



# เริ่มต้นใช้งาน NETPIE กับ ESP8266/NodeMCU-12E

2nd edition

ธีรุณ จิตราพรมา<sup>๑</sup>  
ชัยวัฒน์ ลิ้มพrajิตรวีไล



[www.inex.co.th](http://www.inex.co.th)

## การใช้งาน NETPIE กับ ESP8266/NodeMCU-12E

มีรากฐาน จิตราพรอมมา

ข้อมูลนี้ได้มาจากการสำรวจ

สงวนลิขสิทธิ์ตาม พ.ร.บ. ลิขสิทธิ์ พ.ศ. 2521

ห้ามการลอกเลียนแปลงส่วนหนึ่งส่วนใดของหนังสือเล่มนี้ นอกจะได้รับอนุญาต

ดำเนินการจัดพิมพ์และจำหน่ายโดย

บริษัท อินโนเวติฟ เอ็กเพอริเมนต์ จำกัด

108 ช.สุขุมวิท 101/2 ถ.สุขุมวิท แขวงบางนา เขตบางนา กรุงเทพฯ 10260

โทรศัพท์ 0-2747-7001-4

โทรสาร 0-2747-7005

รายละเอียดที่ปรากฏในหนังสือเล่มนี้ผ่านการตรวจทานอย่างละเอียดและถ้วนถี่ เพื่อให้มีความสมบูรณ์ และถูกต้องมากที่สุดภายใต้เงื่อนไขและเวลาที่พึงมีก่อนการจัดพิมพ์เผยแพร่ ความเสียหายอันอาจเกิดจากการนำข้อมูลในหนังสือเล่มนี้ไปใช้ ทางบริษัท อินโนเวติฟ เอ็กเพอริเมนต์ จำกัด มิได้มีภาระในการรับผิดชอบแต่ประการใด ความผิดพลาดคลาดเคลื่อนที่อาจมีและได้รับการจัดพิมพ์เผยแพร่ออกไป นั้น ทางบริษัทฯ จะพยายามชี้แจงและแก้ไขในการจัดพิมพ์ครั้งต่อไป

# คำชี้แจงจากคณ.:ผู้เขียน/เรียบเรียง

---

การนำเสนอข้อมูลเกี่ยวกับข้อมูลทางเทคนิคและเทคโนโลยีในหนังสือเล่มนี้ เกิดจากความต้องการที่จะอธิบายกระบวนการและหลักการทำงาน ของอุปกรณ์ในภาพรวมด้วยถ้อยคำที่ง่ายเพื่อสร้างความเข้าใจแก่ผู้อ่าน ดังนั้นการแปลคำศัพท์ทางเทคนิคหลายๆ คำอาจไม่ตรงตามข้อบัญญัติของราชบัญญัติยสถาน และมีหลายๆ คำที่ยังไม่มีการบัญญัติอย่างเป็นทางการ คณ.:ผู้เขียนจึงขออนุญาตบัญญัติศัพท์ขึ้นมาใช้ในการอธิบาย โดยมีข้อจำกัดเพื่อข้างอิงในหนังสือเล่มนี้เท่านั้น

ทั้งนี้สาเหตุหลักของข้อชี้แจงนี้มาจากการรับรวมข้อมูลของอุปกรณ์ในระบบสมองกลฝังตัว และเทคโนโลยีหุ่นยนต์สำหรับการศึกษาเพื่อนำมาเรียบเรียงเป็นภาษาไทยนั้นทำได้ไม่ง่ายนัก ทางคณ.:ผู้เขียนต้องทำการรวบรวมและทดลองเพื่อให้แน่ใจว่า ความเข้าใจในกระบวนการทำงานต่างๆ นั้นมีความคลาดเคลื่อนน้อยที่สุด

เมื่อต้องทำการเรียบเรียงอุปกรณ์เป็นภาษาไทย ศัพท์ทางเทคนิคหลายคำมีความหมายที่ทับซ้อนกันมาก การบัญญัติศัพท์จึงเกิดจาก การปฏิบัติจริงร่วมกับความหมายทางภาษาศาสตร์ ดังนั้นหากมีความคลาดเคลื่อนหรือผิดพลาดเกิดขึ้น ทางคณ.:ผู้เขียนขออภัยรับและหากได้รับคำอธิบายหรือชี้แนะจากท่านผู้จะได้ทำการชี้แจงและปรับปรุงข้อผิดพลาดที่อาจมีเหล่านั้นโดยเร็วที่สุด

ทั้งนี้เพื่อให้การพัฒนาสื่อทางวิชาการ โดยเฉพาะอย่างยิ่งกับความรู้ของเทคโนโลยีสมัยใหม่ สามารถดำเนินไปได้อย่างต่อเนื่อง ภายใต้การมีส่วนร่วมของผู้สนใจทุกภาคส่วน

ในหนังสือเล่มนี้ได้ทำการอ้างอิงถึงซอฟต์แวร์และเว็บไซต์หลายแห่ง ทางผู้จัดทำไม่อาจรับประกัน หรือรับรองการคงอยู่ของสินค้าและบริการใดๆ ที่นำเสนอหรืออ้างอิงในหนังสือเล่มนี้ อย่างไรก็ตาม ในขณะจัดทำหนังสือเล่มนี้ ทางผู้จัดทำได้ทำการทดลองและทดสอบภาษาไทย ครอบเวลาและภาษาปรับปรุงล่าสุดในเวลานั้นๆ หากมีการปรับปรุงอื่นๆ ให้ทำการนำเสนอด้วยคลาดเคลื่อนหรือใช้ประโยชน์ไม่ได้ ทางผู้จัดทำจะพยายามปรับปรุงเนื้อหาและเผยแพร่ผ่านทางเว็บไซต์และสื่อสารณะของบริษัท อินโนเวติฟ เอ็กเพอริเม้นต์ จำกัด ที่ [www.inex.co.th](http://www.inex.co.th) และ [www.facebook.com/innovativeexperiment](https://www.facebook.com/innovativeexperiment)

# สารบัญ

---

บทที่ 1 การใช้งาน NETPIE กับ NodeMCU-12E.....	5
บทที่ 2 ไลบรารีของ NETPIE สำหรับการติดต่อแบบ MQTT.....	21
บทที่ 3 ความรู้เบื้องต้นเกี่ยวกับ NETPIE REST API .....	49
บทที่ 4 การประยุกต์ใช้งาน NETPIE ร่วมกับ freeboard.io เพื่อแสดงผลการทำงาน.....	57
บทที่ 5 การประยุกต์ใช้งาน freeboard.io เพื่อสั่งงานอุปกรณ์ผ่าน NETPIE.....	71
บทที่ 6 การใช้งาน NETPIE freeboard.....	91

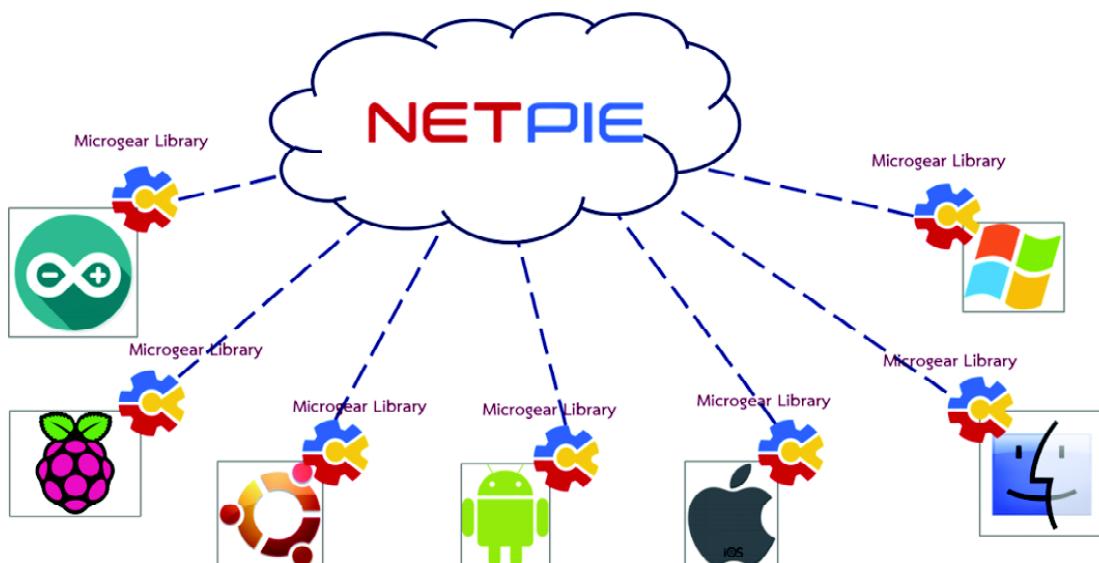
# บทที่ 1

## เริ่มต้นใช้งาน NETPIE กับ NodeMCU-12E

### 1.1 ข้อมูลเบื้องต้นของ NETPIE

#### 1.1.1 อะไรคือ NETPIE?

NETPIE คือ คลาวด์เซอร์ฟเวอร์ที่ให้บริการในรูปแบบ Platform-as-a-Service เพื่ออำนวยความสะดวกให้กับนักพัฒนาในการพัฒนาอุปกรณ์ของตัวเอง เช่น ต่อและเปลี่ยนข้อมูลกันได้ในแบบ Internet of Things หรือ IoT โดยคำว่า *NETPIE* มาจาก *Network Platform for Internet of Everything*



#### 1.1.2 ทำไมต้องเลือกใช้ NETPIE?

##### 1.1.2.1 ข้อดีของการใช้ทรัพยากรของการเชื่อมต่อ :

NETPIE ช่วยให้อุปกรณ์สามารถคุยกันได้โดยผู้พัฒนาไม่ต้องกังวลว่า อุปกรณ์นั้นจะอยู่ที่ใด เพียงนำไฟล์配置ของ NETPIE ไปติดตั้งในอุปกรณ์ NETPIE จะรับหน้าที่คุ้มครองการเชื่อมต่อให้ทั้งหมด ไม่ว่าอุปกรณ์นั้นจะอยู่ในเครือข่ายชนิดใด ลักษณะใด หรือแม้กระทั่งเคลื่อนย้ายไปอยู่ที่ใด ผู้พัฒนาสามารถตัดปัญหาความไขว่คว้าในการที่จะต้องมาออกแบบการเข้าถึงอุปกรณ์จากระยะไกล (remote access) ด้วยวิธีการเดิมๆ เช่น การใช้ fixed public IP หรือการตั้ง port forwarding ในเราเตอร์ หรือการต้องไปลงทะเบียนกับผู้ให้บริการ dynamic DNS ซึ่งทั้งหมดล้วนมีความยุ่งยากและลดความยืดหยุ่นของระบบ ไม่เพียงเท่านั้น NETPIE ยังช่วยให้การเริ่มต้นใช้งานเป็นไปโดยง่าย โดยออกแบบให้อุปกรณ์ ถูกกันพบและเข้าสู่บริการโดยอัตโนมัติ (automatic discovery, plug and play)

### 1.1.2.2 ช่วยลดภาระด้านความปลอดภัยของข้อมูล :

NETPIE ถูกออกแบบให้มีระดับและสิทธิ์ในการเข้าถึงในระดับ fine grain กล่าวคือ ผู้พัฒนาสามารถออกแบบได้เองทั้งหมด เช่น สิ่งใดมีสิทธิ์คุยกับสิ่งใด สิ่งใดมีสิทธิ์หรือไม่-เพียงใดในการอ่านหรือเขียนข้อมูล และสิทธิ์เหล่านี้จะมีอายุการใช้งานเท่าไหร่ หรือถูกเพิกถอนภายใต้เงื่อนไขใดเป็นต้น

### 1.1.2.3 ยึดหยุ่นต่อการขยายระบบ

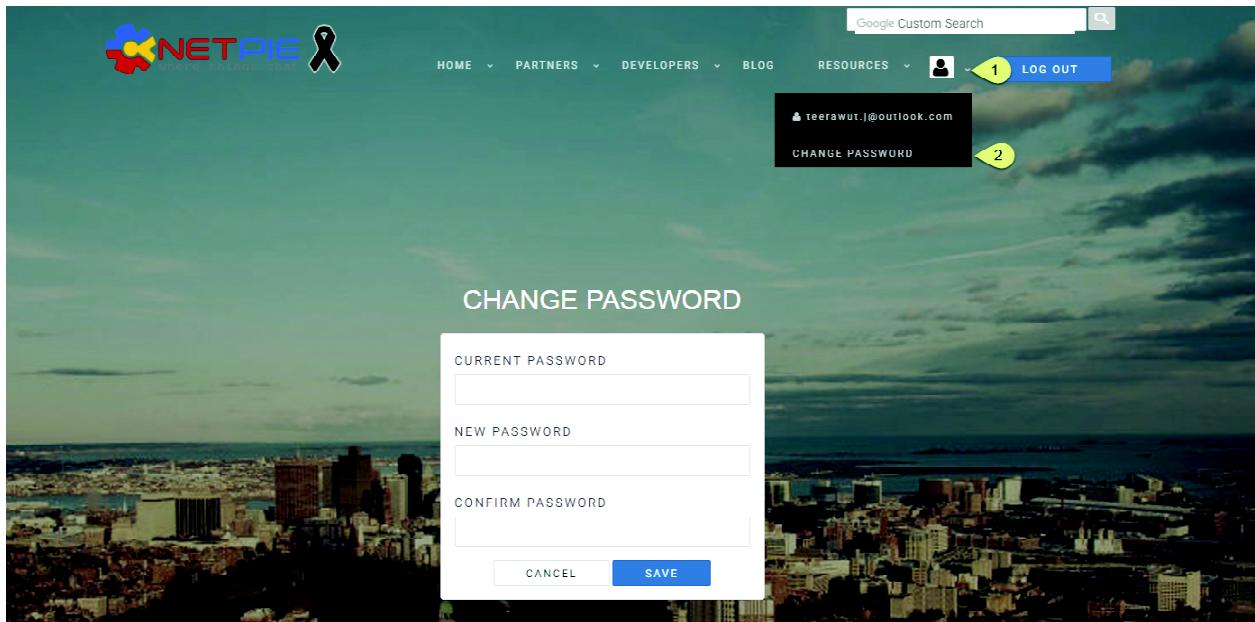
NETPIE มีสถาปัตยกรรมเป็นคลาวด์เซอร์ฟเวอร์ในทุกระดับของระบบ ทำให้เกิดความยึดหยุ่น และคล่องตัวสูงในการขยายตัว นอกจากนี้ ไม่คูดต่างๆ ยังถูกออกแบบให้ทำงานแยกจากกัน เพื่อให้เกิดสภาวะ loose coupling และต่อสารกันด้วยวิธีการ asynchronous messaging ช่วยให้แพล็ตฟอร์มมีความน่าเชื่อถือสูง นำไปใช้ช้าและพัฒนาต่อได้ง่าย

## 1.2 การเตรียมการสำหรับใช้งาน NETPIE

### 1.2.1 สมัครใช้งาน NETPIE

(1.2.1.1) ไปที่เว็บไซต์ [https://netpie.io/sign\\_up](https://netpie.io/sign_up) จะปรากฏหน้าเว็บดังรูปที่ 1-1 กรอกข้อมูลให้เรียบร้อย จากนั้นคลิกที่ปุ่ม SIGN UP เพื่อยืนยันการลงทะเบียน

รูปที่ 1-1 หน้าต่าง CREATE AN ACCOUNT สำหรับกรอกข้อมูลเพื่อลงทะเบียนใช้งาน NETPIE



รูปที่ 1-2 หน้าต่างเปลี่ยนรหัสผ่านใหม่

(1.2.1.2) รอรับข้อความสั้นหรือ SMS จากทาง NETPIE ซึ่งส่งไปยังหมายเลขโทรศัพท์เคลื่อนที่ที่ลงทะเบียนไว้

### ตัวอย่าง SMS

Your one-time password for NETPIE is 536815059323

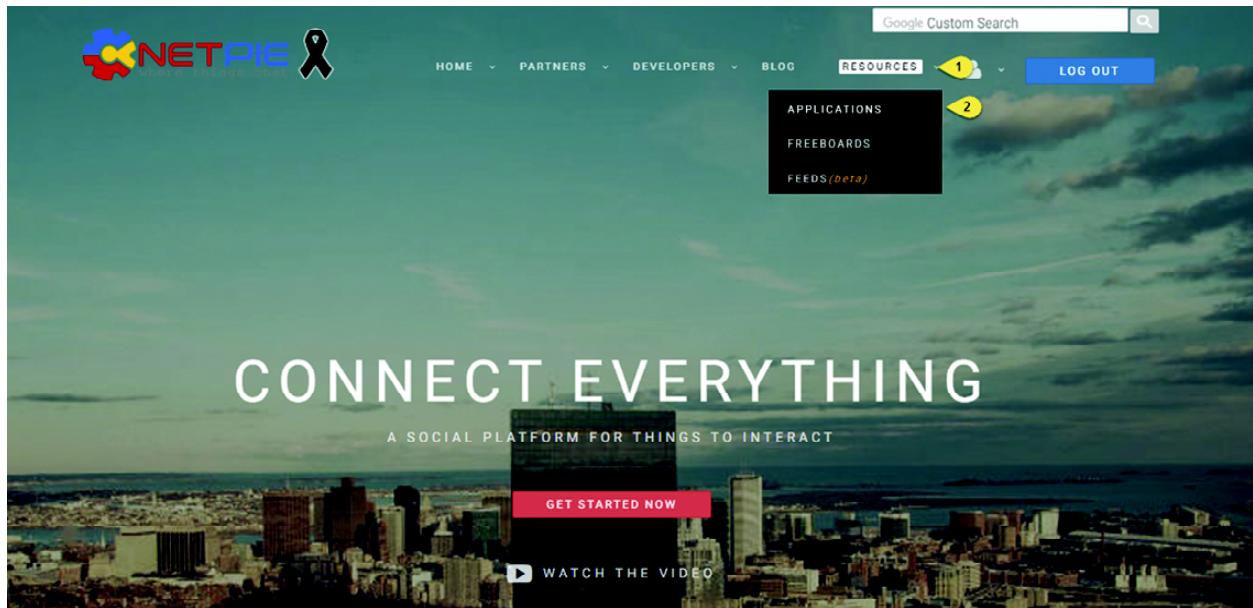
(1.2.1.3) คลิกที่เมนู **LOG IN** เพื่อเข้าสู่ระบบ นำอีเมลที่ลงทะเบียนไว้ใส่ในช่อง **USERNAME OR EMAIL ADDRESS** และนำรหัสผ่านที่ได้รับจาก SMS ใส่ในช่อง **PASSWORD** แล้วคลิกปุ่ม **LOGIN**

(1.2.1.4) แนะนำให้เปลี่ยนรหัสผ่าน โดยคลิกที่รูปไอคอนผู้ใช้งานที่มุมบนด้านซ้าย (ลูกศรหมายเลข 1 ในรูปที่ 1-2) จากนั้นเลือกไปที่เมนู **CHANGE PASSWORD** (ลูกศรหมายเลข 2 ในรูปที่ 1-2) หน้าต่าง **CHANGE PASSWORD** ปรากฏขึ้นมาดังรูปที่ 1-2 ใส่รหัสผ่านเดิมที่ช่อง **CURRENT PASSWORD** จากนั้นป้อนรหัสผ่านใหม่ที่ต้องการลงในช่อง **NEW PASSWORD** และช่อง **CONFIRM PASSWORD** แล้วคลิกปุ่ม **SAVE** เพื่อยืนยันการเปลี่ยนรหัสผ่าน

## 1.2.2 การสร้างแอปพลิเคชัน

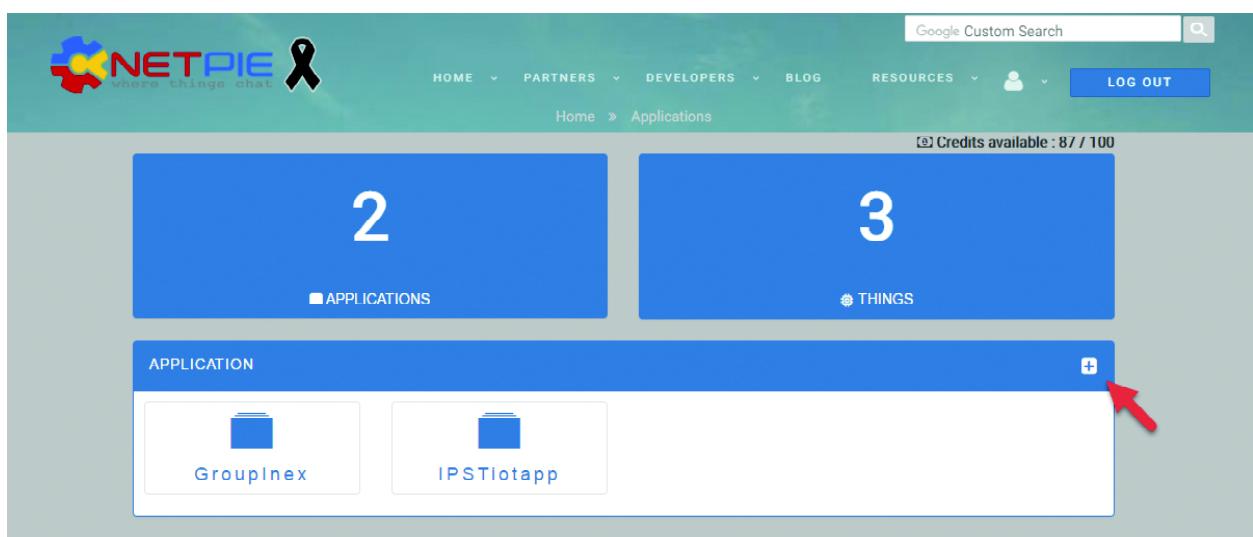
หลังจากสมัครสมาชิกและล็อกอินเรียบร้อยแล้ว ผู้ใช้งานสามารถสร้างแอปพลิเคชันตามขั้นตอนดังต่อไปนี้

(1.2.2.1) เลือกที่ **RESOURCES** คลิกที่เมนู **APPLICATIONS** เพื่อเข้าไปยังหน้า **APPLICATION MANAGEMENT** ดังรูปที่ 1-3

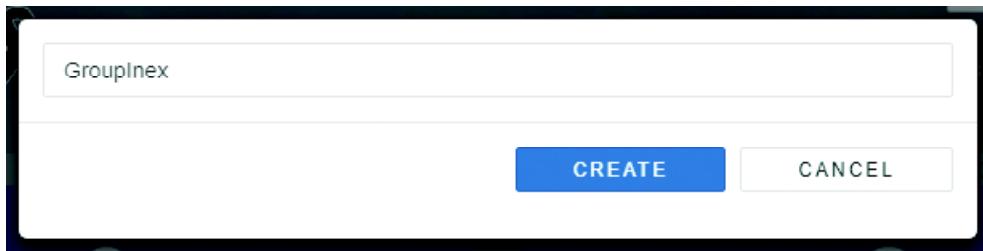


รูปที่ 1-3 เลือกเมนู APPLICATIONS เพื่อเตรียมการสร้างแอปพลิเคชันสำหรับเชื่อมต่อกับ NETPIE

(1.2.2.2) ปรากฏหน้าต่างแสดงแอปพลิเคชันทั้งหมดที่ผู้ใช้งานมีอยู่ การสร้างแอปพลิเคชันใหม่ทำได้โดยการคลิกที่ปุ่ม + ดังรูปที่ 1-4



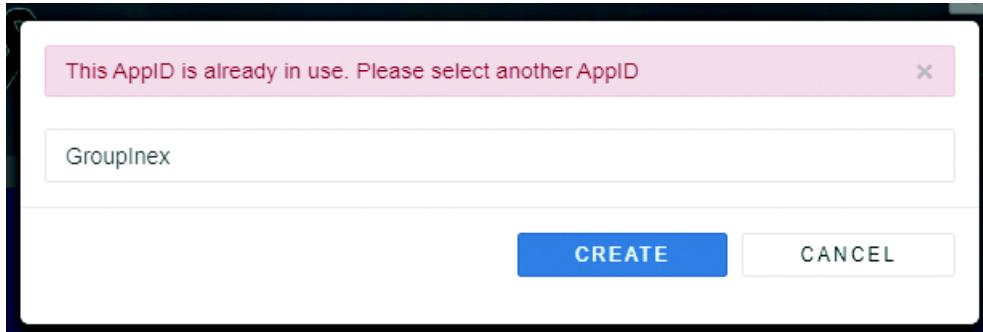
รูปที่ 1-4 แสดงการเพิ่มแอปพลิเคชันใหม่ โดยการคลิกปุ่ม +



รูปที่ 1-5 หน้าต่างสำหรับสร้างแอปพลิเคชันเพิ่มเติมใน NETPIE

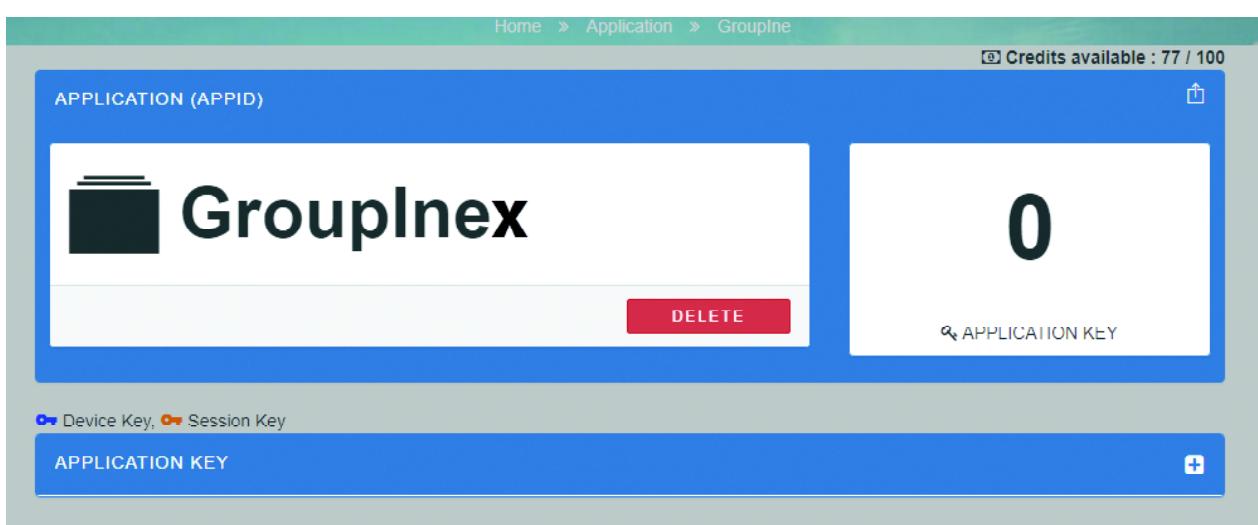
(1.2.2.3) ปรากฏหน้าต่างสำหรับกำหนดรหัสประจำตัวของแอปพลิเคชัน (**Application ID** หรือ **AppID**) ตามที่ต้องการ แล้วคลิกที่ปุ่ม **CREATE** ดังรูปที่ 1-5

(1.2.2.4) ถ้าหากมีหน้าต่างแจ้งเตือนตามรูปที่ 1-6 มีความหมายว่า ชื่อ **Application ID** หรือ **AppID** ที่เลือกไว้ใช้งานไม่ได้ เพราะมีผู้อื่นเลือกใช้งานไปแล้ว

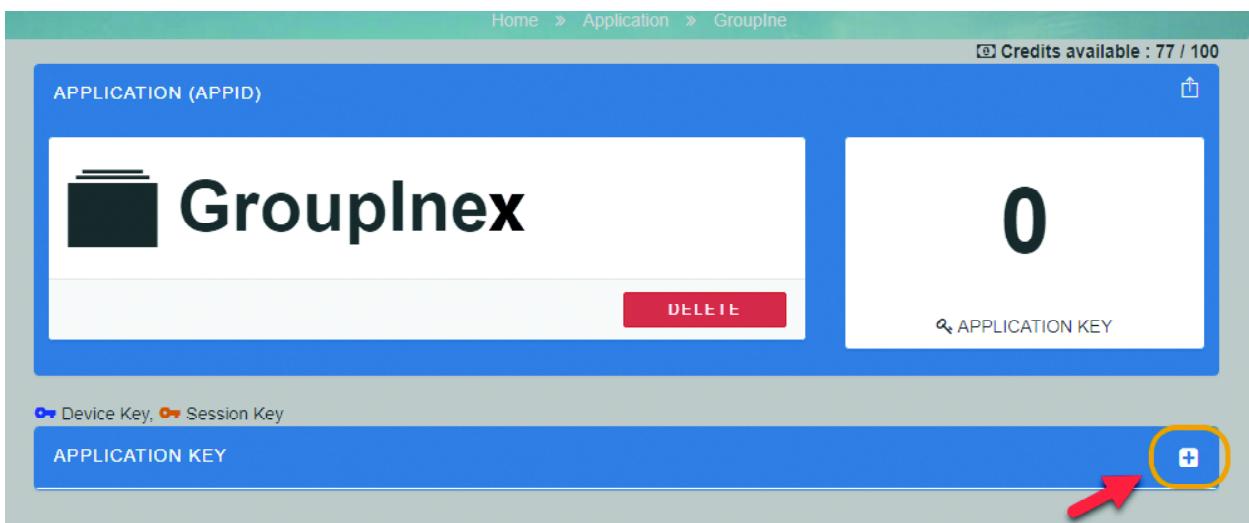


รูปที่ 1-6 หน้าต่างแจ้งเตือนว่า ชื่อของ **Application ID** หรือ **AppID** มีผู้อื่นเลือกใช้ไปแล้ว ต้องกำหนดใหม่

(1.2.2.5) หลังจากสร้าง **Application ID** สำเร็จ ระบบจะพาไปที่หน้าของแอปพลิเคชันที่สร้างขึ้นใหม่โดยอัตโนมัติ ดังรูปที่ 1-7



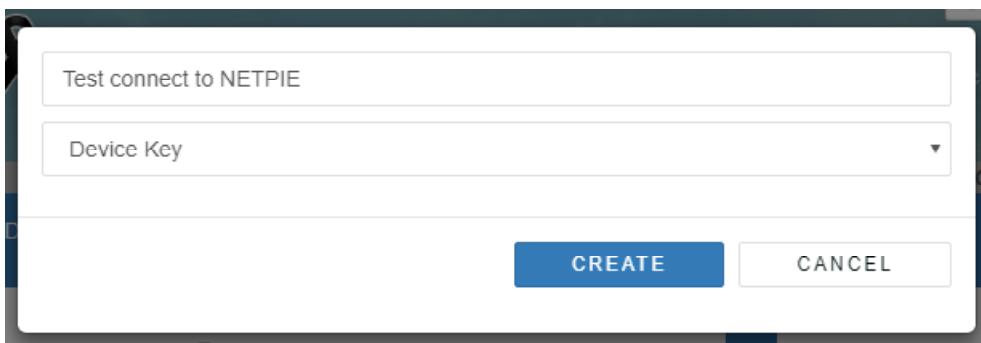
รูปที่ 1-7 หน้าของแอปพลิเคชันที่ได้รับการสร้างขึ้นใหม่โดยอัตโนมัติ หลังจากการกำหนดชื่อของ **Application ID** หรือ **AppID** ทำได้ถูกต้อง



รูปที่ 1-8 เริ่มสร้าง Application Key

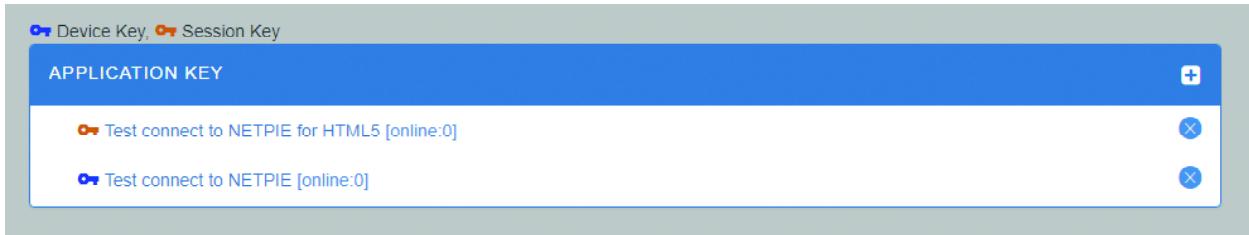
(1.2.2.6) สร้าง Application Key โดยคลิกที่ปุ่ม + ดังรูปที่ 1-8

(1.2.2.7) กำหนดชื่อของ Application Key ตามต้องการ ในตัวอย่างใช้ข้อความว่า **Test connect to NETPIE** และเลือกชนิดของ Key ใน Drop-down box ให้เป็น **Device Key** ดังรูปที่ 1-9



รูปที่ 1-9 กำหนดชื่อและชนิดของ Application Key

(1.2.2.8) เมื่อสร้าง Device Key ตามที่ต้องการ ได้แล้ว ผู้ใช้งานสามารถตรวจสอบโดยคลิกที่ชื่อ Key ที่สร้างไว้ดังรูปที่ 1-10



รูปที่ 1-10 Application Key ที่เกิดขึ้นใหม่ ในที่นี้คือ Test connect to NETPIE

## กรอบแยก 1 : เกี่ยวกับ Application Key

Application Key หรือรหัสกุญแจมี 2 ประเภท คือ

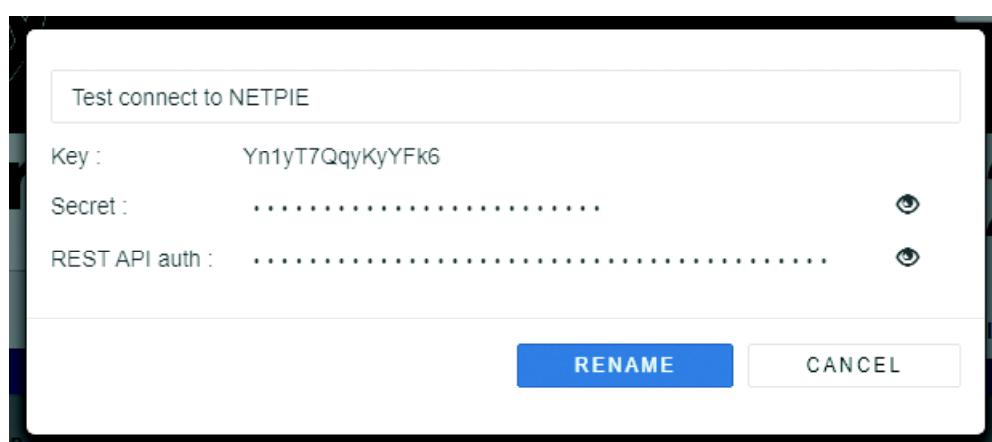
1. Device Key คือ รหัสกุญแจที่ใช้กับอุปกรณ์ประเภทภายนอก โดย Device Key ของอุปกรณ์นั้นจะถูกจดจำไปตลอด เมื่ออุปกรณ์จะไม่ได้เชื่อมต่อกับ NETPIE ตลอดเวลา เมื่อใด มีการเชื่อมต่อเข้ามาใหม่ จะถูกมองว่าเป็นอุปกรณ์ตัวเดิม จึงใช้ประโยชน์จากการหักกุญแจแบบนี้ ในการจดจำเพื่อข้างต้นถึงอุปกรณ์ตัวนั้นๆ ได้

2. Session Key คือ รหัสกุญแจที่ใช้กับอุปกรณ์ที่มีการใช้งานแบบไม่ถาวร เช่น บราวเซอร์ เป็นต้นจากยกเลิกการเชื่อมต่อหรือปิดบราวเซอร์ไป ตัวตนของอุปกรณ์นั้นจะถูกลบพิ้ง เมื่อเชื่อมต่อเข้ามาอีกครั้ง ระบบจะสร้างรหัสกุญแจใหม่ โดย NETPIE จะไม่ถือว่า การที่บราวเซอร์ออนไลน์ แต่ละครั้งมีตัวตนเดียวกัน ซึ่งแตกต่างจากการนี้ Device Key

จึงขอแนะนำให้ใช้ Device Key กับชาร์ดแวร์ เช่น ESP8266 ซึ่งในที่นี้ก็คือ NodeMCU-12E หรือบอร์ดคอมพิวเตอร์ขนาดเล็กอย่าง Raspberry Pi และเลือกใช้ Session Key กับบราวเซอร์ หรือแดชบอร์ดที่พัฒนาด้วย HTML5 เช่น Freeboard ที่ทาง NETPIE ได้จัดทำขึ้น

(1.2.2.9) ตรวจสอบข้อมูลรายละเอียด รหัสกุญแจต่างๆ โดยคลิกที่ **Test connect to NETPIE** รายละเอียดของรหัสกุญแจจะปรากฏขึ้นมาตามรูปที่ 1-11 อันประกอบด้วย

- **Application Key Name** ใช้ระบุอุปกรณ์ของผู้ใช้งานภายใต้ AppID หนึ่งเท่านั้น ต้องชื่อ ชี้กับผู้ใช้งานรายอื่นได้ และเปลี่ยนแปลงชื่อได้ตามที่ต้องการ
- **Key** เป็นกุญแจที่อุปกรณ์ใช้สำหรับเชื่อมต่อ NETPIE
- **Secret** เป็นรหัสลับที่ต้องใช้คู่กับ Key เพื่อทำให้อุปกรณ์นั้นเชื่อมต่อกับ NETPIE ได้
- **REST API auth** เป็นรหัสที่เรียกต่อกับรหัสลับสำหรับการใช้งานด้วย REST API

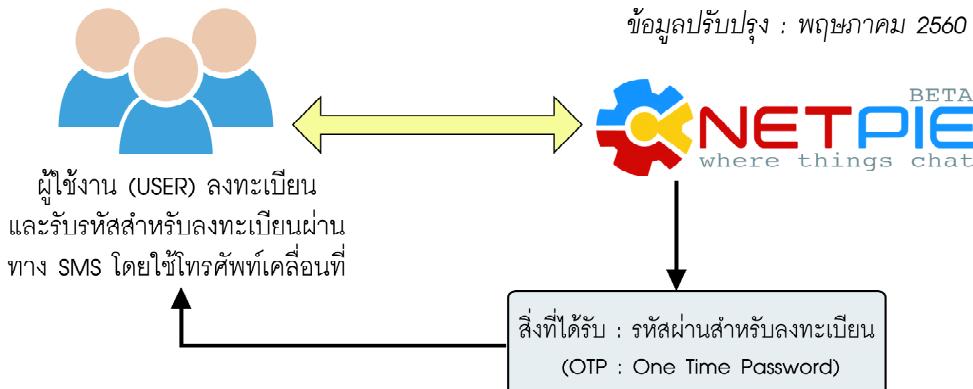


รูปที่ 1-11 รายละเอียดของ Application Key ที่ต้องการตรวจสอบ

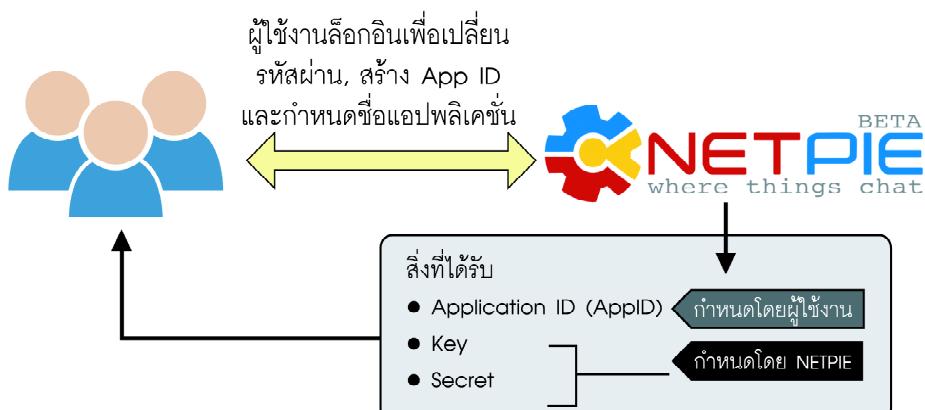
ในรูปที่ 1-12 แสดงภาพรวมของการตั้งค่าบน NETPIE เพื่อนำรหัสต่างๆ ไปใช้ในการติดต่อกับอุปกรณ์ชาร์ดแวร์ที่ต้องการเชื่อมต่อกับ NETPIE

## ลงทะเบียน

ข้อมูลบัญชี : พฤษภาคม 2560



## สร้าง App ID รับรหัส Key และ Secret



- ผู้ใช้งานที่ลงทะเบียนไว้ จะได้รับ App ID ที่มีหน่วยใช้งาน 100 เครดิต (credit) ต่อหมายเลขโทรศัพท์เคลื่อนที่ที่ใช้ลงทะเบียน
- ในการบริการปกติ 1 อุปกรณ์จะใช้หน่วยใช้งาน 1 เครดิต (ยกเว้น บริการ Feed จะต้องใช้ 4 เครดิตต่อ 1 อุปกรณ์) ดังนั้น 1 App ID จึงรองรับอุปกรณ์ได้สูงสุด 100 อุปกรณ์ หากไม่เลือกใช้บริการ Feed โดยไม่มีค่าใช้จ่าย หากต้องการมากกว่านี้ ถือว่าต้องการเป็นผู้ใช้งานในระดับ Enterprise ต้องติดต่อกับสาขาวิชา โดยตรงเพื่อย้ายจำนวน ซึ่งมีค่าใช้จ่าย
- Feed คือ การให้บริการพื้นที่เก็บข้อมูลในช่วงเวลาต่างๆ โดยข้อมูลจะได้รับการเก็บแบบต่อเนื่องสะสมกันไปตลอด ผู้ใช้งานสามารถเรียกอุปกรณ์เมื่อใดก็ได้

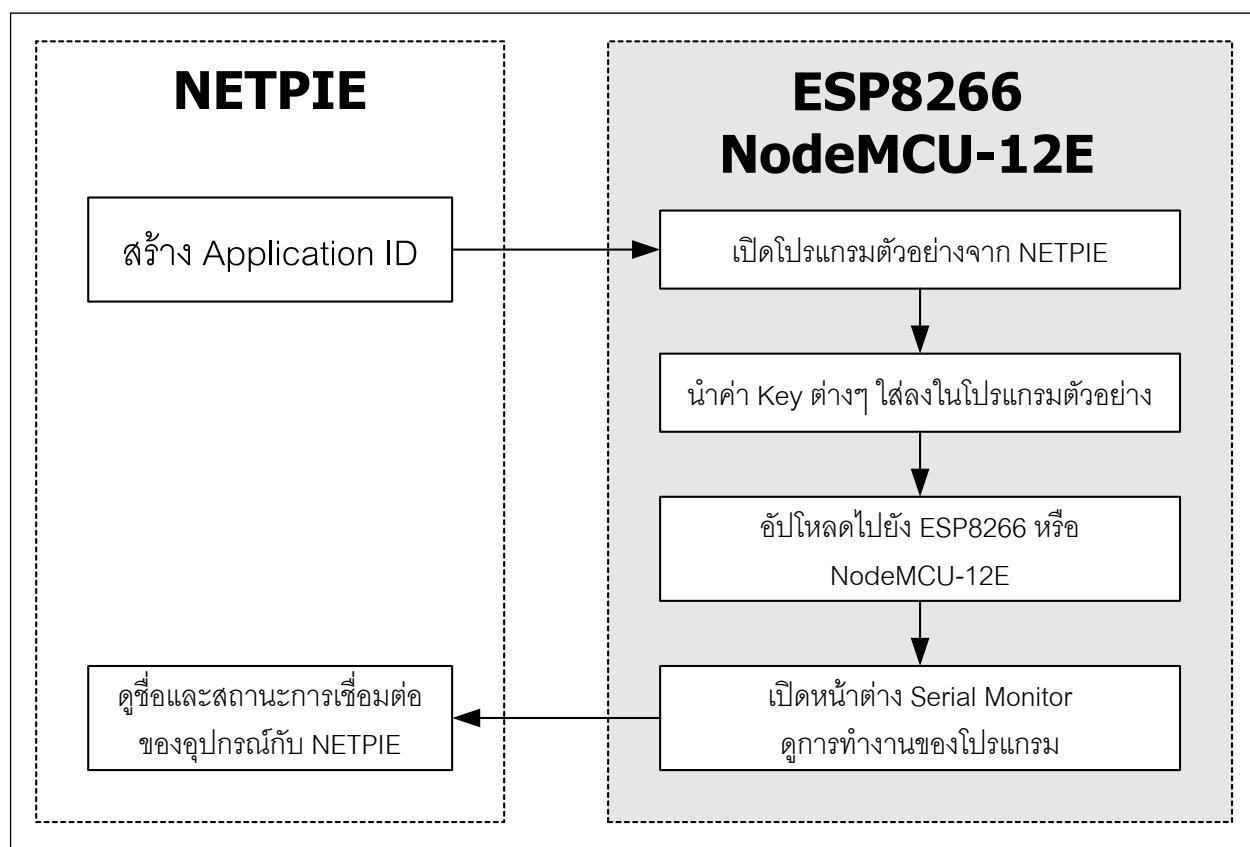
รูปที่ 1-12 ได้จะແກຣມສະບັບນີ້ຕອນກາລົງທະບຽນຈົນຄືກາເຕີຍມກາຮົດທີ່ NETPIE ເພື່ອນຳຮັສສຳຄັງໄປໃຊ້ໃນການຕິດຕໍ່ກັບເອັບພລິເຄີ່ນນັນ NETPIE

## 1.3 ขั้นตอนการใช้ ESP8266 และ NodeMCU-12E กับ NETPIE

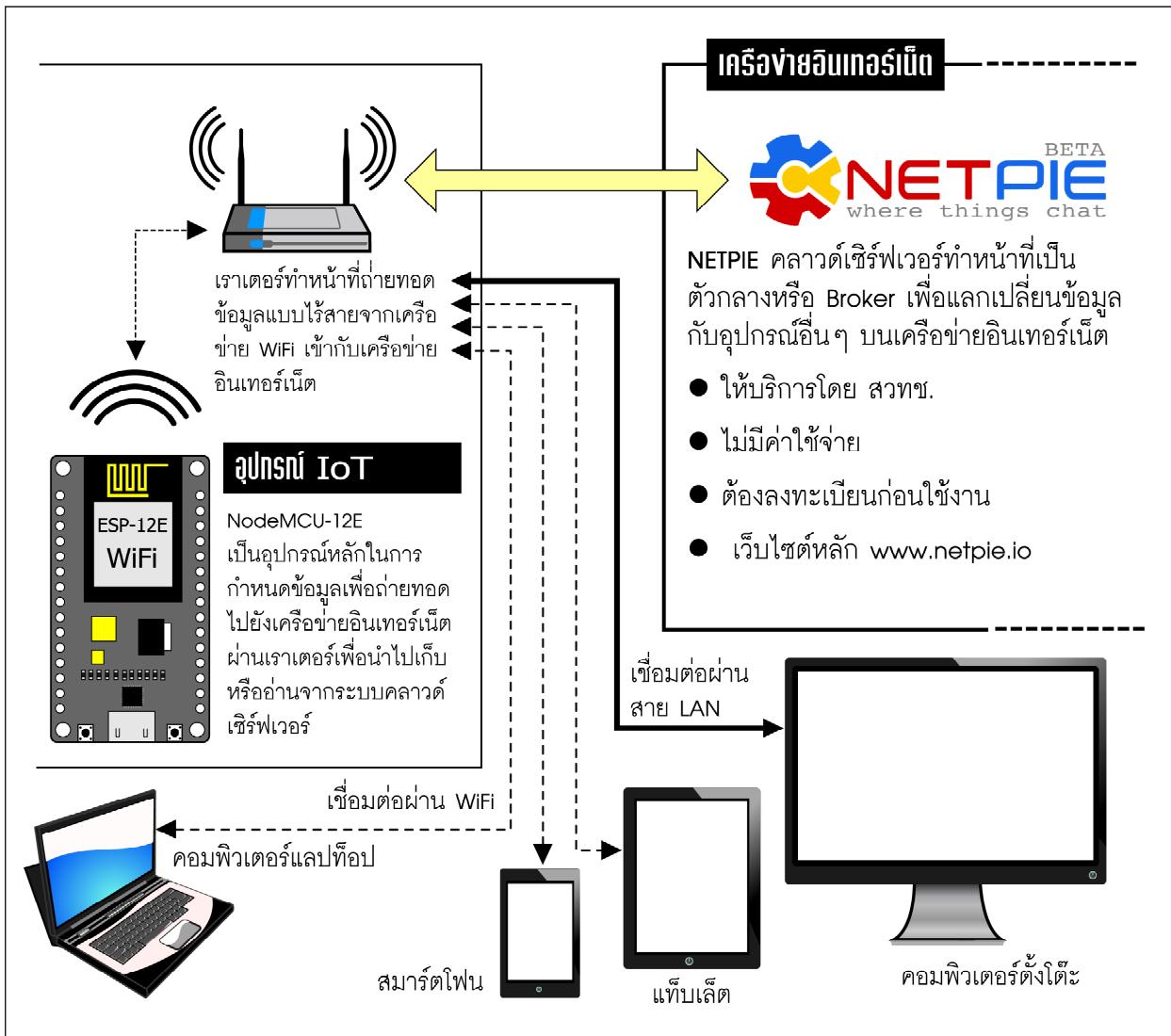
ทาง NETPIE ได้จัดทำไลบรารีที่ใช้งานกับโมดูล WiFi คอนโทรลเลอร์ยอดนิยมและราคาประหยัดอย่าง ESP8266 ที่ใช้ Arduino IDE ในการพัฒนาโปรแกรมไว้แล้ว เพื่อให้ผู้พัฒนาใช้งานได้อย่างสะดวกมากขึ้น โดยดาวน์โหลดไลบรารีได้จาก <https://github.com/netpieio/microgear-esp8266-arduino/archive/master.zip>

เมื่อดาวน์โหลดมาแล้ว แตกไฟล์ เก็บไว้ที่โฟลเดอร์ที่ชื่อว่า **libraries** ของ Arduino IDE ที่ใช้ในการพัฒนา ESP8266 หรือ NodeMCU-12E ในที่นี่ติดตั้ง Arduino1.6.9 ขึ้นไป (ปัจจุบันเป็นเวอร์ชัน 1.8.3) ดังนั้นโฟลเดอร์สำหรับเก็บไลบรารีจะอยู่ที่ C:\Arduino1.6.9\libraries หรือ C:\Arduino18\libraries

ในรูปที่ 1-13 แสดงขั้นตอนการพัฒนาโปรแกรมเพื่อติดต่อและใช้งาน NETPIE ของ NodeMCU-12E หรือโมดูล ESP8266 ในแบบอื่นๆ การอธิบายขั้นตอนการพัฒนาจะอ้างอิงกับรูปที่ 1-13 นี้เป็นหลัก ส่วนการเชื่อมต่อทาง硬件แล้วแสดงดังรูปที่ 1-14



รูปที่ 1-13 ไดอะแกรมแสดงขั้นตอนการการพัฒนาโปรแกรมเพื่อติดต่อและใช้งาน NETPIE ของ NodeMCU-12E หรือโมดูล ESP8266 ในอนุกรมอื่น ๆ



รูปที่ 1-14 ไดอะแกรมแสดงภาพรวมของการเชื่อมต่อ NodeMCU เพื่อใช้งานกับ NETPIE คลาวด์เซิร์ฟเวอร์ร่วมกับอุปกรณ์อื่น ๆ ทั้งคอมพิวเตอร์, สมาร์ตโฟน และแท็บเล็ต

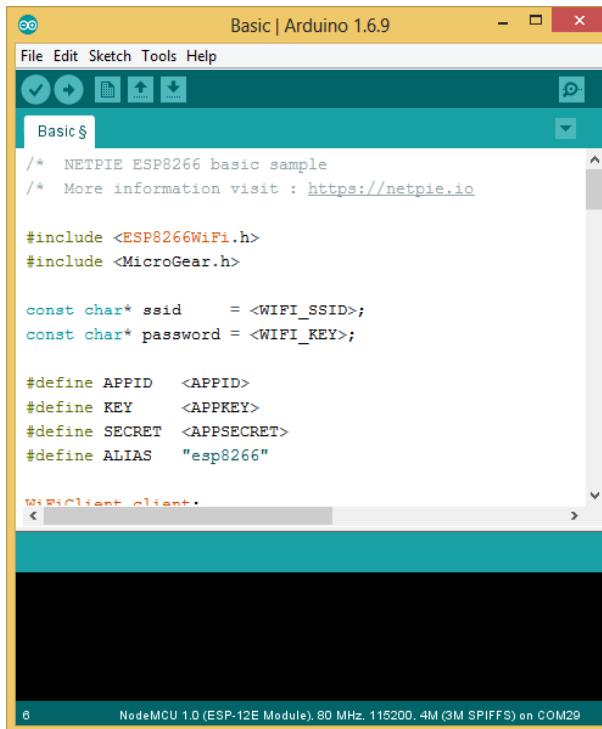
ตัวอย่างขั้นตอนการพัฒนาโปรแกรมบน Arduino IDE สำหรับ NodeMCU-12E มีดังนี้

(1.3.1) เปิดโปรแกรม Arduino IDE (แนะนำใช้วีโอร์ชันที่ INEX ปรับปรุงไว้พร้อมสำหรับการใช้งานกับ ESP8266 และ NodeMCU-12E เปิดไฟล์ตัวอย่าง โดยไปที่ **File>Examples>ESP8266 Microgear >Basic** จะแสดงหน้าต่างของโค้ดดังรูปที่ 1-15 ส่วนรายละเอียดของโค้ดแสดงในโปรแกรมที่ 1-1

(1.3.2) นำรหัสต่างๆ ใส่ลงโปรแกรมที่ 1-1 ดังแสดงหน้าต่างของโค้ดที่ใส่รหัสแล้วในรูปที่ 1-16 โดยมี 2 รหัสที่ต้องกำหนดเองโดยผู้ใช้งาน นั่นคือ

**ssid** คือ ชื่อ WiFi ที่ต้องการเชื่อมต่อ

**password** คือ รหัสผ่านของ WiFi ที่ต้องการเชื่อมต่อ



```

/*
 * NETPIE ESP8266 basic sample
 * More information visit : https://netpie.io
 */

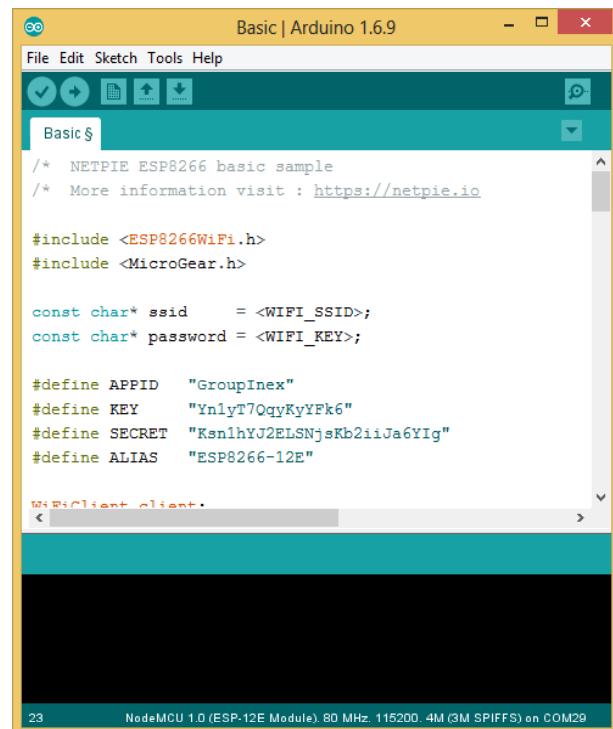
#include <ESP8266WiFi.h>
#include <MicroGear.h>

const char* ssid      = <WIFI_SSID>;
const char* password = <WIFI_KEY>;

#define APPID    <APPID>
#define KEY     <APPKEY>
#define SECRET  <APPSECRET>
#define ALIAS   "esp8266"

WiFiClient client.

```



```

/*
 * NETPIE ESP8266 basic sample
 * More information visit : https://netpie.io
 */

#include <ESP8266WiFi.h>
#include <MicroGear.h>

const char* ssid      = "GroupInex";
const char* password = "Yn1yT7QqyKyYFk6";

#define APPID    "GroupInex"
#define KEY     "Yn1yT7QqyKyYFk6"
#define SECRET  "KsnlhYY2ELSNjsRb2iiJa6YIg"
#define ALIAS   "ESP8266-12E"

WiFiClient client.

```

รูปที่ 1-15 หน้าต่าง Arduino IDE แสดงโค้ดตัวอย่างสำหรับทดสอบการเชื่อมต่อกับ NETPIE ให้ทำการกำหนดรหัสที่เกี่ยวข้องลงในโปรแกรม

รูปที่ 1-16 หน้าต่าง Arduino IDE แสดงโค้ดตัวอย่างสำหรับทดสอบการเชื่อมต่อกับ NETPIE เมื่อกำหนดรหัสคีย์ที่เกี่ยวข้องแล้ว

```

#include <ESP8266WiFi.h>           // ไลบรารีของ ESP8266 สำหรับพัฒนาด้วย Arduino
#include <MicroGear.h>             // ไลบรารี MicroGear ของ NETPIE

const char* ssid = <WIFI_SSID>;    // กำหนดรหัสประจำตัวของเครือข่าย WiFi
const char* password = <WIFI_KEY>; // กำหนดรหัสผ่านของเครือข่าย WiFi

#define APPID <APPID>           // กำหนด AppID ของอุปกรณ์ที่เชื่อมต่อ กับ NETPIE
#define KEY   <APPKEY>          // กำหนดรหัสคีย์ของอุปกรณ์ที่เชื่อมต่อ กับ NETPIE
#define SECRET <APPSECRET>       // กำหนดรหัสลับของอุปกรณ์ที่เชื่อมต่อ กับ NETPIE
#define ALIAS "esp8266"          // กำหนดชื่อของอุปกรณ์ที่เชื่อมต่อ กับ NETPIE

WiFiClient client;
AuthClient *authclient;

int timer = 0;
MicroGear microgear(client);

void onMsghandler(char *topic, uint8_t* msg, unsigned int msglen) // ฟังก์ชันตรวจสอบการประมวลผลข้อความ
{
    Serial.print("Incoming message -> ");
    msg[msglen] = '\0';
    Serial.println((char *)msg);
}

void onFoundgear(char *attribute, uint8_t* msg, unsigned int msglen)
// ฟังก์ชันตรวจสอบอุปกรณ์ที่ต้องการเชื่อมต่อ กับ NETPIE
{
    Serial.print("Found new member -> ");
    for (int i=0; i<msglen; i++)
        Serial.print((char)msg[i]);
    Serial.println();
}

void onLostgear(char *attribute, uint8_t* msg, unsigned int msglen) // ฟังก์ชันตรวจสอบอุปกรณ์หลุดการเชื่อมต่อ กับ NETPIE
{
    Serial.print("Lost member -> ");
    for (int i=0; i<msglen; i++)
        Serial.print((char)msg[i]);
    Serial.println();
}

void onConnected(char *attribute, uint8_t* msg, unsigned int msglen) // ฟังก์ชันตรวจสอบการเชื่อมต่อของอุปกรณ์ กับ NETPIE
{
    Serial.println("Connected to NETPIE...");
    microgear.setName("mygear");
}

void setup()
{ /* Event listener */
    microgear.on(MESSAGE,onMsghandler);
    microgear.on(PRESENT,onFoundgear);
}

```

โปรแกรมที่ 1-1 ไฟล์ **Basic.ino** ตัวอย่างโปรแกรมสำหรับเริ่มต้นเขื่อมต่อ NodeMCU-12E กับ NETPIE (มีต่อ)

```

microgear.on(ABSENT,onLostgear);
microgear.on(CONNECTED,onConnected);
Serial.begin(115200);
Serial.println("Starting...");
if (WiFi.begin(ssid, password))
{
    while (WiFi.status() != WL_CONNECTED) // วนรอการเชื่อมต่อ WiFi
    {
        delay(500);
        Serial.print(".");
    }
}
Serial.println("WiFi connected"); // แสดงข้อความแจ้งการเชื่อมต่อ WiFi สำเร็จ
Serial.println("IP address: ");
// แสดงหมายเลข IP แอดเดรสของเครือข่าย WiFi ที่เชื่อมต่อ
Serial.println(WiFi.localIP());

//microgear.resetToken(); // หากต้องการรีเซ็ตโทเคนให้ปัดเครื่องหมาย //
microgear.init(KEY,SECRET,ALIAS);
microgear.connect(APPID);
}

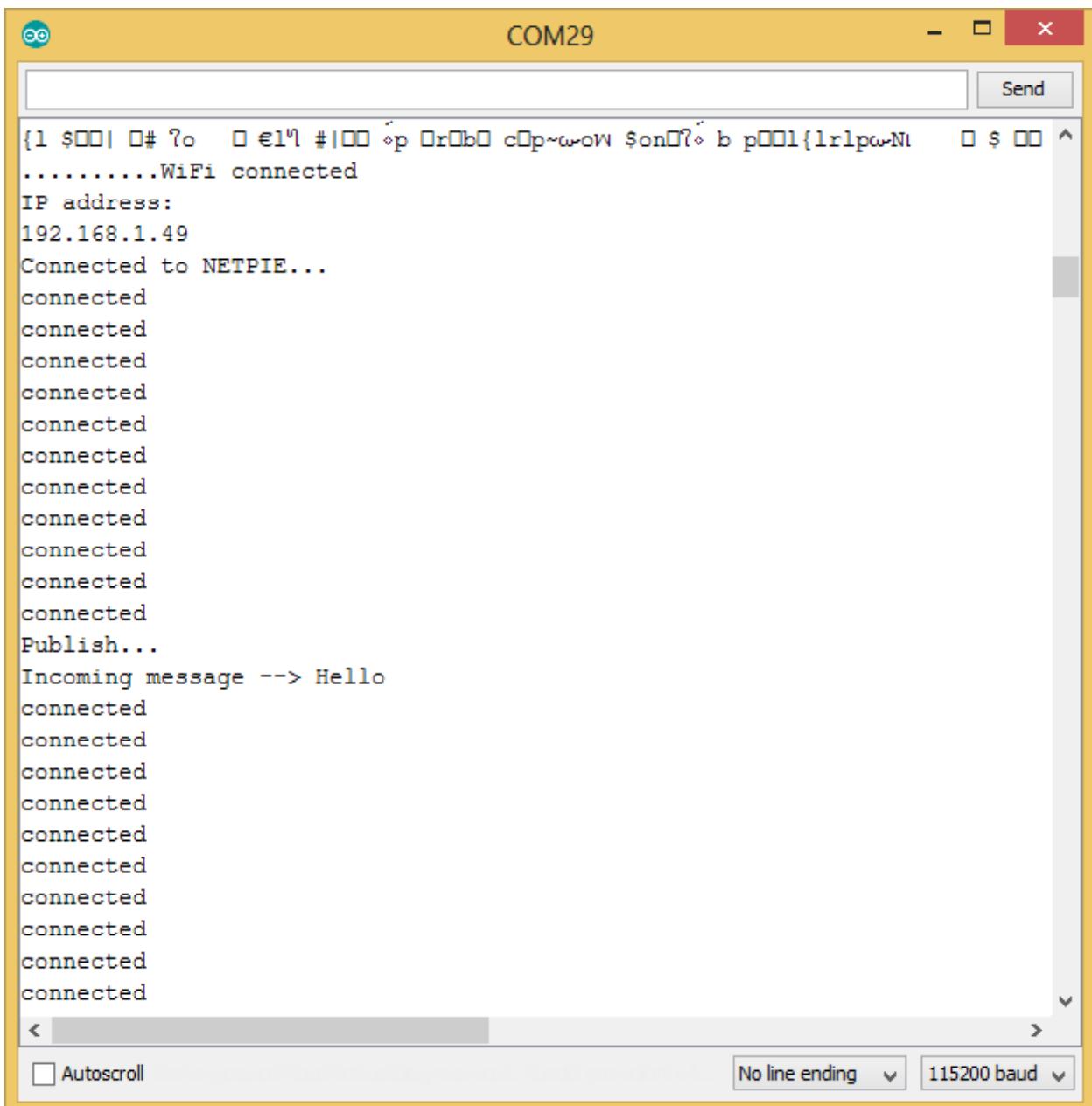
void loop()
{
    if (microgear.connected()) // รอการเชื่อมต่อกับ NETPIE
    {
        Serial.println("connected"); // แสดงข้อความเชื่อมต่อกับ NETPIE สำเร็จ
        microgear.loop();
        if (timer >= 1000)
        {
            Serial.println("Publish..."); // แสดงข้อความแจ้งสถานะการกระจายข้อมูลหรือ Publish
            microgear.chat("mygear","Hello"); // ส่งข้อความไปยัง NETPIE
            timer = 0;
        }
        else timer += 100;
    }
    else
    {
        Serial.println("connection lost, reconnect..."); // แสดงข้อความการเชื่อมต่อกับ NETPIE ไม่สำเร็จ
        if (timer >= 5000)
        {
            microgear.connect(APPID);
            timer = 0;
        }
        else timer += 100;
    }
    delay(100);
}

```

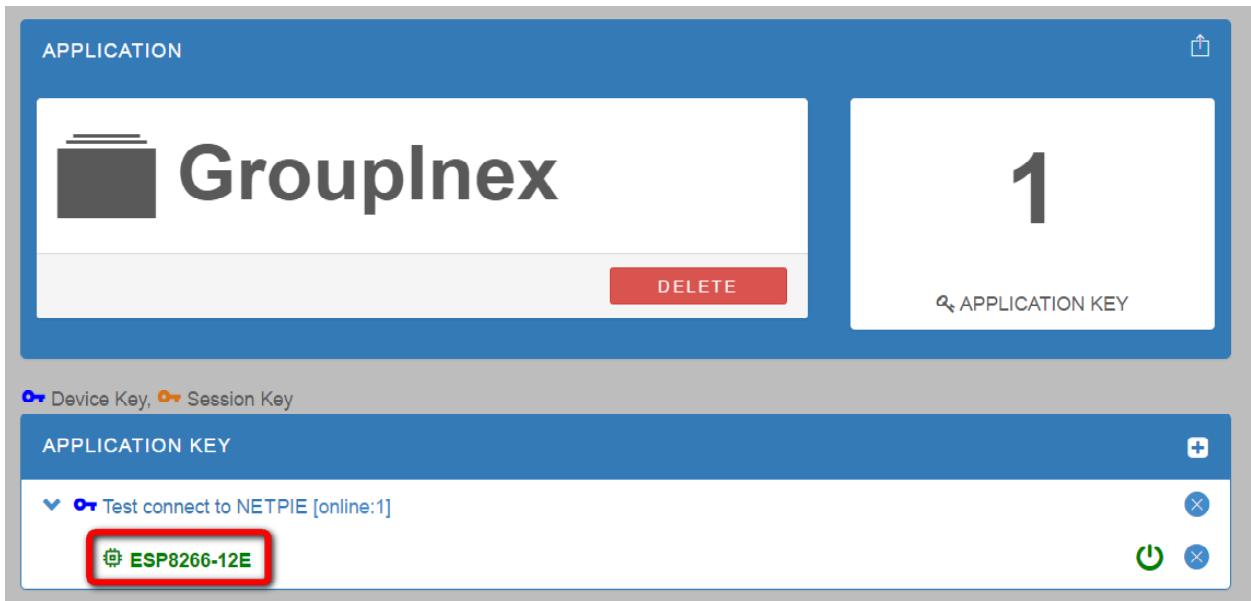
---

**โปรแกรมที่ 1-1 ไฟล์ Basic.ino ตัวอย่างโปรแกรมสำหรับเริ่มต้นเชื่อมต่อ NodeMCU-12E กับ NETPIE (จบ)**

- (1.3.3) เชื่อมต่อ NodeMCU-12E กับพอร์ต USB ของคอมพิวเตอร์
  - (1.3.4) อัปโหลดโค้ดไปยัง NodeMCU-12E ซึ่งในขั้นตอนนี้อาจใช้เวลาพอสมควร
  - (1.3.5) เมื่ออัปโหลดเสร็จแล้ว ทำการเปิดหน้าต่าง Serial Monitor และหน้าเว็บที่สร้าง AppID จากขั้นตอนในหัวข้อ (1.2) เพื่อสังเกตการทำงานของโปรแกรม จะได้ผลการทำงานดังรูปที่ 1-17



รูปที่ 1-17 หน้าต่าง Serial Monitor แสดงการทำงานเมื่อ NodeMCU-12E เชื่อมต่อกับ NETPIE ได้



รูปที่ 1-18 หน้าเว็บของ NETPIE ที่แสดงสถานะการติดต่อระหว่างอุปกรณ์ซึ่งในที่นี่คือ ESP8266 (Node MCU-12E) กับ Application ID – GroupInex

### (1.3.6) การทำงานที่เกิดขึ้นคือ

เมื่อเกิดการการส่งค่าไปยัง NETPIE และดงข้อความ "Publish..."

เมื่อเกิดข้อมูลส่งเข้าไปยัง NETPIE และดงข้อความ Incoming message → Hello หน้าเว็บที่สร้าง AppID และแสดงผลดังรูปที่ 1-18

### (1.3.7) ไปที่หน้าเว็บ APPLICATION ของ AppID ที่ชื่อ GroupInex ดังรูปที่ 1-18

จะเห็นข้อความ **ESP8266-12E** ซึ่งเป็นชื่ออุปกรณ์ที่ตั้งไว้ในโปรแกรมของ NodeMCU-12E หากตัวอักษรเป็นสีเขียว จะหมายถึง อุปกรณ์กำลังออนไลน์หรือเชื่อมต่อกับ NETPIE อยู่

ทั้งหมดที่นำเสนอในบทนี้เป็นการเริ่มต้นใช้งานและทดสอบการทำงานระหว่างอุปกรณ์ สาร์ดแวร์กับคลาวด์เซิร์ฟเวอร์ที่ชื่อ NETPIE จะเห็นได้ว่า มีขั้นตอนไม่มากและไม่ซับซ้อน จึงเหมาะสมอย่างยิ่งสำหรับการเรียนรู้เพื่อนำไปสู่การพัฒนาอุปกรณ์ IoT ด้วยตัวเอง





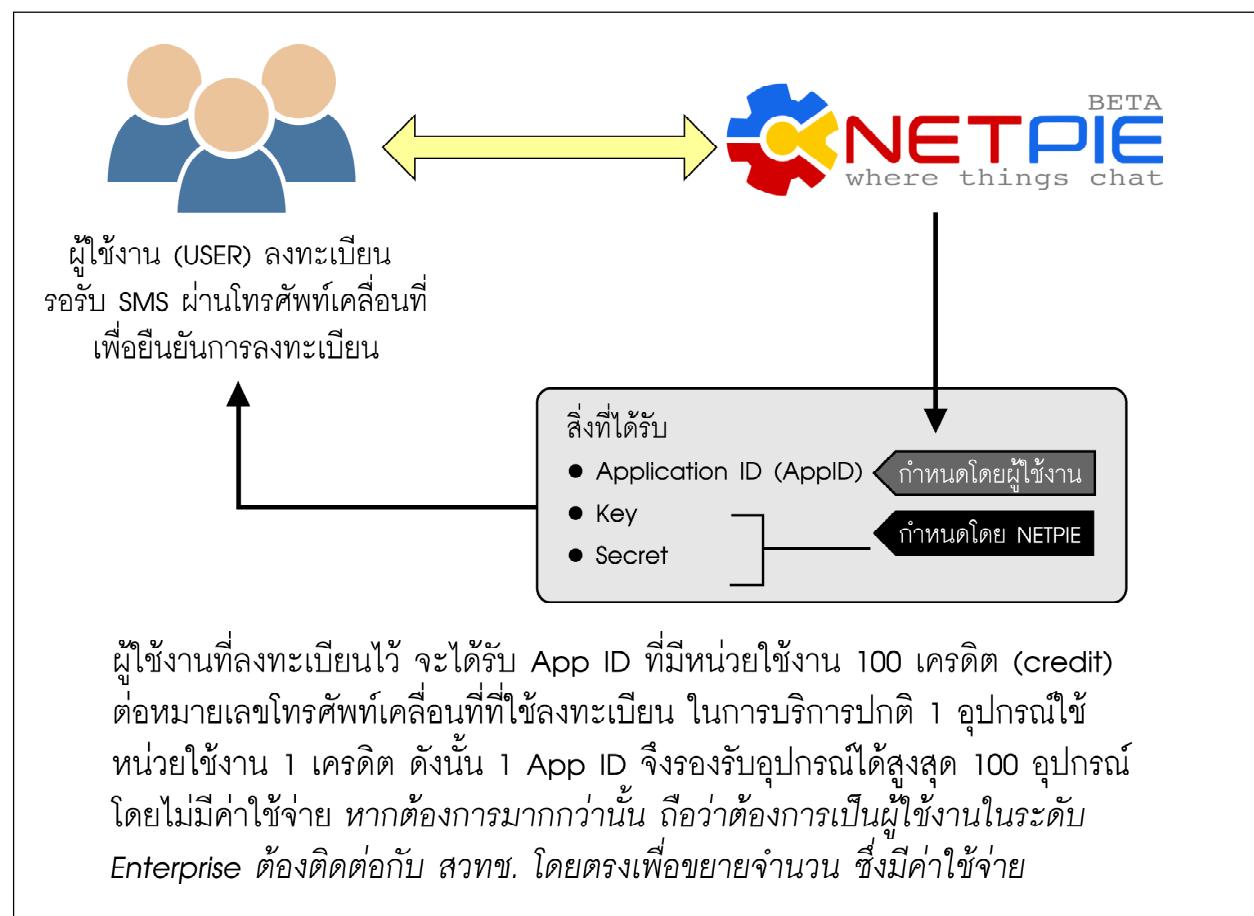
## บทที่ 2

# ไลบรารีของ NETPIE สำหรับการติดต่อแบบ MQTT

ในการเขียนโปรแกรมติดต่อกับ NETPIE ของชาร์ดแวร์ IoT ทุกแพล็ตฟอร์มจะต้องติดตั้งไฟล์ ไลบรารีหลักที่ชื่อว่า **microgear** ในบันนินำเสนอย่างละเอียดกลไกการทำงาน และอธิบายถึงฟังก์ชันต่างๆ ที่สำคัญของไลบรารี **microgear** เพื่อช่วยสร้างความเข้าใจในการเขียนโปรแกรมเพื่อติดต่อกับ NETPIE ผ่านโปรโตคอล MQTT (Message Queuing Telemetry Transport) ได้เพิ่มขึ้น

### 2.1 นิยามที่ควรทราบเบื้องต้น

ในการอธิบายเกี่ยวกับ ไลบรารี **microgear** จะมีคำศัพท์เฉพาะมากพอสมควร ซึ่งจะได้ทำการอธิบายศัพท์เหล่านี้เพิ่มเติมเมื่อมีการกล่าวถึงหรืออ้างอิงถึง สำหรับคำศัพท์ที่ควรทราบในเบื้องต้นมีดังนี้



**รูปที่ 2-1** ไดอะแกรมแสดงความเกี่ยวข้องของส่วนประกอบต่าง ๆ ที่ต้องทราบในการใช้งานไลบรารี **microgear** สำหรับพัฒนาโปรแกรมให้แก่อุปกรณ์เพื่อติดต่อกับ **NETPIE**

1. อุปกรณ์ที่นำมาเชื่อมต่อกับ NETPIE จะถูกเรียกว่า เกียร์ (Gear) โดยในแต่ละแอปพลิเคชันอาจมีอุปกรณ์หรือเกียร์หลายตัว และยังสามารถสื่อสารกันเองภายในได้เลบรหัสประจำตัวของแอปพลิเคชัน (AppID) เดียวกัน โดย NETPIE เปิดให้ในแต่ละ AppID มีหน่วยใช้งานได้สูงสุด 100 เครดิตแบบไม่มีค่าใช้จ่าย และหนึ่งอุปกรณ์ต้องการหน่วยใช้งาน 1 เครดิต ดังนั้นในแต่ละ AppID จึงใช้งานได้สูงสุด 100 อุปกรณ์ โดยไม่มีค่าใช้จ่าย

2. ผู้ใช้งานแต่ละคนที่ทำการลงทะเบียนกับ NETPIE จะต้องสร้าง AppID ขึ้นมาเอง

3. แต่ละอุปกรณ์หรือแต่ละเกียร์จะมีรหัสกุญแจหรือคีย์ (key) เป็นของตัวเอง เพื่อใช้ระบุตัวตนของอุปกรณ์

4. แต่ละคีย์จะมีรหัสอีกชุดหนึ่งเพื่อใช้ในการระบุตัวตนเรียกว่า ซีเคร็ต (Secret) หรืออาจเรียกว่า รหัสลับ

5. Alias หมายถึง ชื่อของอุปกรณ์

6. scope หมายถึง ขอบเขตการทำงานของอุปกรณ์

## 2.2 ฟังก์ชันหลักของไลบรารี microgear สำหรับ ESP8266

ไลบรารี microgear ทำงานได้กับอุปกรณ์หลากหลาย รวมถึงโมดูล WiFi คอนโทรลเลอร์อย่าง ESP8266 ด้วย โดยรุ่นของ ESP8266 ที่รองรับได้แก่ **ESP-01, ESP-07, ESP-12E, ESP-12F, NodeMCU V1 ถึง V3** โดยแท้จริงแล้ว NodeMCU V1 ถึง V3 ล้วนใช้ชิป ESP8266-12E ตัวเดียวกัน ต่างกันที่ชิปแปลงสัญญาณ USB เป็น UART โดย V1 และ V2 จะเหมือนกันทุกประการ ต่างกันเพียงผู้ผลิตนั่นคือใช้ชิปเบอร์ CP2102 ของ Silicon Labs ส่วน V3 ใช้ชิปเบอร์ CH340 อันเป็นชิปเฉพาะที่ผลิตขึ้นในจีน) สำหรับในหนังสือเล่มนี้ hardware NodeMCU-12E ซึ่งเหมือนกับ NodeMCU V2 หรือ NodeMCU Dev Kit 1.0

ดาวน์โหลดไลบรารี microgear พร้อมตัวอย่างเบื้องต้นสำหรับใช้กับ ESP8266 หรือ NodeMCU-12E ได้จาก <https://github.com/netpieio/microgear-esp8266-arduino/archive/master.zip> จากนั้นทำการแตกไฟล์ และเปลี่ยนชื่อไฟล์เดอร์เป็น **microgear-esp8266-arduino** คัดลอกไปยัง **C:\Arduino18\libraries** ในกรณีที่ใช้ Arduino IDE ที่ INEX นำมาปรับปรุง หรืออาจเปลี่ยนแปลงตามเวอร์ชันที่ใช้งาน แนะนำให้ลงกับ Arduino IDE เวอร์ชัน 1.6.9 ขึ้นไป

ฟังก์ชันหลักที่ควรทราบของ microgear เพื่อนำไปใช้ในการเขียนโปรแกรมติดต่อกันมีดังนี้

## 2.2.1 microgear.init

เป็นฟังก์ชันเตรียมการตั้งค่าหรืออินิเชียลเพื่อเริ่มต้นใช้งานไลบรารี microgear

หมายเหตุ :

หากพบเครื่องหมาย [ ] ครอบพารามิเตอร์ของฟังก์ชันไว้ หมายความว่า จะใส่พารามิเตอร์ตัวนั้นหรือไม่ก็ได้  
รูปแบบ

```
microgear.init(char* key,char* secret[,char* alias])
```

พารามิเตอร์

key คือ ชื่อหรือคีย์สำหรับอ้างอิงตัวตนของเกียร์หรืออุปกรณ์

secret คือ รหัสลับหรือ secret ของคีย์ (key) ใช้ประกอบในการยืนยันตัวตน

alias คือ ชื่อของอุปกรณ์

ตัวอย่างที่ 2-1

```
#define KEY      "jKExiZ21odCy0si"
#define SECRET   "3xPhAhuQ3DpTt9J5Wqy3b6B3SAzrar"
#define ALIAS     "ESP8266-12E"
microgear.init(KEY,SECRET,ALIAS);
กำหนดชื่อ KEY แทนคีย์ jKExiZ21odCy0si
กำหนดชื่อ SECRET แทนรหัส 3xPhAhuQ3DpTt9J5Wqy3b6B3SAzrar
```

## 2.2.2 microgear.on()

เนื่องจากการทำงานของ microgear เป็นแบบ event driven จึงต้องตอบสนองต่อเหตุการณ์ต่างๆ ด้วยการเขียน callback function โดยฟังก์ชัน microgear.on() นี้เป็นการกำหนดฟังก์ชันที่ต้องทำงานเมื่อสูญเสียก

รูปแบบ

```
microgear.on(event,callback)
```

พารามิเตอร์

event คือ ชื่อของเหตุการณ์

callback คือ ฟังก์ชันที่ต้องการให้ทำงานเมื่อเกิดเหตุการณ์ที่ระบุไว้ขึ้น

NETPIE มี event หรือเหตุการณ์ที่เกิดขึ้นดังนี้

### (2.2.2.1) เหตุการณ์ CONNECTED

เกิดขึ้นเมื่อ ไลบรารี microgear เชื่อมต่อกับชาร์ดแวร์สำเร็จ

#### ตัวอย่างที่ 2-2

```
microgear.on (CONNECTED, onConnected);
```

ฟังก์ชัน onConnected จะทำงานหากเกิดเหตุการณ์ไลบรารี microgear เชื่อมต่อกับชาร์ดแวร์สำเร็จหรือ CONNECTED

### (2.2.2.2) เหตุการณ์ MESSAGE

เหตุการณ์นี้จะเกิดขึ้นเมื่อมีข้อความเข้ามา พร้อมกับส่งผ่านข้อมูลเกี่ยวกับข้อความหรือ message นั้นๆ มาทางอะกิวเมนต์ของฟังก์ชันที่กำหนดให้ทำงานเมื่อเกิดเหตุการณ์นี้ขึ้น (callback function)

#### ตัวอย่างที่ 2-3

```
microgear.on (MESSAGE, onMsghandler);
```

ฟังก์ชัน onMsghandler จะทำงานเมื่อมีข้อความหรือ message ที่กำหนดปรากฏขึ้นในการติดต่อ

### (2.2.2.3) เหตุการณ์ PRESENT

เหตุการณ์นี้จะเกิดขึ้นเมื่อมีอุปกรณ์หรือเกียร์ที่อยู่ใน AppID เดียวกันทำการเชื่อมต่อเข้ามาใน NETPIE

#### ตัวอย่างที่ 2-4

```
microgear.on (PRESENT, onFoundgear);
```

ฟังก์ชัน onFoundgear จะทำงานเมื่อมีอุปกรณ์ใน AppID เดียวกันเชื่อมต่อหรือทำการออนไลน์เข้ามาใน NETPIE

### (2.2.2.4) เหตุการณ์ ABSENT

เหตุการณ์นี้จะเกิดขึ้นเมื่ออุปกรณ์หรือเกียร์ที่อยู่ใน AppID เดียวกันหายไปจากการเชื่อมต่อกับ NETPIE หรืออффไลน์ (off line)

#### ตัวอย่างที่ 2-5

```
microgear.on (ABSENT, onLostgear);
```

ฟังก์ชัน onLostgear จะทำงานเมื่อมีอุปกรณ์ใน AppID เดียวกันหายไปจากการเชื่อมต่อหรืออффไลน์ไปจาก NETPIE

นอกจากนี้ การเรียกใช้ callback function กับเหตุการณ์หรือ Event ต่างๆ อาจคูเพิ่มเติมได้ในตัวอย่างอื่นๆ ที่จะได้กล่าวถึงต่อไป

### 2.2.3 microgear.setAlias()

เป็นฟังก์ชันกำหนดชื่อของอุปกรณ์เมื่อเชื่อมต่อกับ NETPIE ได้สำเร็จ

#### รูปแบบ

```
microgear.setAlias(char* alias)
```

#### พารามิเตอร์

alias คือ ชื่ออุปกรณ์ที่กำหนด

#### ตัวอย่างที่ 2-6

```
microgear.setAlias("espChat")
```

กำหนดชื่อของอุปกรณ์ที่เชื่อมต่อกับ NETPIE สำเร็จให้มีชื่อว่า espChat

### 2.2.4 microgear.subscribe()

ตัวอุปกรณ์อาจมีความสนใจในหัวข้อหรือ topic ใดเป็นการเฉพาะ จึงใช้ฟังก์ชัน subscribe() ในการบอกรับข้อมูลหรือ message ของหัวข้อนั้นๆ ได้โดยใช้ควบคู่กับฟังก์ชัน microgear.publish()

#### รูปแบบ

```
microgear.subscribe(char* topic)
```

#### พารามิเตอร์

topic คือ ชื่อหรือหัวข้อที่ต้องการรับข้อมูล

#### ตัวอย่างที่ 2-7

```
microgear.subscribe("/sub2esp")
```

กำหนดให้อุปกรณ์รับข้อมูลจาก sub2esp

### 2.2.5 microgear.unsubscribe()

ยกเลิกการ subscribe หรือยกเลิกการบอกรับข้อมูล

#### รูปแบบ

```
microgear.unsubscribe (char* topic)
```

#### พารามิเตอร์

topic ชื่อของหัวข้อหรือ topic ที่ต้องการยกเลิกการบอกรับข้อมูล มีรูปแบบเป็น string

#### ตัวอย่างที่ 2-8

```
microgear.unsubscribe ("/sub2esp")
```

ยกเลิกการบอกรับข้อมูลหรือข้อมูลจาก sub2esp

## 2.2.6 microgear.publish()

เป็นฟังก์ชันที่ใช้ส่งข้อความแบบไม่เจาะจงผู้รับไปยังหัวข้อหรือ topic ที่กำหนด ซึ่งจะมีแต่ อุปกรณ์หรือเกียร์ที่ทำการบอกรับข้อมูลหรือข้อความในหัวข้อเดียวกันนี้เท่านั้นที่จะได้รับข้อความ

### รูปแบบ

```
bool microgear.publish(char* topic, char* message [, bool retained])
bool microgear.publish(char* topic, double topic [, bool retained])
bool microgear.publish(char* topic, double message, int decimal [, 
bool retained])
bool microgear.publish(char* topic, int message [, bool retained])
bool microgear.publish(char* topic, String message [, bool retained])
```

### พารามิเตอร์

topic คือ หัวข้อหรือแหล่งส่งข้อมูล

message คือ ข้อมูลหรือข้อความที่ต้องการส่ง

decimal คือ จำนวนเลขทศนิยมที่ต้องการ

retained เป็นพารามิเตอร์ที่แจ้งให้เก็บรักษาข้อมูลไว้

กำหนดเป็น true เลือกเก็บรักษาข้อมูล

กำหนดเป็น false เลือกไม่ต้องการเก็บข้อมูล

### ตัวอย่างที่ 2-9

```
microgear.publish("/sub2esp ","Hello Inex ")
```

กำหนดให้ส่งข้อความ Hello Inex ไปยังสมาชิกหรืออุปกรณ์ที่บอกรับหัวข้อ sub2esp โดยไม่ต้องการเก็บ รักษาข้อมูลไว้

## 2.2.7 microgear.chat()

เป็นฟังก์ชันส่งข้อมูลทำงานเหมือนการ publish นั้นคือ มีหัวข้อที่กำหนดและข้อความที่ต้องการส่ง

### รูปแบบ

```
bool microgear.chat(char* target, char* message)
bool microgear.chat(char* target, int message)
bool microgear.chat(char* target, double message)
bool microgear.chat(char* target, double, int decimal)
bool microgear.chat(char* target, String message)
```

### พารามิเตอร์

topic คือ หัวข้อหรือแหล่งข้อมูล

message คือ ข้อมูลหรือข้อความที่ต้องการส่ง

decimal คือ จำนวนเลขทศนิยมที่ต้องการ

### ตัวอย่างที่ 2-10

```
microgear.chat("Chat2HTML", "Hello Inex ")
```

กำหนดให้ส่งข้อความ Hello Inex ไปยังスマาร์ทโฟนอุปกรณ์ที่บอกรับหัวข้อ Chat2HTML

## 2.2.8 microgear.connect()

เป็นฟังก์ชันที่ใช้เชื่อมต่อกับ NETPIE โดยต้องระบุชื่อ AppID ที่อุปกรณ์ต้องการเชื่อมต่อ

### รูปแบบ

```
microgear.connect(char* appid)
```

### พารามิเตอร์

appid คือ ชื่อ AppID ที่ต้องการเชื่อมต่อ

### ตัวอย่างที่ 2-11

```
#define APPID "GroupInex"
```

```
microgear.connect(APPID)
```

กำหนด AppID เป็น GroupInex จากนั้นอุปกรณ์จะเชื่อมต่อด้วยฟังก์ชัน microgear.connect(APPID)

## 2.2.9 microgear.connected()

เป็นฟังก์ชันตรวจสอบสถานะการเชื่อมต่อ

### รูปแบบ

```
microgear.connected()
```

### การคืนค่า

true - ในกรณีที่เชื่อมต่อได้

false - ในกรณีที่เชื่อมต่อไม่ได้

## 2.2.10 microgear.loop()

เป็นฟังก์ชันวนตรวจสอบการทำงานของเหตุการณ์ต่างๆ ที่เกิดขึ้นกับโปรแกรมในระหว่างการติดต่อกับ NETPIE

## 2.3 ไลบรารี microgear สำหรับ HTML และ Java Script

microgear-html5 คือ ไลบรารีที่ทำหน้าที่เปลี่ยนเว็บบราวเซอร์ให้เป็นหนึ่งในอุปกรณ์หรือเกียร์เพื่อสื่อสารกับ อุปกรณ์ในแพล็ตฟอร์มอื่น ไม่ว่าจะเป็น Arduino, Raspberry pi หรือคอมพิวเตอร์เพื่อการพัฒนาแอปพลิเคชันสำหรับอุปกรณ์ IoT ผู้พัฒนาระบบสามารถนำไลบรารีนี้ไปพัฒนา IoT คอนโซลหรือโมบายล์แอปพลิเคชันได้ โดยเขียนโปรแกรมด้วยภาษา HTML และ Java Script

ในการทดสอบการรับส่งข้อมูลเพื่อจะได้เห็นภาพได้ชัดเจน การสร้างหน้าจอแสดงผลหรือแดชบอร์ด (Dashboard) เพื่อค่อยตรวจสอบหรือทดสอบการรับส่งข้อมูลจึงเป็นสิ่งจำเป็น ทาง NETPIE จึงได้จัดทำไลบรารีที่ใช้กับโปรแกรมภาษา JavaScript ด้วย ทำให้การออกแบบเว็บเพจที่ทำหน้าที่เป็นหน้าจอแสดงผลหรือแดชบอร์ดเป็นเรื่องง่ายขึ้น เพราะมีฟังก์ชันให้ใช้งานได้สะดวก ง่ายต่อการพัฒนา ฟังก์ชันต่างๆ มีความใกล้เคียงกับ ไลบรารีที่ใช้กับ ESP8266 ศึกษาการใช้ฟังก์ชันต่างๆ ได้ที่ <https://github.com/NETPIEio/microgear-html5> โดยไลบรารีนี้รองรับเว็บบราวเซอร์ยอดนิยมทั้ง Chrome, Firefox, Opera, Safari, Internet Explorer และ Edge

พอร์ตสื่อสารที่ใช้งานจะขึ้นกับโหมดการทำงาน

1. กรณีทำงานในโหมด TLS พอร์ตที่ใช้งานคือ 8081 และ 8084 โดยปกติ microgear จะใช้โหมดนี้เป็นหลัก

2. กรณีทำงานในโหมด Non-TLS พอร์ตที่ใช้งานคือ 8080 และ 8083

ดาวน์โหลด microgear.js จาก :

<https://raw.githubusercontent.com/NETPIEio/microgear-html5/master/microgear.js>

หรือเรียกใช้ตรงจากเว็บ NETPIE.io โดยใช้แท็ก :

```
<script src="https://NETPIE.io/microgear.js"></script>
```

ฟังก์ชันหลักที่ควรทราบของ ไลบรารีนี้ทำการนำเสนอโดยสรุป ดังต่อไปนี้

### 2.3.1 Microgear.create (config)

เป็นฟังก์ชันที่ใช้ในการสร้างอปเจ็กต์ของอุปกรณ์

#### รูปแบบ

```
Microgear.create({
  gearkey: APPKEY,
  gearsecret: APPSECRET,
  gearalias: ALIAS
});
```

#### แอ็ตทริบิวต์ (Attribute)

config เป็น json object ที่มีแอ็ตทริบิวต์ดังนี้

gearkey คือ ชื่อหรือ key ของอุปกรณ์ที่ต้องการให้ทำงาน ใช้อังกฤษตัวตนของอุปกรณ์ รูปแบบเป็น string

gearsecret คือ รหัสลับหรือ secret ของชื่อหรือ key ใช้ในการยืนยันตัวตนของอุปกรณ์ รูปแบบเป็น string

gearalias คือ การระบุชื่อของอุปกรณ์เมื่อเชื่อมต่อได้สำเร็จ

#### ตัวอย่างที่ 2-12

```
const APPKEY = "KEpgmF8opvcLVkT";
const APPSECRET = "42lxBf40Yt7gIwri90PqplfJQp5wnQ";
const ALIAS="HTMLChat";
var microgear = Microgear.create({
  gearkey: APPKEY,
  gearsecret: APPSECRET,
  gearalias: ALIAS
});
```

ในตัวอย่างที่ 2-13 อปเจ็กต์ที่มารองรับคือ **microgear**

### 2.3.2 microgear.connect

เป็นฟังก์ชันที่ใช้เชื่อมต่อกับ NETPIE

#### รูปแบบ

```
void microgear.connect (AppID, callback)
```

#### พารามิเตอร์

AppID คือ รหัสประจำตัวของแอปพลิเคชันที่อุปกรณ์จะทำการเชื่อมต่อ รูปแบบข้อมูลเป็น string

#### ตัวอย่างที่ 2-13

```
microgear.connect ("GroupInex");
```

กำหนดให้เชื่อมต่อกับ AppID ที่ชื่อ GroupInex

### 2.3.3 microgear.chat

เป็นฟังก์ชันส่งข้อความไปยังอุปกรณ์หรือเกียร์ที่ต้องการรับข้อความ ฟังก์ชันนี้มีการทำงานเหมือนกับการส่งข้อมูลหรือ publish

#### รูปแบบ

```
void microgear.chat (gearname, message)
```

#### พารามิเตอร์

gearname คือ ชื่อหัวข้อการติดต่อของอุปกรณ์ที่ต้องการจะส่งข้อความไปถึง มีรูปแบบข้อมูลเป็น String

message คือ ข้อความที่ต้องการส่ง มีรูปแบบข้อมูลเป็น String

#### ตัวอย่างที่ 2-14

```
microgear.chat ("Chat2esp", "Hello Inex");
```

กำหนดให้ส่งข้อความ Hello Inex ไปยังหัวข้อชื่อ Chat2esp

ถ้ามีอุปกรณ์ตัวใดกำหนดให้รับข้อความที่ส่งถึงหัวข้อการติดต่อที่ชื่อว่า Chat2esp ก็จะเห็นข้อความ Hello Inex นี้ด้วย โดยฟังก์ชันที่ใช้ในการกำหนดสิทธิ์ของการรับข้อความคือ microgear.setname()

### 2.3.4 microgear.publish

เป็นฟังก์ชันที่ใช้ส่งข้อความแบบไม่เจาะจงผู้รับ ผู้พัฒนาสามารถใช้ฟังก์ชันนี้ในการส่งเผยแพร่หรือ publish ไปยัง topic หรือหัวข้อที่กำหนดได้ ซึ่งจะมีแต่ microgear ที่ subscribe topic หรือบอกรับหัวข้อนี้เท่านั้นที่จะได้รับข้อความ

#### รูปแบบ

```
void microgear.publish (topic, message[, retained])
```

#### พารามิเตอร์

topic คือ ชื่อที่ต้องการจะส่งข้อความไปถึง มีรูปแบบตัวแปรเป็น String

message คือ ข้อความที่ต้องการส่ง มีรูปแบบตัวแปรเป็น String

retained คือ การเลือกเก็บรักษาข้อมูลล่าสุด

true - หากต้องการเก็บรักษาข้อมูลล่าสุด

false - ไม่ต้องการเก็บรักษาข้อมูลล่าสุด

#### ตัวอย่างที่ 2-15

```
microgear.publish ("/Chat2esp", "Hello Inex");
```

### 2.3.5 microgear.subscribe

microgear อาจมีความสนใจใน topic หรือหัวข้อใดเป็นการเฉพาะ ผู้พัฒนาสามารถใช้ฟังก์ชัน subscribe() ในการบอกรับ message หรือข้อมูลของ topic นั้นได้

#### รูปแบบ

```
void microgear.subscribe (topic)
```

#### พารามิเตอร์

topic คือ ชื่อที่ต้องการจะรับข้อมูล มีรูปแบบเป็น string

#### ตัวอย่างที่ 2-16

```
microgear.subscribe ("/Chat2esp " );
```

### 2.3.6 microgear.unsubscribe

เป็นฟังก์ชันยกเลิกการ subscribe หรือการบอกรับข้อมูล

#### รูปแบบ

```
void microgear.unsubscribe (topic)
```

#### พารามิเตอร์

topic ชื่อหัวข้อหรือรายการที่ต้องการยกเลิกการบอกรับข้อมูล มีรูปแบบเป็น string

#### ตัวอย่างที่ 2-17

```
microgear.unsubscribe ("/Chat2esp")
```

### 2.3.7 microgear.resettoken

เป็นฟังก์ชันส่งคำสั่ง revoke token ไปยัง NETPIE เพื่อล้างการเชื่อมต่อของอุปกรณ์ออกจาก NETPIE และลบโทเคน (token : สิ่งที่ใช้แสดงสิทธิ์ในการเข้ามายังต่อ) ออกจากแคช (cache : หน่วยความจำสำหรับเก็บข้อมูลข่าวคราว) อันเป็นการรีเซ็ตการเชื่อมต่อ อุปกรณ์หรือเกียร์ต้องร้องขอการเชื่อมต่อเข้ามาใหม่

#### รูปแบบ

```
void microgear.resettoken (callback)
```

#### พารามิเตอร์

callback คือ ฟังก์ชันที่จะถูกเรียกให้ทำงาน เมื่อการรีเซ็ตโทเคนเสร็จสิ้น

#### ตัวอย่างที่ 2-18

เนื่องจาก resettoken() เป็นฟังก์ชันแบบอะซิงโครนัส หากต้องการเชื่อมต่ออุปกรณ์เข้ามาใหม่ หลังจากได้กระทำฟังก์ชัน resettoken ต้องเพียงโค้ดในลักษณะนี้

```
microgear.resettoken(function(result) {
    microgear.connect(APPID);
});
```

## 2.4 Events - เหตุการณ์ที่เกิดขึ้นในการทำงานของไลบรารี microgear-html5

แอปพลิเคชันที่ทำงานกับไลบรารี microgear จะมีการทำงานในแบบ event driven คือ ทำงานตอบสนองต่อเหตุการณ์หรือ event ที่เกิดขึ้นด้วยการเขียน callback function มารองรับในรูปแบบดังนี้

```
microgear.on
void microgear.on (event, callback)
```

### พารามิเตอร์

event คือ ชื่อเหตุการณ์ รูปแบบข้อมูลเป็น String

callback คือ เป็นฟังก์ชันที่ต้องการให้ทำงานเมื่อเกิดเหตุการณ์นั้นๆ

จนถึงเวอร์ชันปัจจุบันของ NETPIE (เมษายน 2559) กำหนดให้ตอบสนองต่อเหตุการณ์หรือ event รวม 7 กรณีดังนี้

### 2.4.1 เหตุการณ์ connected

เกิดขึ้นเมื่ออุปกรณ์หรือเกียร์เชื่อมต่อกับแพล็ตฟอร์มสำเร็จ

#### ตัวอย่างที่ 2-19

```
microgear.on("connected", function() {
    console.log("connected");
});
```

### 2.4.2 เหตุการณ์ closed

เกิดขึ้นเมื่ออุปกรณ์หรือเกียร์ตัดการเชื่อมต่อกับแพล็ตฟอร์ม

#### ตัวอย่างที่ 2-20

```
microgear.on("closed", function() {
    console.log("closed");
});
```

### 2.4.3 เหตุการณ์ rejected

เป็นเหตุการณ์ที่เกิดขึ้นเมื่ออุปกรณ์หรือเกียร์เชื่อมต่อไม่สำเร็จ เนื่องจากสิทธิ์ในการเชื่อมต่อ หรือ token ถูกปฏิเสธ อาจเป็นเพราะถูกลบล้าง (revoke) หรือถูกปิด (disable)

#### ตัวอย่างที่ 2-21

```
microgear.on("rejected", function(info) {
    console.log("Connection rejected: "+info);
});
```

## 2.4.4 เหตุการณ์ error

เกิดขึ้นเมื่อมีความผิดพลาดเกิดขึ้นภายในอุปกรณ์หรือเกียร์ที่ทำการเชื่อมต่ออยู่

### ตัวอย่างที่ 2-22

```
microgear.on("error", function(err) {
    console.log("Error: "+err);
});
```

## 2.4.5 เหตุการณ์ message

เหตุการณ์นี้เกิดขึ้นเมื่อมีข้อความหรือ message เข้ามา พร้อมกับส่งผ่านข้อมูลที่เกี่ยวกับข้อความ หรือ message นั้นมาทางอะกิวเมนต์ของฟังก์ชันที่ถูกกำหนดให้ทำงาน

### ตัวอย่างที่ 2-23

```
microgear.on("message", function(topic,msg) {
    console.log("Incoming message: "+msg);
});
```

## 2.4.6 เหตุการณ์ present

เหตุการณ์นี้เกิดขึ้นเมื่ออุปกรณ์หรือเกียร์ภายใต้รหัสแอปพลิเคชันหรือ AppID เดียวกัน ปรากฏตัวขึ้นมาหรือออนไลน์เพื่อเชื่อมต่อ กับ NETPIE

### ตัวอย่างที่ 2-24

```
microgear.on("present", function(event) {
    console.log("New friend found: "+event.gearkey);
});
```

## 2.4.7 เหตุการณ์ absent

เหตุการณ์นี้จะเกิดขึ้นเมื่ออุปกรณ์หรือเกียร์ภายใต้รหัสแอปพลิเคชันหรือ AppID เดียวกันหายไปจากการเชื่อมต่อกับ NETPIE หรือเกิดการออฟไลน์

### ตัวอย่างที่ 2-25

```
microgear.on("absent", function(event) {
    console.log("Friend lost: "+event.gearkey);
});
```

## 2.5 สรุปการใช้ฟังก์ชันรับ-ส่งข้อมูล

ฟังก์ชัน `microgear.setAlias()` ใช้กำหนดชื่อของอุปกรณ์ โดยใช้กับฟังก์ชัน `microgear.chat()` เพื่อส่งข้อมูลไปยังอุปกรณ์ตัวนั้นๆ

ฟังก์ชัน `microgear.subscribe()` ใช้กำหนดสิทธิ์การเข้าถึงของหัวข้อหรือ topic นั้นๆ ใช้กับฟังก์ชัน `microgear.publish()` สำหรับส่งข้อมูล

อย่างไรก็ตาม หนึ่งอุปกรณ์สามารถใช้ฟังก์ชันทั้งสองอย่างพร้อมกันได้

## 2.6 การใช้งาน JavaScript เนื้องตื้น

เนื่องจาก JavaScript เป็นภาษาที่มีการทำงานบนคอมพิวเตอร์ของผู้ที่เปิดใช้งาน ไม่ใช่ทำงานบนเซิร์ฟเวอร์ จึงทำให้นำมาใช้จัดการในเรื่องการแสดงผล หรือกำหนดรูปแบบ ลูกเล่น ได้อย่างมาก anyak การใช้งาน JavaScript ชุดคำสั่งจะถูกกำหนดไว้ระหว่างคำสั่ง `<script>` และ `</script>`

### 2.6.1 ส่วนประกอบของโปรแกรมภาษา HTML หรือ HTML element

ประกอบด้วย

**แท็กเริ่ม + แอตทริบิวต์ + เนื้อหา + แท็กปิด**

#### ตัวอย่างที่ 2-26

```
<b id="demo">Hello Inex </b>
```

#### คำอธิบาย

- โปรแกรม HTML เริ่มตัวยแท็กเริ่มต้นคือ `<b>` และ `id` คือแอตทริบิวต์ของแท็กนี้ กำหนดชื่อว่า `demo` >
- เนื้อหาของโปรแกรม HTML ในที่นี่คือ `Hello Inex`
- ปิดท้ายด้วย `</b>`
- วัตถุประสงค์ของแท็ก `<b>` คือ กำหนดให้แสดงผลเป็นตัวหนา (bold)

#### ตัวอย่างที่ 2-27

```
<p id="demo">JavaScript can change HTML content </p>
<button type="button" onclick="document.getElementById('demo').innerHTML
= 'Hello Inex!'">
Click Me!</button>
```

#### หมายเหตุ

โปรแกรมในบรรทัด `<button type="button" onclick="document.getElementById('demo').innerHTML = 'Hello Inex!'">` จะต้องพิมพ์ต่อเนื่องบนบรรทัดเดียวกัน เนื่องจากข้อจำกัดของหน้ากระดาษจึงแสดงในตัวอย่างเป็น 2 บรรทัดที่ต่อเนื่องกัน

### คำอธิบาย

เมื่อเปิดหน้าเว็บเพจนี้ขึ้นมาจะมีข้อความ JavaScript can change HTML content แสดงขึ้นมา จะเกิดการเปลี่ยนแปลงข้อความเมื่อกดปุ่ม Click Me !

### ผลลัพธ์

- (1) เมื่อเริ่มเปิดหน้าเว็บเพจ จะได้ผลดังรูปที่ 2-2



- (2) เมื่อคลิกปุ่ม Click Me! ข้อความเดิมก่อนหน้า จะเปลี่ยนเป็นข้อความ Hello Inex! ดังรูปที่ 2-3



## 2.6.2 การเปลี่ยนแปลงเนื้อหาหรือ element ของ HTML

การเปลี่ยนแปลงเนื้อหาในโปรแกรม HTML ใช้คำสั่ง `document.getElementById()` ในการจ้างถึงอัลิเมนต์ใดๆ จะใช้ `id` ของแต่ละ要素ทรีบิวต์เป็นตัวกำหนด และใช้ `.innerHTML` เป็นการเปลี่ยนแปลงเนื้อหาของอัลิเมนต์นั้นๆ

### ตัวอย่างที่ 2-28

```
<script>
function myFunction()
{
    document.getElementById("demo").innerHTML='Hello Inex';
    console.log('Hello Inex');
}
</script>
<p id="demo">JavaScript can change HTML content </p>
<button type="button" onclick="myFunction()">Click Me!</button>
```

### คำอธิบาย

ปุ่มกดมีแอตทริบิวต์ที่ใช้กำหนดเหตุการณ์เมื่อมีการคลิกปุ่มที่ชื่อว่า onclick เมื่อกดการคลิกปุ่มนี้เรียกให้ฟังก์ชัน myFunction ซึ่งภายในฟังก์ชันประกอบด้วยการเปลี่ยนแปลงข้อความและการดีบักเพื่อตรวจสอบการทำงาน ด้วยการใช้ฟังก์ชัน console.log เพื่อแสดงผลการทำงาน

### ผลลัพธ์

- (1) เมื่อเริ่มเปิดหน้าเว็บเจนี้ขึ้นมา ให้กด F12 และเลือกที่หัวข้อ Console จะปรากฏหน้าเว็บดังรูปที่ 2-4



- (2) เมื่อคลิกปุ่ม ข้อความที่แสดงอยู่จะเปลี่ยนเป็น Hello Inex ที่ด้านล่าง เนื่องจากใช้คำสั่ง console.log('Hello Inex') ดังรูปที่ 2-5



## 2.7 ตัวอย่างการติดต่อกับ NETPIE ของ NodeMCU-12E ภายใต้โปรโตคอล MQTT

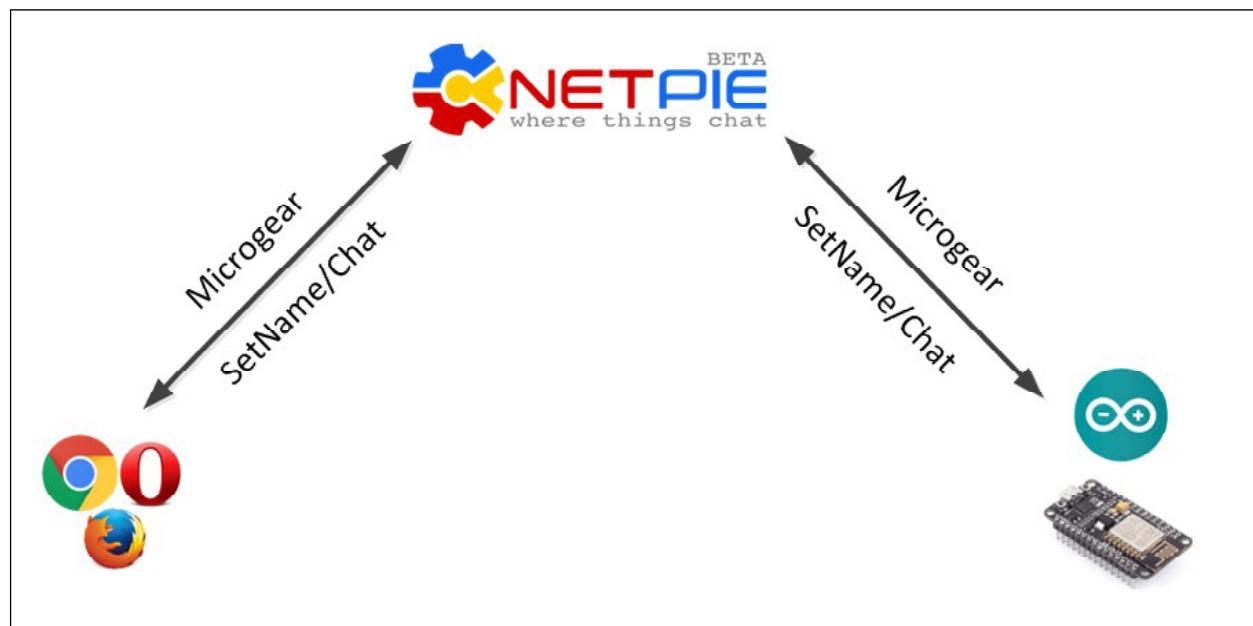
ในหัวข้อนี้นำเสนอตัวอย่างการติดต่อระหว่างอุปกรณ์ชาร์ดแวร์ซึ่งก็คือ NodeMCU-12E และคอมพิวเตอร์ (ผ่านเว็บбраузอร์) กับ NETPIE โดยมีไกด์โปรแกรมหาทำงานในการพร้อมแสดงดังรูปที่ 2-6

จากรูปที่ 2-6 การรับส่งข้อความจะใช้ฟังก์ชัน `microgear.setName()` และ `microgear.chat()` เป็นหลักในการสื่อสารกับ NETPIE โดยฟังก์ชัน `microgear.setName()` ใช้ในการกำหนดลิฟท์เพื่อเข้าถึงหัวข้อการติดต่อและฟังก์ชัน `microgear.chat()` ใช้ในการส่งข้อมูลไปยังหัวข้อการติดต่อที่ต้องการ โดยสร้างหน้าเว็บด้วยภาษา HTML และ JavaScript เพื่อแสดงผลการรับส่งข้อมูล

### 2.7.1 พัฒนาโปรแกรมบนฝั่ง NodeMCU

เปิดโปรแกรม Arduino IDE 1.6.5r5 ที่มีการผนวกชาร์ดแวร์ NodeMCU-12E และทำการติดตั้งไลบรารี `microgear` สำหรับติดต่อกับ NETPIE ไว้แล้ว จากนั้นพิมพ์โปรแกรมที่ 2-1

จากนั้นทำการบันทึกไฟล์ในชื่อ **NETPIEChat.ino** และอัปโหลดโค้ดไปยังแพรวงจร NodeMCU-12E โปรแกรมจะทำงานทันทีแบบอัตโนมัติหลังจากอัปโหลดโค้ดเสร็จ หากต้องการคูณผลการทำงาน เปิดคู่ได้จากหน้าต่าง Serial Monitor ของ Arduino IDE โดยเลือกอัตราบอตเป็น 115,200 บิตต่อวินาที



รูปที่ 2-6 กระบวนการรับ-ส่งข้อความระหว่างอุปกรณ์ไปยัง NETPIE

```

/* NETPIE ESP8266 basic sample */  

/* More information visit : https://netpie.io */  
  

#include <MicroGear.h>  

#include <ESP8266WiFi.h>  

const char* ssid = <SSID>  

const char* password = <PASS>  
  

#define APPID      "GroupInex"  

#define KEY        "YnlyT7QqyKyYFk6"  

#define SECRET     "Ksn1hYJ2ELSNjsKb2iiJa6YIg"  

#define ALIAS      "ESPChat"  
  

WiFiClient client;  

int count = 0;  

int timer = 0;  

MicroGear microgear(client);  

/* If a new message arrives, do this */  

void onMsghandler(char *topic, uint8_t* msg, unsigned int msglen)  

{  

    msg[msglen] = '\0';  

    Serial.print(String("Topic-> ") + topic);  

    Serial.println(String(",Massage-> ") + (char *)msg);  

}  

/* When a microgear is connected, do this */  

void onConnected(char *attribute, uint8_t* msg, unsigned int msglen)  

{  

    Serial.println("Connected to NETPIE...");  

}  

void setup()  

{  

    /* Event listener */  

    microgear.on(MESSAGE, onMsghandler);  

    microgear.on(CONNECTED, onConnected);  

    Serial.begin(115200);  

    Serial.println("Starting...");  

    if (WiFi.begin(ssid, password))  

    {  

        while (WiFi.status() != WL_CONNECTED)  

        {  

            delay(500);  

            Serial.print(".");  

        }
    }
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
    microgear.init(KEY, SECRET, ALIAS);
    microgear.connect(APPID);
}

```

โปรแกรมที่ 2-1 ไฟล์ **NETPIEChat.ino** โปรแกรมภาษา C/C++ ที่อัปโหลดไปยัง NodeMCU-12E เพื่อติดต่อกับ NETPIE ผ่านเครือข่าย WiFi (มีต่อ)

```

void loop()
{
    if (microgear.connected())
    {
        microgear.loop();
        if (timer >= 1000)
        {
            count++;
            microgear.chat("HTMLChat", count);
            timer = 0;
        }
        else timer += 100;
    }
    else
    {
        Serial.println("connection lost, reconnect...");
        if (timer >= 5000)
        {
            microgear.connect(APPID);
            timer = 0;
        }
        else timer += 100;
    }
}

```

### คำอธิบายโปรแกรม

กำหนดให้ NodeMCU มีชื่อว่า **ESPChat** โดยมีตัวแปรที่ชื่อว่า **ALIAS** เป็นตัวกำหนด นั่นหมายความ ว่า ทุกครั้งที่อุปกรณ์ตัวอื่นๆ ที่เข้าฟังก์ชัน **chat()** และมีหัวข้อชื่อ **ESPChat** เป็นหัวข้อในการส่งข้อมูล จะได้ รับข้อความทุกครั้ง พึงก์ชันที่จะทำงานเมื่อเกิดเหตุการณ์นี้คือ **onMsgHandler()** โดยจะแสดงชื่อ Topic และข้อความที่ส่งเข้ามา

ในฟังก์ชัน **loop** ตรวจสอบการเชื่อมต่อกับ Netpie ด้วยคำสั่ง **if(microgear.connected)** ถ้าเชื่อมต่อได้ พึงก์ชัน **microgear.loop()** จะตรวจสอบการทำงานทั้งหมดว่า มีเหตุการณ์ใดเกิดขึ้นบ้าง จากนั้นค่าของตัวแปร **count** จะเพิ่มขึ้นจากเดิมหนึ่งค่า และส่งไปยังอุปกรณ์ที่ชื่อว่า **HTMLChat** ด้วยคำสั่ง **microgear.chat("HTMLChat", count)**

ในกรณีที่เชื่อมต่อ NETPIE ไม่ได้ จะแสดงข้อความแจ้งทางหน้าต่าง Serial Monitor ของ Arduino IDE ด้วยข้อความ **connection lost, reconnect...** และทำการเชื่อมต่อใหม่ด้วยฟังก์ชัน **microgear.connect(APPID)** อีกครั้ง

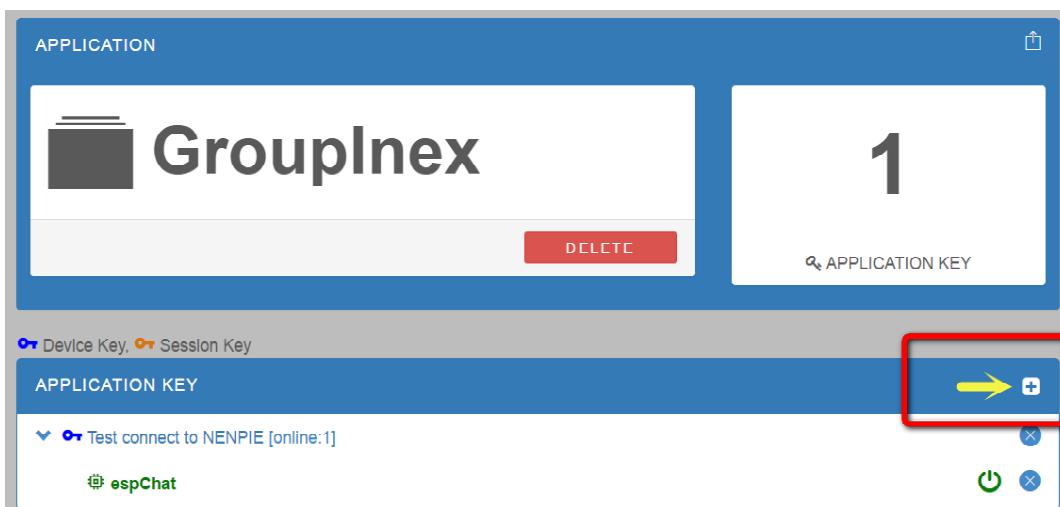
---

**โปรแกรมที่ 2-1 ไฟล์ **NETPIEChat.ino** โปรแกรมภาษา C/C++ ที่อัปโหลดไปยัง NodeMCU-12E เพื่อติดต่อกับ NETPIE ผ่านเครือข่าย WiFi (จบ)**

## 2.7.2 เตรียมการที่สำหรับ NETPIE เพื่อสร้างเว็บเพจสำหรับเป็นแดชบอร์ด

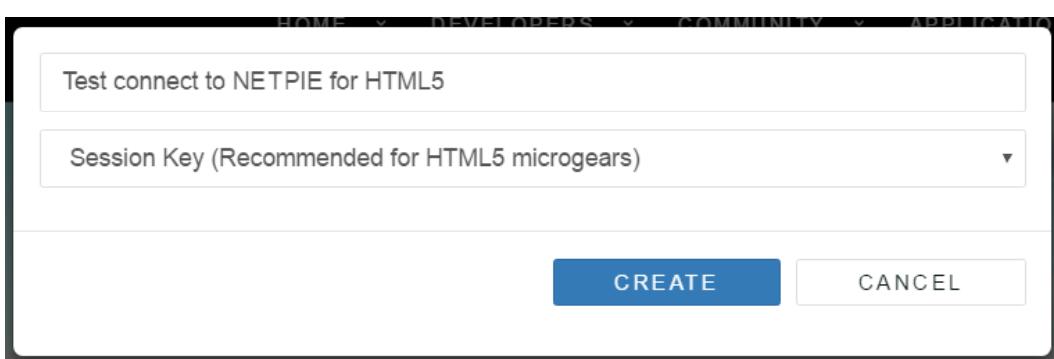
เว็บเพจที่ใช้แสดงผลการทำงานของโปรแกรม NETPIEChat.ino นี้พัฒนาขึ้นจากภาษา HTML ซึ่งจัดว่าเป็นหนึ่งในอุปกรณ์หรือเกียร์ของ NETPIE ดังนั้นจึงต้องมีการสร้างรหัสกุญแจหรือคีย์เพื่อผนวกเข้าไปใน AppID ของผู้พัฒนาด้วย NETPIE ได้ออกแบบรหัสกุญแจหรือคีย์ที่ใช้สำหรับการเปลี่ยนเว็บบราวเซอร์ให้เป็นหนึ่งในอุปกรณ์หรือเกียร์ เรียกว่า **เซสชันคีย์ (Session Key)** เพื่อให้เหมาะสมกับการใช้งานกับ NETPIE มีขั้นตอนการสร้างเซสชันคีย์ดังนี้

(2.7.2.1) ไปยังหน้าเว็บของ NETPIE คลิกที่เมนู APPLICATIONS เข้าไปยังหน้าต่างของ AppID ของผู้พัฒนาซึ่งในที่นี่คือ GroupInex จากนั้นคลิกที่รูปเครื่องหมายบวกดังรูปที่ 2-7



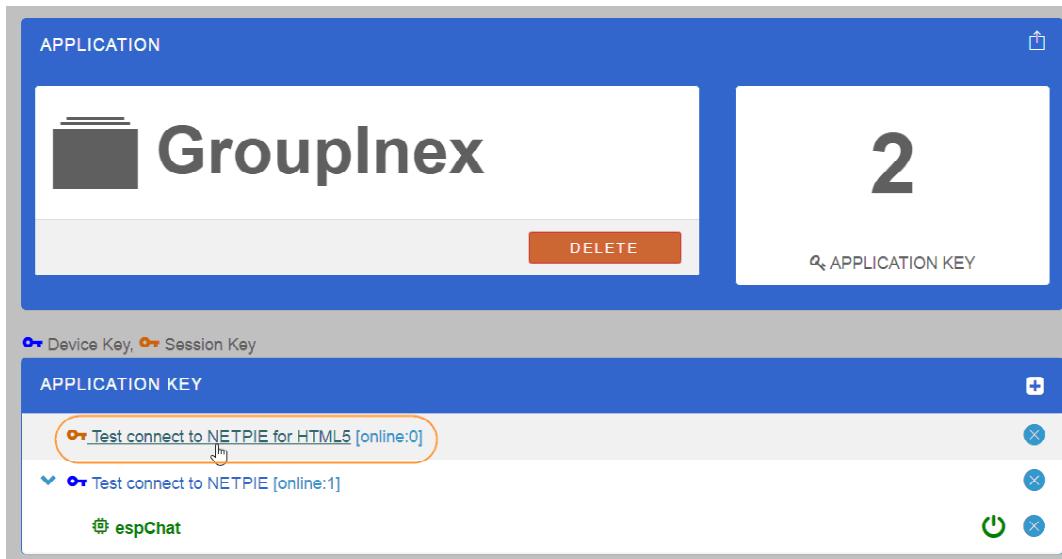
รูปที่ 2-7 หน้าต่าง AppID ของ GroupInex คลิกปุ่ม + เพื่อเพิ่มรหัสกุญแจหรือคีย์ตัวใหม่

(2.7.2.2) ใส่ชื่อหรือรายละเอียดของคีย์ตามต้องการ ในตัวอย่างใช้ข้อความว่า **Test connect to NETPIE for HTML5** และเลือกรูปแบบของคีย์เป็น **Session Key (Recommended for HTML5 microgears)** ดังรูปที่ 2-8 จากนั้นคลิกปุ่ม **CREATE** เพื่อสร้างคีย์



รูปที่ 2-8 กำหนดรายละเอียดของคีย์ที่ต้องการสร้างและรูปแบบของคีย์เป็น **Session Key (Recommended for HTML5 microgears)**

(2.7.2.3) เมื่อสร้างรหัสกุญแจหรือคีย์เสร็จแล้ว คีย์ที่เกิดขึ้นใหม่จะปรากฏในรายการ APPLICATION KEY ดังรูปที่ 2-9 และจำนวนของคีย์จะเพิ่มจาก 1 เป็น 2 จากนั้นให้คลิกที่ชื่อของคีย์เพื่อเข้าไปดูข้อมูลและรหัสต่างๆ

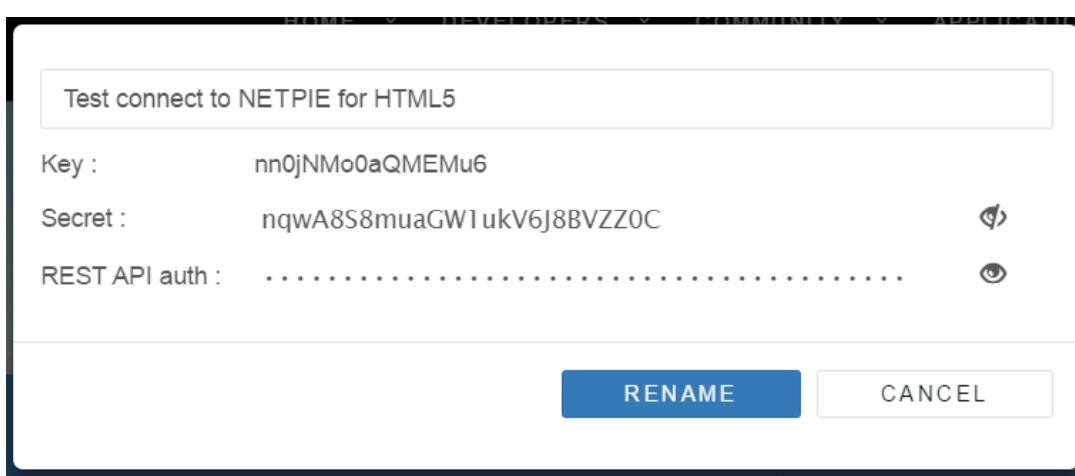


รูปที่ 2-9 คลิกเข้าไปดูรหัสต่าง ๆ ที่ต้องใช้ในการติดต่อกับคีย์ Test connect to NETPIE for HTML5

(2.7.2.4) จะปรากฏหน้าต่างแสดงรหัสต่างๆ ที่ต้องนำไปใช้กับการอ่านแบบเว็บเพจเพื่อใช้เป็นเดชบอร์ด ดังรูปที่ 2-10

สรุปรหัสทั้งหมดที่ต้องใช้มีดังนี้

```
APPID = "GroupInex"
KEY = "nn0jNM00aQMEMu6"
SECRET = "nqwA8S8muaGW1ukV6J8BVZZ0C"
```



รูปที่ 2-10 รหัสที่ต้องใช้ในการเขียนโปรแกรมเพื่อสร้างเดชบอร์ด

(2.7.2.5) จากนั้นทำการเขียนโปรแกรมภาษา HTML เพื่อสร้างเว็บเพจ โดยมีชอร์ตโค้ดแสดงในโปรแกรมที่ 2-2 ตั้งชื่อไฟล์เป็น **ChatDashboard.html** เพื่อทำงานควบคู่กับโปรแกรม **NETPIEChat.ino** (โปรแกรมที่ 2-1) ที่ได้อัปโหลดโค้ดไปยัง NodeMCU-12E ก่อนหน้านี้แล้ว การสร้างไฟล์โปรแกรม ChatDashboard.html ทำได้ด้วยการใช้โปรแกรมเท็กซ์เตอเรอร์ อาทิ Notepad, Notepad ++ หรือ Wordpad พิมพ์โค้ดตามที่แสดงในโปรแกรมที่ 2-2 บันทึกชื่อไฟล์เป็น ChatDashboard.html ไฟล์ของโปรแกรมจะได้รับการติดตั้งลงในคอมพิวเตอร์ เพื่อให้คอมพิวเตอร์ตัวนั้นๆ ติดต่อกับ NETPIE และ NodeMCU-12E โดยอุปกรณ์ทั้งหมดจะทำงานภายใต้รหัสประจำตัวแอปพลิเคชันหรือ AppID ตัวเดียวกัน นั่นคือ **GroupInex** สำหรับผู้พัฒนาที่ต้องแก้ไขโปรแกรมที่บรรทัดซึ่งใช้ประกาศค่า AppID, Key และ Secret ให้ตรงกับทั้งหมดเบียนและสร้างขึ้นจากตัวอย่างการสร้างรหัสในขั้นตอนที่ (2.7.2.4) ทำให้ต้องแก้ไขโปรแกรมเป็นดังนี้ (หากตามมาตลอด และใช้ App ID ชื่อ **GroupInex**)

```
const APPID = "GroupInex";
const KEY = "nn0jNMo0aQMEMu6";
const SECRET = "nqwA8S8muaGW1ukV6J8BVZZ0C";
```

---

```
<script src="https://NETPIE.io/microgear.js"></script>
<script>
    const APPID = "GroupInex";
    const KEY = "nn0jNMo0aQMEMu6";
    const SECRET = "nqwA8S8muaGW1ukV6J8BVZZ0C";
    const ALIAS= "HTMLChat";
    var microgear = Microgear.create({
        gearkey: KEY,
        gearsecret: SECRET,
        gearalias: ALIAS
    });
    microgear.on('message',function(topic,msg) {
        document.getElementById("data").innerHTML = "Count="+msg;
        console.log('Topic='+topic +', message='+msg);
    });
    microgear.on('connected', function() {
        document.getElementById("data").innerHTML = "Now I am connected with NETPIE...";
        console.log('connected with NETPIE...');

    });
    microgear.on('present', function(event) {
        console.log(event);
    });
</script>
```

โปรแกรมที่ 2-2 ไฟล์ **ChatDashboard.html** โปรแกรมสร้างหน้าเว็บหรือเว็บเพจเพื่อแสดงผลการทำงานของ NodeMCU-12E กับ NETPIE โดยใช้งานควบคู่กับโปรแกรมที่ 2-1 ไฟล์ **NETPIEChat.ino** (มีต่อ)

```

microgear.on('absent', function(event) {
  console.log(event);
});
microgear.resettoken(function(result) {
  microgear.connect(APPID);
});
function myFunction() {
  var x = document.getElementById("myText").value;
  microgear.chat("ESPChat",x);
  console.log('message=' + x);
}
</script>

<center>
  <h1><div id="data">_____</div></h1>
  <h1>Put message: <input type="text" id="myText"> <button onclick="myFunction()">
Send </button> <h1>
</center>

```

### คำอธิบายโปรแกรมเพิ่มเติม

เมื่อเปิดเว็บเพจนี้ด้วยเบราว์เซอร์ เช่น Chrome จะทำให้คอมพิวเตอร์ของผู้ใช้งานเสรีมีอนาคตเป็นอุปกรณ์ อีกตัวหนึ่งของ AppID ที่ชื่อ GroupInex เมื่อคุณรันเครื่องต่อไปยัง NETPIE ได้สำเร็จ คุณรันที่ชื่อ HTMLChat จะปรากฏให้เห็นบนหน้า APPLICATION KEY พารามิเตอร์ที่ใช้กำหนดคือ const ALIAS= "HTMLChat" โดยอุปกรณ์ตัวนี้มีสิทธิ์เข้าถึงอุปกรณ์ที่ชื่อว่า Chat2HTML ด้วยคำสั่ง microgear.setname('Chat2HTML') NodeMCU-12E ที่อัปโหลดโดยด้วยไฟล์ NETPIEChat.ino จะใช้ gearname นี้ส่งข้อมูลไปยัง NETPIE ดังนั้นเว็บเพจที่สร้างขึ้นนี้จะทำการท่าน้ำที่แสดงผลข้อมูลที่ NodeMCU-12E ส่งเข้ามา

เมื่อกิດเหตุการณ์ส่งข้อความเข้ามายังอุปกรณ์ ฟังก์ชันที่ถูกกำหนด Events : message เอาไว้จะทำงานทันที พร้อมกับส่งผ่านข้อมูลเกี่ยวกับ Topic และ Message นั้นๆ มาทางพารามิเตอร์ของ callback function แล้วนำข้อความมาแสดงบนหน้าเว็บเพจด้วยคำสั่ง document.getElementById("data").innerHTML = "Count="+msg และทำการดีบักเพื่อตรวจสอบหัวข้อ (Topic) และข้อความ (Message) ที่เข้ามาด้วยฟังก์ชัน console.log('Topic='+topic+', message='+msg)

การส่งข้อมูลจะใช้ฟังก์ชัน microgear.chat ("ESPChat ", x) โดยส่งไปยังอุปกรณ์ที่ชื่อว่า ESPChat

---

**โปรแกรมที่ 2-2 ไฟล์ ChatDashboard.html โปรแกรมสร้างหน้าเว็บหรือเว็บเพจเพื่อแสดงผลการทำงานของ NodeMCU-12E กับ NETPIE โดยใช้งานควบคู่กับโปรแกรมที่ 2-1 ไฟล์ NETPIEChat.ino (จบ)**

### 2.7.3 ทดสอบการทำงาน

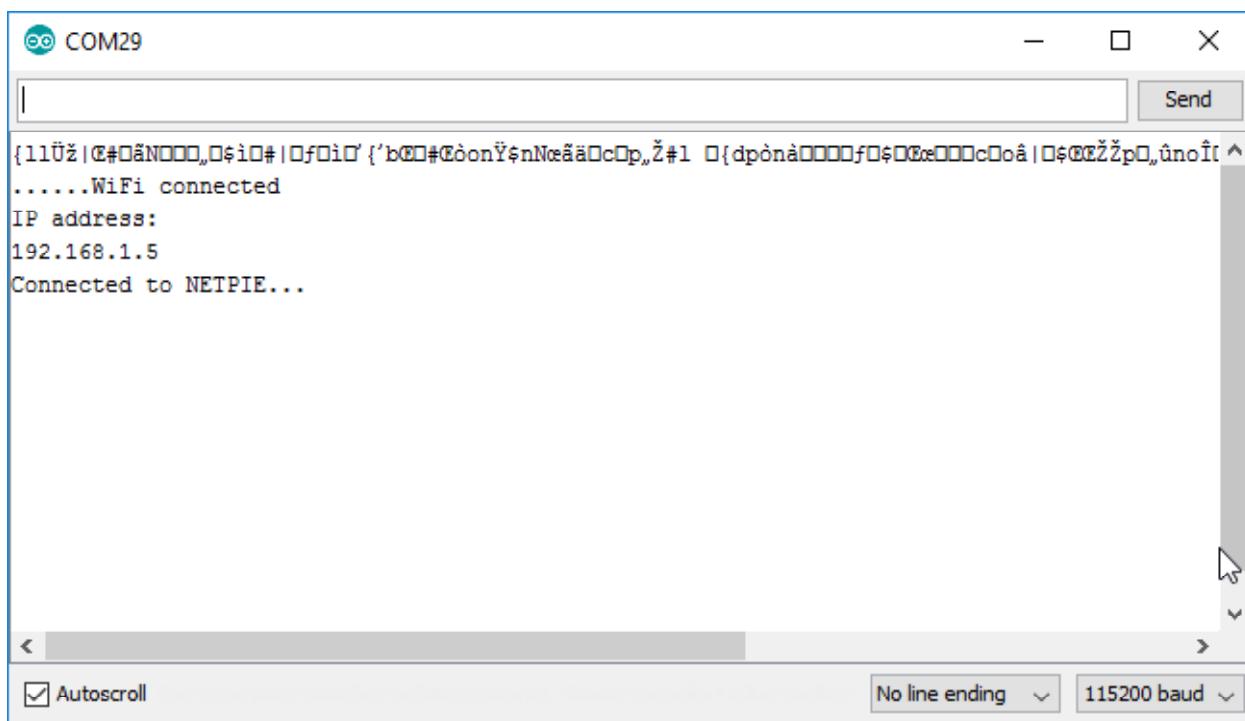
(2.7.3.1) แก้ไขโปรแกรมที่ 2-1 และ 2-2 โดยเปลี่ยน AppID , Key และ Secret เป็นของผู้พัฒนาเอง

(2.7.3.2) อัปโหลดโปรแกรม NETPIEChat.ino ลงบน NodeMCU-12E

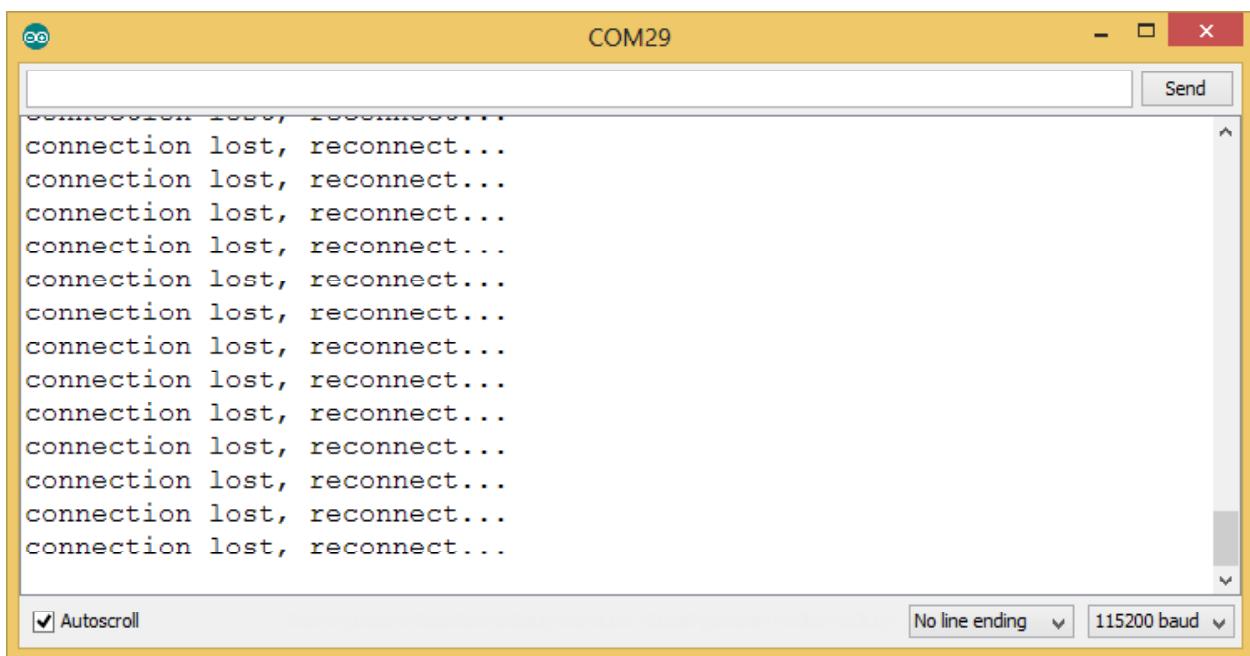
(2.7.3.3) ดำเนินการทดสอบแสดงข้อความบนเว็บเพจที่ส่งมาจาก NodeMCU-12E ที่หน้าต่างของโปรแกรม Arduino IDE ให้คลิกปุ่มเปิดหน้าต่าง Serial Monitor ขึ้นมา

ที่ข้อความสุดท้ายแสดงข้อความ *Connected to NETPIE...* นั่นแสดงว่า NodeMCU-12E เชื่อมต่อ กับ NETPIE ได้สำเร็จ ดังรูปที่ 2-11

แต่ถ้าแสดงข้อความ *connection lost, reconnect...* ดังรูปที่ 2-12 แสดงว่า เชื่อมต่อไม่ได้ ให้ตรวจสอบการเชื่อมต่อทั้งระบบของผู้ใช้งาน และอาจเกิดจากความเร็วของอินเทอร์เน็ตที่ใช้งานต่อไปนี้



รูปที่ 2-11 หน้าต่าง Serial Monitor ของ Arduino IDE และแสดงสถานะการเชื่อมต่อ NodeMCU-12E กับ NETPIE สำเร็จ (ตำแหน่งพอร์ต COM เปลี่ยนแปลงไปตามผู้พัฒนา ในที่นี่เป็น COM29)



รูปที่ 2-12 หน้าต่าง Serial Monitor แสดงสถานะการเชื่อมต่อ NodeMCU-12E กับ NETPIE ไม่สำเร็จ

#### (2.7.3.4) เปิดไฟล์ ChatDashboard.html ด้วยเว็บบราวเซอร์

จะแสดงผลดังรูปที่ 2-13 ในขณะยังไม่ได้เชื่อมต่อ กับ NETPIE

เมื่อเชื่อมต่อกับ NETPIE ได้ หน้าเว็บจะแสดงข้อความ *Now I am connected with NETPIE...* ดังรูปที่ 2-14

#### (2.7.3.5) จากนั้นตรวจสอบที่หน้า Key Management ของตัวเอง

จะปรากฏชื่ออุปกรณ์ที่ได้กำหนดไว้ดังรูปที่ 2-15



รูปที่ 2-13 เว็บเพจ ChatDashboard.html ที่สร้างขึ้น ขณะที่ยังไม่เชื่อมต่อกับ NETPIE



รูปที่ 2-14 เว็บเพจ ChatDashboard.html ที่สร้างขึ้น เมื่อเชื่อมต่อกับ NETPIE สำเร็จ



รูปที่ 2-15 หน้าต่าง Key Management แสดงชื่ออุปกรณ์เมื่อเชื่อมต่อกับ NETPIE ได้สำเร็จ

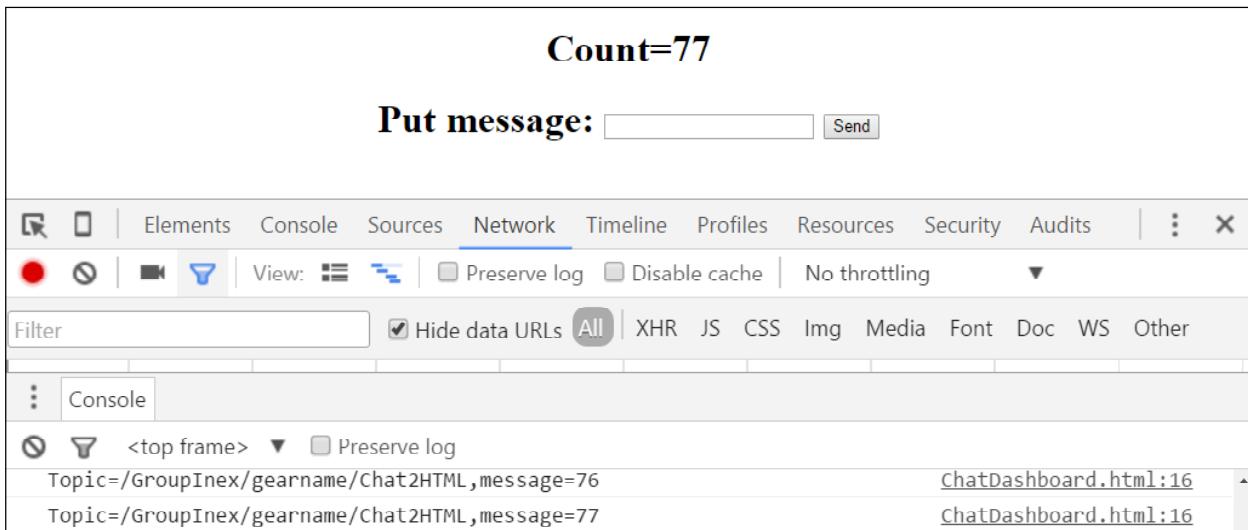
(2.7.3.6) ถ้ามีการใช้ gearname ชื่อ Chat2HTML ในการส่งข้อมูล

หน้าเว็บนี้จะได้รับข้อมูลทั้งหมดดังรูปที่ 2-16 และจำนวนที่เพิ่มขึ้น



รูปที่ 2-16 หน้าเว็บแสดงค่าการนับที่เพิ่มขึ้น เนื่องจากการส่งข้อมูลจาก NodeMCU-12E ผ่าน NETPIE

(2.7.3.7) ถ้าต้องการดีบักเพื่อคุ้มครอง gearname และข้อความ (message) ที่เกิดขึ้น ให้กดคีย์ F12 แล้วเลือกที่หัวข้อ Console จะปรากฏหน้าเว็บดังรูปที่ 2-17



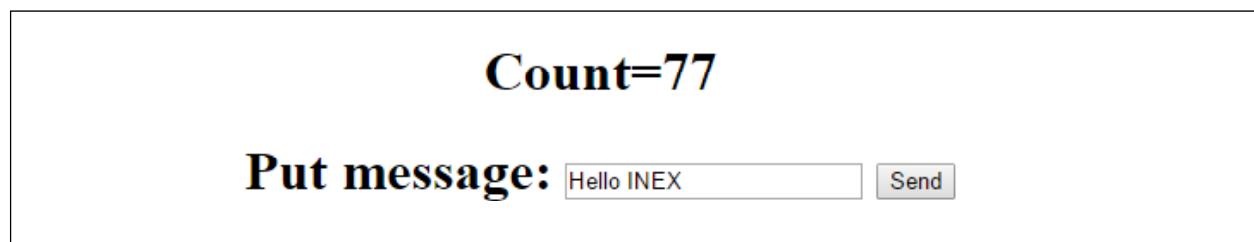
รูปที่ 2-17 หน้าเว็บแสดงชื่อ gearname และข้อความ (message) ที่เกิดขึ้น

## 2.7.4 ทดสอบการแสดงข้อความบน Serial Monitor ที่ส่งมาจาก HTML

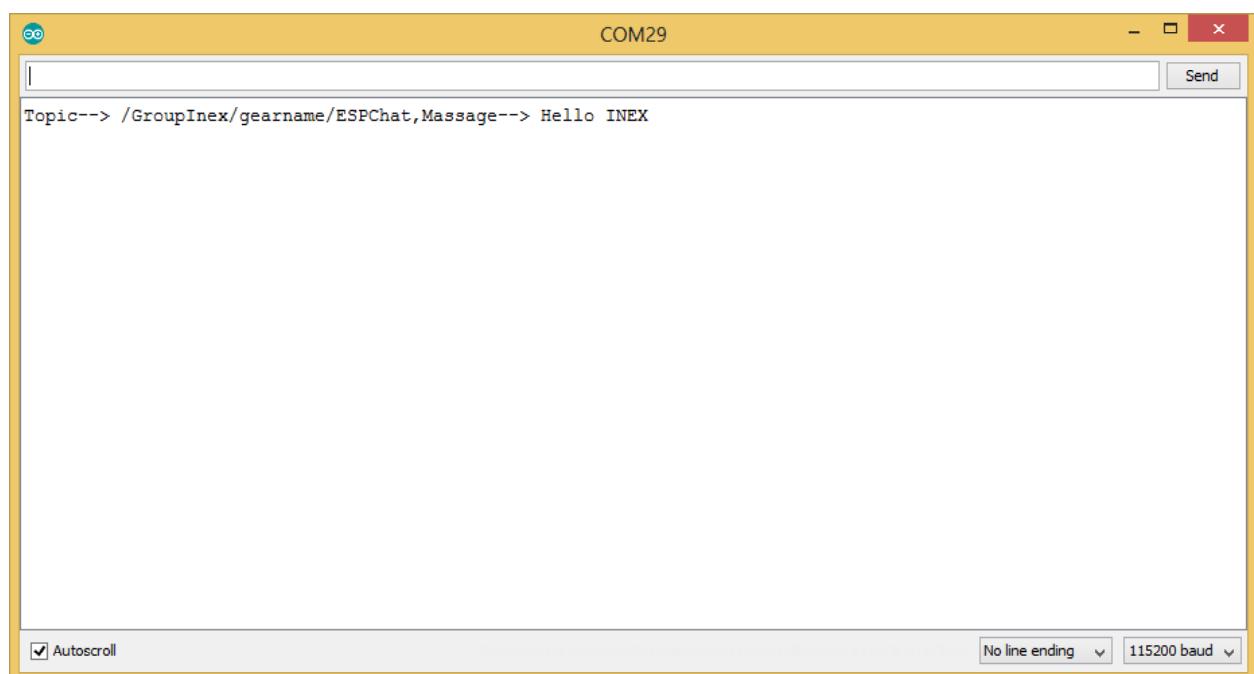
(2.7.4.1) ต่อเนื่องจากการทดสอบในหัวข้อ 2.7.3 ให้เปิดหน้าเว็บและหน้าต่าง Serial Monitor ขึ้นมาพร้อมกัน จากนั้นป้อนข้อความที่ต้องการดังรูปที่ 2-18 แล้วคลิกปุ่ม Send เพื่อส่งข้อมูล

(2.7.4.2) เมื่อคลิกปุ่ม Send เพื่อส่งแล้ว ข้อความนั้นจะมาปรากฏบนหน้าต่าง Serial Monitor ดังรูปที่ 2-19

จากรูปที่ 2-19 ชี้ว่า Topic คือ “/GroupInex/gearname/Chat2esp” จากข้อความนี้ หากมีอุปกรณ์ที่ได้สิทธิ์เข้าถึงข้อมูลมากกว่าหนึ่งหัวข้อ ผู้พัฒนาสามารถนำ Topic มาเป็นเงื่อนไขในการแยกแยะว่า ข้อความไหนเป็นของอุปกรณ์ตัวใดได้ด้วย



รูปที่ 2-18 แสดงการป้อนข้อความเพื่อส่งมายัง NETPIE ก่อนส่งต่อไปแสดงผลที่หน้าต่าง Serial Monitor ของ NodeMCU-12E



รูปที่ 2-19 หน้าต่าง Serial Monitor แสดงการรับข้อความจากเว็บเพจ ChatDashboard.html ที่ส่งผ่าน NETPIE





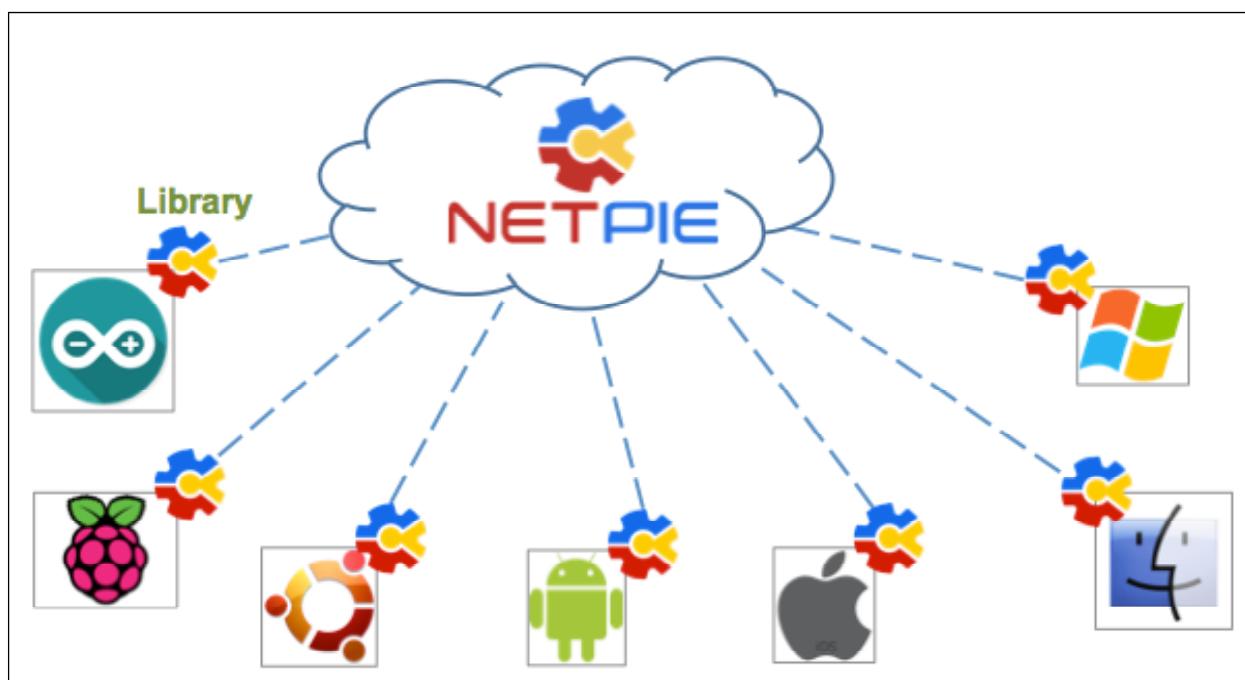
# บทที่ 3

## ความรู้เบื้องต้นเกี่ยวกับ NETPIE REST API

NETPIE ได้จัดเตรียมส่วนต่อประสานโปรแกรมประยุกต์หรือ API (Application Programming Interface) ในรูปแบบที่เรียกว่า **REST API** (REpresentational State Transfer) เพื่อช่วยให้ติดต่อสื่อสารกับ Microgear หรืออุปกรณ์ชนิดอื่นๆ ผ่านทางโปรโตคอล HTTP ที่เข้าถึงได้ง่าย โดยไม่ยึดติดกับภาษาที่ใช้ในการพัฒนาโปรแกรมหรือตัวชาร์ดแวร์ นำไปประยุกต์ใช้กับเว็บเซิร์ฟเวอร์แบบดังเดิมได้ หรือจะเรียกผ่านการเขียนโปรแกรมแบบบรรทัดคำสั่งหรือคอมมานด์ไลน์ รวมไปถึงการเชื่อมต่อกับเว็บเซอร์วิสต่างๆ ก็ทำได้เช่นกัน

### 3.1 อะไรคือ REST API

REST ย่อมาจาก REpresentational State Transfer เป็นรูปแบบสถาปัตยกรรมของระบบโครงข่ายหรือเน็ตเวิร์กที่รองรับการเปลี่ยนสถานะรูปแบบของการแสดงผล โดยใช้ปริมาณข้อมูลที่รับส่งน้อยซึ่งหมายความว่าข้อมูลที่ส่งกลับมายังผู้ใช้จะมีขนาดเล็กและรวดเร็ว ควบคุมขนาดของหน่วยความจำเพื่อให้ขนาดและราคาของอุปกรณ์ IoT ไม่สูงจนเกินไป



รูปที่ 3-1 แนวคิดของ REST API ที่ NETPIE นำมาใช้และพัฒนา เพื่อให้ NETPIE สามารถติดต่อกับแพลตฟอร์มต่างๆ ได้อย่างหลากหลาย

แนวคิดของ REST API กำเนิดขึ้นในปี ค.ศ. 2000 โดย **Ray Fielding** ที่ Univeristy of California Irvine ในสหรัฐอเมริกา Ray พัฒนา REST ขึ้นเพื่อใช้ในเครือข่ายของมหาวิทยาลัยที่มีการเชื่อมต่อของอุปกรณ์และโปรแกรมประยุกต์ที่หลากหลาย ทั้งยังรองรับการทำงานแบบเว็บเปิดหรือ Open Web ด้วยการทำงานของ REST จะเน้นไปที่การบริหารข้อมูลในรูปแบบของทรัพยากร (resource) เป็นหลัก

### 3.1.1 ข้อดีของ REST

1. ดำเนินการตามปรัชญา Open Web ทำให้เกิดโอกาสในการพัฒนาต่ออยอดได้
2. ง่ายต่อการนำไปใช้งานและการดูแลรักษา
3. แยกการทำงานอย่างชัดเจนระหว่าง ไคลเอ็นต์ (เครื่องลูก - ผู้ใช้งาน) กับเซิร์ฟเวอร์
4. การสื่อสารข้อมูลไม่ถูกกำหนดให้มีเพียงเอกสารลักษณ์เดียว จึงยืดหยุ่น และดัดแปลงได้
5. ไคลเอ็นต์สามารถเก็บรักษาข้อมูลไว้ได้ เพื่อป้องกันการรบกวนจากการส่งข้อมูลซ้ำๆ หลายรอบจากแหล่งต่างๆ
6. ให้ผลลัพธ์ของข้อมูลได้หลายรูปแบบ ทั้ง JSON และ XML จึงทำให้นำไปใช้งานกับโปรแกรมประยุกต์ที่พัฒนาด้วยโปรแกรมภาษาต่างๆ ได้ รวมถึงฮาร์ดแวร์ด้วย

### 3.1.2 ข้อเสียของ REST

1. ต้องทำงานที่ระดับสูงสุดของโปรโตคอล HTTP
2. ยากที่จะบังคับให้มีขั้นตอนการอนุมัติในระบบ เพื่อช่วยรักษาความปลอดภัยให้แก่ข้อมูล ทั้งนี้ก็เนื่องมาจากแนวคิดของ Open Web ที่ต้องการให้เกิดการเชื่อมต่อกันได้อย่างสะดวก และไม่มีเงื่อนไข เพื่อให้รองรับกับแพลตฟอร์มได้หลากหลายทั้ง โปรแกรมประยุกต์ที่พัฒนามาจากภาษาคอมพิวเตอร์ที่ต่างกัน และฮาร์ดแวร์ที่มีความหลากหลาย ทั้งสมาร์ตโฟน แท็บเล็ต อุปกรณ์ IoT และคอมพิวเตอร์

### 3.1.3 จะใช้ REST เมื่อใด ?

REST จะมีประโยชน์อย่างมากหากระบบที่ประกอบด้วย ไคลเอ็นต์และเซิร์ฟเวอร์ต้องทำงานผ่านเว็บไซต์ หรือผ่านสภาพแวดล้อมแบบเว็บ (web environment) และตัวไคลเอ็นต์หรือผู้ใช้งานไม่มีความจำเป็นต้องรับข้อมูลตลอดเวลา ซึ่งจะสอดคล้องกับการพัฒนาอุปกรณ์ IoT เพราะในแนวคิดของ IoT อุปกรณ์จะต้องคุยกันเอง สื่อสารข้อมูลกันเองได้ โดยอาศัยคลาวด์เซิร์ฟเวอร์เป็นตัวกลาง ผู้ใช้งานอาจเข้ามายังอนิเตอร์หรือตรวจสอบข้อมูลได้ หรือเข้ามาสั่งการได้เป็นครั้งคราว

NETPIE ถูกพัฒนาขึ้นมาภายใต้แนวคิดที่ให้อุปกรณ์หรือ Things ต้องสามารถติดต่อหรือ “คุย” กัน ได้เองอย่างอัตโนมัติ ข้อมูลที่เกิดขึ้นจะมีเท่าที่จำเป็น เพื่อให้การทำงานรวดเร็ว และเร็วพอต่อความต้องการของผู้พัฒนา

ตัวอย่างการนำ REST API ไปใช้งานที่เห็นเด่นชัดคือ ในสื่อสังคมออนไลน์ (social media), เครือข่ายสังคมออนไลน์ (social network), ระบบสนทนาน่าสนใจ (web chat), บริการในระบบสื่อสารเคลื่อนที่ (mobile service) เป็นต้น ส่วนหน่วยงานที่ใช้ REST API ที่ยกมาเป็นตัวอย่างได้มี Twitter, LinkedIn และ Slack

## 3.2 คีย์ที่ใช้งาน REST API ของ NETPIE

ในการทดสอบและใช้งาน REST API ของ NETPIE จะต้องใช้รหัส 3 ตัวคือ **AppID**, **AppKey** และ **AppSecret** เมื่อมองกับทุกอุปกรณ์ก่อนหน้านี้ที่จะนำมาเชื่อมต่อกับ NETPIE วิธีการสร้างรหัสทั้งหมดจะมีขั้นตอนเหมือนกับในบทที่ 2 เพียงเปลี่ยนชนิดของคีย์เป็น Device เมื่อมองกับการทดสอบในบทที่ 1 อย่างไรก็ตาม ผู้พัฒนาสามารถใช้ Key และ Secret ที่ได้จากการทดสอบในบทที่ 1 ในการใช้งานกับ REST API ได้ ซึ่งก็คือ

```
AppID = GroupIndex
Key= Yn1yT7QqyKyYfk6
Secret= Ksn1hYJ2ELSNjsKb2iiJa6YIg
```

## 3.3 การใช้งานเบื้องต้น

### 3.3.1 การยืนยันตัวตน

NETPIE REST API ช่วยให้นักพัฒนาสามารถส่งหรือเผยแพร่ข้อมูล (publish), บอกรับข้อมูล (subscribe) ผ่านทาง REST API ได้ โดยให้บริการหรือมีอินด์พอยต์อยู่ที่ <https://api.netpie.io> ดังนั้นหากต้องการดำเนินการเรื่องใดกับข้อมูล จะต้องยืนยันสิทธิ์และตัวตนในการเข้าถึงก่อนใช้งานทุกครั้ง ซึ่งมี 2 วิธี

#### 3.3.1.1 ส่งผ่าน HTTP เออดเดอร์แบบ Basic Authentication

เป็นการยืนยันตัวตนแบบพื้นฐานด้วยพารามิเตอร์ 2 ตัว ดังนี้

```
Username : KEY
Password : SECRET
```

#### ตัวอย่างที่ 3-1

การใช้ Basic Authentication ด้วย cURL

```
$ curl -X GET "http://www.domainname.com/resources" -u Key:Secret
```

### 3.3.1.2 ส่งผ่านทาง URL พารามิเตอร์

มีรูปแบบดังนี้

?auth=KEY : SECRET

#### ตัวอย่างที่ 3-2

การใช้ URL พารามิเตอร์ด้วย cURL

```
$ curl -X GET "http://www.domainname.com/resources?auth=Key:Secret"
```

## 3.3.2 องค์ประกอบที่เกี่ยวข้อง

### 3.3.2.1 Topic

Topic หรือหัวข้อการติดต่อ เป็นจุดแลกเปลี่ยนข้อมูลความระหว่าง microgear ลักษณะการเขียนจะอยู่ในรูปของพาร์เซ่น /home/bedroom/temp โดย microgear สามารถ PUT/publish (ส่งหรือเผยแพร่) และ GET/subscribe (รับหรืออกรับ) ไปยัง topic ที่ต้องการได้

### 3.3.2.2 Postbox

เป็นพื้นที่สำหรับเก็บข้อมูลแบบมีลำดับหรือคิว (queue) โดยข้อมูล (message) ที่ถูกส่งเข้าไปใน postbox จะถูกเก็บสะสมไว้ จนกว่าจะมีการอ่านออกໄไป ข้อมูลหรือข้อมูลที่ถูกอ่านแล้วจะหายไปจาก postbox ทันที หมายความว่าจะใช้เป็นเครื่องมือสื่อสารกับ microgear หรืออุปกรณ์ที่ไม่สามารถออนไลน์ หรือเชื่อมต่อกับเครือข่ายอินเทอร์เน็ตได้ตลอดเวลา เช่น PHP script

### 3.3.2.3 Microgear

เป็นอุปกรณ์ที่นำมาเชื่อมต่อกับ NETPIE ผู้ใช้งานสามารถส่งข้อมูลตรงไปยัง microgear ที่ต้องการได้โดยผ่านทาง Alias หรือนามแฝงของอุปกรณ์ตัวนั้นๆ

## 3.4 ตัวอย่างการใช้งาน

### 3.4.1 Topic

**3.4.1.1 เมื่อต้องการเผยแพร่ข้อมูลด้วย HTTP request แบบ PUT มีรูปแบบการใช้งานดังนี้**

**PUT /topic/{AppID}/{topic}**

เป็นการเผยแพร่ข้อมูลหรือข้อความไปยังหัวข้อหรือ topic ของ AppID ที่ระบุ

#### URL parameter

retain สั่งให้เก็บค่าไว้ (เฉพาะค่าล่าสุดเทียบค่าเดียว)

#### Body

เป็นข้อความที่จะส่ง หากต้องการลบค่าที่ retain ไว้ให้ส่งแบบ retain และใช้ body เป็น string เป็น

#### ตัวอย่างที่ 3-3

บน NETPIE มี AppID ชื่อ GroupInex ต้องการใช้ REST API ในการส่งข้อความแบบ retain ว่า “ON” ไปยัง microgear ที่บอกรับ (subscribe) หัวข้อ “GroupInex/gearname/plug” เทียบค่าสั่งด้วย cURL command line ได้ดังนี้

```
$ curl -X PUT "https://api.netpie.io/topic/GroupInex/GroupInex/gearname/plug?retain"-d"ON" -u Yn1yT7QqyKyYFk6:Ksn1hYJ2ELSNjsKb2iiJa6YIg  
โดยที่
```

Yn1yT7QqyKyYFk6 คือ Key ของ AppID ที่ชื่อ GroupInex

Ksn1hYJ2ELSNjsKb2iiJa6YIg คือ Secret ของ AppID ที่ชื่อ GroupInex

#### ผลลัพธ์แบบ JSON

```
{"code":200, "message":"Success"}
```

**3.4.1.2 เมื่อต้องการอ่านหรือรับข้อมูลด้วย HTTP request แบบ GET มีรูปแบบดังนี้**

**GET /topic/{AppID}/{topic}**

เป็นการอ่านข้อความจาก AppID ที่หัวข้อหรือ topic ตามที่ระบุ โดยโคลเล็คต์จะได้รับเฉพาะข้อความล่าสุดที่ถูก retain หรือเก็บไว้ล่าสุดก่อนหน้านี้

#### ตัวอย่างที่ 3-4

```
$ curl -X GET "https://api.netpie.io/topic/GroupInex/GroupInex/gearname/plug" -u Yn1yT7QqyKyYFk6:Ksn1hYJ2ELSNjsKb2iiJa6YIg  
โดยที่
```

Yn1yT7QqyKyYFk6 คือ Key ของ AppID ที่ชื่อ GroupInex

Ksn1hYJ2ELSNjsKb2iiJa6YIg คือ Secret ของ AppID ที่ชื่อ GroupInex

#### ผลลัพธ์แบบ JSON

```
[{"topic":"/GroupInex/gearname/plug","payload":"ON","qos":0,"retain":true}]
```

การอ่านข้อมูลความยังทำได้อีกวิธีหนึ่งคือ ใช้เว็บรวมเซอร์ แต่การยืนยันตัวตน (authentication) จะต้องใช้แบบ URL พารามิเตอร์ ซึ่งวิธีการนี้จะนำไปใช้กับเว็บ Freeboard.io เพื่อสร้างแดชบอร์ด สำหรับการแสดงผลแบบสวยงาม ซึ่งจะได้อธิบายในบทถัดไป

### ตัวอย่างที่ 3-5

`https://api.netpie.io/topic/GroupInex/GroupInex/gearname/plug?auth=Yn1yT7QqyKyYFk6:Ksn1hYJ2ELSNjsKb2iiJa6YI`

โดยที่

`Yn1yT7QqyKyYFk6` คือ Key ของ AppID ที่ใช้กับ GroupInex

`Ksn1hYJ2ELSNjsKb2iiJa6YIg` คือ Secret ของ AppID ที่ใช้กับ GroupInex

### ผลลัพธ์แบบ JSON

`[{"topic":"/GroupInex/gearname/plug","payload":"ON","qos":0,"retain":true}]`

## 3.4.2 Postbox

### 3.4.2.1 เมื่อต้องการเผยแพร่ข้อมูลด้วย HTTP request แบบ PUT มีรูปแบบการใช้งานดังนี้

`PUT /postbox/{AppID}/{postboxname}`

เป็นการส่งข้อมูลไปยัง postbox ที่ใช้กับ postboxname ของ AppID ที่ใช้งาน

### URL parameter

tag - ผู้ส่งสามารถติดแท็กหรือระบุสิ่งที่สนใจให้ข้อมูลได้ เพื่อความสะดวกในการเลือกอ่านเฉพาะ ข้อมูลที่สนใจ

### Body

ข้อมูลที่จะส่งเป็น plain text string หากมีการเข้ารหัสด้วยรูปแบบ json ปลายทางจะต้องนำ string หรือสายอักขระไป parse หรือวิเคราะห์เอง

### ตัวอย่างที่ 3-6

```
$ curl -X PUT "https://api.netpie.io/postbox/GroupInex/webbox?tag=error" -d "ON" -u Yn1yT7QqyKyYFk6: Ksn1hYJ2ELSNjsKb2iiJa6YIg
```

โดยที่

`Yn1yT7QqyKyYFk6` คือ Key ของ AppID ที่ใช้กับ GroupInex

`Ksn1hYJ2ELSNjsKb2iiJa6YIg` คือ Secret ของ AppID ที่ใช้กับ GroupInex

### ผลลัพธ์แบบ JSON

`{"code":200,"message":"Success"}`

### 3.4.2.2 เมื่อต้องการอ่านหรือรับข้อมูลด้วย HTTP request แบบ GET มีรูปแบบดังนี้

**GET /postbox/{AppID}/{postboxname}**

อ่านข้อมูลที่อยู่ใน postbox ของ AppID ที่กำหนด โดยเรียงตามลำดับเวลา ข้อมูลที่เข้ามาถูกอ่านก่อนจะถูกอ่านก่อน

#### URL parameter

tag - ไม่จำเป็นต้องระบุ แต่หากระบุ จะเป็นการเจาะจงอ่านเฉพาะข้อมูลที่ติดแท็กเท่านั้น

#### ตัวอย่างที่ 3-7

```
$ curl -X GET "https://api.netpie.io/postbox/GroupInex/webbox?tag=error" -u YnlyT7QqyKyYFk6:Ksn1hYJ2ELSNjsKb2iiJa6YIg
```

#### ผลลัพธ์แบบ JSON

```
{"msg": "ON", "ts": 1455180348, "tag": "error"}
```

ทั้งหมดที่นำเสนอในบทนี้อาจมีความซับซ้อนและใช้คำศัพท์ทาง โปรแกรมมิ่ง (programming) และเว็บแอป พลิกาชัน (web application) หากพอดีควร นั่นเป็นความจำเป็นของนักพัฒนาอุปกรณ์ IoT ที่ต้องมีความรู้พอสมควรเกี่ยวกับการพัฒนาเว็บแอปพลิกาชัน และยังมีต้องมาทำงานกับคลาวเตอร์ฟิเวอร์ อย่าง NETPIE แล้ว ผู้พัฒนาอาจจะต้องมีความรู้ด้านการเขียน โปรแกรมและพัฒนาแอปพลิกาชันบนเว็บ เพื่อให้การพัฒนาอุปกรณ์ IoT เกิดขึ้นและล้มฤทธิ์ผลตามที่ตั้งใจ





## บทที่ 4

# การประยุกต์ใช้งาน NETPIE ร่วมกับ freeboard.io เพื่อแสดงผลการทำงาน

---

NETPIE คือ คลาวเตอร์ฟิเวอร์ที่ไม่มีส่วนแสดงผลหรือแดชบอร์ดเป็นของตัวเองเหมือนกับ dweet.io หากการทำงานของอุปกรณ์ทั้งหมดไม่ได้มีความจำเป็นต้องมีการแสดงผลใดๆ การทำงานของ NETPIE เพียงพอต่อการบริการแก่ผู้ใช้งาน การทำให้เห็นการแสดงผลของ NETPIE อาจใช้วิธีการสร้างเว็บเพจขึ้นมาดังแนวทางที่แนะนำในบทที่ 2 ของหนังสือเล่มนี้ ซึ่งจะสร้างให้มีความสวยงามหรือดูเรียบง่ายได้ตามความสามารถในการสร้างเว็บเพจของผู้พัฒนาโปรแกรม

ถ้าหากมีความต้องการให้มีการแสดงผลผ่านเว็บในแบบสวยงามพอสมควร แต่ต้องการลดภาระในการเขียนโปรแกรมเพื่อสร้างเว็บ ทางเลือกที่น่าสนใจคือ การใช้เว็บไซต์ที่ทำหน้าที่เป็นแดชบอร์ดหรือหน้าปัดแสดงผล แล้วเขียนโปรแกรมหรือสคริปต์เพียงเล็กน้อย ก็จะได้แดชบอร์ดที่สวยงาม ทางเลือกที่นำมาแนะนำในบทนี้คือ freeboard.io

ข้อมูลเบื้องต้นของ freeboard ขอให้อ่านจากหนังสือเริ่มต้นเรียนรู้และพัฒนาอุปกรณ์ Internet of Things (IoT) กับ NodeMCU-12E ทั้งการลงทะเบียนเพื่อใช้งานและการใช้งานเบื้องต้น

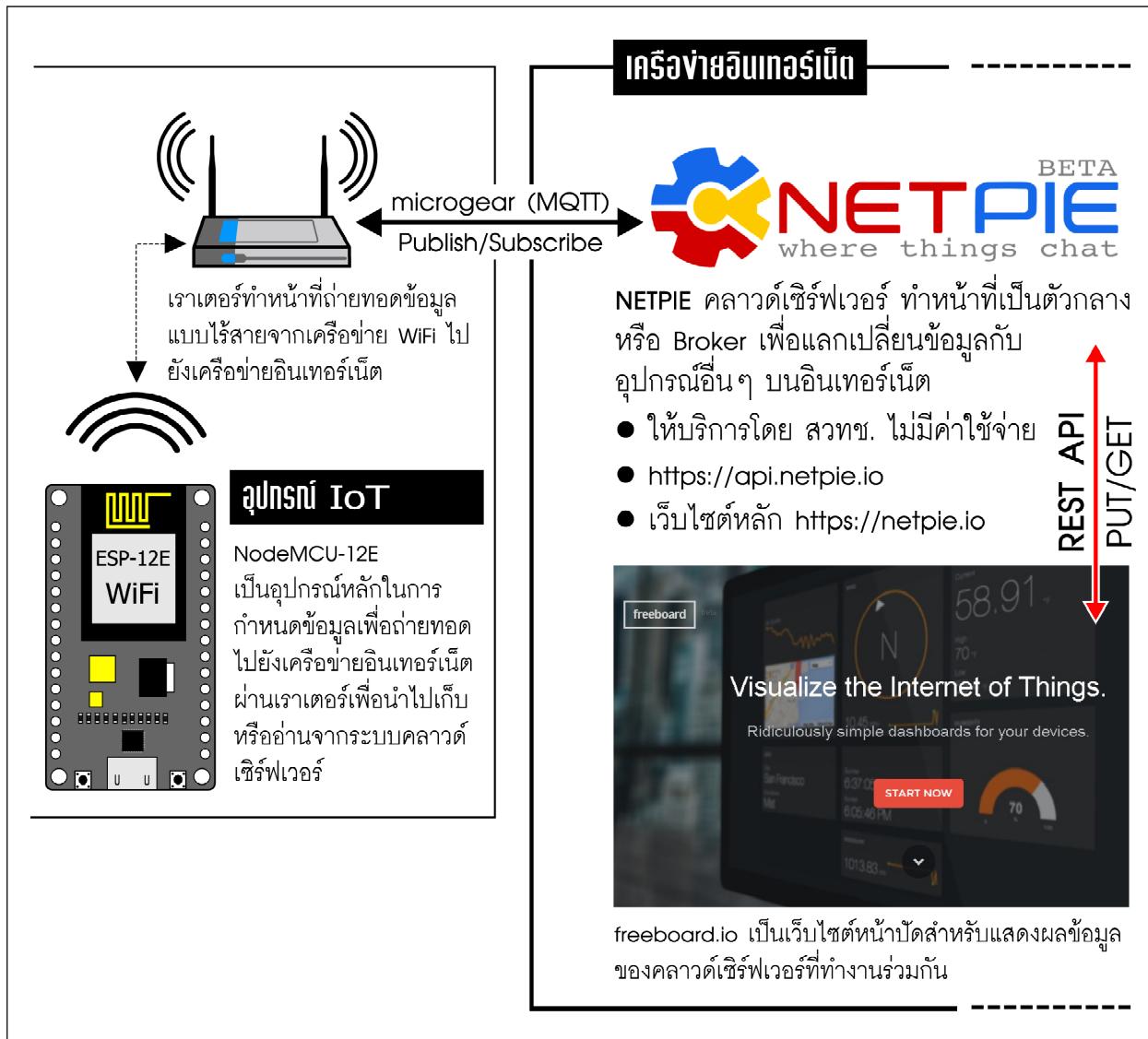
### 4.1 ภาพรวมการเชื่อมต่อ

ในรูปที่ 4-1 แสดงการเชื่อมต่อของระบบทั้งหมด โดย NETPIE เป็นตัวกลางในการทำงาน จะเห็นได้ว่า ในการติดต่อระหว่าง NETPIE กับ NodeMCU-12E และ NETPIE กับ freeboard.io ใช้โปรโตคอลที่แตกต่างกัน

โดยการติดต่อกับ NodeMCU-12E จะใช้โปรโตคอล MQTT ซึ่งใช้ไลบรารี microgear ที่ทาง NETPIE จัดทำขึ้น

ส่วนการติดต่อกับ freeboard จะใช้โปรโตคอล REST API ในการสื่อสาร

เนื่องจากการตอบกลับของ NETPIE เมื่อใช้ REST API จะอยู่ในรูปแบบ JSON ดังนั้นการเพิ่ม Datasource บน freeboard จึงต้องเลือกแบบ JSON ด้วย และก่อนที่จะเพิ่ม Datasource บน freeboard นั้น NodeMCU-12E ต้องส่งข้อมูลความไปยัง NETPIE ก่อน



รูปที่ 4-1 กระบวนการสื่อสารระหว่าง Freeboard และ NodeMCU-12E-12E โดยมี NETPIE เป็นตัวกลาง

## 4.2 ตัวอย่างการอ่านค่าจาก NodeMCU มาแสดงผลบน freeboard ผ่าน NETPIE

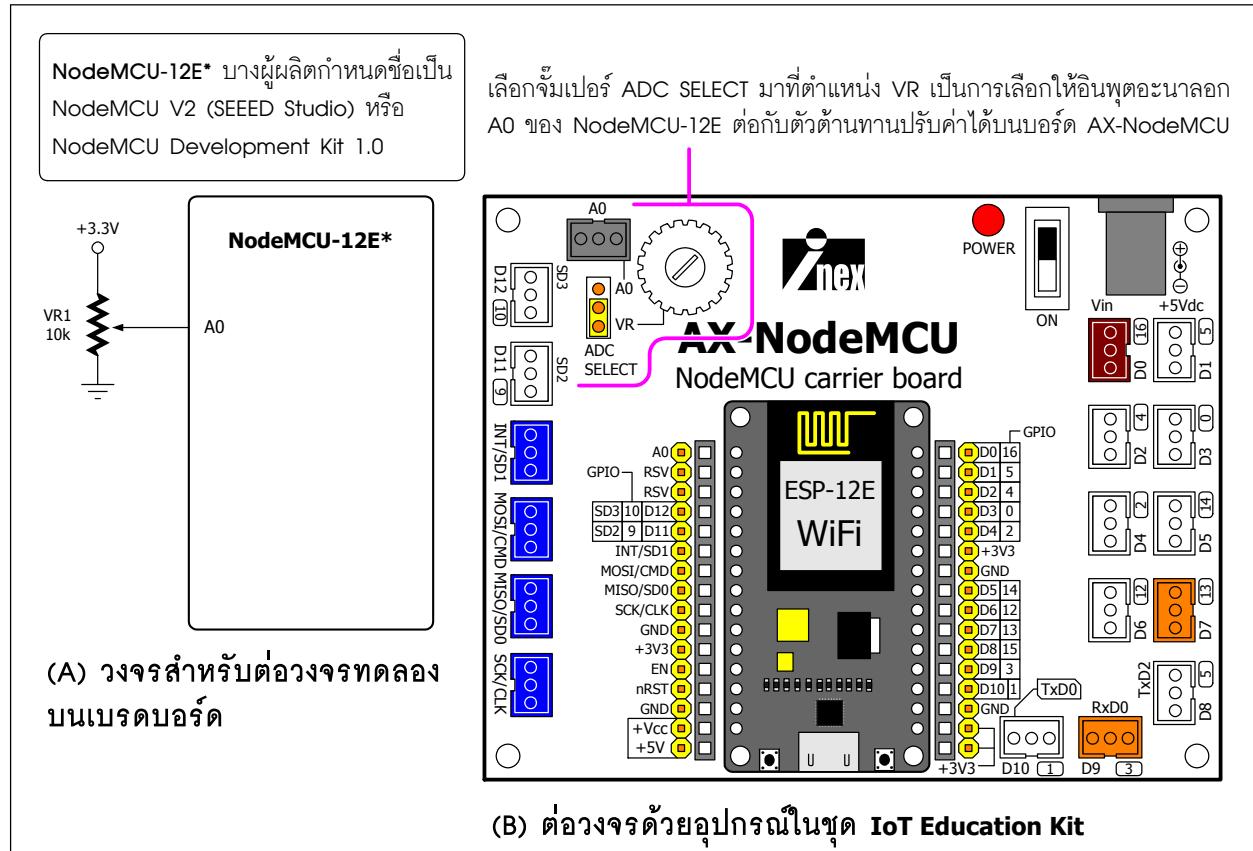
ตัวอย่างที่นำเสนอนี้คือ กำหนดให้ NodeMCU-12E ส่งค่าของสัญญาณอะนาล็อกที่อ่านได้จากอินพุต A0 ไปยัง NETPIE โดยใช้ Topic ที่ชื่อว่า /valAnalog ด้วยคำสั่ง publish("/valAnalog", -{Analog},1) โดยตัวเลข 1 หมายความว่า ข้อมูลที่ส่งขึ้นไปจะยังคงเก็บรักษาไว้ เพื่อให้กระบวนการของ REST API อ่านข้อมูล โดยการส่งแบบนี้จะส่งได้ 10 ข้อมูลต่อ 1 วินาที จากนั้นให้ freeboard อ่านข้อมูลจาก NETPIE ด้วยโปรโตคอล REST API ทุกๆ 1 วินาที

### 4.2.1 ขั้นตอนทางฝั่ง NodeMCU-12E

(4.2.1.1) เปิดโปรแกรม Arduino IDE 1.6.5r5 เลือกบอร์ดเป็น NodeMCU1.0 และเลือกพอร์ตเข้ามือถือให้ถูกต้อง

(4.2.1.2) พิมพ์โปรแกรมที่ 4-1 บันทึกไฟล์ในชื่อ NETPIEPubAnalog.ino

(4.2.1.3) ใช้วงจรในรูปที่ 4-2 ในการทดลอง ซึ่งก็คือ การใช้ตัวด้านท่านปรับค่าได้บนบอร์ด AX-NodeMCU ในการทดลองนั้นเอง เพียงเลือกตัวจ้มเปลอร์มาอย่างตำแหน่ง VR



รูปที่ 4-2 วงจรและการใช้บอร์ด AX-NodeMCU ในการทดสอบ

```

#include <MicroGear.h>
#include <ESP8266WebServer.h>
const char* ssid = <SSID>
const char* password = <PASS>
#define APPID "GroupInex"
#define KEY "Yn1yT7QqyKyYFk6"
#define SECRET "Ksn1hYJ2ELSNjsKb2iiJa6YIg"
#define ALIAS "esp"

WiFiClient client;
int timer = 0;
unsigned long previousMillis = 0;
const long interval = 250;
MicroGear microgear(client);

void onMsghandler(char *topic, uint8_t* msg, unsigned int msglen)
{
    msg[msglen] = '\0';
    Serial.print(String("Topic--> ") + topic);
    Serial.println(String(",Massage--> ") + (char *)msg);
}
void onConnected(char *attribute, uint8_t* msg, unsigned int msglen)
{
    Serial.println("Connected to NETPIE....Now..");
    microgear.subscribe("/valAnalog");
}
void setup()
{
    Serial.begin(115200);
    microgear.on(MESSAGE, onMsghandler);
    microgear.on(CONNECTED, onConnected);
    Serial.println("Starting...");
    if (WiFi.begin(ssid, password))
    {
        while (WiFi.status() != WL_CONNECTED)
        {
            delay(500);
            Serial.print(".");
        }
    }
    Serial.println("WiFi connected" + WiFi.SSID());
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
    //uncomment the line below if you want to reset token-->
    microgear.resetToken();
    microgear.init(KEY, SECRET, ALIAS);
    microgear.connect(APPID);
}

```

โปรแกรมที่ 4-1 ไฟล์ NETPIEPubAnalog.ino โปรแกรมสำหรับ NodeMCU-12E เพื่ออ่านค่าอินพุตจาก analog ที่ต่อเข้าไปยัง NETPIE และแสดงผลด้วยเดชบอร์ดที่ freeboard.io (มีต่อ)

```

void loop()
{
    if (microgear.connected())
    {
        microgear.loop();
        unsigned long currentMillis = millis();
        if (currentMillis - previousMillis >= interval)
        {
            // save the last time
            previousMillis = currentMillis;
            int anVal=analogRead(A0);
            microgear.publish("/valAnalog",anVal,true);
        }
    }
    else
    {
        Serial.println("connection lost, reconnect...");
        microgear.connect(APPID);
        delay(timer);
        timer += 100;
    }
}

```

### คำอธิบายโปรแกรมเพิ่มเติม

จากโปรแกรม คุ้มครองจะบอกรับข้อมูลหรือ subscribe โดยใช้คำสั่ง microgear.subscribe("/valAnalog") มีหัวข้อการติดต่อหรือ Topic ที่ชื่อว่า /valAnalog ซึ่งเป็นหัวข้อเดียวกันกับการเผยแพร่ข้อมูลหรือ publish และการ publish ในครั้งนี้ต้องเก็บรักษาข้อมูลไว้ด้วย โดยใช้คำสั่ง microgear.publish("/valAnalog",bufAn,1) แต่ข้อจำกัดในการนี้คือ ส่งข้อมูลไปยัง NETPIE ได้ 10 ข้อมูลต่อ 1 วินาที ดังนั้นจึงใช้ฟังก์ชัน millis() มาช่วยในการสร้าง เงื่อนไขกำหนดเวลาส่งข้อมูล

ในการ publish แบบเก็บรักษาข้อมูลไว้ ทุกครั้งที่คุ้มครองมีการ subscribe โดยใช้ Topic เดียวกัน คุ้มครอง ตัวนั้นจะได้รับข้อมูลล่าสุดที่ส่งไปยัง NETPIE ก่อน 1 ครั้งเสมอ

---

**โปรแกรมที่ 4-1 ไฟล์ NETPIEPubAnalog.ino โปรแกรมสำหรับ NodeMCU-12E เพื่ออ่านค่าอินพุตอะนาล็อก ส่งขึ้นไปยัง NETPIE และแสดงผลด้วยแดชบอร์ดที่ freeboard.io (จบ)**

```

COM9
Send
Incoming message --> /GroupInex/valAnalog : 373
Incoming message --> /GroupInex/valAnalog : 373
Incoming message --> /GroupInex/valAnalog : 370
Incoming message --> /GroupInex/valAnalog : 369
Incoming message --> /GroupInex/valAnalog : 368
Incoming message --> /GroupInex/valAnalog : 367
Incoming message --> /GroupInex/valAnalog : 368
Incoming message --> /GroupInex/valAnalog : 364
Incoming message --> /GroupInex/valAnalog : 355
Incoming message --> /GroupInex/valAnalog : 352
Incoming message --> /GroupInex/valAnalog : 345
Incoming message --> /GroupInex/valAnalog : 339

```

Autoscroll      No line ending      115200 baud

รูปที่ 4-3 หน้าต่าง Serial Monitor แสดงผลการทำงานของโปรแกรมที่ 4-1

(4.2.1.4) อัปโหลดโปรแกรมลงบน NodeMCU-12E และให้เปิดหน้าต่าง Serial Monitor เพื่อดูผลการทำงาน ดังรูปที่ 4-3

จากรูปจะเห็นว่า หัวข้อหรือ Topic จริงๆ ของการส่งข้อมูลนี้คือ */GroupInex/valAnalog* ดังนั้นจะนำ Topic นี้ไปอ่านข้อมูลโดยใช้ REST API และทดสอบการอ่านข้อมูลด้วยเว็บบราวเซอร์ต่อไป

(4.2.1.5) เปิดเว็บบราวเซอร์ และป้อน URL ต่อไปนี้

**https://api.NETPIE.io/topic/GroupInex/valAnalog?auth=Yn1yT7QqyKyYFk6:Ksn1hYJ2ELSNjsKb2iiJa6YIg**

โดยที่

**Yn1yT7QqyKyYFk6** คือ Key ของ AppID ที่ใช้ GroupInex

**Ksn1hYJ2ELSNjsKb2iiJa6YIg** คือ Secret ของ AppID ที่ใช้ GroupInex

#### ผลลัพธ์ในแบบ JSON

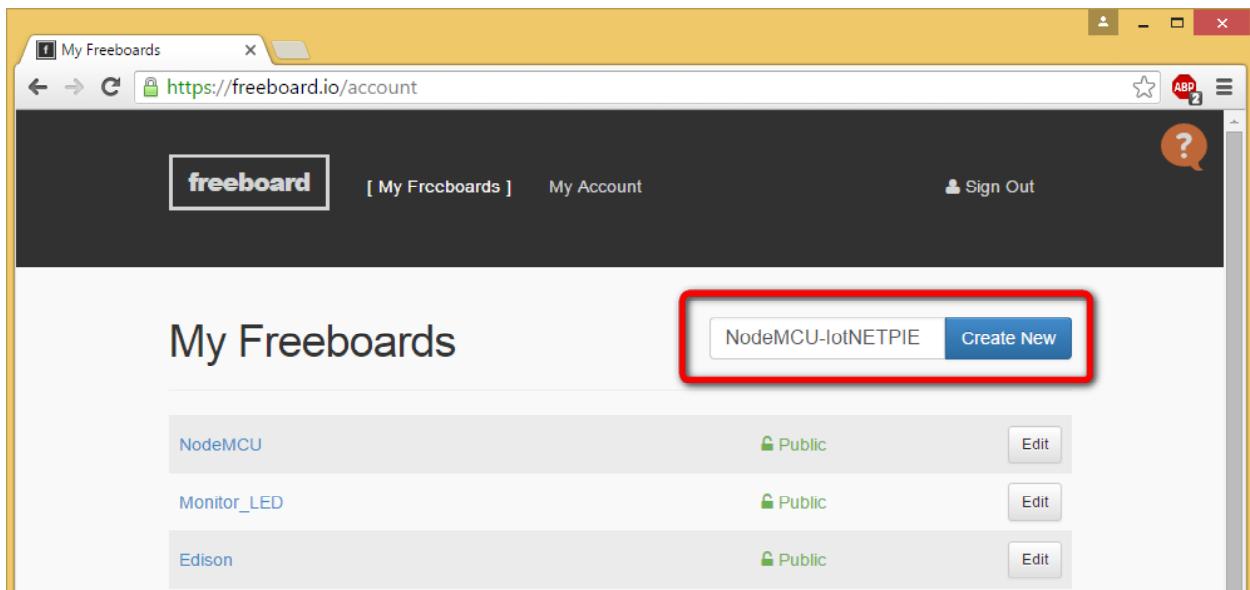
[{"topic":"/GroupInex/valAnalog","payload":"476","qos":0,"retain":true}]

#### 4.2.2 การออกแบบ Dashboard บน Freeboard.io

มีขั้นตอนดังนี้

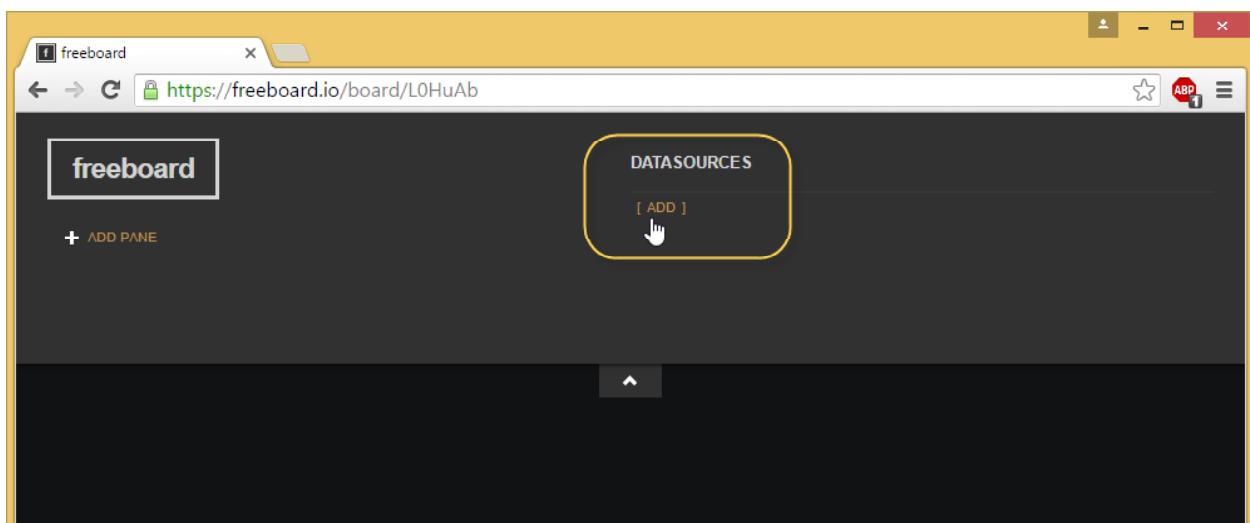
(4.2.2.1) ไปที่เว็บไซต์ freeboard.io ทำการล็อกอินเพื่อเข้าใช้งาน หรือถ้ายังไม่มีบัญชี ให้ทำการลงทะเบียนให้เรียบร้อยก่อน

(4.2.2.2) สร้างแดชบอร์ดใหม่ ตั้งชื่อเป็น NodeMCU-IotNETPIE ดังรูปที่ 4-4



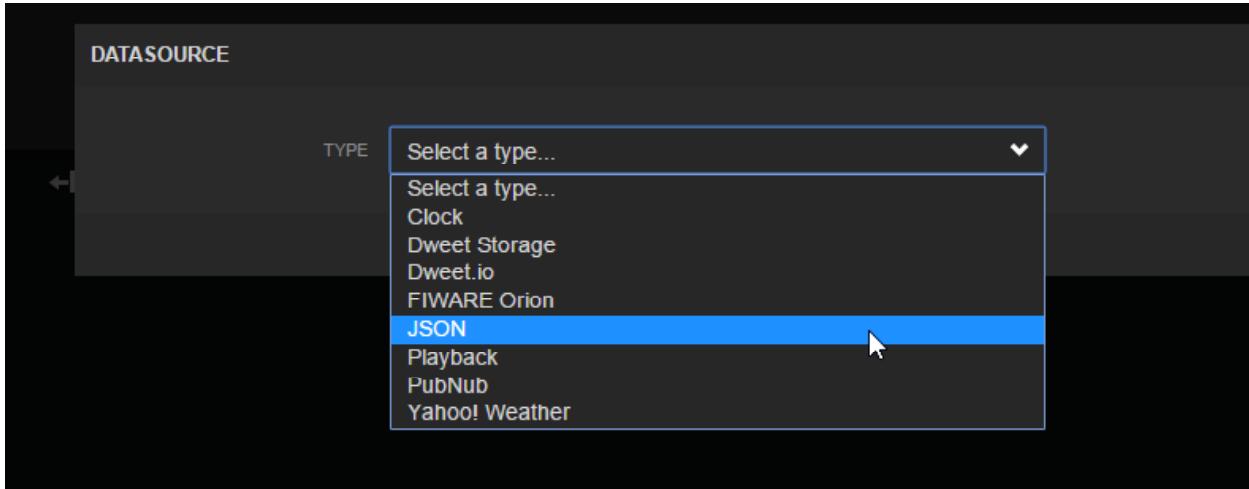
รูปที่ 4-4 เข้าสู่เว็บไซต์ freeboard.io ทำการล็อกอินและสร้างแดชบอร์ดใหม่ในชื่อ NodeMCU-12E-IotNETPIE

(4.2.2.3) เพิ่ม DATA SOURCES ใน freeboard โดยคลิกที่ ADD ดังรูปที่ 4-5



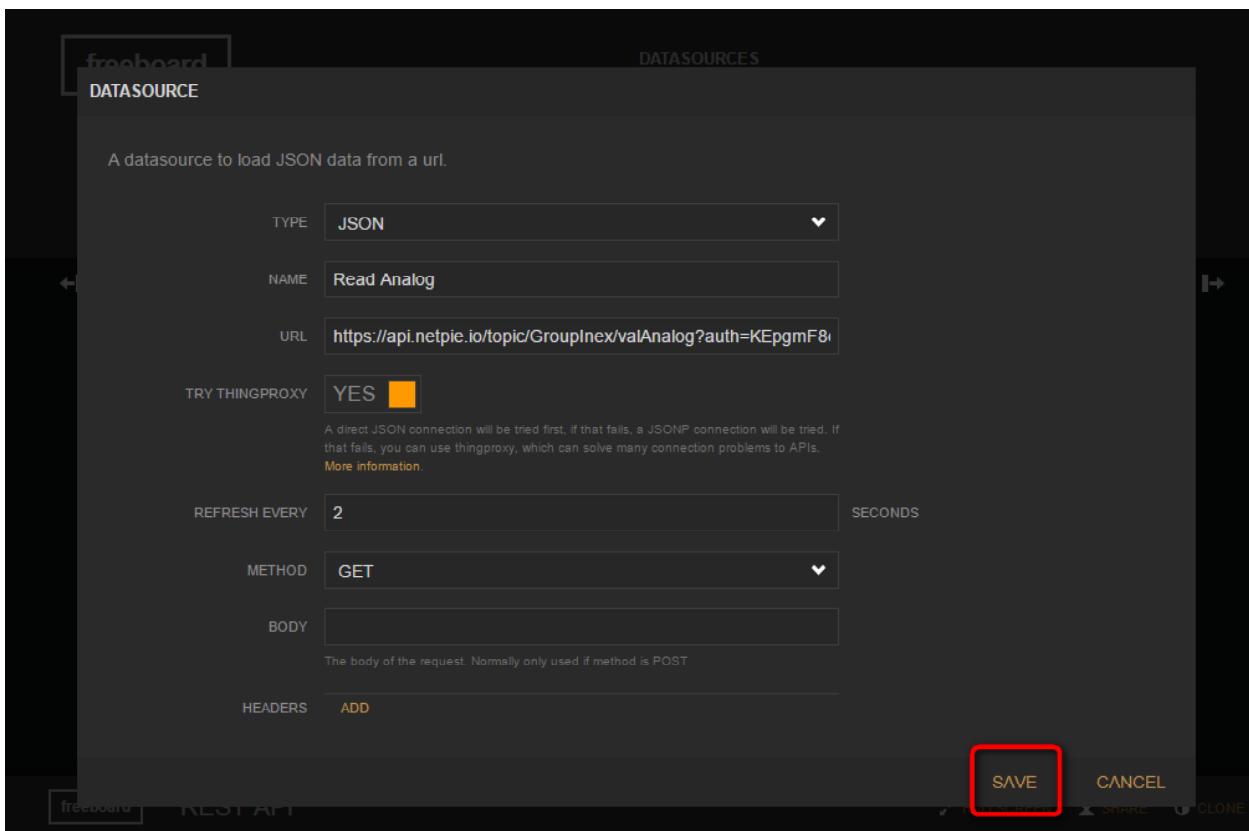
รูปที่ 4-5 เพิ่มแหล่งข้อมูลหรือ DATA SOURCES

(4.2.2.4) เลือกรายการชนิดของ DATASOURCE เป็นแบบ JSON ดังรูปที่ 4-6



รูปที่ 4-6 เลือกแหล่งข้อมูลเป็น JSON

(4.2.2.5) จะปรากฏหน้าต่างดังรูปที่ 4-7



รูปที่ 4-7 ตั้งค่าของ DATASOURCE เมื่อเลือกชนิดข้อมูลเป็น JSON

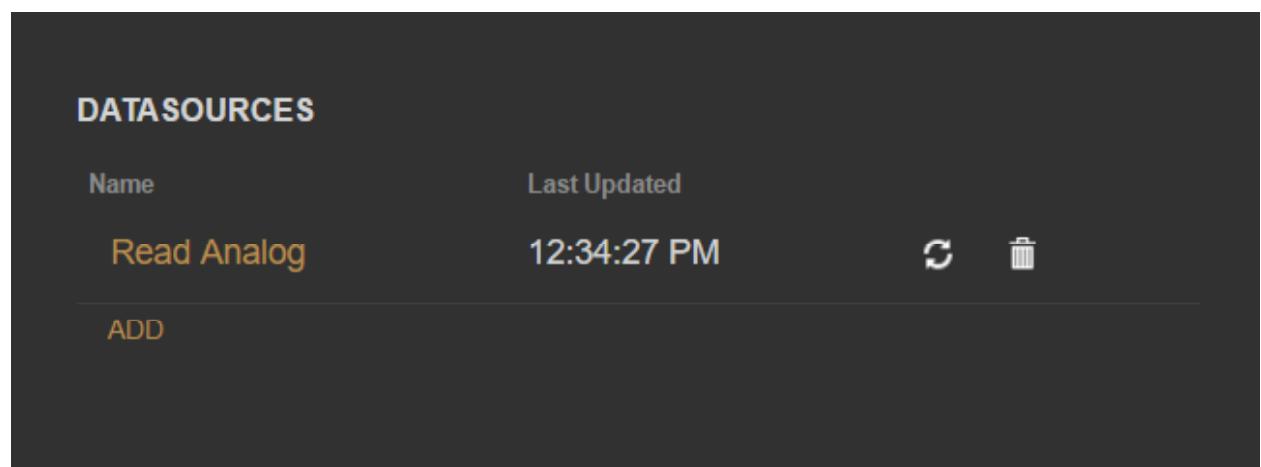
ที่ชื่อง NAME - ตั้งชื่อได้ตามต้องการ

ที่ชื่อง URL - ใช้ตัวແໜ່ງເດືອນກັບກາຣທົດສອບດ້ວຍເວັບບວກເຫຼົອໃນທີ່ຜ່ານມາ ນັ້ນຄືອ

`https://api.NETPIE.io/topic/GroupIndex/valAnalog?auth=Yn1yT7QqyKyYFk6: Ksn1hYJ2ELSNjsKb2iiJa6YIg`

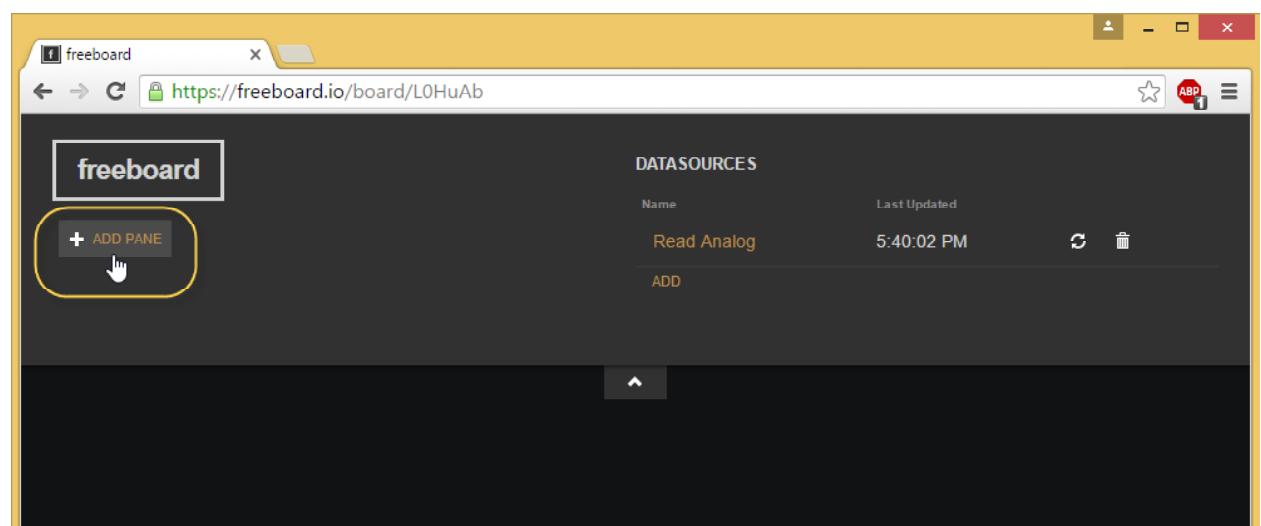
ที่ชื่อง REFRESH EVERY - ໃຊ້ກຳທັນເວລາທີ່ຕ້ອງກາຮ່ານຂໍ້ອຄວາມ ມ່ວຍເປັນວິນາທີ່  
ຈາກນັ້ນຄືກີ່ມີ **SAVE** ເພື່ອບັນທຶກ

(4.2.2.6) ເມື່ອບັນທຶກແລ້ວ ຈະມີໜີ້ຂອງ DATASOURCES ໄໝມທີ່ສ່ຽງຂຶ້ນ ປຣາກງູ້ຂຶ້ນມາ ໃນທີ່ນີ້ຄືອ  
ໜີ້ຂອງ **Read Analog** ພ້ອມກັບເວລາລ່າສຸດໃນກາຮ່ານຂໍ້ອຄວາມດັ່ງຮູບທີ່ 4-8

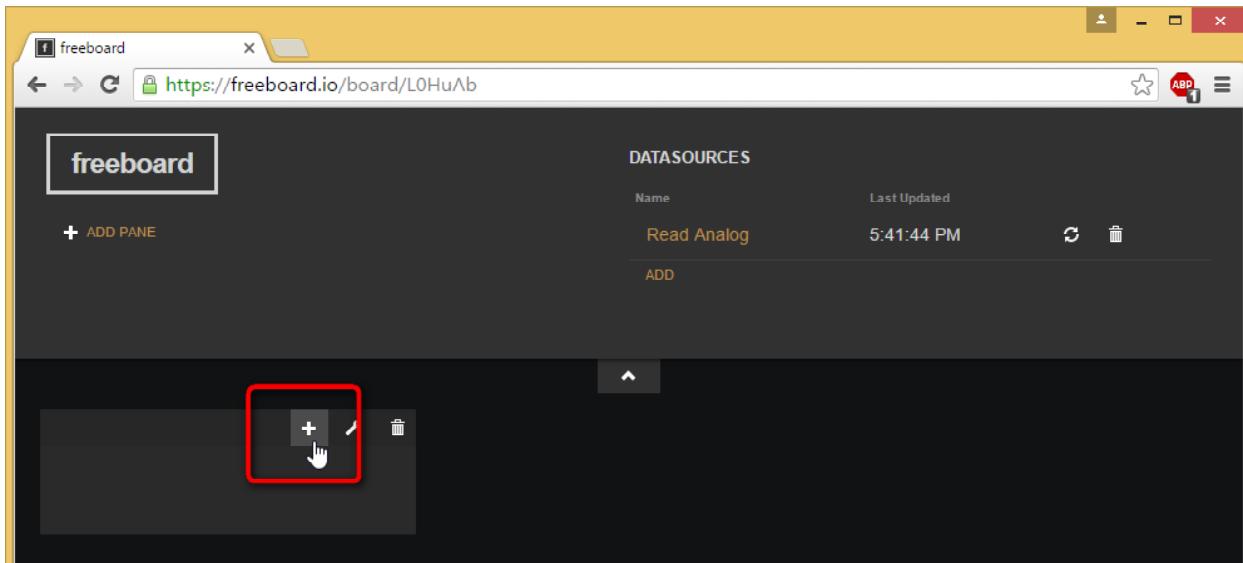


ຮູບທີ່ 4-8 ແສດ **DATASOURCES** ໄໝມທີ່ເກີດຂຶ້ນໃນໜີ້ຂອງ **Read Analog**

(4.2.2.7) ເພີ່ມສ່ວນແສດງຜລໂຮງ Pane ໂດຍຄືກີ່ **ADD PANE** ດັ່ງຮູບທີ່ 4-9



ຮູບທີ່ 4-9 ແສດກາເພີ່ມສ່ວນແສດງຜລ



รูปที่ 4-10 เพิ่มอุปกรณ์แสดงผล

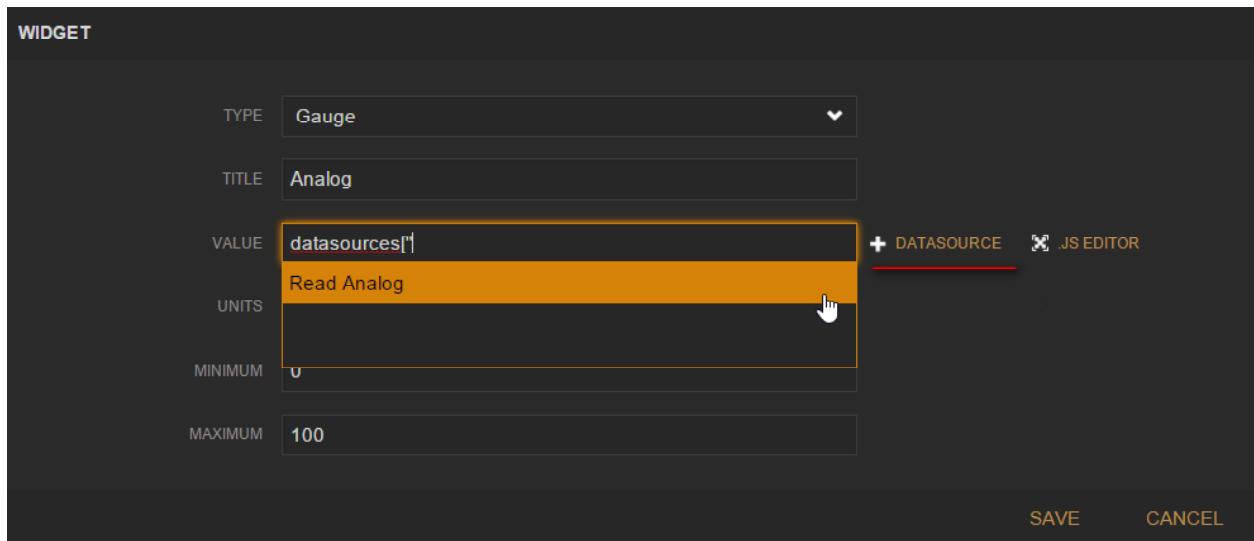
(4.2.2.8) เพิ่มอุปกรณ์แสดงผลหรือ Widget โดยคลิกที่เครื่องหมาย + ตามรูปที่ 4-10

(4.2.2.9) เลือกอุปกรณ์แสดงผลเป็นแบบ Gauge ดังรูปที่ 4-11 (ก) จะปรากฏให้ระบุค่าต่าง ดังรูปที่ 4-11 (ข) กำหนดชื่อ (TITLE) เป็น Analog

The image contains two screenshots of the Freeboard 'Widget' configuration interface. The top screenshot shows a 'TYPE' dropdown menu with options like 'Select a type...', 'Google Map', 'HTML', etc., and 'Gauge' is highlighted with a blue selection bar. The bottom screenshot shows the 'Widget' configuration form with fields for 'TITLE' (set to 'Analog'), 'VALUE', 'UNITS', 'MINIMUM' (set to '0'), and 'MAXIMUM' (set to '100'). The 'TITLE' field is highlighted with a red box. Both screenshots have a white rounded rectangle with the text '4-11 (ก)' or '4-11 (ข)' in the bottom right corner.

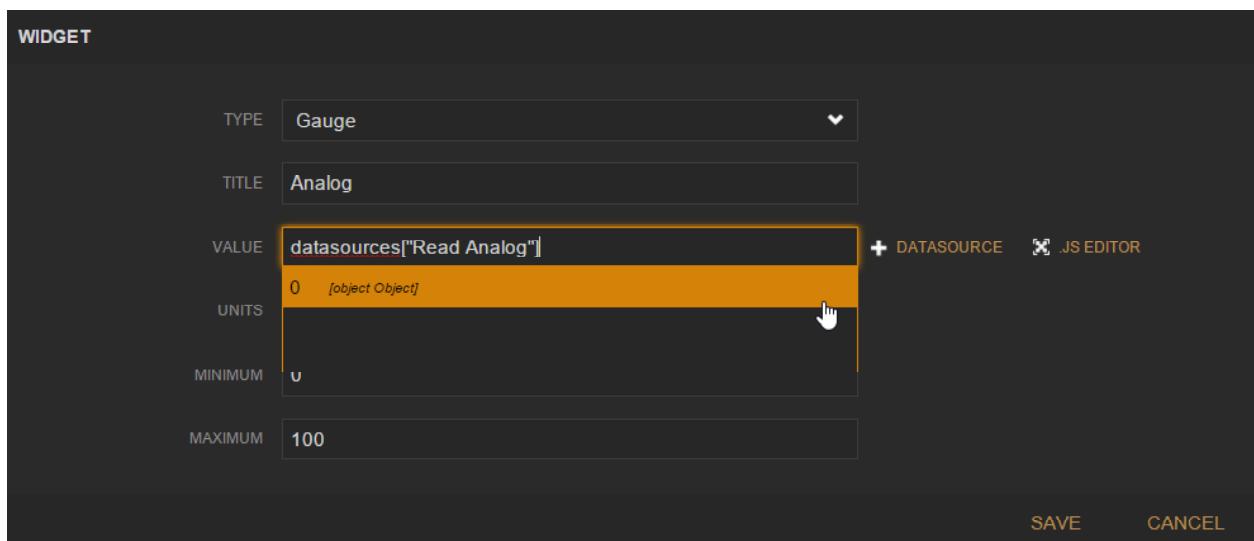
รูปที่ 4-11 เลือกวิดเก็ตหรืออุปกรณ์แสดงผลเป็น Gauge และกำหนดค่า

(4.2.2.10) ที่ช่อง VALUE ให้เลือก DATASOURCE ที่ชื่อว่า **Read Analog** โดยคลิกที่ + DATASOURCE หนึ่งครั้ง จะปรากฏช่องให้เลือกดังรูปที่ 4-12 จากนั้นคลิกเลือกรายการ **Read Analog**



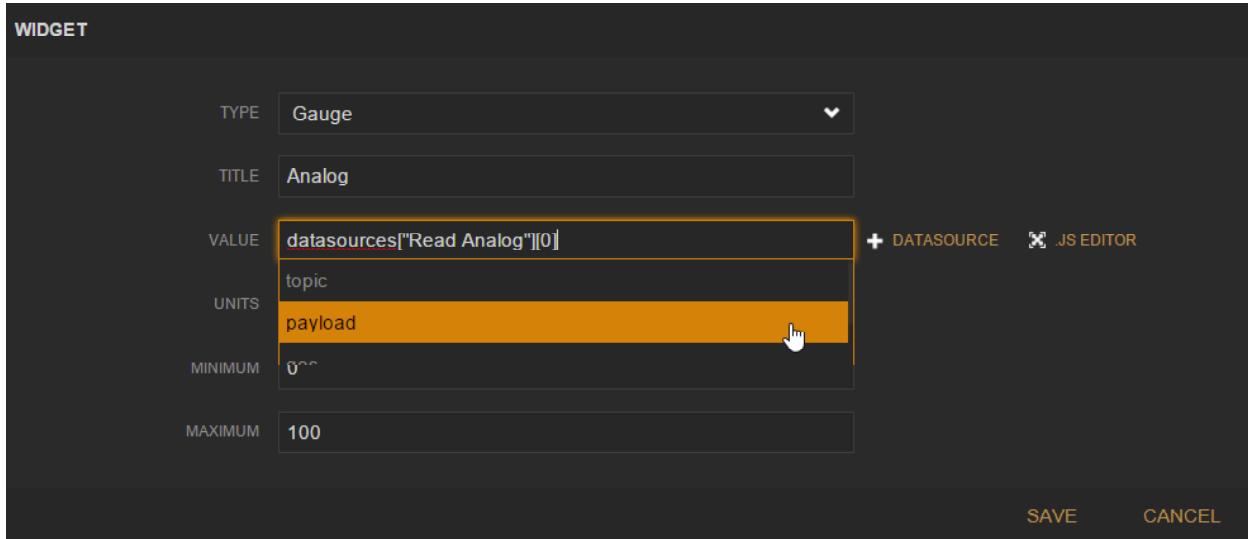
รูปที่ 4-12 เลือกแหล่งกำเนิดข้อมูลสำหรับช่อง VALUE ของตัวแสดงผลแบบ Gauge

(4.2.2.11) เมื่อกลิกลแล้ว จะปรากฏรายการให้เลือกดังรูปที่ 4-13 สาเหตุที่แสดงรายการแบบนี้เนื่องจาก หากสังเกตจากข้อมูลตอบกลับในขั้นตอนการใช้เว็บбраузเชอร์ทดสอบในขั้นตอนที่ (4.2.1.4) จะได้ข้อมูลในรูปแบบ JSON ที่มีเครื่องหมาย [ ] ครอบไว้ หมายถึง ข้อมูลนี้ถูกเก็บไว้ในแบบอาร์เรย์ และมีชุดเดียว ดังนั้นการกำหนดตัวชี้ค่า Index จึงเป็นเลข 0 จากนั้นคลิกเลือกหนึ่งครั้ง



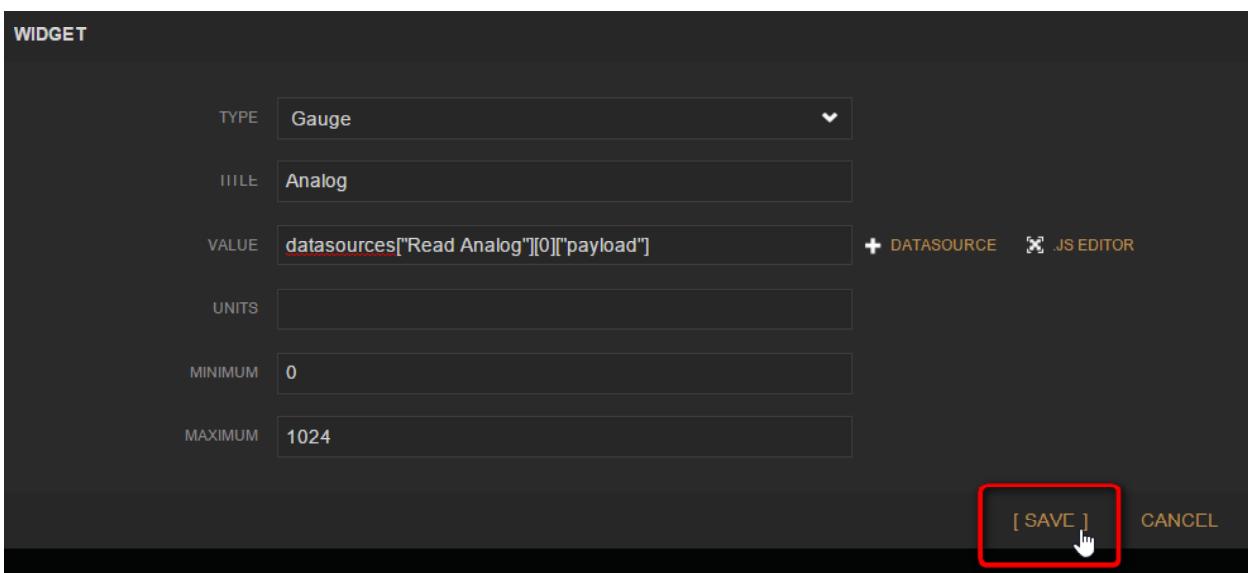
รูปที่ 4-13 เลือกรูปแบบข้อมูลที่เป็นแบบอาร์เรย์

(4.2.2.12) เมื่อกลิกแล้ว จะปรากฏข้อมูลทั้งหมดดังรูปที่ 4-14 ให้เลือกรายการ payload เพราะเป็นตัวระบุค่าอะนาล็อก



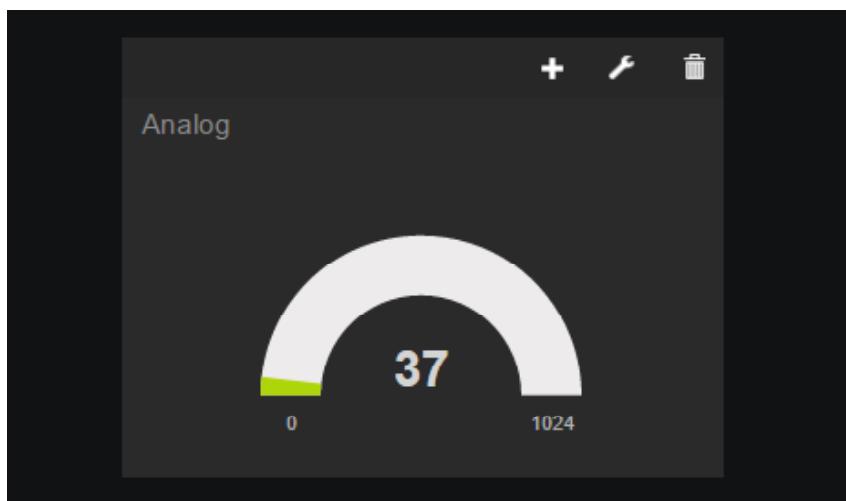
รูปที่ 4-14 เลือกพารามิเตอร์ของช่อง VALUE เพิ่มเติม

(4.2.2.13) ระบุค่าสูงสุด (MAXIMUM) เป็น 1024 ดังรูปที่ 4-15 แล้วคลิกที่ปุ่ม SAVE



รูปที่ 4-15 กำหนดค่าต่ำสุดเป็น 0 และสูงสุดเป็น 1024 แล้วบันทึกค่า

(4.2.2.14) เมื่อทำการบันทึกแล้ว จะได้ตัวแสดงผลแบบ Gauge ดังรูปที่ 4-16 จากนั้นทดลองปรับค่าของตัวต้านทานบนบอร์ด AX-NodeMCU จะเห็นการเปลี่ยนแปลงของ Gauge เกิดขึ้น



รูปที่ 4-16 ตัวแสดงผลแบบ Gauge ของ freeboard ที่เลือกใช้สำหรับการทดลองในบทนี้ ค่าของมันจะเปลี่ยนแปลงตามการหมุนตัวต้านทานปรับค่าได้บนบอร์ด AX-NodeMCU





# บทที่ 5

## การประยุกต์ใช้งาน freeboard.io เพื่อสั่งงานอุปกรณ์ผ่าน NETPIE

---

ในบทที่ผ่านมาเป็นการนำเสนอการทำงานร่วมกันของ NETPIE, NodeMCU-12E และ freeboard เพื่อแสดงค่าของสัญญาณอะนาล็อกที่ขาอินพุต A0 ของ NodeMCU-12E โดยข้อมูลจาก NodeMCU-12E จะถูกส่งมาเก็บไว้ที่คลาวด์เซิร์ฟเวอร์ ซึ่งก็คือ NETPIE จากนั้น freeboard จะเข้ามาอ่านค่าจาก NETPIE เพื่อนำไปแสดงผลบนแดชบอร์ดที่สร้างขึ้นต่อไป มาในบทนี้จะเพิ่มการเรียนรู้ไปอีกขั้นหนึ่ง จากการใช้งานแดชบอร์ดบนเว็บ freeboard.io เพียงเพื่อการแสดงผล มาในบทนี้จะเพิ่มปุ่มสำหรับกดเพื่อควบคุมสถานะทางลốiจิกของขาพอร์ตดิจิตอลของ NodeMCU-12E เป็นการใช้งานแดชบอร์ดบนเว็บ freeboard.io ในอีกรูปแบบหนึ่ง

### 5.1 เตรียมการ

ก่อนที่จะถึงขั้นตอนในการเขียนโปรแกรมไม่ว่าจะเป็นบน NodeMCU-12E หรือบนเว็บ freeboard.io สิ่งที่ต้องเตรียมมีดังนี้

#### 5.1.1 รหัสคีย์ทั้งหมดที่ใช้เชื่อมต่อกับ NETPIE

```
AppID = GroupIndex
Key= Yn1yT7QqyKyYFk6
Secret= Ksn1hYJ2ELSNjsKb2iiJa6YIg
```

#### 5.1.2 หัวข้อของการติดต่อหรือ Topic

5.1.2.1 ใช้ Topic ชื่อ /Send\_stLED ในการส่งข้อความโดยใช้ REST API จาก freeboard ดังนั้นที่ NodeMCU-12E จะต้องทำการบอกรับหรือ subscribe หัวข้อการติดต่อนี้

5.1.2.2 ใช้ Topic ชื่อ /Feedback\_stLED เพื่อส่งให้ NodeMCU-12E ส่งค่าสถานะกลับมา ยัง freeboard แล้ว ดังนั้น NodeMCU-12E จะต้องทำการเผยแพร่ข้อมูลหรือ publish แบบ retain เพื่อส่งค่ากลับมา ส่วน freeboard จะใช้ REST API ในการอ่านข้อความของ Topic นี้

### 5.1.3 เตรียม URL ที่ใช้กับ freeboard

5.1.3.1 จากข้อ 5.1.2.1 เว็บ freeboard.io ใช้ REST API ในการส่งข้อมูลความที่ไปยัง NodeMCU-12E ถ้าส่ง “1” คือสถานะเปิดและ “0” คือสถานะปิด ใช้การส่งแบบ PUT โดยมีรูปแบบ URL ดังนี้

```
https://api.NETPIE.io /topic/{AppID}/{topic}?retain&auth={AppKey}:{AppSecret}
```

ตัวอย่างที่ 5-1

```
https://api.NETPIE.io/topic/GroupInex/Send_stLED?retain&auth=Yn1yT7QqyKyYFk6:Ksn1hYJ2ELSNjsKb2iiJa6YIg
```

โดยที่

Yn1yT7QqyKyYFk6 คือ Key ของ AppID ที่ใช้ GroupInex

Ksn1hYJ2ELSNjsKb2iiJa6YIg คือ Secret ของ AppID ที่ใช้ GroupInex

5.1.3.2 จากข้อ 5.1.2.2 เว็บ freeboard.io จะใช้ REST API อ่านข้อมูลที่ NodeMCU-12E ส่งค่ากลับมา โดยใช้การอ่านแบบ GET มีรูปแบบ URL ดังนี้

```
https://api.NETPIE.io /topic/{AppID}/{topic}?auth={AppKey}:{AppSecret}
```

ตัวอย่างที่ 5-2

```
https://api.NETPIE.io/topic/GroupInex/Feedback_stLED?auth=Yn1yT7QqyKyYFk6:Ksn1hYJ2ELSNjsKb2iiJa6YIg
```

โดยที่

Yn1yT7QqyKyYFk6 คือ Key ของ AppID ที่ใช้ GroupInex

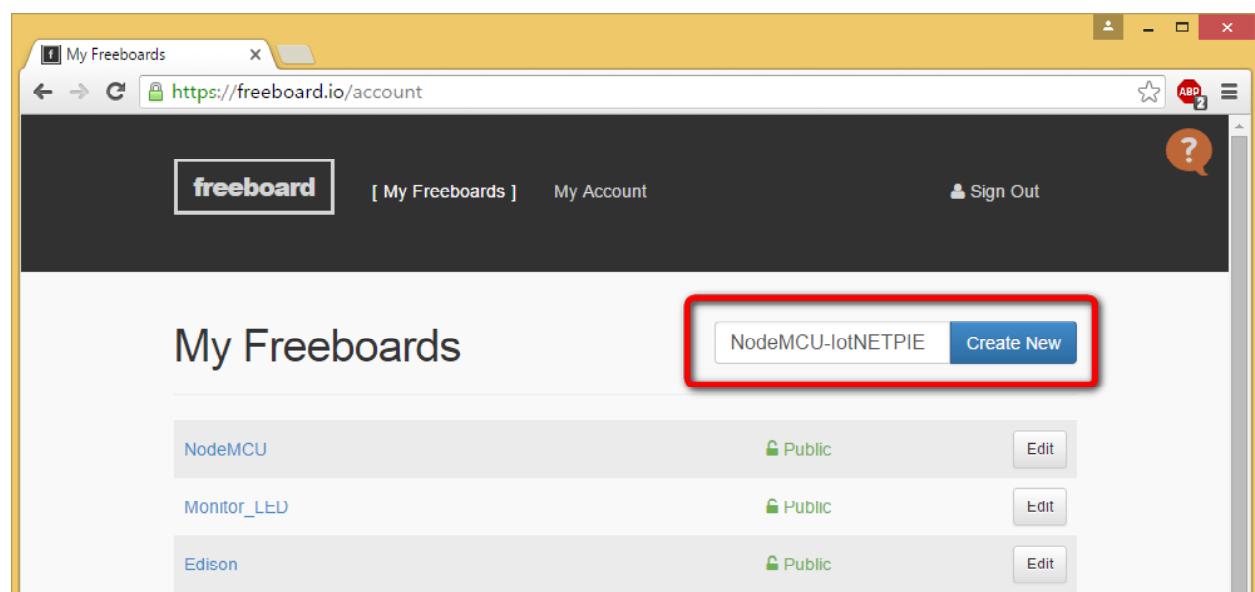
Ksn1hYJ2ELSNjsKb2iiJa6YIg คือ Secret ของ AppID ที่ใช้ GroupInex

## 5.2 สร้างแดชบอร์ดสำหรับควบคุม LED

### 5.2.1 การเตรียมการสำหรับ freeboard.io

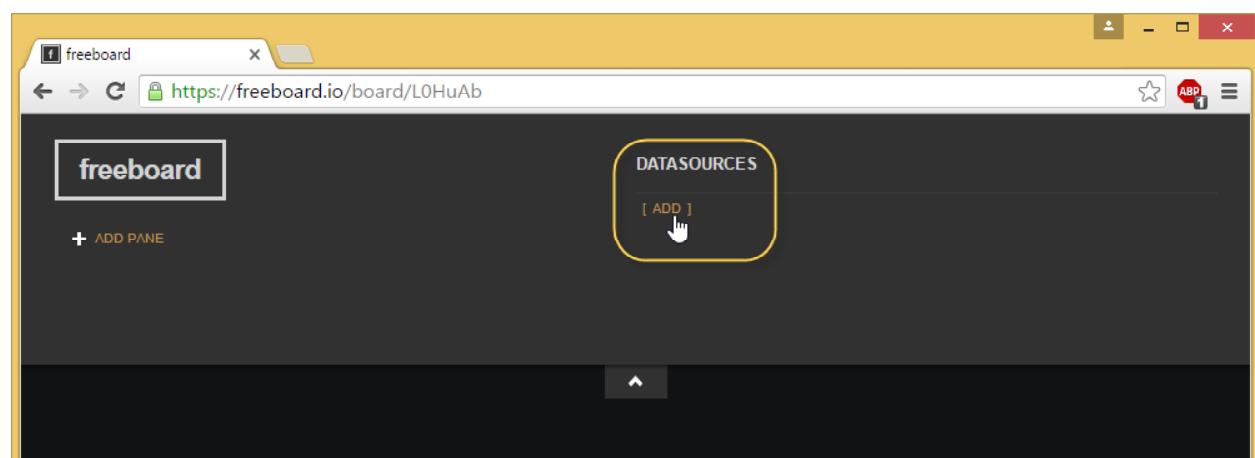
(5.2.1.1) ไปที่เว็บไซต์ freeboard.io ทำการล็อกอินเพื่อเข้าใช้งาน หรือถ้ายังไม่มีบัญชี ให้ทำการลงทะเบียนก่อน

(5.2.1.2) สร้างแดชบอร์ดใหม่ ตั้งชื่อเป็น NodeMCU-IoTNETPIE และคลิกปุ่ม Create new ดังรูปที่ 5-1 หรืออาจใช้แดชบอร์ดเดิมที่สร้างไว้ก็ได้



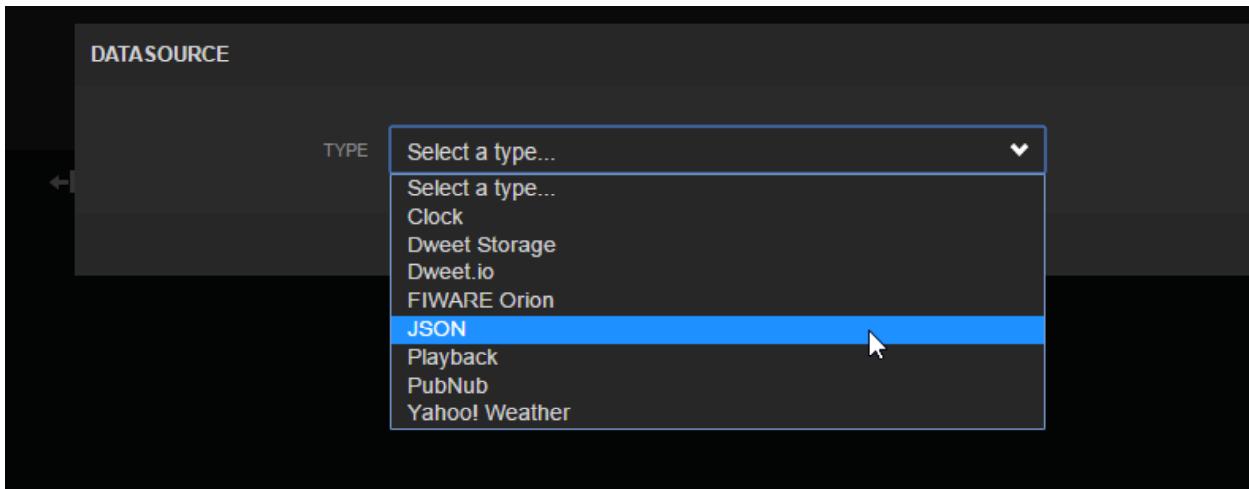
รูปที่ 5-1 แสดงหน้าเว็บ freeboard.io เมื่อเริ่มต้นสร้างแดชบอร์ดใหม่

(5.2.1.3) เพิ่ม DATA SOURCES ใน freeboard โดยคลิกที่ [ADD] ดังรูปที่ 5-2



รูปที่ 5-2 เพิ่มแหล่งข้อมูล DATA SOURCES ให้แก่แดชบอร์ด

(5.2.1.4) เพิ่ม DATASOURCE โดยเลือกรายการ JSON ตามรูปที่ 5-3



รูปที่ 5-3 เลือกแหล่งข้อมูลเป็น JSON

(5.2.1.5) เมื่อคลิกเลือกแล้ว จะปรากฏหน้าต่างดังรูปที่ 5-4

A datasource to load JSON data from a url.

TYPE	JSON
NAME	LED Status
URL	https://api.netpie.io /topic/GroupInex/Feedback_stLED?auth=KE
TRY THINGPROXY	YES <input checked="" type="checkbox"/>
A direct JSON connection will be tried first, if that fails, a JSONP connection will be tried. If that fails, you can use thingproxy, which can solve many connection problems to APIs. <a href="#">More information.</a>	
REFRESH EVERY	1 SECONDS
METHOD	GET
BODY	The body of the request. Normally only used if method is POST
HEADERS	ADD

รูปที่ 5-4 ตั้งค่าของ DATASOURCE เมื่อเลือกชนิดข้อมูลเป็น JSON

ที่ชื่อของ NAME - ตั้งชื่อได้ตามต้องการ

ที่ชื่อของ URL - ใช้ลิงก์ที่อ่านค่าจาก NodeMCU-12E ในหัวข้อ 5.1.3.2 นั่นคือ

`https://api.NETPIE.io/topic/GroupInex/Feedback_stLED?  
auth=Yn1yT7QqyKyYFk6:Ksn1hYJ2ELSNjsKb2iiJa6YIg`

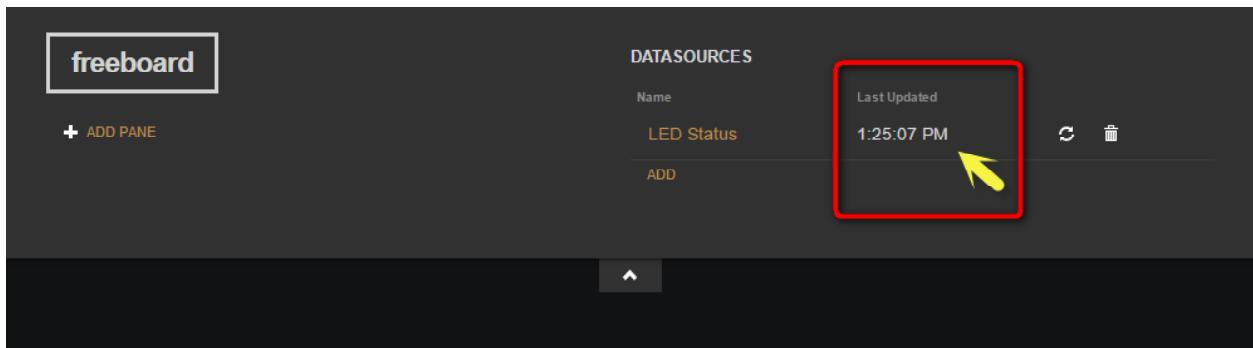
โดยที่

Yn1yT7QqyKyYFk6 คือ Key ของ AppID ที่ใช้ GroupInex

Ksn1hYJ2ELSNjsKb2iiJa6YIg คือ Secret ของ AppID ที่ใช้ GroupInex

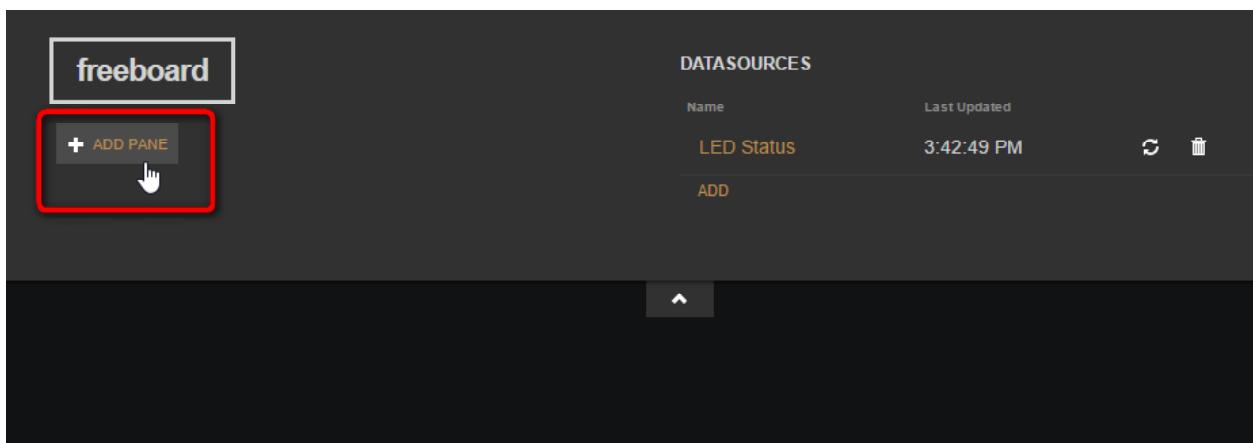
ที่ชื่อของ REFRESH EVERY - ใช้กำหนดเวลาที่ต้องการอ่านข้อมูลความหน่วงเป็นวินาที จากนั้นคลิกที่ปุ่ม SAVE เพื่อบันทึก

เมื่อบันทึกแล้ว จะมีชื่อ DATASOURCE ที่กำหนดไว้ และแสดงเวลาล่าสุดในการอ่านข้อมูลดังรูปที่ 5-5



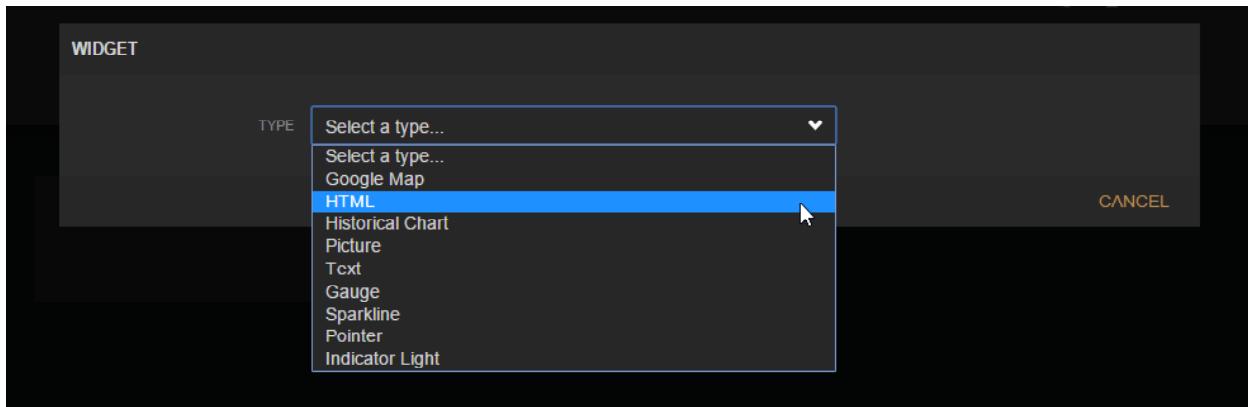
รูปที่ 5-5 แสดง DATA SOURCES ใหม่ที่เกิดขึ้นในชื่อ LED Status

(5.2.1.6) เพิ่มส่วนแสดงผลหรือ Pane โดยคลิกที่ ADD PANE ดังรูปที่ 5-6



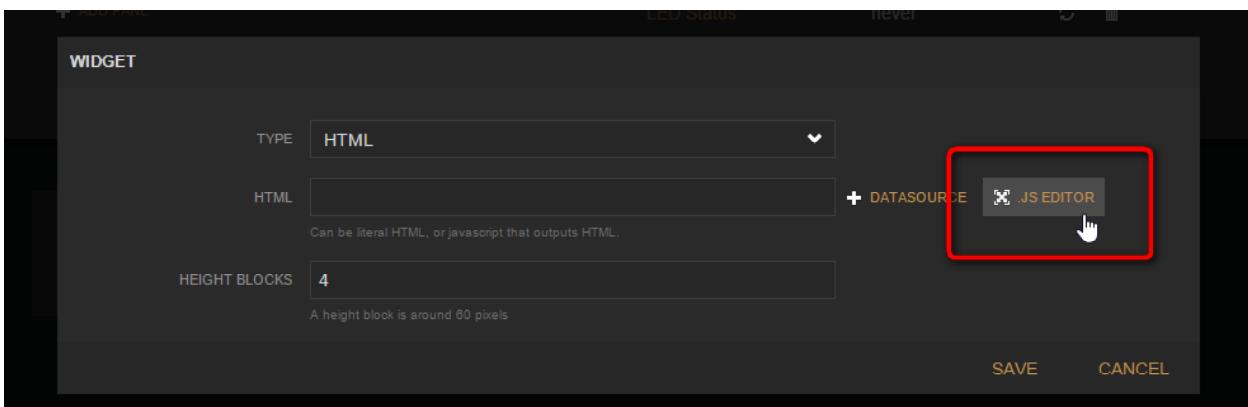
รูปที่ 5-6 คลิกปุ่ม +ADD PANE เพื่อเพิ่มส่วนแสดงผลให้แก่แดชบอร์ด

(5.2.1.7) เลือกรูปแบบของอุปกรณ์แสดงผลเป็น HTML ตามรูปที่ 5-7



รูปที่ 5-7 เลือกรูปแบบของอุปกรณ์แสดงผลเป็น HTML

(5.2.1.8) เมื่อเลือกรูปแบบของอุปกรณ์แสดงผลเป็นแบบ HTML นั้นหมายความว่า ผู้พัฒนาสามารถเขียนสคริปต์เพื่อกำหนดลักษณะของอุปกรณ์แสดงผลเองได้ ในที่นี้ต้องการสร้างปุ่มกด จึงเลือกที่จะสร้างจากสคริปต์ของภาษา Java หรือ JavaScript (Java Script) คลิกที่ ปุ่ม JS editor ทางขวาสุด ดังรูปที่ 5-8



รูปที่ 5-8 แสดงการเลือกเปิดหน้าต่างภาษาสคริปต์เพื่อเขียนโปรแกรมสั่นสำหรับสร้างปุ่มกด

(5.2.1.9) เมื่อเลือกเข้ามาข้าง JS editor จะปรากฏหน้าต่างสำหรับเขียนโปรแกรมดังรูปที่ 5-9

```
This javascript will be re-evaluated any time a datasource referenced here is updated, and the value you return will be displayed in the widget. You can assume this javascript is wrapped in a function of the form function(datasources) where datasources is a collection of javascript objects (keyed by their name) corresponding to the most current data in a datasource.

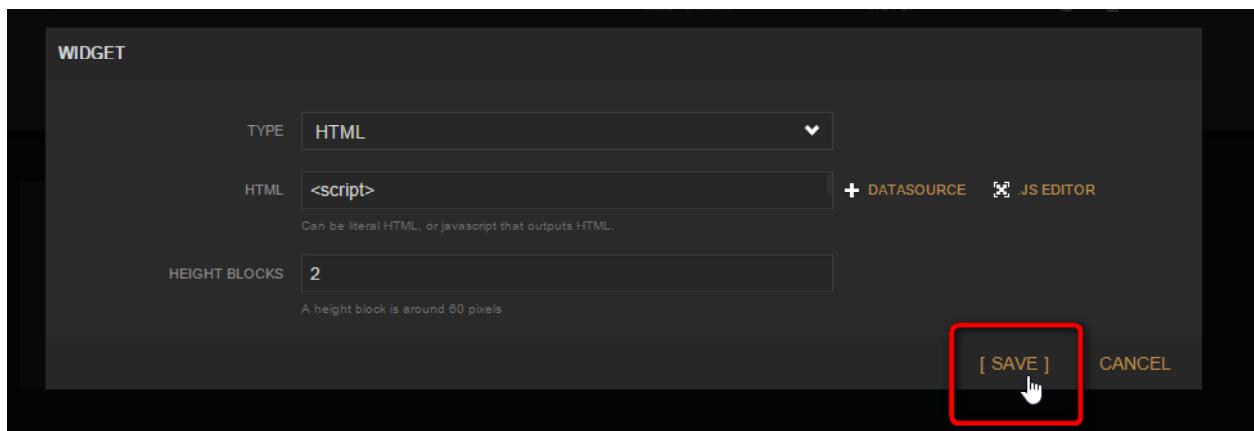
1 // Example: Convert temp from C to F and truncate to 2 decimal places.
2 // return (datasources["MyDatasource"].sensor.tempINF * 1.8 + 32).toFixed(2);
```

รูปที่ 5-9 แสดงหน้าต่างภาษาสคริปต์เอดิเตอร์เพื่อเขียนสคริปต์สำหรับสร้างปุ่มกด

(5.2.1.10) เพิ่มสคริปต์ภาษาจาวาดังนี้ โดยต้องเปลี่ยน **Key**, **Secret**, **AppID** เป็นของผู้พัฒนาตามที่เตรียมไว้

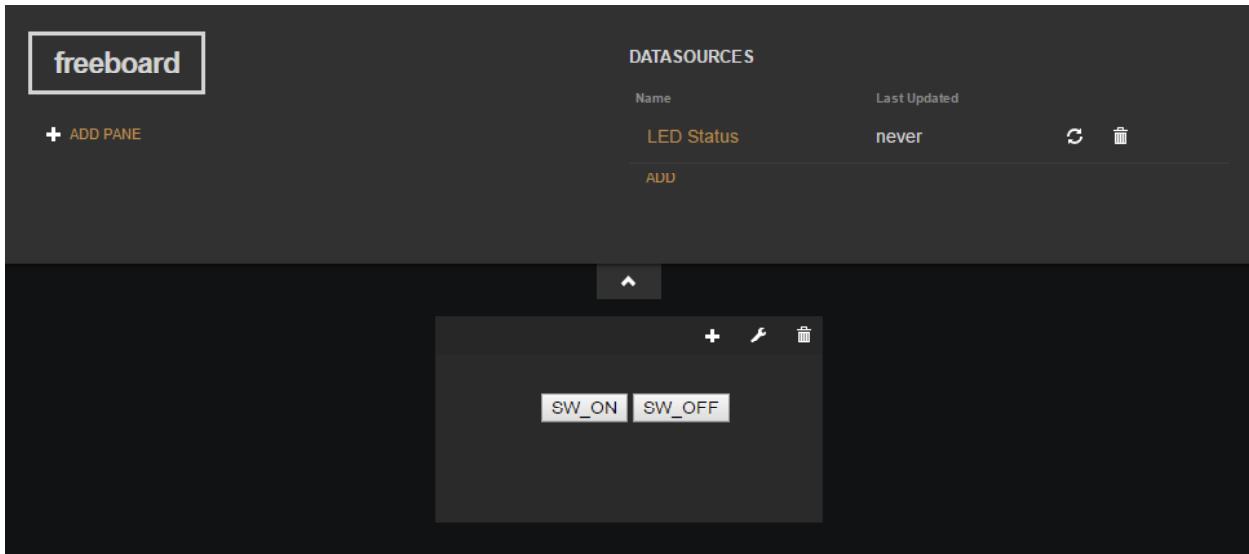
```
<script>
var KEY = 'Yn1yT7QqyKyYFk6';
var SECRET = 'Ksn1hYJ2ELSNjsKb2iiJa6YIg';
var APPID = 'GroupInex';
var Topic = '/Send_stLED';
function switchPressoff()
{
    var url='https://api.NETPIE.io/topic/' +APPID+Topic+'?retain&auth='
    '+KEY+':'+SECRET;
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.open('PUT',url,true);
    xmlhttp.send('0');
}
function switchPresson()
{
    var url='https://api.NETPIE.io/topic/' +APPID+Topic+'?retain& auth='
    '+KEY+':'+SECRET;
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.open('PUT',url,true);
    xmlhttp.send('1');
}
</script>
<center>
<div style="margin-top:30px; margin-left:10px;">
<button onclick="switchPresson()" id="Button1">SW_ON</button>
<button onclick="switchPressoff()" id="Button2">SW_OFF</button>
</div>
</center>
```

(5.2.1.11) คลิกที่ปุ่ม CLOSE ที่อยู่มุมขวาล่าง จะกลับมาขึ้นหน้าต่างเลือกรูปแบบของอุปกรณ์แสดงผลอีกครั้ง ทำการบันทึกโดยคลิก SAVE ตามรูปที่ 5-10



รูปที่ 5-10 บันทึกการแก้ไขอุปกรณ์แสดงผลแบบ HTML ซึ่งนำมาใช้ในการสร้างปุ่มกด

(5.2.1.12) เมื่อบันทึกแล้ว กลับมาขึ้นหน้าต่างหลัก จะพบปุ่มสีเหลืองขึ้นมา 2 ปุ่ม ชื่อ SW\_ON และ SW\_OFF เตรียมไว้สำหรับเปิดปิด LED ดังรูปที่ 5-11



รูปที่ 5-11 แดชบอร์ด LED Status ที่เริ่มต้นด้วยการสร้างปุ่มควบคุม 2 ตัว

## 5.2.2 ตรวจสอบข้อมูลที่ส่งไปยัง NETPIE

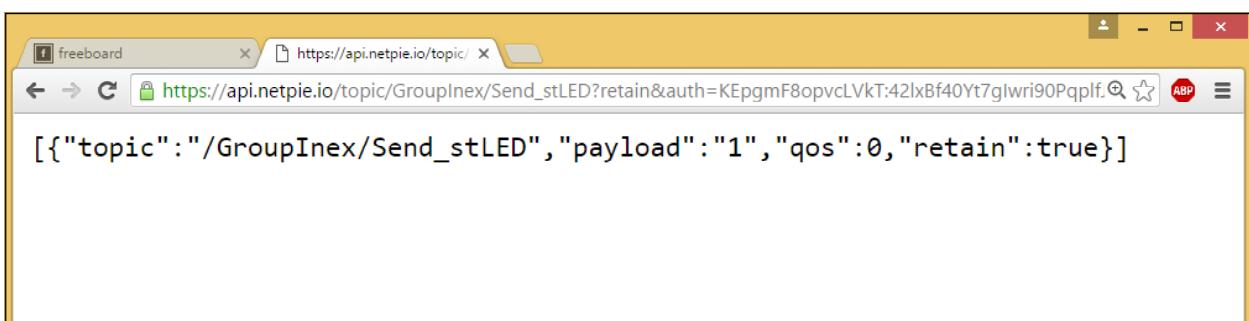
มีขั้นตอนดังนี้

(5.2.2.1) เปิดเว็บбраウเซอร์ แล้วป้อน URL ต่อไปนี้ที่แอ็คเดรสบาร์ของเว็บбраウเซอร์

```
https://api.NETPIE.io/topic/GroupInex/Send_stLED?retain&
auth=Yn1yT7QqyKyYFk6:Ksn1hYJ2ELSNjsKb2iiJa6YIg
```

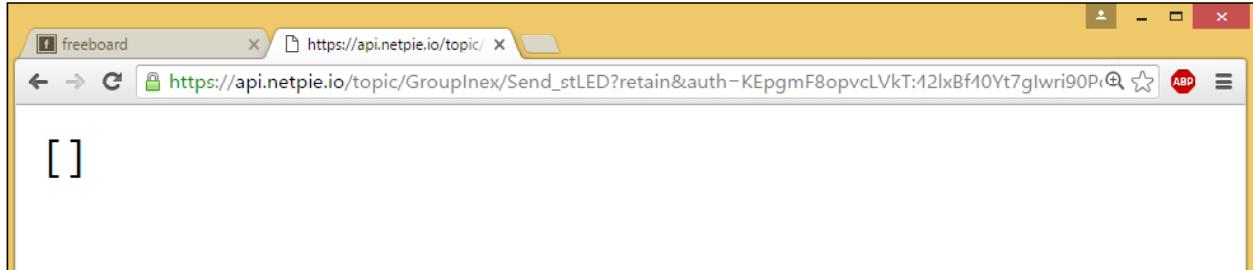
(5.2.2.2) ถ้าหากการส่งและรับค่าลูกศรต้อง อ่านค่าผลลัพธ์ได้ จะปรากฏข้อความต่อไปนี้ที่เว็บбраウเซอร์ ดังรูปที่ 5-12

```
[{"topic":"/GroupInex/Send_stLED", "payload":"1", "qos":0,
"retain":true}]
```



รูปที่ 5-12 การตอบกลับจาก NETPIE ในกรณีที่การทำงานลูกศรต้อง

(5.2.2.3) แต่ถ้าหากส่งค่าไม่ได้ จะแสดงข้อความดังรูปที่ 5-13 ให้กลับไปตรวจสอบสคริปต์ใหม่อีกครั้ง



รูปที่ 5-13 การตอบกลับในกรณีที่ติดต่อกันไม่ได้

### 5.2.3 การเตรียมการที่ฝั่ง NodeMCU-12E

(5.2.3.1) เปิดโปรแกรม Arduino IDE 1.6.5r5 เลือกบอร์ดเป็น NodeMCU1.0 และเลือกพอร์ตเขื่อนต่อให้ถูกต้อง

(5.2.3.2) พิมพ์โปรแกรมที่ 5-1 บันทึกไฟล์ในชื่อ **NETPIELED.ino** และอัปโหลดไปยัง NodeMCU-12E

```
#include <MicroGear.h>
#include <ESP8266WiFi.h>

const char* ssid = <SSID>
const char* password = <PASS>

#define APPID "GroupInex"
#define KEY "Yn1yT7QqyKyYFk6"
#define SECRET "Ksn1hYJ2ELSNjsKb2iiJa6YIg"
#define ALIAS "espLED"

int timer = 0;
int LEDPin = D4;
int st;

WiFiClient client;
MicroGear microgear(client);

void onMsghandler(char *topic, uint8_t* msg, unsigned int msglen)
{
    msg[msglen] = '\0';
    Serial.print("Incoming message -> ");
    Serial.print(topic);
    Serial.print(" : ");
    Serial.println(msg);
}
```

โปรแกรมที่ 5-1 ไฟล์ **NETPIELED.ino** โปรแกรมสำหรับ NodeMCU-12E เพื่อเปิดปิด LED จากการรับค่ามาจากการ **NETPIE** (มีต่อ)

```

Serial.println((char *)msg);
String Topic = topic;
String stateStr = (char *)msg;
if (Topic.equals("/GroupInex/Send_stLED"))
{
    st = stateStr.toInt();
    Serial.println(st);
    digitalWrite(LEDPin, st);
    if (digitalRead(LEDPin) == HIGH)
    {
        microgear.publish("/Feedback_stLED", "1", 1);
    }
    else
    {
        microgear.publish("/Feedback_stLED", "0", 1);
    }
}
void onConnected(char *attribute, uint8_t* msg, unsigned int msglen)
{
    Serial.println("Connected to NETPIE....Now..");
    microgear.subscribe("/Send_stLED");
}
void setup()
{
    Serial.begin(115200);
    microgear.on(MESSAGE, onMsghandler);
    microgear.on(CONNECTED, onConnected);
    Serial.println("Starting...");
    pinMode(LEDPin, OUTPUT);
    if (WiFi.begin(ssid, password))
    {
        while (WiFi.status() != WL_CONNECTED)
        {
            delay(500);
            Serial.print(".");
        }
    }
    Serial.println("WiFi connected" + WiFi.SSID());
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());

    //uncomment the line below if you want to reset token -->
    microgear.resetToken();
    microgear.init(KEY, SECRET, ALIAS);
    microgear.connect(APPID);
}
void loop()
{
    if (microgear.connected())
    {
        microgear.loop();
    }
}

```

โปรแกรมที่ 5-1 ไฟล์ **NETPIELED.ino** โปรแกรมสำหรับ NodeMCU-12E เพื่อเปิดปิด LED จากการรับค่ามาจากการ **NETPIE** (มีต่อ)

```

    else
    {
        Serial.println("connection lost, reconnect...");
        microgear.connect(APPID);
        delay(timer);
        timer += 100;
    }
}

```

### คำอธิบายโปรแกรมเพิ่มเติม

เมื่อโปรแกรมทำงานจะกำหนด event หรือเหตุการณ์ที่ต้องการให้เกิดขึ้น 2 เหตุการณ์คือ เมื่อเชื่อมต่อ NETPIE ได้สำเร็จด้วยคำสั่ง microgear.on (CONNECTED, onConnected) และเมื่อมีการส่งข้อความเข้ามาใน Topic ที่กำหนดสิทธิ์การเข้าถึงไว้ด้วยคำสั่ง microgear.on (MESSAGE, onMsgHandler) จากนั้นเชื่อมต่อ WiFi เมื่อเชื่อมต่อได้สำเร็จจะเชื่อมต่อกับ NETPIE ด้วยคำสั่ง microgear.connect (APPID)

หากเชื่อมต่อได้สำเร็จ ฟังก์ชัน onConnected ก็จะทำงาน ในฟังก์ชันนี้จะมีการกำหนดสิทธิ์เข้าถึงข้อมูลของ Topic ที่ชื่อว่า /Send\_stLED ด้วยคำสั่ง microgear.subscribe ("/Send\_stLED") เมื่อคำสั่งนี้ทำงาน จะเกิดเหตุการณ์ส่งข้อความเกิดขึ้น ส่งผลให้ฟังก์ชัน onMsgHandler ทำงานทันที

การทำงานในฟังก์ชัน onMsgHandler ประกอบด้วย การตรวจสอบ Topic โดยใช้คำสั่ง String.equals () เพื่อตรวจสอบว่า ข้อความในตัวแปร String เป็นข้อความที่ต้องการหรือไม่ ถ้าใช่จะคืนค่าเป็น True ถ้าไม่ใช่ คืนค่าเป็น False จากโปรแกรมใช้คำสั่ง if (Topic.equals ("/GroupInex/Send\_stLED")) ในการตรวจสอบ เมื่อใช้ Topic นี้จะสั่งให้ LED ที่ต่อ กับขา D4 ทำงานด้วยคำสั่ง digitalWrite(LEDPin, st) แล้วอ่านสถานะของขา D4 อีกครั้งพร้อมกับตรวจสอบโดยใช้คำสั่ง if (digitalRead(LEDPin) == HIGH) เพื่อส่งยผลลัพไปยัง NETPIE

#### **หากสถานะขาเป็น HIGH**

จะส่งกลับด้วยคำสั่ง microgear.publish ("/Feedback\_stLED", "1", 1)

#### **ถ้าสถานะขาเป็น LOW**

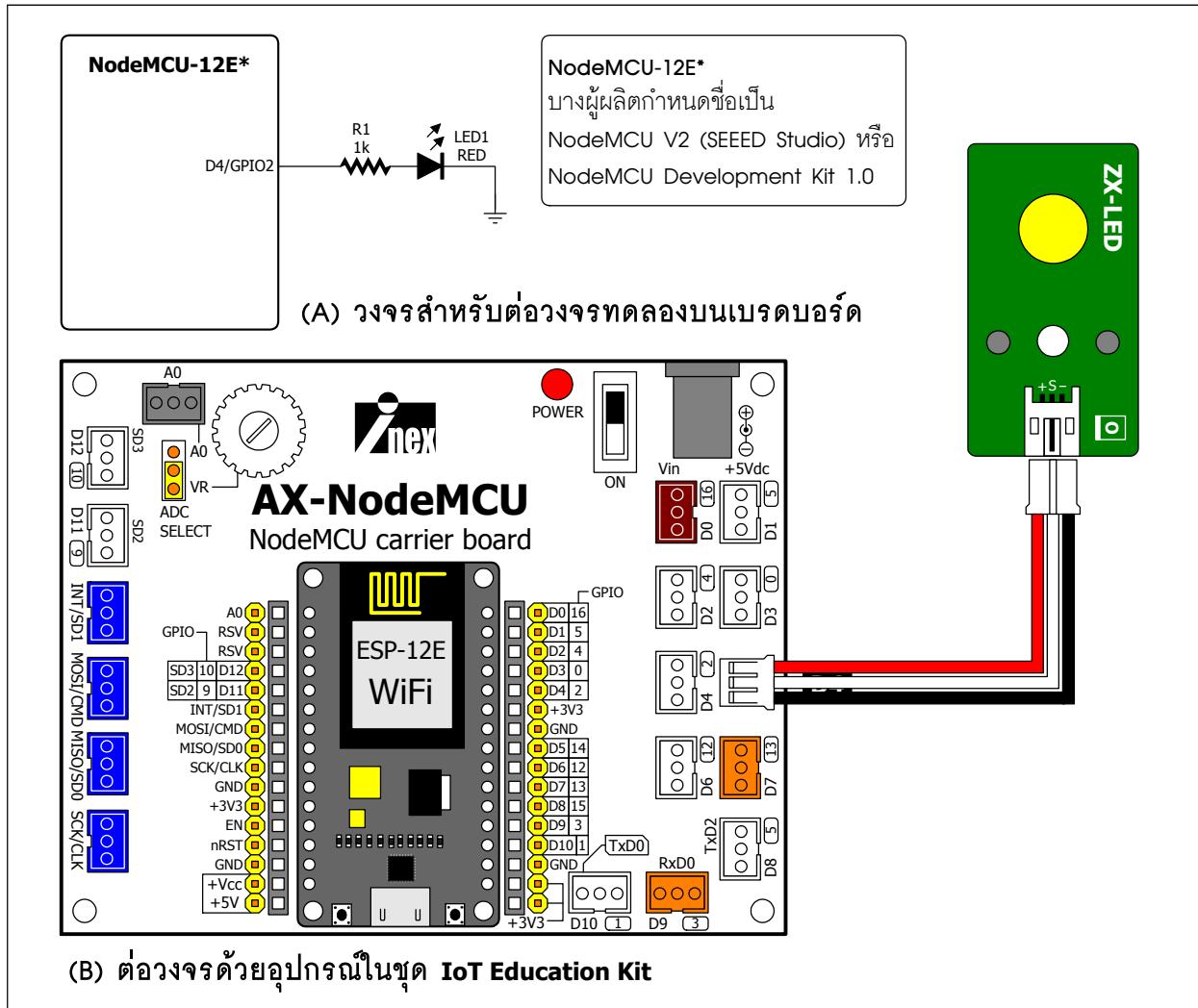
จะตอบกลับด้วยคำสั่ง microgear.publish ("/Feedback\_stLED", "0", 1)

สาเหตุที่ต้องส่งสถานะกลับไป เนื่องจากว่า ถ้าคุ้มกันน์เกิดความเสียหายหรือคุ้มกันน์เชื่อมต่ออินเทอร์เน็ตไม่ได้ ผู้ใช้งานจะสามารถตรวจสอบคุ้มกันน์ได้ด้วย

การทำงานในฟังก์ชัน loop จะคอยตรวจสอบว่า คุ้มกันน์ยังคงเชื่อมต่อกับ NETPIE อยู่หรือไม่ด้วยคำสั่ง if (microgear.connected ())

ถ้าเชื่อมต่ออยู่ ฟังก์ชัน microgear.loop () จะคอยตรวจสอบการเกิดเหตุการณ์ต่างๆ

ถ้าหากเชื่อมต่อ NETPIE ไม่ได้ ก็จะเชื่อมต่อใหม่ด้วยฟังก์ชัน microgear.connect (APPID) อีกครั้ง จากนั้นกระบวนการทำงานก็จะเริ่มต้นใหม่



รูปที่ 5-14 วงจรและการต่อวงจรเพื่อควบคุมการทำงานของ LED ที่ต่อ กับ NodeMCU-12E ผ่าน NETPIE

(5.2.3.3) ต่อ LED เข้าที่ขาพอร์ต D4 ดังรูปที่ 5-14

(5.2.3.4) ตรวจสอบสถานะการตอบกลับของ NodeMCU-12E โดยเปิดเว็บเบราว์เซอร์แล้วป้อน URL ต่อไปนี้ไปยังแอ็คเดรสบาร์

```
https://api.NETPIE.io/topic/GroupInex/Feedback_stLED?
retain& auth= YnlyT7QqyKyYFk6:Ksn1hYJ2ELSNjsKb2iiJa6YIg
```

(5.2.3.5) หากการเชื่อมต่อถูกต้อง จะได้รับข้อความตอบกลับดังรูปที่ 5-15

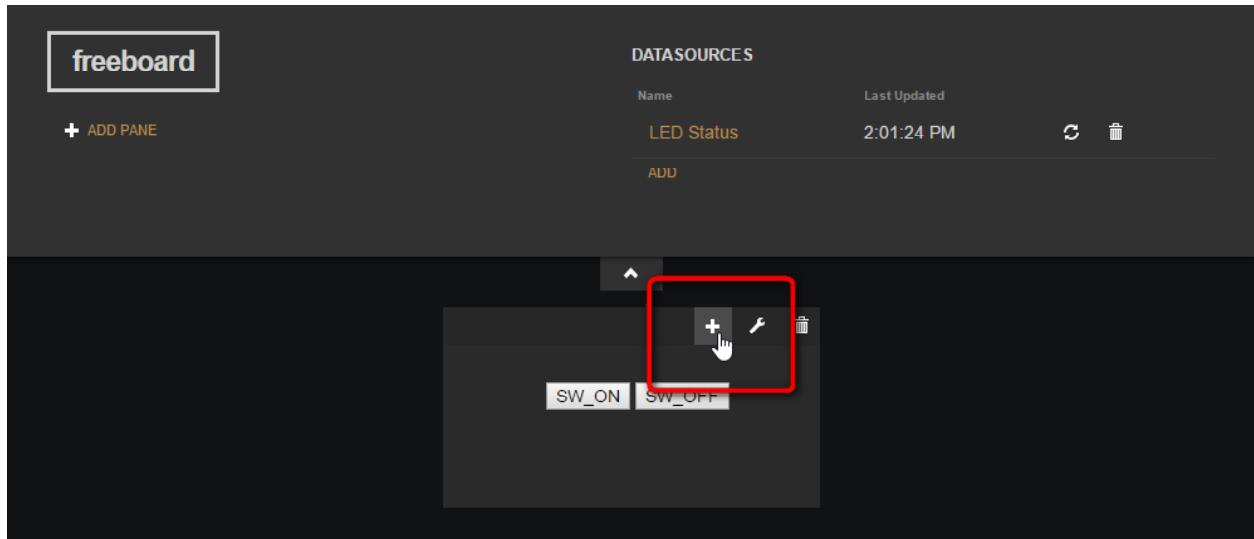


รูปที่ 5-15 แสดงการตอบกลับจาก NETPIE เมื่อได้รับข้อความจาก NodeMCU-12E

### 5.2.4 เพิ่มอุปกรณ์แสดงผล

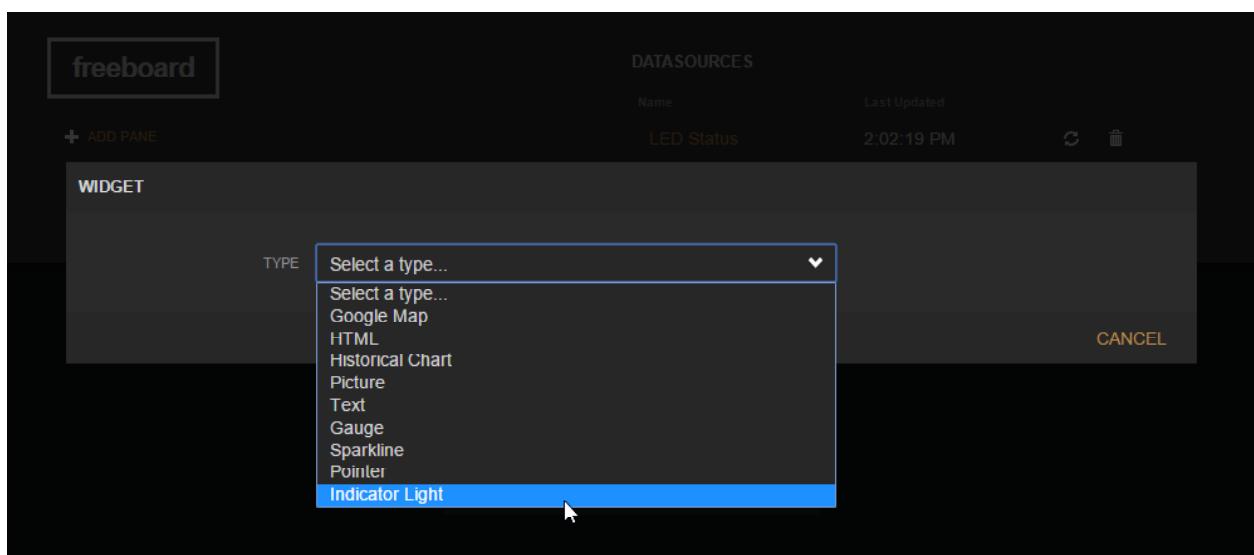
กลับไปที่แดชบอร์ด LED Status ที่ freeboard อีกรั้งเพื่อเพิ่มอุปกรณ์แสดงผล มีขั้นตอนดังนี้

(5.2.4.1) เพิ่มอุปกรณ์แสดงผลหรือวิดเก็ต (widget) ที่ใช้แสดงสถานะการทำงานของ LED โดยคลิกที่เครื่องหมาย + ดังรูปที่ 5-16



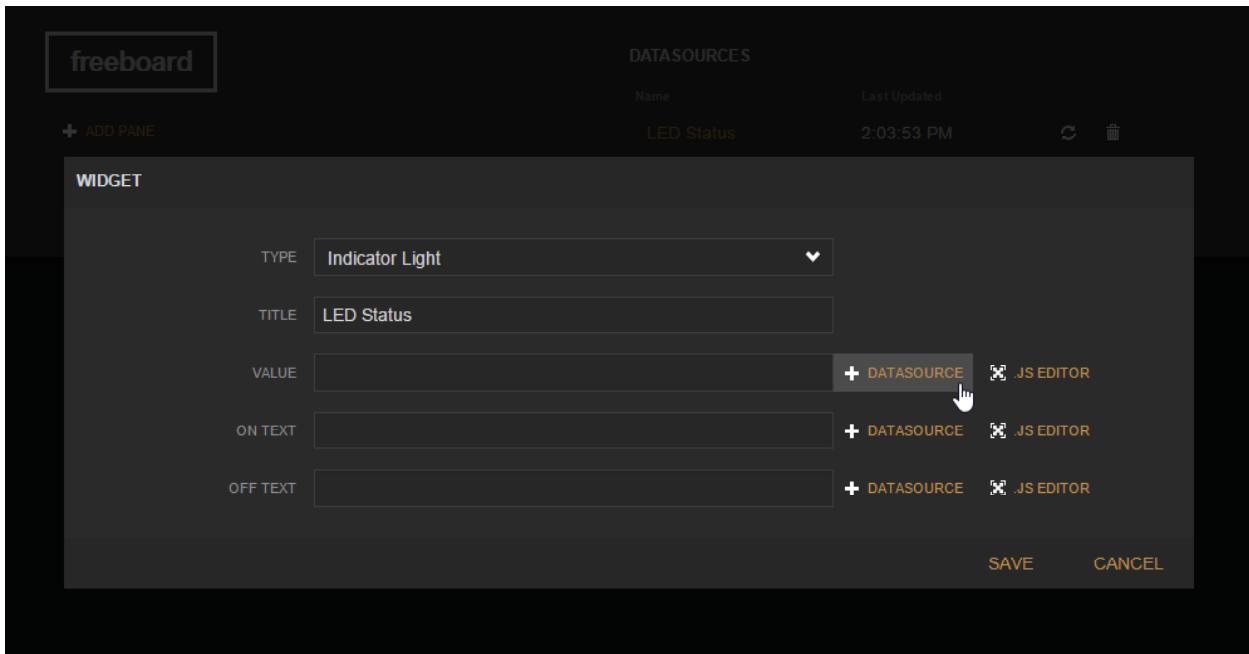
รูปที่ 5-16 เพิ่มอุปกรณ์แสดงผลให้แก่แดชบอร์ด

(5.2.4.2) เลือกชนิด (TYPE) ของอุปกรณ์แสดงผลหรือวิดเก็ตในรูปแบบ Indicator Light ดังรูปที่ 5-17



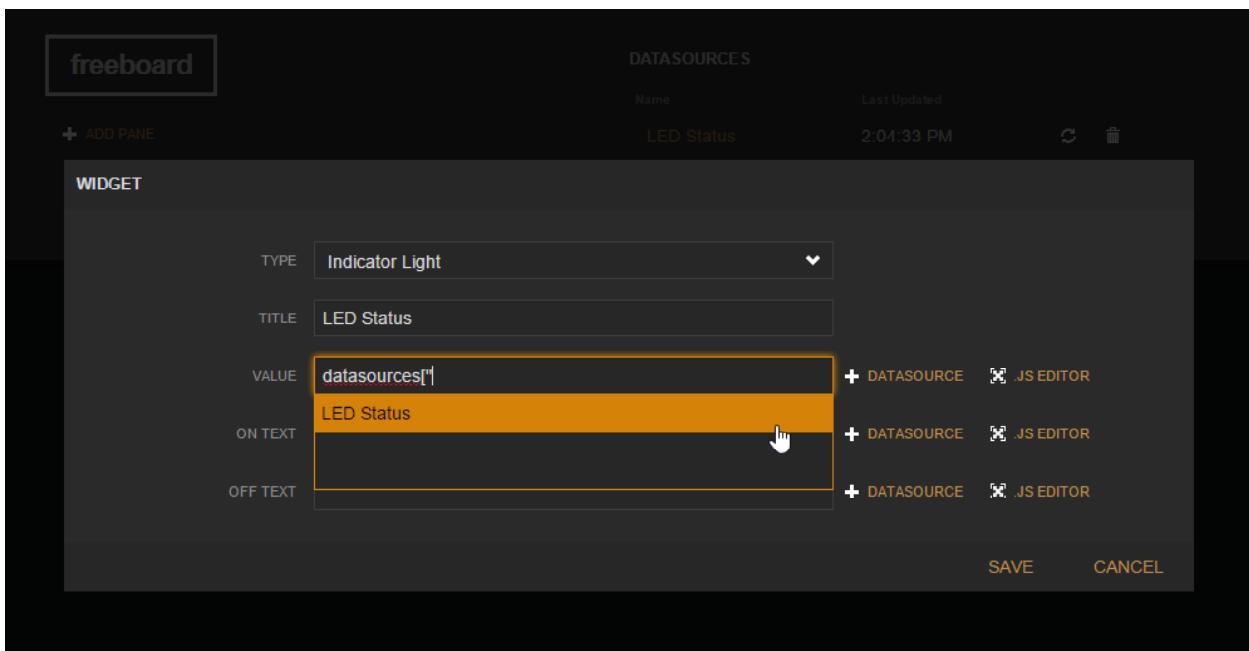
รูปที่ 5-17 เลือกชนิดของอุปกรณ์แสดงผลเป็น Indicator Light สำหรับแดชบอร์ด LED Status

(5.2.4.3) จากนั้นกำหนดรายละเอียดขั้นต้นตามรูปที่ 5-18



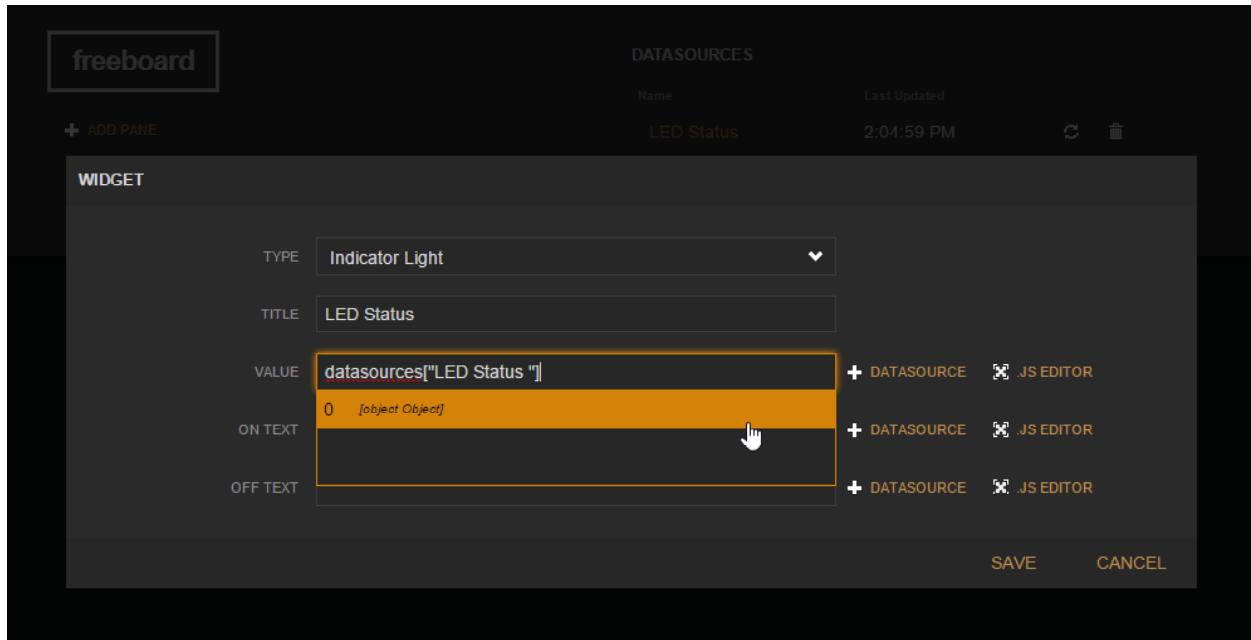
รูปที่ 5-18 ทำการตั้งค่าให้กับ Indicator Light ที่ใช้เป็นตัวแสดงผลของแดชบอร์ด LED Status

(5.2.4.4) เลือกแหล่งข้อมูล คลิกที่ DATASOURCE เลือก LED Status ดังรูปที่ 5-19



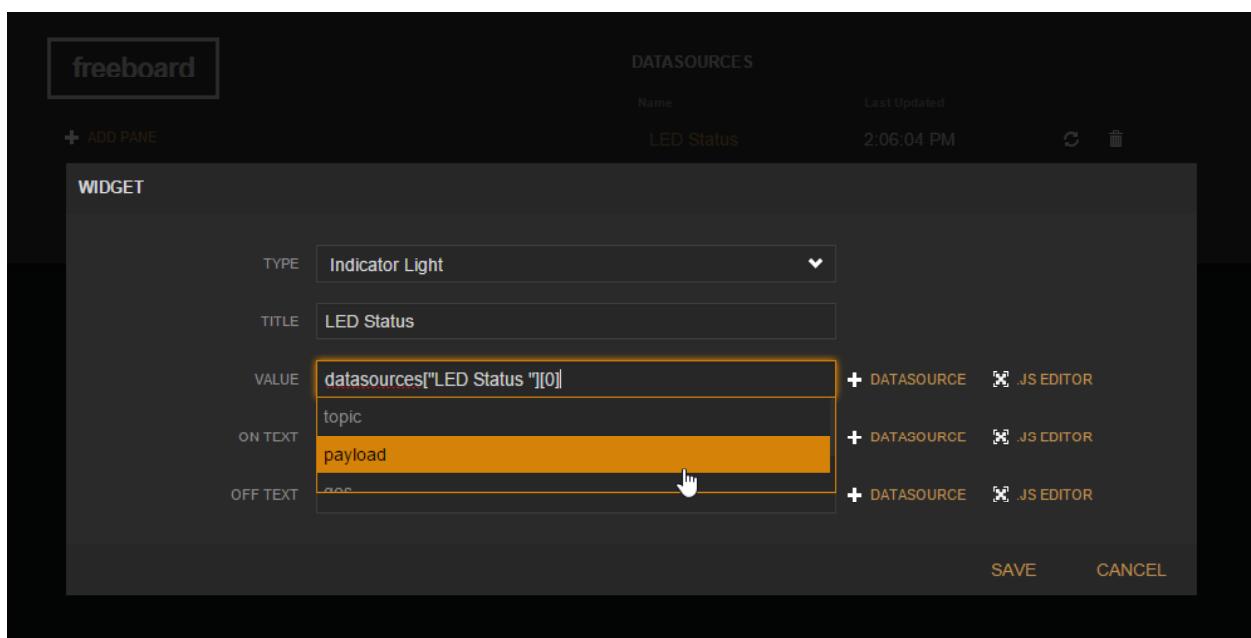
รูปที่ 5-19 เลือกแหล่งข้อมูลเป็น LED Status ที่ช่อง VALUE ของ DATASOURCE ในอุปกรณ์แสดงผลของแดชบอร์ด LED Status

(5.2.4.5) ตามด้วยเลือกชนิดข้อมูล **0 [object Object]** ดังรูปที่ 5-20



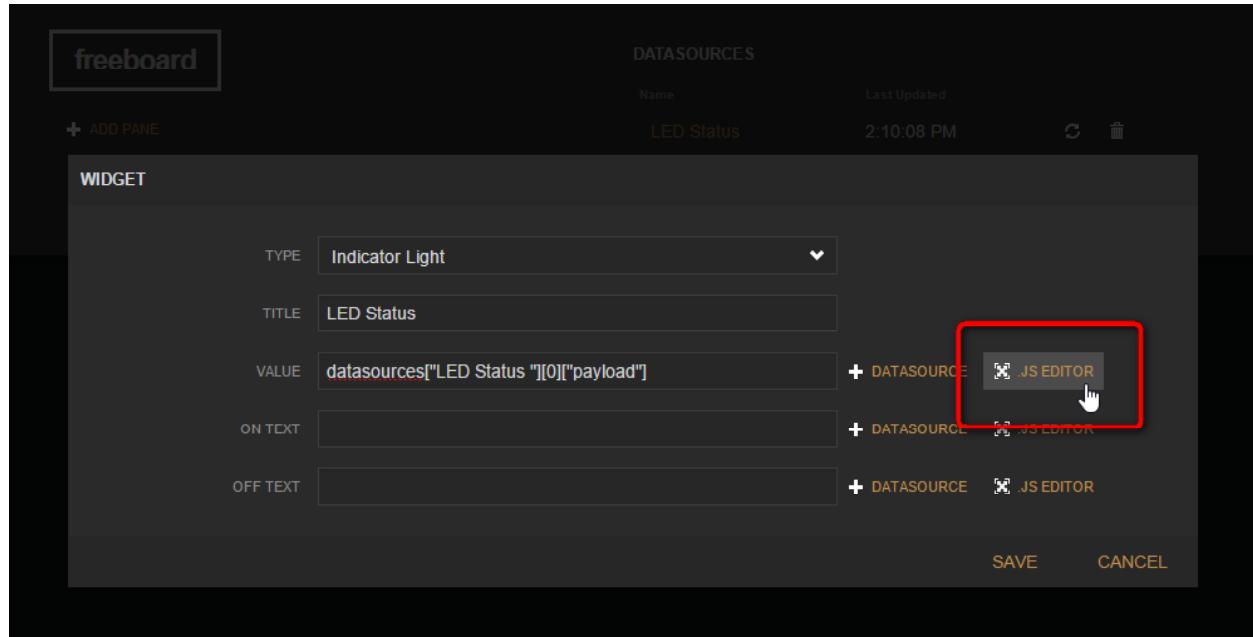
รูปที่ 5-20 เลือกชนิดข้อมูลของ LED Status ที่ช่อง VALUE ของ DATA SOURCE

(5.2.4.6) แล้วเลือกชนิดข้อมูล payload ดังรูปที่ 5-21



รูปที่ 5-21 เลือกชื่อหัวข้อการติดต่อเป็น payload ที่ช่อง VALUE ของ DATA SOURCE

(5.2.4.7) ทำการแก้ไข DATASOURCE โดยคลิกที่ **JS EDITOR** เพื่อเปิดหน้าต่างเอดิเตอร์ของJAVAสคริปต์ ดังรูปที่ 5-22



รูปที่ 5-22 การเลือกเปิดหน้าต่าง JS editor เพื่อแก้ไขข้อมูลของ DATASOURCE ที่เลือกมาใช้งาน

(5.2.4.8) หน้าต่าง JS editor ปรากฏขึ้น พร้อมกับแสดงชื่อ VALUE ของ DATASOURCE ที่ต้องการแก้ไข ดังรูปที่ 5-23

```
This javascript will be re-evaluated any time a datasource referenced here is updated, and the value you return will be displayed in the widget. You can assume this javascript is wrapped in a function of the form function(datasources) where datasources is a collection of javascript objects (keyed by their name) corresponding to the most current data in a datasource.

1 datasources["LED Status "][@][ "payload"]
```

รูปที่ 5-23 หน้าต่าง JS editor หรือJAVAสคริปต์เอดิเตอร์สำหรับแก้ไขข้อมูลของแหล่งกำเนิดข้อมูลที่เลือกมาใช้งาน จากในรูปคือ LED Status และ payload

(5.2.4.9) เพิ่มและแก้ไขสคริปต์คำสั่งดังต่อไปนี้

```
return(parseInt(datasources["LED Status "][0]["payload"]))
```

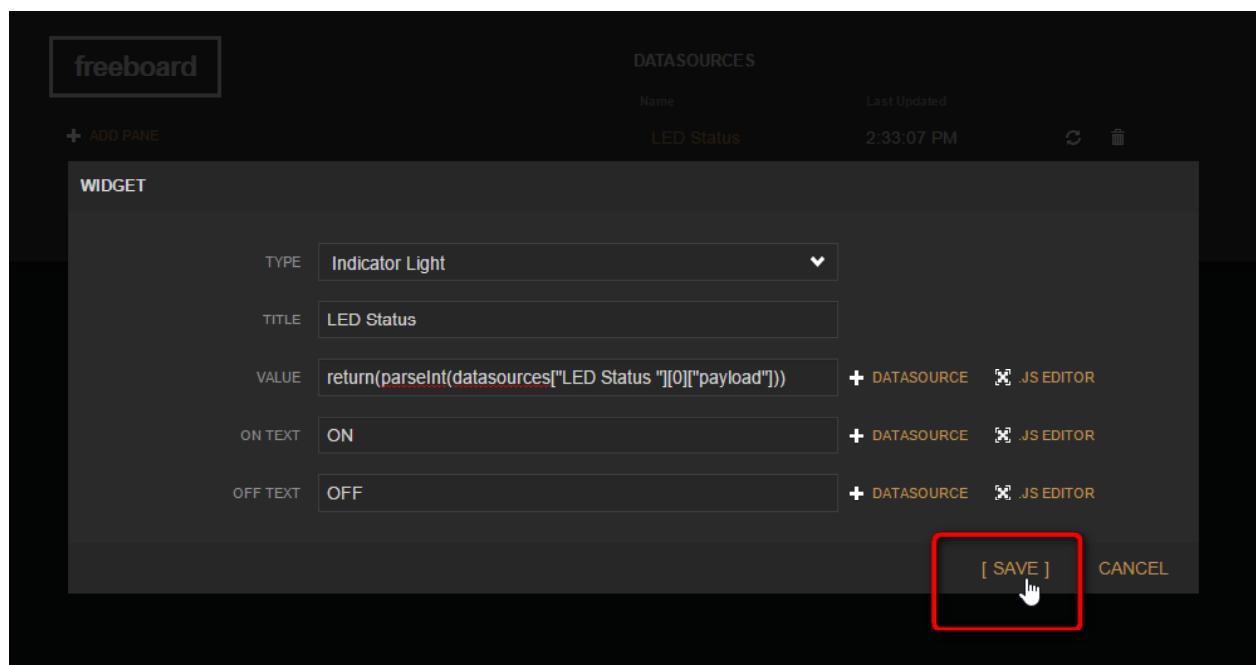
ถ้าสั่งเก็ตจากการตอบกลับของ NETPIE ค่าของ payload จะให้ผลลัพธ์ในรูปแบบ String แต่เนื่องจากค่า Value ที่ต้องการคือ True กับ False หรือ 0 กับ 1 ในรูปแบบ Integer ดังนั้นการแก้ไขสคริปต์จึงเป็นการแปลงค่าจากตัวแปร String เป็น Integer

(5.2.4.10) กลับมาที่หน้าต่างตั้งค่า ให้เพิ่มรายละเอียดในส่วนที่เหลือ

ที่ช่อง ON TEXT เลือกเป็น ON

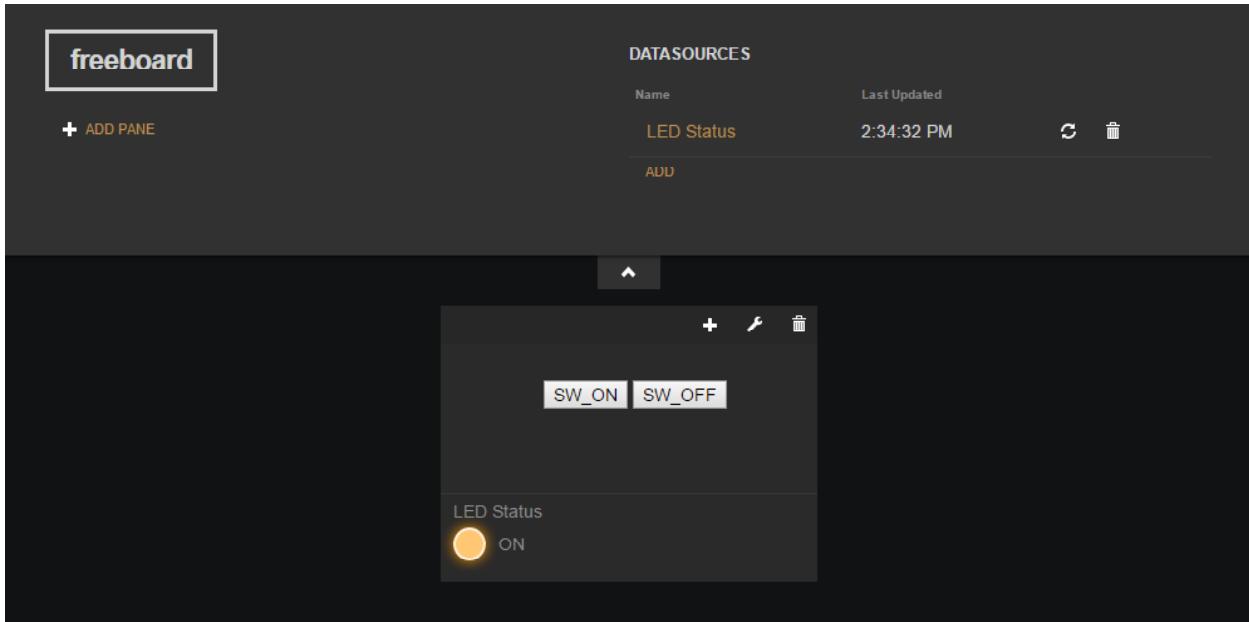
ที่ช่อง OFF TEXT เลือกเป็น OFF

แล้วคลิก SAVE ดังรูปที่ 5-24



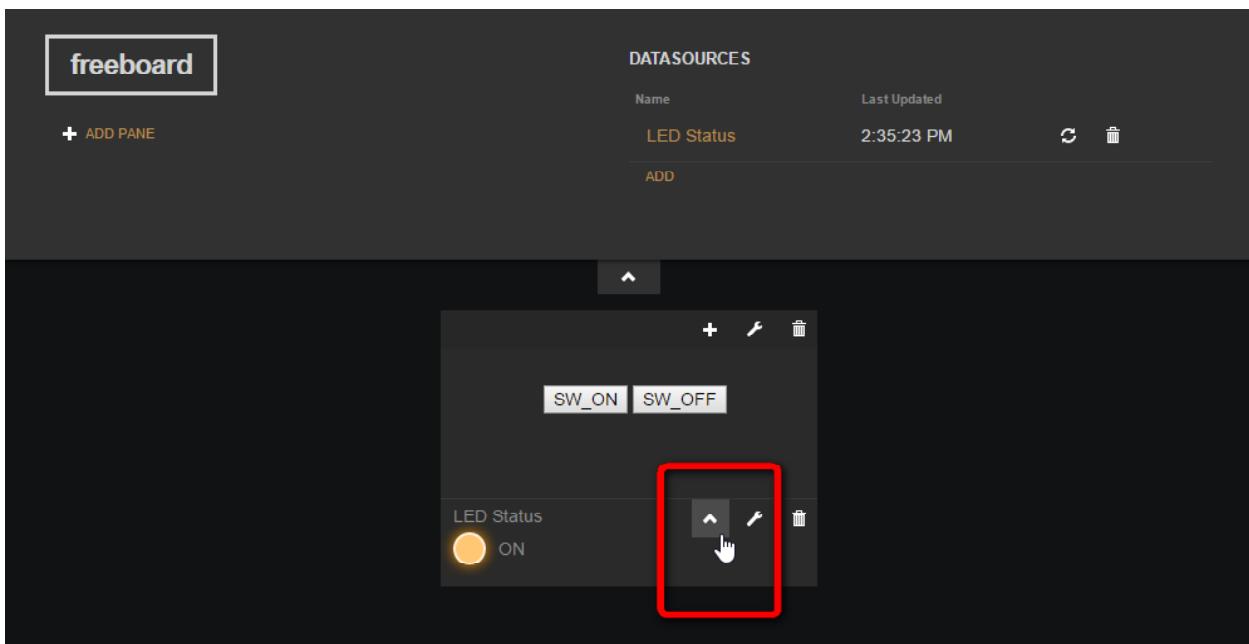
รูปที่ 5-24 หน้าต่างแสดงบทสรุปของการตั้งค่าสำหรับอุปกรณ์แสดงผลที่ชื่อ LED Status สำหรับใช้ในแดชบอร์ดควบคุม LED

(5.2.4.11) จะได้แดชบอร์ดของกราฟิกควบคุม LED ผ่านอินเทอร์เน็ตดังรูปที่ 5-25



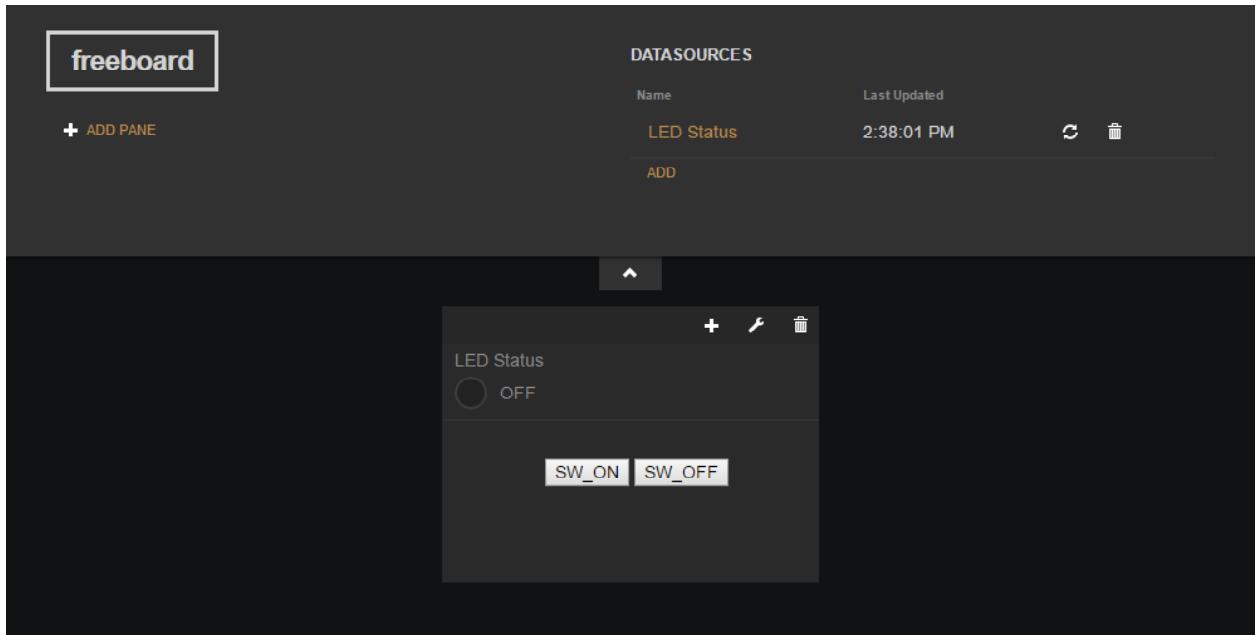
รูปที่ 5-25 แดชบอร์ดสำหรับควบคุม LED ผ่าน NETPIE บนเว็บไซต์ freeboard.io เมื่อสร้างเสร็จ

(5.2.4.12) ถ้าต้องการให้ LED เลื่อนขึ้นไปด้านบน ให้คลิกที่ลูกศรดังรูปที่ 5-26 จนได้ตำแหน่งตามต้องการ



รูปที่ 5-26 แสดงการปรับแต่งตำแหน่งของรูป LED ด้วยปุ่มลูกศรบนแดชบอร์ด LED Status

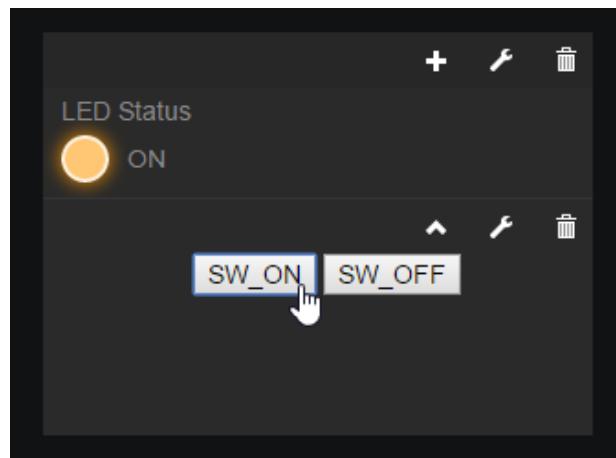
จะได้เดชบอร์ดสำหรับควบคุม LED ผ่านอินเทอร์เน็ต โดยใช้ NETPIE เป็นคลาวด์เซิร์ฟเวอร์ และแสดงผลด้วย freeboard และควบคุมอุปกรณ์ด้วย NodeMCU-12E ดังรูปที่ 5-27



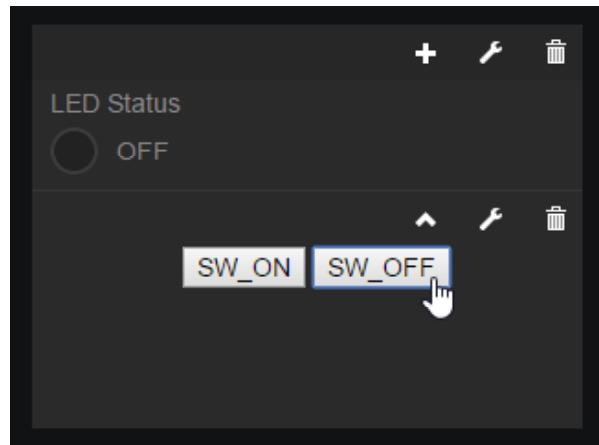
รูปที่ 5-27 NETPIE LED เดชบอร์ดที่สร้างขึ้นเพื่อควบคุม LED ที่ต่อกับขา D4 ของ NodeMCU-12E โดยทำงานผ่าน NETPIE และแสดงผลด้วย freeboard.io

### 5.3 ทดสอบการทำงานทั้งระบบ

(5.3.1) คลิกที่ปุ่ม SW\_ON จะทำให้ LED ติดเป็นสีเหลือง และ LED ที่ต่อกับขา D4 ของ NodeMCU-12E จะติดสว่างตามด้วย การติดของ LED อาจซักว่าการแสดงผลที่หน้าเดชบอร์ด ขึ้นอยู่กับความเร็วของการสื่อสารข้อมูลบนเครือข่ายอินเทอร์เน็ต



(5.3.2) คลิกที่ปุ่ม SW\_OFF จะทำให้ LED เปลี่ยนเป็นสีขาว แทนการดับ และ LED ที่ต่อ กับขา D4 ของ NodeMCU-12E จะดับลงตามคิว การดับของ LED อาจช้ากว่าการแสดงผลที่หน้าจอ ขบอร์ด ขึ้นอยู่กับความเร็วของการสื่อสารข้อมูลบนเครือข่ายอินเทอร์เน็ต เช่นกัน



# บทที่ 6

## การใช้งาน NETPIE Freeboard

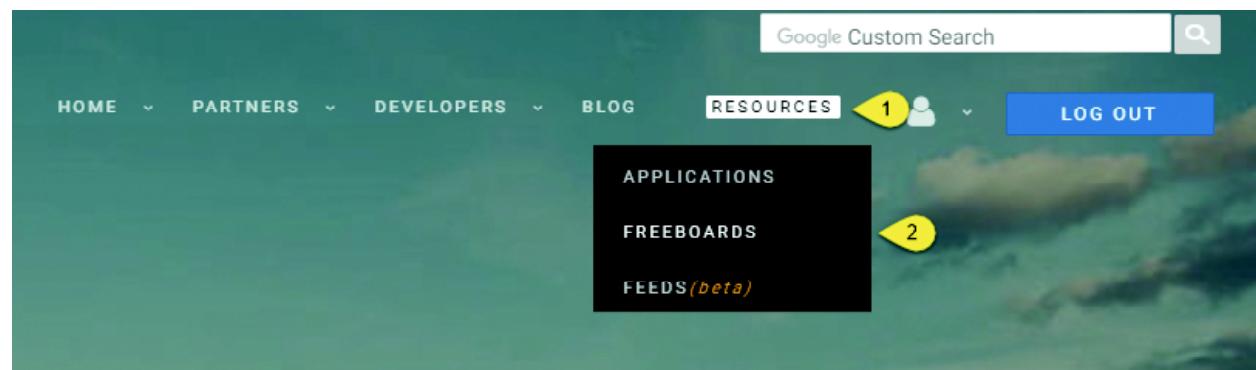
จากที่นำเสนอมาในบทที่ 3 ถึง 5 NETPIE ทำหน้าที่เป็นคลาวเซอร์ฟเวอร์หลักและใช้ REST API และ Freeboard.io ในการติดต่อกับส่วนแสดงผลหรือแดชบอร์ดตัวอื่น ในการปรับปรุง NETPIE ตั้งแต่เดือนพฤษภาคม พ.ศ. 2560 เป็นต้นมา คณะพัฒนา NETPIE ได้ทำการพนวกส่วนแสดงผลหรือ แดชบอร์ด Freeboard รวมเข้าไปในระบบของ NETPIE เนื่องจาก Freeboard เป็นแดชบอร์ดแบบ โอเพ่นซอร์สและเปิดโอกาสให้ผู้พัฒนาจากทั่วโลกสามารถนำโค้ดไปพัฒนาต่อได้ด้วย

ในบทนี้จึงขอนำเสนอแนวทางและตัวอย่างการใช้งาน NETPIE-Freeboard เพื่อทำให้ NETPIE เป็นคลาวด์เซอร์ฟเวอร์ที่มีความสมบูรณ์มีส่วนแสดงผลเป็นของตัวเอง

### 6.1 การใช้งานและตั้งค่า NETPIE Freeboard ผ่านทางหน้าเว็บ NETPIE

เดิมที่ NETPIE ไม่มีส่วนของหน้าปัดแสดงผลหรือแดชบอร์ด ผู้พัฒนาจะต้องกำหนดค่าเพื่อ เชื่อมต่อ NETPIE เข้ากับเว็บเพจที่ทำหน้าที่เป็นหน้าปัดแสดงผลหรือแดชบอร์ดเอง แดชบอร์ดตัวหนึ่ง ที่ได้รับความนิยมสูงคือ **Freeboard** ดังนั้นเพื่ออำนวยความสะดวกแก่ผู้พัฒนาและใช้งาน NETPIE คณะผู้พัฒนา NETPIE จึงได้เพิ่มเติมความสามารถของ NETPIE ให้เชื่อมต่อและใช้งานกับ Freeboard ได้สะดวกขึ้น มีขั้นตอนดำเนินการดังนี้

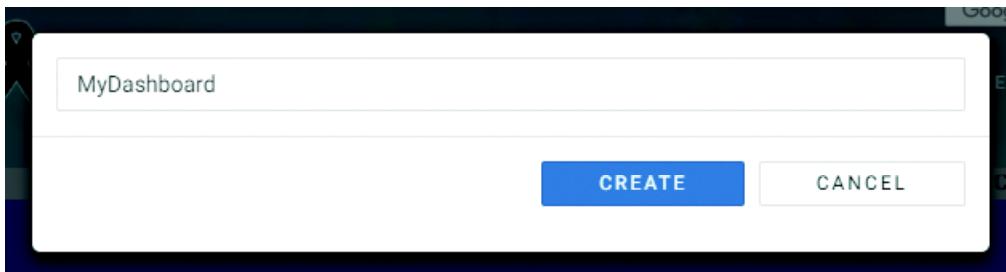
(6.1.1) เข้าสู่ระบบ NETPIE ทำการ Log in และเลือกเมนู **RESOURCES > FREEBOARDS** ตามรูปที่ 6-1



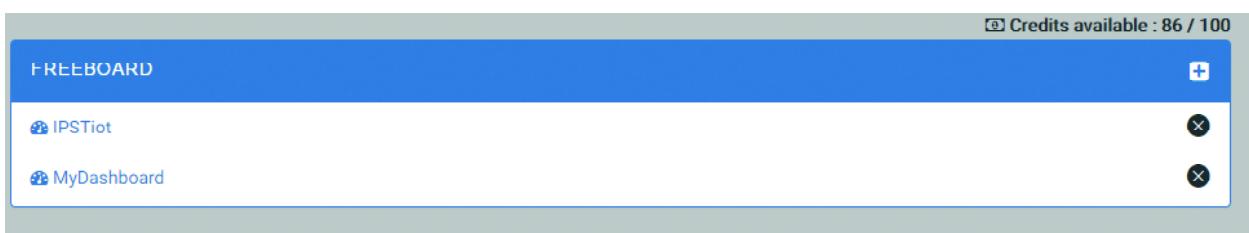
รูปที่ 6-1 การเลือกเข้าสู่เมนูของ Freeboard



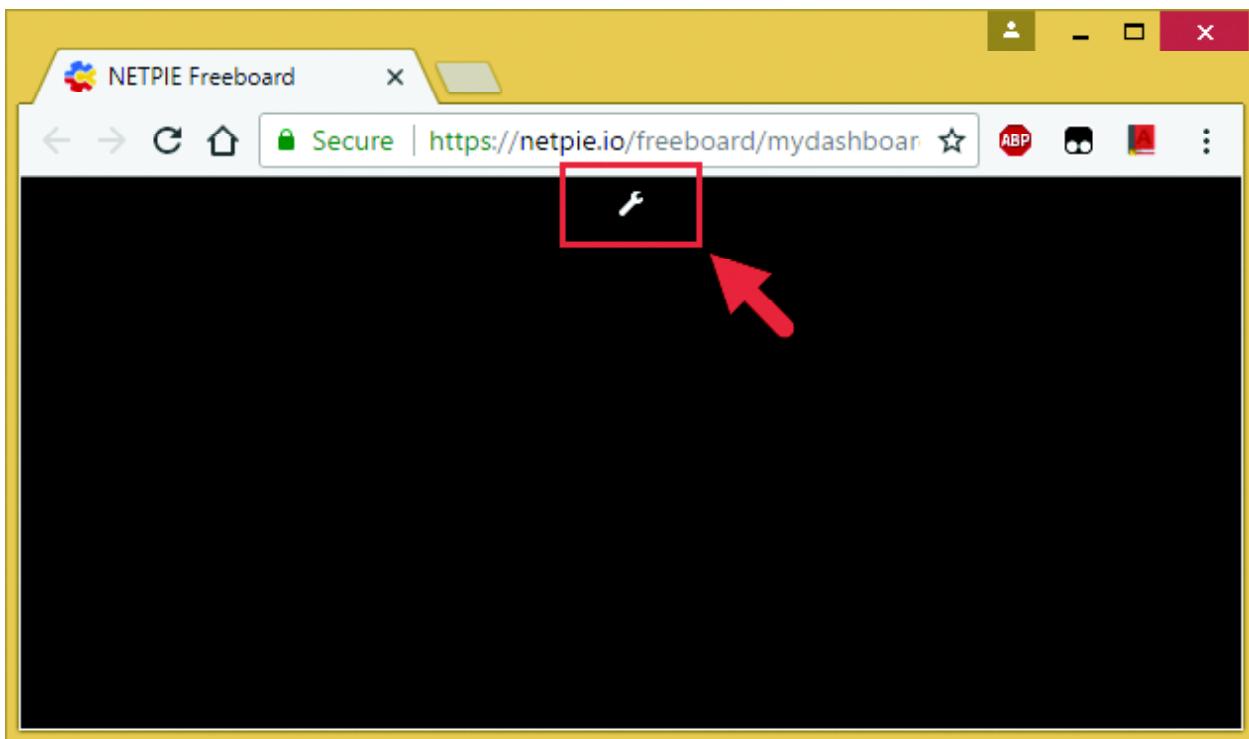
รูปที่ 6-2 เริ่มต้นสร้างแดชบอร์ดใหม่บน Freeboard



รูปที่ 6-3 กำหนดชื่อของหน้าปัดสำหรับแสดงผลและเริ่มต้นการสร้างด้วยการคลิกปุ่ม CREATE



รูปที่ 6-4 รายชื่อของส่วนแสดงผลของ Freeboard ที่มีอยู่



รูปที่ 6-5 ปุ่มตั้งค่าของ Freeboard

(6.1.2) คลิกเครื่องหมาย + เพื่อสร้างส่วนแสดงผลใหม่ด้วย Freeboard ขึ้นมา ดังรูปที่ 6-2

(6.1.3) ตัวชี้อ Freeboard และคลิกปุ่ม CREATE ดังรูปที่ 6-3

(6.1.4) เมื่อคลิกปุ่ม CREATE จะมีรายชื่อ Freeboard ที่เคยสร้างไว้ ดังรูปที่ 6-4 หากต้องการเข้าไปตั้งค่าหรือใช้งาน ให้คลิกที่ชื่อของหน้าปัด Freeboard ที่ต้องการได้ทันที

(6.1.5) หากไม่แสดงรายการใดๆ ให้เห็น คลิกที่ปุ่มตั้งค่าดังรูปที่ 6-5 จะมีรายการพารามิเตอร์ทั้งหมดที่ต้องทำการตั้งค่าปรากฏขึ้นมา

(6.1.6) ตัวอย่างการตั้งค่าเพื่อใช้งาน Freeboard

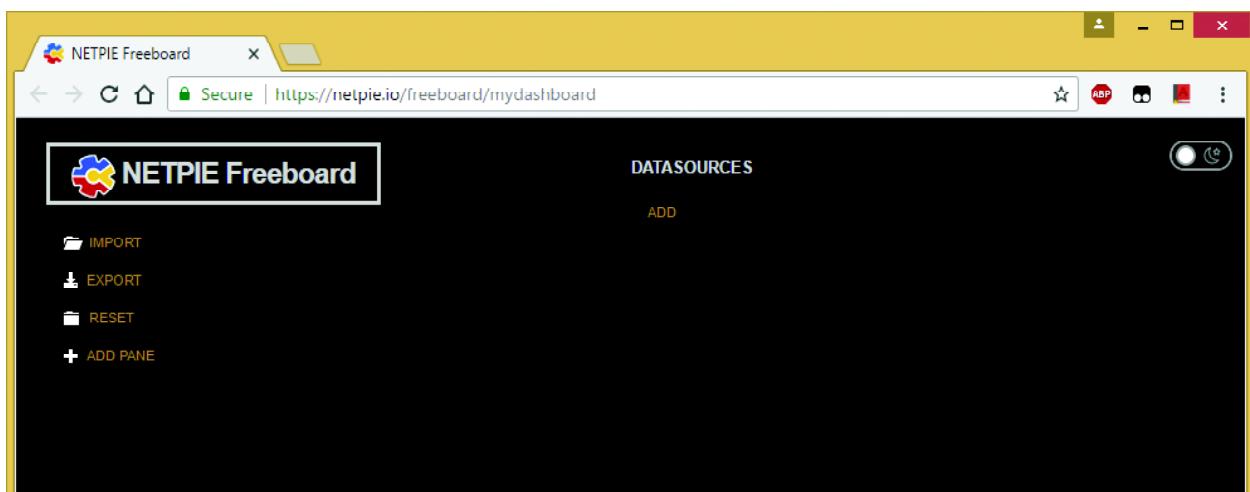
### Main Menu

- **IMPORT** เป็นเมนูสำหรับนำไฟล์ที่เก็บค่าคุณสมบัติหรือ Configuration ของหน้า Freeboard ที่บันทึกเก็บไว้
- **EXPORT** เป็นเมนูสำหรับนำไฟล์ Configuration ออกไปเก็บไว้
- **RESET** เป็นเมนูสำหรับล้าง Datasource และ Widget ที่สร้างไว้
- **ADD PANE** เป็นเมนูสำหรับเพิ่ม Panel หรือหน้าปัดสำหรับวาง Widget

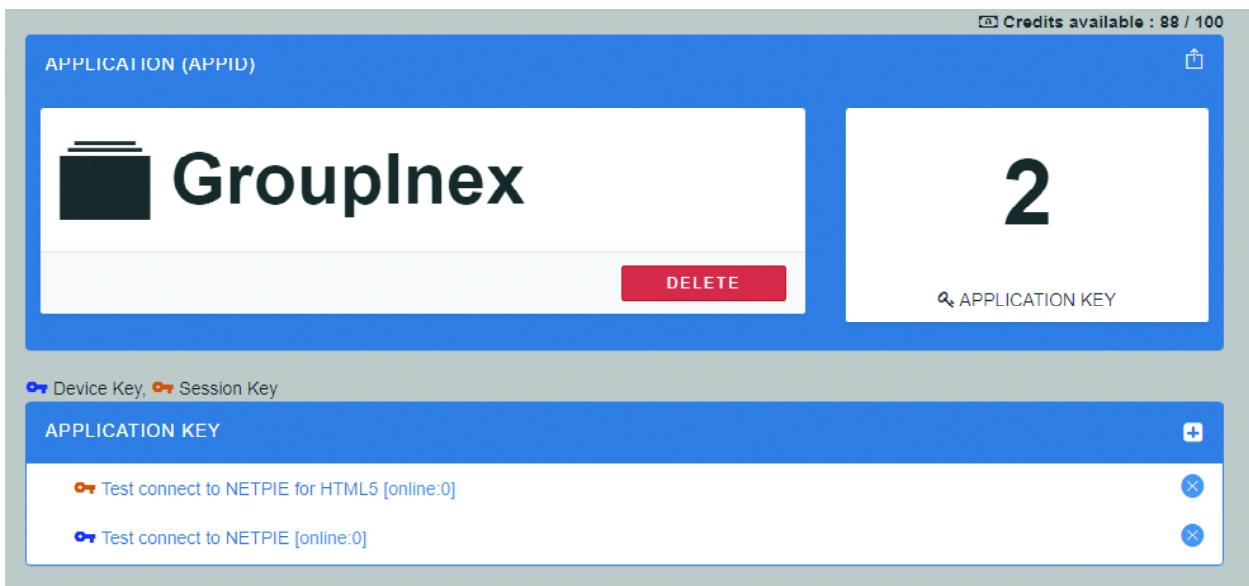
### Datasources Menu

- **ADD** เป็นเมนูสำหรับเพิ่มแหล่งข้อมูลที่ต้องการนำมาแสดงผล

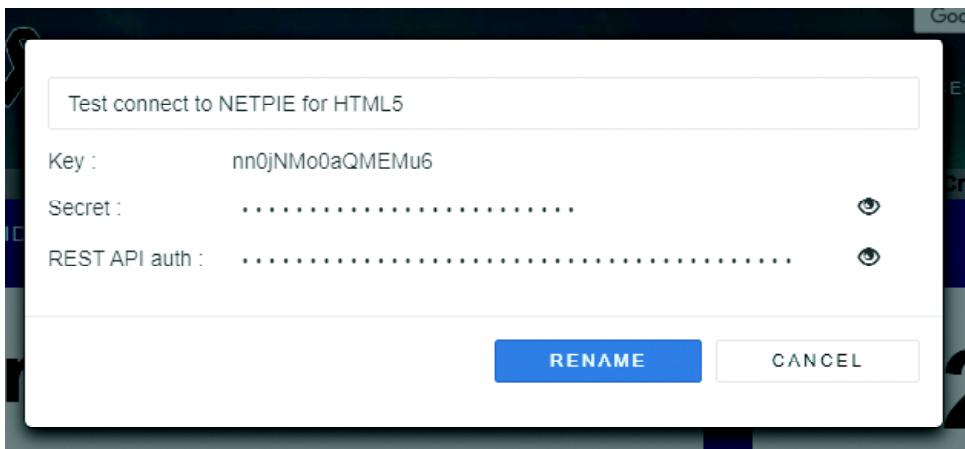
(6.1.7) เพิ่มแหล่งข้อมูลหรือ Datasources โดยก่อนที่จะตั้งค่าเพื่อดึงข้อมูลจาก NETPIE มาแสดงที่ Freeboard จะต้องเตรียมรหัสกุญแจหรือ Key ที่ใช้สำหรับ HTML5 หรือที่เรียกว่า Session Key เสียก่อน ในตัวอย่างนี้ได้สร้างรหัสกุญแจไว้ทั้ง 2 ประเภทและอยู่ภายใต้ AppID เดียวกัน ดังรูปที่ 6-6



รูปที่ 6-6 แสดงหน้าต่างสำหรับตั้งค่าของ Freeboard



รูปที่ 6-7 แสดงให้เห็นว่า มีการสร้างรหัสกุญแจทั้ง 2 ประเภทไว้แล้ว ทั้ง Device Key และ Session Key

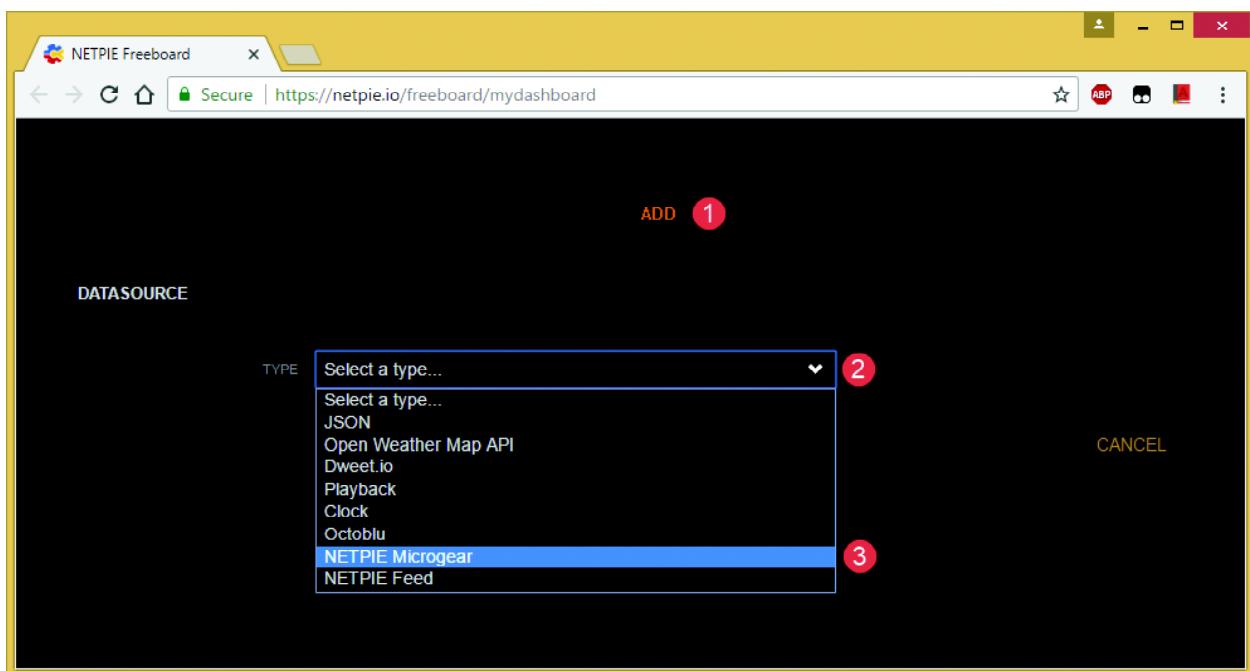


รูปที่ 6-8 แสดงหน้าต่างของ Application Key ที่ชื่อว่า Test connect to NETPIE for HTML5

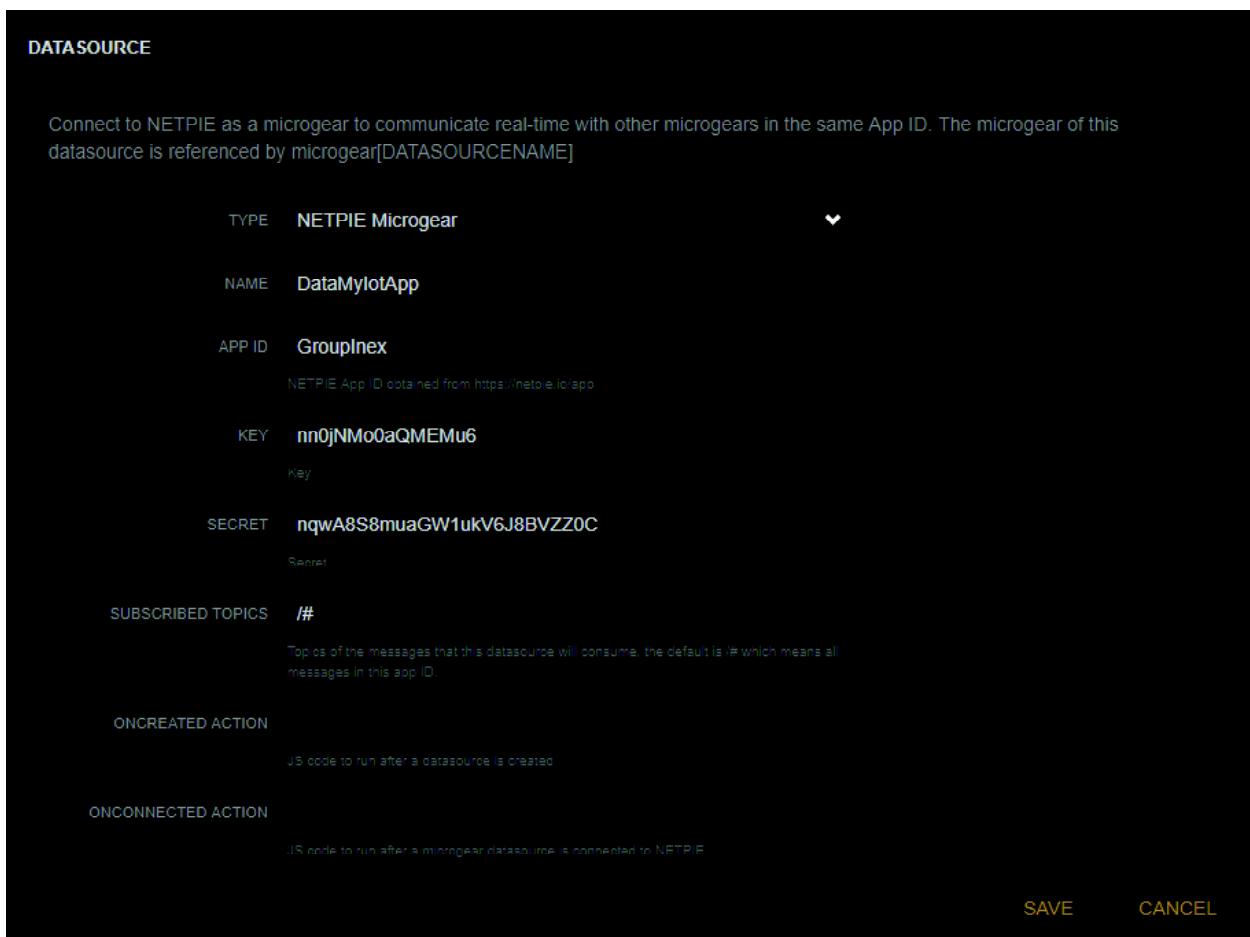
(6.1.8) จากรูปที่ 6-7 เลือกใช้ Key ที่มีชื่อว่า Test connect to NETPIE for HTML5 ! เพราะสร้างรหัสกุญแจเป็นแบบ Session Key เอ้าไว้ จากนั้นคลิกที่ชื่อ Key แล้วนำข้อมูลไปกรอกในขั้นตอนต่อไป ดังรูปที่ 6-8

(6.1.9) เพิ่ม Datasources โดยคลิกปุ่ม ADD (วงกลม 1 ในรูปที่ 6-9) จากนั้นคลิกที่ช่อง TYPE เพื่อเลือกชนิดของแหล่งข้อมูล (วงกลม 2 ในรูปที่ 6-9) และเลือกรายการ NETPIE Microgear (วงกลม 3 ในรูปที่ 6-9)

(6.1.10) กำหนดชื่อ Datasources ตามต้องการ ในช่อง NAME ตามด้วย APPID, KEY, SECRET ที่ได้จากการสร้างรหัสกุญแจแบบ Session Key และช่อง SUBSCRIBERD TOPIC ใส่/# หมายถึง ทุกการส่งข้อมูลที่อยู่ภายใต้ AppID นี้จะเห็นข้อมูลทั้งหมด ดังรูปที่ 6-10



รูปที่ 6-9 แสดงลำดับในการเพิ่ม Datasources ให้กับ Freeboard



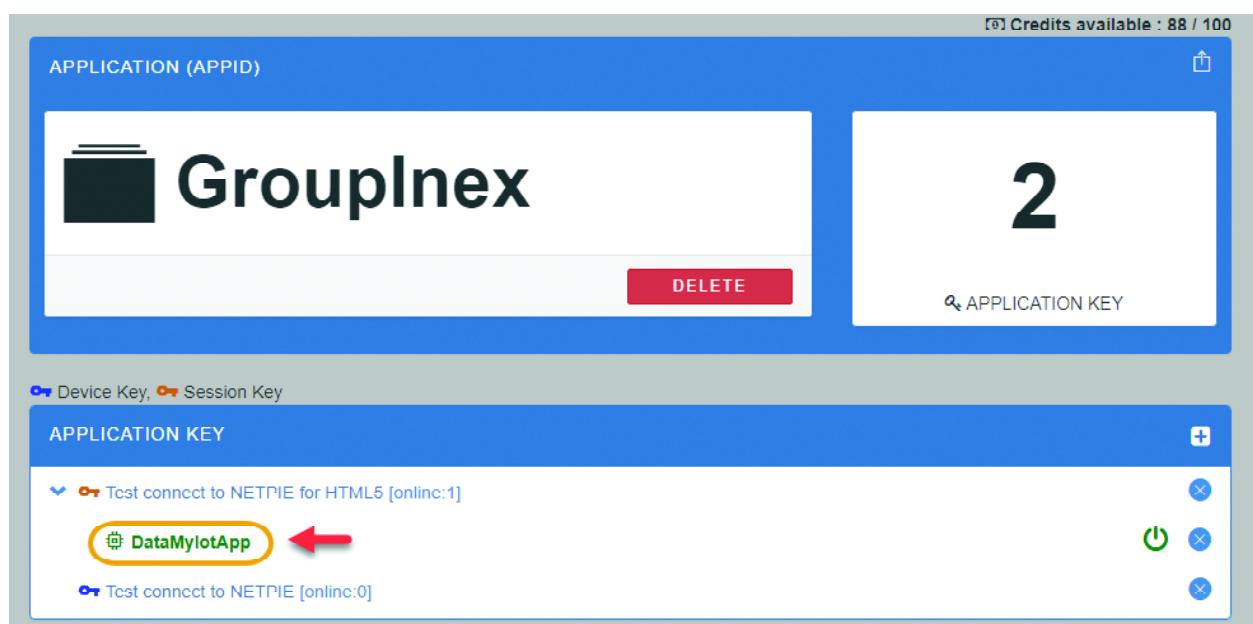
รูปที่ 6-10 กำหนดข้อและรหัสกุญแจที่เกี่ยวข้องกับ Datasources

DATASOURCES		
Name	Last Updated	
DataMyIotApp	never	 
<a href="#">ADD</a>		

รูปที่ 6-11 หน้าต่างแสดงการเกิดขึ้นใหม่ของแดชบอร์ดที่ชื่อ DataMyIotApp

(6.1.11) เมื่อกรอกข้อมูลครบเรียบร้อยแล้ว ให้คลิกปุ่ม SAVE รายชื่อ DataSources ที่สร้างขึ้นใหม่จะปรากฏขึ้นมา ในที่นี้คือ DataMyIotApp ดังรูปที่ 6-11

(6.1.12) กลับไปดูที่ APPLICATION MANAGEMENT จะเห็นชื่อ DataSources ที่สร้างขึ้นมา โดยปรากฏเป็นชื่ออุปกรณ์ที่ใช้รหัสกุญแจแบบ Session Key นั่นคือ DataIotApp ดังรูปที่ 6-12



รูปที่ 6-12 แสดงการเกิดขึ้นใหม่ของอุปกรณ์ที่ใช้รหัสกุญแจแบบ Session Key ที่ชื่อ DataIotApp

## 6.2 การใช้งานวิดเก็ตหรืออุปกรณ์แสดงผลต่างๆ บน NETPIE Freeboard

### 6.2.1 ใช้งาน Gauge

Gauge เป็นอุปกรณ์แสดงผลที่บ่งบอกขนาดข้อมูลโดยใช้แบบสี ใช้วัดว่า ข้อมูลที่อ่านเข้ามา ถึงจุดสูงสุดหรือยัง ก่อนที่จะเพิ่ม Gauge เข้าไปในการแสดงผล ผู้พัฒนาจะต้องส่งค่าขึ้นไปเก็บไว้บน NETPIE เสียก่อน มีนะนั้น จะไม่มีข้อมูลให้เลือกในขั้นตอนตั้งค่าใช้งาน

#### ตัวอย่างการอ่านค่าจาก NodeMCU มาแสดงผลบน NETPIE freeboard ด้วย Gauge

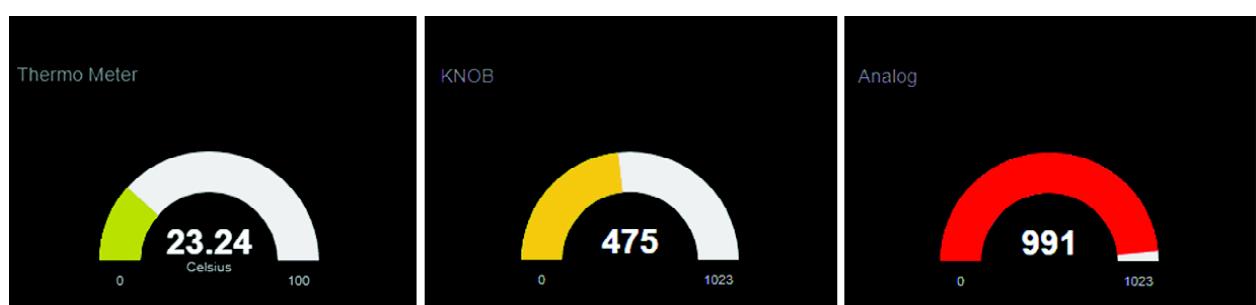
##### ขั้นตอนการตั้งค่า NodeMCU-12E

(6.2.1.1) เปิดโปรแกรม Arduino IDE 1.6.9 เลือกบอร์ดเป็น **NodeMCU1.0 (ESP-12E module)** และเลือกพอร์ตเชื่อมต่อให้ถูกต้อง

(6.2.1.2) พิมพ์โปรแกรมที่ 6-1 บันทึกไฟล์ในชื่อ **ReadA0\_to\_Netpie.ino** โดยโปรแกรมนี้จะ กำหนดให้ NodeMCU-12E อ่านค่าอะนาล็อกที่ขา A0 จากนั้นจะส่งค่าไปยัง NETPIE โดยใช้คำสั่ง microgear.chat("ESPID\_02", valA0) จะถูกส่งทุกๆ 1 วินาที หัวข้อหรือ Topic จริงๆ ของการ ส่งข้อมูลในครั้งนี้คือ **/GroupIndex/valAnalog** ดังนั้นจะนำ Topic หรือหัวข้อนี้ไปอ่านข้อมูลโดยใช้ REST API และทดสอบการอ่านข้อมูลด้วยเว็บบราวเซอร์ต่อไป

(6.2.1.3) ใช้งานในรูปที่ 6-14 ในการทดลอง ซึ่งก็คือ การใช้ตัวด้านหน้าปรับค่าได้บนบอร์ด AX-NodeMCU ในการทดลองนั้นเอง เพียงเลือกตัวอัลเมอร์มายังตำแหน่ง VR

(6.2.1.4) อัปโหลดโปรแกรมลงบน NodeMCU-12E แล้วให้เปิดหน้าต่าง **Serial Monitor** เพื่อ ดูผลการทำงาน ตั้งค่าอัตราบอเดี้ยมเป็น 115200 บิตต่อวินาที จากนั้นทดลองปรับค่าของตัวด้านหน้าปรับค่าได้บนบอร์ด AX-NodeMCU จะเห็นการเปลี่ยนแปลงค่าที่หน้าต่าง **Serial Monitor** ตามการปรับค่าของตัวด้านหน้าต่าง ดังรูปที่ 6-15



รูปที่ 6-13 ตัวอย่างของวิดเก็ตแบบ Gauge ที่มีใน NETPIE Freeboard

```
/*
 * NETPIE ESP8266 basic sample
 */

#include <ESP8266WiFi.h>
#include <MicroGear.h>

const char* ssid      = "SSID";
const char* password = "Password";

#define APPID    "GroupInex"
#define KEY      "YnlyT7QqyKyYFk6"
#define SECRET   "Ksn1hYJ2ELSNjsKb2iiJa6YIg"
#define ALIAS    "ESPID_01"

WiFiClient client;

int timer = 0;
MicroGear microgear(client);

/* If a new message arrives, do this */
void onMsghandler(char *topic, uint8_t* msg, unsigned int msglen)
{
    Serial.print("Incoming message --> ");
    msg[msglen] = '\0';
    Serial.println((char *)msg);
}

/* When a microgear is connected, do this */
void onConnected(char *attribute, uint8_t* msg, unsigned int msglen)
{
    Serial.println("Connected to NETPIE... ");
    /* Set the alias of this microgear ALIAS */
    microgear.setAlias(ALIAS);
}

void setup()
{
    /* Add Event listeners */
    /* Call onMsghandler() when new message arraives */
    microgear.on(MESSAGE,onMsghandler);
    /* Call onConnected() when NETPIE connection is established */
    microgear.on(CONNECTED,onConnected);
    Serial.begin(115200);
    Serial.println("Starting... ");

    /* Initial WIFI, this is just a basic method to configure WIFI on ESP8266.*/
    if (WiFi.begin(ssid, password))
    {
        while (WiFi.status() != WL_CONNECTED)
        {
            delay(500);
            Serial.print(".");
        }
    }
}
```

โปรแกรมที่ 6-1 ไฟล์ ReadAO\_to\_Netpie.ino สำหรับ NodeMCU-12E เพื่อทดสอบการทำงานของวิดเก็ต Gauge ของ NETPIE Freeboard (มีต่อ)

```

Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());

/* Initial with KEY, SECRET and also set the ALIAS here */
microgear.init(KEY, SECRET, ALIAS);

/* connect to NETPIE to a specific APPID */
microgear.connect(APPID);
}

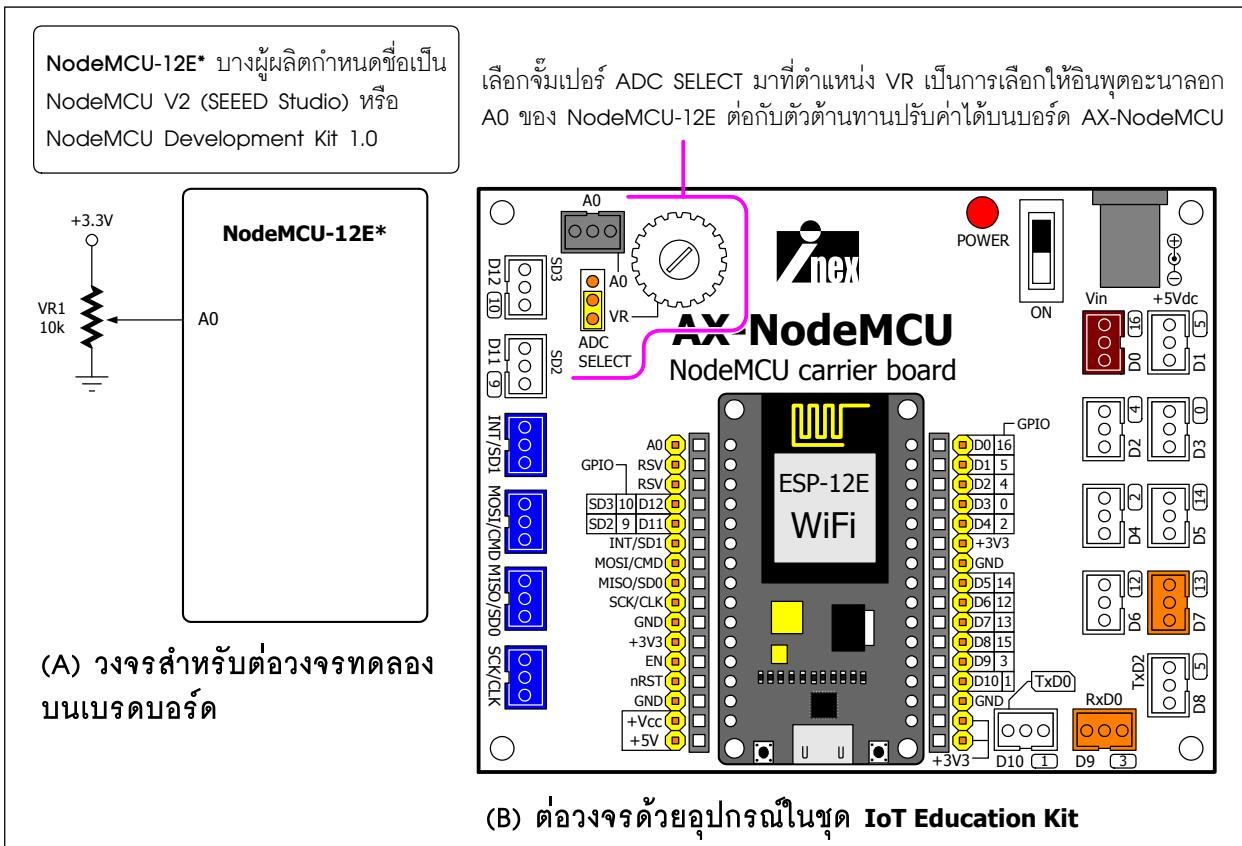
void loop()
{
    int valA0 = analogRead(A0);
    /* To check if the microgear is still connected */
    if (microgear.connected())
    {
        Serial.println("connected");

        /* Call this method regularly otherwise the connection may be lost */
        microgear.loop();
        if (timer >= 1000)
        {
            Serial.println("Publish...");
            Serial.println("Analog pin A0 = " + String(valA0));
            /* Chat with the microgear named ALIAS which is myself */
            microgear.chat("ESPID_02", valA0);
            timer = 0;
        }
        else timer += 100;
    }
    else
    {
        Serial.println("connection lost, reconnect...");
        if (timer >= 5000)
        {
            microgear.connect(APPID);
            timer = 0;
        }
        else timer += 100;
    }
    delay(100);
}

```

---

**โปรแกรมที่ 6-1 ไฟล์ ReadAO\_to\_Netpie.ino สำหรับ NodeMCU-12E เพื่อทดสอบการทำงานของวิดเก็ต Gauge ของ NETPIE Freeboard (จบ)**



รูปที่ 6-4 วงจรและการใช้บอร์ด AX-NodeMCU ในการทดสอบ

```
connected
connected
connected
connected
connected
connected
connected
connected
connected
Publish...
Analog pin A0 = 269
connected
connected
connected
connected
connected
connected
connected
connected
```

Autoscroll      No line ending      115200 baud      Clear output

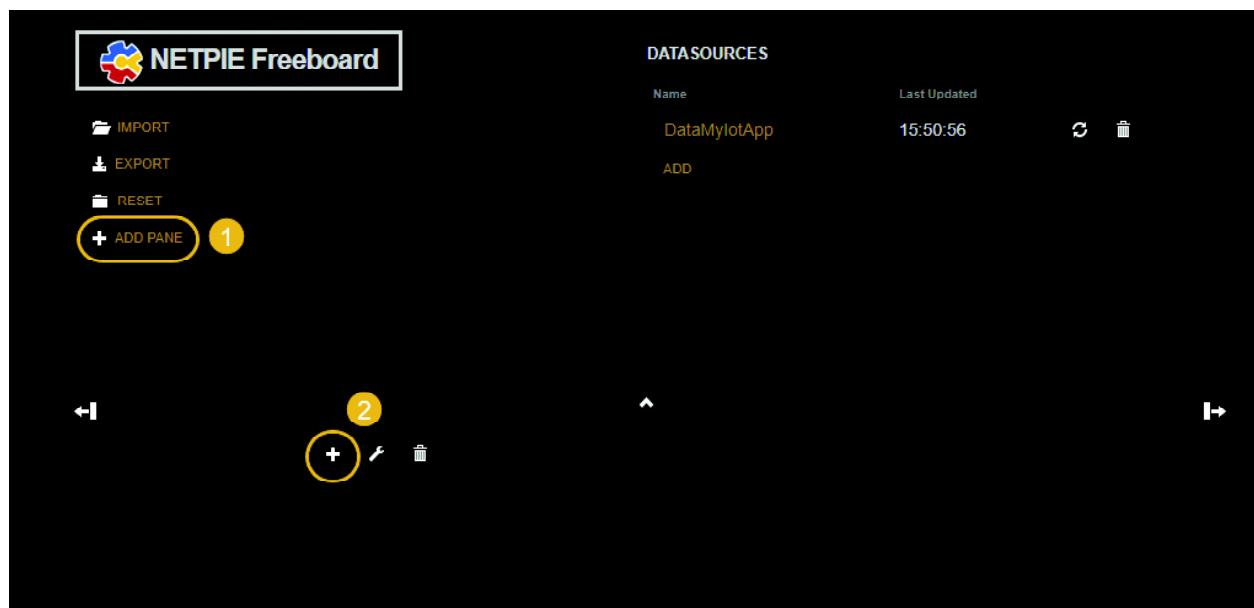
รูปที่ 6-15 หน้าต่าง Serial Monitor แสดงการทำงานของ NodeMCU-12E ในการอ่านอะนาล็อกจากตัวต้านทานปรับค่าได้และส่งค่าไปยัง NETPIE

### ขั้นตอนเพิ่ม Gauge บน NETPIE Freeboard

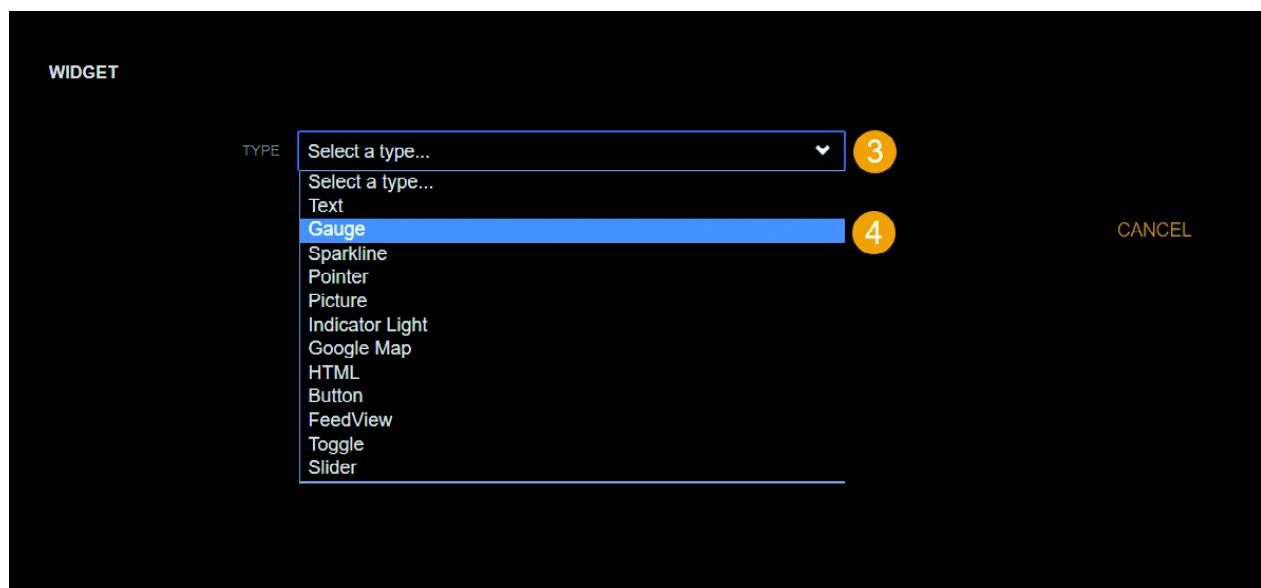
(6.2.1.5) เข้าสู่ระบบ NETPIE ทำการ Log in แล้วเลือกเมนู RESOURCES > FREEBOARDS เพื่อเตรียมสร้างหน้าต่างแสดงผล ตามขั้นตอนในหัวข้อ 6.1

(6.2.1.6) เพิ่ม Datasources โดยเลือก Datasources ที่มีชื่อว่า **DataMyIotApp** เริ่มจากคลิกที่ **ADD PANE** จากนั้นคลิกเครื่องหมาย + เพื่อเลือกเพิ่มวิดเก็ตหรืออุปกรณ์

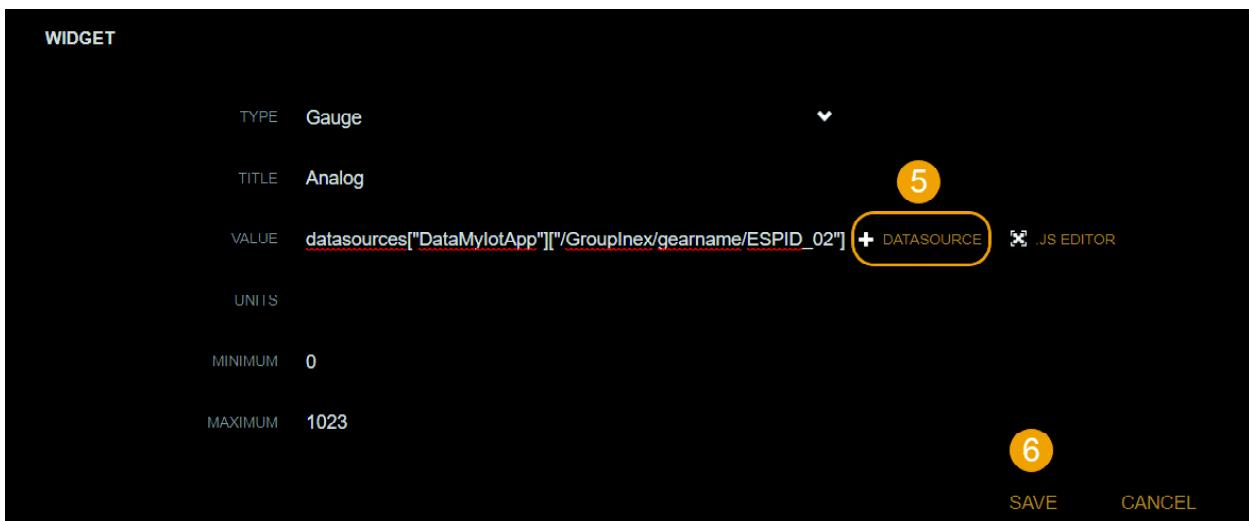
(6.2.1.7) คลิกปุ่มลูกศรลงด้านล่าง (วงกลมหมายเลข 3 ในรูปที่ 6-17) จากนั้นเลือกรายการ Gauge (วงกลมหมายเลข 4 ในรูปที่ 6-17)



รูปที่ 6-16 เริ่มต้นสร้าง Datasources สำหรับวิดเก็ต Gauge



รูปที่ 6-17 เริ่มต้นสร้าง Datasources สำหรับวิดเก็ต Gauge



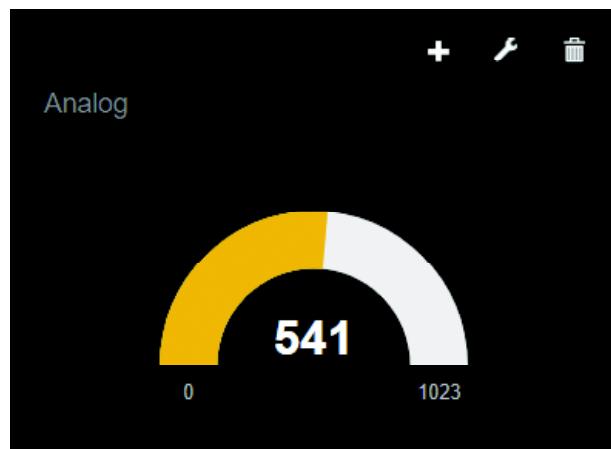
รูปที่ 6-18 ตั้งค่าพารามิเตอร์ให้กับวิดเก็ต Gauge สำหรับใช้แสดงผลบน NETPIE Freeboard

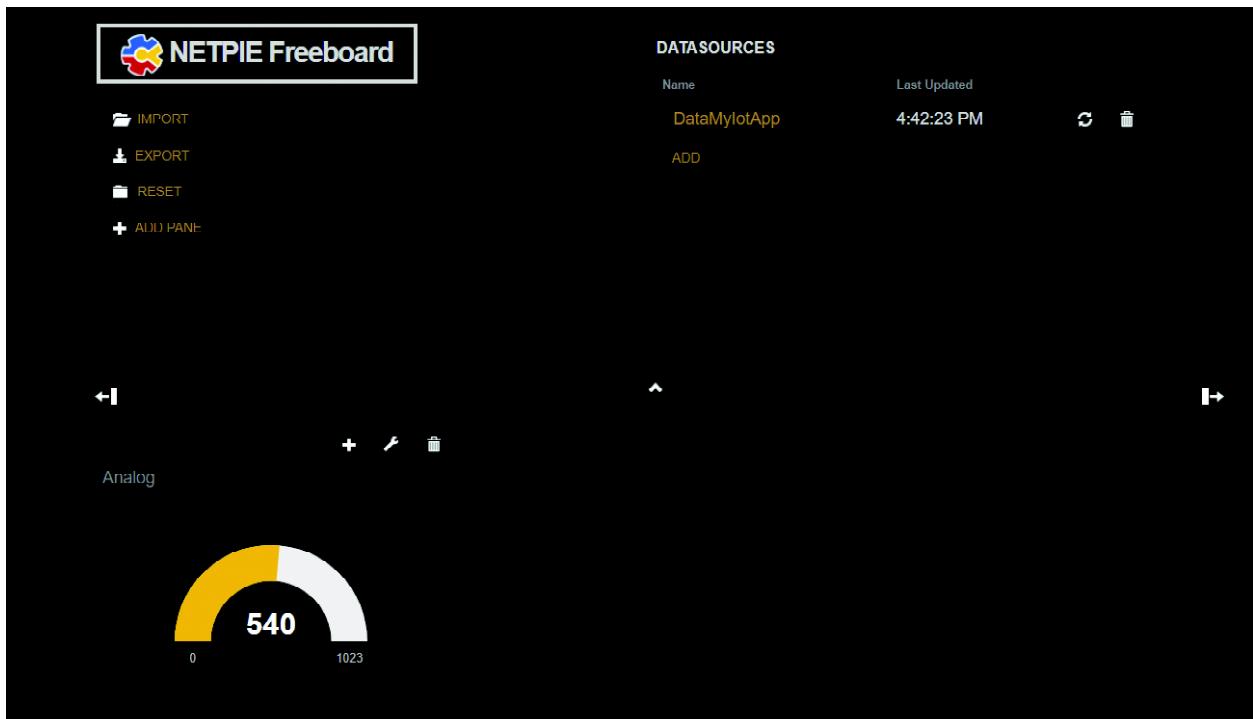
(6.2.1.8) หน้าต่างตั้งค่าพารามิเตอร์ของวิดเก็ต Gauge ปรากฏขึ้นมาดังรูปที่ 6-18 ตั้งค่าดังนี้

```
TYPE : Gauge
TITLE : Analog
VALUE : datasources ["DataMyIotApp"] ["/GroupInex/gearname/ESPID_02"]
MINIMUM : 0
MAXIMUM : 1023
```

(6.2.1.9) คลิกที่ปุ่ม **DATASOURCE** (วงกลม 5 ในรูปที่ 6-18) เพื่อเลือกข้อมูลที่ต้องการแสดงให้สังเกตข้อความ **ESPID\_02** เป็นชื่อที่ใช้ส่งข้อมูลมายัง NETPIE จากคำสั่ง `microgear.chat-("ESPID_02", valA0)` ของไฟล์ **ReadA0\_to\_Netpie.ino** (โปรแกรมที่ 6-1) ถ้าไม่มีให้เลือก หมายความว่า ยังไม่มีข้อมูลส่งมา�ัง NETPIE ต้องตรวจสอบ Key ต่างๆ ที่เกี่ยวข้องและตั้งค่าฐานอินเทอร์เน็ตที่ทำการเชื่อมต่อ จากนั้น คลิกปุ่ม **SAVE** (วงกลม 6 ในรูปที่ 6-18) เพื่อบันทึกการตั้งค่า

จากนั้นที่หน้าต่างของ Freeboard จะแสดงผลมาตรวัดในแบบ Gauge ขึ้นมา





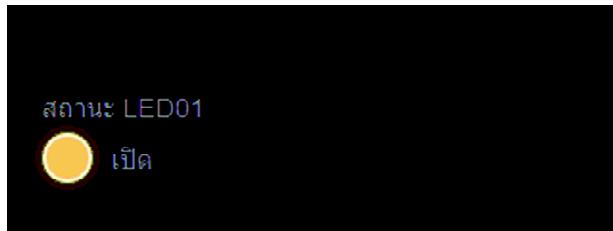
รูปที่ 6-19 การแสดงผลการทำงานของวิดเก็ต Gauge บน NETPIE Freeboard เพื่อแสดงค่าของสัญญาณอะนาลอกที่อ่านได้แล้วส่งมาแสดงผลที่ NETPIE Freeboard ผ่านเครือข่ายอินเทอร์เน็ต

#### ทดสอบการทำงาน

(6.2.1.10) ทดสอบการทำงานด้วยการปรับหมุนตัวด้านท่านปรับค่าได้บนบอร์ด AX-NodeMCU จะเห็นการเปลี่ยนแปลงของมาตรวัด Gauge ดังแสดงในรูปที่ 6-19

## 6.2.2 ใช้งาน Indicator light

Indicator light เป็นตัวแสดงผลที่เลียนแบบหลอดไฟ มี 2 สถานะของการทำงานคือ ติดและดับ



### ตัวอย่างการใช้งานวิธีเก็ต Indicator Light แสดงผลการทำงานของ NodeMCU-12E บน NETPIE freeboard

มีขั้นตอนดังนี้

#### ขั้นตอนทางฝั่ง NodeMCU-12E

(6.2.2.1) เปิดโปรแกรม Arduino IDE เลือกบอร์ดเป็น **NodeMCU1.0 (ESP-12E module)** และเลือกพอร์ตเชื่อมต่อให้ถูกต้อง

(6.2.2.2) พิมพ์โปรแกรมที่ 6-2 บันทึกไฟล์ในชื่อ **switchToggle\_to\_Netpie.ino** โดยโปรแกรมนี้จะกำหนดให้ NodeMCU-12E อ่านค่าสถานะล็อกอินที่อินพุต D3 เพื่อควบคุมการทำงานของ LED ที่ต่ออยู่กับขา D5 ของ NodeMCU-12E จากนั้นจึงส่งค่าไปยัง NETPIE ทุกครั้งที่กดสวิตช์ LED จะทำการกลับสถานะการทำงานในปัจจุบัน เช่น หากเดิม LED ดับจะเปลี่ยนเป็นติดเมื่อกดสวิตช์ และหากเดิม LED ติดอยู่ จะดับลงเมื่อกดสวิตช์ แล้วส่งค่าสถานะปัจจุบันไปยัง NETPIE

(6.2.2.3) ใช้งานในรูปที่ 6-20 ในการทดลอง

(6.2.2.4) อัปโหลดโปรแกรมลงบน NodeMCU-12E แล้วให้เปิดหน้าต่าง Serial Monitor เพื่อคุ้มการทำงาน ตั้งค่าอัตราบอเดิน 115200 บิตต่อวินาที จากนั้นคุณการแสดงผลของ LED ที่ต่อ กับขา D5 ของ NodeMCU-12E ทำการกดสวิตช์ที่ต่อ กับขา D3 แล้วสังเกตการทำงานของ LED และการส่งค่าไปยัง NETPIE จากหน้าต่าง Serial Monitor

```
#include <ESP8266WiFi.h>
#include <MicroGear.h>

const char* ssid      = "Your SSID";           // Your SSID
const char* password = "Your password";        // Your password

#define APPID      "GroupInex"
#define KEY        "Yn1yT7QqyKyYFk6"
#define SECRET    "Ksn1hYJ2ELSNjsKb2iiJa6YIg"
#define ALIAS      "ESPID_01"

WiFiClient client;

int timer = 0;
int swPin = D3;
int ledPin = D5;
bool statusLed, lastStatusLed;

MicroGear microgear(client);
/* If a new message arrives, do this */
void onMsghandler(char *topic, uint8_t* msg, unsigned int msglen)
{
    Serial.print("Incoming message --> ");
    msg[msglen] = '\0';
    Serial.println((char *)msg);
}

/* When a microgear is connected, do this */
void onConnected(char *attribute, uint8_t* msg, unsigned int msglen)
{
    Serial.println("Connected to NETPIE...");
    /* Set the alias of this microgear ALIAS */
    microgear.setAlias(ALIAS);
}

void setup()
{
    pinMode(swPin, INPUT);
    pinMode(ledPin, OUTPUT);
    attachInterrupt(swPin, toggle, FALLING);

    /* Add Event listeners */
    /* Call onMsghandler() when new message arrives */
    microgear.on(MESSAGE, onMsghandler);

    /* Call onConnected() when NETPIE connection is established */
    microgear.on(CONNECTED, onConnected);
    Serial.begin(115200);
    Serial.println("Starting...");
```

**โปรแกรมที่ 6-2 ไฟล์ switchToggle\_to\_Netpie.ino สำหรับ NodeMCU-12E เพื่อทดสอบการทำงานของวิดเก็ต Indicator Light ของ NETPIE Freeboard (มีต่อ)**

```

/* Initial WIFI, this is just a basic method to configure WIFI on ESP8266. */
unsigned long timeoutconnect = millis();
if (WiFi.begin(ssid, password))
{
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
        if (millis() - timeoutconnect > 8000)
        {
            break;
        }
    }
}

Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
/* Initial with KEY, SECRET and also set the ALIAS here */
microgear.init(KEY, SECRET, ALIAS);

/* connect to NETPIE to a specific APPID */
microgear.connect(APPID);
}

void loop()
{ /* To check if the microgear is still connected */
if (microgear.connected())
{
    Serial.println("connected");
    /* Call this method regularly otherwise the connection may be lost */
    microgear.loop();
    if (lastStatusLed != statusLed)
    {
        microgear.chat("ESPID_02/LED", statusLed); // เสียบ LED
        lastStatusLed = statusLed;
    }
}
else
{
    Serial.println("connection lost, reconnect...");
    if (timer >= 5000)
    {
        microgear.connect(APPID);
        timer = 0;
    }
    else timer += 100;
}
delay(100);
}

```

โปรแกรมที่ 6-2 ไฟล์ `switchToggle_to_Netpie.ino` สำหรับ NodeMCU-12E เพื่อทดสอบการทำงานของวิดเก็ต **Indicator Light** ของ NETPIE Freeboard (มีต่อ)

```

void toggle()
{
    statusLed = !statusLed;
    digitalWrite(ledPin, statusLed);
}

```

### คำอธิบายโปรแกรมเพิ่มเติม

เมื่อรันโปรแกรม LED จะดับ หากสวิตช์ที่ต่ออยู่กับขาพอร์ต D3 ถูกกด LED จะเปลี่ยนเป็นติดสว่าง และส่ง สถานะของ LED ไปยัง NETPIE ด้วย หากกดสวิตช์อีกครั้ง LED จากเดิมที่ติดสว่าง จะกลับมาดับอีกครั้ง พร้อมส่ง ค่าไปยัง NETPIE ด้วย การทำงานของโปรแกรมนี้จะเป็นอย่างปกติ ไม่ว่า NodeMCU-12E จะเชื่อมต่อ กับ Wi-Fi ไม่ได้ หรือการเชื่อมต่อ กับ NETPIE หลุดกระหันหัน การกดปุ่มเพื่อเปิด-ปิด LED ยังคงใช้งานได้อย่างปกติ ด้วยคุณสมบัติ การตอบสนองต่อการอินเตอร์รัปต์ หรือการขัดจังหวะภายนอกที่ขา D3

จากโปรแกรมกำหนดให้ขาพอร์ต D3 ทำงานในลักษณะตอบสนองสัญญาณอินเตอร์รัปต์ จากการภายนอก โดย ทำงานเมื่อตรวจสอบสัญญาณขอบขาลง (falling edge) จากนั้นเรียกฟังก์ชัน toggle()

รูปแบบคำสั่งการตอบสนองสัญญาณอินเตอร์รัปต์ จากการภายนอก

attachInterrupt (pin, ISR, mode)

Pin : ขาพอร์ตของ NodeMCU-12E ใช้ได้ทั้งหมด ยกเว้นขา GPIO16 หรือ D0

ISP: ฟังก์ชันตอบสนองอินเตอร์รัปต์ที่ต้องไปทำงาน

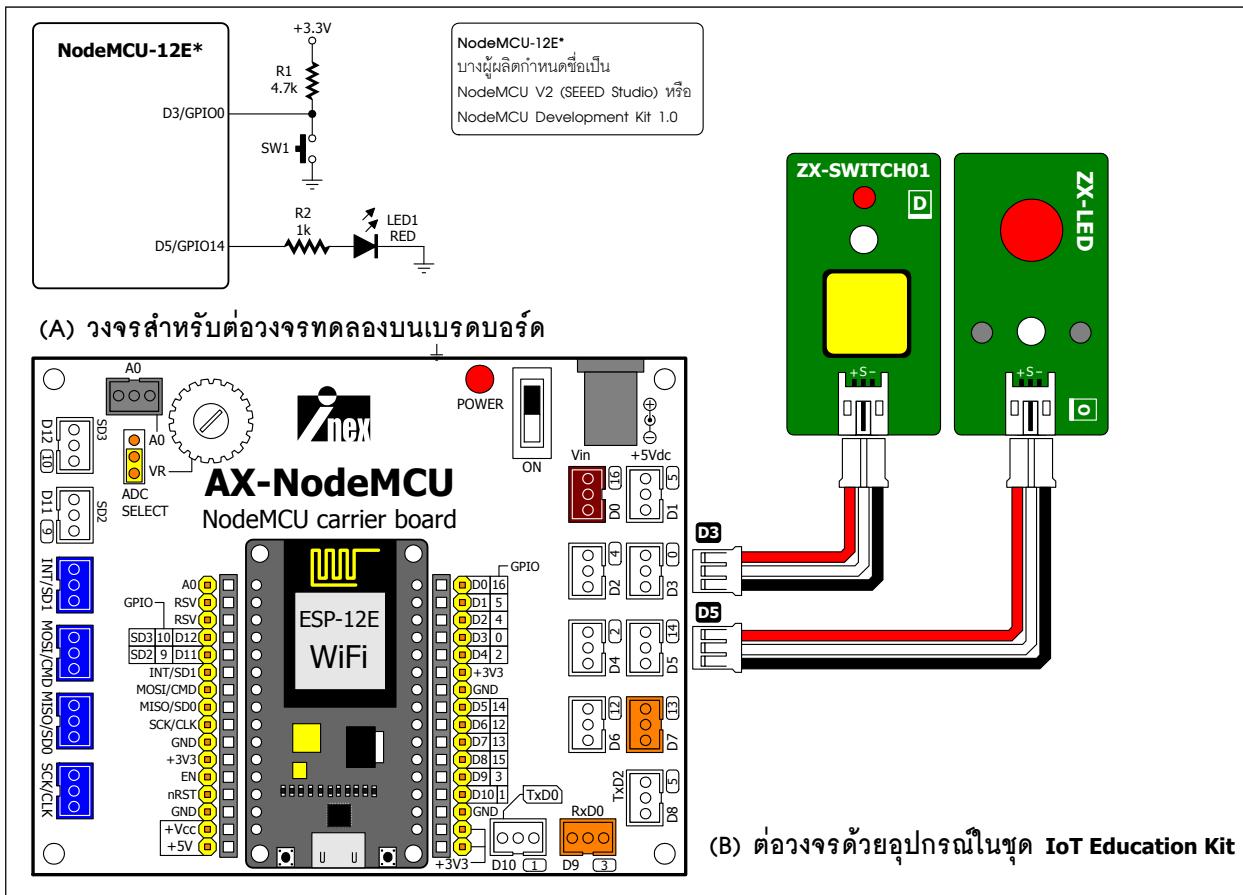
Mode : ลักษณะสัญญาณที่ต้องการตรวจสอบ

CHANGE : ตรวจสอบทั้งขอบขาขึ้นและลงของสัญญาณ

RISING : ตรวจสอบขอบขาขึ้นของสัญญาณ

FALLING: ตรวจสอบขอบขาลงของสัญญาณ

**โปรแกรมที่ 6-2** ไฟล์ switchToggle\_to\_Netpie.ino สำหรับ NodeMCU-12E เพื่อทดสอบการทำงานของวิดเก็ต Indicator Light ของ NETPIE Freeboard (จบ)



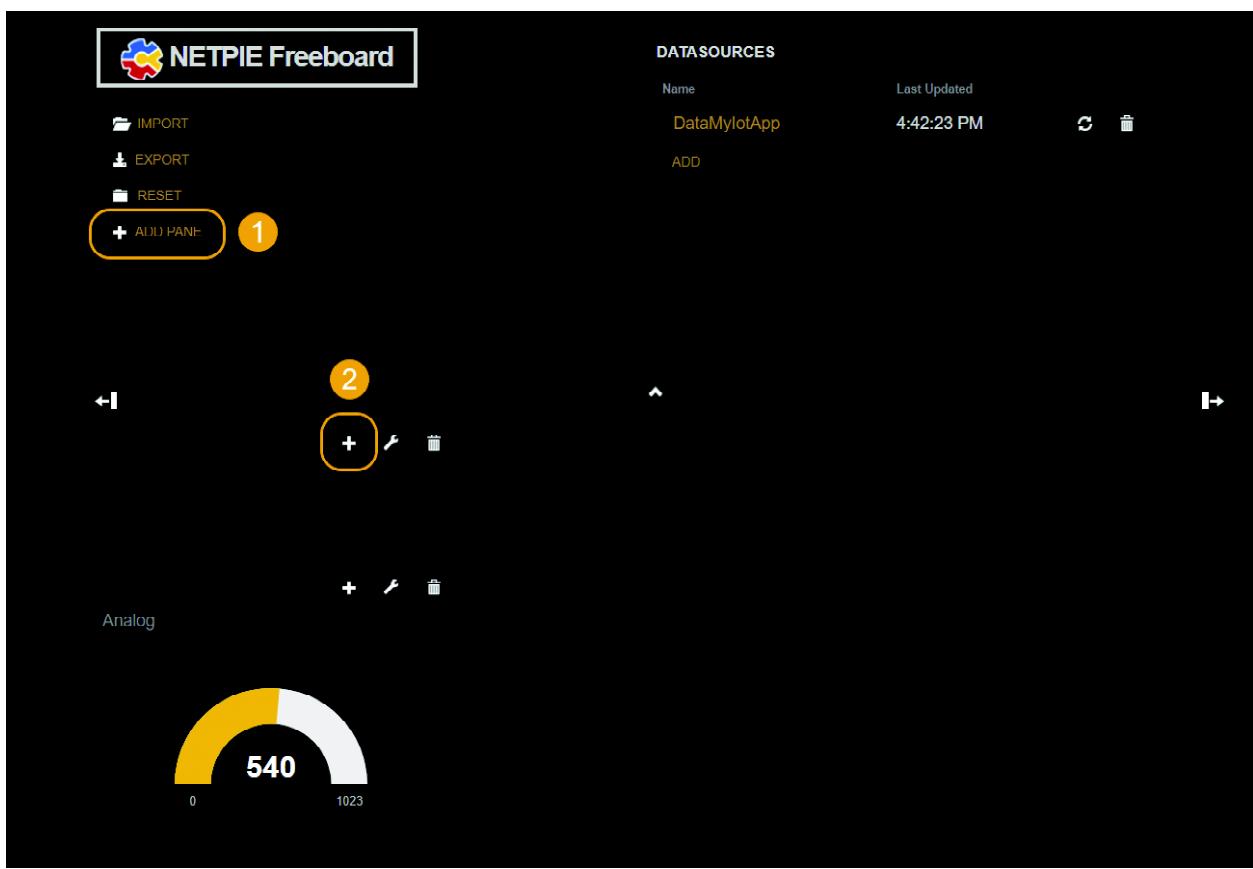
รูปที่ 6-20 วงจรและการใช้บอร์ด AX-NodeMCU ในการทดสอบการใช้งานวิดเก็ต Indicator Light บน NETPIE Freeboard

### ขั้นตอนการเพิ่ม Indicator Light บน NETPIE Freeboard

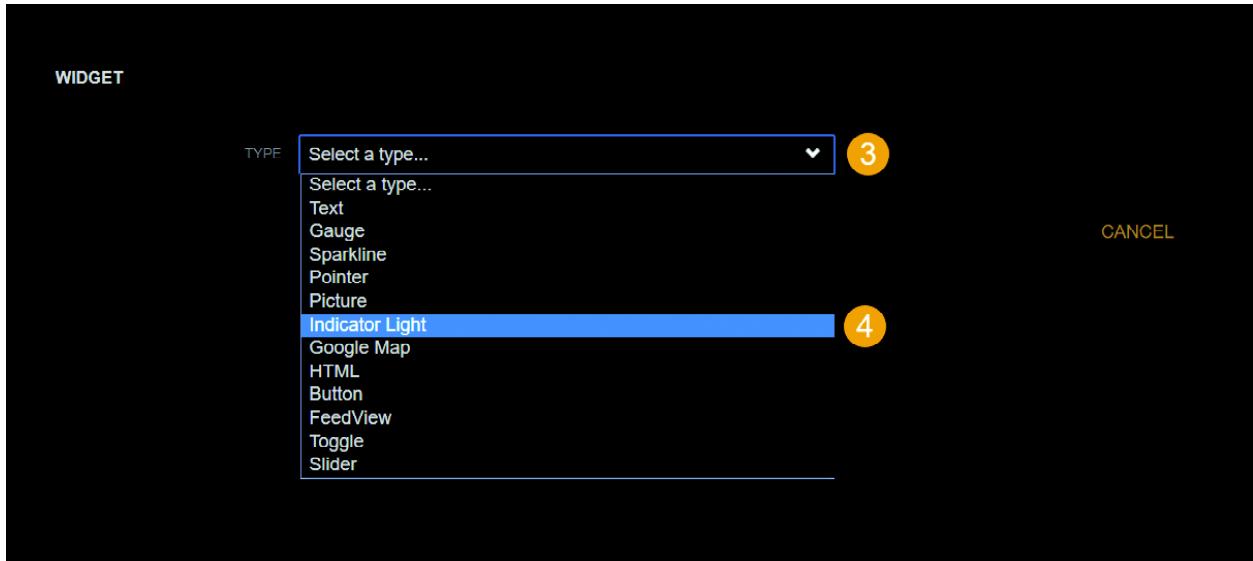
(6.2.2.5) เข้าสู่ระบบ NETPIE ทำการ Log in แล้วเลือกเมนู **RESOURCES > FREEBOARDS** เพื่อเตรียมสร้างหน้าต่างแสดงผล ตามขั้นตอนในหัวข้อ 6.1

(6.2.2.6) ยังคงเลือก Datasources ที่มีชื่อว่า **DataMyIotApp** จากนั้นคลิกที่ **ADD PANE** (วงกลมหมายเลข 1 ในรูปที่ 6-21) ตามด้วยคลิก + (วงกลมหมายเลข 2 ในรูปที่ 6-21) เพื่อสร้างพื้นที่ว่างวิดเก็ต หรืออุปกรณ์ ตามรูปที่ 6-21

(6.2.2.7) หน้าต่างสำหรับเลือก Datasources ปรากฏขึ้นมา ดังรูปที่ 6-22 คลิกเลือก **TYPE** หรือ **ชนิดของวิดเก็ต** (วงกลมหมายเลข 3 ในรูปที่ 6-22) จากนั้นเลือก **Indicator Light** (วงกลมหมายเลข 4 ในรูปที่ 6-22)



รูปที่ 6-21 เริ่มต้นสร้าง Datasources (ทำต่อจากการสร้างวิดเก็ต Gauge ในหัวข้อ 6.2.1)



รูปที่ 6-22 เลือกชนิดของอุปกรณ์แสดงผลหรือวิดเก็ต

(6.2.2.8) กำหนดค่าพารามิเตอร์ของวิดเก็ต Indicator Light ดังรูปที่ 6-23

TYPE : Indicator Light

TITLE : LED

VALUE : `return(datasources["DataMyIotApp"]["/GroupInex/gearname/ESPID_02/LED"]=="0"?0:1)`

ON TEXT : ON

OFF TEXT : OFF



รูปที่ 6-23 หน้าต่างแสดงการกำหนดค่าพารามิเตอร์ของวิดเก็ต Indicator Light รวมถึงการแก้ไขโค้ด Java Script ของวิดเก็ตด้วย

(6.2.2.9) คลิกหัวข้อ **DATASOURCES** (วงกลมหมายเลข 5 ในรูปที่ 6-23) เพื่อเลือกข้อมูลมาแสดง สังเกตว่า ข้อมูลที่ต้องการแสดงมีข้อความลงท้ายว่า **ESPID\_02/LED** ข้อความนี้มาจากคำสั่ง **microgear.chat("ESPID\_02/LED", statusLed)** ที่ใช้เป็นหัวข้อหรือ Topic ของการส่งสถานะของ LED ที่เกิดจากการกดปุ่ม

(6.2.2.10) จากนั้นคลิกหัวข้อ **.JS EDITOR** เพื่อเข้าไปแก้ไข โค้ด Java Script ให้วิดเก็ต Indicator Light เกิดการแสดงผลดังในกรอบย่อของรูปที่ 6-24 โดยเพิ่มเติมดังนี้

```
return (datasources ["DataMyIotApp"] ["/GroupIndex/gearname/
ESPID_02/LED"] == "0" ? 0 : 1)
```

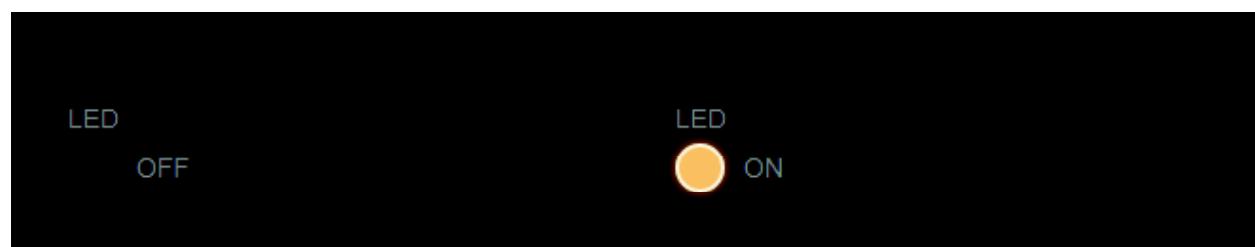
จากชุดคำสั่งนี้ เมื่ออ่านค่ามาเป็น “0” แต่อยู่ในรูปแบบข้อความ ซึ่งนำไปใช้กับวิดเก็ตนี้ไม่ได้ จะต้องเปลี่ยนให้อยู่ในรูปแบบตัวเลข 1 : 0 หรือ true : false เท่านั้น ด้วยชุดคำสั่งนี้จะส่งค่าเป็นตัวเลข 0 ออกไป ถ้าหากไม่ใช่ตัวเลข “0” จะส่งค่ากลับเป็น 1 ทันที ซึ่งใช้ได้ในกรณีที่ต้องการเพียงเลข 1 กับ 0 เท่านั้น

(6.2.2.11) คลิกปุ่ม **CLOSE** เพื่อจบการแก้ไขและกลับมาข้างหน้าต่าง WIDGET อีกครั้ง คลิกปุ่ม **SAVE** (วงกลมหมายเลข 7 ในรูปที่ 6-23)

### ทดสอบการทำงาน

(6.2.2.12) กดสวิตช์ที่ต่อเข้ากับขาพอร์ต D3 ของ NodeMCU และวัดการทำงานของ LED ที่ต่อ กับขาพอร์ต D5 และที่หน้าต่างแสดงผลของ NETPIE Freeboard ที่มีการเพิ่มวิดเก็ต Indicator Light ดังรูปที่ 6-24

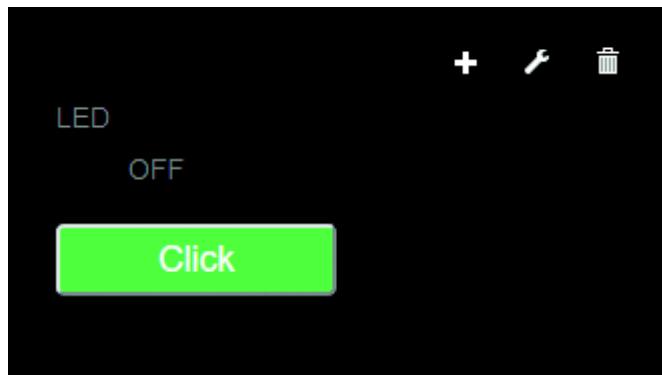
(6.2.2.12) หากทดสอบกดปุ่ม **Refresh** หรือกดคีย์ F5 บนคีย์บอร์ดของคอมพิวเตอร์ จะพบว่า การแสดงผลวิดเก็ต Indicator Light ของ NETPIE Freeboard อาจจะไม่ตรงตามสถานะจริงของ LED แก้ปัญหานี้ได้ด้วยการเพิ่มส่วนที่ส่งคำสั่งไปตาม NodeMCU-12E เพื่อขอทราบสถานะที่แท้จริงในทุกครั้งที่มีการรีเฟรชหรือเรียกหน้าเว็บให้แสดงผลใหม่ ซึ่งจะได้อธิบายในหัวข้อถัดไปในเรื่องการใช้งานวิดเก็ต Button



รูปที่ 6-24 แสดงวิดเก็ต Indicator Light ที่ใช้แสดงการทำงานของ LED ที่ควบคุมด้วยสวิตช์

### 6.2.3 ใช้งาน Button

**Button** เป็นตัวช่วยให้ผู้ใช้งานสามารถสั่งงานผ่านหน้า NETPIE Freeboard ไปยังอุปกรณ์ที่ต้องการ โดยกำหนดได้ว่า จะใช้ปุ่มนี้สั่งงาน NodeMCU-12E หรืออุปกรณ์ใดๆ ที่อยู่ภายใต้การเชื่อมต่อของ NETPIE



รูปที่ 6-25 หน้าตาของวิดเก็ต Button ที่มีชื่อปุ่มว่า Click ทำงานบนหน้าเว็บ NETPIE Freeboard

ในตัวอย่างการใช้งานวิดเก็ต Button ที่จะอธิบายต่อไปนี้ใช้แก่ปัญหาการแสดงผลของวิดเก็ต Indicator Light ที่ไม่ตรงกับสถานะจริงของ LED จากตัวอย่างในหัวข้อ 6.2.2 โดยควบคุม LED ทั้ง 2 ช่องทางคือ สั่งโดยตรงจากสวิตซ์ที่ต่อ กับ NodeMCU-12E และสั่งงานผ่านหน้าเว็บ NETPIE Freeboard

**ตัวอย่างการใช้งานวิดเก็ต Button เพื่อสั่งการให้ NodeMCU-12E ขับ LED ผ่าน NETPIE freeboard**

มีขั้นตอนดังนี้

ขั้นตอนทั้งหมด NodeMCU-12E

(6.2.3.1) เปิดโปรแกรม Arduino IDE เลือกบอร์ดเป็น NodeMCU1.0 (ESP-12E module) และเลือกพอร์ตเชื่อมต่อให้ถูกต้อง

(6.2.3.2) พิมพ์โปรแกรมที่ 6-3 บันทึกไฟล์ในชื่อ NodeMCU\_send\_receive\_Netpie.ino

```

#include <ESP8266WiFi.h>
#include <MicroGear.h>

const char* ssid      = "SSID";
const char* password = "Password";

#define APPID    "GroupInex"
#define KEY      "Yn1yT7QqyKyYFk6"
#define SECRET   "Ksn1hYJ2ELSNjsKb2iiJa6YIg"
#define ALIAS    "ESPID_01"

WiFiClient client;

int timer = 0;
int swPin = D3;
int ledPin = D5;
bool statusLed, lastStatusLed;

MicroGear microgear(client);

/* If a new message arrives, do this */
void onMsgHandler(char *topic, uint8_t* msg, unsigned int msglen)
{
    Serial.print("Incoming message --> ");
    msg[msglen] = '\0';
    Serial.println((char *)msg);
    if (msg[0] == 'L')
    {
        statusLed = !statusLed;
        digitalWrite(ledPin, statusLed);
    }
    else if (msg[0] == '!')
    {
        microgear.chat("ESPID_02/LED", statusLed);
    }
}
/* When a microgear is connected, do this */
void onConnected(char *attribute, uint8_t* msg, unsigned int msglen)
{
    Serial.println("Connected to NETPIE...");
    /* Set the alias of this microgear ALIAS */
    microgear.setAlias(ALIAS);
}

void setup()
{
    pinMode(swPin, INPUT);
    pinMode(ledPin, OUTPUT);
    attachInterrupt(digitalPinToInterrupt(swPin), toggle, FALLING);
}

```

**โปรแกรมที่ 6-3 ไฟล์ NodeMCU\_send\_receive\_Netpie.ino สำหรับ NodeMCU-12E เพื่อทดสอบการทำงานของวิดเก็ตButton ของ NETPIE Freeboard (มีต่อ)**

```

/* Add Event listeners */
/* Call onMsgHandler() when new message arrives */
microgear.on(MESSAGE, onMsgHandler);
/* Call onConnected() when NETPIE connection is established */
microgear.on(CONNECTED, onConnected);

Serial.begin(115200);
Serial.println("Starting...");

/* Initial WIFI, this is just a basic method to configure WIFI on ESP8266.*/
unsigned long timeoutconnect = millis();
if (WiFi.begin(ssid, password))
{
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
        if (millis() - timeoutconnect > 8000)
        {
            break;
        }
    }
}
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());

/* Initial with KEY, SECRET and also set the ALIAS here */
microgear.init(KEY, SECRET, ALIAS);
/* connect to NETPIE to a specific APPID */
microgear.connect(APPID);
}

void loop()
{
    /* To check if the microgear is still connected */
    if (microgear.connected())
    {
        Serial.println("connected");
        /* Call this method regularly otherwise the connection may be lost */
        microgear.loop();
        if (lastStatusLed != statusLed)
        {
            microgear.chat("ESPID_02/LED", statusLed);
            lastStatusLed = statusLed;
        }
    }
}

```

โปรแกรมที่ 6-3 ไฟล์ NodeMCU\_send\_receive\_Netpie.ino สำหรับ NodeMCU-12E เพื่อทดสอบการทำงานของวิธีกดButton ของ NETPIE Freeboard (มีต่อ)

```

else
{
    Serial.println("connection lost, reconnect...") ;
    if (timer >= 5000)
    {
        microgear.connect(APPID) ;
        timer = 0;
    }
    else timer += 100;
}
delay(100);
}

void toggle()
{
    statusLed = !statusLed;
    digitalWrite(ledPin, statusLed);
}

```

### คำอธิบายการทำงานของโปรแกรมเพิ่มเติม

บุ๊ดคำสั่งที่เป็นส่วนของการรับค่าเมื่อมีการส่งข้อความถึงอุปกรณ์จะอยู่ในฟังก์ชัน `onMsgHandler` โดยใช้พารามิเตอร์ `msg` มาเป็นเงื่อนไขในการตรวจสอบว่า ข้อความที่เข้ามาต้องการให้ทำอะไร โดยมีเงื่อนดังนี้

```

if ( msg[0] == 'L')
// ถ้าข้อความเท่ากับ “L” หมายถึง ข้อความที่ใช้สั่งเปิด-ปิด LED
{
    statusLed = !statusLed; // กลับสถานะตัวแปร statusLed
    digitalWrite(ledPin, statusLed);
    // กำหนดให้การติด-ดับ LED ด้วยค่าตัวแปร statusLed
}
else if (msg[0] == '!')
// ถ้าข้อความเท่ากับ “!” หมายถึง ข้อความที่ใช้ถามสถานะล่าสุดของ LED
{
    microgear.chat("ESPID_02/LED", statusLed);
    // ส่งสถานะ LED กลับไปยัง NETPIE
}

```

**โปรแกรมที่ 6-3 ไฟล์ `NodeMCU_send_receive_Netpie.ino` สำหรับ NodeMCU-12E เพื่อทดสอบการทำงานของวิดเก็ต `Button` ของ NETPIE Freeboard (จบ)**

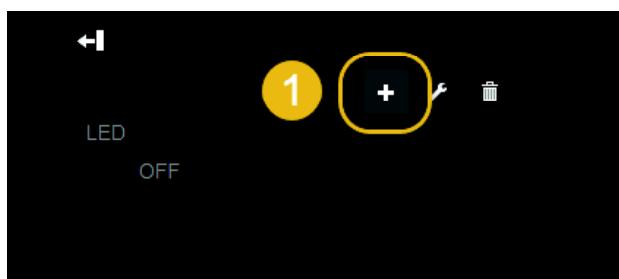
(6.2.3.3) ใช้งานในรูปที่ 6-20 ในการทดลอง

(6.2.3.4) อัปโหลดโปรแกรมลงบน NodeMCU-12E จากนั้นดูการแสดงผลของ LED ที่ต่อ กับขา D5 ของ NodeMCU-12E ทำการกดสวิตช์ที่ต่อ กับขา D3 แล้วสังเกตการทำงานของ LED

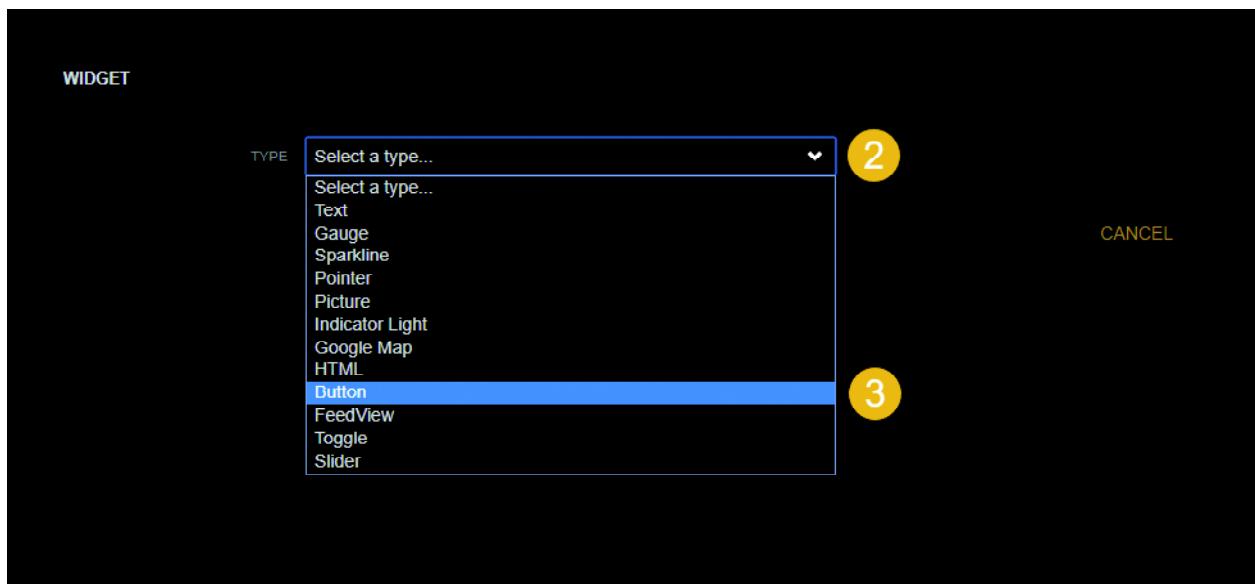
#### ขั้นตอนการเพิ่ม Button บน NETPIE Freeboard

(6.2.3.5) ในตัวอย่างนี้นำเสนองานสร้างวิดเก็ต Button ให้อยู่ร่วมกับ Indicator Light เริ่มต้นด้วยไปที่กรอบแสดงผลหรือพานะ LED คลิกที่เครื่องหมาย + (วงกลมหมายเลข 1 ในรูปที่ 6-26)

(6.2.3.6) หน้าต่างสำหรับเลือกชนิดของวิดเก็ตแสดงขึ้นมา คลิกที่ช่อง TYPE (วงกลมหมายเลข 2 ในรูปที่ 6-27) จากนั้นเลื่อนรายการลงมาเลือก Button (วงกลมหมายเลข 3 ในรูปที่ 6-27)



รูปที่ 6-26 คลิกเครื่องหมาย + ที่หน้าปัดของ LED ที่สร้างไว้ก่อนหน้านี้จากหัวข้อที่ 6.2.2 เพื่อเพิ่มวิดเก็ต



รูปที่ 6-27 เลือกเพิ่มวิดเก็ต Button เพื่อใช้ในการควบคุมการทำงานของ LED และวิดเก็ต Indicator Light บน NETPIE Freeboard

(6.2.3.7) หน้าต่างสำหรับตั้งค่าพารามิเตอร์ของวิดเก็ต Button ปรากฏขึ้นมา ตั้งค่าดังรูปที่ 6-28

TYPE : Button

BUTTON CAPTION : Click (กำหนดชื่อของปุ่ม - วงกลมหมายเลข 4)

BUTTON COLOR : Green (กำหนดสีของปุ่ม - วงกลมหมายเลข 5)

ONCLICK ACTION : microgear["DataMyIotApp"].chat("ESPID\_01","L")

ONCREATED ACTION : microgear["DataMyIotApp"].chat("ESPID\_01", "!" )

โดย

- **ONCLICK ACTION** (วงกลมหมายเลข 6) หมายถึง เลือกให้ทำงานเมื่อกดปุ่ม ในที่นี่คือ กำหนดให้ส่งข้อความต่อไปนี้ไปยังปลายทางเมื่อกดปุ่ม

- **microgear["DataMyIotApp"].chat("ESPID\_01","L")**

- **DataMyIoTApp** เป็นชื่อ DataSource ที่ได้สร้างไว้

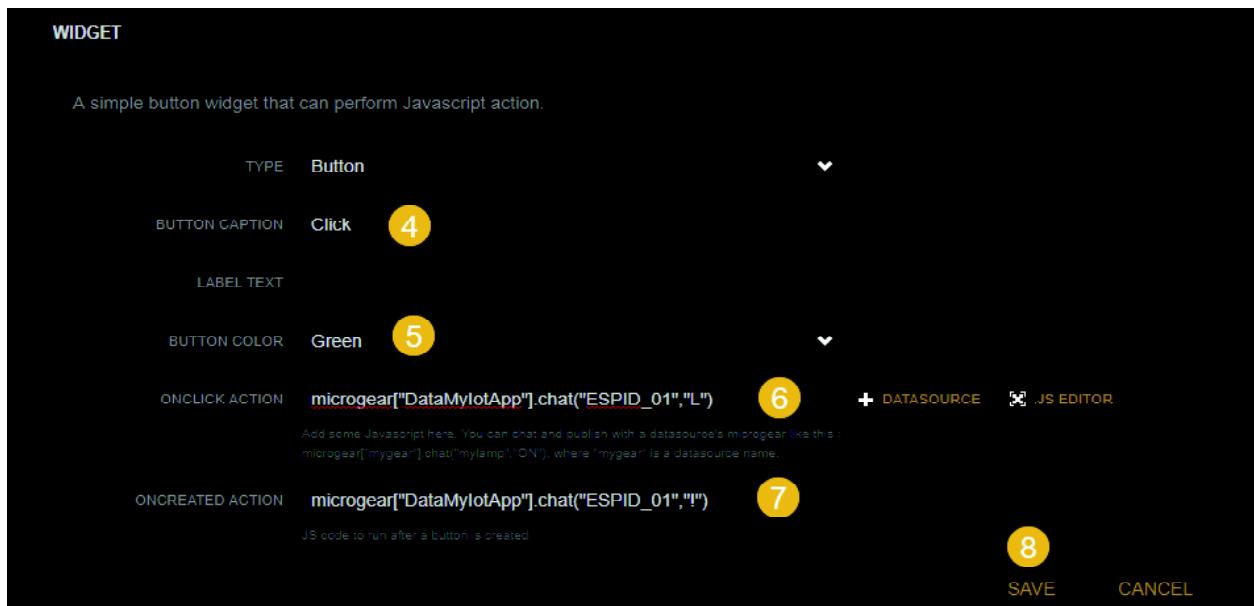
- **ESPID\_01** เป็นชื่อของอุปกรณ์ปลายทางที่ต้องการส่งข้อความไปให้

- **L** เป็นข้อความที่ต้องการส่งไปยังอุปกรณ์ที่ชื่อว่า **ESPID\_01**

- **ONCREATED ACTION** (วงกลมหมายเลข 7) หมายถึง เลือกให้ทำงานเกิดการสร้างปุ่มหรือเปิดหน้าเว็บขึ้นมาใหม่ โดยในที่นี่กำหนดให้ส่งข้อความต่อไปนี้

```
microgear["DataMyIotApp"].chat("ESPID_01", "!" )
```

นั่นคือ เมื่อมีการสร้างปุ่มขึ้นมาบนหน้าเว็บ เหตุการณ์ที่จะเกิดขึ้นประกอบด้วย ปุ่ม Save ถูกกด, เปิดหน้าเว็บ Freeboard ขึ้นมาใหม่, การกดรีเฟรชหน้าเว็บ เป็นต้น จึงใช้ความสามารถนี้ในการส่งข้อความ ! ตามอุปกรณ์ที่ชื่อ **ESPID\_01** ว่าสถานะล่าสุดของ LED เป็นอย่างไร



รูปที่ 6-28 หน้าต่างตั้งค่าพารามิเตอร์ของวิดเก็ต Button

(6.2.3.8) คลิกปุ่ม Save จะได้หน้าปัดของปุ่ม Click ที่สร้างขึ้นจากวิดเก็ต Button (เหมือนกับรูปที่ 6-26) สำหรับใช้งานต่อไป

#### ทดสอบการทำงาน

(6.2.3.9) คลิกปุ่ม Click จากนั้นสังเกต LED ที่ต่อ กับขาพอร์ต D5 NodeMCU-12E และวิดเก็ตรูป LED ที่อยู่บนหน้าเว็บ NETPIE Freeboard

(6.2.3.10) กดสวิตช์ที่ต่อ กับขาพอร์ต D3 ของ NodeMCU-12E สังเกตการทำงานของ LED ที่ส่องส่วน

(6.2.3.11) กดสวิตช์เพื่อทำให้ LED ดับ จากนั้นทำการเรียกหน้าเว็บให้แสดงผลใหม่หรือ Refresh โดยกดคีย์ F5 จากนั้นสังเกตว่า LED ที่อยู่บนหน้าเว็บ NETPIE Freeboard ดับเหมือนกับตัวอุปกรณ์จริงหรือไม่

### 6.2.4 ใช้งาน Toggle

Toggle เป็นวิดเก็ตที่ใช้งานเหมือนกับ Button ที่เพิ่มเติมคือ มีการระบุไว้ว่า กำลังกดปุ่ม ON หรือ OFF จึงคูเมื่อนเป็นการรวมวิดเก็ต Indicator light และ Button เข้าด้วยกัน ดังแสดงตัวอย่างในรูปที่ 6-29

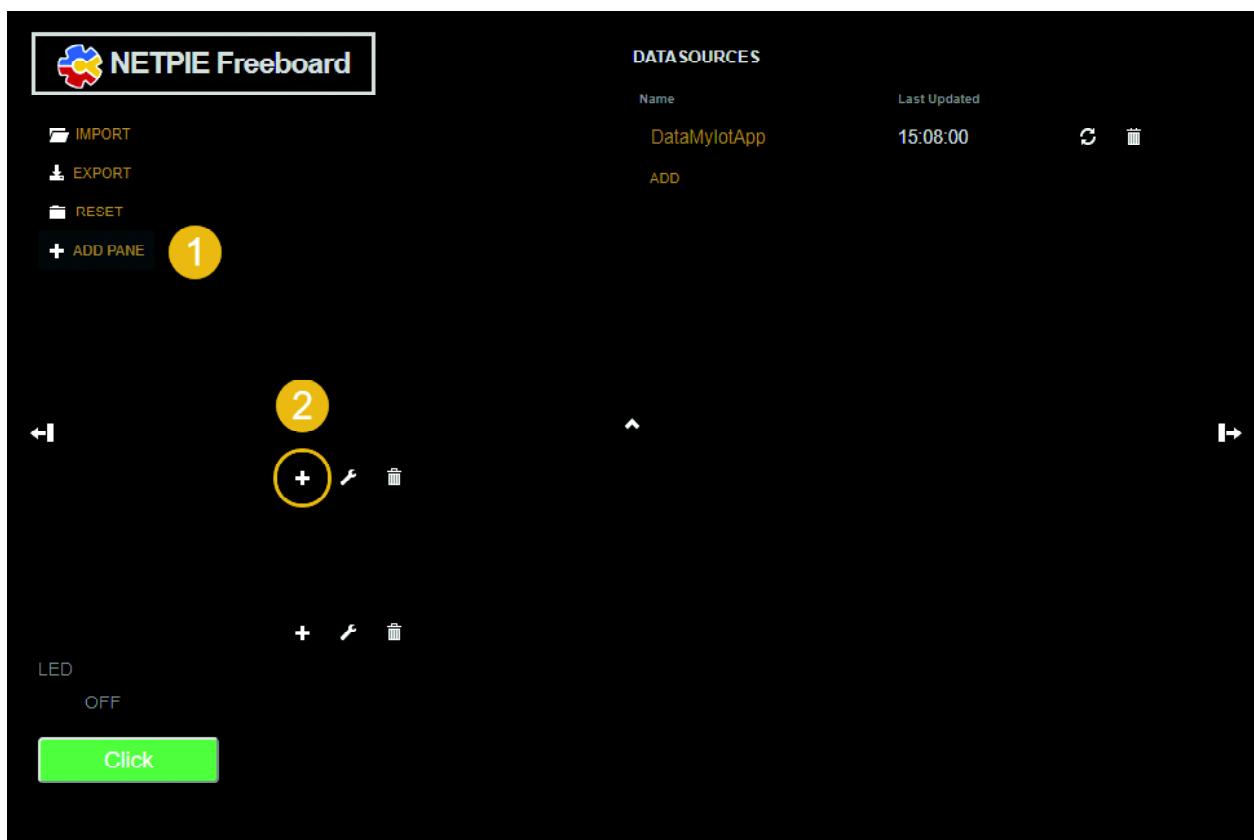


รูปที่ 6-29 ตัวอย่างของวิดเก็ต Toggle ที่ใช้งานบนหน้าเว็บ NETPIE Freeboard

#### ตัวอย่างการใช้งาน

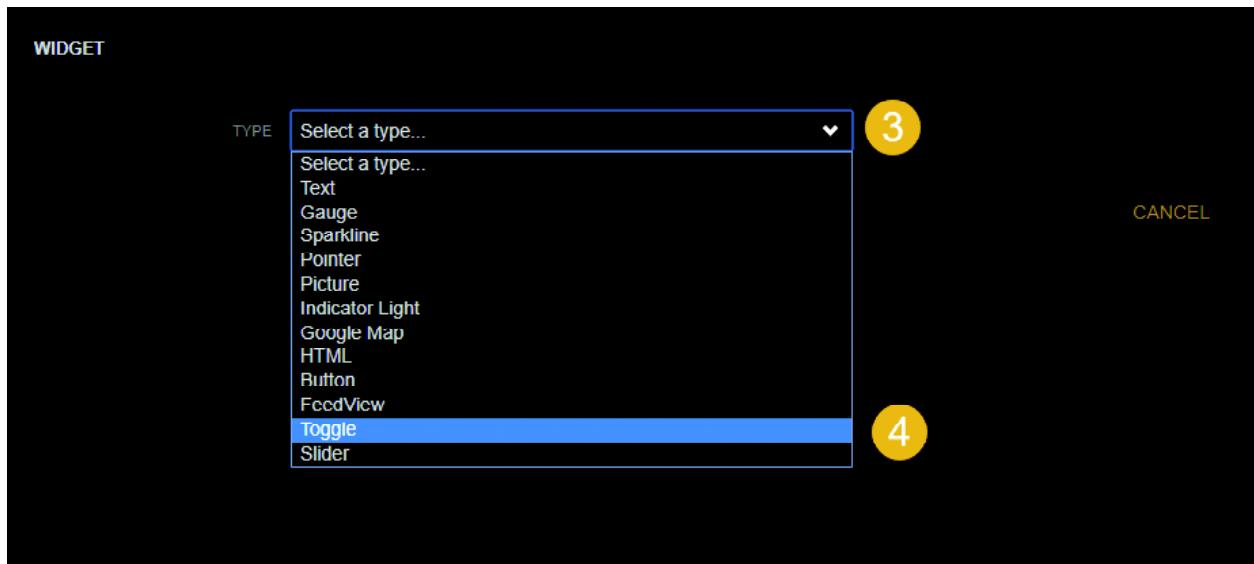
(6.2.4.1) เปิดโปรแกรม Arduino IDE เลือกบอร์ดเป็น **NodeMCU1.0 (ESP-12E module)** และเลือกพอร์ตเชื่อมต่อให้ถูกต้อง

(6.2.4.2) ใช้โปรแกรมที่ 6-3 ในการทดสอบเหมือนเดิม แต่ในส่วนของ NETPIE Freeboard จะเปลี่ยนมาใช้วิดเก็ต Toggle ในการแสดงผลลั่งงานแทน เริ่มจากกลับไปที่หน้าต่างหลักของ NETPIE Freeboard คลิกที่ **ADD PANE** (วงกลมหมายเลข 1 ในรูปที่ 6-30) จากนั้นคลิกเครื่องหมาย + เพื่อเพิ่มวิดเก็ตหรืออุปกรณ์แสดงผล (วงกลมหมายเลข 2 ในรูปที่ 6-30)



รูปที่ 6-30 การเพิ่มวิดเก็ต Toggle เพื่อใช้แสดงผลและสั่งงานอุปกรณ์บน NETPIE Freeboard

(6.2.4.3) จากนั้นคลิกเลือก TYPE ((วงกลมหมายเลข 3 ในรูปที่ 6-31) แล้วเลือกรายการ Toggle (วงกลมหมายเลข 4 ในรูปที่ 6-31)



รูปที่ 6-31 เลือกชนิดของวิดเก็ตเป็น Toggle

(6.2.4.4) หน้าต่างสำหรับตั้งค่าพารามิเตอร์ของวิดเก็ต Toggle ปรากฏขึ้นมา ตั้งค่าดังรูปที่ 6-32

TYPE : Toggle

TOGGLE CAPTION : Control LED (กำหนดชื่อของอุปกรณ์ - วงกลมหมายเลข 5)

TOGGLE STATE : datasources ["DataMyIotApp"] ["/GroupInex/gearname/ESPID\_02/LED"] == "0" ? 0 : 1 (เป็นการเลือก DATASOURCE เพื่อนำข้อมูลมาแสดงสถานะของ LED ที่ส่งเข้ามา ต้องปิดท้ายด้วยข้อความ ESPID\_02/LED - วงกลมหมายเลข 6)

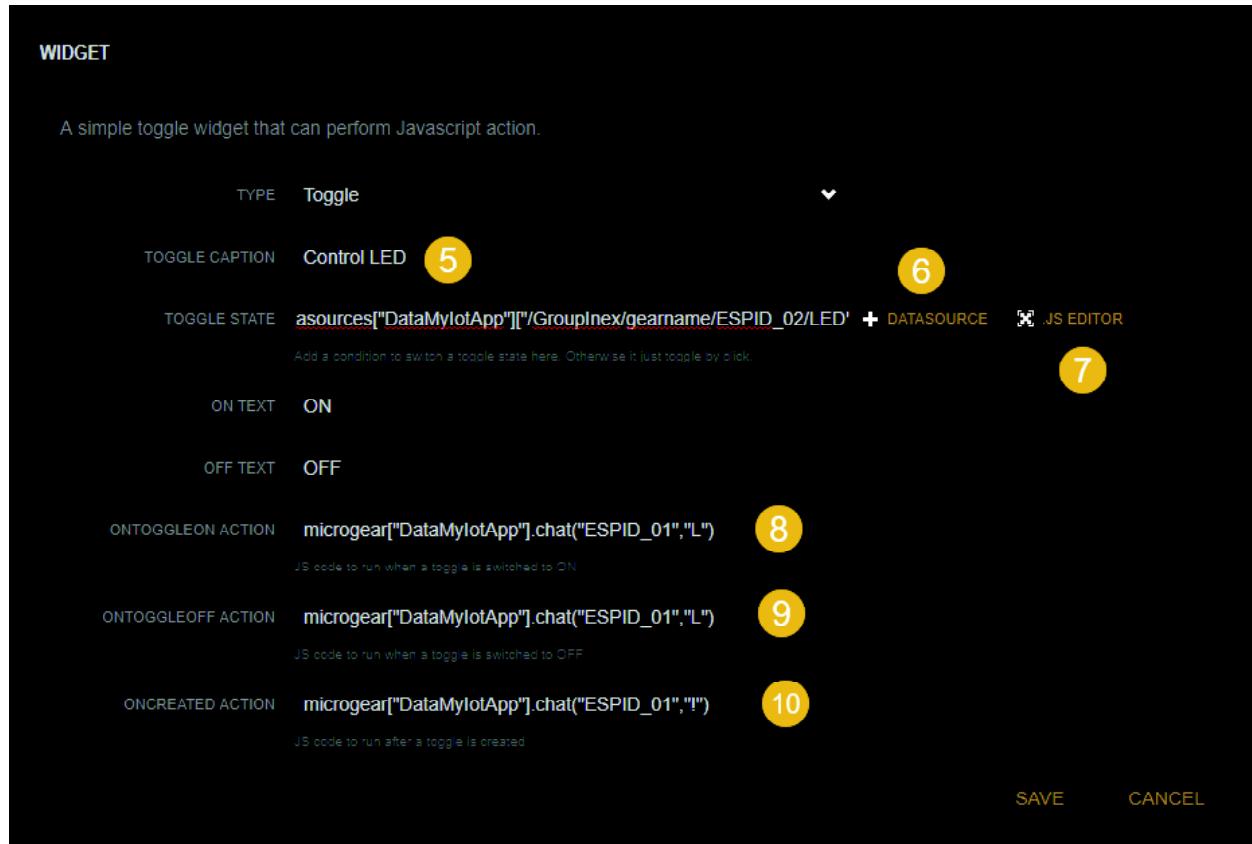
ON TEXT : ON

OFF TEXT : OFF

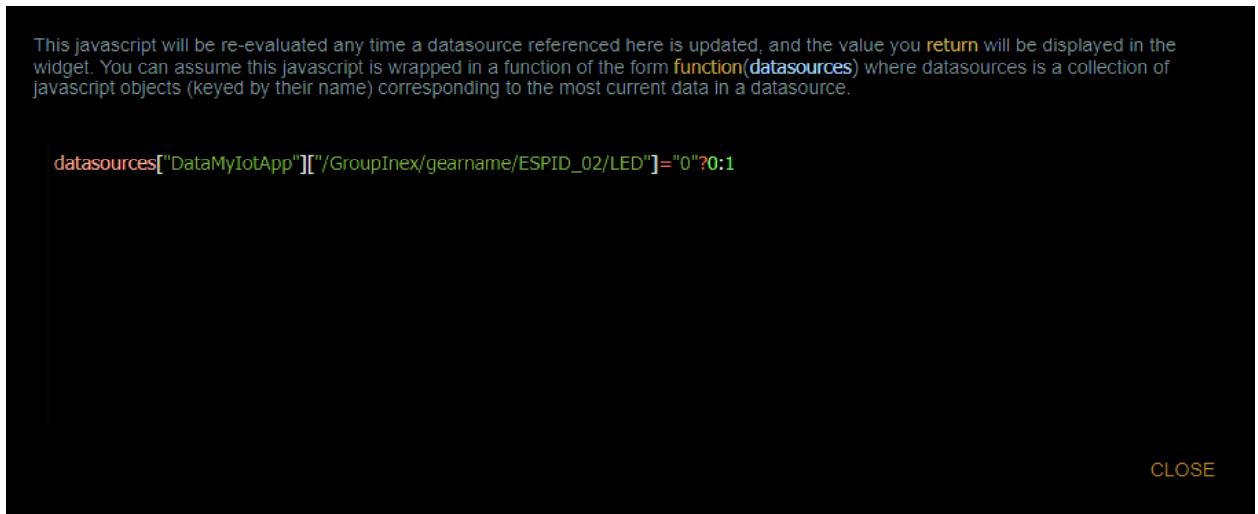
ONTOGGLEON ACTION : microgear ["DataMyIotApp"].chat ("ESPID\_01", "L")  
(เป็นการเพิ่มชุดคำสั่งส่งข้อความไปยังอุปกรณ์ที่ชื่อว่า ESPID\_01 เพื่อสั่งงานให้ LED ติด - วงกลมหมายเลข 8)

ONTOGGLEOFF ACTION : microgear ["DataMyIotApp"].chat ("ESPID\_01", "L")  
(เป็นการเพิ่มชุดคำสั่งส่งข้อความไปยังอุปกรณ์ที่ชื่อว่า ESPID\_01 เพื่อสั่งงานให้ LED ดับ - วงกลมหมายเลข 9)

ONCREATED ACTION : microgear ["DataMyIotApp"].chat ("ESPID\_01", "!")  
(เป็นการเพิ่มชุดคำสั่งเพื่อส่งไปถ้าสถานะ LED จากอุปกรณ์ที่ชื่อว่า ESPID\_01 - วงกลมหมายเลข 10)



รูปที่ 6-32 หน้าต่างตั้งค่าพารามิเตอร์ของวิดเก็ต Toggle



รูปที่ 6-33 หน้าต่าง JS EDITOR สำหรับแก้ไขโค้ดของ ONTOGGLEON ACTION ในวิดเก็ต Toggle เพื่อกำหนดการแสดงผลของ LED

นอกจากนี้ จะต้องทำการแก้ไขโค้ด Java Script ของ ONTOGGLEON ACTION ด้วย โดยคลิกที่หัวข้อ **.JS EDITOR** (วงกลมหมายเลข 7 ในรูปที่ 6-32) จากนั้นจะปรากฏหน้าต่างเอดิเตอร์ เพื่อแก้ไขโค้ด ให้ทำการแก้ไขเป็น `datasources["DataMyIotApp"]["/GroupInex/gearname/ESPID_02/LED"]=="0"?0:1` ดังรูปที่ 6-33 เมื่อแก้ไขเสร็จแล้วให้คลิกปุ่ม CLOSE จากนั้นจะกลับมาบังหน้าต่างแสดงพารามิเตอร์ของวิดเก็ต Toggle (รูปที่ 6-32) คลิกปุ่ม Save เพื่อบันทึก การตั้งค่าทั้งหมด

(6.2.4.5) เมื่อคลิกปุ่ม Save จะได้หน้าปัดของวิดเก็ต **Toggle Click** ที่มีทั้งปุ่มกดและ LED แสดงผลอยู่ในตัวเดียวกัน เมื่ອันกับรูปที่ 6-30 สำหรับใช้งานต่อไป

#### ทดสอบการทำงาน

(6.2.4.6) คลิกปุ่ม ON จากนั้นสังเกต LED ที่ต่อ กับขาพอร์ต D5 ของ NodeMCU-12E และรูป LED ในวิดเก็ต Toggle ที่อยู่บนหน้าเว็บ NETPIE Freeboard

(6.2.4.7) คลิกปุ่มช้าๆ อีกรึ่ง สังเกต LED ที่ต่อ กับขาพอร์ต D5 NodeMCU-12E และรูป LED ในวิดเก็ต Toggle ที่อยู่บนหน้าเว็บ NETPIE Freeboard

(6.2.4.8) กดสวิตซ์ที่ต่อ กับขาพอร์ต D3 ของ NodeMCU-12E สังเกตการทำงานของ LED ทั้งสองส่วน

(6.2.4.9) กดสวิตซ์เพื่อทำให้ LED ดับ จากนั้นทำการเรียกหน้าเว็บให้แสดงผลใหม่หรือ Refresh โดยกดคีย์ F5 จากนั้นสังเกตว่า LED ที่อยู่บนหน้าเว็บ NETPIE Freeboard

## 6.2.5 ใช้งาน Slider

Slider เป็นวิดเก็ตสำหรับกำหนดค่าตัวเลขเพื่อส่งไปยังอุปกรณ์ปลายทางที่ต้องการรับ พื้นที่ทั้งนำค่าที่ต้องการกลับมาแสดงในรูปแบบตัวเลื่อน ได้ด้วย ดังแสดงตัวอย่างในรูปที่ 6-34



รูปที่ 6-34 ตัวอย่างของวิดเก็ต Slider ที่ใช้งานบนหน้าเว็บ NETPIE Freeboard

### ตัวอย่างการใช้งานวิดเก็ต Slider เพื่อส่งการให้ NodeMCU-12E ขับ LED 3 สี ผ่าน NETPIE freeboard

มีขั้นตอนดังนี้

#### ขั้นตอนการฝึก NodeMCU-12E

(6.2.5.1) เปิดโปรแกรม Arduino IDE เลือกบอร์ดเป็น **NodeMCU1.0 (ESP-12E module)** และเลือกพอร์ตเชื่อมต่อให้ถูกต้อง

(6.2.5.2) พิมพ์โปรแกรมที่ 6-4 บันทึกไฟล์ในชื่อ **Netpie\_Control\_LED3C.ino**

(6.2.5.3) ใช้งานในรูปที่ 6-35 ในการทดลอง

(6.2.5.4) อัปโหลดโปรแกรมลงบน NodeMCU-12E จากนั้นคุณสามารถแสดงผลของ LED 3 สี RGB ที่ต่อ กับขา D0 (ขาอินพุตสีแดง), D1 (ขาอินพุตสีเขียว) และ D2 (ขาอินพุตสีน้ำเงิน) ของ NodeMCU-12E

#### ขั้นตอนการเพิ่ม Slider บน NETPIE Freeboard

(6.2.5.5) ในตัวอย่างนี้ นำเสนองานการสร้างวิดเก็ต Slider 3 ตัวเพื่อใช้ควบคุมการแสดงผลของสี แต่ละสีของ LED 3 สี RGB (ใช้แพงวงจร ZX-LED3C ในการทำงาน) เริ่มต้นด้วยไปที่กรอบแสดงผล หรือพานะ LED คลิกที่ **ADD PANE** (วงกลมหมายเลข 1 ในรูปที่ 6-36) ตามด้วยคลิกที่เครื่องหมาย + (วงกลมหมายเลข 2 ในรูปที่ 6-36)

```

/*Slider widget of NETPIE Freeboard example code with NodeMCU-12E */
#include <ESP8266WiFi.h>
#include <MicroGear.h>

const char* ssid      = "SSID";
const char* password = "Password";

#define APPID    "GroupInex"
#define KEY      "YnlyT7QqyKyYFk6"
#define SECRET   "KsnlhYJ2ELSNjsKb2iiJa6YIg"
#define ALIAS    "ESPID_01"

WiFiClient client;
int timer = 0;
int pinLED_B = D0;
int pinLED_G = D1;
int pinLED_R = D2;
int colorR = 500;
int colorG = 500;
int colorB = 500;

MicroGear microgear(client);

/* If a new message arrives, do this */
void onMsgHandler(char *topic, uint8_t* msg, unsigned int msglen)
{
    Serial.print("Incoming message --> ");
    msg[msglen] = '\0';
    Serial.println((char *)msg);
    String color;
    color.concat((char)msg[1]);
    color.concat((char)msg[2]);
    color.concat((char)msg[3]);
    color.concat((char)msg[4]);
    color.trim();
    if (msg[0] == 'R')
    {
        Serial.println("color R =" + color);
        colorR = color.toInt();
        analogWrite(pinLED_R, colorR);
        microgear.chat("ESPID_02/LED/R", colorR);
    }
    else if (msg[0] == 'G')
    {
        Serial.println("color G =" + color);
        colorG = color.toInt();
        analogWrite(pinLED_G, colorG);
        microgear.chat("ESPID_02/LED/G", color);
    }
}

```

**โปรแกรมที่ 6-4 ไฟล์ Netpie\_Control\_LED3C.ino สำหรับ NodeMCU-12E เพื่อทดสอบการทำงานของวิดเก็ต Slider ของ NETPIE Freeboard (มีต่อ)**

```

else if (msg[0] == 'B')
{
    Serial.println("color B =" + color);
    colorB = color.toInt();
    analogWrite(pinLED_B, colorB);
    microgear.chat("ESPID_02/LED/B", color);
}
else if (msg[0] == 'r')
{
    microgear.chat("ESPID_02/LED/R", colorR);
}
else if (msg[0] == 'g')
{
    microgear.chat("ESPID_02/LED/G", colorG);
}
else if (msg[0] == 'b')
{
    microgear.chat("ESPID_02/LED/B", colorB);
}
}

/* When a microgear is connected, do this */
void onConnected(char *attribute, uint8_t* msg, unsigned int msglen)
{
    Serial.println("Connected to NETPIE...");
    /* Set the alias of this microgear ALIAS */
    microgear.setAlias(ALIAS);
}

void setup()
{
    pinMode(pinLED_R, OUTPUT);
    pinMode(pinLED_G, OUTPUT);
    pinMode(pinLED_B, OUTPUT);
    analogWrite(pinLED_R, colorR);
    analogWrite(pinLED_G, colorG);
    analogWrite(pinLED_B, colorB);

    /* Add Event listeners */
    /* Call onMsgHandler() when new message arrives */
    microgear.on(MESSAGE, onMsgHandler);

    /* Call onConnected() when NETPIE connection is established */
    microgear.on(CONNECTED, onConnected);
    Serial.begin(115200);
    Serial.println("Starting...");
}

```

โปรแกรมที่ 6-4 ไฟล์ Netpie\_Control\_LED3C.ino สำหรับ NodeMCU-12E เพื่อทดสอบการทำงานของวิดเก็ต Slider ของ NETPIE Freeboard (มีต่อ)

```

/* Initial WIFI, this is just a basic method to configure WIFI on ESP8266.*/
if (WiFi.begin(ssid, password))
{
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
}
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());

/* Initial with KEY, SECRET and also set the ALIAS here */
microgear.init(KEY, SECRET, ALIAS);

/* connect to NETPIE to a specific APPID */
microgear.connect(APPID);
}

void loop()
{
    /* To check if the microgear is still connected */
if (microgear.connected())
{
    Serial.println("connected");
    /* Call this method regularly otherwise the connection may be lost */
    microgear.loop();
}
else
{
    Serial.println("connection lost, reconnect...");
    if (timer >= 5000)
    {
        microgear.connect(APPID);
        timer = 0;
    }
    else timer += 100;
}
delay(100);
}

```

### คำอธิบายการทำงานของโปรแกรมเพิ่มเติม

- ชุดคำสั่งที่รอรับการเปลี่ยนแปลงค่าสีอยู่ในฟังก์ชัน onMsgHandler ข้อความที่ส่งเข้ามามีรูปแบบดังนี้
 

Rxxxxx หมายถึง กำหนดค่าสีแดงด้วยข้อมูล xxxx มีค่า ๐ ถึง 1023 เช่น  
R512 หมายถึง ค่าของสีแดงเท่ากับ 512

**โปรแกรมที่ 6-4 ไฟล์ Netpie\_Control\_LED3C.ino สำหรับ NodeMCU-12E เพื่อทดสอบการทำงานของวิดเก็ต Slider ของ NETPIE Freeboard (มีต่อ)**

Gyyyy หมายถึง กำหนดค่าสีเป็นด้วยข้อมูล yyyy มีค่า 0 ถึง 1023 เช่น

G45 หมายถึง ค่าของสีเขียวเท่ากับ 45

Bzzzz หมายถึง กำหนดค่าสีน้ำเงินด้วยข้อมูล zzzz มีค่า 0 ถึง 1023 เช่น

B108 หมายถึง ค่าของสีน้ำเงินเท่ากับ 108

- การแยกตัวอักษรประจำสีและค่าสีออกจากกันใช้ฟังก์ชัน concat เพื่อนำข้อความแต่ละลำดับมาต่อกันให้เกิดเป็นค่าตัวแปรใหม่ที่มีเฉพาะค่าตัวเลขของสีเท่านั้น เช่น ค่าที่ส่งเข้ามาเป็น G1012

ลำดับ	0	1	2	3	4
-------	---	---	---	---	---

ข้อความ	'G'	'1'	'0'	'1'	'2'
---------	-----	-----	-----	-----	-----

ตัวอักษร G อยู่ในลำดับ 0 ดังนั้นค่าประจำสีจะอยู่ที่ลำดับที่ 1 ถึง 4 รูปแบบคำสั่งจึงเป็นดังนี้

```
String color;
color.concat((char)msg[1]);
color.concat((char)msg[2]);
color.concat((char)msg[3]);
color.concat((char)msg[4]);
color.trim(); // ตัดช่องว่างด้านข้อความและห้ายข้อความออก
```

- บุ๊กคำสั่งแยกค่าประจำในแต่ละสีมีดังนี้

#### ตัวอย่างแยกสีแดง

```
if (msg[0] == 'R')
{
    Serial.println("color R =" + color);
    colorR = color.toInt(); // แปลงข้อความเป็นค่าตัวเลข
    analogWrite(pinLED_R, colorR); // กำหนดความสว่างของสี
    microgear.chat("ESPID_02/LED/R", colorR); // ส่งค่ากลับไปยัง NETPIE
}
```

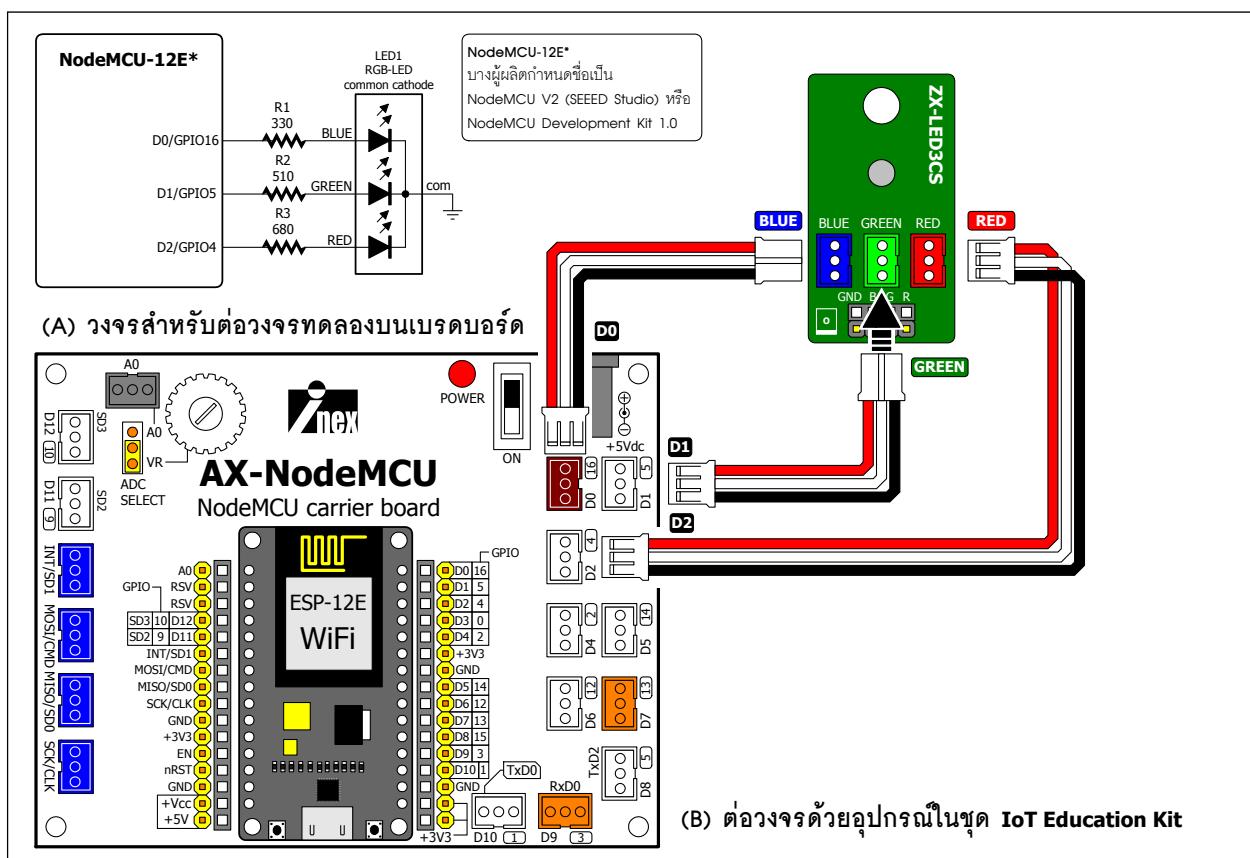
- บุ๊กคำสั่งส่งค่ากลับไปยัง NETPIE เมื่อกดเหตุการณ์เปิดหน้า Freeboard ขึ้นมาใหม่

#### ตัวอย่างสีแดง

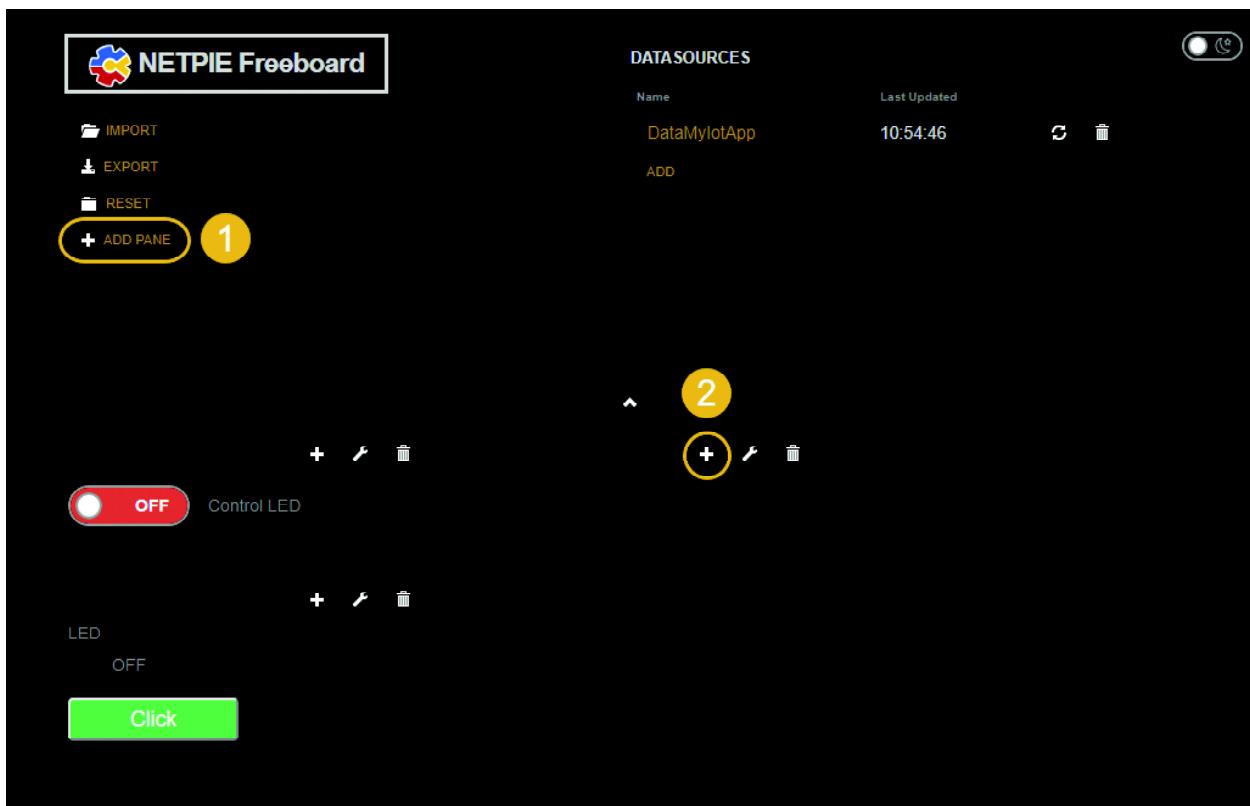
```
if (msg[0] == 'r')
{
    microgear.chat("ESPID_02/LED/R", colorR);
```

---

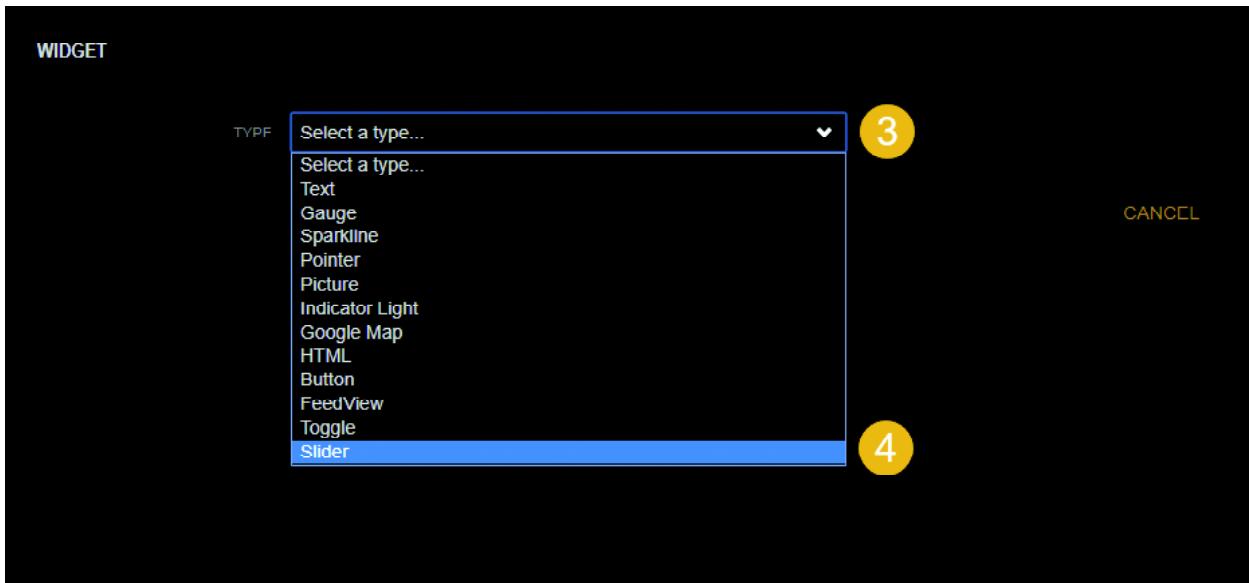
โปรแกรมที่ 6-4 ไฟล์ Netpie\_Control\_LED3C.ino สำหรับ NodeMCU-12E เพื่อทดสอบการทำงานของวิดเก็ต Slider ของ NETPIE Freeboard (มีต่อ)



รูปที่ 6-35 วงจรและการใช้บอร์ด AX-NodeMCU ในการทดสอบการใช้งานวิดเก็ต Slider บน NETPIE Freeboard



รูปที่ 6-36 เริ่มต้นเพิ่มวิดเก็ต Slider เพื่อใช้สั่งงานอุปกรณ์บน NETPIE Freeboard



รูปที่ 6-37 เลือกชนิดของวิดเก็ตเป็น Slider

(6.2.5.6) หน้าต่างสำหรับเลือกชนิดของวิดเก็ตแสดงขึ้นมา คลิกที่ช่อง **TYPE** (วงกลมหมายเลข 3 ในรูปที่ 6-37) จากนั้นเลื่อนรายการลงมาเลือก **Slider** (วงกลมหมายเลข 4 ในรูปที่ 6-37)

(6.2.5.6) หน้าต่างการตั้งค่าพารามิเตอร์แสดงขึ้นมา ทำการกำหนดค่าเพื่อสร้างวิดเก็ต Slider ของสีแดงก่อน ดังรูปที่ 6-38 ซึ่งมีรายละเอียดดังนี้

```

TYPE : Slider
SLIDER CAPTION : Red
FILLED COLOR : Red
MIN VALUE : 0
MAX VALUE : 1023
STEP : 1
INTIAL VALUE : 0
AUTO UPDATED VALUE : datasources["DataMyIotApp"] [/GroupIndex/
gearnname/ESPID_02/LED/R"]
ONSLIDER ACTION : microgear["DataMyIotApp"].chat("ESPID_01",'R'+value)
ONCREATED ACTION : microgear["DataMyIotApp"].chat("ESPID_01",'r')

```

คลิกปุ่ม Save

(6.2.5.7) จะได้วิดเก็ต Slider เพื่อปรับค่าแสงสีแดงดังรูปที่ 6-39 ทดสอบด้วยการลองปรับเลื่อนเพื่อเปลี่ยนค่าและทดลองกดปุ่ม Refresh ที่หน้าเว็บ NETPIE Freeboard เพื่อทดสอบว่า ยังคงเป็นค่าล่าสุดที่เลือกไว้หรือไม่

**WIDGET**

A slider widget that can perform Javascript action.

**TYPE** Slider

**SLIDER CAPTION** Red

**FILLED COLOR** Red

**DISPLAY VALUE** YES

**MIN VALUE** 0

**MAX VALUE** 1023

**STEP** 1

**INITIAL VALUE** 0

The default value set only the first time the widget is loaded.

**AUTO UPDATED VALUE** `sources["DataMyIoTApp"]"/"GroupIndex/gearname/ESPID_02/LED/R"]`

Slider will be updated upon the change of variables (e.g. other data sources).

**ONSTART ACTION**

Add some Javascript here. You can access to a slider attribute using variables 'value' and 'percent'.

**ONSLIDE ACTION** `microgear["DataMyIoTApp"].chat("ESPID_01",'R'+value)`

Add some Javascript here. You can access to a slider attribute using variables 'value' and 'percent'.

**ONSTOP ACTION**

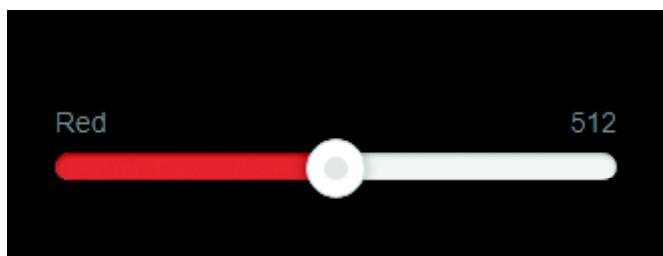
Add some Javascript here. You can access to a slider attribute using variables 'value' and 'percent'.

**ONCREATED ACTION** `microgear["DataMyIoTApp"].chat("ESPID_01",'I')`

JS code to run after a button is created

**SAVE** **CANCEL**

รูปที่ 6-38 หน้าต่างตั้งค่าพารามิเตอร์ของวิดเก็ต Slider



รูปที่ 6-39 วิดเก็ต Slider สำหรับปรับค่าแสงสีแดงของ LED 3 สี RGB



The screenshot shows the Serial Monitor window titled "COM4". The window displays a series of messages indicating communication between the NodeMCU-12E and a device (labeled "R507, R536, R544"). The messages include "connected", "Incoming message --> R507", "color R =507", "connected", "Incoming message --> R536", "color R =536", "connected", "connected", "Incoming message --> R544", "color R =544", "connected", and "Incoming message --> R544", "color R =544", "connected". The bottom of the window shows settings for "No line ending" and "115200 baud" with a "Clear output" button.

รูปที่ 6-40 หน้าต่าง Serial Monitor แสดงการทำงานของ NodeMCU-12E เมื่อได้รับและส่งค่าของแสง สีแดงจาก NETPIE Freeboard เพื่อนำไปขับ LED 3 สี RGB บนแพงวงจร ZX-LED3C ให้แสดงสีแดง โดยมีความสว่างตามการปรับค่าที่วิดเก็ต Slider ของสีแดง

(6.2.5.8) หากต้องการตรวจสอบค่าที่ส่งไปยัง NETPIE Freeboard ของ NodeMCU-12E ทำได้โดยการเปิดหน้าต่าง Serial Monitor เลือกอัตราบaud เป็น 115,200 บิตต่อวินาที จะได้ผลดังรูปที่ 6-40

(6.2.5.9) จากนั้นทำการเพิ่ม Slider ให้ครบทั้ง 3 สี โดยค่าพารามิเตอร์ของวิดเก็ต Slider ต้องเป็นสีเขียวและสีน้ำเงินมีรายละเอียดดังนี้

#### พารามิเตอร์ของวิดเก็ต Slider สีเขียว

```

TYPE : Slider
SLIDER CAPTION : Green
FILLED COLOR : Green
MIN VALUE : 0
MAX VALUE : 1023
STEP : 1
INTIAL VALUE : 0
AUTO UPDATED VALUE : datasources["DataMyIotApp"] [/GroupIndex/
gearname/ESPID_02/LED/G"]
ONSLIDER ACTION : microgear["DataMyIotApp"].chat("ESPID_01",'G'+value)
ONCREATED ACTION : microgear["DataMyIotApp"].chat("ESPID_01",'g')

```

### พารามิเตอร์ของวิดเก็ต Slider สีน้ำเงิน

```

TYPE : Slider
SLIDER CAPTION : Blue
FILLED COLOR : Blue
MIN VALUE : 0
MAX VALUE : 1023
STEP : 1
INTIAL VALUE : 0
AUTO UPDATED VALUE : datasources["DataMyIotApp"] [/GroupIndex/
gearname/ESPID_02/LED/B"]
ONSLIDER ACTION : microgear["DataMyIotApp"].chat("ESPID_01",'B'+value)
ONCREATED ACTION : microgear["DataMyIotApp"].chat("ESPID_01",'b')

```

(6.2.5.10) เมื่อสร้างครบจะได้หน้าปัดแสดงผลดังรูปที่ 6-34

(6.2.5.11) ทดลองปรับค่าบนวิดเก็ต Slider ทั้ง 3 ตัว LED 3 สี RGB บนแพงวงจร ZX-LED3C ที่ต่อ กับ NodeMCU-12E จะสว่างและเปลี่ยนสีตามการเปลี่ยนแปลงค่าของ Slider เมื่อสั่งงานผ่าน NETPIE Freeboard !ท่านนั้น

## 6.3 สั่งท้าย

ทั้งหมดที่นำเสนอในหนังสือเล่มนี้เป็นการแนะนำให้ผู้สนใจเริ่มต้นพัฒนาอุปกรณ์ IoT ได้รู้จักกับการใช้งานคลาวด์เซิร์ฟเวอร์ NETPIE ที่พัฒนาโดยคณะนักพัฒนาชาวไทย ภายใต้การสนับสนุนโดยสำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ หรือ สวทช. จะเห็นว่า NETPIE มีความสามารถที่ดีเพียงพอและใช้งานไม่ยาก มีไลบรารีที่ช่วยให้อุปกรณ์ทางสารคดware สมัยใหม่อย่าง ESP8266 หรือ NodeMCU ให้สามารถเชื่อมต่อกับ NETPIE ได้สะดวก รวมถึง ไลบรารีสำหรับ NodeJS และ HTML5 เพื่อรองรับการพัฒนาแอปพลิเคชันของอุปกรณ์ IoT ด้วย

ผู้สนใจศึกษาข้อมูลเพิ่มเติมรวมถึงเข้าชมตัวอย่างและดาวน์โหลดไฟล์ไลบรารีติดตามได้ที่ <https://netpie.io> รวมถึงการลงทะเบียนเพื่อเข้าใช้งานด้วย และติดต่อทางอีเมลได้ที่ support@netpie.io

ขอขอบคุณคณะนักพัฒนา NETPIE ภายใต้การดูแลโดย ดร.พนิตา พงษ์ไพบูลย์ นักวิจัยจากห้องปฏิบัติการวิจัยเทคโนโลยีเครือข่ายเน็ตเวก ที่ช่วยกันสร้างสรรค์ให้ NETPIE คือ แพลตฟอร์ม IoT เพื่อนักพัฒนาและอุตสาหกรรมไทย



