Bird Feeder

1. Introduction

Bird Feeder is a project that aims to build a self-contained system which integrates a Pet Feeder manufactured by Nooie which will dispense food for birds via donations through a livestream. The Bird Feeder is projected to be placed in the bird house of Kalisto Tierpark located in Kamp-Lintfort.

The Nooie Pet Feeder features a microcontroller board manufactured by Tuya. Tuya is an IoT and cloud development service provider with a varied range of products. The pet feeder can be controlled through Tuya's cloud development platform. While cloud development is useful, we would like to control the pet feeder locally for security and latency reasons.

To control the pet feeder locally, we will be using Home Assistant. Home Assistant is a Linux based open-source software generally used for smart home automation. We will be using a Raspberry Pi to install Home Assistant, which will be our main control hub.

Additionally, an e-paper display will be used to display last donation information to show people physically present that this service actually exists.

2. Materials

- Raspberry Pi 4.
- Nooie Pet Feeder.
- Micro SD card.
- SD card reader.
- Ethernet Cable.
- A smartphone which is compatible with Tuya Smart application.
- XIAO Esp32S3.
- XIAO eInk Expansion Board.
- 7.5in E-ink Display.
- Camera (Any).
- PC (Any).

2.1 Raspberry Pi 4

- The Raspberry Pi 4 is a powerful single-board computer that can power through many affordable electronics.
- Home Assistant OS is installed on the RaPi4 to act as a hub for the different components and as an MQTT Broker.

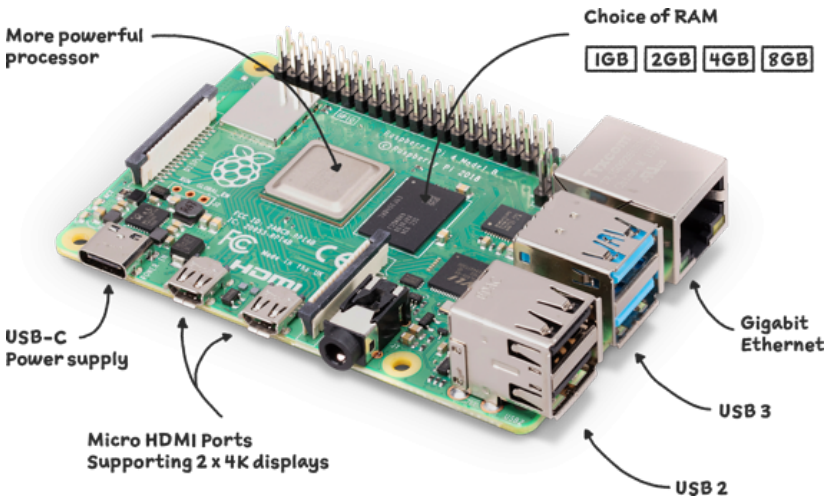

Fig. 1: Raspberry Pi 4

2.2 Xiao Esp32S3

- XIAO ESP32S3 is a tiny and cool device that combines the ESP32-S3R8 processor and support for both 2.4 GHz Wi-Fi and Bluetooth 5.0. The other ones have 8 MB PSRAM, 8 MB Flash, and an external SD card slot.

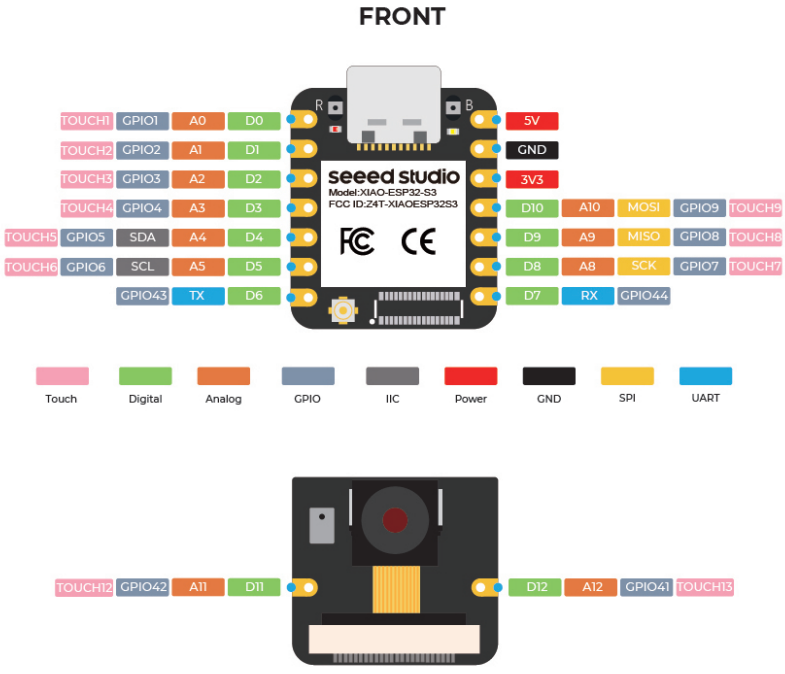

Fig. 2: XIAO Esp32s3 Pinout

- It is mainly used to control the e-paper display using the XIAO eInk Expansion Board

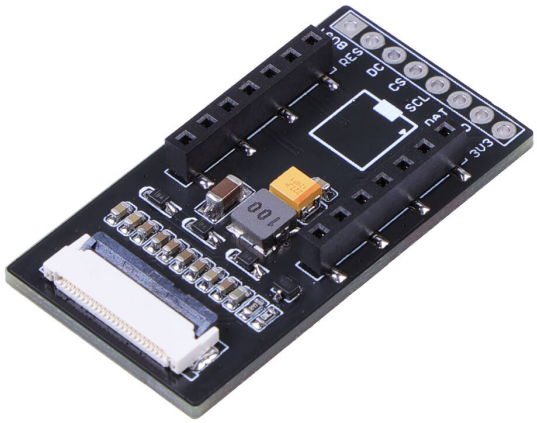

Fig. 3: XIAO eInk Expansion Board

2.3 7.5in E-ink Display

- An E-Ink display, also referred to as e-paper, is a display technology that is synonymous with its low power consumption and visual appearance—mimicking ink on paper.
- Used to display last donation information.

Fig. 4: 7.5" E-Ink Display

3. Home Assistant on RaPi

- System-Architecture
- Home Assistant Installation

Home Assistant Installation

This guide was written using Home Assistant version 2024.5.x.

- Credits go to Taycan.

Raspberry Pi Setup

1. Install Raspberry Pi imager https://www.raspberrypi.com/software/.

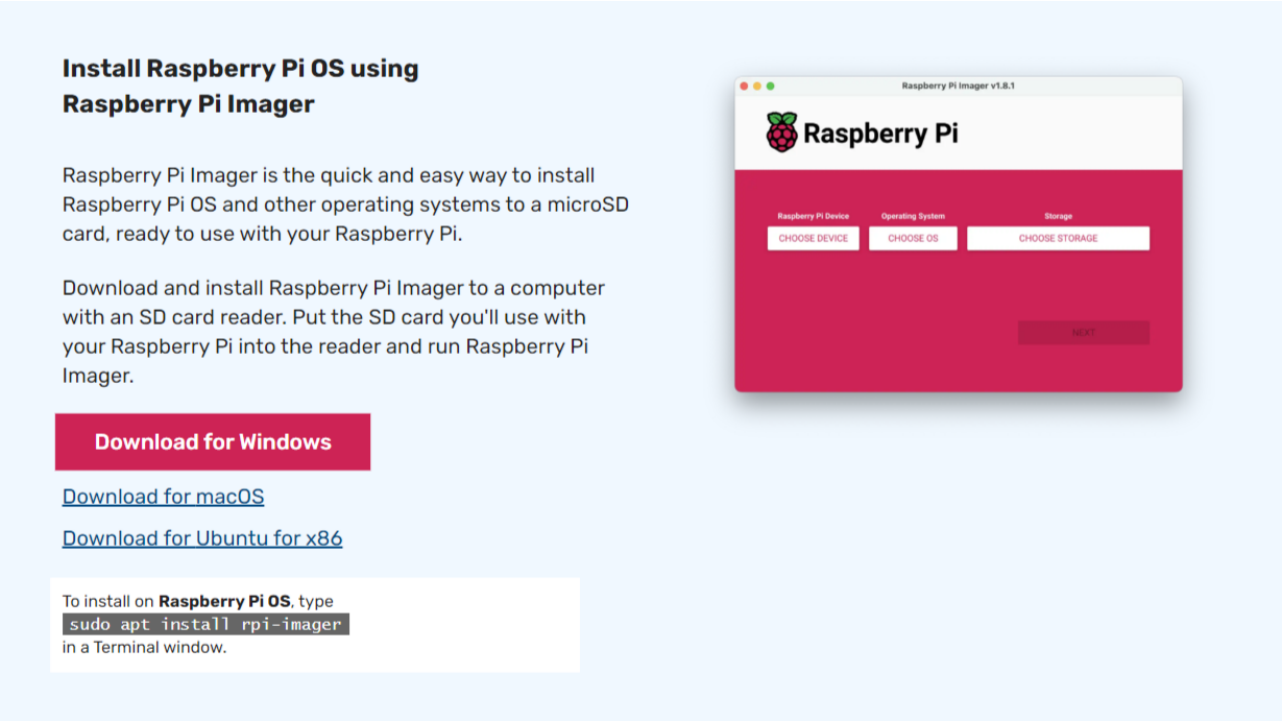

Fig. 5: Pick the operating system you are using

2. Insert the Micro SD card to the SD card reader.
3. Run the application.
4. Choose Device (Raspberry Pi 4 in this case).
5. Choose Operating System (“Other specific-purpose OS” → “Home assistants and home automation” → “Home Assistant” → “Home Assistant OS 12.x”).
6. Choose Storage.

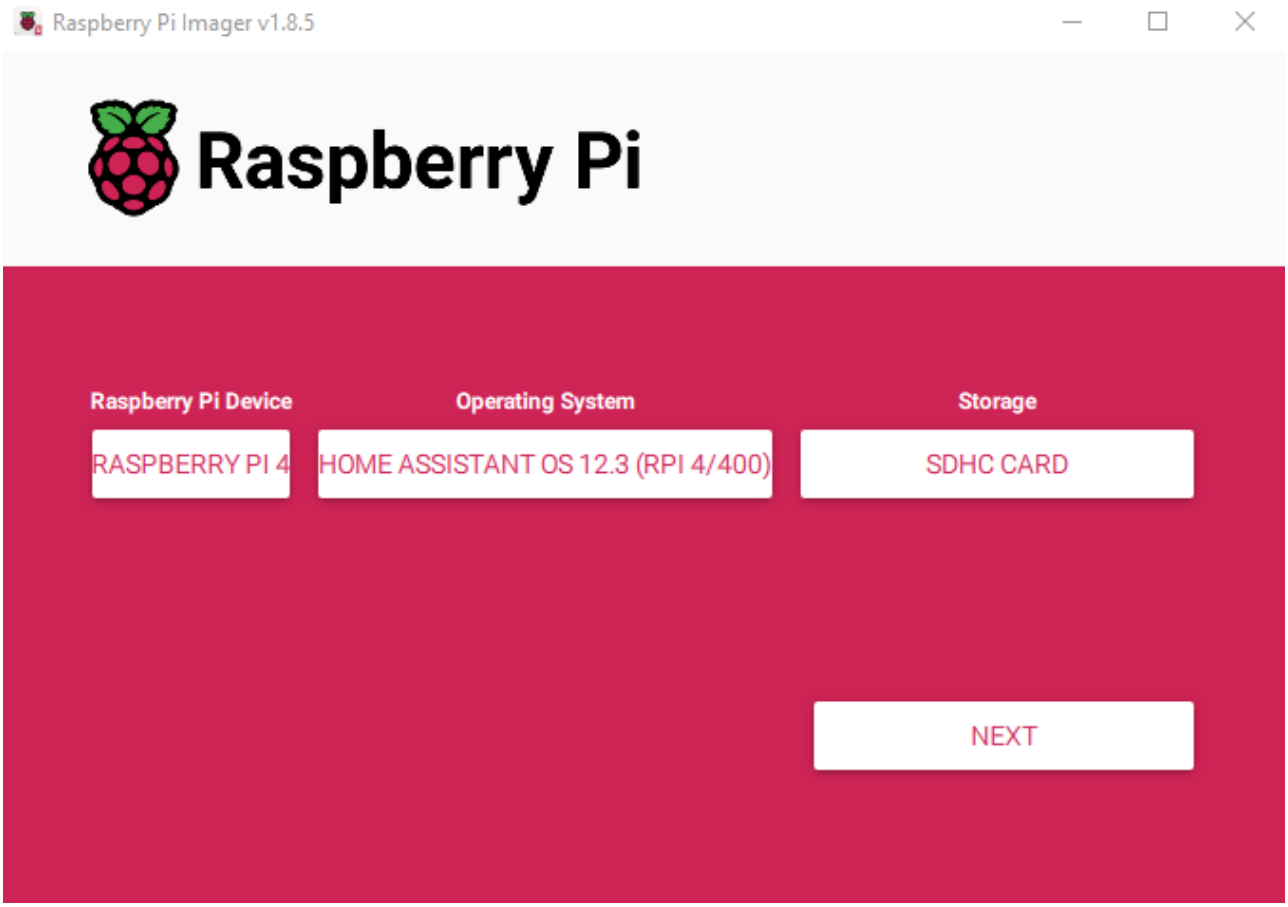

Fig. 6: Home Assistant Installation, In this case, Raspberry Pi 4

7. Click “Next” → “Yes”. (Ignore Microsoft Error messages).
8. Remove the Micro SD card.
9. Insert the Micro SD card into the Raspberry Pi’s Micro SD card slot.
10. Connect Raspberry Pi to power via the USB-C (or Micro-USB depending on the model) port.
11. Connect the Raspberry Pi to your network via the ethernet port on the Pi, make sure your computer is connected to the same network as well.
12. Home Assistant Installation happens automatically if the Micro SD card is inserted and the Raspberry Pi is powered on. This might take a few minutes. If you want to monitor the installation, connect the Raspberry Pi through its mini-HDMI port to a display.
13. Open a web explorer, navigate to http://homeassistant.local:8123/ and hit enter.
14. Wait for the setup to finish and click on “Create My Smart Home”.
15. Create a user using your preferred credentials.

4. Implementation

1. System Architecture.
2. Connecting ESP32 to e-ink display using the extension board. **(Display Component)**.
3. Installing ESP-Home on *HA*.
4. Connect **Display Component** to ESP-Home.
5. Setting up MQTT Broker on *HA*.
6. Rewriting the **Display Component** code from C++ to ESP-Home yaml syntax.
7. Setting up Frontend website.
8. Configuring website to use MQTT QoS-2.
9. Configuring Nooie-pet feeder on HA using LocalTuya.
10. Creating Automation function for pet feeder to trigger on MQTT Message.

4.1 System Architecture

- Raspberry Pi:
  - Home Assistant is flashed on a Raspberry Pi.
  - Local Tuya and MQTT Broker are installed and configured on *HA*.
  - Pet Feeder is connected to Local Tuya using LAN/WAN.
  - Both Pet Feeder and **Display Component** trigger on MQTT Messages.
- A PC Acting as a server to hold services:
  - OBS/Streaming connected to a camera in the exhibit.
  - A full-stack web application that hosts the stream and acts as an interactive environment for people to watch and potentially donate.
  - The front-end side sends a MQTT message on donation to trigger pet feeder and **Display Component**.

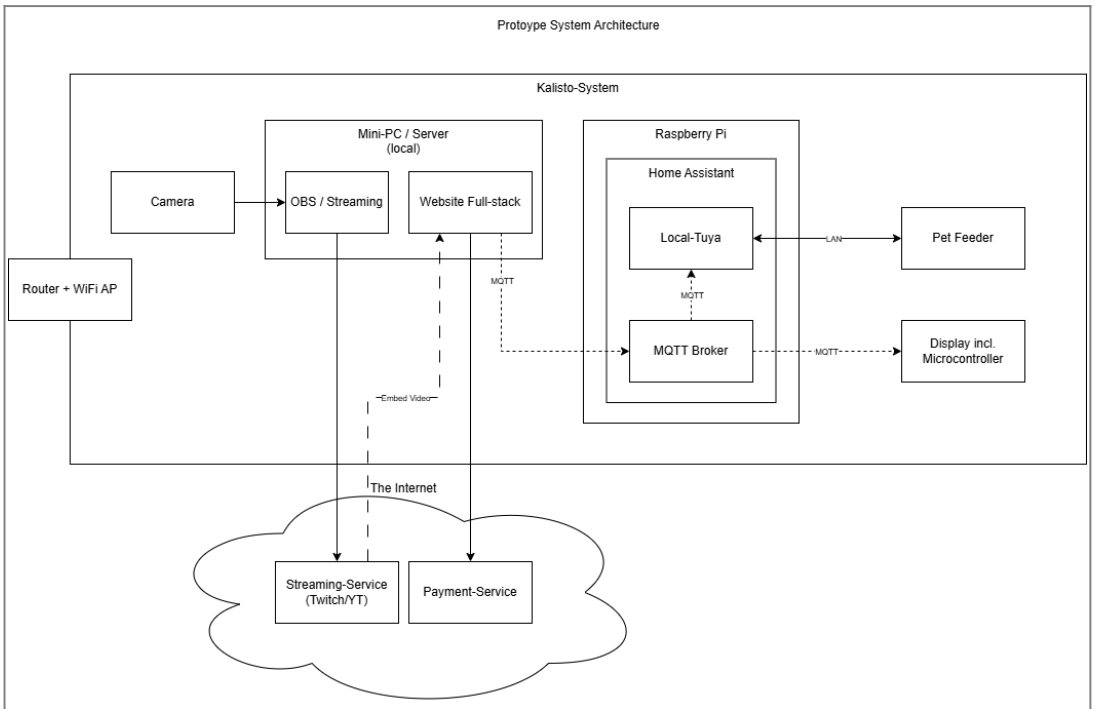

Diagram 1: System Architecture

4.2 Connecting ESP32 to e-ink display using the extension board.

- Connecting the different parts was very simple. The microcontroller was connected to the extension board seamlessly as they are already compatible.
- Display was connected to the extension board using its 24-pin cable. Warning, the cable is very fragile 

Fig. 7: Display Component

- The following code file was implemented using **Seeed Studio**'s wiki/documentation page.
- The e-ink display needs to be refreshed fully or partially between data changes, otherwise the text/image will not show properly.
- The trickiest part was figuring out which pins to use for these pins: `BUSY, RES, DC, CS`.
- The display using bitmaps for... displaying text/images.
- A bitmap uses a grid of pixels to display text/images. Each pixel in the bitmap corresponds to a pixel on the e-ink display.
- A Converter can used to convert normal images to cpp bitmap files.

display_code.ino

```
//Most Important Import
#include <SPI.h>

#include "Display_EPD_W21_spi.h"
#include "Display_EPD_W21.h"
#include "frame.h"
#include "GUI_Paint.h"
#include "font.h"
```

```
#if 1
unsigned char BlackImage[EPD_ARRAY]; //Define canvas space
#endif

//Most Important Part
void setup() {
  pinMode(D5, INPUT); //BUSY
  pinMode(D0, OUTPUT); //RES
  pinMode(D3, OUTPUT); //DC
  pinMode(D1, OUTPUT); //CS

  //SPI
  SPI.beginTransaction(SPISettings(10000000, MSBFIRST, SPI_MODE0));
  SPI.begin();

  Serial.begin(9600);
  Serial.print(111);
}

//Tips//
/*
1.Flickering is normal when EPD is performing a full screen update to
clear ghosting from the previous image so to ensure better clarity and
legibility for the new image.
2.There will be no flicker when EPD performs a partial refresh.
3.Please make sue that EPD enters sleep mode when refresh is completed
and always leave the sleep mode command. Otherwise, this may result in
a reduced lifespan of EPD.
4.Please refrain from inserting EPD to the FPC socket or unplugging it
when the MCU is being powered to prevent potential damage.)
5.Re-initialization is required for every full screen update.
6.When porting the program, set the BUSY pin to input mode and other
pins to output mode.
*/

void loop() {
  Serial.print(111);

  #if 1 //Partial refresh demostration.

  EPD_Init_Fast(); //Full screen refresh initialization.
  EPD_WhiteScreen_White(); //Clear screen function.
  EPD_DeepSleep(); //Enter the sleep mode and please do not delete it,
otherwise it will reduce the lifespan of the screen.
  delay(2000); //Delay for 2s.
  //Partial refresh demo support displaying a clock at 5 locations with
00:00.  If you need to perform partial refresh more than 5 locations,
please use the feature of using partial refresh at the full screen
demo.
  //After 5 partial refreshes, implement a full screen refresh to clear
```

```
the ghosting caused by partial refreshes.
  ////////////////////Partial refresh
time/////////////////////////////////////
  Paint_NewImage(BlackImage, EPD_WIDTH, EPD_HEIGHT, 0, WHITE); //Set
canvas parameters, GUI image rotation, please change 0 to 0/90/180/270.
  Paint_SelectImage(BlackImage); //Select current settings.
  EPD_Init(); //Full screen refresh initialization.
  Paint_Clear(WHITE); //Clear canvas.
  const char * Message = "Welcome to Kalisto";
  Paint_DrawString_EN(400 - (((strlen(Message) * 32) / 2)), 240 - (64 /
2), Message, & Font64, WHITE, BLACK);
  EPD_Display(BlackImage); //Display GUI image.
  EPD_DeepSleep(); //EPD_DeepSleep,Sleep instruction is necessary,
please do not delete!!!
  delay(20000); //Delay for 2s.

  // Full screen update clear the screen.
  EPD_Init(); //Full screen refresh initialization.
  EPD_WhiteScreen_White(); //Clear screen function.
  EPD_DeepSleep(); //Enter the sleep mode and please do not delete it,
otherwise it will reduce the lifespan of the screen.
  delay(2000); //Delay for 2s.

  #endif

  delay(300000); // The program stops here
}
```

4.3 Installing ESP-Home on HA.

- **ESP-Home** enables advanced functionality on ESP devices without deep programming knowledge. Despite that fact, it has a specific syntax to follow and requires some learning. It acts a hub for all connected ESP devices.
- Following the **ESP-Home** Documentation website is essential for installation and general knowledge.
- To install **ESP-Home** on *HA*, this page is far better than any steps I can write.

4.4 Setting up MQTT Broker on HA.

- MQTT originally stood for MQ Telemetry Transport. It is an extremely lightweight publish/subscribe messaging transport protocol on top of TCP/IP designed for machine-to-machine or Internet of Things connectivity. It's a very powerful protocol for transmission in smart home devices.
- MQTT will be used for communication between the different components.
- A MQTT Broker called Mosquitto can be installed to **HA** as an add-on. This page contains detailed instructions for installation and configuration.

4.5 Connect Display Component to ESP-Home.

- Adding an ESP component is very simple and requires simple steps to be achieved.
- 1. Click on “New Device”

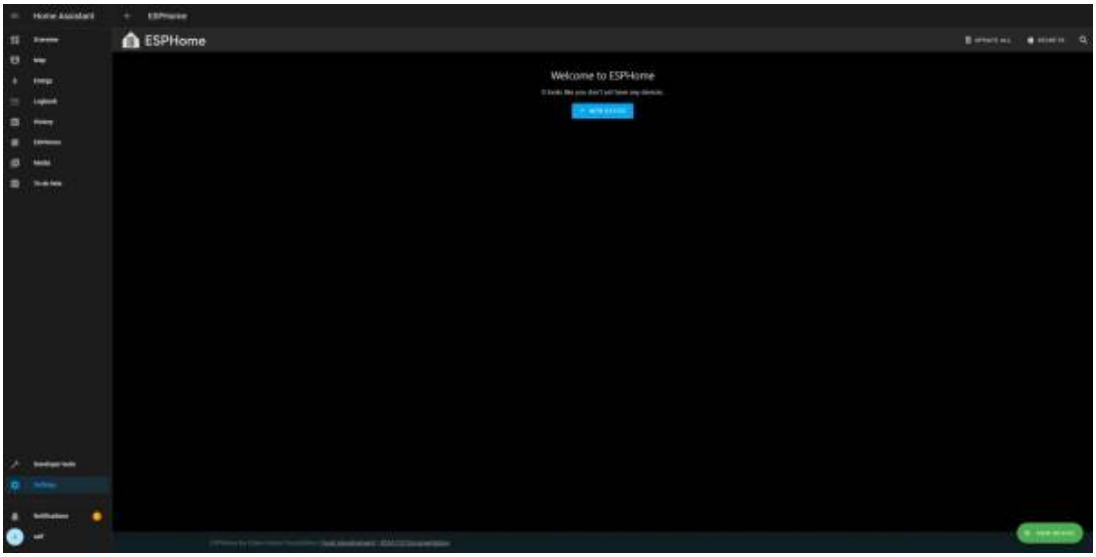

Fig. 8

- 2. Click “Continue”

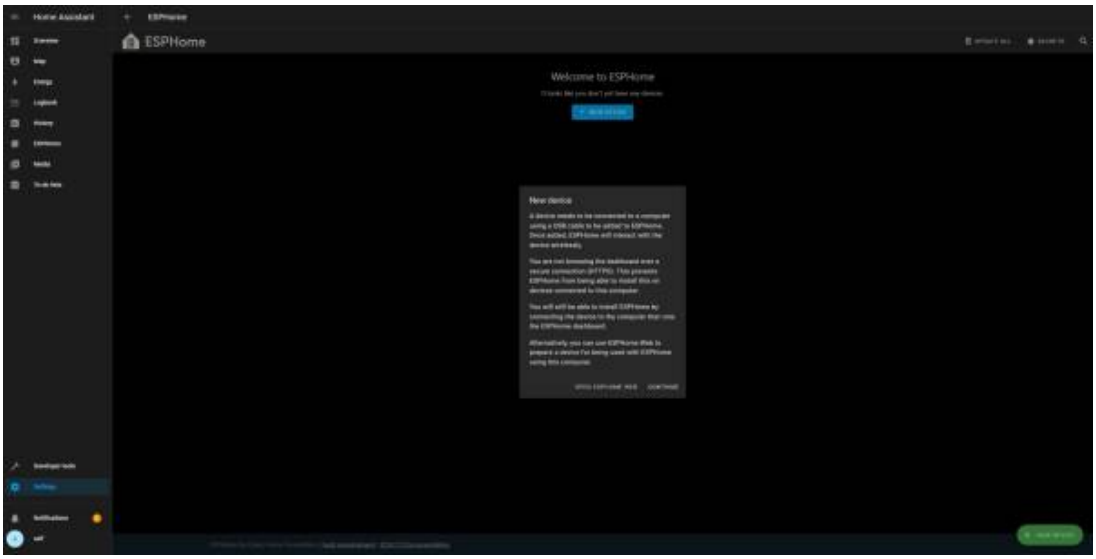

Fig. 9

- 3. Enter a name for the device and give the Wifi SSID and password for the device to establish connection wirelessly. The click “Next”

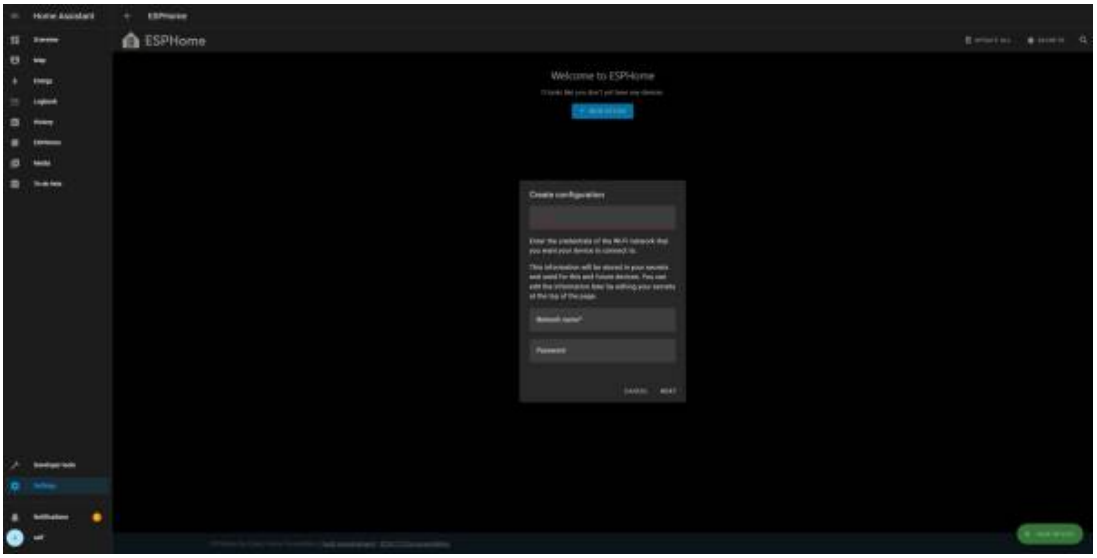

Fig. 10

- 4. Make sure “Use recommended settings” choose the device type, in our case it is “ESP32-S3”.

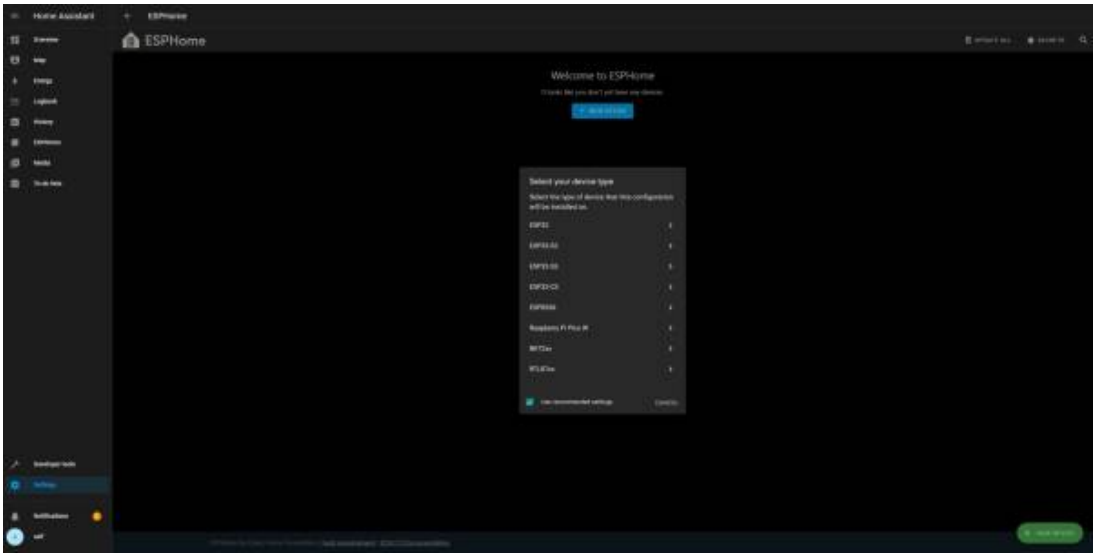

Fig. 11

- 5. Copy the encryption key safely for future use then click “Install”.

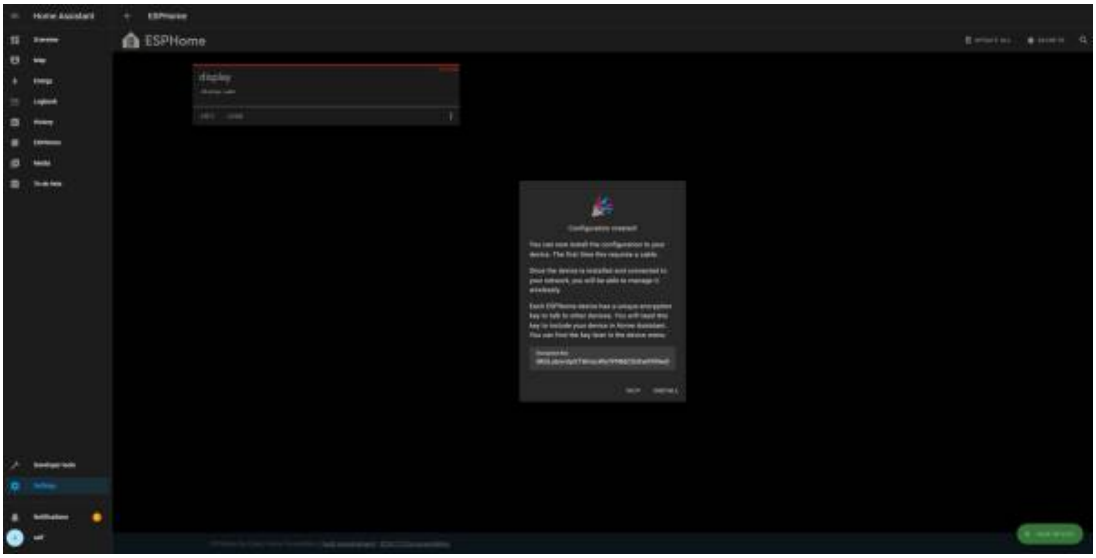

Fig. 12

- 6. This depends on how the device is connected. For the first time installation it is better to “Plug into the computer running ESPHome Dashboard”.

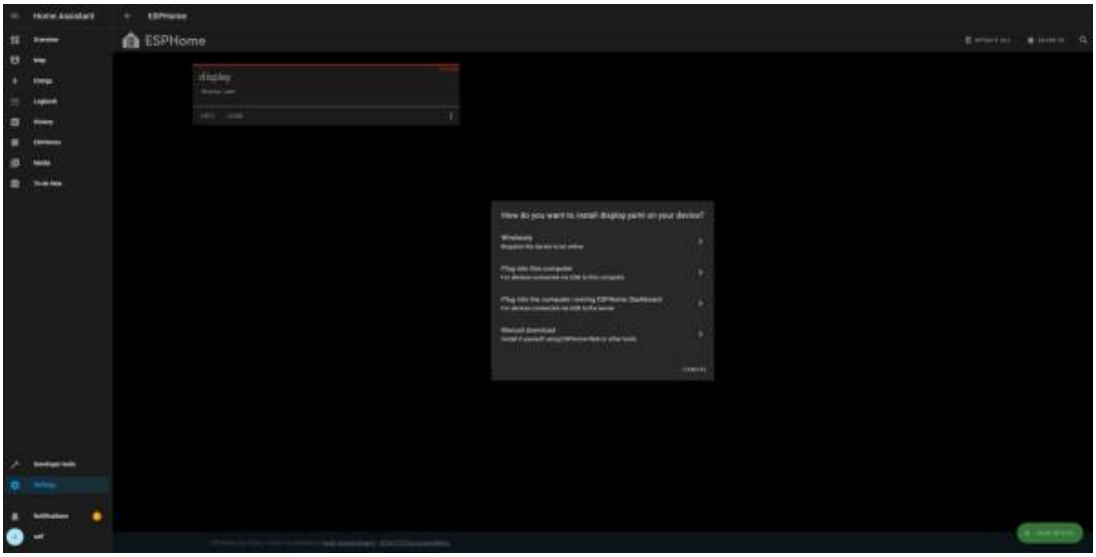

Fig. 13

- 7. Choose the correct port. In this case it is only one.

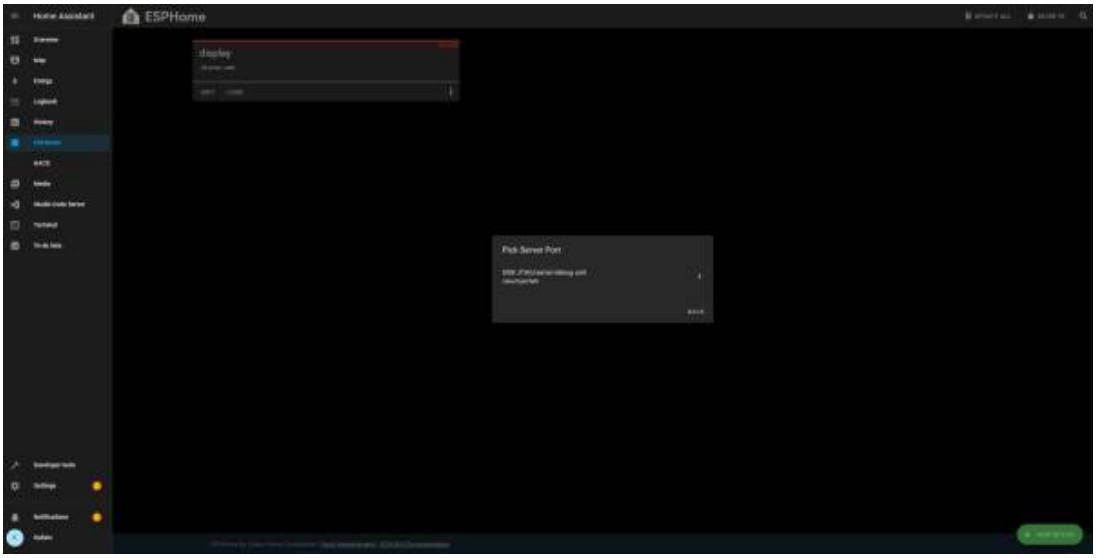

Fig. 14

- 8. Wait for the first installation to pass. You can see if it was successful or not but seeing whether the device connected to the wifi connection or not.

4.6 Rewriting the Display Component code from C++ to ESP-Home yaml syntax.

- The ESPHome documentation proven to be the best way to fulfil this step.
  - ESPHome Core Configuration , SPI, MQTT, text_sensor and Display were the most used pages in the documentation to reach this code.
- ESPHome Configuration: specifies the name of ESPHome project and on-boot actions.
- Configuration of ESP32 Board and Framework: In this, the ESP32 board and framework used are specified.
- Logging and API: Switches on logging and the Home Assistant API—all encrypted.
- WiFi Configuration: Provide WiFi credentials and setup fallback hotspot.
- MQTT Integration: Provide details of the MQTT broker to connect for communication.
- Display Setup: Specify your e-ink display model, pins, and update intervals.
- Text Sensor: Subscribes to MQTT topics to show messages on the e-ink display.

display_code.yaml

```
esphome:
  name: display
  friendly_name: display
  on_boot:
      priority: 200.0
      then:
        - component.update: eink_display
        - delay: 5s
        - logger.log: "Initial sensor data received: Refreshing
display..."
        - script.execute: update_screen

esp32:
```

```
  board: esp32-s3-devkitc-1
  framework:
    type: arduino

# Enable logging
logger:

# Enable Home Assistant API
api:
  encryption:
    key: "7QLJzR30itSBn30kootHbIE6FI8jmUtAl7/fcytRxis="

ota:
  platform: esphome
  password: "81ceca8aa7f29a76601df5539733d32f"

wifi:
  ssid: !secret wifi_ssid
  password: !secret wifi_password

  # Enable fallback hotspot (captive portal) in case wifi connection
fails
  ap:
    ssid: "Display Fallback Hotspot"
    password: "WgV5TWXpXcpT"

captive_portal:

script:
  - id: update_screen
    then:
      - component.update: eink_display

font:
  - file: "gfonts://Roboto"
    id: roboto_64
    size: 64

spi:
  clk_pin: GPIO7
  mosi_pin: GPIO9

mqtt:
  broker: hafeeder.local
  port: 1883
  username: "mqtt"
  password: "1234"

text_sensor:
  - platform: mqtt_subscribe
    name: "Data from topic"
```

```
    id: mytext
    topic: birdfeeder
    qos: 2
    on_value:
      then:
        - script.execute: update_screen

display:
  - platform: waveshare_epaper
    model: 7.50in-bV3
    id: eink_display
    cs_pin: GPIO2
    dc_pin: GPIO4
    busy_pin: GPIO6
    reset_pin: GPIO1
    update_interval: never
    lambda: |-
      it.printf(400, 240, id(roboto_64), TextAlign::TOP_CENTER,
id(mytext).state.c_str());
```

4.7 Setting up Frontend website.

- I built a simple react frontend application for sake of testing and showing the viability of the whole concept.
- The application can be found on a Github Repo

Fig. 15

4.8 Configuring Nooie-pet feeder on HA using LocalTuya.

- Credits go to Taycan.

HACS Installation

Home Assistant Community Store (HACS) is a third-party download manager for Home Assistant which contains various custom integrations. We need to install LocalTuya integration through HACS to locally control the pet feeder. This guide was written using HACS version 1.34.0.

1. Go to your user profile and enable “Advanced mode”.

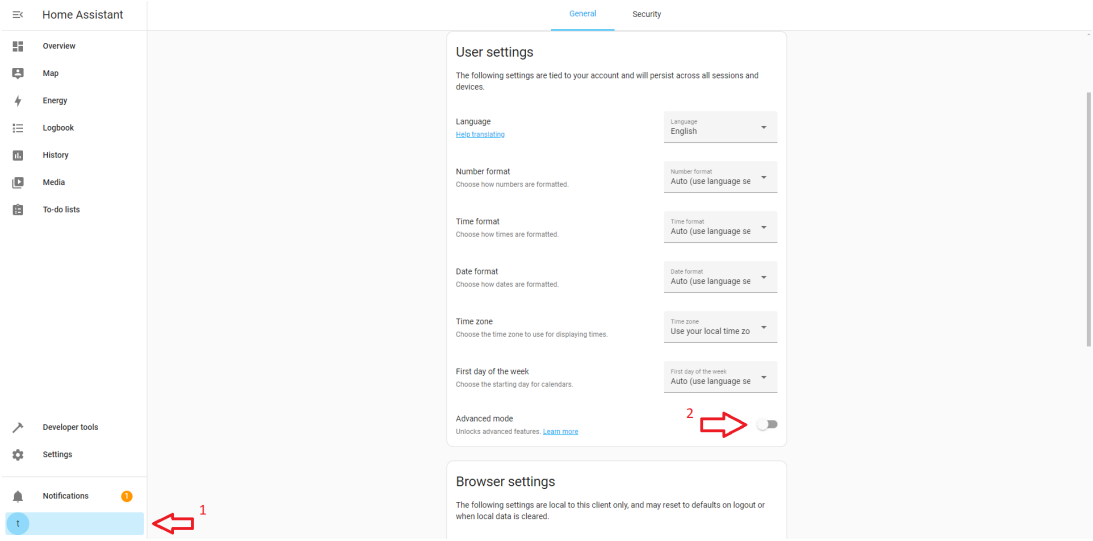

Fig. 16: User profile is found on the bottom right.

1. Go “Settings” → “Add-ons” → “Add-on store” → search “SSH” → Install “Terminal & SSH”.
2. After the installation is complete, enable “Show in sidebar”

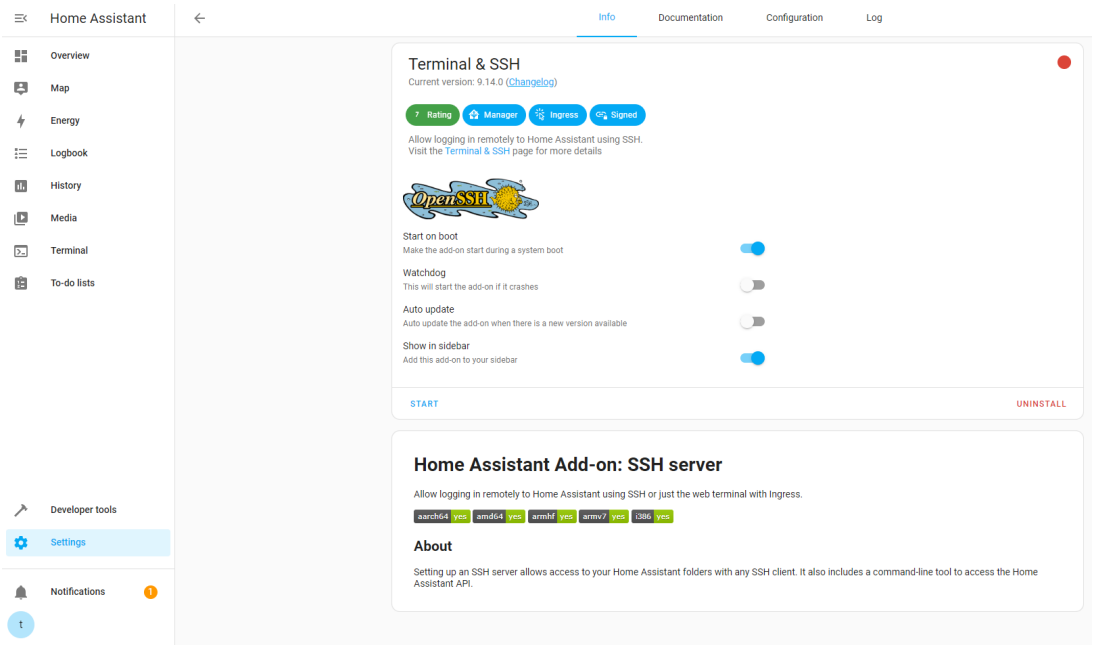

Fig. 17: SSH Add-on options after installation

1. Navigate to “Terminal” on the sidebar, if it appears not to be running or if you are receiving error codes, refresh the instance.
2. Copy and paste (CTRL+SHIFT+V to paste) the following command to the terminal and hit Enter.

This will initiate the installation of HACS.

```
wget -O - https://get.hacs.xyz | bash -
```

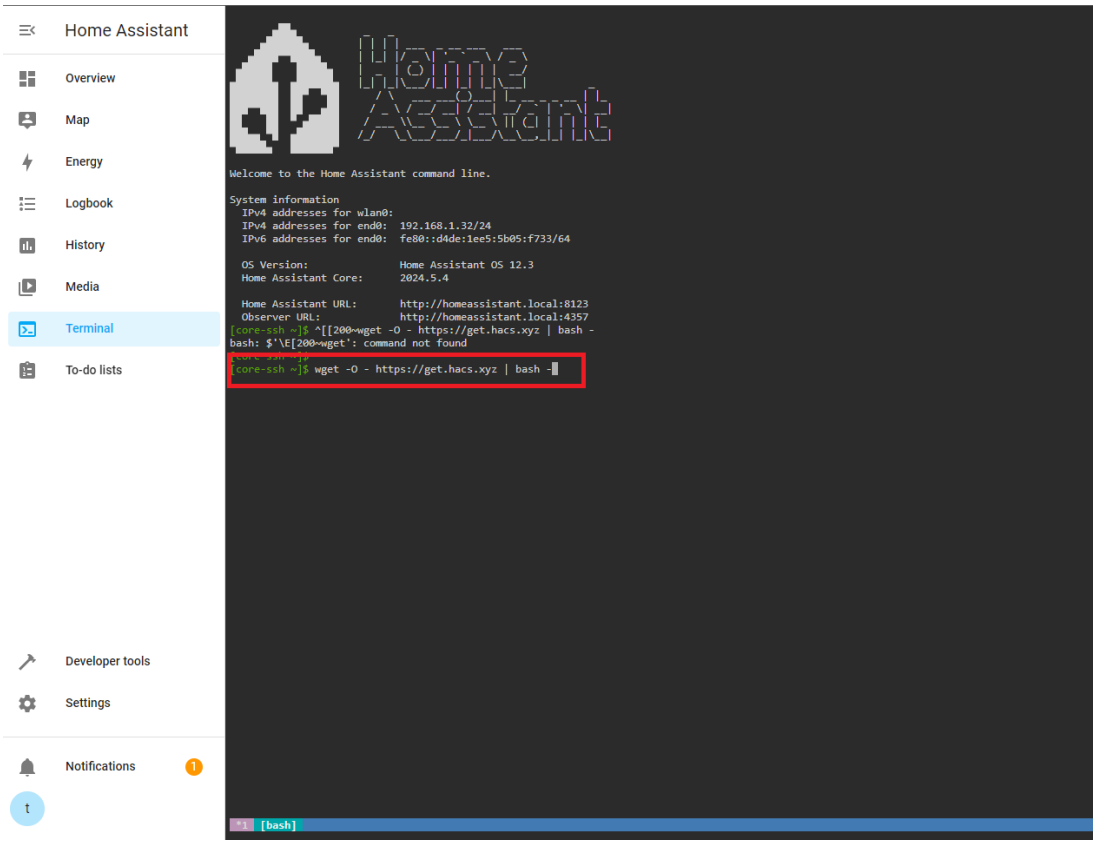

Fig. 18: HA SSH

1. After the installation is completed, restart the home assistant (“Settings” → three dots on the top right → “Restart Home Assistant” → “Restart Home Assistant”).
2. Home Assistant should automatically restart, if the browser crashes, refresh the browser page to access Home Assistant again.
3. Navigate to “Settings” → “Devices & Services” → Click “Add Integration” on the bottom right corner.
4. Type “HACS” on the search bar, click on the result.

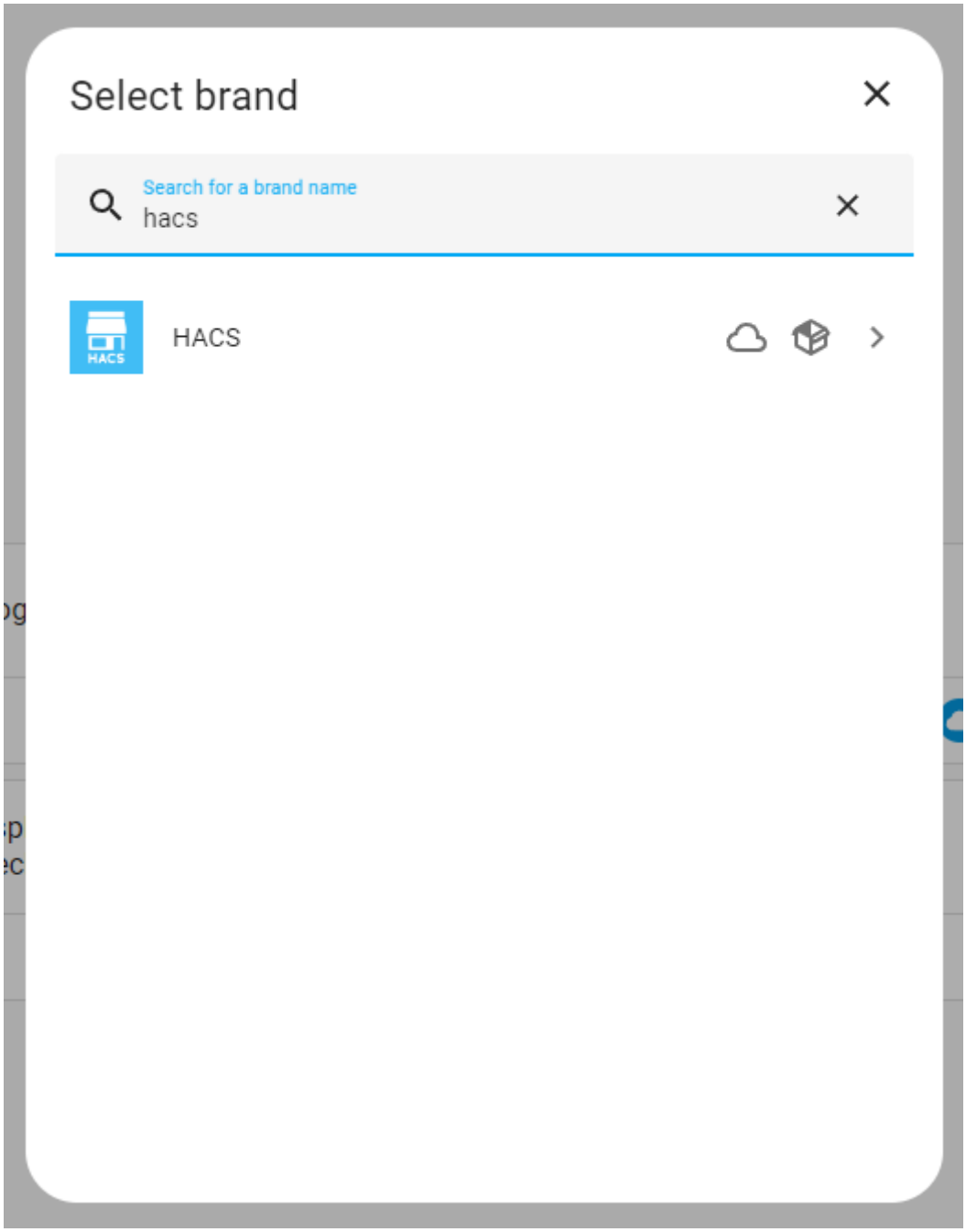

Fig. 19

1. Check all but the last checkbox and submit.

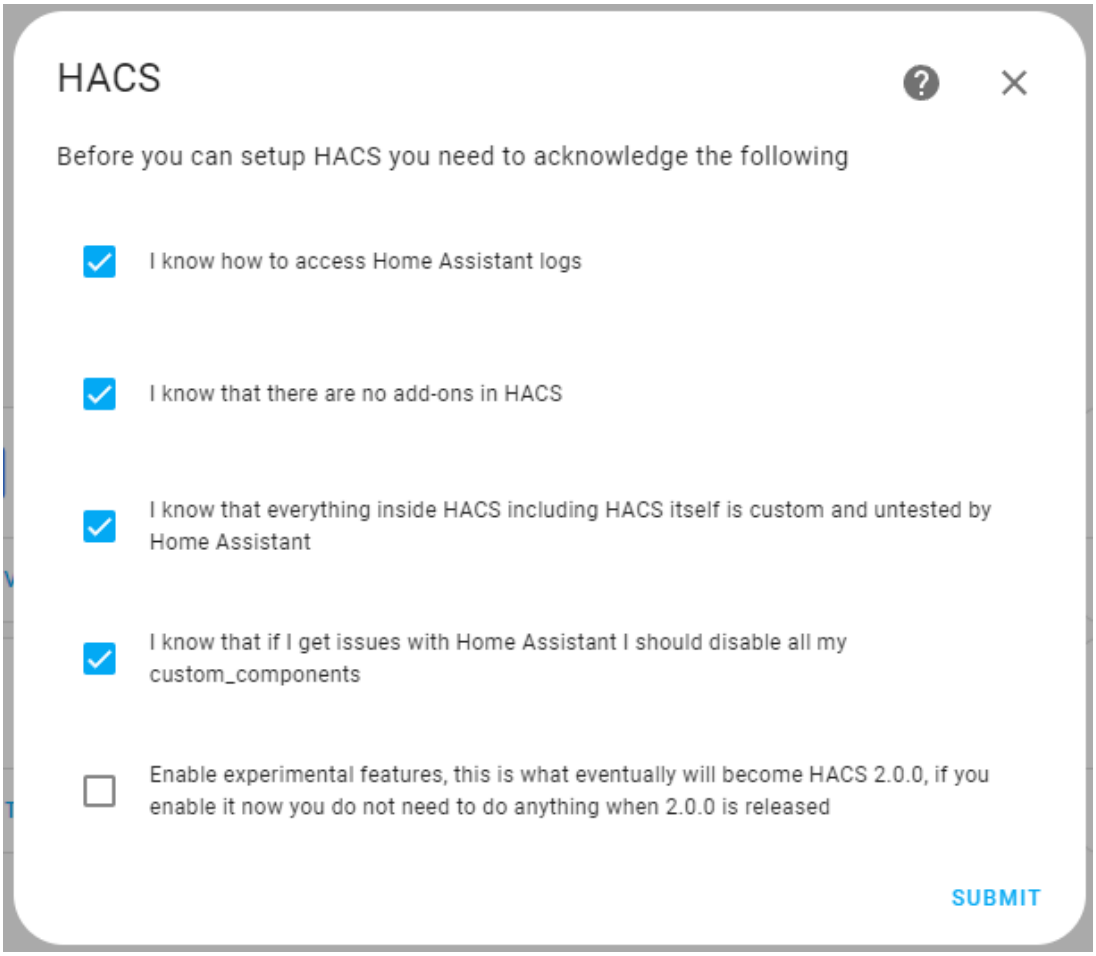

Fig. 20

1. Create a github account if you don’t have one.
2. HACS will ask you to activate device through your github account. Follow the instructions provided and link HACS with your github account.

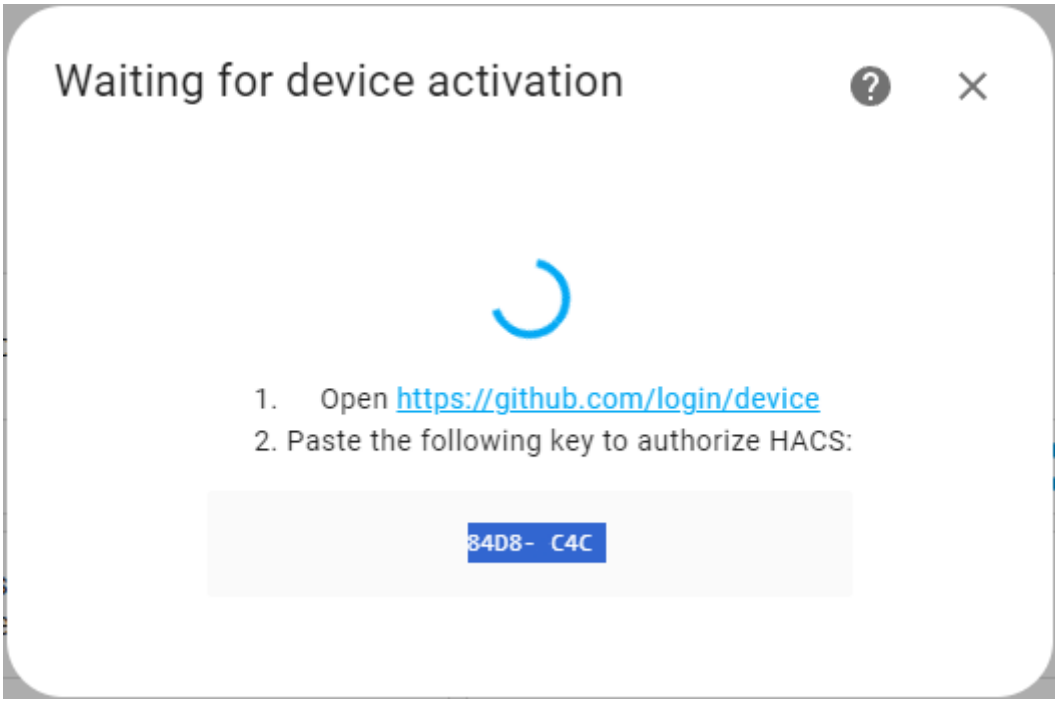

Fig. 21: Copy highlighted code and click on the github link

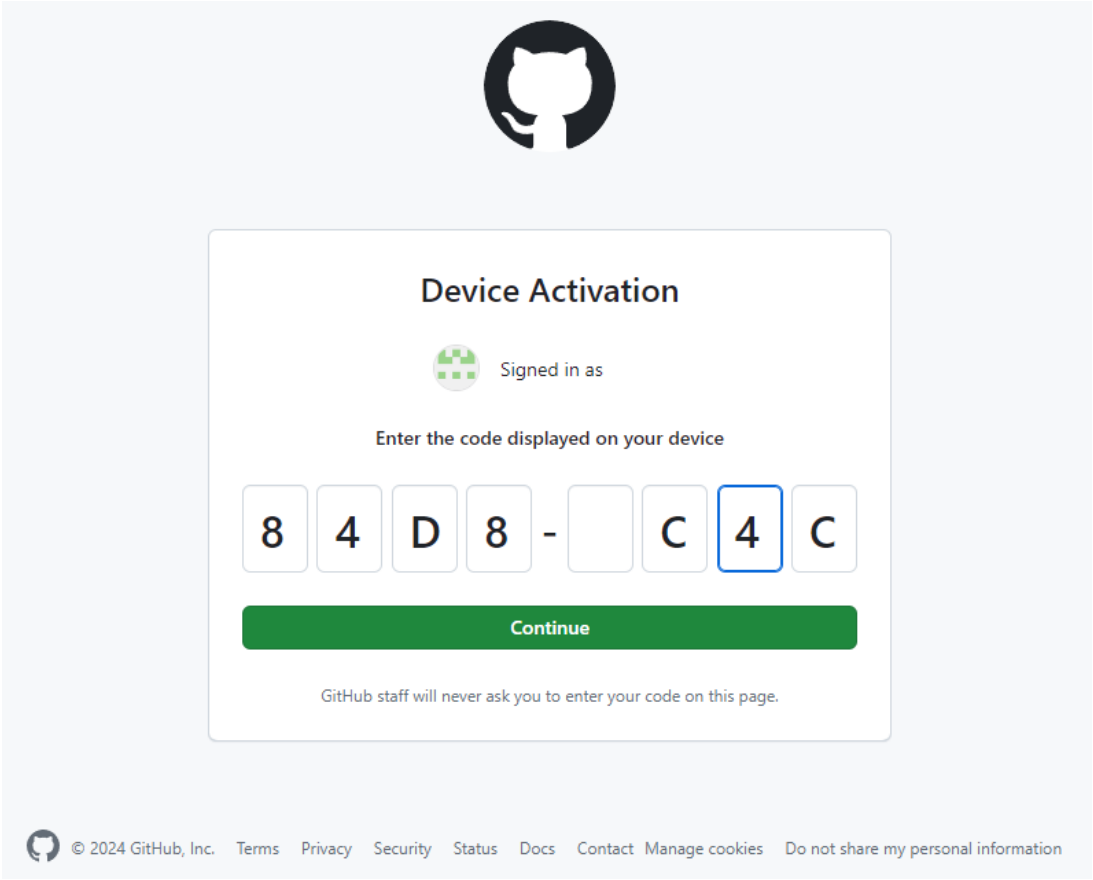

Fig. 22: Paste the code here.

1. Click on “Authorize” and exit github.
2. If everything is set correctly you should have installed HACS on your Home Assistant. Do not pick an area on the pop-up and click on “finish”.
3. Now HACS should show up on the sidebar of Home Assistant UI.

LocalTuya integration Installation

LocalTuya integration is what we need to control the pet feeder locally. This guide was written for LocalTuya version 5.2.1.

1. On the HACS tab, click on “Integrations”.

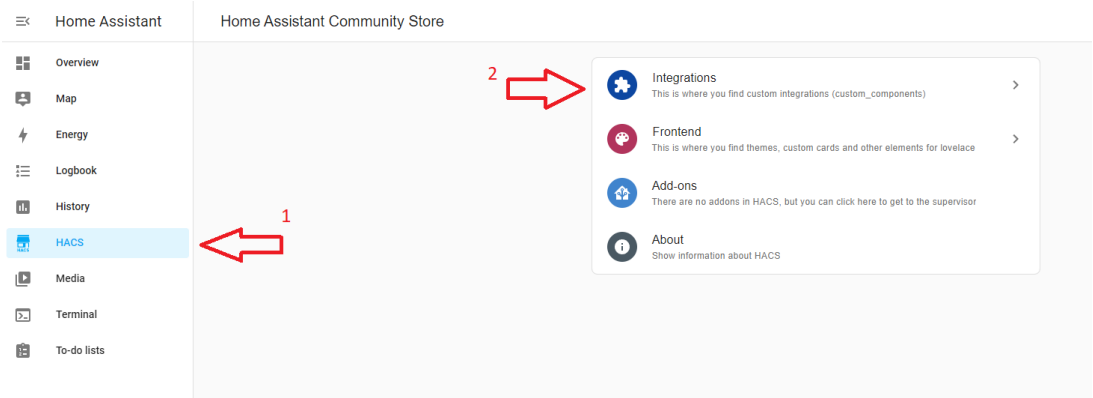

Fig. 23

1. Click on “Explore & Download Repositories” on the bottom right, on the pop-up search, type

“LocalTuya”. Click on the result.

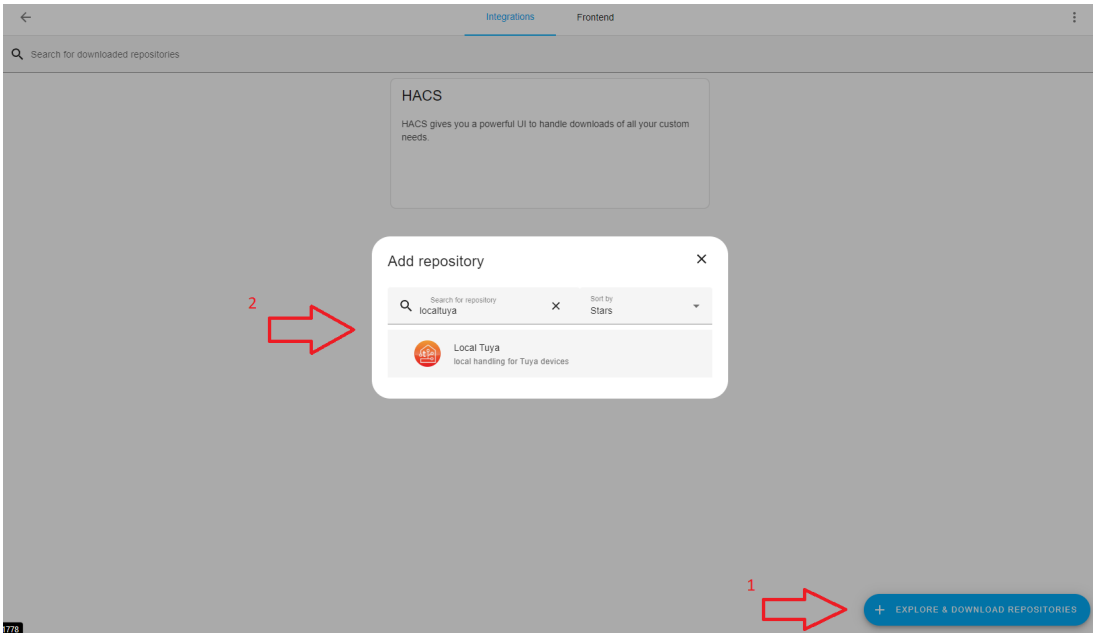

Fig. 24

1. Click on “Download” on the bottom right. Click on “Download” again on the pop-up.

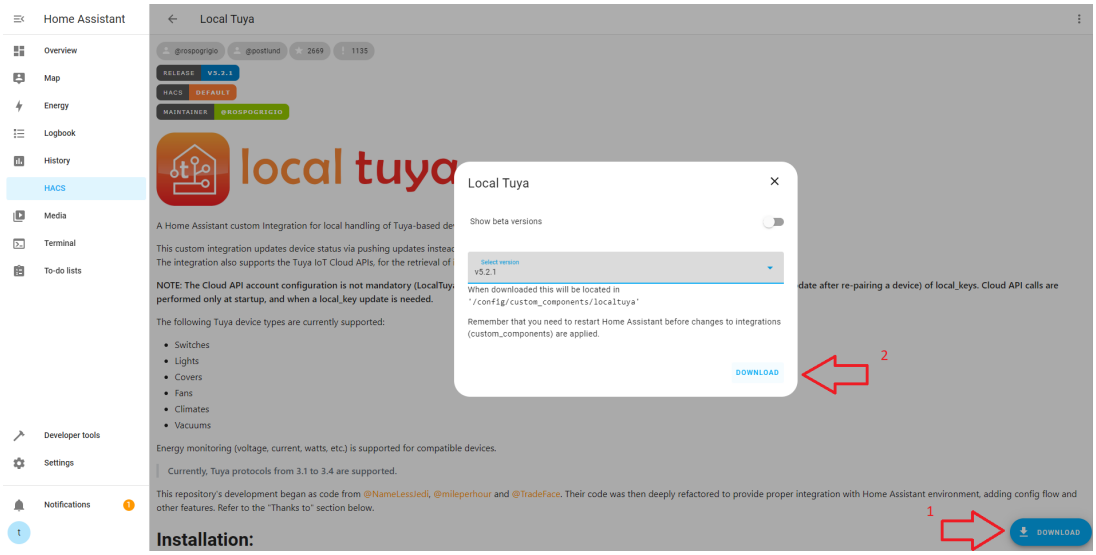

Fig. 25

Adding Pet Feeder to Home Assistant

1. Download the “Tuya Smart” app to your smartphone (version 5.13.0 was used for this guide).
2. Press both buttons on the pet feeder at the same time.
3. Run the “Tuya Smart” app, press “Add Device”, the Pet feeder should appear automatically, turn on your Bluetooth if it doesn’t. Insert the wireless credentials and press login. **Make sure the pet feeder and the Home Assistant are in the same network**.
4. Navigate to Tuya IoT development platform and create an account.

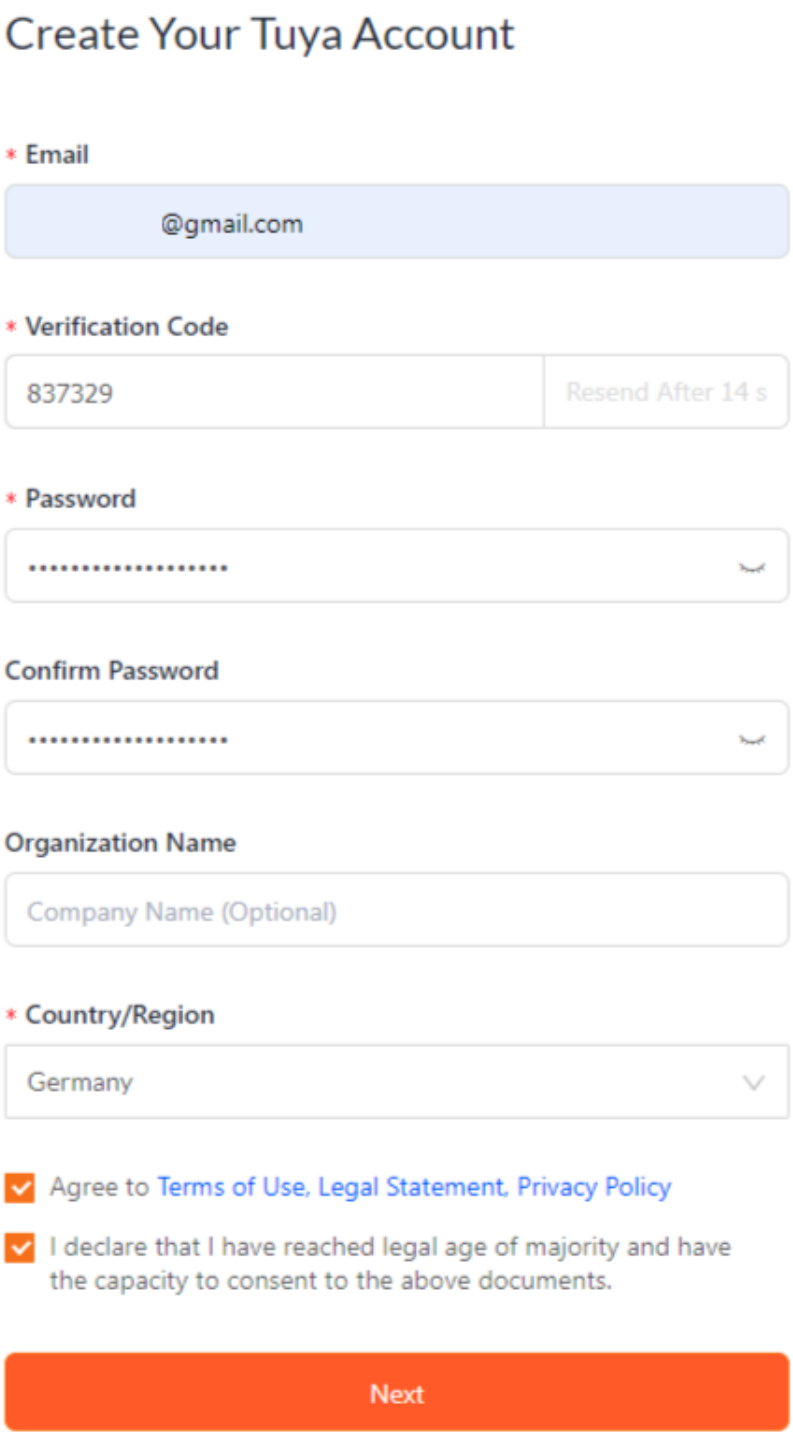

Fig. 26

5. Once the account is created, navigate to the developer platform on the top right corner of the page you were redirected to.
6. Skip the tutorials, hover the cursor over “Cloud” on the sidebar and click on “Development”.

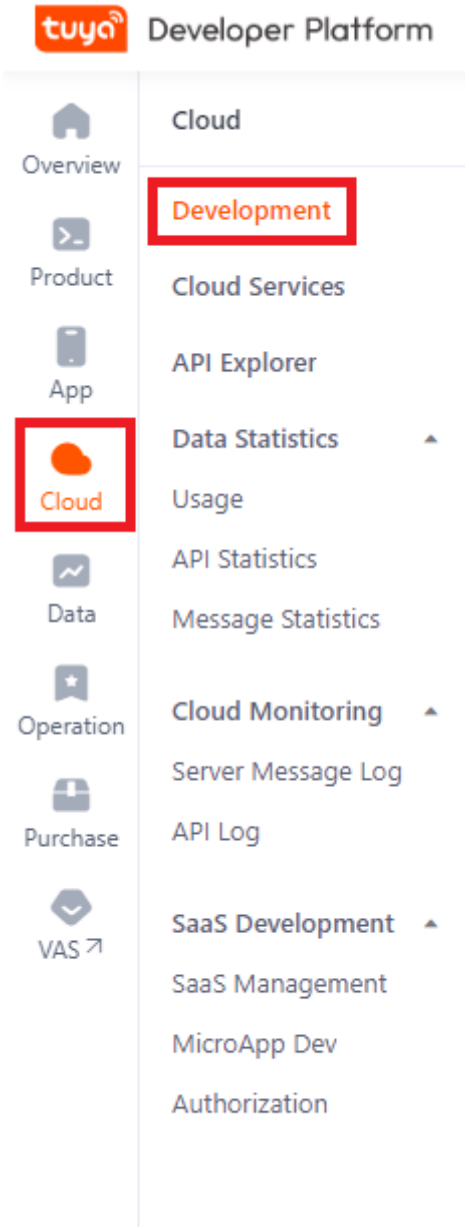

Fig. 27

7. Click on “Create Cloud Project“ on the Development page. Fill in the project details. Pick “Custom” Development method and “Central European Data Center”, pick the rest as you see fit.

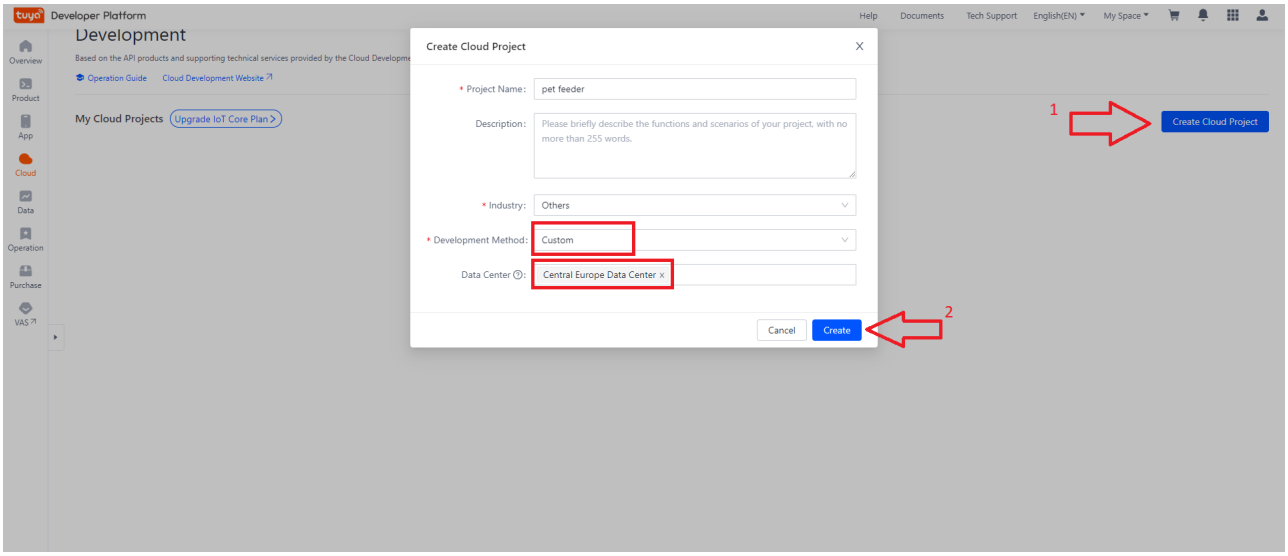

Fig. 28

8. In the next step, add “Smart Home Basic Service” and ”[Deprecate]Device Log Query“ to the selected API services. Click on “Authorize”.

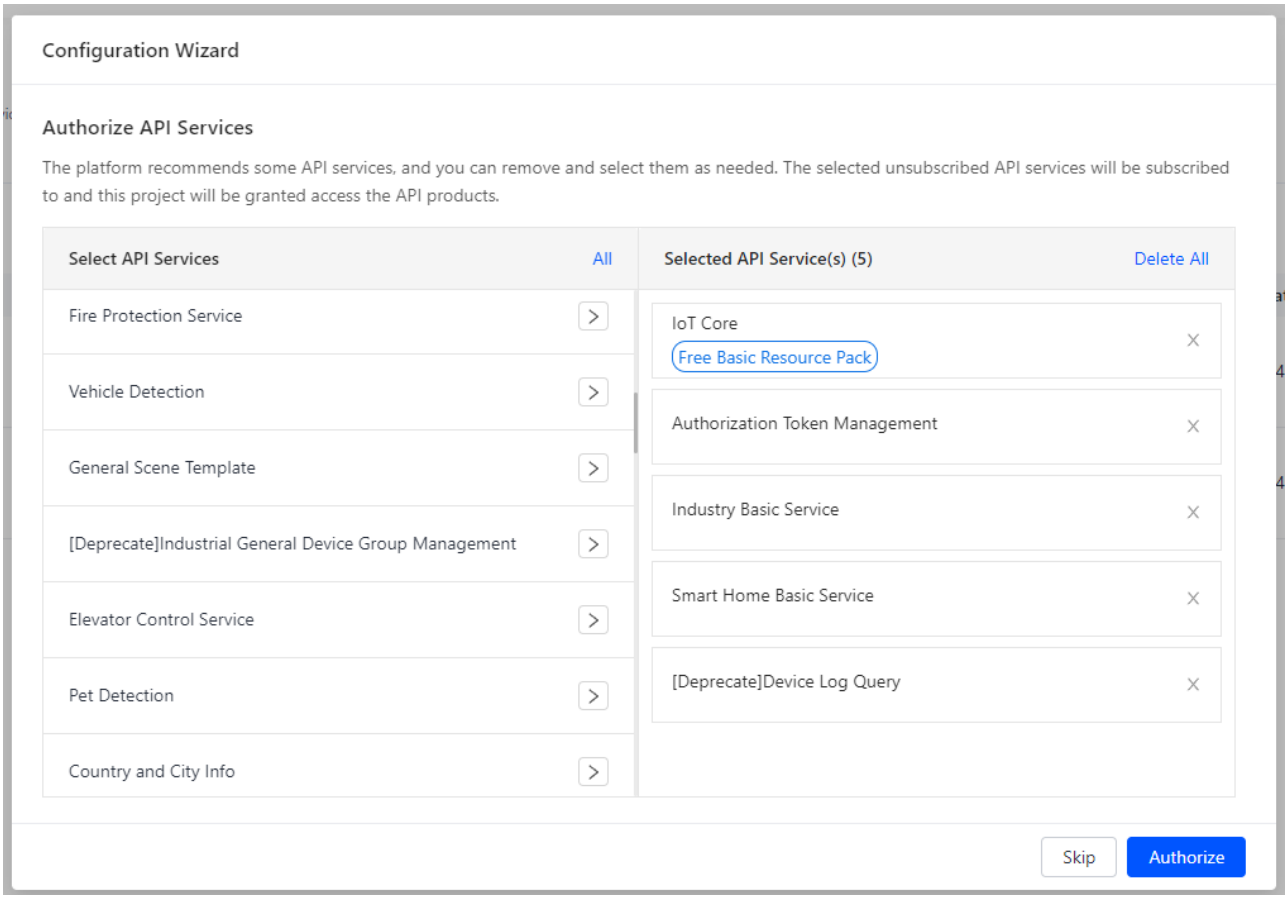

Fig. 29

9. Now that the project is created, let’s add the pet feeder to the project. Navigate to “Devices” → “Link Tuya App Account” → “Add App Account”.

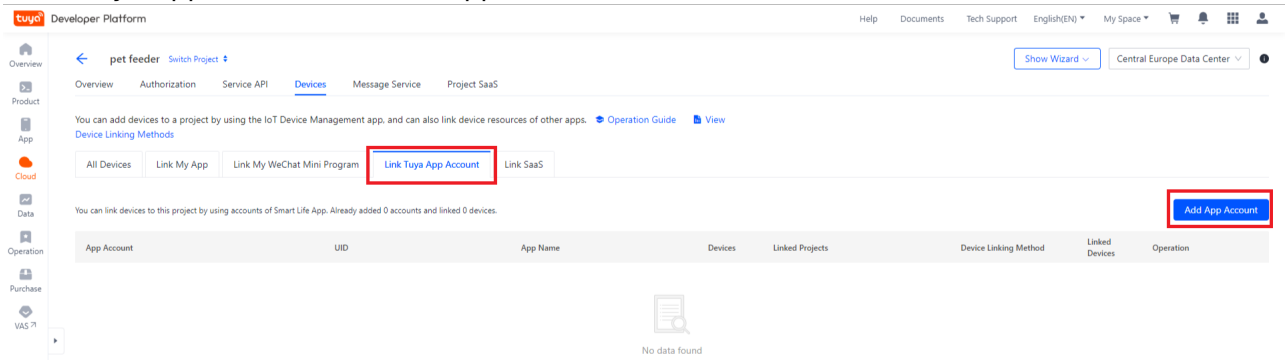

Fig. 30

10. A QR code will appear, open the “Tuya Smart” app on your phone, press on “Me” on the bottom right and then press on the scan icon on the top right, scan the QR code.
11. After the QR code is scanned, pick “Automatic Device Linking Method” and click on OK.

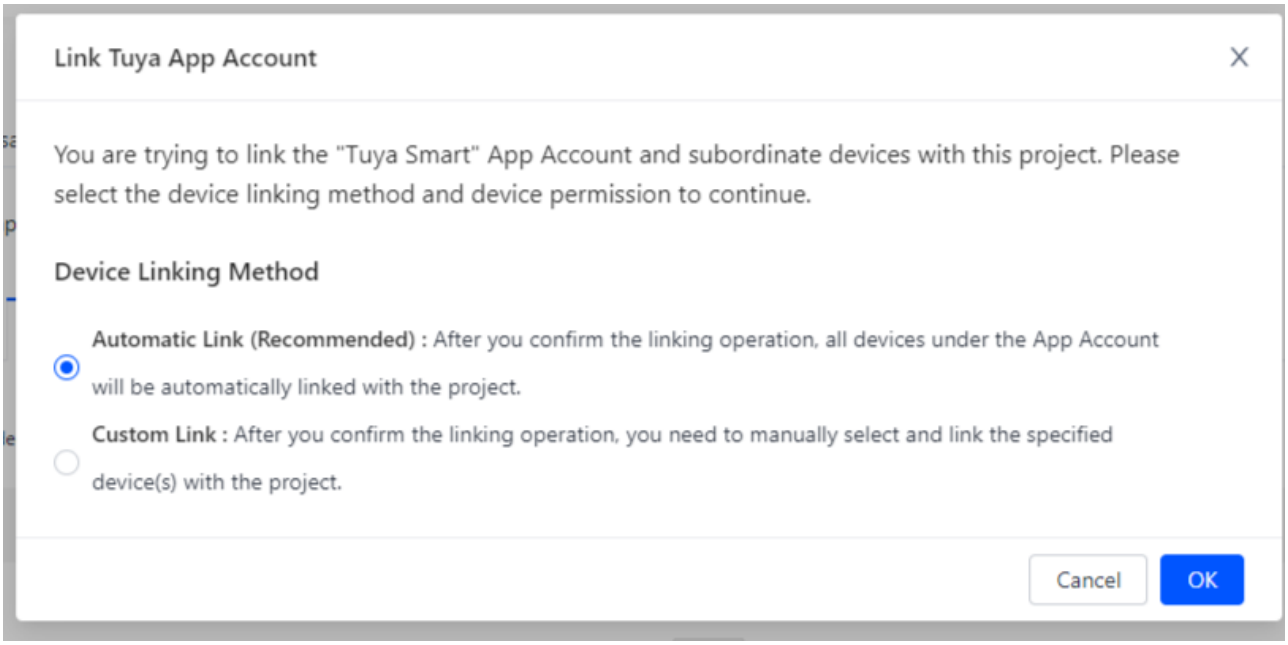

Fig. 31

12. Once the linking is complete, you should be able to see your account details on the “Link Tuya App Account” tab. **Make note of the “UID”**, we will use this later. Click on “Manage Devices”.

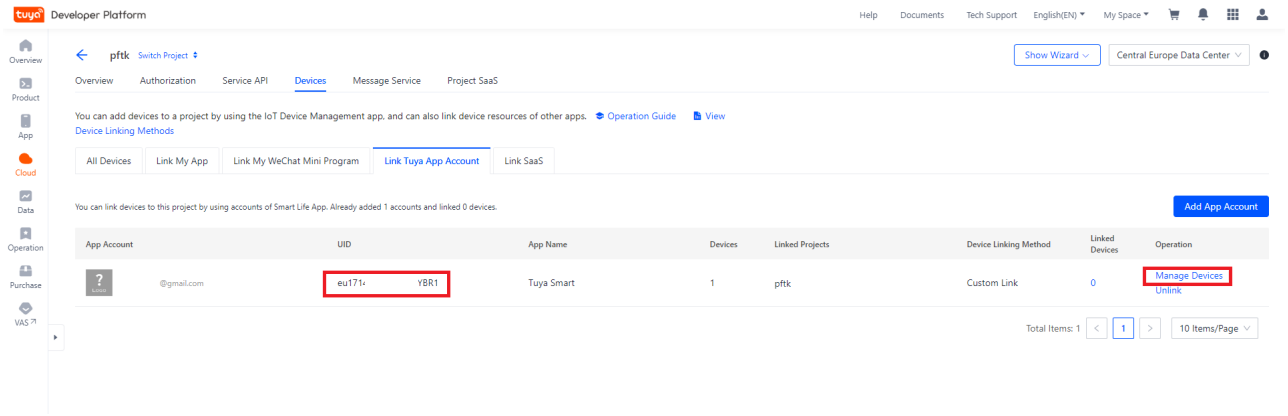

Fig. 32

13. If the Pet Feeder is shown as unlinked, check the checkbox next to the pet feeder and hit “Link Device”.

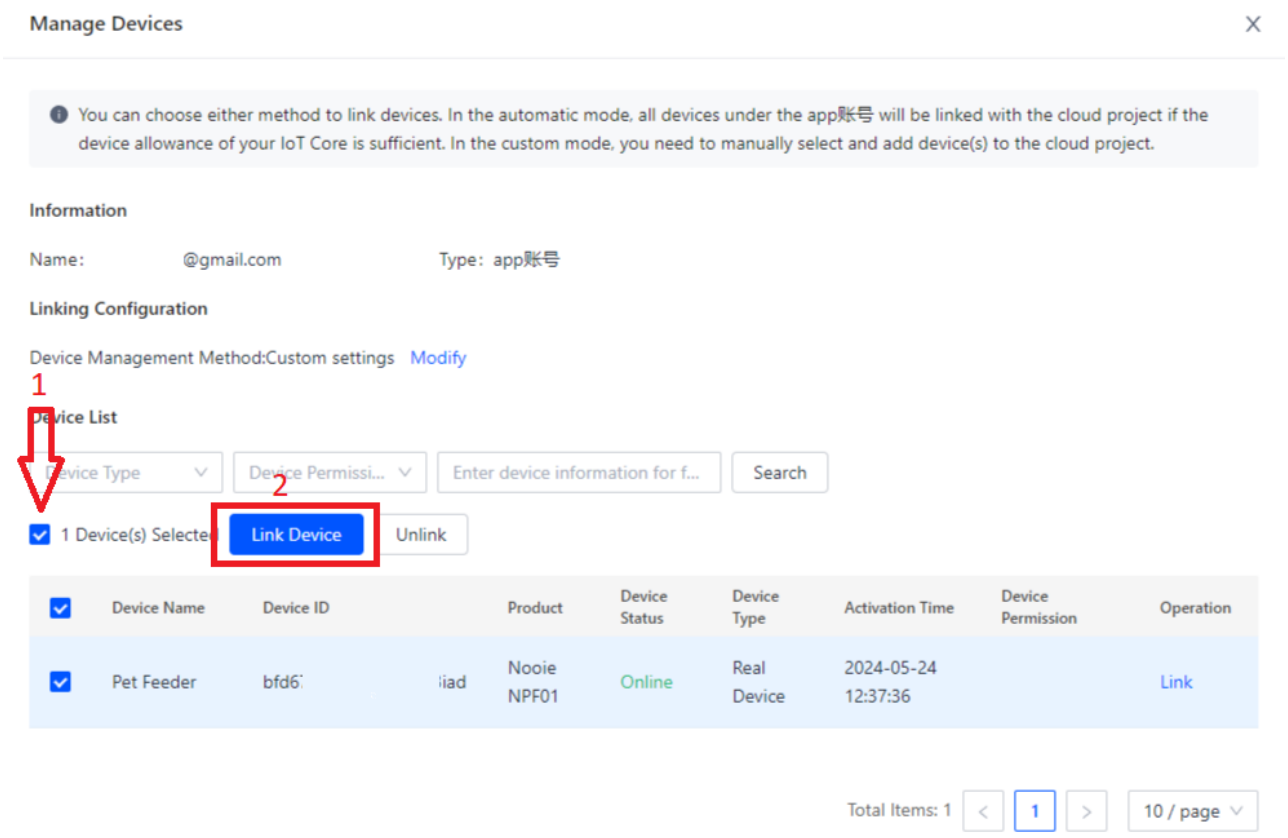

Fig. 33

14. Confirm the pet feeder exists on the “All Devices” tab, do not close this webpage.

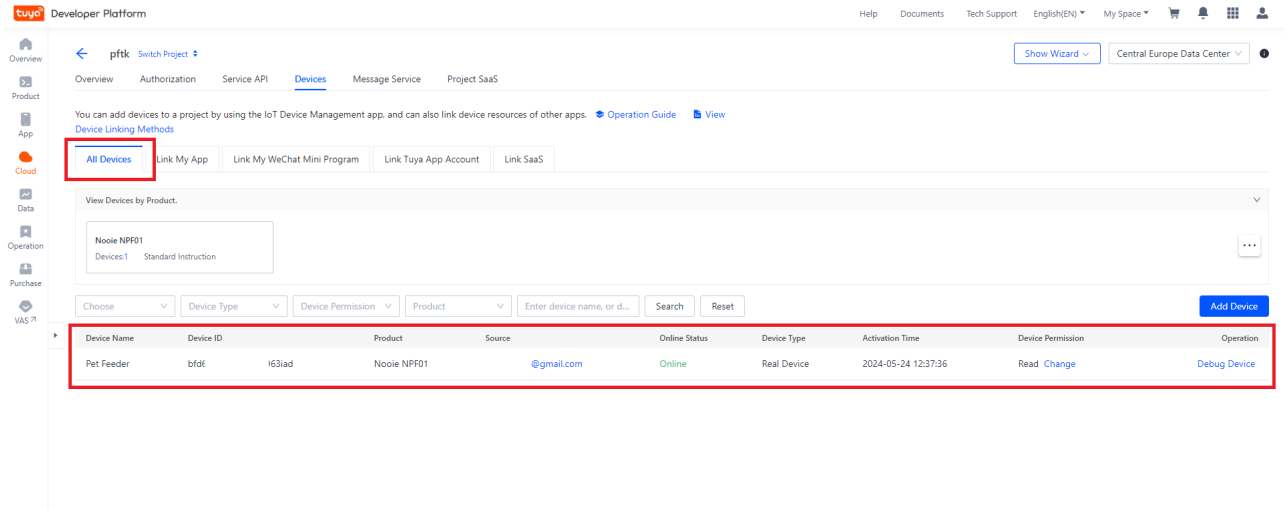

Fig. 34

15. Navigate back to the Home Assistant UI
16. Go to “Settings” → “Devices & Services” → “Add Integration” on the bottom right corner, Search for “LocalTuya”. Click on “LocalTuya Integration”.

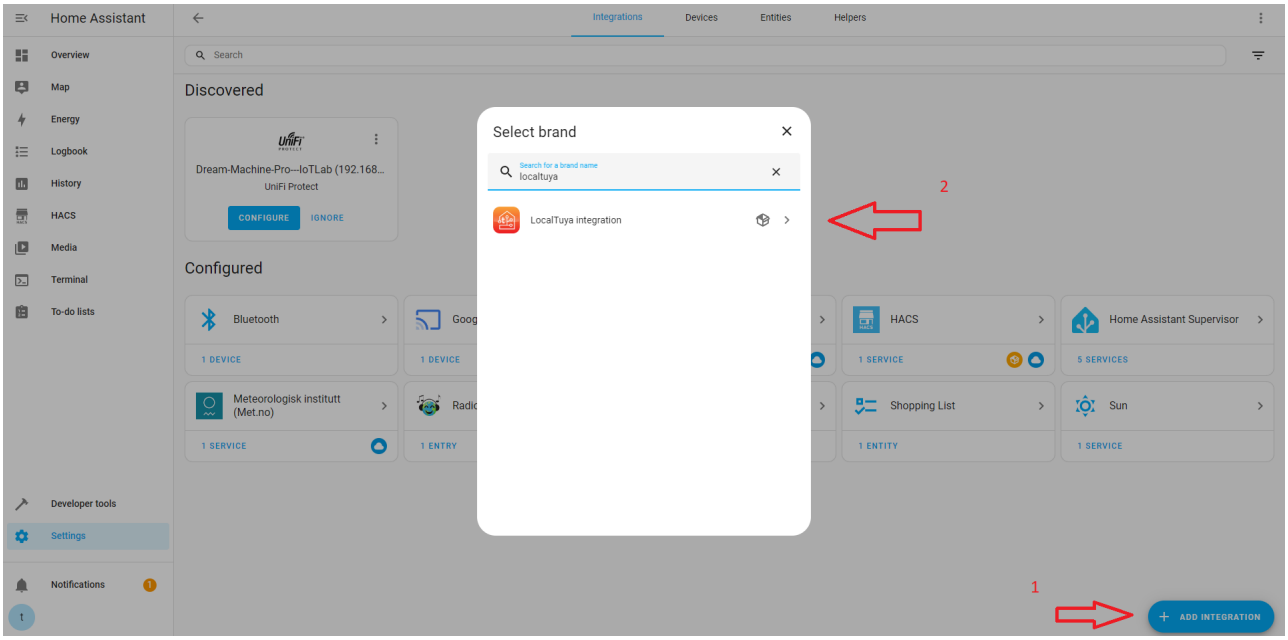

Fig. 35

17. Once LocalTuya is installed, Cloud API configuration pop up will appear. Make sure API server region is EU. For the rest of the credentials, we need to navigate back to Tuya developer platform.

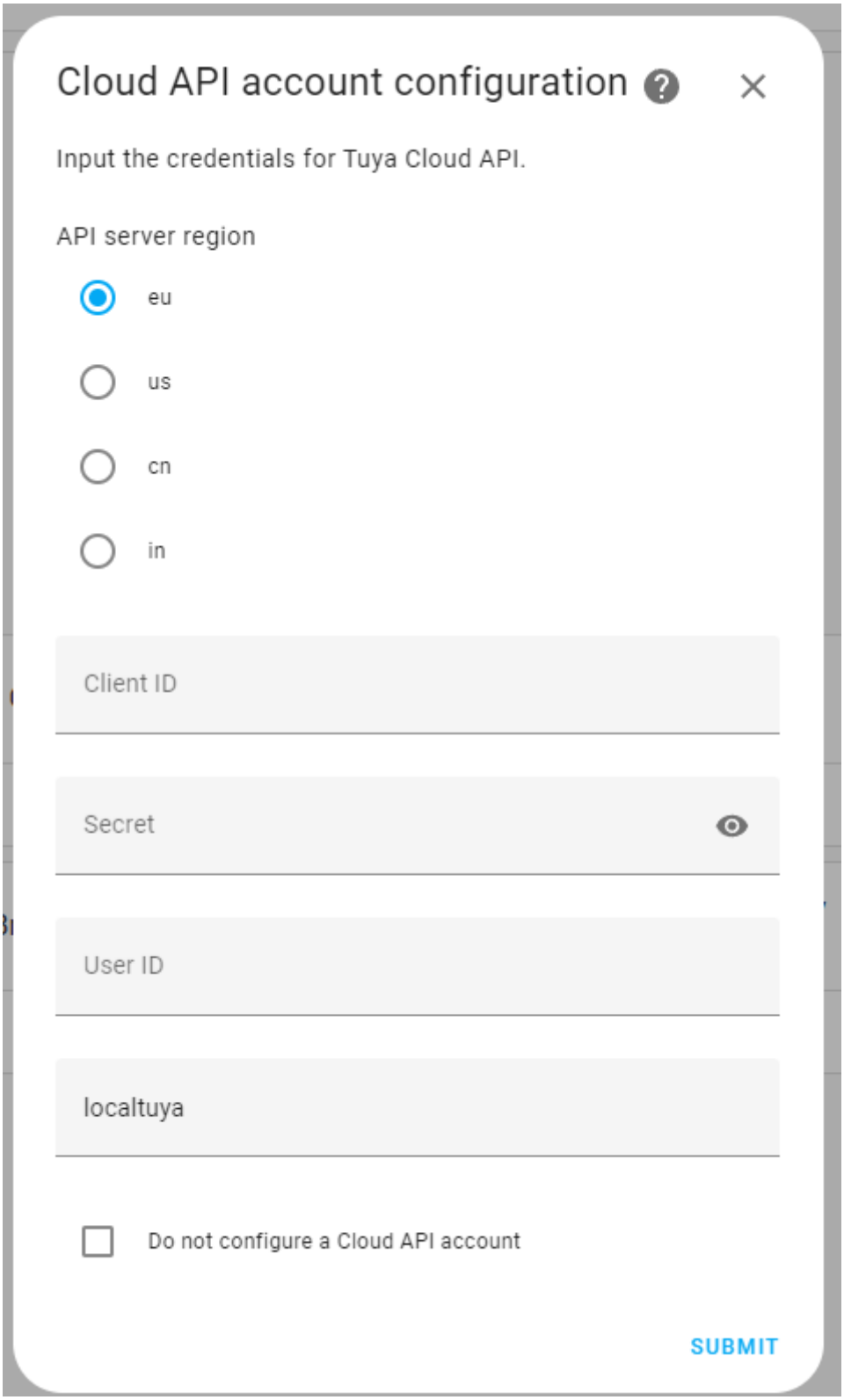

Fig. 36

18. On the left sidebar, hover on “Cloud” → “Development” → “Open Project”. The “Client ID” and the “Secret” can be found under the “Overview” tab of the cloud project. (not the Overview on the sidebar, the one under our project name).

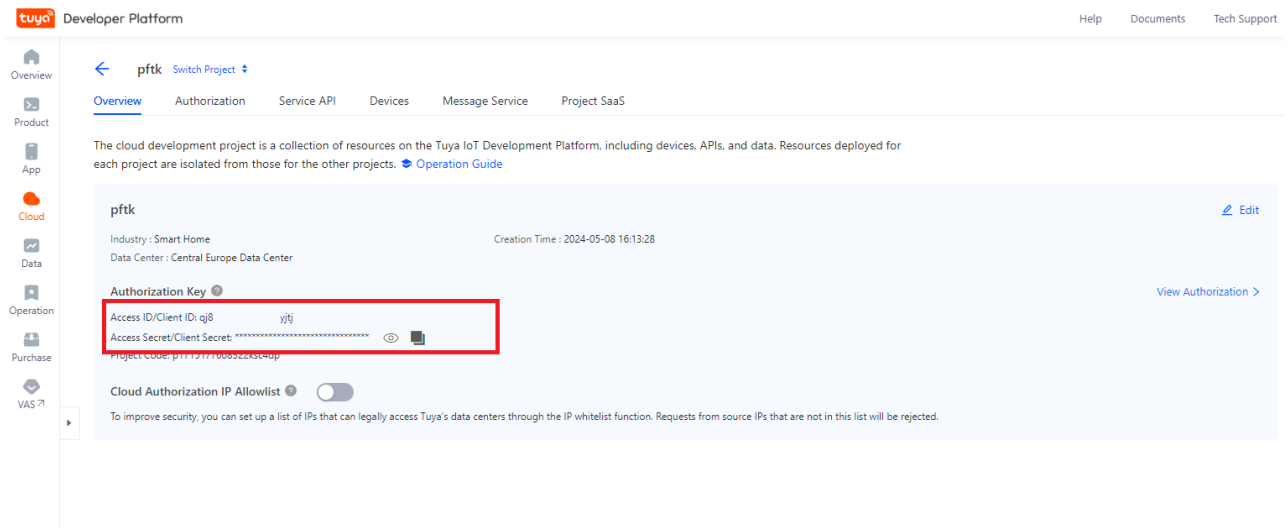

Fig. 37

19. Insert the credentials back to Home Assistant. Use the previously mentioned “UID” from “Link Tuya App Account” tab and click on “Submit”.
20. LocalTuya Integration is now linked with your Tuya Development Account. We still need to configure the pet feeder for local use.
21. You will be redirected to LocalTuya integration page, if not, Navigate to “Settings” → “Devices & Integrations” → “LocalTuya”.
22. Click on “Configure” on the same page → “Add a New Device” → “Pet Feeder” (which should be recognized automatically) and submit.

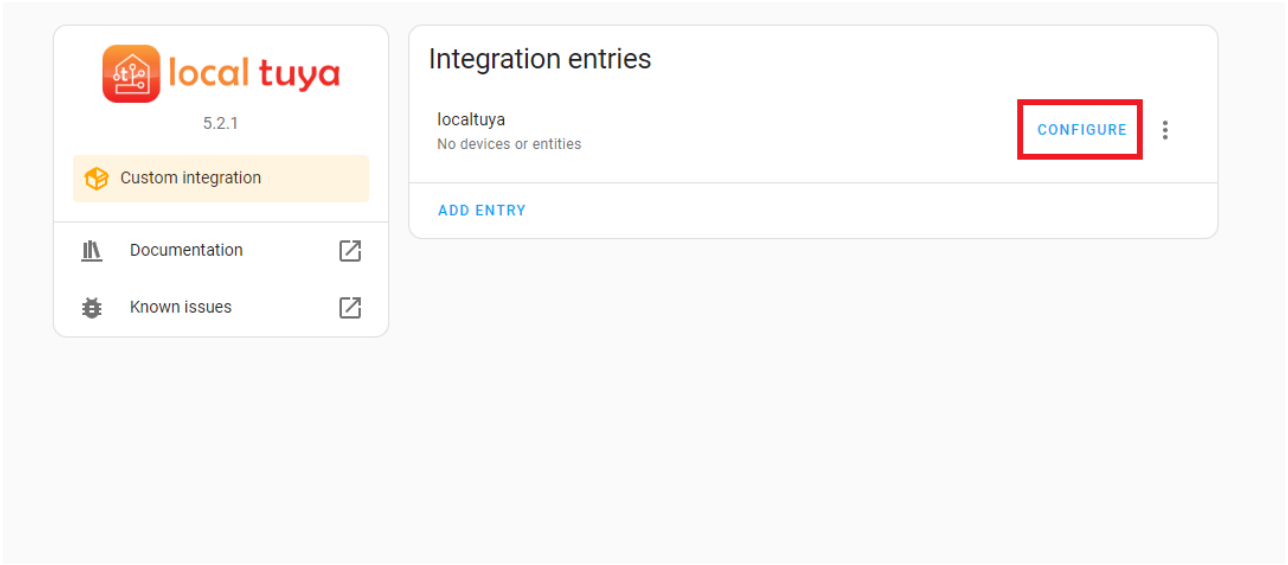

Fig. 38

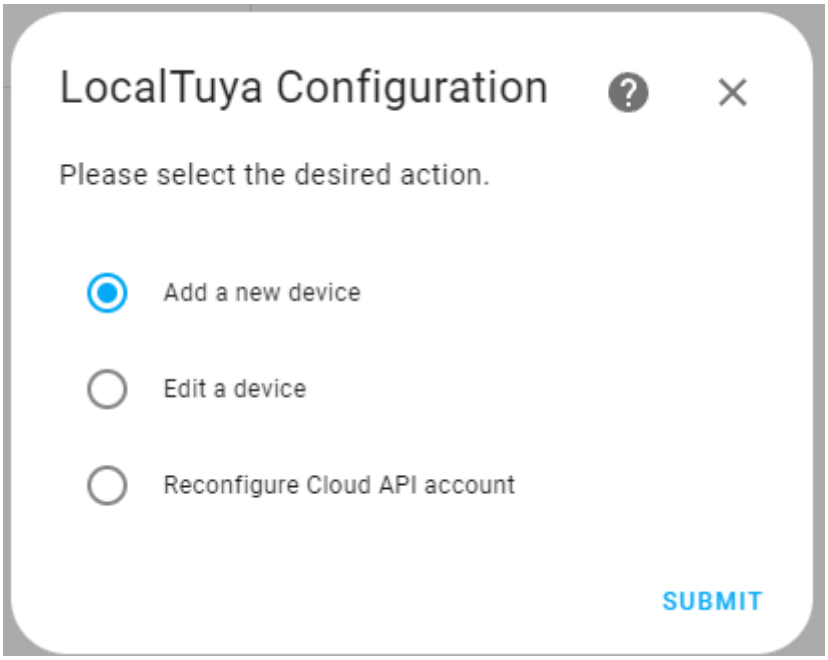
LocalTuya Configuration
Please select the desired action.
Add a new device
Edit a device
Reconfigure Cloud API account
SUBMIT

Fig. 39


Add a new device
Pick one of the automatically discovered devices or ... to manually to add a device.
Discovered Devices
Pet Feeder (192.168.20)
...
SUBMIT

Fig. 40

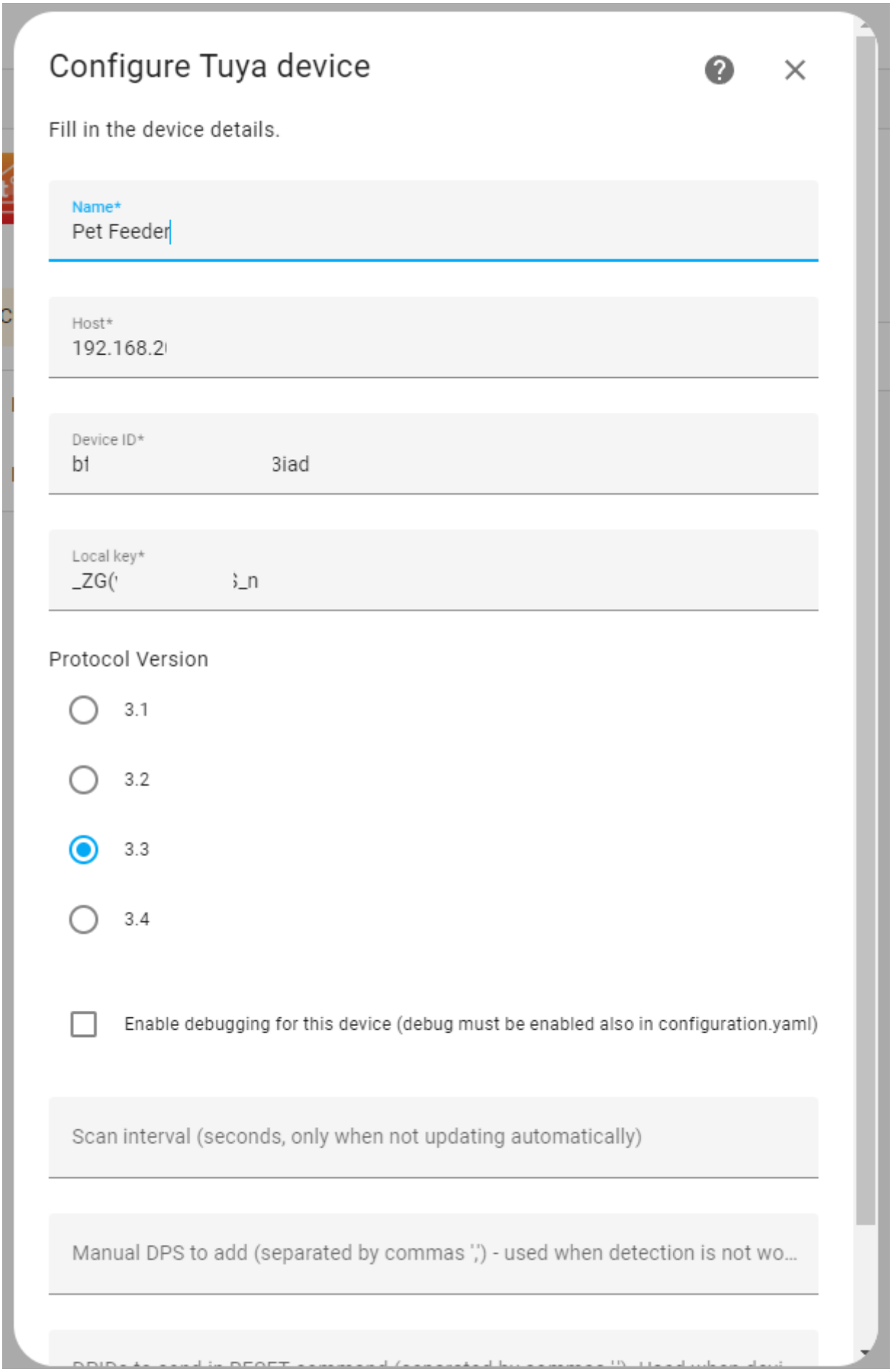

Fig. 41

1. Skip to step 23 if the credentials of the Pet Feeder are configured automatically. If the Pet Feeder's configuration is **not set automatically**, navigate back to the Tuya Developer

Platform, hover on the “Cloud” icon on the sidebar → “API Explorer”.

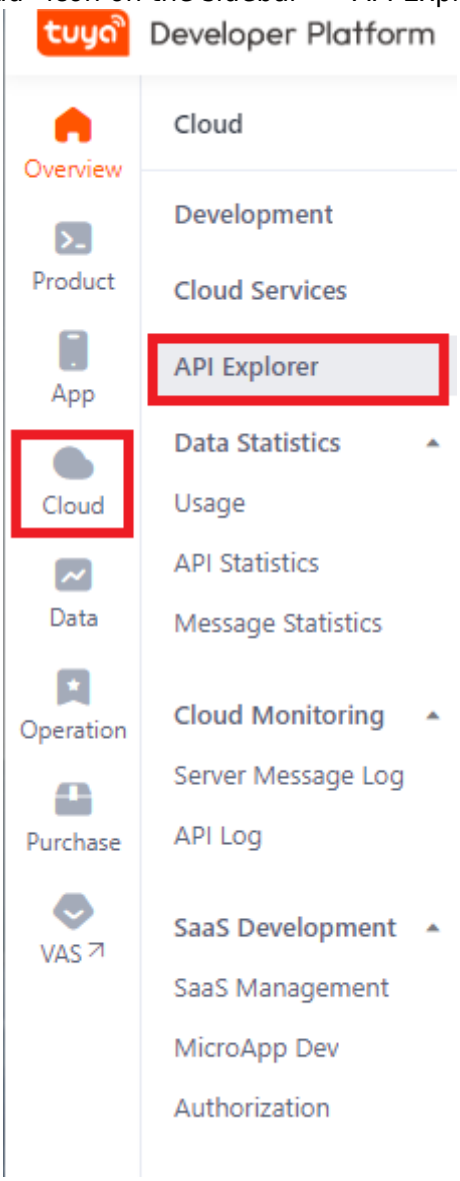

Fig. 42: Optional step, refer to step 22.a.

2. On the API explorer, Navigate to “Query Devices in Project” on “Device Management”.
3. Type “1” to “page_size” and click on “Submit Request”.
4. The response query gives us credentials of the device. We can find the local key and device id (called “id”).

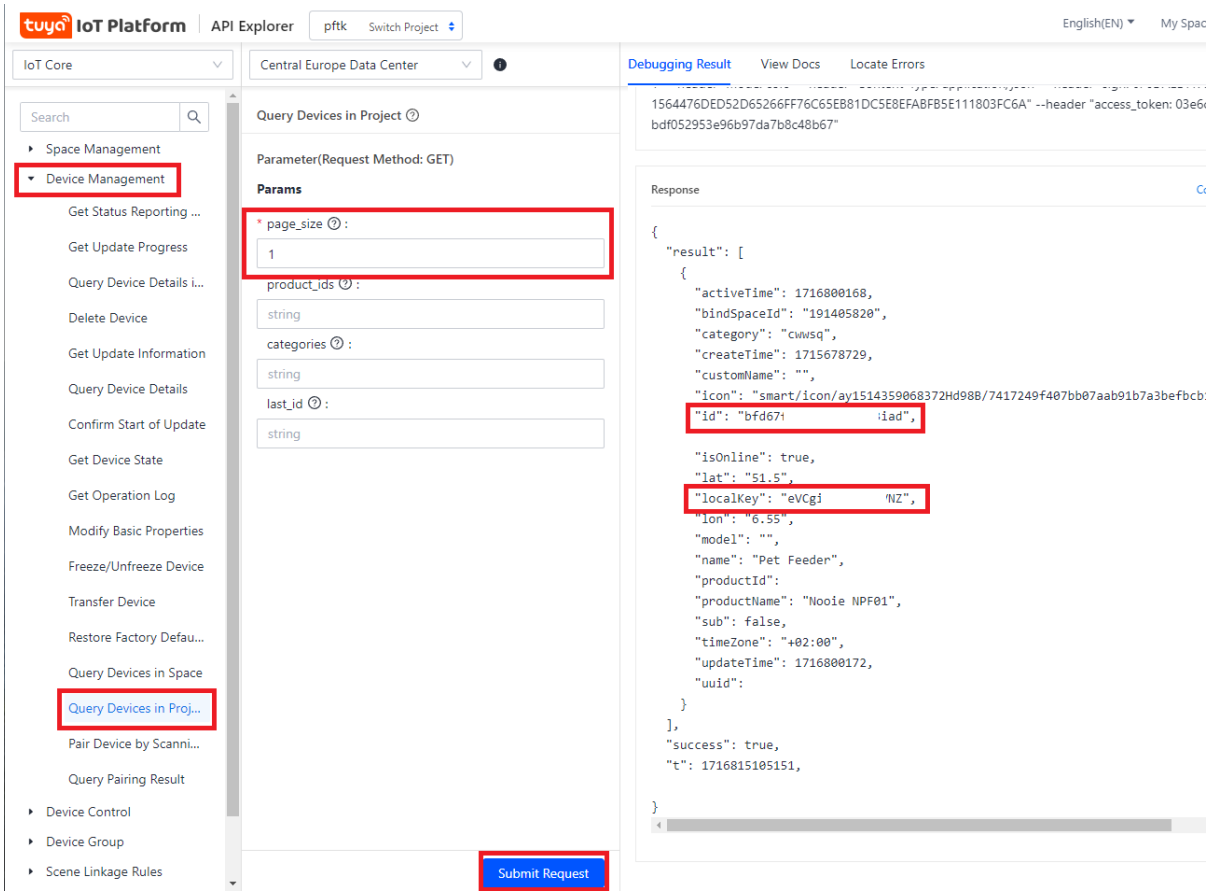

Fig. 43: Optional step, refer to step 22.a.

5. The Host address can be obtained via your router's web interface.

23. After you submit, you will be met with the "Entity Type Selection" pop-up. The pet feeder has different functions such as dispensing food or turning the LED on the pet feeder on and off. Each of these functions has their own "Data Point ID" to communicate with the cloud. We need to intercept these Data Points and create separate entities to control the device via the Home Assistant.

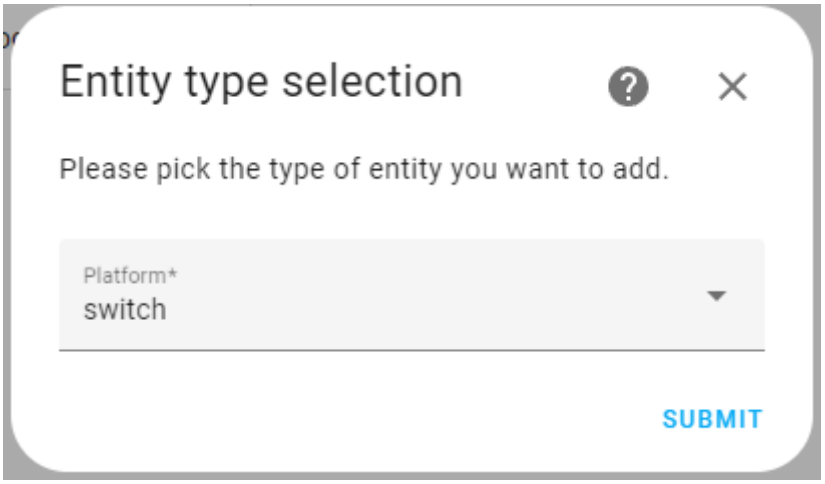

Fig. 44

24. Do not exit the Home Assistant instance and navigate back to Tuya Developer Platform. Open your project and navigate to "All Devices" tab under "Devices" and click on "Debug Device".

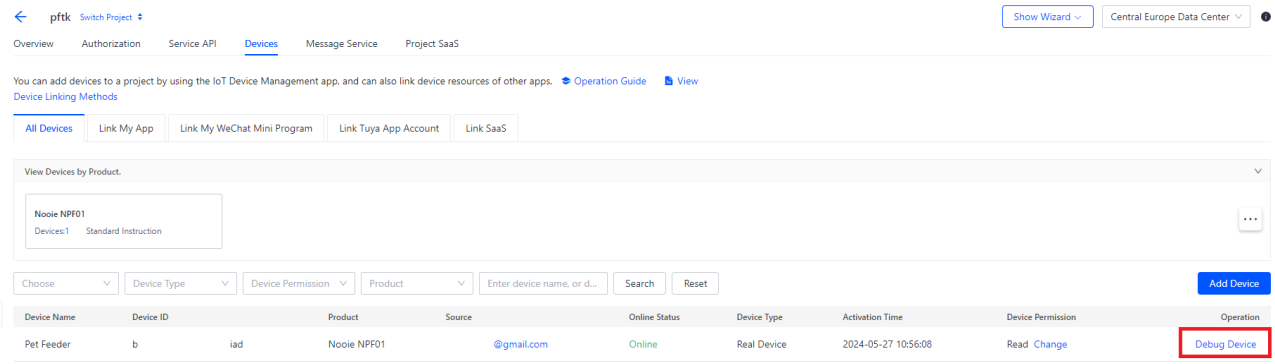

Fig. 45

25. On the device debugging page, under “Standard Instructions Set” are the functions of the device we can set new values for. Under “Standard Status Set” are the previously mentioned functions and the available sensors of the device.
26. To find out which Data Point is associated with which function of the device, navigate to “Device Logs” tab.

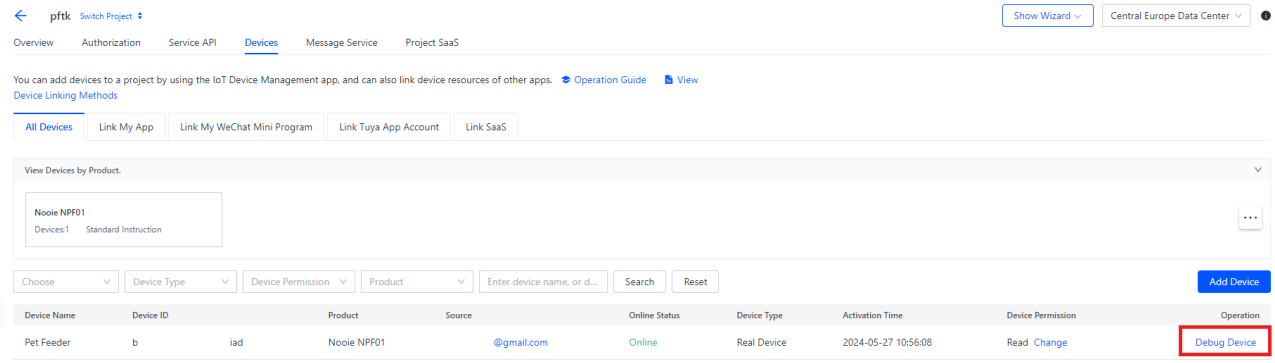

Fig. 46

27. Turn the developer console on of your browser (CTRL+SHIFT+I or F12 for Google Chrome). Navigate to “Network” tab on the developer console.
28. Notice the “Select DP ID” dropdown menu on the top left side of the page.
29. While the Developer console is on, click on the DP ID dropdown menu and select “Manual Feed”, click on “Search”.
30. After you click on “Search”, you should see the network tab of your developer console update with several different logs. “list” is the one we are looking for.

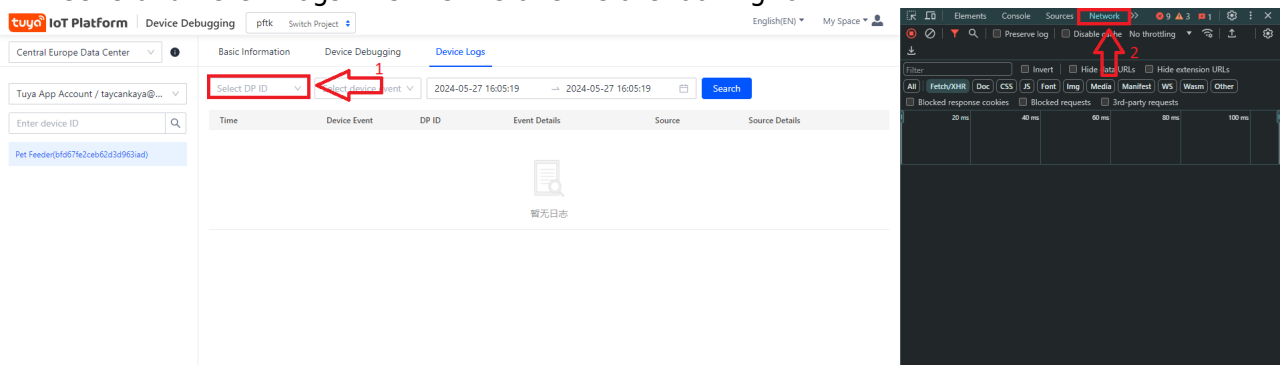

Fig. 47

31. Click on “list”, and navigate to the “Payload” tab, “Code” is the Data Point ID we are looking for. In this case it’s 3.

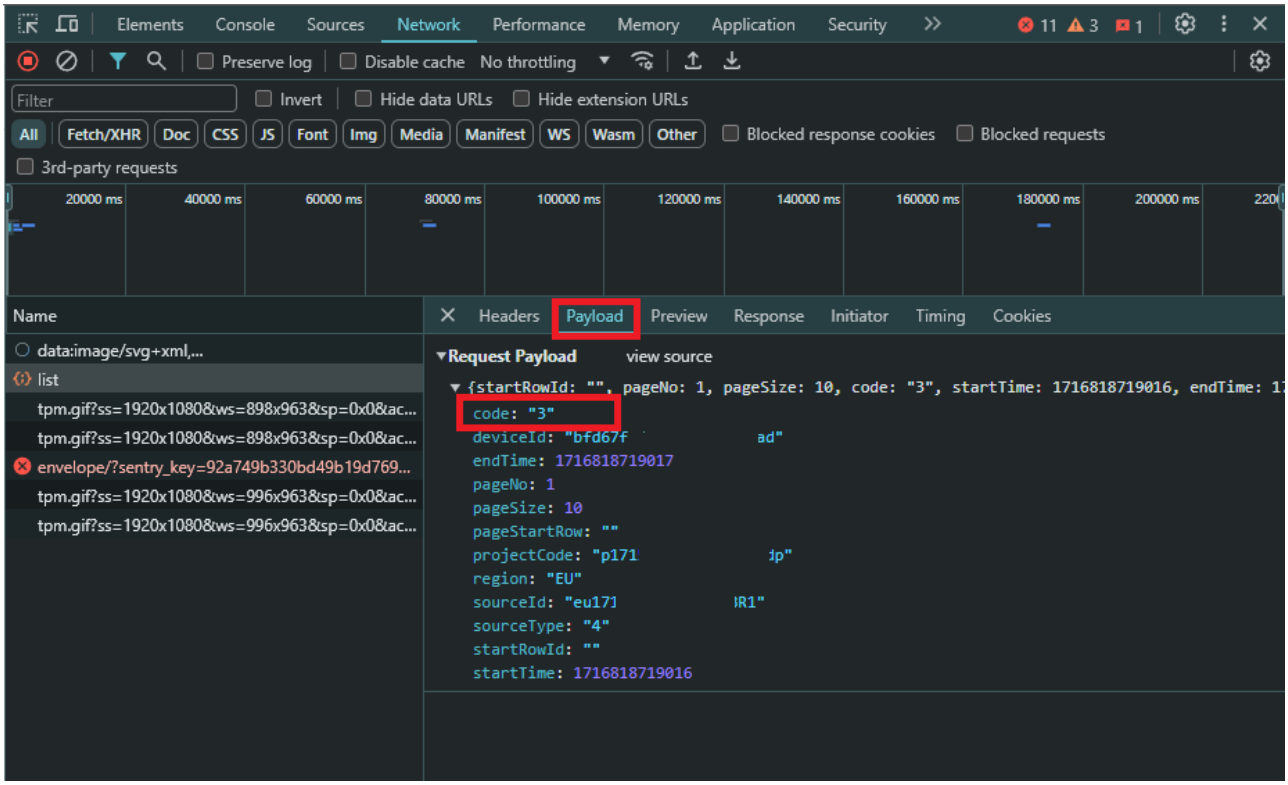

Fig. 48

32. The ID is our Data Point, which is 3.
33. Navigate back to “Device Debugging” tab, make note of the manual_feed’s type and values. It’s an integer with a minimum value of 1, maximum value of 20 and with a step of 1.

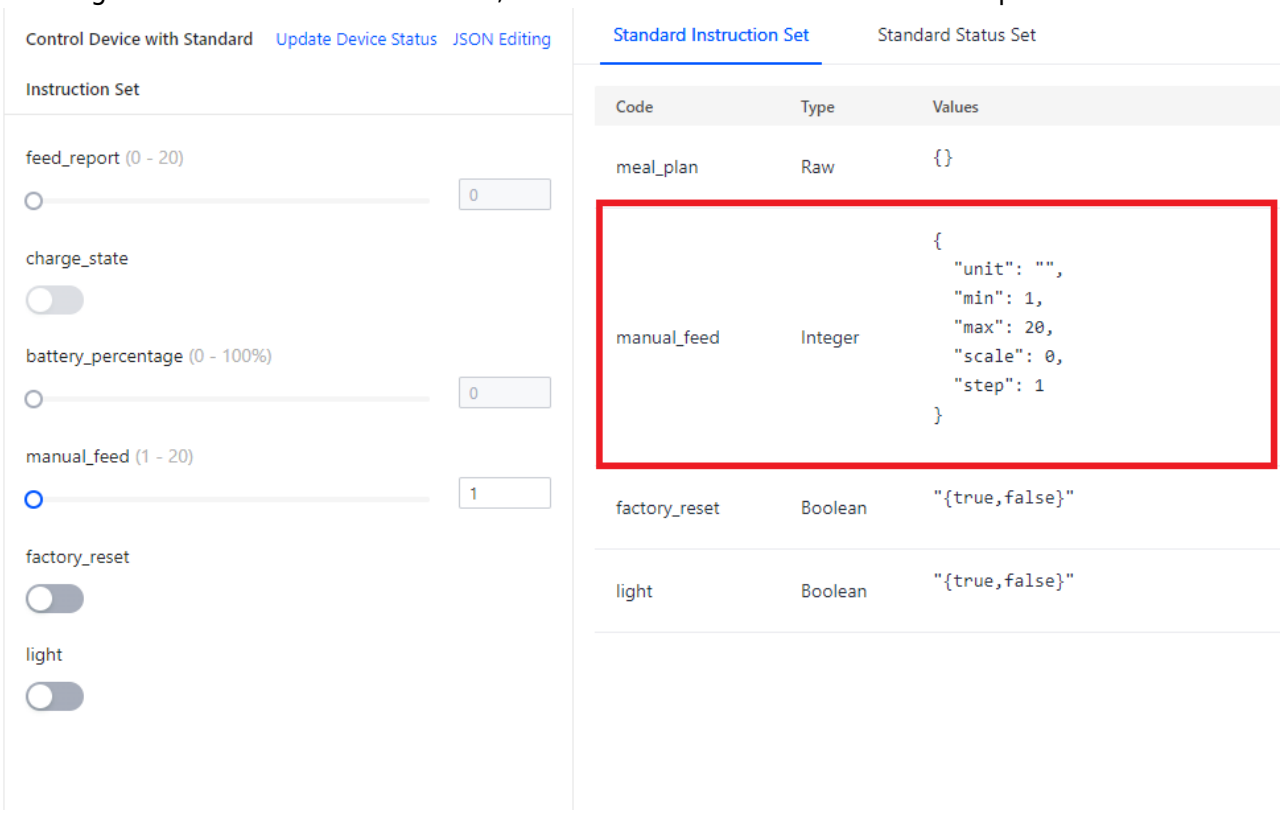

Fig. 49

34. Navigate back to Home Assistant, last time, we were at the Entity Type Selection pop-up. “Settings” → “Devices & Services” → “LocalTuya” → “Configure” → “Add a new Device” → ”Submit” → “Submit”.
35. Let’s add the manual feed function to LocalTuya. LocalTuya includes integers in “number”, pick it from the dropdown menu and submit.

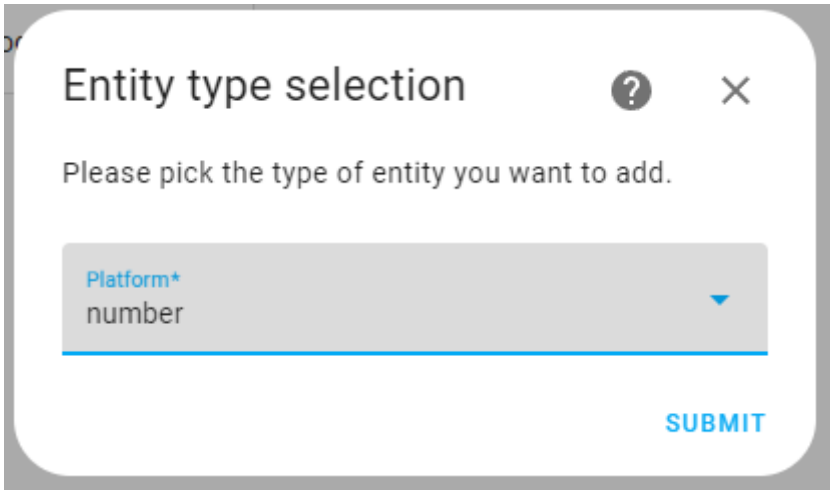

Fig. 50

36. The “ID” dropdown menu includes DP IDs of the pet feeder, we know 3 is manual feed. From the dropdown menu, pick 3. Call it Manual Feed. The minimum value is 1, maximum value is 20 and the increment between values is 1. Click “Submit’.

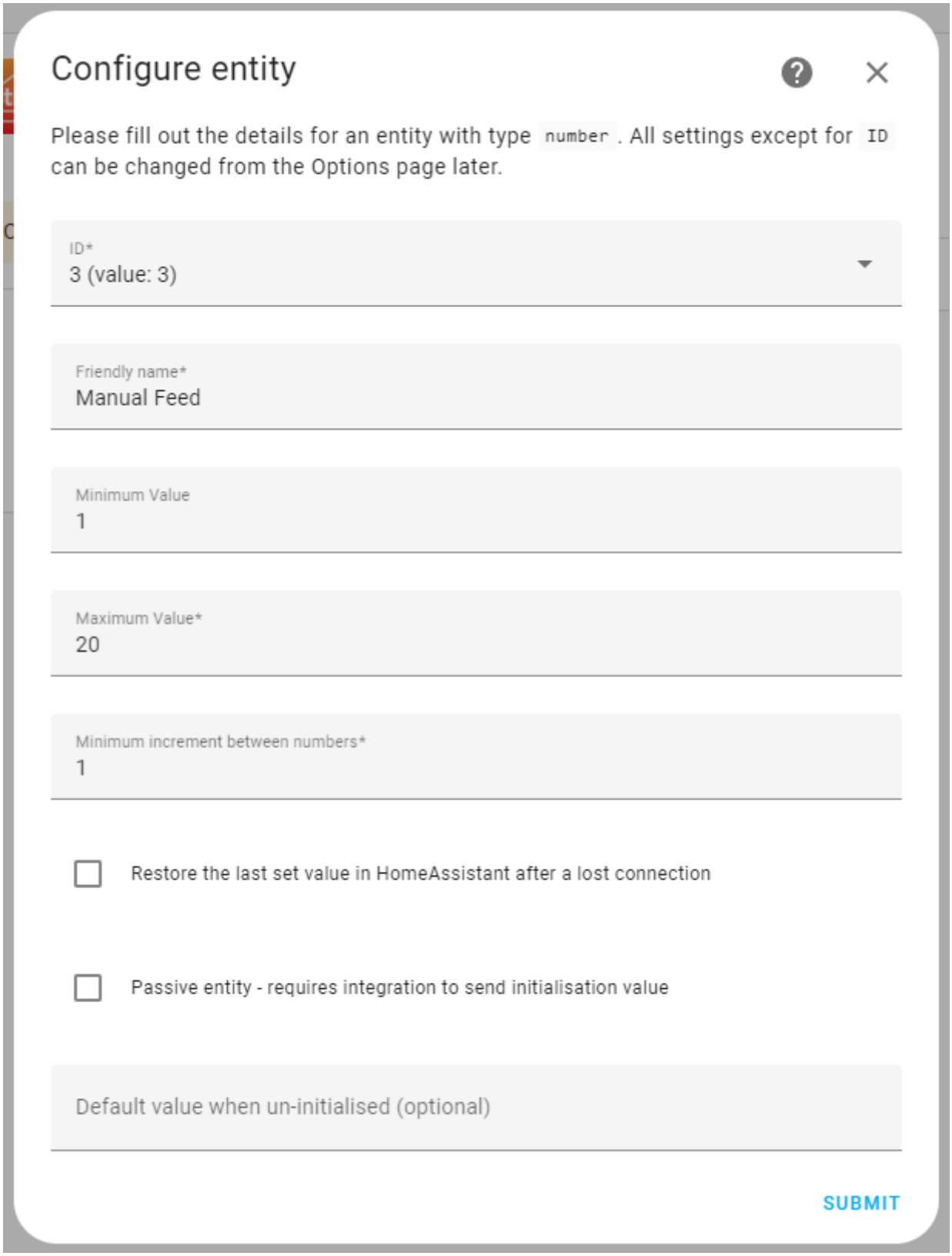

Fig. 51

37. On the next step, the entity type selection will appear again, for now, check the “Do not add any more entities” and click “Submit” (we can always add more later).

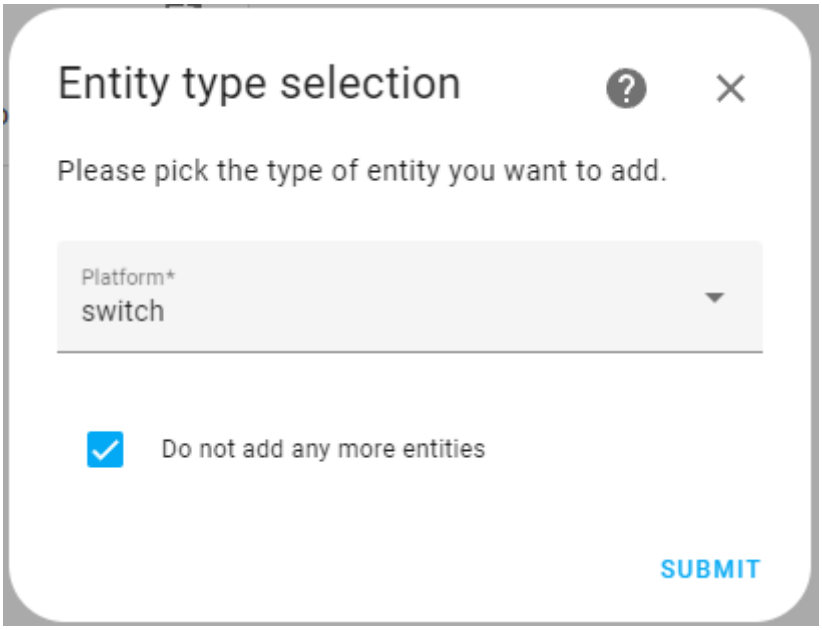

Fig. 52

38. Now let’s see if the manual feed entity works. Click on “1 device” below the logo of localtuya.

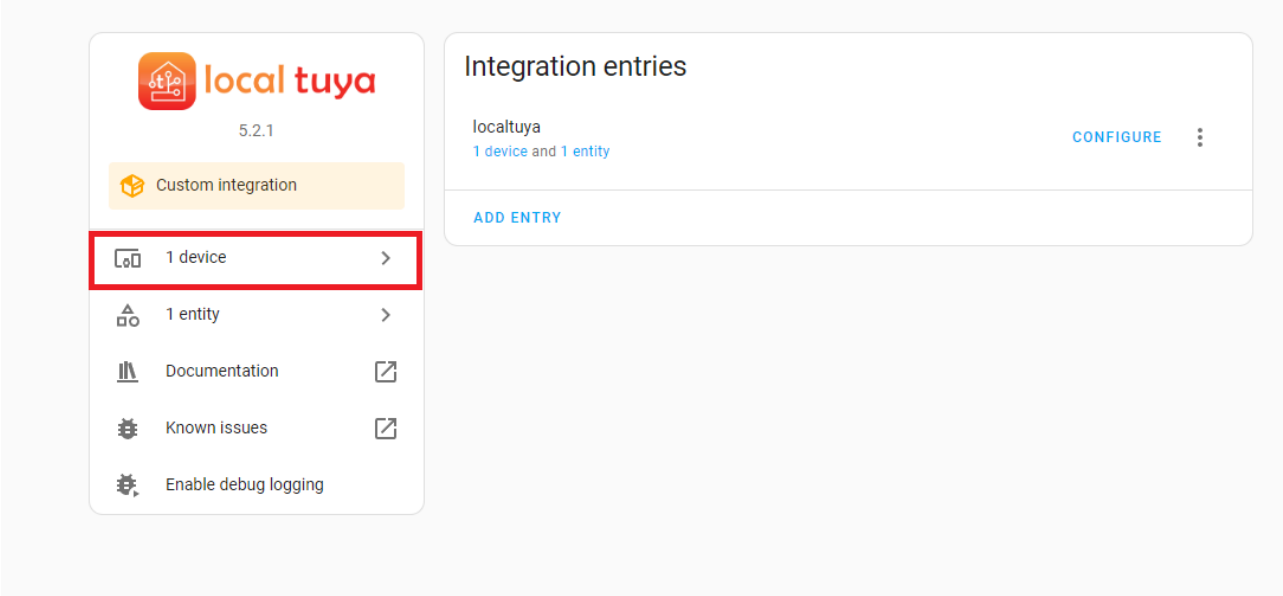

Fig. 53

39. Slide the Manual Feed to any value and check the Pet Feeder is dispensing food.

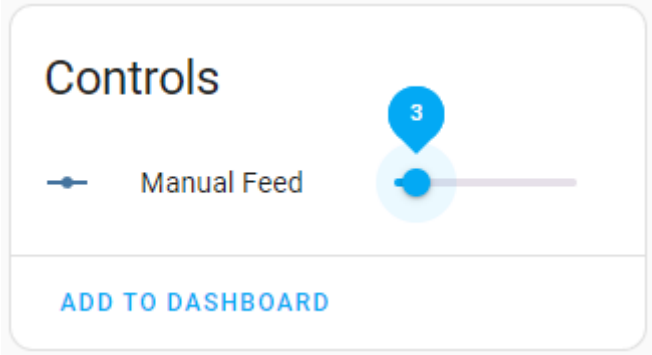

Fig. 54

40. Add more entities by repeating step 26 and beyond.

4.9 Creating Automation function for pet feeder to trigger on MQTT Message.

1. This step was by far the easiest in the whole process.

2. In the “When” section, choose MQTT and set the topic to the one we already established in the MQTT Broker.
3. In the “Then Do” section, choose the device, in our case “Pet Feeder”. Choose the action already established in the previous step “Manual Feed” then set the value to be between 1-10.

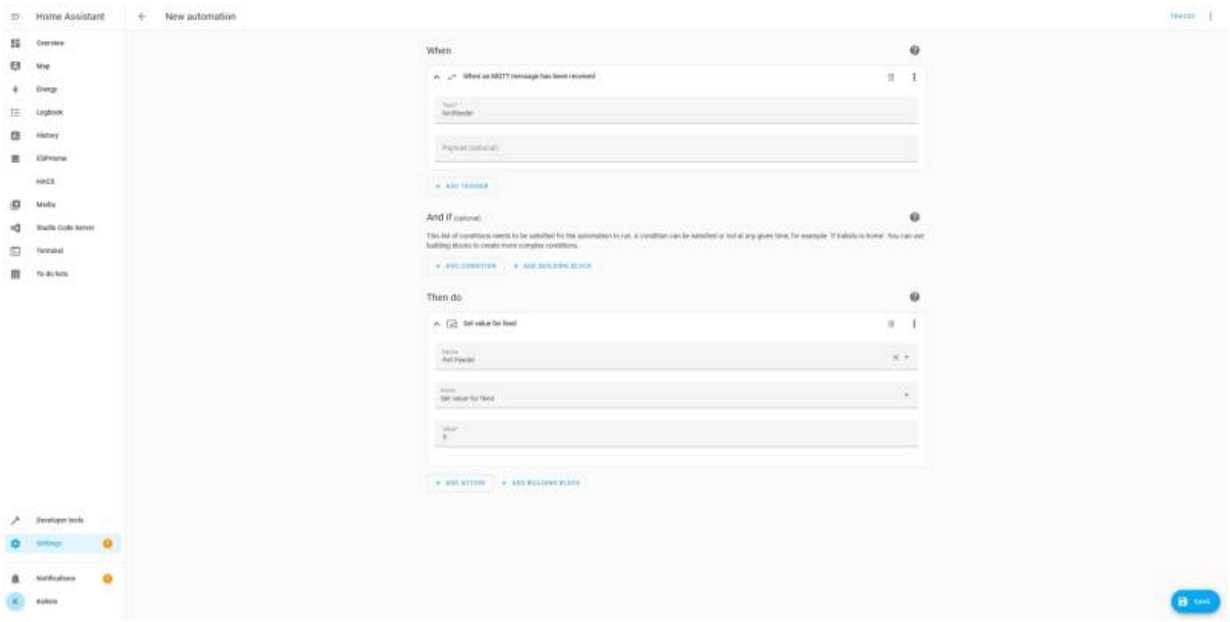

Fig. 55: Pet feeder automation configuration

5. Results and Discussion

- The concept has proven to be very much viable with all the requirements fulfilled in a prototype and ready to be moved to the next steps.
- Using MQTT for communication between the different devices has proven simple yet efficient.
- The e-ink display was a great choice for this project due to its high visibility in daylight and power efficiency.
- To display payer information, the payload of an MQTT message can be used.

6. Future Steps

- Build a casing for the display component to keep it protected and portable without any fear of damaging connectors.
- Further develop the design of the text/images shown on the e-ink display.
- Figure out whether the Kalisto main website can be used to integrate this whole system or not.
    - If not, further development of the web-application must proceed.
- Payment can be handled using Stripe, a comprehensive suite of tools for handling online financial transactions.
- Payer information should be stored in a queue like structure to give a chance for the display to show all donations.

7. Obstacles

- It was also a difficult and challenging accomplishment of the particular system architecture to ensure high level expertise with precision.
    - This can be shown from the different system architecture diagrams here.
- Finds the correct pin configuration for between the XIAO ESP-32 and the e-ink display.
- The e-ink display cable broke at some point during the implementation and testing and another display was ordered to continue with the project.
- Just before the presentation, as there was another team using a RaPi and the same Nooie Pet Feeder connected on the same network, there was a conflict between both the 2 RaPis. This was solved by changing both the hostname and the port for one of them.

8. Key Learnings

- The different communication protocols such as SPI and I2C.
- E-ink displays and how they function.
- Home Assistant and most of its features.
- Automation and triggers
- ESPHome and how its yaml syntax works.
- MQTT, Brokers, Topics, QoS, etc.
- Networking Basics.

9. References

- https://wiki.seeedstudio.com/xiao_esp32s3_getting_started/
- https://wiki.seeedstudio.com/XIAO-eInk-Expansion-Board/
- https://www.home-assistant.io/docs/
- https://esphome.io/
- https://www.npmjs.com/package/precompiled-mqtt