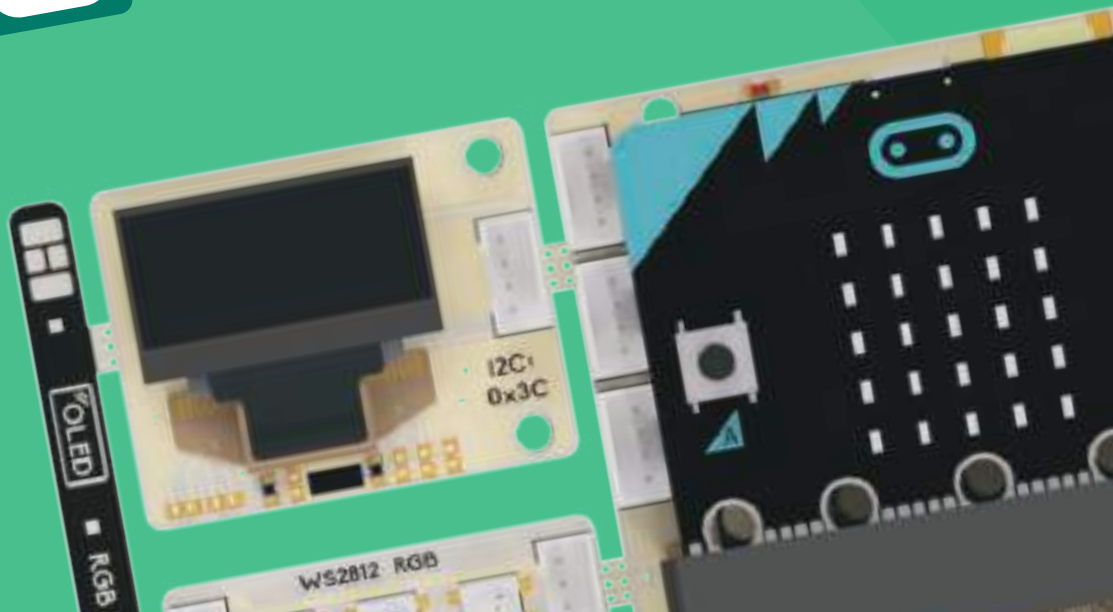


# MakeCode

## Extension



# Project Book



Copyright © 2024 Robotistan

Except for commercial usage, you can copy, reproduce and edit photos and content in this book by referring

**Author:** Selim Gayretli

**Translate & Editor:** Naze Gizem Ozer

**Designer:** Elanur Tokalak

## PicoBricks Developer Team

Project Manager - Yasir Çiçek

Chief Developer - M. Suat Morkan

Product Developer - Naze Gizem Ozer

Industrial Design Developer - Sercan Okay

Graphic Designer Developer - Elanur Tokalak

Hardware & Software Developer - Atakan Ozturk

Learning Content & Software Developer - Selim Gayretli



Powered by



What Is PicoBricks? .....	5-6
What Is Micro:Bit? .....	7
What Is MakeCode? .....	8-10
<b>PROJECTS</b>	
Blink .....	11-16
Action - Reaction .....	17-22
Color Cards .....	23-26
PicoBricks Piano .....	27-31
RGB LED Control Panel .....	32-38
Thermometer .....	39-44
Smart Cooler .....	45-50
Night and Day .....	51-59
Fizz - Buzz .....	60-66
Depth Meter .....	67-72
Morse Code Cryptography .....	73-78
Car Parking System .....	79-82
Table Lamp .....	83-88
IoT Control Panel .....	89-96
IoT Vase .....	97-106
Coin Dispenser.....	107-112
Gesture Controlled ARM Pan Tilt .....	113-119
3D Labyrinth .....	120-125
Radar .....	126-134
PicoBricks Logo Lamp .....	135-138

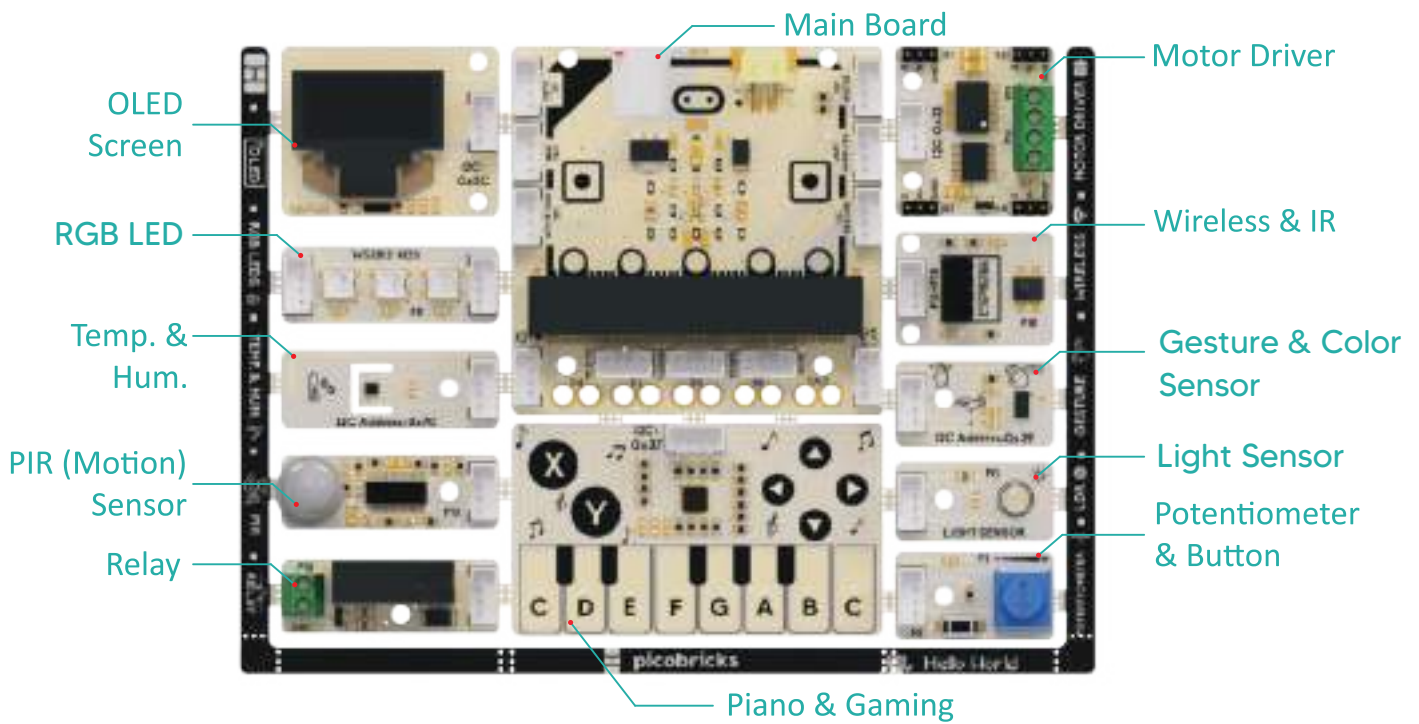
# CONTENTS

## ADD ON

London Eye .....	<b>139-147</b>
Mars Explorer .....	<b>148-162</b>
Trash Tech .....	<b>163-173</b>
Money Box .....	<b>174-186</b>
Safe Box .....	<b>187-200</b>

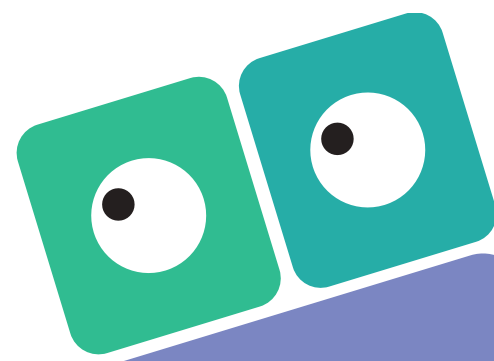


# What Is PicoBricks?



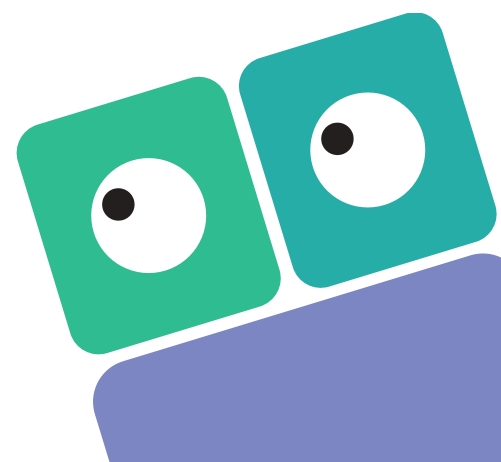
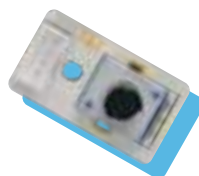
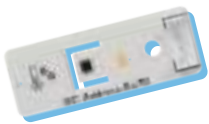
PicoBricks is a development board that eliminates the difficulties experienced in physical programming. You can invest the time saved from these challenges to create more creative projects.

Thanks to its modular structure, PicoBricks eliminates challenges such as soldering, cable clutter, incorrect pin connections, etc., experienced in physical programming. Additionally, its microcontroller board Micro:Bit, being easily programmable and supporting various coding platforms, further eradicates programming difficulties during the coding phase.





PicoBricks supports both block-based and text-based programming tools. With MakeCode Blocks and MicroBlocks IDE, we can code our projects quickly by using block-based programming. Block-based programming tools eliminate many difficulties such as punctuation marks and functions while writing code. This makes it an effective method for developing algorithmic skills necessary for programming education, especially for young age groups or beginners. With PicoBricks, while developing projects, we can easily create complex projects by simply dragging a few code blocks onto our project page by using MakeCode and MicroBlocks programs. Additionally, PicoBricks supports the C programming language in Arduino IDE and the MicroPython programming language in Thonny IDE. Arduino IDE and Thonny IDE are the most commonly used programming tools for physical programming education among text-based programming tools. Thonny IDE eliminates punctuation (Syntax) errors frequently encountered in text-based programming languages due to its support for the MicroPython language.

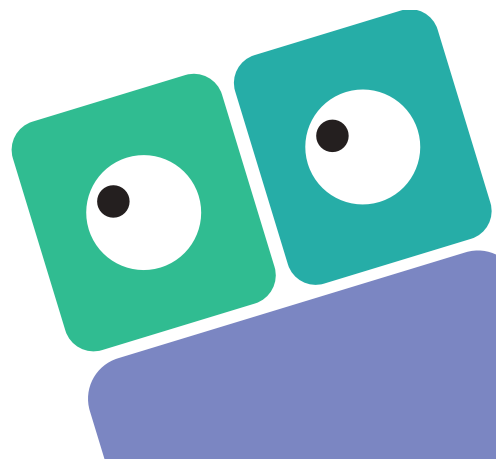
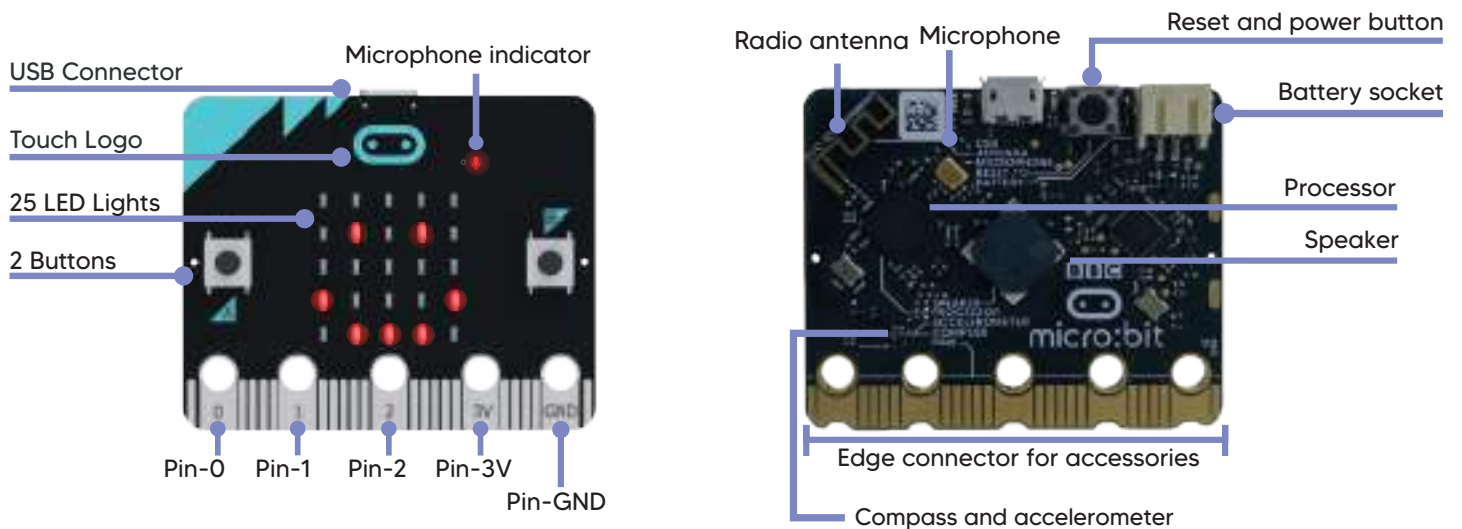


# What Is Micro:Bit?

The Micro:Bit is a microcontroller board that features a 5x5 LED matrix, 2 buttons, an accelerometer, a compass, a speaker, and a microphone sensor on its front face. Additionally, you can connect various sensors to the Micro:Bit through the pinouts on its underside.

With PicoBricks, we can connect 13 different sensors to the Micro:Bit without the need for jumper cable connections.

After placing the Micro:Bit on the PicoBricks Main Board Module, we can use the following sensors without the need for jumper cable connections.





# What Is MakeCode?

MakeCode for Micro:Bit is a block-based software development editor developed by Microsoft. With MakeCode, you can quickly and easily create programming steps for robotic coding projects prepared using the Micro:Bit microcontroller board.

Using the simulation feature, you can simulate your code blocks even if your circuit is not ready, and at the same time, you can examine the Python or JavaScript output of the code blocks you have created within the editor.

MakeCode is not just an editor but also a maker environment where you can publish your projects. By publishing your prepared projects on your MakeCode account, you can share them with us through <https://community.robotistan.com/>.

## How to Use MakeCode Micro:Bit

- Let's open "<https://makecode.microbit.org/#>" in the browser.



**A** [makecode.microbit.org/#](https://makecode.microbit.org/#)

**B** My Projects View All  
New Project  
Color Cards Project Getting Started with PicoBricks  
2 minutes ago

**C** Create a Project  
Give the project a name  
firstProjectBlink  
Create

**D** Connect Device  
Download as File  
Help  
Download firstProjectBlink

Click the "New Project" button.

Give project a name and click "Create" button.

Select "Connect Device".



1. Connect your micro:bit to your computer

E



4

Click the "Next" button.



2. Pair your micro:bit to your browser

F

Press the Pair button below.

A window will appear in the top of your browser.

Select the micro:bit device and click Connect.



5

Click the "Pair" button.



makecode.microbit.org wants to connect

G

BBC micro:bit ONY'S-DAP

Click the "Connect" button.

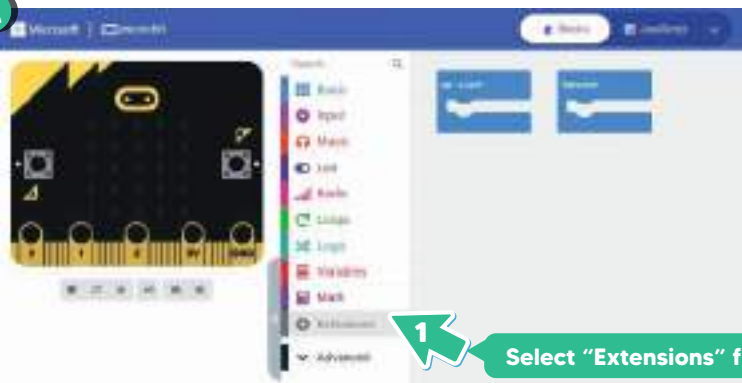
6

Connect

Cancel

## How to Use MakeCode Micro:Bit

A



1

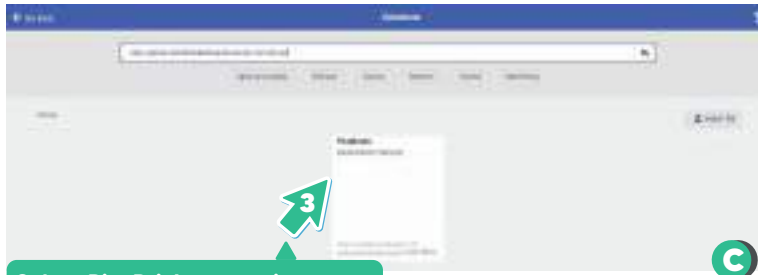
Select "Extensions" from the block menu.

Paste this URL (<https://github.com/Robotistan/picobricks-for-microbit-ext>) from the search panel.

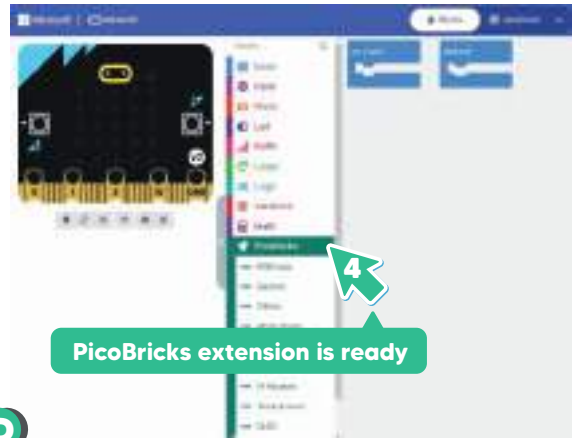
2



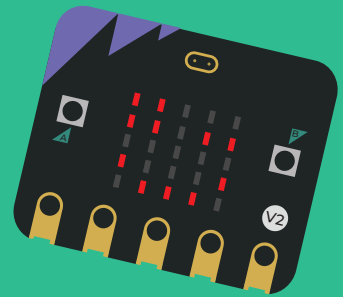
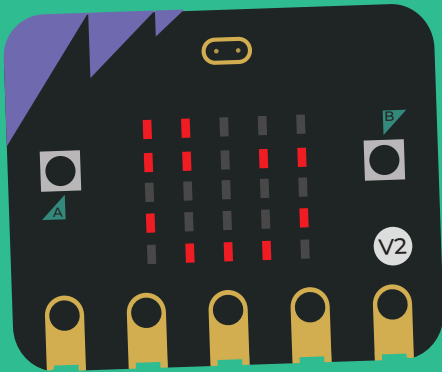
B



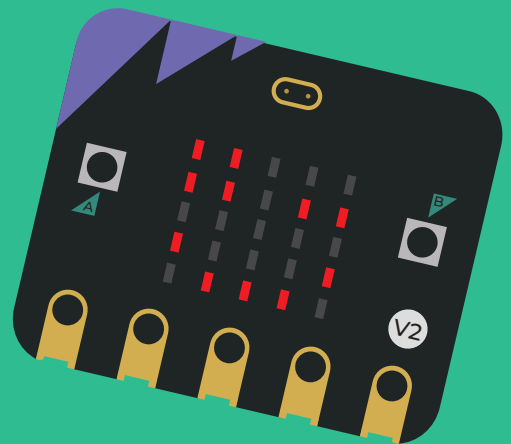
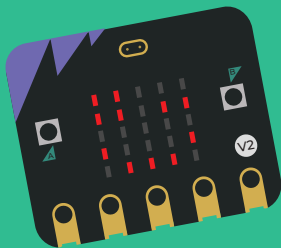
Select PicoBricks extension.



PicoBricks extension is ready



# Blink



# Blink Project

An employee starting a new job in real life usually takes on the most basic tasks. A janitor learns to use a broom, a chef learns kitchen utensils, and a waiter learns tray carrying. We can multiply these examples. The first code written by those who are new to software development is generally known as "Hello World." The language they use prints "Hello World" to the screen or console window when the program starts, marking the initial step in programming. It's akin to a baby starting to crawl... The first step in robotic coding, also known as physical programming, is the Blink application. In robotic coding, blinking symbolizes a significant moment. Simply by connecting an LED to the circuit board and coding it, the LED can be made to continuously blink. Ask individuals who have developed themselves in the field of robotic coding how they reached this level. The answer they will give you typically starts with: "It all began with lighting up an LED!"

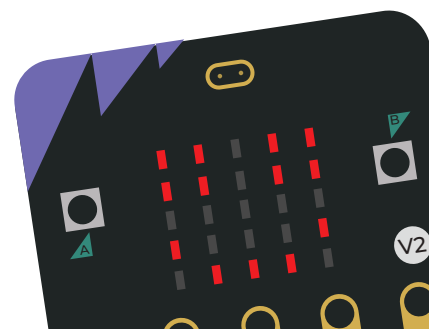
LEDs and screens are electronic circuit components that provide visual output. Thanks to output elements, a programmer can concretely determine at which stage their code is progressing. With PicoBricks, Micro:Bit includes 25 LEDs (5x5 Matrix) and a 128x64 OLED screen. When starting robotic coding with PicoBricks, printing "Hello World" on the OLED screen and winking with matrix LEDs are equally straightforward.

## Project Details:

Project Link: [https://makecode.microbit.org/\\_Pzk0xHhbbYtK](https://makecode.microbit.org/_Pzk0xHhbbYtK)

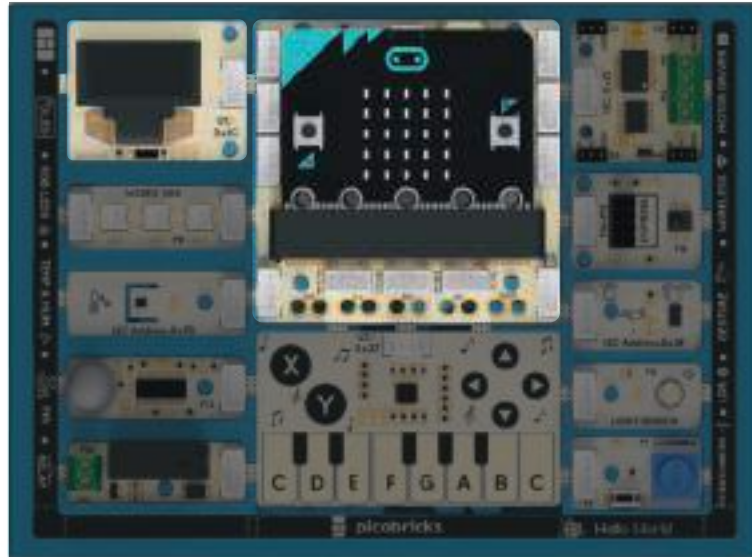


In this project, we will make the emoji we created using the Micro:Bit Matrix LEDs wink while displaying "Hello World" on the OLED screen.

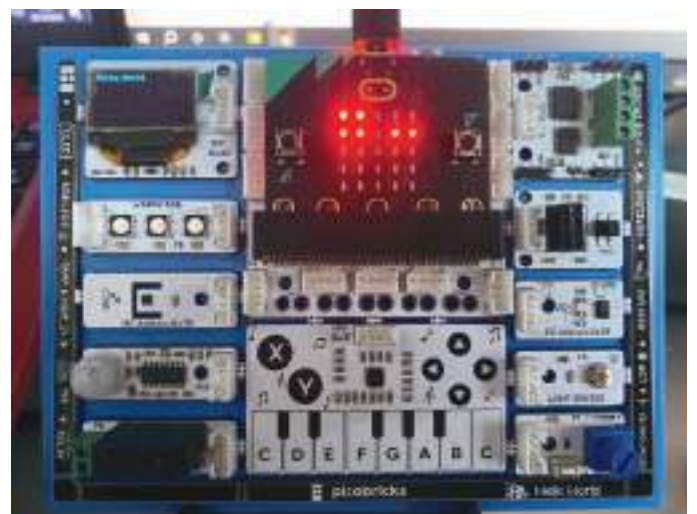
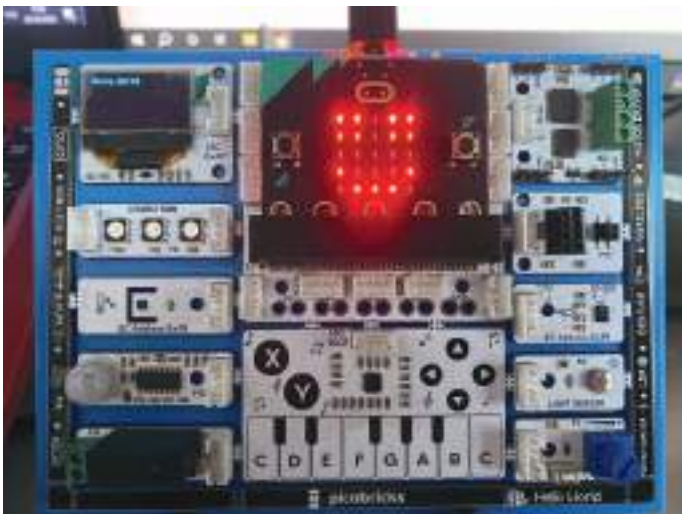


## ● Connection Diagram:

You can prepare this project without making any wiring connections.



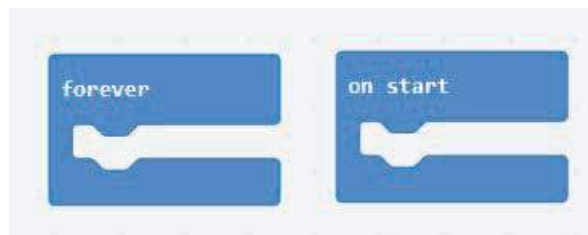
## ● Project Images:



## ● MakeCode Code of The Project:

If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

1. When we first open the project page, we encounter "forever" and "on start" blocks on the screen. The "forever" block ensures that the dragged expressions inside it run continuously when the project is executed. "On start," on the other hand, runs the expressions inside it once when the project is executed and then stops.



2. Drag the code block from the OLED blocks in the PicoBricks extension that initializes the OLED screen to start the OLED screen when the project is run.



3. Drag the code block from the OLED blocks in the PicoBricks library that writes the desired string expression on the OLED, and write the desired expression in the blank space.



4. Drag the “show leds” block from the “Basic” blocks, and click on the Micro:bit Matrix LEDs to create a smiling face emoji.



5. Let's use the 'pause (ms)' block from the 'Basic' blocks to make our code wait for 100 milliseconds.



6. Drag the "show leds" block from the "Basic" blocks, and by clicking on the Micro:Bit Matrix LEDs, create a winking face emoji.



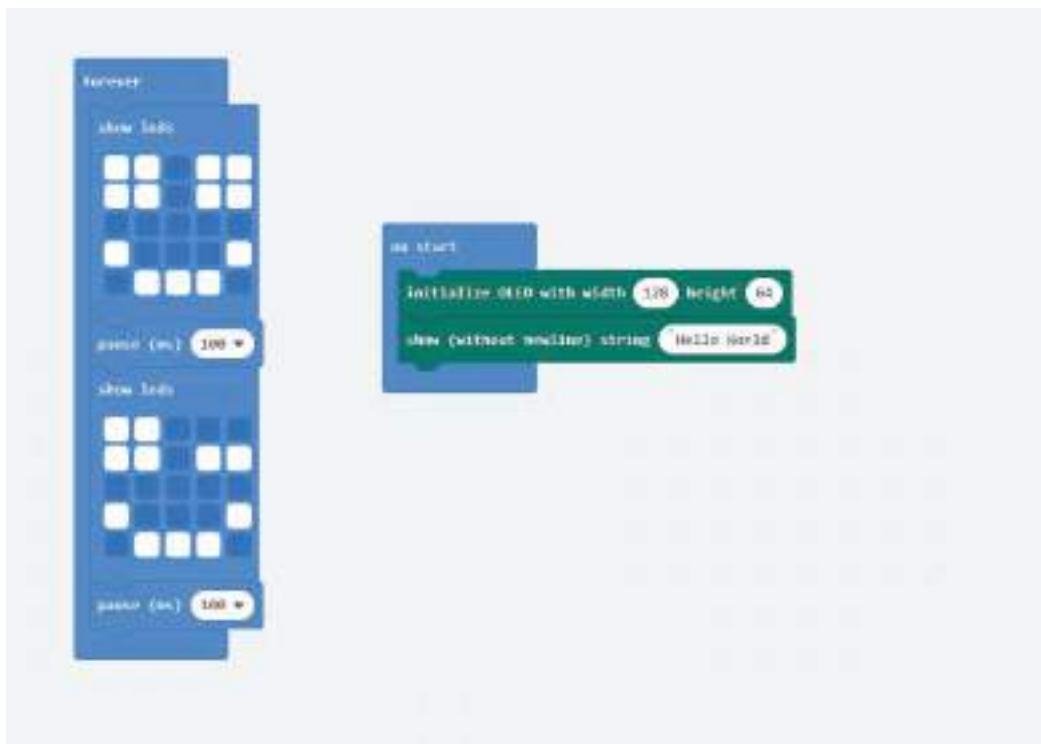
You can right-click on the above "show leds" block and copy it. After copying, click on the top right of two LEDs to turn them off.



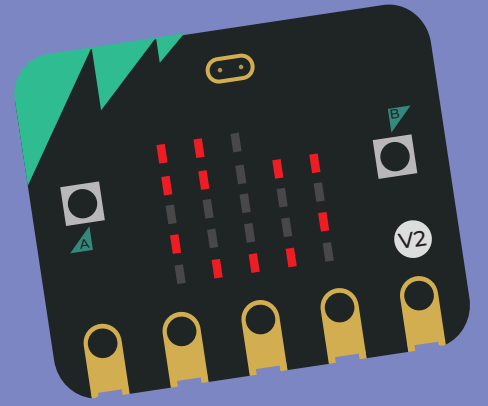
7. Let's use the "pause (ms)" block from the "Basic" blocks to make the code wait for 100ms.



8. The project code is ready!







# Action Reaction



# Action-Reaction

As Newton explained in his laws of motion, a reaction occurs against every action. Electronic systems receive commands from users and perform their tasks. Usually a keypad, touch screen or a button is used for this job. Electronic devices respond verbally, in writing or visually to inform the user that their task is over and what is going on during the task. In addition to informing the user of these reactions, it can help to understand where the fault may be in a possible malfunction.

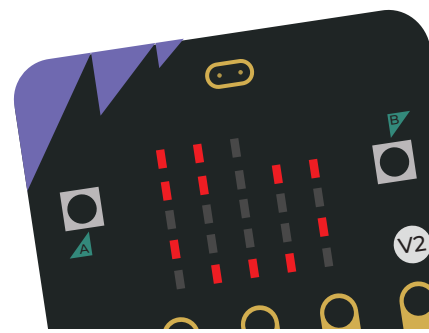
Buttons are circuit components through which we can provide input. Different types of buttons are used in electronic systems: toggle switches, push buttons and more. PicoBricks has a total of 3 push buttons, with 1 on the potentiometer and button module and 2 on the Micro:Bit. Push buttons function similarly to switches; they conduct current when pressed and do not conduct when released. PicoBricks has a total of 3 push buttons, with 1 push button on the potentiometer and button module, and 2 push buttons on the Micro:Bit. Push buttons operate like switches. Push buttons transmit current when pressed and do not transmit when released.

## Project Details:

Project Link: [https://makecode.microbit.org/\\_axa1HJFgX8g5](https://makecode.microbit.org/_axa1HJFgX8g5)

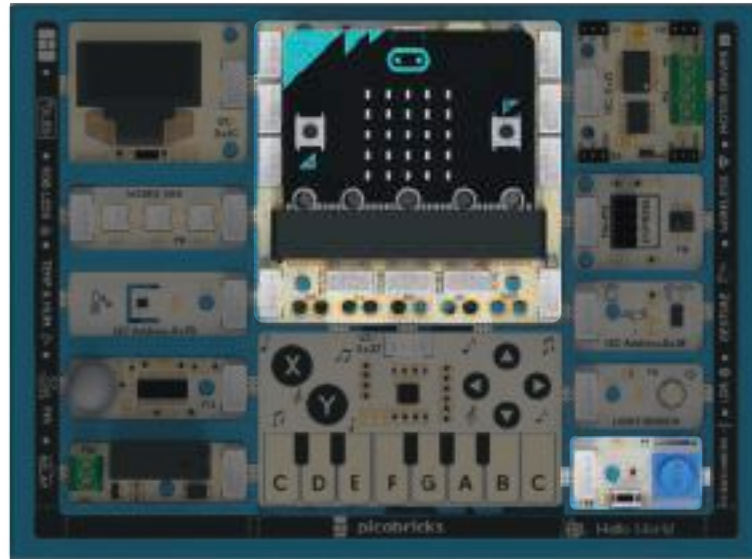


In the project, when the button on the potentiometer & button module is pressed, we will make the smiley face emoji we created on the Micro:Bit LED matrix blink.

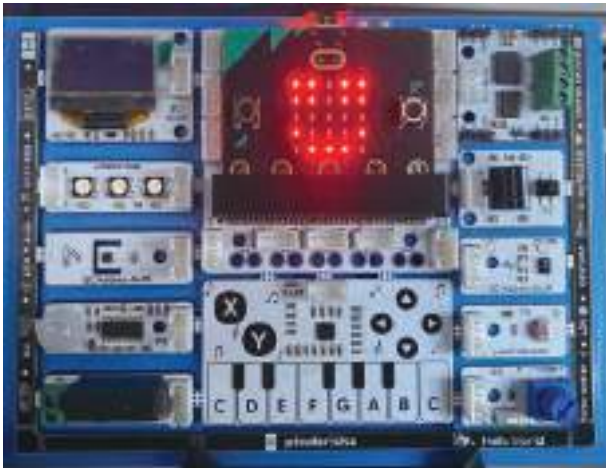


## ● Connection Diagram:

You can prepare this project without making any cable connections.



## ● Project Images:



## ● MakeCode Code of The Project:

If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

1. We will use the 'forever' block to ensure that the operations we want to perform in the project continuously repeat.

! You can delete "on start" block



2. We need to create a conditional structure to check whether the button is pressed in our project. For this, we will use the "if-else" block.



3. Let's create a conditional structure in the "if-else" block's condition area by using the "Button Read" block from PicoBricks extension blocks and the equality block from "logic" blocks, indicating "When PicoBricks Button is Pressed."



4. To make the Micro:Bit Matrix LEDs display a blinking emoji sign when the button is pressed, we will use the "show leds" block from the "Basic" blocks.

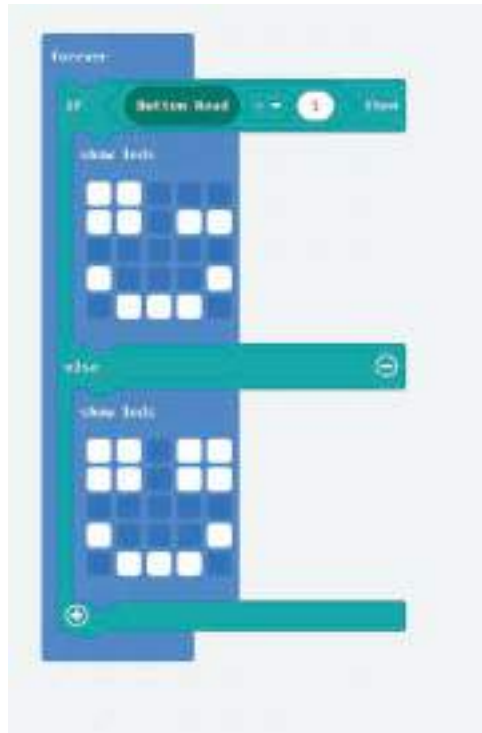
! You can create the desired shape by clicking on the square areas.



5. In times when the button is not pressed, drag the emoji we created with the "show leds" block into the "else" section to display a smiling face emoji on the screen.



6. The project code is ready!



A graphic featuring four overlapping, slightly tilted rectangular cards in red, dark blue, yellow, and green, set against a teal background. The text 'COLOR CARDS' is centered over the cards in a bold, white, sans-serif font with a black drop shadow.

# COLOR CARDS

# Color Cards Project

In these days, sensors that perceive the color of passing objects are commonly used in factories to alleviate workforce. For instance, different products moving on a production line can be directed to the correct conveyor belt thanks to color-sensing sensors. Many sectors employ more advanced versions of these sensors in their factories due to this feature. With the gesture module ([color and motion sensor](#)) we will use in this project, we can detect the colours of objects around PicoBricks.

The gesture module produces three numerical outputs as R (**RED**), G (**Green**), and B (**BLUE**) while detecting the colors of the object in front of it. When we use these outputs as the values of the RGB LED, a single color value is formed, and this color is the color of the object in front of the gesture sensor.



The environment light level, distance to the object, and the object's opacity can affect the value detected by the gesture module. The recommended distance should be around 5 cm on average.

## Project Details:

Project Link: <https://makecode.microbit.org/S02387-86095-09142-74813>



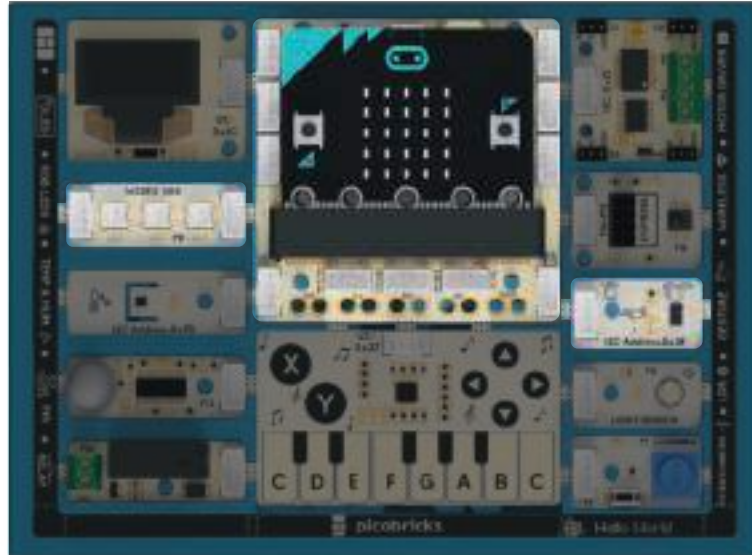
In this project, we will enable the gesture module to detect the colors of color cards we create by cutting colored cardboard, colored A4 paper, etc. This way, we will ensure that all 3 RGB LEDs in the RGB LED module light up in the same color. To do this, let's hold these colored papers in front of the gesture module.





## ● Connection Diagram:

You can prepare this project without making any cable connections.



## ● Project Images:



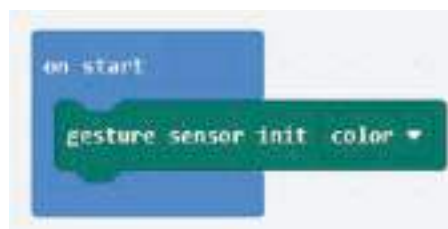
## ● MakeCode Code of The Project:

If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

1. When we first open the project page, we find 'forever' and 'on start' blocks. The 'forever' block ensures that the expressions dragged into it continuously run when the project is executed. On the other hand, 'On start' runs and stops the expressions inside it once when the project is executed.



2. After dragging the 'gesture sensor init' block from the gesture blocks into the 'on start' block, select 'color' from the options to initiate the color feature.



3. Drag the block from the RGB LED blocks that allows us to define the pin number of the RGB LED into the 'on start' block.





# Piano

# PicoBricks Piano Project

Advancements in electronics have led to the digitization of music instruments that were difficult and expensive to produce. Pianos are at the forefront of these instruments. Each key of digital pianos generates electrical signals at different frequencies, in this way, allowing them to play 88 different notes from their speakers. Factors such as the delay time of the keys on digital instruments, the quality of the speakers, and the resolution of the sound have emerged as quality-affecting elements. In electric guitars, vibrations on the strings are digitized instead of keys. In wind instruments, played notes can be converted into electrical signals and recorded through high-resolution microphones attached to the sound output. These developments in electronics have facilitated access to high-cost musical instruments and diversified music education.

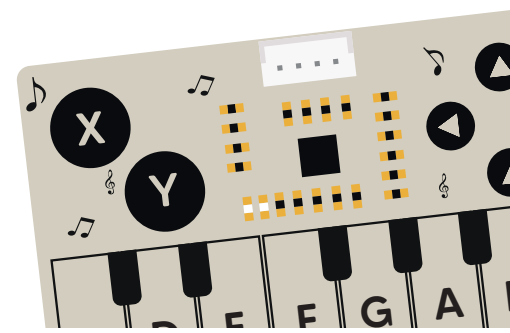
In this project, we will create a touch-sensitive piano by using the PicoBricks Touch & Piano module.

## Project Details:

Project Link: <https://makecode.microbit.org/S00485-65995-75778-92836>

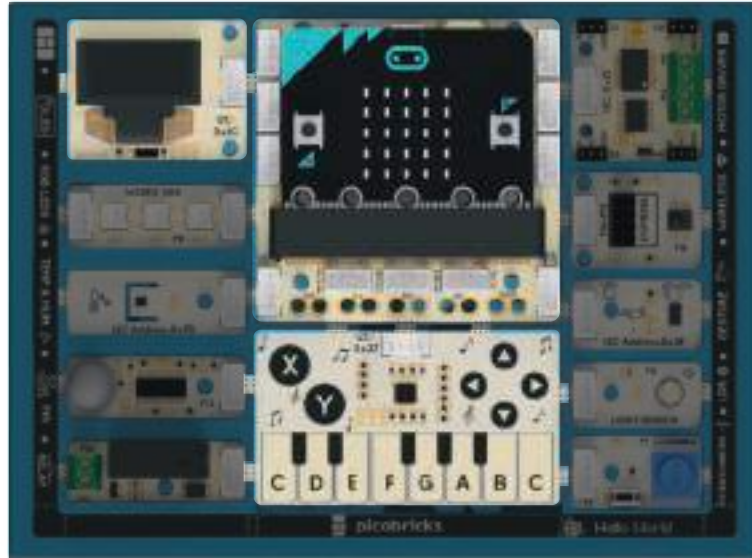


In this project, we will use the PicoBricks Touch & Piano module to play the desired note on the buzzer of the Micro:Bit based on the touch sensor. We will print the value of the pressed note on the Micro:Bit Matrix LEDs, and we will also display the texts "PicoBricks" and "Piano" on the PicoBricks OLED screen.

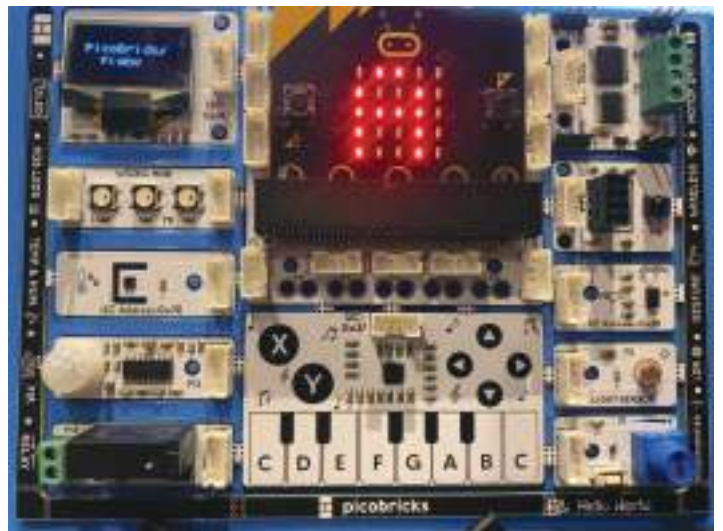
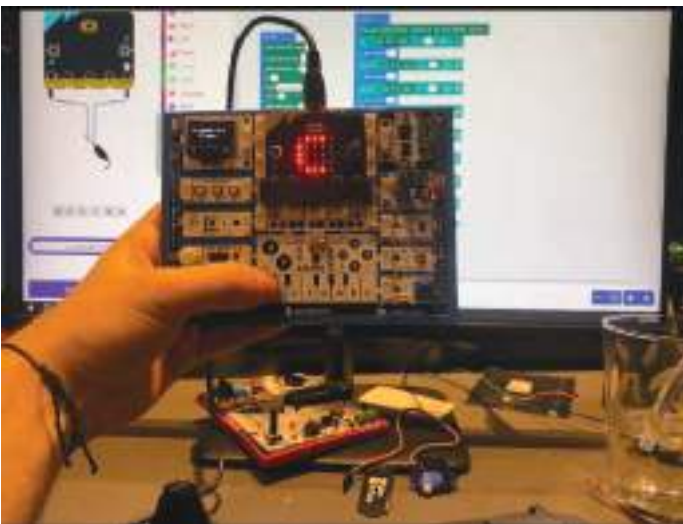


## ● Connection Diagram:

You can prepare this project without making any cable connections.



## ● Project Images:



## ● MakeCode Code of The Project:

If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

1. When the project is initiated, drag the code blocks that initialize the Touch & Piano module and the OLED screen module into the 'on start' block. Afterward, write the expressions 'PicoBricks' and 'Piano' in the X-Y positions determined on the OLED screen.



2. After touching the designated areas for notes in the Touch & Piano module, drag the “play piano buttons passive and tone buttons active” block to play the desired notes.



(If the “Volume” feature were enabled, touching the “up” button would **increase** the buzzer sound, and touching the “down” button would **decrease** the sound.)

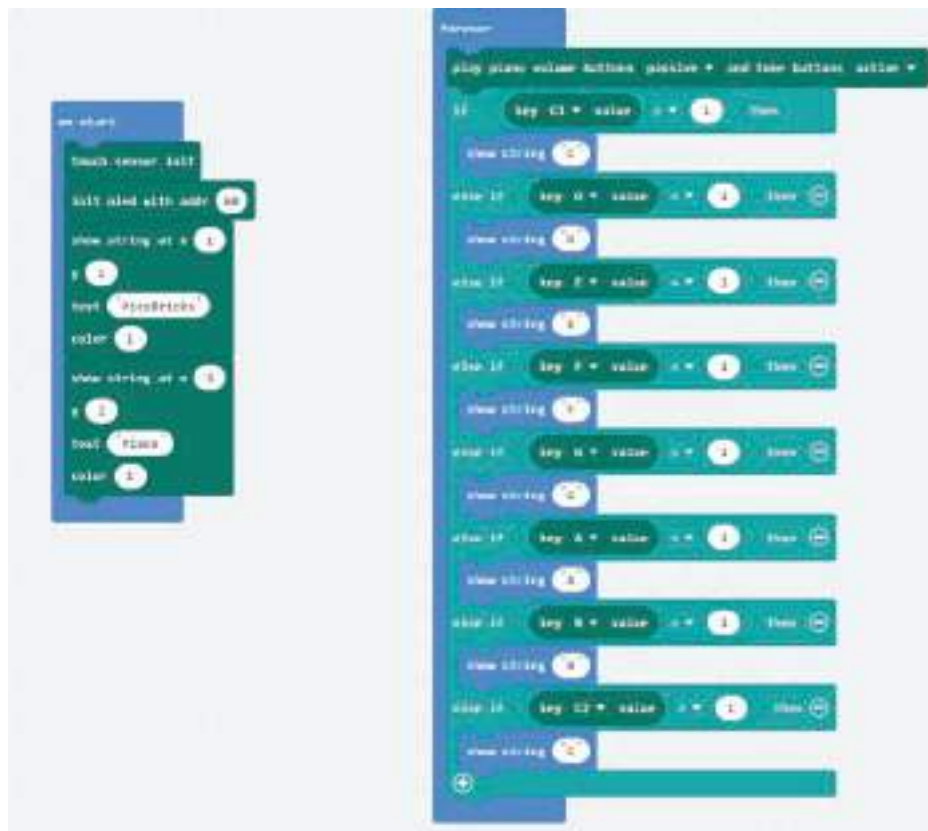


3. Create the necessary conditional structure to print the value of the touched note on the Micro:Bit Matrix LEDs after touching the designated areas for notes in the Touch & Piano module.



```
forever loop
  play piano volume button piano + and tone button action +
  if key 01 = value = 1 then
    show string C
  when 01 = value = 1 then
    show string C#
  when 02 = value = 1 then
    show string D
  when 03 = value = 1 then
    show string D#
  when 04 = value = 1 then
    show string E
  when 05 = value = 1 then
    show string F
  when 06 = value = 1 then
    show string F#
  when 07 = value = 1 then
    show string G
  when 08 = value = 1 then
    show string G#
  when 09 = value = 1 then
    show string A
  when 10 = value = 1 then
    show string A#
  when 11 = value = 1 then
    show string B
  show string C
```

4. The project code is ready!



```
when started
  touch sensor fall
  fall used with addr 00
  show string at 1
  1
  text Piano
  color 1
  show string at 1
  1
  text Piano
  color 1
  forever loop
    play piano volume button piano + and tone button action +
    if key 01 = value = 1 then
      show string C
    when 01 = value = 1 then
      show string C#
    when 02 = value = 1 then
      show string D
    when 03 = value = 1 then
      show string D#
    when 04 = value = 1 then
      show string E
    when 05 = value = 1 then
      show string F
    when 06 = value = 1 then
      show string F#
    when 07 = value = 1 then
      show string G
    when 08 = value = 1 then
      show string G#
    when 09 = value = 1 then
      show string A
    when 10 = value = 1 then
      show string A#
    when 11 = value = 1 then
      show string B
    show string C
```



# **RGB LED Control Panel**



# RGB LED Control Panel Project

In these days, RGB LEDs, used in various areas such as billboards, traffic lights, warning signs, etc., have a fundamental feature of being able to obtain intermediate colors by taking values between 0-255 for red, green, and blue colours. In fact, with RGB LEDs, we can create animations by changing colors on a panel we create.

There are three RGB LEDs on PicoBricks. By using the MakeCode editor, we can obtain various color outputs by setting the desired RGB values for each of these RGB LEDs. In this project, we will examine in detail how RGB LEDs work by changing color values with the potentiometer module and buttons.

## Project Details:

Project Link: <https://makecode.microbit.org/S99470-50205-44854-71948>

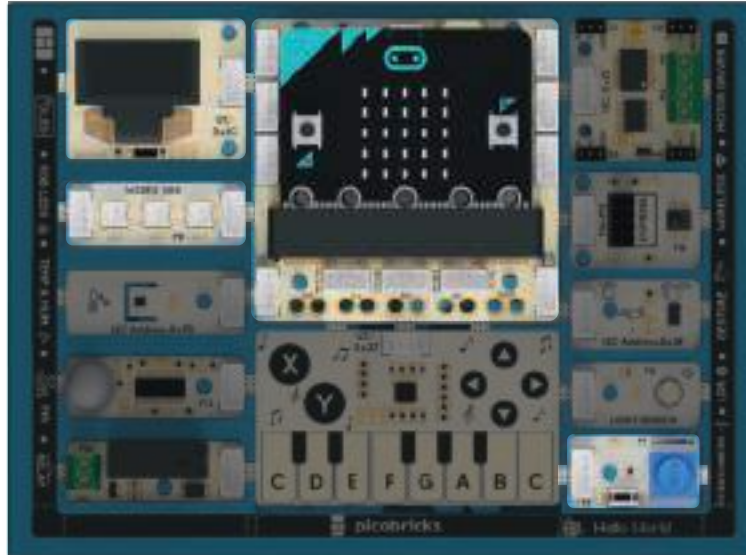


Using the PicoBricks potentiometer module, we will adjust color values between 0-255. By pressing the button on the PicoBricks Potentiometer & Button module, we will set the color value to red; by pressing Micro:Bit A button, we will set it to green, and by pressing Micro:Bit B button, we will set it to blue. This way, we will observe the instant changes in the values of the three RGB LEDs on the PicoBricks RGB LED module. At the same time, the color values will be updated on the PicoBricks OLED screen each time we press a button.



## ● Connection Diagram:

You can prepare this project without making any cable connections.



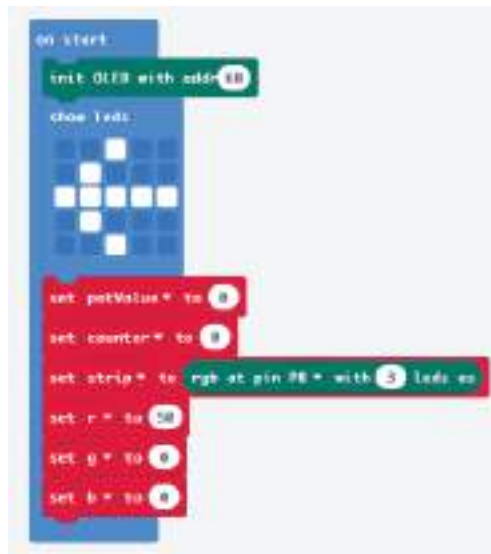
## ● Project Images:



## ● MakeCode Code of The Project:

If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

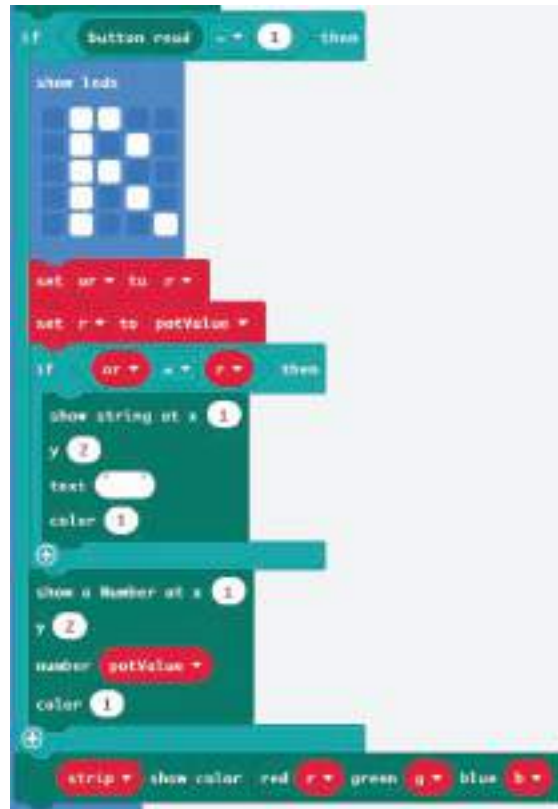
1. Let's create an arrow sign using the Micro:Bit Matrix LEDs to represent the OLED screen and RGB LEDs on the left side. Then, let's define the "potValue," "r," "g," and "b" variables needed for the project.



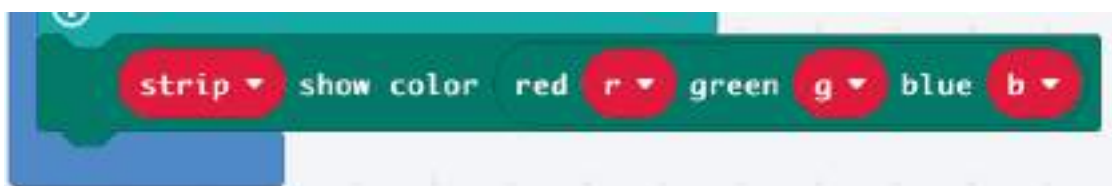
2. Let's assign the value of the "potValue" variable by using the "pot read" block, limiting it to the range of 0-255. If the "potValue" is different from its previous value, let's print it to the specified x and y coordinates.



3. "When the PicoBricks button is pressed, if the 'r' variable is different from its previous value, it writes the 'r' value to the specified x and y coordinates and creates the letter 'R' in the 'Micro:Bit' matrices."



4. Each time the loop iterates, RGB values are updated instantly based on the 'r,' 'g,' and 'b' variables.



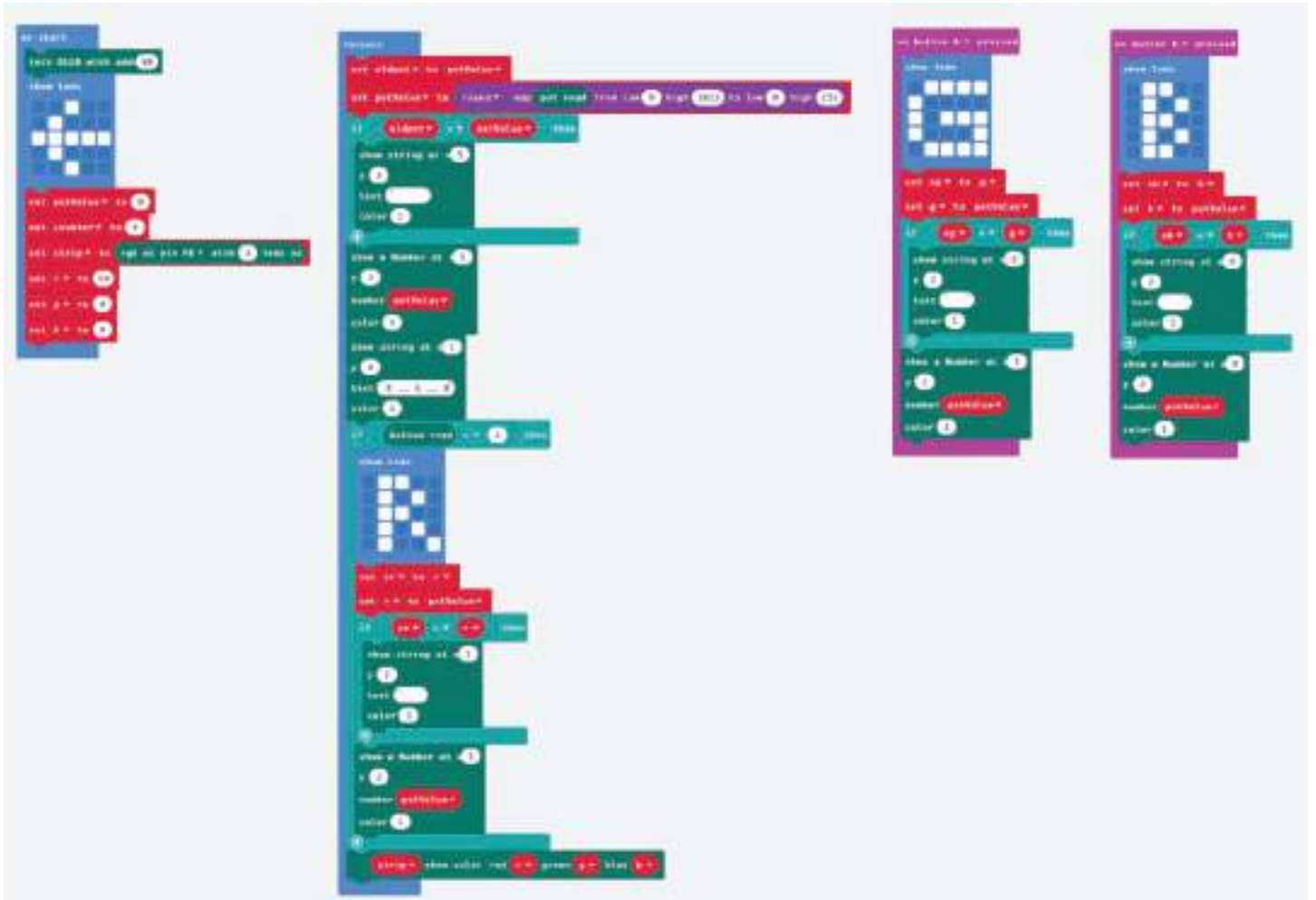
5. When the Micro:Bit A button is pressed, if the 'g' variable is different from its previous value, it writes the 'g' value to the specified x and y coordinates and creates the letter 'G' in the 'Micro:Bit' matrices.

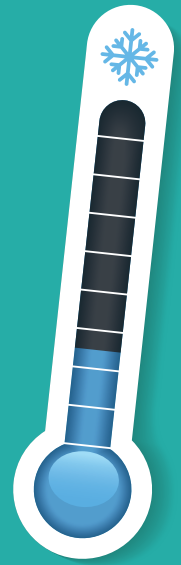


6. "When the Micro:Bit A button is pressed, if the 'b' variable is different from its previous value, it writes the 'b' value to the specified x and y coordinates and creates the letter 'B' in the 'Micro:Bit' matrices."



## 7. The project code is ready!





# Thermometer



# Thermometer Project

Sensors are the sensory organs of electronic systems. To perceive, we use our skin, eyes for seeing, ears for hearing, tongue for tasting, and nose for smelling. Picobricks already has many sensory organs (sensors), and new ones can also be added. By using sensors such as humidity, temperature, light, and many others, you can interact with the environment. PicoBricks can measure ambient temperature without the need for any other environmental components.

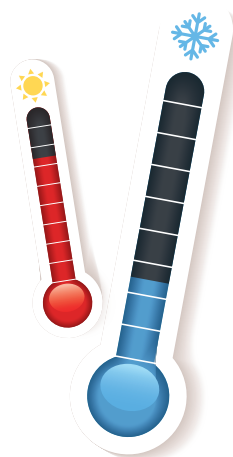
Ambient temperature is used in situations where continuous monitoring of temperature changes is required, such as in greenhouses, incubators, and environments used for transporting medications. If you are going to perform a task related to temperature changes in your projects, you should know how to measure ambient temperature. In this project, you will prepare a thermometer with PicoBricks that displays ambient temperature on the OLED screen. Using the PicoBricks potentiometer module, you can instantly change the displayed temperature value on the OLED screen between Fahrenheit and Celsius.

## Project Details:

Project Link: <https://makecode.microbit.org/S17960-12849-27544-24508>



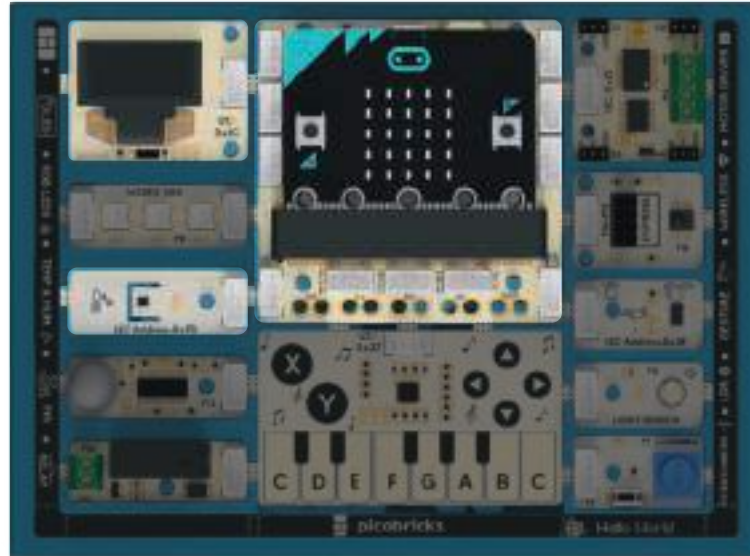
Thanks to the PicoBricks Temperature & Humidity module, we will display the temperature and humidity values detected from the environment on the OLED screen by using the Potentiometer module, either in Celsius or Fahrenheit.





## ● Connection Diagram:

You can prepare this project without making any cable connections.



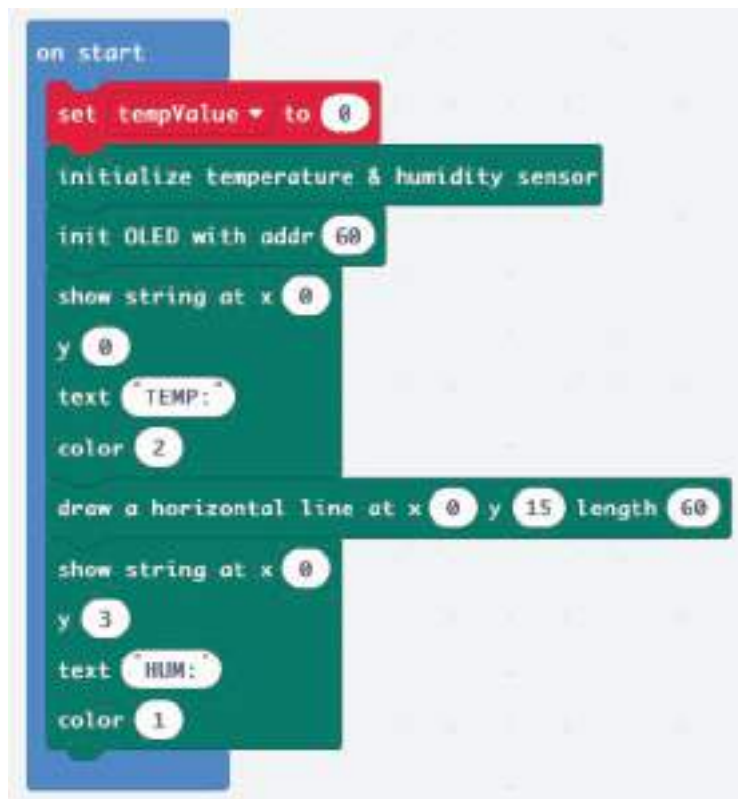
## ● Project Images:



## ● MakeCode Code of The Project:

If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

1. At the beginning of the project, set the variable named 'tempValue' to '0'. Initialize the Temp&Hum module and write the expressions 'TEMP' and 'HUM' to the specified x and y positions on the screen.



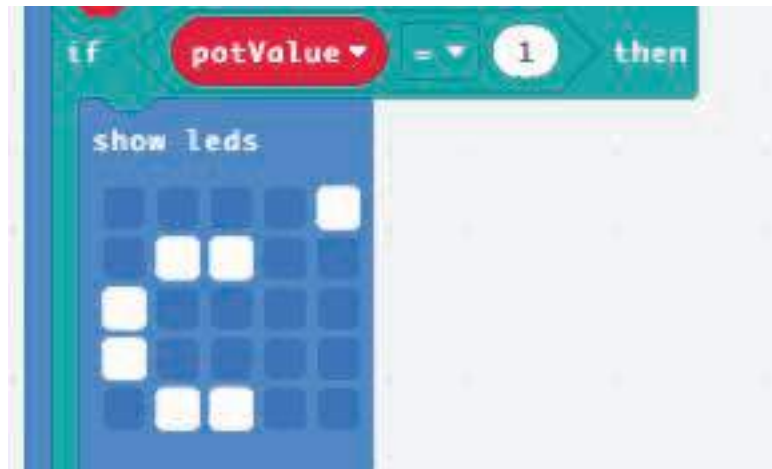
```
on start
  set tempValue to 0
  initialize temperature & humidity sensor
  init OLED with addr 60
  show string at x 0
  y 0
  text TEMP:
  color 2
  draw a horizontal line at x 0 y 15 length 60
  show string at x 0
  y 3
  text HUM:
  color 1
```

2. Let's create a variable named 'potValue' and limit its value from 0 to 1023 to a range of 1 to 2.



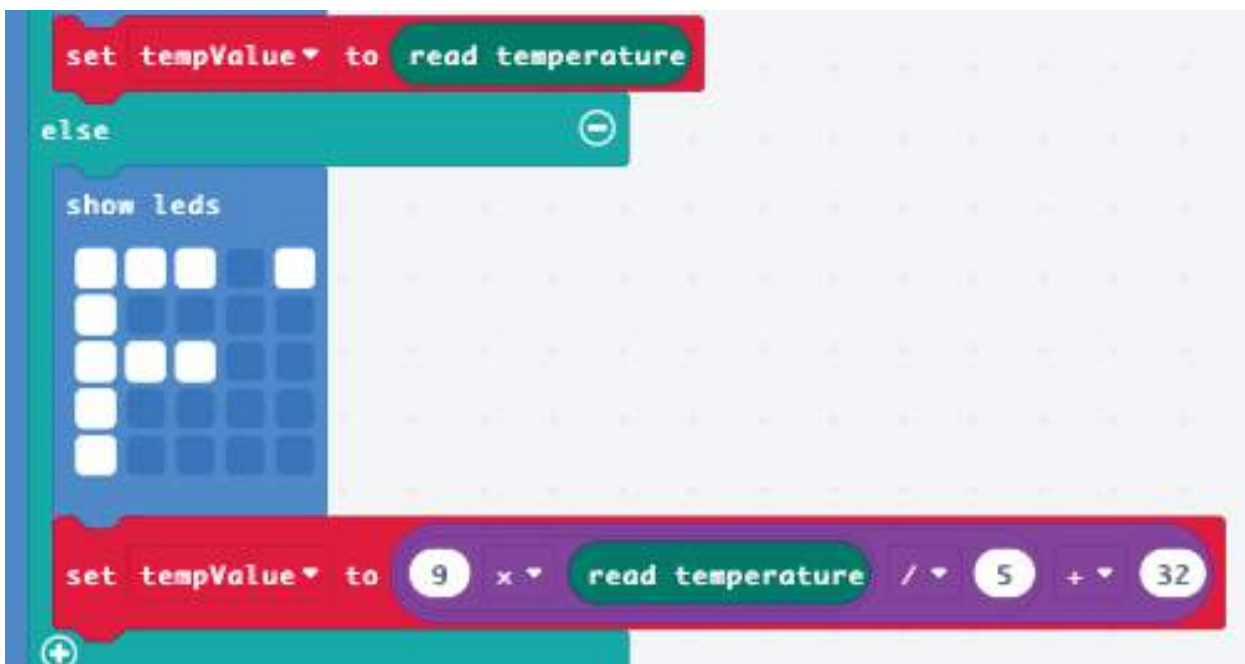
```
forever
  set potValue to round map pot read from low 0 high 1023 to low 1 high 2
```

3. If potValue equals 1, create the °C symbol on the Micro:Bit LED Matrix, and set the 'tempValue' variable equal to the value of the 'read temperature' block.



4. Fahrenheit = (Celcius\* 9/5) +32

If potValue is equal to 2, create the °F symbol on the Micro:Bit LED Matrix. Convert the 'read temperature' value to Fahrenheit using the formula and assign it to the 'tempValue' variable.



5. Print the 'tempValue' variable and the value of the 'read humidity' block as integers on the OLED screen at the specified X and Y positions.

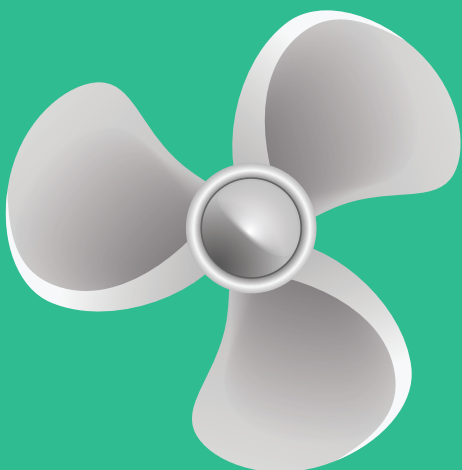
The image shows two separate code blocks from a block-based programming environment. The first block is for displaying temperature: it sets the x-coordinate to 5 and the y-coordinate to 0, then displays the value of the 'tempValue' variable, rounded to an integer, in color 1. The second block is for displaying humidity: it sets the x-coordinate to 5 and the y-coordinate to 3, then displays the value of the 'read humidity' block, rounded to an integer, in color 1.

6. The project code is ready!

The image shows the complete project code for an OLED display. It starts with an 'on start' block that initializes the temperature and humidity sensors, sets the I2C address to 0x48, and displays the strings 'TEMP' and 'HUM' at y-coordinates 0 and 2 respectively. A horizontal line is drawn at y=15 with a length of 40. The main loop consists of several blocks: it reads the temperature from the sensor (range 0 to 1023), checks if the 'tempValue' is greater than 1, displays the temperature in color 1, and then reads the humidity from the sensor (range 0 to 1023). It calculates the average of the two temperature readings and displays it at x=5, y=0. Finally, it displays the rounded humidity value at x=5, y=3.



# Smart Cooler



# Smart Cooler Project

To cool off during the summer months and warm up in the winter months, air conditioners are used. Air conditioners adjust the heating and cooling degrees based on the temperature of the environment. Ovens, on the other hand, strive to reach and maintain the temperature value set by the user while cooking. Both of these electronic devices use special temperature sensors to control the temperature. Additionally, in greenhouses, temperature and humidity are measured together. To maintain a balance at the desired level, a fan is used to provide air circulation.

You can measure temperature and humidity separately with PicoBricks and interact with the environment using these measurements. In this project, we will prepare a cooling system with PicoBricks that automatically adjusts fan speed based on temperature. This way, you will learn about the operation of a DC motor system and how to adjust motor speed.

## Project Details:

In this project, we will adjust the speed of the fan connected to the motor driver based on the value obtained from the temperature and humidity module. The fan connected to the motor driver will operate when the temperature exceeds a certain value. If the temperature falls below a certain value, the fan will stop.

## Connection Diagram:

You can prepare this project without making any cable connections.



# Smart Cooler Project

To cool off during the summer months and warm up in the winter months, air conditioners are used. Air conditioners adjust the heating and cooling degrees based on the temperature of the environment. Ovens, on the other hand, strive to reach and maintain the temperature value set by the user while cooking. Both of these electronic devices use special temperature sensors to control the temperature. Additionally, in greenhouses, temperature and humidity are measured together. To maintain a balance at the desired level, a fan is used to provide air circulation.

You can measure temperature and humidity separately with PicoBricks and interact with the environment using these measurements. In this project, we will prepare a cooling system with PicoBricks that automatically adjusts fan speed based on temperature. This way, you will learn about the operation of a DC motor system and how to adjust motor speed.

## Project Details:

Project Link: <https://makecode.microbit.org/S85586-67410-33813-60113>

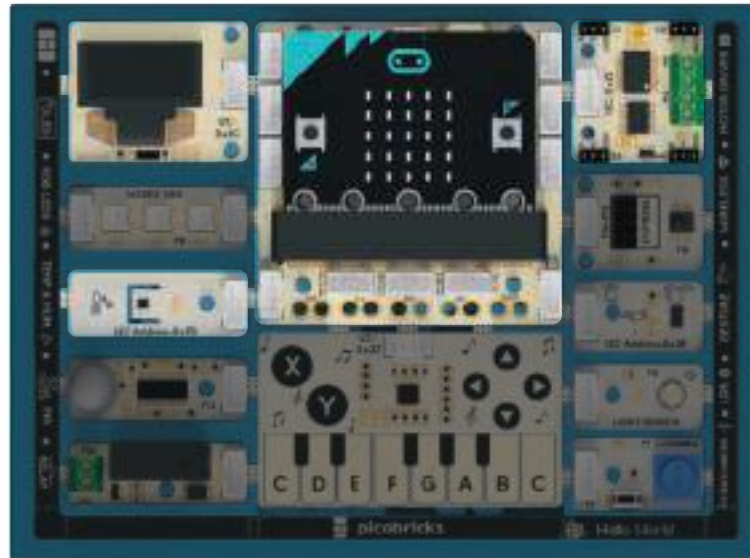


In this project, we will adjust the speed of the fan connected to the motor driver based on the value obtained from the temperature and humidity module. The fan connected to the motor driver will operate when the temperature exceeds a certain value. If the temperature falls below a certain value, the fan will stop.

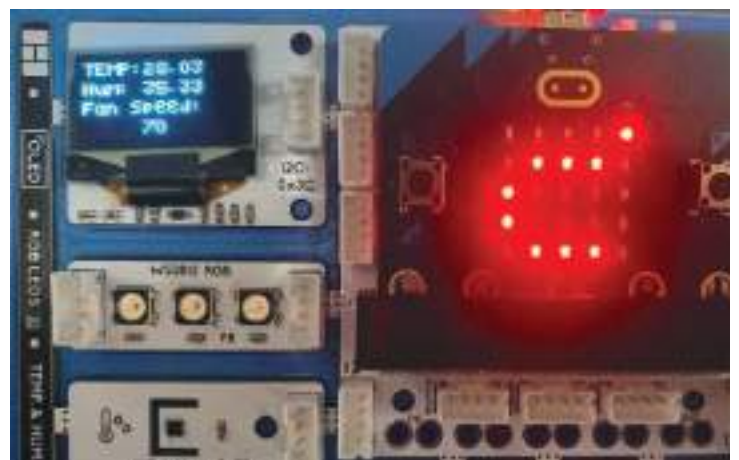


## ● Connection Diagram:

You can prepare this project without making any cable connections.



## ● Project Images:





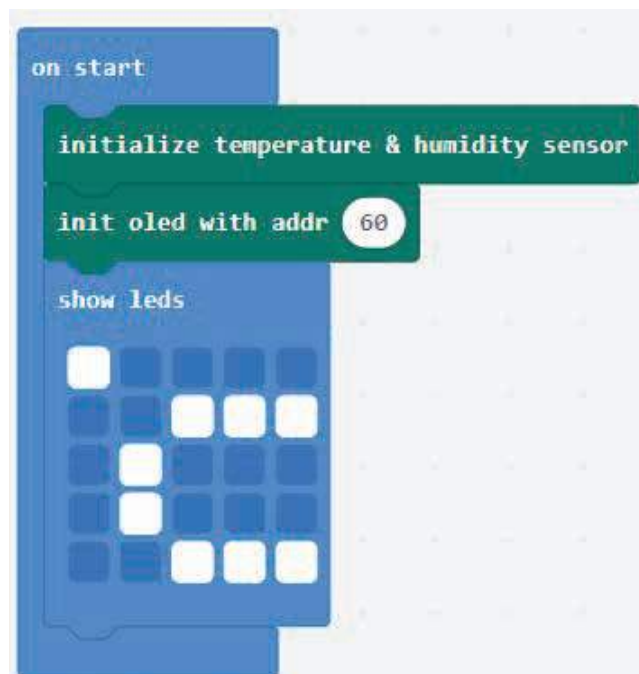
## ● MakeCode Code of The Project:

If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

1. Let's drag the blocks that initialize the Temperature & Humidity and OLED display modules from PicoBricks into the 'on start' block.



2. When the project is started, let's create the Celsius (°C) symbol on the Micro:Bit Matrix LEDs.



3. Let's create the operations in our project that we want to repeat continuously within the 'forever' block.

a) Let's print the temperature and humidity values to the specified location on the OLED screen.

b) By defining a variable named 'motorSpeed,' let's adjust the Temperature value, which ranges from 0 to 40, to a scale of 0 to 100.

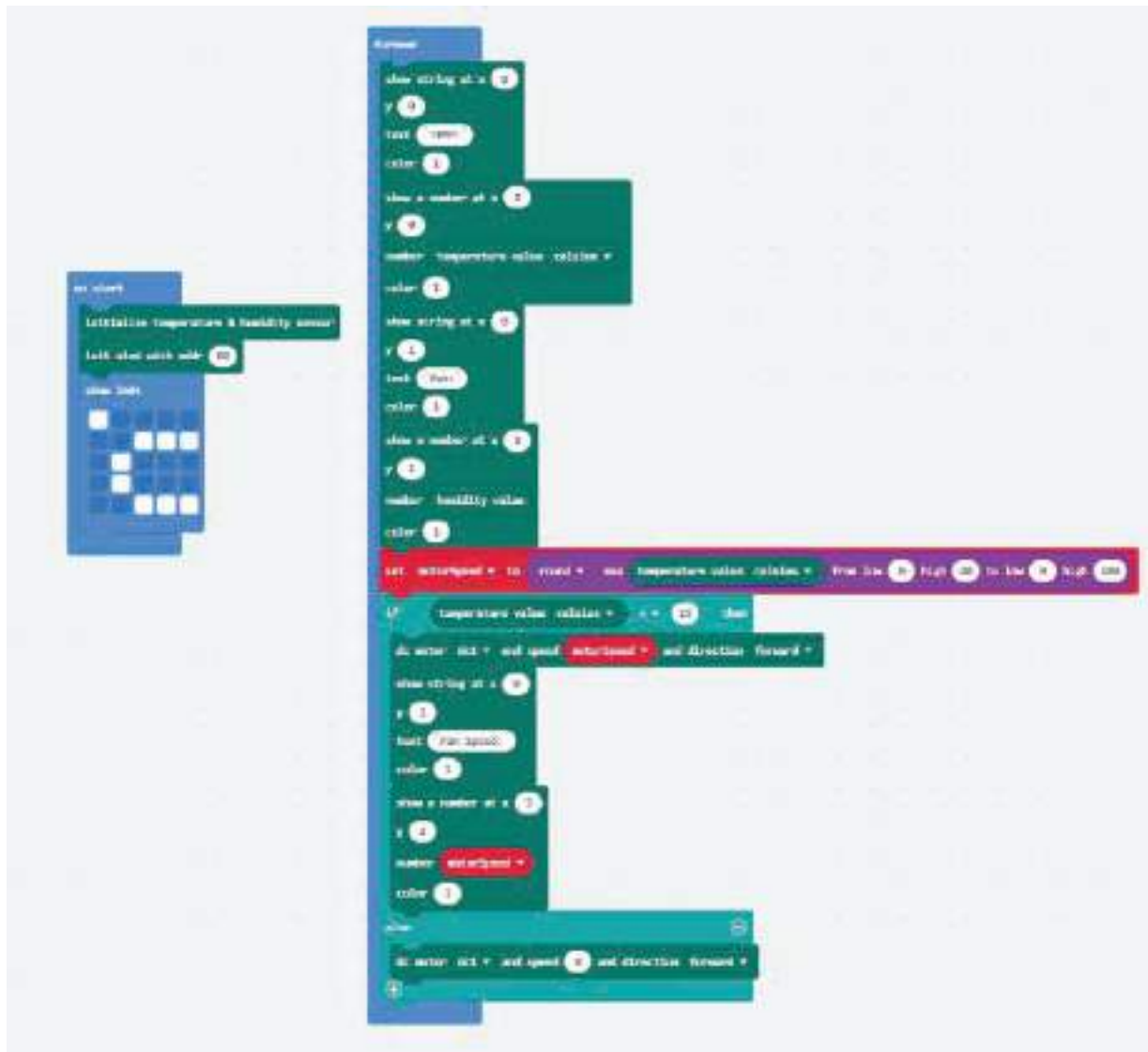


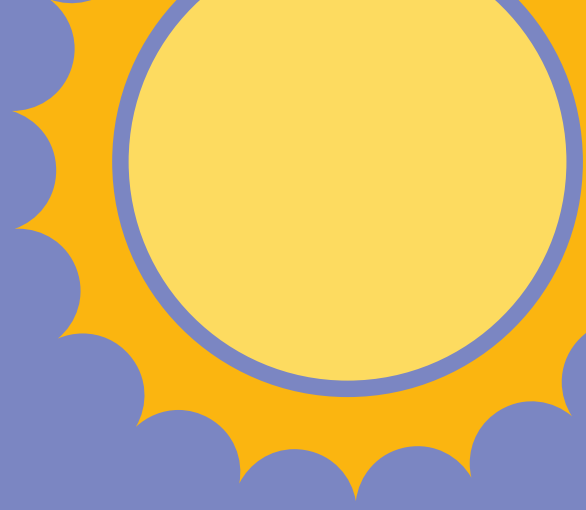
c) If the temperature value is greater than 25, run the DC motor connected to the motor driver at the speed specified by the 'motorSpeed' variable. Display the 'motorSpeed' value on the OLED screen at the specified location.

d) Otherwise, stop the motor.



#### 4. The project code is ready!





# Night and Day



# Night and Day Project

How about playing the Day and Night game electronically, a game often played in schools? In this game, when the teacher says "night," we bend our heads, and when the teacher says "day," we raise our heads. It's a game that involves using your attention and reflexes. In this project, we will use a 0.96" 128x64 pixel I2C OLED screen. Since OLED screens can be used as an artificial light source, you can reflect the characters on the screen onto any desired plane using a lens or a mirror. Systems that can project information, road, and traffic data onto smart glasses and car windows can be created using OLED screens. Light sensors are devices that can measure the light levels in the environment, also known as photodiodes. The electrical conductivity of the sensor changes when exposed to light. By coding, we will control the light sensor and develop electronic systems affected by the amount of light.

## Project Details:

Project Link: <https://makecode.microbit.org/S68795-47745-24924-74027>

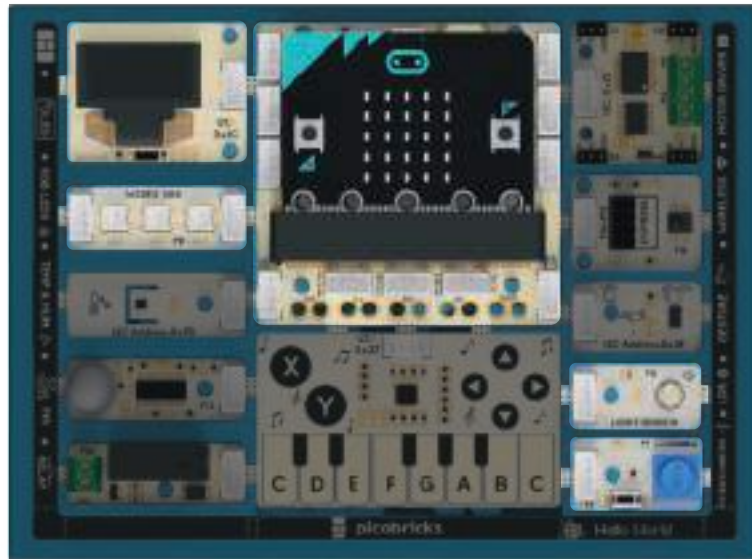


First, we will provide the player to press the button to start the game. Then, on PicoBricks' OLED screen, we will randomly display the expressions NIGHT and DAY for 2 seconds each. If the word NIGHT is displayed on the OLED screen, the player must cover the LDR sensor with their hand within 2 seconds. If the word DAY is displayed on the OLED screen, the player must remove their hand from the LDR sensor within 2 seconds. Each correct response from the player will earn them 10 points and create a checkmark (✓) icon on the Micro:Bit Matrix LEDs. When the player gives an incorrect response, the game will end, and the screen will display a written message indicating the end of the game along with a cross ( X ) icon on the Matrix LEDs. The buzzer will play a sound in a different tone, and the OLED screen will show the score information. If the player achieves a total of 10 correct responses and earns 100 points, the message "Congrats!!!" will be displayed on the OLED screen at the designated positions.



## ● Connection Diagram:

You can prepare this project without making any cable connections.



## ● Project Images:



## ● MakeCode Code of The Project:

If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

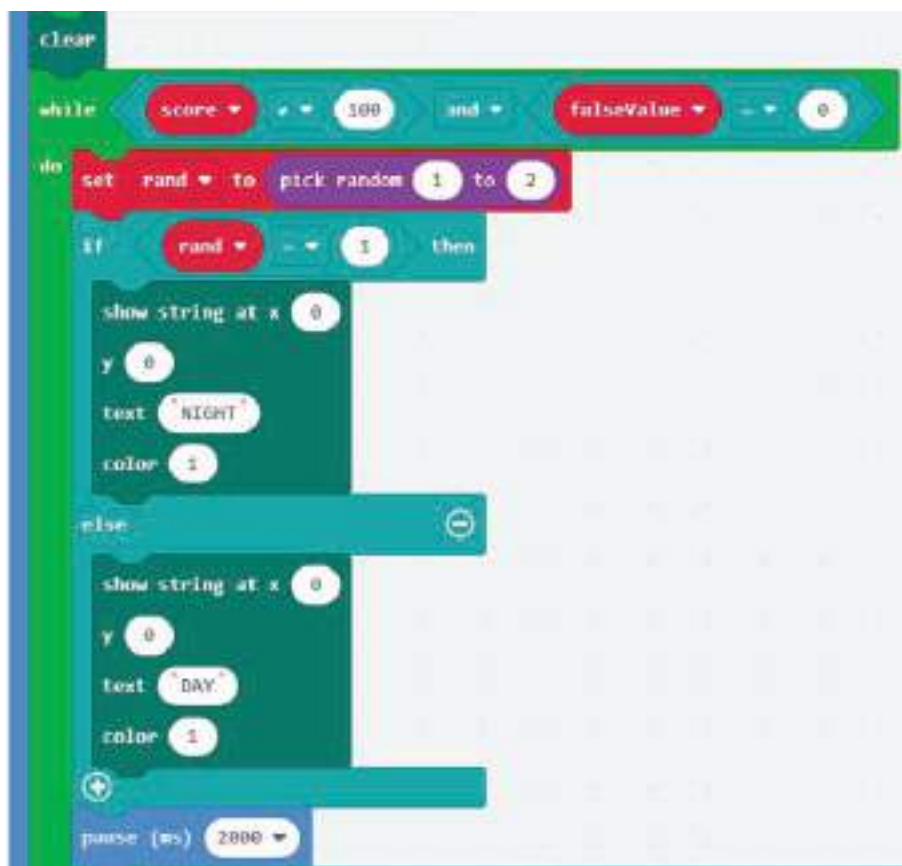
1. In this project, we will perform all the operations within the "on start" block. Since we will be using the OLED screen, let's start our project by dragging the block that initializes the OLED screen from the PicoBricks extension.
2. To keep track of the player's score, let's define a variable called "score" and set its initial value to "0."
3. To stop the game when the player makes a wrong move, let's define a variable called "falseValue" and set its initial value to "0."



4. Let's print the expressions "Press BUTTON" and "to START!" on the OLED screen at the specified x and y coordinates until the PicoBricks button is pressed.



5. After pressing the button, let's clear the OLED screen.
6. Let's create a loop structure that runs until the score reaches 100 or the user makes a wrong move.
7. Assign randomly the numbers 1 or 2 to the variable named "rand" each time the loop iterates.
8. If the value of the 'rand' variable is 1, display 'NIGHT' on the OLED screen at the specified X - Y coordinates. If its value is 2, display 'DAY'.
9. Let's wait for 2000 milliseconds (2 seconds) for the user to respond.





**10.** If the player covers the LDR sensor (**LDR Value > 400**) and **"NIGHT"** is displayed on the OLED screen (**rand=1**), let's create a checkmark icon (✓) on the Matrix LEDs and increase the **"score"** value by 10.

**11.** If the player do not cover the LDR sensor (**LDR Value < 400**) and **"NIGHT"** is displayed on the OLED screen (**rand=1**), let's create an X icon on the Matrix LEDs. Set the **"falseValue"** to 1 and terminate the loop.

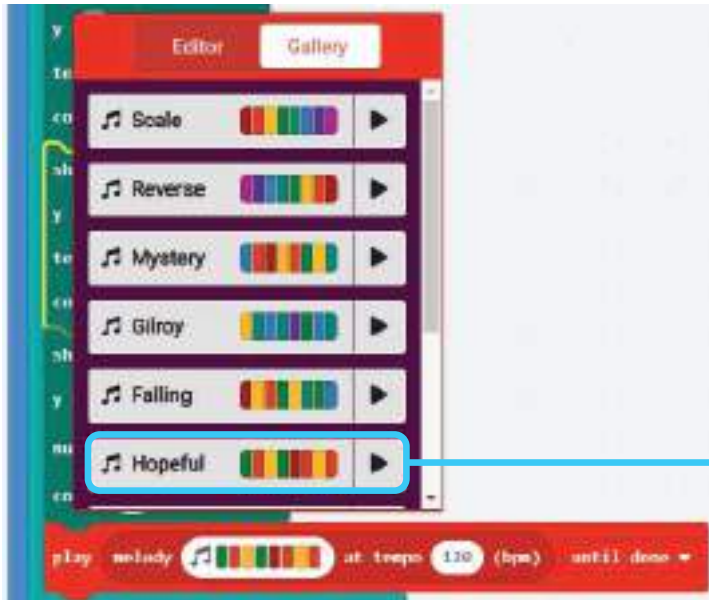
**12.** If the player covers the LDR sensor with her/his finger (**LDR Value > 400**) and **"DAY"** is displayed on the OLED screen (**rand=2**), let's create an X icon on the Matrix LEDs. Set the **"falseValue"** to 1 and terminate the loop.

**13.** If none of the above conditions are met ( if **"DAY"** is displayed on the OLED screen and the player has not covered the LDR sensor with their finger), let's create a checkmark icon (✓) on the Matrix LEDs and increase the value of the **"score"** by 10.

**14.** After our conditional structures are executed, let's clear the OLED screen.



15. After the loop terminates, if the value of the "Score" variable is 100, let's print the values of "Congrats!!!," "Top Score:", and the "score" variable at the specified X-Y coordinates on the OLED screen. Additionally, let's play the melody named "hopeful" from the buzzer.



Choose the "Hopeful" melody.



16. After the loop terminates, if the value of the "falseValue" variable is 1, let's print the values of "Game Over," "Score:", and the "score" variable at the specified X-Y coordinates on the OLED screen.



1

2

Fizz

4

Buzz

7

# Fizz - Buzz Game

Fizz

11

8

Fizz

Buzz

# Fizz - Buzz Game Project

There are some games that every programmer spends time on. Fizz Buzz is one of them. Every programmer who has made some progress in a programming language has created the algorithm for the Fizz-Buzz game, aiming to master that language by writing this game. The Fizz-Buzz game is frequently preferred in programming language education because its algorithm includes both conditional statements and loop structures, helping to grasp the steps of computational thinking. At the same time, while playing this game, we improve our quick decision-making and mathematical thinking skills.

Thanks to PicoBricks, we can code this game by using electronic components and experience it physically.

## Project Details:

Project Link: <https://makecode.microbit.org/S33540-45790-68635-11193>

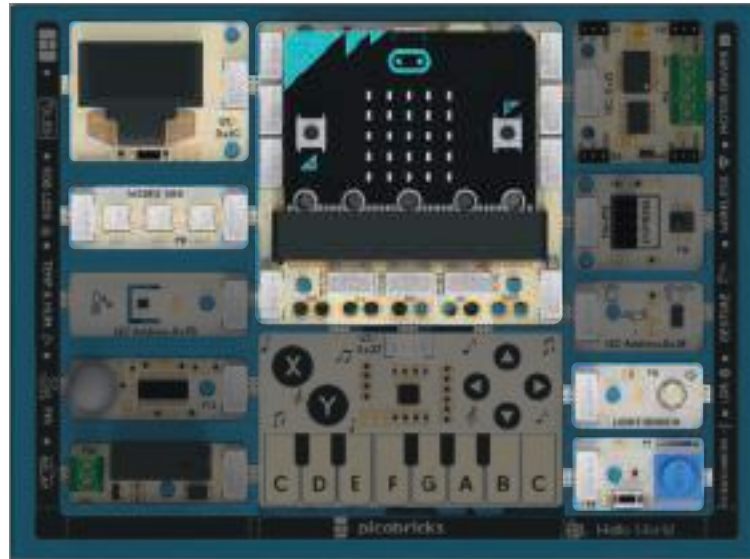


In this project, we will create the Fizz-Buzz game algorithm and code using PicoBricks along with the button, RGB LED, and OLED screen module with Micro:Bit. The Fizz-Buzz game is played by counting numbers from 1 to 100. Starting from 1, when a number that is a multiple of 3 is reached, "Fizz" is said. When a number that is a multiple of 5 is reached, "Buzz" is said. When a number is a multiple of both 3 and 5, "Fizz-Buzz" is said instead of the number.

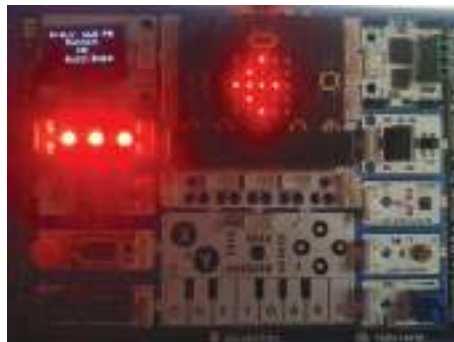
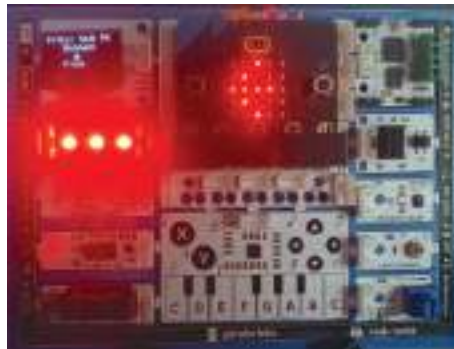


## ● Connection Diagram:

You can prepare this project without making any cable connections.



## ● Project Images:



## ● MakeCode Code of The Project:

If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

1. When our project is run, let's drag the necessary blocks into the "on start" block to initialize the OLED screen and RGB LED, display a heart icon on the Micro:Bit Matrix LEDs, and write "Press the A Button to start Fizz-Buzz" at specified positions on the OLED screen.

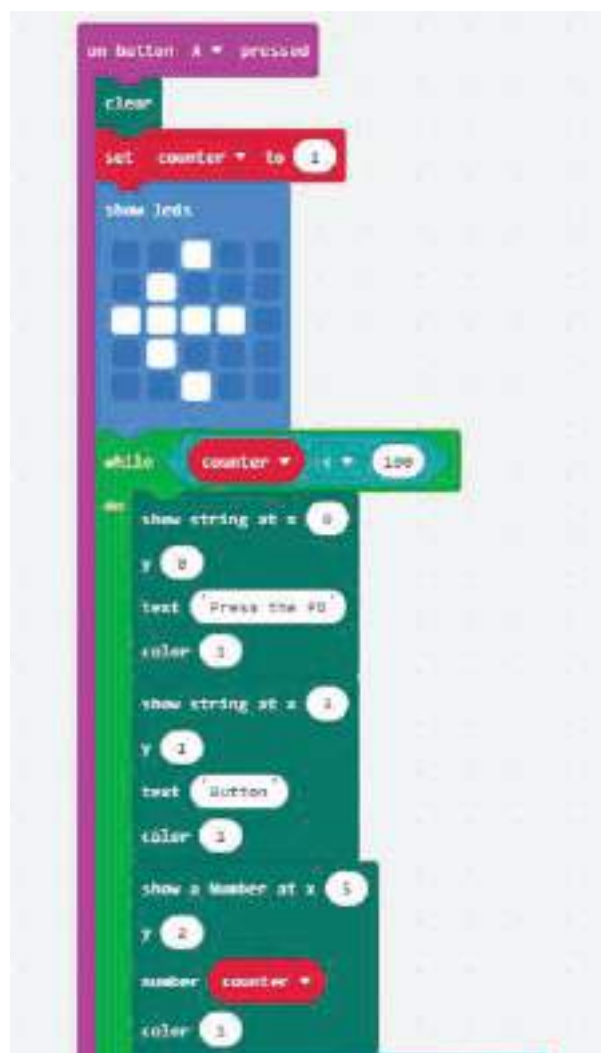


2. After our project is started, let's take the "on button A pressed" block from the "Input" blocks so that our game begins when the A button on the Micro:Bit is pressed. We will create the necessary algorithm for our game within this block.

a) In our game, since we will be displaying the counter and fizz, buzz, and fizz-buzz values on the OLED screen, let's first clear the screen when the button is pressed. Create a variable named "counter" and display a left arrow icon on the Matrix LEDs.

b) Since the actions we want to perform in the game will repeat for the numbers from 1 to 100, let's create a loop that repeats 100 times using the "while" block from the "Loop" blocks.

c) Let's create the expressions that need to be displayed on the OLED screen within this loop.





3. To make the counter in our project repeat each time we press the PicoBricks button, let's create the following conditional structure. Each time the button is pressed, a B note will play from the buzzer, the RGB LEDs will turn off, and the undesired expressions on the screen will be cleared.



4. When a number that is a multiple of 3 is encountered, "Fizz" will be displayed on the screen. When a number that is a multiple of 5 is encountered, "Buzz" will be displayed on the screen. When a number that is a multiple of both 3 and 5 is encountered, "Fizz-Buzz" will be displayed on the screen. Let's create the necessary conditional structure for this.



## 5. The project code is ready!

```
on start
  left GLED with 1
  set strip to rgb at pi#* with 3 leds as
  show icon
  show string a 2
  y 8
  text Press the
  color 1
  show string a 2
  y 1
  text A Button to
  color 1
  show string a 3
  y 2
  text start
  color 1
  show string a 2
  y 3
  text Fizz-Buzz
  color 1

on button pressed
  clear
  set counter to 1
  show leds
  while counter <= 100
    show string a 8
    y 8
    text Press the
    color 1
    show string a 3
    y 1
    text Button
    color 1
    show a Number a 5
    y 2
    number counter
    color 1
    if button state = 1 then
      changecounter by 1
      play tone High for 1 beat until done
      show string a 1
      y 3
      text 
      color 1
      strip show colorblack
    if remainder of counter / 3 = 0 then
      show string a 1
      y 3
      text Fizz
      color 1
      strip show colored*
    if remainder of counter / 5 = 0 then
      show string a 1
      y 3
      text Buzz
      color 1
      strip show colorblue*
    if remainder of counter / 15 = 0 then
      show string a 1
      y 3
      text Fizz-Buzz
      color 1
      strip show colorpurple*
```

# Depth Meter



# Depth Meter Project

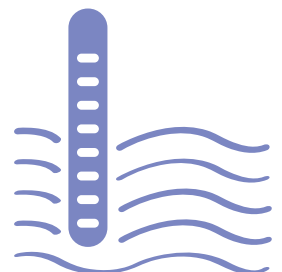
Sometimes, we use depth-measuring machines to measure the quantity of a beverage or mixtures of liquid materials poured into a glass. The fundamental variable that needs to be known for these machines to measure depth is the depth value measured when the container is empty. After defining this information to the measurement devices, the device performs the measurement process by using various sensors such as ultrasonic distance sensor, IR sensor, etc.

## Project Details:

Project Link: <https://makecode.microbit.org/S66925-54085-67601-89663>

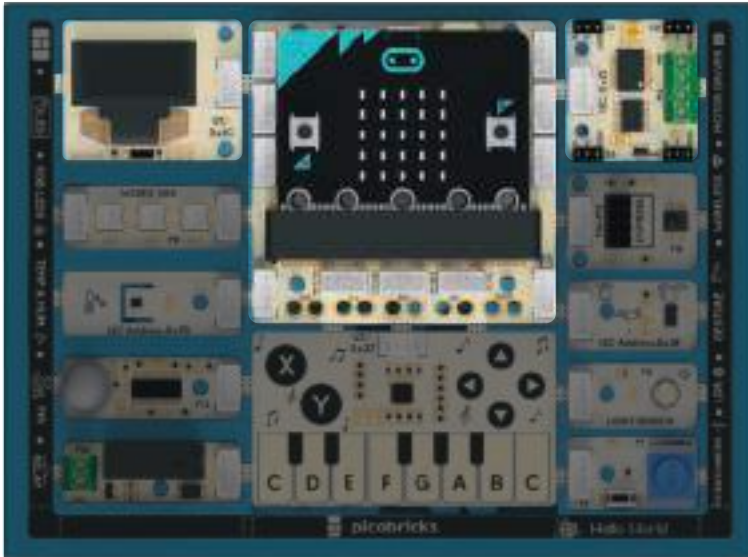


In this project, we will control the water pump connected to the motor driver based on the value measured by the ultrasonic distance sensor we connect to PicoBricks. We will transfer the desired amount of liquid from the container filled with liquid to the empty one. To determine the depth of the glass, we will use the potentiometer & button module.



## ● Connection Diagram:

You can prepare this project without making any cable connections.



## ● Project Images:



## ● MakeCode Code of The Project:

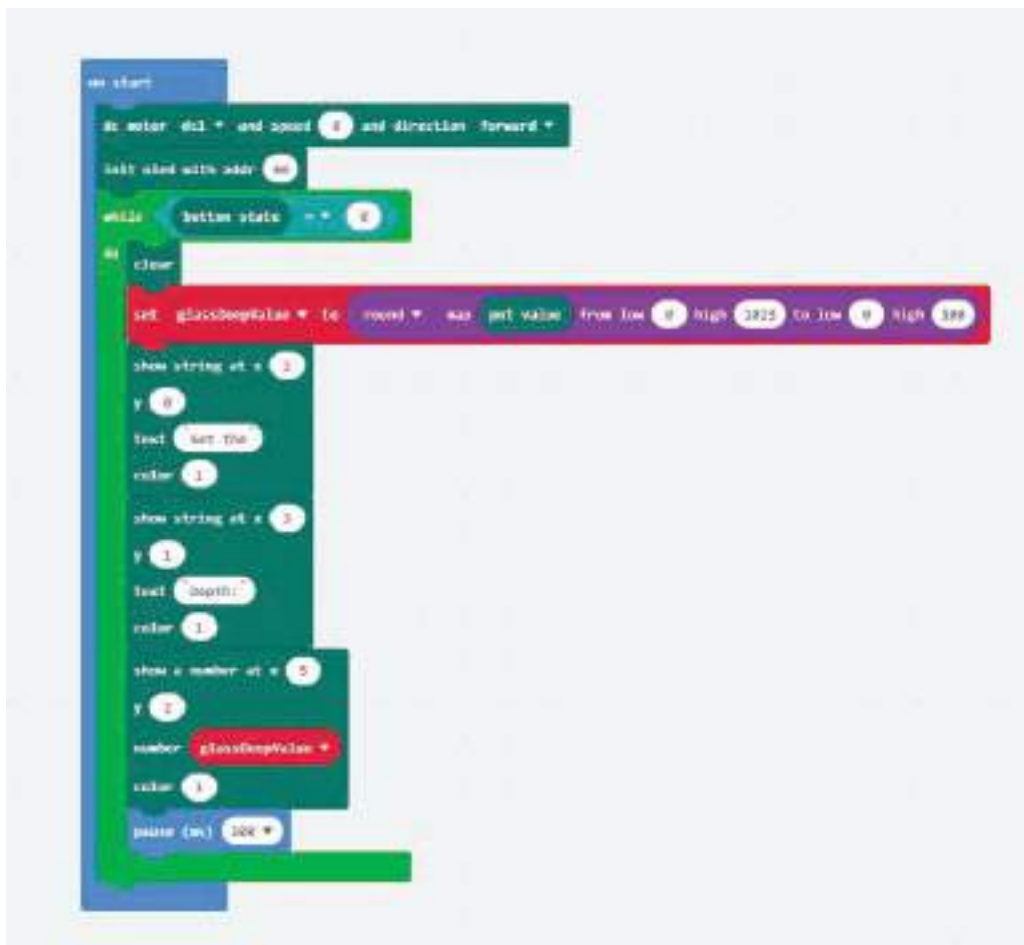
If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

1. When the project is started, inside the "on start" block:

a) Let's drag the code block that resets the water pump if the water pump is running due to the previous measurement.

b) Let's drag the code block required to initialize the OLED display.

c) Let's determine the depth of the empty container we want to fill with liquid by using the potentiometer until the button is pressed. For this process, let's create a conditional loop structure using the "while" block and assign the value of the potentiometer to a variable named "glassDeepValue".



2. Let's define all the operations that will take place after the button is pressed within the "forever" block.

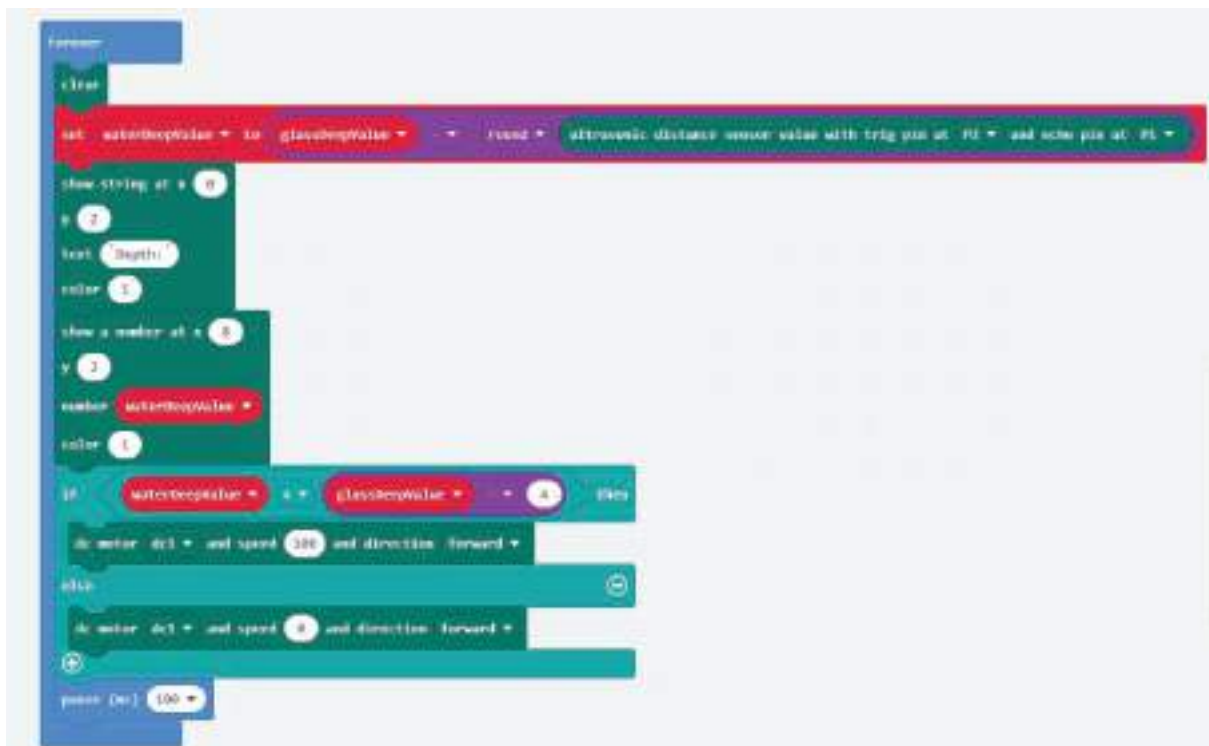
a) First of all, let's clear the OLED screen.

b) Let's determine the value of the variable named "waterDeepValue" using the formula required to measure the amount of liquid inside the container. To determine the depth of the liquid in a container filled with liquid, subtract the value measured by the distance sensor from the depth measured when the container is empty (glassDeepValue).

c) Let's print the measured value (**waterDeepValue**) **"Depth:"** on the OLED screen at the specified x-y coordinates.

d) Let's create the code blocks that enable the transfer of water from the full container to the empty container by running the water pump connected to the motor driver until a sufficient amount of liquid is reached in the container to be filled.

e) To continue this loop every 100ms, drag the "pause (ms) 100" block to the end.







A ● —

F ● ● — ●

K — ● —

P ● — — ●

# Morse Code Cryptography

B — ● ● ●

S ● ● ●

C — ● — ●

# Morse Code Cryptography Project

People sometimes utilize some passwords to protect their physical belongings or written/visual content. The diversity of symbols used in passwords contributes to the strength of the password. Similarly, not using easily guessable personal data in passwords will enhance the strength of your password.

Cryptography refers to the processes that render readable data incomprehensible to unauthorized individuals.

In Morse Code, there are distinct long and short signals corresponding to each letter. Each character of the text to be encrypted is encoded using the short and long signals in Morse Code. When these signals are combined as a whole and deciphered, the encrypted text is revealed. The long and short signals in Morse Code can be created using sound or light. The most well-known example of this is the SOS distress signal. With a flashlight or similar light source, a call for help can be made by sequentially emitting three long, three short, and three long signals. This is because in Morse Code, the letter "s" is represented by (...) three short signals, and the letter "o" is represented by (---) three long signals

S                      O                      S  
( ● ● ● )            ( --- )                ( ● ● ● )

## Morse Code Alphabet:

A ● —	B — ● ● ●	C — ● — ●	D — ● ●	E ●
F ● ● — ●	G — — ●	H ● ● ● ●	I ● ●	J ● — — —
K — ● —	L ● — ● ●	M — —	N — ●	O — — —
P ● — — ●	Q — — ● —	R ● — ●	S ● ● ●	T —
U ● ● —	V ● ● ● —	W ● — —	X — ● ● —	Y — ● — —
Z — — ● ●				

## Project Details:

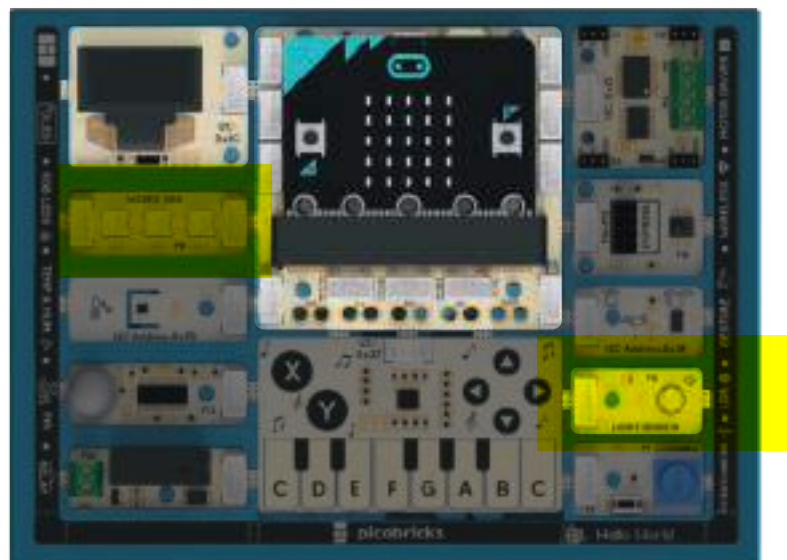
Project Link: [https://makecode.microbit.org/\\_aWX1Pdgm4Abi](https://makecode.microbit.org/_aWX1Pdgm4Abi)



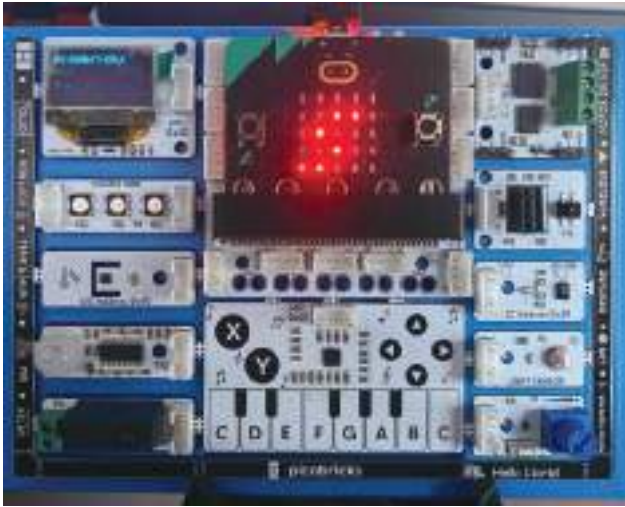
In this project, we will encrypt the specified text by using Morse Code within the code, utilizing the PicoBricks RGB LED module. Each character of the encrypted text will be displayed on the Micro:Bit matrix LED, and its Morse Code equivalent will be shown on the PicoBricks OLED screen.

## Connection Diagram:

You can prepare this project without making any cable connections.



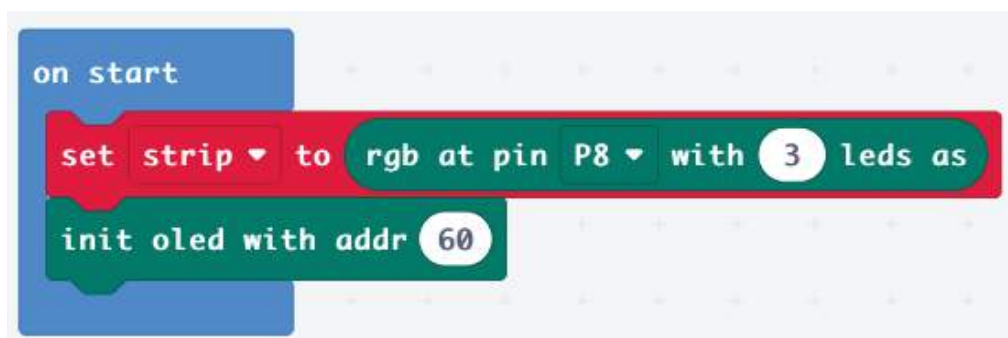
## Project Images:



## MakeCode Code of The Project:

If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

1. Since we will be using the RGB LED module and OLED screen module, let's drag the blocks that define the pins of these modules and initialize these modules into the 'on start' block.



2. Let's define Morse Code into a list called "morse".



3. Firstly, let's define the 'alphabet' list by entering the alphabet and then define the numbers.



4. Let's create the following code blocks to display the specified text encrypted with RGB LED when we press the A button on the Micro:Bit:

```

on button A pressed
  set passwordText to picobricks
  for index from 0 to length of passwordText - 1
    clear
    show string at 0
    y 0
    text passwordText
    color 1
    show string char from passwordText at index
    show string at 0
    y 1
    text word* get value at alphabet* find index of char from passwordText at index*
    color 1
    set j* to 0
    for i from 0 to length of word* get value at alphabet* find index of char from passwordText at index* - 1
      show string at 0
      y 2
      text char from word* get value at alphabet* find index of char from passwordText at index* at j*
      color 1
      if char from word* get value at alphabet* find index of char from passwordText at index* at j* == 0 then
        strip* show color white*
        pause (ms 500)
      else
        strip* show color white*
        pause (ms 500)
    strip* clear
    pause (ms 100)
  
```



Enter the text you want to encrypt into the variable named "passwordText."

```

on button A pressed
  set passwordText to 'picobricks'
  
```

# Car Parking System



# Car Parking System Project

Today, buildings such as hospitals, schools, business centers, etc., often have open or closed parking lots where a large number of people enter and exit. The main reason for the construction of these parking lots is the significant increase in automotive usage in cities. Barrier systems are installed at the entrances of these parking lots to control access. While people were assigned to control these barrier systems in the past, nowadays, with the advancement of sensor technologies, automatic access systems are used. Vehicles are detected using various sensors, the barriers are raised using motor systems, and vehicle passage is allowed.

## Project Details:

Project Link: <https://makecode.microbit.org/S49144-51181-40312-54356>



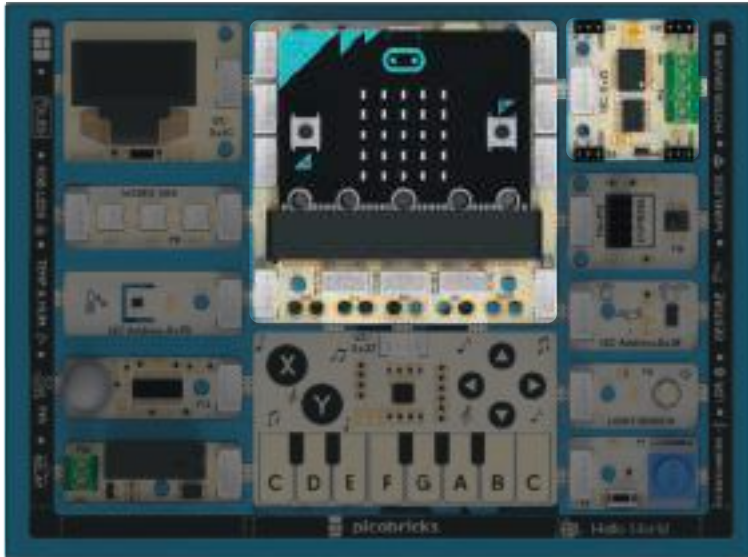
In this project, we will use the ultrasonic distance sensor connected to PicoBricks to create a barrier system by using waste bins found in our home, depending on the value detected by the sensor. By moving the servo motor connected to the prepared barrier system to the desired angle and we will allow vehicle passage. A checkmark icon (✓) will appear on the Micro:Bit Matrix LED when permission is granted for passage, and a cross (X) icon will appear when permission is denied.





## ● Connection Diagram:

You can prepare this project without making any cable connections.



HC-SR04



Servo Motor

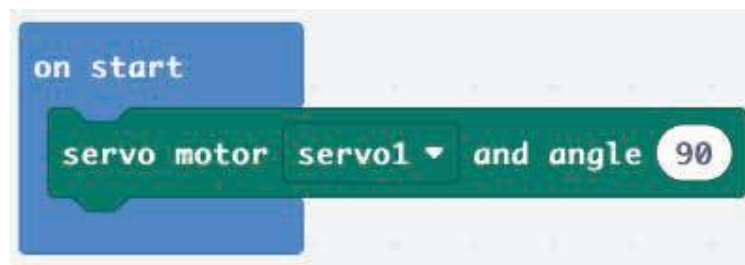
## ● Project Images:



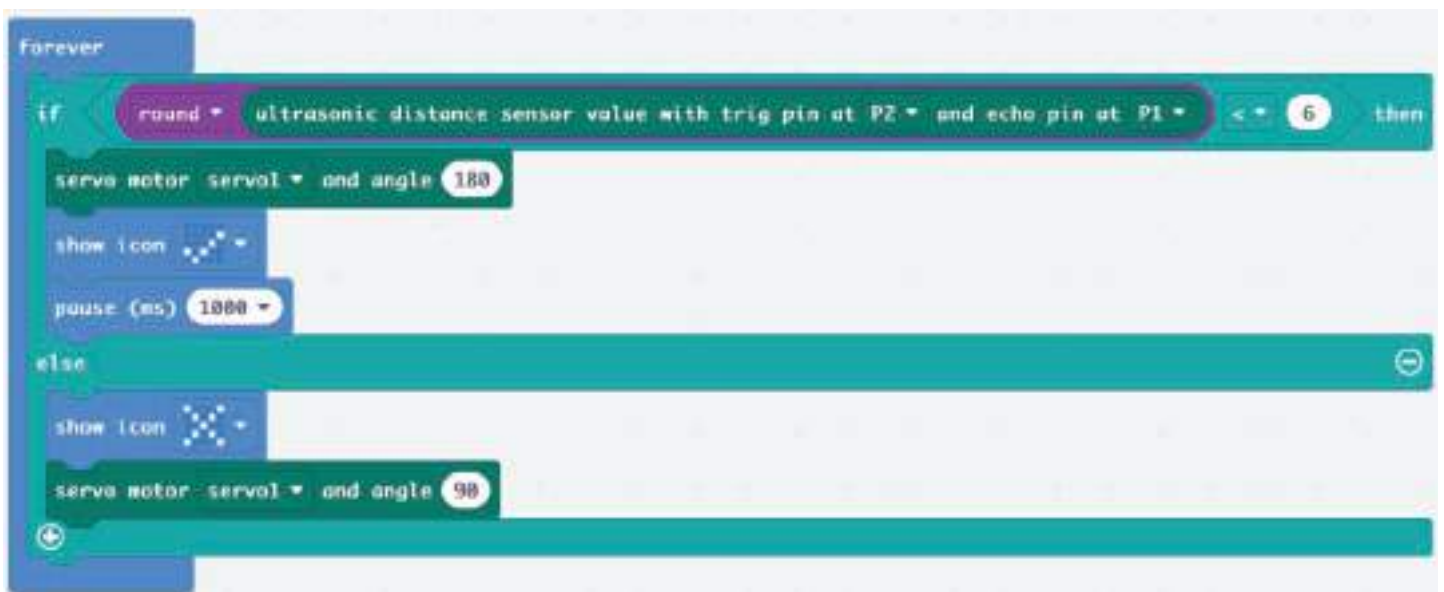
## ● MakeCode Code of The Project:

If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

1. When the project starts, drag the code block that brings the servo motor to its initial position (90 degrees angle) into the "on start" block.



2. If the value of the distance sensor is less than 6 (indicating that a vehicle has arrived at the parking lot), bring the servo motor connected to the motor driver to 180-degree angle (lift the barrier) and wait for 1 second for the vehicle to pass (You can adjust this waiting time according to your preference). Otherwise, move the servo motor to a 90-degree angle and keep the barrier closed.





# Table Lamp

# Table Lamp Project

Many of us, for reasons such as studying, reading books, preparing reports, etc., prefer to illuminate only our desks instead of turning on all the lights in the room at night. The desk lamps we use at home typically use RGB LEDs as light sources. This is because RGB LEDs can emit light in desired color tones. Exposure to certain lights for extended periods can negatively impact our eye health. In such cases, quick transitions between desired colors can be achieved using the color values of RGB LEDs, ranging from 0 to 255. Additionally, RGB LEDs can operate without requiring large power sources. Therefore, desk lamps can be easily illuminated with their own power sources.

In this project, we will add various features to a table lamp in our home by using PicoBricks modules.

(You can use any table lamp in your home.)

## Project Details:

Project Link: <https://makecode.microbit.org/S11132-37840-21881-33617>

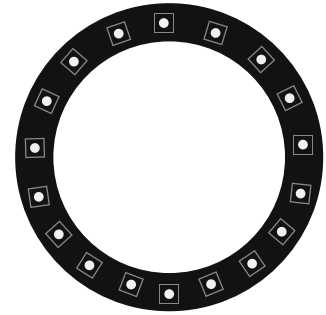
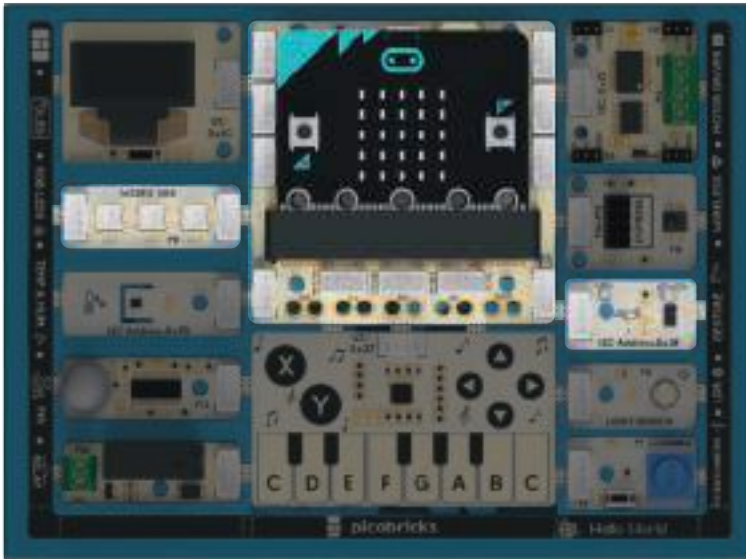


In this project, we will illuminate a desk lamp by using PicoBricks RGB LED and Gesture modules based on the directional movements we make with our hands. After placing the Gesture and RGB LED modules, when we move our hand to the left over the Gesture Module, the RGB LEDs illuminate according to the color counter. After moving our hand up or down over the Gesture Module, when we move our hand to the right again, the color of the RGB LED changes. To turn off the desk lamp, you can move your hand to the right over the Gesture Module.



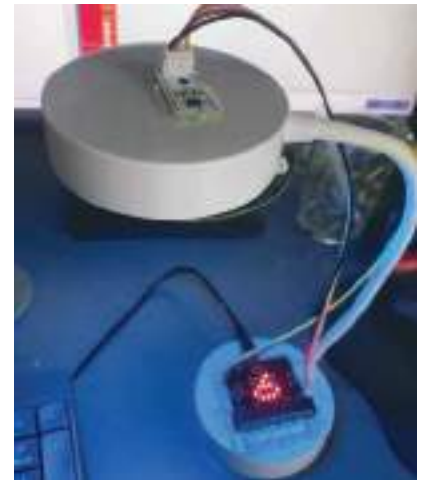
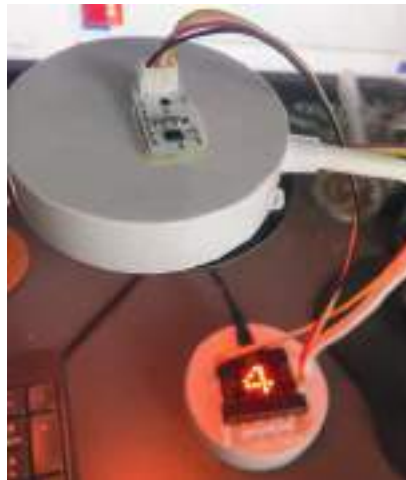
## ● Connection Diagram:

You can prepare this project by breaking PicoBricks modules at proper points.



Addressable Ring  
RGB LEDs

## ● Project Images:





4. Let's prepare the code blocks that detect hand movements by using the Gesture module and give color output on the RGB LED based on the detected values.

5. Illuminates the RGB LED based on the "colorCounter" variable.

```
on gesture left  
  show number colorCounter  
  strip show color red  
  strip show color  
    plus  
    get value of colorCounter  
  plus  
    get value of colorCounter  
  plus  
    get value of colorCounter
```

6. Decreases the "colorCounter" variable.

```
on gesture down  
  change colorCounter by 1  
  if colorCounter >= 5 then  
    set colorCounter to 0  
  show number colorCounter
```

7. Increases the "colorCounter" variable.

```
on gesture up  
  change colorCounter by -1  
  if colorCounter <= 0 then  
    set colorCounter to 5  
  show number colorCounter
```

8. Turns off the RGB LEDs.

```
on gesture right  
  strip clear  
  show icon
```

## 9. The project code is ready!



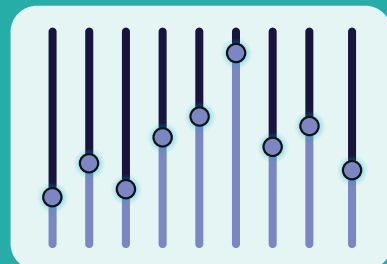




CONTROL PANEL

# IoT Control Panel

Control Panel A



# IoT Control Panel Project

With advancing technology, electronic devices can now communicate with each other. For example, we can control our electronic home appliances such as washing machines, dishwashers, ovens, etc., from our phones. The ability of electronic devices to communicate with each other through internet networks is defined as the "Internet of Things (IoT)." Each device that communicates with others is considered an object. Thanks to the Internet of Things (IoT), we can control a device even if we are at a distant location. In this way, we can eliminate potential dangers that may occur in an environment where we are not present. By using the Internet of Things, we can control devices and also write the data we obtain from these devices to a server in the internet environment. In this way, we can instantly learn any data we want at any location, even if we are not physically present there. **The ThingSpeak application we will use in this project is software that allows us to store the data we obtain from different devices on a server we create and monitor real-time changes in the data.**

## Project Details:

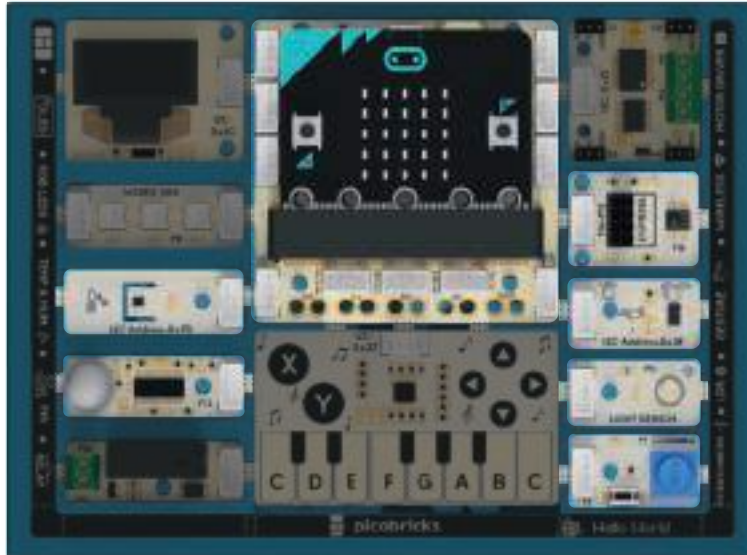
Project Link: <https://makecode.microbit.org/S97455-83875-95111-83258>



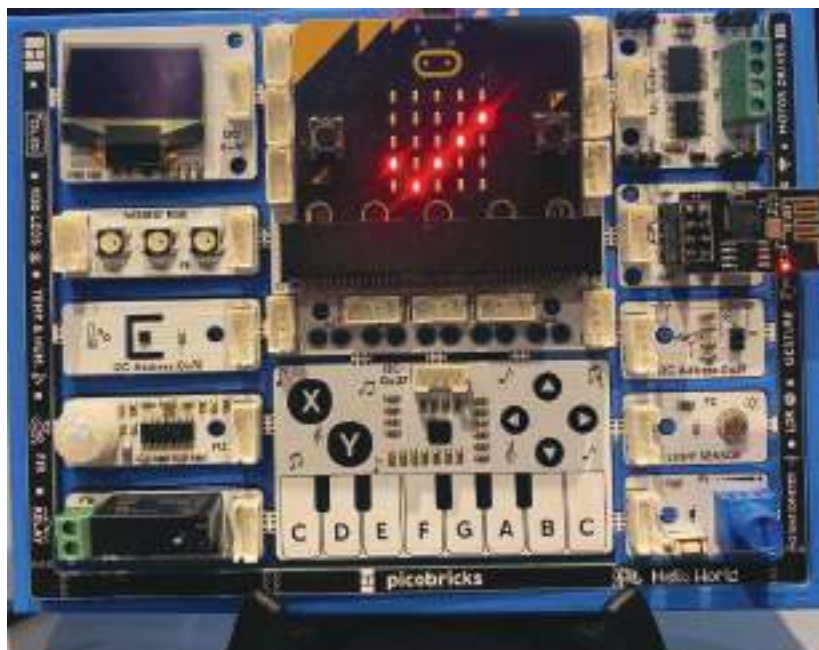
In this project, we will write the data obtained from PicoBricks modules; TEMP&HUM, PIR, LDR, and the Proximity feature in the Gesture module, to a channel created in ThingSpeak and control the changes in the values.

## ● Connection Diagram:

You can prepare this project without making any cable connections.



## ● Project Images:

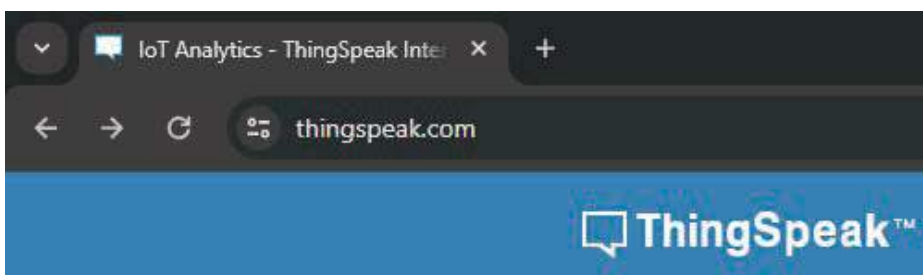


## FicoBricks IoT Control Panel

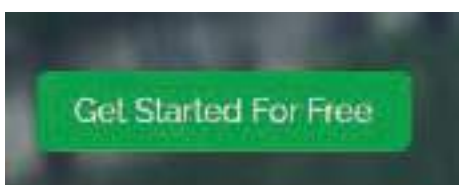


## Usage of ThingSpeak:

1. Go to the <https://thingspeak.com/> address.



2. Click the "Get Started For Free" button.



3. If you have a MathWorks account for ThingSpeak, you can log in by entering your email and password. If you do not have an account, you can create a new one by clicking the “Create one” button.

The screenshot shows the ThingSpeak login interface. At the top, there are navigation links for Channels, Apps, and Support, along with Commercial Use and How to Buy options. The main text explains that users must sign in with an existing MathWorks account or create a new one. It also mentions that non-commercial users can use ThingSpeak for free, while commercial users are eligible for a trial. Below the text is a login form with an Email field and a Next button. To the right of the form is a diagram illustrating the data flow: Smart Connected Devices send data to a cloud labeled 'DATA AGGREGATION AND ANALYTICS ThingSpeak', which then feeds into a MATLAB environment for 'ALGORITHM DEVELOPMENT SENSOR ANALYTICS'.

## Create MathWorks Account

### Email Address

picobricks@example.com

**i** To access your organization's MATLAB licenses, use your school or work email.

### Location

United States

### First Name

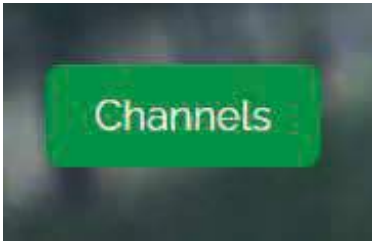
### Last Name

Continue

Cancel

This site is protected by reCAPTCHA and the Google Privacy Policy and Terms of Service apply.

4. After logging into your account, click on the “Channels” button.



5. Click the “New Channel” button.



6. Fill in the information in the window that opens as follows, and click the “Save Channels” button.

A screenshot of a "New Channel" form. The form has the following fields:

- Name: RioBriks IoT Control Panel
- Description: (empty)
- Field 1: Temperature (checked)
- Field 2: Humidity (checked)
- Field 3: RH (checked)
- Field 4: Proximity (checked)
- Field 5: LDR (checked)
- Field 6: Potentiometer (LDR) (checked)
- Field 7: (empty) (unchecked)
- Field 8: (empty) (unchecked)
- Metadata: (empty)
- Tags: (empty)
- Link to External Site: (empty)
- Link to GitHub: (empty)

At the bottom of the form is a green button labeled "Save Channel".

7. We have now created the necessary channel to print the data obtained from PicoBricks modules. We will use the following 'API Keys' in the code blocks we will create in MakeCode to write the data to the channel.



## ● MakeCode Code of The Project:

If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

1. Let's drag the code blocks that initialize the modules whose data we will print in our project. Then, create a conditional structure that displays a smiling face emoji on the matrix LEDs if the Wi-Fi module is successfully initialized and connected to the Wi-Fi network, and displays a sad face emoji if it's not initialized.



2. Using the 'upload data to thingspeak' block from the PicoBricks extension, increase the number of fields to six using the '+' button. Then, drag the code blocks of the modules from which we want to receive data into the respective field areas. Finally, just before the loop repeats to check if the data has been sent, create a conditional structure with a (✓) icon if the data has been sent and an X icon if it hasn't.



3. The Code of The Project is Ready!





# IoT Vase



# IoT Vase Project

Flowers thrive when provided with the right conditions. Especially for the flowers we nurture at home, providing the correct conditions can be a bit challenging, making their care more complex. The optimal conditions for flowers include values such as the brightness of the environment, soil moisture, and the temperature and humidity of the surroundings. Particularly, some flowers we grow may prefer sunlight, while for others, sunlight can have the opposite effect. Similarly, while some flowers need frequent watering, others may react negatively when too much water is poured.

In this way, we can provide various examples based on different factors and types of flowers. To overcome such negative situations, we occasionally change the locations of flowers or adjust our watering frequency based on the type of flower. However, these measures may not be applicable when we are not at home. In such situations, IoT devices come into play. Through IoT (Internet of Things), as we learned and experienced in the "IoT Control Panel" project, we can communicate our devices with each other. Thanks to advancing IoT technologies, we can monitor the condition of our flower instantly by connecting the vase we use at home to our phone. The values we will monitor can vary based on the type and number of sensors attached to our vase.

In this project, we will communicate any vase we use at home with our smart devices by using PicoBricks modules and ThingSpeak.

## Project Details:

Project Link: <https://makecode.microbit.org/S08509-91200-31407-52307>



In this project, we will obtain some data from the PicoBricks modules, specifically the Temperature & Humidity module, the LDR module, and the soil moisture sensor module connected to the P1-P2 connector.

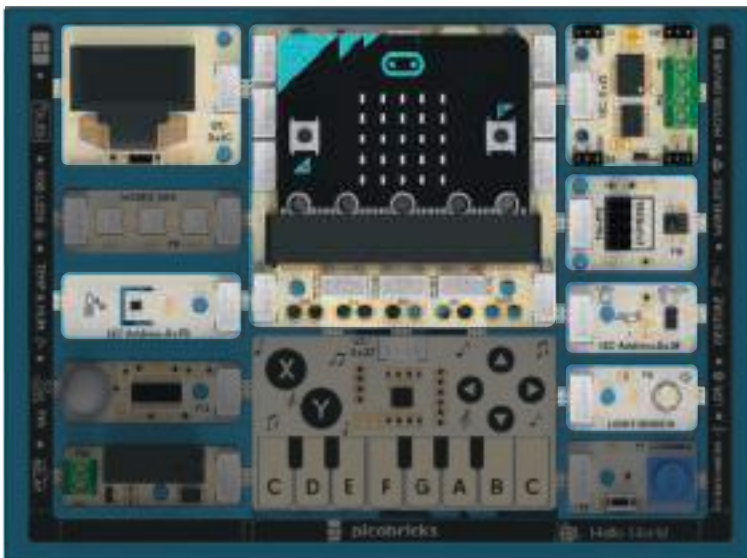
We will transfer these data to the "IoT Vase" channel that we created on ThingSpeak. At the same time, we will display these transferred data on the PicoBricks OLED screen. Additionally, we will ensure the watering of the plant by running the water pump connected to the motor driver based on the values obtained from the soil moisture sensor.



If you want, you can manually stop the water pump by pressing the A button on the Micro:Bit.

## ● Connection Diagram:

You can prepare this project by breaking PicoBricks modules at suitable points.



Water Pump



Soil Moisture Sensor

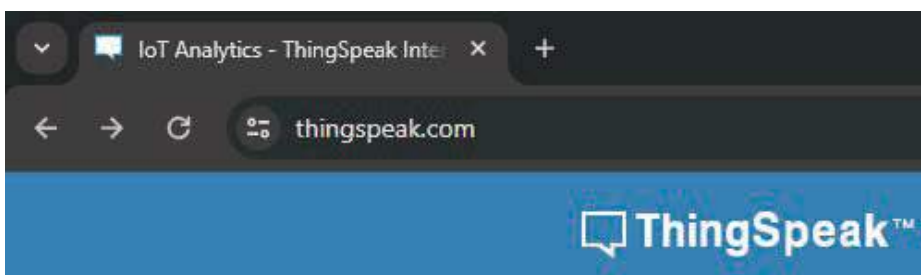
## Project Images:



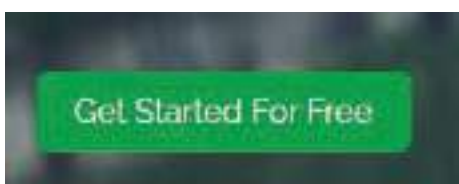


## Usage of ThingSpeak:

1. Go to the <https://thingspeak.com/> address.



2. Click the "Get Started For Free" button.



- If you have a MathWorks account for ThingSpeak, you can log in by entering your email and password. If you do not have an account, you can create a new one by clicking the “Create one” button.

The screenshot shows the ThingSpeak login interface. At the top, there are navigation links for Channels, Apps, and Support, along with Commercial Use and How to Buy options. The main text explains that users must sign in with an existing MathWorks account or create a new one. It also mentions that non-commercial users can use ThingSpeak for free, while commercial users are eligible for a time-limited free evaluation. Below the text is a login form with an Email input field and a button labeled "Login". To the right of the form is a diagram illustrating the data flow: Smart Connected Devices send data to a cloud labeled "DATA AGGREGATION AND ANALYTICS ThingSpeak". This cloud then sends data to a MATLAB interface labeled "ALGORITHM DEVELOPMENT SENSOR ANALYTICS".

## Create MathWorks Account

### Email Address

✓

**i** To access your organization's MATLAB license, use your school or work email.

### Location

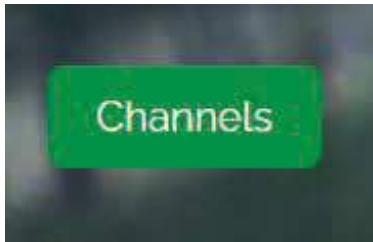
▼

### First Name

### Last Name

This site is protected by reCAPTCHA and the Google Privacy Policy and Terms of Service apply.

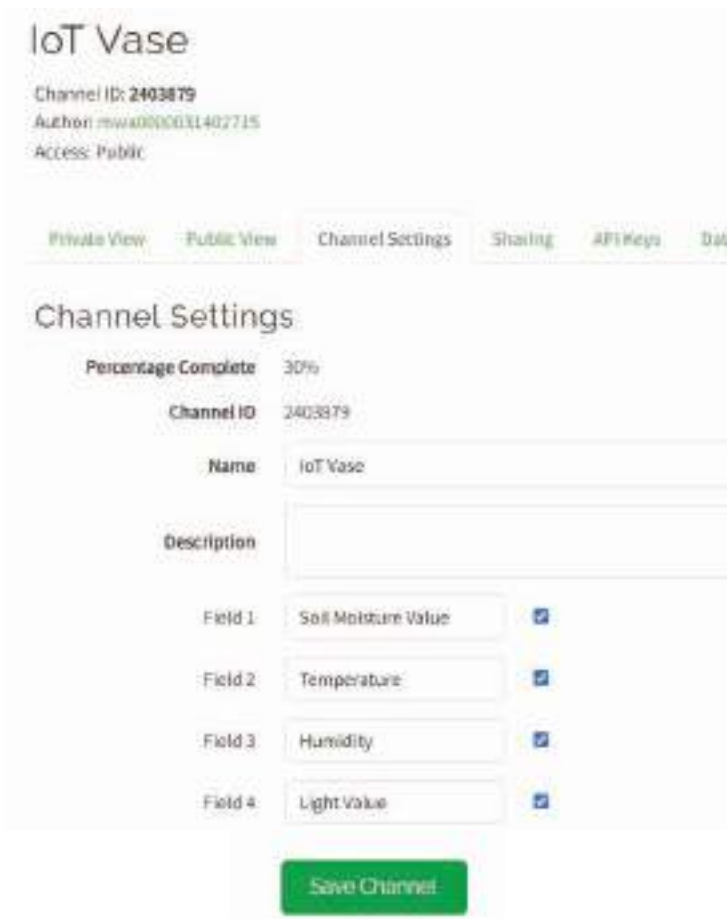
4. After logging into your account, click on the “Channels” button.



5. Click the “New Channel” button.



6. Fill in the information in the window that opens as follows, and click the “Save Channels” button.



**IoT Vase**  
Channel ID: 2403879  
Author: mwax0000031402715  
Access: Public

Private View Public View **Channel Settings** Sharing API Keys Data

**Channel Settings**

Percentage Complete 30%

Channel ID 2403879

Name IoT Vase

Description

Field 1: Soil Moisture Value

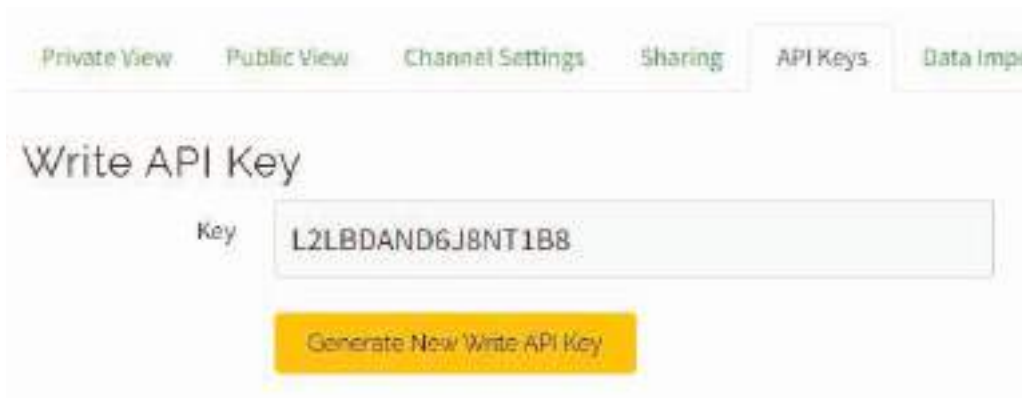
Field 2: Temperature

Field 3: Humidity

Field 4: Light Value

Save Channel

7. We have now created the necessary channel to print the data obtained from PicoBricks modules. We will use the following 'API Keys' in the code blocks we will create in MakeCode to write the data to the channel.



## ● MakeCode Code of The Project:

If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

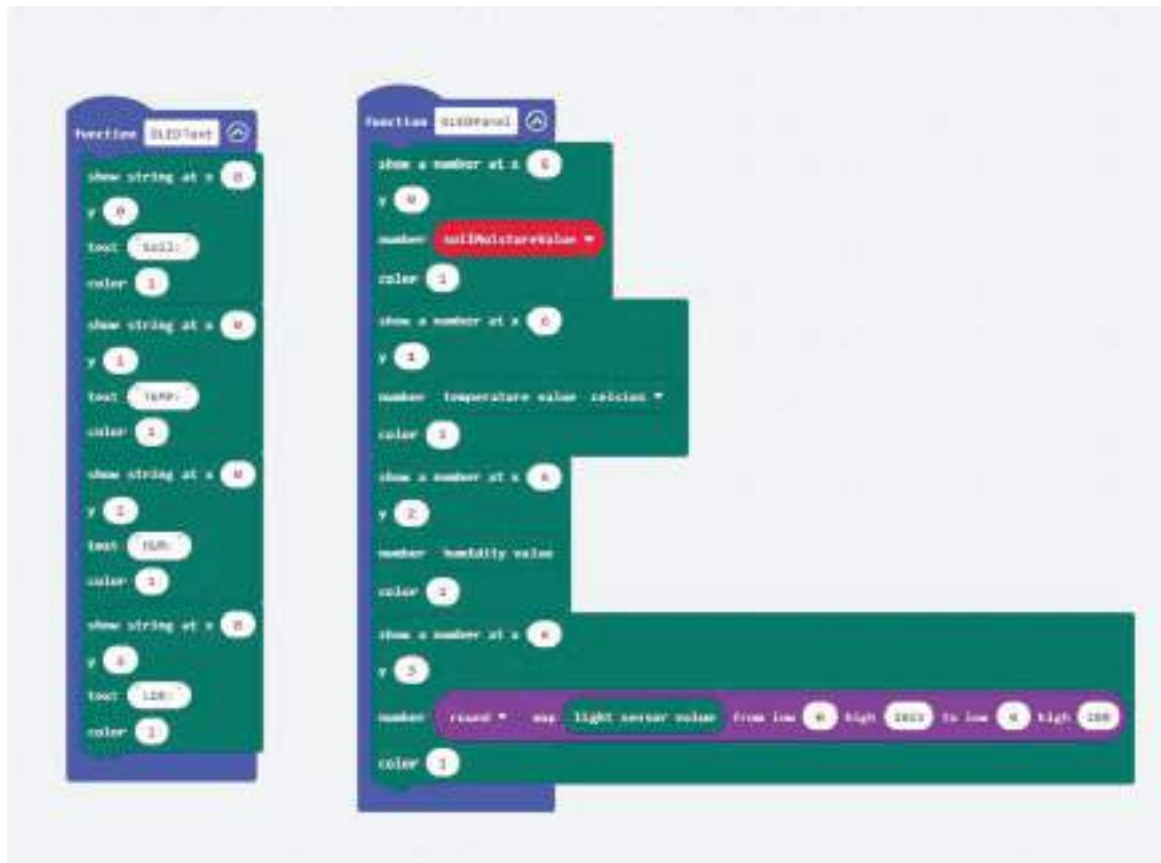
1. Let's create conditional structures within the "on start" block that check if the necessary sensors have been initialized and if the Wi-Fi module has been initiated and connected to Wi-Fi after the project is launched.



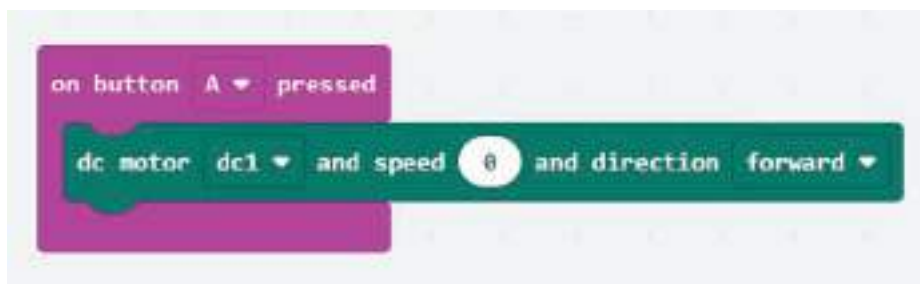


2. Let's proceed by defining the functions we will use within the "forever" block.

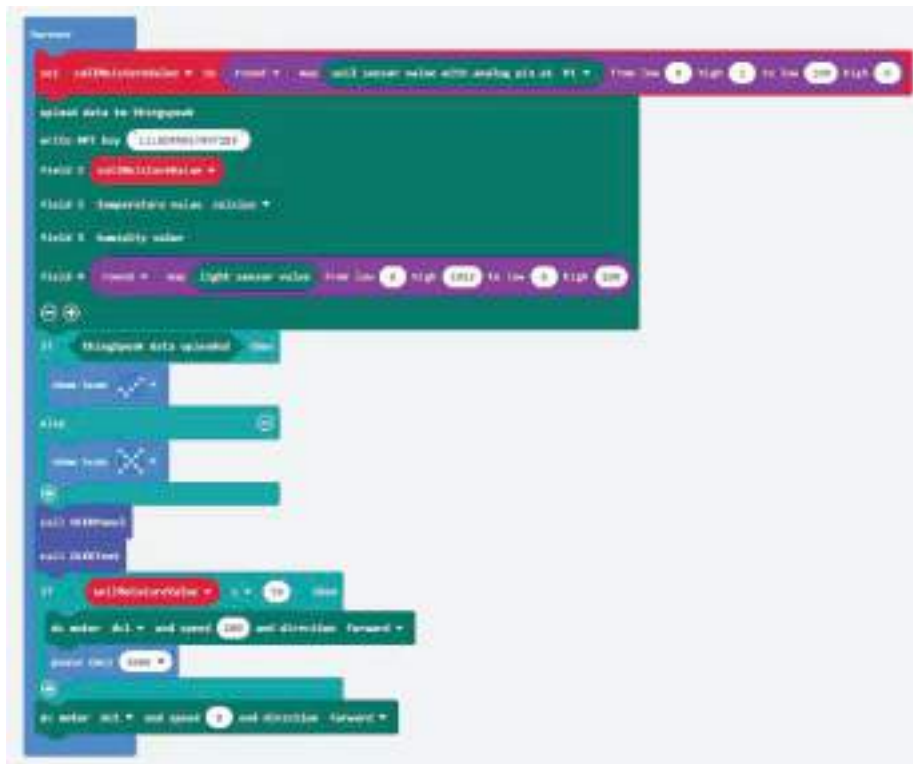
a) Let's create two functions named "OLEDText" and "OLEDPanel" to define the code blocks that enable writing the data obtained from sensors and the necessary textual expressions on the OLED screen. Then, let's assign these functions within these blocks.



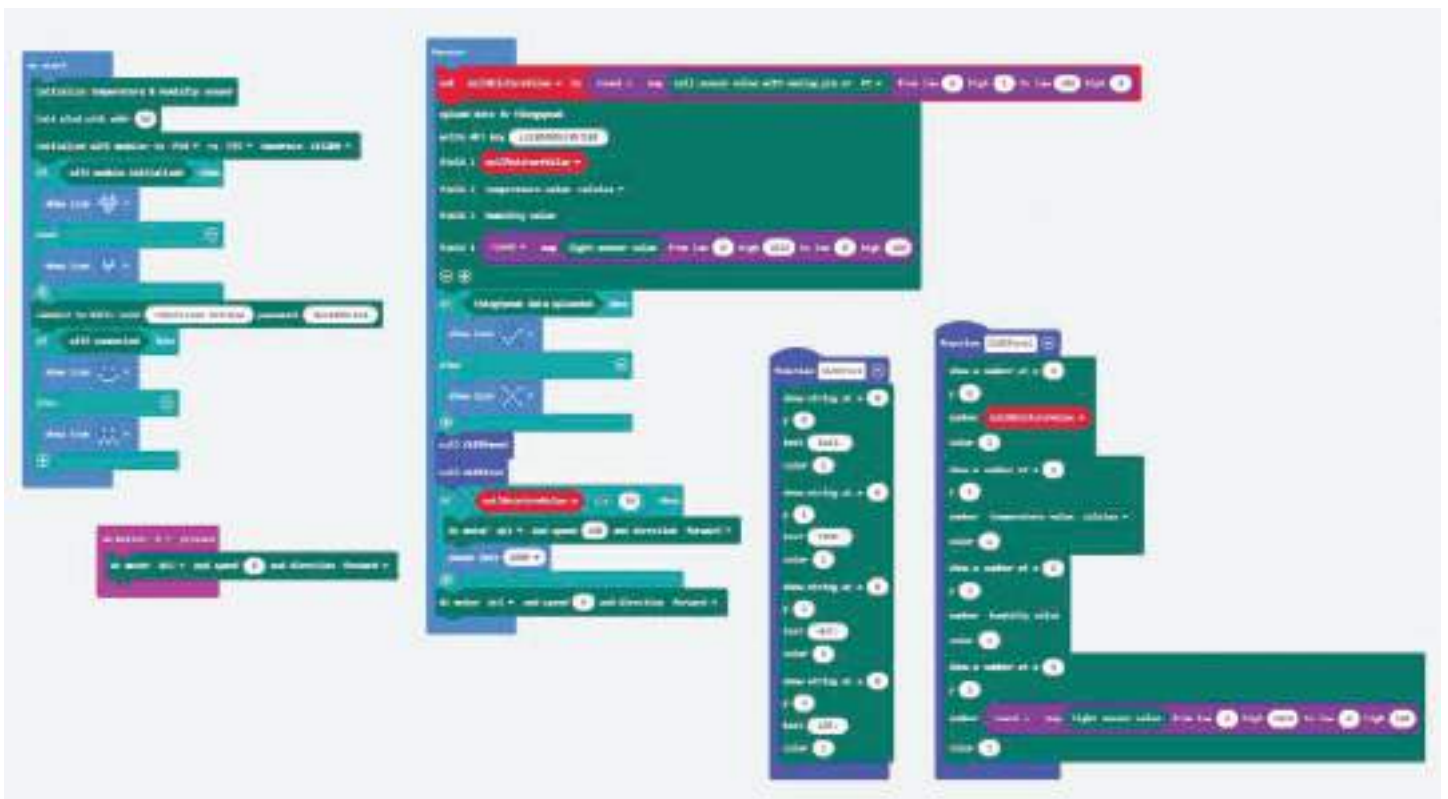
3. Let's create the code blocks that stop the motor when the A button on the Micro:Bit is pressed, in the "on button A pressed" block, which we took from the "input" blocks.



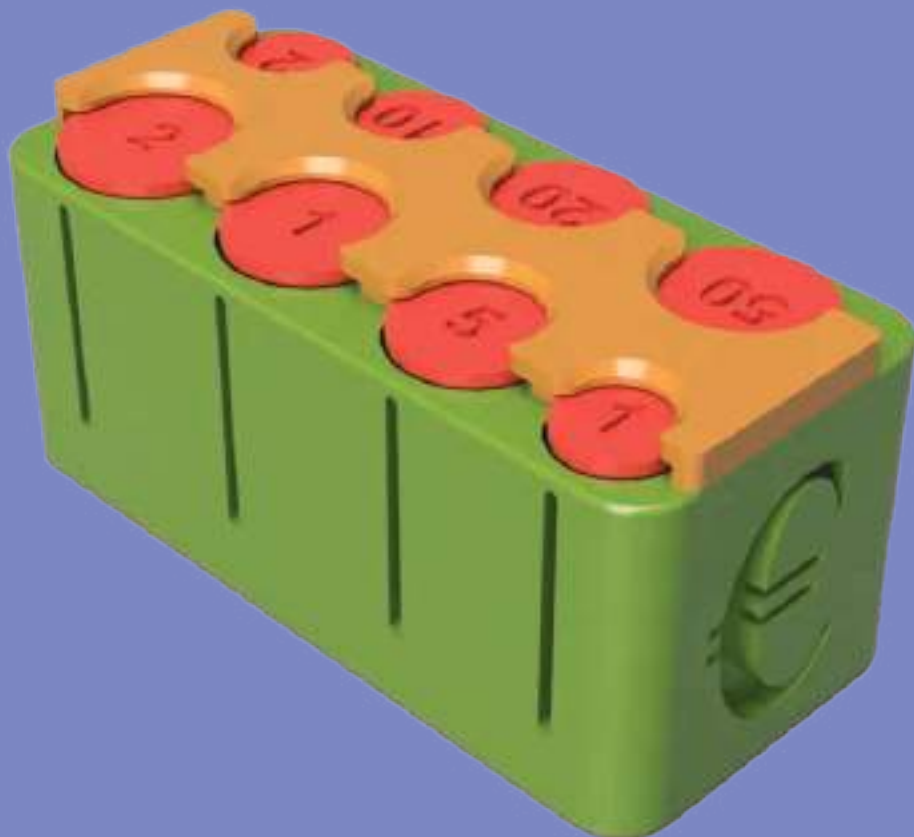
4. Let's create code blocks within the "forever" loop that transfer the data we obtained to the channel we have created and run the motor based on the received data.



5. The Code of The Project is Ready!



# Coin Dispenser



# Coin Dispenser Project

Some people dislike carrying coin in their pockets. This might be due to the extra weight it adds or the noise it makes while walking. For others, collecting coins could be a hobby. However, when collecting coins, we may struggle to separate them. The easiest way to separate coins is by their dimensions. Each coin with different values also has different dimensions. By using the dimensions of the box where we collect the coins, we can quickly separate them. This way, each coin fits into its corresponding box based on its value and can be separated quickly. Moreover, this separation process also makes it easier to count the coins.

## Project Details:

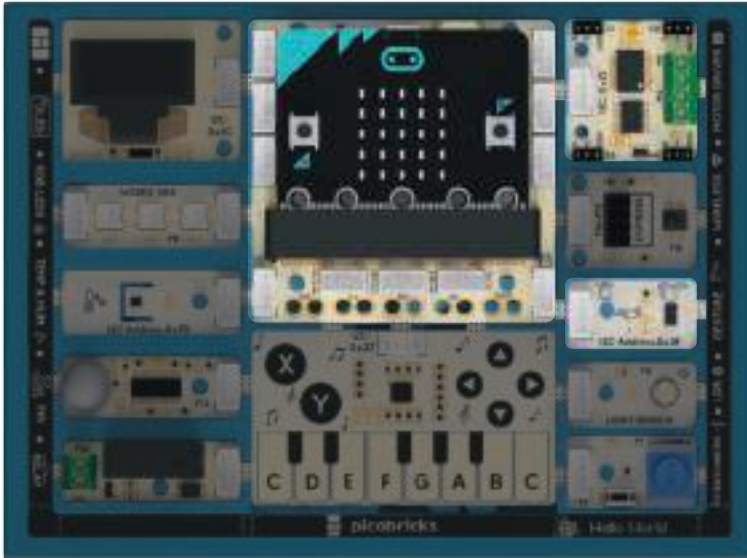
Project Link: <https://makecode.microbit.org/S74621-15169-75095-95958>



In this project, we will use a 3D printer to create a coin dispenser that can be controlled using hand gestures through a gesture module. When we make a rightward gesture with the gesture sensor, the coin dispenser will use a gear system to launch the bottom coin. When we make a leftward gesture, the gear system will pull itself to the left.

## ● Connection Diagram:

You can prepare this project by breaking down PicoBricks modules at proper points.



Servo Motor

## ● Project Images:



## ● The STL Files of The Project:

You can access the STL files of the project by scanning the QR code or opening the link in your browser.

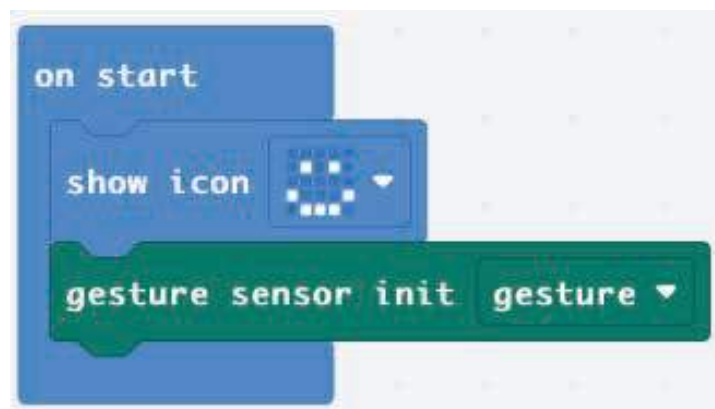
<https://rft.ist/thingiverse>



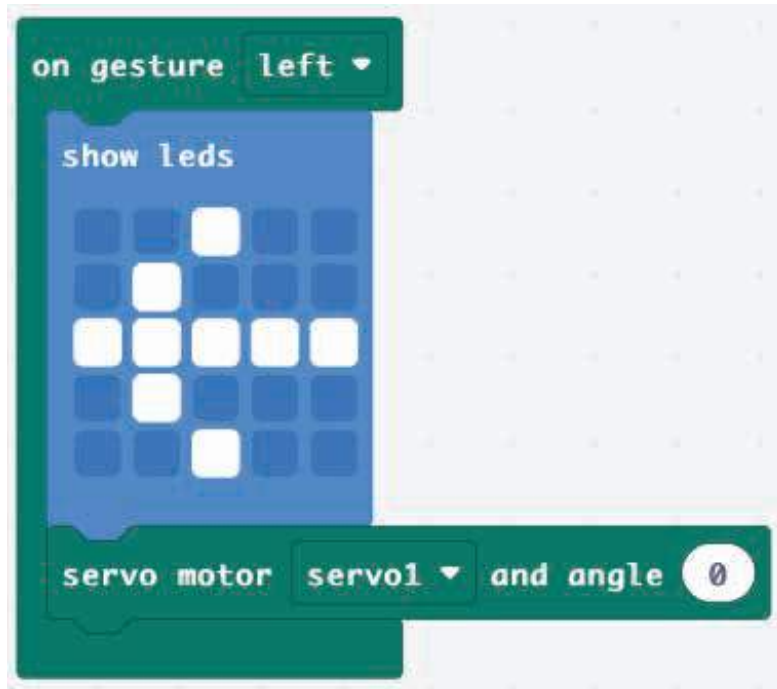
## ● MakeCode Code of The Project:

If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

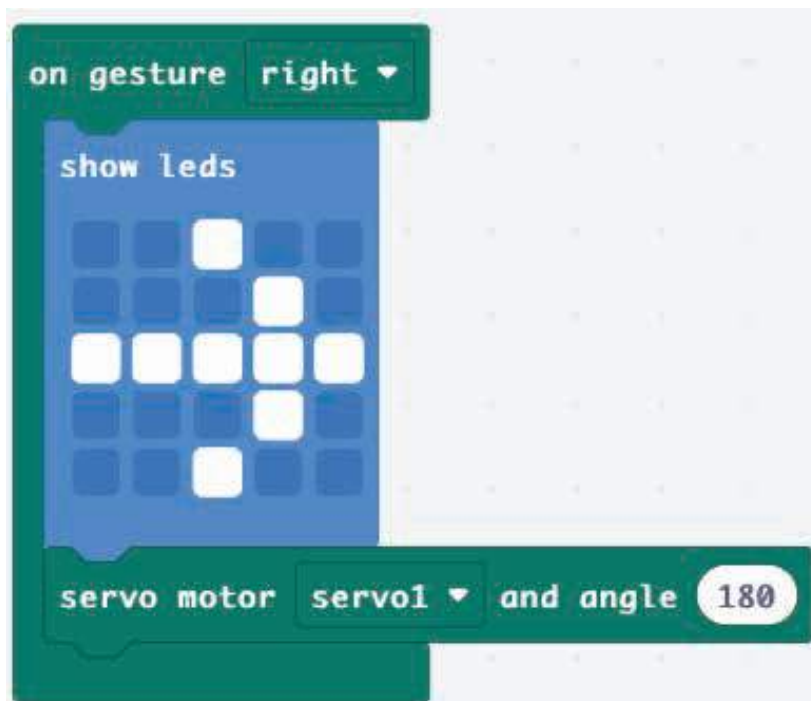
1. When the project is started, drag the "show icon" and "gesture sensor init gesture" blocks into the "on start" block to create a smiling face icon on the Micro:Bit matrix LED and initialize the gesture module.



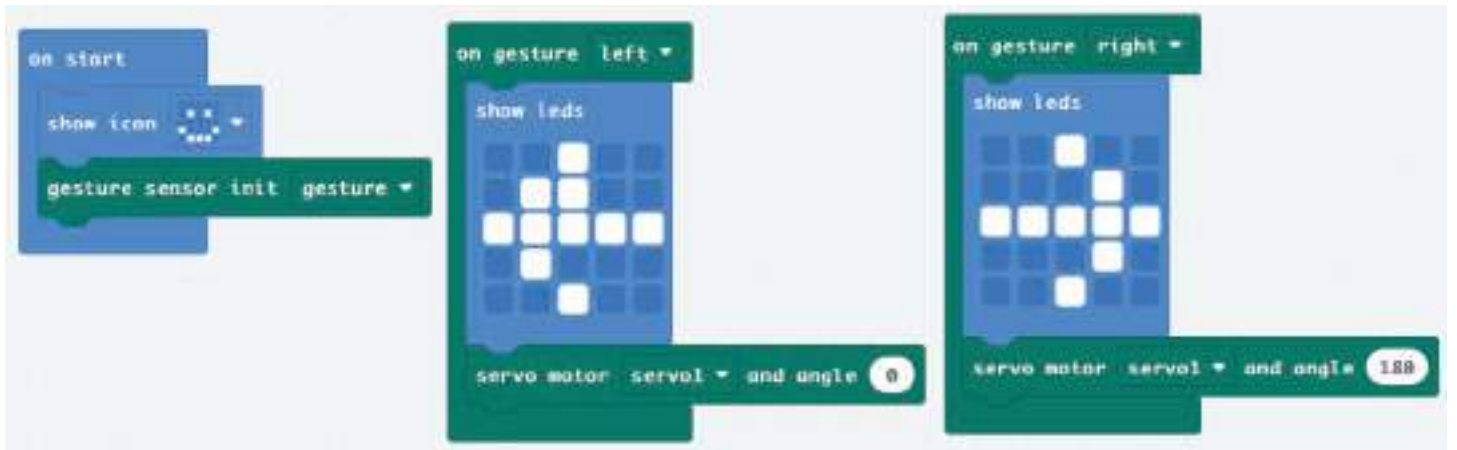
2. When we make a leftward gesture with our hand on the gesture module, the angle of the servo motor connected to the gear mechanism of the coin dispenser becomes 0, and it moves to the left.



3. When we make a rightward gesture with our hand on the gesture module, the angle of the servo motor connected to the gear mechanism of the coin dispenser becomes 180, and it moves to the right.



#### 4. The Code of The Project is Ready!





# Gesture Controlled ARM Pan Tilt



# Gesture Controlled ARM Pan Tilt Project

Robot arms have replaced human labor in the industrial field. They undertake tasks such as carrying and rotating loads that are too heavy or large for a human to handle in factories. Their ability to be positioned with precision up to one-thousandth of a millimeter surpasses the precision achievable by human hands. When you watch production videos of automobile factories, you will see how crucial robot arms are. They are called "robots" because they can perform the same task infinitely by being programmed. The reason for calling them "arms" is because they have an articulated structure similar to our arms. The number of axes a robot arm can rotate and move in determines its degrees of freedom. Robot arms are also used in carving and shaping aluminum and various metals. These devices, known as 7-axis CNC routers, can shape metals similar to how a sculptor shapes clay.

Depending on the purpose of use in robot arms, both [stepper motors](#) and [servo motors](#) are utilized. PicoBricks enables you to create projects using servo motors.

## Project Details:

Project Link: [https://makecode.microbit.org/\\_0ysV6Y0Hbg54](https://makecode.microbit.org/_0ysV6Y0Hbg54)



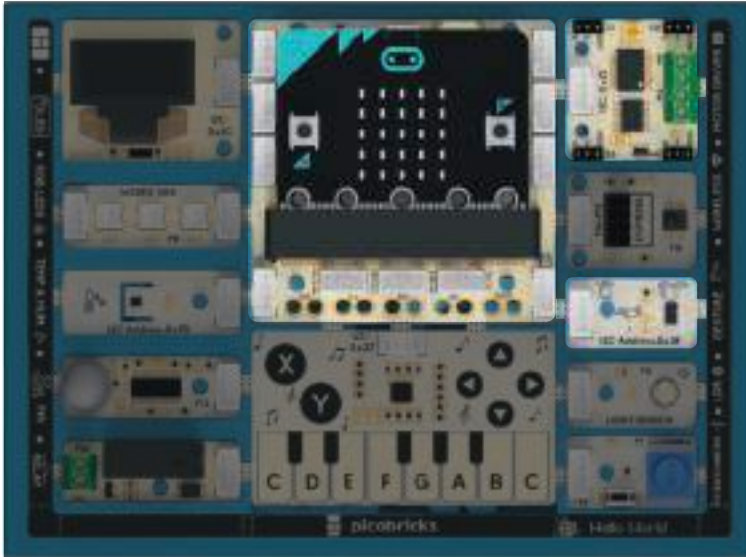
In this project, we will use the "[gesture](#)" feature of the PicoBricks gesture module to detect up-down, right, and left hand movements, and move a pan-tilt system accordingly. Additionally, when we press the "[A](#)" button on the Micro:Bit, we will reset the servo motors to their initial positions to center the system.



By mounting the RGB LED module on the front surface of this system, we can create a lighting system that can move in two axes.

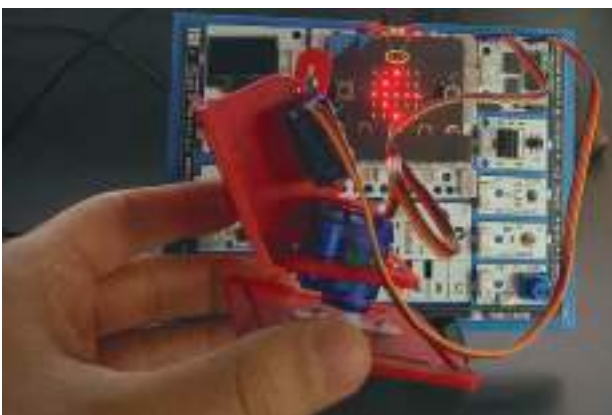
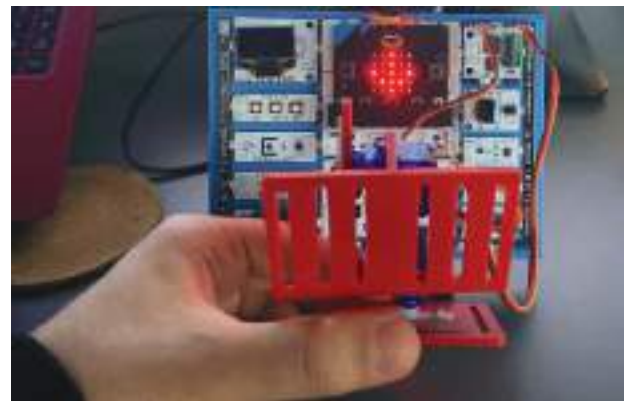
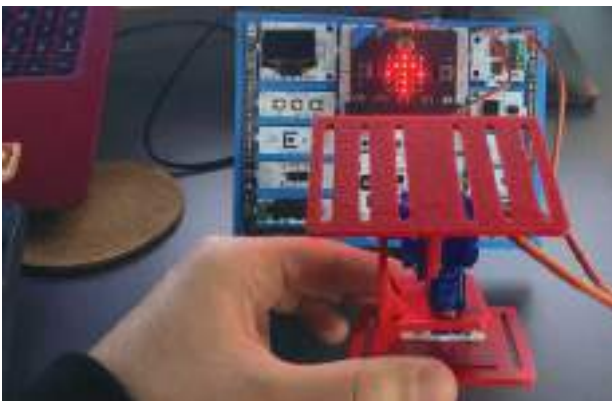
## ● Connection Diagram:

You can prepare this project by breaking down PicoBricks modules at proper points.

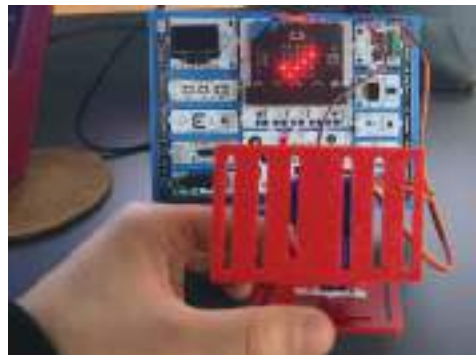
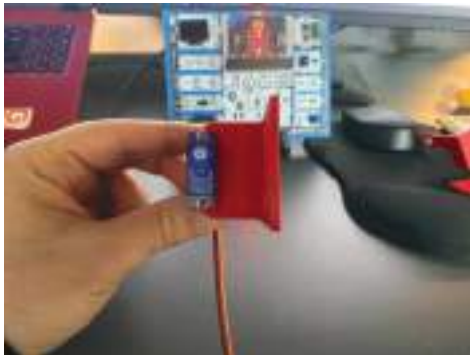


Servo Motor

## ● Project Images:



## ● Installation Images:



## ● The STL Files of The Project:

You can access the STL files of the project by scanning the QR code or opening the link in your browser.

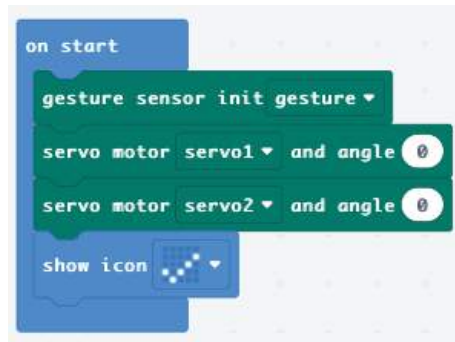
<https://rbt.ist/thingiverse>



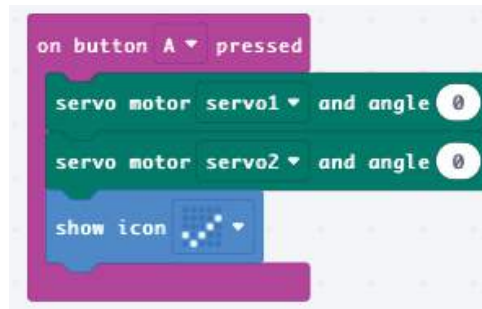
## ● MakeCode Code of The Project:

If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

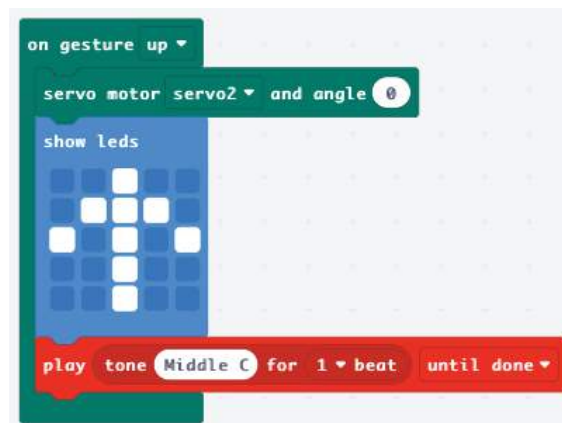
1. When the project is started, the gesture sensor is initialized, servo angles are centered, and a checkmark is displayed on the Matrix LEDs.



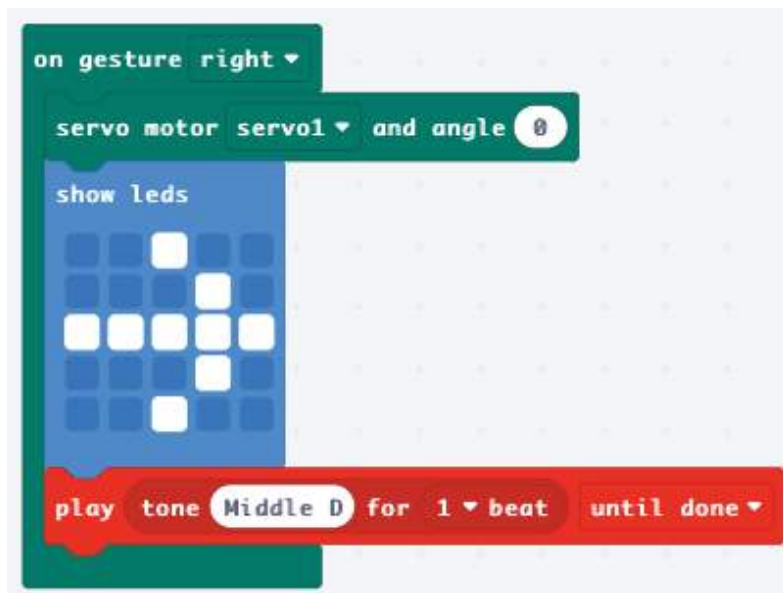
2. When the A button of the Micro:Bit is pressed, center the servo motors and display a checkmark on the Matrix LEDs.



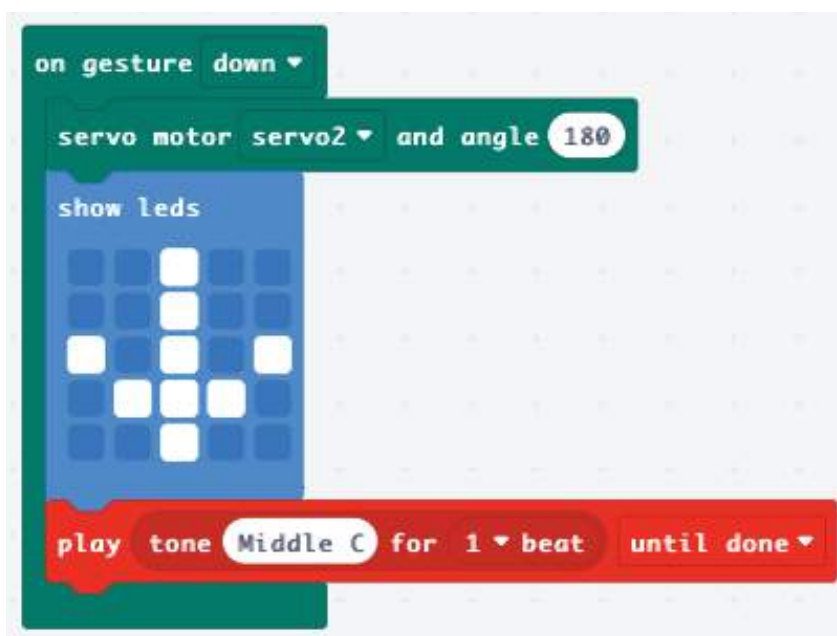
3. When the gesture sensor detects an upward movement, it sets the angle of the second servo motor to 0 degrees, creates an upward arrow on the Matrix LEDs, and plays the C note on the buzzer.



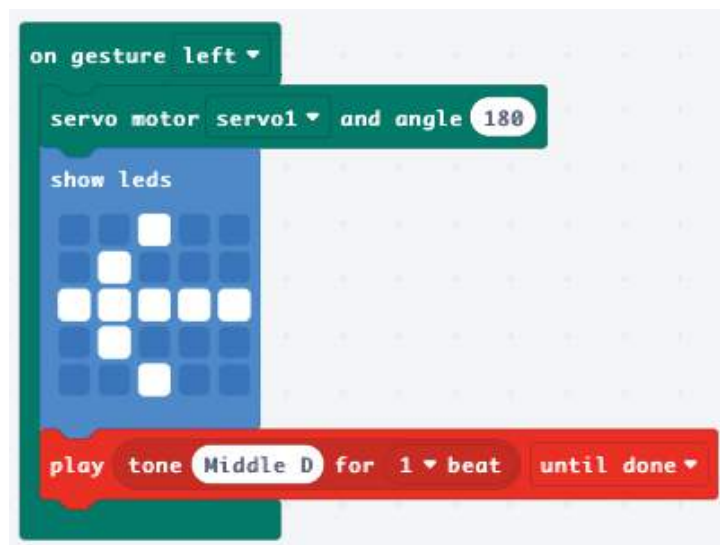
4. When the gesture sensor detects a rightward movement, it sets the angle of the first servo motor to 180 degrees, creates a left arrow on the Matrix LEDs, and plays the D note on the buzzer.



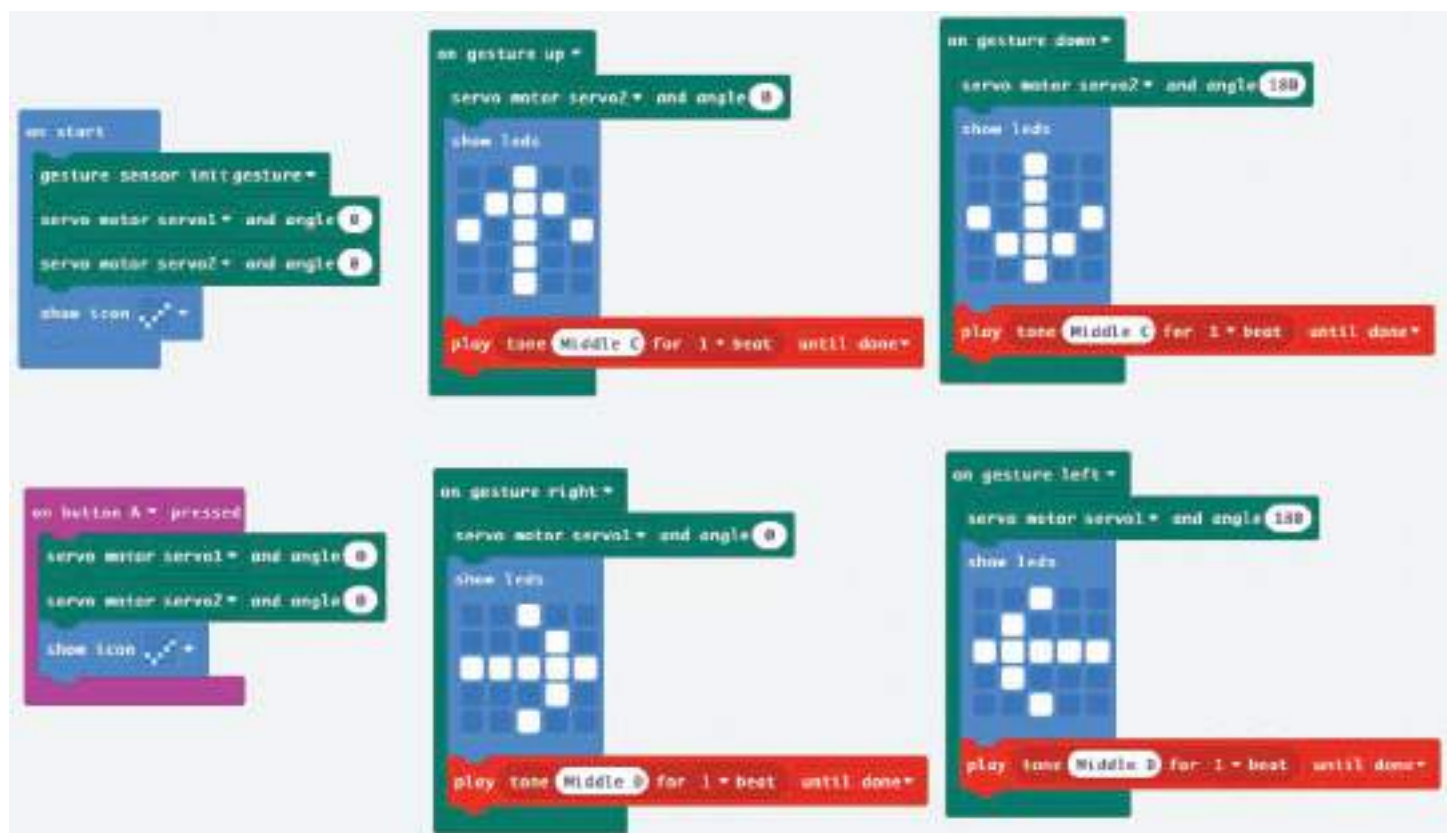
5. When the gesture sensor detects a downward movement, it sets the angle of the second servo motor to 180 degrees, creates a downward arrow on the Matrix LEDs, and plays the C note on the buzzer.



6. When the gesture sensor detects a leftward movement, it sets the angle of the first servo motor to 0 degrees, creates a right arrow on the Matrix LEDs, and plays the D note on the buzzer.



7. The Code of The Project is Ready!



# 3D Labyrinth





## 3D Labyrinth Project

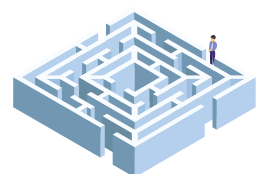
There are multiple ways to exit a maze, but the most well-known method is to follow the wall with your left/right hand. Although this method may take some time, you can definitely get out of the maze by consistently touching the wall. Maze tests enhance problem-solving skills. Someone who frequently solves maze tests can quickly come up with solutions to the problems they encounter. In this project, we will design a maze and the necessary mechanical parts to move the maze by using a 3D printer.

### Project Details:

Project Link: <https://makecode.microbit.org/S96837-16148-67126-29091>

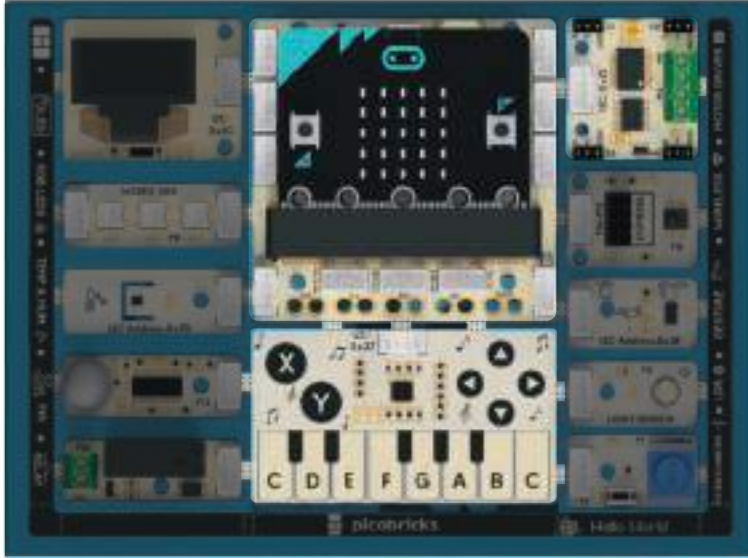


Let's create a maze project that can move in right, left, up, and down directions by assembling 3D printer parts as shown in the visuals. To ensure the movement of the maze in this project, we will utilize two servo motors connected to the PicoBricks motor driver. The direction keys on the PicoBricks Touch & Piano module will be used to move the servo motors in the desired direction. By using the Right, Left, Up, and Down direction keys, we will control the direction and attempt to navigate the ball placed inside the maze to the exit.



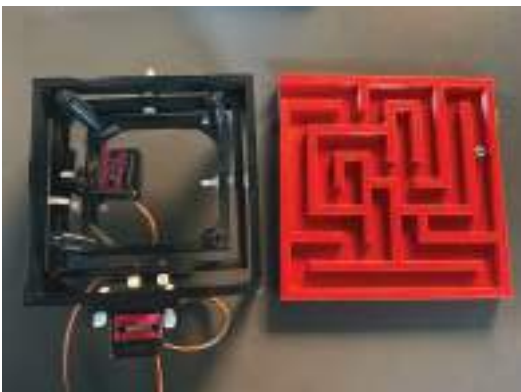
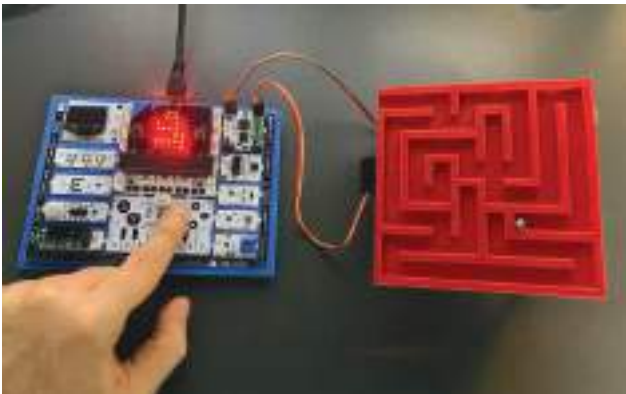
## ● Connection Diagram:

You can prepare this project by breaking down PicoBricks modules at proper points.



Servo Motor

## ● Project Images:



## ● The STL Files of The Project:

You can access the STL files of the project by scanning the QR code or opening the link in your browser.

<https://rbt.ist/thingiverse>



## ● MakeCode Code of The Project:

If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

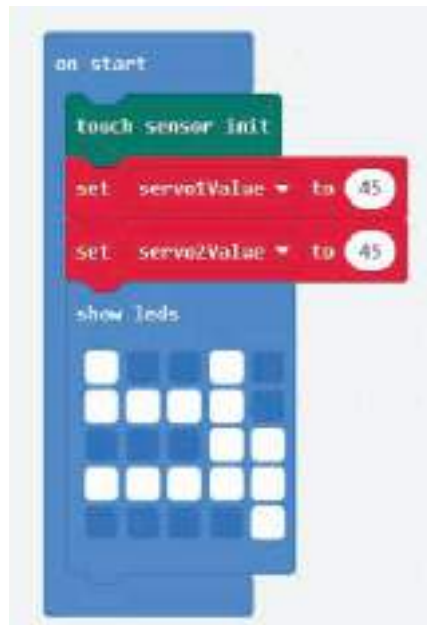
1. When we create our project, two blocks, named '**on start**' and '**forever**,' appear on the screen. The blocks dragged into the '**on start**' block are executed only once when the Micro:Bit is run. '**Forever**' repeats the blocks dragged into it continuously until the program is stopped. Since we will not use the '**forever**' block in this project, you can delete it.



2. When our project starts, drag the code block that initializes the Touch sensor into the '**on start**' block for the Touch sensor.

3. Create two variables named '**servo1Value**' and '**servo2Value**' and set their initial values to '**45**'. The reason for this is to position Servo 1 and Servo 2 at 45 degrees when the project is started. This will align the maze to the center.

4. Finally, use the 'show leds' block to create a maze-like shape on the Micro:Bit Matrix LED and then use the LEDs to form a shape resembling a maze.



5. At the beginning of the 'forever' block, place the Servo 2 and Servo 1 variables inside the servo motor block obtained from the motor blocks. Each time the loop iterates, set the angles of servo 2 and servo 1 to the values of the 'servo2Value' and 'servo1Value' variables, respectively.



6. At the beginning of the 'forever' block, place the Servo 2 and Servo 1 variables inside the servo motor block obtained from the motor blocks. Each time the loop iterates, set the angles of servo 2 and servo 1 to the values of the 'servo2Value' and 'servo1Value' variables, respectively.

a) When we touch the 'Up' button on the touch sensor, let's decrease the 'servo1Value' variable by one. If 'servo1Value' is 15, set its value to 16. This way, the variable will not go below 15.

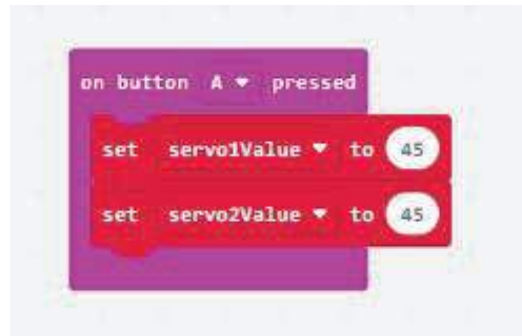
**b)** When we touch the 'Down' button on the touch sensor, let's increase the 'servo1Value' variable by one. If 'servo1Value' is 58, set its value to 57. This way, the variable will not exceed 58.

**c)** When we touch the 'Right' button on the touch sensor, let's increase the 'servo2Value' variable by one. If 'servo2Value' is 80, set its value to 79. This way, the variable will not exceed 80.

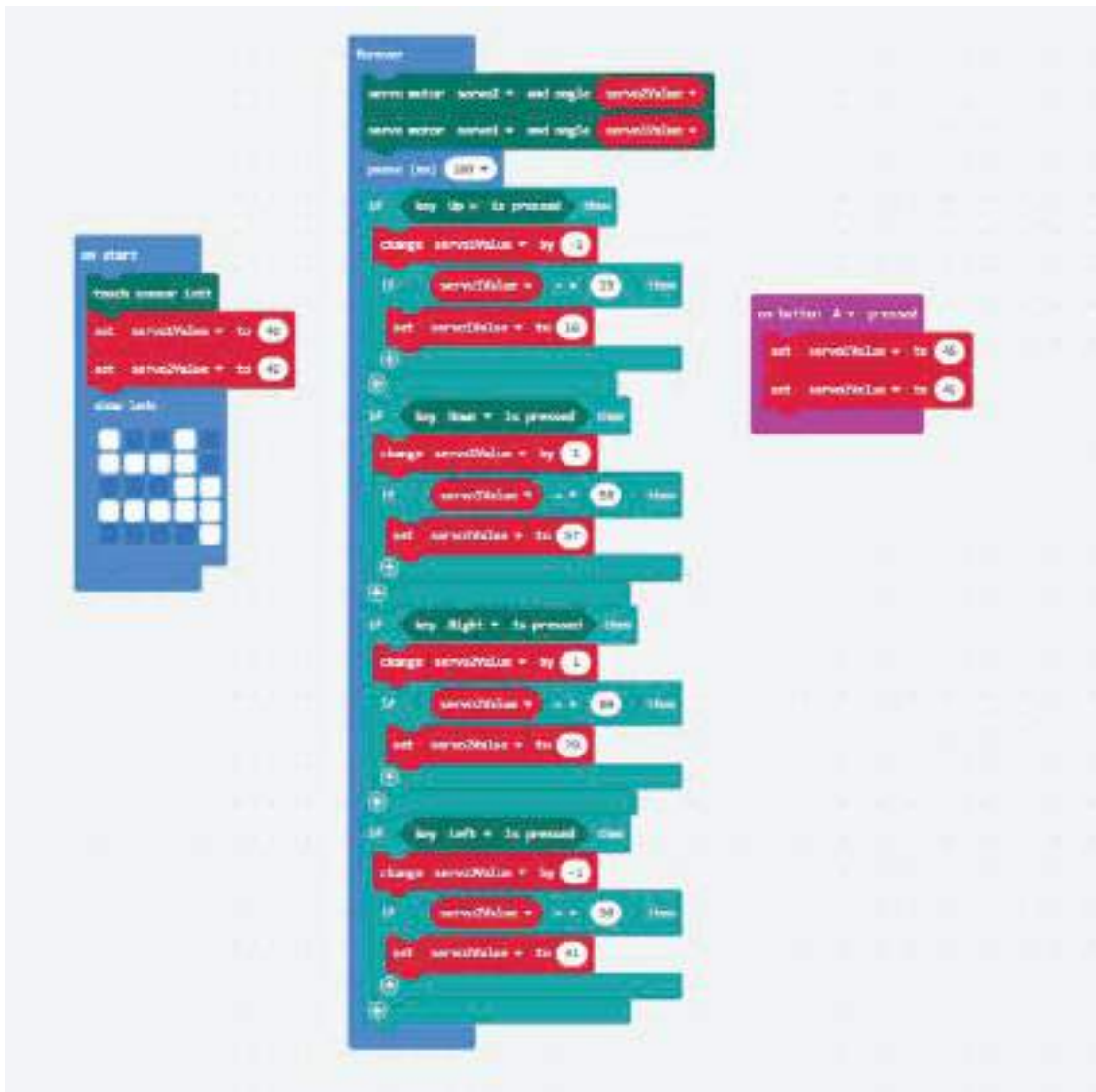
**d)** When we touch the 'Left' button on the touch sensor, let's decrease the 'servo2Value' variable by one. If 'servo2Value' is 30, set its value to 31. This way, the variable will not go below 30.



7. When we press the A button on the Micro:Bit, inside the 'on button A pressed' block obtained from the 'input' blocks, set the 'servo1Value' and 'servo2Value' variables to 45 to center the maze.



8. The Code of The Project is Ready!





# Radars

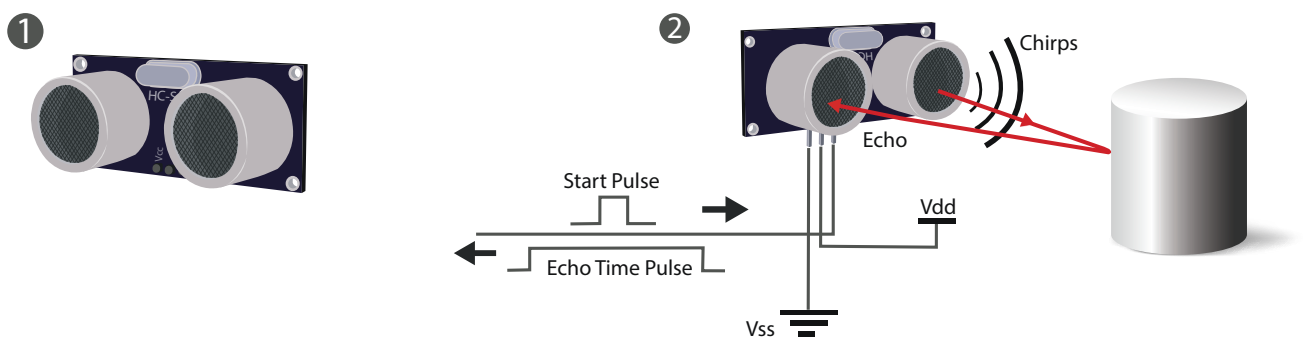


# Radar Project

The radar is a device that detects objects in its surroundings, their direction of movement, their speeds, and other values through radio waves. The effective ranges of radars can vary. Depending on this variation, their applications change. Radars are frequently used to ensure security in various vehicles such as ships, airplanes, etc., and in military areas.

Since radar operates with radio waves, which sensors on PicoBricks or in the set can we use to create a sample radar project? Among the PicoBricks modules and sensors in the set, the sensors with distance measuring capability are the Ultrasonic Distance Sensor (HC-SR04) and the gesture module. Due to the limited range of distances that the gesture module can measure, the ultrasonic distance sensor would be more suitable for a project of this kind.

Ultrasonic distance sensors detect objects around them by using sound waves. As explained in the diagram below, the distance to the object in front is determined by calculating the time it takes for the sound wave emitted from the Trig pin to hit the Echo pin.



In this project, we will create a radar project using the PicoBricks, ultrasonic distance sensor, and servo motor.





## Project Details:

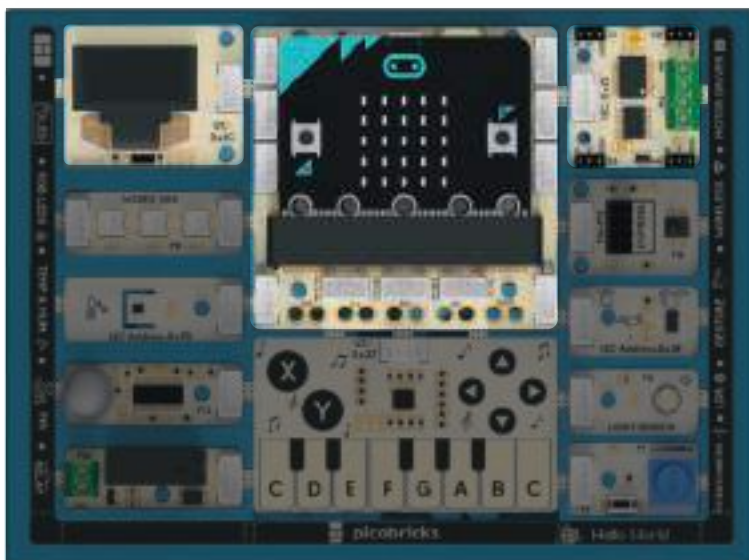
Project Link: <https://makecode.microbit.org/S81455-39853-02069-28632>



In this project, we will implement a radar project by rotating the servo motor connected to PicoBricks' motor driver based on the value detected by the ultrasonic distance sensor connected to the P1-P2 pins, within a 0-180 degree angle. The radar will move in the range of 0-180 degrees until it detects a value within the range determined by the potentiometer module. If an object is detected within the specified range, it will stop moving, emit a warning sound from the buzzer, and display the distance and angle of the object from the radar on the OLED screen.

## Connection Diagram:

You can prepare this project by breaking down PicoBricks modules at proper points.

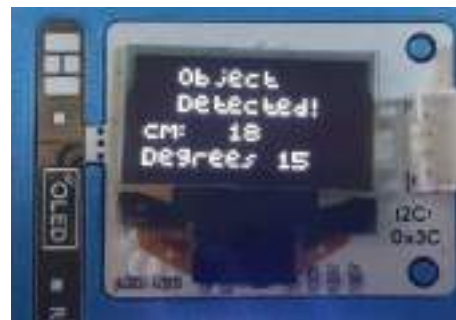


Servo Motor



HC-SR04

## Project Images:



## The STL Files of The Project:

You can access the STL files of the project by scanning the QR code or opening the link in your browser.

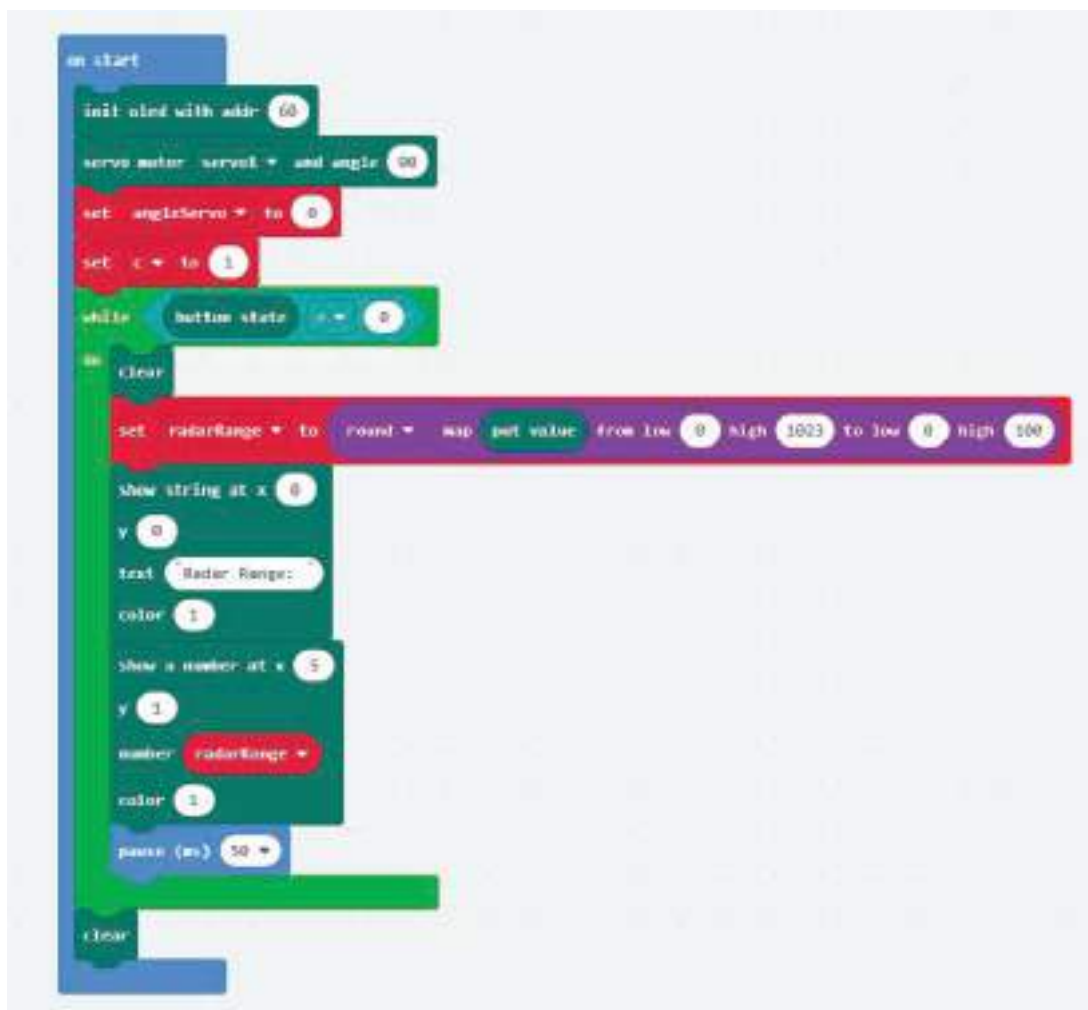
<https://rbt.ist/thingiverse>



## ● MakeCode Code of The Project:

If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

1. When the project is initiated, let's create the necessary code blocks within the “on start” block to determine the range of the radar with the potentiometer until the button is pressed.



2. Let's create code blocks that move the servo motor connected to the motor driver in the range of 0-180 degrees after a button press until an object enters the range of the ultrasonic distance sensor. During this scanning process, "Scanning..." will be displayed on the OLED screen.

```
forever
  clear
  while (round (ultrasonic distance sensor value with trig pin at P2 and echo pin at P1) < radarRange)
    do
      show string at x 2
      y 2
      text: "Scanning..."
      color 1
      servo motor: servo1 and angle: angleServo
      if (c == 1) then
        change: angleServo by 5
      +
      if (c == 0) then
        change: angleServo by -5
      +
      if (angleServo == 180) then
        set: c to 0
      +
      if (angleServo == 0) then
        set: c to 1
      +
      pause (ms) 10
  clear
```

3. When an object is detected within the set range, the code will exit the loop that we created above and perform the following operations.

a) Assign the distance to the object to a variable named “objectDistance” and print this value on the OLED screen.

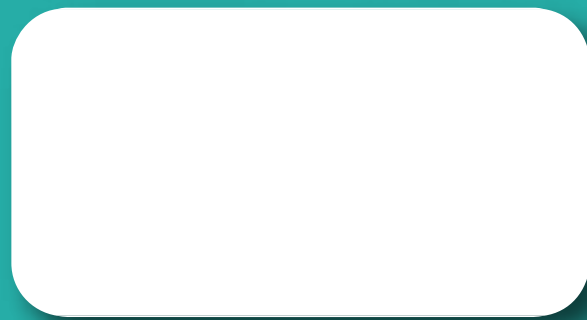
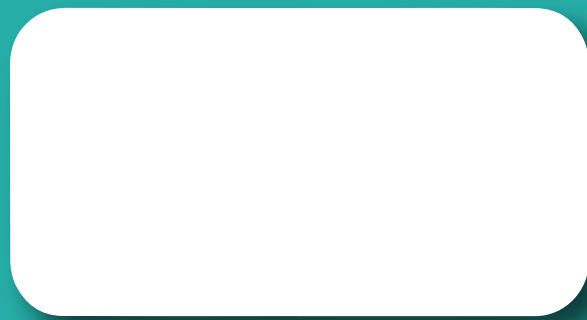
b) Print the angle between the radar and the object on the OLED screen.

c) Play the “Middle C” note from the buzzer.





# PicoBricks Logo Lamp



# PicoBricks Logo Lamp Project

In these days, the usage areas of 3D printers have significantly expanded. 3D printers are utilized for various purposes in many sectors such as healthcare, automotive, education, and more. The raw materials used by 3D printers for printing can vary depending on the intended use of the produced part. For instance, with a 3D printer that uses cement as a raw material, we can print a house. In this project, we will prepare a lamp by creating color animations using the 3D-printed PicoBricks Logo and the PicoBricks RGB LED module.

Color animations are used in various areas such as advertising panels, celebration areas, etc., to attract attention. In these systems, which are created by illuminating a LED with different colors at specific time intervals, RGB LEDs are commonly used. The main reason for the use of RGB LEDs in these systems is the ability to easily create desired color tones by utilizing color values ranging from 0 to 255.

## Project Details:

Project Link: <https://makecode.microbit.org/S00169-53176-41684-15510>



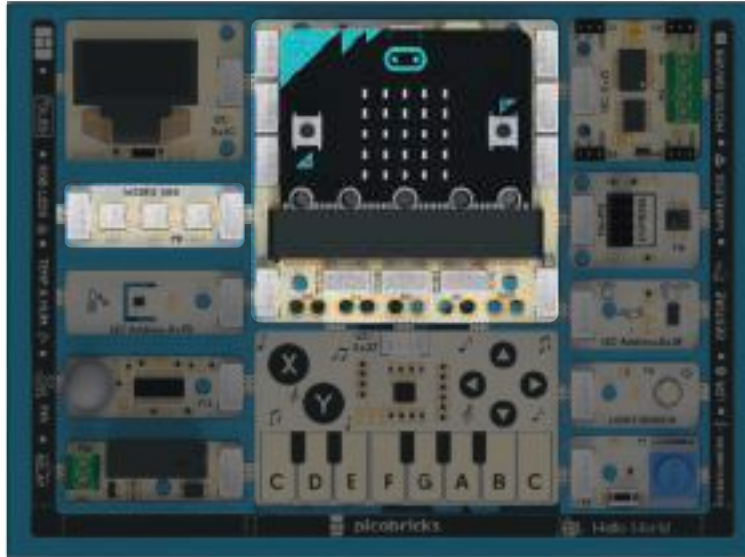
In this project, we will create color animations by placing the addressable RGB LEDs connected to the PicoBricks RGB LED module inside the 3D-printed PicoBricks Logo lamp.



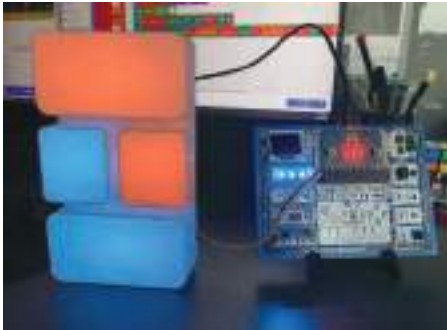


## ● Connection Diagram:

You can prepare this project by breaking down PicoBricks modules at proper points.



## ● Project Images:



## The STL Files of The Project:

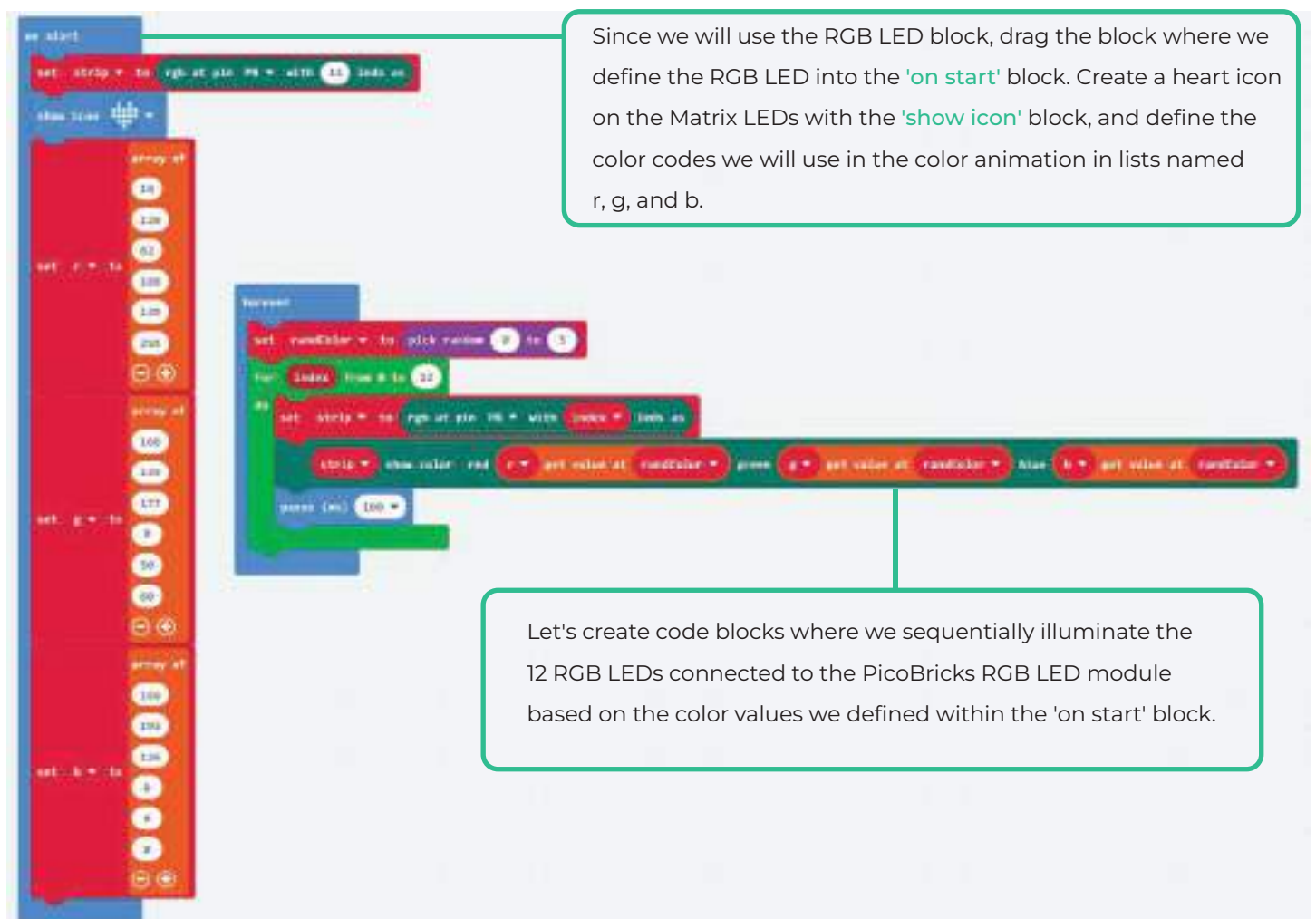
You can access the STL files of the project by scanning the QR code or opening the link in your browser.

<https://rbt.ist/thingiverse>



## MakeCode Code of The Project:

If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

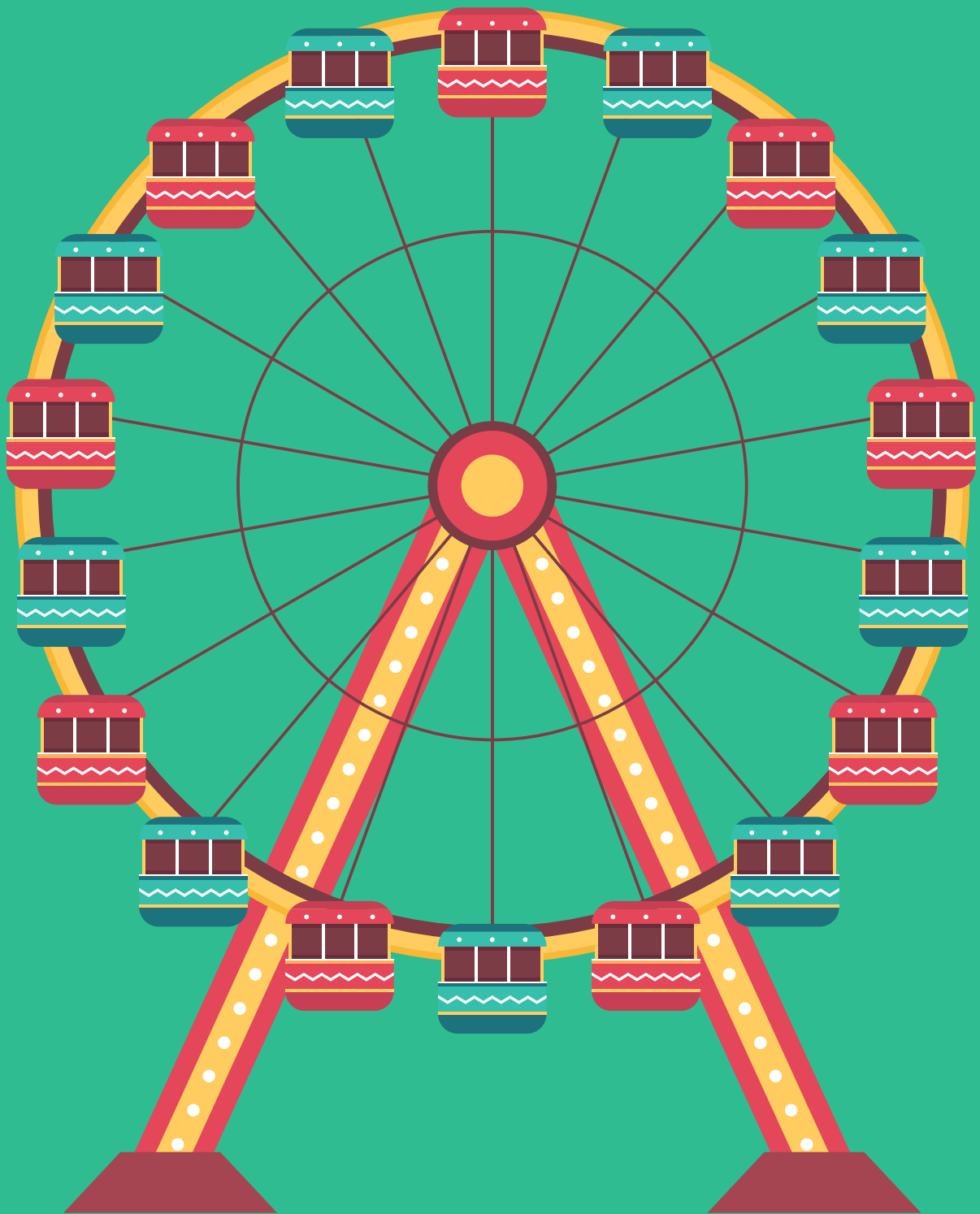


The image shows a screenshot of the MakeCode editor interface. On the left, there is a vertical stack of blocks under the 'on start' event. The first block is 'set strip to rgb at pin PB with leds as'. Below it are three 'array of' blocks, each containing a list of 12 numbers representing color values for red, green, and blue channels. The first array contains values: 10, 120, 52, 100, 120, 200, 100, 100, 100, 100, 100, 100. The second array contains: 100, 120, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100. The third array contains: 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100. On the right, there is a 'forever' loop block. Inside the loop, there is a 'set randomizer to pick random 0 to 5' block, followed by a 'for index from 0 to 32' block. Inside the 'for' loop, there is a 'set strip to rgb at pin PB with index leds as' block, followed by a 'strip show color red r get value at randomizer array g get value at randomizer array b get value at randomizer array' block. Below the 'for' loop is a 'pause (ms) 100' block.

Since we will use the RGB LED block, drag the block where we define the RGB LED into the 'on start' block. Create a heart icon on the Matrix LEDs with the 'show icon' block, and define the color codes we will use in the color animation in lists named r, g, and b.

Let's create code blocks where we sequentially illuminate the 12 RGB LEDs connected to the PicoBricks RGB LED module based on the color values we defined within the 'on start' block.

# Ferris Wheel



# Ferris Wheel Project

A Ferris wheel is an amusement ride where seats are positioned around a circle that rotates on a central axis. PicoBricks Ferris Wheel is a project kit where you can adjust the speed of the Ferris wheel based on the value of the potentiometer by using the potentiometer, motor driver, and mainboard module on PicoBricks.

## Project Details:

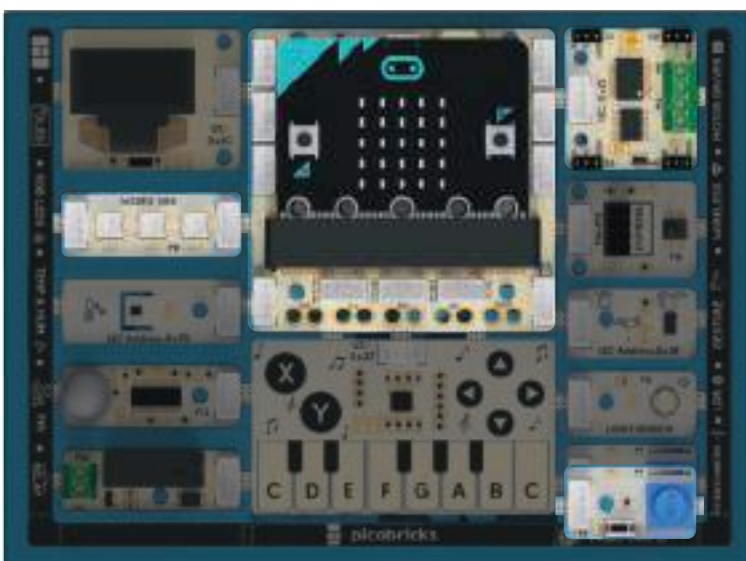
Project Link: <https://makecode.microbit.org/S46452-73338-57662-54218>



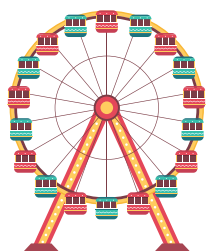
In this project, we will control the rotation speed of the Ferris Wheel based on the speed of the DC motor by using the PicoBricks Potentiometer module.

## Connection Diagram:

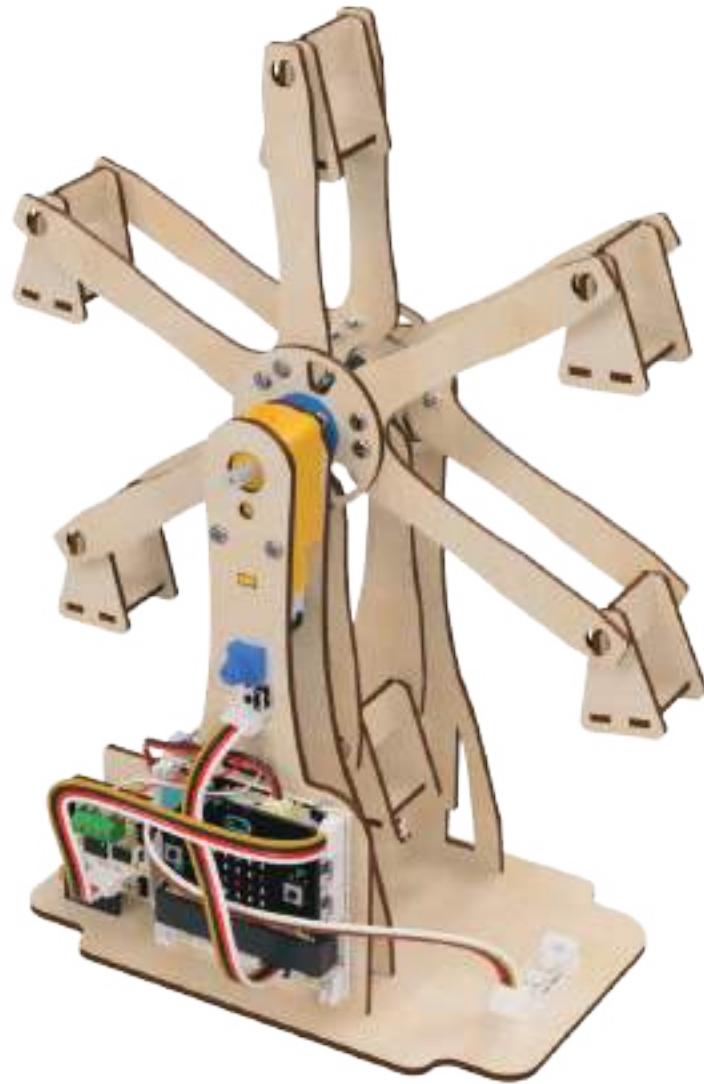
You can prepare this project by breaking down PicoBricks modules at proper points.



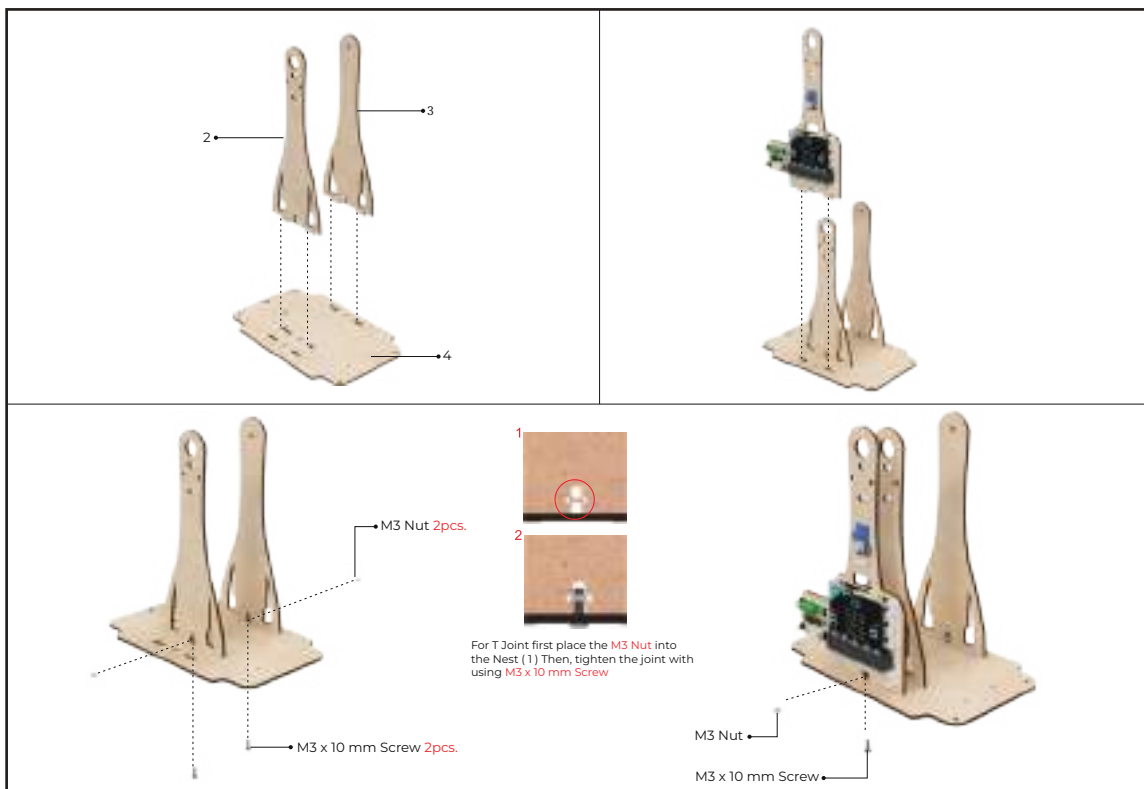
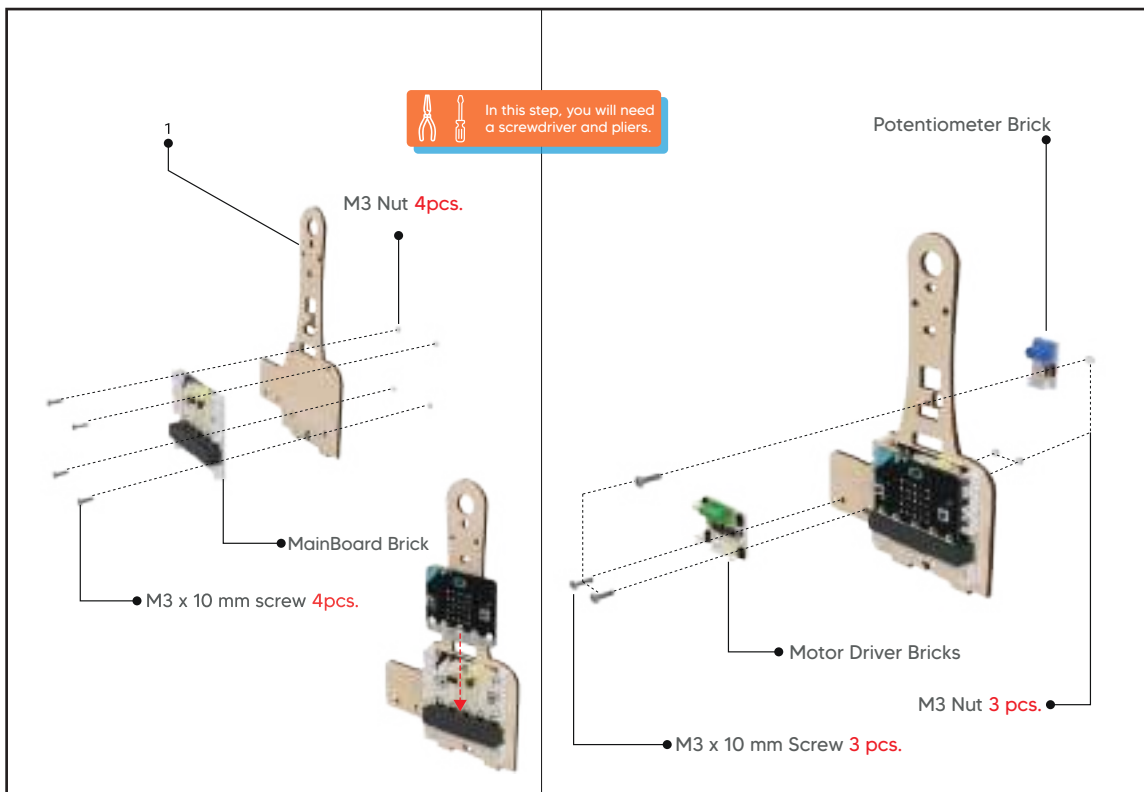
DC Motor

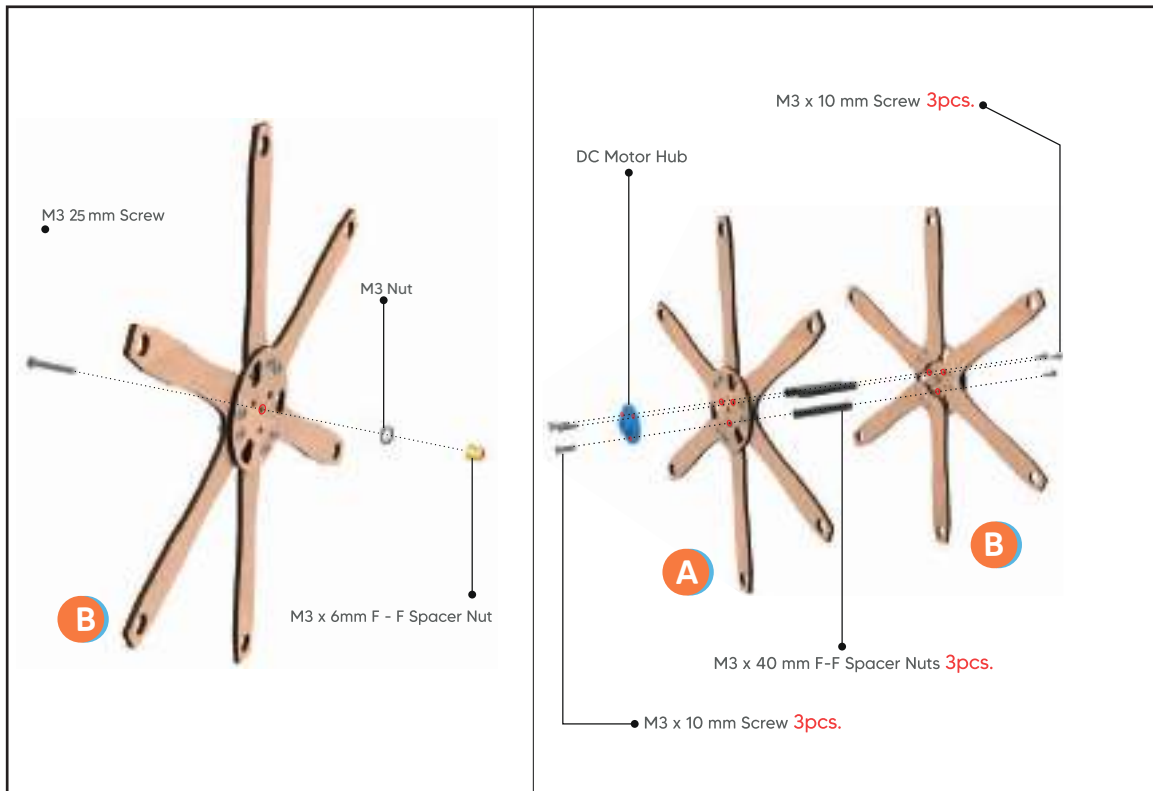
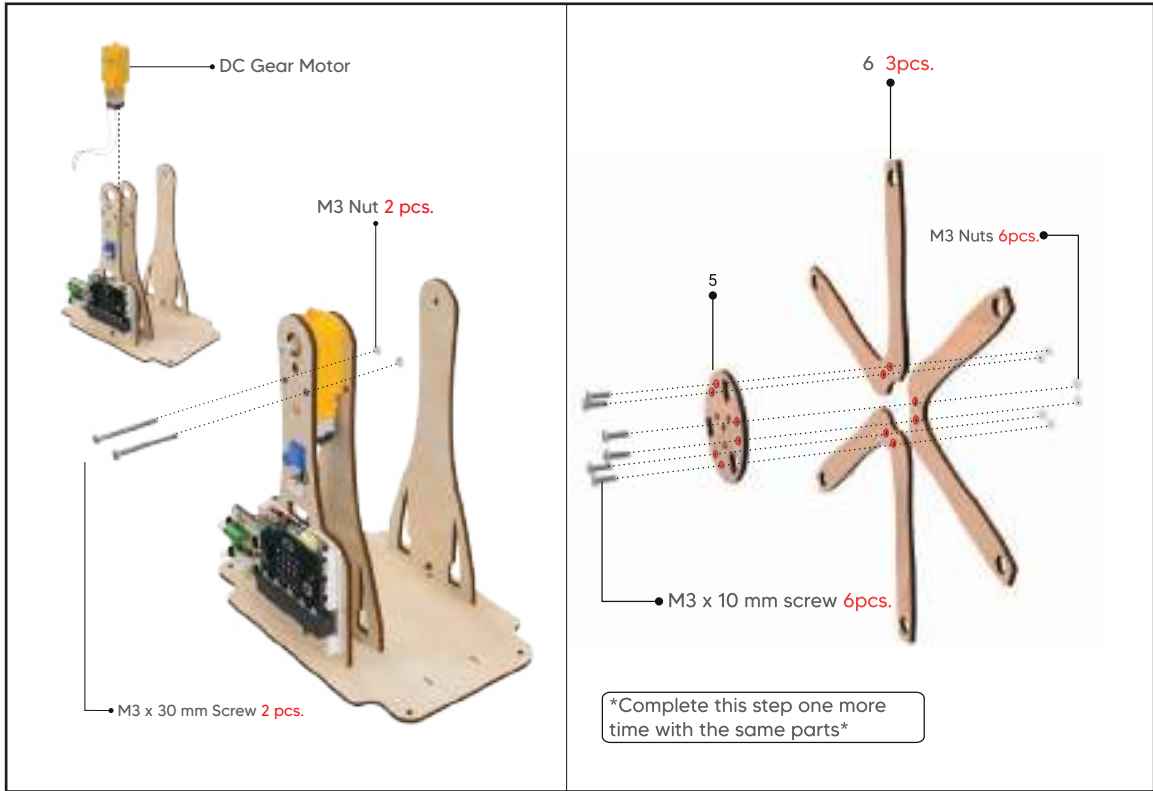


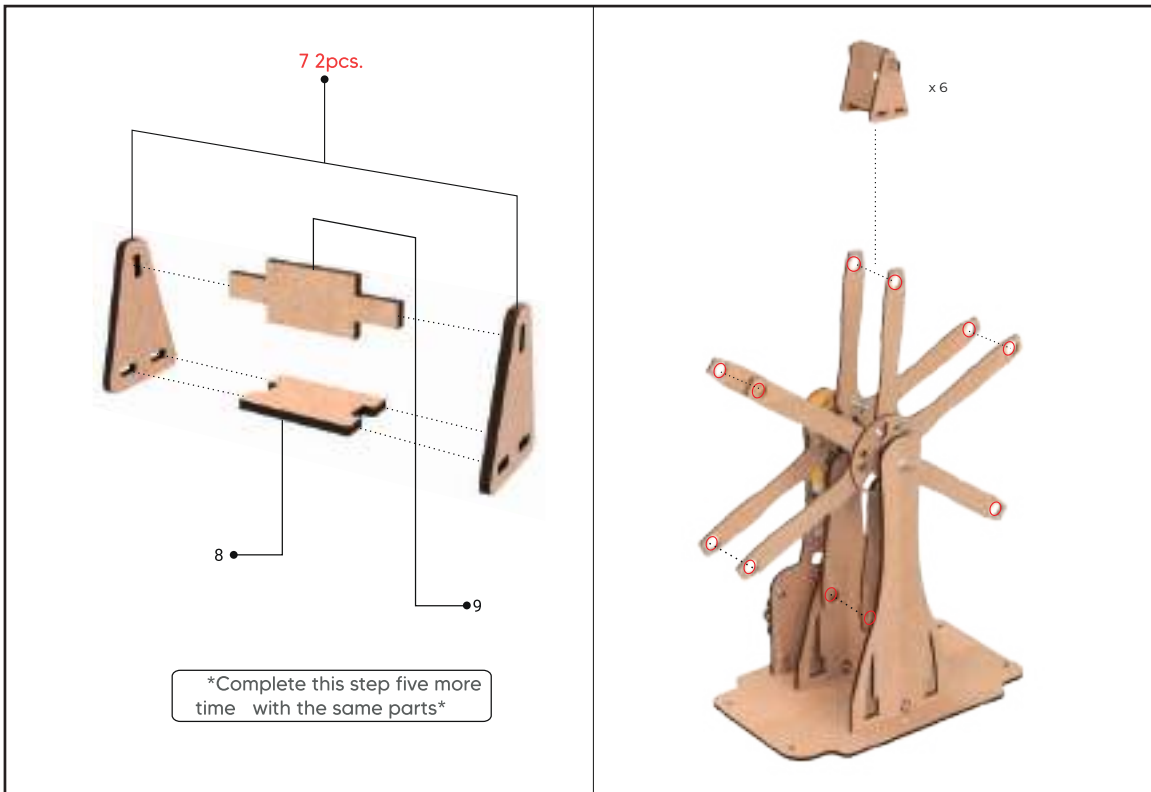
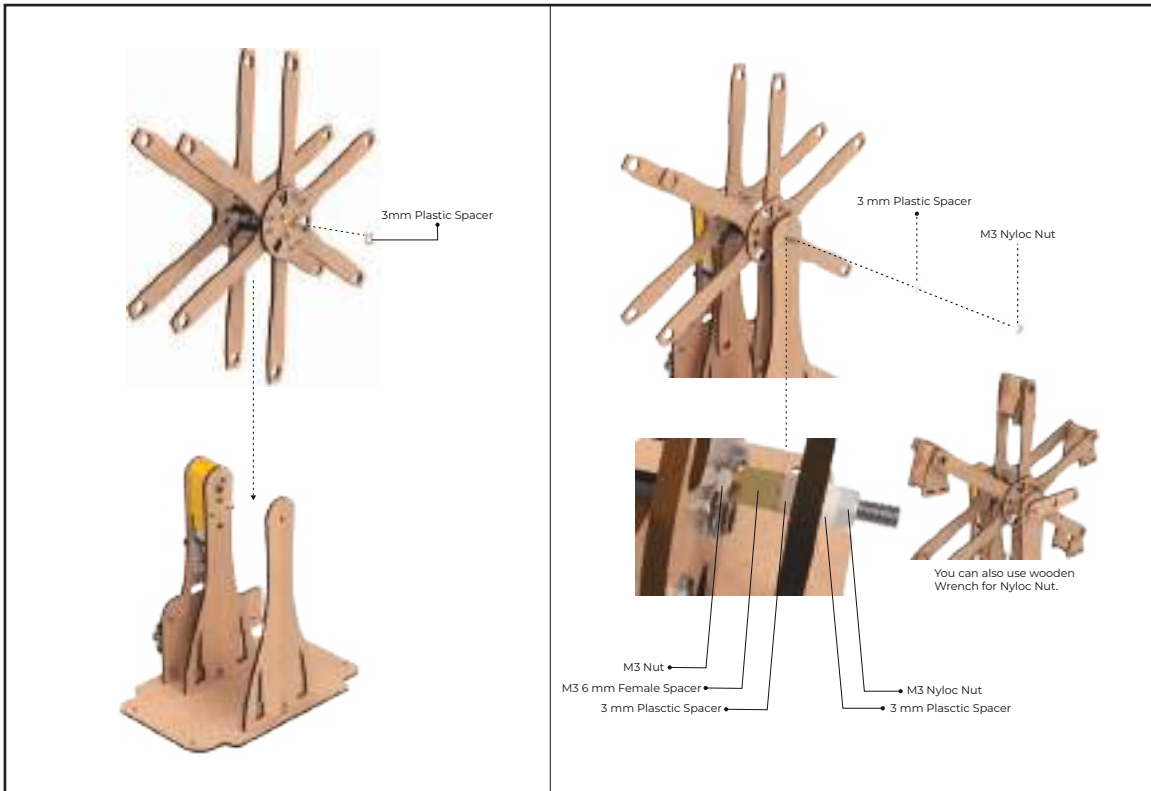
## ● Project Images:



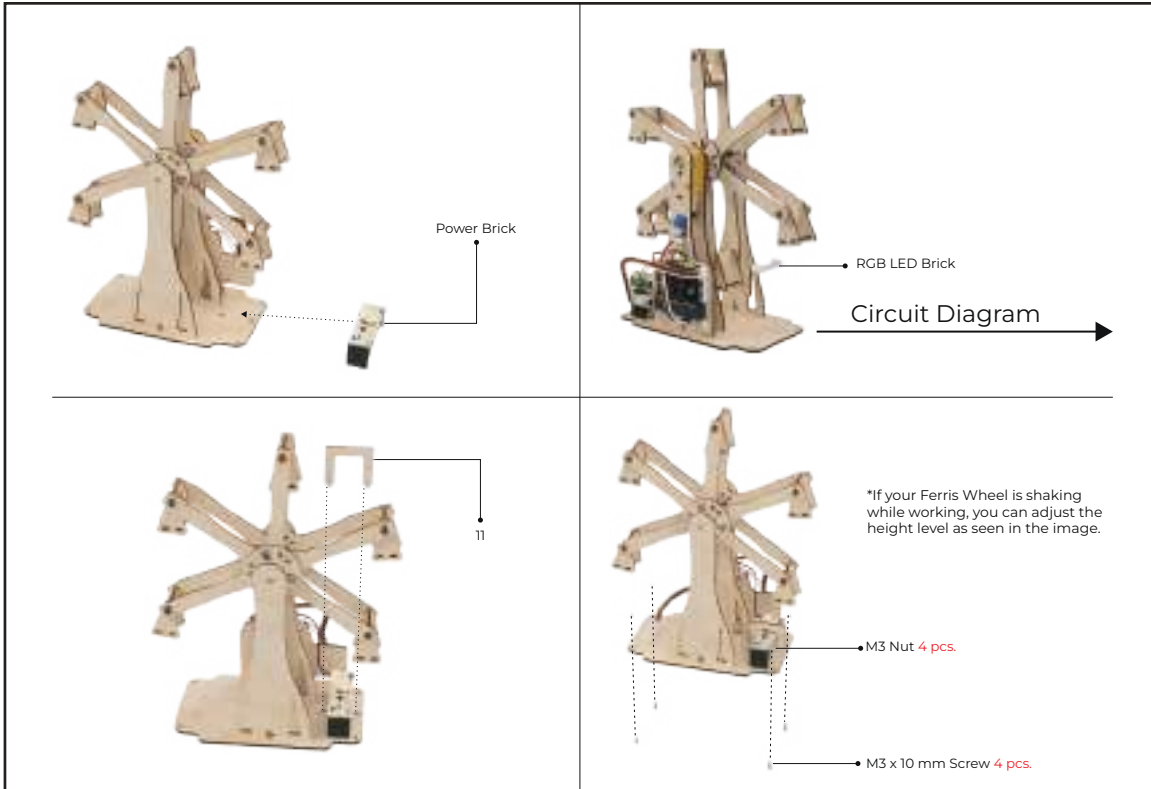
# Setup Steps of The Project:





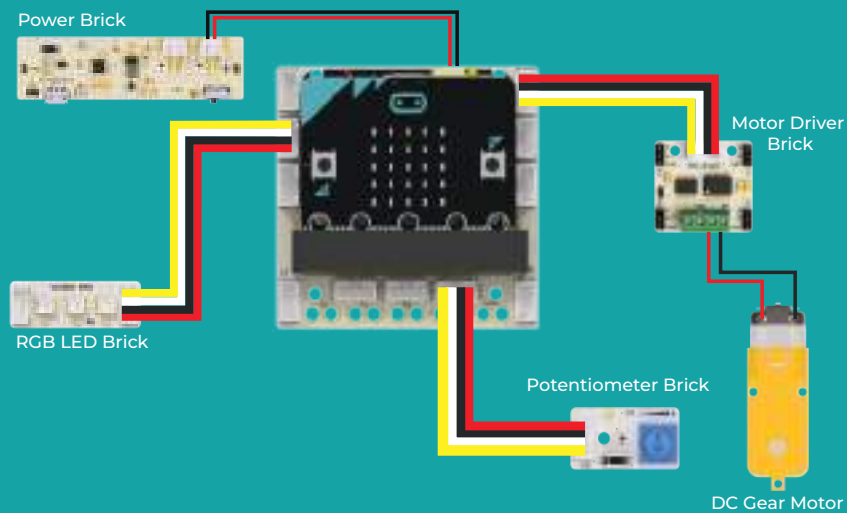






## Setting Up The Circuit

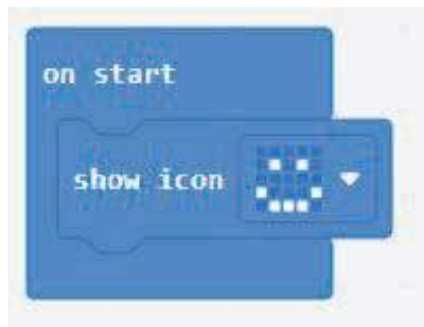
et to know the circuit elements of Mini Tank that we completed the setup and make the circuit setup coBricks modules.



## ● MakeCode Code of The Project:

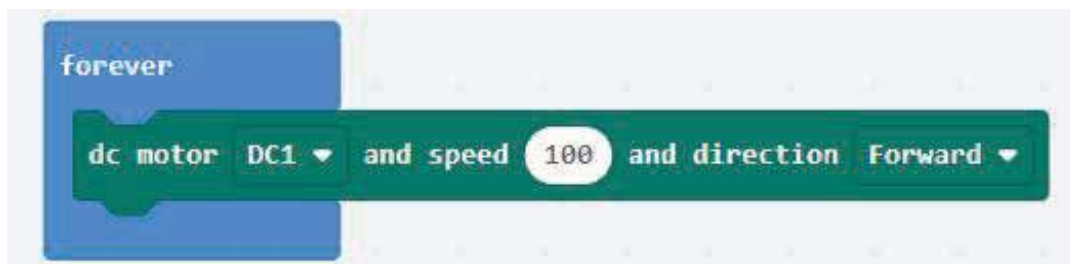
If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

1. Drag the "show icon" block from the "basic" blocks into the "on start" block, to display a smiley face emoji on the Micro:Bit matrix LED when we run our project.



2. When we run the project, follow the steps below to change the speed of the motor according to the value of the potentiometer.

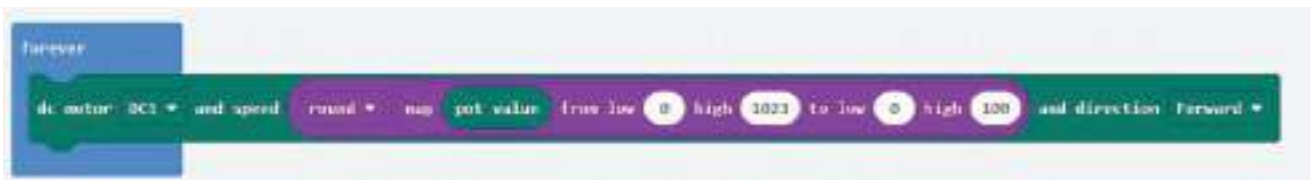
a) First, let's pay attention to which motor connector on the motor driver module we have connected the Ferris Wheel's motor to. Then, drag the DC motor block from the "PicoBricks" blocks into the "forever" block.



b) Let's change the value of the potentiometer from 0-1023 to 0-100.



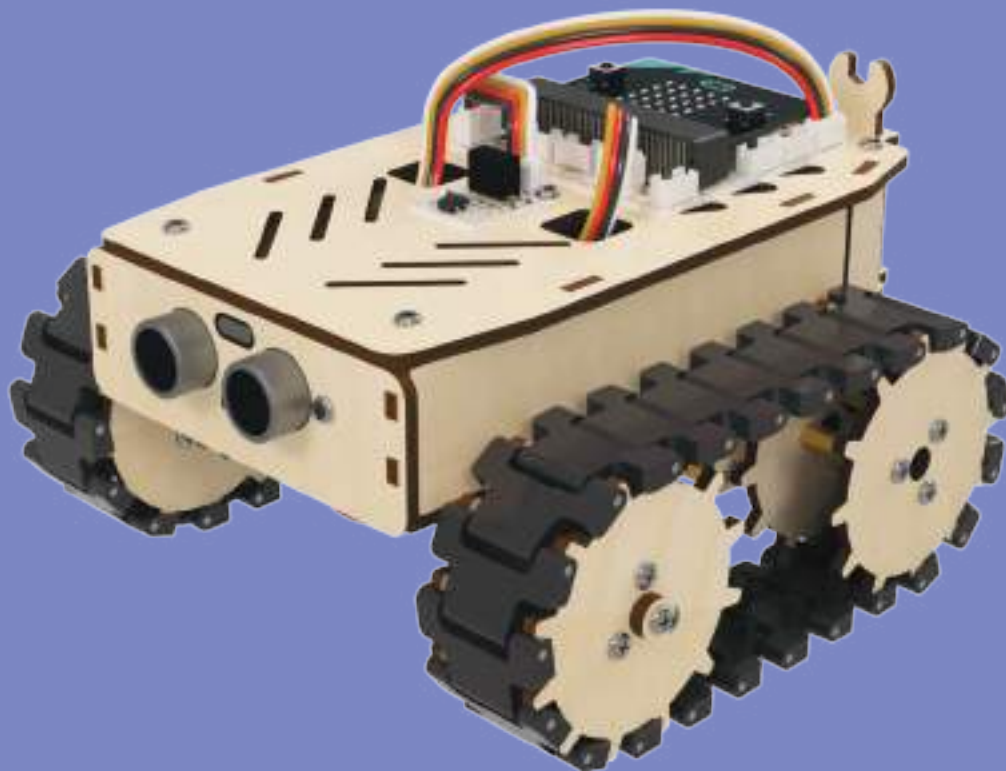
c) Let's drag the code block we created in the previous step into the speed field of the DC motor block.



d) The Code of The Project is Ready!



# Mars Explorer



# Mars Explorer Project

Tanks, with their tracked structures, are vehicles that can easily move on rough terrains. Tracks consist of multiple sequential wheels or rollers surrounded by a belt.

The PicoBricks Mars Explorer Car is a wooden project kit that utilizes two DC motors and a tracked platform. This robot car, controllable remotely with a remote controller thanks to the IR receiver, can decide on its movements by detecting surrounding objects through the front of its distance sensor.

## Project Details:

Project Link: <https://makecode.microbit.org/S42440-71908-13849-03682>

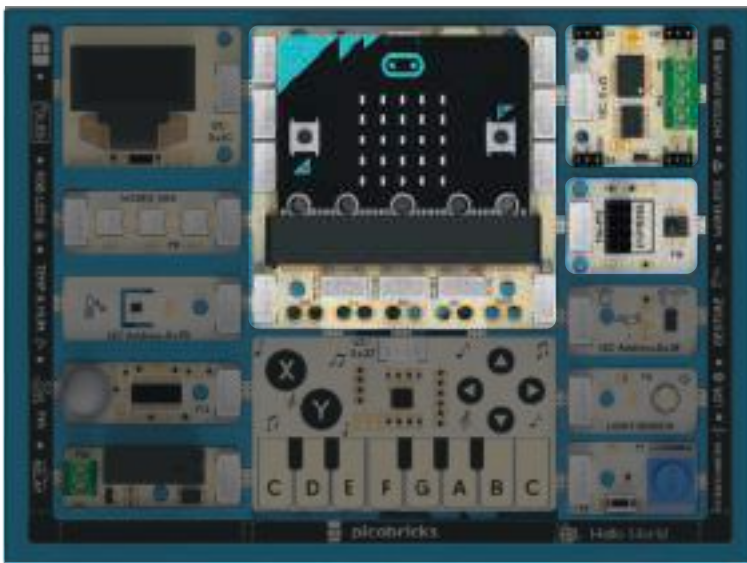


In this project, we will control two DC motors that connected to motor driver by using IR receiver on the PicoBricks wireless module with the remote controller. The robot car moves in the desired direction thanks to the DC motors. Additionally, if the HC-SR04 distance sensor on the robot car kit detects an object within 15 cm, the robot car will stop.



## ● Connection Diagram:

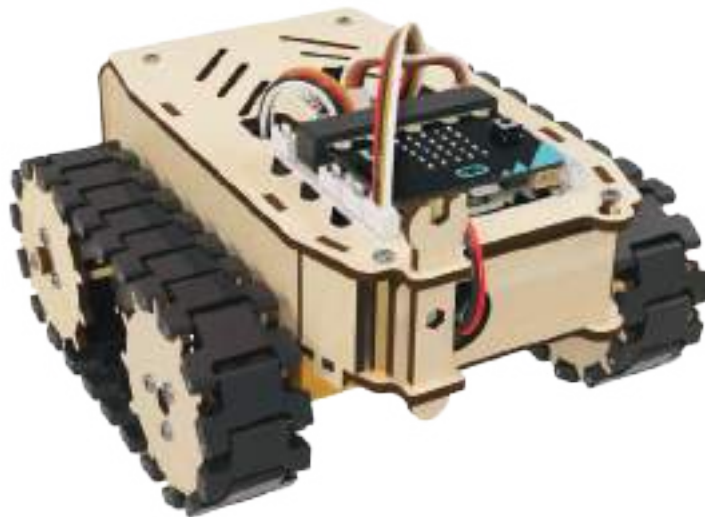
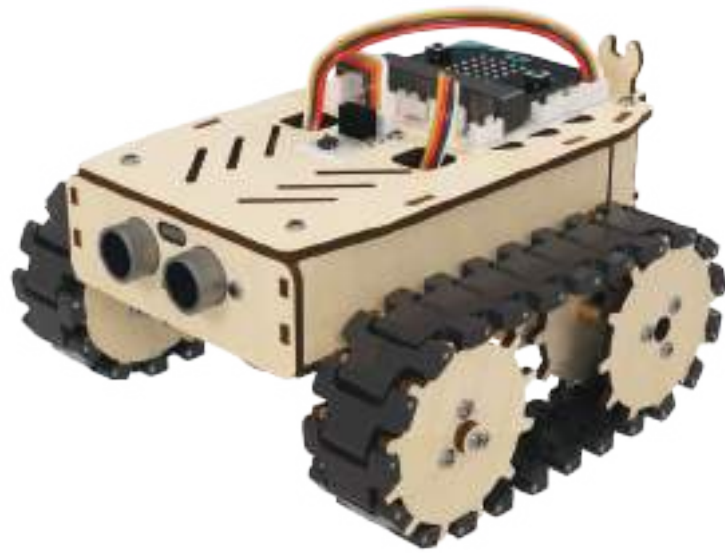
You can prepare this project by breaking PicoBricks modules at proper points.



x2 DC Motor



## ● Project Images:



# Setup Steps of The Project:

In this step, you will need a screwdriver and pliers.

① M3 x 10 mm Screw

M3 Nut

Apply the same steps for other side

1 For T Slot joint setup first, place the M3 Nut into the Nest (1) Then tighten the Joint with using M3 x 10 mm Screw.(2)

② M3 x 8 mm Screw 4pcs.

M3 x 30 mm Female Spacer

Make sure that this triangle marker is facing upwards.

③

3

④

4

⑤ Gear DC Motor

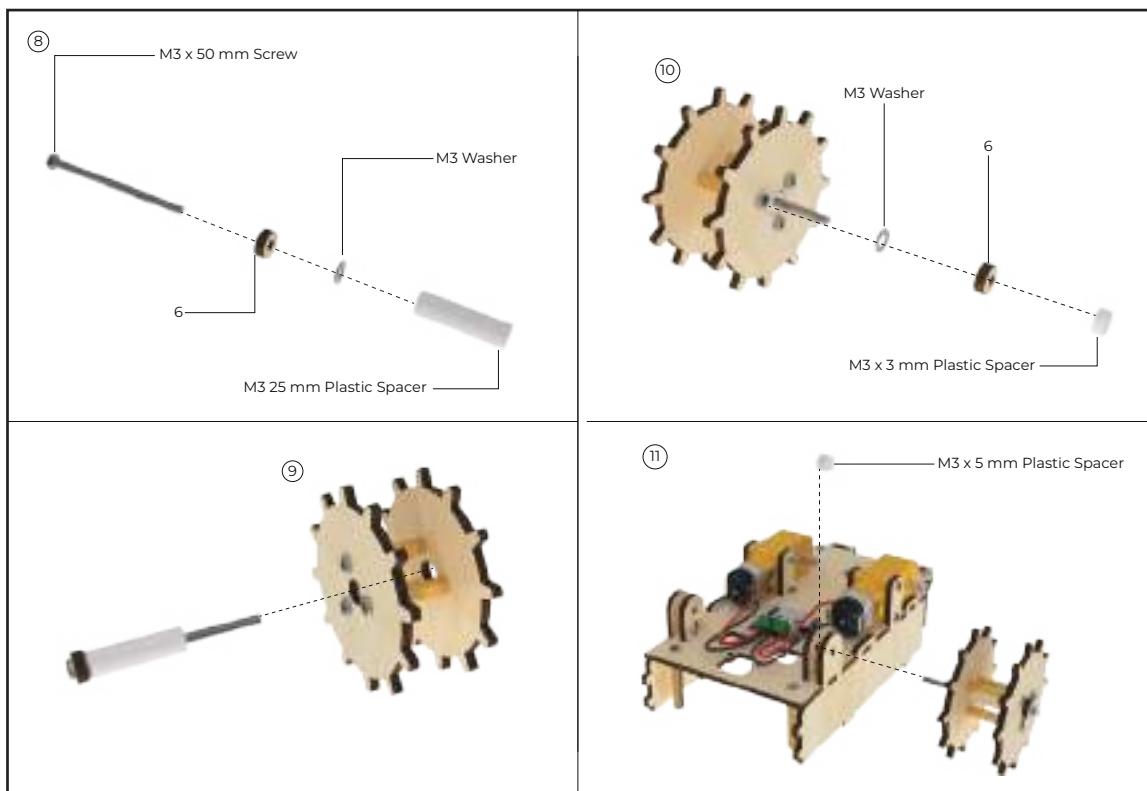
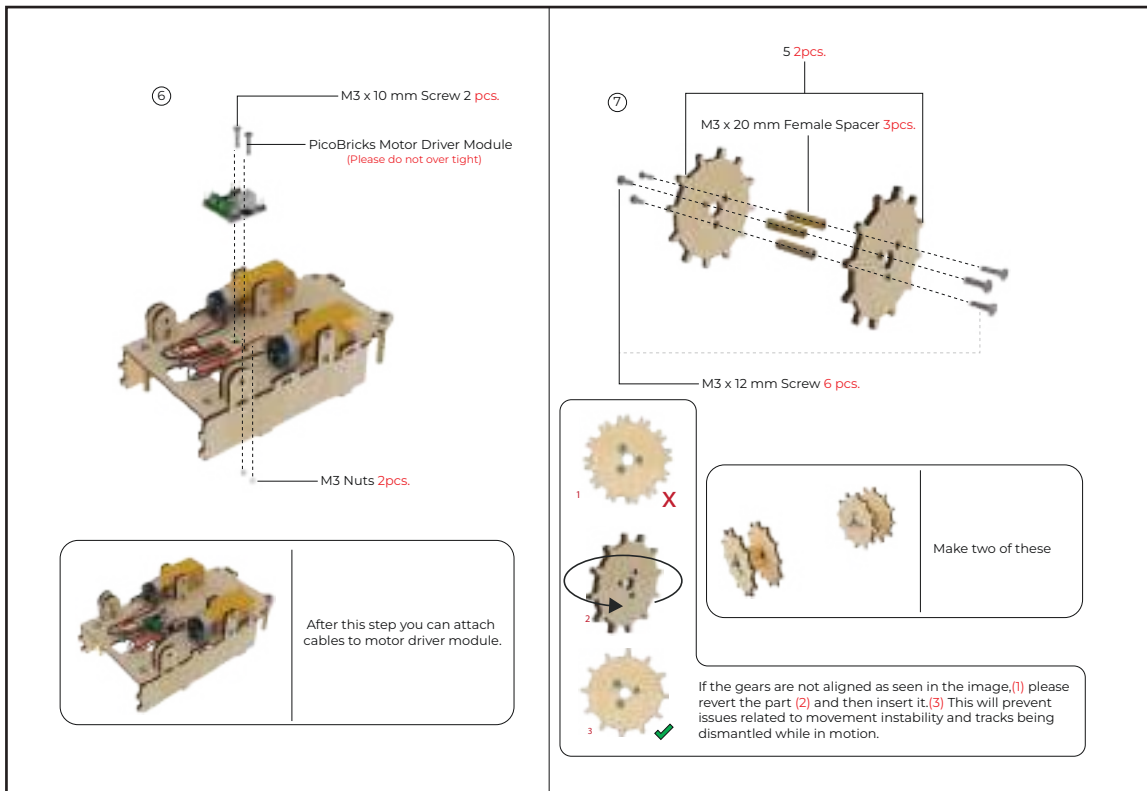
Wooden Wrench can be used to increase leverage

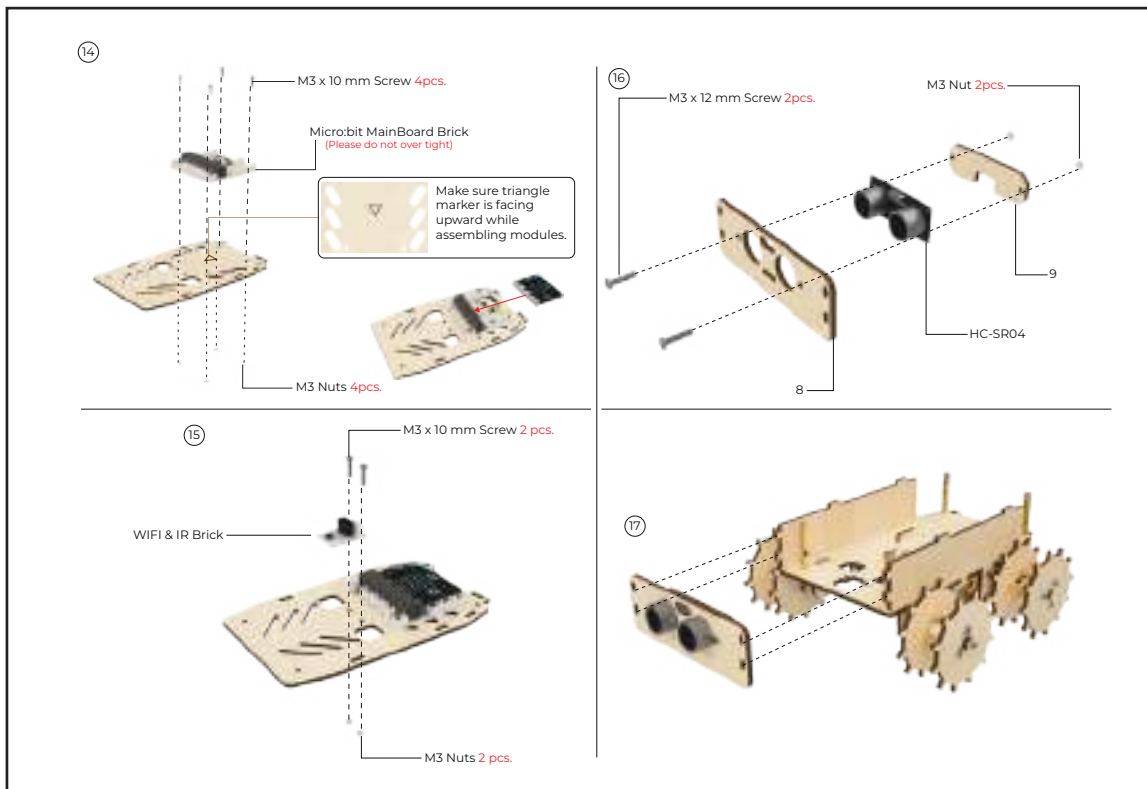
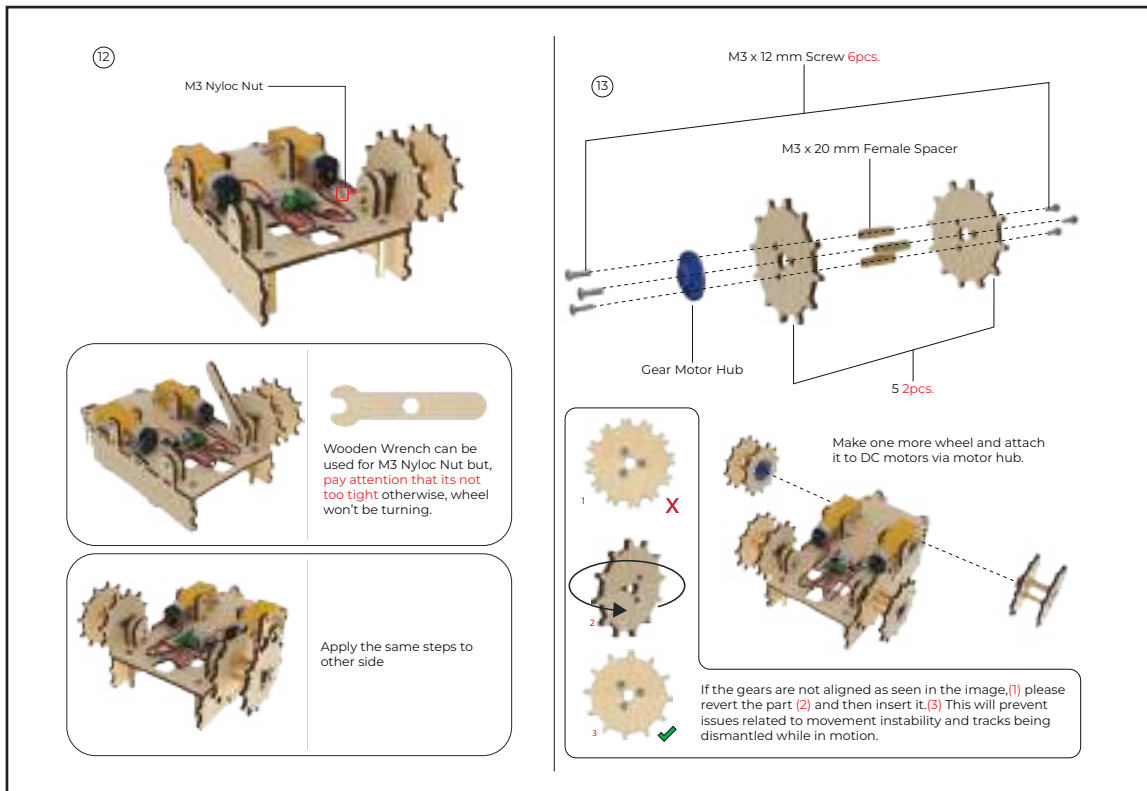
M3 Nuts 2 pcs.

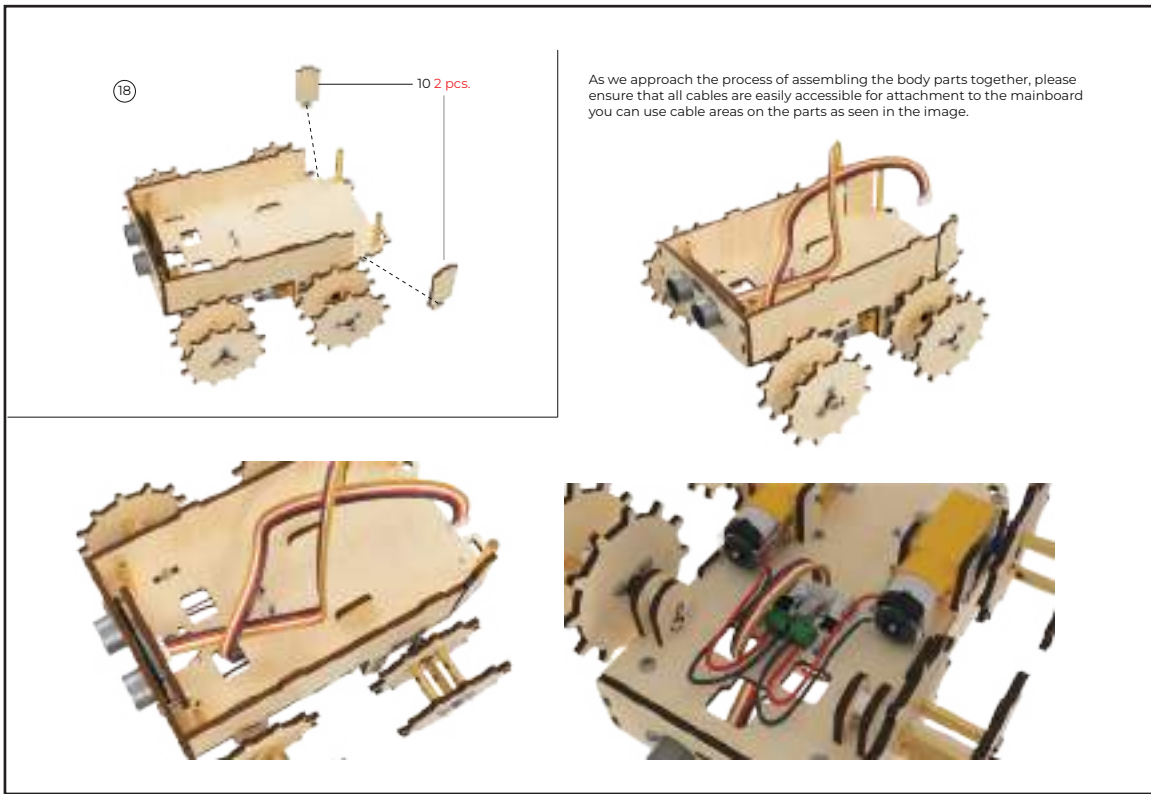
M3 x 30 mm Screw 2pcs.

Apply the same steps for other side



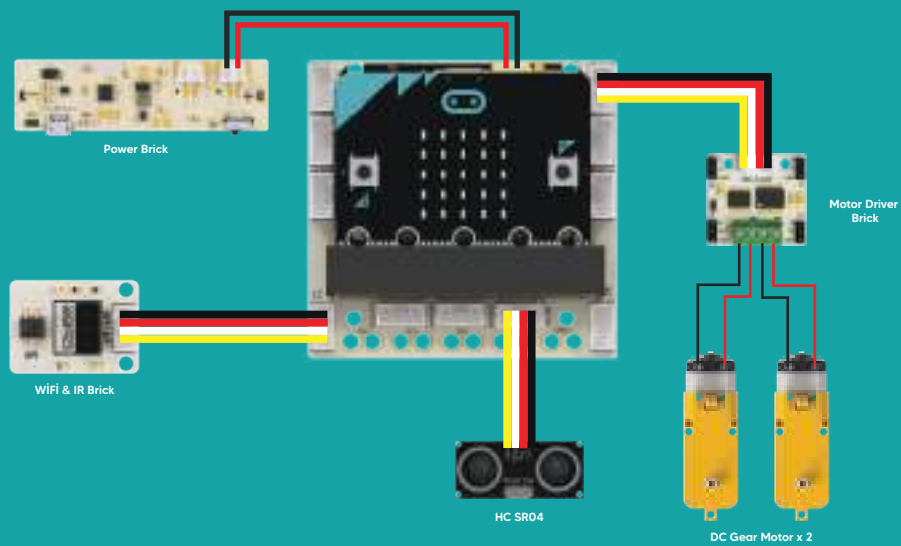


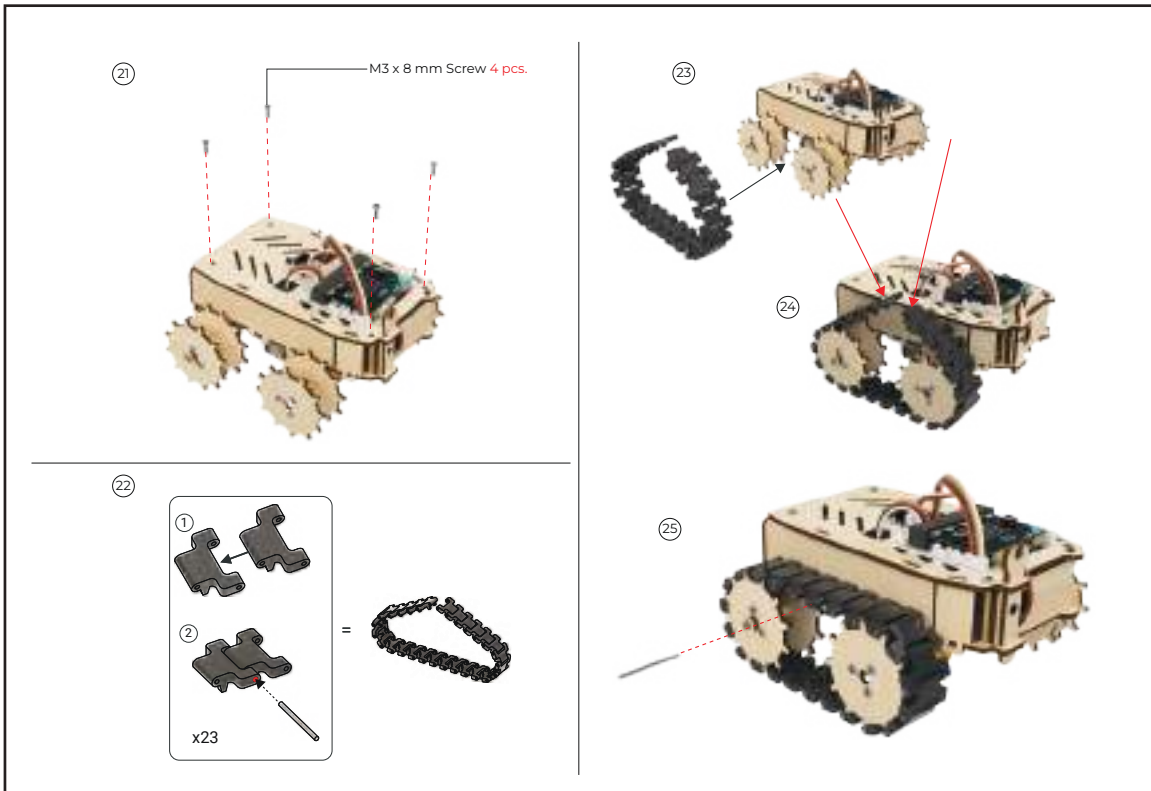
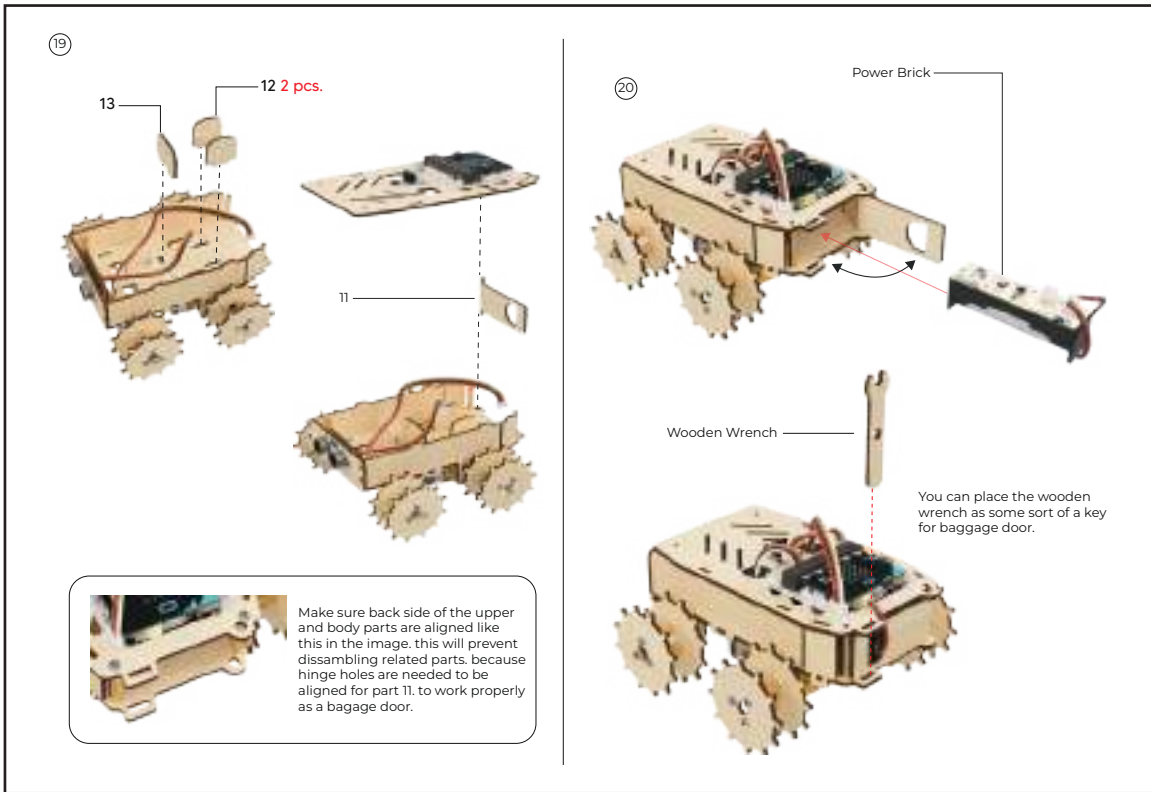




## Setting Up The Circuit

Let's get to know the circuit elements of Mini Tank that we completed the setup and make the circuit setup with PicoBricks modules.







Attach the tracks for the other side by following previous assembly steps.

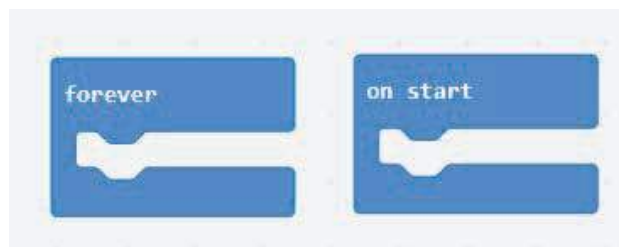


Assembly is finished and you can move on to coding steps.

## MakeCode Code of The Project:

If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

1. When we create the project, two blocks named "on start" and "forever" appear on the screen. The blocks dragged into the "on start" block are executed only once when the Micro:Bit is run. The "forever" block, on the other hand, continuously repeats the blocks dragged into it until the program is stopped. Since we will not use the "forever" block in this project, we can delete it.

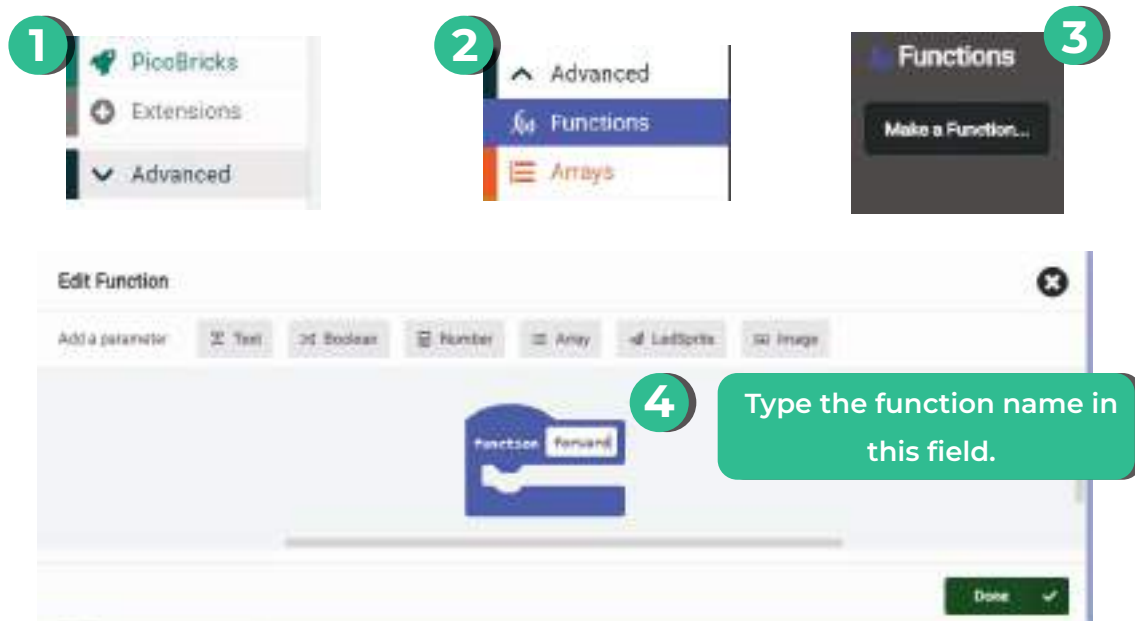


2. Firstly, let's define the functions that control the DC motors for the directions in which the robot car will go.

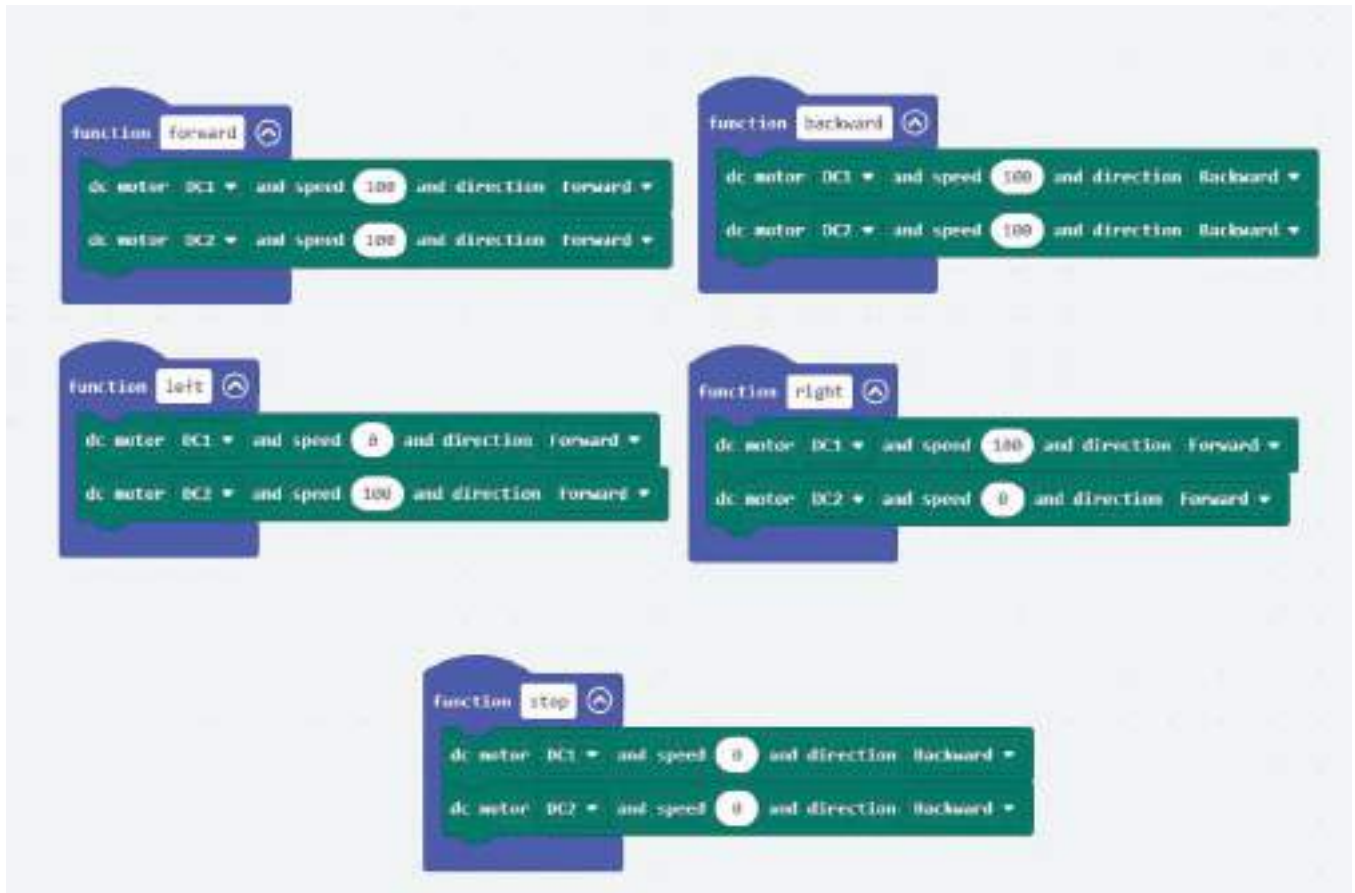


Defining these functions will make the code blocks we create for future projects with the tank set more readable.

a) To create a function with MakeCode, follow the steps below.



b) Let's create 5 functions named Forward, Backward, Right, Left, and Stop. Then, drag the DC Motor blocks from the "Motor" blocks in the "PicoBricks" extension into the function blocks as shown in the visual below.



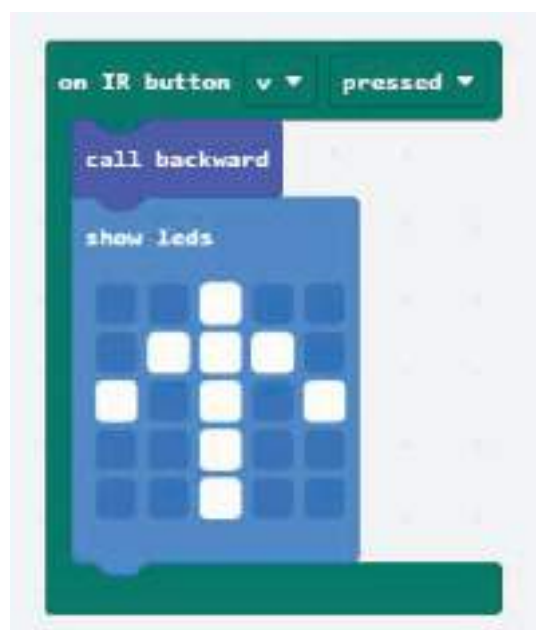
3. After defining the functions, drag the "connect IR received" into the "on start" blocks to be able to use IR blocks in the "PicoBricks" extension.



4. When the forward arrow button on the IR remote is pressed, if the value of the distance sensor is less than 15 cm, call the “stop” function. Otherwise, call the “forward” function and display a forward arrow symbol with matrix LEDs.

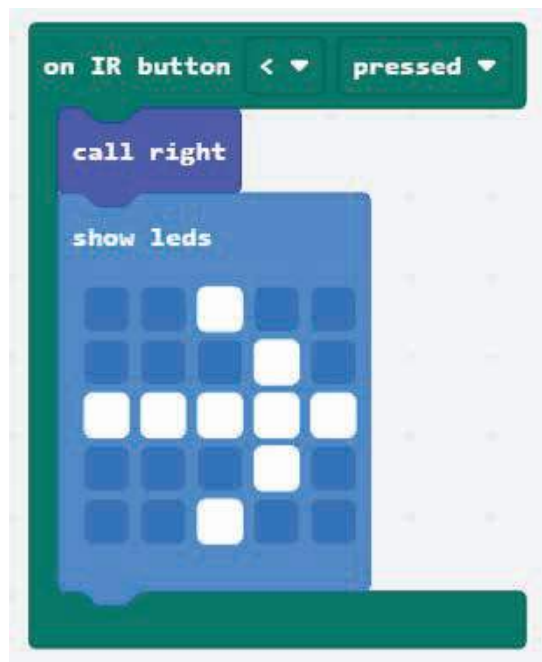


5. When the backward arrow button on the IR remote is pressed, call the "backward" function and display a backward arrow symbol with matrix LEDs.

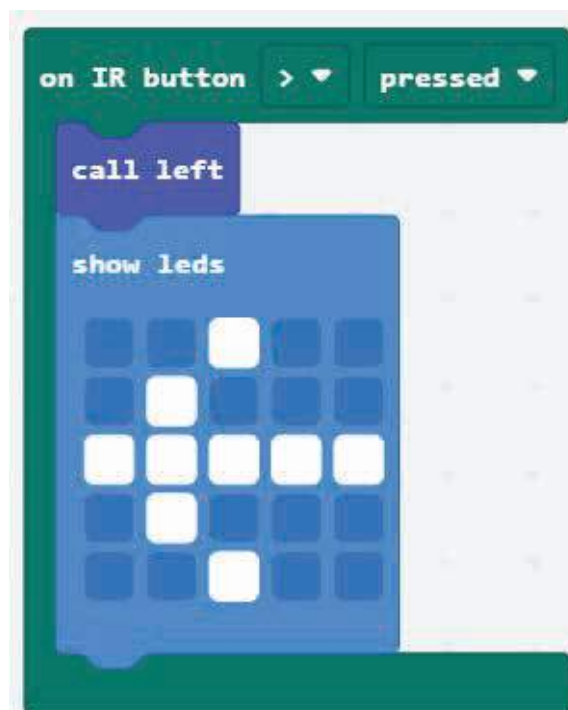




6. When the right arrow button is pressed, call the “right” function and display a right arrow symbol with matrix LEDs.



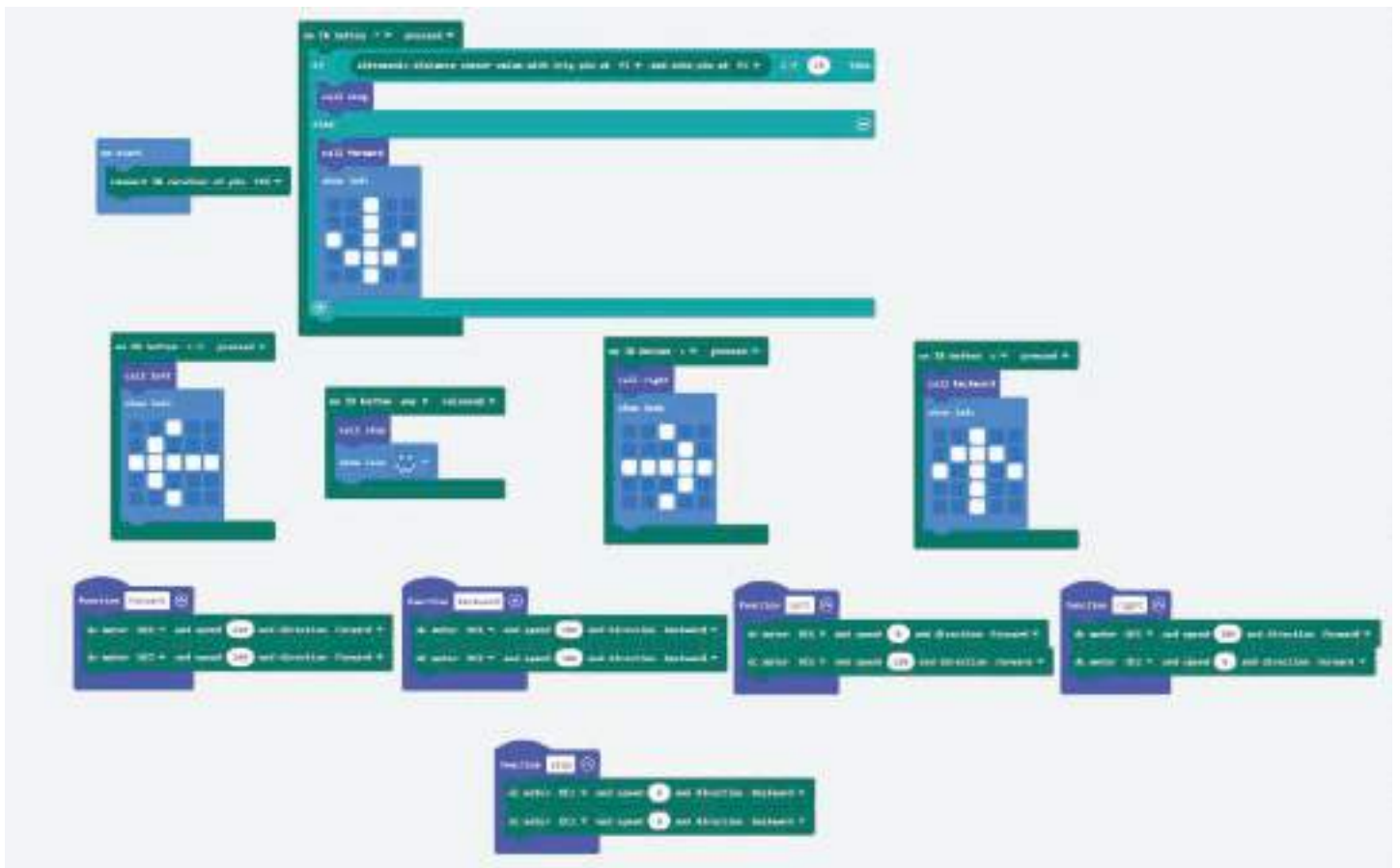
7. When the left arrow button on the IR remote is pressed, call the "left" function and display a left arrow symbol with matrix LEDs.



8. If none of the buttons on the IR remote are pressed, call the “stop” function and create a smiling face symbol with the LED matrix.



9. The Code of The Project is Ready!



# Trash Tech



# Trash Tech Project

The Trash Tech Kit is an educational robotics programming kit designed to gain environmental awareness in children.

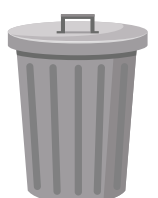
The Trash Tech is a fun kit that allows you to assemble the wooden pieces, sensors, and PicoBricks modules included in the set as specified in the installation guide. The goal of the project is to create an electronic trash bin that opens its lid by detecting objects using the HC-SR04 distance sensor located at its front.

## Project Details:

Project Link: <https://makecode.microbit.org/S38776-61015-05859-24913>

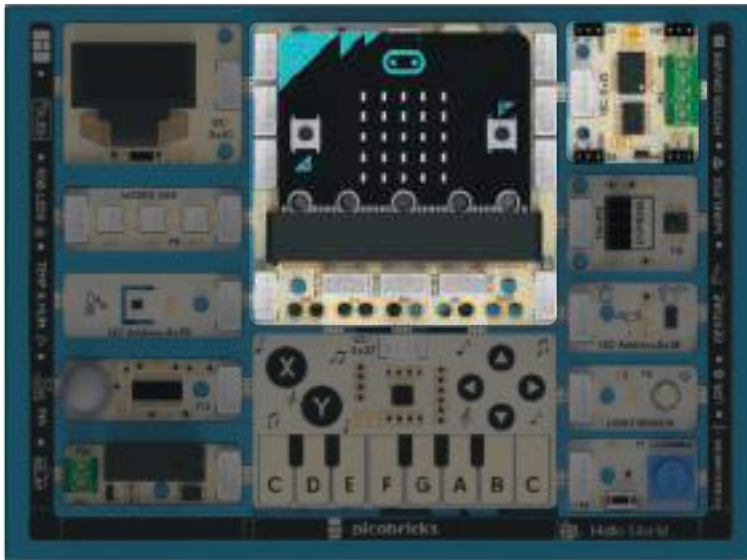


In this project, we detect the distance of our hand using the HCSR04 distance sensor and move the servo motor connected to the motor driver to the desired angle. This way, the lid of the trash bin opens.

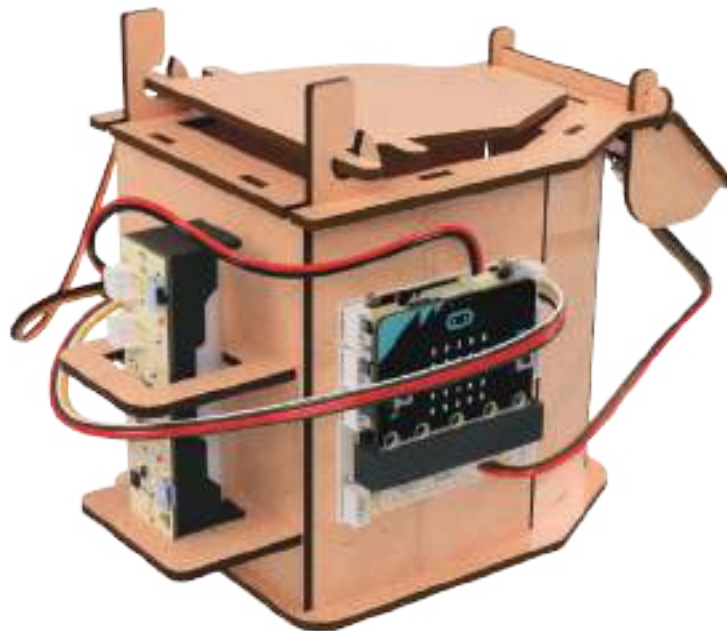
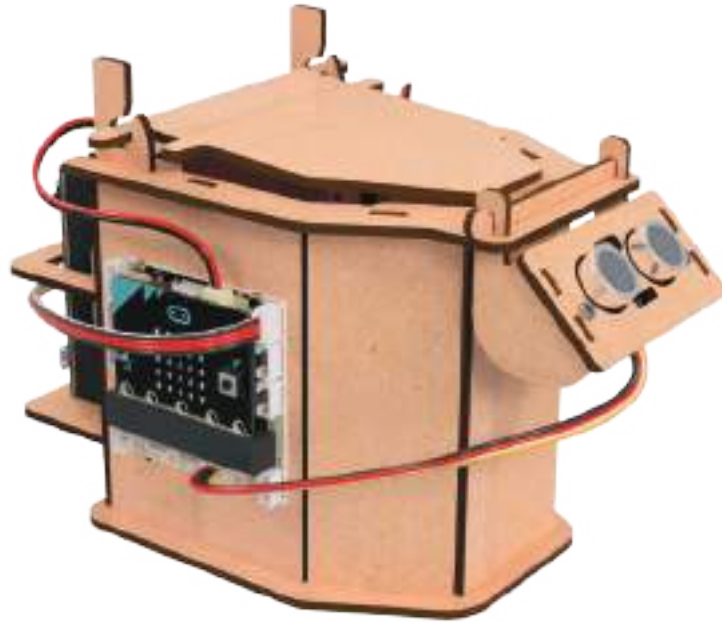


## ● Connection Diagram:

You can prepare this project by breaking the PicoBricks modules at suitable points.



## ● Project Images:



# Setup Steps of The Project:

## Servo Motor Calibration

Before starting the assembly, you have to manually calibrate the angles of the servo motors. Otherwise, Servo Motors won't be working properly.

Attach the servo horn to the servo motor (1)

Remove the servo horn from the servo motor (3)

Slowly turn the servo horn counterclockwise (6) until it is parallel with the servo motor, as seen in the image.(7)

Then slowly turn the servo horn clockwise until it stops. It is not a problem if the servo horn is not the same as the angle shown in the image above. The important thing here is that you have hit the last angle of the servo. (2)

Reattach (4) and reposition the servo horn perpendicular to the servo motor as shown. (5)

When this step is finished, it means that the servo motor is in the center position. It is important that you apply this process to other servo motors in the set. After processing the other motors, remove the servo horn and set aside for assembly.

In this step, you will need a screwdriver and pliers.

Pay attention to the direction of the Servo Motor.

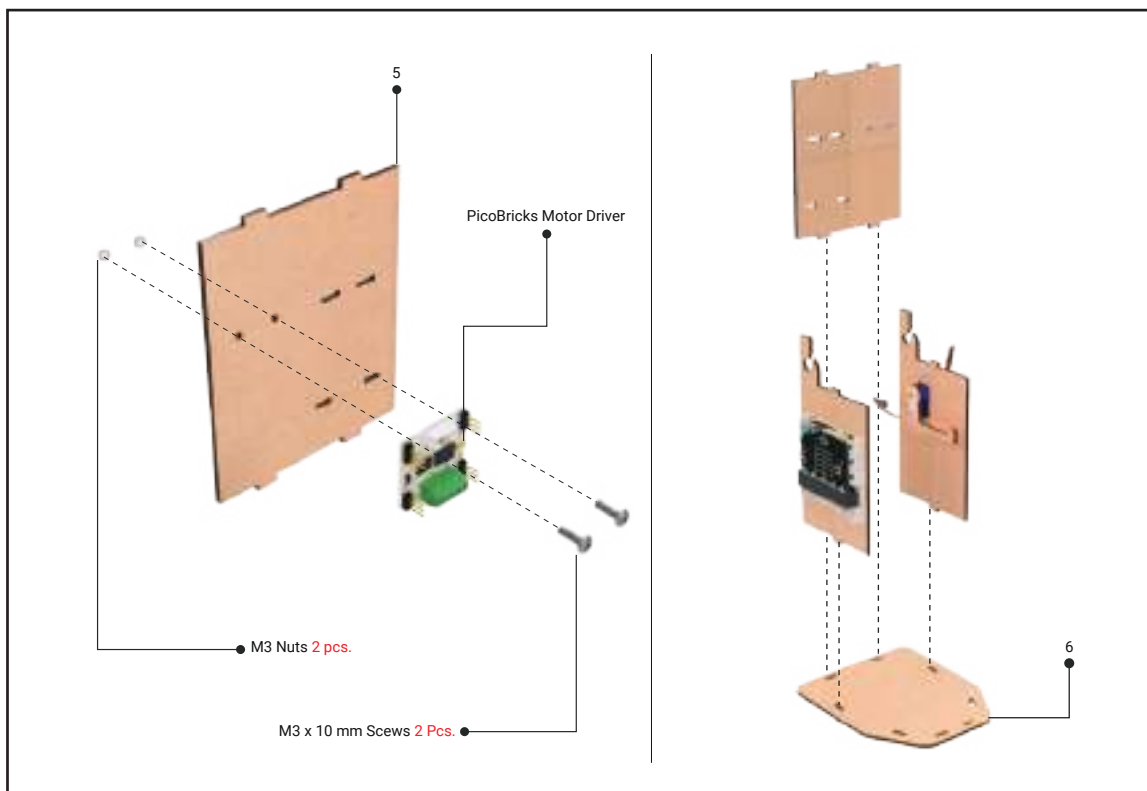
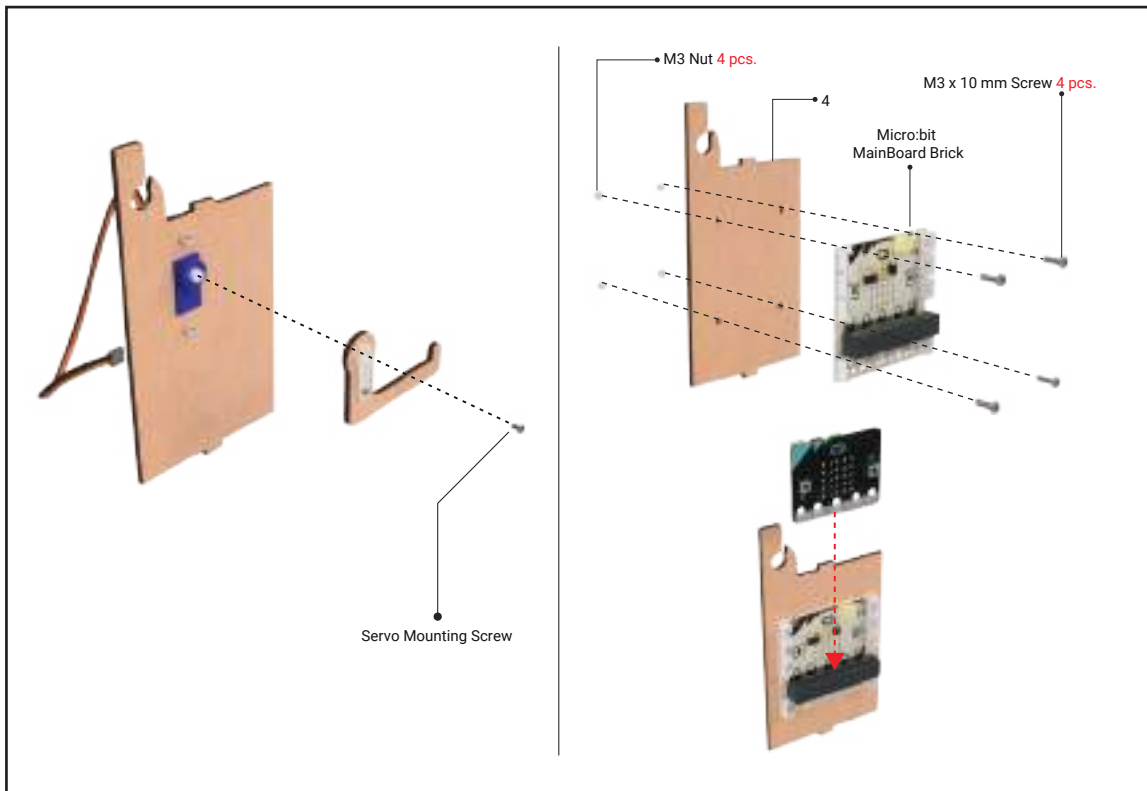
M3 Nuts 2 Pcs.

SG-90 Servo Motor

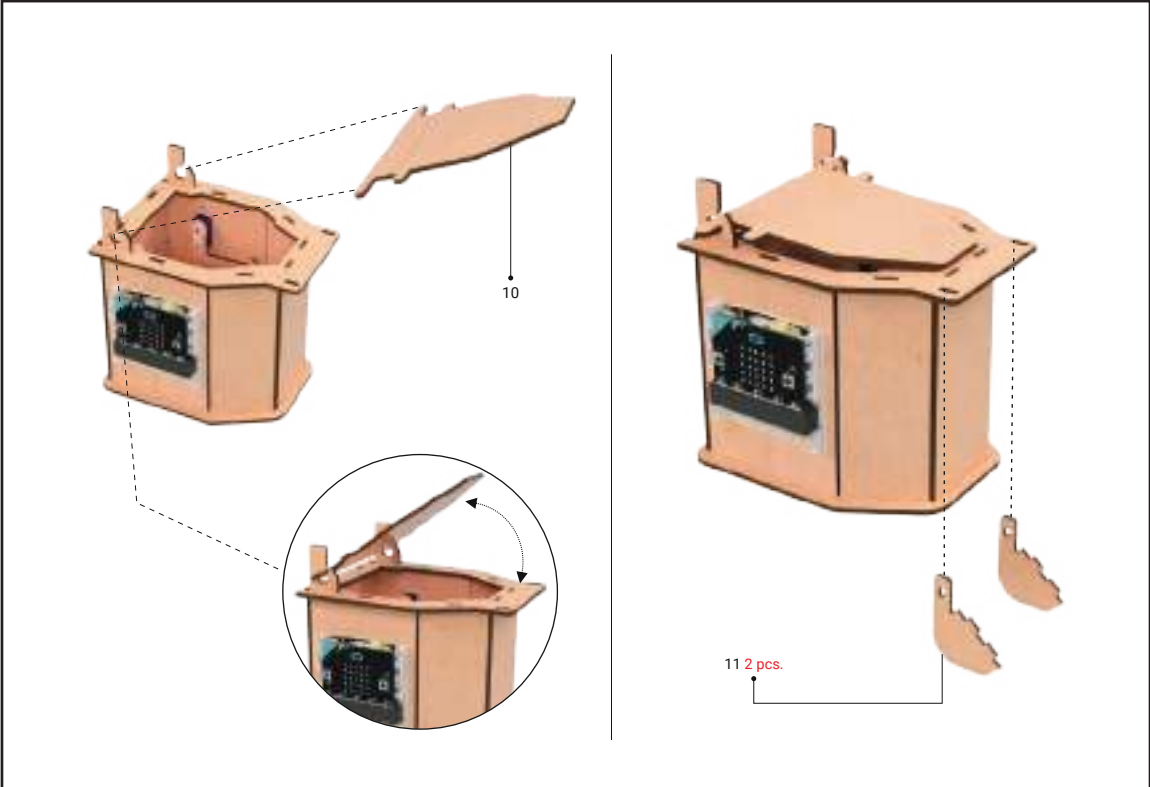
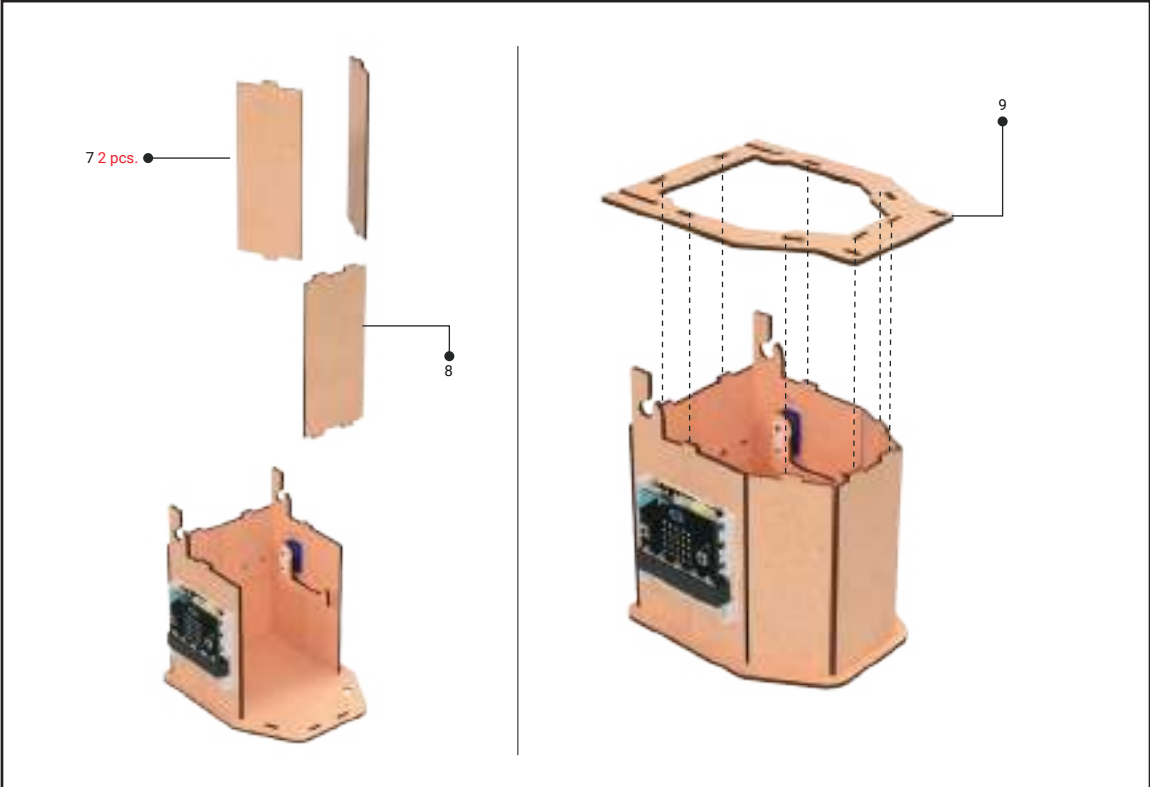
M3 x 12 mm Screws 2 Pcs.

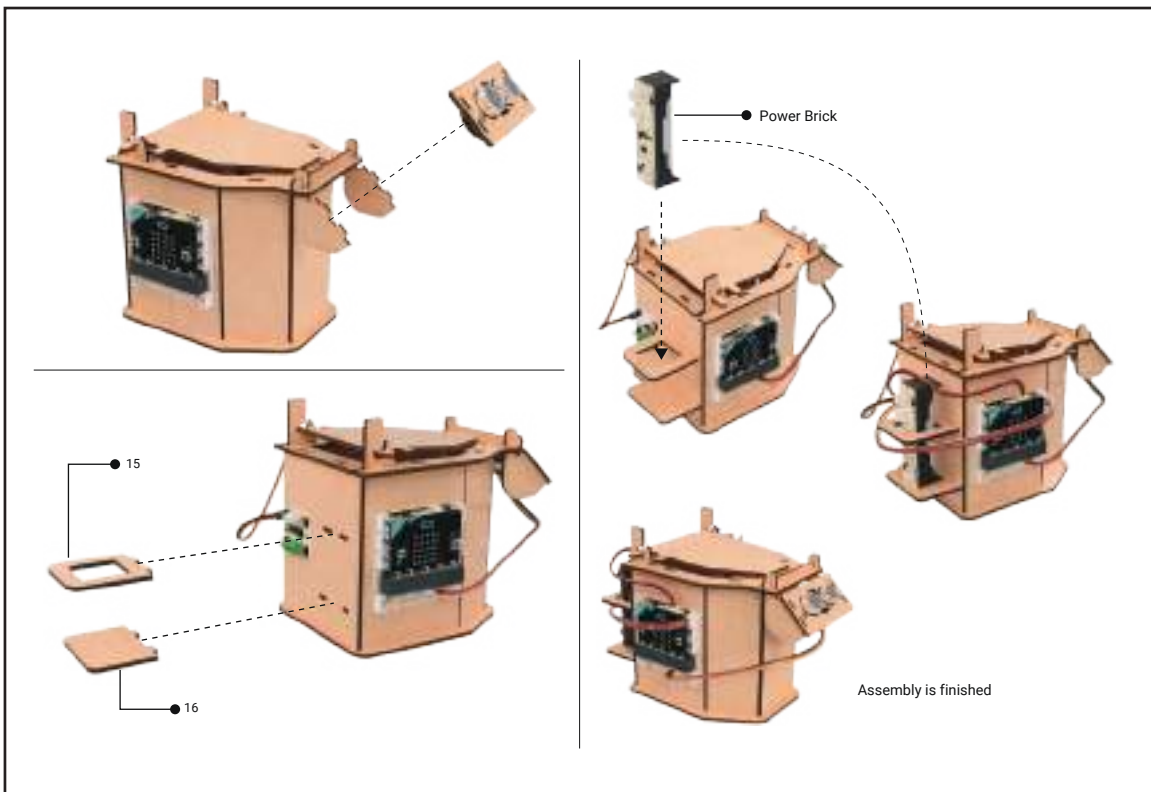
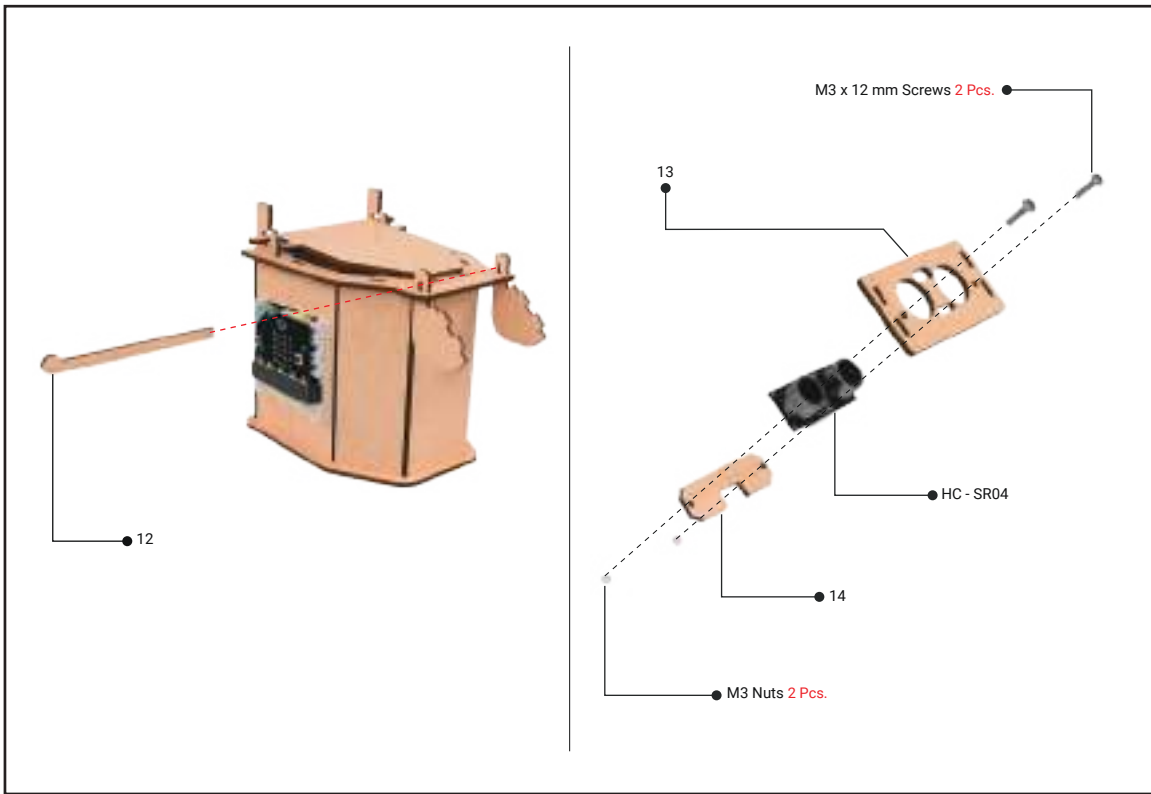
Servo Horn

Servo Horn Screw



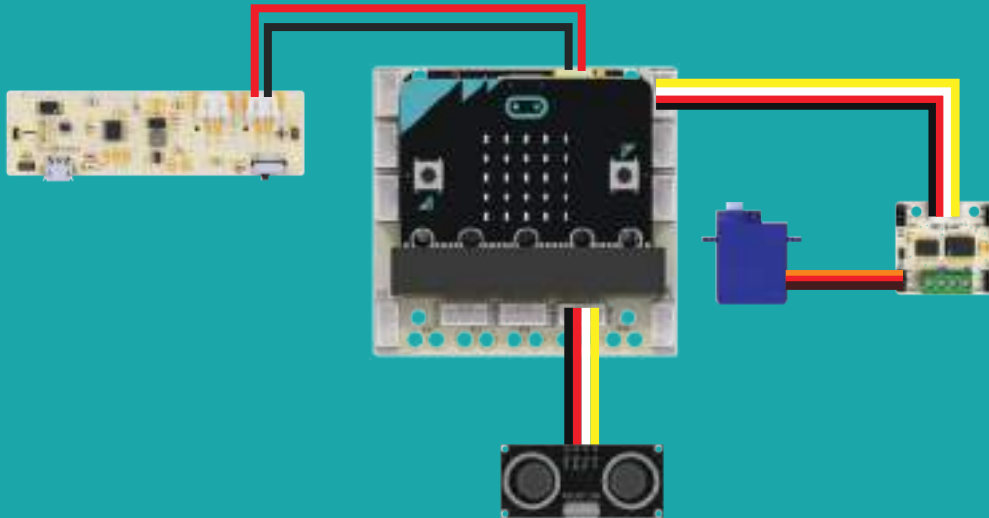






## Unplugged 4: Setup of Circuit

Let's get to know the circuit elements of the smart trash bin that we have completed and set up the circuit with PicoBricks modules.



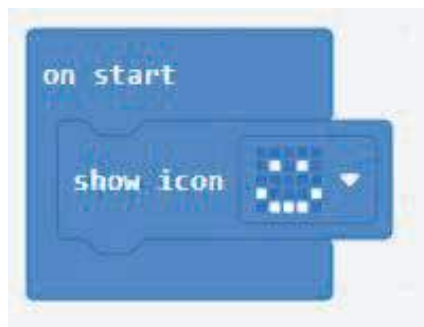
## ● MakeCode Code of The Project:

If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

1. When we create the project, two blocks named "on start" and "forever" appear on the screen. The "on start" block runs the dragged blocks inside it only once when the Micro:Bit is executed. The "forever" block, on the other hand, repeats the dragged blocks inside it continuously until the program is stopped.



2. When the project starts, drag the "show icon" block from the "Basic" blocks to create a smiley face on the Matrix LEDs of the Micro:Bit.



3. Drag the operations that we want to repeat continuously when the project starts into the "forever" block.

a) When the distance sensor detects a value between 1 and 9, use the "show icon" block to create a ( ) icon on the Matrix LEDs.

b) Move the servo motor to 90 degrees to open the lid of the trash bin.

c) Wait for 200 milliseconds, then move the servo motor to 180 degrees to close the lid of the trash bin.

d) Wait for 2 seconds.

e) If the distance sensor does not detect a value between 1 and 9, move the servo motor to 180 degrees and create a smiley face icon on the Matrix LEDs by using the "show icon" block.



d) The Code of The Project is Ready!



# Money Box



# Money Box Project

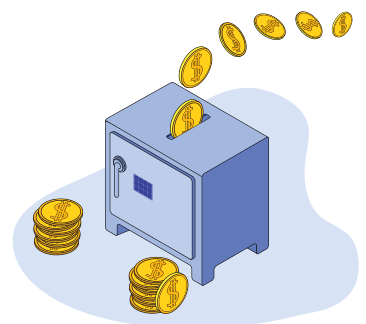
PicoBricks Money Box can detect objects placed in its receptacle through distance sensor in the front of it and automatically lifts its receptacle to take in these objects.

## Project Details:

Project Link: <https://makecode.microbit.org/S09244-63172-64381-78563>

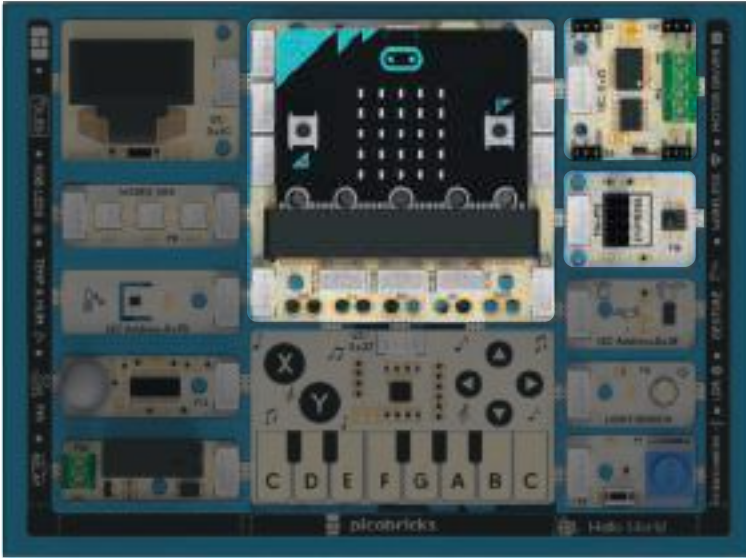


In this project, when the distance sensor detects the object placed in the receptacle, the servo motor connected to the motor driver is adjusted to the angle specified in the code, and then the object inside the receptacle is dropped into the Money Box.



## ● Connection Diagram:

You can assemble this project by breaking apart the PicoBricks modules at the proper points.





## ● Project Images:



# Setup Steps of The Project:

## Servo Motor Calibration

Before starting the assembly, you have to manually calibrate the angles of the servo motors. Otherwise, Servo Motors won't be working properly.

**(1)** Attach the servo horn to the servo motor (1)

**(2)** Then slowly turn the servo clockwise until it stops. It is not a problem if the servo horn is not the same as the angle shown in the image above. The important thing here is that you have hit the last angle of the servo. (2)

**(3)** Remove the servo horn from the servo motor (3)

**(4)** Reattach (4) and reposition the servo horn perpendicular to the servo motor as shown. (5)

**(5)**

**(6)** Slowly turn the servo horn counterclockwise (6) until it is parallel with the servo motor, as seen in the image. (7)

**(7)**

When this step is finished, it means that the servo motor is in the center position. It is important that you apply this process to other servo motors in the set. After processing the other motors, remove the servo horn and set aside for assembly.

**1** M3 Nut 4 Pcs.  
Micro:Bit Mainboard Brick  
M3 x 10 mm Screw 4 Pcs.

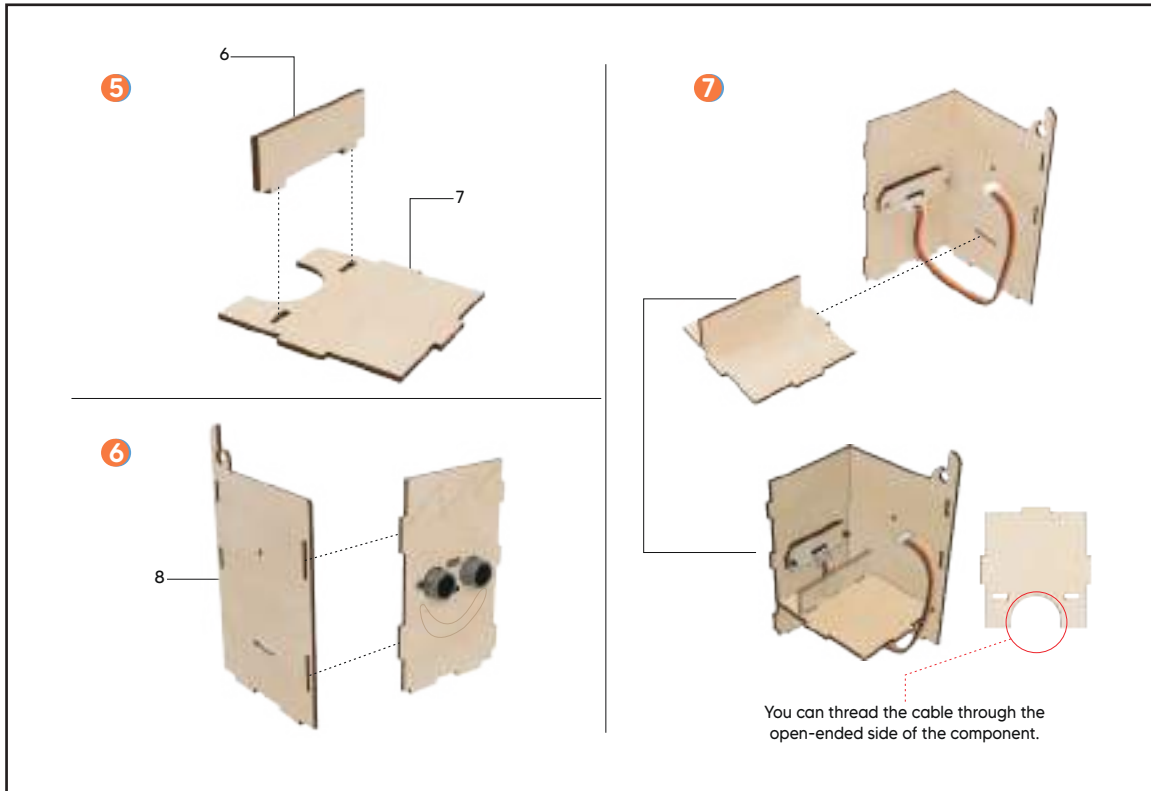
**2** M3 Nut 2 Pcs.  
Motor Driver Brick  
M3 x 10 mm Screw 2 Pcs.

**3** M3 Nut 2 Pcs.  
HC-SR04 Distance  
M3 x 12 mm Screw 2 Pcs.

**4** M3 Nut 2 Pcs.  
Servo Motor  
M3 x 12 mm Screw 2 Pcs.

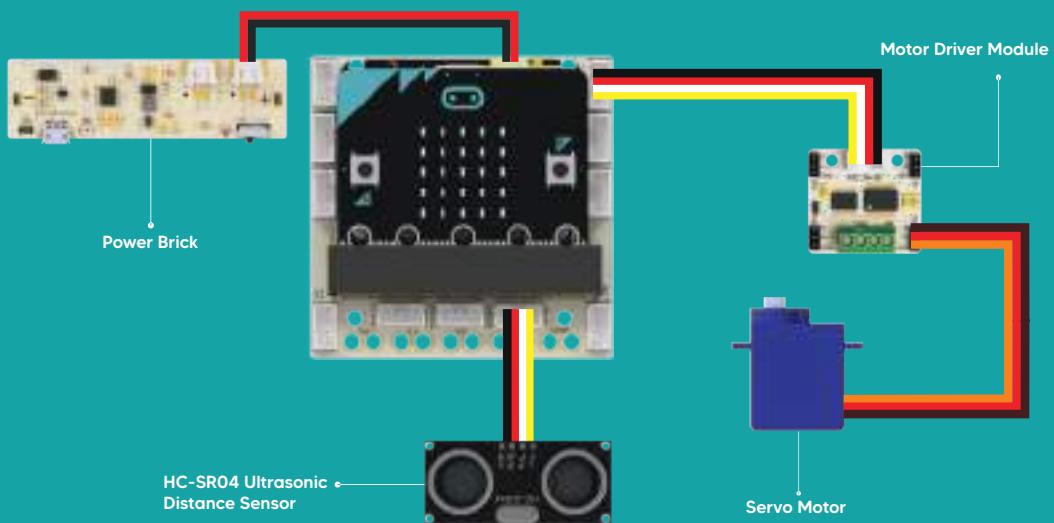
Make sure Servo Motor is facing the same direction as seen in the image

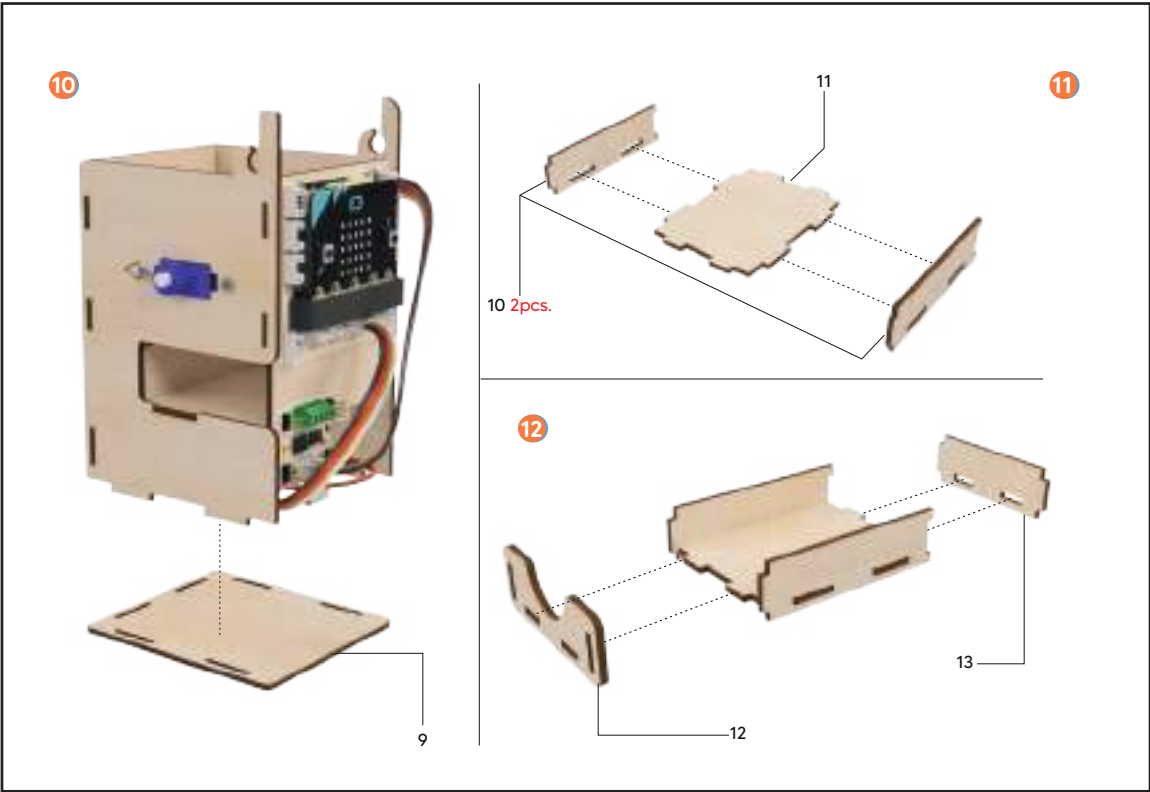
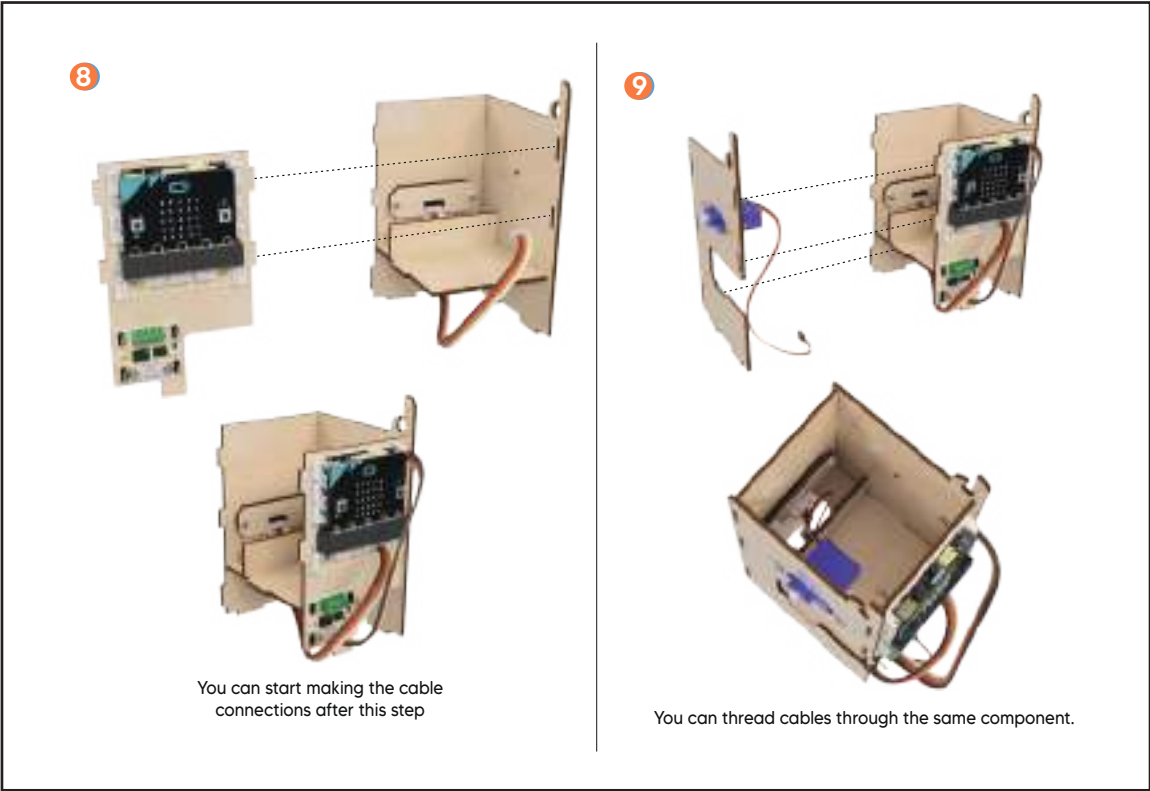
\*Don't forget to calibrate the servo motor you can move back to page 4 for it.

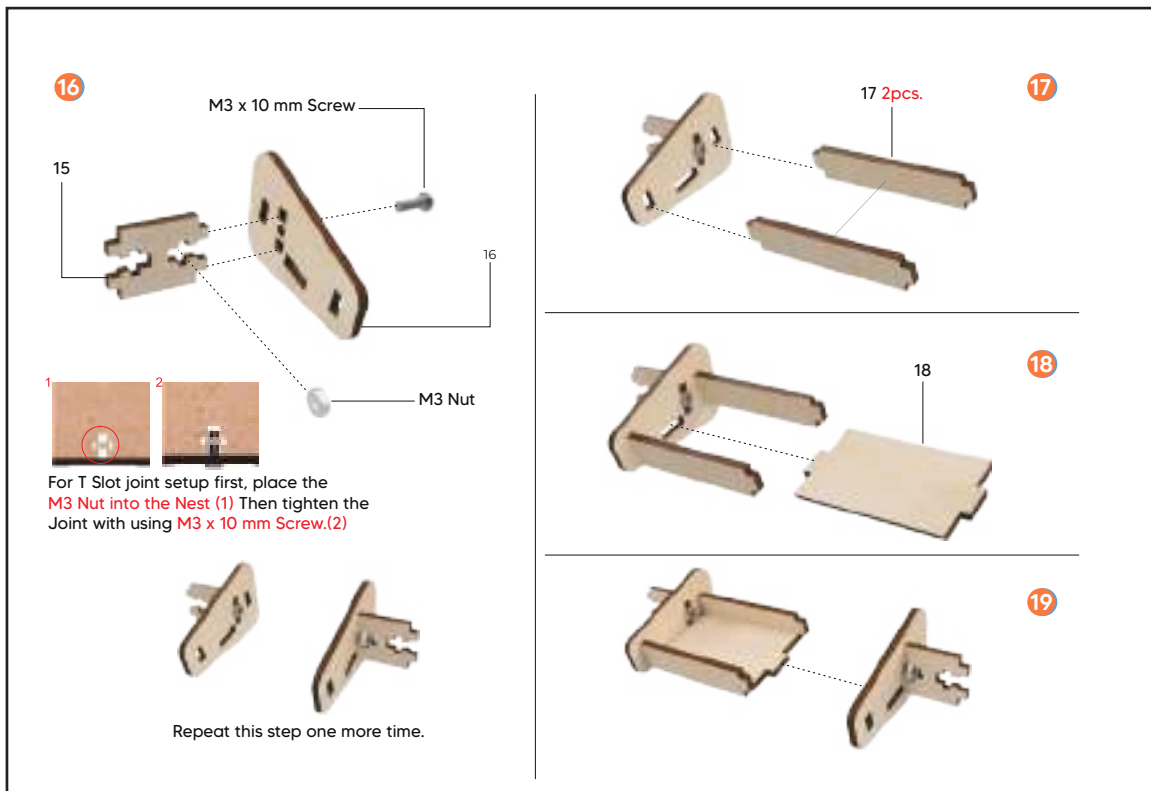
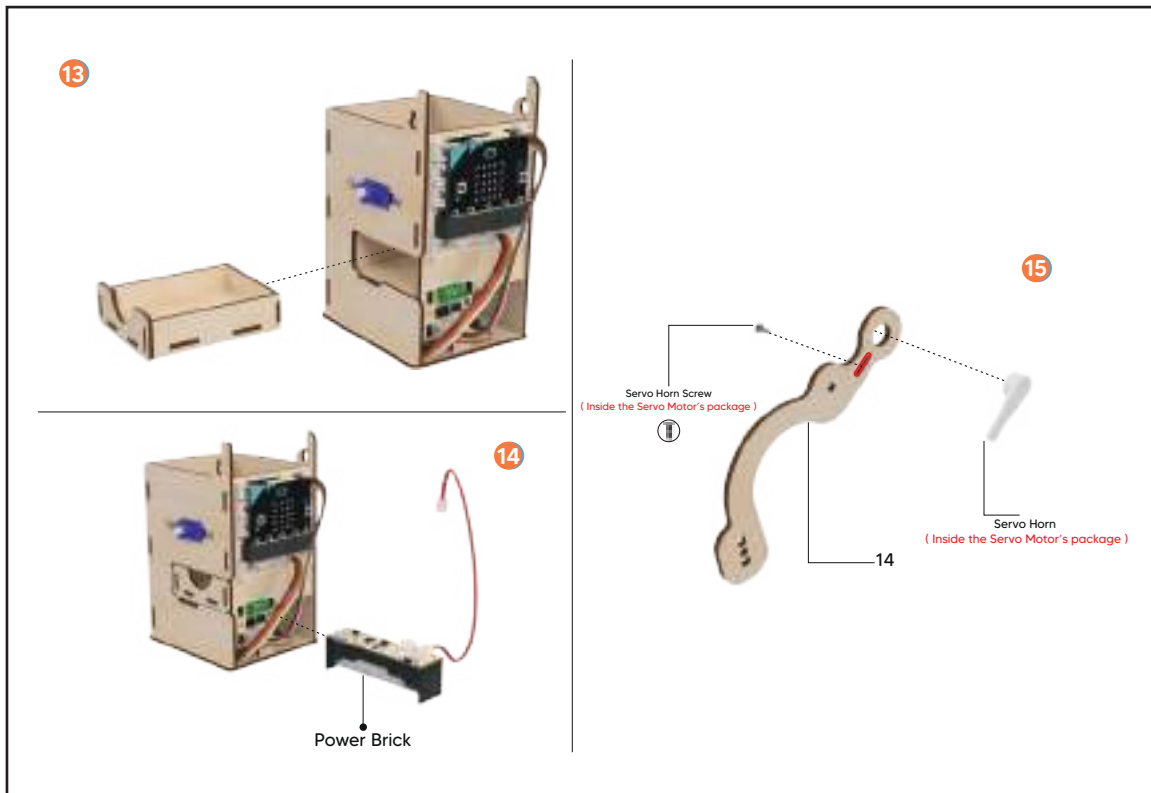


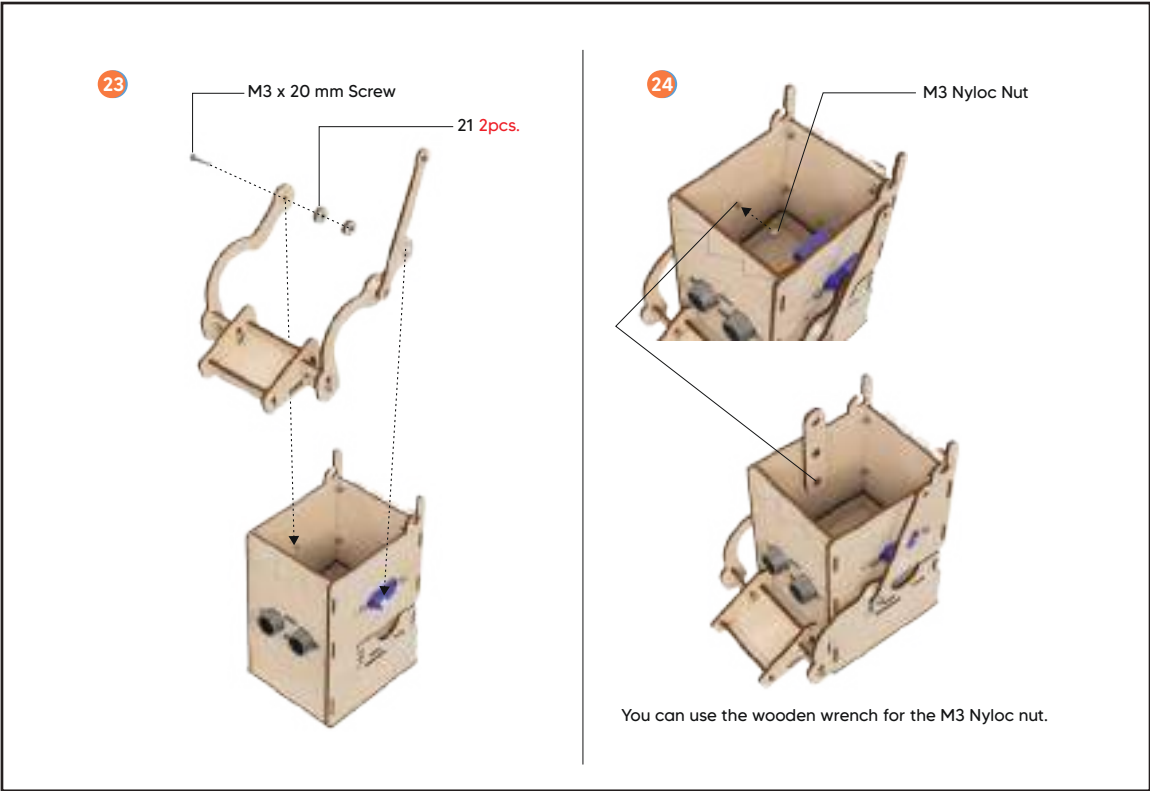
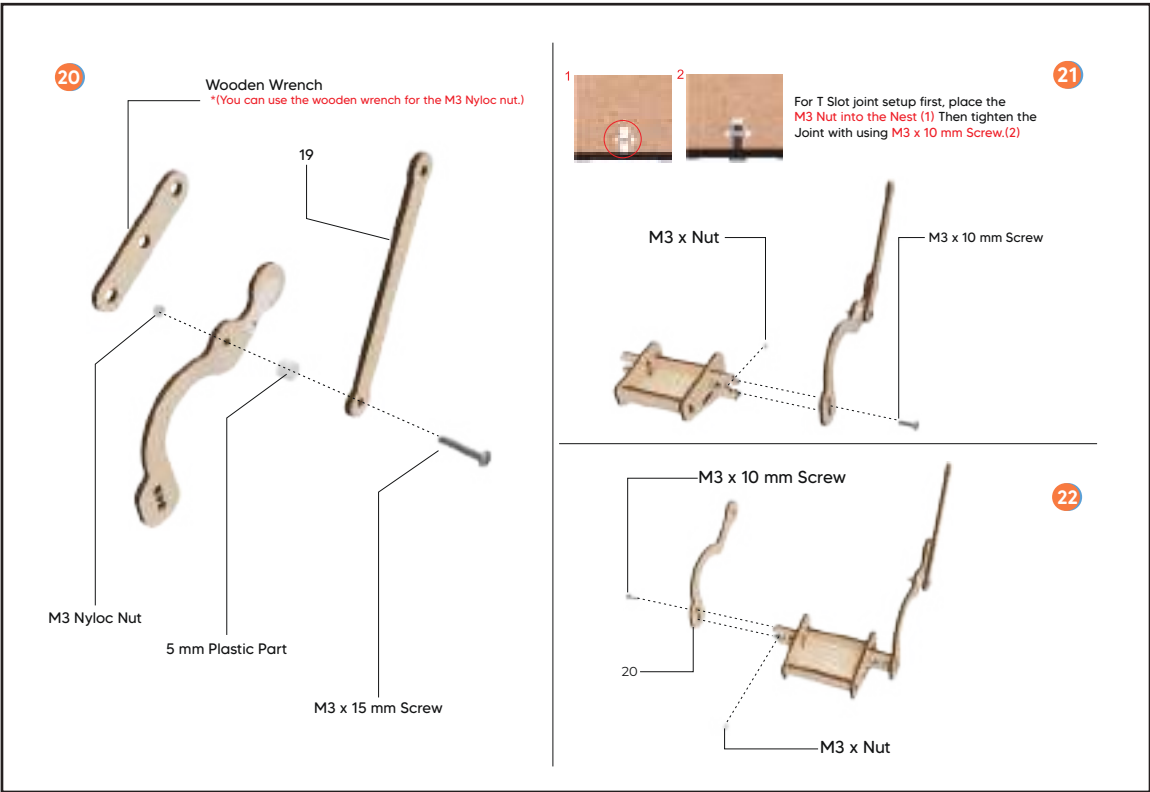
## Setting Up The Circuit

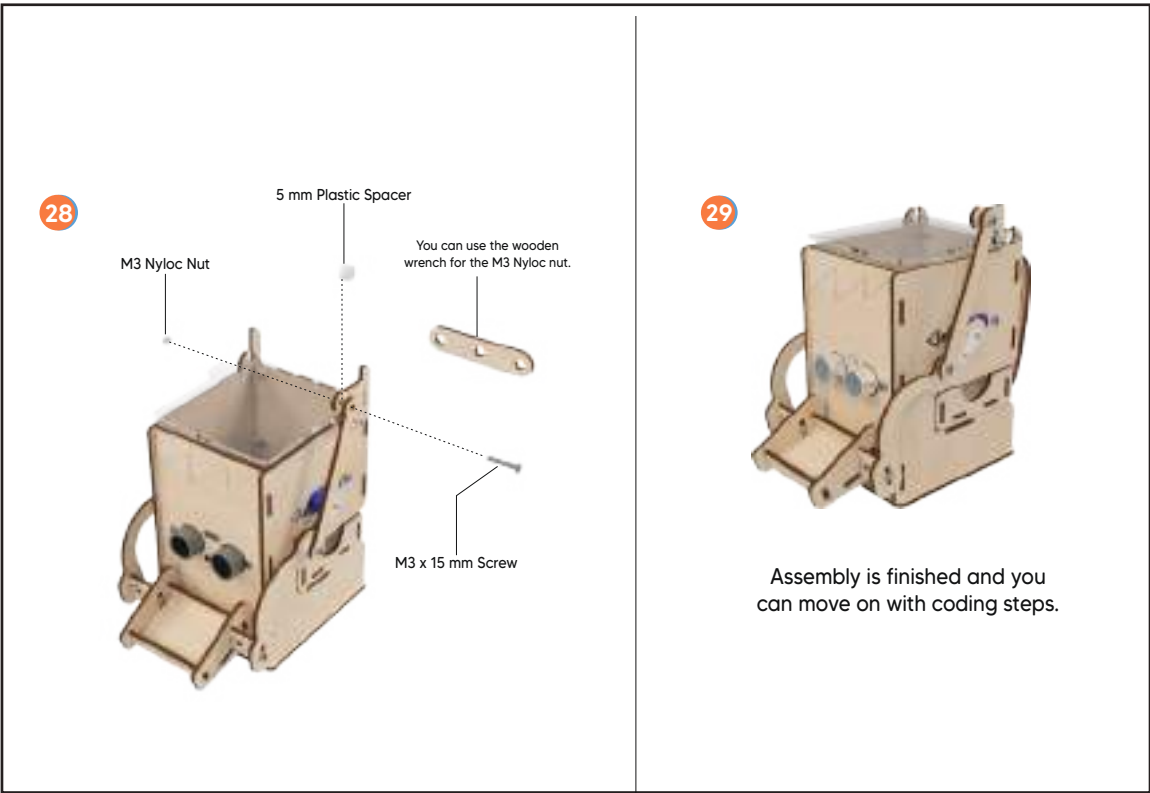
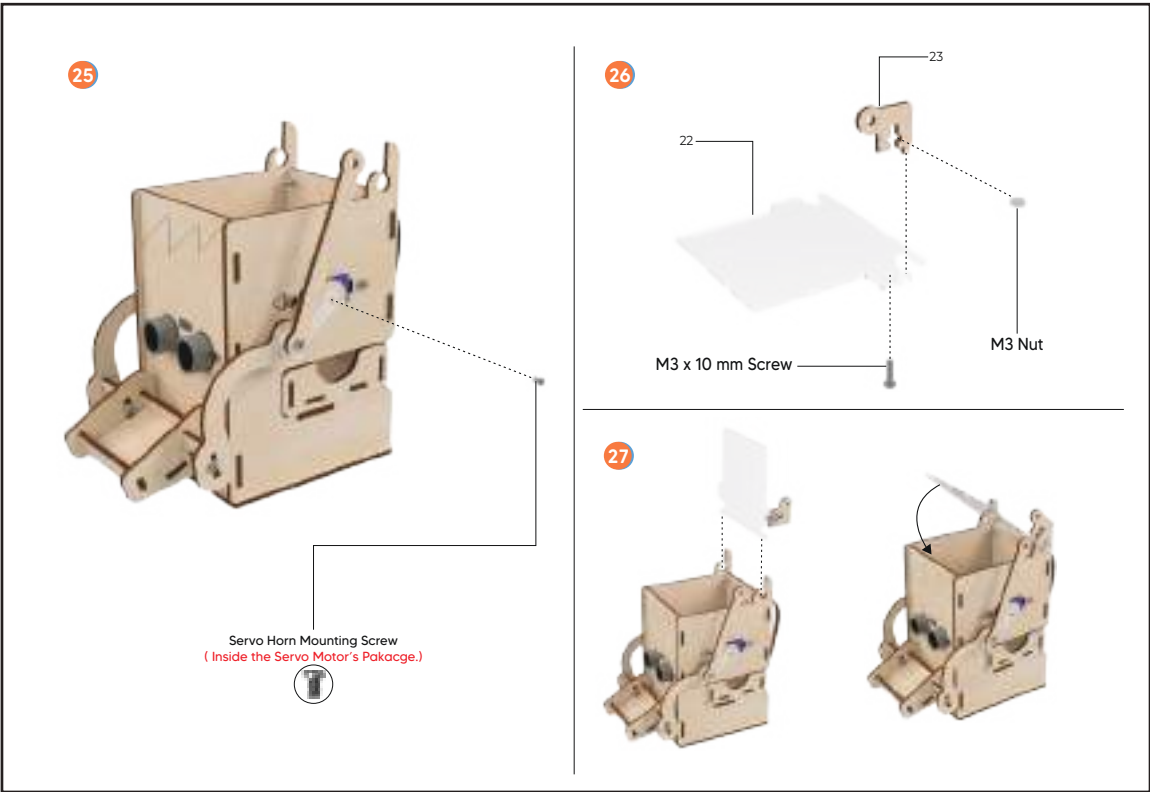
Let's get to know the circuit elements of Money Box that we completed the setup and make the circuit setup with PicoBricks modules.







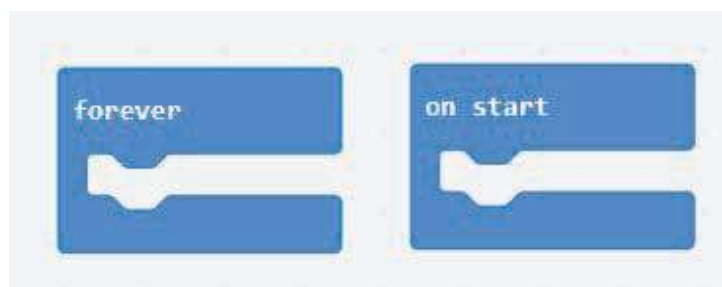




## ● MakeCode Code of The Project:

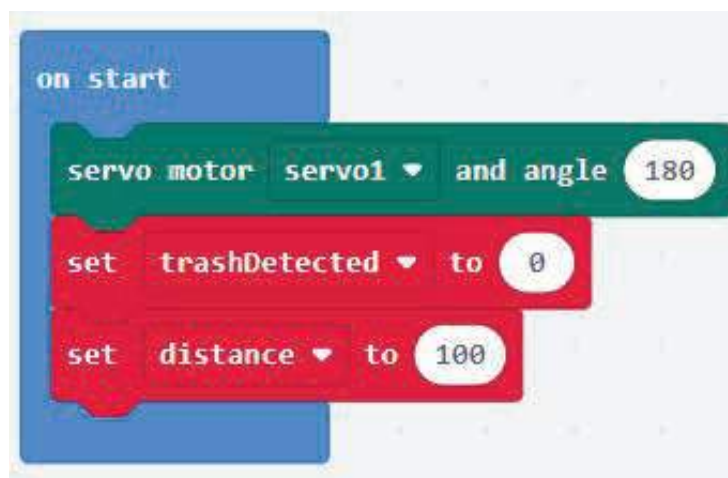
If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

1. When we create the project, we will see two blocks named “on start” and “forever” on the OLED screen. The blocks dragged into the 'on start' section are executed only once. On the other hand, 'Forever' repeats the blocks dragged into it continuously until the program is stopped.



2. Let's close the lid of the money box by moving the servo motor to a 145-degree angle when the project starts, which is connected to the motor driver.

3. Let's create the variables named 'trashDetected' and 'distance', which are necessary for the project, and assign their initial values.





4. Let's drag the expressions that we want to repeat continuously in the project into the 'forever' block:

- a) Let's create a variable named 'rawdistance' and assign the value of the distance sensor to this variable.
- b) If the value of the 'rawdistance' variable is less than 1200, let's assign this value to the 'distance' variable.
- c) Let's close the lid of the money box by moving the servo motor to 145 degrees and wait for 500 milliseconds.
- d) If the value of the 'distance' variable is less than 7, let's set the 'trashDetected' variable to 1 and wait for 300 milliseconds.
- e) If the value of the 'distance' variable is greater than 12 and the value of the 'trashDetected' variable is 1, set the 'trashDetected' value to 0, and move the servo motor to 90 degrees to allow the object in the hopper to fall into the money box.

```
forever loop:  
  set rawdistance to ultrasonic distance sensor value with trig pin at P2 and echo pin at P1  
  if rawdistance < 1200 then  
    set distance to rawdistance  
    servo motor: servo1 set angle 145  
    pause (ms) 500  
  if distance < 7 then  
    set trashDetected to 1  
    pause (ms) 300  
  if distance > 12 and trashDetected = 1 then  
    set trashDetected to 0  
    servo motor: servo1 set angle 90  
    pause (ms) 500
```

## 5. The Code of The Project is Ready!

The image shows a Scratch script for a robot project. It consists of two main parts: an 'on start' block and a 'forever' loop.

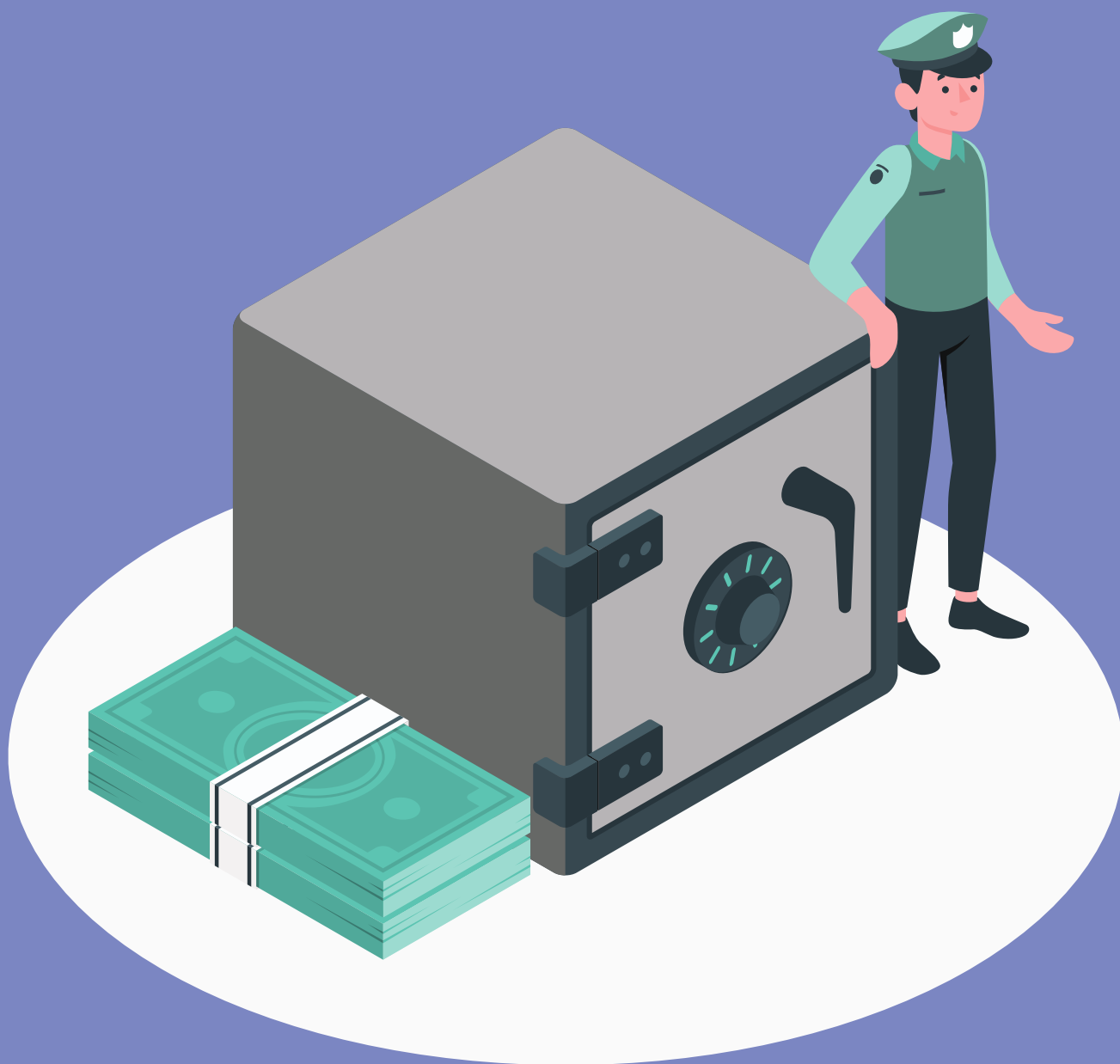
**on start:**

- servo motor: servo1 → and angle: 180
- set trashDetected = to 0
- set distance = to 180

**forever loop:**

- set rawDistance = to ultrasonic distance sensor value with trig pin at P2 → and echo pin at P1 →
- if rawDistance < 1200 then
  - set distance = to rawDistance
- power (w): 500
- servo motor: servo1 → and angle: 180
- if distance < 5 then
  - set trashDetected = to 1
- power (w): 200
- if distance < 0 and trashDetected < 1 then
  - set trashDetected = to 0
  - servo motor: servo1 → and angle: 90
  - power (w): 500

# Safe Box



# Safe Box Project

The PicoBricks Safe Box is an educational project kit designed to create a safe box that automatically locks after assembling the wooden pieces and PicoBricks modules according to the installation steps.

In this project, when the correct password is entered by using a potentiometer and button, the door of the safe opens. After closing the door, it automatically locks thanks to the LDR sensor inside the safe.

## Project Details:

Project Link: <https://makecode.microbit.org/S03779-88288-97416-86624>

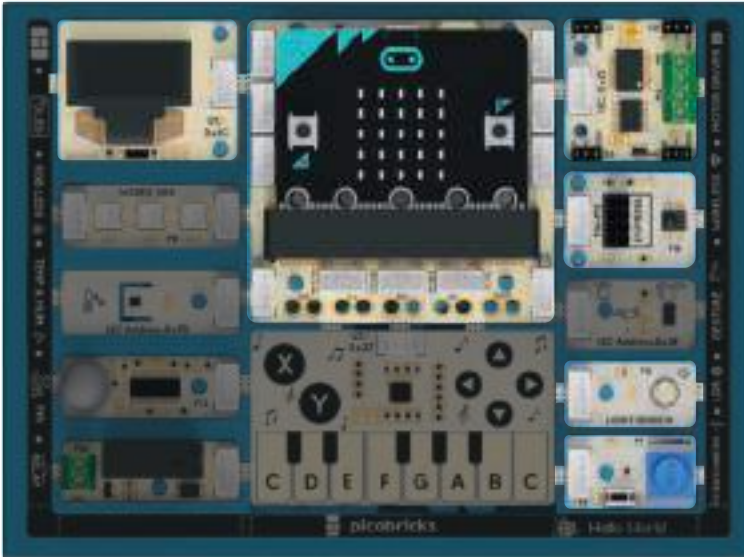


In this project, when we correctly enter the password we have set in the code by using the potentiometer and button module, the servo motor moves to the specified position, and the door opens. Through the LDR module, the closure of the safe box lid is detected, triggering the servo motor to operate and lock the door. We will create the code blocks that enable these functions. With the PicoBricks OLED display module and Micro:Bit Matrix LEDs in the Safe Box project, we will obtain visual output.

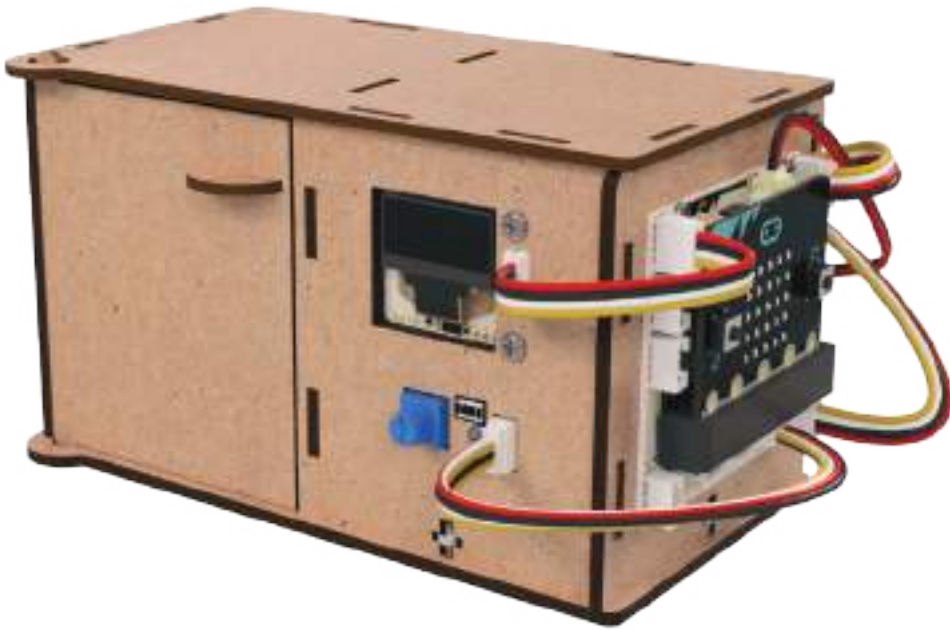


## ● Connection Diagram:

You can assemble this project by breaking the PicoBricks modules at the proper points.



## ● Project Images:



# Setup Steps of The Project:

## Servo Motor Calibration

Before starting the assembly, you have to manually calibrate the angles of the servo motors. Otherwise, Servo Motors won't be working properly.

**(1)**  
Attach the servo horn to the servo motor (1)

**(2)**  
Then slowly turn the servo horn clockwise until it stops. It is not a problem if the servo horn is not the same as the angle shown in the image above. The important thing here is that you have hit the last angle of the servo. (2)

**(3)**  
Remove the servo horn from the servo motor (3)

**(4)**  
**(5)**  
Reattach (4) and reposition the servo horn perpendicular to the servo motor as shown. (5)

**(6)**  
Slowly turn the servo horn counterclockwise (6) until it is parallel with the servo motor, as seen in the image. (7)

**(7)**  
When this step is finished, it means that the servo motor is in the center position. It is important that you apply this process to other servo motors in the set. After processing the other motors, remove the servo horn and set aside for assembly.

1  
2

PicoBricks LDR Brick

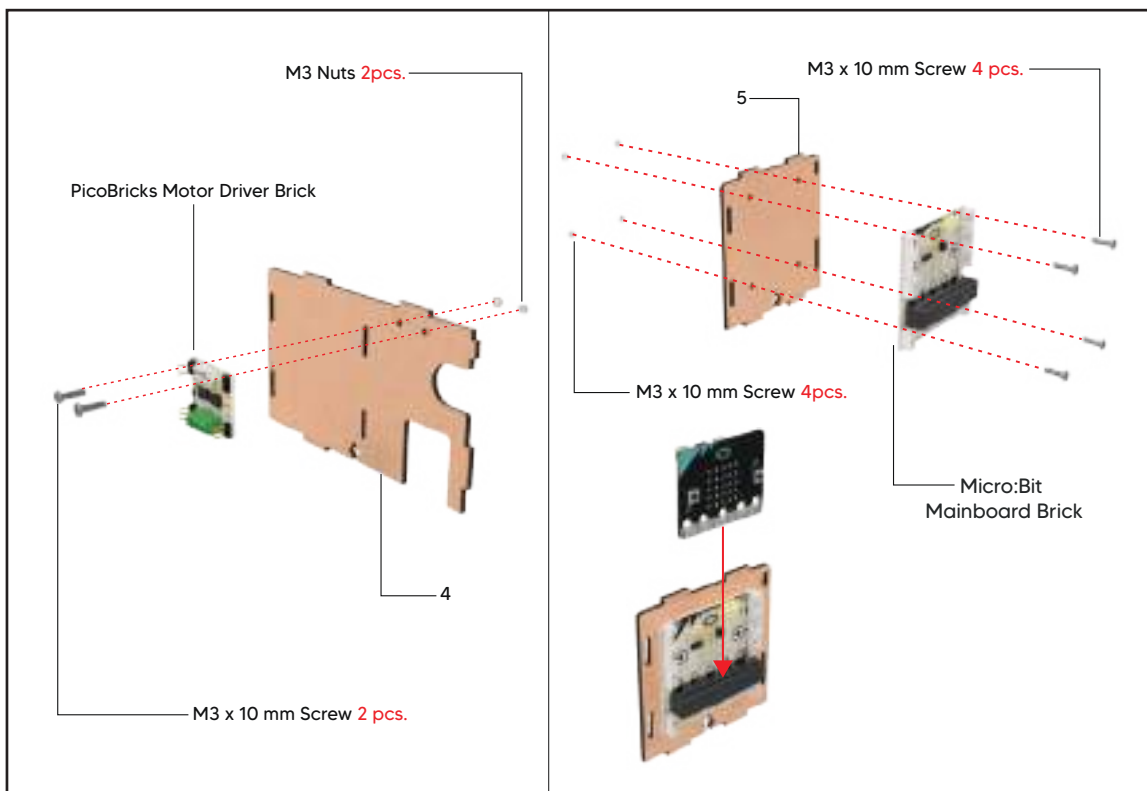
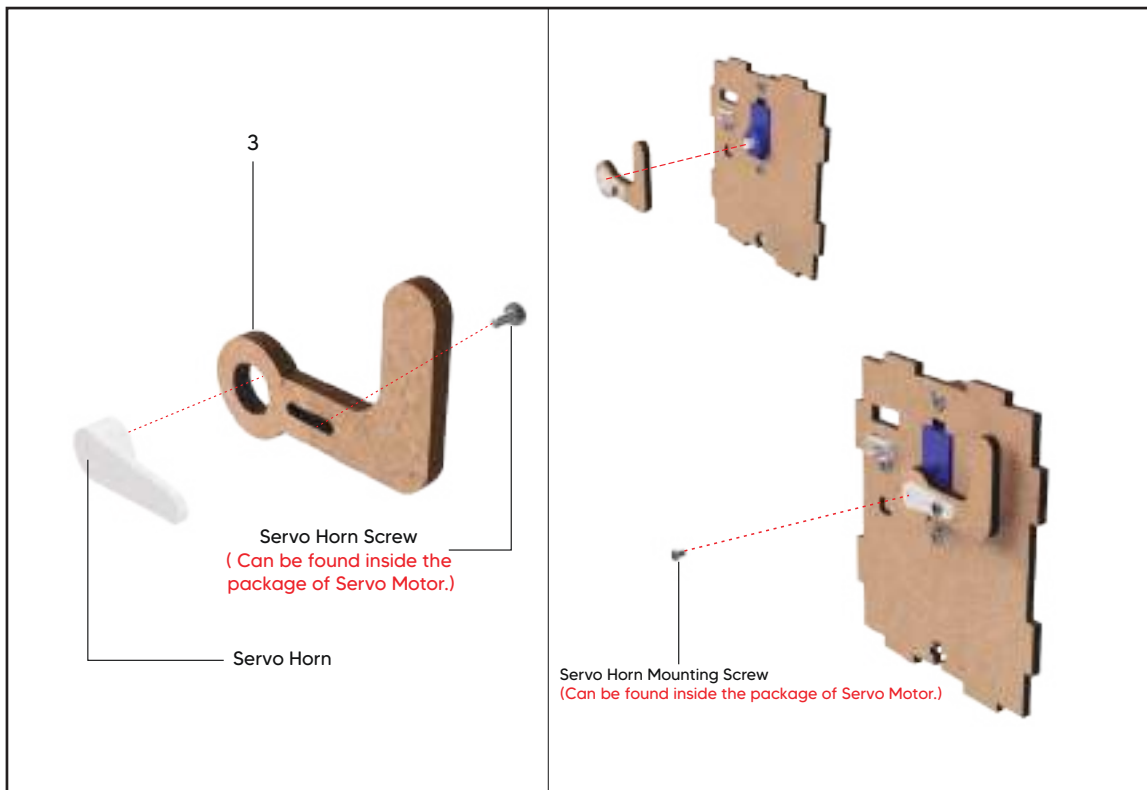
Servo Motor

M3 x 12 mm Screw 2 pcs.

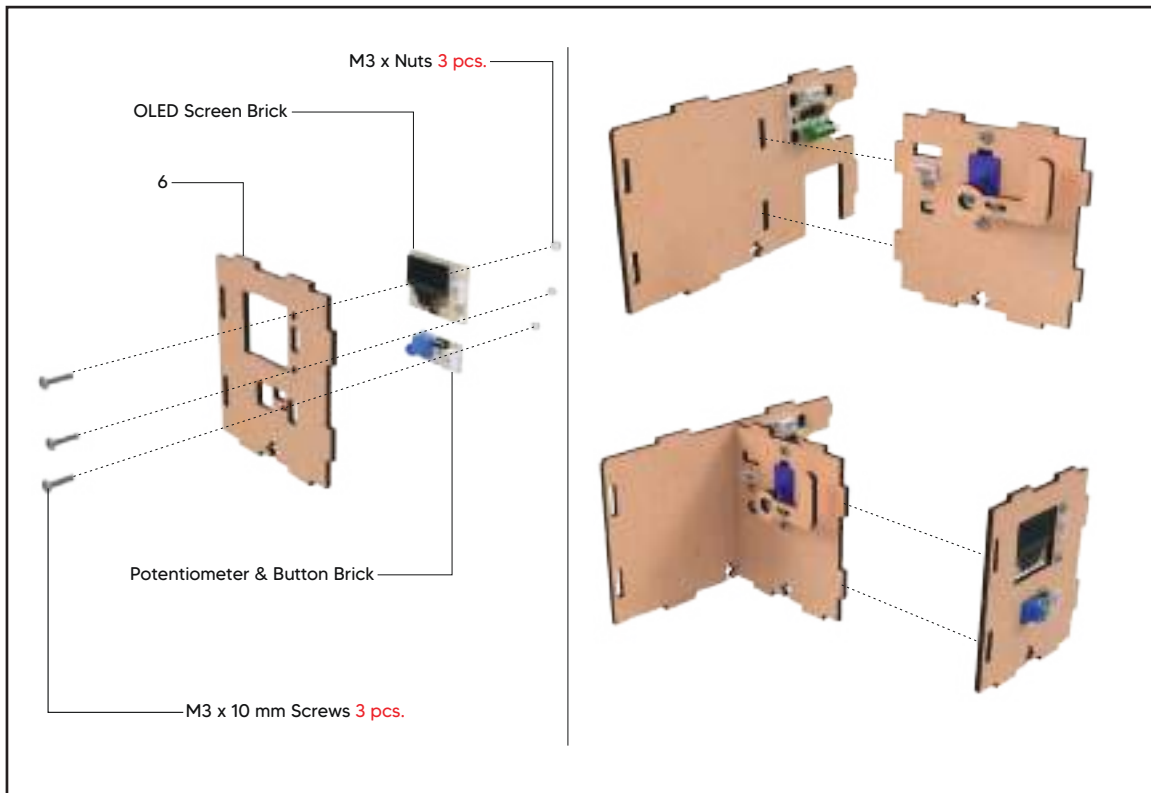
M3 Nut 2 pcs.

M3 Nut

M3 x 10 mm Screw

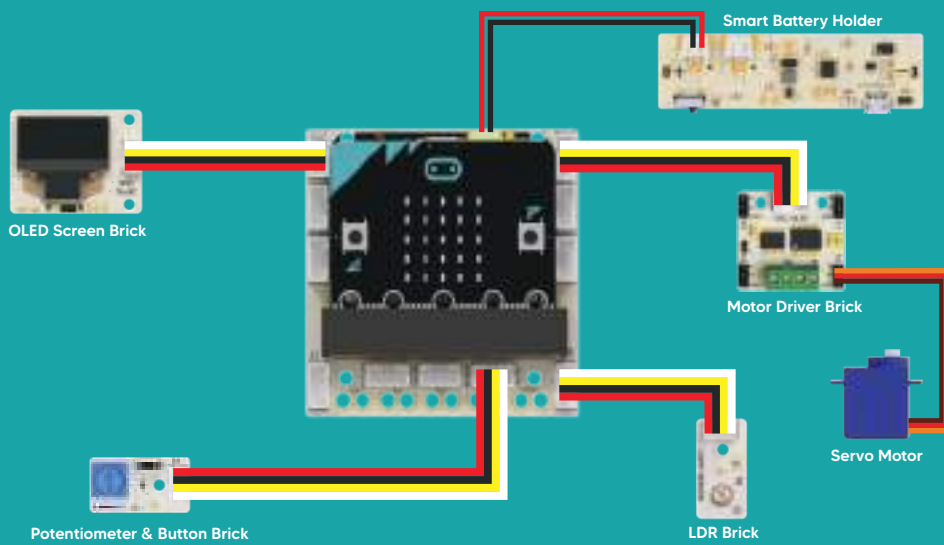


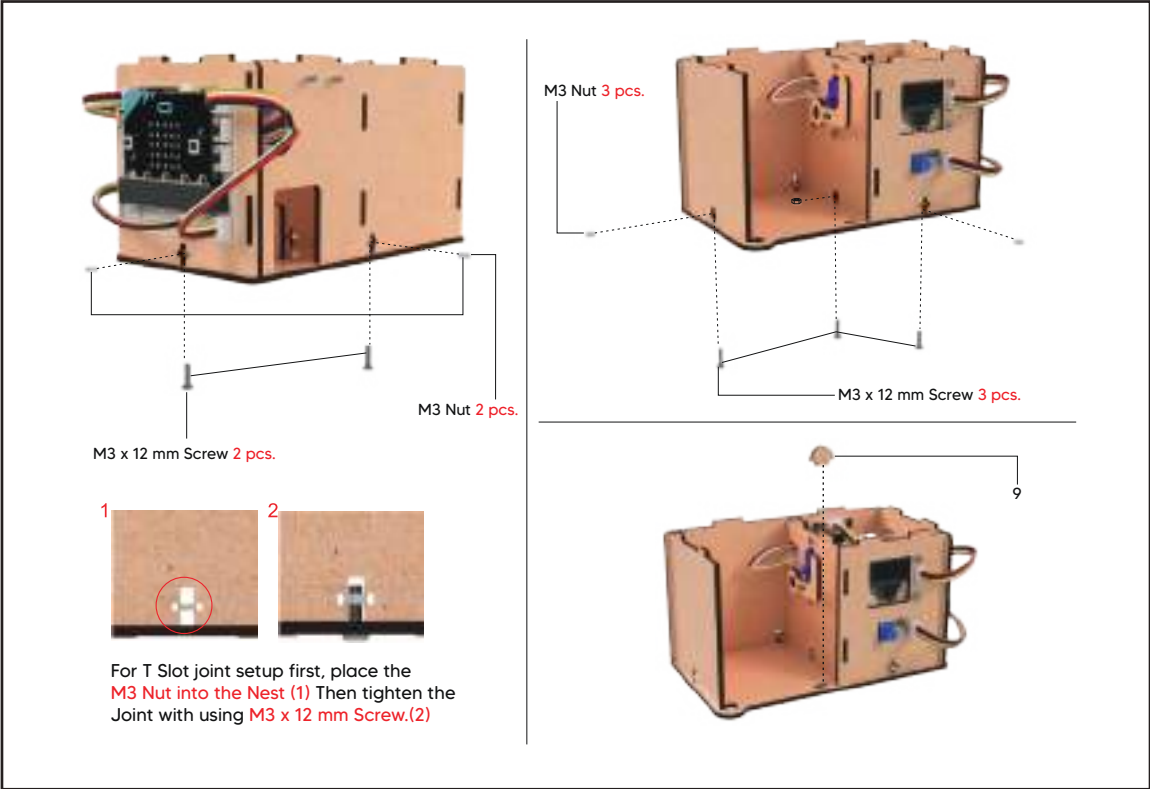
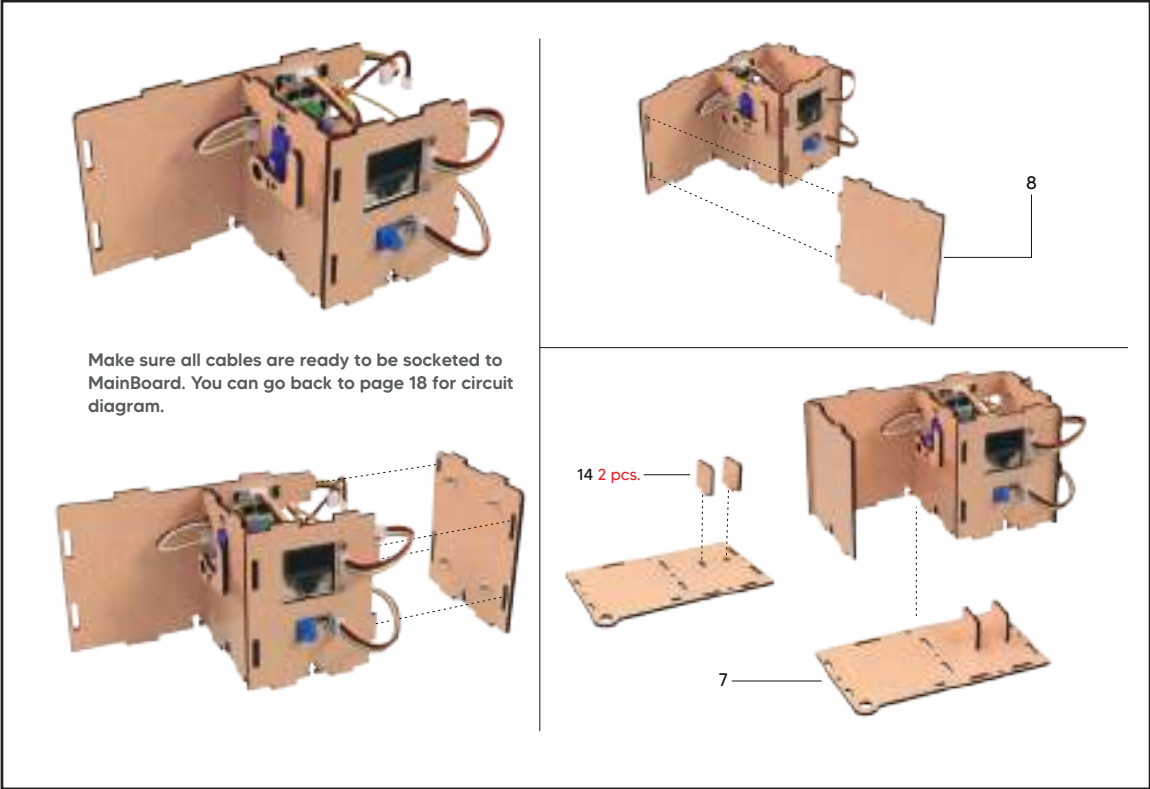


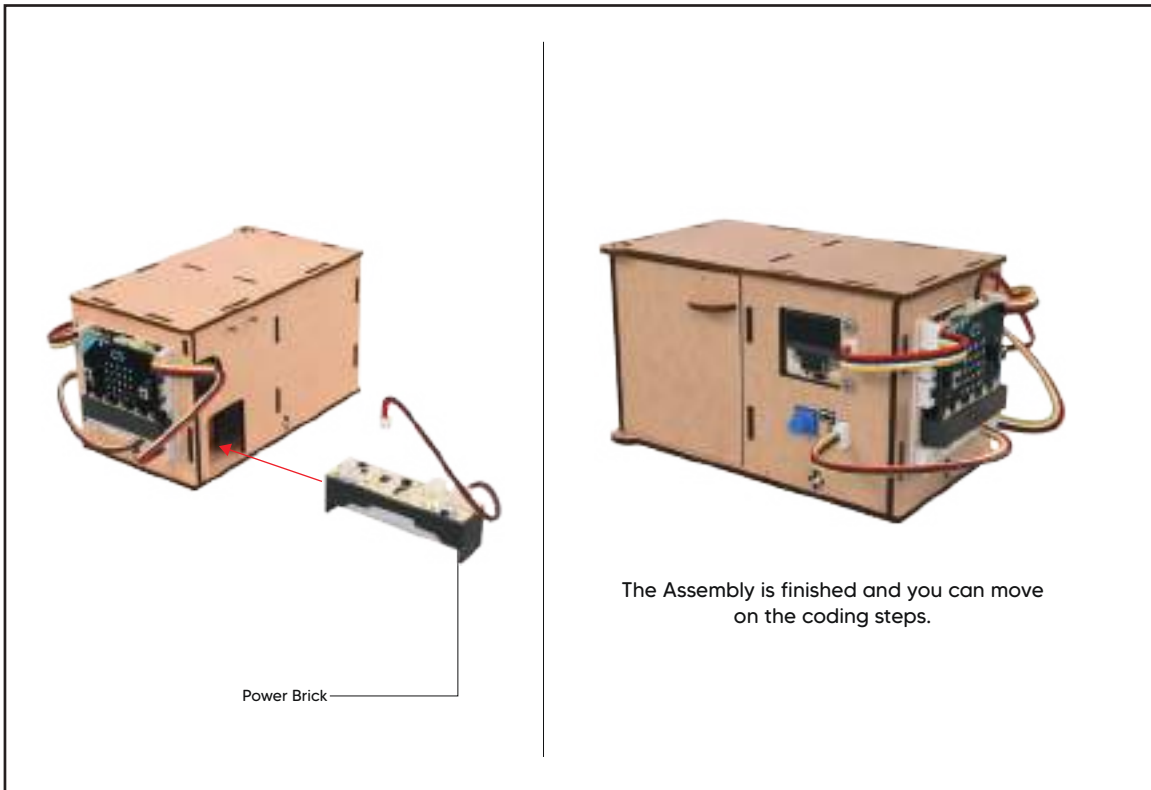
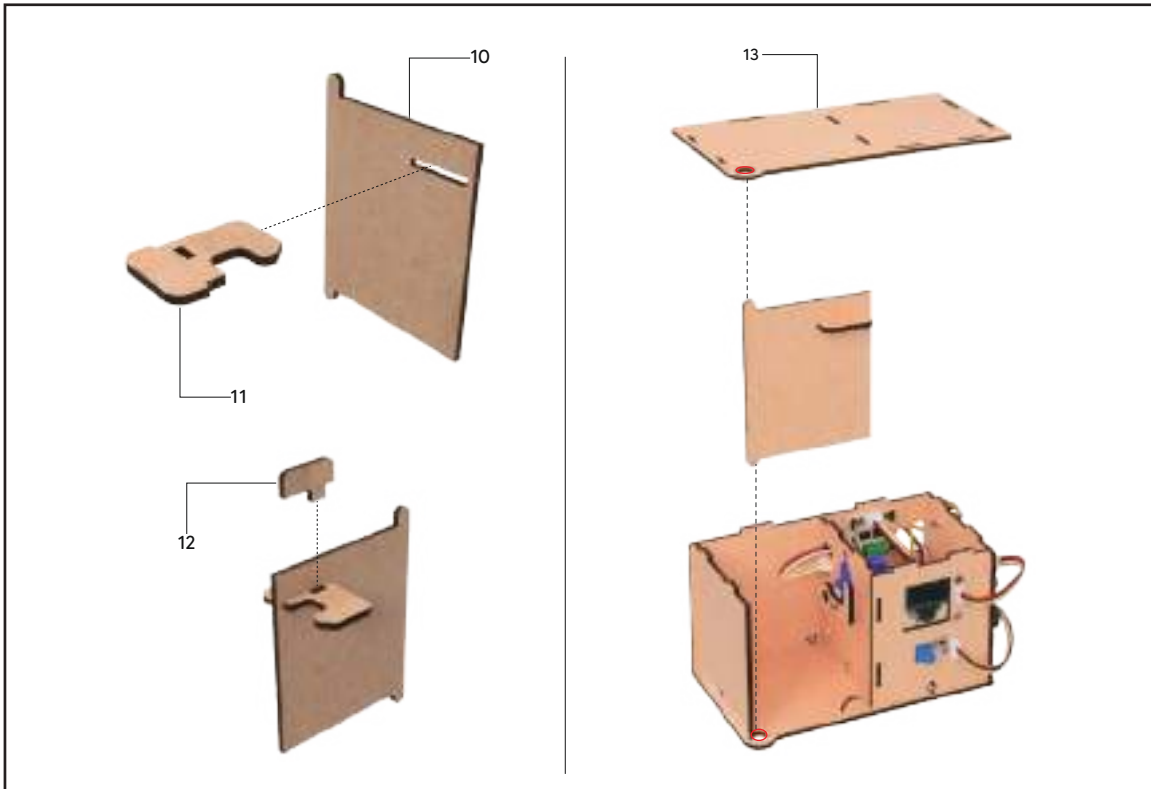


## Setting Up The Circuit

Let's get to know the circuit elements of Safe Box that we completed the setup and make the circuit setup with PicoBricks modules.



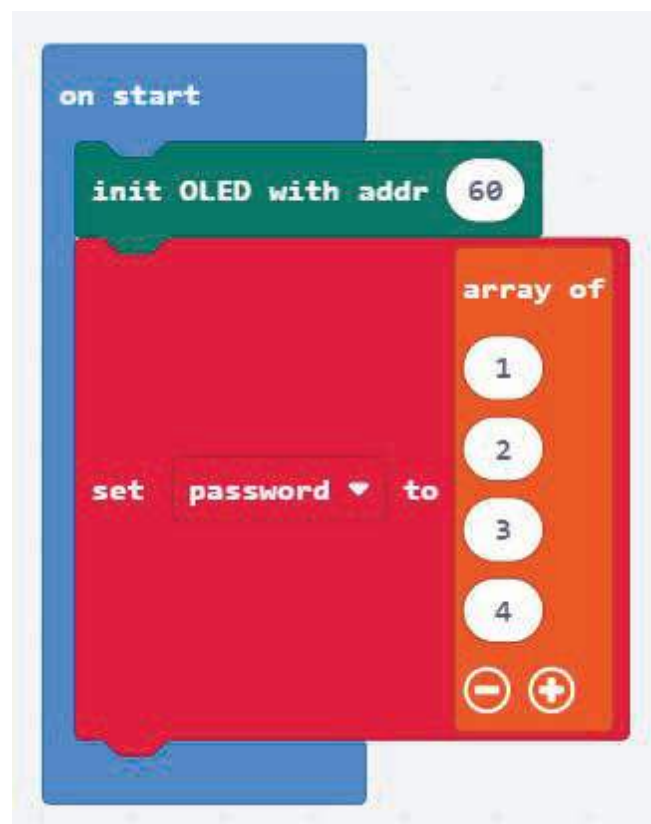




## ● MakeCode Code of The Project:

If you have completed the PicoBricks - MakeCode connection and add-on installation steps, the coding steps to follow for the first project are detailed in the visual below.

1. Since we will be using an OLED screen in our project, let's drag the code block that initializes the OLED screen into the 'on start' block and define the password for the Safe Box in a list called 'password'.



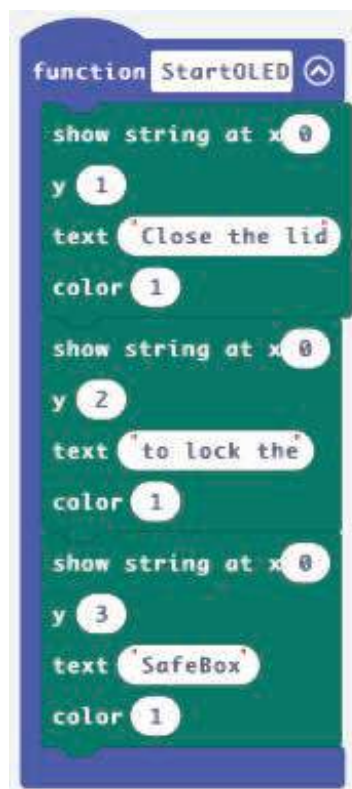
2. Let's define the functions we will use in our project.

a) In the function named 'controlLoop,' the user-entered password is compared with the correct password.



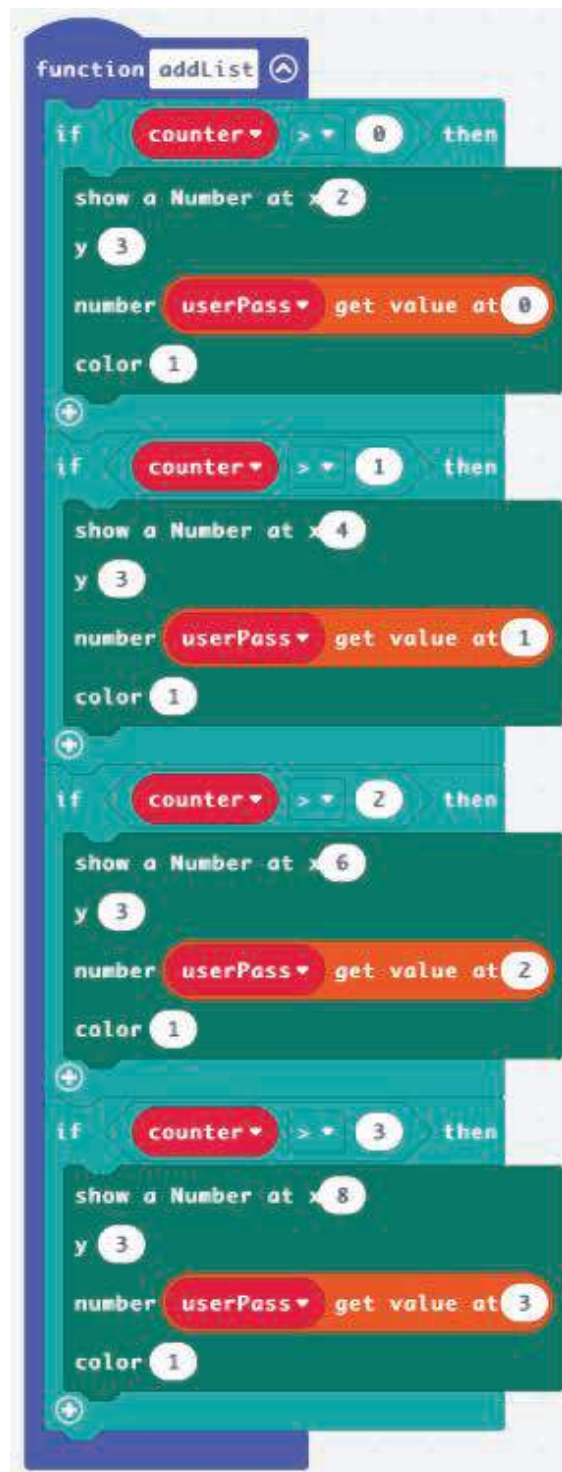
```
function controlLoop
  for index from 0 to 4
  do
    if password get value at index == userPass get value at index then
      set control to 1
    else
      show icon [X]
  end
  if control == 1 then
    show icon [checkmark]
    servo motor Servo1 and angle 90
    pause (sec) 1000
    clear
  end
end
```

b) Let's define the text that should be displayed on the OLED screen when the Safe Box door is open in a function called 'startOLED'.



```
function StartOLED
  show string at x 0
  y 1
  text 'Close the lid'
  color 1
  show string at x 0
  y 2
  text 'to lock the'
  color 1
  show string at x 0
  y 3
  text 'SafeBox'
  color 1
end
```

c) The function that sends the values determined by the user with a potentiometer to a list named “userPass” after pressing the button.



```
function addList
  if counter > 0 then
    show a Number at x 2
    y 3
    number userPass get value at 0
    color 1
  if counter > 1 then
    show a Number at x 4
    y 3
    number userPass get value at 1
    color 1
  if counter > 2 then
    show a Number at x 6
    y 3
    number userPass get value at 2
    color 1
  if counter > 3 then
    show a Number at x 8
    y 3
    number userPass get value at 3
    color 1
```

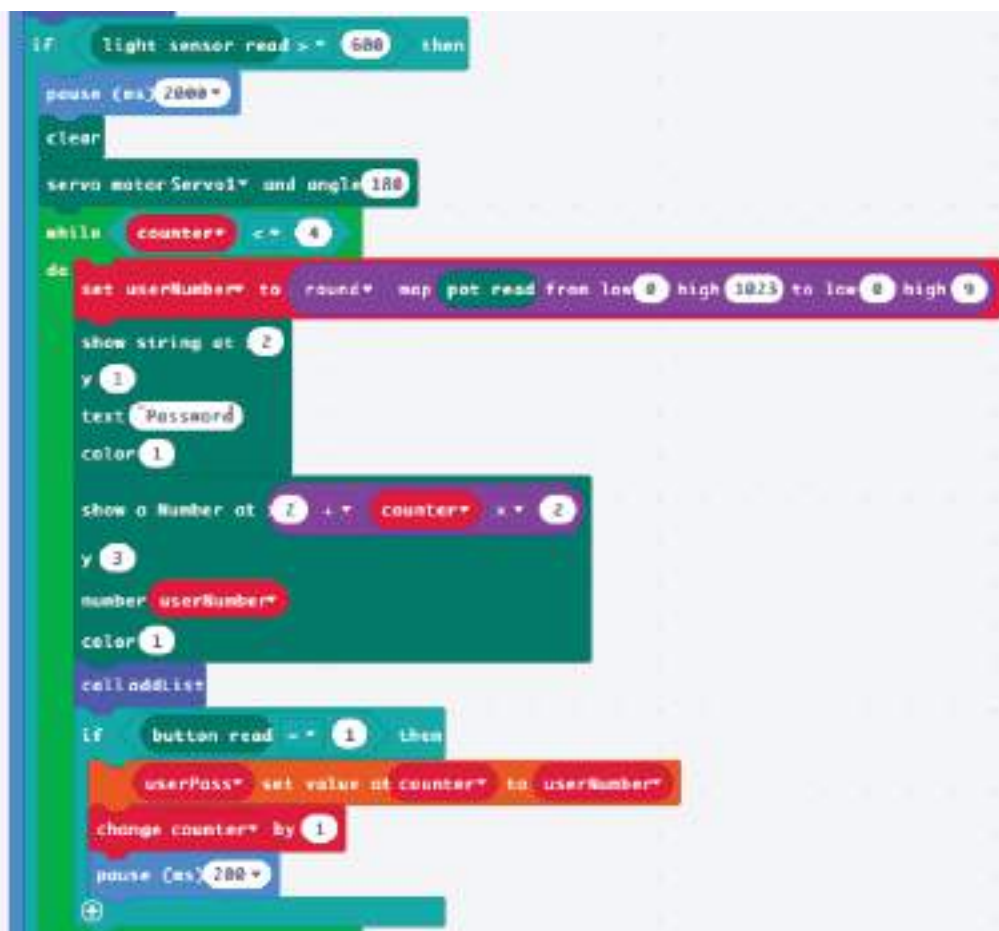
The image shows a Scratch code block for a function named 'addList'. It contains four conditional blocks, each starting with 'if counter > [number] then'. Each block contains the following actions: 'show a Number at x [number] y 3', 'number userPass get value at [number]', and 'color 1'. The numbers in the 'if' conditions and the 'show a Number' blocks are 0, 1, 2, and 3 respectively. The 'number userPass get value at' blocks use the same numbers as the 'if' conditions. The 'color' block in each conditional block is set to 1.

3. Let's create the main loop of our code inside the "forever" block by calling the functions we have created.

a) If the door is closed, let's create a smiling face emoji on the Micro:Bit LED matrix using the "show icon" block and call the "startOLED" function.



b) If the Safe Box door is closed, (if the LDR sensor value is less than 600) repeat until the "counter" variable reaches 4 (until the user creates a 4-character password). Whenever the user presses a button, define the password in the "userPass" list using the "addList" function.



4. Call the function named "controlList" to check the password entered by the user, and if the password is correct, open the door.



5. Wait for half a second. Reset the "control" and "counter" variables.



6. The Code of The Project is Ready!



# GRAHAM BELL



"What this power is I can not say; all I know is that it exists and it becomes available only when a man is in that state of mind in which he knows exactly what he wants and is fully determined not to quit until he finds it."