# 9. How to build your Own Home Assistant Environment?

Difficult Level: 

Building your own home assistant can be an exciting project, and there are several ways to go about it. One popular option is to use open-source software like Home Assistant, which can run on various platforms including Linux, Raspberry Pi, and Windows. Here's a step-by-step guide on how to set it up on each platform:

Here, we are going to build the whole system on a Raspberry Pi, and you can also use window or Linux OS to build the same environment for Home assistant.

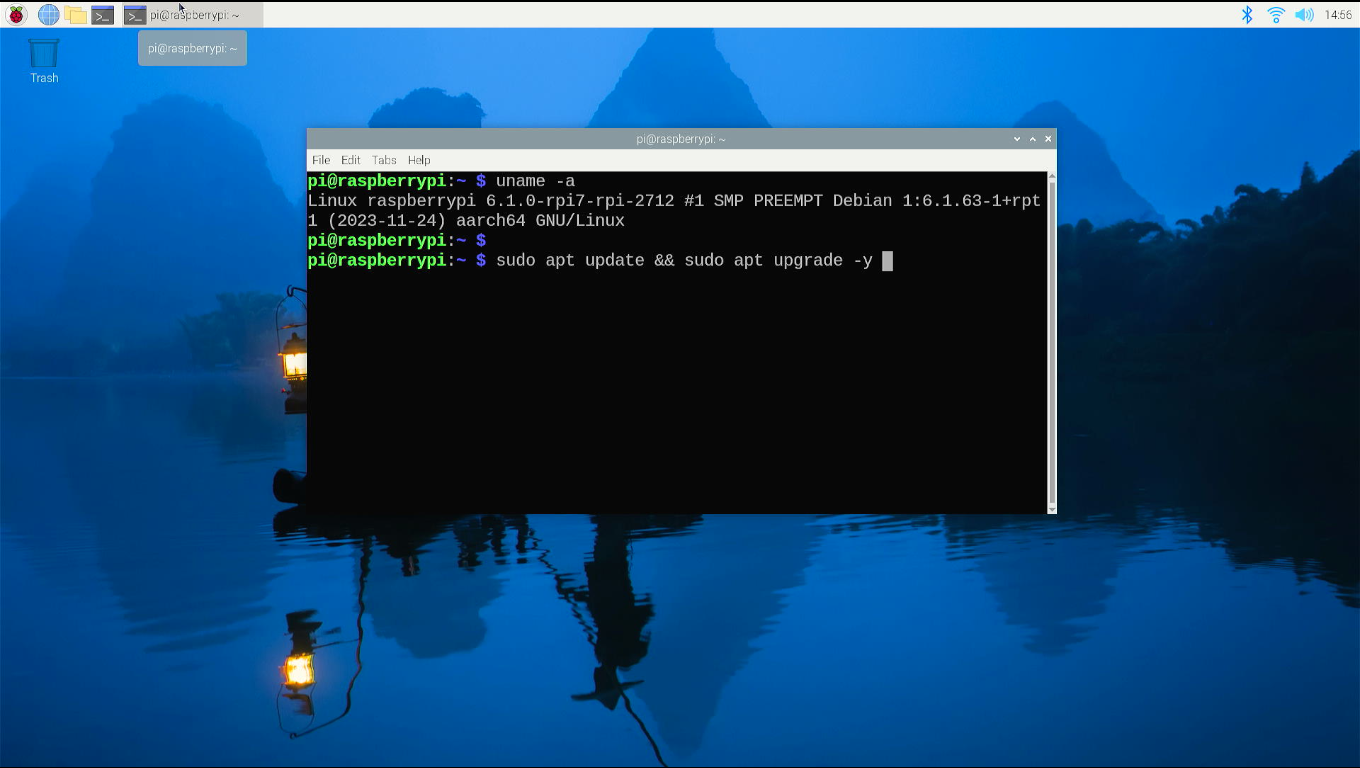
### Raspberry Pi:

* **Set up Raspberry Pi:** Install Raspberry Pi OS (for examples: Bullseye or Buster) on your Raspberry Pi.
* **Update your Pi:** Run the following commands to update your system:

Open a terminal and typing:

**sudo apt update**

**sudo apt upgrade**

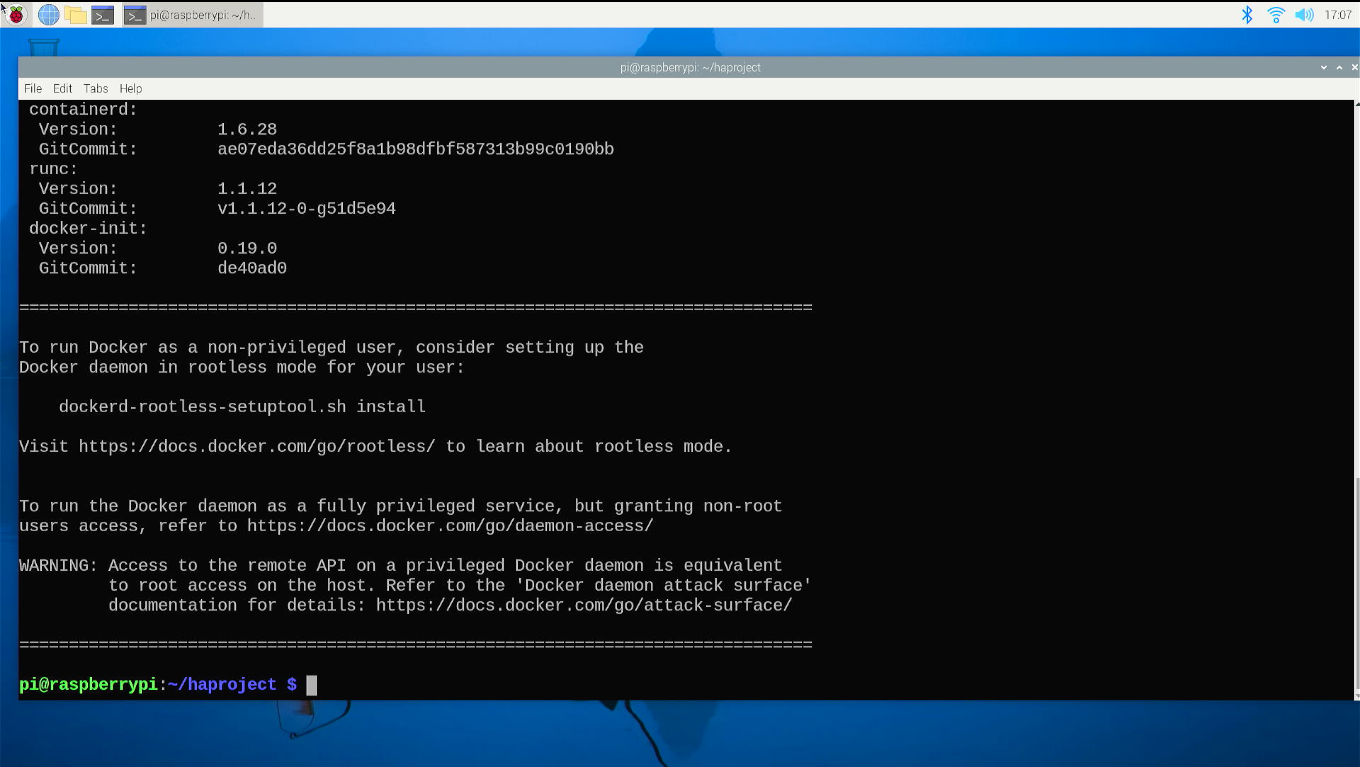


#### Install Home Assistant

curl -fsSL https://test.docker.com -o test-docker.sh

sudo sh test-docker.sh

Installation successful as following figure.



#### Install Docker environment

**Open a terminal on desktop and typing following command:**

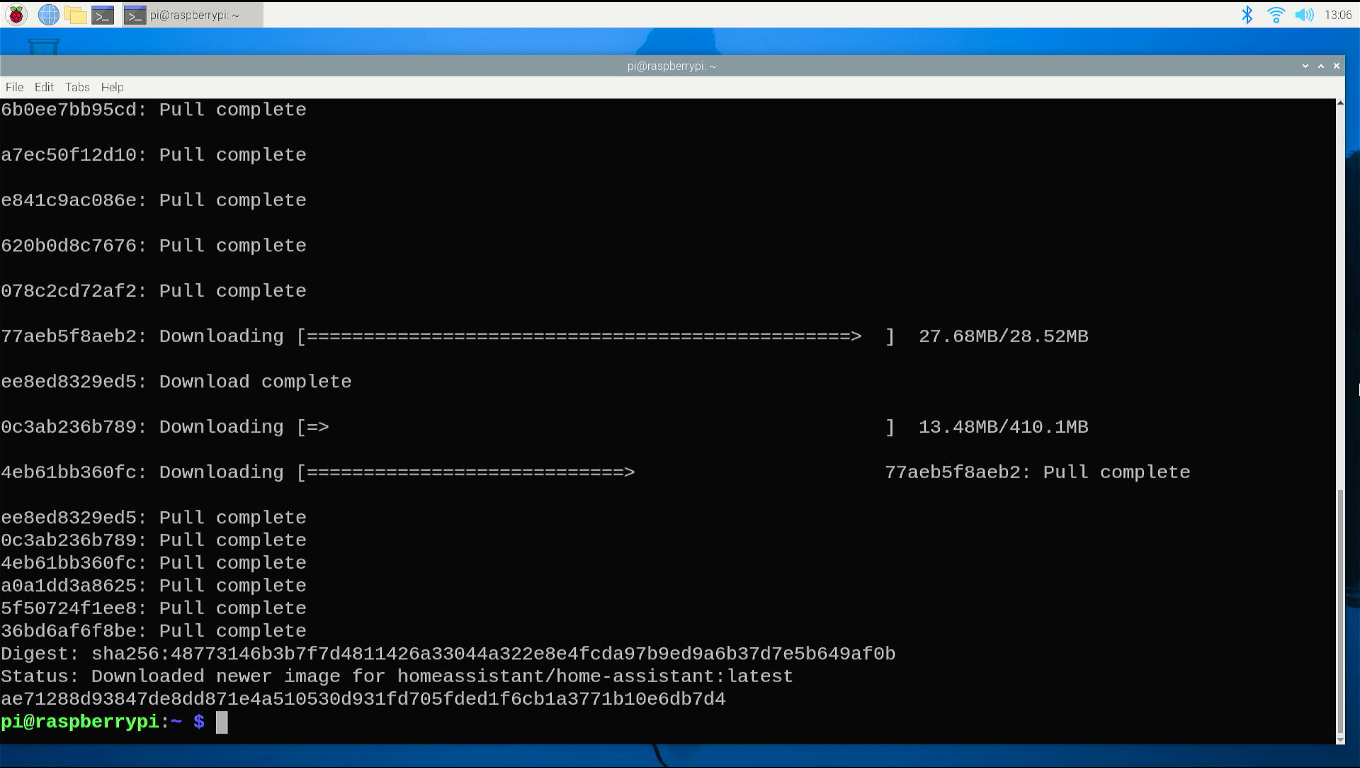
sudo docker run -d --name=homeassistant \

-v /opt/homeassistant:/config \

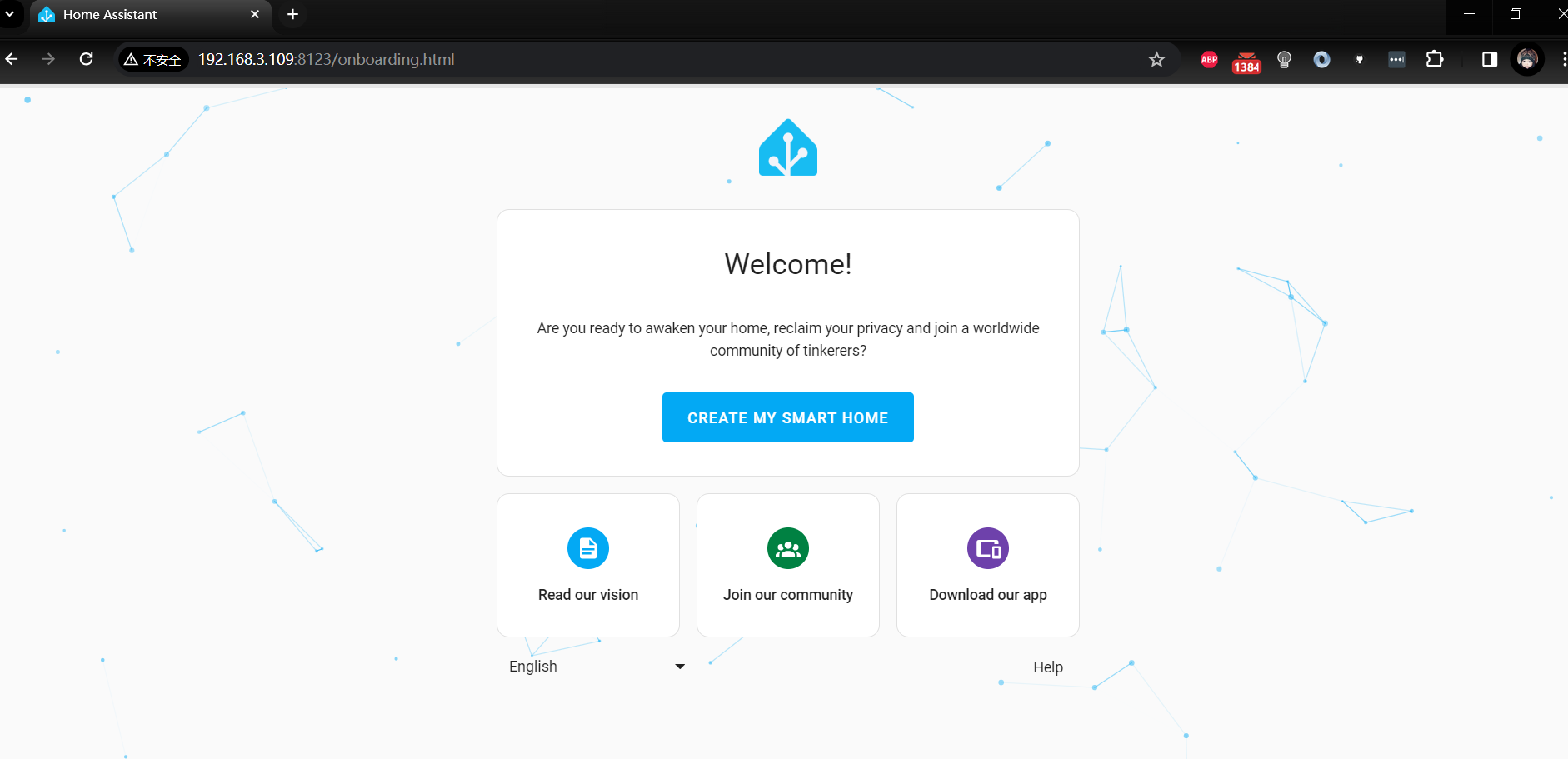
--net=host homeassistant/home-assistant

**and then, please wait for a while until it’s done.**

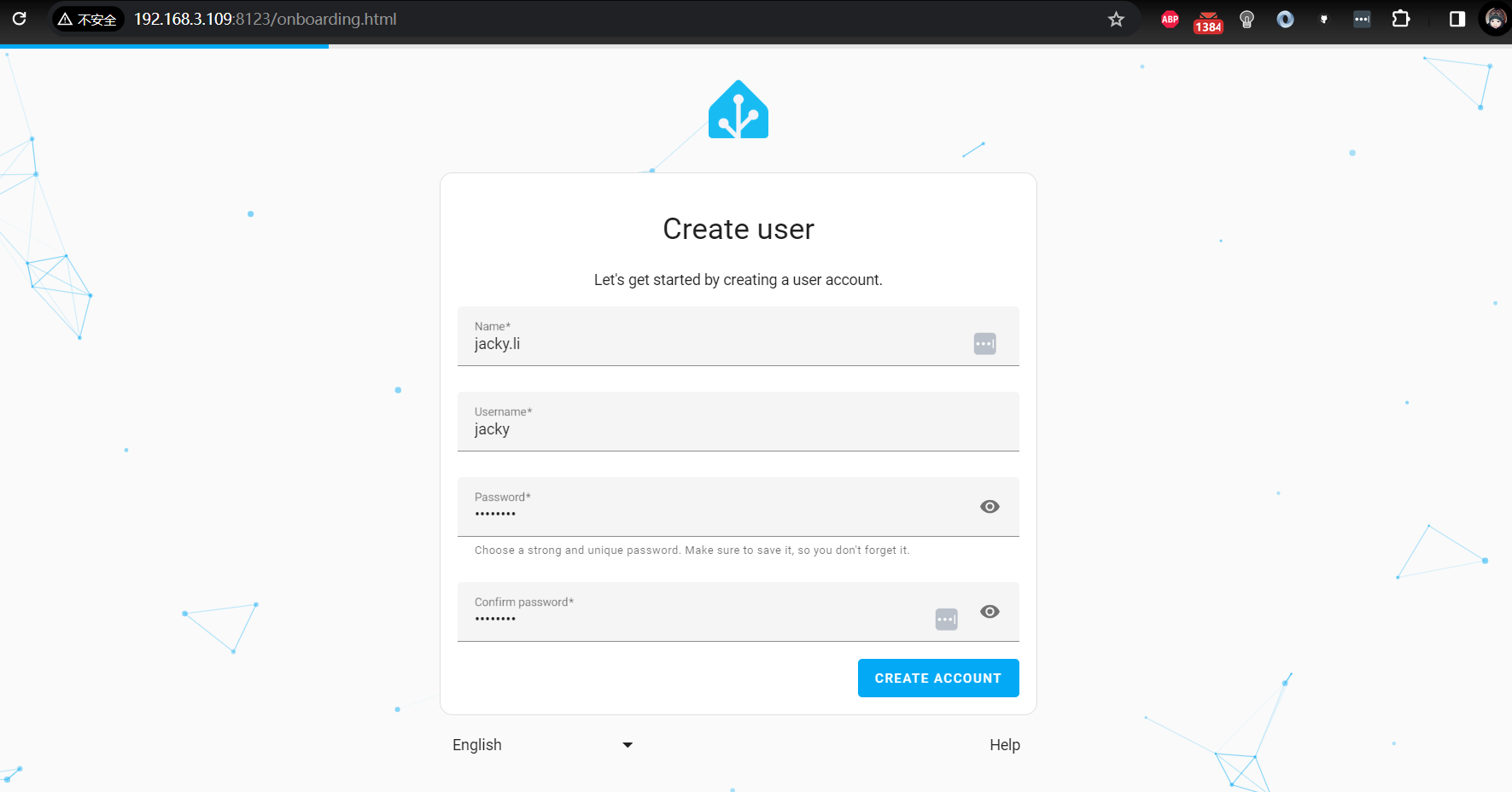




**First, enter Home Assistant through the web page to configure a default user password. The default Web access port is 8123. After entering, click Create My Smart Home.**



**Create a user to be administrator, please do remember the username and password that you created.**



**Click finish.**



**You will access the backend of the home-assistant, and next step is to install the MQTT server and modify the configuration.yaml file to make home assistant can display the information sending from Arduino UNO R4 Wi-Fi.**

#### Install MQTT Service

**Next, to install the MQTT service, you need to create the corresponding directory first.**

* **Create directories:**

sudo mkdir -p /opt/mosquitto/config

sudo mkdir -p /opt/mosquitto/data

sudo mkdir -p /opt/mosquitto/log



* **Create a default configuration file: /opt/mosquitto/config/mosquitto.conf**

listener 1883 0.0.0.0

persistence true

persistence\_location /mosquitto/data/

log\_dest file /mosquitto/log/mosquitto.log

# password\_file /mosquitto/config/password

per\_listener\_settings false

allow\_anonymous false



Save it and quit.

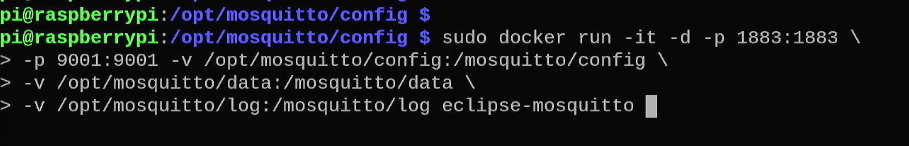
**Install the MQTT server. After running it, please record the returned ID.**

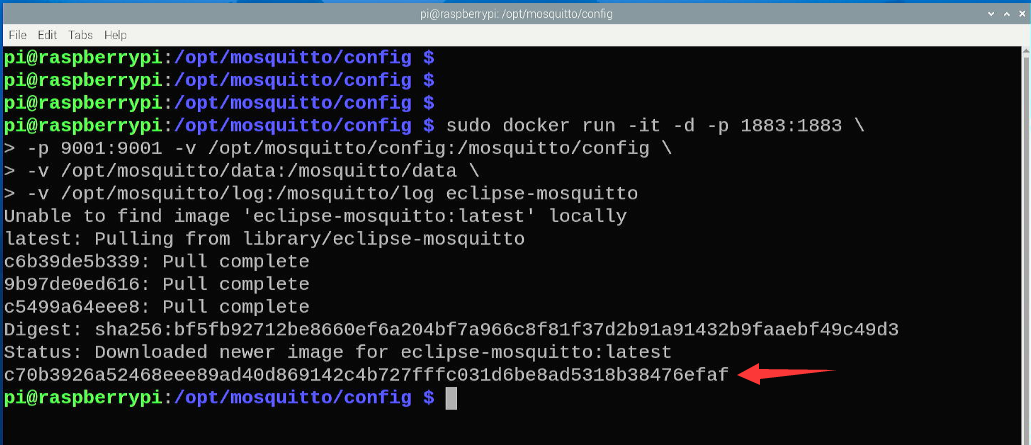
sudo docker run -it -d -p 1883:1883 \

-p 9001:9001 -v /opt/mosquitto/config:/mosquitto/config \

-v /opt/mosquitto/data:/mosquitto/data \

-v /opt/mosquitto/log:/mosquitto/log eclipse-mosquitto

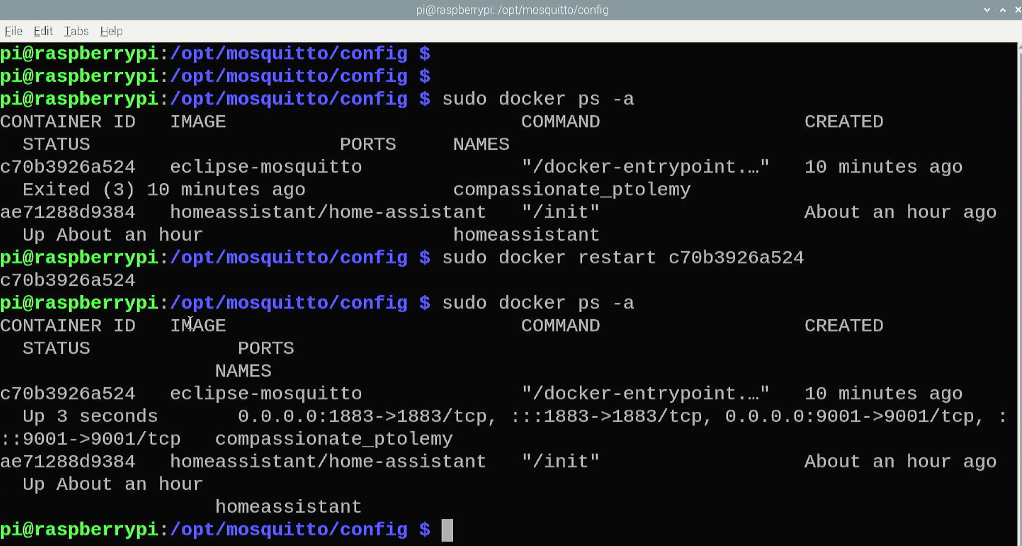




#### Manager Container

**You will need it later. If you do not remember it, please execute docker ps -a to obtain it, if service does not up, please restart the docker by using:**

sudo docker restart c70b3926a524



Enter the container and replace the ID with the ID just generated.

sudo docker exec -it c70b3926a524 sh

and then you need to generate password.

touch /mosquitto/config/password

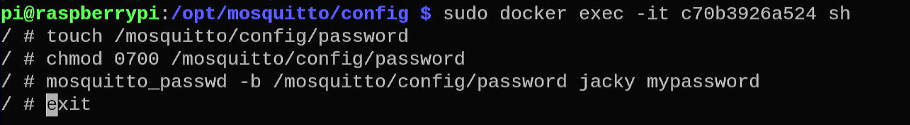
chmod 0700 /mosquitto/config/password

mosquitto\_passwd -b /mosquitto/config/password jacky mypassword

exit

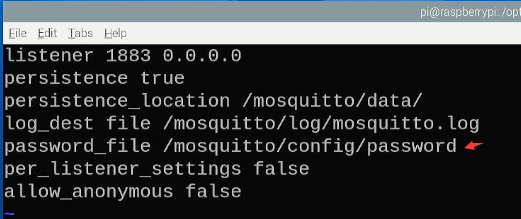
NOTE: please replace “jacky mypassword” with your own username and password.

It will be like:

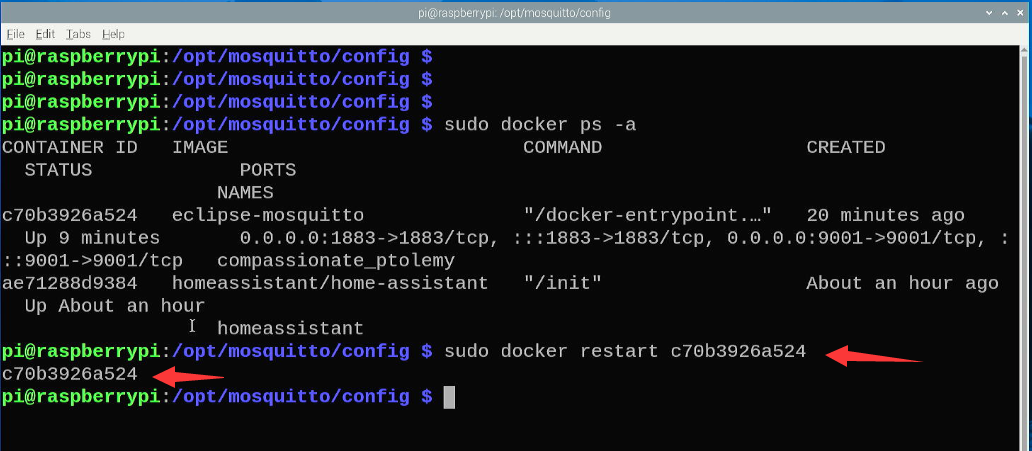


Next step is to enable password authentication by editing:

**/opt/mosquitto/config/mosquitto.conf** file, and remove the “#” hash mark before the password\_file parameter, and do remember restart all docker.



##### Restart container



##### Modify Configuration file

Next step, navigate to **/opt/homeassistant/configuration.yaml** configuration file and add an MQTT configuration to facilitate the next step.

mqtt:

  switch:

      - name: "Bedroom Switch"

        command\_topic: "home/bedroom/switch1/set"

        availability:

          - topic: "home/bedroom/switch1/available"

        payload\_on: "ON"

        payload\_off: "OFF"

        qos: 0

        retain: true

  sensor:

    - name: "Temperature Input 1"

      state\_topic: "uno/temp\_in1"

      unit\_of\_measurement: "°C"

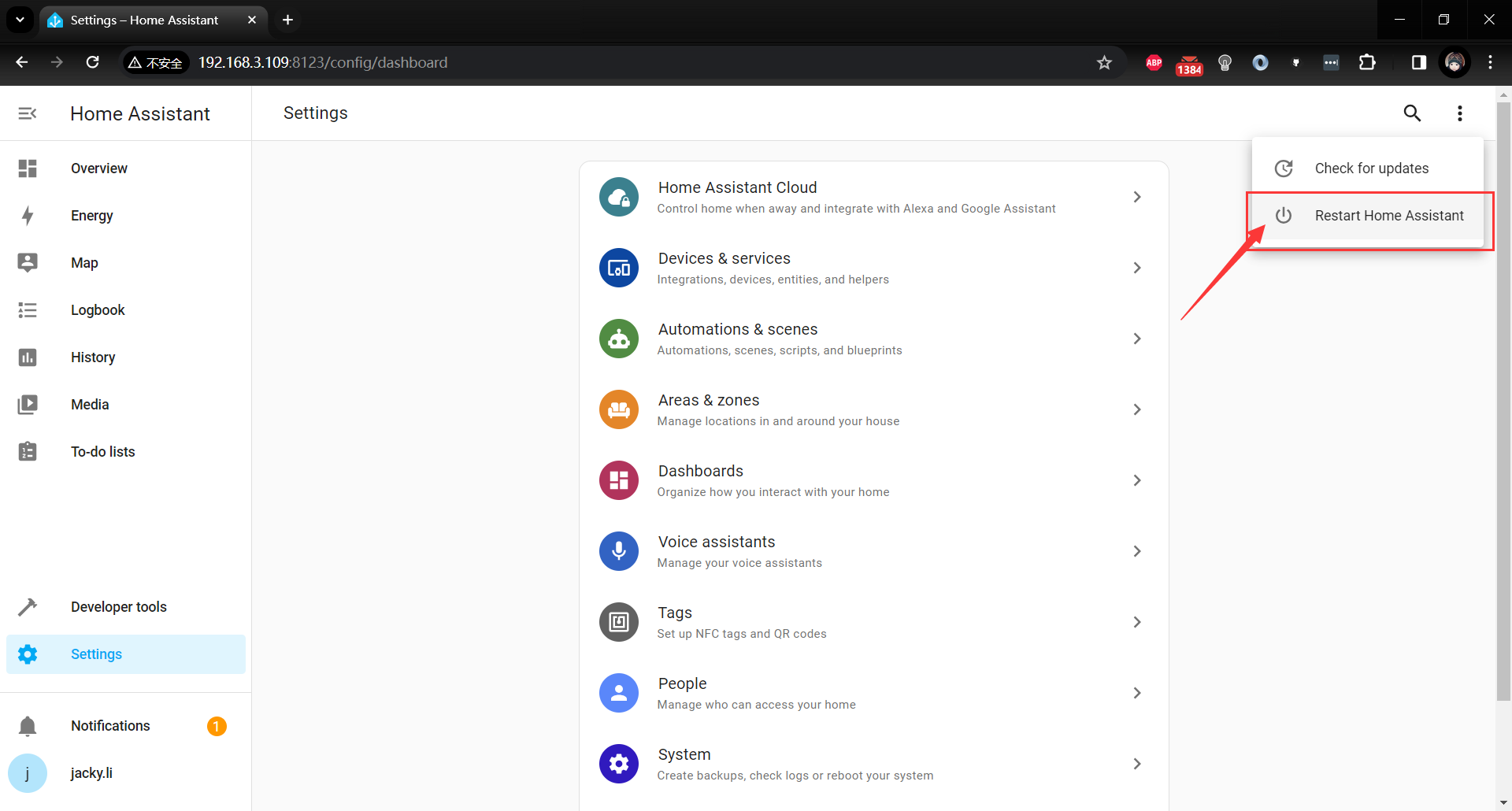
      value\_template: "{{ value }}"



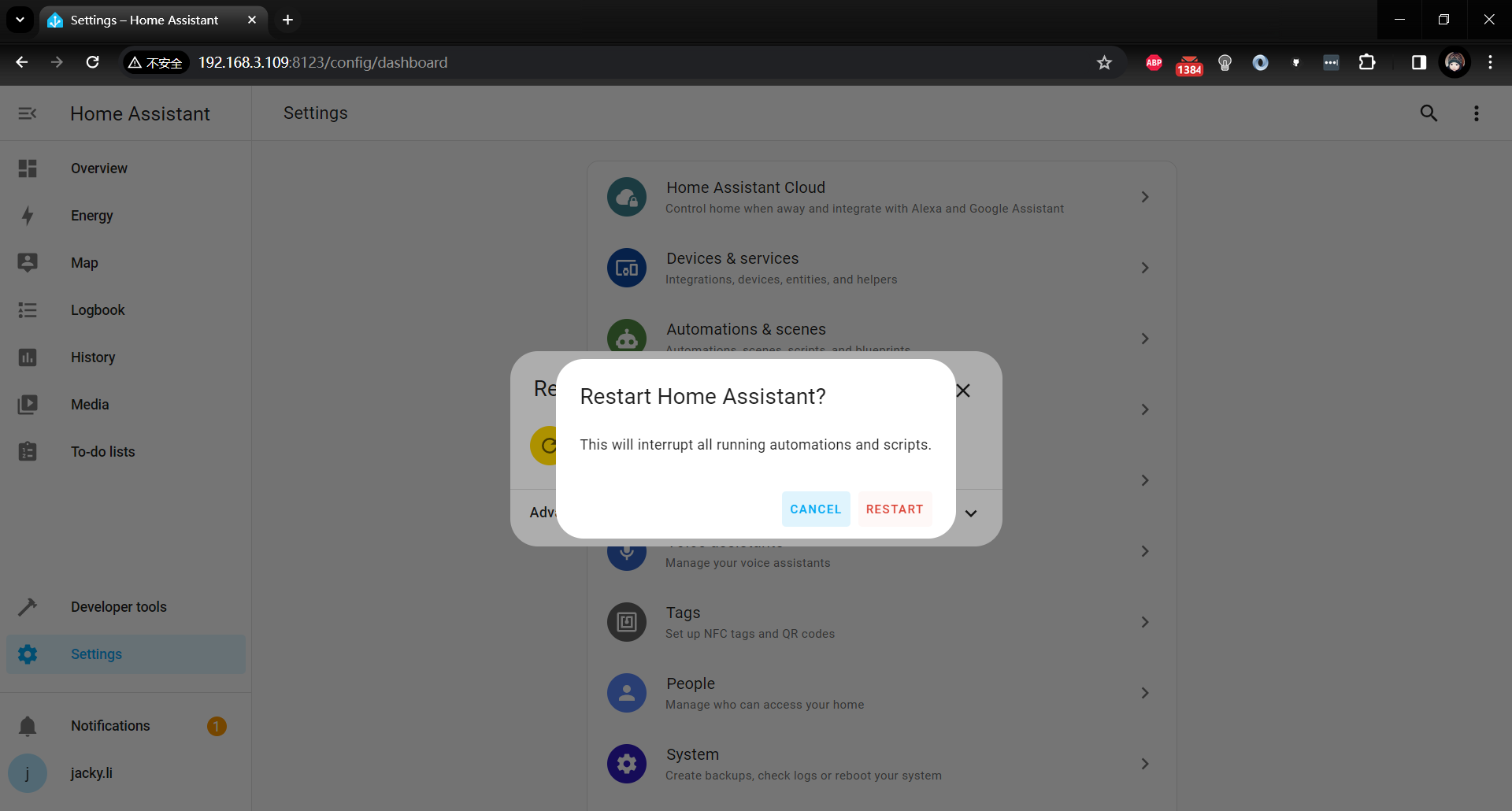
In this case, we have added two parts, switch is a button on the Home assistant which can be toggled to change status and send the result to Arduino UNO R4 via Wi-Fi connections.

##### Restart Home assistant

Reload the configuration by restart Home assistant on web console, please follow the steps in following figures.

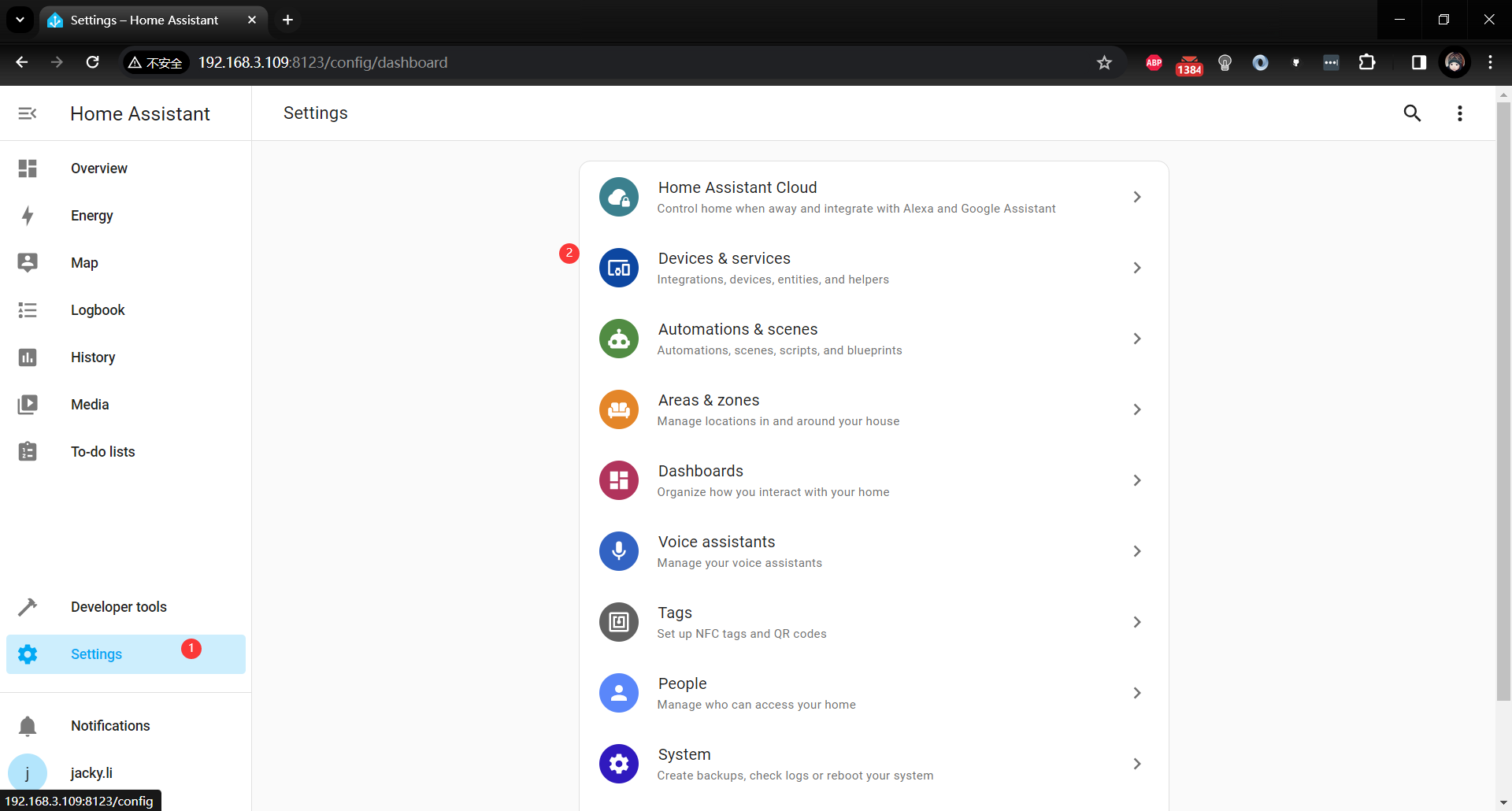


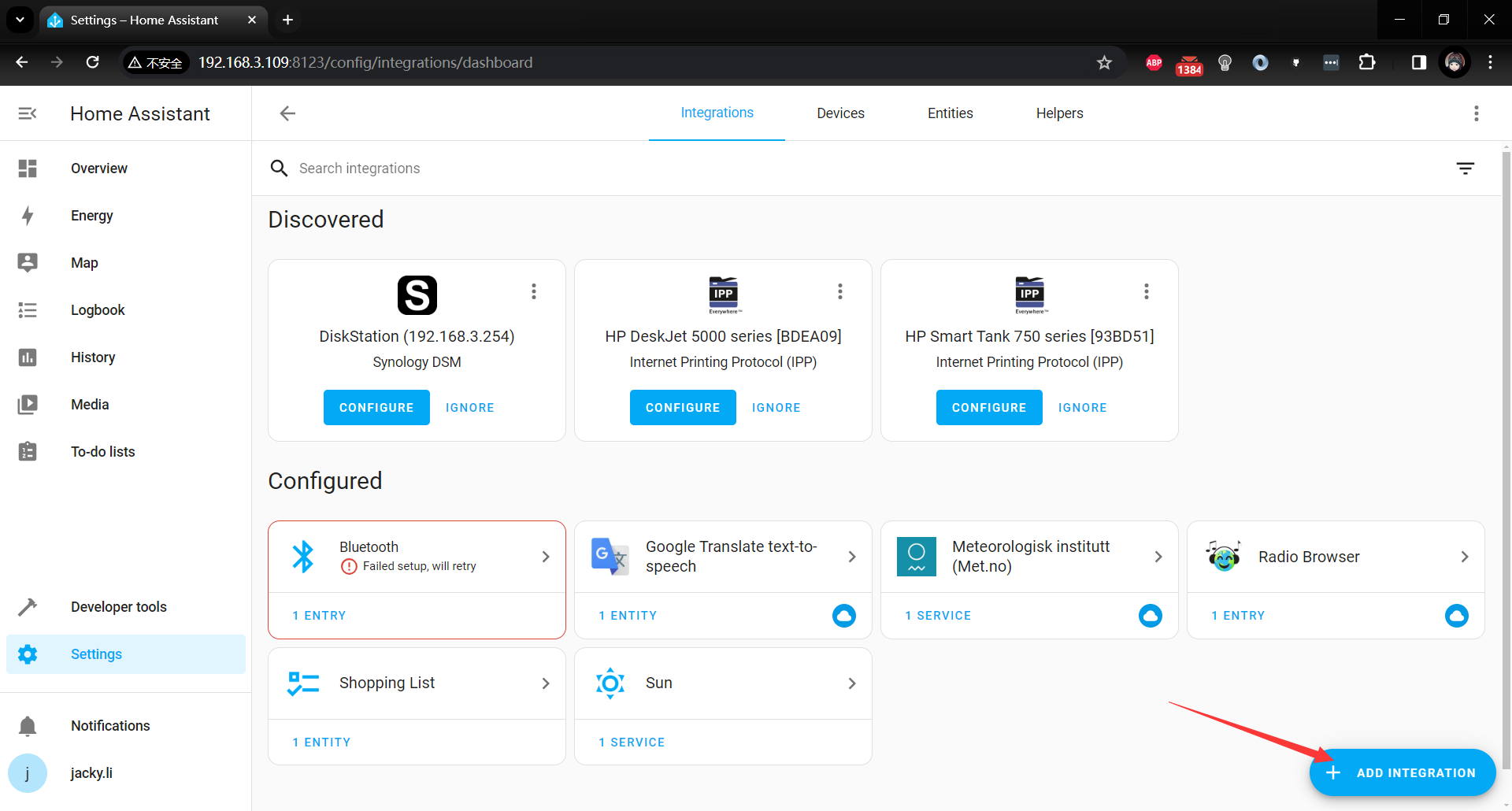


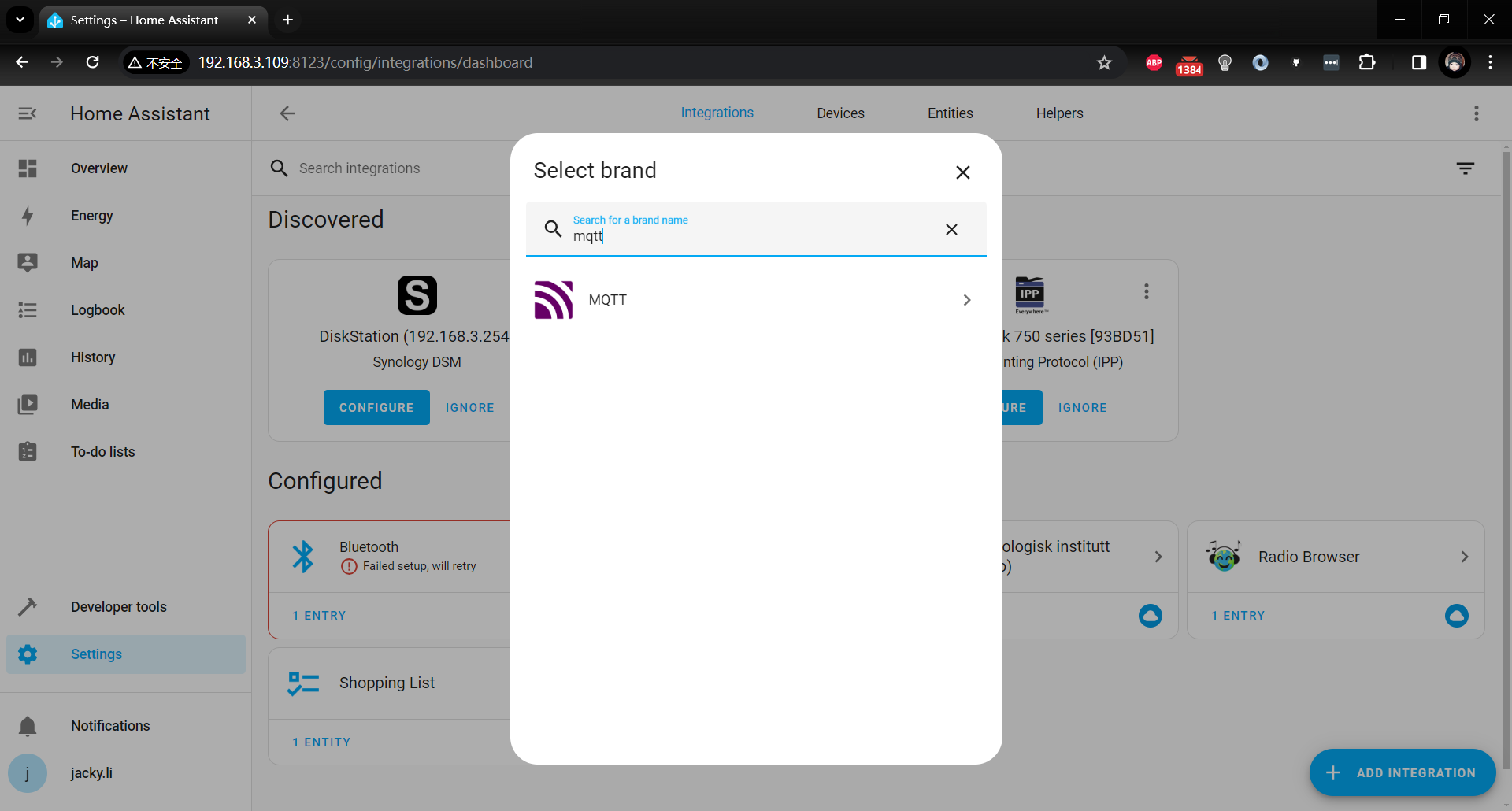


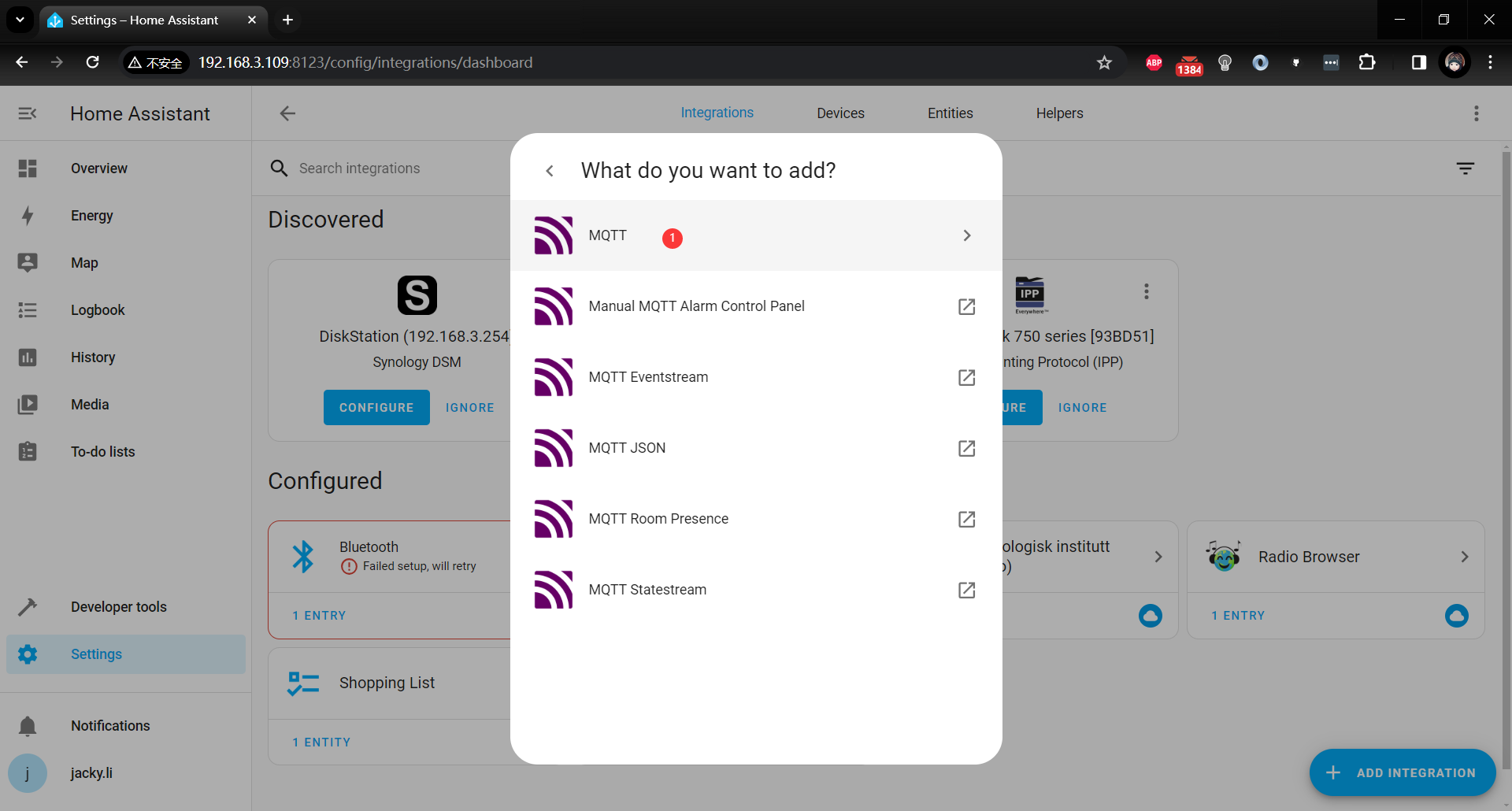
##### Adding integrations

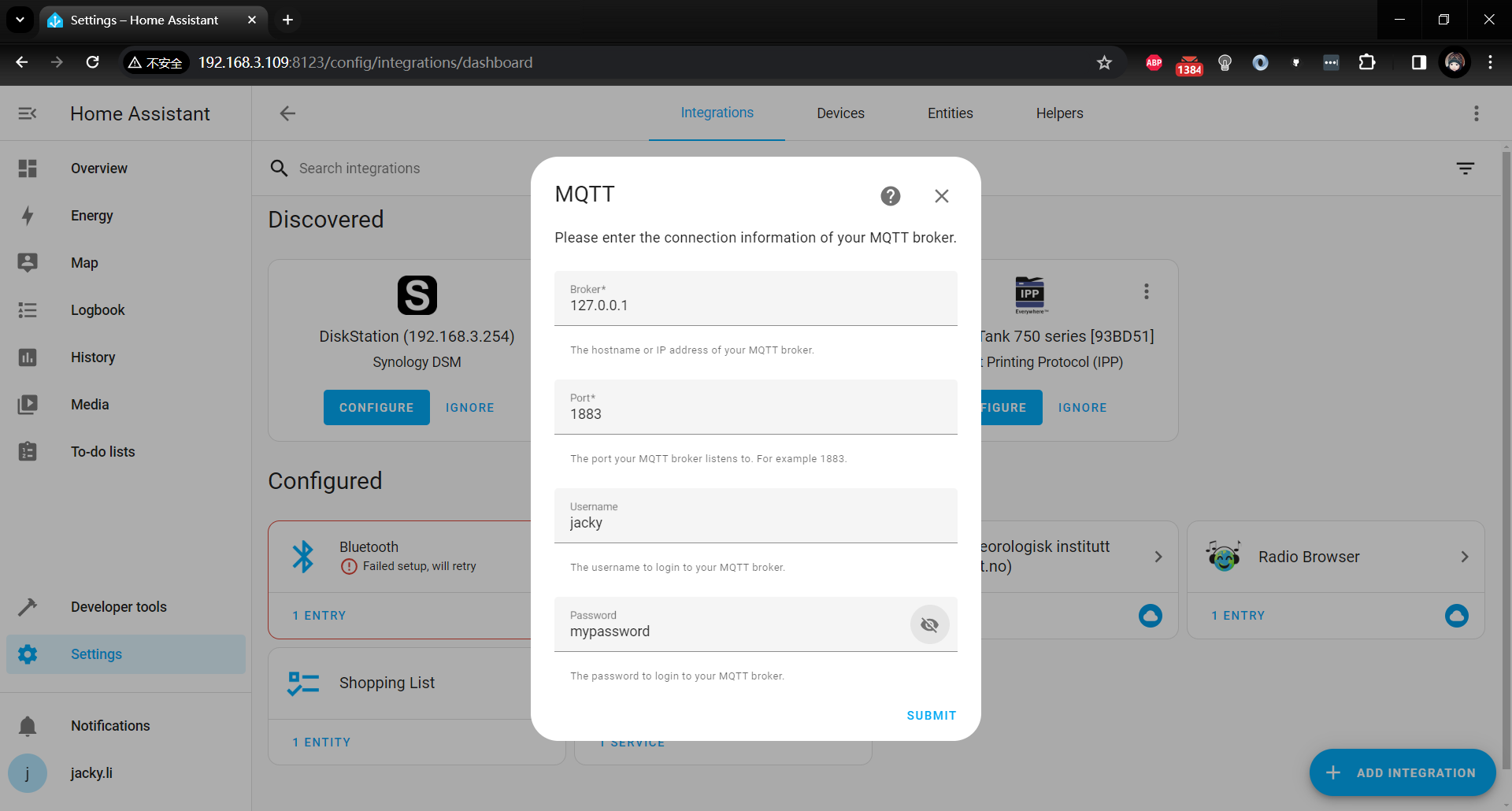
Next step is to add a MQTT server integration to the configuration.

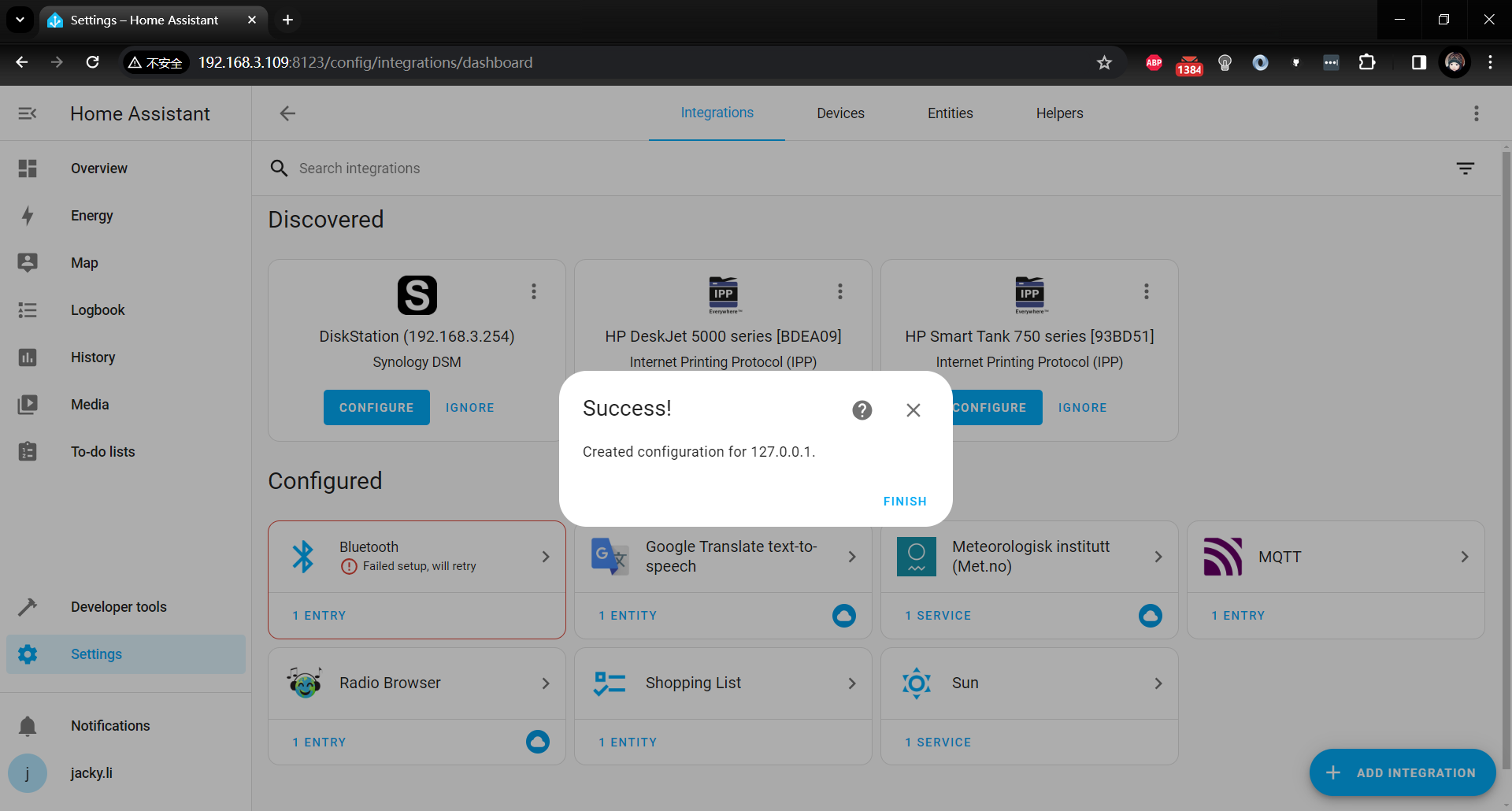




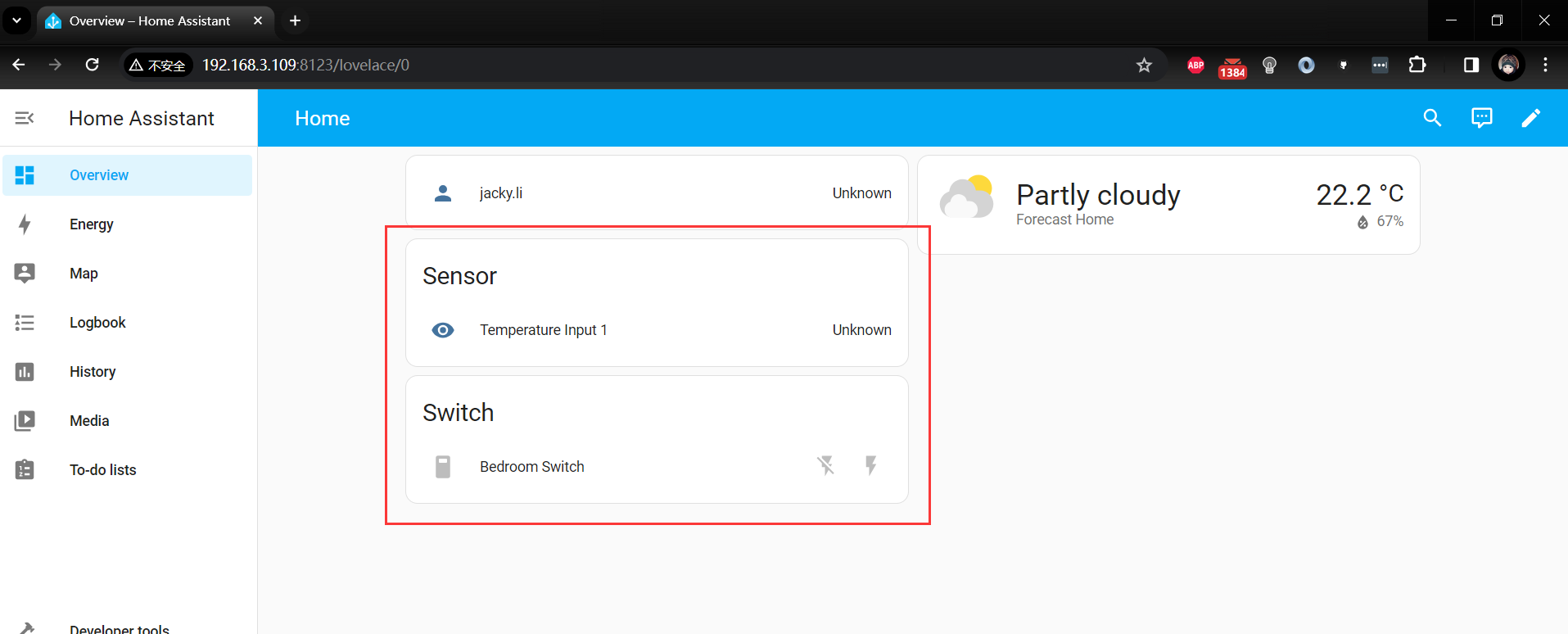








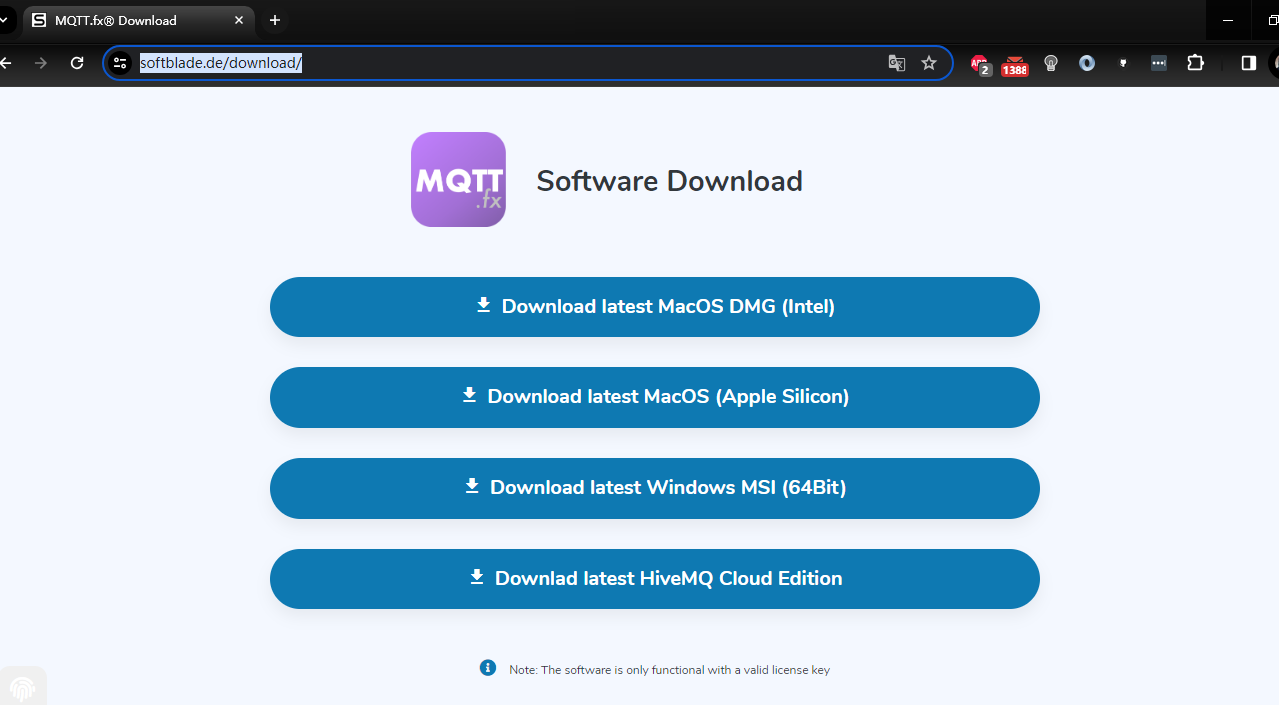
and you will find there are two new icons in dashboard overview.



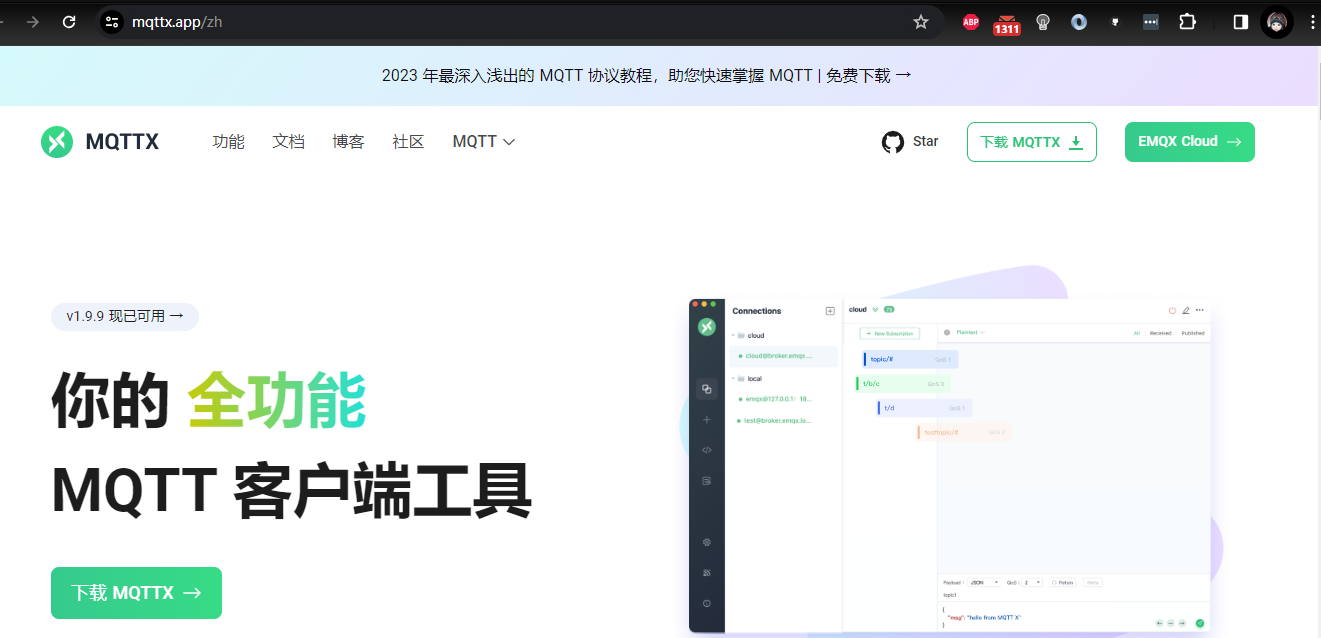
##### [Optional] Test broker server

If you want to test MQTT broker，you can just download the MQTTX test tool from internet.

<https://www.softblade.de/download/>



or **mqttx.app**

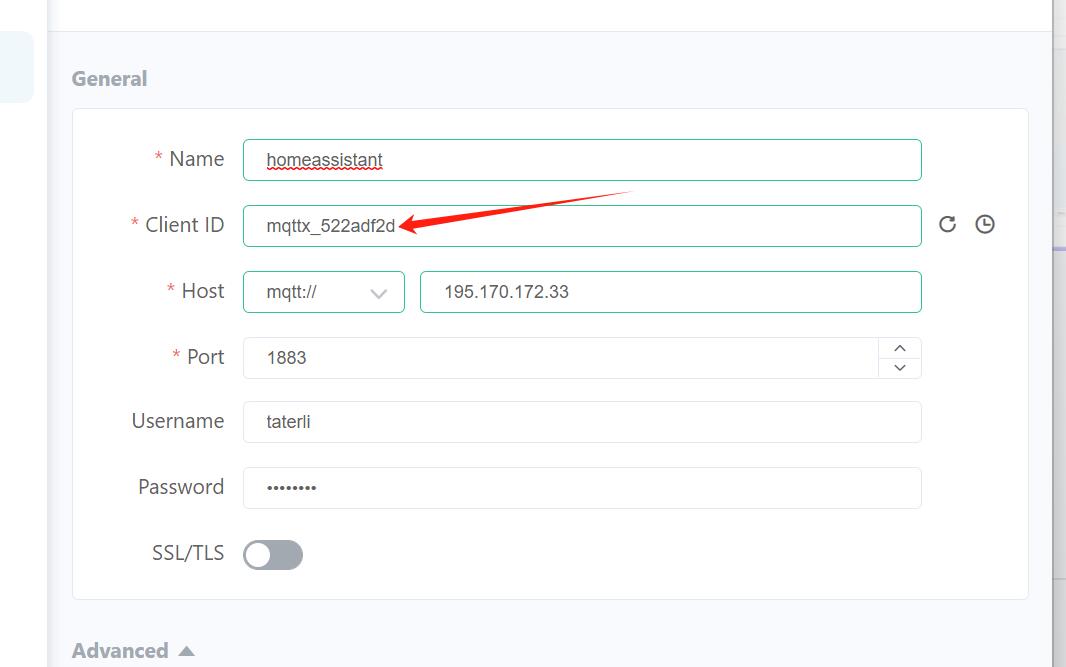


After downloading the software, please install it according to the software guide, and then open the MQTTX and filling the blank of your situation. here you need to make sure the Client ID field must be unique. Input the MQTT host’s address or Domain Name.

If you have been setting username and password for your MQTT host, you can put it into the blank.

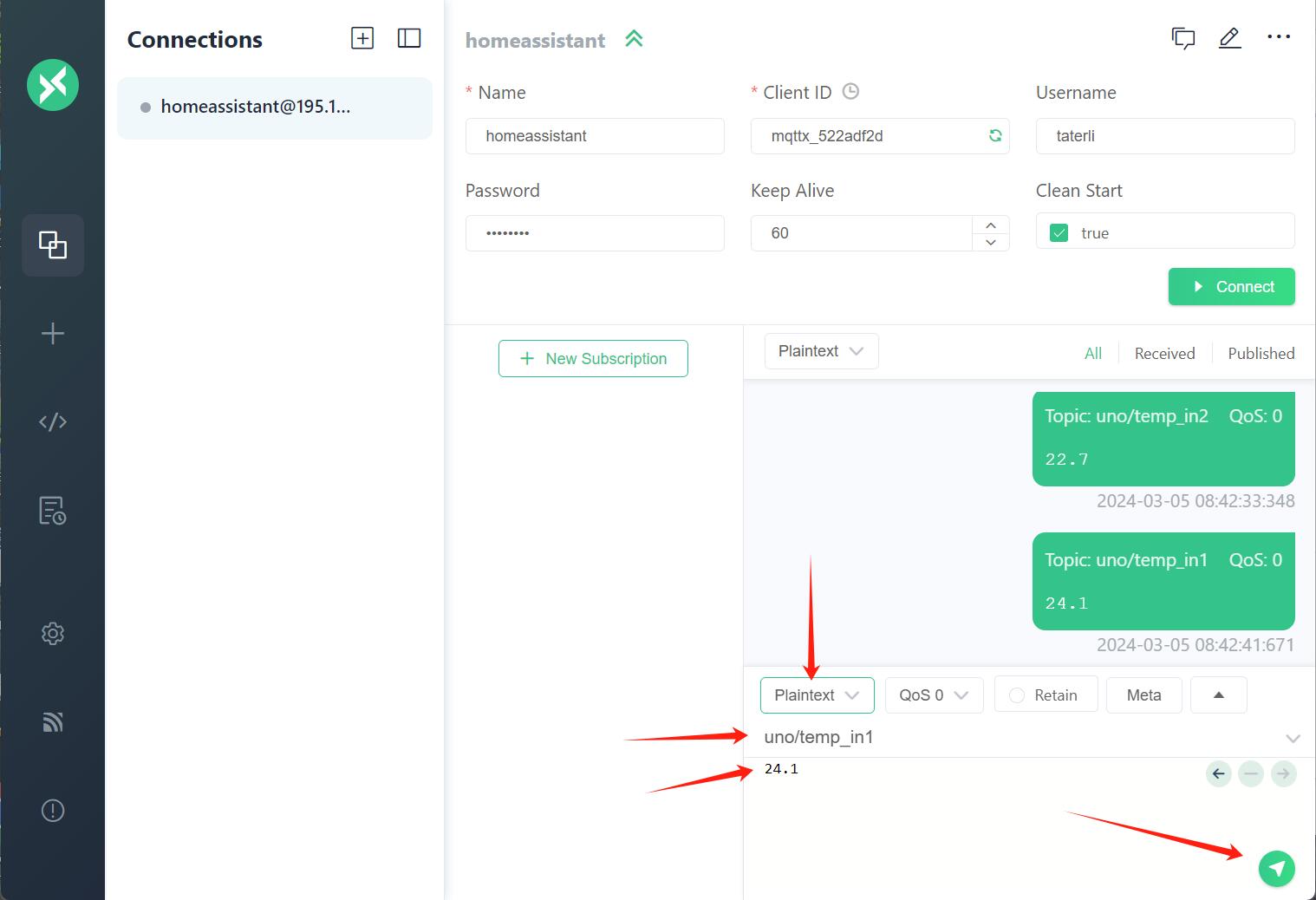
For example:

Test broker server by using MQTTX.app



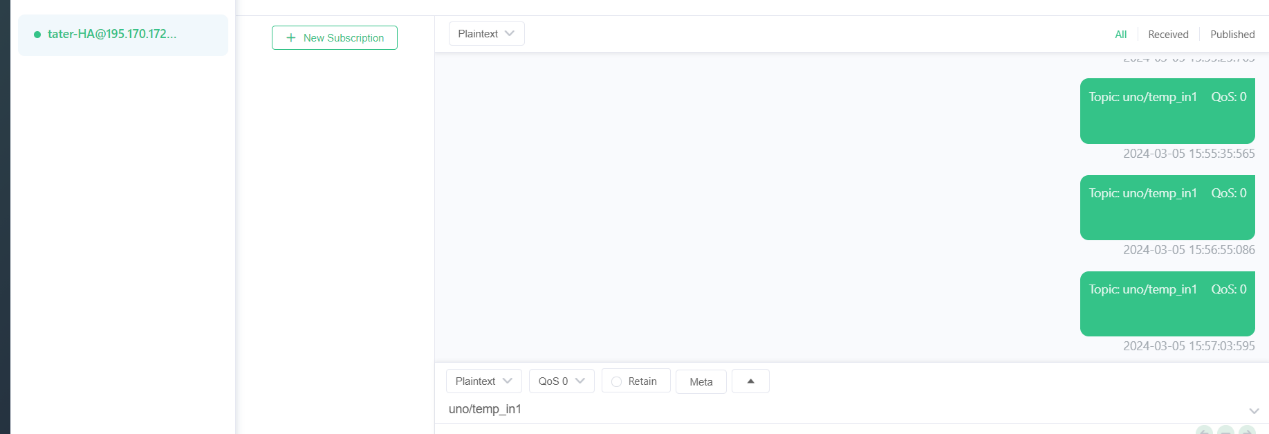
* **HOST**： MQTT host IP or domain name
* **Port**： 1883 default listening port, If the host supports SSL/TLS feature, you need to turn on the switch below, and the default port will be 8883.
* **Username/Password:** MQTT broker host access token.

You can pass the plaintext type message to a topic and send some data to broker server according to following figure.



Select plaintext and then find topic blank, filling the data into the data part and click green arrow icon to send the message to broker HOST.

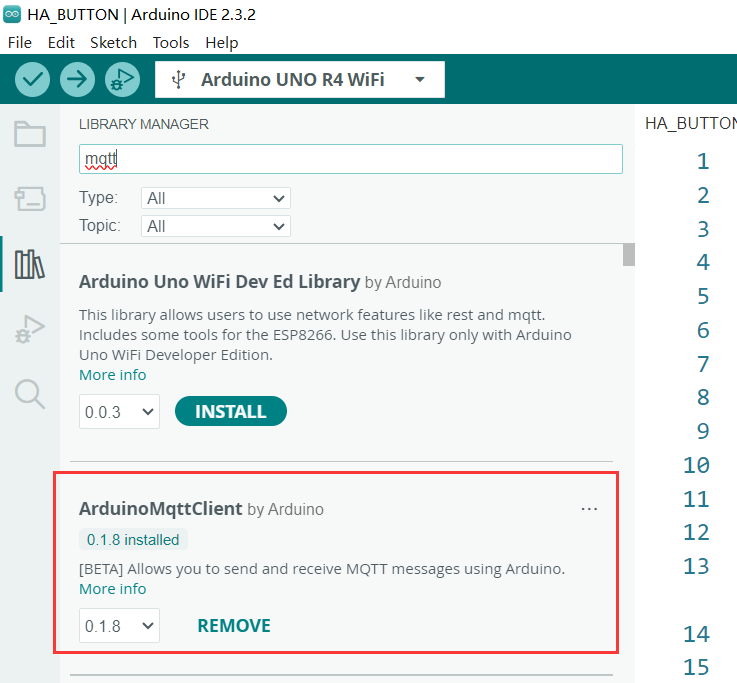
When the data has been sent successfully, it will show on the testing broker host at background administration page.



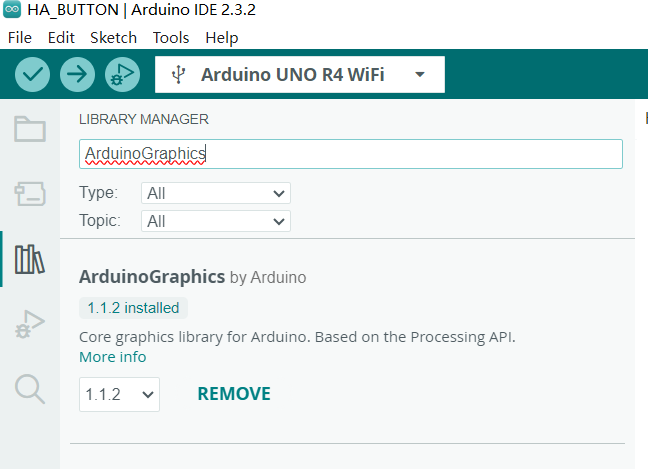
#### Arduino UNO R4 Wi-Fi Part

##### Install library

* Requirement: ArduinoMqttClient library.

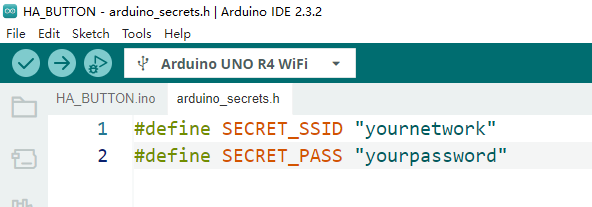


* Requirement2: ArduinoGraphics library



* **arduino\_secrets.h** file





##### Demo code

#include <ArduinoMqttClient.h>

#include <WiFiS3.h>

#include "ArduinoGraphics.h"

#include "Arduino\_LED\_Matrix.h"

#include <string.h>

#include <stdlib.h>

#include "arduino\_secrets.h"

///////please enter your sensitive data in the Secret tab/arduino\_secrets.h

char ssid[] = SECRET\_SSID;  // your network SSID (name)

char pass[] = SECRET\_PASS;  // your network password (use for WPA, or use as key for WEP)

ArduinoLEDMatrix matrix;

byte frame[8][12] = {

  { 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0 },

  { 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0 },

  { 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0 },

  { 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0 },

  { 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0 },

  { 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0 },

  { 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0 },

  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }

};

// To connect with SSL/TLS:

// 1) Change WiFiClient to WiFiSSLClient.

// 2) Change port value from 1883 to 8883.

// 3) Change broker value to a server with a known SSL/TLS root certificate

//    flashed in the WiFi module.

WiFiClient wifiClient;

MqttClient mqttClient(wifiClient);

const char broker[] = "192.168.3.109";

// const char broker[] = "195.170.172.33";

int port = 1883;

int buzzer = 9;

void drawText(const char text[]){

  matrix.beginDraw();

  matrix.stroke(0xFFFFFFFF);

  matrix.textScrollSpeed(50);

  // add the text

  matrix.textFont(Font\_5x7);

  matrix.beginText(0, 1, 0xFFFFFF);

  matrix.println(text);

  matrix.endText(SCROLL\_LEFT);

  matrix.endDraw();

}

void setup() {

  //Initialize serial and wait for port to open:

  matrix.begin();

  matrix.renderBitmap(frame, 8, 12);

  Serial.begin(9600);

  pinMode(buzzer, OUTPUT);

  digitalWrite(buzzer, LOW);

  while (!Serial) {

    ;  // wait for serial port to connect. Needed for native USB port only

  }

  // attempt to connect to WiFi network:

  Serial.print("Attempting to connect to WPA SSID: ");

  Serial.println(ssid);

  while (WiFi.begin(ssid, pass) != WL\_CONNECTED) {

    // failed, retry

    Serial.print(".");

    delay(5000);

  }

  Serial.print("IP address:");

  Serial.println(WiFi.localIP());

  Serial.println("You're connected to the network");

  Serial.println();

  // You can provide a unique client ID, if not set the library uses Arduino-millis()

  // Each client must have a unique client ID

  mqttClient.setId("uno\_R4\_01");

  // You can provide a username and password for authentication

  mqttClient.setUsernamePassword("jacky", "mypassword");

  Serial.print("Attempting to connect to the MQTT broker: ");

  Serial.println(broker);

  if (!mqttClient.connect(broker, port)) {

    Serial.print("MQTT connection failed! Error code = ");

    Serial.println(mqttClient.connectError());

    while (1){

    if (!mqttClient.connect(broker, port)) {

    Serial.print("MQTT connection failed! Error code = ");

    Serial.println(mqttClient.connectError());

      }

    }

  }

  Serial.println("You're connected to the MQTT broker!");

  Serial.println();

  Serial.println("Sending available to topic!");

  // send message, the Print interface can be used to set the message contents

  mqttClient.beginMessage("home/bedroom/switch1/available");

  mqttClient.print("online");

  mqttClient.endMessage();

  Serial.println();

  mqttClient.onMessage(onMqttMessage);

  mqttClient.subscribe("home/bedroom/switch1/set");

}

void onMqttMessage(int messageSize) {

  char command[3];

  int i = 0;

  if (mqttClient.messageTopic().equals("home/bedroom/switch1/set")) {

    while (mqttClient.available()) {

      command[i++] = (char)mqttClient.read();

    }

    if (command[0] == 'O' && command[1] == 'N') {

      Serial.println("on led");

      digitalWrite(buzzer, HIGH);

       drawText("  ON!  ");

    } else if (command[0] == 'O' && command[1] == 'F' && command[2] == 'F') {

      Serial.println("off led");

      digitalWrite(buzzer, LOW);

       drawText("  OFF!  ");

    }

  };

}

void loop() {

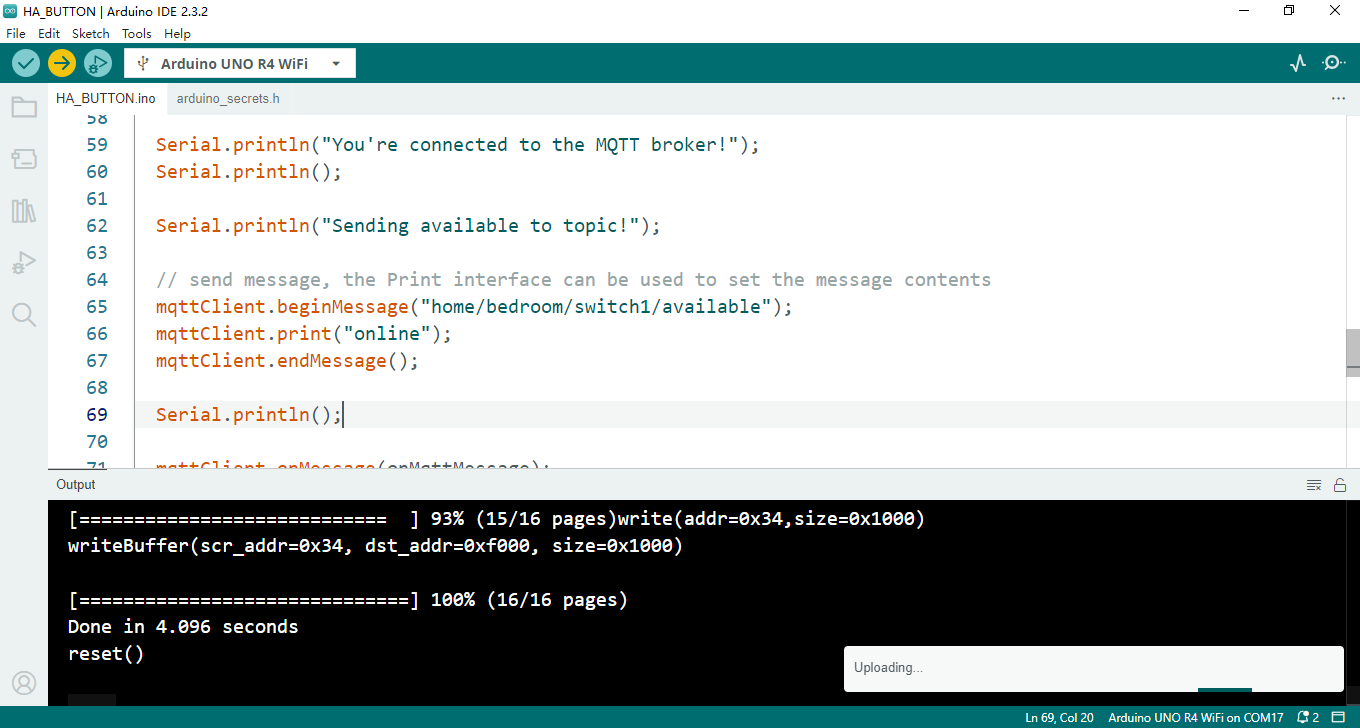
  // call poll() regularly to allow the library to send MQTT keep alives which

  // avoids being disconnected by the broker

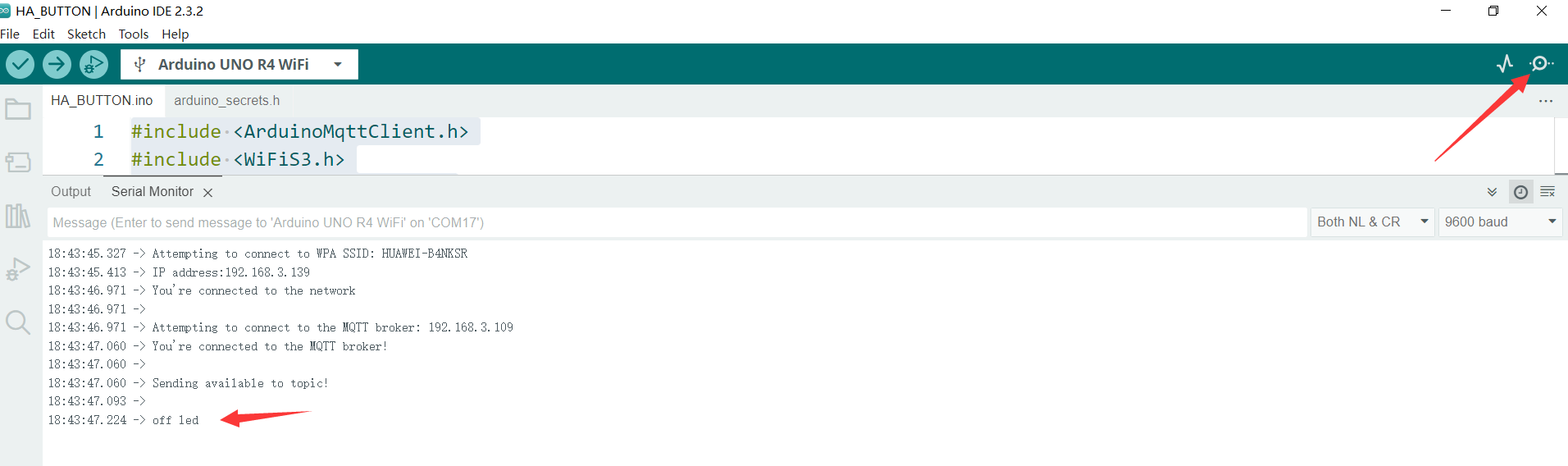
  mqttClient.poll();

}

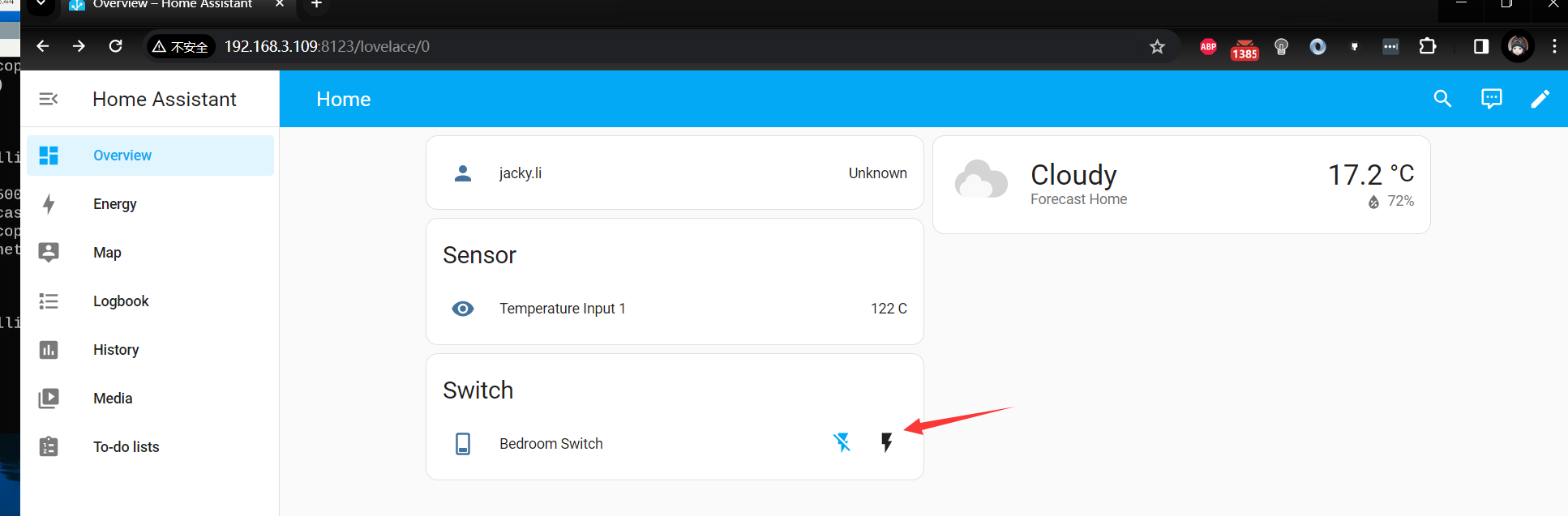
##### Compile & Upload sketch



Open serial monitor



and then open a browser, you can see the switch can be toggled.



Once you click the icon, it will send “ON” or “OFF” to arduino UNO R4, and it will be trigger like a switch, in this case, we connected a buzzer to Pin 9, so once you click the icon, the buzzer will be beeping.

#### Additional Notes:

Make sure to check the official Home Assistant documentation for any updates or specific instructions for your platform.

You can customize your Home Assistant setup by integrating various smart devices and services using the Home Assistant integrations.

Ensure your system has sufficient resources (CPU, RAM, storage) to run Home Assistant smoothly, especially if you plan to use it for complex automation tasks.

From now on, you have got a Home assistant system in hand, you can add all kinds of sensor to your home assistant center and manage all of them.

Let’s get into next chapter!