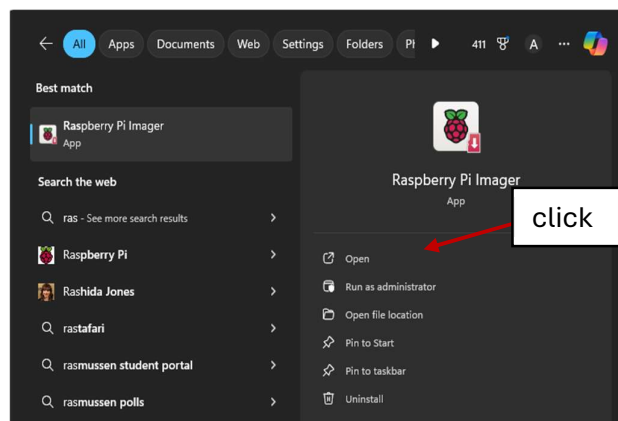


ขั้นตอนการติดตั้ง

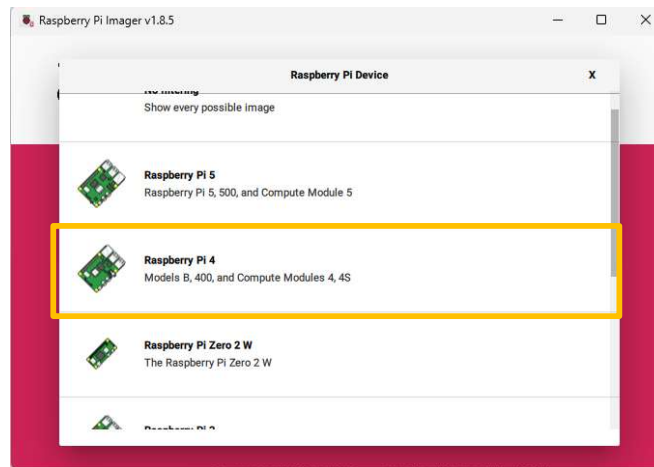
1. ขั้นตอนการติดตั้งเพื่อใช้งานระบบทั้งหมด

a) ติดตั้ง Raspberry Pi OS

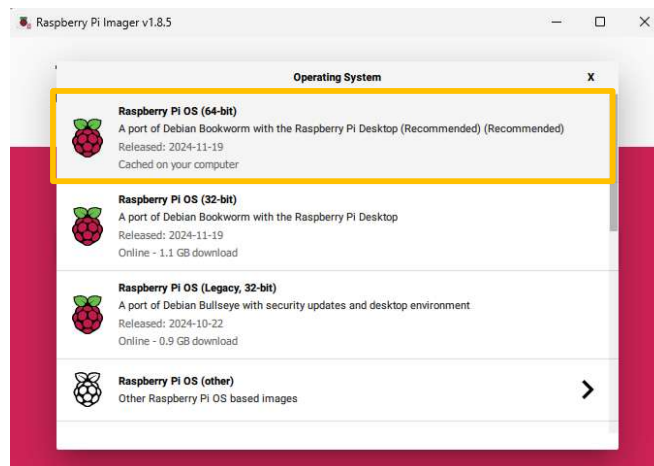
- i) อุปกรณ์ที่ต้องใช้
 - 1) Raspberry Pi 4 Model B
 - 2) MicroSD Card (แนะนำขนาด 16GB หรือ 32GB, Class 10 ขึ้นไป)
 - 3) เครื่องคอมพิวเตอร์ (Windows, macOS หรือ Linux)
 - 4) อะแดปเตอร์จ่ายไฟ (USB-C 5V/3A สำหรับ Raspberry Pi 4)
 - 5) สาย HDMI และจอภาพ (ถ้าต้องการต่อจอแสดงผล)
 - 6) คีย์บอร์ดและเมาส์ (ถ้าต้องการตั้งค่าผ่าน GUI)
- ii) ขั้นตอนที่ 1: ดาวน์โหลดและติดตั้ง Raspberry Pi OS ลงบน MicroSD Card
 - 1) ดาวน์โหลด Raspberry Pi Imager
 - ไปที่เว็บไซต์ <https://www.raspberrypi.com/software/>
 - ดาวน์โหลดและติดตั้ง Raspberry Pi Imager สำหรับระบบปฏิบัติการของคุณ (Windows/macOS/Linux)
 - 2) ใส่ MicroSD Card เข้าไปในเครื่องคอมพิวเตอร์
 - ใช้ Card Reader หากคอมพิวเตอร์ไม่มีช่องใส่ MicroSD
 - 3) เลือก OS และติดตั้งลง MicroSD Card
 - เปิดโปรแกรม Raspberry Pi Imager ที่ช่องค้นหา(Search)



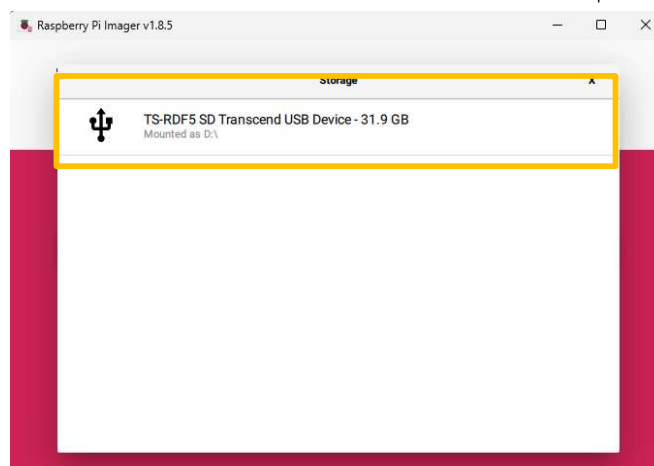
- กด "Choose Device" เลือก Raspberry Pi Module ที่ใช้



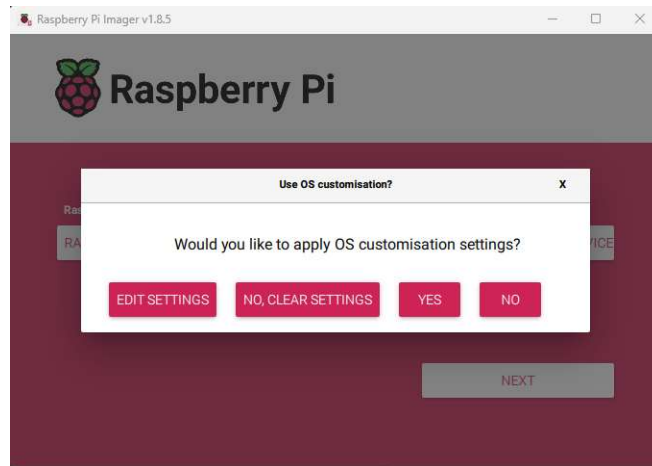
- กด "Choose OS" เลือก "Raspberry Pi OS (64-bit)" หรือ "Raspberry Pi OS (32-bit)" ตามที่ต้องการ



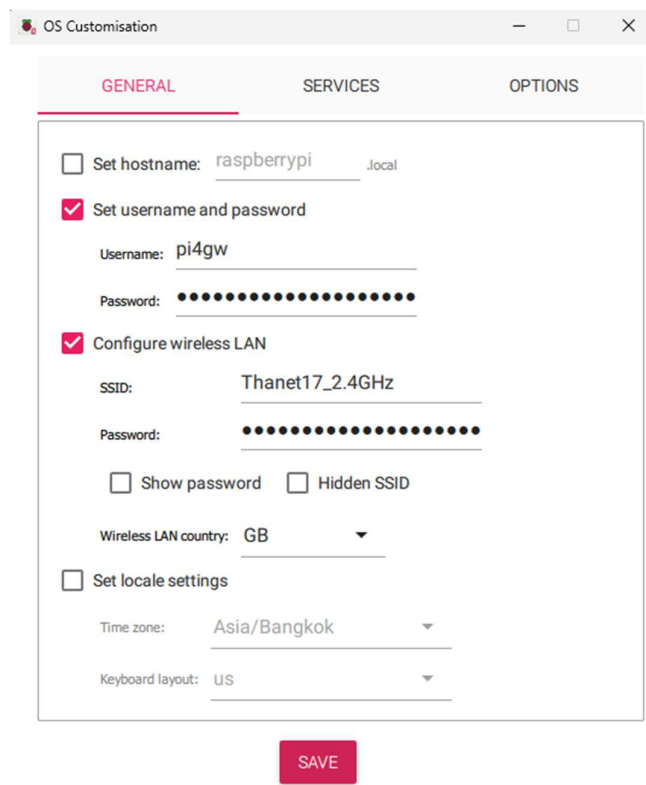
- กด "Choose Storage" → เลือก MicroSD Card ของคุณ แล้วกด NEXT



- กด EDIT SETTINGS เพื่อตั้งค่า User และ WiFi เพื่อ login เข้าหน้าDesktop ของ PI



- จากนั้นทำการตั้ง ชื่อ USER , รหัสผ่าน, ชื่อ WiFi และรหัสผ่าน ให้เรียบร้อย
เมื่อตั้งเสร็จกด SAVE
(ห้ามลืมชื่อ กับ รหัสผ่านเพราะจะใช้ login เข้าหน้าDesktop ของ PI)



- กด "Write" → โปรแกรมจะเขียน OS ลงการ์ดและฟอร์แมตข้อมูลเก่าเป็นอันเสร็จสิ้น



- ทำการลง OS เสร็จเรียบร้อยแล้ว พร้อมนำไปติดตั้งที่ตัวบอร์ด Raspberry pi

iii) ขั้นตอนที่ 2: ตั้งค่า Raspberry Pi ก่อนเปิดเครื่อง

1) เปิดใช้งาน SSH (ถ้าใช้แบบ Headless – ไม่มีจอ)

- หลังจากเขียน OS ลง MicroSD Card แล้ว ให้นำการ์ดออกและใส่กลับเข้าไปที่คอมพิวเตอร์
- ไปที่ไดร์ฟของ MicroSD (ชื่อ boot)
- สร้างไฟล์ชื่อ ssh (ห้ามมีนามสกุลไฟล์) เพื่อเปิดใช้งาน SSH

2) ตั้งค่า Wi-Fi (ถ้าใช้แบบ Headless)

- ในโฟลเดอร์ boot สร้างไฟล์ชื่อ wpa_supplicant.conf และใส่ข้อมูลดังนี้ (เปลี่ยน SSID และ PASSWORD ตามเครือข่ายของคุณ)

```
country=TH
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
    ssid="YOUR_WIFI_SSID"
    psk="YOUR_WIFI_PASSWORD"
    key_mgmt=WPA-PSK
}
```

- บันทึกและนำการ์ดไปใส่ Raspberry Pi

iv) ขั้นตอนที่ 3: เปิดเครื่องและตั้งค่าเริ่มต้น

1) ใส่ MicroSD Card ลงใน Raspberry Pi

2) เชื่อมต่ออะแดปเตอร์ไฟ USB-C เพื่อเปิดเครื่อง

3) ถ้าใช้จอแสดงผล

- Raspberry Pi จะบูตเข้า GUI (Desktop) และให้ตั้งค่าภาษา, โซนเวลา และเชื่อมต่อ Wi-Fi

4) ถ้าใช้แบบ Headless (ไม่มีจอ)

- ใช้ SSH เพื่อเชื่อมต่อจากคอมพิวเตอร์ (หาหมายเลข IP ของ Raspberry Pi ในเราเตอร์หรือใช้ ping raspberrypi.local)

- ใช้คำสั่ง

```
ssh pi@raspberrypi.local
```

รหัสผ่านเริ่มต้น: raspberry

v) ขั้นตอนที่ 4: อัปเดตระบบและตั้งค่าพื้นฐาน

1) อัปเดตแพ็คเกจ

```
sudo apt update && sudo apt upgrade -y
```

2)

```
sudo raspi-config
```

- ตั้งค่า Wi-Fi, เปิดใช้งาน I2C, SPI, Camera หรือสิ่งที่ต้องการ
- แล้วพิมพ์คำสั่งใน Command Line เพื่อที่จะดู ip ของ raspi ใช้คำสั่ง if config

3) เปลี่ยนรหัสผ่านเริ่มต้น

- passwd

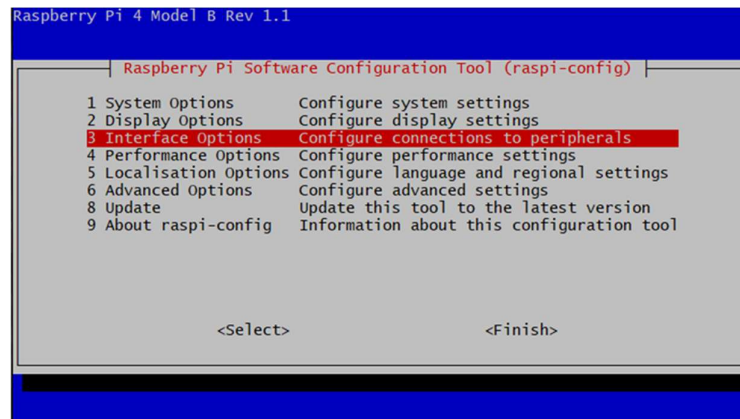
b) **ติดตั้ง LoraWan Gateway บน Raspberry Pi 4 สำหรับ WM1302**

vi) ต่อมาหลังจากเข้าสู่ระบบใน Raspberry Pi แล้วพิมพ์คำสั่งใน Command Line แล้วจะได้หน้าต่าง config raspi ดังนี้

```
sudo raspi-config
```

```
pi4gw@raspberrypi: ~  
login as: pi4gw  
pi4gw@192.168.1.208's password:  
Linux raspberrypi 6.6.74+rpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.6.74-1+rpt1 (2025-01-27) aarch64  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Mon Feb 17 17:57:15 2025  
pi4gw@raspberrypi:~$ sudo raspi-config  
pi4gw@raspberrypi:~$ sudo raspi-config
```

1) Select 'Interface Options'



- เลือก **SPI** จากนั้นเลือก **Yes** เพื่อเปิดใช้งาน
- เลือก **I2C** จากนั้นเลือก **Yes** เพื่อเปิดใช้งาน
- เลือก **Serial Port** หลังจากนั้นเลือก **No** เมื่อเจอคำถามนี้ Would you like a login shell... และ เลือก **Yes** เมื่อเจอคำถามนี้ Would you like the serial port hardware...
- เมื่อ ตั้งค่าเสร็จแล้ว เลือก **Finish** ให้ทำการ Reboot pi

- 2) ต่อมา ติดตั้ง git และ ดาวน์โหลด ไบบรารีและโปรแกรมสำหรับ SX1302 LoRa Gateway จาก GitHub พิมพ์คำสั่งนี้ใน Command Line

```
sudo apt update
sudo apt install -y git
cd ~
git clone https://github.com/Lora-net/sx1302\_hal
```

- 3) ย้ายไปยังsx1302_halโฟลเดอร์และคอมไพล์ทุกอย่าง

```
cd ~/sx1302_hal
make
```

- 4) แก้ไขreset pinสำหรับ SX1302 และ SX1261 ในreset_lgw.sh

```
nano tools/reset_lgw.sh
```

- 5) ในไฟล์ reset_lgw.sh ค่าเดิมจะเป็นอย่างนี้ ต้องเปลี่ยนเป็นขาของ wm1302 gateway SX1302_RESET_PIN=17 SX1261_RESET_PIN=5

```
# GPIO mapping has to be adapted with HW
SX1302_RESET_PIN=17 # SX1302 reset
SX1302_POWER_EN_PIN=18 # SX1302 power enable
SX1261_RESET_PIN=5 # SX1261 reset (LBT / Spectral Scan)
AD5338R_RESET_PIN=13 # AD5338R reset (full-duplex CN490
reference design
```

- 6) บันทึกการเปลี่ยนแปลงเหล่านี้โดยกด CTRL + x และ y สุดท้ายกด Enter เพื่อปิดตัวแก้ไขข้อความ

- 7) คัดลอกไฟล์ reset_lgw.sh ไปยัง packet_forwarder โฟลเดอร์

```
cp tools/reset_lgw.sh packet_forwarder/
cd packet_forwarder
```

- 8) แก้ไขไฟล์ nano global_conf.json.sx1250.AS923.USB

- ส่วนของ radio_0 ให้เปลี่ยน tx_freq_min=923200000 และ tx_freq_max=924200000

```

    },
    "radio_0": {
      "enable": true,
      "type": "SX1250",
      "freq": 923200000,
      "rssi_offset": -215.4,
      "rssi_tcomp": {"coeff_a": 0, "coeff_b": 0, "coeff_c": 20.41, "coeff_d": 2162.56, "coeff_e": 0},
      "tx_enable": true,
      "tx_freq_min": 923200000,
      "tx_freq_max": 924200000,
      "tx_power": {
        ("rf_power": 0, "pa_gain": 0, "pwr_idx": 0),
        ("rf_power": 12, "pa_gain": 0, "pwr_idx": 15),
        ("rf_power": 13, "pa_gain": 0, "pwr_idx": 16),
        ("rf_power": 14, "pa_gain": 0, "pwr_idx": 17)
      }
    }
  }
}

```

- ส่วนของ gateway_conf ให้เปลี่ยน ค่า gateway_ID= Gateway IDที่ได้ลงทะเบียนไว้กับchirpstack และ server_address= ip address Pi

```

"gateway_conf": {
  "gateway_ID": "918b90c956403cbd",
  "server_address": "172.20.10.4",
  "serv_port_up": 1700,
  "serv_port_down": 1700,
  "keepalive_interval": 10,
  "stat_interval": 30,
  "push_timeout_ms": 100,
  "forward_crc_valid": true,
  "forward_crc_error": false,
  "forward_crc_disabled": false,
}

```

หน้า chirpstack

	Last seen	Gateway ID	Name	Region ID	Region common name
Offline	2025-02-12 14:45:47	bfa7f979bca2a7d6	TESTGateway	as923	AS923
Online	2025-02-12 14:42:15	918b90c956403cbd	gw02	as923	AS923

link code gateway:

<https://github.com/MrArmiami/Project-install/blob/main/Install%20Gateway>

9) ทำการรันไฟล์ global_conf.json.sx1250.AS923.USB จากนั้นใช้คำสั่ง

lora_pkt_fwd เพื่อ LoraWan Gateway บน Raspberry Pi 4 สำหรับ WM1302

`./lora_pkt_fwd -c global_conf.json.sx1250.AS923`

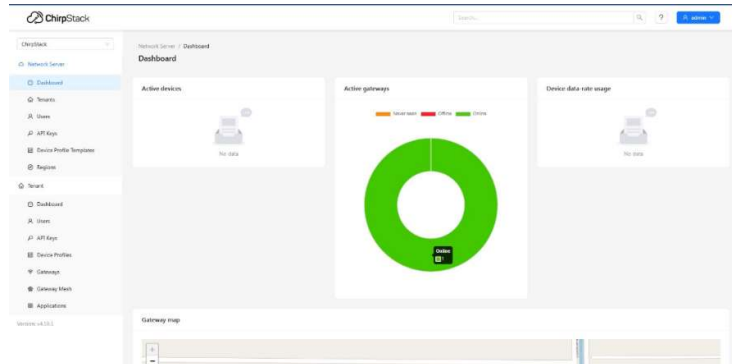
จะได้หน้าต่างประมาณนี้ บน terminal pi

```

*** Packet Forwarder ***
Version: 2.1.0
*** SX1302 HAL library version info ***
Version: 2.1.0;
***
INFO: Little endian host
INFO: found configuration file global_conf.json.sx1250.EU868, parsing it
INFO: global_conf.json.sx1250.EU868 does contain a JSON object named SX130x_conf
INFO: com_type SPI, com_path /dev/spidev0.0, lorawan_public 1, clksrc 0, full_duplex 0
INFO: antenna_gain 0 dBi
INFO: Configuring legacy timestamp
INFO: Configuring Tx Gain LUT for rf_chain 0 with 16 indexes for sx1250
INFO: radio 0 enabled (type SX1250), center frequency 867500000, RSSI offset -21
INFO: radio 1 enabled (type SX1250), center frequency 868500000, RSSI offset -21
INFO: Lora multi-SF channel 0> radio 1, IF -400000 Hz, 125 kHz bw, SF 5 to 12
INFO: Lora multi-SF channel 1> radio 1, IF -200000 Hz, 125 kHz bw, SF 5 to 12
INFO: Lora multi-SF channel 2> radio 1, IF 0 Hz, 125 kHz bw, SF 5 to 12
INFO: Lora multi-SF channel 3> radio 0, IF -400000 Hz, 125 kHz bw, SF 5 to 12
INFO: Lora multi-SF channel 4> radio 0, IF -200000 Hz, 125 kHz bw, SF 5 to 12
INFO: Lora multi-SF channel 5> radio 0, IF 0 Hz, 125 kHz bw, SF 5 to 12
INFO: Lora multi-SF channel 6> radio 0, IF 200000 Hz, 125 kHz bw, SF 5 to 12
INFO: Lora multi-SF channel 7> radio 0, IF 400000 Hz, 125 kHz bw, SF 5 to 12
INFO: Lora std channel> radio 1, IF -200000 Hz, 250000 Hz bw, SF 7, Explicit header
INFO: FSK channel> radio 1, IF 300000 Hz, 125000 Hz bw, 50000 bps datarate
INFO: global_conf.json.sx1250.EU868 does contain a JSON object named gateway_conf
INFO: gateway MAC address is configured to C69201F5402DBD24
INFO: server hostname or IP address is configured to "aul.cloud.thethings.network"

```


10) แล้วสถานะ gateway บน Chirpstack ก็จะขึ้นสถานะออนไลน์



b) ติดตั้ง Chirpstack Gateway OS บน Raspberry Pi 4

การติดตั้ง ChirpStack v4 Docker บน Raspberry Pi 4 ทำตามขั้นตอนนี้:

- 1) อัปเดตระบบแล้วติดตั้ง Docker บน terminal Pi

```
sudo apt update && sudo apt upgrade -y  
curl -fsSL https://get.docker.com | sudo sh  
sudo usermod -aG docker $USER2)
```

- 2) รีบูตเครื่อง เพื่อให้สิทธิ์ Docker ทำงาน บน terminal Pi -> sudo reboot
- 3) ติดตั้ง Docker Compose และ ตรวจสอบเวอร์ชันบน terminal Pi

```
sudo apt install -y python3-pip  
pip3 install docker-compose  
docker-compose version
```

- 4) ดาวน์โหลดไฟล์ Compose ของ ChirpStack แล้วเข้าไปที่ไฟล์ chirpstack-docker

```
git clone https://github.com/chirpstack/chirpstack-docker.git  
cd chirpstack-docker
```

- 5) แก้ไขไฟล์ docker-compose.yml โดยเลือก port ที่ไม่ชนกับ Pi

```
nano docker-compose.yml
```

- ส่วน chirpstack เปลี่ยนport เป็น.”8080:8080”

```
services:
  chirpstack:
    image: chirpstack/chirpstack:4
    command: -c /etc/chirpstack
    restart: unless-stopped
    volumes:
      - ./configuration/chirpstack:/etc/chirpstack
    depends_on:
      - postgres
      - mosquitto
      - redis
    environment:
      - MQTT_BROKER_HOST=mosquitto
      - REDIS_HOST=redis
      - POSTGRES_HOST=postgres
    ports:
      - "8080:8080"
```

Ports

- ต่อมาส่วนของ chirpstack-gateway-bridge เปลี่ยนport เป็น.1700:1700udp แล้วก็อย่าลืมเปลี่ยนภูมิภาค เป็น as923 ด้วย ตอนแรกที่ไฟล์เดิมจะกำหนด ไว้ อันเดิมคือ eu868

```
chirpstack-gateway-bridge:
  image: chirpstack/chirpstack-gateway-bridge:4
  restart: unless-stopped
  ports:
    - "1700:1700/udp"
  volumes:
    - ./configuration/chirpstack-gateway-bridge:/etc/chirpstack-gateway-bridge
  environment:
    - INTEGRATION_MQTT_EVENT_TOPIC_TEMPLATE=as923/gateway/{{ .GatewayID }}/event/{{ .EventType }}
    - INTEGRATION_MQTT_STATE_TOPIC_TEMPLATE=as923/gateway/{{ .GatewayID }}/state/{{ .StateType }}
    - INTEGRATION_MQTT_COMMAND_TOPIC_TEMPLATE=as923/gateway/{{ .GatewayID }}/command/#
  depends_on:
    - mosquitto
```

Ports

Region

- ต่อมาส่วนของ chirpstack-gateway-bridge-basicstation เลือกport “3001:3001” และเปลี่ยนภูมิภาคเป็น as923

```
chirpstack-gateway-bridge-basicstation:
  image: chirpstack/chirpstack-gateway-bridge:4
  restart: unless-stopped
  command: -c /etc/chirpstack-gateway-bridge/chirpstack-gateway-bridge-basicstation-as923.toml
  ports:
    - "3001:3001"
  volumes:
    - ./configuration/chirpstack-gateway-bridge:/etc/chirpstack-gateway-bridge
  depends_on:
    - mosquitto
```

- ต่อมาส่วนของ chirpstack-rest-api เลือกport “8090:8090”

```
chirpstack-rest-api:
  image: chirpstack/chirpstack-rest-api:4
  restart: unless-stopped
  command: --server chirpstack:8080 --bind 0.0.0.0:8090 --insecure
  ports:
    - "8090:8090"
  depends_on:
    - chirpstack
```

- ต่อมาส่วนของ postgres กับ redis ไม่ต้องตั้งค่าอะไร

```
postgres:
  image: postgres:14-alpine
  restart: unless-stopped
  volumes:
    - ./configuration/postgresql/initdb:/docker-entrypoint-initdb.d
    - postgresqldata:/var/lib/postgresql/data
  environment:
    - POSTGRES_USER=chirpstack
    - POSTGRES_PASSWORD=chirpstack
    - POSTGRES_DB=chirpstack

redis:
  image: redis:7-alpine
  restart: unless-stopped
  command: redis-server --save 300 1 --save 60 100 --appendonly no
  volumes:
    - redisdata:/data
```

- ต่อมาส่วนของ mosquito เลือก port “1884:1884”

```
mosquitto:
  image: eclipse-mosquitto:2
  restart: unless-stopped
  ports:
    - "1884:1884"
  volumes:
    - ./configuration/mosquitto/config:/mosquitto/config/

volumes:
  postgresqldata:
  redisdata:
```

- หลังจากตั้งค่าตามนี้หมดแล้วกดบันทึก (CTRL + X -> Y -> ENTER)
- 6) แก้ไขไฟล์ chirpstack.toml โดยใช้คำสั่ง nano บน terminal pi ดังนี้
- เข้าไปยังโฟลเดอร์ configuration โดยใช้คำสั่ง cd
cd configuration
 - เข้าไปยังโฟลเดอร์ chirpstack โดยใช้คำสั่ง cd
cd chirpstack
 - จากนั้นแก้ไขไฟล์ chirpstack.toml โดยเปลี่ยน region=as923 กับ port=1884

```
# Multiple regions can be enabled simultaneously. Each region must match
# the "name" parameter of the region configuration in "[regions]".
enabled_regions=[
  "as923",
]

# API interface configuration.
[api]

# interface:port to bind the API interface to.
bind="0.0.0.0:8080"

# Secret.
# This secret is used for generating login and API tokens, make sure this
# is never exposed. Changing this secret will invalidate all login and API
# tokens. The following command can be used to generate a random secret:
# openssl rand -base64 32
secret="you-must-replace-this"

[integration]
enabled=["mqtt"]

[integration.mqtt]
server="tcp://$MQTT_BROKER_HOST:1884/"
json=true
```

หลังจากตั้งค่าตามนี้หมดแล้วกดบันทึก (CTRL + X -> Y -> ENTER)

- เข้าไปยังโฟลเดอร์ chirpstack-gateway-bridge โดยใช้คำสั่ง cd
cd chirpstack-gateway-bridge

- แก้ไขไฟล์ chirpstack-gateway-bridge.toml โดยเปลี่ยน port=1884

```
GNU nano 7.2 chirpstack-gateway-bridge.toml
# See https://www.chirpstack.io/gateway-bridge/install/config/ for a full
# configuration example and documentation.

[integration.mqtt.auth.generic]
servers=["tcp://mosquitto:1884"]
username=""
password=""
```

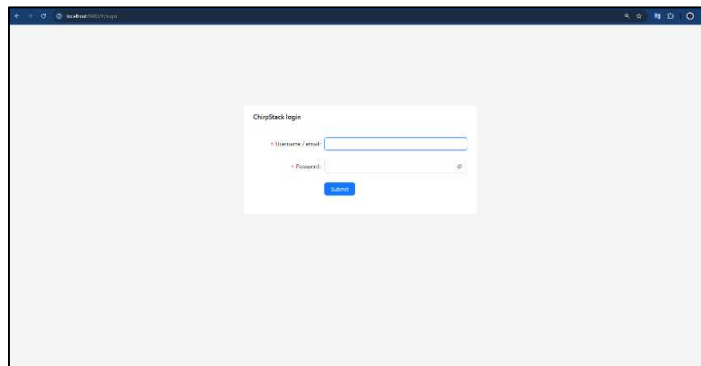
- หลังจากตั้งค่าตามนี้หมดแล้วกดบันทึก (CTRL + X -> Y -> ENTER) และ cd กลับไปที่หน้า chirpstack-docker
cd chirpstack-docker

- ทำการรันserver chirpstack docker v4 โดยใช้คำสั่งนี้

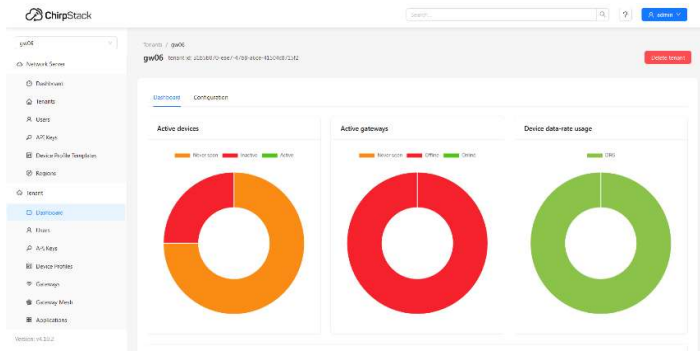
docker-compose up -d

```
pi@raspberrypi:~/chirpstack-docker$ docker-compose up -d
WARN[0000] Found orphan containers ([chirpstack-docker-chirpstack-gateway-bridge-as923-1]) for this project
. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphan
phans flag to clean it up.
[+] Running 7/7
 Container chirpstack-docker-postgres-1 Running 0.0s
 Container chirpstack-docker-redis-1 Running 0.0s
 Container chirpstack-docker-mosquitto-1 Running 0.0s
 Container chirpstack-docker-chirpstack-1 Running 0.0s
 Container chirpstack-docker-chirpstack-rest-api-1 Running 0.0s
 Container chirpstack-docker-chirpstack-gateway-bridge-1 Running 0.0s
 Container chirpstack-docker-chirpstack-gateway-bridge-basicstation-1 Running 0.0s
```

- เข้าสู่ระบบ ChirpStack Web Interface เข้าไปที่ http://localhost:8080/ หรือ http://IP-ADDRESS-OF-YOUR-PC:8080/ จะเจอหน้าจอเข้าสู่ระบบ
ข้อมูลสำหรับเข้าสู่ระบบ (ค่าเริ่มต้น)
Username: admin
Password: admin



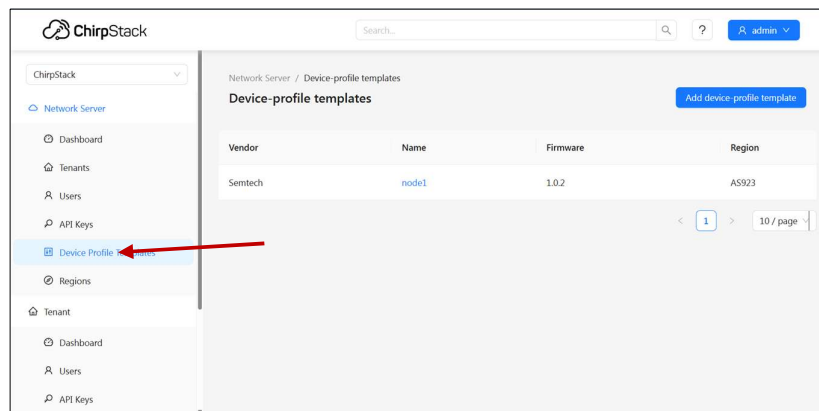
หลัง login เสร็จจะเป็นหน้า ChirpStack Web Interface



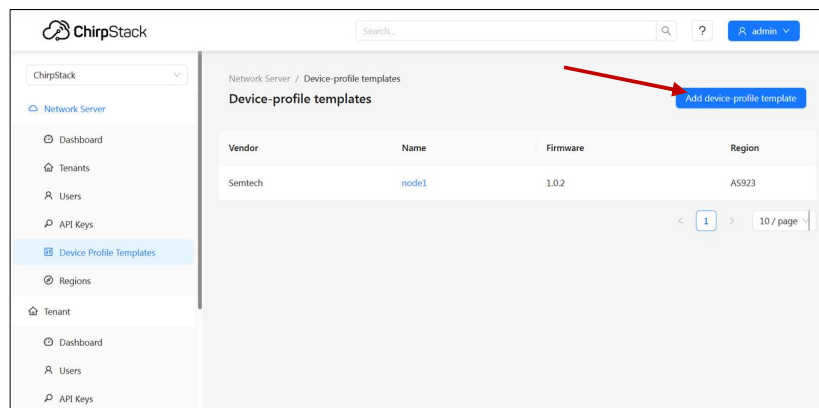
c) การ Add device ใน ChirpStack Web Interface

การเพิ่มอุปกรณ์ ใน ChirpStack Web Interface ทำตามขั้นตอนนี้:

- 1) Select -> Device-profile templates



- 2) Select -> Add device-profile templates



3) ตั้งค่า device-profile templates

The screenshot shows the 'Add device profile template' page in the AWS IoT console. The breadcrumb trail is 'Network Service / Device profile templates / Add'. The page title is 'Add device profile template'. The 'Region' is 'us-east-1'. The 'AWS region' is 'us-east-1'. The 'Device profile template' is 'IoTDeviceProfileTemplate'. The 'Add' button is highlighted with a red arrow.

4) ตั้งค่า device-profile Select -> Add device-profile

ChirpStack

Search...

admin

Tenants / ChirpStack / Device profiles

Device profiles

Add device profile

Name	Region	MAC version	Revision	Supports OTAA	Supports Class B	Supports Class C
node2	AS923	LoRaWAN 1.0.3	A	yes	no	no

< 1 > 10 / page

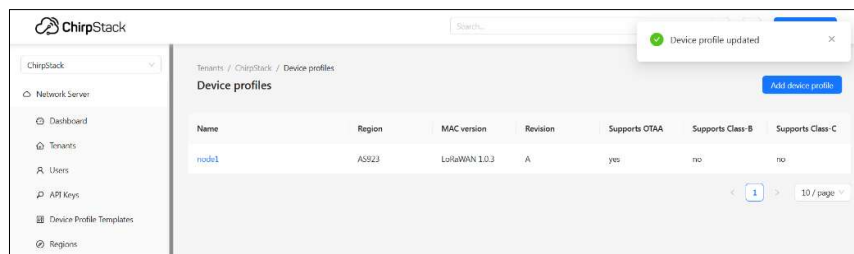
5) เลือก template ที่ได้สร้างไว้

The screenshot shows the ChirpStack web interface. A modal dialog titled "Select device-profile template" is open. It contains a table with the following data:

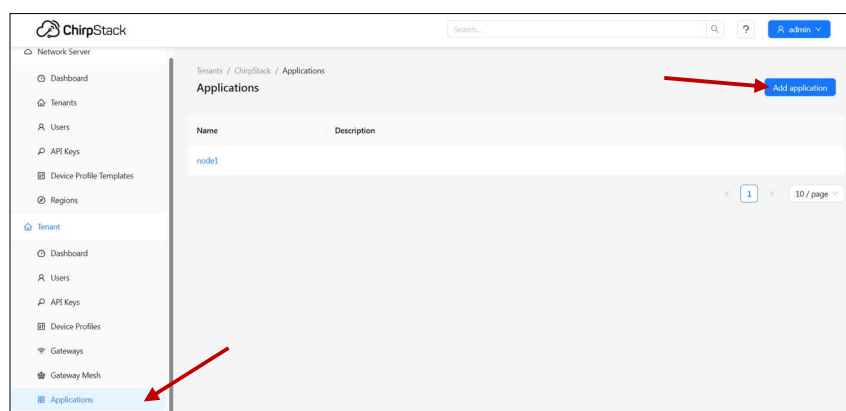
Manufacturer	Node ID	Firmware Version
Semtech	node1	FW version: 1.0.2 / AS923

A red arrow points to the "OK" button in the bottom right corner of the dialog.

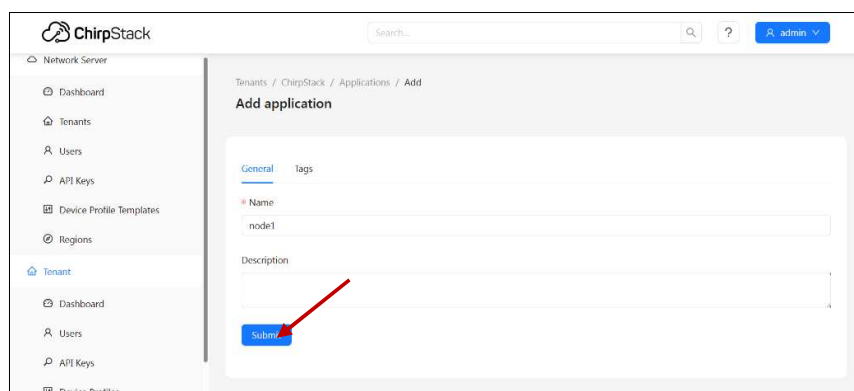
6) เมื่อกดไปแล้วก็จะเพิ่ม template device สำเร็จ



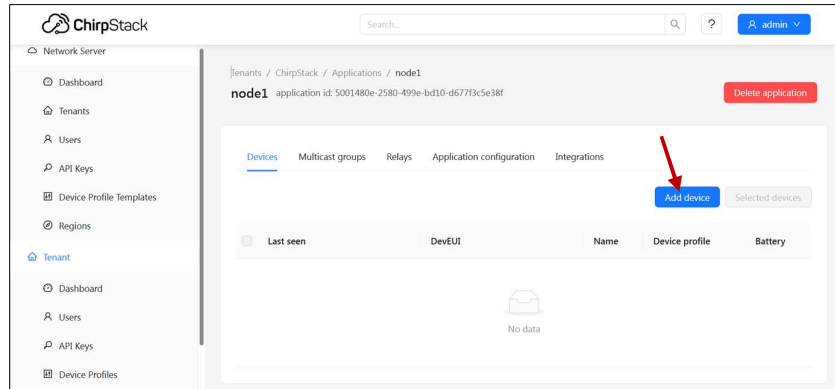
7) click -> Applications เพื่อที่จะ Add device-profile



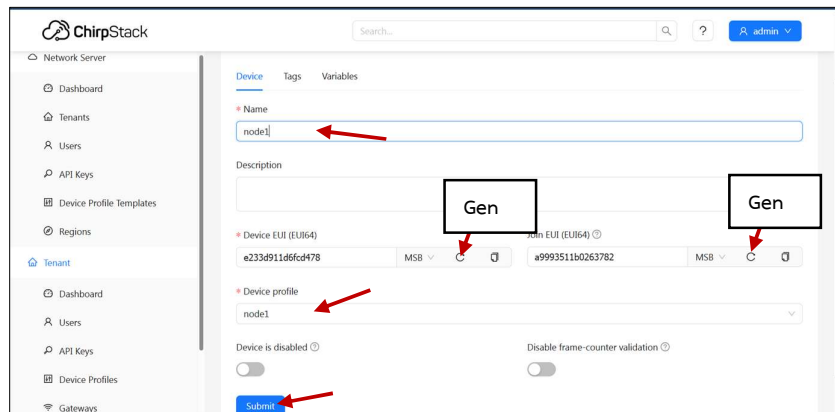
8) ตั้งชื่ออุปกรณ์



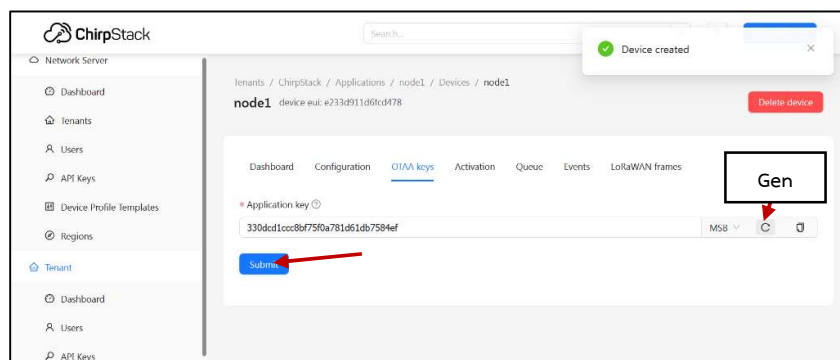
9) click -> Add device



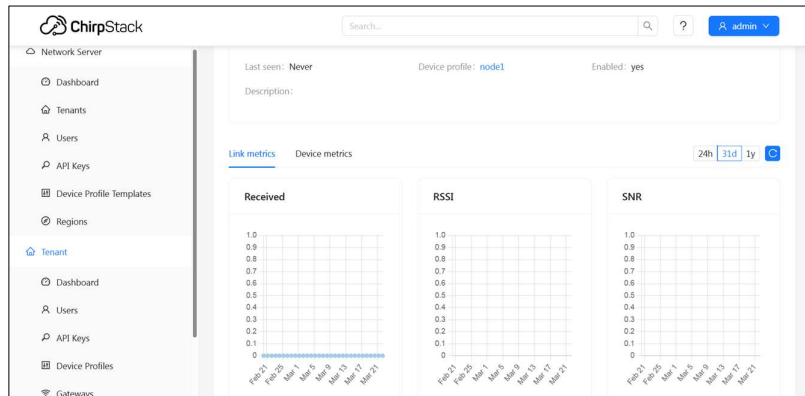
10) หน้า add device กด Generate Device EUI และ App EUI (แนะนำให้จดลง google sheet หรือ notepad) และเลือก Device profile เป็น ที่ได้สร้างไว้ข้างต้น จากนั้นกด submit



ต่อไป กด Generate App key แล้วกด submit



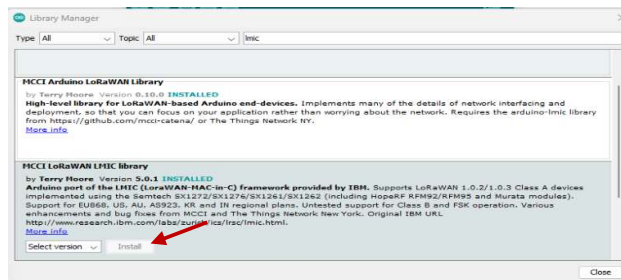
11) จะได้หน้าต่างอุปกรณ์ที่ลงทะเบียนกับchirpstack แล้ว



เขียนโค้ดตัวโหนดที่ต้องการจะเชื่อมต่อ แบบ OTAA กับ ChirpStack Web Interface code

clickhere: <https://github.com/MrArmiami/Project-install/blob/main/Add%20device%20Chirpstack>

- ลงไลบรารี MCCI_LoRaWAN_LMIC_library ได้ที่ Tools->Manage Libraries->พิมพ์ Imic ->เลือกMCCI LoRaWAN LMIC library



กดInstall

- ต่อไปแก้ไขไฟล์ lmic_project_config.h ที่โฟลเดอร์ MCCI_LoRaWAN_LMIC_libraryเพื่อให้ตัวnode สามารถส่งสัญญาณไปหา WM1302Gateway และ chirpstack ได้
path : MCCI_LoRaWAN_LMIC_library-> project_config-> lmic_project_config.h เปิดใช้งาน Region as923 TH

```
C:\Users\armko\OneDrive\Documents\Arduino\libraries\MCC10aRiWAN\LMIC_library\project_config > C:\mic_project_config
1 // project-specific definitions
2 #define CFG_eu868 1
3 // #define CFG_us915 1
4 // #define CFG_us915 1
5 #define CFG_as923 1
6 // #define LMIC_COUNTRY_CODE LMIC_COUNTRY_CODE_JP /* for as923-JP; also define CFG_as923 */
7 // #define CFG_kr920 1
8 // #define CFG_in866 1
9 #define CFG_x1276_radio 1
10 // #define CFG_x1276_radio 1
11 // #define CFG_x1262_radio 1
12 // #define ARDUINO_heltec_wifi_lora_32_V3
13 // #define LMIC_USE_INTERRUPTS
```

กด SAVE

- ต่อมาในส่วนของโค้ด APPEUI, DEVEUI, และ APPKEY ที่ได้จัดไว้ก่อนหน้านี้ ทำการแปลงค่าตามลำดับไบนารี (Endianness Conversion) โดยค่าที่อยู่ในรูปแบบ big-endian (จากซ้ายไปขวา) แปลงเป็น little-endian (จากขวาไปซ้าย) สำหรับการใช้งานใน Arduino หรือ LoRaWAN library ที่ต้องการให้ไบนารีเรียงลำดับใหม่ แปลงแค่ APPEUI กับ DEVEUI จากที่ได้ลงทะเบียบไว้คือ

APPEUI= a9ce7f0dbcd3ec22

แปลงค่า APPEUI ได้ = { 0x22, 0xEC, 0xD3, 0xBC, 0x0D, 0x7F, 0xCE, 0xA9 }

DEVEUI= 07f9bab5f97a755e

แปลงค่า DEVEUI ได้ = { 0x5E, 0x75, 0x7A, 0xF9, 0xB5, 0xBA, 0xF9, 0x07 }

APPKEY= e72f8b4b99b9cc5941d7e47da6e371be

ไม่ได้แปลงแต่ต้องนำมาไว้ในอาร์เรย์เหมือนกัน

```
= { 0xE7, 0x2F, 0x8B, 0x4B, 0x99, 0xB9, 0xCC, 0x59, 0x41, 0xD7,
    0xE4, 0x7D, 0xA6, 0xE3, 0x71, 0xBE }
```

นำค่าที่ได้มาใส่ในส่วนของ Join OTAA ของโค้ด

[illegible]

- จากนั้นกำหนดขา ตัว lorawan Module ตอนนีใช้ sx1276 923 Mhz

[illegible]

- ส่วนของการปรับค่าการส่งข้อมูลทุกๆ TX_INTERVAL = 10;

[illegible]

- ส่วนของข้อมูลที่จะส่งไปยัง chirpstack

```
// ArduinoProMiniV220204
// IMC1.cpmode & DP_V00000000
if (Serial.print(F("Top ->IMC1000", not_sendin*))}

} else {
    // Datasheet IMC100
    float temperature = bme.readTemperature();
    float humidity = bme.readHumidity();

    // Datasheet IMC101
    float baroVoltage = analogRead(Voltage_V1);
    float current_mA = analogRead(Potentiometer_mA1);
    float loadVoltage = analogRead(Voltage_V2) / 2100.0;

    int analogValue = analogRead(CELL_MOISTURE_ANALOG_PIN);
    int voltMeasurePercent = map(analogValue, 0, 9, 0, 100);

    // Data received
    send_payload(1);
    sprintf(payload, "%f,%f,%f,%f,%f,%f,%f,%f,%f,%f", temperature, humidity, loadVoltage, current_mA, soilMoisturePercent);

    // Send data to IMC
    IMC.sendData2((uint8_t*)payload, sizeof(payload));
    Serial.print(F("Task completed"));
    Serial.println(payload);
}
```

- เมื่อปรับโค้ดเสร็จแล้ว กด Upload

- ผลที่ได้บน Serial monitor

```
COMS
19:52:56.354 -> mode:DIO, clock div:1
19:52:56.354 -> load:0x3ffff0030, len:4832
19:52:56.354 -> load:0x40078000, len:16460
19:52:56.354 -> load:0x40080400, len:4
19:52:56.354 -> load:0x40080404, len:3504
19:52:56.354 -> entry 0x400805cc
19:52:56.492 -> Starting
19:52:56.680 -> Packet queued: T:30.18,H:70.29,V:0.88,I:-0.40,S:0.00
19:53:06.668 -> OP_TXRPEND, not sending
19:53:16.654 -> Packet queued: T:30.21,H:69.71,V:0.88,I:-0.40,S:0.00
19:53:26.677 -> Packet queued: T:30.19,H:69.61,V:0.88,I:-0.10,S:0.00
19:53:36.661 -> Packet queued: T:30.19,H:69.54,V:0.88,I:-0.40,S:0.00
19:53:46.698 -> Packet queued: T:30.22,H:69.49,V:0.88,I:-0.40,S:3.08
19:53:56.690 -> Packet queued: T:30.24,H:69.40,V:0.88,I:-0.50,S:1.74
```

- ผลที่ได้บน Wm1302 Gateway

Gateway รับข้อมูลจากโหนด แล้วส่งขึ้น chirpstack

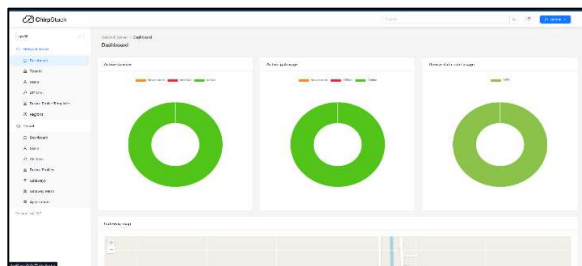
```

server: .address = eui.cloud.thethings.network; .port = 1700; .enable = 1
server: .address = 192.168.1.34; .port = 1680; .enable = 0
Gateway Config
  LORA Gateway
  Single Channel Gateway
  Latitude=9.80
  Longitude=100.49
  Altitude=10
Trying to detect module with NSS=6 DI00=7 Reset=0 Led1=4
SK1276 detected, starting.
Gateway ID: dc:a6:32:ff:ff:0e:
Listening at SF7 on 923.200000 Mhz.

stat update: 2021-06-12 12:27:10 GMT no packet received yet
Packet RSSI: -40, RSSI: -111, SNR: 9, Length: 35 Message: '0j...&.....V...T
...{&...
rxpk update: [{"rxpk":{"tmst":3194625058,"freq":923.2,"chan":0,"rfch":0,"stat":1,
"modu":"LORA","data":"SF7BW125","codr":4/5,"rssi":-40,"lsnr":9.0,"size":35,"d
ata":"QGoVH5VH3RoBqh7VyVtZwBJ+9dLbSZ1bkqgx4yJxWMihgr4="}}]}

```

- ผลที่ได้บน chirpstack
device จะขึ้นสถานะ online



แล้วdata ที่ได้รับจาก gateway จะมี ดังนี้

```

deviceName: "bme"
devEui: "af78cb2a63d020dd"
deviceClassEnabled: "CLASS_A"
tags: {} 0 keys
devAddr: "006596b2"
adr: true
dr: 5
fCnt: 17
fPort: 1
confirmed: false
data: "VDoyMS44O5xIOjYwJjc1"
rxInfo: {} 1 item
  0: {} 10 keys
    gatewayId: "918b90c956403cbd"
    uplinkId: 54785
    nsTime: "2025-01-14T18:55:15.571589359+00:00"
    rssi: -65
    snr: 13
    channel: 3
    location: {} 2 keys
      latitude: 14.08629832648065
      longitude: 100.73072433471681
    context: "g2usQA=="
    metadata: {} 2 keys
      region_common_name: "AS923"
      region_config_id: "as923"
    crcStatus: "CRC_OK"

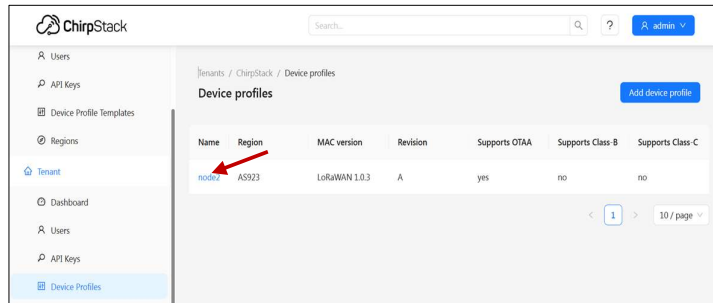
```

data จะถูกเข้ารหัสในรูปแบบ Base64 จึงต้องสร้าง function()decode เพื่อแปลงข้อมูลที่gatewayส่งมาให้อยู่ในรูปแบบตัวเลขปกติ

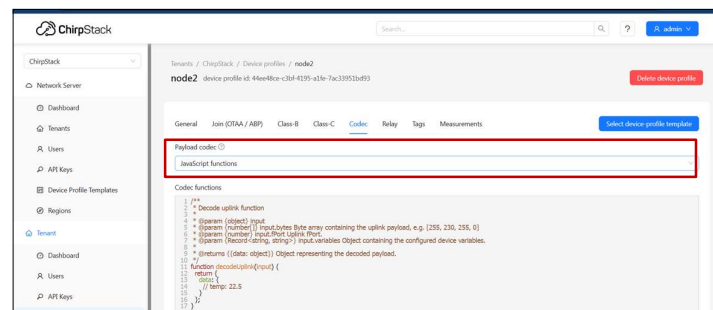
function() code : [https://github.com/MrArmiami/Project-install/blob/main/function\(%20\)Decode](https://github.com/MrArmiami/Project-install/blob/main/function(%20)Decode)

ขั้นตอนการเขียน function()decode

- 1) คลิก -> Device Profile
- 2) เลือก -> node2



- 3) คลิก Codec แล้วเลือก Javascript functions จากนั้นเขียนโค้ดที่ตามให้
เป็นอันเสร็จสิ้นในการเขียน functions decode บน Chirpstack



หลังจากสร้าง functions decode เสร็จจะได้ข้อมูลที่เป็นตัวเลขมา



- เป็นการAdd อุปกรณ์ เสร็จสิ้น

- ADD device เพิ่มอีก 7 ชุดตามวิธีด้านข้างต้น

d) **ติดตั้ง Local Server บน Raspberry Pi 4**

สร้าง Local Server บน Raspberry Pi ที่รันบนพอร์ต 5000 ได้โดยใช้ **Flask** ใน Python ตามขั้นตอนนี้:

- 1) ติดตั้ง Flask โดยใช้ terminal pi พิมพ์คำสั่ง
pip install flask

- 2) สร้างไฟล์ app.py

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def home():
    return "Hello, Raspberry Pi!"
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

- 3) รันเซิร์ฟเวอร์
python app.py
- 4) เป็นการเสร็จสิ้นการติดตั้ง