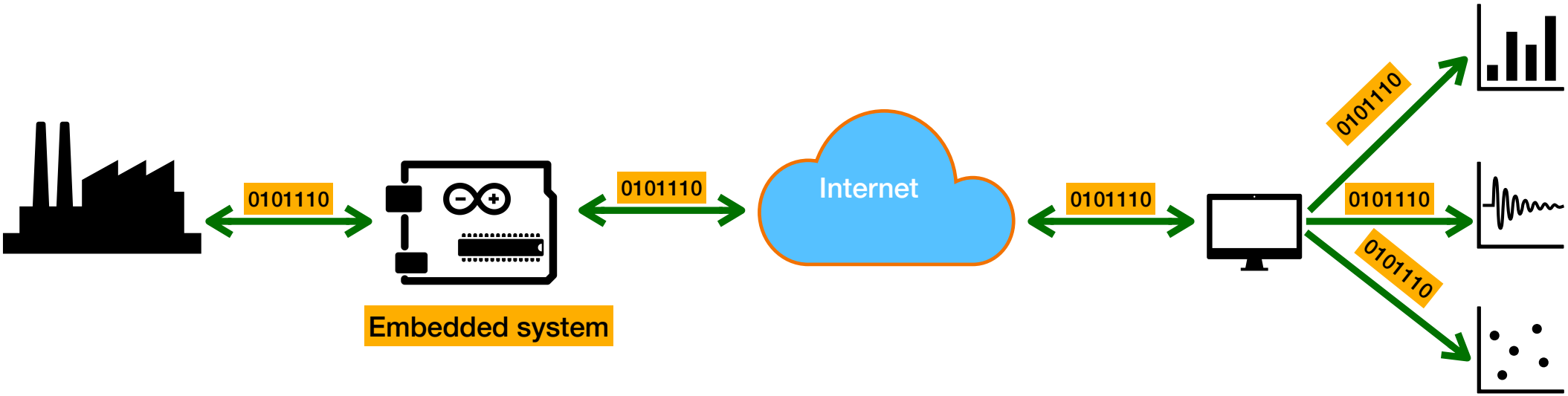
IoT with ESP32

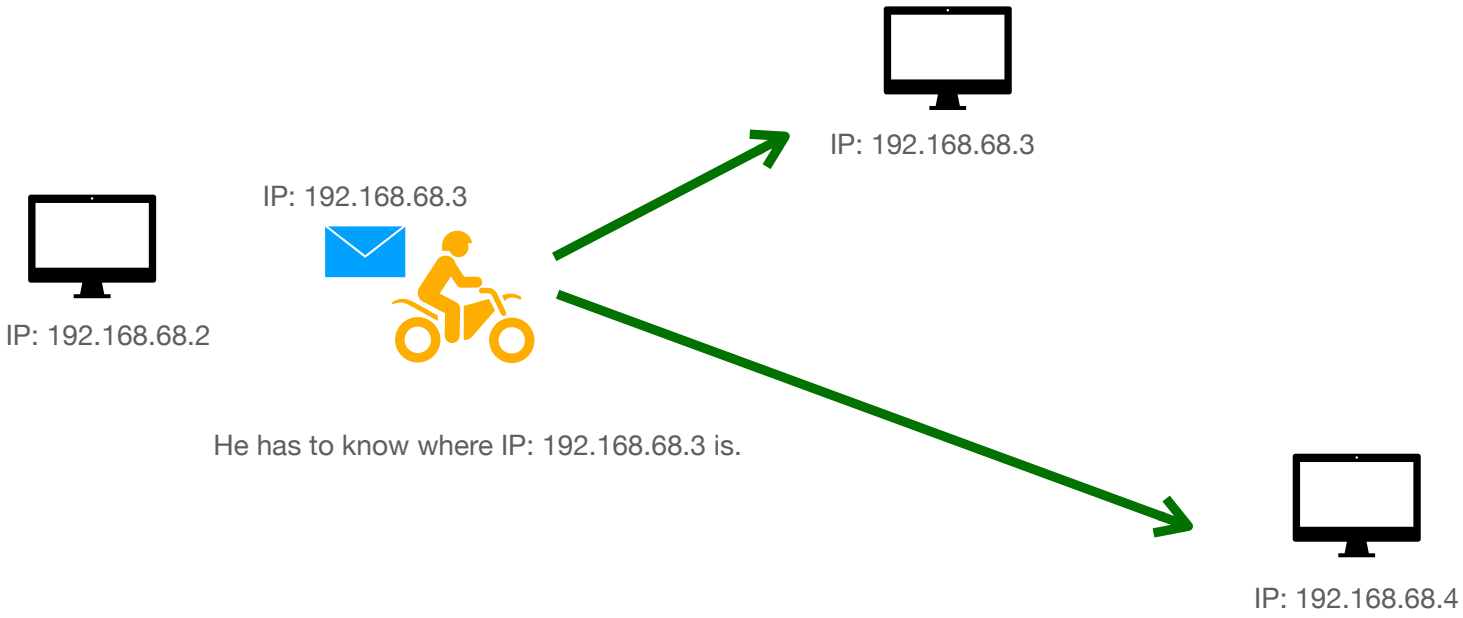
Apichart Intarapanich

IoT

Machines talk to machines

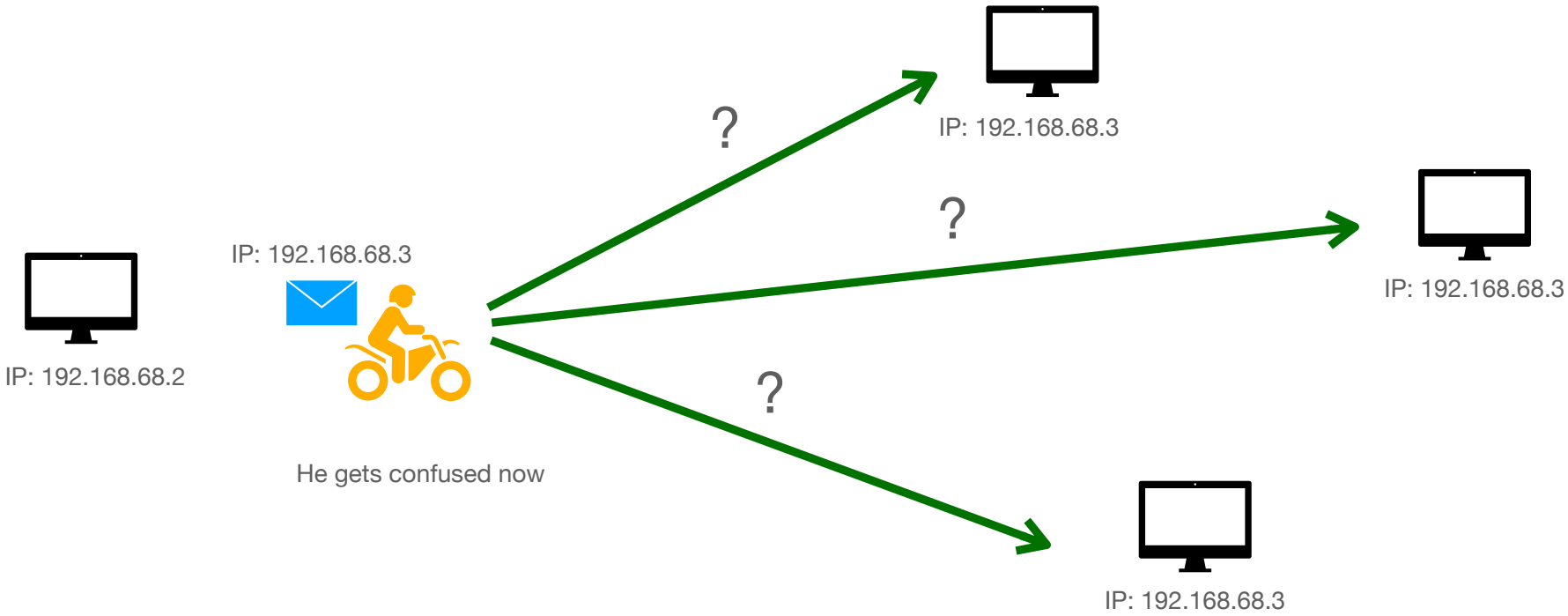


How does the internet work?



IP address

IP address should be unique



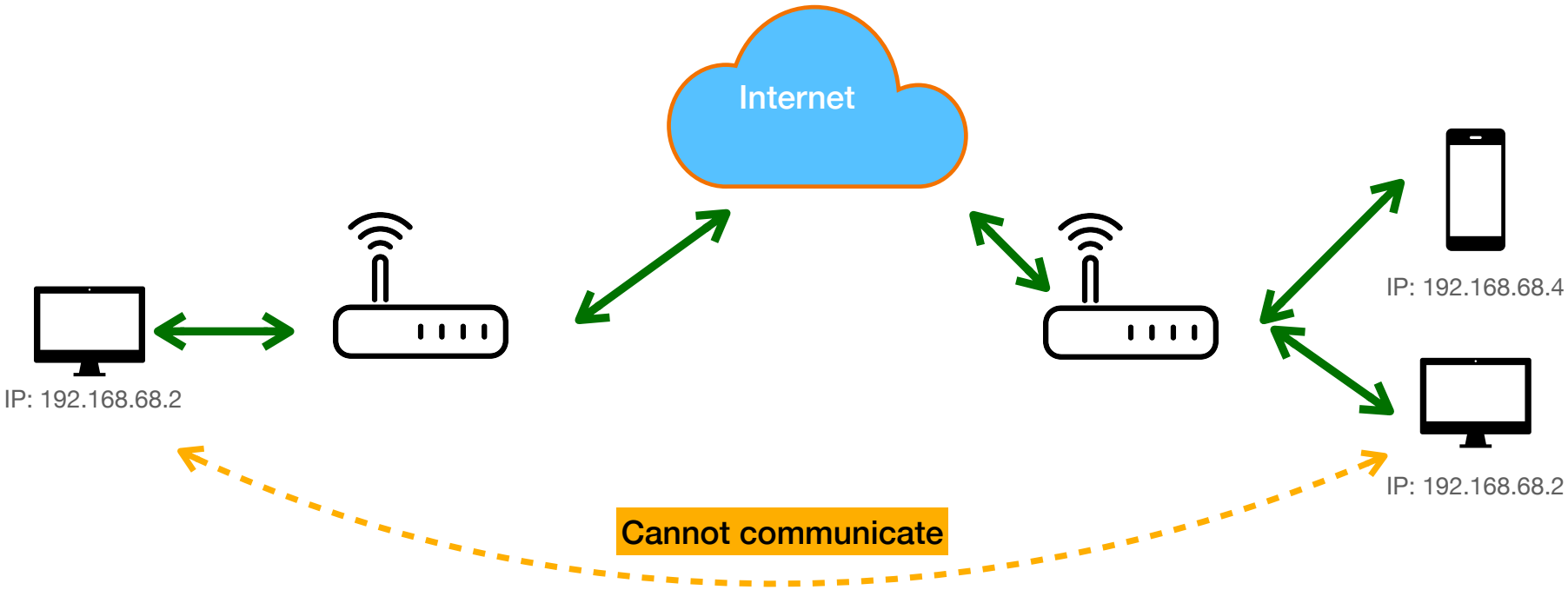
IP Version 4

32-bit IP address

- 4 billion IP address but there are 7 billions people on the planet
- To solve this problem, we use Network Translation Table (NAT) at the router
- The unique IP V4 address called *public IP.* As name infer, the address can be seen by everyone on the planet.

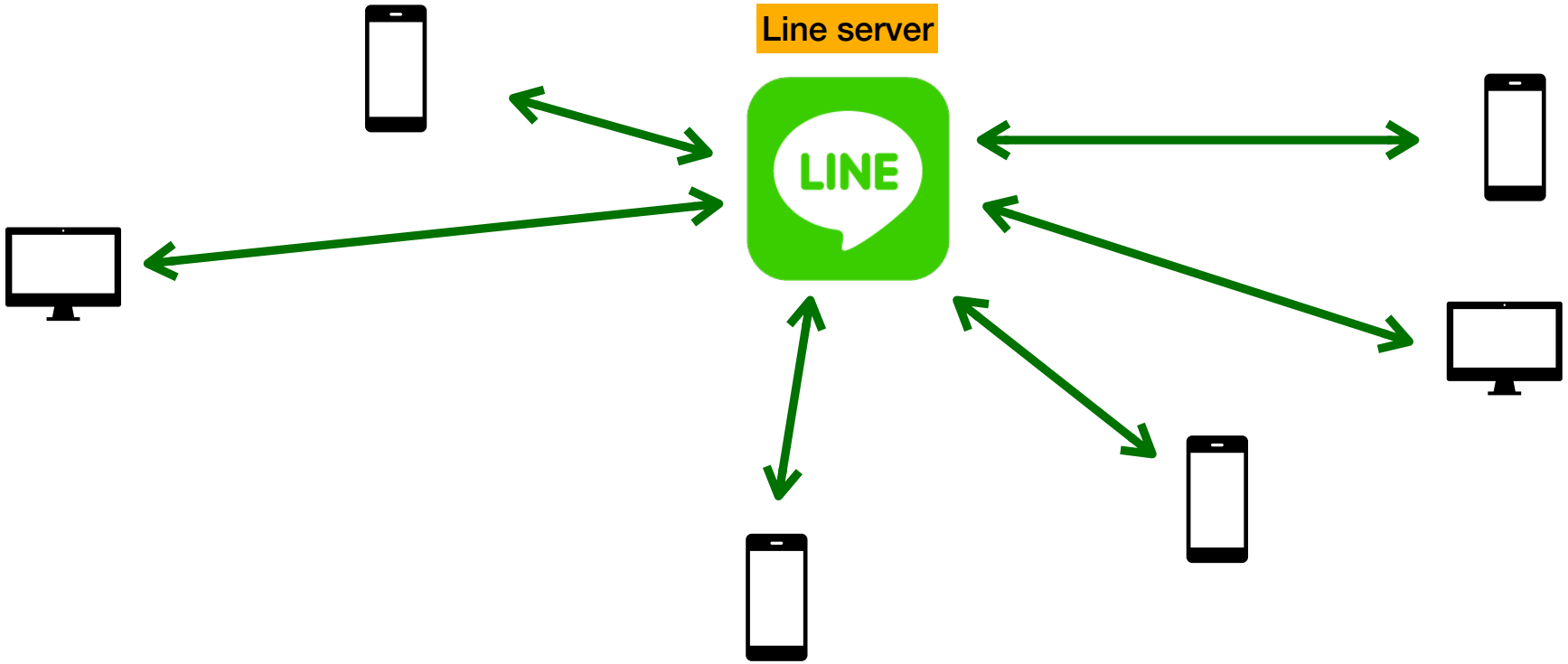
NAT Problem

Direct end-to-end communication



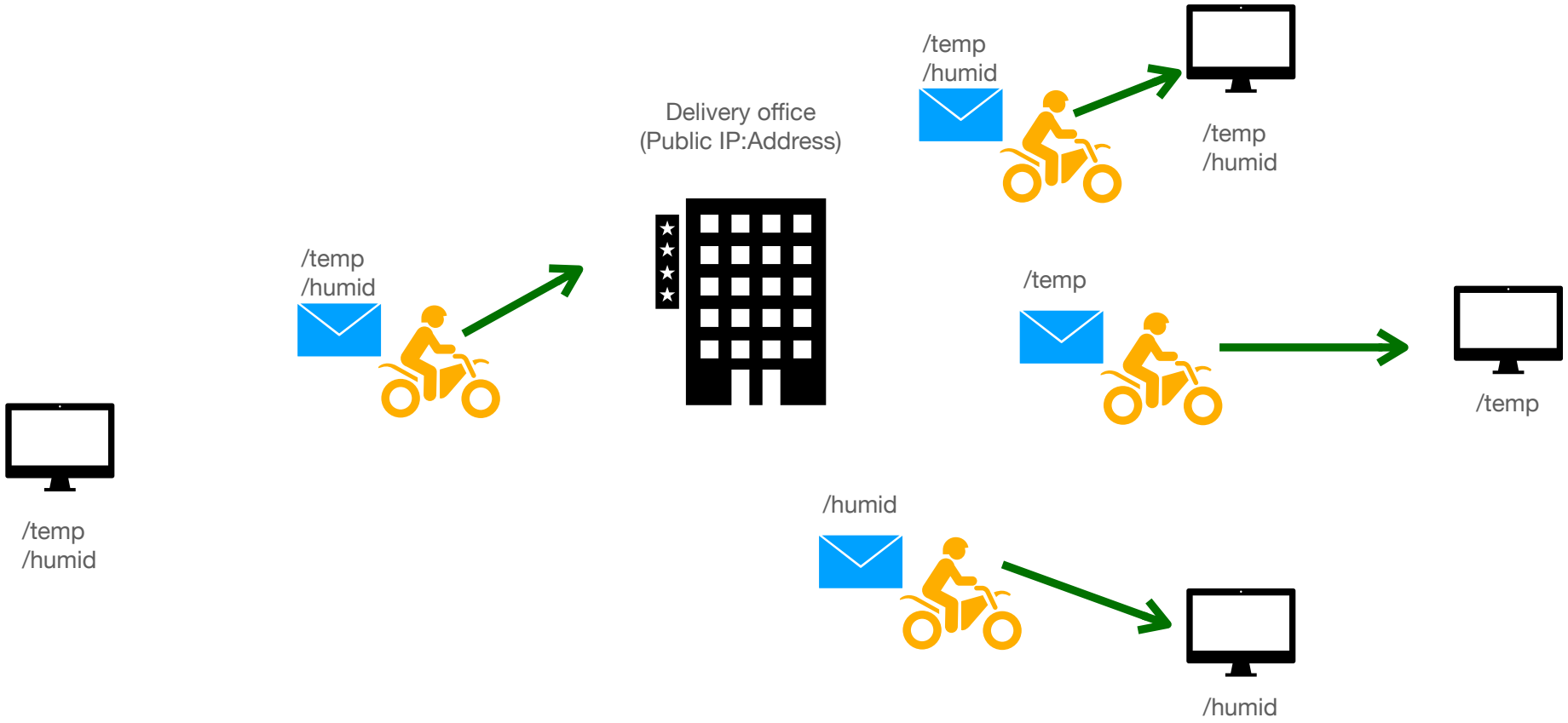
Chat app

Line chat



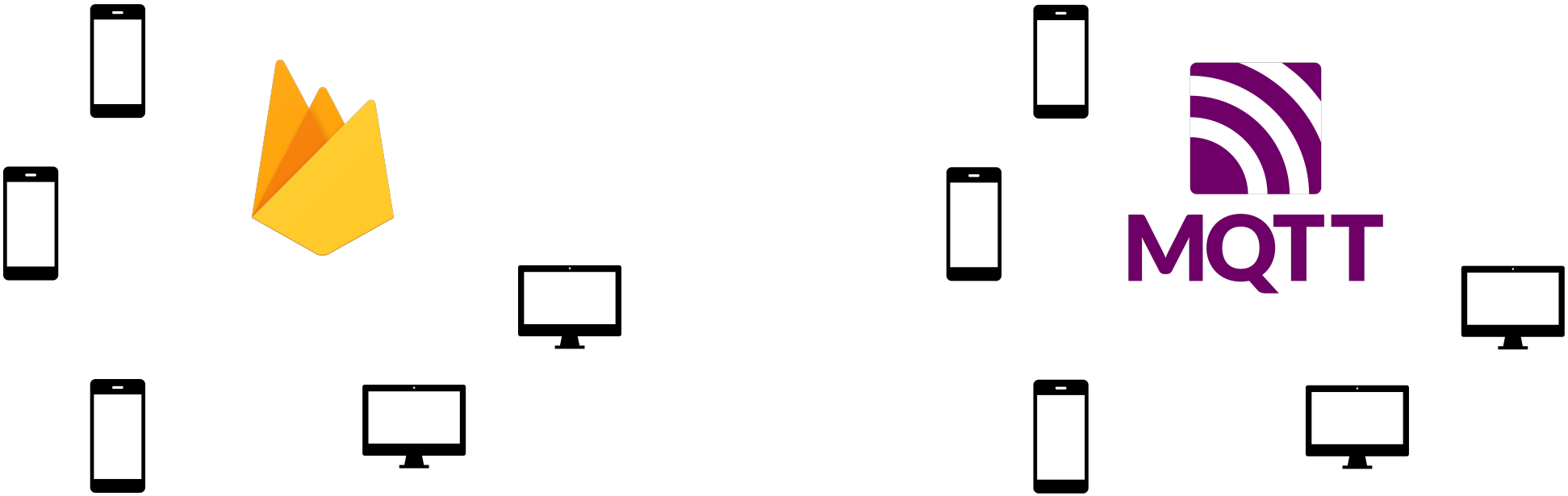
Create our own transport system

Create a private virtual network

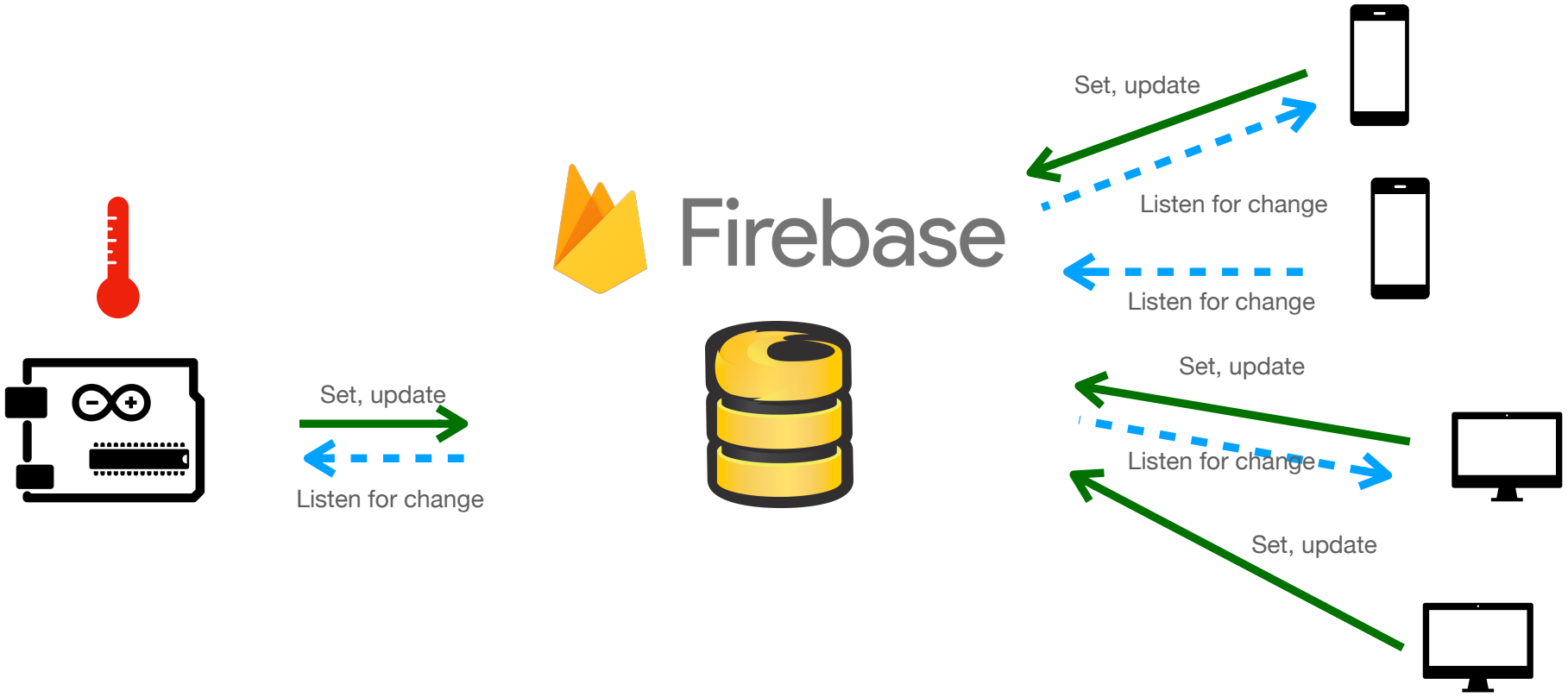


Delivery office

In this class we will use Google Firebase and MQTT Broker

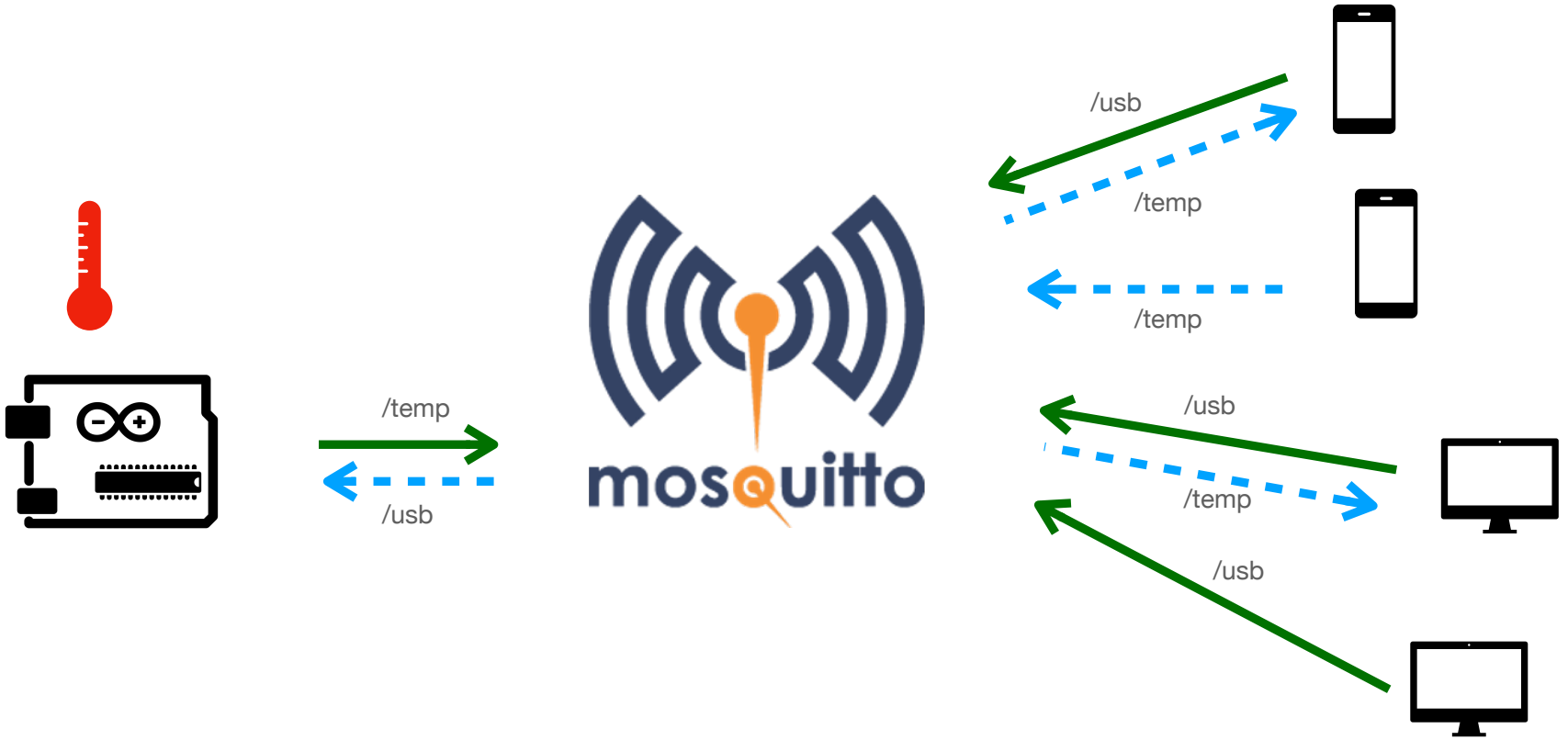


Firebase realtime database



MQTT publisher and subscriber

Pub/sub model



Topics and QoS

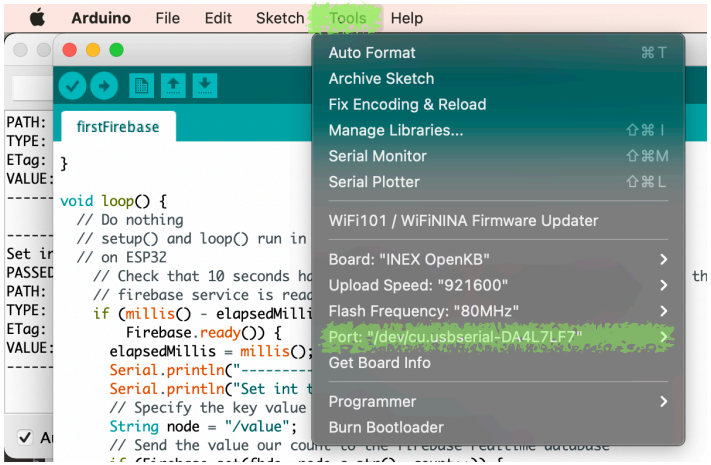
- Topic : /temp
- QoS levels : 0, 1 and 2
  - At most once (0)
    - This service level guarantees a best-effort delivery
  - At least once (1)
    - QoS level 1 guarantees that a message is delivered at least one time to the receiver
  - Exactly once (2)
    - This level guarantees that each message is received only once

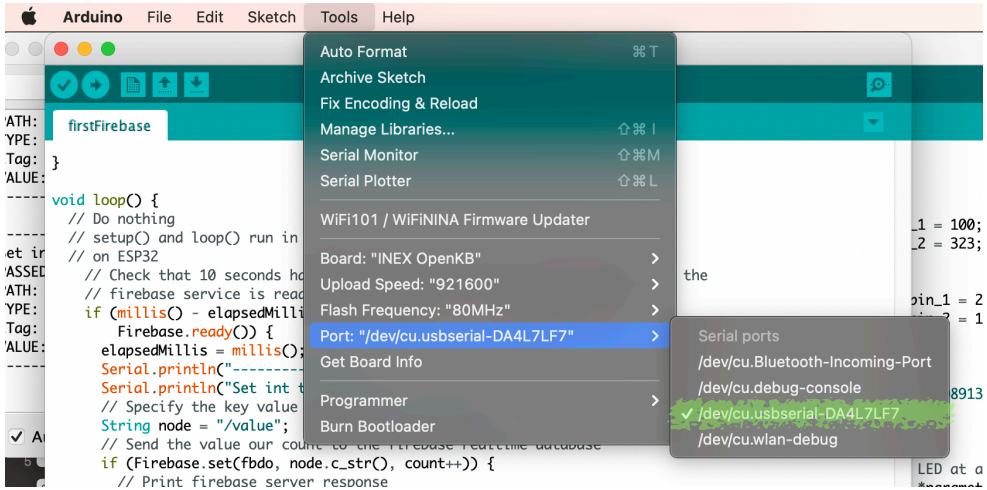
Why is MQTT protocol used in IoT

- There are many protocol to connect computers over the internet
- Only few of them can be fit in embedded system
- MQTT is a protocol that can fit in embedded system

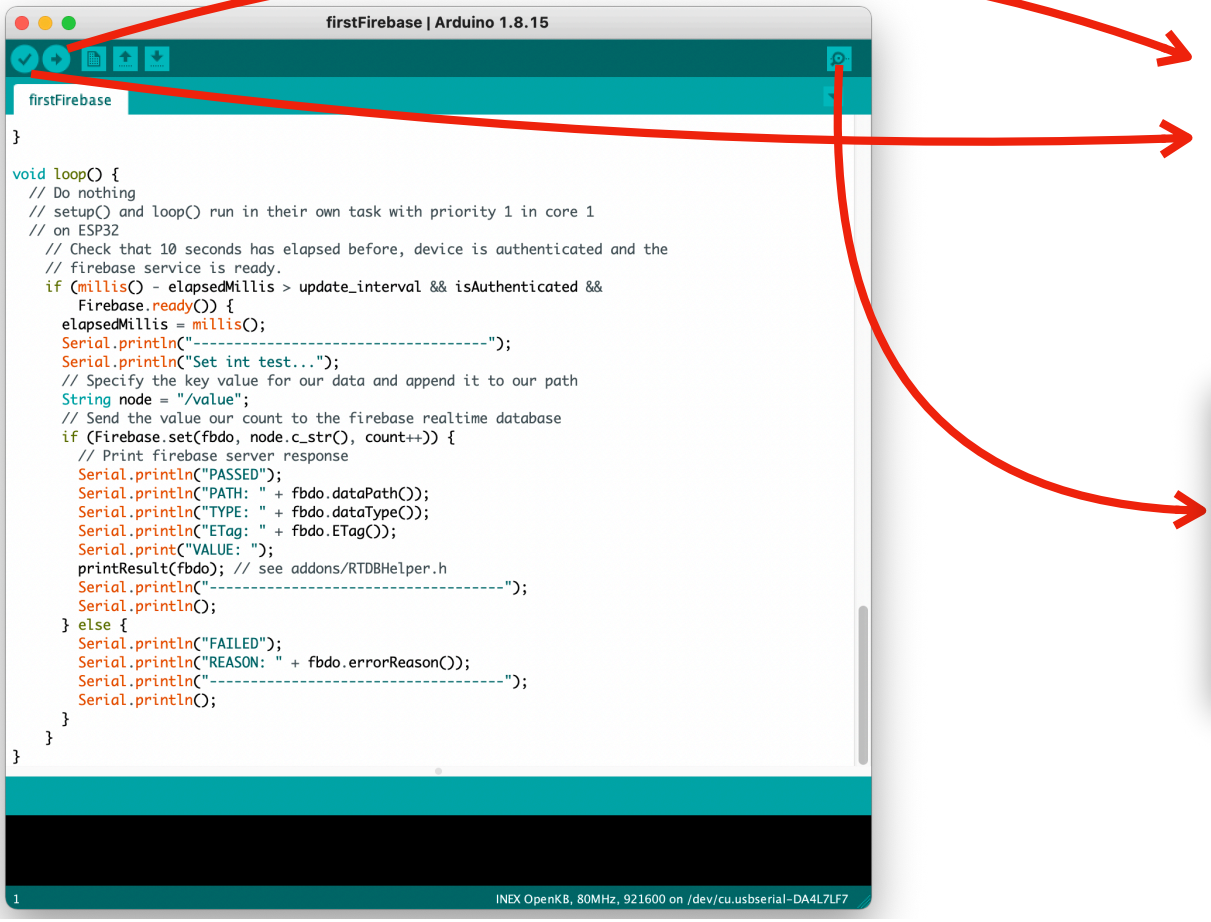
Basic Arduino IDE

Check board connection





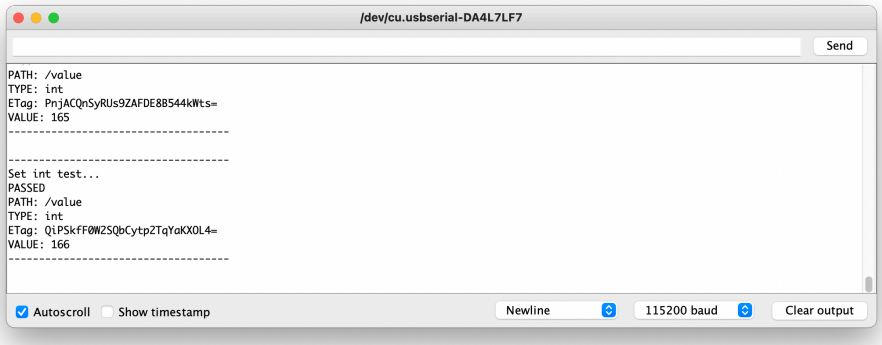
Basic Arduino IDE



Compile program

Compile and download

Open serial terminal



Sensors test

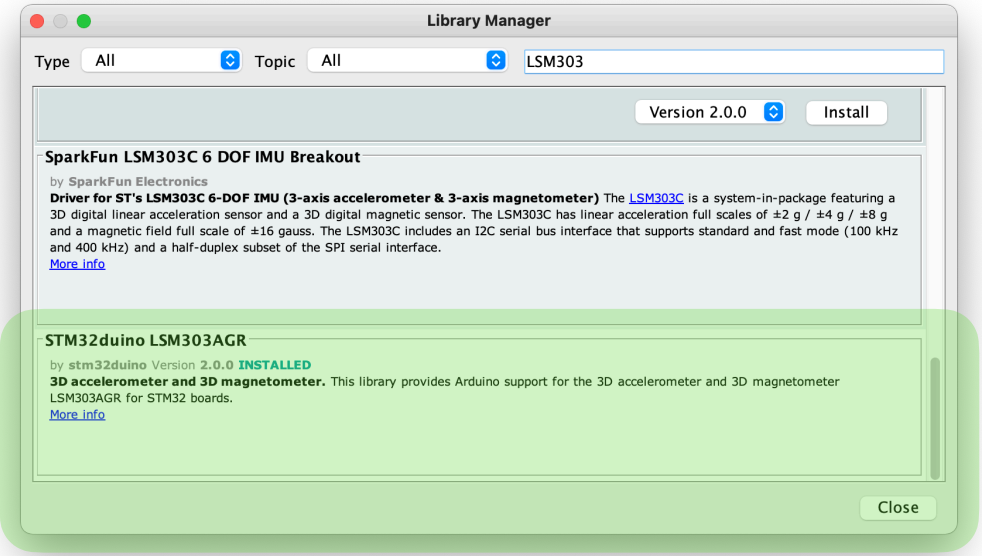
Test temperature sensor

readTemp.ino **LM73**

```
#include <Wire.h>
#define LM73_ADDR 0x4D

int analog_value = 0;
double temp=0;

// Pins
static const int led_pin_1 = 2;
static const int led_pin_2 = 12;

float readTemperature() {
  Wire1.beginTransmission(LM73_ADDR);
  Wire1.write(0x00); // Temperature Data Register
  Wire1.endTransmission();

  uint8_t count = Wire1.requestFrom(LM73_ADDR, 2);
  float temp = 0.0;
  if (count == 2) {
    byte buff[2];
    buff[0] = Wire1.read();
    buff[1] = Wire1.read();
    temp += (int)(buff[0]<<1);
    if (buff[1]&0b10000000) temp += 1.0;
    if (buff[1]&0b01000000) temp += 0.5;
    if (buff[1]&0b00100000) temp += 0.25;
    if (buff[0]&0b10000000) temp *= -1.0;
  }
  return temp;
}

void setup() {
  Serial.begin(115200);
  Wire1.begin(4, 5);
}

void loop() {
      temp = readTemperature();
      Serial.print("Temp:");Serial.println(temp);
      delay(5000);
}
```

Test LSM303 sensor

`lsm303Test.ino` LSM303 is an accel and mag sensors




```
#include <LSM303AGR_ACC_Sensor.h>
#include <LSM303AGR_MAG_Sensor.h>

#define DEV_I2C Wire1   //Define which I2C bus is
used. Wire1 for the Arduino Due
#define SerialPort Serial


// Components.
LSM303AGR_ACC_Sensor Acc(&DEV_I2C);
LSM303AGR_MAG_Sensor Mag(&DEV_I2C);

void setup() {
  // Led.
  pinMode(12, OUTPUT);
  DEV_I2C.begin(4, 5);

  // Initialize serial for output.
  SerialPort.begin(115200);

  // Initialize I2C bus.
  DEV_I2C.begin();

  // Initlialize components.
  Acc.begin();
  Acc.Enable();
  Acc.EnableTemperatureSensor();
  Mag.begin();
  Mag.Enable();
}

void loop() {
  // Led blinking.
  digitalWrite(12, HIGH);
  delay(250);
  digitalWrite(12, LOW);
  delay(250);

  // Read accelerometer LSM303AGR.
  int32_t accelerometer[3];
  Acc.GetAxes(accelerometer);

  // Read temperature LSM303AGR.
  float temperature;
  Acc.GetTemperature(&temperature);

  // Read magnetometer LSM303AGR.
  int32_t magnetometer[3];
  Mag.GetAxes(magnetometer);

  // Output data.
  SerialPort.print("| Acc[mg]: ");
  SerialPort.print(accelerometer[0]);
  SerialPort.print(" ");
  SerialPort.print(accelerometer[1]);
  SerialPort.print(" ");
  SerialPort.print(accelerometer[2]);
  SerialPort.print(" | Mag[mGauss]: ");
  SerialPort.print(magnetometer[0]);
  SerialPort.print(" ");
  SerialPort.print(magnetometer[1]);
  SerialPort.print(" ");
  SerialPort.print(magnetometer[2]);
  SerialPort.print(" | Temp[C]: ");
  SerialPort.print(temperature, 2);
  SerialPort.println(" |");
}
```

Create our own server

Software installation

- Server app : Nodejs
- MQTT broker : mosquitto
- Application development environment : Node-red

Install nodejs on the server

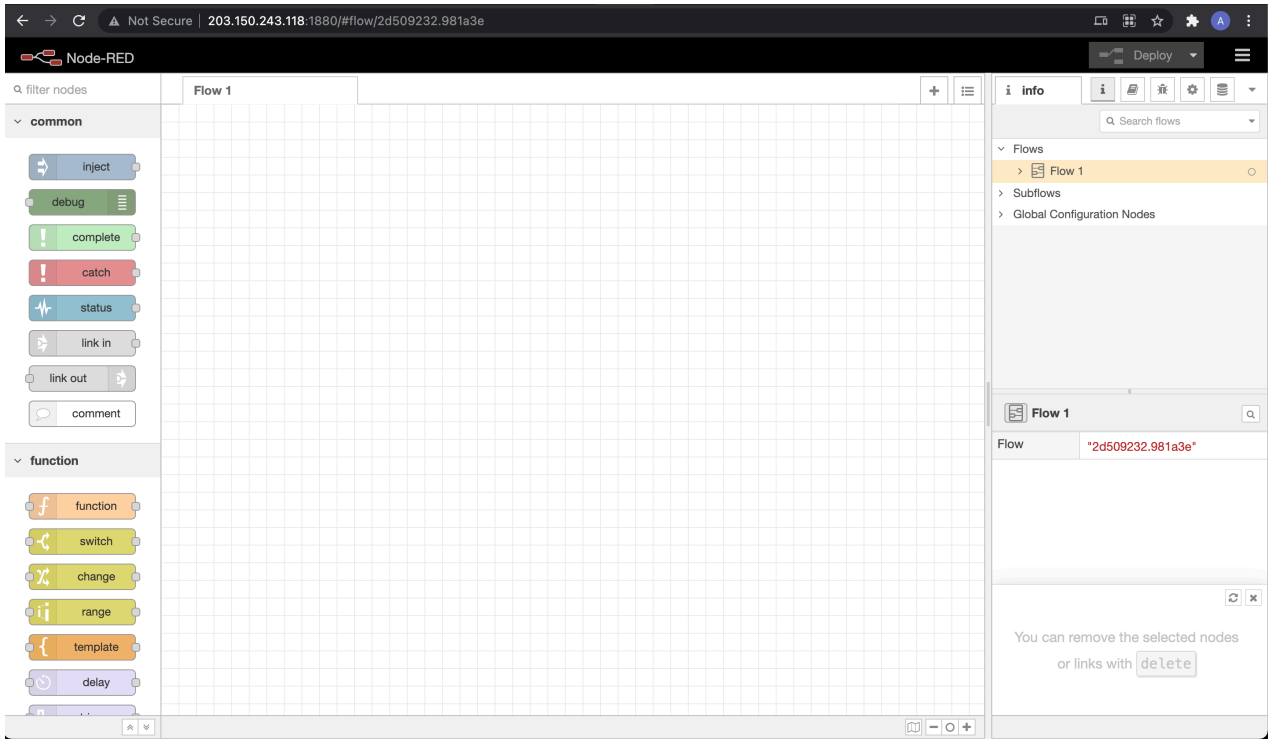
Link to command lines https://github.com/fllay/iotTraining/wiki/Server-software-installation

```
sudo apt-get update
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.2/install.sh | bash
exit

ssh root@203.150.243.118

root@voip:~#  nvm --version
0.35.2

root@voip:~# nvm install 10.5
Downloading and installing node v10.5.0...
Downloading https://nodejs.org/dist/v10.5.0/node-v10.5.0-linux-x64.tar.xz...
######################################################################## 100.0%
Computing checksum with sha256sum
Checksums matched!
Now using node v10.5.0 (npm v6.1.0)
Creating default alias: default -> 10.5 (-> v10.5.0)

root@voip:~# node -v
v10.5.0
root@voip:~# npm -v
6.1.0
```

Install node red on the server

```
npm install -g --unsafe-perm node-red
```

http://203.150.243.118:1880/



Start node red at boot

```
root@voip:~# sudo npm install -g pm2

[..................] | fetchMetadata: sill resolveWithNewModule moment@2.29.1 checking
installable status
:
:
/usr/bin/pm2-runtime -> /usr/lib/node_modules/pm2/bin/pm2-runtime


+ pm2@5.1.0
added 180 packages from 203 contributors in 190.922s


root@voip:~# which node-red
/root/.nvm/versions/node/v10.5.0/bin/node-red



root@voip:~# pm2 start /root/.nvm/versions/node/v10.5.0/bin/node-red -- -v


[PM2] Spawning PM2 daemon with pm2_home=/root/.pm2
[PM2] PM2 Successfully daemonized
[PM2] Starting /root/.nvm/versions/node/v10.5.0/bin/node-red in fork_mode (1 instance)
[PM2] Done.
```

id	name	mode	↺	status	cpu	memory
0	node-red	fork	0	online	0%	22.3mb

```
root@voip:~# pm2 save
[PM2] Saving current process list...
[PM2] Successfully saved in /root/.pm2/dump.pm2


root@voip:~# pm2 startup
[PM2] Init System found: systemd
Platform systemd
Template

:
:
+---------------------------------------+
[PM2] Freeze a process list on reboot via:
$ pm2 save

[PM2] Remove init script via:
$ pm2 unstartup systemd

root@voip:~# pm2 list
```

id	name	mode	↺	status	cpu	memory
0	node-red	fork	0	online	0%	77.0mb

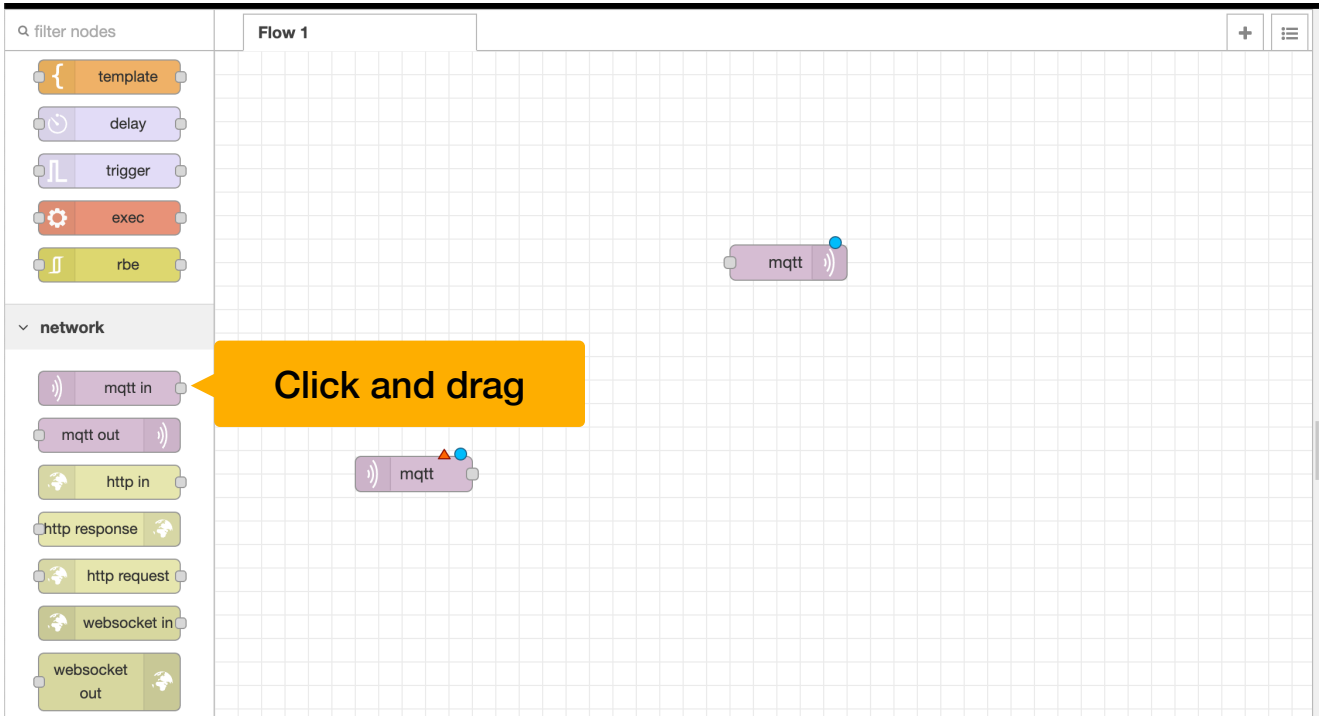
Install MQTT Broker

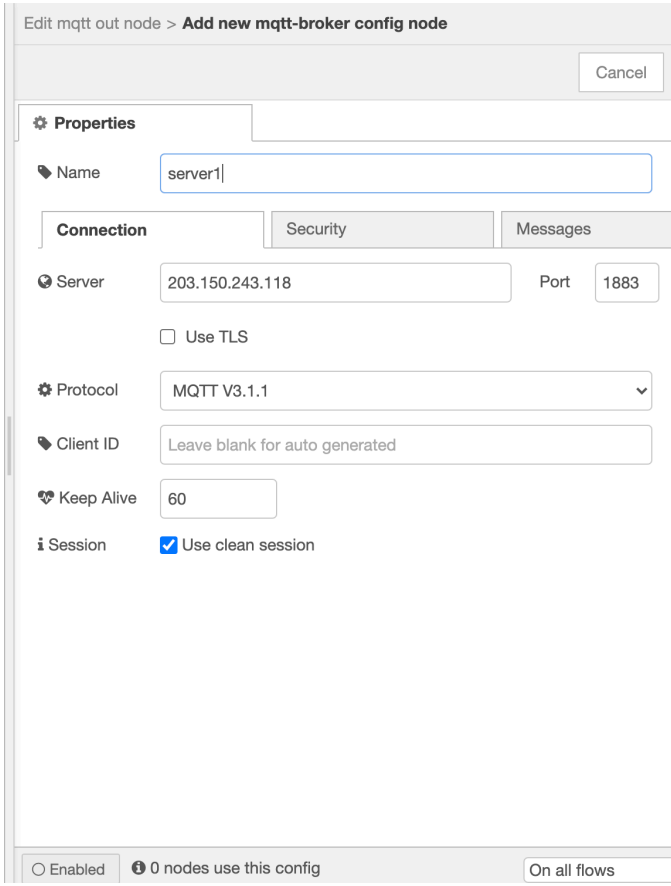
```
root@voip:~# sudo apt update -y && sudo apt install mosquitto mosquitto-clients -y
```

```
root@voip:~# sudo systemctl status mosquitto
● mosquitto.service - LSB: mosquitto MQTT v3.1 message broker
   Loaded: loaded (/etc/init.d/mosquitto; generated)
   Active: active (running) since Wed 2021-07-14 22:52:48 +07; 2min 32s ago
     Docs: man:systemd-sysv-generator(8)
    Tasks: 1 (limit: 1151)
   CGroup: /system.slice/mosquitto.service
           └─5133 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

Jul 14 22:52:48 voip systemd[1]: Starting LSB: mosquitto MQTT v3.1 message broker...
Jul 14 22:52:48 voip mosquitto[5127]:  * Starting network daemon: mosquitto
Jul 14 22:52:48 voip mosquitto[5127]:    ...done.
Jul 14 22:52:48 voip systemd[1]: Started LSB: mosquitto MQTT v3.1 message broker.
```

Test MQTT with node red

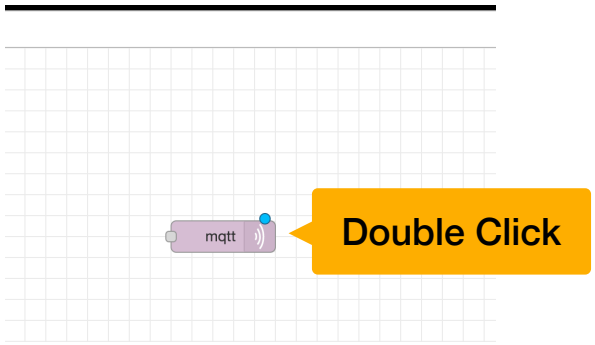
Create MQTT nodes

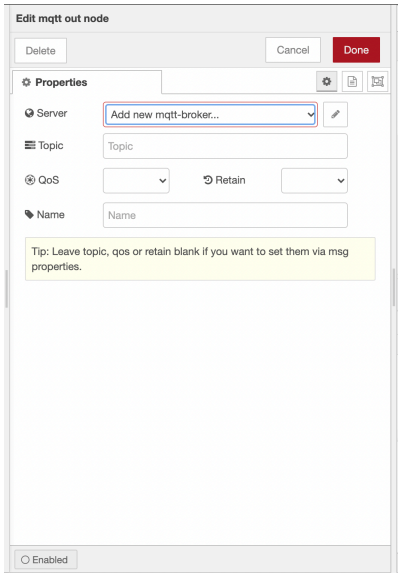


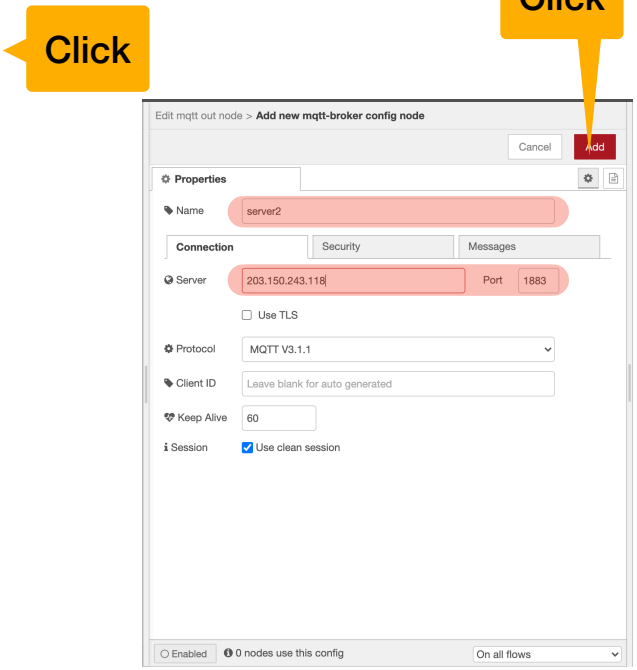


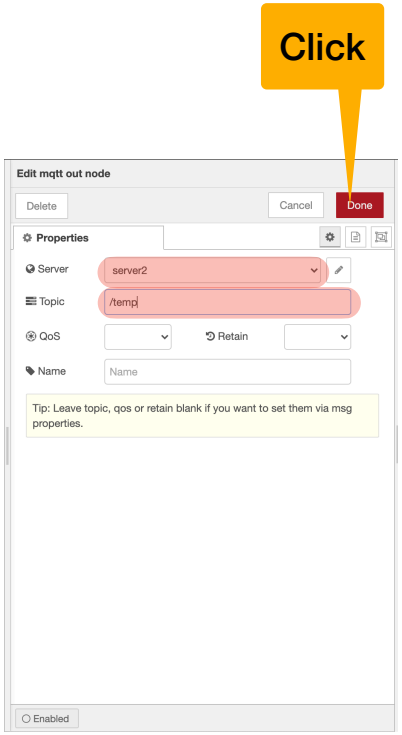
Test MQTT with node red

Config MQTT nodes



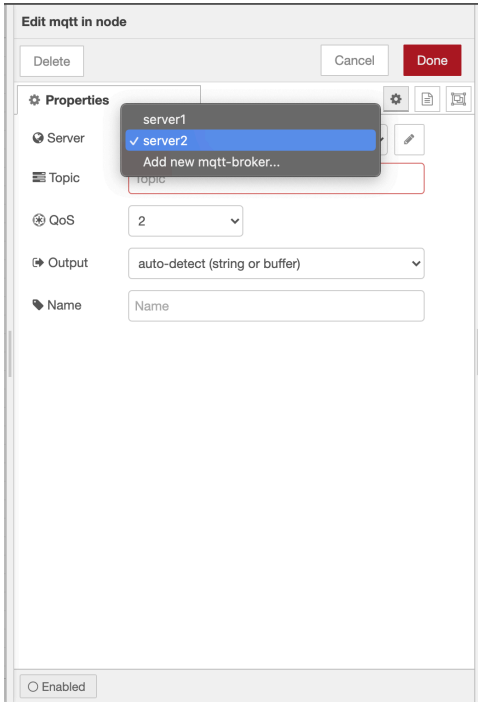


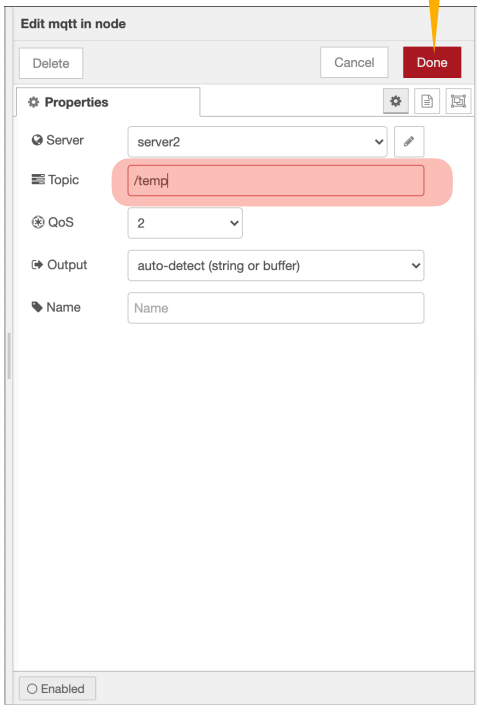




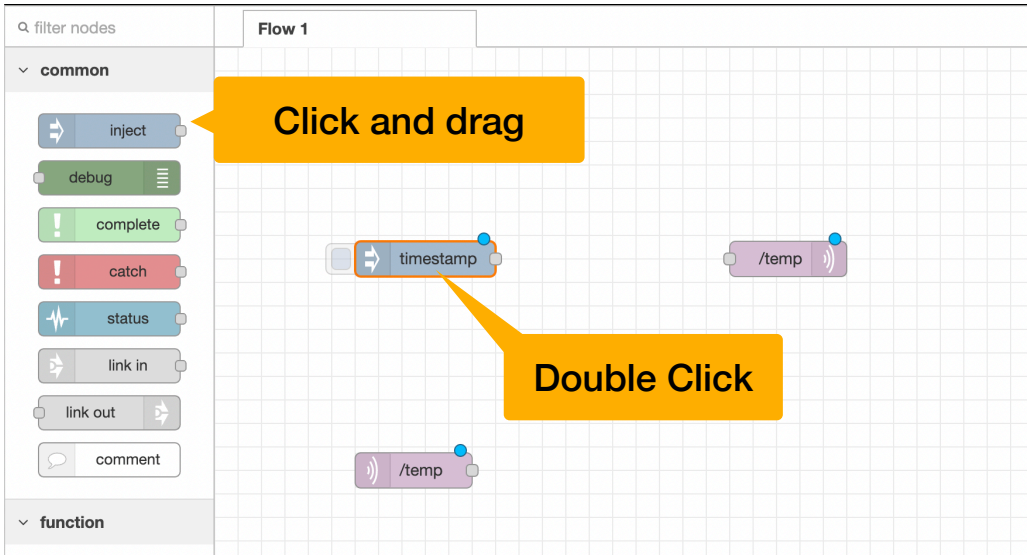
Test MQTT with node red

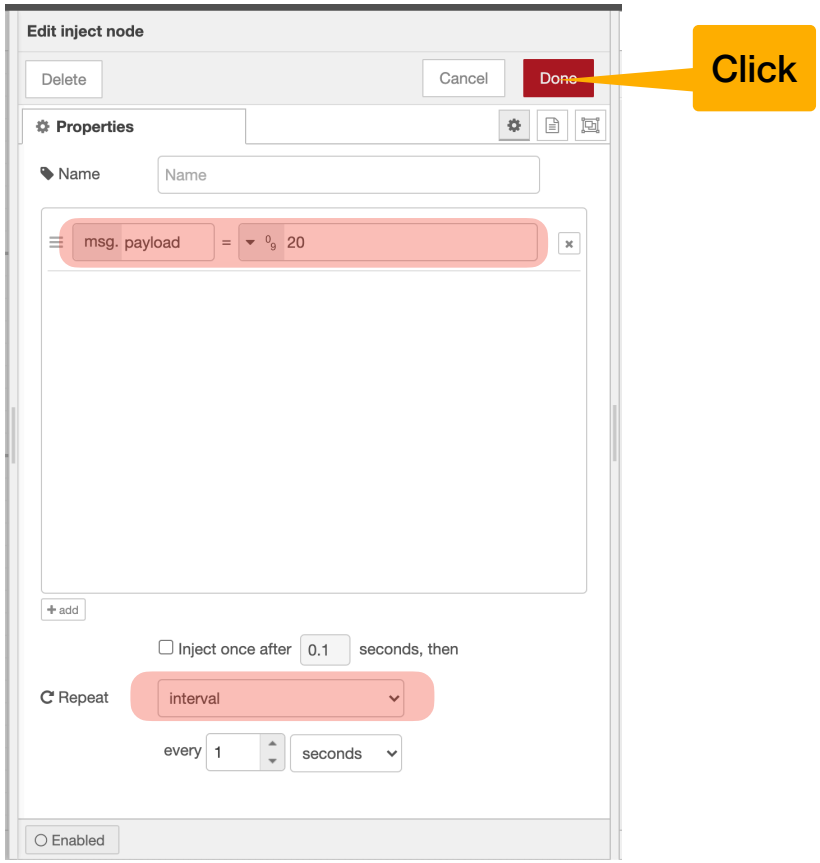
Config MQTT nodes



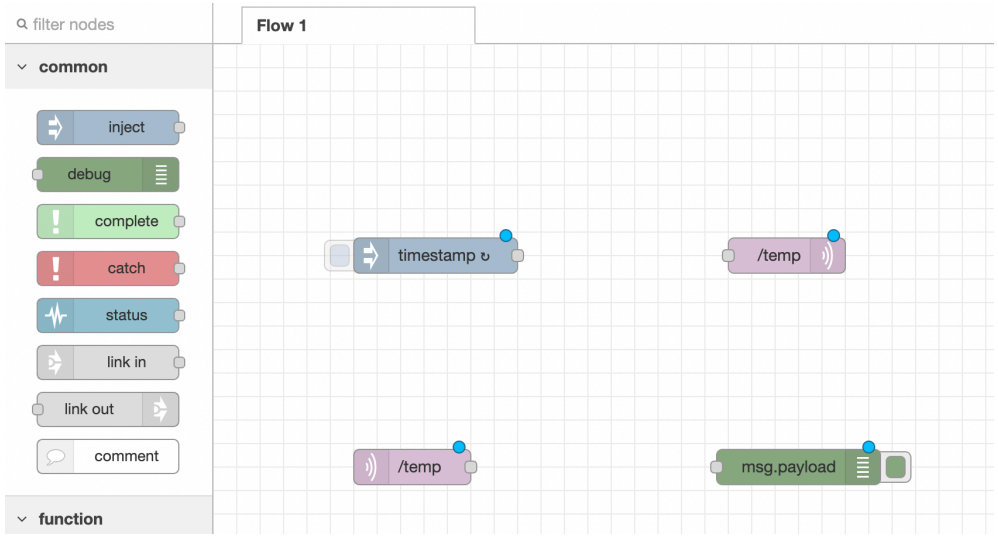


Create message source



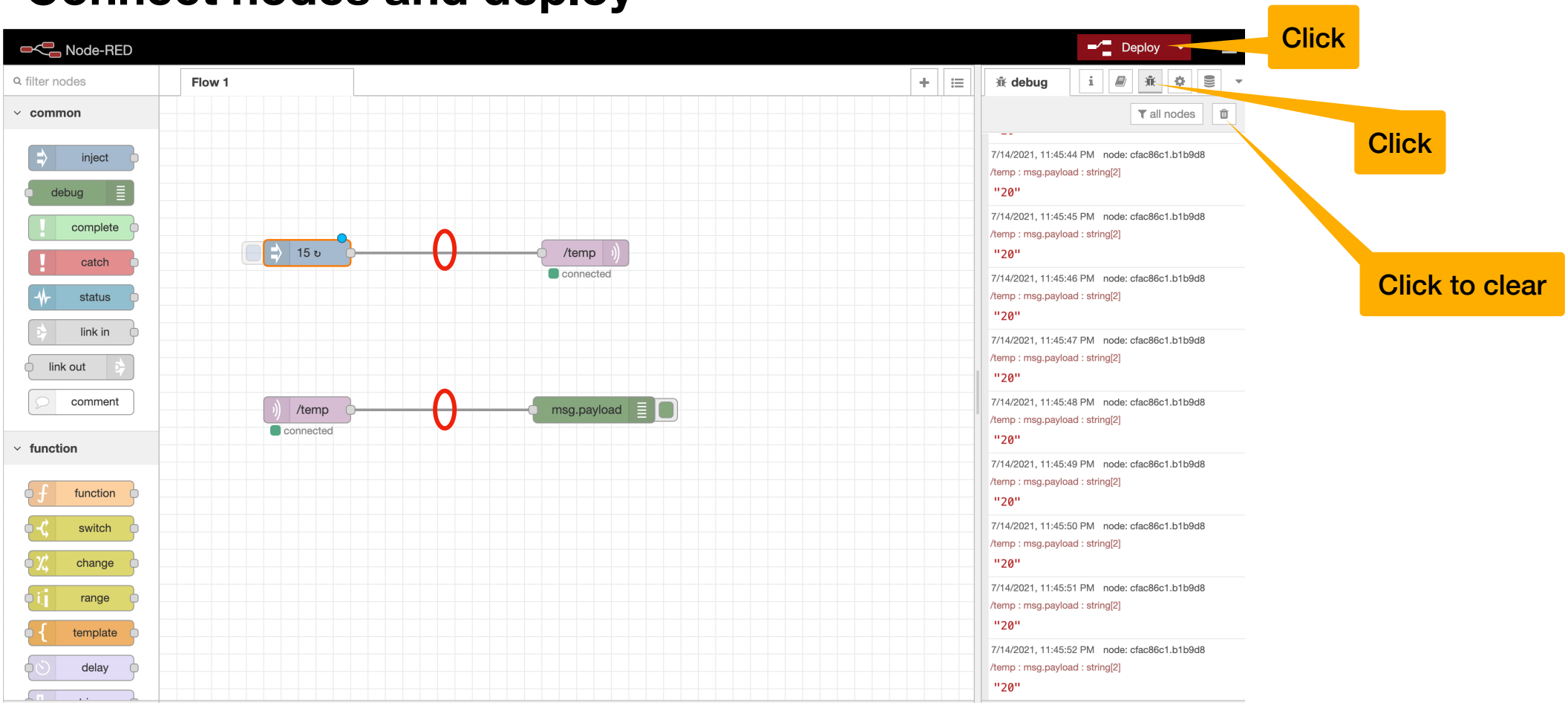


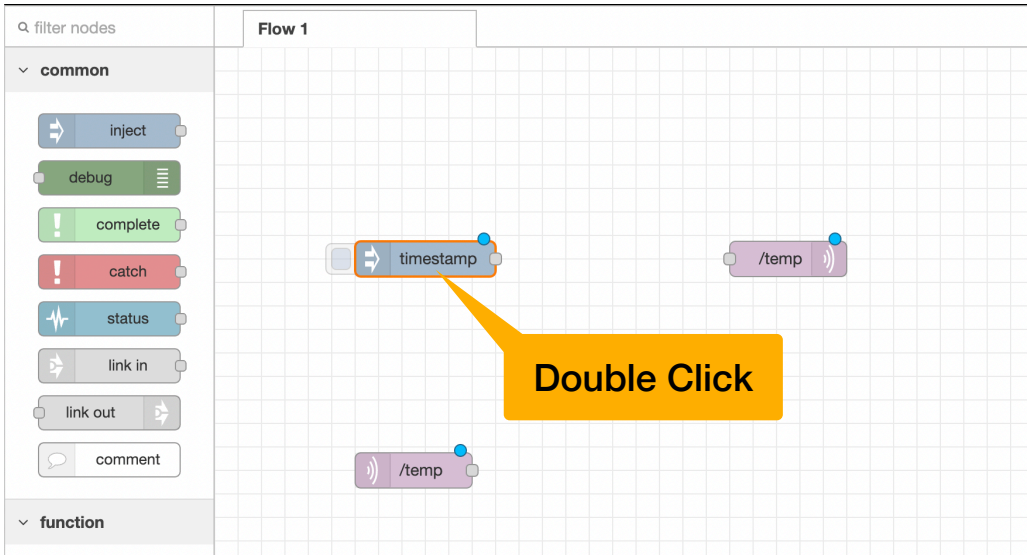
Create message debug

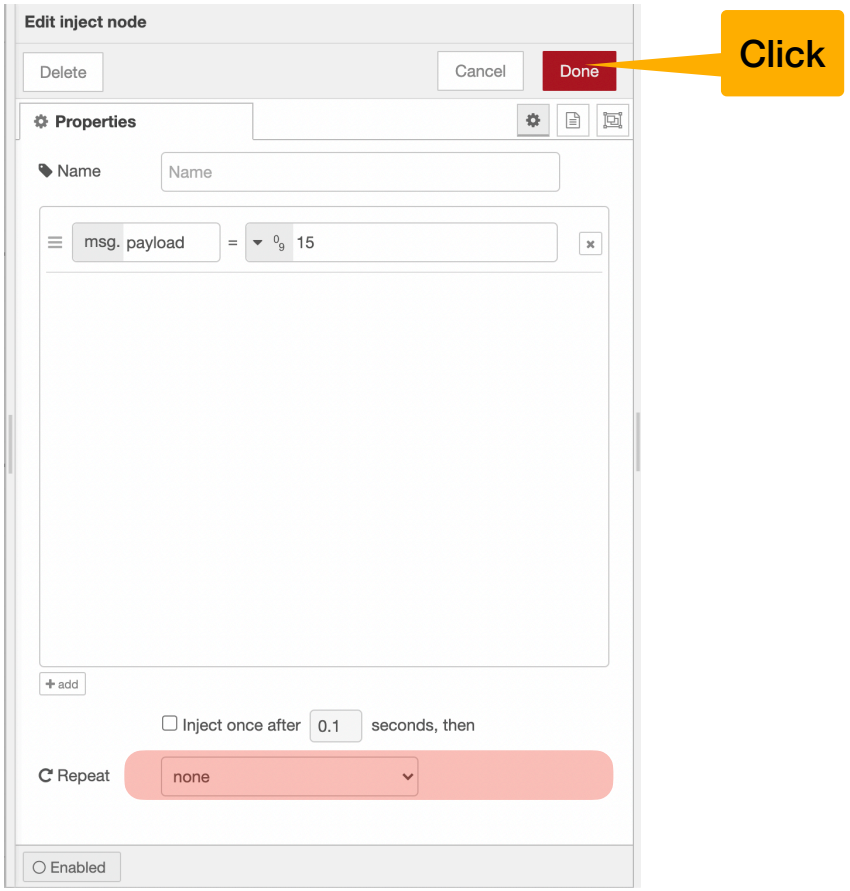


Connect nodes

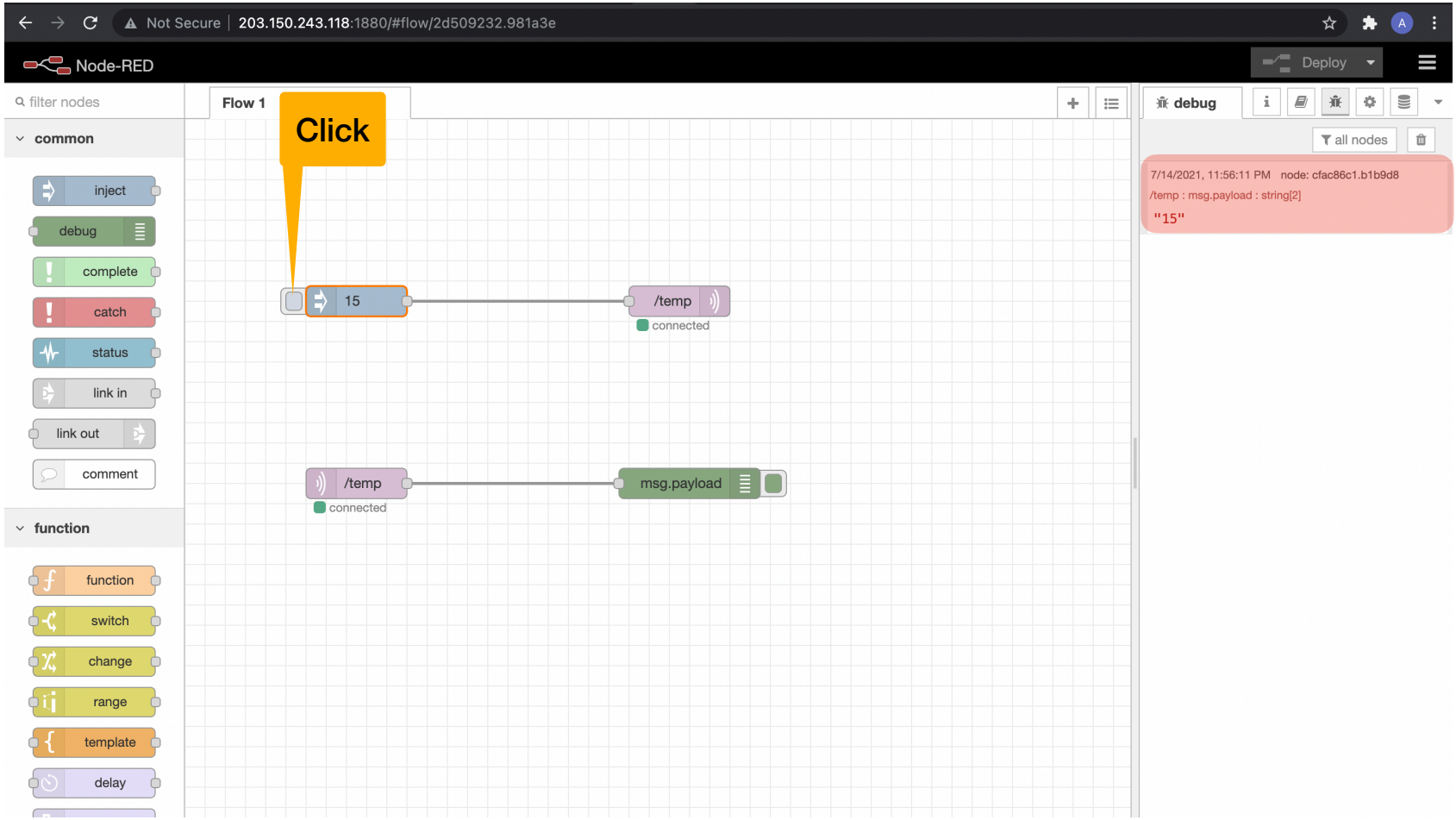
Connect nodes and deploy




filter nodes
Flow 1
common
inject
debug
complete
catch
status
link in
link out
comment
function
timestamp
/temp
/temp
Double Click

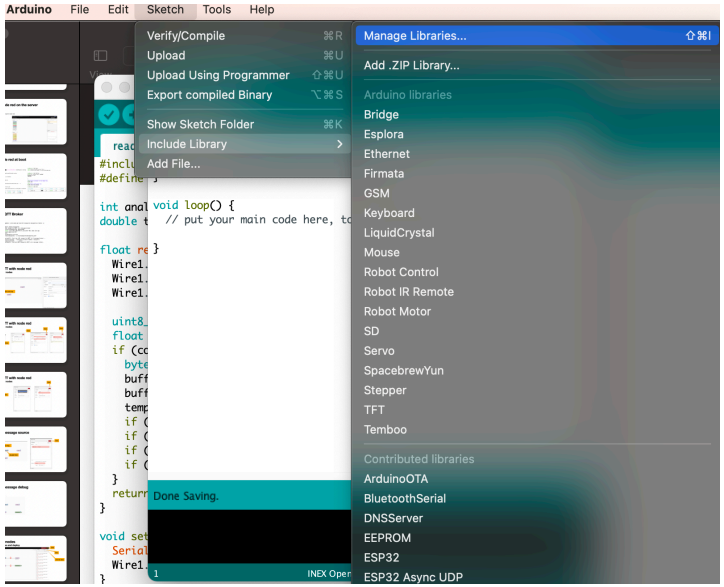

Edit inject node
Delete
Cancel
Done
Click
Properties
Name
Name
msg. payload
=
15
+ add
Inject once after
0.1
seconds, then
Repeat
none
Enabled

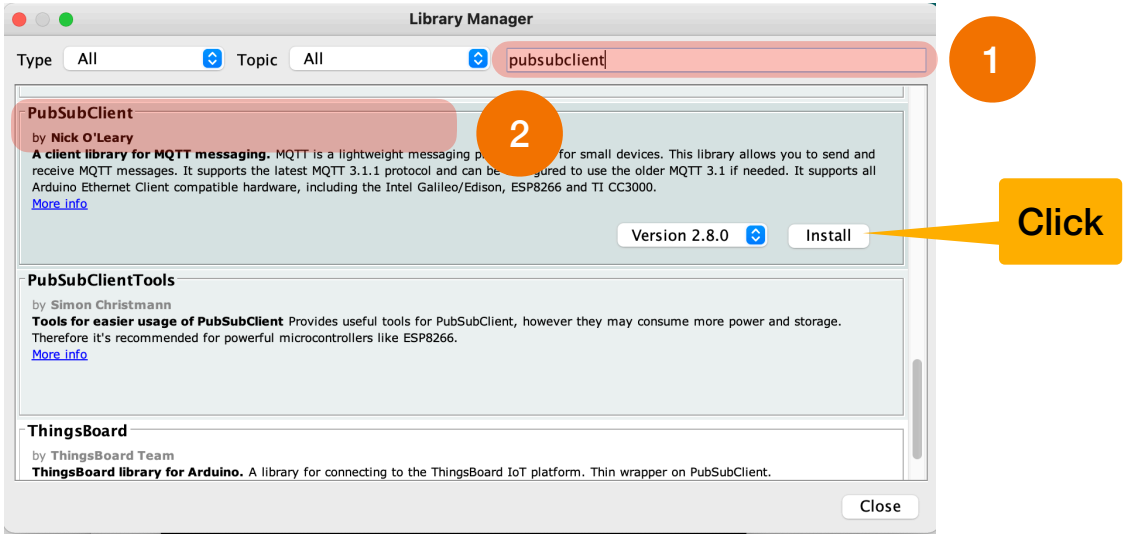
Test click to inject



Using ESP32 with our server

Install Arduino pubsubclient





Using ESP32 with our server

ESP32 MQTT connection to Mosquito

- Use ESP32 connect to wifi
- Once WiFi is connected, we create a MQTT client on ESP32.
- Then, we connect the MQTT client to our Mosquito server
- We can then create topics for publish and subscribe

Using ESP32 with our server

Connect to WiFi and MQTT broker : esp32mqtt.ino

```
void setup_wifi() {

    delay(10);
    // We start by connecting to a WiFi network
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
      digitalWrite(led_pin_1, HIGH);
      digitalWrite(led_pin_2, HIGH);
      delay(500);
      digitalWrite(led_pin_2, LOW);
      digitalWrite(led_pin_1, LOW);
      delay(500);


    }

    digitalWrite(led_pin_1, LOW);
    Serial.println("Connected to the WiFi
network");

    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}
```

```
const char* ssid = "CAKE";
const char* password = "0891300633";
const char* mqtt_server = "203.150.243.118";
```

Put your IP here

```
void setup() {
      // Configure pin
    pinMode(led_pin_1, OUTPUT);
    pinMode(led_pin_2, OUTPUT);
    Serial.begin(115200);
    setup_wifi();
    client.setServer(mqtt_server, 1883);

    if (!client.connected()) {

      digitalWrite(led_pin_2, HIGH);
      delay(500);
      digitalWrite(led_pin_2, LOW);
      delay(500);
     }

    digitalWrite(led_pin_2, LOW);
    client.setCallback(callback);
}
```

Using ESP32 with our server

Publish and subscribe to topics : `esp32mqtt.ino`

```
void setup() {
      // Configure pin
    pinMode(led_pin_1, OUTPUT);
    pinMode(led_pin_2, OUTPUT);
    Serial.begin(115200);
    setup_wifi();
    client.setServer(mqtt_server, 1883);

    if (!client.connected()) {

      digitalWrite(led_pin_2, HIGH);
      delay(500);
      digitalWrite(led_pin_2, LOW);
      delay(500);
     }

    digitalWrite(led_pin_2, LOW);
    client.setCallback(callback)
}
```

```
void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    for (int i = 0; i < length; i++) {
        Serial.print((char)payload[i]);
    }
    Serial.println();
}
```

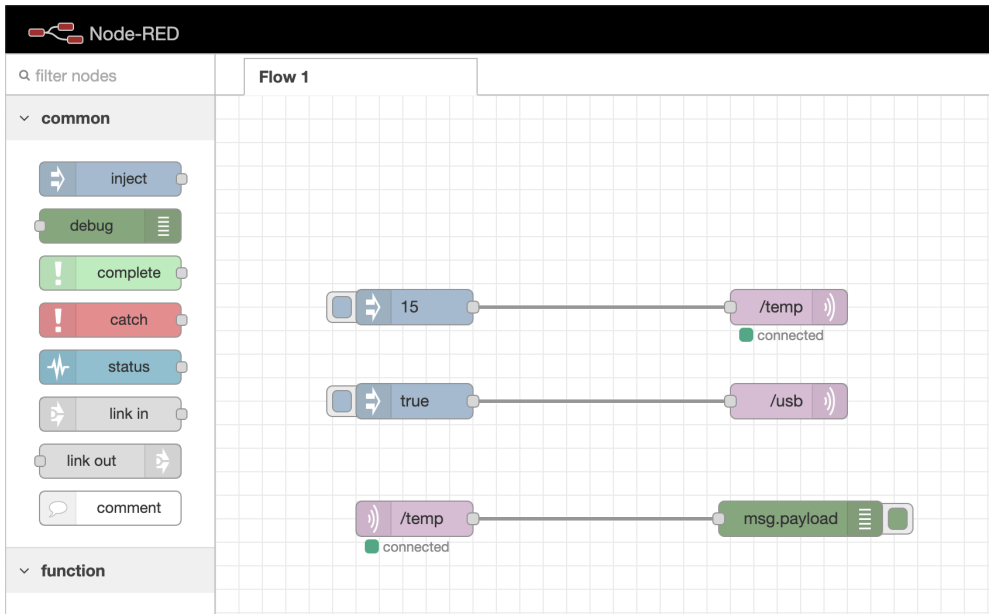
```
void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        // Attempt to connect
        if (client.connect("ESP32Client")) {
            Serial.println("connected");
            // Once connected, publish an announcement...
            client.publish("/temp", 0);
            // ... and resubscribe
            client.subscribe("/usb");
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            // Wait 5 seconds before retrying
            delay(5000);
        }
    }
}
```

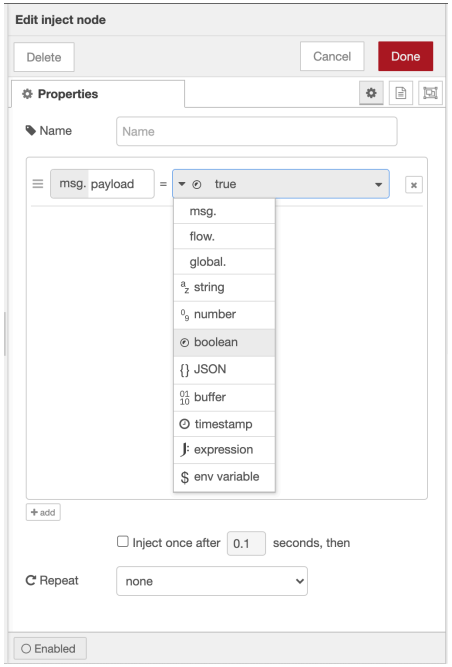
Publish a topic

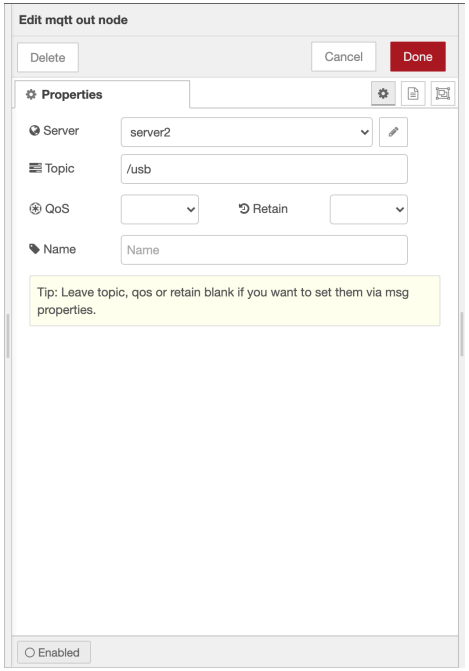
Subscribe to a topic

Add one more publisher to node red

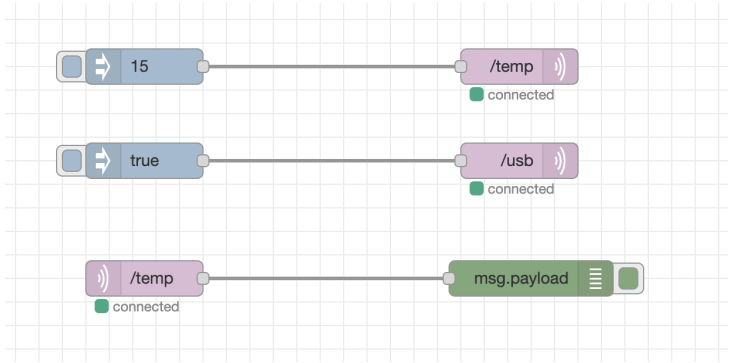
It will be used for turn on/off an USB device

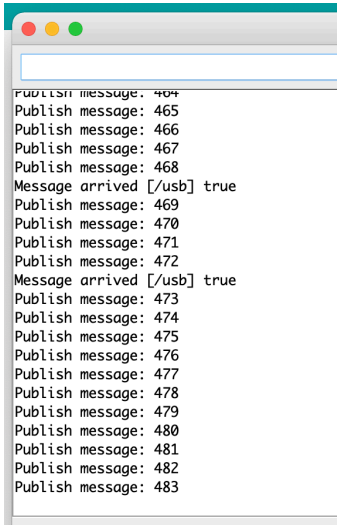






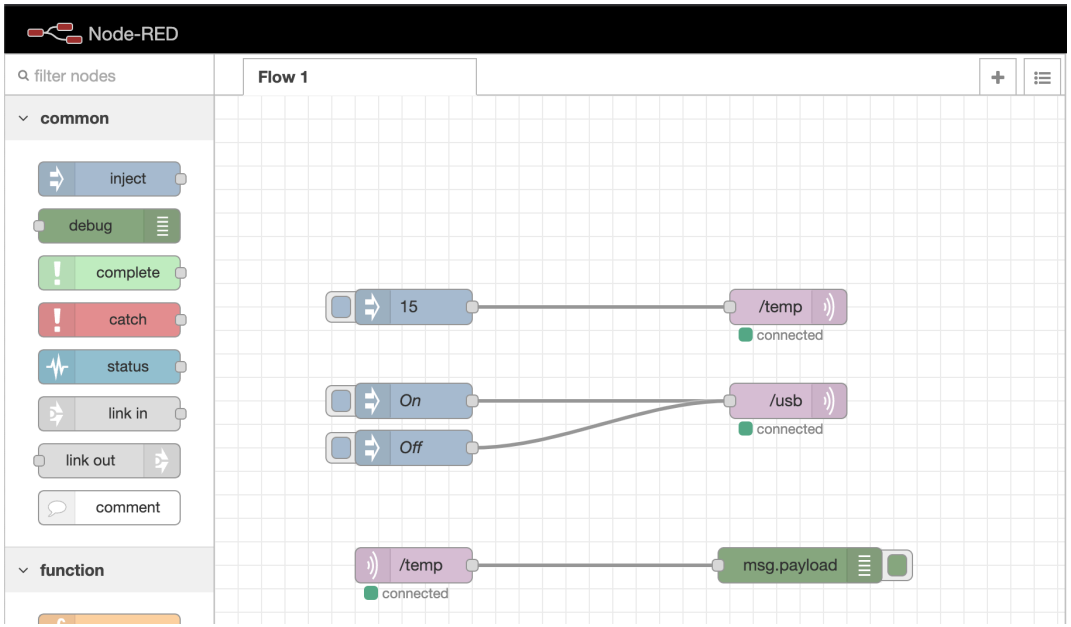
Test publish and subscribe

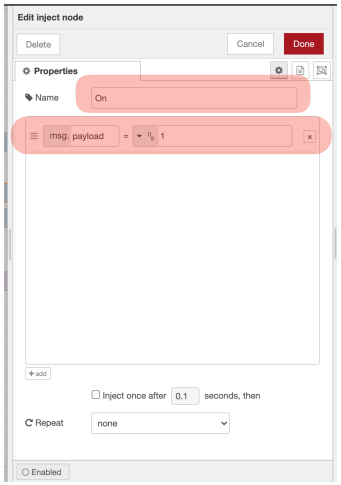


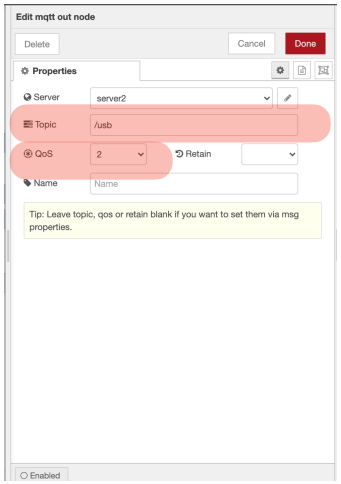


Using an USB device with MQTT

Node red flow







Using an USB device with MQTT

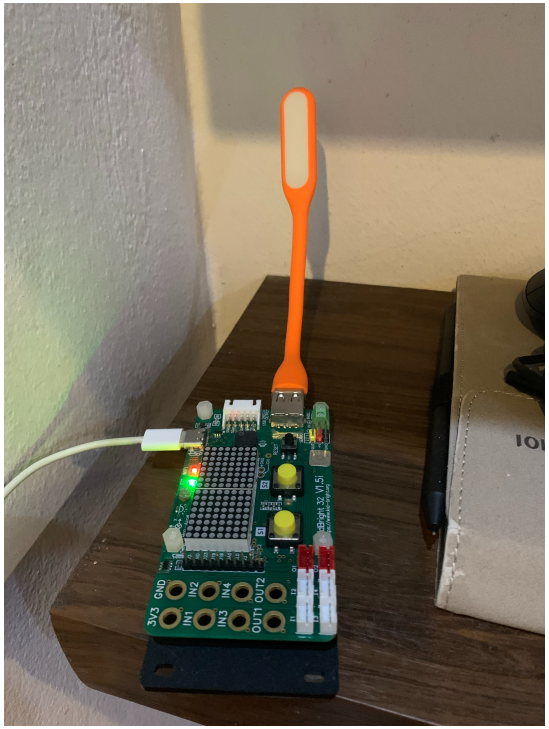
Modify ESP32 code : esp32mqttTempAndUSB.ino

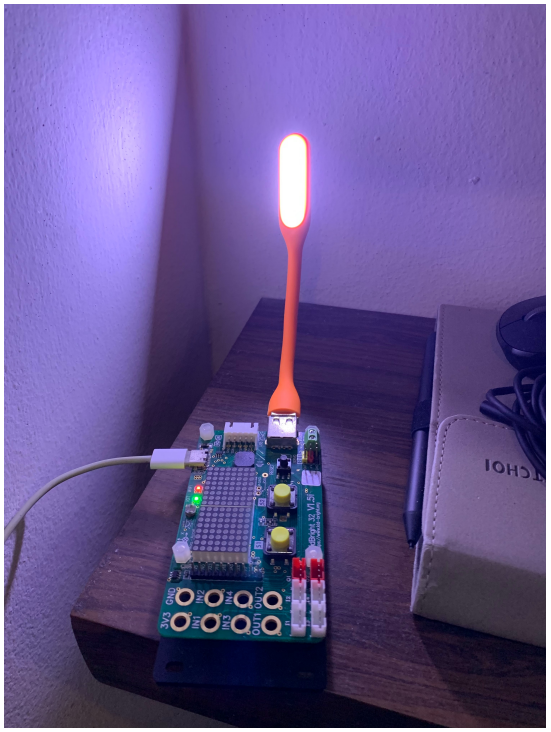
```
void callback(char* topic, byte* payload, unsigned int length) {

    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    if(strcmp((char *) topic, "/usb") == 0){
      payload[length] = '\0';
      String s = String((char*)payload);
      int f = s.toInt();
      for (int i = 0; i < length; i++) {
        Serial.print((char)payload[i]);
      }
      if(f == 1){
        digitalWrite(usb_pin, LOW);
        Serial.print("1");
      }else {
        digitalWrite(usb_pin, HIGH);
        Serial.print("0");
      }
    }
    Serial.println();
}
```

```
// Pins
static const int led_pin_1 = 2; //WiFI
static const int led_pin_2 = 12; //IoT
static const int usb_pin = 25; //USB
```

```
void setup() {
      // Configure pin
    pinMode(led_pin_1, OUTPUT);
    pinMode(led_pin_2, OUTPUT);
    pinMode(usb_pin, OUTPUT);

    digitalWrite(usb_pin, HIGH);
```

Connect Hardware and test

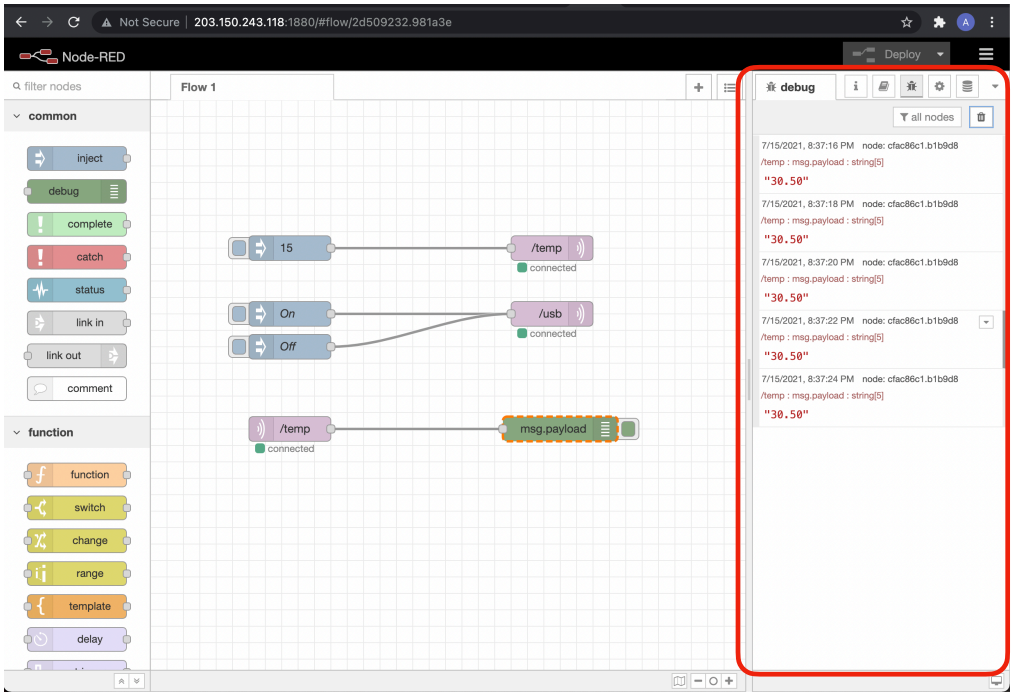




Publish temperature to Node red

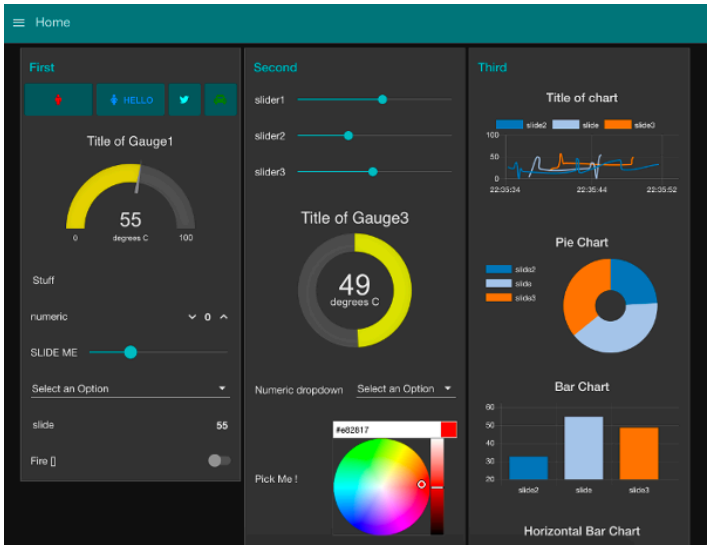
esp32mqttTempAndUSB.ino

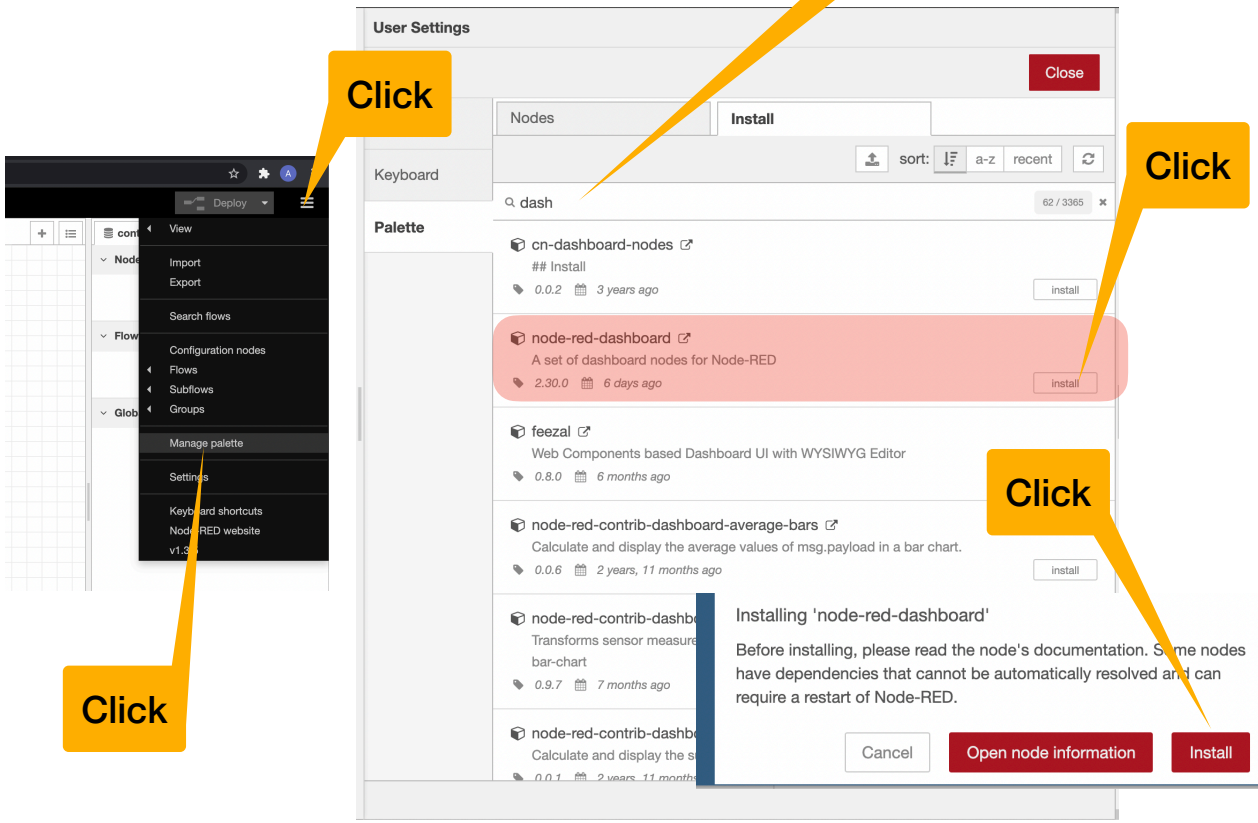
```
void loop() {

    if (!client.connected()) {
        reconnect();
    }
    client.loop();

    long now = millis();
    if (now - lastMsg > 2000) {
        lastMsg = now;
        ++value;
        //snprintf (msg, 75,  value);
        sprintf(msg,"%2.2f",readTemperature());
        //Serial.print("Publish message: ");
        //Serial.println(msg);
        client.publish("/temp", msg);
    }
}
```



Node red dashboad

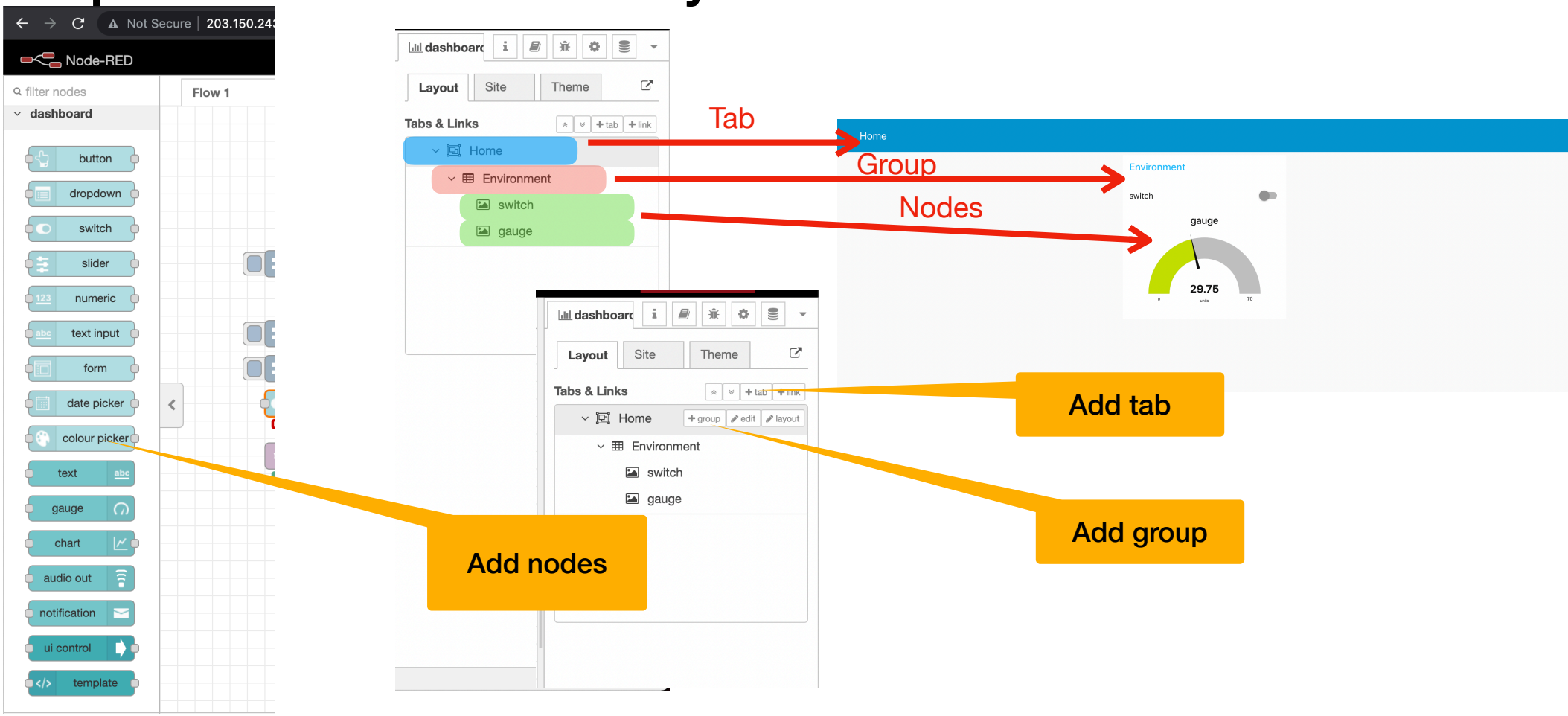
Dashboard can display data in a nice format





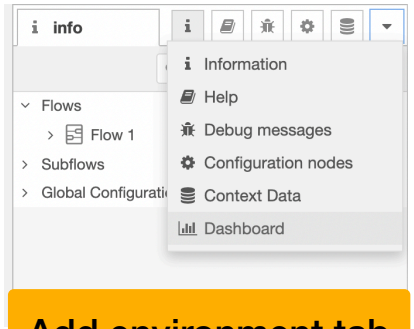
Check dashboard component

Components and dashboard layout

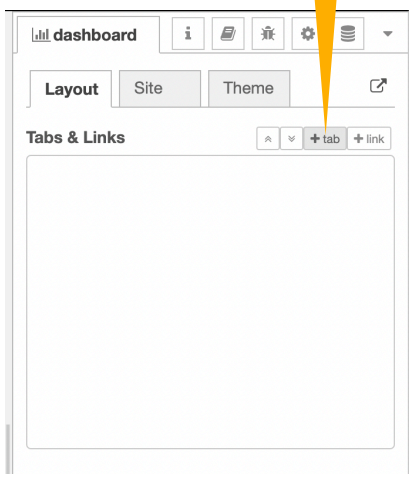


Create the first dash board

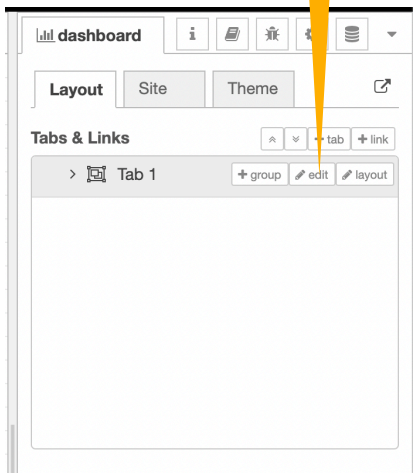
Create an Environment tab



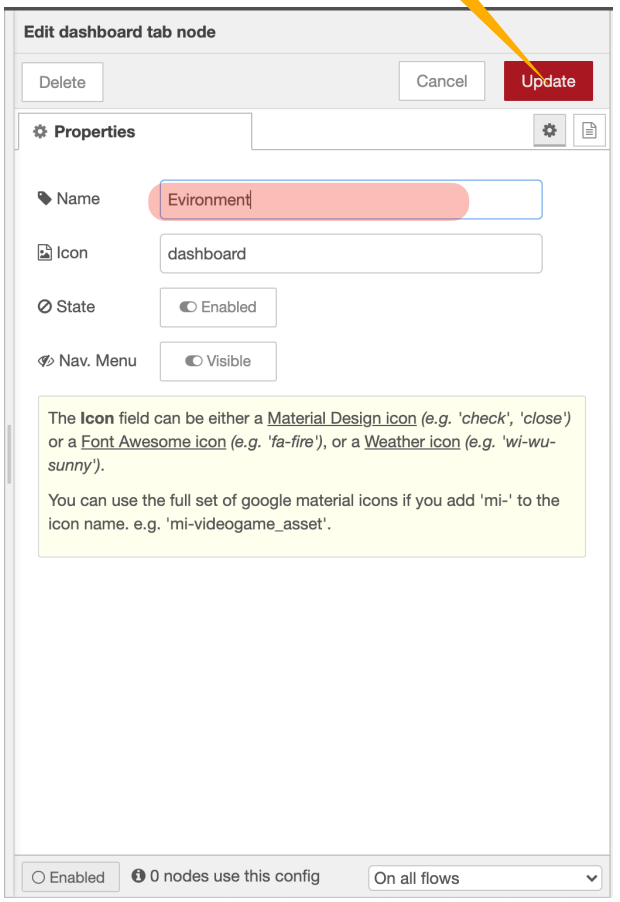
Add environment tab

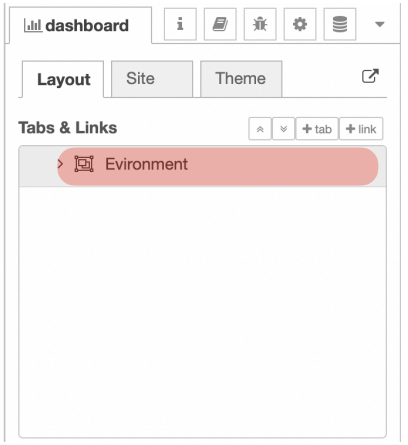


Edit environment tab



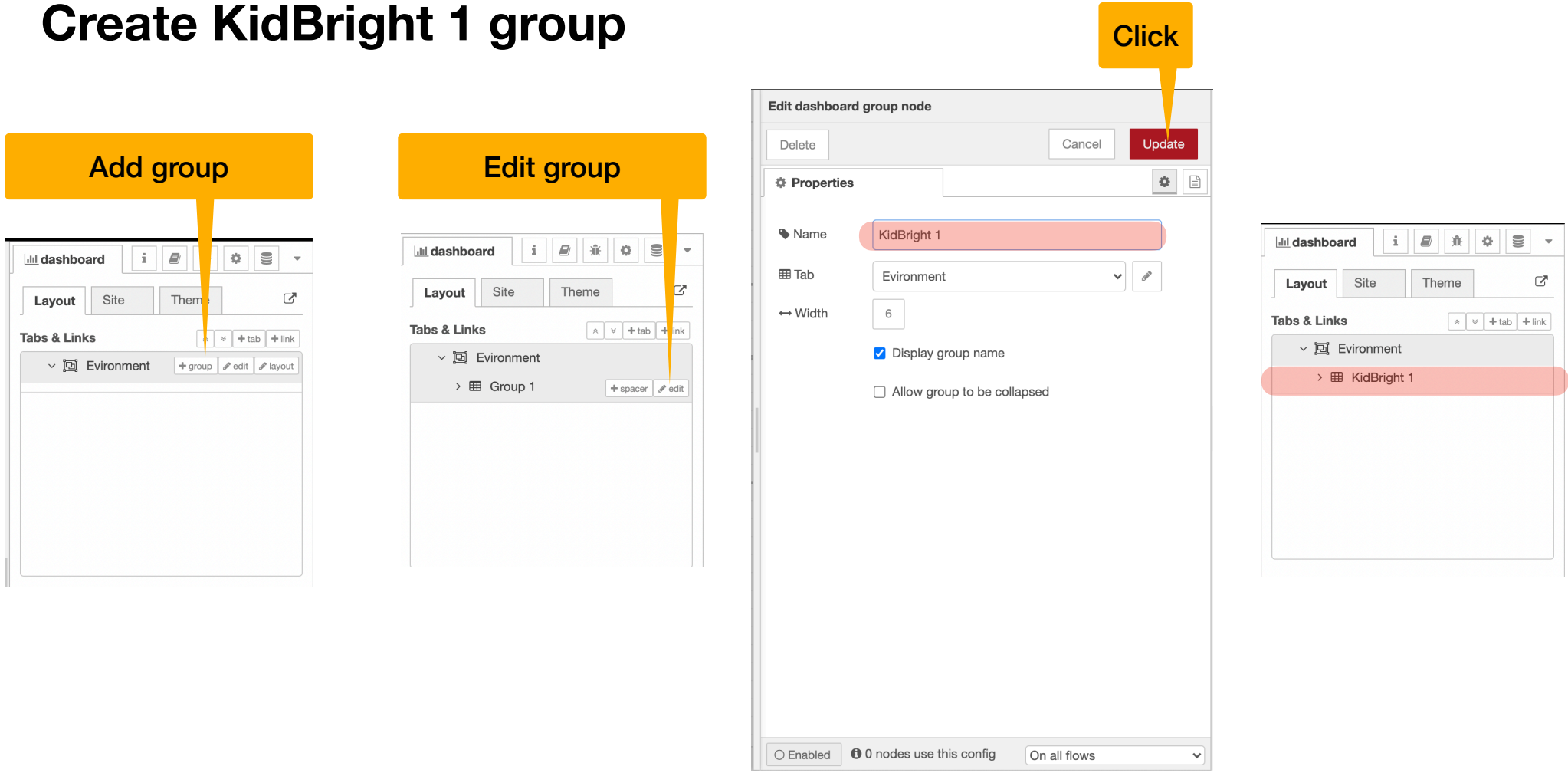
Click





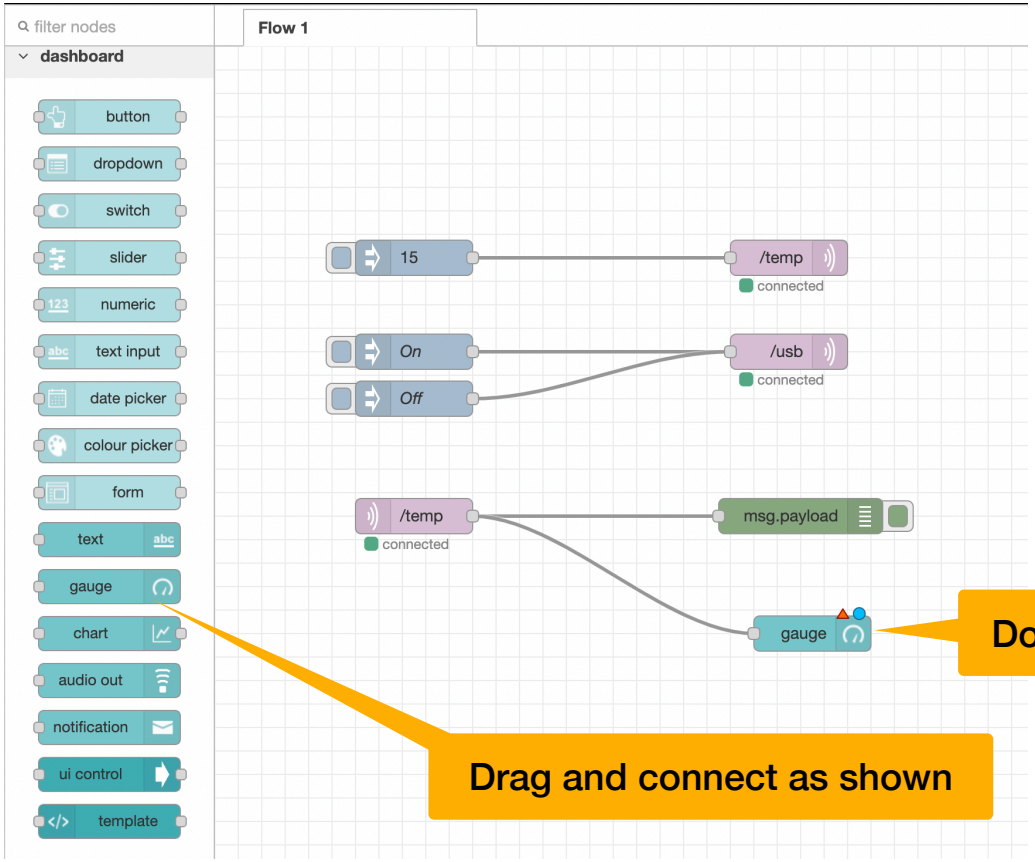
Create the first dash board

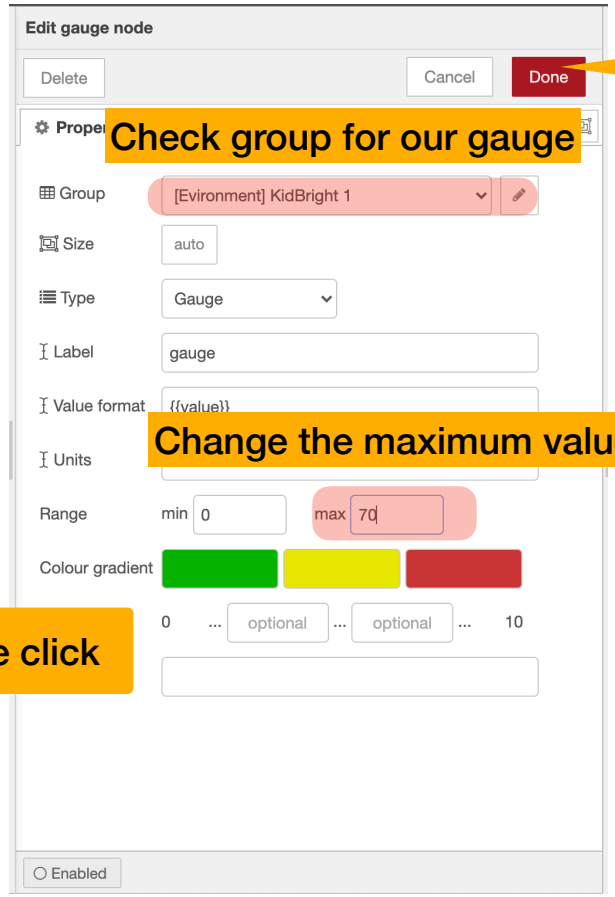
Create KidBright 1 group



Create the first dash board

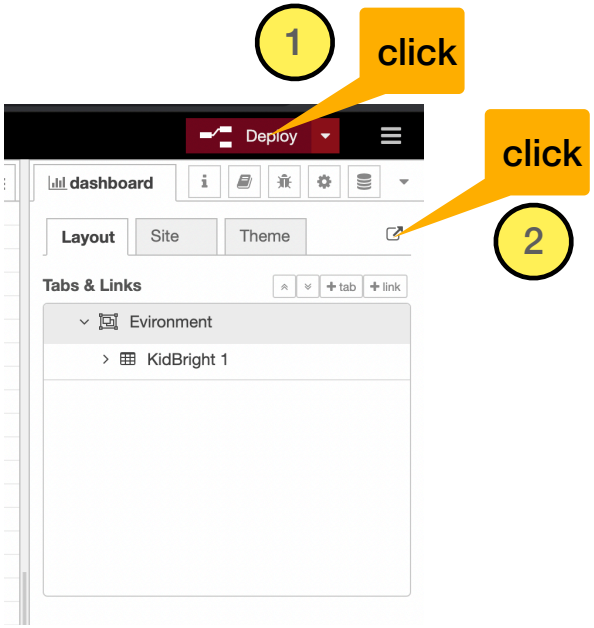
Using Gauge node to display temperature

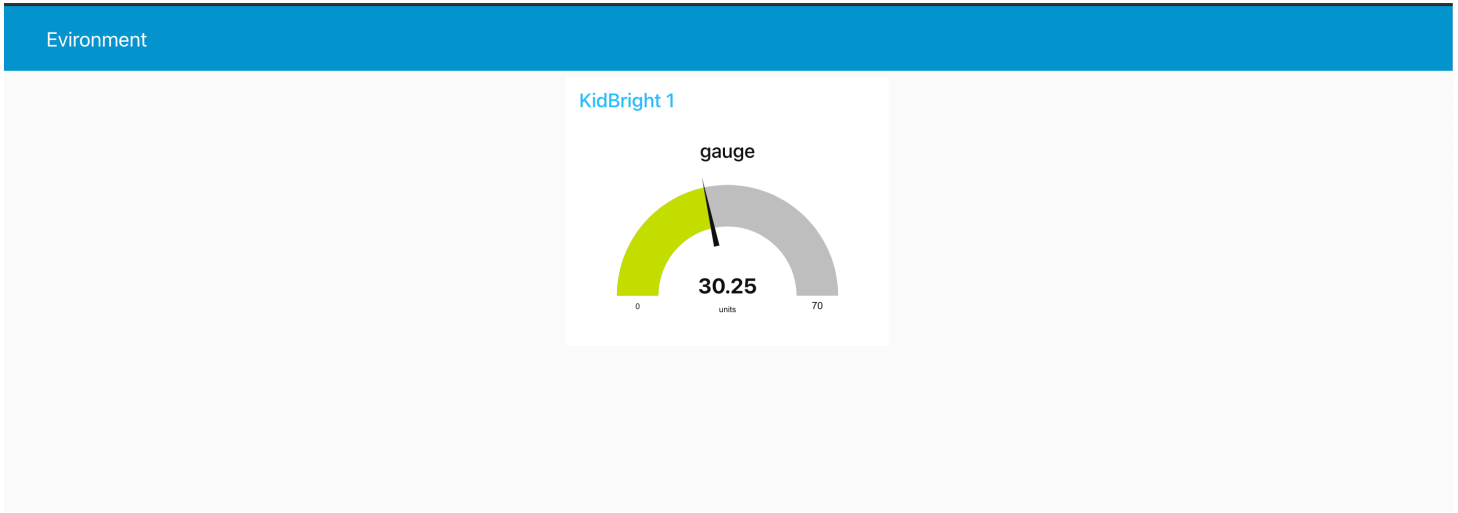




Create the first dash board

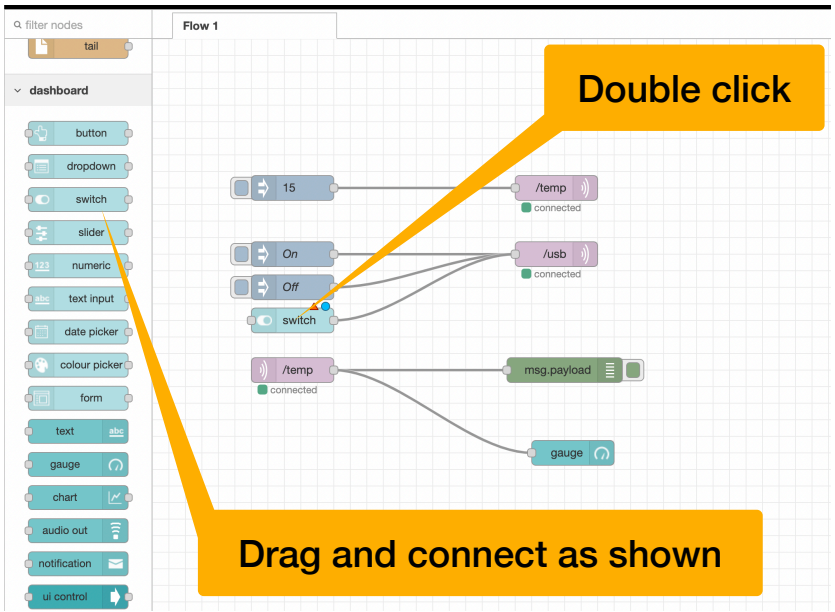
Deploy our dashboard

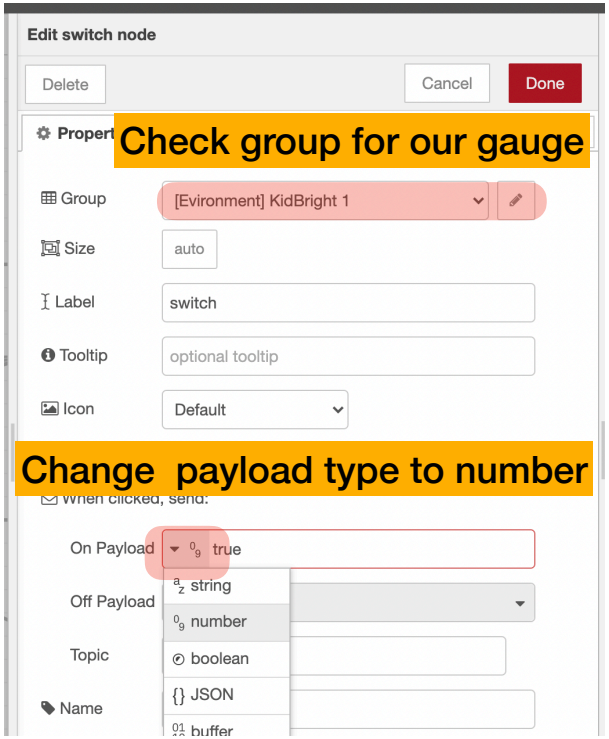


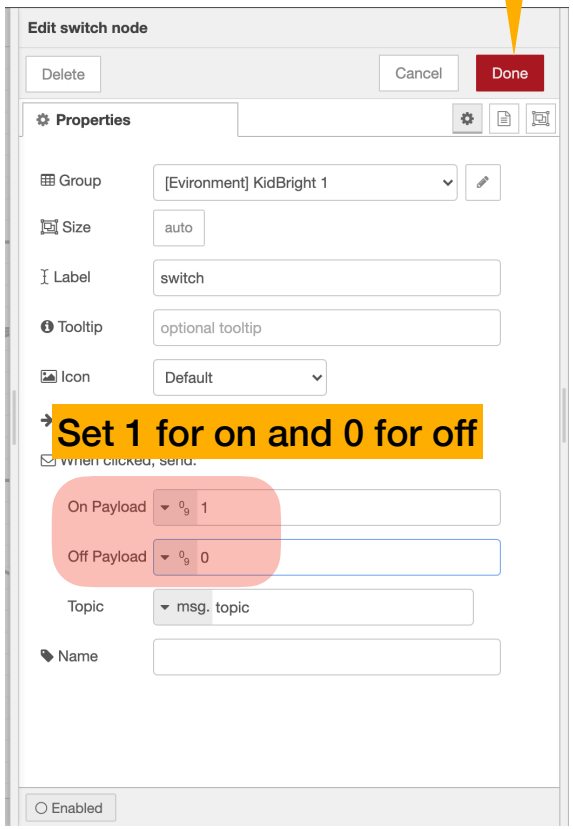


Add a switch to control USB light

Add dashboard switch

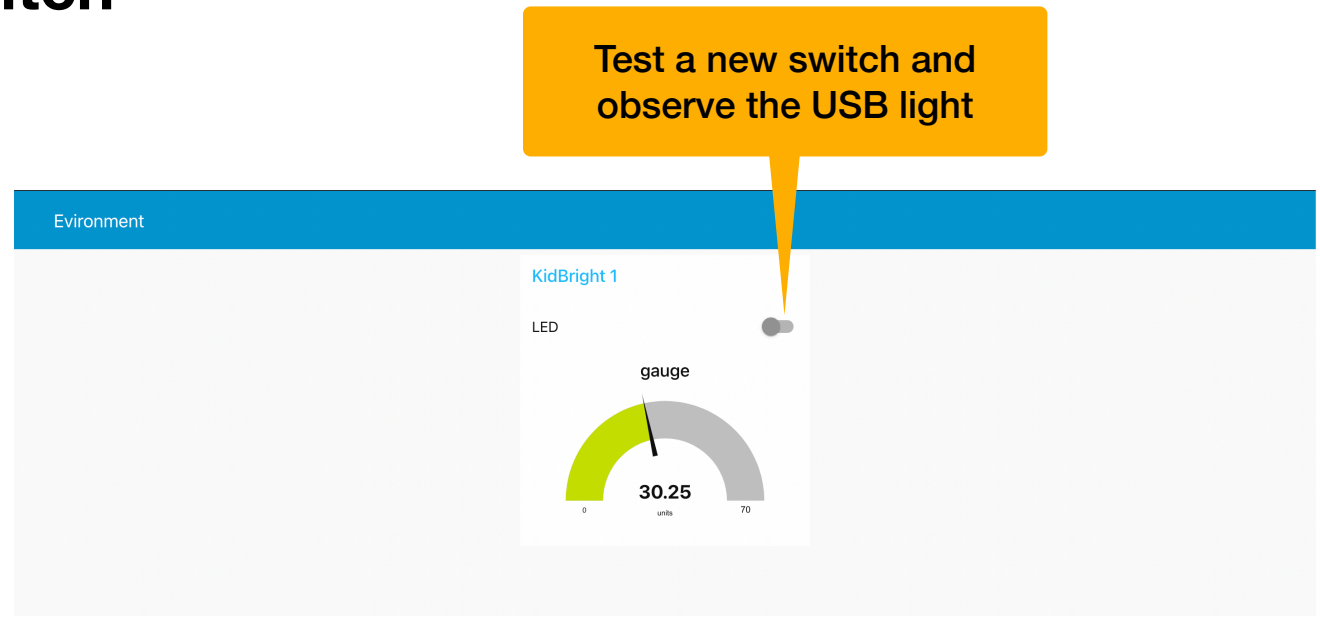






Add a switch to control USB light

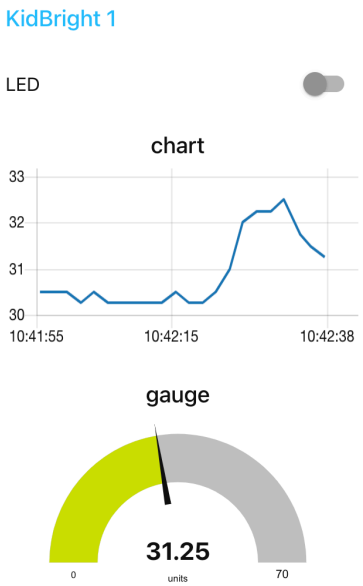
Deploy and test switch



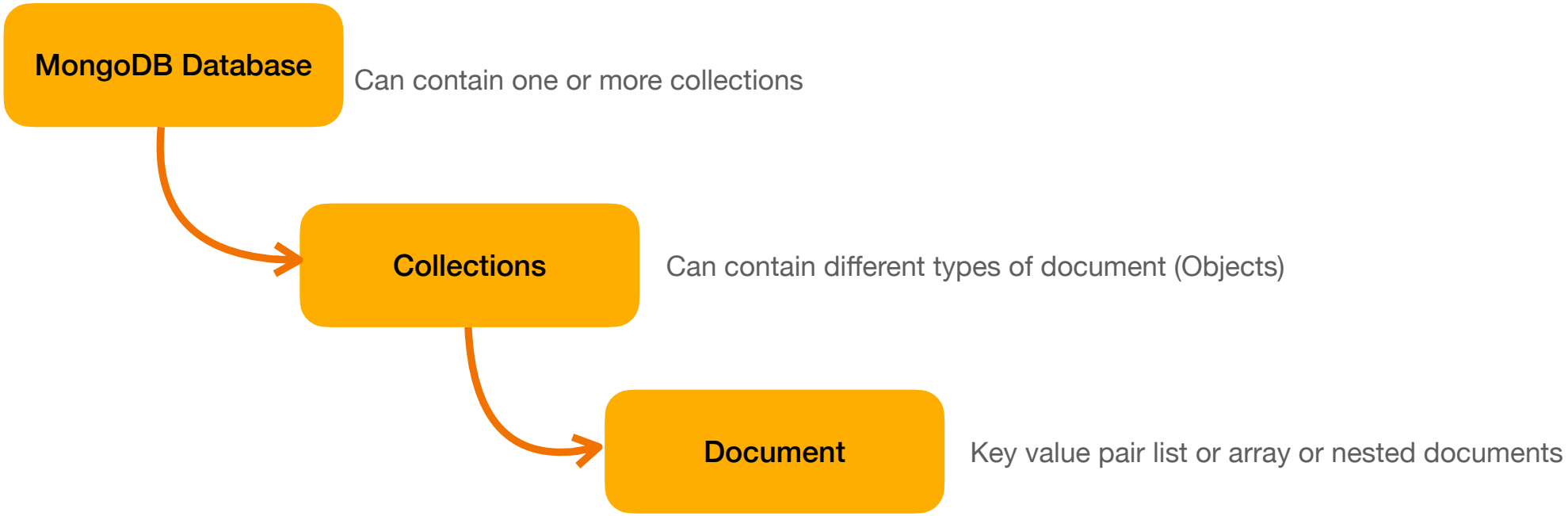
Due to bug in switch, you may need to set the value 0 and 1 many times until there is no error when deploy button is click. You can reconfig switch by double click on the switch icon in flow

Exercise

Display temperature using graph



MongoDB : NoSQL database



Install and config mongoldb

Link to command lines https://github.com/fllay/iotTraining/wiki/MongoDB-installation

```
sudo apt install -y mongodb
```

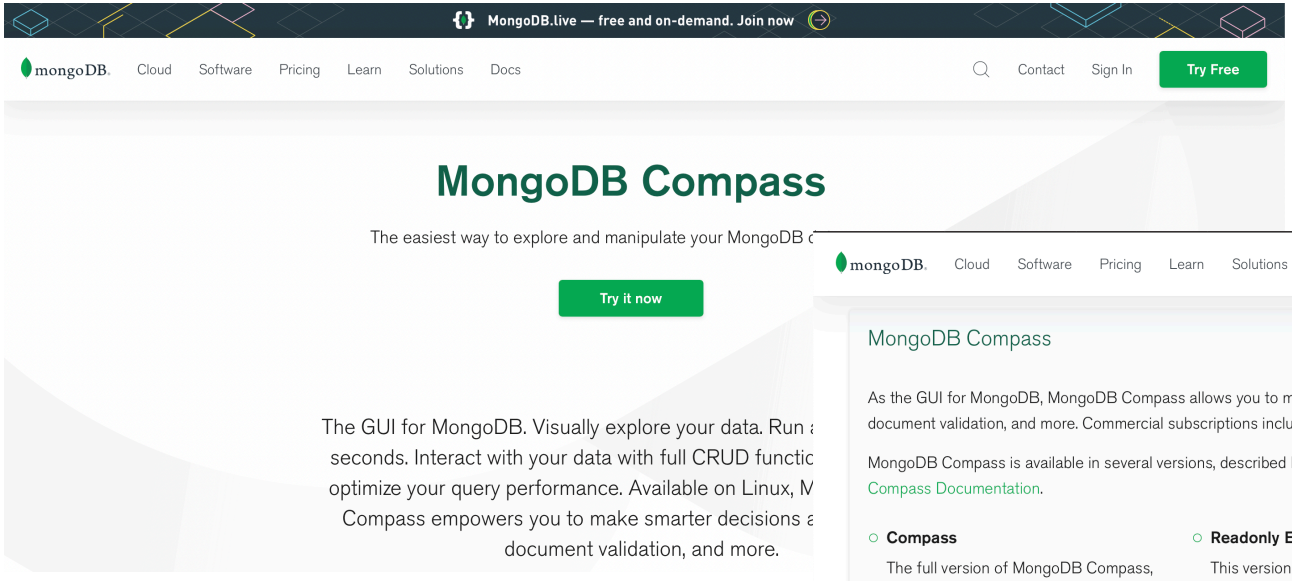
```
sudo systemctl status mongodb
```

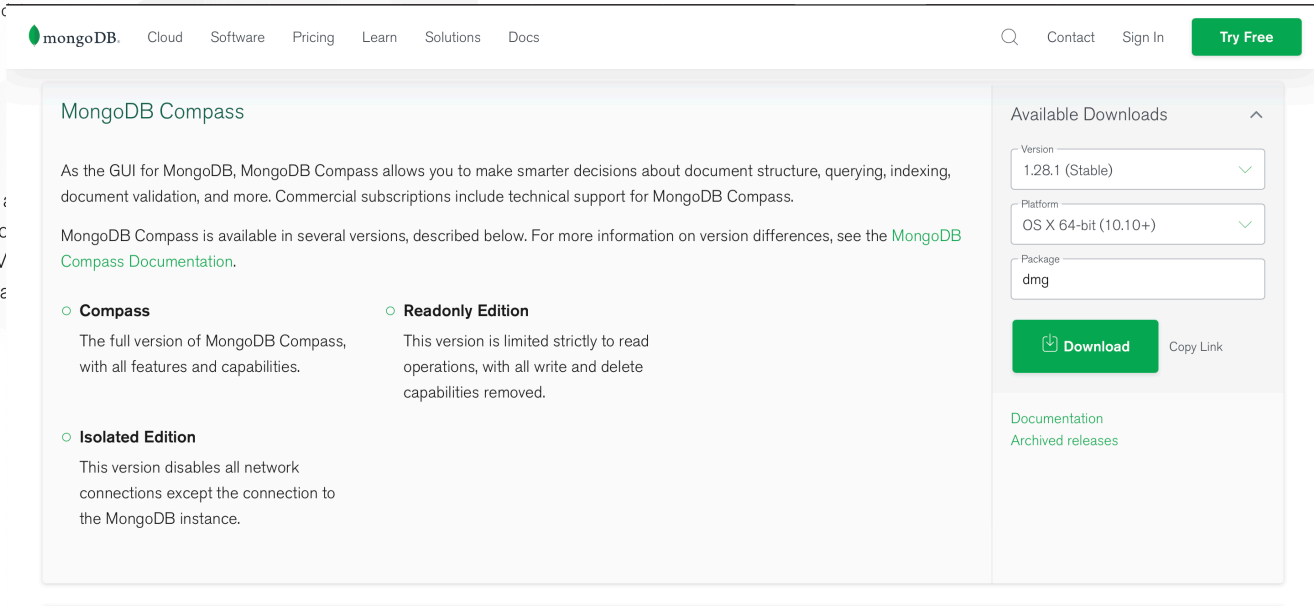
```
root@voip:~# sudo systemctl status mongodb
● mongodb.service - An object/document-oriented database
   Loaded: loaded (/lib/systemd/system/mongodb.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2021-07-16 11:30:37 +07; 31s ago
     Docs: man:mongod(1)
 Main PID: 22049 (mongod)
    Tasks: 23 (limit: 1151)
   CGroup: /system.slice/mongodb.service
           └─22049 /usr/bin/mongod --unixSocketPrefix=/run/mongodb --config /etc/mongodb.conf

Jul 16 11:30:37 voip systemd[1]: Started An object/document-oriented database.
```

```
sed "s;^[# ]*\(bind_ip * = *\)\(.*\);# \1\2\n\10.0.0.0;" "/etc/mongodb.conf"  > /etc/mongodb.conf.new

mv /etc/mongodb.conf.new  /etc/mongodb.conf

systemctl restart  mongodb
```

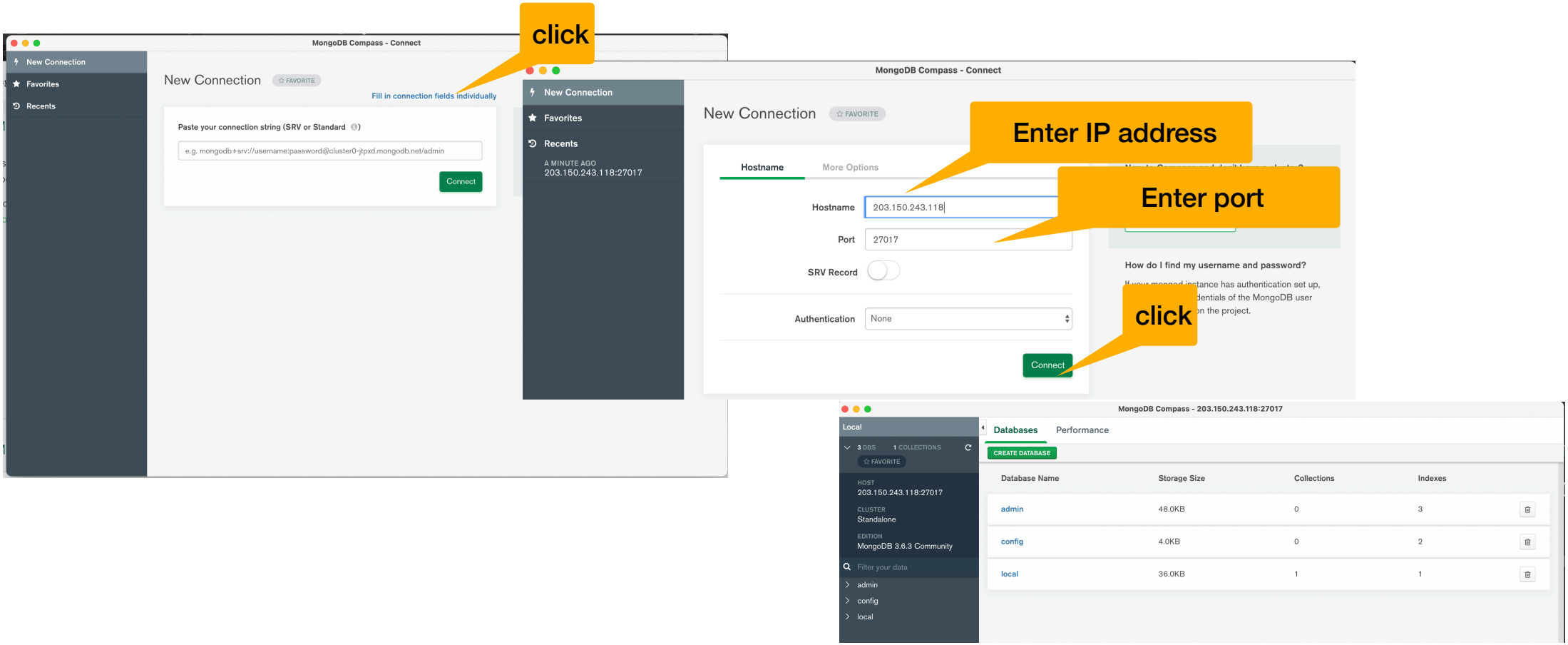
Download and install MongoDB Compass

MongoDB Compass allows database monitoring from PC

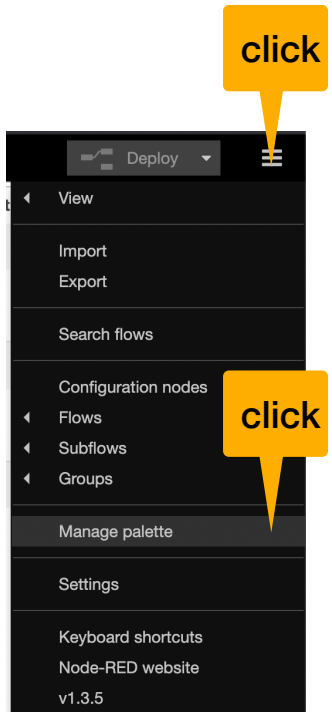


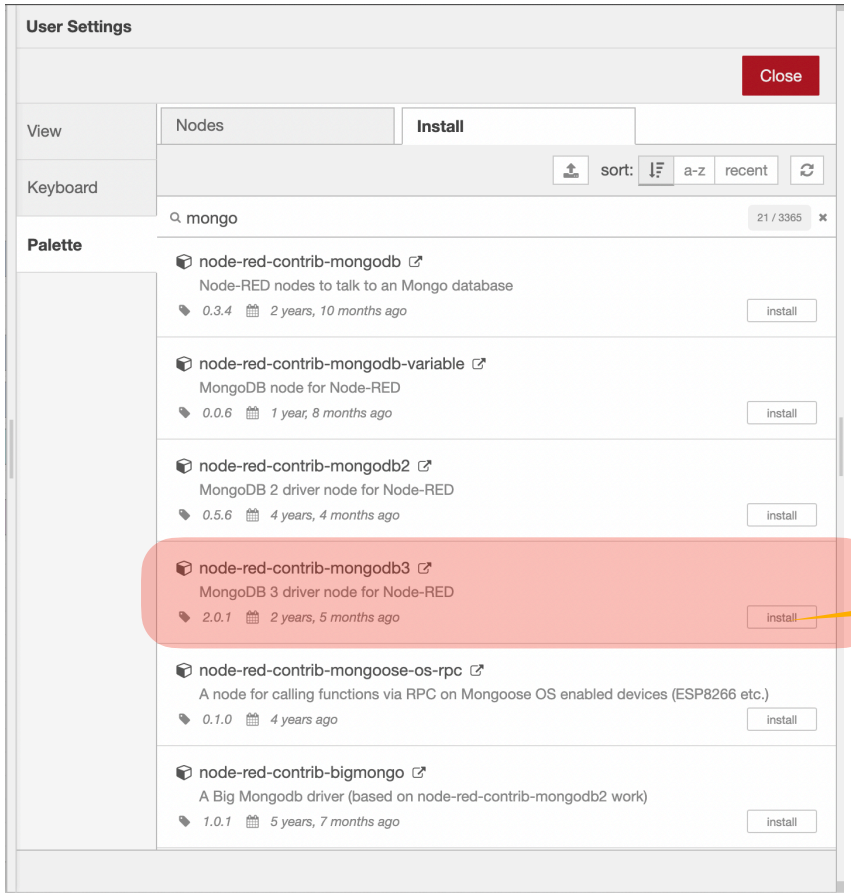


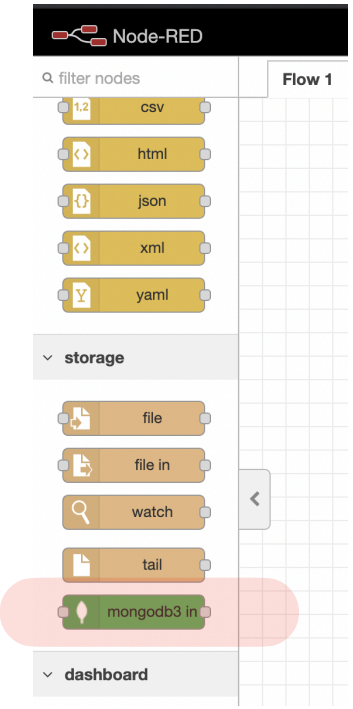
Connect MongoDB Compass to database



Install node-red MongoDB

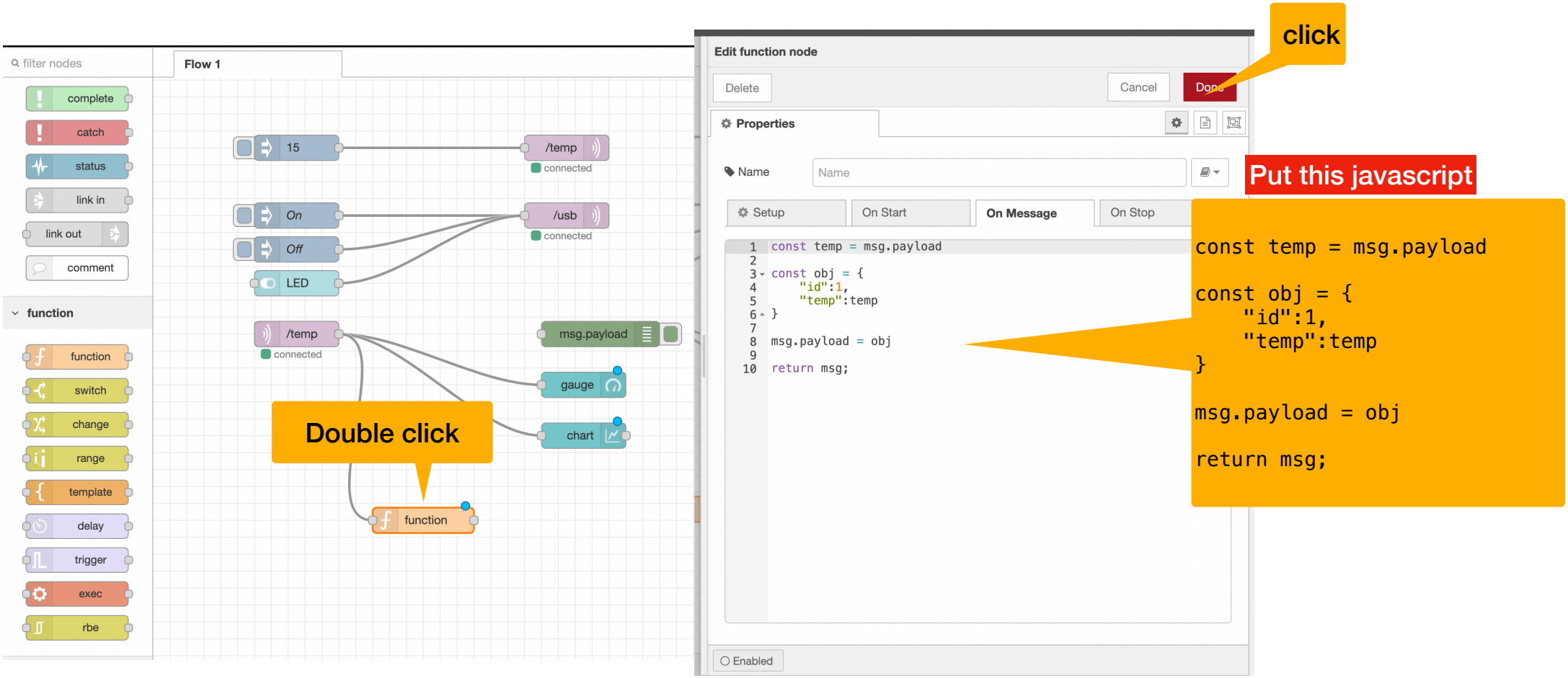






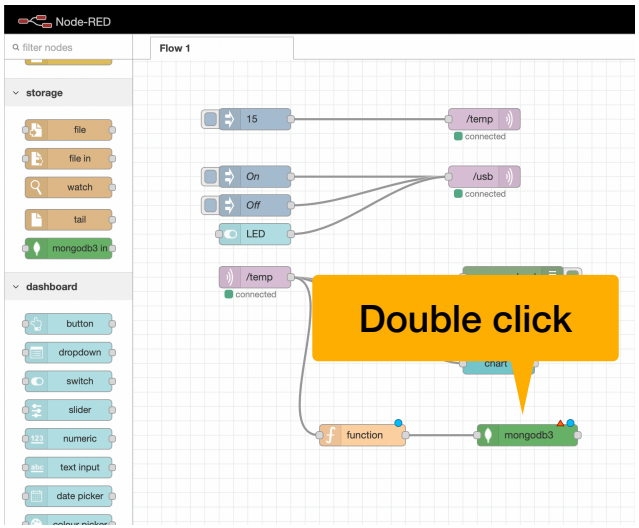
Create node-red flow

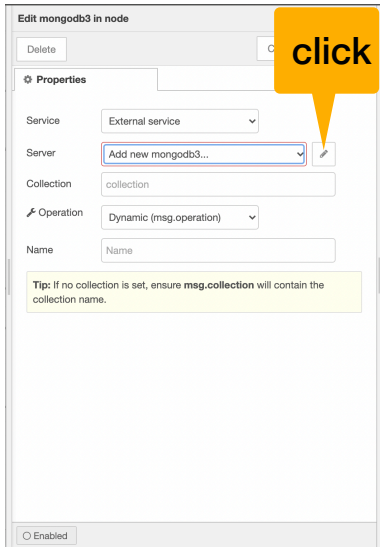
Put a function block

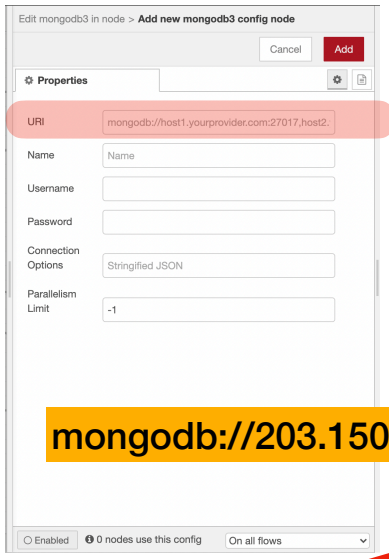


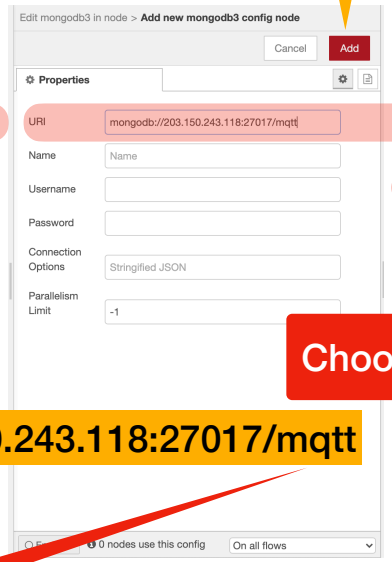
Create node-red flow

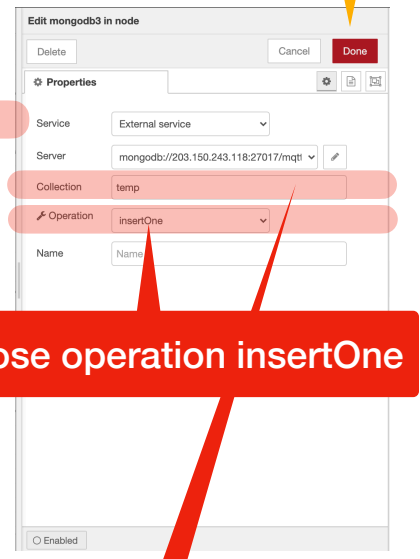
Config mongoDB block









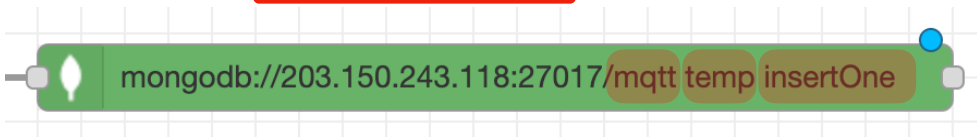


Choose operation insertOne

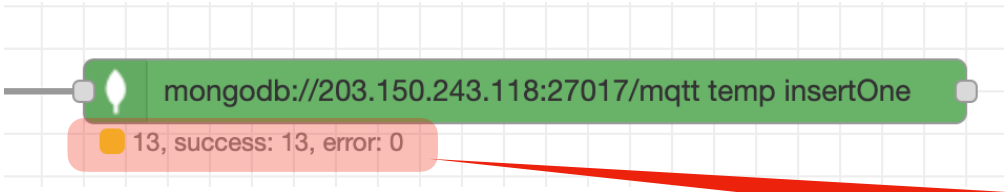
Database name

Collection name

temp

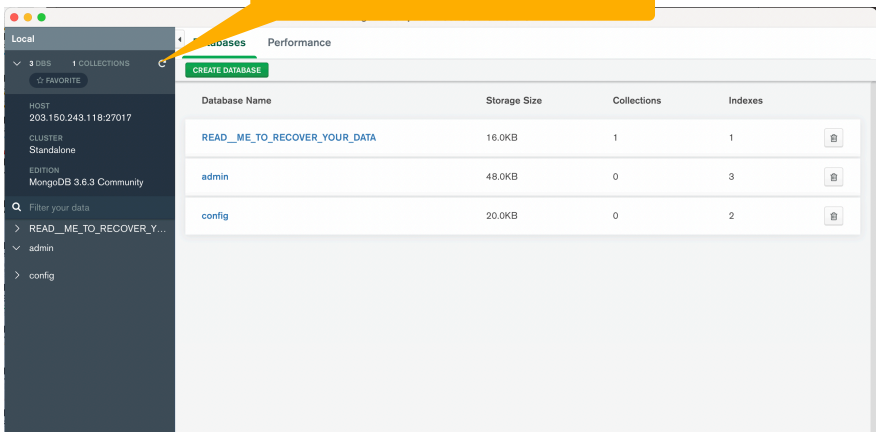


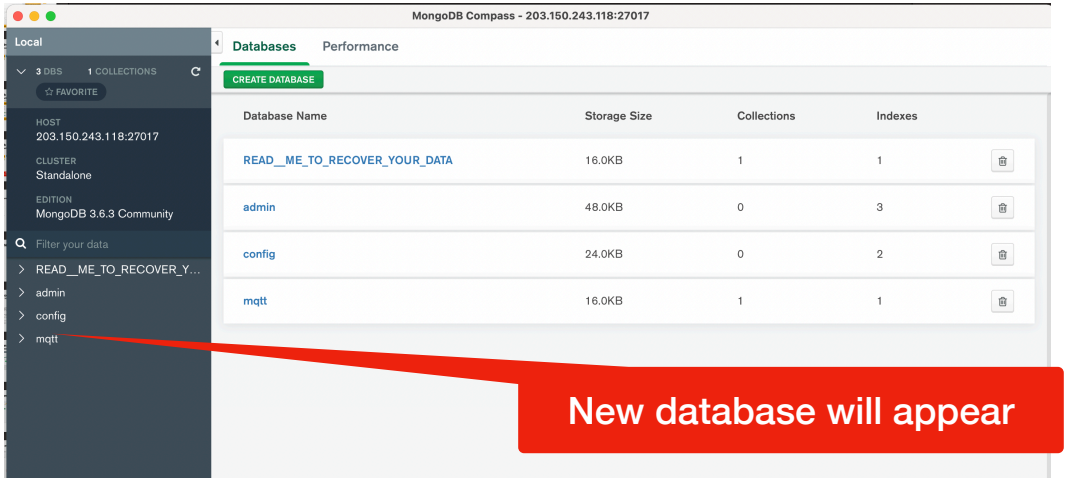
Deploy and turn KidBright on



MomgoDB node is running

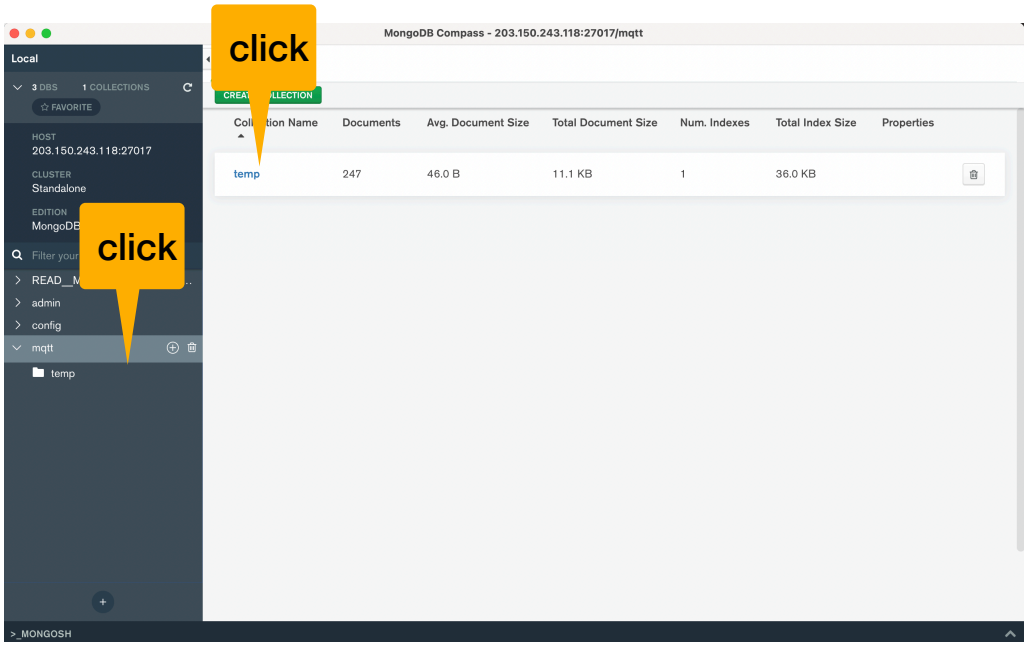
click to refresh

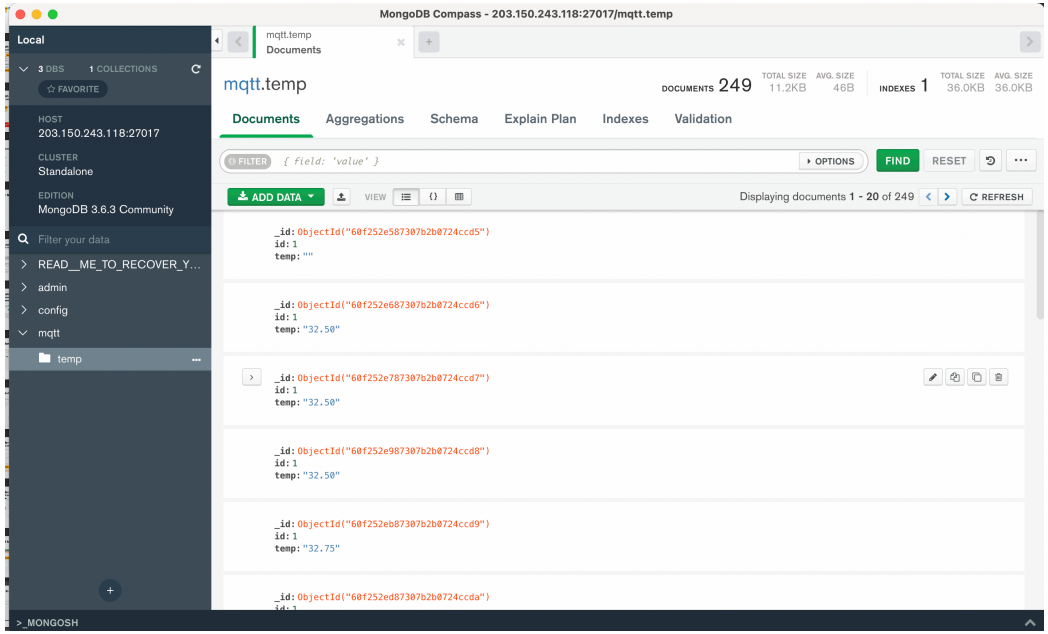




New database will appear

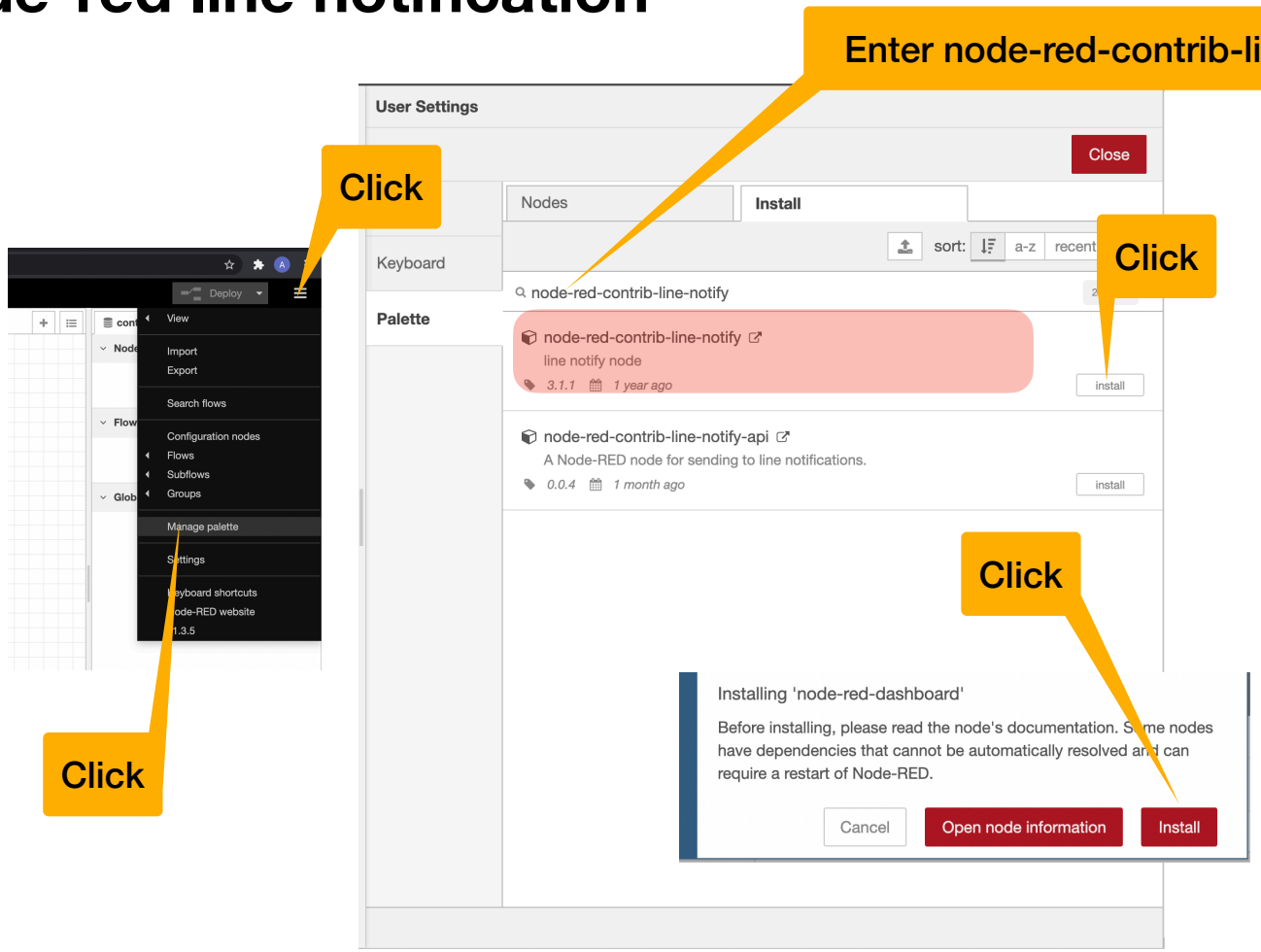
Verify data in database





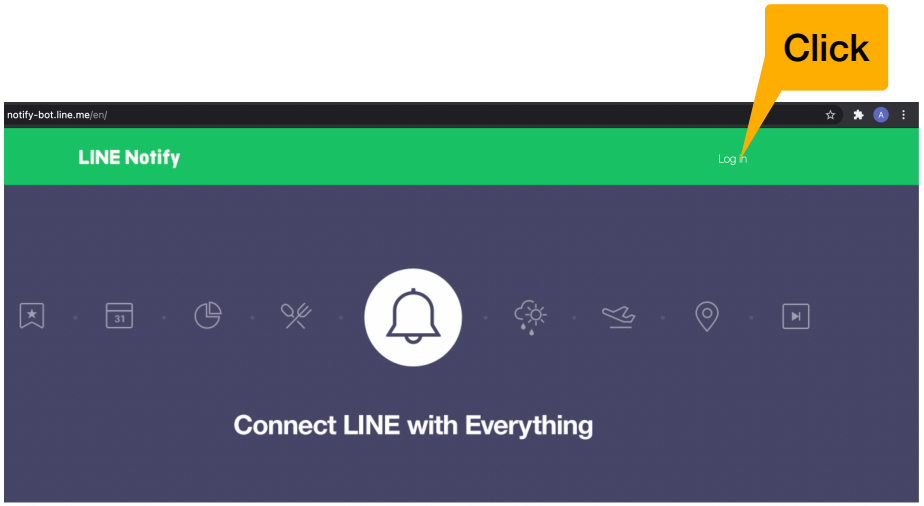
Line notify : Show notification on LINE app

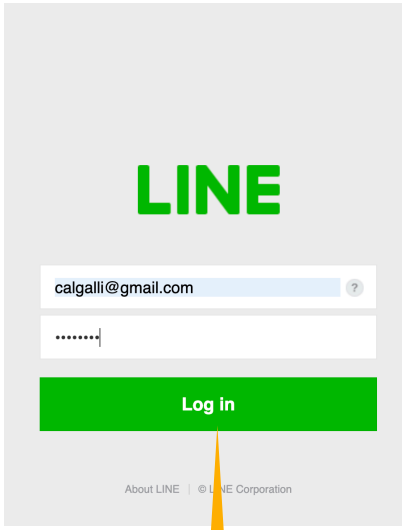
Install node-red line notification



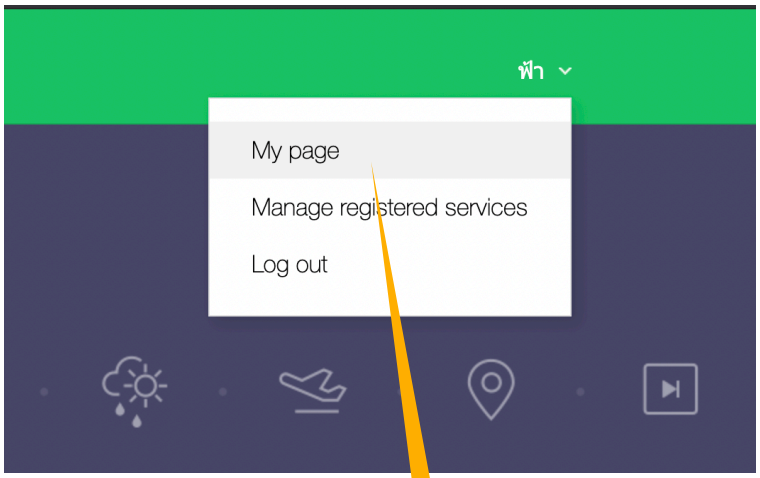
Get Line notification token

Login https://notify-bot.line.me/en/





Enter e-mail and password



Click My page

Generate token

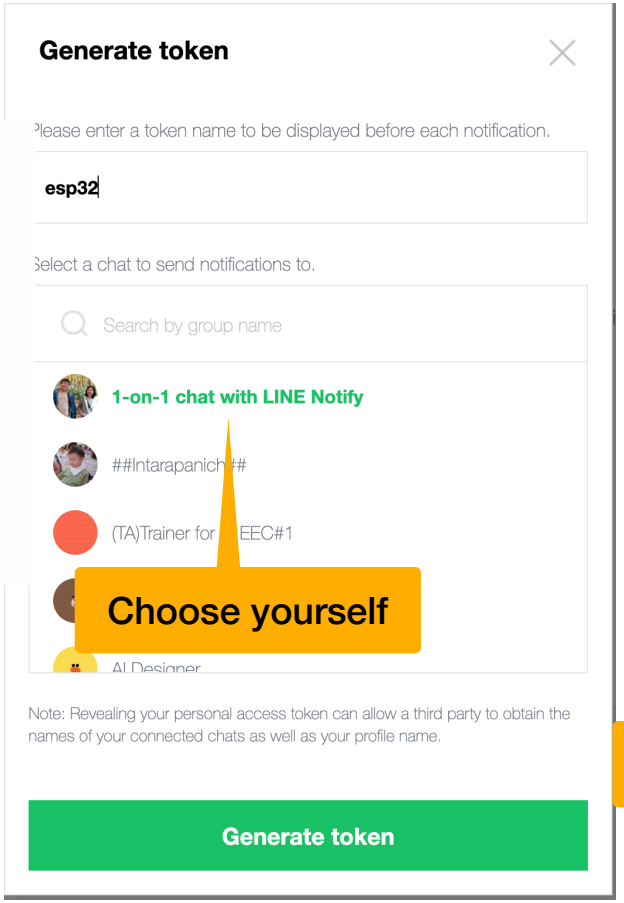
Generate access token (For developers)

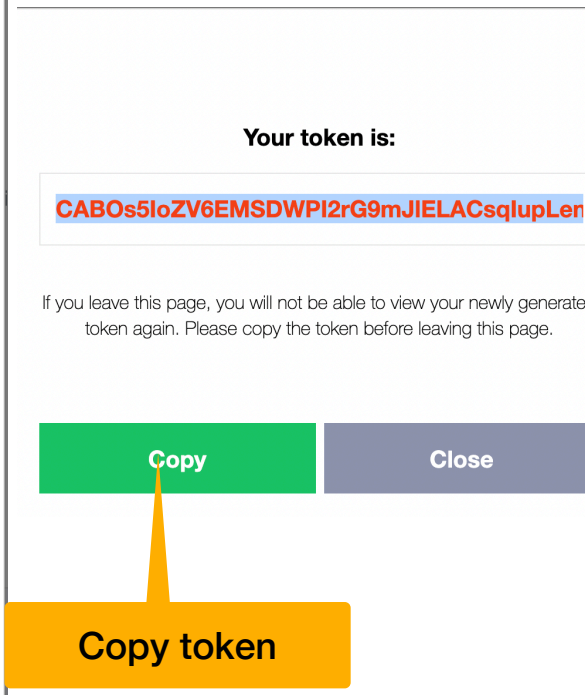
By using personal access tokens, you can configure notifications without having to add a web service.

Generate token

LINE Notify API Document

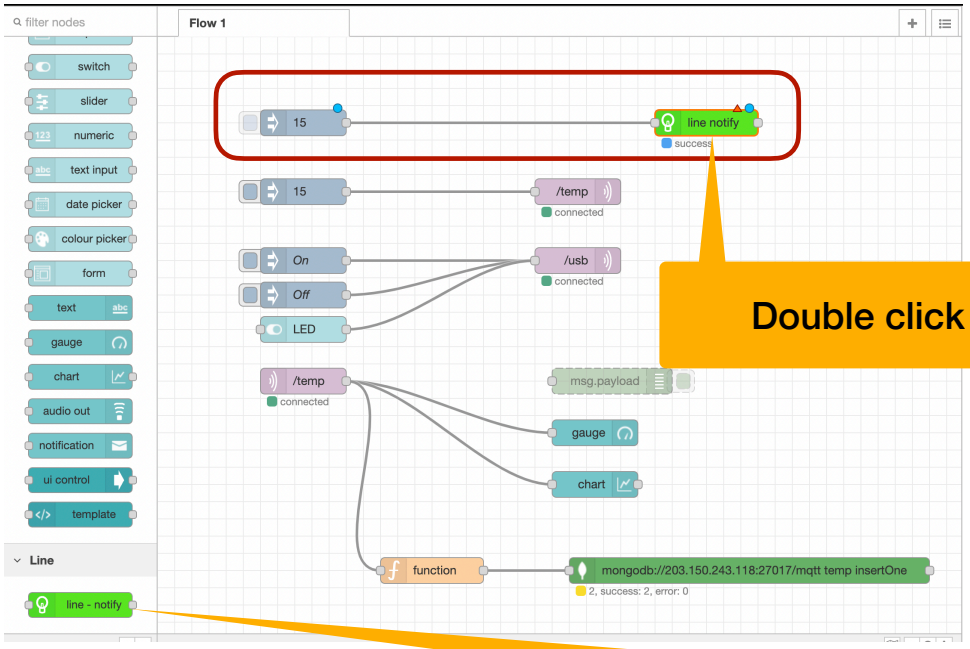
Click





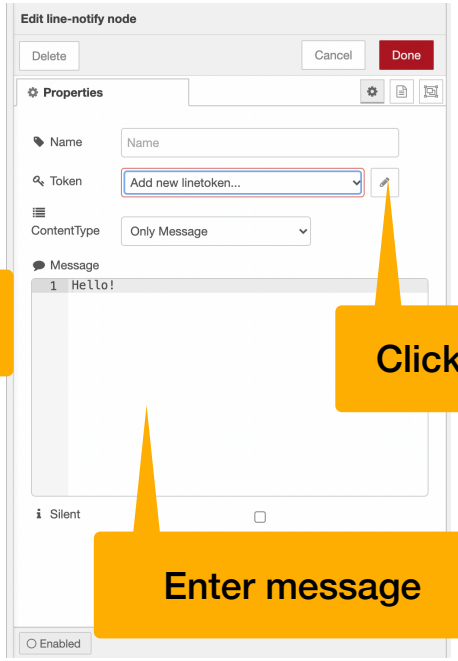
Config line notify

Enter the token and message



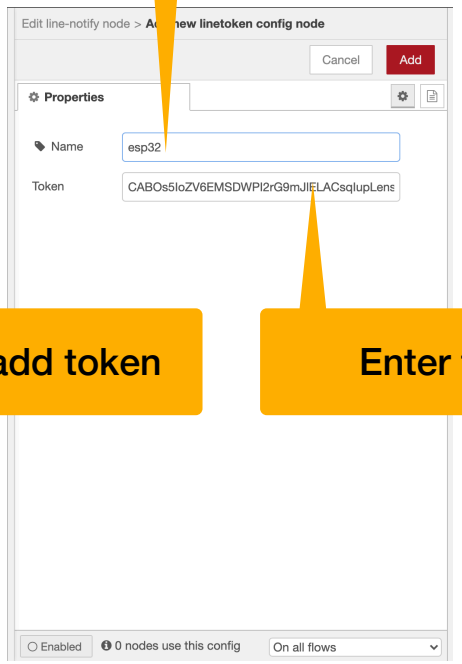
Insert line notify

Double click



Click to add token

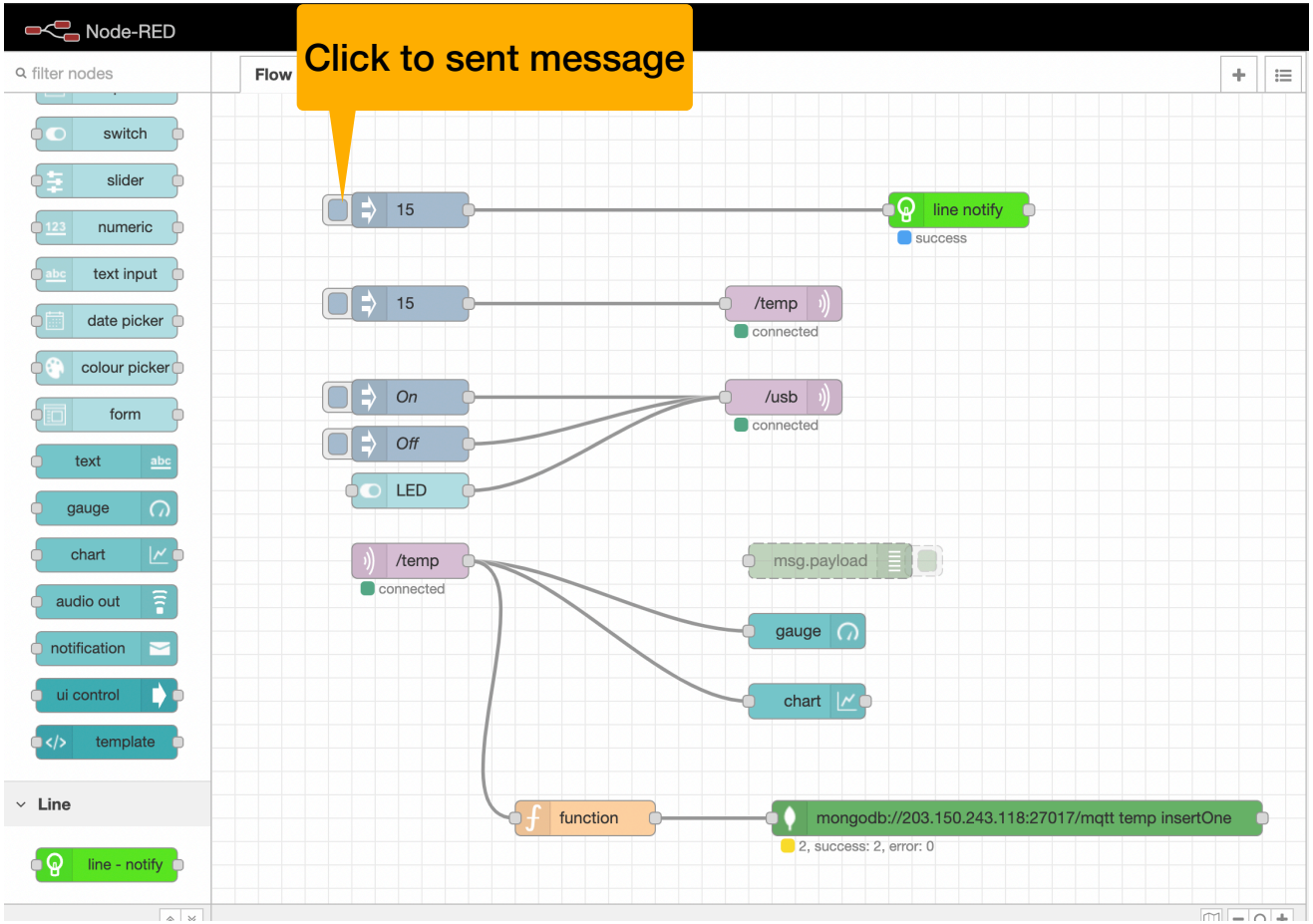
Enter message



Enter token name

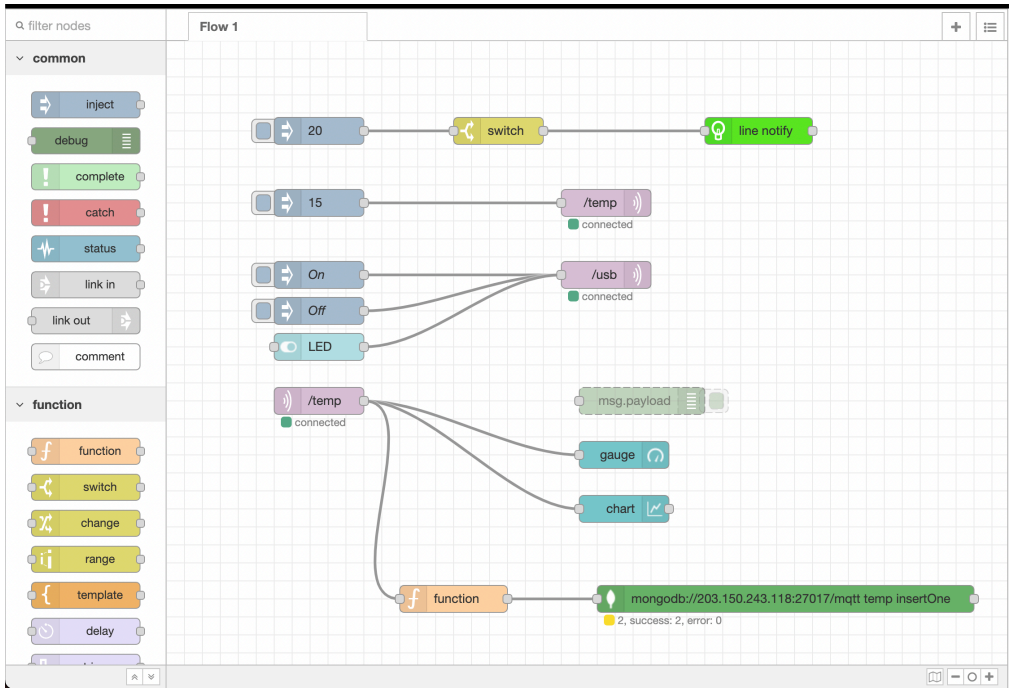
Enter token

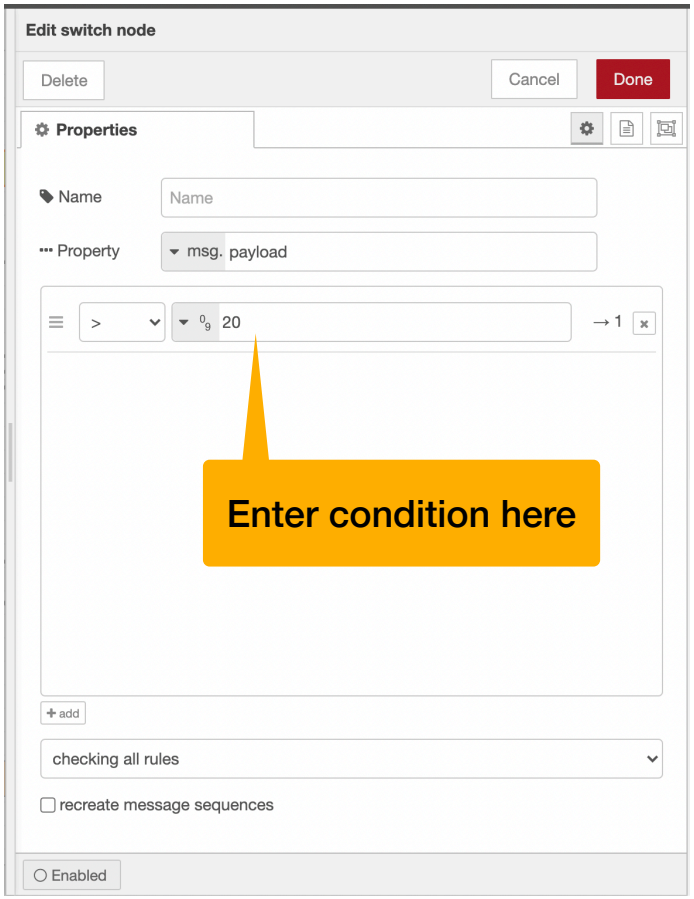
Test Line notification



Add switch to compare value

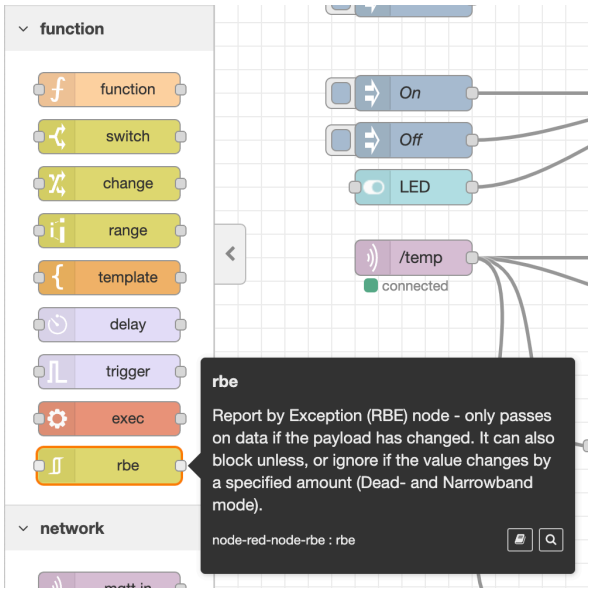
If the value is greater than 20 then sent message

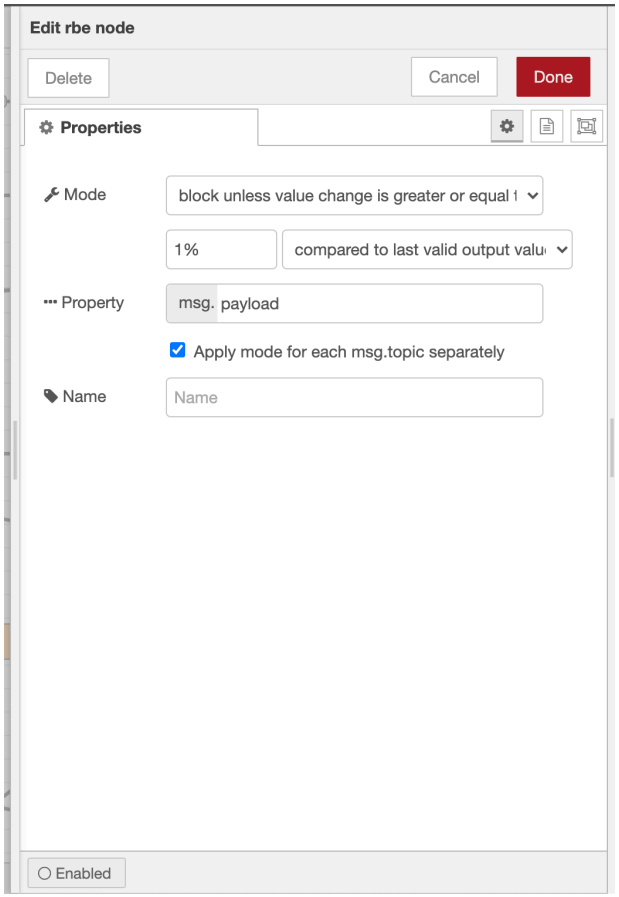




Exercise : check temperature from KidBright

Check temperature change and notify when it is higher than a threshold

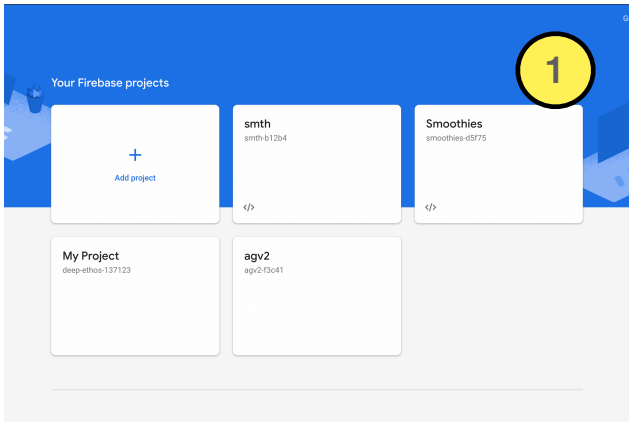


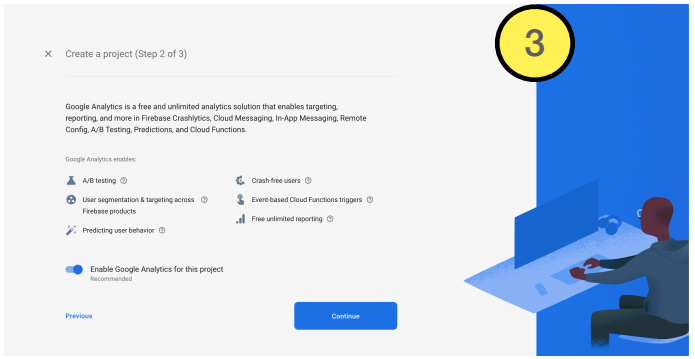


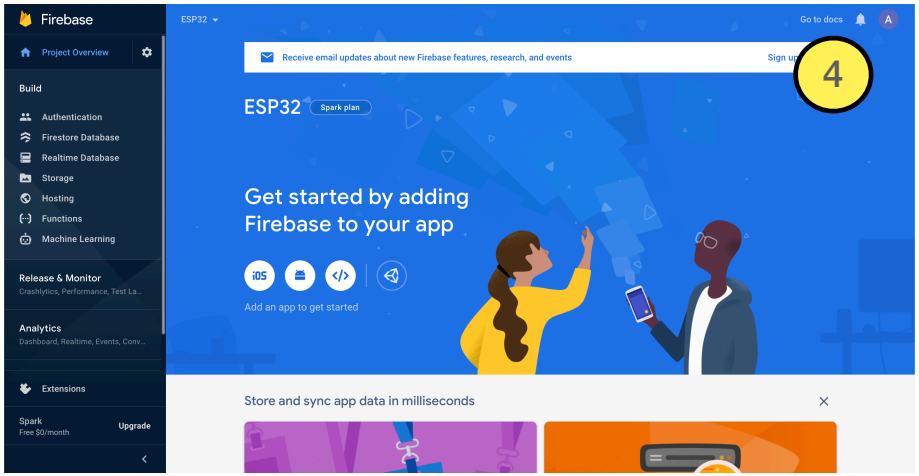
Firebase : a real time database

Create a Firebase project

Go to https://console.firebase.google.com/

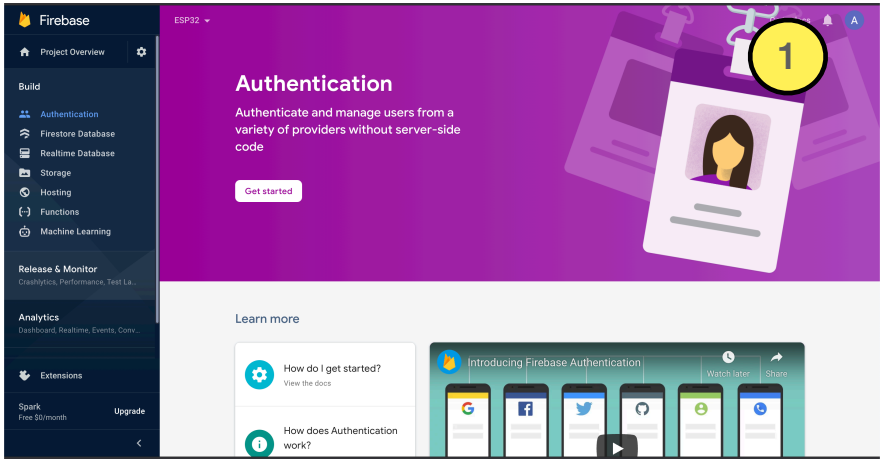


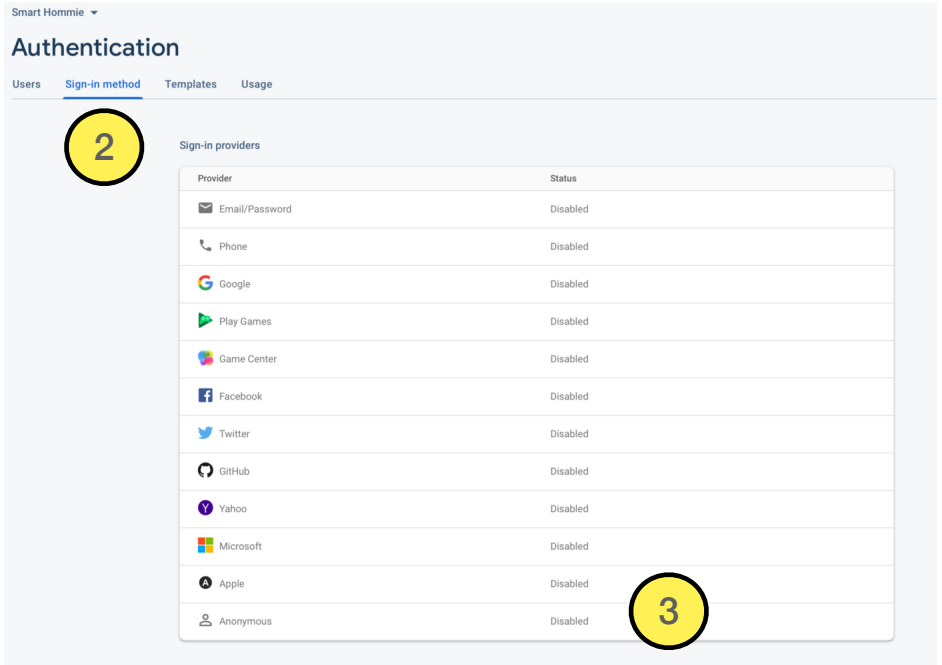


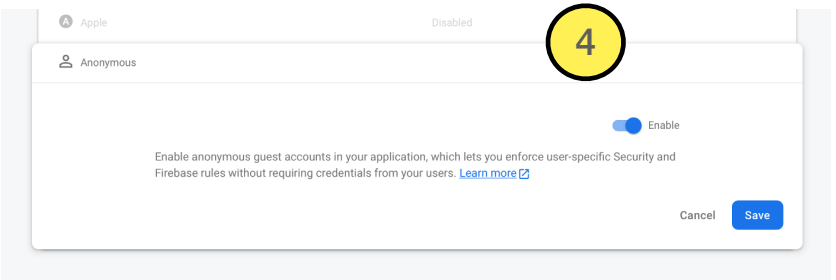


Set up Authentication

Enable *Anonymous Sign In*

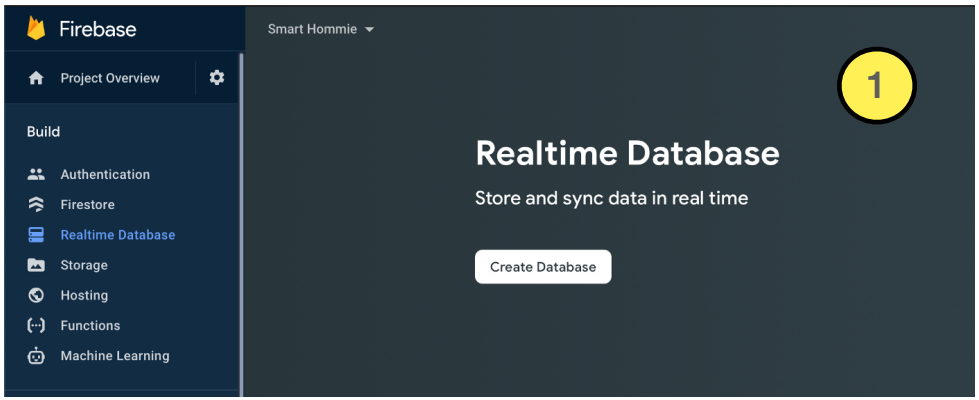


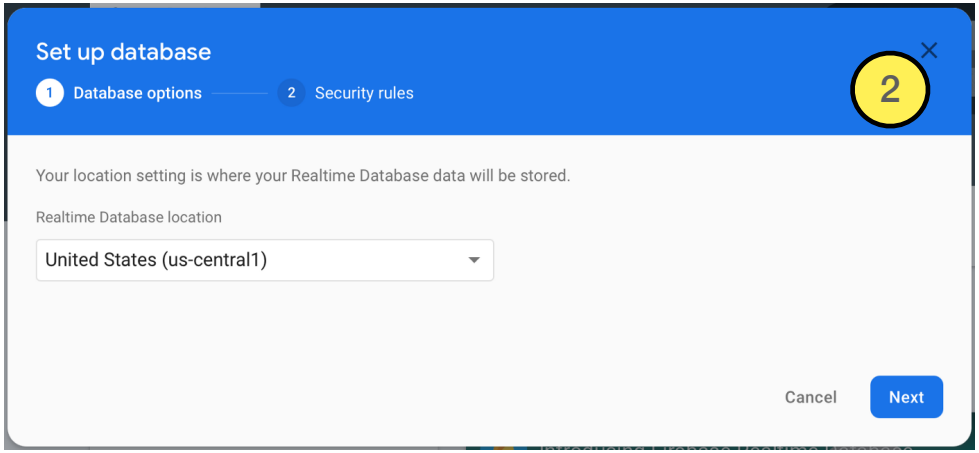


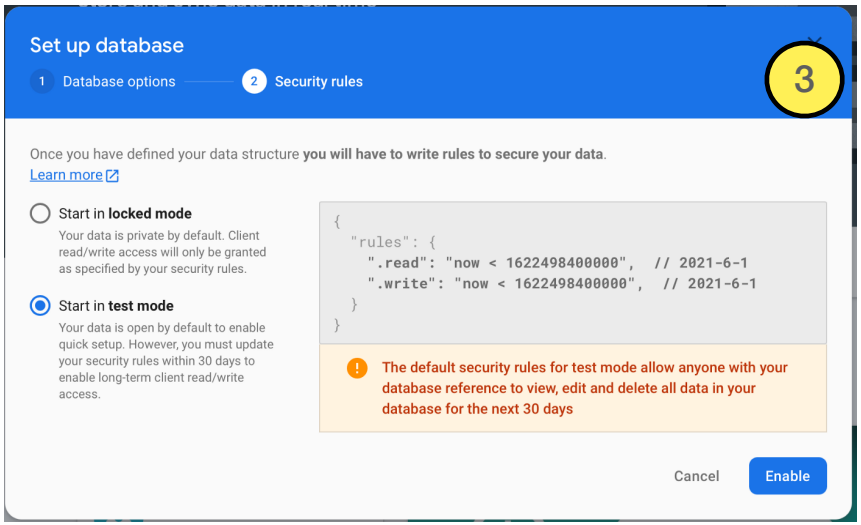


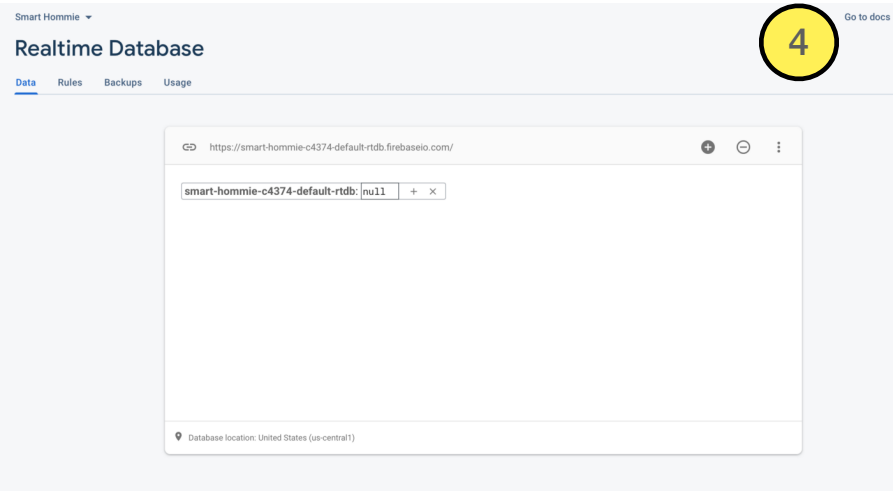
Create a database

create a database for our data



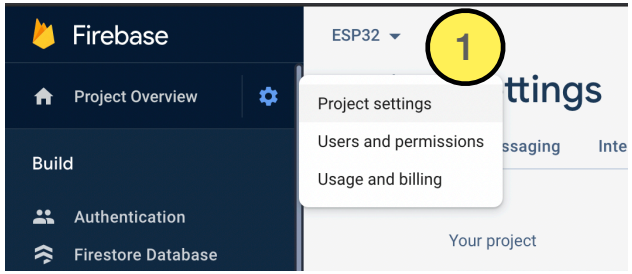


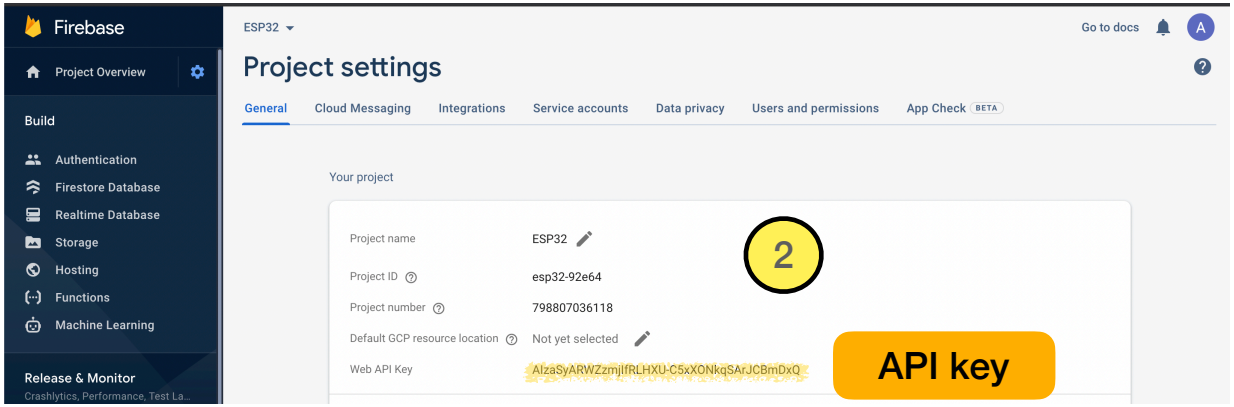


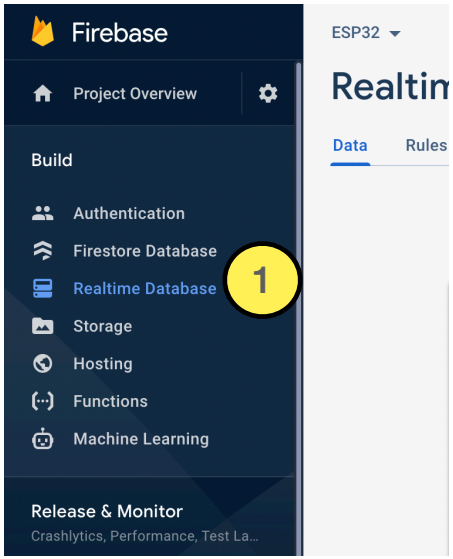


Get API Key and database URL for ESP32

API key and database URL will be used for ESP32 authentication







Firebase ESP32 publish data

ESP32 sets values in database

Define authentication API_KEY and DATABASE_URL

```
// Your Firebase Project Web API Key
#define API_KEY "AIzaSyARWZzmjIfRLHXU-C5xXONkqSArJCBmDxQ"
// Your Firebase Realtime database URL
#define DATABASE_URL "https://esp32-92e64-default-rtdb.asia-southeast1.firebasedatabase.app"
```

```
Serial.printf("Set int... %s\n", Firebase.RTDB.setFloat(&fbdo, "/sensor/temp", readTemperature()) ? "ok" :fbdo.errorReason().c_str());
```

ESP32 the first Firebase program

`fisrFirebase.ino`

```
// Your Firebase Project Web API Key
#define API_KEY "AIzaSyCG59xubOTJe2gMqg1hz2NmcLvY_0AK-_M"
// Your Firebase Realtime database URL
#define DATABASE_URL "https://esp32-b74d2-default-rtdb.asia-southeast1.firebasedatabase.app/"

const char *ssid = "CAKE";
const char *password = "0891300633";
```

Set authentication

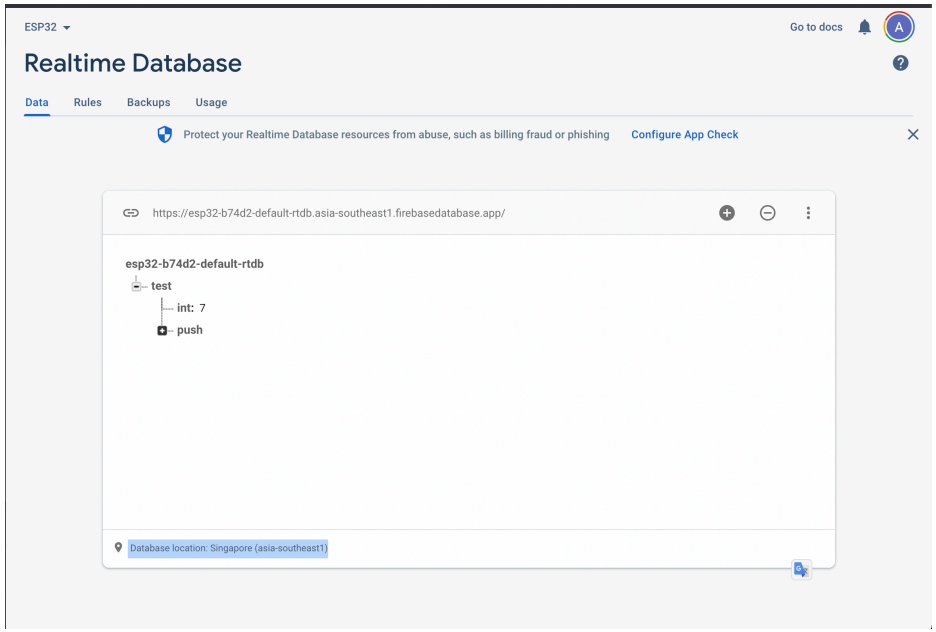
```
void firebase_init() {
  // configure firebase API Key
  config.api_key = API_KEY;
  // configure firebase realtime database url
  config.database_url = DATABASE_URL;
  // Enable WiFi reconnection
  Firebase.reconnectWiFi(true);
  Serial.println("------------------------------------");
  Serial.println("Sign up new user...");
  // Sign in to firebase Anonymously
  if (Firebase.signUp(&config, &auth, "", "")) {
    Serial.println("Success");
    isAuthenticated = true;
    // Set the database path where updates will be loaded for this device
    databasePath = "/";
    fuid = auth.token.uid.c_str();
  } else {
    Serial.printf("Failed, %s\n", config.signer.signupError.message.c_str());
    isAuthenticated = false;
  }
  // Assign the callback function for the long running token generation task,
  // see addons/TokenHelper.h
  config.token_status_callback = tokenStatusCallback;
  // Initialise the firebase library
  Firebase.begin(&config, &auth);
}
```

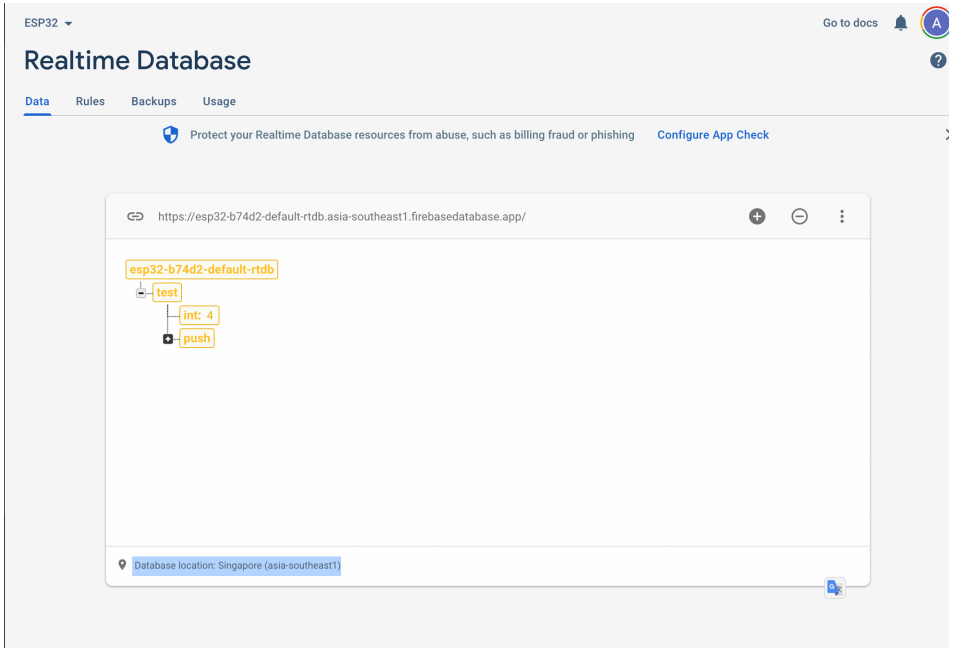
Connect to firebase

```
void loop() {
  if (Firebase.ready() && (millis() - sendDataPrevMillis > 15000 || sendDataPrevMillis == 0))
  {
    sendDataPrevMillis = millis();

    Serial.printf("Set int... %s\n", Firebase.RTDB.setInt(&fbdo, "/test/int", count) ? "ok" : fbdo.errorReason().c_str());

    Serial.printf("Get int... %s\n", Firebase.RTDB.getInt(&fbdo, "/test/int") ? String(fbdo.intData()).c_str() : fbdo.errorReason().c_str());

    FirebaseJson json;
    json.add("value", count);

    Serial.printf("Push json... %s\n", Firebase.RTDB.pushJSON(&fbdo, "/test/push", &json) ? "ok" : fbdo.errorReason().c_str());

    json.set("value", count + 100);
    Serial.printf("Update json... %s\n\n", Firebase.RTDB.updateNode(&fbdo, String("/test/push/" + fbdo.pushName()).c_str(), &json) ? "ok" : fbdo.errorReason().c_str());

    count++;
  }
}
```

Send data to firebase

Observe data in the database

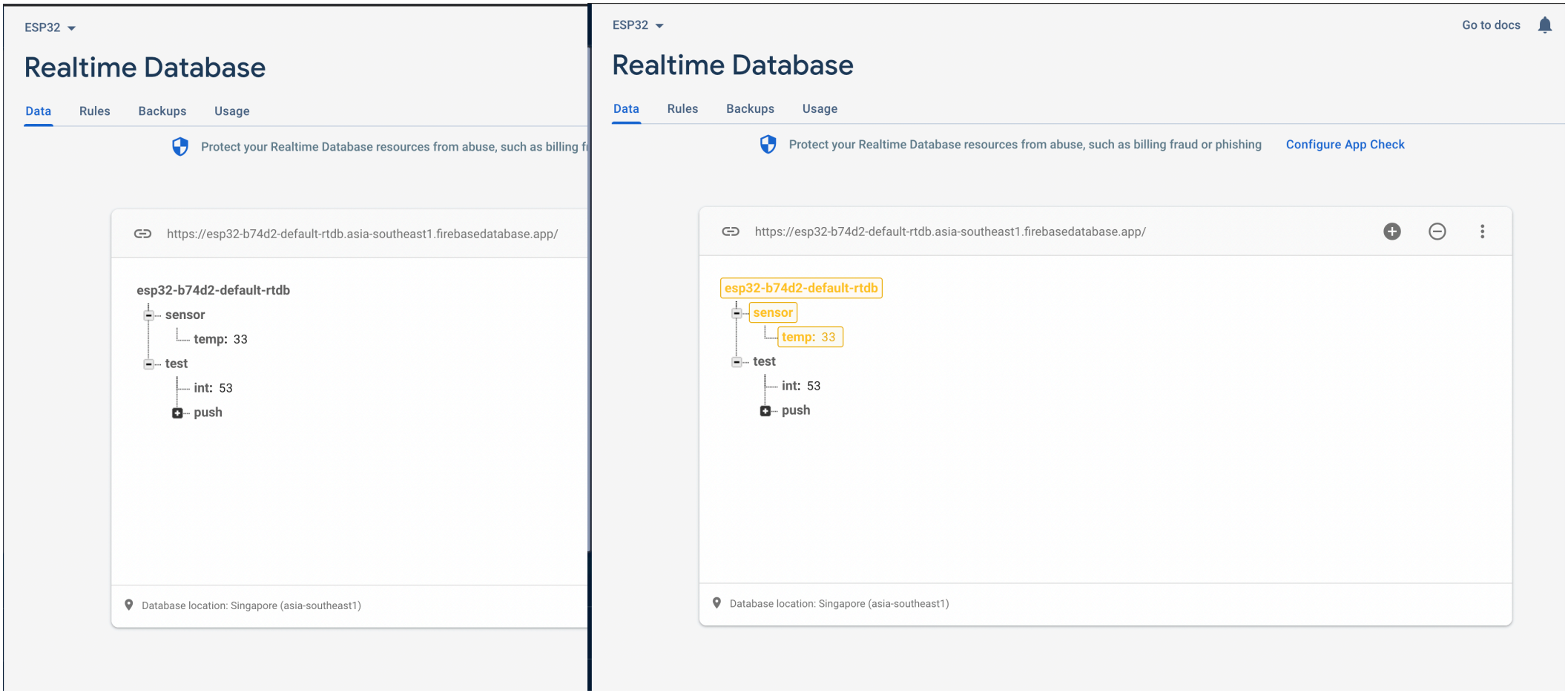




ESP32 publish temperature to firebase

sendTempToFirebase.ino

```
float readTemperature() {
  Wire1.beginTransmission(LM73_ADDR);
  Wire1.write(0x00); // Temperature Data Register
  Wire1.endTransmission();

  uint8_t count = Wire1.requestFrom(LM73_ADDR, 2);
  float temp = 0.0;
  if (count == 2) {
    byte buff[2];
    buff[0] = Wire1.read();
    buff[1] = Wire1.read();
    temp += (int)(buff[0]<<1);
    if (buff[1]&0b10000000) temp += 1.0;
    if (buff[1]&0b01000000) temp += 0.5;
    if (buff[1]&0b00100000) temp += 0.25;
    if (buff[0]&0b10000000) temp *= -1.0;
  }
  return temp;
}

void firebase_init() {
  // configure firebase API Key
  config.api_key = API_KEY;
  // configure firebase realtime database url
  config.database_url = DATABASE_URL;
  // Enable WiFi reconnection
  Firebase.reconnectWiFi(true);
  Serial.println("------------------------------------");
  Serial.println("Sign up new user...");
  // Sign in to firebase Anonymously
  if (Firebase.signUp(&config, &auth, "", "")) {
    Serial.println("Success");
    isAuthenticated = true;
    // Set the database path where updates will be loaded for this device
    databasePath = "/";
    fuid = auth.token.uid.c_str();
  } else {
    Serial.printf("Failed, %s\n", config.signer.signupError.message.c_str());
    isAuthenticated = false;
  }
  // Assign the callback function for the long running token generation task,
  // see addons/TokenHelper.h
  config.token_status_callback = tokenStatusCallback;
  // Initialise the firebase library
  Firebase.begin(&config, &auth);

void setup() {

  // Configure pin
  pinMode(led_pin_1, OUTPUT);
  pinMode(led_pin_2, OUTPUT);

  Serial.begin(115200);
  Wire1.begin(4, 5);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    digitalWrite(led_pin_1, HIGH);
    delay(500);
    digitalWrite(led_pin_1, LOW);
    delay(500);
  }
  digitalWrite(led_pin_1, LOW);
  Serial.println("Connected to the WiFi network");
  firebase_init();

}

void loop() {
  if (Firebase.ready() && (millis() - sendDataPrevMillis > update_interval || sendDataPrevMillis == 0))
  {
    Serial.println(readTemperature());
    sendDataPrevMillis = millis();

    Serial.printf("Set int... %s\n", Firebase.RTDB.setFloat(&fbdo, "/sensor/temp", readTemperature()) ? "ok" :
fbdo.errorReason().c_str());

    Serial.printf("Get int... %s\n", Firebase.RTDB.getFloat(&fbdo, "/sensor/temp") ? String(fbdo.intData()).c_str() :
fbdo.errorReason().c_str());

  }
}
```


ESP32
Realtime Database
Data
Rules
Backups
Usage
Protect your Realtime Database resources from abuse, such as billing f
https://esp32-b74d2-default-rtdb.asia-southeast1.firebasedatabase.app/
esp32-b74d2-default-rtdb
sensor
temp: 33
test
int: 53
push
Database location: Singapore (asia-southeast1)
ESP32
Go to docs
Realtime Database
Data
Rules
Backups
Usage
Protect your Realtime Database resources from abuse, such as billing fraud or phishing
Configure App Check
https://esp32-b74d2-default-rtdb.asia-southeast1.firebasedatabase.app/
esp32-b74d2-default-rtdb
sensor
temp: 33
test
int: 53
push
Database location: Singapore (asia-southeast1)

Firebase ESP32 listens to data

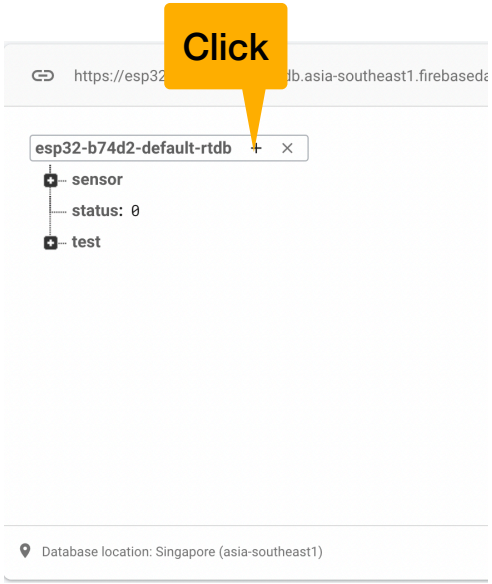
`firebaseDataChangeCallback.ino`

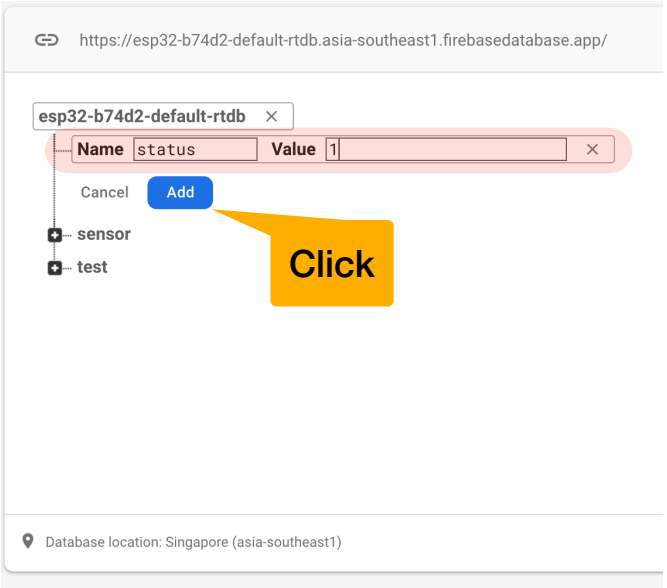
```
void firebase_init() {
  // configure firebase API Key
  config.api_key = API_KEY;
  // configure firebase realtime database url
  config.database_url = DATABASE_URL;
  // Enable WiFi reconnection
  Firebase.reconnectWiFi(true);
  Serial.println("------------------------------------");
  Serial.println("Sign up new user...");
  // Sign in to firebase Anonymously
  if (Firebase.signUp(&config, &auth, "", "")) {
    Serial.println("Success");
    isAuthenticated = true;
    // Set the database path where updates will be loaded for this device
    databasePath = "/";
    fuid = auth.token.uid.c_str();
  } else {
    Serial.printf("Failed, %s\n", config.signer.signupError.message.c_str());
    isAuthenticated = false;
  }
  // Assign the callback function for the long running token generation task,
  // see addons/TokenHelper.h
  config.token_status_callback = tokenStatusCallback;
  // Initialise the firebase library
  Firebase.begin(&config, &auth);

  if (!Firebase.RTDB.beginStream(&stream, "/status"))
    Serial.printf("sream begin error, %s\n\n", stream.errorReason().c_str());

  Firebase.RTDB.setStreamCallback(&stream, streamCallback, streamTimeoutCallback);
}
```

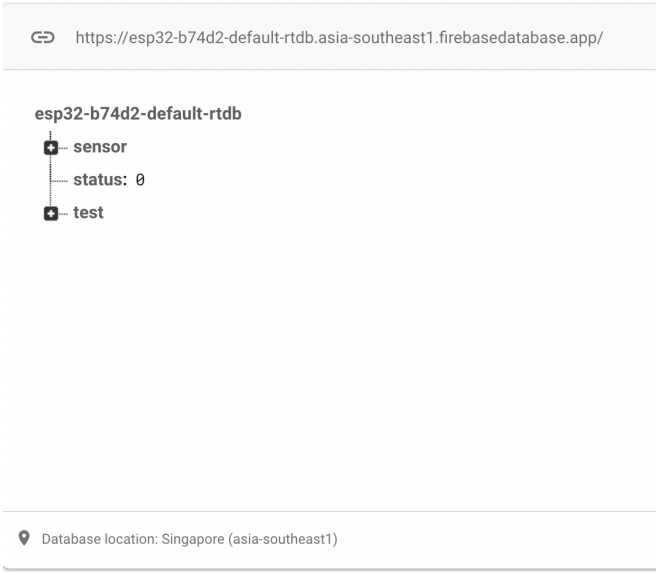
```
void streamCallback(FirebaseStream data)
{
  Serial.printf("sream path, %s\nevent path, %s\ndata type, %s\nevent type, %s\n\n",
                data.streamPath().c_str(),
                data.dataPath().c_str(),
                data.dataType().c_str(),
                data.eventType().c_str());
  printResult(data); //see addons/RTDBHelper.h
  Serial.println();

  if(data.intData() == 1) {
    digitalWrite(usb_pin,LOW);
  } else if(data.intData() == 0){
    digitalWrite(usb_pin, HIGH);
  }
}

void streamTimeoutCallback(bool timeout)
{
  if (timeout)
    Serial.println("stream timeout, resuming...\n");
}
```

Firebase ESP32 listens to data

Create new data value

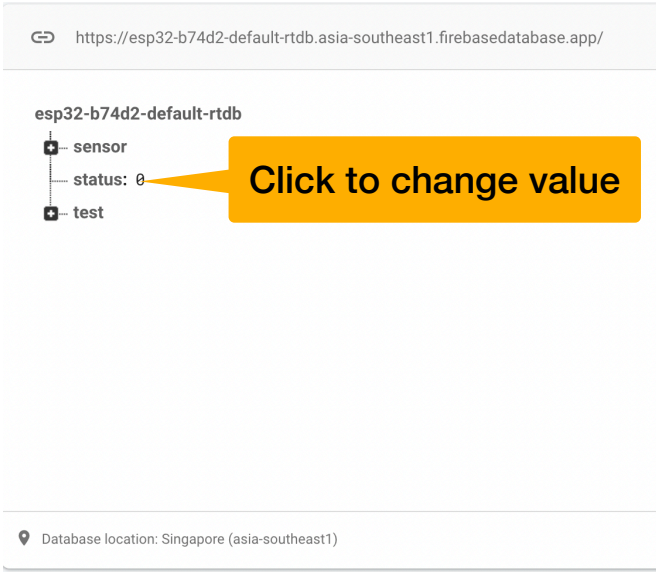




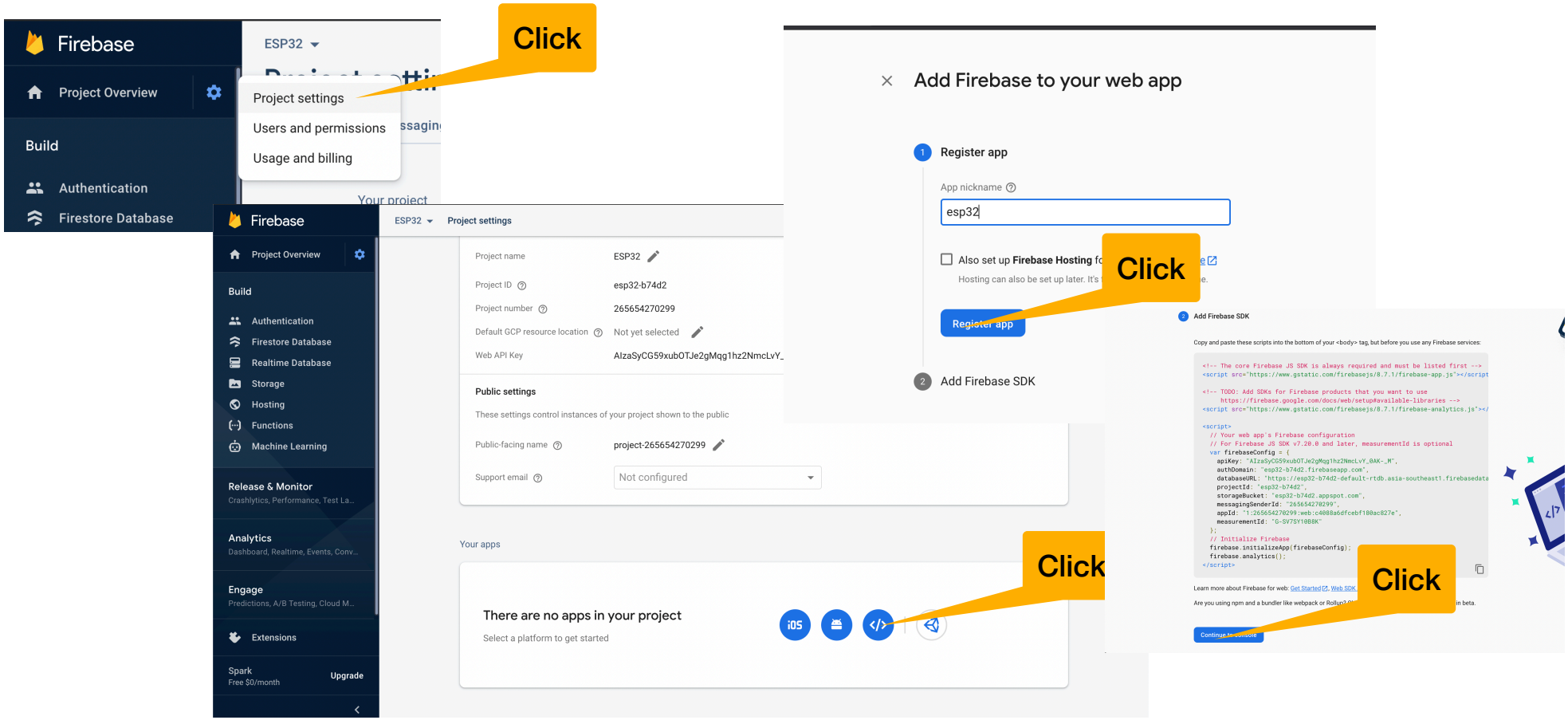


Test by changing status value

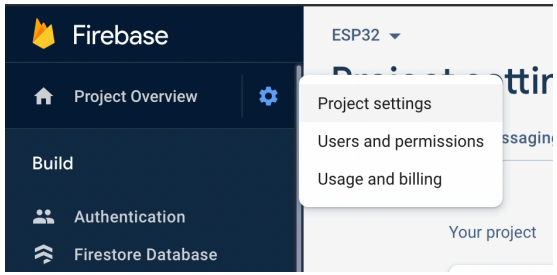
Observe USB light

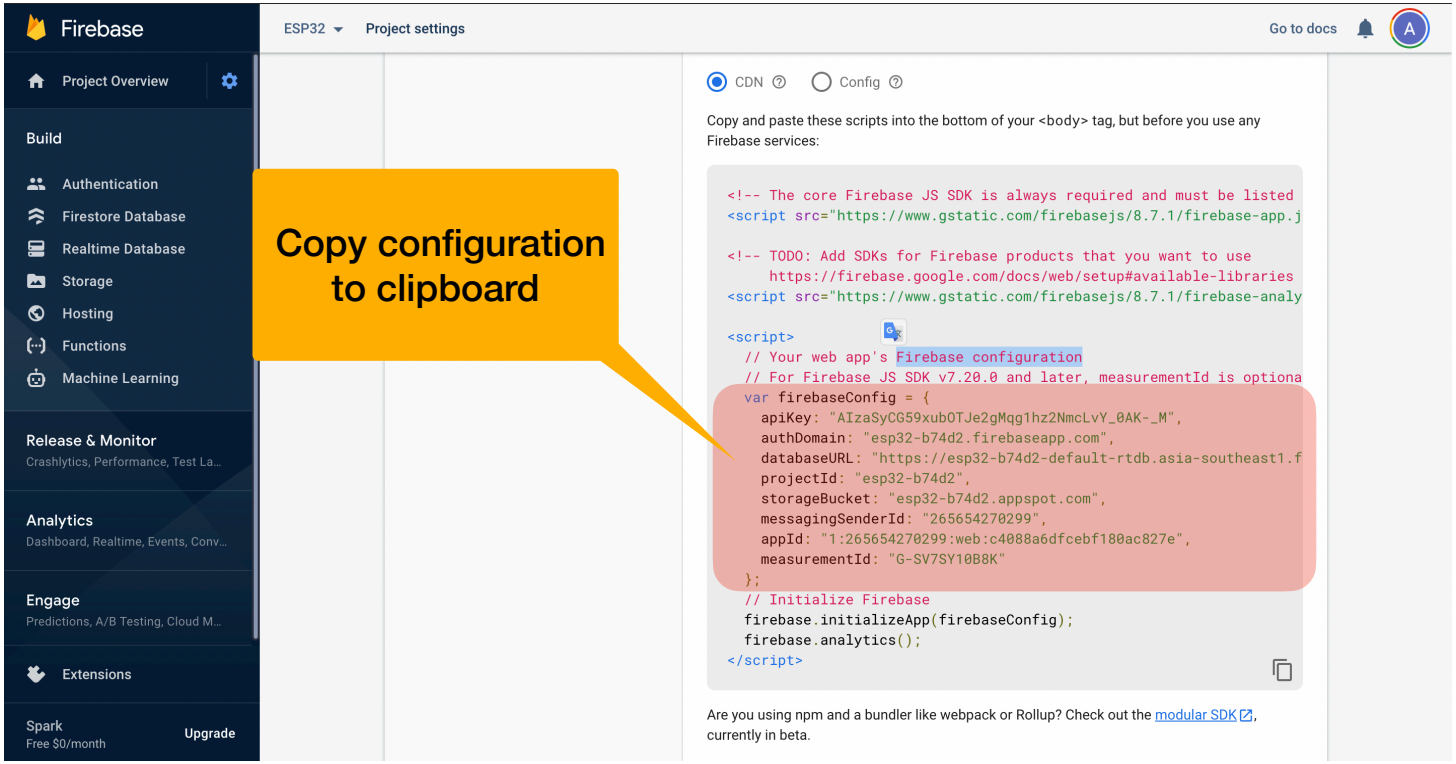


Web app dashboard



Get Firebase configuration





HTML file

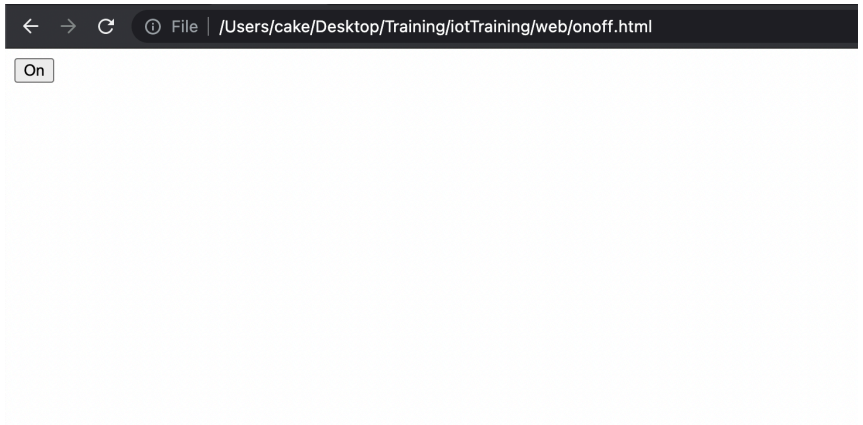
onoff.html

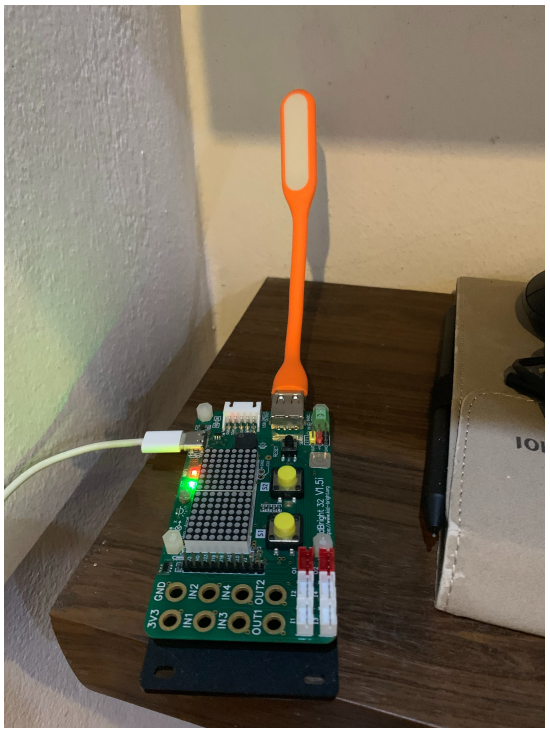
```
<html>
  <head>
    <script
      type="text/javascript"
      src="https://www.gstatic.com/charts/loader.js"
    ></script>
    <!-- The core Firebase JS SDK is always required and must be listed first -->
    <script src="https://www.gstatic.com/firebasejs/8.6.5/firebase-app.js"></script>

    <!-- TODO: Add SDKs for Firebase products that you want to use
     https://firebase.google.com/docs/web/setup#available-libraries -->
    <script src="https://www.gstatic.com/firebasejs/8.6.5/firebase-analytics.js"></script>
    <script src="https://www.gstatic.com/firebasejs/8.6.5/firebase-database.js"></script>

    <script>
      // Your web app's Firebase configuration
      // For Firebase JS SDK v7.20.0 and later, measurementId is optional

      var firebaseConfig = {
        apiKey: "AIzaSyCG59xubOTJe2gMqg1hz2NmcLvY_0AK-_M",
        authDomain: "esp32-b74d2.firebaseapp.com",
        databaseURL:"https://esp32-b74d2-default-rtdb.asia-
southeast1.firebasedatabase.app",
        projectId: "esp32-b74d2",
        storageBucket: "esp32-b74d2.appspot.com",
        messagingSenderId: "265654270299",
        appId: "1:265654270299:web:81685cf929e28a4dac827e",
        measurementId: "G-0F2YTRG1GX",
      };
      // Initialize Firebase
      firebase.initializeApp(firebaseConfig);
      firebase.analytics();

      if (!firebase.apps.length) {
        firebase.initializeApp(firebaseConfig);
      }
      this.database = firebase.database();

      function onoff() {
        currentvalue = document.getElementById("onoff").value;
        if (currentvalue == "Off") {
          document.getElementById("onoff").value = "On";
          firebase.database().ref("status").set(0);
        } else {
          document.getElementById("onoff").value = "Off";
          firebase.database().ref("status").set(1);
        }
      }
    </script>
  </head>

  <body>
    <input type="button" value="On" id="onoff" onclick="onoff();" />
  </body>
</html>
```

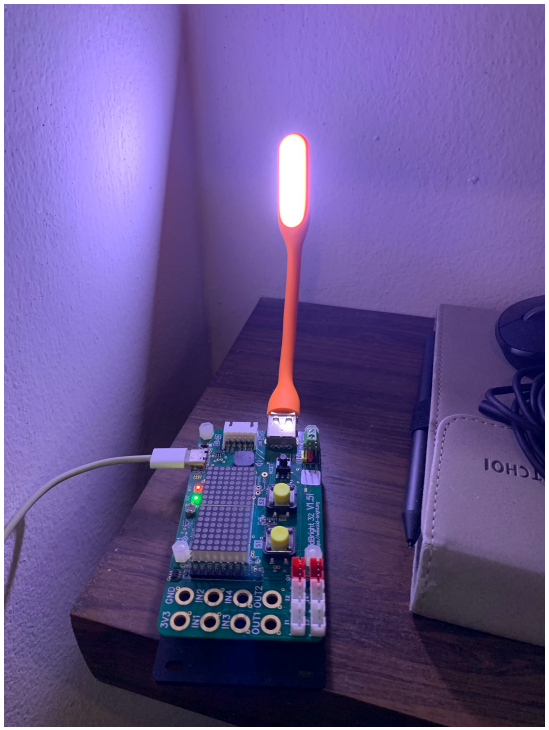
Paste clipboard

Load `onoff.html` using Chrome

Test on/off switch







Appendix

KidBright IDE