



# เริ่มต้นใช้งาน NETPIE กับ ESP8266/NodeMCU-12E

ธีรวุฒิ จิตราพรหมา<sup>๕</sup>  
ชัยวัฒน์ ลิ้มพrajitrawi/l



[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

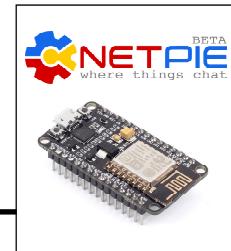
---

บทที่ 1 การใช้งาน NETPIE กับ NodeMCU-12E.....	7
บทที่ 2 ไลบรารีของ NETPIE สำหรับการติดต่อแบบ MQTT.....	25
บทที่ 3 ความรู้เบื้องต้นเกี่ยวกับ NETPIE REST API .....	53
บทที่ 4 การประยุกต์ใช้งาน NETPIE ร่วมกับ freeboard.io เพื่อแสดงผลการทำงาน.....	61
บทที่ 5 การประยุกต์ใช้งาน freeboard เพื่อสั่งงานอุปกรณ์ผ่าน NETPIE.....	75



# บทที่ 1

## เริ่มต้นใช้งาน NETPIE กับ NodeMCU-12E



### 1.1 ข้อมูลเบื้องต้นของ NETPIE

#### 1.1.1 อะไรคือ NETPIE?

NETPIE คือ คลาวด์เซิร์ฟเวอร์ที่ให้บริการในรูปแบบ Platform-as-a-Service เพื่ออำนวยความสะดวกให้กับนักพัฒนาในการพัฒนาอุปกรณ์ของตัวเอง เช่นต่อและแลกเปลี่ยนข้อมูลกันได้ในแบบ Internet of Things หรือ IoT โดยคำว่า *NETPIE* มาจาก *Network Platform for Internet of Everything*

#### 1.1.2 ทำไมต้องเลือกใช้ NETPIE?

##### 1.1.2.1 ช่วยลดการใช้ทรัพยากรของการเชื่อมต่อ :

NETPIE ช่วยให้อุปกรณ์สามารถคุยกันได้โดยผู้พัฒนาไม่ต้องกังวลว่า อุปกรณ์นั้นจะอยู่ที่ใด เพียงนำไลบรารีของ NETPIE ไปติดตั้งในอุปกรณ์ NETPIE จะรับหน้าที่ดูแลการเชื่อมต่อให้ทั้งหมด ไม่ว่าอุปกรณ์นั้นจะอยู่ในเครือข่ายชนิดใด ลักษณะใด หรือแม้กระทั่งเคลื่อนย้ายไปอยู่ที่ใด ผู้พัฒนาสามารถตัดปัญหากวนใจในการที่จะต้องมาออกแบบการเข้าถึงอุปกรณ์จากระยะไกล (remote access) ด้วยวิธีการเดิมๆ เช่น การใช้ fixed public IP หรือการตั้ง port forwarding ในเราเตอร์ หรือการต้องไปลงทะเบียนกับผู้ให้บริการ dynamic DNS ซึ่งทั้งหมดล้วนมีความยุ่งยากและลดความยืดหยุ่นของระบบ ไม่เพียงเท่านั้น NETPIE ยังช่วยให้การเริ่มต้นใช้งานเป็นไปโดยง่าย โดยออกแบบให้อุปกรณ์ถูกตั้งเป็นและเข้าสู่บริการโดยอัตโนมัติ (automatic discovery, plug and play)

##### 1.1.2.2 ช่วยลดภาระด้านความปลอดภัยของข้อมูล :

NETPIE ถูกออกแบบให้มีระดับและสิทธิ์ในการเข้าถึงในระดับ fine grain กล่าวคือ ผู้พัฒนาสามารถออกแบบได้เองทั้งหมด เช่น สิ่งใดมีสิทธิ์คุยกับสิ่งใด สิ่งใดมีสิทธิ์หรือไม่-เพียงใดในการอ่าน หรือเขียนข้อมูล และสิทธิ์เหล่านี้จะมีอายุการใช้งานเท่าไหร หรือถูกเพิกถอนภายใต้เงื่อนไขใด เป็นต้น

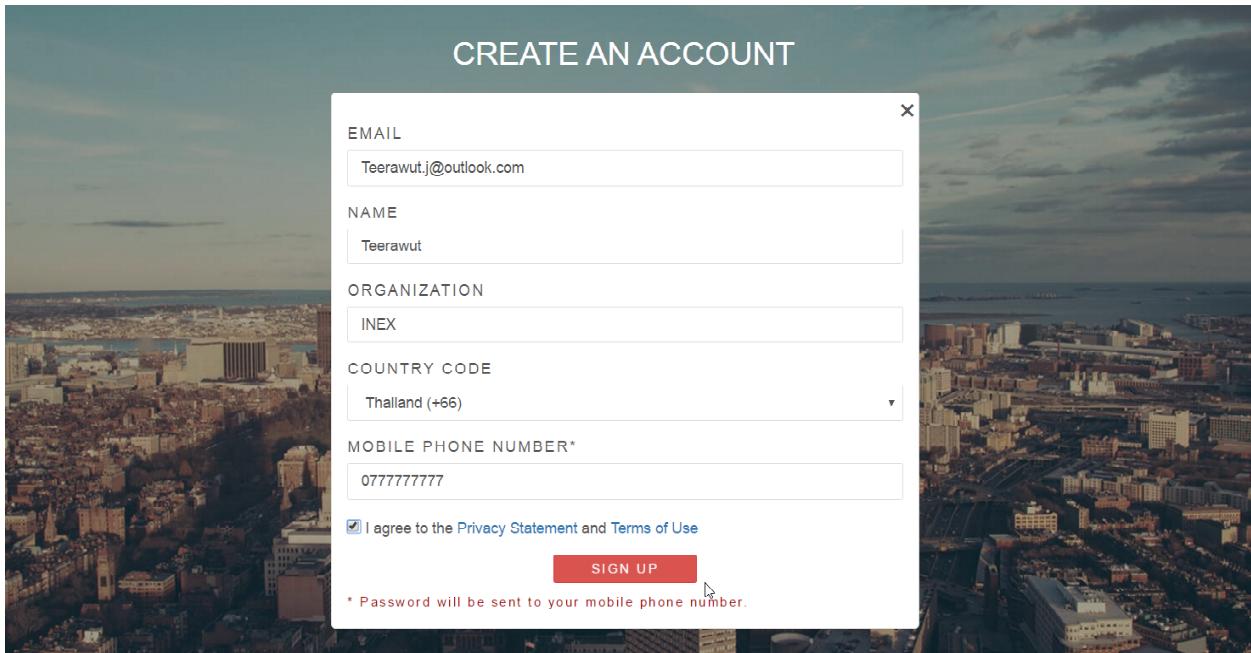
##### 1.1.2.3 ยืดหยุ่นต่อการขยายระบบ

NETPIE มีสถาปัตยกรรมเป็นคลาวด์เซิร์ฟเวอร์ย่างแท้จริงในทุกระดับของระบบ ทำให้เกิดความยืดหยุ่นและคล่องตัวสูงในการขยายตัว นอกจากนี้ ไม่ต้องต่อสาย ยังถูกออกแบบให้ทำงานแยกจากกัน เพื่อให้เกิดสภาพภาวะ loose coupling และสื่อสารกันด้วยวิธีการ asynchronous messaging ช่วยให้แพล็ตฟอร์มมีความน่าเชื่อถือสูง นำไปใช้ช้าและพัฒนาต่อได้ง่าย ดังนั้นผู้พัฒนาไม่จำเป็นต้องกังวลกับการขยายตัวเพื่อรับโหลดที่เพิ่มขึ้นในระบบอีกต่อไป

## 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



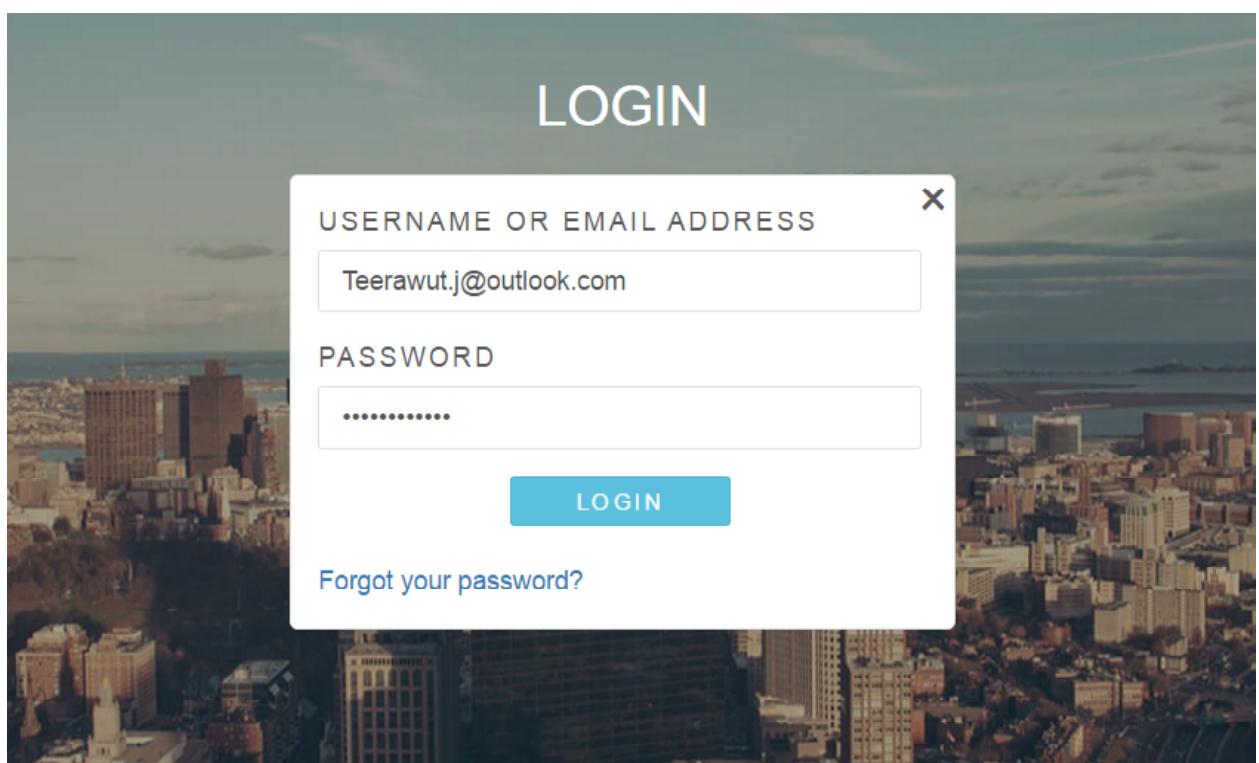
รูปที่ 1-1 หน้าต่าง CREATE AN ACCOUNT สำหรับกรอกข้อมูลเพื่อลงทะเบียนใช้งาน NETPIE

(1.2.1.2) รอรับข้อความสั้นหรือ SMS จากทาง NETPIE ซึ่งส่งไปยังหมายเลขโทรศัพท์เคลื่อนที่ที่ลงทะเบียนไว้

#### ตัวอย่าง SMS

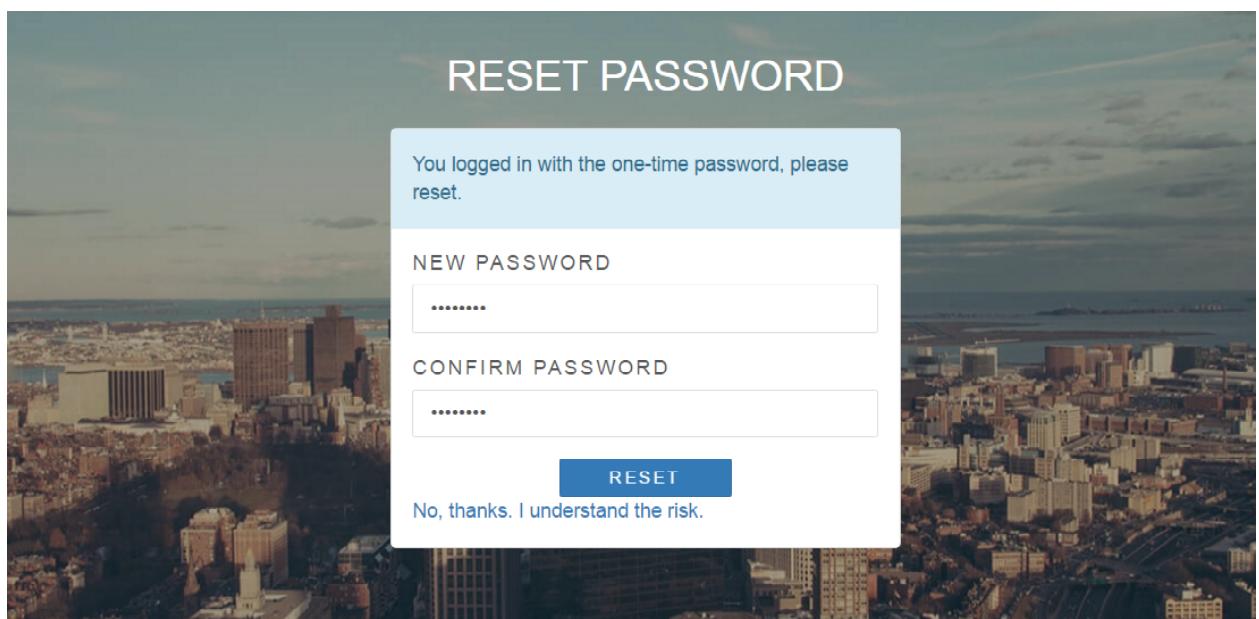
Your one-time password for NETPIE is 536815059323

(1.2.1.3) คลิกที่เมนู **LOG IN** เพื่อเข้าสู่ระบบ นำอีเมล์ที่ลงทะเบียนไว้ใส่ในช่อง **USERNAME OR EMAIL ADDRESS** และนำรหัสผ่านที่ได้รับจาก SMS ใส่ในช่อง **PASSWORD** แล้วคลิกปุ่ม **LOGIN** ดังรูปที่ 1-2

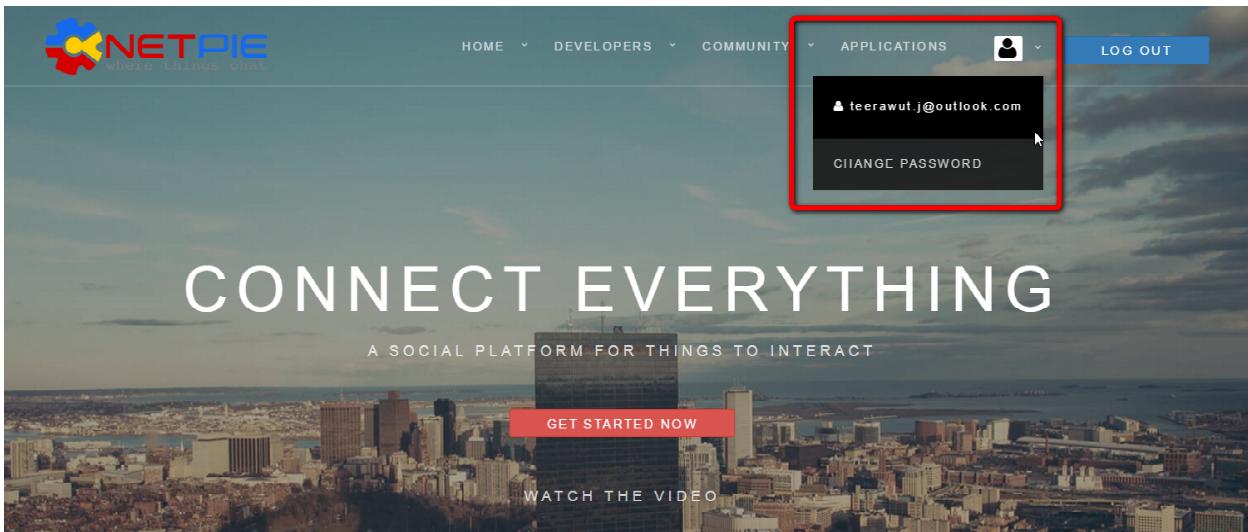


รูปที่ 1-2 หน้าต่าง LOGIN เพื่อเข้าสู่ระบบ

(1.2.1.4) เข้าสู่หน้าต่าง RESET PASSWORD เพื่อเปลี่ยนรหัสผ่านใหม่ให้จำได้ง่ายขึ้น ดังรูปที่ 1-3



รูปที่ 1-3 หน้าต่างเปลี่ยนรหัสผ่านใหม่



รูปที่ 1-4 แสดงไอคอนที่เกิดขึ้นหลังจากล็อกอินเข้าสู่ระบบของ NETPIE

(1.2.1.5) เมื่อลงทะเบียนเสร็จแล้ว จะปรากฏไอคอนผู้ใช้ ถ้าเลื่อนมาลงไปชี้ จะแสดง USERNAME หรือ E-mail ที่ได้ล็อกอินไว้

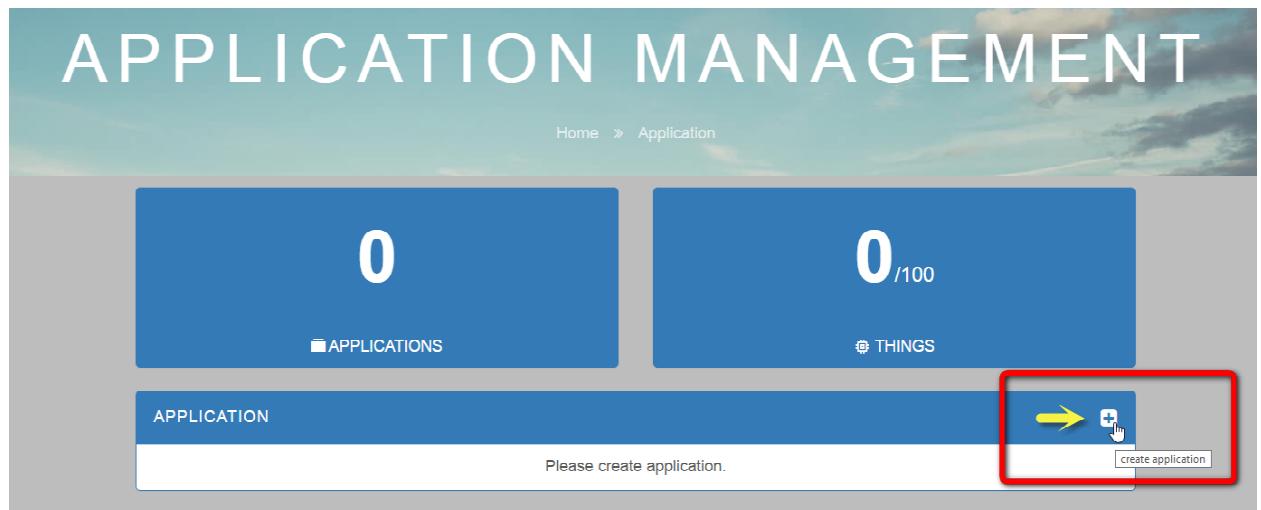
## 1.2.2 สร้างรหัสแอปพลิเคชันประจำตัวหรือ Application ID (AppID)

เมื่อสมัครใช้งานและล็อกอินเข้าสู่ระบบแล้ว ลำดับต่อไปคือ การสร้างรหัสแอปพลิเคชันประจำตัวหรือ Application ID (AppID) ซึ่งผู้ใช้งานแต่ละคน (ตรวจสอบจากหมายเลขโทรศัพท์เคลื่อนที่) จะได้ค่านะสูงสุด 10 AppID และในแต่ละ AppID มีอุปกรณ์หรือเกียร์ (gear) ได้สูงสุด 100 ตัว

(1.2.2.1) คลิกที่เมนู APPLICATIONS เพื่อเข้าสู่การสร้าง AppID ดังรูปที่ 1-5



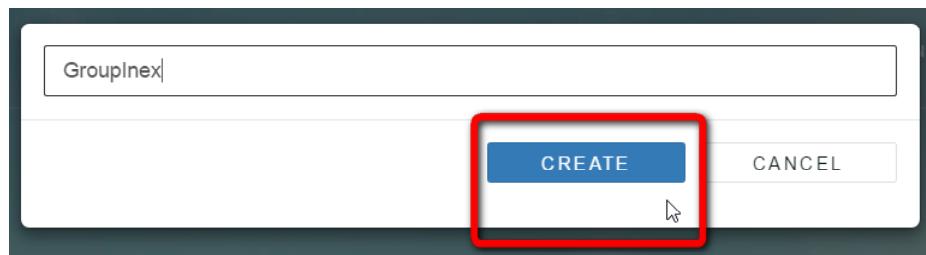
รูปที่ 1-5 เลือกเมนู APPLICATION เพื่อเริ่มต้นสร้าง Application ID (AppID)



รูปที่ 1-6 เริ่มต้นสร้าง AppID

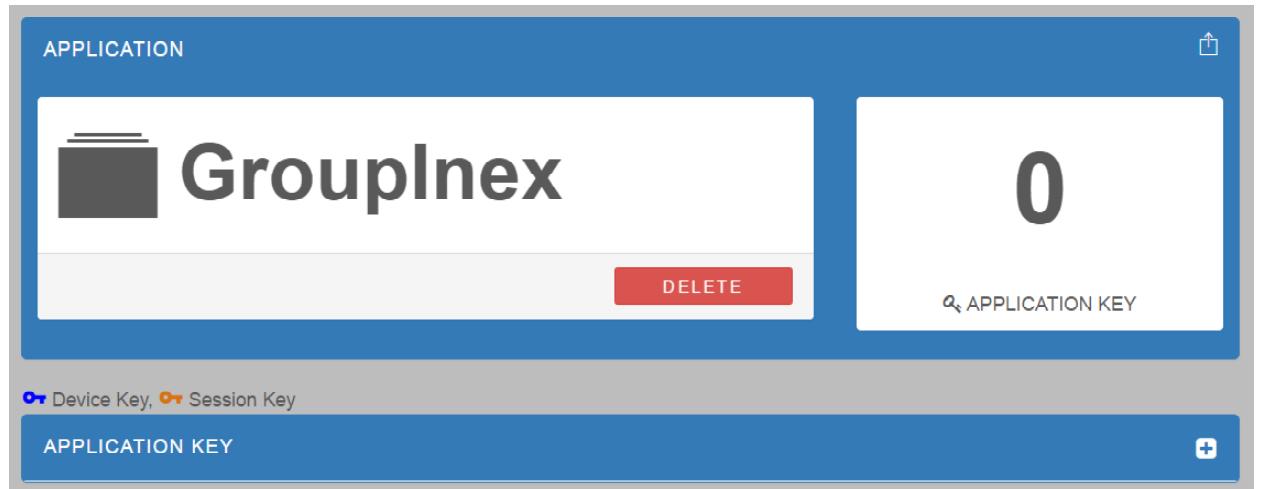
(1.2.2.2) เข้าสู่หน้าต่างคลิกที่รูปเครื่องหมายบวกเพื่อสร้าง AppID ดังรูปที่ 1-6

(1.2.2.3) กำหนดชื่อ AppID ตามต้องการในตัวอย่างรูปที่ 1-7 ใช้ชื่อว่า **GroupInex** จากนั้นคลิกปุ่ม **CREATE** เพื่อสร้าง AppID การตั้งชื่อ AppID จะต้องไม่ซ้ำกับคนอื่น ต้องพยายามเลือกชื่อที่เฉพาะตัวจริงๆ และต้องไม่ลืม เมื่องจาก AppID นี้จะถูกนำไปใช้ในการพัฒนาโปรแกรมต่อไป



รูปที่ 1-7 กำหนดชื่อ AppID

(1.2.2.4) ที่หน้าต่าง APPLICATION จะแสดงช่องของชื่อ AppID ที่สร้างขึ้น ดังรูปที่ 1-8

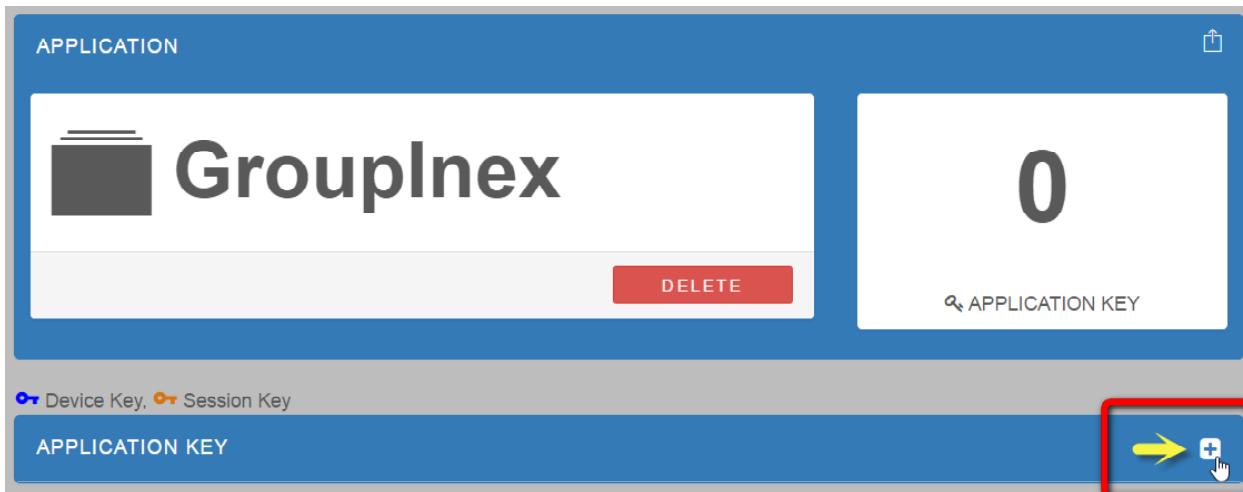


รูปที่ 1-8 หน้าต่าง APPLICATION ของ AppID ที่ชื่อ GroupInex

### 1.2.3 สร้างชื่อของแอปพลิเคชัน

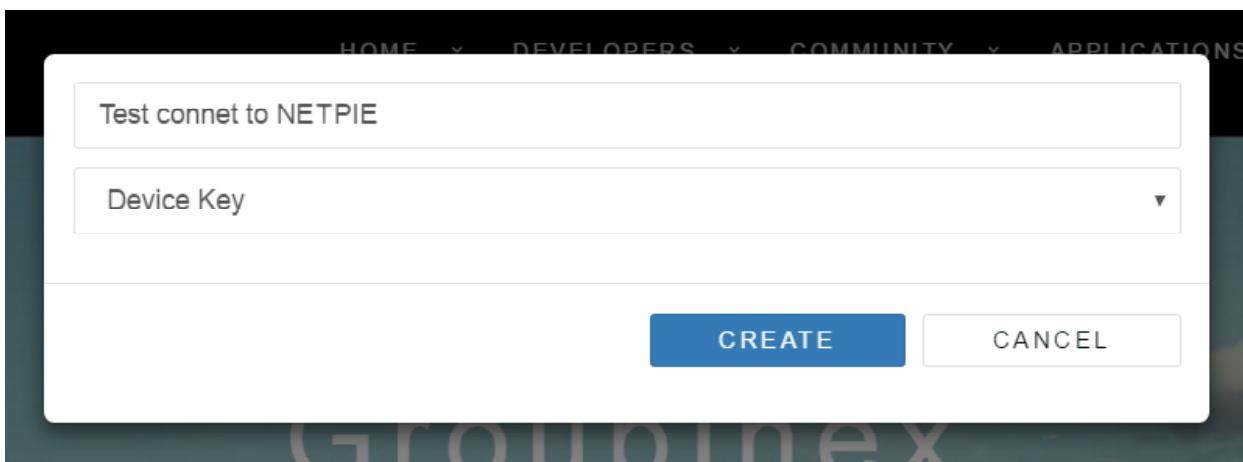
เมื่อได้ AppID แล้ว ต่อไปเป็นการสร้างชื่อของแอปพลิเคชัน

(1.2.3.1) คลิกที่รูปเครื่องหมายบากดังรูปที่ 1-9

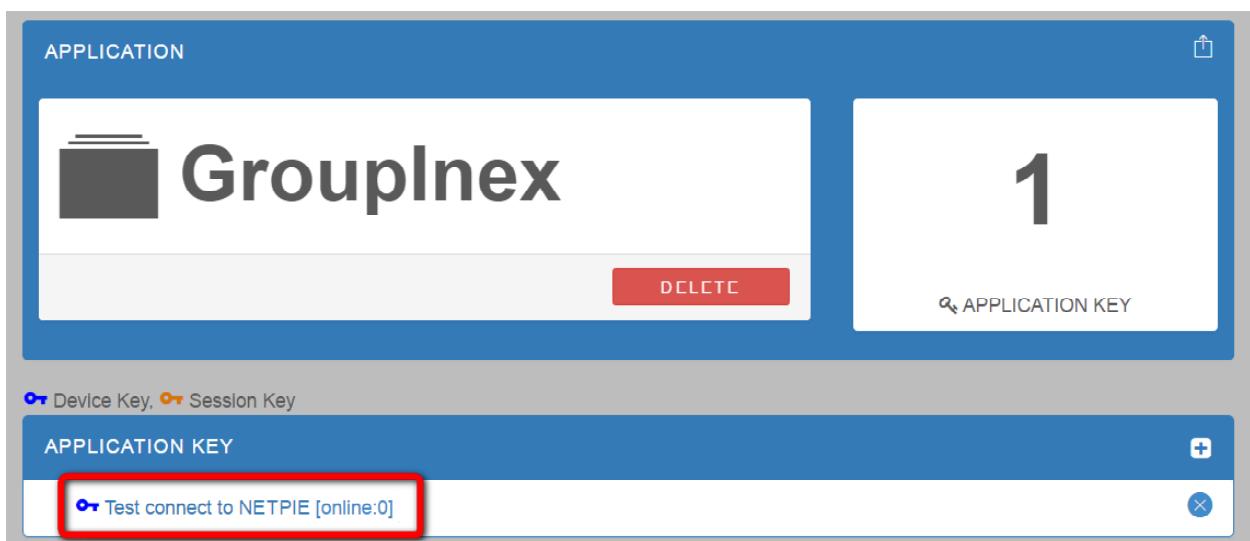


รูปที่ 1-9 เริ่มต้นสร้างชื่อของแอปพลิเคชัน

(1.2.3.2) กำหนดชื่อของแอปพลิเคชันตามต้องการ ในตัวอย่างใช้ข้อความว่า **Test connect to NETPIE** และเลือกรูปแบบของรหัสกุญแจหรือคีย์เป็น **Device Key** ดังรูปที่ 1-10 จากนั้นคลิกที่ปุ่ม **CREATE** เพื่อสร้างแอปพลิเคชัน



รูปที่ 1-10 กำหนดชื่อของแอปพลิเคชัน



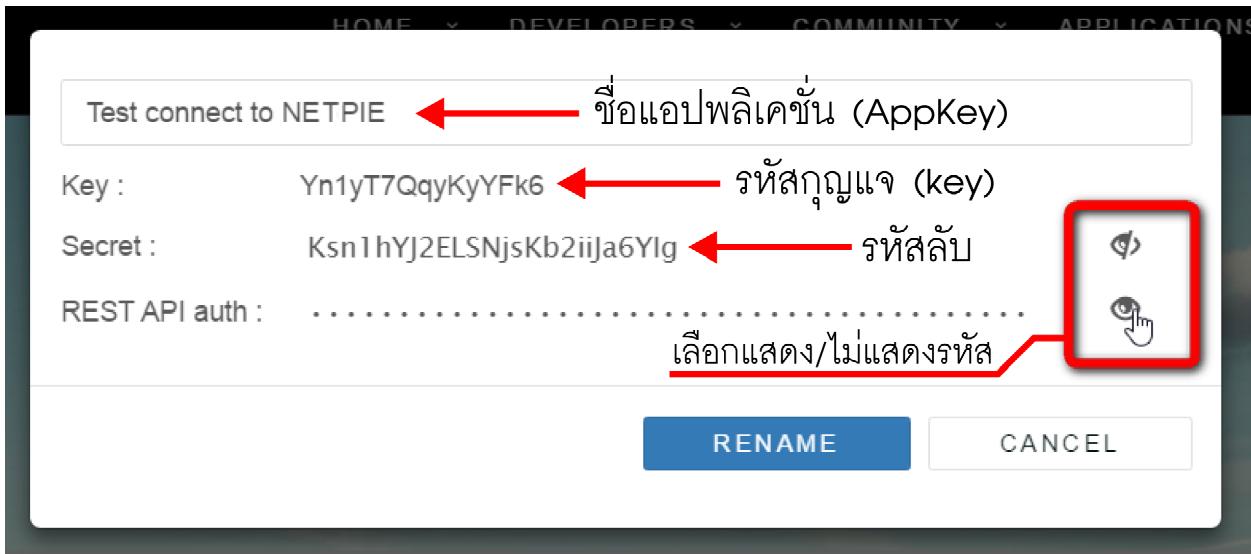
รูปที่ 1-11 แสดงข้อของแอปพลิเคชันที่สร้างขึ้นบนหน้าต่าง AppID ของ GroupInex และมีเลขแสดงจำนวน แอปพลิเคชันที่เริ่มนับหนึ่ง

(1.2.33) เมื่อสร้างเสร็จแล้ว ข้อที่สร้างขึ้นจะปรากฏขึ้นที่หน้าต่าง APPLICATION ดังรูปที่ 1-11

(1.2.3.4) ตรวจสอบรหัสต่างๆ ของข้อของแอปพลิเคชัน ซึ่งมีด้วยกัน 2 ตัวคือ รหัสคุณเจหรือ Application Key (AppKey) และรหัสลับหรือ Application Secret (AppSecret) ที่ใช้แสดงตัวตน โดยการคลิกที่ข้อของแอปพลิกัน (ในที่นี้คือ Test connect to NETPIE) ดังรูปที่ 1-12 จะปรากฏรหัสต่างๆ ที่จำเป็นต้องใช้ดังรูปที่ 1-13 สำหรับรหัสบางตัว เช่น Secret และ REST API auth ระบบจะปกปิดไว้ หากต้องการดูให้คลิกที่รูปดวงตา รหัสจะปรากฏขึ้นมา ดังรูปที่ 1-13



รูปที่ 1-12 การเลือกดูรหัสของแอปพลิเคชันที่อุปกรณ์ต้องนำไปใช้ในการติดต่อกับแอปพลิเคชันบน NETPIE



รูปที่ 1-13 แสดงรหัสของแอปพลิเคชันที่อุปกรณ์ต้องนำไปใช้ในการติดต่อกับ NETPIE

(1.2.3.5) จากรูปที่ 1-12 ข้อมูลที่ต้องจดจำและนำไปใช้งานมี 3 ตัวคือ AppID, AppKey และ AppSecret จากรูปสรุปได้ดังนี้

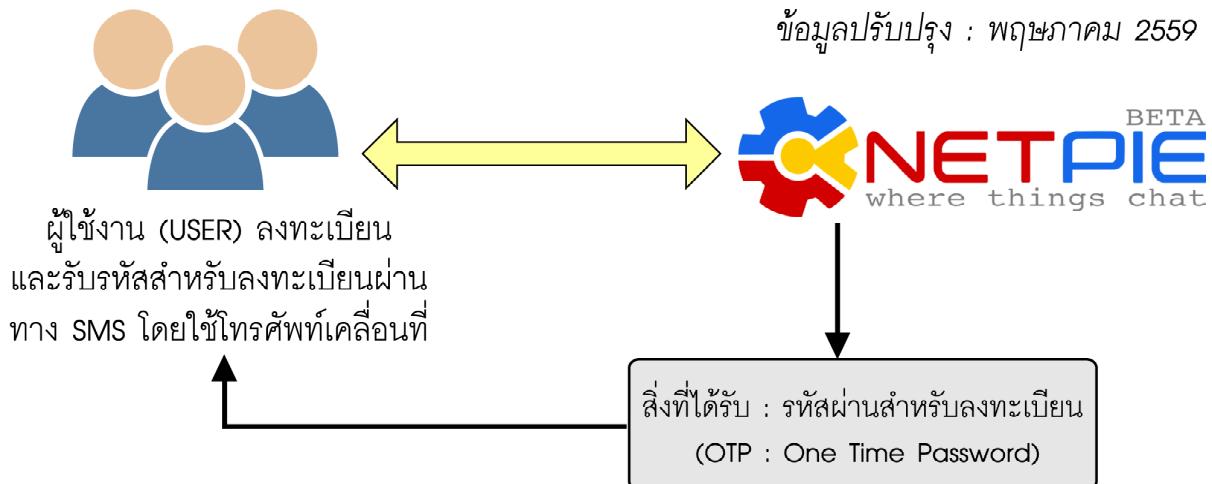
**AppID = GroupIndex**

**AppKey= Yn1yT7QqyKyYFk6**

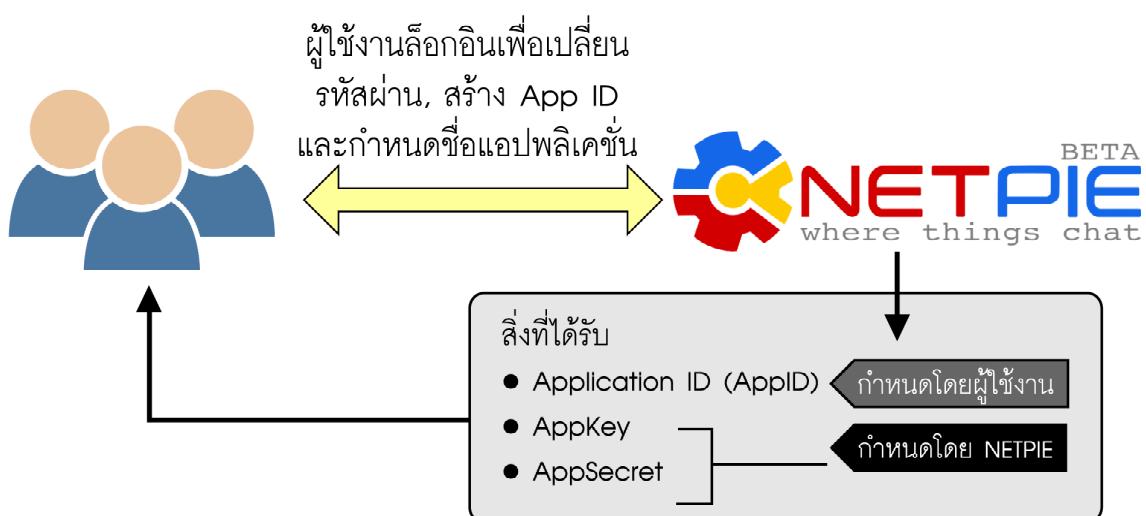
**AppSecret= Ksn1hYJ2ELSNjsKb2iiJa6YIg**

ในรูปที่ 1-14 แสดงภาพรวมของการตั้งค่าบน NETPIE เพื่อนำรหัสต่างๆ ไปใช้ในการติดต่อกับอุปกรณ์硬件ที่ต้องการเชื่อมต่อกับ NETPIE

## ลงทะเบียน



## สร้าง App ID รับ AppKey และ AppSecret



- ผู้ใช้งานที่ลงทะเบียนไว้ จะได้รับ App ID 10 รหัส ต่อหมายเลขโทรศัพท์เคลื่อนที่ที่ใช้ลงทะเบียน
- ใน 1 App ID รองรับอุปกรณ์ได้ 100 อุปกรณ์ โดยไม่มีค่าใช้จ่าย หากต้องการมากกว่านี้ ถือว่าต้องการเป็นผู้ใช้งานในระดับ Enterprise ต้องติดต่อกับ สวทช. โดยตรงเพื่อย้ายจำนวน ซึ่งอาจมีค่าใช้จ่ายได้

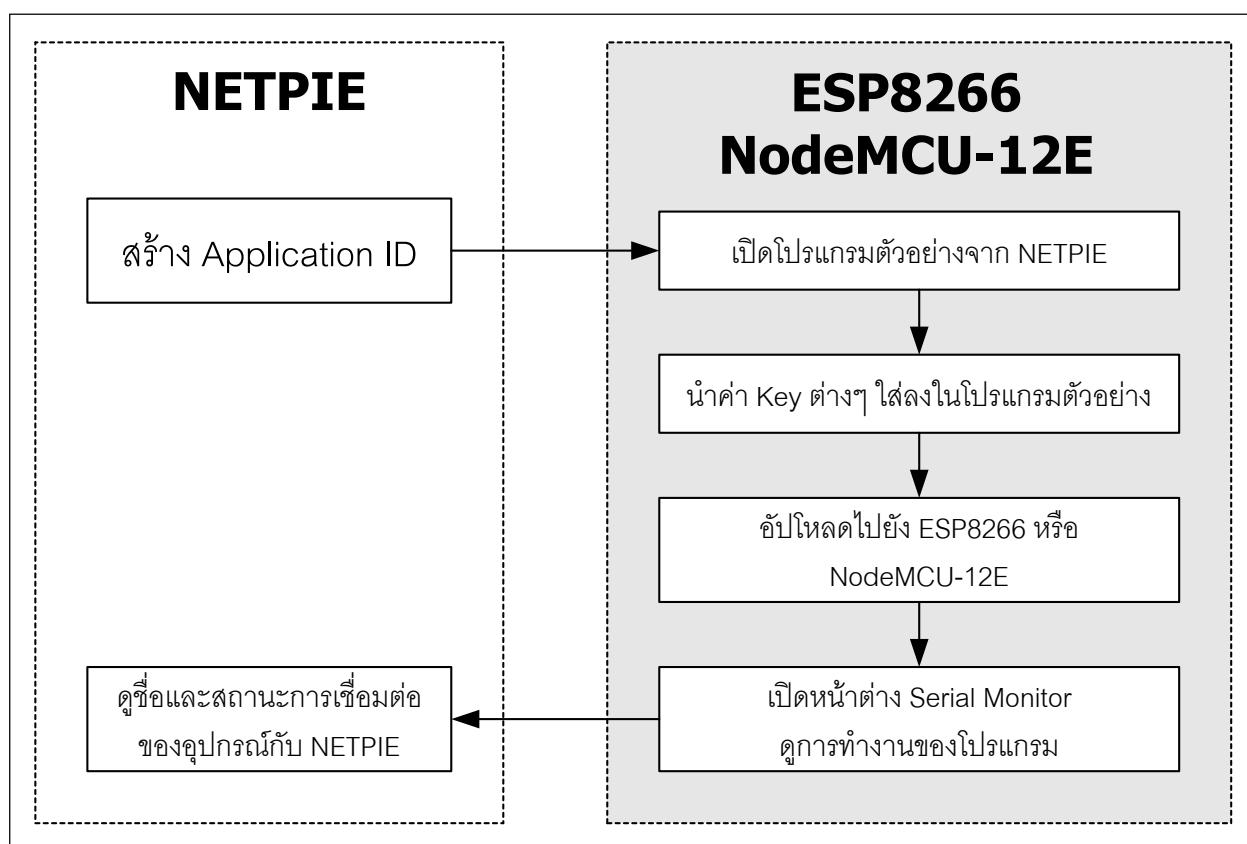
รูปที่ 1-14 ได้อะแกรมสรุปขั้นตอนการลงทะเบียนจนถึงการเตรียมการที่ NETPIE เพื่อนำรหัสสำคัญไปใช้ในการติดต่อกับแอปพลิเคชันบน NETPIE

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

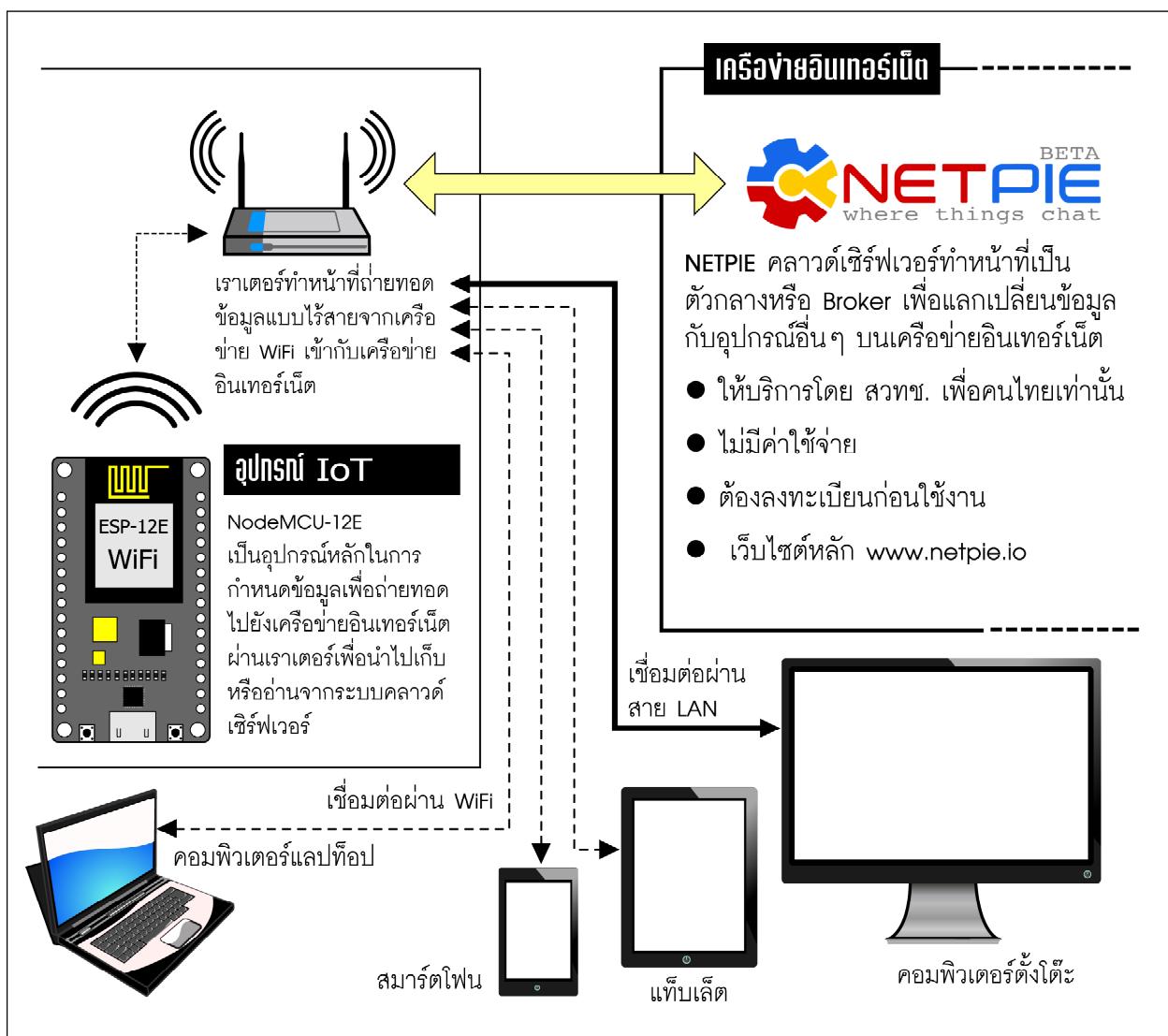
ทาง NETPIE ได้จัดทำไลบรารีที่ใช้งานกับโมดูล WiFi คอนโทรลเลอร์ยอดนิยมและราคาประหยัดอย่าง ESP8266 ที่ใช้ Arduino IDE ในการพัฒนาโปรแกรมไว้แล้ว เพื่อให้ผู้พัฒนาใช้งานได้อย่างสะดวกมากขึ้น โดยดาวน์โหลดไลบรารีได้จาก <https://github.com/netpieio/microgear-esp8266-arduino/archive/master.zip>

เมื่อดาวน์โหลดมาแล้ว ทำการแตกไฟล์เก็บไว้ที่โฟลเดอร์ที่ชื่อว่า **libraries** ของ Arduino IDE ที่ใช้ในการพัฒนา ESP8266 หรือ NodeMCU-12E ในที่นี้ติดตั้ง Arduino 1.6.5 ดังนั้นโฟลเดอร์สำหรับเก็บไลบรารีจะอยู่ที่ C:\Arduino1.6.5r5\libraries

ในรูปที่ 1-15 แสดงขั้นตอนการพัฒนาโปรแกรมเพื่อติดต่อและใช้งาน NETPIE ของ NodeMCU-12E หรือโมดูล ESP8266 ในแบบอื่นๆ การอธิบายขั้นตอนการพัฒนาจะอ้างอิงกับรูปที่ 1-15 นี้เป็นหลัก ส่วนการเชื่อมต่อทาง硬件แล้วแสดงดังรูปที่ 1-16



รูปที่ 1-15 ไดอะแกรมแสดงขั้นตอนการพัฒนาโปรแกรมเพื่อติดต่อและใช้งาน NETPIE ของ NodeMCU-12E หรือโมดูล ESP8266 ในอนุกรมอื่นๆ



รูปที่ 1-16 ไดอะแกรมแสดงภาพรวมของการเชื่อมต่อ NodeMCU เพื่อใช้งานกับ NETPIE คลาวเซิร์ฟเวอร์ร่วมกับอุปกรณ์อื่น ๆ ทั้งคอมพิวเตอร์, สมาร์ตโฟน และแท็บเล็ต

## 18 ● การใช้งาน NETPIE กับ NodeMCU-12E

```
#include <AuthClient.h>           // แนวโน้มไลบรารีของ NETPIE 4 ไฟล์
#include <MicroGear.h>
#include <MQTTClient.h>
#include <SHA1.h>
#include <Arduino.h>             // ไลบรารีสำหรับการพัฒนาด้วย Arduino IDE
#include <ESP8266WiFi.h>          // ไลบรารีของ ESP8266 สำหรับพัฒนาด้วย Arduino
#include <EEPROM.h>
#include <MicroGear.h>            // แนวโน้มไลบรารี 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>           // กำหนดรหัสศิษย์ของอุปกรณ์ที่เชื่อมต่อ 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) // ฟังก์ชันตรวจสอบการปรากฏของข้อความ
{
    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");
}
```

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

```

void setup()
{
  /* Event listener */
  microgear.on(MESSAGE,onMsgHandler);
  microgear.on(PRESENT,onFoundgear);
  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(".");
      // พิมพ์เครื่องหมาย . แสดงสถานะการเชื่อมต่อ
    }
  }
  Serial.println("WiFi connected"); // แสดงข้อความแจ้งการเชื่อมต่อ WiFi สำเร็จ
  Serial.println("IP address: ");
  // แสดงหมายเลข IP แอดเดรสของเครือข่าย WiFi ที่เชื่อมต่อ
  Serial.println(WiFi.localIP());

  //microgear.resetToken(); // หากต้องการรีเซ็ตโทเคนให้ปลดเครื่องหมาย // ออก
  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");
      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);
}

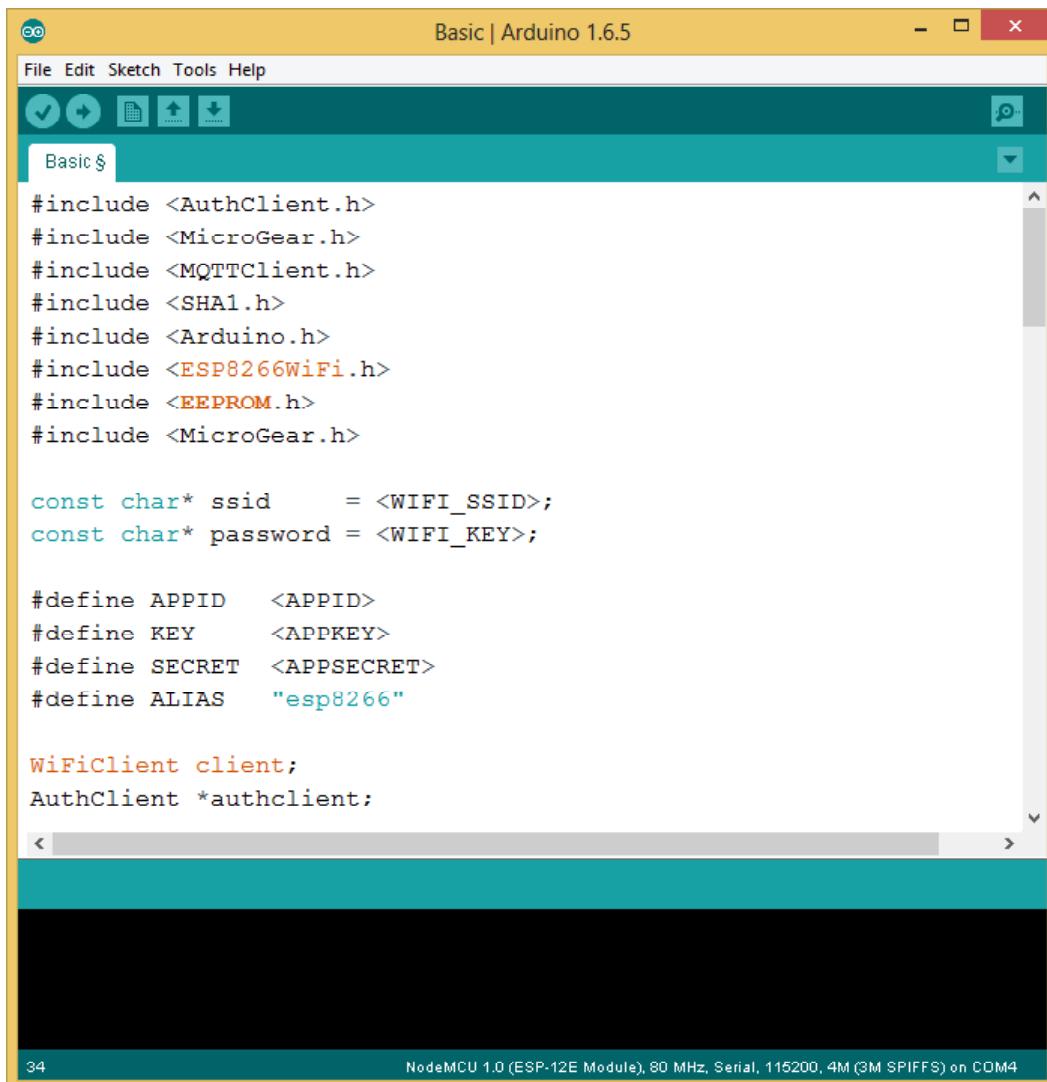
```

---

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

ตัวอย่างขั้นตอนการพัฒนาโปรแกรมบน Arduino IDE สำหรับ NodeMCU-12E มีดังนี้

(1.3.1) เปิดโปรแกรม Arduino IDE 1.6.5r5 เปิดไฟล์ตัวอย่าง โดยไปที่ **File>Examples>ESP8266 Microgear>Basic** จะปรากฏหน้าต่างของโค้ดดังรูปที่ 1-17 ส่วนรายละเอียดของโค้ดแสดงในโปรแกรมที่ 1-1



```
#include <AuthClient.h>
#include <MicroGear.h>
#include <MQTTClient.h>
#include <SHA1.h>
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <EEPROM.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;
AuthClient *authclient;
```

รูปที่ 1-17 หน้าต่างหลักของ Arduino IDE เส้นทางโค้ดตัวอย่าง

(1.3.2) นำรหัสต่างๆ ใส่ลงโปรแกรมที่ 1-1 ดังแสดงในรูปที่ 1-18

*ssid* คือ ชื่อ WiFi ที่ต้องการเชื่อมต่อ

*password* คือ รหัสผ่านของ WiFi ที่ต้องการเชื่อมต่อ

```

Basic | Arduino 1.6.5
File Edit Sketch Tools Help
Basic §
#include <AuthClient.h>
#include <MicroGear.h>
#include <MQTTClient.h>
#include <SHA1.h>
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <EEPROM.h>
#include <MicroGear.h>

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

#define APPTID    "GroupInex"
#define KEY       "Yn1yT7QqyKyYFk6"
#define SECRET   "Ksn1hYJ2ELSNjsKb2iiJa6YIg"
#define ALIAS     "ESP8266-12E"

WiFiClient client;
AuthClient *authclient;

```

NodeMCU 1.0 (ESP-12E Module), 80 MHz, Serial, 115200, 4M (3M SPIFFS) on CUM4

รูปที่ 1-18 แสดงโค้ดที่ได้รับการกำหนดค่าพารามิเตอร์และรหัสต่าง ๆ เพื่อใช้ในการเชื่อมต่อกับ NETPIE

(1.3.3) เชื่อมต่อ NodeMCU-12E กับพอร์ต USB ของคอมพิวเตอร์ กำหนดค่าการเชื่อมต่อให้ถูกต้อง

(1.3.4) อัปโหลดโค้ดไปยัง NodeMCU-12E ซึ่งในขั้นตอนนี้อาจใช้เวลาพอสมควร

(1.3.5) เมื่ออัปโหลดเสร็จแล้ว ทำการเปิดหน้าต่าง Serial Monitor และหน้าเว็บที่สร้าง AppID จากขั้นตอนในหัวข้อ (1.2) เพื่อสังเกตการทำงานของโปรแกรม จะได้ผลการทำงานดังรูปที่ 1-19

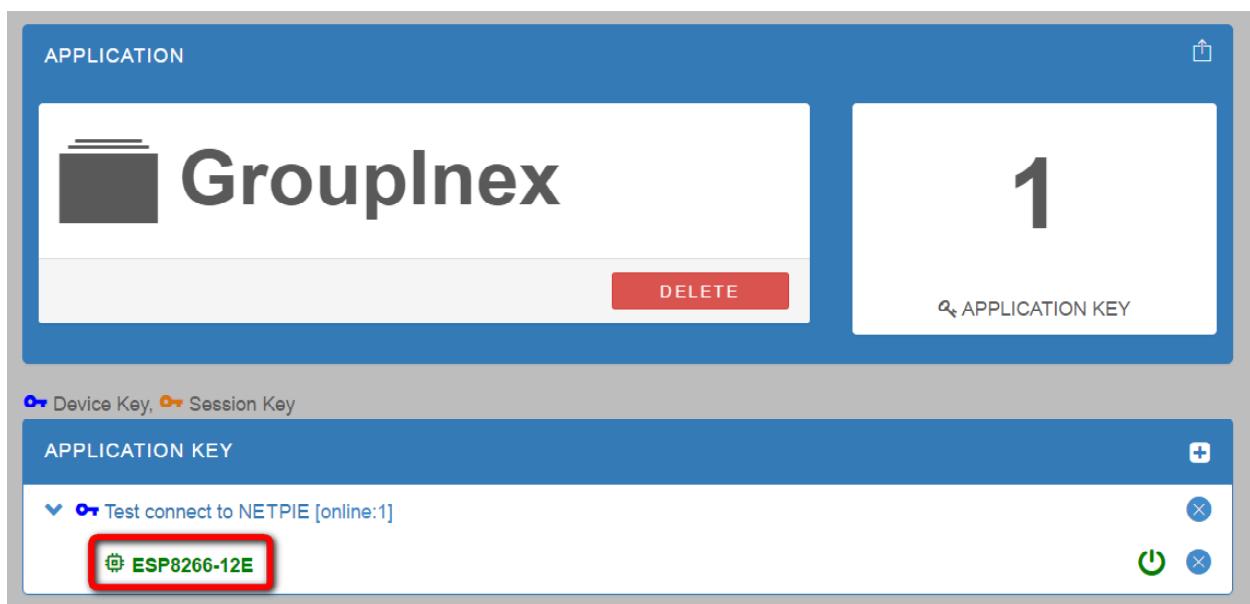
เมื่อเกิดการการส่งค่าไปยัง NETPIE แสดงข้อความ "Publish..."

เมื่อเกิดข้อมูลส่งเข้าไปยัง NETPIE แสดงข้อความ Incoming message → Hello

หน้าเว็บที่สร้าง AppID แสดงผลดังรูปที่ 1-20

The screenshot shows a Windows-style application window titled "COM4". The main area contains a text log of serial communication. The log starts with "..... WiFi connected" followed by "IP address: 192.168.1.46". It then shows a timestamp "Time stamp : 1461036683" and a token state "Token rom state == C;". The text continues with "Going to connect to MQTT broker" and a long string of random characters: "9Wh1GbEb3Th5mBWA%Yn1yT7QqyKyYFk6%1461036688 Z7kkk8WT4GeijBzgn0UpAsZPKWQ= gb.netpie.io:1883". It then says "Connecting to : gb.netpie.io:1003" and "Connected to NETPIE...". Below this, there is a series of "connected" messages followed by "Publish...". An incoming message is shown as "Incoming message --> Hello". At the bottom of the window, there are controls for "Autoscroll", "No line ending", and a baud rate selection set to "115200 baud".

รูปที่ 1-19 หน้าต่าง Serial Monitor แสดงการทำงานเมื่อ NodeMCU-12E เชื่อมต่อกับ NETPIE ได้



รูปที่ 1-20 หน้าเว็บของ NETPIE ที่แสดงสถานะการติดต่อระหว่างอุปกรณ์ชื่อในที่นี่คือ ESP8266 (Node MCU-12E) กับ Application ID – GroupInex

(1.3.6) ไปที่หน้าเว็บ APPLICATION ของ AppID ที่ชื่อ GroupInex ดังรูปที่ 1-20

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

ทั้งหมดที่นำเสนอในบทนี้เป็นการเริ่มต้นใช้งานและทดสอบการทำงานระหว่างอุปกรณ์ สารคดแวร์กับคลาวด์เซิร์ฟเวอร์ที่ชื่อ NETPIE จะเห็นได้ว่า มีขั้นตอนไม่นักและไม่ซับซ้อน จึงเหมาะสมอย่างยิ่งสำหรับการเรียนรู้เพื่อนำไปสู่การพัฒนาอุปกรณ์ IoT ด้วยตัวเอง





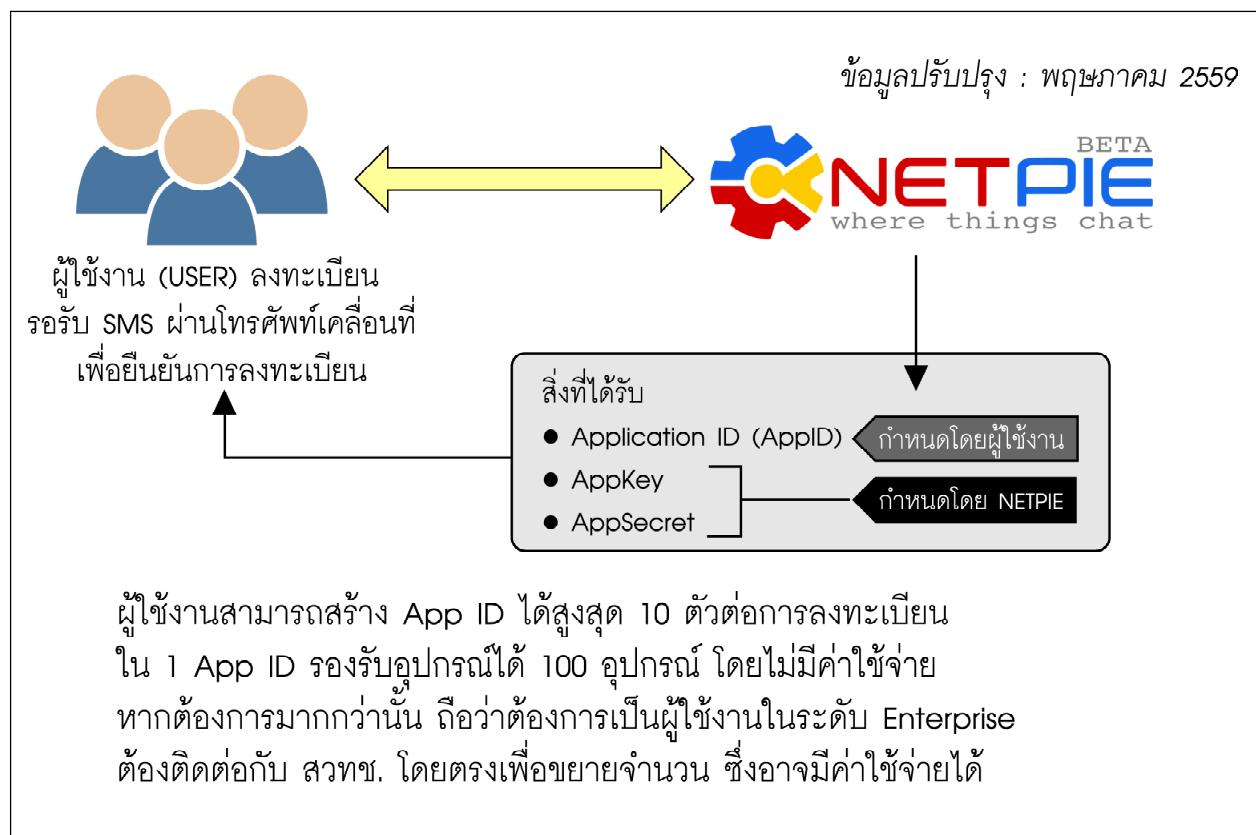
## บทที่ 2

# ไลบรารีของ NETPIE สำหรับการติดต่อแบบ MQTT

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

### 2.1 นิยามที่ควรทราบเบื้องต้น

ในการอธิบายเกี่ยวกับ ไลบรารี **microgear** จะมีคำศัพท์เฉพาะมากพอสมควร ซึ่งจะได้ทำการอธิบายศัพท์เหล่านี้เพิ่มเติมเมื่อมีการกล่าวถึงหรืออ้างอิงถึง สำหรับคำศัพท์ที่ควรทราบในเบื้องต้นมีดังนี้



รูปที่ 2-1 ไดอะแกรมแสดงความเกี่ยวข้องของส่วนประกอบต่างๆ ที่ต้องทราบในการใช้งานไลบรารี **microgear** สำหรับพัฒนาโปรแกรมให้แก่อุปกรณ์เพื่อติดต่อกับ NETPIE

1. อุปกรณ์ที่นำมาเชื่อมต่อกับ NETPIE จะถูกเรียกว่า เกียร์ (Gear) โดยในแต่ละแอปพลิเคชันอาจมีอุปกรณ์หรือเกียร์หลายตัว และยังสามารถสื่อสารกันเองภายในได้โดยรหัสประจำตัวของแอปพลิเคชัน (AppID) เดียวกัน โดย NETPIE เปิดให้ในแต่ละ AppID มีอุปกรณ์ได้สูงสุด 100 อุปกรณ์

2. ผู้ใช้งานแต่ละคนที่ทำการลงทะเบียนกับ NETPIE จะต้องสร้าง AppID ขึ้นมาเอง (สูงสุด 10 AppID)

3. แต่ละอุปกรณ์หรือแต่ละเกียร์จะมีรหัสกุญแจหรือคีย์ (key) เป็นของตัวเอง เพื่อใช้ระบุตัวตนของอุปกรณ์

4. แต่ละคีย์จะมีรหัสอิกซ์คลูฟฟ์เพื่อใช้ในการระบุตัวตนเรียกว่า 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 อันเป็นชิปเอนพาที่ผลิตขึ้นในจีน) สำหรับในหนังสือเล่มนี้ ฮาร์ดแวร์ 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 และเปลี่ยนชื่อโฟลเดอร์เป็น **microgear-esp8266-arduino** คัดลอกไปยัง **C:\Arduino1.6.5r5\libraries**

ฟังก์ชันหลักที่ควรทราบของ microgear เพื่อนำไปใช้ในการเขียนโปรแกรมติดต่อกันมีดังนี้

## 2.2.1 microgear.init

เป็นฟังก์ชันเตรียมการตั้งค่าหรืออินิเชียลเพื่อเริ่มต้นใช้งานไลบรารี microgear

### รูปแบบ

```
microgear.init(char* gearkey, char* gearsecret)
microgear.init(char* gearkey, char* gearsecret, char* gearalias)
microgear.init(char* gearkey, char* gearsecret, char* gearalias, char* scope)
```

### พารามิเตอร์

gearkey คือ ชื่อหรือคีย์สำหรับอ้างอิงตัวตนของเกียร์หรือคุปกรณ์

gearsecret คือ รหัสลับหรือ secret ของคีย์ (key) ใช้ประกอบในกระบวนการยืนยันตัวตน

gearalias คือ ชื่อของคุปกรณ์

scope คือ ขอบเขตการทำงานที่ต้องการ

### ตัวอย่างที่ 2-1

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

## 2.2.2 microgear.on()

เนื่องจากการทำงานของ microgear เป็นแบบ event driven จึงต้องตอบสนองต่อเหตุการณ์ต่างๆ ด้วยการเปลี่ยน callback function โดยฟังก์ชัน microgear.on() นี้เป็นการทำหน้าที่ต้องทำงานเมื่อถูกเรียก

### รูปแบบ

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

### พารามิเตอร์

event คือ ชื่อของเหตุการณ์

callback คือ ฟังก์ชันที่ต้องการให้ทำงานเมื่อเกิดเหตุการณ์ที่ระบุไว้ข้างบน

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* gearalias)
```

#### พารามิเตอร์

gearalias คือ ชื่ออุปกรณ์ที่กำหนด

#### ตัวอย่างที่ 2-6

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

กำหนดชื่อของอุปกรณ์ที่เชื่อมต่อกับ NETPIE สำหรับให้มีชื่อว่า 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()

เป็นฟังก์ชันที่ใช้ส่งข้อมูลแบบไม่เฉพาะเจาะจงผู้รับไปยังหัวข้อหรือ topic ที่กำหนด ซึ่งจะมีแต่ อุปกรณ์หรือเกียร์ที่ทำการบอกรับข้อมูลหรือข้อมูลในหัวข้อเดียวกันนี้เท่านั้นที่จะได้รับข้อมูล

### รูปแบบ

```
microgear.publish(char* topic, char* message)
microgear.publish(char* topic, char* message, bool retained)
```

### พารามิเตอร์

topic คือ หัวข้อหรือแหล่งข้อมูล

message คือ ข้อมูลหรือข้อมูลที่ต้องการส่ง

retained เป็นพารามิเตอร์ที่แจ้งให้เก็บรักษาข้อมูลไว้ หากกำหนดเป็น “1” ถ้าไม่ต้องการเก็บข้อมูล ต้อง กำหนดเป็น “0” หรือไม่ใส่ค่าต่อท้าย โดยการเก็บรักษาข้อมูลหรือ retained จะใช้ในกรณีค่า ด้วยวิธี REST API (จะได้กล่าวถึงในภายหลัง)

### ตัวอย่างที่ 2-9

```
microgear.publish("/sub2esp ","Hello Inex ")
กำหนดให้ส่งข้อมูล Hello Inex ไปยังสมาชิกหรืออุปกรณ์ที่บอกรับหัวข้อ sub2esp โดยไม่ต้องการเก็บ รักษาข้อมูลไว้
```

## 2.2.7 microgear.setName()

เป็นฟังก์ชันสมัครบอกรับข้อมูลจากหัวข้อที่สนใจ

### รูปแบบ

```
microgear.setName(char* gearname)
```

### พารามิเตอร์

gearname คือ ชื่อของหัวข้อที่ต้องการสมัครเข้ารับข้อมูลหรือเรียกอีกอย่างคือ topic

### ตัวอย่างที่ 2-10

```
microgear.setName("Chat2esp")
```

เป็นการสมัครบอกรับข้อมูลหรือข้อมูลจากหัวข้อ Chat2esp

## 2.2.8 microgear.chat()

เป็นฟังก์ชันส่งข้อมูลทำงานเหมือนการ publish นั่นคือ มีหัวข้อที่กำหนดและข้อความที่ต้องการส่ง

รูปแบบ

```
microgear.chat(char* targetgear, char* message)
```

พารามิเตอร์

targetgear คือ ชื่อหัวข้อที่ต้องการส่งหรือ topic

message คือ ข้อความที่ต้องการส่ง

ตัวอย่างที่ 2-11

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

กำหนดให้ส่งข้อความ Hello Inex ไปยังสมาชิกหรืออุปกรณ์ที่บอกรับหัวข้อ Chat2HTML

## 2.2.9 microgear.connect()

เป็นฟังก์ชันที่ใช้เชื่อมต่อกับ NETPIE โดยต้องระบุชื่อ AppID ที่อุปกรณ์ต้องการเชื่อมต่อ

รูปแบบ

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

พารามิเตอร์

appid คือ ชื่อ AppID ที่ต้องการเชื่อมต่อ

ตัวอย่างที่ 2-12

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

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

กำหนด AppID เป็น GroupInex จากนั้นอุปกรณ์จะเข้ามายังฟังก์ชัน microgear.connect(APPID)

## 2.2.10 microgear.connected()

เป็นฟังก์ชันตรวจสอบสถานะการเชื่อมต่อ

รูปแบบ

```
microgear.connected()
```

การคืนค่า

true - ในการนี้ที่เชื่อมต่อได้

false - ในการนี้ที่เชื่อมต่อไม่ได้

## 2.2.11 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 2-13

```
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 2-14

```
microgear.connect ("GroupInex");
กำหนดให้เชื่อมต่อกับ AppID ที่ชื่อ GroupInex
```

### 2.3.3 microgear.setname

เป็นฟังก์ชันที่ใช้ในการกำหนดสิทธิ์ในการเข้าถึงของชื่อของหัวข้อการติดต่อหรือ gearname ซึ่งสามารถกำหนดเป็นชื่อเรียกเมื่อมีการเรียกใช้ฟังก์ชัน chat() ดังนั้นฟังก์ชันนี้จึงมีการทำงานเหมือนกับการสมัครขอรับข้อมูลหรือ subscribe

#### รูปแบบ

```
void microgear.setname (gearname)
```

#### พารามิเตอร์

gearname คือชื่อหัวข้อการติดต่อของอุปกรณ์หรือเกียร์ ใช้รูปแบบข้อมูลเป็น string

#### ตัวอย่างที่ 2-15

```
microgear.setname ("Chat2HTML");
กำหนดชื่อหัวข้อที่ต้องการเข้าถึง ในตัวอย่างนี้คือ Chat2HTML
```

### 2.3.4 microgear.chat

เป็นฟังก์ชันส่งข้อความไปยังอุปกรณ์หรือเกียร์ที่ต้องการรับข้อความ ฟังก์ชันนี้มีการทำงานเหมือนกับการส่งข้อมูลหรือ publish

#### รูปแบบ

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

#### พารามิเตอร์

gearname คือ ชื่อหัวข้อการติดต่อของอุปกรณ์ที่ต้องการจะส่งข้อความไปถึง มีรูปแบบข้อมูลเป็น String

message คือ ข้อความที่ต้องการส่ง มีรูปแบบข้อมูลเป็น String

#### ตัวอย่างที่ 2-16

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

กำหนดให้ส่งข้อความ Hello Inex ไปยังหัวข้อชื่อ Chat2esp

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

### 2.3.5 microgear.resettoken

เป็นฟังก์ชันส่งคำสั่ง revoke token ไปยัง NETPIE เพื่อลบล้างการเชื่อมต่อของอุปกรณ์ออกจาก NETPIE และทำการลบโทเคน (token : สิ่งที่ใช้แสดงสิทธิ์ในการเชื่อมต่อ) ออกจากแคช (cache : หน่วยความจำสำหรับเก็บข้อมูลชั่วคราว) อันเป็นการรีเซ็ตการเชื่อมต่อ ส่งผลให้อุปกรณ์หรือเกียร์ต้องร้องขอการเชื่อมต่อเข้ามาใหม่

#### รูปแบบ

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

#### พารามิเตอร์

callback คือ ฟังก์ชันที่จะถูกเรียกให้ทำงาน เมื่อการรีเซ็ตโทเคนเสร็จสิ้น

#### ตัวอย่างที่ 2-17

เนื่องจาก resettoken() เป็นฟังก์ชันแบบอะซิงโครอนัส หากต้องการเชื่อมต่ออุปกรณ์เข้ามาใหม่ หลังจากได้กระทำฟังก์ชัน resettoken ต้องเปลี่ยนโค้ดในลักษณะนี้

```
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 2-18

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

### 2.4.2 เหตุการณ์ closed

เกิดขึ้นเมื่ออุปกรณ์หรือเกียร์ตัดการเชื่อมต่อกับแพล็ตฟอร์ม

#### ตัวอย่างที่ 2-19

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

### 2.4.3 เหตุการณ์ rejected

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

#### ตัวอย่างที่ 2-20

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

### 2.4.4 เหตุการณ์ error

เกิดขึ้นเมื่อมีความผิดพลาดเกิดขึ้นภายในอุปกรณ์หรือเกียร์ที่ทำการเชื่อมต่ออยู่

#### ตัวอย่างที่ 2-21

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

## 2.4.5 เหตุการณ์ message

เหตุการณ์นี้เกิดขึ้นเมื่อมีข้อความหรือ message เข้ามาพร้อมกับส่งผ่านข้อมูลที่เกี่ยวกับข้อความ หรือ message นั้นมาทางอะคิวเมนต์ของฟังก์ชันที่ถูกกำหนดให้ทำงาน

### ตัวอย่างที่ 2-22

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

## 2.4.6 เหตุการณ์ present

เหตุการณ์นี้เกิดขึ้นเมื่ออุปกรณ์หรือเกียร์ภายใต้รหัสแอปพลิเคชันหรือ AppID เดียวกัน ปรากฏตัวขึ้นมาหรือออนไลน์เพื่อเชื่อมต่อ กับ NETPIE

### ตัวอย่างที่ 2-23

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

## 2.4.7 เหตุการณ์ absent

เหตุการณ์นี้จะเกิดขึ้นเมื่ออุปกรณ์หรือเกียร์ภายใต้รหัสแอปพลิเคชันหรือ AppID เดียวกันหายไปจากการเชื่อมต่อกับ NETPIE หรือเกิดการอฟไล์น

### ตัวอย่างที่ 2-24

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

## 2.5 สรุปการใช้ฟังก์ชันรับ-ส่งข้อมูล

ฟังก์ชัน `microgear.setName()` ใช้กำหนดลิทธิ์การเข้าถึงหัวข้อการติดต่อของอุปกรณ์ หรือเกียร์หรือ `gearname` ใช้คู่กับฟังก์ชัน `microgear.chat()` เพื่อส่งข้อมูล

ฟังก์ชัน `microgear.subscribe()` ใช้กำหนดลิทธิ์การเข้าถึงของหัวข้อหรือ topic นั้นๆ ใช้คู่กับฟังก์ชัน `microgear.publish()` สำหรับส่งข้อมูล

อย่างไรก็ตาม หนึ่งอุปกรณ์สามารถใช้ฟังก์ชันทั้งสองอย่างพร้อมกันได้

## 2.6 การใช้งาน JavaScript เป็นต้น

เนื่องจาก JavaScript เป็นภาษาที่มีการทำงานบนคอมพิวเตอร์ของผู้ที่เปิดใช้งาน ไม่ใช่ทำงานบนเซิร์ฟเวอร์ จึงทำให้นำมาใช้จัดการในเรื่องการแสดงผล หรือกำหนดรูปแบบ ลูกเล่น ได้อย่างมาก หมาย การใช้งาน JavaScript ชุดคำสั่งจะถูกกำหนดไว้ระหว่างคำสั่ง `<script>` และปิดท้ายด้วยคำสั่ง `</script>`

### 2.6.1 ส่วนประกอบของโปรแกรมภาษา HTML หรือ HTML element

ประกอบด้วย

**แท็กเริ่ม + แอตทริบิวต์ + เนื้อหา + แท็กปิด**

#### ตัวอย่างที่ 2-25

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

#### คำอธิบาย

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

#### ตัวอย่างที่ 2-26

```
<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

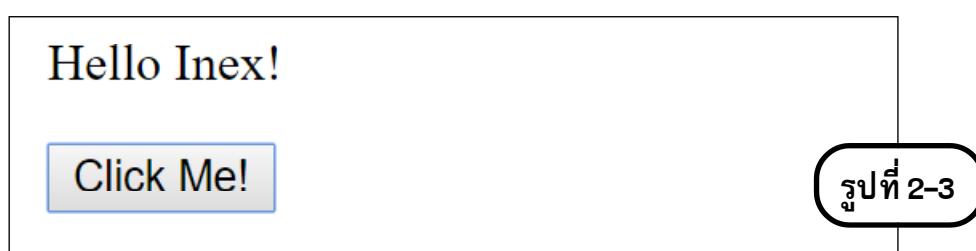
เมื่อเปิดหน้าเว็บเพจนี้ขึ้นมาจะมีข้อความ **JavaScript can change HTML content** แสดงขึ้นมา จะเกิดการเปลี่ยนแปลงข้อความเมื่อกดปุ่ม **Click Me !**

### ผลลัพธ์

(1) เมื่อเริ่มเปิดหน้าเว็บเพจ จะได้ผลตั้งรูปที่ 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()` ในการอ้างถึงอิเลิมентаใดๆ จะใช้ `id` ของแต่ละแอตทริบิวต์เป็นตัวกำหนด และใช้ `.innerHTML` เป็นการเปลี่ยนแปลงเนื้อหาของอิเลิมентаนั้นๆ

### ตัวอย่างที่ 2-27

```
<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

บุํนกดมีแอตทริบิวต์ที่ใช้กำหนดเหตุการณ์เมื่อมีการคลิกปุ่มที่ชื่อว่า `onclick` เมื่อกดการคลิกปุ่มนี้เรียกใช้ฟังก์ชัน `myFunction` ซึ่งภายในฟังก์ชันประกอบด้วยการเปลี่ยนแปลงข้อความและการตีบักเพื่อตรวจสอบการทำงาน ด้วยการใช้ฟังก์ชัน `console.log` เพื่อแสดงผลการทำงาน

### ผลลัพธ์

(1) เมื่อเริ่มเปิดหน้าเว็บเพจนี้ขึ้นมา ให้กด F12 แล้วเลือกที่หัวข้อ Console จะปรากฏหน้าเว็บดังรูปที่ 2-4



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



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

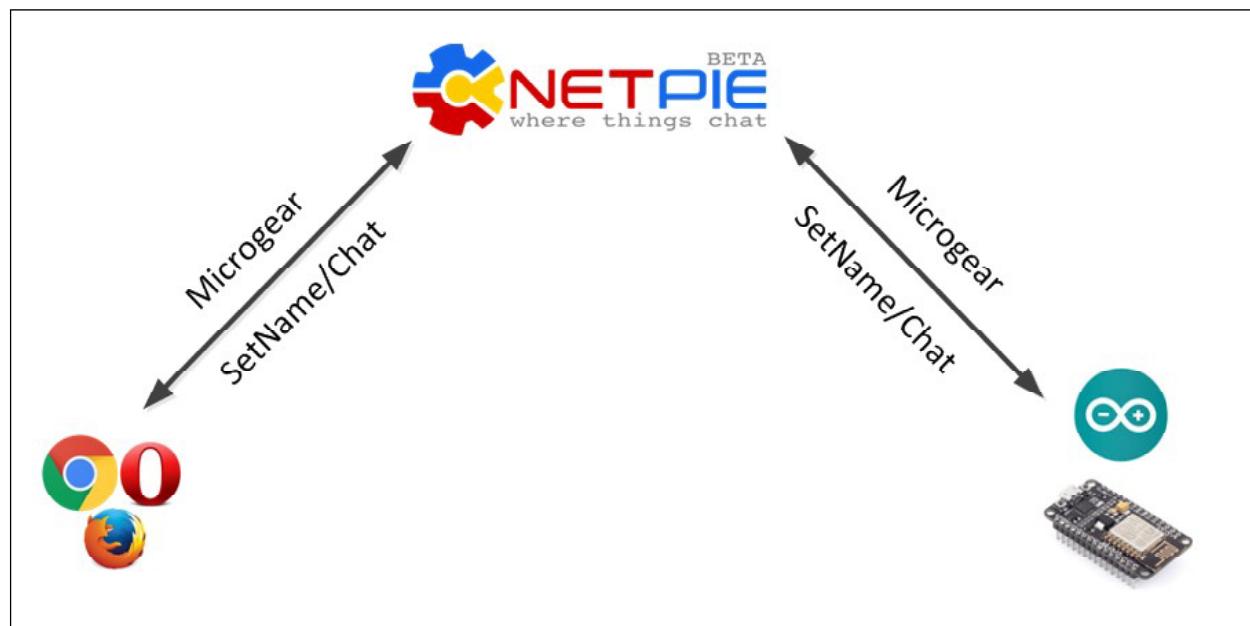
ในหัวข้อนี้นำเสนอตัวอย่างการติดต่อระหว่างอุปกรณ์ชาร์ดแวร์ซึ่งก็คือ NodeMCU-12E และคอมพิวเตอร์ (ผ่านเว็บбраузอร์) กับ NETPIE โดยมีไกด์โปรแกรมหาทำงานในการพร้อมแสดงดังรูปที่ 2-6

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

```

#include <AuthClient.h>
#include <MicroGear.h>
#include <MQTTClient.h>
#include <SHA1.h>
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <EEPROM.h>

const char* ssid      = <WIFI_SSID>;
const char* password = <WIFI_KEY>;
#define APPID      "GroupInex"
#define APPKEY     "Yn1yT7QqKyYFk6"
#define APPSECRET  "Ksn1hYJ2ELSNjsKb2iiJa6YIg"
#define ALIAS      "espChat"

WiFiClient client;
AuthClient *authclient;
int count=0;
int timer = 0;
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... ");
    microgear.setName("Chat2esp");
}

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: ");
}

```

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

```

Serial.println(WiFi.localIP());
//uncomment the line below if you want to reset token ->
// microgear.resetToken();
microgear.init(APPKEY, APPSECRET, ALIAS);
microgear.connect(APPID);
}
void loop()
{
    if (microgear.connected())
    {
        microgear.loop();
        count++;
        String str=String(count);
        microgear.chat("Chat2HTML", (char*)str.c_str());
        delay(250);
    }
    else
    {
        Serial.println("connection lost, reconnect...");
        if (timer >= 5000)
        {
            microgear.connect(APPID);
            timer = 0;
        }
        else timer += 100;
    }
}

```

### คำอธิบายโปรแกรม

โปรแกรมจะถูกกำหนดให้ได้ลิฟท์ข้ามชื่อหัวข้อ Chat2esp โดยใช้คำสั่ง microgear.setName("Chat2esp") นั่นแสดงว่า ทุกครั้งที่เกิดเหตุการณ์ จะใช้หัวข้อที่ชื่อว่า **Chat2esp** เป็นหัวข้อในการส่งข้อ นูดไปยัง NodeMCU-12E พังก์ชันที่จะทำงานเมื่อเกิดเหตุการณ์คือ onMsgHandler() โดยจะแสดงชื่อ topic และข้อความที่ส่งเข้ามา

ในพังก์ชัน loop() วนตัวตรวจสอบการเชื่อมต่อ กับ NETPIE ด้วยคำสั่ง if(microgear.connected) ถ้าเชื่อมต่อได้ พังก์ชัน microgear.loop() ค่อยตรวจสอบการทำงานทั้งหมดว่า มีเหตุการณ์หรืออีเวนต์ ใดเกิดขึ้นบ้าง จากนั้นตัวแปร count จะเพิ่มขึ้นหนึ่งค่า แล้วเปลี่ยนชนิดตัวแปร count ให้เป็นแบบ char ที่เป็น อะเรย์ก่อนส่งไปยัง topic ที่ชื่อว่า Chat2HTML ด้วยคำสั่ง microgear.chat ("Char2HTML", (char\*) str.c\_str()) ถ้ามีอุปกรณ์ใดที่อยู่รับข้อมูลที่ใช้หัวข้อหรือ topic นี้ ก็จะเห็นข้อมูลนี้ เช่นกัน

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

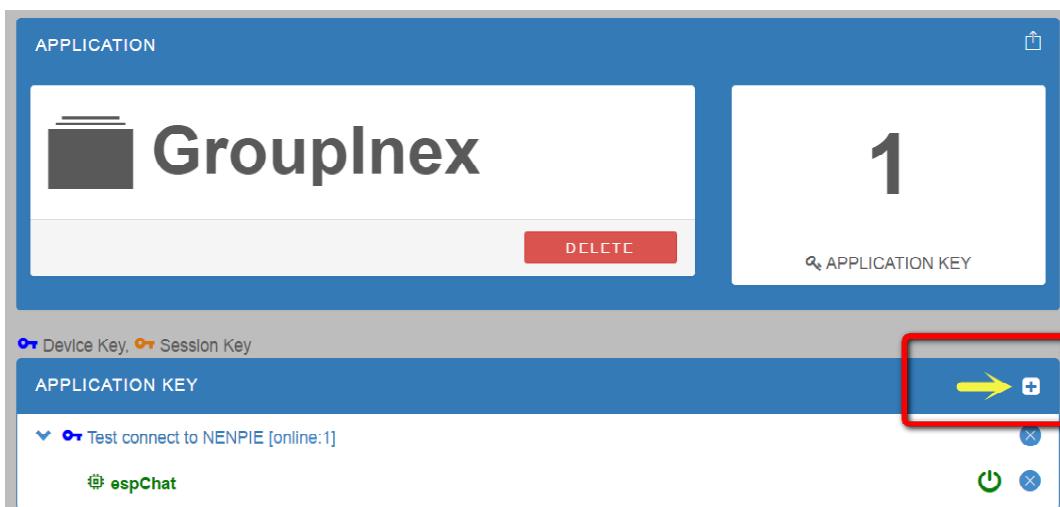
---

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

## 2.7.2 เตรียมการที่สำหรับ NETPIE เพื่อสร้างเว็บเพจสำหรับเป็นแดชบอร์ด

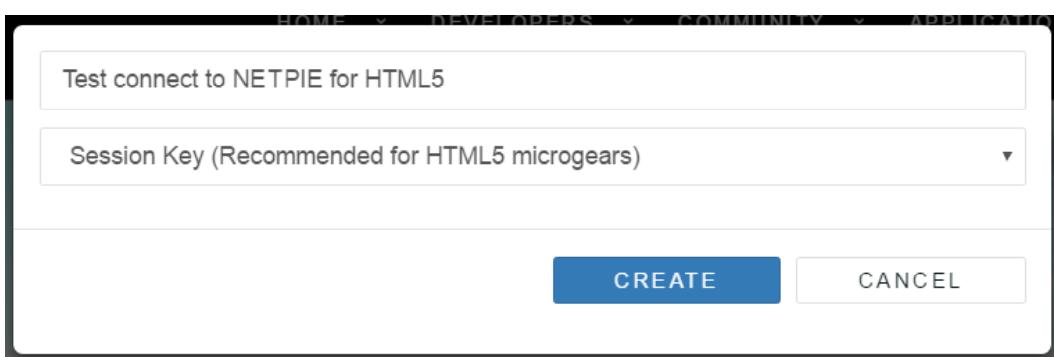
เว็บเพจที่ใช้แสดงผลการทำงานของโปรแกรม NETPIEChat.ino นี้พัฒนาขึ้นจากภาษา HTML ซึ่งจัดว่าเป็นหนึ่งในอุปกรณ์หรือเกียร์ของ NETPIE ดังนั้นจึงต้องมีการสร้างรหัสกุญแจหรือคีย์เพื่อผนวกเข้าไปใน AppID ของผู้พัฒนาด้วย NETPIE ได้ออกแบบรหัสกุญแจหรือคีย์ที่ใช้สำหรับการเปลี่ยนเว็บบราวเซอร์ให้เป็นหนึ่งในอุปกรณ์หรือเกียร์ 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 ที่หน้าต่าง 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

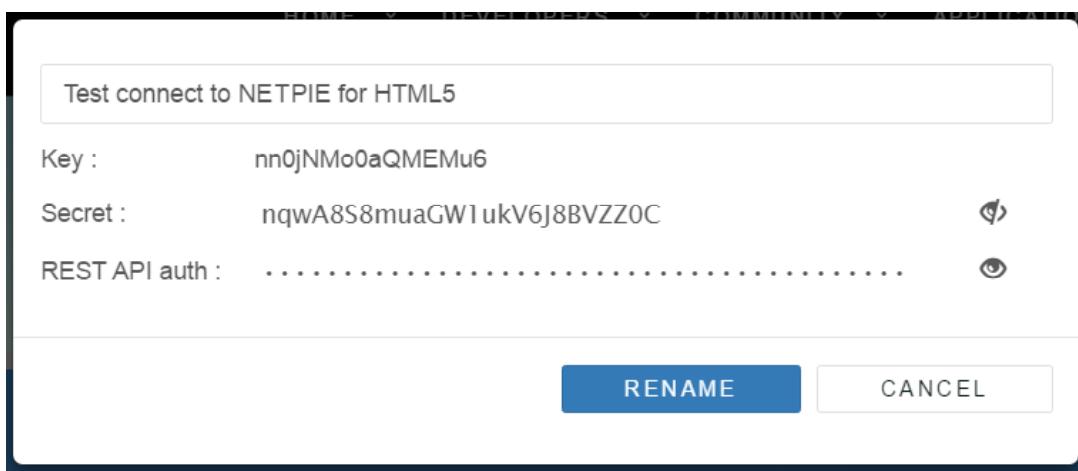


รูปที่ 2-9 คลิกเข้าไปดูรหัสต่าง ๆ ที่ต้องใช้ในการติดต่อกับคีย์ Test connect to NETPIE for HTML5

(2.7.2.4) จะปรากฏหน้าต่างแสดงรหัสต่างๆ ที่ต้องนำไปใช้กับการอ่านแบบเว็บเพจเพื่อใช้เป็นเดชบอร์ด ดังรูปที่ 2-10

สรุปรหัสทั้งหมดที่ต้องใช้มีดังนี้

```
APPID      = "GroupInex"
APPKEY    = "nn0jNM0aQMEMu6"
APPSECRET = "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 APPKEY = "nn0jNM0aQMEMu6";
const APPSECRET = "nqwA8S8muaGW1ukV6J8BVZZ0C";
```

```
<script src="https://NETPIE.io/microgear.js"></script>
<script>
    const APPID = "GroupInex";
    const APPKEY = "nn0jNM0aQMEMu6";
    const APPSECRET = "nqwA8S8muaGW1ukV6J8BVZZ0C";
    const ALIAS= "HTMLChat";

    var microgear = Microgear.create({
        gearkey: APPKEY,
        gearsecret: APPSECRET,
        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...";
        microgear.setname('Chat2HTML');
        console.log('connected with NETPIE...');

    });
</script>
```

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

```

microgear.on('present', function(event) {
  console.log(event);
});
microgear.on('absent', function(event) {
  console.log(event);
});
microgear.resettoken(function(err) {
  microgear.connect(APPID);
});

function myFunction() {
  var x = document.getElementById("myText").value;
  microgear.chat("Chat2esp",x);
  console.log('massage=' + 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 จะทำให้คอมพิวเตอร์ของผู้ชมนาเสมีค่าเป็นอุปกรณ์ อีกตัวหนึ่งของ AppID ชื่อ GroupInex เมื่ออุปกรณ์เขื่อมต่อไปยัง NETPIE ได้สำเร็จ อุปกรณ์ที่ชื่อ HTMLChat จะปรากฏให้เห็นในหน้า Key Management พารามิเตอร์ที่ใช้กำหนดคือ const ALIAS= "HTMLChat" อุปกรณ์ตัวนี้มีสิทธิ์เข้าถึง gearname ที่ชื่อว่า Chat2HTML ด้วยคำสั่ง microgear.setname('Chat2HTML') NodeMCU-12E ที่อัปโหลดโดยด้วยไฟล์ NETPIEChat.ino จะใช้ gearname นี้ส่งข้อมูลไปยัง NETPIE ดังนั้นเว็บเพจที่สร้างขึ้นนี้จึงทำงานที่แสดงผลข้อมูลที่ NodeMCU-12E ส่งเข้ามา

เมื่อกดเหตุการณ์ส่งข้อความขึ้น นั่นคือเกิดเหตุการณ์ message ฟังก์ชัน myFunction ซึ่งได้รับการกำหนดให้เป็น call back function จะทำงานทันที เพื่อนำข้อความนั้นมาแสดงบนเว็บเพจด้วยคำสั่ง document.getElementById("data").innerHTML = "Count=" + msg และตีบักเพื่อตรวจสอบ หัวข้อและข้อความที่เข้ามาด้วยฟังก์ชัน console.log('Topic=' + topic + ', message=' + msg)

การส่งข้อมูลจะใช้ฟังก์ชัน microgear.chat ("Chat2esp", x) โดย gearname ที่ใช้คือ Chat2esp ดังนั้นหากอุปกรณ์ตัวใดที่ได้รับการกำหนดสิทธิ์ให้เข้าถึง gearname นี้ ก็จะเห็นข้อความนี้เช่นกัน

---

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

### 2.7.3 ทดสอบการทำงาน

(2.7.3.1) แก้ไขโปรแกรมที่ 2-1 และ 2-2 โดยเปลี่ยน AppID , AppKey และ AppSecret เป็นของตัวผู้พัฒนาของ

(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 เชื่อมต่อกับ NETPIE ได้สำเร็จ ดังรูปที่ 2-11

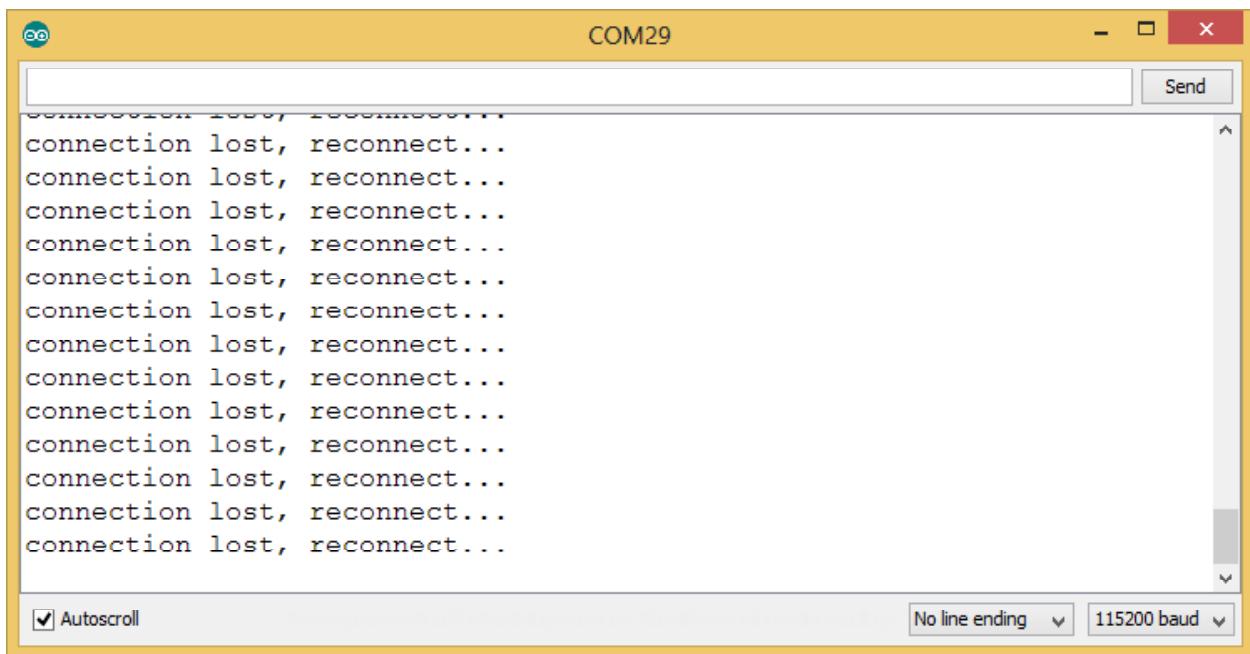
แต่ถ้าแสดงข้อความ *connection lost, reconnect...* ดังรูปที่ 2-12 แสดงว่า เชื่อมต่อไม่ได้ ให้ตรวจสอบการเชื่อมต่อทั้งระบบของผู้ใช้งาน และอาจเกิดจากความเร็วของอินเทอร์เน็ตที่ใช้งานต่อเกินไป

```

..... WiFi connected
IP address:
192.168.1.46
Time stamp : 1461061948
Token rom state == C;
Going to connect to MQTT broker
SpUuQyDsvzkkLeQg
SpUuQyDsvzkkLeQg%Yn1yT7QqyKyYFk6%1461061954
cZKPaZm/9A6525znFxBoBhsSYd8=
gb.netpie.io:1883
Connecting to : gb.netpie.io:1883
Connected to NETPIE...

```

รูปที่ 2-11 หน้าต่าง Serial Monitor ของ Arduino IDE แสดงสถานะการเชื่อมต่อ NodeMCU-12E กับ NETPIE สำเร็จ (ตำแหน่งพอร์ต COM เปลี่ยนแปลงไปตามผู้พัฒนา ในที่นี้เป็น 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

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

#### (2.7.3.5) จากนั้นตรวจสอบที่หน้า Key Management ของตัวเอง

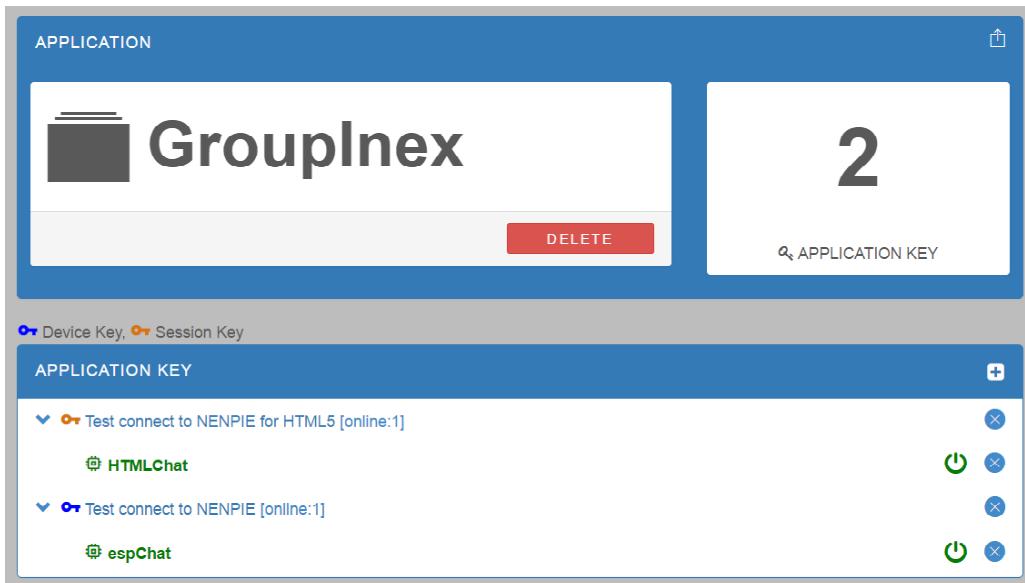
จะปรากฏชื่ออุปกรณ์ที่ได้กำหนดไว้ดังรูปที่ 2-15



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

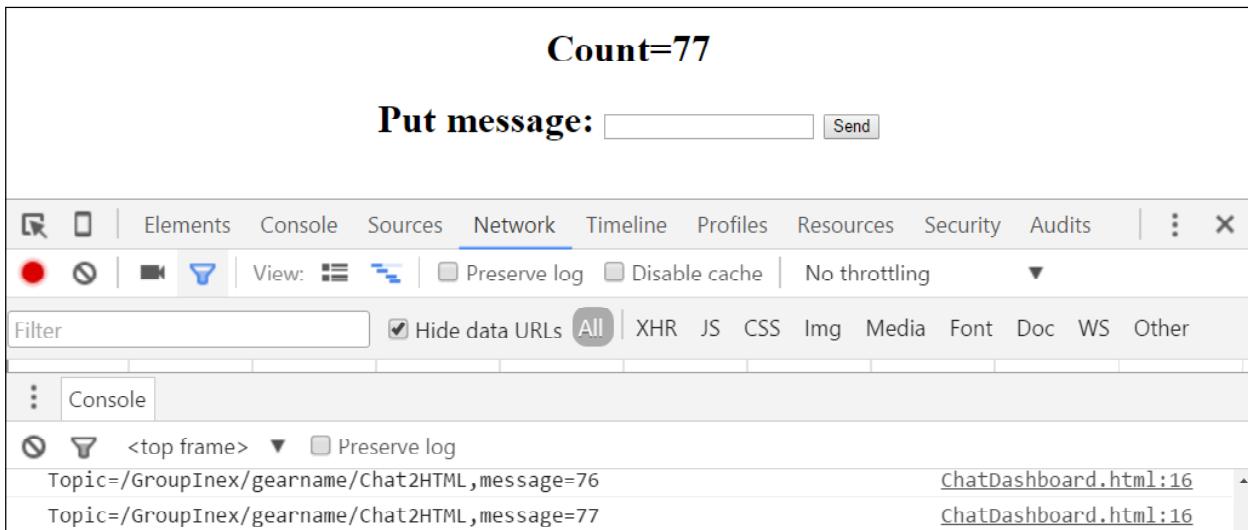
(2.7.3.6) ถ้ามีการใช้ gearname ชื่อ Chat2HTML ในการส่งข้อมูล

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 ด้วย

การทำงานของ REST จะเน้นไปที่การบริหารข้อมูลในรูปแบบของทรัพยากร (resource) เป็นหลัก

### 3.1.1 ข้อดีของ REST

1. ดำเนินการตามปรัชญา Open Web ทำให้เกิดโอกาสในการพัฒนาต่ออยอดได้
2. ง่ายต่อการนำไปใช้งานและการดูแลรักษา
3. แยกการทำงานอย่างชัดเจนระหว่าง ไอคลอินต์ (เครื่องลูก - ผู้ใช้งาน) กับเซิร์ฟเวอร์
4. การสื่อสารข้อมูลไม่ถูกกำหนดให้มีเพียงเอกสารลักษณะเดียว จึงยืดหยุ่น และดัดแปลงได้
5. ไอคลอินต์สามารถเก็บรักษาข้อมูลไว้ได้ เพื่อป้องกันการรับภัยจากการส่งข้อมูลซ้ำๆ หลายรอบจากแหล่งต่างๆ
6. ให้ผลลัพธ์ของข้อมูล ได้หลายรูปแบบ ทั้ง JSON และ XML จึงทำให้นำไปใช้งานกับโปรแกรมประยุกต์ที่พัฒนาด้วยโปรแกรมภาษาต่างๆ ได้ รวมถึง莎ร์ดแวร์ด้วย

### 3.1.2 ข้อเสียของ REST

1. ต้องทำงานที่ระดับสูงสุดของโปรโตคอล HTTP
  2. ยากที่จะบังคับให้มีขั้นตอนการอนุมัติในระบบ เพื่อช่วยรักษาความปลอดภัยให้แก่ข้อมูล
- ทั้งนี้ก็เนื่องมาจากการออกแบบ REST ที่ต้องการให้เกิดการเชื่อมต่อกันได้อย่างสะดวก และไม่มีเงื่อนไข เพื่อให้รองรับกับแพลตฟอร์มได้หลากหลายทั้ง โปรแกรมประยุกต์ที่พัฒนามาจากภาษาคอมพิวเตอร์ที่ต่างกัน และ莎ร์ดแวร์ที่มีความหลากหลาย ทั้งสมาร์ตโฟน แท็บเล็ต อุปกรณ์ IoT และคอมพิวเตอร์

### 3.1.3 จะใช้ REST เมื่อใด ?

REST จะมีประโยชน์อย่างมากหากระบบที่ประกอบด้วย ไอคลอินต์และเซิร์ฟเวอร์ต้องทำงานผ่านเว็บไซต์ หรือผ่านสภาพแวดล้อมแบบเว็บ (web environment) และตัว ไอคลอินต์หรือผู้ใช้งานไม่มีความจำเป็นต้องรับข้อมูลตลอดเวลา ซึ่งจะสอดคล้องกับการพัฒนาอุปกรณ์ IoT เพราะในแนวคิดของ IoT อุปกรณ์จะต้องคุยกันเอง สื่อสารข้อมูลกันเองได้ โดยอาศัยคลาวด์เซิร์ฟเวอร์เป็นตัวกลาง ผู้ใช้งานอาจเข้ามายอนิเตอร์หรือตรวจสอบข้อมูลได้ หรือเข้ามาสั่งการได้เป็นครั้งคราว

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

ตัวอย่างการนำ REST API ไปใช้งานที่เห็นเด่นชัดคือ ในสื่อสังคมออนไลน์ (social media), เครือข่ายสังคมออนไลน์ (social network), ระบบสนทนาผ่านเว็บ (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 เพื่อมองกับทุกอุปกรณ์ก่อนหน้านี้ที่จะนำมาเชื่อมต่อกับ NETPIE วิธีการสร้างรหัสทั้งหมดจะมีขั้นตอนเหมือนกับในบทที่ 2 เพียงเปลี่ยนชนิดของคีย์เป็น Device เพื่อมองกับการทดสอบในบทที่ 1 อย่างไรก็ตาม ผู้พัฒนาสามารถใช้ AppKey และ AppSecret ที่ได้จากการทดสอบในบทที่ 1 ในการใช้งานกับ REST API ได้ ซึ่งก็คือ

```
AppID = GroupIndex  
AppKey= Yn1yT7QqyKyYFk6  
AppSecret= Ksn1hYJ2ELSNjsKb2iiJa6YIg
```

## 3.3 การใช้งานเบื้องต้น

### 3.3.1 การยืนยันตัวตน

NETPIE REST API ช่วยให้นักพัฒนาสามารถส่งหรือเผยแพร่ข้อมูล (publish), บอกรับข้อมูล (subscribe) ผ่านทาง REST API ได้ โดยให้บริการหรือมีอินด์พอยต์อยู่ที่ <https://api.netpie.io> ดังนั้น หากต้องการดำเนินการเรื่องใดกับข้อมูล จะต้องทำการยืนยันสิทธิ์และตัวตนในการเข้าถึงก่อนใช้งานทุกครั้ง ซึ่งตอนนี้มี 2 วิธี

#### 3.3.1.1 ส่งผ่าน HTTP เอดเดอร์แบบ Basic Authentication

เป็นการยืนยันตัวตนแบบพื้นฐานด้วยพารามิเตอร์ 2 ตัว ดังนี้

```
Username : AppKEY  
Password : AppSECRET
```

#### ตัวอย่างที่ 3-1

การใช้ Basic Authentication ด้วย cURL

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

### 3.3.1.2 ส่งผ่านทาง URL พารามิเตอร์

มีรูปแบบดังนี้

?auth=AppKEY : AppSECRET

#### ตัวอย่างที่ 3-2

การใช้ URL พารามิเตอร์ด้วย cURL

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

### 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 จนกว่าจะมีการอ่านออกไป ข้อมูลหรือข้อมูลที่ถูกอ่านแล้วจะหายไปจาก postbox ทันที เหมาะที่จะใช้เป็นเครื่องมือสื่อสารกับ microgear หรืออุปกรณ์ที่ไม่สามารถออนไลน์หรือเชื่อมต่อกับเครือข่ายอินเทอร์เน็ตได้ตลอดเวลา 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 (เฉพาะค่าล่าสุดเทียบค่าเดิม)

#### Body

เป็นข้อความที่จะส่ง หากต้องการลบค่าที่ retain ไว้ให้ส่งแบบ retain และใช้ body เป็น string เป็นล่า

#### ตัวอย่างที่ 3-3

บน NETPIE มี AppID ซึ่ง GroupInex ต้องการใช้ REST API ในการส่งข้อความแบบ retain ว่า “ON” ไปยัง microgear ที่บอกรับ (subscribe) หัวข้อ “GroupInex/gearname/plug” เผยแพร่คำสั่งด้วย 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 AppKey ของ AppID ที่ซื้อ GroupInex

Ksn1hYJ2ELSNjsKb2iiJa6YIg คือ AppSecret ของ AppID ที่ซื้อ 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 โดยไคลเอนต์จะได้รับเฉพาะข้อความล่าสุดที่มี retain หรือเก็บไว้ล่าสุดก่อนหน้านี้

#### ตัวอย่างที่ 3-4

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

Yn1yT7QqyKyYFk6 คือ AppKey ของ AppID ที่ซื้อ GroupInex

Ksn1hYJ2ELSNjsKb2iiJa6YIg คือ AppSecret ของ AppID ที่ซื้อ GroupInex

#### ผลลัพธ์แบบ JSON

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

การอ่านข้อมูลความยังทำได้อีกวิธีหนึ่งคือ ใช้เว็บбраузอร์ 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 AppKey ของ AppID ที่ใช้ GroupInex

Ksn1hYJ2ELSNjsKb2iiJa6YIg คือ AppSecret ของ AppID ที่ใช้ 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 ที่ชื่อ 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 AppKey ของ AppID ที่ใช้ GroupInex

Ksn1hYJ2ELSNjsKb2iiJa6YIg คือ AppSecret ของ AppID ที่ใช้ 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) มากพอสมควร นั่นเป็นความจำเป็นของนักพัฒนาอุปกรณ์ IoT ที่ต้องมีความรู้พอสมควรเกี่ยวกับการพัฒนาเว็บแอปพลิกาชัน และยังมีต้องมาทำงานกับคลาวเตอร์ฟิเวอร์อย่าง NETPIE แล้ว ผู้พัฒนาจึงต้องมีความรู้ด้านการเขียน 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

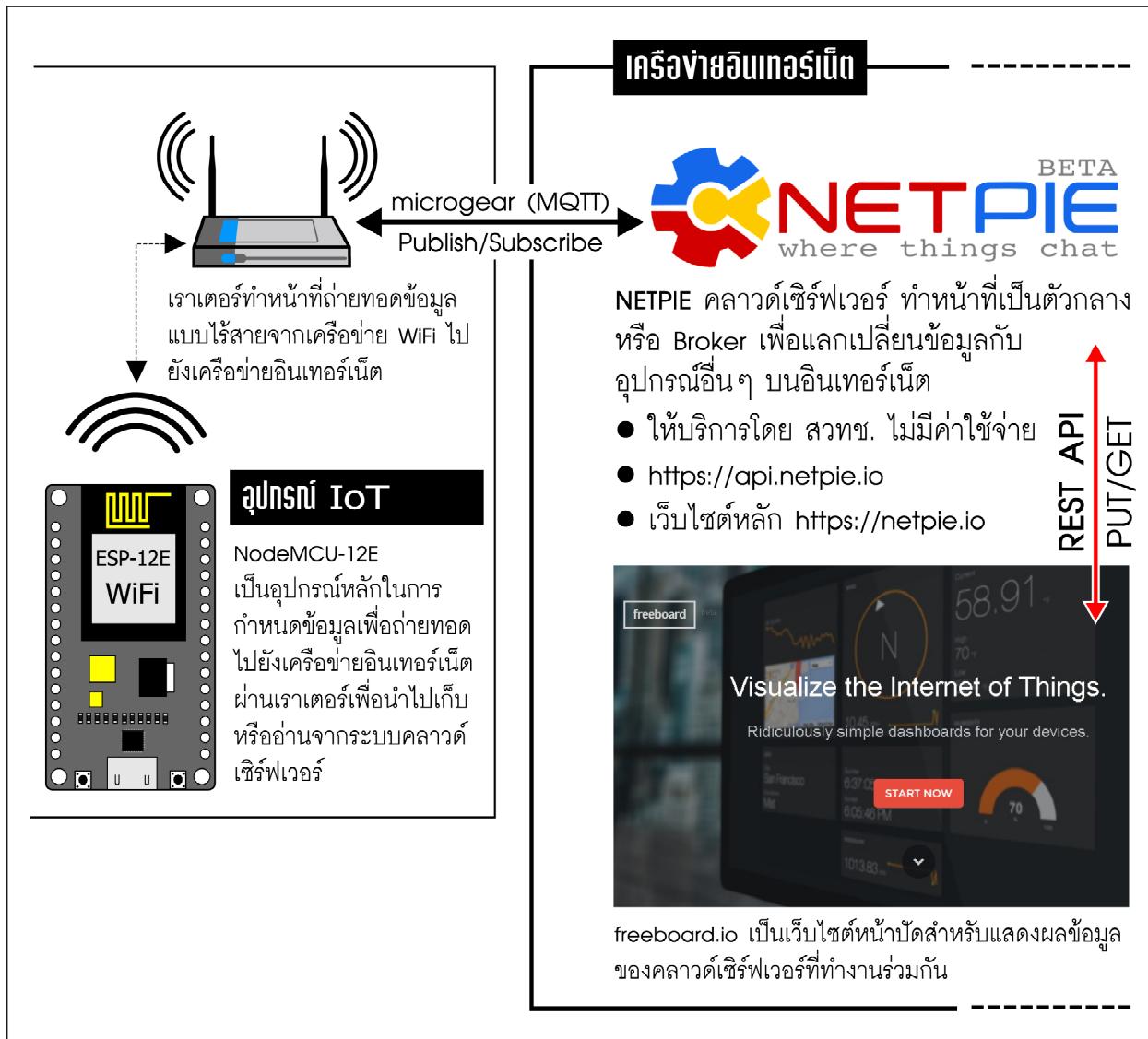
### 4.1 ภาพรวมการเชื่อมต่อ

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 ในการล้อกรับ

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

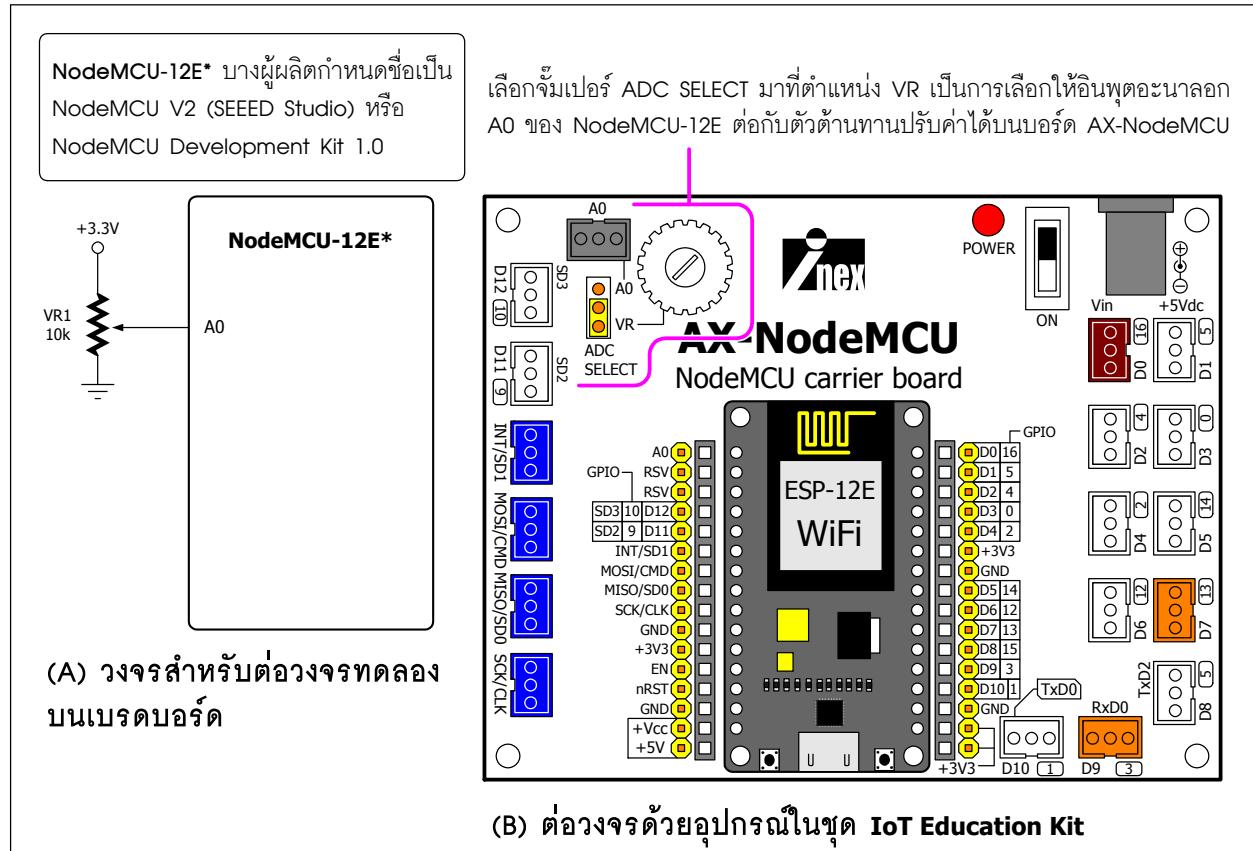
ตัวอย่างที่นำเสนอนี้คือ กำหนดให้ NodeMCU-12E ส่งค่าของสัญญาณอะนาล็อกที่อ่านได้จากอินพุต A0 ไปยัง NETPIE โดยใช้ Topic ที่ชื่อว่า /valAnalog ด้วยคำสั่ง publish("/valAnalog", -{Analog},1) โดยตัวเลข 1 หมายความว่า ข้อมูลที่ส่งขึ้นไปจะยังคงเก็บรักษาไว้ เพื่อให้กระบวนการของ REST API อ่านข้อมูล โดยการส่งแบบนี้จะส่งได้ 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 <AuthClient.h>
#include <MicroGear.h>
#include <MQTTClient.h>
#include <SHA1.h>
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
#include <EEPROM.h>

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

#define APPID      "GroupInex"
#define APPKEY     "Yn1yT7QqyKyYFk6"
#define APPSECRET  "Ksn1hYJ2ELSNjsKb2iiJa6YIg"
#define ALIAS      "esp"

int timer=0;
WiFiClient client;
AuthClient *authclient;

unsigned long previousMillis = 0;
const long interval = 1000;
MicroGear microgear(client);

void onMsghandler(char *topic, uint8_t* msg, unsigned int msglen)
{
    Serial.print("Incoming message -> ");
    Serial.print(topic);
    Serial.print(" : ");
    char strState[msglen];
    for (int i = 0; i < msglen; i++)
    {
        strState[i] = (char)msg[i];
        Serial.print((char)msg[i]);
    }
    Serial.println();
}
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)
    }
}

```

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

```

    {
        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(APPKEY,APPSECRET,ALIAS);
    microgear.connect(APPID);
}
void loop()
{
    if (microgear.connected())
    {
        microgear.loop();
        unsigned long currentMillis = millis();
        if(currentMillis - previousMillis >= interval)
        {           // save the last time
            previousMillis = currentMillis;
            String asciiAn=String(analogRead(A0));
            char bufAn[asciiAn.length()+1];
            asciiAn.toCharArray(bufAn,asciiAn.length());
            microgear.publish("/valAnalog",bufAn,1);
        }
    }
    else
    {
        Serial.println("connection lost, reconnect..."); 
        microgear.connect(APPID);
        delay(timer);
        timer+=100;
    }
}
}

```

### คำอธิบายโปรแกรมเพิ่มเติม

จากโปรแกรม อุปกรณ์จะบอกรับข้อมูลหรือ subscribe โดยใช้คำสั่ง microgear.subscribe("/valAnalog") มีหัวข้อการติดต่อหรือ Topic ที่ชื่อว่า /valAnalog ซึ่งเป็นหัวข้อเดียวกันกับการเผยแพร่ข้อมูลหรือ publish และการ publish ในครั้งนี้ต้องเก็บรักษาข้อมูลไว้ด้วย โดยใช้คำสั่ง microgear.publish("/valAnalog",bufAn,1) แต่ข้อจำกัดในกรณีนี้คือ ส่งข้อมูลมาไปยัง NETPIE ได้ 10 ข้อมูลต่อ 1 วินาที ดังนั้นจึงใช้ฟังก์ชัน millis() มาช่วยในการสร้างเพื่อนໃนกำหนดเวลาส่งข้อมูล

ในการ publish แบบเก็บรักษาข้อมูลไว้ ทุกครั้งที่อุปกรณ์มีการ subscribe โดยใช้ Topic เดียวกัน อุปกรณ์ตัวนั้นจะได้รับข้อมูลล่าสุดที่ส่งไปยัง NETPIE ก่อน 1 ครั้งเสมอ

---

**โปรแกรมที่ 4-1 ไฟล์ NETPIEPubAnalog.ino โปรแกรมสำหรับ NodeMCU-12E เพื่ออ่านค่าอินพุตอะนาล็อกส่งเข้าไปยัง NETPIE และแสดงผลด้วยแดชบอร์ดที่ freeboard.io (มีต่อ)**

```

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 AppKey ของ AppID ที่ใช้ GroupInex

`Ksn1hYJ2ELSNjsKb2iiJa6YIg` คือ AppSecret ของ AppID ที่ใช้ GroupInex

#### ผลลัพธ์ในแบบ JSON

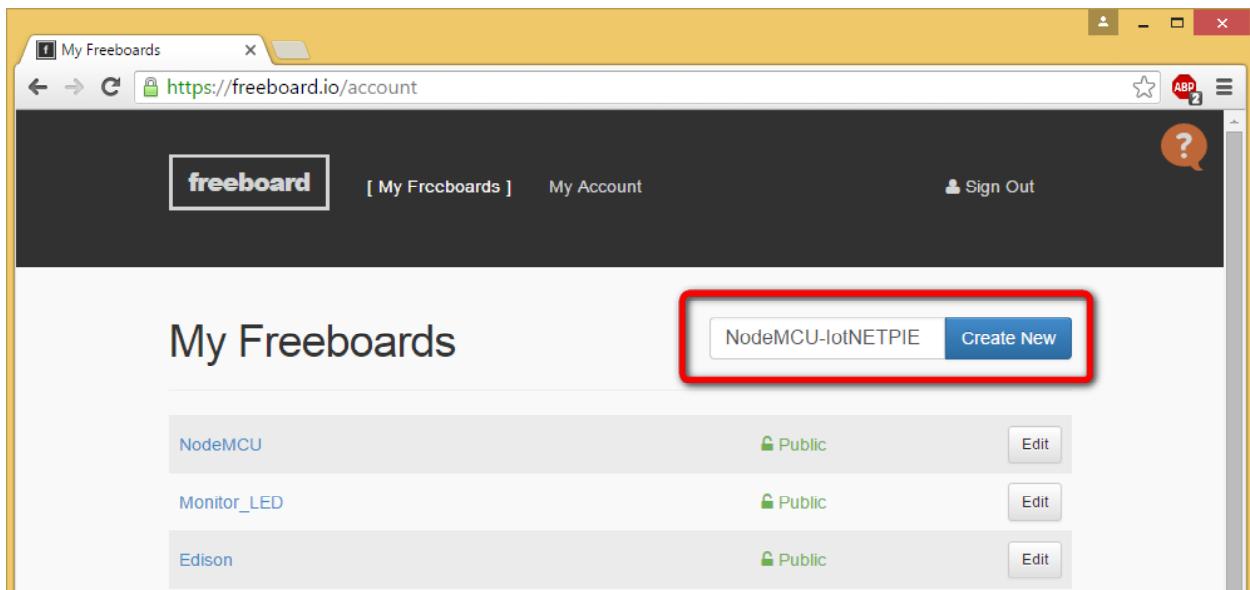
`[{"topic":"/GroupInex/valAnalog","payload":"476","qos":0,"retain":true}]`

#### 4.2.2 การออกแบบ Dashboard บน Freeboard.io

มีขั้นตอนดังนี้

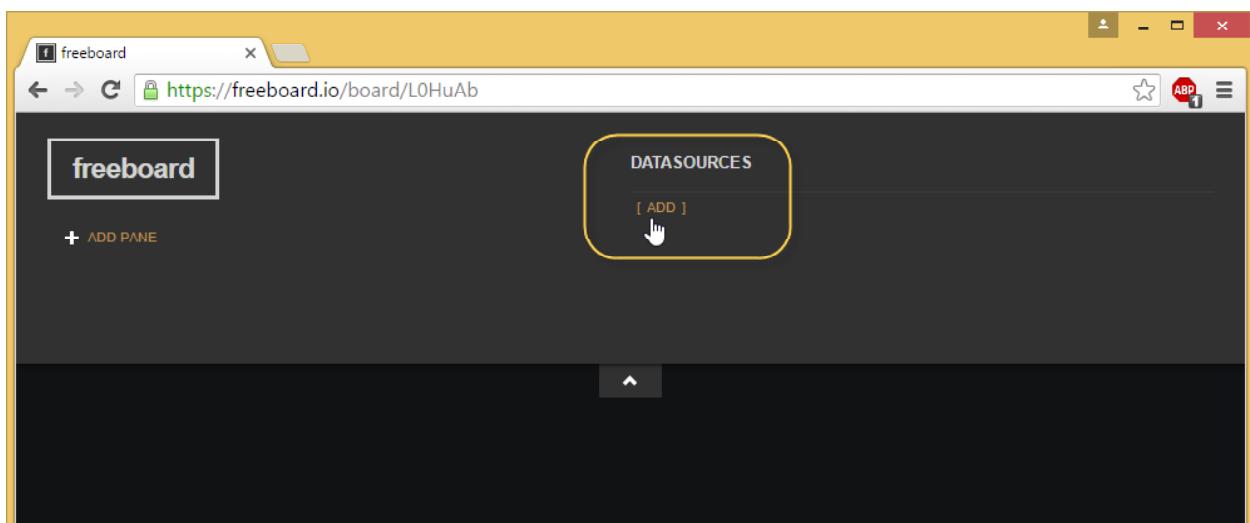
(4.2.2.1) ไปที่เว็บไซต์ freeboard.io ทำการล็อกอินเพื่อเข้าใช้งาน หรือถ้ายังไม่มีบัญชี ให้ทำการลงทะเบียนให้เรียบร้อยก่อน

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



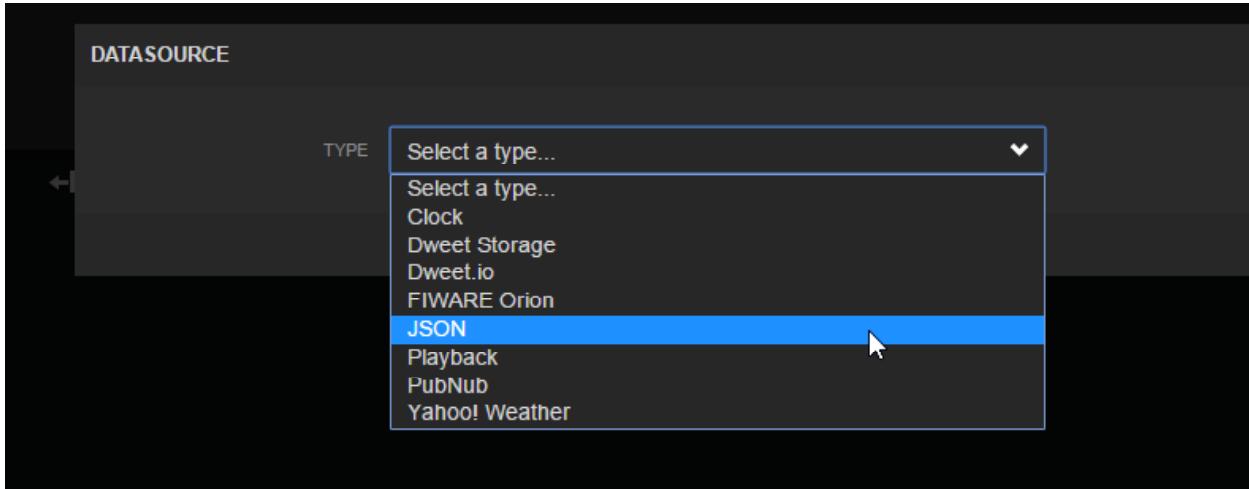
รูปที่ 4-4 เข้าสู่เว็บไซต์ freeboard.io ทำการล็อกอินและสร้างแดชบอร์ดใหม่ในชื่อ NodeMCU-12E-IotNETPIE

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



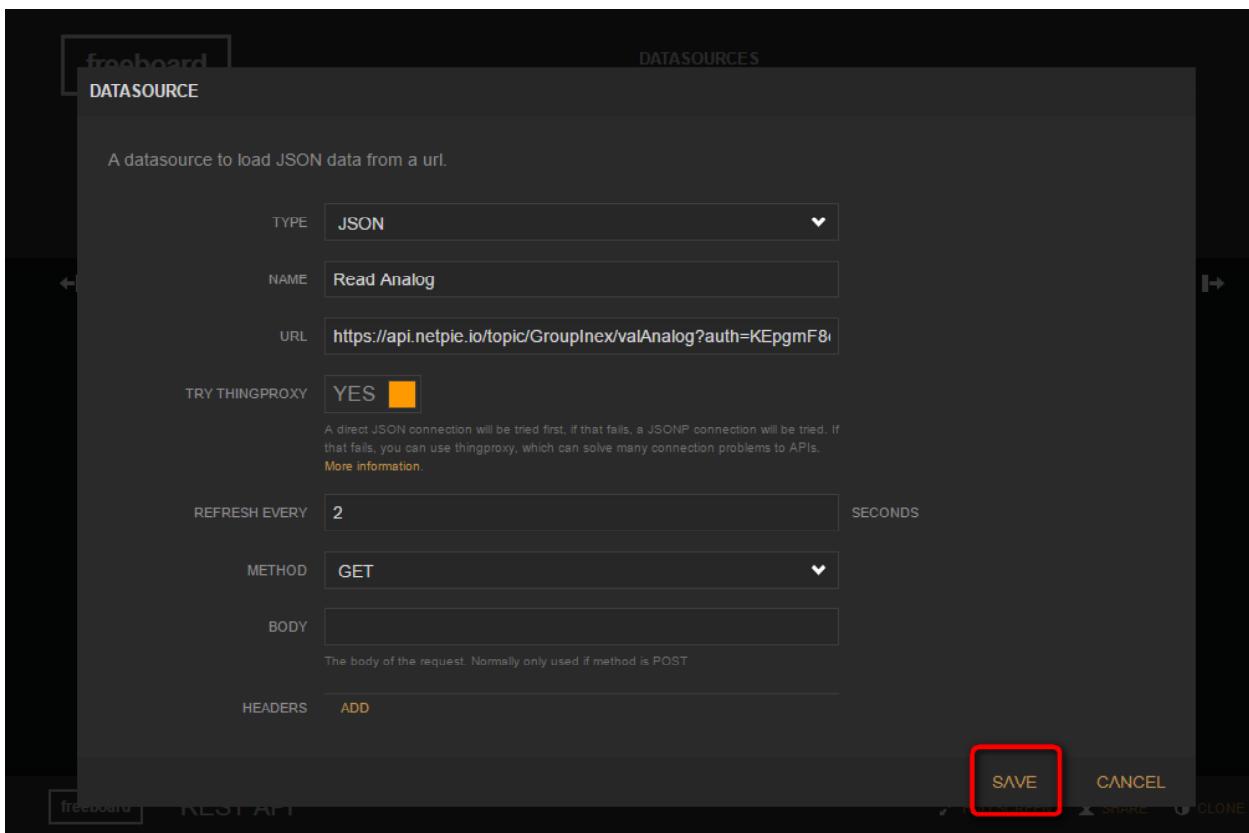
รูปที่ 4-5 เพิ่มแหล่งข้อมูลหรือ DATA SOURCES

(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

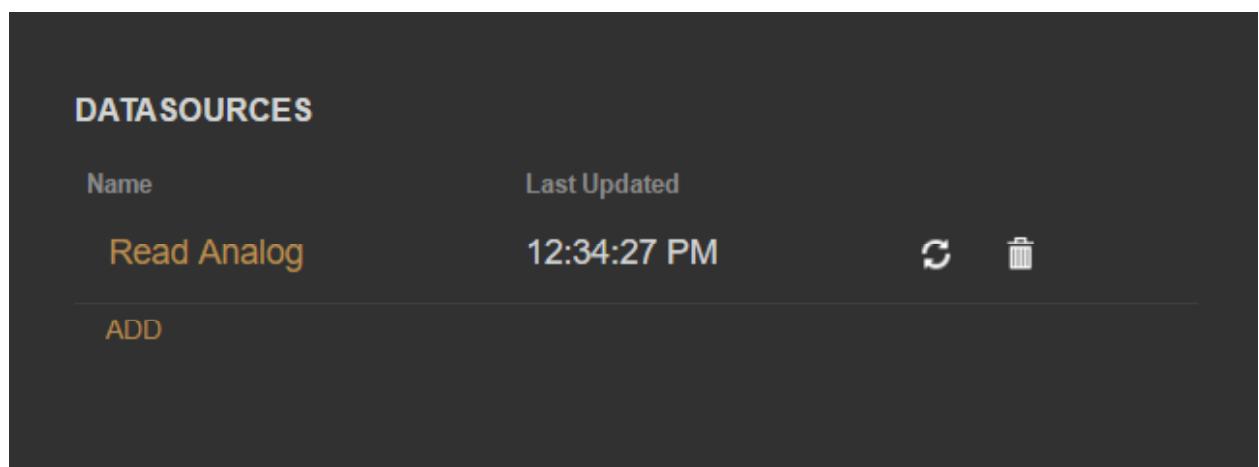
ที่ชื่อง NAME - ตั้งชื่อได้ตามต้องการ

ที่ชื่อง URL - ใช้ตัวແໜ່ງເດືອນກັບກາຣທົດສອບດ້ວຍເວັບບວກເຫຼວໜ້ອທີ່ຜ່ານມາ ນັ້ນຄືອ

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

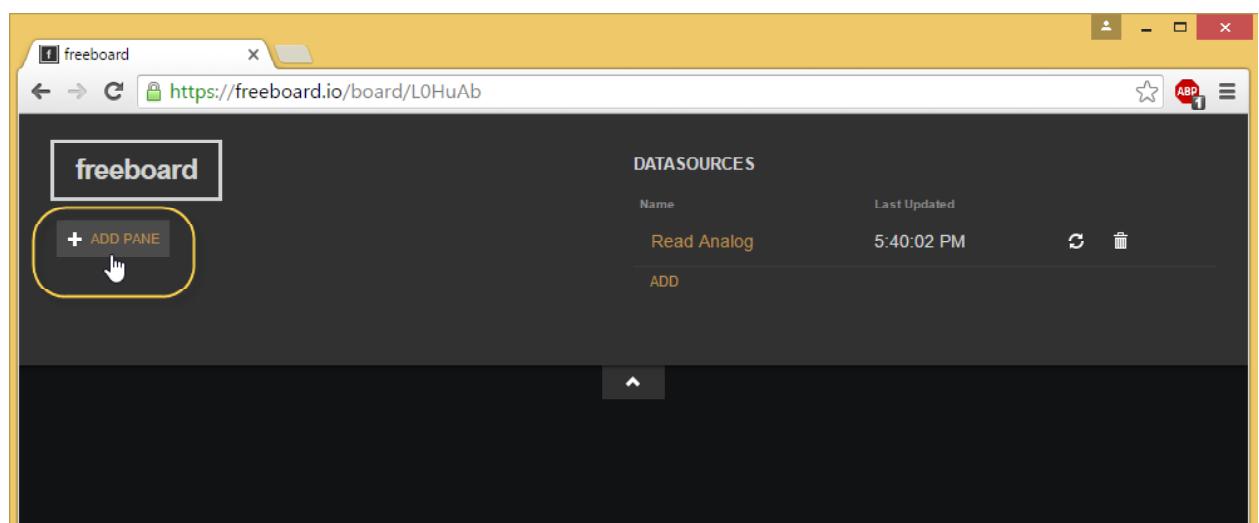
ที่ชื่อง REFRESH EVERY - ໃຊ້ກຳທັນເວລາທີ່ຕ້ອງກາຮ່ານຂໍ້ອຄວາມ ມ່ວຍເປັນວິນາທີ່  
ຈາກນັ້ນຄືກີ່ມີ **SAVE** ເພື່ອບັນທຶກ

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

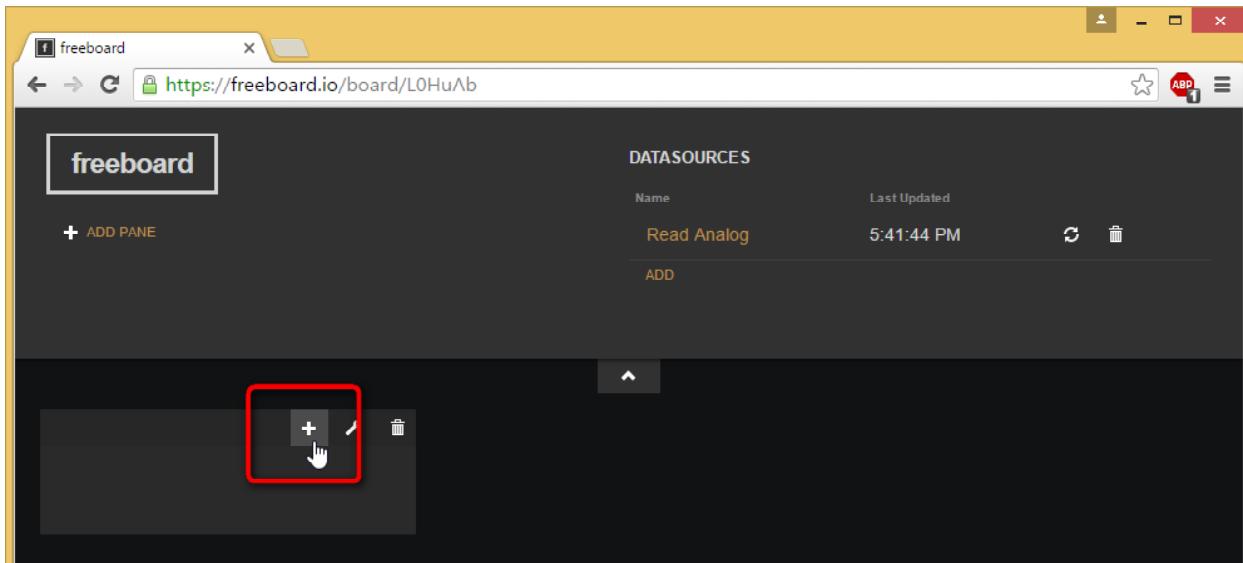


ຮູບທີ່ 4-8 ແສດ **DATASOURCES** ໄໝມທີ່ເກີດຂຶ້ນໃນໜີ້ຂອງ **Read Analog**

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



ຮູບທີ່ 4-9 ແສດກາເພີ່ມສ່ວນແສດງຜລ



รูปที่ 4-10 เพิ่มอุปกรณ์แสดงผล

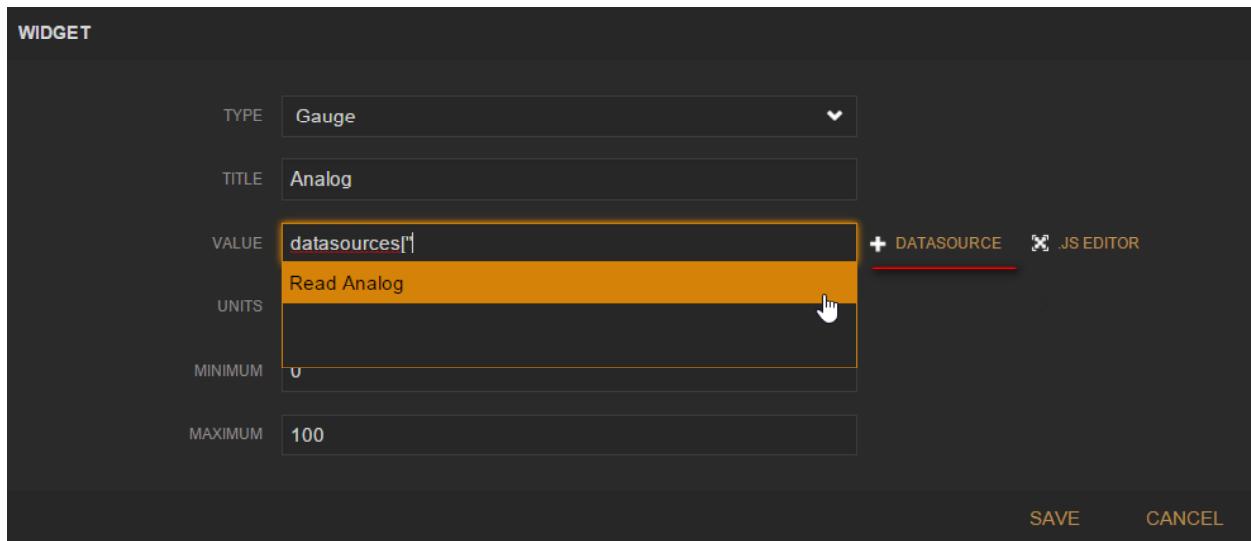
(4.2.2.8) เพิ่มอุปกรณ์แสดงผลหรือ Widget โดยคลิกที่เครื่องหมาย + ตามรูปที่ 4-10

(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 'Widget' configuration dialog. The top screenshot shows a dropdown menu labeled 'TYPE' 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 'TITLE' set to 'Analog', which is highlighted with a red box. Other fields include 'VALUE', 'UNITS', 'MINIMUM' (set to 0), and 'MAXIMUM' (set to 100). There are also buttons for '+ DATA SOURCE' and 'JS EDITOR'.

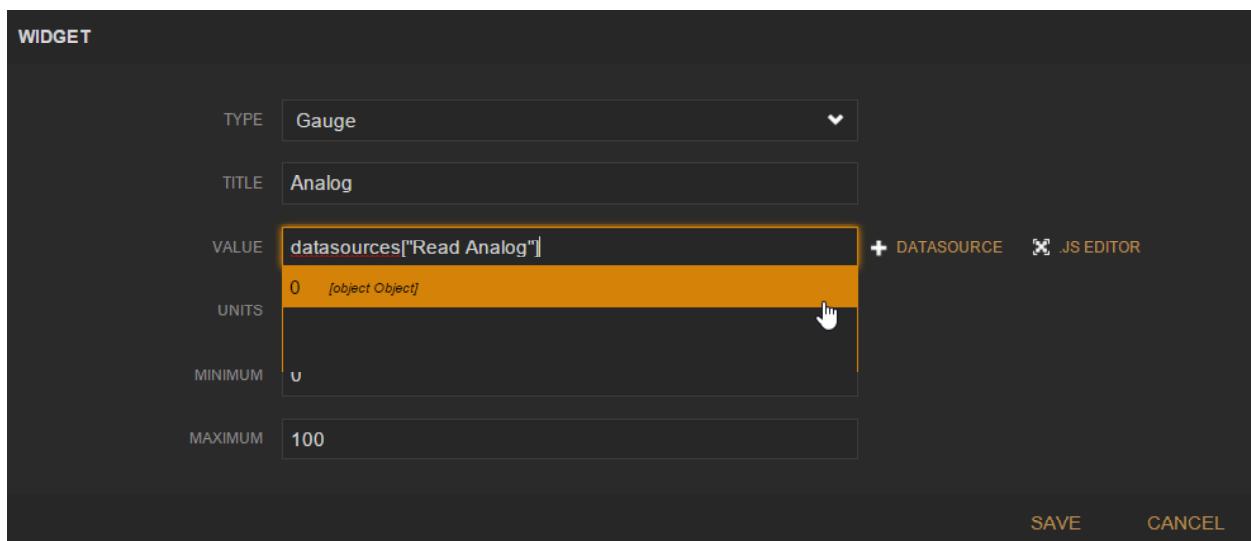
รูปที่ 4-11 เลือกวิดเก็ตหรืออุปกรณ์แสดงผลเป็น Gauge และกำหนดค่า

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



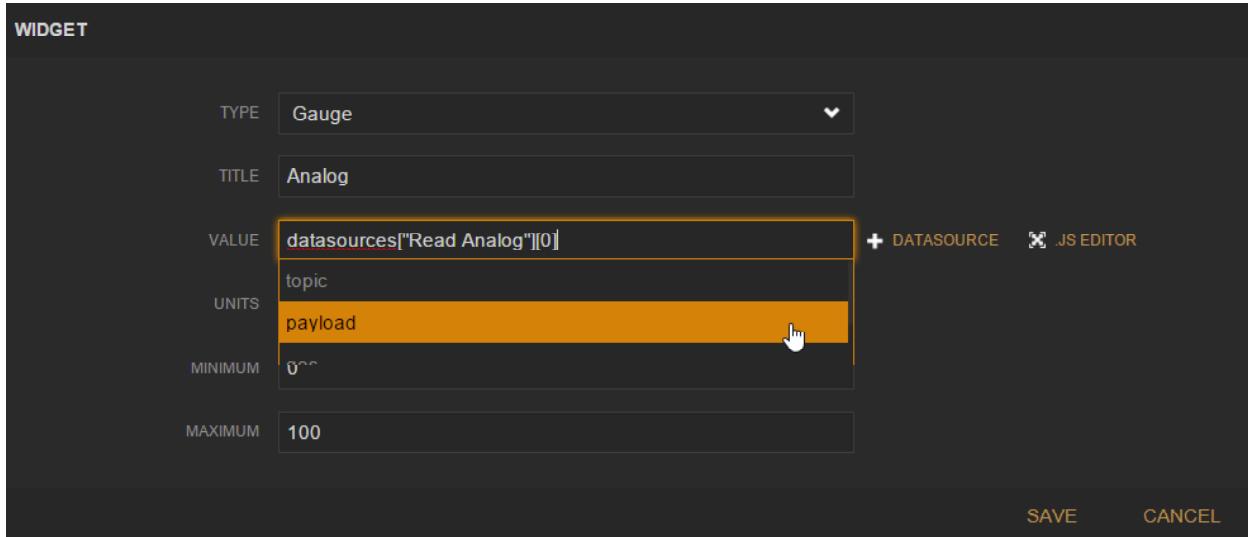
รูปที่ 4-12 เลือกแหล่งกำเนิดข้อมูลสำหรับช่อง VALUE ของตัวแสดงผลแบบ Gauge

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



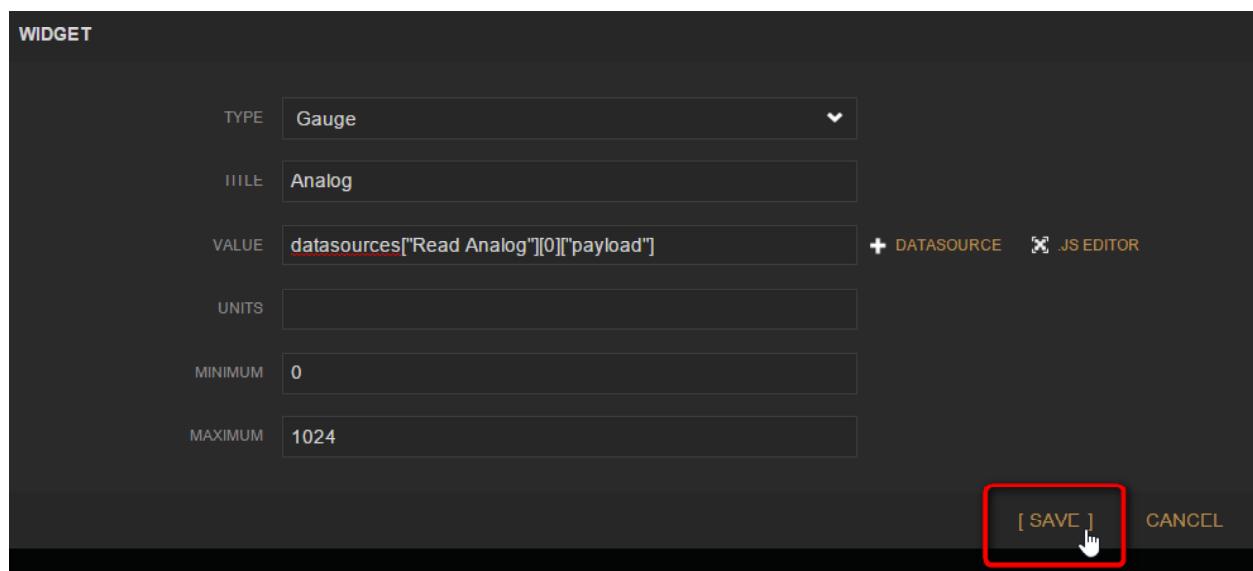
รูปที่ 4-13 เลือกรูปแบบข้อมูลที่เป็นแบบอาร์เรย์

(4.2.2.12) เมื่อกลิกแล้ว จะปรากฏข้อมูลทั้งหมดดังรูปที่ 4-14 ให้เลือกรายการ payload เพราะเป็นตัวระบุค่าอะนาล็อก



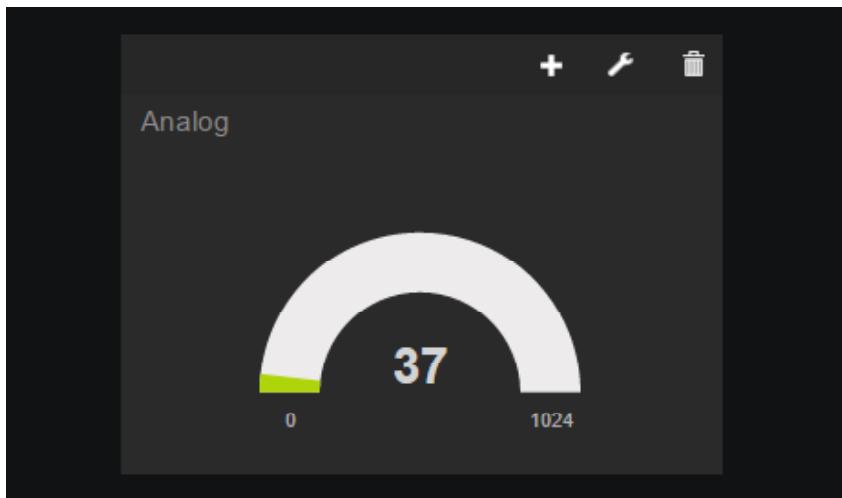
รูปที่ 4-14 เลือกพารามิเตอร์ของช่อง VALUE เพิ่มเติม

(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 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 มาในบทนี้จะเพิ่มปุ่มสำหรับกดเพื่อควบคุมสถานะทางลอดจิกของขาพอร์ตติดจิตอลของ NodeMCU-12E เป็นการใช้งานแดชบอร์ดบนเว็บ freeboard.io ในอีกรูปแบบหนึ่ง

### 5.1 เตรียมการ

ก่อนที่จะถึงขั้นตอนในการเขียนโปรแกรมไม่ว่าจะเป็นบน NodeMCU-12E หรือบนเว็บ freeboard.io ถึงที่ต้องเตรียมมีดังนี้

#### 5.1.1 รหัสคีย์ทั้งหมดที่ใช้เชื่อมต่อกับ NETPIE

```
AppID = GroupIndex
AppKey= Yn1yT7QqyKyYFk6
AppSecret= Ksn1hYJ2ELSNjsKb2iiJa6YIg
```

#### 5.1.2 หัวข้อของการติดต่อหรือ Topic

5.1.2.1 ใช้ Topic ชื่อ **/Send\_stLED** ในการส่งข้อความโดยใช้ REST API จาก freeboard ดังนี้ที่ NodeMCU-12E จะต้องทำการบอกรับหรือ subscribe หัวข้อการติดต่อ

5.1.2.2 ใช้ Topic ชื่อ **/Feedback\_stLED** เพื่อส่งให้ NodeMCU-12E ส่งค่าสถานะกลับมาที่ 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 ในการส่งข้อมูลความเปลี่ยนแปลง 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 AppKey ของ AppID ที่ใช้ GroupInex

Ksn1hYJ2ELSNjsKb2iiJa6YIg คือ AppSecret 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 AppKey ของ AppID ที่ใช้ GroupInex

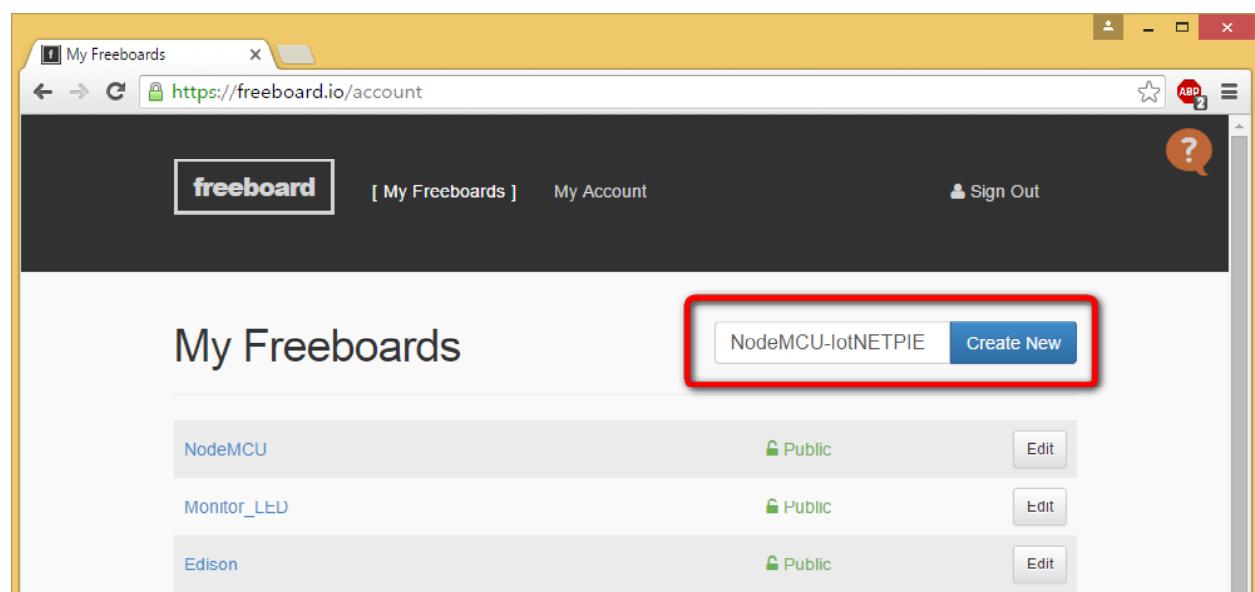
Ksn1hYJ2ELSNjsKb2iiJa6YIg คือ AppSecret ของ AppID ที่ใช้ GroupInex

## 5.2 สร้างแดชบอร์ดสำหรับควบคุม LED

### 5.2.1 การเตรียมการสำหรับ freeboard.io

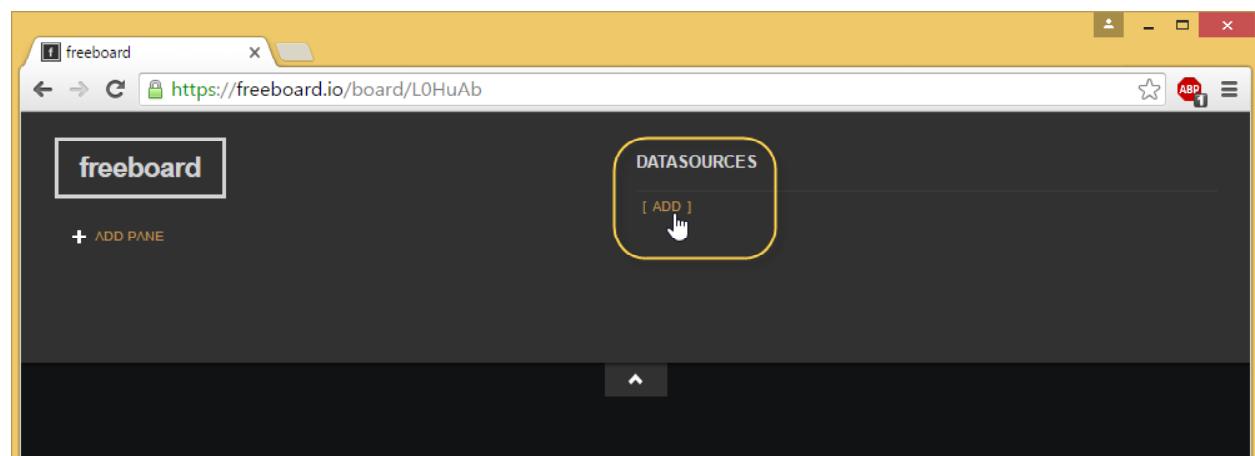
(5.2.1.1) ไปที่เว็บไซต์ freeboard.io ทำการล็อกอินเพื่อเข้าใช้งาน หรือถ้ายังไม่มีบัญชี ให้ทำการลงทะเบียนก่อน

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



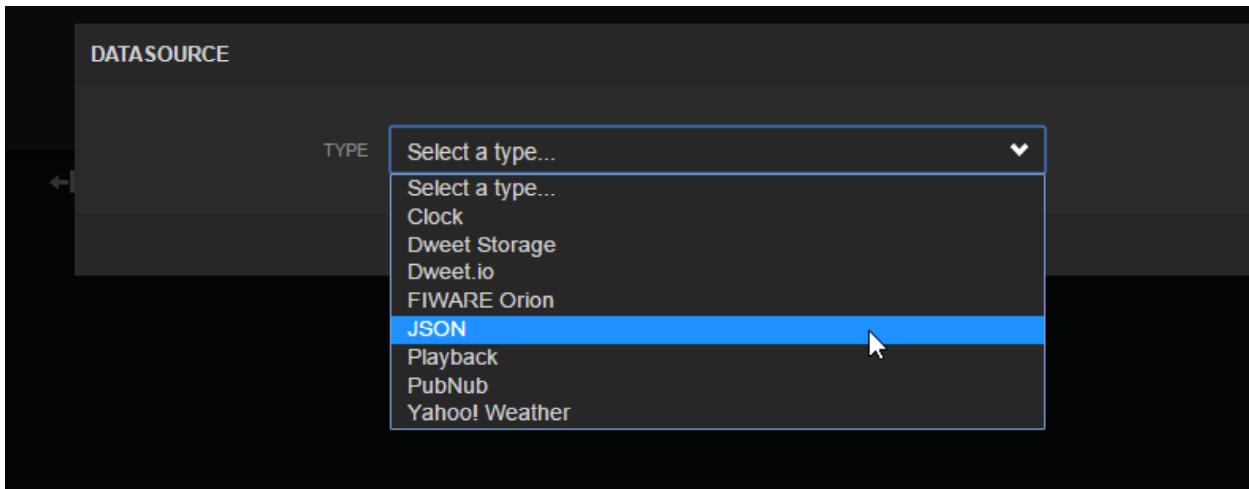
รูปที่ 5-1 แสดงหน้าเว็บ freeboard.io เมื่อเริ่มต้นสร้างแดชบอร์ดใหม่

(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

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

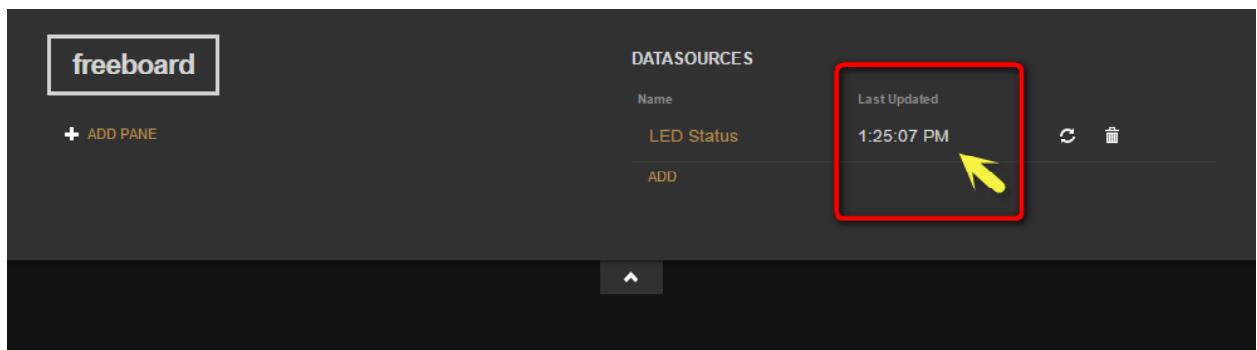
`Yn1yT7QqyKyYFk6` คือ AppKey ของ AppID ที่ชื่อ GroupInex

`Ksn1hYJ2ELSNjsKb2iiJa6YIg` คือ AppSecret ของ AppID ที่ชื่อ GroupInex

ที่ชื่อของ REFRESH EVERY - ใช้กำหนดเวลาที่ต้องการอ่านข้อมูลความหน่วงเป็นวินาที

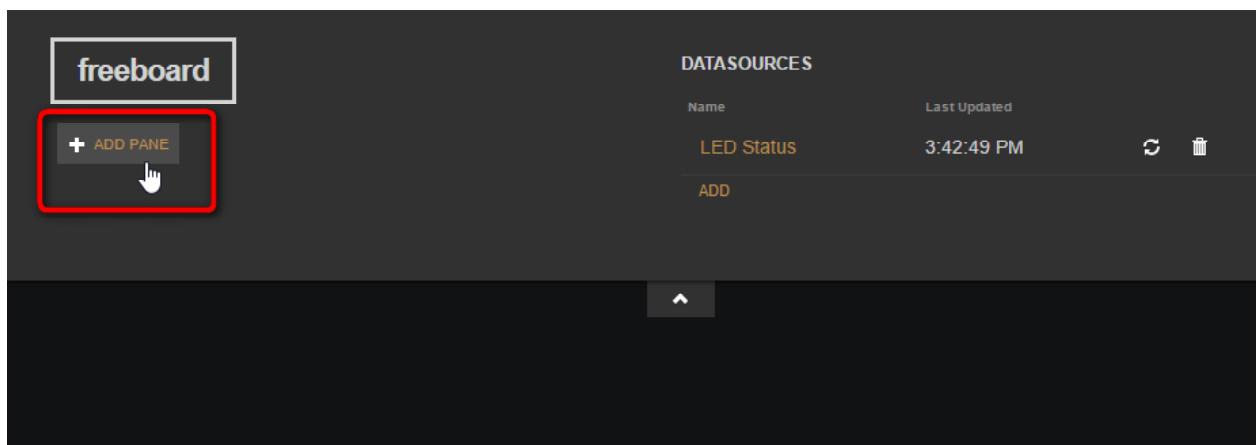
จากนั้นคลิกที่ปุ่ม SAVE เพื่อบันทึก

เมื่อบันทึกแล้ว จะมีชื่อ DATASOURCE ที่กำหนดไว้ และแสดงเวลาล่าสุดในการอ่านข้อมูลดังรูปที่ 5-5



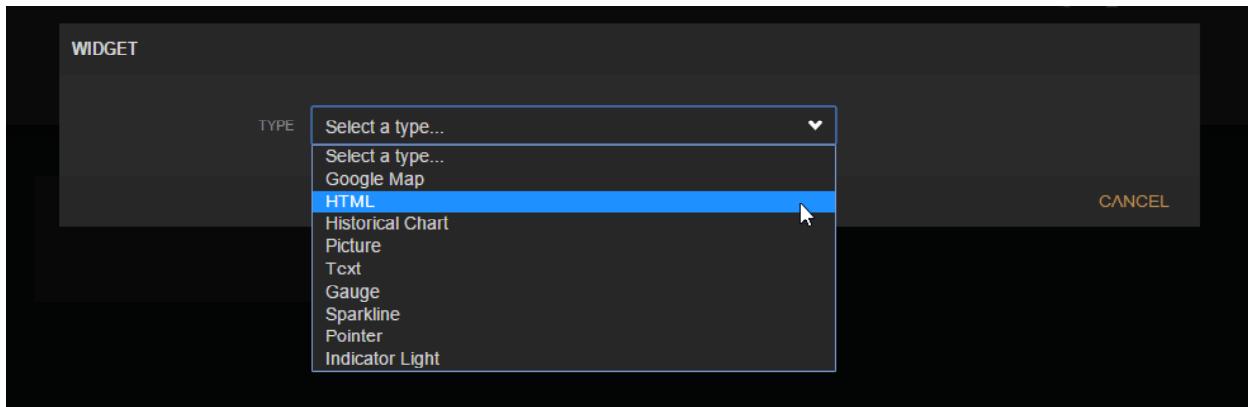
รูปที่ 5-5 แสดง DATASOURCES ใหม่ที่เกิดขึ้นในชื่อ LED Status

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



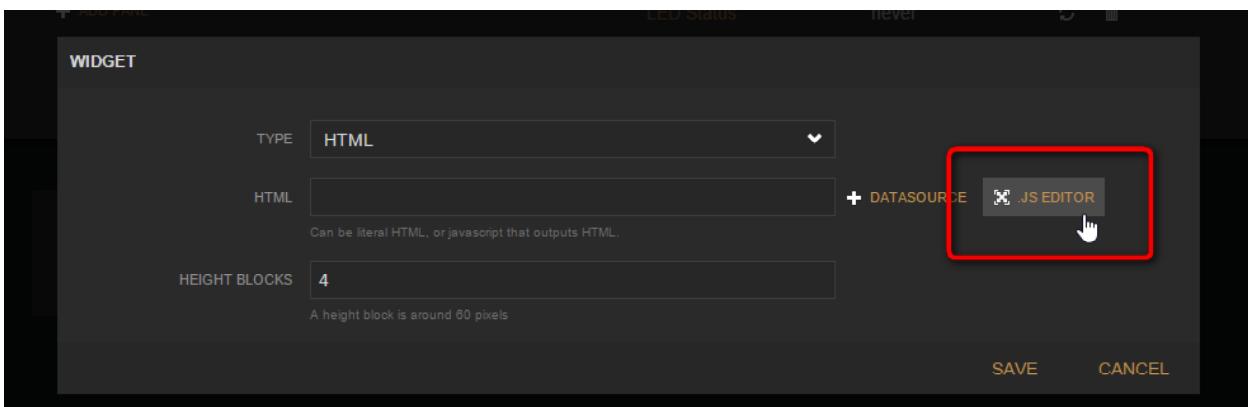
รูปที่ 5-6 คลิกปุ่ม +ADD PANE เพื่อเพิ่มส่วนแสดงผลให้แก่แดชบอร์ด

(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 แสดงการเลือกเปิดหน้าต่าง Java Script เพื่อเขียนโปรแกรมสั่นสำหรับสร้างปุ่มกด

(5.2.1.9) เมื่อเลือกเข้ามาอยู่ 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.

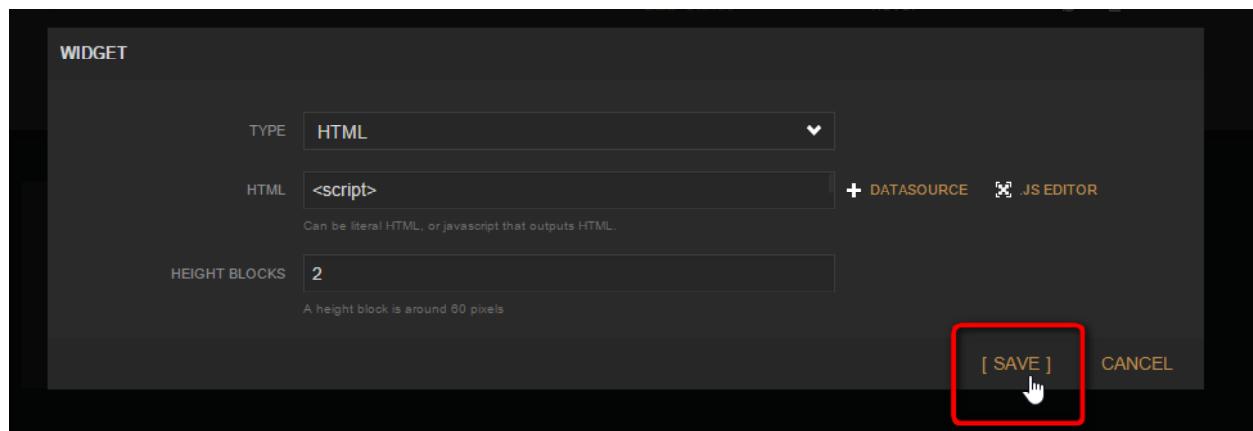
1 // Example: Convert temp from C to F and truncate to 2 decimal places.
2 // return (datasources["MyDatasource"].sensor.tempINF * 1.8 + 32).toFixed(2);
```

รูปที่ 5-9 แสดงหน้าต่าง Java Script เอ迪เตอร์เพื่อเขียนสคริปต์สำหรับสร้างปุ่มกด

(5.2.1.10) เพิ่มสคริปต์ภาษาจาวาดังนี้ โดยต้องเปลี่ยน AppKey, AppSecret, AppID เป็นของผู้พัฒนาเองตามที่เตรียมไว้

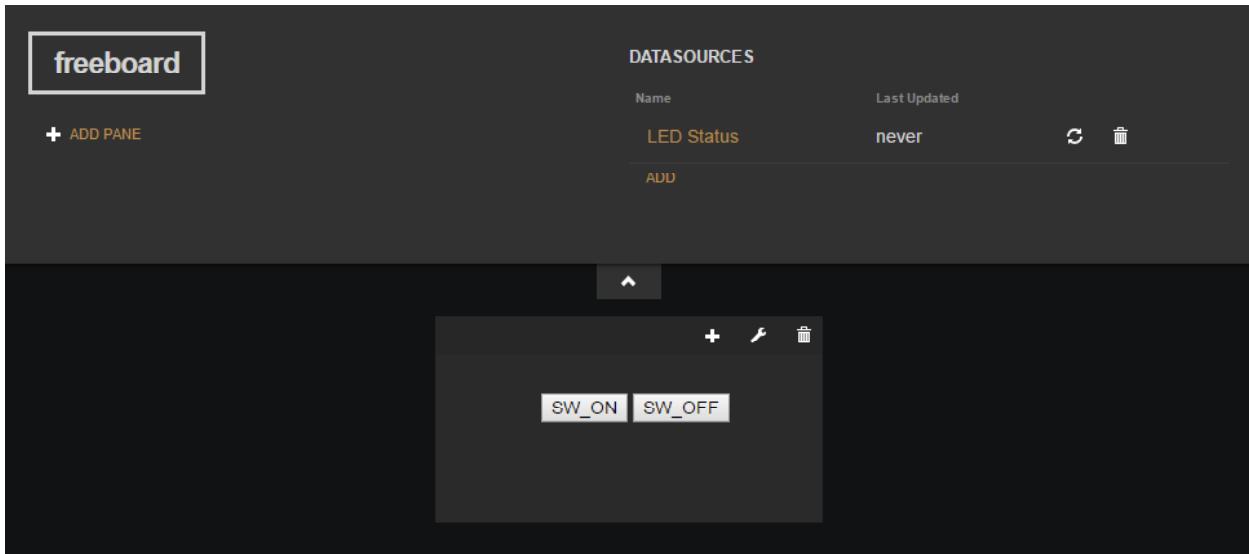
```
<script>
var APPKEY = 'Yn1yT7QqyKyYFk6';
var APPSECRET = 'Ksn1hYJ2ELSNjsKb2iiJa6YIg';
var APPID = 'GroupInex';
var Topic = '/Send_stLED';
function switchPressoff()
{
    var url='https://api.NETPIE.io/topic/'+APPID+Topic+'?retain&auth='
    '+APPKEY+':'+APPSECRET;
    var xhttp = new XMLHttpRequest();
    xhttp.open('PUT',url,true);
    xhttp.send('0');
}
function switchPresson()
{
    var url='https://api.NETPIE.io/topic/'+APPID+Topic+'?retain&auth='
    '+APPKEY+':'+APPSECRET;
    var xhttp = new XMLHttpRequest();
    xhttp.open('PUT',url,true);
    xhttp.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 จะกลับมาขึ้นหน้าต่างเลือกรูปแบบของอุปกรณ์ แสดงผลอีกครั้ง 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

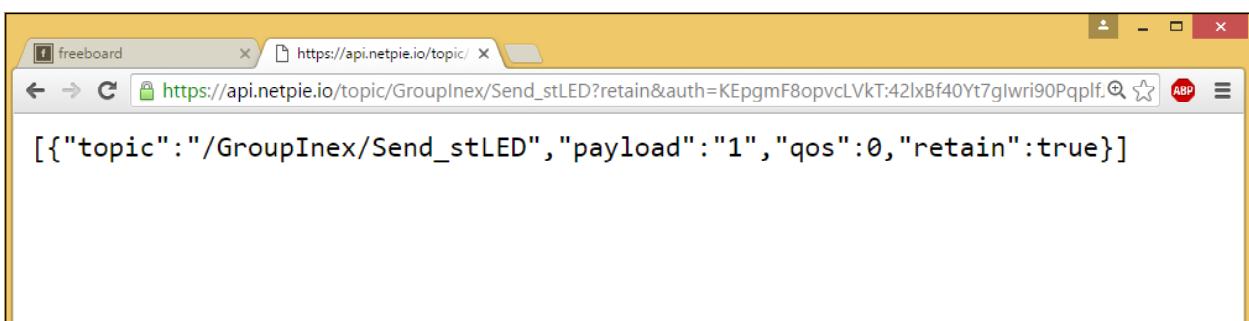
มีขั้นตอนดังนี้

(5.2.2.1) เปิดเว็บбраウเซอร์ แล้วป้อน URL ต่อไปนี้ที่แอ็คเดรสบาร์ของเว็บбраウเซอร์

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

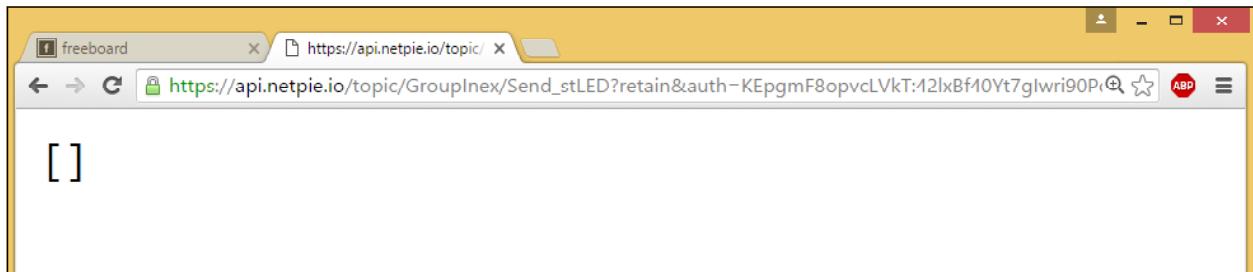
(5.2.2.2) ถ้าหากการส่งและรับค่าลูกศรต้อง อ่านค่าผลลัพธ์ได้ จะปรากฏข้อความต่อไปนี้ที่เว็บбраウเซอร์ ดังรูปที่ 5-12

```
[{"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 <AuthClient.h>
#include <MicroGear.h>
#include <MQTTClient.h>
#include <SHA1.h>
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
#include <EEPROM.h>

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

#define APPID      "GroupInex"
#define APPKEY     "Yn1yT7QqyKyYFk6"
#define APPSECRET  "Ksn1hYJ2ELSNjsKb2iiJa6YIg"
#define ALIAS      "espLED"

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

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

```

WiFiClient client;
AuthClient *authclient;
MicroGear microgear(client);
unsigned long previousMillis = 0;
const long interval = 1000;

void onMsghandler(char *topic, uint8_t* msg, unsigned int msglen)
{
    Serial.print("Incoming message -> ");
    Serial.print(topic);
    Serial.print(" : ");
    char strState[msglen];
    for (int i = 0; i < msglen; i++)
    {
        strState[i] = (char)msg[i];
        Serial.print((char)msg[i]);
    }
    Serial.println();
    String Topic=topic;
    String stateStr = String(strState).substring(0,msglen);
    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...");
}

```

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

```

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(APPKEY,APPSECRET,ALIAS);
    microgear.connect(APPID);
}
}

void loop()
{
    if (microgear.connected())
    {
        microgear.loop();
    }
    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 ที่กำหนดศิทธิ์การเข้าถึงไว้ด้วยคำสั่ง microgear.on(MESSAGE, onMsgHandler) จากนั้นเชื่อมต่อ WiFi

เมื่อเชื่อมต่อได้สำเร็จจะเชื่อมต่อกับ NETPIE ด้วยคำสั่ง microgear.connect(APPID)

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

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

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 เมื่อใช่ Topic นี้จะสั่งให้ LED ที่ต่อ กับขา D4 ทำงานด้วยคำสั่ง `digitalWrite(LEDPin,st)` และอ่านสถานะของขา D4 อีกครั้งพร้อมกับตรวจสอบโดยใช้คำสั่ง `if(digitalRead(LEDPin) == HIGH)` เพื่อส่งผลลัพธ์ไปยัง NETPIE

### หากสถานะขาเป็น HIGH

จะส่งกลับด้วยคำสั่ง `microgear.publish("/Feedback_stLED","1",1)`

### ถ้าสถานะขาเป็น LOW

จะตอบกลับด้วยคำสั่ง `microgear.publish("/Feedback_stLED","0",1)`

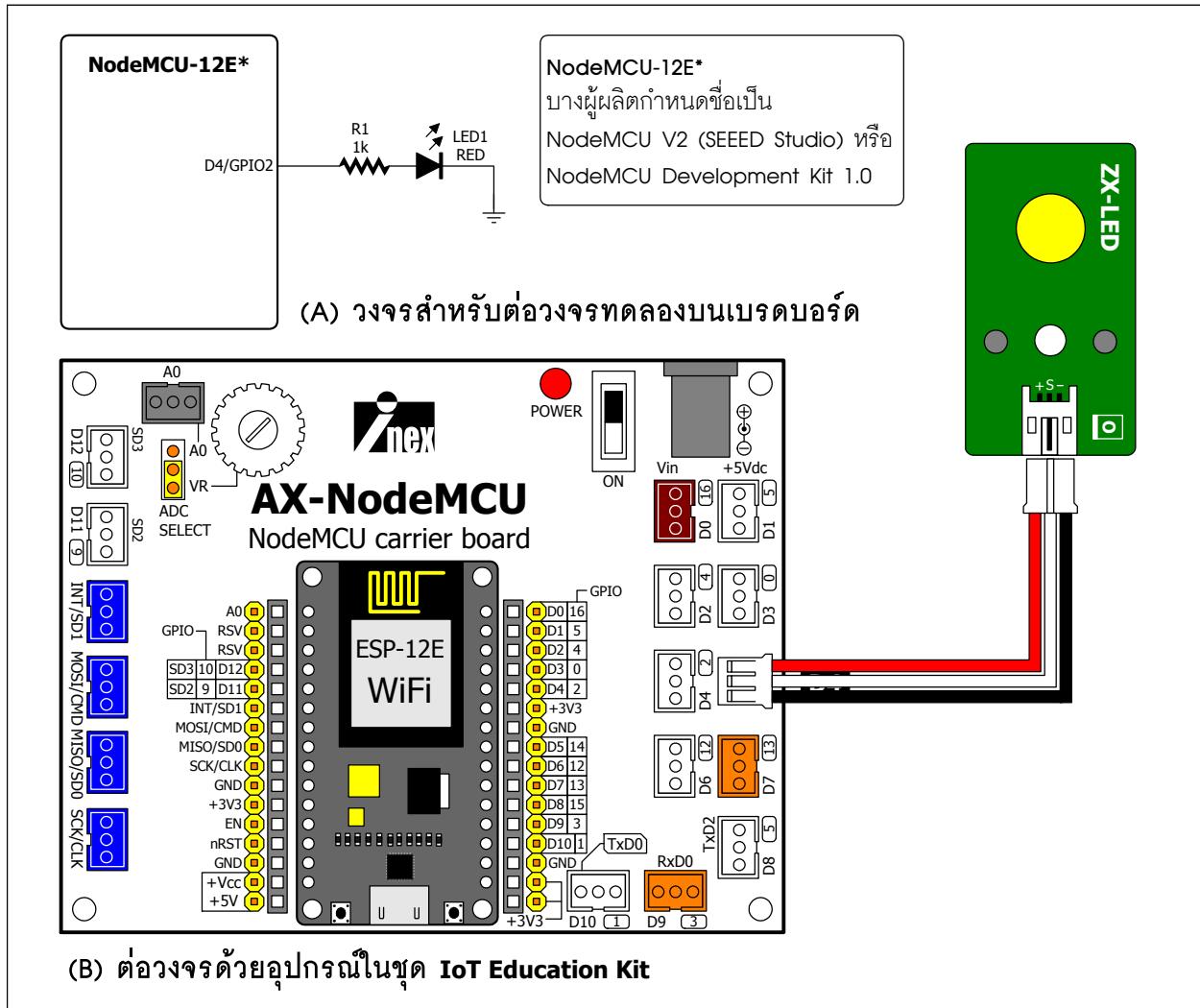
สาเหตุที่ต้องส่งสถานะกลับไป เนื่องจากว่า ถ้าอุปกรณ์เกิดความเสียหายหรืออุปกรณ์เชื่อมต่ออินเทอร์เน็ต ไม่ได้ ผู้ใช้งานจะสามารถตรวจสอบอุปกรณ์ได้ด้วย

การทำงานในฟังก์ชัน `loop` จะคอยตรวจสอบว่า อุปกรณ์ยังคงเชื่อมต่อ กับ NETPIE อยู่หรือไม่ด้วยคำสั่ง `if(microgear.connected())`

ถ้าเชื่อมต่ออยู่ ฟังก์ชัน `microgear.loop()` จะคอยตรวจสอบการเกิดเหตุการณ์ต่างๆ

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

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



รูปที่ 5-14 วงจรและการต่อวงจรเพื่อควบคุมการทำงานของ LED ที่ต่อกับ NodeMCU-12E ผ่าน NETPIE

(5.2.3.3) ต่อ LED เข้าที่ขาพอร์ต D4 ดังรูปที่ 5-14

(5.2.3.4) ตรวจสอบสถานะการตอบกลับของ NodeMCU-12E โดยเปิดเว็บเบราว์เซอร์แล้วป้อน URL ต่อไปนี้ไปยังแอ็คเดรสนาร์

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

(5.2.3.5) หากการเชื่อมต่อถูกต้อง จะได้รับข้อความตอบกลับดังรูปที่ 5-15

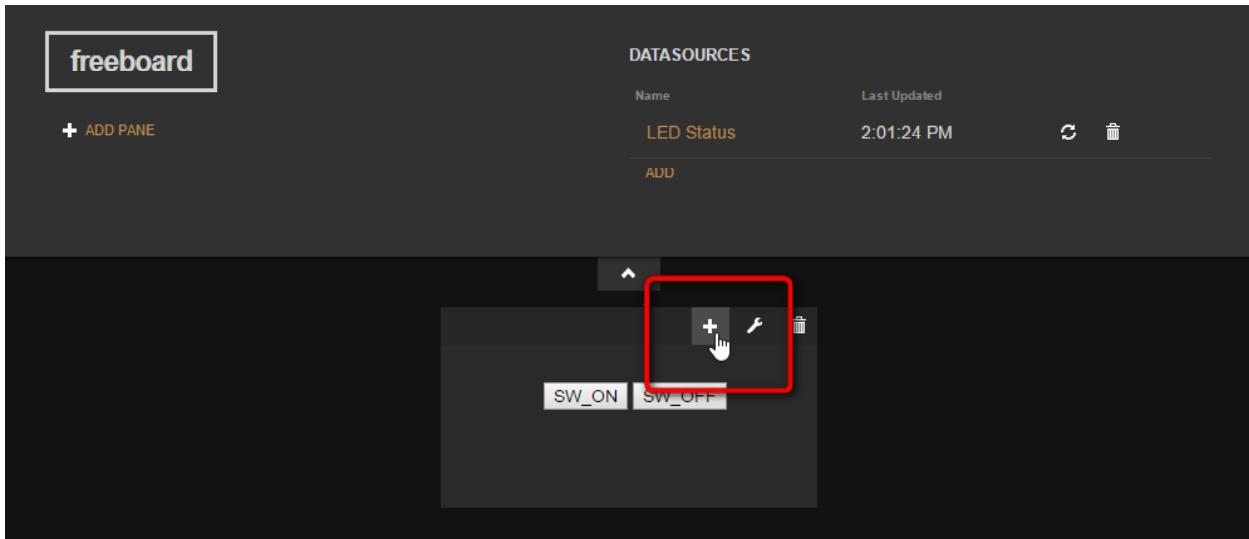


รูปที่ 5-15 แสดงการตอบกลับจาก NETPIE เมื่อได้รับข้อมูลจาก NodeMCU-12E

## 5.2.4 เพิ่มอุปกรณ์แสดงผล

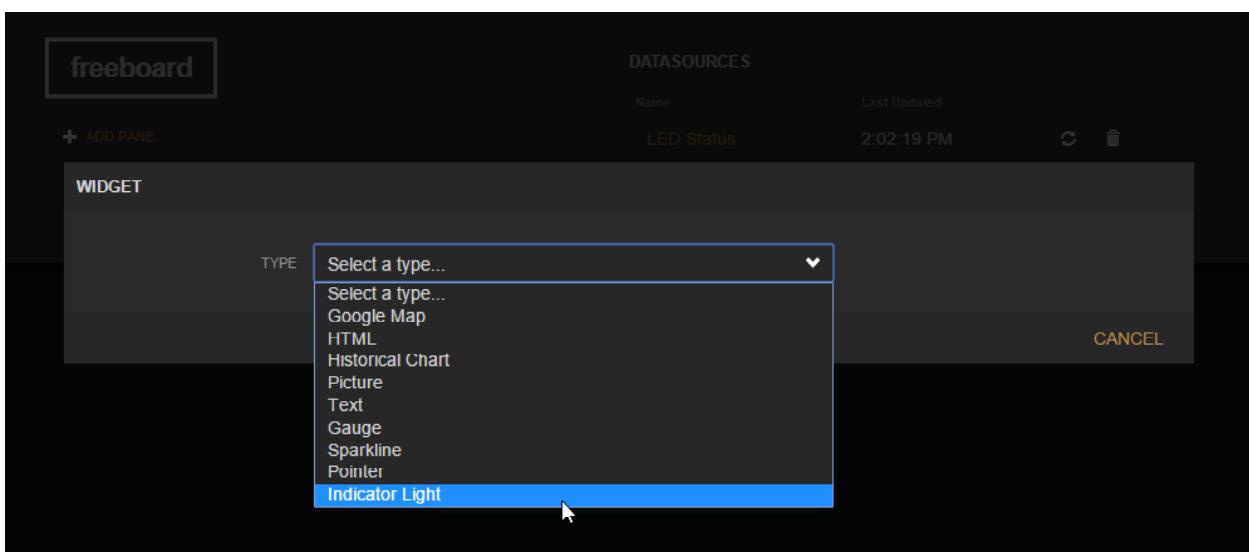
กลับไปที่แดชบอร์ด LED Status ที่ freeboard อีกครั้งเพื่อเพิ่มอุปกรณ์แสดงผล มีขั้นตอนดังนี้

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



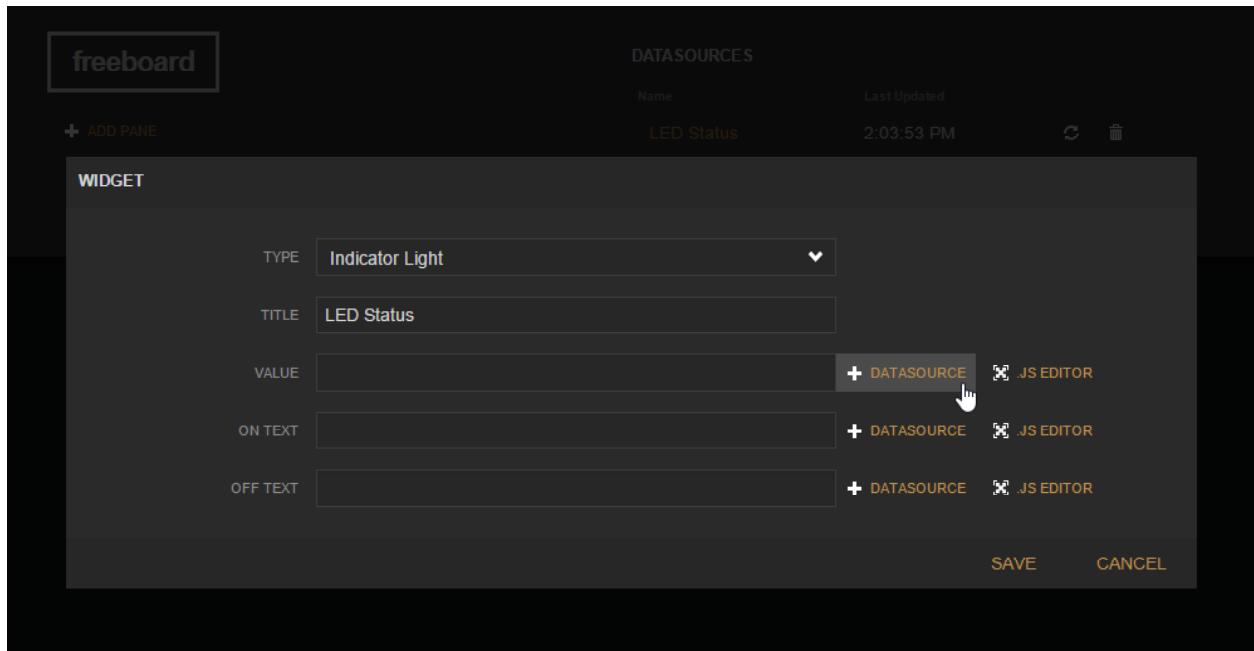
รูปที่ 5-16 เพิ่มอุปกรณ์แสดงผลให้แก่แดชบอร์ด

(5.2.4.2) เลือกชนิด (TYPE) ของอุปกรณ์แสดงผลหรือวิดเก็ต ในรูปแบบ Indicator Light ดังรูปที่ 5-17



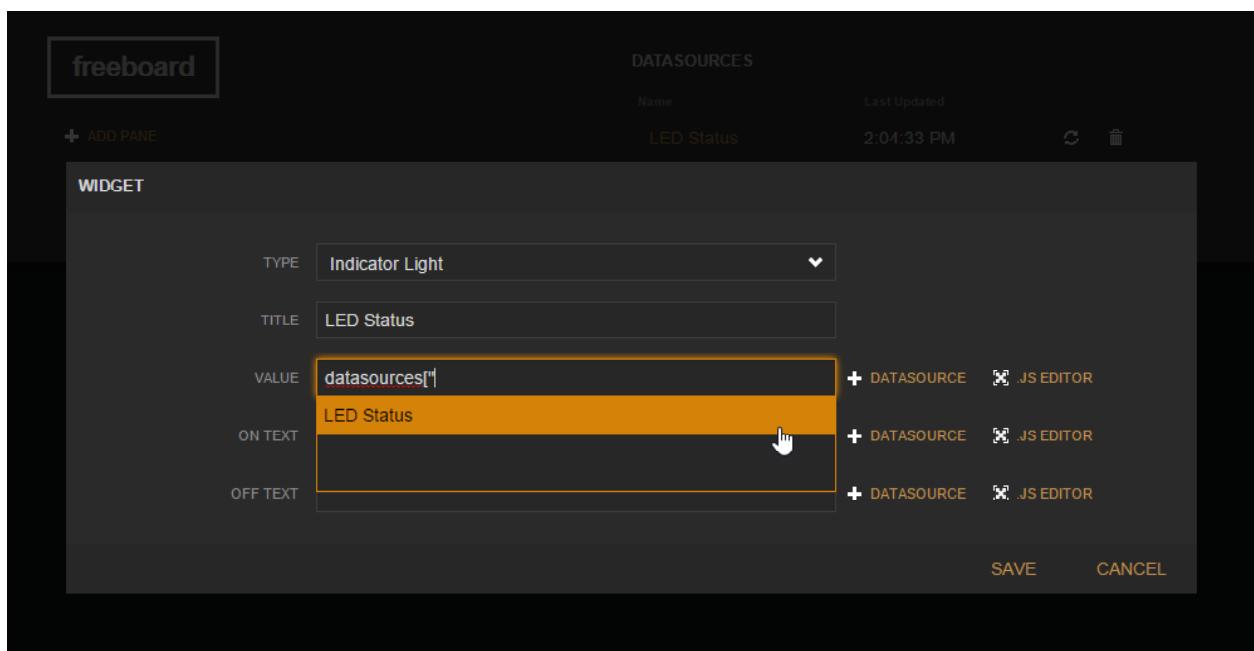
รูปที่ 5-17 เลือกชนิดของอุปกรณ์แสดงผลเป็น Indicator Light สำหรับแดชบอร์ด LED Status

(5.2.4.3) จากนั้นกำหนดรายละเอียดขึ้นต้นตามรูปที่ 5-18



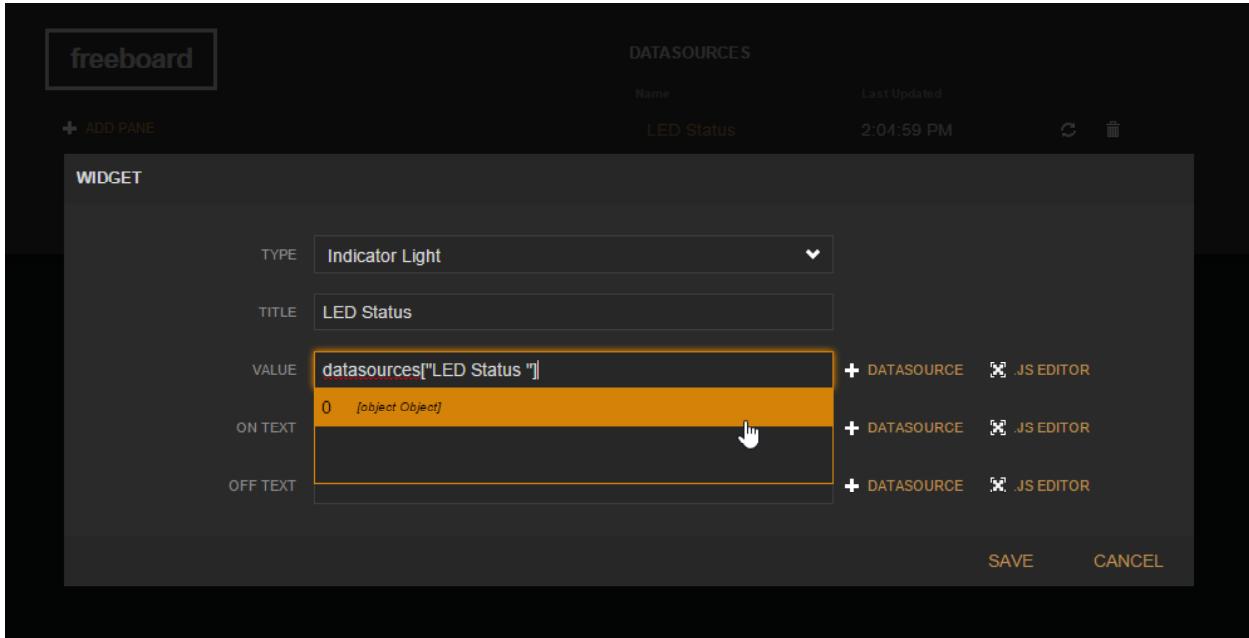
รูปที่ 5-18 ทำการตั้งค่าให้กับ Indicator Light ที่ใช้เป็นตัวแสดงผลของเดชบอร์ด LED Status

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



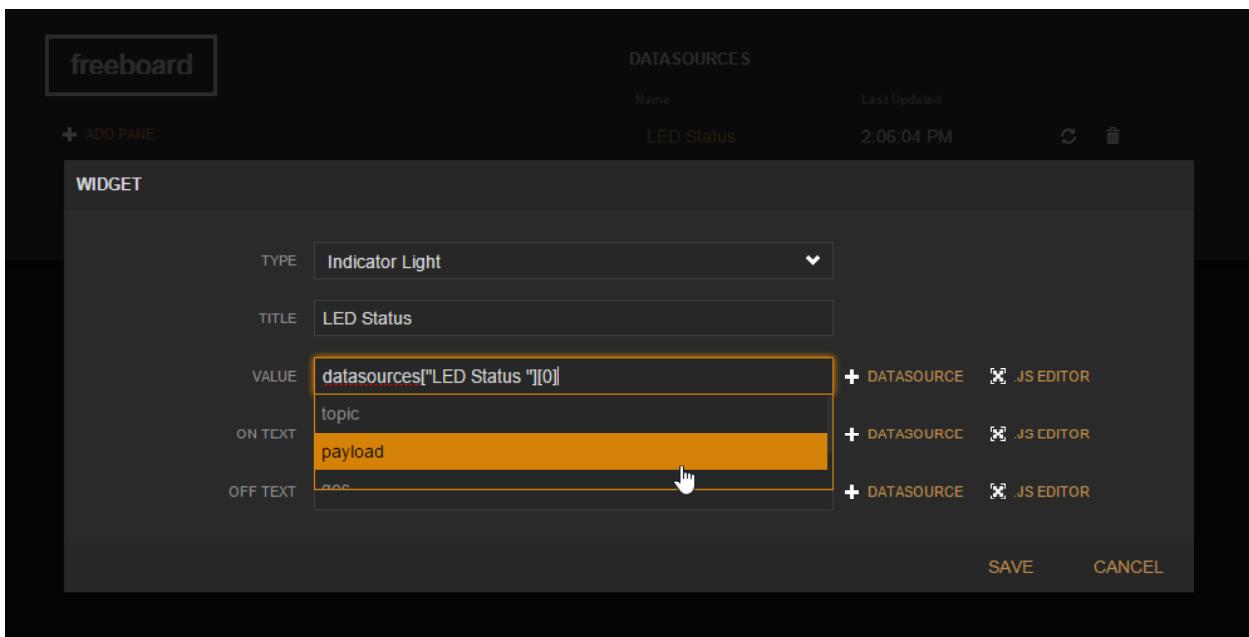
รูปที่ 5-19 เลือกแหล่งข้อมูลเป็น LED Status ที่ซ่อง VALUE ของ DATASOUCE ในอุปกรณ์แสดงผลของเดชบอร์ด LED Status

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



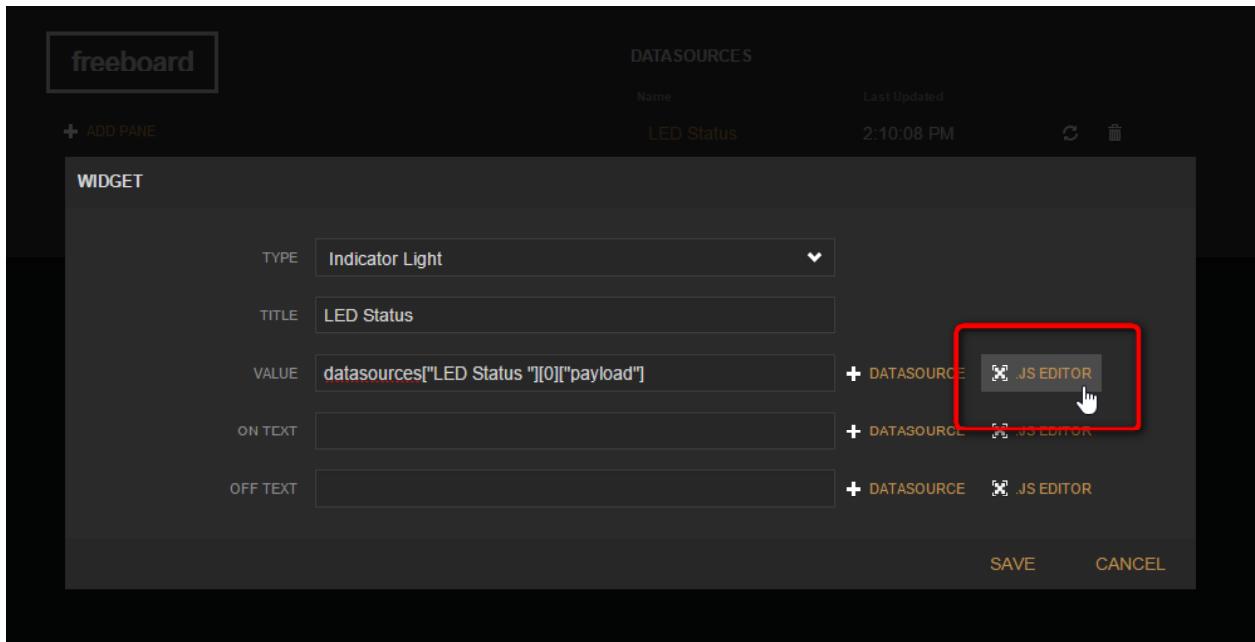
รูปที่ 5-20 เลือกชนิดข้อมูลของ LED Status ที่ช่อง VALUE ของ DATASOURCE

(5.2.4.6) แล้วเลือกชนิดข้อมูล payload ดังรูปที่ 5-21



รูปที่ 5-21 เลือกชื่อหัวข้อการติดต่อเป็น payload ที่ช่อง VALUE ของ DATASOURCE

(5.2.4.7) ทำการแก้ไข DATASOURCE โดยคลิกที่ **JS EDITOR** เพื่อเปิดหน้าต่างエ迪เตอร์ของ javascrip 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 หรือ javascrip ต้องแก้ไขข้อมูลของแหล่งกำเนิดข้อมูลที่เลือกมาใช้งาน จากในรูปคือ LED Status และ payload

(5.2.4.9) เพิ่มและแก้ไขสคริปต์คำสั่งดังต่อไปนี้

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

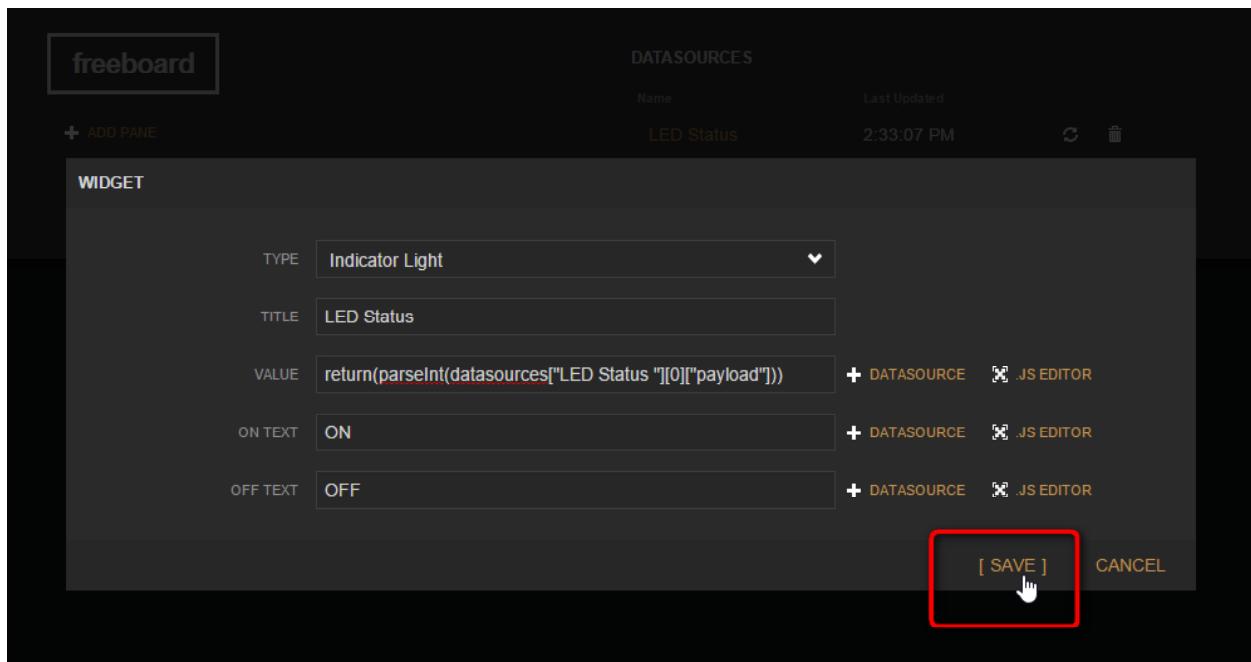
ถ้าสั่งเกตจากการตอบกลับของ NETPIE ค่าของ payload จะให้ผลลัพธ์ในรูปแบบ String แต่เนื่องจากค่า Value ที่ต้องการคือ True กับ False หรือ 0 กับ 1 ในรูปแบบ Integer ดังนั้นการแก้ไขสคริปต์จึงเป็นการแปลงค่าจากตัวแปร String เป็น Integer

(5.2.4.10) กลับมาที่หน้าต่างตั้งค่า ให้เพิ่มรายละเอียดในส่วนที่เหลือ

ที่ช่อง ON TEXT เลือกเป็น ON

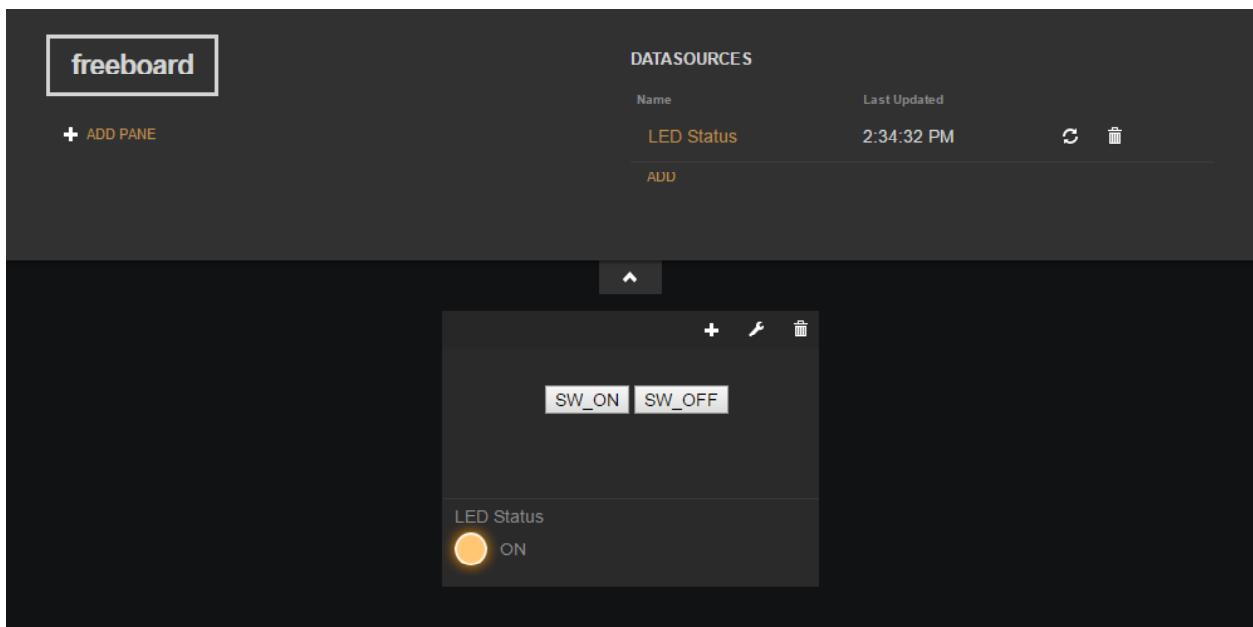
ที่ช่อง OFF TEXT เลือกเป็น OFF

แล้วคลิก SAVE ดังรูปที่ 5-24



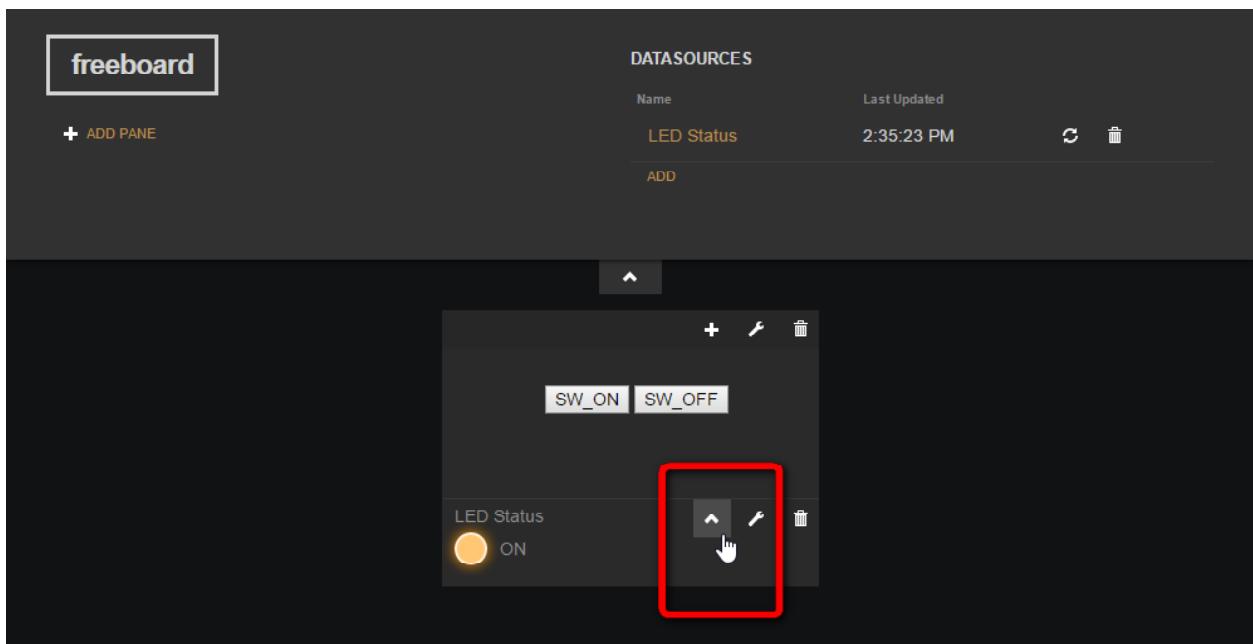
รูปที่ 5-24 หน้าต่างแสดงบทสรุปของการตั้งค่าสำหรับอุปกรณ์แสดงผลที่ชื่อ LED Status สำหรับใช้ในแดชบอร์ดควบคุม LED

(5.2.4.11) จะได้เดชบอร์ดของการควบคุม LED ผ่านอินเทอร์เน็ตดังรูปที่ 5-25



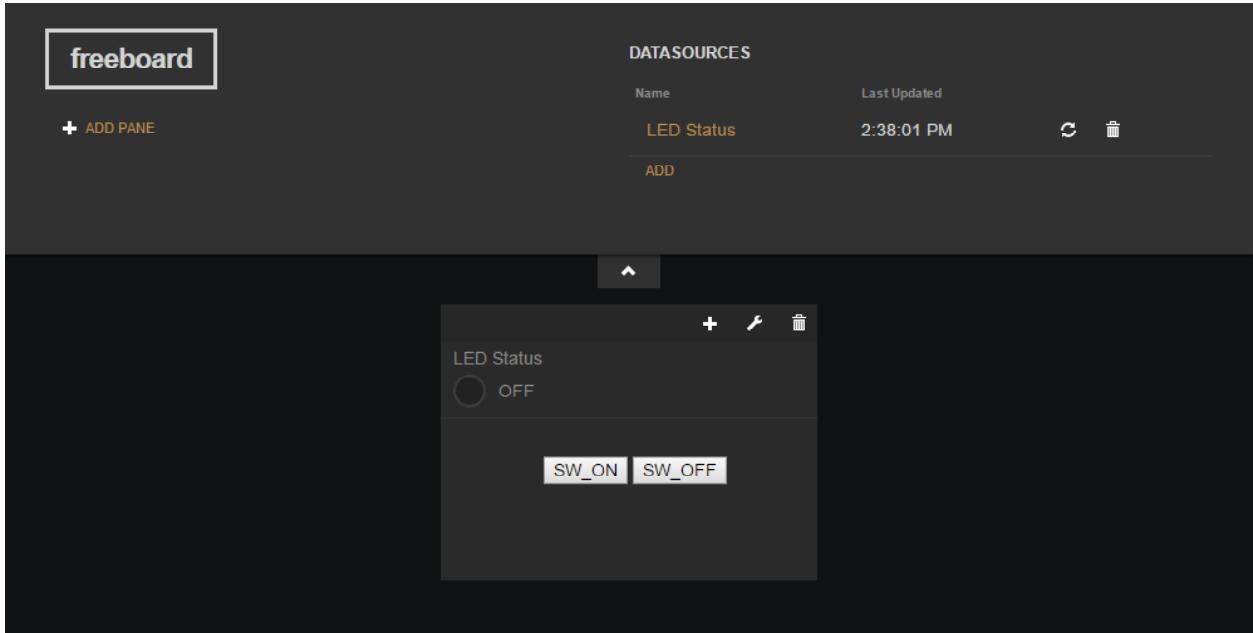
รูปที่ 5-25 เดชบอร์ดสำหรับควบคุม LED ผ่าน NETPIE บนเว็บไซต์ freeboard.io เมื่อสร้างเสร็จ

(5.2.4.12) ถ้าต้องการให้ LED เลื่อนขึ้นไปด้านบน ให้คลิกที่ลูกศร ดังรูปที่ 5-26 จนได้ตำแหน่งตามต้องการ



รูปที่ 5-26 แสดงการปรับแต่งตำแหน่งของรูป LED ด้วยปุ่มลูกศรบนเดชบอร์ด LED Status

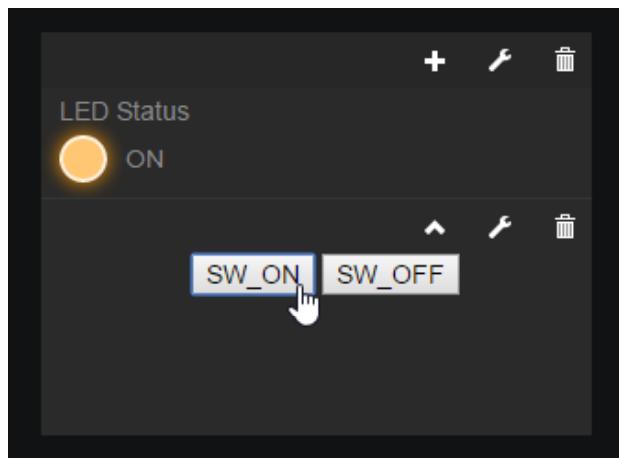
จะได้เดชบอร์ดสำหรับควบคุม LED ผ่านอินเทอร์เน็ต โดยใช้ NETPIE เป็นคลาวด์เซิร์ฟเวอร์ และแสดงผลด้วย freeboard และควบคุมอุปกรณ์ด้วย NodeMCU-12E ดังรูปที่ 5-27



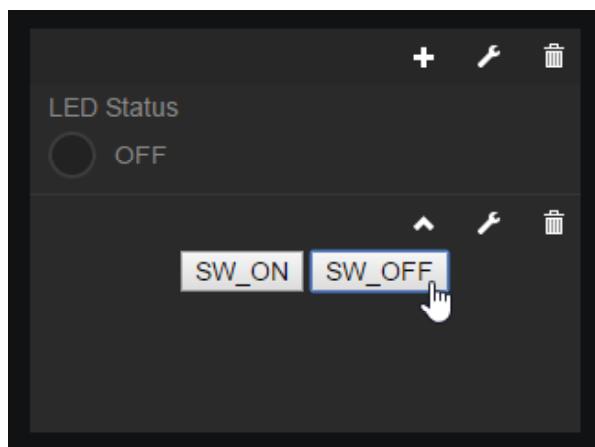
รูปที่ 5-27 NETPIE LED เดชบอร์ดที่สร้างขึ้นเพื่อควบคุม LED ที่ต่ออยู่กับขา D4 ของ NodeMCU-12E โดยทำงานผ่าน NETPIE และแสดงผลด้วย freeboard.io

### 5.3 ทดสอบการทำงานทั่วระบบ

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



(5.3.2) คลิกที่ปุ่ม SW\_OFF จะทำให้ LED เปลี่ยนเป็นสีขาว แทนการดับ และ LED ที่ต่อ กับขา D4 ของ NodeMCU-12E จะดับลงตามด้วย การดับของ LED อาจช้ากว่าการแสดงผลที่หน้าแดชบอร์ด ขึ้นอยู่กับความเร็วของการต่อสารข้อมูลบนเครือข่ายอินเทอร์เน็ต เช่นกัน



## สรุปท้าย

ทั้งหมดที่นำเสนอในหนังสือเล่มนี้เป็นการแนะนำให้ผู้สนใจเริ่มต้นพัฒนาอุปกรณ์ IoT ได้รู้จักกับการใช้งานคลาวด์เซอร์ฟเวอร์ NETPIE ที่พัฒนาโดยคณะนักพัฒนาชาวไทย ภายใต้การสนับสนุนโดยสำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ หรือ สวทช. จะเห็นว่า NETPIE มีความสามารถที่ดีเพียงพอและใช้งานไม่ยาก มีไลบรารีที่ช่วยให้อุปกรณ์ทางสารคดแวร์สมัยใหม่อย่าง ESP8266 หรือ NodeMCU ให้สามารถเชื่อมต่อกับ NETPIE ได้สะดวก รวมถึงไลบรารีสำหรับ NodeJS และ HTML5 เพื่อรองรับการพัฒนาแอปพลิเคชันของอุปกรณ์ IoT ด้วย

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

สำหรับนักพัฒนาที่มีความชำนาญแล้ว และต้องการให้ NETPIE รองรับอุปกรณ์มากกว่า 100 ตัวภายในได้ Application ID ของตัวนักพัฒนาของทาง NETPIE ที่มีบริการแบบ Enterprise ให้ใช้งานซึ่งจะมีค่าใช้จ่ายเพิ่มเติมอย่างเหมาะสม

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

