



# Low Power Wide Area (LPWA)

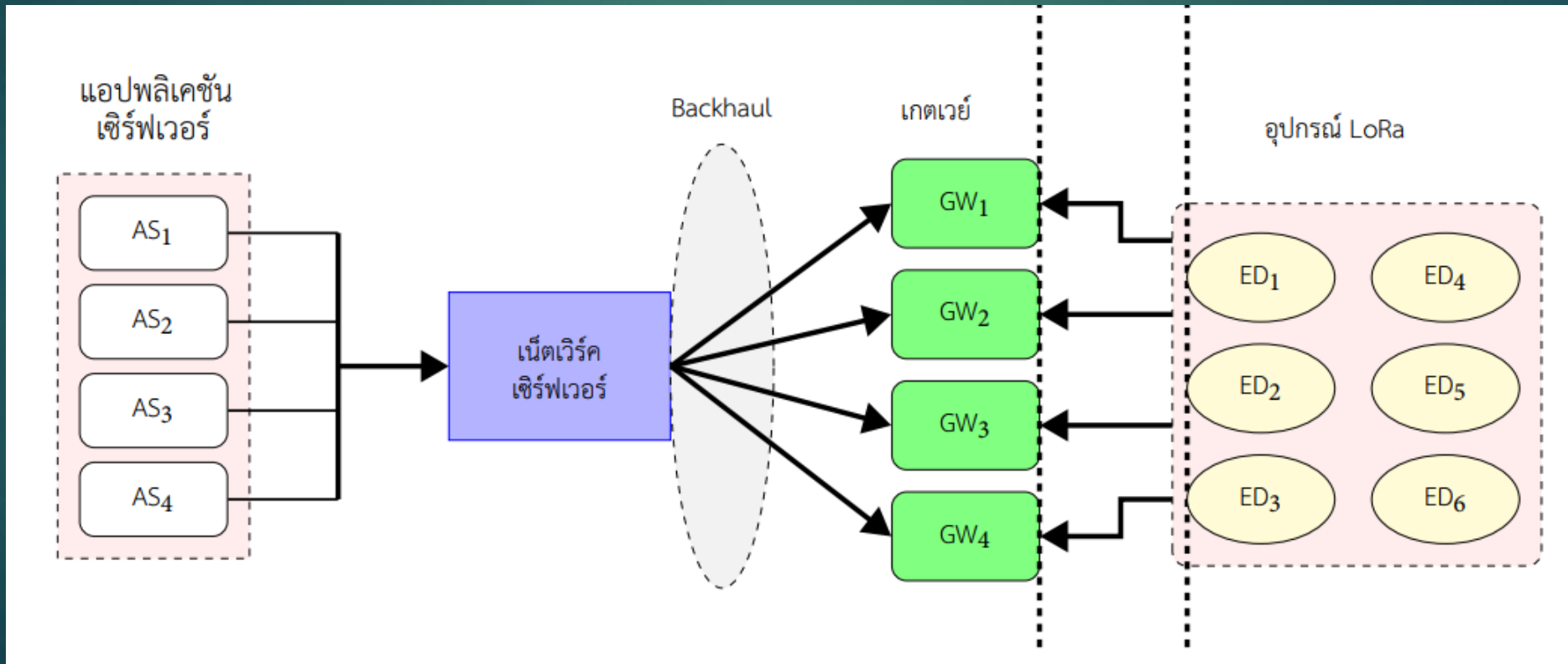
# LoRa

## ▶ LoRa

- ▶ การสื่อสารในช่วง 1 GHz
- ▶ การมอดูเลตด้วยเทคนิคเชิร์ปสเปกตรัม (Chirp Spread Spectrum: CSS)
- ▶ เนื่องจากการส่งที่ได้ในระยะไกล และสามารถป้องกันการรบกวนได้ดี (Interference Robustness)

# สถาปัตยกรรมเน็ตเวิร์ก LoRa

- ▶ ประกอบด้วย 2 ส่วนหลัก คือ
  - ▶ LoRaWAN
  - ▶ LoRa Node



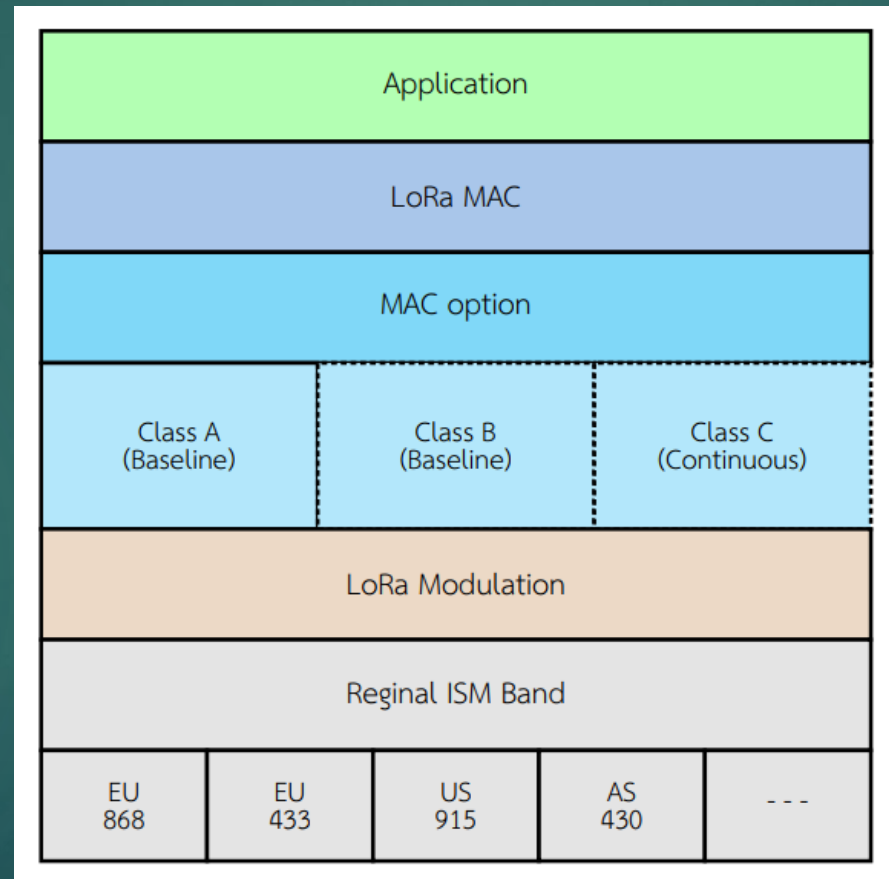
# ช่วงความถี่

LoRa

WiFi, Bluetooth, Zigbee



# LoRa Protocol Stack



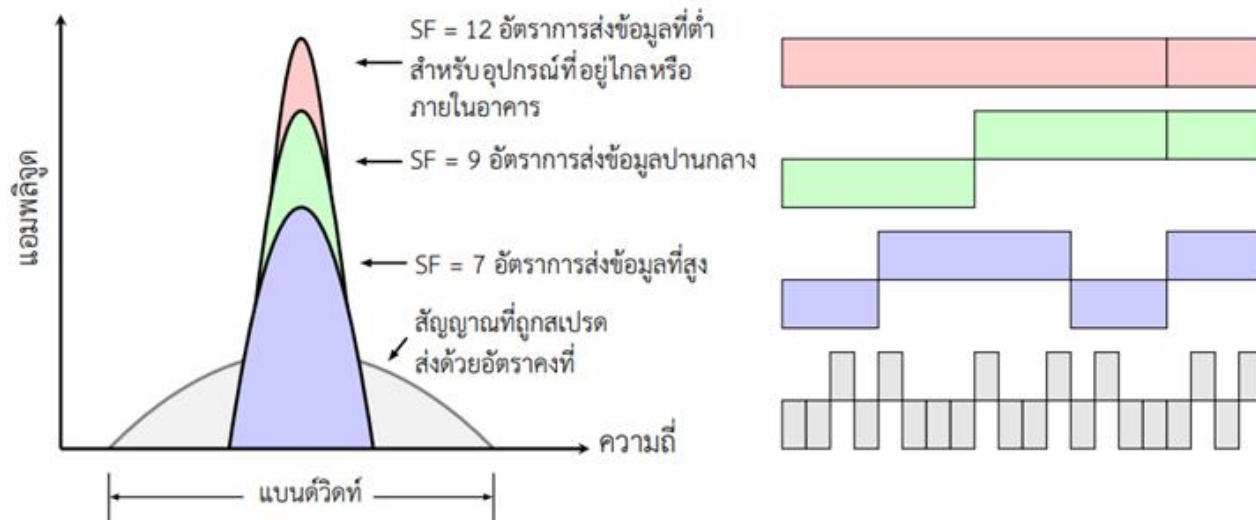
# Data Rate

$$R_b = SF \times \frac{BW}{2^{SF}} \times CR$$

- $R_b$  เป็นอัตราเร็วในการส่งข้อมูล

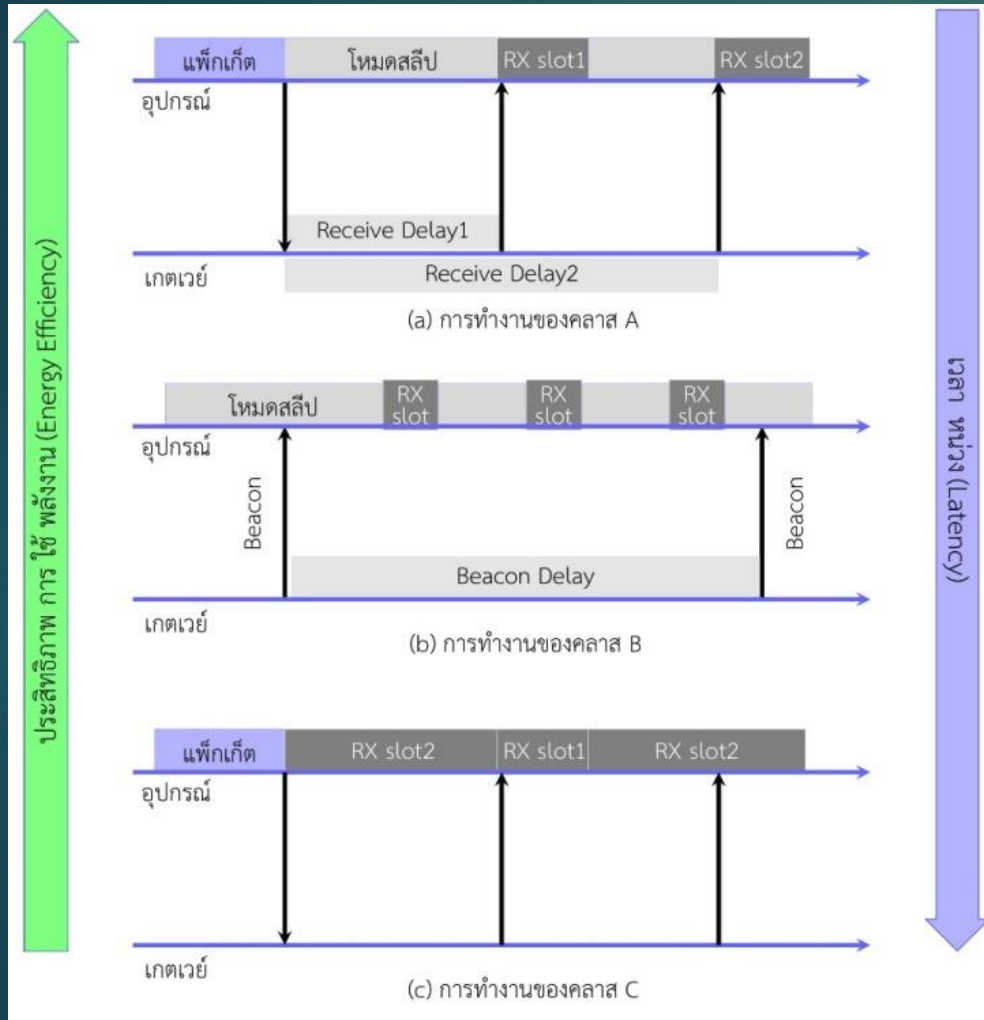
แบนด์วิธ	SF = 7	SF = 8	SF = 9	SF = 10	SF = 11	SF = 12
125 KHz	-123	-126	-129	-132	-133	-136
250 KHz	-120	-123	-125	-128	-130	-133
500 KHz	-116	-119	-122	-125	-128	-130

# การส่งข้อมูล - ระยะทาง



สเปรตแฟกเตอร์ (SF)	อัตราความเร็วของการส่งข้อมูล (Bit Rate: bps)	ระยะทาง (กิโลเมตร)
SF7	5,470	2
SF8	3,125	4
SF9	1,760	6
SF10	980	8
SF11	440	10
SF12	290	14

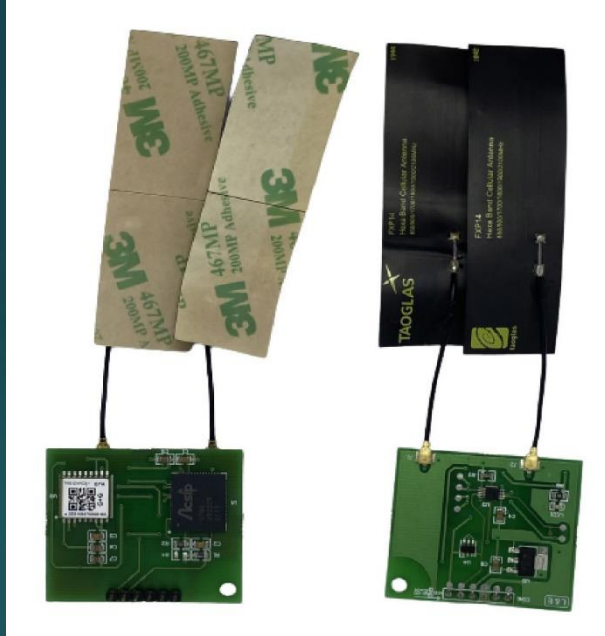
# โปรโตคอล LoRaWAN



- ▶ **คลาส A:** ใช้พลังงานต่ำที่สุด แต่มีข้อเสียคือการสื่อสารที่ไม่ยืดหยุ่นเท่าที่ควร การสื่อสารของคลาส A แสดงในรูปแบบที่ (A)
- ▶ **คลาส B:** เป็นการสื่อสารแบบสองทาง (Bi-directional) พร้อมการใช้เบคอน (Beacon) จากเกตเวย์เพื่อให้เน็ตเวิร์กเซิร์ฟเวอร์ทราบว่าเมื่อใดที่อุปกรณ์ปลายทางพร้อมที่จะรับ ดังแสดงในรูปแบบที่ 10.17 (B)
- ▶ **คลาส C:** กำหนดให้มีช่วงเวลารับข้อมูลแบบต่อเนื่อง (Continuous) ดังแสดงในรูปแบบที่ 10.17 (C) ทำให้มีการใช้พลังงานมากที่สุด



# การทำงานอุปกรณ์

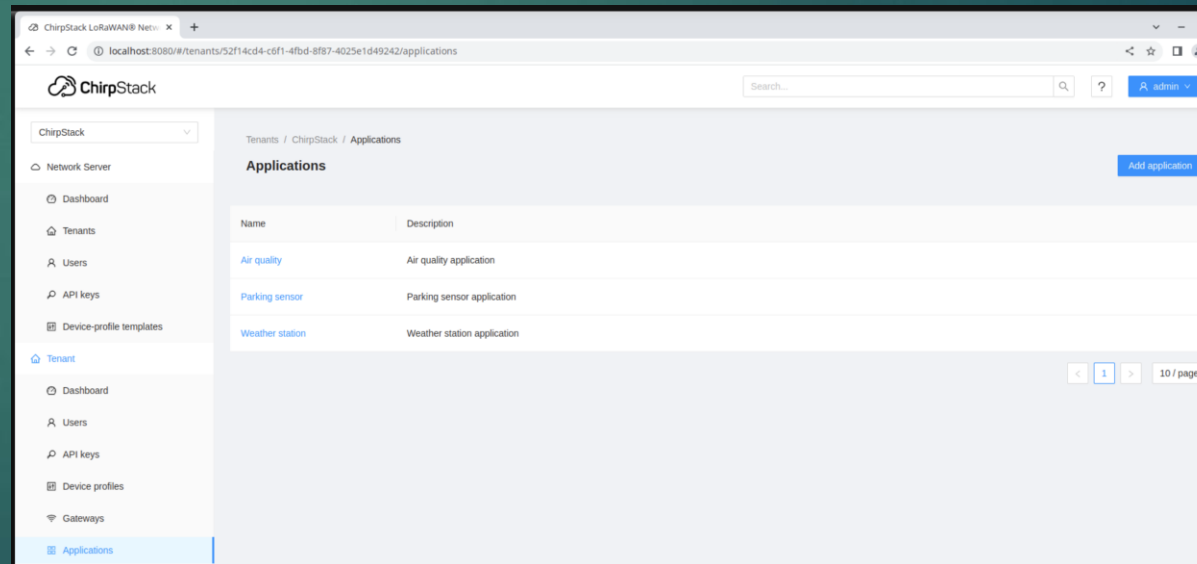


โมดูล LoRa

จากบริษัท แอล แอนด์ อี แมนูแฟคเจอริ่ง จำกัด



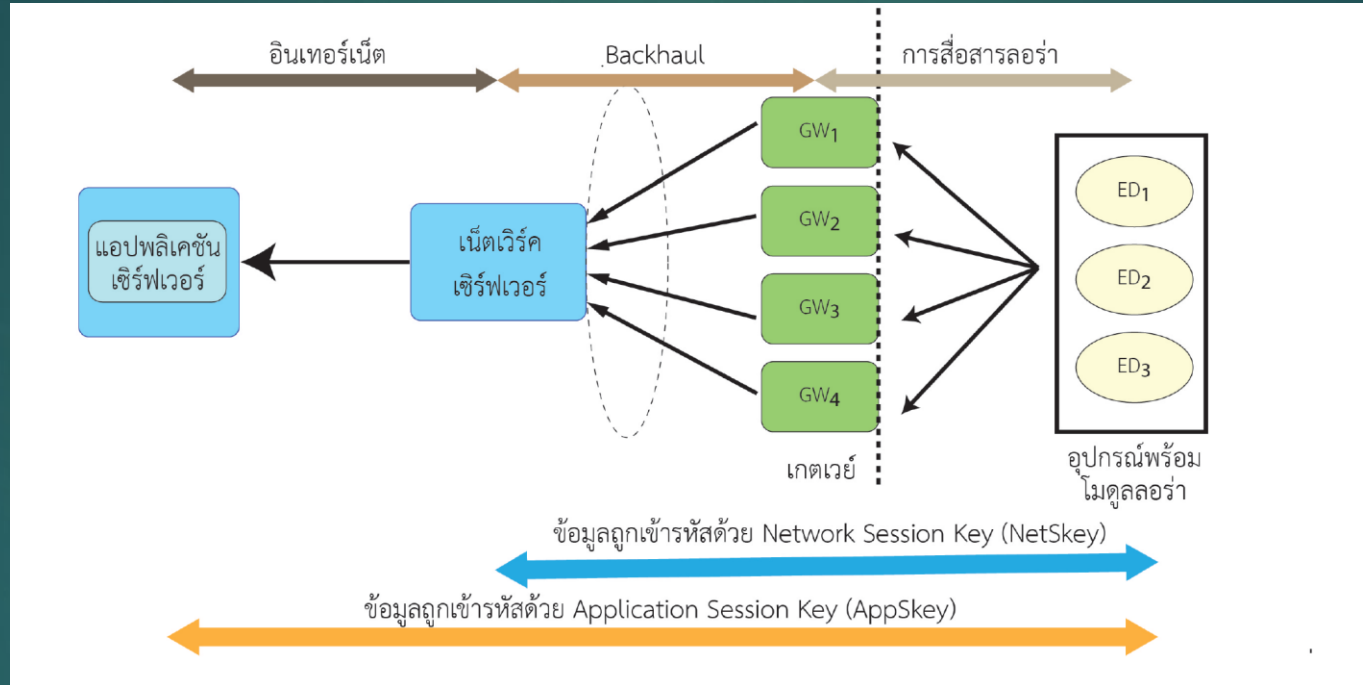
Gateway



LoRaWAN

# ขั้นตอนการสร้างระบบ

- ▶ 1. ติดตั้ง ChirpStack บน PC หรือ Raspberry Pi
- ▶ 2. กำหนด Gateway ที่จะใช้ให้กับระบบ
- ▶ 3. นำโหนดที่จะใช้เข้าสู่ระบบ



- ▶ 1. Application Session Key (AppSKey) เพื่อใช้สำหรับการเข้ารหัสข้อมูล
- ▶ 2. Network Session Key (NwkSKey) เพื่อใช้สำหรับการพิสูจน์ (Authentication) ทราบตัวตนระหว่างอุปกรณ์ LoRaWAN กับเน็ตเวิร์กเซิร์ฟเวอร์

# การเปิดใช้งานอุปกรณ์

- ▶ 1. **Activation By Personalisation (ABP)** เป็นการเปิดใช้งานโดยผู้ใช้กำหนดหมายเลขอุปกรณ์ที่จะใช้ (DevAddr), NwkSKey และ AppSKey
- ▶ 2. **Over-The-Air-Activation (OTAA)** เป็นการเปิดใช้โดยให้เครือข่ายกำหนดค่าคือ AppSKey และ NwkSKey ขึ้นเมื่อต้องการนำอุปกรณ์มาใช้งาน

# การใช้งานแบบ OTAA

- ▶ ผู้ใช้ต้องทราบถึงค่าที่เครือข่ายต้องการ ได้แก่ DecEUI, AppEUI และ AppKey
- ▶ DevEUI (ขนาด 8 ไบต์, EUI64) เป็นหมายเลขประจำตัวของอุปกรณ์
- ▶ AppEUI/JoinEUI (ขนาด 8 ไบต์, EUI64) สำหรับการการพิสูจน์ทราบตัวตนของ

# การพัฒนา

- ▶ 1.กำหนดโปรไฟล์ของอุปกรณ์
  - ▶ General
  - ▶ Join (OTAA / ABP)
  - ▶ Class-C

# General

- ▶ กำหนดชื่อ
- ▶ กำหนด Region

Tenants / chatchai khunboa / Device profiles / Add

### Add device profile

General Join (OTAA / ABP) Class-B Class-C Codec Tags Measurements [Select device-profile template](#)

กำหนดชื่อตามที่ต้องการ

\* Name  
IoT\_micropython

Description  
A profile example for book

AS=923 สำหรับการใช้งานในประเทศไทย

\* Region  
AS923

Region configuration  
AS923

\* MAC version  
LoRaWAN 1.0.3

\* Regional parameters revision  
A

\* ADR algorithm  
Default ADR algorithm (LoRa only)

Flush queue on activate  
☒

\* Expected uplink interval (secs)  
3600

Device-status request frequency (req/day)  
1

[Submit](#)

# Join(OTAA / ABP)

[General](#) [Join \(OTAA / ABP\)](#) [Class-B](#) [Class-C](#) [Codec](#) [Tags](#) [Measurements](#)

---

Device supports OTAA

☒

[Submit](#)



# Class-C

[General](#) [Join \(OTAA / ABP\)](#) [Class-B](#) [Class-C](#) [Codec](#) [Tags](#) [Measurements](#)

Device supports Class-C

☒

\* Class-C confirmed downlink timeout (seconds) [?](#)

^

v

Submit

# การเพิ่มแอปพลิเคชัน

## ▶ เพิ่มแอปพลิเคชันโปรไฟล์


\*

Name

IoT Applicatoion

Description

IoT Book Applications



Submit

# เพิ่มอุปกรณ์

Tenants / chatchai khunboa / Applications / IoT Applicatoion

**IoT Applicatoion** application id: ba09c8a1-8456-4869-9283-9d50b22d4d70

Delete application

Devices

Multicast groups

Application configuration

Integrations

Add device

Selected devices

<input type="checkbox"/>	Last seen	DevEUI	Name	Device profile	Battery
--------------------------	-----------	--------	------	----------------	---------

# เพิ่มอุปกรณ์ไปยังแอปพลิเคชันที่กำหนดขึ้น

Device Tags Variables

\* Name

Lora\_with\_GNSS

Description

\* Device EUI (EUI64)

9c65f9fffe42ec2c MSB C

\* Device profile

IoT\_micropython

Device is disabled ?

Disable frame-counter validation ?

Submit



ปรากฏ

Dashboard Configuration OTAA keys Activation Queue Events LoRaWAN frames

\* Application key ?

..... MSB C

Submit

# อุปกรณ์ลอราโมดูล

- ▶ โมดูล S76S เป็นชิปที่ถูกพัฒนาขึ้นภายในประกอบด้วย  
โมดูลการสื่อสารลอรา SX1276 ของบริษัท Semtech และ  
ไมโครคอนโทรลเลอร์พลังงานต่ำขนาด 32 บิต STM32L07x จาก  
STMicroelectronics



# คำสั่งที่ทางบริษัท AcSip (สำหรับ OTAA)

คำสั่ง	รายละเอียด
sip reset	รีเซ็ตตัวอุปกรณ์เหมือนออกจากโรงงาน
mac set_ch_freq <Id> <ความถี่>	กำหนดช่องสัญญาณและความถี่ที่ใช้
mac set_deveui <DevEUI>	กำหนดหมายเลข EUI ของอุปกรณ์ขนาด 8 ไบต์
mac get_deveui	ตรวจสอบค่า Device EUI
mac set_appkey <AppKey>	กำหนด Application Key ขนาด 16 ไบต์
mac get_appkey	ตรวจสอบค่า Application key
mac join <โหมด>	กรณีเป็น OTAA ให้ใส่ค่า OTAA ที่โหมด
mac get_join_status	แสดงสถานะ joined หรือ unjoined
mac set_deveui <DevEUI>	กำหนดช่องหมายเลข EUI ของอุปกรณ์
mac set_class <Class>	กำหนดให้ทำงานคลาส A หรือ C
mac save	จัดเก็บค่าที่กำหนดลงในหน่วยความจำ


# เริ่มต้น Reset บอร์ด


- ▶ `>>> uart = UART(2, baudrate=115200, bits=8, parity=None, stop=1, timeout=200)`
- ▶ `>>> uart.write("sip reset")`
- ▶ `>>> uart.write("mac get_deveui")`
- ▶ `>>> print(uart.read().decode(utf8))`
- ▶ `>> 9c65f9fffe42ec2c`
- ▶ `>>> uart.write("mac get_appkey")`
- ▶ `>>> print(uart.read().decode(utf8))`
- ▶ `>> b9b79a*****152e58`

# ขั้นตอนการเริ่มต้นใช้งาน

- ▶ 1. กำหนดช่องความถี่ที่ใช้
- ▶ 2. กำหนดการสื่อสารแบบคลาส C (Class C)
- ▶ 3. กำหนดค่า APPKEY
- ▶ 4. สั่ง Join แบบ OTAA
- ▶ 5. ตรวจสอบสถานะการ Join
- ▶ 6. ทดสอบส่งข้อมูล



- 
- ▶ `>>> from machine import UART, Pin`
  - ▶ `>>> import time`
  - ▶ `>>> uart = UART(2, baudrate=115200, bits=8, parity=None, stop=1,`
  - ▶ `timeout=100)`
  - ▶ `>>> uart.write("mac set_ch_freq 0 923200000")`
  - ▶ `>>> uart.write("mac set_ch_freq 1 923400000")`
  - ▶ `>>> uart.write("mac set_ch_freq 2 922000000")`
  - ▶ `>>> uart.write("mac set_rx2 2 923200000")`
  - ▶ `>>> print(uart.read().decode('utf-8'))`
  - ▶ `>>> uart.write("mac set_class C")`
  - ▶ `>>> print(uart.read().decode('utf-8'))`

- 
- ▶ `>>> uart.write("mac set_appkey  
c88e32c03eaf420faba87a4b1156e804")`
  - ▶ `>>> print(uart.read().decode('utf-8'))`
  - ▶ `>>> uart.write("mac join_ota")`
  - ▶ `>>> print(uart.read().decode('utf-8'))`
  - ▶ `>> accepted`
  - ▶ `>>> uart.write("mac get_join_status")`
  - ▶ `>>> print(uart.read().decode('utf-8'))`
  - ▶ `>> joined`
  - ▶ `>>> uart.write("mac tx_ucnf 2 1234")`
  - ▶ `>>> print(uart.read().decode('utf-8'))`
  - ▶ `>> tx_ok`

# ส่งข้อมูล

```
max tx <Data Type> <Port Number> <Data>
```

```
import esp32
from machine import UART, Pin
from binascii import hexlify
import time

data = "hall= {:d}, temp = {:.2f}"
uart = UART(2, baudrate=115200, bits=8, parity=None, stop=1,
            timeout=200)

i = 0
for i in range(2):
    lora_data = data.format(esp32.hall_sensor(), esp32.
                            raw_temperature())
    hexa_data = hexlify(lora_data).decode("utf-8")
    uart.write("mac tx ucnf 2 {}".format(hexa_data))
    print(lora_data)
    time.sleep(5)
```