Magicblocks

unknown

Feb 15, 2023

CONTENTS

The Internet of Things (IoT) is a network of physical devices, vehicles, buildings, and other items that are embedded with sensors, software, and connectivity, enabling them to collect and exchange data. It allows everyday objects to be connected to the internet and communicate with each other, creating smart systems that can be controlled and monitored remotely. Examples of IoT devices include smart thermostats, refrigerators, door locks, and cameras. Additionally, IoT is used in various industries such as transportation and manufacturing to track vehicles, optimize routes, and monitor machinery performance. Having an easy way to learn and develop IoT applications is crucial for both students and industry experts. It allows them to quickly prototype and test their ideas, saving time and resources. User-friendly tools such as dashboards make it easier to understand and analyze data from IoT devices, helping to make better decisions and improve the overall performance of the systems. For students, learning about and experimenting with IoT technology can provide hands-on experience and skills that are in high demand in the job market. For industry experts, easy-to-use tools can help them stay competitive in the market and quickly adapt to new technologies.

CHAPTER

ONE

WHAT IS MAGICBLOCKS.IO

magicbicblocks.io is an easy-to-use IoT platform that supports ESP32-based development boards and utilizes Node-RED, a visual programming tool that allows for drag-and-drop connection of different IoT devices and services, eliminating the need for complex coding. The platform offers pre-built blocks and sample applications, supports a variety of sensors and actuators, and provides cloud-based data storage and remote access. Overall, magicbicblocks.io is a powerful tool for anyone looking to quickly prototype and develop IoT applications.

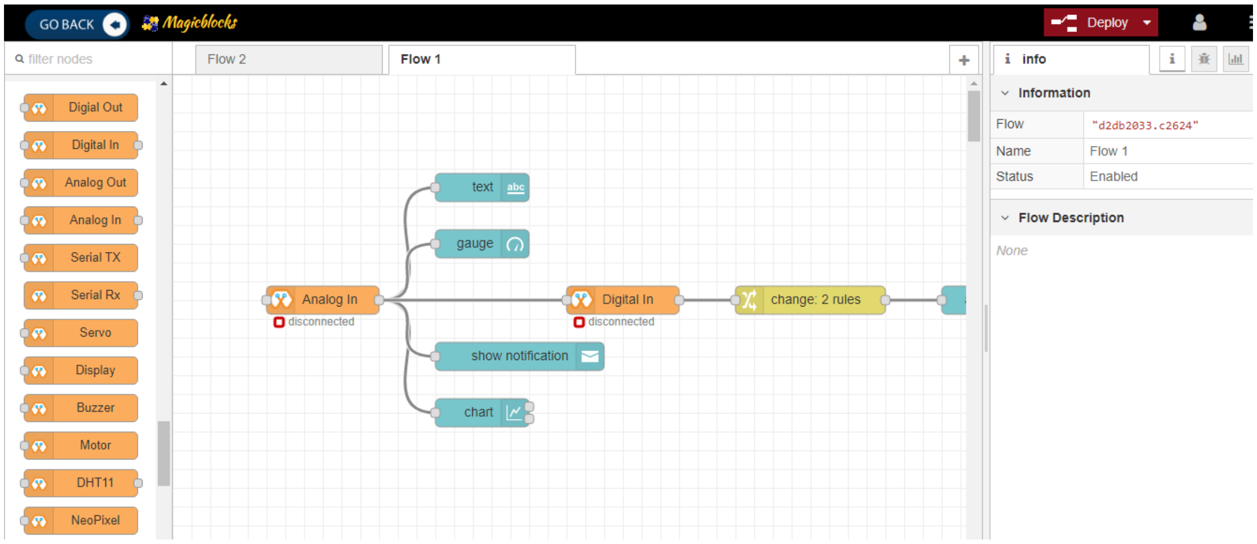
1.1 How Magicblocks works

The three main components of IoT are devices, cloud, and connectivity. Devices are the physical objects that are embedded with sensors and software that collect and exchange data. Cloud refers to the remote servers where this data is stored and analyzed. Connectivity is the means by which these devices and cloud services communicate, such as Wi-Fi, Bluetooth, or cellular networks. In a system using magicblocks, it acts as the cloud component, where data is stored and analyzed. The devices component is fulfilled by using ESP32-based hardware which are embedded with sensors and software. The connectivity method used is WiFi, which enables the communication between the devices and the cloud. Together, these components make it possible to create a smart system that can be monitored and controlled remotely.

1.2 Magicblocks.io Playground

Any ioT creation you want to connect with nodes without any coding and your project running in the cloud platform the development board you use will not have much memory or more processing capability, unless you want to create Image Processing, Character Recognizing, DB Handling, you can create any project on it.

You can also create attractively online dashboards online without any coding



Information

Flow	"d2db2033.c2624"
Name	Flow 1
Status	Enabled

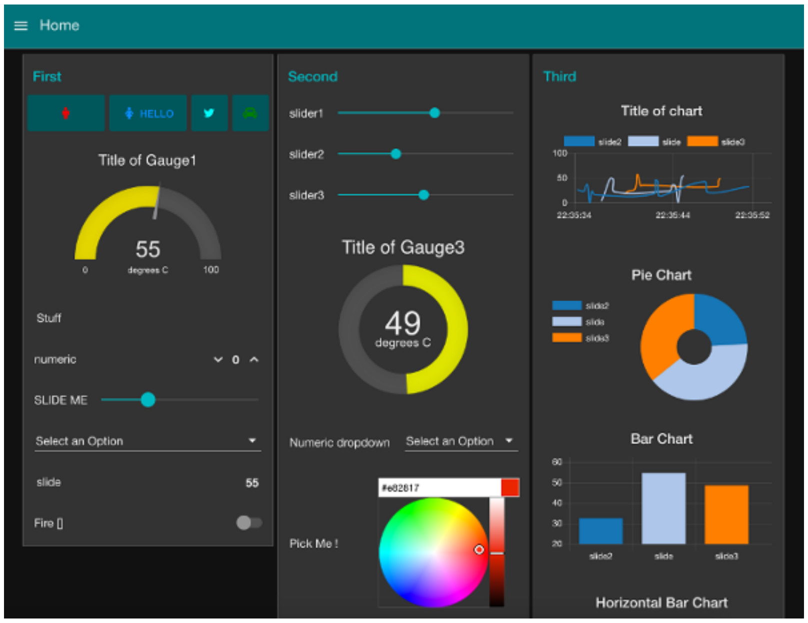
Flow Description

None

1.3 Magicblocks.io Dashboard

In Magicblocks you can also create Web based-dashboards attractively for your IoT designs without any codes. You need to connect the necessary widgets to the dashboard that you want and to configure the settings you need to reach them. Then, you can access your Online Dashboard from the dashboard menu on your dashboard, where you can click on the URL in the browser to access the Dashboard via any Device Device such as Smart Phone, Tablet, PC. Any ioT creation you want toe connect with nodes without any coding and your project running in the cloud platform the development board you use will not have much memory or more processing capability, unless you want to create Image Processing, Character Recognizing, DB Handling, you can create any project on it.

You can also create attractively online dashboards online without any coding



CHAPTER

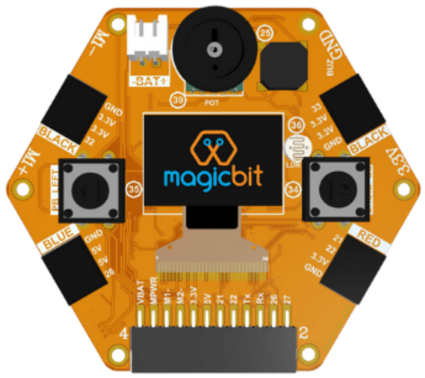
TWO

HARDWARE

The hardware or device component in IoT refers to the physical objects embedded with sensors, software, and connectivity that collect and exchange data, creating smart systems that can be controlled and monitored remotely.

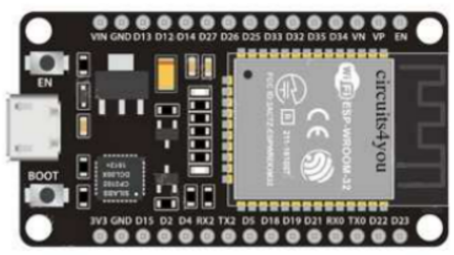
We use ESP32 based hardware for Magicblocks.io. ESP32 is a powerful and versatile microcontroller that is well-suited for IoT projects. Its combination of low cost, small form factor, and built-in wireless capabilities make it an ideal choice for a wide range of applications. And the availability of development boards makes it easy for developers to prototype and test their ideas. Using ESP32 for IoT projects is a great choice for several reasons:

- It is low cost, yet powerful enough for many IoT applications.
- It has built-in wireless capabilities, eliminating the need for separate modules.
- It has a small form factor, making it easy to integrate into a variety of devices.
- It has a large and active community of developers, providing a wealth of resources and support.

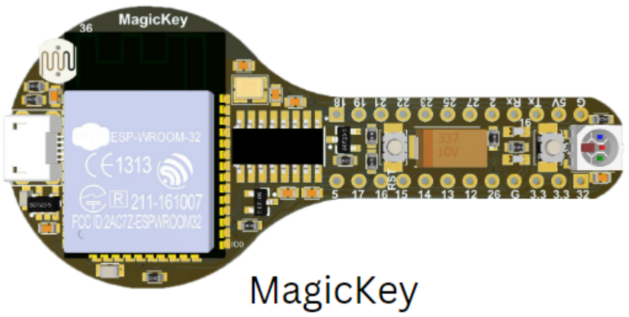


Magicbit

M5 Stack



ESP32 Dev Kit



MagicKey

2.1 ESP32 Dev Kit

ESP32 Dev Kit is a compact development board based on the ESP32 microcontroller, designed for easy prototyping and programming. It has an on-board USB-to-Serial converter and built-in support for a wide range of sensors and actuators. It is a popular development board for IoT projects.

2.2 Magicbit

Magicbit is an ESP32-based development board that features in-built battery charger, WiFi & Bluetooth connectivity, integrated sensors and actuators, an internal OLED display, and a plug & play feature for easy connection of accessories, enabling users to easily test and design projects.

2.3 MagicKey

MagicKey is a breadboard compatible ESP32 based development board with on board LED, push button, LDR and RGB neo pixel. It support all software as same as Magicbit.

2.4 M5 Stack

M5Stack is a line of modular, stackable, and portable development boards based on the ESP32 microcontroller, which are often used for IoT projects and also include additional modules for more functionality.

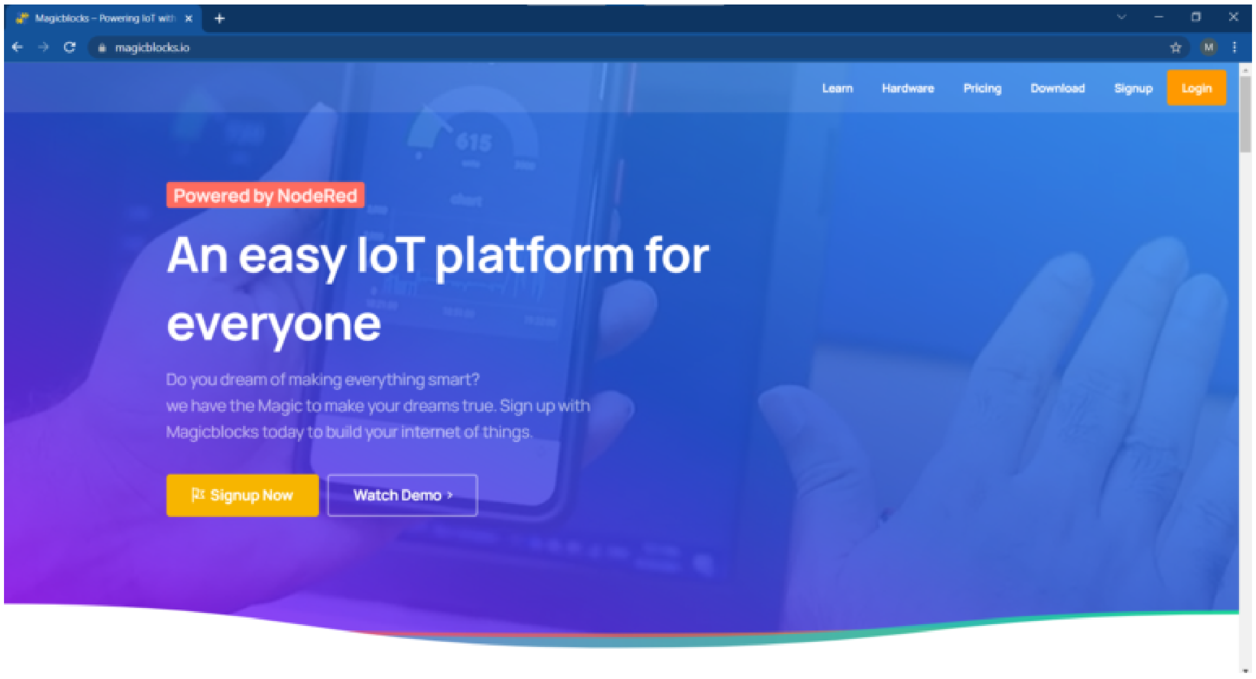
CHAPTER

THREE

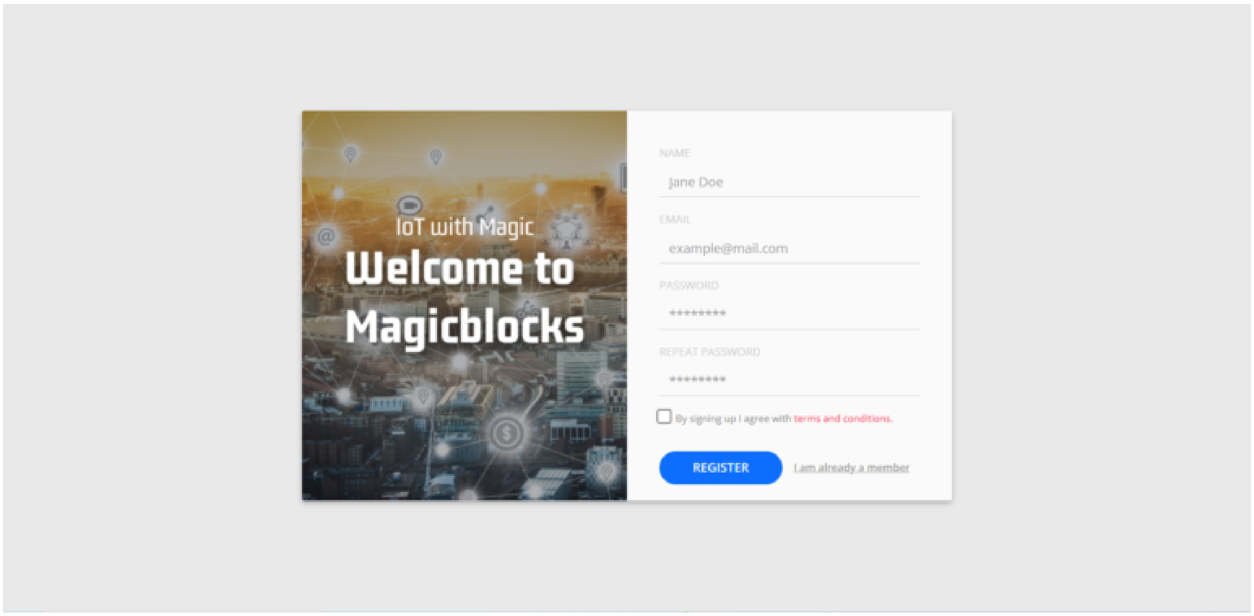
GETTING STARTED

3.1 Create Magicblocks.io Account

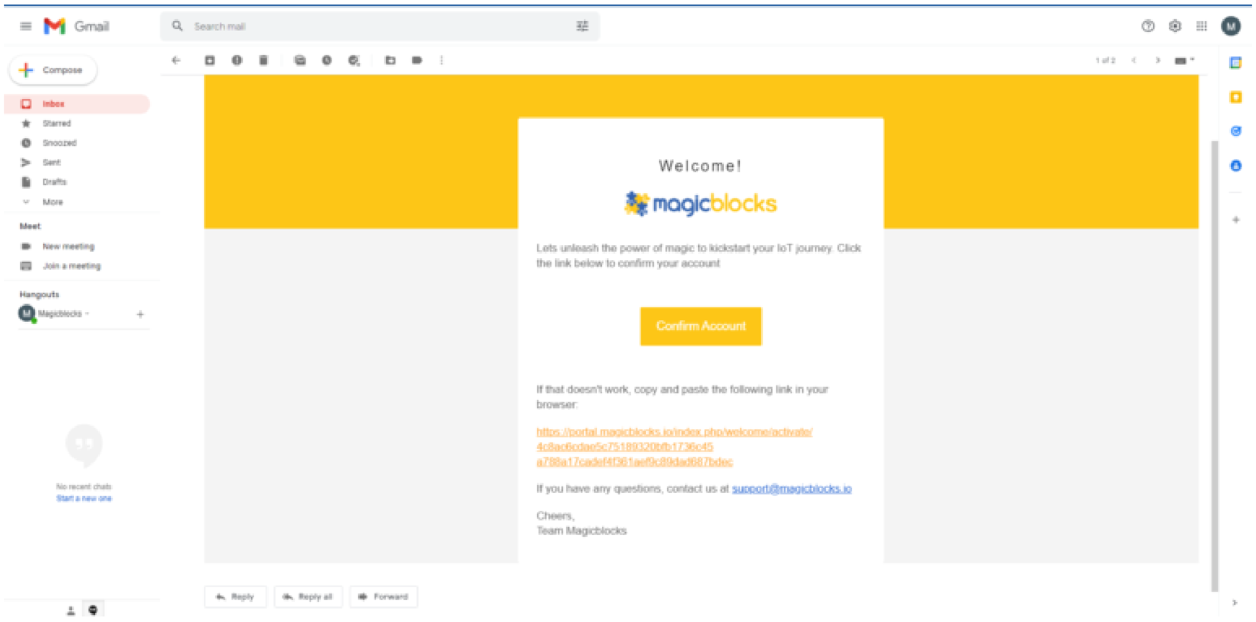
- Go to magicblocks website https://magicblocks.io/
- Select SIGNUP


Magicblocks – Powering IoT with
magicblocks.io
Learn
Hardware
Pricing
Download
Signup
Login
Powered by NodeRed
An easy IoT platform for everyone
Do you dream of making everything smart?
we have the Magic to make your dreams true. Sign up with Magicblocks today to build your internet of things.
Signup Now
Watch Demo ›

- Click on the SIGNUP button after inserted your details



- Go to the email account you provided and activate your Magicblocks account with the Activation Link
- Follow the Activation Link which we have sent you as an email.



Welcome!

magicblocks

Lets unleash the power of magic to kickstart your IoT journey. Click the link below to confirm your account

Confirm Account

If that doesn't work, copy and paste the following link in your browser:

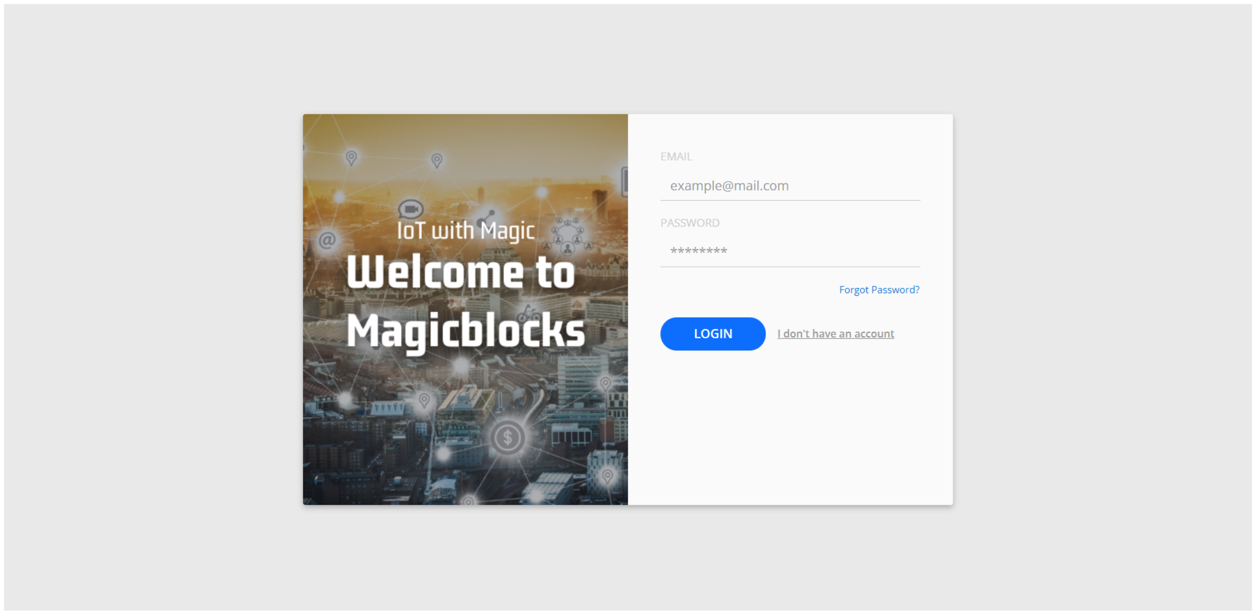
https://portal.magicblocks.io/index.php/welcome/activate/4c8ac6cdae5c75189320bfb1736c45a788a17cadef4f361aef9c89dad687bdec

If you have any questions, contact us at support@magicblocks.io

Cheers,
Team Magicblocks

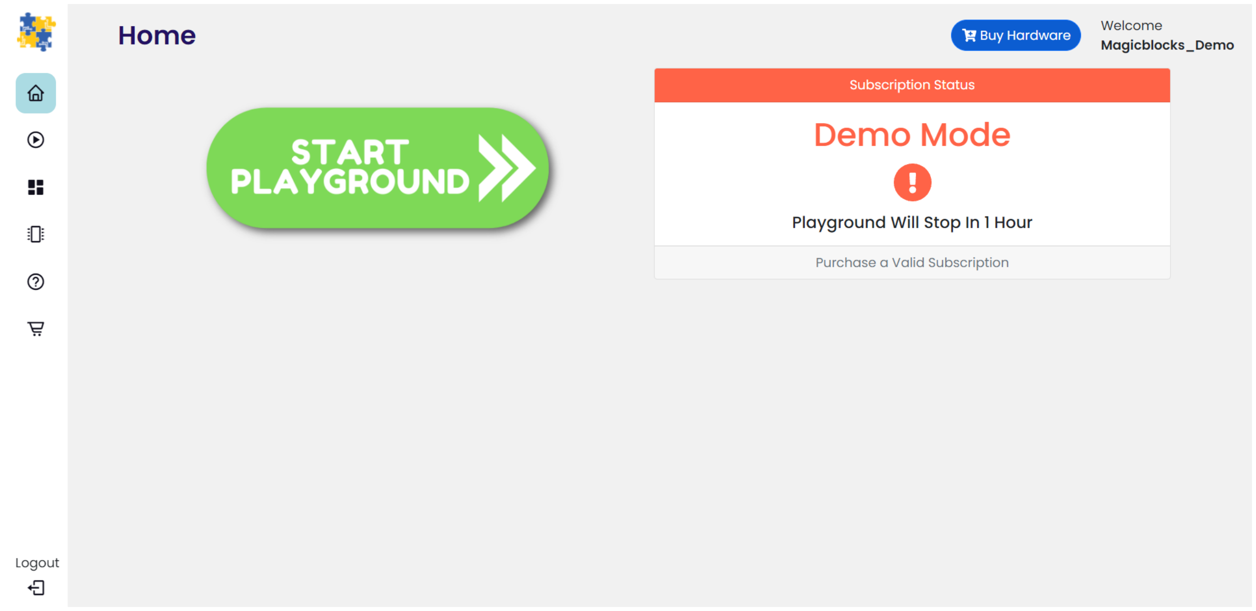
3.2 Login to Your Magicblocks.io Account

- Go to Magicblocks.io official website. https://magicblocks.io/
- Select LOGIN
- Provide your login details.
- Enter your email address and the Magicblocks Password and sign in to magicblocks



3.3 Start the Playground

The playground is the visual programming environment based on Node-Red that has been customized for seamless integration with hardware devices to enable IoT. When you log in for the first time, your playground will not be running. If you do not have a valid subscription, you will be allowed to run the playground only for 1 hour continuously before it is automatically stopped. You will need to wait for 1hour to start the playground again. You can subscribe to the Standard Subscription by entering the coupon code in the Subscription tab provided with your Magicbit device. If you have any issue please write to info@magicbit.cc

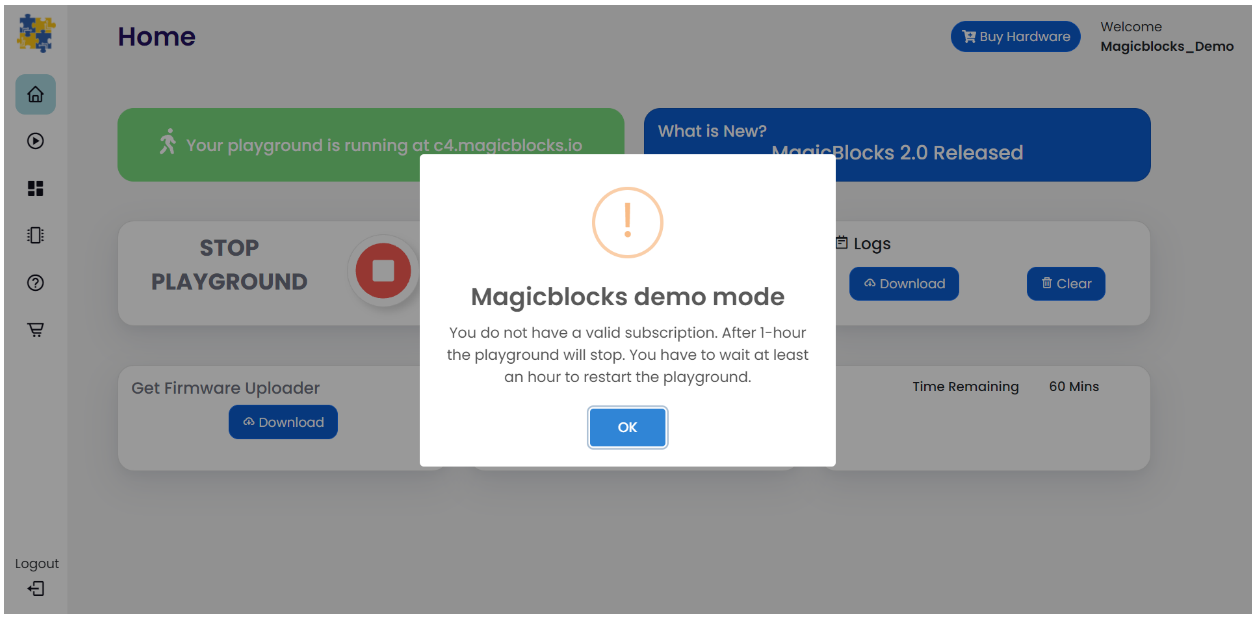


Subscription Status

Demo Mode

Playground Will Stop In 1 Hour

Purchase a Valid Subscription



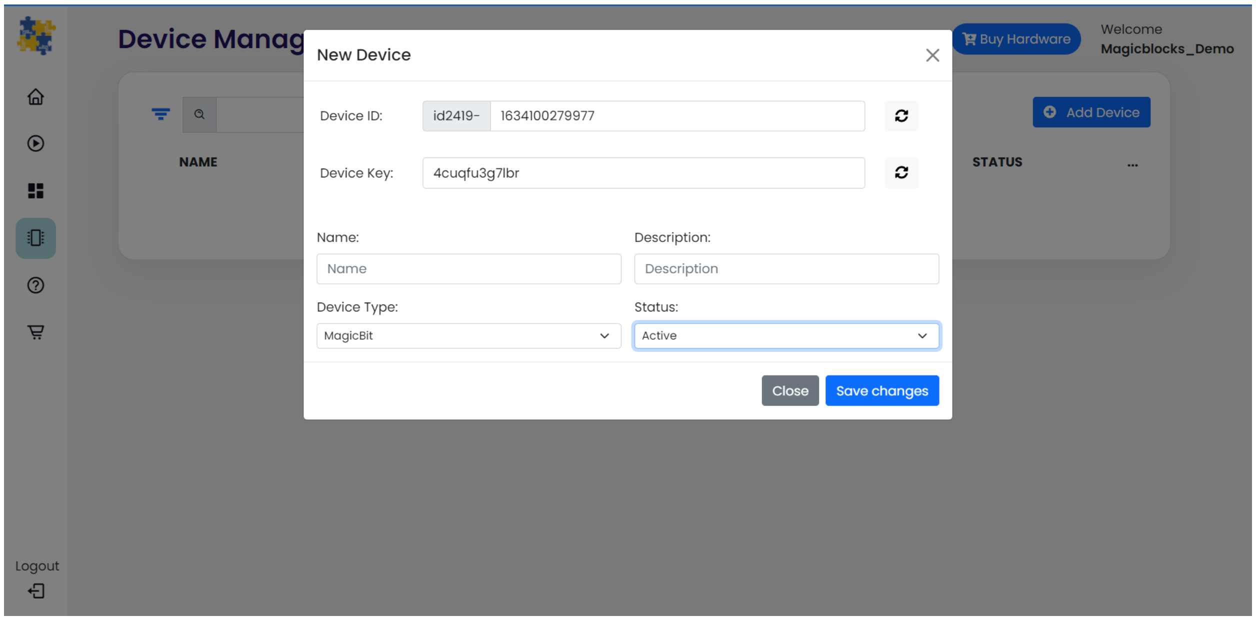
Magicblocks demo mode

You do not have a valid subscription. After 1-hour the playground will stop. You have to wait at least an hour to restart the playground.

OK

3.4 Create a new device

Go to Device Manager and add a new device. Select your **device type** & keep the status as **Active**. You can use any name and description. Click save changes to finish.



New Device

Device ID: id2419- 1634100279977

Device Key: 4cuqfu3g7lbr

Name: Name

Description: Description

Device Type: MagicBit

Status: Active

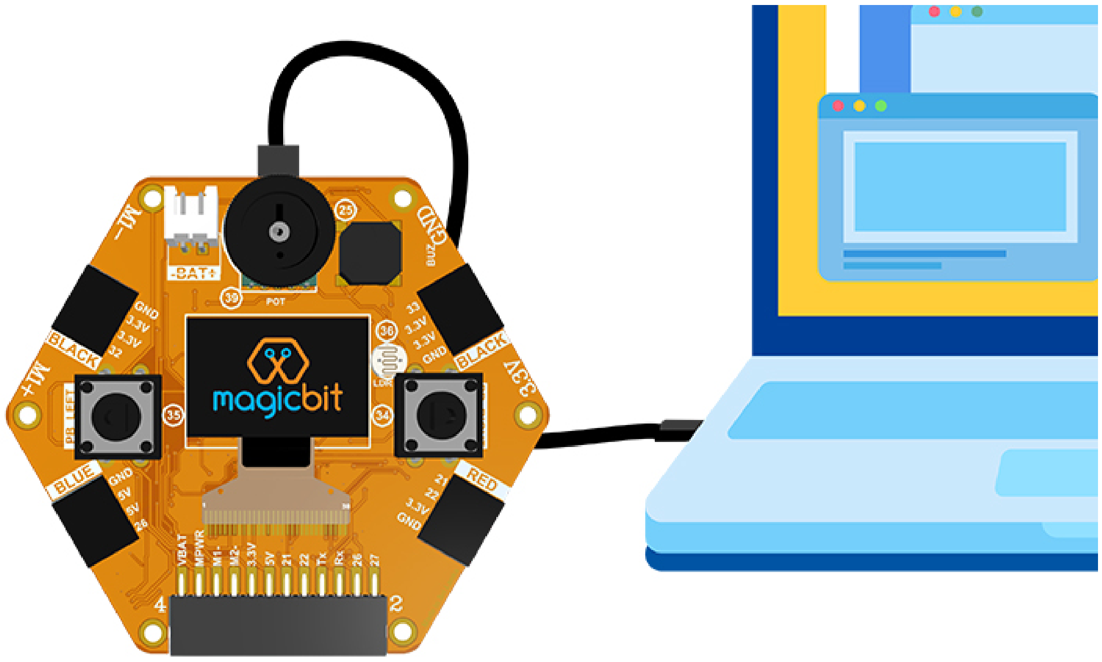
Close Save changes

3.5 Setting up a device

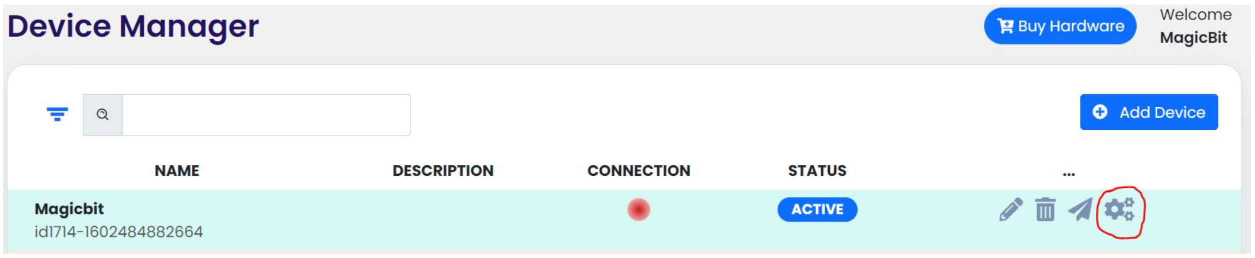
To set up your device, follow the relevant steps for your device given below.

Magicbit

Connect your device to computer using USB cable.



click on the setting icon on device manager of your device.



Device Manager

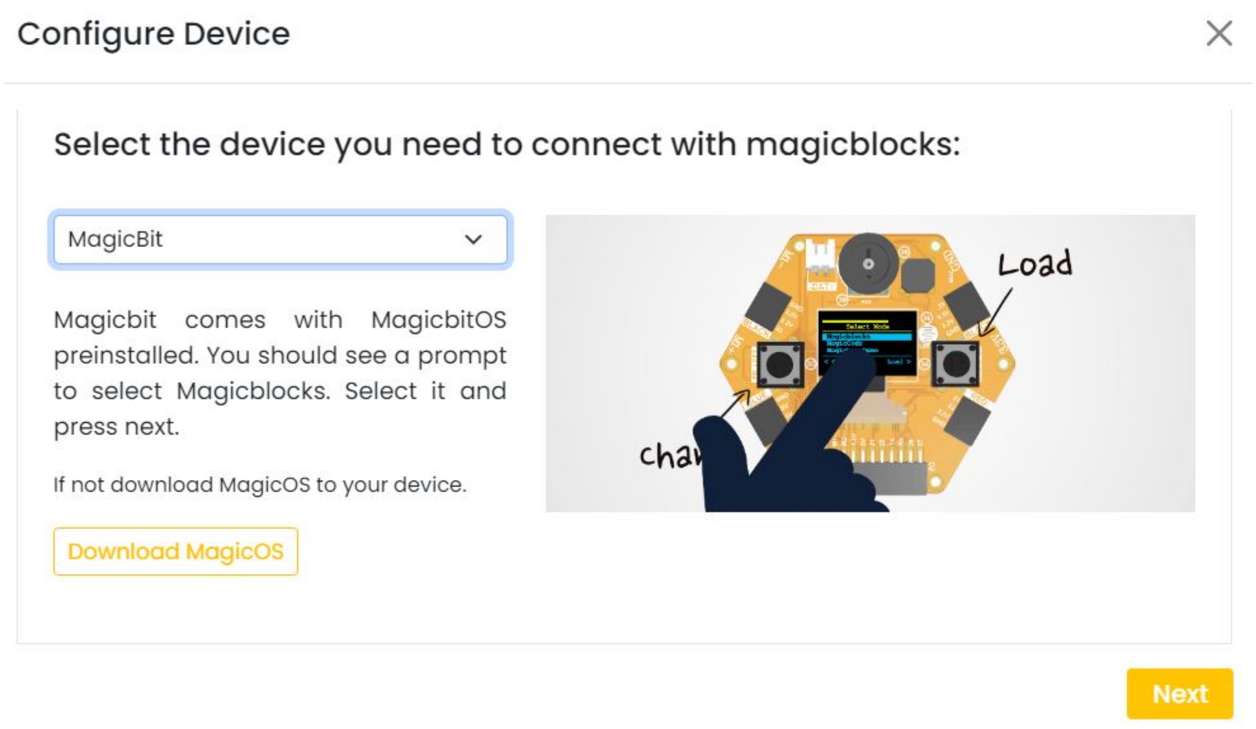
Buy Hardware

Welcome MagicBit

Add Device

NAME	DESCRIPTION	CONNECTION	STATUS	...
Magicbit id1714-1602484882664		●	ACTIVE	

Select your device and follow instructions to load magicblocks.io firmware.



Configure Device

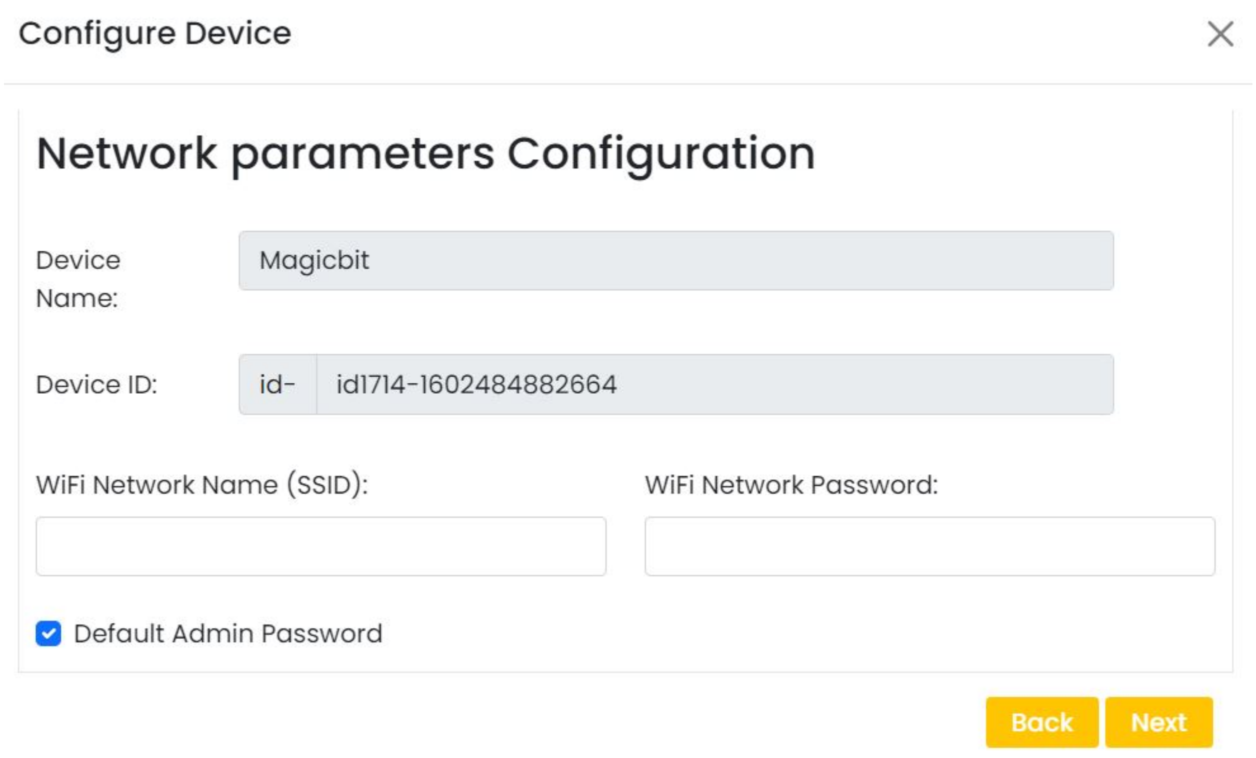
Select the device you need to connect with magicblocks:

MagicBit

Magicbit comes with MagicbitOS preinstalled. You should see a prompt to select Magicblocks. Select it and press next.

If not download MagicOS to your device.

On next window enter your internet router WiFi network name (SSID) and password



Configure Device

Network parameters Configuration

Device Name: Magicbit

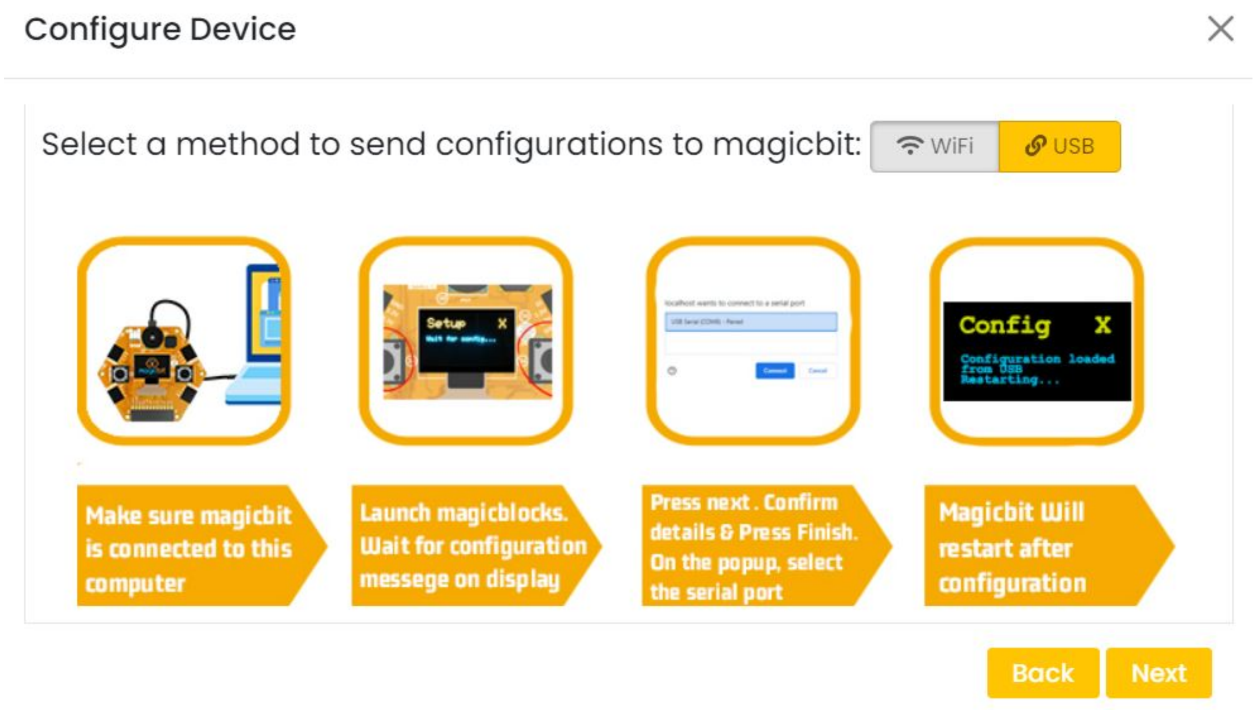
Device ID: id- id1714-1602484882664

WiFi Network Name (SSID):

WiFi Network Password:

☑ Default Admin Password

On next stage configure the device using USB method. If it didn't work you can retry again or use WIFi option to configure device. You will get a message after the successful configuration.


Configure Device

Select a method to send configurations to magicbit:

Make sure magicbit is connected to this computer

Launch magicblocks. Wait for configuration messege on display

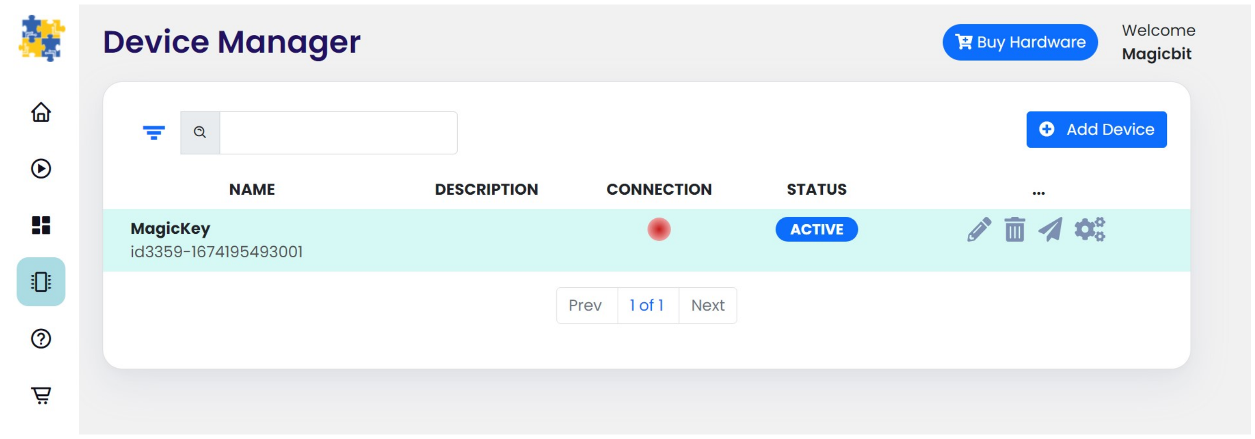
Press next. Confirm details & Press Finish. On the popup, select the serial port

Magicbit Will restart after configuration

MagicKey

Connect your device to computer using USB cable.

click on the setting icon on device manager of your device.


Device Manager

Buy Hardware

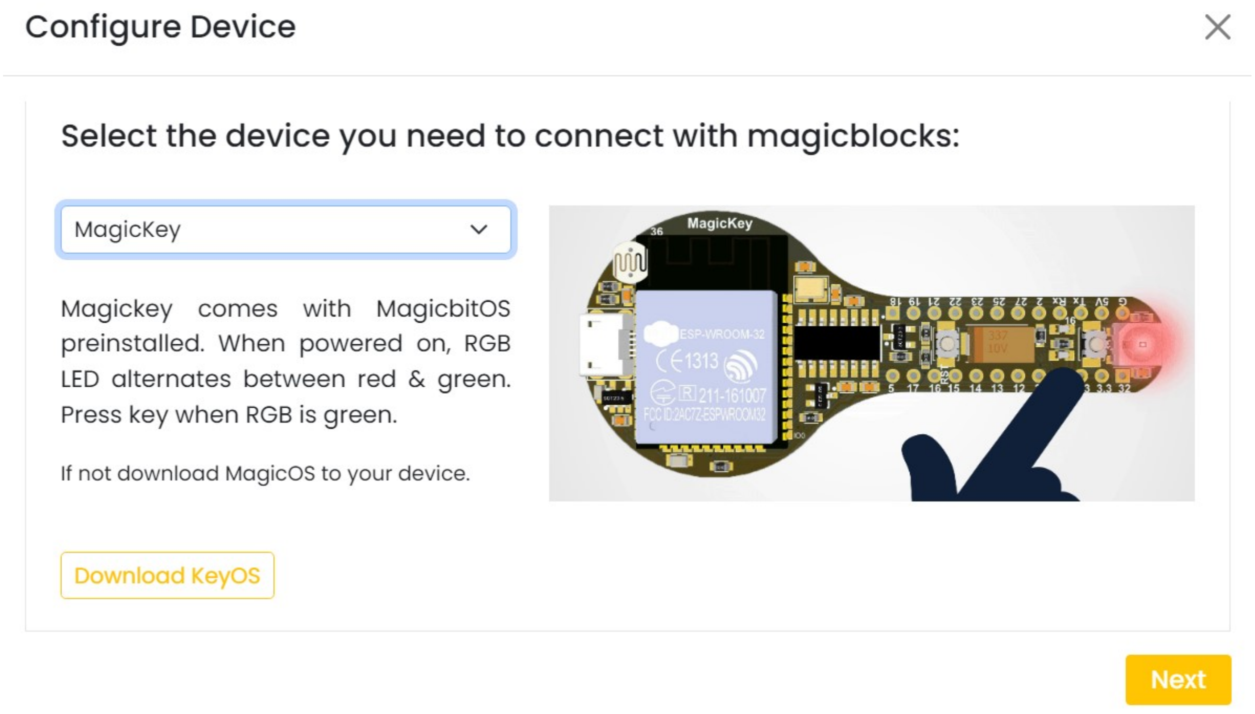
Welcome
Magicbit

Add Device

NAME	DESCRIPTION	CONNECTION	STATUS	...
MagicKey id3359-1674195493001		●	ACTIVE	

Prev 1 of 1 Next

Select your device and follow instructions to load magicblocks.io firmware.


Configure Device

Select the device you need to connect with magicblocks:

MagicKey

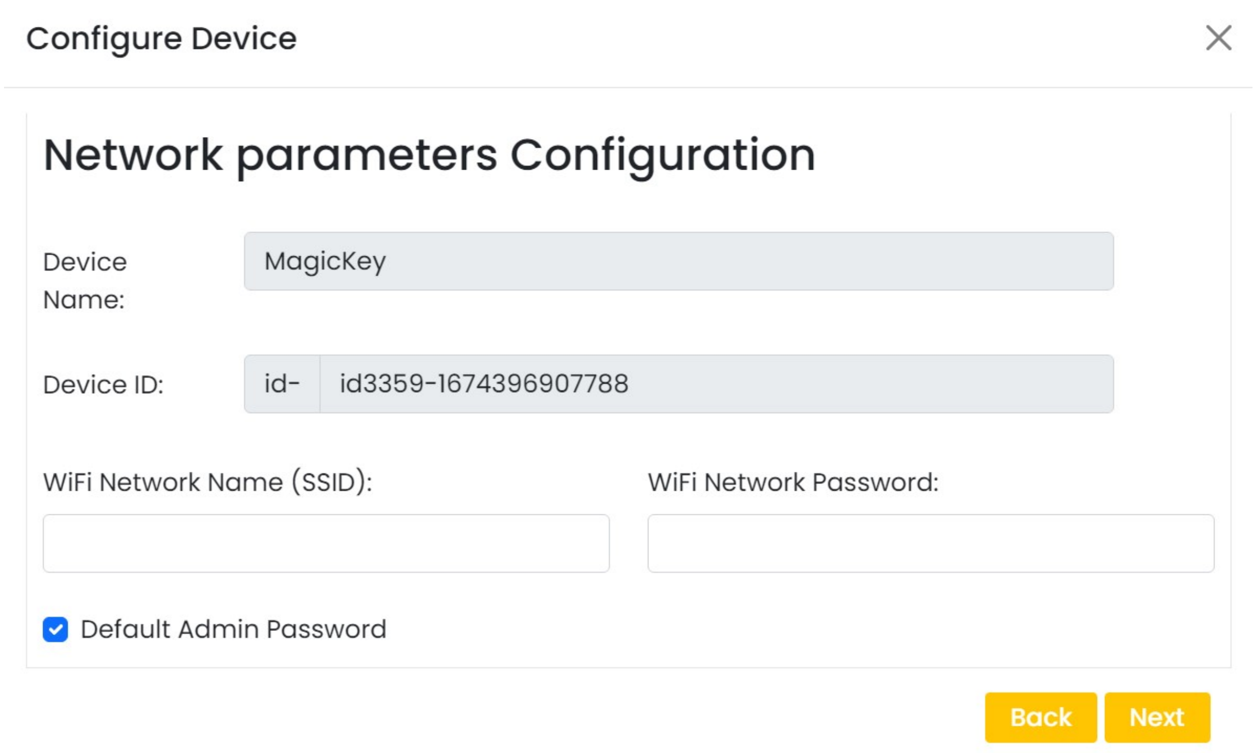
Magickey comes with MagicbitOS preinstalled. When powered on, RGB LED alternates between red & green. Press key when RGB is green.

If not download MagicOS to your device.

Download KeyOS

Next

On next window enter your internet router WiFi network name (SSID) and password



Configure Device

Network parameters Configuration

Device Name: MagicKey

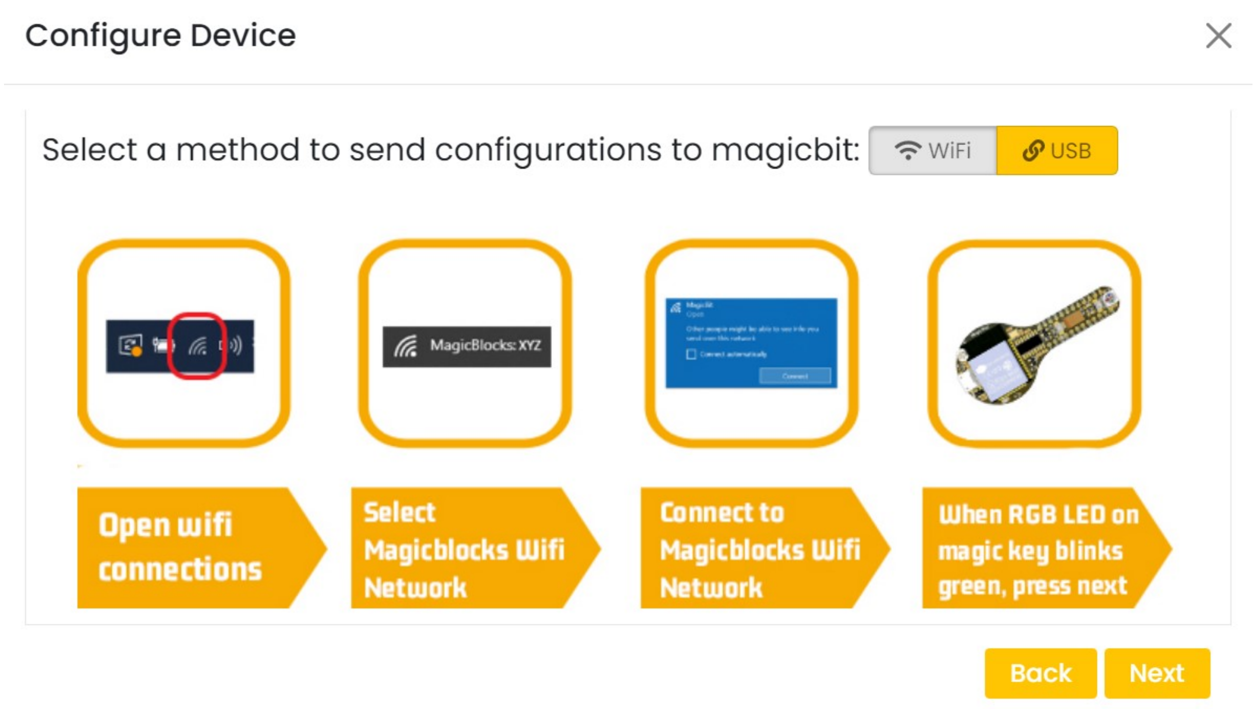
Device ID: id- id3359-1674396907788

WiFi Network Name (SSID):

WiFi Network Password:

☑ Default Admin Password

On next stage configure the device using USB method. If it didn't work you can retry again or use WIFi option to configure device. You will get a message after the successful configuration.



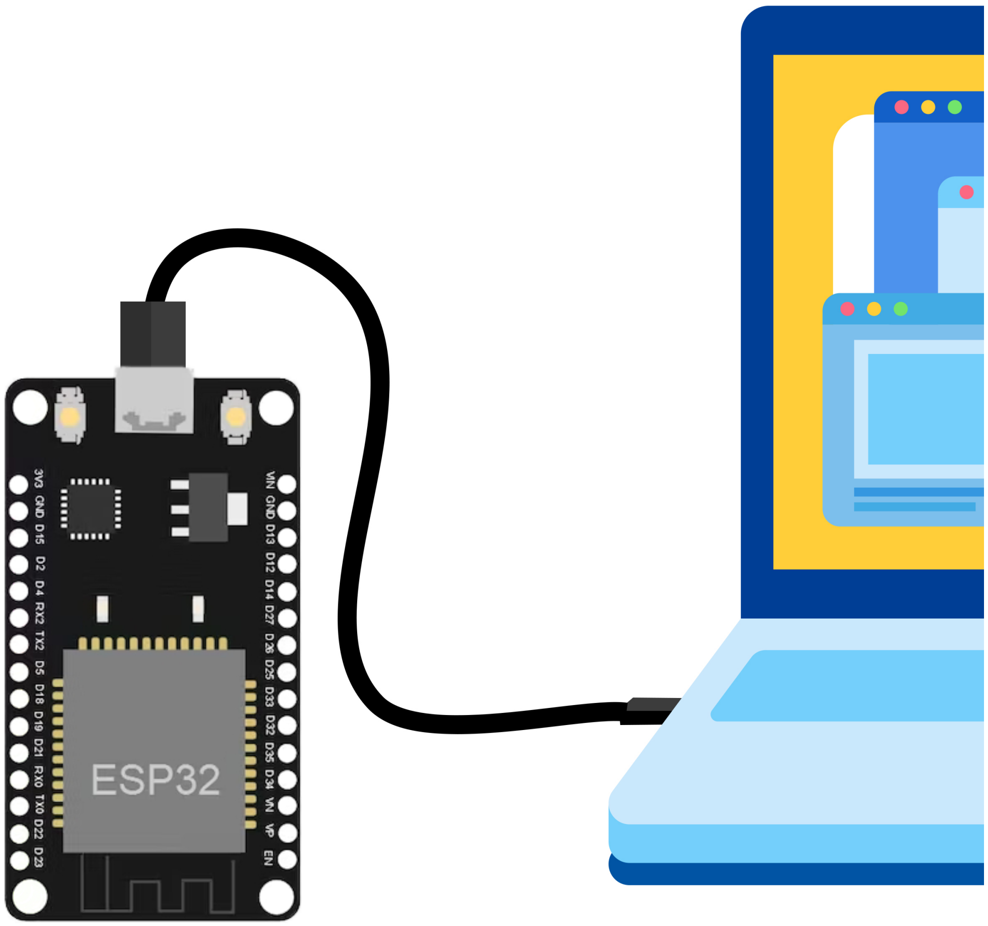
Configure Device

Select a method to send configurations to magicbit: WiFi [✓ USB]

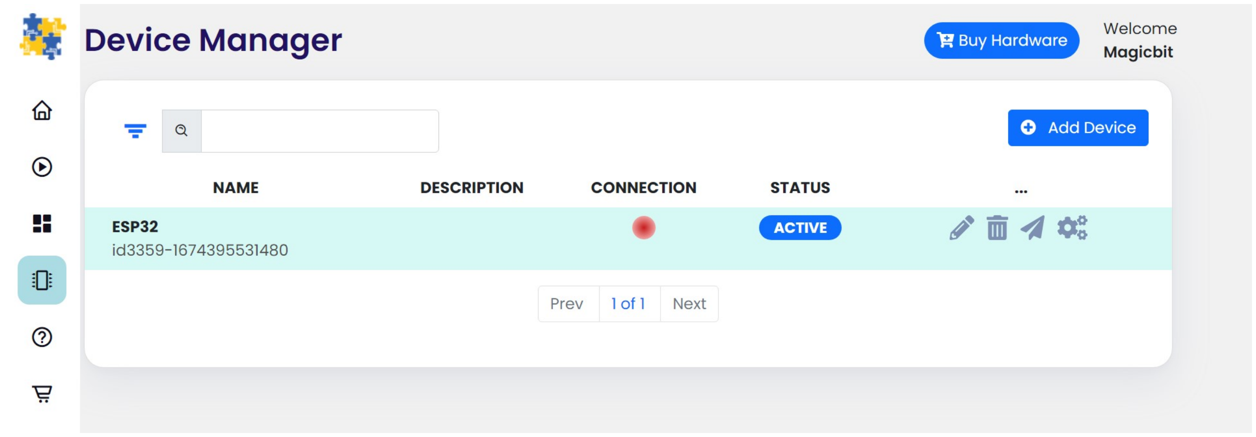
Open wifi connections
MagicBlocks:XYZ
Select Magicblocks Wifi Network
Connect to Magicblocks Wifi Network
When RGB LED on magic key blinks green, press next
Back
Next

Generic ESP32

Connect your device to computer using USB cable.

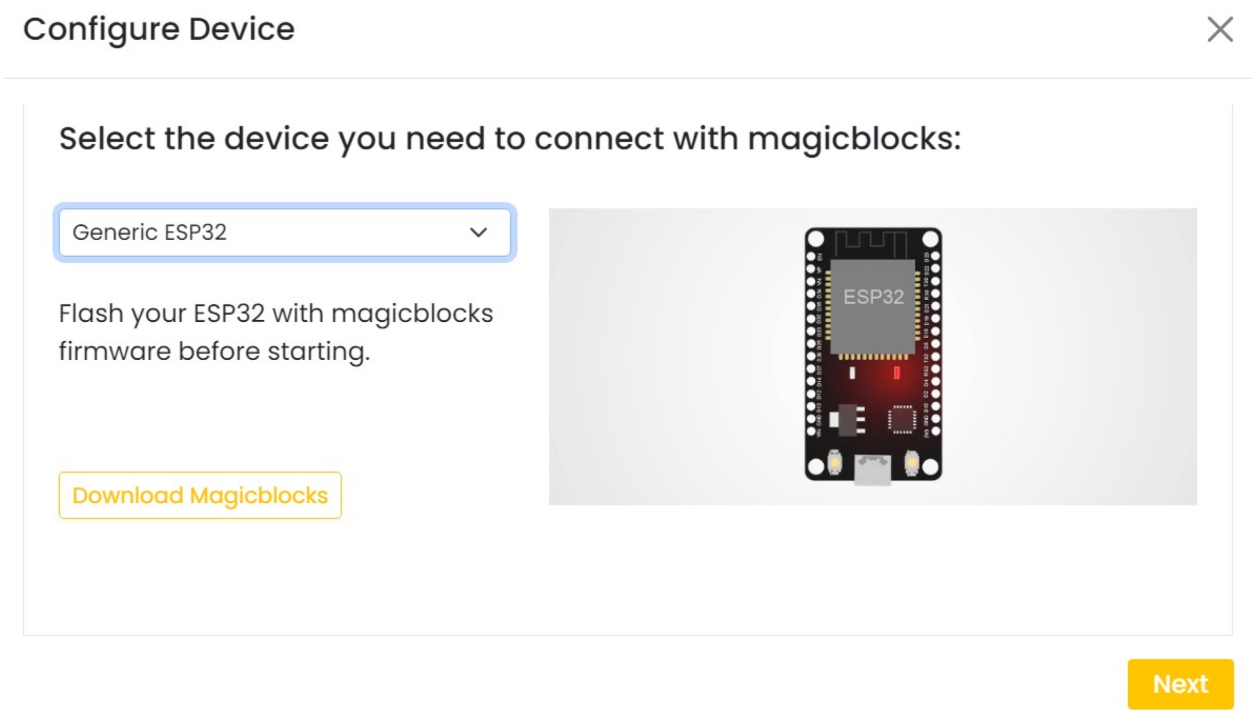

ESP32
3V3 GND D15 D2 D4 RX2 TX2 D5 D18 D19 D21 RX0 TX0 D22 D23
VIN GND D13 D12 D14 D27 D26 D25 D33 D32 D35 D34 VN VP EN

click on the setting icon on device manager of your device.



NAME	DESCRIPTION	CONNECTION	STATUS	...
ESP32 id3359-1674395531480		●	ACTIVE	

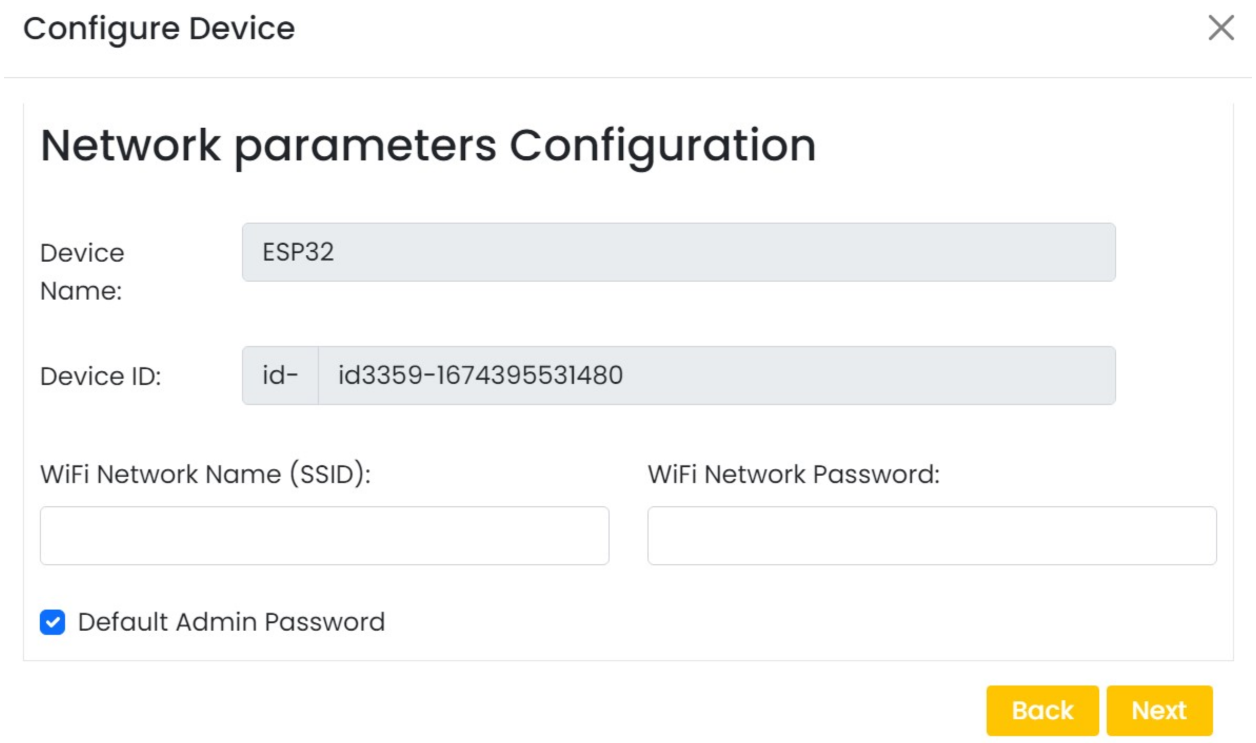
Select your device and follow instructions to load magicblocks.io firmware.



Select the device you need to connect with magicblocks:

Flash your ESP32 with magicblocks firmware before starting.

On next window enter your internet router WiFi network name (SSID) and password



Configure Device

Network parameters Configuration

Device Name: ESP32

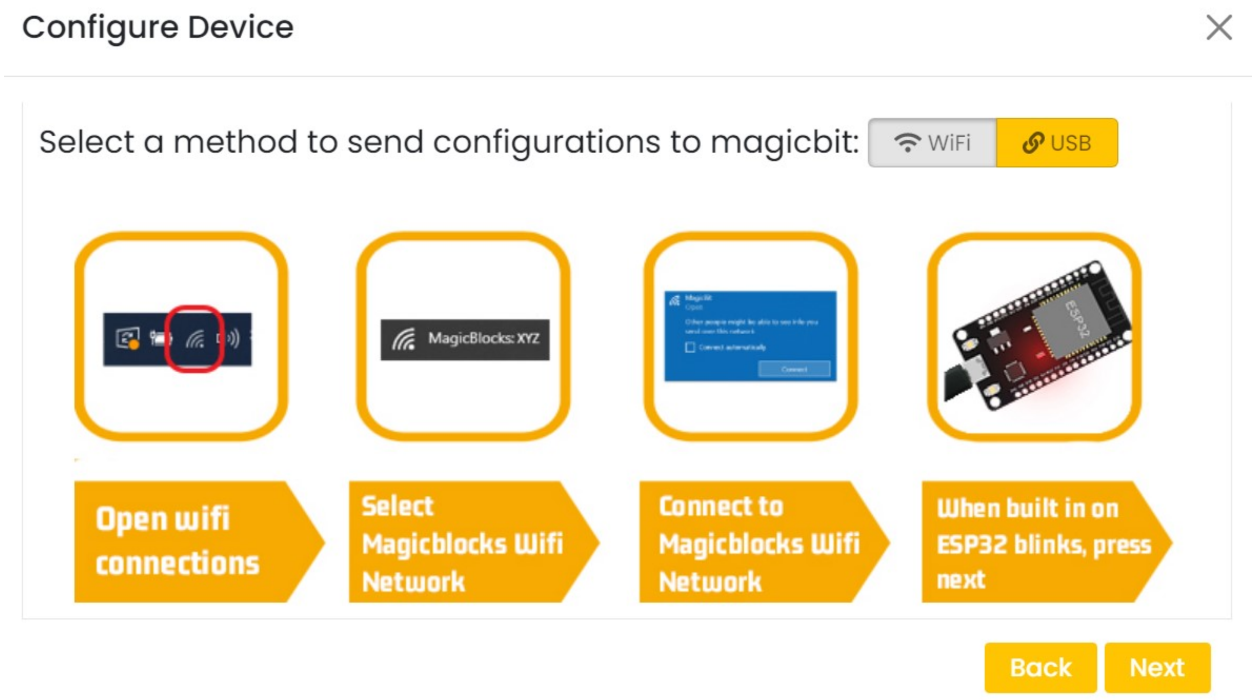
Device ID: id- id3359-1674395531480

WiFi Network Name (SSID):

WiFi Network Password:

[✓] Default Admin Password

On next stage configure the device using USB method. If it didn't work you can retry again or use WIFi option to configure device. You will get a message after the successful configuration.



Configure Device

Select a method to send configurations to magicbit: WiFi [✓ USB]

MagicBlocks: XYZ
Open wifi connections
Select Magicblocks Wifi Network
Connect to Magicblocks Wifi Network
When built in on ESP32 blinks, press next
Back
Next

CHAPTER

FOUR

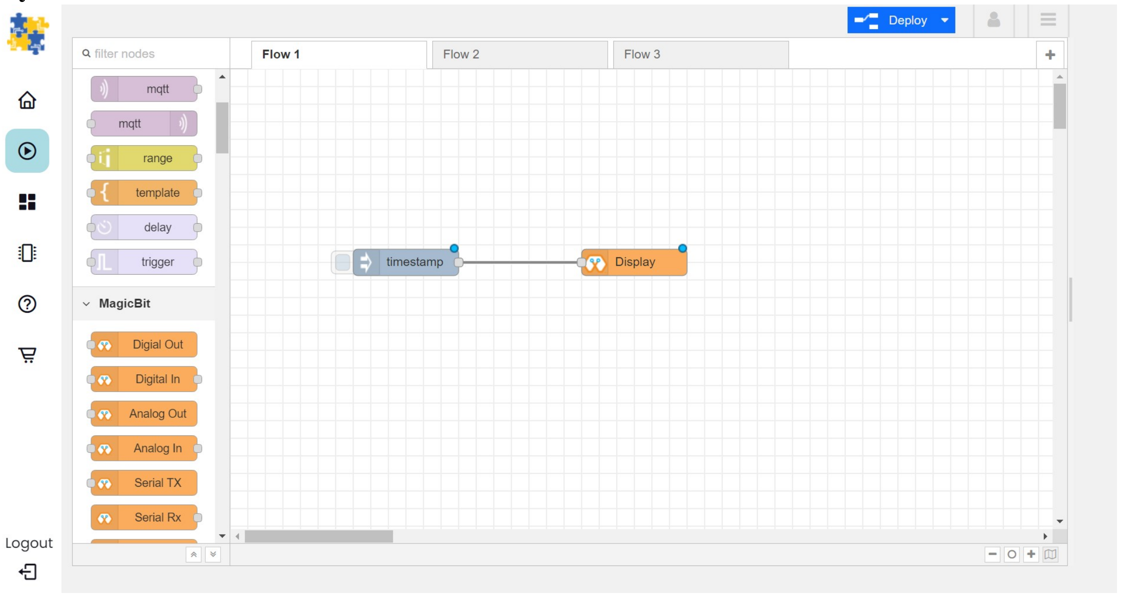
YOUR FIRST IOT PROJECT

Let's start magic with by controlling your device over the internet! Select your device and follow the instructions below to build your first IoT project with Magicblocks.

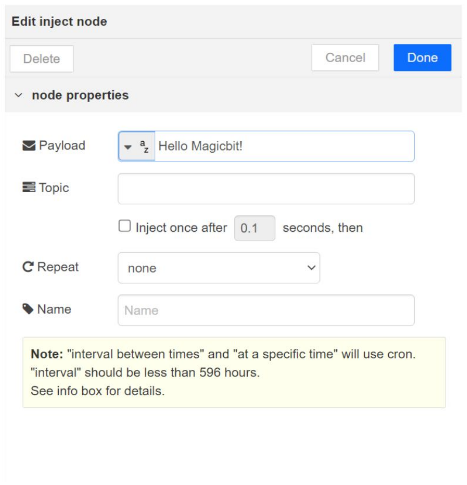
Magicbit

Let's start magic with displaying a text on Magicbit display.

- Open the Device Manager in your magicblocks account
- Plug in your Magicbit that was setup in the previous section and wait for it to connect
- Open the playground
- **Drag and drop a Display block from under the category Magicbit, and an inject block under the input category and connect them.**



- Double click on the inject node set payload type as String and type "Hello Magicbit" in the text box. You can set a topic too.
- **Double click on the Display block and Select the device Id**



Edit inject node

Delete Cancel Done

node properties

Payload: Hello Magicbit!

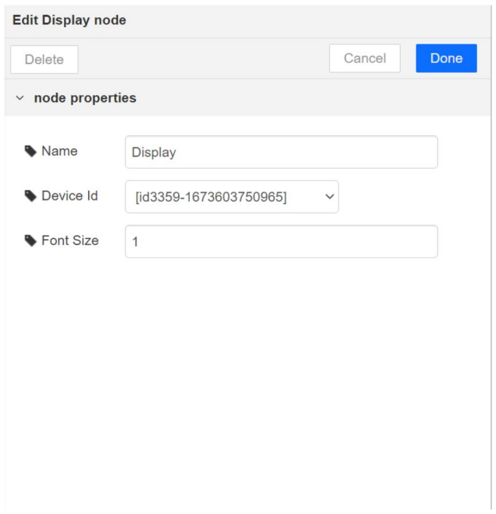
Topic:

☐ Inject once after 0.1 seconds, then

Repeat: none

Name: Name

Note: "interval between times" and "at a specific time" will use cron.
"interval" should be less than 596 hours.
See info box for details.



Edit Display node

Delete Cancel Done

node properties

Name: Display

Device Id: [id3359-1673603750965]

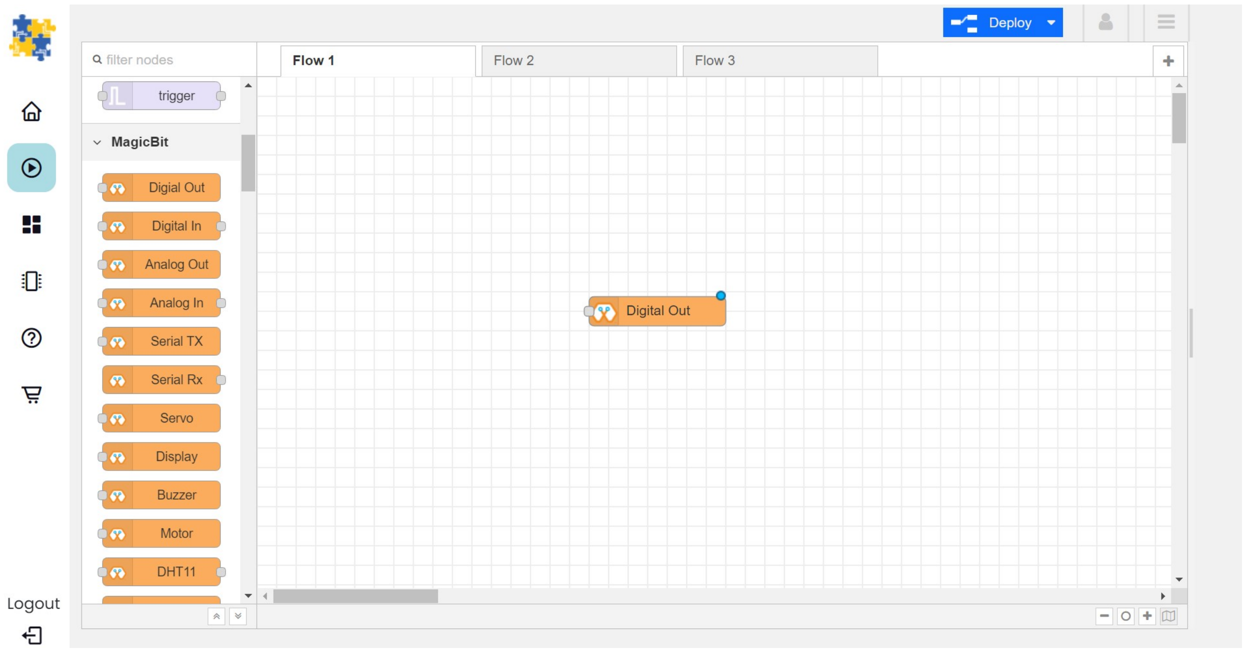
Font Size: 1

- Click deploy
- Click button in Inject node to see the magic. You can set any text from any where in the world!!!

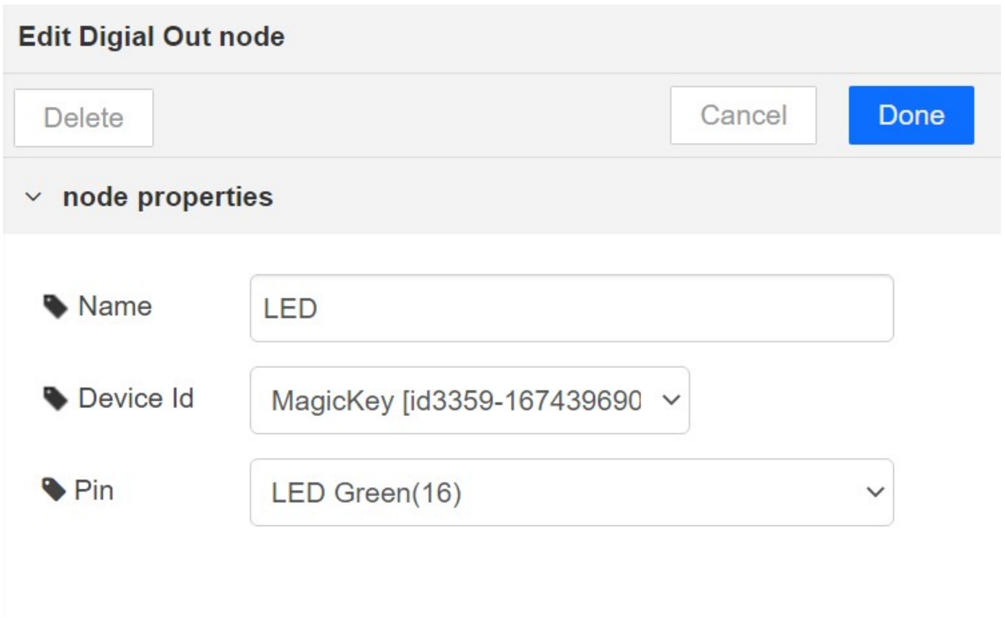
MagicKey

Let's start magic with displaying a text on Magicbit display.

- Open the Device Manager in your magicblocks account
- Plug in your MagicKey that was setup in the previous section and wait for it to connect
- Open the playground
- Drag and drop the **Digital Out** block under the category **Magicbit/ESP32**



- Double click on the **Digital Out** block and select device ID
- Set LED connected pin (Pin 16 in MagicKey)



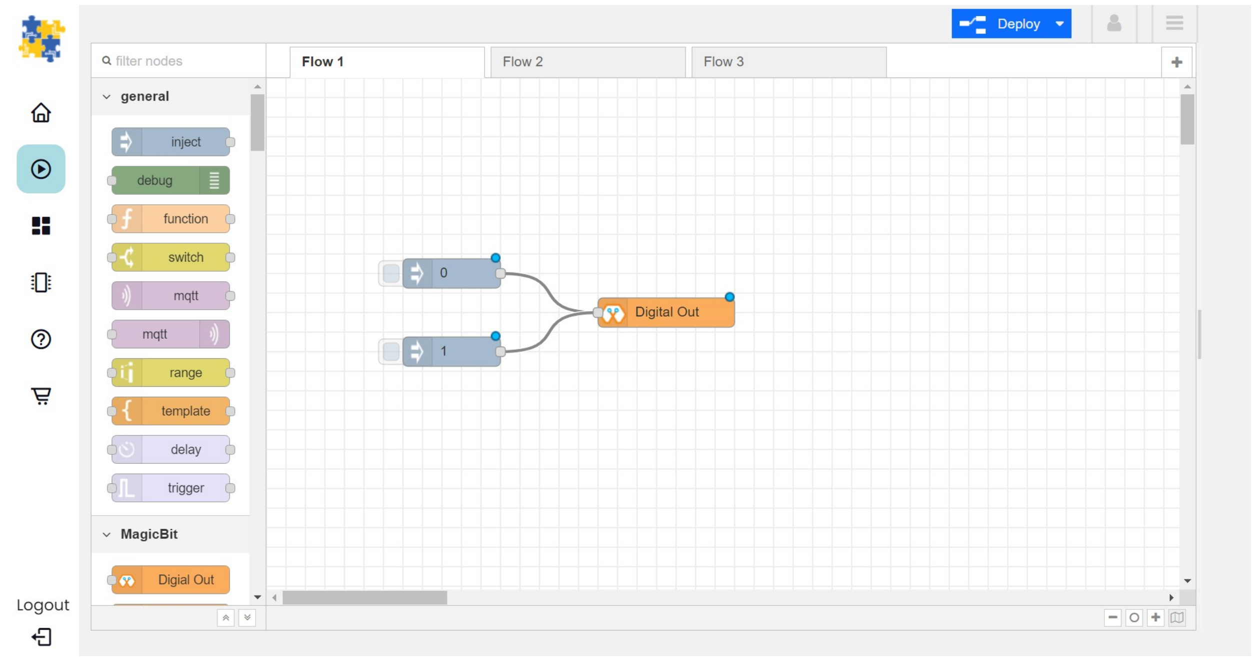
Edit Digial Out node

Delete | Cancel | Done

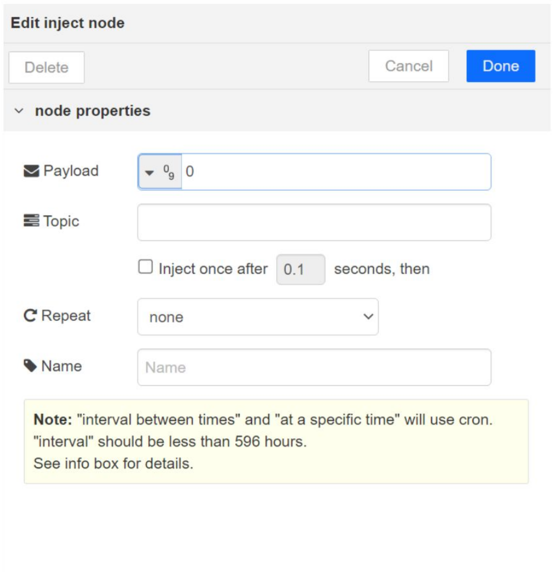
node properties

Name	LED
Device Id	MagicKey [id3359-167439690
Pin	LED Green(16)

- Drag and drop two **inject** blocks under the **input** category and connect them to the **Digital Out** block



- In one **inject** node set payload type as Number and set value to 0
- In Other **inject** node set payload type as Number and set value to 1



Edit inject node

Delete | Cancel | Done

node properties

Payload: 0

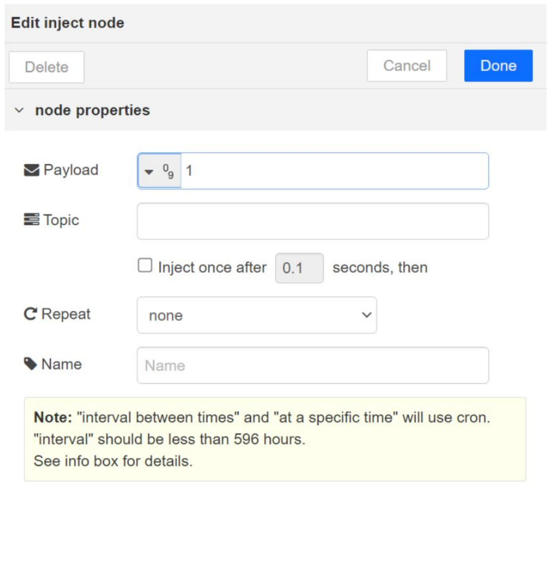
Topic:

☐ Inject once after 0.1 seconds, then

Repeat: none

Name: Name

Note: "interval between times" and "at a specific time" will use cron.
"interval" should be less than 596 hours.
See info box for details.



Edit inject node

Delete | Cancel | Done

node properties

Payload: 1

Topic:

☐ Inject once after 0.1 seconds, then

Repeat: none

Name: Name

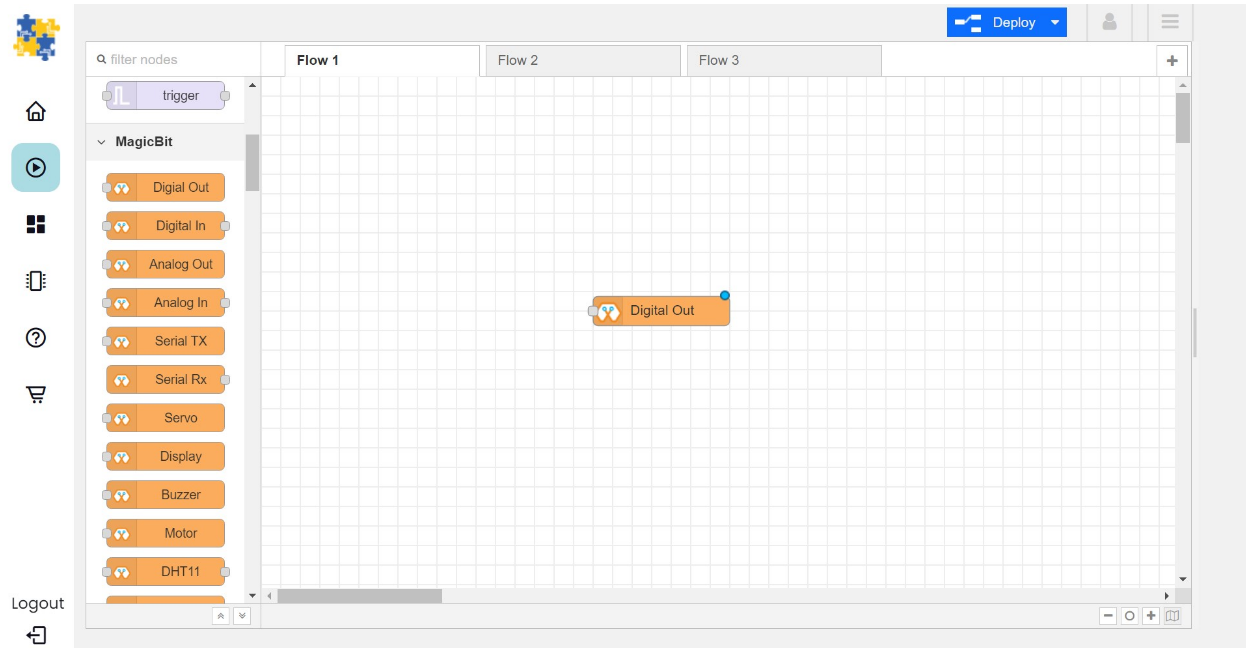
Note: "interval between times" and "at a specific time" will use cron.
"interval" should be less than 596 hours.
See info box for details.

- Click deploy
- Click button in Inject nodes to see the magic. You can control this LED from any where in the world!!!

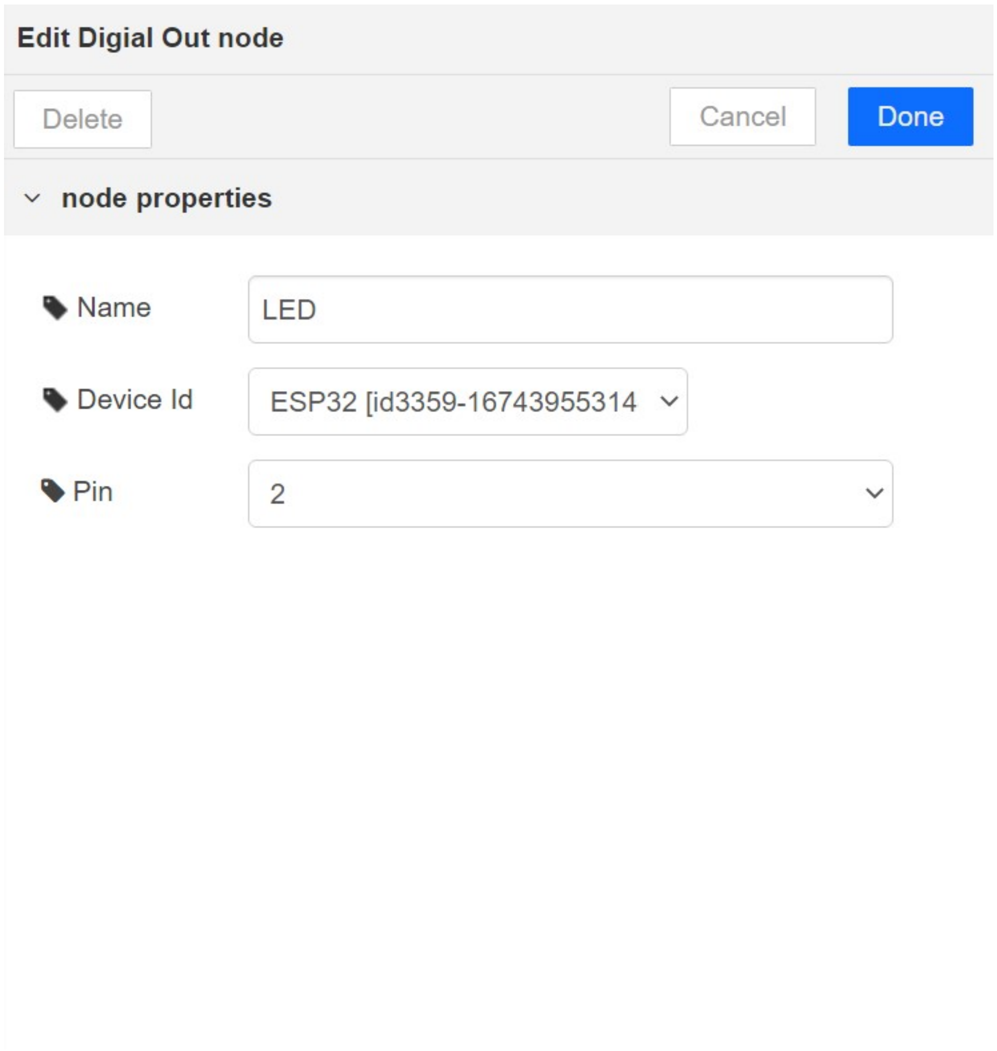
Generic ESP32

Open the Device Manager in your magicblocks account

- Turn on your ESP32 board that was setup in the previous section and wait for it to connect
- Goto the playground
- Drag and drop the **Digital Out** block under the category **Magicbit/ESP32**



- Double click on the **Digital Out** block and select device ID
- Set LED connected pin (Pin 16 in MagicKey)



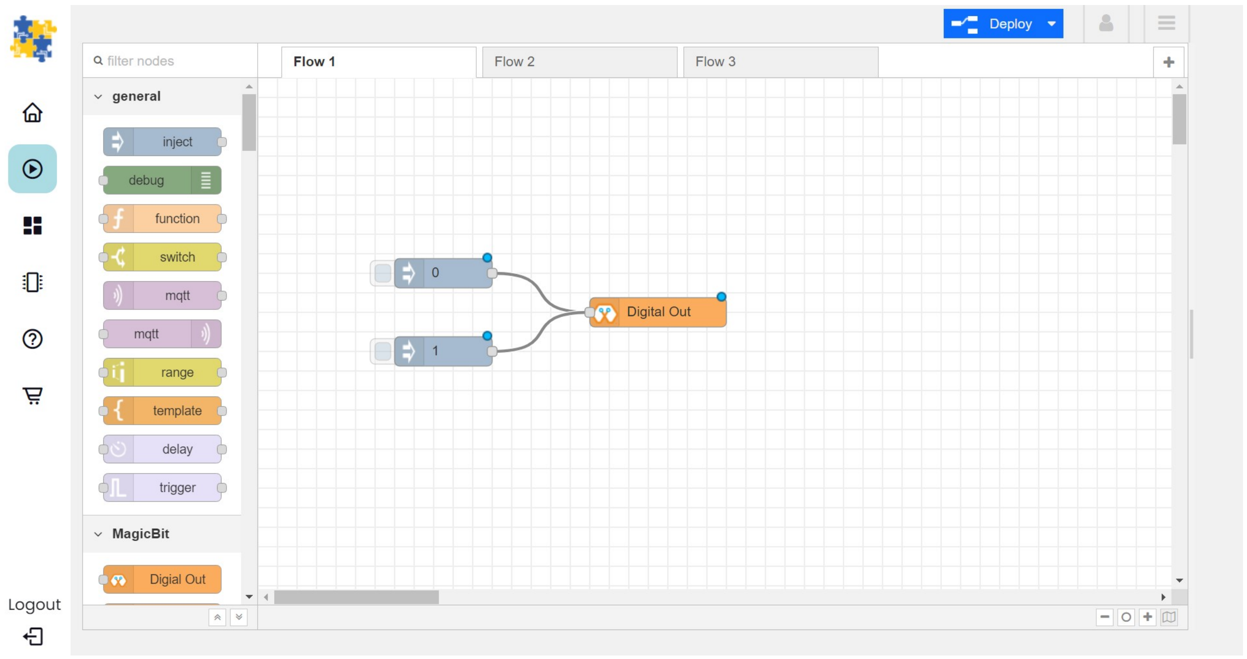
Edit Digial Out node

Name: LED

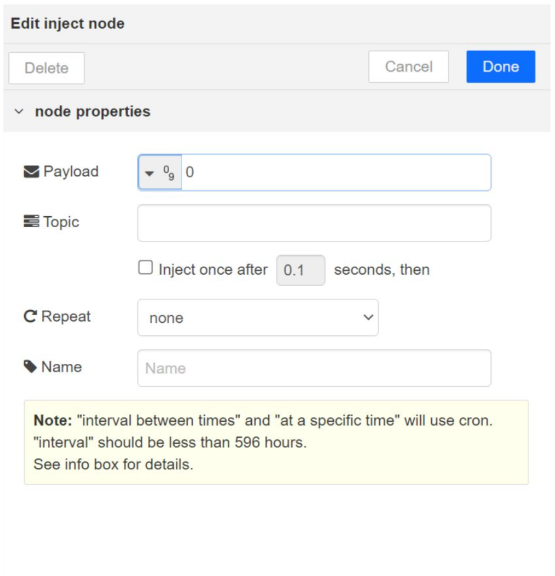
Device Id: ESP32 [id3359-16743955314

Pin: 2

- Drag and drop two **inject** blocks under the **input** category and connect them to the **Digital Out** block



- In one **inject** node set payload type as Number and set value to 0
- In Other **inject** node set payload type as Number and set value to 1



Edit inject node

Delete Cancel Done

node properties

Payload: 0

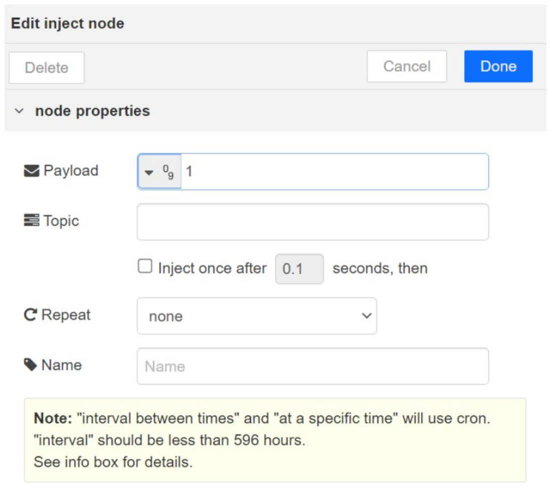
Topic:

Inject once after 0.1 seconds, then

Repeat: none

Name: Name

Note: "interval between times" and "at a specific time" will use cron.
"interval" should be less than 596 hours.
See info box for details.



Edit inject node

Delete Cancel Done

node properties

Payload: 1

Topic:

Inject once after 0.1 seconds, then

Repeat: none

Name: Name

Note: "interval between times" and "at a specific time" will use cron.
"interval" should be less than 596 hours.
See info box for details.

- Click deploy
- Click button in Inject nodes to see the magic. You can control this LED from any where in the world!!!

If you are not familiar with Magicblocks or Nodered you can quickly learn essential features from here. https://magicbit-magicblocksio.readthedocs.io/en/latest/#playground

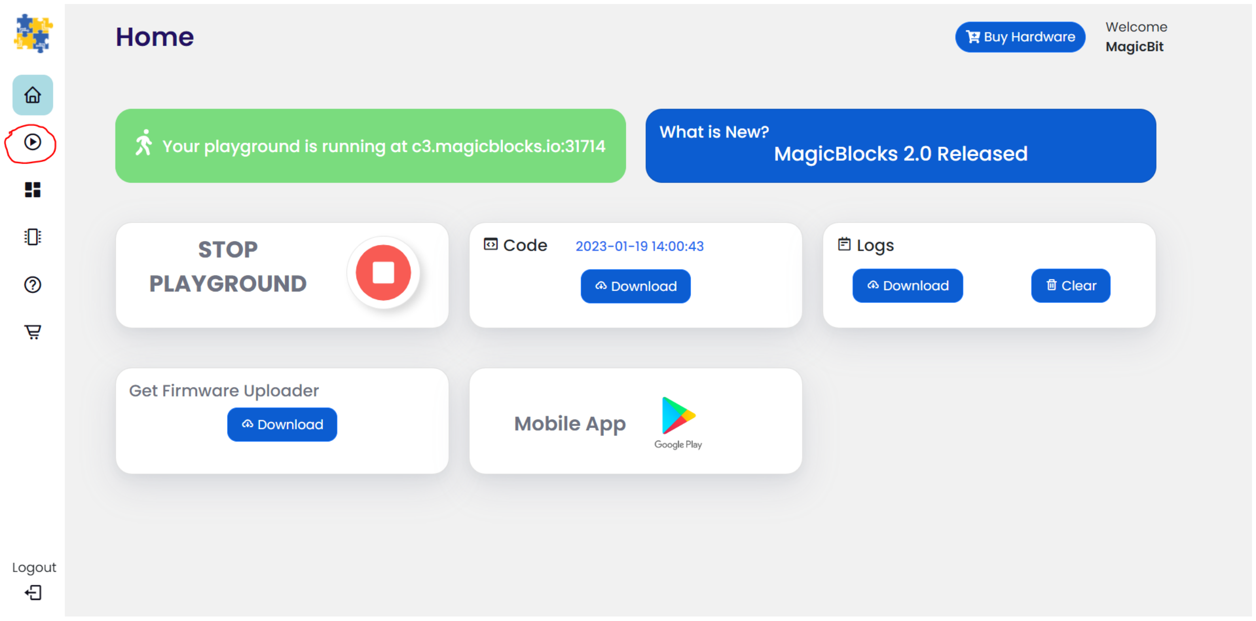
CHAPTER

FIVE

PLAYGROUND

5.1 Go to the playground!

- Login to Magicblocks and click on Playground Button to open the playground. To open the playground you should have started it from the home page.



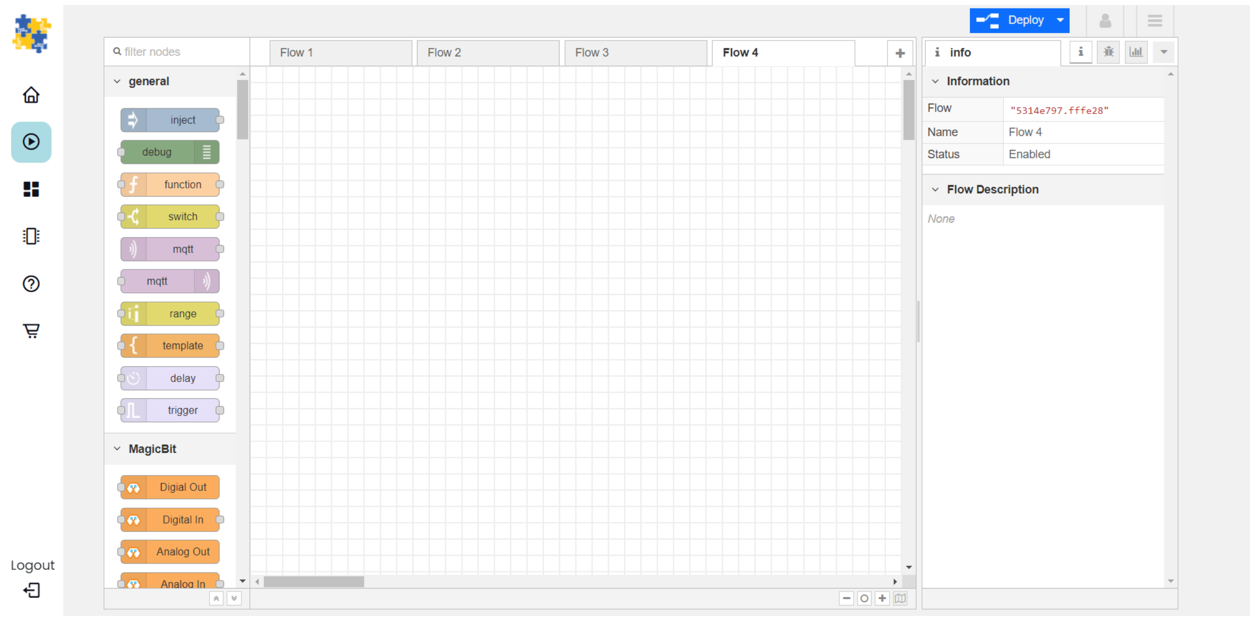
Welcome
MagicBit

Your playground is running at c3.magicblocks.io:31714

What is New?
MagicBlocks 2.0 Released

Code 2023-01-19 14:00:43

- The playground is a hosted instance of a customized version of the open source Node-Red application. On the left you will find the palette where all the blocks will reside. You can drag and drop the blocks from the palette to the canvas and start rolling very quickly. Once done click the **Deploy** button on the top right corner and all your changes will be saved. In the next section we will go through some examples covering all the relevant blocks



The editor window consists of four components:

- The header at the top, containing the deploy button, main menu, and, if user authentication is enabled, the user menu.
- The palette on the left, containing the nodes available to use.
- The main workspace in the middle, where flows are created.
- The sidebar on the right.

The main workspace is where flows are developed by dragging nodes from the palette and wiring them together. The workspace has a row of tabs along the top; one for each flow and any subflows that have been opened.

5.2 Flow

5.3 Adding a flow

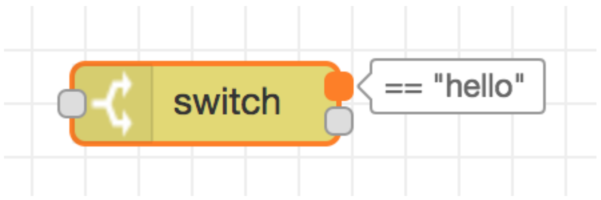
To add a new flow, click the ✚ button in the top bar.

5.4 Nodes

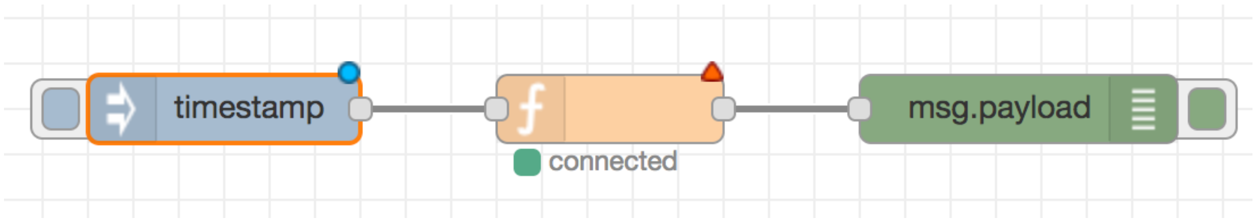
Nodes can be added to the workspace by either:

- Dragging them from the palette
- Using the quick-add dialog
- Importing from the library or clipboard

Nodes are joined together by wires via their ports. A node can have at most one input port and many output ports. A port may have a label that is displayed when the mouse hovers over it. A node may specify labels, for example, the Switch node shows the rule that matches the port. The labels can also be customised in the node edit dialog.



Some nodes display a status message and icon below the node. This is used to indicate the runtime state of the node - for example, the MQTT nodes indicate if they are currently connected or not.



If a node has any undeployed changes, it displays a blue circle above it. If there are errors with its configuration, it displays a red triangle.

Some nodes include a button on either its left or right edge. These allow some interaction with the node from within the editor. The Inject and Debug nodes are the only core nodes that have buttons.

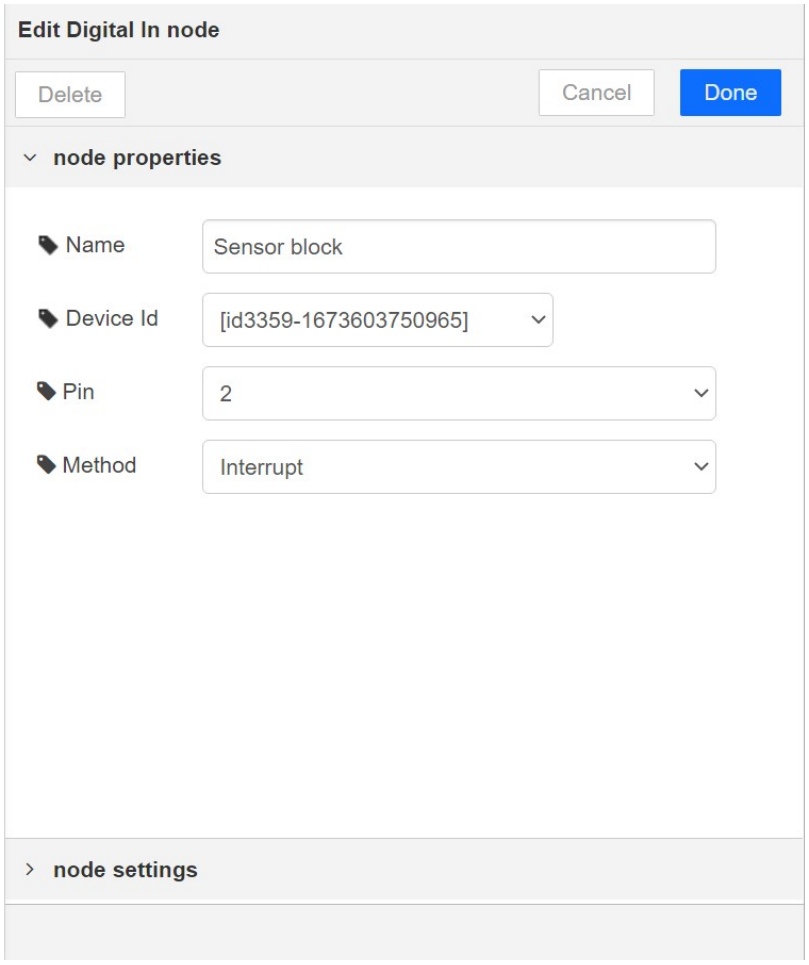
5.5 Editing node configuration

A node's configuration can be edited by double clicking on the node, or pressing **Enter** when the workspace has focus. If multiple nodes are selected, the _first_ node in the selection will be edited.

Every block that interact with your Magicbit has a property called Device ID where you need to specify to which device this block belongs to. This is important because you will be working with multiple devices in a typical IoT project. You can find device ID from magicblocks device manager.

For most nodes that interact with the Magicbit, a PIN has to be specified. This let's the Magicbit know which pin the device you intend to use is connected to.

Other than these, different blocks may have additional properties depending on which parameters are required to be specified to achieve the intended outcome from the device.


Edit Digital In node

Delete Cancel Done

node properties

Name: Sensor block

Device Id: [id3359-1673603750965]

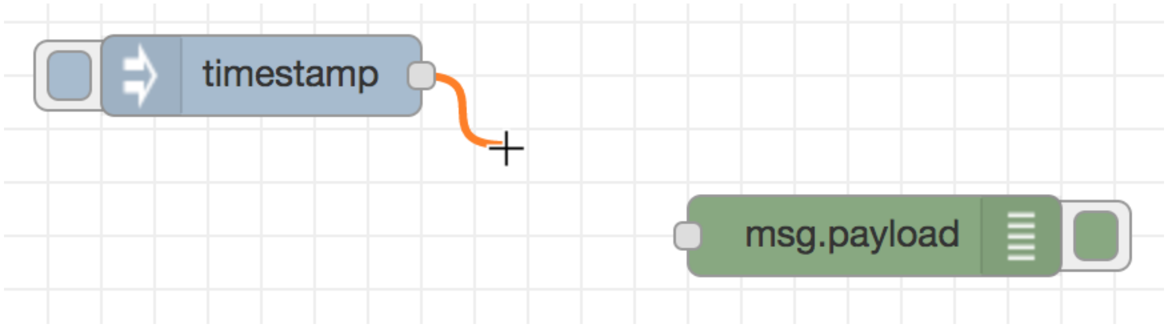
Pin: 2

Method: Interrupt

node settings

5.6 Wires

Nodes are wired together by pressing the left-mouse button on a node's port, dragging to the destination node and releasing the mouse button.



Alternatively, if the **Ctrl/Command** key is held down, the left-mouse button can be clicked (and released) on a node's port and then clicked on the destination. If the **Ctrl/Command** key remains held and the just-wired destination node has an output port, a new wire is started from that port. This allows a set of nodes to be quickly wired together.

This can also be combined with the Quick-Add dialog that is triggered by a **Ctrl/Command-Click** on the workspace to quickly insert new nodes and have them already wired to previous nodes in the flow.

5.7 Import & Export

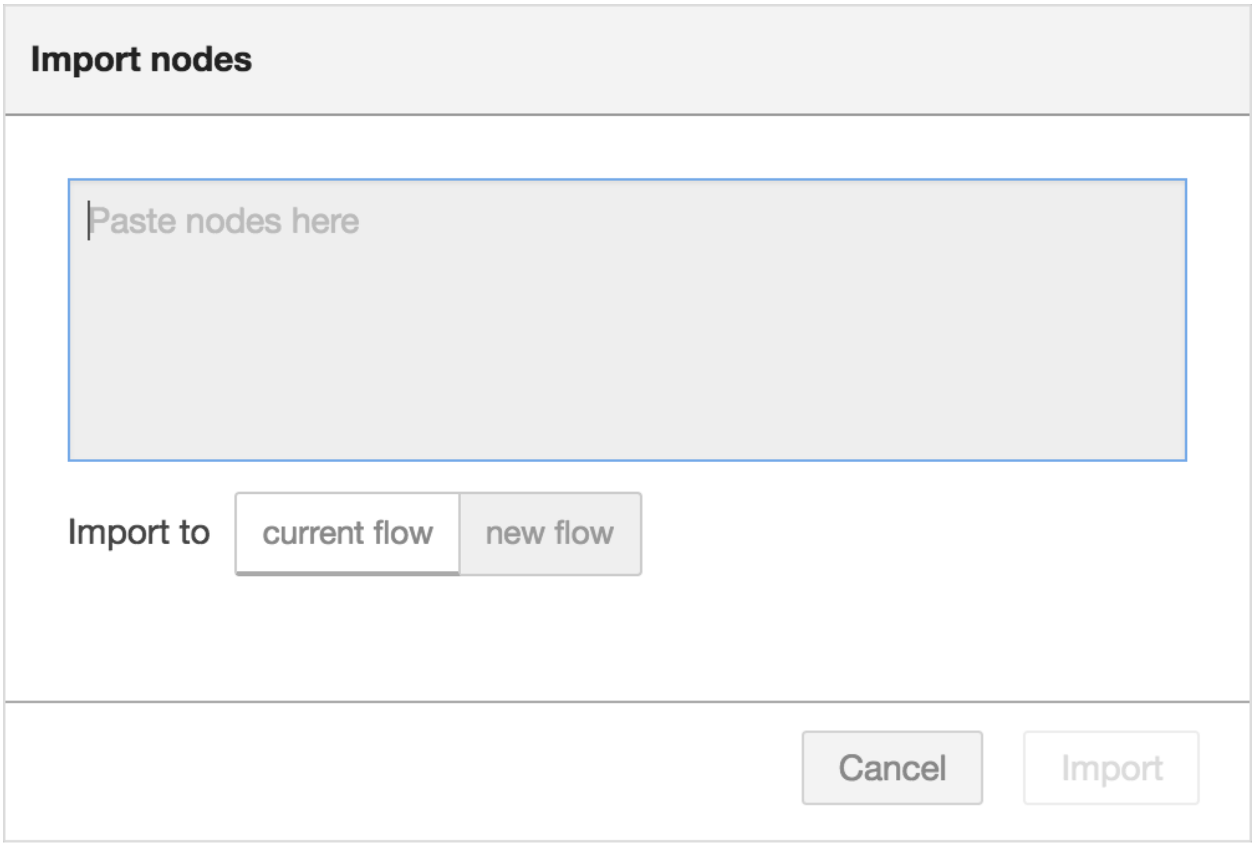
Flows can be imported and exported from the editor using their JSON format, making it very easy to share flows with others.

5.8 Importing flows

To import a flow, open the Import dialog, paste in the flow json and click 'Import'.

The 'Import' button will only be active if valid JSON is pasted into the dialog.

The dialog also offers the option to import the nodes into the current flow, or to create a new flow for them.

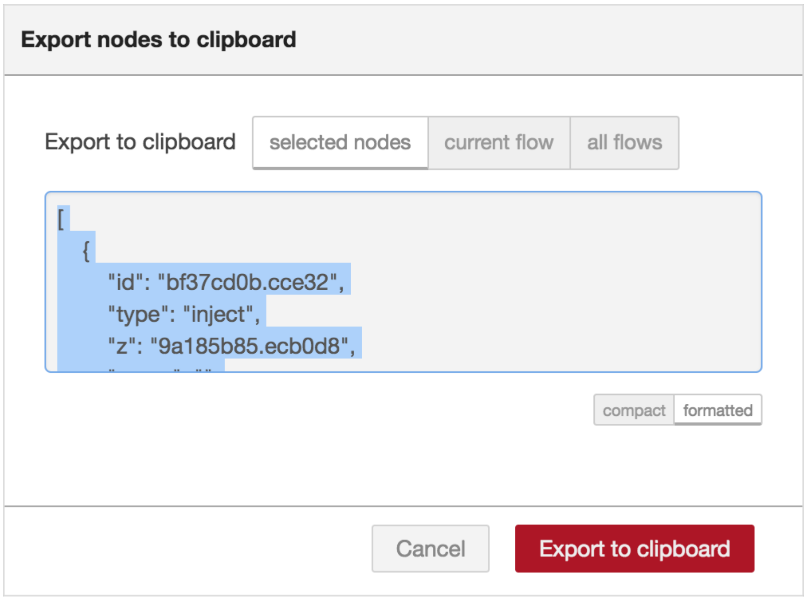


5.9 Exporting flows

The Export dialog can be used to copy flow json out of the editor.

It can export either the selected nodes, the current flow (including its tab node) or the complete flow configuration.

It offers the option to export compact or formatted JSON. The compact option generates a single line of JSON with no whitespace. The formatted JSON option is formatted over multiple lines with full indentation - which can be easier to read.

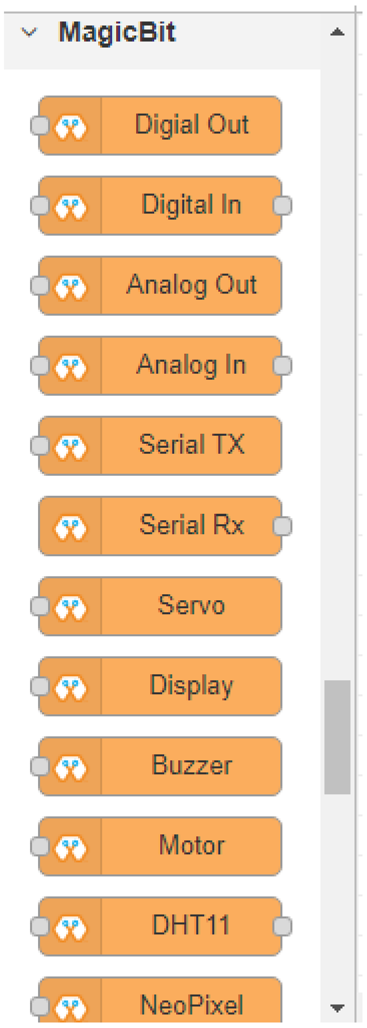


```
[
    {
        "id": "bf37cd0b.cce32",
        "type": "inject",
        "z": "9a185b85.ecb0d8",
```

CHAPTER

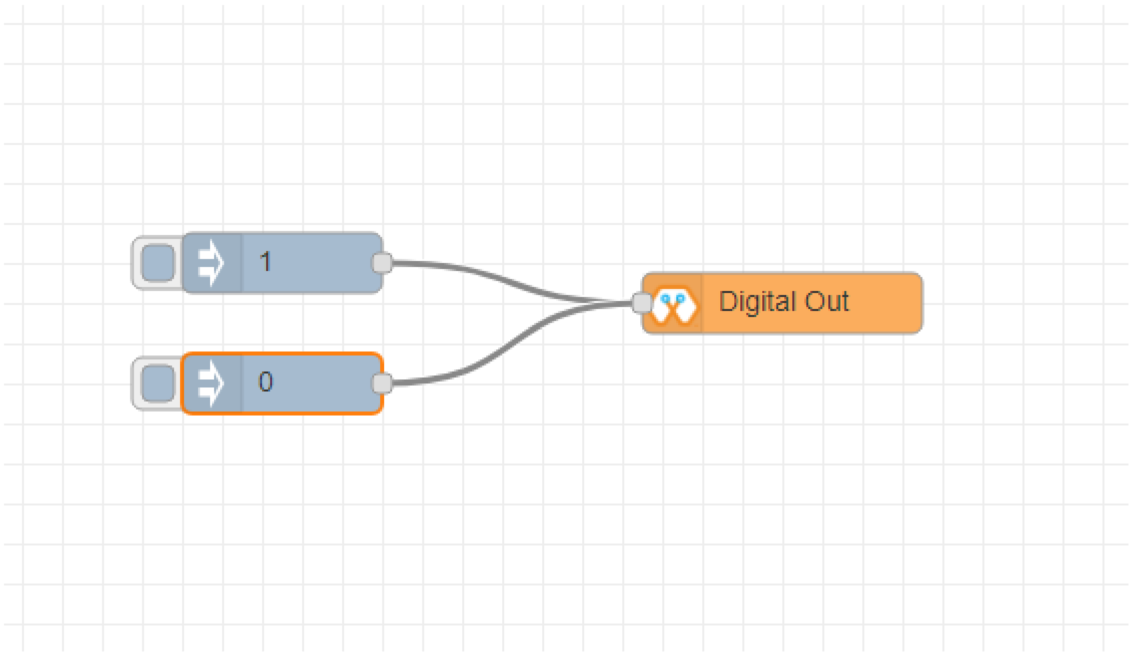
SIX

MAGICBIT BLOCKS



This block set enables you to control individual pins of the device from the playground. The functionality of each block is described below:

6.1 Digital Out



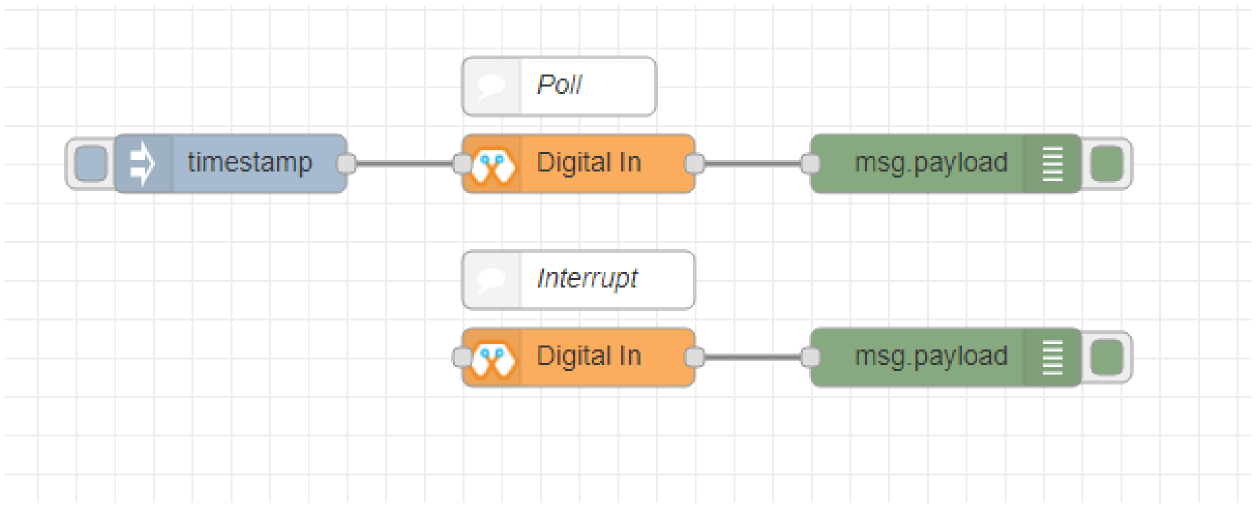
This block is used to set a digital output pin to 1 or 0 based on the input. An input of 1 or true will make the configured pin go HIGH and vice versa

- **Configuration**
  - Pin: pin number of the Digital pin to write to. Available pins can be selected from dropdown list.
  - Name: Any name desired
- **Input**
  - value to be written to the pin. Accepts 1 (true) or 0 (false) eg: {“payload”: 1}

Import the already setup blocks below to try out the function of the Digital Out block:

```
[{"id":"2a06e0c8.00d53","type":"inject","z":"8e70cb96.beec38","name":"","topic":"",
↪"payload":"0","payloadType":"num","repeat":"","crontab":"","once":false,"onceDelay":0.
↪1,"x":170,"y":200,"wires":[["19527767.b23fd9"]]},{"id":"60e26730.7f8d38","type":"inject
↪","z":"8e70cb96.beec38","name":"","topic":"","payload":"1","payloadType":"num","repeat
↪":"","crontab":"","once":false,"onceDelay":0.1,"x":170,"y":280,"wires":[["19527767.
↪b23fd9"]]},{"id":"19527767.b23fd9","type":"DO","z":"8e70cb96.beec38","name":"","epId":"
↪","pin":"","x":370,"y":240,"wires":[]}]
```

6.2 Digital In



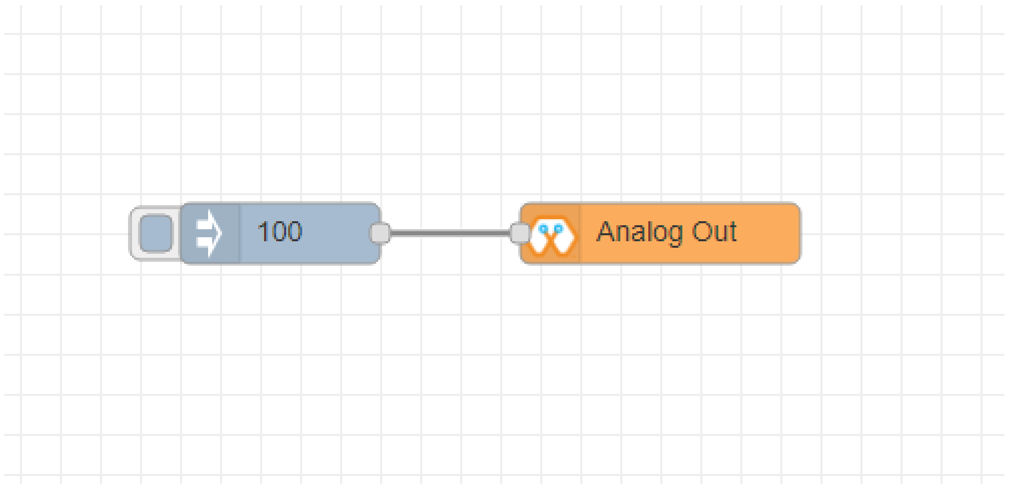
This block will read Input status of a pin. There are two methods to get input 1. Poll - Block need to triggered to get input status. Any input will serve as a trigger. (eg:inject block) 2. Interrupt - If there is any change of pin state of Magicbit block will output the current state Input status can be passed to a another block or viewed on the debug window.

- **Configuration:**
  - Pin: pin number of the digital pin to read .Select from drop down list
  - Name: Any name desired
  - Method: Poll/Interrupt
- **Input**
  - Any input. Used as a trigger
- **Output**
  - Value of the pin as 1 or 0 in the following format and the pin number as the topic

Import the already setup blocks below to try out the function of the Digital In block:

```
[{"id":"4ec6888d.f14568","type":"DI","z":"8e70cb96.beec38","name":"","epId":"","pin":"34
↪","method":0,"x":280,"y":240,"wires":[["44b3d23d.fd52fc"]]},{"id":"44b3d23d.fd52fc",
↪"type":"debug","z":"8e70cb96.beec38","name":"","active":true,"tosidebar":true,"console
↪":false,"tostatus":false,"complete":"false","x":470,"y":240,"wires":[]},{"id":
↪"dd7e85ae.3a35d8","type":"comment","z":"8e70cb96.beec38","name":"Poll","info":"","x
↪":270,"y":200,"wires":[]},{"id":"9e09420f.eaa95","type":"comment","z":"8e70cb96.beec38
↪","name":"Interrupt ","info":"","x":280,"y":320,"wires":[]},{"id":"7cfd827f.97cafc",
↪"type":"DI","z":"8e70cb96.beec38","name":"","epId":"","pin":"35","method":"1","x":280,
↪"y":360,"wires":[["7770f92b.911f78"]]},{"id":"7770f92b.911f78","type":"debug","z":
↪"8e70cb96.beec38","name":"","active":true,"tosidebar":true,"console":false,"tostatus
↪":false,"complete":"false","x":480,"y":360,"wires":[]},{"id":"f62ae0a.d03e52","type":
↪"inject","z":"8e70cb96.beec38","name":"","topic":"","payload":"","payloadType":"date",
↪"repeat":"5","crontab":"","once":false,"onceDelay":0.1,"x":110,"y":240,"wires":[[
↪"4ec6888d.f14568"]]}]
```

6.3 Analog Out



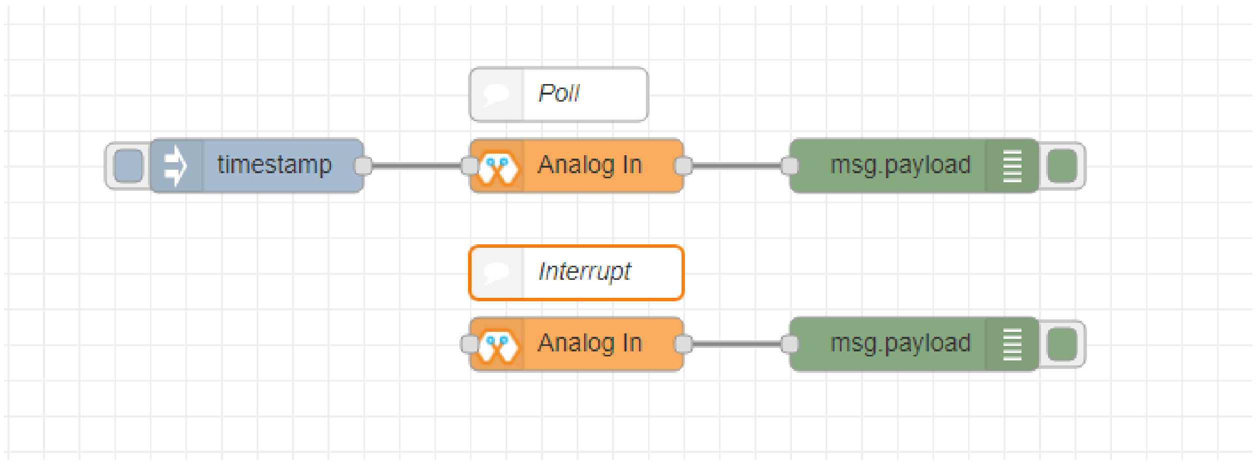
This block used to set PWM to pins of Magicbit. Value should be in range of 0-255. Inject block or output of a another block can be used to set the value.

- **Configuration:**
  - Pin: pin number of the to set PWM. Select from drop down list
  - Name: Any name desired
- **Input**
  - Inject block or any block. Input value should be in range of 0-255

Import the already setup blocks below to try out the function of the Analog Out block:

```
[{"id":"90c1c4fb.315c28","type":"inject","z":"8e70cb96.beec38","name":"","topic":"100",
↪"payload":"","payloadType":"num","repeat":"","crontab":"","once":false,"onceDelay":0.1,
↪"x":110,"y":300,"wires":[["35d83292.14be1e"]]},{"id":"35d83292.14be1e","type":"AO","z":
↪"8e70cb96.beec38","name":"","epId":"","pin":"16","x":300,"y":300,"wires":[]}]
```

6.4 Analog In



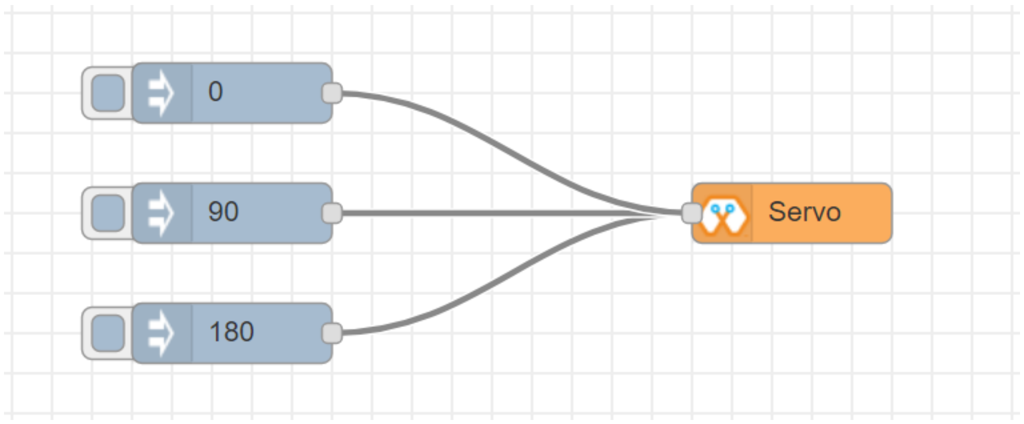
This block will read analog value of the ADC pin of the module. Similar to the digital in block you need to set method to read the value. Any input sent to the block will serve as the trigger. 1. Poll - Block need to triggered to get input status. Any input will serve as a trigger. (eg:inject block) 2. Interrupt - If there is any change greater or less than threshold value of Magicbit block will output the analog value

- **Configuration:**
  - Pin: pin number of the analog pin to read (Required)
  - Name: Any name desired
  - Method: Poll/Interrupt
  - Threshold: If interrupt method selected value return from output if there is any change greater or less than this value
- **Input**
  - Any input. Used as a trigger
- **Output**
  - Value of the pin from 0 to 4096 (12bit ADC) {"payload": 965}

Import the already setup blocks below to try out the function of the Analog In block:

```
[{"id":"90c1c4fb.315c28","type":"inject","z":"8e70cb96.beec38","name":"","topic":"",
↪"payload":"","payloadType":"date","repeat":"5","crontab":"","once":false,"onceDelay":0.
↪1,"x":130,"y":300,"wires":[["71f043eb.d5922c"]]},{"id":"ba86530.51c62b","type":"debug",
↪"z":"8e70cb96.beec38","name":"","active":true,"tosidebar":true,"console":false,
↪"tostatus":false,"complete":"false","x":490,"y":300,"wires":[]},{"id":"27777b37.50fcb4
↪","type":"comment","z":"8e70cb96.beec38","name":"Poll","info":"","x":290,"y":260,"wires
↪":[]},{"id":"1e99e73b.121039","type":"comment","z":"8e70cb96.beec38","name":"Interrupt
↪","info":"","x":300,"y":380,"wires":[]},{"id":"1276b371.20bdad","type":"debug","z":
↪"8e70cb96.beec38","name":"","active":true,"tosidebar":true,"console":false,"tostatus
↪":false,"complete":"false","x":500,"y":420,"wires":[]},{"id":"71f043eb.d5922c","type":
↪"AI","z":"8e70cb96.beec38","name":"","epId":"","pin":"","method":0,"threshold":100,"x
↪":300,"y":300,"wires":[["ba86530.51c62b"]]},{"id":"4752d44e.ecb9ec","type":"AI","z":
↪"8e70cb96.beec38","name":"","epId":"","pin":"","method":"1","threshold":100,"x":300,"y
↪":420,"wires":[["1276b371.20bdad"]]}]
```

6.5 Servo



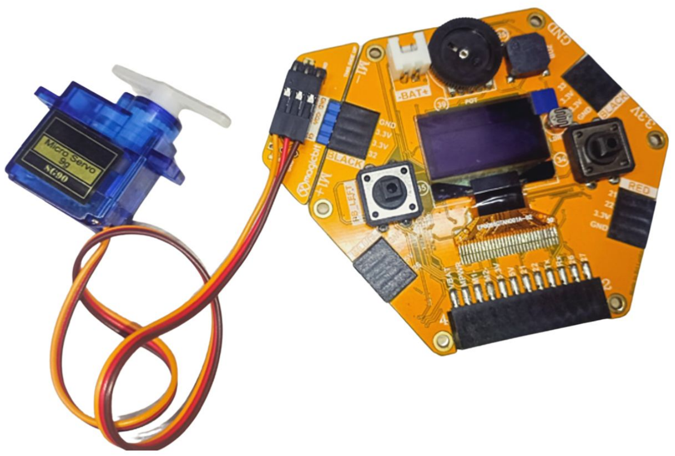
This block will take an input between 0 and 180, and set the servo angle to that position in degrees.

- **Configuration:**
  - Name: Any name desired
  - Device ID: Select device from the drop-down menu
  - Pin: pin number of the Servo (Required)
- **Input**
  - Integer between 0 and 180.

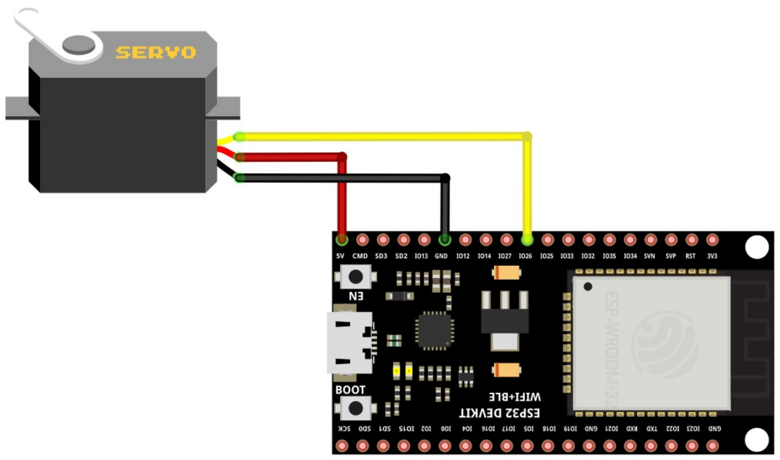
Import the already setup blocks below to try out the function of the Servo block:

```
[{"id":"a3277135.4860a","type":"inject","z":"8e70cb96.beec38","name":"","topic":"0",
↪"payload":"","payloadType":"num","repeat":"","crontab":"","once":false,"onceDelay":0.1,
↪"x":130,"y":240,"wires":[["c4fae3be.28a7b"]]},{"id":"88f7fa5e.ca4808","type":"inject",
↪"z":"8e70cb96.beec38","name":"","topic":"90","payload":"","payloadType":"num","repeat":
↪"","crontab":"","once":false,"onceDelay":0.1,"x":130,"y":300,"wires":[["c4fae3be.28a7b
↪"]]},{"id":"e7bdb8d7.7f3a98","type":"inject","z":"8e70cb96.beec38","name":"","topic":
↪"180","payload":"","payloadType":"num","repeat":"","crontab":"","once":false,"onceDelay
↪":0.1,"x":130,"y":360,"wires":[["c4fae3be.28a7b"]]},{"id":"c4fae3be.28a7b","type":
↪"Servo","z":"8e70cb96.beec38","name":"","epId":"","pin":"","x":350,"y":300,"wires":[]}]
```

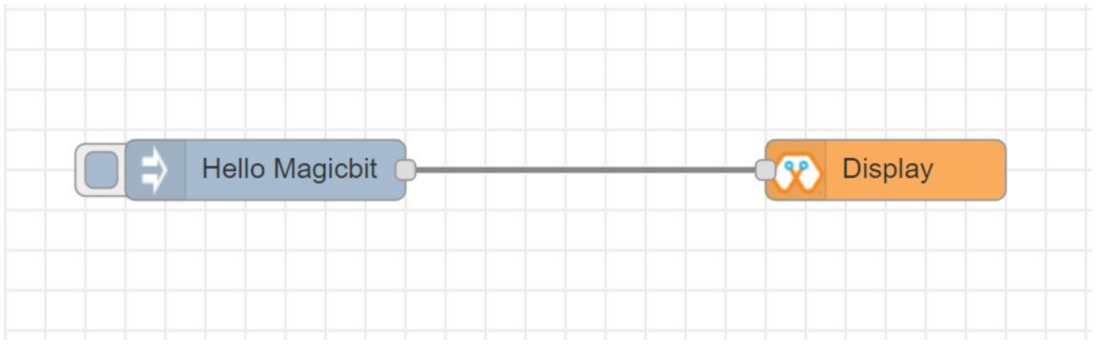
- Connection
  - Magicbit



  - Generic ESP32



6.6 Display



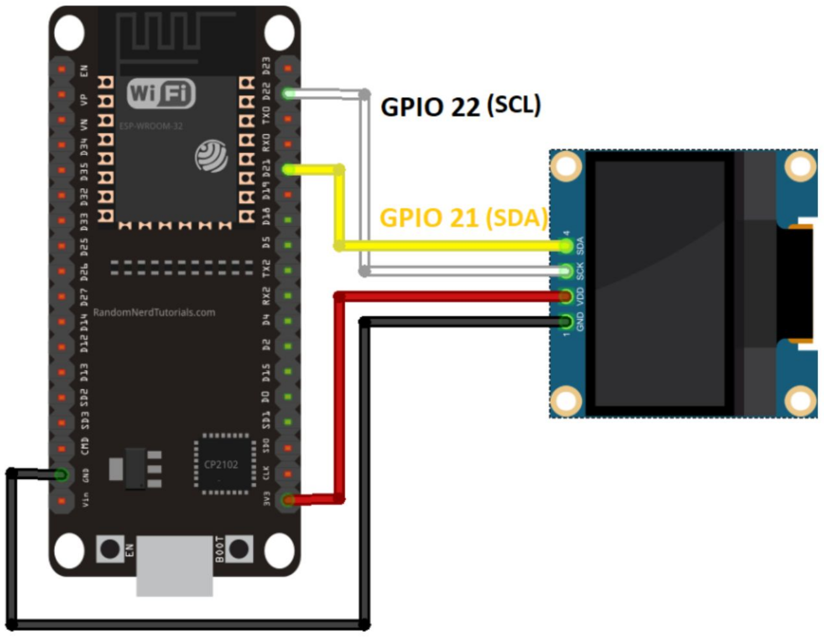
This block will take a text input and display it on the Magicbit's OLED display.

- **Configuration:**
  - Name: Any name desired
  - Device ID: Select device from the drop-down menu
  - Font Size: The font size of the text to be displayed (between 1 to 5)
- **Input**
  - String

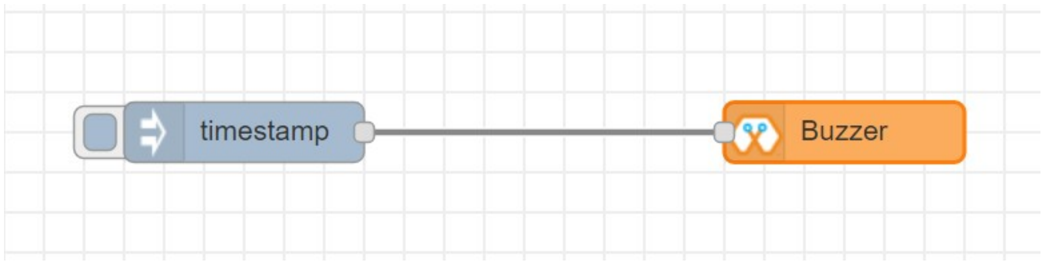
Import the already setup blocks below to try out the function of the Display block:

```
[{"id":"1f2691d0.3098ee","type":"inject","z":"8e70cb96.beec38","name":"","topic":"Hello␣
↪Magicbit","payload":"","payloadType":"str","repeat":"","crontab":"","once":false,
↪"onceDelay":0.1,"x":180,"y":280,"wires":[["a88bce4d.e7d84"]]},{"id":"a88bce4d.e7d84",
↪"type":"OLED","z":"8e70cb96.beec38","name":"","epId":"","fontSize":"1","x":400,"y":280,
↪"wires":[]}]
```

- Connection for Generic ESP32



6.6.1 Buzzer



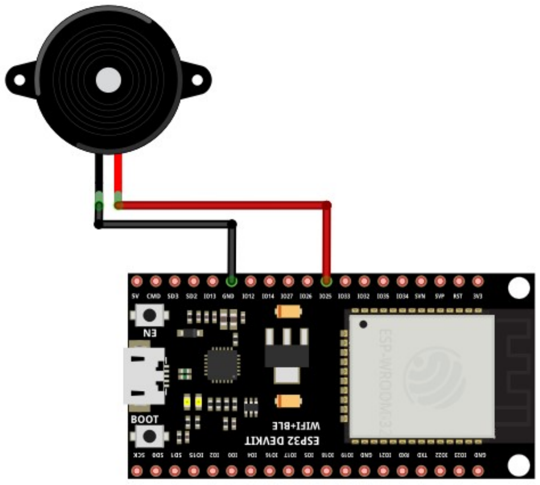
This block will take an input pulse and play a tone for a preset amount of time.

- **Configuration:**
  - Name: Any name desired
  - Device ID: Select device from the drop-down menu
  - Frequency: Frequency of the tone to be played (between 20 Hz and 20 kHz)
  - Duration: How long the tone should be played
- **Input**
  - Any input trigger pulse

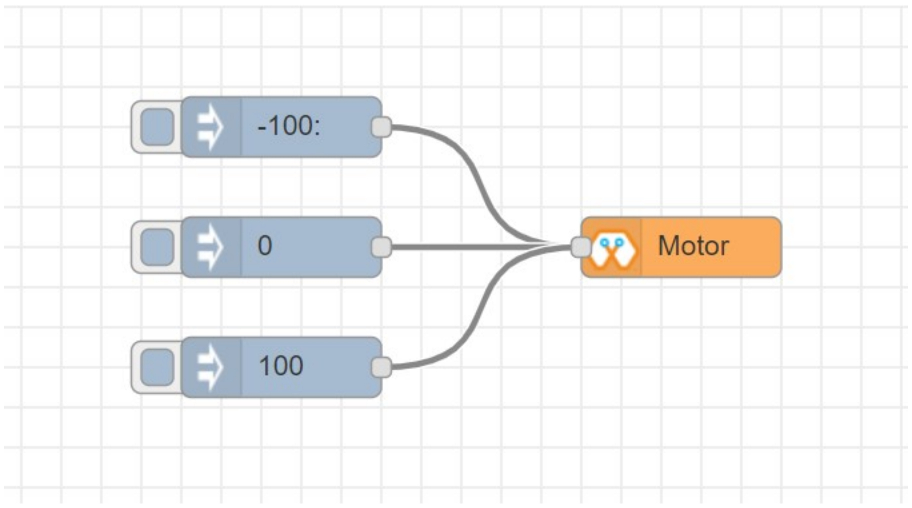
Import the already setup blocks below to try out the function of the Buzzer block:

```
[{"id":"ff346570.167dc8","type":"BUZZ","z":"8e70cb96.beec38","name":"","epId":"",
↪"frequency":1000,"duration":"500","x":460,"y":300,"wires":[]},{"id":"eb281e99.c1c23",
↪"type":"inject","z":"8e70cb96.beec38","name":"","topic":"","payload":"","payloadType":
↪"date","repeat":"1","crontab":"","once":false,"onceDelay":0.1,"x":250,"y":300,"wires
↪":[["ff346570.167dc8"]]}]
```

- Connection for Generic ESP32



6.7 Motor



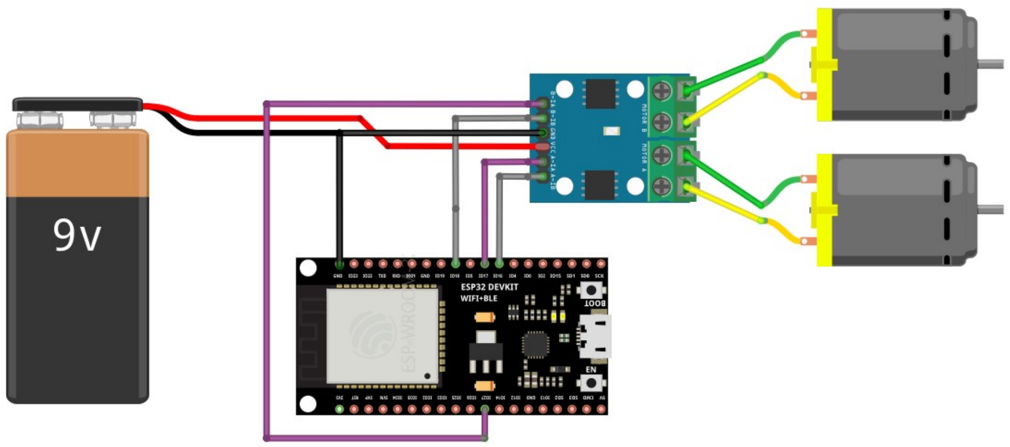
This block will take an input value for msg.payload which is between -100 and 100 and turn the selected motor Foward or Backwards in the given speed.

- **Configuration:**
  - Name: Any name desired
  - Device ID: Select device from the drop-down menu
  - Motor: Select which motor (motor 1 or motor 2) needs to be controlled by the node
- **Input**
  - Integer between -100 and 100 for msg.payload

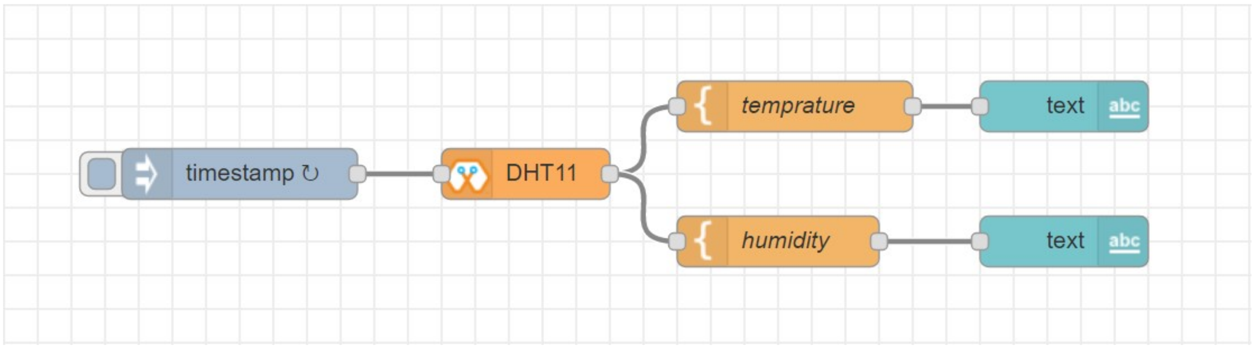
Import the already setup blocks below to try out the function of the Motor block:

```
[{"id":"c01c0ecb.99d91","type":"inject","z":"8e70cb96.beec38","name":"","topic":"-100",
↪"payload":"","payloadType":"num","repeat":"","crontab":"","once":false,"onceDelay":0.1,
↪"x":170,"y":260,"wires":[["a56f3710.8d4f88"]]},{"id":"4a2621f.68e6be","type":"inject",
↪"z":"8e70cb96.beec38","name":"","topic":"","payload":"0","payloadType":"num","repeat":"
↪","crontab":"","once":false,"onceDelay":0.1,"x":170,"y":320,"wires":[["a56f3710.8d4f88
↪"]]},{"id":"c36dde0f.e4332","type":"inject","z":"8e70cb96.beec38","name":"","topic":"",
↪"payload":"100","payloadType":"num","repeat":"","crontab":"","once":false,"onceDelay
↪":0.1,"x":170,"y":380,"wires":[["a56f3710.8d4f88"]]},{"id":"a56f3710.8d4f88","type":"M
↪","z":"8e70cb96.beec38","name":"","epId":"","motor":"1","x":370,"y":320,"wires":[]}]
```

- Connection - Magicbit .. image:: Images/motor_connect_magicbit.jpg
  - Generic ESP32



6.8 DHT11



The DHT11 node can be used to output the temperature and the humidity of the surrounding environment when its triggered by an input pulse.

- **Configuration:**

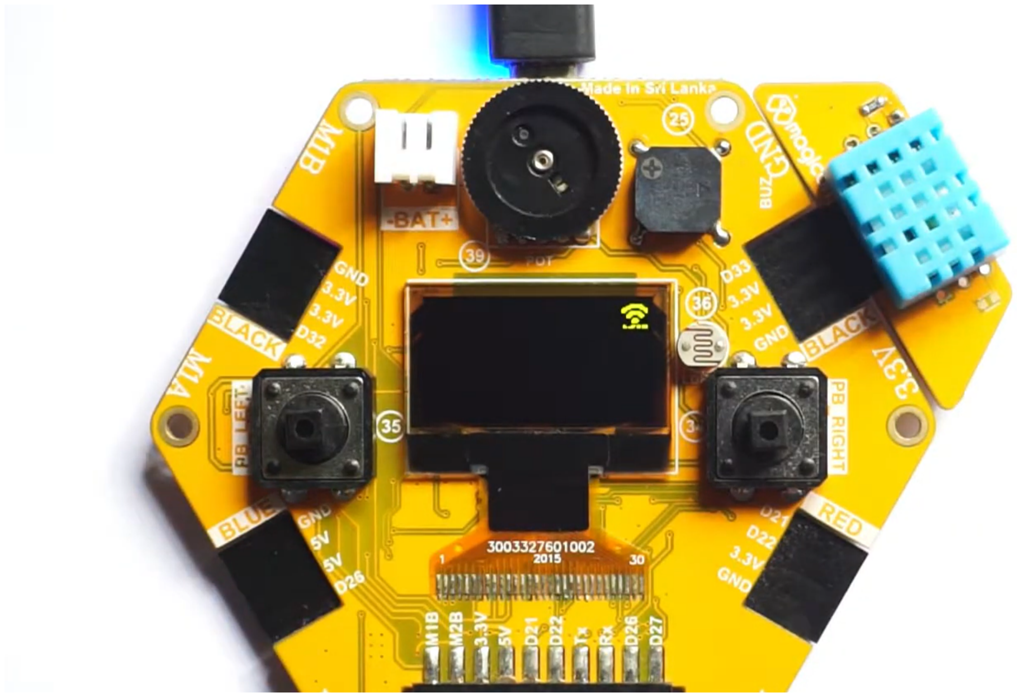
  - Name: Any name desired
  - Device ID: Select device from the drop-down menu
  - Pin: Select the Pin to which the sensor is connected from drop down list
  - Mode: Configure the block to output Temperature only, humidity only or both temperature and humidity together in the same message.

- **Input**
  - Any input trigger pulse
- **Output**
  - If both temperature and humidity option is chosen in Mode: msg.payload.temperature will give temperature in celsius, and msg.payload.humidity will give relative humidity as a percentage.
  - If either temperature or humidity is selected in mode, msg.payload will directly give temperature in celsius or relative humidity as a percentage respectively.

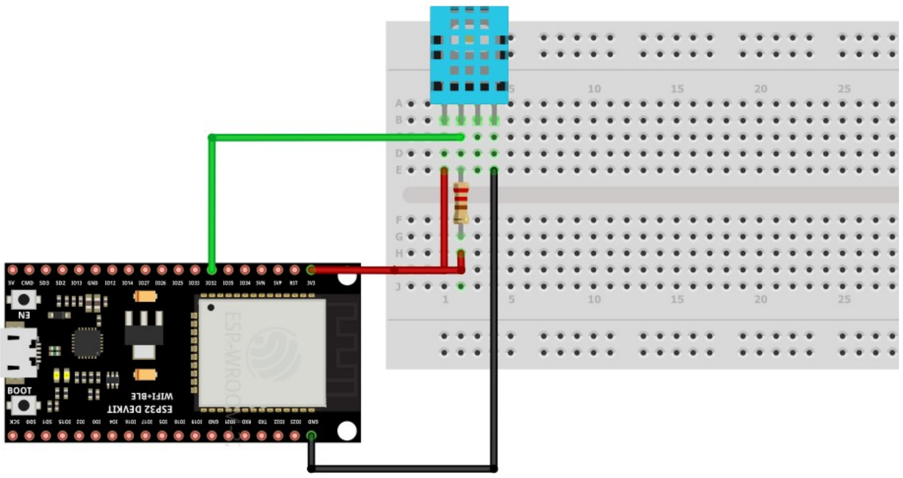
Import the already setup blocks below to try out the function of the DHT11 block:

```
[{"id":"61142508.ed114c","type":"DHT","z":"8e70cb96.beec38","name":"","epId":"id3359-
↪1674810450279","pin":"32","mode":"TH","x":310,"y":260,"wires":[["6a681143.6d77c",
↪"340140d5.c74bb"]]},{"id":"7ca6bc4e.b616d4","type":"inject","z":"8e70cb96.beec38","name
↪":"","topic":"","payload":"","payloadType":"date","repeat":"5","crontab":"","once
↪":false,"onceDelay":0.1,"x":140,"y":260,"wires":[["61142508.ed114c"]]},{"id":"6a681143.
↪6d77c","type":"template","z":"8e70cb96.beec38","name":"","field":"payload","fieldType":
↪"msg","format":"handlebars","syntax":"mustache","template":"Temprature = {{payload.
↪temperature}} °C","output":"str","x":480,"y":220,"wires":[["1b586b40.db1c45"]]},{"id":
↪"1b586b40.db1c45","type":"debug","z":"8e70cb96.beec38","name":"","active":true,
↪"tosidebar":true,"console":false,"tostatus":false,"complete":"false","x":670,"y":220,
↪"wires":[]},{"id":"340140d5.c74bb","type":"template","z":"8e70cb96.beec38","name":"",
↪"field":"payload","fieldType":"msg","format":"handlebars","syntax":"mustache","template
↪":"Relative Humidity = {{payload.humidity}} %","output":"str","x":480,"y":300,"wires
↪":[["68dd224e.68e4dc"]]},{"id":"68dd224e.68e4dc","type":"debug","z":"8e70cb96.beec38",
↪"name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":
↪"false","x":670,"y":300,"wires":[]}]
```

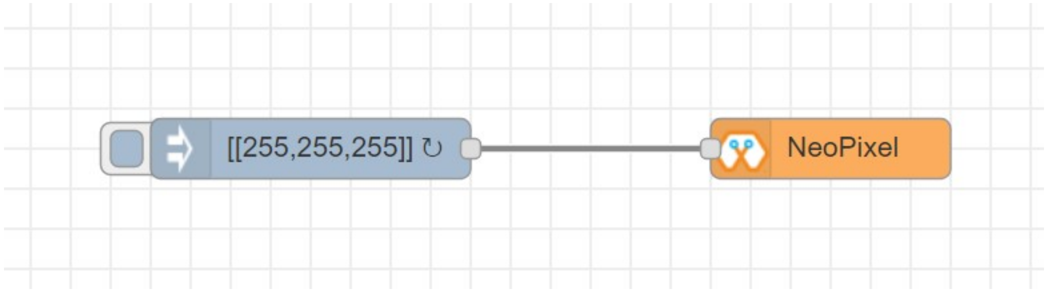
- Connection
  - Magicbit



– Generic ESP32



6.9 NeoPixel



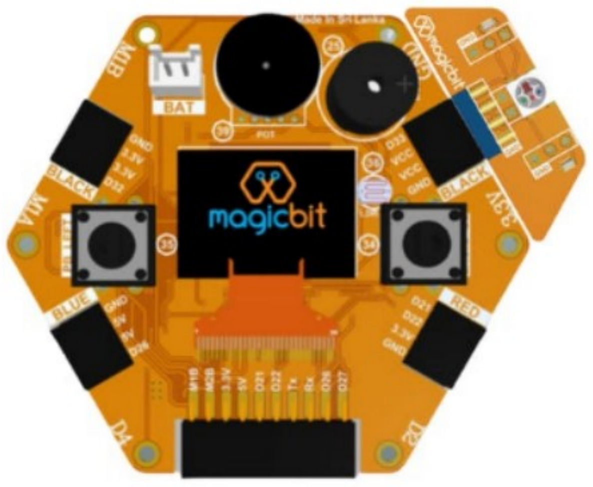
This block will take an array of input with RGB values, to control a series of NeoPixel LEDs.

- **Configuration:**
  - Name: Any name desired
  - Device ID: Select device from the drop-down menu
  - Pin: Select the Pin to which the NeoPixel LED series is connected from drop down list
- **Input**
  - For n number of LEDs, input msg.payload as a JSON object in the form [[R1,G1,B1],[R2,G2,B2]…..[Rn,Gn,Bn]] Where R,G,B (between 0 to 255) values are the intensities of Red, Green and Blue colors respectively.

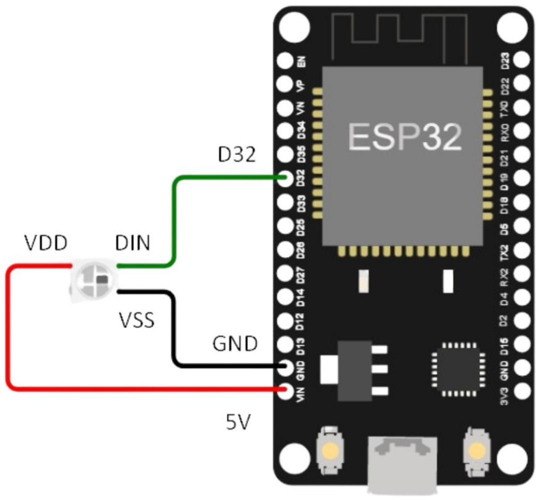
Import the already setup blocks below to try out the function of the NeoPixel block:

```
[{"id":"57f95cdb.67e0a4","type":"inject","z":"8e70cb96.beec38","name":"","topic":"",
↪"payload":"[[255,255,255]]","payloadType":"json","repeat":"","crontab":"","once":false,
↪"onceDelay":0.1,"x":180,"y":200,"wires":[["401a45d9.19fefc"]]},{"id":"401a45d9.19fefc",
↪"type":"LED","z":"8e70cb96.beec38","name":"","epId":"id3359-1674810450279","pin":"26",
↪"x":440,"y":200,"wires":[]}]
```

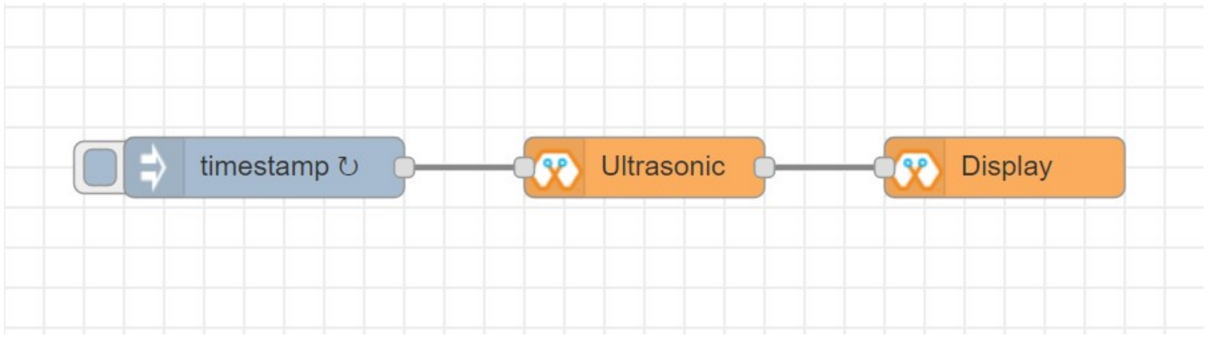
- Connection
  - Magicbit



– Generic ESP32

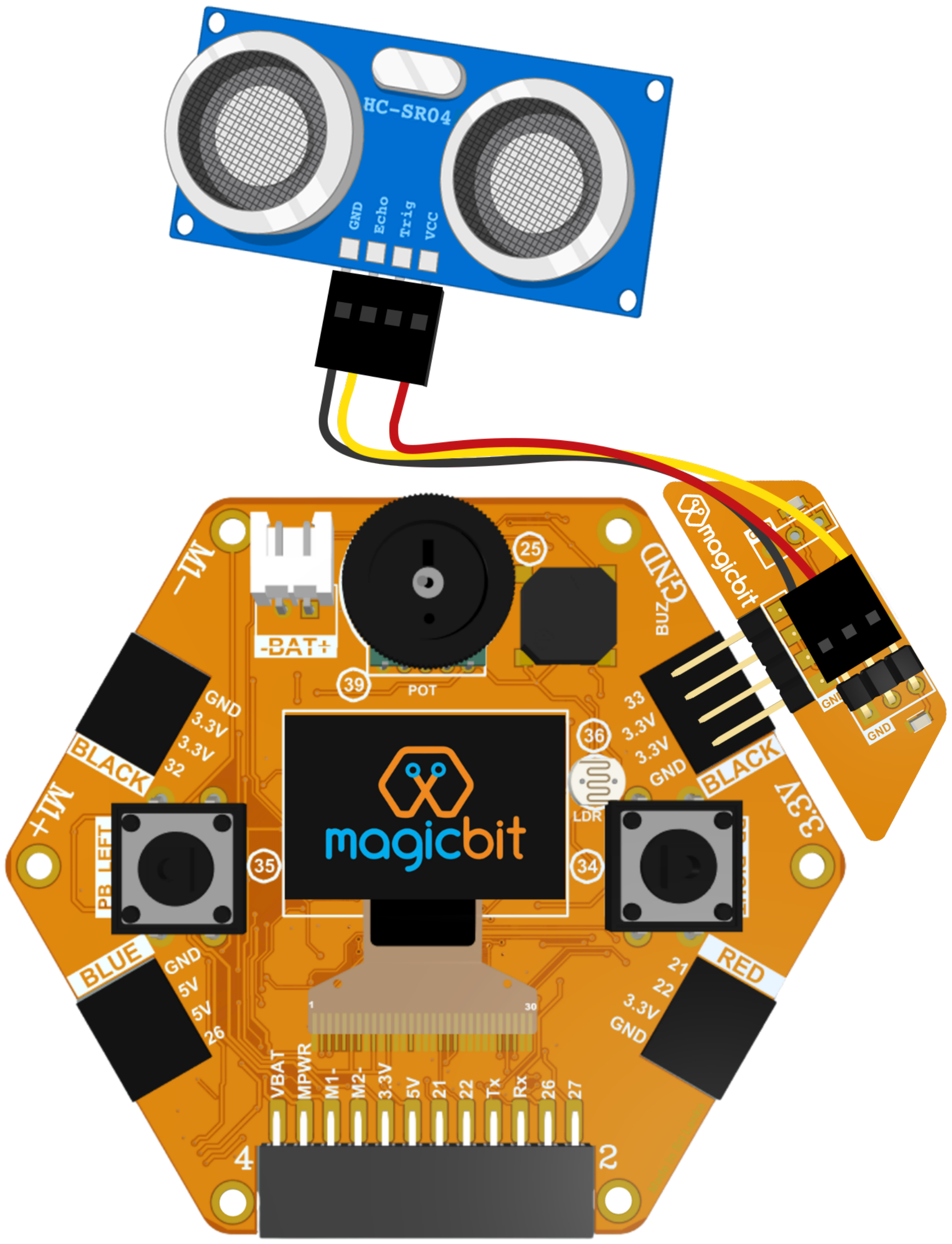


6.10 Ultrasonic

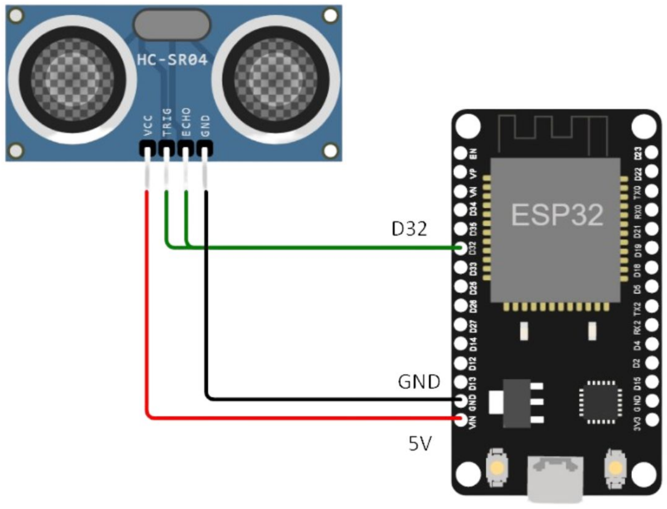


This block will output the distance to an obstacle in centimeters when an input trigger pulse is given.

- **Configuration:**
  - Name: Any name desired
  - Device ID: Select device from the drop-down menu
- **Input**
  - Any input trigger pulse
- **Output**
  - Integer distance to an obstacle in centimeters
- Connection
  - Magicbit



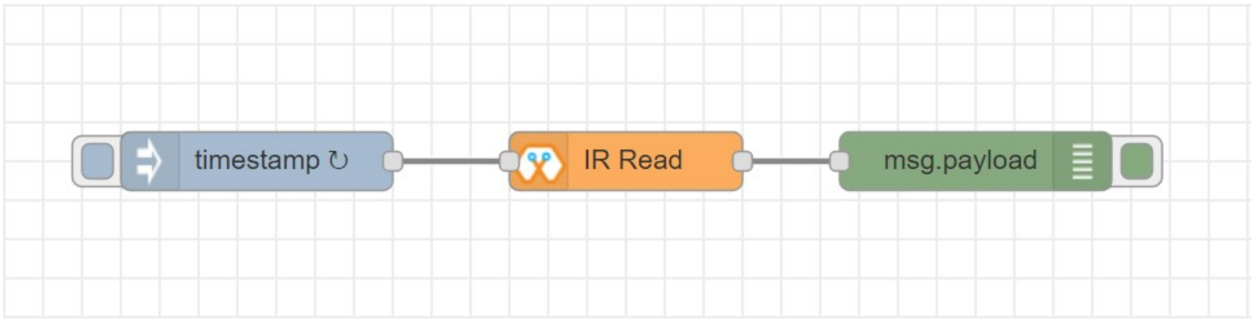
– Generic ESP32



Import the already setup blocks below to try out the function of the Ultrasonic block:

```
[{"id":"d41059f6.2cb2f8","type":"OLED","z":"8e70cb96.beec38","name":"","epId":"",
↪"fontSize":"3","x":540,"y":260,"wires":[]},{"id":"f39d71c7.ce169","type":"inject","z":
↪"8e70cb96.beec38","name":"","topic":"","payload":"","payloadType":"date","repeat":"1",
↪"crontab":"","once":false,"onceDelay":0.1,"x":110,"y":260,"wires":[["807996ca.371c98
↪"]]},{"id":"807996ca.371c98","type":"US","z":"8e70cb96.beec38","name":"","epId":"","x
↪":320,"y":260,"wires":[["d41059f6.2cb2f8"]]}]
```

6.11 IR Read

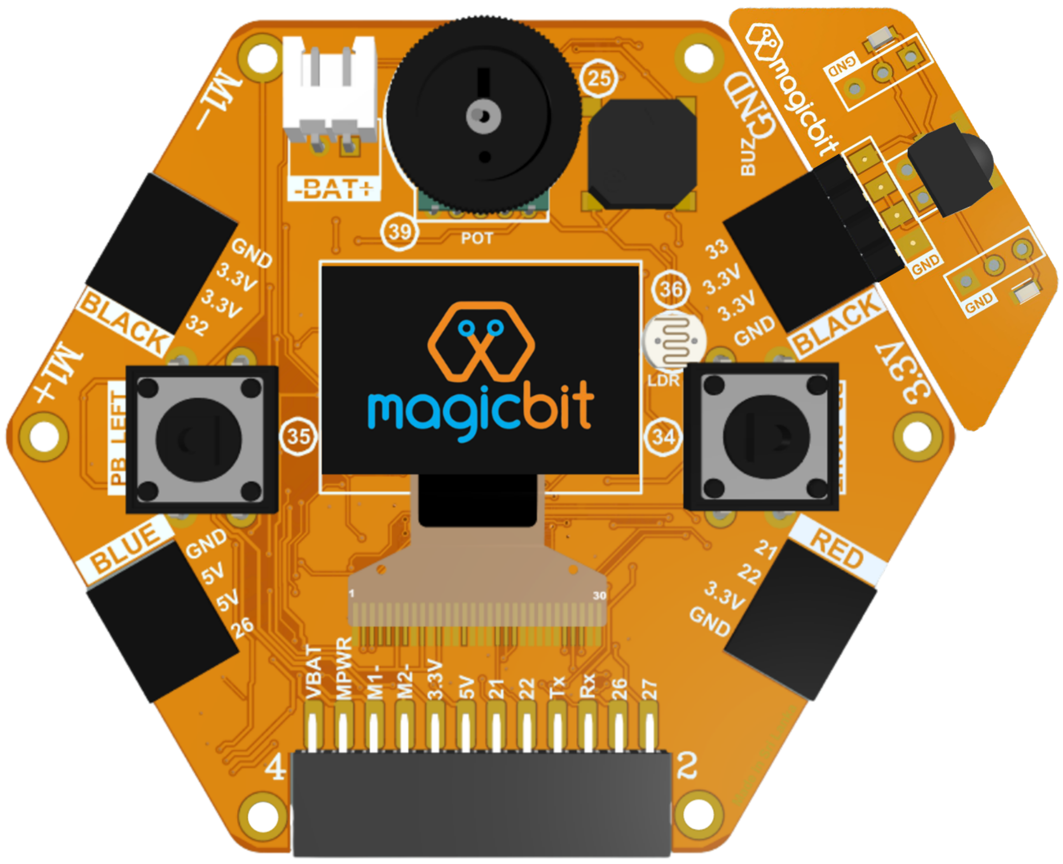


This block will read IR signal using when triggered and save it locally in the FLASH memory of the Magicbit. Two different IR commands can be stored in two memory banks.

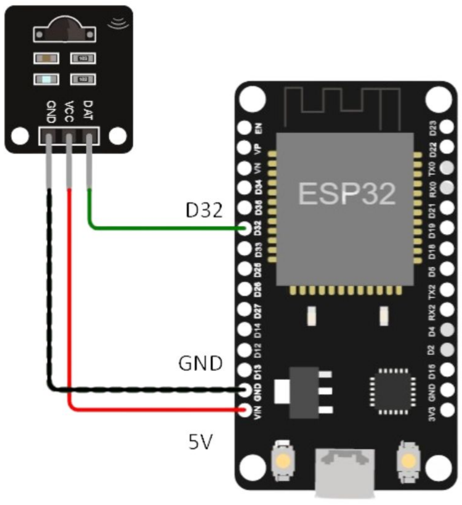
- **Configuration:**
  - Name: Any name desired
  - Device ID: Select device from the drop-down menu
  - Pin: Select the Pin to which the sensor is connected from drop down list

- Command: Select which bank (1 or 2) the IR command should be saved to

- **Input**
  - Any input trigger pulse
- **Output**
  - When an IR command is saved, the node will output msg.payload as { ir_recv: 1 , ir_bank: 2 }
- Connection
  - Magicbit



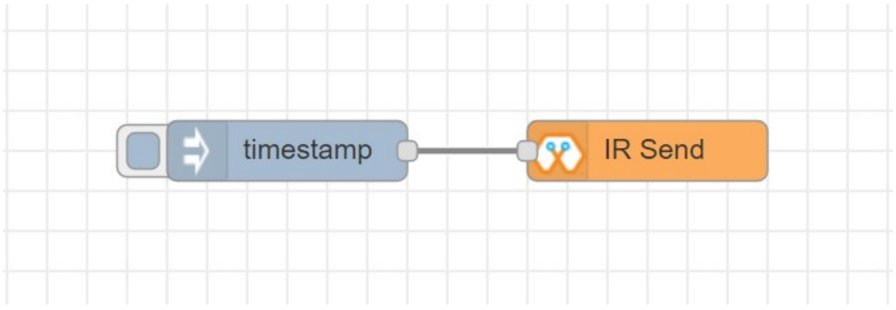
  - Generic ESP32



Import the already setup blocks below to try out the function of the IR Read block:

```
[{"id":"eb367cd3.cfece","type":"inject","z":"8e70cb96.beec38","name":"","topic":"",
↪"payload":"","payloadType":"date","repeat":"1","crontab":"","once":false,"onceDelay":0.
↪1,"x":150,"y":300,"wires":[["165974e1.48f67b"]]},{"id":"165974e1.48f67b","type":"IR_IN
↪","z":"8e70cb96.beec38","name":"","epId":"","pin":"32","bank":1,"x":320,"y":300,"wires
↪":[["c7413bd9.7d5f98"]]},{"id":"c7413bd9.7d5f98","type":"debug","z":"8e70cb96.beec38",
↪"name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":
↪"false","x":510,"y":300,"wires":[]}]
```

6.12 IR Send



This block will transmit IR signal when triggered using the IR transmitter. Two different IR commands can be saved and transmitted.

- **Configuration:**
  - Name: Any name desired
  - Device ID: Select device from the drop-down menu
  - Pin: Select the Pin to which the sensor is connected from drop down list

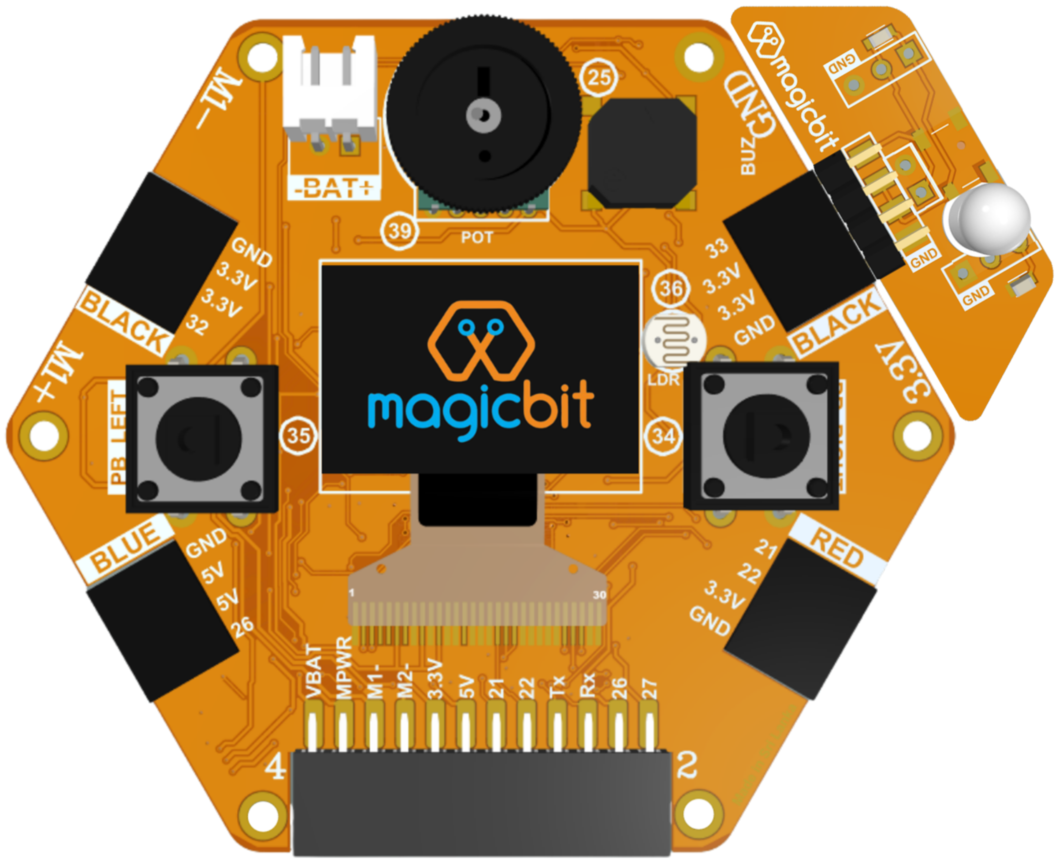
    – Command: Select which bank (1 or 2) the IR command should be saved to

- **Input**
    – Any input trigger pulse

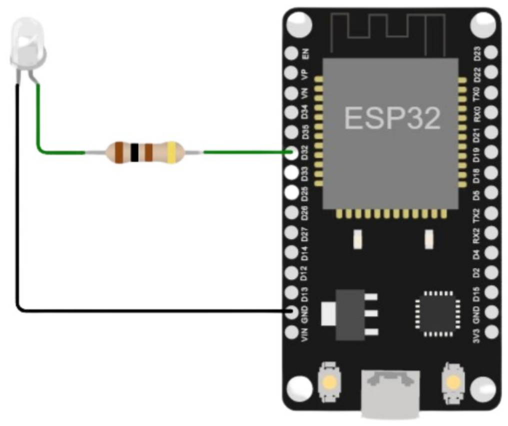
Import the already setup blocks below to try out the function of the IR Read block:

```
[{"id":"eb367cd3.cfece","type":"inject","z":"8e70cb96.beec38","name":"","topic":"",
↪"payload":"","payloadType":"date","repeat":"","crontab":"","once":false,"onceDelay":0.
↪1,"x":140,"y":300,"wires":[["8c0585cf.a31628"]]},{"id":"8c0585cf.a31628","type":"IR_OUT
↪","z":"8e70cb96.beec38","name":"","epId":"","pin":"33","bank":1,"x":340,"y":300,"wires
↪":[]}]
```

- Connection
    – Magicbit



    – Generic ESP32



The transmission range can be improved by connecting the IR emitter through a transistor.

CHAPTER

SEVEN

CORE BLOCKS

The Magicblocks palette includes a default set of nodes that are the basic building blocks for creating flows. This page highlights the core set you should know about.

All nodes include documentation you can see in the Info sidebar tab when you select a node.

- Inject
- Debug
- Function
- Change
- Switch
- Template

7.1 Inject node

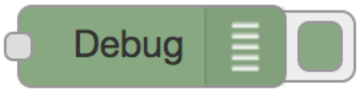
The Inject node can be used to manual trigger a flow by clicking the node's button within the editor. It can also be used to automatically trigger flows at regular intervals.

The message sent by the Inject node can have its **payload** and **topic** properties set.

The **payload** can be set to a variety of different types:

- a flow or global context property value
- a String, number, boolean, Buffer or Object
- a Timestamp in milliseconds since January 1st, 1970

7.2 Debug node



The Debug node can be used to display messages in the Debug sidebar within the editor.

The sidebar provides a structured view of the messages it is sent, making it easier to explore the message.

Alongside each message, the debug sidebar includes information about the time the message was received and which Debug node sent it. Clicking on the source node id will reveal that node within the workspace.

The button on the node can be used to enable or disable its output. It is recommended to disable or remove any Debug nodes that are not being used.

The node can also be configured to send all messages to the runtime log, or to send short (32 characters) to the status text under the debug node.

The page on Working with messages gives more information about using the Debug sidebar.

7.3 Function node

The Function node allows JavaScript code to be run against the messages that are passed through it.

A complete guide for using the Function node is available here .

7.4 Change node

The Change node can be used to modify a message's properties and set context properties without having to resort to a Function node.

Each node can be configured with multiple operations that are applied in order. The available operations are:

- **Set** - set a property. The value can be a variety of different types, or can be taken from an existing message or context property.
- **Change** - search and replace parts of a message property.
- **Move** - move or rename a property.
- **Delete** - delete a property.
- **When** setting a property, the value can also be the result of a JSONata expression. JSONata is a declarative query and transformation language for JSON data.

7.5 Switch node

The Switch node allows messages to be routed to different branches of a flow by evaluating a set of rules against each message.

The node is configured with the property to test - which can be either a message property or a context property.

There are four types of rule:

- **Value** rules are evaluated against the configured property
- **Sequence** rules can be used on message sequences, such as those generated by the Split node
- A JSONata **Expression** can be provided that will be evaluated against the whole message and will match if the expression returns a _true_ value.
- An **Otherwise** rule can be used to match if none of the preceding rules have matched.

The node will route a message to all outputs corresponding to matching rules. But it can also be configured to stop evaluating rules when it finds one that matches.

CHAPTER

EIGHT

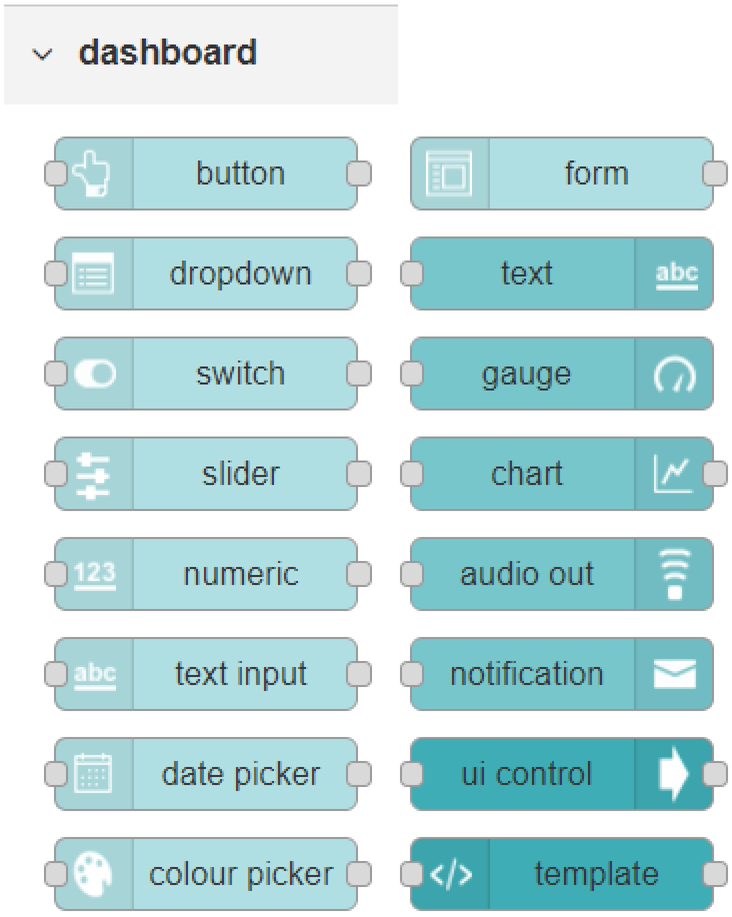
DASHBOARDS

8.1 What is Magicblocks Dashboard?

Magicblocks Dashboard is a module that provides a set of nodes in **Magicblocks** to quickly create a live data dashboard.

Since **Magicblocks** is based on NODE-RED opensource platform you can learn more about dashboard using following links

- http://flows.nodered.org/node/node-red-dashboard
- https://github.com/node-red/node-red-dashboard



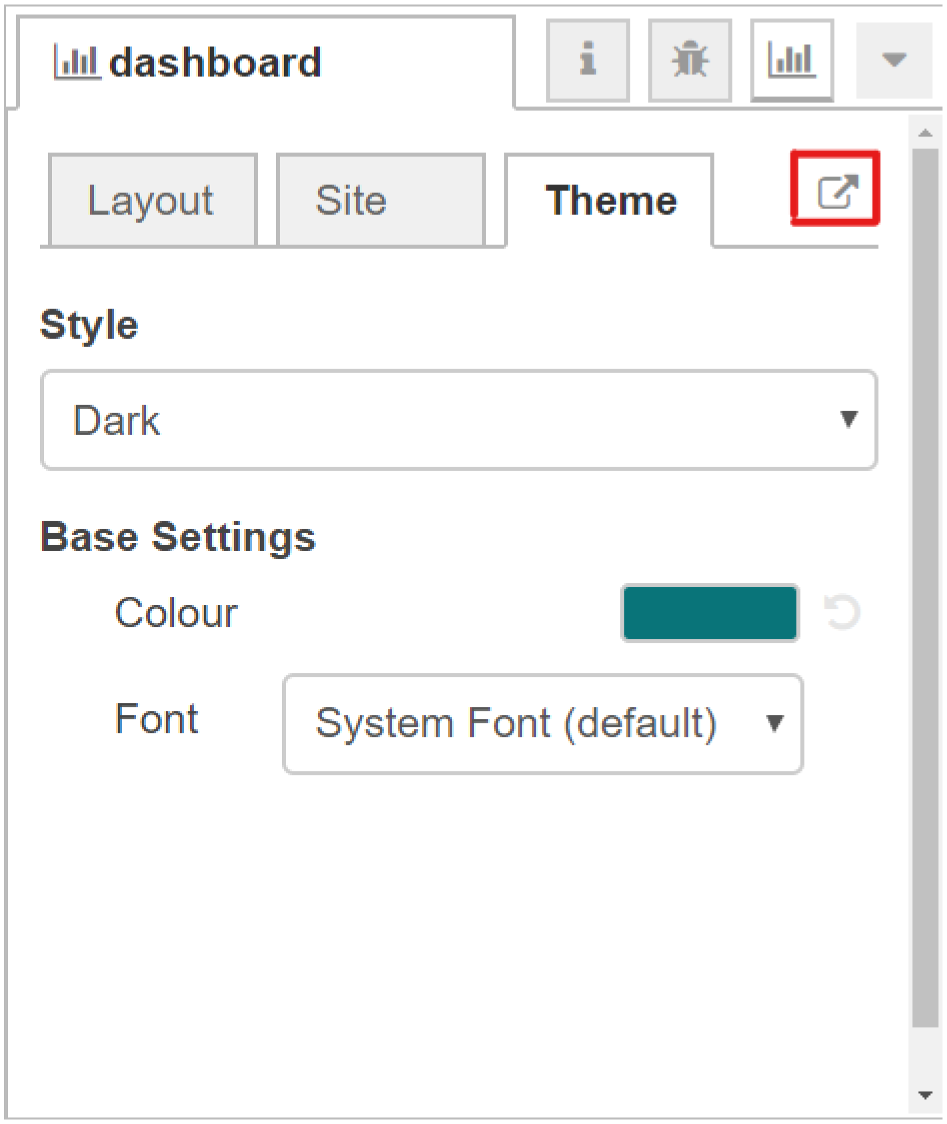
Nodes from the dashboard section provide widgets that show up in your application user interface (UI).

The user interface is organized in tabs and groups. Tabs are different pages on your user interface, like several tabs in a browser. Inside each tab you have groups that divide the tabs in different sections so that you can organize your widgets.

Every widget should have an associated group that determines where the widget should appear on the user interface.

To create a tab and a group follow the following instructions (see figure below):

- On top right corner of the **Magicblocks** window you have a tab called dashboard.
- Select that tab **(1)**. To add a tab to the user interface click on the +tab button **(2)**.
- Once created, you can edit the tab by clicking on the edit button **(3)**



Style

Dark

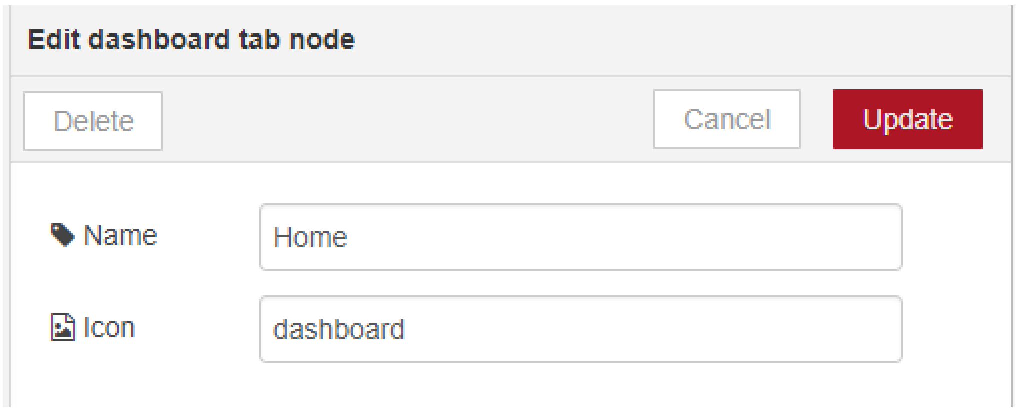
Base Settings

Colour

Font System Font (default)

You can edit the tab's name and change its icon

- Name: you can call it whatever you want
- Icon: you should use a name accordingly to the icon's names in this link: https://klarsys.github.io/angular-material-icons

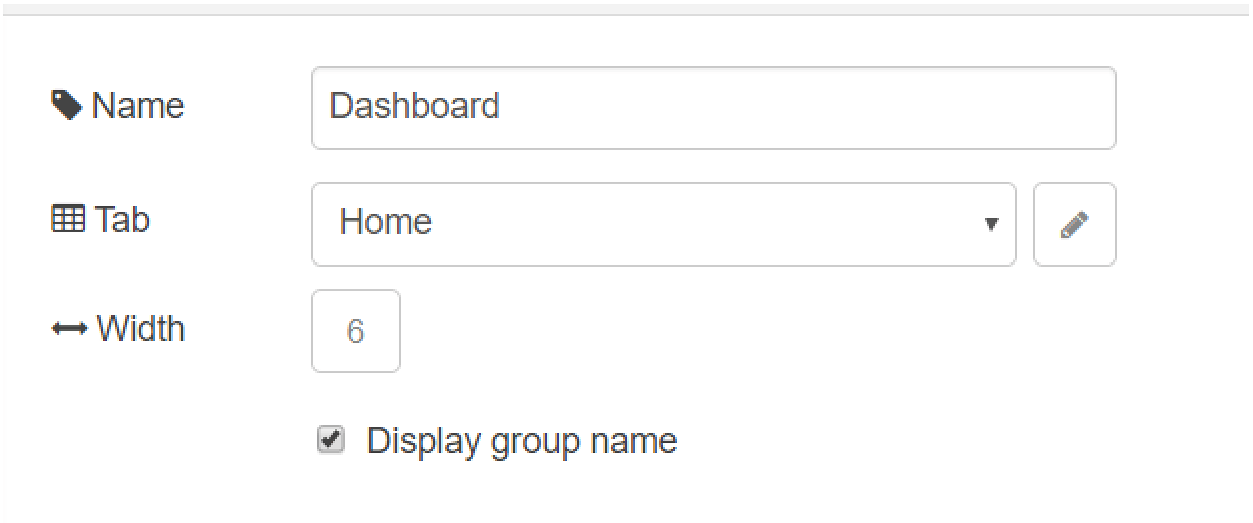

Edit dashboard tab node

Delete Cancel Update

Name: Home

Icon: dashboard

- After creating a tab, you can create several groups under that tab. You need to create at least one group to add your widgets. To add a group to the created tab, you need to click on the +group button **(4)**.
- Then, you can edit the created group by clicking on the edit button **(5)**.


Name: Dashboard

Tab: Home

Width: 6

[✓] Display group name

- You can edit its name, select its corresponding tab and change its width.

8.2 Creating a User Interface – Example

In this section we're going to make a dashboard example to show you how you can build and edit your own dashboard – we won't actually add functionalities to the widgets – we'll do that in future projects. This dashboard will have the following features

- Two different tabs: one called Room and another called Garden
- The Room tab will have two groups and the Garden tab will have one group
- We'll add a color picker and a switch to the room groups
- We'll add a chart to the Garden group

8.3 Creating the Tabs

On the top right corner on the Magicblocks window, select the **dashboard** tab and create two new tabs by clicking on the **+tab** button.

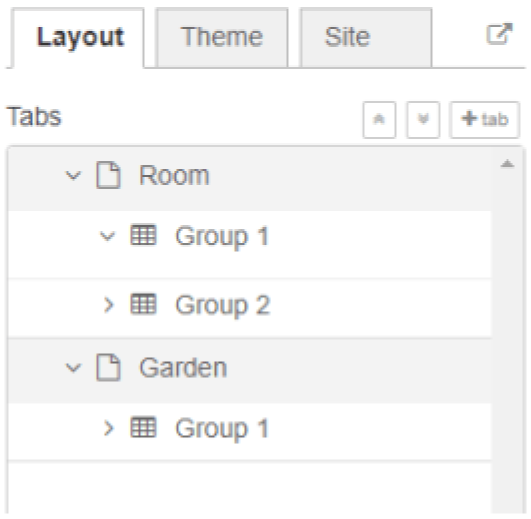
Edit one tab with the following properties

- **Name:** Room
- **Icon:** tv

And the other one with the following

- Name: Garden
- Icon: local_florist

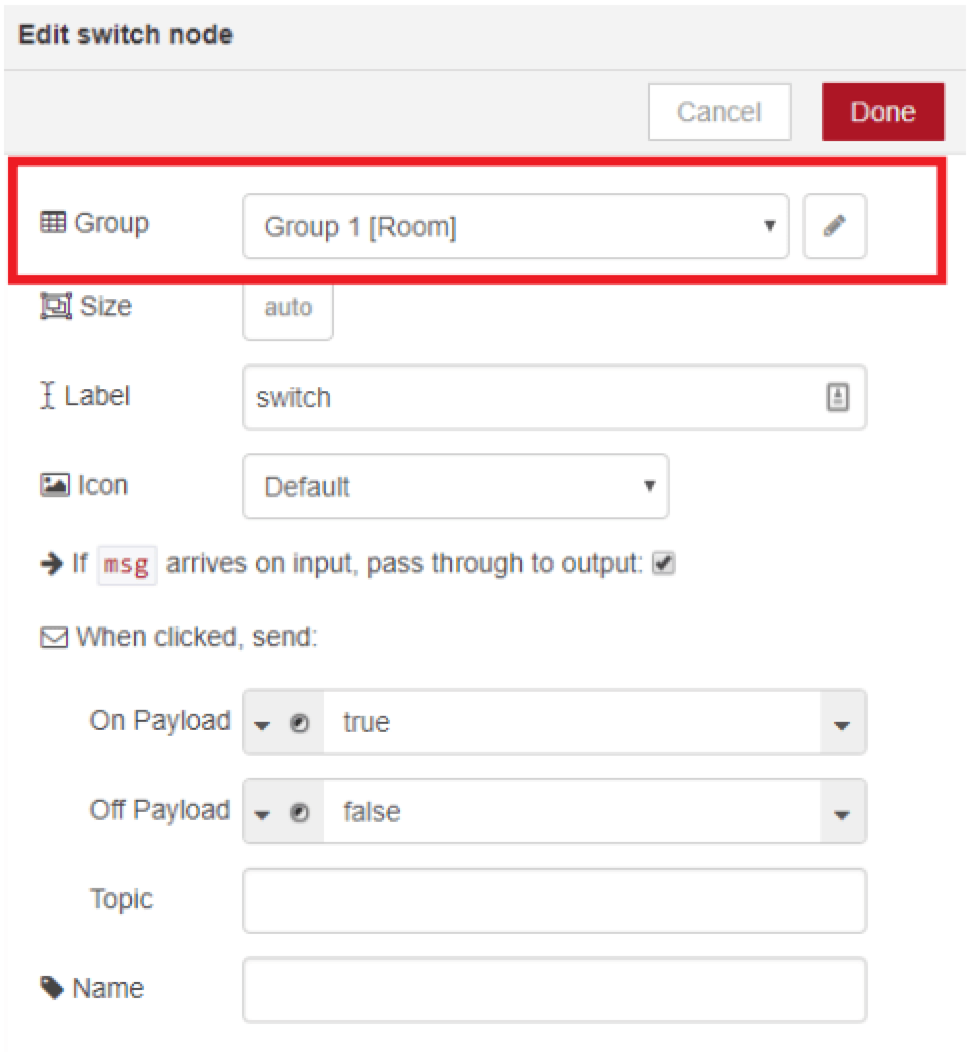
Then, add two groups to the Room tab and one group to the Garden tab. The following figure shows how your dashboard layout looks.



8.4 Adding the Widgets

Add a switch, a slider, a colour picker and a gauge to the flow as show in the following figure .. image:: https://github.com/magicbitlk/Magicbit-Magicblocks.io/blob/master/Images/flow.png?raw=true

Double click on the switch. A new window pops up.


Edit switch node

Cancel Done

Group: Group 1 [Room]

Size: auto

Label: switch

Icon: Default

If msg arrives on input, pass through to output: ✓

When clicked, send:

On Payload: true

Off Payload: false

Topic:

Name:

In this new window you can choose where you want your button widget to appear. In this case we want it to appear in the Room tab, Group 1 as highlighted in red in the previous figure.

Then, do the same for the other widgets but add them to the following groups:

- **slider:** Group 1 [Room]
- **color picker:** Group 2 [Room]
- **gauge:** Group 1 [Garden]

Source: (https://randomnerdtutorials.com/getting-started-with-node-red-dashboard/)