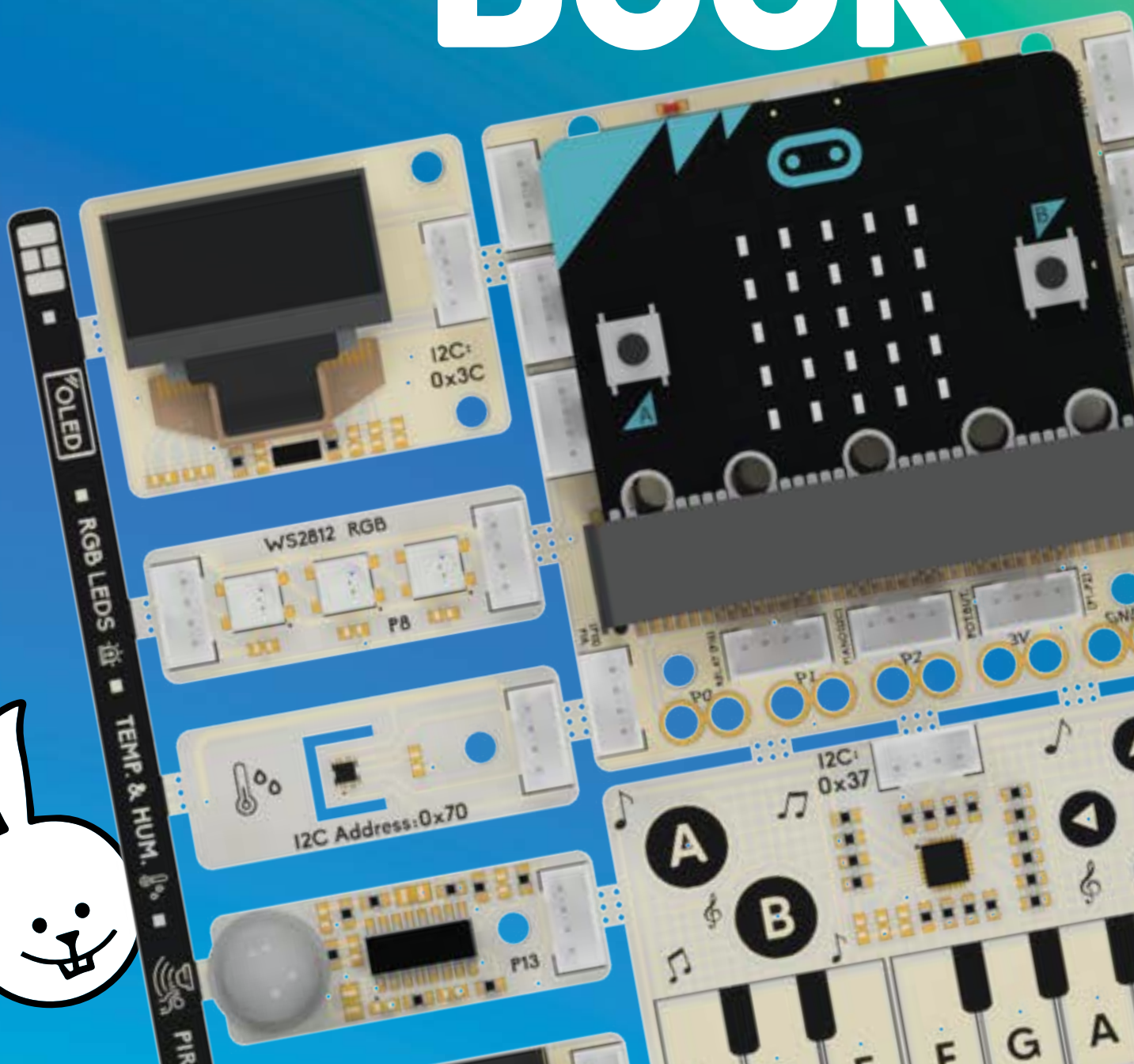




Project Book



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 Özer

Designer: Elanur Tokalak

PicoBricks Developer Team

Project Manager - Yasir Çiçek

Chief Developer - Suat Morkan

Product Developer - Naze Gizem Özer

Industrial Design Developer - Sercan Okay

Graphic Designer Developer - Elanur Tokalak

Hardware & Software Developer - Atakan Öztürk

Learning Content & Software Developer - Selim Gayretli



Powered by

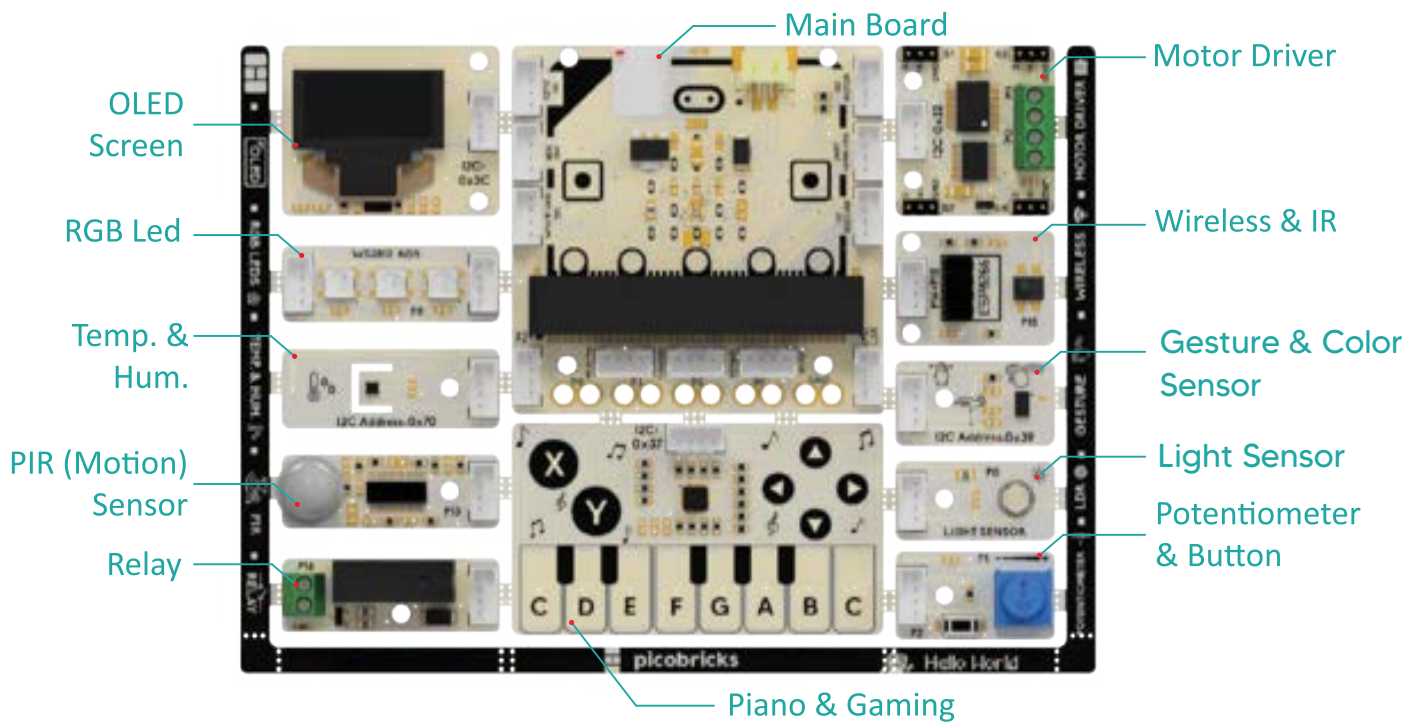


What Is PicoBricks?	5-6
What Is Micro:Bit?	7
What Is MakeCode?	8-10
PROJECTS	
Blink	11-14
Action - Reaction	15-18
Color Cards	19-22
PicoBricks Piano	23-26
RGB Led Control Panel	27-30
Thermometer	31-34
Smart Cooler	35-38
Night and Day	39-42
Fizz - Buzz	43-46
Depht Meter	47-50
Morse Code Cryptogaphty	51-54
Car Parking System	55-58
Table Lamp	59-62
IoT Control Panel	63-69
IoT Vase	70-76
Coin Dispencer.....	77-80
Gesture Controlled ARM Pan Tilt	81-85
3D Labyrinth	86-89
Radar	90-94
PicoBricks Logo Lamp	95-98

ADD ON

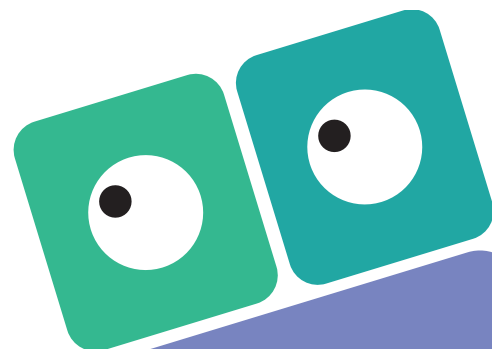
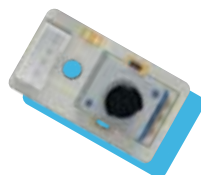
London Eye	99-106
Mars Explorer	107-116
Trash Tech	117-125
Money Box	126-135
Safe Box	136-144

What Is PicoBricks?

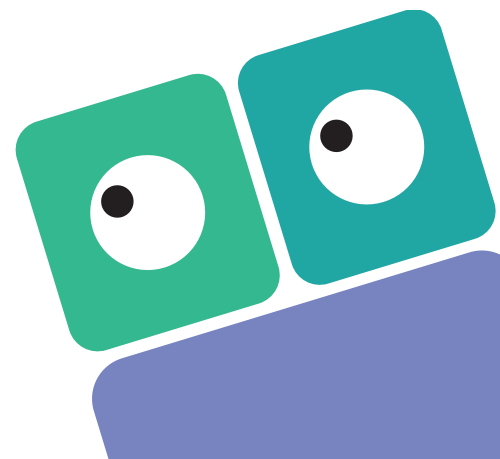
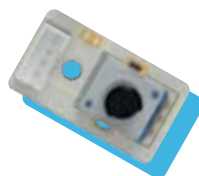
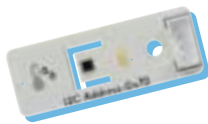


PicoBricks is a development board that eliminates the difficulties experienced in physical programming. You can invest the time saved from these challenges to create more creative projects.

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 MicroBlocks?

MicroBlocks is a free, Scratch-like blocks programming language for learning physical computing with educational microcontroller boards such as the micro:bit, Adafruit Circuit Playground Express, and many others. MicroBlocks is a live environment. Click on a block and it runs immediately, right on the board. Try out commands. See and graph sensor values in real time. No more waiting for code to compile and download. Want to display an animation while controlling a motor? No problem! MicroBlocks lets you write separate scripts for each task and run them at the same time. Your code is simpler to write and easier to understand. MicroBlocks runs on many different boards, but your scripts are portable. Buttons, sensors, and display blocks behave the same on all boards with the relevant hardware. Once you run the code in MicroBlocks, you can disconnect the USB and feed the PicoBricks with a different power source. The code on the card will work automatically.

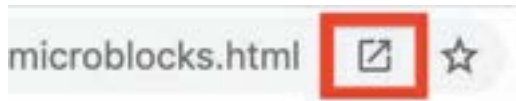
To program PicoBricks with MicroBlocks, let's open <https://microblocks.fun/> in the browser (Google Chrome and Edge browsers are recommended).



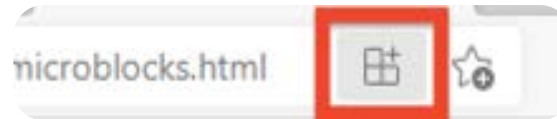
You don't need to install anything to run MicroBlocks in a Chrome or Edge browser; you can run the online editor by clicking the Run button in the menu at the top right of the screen. Alternatively, by clicking the Download button, you can download an off-line version suitable for your operating system and install it on your computer.

You can save MicroBlocks Web editor in your browser and use it without internet access. Run MicroBlocks in your browser to register the MicroBlocks Web app, then click the install button in the upper-right corner of your browser's URL bar.

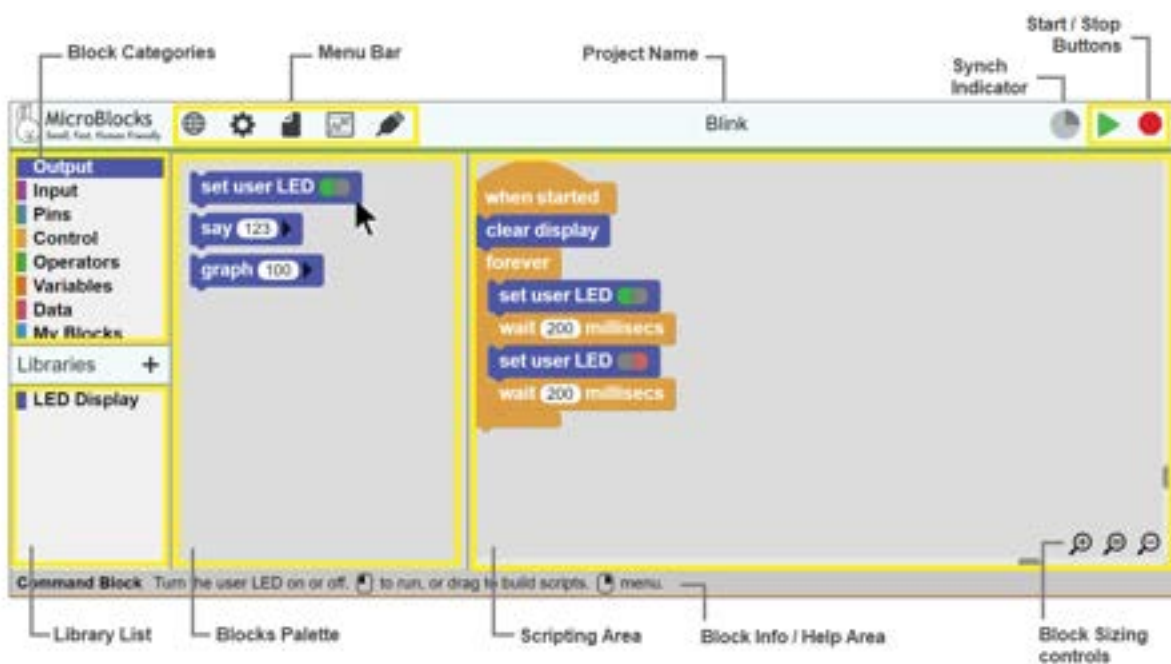
Google Chrome



Microsoft Edge



When you open the MicroBlocks program, you will see the IDE image shown below. You can review the explanation of the IDE components below. For a detailed and most current description of the IDE, please refer to our [User Guide](#) in our WIKI.



1. Menu Bar (🌐 ⚙️ 📁 📊 🗑️): In this section, the first button from left to right allows us to change the language option of the program. The second button is the menu where we can see the operational settings of MicroBlocks and the firmware update selection, while the third button offers the File related options. The fourth button opens a graph window used by the graph block to plot data, while the fifth rightmost button is used to connect to the PicoBricks via the USB interface..

2. Block Categories: This field contains the categories of blocks used for programming in MicroBlocks. Categories are grouped using different colors. As the categories are selected, the relevant blocks will be listed in the Blocks Palette (Field 3).

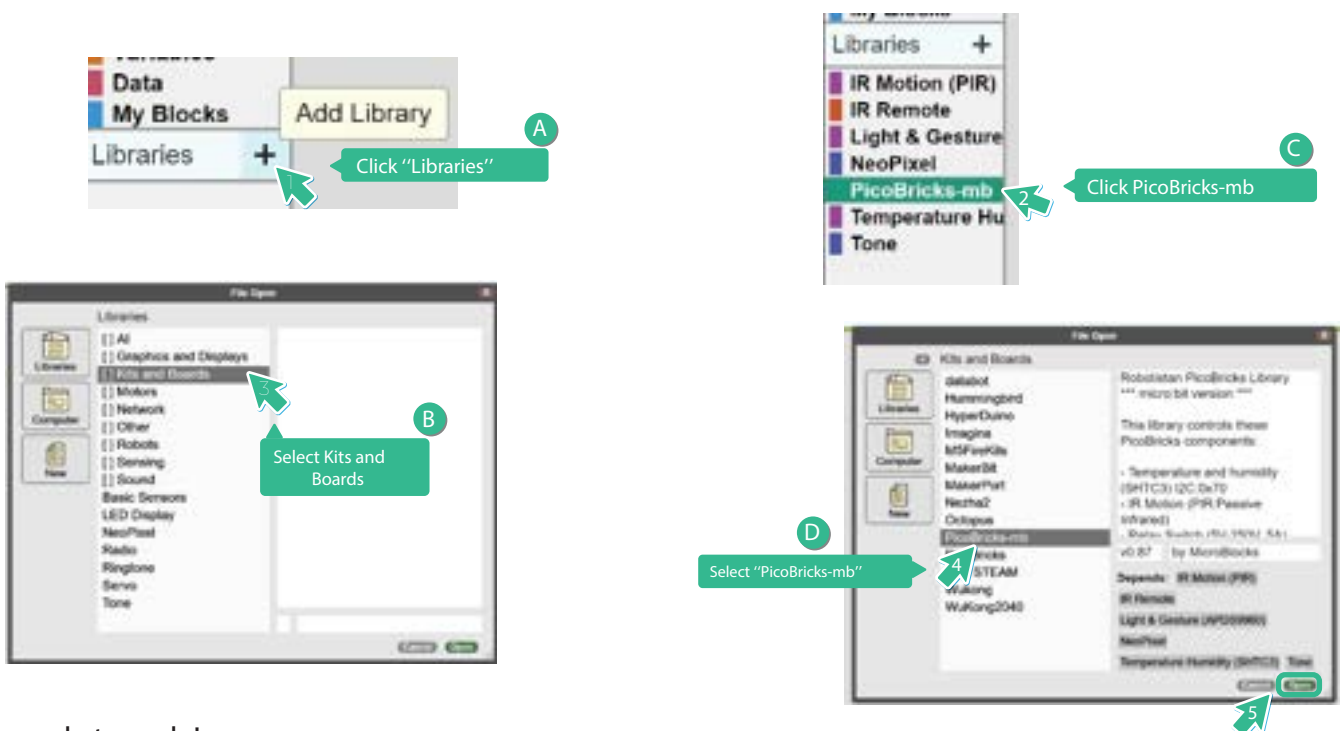
3. Blocks Palette : As selections are made in the Block categories field, blocks with specific functions will be listed in this field. code are written by dragging and dropping the blocks in this area to the Scripting area number

4. Scripting Area: This is the area where all coding activities take place. Users drag and drop blocks into this area to create scripts and custom blocks (functions).

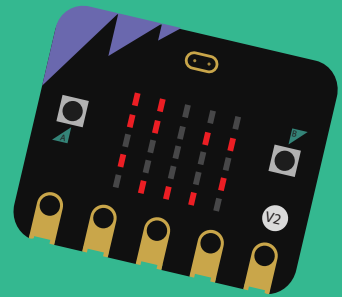
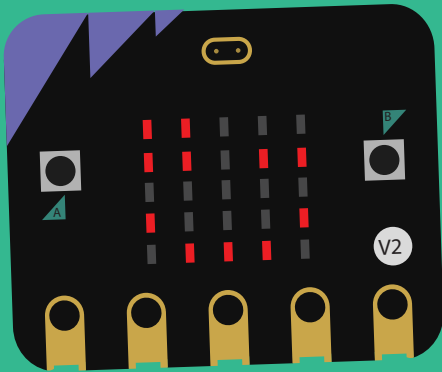
5. Start/Stop Buttons ( :  This area contains two icons, Start and Stop, which are used to control the MicroBlocks programs.

6. Library List ( Libraries  The contents of this area reflects the various libraries that are loaded depending on the requirements of the user scripts and micro devices.

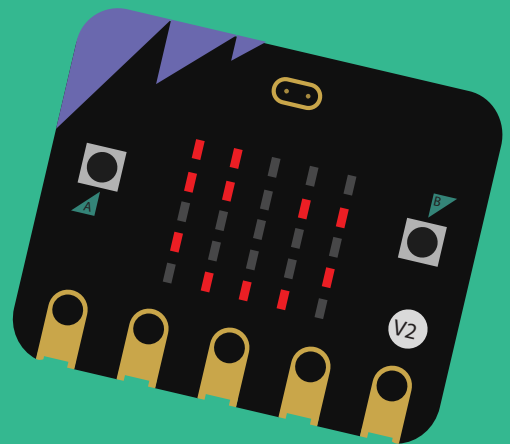
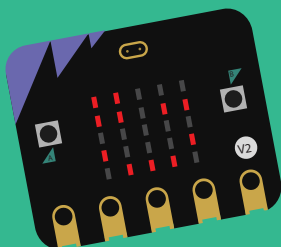
Adding the PicoBricks Library



You're ready to code!



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!"

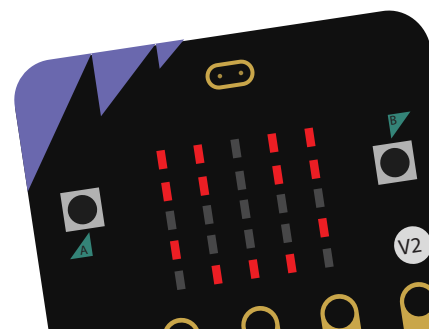
LEDs and screens are electronic circuit components that provide visual output. Thanks to output elements, a programmer can concretely determine at which stage their code is progressing. With PicoBricks, Micro:Bit includes 25 LEDs (5x5 Matrix) and a 128x64 OLED screen. When starting robotic coding with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

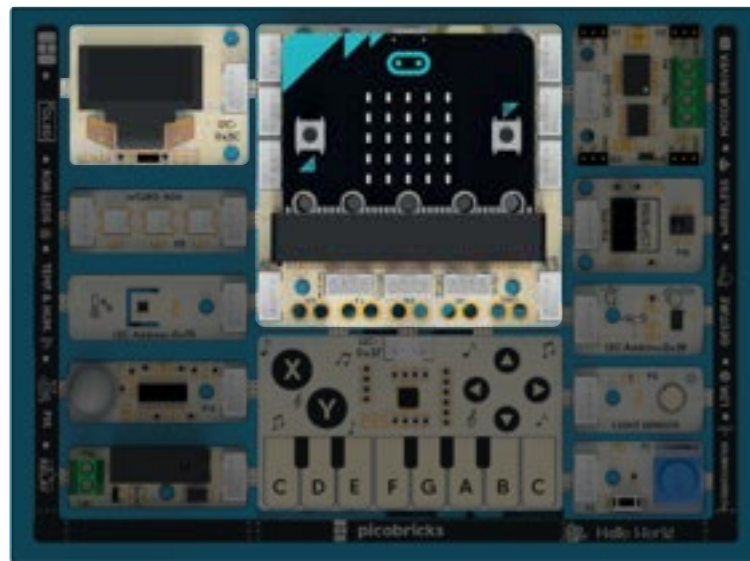
● Project Details:

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.

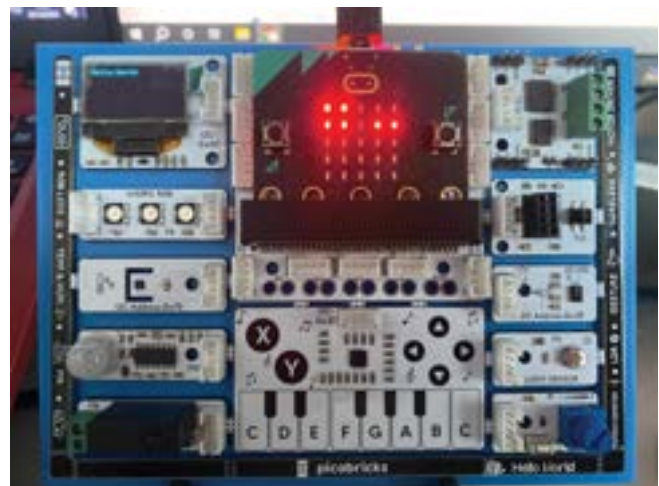
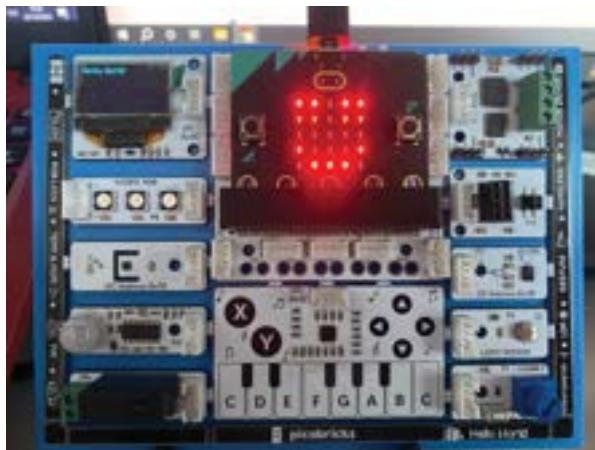
● Connection Diagram:

You can prepare this project without making any wiring connections.

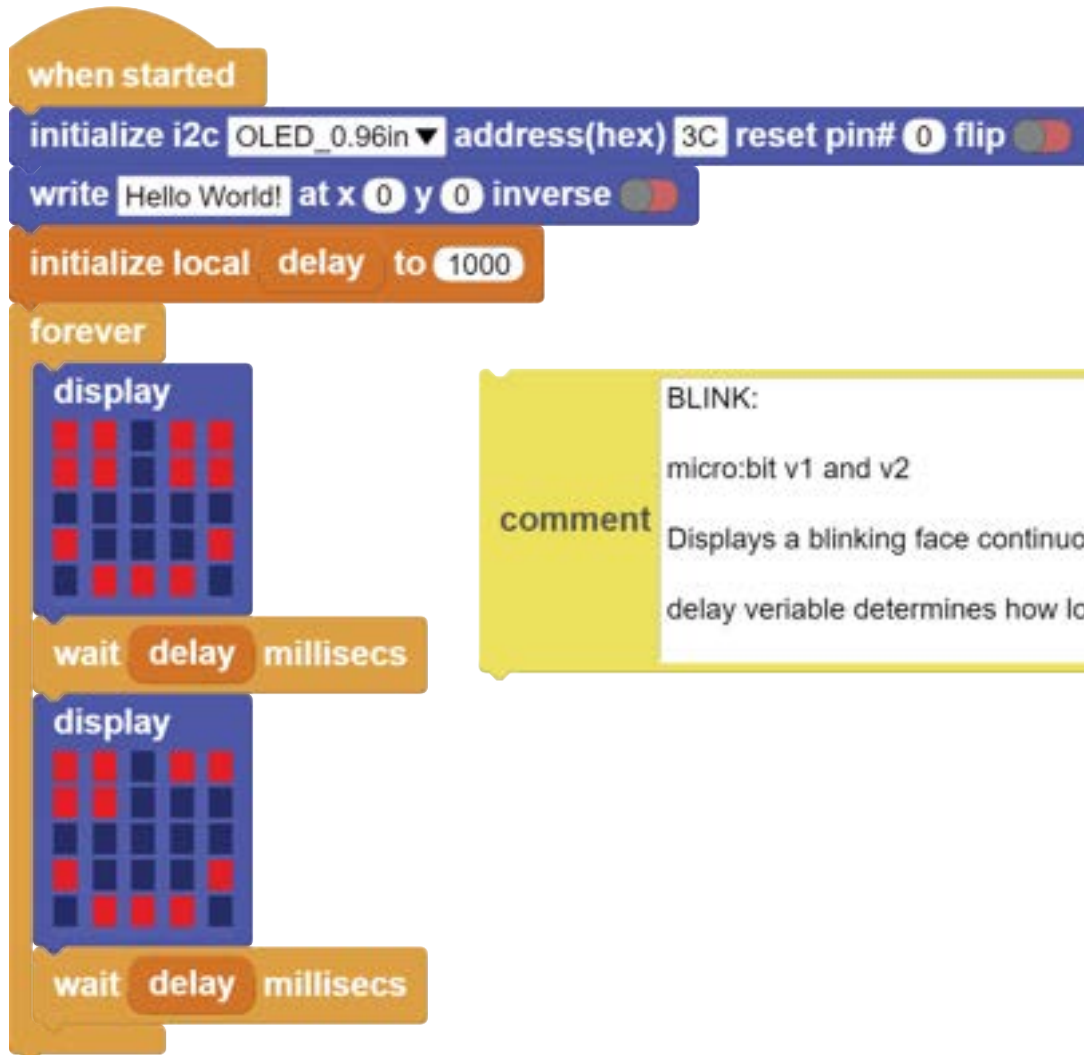




Project Images:



MicroBlocks Code of The Project:



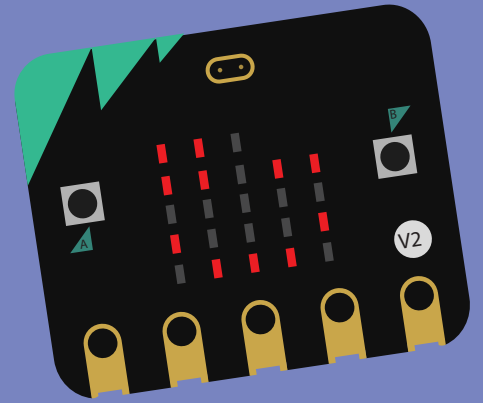
comment

BLINK:

micro:bit v1 and v2

Displays a blinking face continuously.

delay variable determines how long each blink lasts.



Action Reaction



Action-Reaction Project

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.

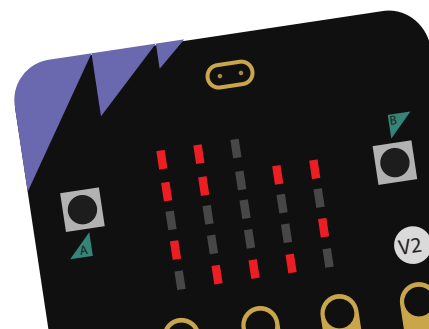
Buttons are circuit components through which we can provide input. Different types of buttons are used in electronic systems: toggle switches, push buttons and more. PicoBricks has a total of 3 push buttons, with 1 on the potentiometer and button module and 2 on the Micro:Bit. Push buttons function similarly 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.

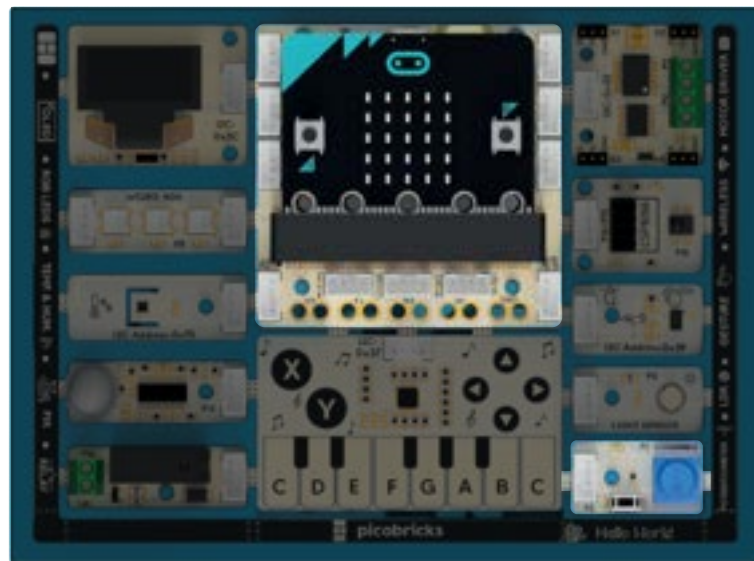
Project Details:

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.

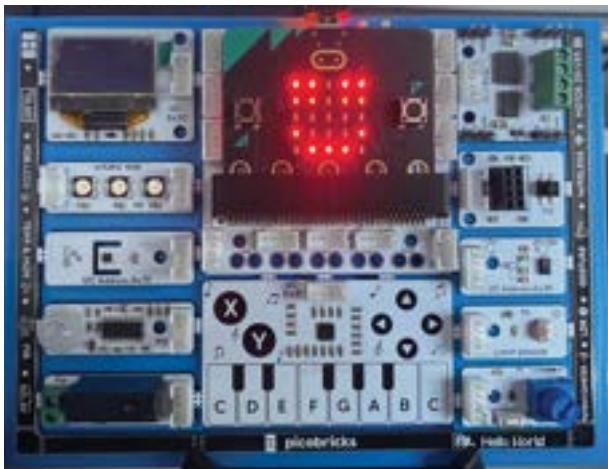
Connection Diagram:

You can prepare this project without making any cable connections.

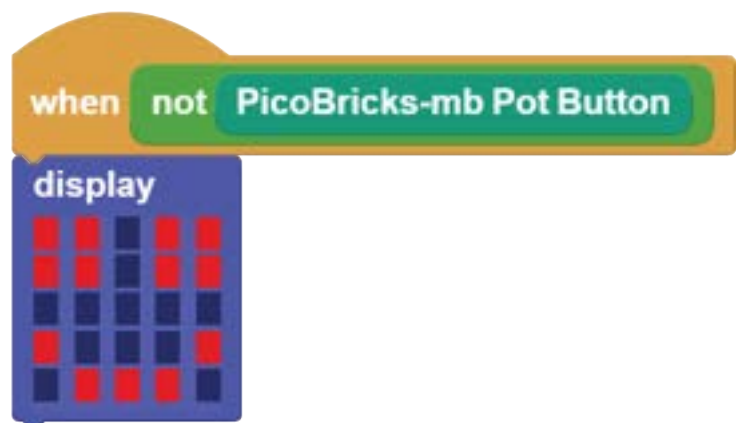
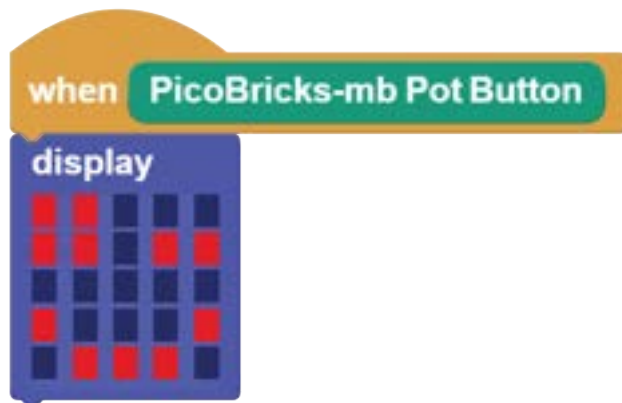
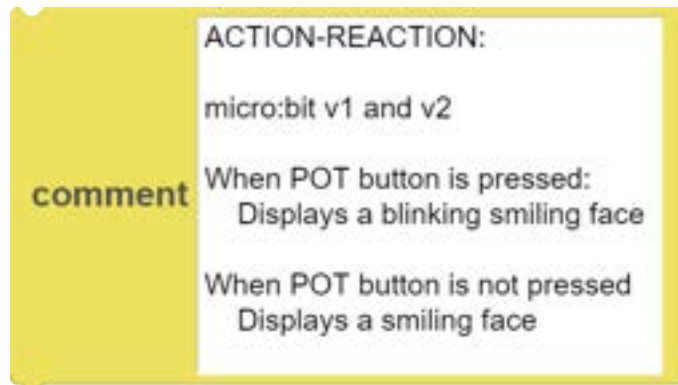




Project Images:



MicroBlocks Code of The Project:



The image features a teal background with four overlapping rectangular cards. From left to right, the cards are red, dark blue, yellow, and dark green. The text 'COLOR CARDS' is centered over the cards in a white, bold, sans-serif font. The cards are slightly offset, with the yellow card being the most prominent in the center.

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.

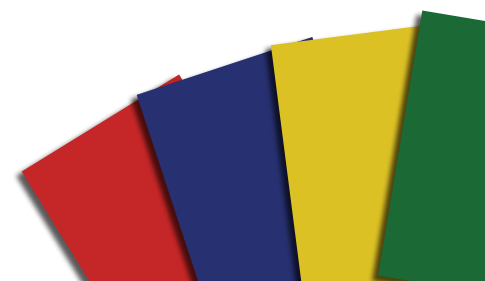
The gesture module produces three numerical outputs as R (RED), G (Green), and B (BLUE) while detecting the colors of the object in front of it. When we use these outputs as the values of the RGB LED, a single colour value is formed, and this colour is the colour of the object in front of the gesture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

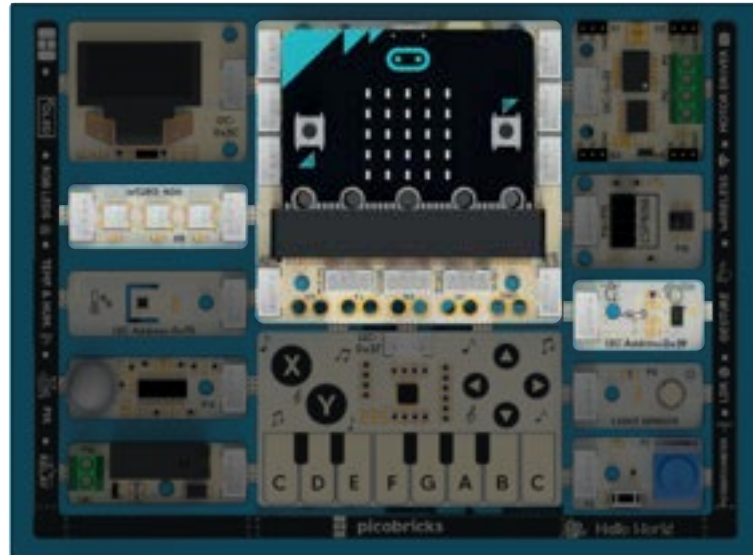
Project Details:

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:

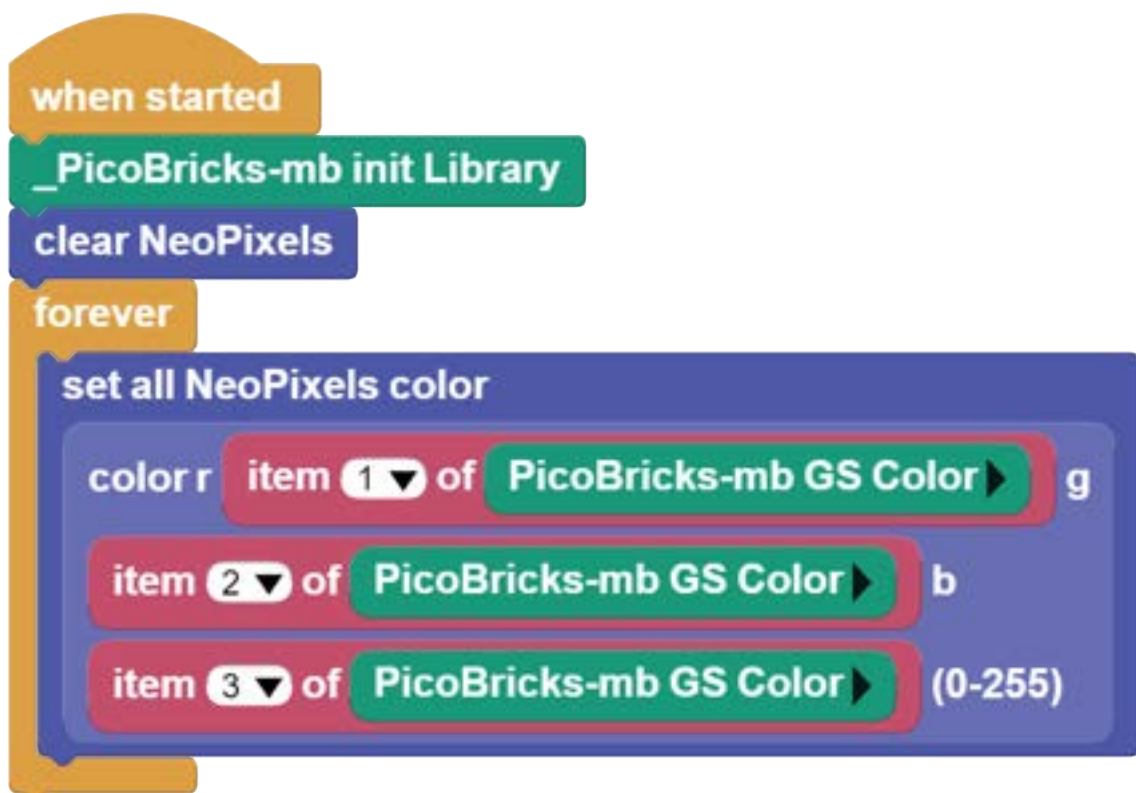
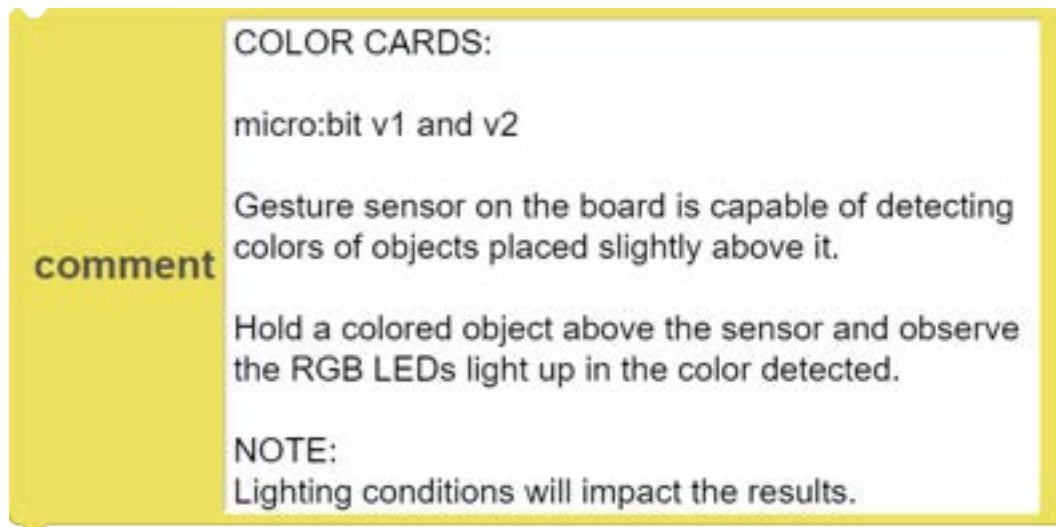
You can prepare this project without making any cable connections.



● Project Images:



MicroBlocks Code of The Project:





Piano

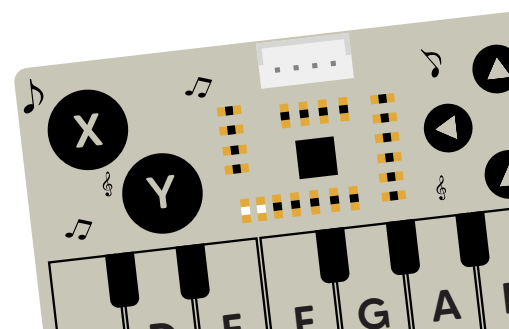
PicoBricks Piano Project

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.

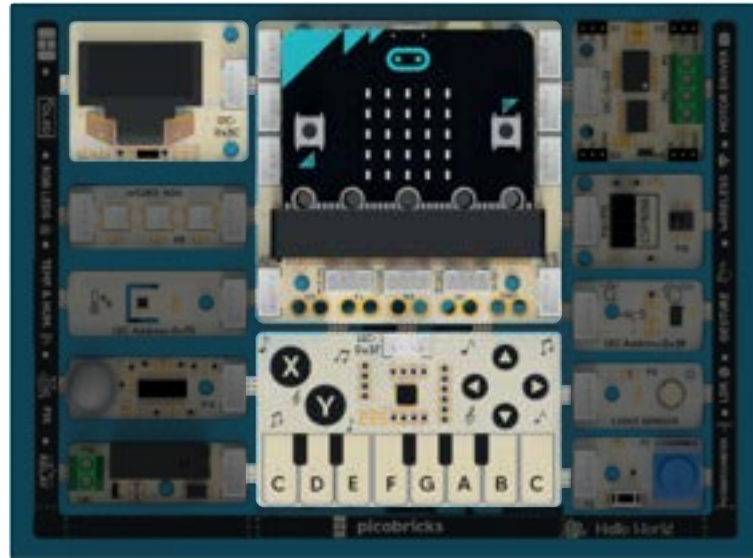
Project Details:

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.

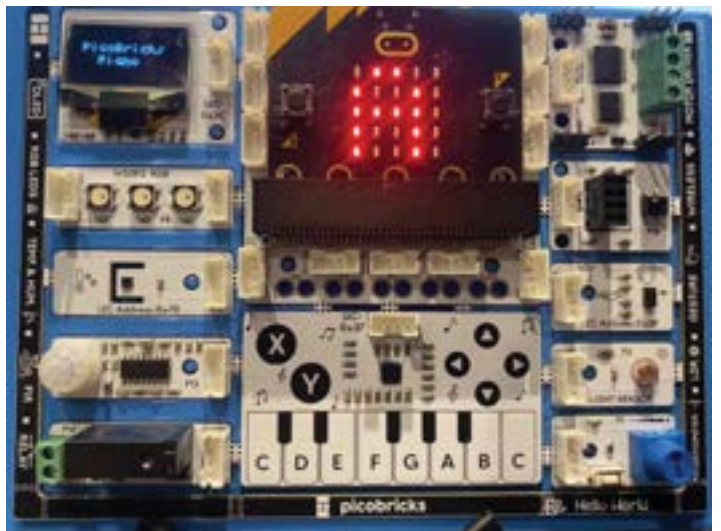
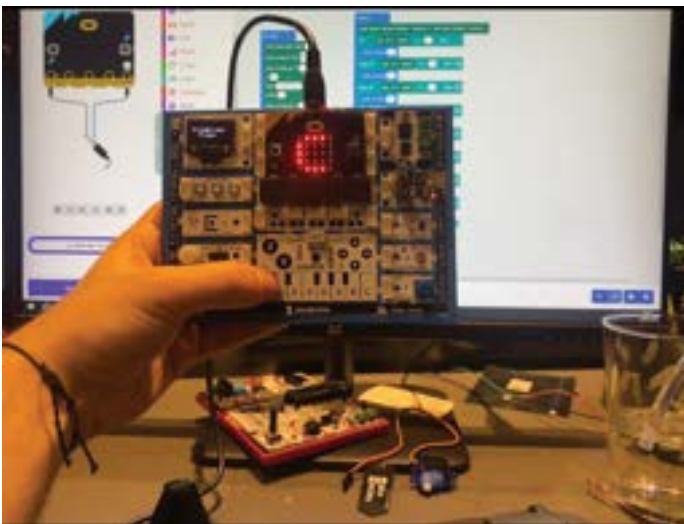


● Connection Diagram:

You can prepare this project without making any cable connections.



● Project Images:



MicroBlocks Code of The Project:

comment PICOBRICKS PIANO:
micro:bit v1 and v2
Touch sensor keyboard is used as a piano.
Touch events are filtered for the notes only.
Each note key touched emits a tone and displays the note on the LED display.

```
when PicoBricks-mb Touchkey ANY pressed ?
  initialize local notes to cdefgabc2
  if micro:bit v2 = board type
    _defer monochrome display updates
    clear display
    write PicoBricks Piano on TFT at x 5 y 5 color green scale 2 wrap
    _resume monochrome display updates
  ▶
  initialize local key to PicoBricks-mb Last key touched
  if c1 ≠ find key in notes
    if c2 = key
      play note item 1 of key octave 1 for 100 ms
    else
      play note key octave 0 for 100 ms
    ◀▶
    display character key
    wait 50 millisecs
  ▶
```



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 MicroBlocks 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.

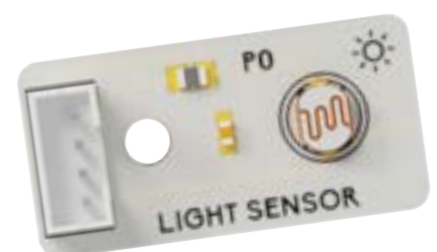
In this project, we will create a touch-sensitive piano by using the PicoBricks Touch & Piano module.

Project Details:

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.

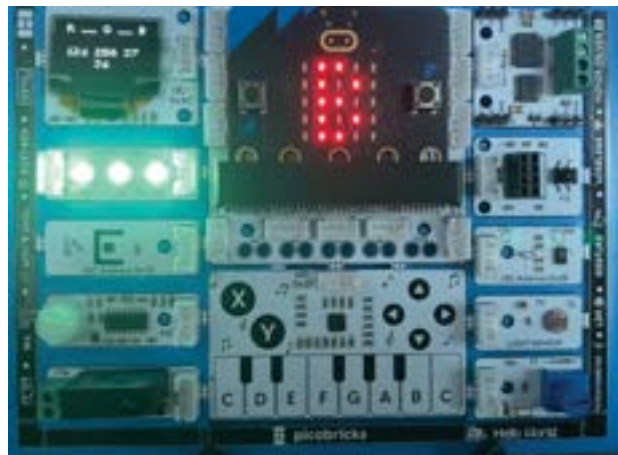
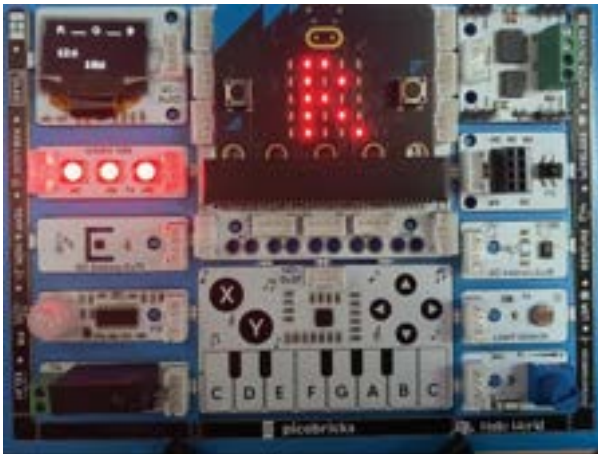
Connection Diagram:

You can prepare this project without making any cable connections.





● Project Images:



MicroBlocks Code of The Project:

RGB LED Control Panel

micro:bit v1 and v2

micro:bit A and B buttons control the RGB color selection. Press them and set it to R, G, or B.

Pot dial sets the color value in the range of 0-255.

To set a color, hold the POT button down, and while pressing it turn the POT dial to a desired number 0-255. When you see the desired number scroll on the LED display, release the button.

When you release the button, the value will be stored and you will see a check mark displayed on the LED display.

You can repeat this process as many times as you wish.

Observe at three RGB LEDs display the color combination you have created.

When started loop will always display the last setting for the RGB.

```

when started
  PicoBricks-mb Init Library
  set colorIndex to 1
  set colors to RGB
  set R to 128
  set G to 128
  set B to 128
  forever
    set all NeoPixels color: color r R g G b B (0-255)
    say R G B B
    wait 100 milliseconds
  
```

```

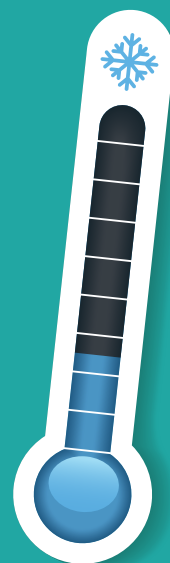
when button A pressed
  change colorIndex by 1
  if colorIndex < 1
    set colorIndex to 1
  display character item colorIndex of colors
  wait 1000 milliseconds
  
```

```

when button B pressed
  change colorIndex by 1
  if colorIndex > 3
    set colorIndex to 3
  display character item colorIndex of colors
  wait 1000 milliseconds
  
```

```

when PicoBricks-mb Pot Button
  repeat until not PicoBricks-mb Pot Button
    initialize local colorValue to
    rescale PicoBricks-mb Potentiometer from (0 - 1023) to (0 - 255)
    scroll text colorValue
    if 1 = colorIndex
      set R to colorValue
    else if 2 = colorIndex
      set G to colorValue
    else if 3 = colorIndex
      set B to colorValue
  display
  
```



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.

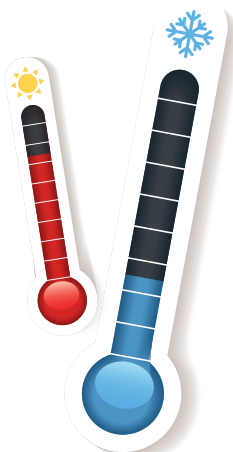
Ambient temperature is used in situations where continuous monitoring of temperature changes is required, such as in greenhouses, incubators, and environments used for transporting medications. If you are going to perform a task related to temperature changes in your projects, you should know how to 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.

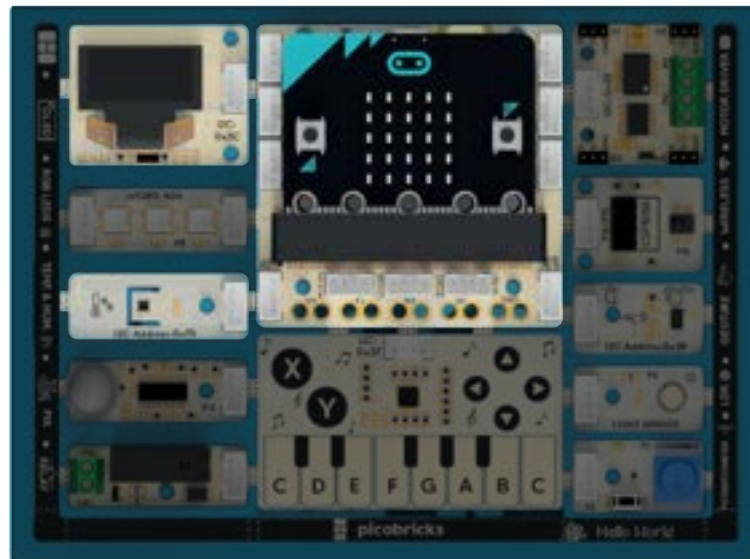
Project Details:

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:



MicroBlocks Code of The Project:

comment
THERMOMETER
micro:bit v1 and v2
micro:bit A and B buttons control the Fahrenheit vs Celcius selection for the Temperature display. Selection made will be flashed on the LED display 3x.
F or C selection will calculate the temperature value accordingly.
When "display" received script displays the Humidity and Temperature values on the LED display, as T ## and H ##.

when started
_PicoBricks-mb init Library
set tempType to C
broadcast display

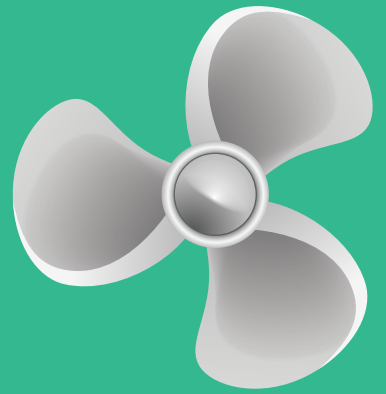
when button A pressed
stop other tasks
set tempType to C
initialize local delay to 200
repeat 3
display character tempType
wait delay millisecs
clear display
wait delay millisecs
broadcast display

when button B pressed
stop other tasks
set tempType to F
initialize local delay to 200
repeat 3
display character tempType
wait delay millisecs
clear display
wait delay millisecs
broadcast display

when display received
forever
if C = tempType
initialize local temp to PicoBricks-mb temperature (*C)
else if F = tempType
$$F = (T \times 9/5) + 32$$

comment
Since math is Integer only, we calculate a parameter for 9/5 as 1.8 and multiply it by 100 to get 180.
We use this in the formula and then divide the result by 100 to normalize.
initialize local temp to
$$\text{PicoBricks-mb temperature (*C)} \times 180 / 100 + 32$$

scroll text join T temp
scroll text join H PicoBricks-mb humidity



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.

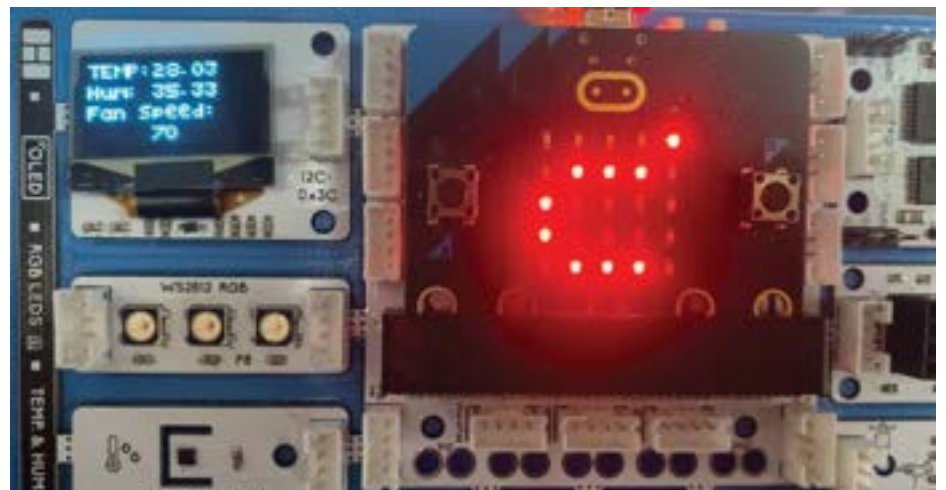
Connection Diagram:

You can prepare this project without making any cable connections.





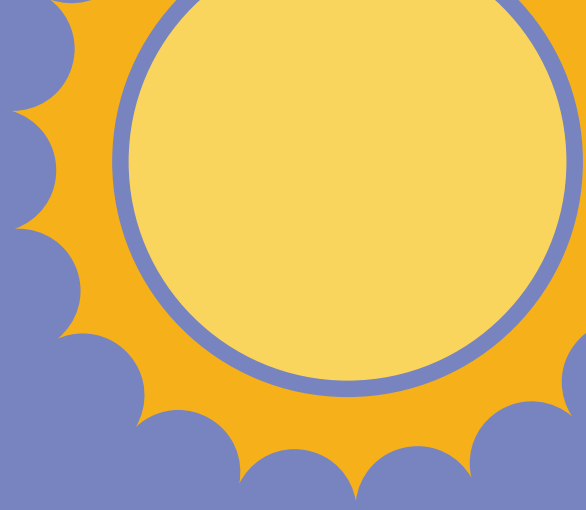
● Project Images:



MicroBlocks Code of The Project:

```
SMART COOLER:  
  
micro:bit v2  
  
comment Temp&Humidity sensor values are monitored.  
If Temp > 22 deg Celcius, Fan motor is turned on.  
Otherwise, motor is off.  
  
Values are displayed on the OLED.
```

```
when started  
  _PicoBricks-mb init Library  
  forever  
    initialize local temp to PicoBricks-mb temperature (°C)  
    set fanSpeed to rescale temp from ( 0 , 40 ) to ( 0 , 255 )  
    _defer monochrome display updates  
    clear display  
    write join TEMP: temp on TFT at x 0 y 0 color green scale 2 wrap  
    write join HUM: PicoBricks-mb humidity on TFT at x 0 y 16  
    color green scale 2 wrap  
    if temp < 22  
      write join FAN: Off on TFT at x 0 y 32 color green scale 2 wrap  
      PicoBricks-mb set motor 1 speed 0 (0-255) dir 0  
    else  
      write join FAN: fanSpeed on TFT at x 0 y 20 color green scale 2  
      wrap  
      PicoBricks-mb set motor 1 speed fanSpeed (0-255) dir 0  
    _resume monochrome display updates  
    wait 500 millisecs
```



Night and Day



Night and Day Project

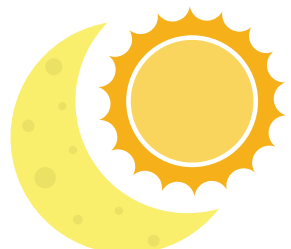
How about playing the Day and Night game electronically, a game often played in schools?

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:

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.



You can prepare this project without making any cable connections.



MicroBlocks Code of The Project:

RIGHT AND DAY:
score to 0
LDR light sensor is used as an input to the DAY & NIGHT prompt
User needs to cover the LDR when NIGHT is displayed and uncover it when DAY is displayed
comment Correct inputs score 10 points, wrong ones terminate the game
100 points a perfect score
User can press the Pot button to restart the game.
NOTE
You may need to adjust the sensor value for your environment for proper selection of day and night.

```
when started
  Picabricks-mk test Library
  set score to 0
  set promptOver to 0
  set score to 0
  clear display
  write "Press POT button to START" on TFT at x 0 y 0 color 0 scale 1 wrap 1
```

```
when Picabricks-mk Pot Button
  wait until not Picabricks-mk Pot Button
  set score to 0
  set promptOver to 0
  set score to 0
  broadcast start
```

```
when error
  defer monochrome display updates
  clear display
  write "Game Over" on TFT at x 0 y 0 color 0
  write "join SCORE score" on TFT at x 0 y 0 color 0 scale 1
  wrap 1
  resume monochrome display updates
```

```
when gameOver
  defer monochrome display updates
  clear display
  write "Game Over" on TFT at x 0 y 0 color 0
  write "join SCORE score" on TFT at x 0 y 0 color 0 scale 1
  wrap 1
  resume monochrome display updates
```

```
when start received
  initialize local LDRthreshold to 50
  initialize local delay to 1000
  repeat until gameOver == true
    initialize local prompt to item 0 of list DAY NIGHT
    clear display
    write prompt on TFT at x 0 y 0 color 0 scale 1 wrap 1
    wait delay milliseconds
    [day = prompt and
    Picabricks-mk light sensor (0-100) % > LDRthreshold]
    change score by 10
    display
    wait delay milliseconds
    else if [night = prompt and
    Picabricks-mk light sensor (0-100) % < LDRthreshold]
    change score by 10
    display
    wait delay milliseconds
    else
    set score to 0
    display
    wait delay milliseconds
    score += 100
    set promptOver to 1
```

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.

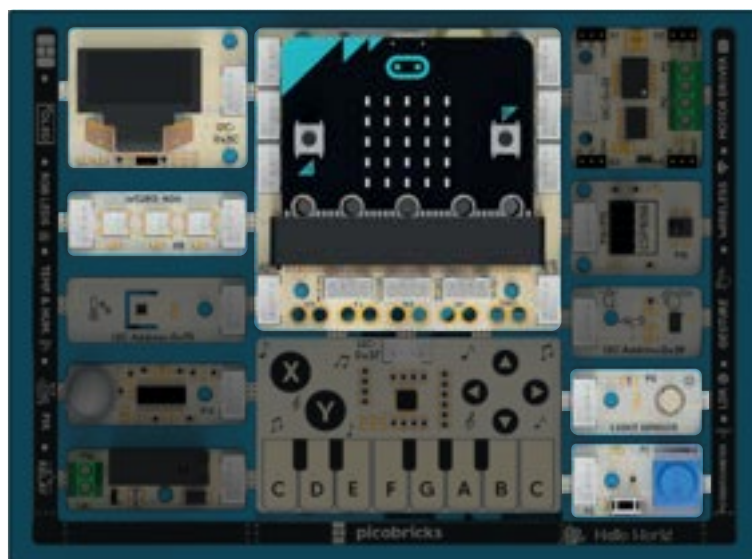
Thanks to PicoBricks, we can code this game by using electronic components and experience it physically.

Project Details:

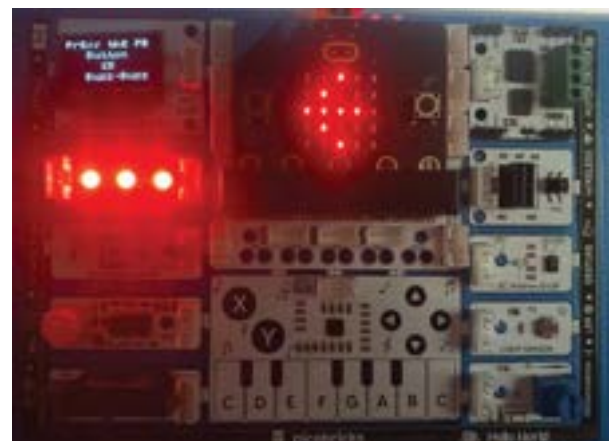
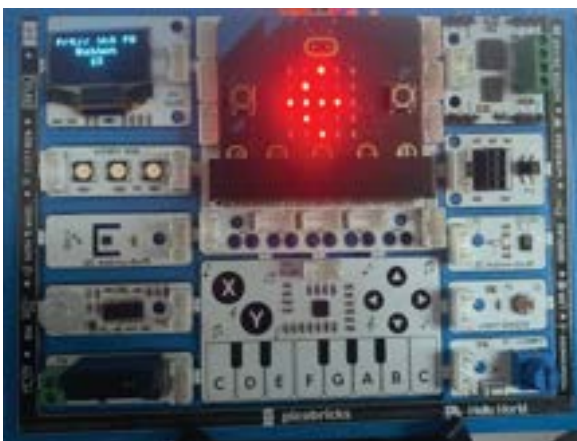
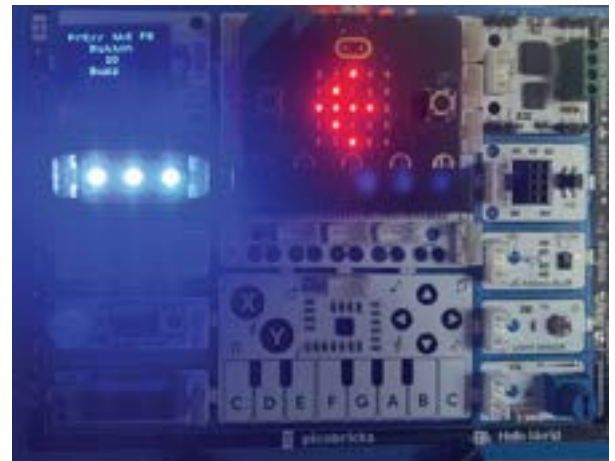
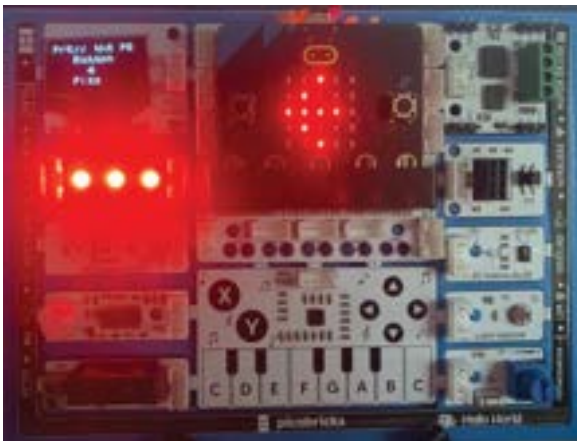
In this project, we will create the Fizz-Buzz game algorithm and code using PicoBricks along with the button, RGB LED, and OLED screen module with Micro:Bit. The Fizz-Buzz game is played by counting numbers from 1 to 100. Starting from 1, when a number that is a multiple of 3 is reached, "Fizz" is said.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



MicroBlocks Code of The Project:

```

when started
  _PicoBricks-mb init Library
  set i to 0
  repeat 999
    clear display
    if i mod 3 = 0 and i mod 5 = 0
      display
      play note f octave 0 for 75 ms
      play note b octave 0 for 75 ms
      clear NeoPixels
      set NeoPixel 2 color blue
      set NeoPixel 3 color green
    else if i mod 3 = 0
      display character F
      play note f octave 0 for 100 ms
      clear NeoPixels
      set NeoPixel 2 color blue
    else if i mod 5 = 0
      display character B
      play note b octave 0 for 100 ms
      clear NeoPixels
      set NeoPixel 3 color green
    else
      display
      clear NeoPixels
      set NeoPixel 3 color red
    write i on TFT at x 1 y 5 color green scale 7 wrap
    wait until PicoBricks-mb Pot Button
    wait 200 milliseconds
    change i by 1
  
```

comment
FIZZ BUZZ GAME
 Algorithm to locate numbers between 0-100 that are mod 3, mod 5, and both.
 OLED display shows incrementing numbers 0-100 at each press of the POT button.
 mod 3 is FIZZ - plays note F, RGB LED = blue, LED = F
 mod 5 is BUZZ - plays note B, RGB LED = green, LED = B
 both are FIZZ-BUZZ - plays notes F & B, RGB LED = green and blue, LED = FB
 Other conditions display LED = X

Depth Meter



Depth Meter Project

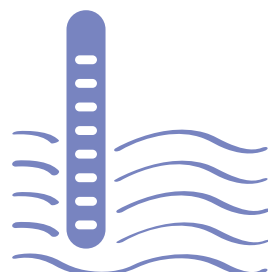
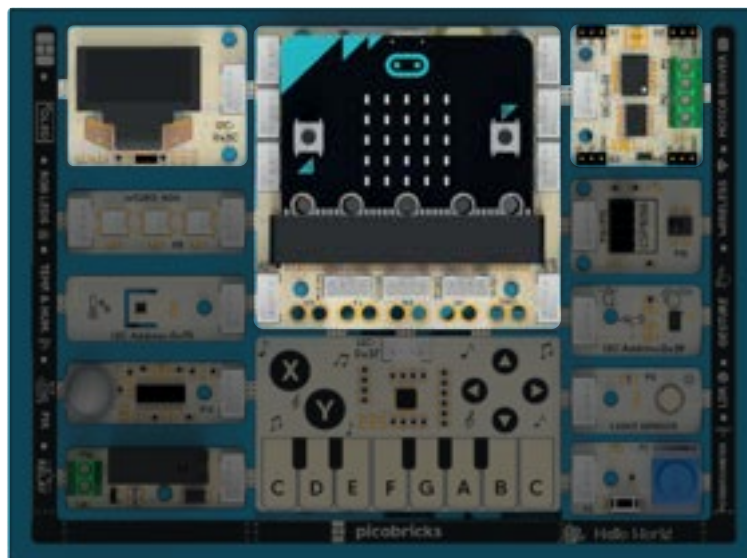
Sometimes, we use depth-measuring machines to measure the quantity of a beverage or mixtures of liquid materials poured into a glass. The fundamental variable that needs to be known for these machines to measure depth is the depth value measured when the container is empty. After defining this information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

Project Details:

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:



MicroBlocks Code of The Project:

```
comment
DEPTH METER:
micro:bit v2

This project controls a pump that transfers water from one glass to another.

Desired water level is set using the Potentiometer. When the value is set, the Pot button
is pressed to record the value and to start the water transfer.

Electric motor pump, attached to the M1 connection, is turned on.
As the water is pumped, the water level is monitored with the distance sensor.
Water level is displayed on the OLED screen.

When the water level reaches the set level, the pump is shut down.

WATER LEVEL WARNING:
Please make sure to test the glassEmptyValue and the desired waterLevel values before
running the project for real. They should be adjusted according to the glasses you will use
in your project.

glassEmptyValue - obtained when the distance sensor value is read with no water in the glass.
waterLevel - desired cm of water height, measured from the table top where the glass is placed.
Incorrect selections may cause a water/electric hazard.
```

```
when started
  _PicoBricks-mb init Library
  comment Set the water level and glass empty read value.
  set glassEmptyValue to 11
  repeat until PicoBricks-mb Pot Button
    set waterLevel to
      rescale PicoBricks-mb Potentiometer from ( 0 , 1023 ) to ( 0 , 8 )
    _defer monochrome display updates
    clear display
    write join Set Water Level: waterLevel on TFT at x 0 y 0 color scale
    2 wrap
    _resume monochrome display updates
    wait 500 millisecs
  clear display
  broadcast got
```

```
when got received
  comment Activate the pump until desired water level is reached.
  repeat until
    waterLevel <=
      glassEmptyValue - distance (cm) trigger 1 echo 2
    PicoBricks-mb set motor 1 speed 128 (0-255) dir 0
    _defer monochrome display updates
    clear display
    write
      join Water Level: glassEmptyValue - distance (cm) trigger 1 echo 2
    on TFT at x 0 y 0 color scale 2 wrap
    _resume monochrome display updates
    wait 200 millisecs
  PicoBricks-mb set motor 1 speed 0 (0-255) dir 0
```

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:

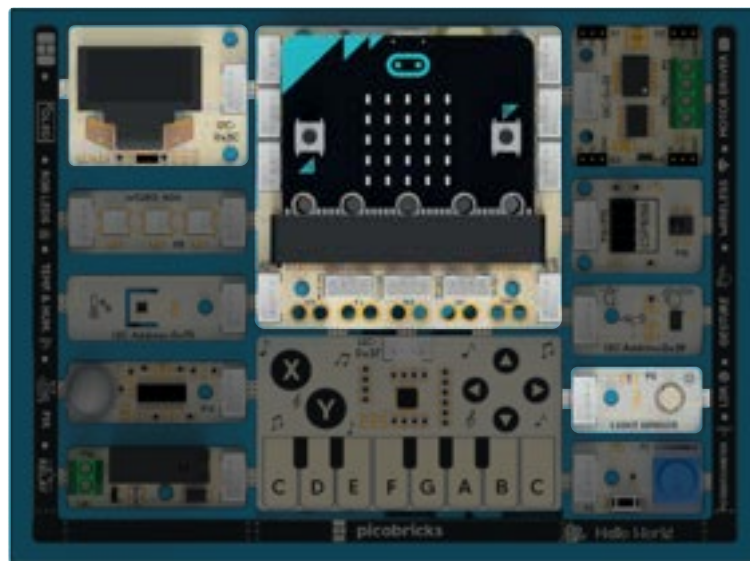
A • —	B — • • •	C — • — •	D — • •	E •
F • • — •	G — — •	H • • • •	I • •	J • — — —
K — • —	L • — • •	M — —	N — •	O — — —
P • — — •	Q — — • —	R • — •	S • • •	T —
U • • —	V • • • —	W • — —	X — • • —	Y — • — —
Z — — • •				

Project Details:

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:



MicroBlocks Code of The Project:

comment
MORSE CODE:
 micro:bit v1 and v2
 This morse code program will work with letters A-Z and numbers 0-9, and the space character.
 Playback speed can be set using an equivalency to words per minute (wpm).
 WPM value input is converted to unitDelay, which is the speed of a single dit or dot.
 It will display each letter of the TEXT on the LED panel for 1 sec.
 followed by the corresponding 1 to 5 symbol morse code version.
 Then it will play out the code in tone and as a blinking LED.
 If the pauseBetweenLetter is set to FALSE, then morse code playback will be non-stop at true speed.
 short mark, dot or dit (): "dit duration" is one time unit long
 long mark, dash or dah (): three time units long
 inter-element gap between the dits and dahs within a character: one dot duration or one unit long
 short gap (between letters): three time units long
 medium gap (between words): seven time units long

define setupCodes
 set symbols to ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
 list
 set morseCodes to

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	..--	1	.-.-.-	2	..-.-	3	...-	4		5	...	6	-...	7	-.-.-	8	-.-.-	9	-.-.-	0	----
---	----	---	-------	---	-------	---	-------	---	-----	---	-------	---	------	---	------	---	----	---	----	---	--------	---	------	---	-----	---	-----	---	-----	---	------	---	--------	---	------	---	------	---	----	---	-----	---	------	---	------	---	------	---	------	---	------	---	--------	---	-------	---	------	---	------	---	-----	---	------	---	-------	---	-------	---	-------	---	------

when started
 _PicoBricks-mb init Library
 setupCodes
 set all NeoPixels color random color
 wait 1000 milliseconds
 clear NeoPixels
 set pauseBetweenLetters to TRUE
 set wpm to 15
 set unitDelay to 1200 / wpm
 initialize local text to HELLO
 for symbol in text
 if  = symbol
 wait 1 x unitDelay milliseconds
 else
 initialize local idx to find symbol in symbols
 set code to item idx of morseCodes
 display character symbol
 if pauseBetweenLetters
 wait 1000 milliseconds
 displayMorse
 call play&FlashMorse
 wait 3 x unitDelay milliseconds
 end for

define displayMorse
 initialize local row to 1
 clear display
 for symbol in code
 if  = symbol
 plot x 1 y row
 else if  = symbol
 plot x 1 y row
 plot x 2 y row
 plot x 3 y row
 change row by 1
 end for

define play&FlashMorse
 for symbol in code
 if  = symbol
 call flashMorse with list unitDelay
 _toneLoop 440000 for unitDelay ms
 else if  = symbol
 call flashMorse with list unitDelay
 _toneLoop 440000 for 3 x unitDelay ms
 wait unitDelay milliseconds
 end for

define flashMorse delay
 set all NeoPixels color
 wait delay milliseconds
 clear NeoPixels

Car Parking System



Car Parking System Project

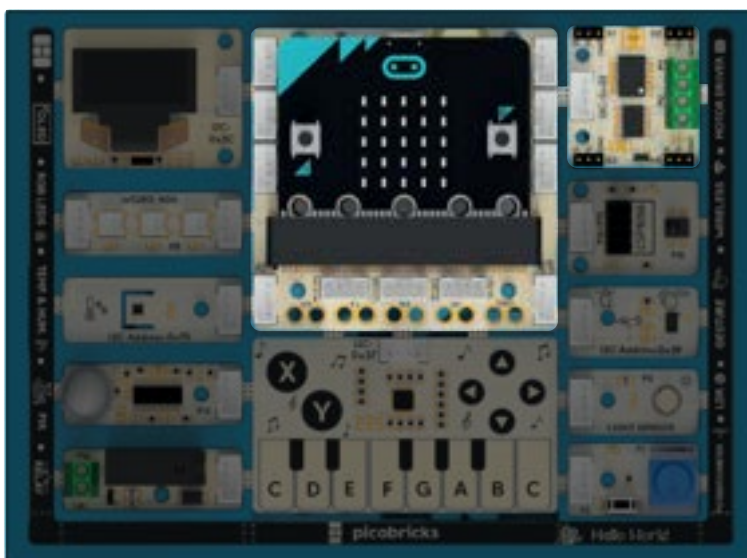
Today, buildings such as hospitals, schools, business centers, etc., often have open or closed parking lots where a large number of people enter and exit. The main reason for the construction of these parking lots is the significant increase in automotive usage in cities. Barrier systems are installed 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:

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.



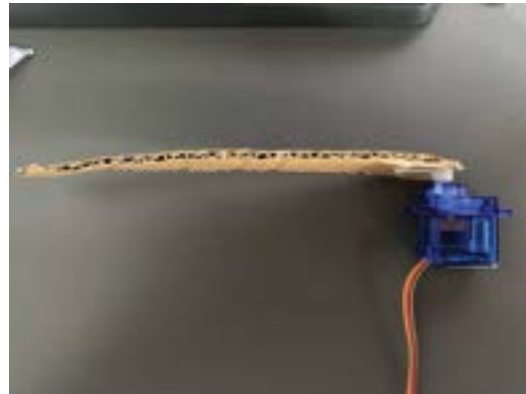
HC-SR04



Servo Motor



Project Images:



MicroBlocks Code of The Project:

```
CAR PARKING SYSTEM:  
  
micro:bit v1 and v2  
  
comment The car park gate is operated by a servo motor.  
The gate has a distance sensor that monitors approaching  
car distances.  
If the car is less than the detection distance away, then  
the gate opens and a checkmark is displayed on the LED  
display. Otherwise gate is closed and an X is displayed.
```

```
when started  
  initialize local detectionDist to 10  
  initialize local delay to 3000  
  forever  
    if distance (cm) trigger 1 echo 2 <= detectionDist  
      PicoBricks-mb set servo 3▼ angle 180 (0-180)  
      display  
      wait delay millisecs  
    else  
      PicoBricks-mb set servo 3▼ angle 90 (0-180)  
      display  
    wait 500 millisecs
```



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.)

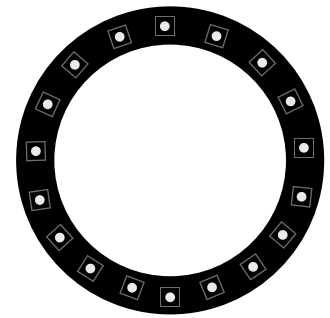
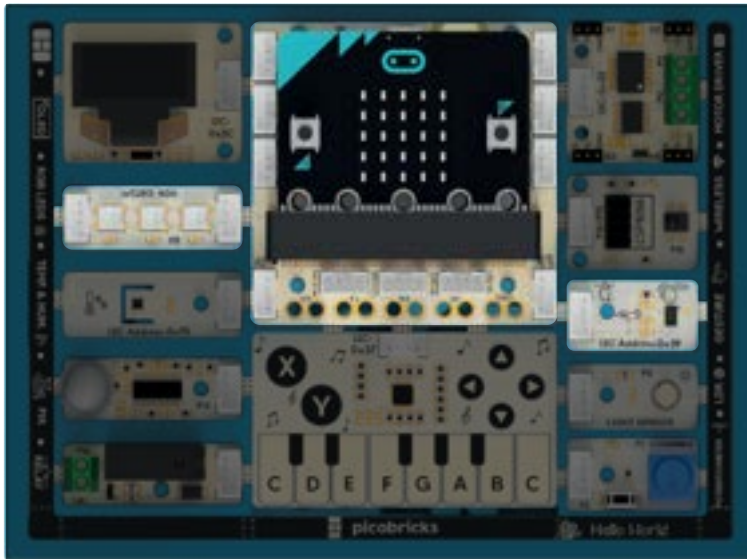
● Project Details:

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:

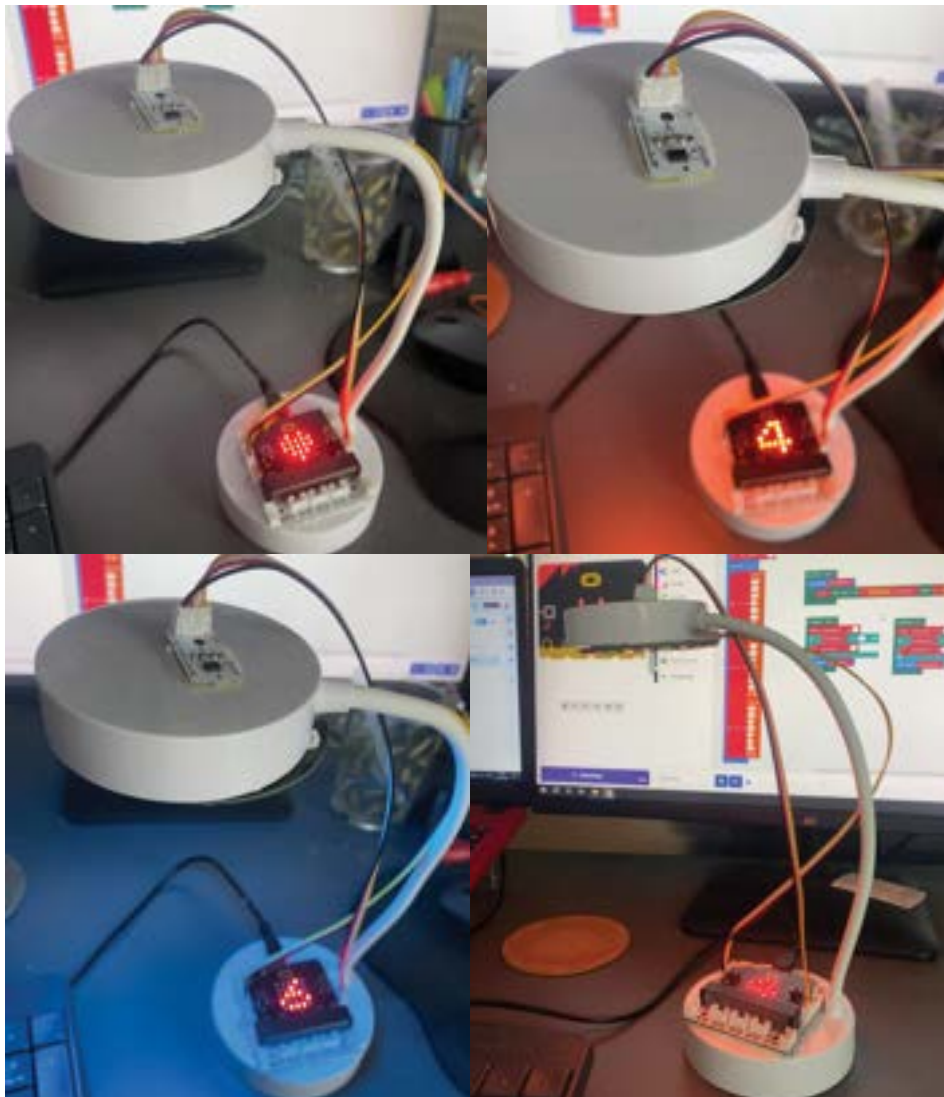
You can prepare this project by breaking PicoBricks modules at proper points.





Addressable Ring
RGB LEDs

Project Images:



MicroBlocks Code of The Project:

```

PICOBRICKS TABLE LAMP:

micro:bit v1 and v2

A Table Lamp that can display seven colors is controlled by
the gesture sensor.

The gesture sensor detects hand movements in four directions
and actions them as such:

comment
Down: reduce coloridx number and display it on LED
Up: increase coloridx number and display it on LED
Right: turn off all NeoPixels and display a Heart on LED
Left: Set all NeoPixels to the color selected by the coloridx.

NOTE:
If you have a strip with more than 3 neopixels, you need to set the
numOfNeoPixels variable accordingly.
  
```

```

when started
  _PicoBricks-mb init Library
  clear NeoPixels
  set numOfNeoPixels to 3
  set coloridx to 1
  set colors to
    list color r 255 g 255 b 255 (0-255)
    color r 128 g 135 b 193 (0-255) color r 188 g 0 b 0 (0-255)
    color r 62 g 177 b 136 (0-255) color r 139 g 50 b 0 (0-255)
    color r 255 g 60 b 0 (0-255) color r 18 g 168 b 168 (0-255)
  if numOfNeoPixels > 3
    attach numOfNeoPixels LED NeoPixel strip to pin
      _pbmb_pin_RGB_LED
  display character coloridx
  
```

```

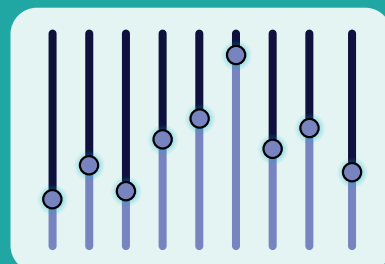
when PicoBricks-mb GS Detected
  if Down = PicoBricks-mb GS Last Gesture
    change coloridx by -1
    set coloridx to max coloridx 1
    display character coloridx
  else if Up = PicoBricks-mb GS Last Gesture
    change coloridx by 1
    set coloridx to min coloridx 7
    display character coloridx
  else if Right = PicoBricks-mb GS Last Gesture
    clear NeoPixels
    display
  else if Left = PicoBricks-mb GS Last Gesture
    set all NeoPixels color item coloridx of colors
  
```



CONTROL PANEL

IoT Control Panel

Control Panel A



IoT Control Panel Project

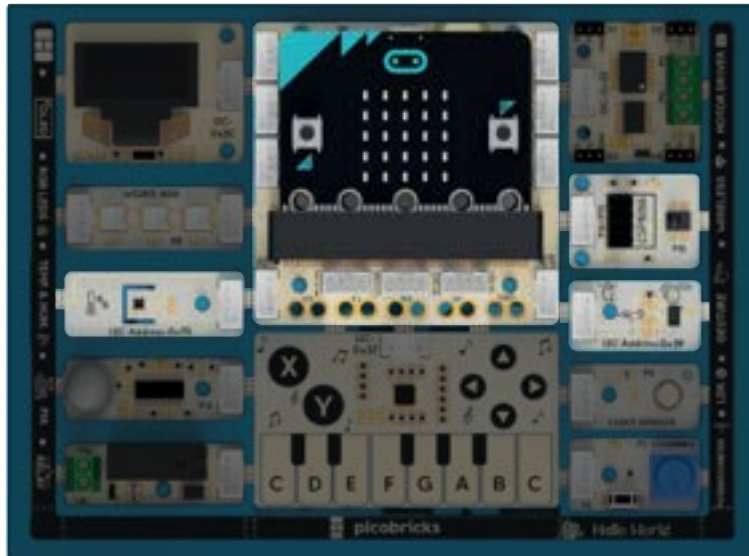
With advancing technology, electronic devices can now communicate with each other. For example, we can control our electronic home appliances such as washing machines, dishwashers, ovens, etc., from our phones. The ability of electronic devices to communicate with each other through internet networks 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:

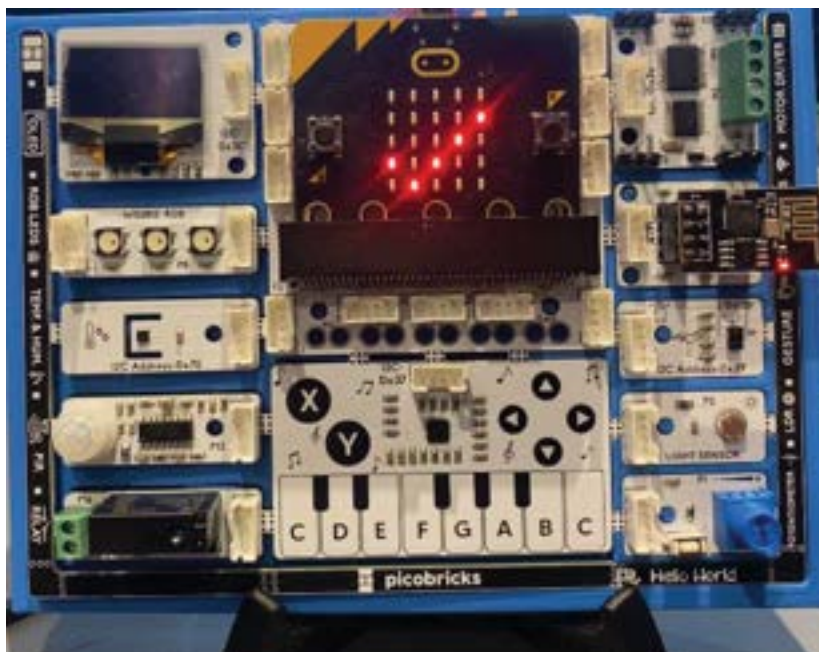
In this project, we will write the data obtained from PicoBricks modules; TEMP&HUM, PIR, LDR, and the Proximity feature in the Gesture module, to a channel created in ThingSpeak and control the changes in the values.

● Connection Diagram:

You can prepare this project without making any cable connections.



● Project Images:



IoT Control Panel Project

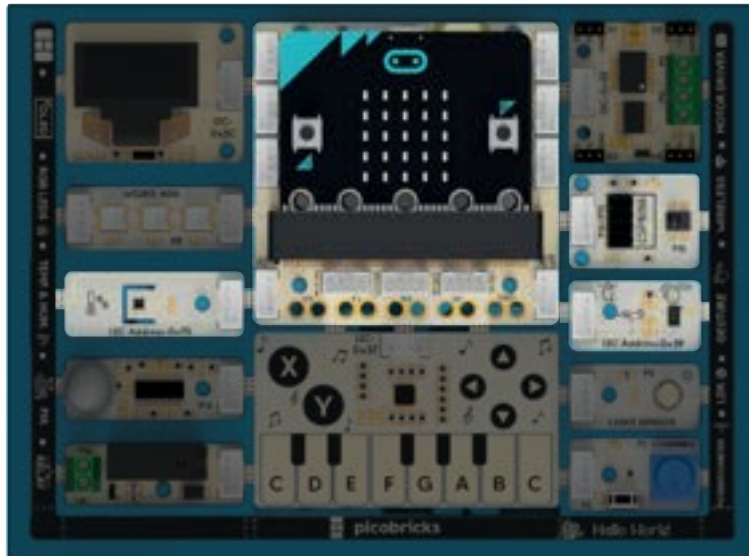
With advancing technology, electronic devices can now communicate with each other. For example, we can control our electronic home appliances such as washing machines, dishwashers, ovens, etc., from our phones. The ability of electronic devices to communicate with each other through internet networks 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:

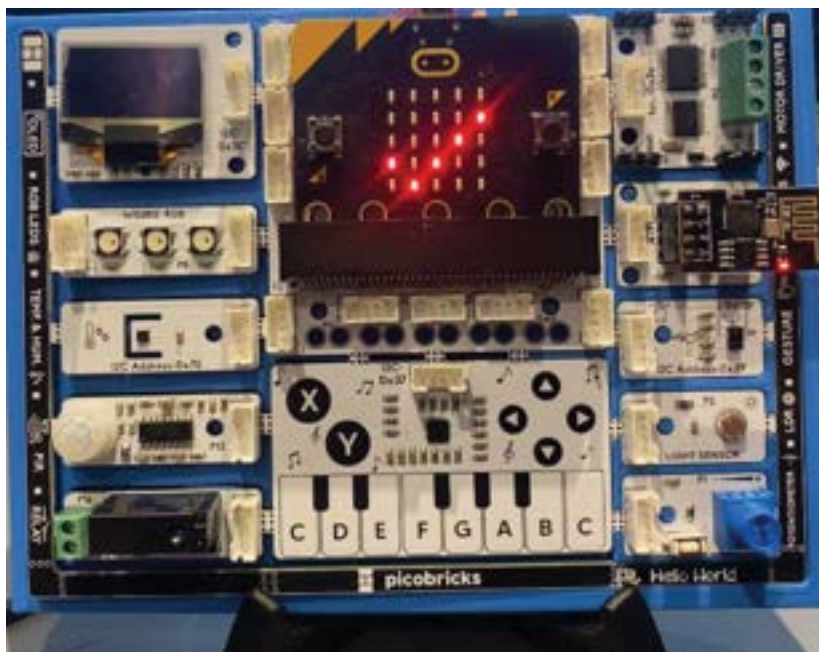
In this project, we will write the data obtained from PicoBricks modules; TEMP&HUM, PIR, LDR, and the Proximity feature in the Gesture module, to a channel created in ThingSpeak and control the changes in the values.

● Connection Diagram:

You can prepare this project without making any cable connections.



● Project Images:



PicoBricks IoT Control Panel

Channel ID: 2403088
Author: PicoBricks (4271)
Private: No

Private Data Public Data Channel Settings Sharing Widgets Data Import/Export

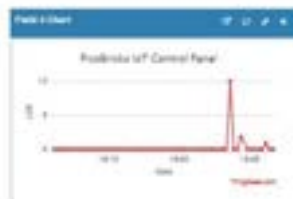
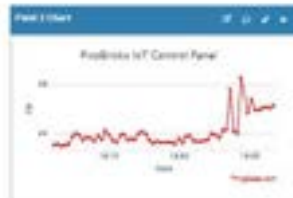
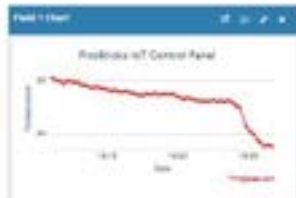
IoT Channel ID: 2403088 Add Property Add Channel Data

IoT API Analytics IoT API Visualization

Channel 2 of 3

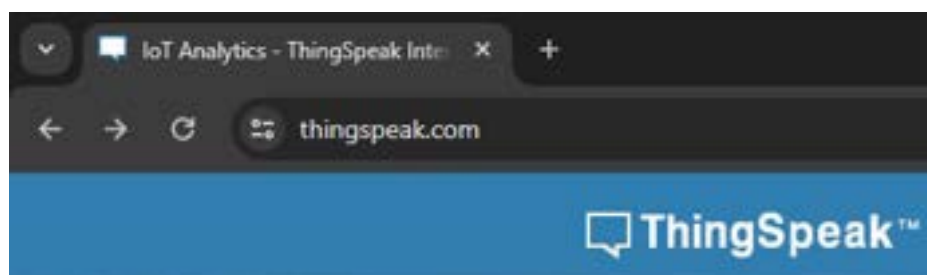
Channel Stats

Created: 2020/03/03 00:00
Last Edited: 2020/03/03 00:00
Active: 2020

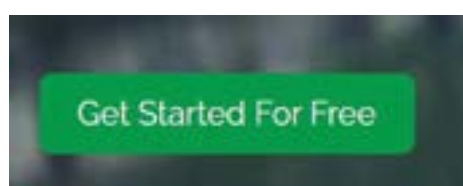


Usage of ThingSpeak:

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.



To use ThingSpeak, you must sign in with your existing MathWorks account or create a new one.

Non-commercial users may use ThingSpeak for free. Free accounts offer limits on certain functionality. Commercial users are eligible for a time-limited free evaluation. To get full access to the MATLAB analysis features on ThingSpeak, log in to ThingSpeak using the email address associated with your university or organization.

To send data faster to ThingSpeak or to send more data from more devices, consider the paid [license options](#) for commercial, academic, home and student usage.



Email

No account? [Create one!](#)

By signing in, you agree to our [privacy policy](#).

Next



Create MathWorks Account

Email Address

picobricks@example.com



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

Location

United States



First Name

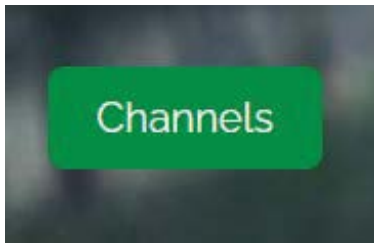
Last Name

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" (filled with "PicoBricks IoT Control Panel"), "Description" (empty), "Field 1" (filled with "Temperature" and a checkbox), "Field 2" (filled with "Humidity" and a checkbox), "Field 3" (filled with "pH" and a checkbox), "Field 4" (filled with "Proximity" and a checkbox), "Field 5" (filled with "LED" and a checkbox), "Field 6" (filled with "Potentiometer (1-1000)" and a checkbox), "Field 7" (empty and unchecked), "Field 8" (empty and unchecked), "Metadata" (empty), "Tags" (empty, with a note "(Tags are optional, separated)"), "Link to External Site" (filled with "http://"), and "Link to GitHub" (filled with "https://github.com/"). At the bottom is a green "Save Channel" button.

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.

PicoBricks IoT Control Panel

Channel ID: 2403305

Author: mwa0000031402715

Access: Private

Private View

Public View

Channel Settings

Sharing

API Keys

Write API Key

Key

J3KIDYOM822EXM55

MicroBlocks Code of The Project:

comment

IoT CONTROL PANEL via THINGSPEAK (ESP01)

This script is an IoT example using the ESP01 as a WIFI module. All ESP01 related scripts are in the My Blocks menu.

This example makes use of the ThingSpeak web site and service to log sensor data. The user will need a ThingSpeak account to obtain a write API. This API is used in the block to do the updates.

Script will create a station on the user's Internet subnet.

Fill out the thingpeak block with desired sensor data placed in its parameters. Maximum of 8 data values can be logged.

The program will initialize ESP01 and login to the user's local Internet via WIFI credentials provided. All progress is displayed on the IDE.

When READY FOR UPDATES message is displayed, press the micro:bit a button to start the updates. Don't forget that ThingSpeak free account update frequency is once every 15 seconds.

logging (TIF) block controls recording of transactions and responses into the ESP01_Log list. Can be used to learn about the operation of the program and debugging.

ESP01 PINS:

-----> ANTENNA (from TOP)

TX * GND ----- *

EN * * ----- * |

* * ----- * |

3V * * RX: ----- *

NOTE: If ESP01 is connected without using the PicoBricks board socket, it needs to have the EN pin strapped HIGH to work.

when button A pressed

thingspeak upload WriteAPI Channel API T1

PicoBricks-mb temperature (°C) T2 PicoBricks-mb humidity T3

PicoBricks-mb GS Proximity T4 PicoBricks-mb PIR detected T5

PicoBricks-mb light sensor (0-100) % T6

PicoBricks-mb Potentiometer T7 T8

when started

***** SERIAL BAUD RATE *****

ESP01 baud rate defaults to 115200. If you have changed it for some reason, ensure that serial open baud is correctly set below.

comment

You can set the ESP01 serial port speed using the setESP01UART speed block.

PicoBricks-mb init Library

set ESP01_LogSize to 10

set colon to string from unicode 5B

set comma to string from unicode 44

set end to

join string from unicode 13 string from unicode 10

set debug to

set ESP01_Log to list

set rawbuffer to

set TCPreset to join OK _end string from unicode 32

_logging

serial open 115200 baud

wait 10 milliseconds

say Preparing ESP01 setup. Please wait till next prompt.

init_ESP01 ssid SSID password PASSWORD

broadcast go

say READY FOR UPDATES

when go received

if not debug

stop this task

forever

say ESP01_display_LOG

wait 1000 milliseconds

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:

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.

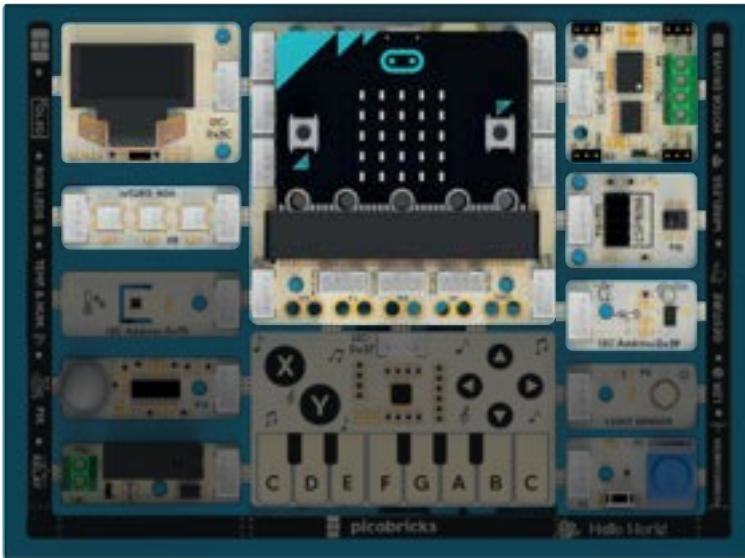




If you want, you can manually stop the water pump by pressing the A button on the Micro:Bit.

Connection Diagram:

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



Water Pump

Project Images:



IoT Vase

Channel ID: 2403879
Author: mac050011402101
Access: Public

Private View Public View Channel Settings Sharing API Keys Data Import / Export

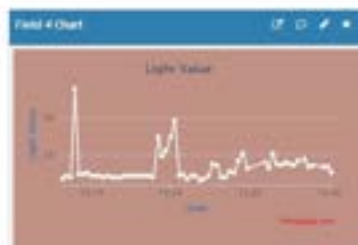
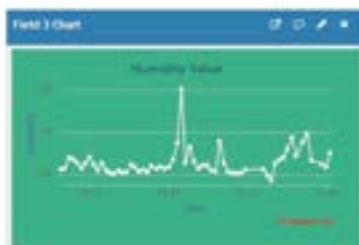
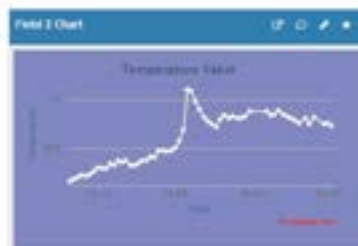
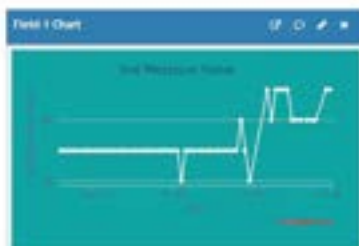
[Add Visualizations](#) [Add Widgets](#) [Export to CSV](#)

[IoT LAB Analysis](#) [IoT LAB Visualizations](#)

Channel 4 of 6

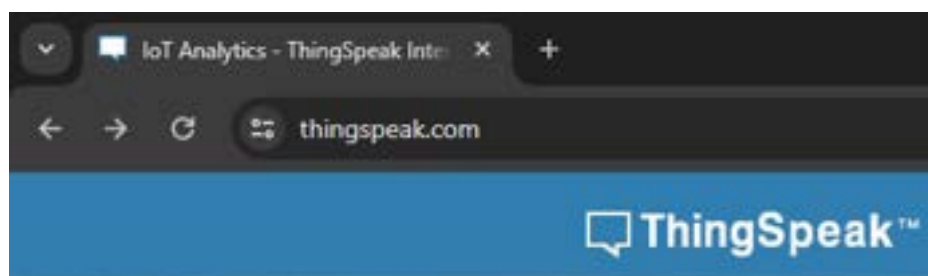
Channel Stats

Created: about 2 hours ago
Last entry: 102 days 11 minutes ago
Entries: 530

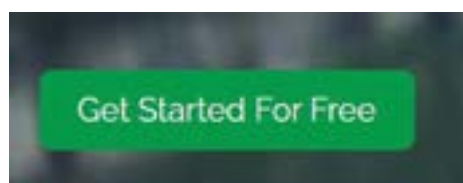


Usage of ThingSpeak:

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.

ThingSpeak™

Channels Apps Support

Commercial Use How to Buy

To use ThingSpeak, you must sign in with your existing MathWorks account or create a new one.

Non-commercial users may use ThingSpeak for free. Free accounts offer limits on certain functionality. Commercial users are eligible for a time-limited free evaluation. To get full access to the MATLAB analysis features on ThingSpeak, log in to ThingSpeak using the email address associated with your university or organization.

To send data faster to ThingSpeak or to send more data from more devices, consider the paid [license options](#) for commercial, academic, home and student usage.

MathWorks

Email

No account? Create one

By signing in, you agree to our [privacy policy](#)

Next

DATA AGGREGATION AND ANALYTICS

ThingSpeak

SMART CONNECTED DEVICES

ALGORITHM DEVELOPMENT
SENSOR ANALYTICS

MATLAB

ALGORITHM DEVELOPMENT
SENSOR ANALYTICS

Create MathWorks Account

Email Address

picobricks@example.com

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

Location

United States

First Name

Last Name

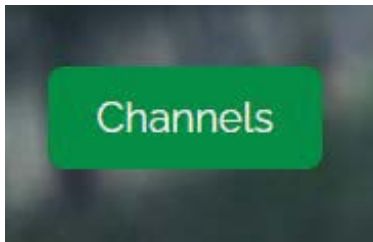
Continue

Cancel

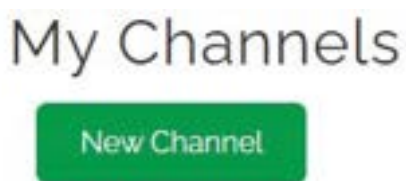
This site is protected by reCAPTCHA and the Google Privacy Policy and Terms of Service apply.

75

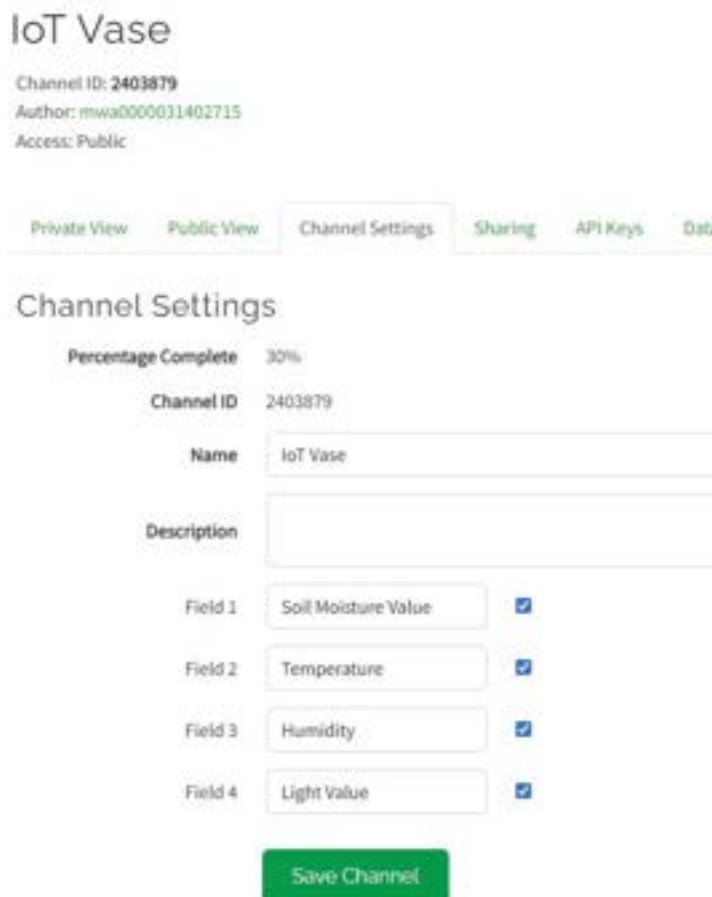
4. After logging into your account, click on the “Channels” button.



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 web form titled "IoT Vase". Below the title, it shows "Channel ID: 2403879", "Author: rmlwa0000031402715", and "Access: Public". There are tabs for "Private View", "Public View", "Channel Settings" (which is active), "Sharing", "API Keys", and "Data". The "Channel Settings" section includes a "Percentage Complete" indicator at 30%. Below this are input fields for "Channel ID" (2403879), "Name" (IoT Vase), and "Description". There are four "Field" entries, each with a text input and a checked checkbox: "Field 1" (Soil Moisture Value), "Field 2" (Temperature), "Field 3" (Humidity), and "Field 4" (Light Value). At the bottom is a green "Save Channel" button.

IoT Vase

Channel ID: 2403879
Author: rmlwa0000031402715
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.

Private View
Public View
Channel Settings
Sharing
API Keys
Data Imp

Write API Key

Key
L2LBDAND6J8NT1B8

Generate New Write API Key

MicroBlocks Code of The Project:

Comment

KIT WARE - vs THINGSPEAK (ESP811):

This script is an IoT example using the ESP811 as a WIFI module. All ESP811 related scripts are in the My Blocks menu.

This example makes use of the ThingSpeak web site and service to log sensor data. The user will need a ThingSpeak account to obtain a write API. This API is used in the block to do the updates.

Script will create a station on the user's Internet subnet.

Fill out the thingspeak block with desired sensor data placed in its parameters. Maximum of 8 data values can be logged.

The program will initialize ESP811 and login to the user's local internet via WIFI credentials provided. All progress is displayed on the IDE.

When READY FOR UPDATES message is displayed, press the micro bit a button to start the updates. (Don't forget that ThingSpeak free account update frequency is once every 15 seconds).

Logging [T/F] block controls recording of transactions and responses into the ESP811_log.txt. Can be used to learn about the operation of the program and debugging.

ESP811 PINs:

```

-----> ANTENNA (from TOP)
TX + - GND -----+
EN + -          -----+
VCC + -          -----+
GND + -          -----+
3V3 + - RX -----+

```

NOTE: If ESP811 is connected without using the PicoBricks board socket, it needs to have the EN pin strapped HIGH to work.

when started

initialize (0) OLED, 0 status, address (hex) 00, reset pin (0) flip

Serial BAUD RATE

ESP811 baud rate defaults to 115200. If you have changed it for some reason, ensure that serial open baud is correctly set below.

comment

You can set the ESP811 serial port speed using the setESP811UART speed block.

PicoBricks-mb init Library

set ESP811 baud to 10

set status to string from unicode 08

set address to string from unicode 44

set pin to

join string from unicode 15, string from unicode 08

set delay to 10

set ESP811 log to 10

set baudrate to 10

set 1 channel to join 04, and 0, string from unicode 00

logging

serial open (115200) baud

wait 10 milliseconds

say Preparing ESP811 setup. Please wait til next prompt

set ESP811 baud 000, password PASSWORD

broadcast get

say READY FOR UPDATES

when pin received

set status

stop time task

summarize

say ESP811_display_100

wait 1000 milliseconds

set setAnalogValue to

rescale read analog pin 0 from 0 0 to 100 0

thingspeak upload WriteAPI 00, 00, 00, 00, 00, 00, 00, 00

PicoBricks-mb temperature (°C) 00, PicoBricks-mb humidity 00

PicoBricks-mb light sensor (0-100) % 00, 00, 00, 00

write join humidity, PicoBricks-mb temperature (°C) at x 0 y 0

inverse

write join HUM, PicoBricks-mb humidity at x 0 y 0 inverse

write join 00, 00, 00, 00, 00, 00, 00, 00 at x 0 y 0 inverse

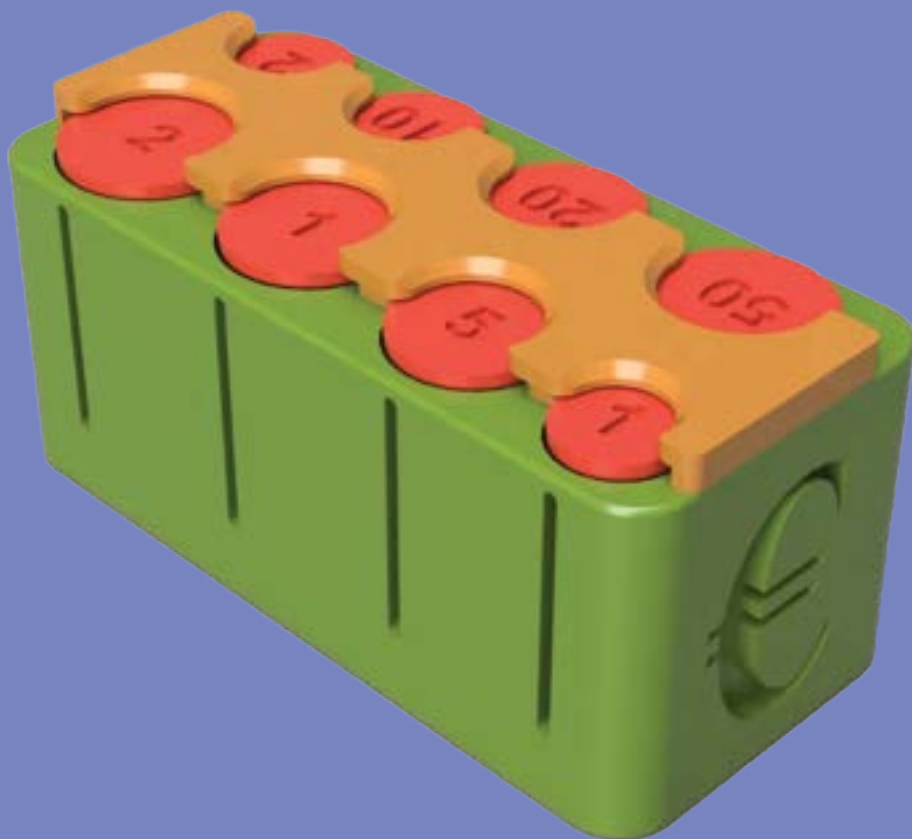
set 00, 00

PicoBricks-mb set motor 00 speed 00 (0-255) dir 00

wait 1000 milliseconds

PicoBricks-mb set motor 00 speed 00 (0-255) dir 00

Coin Dispenser



Coin Dispenser Project

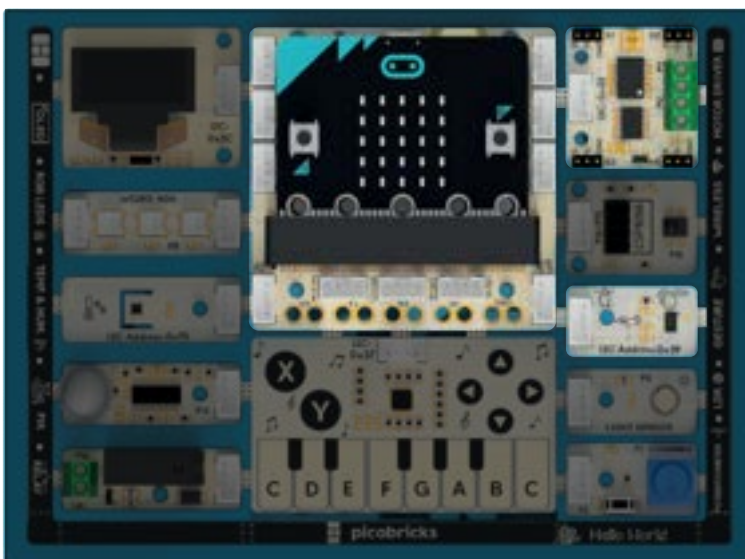
Some people dislike carrying coin in their pockets. This might be due to the extra weight it adds or the noise it makes while walking. For others, collecting coins could be a hobby. However, when collecting coin, we may struggle to separate them. The easiest way to separate coin is by their dimensions. 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:

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.

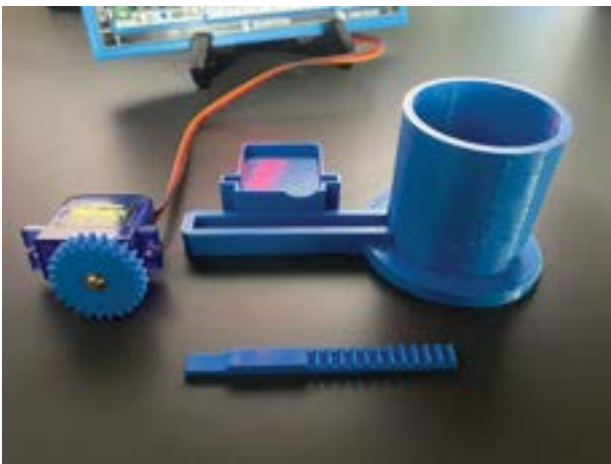
Connection Diagram:

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



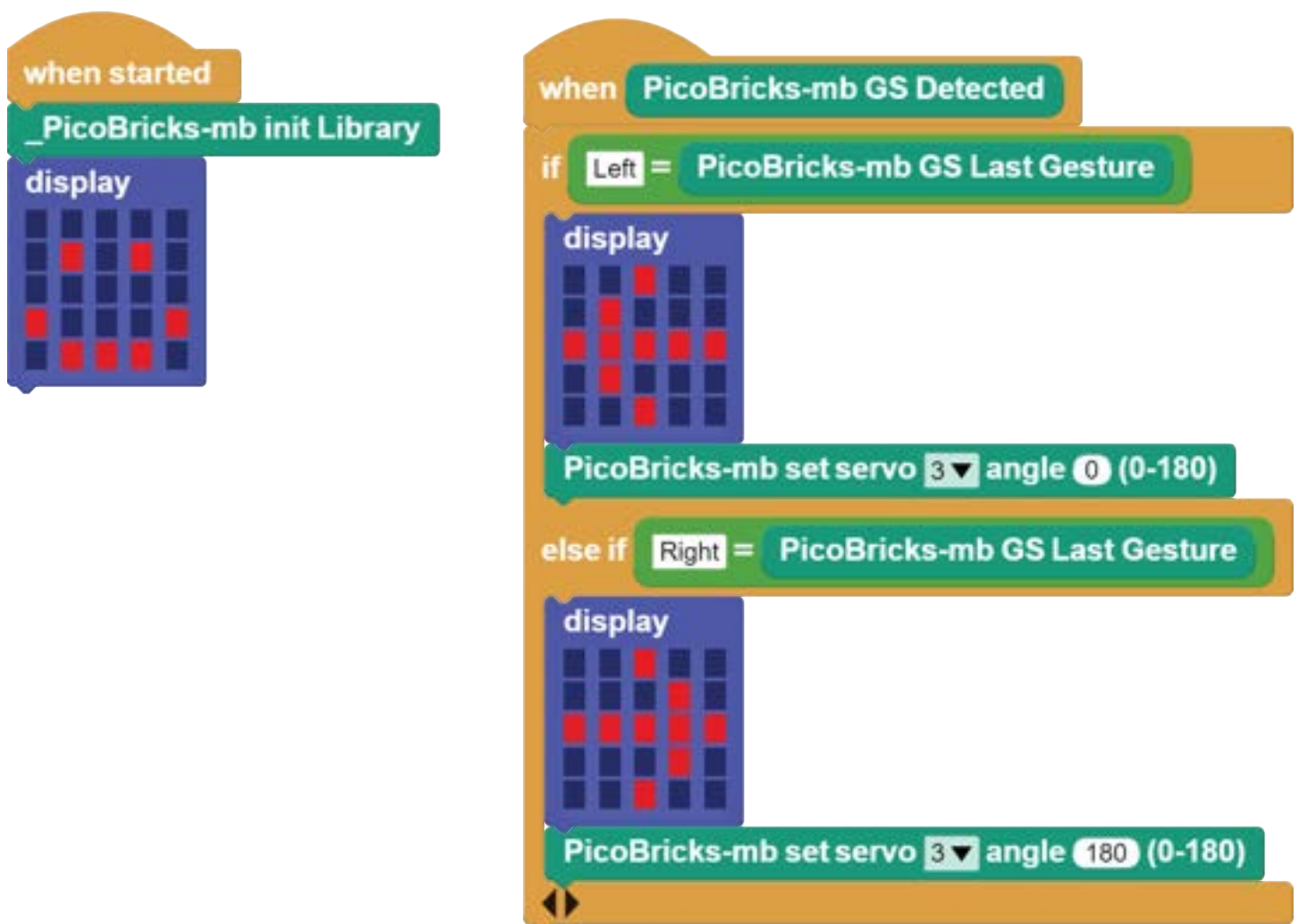
Servo Motor

Project Images:

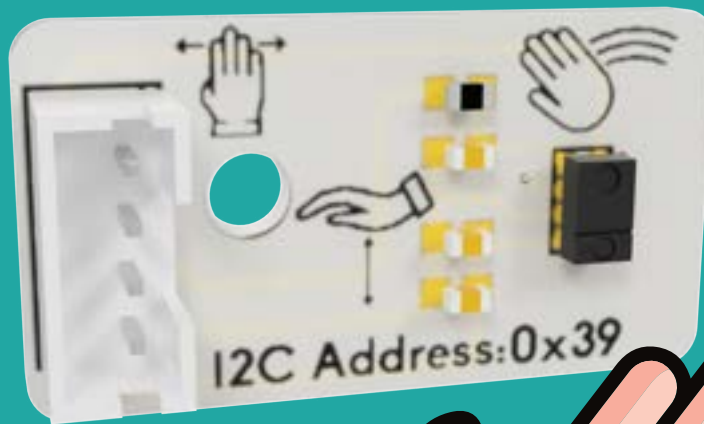


MicroBlocks Code of The Project:

```
COIN DISPENSER:  
  
micro:bit v1 and v2  
  
comment A coin dispenser is driven by a servo motor.  
The Gesture sensor is used to detect hand motions.  
Right and Left movements of the hand will make the  
dispenser dispense coins in the hopper.
```



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.

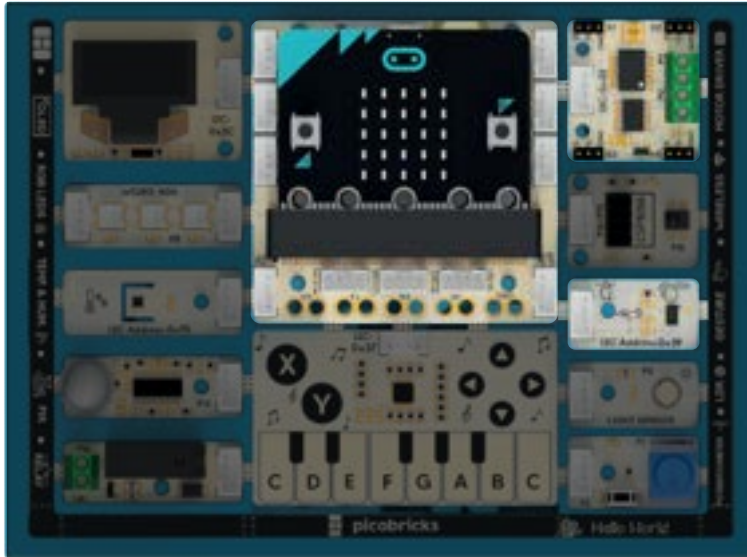
● Project Details:

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.

Note: 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.

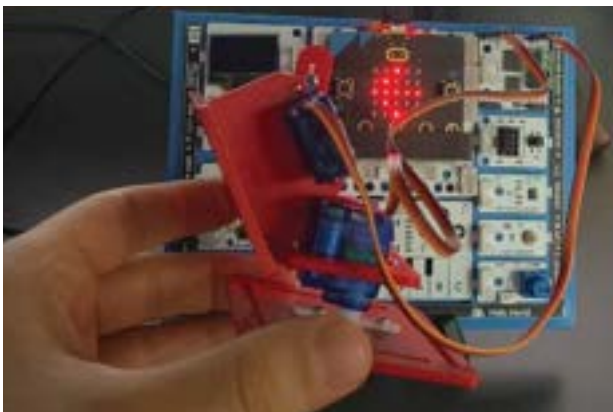
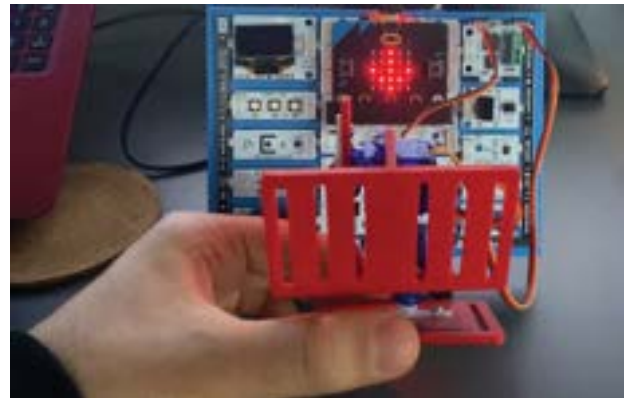
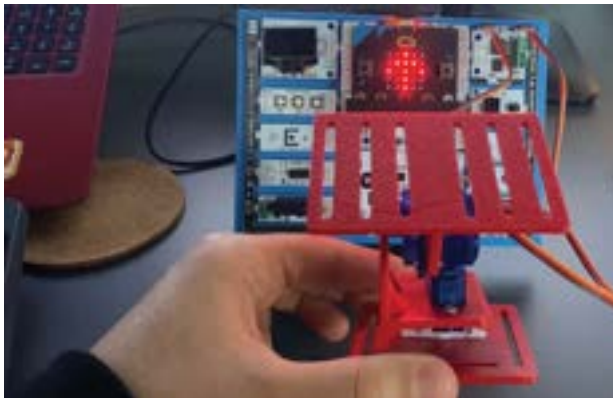
● Connection Diagram:

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

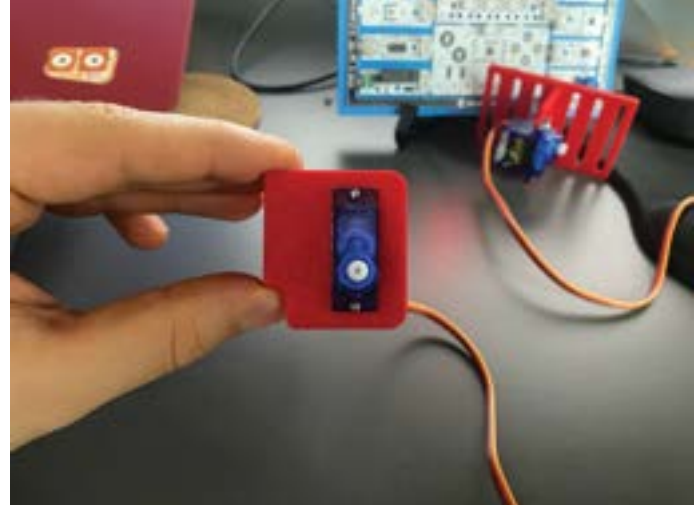
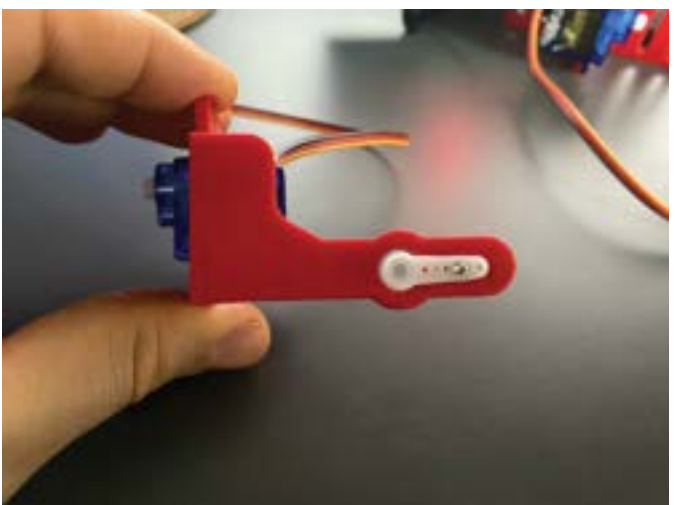
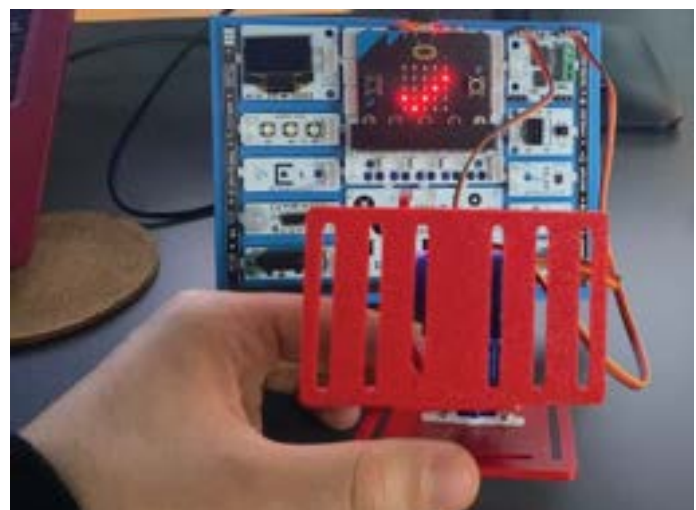
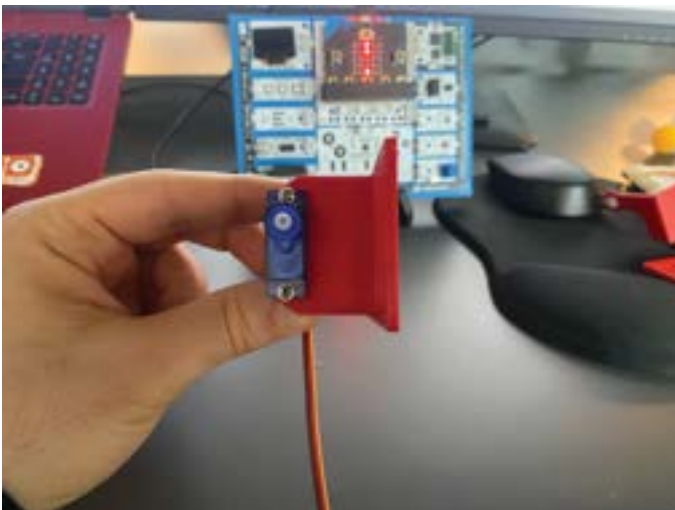
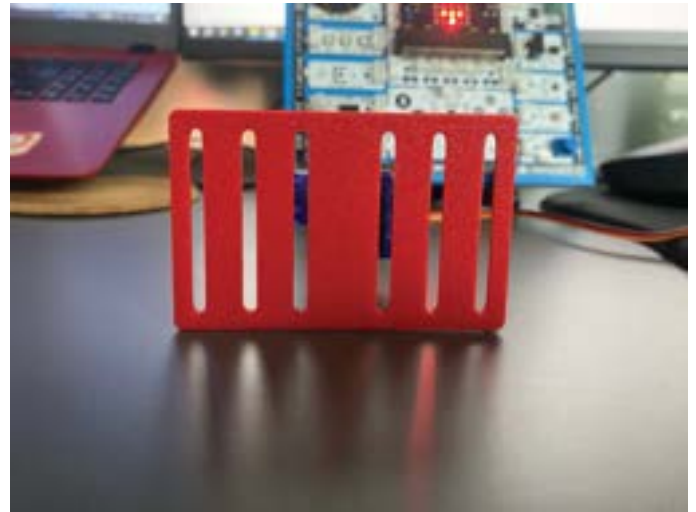


Servo Motor

● Project Images:



● Installation Images:



MicroBlocks Code of The Project:

GESTURE CONTROLLED ROBOT ARM:

micro:bit v1 and v2

Two servo motors are used.
At startup, they are positioned at 0 degrees.

comment mb-A button will reset servos to 0 degrees.

Gesture sensor controls the platforms movements in four directions: up, down, left, and right.

Servo 1 moves Up and Down
Servo 2 moves Left and Right

when started

_PicoBricks-mb init Library

PicoBricks-mb set servo ALL angle 0 (0-180)

display

when button A pressed

_PicoBricks-mb init Library

PicoBricks-mb set servo ALL angle 0 (0-180)

display

when PicoBricks-mb GS Detected

if Up = PicoBricks-mb GS Last Gesture

display

PicoBricks-mb set servo 3 angle 0 (0-180)

play note c octave 0 for 100 ms

else if Down = PicoBricks-mb GS Last Gesture

display

PicoBricks-mb set servo 3 angle 180 (0-180)

play note c octave 0 for 100 ms

else if Right = PicoBricks-mb GS Last Gesture

display

PicoBricks-mb set servo 4 angle 0 (0-180)

play note d octave 0 for 100 ms

else if Left = PicoBricks-mb GS Last Gesture

display

PicoBricks-mb set servo 4 angle 180 (0-180)

play note d octave 0 for 100 ms

3D Labyrinth



3D Labyrinth Project

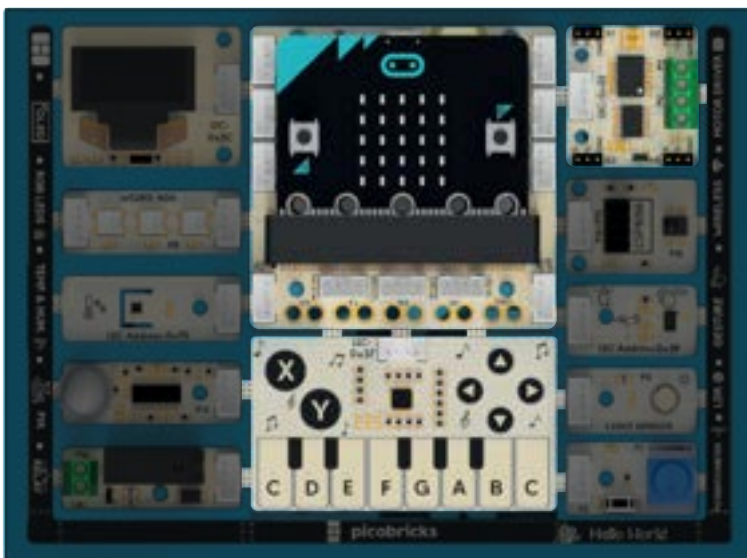
There are multiple ways to exit a maze, but the most well-known method is to follow the wall with your left/right hand. Although this method may take some time, you can definitely get out of the maze by consistently touching the wall. Maze tests enhance problem-solving skills. Someone who frequently 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:

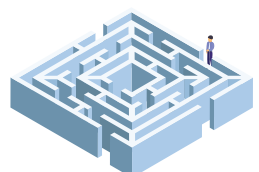
Let's create a maze project that can move in right, left, up, and down directions by assembling 3D printer parts as shown in the visuals. To ensure the movement of the maze in this project, we will utilize two servo motors connected to the PicoBricks motor driver. The direction keys on the PicoBricks 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:

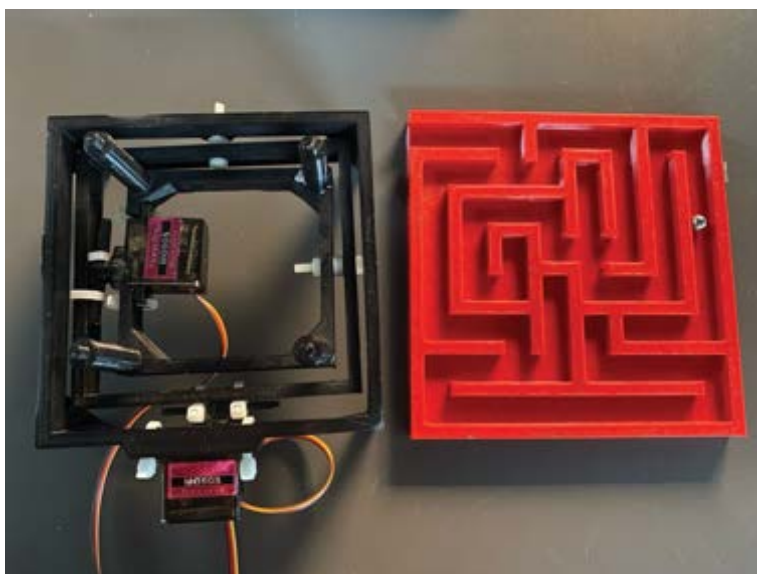
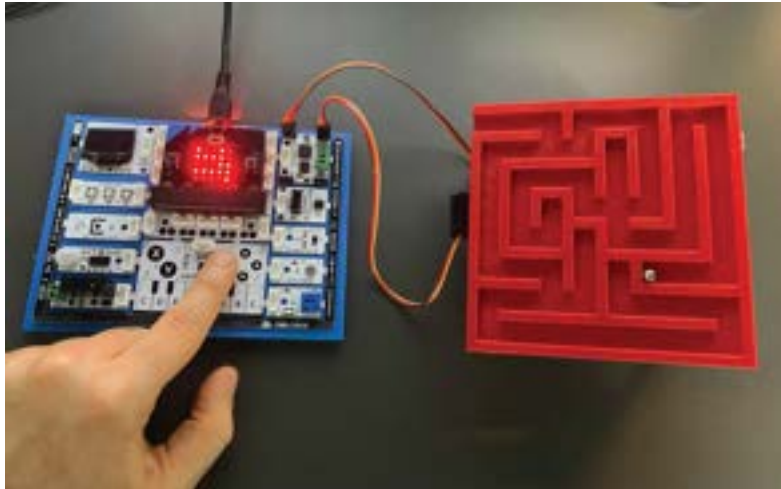
You can prepare this project by breaking down PicoBricks modules at proper points.



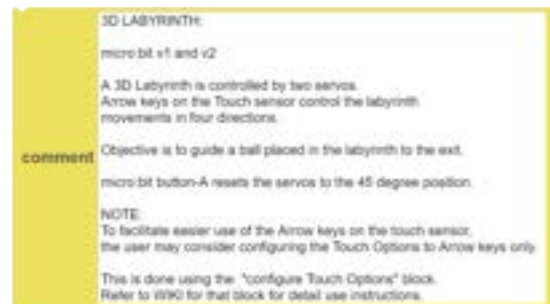
Servo Motor



Project Images:



MicroBlocks Code of The Project:



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.





Radar

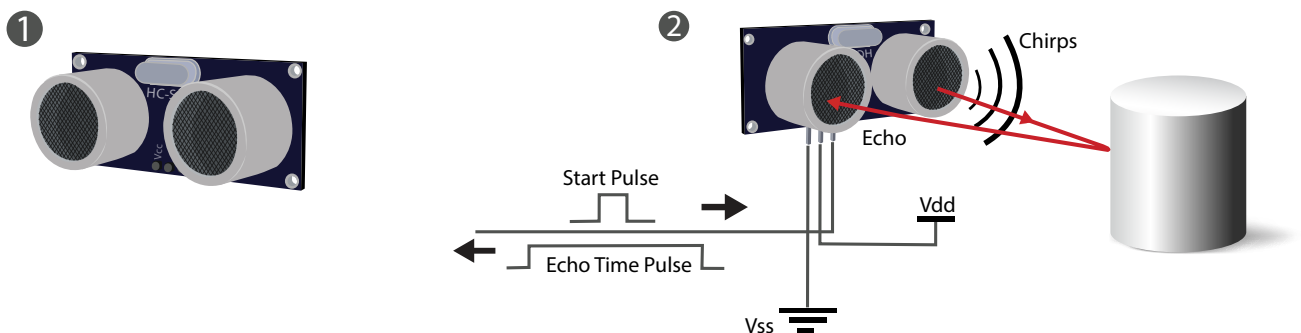


Radar Project

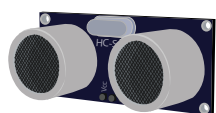
The radar is a device that detects objects in its surroundings, their direction of movement, their speeds, and other values through radio waves. The effective, or ranges, of radars can vary. Depending on this variation, their applications change. Radars are frequently used to ensure security in various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

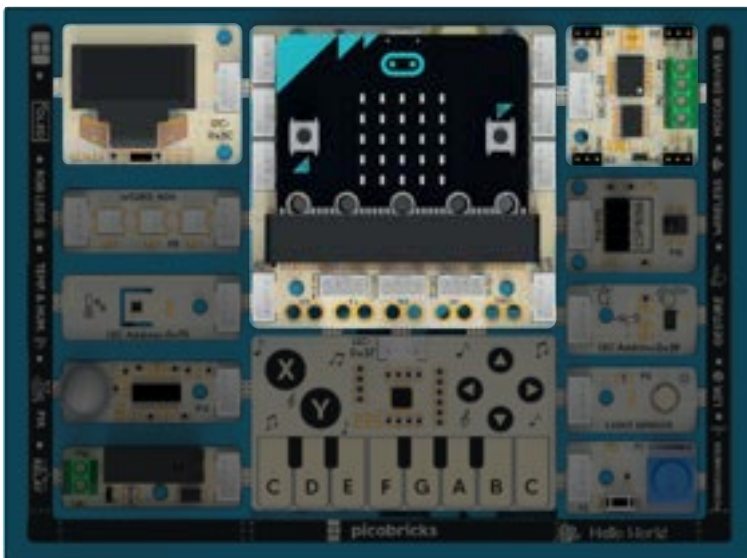


Project Details:

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.

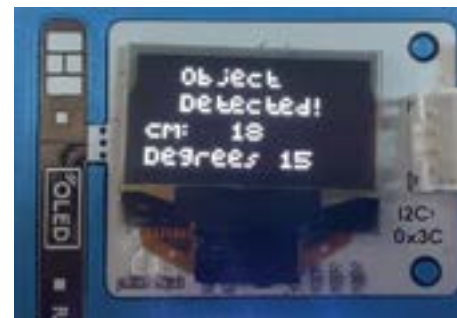


Servo Motor



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.



MicroBlocks Code of The Project:

```

define dispScan
  _defer monochrome display updates
  clear display
  write "Scanner" on TFT at x 0 y 0 color 0 scale 1 wrap 0 bg
  color 0
  write join angle scanAngle on TFT at x 0 y 10 color 0 scale 1
  wrap 0
  write "Object Located at:" on TFT at x 0 y 20 color 0 scale 1 wrap 0
  write join objDistance dist on TFT at x 10 y 40 color 0 scale 1
  wrap 0
  _resume monochrome display updates
  
```

```

when button1 pressed
  repeat until objectDetected
    initialize local dist to distance (cm) trigger 1 echo 1
    if dist <= radarRange
      set objDistance to dist
      set objectDetected to 1
      wait 100 milliseconds
  
```

comment
RADAR:
monobit v2
RADAR project uses a servo-driven distance sensor to detect objects within a set detection range.
User sets the distance using the Pot dial and presses the pot button to store it.
Then the scanning process displays a graphical view of the distance scanner until the object is detected. Then the angle and distance of the object is displayed on the OLED.

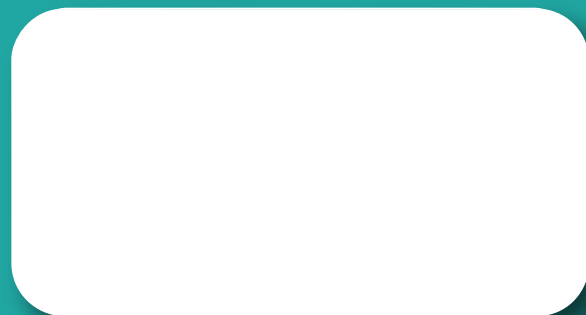
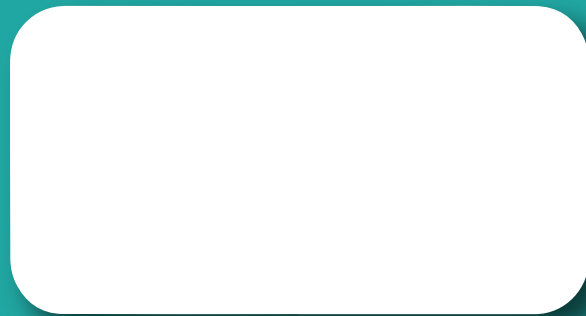
```

when started
  set objectDetected to 0
  set objDistance to 0
  set radarRange to 0
  repeat until PicoBricks-mb Pot Button
    set radarRange to
    rescale PicoBricks-mb PotDialometer from (0, 4095) to (0, 600)
  _defer monochrome display updates
  clear display
  write radarRange on TFT at x 0 y 0 color 0 scale 1 wrap 0
  write radarRange on TFT at x 0 y 10 color 0 scale 1 wrap 0
  _resume monochrome display updates
  wait 200 milliseconds
  broadcast start
  
```

```

when start triggered
  clear display
  write "Scanner" on TFT at x 0 y 0 color 0 scale 1 wrap 0 bg
  color 0
  forever
    for angle in range 0 to 180 by 1
      set scanAngle to angle
      PicoBricks-mb set servo 17 angle angle (0-180)
      draw line on TFT from x 0 y 40 to x
      rescale angle from (0, 600) to (0, 420) y 40 color 0
      if objectDetected
        dispScan
        play note 5 octaves 0 for 200 ms
        return 0
    for angle in range 180 to 0 by 1
      set scanAngle to angle
      PicoBricks-mb set servo 17 angle angle (0-180)
      draw line on TFT from x 0 y 40 to x
      rescale angle from (0, 600) to (0, 420) y 40 color 0
      if objectDetected
        dispScan
        play note 5 octaves 0 for 200 ms
        return 0
  
```

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.

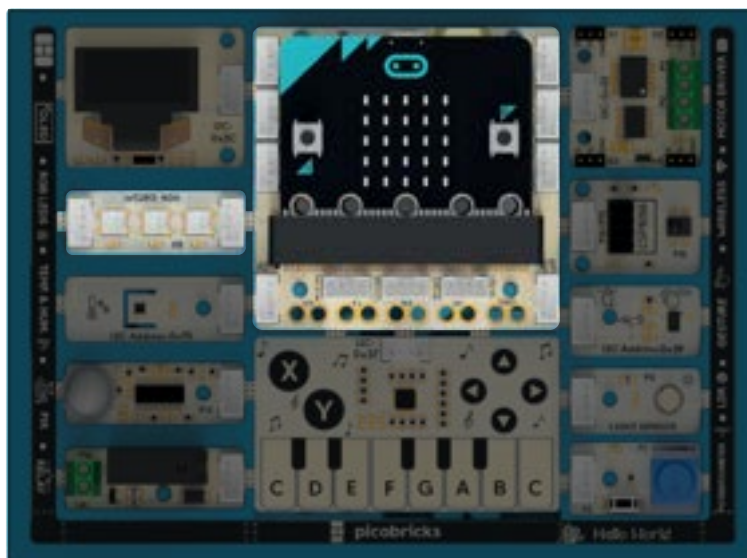
Color animations are used in various areas such as advertising panels, celebration areas, etc., to attract attention. In these systems, which are created by illuminating a LED with different colors at specific time intervals, RGB LEDs are commonly used. The main reason for the use of RGB LEDs in these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

Project Details:

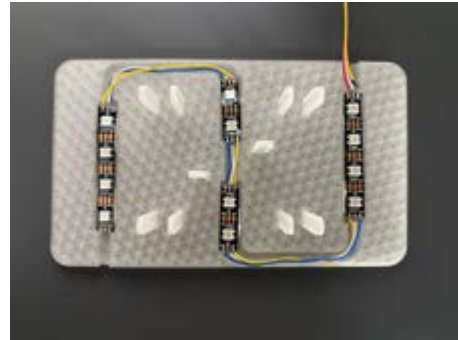
In this project, we will create color animations by placing the addressable RGB LEDs connected to the PicoBricks RGB LED module inside the 3D-printed PicoBricks Logo lamp.

Connection Diagram:

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



Project Images:

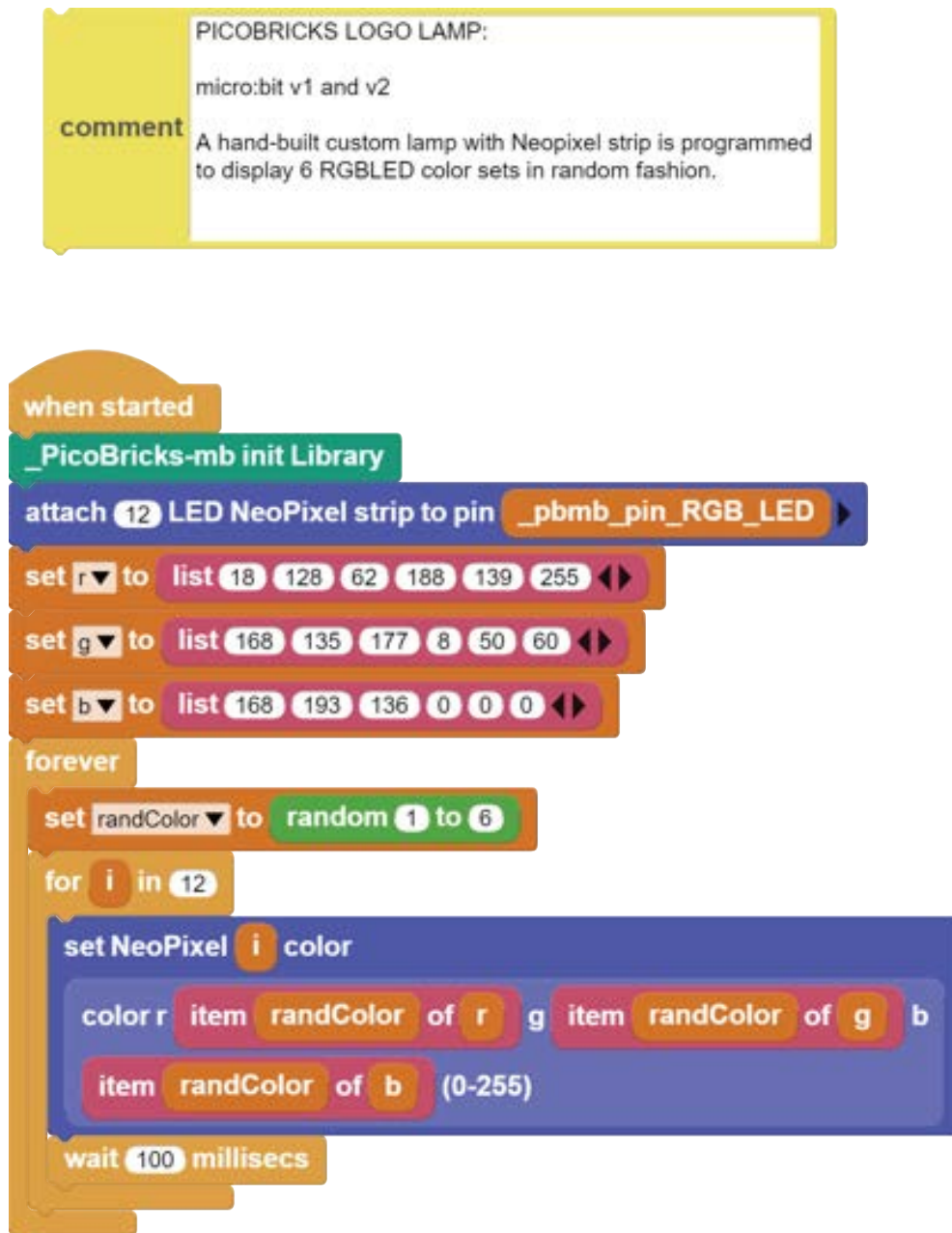


The STL Files of The Project:

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



MicroBlocks Code of The Project:

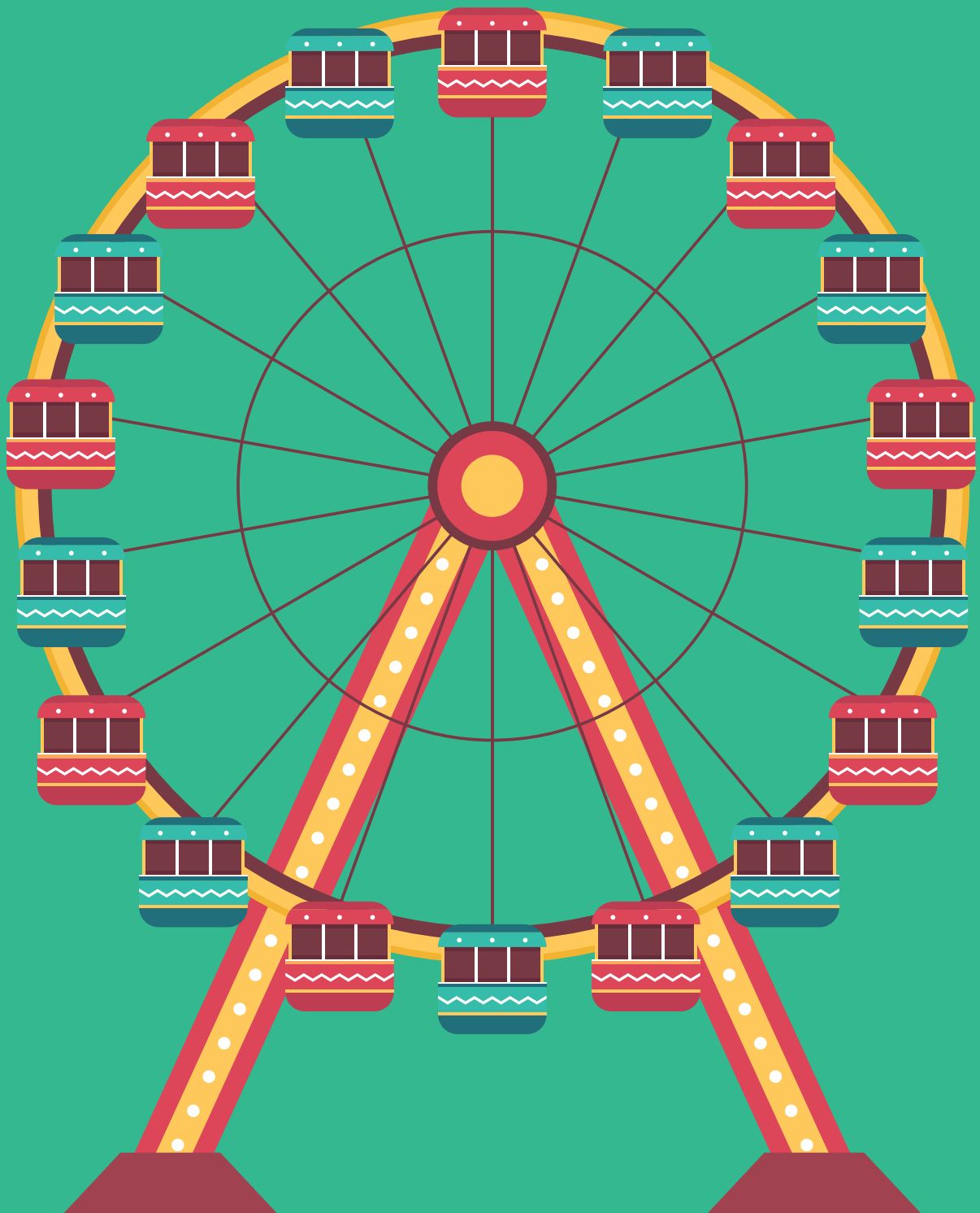


The image shows the MicroBlocks code for a project titled "PICOBRICKS LOGO LAMP". The code is organized into a yellow comment block and a main script area. The comment block contains the title and a description of the project. The main script area starts with a "when started" block, followed by an initialization block "_PicoBricks-mb init Library". The main logic is enclosed in a "forever" loop. Inside the loop, a "randColor" variable is set to a random value between 1 and 6. A "for" loop then iterates from 1 to 12, setting the color of each NeoPixel. The color is determined by the "randColor" variable and the "randColor" of the "r", "g", and "b" lists. The "r" list contains the values [18, 128, 62, 188, 139, 255], the "g" list contains [168, 135, 177, 8, 50, 60], and the "b" list contains [168, 193, 136, 0, 0, 0]. The "randColor" of each list is used to select an item from the list, which is then used to set the color of the NeoPixel. The "b" list is also used to set the color of the NeoPixel, with a note "(0-255)" indicating the range of values. The "for" loop is followed by a "wait 100 millisecs" block.

```
PICOBRICKS LOGO LAMP:
micro:bit v1 and v2
comment A hand-built custom lamp with Neopixel strip is programmed
to display 6 RGBLED color sets in random fashion.

when started
  _PicoBricks-mb init Library
  attach 12 LED NeoPixel strip to pin _pbmb_pin_RGB_LED
  set r to list 18 128 62 188 139 255
  set g to list 168 135 177 8 50 60
  set b to list 168 193 136 0 0 0
  forever
    set randColor to random 1 to 6
    for i in 12
      set NeoPixel i color
      color r item randColor of r g item randColor of g b
      item randColor of b (0-255)
    wait 100 millisecs
```

Ferris Wheel



Ferris Wheel Project

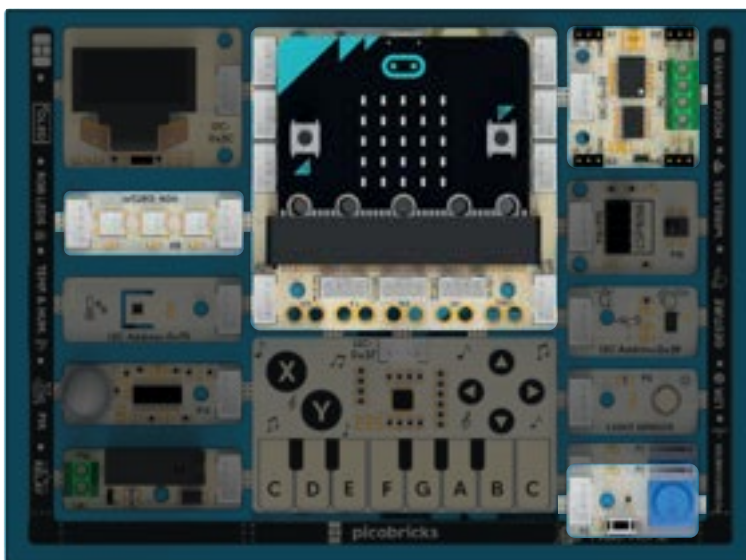
A Ferris wheel is an amusement ride where seats are positioned around a circle that rotates on a central axis. PicoBricks Ferris Wheel is a project kit where you can adjust the speed of the Ferris wheel based on the value of the potentiometer by using the potentiometer, motor driver, and mainboard module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

Project Details:

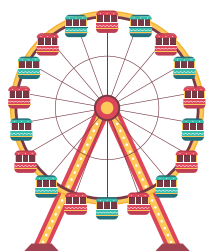
In this project, we will control the rotation speed of the Ferris Wheel based on the speed of the DC motor by using the PicoBricks Potentiometer module.

Connection Diagram:

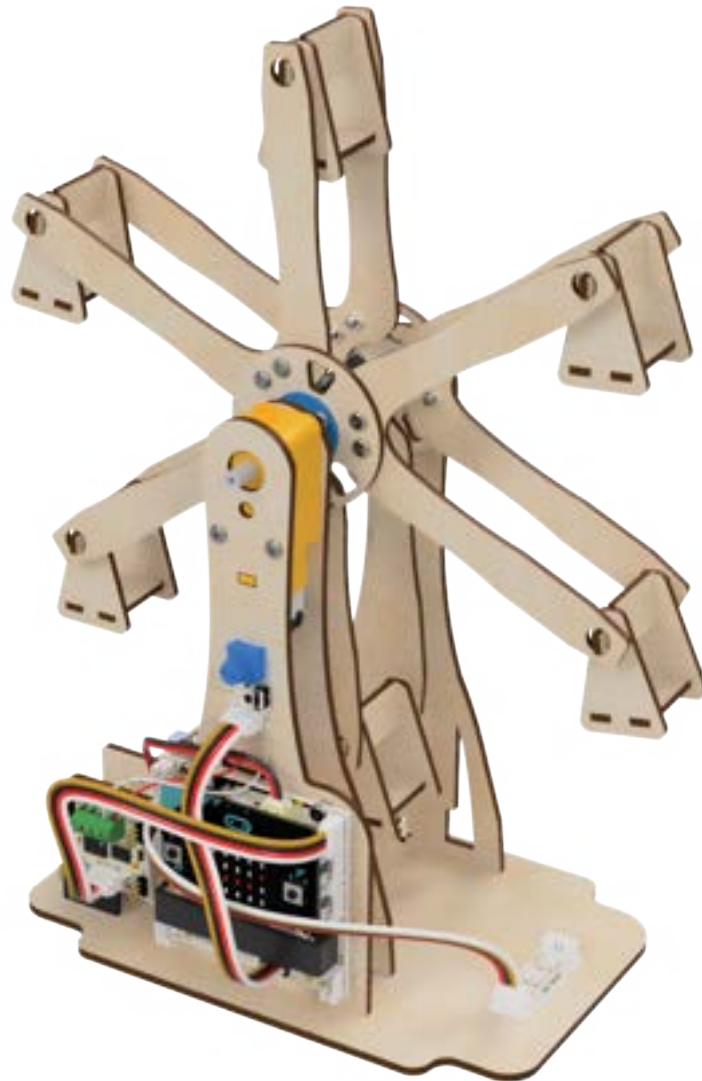
You can prepare this project by breaking down PicoBricks modules at proper points.



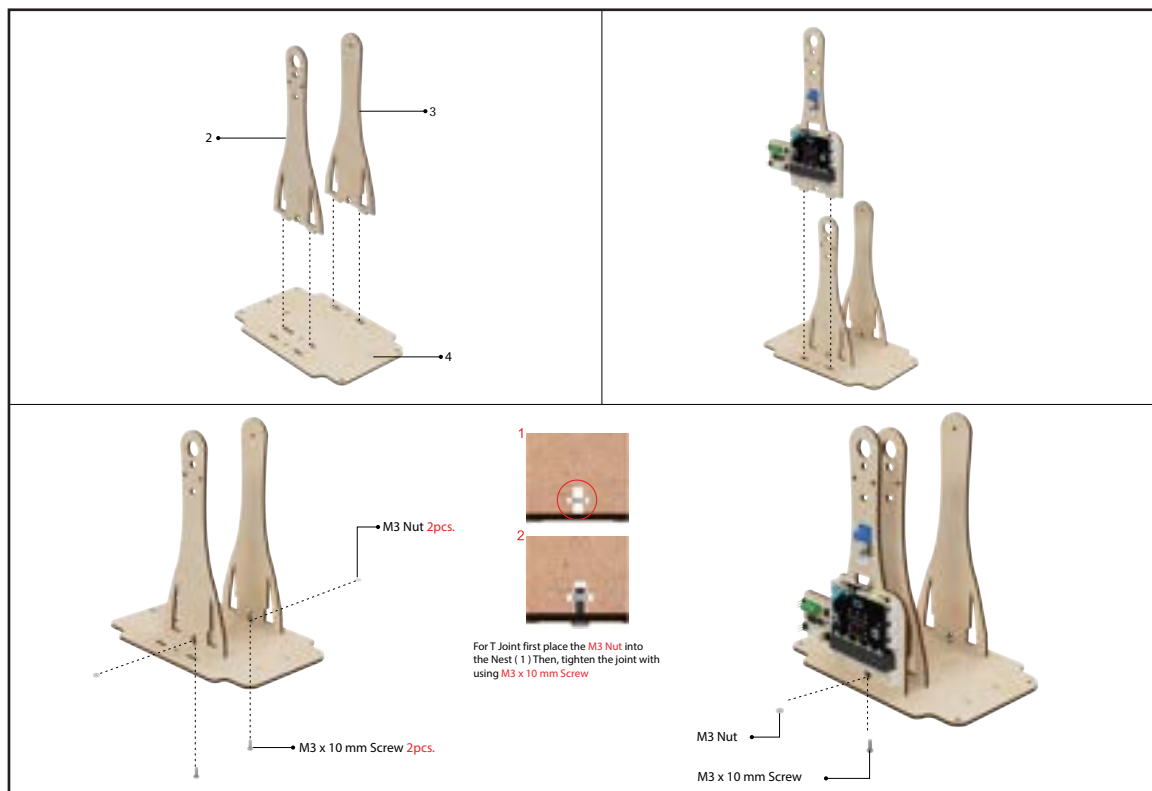
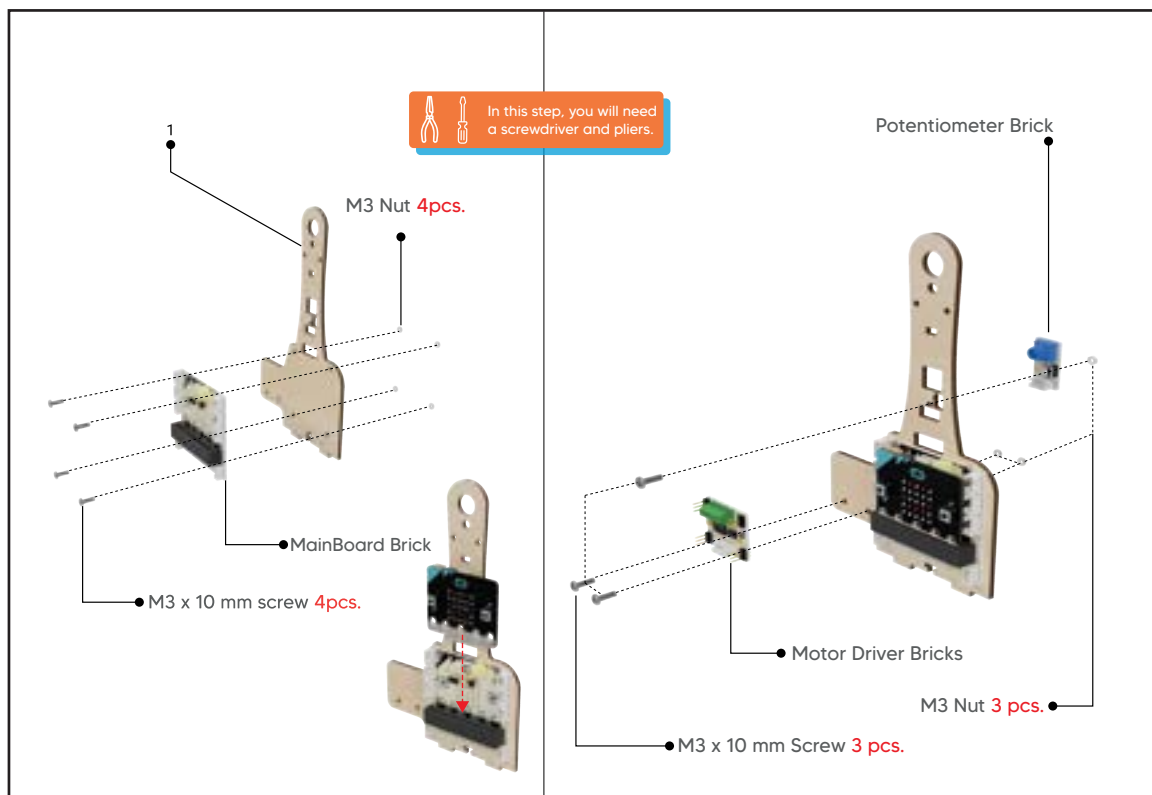
DC Motor

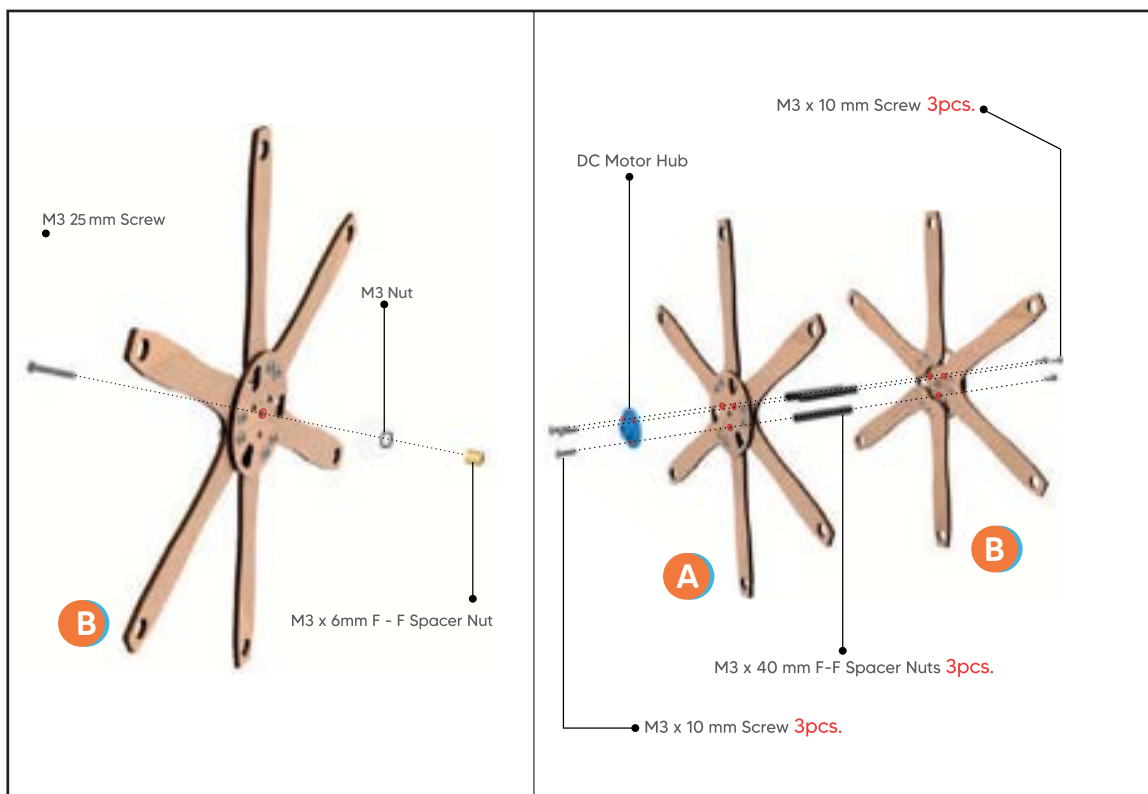
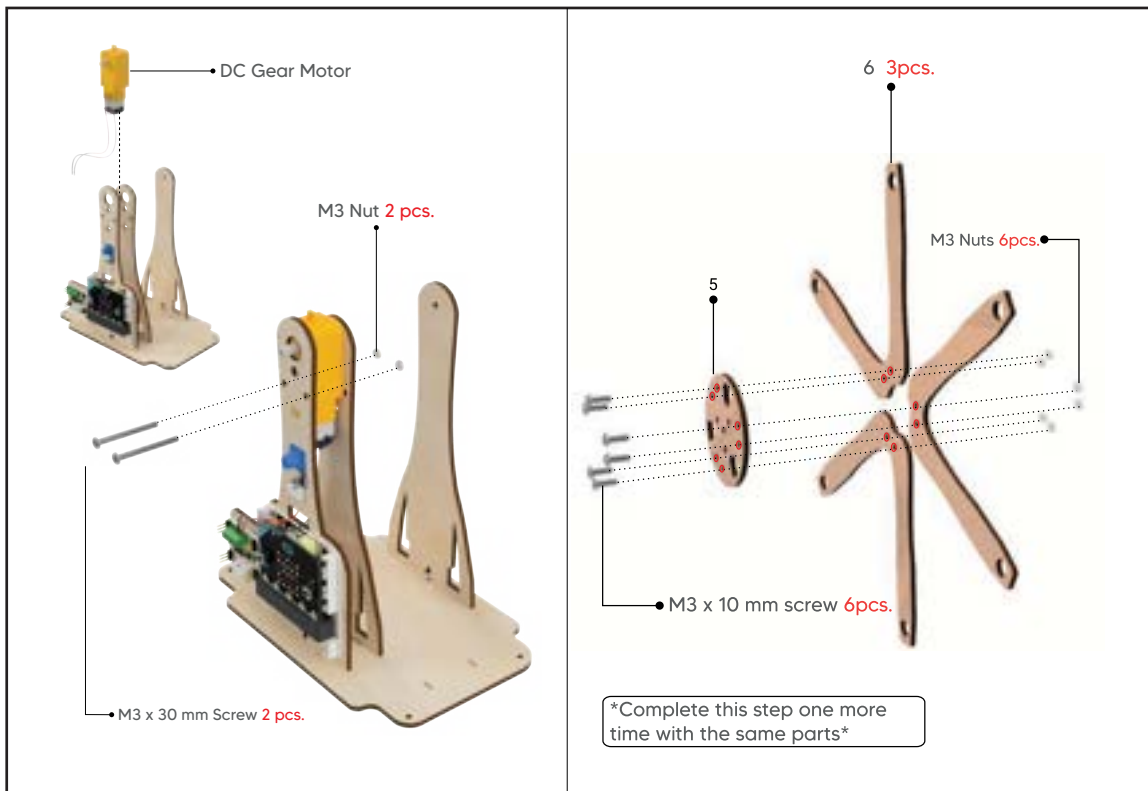


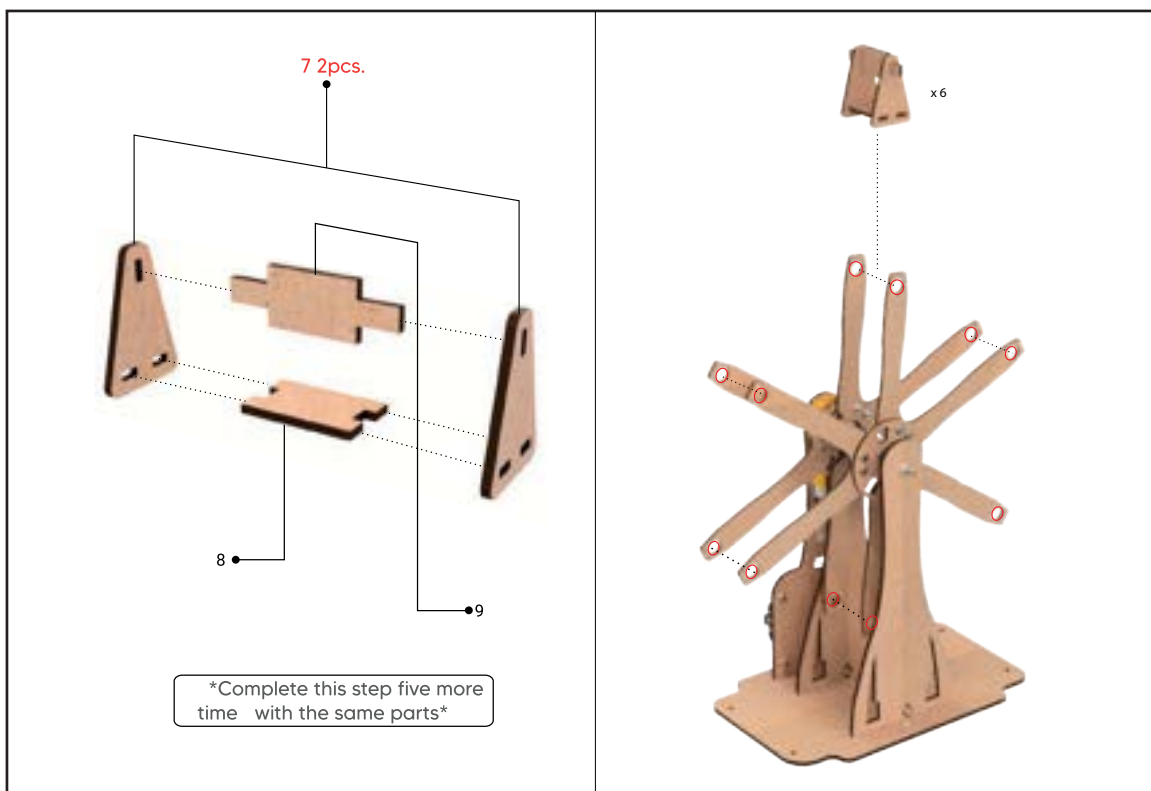
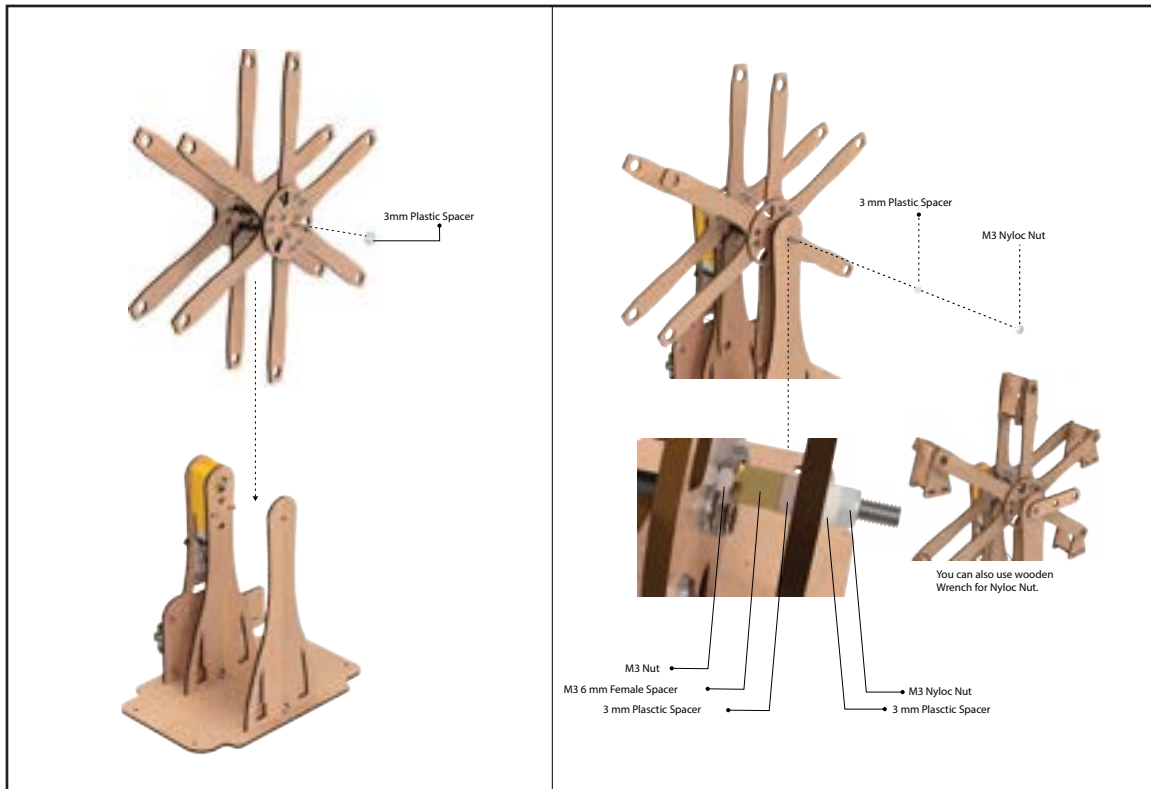
Project Images:

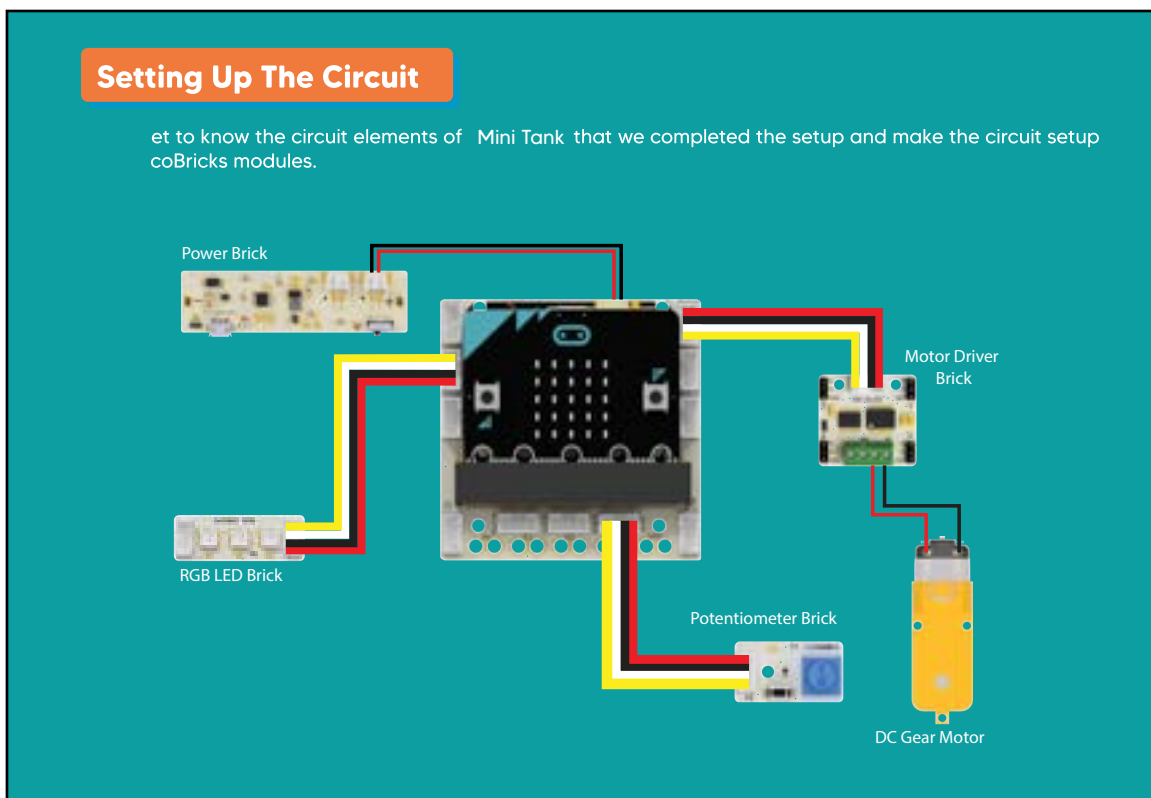
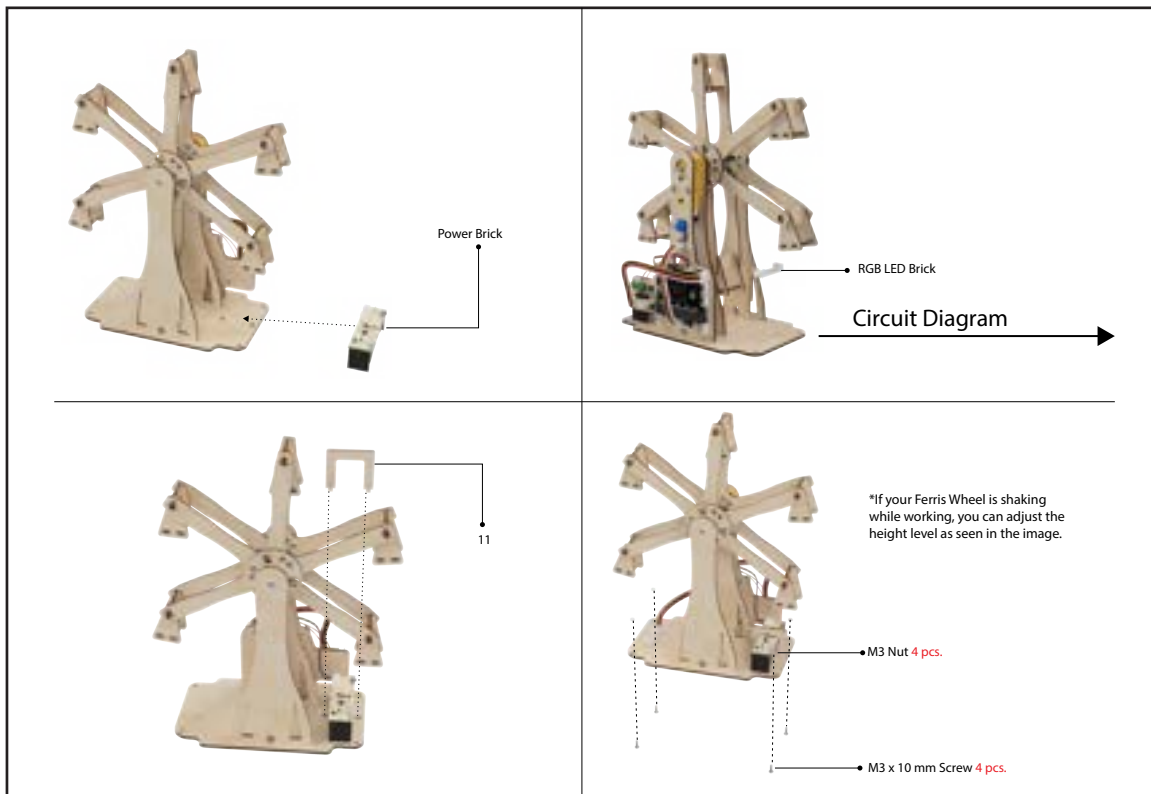


● Setup Steps of The Project:







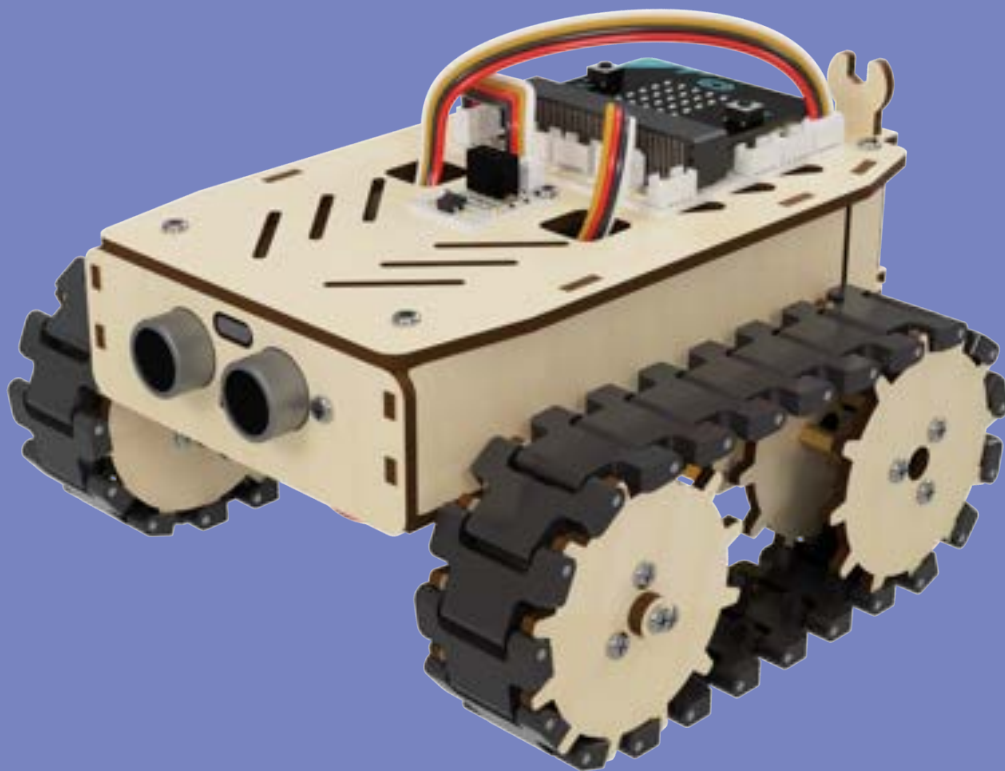


MicroBlocks Code of The Project:

```
PICOBRICKS LOGO LAMP:  
  
micro:bit v1 and v2  
  
comment A hand-built custom lamp with Neopixel strip is programmed  
to display 6 RGBLED color sets in random fashion.
```

```
when started  
  _PicoBricks-mb init Library  
  attach 12 LED NeoPixel strip to pin _pbmb_pin_RGB_LED  
  set r to list 18 128 62 188 139 255  
  set g to list 168 135 177 8 50 60  
  set b to list 168 193 136 0 0 0  
  forever  
    set randColor to random 1 to 6  
    for i in 12  
      set NeoPixel i color  
      color r item randColor of r g item randColor of g b  
      item randColor of b (0-255)  
    wait 100 millisecs
```

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.

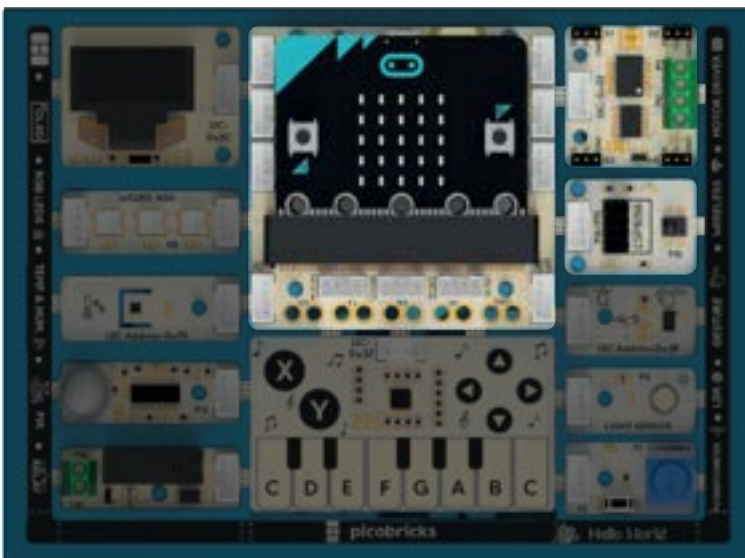
The PicoBricks Mars Explorer Car is a wooden project kit that utilizes two DC motors and a tracked platform. This robot car, controllable remotely with a remote controller thanks to the IR receiver, can decide on its movements by detecting surrounding objects through the front of its distance sensor.

Project Details:

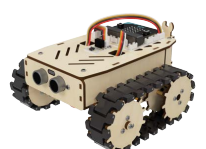
In this project, we will control two DC motors that connected to motor driver by using IR receiver on the PicoBricks wireless module with the remote controller. The robot car moves in the desired direction thanks to the DC motors. Additionally, if the HC-SR04 distance sensor on the robot car kit detects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

Connection Diagram:

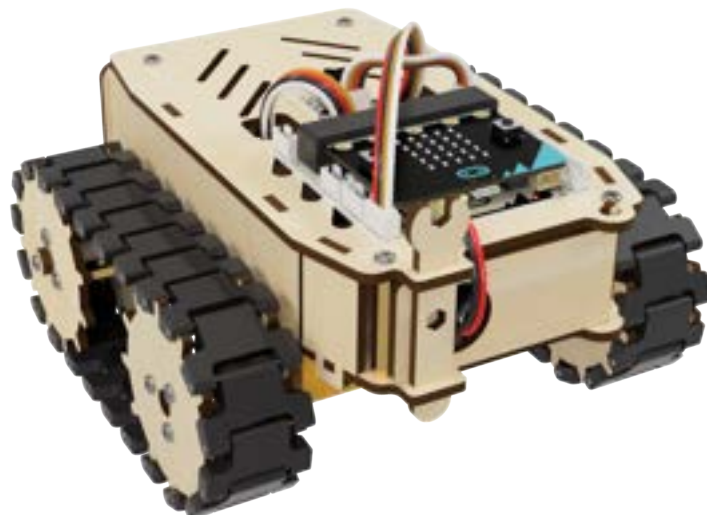
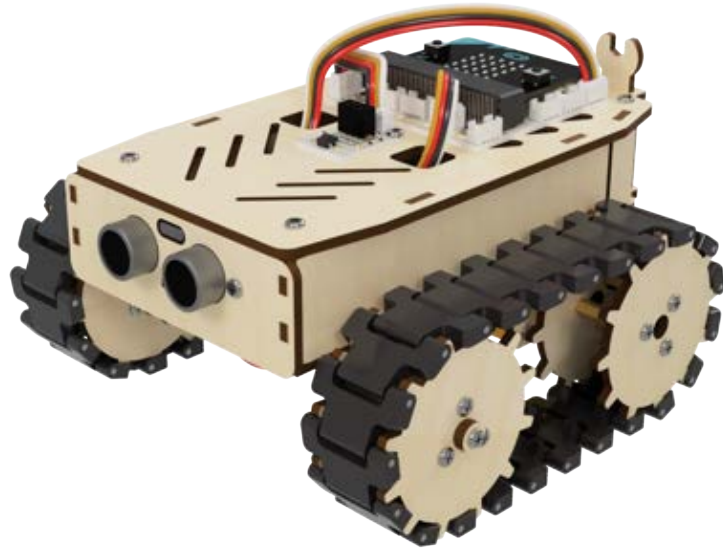
You can prepare this project by breaking PicoBricks modules at proper points.



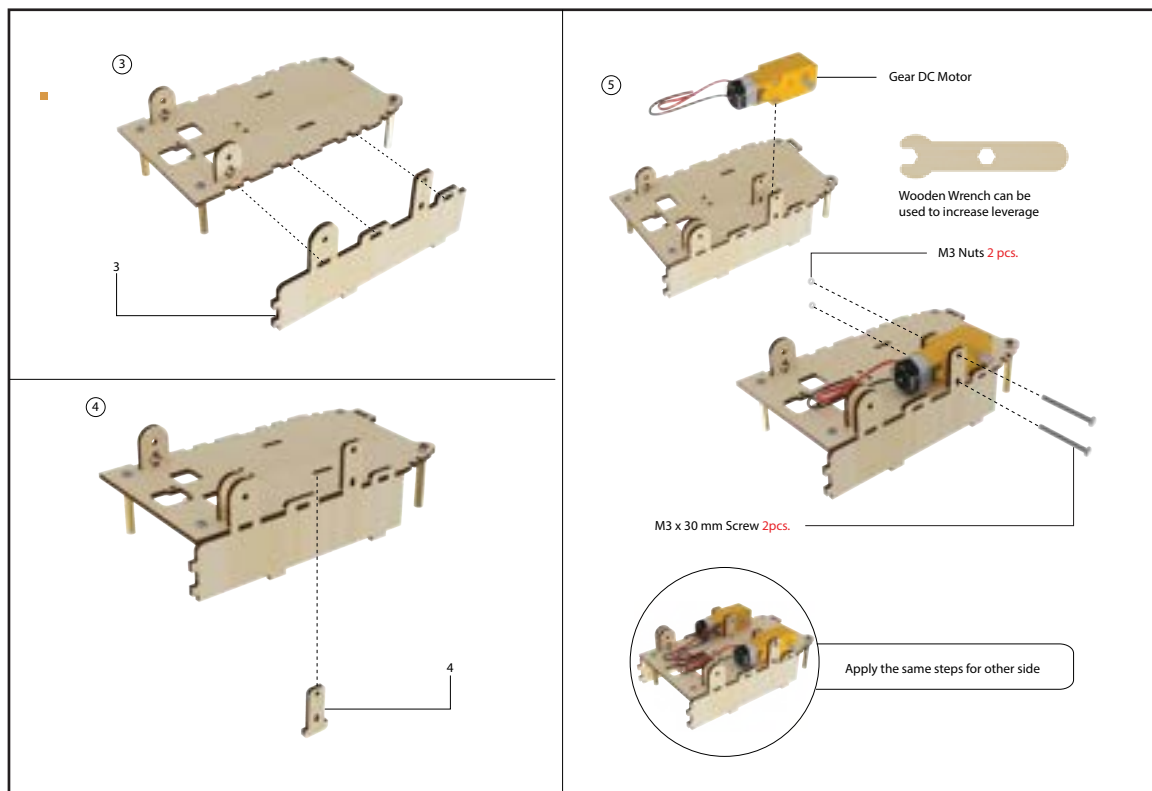
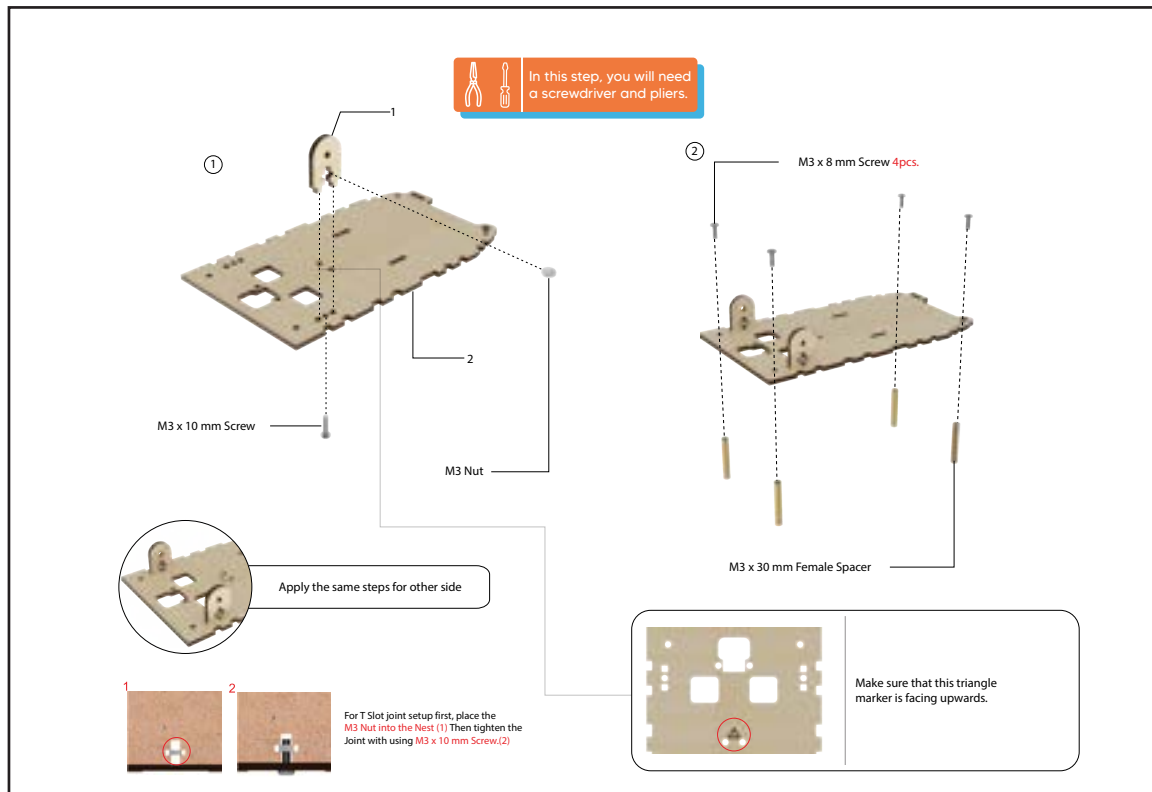
x2 DC Motor

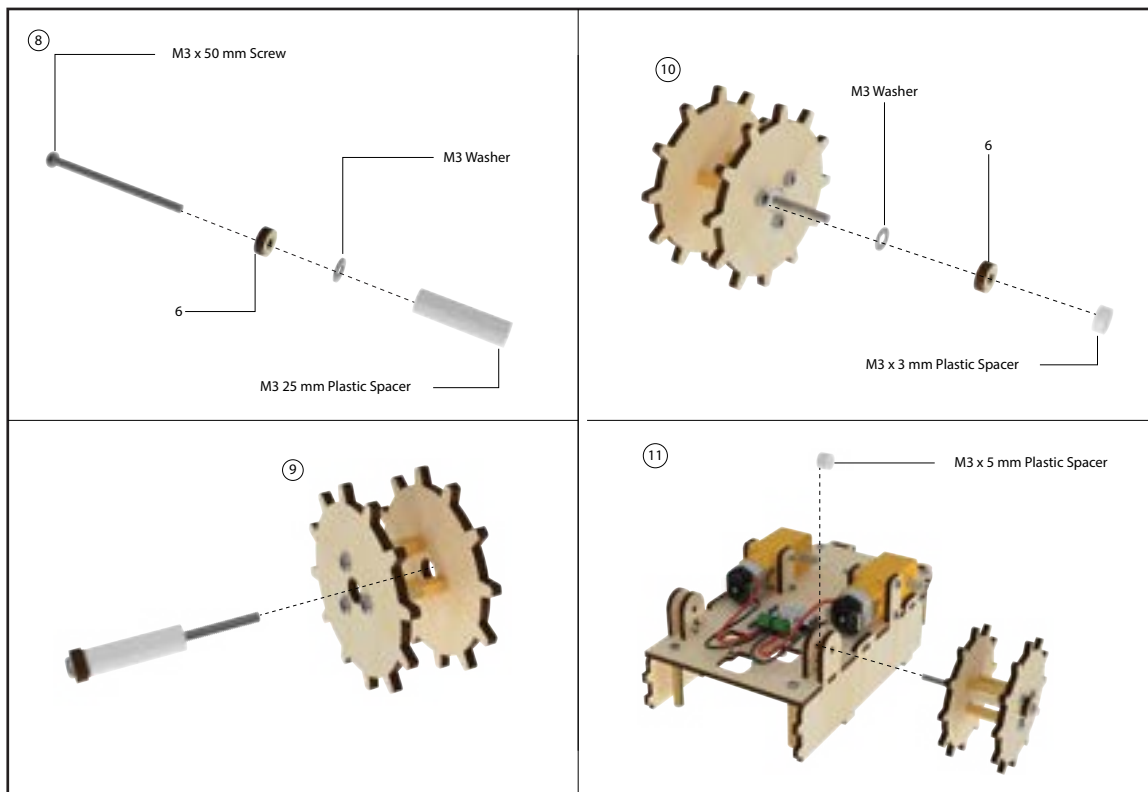
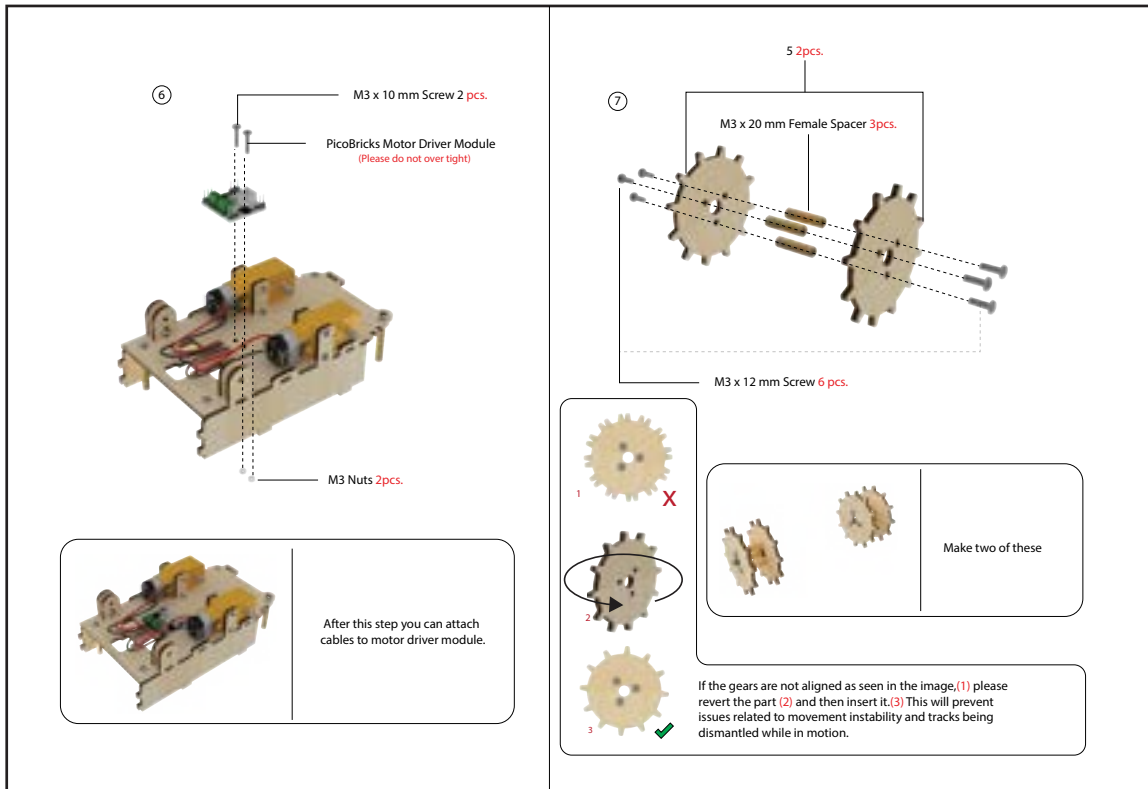


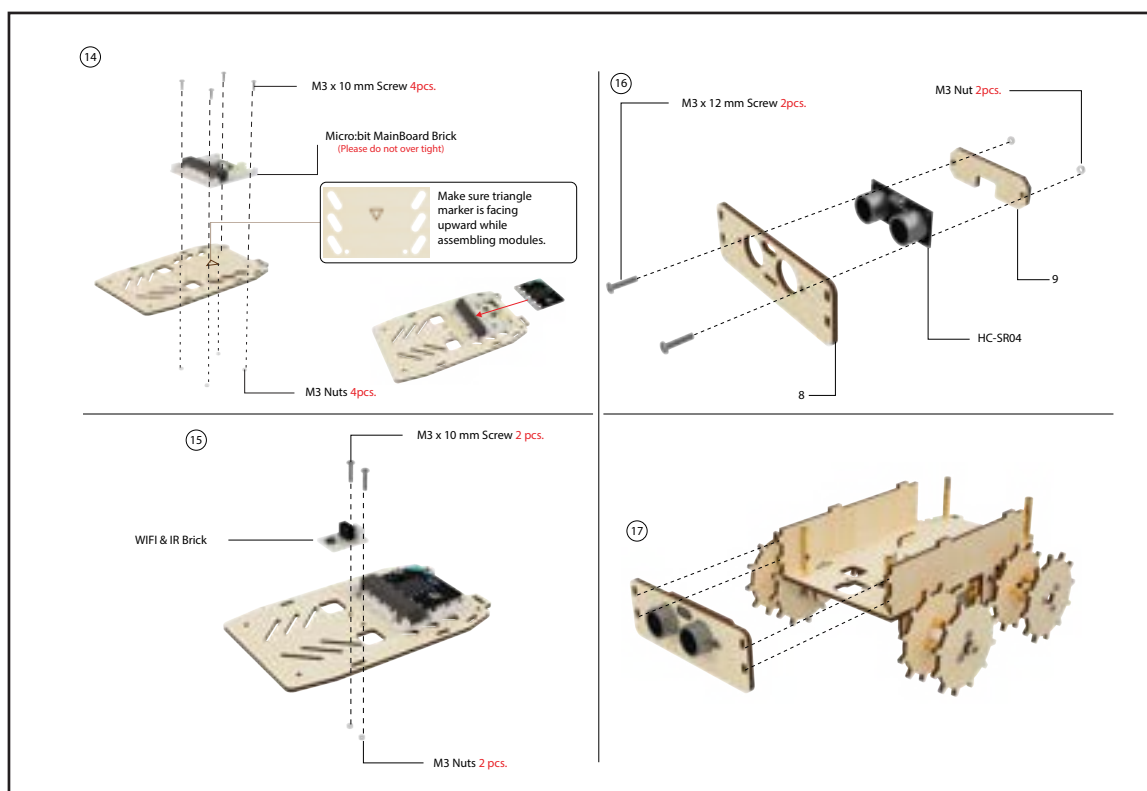
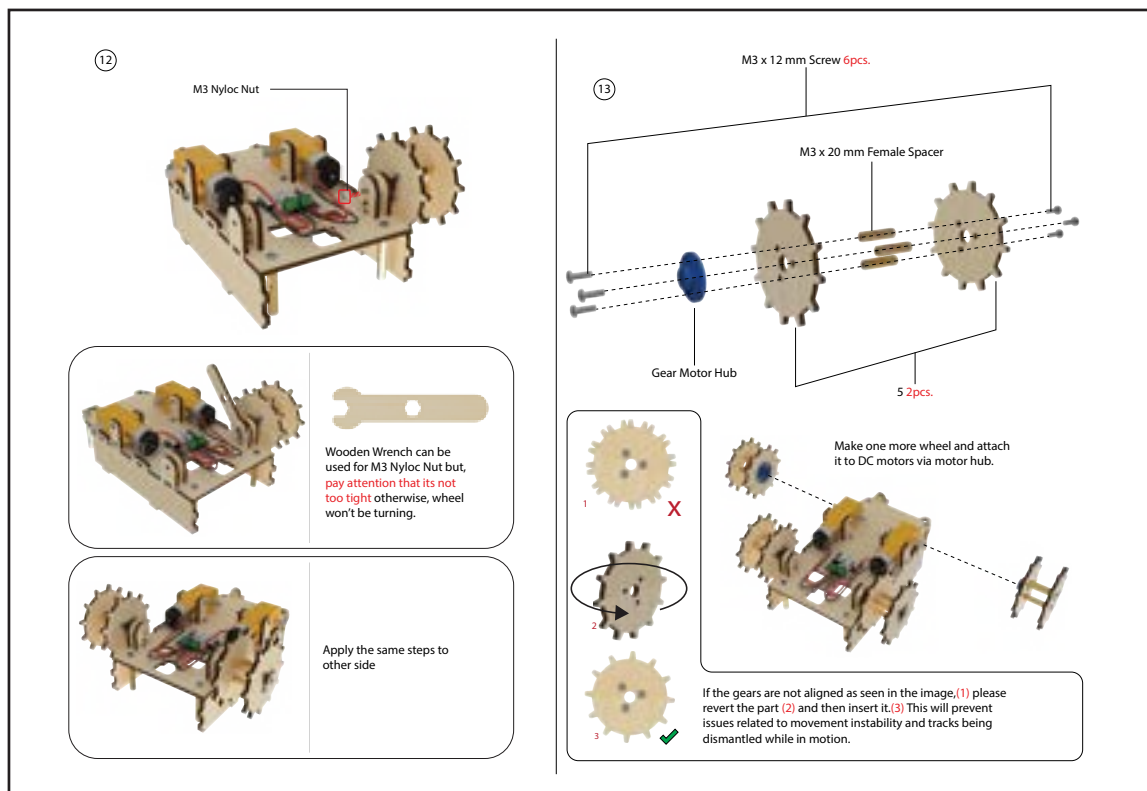
● Project Images:

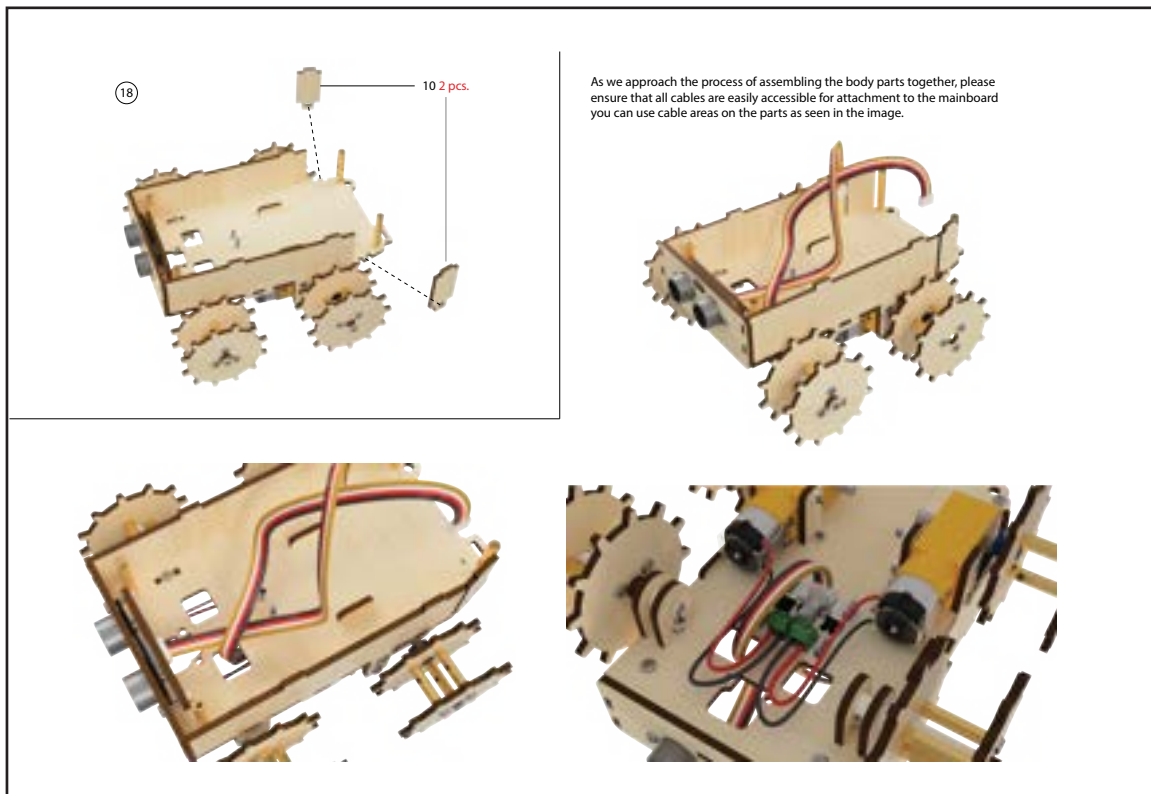


Setup Steps of The Project:



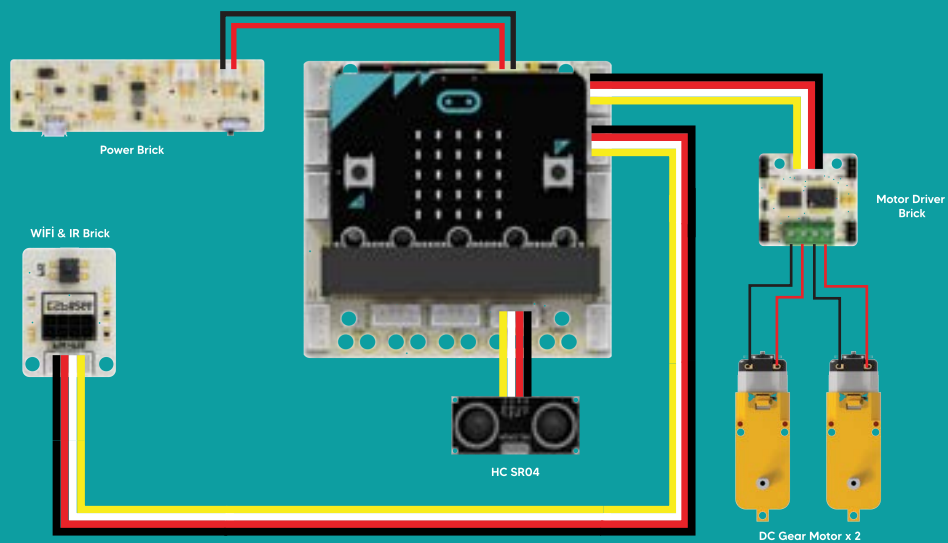


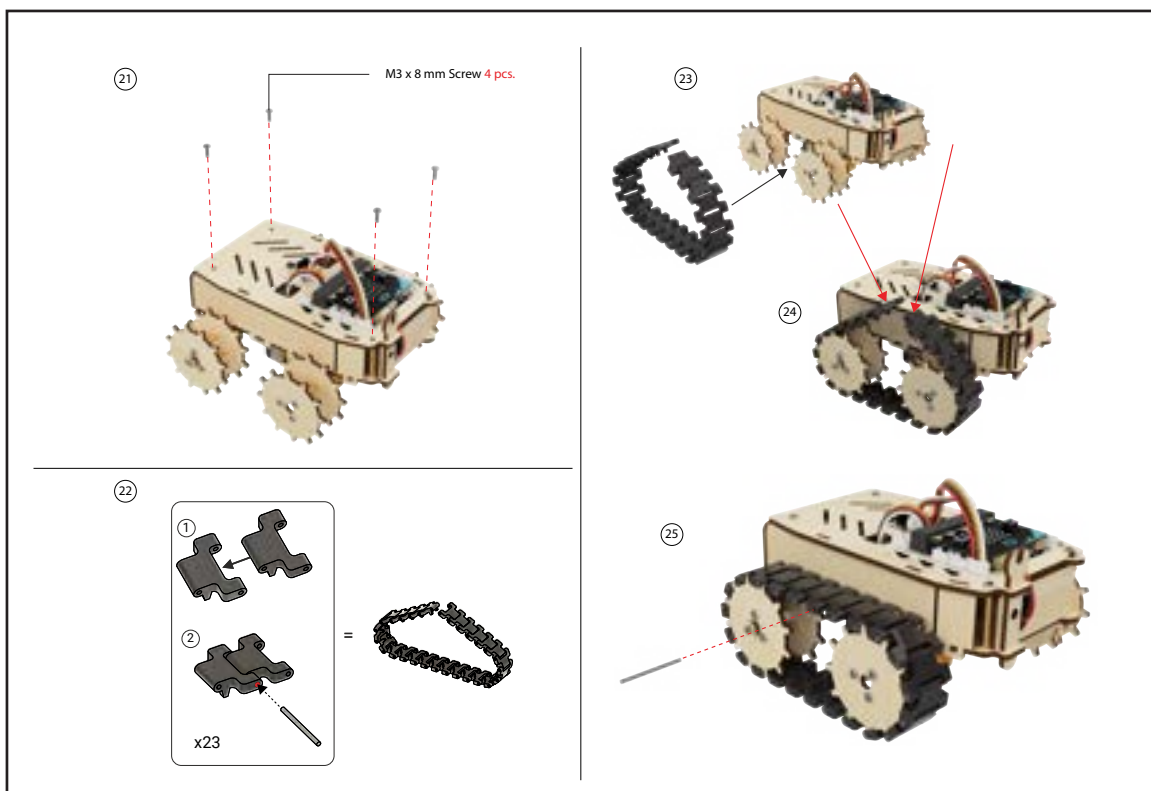
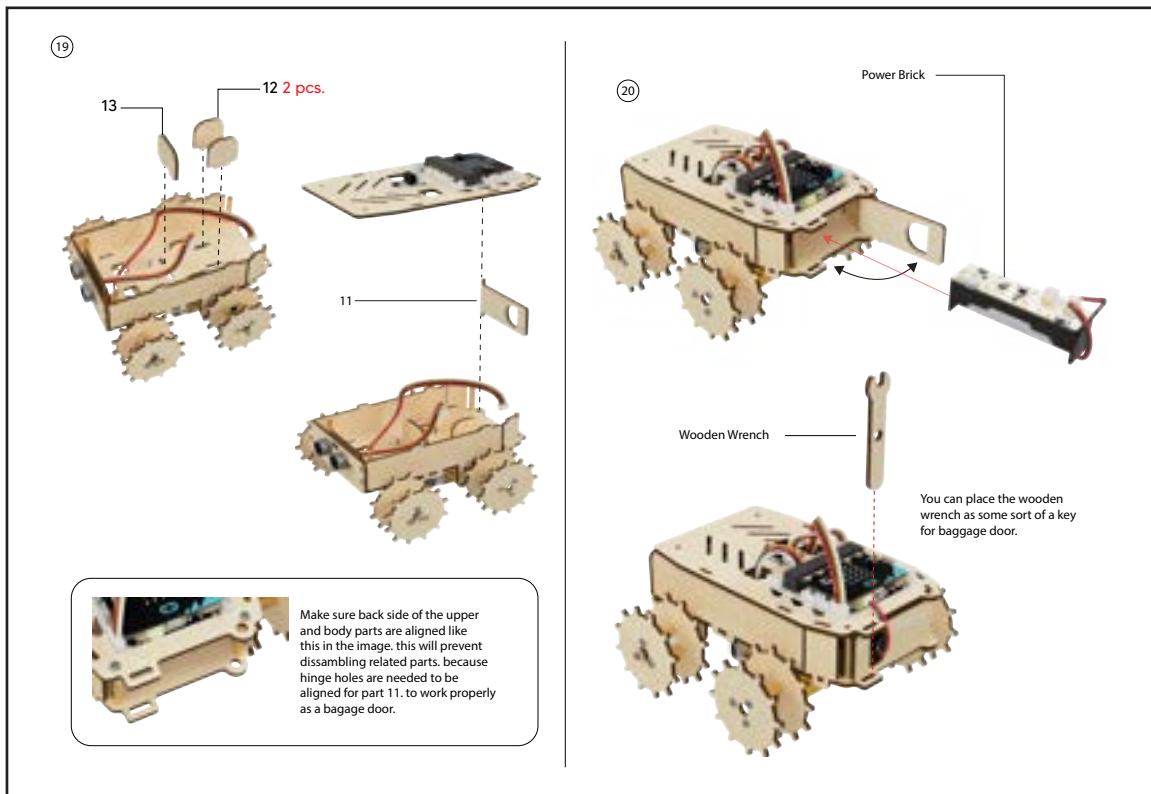


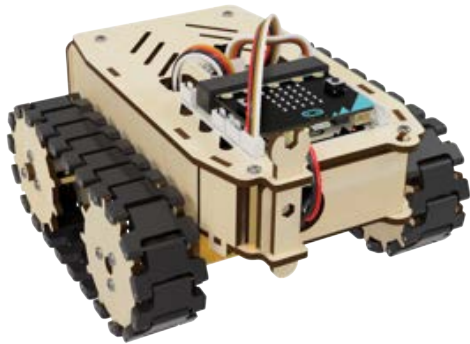


Setting Up The Circuit

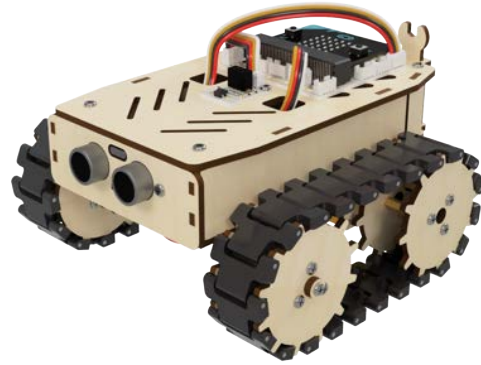
Let's get to know the circuit elements of Mini Tank that we completed the setup and make the circuit setup with PicoBricks modules.







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



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

MicroBlocks Code of The Project:

```

when PicoBricks-mb IR Code Received?
  clear display
  initialize local speed to 200
  if 24 = PicoBricks-mb IR Code
    say Motors: Forward
    display
    PicoBricks-mb set motor ALL speed speed (0-255) dir 0
  else if 82 = PicoBricks-mb IR Code
    say Motors: Backwards
    display
    PicoBricks-mb set motor ALL speed speed (0-255) dir 1
  else if 8 = PicoBricks-mb IR Code
    say Motors: Left
    display
    PicoBricks-mb set motor 1 speed 0 (0-255) dir 0
    PicoBricks-mb set motor 2 speed speed (0-255) dir 0
  else if 90 = PicoBricks-mb IR Code
    say Motors: Right
    display
    PicoBricks-mb set motor 1 speed speed (0-255) dir 0
    PicoBricks-mb set motor 1 speed 0 (0-255) dir 0
  else
    say Motors: Stopped
    display
    PicoBricks-mb set motor ALL speed 0 (0-255) dir 0
  
```

ROBOT CAR:
micro:bit v1 and v2

This car is controlled by the IR Remote controller. It also has a HC-SR04 distance sensor that helps to stop it 15 cm (adjustable) from any obstacles.

Four directional buttons on the IR Remote steer the car in those directions. Any other button pressed will stop all motors.

Distance sensor is default adjusted to stop within 15cm of any obstacles. You can change this as you wish.

NOTE:
HC-SR04 distance sensor shares pins with the Potentiometer.
TRIG: pin 2 ECHO: pin 1

For best results, please adjust the potentiometer dial to the middle (512) setting

```

when started
  forever
    if distance (cm) trigger 1 echo 2 <= 15
      display
      PicoBricks-mb set motor ALL speed 0 (0-255) dir 0
    wait 1000 millisecs
  
```

Trash Tech



Trash Tech Project

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

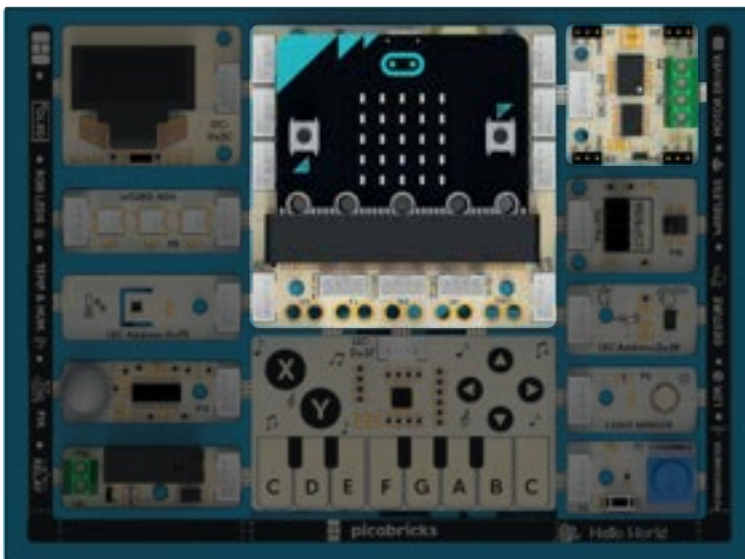
The Trash Tech is a fun kit that allows you to assemble the wooden pieces, sensors, and PicoBricks modules included in the set as specified in the installation guide. The goal of the project is to create an electronic trash bin that opens its lid by detecting objects using the HC-SR04 distance sensor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

Project Details:

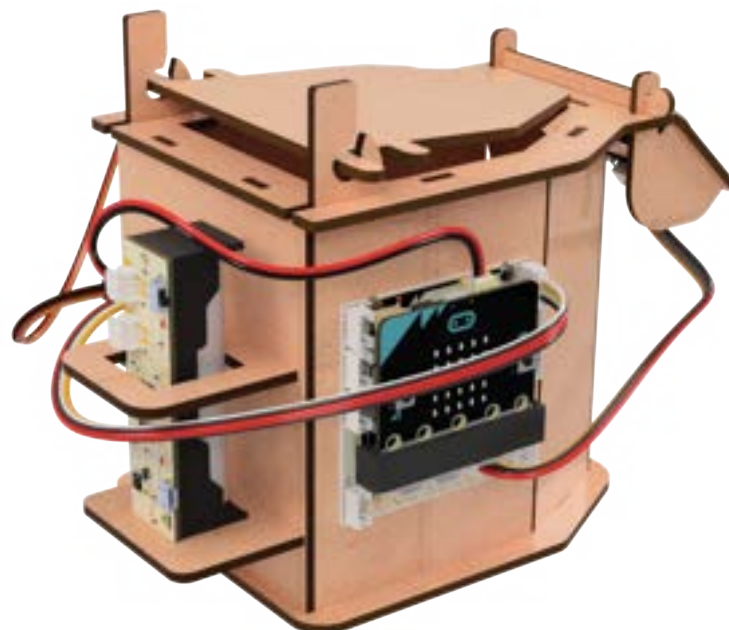
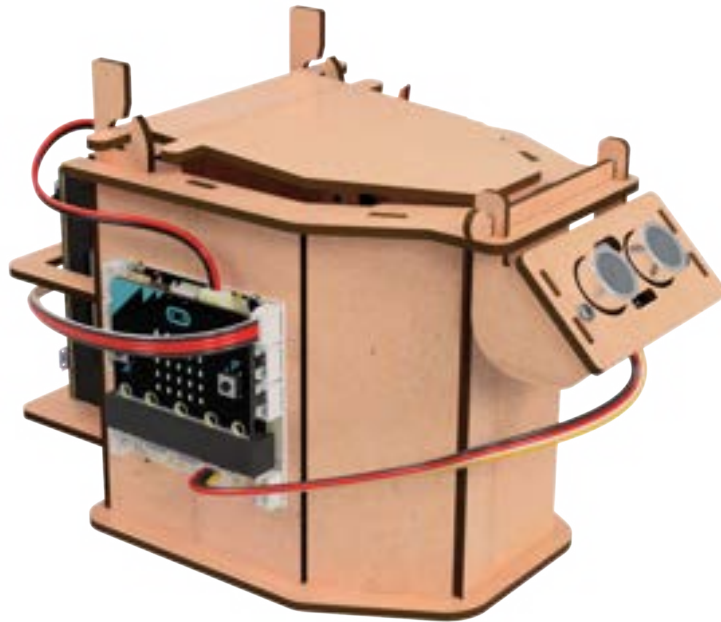
In this project, we detect the distance of our hand using the HCSR04 distance sensor and move the servo motor connected to the motor driver to the desired angle. This way, the lid of the trash bin opens.

Connection Diagram:

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



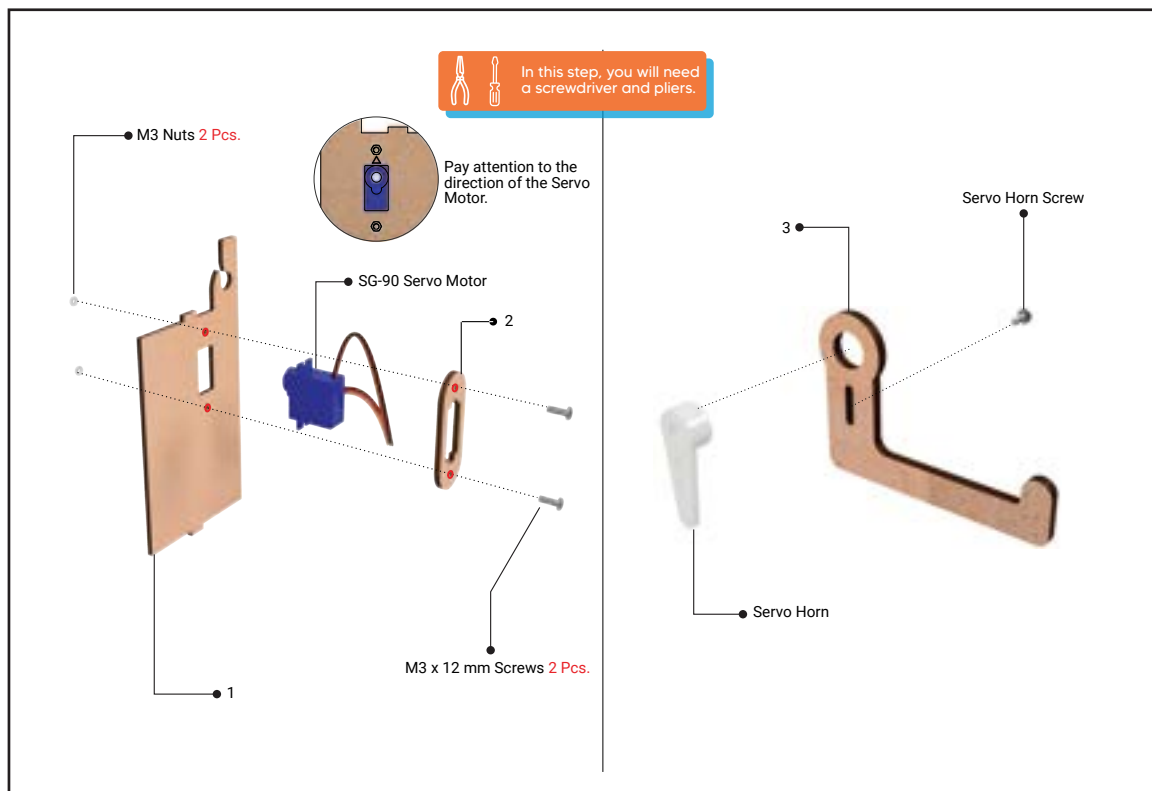
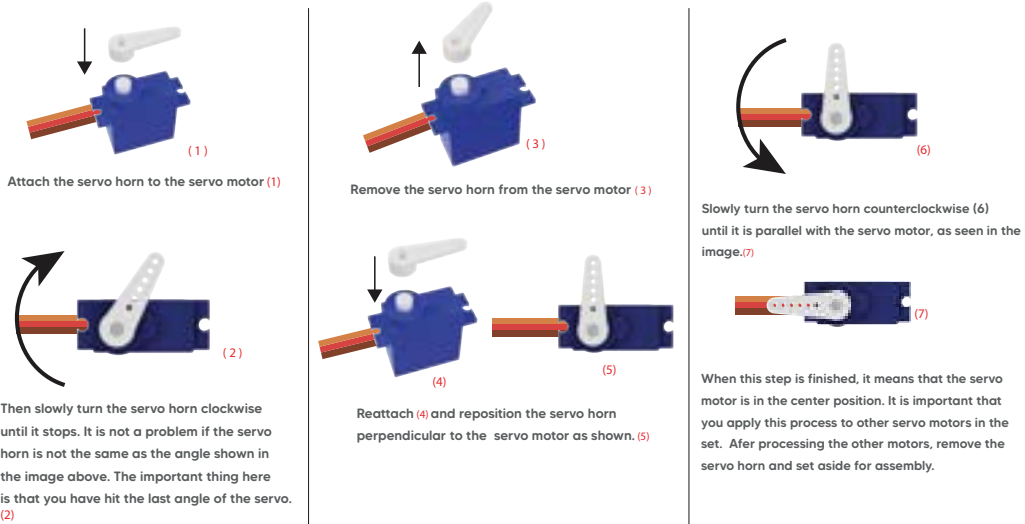
● Project Images:

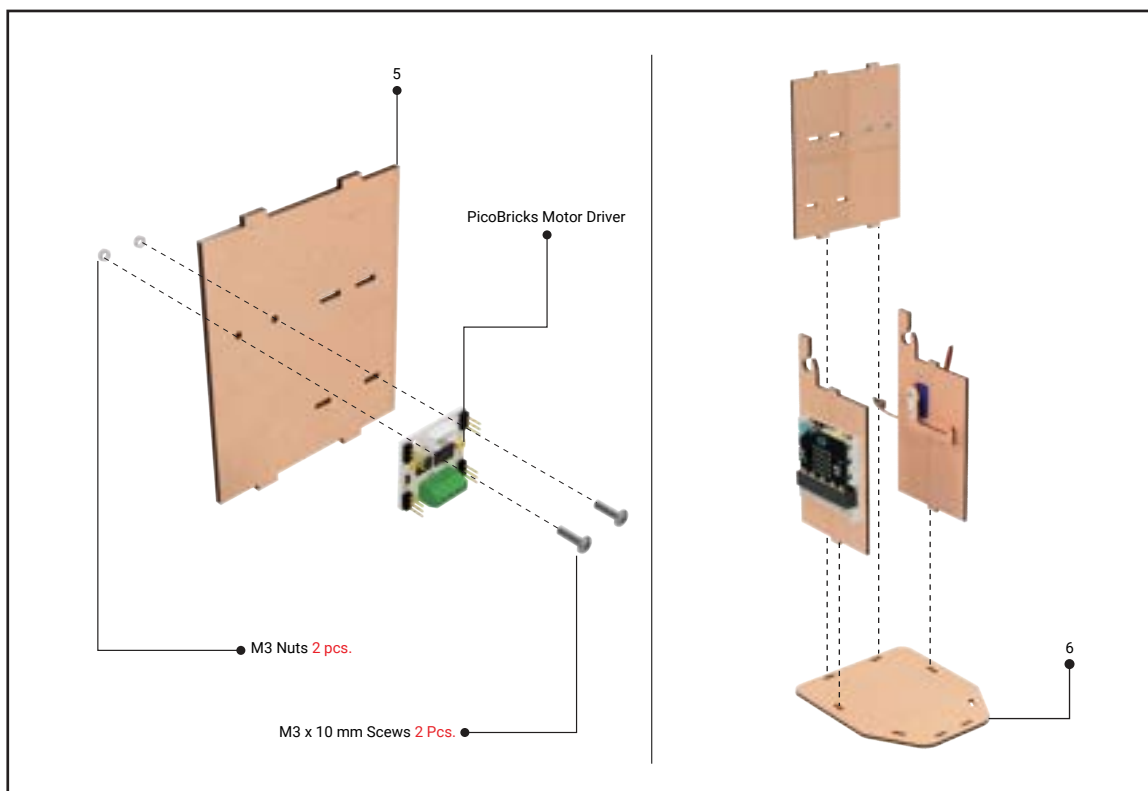
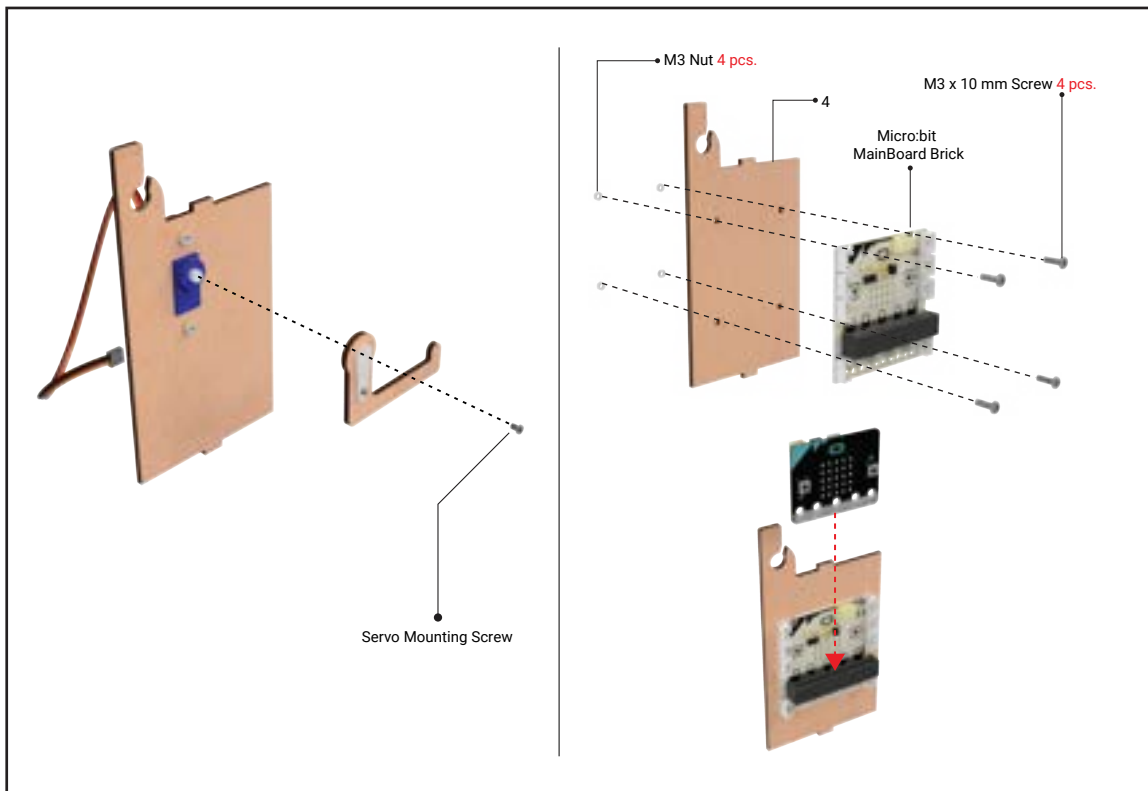


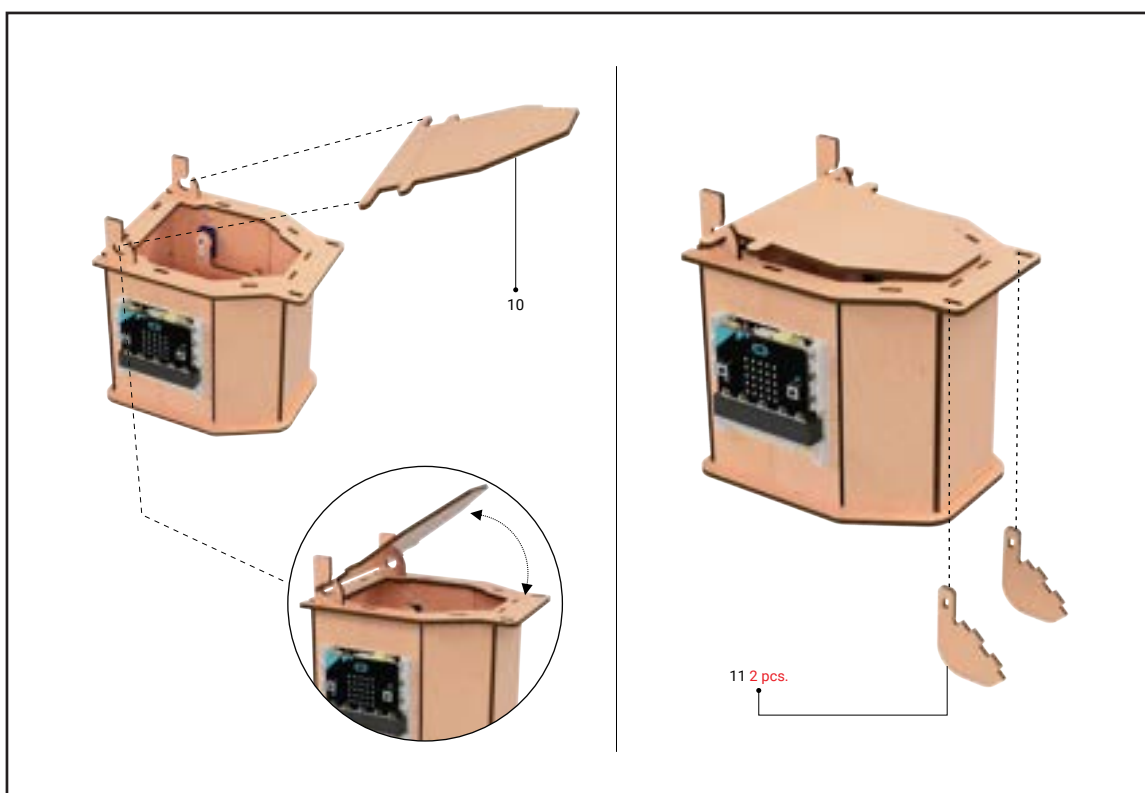
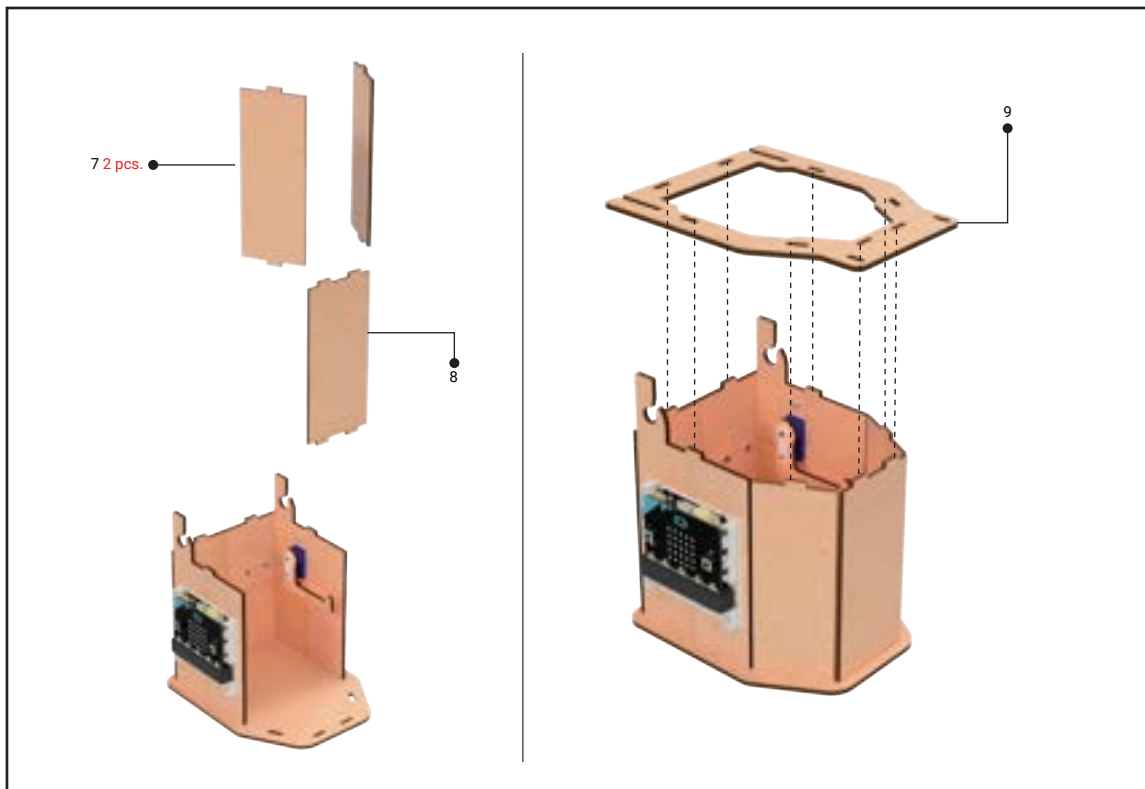
Setup Steps of The Project:

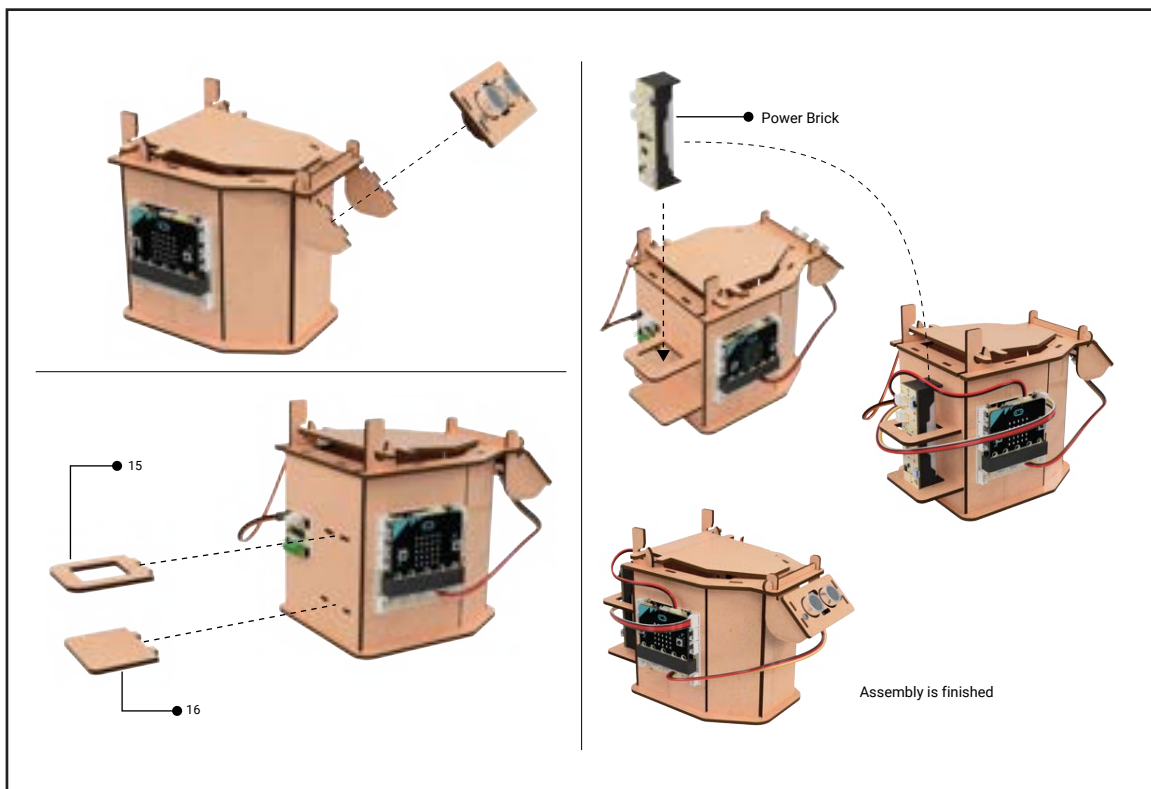
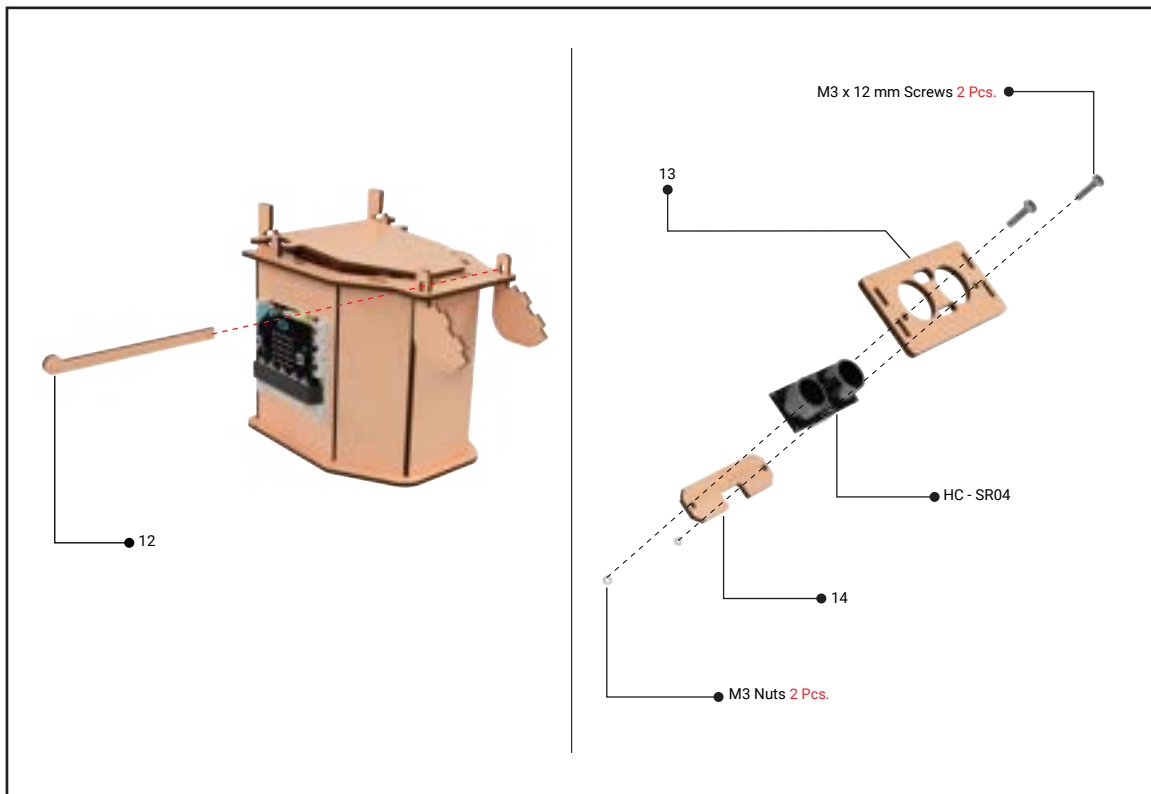
Servo Motor Calibration

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



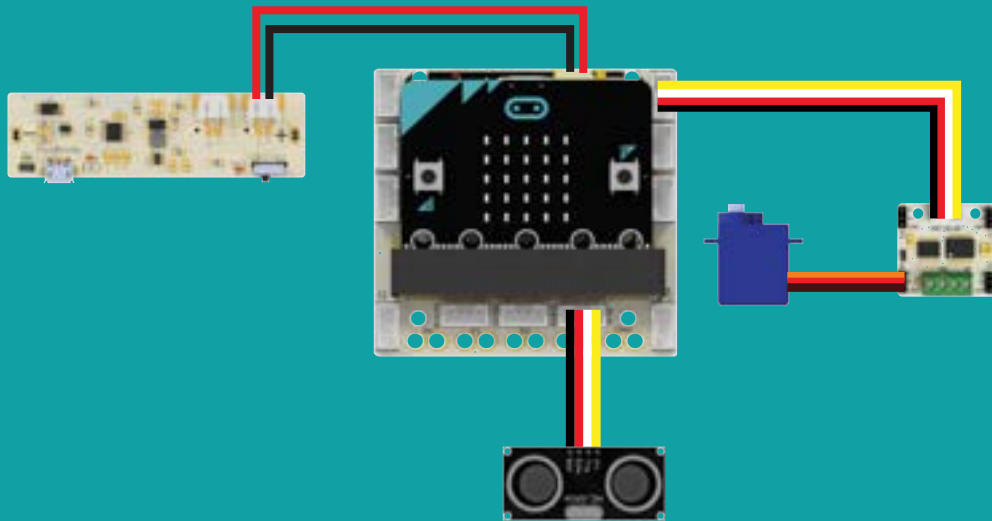






Unplugged 4 : Setup of Circuit

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.



MicroBlocks Code of The Project:

```

when PicoBricks-mb IR Code Received?
  clear display
  initialize local speed to 200
  if 24 = PicoBricks-mb IR Code
    say Motors: Forward
    display
    PicoBricks-mb set motor ALL speed speed (0-255) dir 0
  else if 82 = PicoBricks-mb IR Code
    say Motors: Backwards
    display
    PicoBricks-mb set motor ALL speed speed (0-255) dir 1
  else if 8 = PicoBricks-mb IR Code
    say Motors: Left
    display
    PicoBricks-mb set motor 1 speed 0 (0-255) dir 0
    PicoBricks-mb set motor 2 speed speed (0-255) dir 0
  else if 90 = PicoBricks-mb IR Code
    say Motors: Right
    display
    PicoBricks-mb set motor 1 speed speed (0-255) dir 0
    PicoBricks-mb set motor 1 speed 0 (0-255) dir 0
  else
    say Motors: Stopped
    display
    PicoBricks-mb set motor ALL speed 0 (0-255) dir 0
  
```

ROBOT CAR:
micro:bit v1 and v2

This car is controlled by the IR Remote controller. It also has a HC-SR04 distance sensor that helps to stop it 15 cm (adjustable) from any obstacles.

Four directional buttons on the IR Remote steer the car in those directions. Any other button pressed will stop all motors.

Distance sensor is default adjusted to stop within 15cm of any obstacles. You can change this as you wish.

NOTE:
HC-SR04 distance sensor shares pins with the Potentiometer.
TRIG: pin 2 ECHO: pin 1

For best results, please adjust the potentiometer dial to the middle (512) setting

```

when started
  forever
    if distance (cm) trigger 1 echo 2 <= 15
      display
      PicoBricks-mb set motor ALL speed 0 (0-255) dir 0
    wait 1000 millisecs
  
```

Money Box



Money Box Project

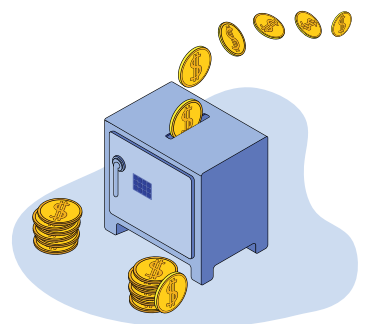
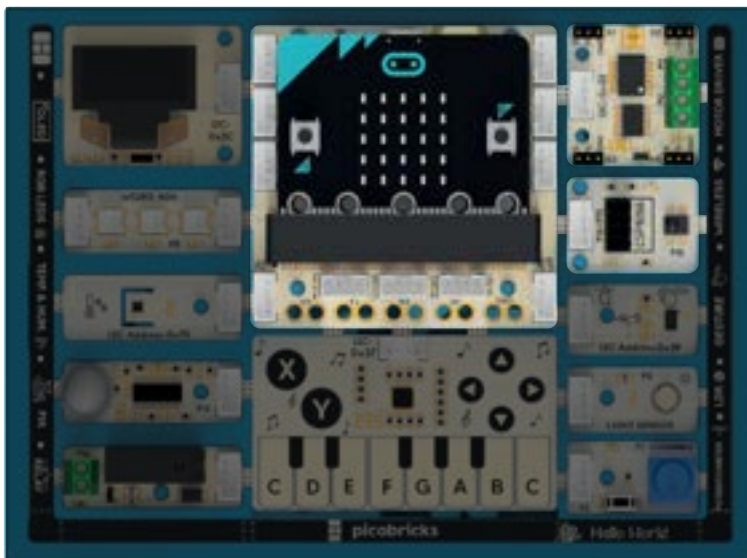
PicoBricks Money Box can detect objects placed in its receptacle through distance sensor in the front of it and automatically lifts its receptacle to take in these objects.

Project Details:

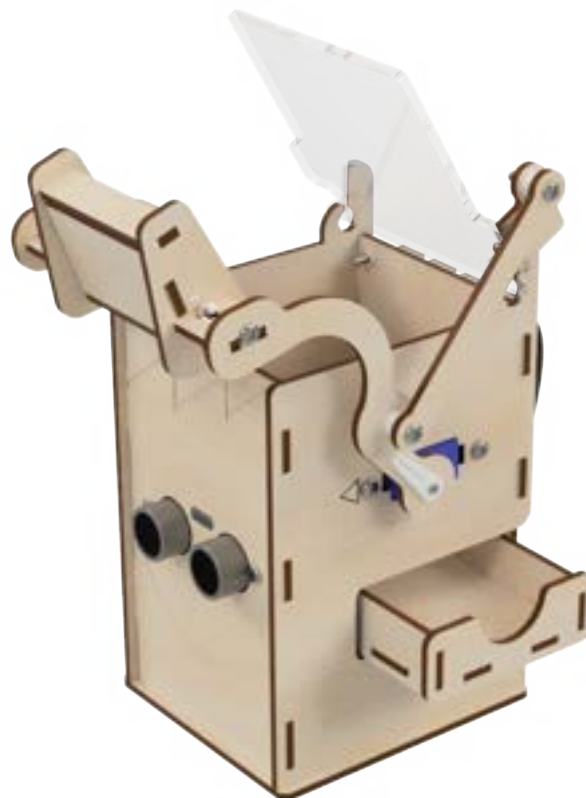
In this project, when the distance sensor detects the object placed in the receptacle, the servo motor connected to the motor driver is adjusted to the angle specified in the code, and then the object inside the receptacle is dropped into the Money Box.

Connection Diagram:

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



● Project Images:



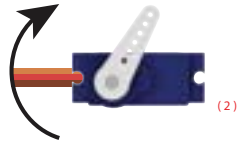
Setup Steps of The Project:

Servo Motor Calibration

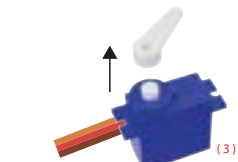
Before starting the assembly, you have to manually calibrate the angles of the servo motors. Otherwise, Servo Motors won't be working properly.



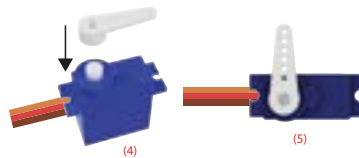
Attach the servo horn to the servo motor (1)



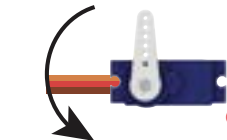
Then slowly turn the servo horn clockwise until it stops. It is not a problem if the servo horn is not the same as the angle shown in the image above. The important thing here is that you have hit the last angle of the servo. (2)



Remove the servo horn from the servo motor (3)



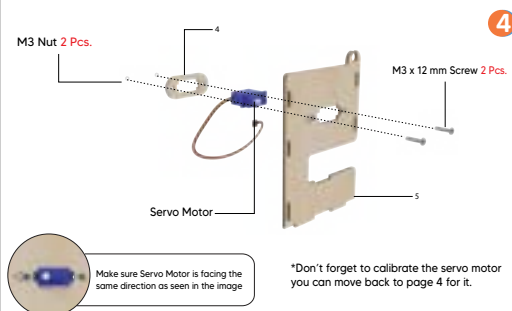
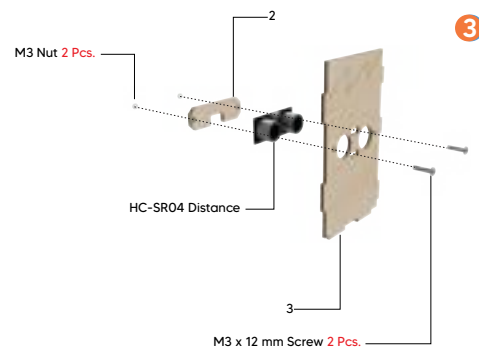
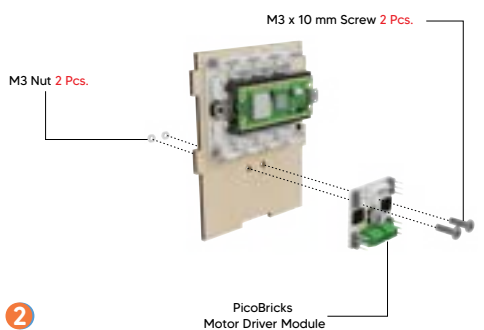
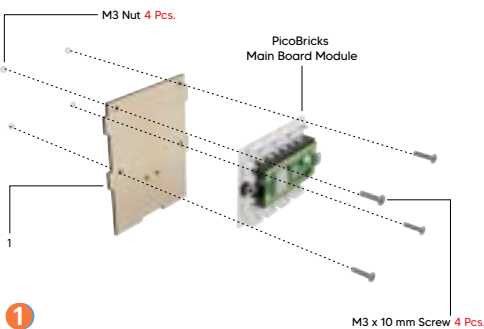
Reattach (4) and reposition the servo horn perpendicular to the servo motor as shown. (5)

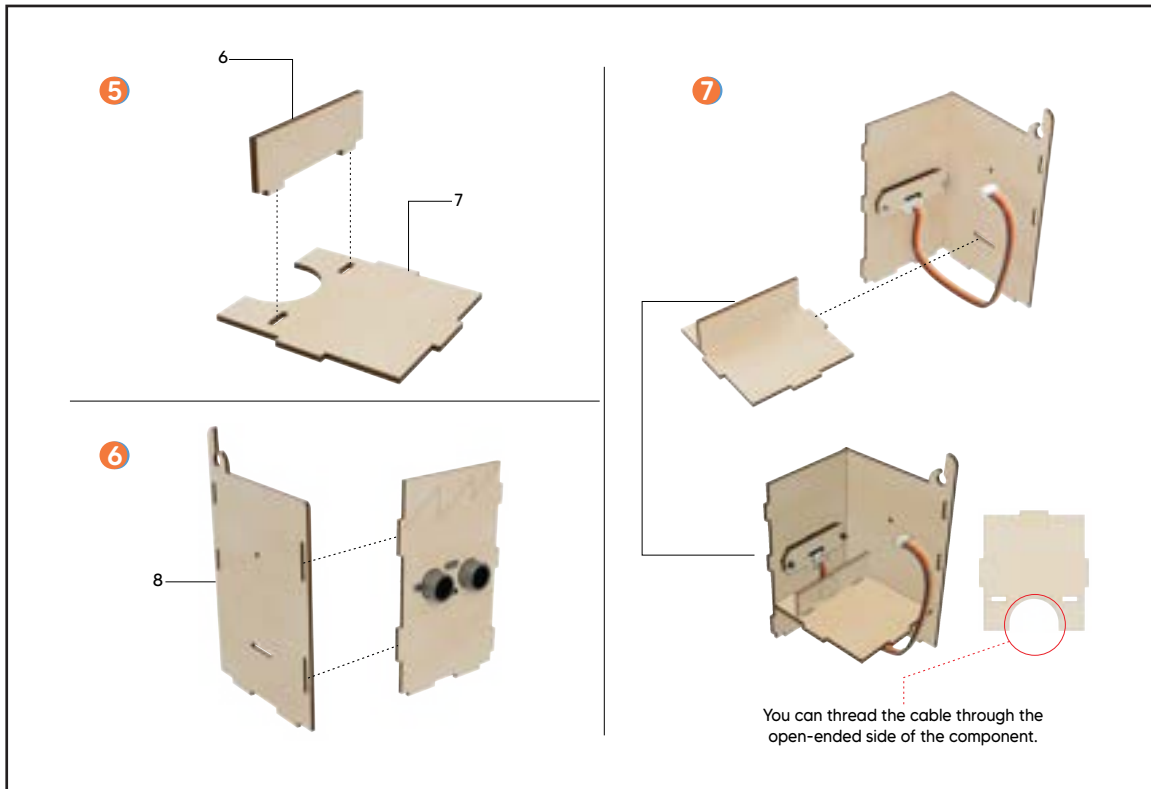


Slowly turn the servo horn counterclockwise (6) until it is parallel with the servo motor, as seen in the image. (7)



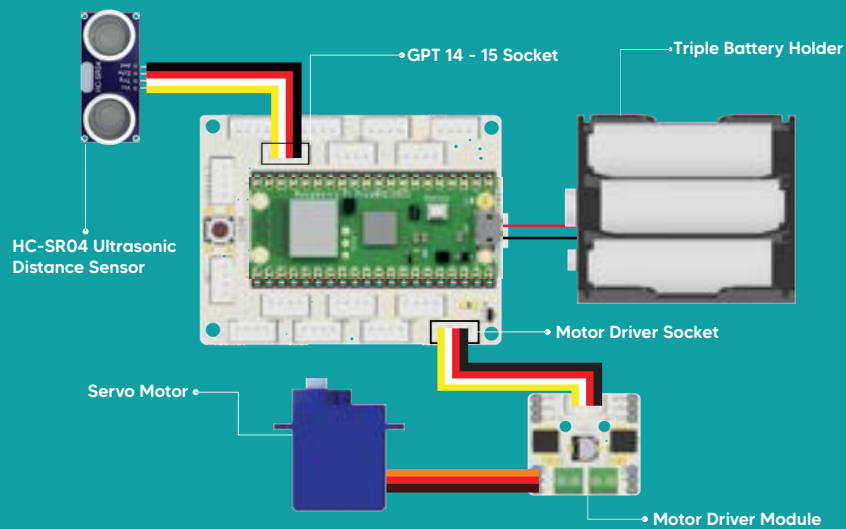
When this step is finished, it means that the servo motor is in the center position. It is important that you apply this process to other servo motors in the set. After processing the other motors, remove the servo horn and set aside for assembly.

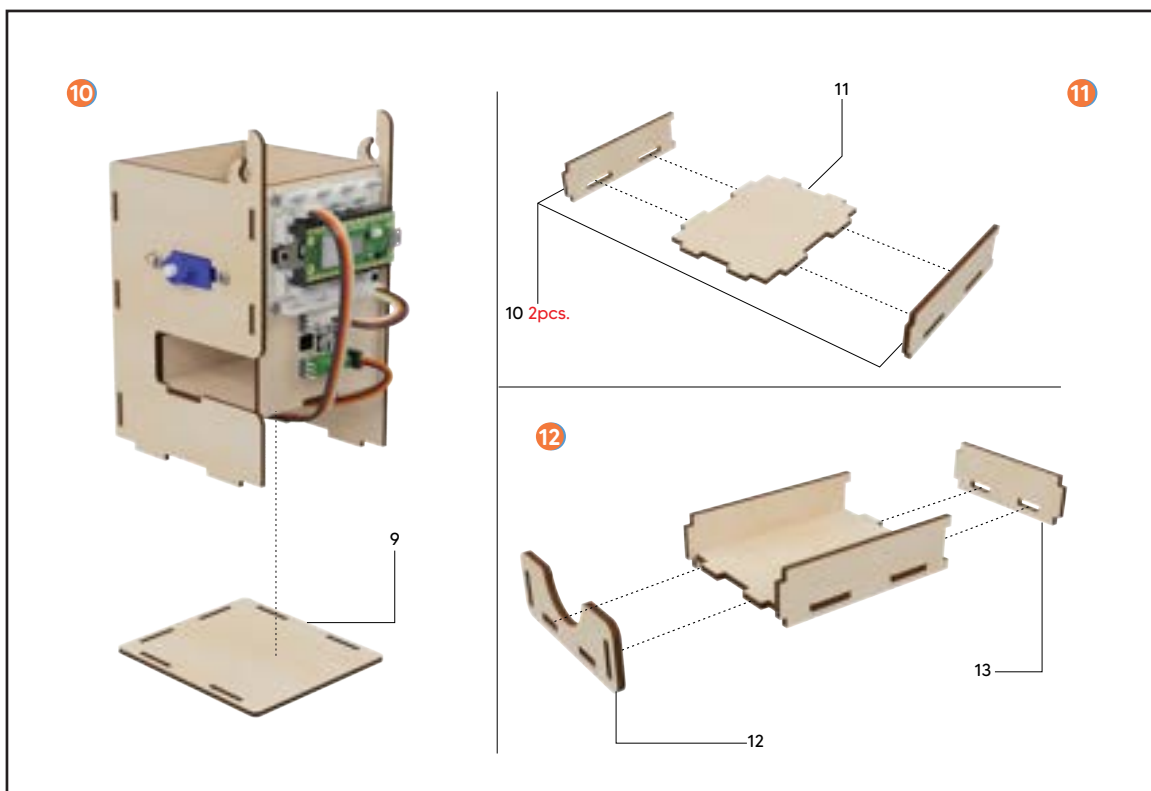
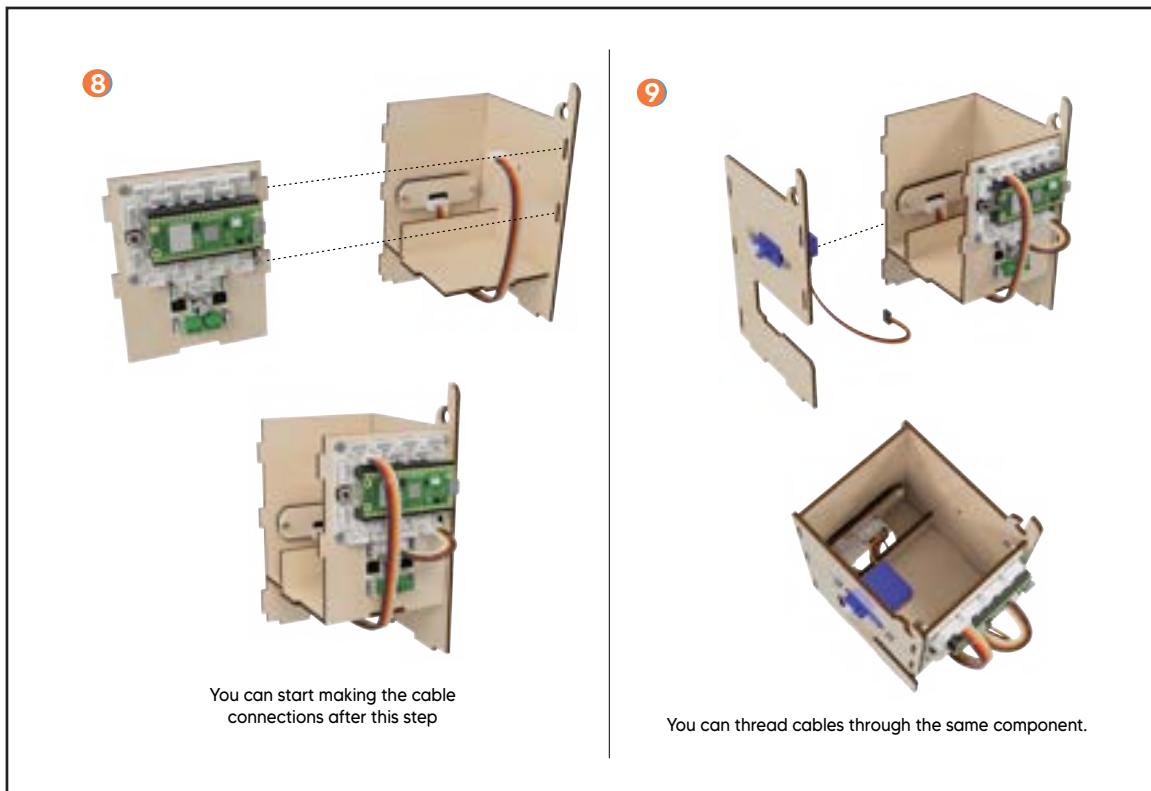


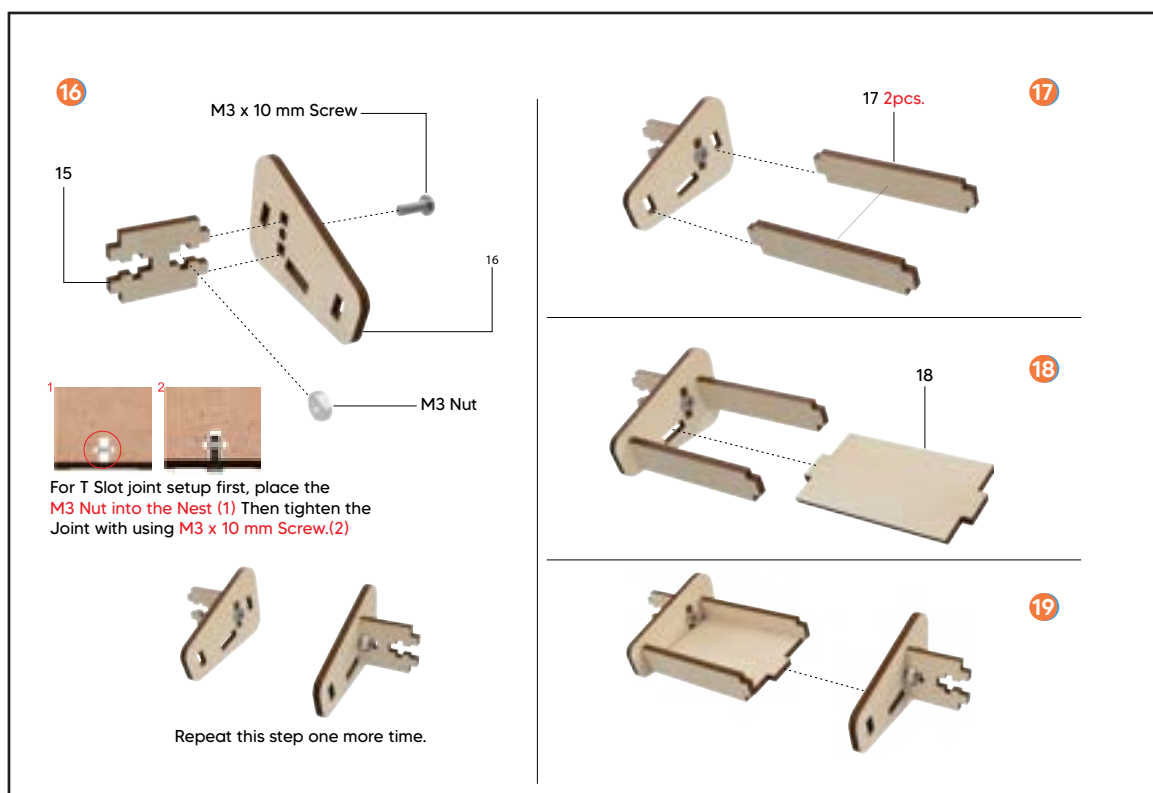
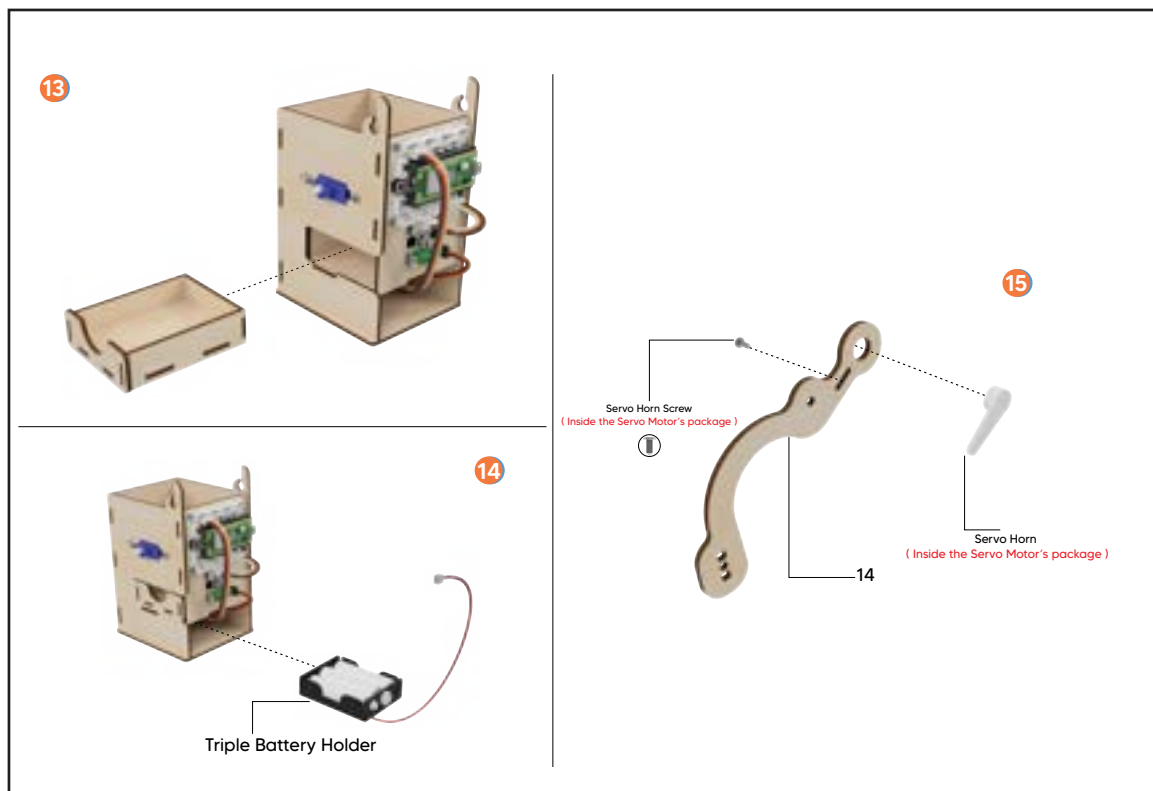


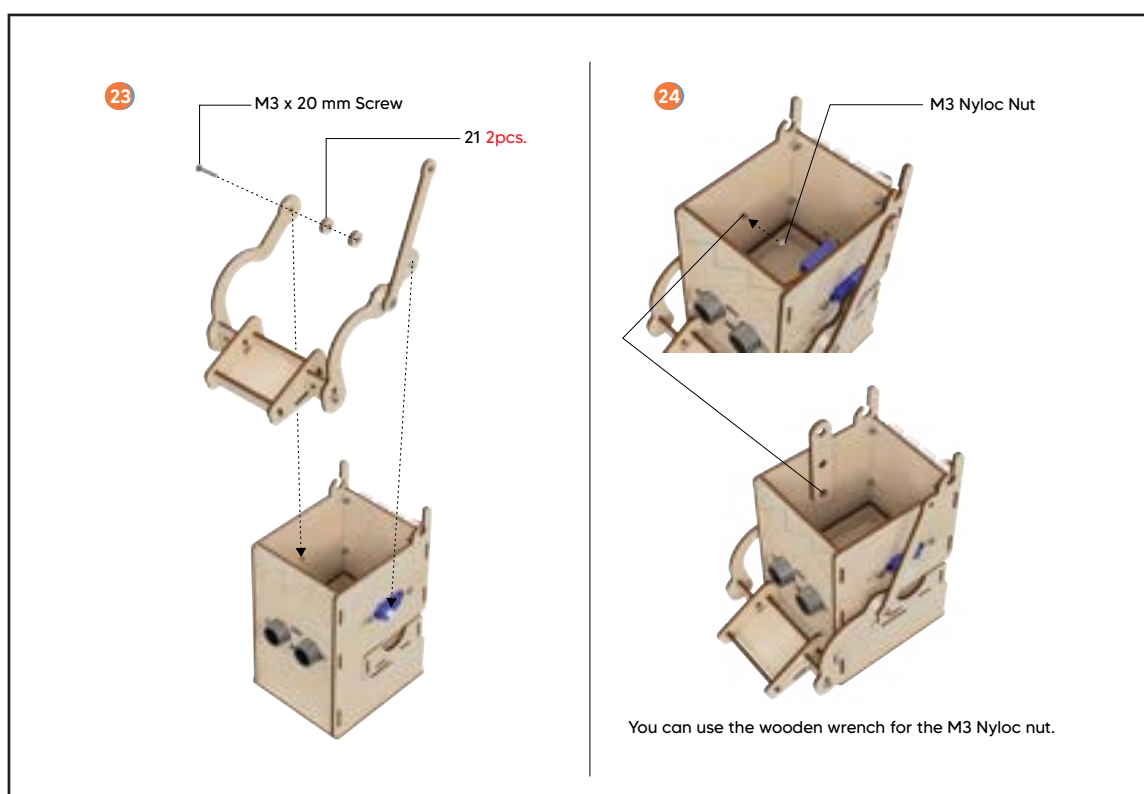
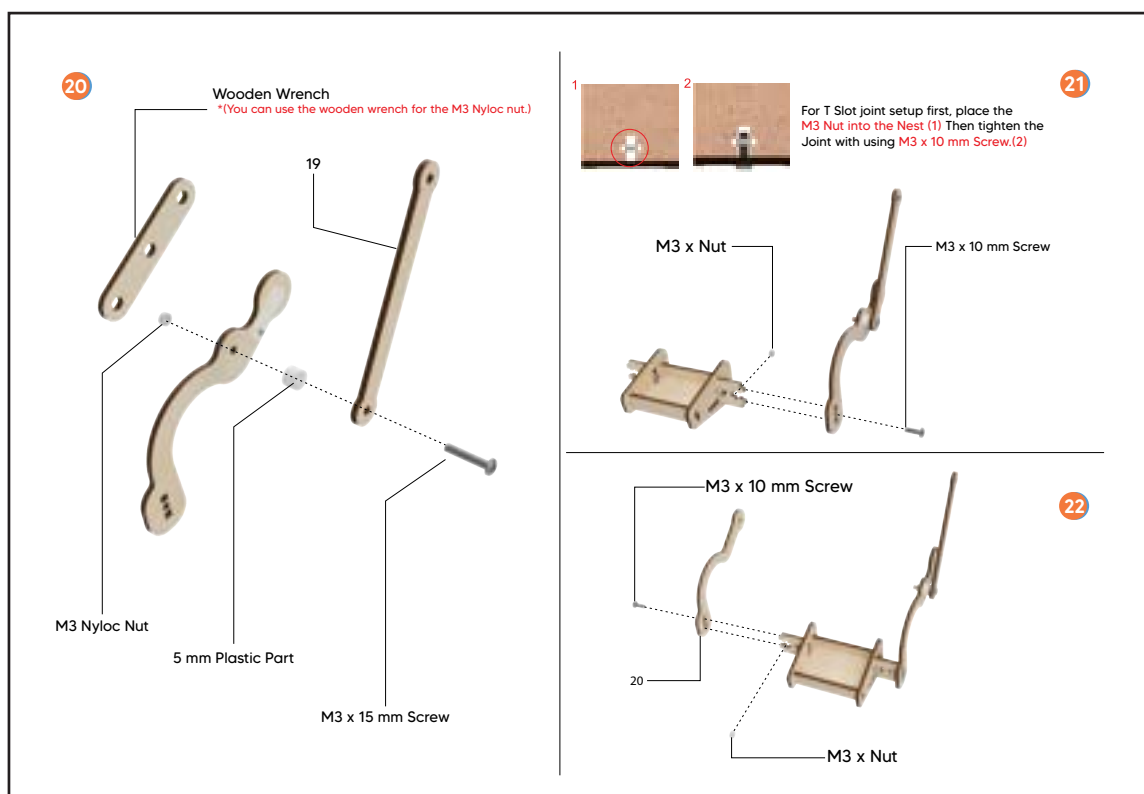
Setting Up The Circuit

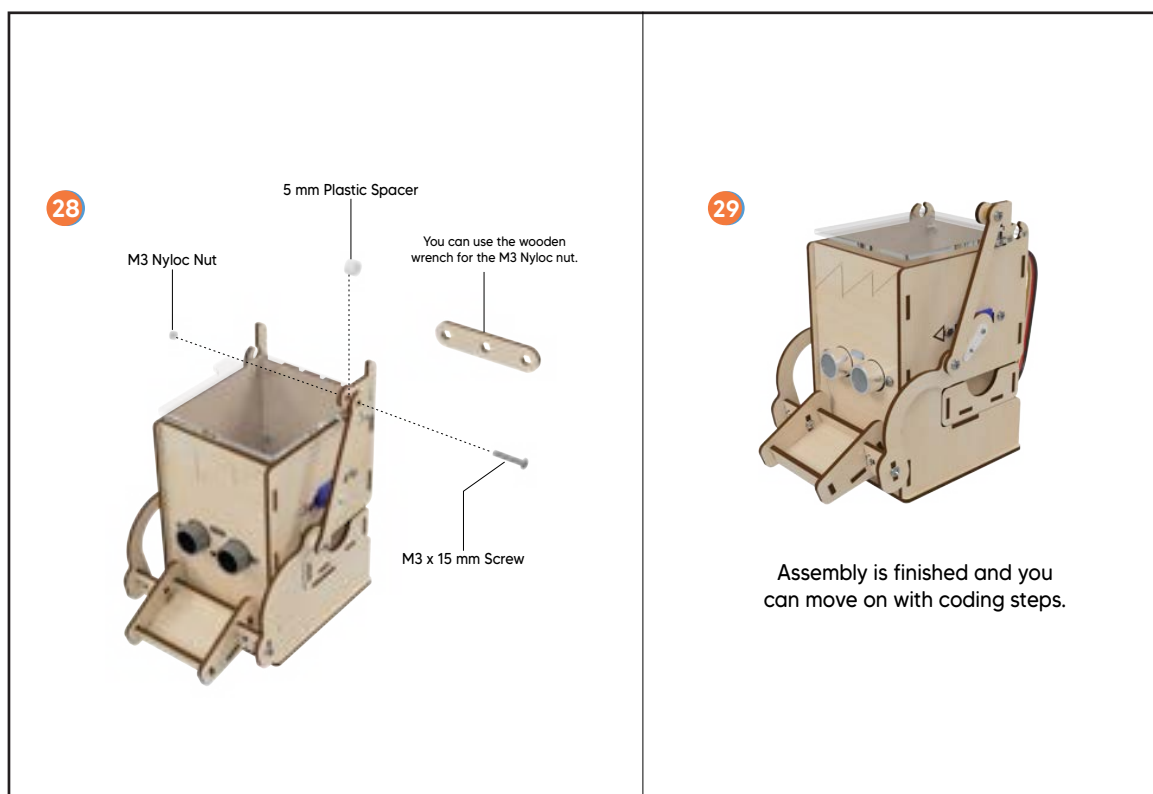
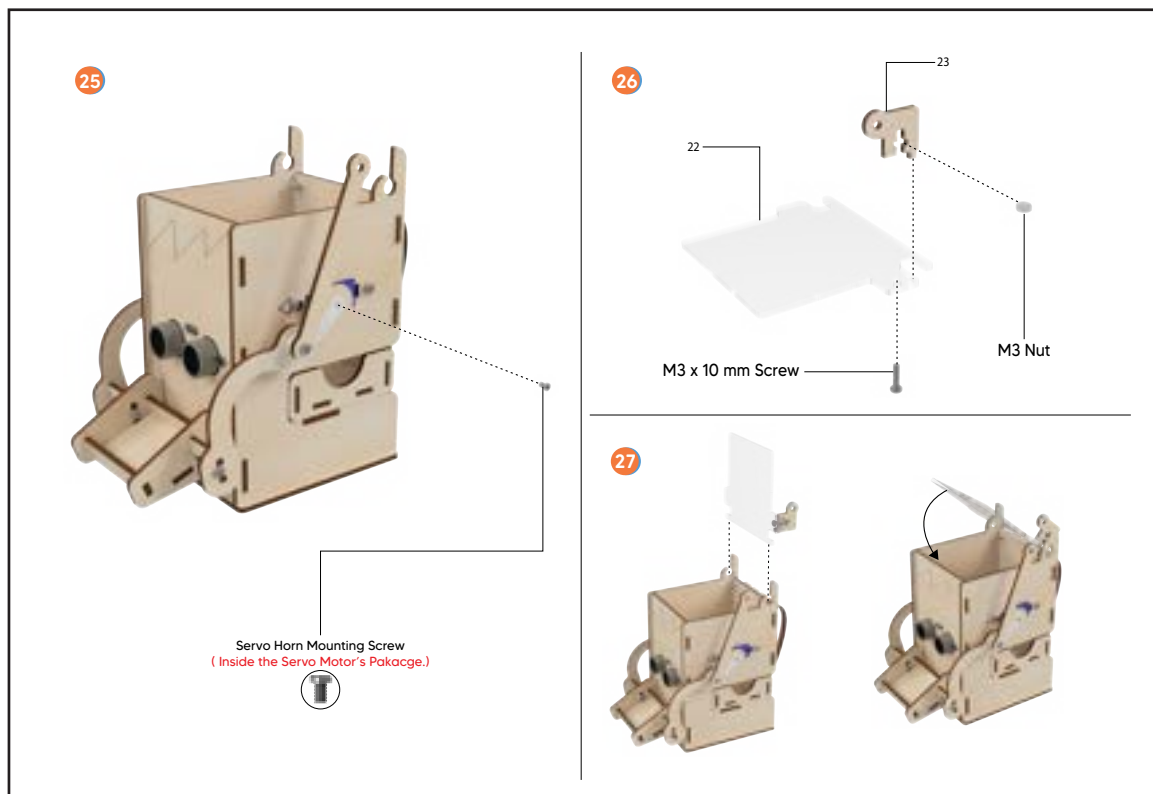
Let's get to know the circuit elements of Money Box that we completed the setup and make the circuit setup with PicoBricks modules.











MicroBlocks Code of The Project:

```
comment
MONEY BOX:

micro:bit v1 and v2

Box is closed by setting the servo to 145 degrees.

Distance sensor detects a hand nearby (less than 7cm away) and sets objectDetected to true.

It then waits until the hand is moved away (>12 cm away).
If objectDetected is true, it drops the coin into the box by setting the servo to 90 degrees.
Otherwise, it closes the box.

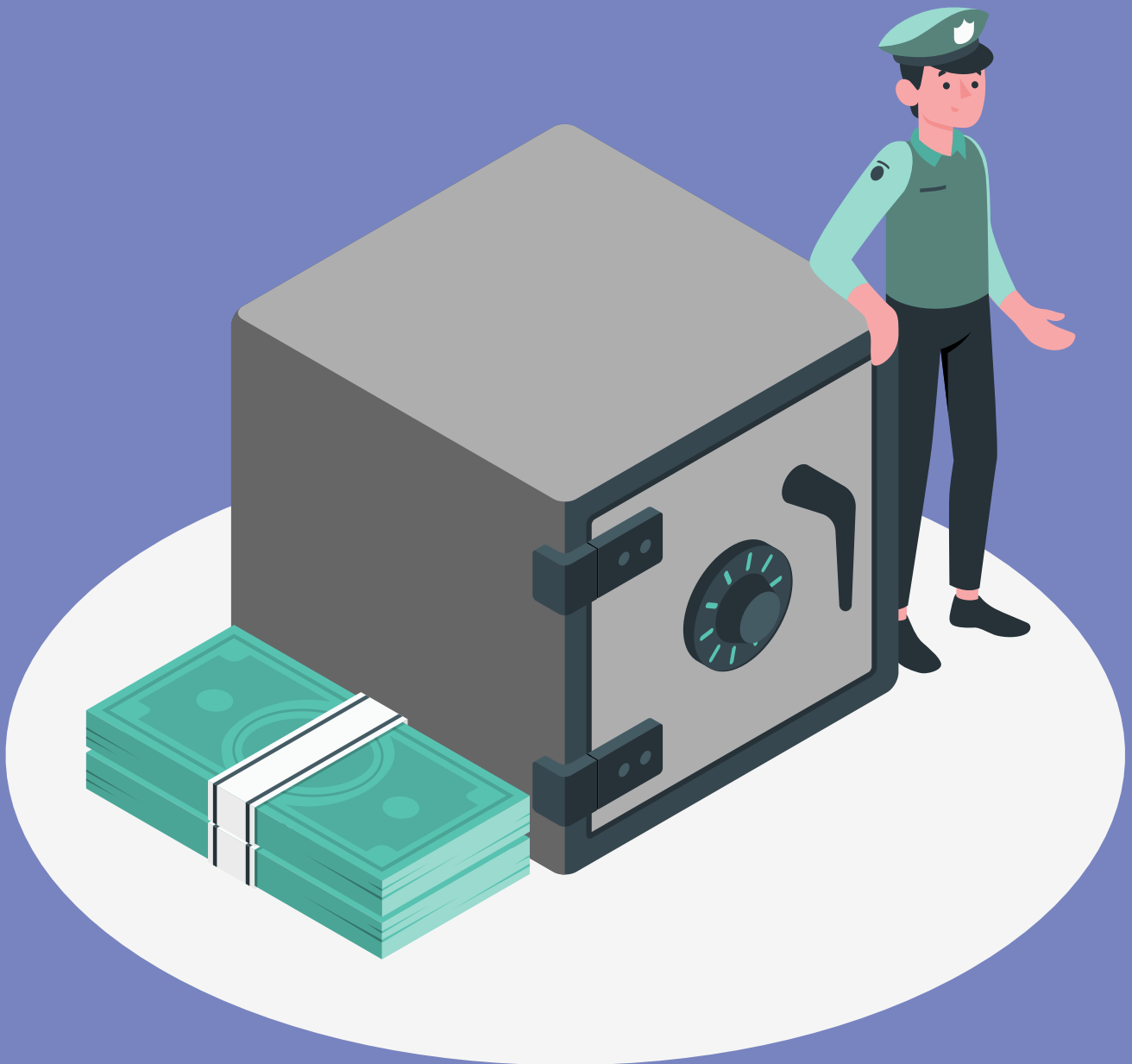
Distance Sensor requires 5V to operate.
```

```
when started
  _PicoBricks-mb init Library
  PicoBricks-mb set servo 3 angle 145 (0-180)
  set objectDetected to 
```

```
when distance (cm) trigger 1 echo 2 <= 7
  set objectDetected to 
```

```
when distance (cm) trigger 1 echo 2 > 12
  if objectDetected
    PicoBricks-mb set servo 3 angle 90 (0-180)
    set objectDetected to 
  else
    wait 500 millisecs
    PicoBricks-mb set servo 3 angle 145 (0-180)
```

Safe Box



Safe Box Project

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

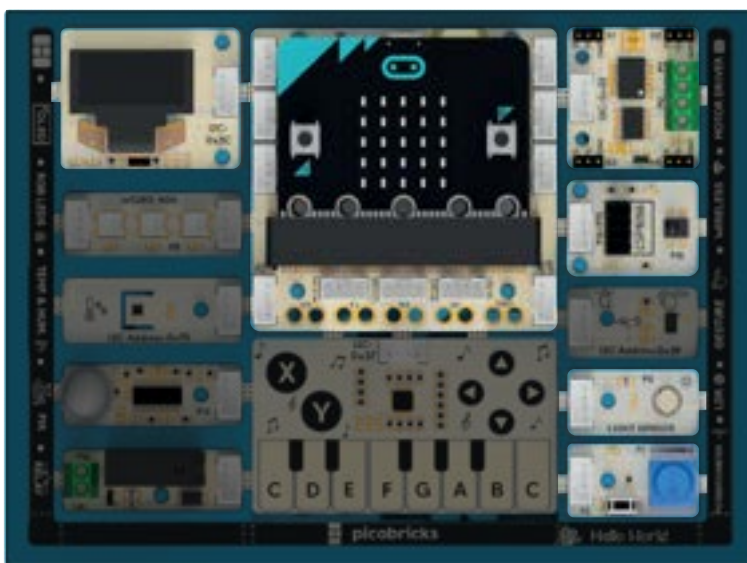
In this project, when the correct password is entered by using a potentiometer and button, the door of the safe opens. After closing the door, it automatically locks thanks to the LDR sensor inside the safe.

Project Details:

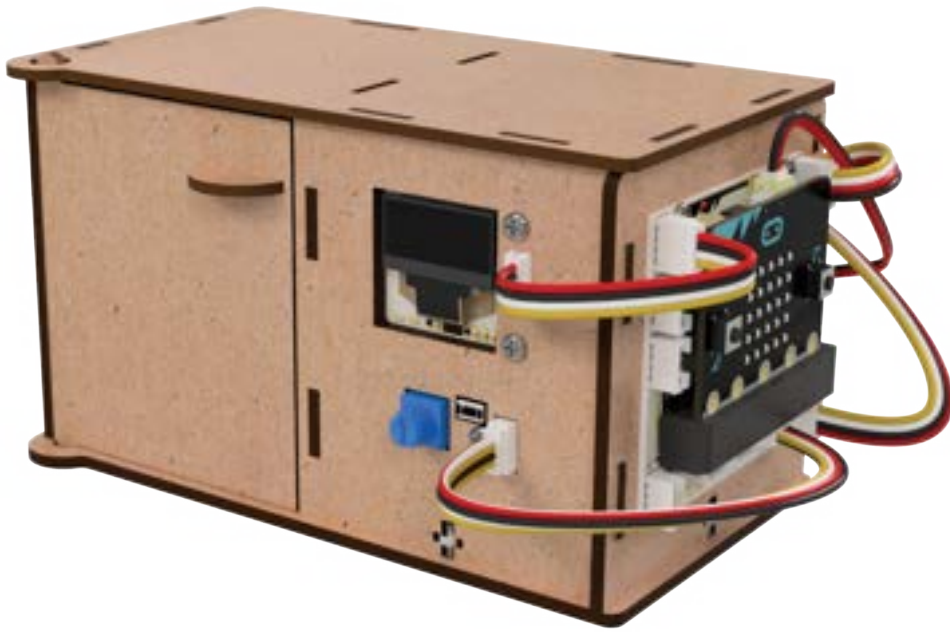
In this project, when we correctly enter the password we have set in the code by using the potentiometer and button module, the servo motor moves to the specified position, and the door opens. Through the LDR module, the closure of the pass box lid is detected, triggering the servo motor to operate and 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 Pass Box project, we will obtain visual output.

Connection Diagram:

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



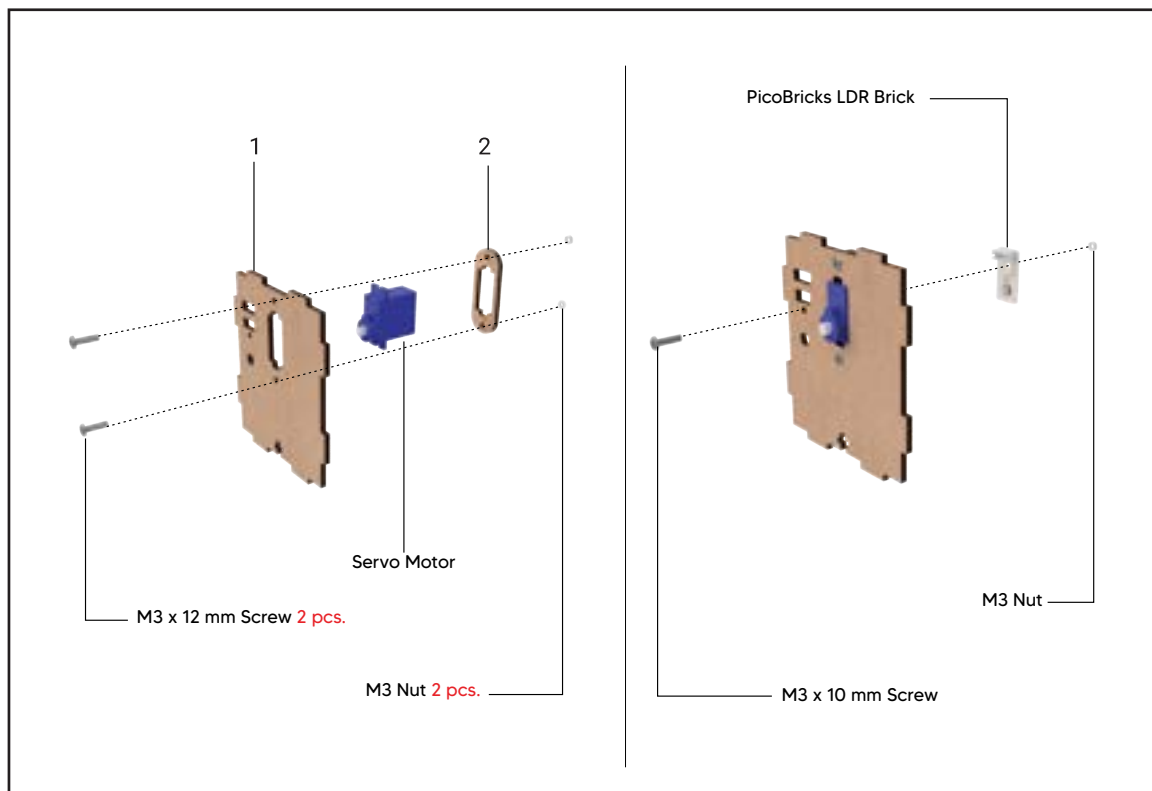
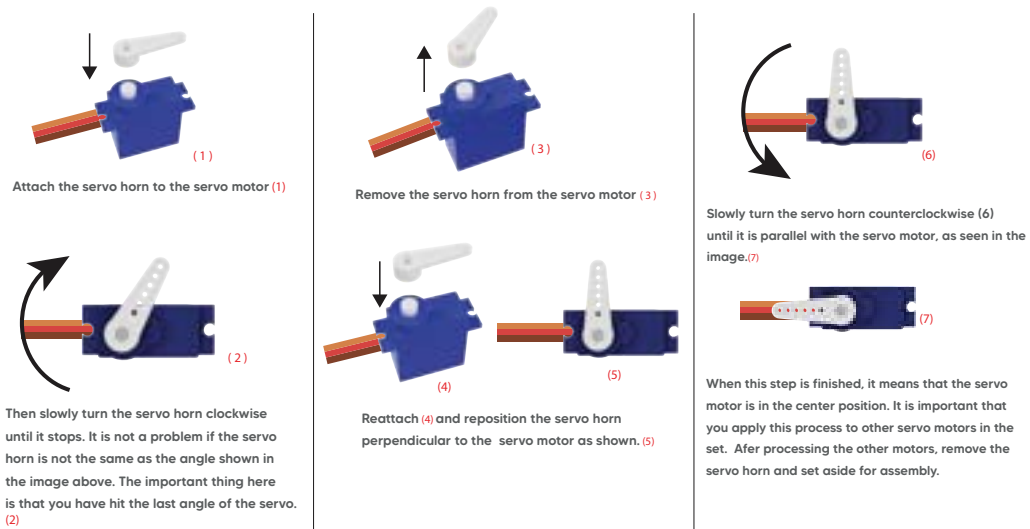
● Project Images:

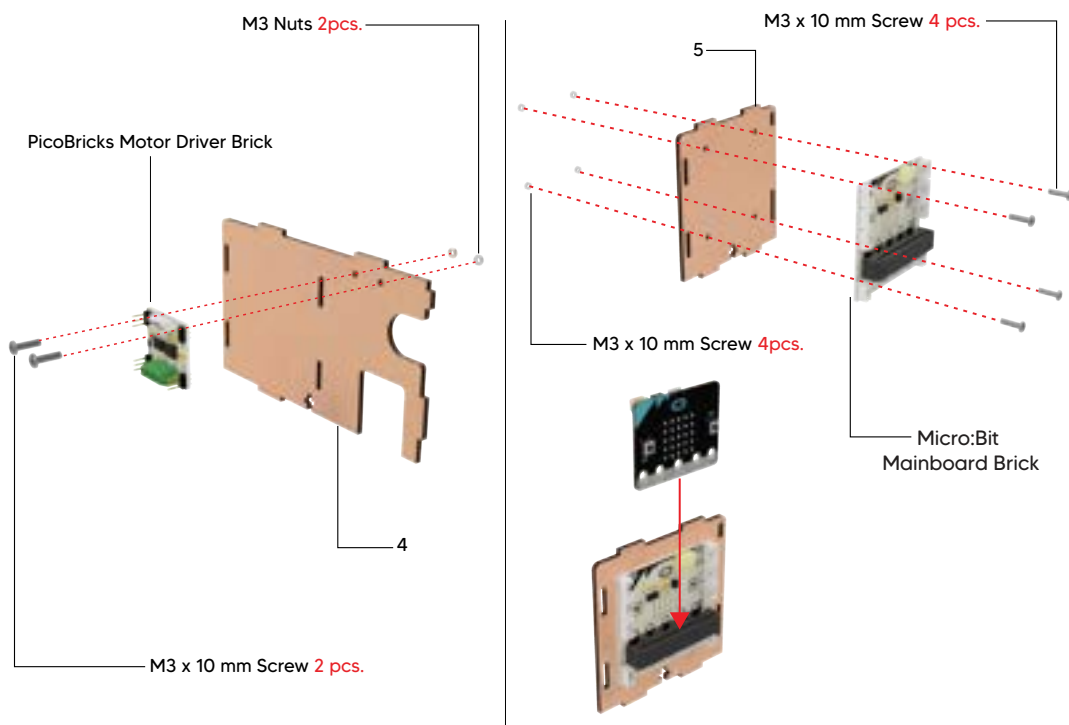
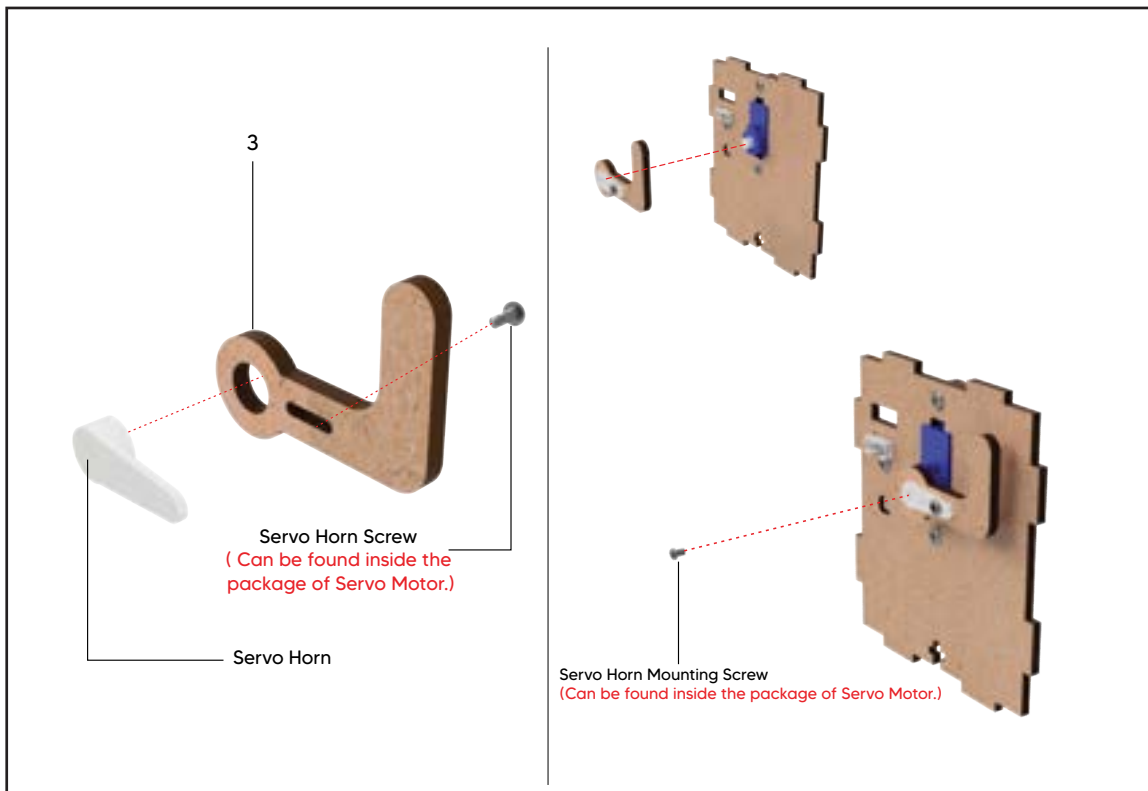


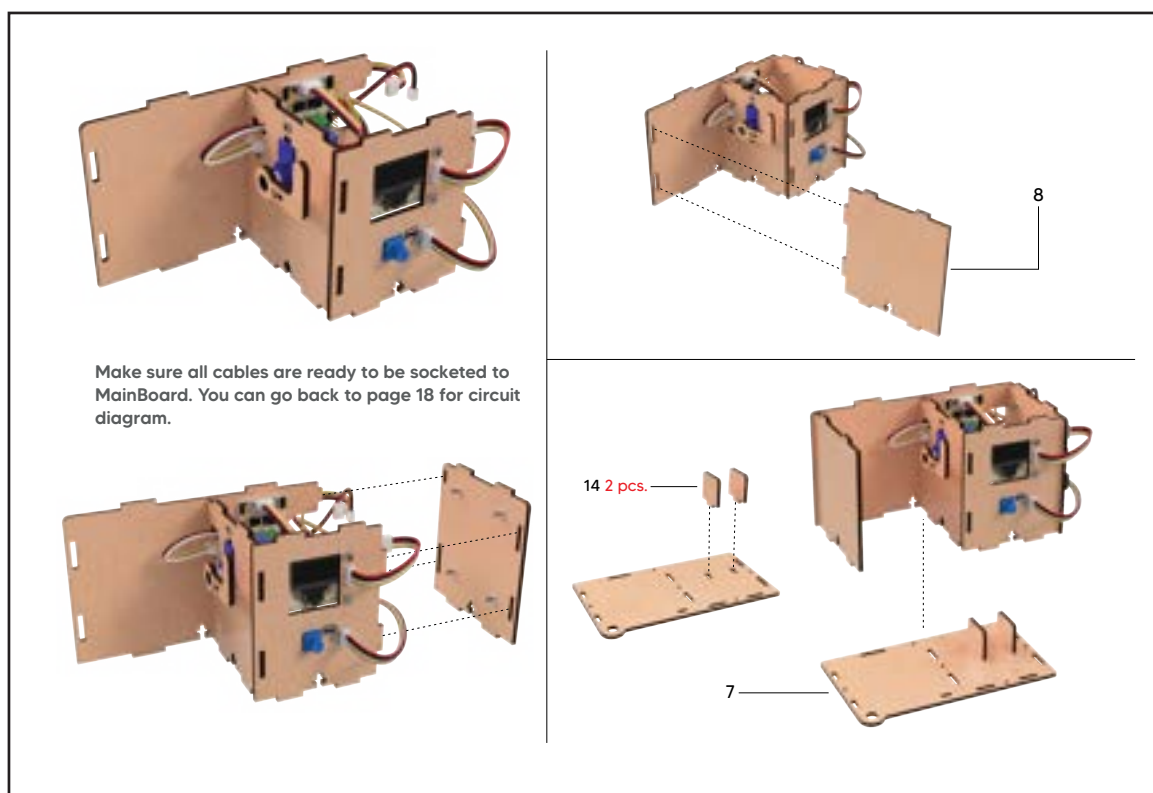
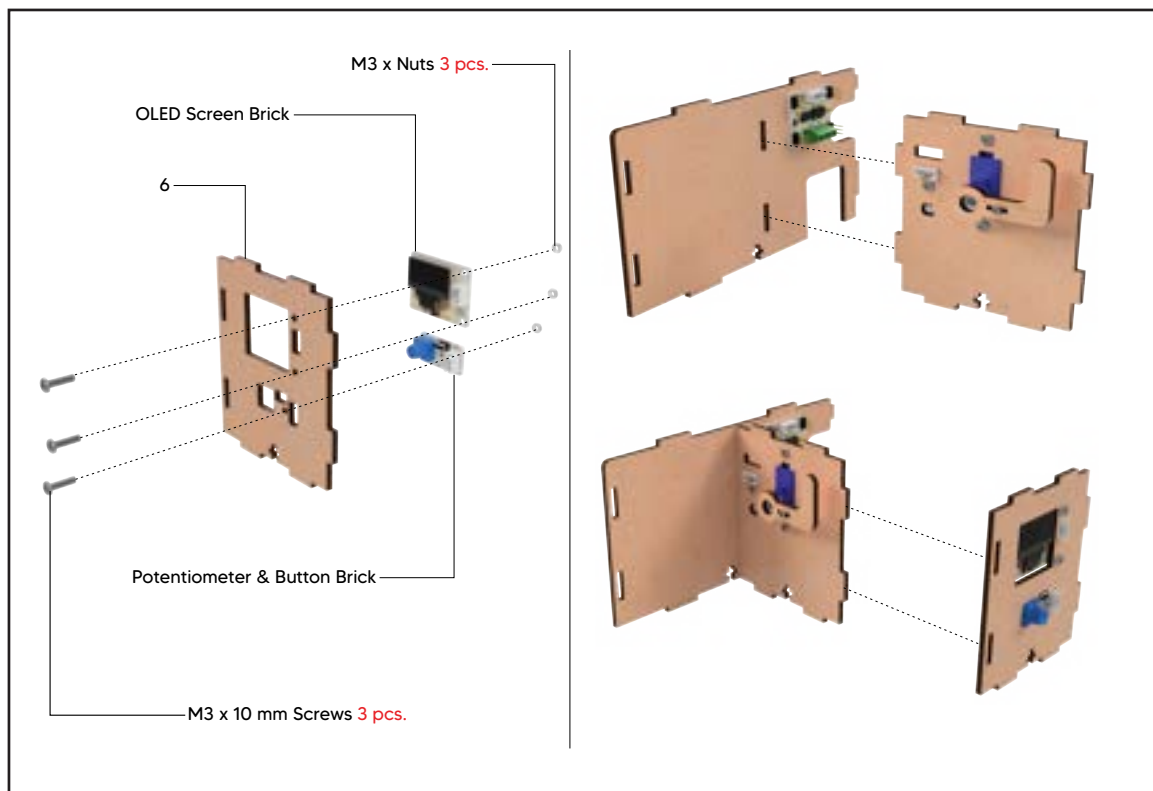
Setup Steps of The Project:

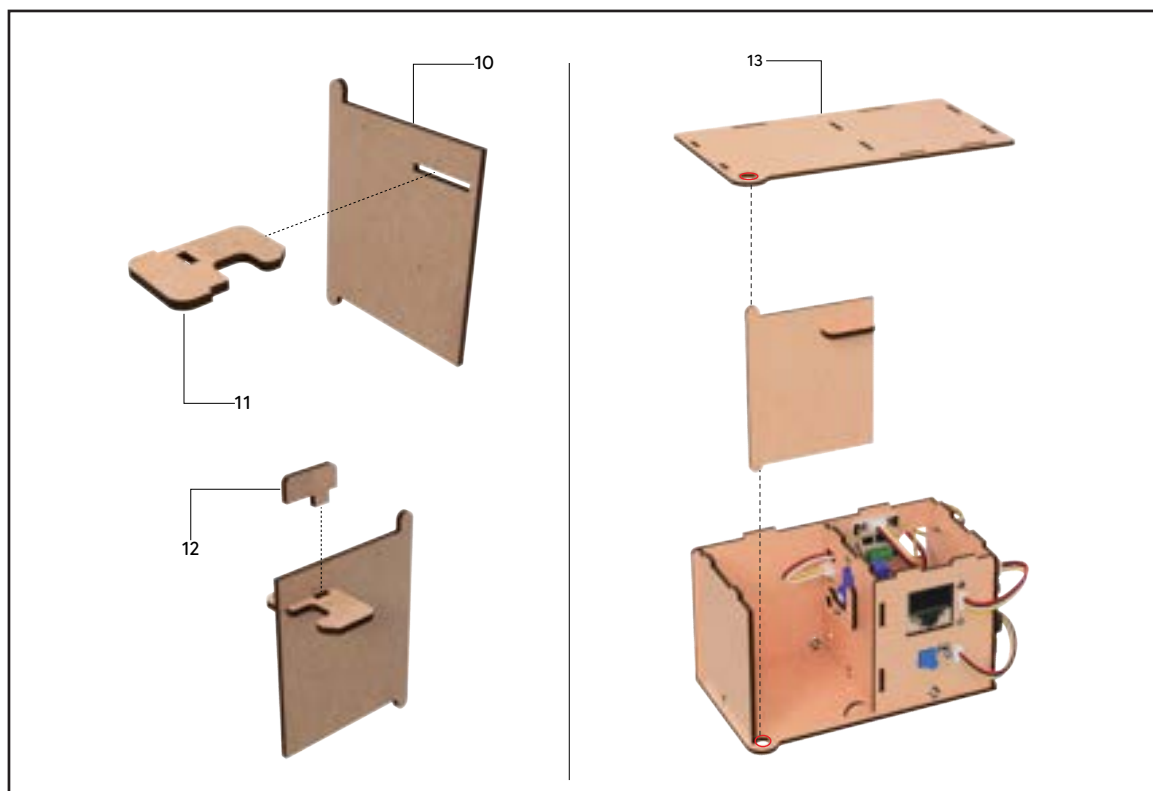
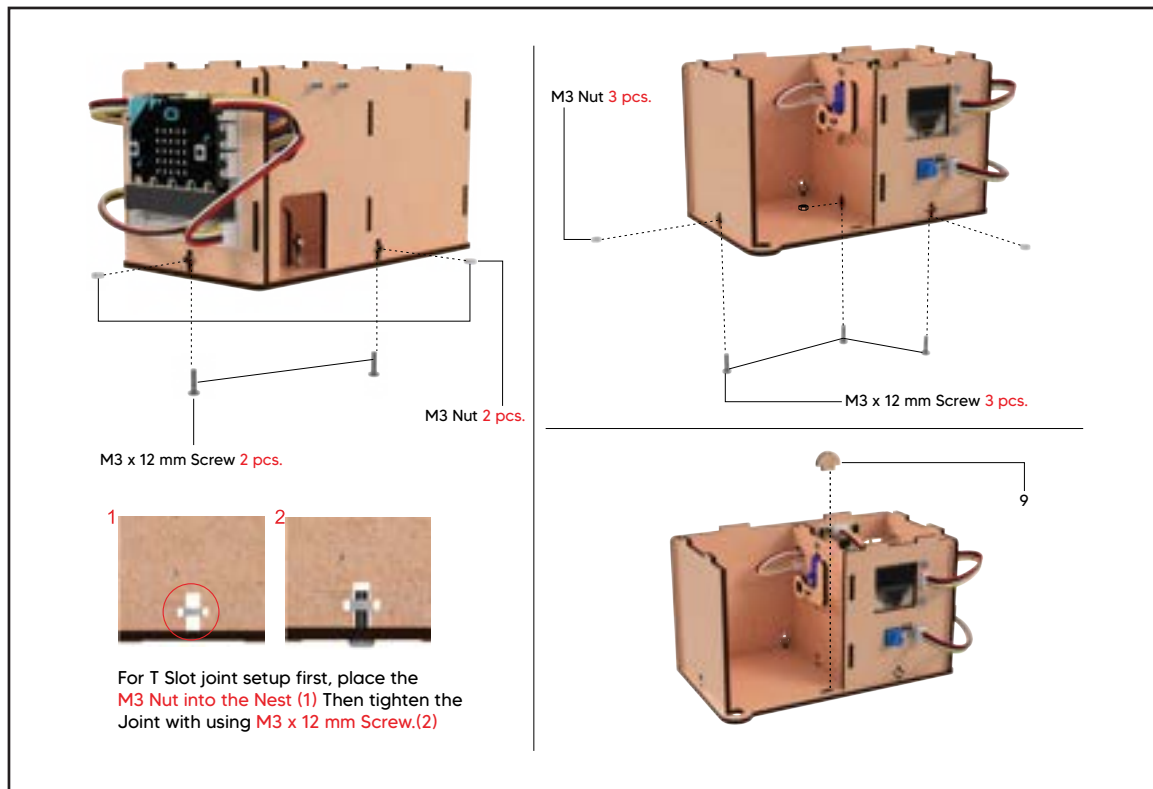
Servo Motor Calibration

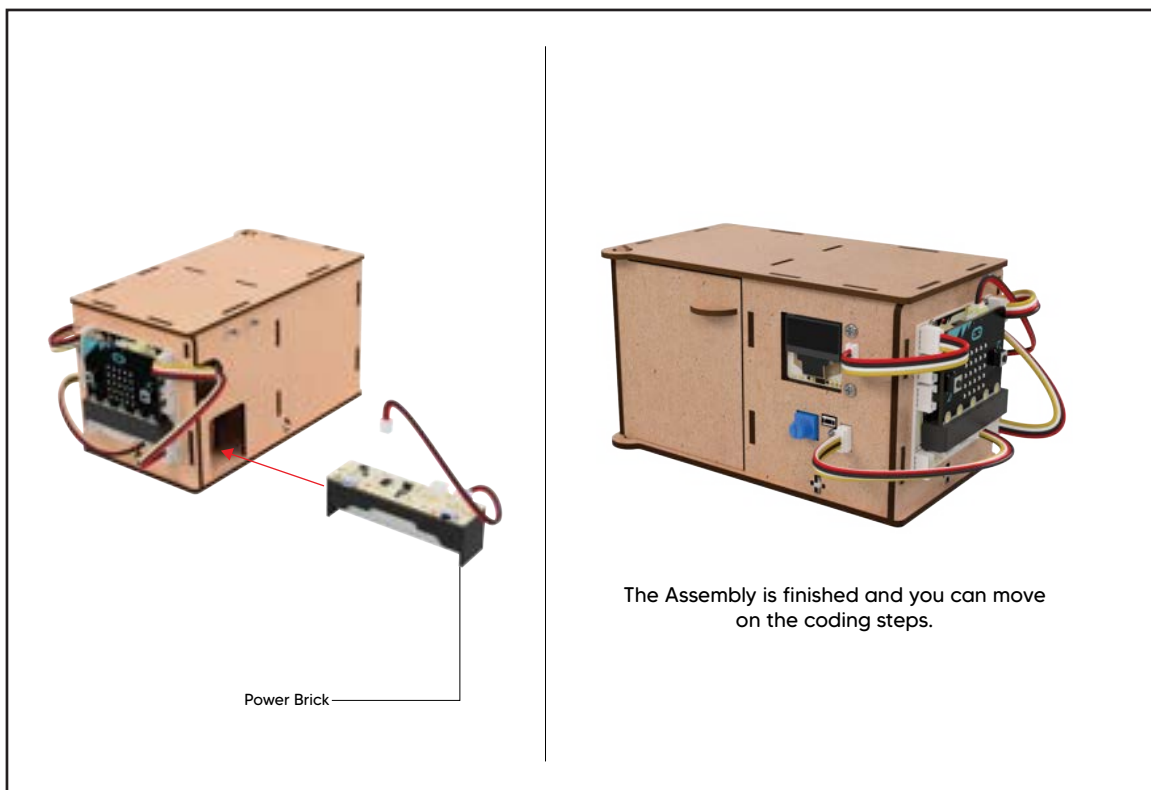
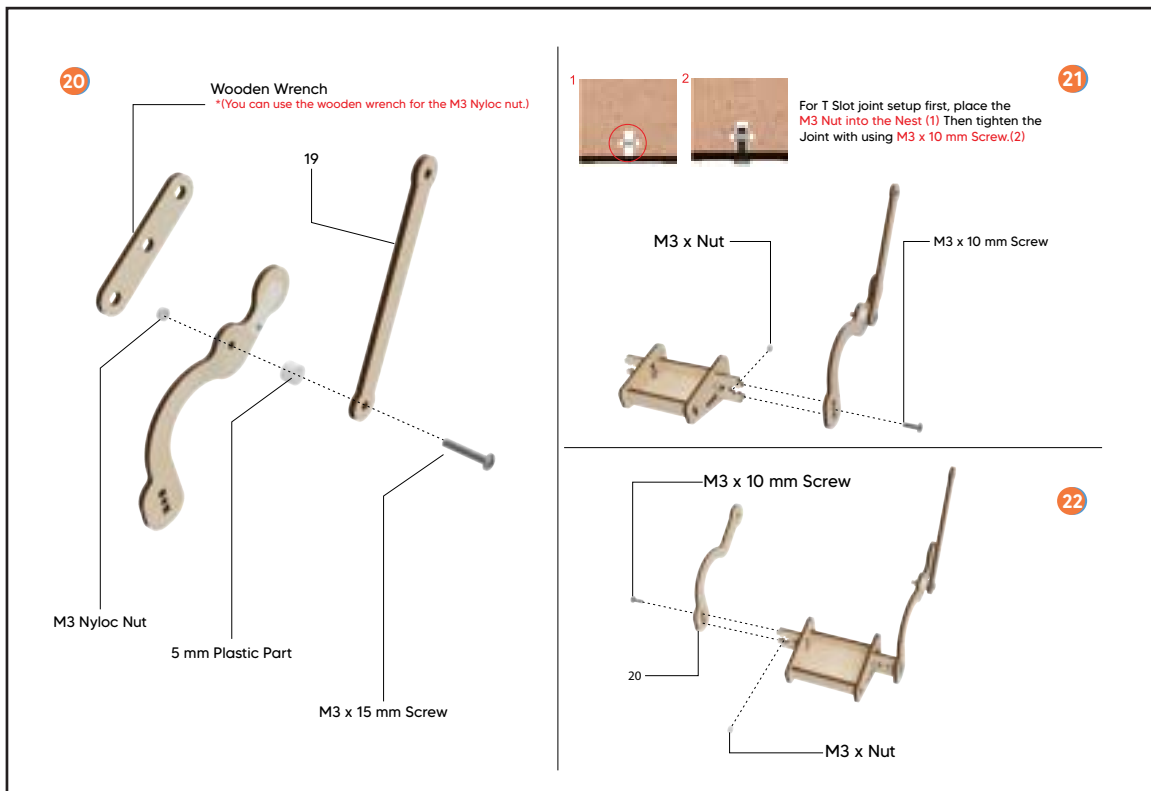
Before starting the assembly, you have to manually calibrate the angles of the servo motors. Otherwise, Servo Motors won't be working properly.











MicroBlocks Code of The Project:

comment
SAFETY BOX:
 micro:bit v2
 A Safety Box is PROTECTED by passcode and a servo controlled box (5V port).
 The Safety Box has a user changeable password code: 1234
OPERATION:
 LDR light sensor controls the box lock mechanism:
 -- Initially the box is OPEN, LDR = high value
 -- User closes the box so LDR = low value
 -- when the LDR value is low (dark), BOX will be locked using the servo.
 -- Lock condition causes the program to enter the cycle of passcode entry.
PASSCODE ENTRY:
 -- the user will enter a 4 digit passcode, using the potentiometer and the Pot button.
 -- Passcode digits are entered using the micro-A and micro-B buttons.
 -- For each digit, user dials the pot value to select the digit value, then presses the POT button to enter it.
 -- When all four digits are entered and the passcode entered matches the internally set code, the box will open.
 -- LED display will show an X for unsuccessful attempts, and a checkmark for success.
NOTE:
 You may need to adjust the LDRTrigger value and Servo open/close angle settings for optimal operation.

```

when started
  _PicoBricks-mb init Library
  set _boxStatus to closed
  set _passCode to 1234
  set _codeEntered to list [ ]
  set _LDRTrigger to 50
  set _done to false
  forever
    if PicoBricks-mb light sensor (0-100) % > LDRTrigger
      set _boxStatus to closed
    else
      convert join items of list _codeEntered to number =
      _passCode
      set _done to true
      set _boxStatus to open
  wait 1000 milliseconds
  
```

```

when _done = true
  comment
    DONE is to control single resolution of
    display message, the don't want to keep
    writing to screen while the box is open.
    WAIT is to enable the visibility of the
    checkmark displayed on passcode match.
  if not _done
    PicoBricks-mb set servo [ ] angle [ ] (0-180)
    wait 500 milliseconds
    clear display
    write Done on TFT at x [ ] y [ ] color [ ] scale [ ] wrap [ ]
    write Wait on TFT at x [ ] y [ ] color [ ] scale [ ] wrap [ ]
    write Check on TFT at x [ ] y [ ] color [ ] scale [ ] wrap [ ]
    _resume monochrome display updates
    set _done to false
  
```

```

when button [A] pressed
  comment
    Selects passcode digit,
    Limits selection to 1-4
  if _boxStatus = true
    change _passCode by [ ]
    _passCode = [ ]
    set _passCode to [ ]
  
```

```

when button [B] pressed
  comment
    Selects passcode digit,
    Limits selection to 1-4
  if _boxStatus = true
    change _passCode by [ ]
    _passCode = [ ]
    set _passCode to [ ]
  
```

```

when _boxStatus = true
  comment on opening is 24 pixels
  PicoBricks-mb set servo [ ] angle [ ] (0-180)
  set _codeEntered to list [ ]
  set _passCode to 1234
  set _done to false
  clear display
  write _passCode on TFT at x [ ] y [ ] color [ ] scale [ ] wrap [ ]
  write _codeEntered on TFT at x [ ] y [ ] color [ ] scale [ ] wrap [ ]
  repeat until _done = true
    getPasscode
    convert join items of list _codeEntered to number =
    _passCode
    write _passCode on TFT at x [ ] y [ ] color [ ] scale [ ] wrap [ ]
    display
    wait 500 milliseconds
  else
    display
    wait 500 milliseconds
  
```

[illegible]

[illegible]

