



UNIVERSITI SAINS ISLAM MALAYSIA  
جامعة العلوم الإسلامية الماليزية  
ISLAMIC SCIENCE UNIVERSITY OF MALAYSIA



## AULAD CAMP + STEM

# Jom Jadi Jurutera!

14 - 15 Disember 2022

Berilmu ■ Berdisiplin ■ Bertakwa / Knowledgeable ■ Disciplined ■ Devout

Universiti Sains Islam Malaysia, Bandar Baru Nilai, 71800, Nilai, Negeri Sembilan, Malaysia

[www.usim.edu.my](http://www.usim.edu.my)

[usimofficial](#)

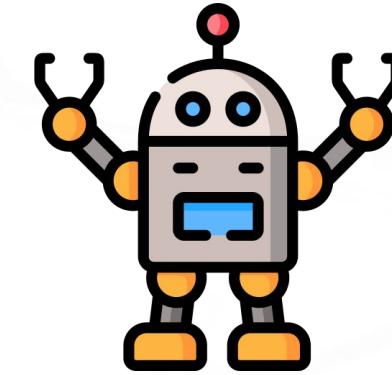
[usimtv](#)

# AULAD CAMP + STEM

# Jom Jadi Jurutera!

Rabu (14 Disember)	Aktiviti
8.00 pagi	Taklimat & Pengenalan
9.00 pagi	MyIoT
1.00 ptg	Rehat
2.00 ptg	MyRobotic
5.00 ptg	Bersurai

Khamis (15 Disember)	Aktiviti
8.00 pagi	Latihan Dalam Kumpulan
9.00 pagi	MySolar
1.00 ptg	Rehat
2.00 ptg	Persembahan / Pembentangan
4.00 ptg	Penutup & Sijil
5.00 ptg	Bersurai

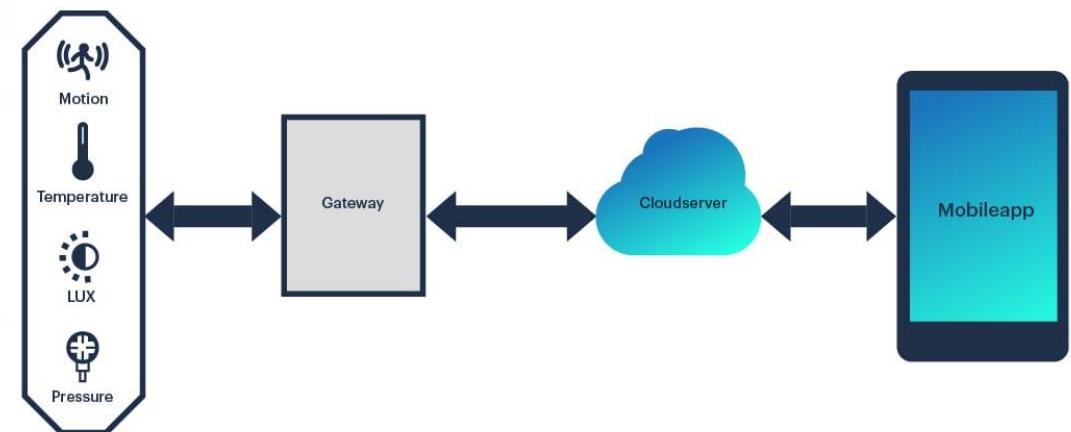


# Introduction of Internet of Things (IoT)



# What is IoT?

The Internet of Things (IoT) describes the network of physical objects—“things”—that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet[1].

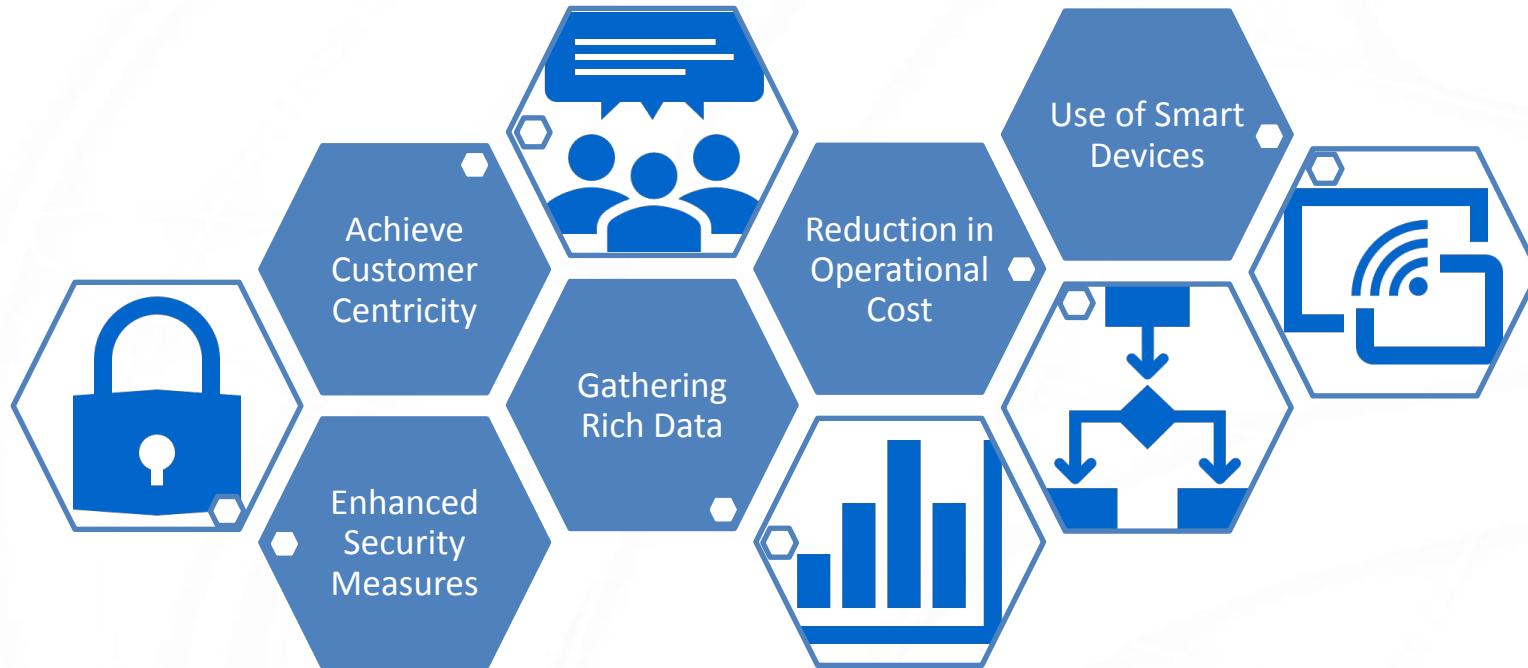


[1] “What is the internet of things (IOT)?,” *What Is the Internet of Things (IoT)?* | Oracle Malaysia. [Online]. Available: <https://www.oracle.com/my/internet-of-things/what-is-iot/>. [Accessed: 10-Dec-2022].



# Why IoT is important?

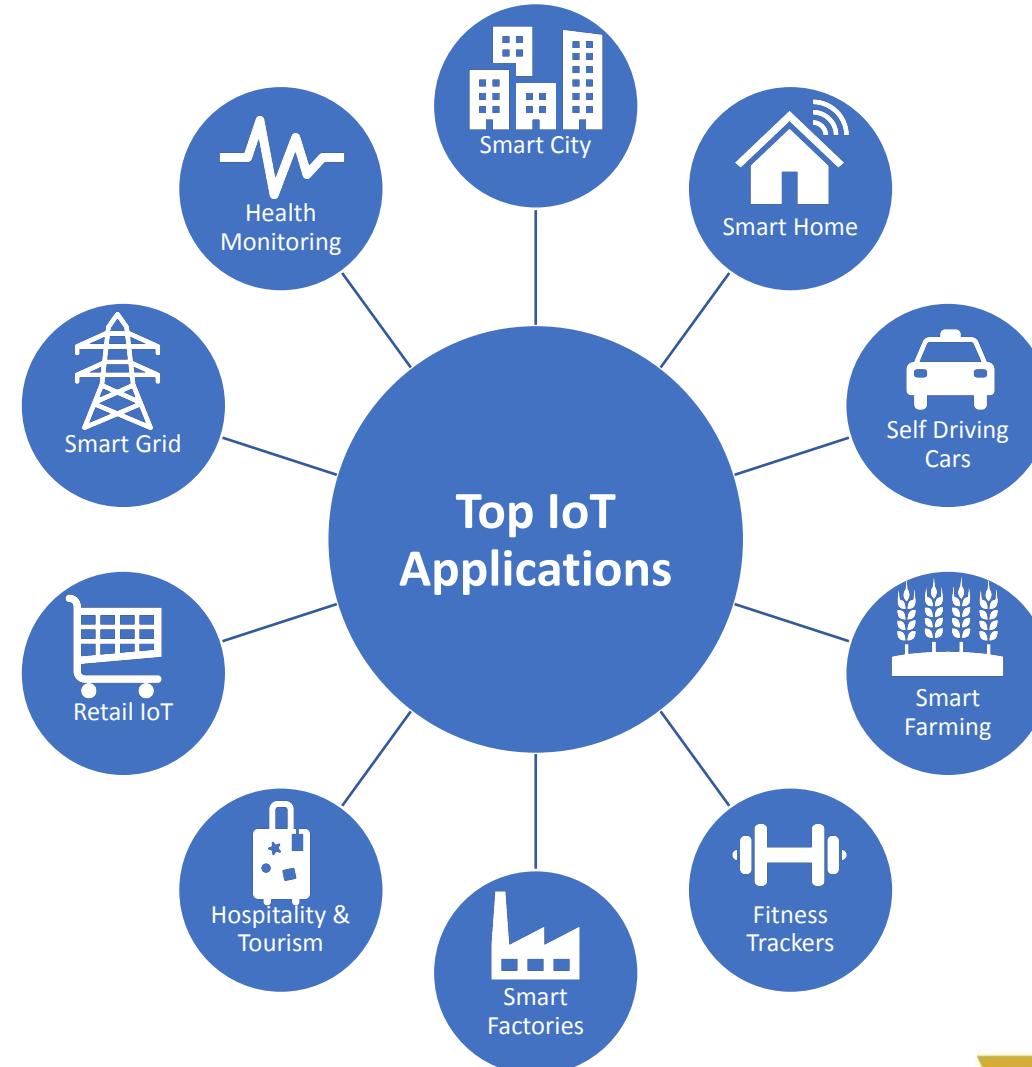
- Internet of Things help people live and work smarter, as well as gain complete control over their lives [2].



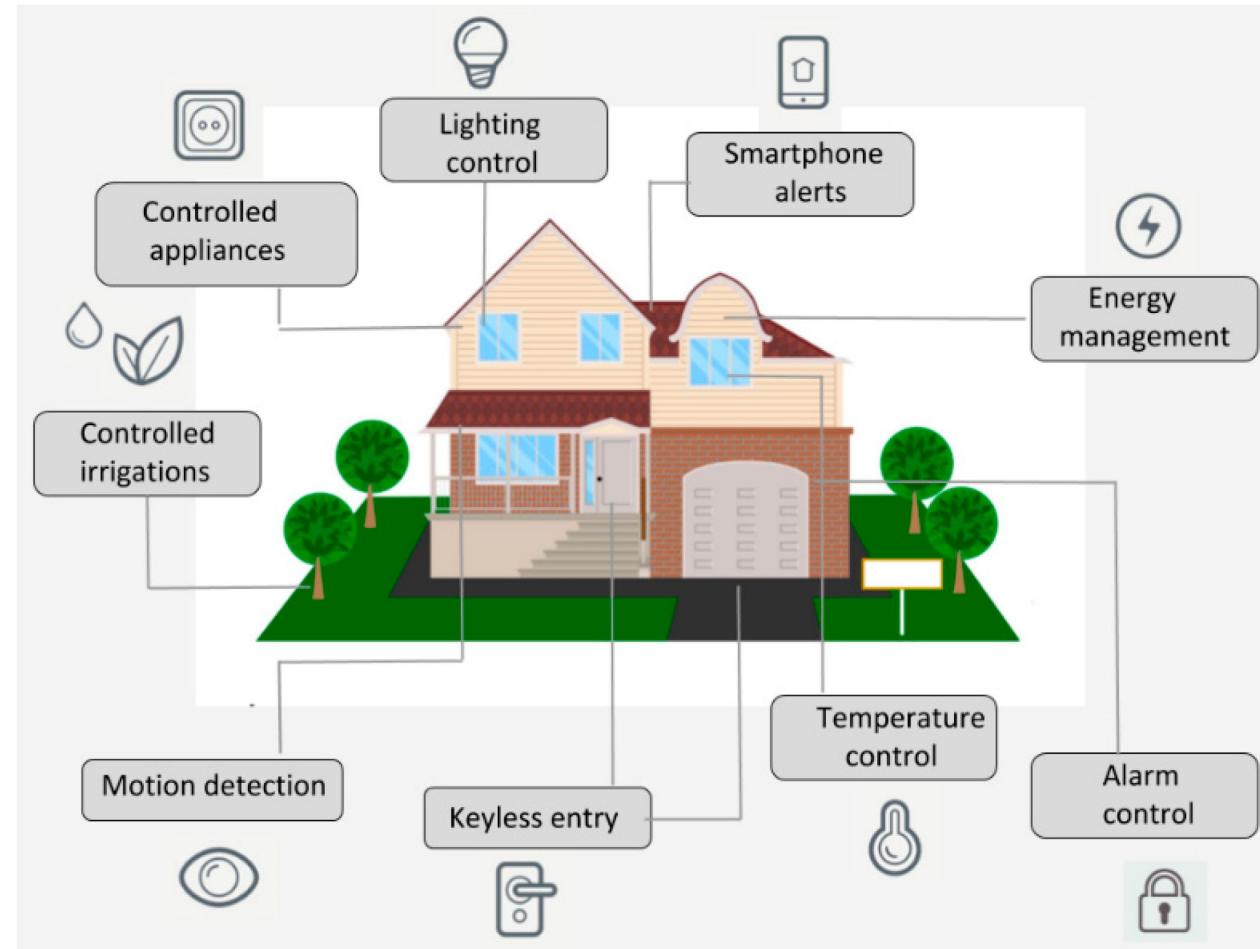
[2] Gillis, Alexander (2021). "What is internet of things (IoT)?". IOT Agenda. Retrieved 17 August 2021.



# What is the application of IoT?



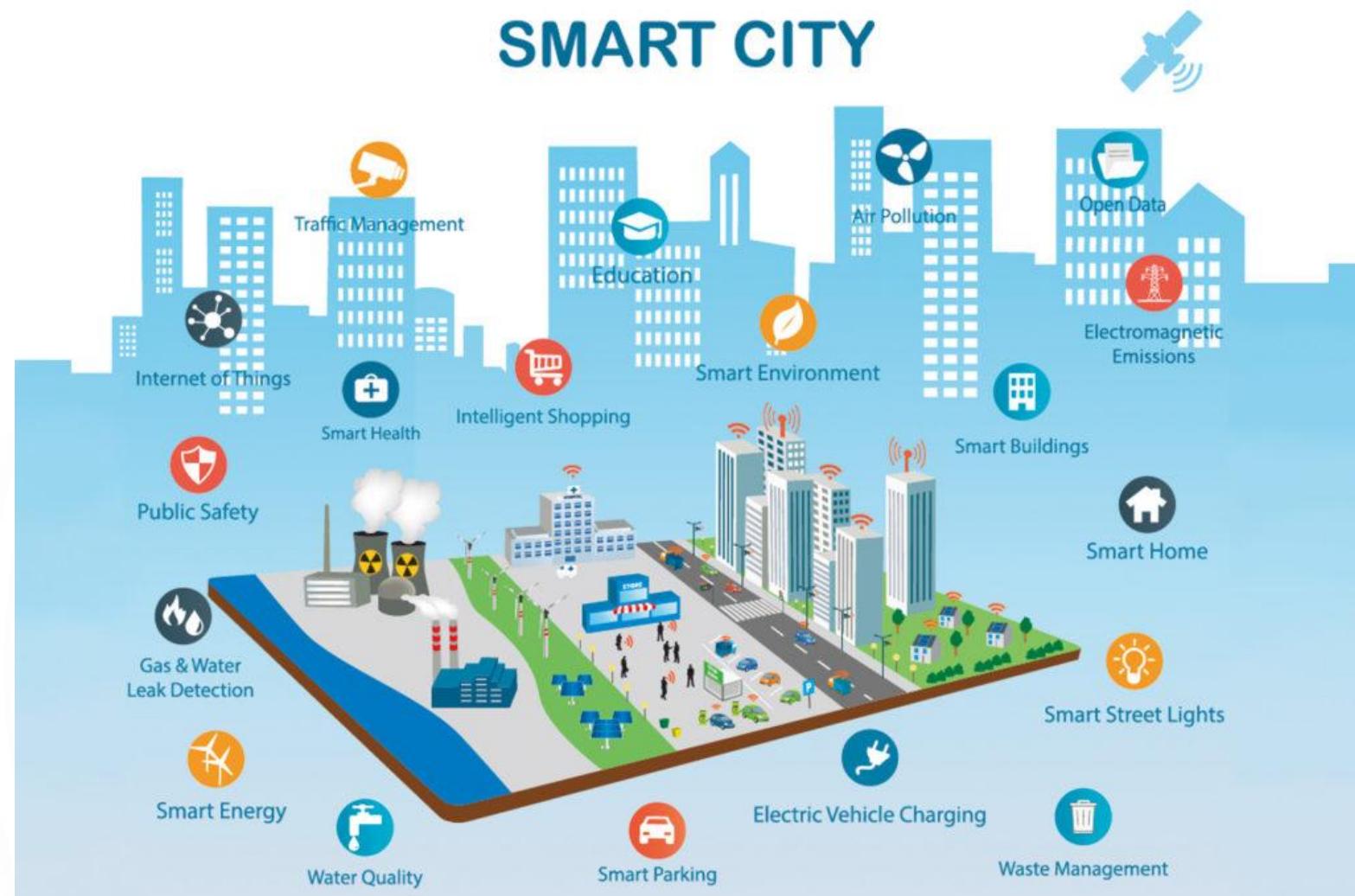
# Smart Home



[4] C. Stolojescu-Crisan, C. Crisan, and B.-P. Butunoi, "An IOT-based Smart Home Automation System," *MDPI*, 30-May-2021. [Online]. Available: <https://www.mdpi.com/1424-8220/21/11/3784>. [Accessed: 10-Dec-2022].



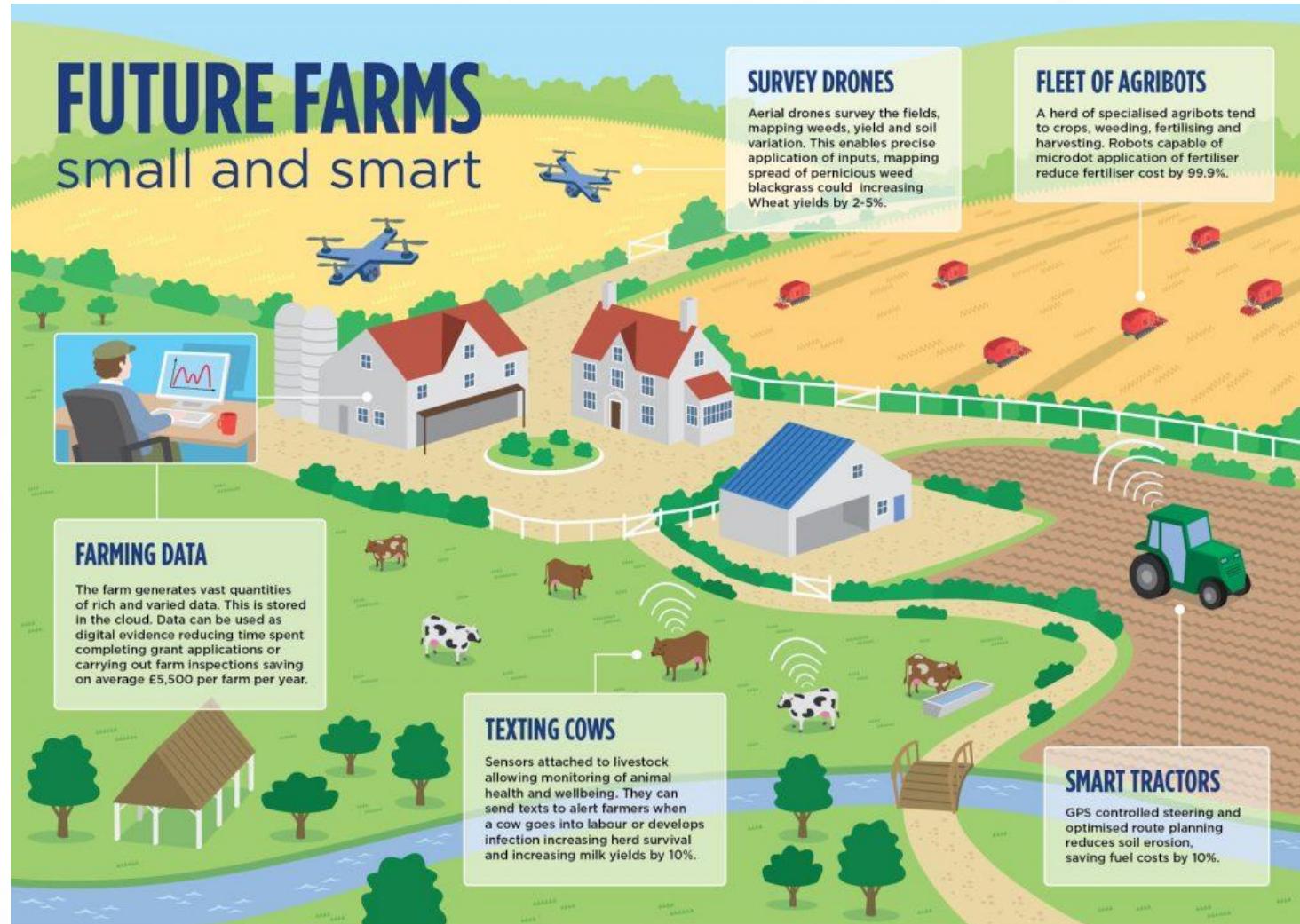
# Smart City



[3] N. Godse, "Smart cities of the future- powered by IOT," *Data Science Central*, 04-May-2022. [Online]. Available: <https://www.datasciencecentral.com/smart-cities-of-the-future-powered-by-iot/>. [Accessed: 10-Dec-2022].



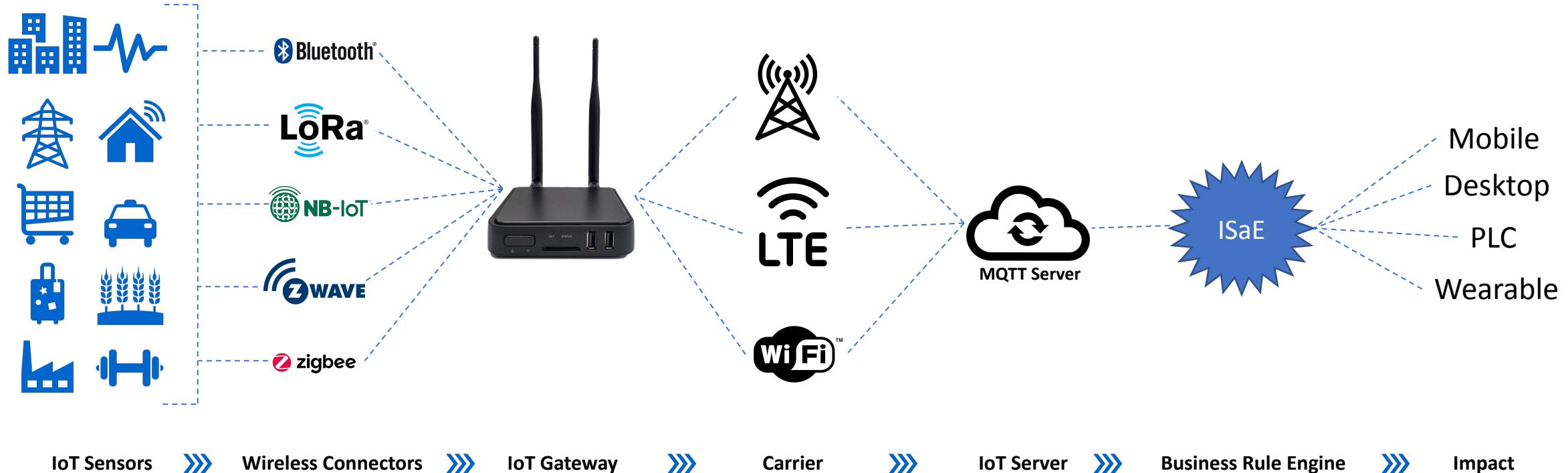
# Smart Farm



[5] R. Burnett, "Smart Farming - Digital Revolution of Agriculture," *MaxBotix Inc.*, 29-Aug-2022. [Online]. Available: <https://www.maxbotix.com/articles/smart-farming.htm>. [Accessed: 10-Dec-2022].



# How IoT works?



# IoT – Node-RED

#Practice Time!

# Demonstration

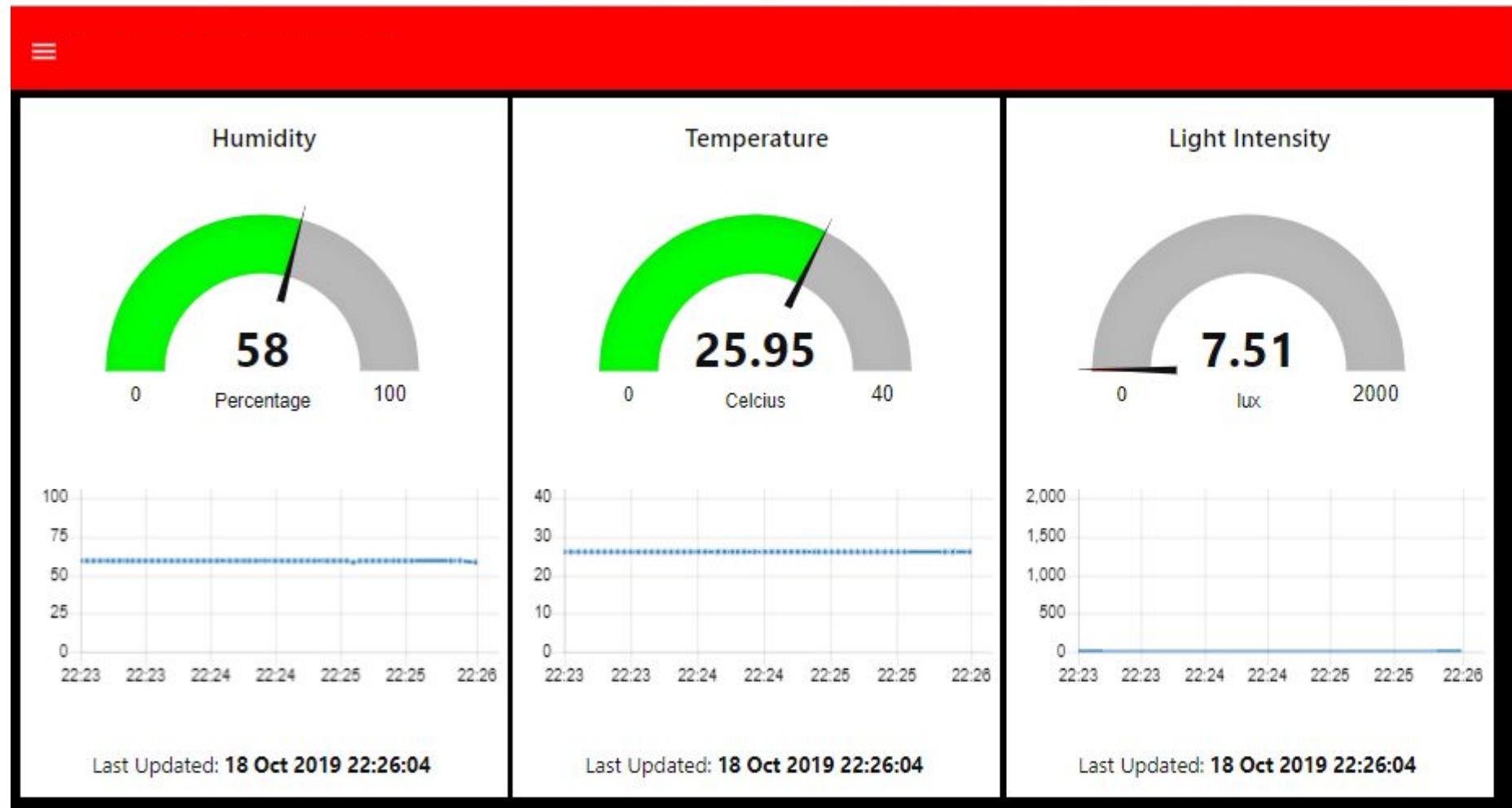
*Sensors*

*(Temperature, Humidity and Light  
intensity)*

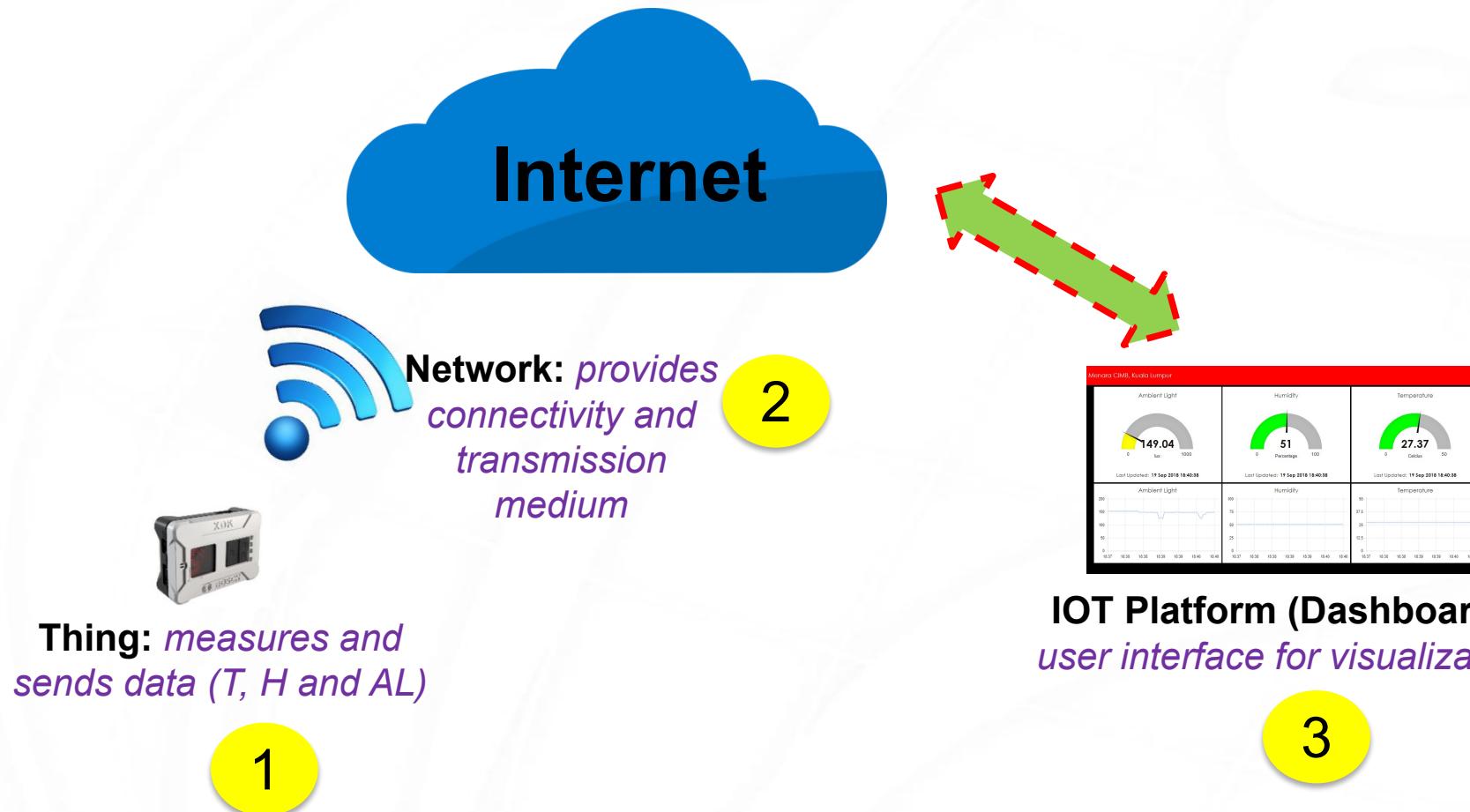
# XDK 110



# Dashboard

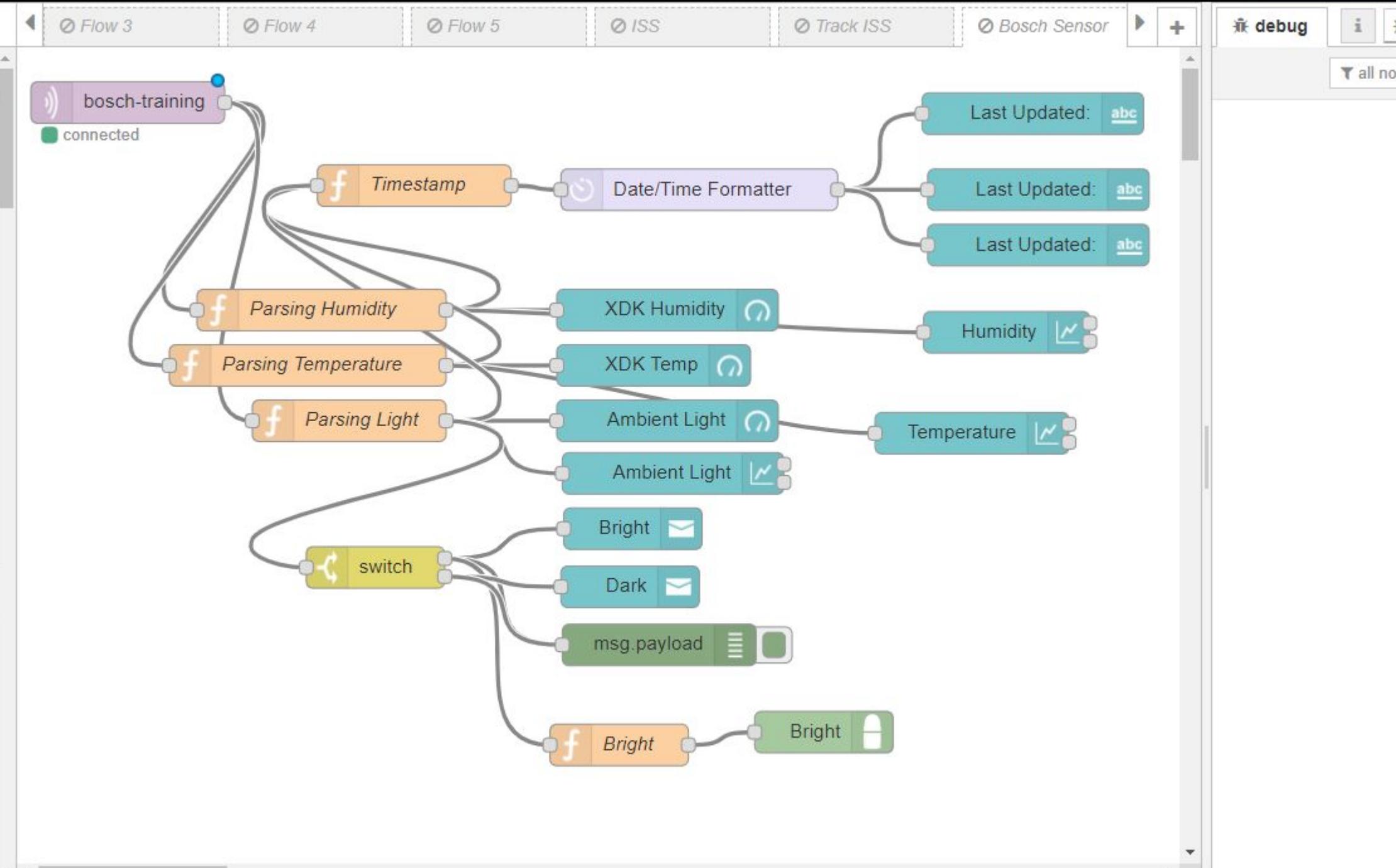


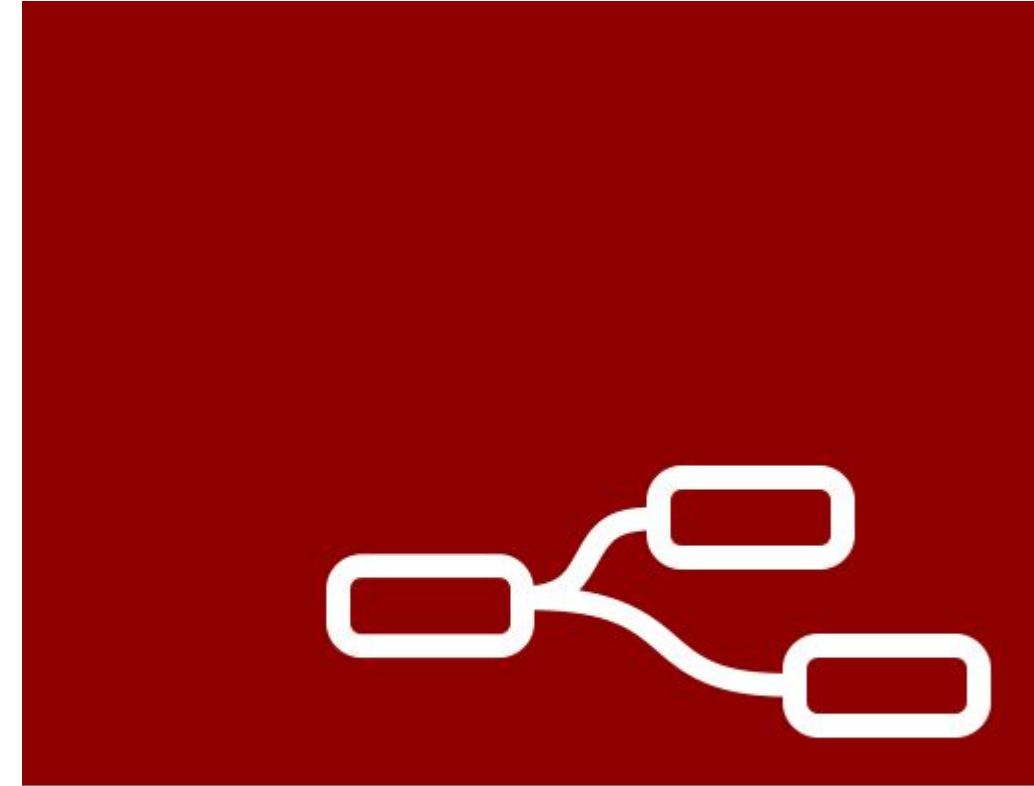
# Demo Setup



filter nodes

- input
  - inject
  - catch
  - status
  - link
  - mqtt
  - http
  - websocket
  - tcp
  - udp
- output
  - debug
  - link
  - mqtt
  - http response
  - websocket

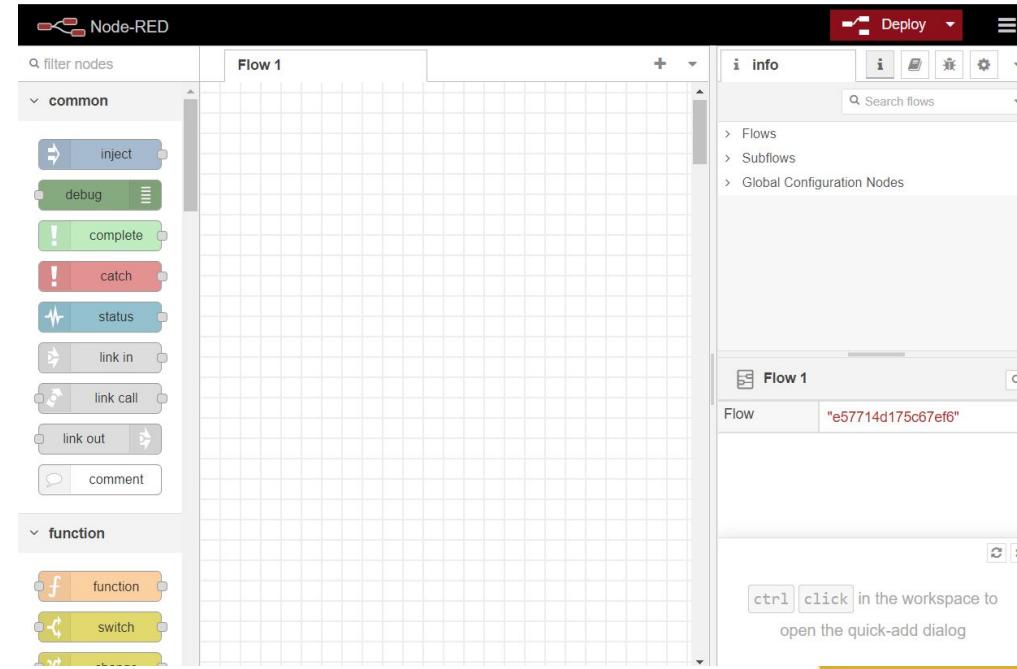




# Node-RED

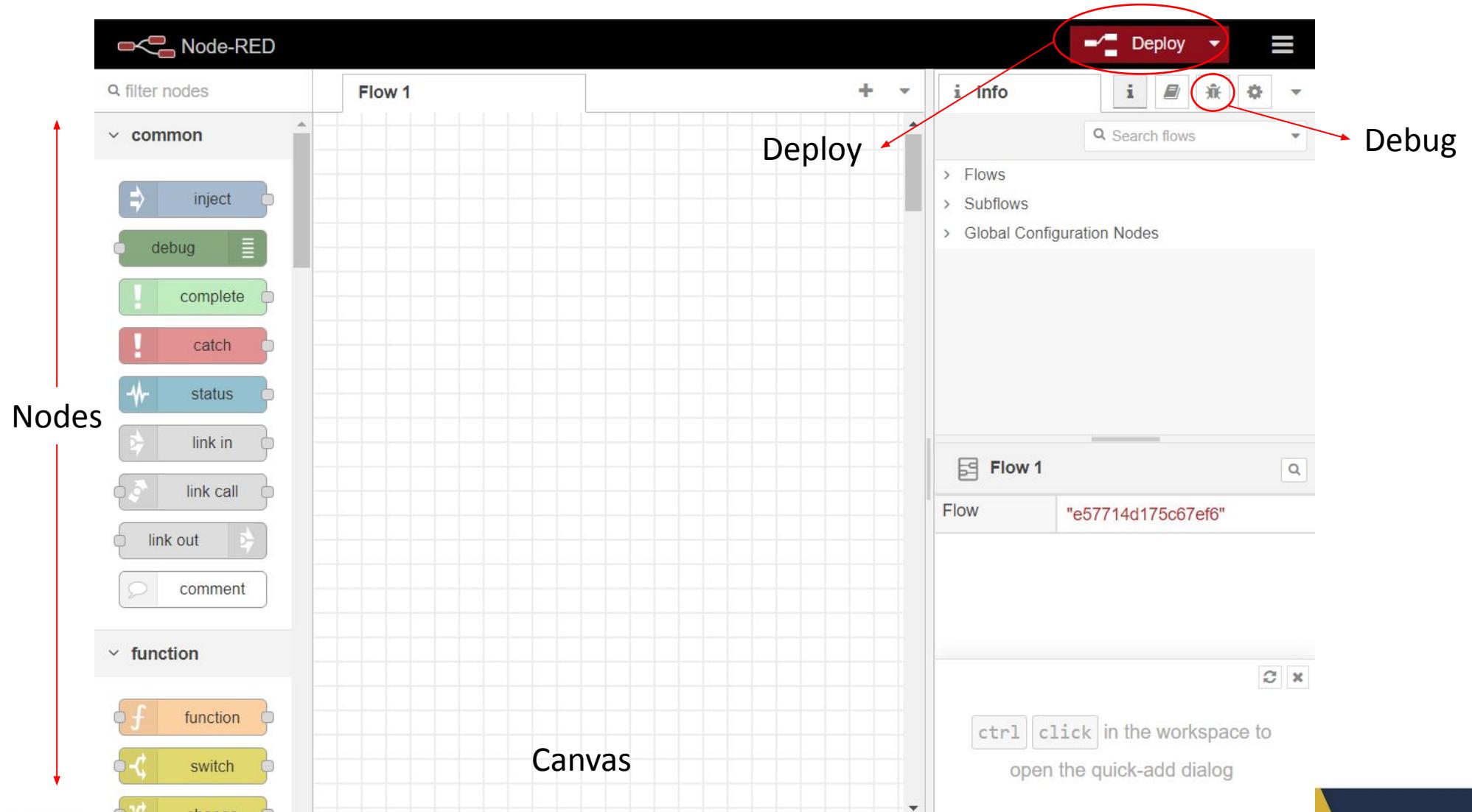
# IoT – Node-RED

- **Node-RED** is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.
- It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.



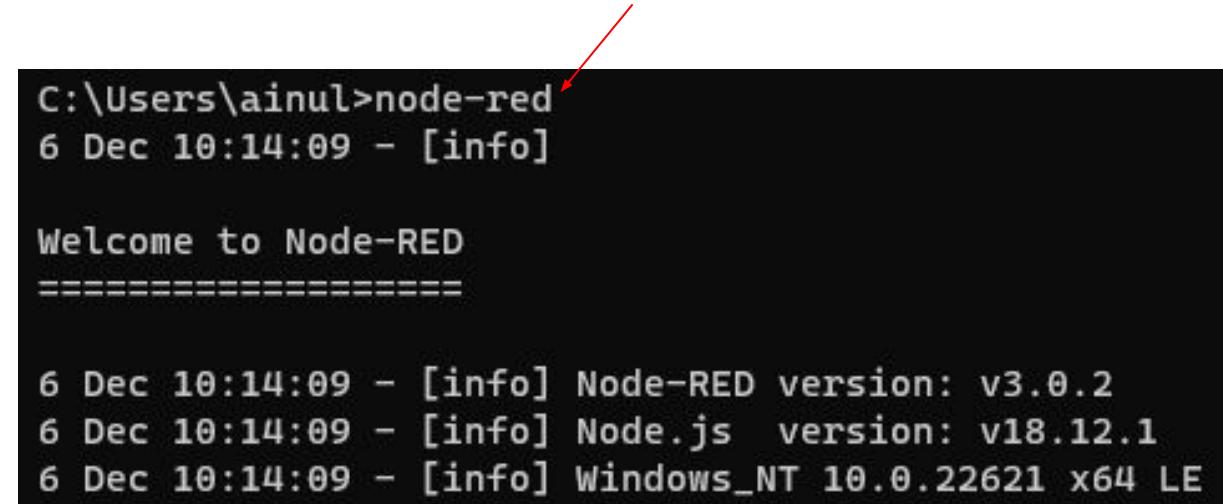
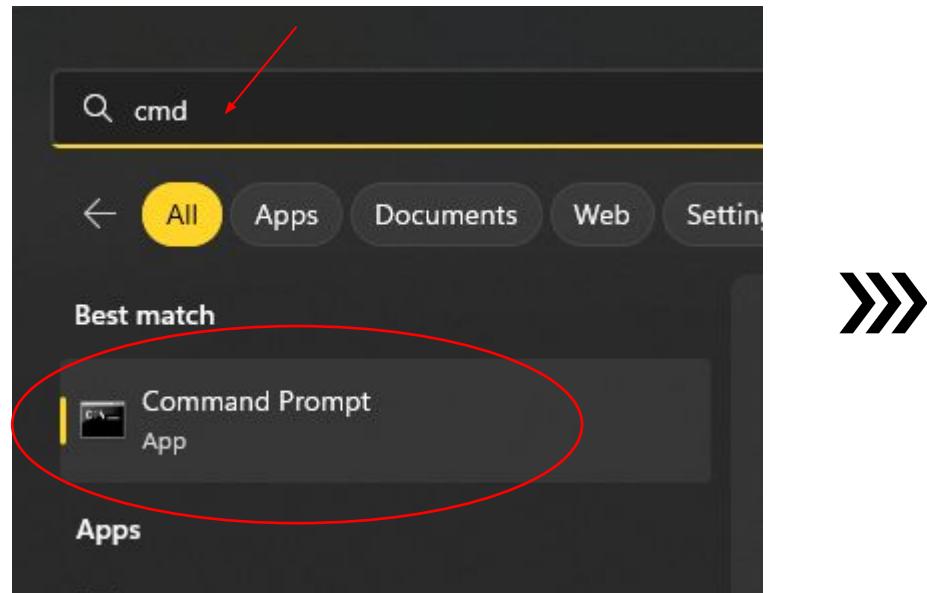
IBM

# IoT – Node-RED



# IoT – Node-RED

1. Press Win key and type “cmd”
2. Open Command Prompt
3. Type “node-red”
4. Press Enter



```
C:\Users\ainul>node-red
6 Dec 10:14:09 - [info]

Welcome to Node-RED
=====
6 Dec 10:14:09 - [info] Node-RED version: v3.0.2
6 Dec 10:14:09 - [info] Node.js version: v18.12.1
6 Dec 10:14:09 - [info] Windows_NT 10.0.22621 x64 LE
```

# IoT – Node-RED

1. Copy the URL
2. Open browser and paste the URL
3. Press Enter

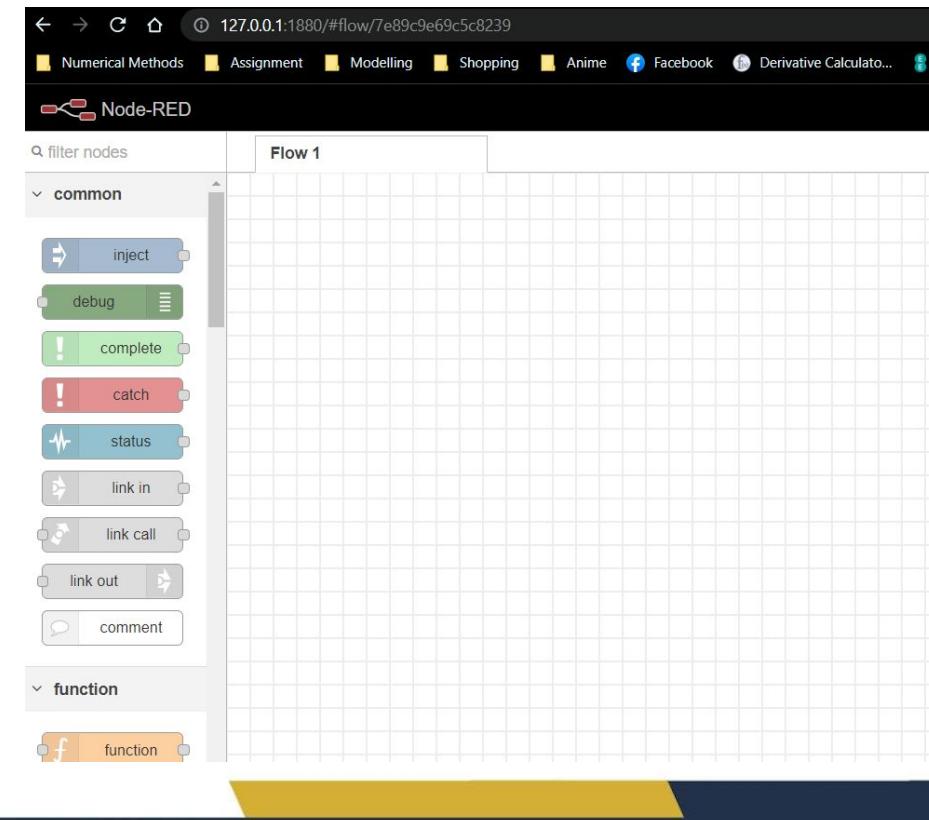
```
Welcome to Node-RED
-----
12 Dec 23:26:44 - [info] Node-RED version: v3.0.2
12 Dec 23:26:44 - [info] Node.js version: v18.12.1
12 Dec 23:26:44 - [info] Windows_NT 10.0.22621 x64 LE
12 Dec 23:26:45 - [info] Loading palette nodes
12 Dec 23:26:46 - [info] Worldmap version 2.31.3
12 Dec 23:26:46 - [info] Dashboard version 3.2.3 started at /ui
12 Dec 23:26:46 - [info] Settings file : C:\Users\ainul\.node-red\settings.js
12 Dec 23:26:46 - [info] Context store : 'default' [module=memory]
12 Dec 23:26:46 - [info] User directory : \Users\ainul\.node-red
12 Dec 23:26:46 - [warn] Projects disabled : editorTheme.projects.enabled=false
12 Dec 23:26:46 - [info] Flows file : \Users\ainul\.node-red\flows.json
12 Dec 23:26:46 - [info] Server now running at http://127.0.0.1:1880/
12 Dec 23:26:46 - [warn]

-----
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

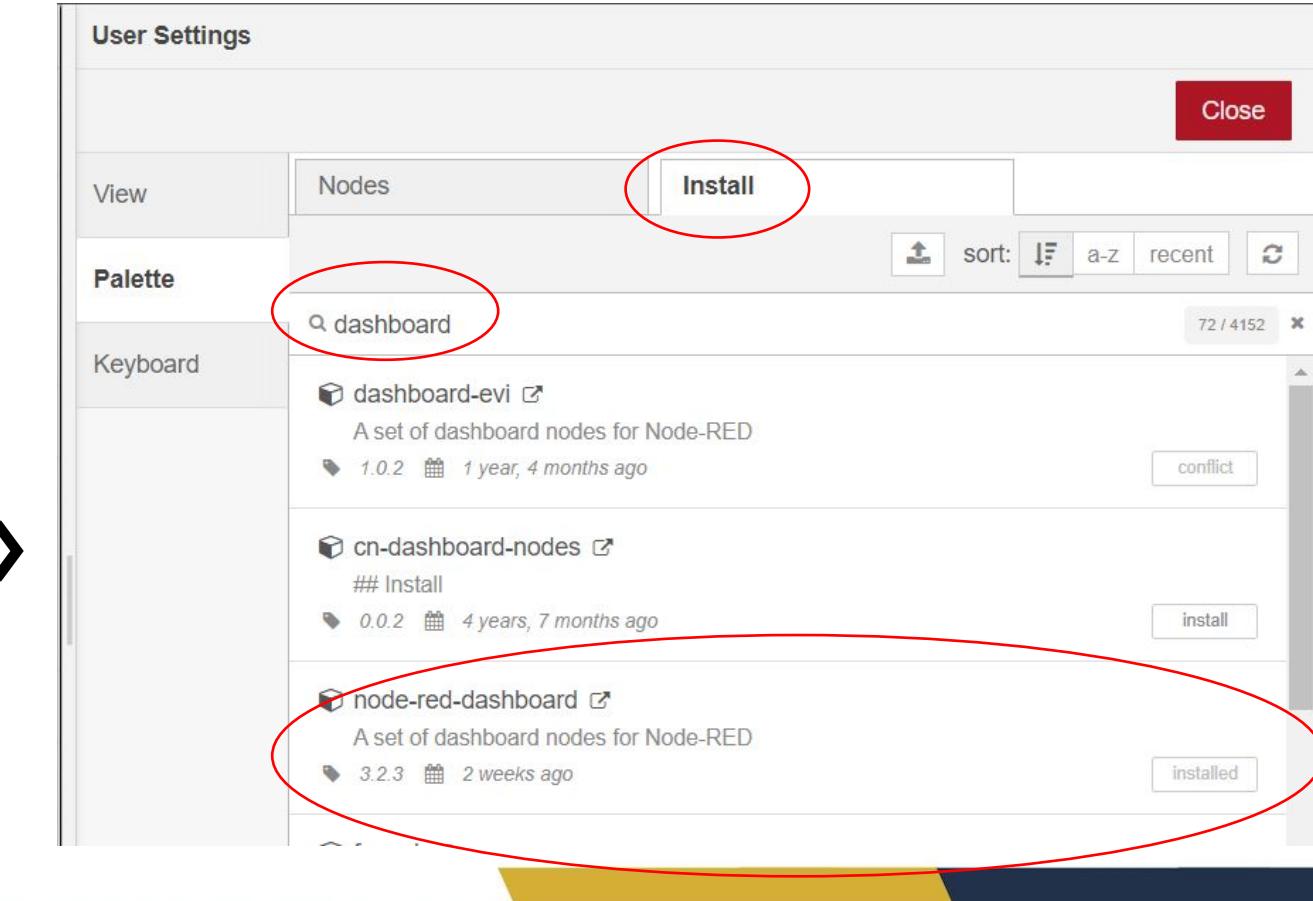
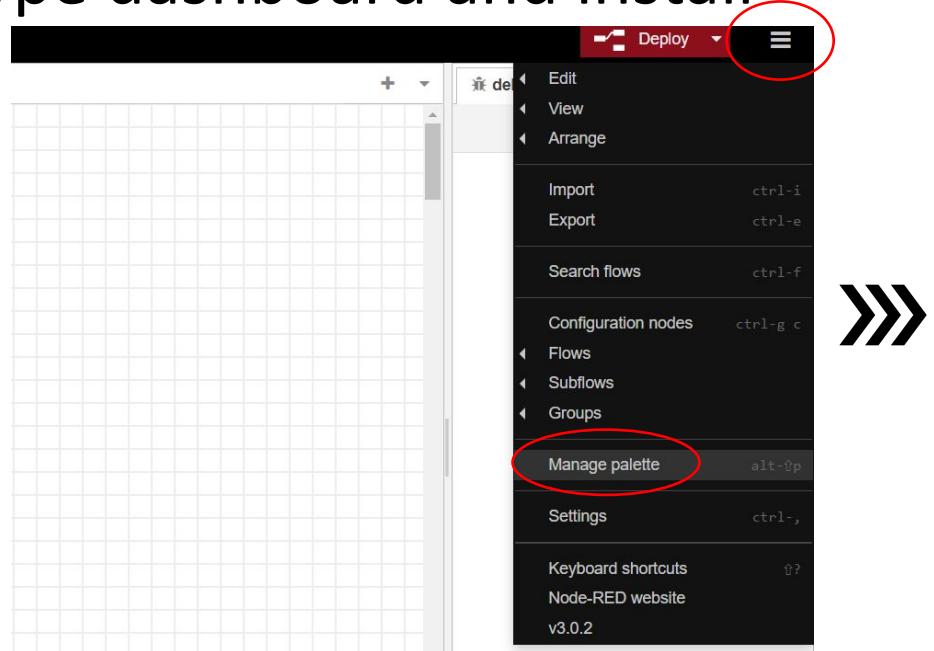
You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.

-----
12 Dec 23:26:46 - [info] Starting flows
```



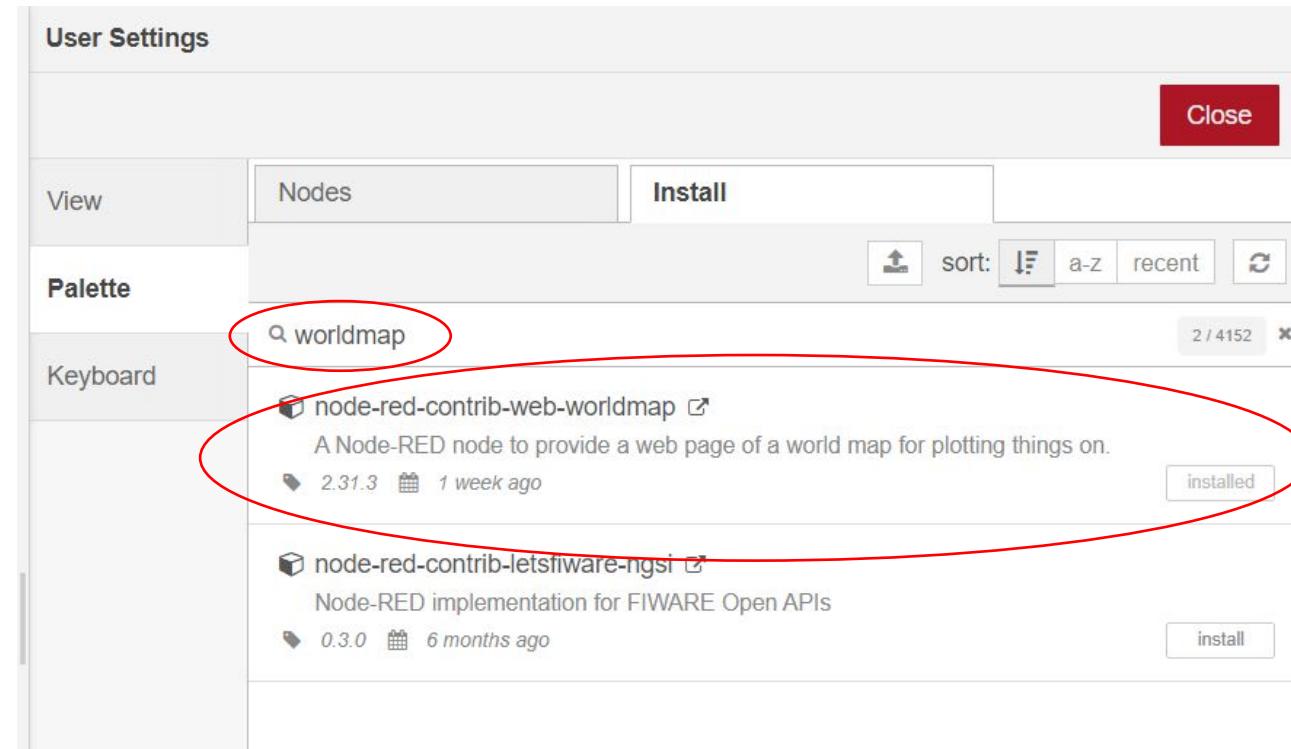
# IoT – Node-RED

1. Press the triple line
2. Click manage palette
3. Click install
4. Type dashboard and install



# IoT – Node-RED

1. Type worldmap
2. Install
3. Click Close



# Task #1

## Hello World!



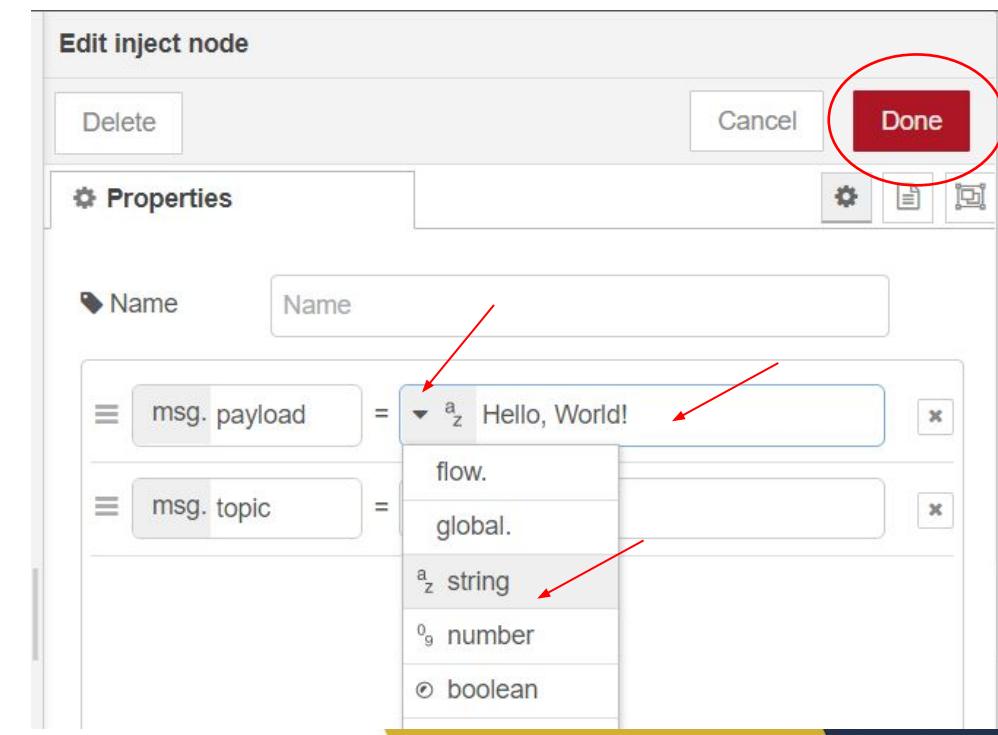
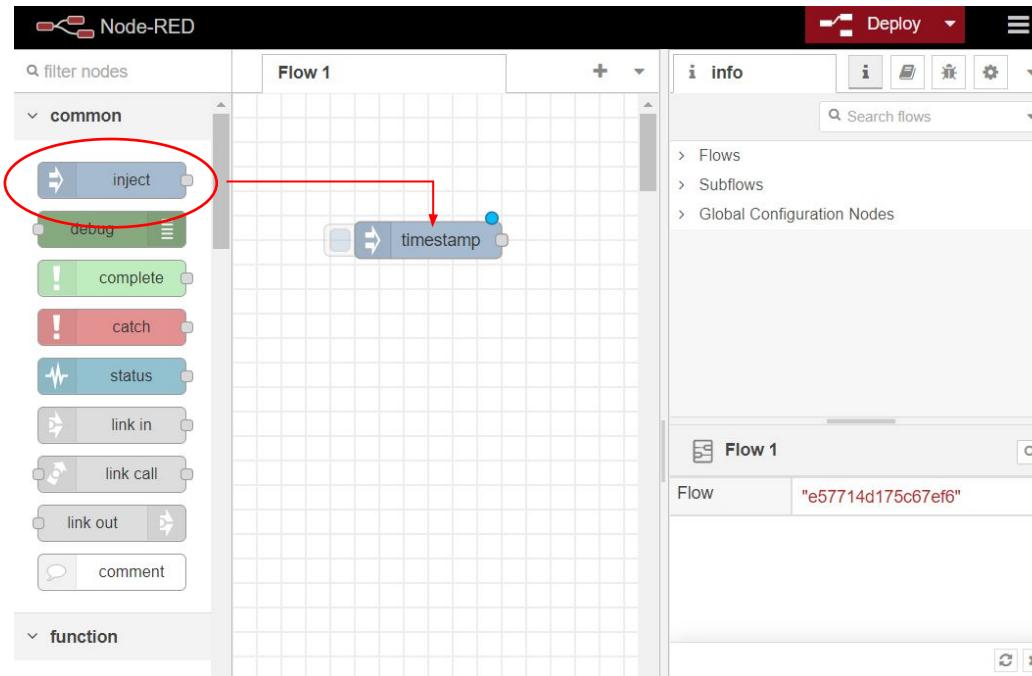


**<hello-world/>**

# IoT – Node-RED

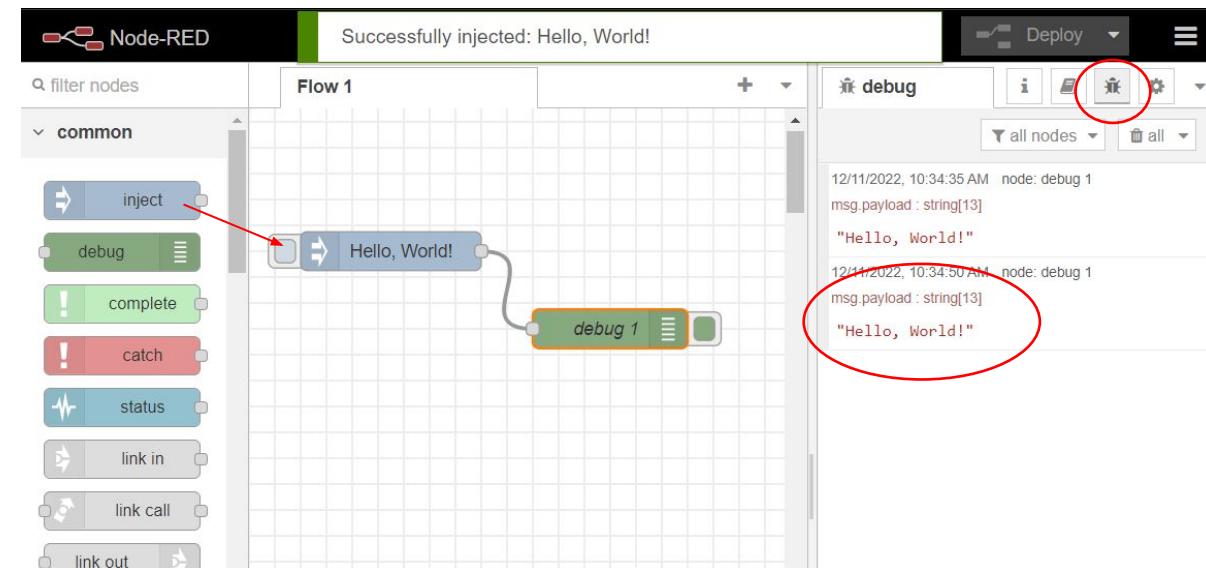
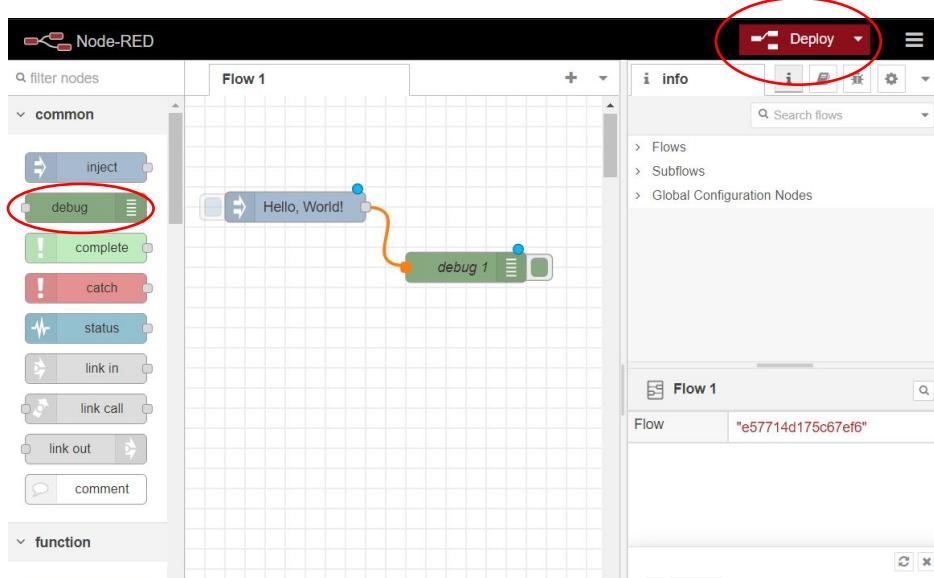
## 1. First Activity

- Drag and place the inject node inside canvas
- Double click the inject node
- In payload field, select string and enter “Hello, “World!



# IoT – Node-RED

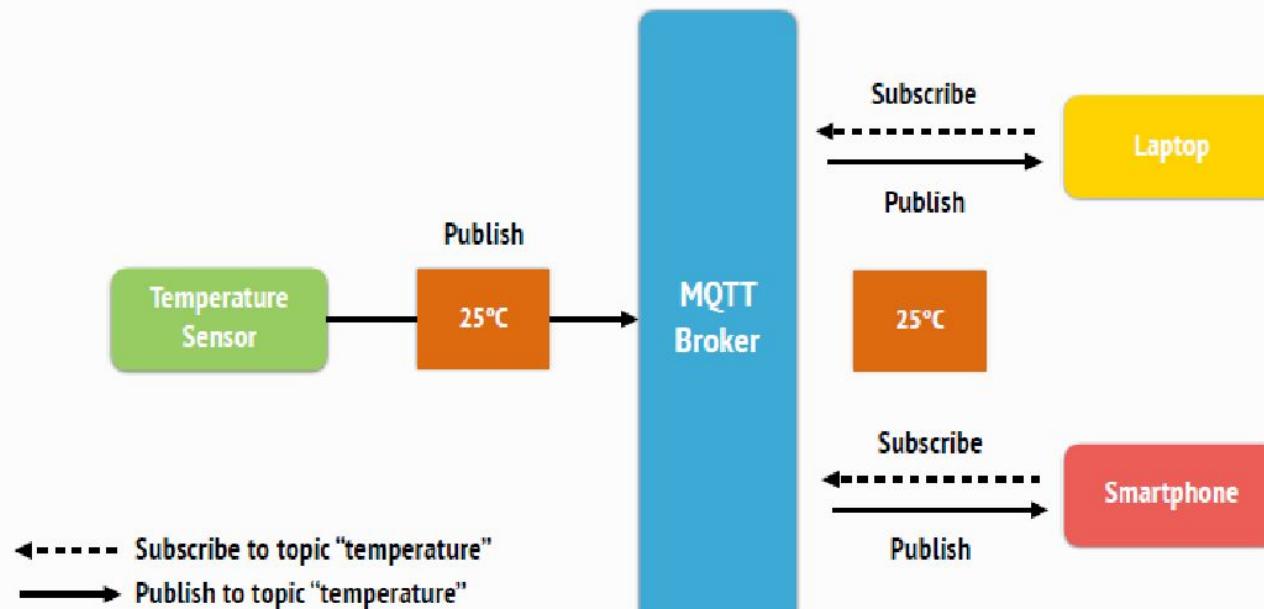
1. Drag and place the debug node inside canvas
2. Connect the nodes together
3. In payload field, select string and enter “Hello, “World!
4. Deploy and Run



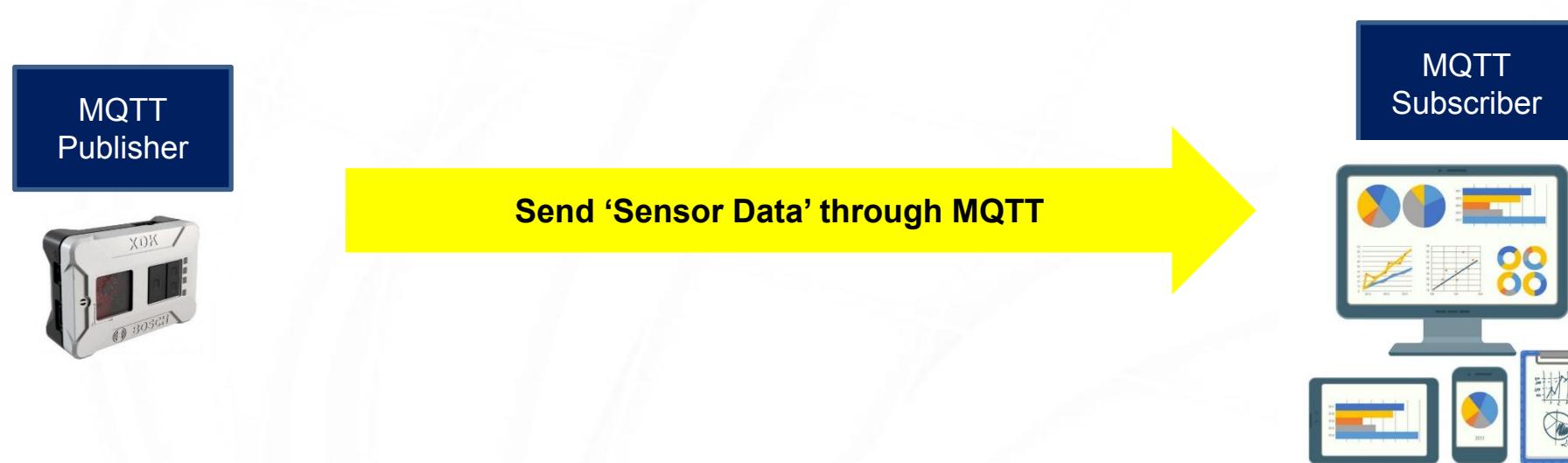
# Task #2

# Send Message (from your computer to my computer)

# MQTT



# Send Data through MQTT



# Exercise

Sensor  
at My  
House

Send 'Data'

Your  
Laptop



MQTT  
Publisher

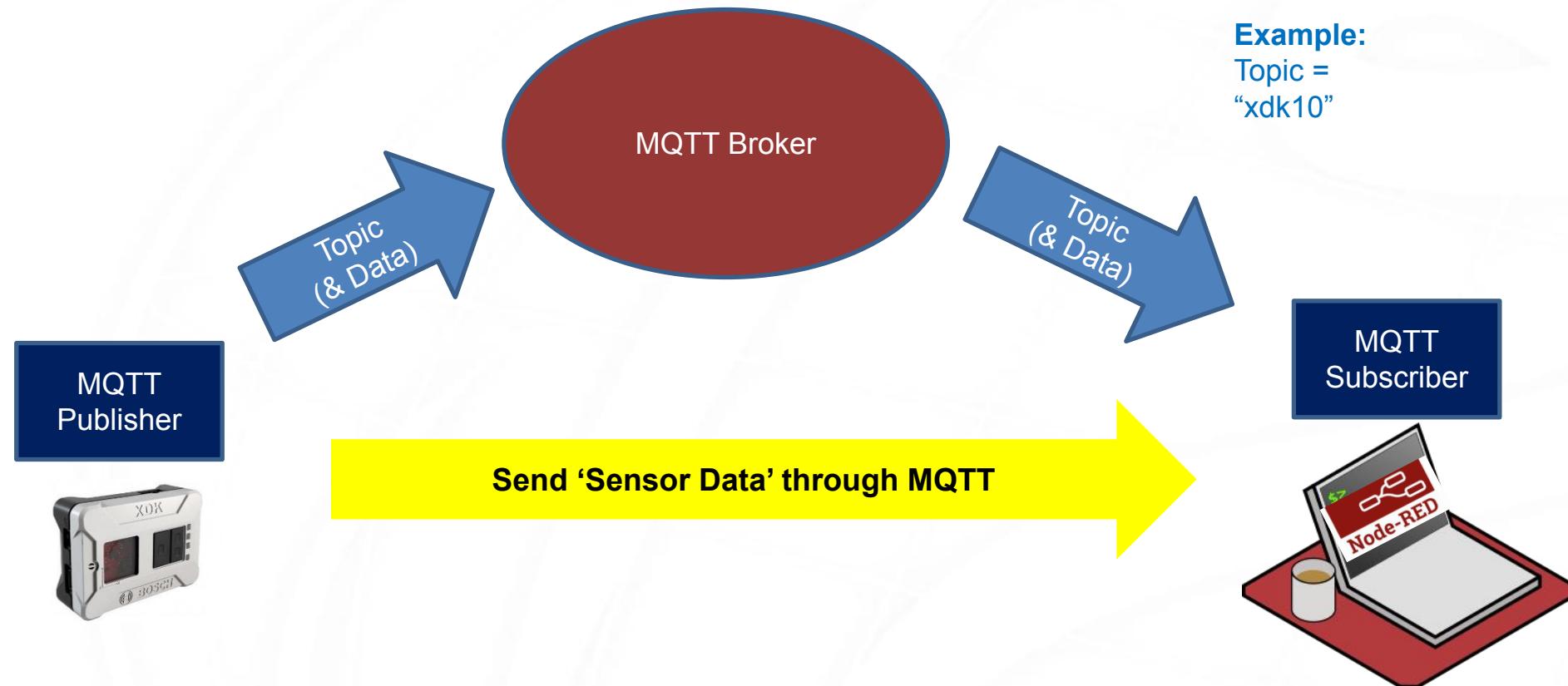
Send 'Data' through MQTT

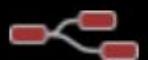


MQTT  
Subscriber



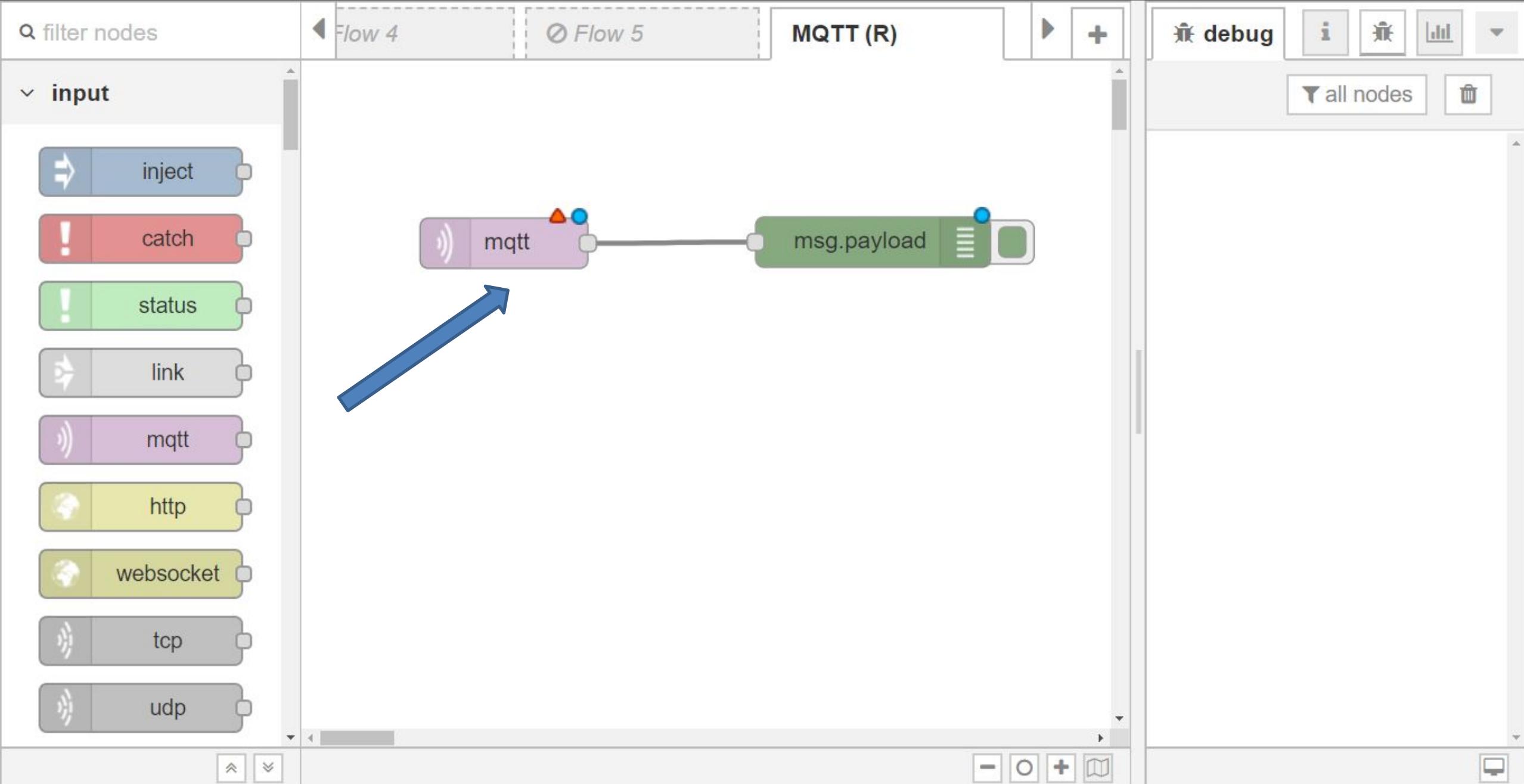
# Send & Receive Data using MQTT





Node-RED

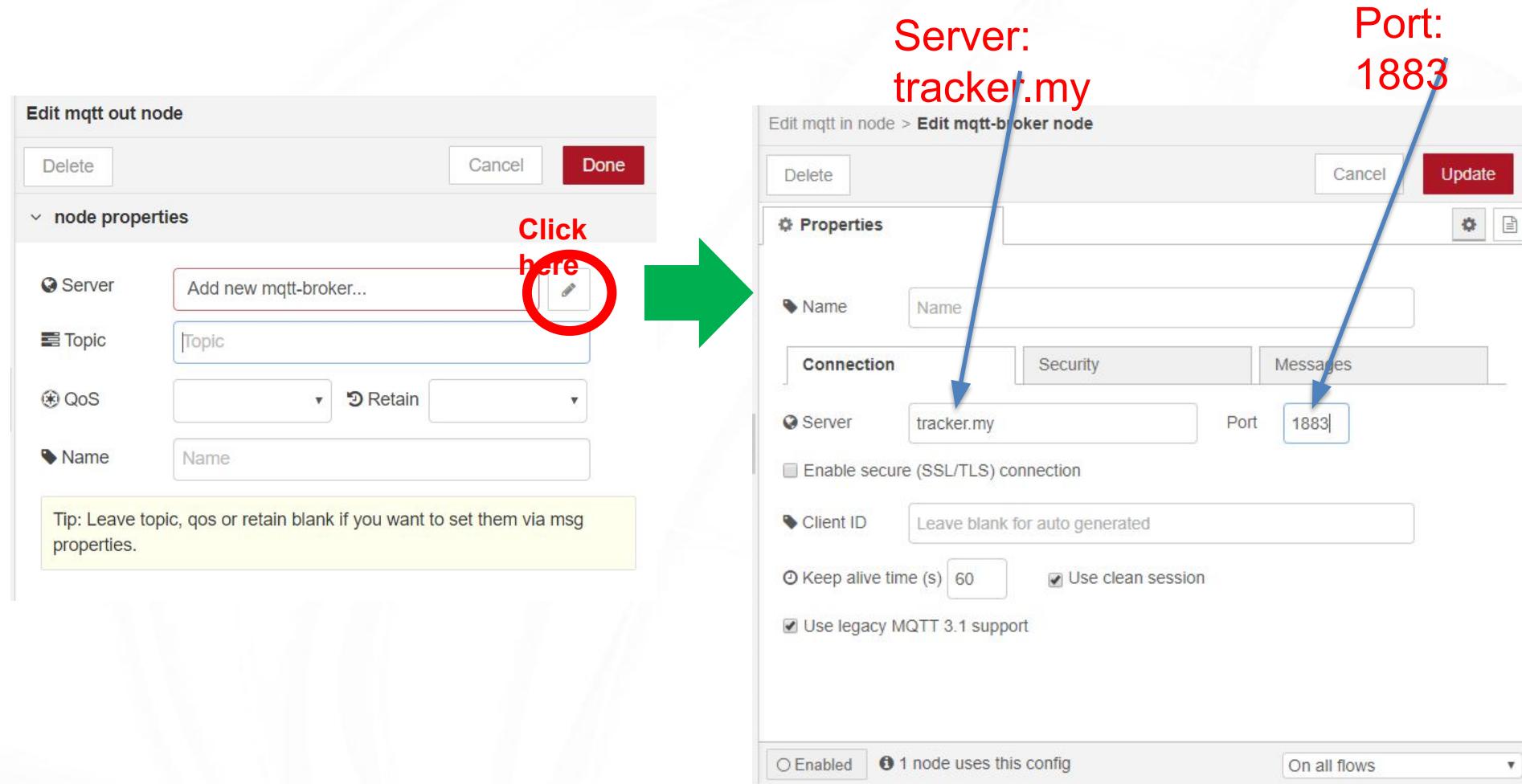
Deploy ▾



# Configure MQTT Broker

**Server: tracker.my**

**Port: 1883**



**Edit mqtt out node**

Delete Cancel Done

✓ node properties

Server: Add new mqtt-broker... 

Topic: Topic

QoS: Retain

Name: Name

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

**Edit mqtt in node > Edit mqtt-broker node**

Delete Cancel Update

Properties

Name: Name

Connection Security Messages

Server: tracker.my Port: 1883

Enable secure (SSL/TLS) connection

Client ID: Leave blank for auto generated

Keep alive time (s): 60 Use clean session

Use legacy MQTT 3.1 support

Enabled: 1 node uses this config On all flows



## Edit mqtt in node

[Delete](#)[Cancel](#)[Done](#)**Properties****Server**

tracker.my:1883

**Topic**

Temp\_1

**QoS**

2

**Output**

auto-detect (string or buffer)

**Name**

Name



# Let's try another mission...



**Crew : 6**

**Days occupied :**

**15 years, 330 days** & counting

(as on 27<sup>th</sup> September 2016)

**Orbital speed :**

7.66 kilometres per second  
(27,600 km/h; 17,100 mph)

**Orbital period:**

92.69 minutes

**Mass :**

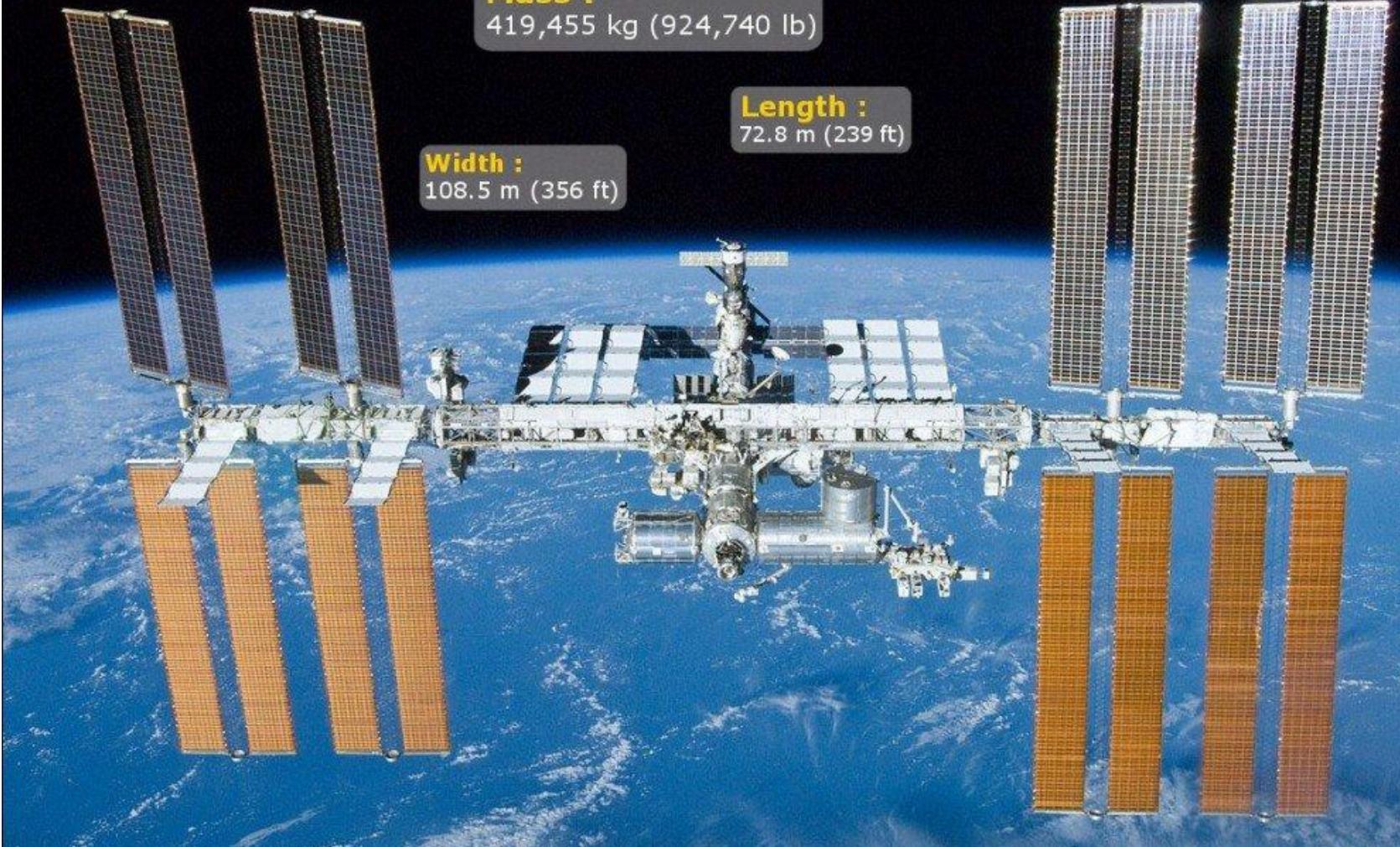
419,455 kg (924,740 lb)

**Width :**

108.5 m (356 ft)

**Length :**

72.8 m (239 ft)



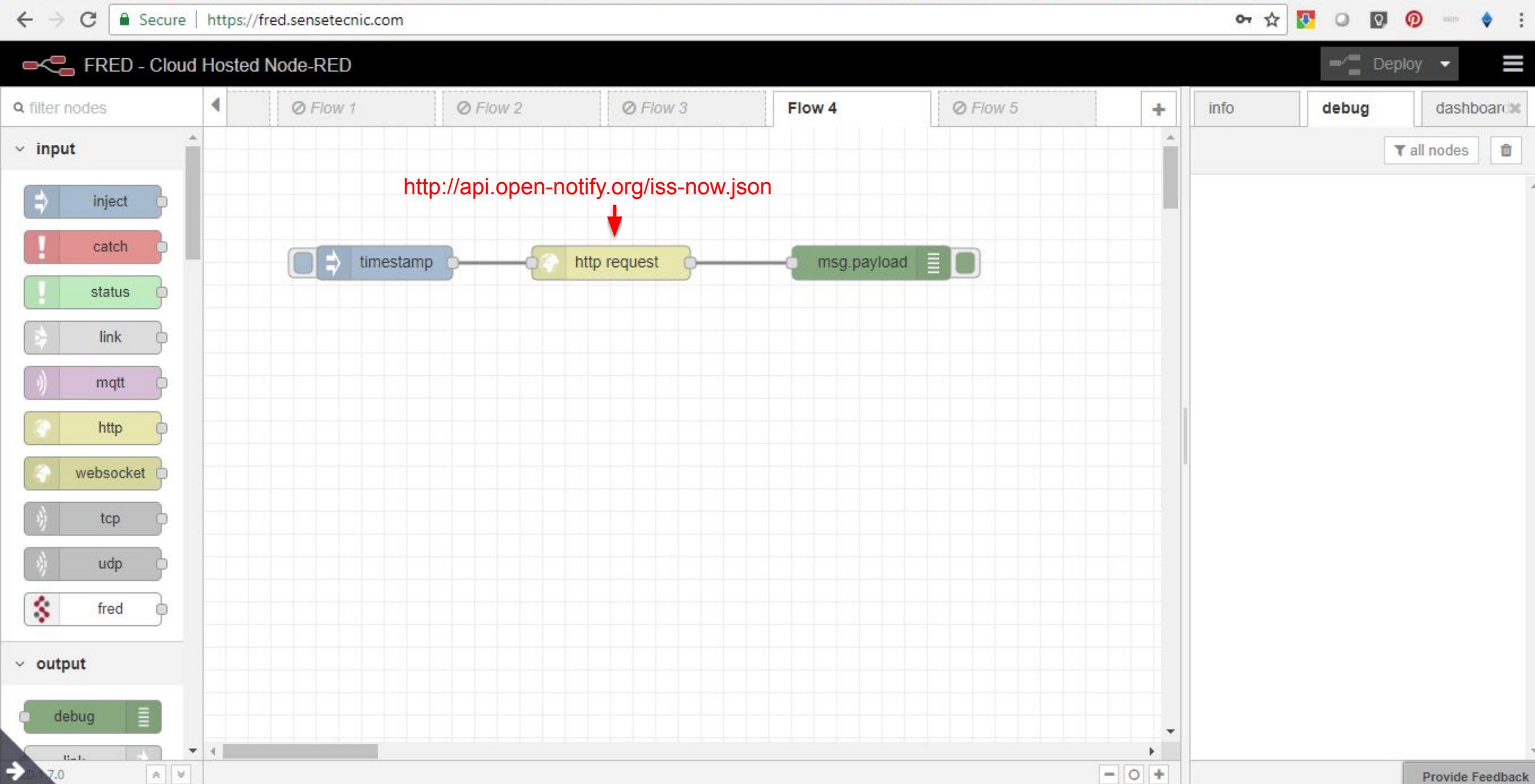
# Where is the International Space Station (ISS)?



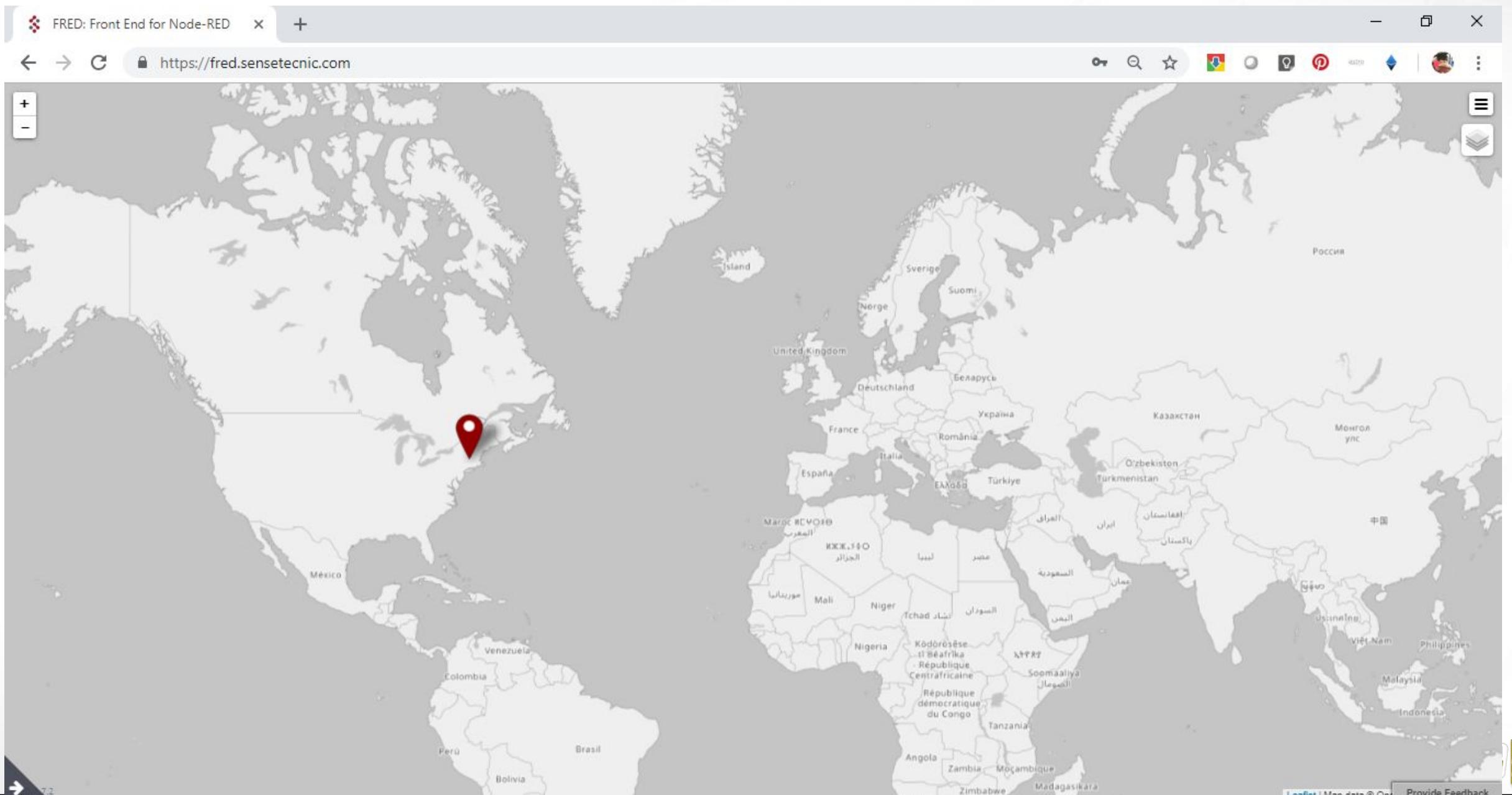
- Exercise: Tracking ISS using Node RED
  - Latitude and Longitude

Source Wikipedia: The **International Space Station (ISS)** is a [space station](#), or a habitable [artificial satellite](#), in [low Earth orbit](#). Its first component launched into orbit in 1998, the last pressurised module was fitted in 2011, and the station is expected to operate until 2028. Development and assembly of the station continues, with components scheduled for launch in 2018 and 2019. The ISS is the largest human-made body in low Earth orbit and can often be seen with the [naked eye](#) from Earth.<sup>[8][9]</sup> The ISS consists of pressurised modules, external trusses, [solar arrays](#), and other components. ISS components have been launched by Russian [Proton](#) and [Soyuz](#) rockets, and American [Space Shuttles](#).<sup>[10]</sup>



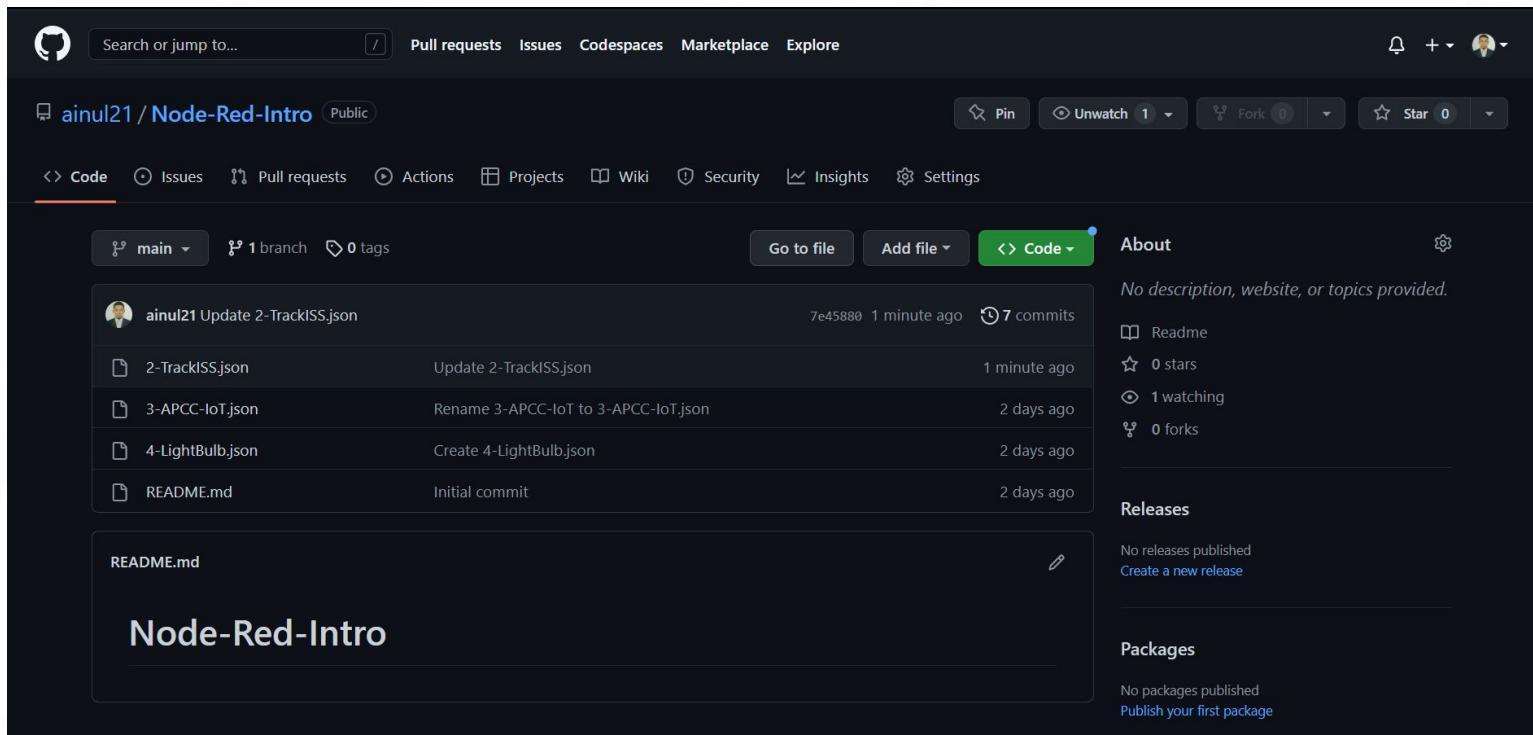


# Display on Map?



# IoT – Node-RED

1. Open browser
2. Type URL
  - [github.com/ainul21/Node-Red-Intro](https://github.com/ainul21/Node-Red-Intro)



The screenshot shows a GitHub repository page for 'ainul21 / Node-Red-Intro'. The repository is public and contains 7 commits. The commits are:

- ainul21 Update 2-TrackISS.json (7e45880, 1 minute ago)
- 2-TrackISS.json (Update 2-TrackISS.json, 1 minute ago)
- 3-APCC-IoT.json (Rename 3-APCC-IoT to 3-APCC-IoT.json, 2 days ago)
- 4-LightBulb.json (Create 4-LightBulb.json, 2 days ago)
- README.md (Initial commit, 2 days ago)

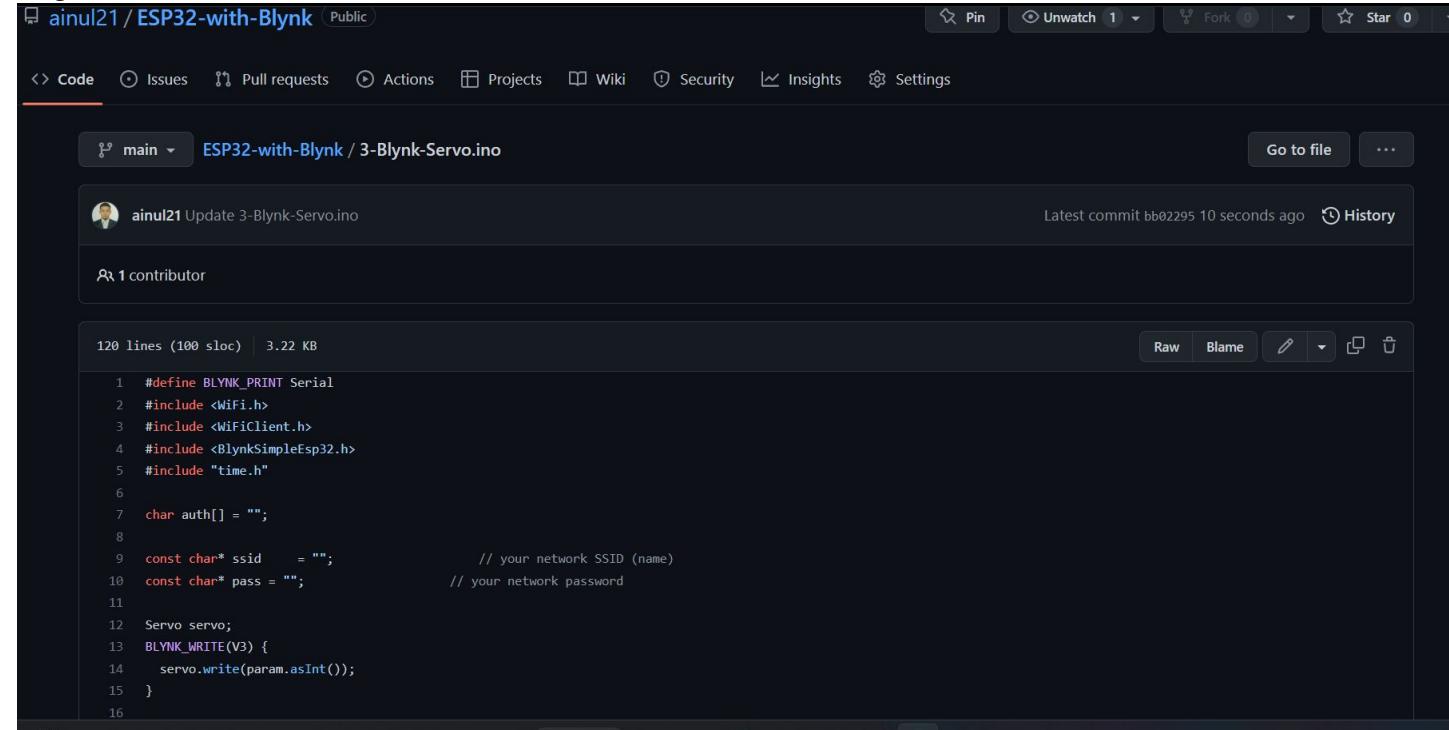
The repository has 0 stars, 1 watching, and 0 forks. There are no releases or packages published.



# IoT – Node-RED

## Task #2

- Go to GitHub
- Open file name: 2-TrackISS.json
- Copy the code

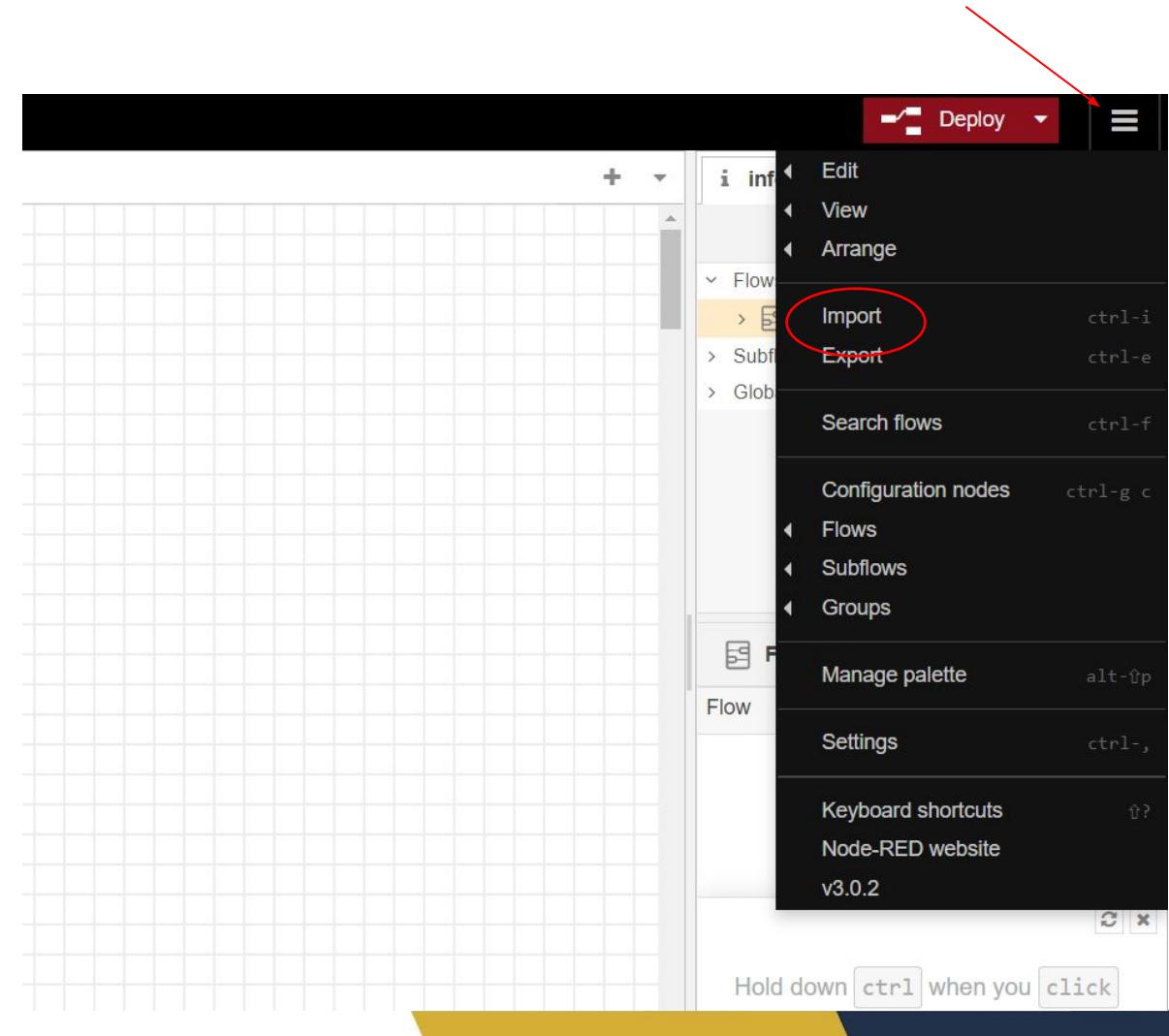


The screenshot shows a GitHub repository page for 'ainul21 / ESP32-with-Blynk'. The repository has 1 contributor and the latest commit was 10 seconds ago. The file '3-Blynk-Servo.ino' is displayed, containing the following code:

```
1 #define BLYNK_PRINT Serial
2 #include <WiFi.h>
3 #include <WiFiClient.h>
4 #include <BlynkSimpleEsp32.h>
5 #include "time.h"
6
7 char auth[] = "";
8
9 const char* ssid      = "";           // your network SSID (name)
10 const char* pass     = "";           // your network password
11
12 Servo servo;
13 BLYNK_WRITE(V3) {
14   servo.write(param.asInt());
15 }
16
```

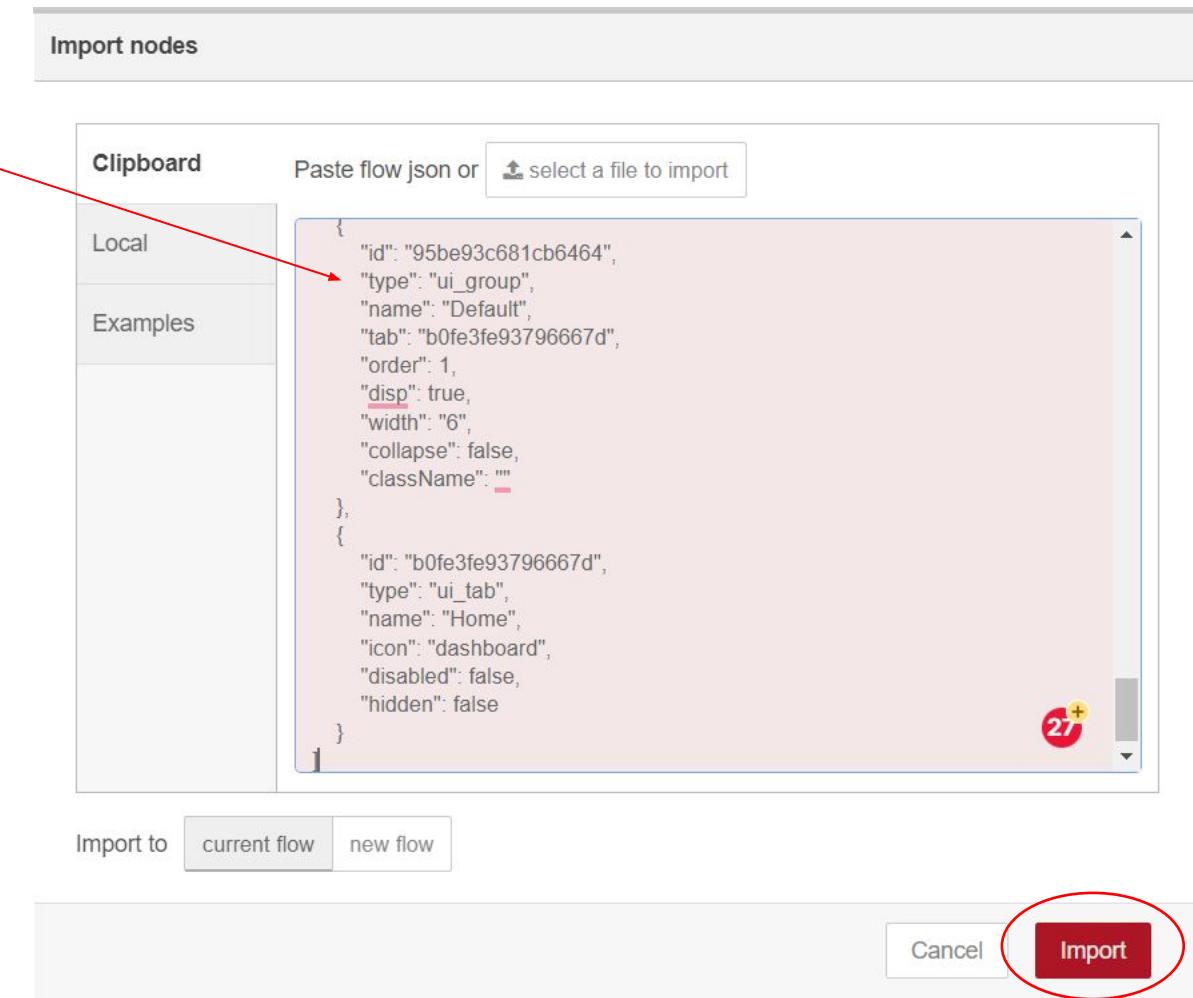
# IoT – Node-RED

1. Click triple line button
2. Click Import



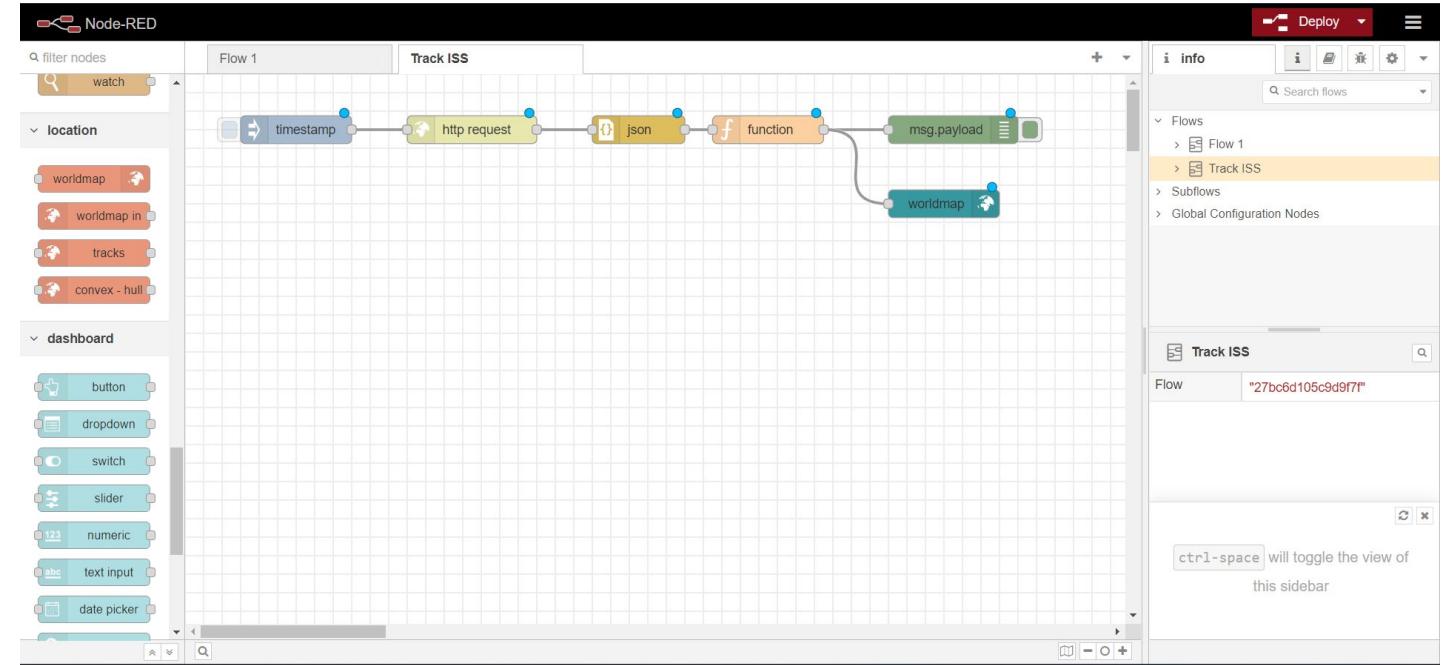
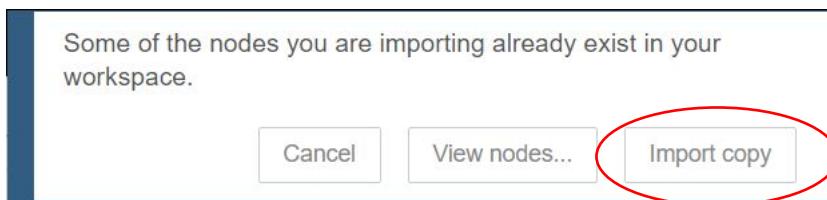
# IoT – Node-RED

1. Paste the code
2. Click Import



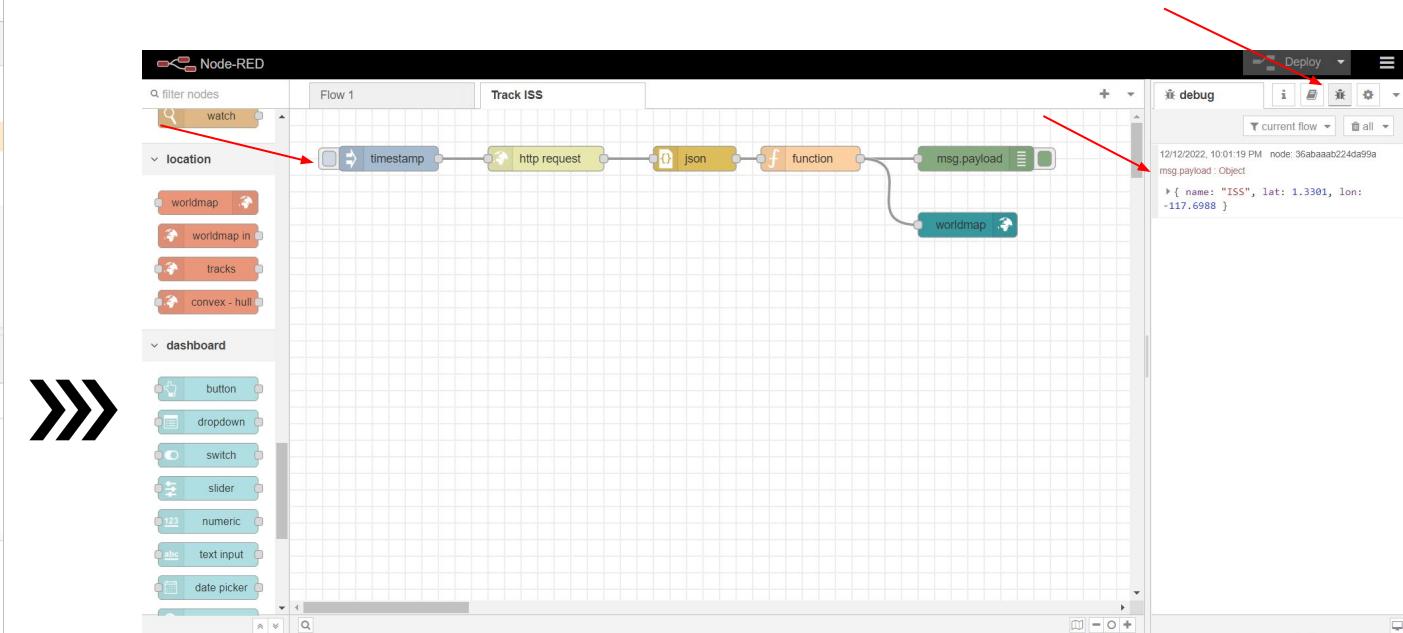
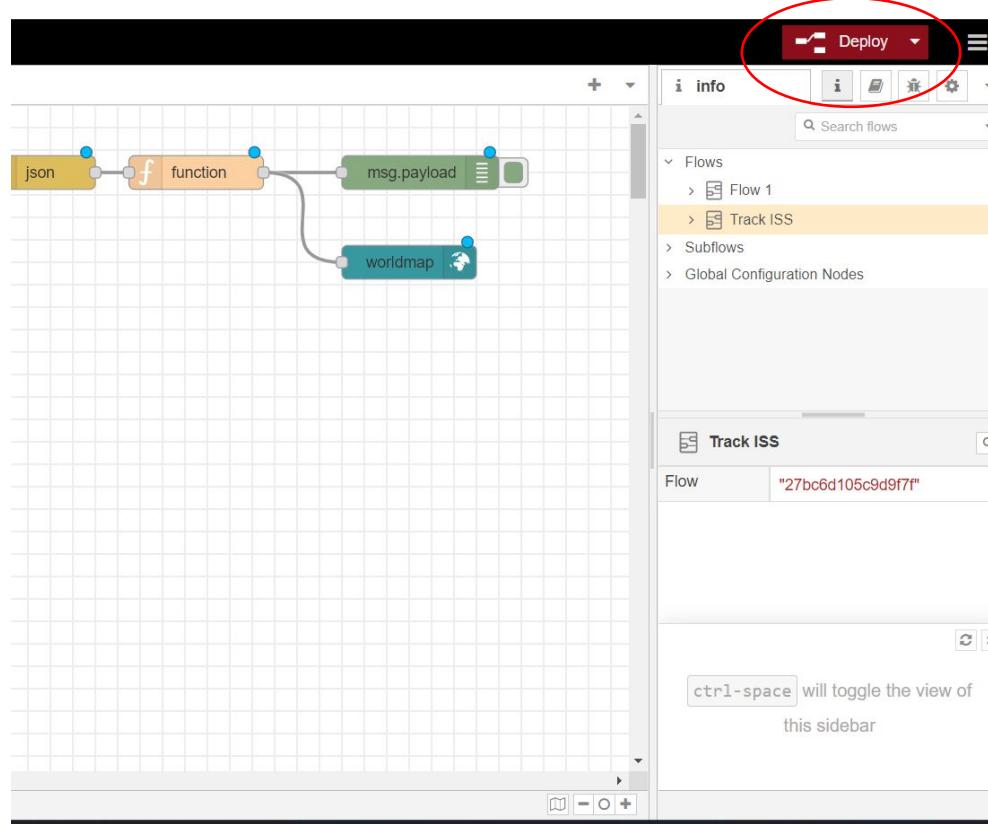
# IoT – Node-RED

1. Click Import copy
2. Import complete



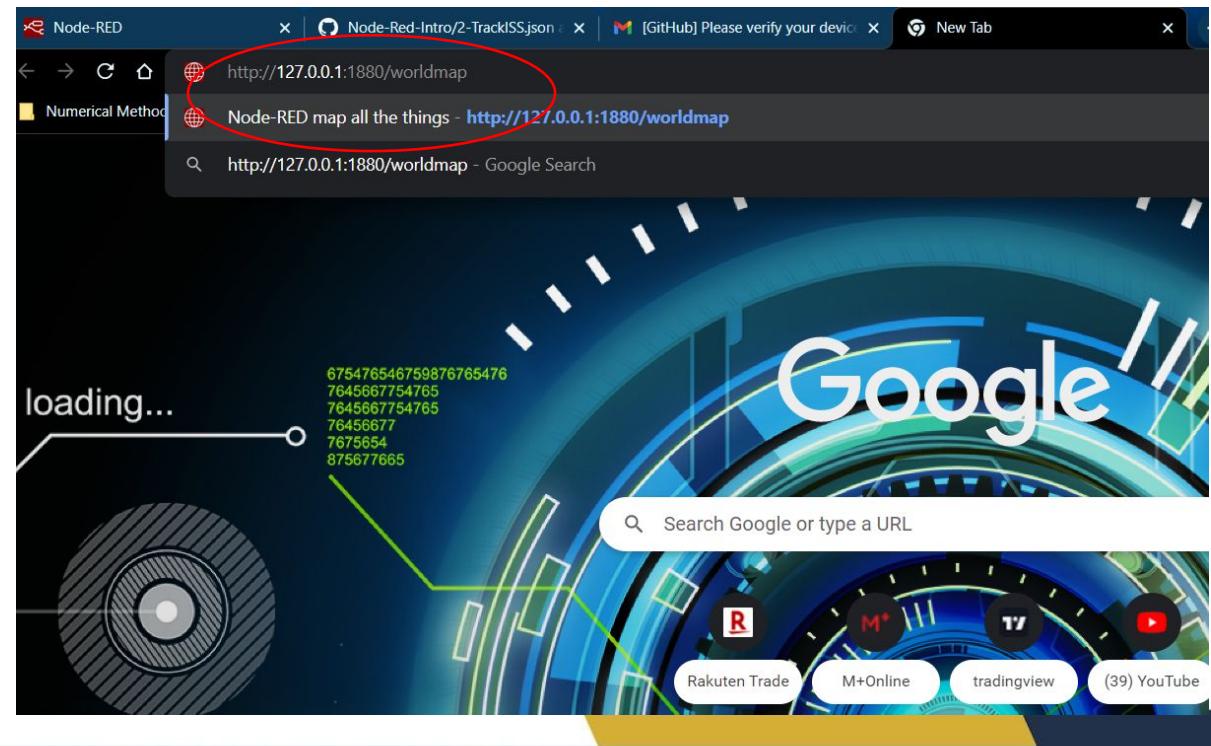
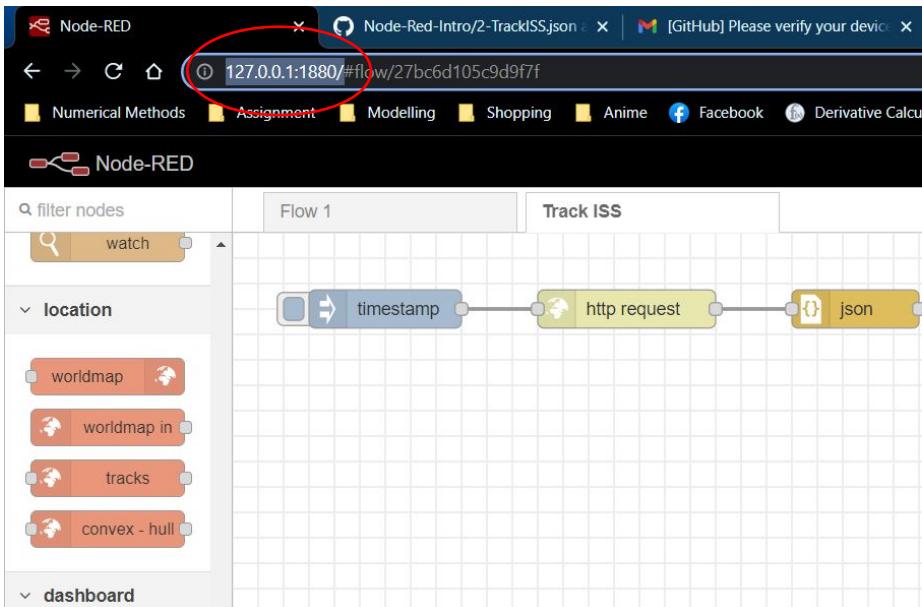
# IoT – Node-RED

1. Click Deploy
2. Run the flow by clicking the inject trigger



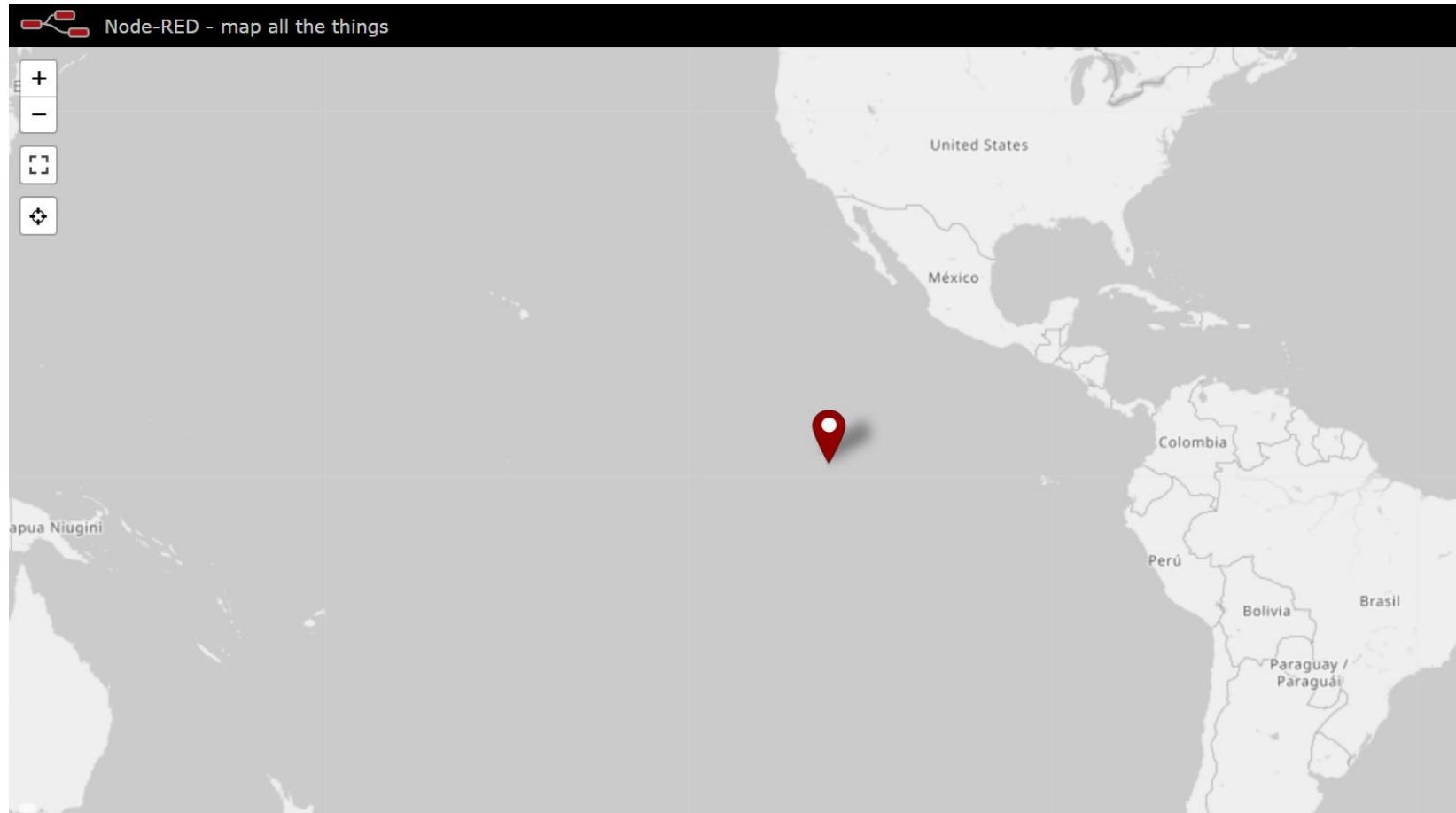
# IoT – Node-RED

1. To open the map, copy the URL as below
2. Open new tab and paste the URL
3. Add worldmap at the end of /
4. Press Enter



# IoT – Node-RED

Zoom and drag to achieve the best result of the tracker



# Final Mission...

# Can you switch on the light bulb from your computer?



# IoT – ESP32 with Blynk

#Practice Time!

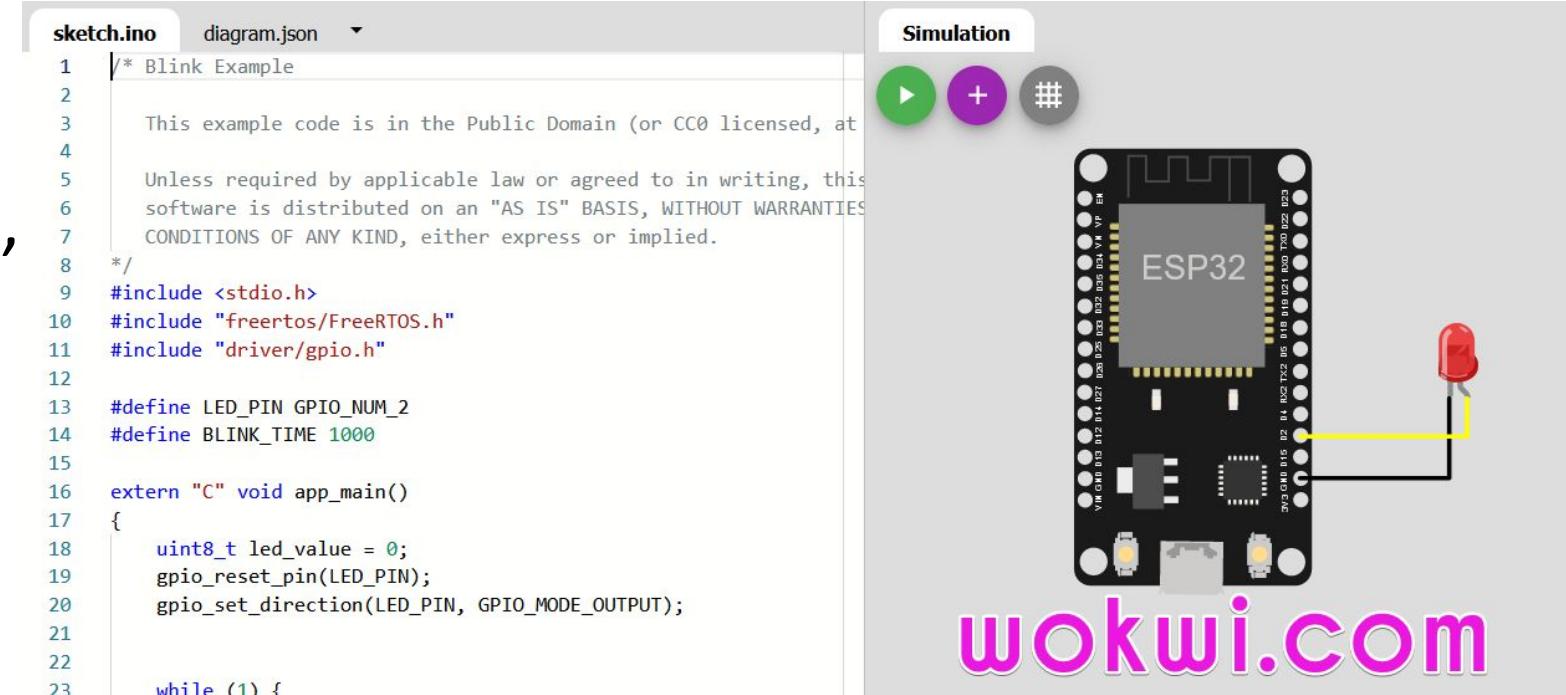
# IoT – ESP32 with Blynk

- **Blynk** is an IoT platform for mobile application and web-based application
- Used to control Arduino, Raspberry Pi and NodeMCU via Internet
- Very user friendly and easy to setup.
- In-app purchase



# IoT – ESP32 with Blynk

- Wokwi is an online electronics simulator
- Used to simulate Arduino, ESP32 and many more
- Has limited amount of passive and active electronic component
- Free



The screenshot shows the Wokwi simulation environment. On the left, the code editor displays the following sketch:

```

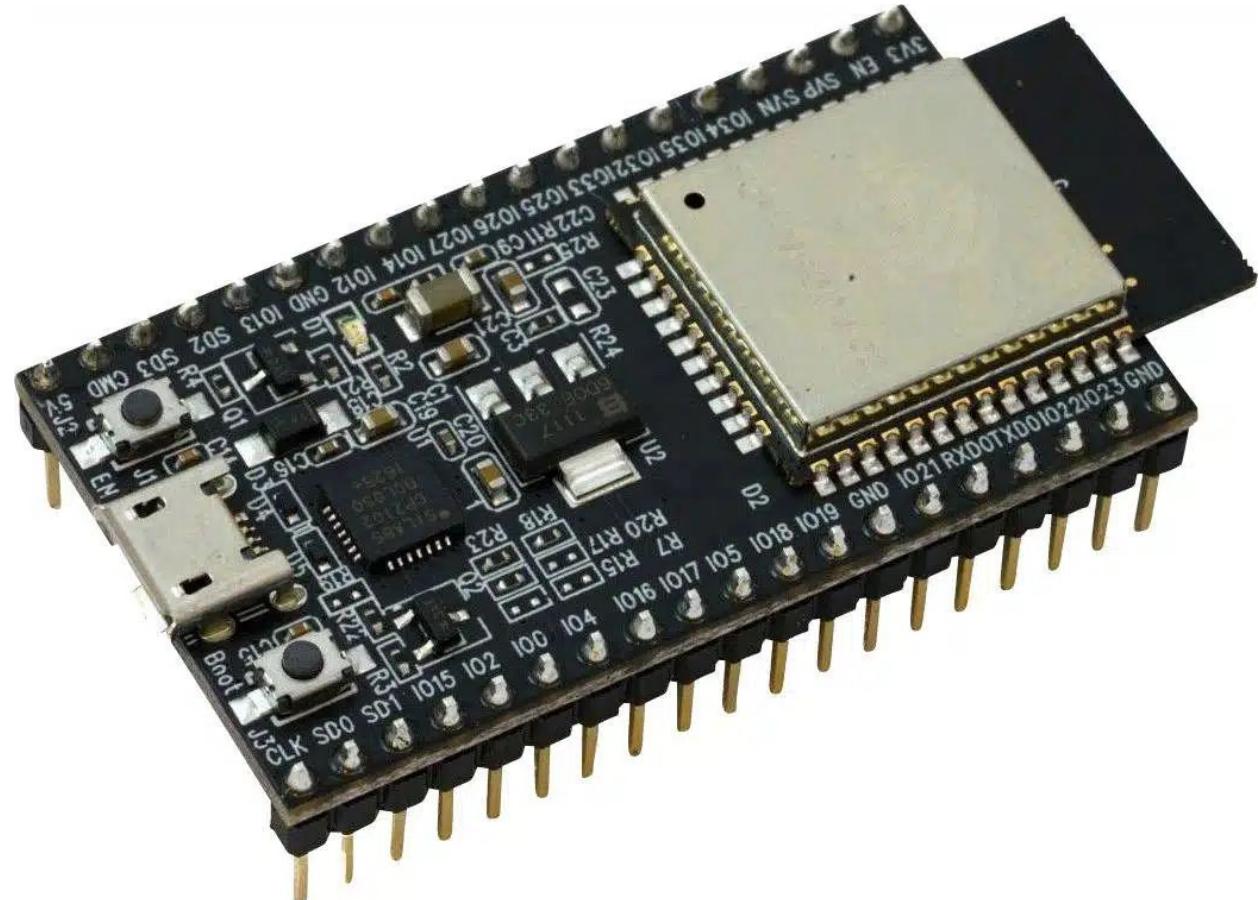
sketch.ino    diagram.json
1  /* Blink Example
2
3  This example code is in the Public Domain (or CC0 licensed, at
4
5  Unless required by applicable law or agreed to in writing, this
6  software is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES
7  CONDITIONS OF ANY KIND, either express or implied.
8 */
9  #include <stdio.h>
10 #include "freertos/FreeRTOS.h"
11 #include "driver/gpio.h"
12
13 #define LED_PIN GPIO_NUM_2
14 #define BLINK_TIME 1000
15
16 extern "C" void app_main()
17 {
18     uint8_t led_value = 0;
19     gpio_reset_pin(LED_PIN);
20     gpio_set_direction(LED_PIN, GPIO_MODE_OUTPUT);
21
22     while (1) {
23

```

The right side of the interface shows a digital breadboard with an ESP32 module. A red LED is connected to pin D2, which is also connected to ground through a resistor. The breadboard also features two push buttons labeled 'A' and 'B'. The Wokwi logo is visible at the bottom right.

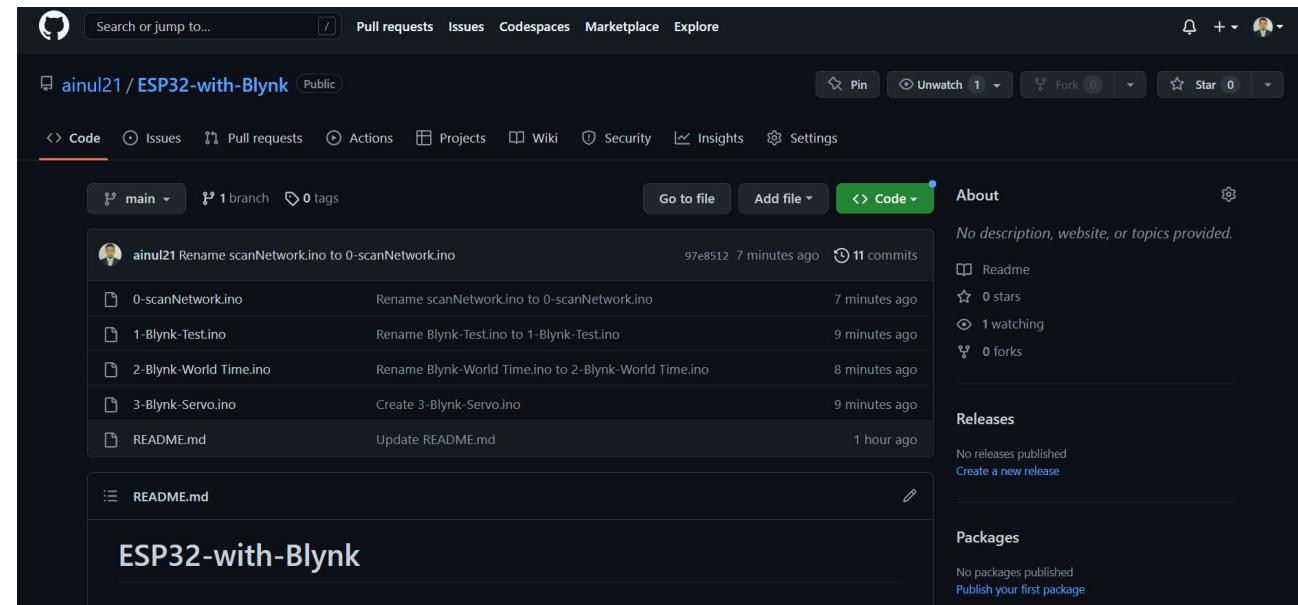
# IoT – ESP32 with Blynk

- **ESP32** microcontroller integrated with Wi-Fi and Bluetooth module.
- Very practical for IoT project
  - Low-cost
  - Low-power
  - Many GPIO pin



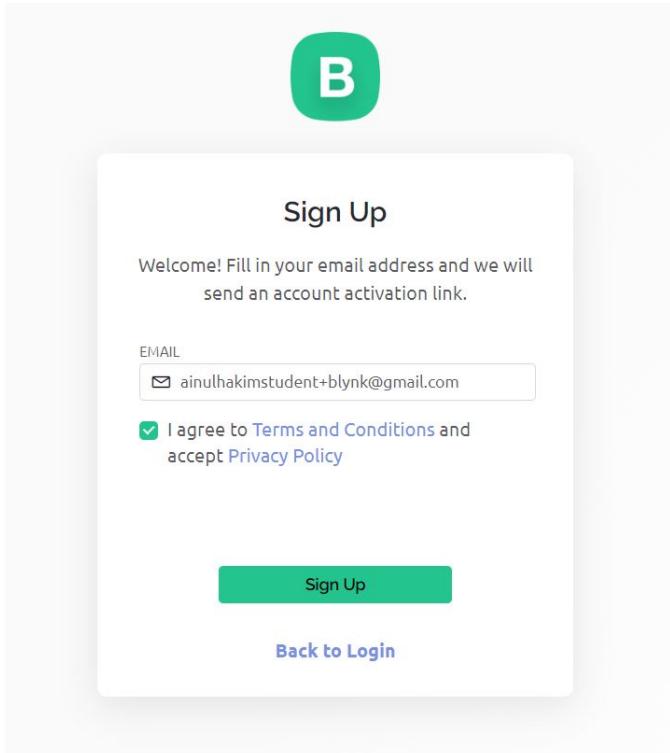
# IoT – ESP32 with Blynk

1. Open browser
2. Type URL
  - [github.com/ainul21/ESP32-with-Blynk](https://github.com/ainul21/ESP32-with-Blynk)
3. Open the Wokwi and Blynk web page,  
the link located at README.md



# IoT – ESP32 with Blynk

1. Open Blynk web page
2. Create new account/login



**B**

**Sign Up**

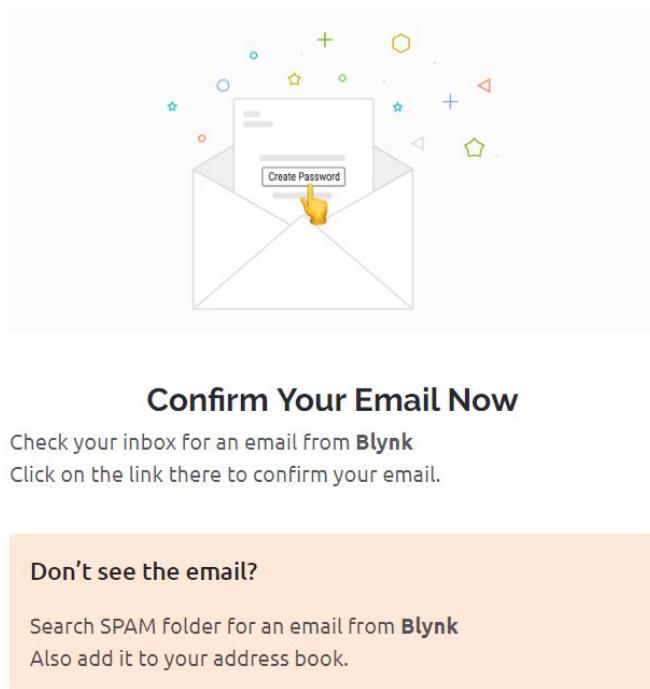
Welcome! Fill in your email address and we will send an account activation link.

EMAIL

I agree to [Terms and Conditions](#) and accept [Privacy Policy](#)

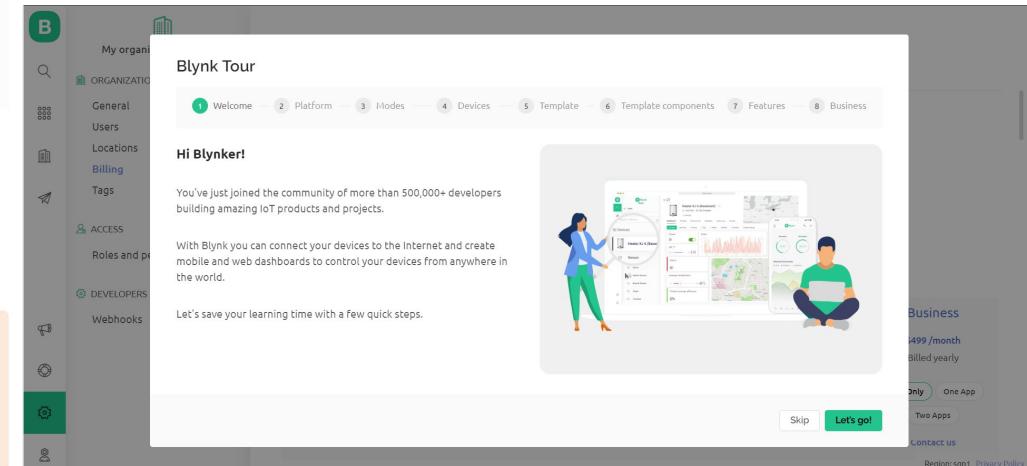
**Sign Up**

[Back to Login](#)



Register new account

Click ESP32



**Blynk Tour**

Hi Blynker!

You've just joined the community of more than 500,000+ developers building amazing IoT products and projects.

With Blynk you can connect your devices to the Internet and create mobile and web dashboards to control your devices from anywhere in the world.

Let's save your learning time with a few quick steps.

**Blynk Tour**

1 Welcome    2 Platform    3 Modes    4 Devices    5 Template    6 Template components    7 Features    8 Business

**ORGANIZATION**

- General
- Users
- Locations
- Billing
- Tags

**ACCESS**

- Roles and permissions

**DEVELOPERS**

- Webhooks

**Business**

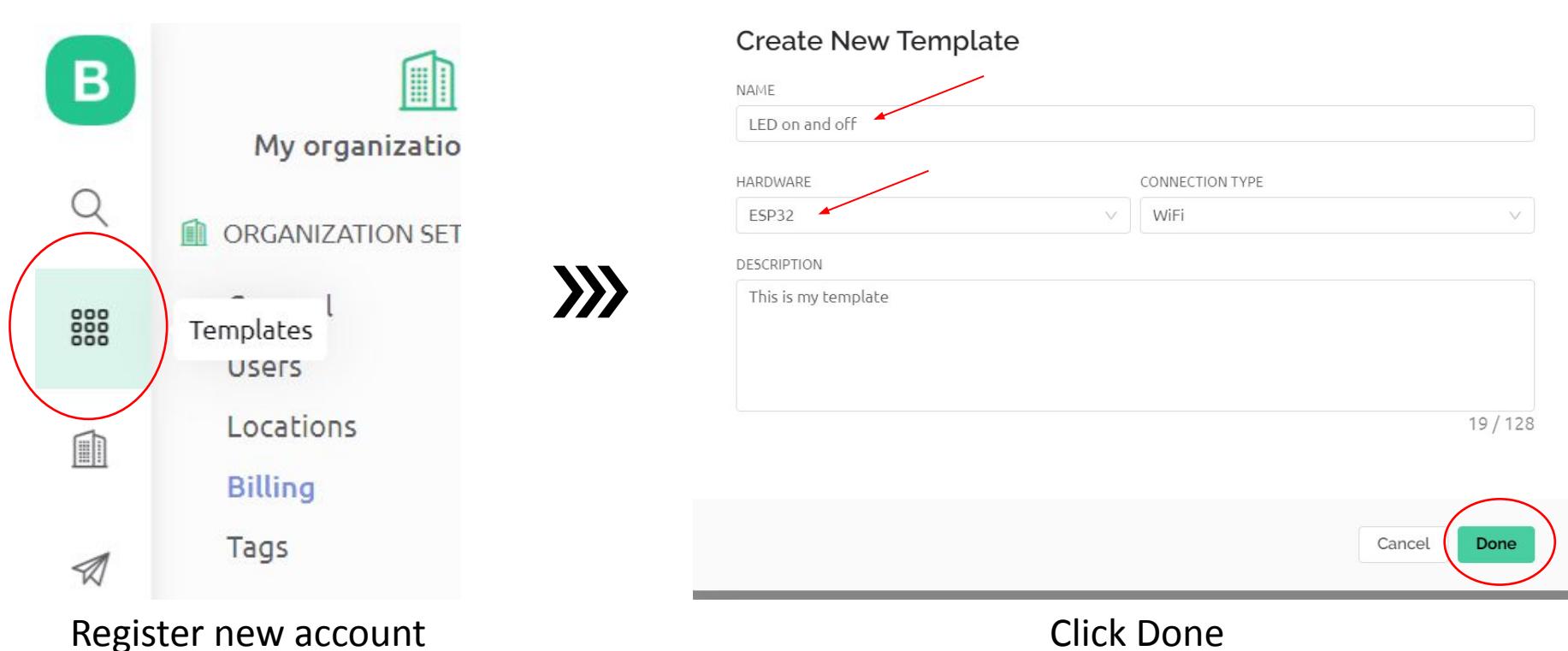
499/month  
Billed yearly

Only One App    Two Apps    Contact us

Skip    Let's go!

# IoT – ESP32 with Blynk

1. Open Templates
2. Create New Template



My organization

ORGANIZATION SET

Templates

Users

Locations

Billing

Tags

Register new account

»»»

Create New Template

NAME  
LED on and off

HARDWARE  
ESP32

CONNECTION TYPE  
WiFi

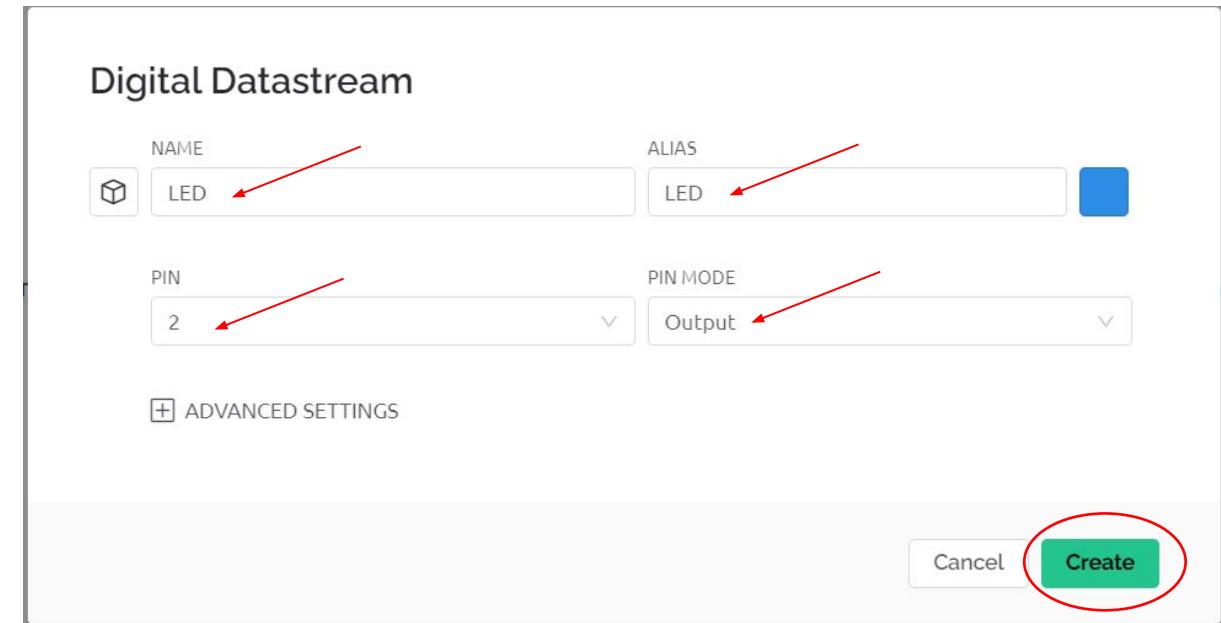
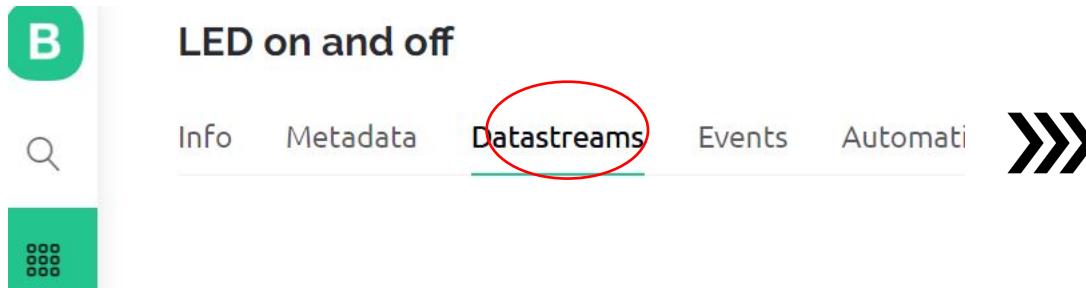
DESCRIPTION  
This is my template  
19 / 128

Cancel Done

Click Done

# IoT – ESP32 with Blynk

1. Open Datastream
2. Create New Datastream
3. Click Save



Digital Datastream

NAME	LED	ALIAS	LED
PIN	2	PIN MODE	Output

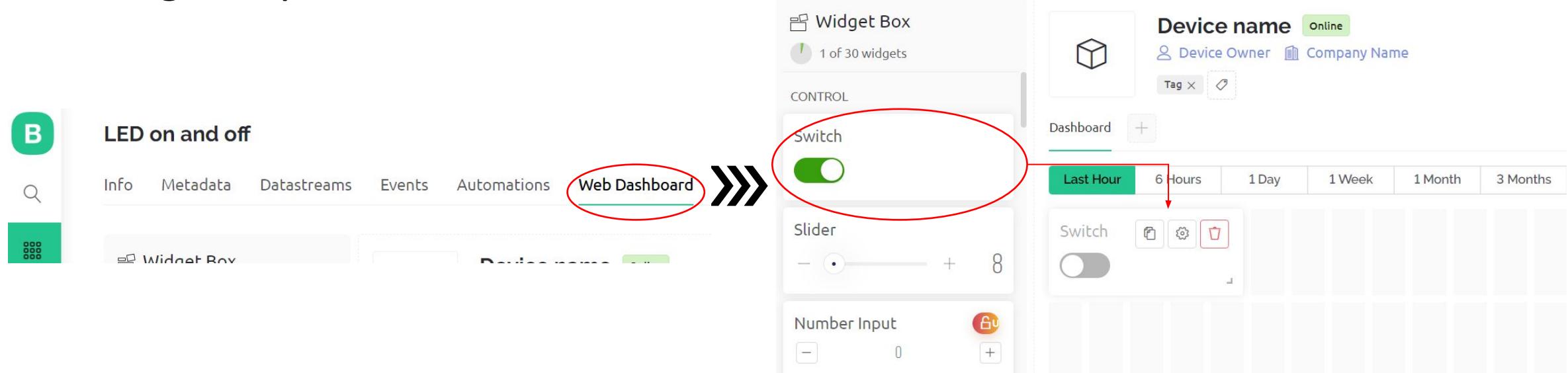
+ ADVANCED SETTINGS

Cancel **Create**

This image shows a configuration dialog for a digital datastream. It has fields for 'NAME' (set to 'LED'), 'ALIAS' (set to 'LED'), 'PIN' (set to '2'), and 'PIN MODE' (set to 'Output'). A red arrow points to each of these four fields. At the bottom right is a 'Create' button, which is also circled in red.

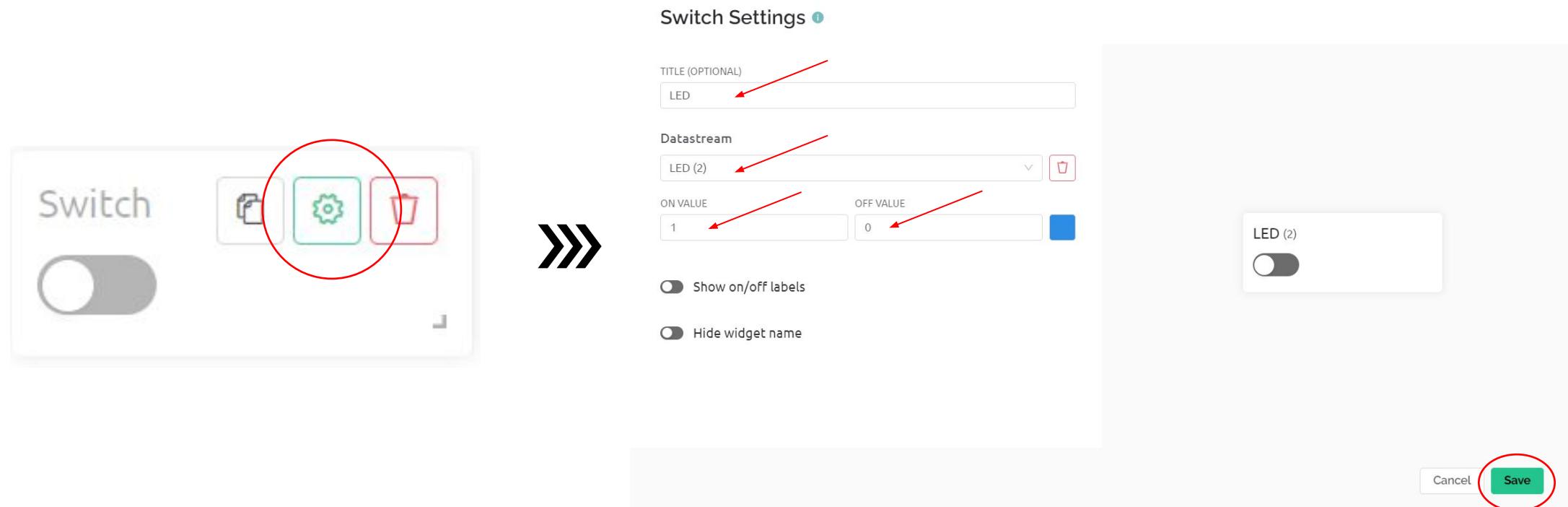
# IoT – ESP32 with Blynk

1. Open Web Dashboard
2. At Widget Box find Switch
3. Drag and place the Switch into canvas



# IoT – ESP32 with Blynk

1. Click setting icon at Switch layout
2. Save the layout

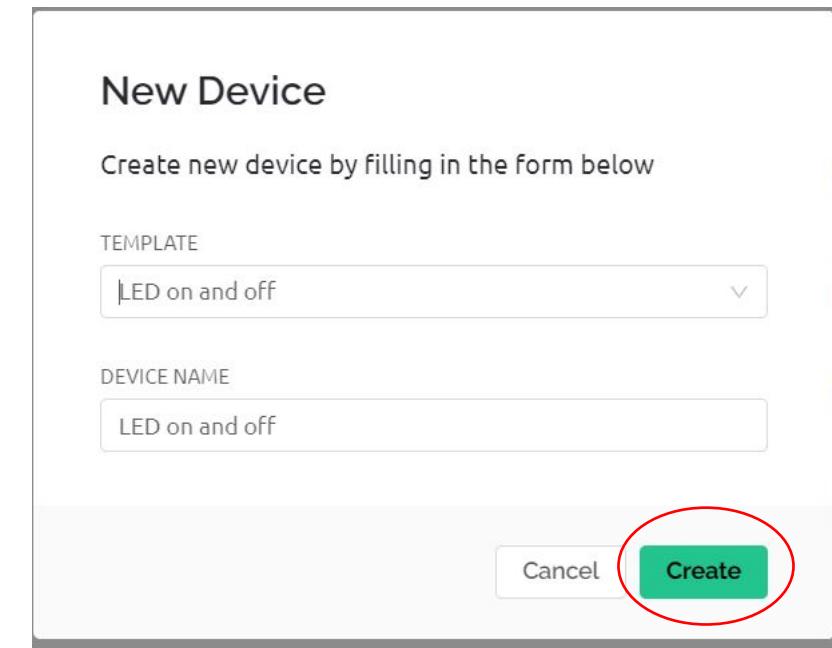
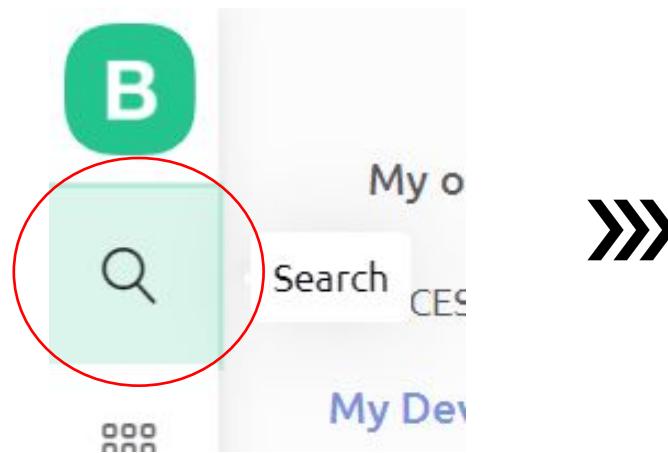


The diagram illustrates the process of saving a Blynk switch layout. It is divided into three main sections:

- Left Panel:** Shows a "Switch" component. A small gear icon representing settings is highlighted with a red circle.
- Middle Panel:** Titled "Switch Settings". It contains several configuration fields:
  - Title (Optional):** LED (highlighted with a red arrow).
  - Datastream:** LED (2) (highlighted with a red arrow).
  - ON VALUE:** 1 (highlighted with a red arrow).
  - OFF VALUE:** 0 (highlighted with a red arrow).
  - Show on/off labels:** A toggle switch.
  - Hide widget name:** A toggle switch.
- Right Panel:** Shows the final saved state of the switch component, labeled "LED (2)" with a toggle switch icon. At the bottom right, there are "Cancel" and "Save" buttons, with the "Save" button highlighted with a red circle.

# IoT – ESP32 with Blynk

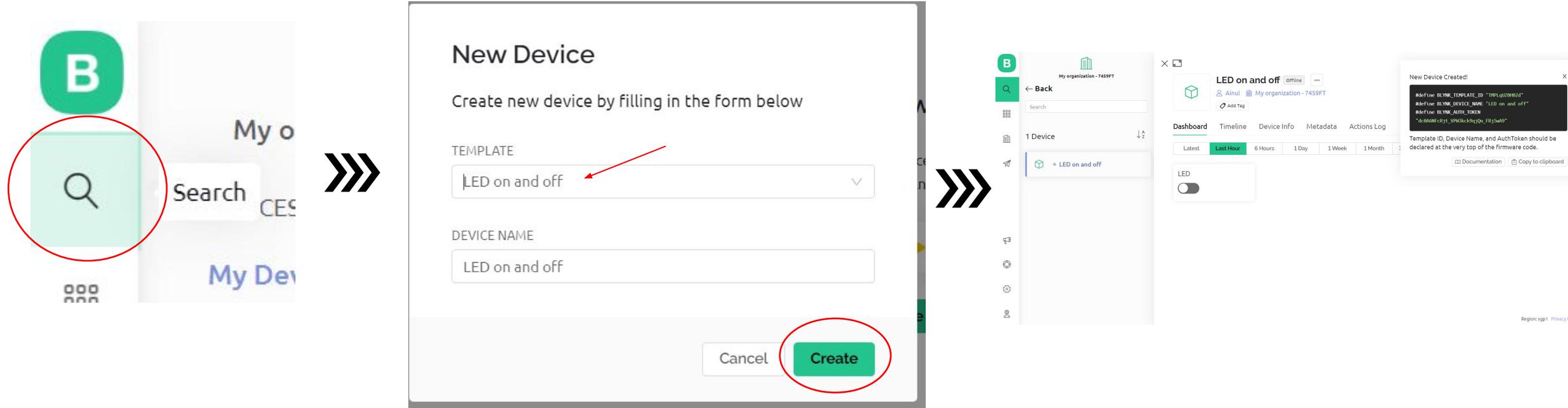
1. Open Search
2. Add New Device, click From Template
3. Create New Device



A screenshot of a 'New Device' creation dialog box. The title 'New Device' is at the top. Below it, a sub-instruction says 'Create new device by filling in the form below'. There are two input fields: 'TEMPLATE' containing 'LED on and off' and 'DEVICE NAME' also containing 'LED on and off'. At the bottom right of the dialog is a green 'Create' button, which is circled in red. To its left is a 'Cancel' button.

# IoT – ESP32 with Blynk

1. Open Search
2. Add New Device, click From Template
3. Create New Device



**New Device**

Create new device by filling in the form below

TEMPLATE

LED on and off

DEVICE NAME

LED on and off

Create

Cancel

My o  
Search CES

My Dev

LED on and off

LED on and off

LED

New Device Created!

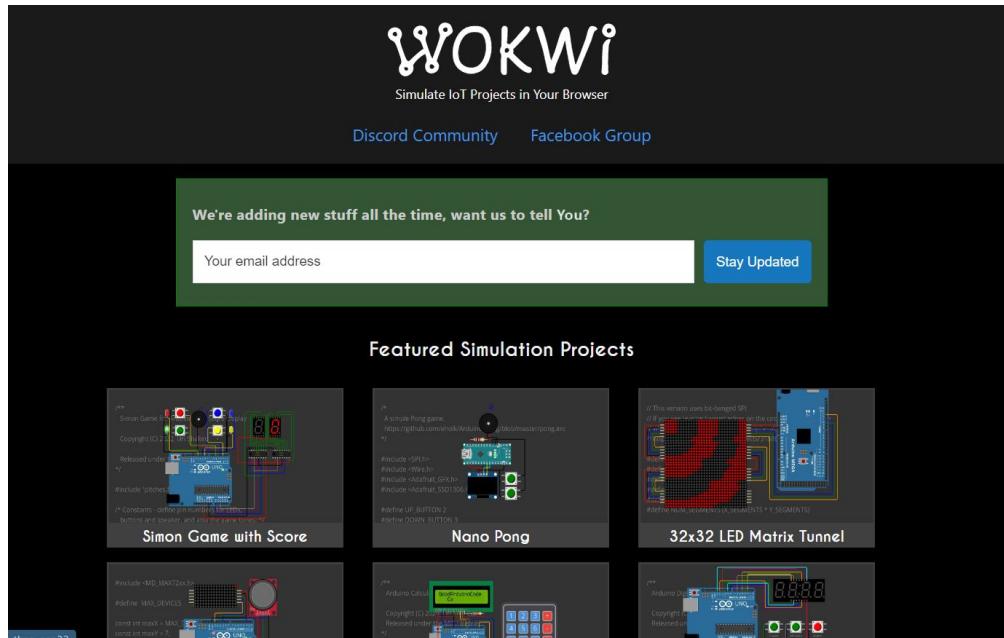
```
#define BLYNK_TEMPLATE_ID "TPPLqjZBH02d"
#define BLYNK_DEVICE_NAME "LED on and off"
#define BLYNK_AUTH_TOKEN
"dcBAQH-Ejt_VNkKckhsk04_FRjSw0"
```

Template ID, Device Name, and AuthToken should be declared at the very top of the firmware code.

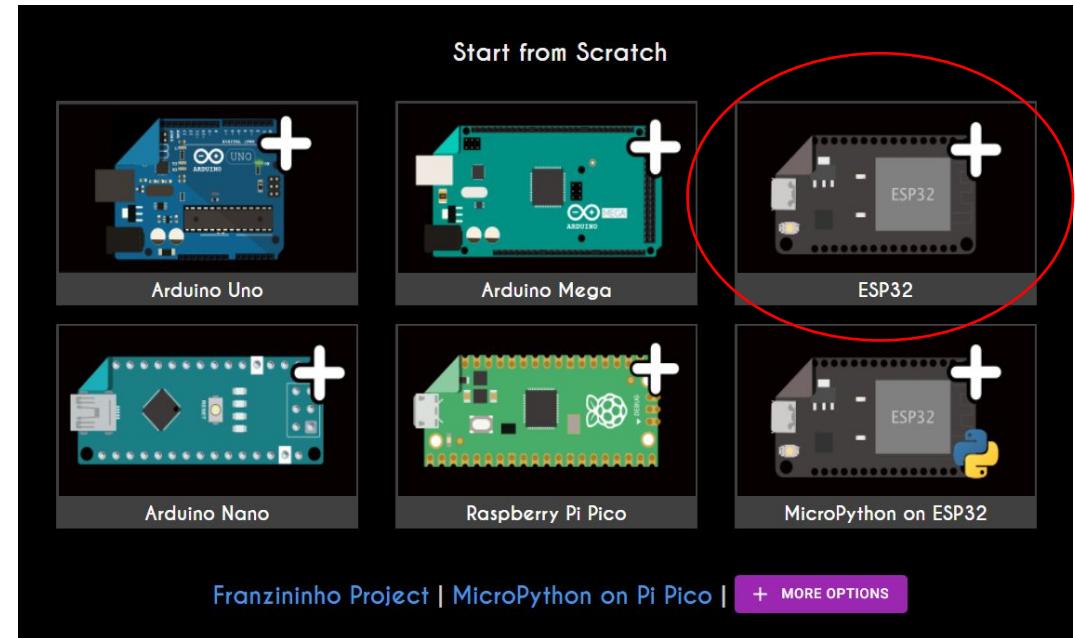
Documentation | Copy to clipboard

# IoT – ESP32 with Blynk

## 1. Open Wokwi web page



Scroll Down

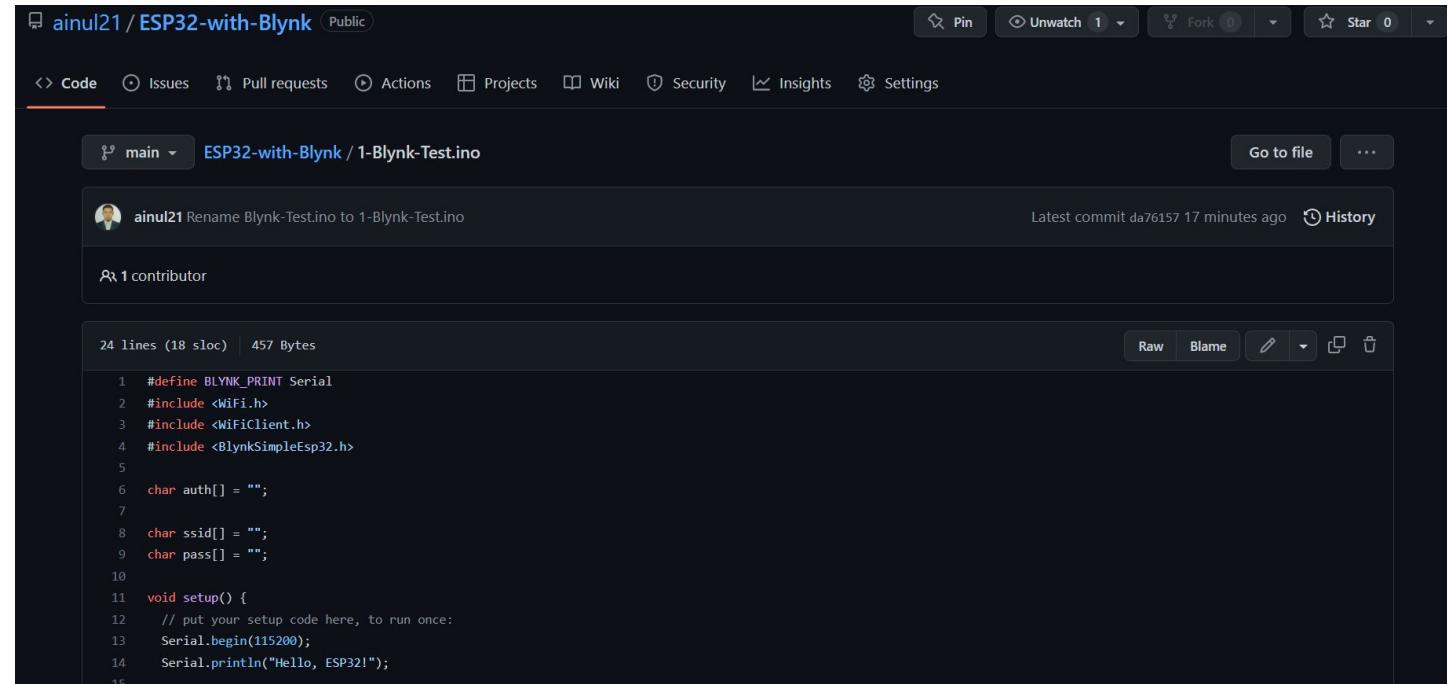


Click ESP32

# IoT – ESP32 with Blynk

## 1. First Activity

- Go to GitHub
- Open file name: 1-Blynk-Test.ino
- Copy the code
- Paste the code at Wikwo
- Add Blynk Library

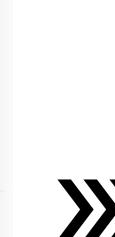
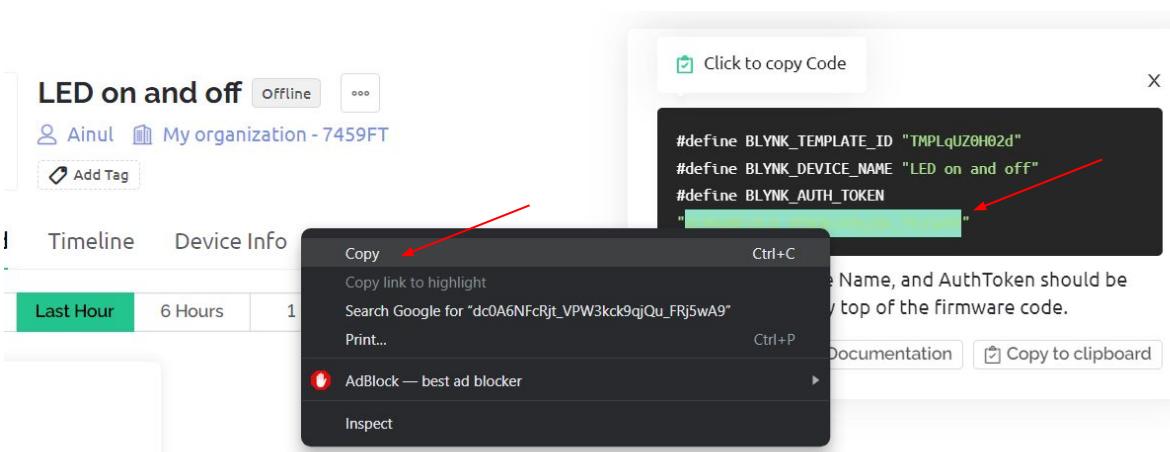
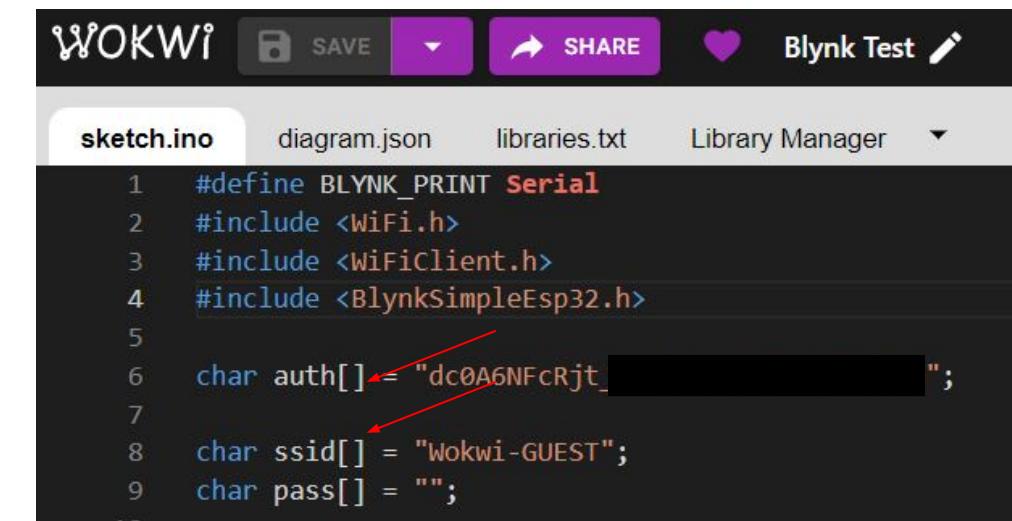


The screenshot shows a GitHub repository named "ESP32-with-Blynk" with a single file, "1-Blynk-Test.ino". The file contains the following code:

```
24 lines (18 sloc) | 457 Bytes
1 #define BLYNK_PRINT Serial
2 #include <WiFi.h>
3 #include <WiFiClient.h>
4 #include <BlynkSimpleEsp32.h>
5
6 char auth[] = "";
7
8 char ssid[] = "";
9 char pass[] = "";
10
11 void setup() {
12     // put your setup code here, to run once:
13     Serial.begin(115200);
14     Serial.println("Hello, ESP32!");
15 }
```

# IoT – ESP32 with Blynk

1. Copy the AuthKey from Blynk
2. Paste at the code
3. Add the SSID

A screenshot of the Wokwi editor. At the top, there's a toolbar with 'SAVE', 'SHARE', and other options. The main area shows a file named 'sketch.ino' with the following code:

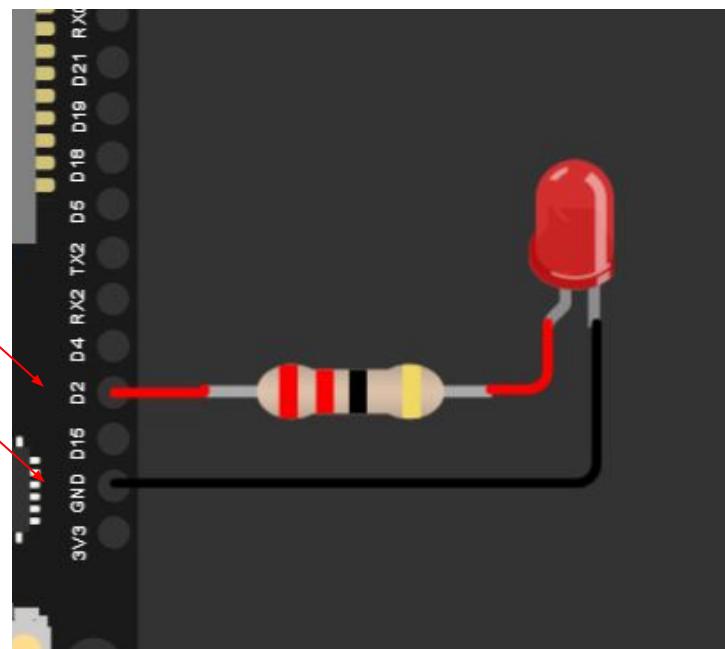
```
#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

char auth[] = "dc0A6NFcRjt_VPW3kck9qjQu_FRj5wA9";
char ssid[] = "Wokwi-GUEST";
char pass[] = "";
```

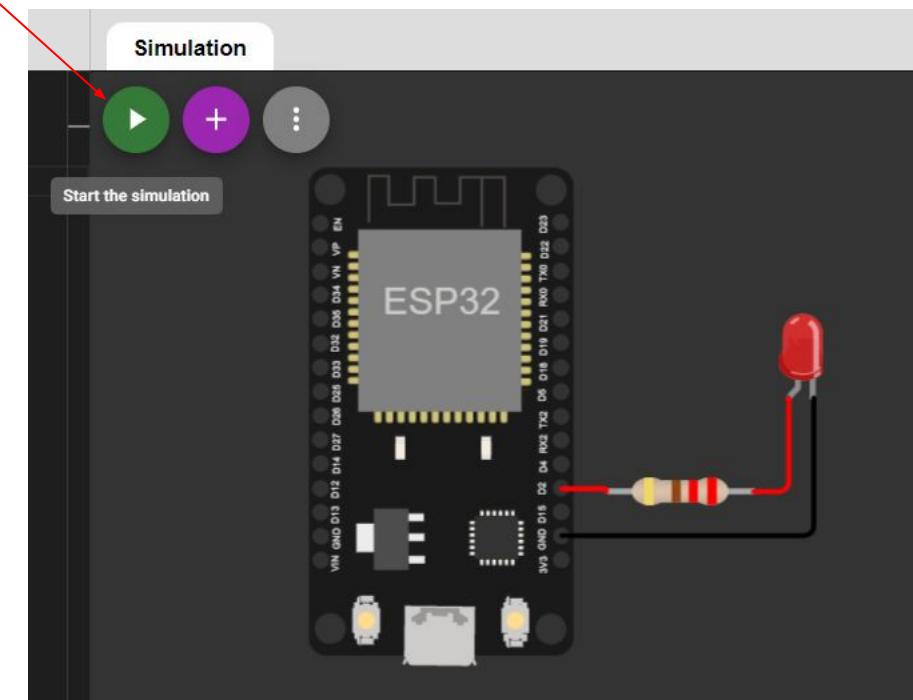
A red arrow points to the 'auth' variable, highlighting the copied value from the Blynk app.

# Iot – ESP32 with Blynk

1. Add Resistor and LED on the canvas
2. Connect correctly the component
3. Start the simulation



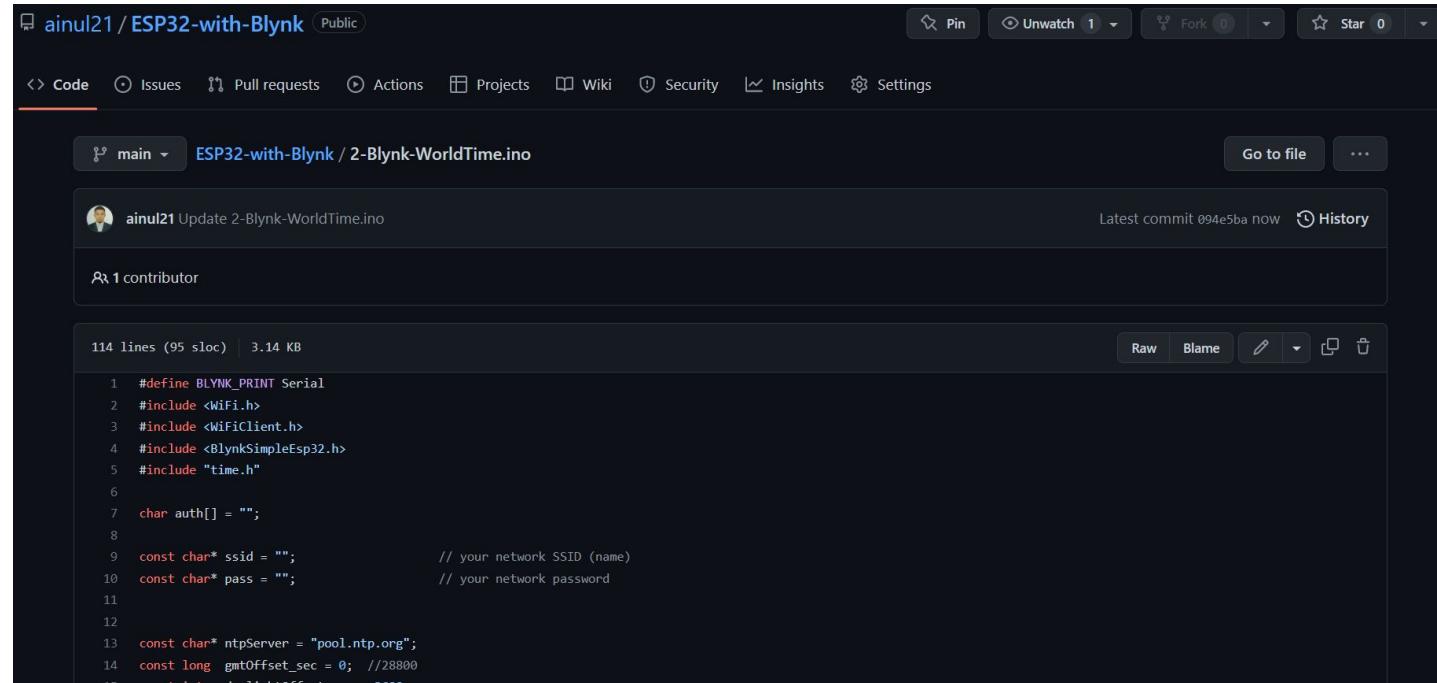
»»



# IoT – ESP32 with Blynk

## 1. Second Activity

- Go to GitHub
- Open file name: 2-Blynk-WorldTime.ino
- Copy the code
- Paste the code at Wikwo
- Add Blynk Library



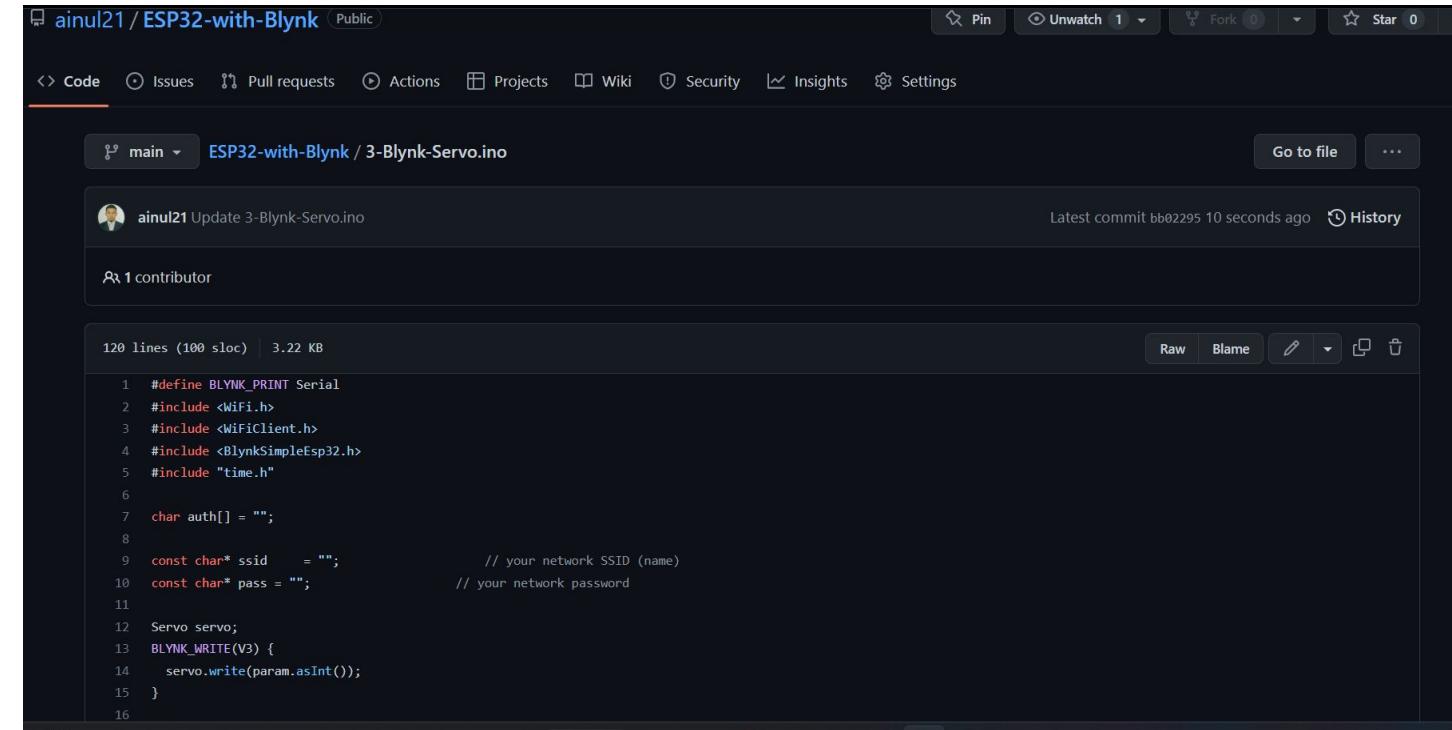
The screenshot shows a GitHub repository page for 'ainul21 / ESP32-with-Blynk'. The repository is public and contains one file, '2-Blynk-WorldTime.ino'. The file has 114 lines (95 sloc) and is 3.14 KB in size. The code includes definitions for BLYNK\_PRINT, WiFi.h, WiFiClient.h, BlynkSimpleEsp32.h, and time.h. It also defines network credentials (ssid and pass), sets up an NTP server (pool.ntp.org), and specifies a GMT offset of 0 seconds.

```
1 #define BLYNK_PRINT Serial
2 #include <WiFi.h>
3 #include <WiFiClient.h>
4 #include <BlynkSimpleEsp32.h>
5 #include "time.h"
6
7 char auth[] = "";
8
9 const char* ssid = ""; // your network SSID (name)
10 const char* pass = ""; // your network password
11
12
13 const char* ntpServer = "pool.ntp.org";
14 const long gmtOffset_sec = 0; //28800
15 const int daylightOffset_sec = 7200;
```

# IoT – ESP32 with Blynk

## 1. Third Activity

- Go to GitHub
- Open file name: 3-Blynk-Servo.ino
- Copy the code
- Paste the code at Wikwo
- Add Blynk Library
- Add ESP32Servo Library

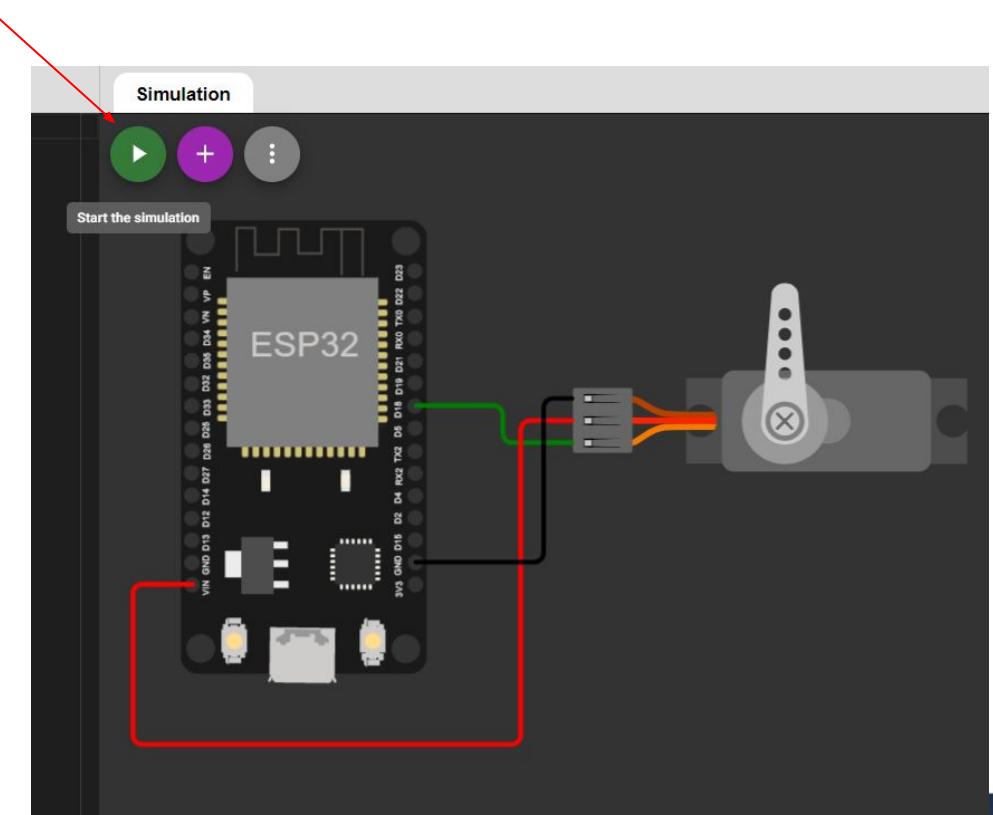
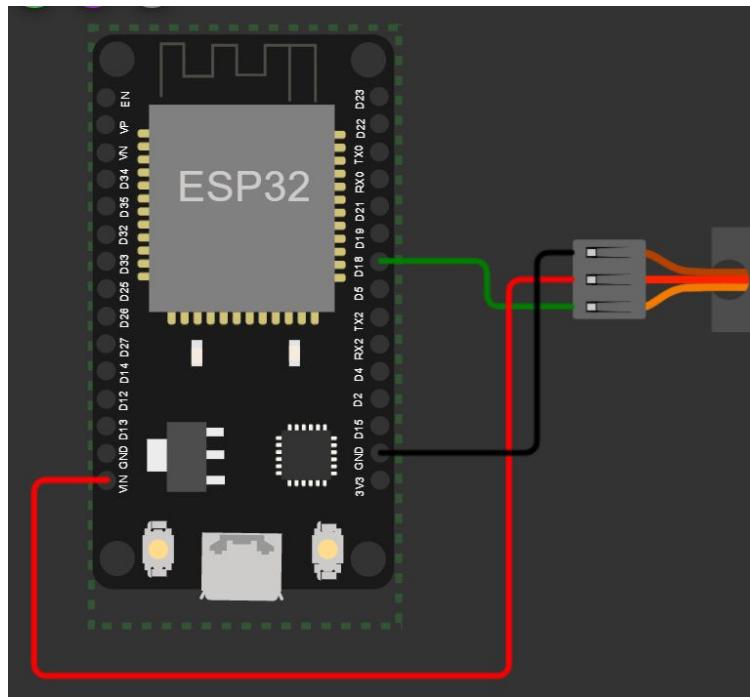


The screenshot shows a GitHub repository page for 'ainul21 / ESP32-with-Blynk'. The repository is public and contains one file, '3-Blynk-Servo.ino'. The commit history shows a single update from 'ainul21' with the message 'Update 3-Blynk-Servo.ino'. The code editor displays the following C++ code:

```
1 #define BLYNK_PRINT Serial
2 #include <WiFi.h>
3 #include <WiFiClient.h>
4 #include <BlynkSimpleEsp32.h>
5 #include "time.h"
6
7 char auth[] = "";
8
9 const char* ssid      = "";           // your network SSID (name)
10 const char* pass      = "";           // your network password
11
12 Servo servo;
13 BLYNK_WRITE(V3) {
14     servo.write(param.asInt());
15 }
16
```

# Iot – ESP32 with Blynk

1. Add Servo on the canvas
2. Connect correctly the component (GND, VIN, D18)
3. Start the simulation





UNIVERSITI SAINS ISLAM MALAYSIA  
جامعة العلوم الإسلامية الماليزية  
ISLAMIC SCIENCE UNIVERSITY OF MALAYSIA

# USIM

Pioneering Islamic Science  
Spearheading Knowledge

[www.usim.edu.my](http://www.usim.edu.my)

