

maker

guide

maker

guide



Maker Guide

ISBN: 978-9948-39-433-4
Authored by: Mohannad Takrouri
Copyright © 2018 Ibtikar Edu Tech Solutions

All rights reserved. No part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written consent of Ibtikar Edu Tech Solutions, including, but not limited to, network storage or transmission, or broadcast for distance learning, except in the case of brief quotations embedded in critical articles or reviews.

Exclusive rights by Ibtikar Edu Tech Solutions for manufacture and export. This book cannot be re-exported from the country to which it is sold by Ibtikar Edu Tech Solutions.

While the advice and information in this book is believed to be true and accurate at the date of publication, neither the author nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Ibtikar Edu Tech Solutions, and its dealers and distributors will not be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Ibtikar Edu Tech Solutions has endeavored to provide trademark information of the companies and products mentioned in this book by the appropriate use of capitals. However, Ibtikar Edu Tech Solutions cannot guarantee the accuracy of this information.

Contents

Getting Started with the Ibtikar Maker Board 4

Introduction to Microcontrollers.....	4
Programming the Microcontroller.....	6
The Ibtikar Maker	8

Creating Your First Program 28

Ardublockly Interface.....	28
What Are the Different Blocks?.....	30
Create, Save and Load Your Projects	37
Program Your Ibtikar Maker	38

Maker Activities 42

On-Board LED	42
LED Grid	44
Read the Buttons	55
Read the Temperature	59
Read Ambient Light.....	60
Buzzer.....	61
Read the Sound Level	65
Pin Pads.....	66
NeoPixels	69
Triple Axis Accelerometer.....	74



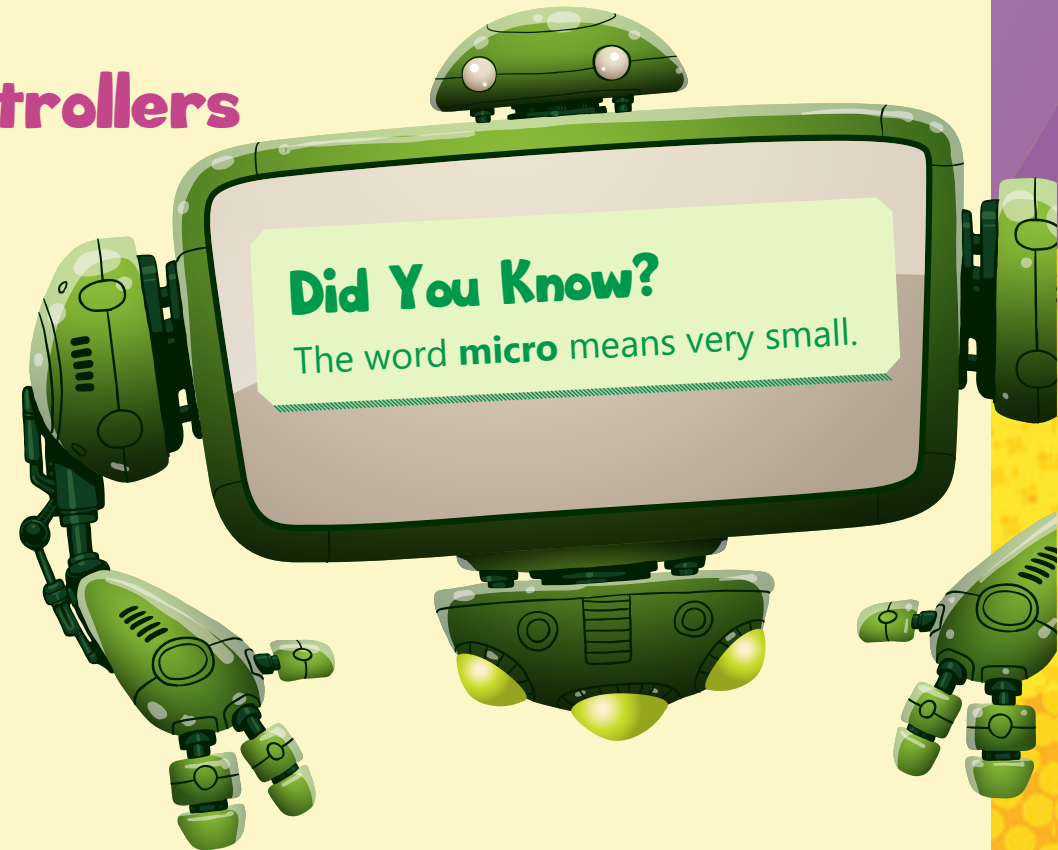
Getting Started with the Ibtikar Maker Board

Introduction to Microcontrollers

Definition

A **microcontroller** is a small computer on a single chip. Unlike your personal computer which can do different tasks at the same time, the microcontroller can only do one task at a time.

A microcontroller is usually **embedded** inside a system. You can think of it as the brain of a system.

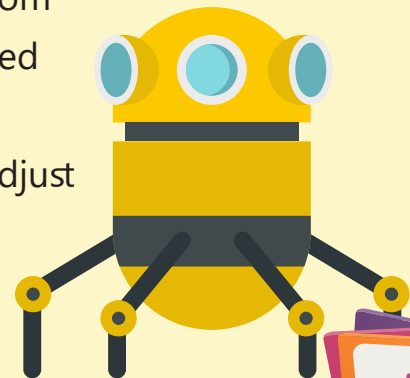


If the system has an input unit for sensing the environment, a control unit for processing the received signals and an output unit for sending out information or to control an output device, then it can be called an **embedded system**.

Where Do You Find Embedded Systems?

Embedded systems control many devices in common use today. You can find microcontrollers in washing machines, microwave ovens, cars, elevators and any smart machine.

An example of an embedded system, is the air conditioner unit that you use in your home and in your car. The air conditioner cools down the air by removing its heat. The main controller of this unit is an embedded computer system that senses the temperature of the room or car, compares it with the desired temperature you choose, then controls the cooling process to adjust the temperature.



Programming the Microcontroller

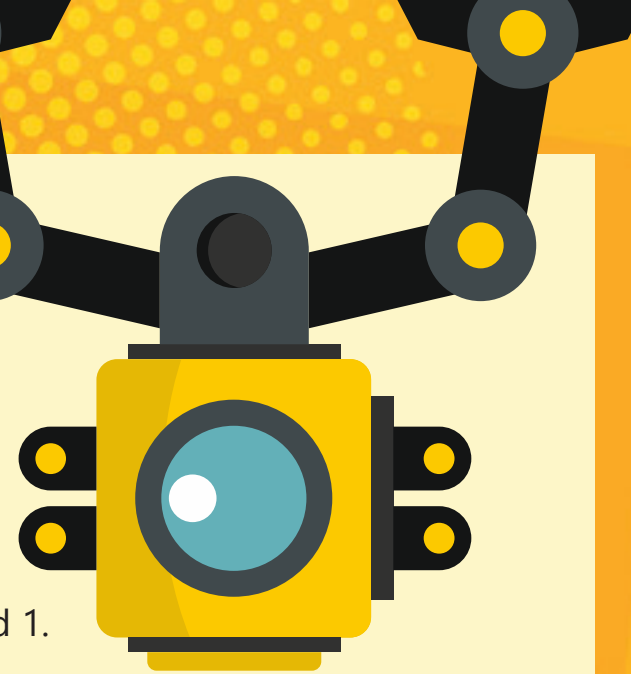
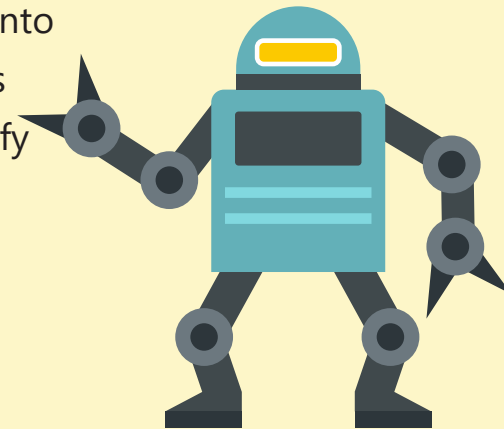
What Is a Program?

A **program** is a set of commands and instructions that can be downloaded to a computer or a microcontroller to do a specific task.

Computers have their own language which is based on two numbers, 0 and 1.

It is very difficult for us as humans

to write in this language. Because of this, programmers have created high level languages which allow us to write computer programs in a language that is like the language we understand. A **compiler** is then used to convert the program into the computer language. This makes it easy for us to modify and understand the programs that are written.



Types of Programming

Microcontrollers can be programmed using either a visual programming interface or a text-based one. In visual programming, you can use graphical elements to create programs. You can also drag and drop program elements, click, use menus, forms, dialogue boxes and so on. Behind each block of your program, there are tens or even hundreds of lines of code. This type of programming helps new users to easily understand programming.

The other type of programming is text-based. In text-based

programming, you must understand the language syntax and rules. You can cut, copy, and paste your code which gives you more flexibility compared to dragging and dropping one block at a time.

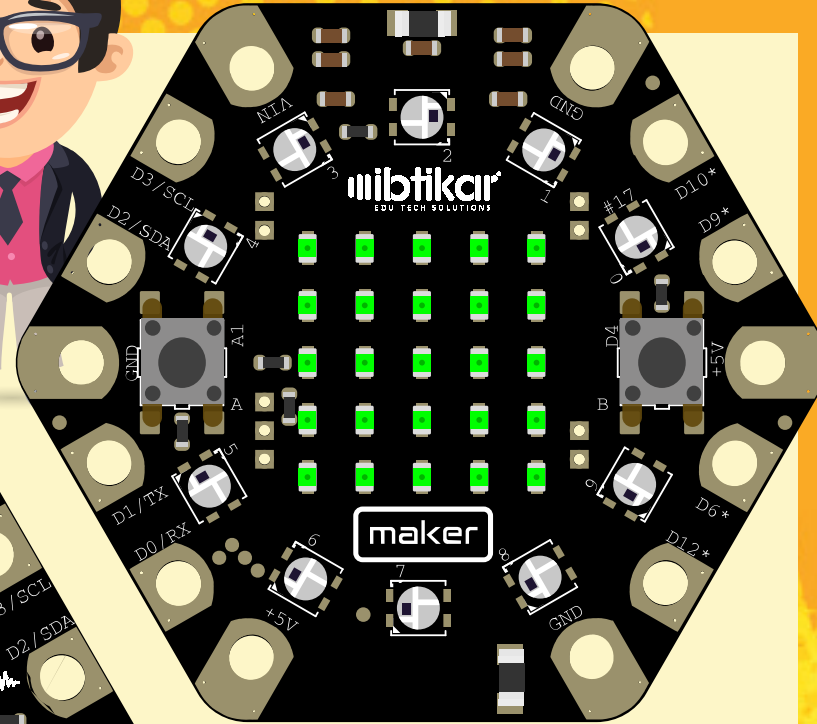
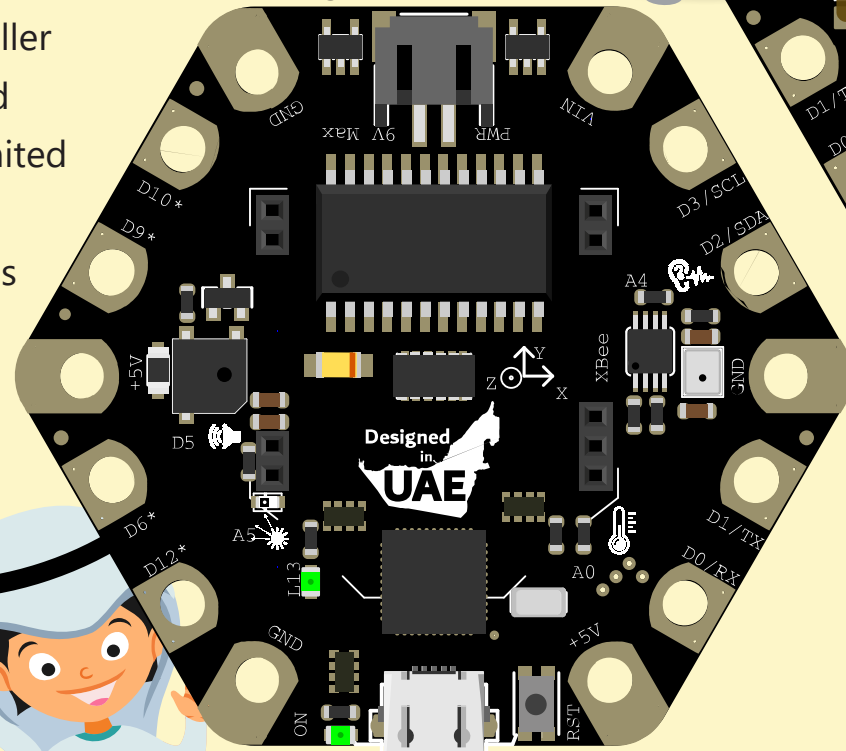


The Ibtikar Maker

The Hardware

Size

The Ibtikar Maker board is a small, hexagon-shaped microcontroller board, designed and developed in the United Arab Emirates. The following two figures show the front and the back of the board.



Features

The Ibtikar Maker board has the following features:

- ▶ 25 LEDs in grid formation
- ▶ 10 NeoPixel RGB LEDs
- ▶ 2 push buttons (left and right)
- ▶ temperature sensor
- ▶ ambient light sensor
- ▶ sound sensor
- ▶ mini speaker (magnetic buzzer)
- ▶ triple axis accelerometer
- ▶ 8 input/output pins which 7 of them can act as capacitive touch inputs
- ▶ XBEE Socket to allow for Wi-Fi or Bluetooth expansion
- ▶ Arduino compatible microcontroller

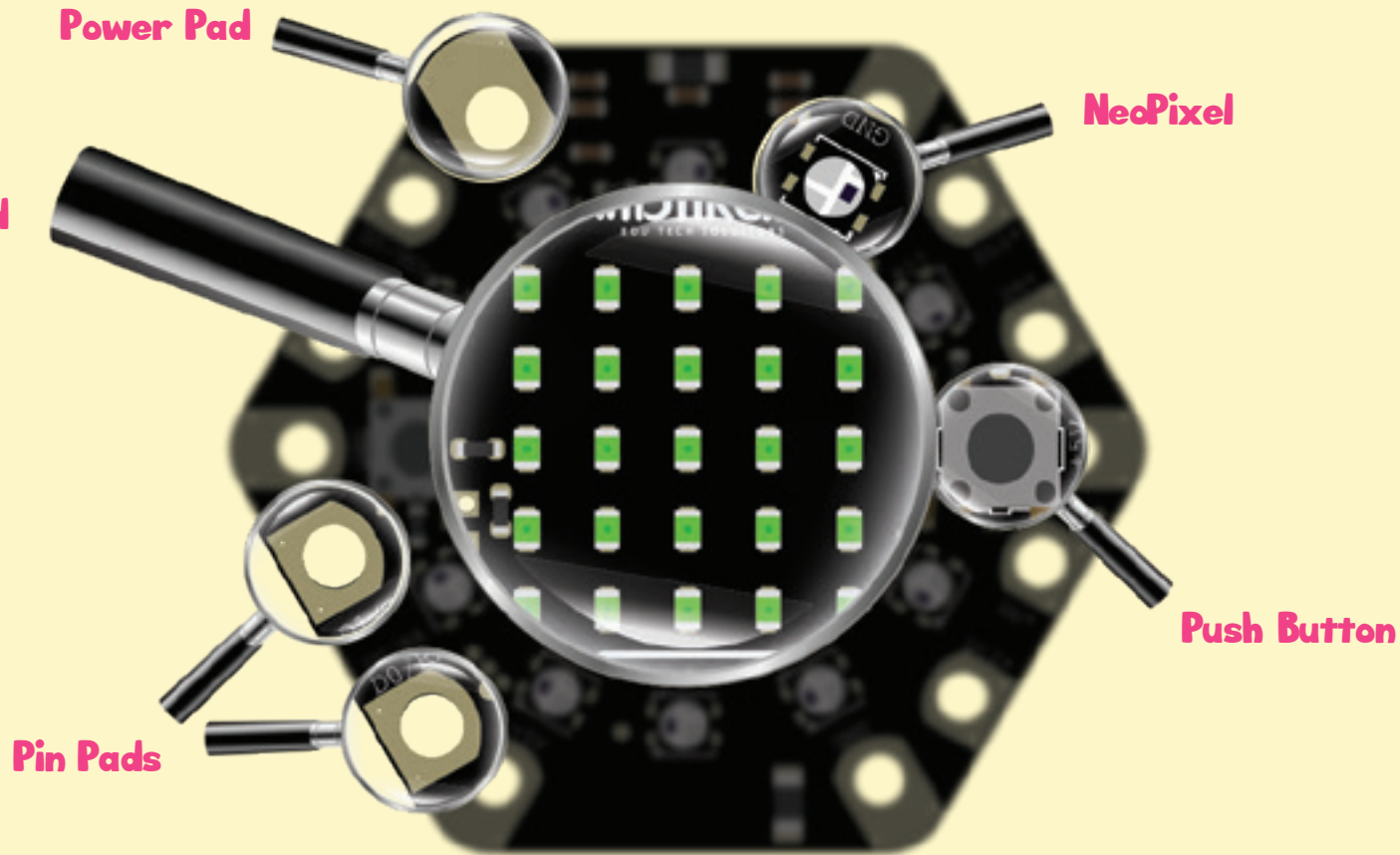




Front and Back

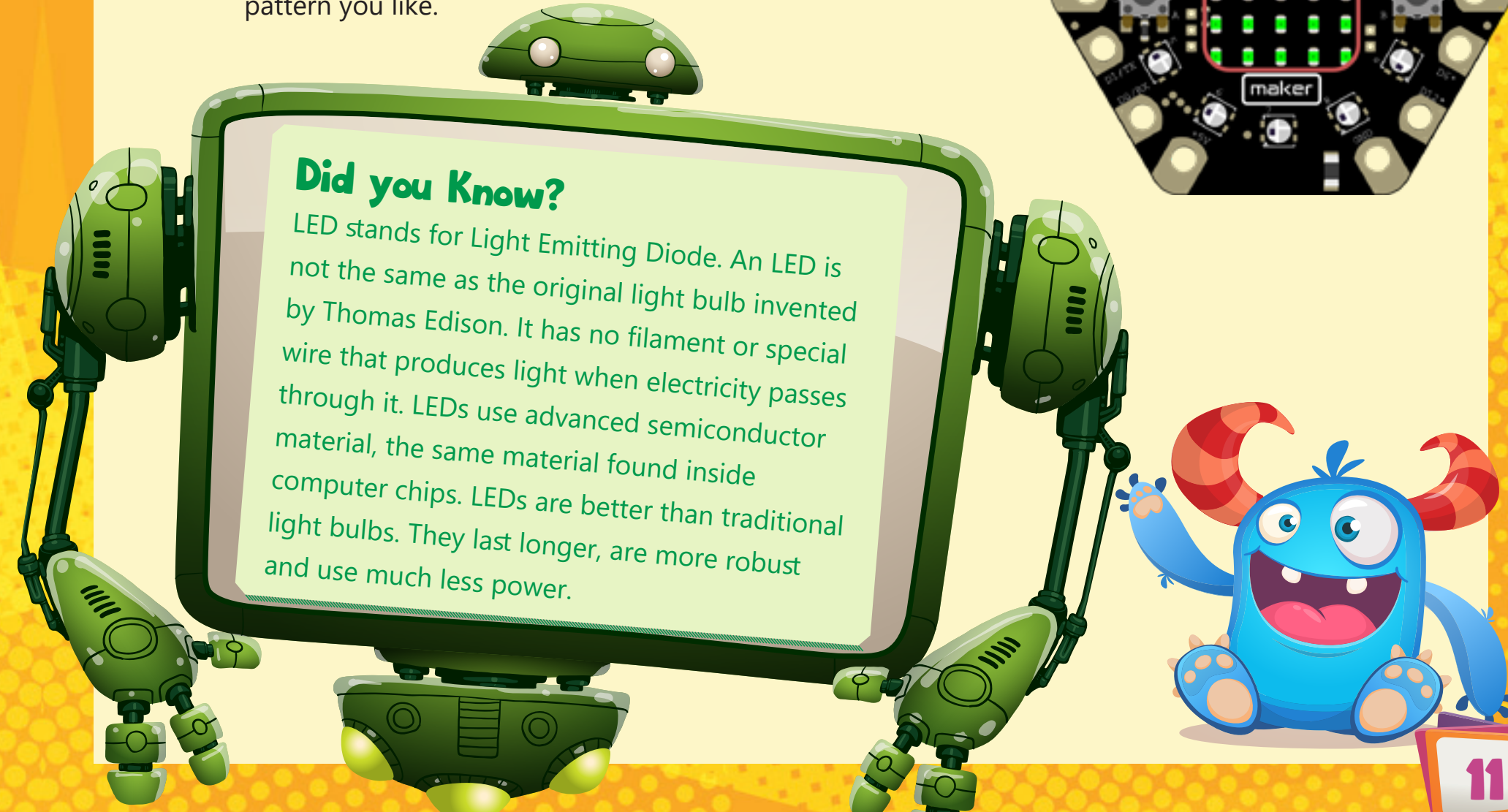
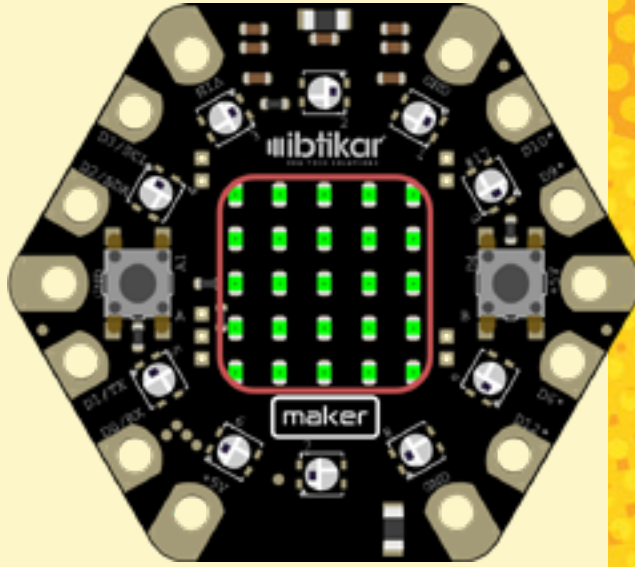
Before you start using the Ibtikar Maker board, it is important for you to know where each component is, and its purpose.

On the front side, there are the following components:



1. LED Grid

There are 25 LEDs which you can turn ON or OFF. Each LED can be controlled individually, which allows you to create patterns. For example, you can show letters, numbers, text, emojis or any other pattern you like.

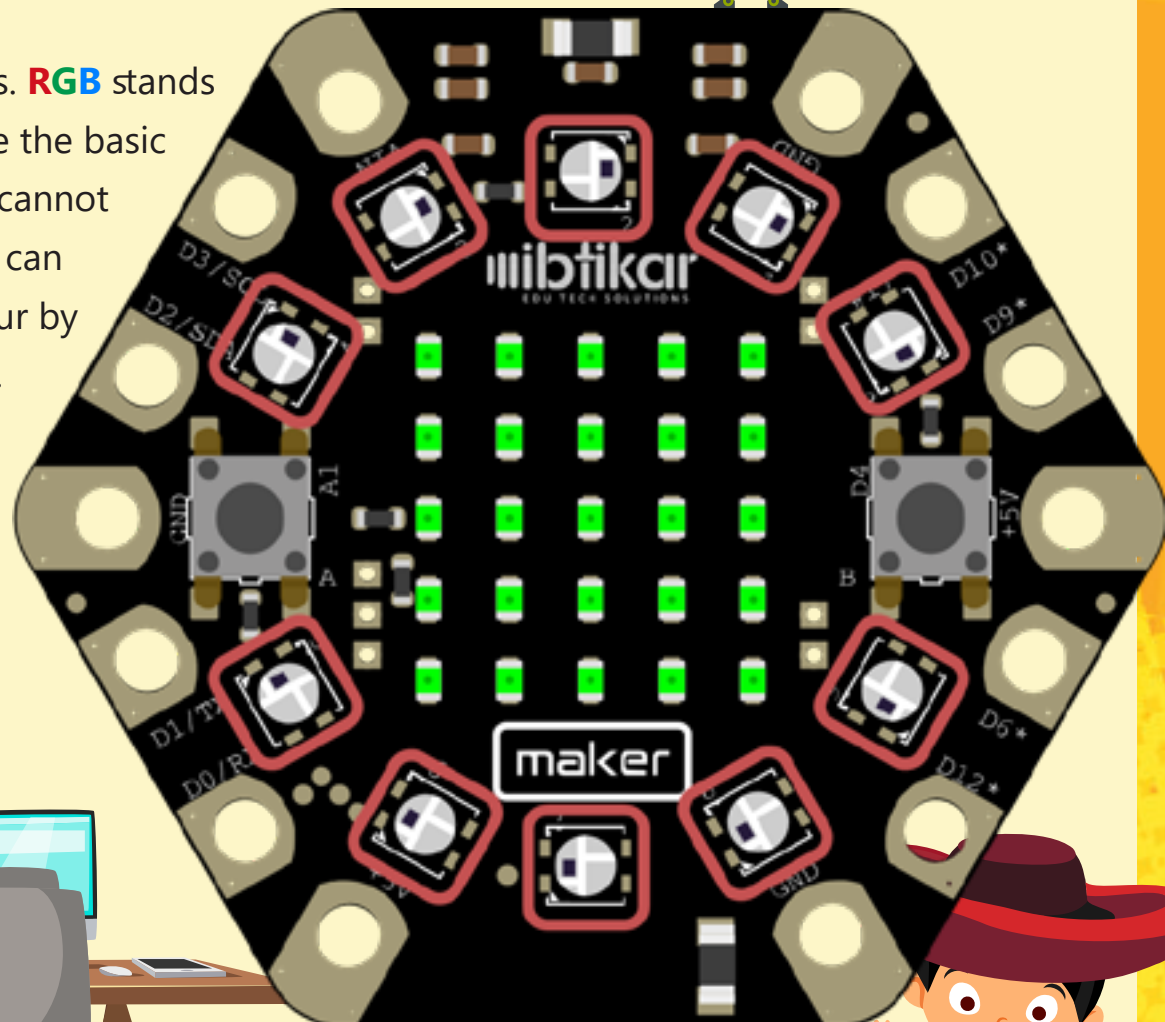


Did you Know?

LED stands for Light Emitting Diode. An LED is not the same as the original light bulb invented by Thomas Edison. It has no filament or special wire that produces light when electricity passes through it. LEDs use advanced semiconductor material, the same material found inside computer chips. LEDs are better than traditional light bulbs. They last longer, are more robust and use much less power.

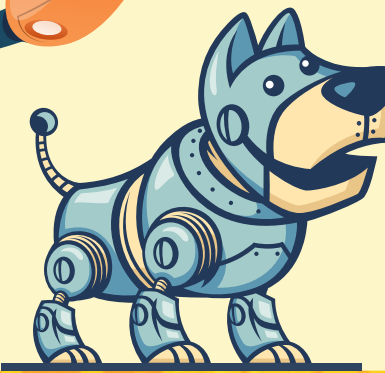
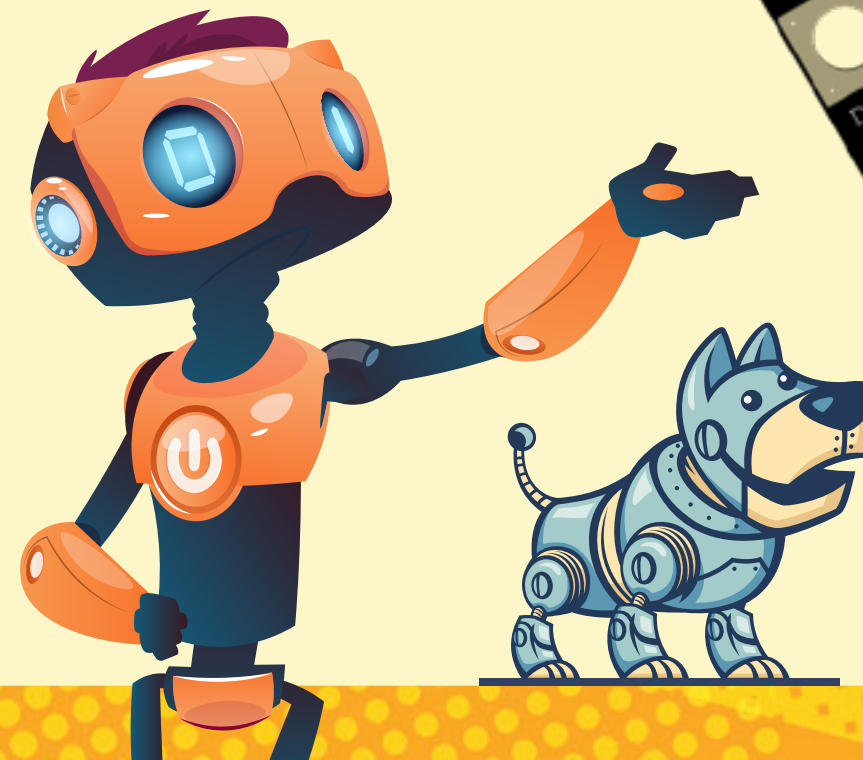
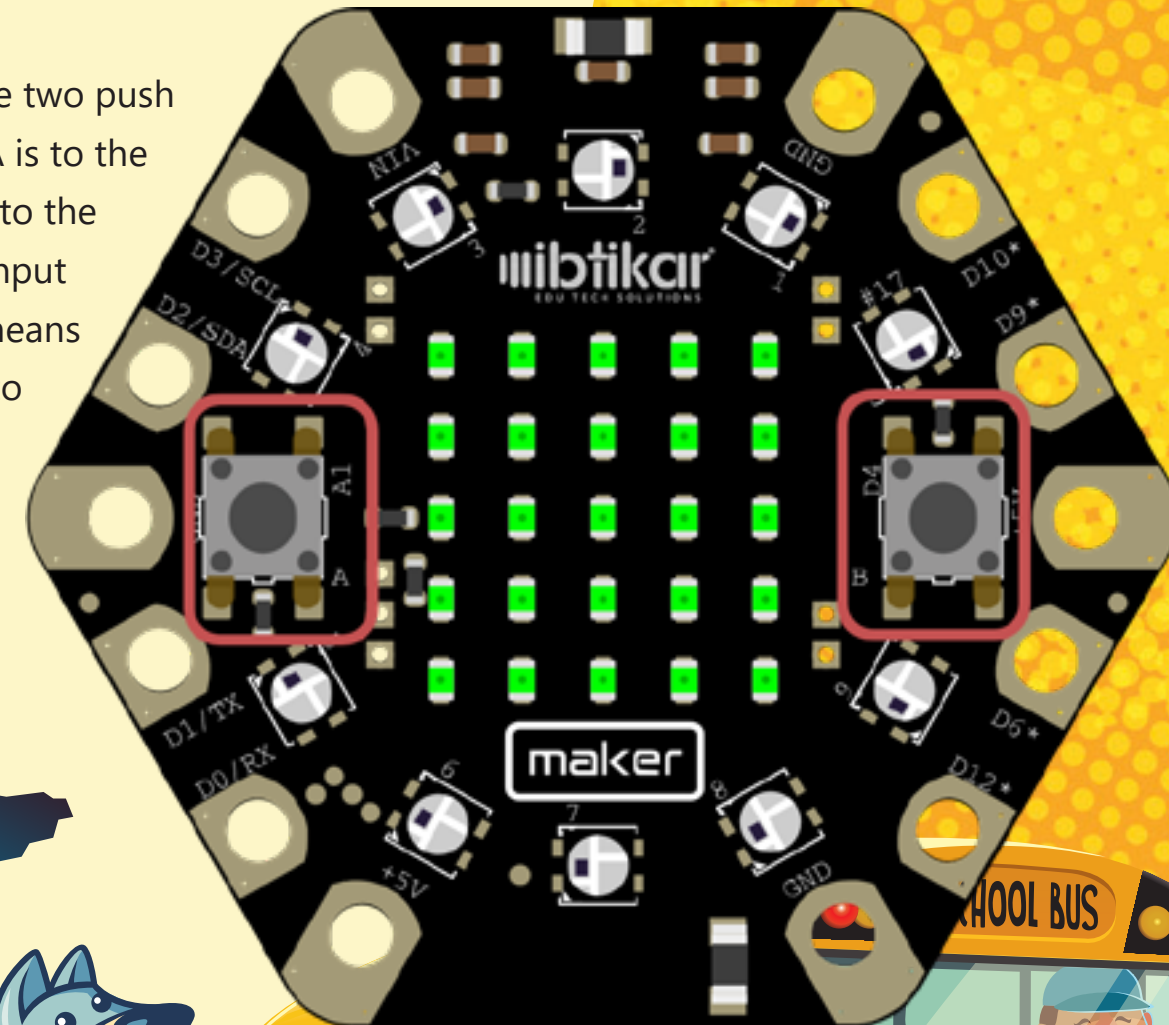
2. NeoPixels

The Ibtikar Maker has 10 RGB LEDs. **RGB** stands for **Red**, **Green** and **Blue** which are the basic colours. Unlike the 25 LEDs which cannot change their colour, the RGB LEDs can be programmed to show any colour by combining the three basic colours. Think of each pixel as three small LEDs combined, with each of these LEDs being a different colour (**Red**, **Green** and **Blue**).

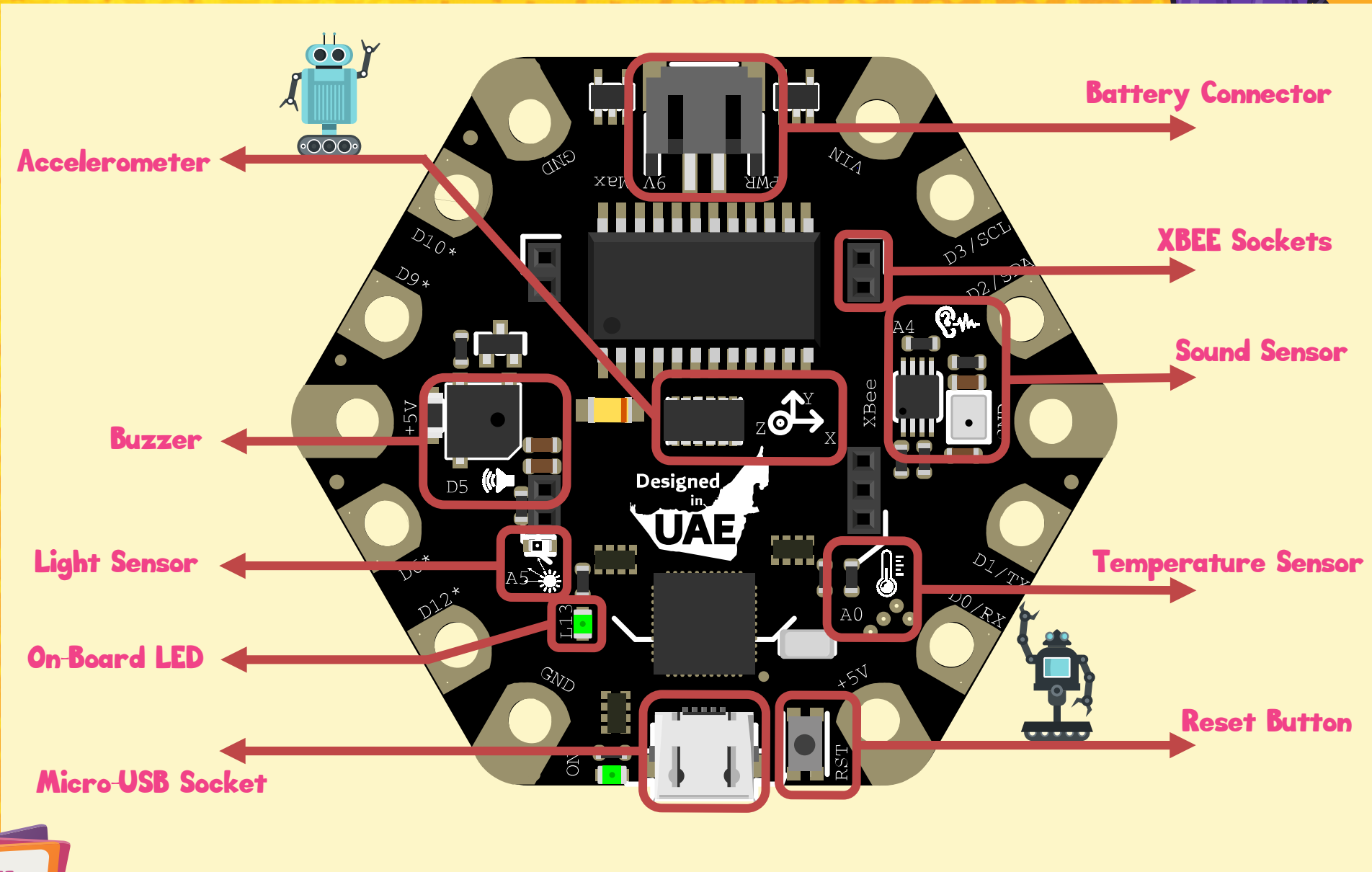


3. Push Buttons

On the front of the board, there are two push buttons labelled, A and B. Button A is to the left of the board, while button B is to the right. These two buttons are user input components on your board. This means that you can program your board to detect when they are pressed or released. Pressed means 1 and released means 0.



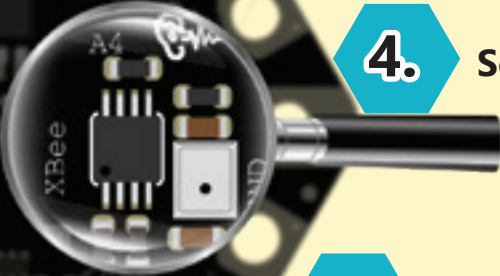
On the back side of the board, there are the following components:



1. Temperature Sensor
This is an analog sensor which measures the board temperature.

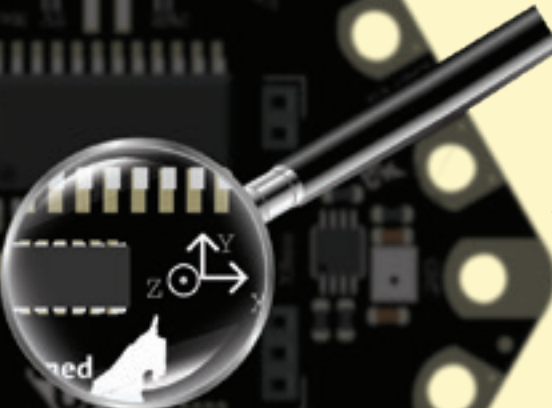
2. Light Sensor
This is an analog sensor which measures the ambient light in the environment. The analog light sensor gives values between 0 and 1023.

3. Buzzer (Mini Speaker)
This is a magnetic buzzer which can play tones. By controlling the frequency going to the buzzer, you can generate different tones.



4. Sound Sensor

This is an analog sound level sensor. You can use it to detect if there is a clap/sound near the board.



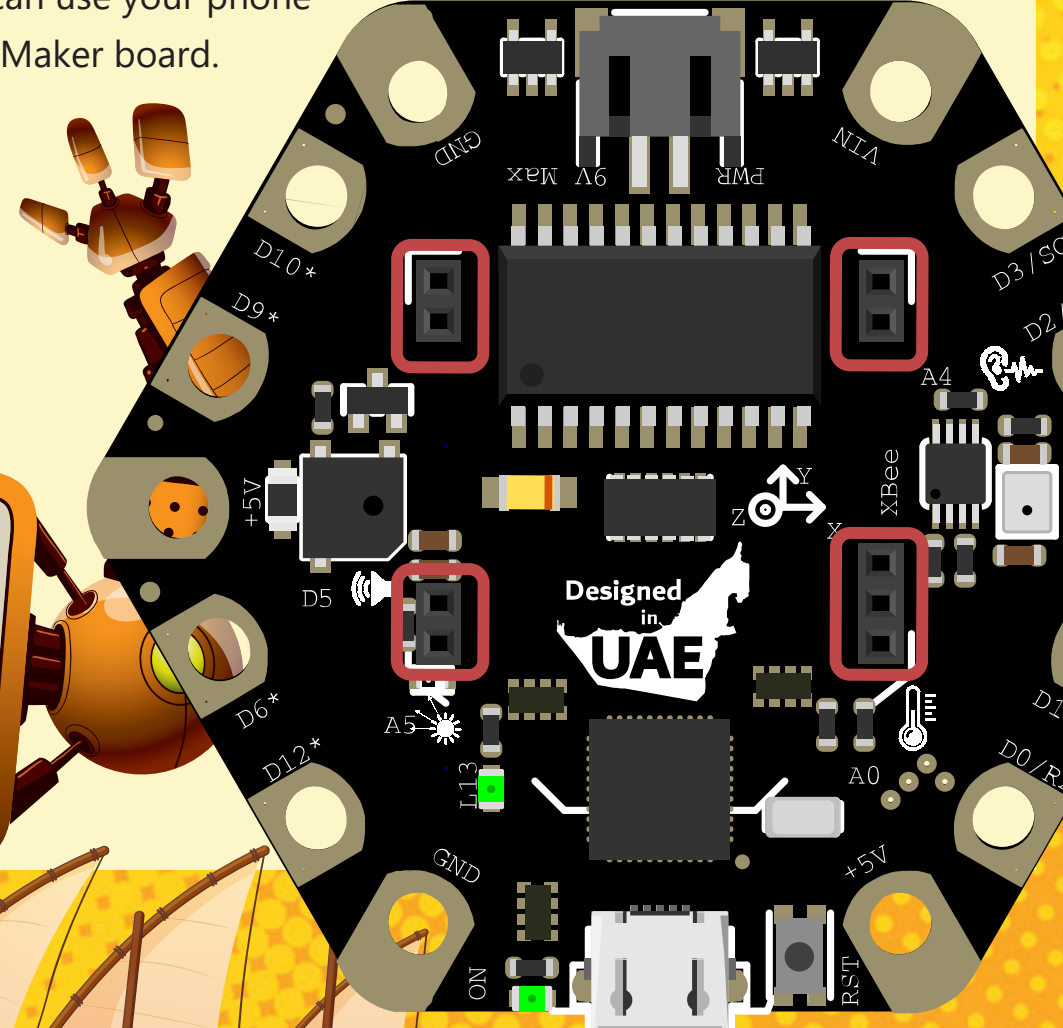
5. Triple-Axis Accelerometer

This sensor is in the middle of the board. Accelerometers are used to measure acceleration, which is how fast something is speeding up or down. An accelerometer can measure static acceleration like gravity, which is useful in detecting tilt, like when your phone tilts.

Accelerometer can also detect dynamic acceleration: the sudden start or stop of an acceleration.

Note

The Ibtikar Maker accelerometer can sense the three axes (X, Y and Z). You can identify the positive and negative directions of each axis by looking at the small three-axis drawing next to the sensor.



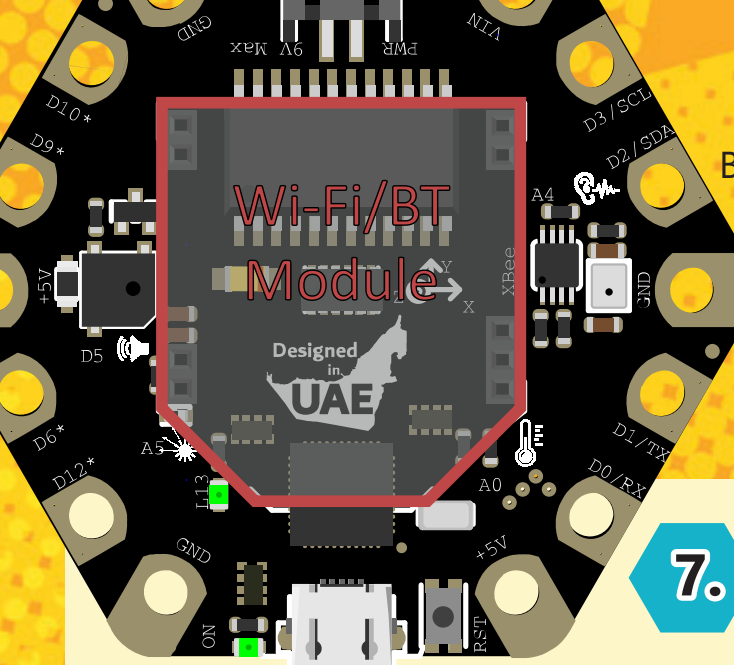
6. XBEE Sockets

These sockets allow for Wi-Fi or Bluetooth expansion. This is useful if you want to wirelessly control your Maker or read from it. You can use your phone or tablet, for example, to connect to the Maker board.



Note

The Wi-Fi/Bluetooth module is not included in the kit and can be purchased separately.



Beside each socket there are some fine, white lines drawn on the board. These lines help you to connect the Wi-Fi/Bluetooth module in the right orientation, as shown.

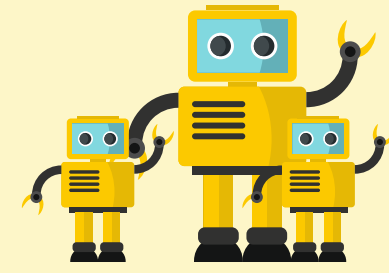
7. On-Board LED

This LED can be controlled by the user and is connected to pin 13.



8. Power LED

This LED turns ON when your Maker is powered.



11. Battery Connector

This connector serves as an external power supply source. You use it to power your board when the USB cable is not connected. This can be extremely useful when you do not want your project to be attached to a computer all the time.



9. Micro-USB Socket

This socket allows you to connect to your computer using a USB cable. You can then program your Maker board, send and receive data and power the board.

10. Reset Button

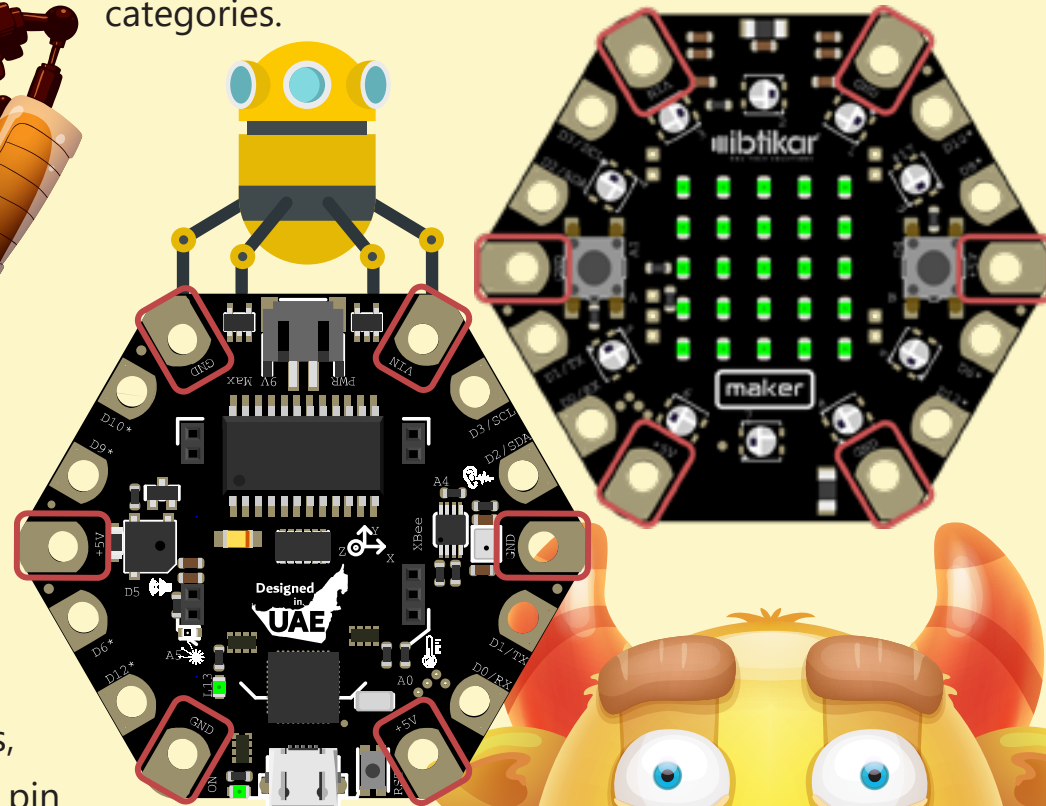
This button restarts or resets the board.



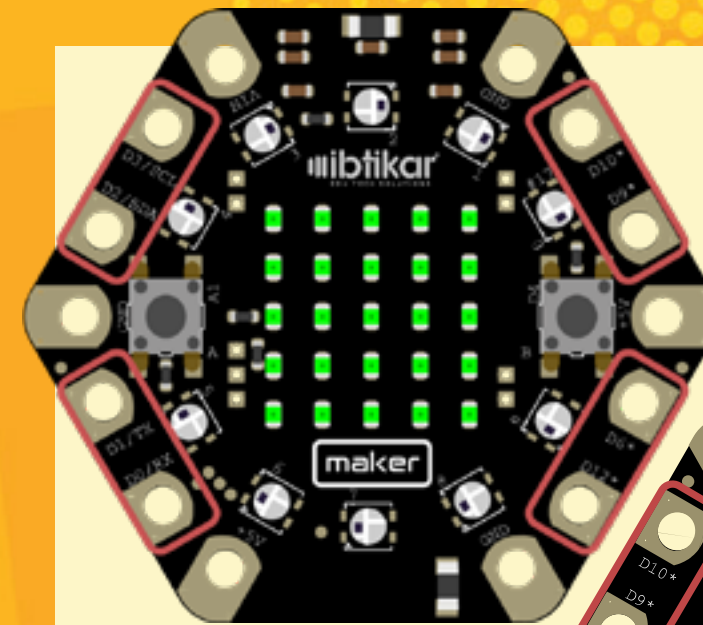
Note

The supply voltage should be between 6V to 9V. Supplying the board with a voltage different to that recommended, may cause the board to malfunction or even damage it.

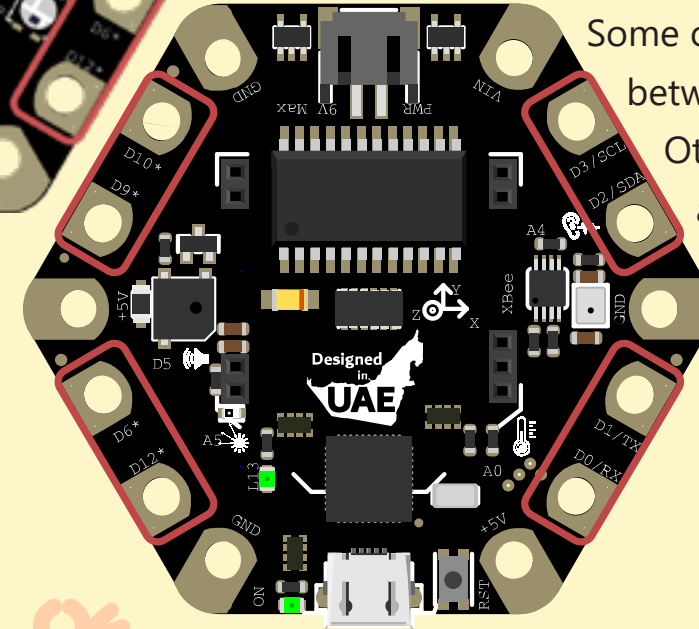
In addition to the previous listed features, there are also 14 pads on the outer edge of the Maker board. These pads can be accessed either from the front or the back side of the board and are divided into two categories.



Power pads: These pads can supply power to external input or output modules. There are three **ground** pins, two **5V** pins and one **Vin** pin. The **Vin** pin has the same voltage as the one coming from the external battery connector.



Pin pads: These pads can be interfaced with extra input and output modules allowing you to extend the capabilities of your Maker board. You can do this by adding another push button for example, or even a module like a motion or a flame sensor, both of which are not on the board.



Some of the pins allow you to communicate between other boards or components. Others allow you to read and write analog and digital signals. These pins are explained in detail in the next section.

Pin Configuration

The Maker board has the following pins. Some of these pins are external (i.e. on the outer edge of the Maker) and some are internal, like the two buttons. It is important to know what each pin is connected to or capable of, to give you full control of your Maker board.

Pin	Function
D0/RX	Free pin, digital input/output, serial communication (receive)
D1/TX	Free pin, digital input/output, serial communication (transmit)
D2/SDA	Free pin, digital input/output, I2C communication
D3/SCL	Free pin, digital input/output, I2C communication, PWM
D4	Button A
D5	Buzzer
D6*	Free pin, digital input/output, analog input, PWM
D7	Accelerometer interrupt (currently not used)
D8	Accelerometer chip select
D9*	Free pin, digital input/output, analog input, PWM
D10*	Free pin, digital input/output, analog input, PWM
D12*	Free pin, digital input/output, analog input
D13	Built-in LED
D17	NeoPixels
A0	Temperature sensor
A1	Button B
A4	Sound sensor
A5	Light sensor

Note

D11, A2, A3 are not exposed on the board.

The Software

Arduino IDE

The Arduino Integrated Development Environment (Arduino IDE) allows you to write and upload codes to your Ibtikar Maker, using a text editor. This open source interface contains a message area, a toolbar with buttons to verify and upload your code and other common functions like create and save files. You can cut, copy, and paste your code.





Ardublockly



In visual programming, you can use graphical elements to create programs. You can also drag and drop program elements, click, use menus, forms, dialogue boxes and so on. Behind each block of your program, there are tens or even hundreds of lines of code. This type of programming helps new users to easily understand programming.

Ardublockly

Ardublockly is a visual programming editor for Arduino. This interface is based on Google's Blockly.



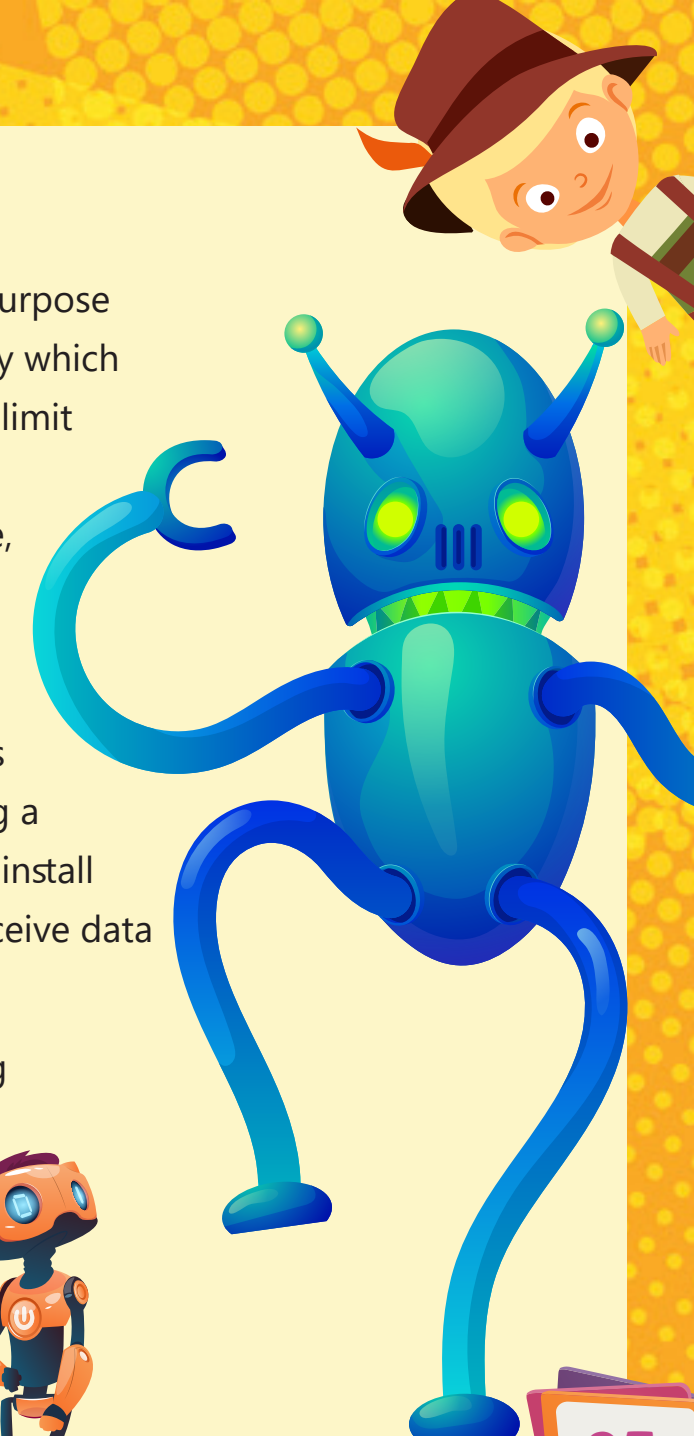
Python

Python is a widely used, high-level programming language for general-purpose programming, which was first released in 1991. It has a design philosophy which emphasises code readability (notably using whitespace indentation to delimit code blocks, rather than curly braces or keywords), and a syntax (a set of rules) which allows programmers to show concepts in fewer lines of code, compared to other languages.

To install the latest version, you can visit the Python official website.

Python can be used to connect serially to your Maker board which allows you to use all the capabilities of Python installed on your computer, using a Python library called PySerial. Unlike Arduino IDE and Ardublockly which install the code, you write to Maker directly; PySerial allows you to send and receive data between your computer and the Maker.

In this manual, we will only be focusing on the Ardublockly programming environment.



Powering the Ibtikar Maker Board

USB connector

The micro USB cable allows you to connect your Maker to the computer, which powers the board and allows you to send and receive data. The USB cable looks like the following:



External Power Supply

The other power option is to use an external supply between 5V to 9V. In your kit, you have a 9V battery holder.

The battery holder has an ON/OFF switch to make it easy to power the Maker, without the need to connect and disconnect the battery each time.



What Is in the Box?

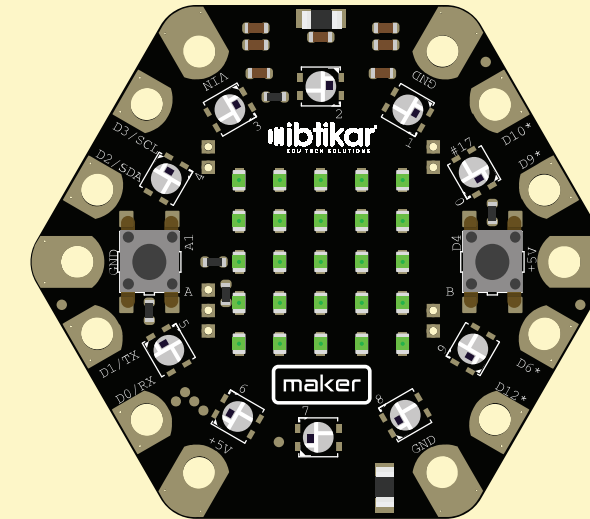
When you open the box, you will find the following:



▶ Micro USB cable



▶ 9V battery holder



▶ Ibtikar Maker board



▶ 8 crocodile cables (4 male, 4 female)

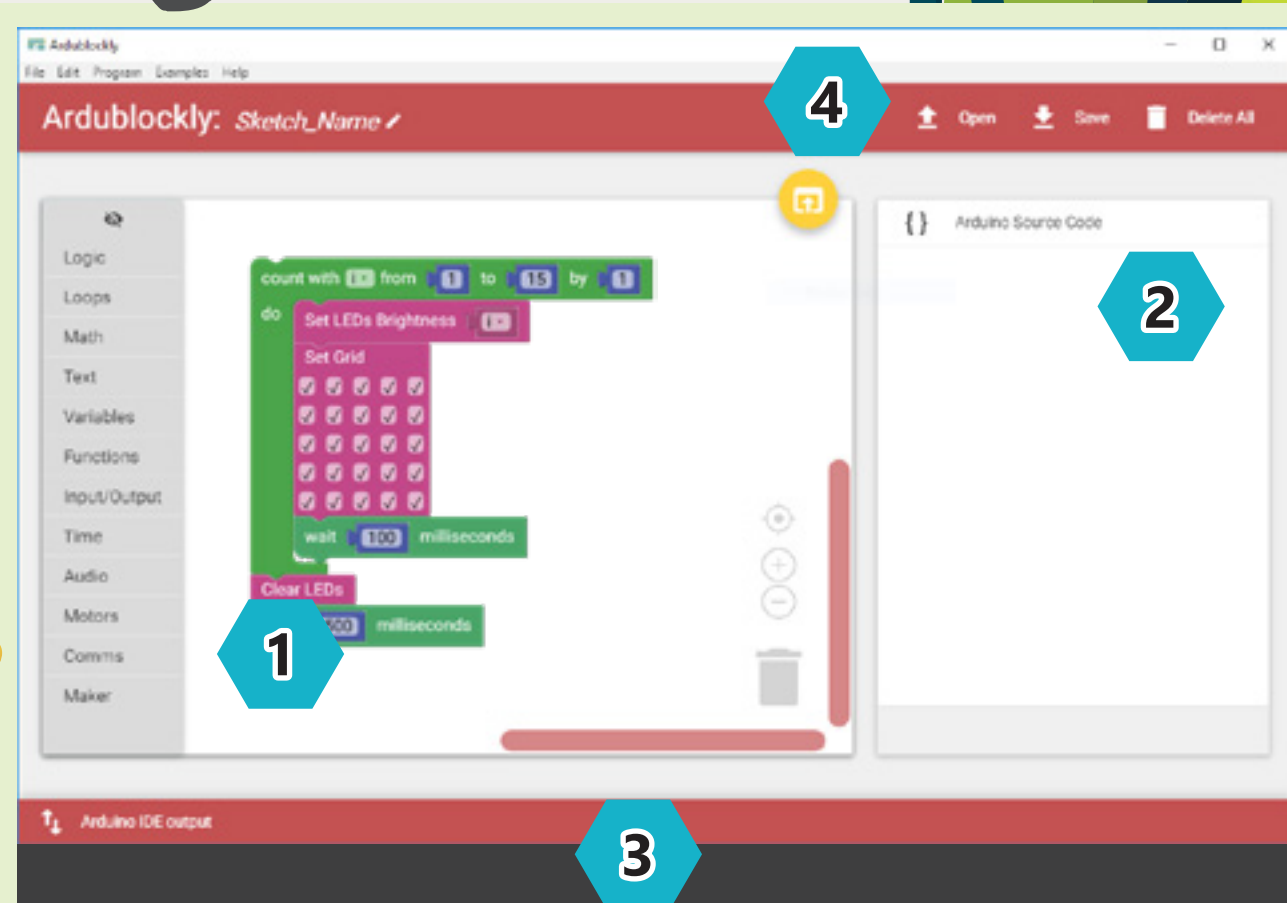


Creating Your First Program



Ardublockly Interface

The Ardublockly interface is shown in the following figure.



The interface is divided into four sections.

- 1.** The **blocks** section: In this section, you drag the blocks you want from the left side and drop them in the white area to form your program. You can zoom in or out, delete blocks or modify their values.
- 2.** The **Arduino Source Code** section: This section shows the Arduino code which corresponds to the blocks you dragged. This code is generated automatically by the software and you cannot modify it unless you made the change in the **blocks** section. You can still copy the Arduino source code to the Arduino IDE interface and change it from there.
- 3.** The **Arduino IDE output** section: When you download your program to the Maker board, the status of the download process or any error message will appear in this section. You can hide or show this section by clicking on the section.
- 4.** The **Quick Access** section: This section contains the common use buttons like, **Open**, **Save** and **Delete All**.

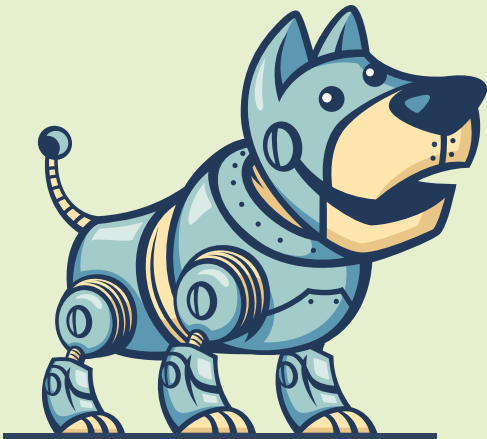


What Are the Different Blocks?

In Ardublockly, there are many categories for the blocks. Each category has a special purpose. The main categories are as follows:

- ▶ Logic
- ▶ Math
- ▶ Variables
- ▶ Time
- ▶ Loops
- ▶ Text
- ▶ Functions
- ▶ Maker

The most common categories are the **Maker**, **Logic** and **Loops**. These are explained in the following pages. The rest of the categories are explained once a block is needed from that category.

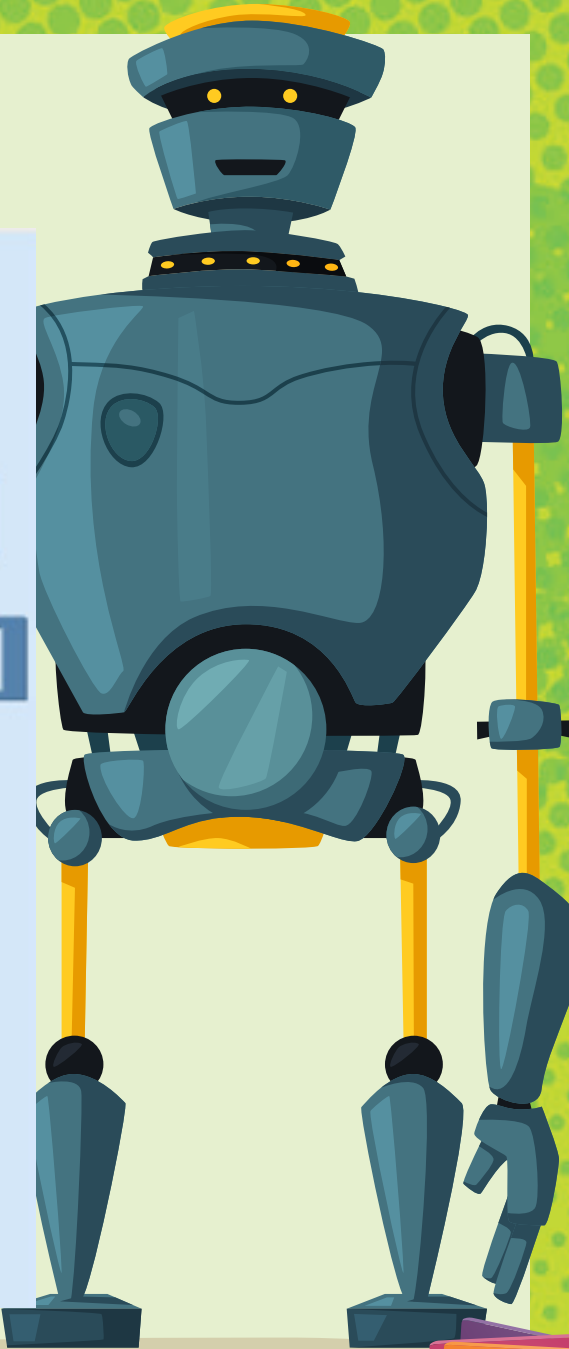
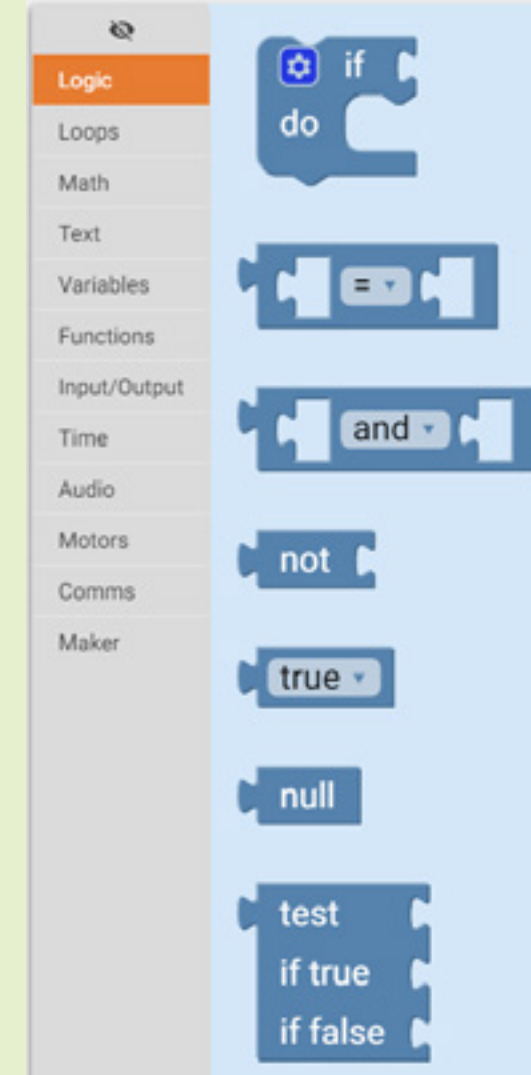
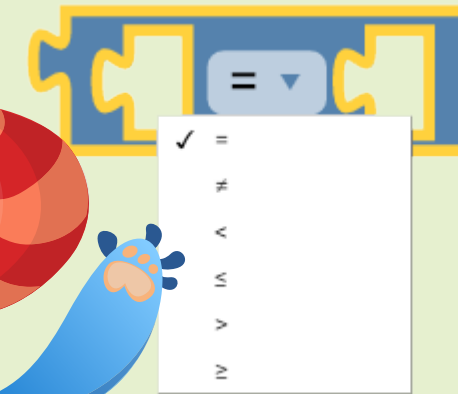


Logic

This category follows the Boolean Algebra system which is a branch of Algebra where the value of the variables has only two possible values; **True** or **False**.

Comparison Operators

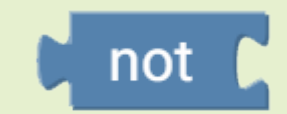
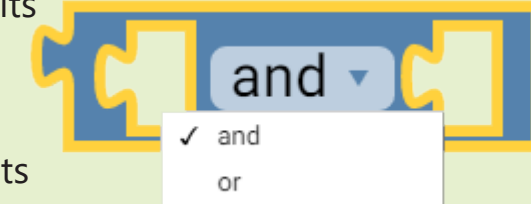
The comparison operators compare the values on either side of them and decide the relation between them. They are also called relational operators. They take two inputs and they return **True** based on how these variables compare to each other.



Operator	Description
=	If the values of two operands are equal, then the condition becomes true.
≠	If the values of two operands are not equal, then condition becomes true.
<	If the value of left operand is less than the value of right operand, then the condition becomes true.
≤	If the value of the left operand is less than or equal to the value of the right operand, then the condition becomes true.
>	If the value of left operand is greater than the value of right operand, then the condition becomes true.
≥	If the value of left operand is greater than or equal to the value of right operand, then the condition becomes true.

Logical Operations

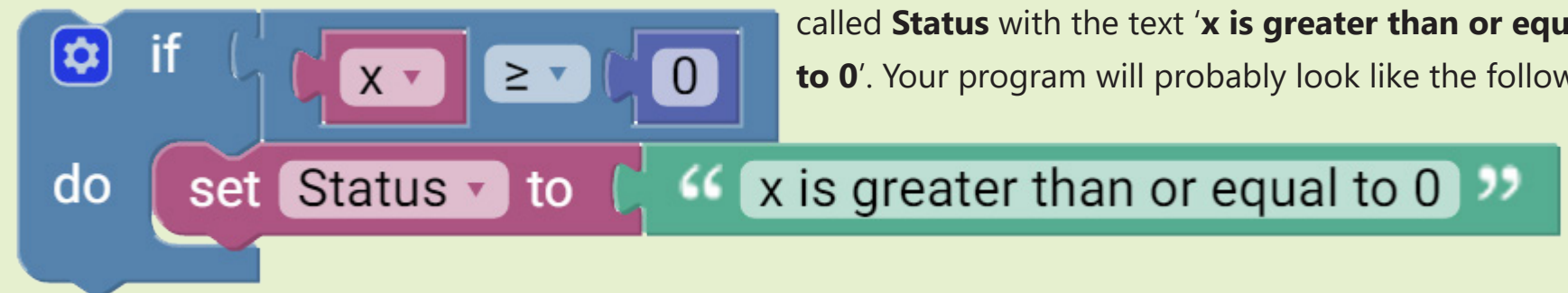
1. The **and** operator which returns **True** when both of its inputs are **True**.
2. The **or** operator which returns **True** when both inputs or one of them is **True**.
3. The **not** operator which converts its input into the opposite. If the input originally is **True**, it will become **False** and vice versa.



Conditional Statements

Conditional statements perform different actions based on their Boolean conditions. There are different ways of doing this. One way is to use the **if** block which you can configure, based on your need.

The **if** block is the simplest form of the conditional statements. If the condition is **True**, then you **do** something. For example; you have a number stored in a variable called **x** and you want to check the value of this variable. If the value is greater than or equal to zero, then you want to update another variable called **Status** with the text '**x is greater than or equal to 0**'. Your program will probably look like the following.



The other form of the conditional **if** statements, is the **if - else** form. This structure allows you to check the condition, and if it is **True** or not. If it is **True**, then you **do** something. **Else** you **do** another action.

For the same example, if the value of **x** is greater than or equal to zero, then you want to update the variable **Status** with the text '**x is greater than or equal to 0**'. Else, then update the variable **Status** with the text '**x is less than 0**'.

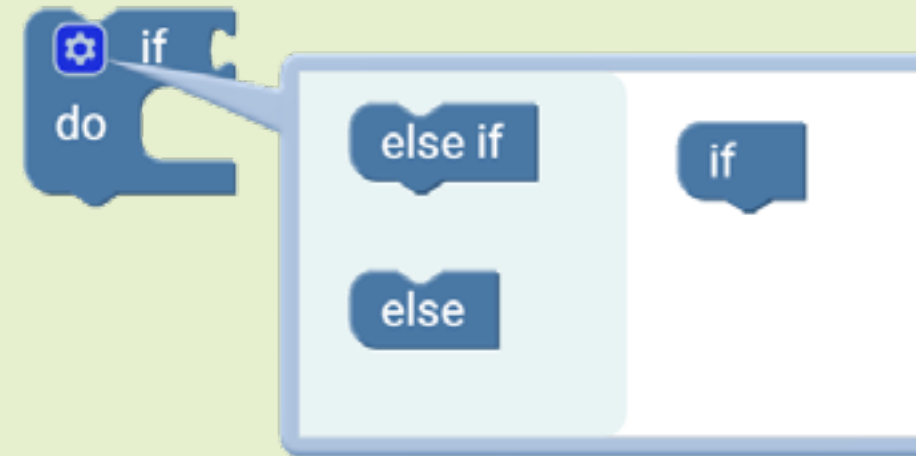
```
if (x ≥ 0)
do
  set Status to "x is greater than or equal to 0"
else
  set Status to "x is less than 0"
```



The third form of the conditional **if** statements, is the **if - else if - else** form. This structure allows you to check multiple conditions and do different actions.

For the same example, if the value of **x** is greater than zero, then you want to update the variable **Status** with the text '**x is greater than 0**'. But if the value of **x** is less than zero, then you want to update the variable **Status** with the text '**x is less than 0**'. Else, update the variable **Status** with the text '**x equals zero**' since it is the last case you may encounter.

```
if (x > 0)
do
  set Status to "x is greater than 0"
else if (x < 0)
do
  set Status to "x is less than 0"
else
  set Status to "x equals zero"
```



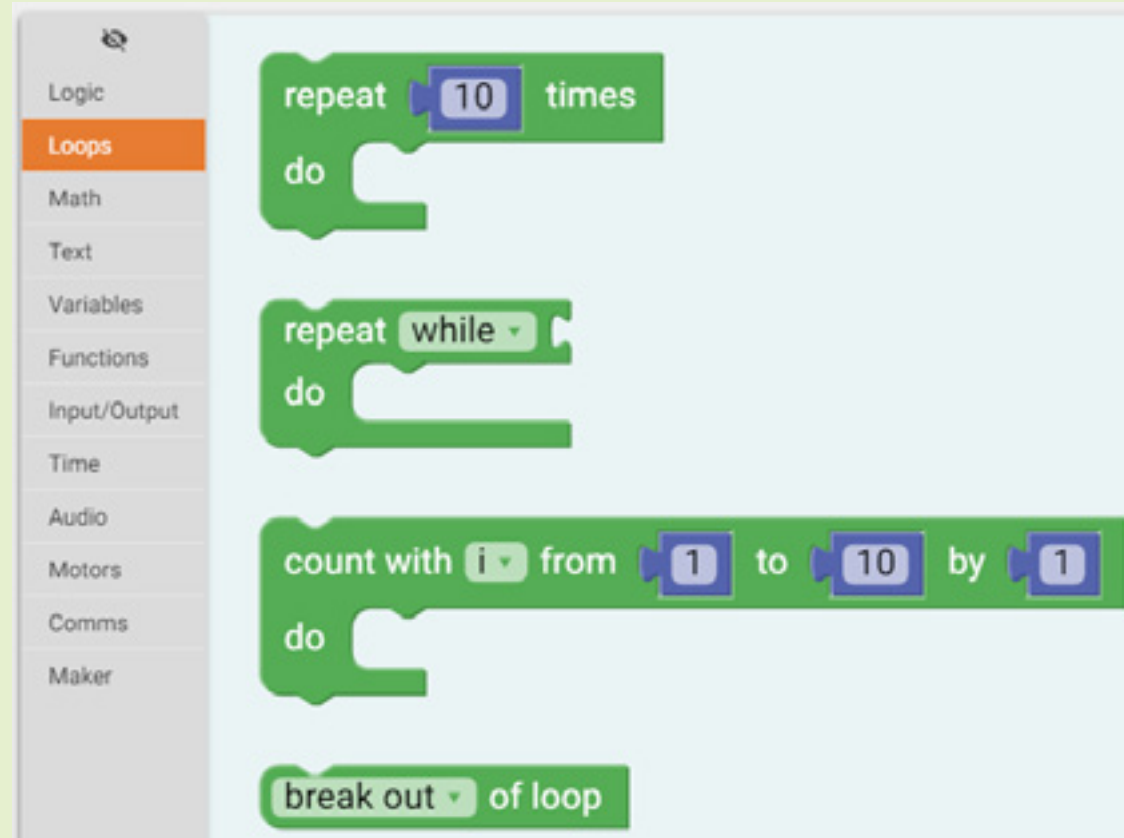
Building the if statement with its different forms can be done simply by clicking on the small gear icon in the **if** block. Then you can drag and drop as much as you want from the **else if** block and at most, one **else** block, and snap them to the **if** block.

Let's say you want an if statement with two else if blocks. After dragging the needed blocks from the left side of the menu and dropping them to the right side, you will end up with the following.



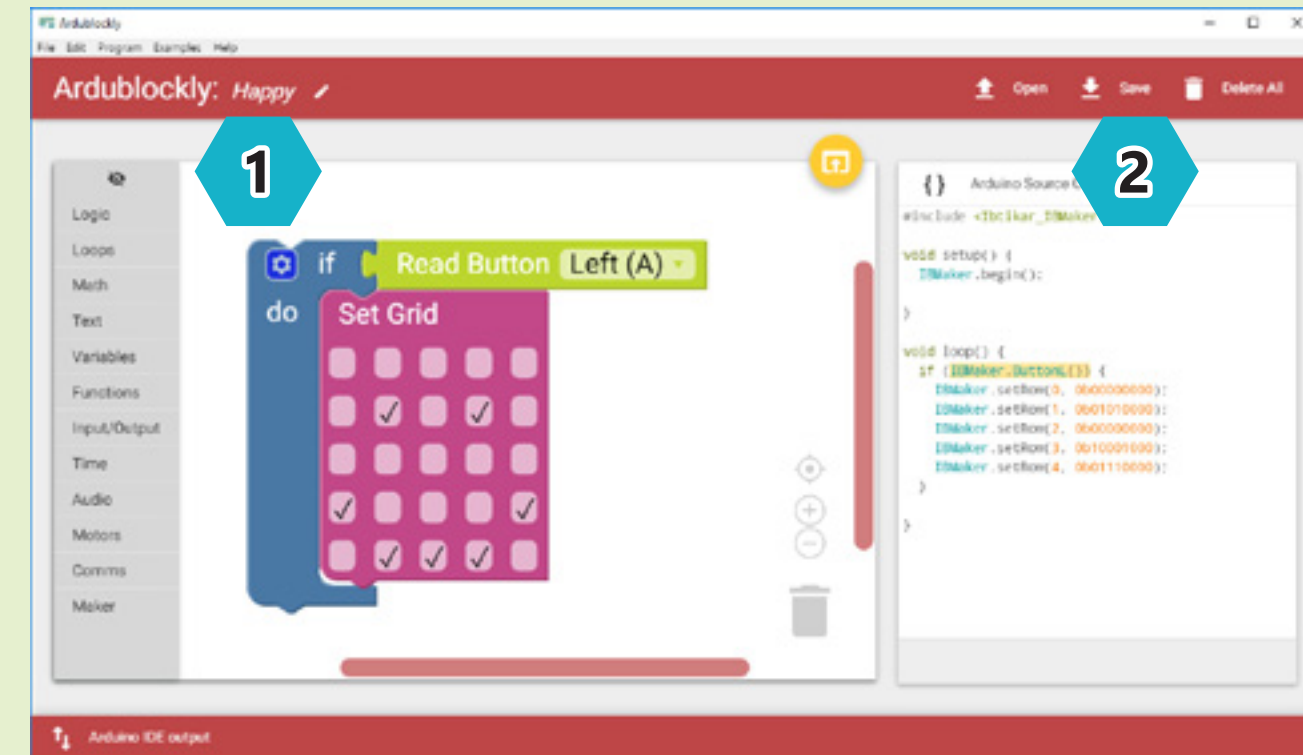
Loops

Sometimes it is important to repeat a program without stopping. One way is to repeat the blocks you need, but this does not make sense if your program is long or if you want the program to run without stopping. Luckily, there are loop blocks which allow you to repeat your program or a part of it for a certain number of times, until a condition is met or to loop for ever.



Create, Save and Load Your Projects

You can **Open**, **Save** and **Delete All** blocks using the **Quick Access** section. **Delete All** is useful if you have many blocks in the programming area and you want to get rid of them at once. If you want to save a program, first write the name you want as shown in **step 1**. The next step is to click on **Save** as shown in **step 2**. A pop-up window will appear which allows you to choose the location of the program.



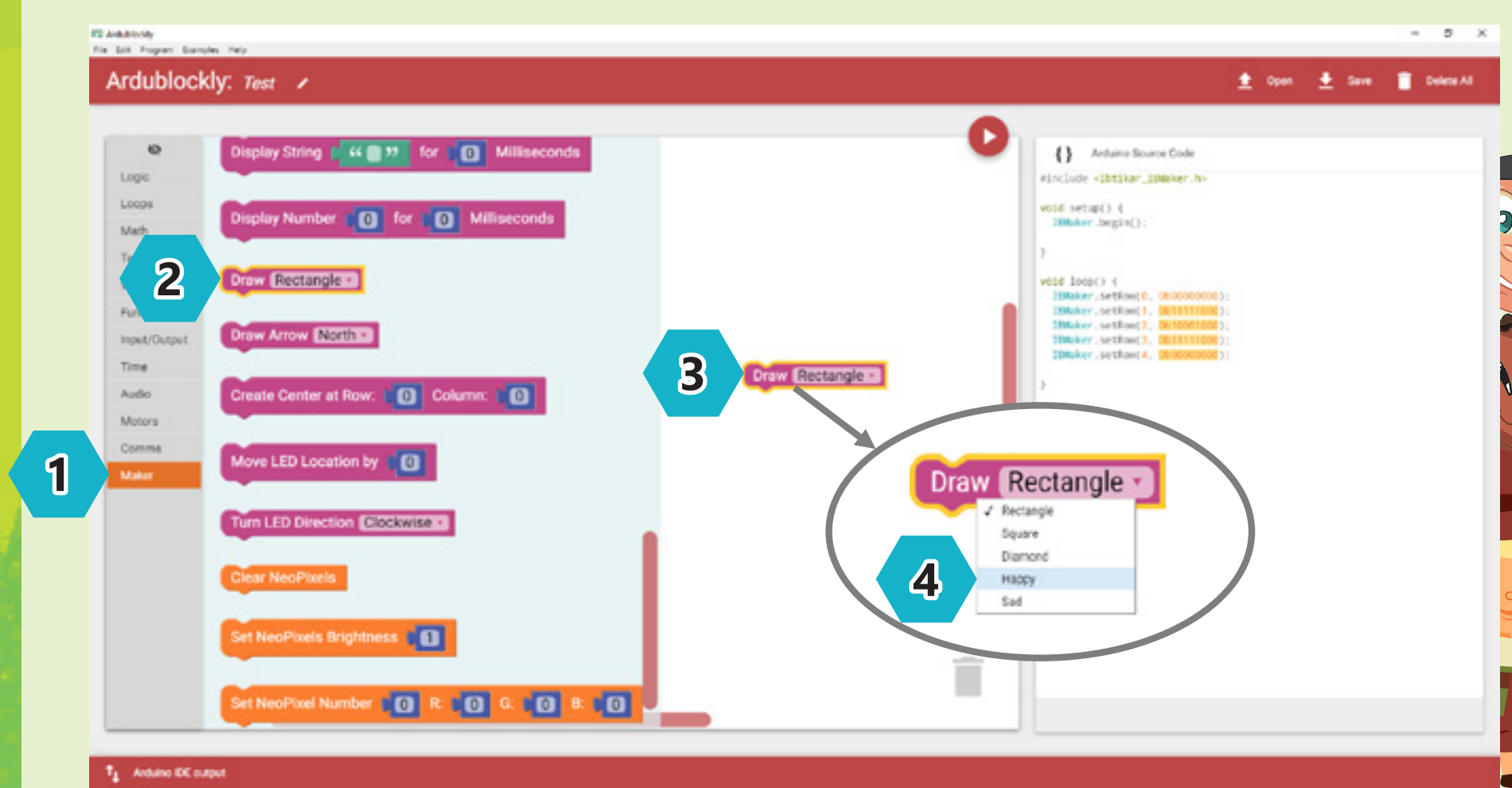
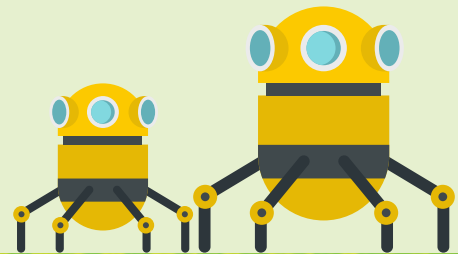
If you decide later to open another program you made before, all you need to do, is to click on **Open** and a pop-up window will appear. Locate the program you want to open and click open.



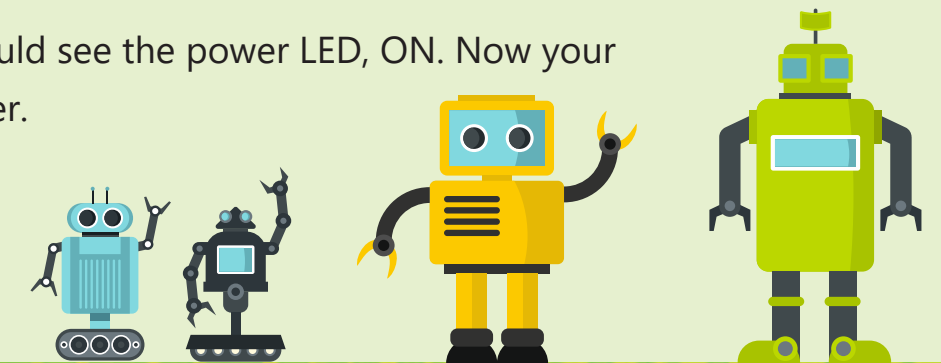
Program Your Ibtikar Maker

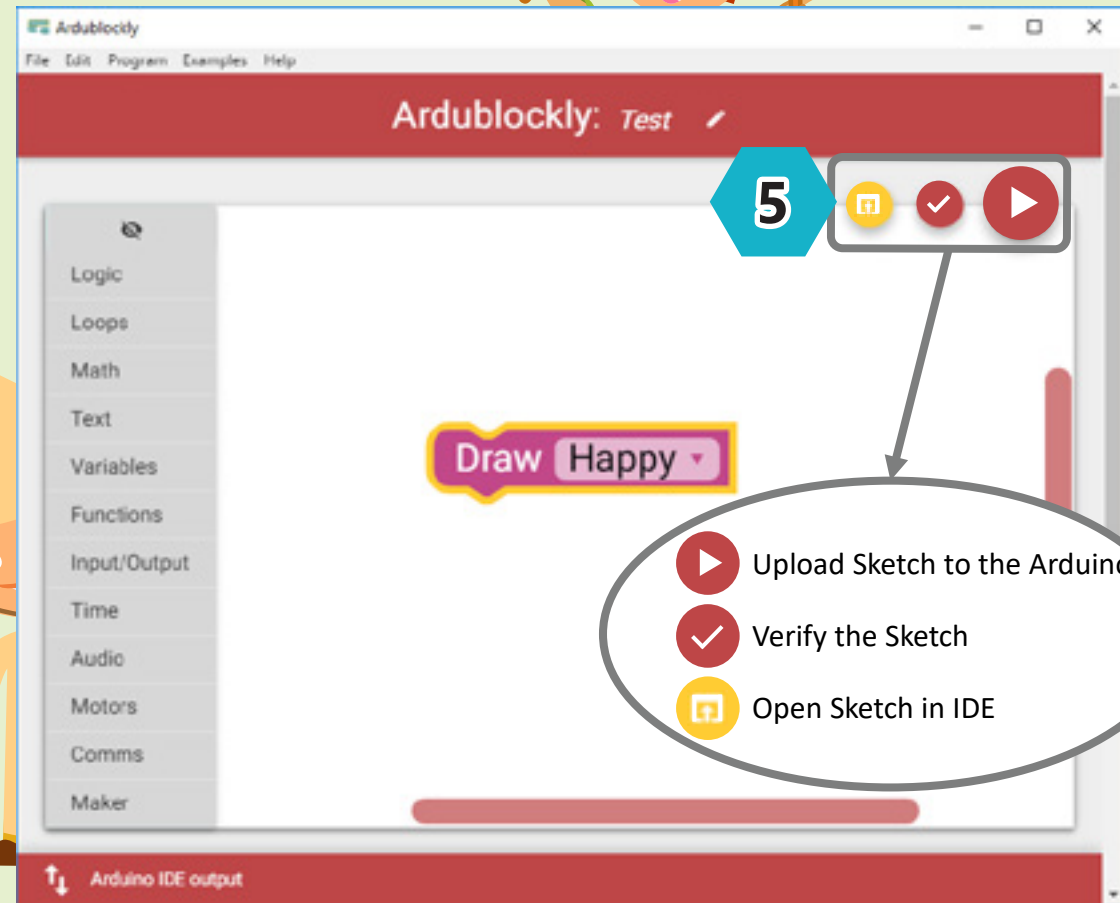
The first program you can try is to display a happy face on your Maker LED grid. To do this you will need the **Draw** block. Follow the next steps:

1. Go to the **Maker** menu.
2. Scroll until you find the **Draw** block.
3. Drag the **Draw** block to the workspace.
4. Change the shape you want to draw from **Rectangle** into **Happy**.



Connect the Maker to your computer. You should see the power LED, ON. Now your program is ready to be uploaded to your Maker.





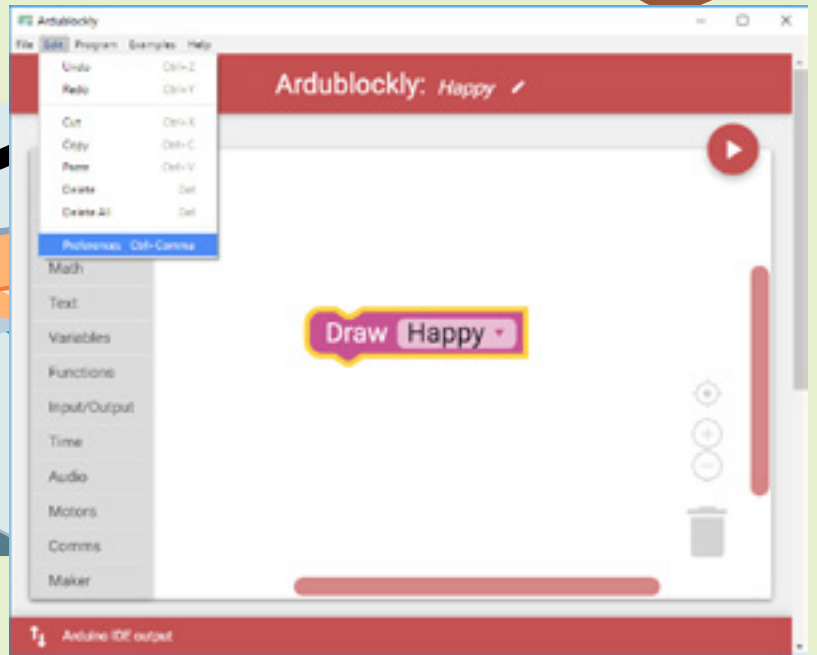
5. Hover the mouse near the download area. The menu will expand, and you will see three options. You can either upload the sketch directly to your maker, verify the code before you install it or open the sketch in the Arduino IDE.

Upload the sketch to the Maker and wait around 10 seconds. You should see the On-Board LED blink as the program is uploaded to the Maker. After that you should see the happy face on the LED grid.



Save your program. You can call it 'Happy'.

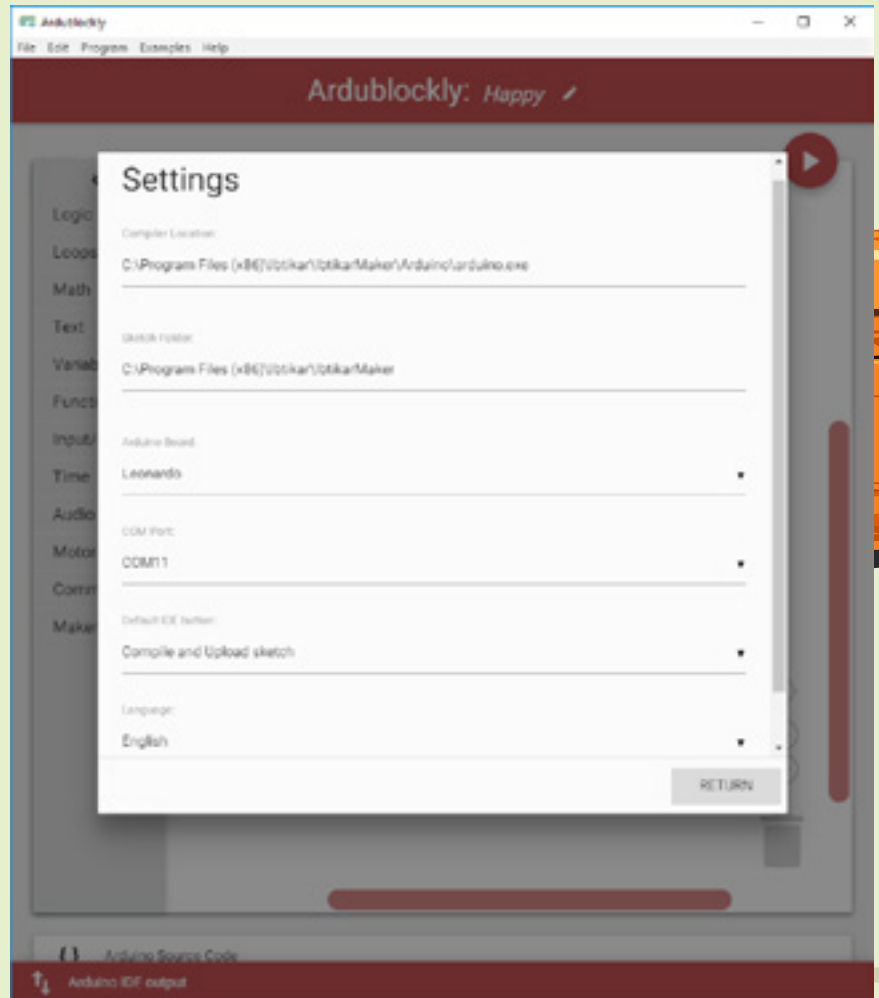
Sometimes, you may receive a message in the Arduino IDE output that the serial port is unavailable. This could happen if you forget to plug your Maker to your computer.



This will open the settings window. Check that the compiler and sketch locations are in the same directory as shown. When you install Ardublockly, these locations are chosen by default. Check that the selected Arduino board is Leonardo and that it is connected to a COM port. The COM port number may be different on your computer. Once all these settings are in place, your Maker is ready to be programmed.

In case you have already plugged in, but you still receive the same message, try the following steps.

From Ardublockly **Edit** menu, go to **Preferences**





Maker Activities

On-Board LED

The on-board LED can be controlled to be ON or OFF. This LED has its own block in the Maker menu.

Activity 1: LED ON

In this activity, the on-board LED will be ON forever. Forever means until you unplug the Maker or the battery dies. You will need one LED block as shown.

Upload the program. The LED will be ON.

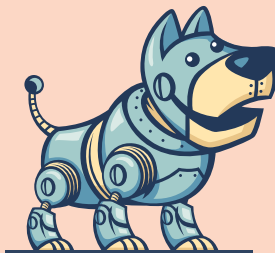


Activity 2: LED Blink

In this activity, the on-board LED will blink each second. You will need two LED blocks and two wait blocks as shown.



Arrange the blocks as shown and upload the program. The LED will blink each second.



LED Grid

Maker can control the 25 LEDs at once or each one individually. In the Maker menu there are many blocks available. Some of these blocks allow you to display numbers, characters, scroll strings, or even draw shapes.

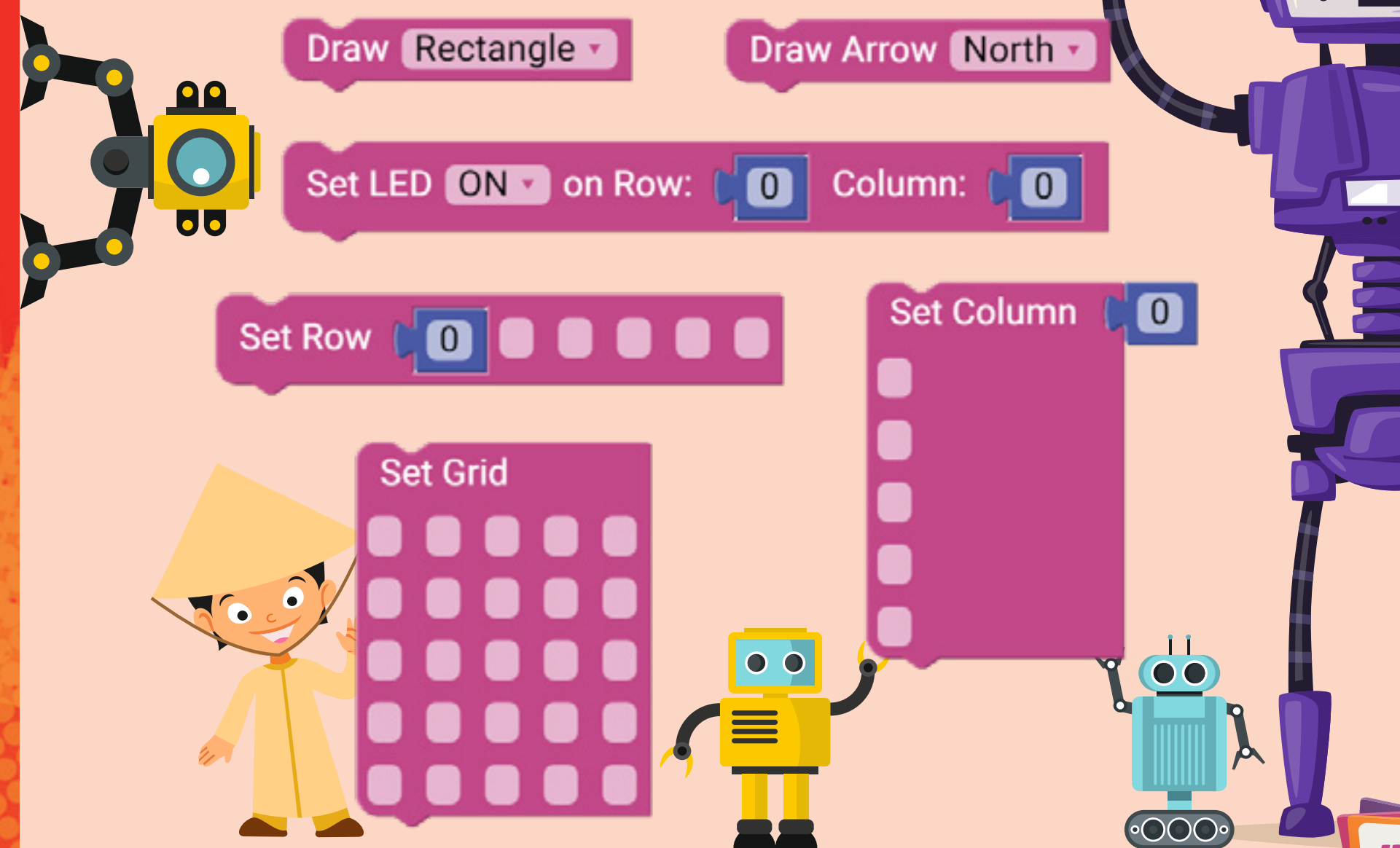
Activity 3: Draw Shapes

You can draw shapes on the Maker using different methods. For example, you can use:

- ▶ The **Draw Shape** block to draw a rectangle, square or a diamond shape or you can draw a happy or a sad face.
- ▶ The **Draw Arrow** block to draw arrows with different directions. You can draw north, north-east, east, east-south, south, south-west, west or north-west.
- ▶ The **Set LED** block to turn ON or OFF an individual LED by specifying the row and column.



- ▶ The **Set Row** block to control the LEDs in a specific row at once.
- ▶ The **Set Column** block to control the LEDs in a specific column at once.

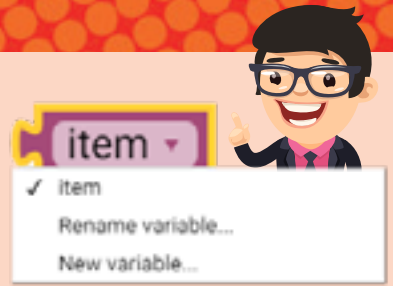




In the following activity, you will create a program to blink each LED individually using two for-loops, one for the rows and one for the columns. You will need the following blocks.

```

count with i from 1 to 10 by 1
do
  count with i from 1 to 10 by 1
  do
    Set LED ON on Row: 0 Column: 0
    Set LED ON on Row: 0 Column: 0
    wait 1000 milliseconds
    item
    wait 1000 milliseconds
    item
  
```



Since you need to go through rows and columns at the same time, then you need two different variables, one for the rows and for the columns. You can create a new variable by clicking on the small arrow in the variable block and then clicking on **New variable**.

Another method is to click on one of the for-loop blocks and then click on **New variable**. Both methods will generate a variable called 'j' automatically.

```

count with i from 0 to 4 by 1
do
  count with j from 0 to 4 by 1
  do
    Set LED ON on Row: i Column: j
    wait 100 milliseconds
    Set LED OFF on Row: i Column: j
    wait 100 milliseconds
  
```

Now, connect the blocks as shown in the figure. Since the LED grid is 5 by 5 and the indexing starts from 0, then each loop should count from 0 to 4.

If you swapped the inner and outer loops, then the LED pattern will be a vertical one instead of horizontal.



Activity 4: Display a Number

To display numbers on the Maker, you need the **Display Number** block.

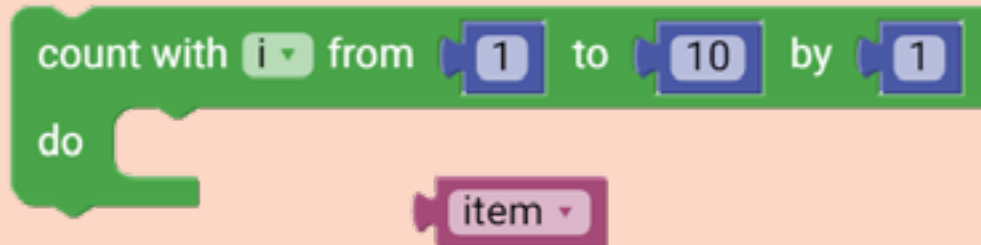
This block accepts integer numbers (numbers without a fraction). It also accepts variables, so you can display a counter for example. In the second parameter, you specify the scrolling step duration.

The following program will display numbers from 1 to 15 with a 100-millisecond scrolling duration. You will need the following blocks:



Now, upload the program to your Maker. You should see a counter from 1 to 15 on the LED grid.

Display Number 0 for 0 Milliseconds



Display Number 0 for 0 Milliseconds

Arrange the blocks and change their parameters as shown. Click on the variable block and change the name of the variable to match the one in the for loop, which is in your case is the variable 'i'.



Activity 5: Display a Character

To display a single character (either a letter, number or a symbol) on the Maker, you need the following block.

This block accepts a single character. In the second parameter, you specify the display duration. This means that the character you want, will be displayed at once without scrolling for the duration you specify.

To see the difference, try the following program. Once you upload it to the Maker, you will see the question mark symbol for 1 second then number 4 for another second. This will repeat forever.

Display Character ' ? ' for 1000 Milliseconds

Display Character ' 4 ' for 1000 Milliseconds

Display Character ' ' for 0 Milliseconds



Activity 6: Scroll a String

If you want to scroll a text, your name for example, you will need a third block called **Display String**. Note the difference between the single quotation in the **Display Character** block and the double quotation in the **Display String** block.

Display String “ ” for 0 Milliseconds

In the following example, we will scroll “Hello Maker” with a duration of 200ms for each scrolling step.

Display String “ Hello Maker ” for 200 Milliseconds

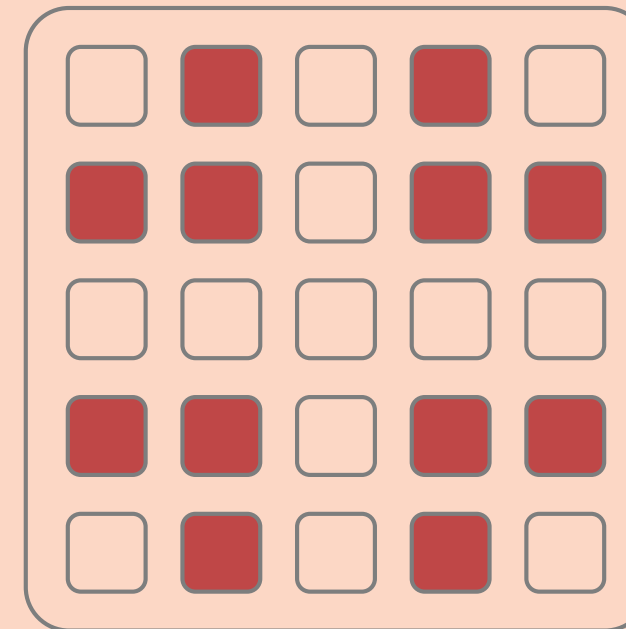
Upload the program to your Maker and check the result. Now scroll your name.



Activity 7: Move and Turn an LED

Since the 25 LEDs form a grid, you can consider them as a coordinate system. This means that you can create an origin, move an LED in a certain direction and turn clockwise or counter clockwise. This can make it easy to create complex patterns in a small number of blocks.

In this activity, you will create the following pattern with one LED being ON at a time.



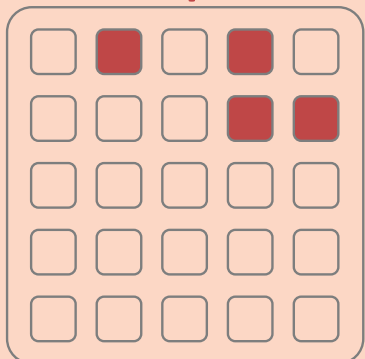
Create Center at Row: 0 Column: 0

Move LED Location by 0

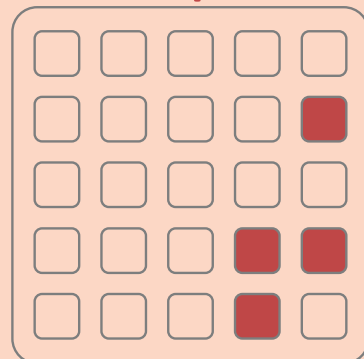
Turn LED Direction Clockwise

You can think of this as a pattern that repeats itself 4 times, as shown.

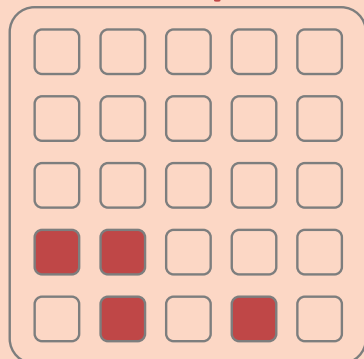
Loop 1



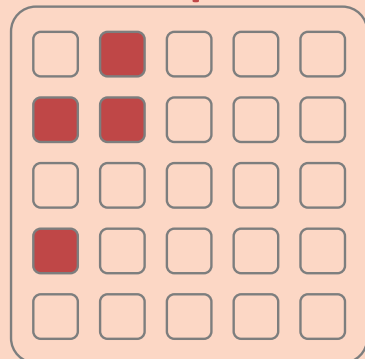
Loop 2



Loop 3



Loop 4



You will need the following blocks. The purple Arduino block is needed when you want some parts of your code to run once and some other parts to run forever.

1 X	Arduino run first:	Create Center at Row: 0 Column: 0	X 1	
1 X	Arduino loop forever:	Move LED Location by 0	X 3	
1 X	repeat 10 times	do	Turn LED Direction Clockwise	X 3
		wait 1000 milliseconds	X 3	

The origin should start from the point (0,1). Then the LED will move two steps, turn clockwise, move one step, turn counter clockwise and finally move one step. If the same pattern is repeated 4 times, you will end up with the required pattern. Since this process is very fast, a time delay is needed in each part. Arrange your program as shown.

Arduino run first:

Create Center at Row: 0 Column: 1

Arduino loop forever:

```

repeat 4 times
do
  Move LED Location by 2
  Turn LED Direction Clockwise
  wait 250 milliseconds
  Move LED Location by 1
  Turn LED Direction Counter Clockwise
  wait 250 milliseconds
  Move LED Location by 1
  Turn LED Direction Clockwise
  wait 250 milliseconds
  
```



Activity 8: LEDs Brightness

The last two blocks that control the LED grid are the **Set LEDs Brightness** and **Clear LEDs**.

The brightness block allows you to change the brightness of the 25 LEDs at once. You can change the brightness from 1 to 15. The clear block turns OFF all LEDs at once. This is useful if you made a pattern and then you want to turn the grid OFF without the need for turning OFF each LED individually.

In this activity, you will increase the brightness from 1 to 15 in steps one at a time. Inside the **loop** block, you will add the brightness block, turn all LEDs ON and wait for 100ms. Once the loop is finished, you will clear all the LEDs at once and wait for 500ms. Your program should look like the following.



Clear LEDs

Set LEDs Brightness 1



Note:

To avoid hurting your eyes do not look directly at the LED grid when the brightness is high.



Read the Buttons

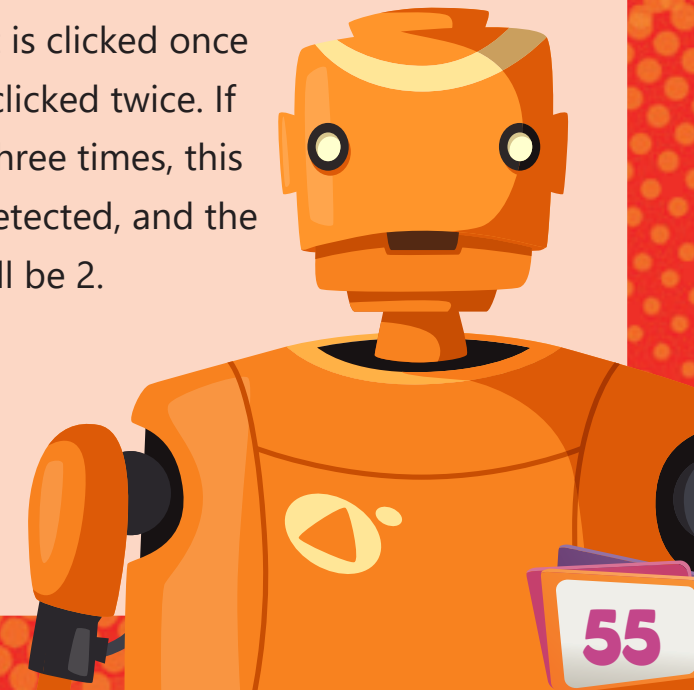
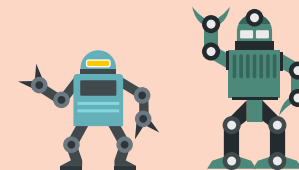
The Maker has two buttons that you can read. These buttons are button A on the left side of the Maker and button B on the right side. In the Maker menu, there are two blocks that you can use, one that checks if the button is pressed or not. The other block returns the number of clicks you clicked on a button.

Read Button Left (A)

Left (A)
Right (B)

Read Button Left (A) Count 2 Times

For both blocks, you need to choose which button you want to read. The second block needs the maximum number of clicks you want to reach. For example, if the value is 2, this means the block returns 0 if it is not clicked, 1 if it is clicked once and 2 if it is clicked twice. If you clicked three times, this will not be detected, and the maximum will be 2.



Activity 9: Read Both Buttons

In this activity, you will read both buttons and display a character on the LED grid. If button A is pressed, then 'A' will be displayed on the grid. Similarly, if button 'B' is pressed, B will be displayed on the grid. Try the following program and upload it to your Maker.



```
if Read Button Left (A)
do
  Display Character 'A' for 200 Milliseconds
if Read Button Right (B)
do
  Display Character 'B' for 200 Milliseconds
```



Activity 10: Detect Multiple Clicks

In this activity, you will read how many times button A is pressed and display a happy face if you reach the maximum number of clicks. If you do not reach the maximum value, you will show a sad face.

Since the **Read Button Count** block will return different values based on how many times you press the button, then it is better to read the block and store the value in a variable. Then based on the value stored, you can take decisions. You will need the following blocks.

Try the following program and upload it to your Maker and see the maximum number of clicks you reached.



```
if do else
  Read Button Left (A) Count 2 Times
  set item to
  Draw Rectangle
  Draw Rectangle
  item 0
  else if
  else
```

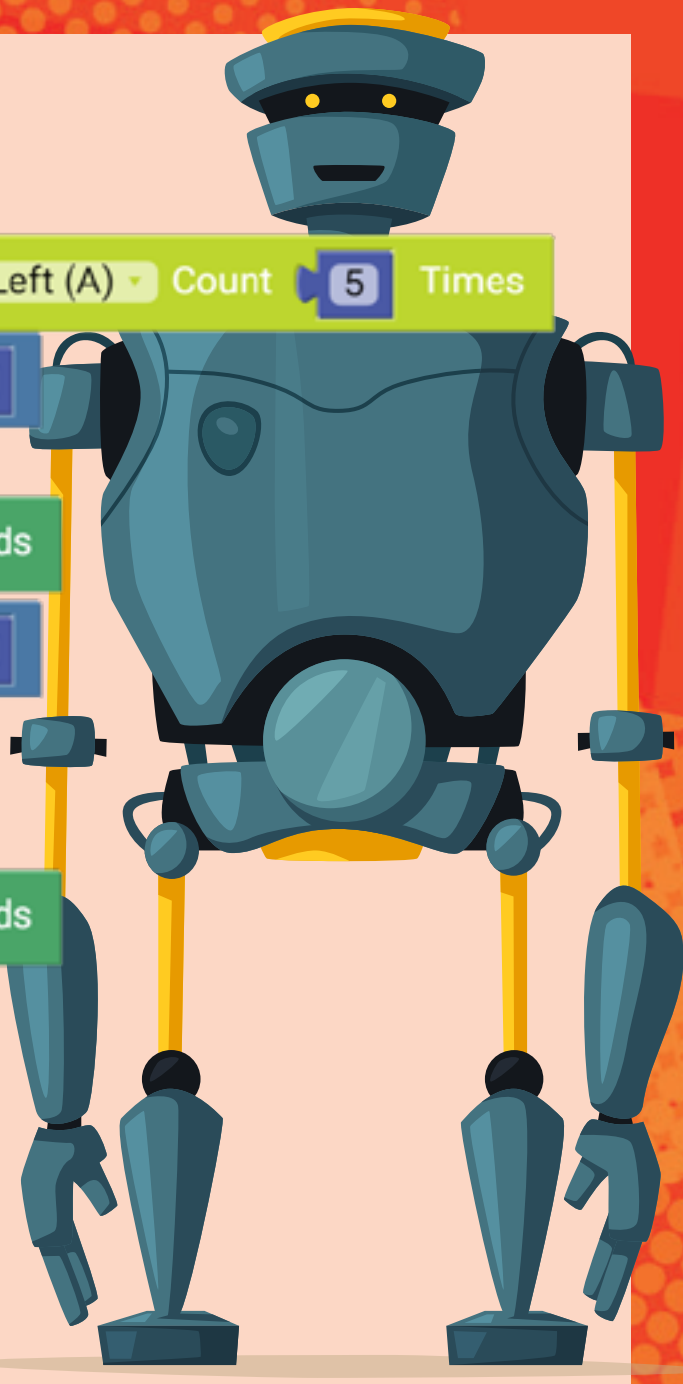
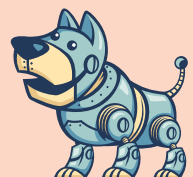
```
set item to Read Button Left (A) Count 5 Times
if
do Draw Happy
else Draw Sad
```



Activity 11: Detect Multiple Clicks (Optimised)

You may have noticed that even when you do not click button A at all, the sad face is still there. You can improve the code by adding another case to check when the number of counts is zero. It is good to add **wait** blocks when you display the happy and sad faces. Try the following program.

```
set item to Read Button Left (A) Count 5 Times
if item = 5
do
  Draw Happy
  wait 500 milliseconds
else if item = 0
do
  Clear LEDs
else
do
  Draw Sad
  wait 500 milliseconds
```



Read the Temperature

The temperature sensor on the Maker can be used to measure the board temperature. The temperature block in the Maker menu allows you to read the temperature in either the Celsius or Fahrenheit unit.



```
Read Temperature Celsius (C)
```

- ✓ Celsius (C)
- Fahrenheit (F)

Activity 12: Display the Temperature

```
Display Number 0 for 0 Milliseconds
```

```
Read Temperature Celsius (C)
```

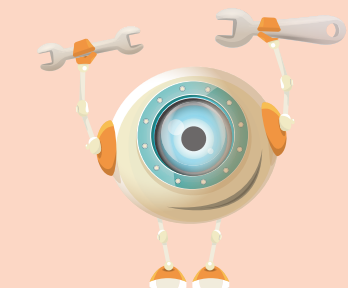
```
Display Number 0 for 0 Milliseconds
```

```
Read Temperature Celsius (C)
```

```
Display String " " for 0 Milliseconds
```

```
Display String " " for 0 Milliseconds
```

In this activity, you will scroll the temperature in both the Celsius and Fahrenheit units on the LED grid. You will need these blocks.



```
Display Number Read Temperature Celsius (C) for 200 Milliseconds
```

```
Display String " C " for 200 Milliseconds
```

```
Display Number Read Temperature Fahrenheit (F) for 200 Milliseconds
```

```
Display String " F " for 200 Milliseconds
```

Try the following program and upload it to your Maker.



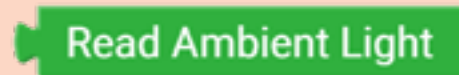
Read Ambient Light

The light sensor on the Maker measures the ambient light. It returns a value between 0 to 1023 representing the light intensity level. Higher values mean the measured light level is high (there is light). You can find its block in the Maker menu.

Activity 13: Display the Ambient Light Value

In this activity, you will scroll the ambient light value on the LED grid. While you are testing, you can use a torch and direct it toward the sensor to notice the change. You will need these blocks.

Try the following program and upload it to your Maker.



Buzzer

The magnetic buzzer generates tones by controlling the frequency and duration of the note needed. The buzzer shows as one block in the Maker menu. This block by itself is a complete program.



Activity 14: Play Different Tones

Using 3 **Play Tone** blocks, generate tones with different frequencies and fixed duration. Then fix the frequency and change the duration. In each case, upload the program to your Maker board and note the difference.




```

Play Tone: Frequency 300 Duration 250
Play Tone: Frequency 400 Duration 250
Play Tone: Frequency 500 Duration 250

```

Program 1:
Variable frequency
Fixed duration

Program 2:
Fixed frequency
Variable duration

```

Play Tone: Frequency 300 Duration 250
Play Tone: Frequency 300 Duration 500
Play Tone: Frequency 300 Duration 750

```

If you tried to upload these programs, you will notice that it is not easy to tell which one is which. It may also be annoying to listen to them because they repeat forever.



If you want to run a program once, add it in the **Arduino run first** block. This block can be added once in a program and it is useful when you want to initialise variables or set properties of some components like brightness for example. Add a **wait** block after each note to have time to listen to it.

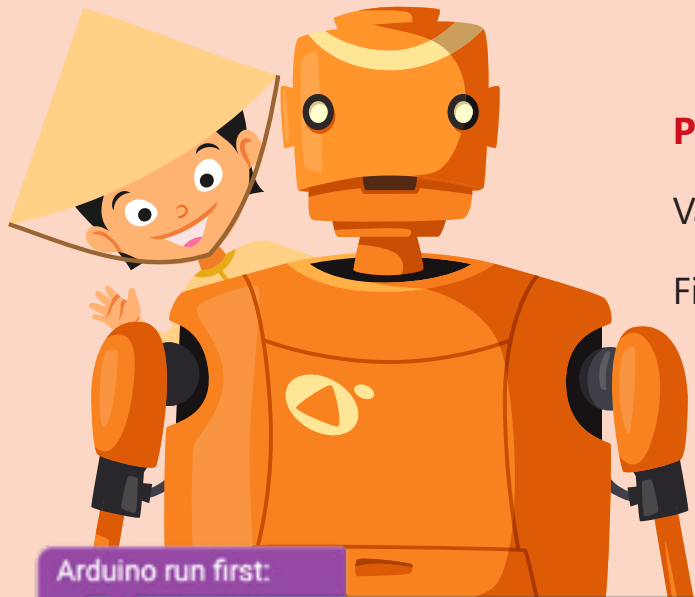
```

Arduino run first:
Arduino loop forever:

```



Your programs should now look like these.



Program 1:
Variable frequency
Fixed Duration

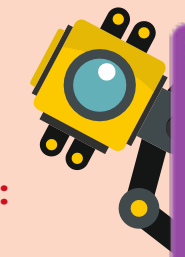
```

Arduino run first:
Play Tone: Frequency 300 Duration 250
wait 500 milliseconds
Play Tone: Frequency 300 Duration 500
wait 500 milliseconds
Play Tone: Frequency 300 Duration 750
wait 500 milliseconds
Arduino loop forever:

```



Can you tell the difference?



```

Arduino run first:
Play Tone: Frequency 300 Duration 250
wait 500 milliseconds
Play Tone: Frequency 400 Duration 250
wait 500 milliseconds
Play Tone: Frequency 500 Duration 250
wait 500 milliseconds
Arduino loop forever:

```



Program 2:
Fixed frequency
Variable Duration



Activity 15: Loop Tone Frequency

The **Play Tone** block accepts parameters as integer numbers or variable. In this activity, you will change the frequency from 100 to 2000 Hz with a step of 50 Hz, using a **for loop** block. The duration will be fixed at 250 milliseconds. The loop and tone blocks will be played once in the code, so an **Arduino run first** block is needed.

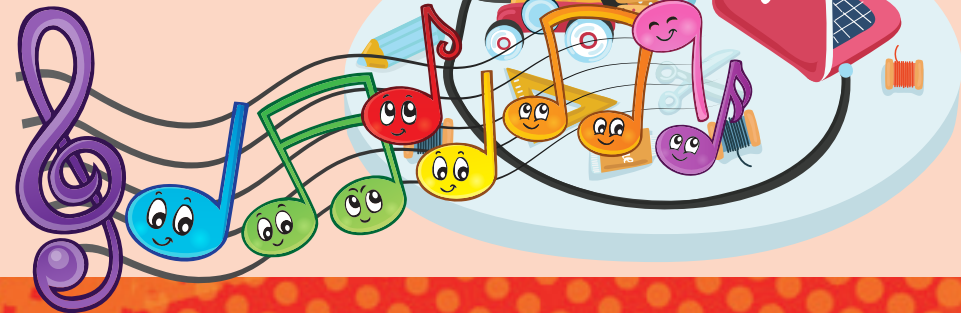
Your program should now look like this.

Arduino run first:

```
count with i from 100 to 2000 by 50
do
  Play Tone: Frequency i Duration 250
```

Arduino loop forever:

Try the program, upload it to your Maker and enjoy the music you have just made.



Note

To avoid damaging the buzzer, do not use higher frequency values (more than 2000Hz).

Read the Sound Level

The sound sensor on the Maker measures the sound intensity level. The sound sensor shows as one block in the Maker menu. In this case, Maker is an exception in that, instead of the value increasing when the sound sensor detects noise, the value decreases. The range you may reach if you clap your hand for example, is between 400 and 0.

Read Sound

Activity 16: React to Sound

In this activity, you will detect the change in the sound level and play a tone. The tone frequency will depend on the sound level detected. You can pass the value of the detected sound directly to the **Play Tone** block.

You can even do more. You can cover a wider range of frequencies by mapping the sound range (0-400) to a frequency range (0-2000) Hz. To avoid hearing noise when the frequency is low, you can put the **Play Tone** block inside an **if** block so that you can hear a tone only when the value exceeds 350.

```
set item to Map Value Read Sound From Range [ 400 - 0 ] to Range [ 0 - 2000 ]
if item > 350
do
  Play Tone: Frequency item Duration 200
  wait 10 milliseconds
wait 1 milliseconds
```

Try this program, upload it to your Maker and clap your hand near the sound sensor. Did you hear anything?

Now put your mouth close and blow air on the sensor. Did you hear different notes?



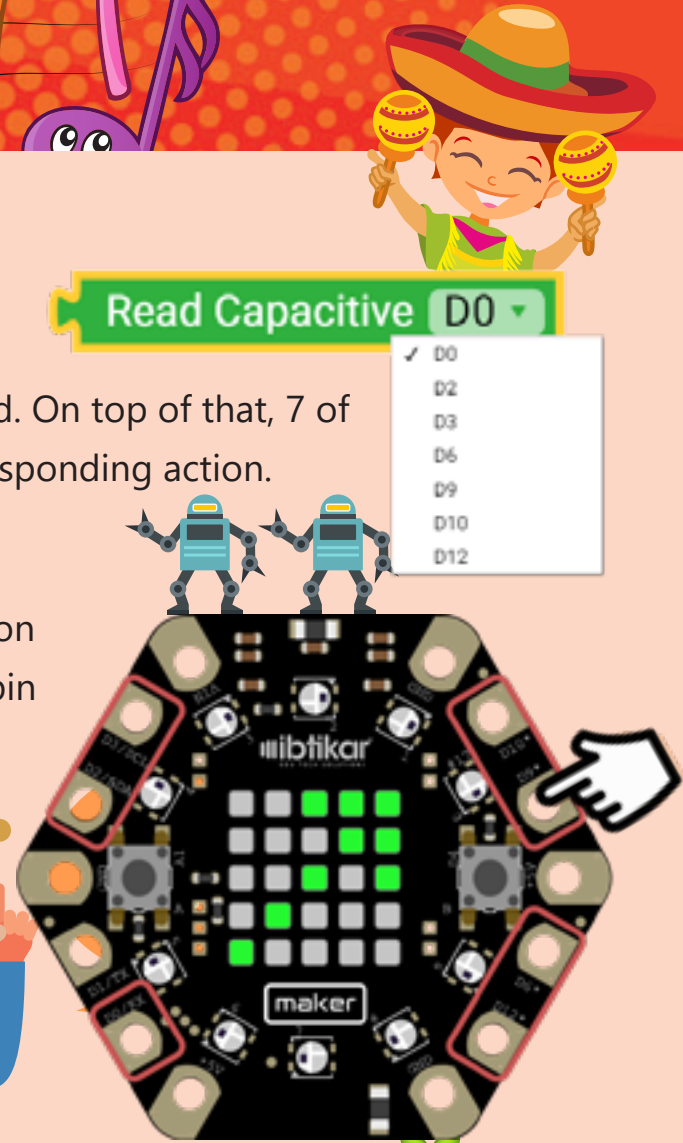
Pin Pads

These 8 pin pads can be interfaced with extra input and output modules allowing you to extend the capabilities of your Maker board. On top of that, 7 of them, can be used as touch sensors. So, you can touch them for a responding action.

Activity 17: Detect Touch

In this activity, you will program the Maker to show an arrow based on which touch pad you pressed. For example, if you touched pin 9 or pin 10, you should see an arrow toward these pins, as shown.

You will pick a threshold, say 200 and make the decisions based on that. Since the threshold will be the same for all touch pads, then you can use a variable and assign the value of 200 to it. This will make it easier if you want to change the value later.



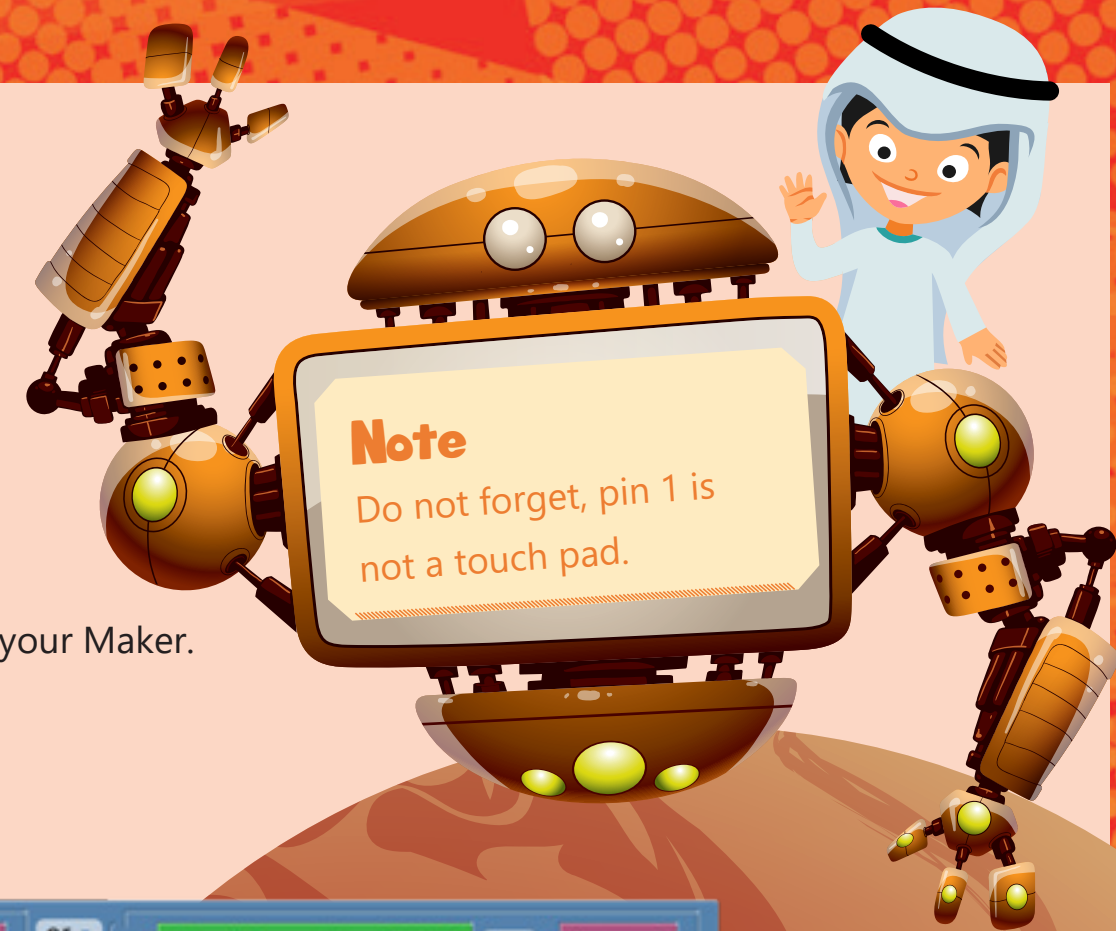
You will need the following blocks.

- 7 X X 1
- 1 X X 7
- 7 X X 3
- 4 X X 4

Try the following program and upload it to your Maker.

```

set item to 200
if Read Capacitive D0 > item
do Draw Arrow South West
if Read Capacitive D2 > item or Read Capacitive D3 > item
do Draw Arrow North West
if Read Capacitive D6 > item or Read Capacitive D12 > item
do Draw Arrow South East
if Read Capacitive D9 > item or Read Capacitive D10 > item
do Draw Arrow North East
    
```



Activity 18: Piano

In this activity, you will program the Maker to play a tone based on which touch pad you pressed. Try the following program and upload it to your Maker.

```
set item to 100
if Read Capacitive D0 > item
do Play Tone: Frequency 400 Duration 250
if Read Capacitive D2 > item
do Play Tone: Frequency 450 Duration 250
if Read Capacitive D3 > item
do Play Tone: Frequency 500 Duration 250
if Read Capacitive D10 > item
do Play Tone: Frequency 550 Duration 250
if Read Capacitive D9 > item
do Play Tone: Frequency 600 Duration 250
if Read Capacitive D6 > item
do Play Tone: Frequency 650 Duration 250
if Read Capacitive D12 > item
do Play Tone: Frequency 700 Duration 250
```

NeoPixels

The Maker has 10 RGB LEDs. Unlike the 25 LEDs which cannot change their colour, the RGB LEDs can be programmed to show any colour by combining the three different colours. Think of each pixel as three small LEDs combined, and each LED has a different colour (Red, Green and Blue).

The NeoPixels have the following blocks in the Maker menu.

```
Set NeoPixel Number 0 R: 0 G: 0 B: 0
```

```
Set NeoPixels Brightness 1
```

```
Clear NeoPixels
```

```
Set NeoPixel Number 0 Color White
```

Like the LED grid blocks, you can set the brightness of all NeoPixels using the **Set NeoPixels** block and turn them all OFF

using the **Clear NeoPixels** block. The other two blocks allow you to choose one of the

NeoPixels and choose its colour, either by specifying the RGB values or choosing a colour directly from the drop-down menu.

Each NeoPixel has a number written next to it. This will help you identifying which one you want to control.

Activity 19: Colour Wheel

In this activity, you will give each NeoPixel a different colour from the drop-down menu. Since there are 10 NeoPixels then you will need 10 blocks.

Try the following program and upload it to your Maker.

Did you see all the NeoPixels coloured?



Set NeoPixel Number 0 Color White

Set NeoPixel Number 1 Color Red

Set NeoPixel Number 2 Color Lime

Set NeoPixel Number 3 Color Yellow

Set NeoPixel Number 4 Color Cyan

Set NeoPixel Number 5 Color Magenta

Set NeoPixel Number 6 Color Green

Set NeoPixel Number 7 Color Purple

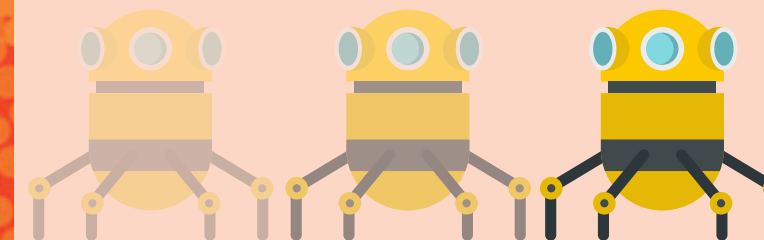
Set NeoPixel Number 8 Color Teal

Set NeoPixel Number 9 Color Navy

Activity 20: Fading Sequence

In this activity, you will change the brightness of the NeoPixels using a **for-loop**. The following program starts by turning OFF all RGB LEDs, then goes to the brightness **for-loop** block. The brightness can be changed from 0 to 255. For each NeoPixel, you can choose whatever colour you like.

A 10ms delay between each brightness update is added. Once the loop is finished, the program will wait half a second before starting from the beginning. Try the following program and upload it to your Maker.



```
Clear NeoPixels
count with i from 0 to 255 by 1
do
  Set NeoPixels Brightness i
  Set NeoPixel Number 0 R: 255 G: 0 B: 0
  Set NeoPixel Number 1 R: 255 G: 127 B: 0
  Set NeoPixel Number 2 R: 255 G: 255 B: 0
  Set NeoPixel Number 3 R: 127 G: 255 B: 0
  Set NeoPixel Number 4 R: 0 G: 255 B: 0
  Set NeoPixel Number 5 R: 0 G: 0 B: 255
  Set NeoPixel Number 6 R: 0 G: 255 B: 255
  Set NeoPixel Number 7 R: 255 G: 0 B: 255
  Set NeoPixel Number 8 R: 75 G: 0 B: 130
  Set NeoPixel Number 9 R: 255 G: 255 B: 255
  wait 10 milliseconds
wait 500 milliseconds
```


Activity 21: Looping RGB LEDs

Sometimes it is easier to loop the NeoPixels number instead of adding 10 blocks. This will help in making your program compact and easier to understand.

In this activity, you will set the 10 NeoPixels to a specific colour using 1 **NeoPixel** block and a **for-loop** block.

Try the following program and upload it to your Maker.

Clear NeoPixels

```
count with LED from 0 to 9 by 1
do
  Set NeoPixel Number LED R: 189 G: 71 B: 71
wait 250 milliseconds
```

Activity 22: Looping Colours and RGB LEDs

What if you want to change the RGB values?

In this case, you can add a separate **for-loop** block for each value. In the following program, you will change the value of the 10 NeoPixels at once. The three RGB loops will start from 0 to 255 with a step of 50. Each time the blue-colour loop finishes, the green-colour loop will change the value of **G** with a step of 50. And each time the green-colour loop finishes, the red-colour loop will change the value of **R** with a step of 50.

Try the following program and upload it to your Maker.

Clear NeoPixels

```
count with R from 0 to 255 by 50
do
  count with G from 0 to 255 by 50
  do
    count with B from 0 to 255 by 50
    do
      count with LED from 0 to 9 by 1
      do
        Set NeoPixel Number LED R: R G: G B: B
      wait 100 milliseconds
```

You can reduce the step in each colour loop, but this will make your program take a longer time to loop over all the colours.

Triple Axis Accelerometer

This sensor is in the middle of the board. Accelerometers are used to measure acceleration, which is how fast something is speeding up or down. An accelerometer can measure static acceleration like gravity, which is useful in detecting tilt, like when your phone tilts.

Enable Acceleration Sensor FALSE ▾

Read Acceleration of Axis X ▾

The accelerometer has two blocks in the Maker menu. The first one is used to enable or disable the sensor. When you want to use the accelerometer, you cannot touch pin 3 and 4 and vice versa. The second block allows you to read the acceleration value in one of the three axes (X, Y or Z).

Note

If you are using the accelerometer sensor, you should not touch or use touch pads 2 and 3 because they are sharing the same data lines. If you touched them accidentally, the Maker will stop working and you will need to reset it using the Reset button on the back side. You can still use the other touch pads.

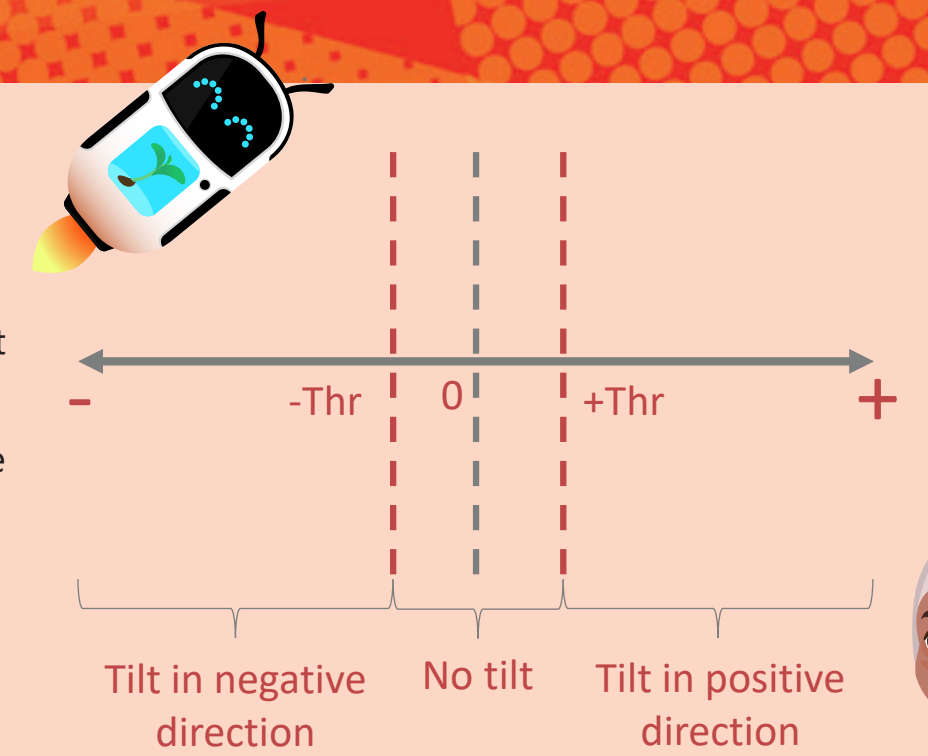
In the Arduino IDE manual, you will learn how to prevent the Maker from hanging forever by using a timer. If you touch the pads and the Maker stops for a certain time, this timer will reset the Arduino automatically.

TOP

The value of the accelerometer could be a positive number, a negative number or a zero. When the value of the acceleration in the X-axis is zero, for example, this means the Maker is not tilted in that direction. Sometimes, you might get a small positive or negative value even if the Maker is not tilted. To avoid this, you will need to measure this offset and compensate for it. A simple, yet effective method of finding the offset is to add or subtract a number from the measurement, and check if this solves the issue. You can keep tuning the offset until you cancel its effect.

The calibration of this sensor is out of the scope of this guide. For more details on how to calibrate the sensor, you can visit the Maker portal (maker.ibtikar.io). In the following examples, the offset values are found using the method explained on the Maker portal.

Another important topic is the sensitivity of the sensor. What if you want the board to detect the tilt after a certain value? This means when you tilt the board, you can create a threshold value, so if you exceed it, you are sure that the Maker is tilted as shown.



Activity 23: Single Axis Tilt Detection

In this activity, you will create a program to detect the tilt in the X-axis direction. You will show the direction as an arrow (East or West) using the LED grid.

In the following activity, you need to first enable the accelerometer sensor, then define a threshold variable and set it to a small value, say 25. Since the accelerometer returns float values, you need to define the variable as float too.

In the main loop, read the sensor in the required direction. If there is an offset, you will need to compensate for it. The offset in the x axis is found to be around 34. The value may differ slightly for your board.

Once the value is read and the offset is compensated for, you need to compare the measured value with the threshold value. If the value is greater than the positive threshold value, draw the east arrow. If it is smaller than the negative threshold value, draw the west arrow. Otherwise, clear the LED grid to indicate that there is no tilt. Try the next program and upload it to your Maker.

```
Arduino run first:
Enable Acceleration Sensor TRUE
set threshold to 25 as Decimal

Arduino loop forever:
set x to Read Acceleration of Axis X +- 33.5
if x > threshold
do Draw Arrow East
else if x < -1 * threshold
do Draw Arrow West
else Clear LEDs
wait 100 milliseconds
```

Activity 24: Multiple Axis Tilt Detection (4 Directions)

In this activity, you will create a program to detect the tilt in the X and Y axes. You will show the direction as an arrow (North, East, South or West) using the LED grid. Like the previous activity, you will create a threshold value so if you exceed it, you are sure that the Maker is tilted. This threshold will be the same for both directions.

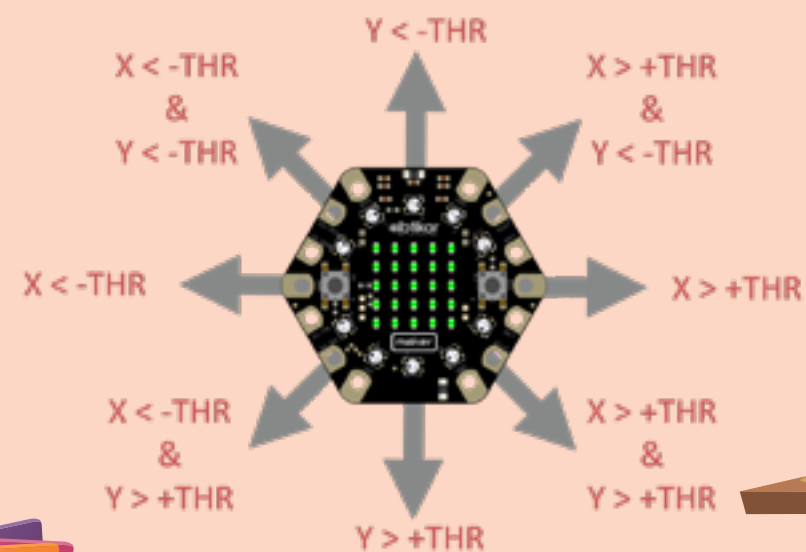
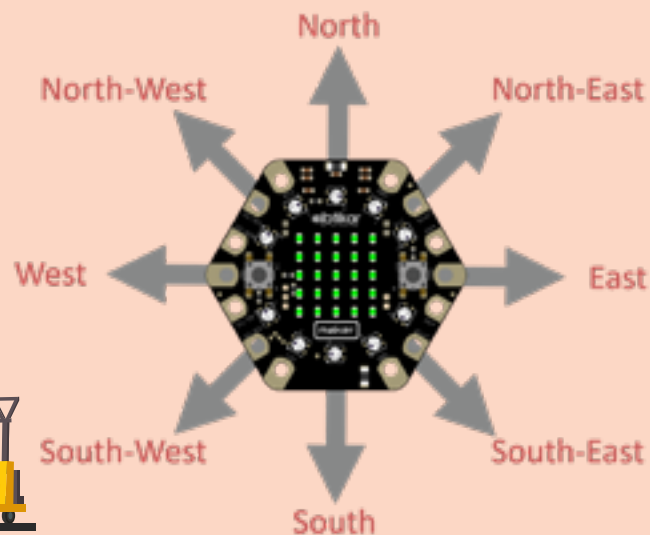
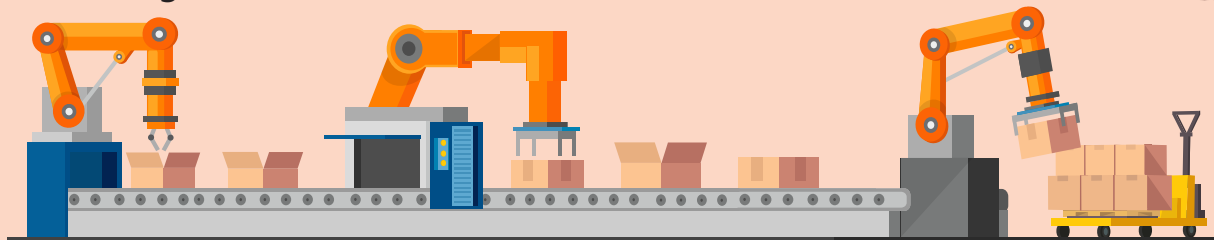
The offset in the x axis is found to be around 34 and the offset in the y axis is found to be 85. These values could be different for your board. Try the next program and upload it to your Maker.

```
Arduino run first:
Enable Acceleration Sensor TRUE
set threshold to 25 as Decimal

Arduino loop forever:
set x to Read Acceleration of Axis X +- 33.5
set y to Read Acceleration of Axis Y +- 85
if x > threshold
do Draw Arrow East
wait 100 milliseconds
else if x < -1 * threshold
do Draw Arrow West
wait 100 milliseconds
else Clear LEDs
if y > threshold
do Draw Arrow South
wait 100 milliseconds
else if y < -1 * threshold
do Draw Arrow North
wait 100 milliseconds
else Clear LEDs
```


Activity 25: Multiple Axis Tilt Detection (8 Directions)

In this activity, you will create a program to detect the tilt in the X and Y axes. You will show the direction as an arrow (North, North-East, East, South-East, South, South-West, West or North-West) using the LED grid.



Like the previous activity, you will create a threshold value so if you exceed it, you are sure that the Maker is tilted. This threshold will be the same for both directions. The offset is the same for both axes as in the previous activity.



Your code will look like the following.

```

Arduino run first:
  Enable Acceleration Sensor TRUE
  set threshold to 25 as Decimal
  Arduino loop forever:
    Clear LEDs
    set x to 33.5
    set y to 85
    if x > threshold
      do Draw Arrow South East
    else if y < -1
      do Draw Arrow North East
    else
      do Draw Arrow East
    else if x < -1
      do Draw Arrow South West
    else if y > threshold
      do Draw Arrow North West
    else
      do Draw Arrow South
    else if y < -1
      do Draw Arrow North
    else
      do Clear LEDs
    wait 100 milliseconds
  
```

Try the previous program and upload it to your Maker.





TM
maker



ibtikar®
EDU TECH SOLUTIONS