**ใบงานที่ 6**

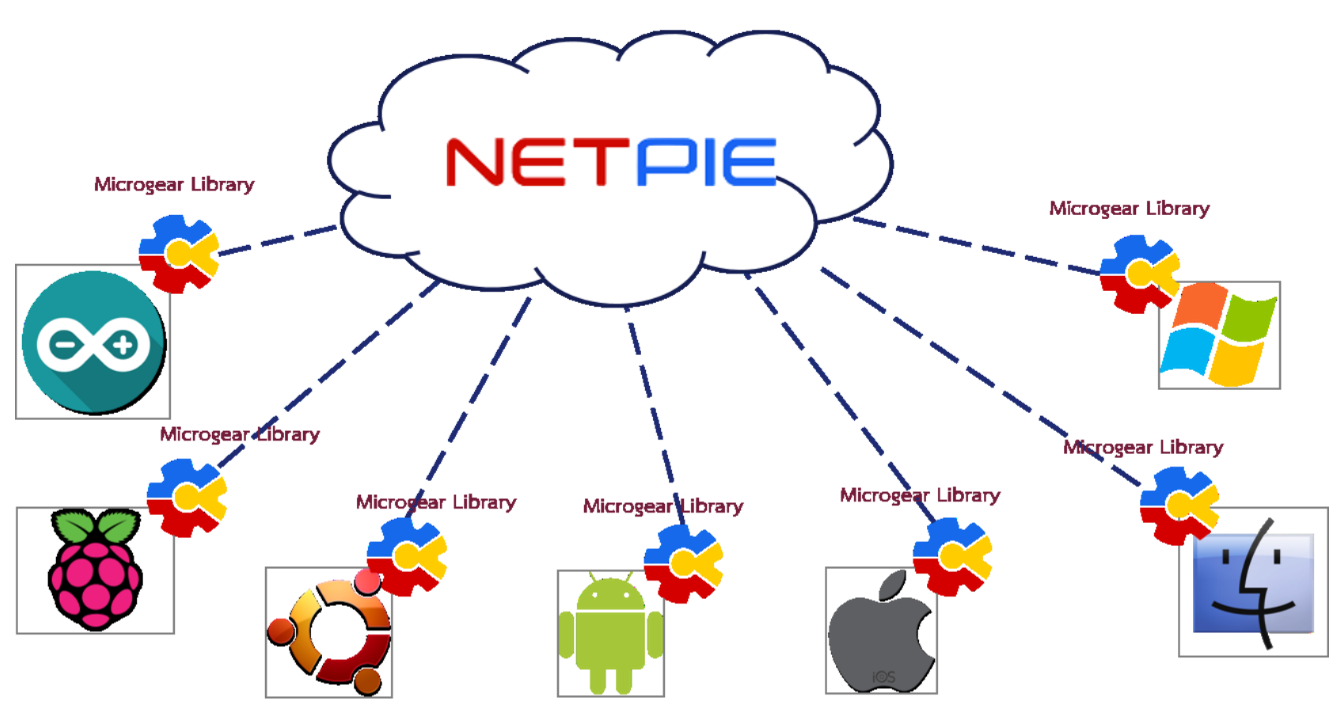
**เอกสารเนื้อหาฉบับเต็มอ่านได้จาก 1.** https://drive.google.com/drive/u/0/folders/0B9jvOTVzGjXJdFZfV3ZvcjhsUjg

**6.1 NETPIE[1]ปัจจุบันเวอร์ชั้น 2015 จะปิดรับสมัครและหยุดพัฒนาแล้ว ให้ใช้รุ่น 2020 แทน**

NETPIE เป็น IoT (Internet of Things) Cloud Platform พัฒนาโดยทีมวิจัยและเปิดให้บุคคลทั่วไปใช้งานผ่าน Web Portal ปัจจุบันมี 2 เวอร์ชั่นคือ NETPIE2015(https://2016.netpie.io/) และ NETPIE2020(https://netpie.io/) โดยเริ่มให้บริการตั้งแต่เดือนกันยายน 2558 เป็นต้นมา NETPIE เป็น Middleware ที่มีหัวใจหลัก (นอกเหนือจากส่วนอื่นๆ) เป็น Distributed MQTT Brokers ซึ่งเป็นเสมือนจุดนัดพบให้สิ่งต่างๆ(Things) มาติดต่อสื่อสารและท างานร่วมกันผ่านวิธีการส่งข้อความแบบ Publish/Subscribe

NETPIE มีโครงสร้างสถาปัตยกรรมเป็นคลาวด์อย่างแท้จริงในทุกองค์ประกอบ ทำให้สามารถขยายตัวได้อย่างอัตโนมัติ (Auto-scale) สามารถดูแลและซ่อมแซมตัวเองได้อัตโนมัติเมื่อส่วนหนึ่งส่วนใดในระบบมีปัญหา (Self-healing, Self-recovery)โดยไม่ต้องพึ่งผู้ดูแลระบบ การบริหารจัดการระบบเป็นแบบ Plug-and-Play ไม่ต้อง Configure หรือปรับแต่ง

ในฝั่งอุปกรณ์ NETPIE มี Client Library หรือที่เรียกว่า Microgear ซึ่งทำหน้าที่สร้างและดูแลช่องทางสื่อสารระหว่างอุปกรณ์กับ NETPIE รวมไปถึงรักษาความปลอดภัยในการส่งข้อมูล Microgear เป็น Open Source และสามารถดาวน์โหลดได้จากhttps://github.com/netpieio โดย ณ ปัจจุบันมี Microgear สำหรับ OS และ Embedded Board หลักๆ ที่เป็นที่นิยมในหมู่นักพัฒนาเกือบทุกชนิด โมเดลการสื่อสารของ NETPIE แสดงไว้ในรูปที่ 1



รูปที่ 1 วิธีการสื่อสารของสิ่งต่างๆ ผ่าน NETPIE

**6.2 ประโยชน์ของ NETPIE[1]**

6.2.1 ช่วยลดการใช้ทรัพยากรของการเชื่อมต่อ

NETPIE ช่วยให้อุปกรณ์สามารถสื่อสารกันได้โดยผู้ใช้ไม่ต้องกังวลว่า อุปกรณ์นั้นจะอยู่ที่ใด เพียงแค่นำ Microgear Library ไปติดตั้งในอุปกรณ์ NETPIE จะรับหน้าที่ดูแลเชื่อมต่อให้ทั้งหมด ไม่ว่าอุปกรณ์นั้นจะอยู่ในเครือข่ายชนิดใด ลักษณะใด หรือแม้กระทั่งเคลื่อนย้ายไปอยู่ที่ใด ผู้ใช้สามารถตัดปัญหาในการเข้าถึงอุปกรณ์จากระยะไกล (Remote Access) ด้วยวิธีการแบบเดิมๆ เช่น การใช้ Fixed Public IP Address หรือการตั้ง Port Forwarding ในเราท์เตอร์และการต้องไปลงทะเบียนกับผู้ให้บริการ Dynamic

DNS ซึ่งทั้งหมดล้วนมีความยุ่งยาก ลดความยืดหยุ่นของระบบ ไม่เพียงเท่านั้น NETPIE ยังช่วยให้การเริ่มต้นใช้งานเป็นไปโดยง่าย โดยออกแบบให้อุปกรณ์ถูกค้นพบและเข้าสู่บริการโดยอัตโนมัติ (Automatic Discovery, Plug-and-Play)

6.2.2 ช่วยลดภาระด้านความปลอดภัยของข้อมูล

NETPIE ถูกออกแบบให้มีระดับและสิทธิ์ในการเข้าถึงในระดับ Fine Grain กล่าวคือผู้ใช้สามารถออกแบบได้เองทั้งหมดว่า สิ่งใดมีสิทธิ์คุยกับสิ่งใด สิ่งใดมีสิทธิ์หรือไม่ - เพียงใดในการอ่านหรือเขียนข้อมูลและสิทธิ์เหล่านี้จะมีอายุการใช้งานนานเท่าใด หรือจะถูกเพิกถอนภายใต้เงื่อนไขใด เป็นต้น

6.2.3 ยืดหยุ่นต่อการขยายระบบ

NETPIE มีสถาปัตยกรรมเป็นคลาวด์เซิร์ฟเวอร์อย่างแท้จริงในทุกองค์ประกอบของระบบ ทำให้เกิด ความยืดหยุ่นและคล่องตัวสูงในการขยายตัว นอกจากนี้โมดูลต่างๆ ยังถูกออกแบบให้ทำงานแยกจากกันเพื่อให้เกิดสภาวะ Loose Coupling และสื่อสารกันด้วยวิธี Asynchronous Messaging ช่วยให้แพลตฟอร์มมีความน่าเชื่อถือได้สูงนำไปใช้ซ้ำและพัฒนาต่อได้ง่าย ดังนั้งผู้พัฒนาไม่จำเป็นต้องกังวลกับการขยายตัวเพื่อรับโหลดที่เพิ่มขึ้นในระบบอีกต่อไป

**6.3 MICROGEAR[1]**

Microgear คือซอฟต์แวร์ไลบรารี่ของ NETPIE ที่ติดตั้งอยู่บนอุปกรณ์ที่ต้องการเชื่อมต่อสื่อสารผ่านคลาวด์ของ NETPIE Microgear เปรียบเสมือนตัวกลางและผู้ช่วยในการสร้างและดูแลการเชื่อมต่อ ให้มีความเสถียร ปลอดภัย ให้การสื่อสารแลกเปลี่ยนข้อมูลระหว่างอุปกรณ์เป็นไปอย่างราบรื่น บทบาทหน้าที่ของ Microgear สามารถแบ่งออกเป็น 4 ด้านคือ

1. ด้านการสื่อสาร (Communication)

Microgear จะเป็นผู้ช่วยในการสร้างการเชื่อมต่อ(Connection) ไปยังคลาวด์ของ NETPIE และคอยตรวจสอบสถานะของการเชื่อมต่อ หากการเชื่อมต่อมีปัญหา Microgear สามารถช่วยเชื่อมต่อให้ใหม่เพื่อให้การสื่อสารเป็นไปได้อย่างราบรื่น นอกจากนี้ Microgear ยังช่วยอำนวยความสะดวก ในการสร้างช่องทางการสื่อสารแบบเข้ารหัสในกรณีที่ผู้ใช้ต้องการ ส่วนการแลกเปลี่ยนข้อมูลระหว่าง Microgear และคลาวด์ของ NETPIE จะใช้โพรโทคอล MQTT ในการสื่อสาร

2. ด้านการยืนยันตัวตน (Authentication)

ในขั้นตอนการสร้างการเชื่อมต่อ Microgear จะช่วยยืนยันตัวตนของอุปกรณ์กับคลาวด์ของ NETPIE โดยการพิสูจน์ตัวตน (Identity) ของอุปกรณ์จะใช้ข้อมูลประกอบกันสามส่วนคือ AppID, App Key และ Token

3. ด้านการขออนุญาตสิทธิ์ (Authorization)

การขออนุญาตสิทธิ์ในการสื่อสารจะเกิดขึ้นในขั้นตอนการสร้างการเชื่อมต่อ ควบคู่กับการยืนยันตัวตน คลาวด์ของ NETPIE จะเป็นผู้ออกใบอนุญาต (Token) ที่ระบุว่าอุปกรณ์ตัวนี้ สามารถสื่อสารได้กับอุปกรณ์ตัวใดบ้าง ในกรณีปกติอุปกรณ์ที่อยู่ภายใต้กลุ่ม AppID เดียวกันเท่านั้น จีงจะมีสิทธิ์สื่อสารกันได้ (ยกเว้นในกรณีการใช้ Freeboard Microgear ที่อนุญาตให้สื่อสารข้าม AppID ได้ ซึ่งจะอธิบายในบทที่ 5[1])

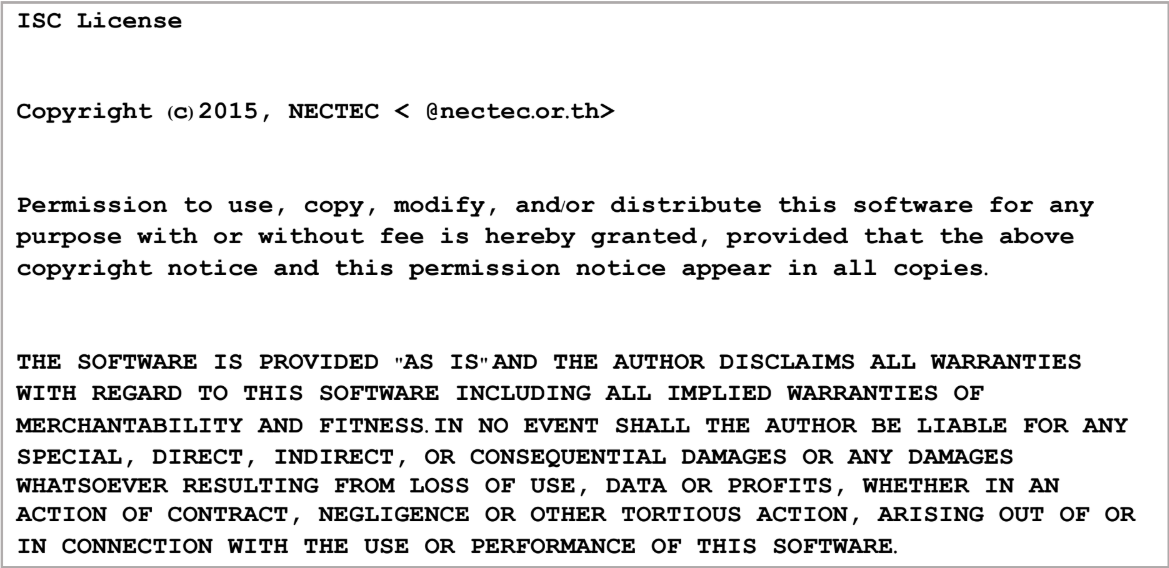
4. ด้านการประสานงาน (Coordination) Microgear

มีฟังก์ชั่นที่ช่วยให้อุปกรณ์ต่างๆ ภายในกลุ่ม AppID เดียวกันทราบสถานะของกันและกัน เช่น ทราบว่ามีอุปกรณ์ใดออนไลน์เข้ามาใหม่ในกลุ่ม หรือมีอุปกรณ์ใดออกไปจากกลุ่ม รวมถึงทราบการเปลี่ยนแปลงสถานะของอุปกรณ์ที่สนใจติดตาม จากข้อมูลดังกล่าวผู้ใช้สามารถกำหนดบทบาทหน้าที่ให้อุปกรณ์ในกลุ่มตามสถานะของอุปกรณ์อื่นๆ ในกลุ่ม เช่น หากเป็นอุปกรณ์ตัวแรกในกลุ่มให้ทำหน้าที่เป็นหัวหน้ากลุ่ม เป็นต้น

Microgear ถูกพัฒนาขึ้นเพื่อให้ทำงานได้กับอุปกรณ์ที่หลากหลาย ในส่วนของซอฟต์แวร์มี Microgear ให้เลือกใช้กับ Programming Language ได้แก่

* Node.js
* Python
* HTML5
* Java
* Android
* C#

สำหรับอุปกรณ์ฮาร์ดแวร์ประเภทไมโครคอนโทรลเลอร์ Microgear เปรียบเสมือน Firmware ซึ่งมี Microgear ที่รองรับ Arduino with Ethernet Shield (ใช้ได้กับ Arduino Mega) และ Microgear สำหรับ WiFi ไมโครคอนโทรลเลอร์ ESP8266 Microgear ที่พัฒนาขึ้นทั้งหมดถูกรวบรวมไว้ที่ https://github.com/netpieio โดยมีสัญญาอนุญาตให้ใช้สิทธิ์แบบเปิดประเภท ISC License ซึ่งอนุญาตให้ทำซ้ำดัดแปลง และ/หรือส่งต่อไลบรารี่นี้ได้ ทั้งในการใช้งานเชิงสาธารณประโยชน์และเชิงพาณิชย์ รายละเอียดของสญัญญาอนุญาตให้ใช้สิทธิ์แบบ ISC License มีดังนี

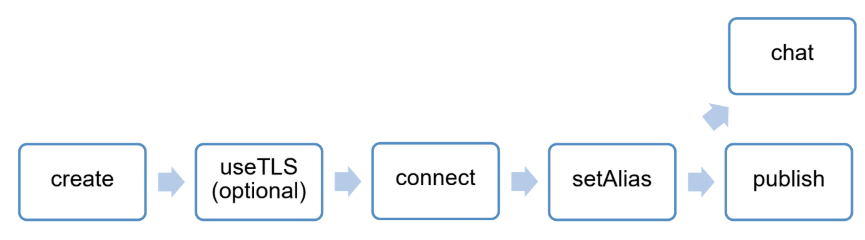


**6.3.1 ฟังก์ชั่นหลักของ Microgear**

Microgear แต่ละชนิดอาจมีชื่อและชนิดของฟังก์ชั่นแตกต่างกันตามลักษณะของการเขียนโปรแกรมในภาษานั้นๆ ในที่นี้ขอยกตัวอย่างฟังก์ชั่นที่มีเหมือนกันอยู่ในหลาย Microgear โดยขออ้างอิงชื่อฟังก์ชั่นจาก HTML5 Microgear สำหรับรายละเอียดฟังก์ชั่นของแต่ละชนิด Microgear สามารถดูได้จากภาคผนวก หรือเอกสาร Readme ใน <https://github.com/netpie.io>

* create สร้าง Microgear เพื่อเริ่มต้นใช้งาน
* connect เชื่อมต่อ Microgear เข้ากับคลาวด์ของ NETPIE
* setAlias กำหนดชื่อเล่นของอุปกรณ์เพื่อใช้ระบุตัวตนของอุปกรณ์ภายใน NETPIE
* chat ส่งข้อความแบบเจาะจงผู้รับ
* publish ส่งข้อความแบบไม่เจาะจงผู้รับไปยังหัวข้อสนทนาที่กำหนด
* subscribe ระบุความสนใจในหัวข้อสนทนา บอกรับข้อความที่เกิดขึ้นบนหัวข้อนั้นๆ
* unsubscribe ยกเลิกการบอกรับข้อความในหัวข้อสนทนาที่เคย subscribe ไว้
* resetToken ยกเลิกใบอนุญาต (Token) และลบใบอนุญาตออกจาก cache บนอุปกรณ์
* useTLS ระบุว่าต้องการสร้างการเชื่อมต่อแบบเข้ารหัสระหว่าง Microgear กับคลาวด์ของ NETPIE
* on ตอบสนองต่อเหตุการณ์ที่สนใจผ่านการเรียก Callback Function

ลำดับในการเรียกฟังก์ชั่นพื้นฐานเพื่อเริ่มส่งข้อมูลเป็นไปตามแผนภาพในรูปที่ 2



รูปที่ 2 ลำดับการเรียกฟังก์ชั่นพื้นฐานของ Microgear

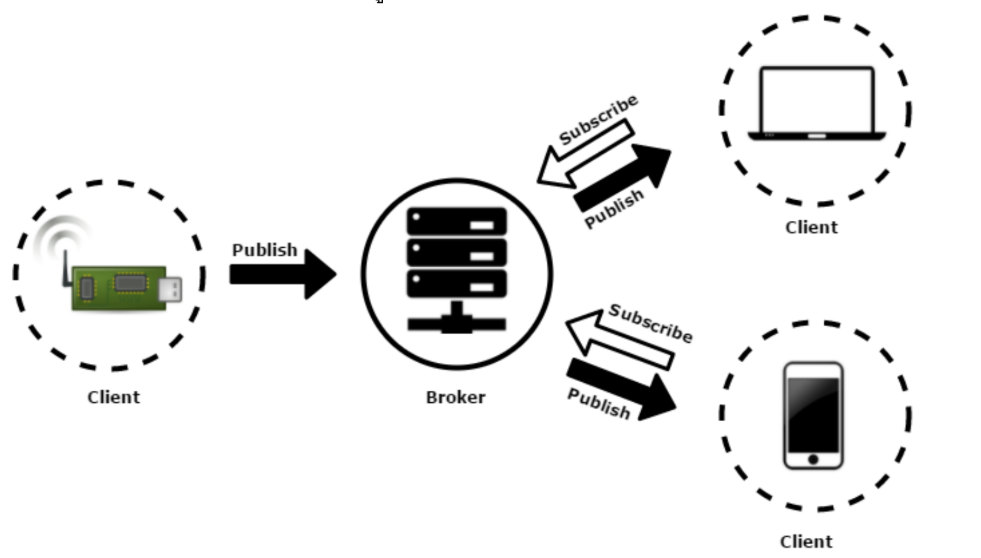
**6.3.2 Events ของ Microgear**

การทำงานของ Microgear เป็นแบบ Event-driven จึงต้องตอบสนองต่อเหตุการณ์ต่างๆ ด้วยการเขียน Callback Function ซึ่งชนิดของเหตุการณ์ที่สามารถเกิดขึ้น มีดังนี้

* connected เกิดขึ้นเมื่อ Microgear เชื่อมต่อกับ NETPIE สำเร็จ
* closed เกิดขึ้นเมื่อ Microgear ปิดการเชื่อมต่อกับ NETPIE
* error เกิดขึ้นเมื่อมีความผิดพลาดเกิดขึ้นกับ Microgear
* message เกิดขึ้นเมื่อมีข้อความเข้ามาที่อุปกรณ์
* present เกิดขึ้นเมื่อมีอุปกรณ์ใน AppID เดียวกันเชื่อมต่อเข้ามาบน NETPIE
* absent เกิดขึ้นเมื่อมีอุปกรณ์ใน AppID เดียวกันหายไปจากการเชื่อมต่อกับ NETPIE

**6.4 MQTT (****MQ TELEMETRY TRANSPORT)**

MQTT เป็นโพรโทคอลสื่อสารชั้นแอปพลิเคชั่นที่รันบน TCP/IP ถูกพัฒนาขึ้นในปี 1999 โดย IBM และ Eurotech สำหรับการมอนิเตอร์สถานะท่อส่งน้ำมันส่วนที่วางผ่านเขตทะเลทราย ด้วยการออกแบบให้เป็นการรับส่งข้อความที่มีน้ำหนักเบามากและเป็นโพรโทคอลเปิด ทำให้ในปัจจุบัน MQTT ถูกนำมาใช้แพร่หลายในการสื่อสารแบบ M2M หรือ IoT เพราะเหมาะสมกับอุปกรณ์ปลายทางที่มีขนาดเล็ก/พลังงานจำกัด หรือในการสื่อสารระยะไกลที่ต้องการการใช้งานแบนด์วิดธ์อย่างมีประสิทธิภาพ โมเดลการสื่อสารของโพรโทคอลแสดงดังรูปที่ 3 ประกอบด้วย 2 ส่วนคือไคลเอนต์ และโบรกเกอร์



รูปที่ 3 โมเดลการสื่อสารของโพรโทคอล MQTT

**6.4.1 โบรกเกอร์**

เป็นจุดศูนย์กลางในการรับส่งข้อความระหว่างไคลเอนต์ วิธีการกำหนดเส้นทาง(Routing) กระทำผ่านหัวข้อ (Topic) โดยไคลเอนต์ Subscribe ในหัวข้อที่ตนต้องการ จากนั้นโบรกเกอร์จะส่งข้อความทั้งหมดที่ถูก Publish ในหัวข้อนั้นๆ ไปให้ ดังนั้นไคลเอนต์จึงสื่อสารกันได้โดยไม่จำเป็นต้องรู้จักกัน ช่วยลดความเกี่ยวพันระหว่างผู้สร้างข้อมูลและผู้ใช้ข้อมูล ส่งผลให้การขยายตัวของเครือข่ายทำได้ง่ายนอกจากนี้หน้าที่ที่สำคัญอีกประการของโบรกเกอร์คือการรักษาความปลอดภัยของไคลเอนต์(Authorization, Authentication) ซึ่งในส่วนนี้สามารถขยายเพิ่มเติม หรือนำไปเชื่อมกับกลไกความปลอดภัยของระบบหลังบ้านที่มีอยู่แล้วได้ ช่วยให้นำโบรกเกอร์เข้าไปใช้งานเป็นส่วนหนึ่งของระบบอื่นๆ ได้ ส่วน Authorization ของ NETPIE ซึ่งจะได้กล่าวถึงในหัวข้อต่อไป(นักศึกษาต้องไปอ่านด้วยตนเองในเอกสาร[1]) ก็ถือเป็นตัวอย่างหนึ่งของการขยายเพิ่มเติมการรักษาความปลอดภัยของโบรกเกอร์ใน MQTT ปัจจุบันมีโบรกเกอร์ MQTT ที่เปิดให้ดาวน์โหลดไปใช้หรือดัดแปลงอยู่หลายราย ได้แก่ Mosquitto, RabbitMQ, Erlang, VerneMQ ฯลฯ

**6.4.2 ไคลเอนต์**

จะเป็นได้ทั้ง Publisher หรือ Subscriber หรือ Publisher/Subscriber พร้อมๆ กัน และจะเป็นอุปกรณ์ใดๆ ก็ได้ที่สามารถรัน MQTT Client Library บน TCP/IP Stack การที่ MQTT ใช้โมเดล Publish/Subscribe ตรรกะส่วนใหญ่จึงไปตกอยู่ในฝั่งโบรกเกอร์ ทำให้ Library มีขนาดเล็ก ติดตั้งได้ง่ายใช้งานได้กับอุปกรณ์ที่มีทรัพยากรจำกัด ไคลเอนต์จำเป็นต้องเปิดการเชื่อมต่อ TCP ไว้ตลอดเพื่อที่โบรกเกอร์จะสามารถผลักข้อความไปให้ได้ หากการเชื่อมต่อถูกตัดขาด โบรกเกอร์จะเก็บข้อความทั้งหมดที่เข้ามาไว้จนกว่าไคลเอนต์จะกลับมาออนไลน์อีกครั้ง

เมื่อเปรียบเทียบ MQTT กับ HTTP (REST) ที่มีสถาปัตยกรรมแบบ Request/Response จะพบว่า MQTT มีความได้เปรียบที่

โบรกเกอร์สามารถผลัก (Push) ข้อความไปยังไคลเอนต์ได้ตามเหตุการณ์(Event-driven) ในขณะที่เมื่อใช้ HTTP ฝั่งไคลเอนต์ต้องคอยสุ่มถาม (Poll) ข้อมูลเป็นระยะๆ และต้องตั้งค่าคาบเวลาการสุ่มถามไว้ก่อนล่วงหน้า โดยแต่ละครั้งต้องมีการสร้างการเชื่อมต่อขึ้นใหม่และอาจจะไม่มีข้อมูลใหม่ใดๆ ให้อัพเดท ดังนั้นหากต้องการให้ระบบทำงานแบบ Real Time หรือใกล้เคียง ย่อมหมายถึงต้องตั้งคาบเวลาการสุ่มถามให้สั้น และทำให้เกิดความสิ้นเปลืองของการใช้ช่องสัญญาณที่ไม่จำเป็นที่ตามมานี่จึงเป็นอีกเหตุผลสำคัญที่ทำให้ MQTT ได้รับความนิยมเหนือ REST สำหรับการใช้งานแบบ M2M นอกเหนือจากการมีน้ำหนักเบา

**6.4.3 MQTT Topics**

MQTT Topic เป็น UTF-8 String ในลักษณะเดียวกับ File Path คือสามารถจัดเป็นลำดับชั้นได้ด้วยการคั่นด้วย “/” ตัวอย่างเช่น myhome/floor-one/room-c/temperature ไคลเอนต์สามารถเลือก Publish หรือ Subscribe เฉพาะ Topic หรือ Subscribe หลาย Topic พร้อมๆ กันโดยใช้ Single-Level Wildcard (+) เช่น myhome/floor-one/+/temperature หมายถึงการขอเขียนหรือรับข้อความ temperature จากทุกๆ ห้องของ myhome/floor-one หรือ Multi-Level Wildcard (#) เช่น myhome/floor-one/# หมายถึง

การขอเขียนหรือรับข้อความทั้งหมดที่มี Topic ขึ้นต้นด้วย myhome/floor-one เป็นต้น

เราสามารถกำหนด Topic อย่างไรก็ได้ โดยมีข้อยกเว้นการขึ้นต้น Topic ด้วยเครื่องหมาย “$” ซึ่งจะจำกัดไว้สำหรับการเก็บสถิติภายในของตัวโบรกเกอร์เท่านั้น ดังนั้นไคลเอนต์จะไม่สามารถ Publish หรือ Subscribe ไปยัง Topic เหล่านี้ได้ โดยทั่วไป Topic เหล่านี้จะขึ้นต้นด้วย $SYS

MQTT vs Message Queues เรามักพบการสับสนระหว่าง MQTT กับ Message Queues โดยคนจำนวนไม่น้อยเชื่อว่า MQ

ใน MQTT มาจากคำว่า Message Queueในความเป็นจริงแล้ว MQ ในที่นี้มาจากชื่อรุ่นผลิตภัณฑ์ที่รองรับโพรโทคอล MQTT ของ IBM ที่เรียกว่า MQ Series และ MQTT ไม่ได้ทำงานในลักษณะ Message Queue กล่าวคือ ข้อความใน MQTTจะถูกส่งให้กับไคลเอนต์ทั้งหมดที่ Subscribe ใน Topic ในขณะที่ข้อความใน Message Queue สามารถถูกดึงออกไปใช้งานโดยผู้ใช้เพียงรายเดียวเท่านั้น

\*\*\*รายละเอียดเพิ่มเติ่มเกี่ยวกับ MQTT นักศึกษาต้องอ่านจากเอกสาร[1] และอาจปรากฏในข้อสอบ\*\*\*

**6.5 AUTHORIZATION และ AUTHENTICATION ใน NETPIE**

เนื่องจาก NETPIE มีศูนย์กลางการสื่อสารเป็น Distributed MQTT Broker ความปลอดภัยจึงขึ้นตรงกับการเข้าถึงโบรกเกอร์ของอุปกรณ์ ดังที่ได้กล่าวไปแล้วในหัวข้อที่แล้ว(ต้องอ่านเพิ่มจาก[1])ว่ามาตรฐาน MQTT ระบุให้ใช้เพียง Username และ Password ในการพิสูจน์ตัวตน (Authentication) ของไคลเอนท์ โดยสามารถกำหนดสิทธิ์การเข้าถึงโบรกเกอร์ของไคลเอนต์แต่ละรายด้วยการผูกเป็นรายการ (List) เข้ากับ Username บนโบรกเกอร์ ในขณะที่ NETPIE นั้น มีข้อมูลประจำตัว (Credentials) ของอุปกรณ์ที่จะมาขอเชื่อมต่อถึงสองชั้นคือ Key และ Token ซึ่งจะช่วยเพิ่มความยืดหยุ่นในการบริหารจัดการการเข้าถึงโบรกเกอร์ของอุปกรณ์ให้มี

ความละเอียด ยืดหยุ่นและมีประสิทธิภาพมากขึ้น ดังนั้นอุปกรณ์จะได้มาซึ่งข้อมูลประจำตัวจากการขออนุญาตสิทธิ์ (Authorization) ซึ่งประกอบด้วยผู้เกี่ยวข้อง 3 ฝ่ ายคือ

1. อุปกรณ์ (Device) ที่จะมาเชื่อมต่อสื่อสารกันให้เกิดแอปพลิเคชั่น ในที่นี้รวมถึงทั้งอุปกรณ์กายภาพและอุปกรณ์เสมือน (Object)

2. ผู้ใช้ (User) ซึ่งเป็นเจ้าของแอปพลิเคชั่น และเป็นผู้กำหนดว่าอุปกรณ์ใดมีสิทธิ์หรือไม่ มากน้อยเพียงใดในแอปพลิเคชั่นของตน

3. NETPIE ทำหน้าที่เป็นตัวกลางในการขออนุญาตสิทธิ์ระหว่างผู้ใช้และอุปกรณ์

**6.6.1 กระบวนการทั้งหมดมีขั้นตอนคือ**

1. ผู้ใช้ลงทะเบียนกับ NETPIE Web Portal และสร้างแอปพลิเคชั่น

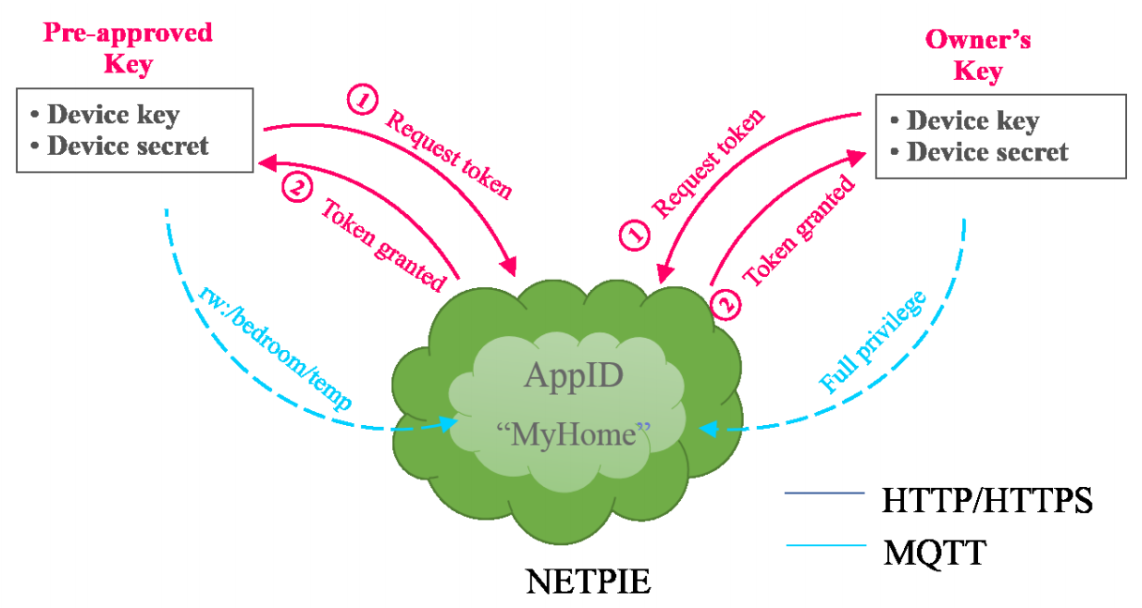
2. ผู้ใช้เพิ่มอุปกรณ์ภายใต้แอปพลิเคชั่นนั้นๆ และได้รับ Key และ Key Secret เพื่อนำมาใส่ลงในโค้ดของ Microgear ในอุปกรณ์

3. เมื่ออุปกรณ์ถูกเปิดใช้งาน Microgear จะเชื่อมต่อเข้ามายัง NETPIE โดยอัตโนมัติ และพิสูจน์ตัวตนด้วย Key และ Key Secret ที่ได้รับในการขอ Token เพื่อเชื่อมต่อและสื่อสารผ่าน NETPIE

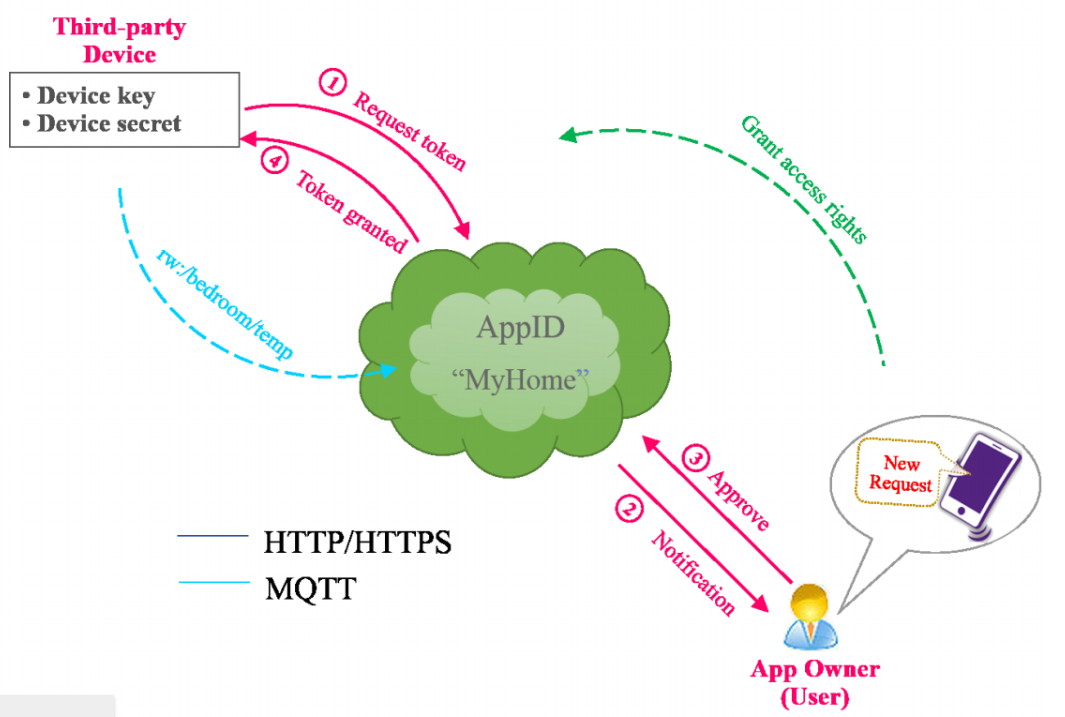
4. ในขั้นตอนนี้ NETPIE มีทางเลือกสองทางเพื่อความสะดวกของผู้ใช้

ทางเลือกที่ 1: Pre-approved Key กล่าวคือผู้ใช้สามารถกำหนดไว้ก่อนล่วงหน้าว่า Key ใดบ้างจะได้รับการอนุญาตสิทธิ์แบบอัตโนมัติ ดังนั้นทันทีที่อุปกรณ์ต่อเข้ามา จะได้รับ Token ทันที นอกจากนี้ผู้ใช้ยังสามารถกำหนดระดับของสิทธิ์ไว้ก่อนได้เช่นกัน เช่น ให้สิทธิ์อุปกรณ์ของผู้ใช้เองแบบเต็ม ให้สิทธิ์อุปกรณ์ของคนอื่นบางส่วน เป็นต้น

ทางเลือกที่ 2: Third-Party Key ซึ่งเหมาะกับกรณีที่ผู้ใช้ให้สิทธิ์อุปกรณ์ของบุคคลอื่นเข้ามาร่วมใช้แอปพลิเคชั่น เมื่ออุปกรณ์ร้องขอ Token เข้ามา NETPIE AUTH Server จะติดต่อไปยังผู้ใช้ ให้เข้ามาอนุญาตสิทธิ์ โดยสามารถกำหนดระดับของสิทธิ์และ NETPIE จะทำการออก Token ให้กับอุปกรณ์และบันทึกระดับของสิทธิ์ที่ได้รับอนุญาตเอาไว้



รูปที่ 4 NETPIE Authorization ในกรณี Pre-approved Key



รูปที่ 5 NETPIE Authorization ในกรณี Third-party Key

เมื่ออุปกรณ์ได้ชุดข้อมูลส่วนตัวครบแล้ว จะนำทั้งหมด (Key, Key Secret, Token, Token Secret)ไปสร้าง Client ID, Username และ Password ในการพิสูจน์ตัวตนเข้าใช้ NETPIE ตามโพรโทคอล MQTT โดยวิธีการสร้างจะทำให้ Username กับ Password มีการเปลี่ยนแปลงทุกครั้งในแต่ละเซสชั่น และมีการแฮชและเข้ารหัส ช่วยป้องกันการดักฟังและการทำซ้ำ

NETPIE รองรับการเชื่อมต่อ MQTT แบบ Persistent Connection เท่านั้น ดังนั้นจะมีการตั้งค่า Client ID ไว้เสมอ หากมีการเพิกถอนสิทธิ์ของอุปกรณ์ Client ID จะถูกลบ อุปกรณ์จะต้องต่อไปยัง NETPIE เพื่อขอ Token ใหม่ โดยไม่ต้องไปเริ่มต้นกระบวนการขอ Key ใหม่ตั้งแต่ต้นโมเดลของ NETPIE นี้เป็นการนำแนวคิดที่ใช้ในสังคมมนุษย์มาปรับใช้ในสังคมของสิ่งของ เช่นเมื่อคนทำผิดกฎจราจร ถูกยึดใบขับขี่ทำให้ไม่มีสิทธิ์ในการขับขี่ยานพาหนะ แต่สามารถไปขอทำใบขับขี่ใหม่ได้ภายหลัง เป็นต้น

**6.6.2 ชนิดของ Key**

NETPIE รองรับอุปกรณ์ที่หลากหลาย นอกจากอุปกรณ์กายภาพเช่น บอร์ดไมโครคอนโทรลเลอร์ต่างๆ แล้ว ยังรองรับอุปกรณ์เสมือนเช่น แอปพลิเคชั่นต่างๆ ซึ่งรวมไปถึงแอปพลิเคชั่นที่เขียนด้วย HTML5 ที่สามารถรันได้บนเบราว์เซอร์ ในกรณีเช่นนี้การเชื่อมต่อไม่สามารถเป็นแบบ Persistent ได้ เนื่องจากต้องมีการปิดเปิดเบราว์เซอร์ตลอดเวลา ดังนั้นเพื่อให้การจัดการ Key และ Token มีประสิทธิภาพ หลีกเลี่ยงการสร้าง Token ทิ้งไว้โดยไม่ได้ใช้งาน NETPIE จึงออกแบบให้มี Key อยู่ 2 ชนิด คือ

1. Device Key

เมื่ออุปกรณ์ได้รับการติดตั้ง Device Key และร้องขอ Token มายัง NETPIE หากได้รับการอนุญาตสิทธิ์ อุปกรณ์จะได้รับ Token ที่ใช้ได้ตลอดไป ไม่มีการหมดอายุ ดังนั้นตราบใดที่ไม่มีการเพิกถอนสิทธิ์อุปกรณ์สามารถใช้ Token เดิมไปได้ตลอดโดยไม่ต้องเข้าสู่กระบวนการ Authorization ใหม่ Device Key จึงเหมาะกับอุปกรณ์กายภาพที่สามารถรักษาช่องการเชื่อมต่อกับ NETPIE ไว้ได้โดยตลอด

2. Session Key

อุปกรณ์ที่ได้รับการติดตั้ง Session Key จะได้รับ Token ที่ใช้ได้เพียงครั้งเดียว (One-time Token) หากอุปกรณ์ยกเลิกการเชื่อมต่อไป และต่อกลับมาใหม่ จะต้องมีการนำ Session Key ไปขอ Token ใหม่ทุกครั้ง Session Key จึงเหมาะกับการใช้งานแอปพลิเคชั่นที่รันบนเบราว์เซอร์ที่ต้องมีการปิดเปิดอยู่เรื่อยๆเพราะหากใช้ Device Key ทุกครั้งที่เบราว์เซอร์ถูกเปิดขึ้นจะมีการร้องขอ Token ใหม่ และเมื่อเบราว์เซอร์ถูกปิดไป Token ที่ได้จะค้างอยู่เช่นนั้นโดยไม่ได้ใช้งานอีก ส่งผลให้ภายใต้ Key เดียวกัน มี Token ทั้งที่ใช้งานและไม่ได้ใช้งานแล้วผูกอยู่เป็นจำนวนมาก สร้างความสับสนซับซ้อนที่ไม่จำเป็นให้กับผู้ใช้ จึงขอเน้นย้ำเป็นอย่างยิ่งว่าไม่ควรใช้ Device Key กับแอปพลิเคชั่นที่รันบนเบราว์เซอร์(HTML5) โดยเด็ดขาด

**6.6.3 ขอบเขตของสิทธิ์(Scope) ในการเข้าถึง NETPIE MQTT Brokers**

เมื่อผู้ใช้อนุญาตสิทธิ์ให้กับอุปกรณ์ จะสามารถระบุขอบเขตของสิทธิ์หรือที่เราเรียกว่า Scope ได้ว่ามากน้อยเพียงใด โดย Scope นี้จะผูกอยู่กับ Token ที่อุปกรณ์จะได้รับ และถูกเก็บไว้ที่ NETPIE รูปแบบของ Scope Identifier เกี่ยวพันโดยตรงกับโครงสร้างการสื่อสารแบบ Publish/Subscribe ไปยัง Topic บนโบรกเกอร์ MQTT มีได้ 3 ลักษณะคือ

1. rw:/topic เช่น rw:/home/bedroom/temperature หมายถึง มีสิทธิ์อ่าน (Subscribe) และเขียน (Publish) ใน home/bedroom/temperature

2. r:/topic เช่น r:/home/# หมายถึง มีสิทธิ์อ่าน (Subscribe) ทุก Topic ที่ขึ้นต้นด้วย home

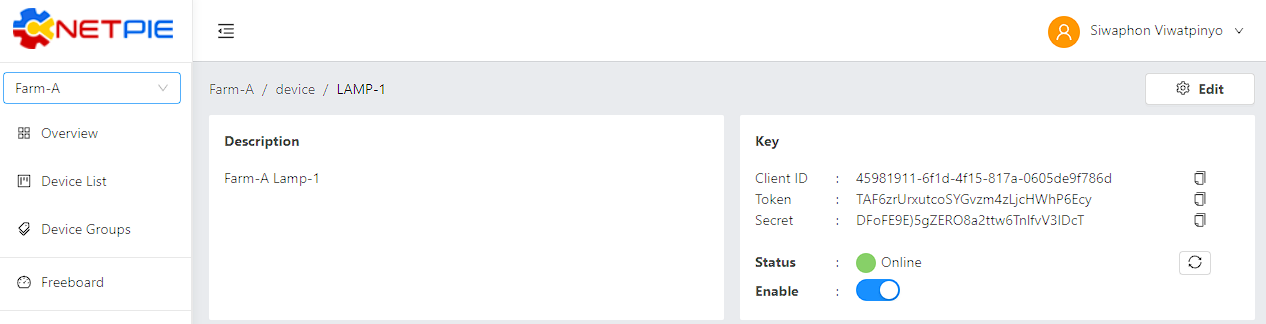
3. w:/topic เช่น w:/home/+/temperature หมายถึง มีสิทธิ์เขียน (Publish) ไปยัง Topic ชื่อ temperature ของทุกๆ ห้องใน homeการกำหนด Scope ที่เกี่ยวข้องกับหลาย Topic นอกจากใช้ Wildcard ดังที่ได้แสดงไปแล้ว สามารถทำได้โดยการเรียง Scope Identifier เข้าด้วยกันแล้วคั่นด้วยเครื่องหมาย Bar "|" ตัวอย่างเช่น rw:/home/bedroom/temperature|r:/home/#|w:/home/+/temperature

**6. ขั้นตอนการสมัครสมาชิก NETPIE2020** [**https://siwaphon.medium.com/netpie-2020-thai-iot-platform-92f30f5e774d**](https://siwaphon.medium.com/netpie-2020-thai-iot-platform-92f30f5e774d)

**7.** ขั้นตอนการทดสอบการใช้ติดต่อกับ NETPIE ด้วย MQTTBox [**https://siwaphon.medium.com/%E0%B8%97%E0%B8%94%E0%B8%AA%E0%B8%AD%E0%B8%9A%E0%B8%81%E0%B8%B2%E0%B8%A3%E0%B9%80%E0%B8%8A%E0%B8%B7%E0%B9%88%E0%B8%AD%E0%B8%A1%E0%B8%95%E0%B9%88%E0%B8%AD%E0%B8%88%E0%B8%B2%E0%B8%81-mqtt-box-%E0%B9%84%E0%B8%9B%E0%B8%A2%E0%B8%B1%E0%B8%87-netpie2020-df09624fbb0c**](https://siwaphon.medium.com/%E0%B8%97%E0%B8%94%E0%B8%AA%E0%B8%AD%E0%B8%9A%E0%B8%81%E0%B8%B2%E0%B8%A3%E0%B9%80%E0%B8%8A%E0%B8%B7%E0%B9%88%E0%B8%AD%E0%B8%A1%E0%B8%95%E0%B9%88%E0%B8%AD%E0%B8%88%E0%B8%B2%E0%B8%81-mqtt-box-%E0%B9%84%E0%B8%9B%E0%B8%A2%E0%B8%B1%E0%B8%87-netpie2020-df09624fbb0c)

8. ให้นักศึกษาเปิด Command Prompt ขึ้นมาจากนั้นพิมพ์คำสั่ง pip install paho-mqtt

9. สร้างไฟล์ชื่อ LAB6\_1\_เลขท้ายรหัสนักศึกษา 3 ตัว.py พิมพ์โค๊ดต่อไปนี้ โดย CLIENT\_ID และ DEVICE\_TOKEN ให้ไปนำมาจาก NETPIE ในส่วนของ Device เช่น



import sys

import time

import paho.mqtt.client as mqtt

import json

import random

NETPIE\_HOST = "mqtt.netpie.io"

CLIENT\_ID = "เอามาจาก Client ID ใน NETPIE" # Client ID ของ Device ที่สร้างขึ้นใน NETPIE

DEVICE\_TOKEN = ""เอามาจาก Token ใน NETPIE" "# Token ของ Device ที่สร้างขึ้นใน NETPIE

sensor\_data = {'temperature': 0, 'humidity': 0}

def on\_connect(client, userdata, flags, rc):

    print("Result from connect: {}:".format(mqtt.connack\_string(rc)))

    client.subscribe("@shadow/data/updated")

client = mqtt.Client(protocol=mqtt.MQTTv311,client\_id=CLIENT\_ID, clean\_session=True)

client.username\_pw\_set(DEVICE\_TOKEN)

client.on\_connect = on\_connect

client.connect(NETPIE\_HOST, 1883)

client.loop\_start()

try:

    while True:

        temperature = random.randint(0, 50)

        humidity = random.randint(0, 100)

        if humidity is not None and temperature is not None:

            humidity = round(humidity)

            temperature = round(temperature)

            print(

                "Temp={0: 0.1f}\*C Humidity={1: 0.1f} %".format(temperature, humidity))

            sensor\_data["temperature"] = temperature

            sensor\_data["humidity"] = humidity

            print(json.dumps({"data": sensor\_data}))

            client.publish("@shadow/data/update",

                           json.dumps({"data": sensor\_data}), 1)

            time.sleep(10)

        else:

            print("Failed to get reading. Try again!")

except KeyboardInterrupt:

    pass

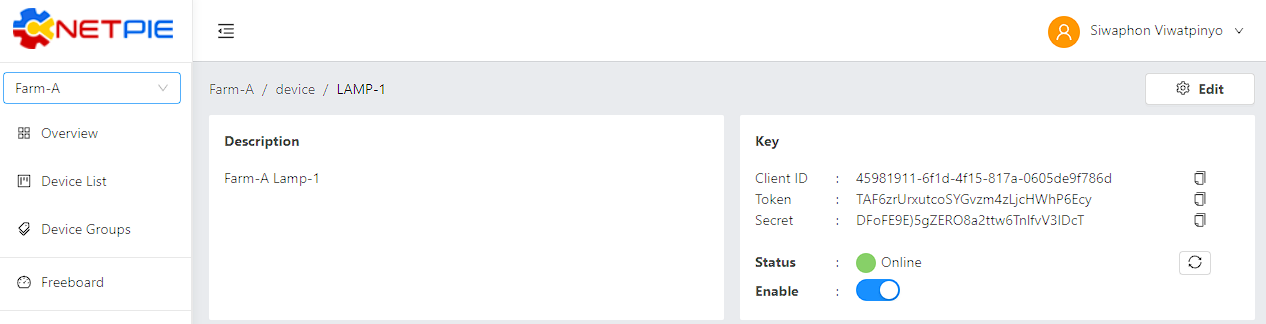
    client.loop\_start()

    client.disconnect()

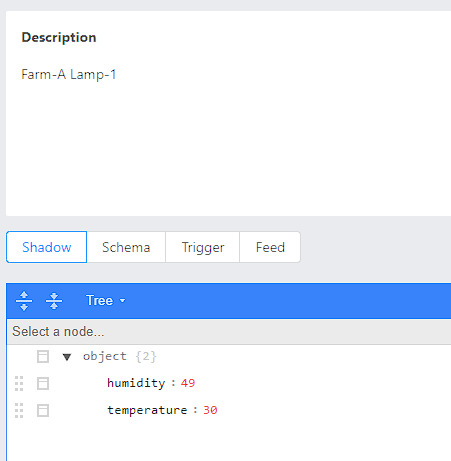
10. ทดลองรันโปรแกรม หากโปรแกรมทำงานได้ปกติให้บันทึกหน้าจอการทำงาน

**บันทึกหน้าจอการทำงาน ......**

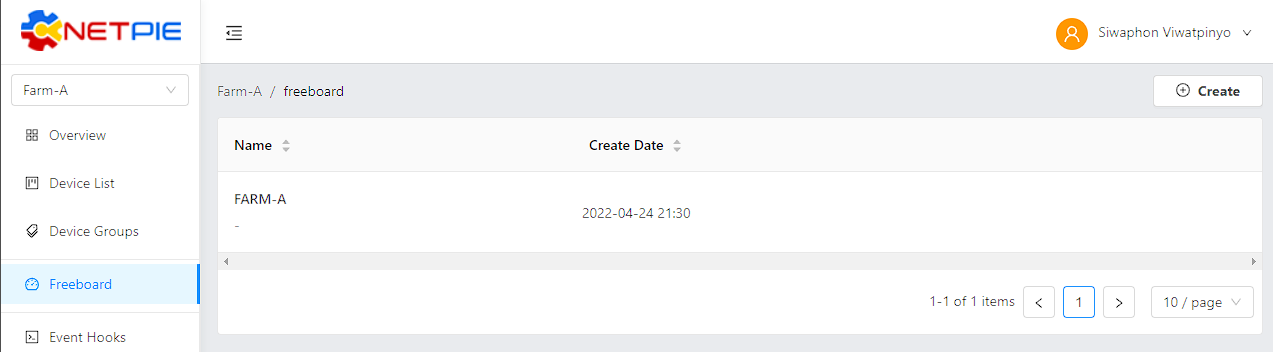
11. เข้าไปตรวจสอบที่ NETPIE ที่ Status จะต้องเป็นสถานะ Online ดังนี้



12. เลือก Shadow จะต้องพบว่ามีการรับค่ามาจากโปรแกรมของเรา ดังนี้



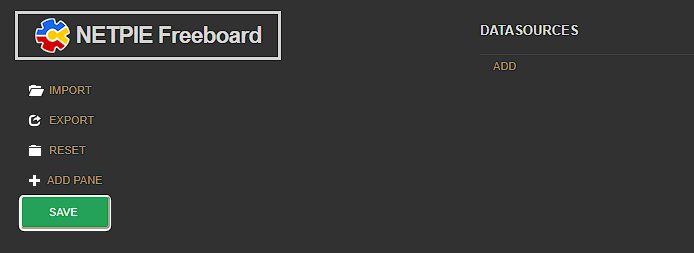
13. ขั้นต่อไปจะเป็นการสร้างส่วนของการแสดงผลด้วย Freeboard โดบเลือกที่ Freeboard > Create กำหนดชื่อเป็นภาษาอังกฤษ ที่ช่อง Name ส่วน Description จะระบุหรือไม่ก็ได้ จากนั้นกดปุ่ม Create



1

2

14. คลิ๊กที่ชื่อ Freeboard จะเข้าสู่ส่วนของ Freeboard โดยเราต้องทำการเพิ่ม Data Source ก่อนเสมอโดยคลิ๊กเลือกที่ ADD



16. กำหนดค่าต่างๆ ดังนี้ แล้วกด SAVE

NAME = ตั้งชื่อ Data Source ซึ่งควรวางแผนให้ดีเนื่องจากในการใช้งานจริงแต่ละแปลง แต่ละอุปกรณ์อาจต้องแยกกันคนละ Data Source

DEVICE ID = ค่า Client ID จาก NETPIE ก่อนหน้านี้

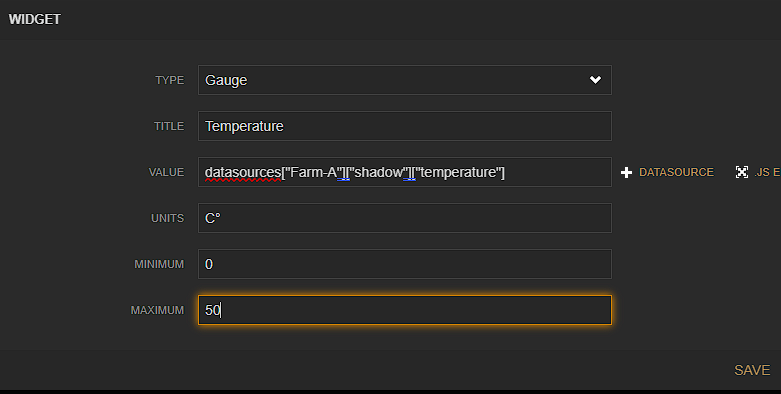
DEVICE TOKEN1 = ค่า Token จาก NETPIE ก่อนหน้านี้

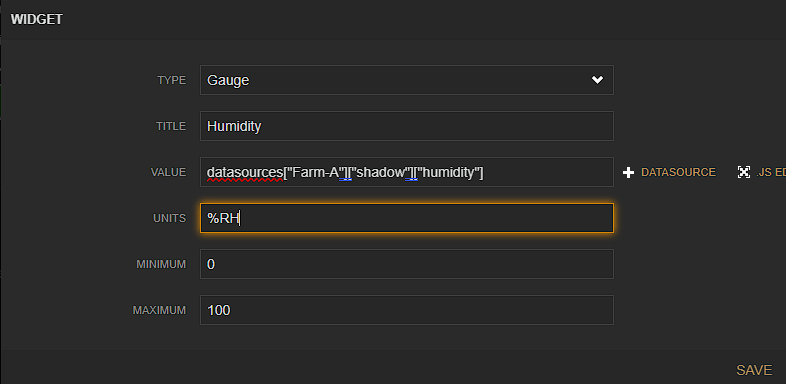
SUBSCRIBED TOPICS = @shadow/data/updated

หมายเหตุ : ศึกษารายละเอียดของ shadow จากเอกสารของ NETPIE

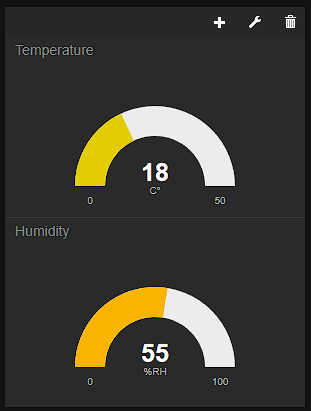
16. เมื่อเพิ่ม Data Source เสร็จแล้วให้ทดลองรันโปรแกรมหรือหากรันไว้อยู่แล้วให้สังเกตุที่หลังชื่อ Data Source จะพบว่าค่า Last Update จะมีการเปลี่ยนแปลง แสดงว่า Data Source สามารถรับค่าจากโปรแกรมของเราได้ จากนั้นกด SAVE (ควรทำเป็นระยะ)

17. เลือก ADD PANE เราจะได้พื้นที่เพลนมาจากนั้นคลิ๊กที่เครื่องหมาย + เลือก type เป็น GAUGE ซึ่งเราจะเพิ่ม Widget ของ GAUGE เข้าไป 2 Widget โดยกำหนดชื่อดังนี้ โดยในส่วนของค่า VALUE ให้เลือกจากเมนู +DATASOURCE ด้านหลังโดยเลือกดังนี้ +DATASOURCE > Farm-A > shadow > เลือก temperature หรือ humidity ให้ตรงตาม TITLE ที่กำหนด





18. หากรันโปรแกรมอยู่แล้วจะเห็นการเปลี่ยนแปลงของ GAUGE ดังนี้



**บันทึกผลหน้า Freeboard ทั้งหน้าให้เห็น DATASOURCE และ Widget ทั้ง 2 ให้ครบ**

**........................................................................................................................................**

19. ค้นคว้าเพิ่มเติมเพื่อทดสอบ Widget Type อื่นๆ

**บันทึกผล โค๊ด, การกำหนดค่า Widget และผลการทำงาน.............................................**

**\*\*\*\*\*\*\* อย่าลืมกดปุ่ม SAVE ทุกครั้งเมื่อมีการเปลี่ยนแปลง \*\*\*\*\*\***

ตอนนี้นักศึกษาได้ทดลองการเชื่อมต่อและส่งค่าไปยัง NETPIE2020 แล้วโดยหัวข้อที่ผ่านมาได้อธิบายถึง MQTT Broker ที่ใช้ MQTT Protocol เพื่อให้อุปกรณ์สามารถติดต่อสื่อสารกันได้โดยจะถูกบริหารจัดการโดย MQTT Broker ซึ่งในตอนนี้คือ NETPIE ที่ทำตัวเป็น IoT Cloud Platform โดย NETPIE มีคุณสมบัติสำคัญคือ

1. Monitoring คือความสามารถในการแสดงค่าข้อมูลของอุปกรณ์หรือเซนเซอร์แบบ Real-Time
2. Controlling คือ การควบคุมการทำงานอุปกรณ์ต่างๆ ผ่าน NETPIE Cloud Platform
3. Data Storage คือ การเก็บข้อมูลที่ได้จากเซนเซอร์หรืออุปกรณ์
4. Notification คือ การแจ้งเตือนความผิดปกติของเซนเซอร์หรืออุปกรณ์จากที่ได้กำหนดไว้

NETPIE

2020

Arduino

Raspberry Pi

Computer

ESP32

Kid bright

Micro bit

Smart Device เช่น

- Smart Meter

- Smart Sensor

MQTT Protocol

Subscribe

Subscribe

Publish / Subscribe

Publish

Publish

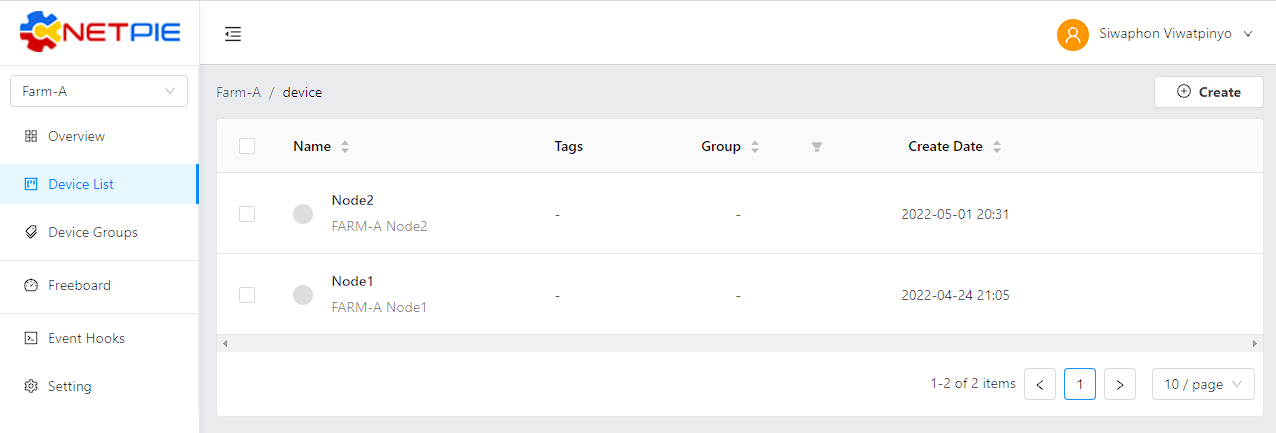
Publish

Publish / Subscribe

การรับส่งข้อมูลระหว่างกันของ MQTT Broker กับ Device ต่างนั้นจะต้องมีการกำหนด Topic เพื่อเป็นหัวข้อในการแลกเปลี่ยนข้อมูล เช่น การกำหนดว่าค่าที่อ่านจากเซนเซอร์ตรวจวัดค่าอุณภูมิของอากาศจะใช้ Topic ว่า AirTemp อุปกรณ์ที่มีหน้าที่อ่านค่าจากเซนเซอร์และส่งไปเก็บที่ MQTT Broker จะต้องทำการเผยแพร่ Topic AirTemp นี้ด้วยคำสั่ง Publish และ เมื่ออุปกรณ์ใดต้องการค่าอุณภูมินี้ไปใช้งานไม่ว่าจะเอาไปแสดงผล หรือจะเอาไปเป็นเงื่อนไขในการทำงานต่อก็ตาม อุปกรณ์นั้นจะต้องติดต่อไปที่ MQTT Broker เพื่อขอติดตาม Topic AirTemp ด้วยคำสั่ง Subscribe และเมื่อมีการ Publish ค่า AirTemp จาออุปกรณ์มายัง MQTT Broker แล้ว MQTT Broker จะทำหน้าที่ส่งต่อข้อมูล AirTemp ไปให้กับ อุปกรณ์ที่ Subscribe Topic AritTemp ทุกตัวแบบ Real-Time

สำหรับ NETPIE2020 จะมีการส่งข้อมูลด้วย MQTT Protocol อยู่ 2 รูปแบบคือการใช้ Topic @shadow/data/update ตามที่นักศึกษาได้ใช้ในการทดลองขั้นต้นก่อนหน้านี้ ซึ่งเป็นการอัพเดทค่าใน Shadow Data โดยส่ง Payload ใน รูปแบบ JSON และเราไม่สามารถระบุชื่อ Topic เองได้ และอีกแบบคือ Topic @msg/topic โดยในรูปแบบนี้ตรงส่วน topic เราจะสามารถกำหนดได้เอง และการส่งสามารถส่งค่าในรูปแบบตัวแปลทั่วไปได้

สำหรับหัวข้อการทดลองถัดไปนี้ให้นักศึกษานำโค๊ดเดิม มาพัฒนาต่อโดยการคัดลอกและทำการเปลี่ยนชื่อเป็น   
LAB6\_2\_รหัสนักศึกษา 3 ตัวหลัง.py จากนั้นปฏิบัติตามคำสั่งต่อไปนี้

1. เข้าไปที่ NETPIE2020 แก้ไขชื่อ Device ใน Device List ให้เป็น Node1
2. สร้าง Device ใหม่ขึ้นมากำหนดชื่อเป็น Node2
3. สร้างไฟล์ชื่อ LAB6\_2\_-รหัสนักศึกษา3ตัวหลัง.py
4. ทดลองรันโปรแกรมว่าสามารอติดต่อ Node1 ได้

# Node1

import time

import paho.mqtt.client as mqtt

import random

import json

NETPIE\_HOST = "mqtt.netpie.io"

# Client ID ของ Device ที่สร้างขึ้นใน NETPIE

CLIENT\_ID = "45981911-6f1d-4f15-817a-0605de9f786d"

# Token ของ Device ที่สร้างขึ้นใน NETPIE

DEVICE\_TOKEN = "TAF6zrUrxutcoSYGvzm4zLjcHWhP6Ecy"

sensor\_data = {'temperature': 0, 'humidity': 0}

def on\_connect(client, userdata, flags, rc):

    print("Result from connect: {}:".format(mqtt.connack\_string(rc)))

    client.subscribe("@shadow/data/updated")

client = mqtt.Client(protocol=mqtt.MQTTv311, client\_id=CLIENT\_ID, clean\_session=True)

client.username\_pw\_set(DEVICE\_TOKEN)

client.on\_connect = on\_connect

#client.on\_message = on\_message

client.connect(NETPIE\_HOST, 1883)

client.loop\_start()

try:

    while True:

        temperature = random.randrange(0, 50)

        humidity = random.randrange(20, 100)

        print("Temp={0: 0.1f}\*C Humidity={1: 0.1f} %  ".format(temperature, humidity))

        sensor\_data["temperature"] = temperature

        sensor\_data["humidity"] = humidity

        print(json.dumps({"data": sensor\_data}))

        client.publish("@shadow/data/update",json.dumps({"data": sensor\_data}), 1)

        time.sleep(5)

except KeyboardInterrupt:

    pass

1. จากข้อ 4 หากโปรแกรมทำงานได้ตามปกติให้ทำการแก้ไขให้สามารถแสดงค่าจากการ Subscribe ทางจอภาพ โดยแก้ไขโค๊ดตามนี้

def on\_connect(client, userdata, flags, rc):

    print("Result from connect: {}:".format(mqtt.connack\_string(rc)))

    client.subscribe("@shadow/data/updated")

def on\_message(client, userdata, msg):

    data1 = str(msg.payload).split(",")

    data2 = (data1[2]).split(":")

    data3 = (data2[1]).split("}")

    Humi = data3[0]

    print("Humidity is    : ", Humi)

    data4 = (data1[1]).split(":")

    Temp = data4[2]

    print("Temperature is : ", Temp)

client = mqtt.Client(protocol=mqtt.MQTTv311,

                     client\_id=CLIENT\_ID, clean\_session=True)

client.username\_pw\_set(DEVICE\_TOKEN)

client.on\_connect = on\_connect

client.on\_message = on\_message

client.connect(NETPIE\_HOST, 1883)

client.loop\_start()

try:

    while True:

        temperature = random.randrange(0, 50)

        humidity = random.randrange(20, 100)

        print(

            "Temp={0: 0.1f}\*C Humidity={1: 0.1f} %  ".format(temperature, humidity))

        sensor\_data["temperature"] = temperature

        sensor\_data["humidity"] = humidity

        print(json.dumps({"data": sensor\_data}))

        client.publish('@shadow/data/update',

                       json.dumps({"data": sensor\_data}), 1)

        time.sleep(5)

except KeyboardInterrupt:

    pass

วิเคราะห์โค๊ดส่วนนี้เพื่อหารูปแบบการแยกข้อมูลให้ได้เฉพาะที่ต้องการ

**ทดลองรันโปรแกรมบันทึกผล .........................................................................................................................................**

1. สร้างไฟล์ชื่อ LAB6\_3\_รหัสนักศึกษา3ตัวหลัง.py แก้ไขโค๊ดให้ทำหน้าที่ Subscribe อย่างเดียวดังนี้

import time

import paho.mqtt.client as mqtt

import random

import json

NETPIE\_HOST = "mqtt.netpie.io"

# Client ID ของ Device ที่สร้างขึ้นใน NETPIE

CLIENT\_ID = ".................Node 2 …………………………………………………………"

# Token ของ Device ที่สร้างขึ้นใน NETPIE

DEVICE\_TOKEN = ".................Node 2 …………………………………………………………"

def on\_connect(client, userdata, flags, rc):

    print("Result from connect: {}:".format(mqtt.connack\_string(rc)))

    client.subscribe("@shadow/data/updated")

def on\_message(client, userdata, msg):

    data1 = str(msg.payload).split(",")

    data2 = (data1[2]).split(":")

    data3 = (data2[1]).split("}")

    Humi = data3[0]

    print("Humidity is    : ", Humi)

    data4 = (data1[1]).split(":")

    Temp = data4[2]

    print("Temperature is : ", Temp)

client = mqtt.Client(protocol=mqtt.MQTTv311,

                     client\_id=CLIENT\_ID, clean\_session=True)

client.username\_pw\_set(DEVICE\_TOKEN)

client.on\_connect = on\_connect

client.on\_message = on\_message

client.connect(NETPIE\_HOST, 1883)

client.loop\_start()

while True:

    client.loop\_start()

# ref

# https://www.youtube.com/watch?v=AkemU\_\_8e8s&t=5891s

# https://netpie.io/tutorials/th/RaspberryPi

**ทดลองรันโปรแกรมทั้ง 2 โปรแกรมบันทึกผลสถานะที่ NETPIE แลบะผลการรันโปรแกรมทั้ง 2 โปรแกรม**

**.....................................................................................................................................................................**

1. สร้างไฟล์ชื่อ LAB6\_4\_รหัสนักศึกษา3ตัวหลัง.py มาทำการแก้ไขเปลี่ยนวิธีการ Subscribe โดยใช้ @msg/topic แทน

import time

import paho.mqtt.client as mqtt

import random

NETPIE\_HOST = "mqtt.netpie.io"

# Client ID ของ Device ที่สร้างขึ้นใน NETPIE

CLIENT\_ID = ".................Node 1 …………………………………………………………"

# Token ของ Device ที่สร้างขึ้นใน NETPIE

DEVICE\_TOKEN = ".................Node 1 …………………………………………………………"

def on\_connect(client, userdata, flags, rc):

    print("Result from connect: {}:".format(mqtt.connack\_string(rc)))

    client.subscribe("@msg/farm")

def on\_message(client, userdata, msg):

    data\_ = str(msg.payload)

    print(data\_)

client = mqtt.Client(protocol=mqtt.MQTTv311,

                     client\_id=CLIENT\_ID, clean\_session=True)

client.username\_pw\_set(DEVICE\_TOKEN)

client.on\_connect = on\_connect

client.on\_message = on\_message

client.connect(NETPIE\_HOST, 1883)

client.loop\_start()

try:

    while True:

        temperature = random.randrange(0, 50)

        humidity = random.randrange(20, 100)

        datasend = str(temperature) + "," + str(humidity)

        client.publish("@msg/farm", datasend, 1)

        time.sleep(5)

except KeyboardInterrupt:

    pass

1. สร้างไฟล์ชื่อ LAB6\_5\_รหัสนักศึกษา3ตัวหลัง.py คัดลอกโค๊ดจาก LAB6\_3 มาทำการแก้ไขเปลี่ยนวิธีการ Subscribe โดยใช้ @msg/topic แทน

import paho.mqtt.client as mqtt

NETPIE\_HOST = "mqtt.netpie.io"

# Client ID ของ Device ที่สร้างขึ้นใน NETPIE

CLIENT\_ID = ".................Node 2 …………………………………………………………"

# Token ของ Device ที่สร้างขึ้นใน NETPIE

DEVICE\_TOKEN = ".................Node 2 …………………………………………………………"

def on\_connect(client, userdata, flags, rc):

    print("Result from connect: {}:".format(mqtt.connack\_string(rc)))

    client.subscribe("@msg/farm")

def on\_message(client, userdata, msg):

    data\_ = (msg.payload)

    datarecv = (data\_).decode("utf-8").split(",")

    temp = datarecv[0]

    humi = datarecv[1]

    print("Data Recive : ", datarecv)

    print("Temperature", temp)

    print("Humidity", humi)

client = mqtt.Client(protocol=mqtt.MQTTv311,

                     client\_id=CLIENT\_ID, clean\_session=True)

client.username\_pw\_set(DEVICE\_TOKEN)

client.on\_connect = on\_connect

client.on\_message = on\_message

client.connect(NETPIE\_HOST, 1883)

client.loop\_start()

while True:

    client.loop\_start()

# ref

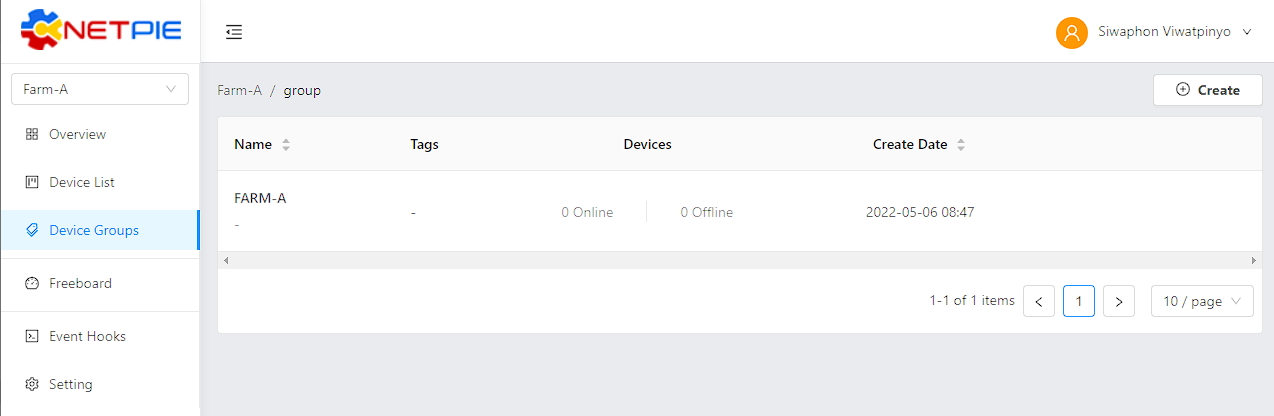
# https://www.youtube.com/watch?v=AkemU\_\_8e8s&t=5891s

# https://netpie.io/tutorials/th/RaspberryPi

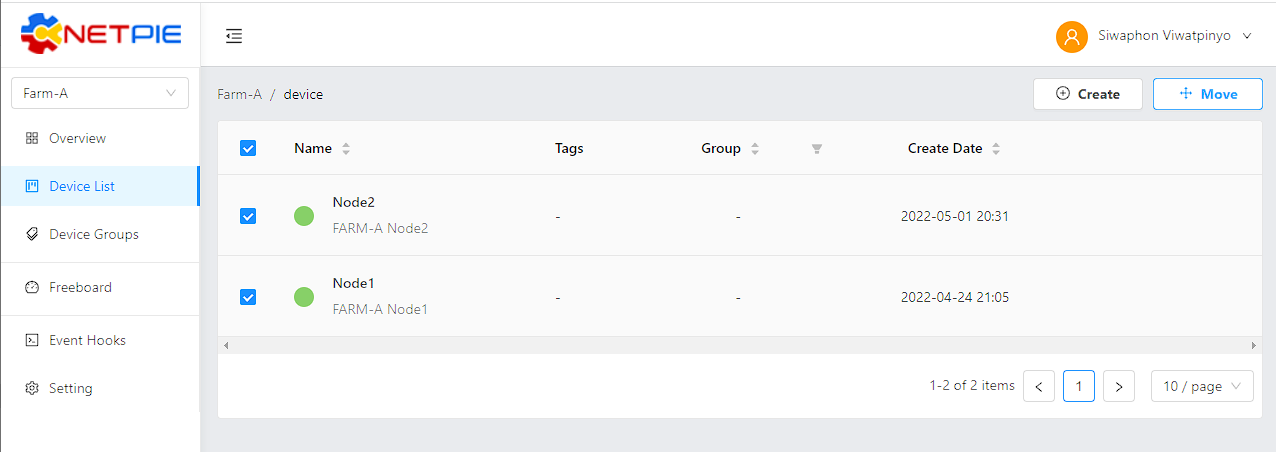
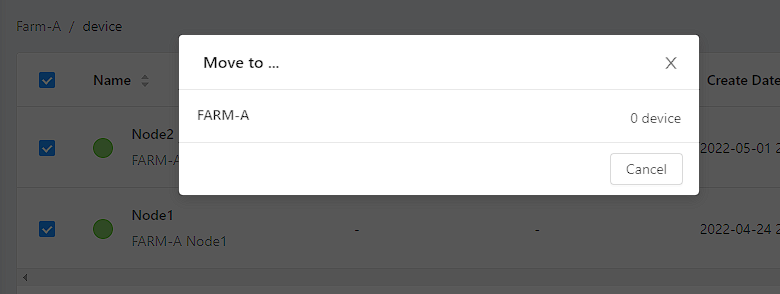
**รันโปรแกรม** LAB6\_4 **และ** LAB6\_5 **บันทึกผลการรัน**

**................................................................................................................................................................................**

1. ที่ NETPIE ไปที่เมนู Device Groups เลือกที่ Create เพื่อสร้างกรุ๊ปชื่อ FARM-A



1. กลับไปที่ Device List ติ๊กถูกเลือก Node1 และ Node2 จากนั้นเลือก Move เลือกชื่อ FARM-A เพื่อย้าย Node1 และ Node2 เข้าไปอยู่ในกลุ่มเดียวกันคือกรุ๊ป FARM-A



**1**

**2**

**3**

**4**

.

1. สังเกตุที่ข้อมูล Group ของ Node ทั้งสองจะเป็นชื่อกรุ๊ปเดียวกัน ให้กลับไปดูผลการรันโปรแกรม LAB6\_4 **และ** LAB6\_5ใหม่บันทึกผล

**บันทึกผลการรันโปรแกรม...................................................................................................................**

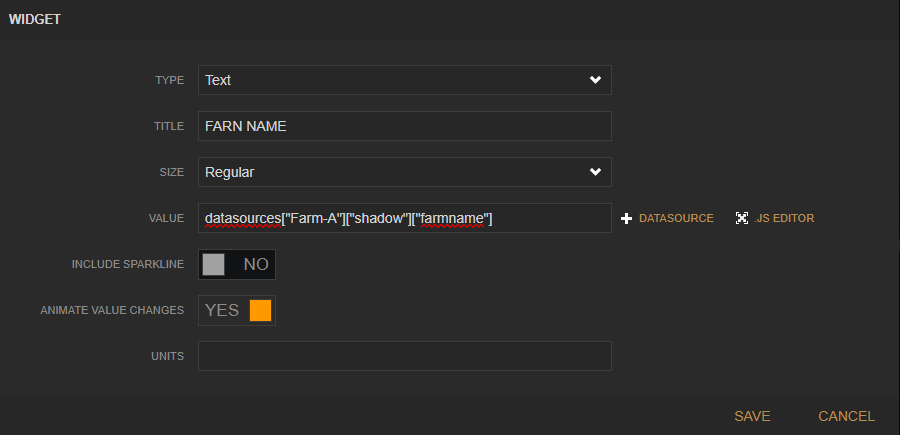
**อธิบายสาเหตุที่เป็นเช่นนั้นโดยการค้นคว้า................................................................................................**

1. ทดลองเพิ่ม Node ให้ครบ 5 Node และกำหนดค่าให้อยู่กรุ๊ปเดียวกับ Node1 และ Node2 โดยกำหนดค่า topic ของ Node3 เป็น water และสมมุติให้เป็นเซนเซอร์วัดระดับน้ำทำหน้าที่ publish ค่าและมีการบันทึกผลการรัน ส่วน Node4 รับเฉพาะค่าระดับน้ำจาก Node3 เท่านั้น ส่วน Node5 ให้รับค่าของ Node1 และ Node3 มาแสดงโดยนักศึกษาปรับแต่งรูปแบบการแสดงผลให้ดูง่าย เข้าใจง่ายว่าค่าเหล่านั้นมาจากไหนบ้าง สร้าง Folder ชื่อ LAB6-12 นำเฉพาะโค๊ดทั้งหมดของข้อ 12 ใส่รวมกันและทำการบีบอัดเป็น .rar หรือ .zip ส่งเข้าระบบร่วมกับไฟล์อื่นๆ

**บันทึกผลการรันโปรแกรม...................................................................................................................**

**Free Board การสร้างและเรียกใช้** Schema

1. **ให้นักศึกษาสร้าง Widget TEXT ที่ NETPIE โดยกำหนดค่าดังนี้**



1. ให้คัดลอกไฟล์ LAB6\_1 มาทำการแก้ไขเพื่อให้ส่งค่าได้อย่างเดียว และให้เพิ่มการส่งค่าข้อความชื่อฟาร์มไปด้วย โดยสร้างไฟล์ใหชื่อ LAB6\_6\_xxx.py และคัดลอกโค๊ดมาวาง แก้ไขให้โค๊ดทำหน้าที่ Publish อย่างเดียวแบบ shadow

**ทดลองรันโปรแกรมและบันทึกผล**

**1. บันทึกผลการรันโปรแกรม ..................................................................................**

**2. บันทึกผลของ Widget TEXT...................................................................................**

1. เลือกที่ Node1 ใน Device List เลือกไปที่ Schema เลือกที่ Tree เปลี่ยนเป็น Code คัดลอกโค๊ดต่อไปนี้ไปวาง

{

"additionalProperties": false,

"properties": {

"humidity": {

"operation": {

"store": {

"ttl": "7d"

}

},

"type": "number"

},

"temperature": {

"operation": {

"store": {

"ttl": "7d"

},

"transform": {

"expression": "(($.temperature)\*1.8) + 32"

}

},

"type": "number"

},

"farmname": {

"operation": {

"store": {

"ttl": "7d"

}

},

"type": "string"

}

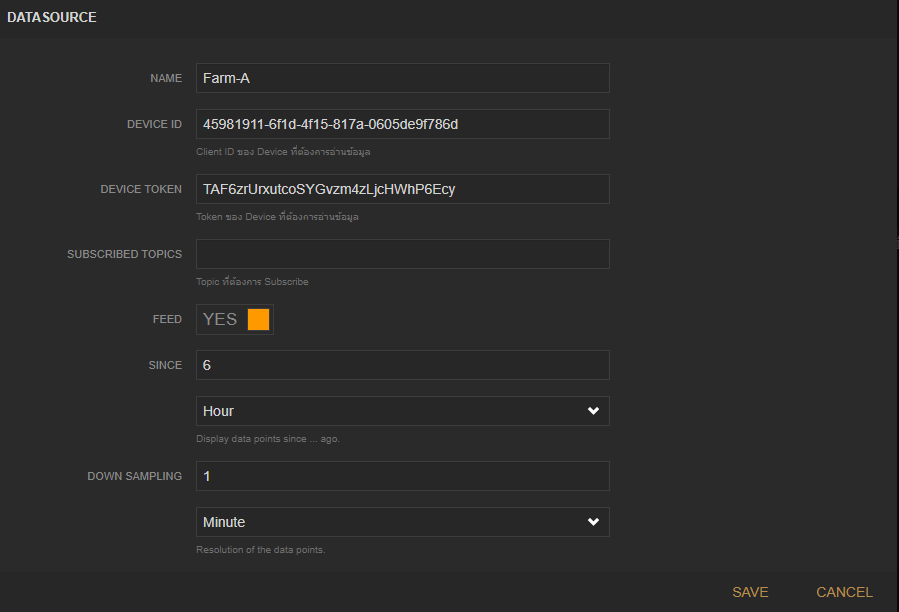
}

}

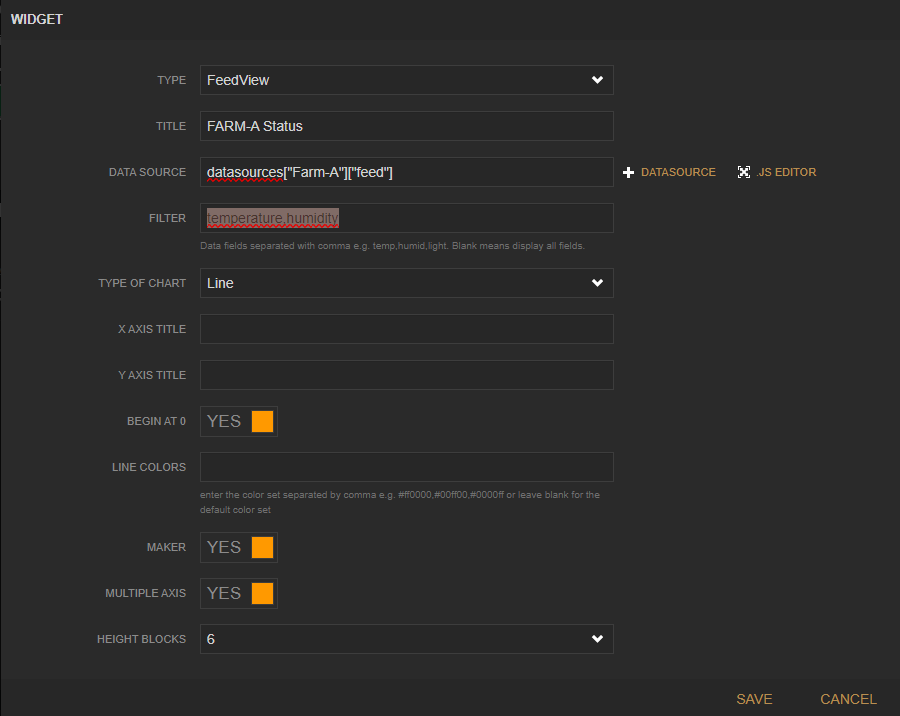
ในการจะใช้งาน FEED ซึ่งมีโควตาจำกัดควรระวังด้วย จำเป็นต้องมาสร้าง Schema เพื่อกำหนดค่าให้มีการบันทึกค่าไว้ในลักษณะ Timeseries โดยสามารถกำหนดระยะเวลาให้ค่าคงอยู่ได้เพื่อให้ FEED สามมรถดึงค่าไปแสดงผลได้ในใบงานนี้กำหนดไว้ที่ 7 วัน นอกจากนี้ยังสามารถใช้ฟังก์ชั่นแปลงค่า(transform) และคำนวณค่า(expression)ได้โดยในใบงานนี้จะแปลงจากเซลเซียลให้เป็นฟาเรนไฮ

ในการกำหนดค่าต้องกำหนดชื่อ properties ให้ตรงกับค่าที่รับมาจาก Device

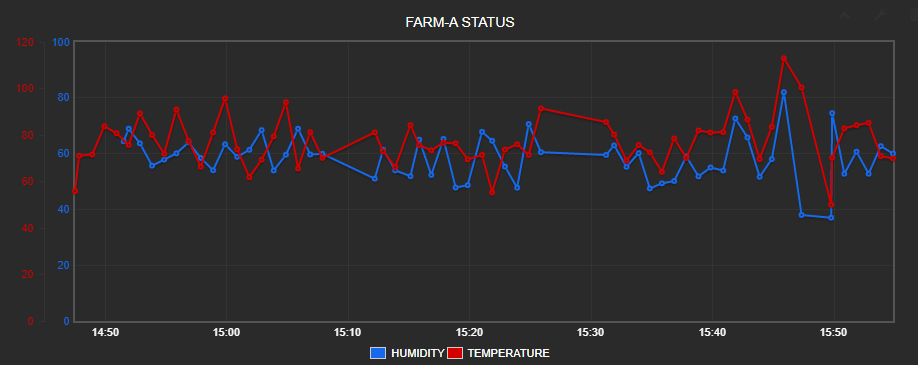
1. เข้าไปที่ Freeboard เลือกที่ Datasource เลือกที่ FEED ให้เป็น YES



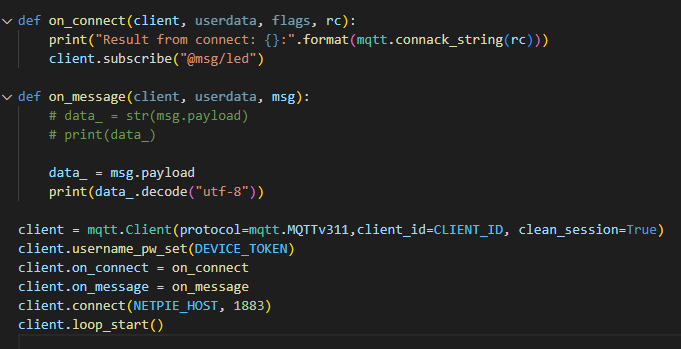
1. เพิ่ม PANE เข้ามาจากนั้นเพิ่ม Widget FeedView เข้ามา กำหนดชื่อ TITLE เลือก DATASOURCE และเลือกเป็น FEED ในส่วนของ Filter ให้ระบุค่า TOPPIC ที่ต้องการที่โปรแกรมเราส่งมา ในที่นี้คือค่า temperature,humidity ให้ตรงกับในโปรแกรม เลือก TYPE OF CHART เป็น Line และเลือก Begin at 0 เป็น YES กด Save



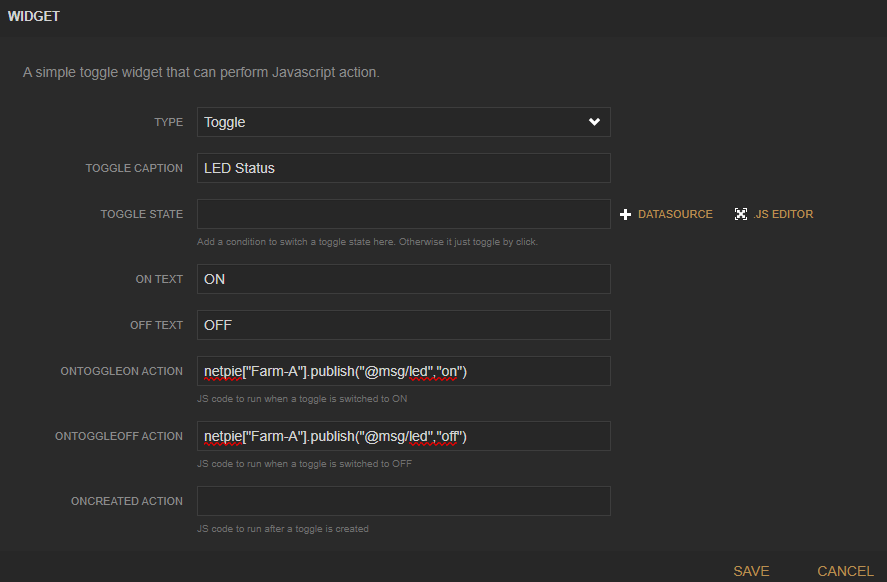
ตัวอย่างการแสดงผลของ FEED ขณะรันโปรแกรม LAB6\_6 ได้ด้วย



1. แก้ไขโปรแกรมให้ Subscribe ด้วย topic ดังนี้ @msg/led เพิ่ม on\_message ให้แสดงค่าที่ subscribe เข้ามา



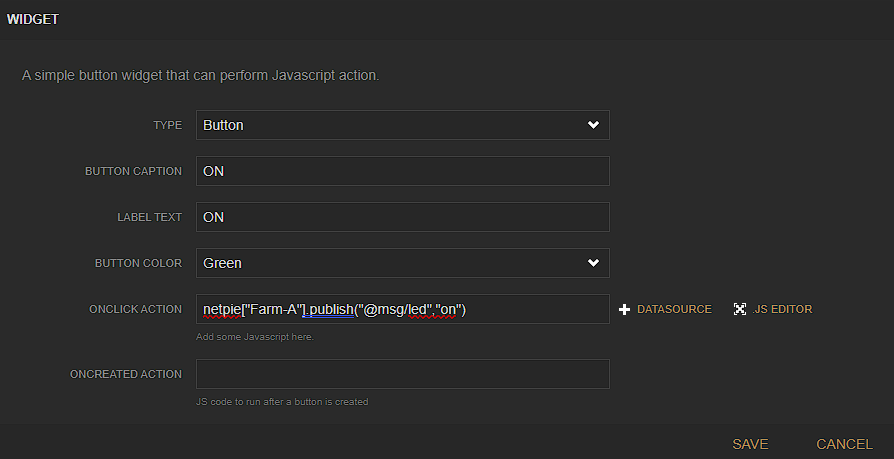
1. ไปที่ Freeboard เพิ่ม PANE เข้ามาจากนั้นเพิ่ม toggle กำหนดค่าดังนี้ ยกเว้นส่วนของ datasource กำหนดให้ตรงตามที่นักศึกษากำหนดหากมีการเปลี่ยนไปจากใบงาน จากนั้นกดบันทึก

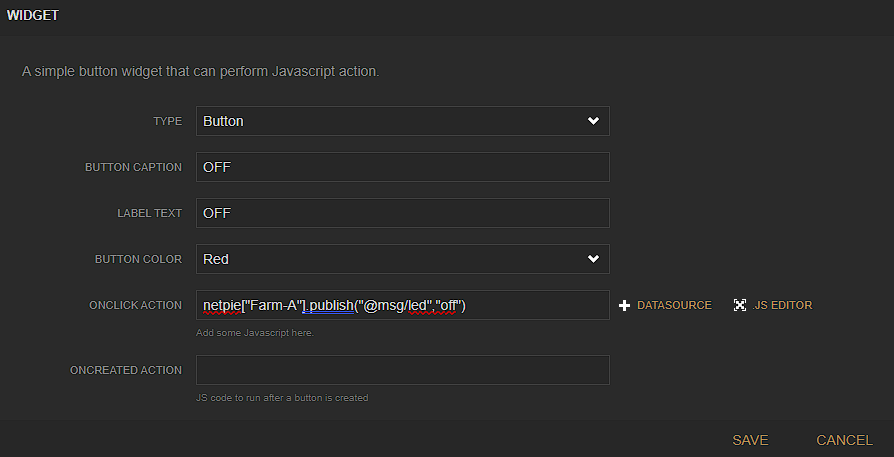


รันโปรแกรมและกดที่ Toggle สังเกตการณ์เปลี่ยนแปลงที่ผลการทำงานของโปรแกรม และบันทึกผลทั้ง 2 ส่วน

**บันทึกผล**...................................................................................................................................................................

1. ที่ PANE เดียวกับ Toggle เพิ่ม Button มา 2 ตัวโดยให้ทั้งต้อง 2 Publish ด้วย topic @msg\led โดยตัวแรกให้ส่งค่า on และอีกตัวส่งค่า off ดังตัวอย่าง





1. ให้แก้ไขโค๊ดโดยมีเงื่อนไขว่าหากมีการกดปุ่ม ON/OFF และ Toggle ที่ Freeboard ให้แสดงข้อความว่า Button On Push, Button Off Push และ Toggle On Push / Toggle Off Push

**บันทึกผลการรัน .........................................................................................................................................................**

1. **ให้นักศึกษาออกแบบระบบโดยพยายามใช้ความรู้ที่ได้ทำใบงานมา พร้อมทั้งการค้นคว้าเพิ่มเติม โดยใช้ NETPIE และสามารถเพิ่มระบบอื่นๆ ที่สามารถทำงานร่วมกันได้ เช่น Firebase, Database, Web เป็นต้น งานรายบุคคลห้ามซ้ำกัน งานที่ซ้ำคะแนนเต็มจะหารด้วยจำนวนคนซ้ำครับ**