mode, BerryBot can autonomously follow the black lines on the track using the line tracker sensor located underneath it.



You can place BerryBot on the black path on the track to make it follow the road.

PicoBricks Go!

To run the line following mode in the PicoBricks Go! mobile application, you can follow the steps below.

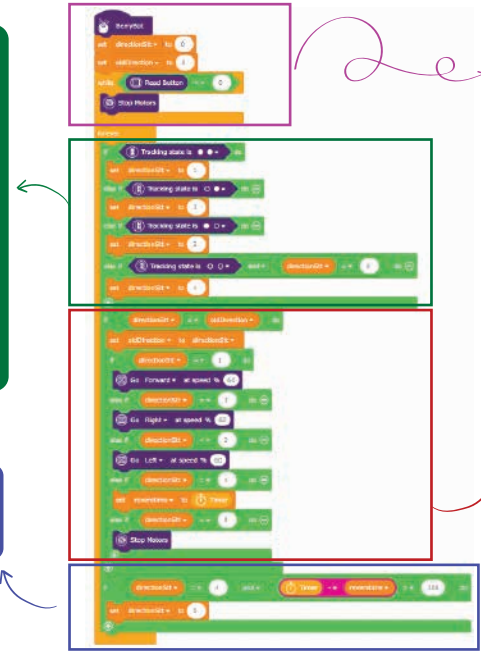


pico bricks IDE

The code blocks for the line tracker mode prepared using PicoBricks IDE are given in the image below.

Let's create the condition structure necessary to determine the direction assignments based on the values detected by the line following sensor located under BerryBot. While making these assignments, we will change the value of the "directionStt" variable that we created at the beginning of our code. The numerical equivalents of the directionvalues are as follows:

- Stop = 0
- Forward = 1
- Left = 2
- Right = 3
- Backward = 4



After starting our code with the BerryBot block, let's create two variables named "directionStt" and "oldDirection" to assign directions. Then, let's create a loop that keeps the BerryBot stationary until a button is pressed.

Let's create the condition structure where the movements that the motors need to perform based on the determined direction values are called.

Let's create the condition structure that determines the required duration value for BerryBot to move backward.

Thonny IDE (MicroPython)

lineTracker.py ×

```
##########Libraries##########
from time import sleep
from machine import Pin, PWM, UART, Timer, ADC
import time, utime
from berrybot import TB6612
##########Pin Defination##########
MOTOR_A1_PIN = 25
MOTOR_A2_PIN = 24
MOTOR_B1_PIN = 22
MOTOR_B2_PIN = 23
MOTOR_PWM_A_PIN = 15
MOTOR_PWM_B_PIN = 21
LEFT_TRACKER = 26
RIGHT_TRACKER = 27
MODE_BUTTON = 10

#############Variables####################
Mid_Speed = 58000
Low_Speed = 43000

STOP = 0
FWD = 1
LEFT = 2
RIGHT = 3
BWD = 4

threshold = 60000
directionStt = STOP
oldDirection = STOP

##########Pin Initialization##########
motor = TB6612(MOTOR_A1_PIN, MOTOR_A2_PIN, MOTOR_B1_PIN, MOTOR_B2_PIN, MOTOR_PWM_A_PIN, MOTOR_PWM_B_PIN)
leftSensor = ADC(Pin(LEFT_TRACKER))
rightSensor = ADC(Pin(RIGHT_TRACKER))
push_button = Pin(MODE_BUTTON,Pin.IN,Pin.PULL_DOWN)

```

rbt.ist/linetrackermp

Scan the QR code to access the code.

Arduino IDE

lineTracker.ino

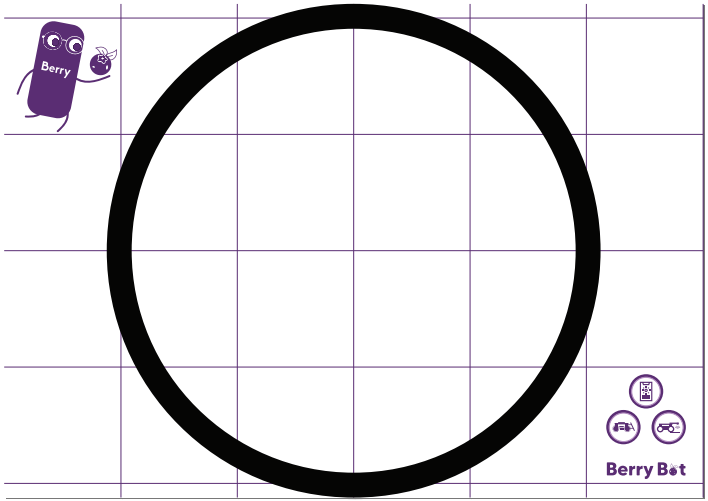
```
// Libraries
#include <Wire.h>
#include <stdio.h>

// Pin Defination
#define MOTOR_B1 22
#define MOTOR_B2 23
#define MOTOR_A2 24
#define MOTOR_A1 25
#define PWM_A 15
#define PWM_B 21
#define LEFT_SENSOR 26
#define RIGHT_SENSOR 27
#define MODE_BUTTON 10

#define TRACKER_THRESHOLD 900

#define STOP  0
#define FWD   1
#define BWD   2
#define RIGHT 3
#define LEFT  4

// Variable
int leftSensor = 0;
int rightSensor = 0;
uint8_t directionStt = STOP;
uint8_t oldDirection = STOP;
unsigned long reverseTime = 0;

void attachMotor()
{
  pinMode(MOTOR_A1, OUTPUT);
  pinMode(MOTOR_A2, OUTPUT);
  pinMode(MOTOR_B1, OUTPUT);
  pinMode(MOTOR_B2, OUTPUT);
  pinMode(PWM_A, OUTPUT);
  pinMode(PWM_B, OUTPUT);
}

```

rbt.ist/linetrackerarduino

Scan the QR code to access the code.

Sumo Robot

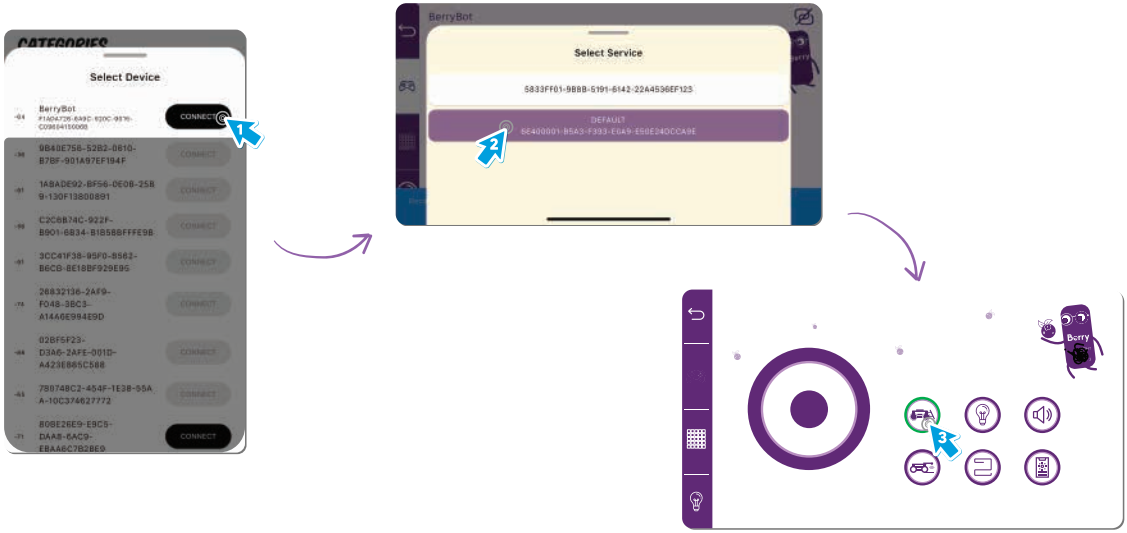
In sumo mode, when BerryBot is placed in an area surrounded by black lines on the track, it pushes objects within the area surrounded by the black line out of the area while staying within the area itself. BerryBot uses its front distance sensor and the line following sensor underneath it for this process. Using the line following sensor, it detects the color of the ground and stays within the black-lined area of the sumo track. Using its front distance sensor, it detects objects on the track.



You can use the circle on the track surrounded by black lines by placing the materials that BerryBot will release and BerryBot itself inside it.

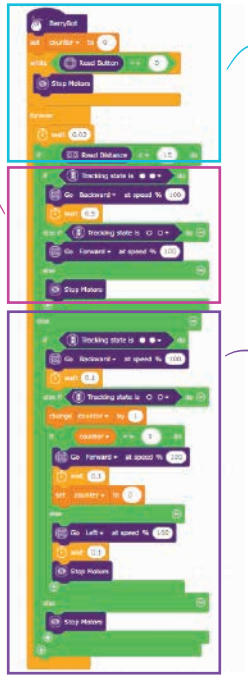
PicoBricks Go!

To run the line following mode in the PicoBricks Go! mobile application, you can follow the steps below.



pico bricks IDE

The code blocks for the line tracker mode prepared using PicoBricks IDE are given in the image below.



Let's start our code with the BerryBot block, then create a loop that stops BerryBot until the button is pressed. After that, let's create an infinite loop and set up the necessary conditional structure for the robot to detect objects in front of it.

If both receivers of BerryBot's line following sensor detect the line, let's create code blocks that make BerryBot move backward to stay within the track. If both receivers detect the white color and the distance value is less than 15 (meaning there is an object in front of it), let's create code blocks that make it move forward.

If there is no object in front of BerryBot and the receivers detect the line, let's create code blocks that make it move backward. If the receivers detect the white color, let's create code blocks that make it move left. If none of these conditions are met, let's create code blocks that stop BerryBot.

Thonny IDE (MicroPython)

sumoRobot.py ×

```
##########Libraries##########
from machine import Pin, PWM, UART, Timer, ADC
from time import sleep
from berrybot import TB6612, HCSR04
##########Pin Defination##########
MOTOR_A1_PIN = 25
MOTOR_A2_PIN = 24
MOTOR_B1_PIN = 22
MOTOR_B2_PIN = 23
MOTOR_PWM_A_PIN = 15
MOTOR_PWM_B_PIN = 21
LEFT_TRACKER = 26
RIGHT_TRACKER = 27
MODE_BUTTON = 10
############Variables###################
MotorSpeed = 50000
threshold = 60000
counter = 0
##########Pin Initialization##########
motor = TB6612(MOTOR_A1_PIN, MOTOR_A2_PIN, MOTOR_B1_PIN, MOTOR_B2_PIN, MOTOR_PWM_A_PIN, MOTOR_PWM_B_PIN)
sensor = HCSR04(trigger_pin=8, echo_pin=9, echo_timeout_us=10000)
leftSensor = ADC(Pin(LEFT_TRACKER))
rightSensor = ADC(Pin(RIGHT_TRACKER))
push_button = Pin(MODE_BUTTON,Pin.IN,Pin.PULL_DOWN)

while (push_button.value()==0):
    motor.stop()

while True:
    distance = sensor.distance_cm()
    #print(distance)

```

rbt.ist/sumorobotmp

Scan the QR code to access the code.

Arduino IDE

sumoRobot.ino

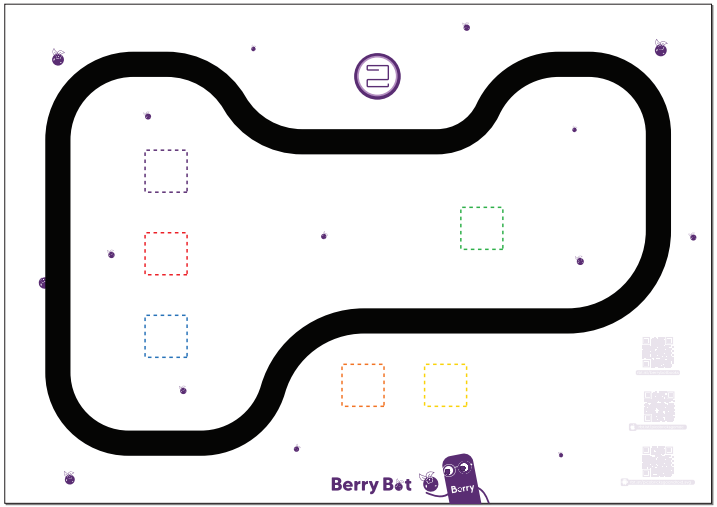
```
// Libraries
#include <Wire.h>
#include <stdio.h>

// Pin Defination
#define MOTOR_B1 22
#define MOTOR_B2 23
#define MOTOR_A2 24
#define MOTOR_A1 25
#define PWM_A 15
#define PWM_B 21
#define LEFT_SENSOR 26
#define RIGHT_SENSOR 27
#define TRIG_PIN 8
#define ECHO_PIN 9
#define MODE_BUTTON 10

#define TRACKER_THRESHOLD 1000
#define HighSpeed 255

// Variable
long duration;
int distance;
int leftSensor = 0;
int rightSensor = 0;
int counter = 0;
unsigned long reverseTime = 0;

void attachMotor()
{
  pinMode(MOTOR_A1, OUTPUT);
  pinMode(MOTOR_A2, OUTPUT);
  pinMode(MOTOR_B1, OUTPUT);
  pinMode(MOTOR_B2, OUTPUT);
  pinMode(PWM_A, OUTPUT);
  pinMode(PWM_B, OUTPUT);
}

void Forward(int speed)
{
  digitalWrite(MOTOR_A1, HIGH);
  digitalWrite(MOTOR_A2, LOW);
  digitalWrite(MOTOR_B1, HIGH);
  digitalWrite(MOTOR_B2, LOW);
  analogWrite(PWM_A, speed);
  analogWrite(PWM_B, speed);
}
```

rbt.ist/sumorobotarduino

Scan the QR code to access the code.

Light Tracker

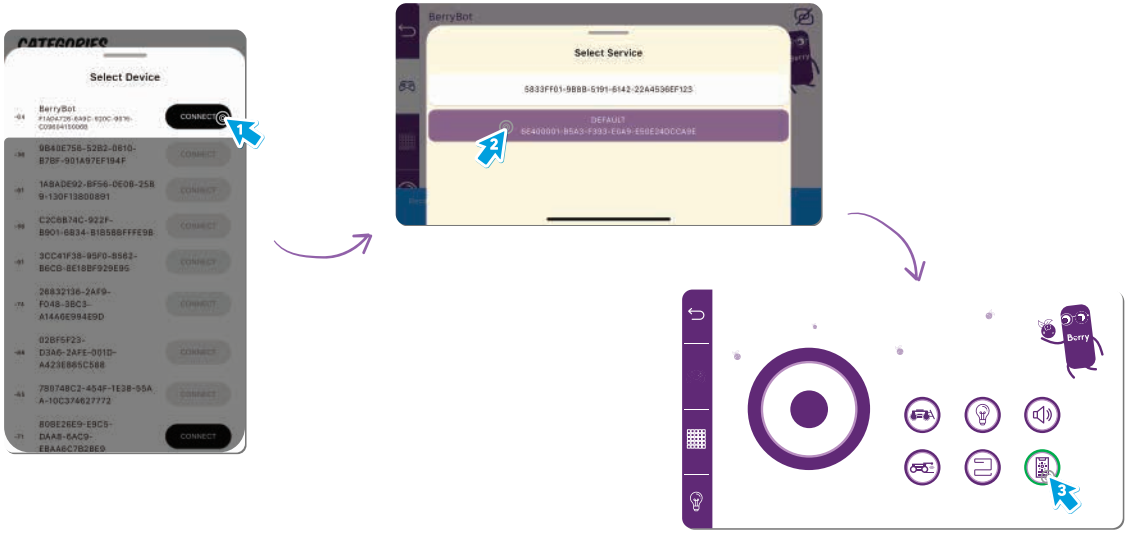
In light tracker mode, BerryBot uses the data from two light sensors on its main board to move towards the area with more light. When using this mode, you can direct any light source towards BerryBot, and it will move towards the light source.



Using the checkered areas on the track and a light source, you can move BerryBot to the desired coordinates.

PicoBricks Go!

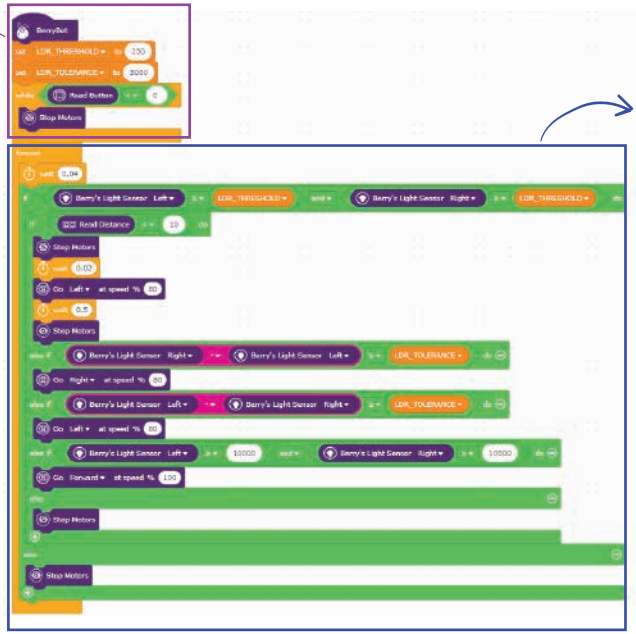
To run the line following mode in the PicoBricks Go! mobile application, you can follow the steps below.



pico bricks IDE

The code blocks for the light following mode prepared using PicoBricks IDE are given in the image below.

After starting our code with the BerryBot block, let's create a loop that keeps BerryBot stopped until the button is pressed. Then, let's define two variables for the tolerance and threshold values that the light sensor will detect and assign their values.



In the marked area, the necessary condition structure for BerryBot to follow the light is created. For BerryBot to follow the light:

If the data from the left and right light sensors is greater than or equal to the threshold value:

If there is any object in front of BerryBot, perform the stop, turn left, and stop movements.

If there is no object in front of it and the data from the right light sensor is greater, move right.

If there is no object in front of it and the data from the left light sensor is greater, move left

If none of these conditions are met, stop.

Thonny IDE (MicroPython)

lightTracker.py ×

```
##########Libraries##########
from time import sleep
from machine import Pin, PWM, UART, Timer, ADC
import time, utime
from berrybot import TB6612, HCSR04
##########Pin Defination##########
MOTOR_A1_PIN = 25
MOTOR_A2_PIN = 24
MOTOR_B1_PIN = 22
MOTOR_B2_PIN = 23
MOTOR_PWM_A_PIN = 15
MOTOR_PWM_B_PIN = 21
LDR_L_PIN = 29
LDR_R_PIN = 28
MODE_BUTTON = 10
#############Variables####################
LDR_THRESHOLD = 250
LDR_TOLERANCE = 5000

##########Pin Initialization##########
motor = TB6612(MOTOR_A1_PIN, MOTOR_A2_PIN, MOTOR_B1_PIN, MOTOR_B2_PIN, MOTOR_PWM_A_PIN, MOTOR_PWM_B_PIN)
sensor = HCSR04(trigger_pin=8, echo_pin=9, echo_timeout_us=10000)
ldr_left = ADC(LDR_L_PIN)
ldr_right = ADC(LDR_R_PIN)
push_button = Pin(MODE_BUTTON,Pin.IN,Pin.PULL_DOWN)
while (push_button.value()==0):
    motor.stop()

while True:

    LDR_L = ldr_left.read_u16()
    LDR_R = ldr_right.read_u16()
```

rbt.ist/lighttrackermp

Scan the QR code to access the code.

Arduino IDE

lineTracker.ino

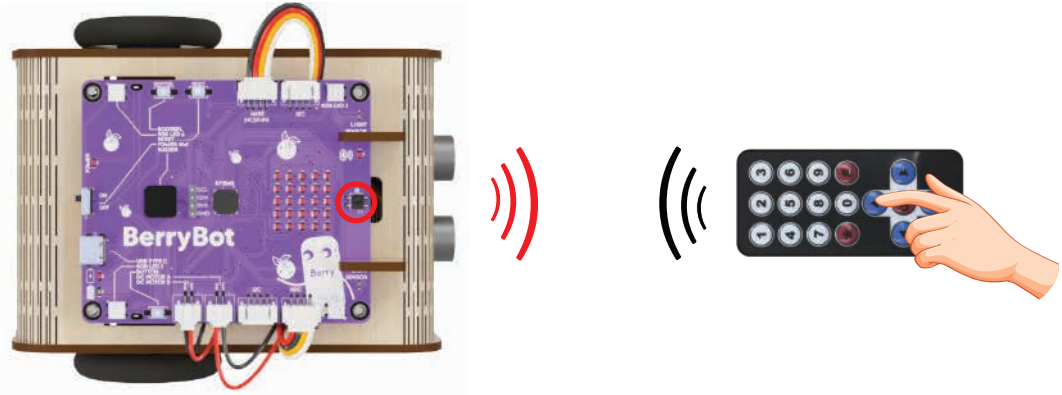
```
// Libraries
#include <Wire.h>
#include <stdio.h>

// Pin Defination
#define MOTOR_B1 22
#define MOTOR_B2 23
#define MOTOR_A2 24
#define MOTOR_A1 25
#define PWM_A 15
#define PWM_B 21
#define LEFT_SENSOR 26
#define RIGHT_SENSOR 27
#define MODE_BUTTON 10

#define TRACKER_THRESHOLD 900

#define STOP  0
#define FWD   1
#define BWD   2
#define RIGHT 3
#define LEFT  4

// Variable
int leftSensor = 0;
int rightSensor = 0;
uint8_t directionStt = STOP;
uint8_t oldDirection = STOP;
unsigned long reverseTime = 0;

void attachMotor()
{
  pinMode(MOTOR_A1, OUTPUT);
  pinMode(MOTOR_A2, OUTPUT);
  pinMode(MOTOR_B1, OUTPUT);
  pinMode(MOTOR_B2, OUTPUT);
  pinMode(PWM_A, OUTPUT);
  pinMode(PWM_B, OUTPUT);
}

```

rbt.ist/lighttrackerarduino

Scan the QR code to access the code.

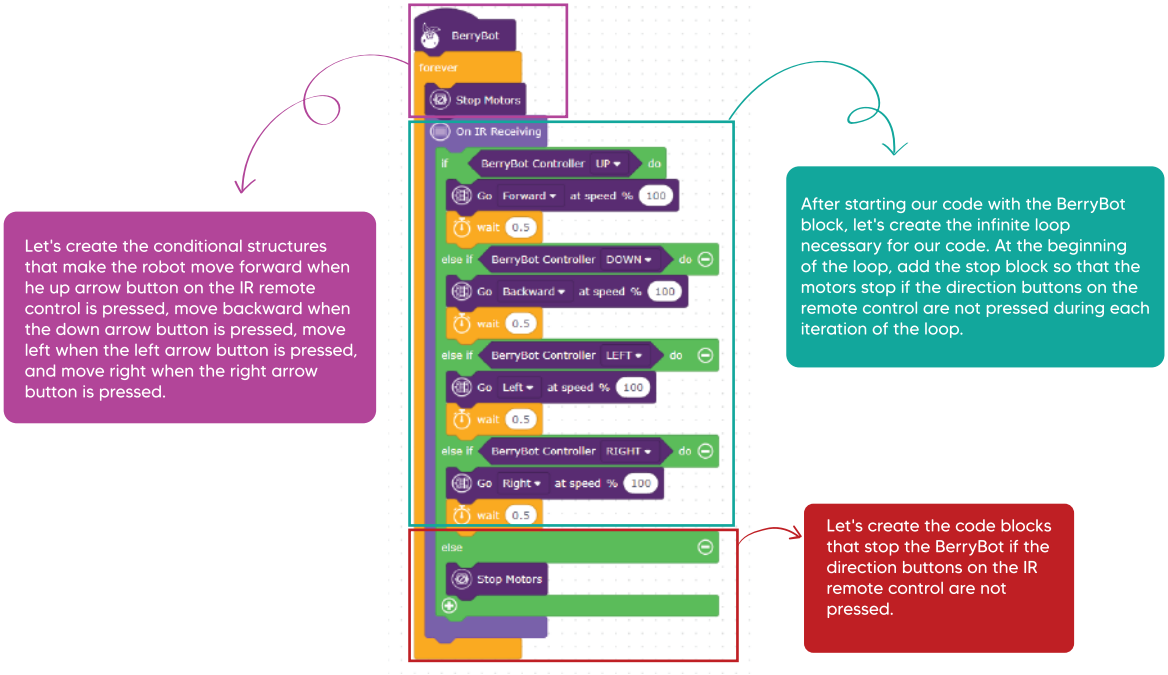
IR Remote Control

In IR remote control mode, the BerryBot can be controlled using the buttons on the remote control included in the set and the IR receiver sensor on the mainboard.



pico bricks IDE

The code blocks for the line tracker mode prepared using PicoBricks IDE are given in the image below.



Let's create the conditional structures that make the robot move forward when he up arrow button on the IR remote control is pressed, move backward when the down arrow button is pressed, move left when the left arrow button is pressed, and move right when the right arrow button is pressed.

After starting our code with the BerryBot block, let's create the infinite loop necessary for our code. At the beginning of the loop, add the stop block so that the motors stop if the direction buttons on the remote control are not pressed during each iteration of the loop.

Let's create the code blocks that stop the BerryBot if the direction buttons on the IR remote control are not pressed.

Thonny IDE (MicroPython)

remote.py *

```
##########Libraries##########
from machine import Pin, PWM, Timer, ADC
import time, utime
from berrybot import TB6612, NEC_ABC, NEC_16, IR_RX
##########Pin Defination##########
TX_PIN = 0
RX_PIN = 1
MOTOR_A1_PIN = 25
MOTOR_A2_PIN = 24
MOTOR_B1_PIN = 22
MOTOR_B2_PIN = 23
MOTOR_PWM_A_PIN = 15
MOTOR_PWM_B_PIN = 21
IR_PIN = 20
############Variables####################
ir_data = 0
data_rcvd = False
##########Function Declaration##########
def ir_callback(data, addr, ctrl):
    global ir_data
    global ir_addr, data_rcvd
    if data > 0:
        ir_data = data
        ir_addr = addr
        print('Data {:02x} Addr {:04x}'.format(data, addr))
        data_rcvd = True

ir = NEC_16(Pin(IR_PIN, Pin.IN), ir_callback)
motor = TB6612(MOTOR_A1_PIN, MOTOR_A2_PIN, MOTOR_B1_PIN, MOTOR_B2_PIN, MOTOR_PWM_A_PIN, MOTOR_PWM_B_PIN)

while True:
    motor.stop()
    if data_rcvd == True:
        data_rcvd = False
        if ir_data == IR_RX.number_up:
            motor.forward(65535)
            time.sleep_ms(500)
        elif ir_data == IR_RX.number_down:
```

rbt.ist/remotemp

Scan the QR code to access the code.

Arduino IDE

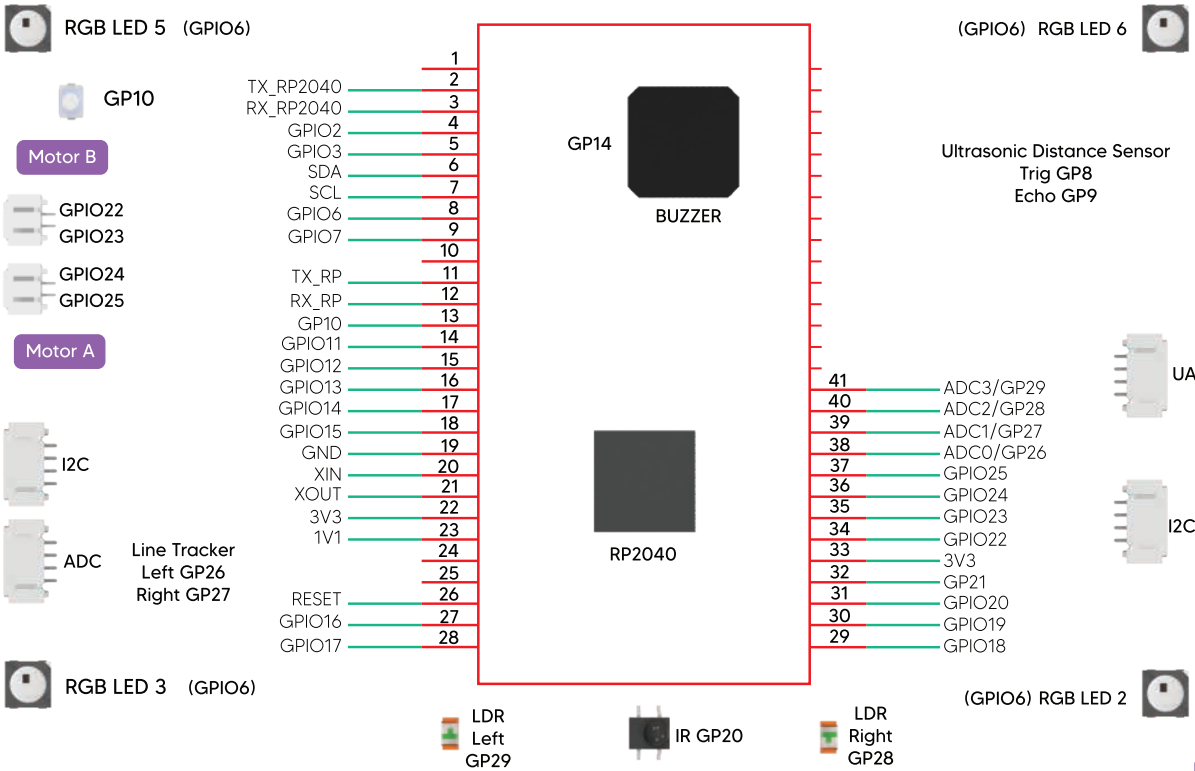
remote.ino

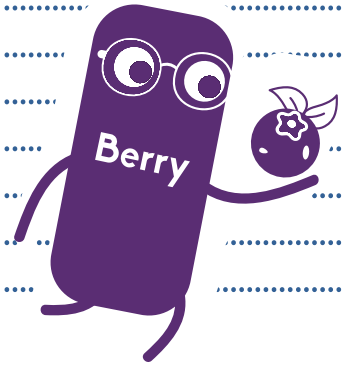
```
// Libraries
#include <IRremote.h>

// Pin Definition
#define IR_PIN 20
#define PWM_A 15
#define PWM_B 21
#define MOTOR_B1 22
#define MOTOR_B2 23
#define MOTOR_A2 24
#define MOTOR_A1 25

//IR Button Numbers
#define number_1 69
#define number_2 70
#define number_3 71
#define number_4 68
#define number_5 64
#define number_6 67
#define number_7 7
#define number_8 21
#define number_9 9
#define number_0 25
#define button_up 24
#define button_down 82
#define button_right 90
#define button_left 8
#define button_ok 28

#define DECODE_NEC

void attachMotor()
{
  pinMode(MOTOR_A1, OUTPUT);
  pinMode(MOTOR_A2, OUTPUT);
  pinMode(MOTOR_B1, OUTPUT);
  pinMode(MOTOR_B2, OUTPUT);
  pinMode(PWM_A, OUTPUT);
  pinMode(PWM_B, OUTPUT);
}

```

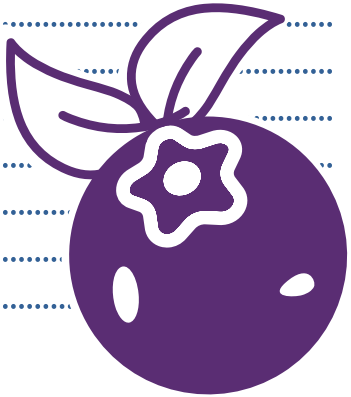
rbt.ist/remotearduino

Scan the QR code to access the code.

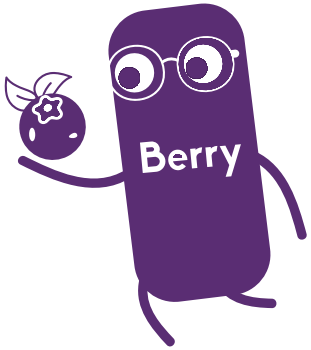
BerryBot Pinout




Berry



BerryBot



GitHub

rbt.ist/berrybotgithub

RobotistanINC

Selim GAYRETLI (Content) - Mehmet Suat MORKAN (Editor) - Atakan OZTURK (Hardware) - Elanur Tokalak (Designer) - Sercan OKAY (Instrial Designer)

3 Germany Drive STE 4 #1430 Wilmington, DE 19804
hello@robotistan.com