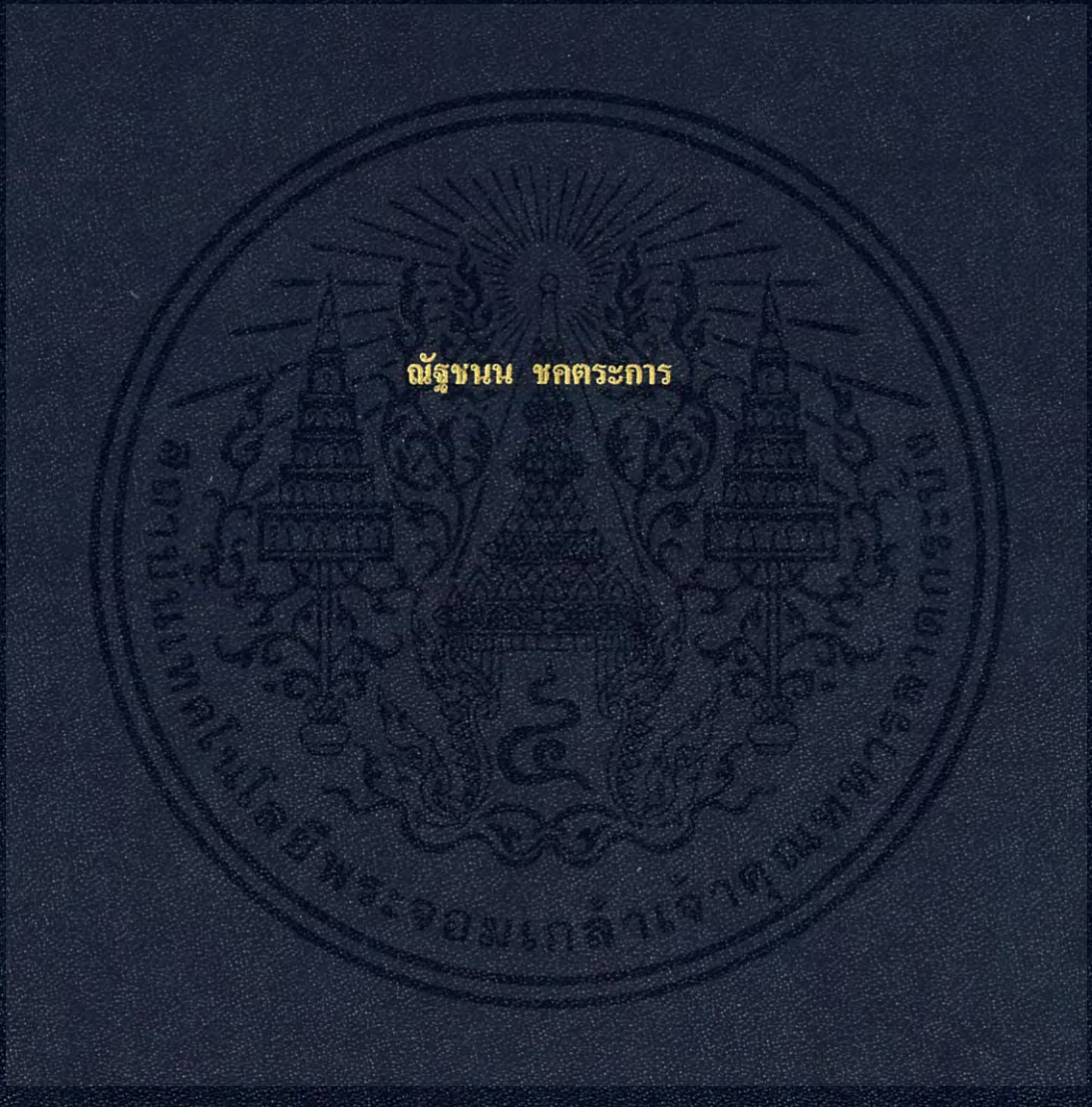
# การประยุกต์ใช้เครือข่ายแบบเมช

# MESH NETWORK APPLICATION



ณัฐชนน ชคตระการ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2561

# การประยุกต์ใช้เครือข่ายแบบเมช

# MESH NETWORK APPLICATION

ณัฐชนน ชคตระการ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2561

ปริญญานิพนธ์ปีการศึกษา 2561

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การประยุกต์ใช้เครือข่ายแบบเมช

MESH NETWORK APPLICATION

ผู้จัดทำ

1. นายณัฐชนน ชคตระการ รหัสนักศึกษา 58010366

_____ อาจารย์ที่ปรึกษา

(ผู้ช่วยศาสตราจารย์ ธนา หงษ์สุวรรณ)

# การประยุกต์ใช้เครือข่ายแบบเมช

นายณัฐชนน ชคตระการ 58010366
ผู้ช่วยศาสตราจารย์ ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา
ปีการศึกษา 2561

**บทคัดย่อ**

ปริญญานิพนธ์นี้จัดทำขึ้นเพื่อส่งเสริมการเรียนรู้และการประยุกต์ใช้งานระบบเครือข่ายไร้สายแบบเมชโดยในโครงงานนี้จะทำการประยุกต์ใช้งานในรูปแบบบ้านสวนอัจฉริยะซึ่งจะมีการสั่งการควบคุมการเปิด/ปิดไฟหรือน้ำผ่านอุปกรณ์ และ การดูอุณหภูมิ/ความชื้นผ่านเซนต์เซอร์ ในโครงงานนี้ผู้ศึกษาสามารถเรียนรู้และฝึกใช้งานการเขียนโปรแกรมสร้างเครือข่ายไร้สายแบบเมชผ่านบอร์ดทดลอง ESP32 ไปจนถึงการตั้งค่าและเขียนโปรแกรมเพื่อใช้งานการเชื่อมต่อภายนอกผ่านโปรโตคอล MQTT และ แสดงผลลัพท์ผ่าน Web Browser

# MESH NETWORK APPLICATION

Mr.Nutchanon Chakhatrakan 58010366

Asst. Prof.Thana Hongsuwan Advisor

Academic Year 2018

## ABSTRACT

This thesis is designed to promote the learning and application of wireless mesh networks. In this project will be applied in the form of smart garden home , which will command to controls on / off the lights and watering ,and view current temperature / humidity via sensor. In this project, learners can learn and practice programming, creating a wireless mesh network, with the ESP32 experiment board, to set up and programmable external interfaces via the MQTT protocol and show online result via web browser.

# กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดีด้วยความช่วยเหลือจากหลายฝ่ายทั้งในทางตรงและทางอ้อม ปริญญานิพนธ์ฉบับนี้จะสำเร็จลงไม่ได้หากปราศจากความช่วยเหลือของบุคคลเหล่านี้

ขอขอบคุณ อาจารย์ที่ปรึกษา คือ อาจารย์ธนา หงษ์สุวรรณ เป็นผู้ให้คำแนะนำ คำปรึกษา และให้ความช่วยเหลือตลอดการทำโครงงาน ซึ่งทำให้การทำงานต่างๆเป็นไปได้อย่างราบรื่นและสำเร็จลุล่วงไปด้วยดี

ขอขอบคุณอาจารย์และบุคลากรต่างๆในสาขาวิชาวิศวกรรมคอมพิวเตอร์ที่ได้ให้คำแนะนำและสั่งสอนความรู้ต่างๆมาโดยตลอด

ขอขอบคุณรุ่นพี่และเพื่อนหลายๆคนในภาควิชาวิศวกรรมคอมพิวเตอร์ที่ได้ให้คำแนะนำ คำปรึกษา และแบ่งปันความรู้ในทุกๆด้าน

ในสุดท้ายนี้ ขอขอบคุณบิดา มารดา และครอบครัวที่ให้การเลี้ยงดู สั่งสอน และให้การสนับสนุนพร้อมทั้งให้โอกาสในการศึกษาและให้กำลังใจเสมอมา

ณัฐชนน ชคตระการ

# สารบัญ

หน้า

# สารบัญ(ต่อ)

หน้า

# สารบัญ(ต่อ)

หน้า

# สารบัญรูป

รูป หน้า

# บทที่ 1
# บทนำ

## 1.1 ที่มาและความสำคัญ

ในปัจจุบันมีเทคโนโลยี IoT สำหรับควบคุมอุปกรณ์ภายในบ้านหลายอย่าง เช่น ควบคุมการเปิดปิดของเครื่องใช้ไฟฟ้า ควบคุมกลอนประตู ดูอุณหภูมิในบ้าน กล้องตรวจจับขโมย และอื่นๆ ทว่า อุปกรณ์ IoT ทั้งหลายนี้สามารถใช้ได้ในบริเวณที่มีสัญญาณ WIFI เท่านั้น หากใช้ในพื้นที่กว้าง บริเวณที่ไม่มีสัญญาณ WIFI ซึ่งทำให้ไม่สามารถใช้งานอุปกรณ์เหล่านี้ได้ ซึ่งหากต้องการนำอุปกรณ์ IoT ไปใช้ในระยะไกลสามารถต้องใช้ Repeater หรือติดตั้งเครื่องปล่อยสัญญาณ WIFI เพิ่มอีกทั้งยังต้องติดตั้งสายไฟสำหรับอุปกรณ์แต่ละตัว

ดังนั้นจึงมีแนวคิดว่าหากใช้ระบบ WIFI Mesh Network จะสามารถใช้ในบริเวณกว้างได้ง่ายขึ้น เช่น สามารถใช้เปิด/ปิดโคมไฟในสวนหรือประตูรั้ว ใช้เปิด/ปิดระบบรดน้ำในสวนไร่หรือสนามหญ้า ใช้เปิด/ปิดระบบน้ำพุประดับในสวน ใช้รับเซนเซอร์อุณหภูมิ/ความชื้นในสวนหรือไร่ ใช้ตรวจจับขโมยที่เข้ามาในสวนไร่ ใช้ตรวจวัดสภาพดินในสวนไร่ทางการเกษตร ใช้ตรวจวัดคุณภาพน้ำในสวนไร่ทางการเกษตร

## 1.2 วัตถุประสงค์ของโครงงาน

1) นำระบบ WIFI Mesh Network มาใช้ในเทคโนโลยี IoT เพื่อใช้ในบริเวณที่สัญญาณ WIFI ไม่สามารถครอบคลุมได้อย่างทั่วถึงได้ด้วยเครื่องปล่อยสัญญาณ WIFI เพียงเครื่องเดียว
2) ส่งเสริมการเรียนรู้การใช้อุปกรณ์ IOT และ WIFI Mesh Network
3) เพื่อให้ผู้ที่มาศึกษาโครงงานนี้สามารถนำ WIFI Mesh Network ไปประยุกต์ใช้ได้ง่ายขึ้น

## 1.3 ประโยชน์ที่คาดว่าจะได้รับ

1) สามารถใช้ WIFI Mesh Network ในการตรวจสอบและควบคุมเซนเซอร์แบบต่างๆ
2) เรียนรู้วิธีใช้และวิธีการประยุกต์ใช้ WIFI Mesh Network

### 1.4 ขอบเขตของโครงงาน

สร้างโมดูลเอนกประสงค์ ที่ทำงานบน WIFI Mesh Network โดยมี Input 2 ช่อง และ Output 2 ช่อง และสามารถควบคุมผ่านอินเทอร์เน็ตได้

# บทที่ 2
# ทฤษฎีที่เกี่ยวข้อง

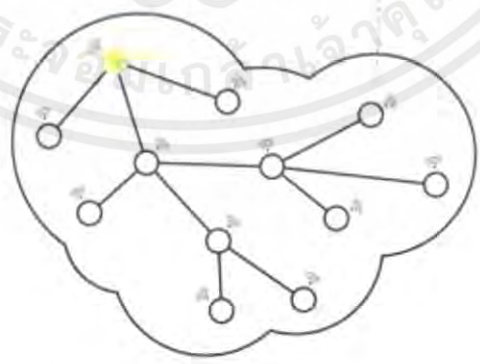
## 2.1 WIFI Mesh Network

### 2.1.1 WIFI Mesh Network คืออะไร?

Mesh Network คือ โครงสร้างของ Network แบบหนึ่งที่มีลักษณะที่แต่ละเครื่อง ซึ่งเรียกว่า โหนด หรือ Node ซึ่งเชื่อมต่อถึงกันหมดและทำงานร่วมกันในการส่งผ่านข้อมูลจากโหนดหนึ่งไปยังอีกโหนดหนึ่งซึ่งการทำงานในการส่งผ่านข้อมูลไม่ขึ้นตรงต่อกันแบบตายตัว นั่นหมายความว่าโครงสร้างของ Mesh นั้นสามารถปรับเปลี่ยนได้ตลอดเวลา ซึ่งทำให้เมื่อมีโหนดไหนเสียหาย มันจะทำการหาเส้นทางอื่นในการส่งอัตโนมัติ

WIFI Mesh Network คือระบบ Mesh Network ที่ใช้กับระบบไร้สายผ่าน WIFI ซึ่งจะมีความสามารถในการค้นหาและเชื่อมต่อโหนดใหม่ได้ด้วยตัวมันเองดังนั้นหากโหนดใดเสียหายก็สามารถติดตั้งโหนดใหม่เข้ามาเพื่อใช้แทนโหนดเดิมได้เลยโดยไม่จำเป็นต้องตั้งค่าใหม่

ซึ่งโหนดสามารถเป็นได้ทั้ง Router Mobile Phone อุปกรณ์ขนาดเล็ก โดยในเครือข่ายจะมีโหนดบางเป็นโหนดสำหรับติดต่อภายนอกเครือข่ายเช่น เชื่อมต่อกับอินเตอร์เน็ต



**รูปที่ 2.1 Wireless Mesh Network**

### 2.1.2 ข้อดีของ WIFI Mesh Network

WIFI Mesh Network นั้นมีข้อดีดังต่อไปนี้

- ต้นทุนน้อยลง โดยเฉพาะเมื่อต้องการวางระบบเน็ตเวิร์กที่ครอบคลุมเนื้อที่กว้าง
- การใช้จำนวนโหนดที่เพิ่มขึ้นหมายถึง ความเร็วในการส่งข้อมูลภายในเพิ่มขึ้น
- สะดวก โดยเฉพาะในที่ที่การวางสายบางครั้งเป็นเรื่องลำบาก เช่น ในสนามกีฬาขนาดใหญ่ โรงงาน หรือ บนอุปกรณ์ที่เคลื่อนที่ได้
- เหมาะอย่างยิ่งสำหรับบางสถานที่ ที่ในบางครั้งสัญญาณจะโดนปิดกั้น เช่น ในสวนสนุกเมื่อบางครั้ง เครื่องเล่นวิ่งผ่านสัญญาณขาดหายบางช่วง แต่ถ้าเป็น Mesh Network การสื่อสารจะใช้โหนดอื่นๆ ข้างเคียงส่งข้อมูลแทนได้
- WIFI Mesh network นั้นเรียกได้ว่า เป็นเน็ตเวิร์กที่ "ปรับตัวเอง" คือ รู้จักโหนดใหม่อัตโนมัติ โดยไม่ต้องอาศัยผู้ดูแล
- WIFI Mesh network ซ่อมแซมตัวเองได้ กรณีที่โหนดบางตัวไม่สามารถทำงานได้ การค้นหา และเปลี่ยนเส้นทางไปยังโหนดอื่นๆ ข้างเคียงเกิดขึ้นโดยอัตโนมัติ
- ติดตั้งง่าย ขยายง่าย

### 2.1.3 การประยุกต์ใช้งาน

ตัวอย่างการประยุกต์ใช้งานเช่น

- ตรวจสอบและควบคุมอุปกรณ์ในบ้าน
- ตรวจสอบและควบคุมอุปกรณ์ในโรงงานอุตสาหกรรม
- สื่อสารและสั่งการกองทัพ
- ตรวจสอบสภาพแวดล้อม

## 2.2 ESP 32

ESP32 เป็นชื่อของไอซีไมโครคอนโทรลเลอร์ที่รองรับการเชื่อมต่อ WIFI และ Bluetooth 4.2 BLE ในตัว ผลิตโดยบริษัท Espressif จากประเทศจีน เป็นไอซีที่พัฒนาเพิ่มเติมจาก ESP8266 โดยตัวไอซี ESP32 มีสเปคโดยละเอียด ดังนี้

- ซีพียูใช้สถาปัตยกรรม Tensilica LX6 แบบ 2 แกนสมอง สัญญาณนาฬิกา 240MHz
- มีแรมในตัว 512KB
- รองรับการเชื่อมต่อรอมภายนอกสูงสุด 16MB
- มาพร้อมกับ WIFI มาตรฐาน 802.11 b/g/n รองรับการใช้งานทั้งในโหมด Station softAP และ Wi-Fi direct
- มีบลูทูธในตัว รองรับการใช้งานในโหมด 2.0 และโหมด 4.0 BLE
- ใช้แรงดันไฟฟ้าในการทำงาน 2.6V ถึง 3V
- ทำงานได้ที่อุณหภูมิ -40°C ถึง 125°C

ขาใช้งานรองรับการเชื่อมต่อบัสต่าง ๆ ดังนี้

- รองรับ GPIO จำนวน 32 ช่อง
- รองรับ UART จำนวน 3 ช่อง
- รองรับ SPI จำนวน 3 ช่อง
- รองรับ I2C จำนวน 2 ช่อง
- รองรับ ADC จำนวน 12 ช่อง
- รองรับ DAC จำนวน 2 ช่อง
- รองรับ I2S จำนวน 2 ช่อง
- รองรับ PWM / Timer ทุกช่อง
- รองรับการเชื่อมต่อกับ SD-Card

รองรับฟังก์ชั่นเกี่ยวกับความปลอดภัยต่าง ๆ ดังนี้

- รองรับการเข้ารหัส WIFI แบบ WEP และ WPA/WPA2 PSK/Enterprise
- มีวงจรเข้ารหัส AES / SHA2 / Elliptical Curve Cryptography / RSA-4096 ในตัว

ในด้านประสิทธิ์ภาพการใช้งาน ตัว ESP32 สามารถทำงานได้ดี โดย

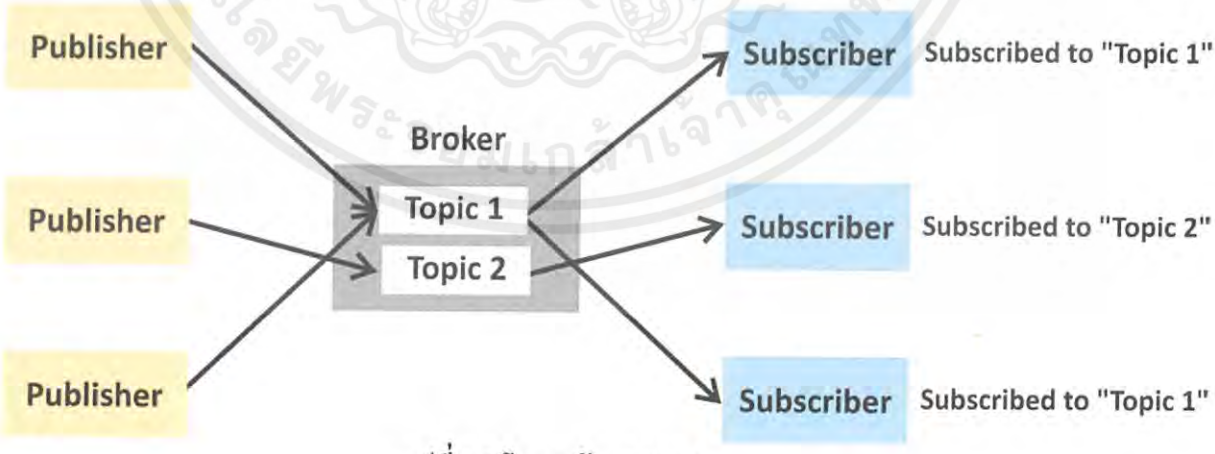
- รับ – ส่ง ข้อมูลได้ความเร็วสูงสุดที่ 150Mbps เมื่อเชื่อมต่อแบบ 11n HT40 ได้ความเร็วสูงสุด 72Mbps เมื่อเชื่อมต่อแบบ 11n HT20 ได้ความเร็วสูงสุดที่ 54Mbps เมื่อเชื่อมต่อแบบ 11g และได้ความเร็วสูงสุดที่ 11Mbps เมื่อเชื่อมต่อแบบ 11b
- เมื่อใช้การเชื่อมต่อผ่านโปรโตคอล UDP จะสามารถรับ – ส่งข้อมูลได้ที่ความเร็ว 135Mbps
- ในโหมด Sleep ใช้กระแสไฟฟ้าเพียง 2.5uA

## 2.3 MQTT

MQTT (Message Queuing Telemetry Transport) เป็นโปรโทคอลที่ออกแบบมาให้มีขนาดเล็กสำหรับการสื่อสารแบบ M2M ( Machine to Machine ) โดยถือกำเนิดจากวิศวกรจาก IBM และ Eurotech ในปี 1999 เพื่อนำไปใช้ในระบบ SCADA (Supervisory Control and Data Acquisition) สำหรับเชื่อมต่อท่อส่งน้ำมันบนเครือข่ายที่ไม่มีความเสถียรอย่างอินเตอร์เน็ตดาวเทียม ก่อนที่จะถูกบริจาคกลายเป็น Open Standard ในปี 2014 โดย OASIS

MQTT เป็นสถาปัตยกรรมแบบ Client/Server ซึ่งมี Topology แบบ hub-and-spoke sensor ปลายทางจะทำหน้าที่เป็น Client ซึ่งสร้างเชื่อมต่อแบบ TCP ไปยัง Server ที่มีชื่อเรียกอีกชื่อว่า Broker ซึ่งมีหน้าที่เป็นเสมือนท่อส่งข้อมูลในการรับส่ง 'Message' ระหว่าง Client ที่เป็นได้ทั้ง Publisher และ Subscriber นั่นเอง

- Client หมายถึง Publisher หรือ Subscriber ที่เชื่อมต่อแบบรวมศูนย์ไปยัง Broker ซึ่งสามารถเชื่อมต่อได้ทั้งแบบ persistent ที่สร้าง Session ค้างไว้เปิดตลอดเวลาเพื่อติดต่อกับ Broker ซึ่งตรงกันข้ามกับ Client ที่เชื่อมต่อแบบ Transient ซึ่ง Broker ไม่สามารถติดตามสถานะได้
- Broker เป็น Software ที่ทำหน้ารับข้อความทั้งหมดที่ได้จาก Publisher แล้วจึงส่งต่อไปให้ Subscriber ตามแต่ Topic ที่ Client ได้ Subscribe ไว้
- Topic เป็นเหมือน Address หรือ Endpoint บน Broker ที่ Client เชื่อมต่อเพื่อรับส่งข้อความระหว่างกัน




**รูปที่ 2.2 โครงสร้าง MQTT**

MQTT เป็นเหมือนสเปคของซอฟท์แวร์ที่มี API ไม่กี่ตัวในการเชื่อมต่อ Client เข้าด้วยกัน จึงไม่สามารถใช้เป็นตัวกลางในการจัดเก็บและกระจายข้อมูล ( Store-and-Forward ) เหมือนเช่นในระบบ MoM (Message Oriented Middleware) ที่ทำหน้าที่ในการจัดการคิวในการกระจายข้อมูลในระบบที่ต้องการความน่าเชื่อถือและมีข้อความจำนวนมาก ดังนั้นจึงมีการนำ MQTT ไปประยุกต์ใช้ร่วมกับ MoM เช่น RabbitMQ หรือ Redis เพื่อให้สามารถใช้งานได้อย่างมีประสิทธิภาพในเชิงพาณิชย์

### 2.3.1 ข้อดีของ MQTT

ถูกออกแบบมาให้มีขนาดเล็กเหมาะกับการนำไปใช้กับระบบคลาวด์ที่ให้บริการแบบรวมศูนย์เพราะถูกออกแบบให้เหมาะกับการกระจายข้อมูลแบบ Many-to-Many เช่น IoT Platform อีกทั้งยังเป็นมิตรกับ Network Engineer มากเนื่องจาก Device สามารถทำการสร้าง Session แลกเปลี่ยนข้อมูลกันได้โดยไม่ต้องทำการตั้งค่า NAT ให้วุ่นวายนักพัฒนาสามารถนำไปใช้กับร่วมกับ TLS/SSL เพื่อเพิ่มความปลอดภัยในการรับส่งข้อมูล

### 2.3.2 ข้อเสียของ MQTT

Client ทุกตัวต้องรองรับ TCP และทำการสร้างการเชื่อมต่อกับ Broker ไว้ตลอดเวลา ซึ่งอาจเกิดปัญหาได้หากอยู่ในเครือข่ายที่ไม่เสถียร

## 2.4 JSON

JSON ย่อมาจากคำว่า Java Script Object Notation ซึ่งหากนิยามในภาษาไทย คือ เครื่องหมายที่ใช้แทนวัตถุที่เป็นข้อมูลที่สามารถทำงานได้กับภาษา Java Script

ภาษา Java Script เป็นภาษาสำหรับการโปรแกรมบนเว็บเบราเซอร์ (Web Browser) ซึ่งเป็นภาษาประเภททำงานบนเครื่องลูกข่าย (Client-Side Programming) ซึ่งภาษา Java Script ช่วยให้นักพัฒนาสามารถโปรแกรมจัดการข้อมูลบนหน้าเว็บไซต์ได้อย่างสะดวกและมีประสิทธิภาพ

JSON เป็นโครงสร้างข้อมูลชนิดหนึ่งที่สามารถทำงานร่วมกับภาษา Java Script ได้อย่างมีประสิทธิภาพ ซึ่ง JSON เป็นโครงสร้างสำหรับการจัดเก็บข้อมูลและใช้ในการแลกเปลี่ยนข้อมูลผ่านเครือข่ายอินเทอร์เน็ตได้ อีกทั้ง JSON สามารถแปลงให้เป็นโครงสร้าง ของภาษา XML (eXtension Markup Language) ได้อย่างสะดวกรวดเร็ว

### 2.4.1 โครงสร้าง JSON มีข้อดีดังนี้

1) โครงสร้าง JSON ใช้อักขระอักษร (Text) ในการระบุโครงสร้างข้อมูล ดังนั้นสามารถทำการแก้ไขได้โดยง่าย
2) โครงสร้าง JSON สามารถใช้ในการแลกเปลี่ยนข้อมูลผ่านระบบเครือข่ายอินเทอร์เน็ตได้อย่างสะดวกรวดเร็ว เนื่องจากมีโครงสร้างที่ไม่ซับซ้อน หรือเรียกว่า เป็นรูปแบบโครงสร้างข้อมูลที่มีน้ำหนักเบา (Lightweight DataInterchange Format) กล่าวคือ เป็นโครงสร้างข้อมูลที่มีความกระชับของไวยากรณ์ (Syntax) สูง ซึ่งประกอบไปด้วยไวยากรณ์ที่จำเป็นเท่านั้น และเป็นสัญลักษณ์ที่มีจำนวนน้อย
3) โครงสร้าง JSON มีความเป็นอิสระของข้อมูล (Data-Independent) หมายความว่า โครงสร้าง JSON ไม่ยึดติดหรือไม่ขึ้นกับซอฟต์แวร์หรือระบบปฏิบัติการใดๆ รวมถึงไม่ขึ้นกับเครือข่ายหรืออุปกรณ์ด้วย ทำให้นักพัฒนาสามารถจัดการข้อมูลในโครงสร้าง JSON ได้ง่ายและไม่กระทบต่อระบบอื่นๆ เช่น สามารถแลกเปลี่ยนโครงสร้างข้อมูล JSON ระหว่างเครื่องคอมพิวเตอร์ตั้งโต๊ะกับโทรศัพท์สมาร์ตโฟน ได้
4) โครงสร้างของ JSON เป็นโครงสร้างที่สามารถอ่านและทำความเข้าใจได้ง่ายทั้งมนุษย์และโปรแกรมคอมพิวเตอร์ หรือที่เรียกว่า สามารถอธิบายข้อมูลของตัวเองได้ (Self-Describing) โดยผู้อ่านจะทราบว่า ข้อมูลประกอบไปด้วยเขตข้อมูล (Field) หรือคุณลักษณะ (Attribute) ใดบ้าง ข้อมูลมีจำนวนระเบียนเท่าใด เป็นต้น

**2.4.2 ไวยากรณ์เจสัน (JSON Syntax)**

1) ข้อมูลในโครงสร้าง JSON จะต้องกำหนดเป็นคู่ (Pairs) โดยจะต้องมีชื่อของข้อมูล (Name) และเนื้อของข้อมูล (Values) มาเป็นคู่กันเสมอ เช่น "firstName":"Nutthapat Ka" จากโครงสร้างนี้แปลได้ว่า Nutthapt Ka คือเนื้อข้อมูล (Values) ที่มีชื่อของเนื้อข้อมูล (Name หรือ Attribute) นี้ว่า firstName นั่นเองโดยที่เนื้อข้อมูลสามารถเป็นค่าต่างๆ ดังนี้

   ก. ค่าจำนวนตัวเลข อาจเป็นจำนวนเต็ม (Integer) หรือจำนวนทศนิยม (Floating Point)

   ข. สายอักขระ (String) หรือข้อมูลตัวอักษรจะต้องใส่ภายใต้ เครื่องหมาย Double Quote ( “ ” ) เสมอ

   ค. ค่าบูลีน (Boolean) หรือค่าข้อเท็จจริงสามารถกำหนดเป็น true หรือ false ได้

   ง. หากเนื้อข้อมูลมีลักษณะเป็นอาร์เรย์ (Arrays) ให้ใช้เครื่องหมาย Square brackets ( [ ] ) ครอบข้อมูลเหล่านั้น

   จ. แต่ละระเบียนหรือแต่ละวัตถุข้อมูล ให้ครอบด้วยเครื่องหมาย Curly braces ( { } )

   ฉ. ข้อควรระวัง โครงสร้าง JSON มีลักษณะเป็น Case-Sensitive อักษรตัวใหญ่-เล็กมีความแตกต่างกัน ดังนั้นการกำหนดชื่อของเนื้อข้อมูล (Name หรือ Attribute) มีผลต่อการเข้าถึงข้อมูล เช่น firstName จะเป็นคนละตัวกับ firstname

2) ข้อมูลแต่ละคู่ จะคั่นด้วยเครื่องหมาย Comma ( , )
3) เครื่องหมาย { } ใช้ในการครอบวัตถุข้อมูลหนึ่งๆหรือคั่นข้อมูลแต่ละระเบียน (Record)
4) เครื่องหมาย [ ] ใช้ในการสนับสนุนการทำงานแบบอาร์เรย์ (Arrays)

## 2.5 PainlessMesh

PainlessMesh เป็น Software Library ฟรีสำหรับ Arduino IDE ใช้ในการเขียนบอร์ด ESP32 และ ESP8266 ใช้เพื่อสร้างแลบควบคุม WIFI Mesh Network โดยแต่ละโหนดจะไม่ใช้ IP Address แต่จะสร้างรหัสประจำตัวเพื่อใช้ในการระบุตัวตนขึ้นมาเมื่อเปิดเครื่องแทน ขนาดของเครือข่ายที่รองรับขึ้นอยู่กับหน่วยความจำที่บันทึกโครงสร้างของเครือข่าย ยิ่งหน่วยความจำมากเท่าใหร่ยิ่งสามารถมีขนาดใหญ่ได้เท่านั้น

สามารถดูรายละเอียดและ Download ได้ที่ https://github.com/gmag11/painlessMesh

# บทที่ 3

# การวิเคราะห์ ออกแบบ และพัฒนาระบบ

## 3.1 ความต้องการของระบบ

### 3.1.1 Speccification

#### 3.1.1.1 บอร์ดESP32

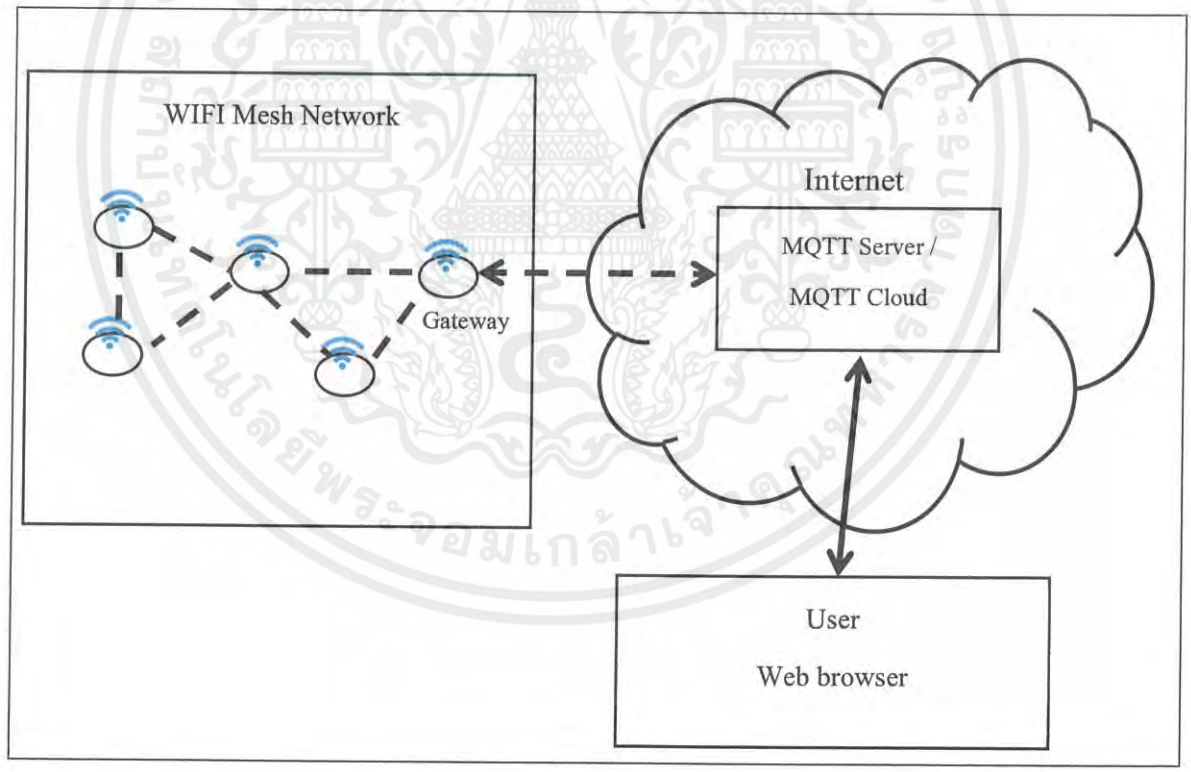
1) สามารถใช้ไลบรารี่ ที่เขียนด้วยภาษา C++ ใน Arduino IDE ได้

#### 3.1.1.2 Browser

1) สามารถอ่านไฟล์ภาษา HTMI

2) สามารถอ่านไฟล์ภาษา Java Scrip

## 3.2 โครงสร้างของระบบ




**รูปที่ 3.1 โครงสร้างโดยรวม**

จากรูปที่ 3.1 โครงสร้างโดยรวม อธิบายได้ดังนี้

ในส่วน WIFI Mesh Network เป็นเครือข่ายภายในซึ่งทำหน้าที่ควบคุม/รับค่าต่างๆภายในบ้าน จะมีโหนดที่เชื่อมต่อไปยัง Internet ซึ่งในโครงงานนี้จะขอเรียกว่าเรียกว่า Gateway Node โดย Gateway Node นี้จะรับข้อมูลคำสั่งจาก MQTT Server / MQTT Cloud และส่งคำสั่งที่ได้รับไปให้โหนดที่ถูกสั่งการ เมื่อโหนดนั้นๆทำตามคำสั่งที่ได้รับเสร็จแล้วหรือเมื่อถึงเวลาที่กำหนด จะส่งข้อมูลย้อนกลับไปเพื่อบอกว่าได้ทำอะไรผ่านทาง Gateway Node

สำหรับผู้ใช้งานที่ต้องการสั่งการระบบ จะสั่งการผ่าน Web Browser ซึ่ง web Browser จะสร้างแล้วส่งคำสั่งไปควบคุมระบบผ่านจาก MQTT Server / MQTT Cloud ไปยัง Gateway Node

ระบบเครือข่ายภายในเป็นระบบเครือข่ายที่สร้างขึ้นมาเพื่อประยุกต์ใช้ WIFI Mesh Network ในโครงงานนี้จะเรียกโหนดที่อ่านหรือแสดงค่าว่า I/O Node ซึ่ง I/O Node นี้จะออกแบบมาให้มี Input 2ช่อง และ Output 2 ช่อง แต่ละช่องจะมีชนิดการทำงาน 3 แบบ คือ แบบ Digital , แบบ Analog และแบบ I2C

## 3.3 โครงสร้างการรับส่งข้อมูลของเครือข่าย

ในเครือข่ายจะรับส่งข้อมูลกันในรูปแบบ JSON โดย ใช้ Topic ในการกำหนดการทำงานของข้อมูล

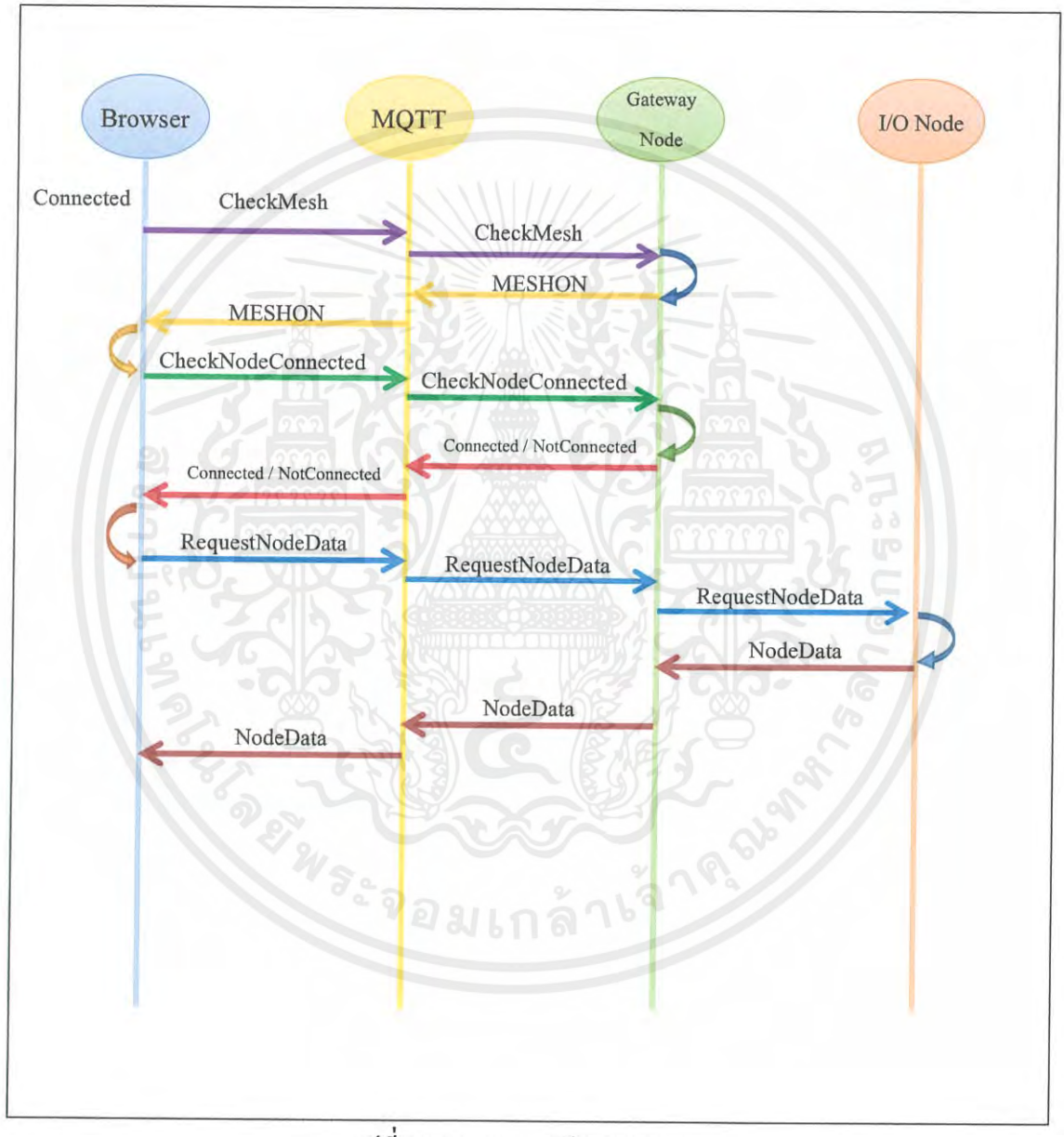
### 3.3.1 ชนิดของ Topic ที่ใช้

1) CheckMesh ใช้เพื่อตรวจสอบว่า WIFI Mesh Network ทำงานได้และเชื่อมต่อได้
2) MESHON ใช้เพื่อบอกว่า WIFI Mesh Network กำลังทำงานและเชื่อมต่อได้
3) CheckNodeConnected ใช้เพื่อตรวจสอบว่ามีโหนดใดเชื่อมต่อได้บ้าง
4) Connected ใช้เพื่อบอกว่าโหนดนั้นสามารถเชื่อมต่อได้
5) NotConnected ใช้เพื่อบอกว่าโหนดนั้นไม่สามารถเชื่อมต่อได้
6) Config ใช้เพื่อตั้งค่าโหนดต่างๆ
7) Control ใช้เพื่อควบคุมโหนดต่างๆ
8) RequestNodeData ใช้เพื่อขอข้อมูลของโหนดต่างๆ
9) SetGatewayID ใช้เพื่อบอกโหนดอื่นว่าโหนดนั้นเป็น Gateway Node
10) Register ใช้เพื่อบอก ID ของโหนดต่างๆ
11) NodeData ใช้เพื่อบอกข้อมูลของโหนดนั้น

12) UpdateData ใช้เพื่อบอกข้อมูลของโหนดนั้น

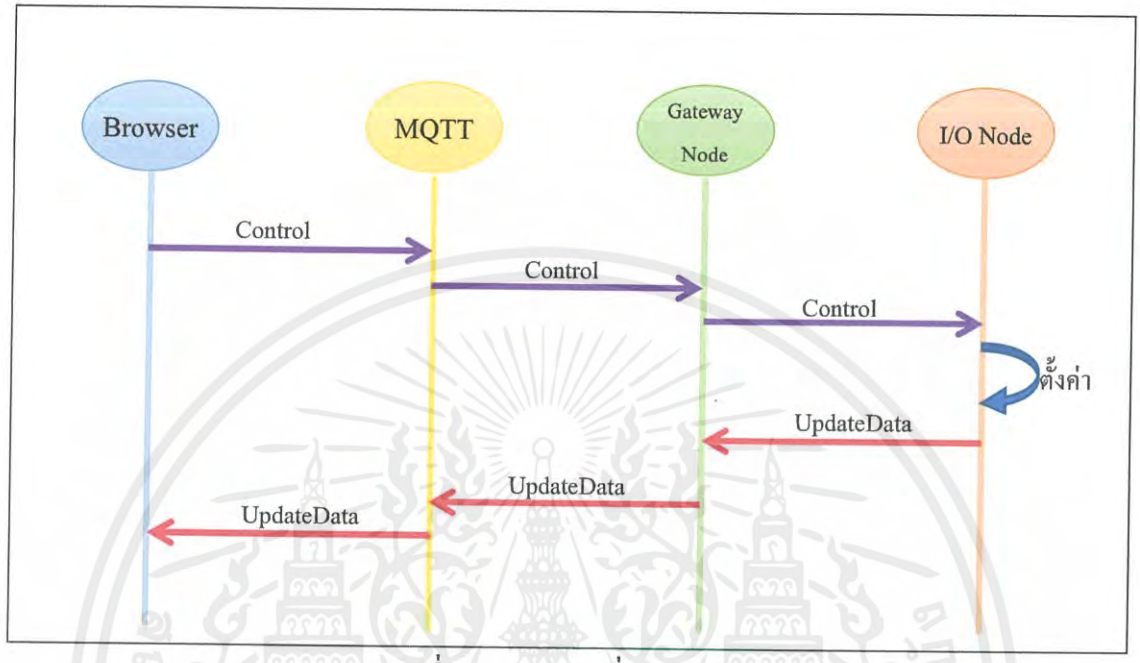
### 3.3.2 Dataflow ของข้อมูลในการทำงานต่างๆ

1) เปิด Browser



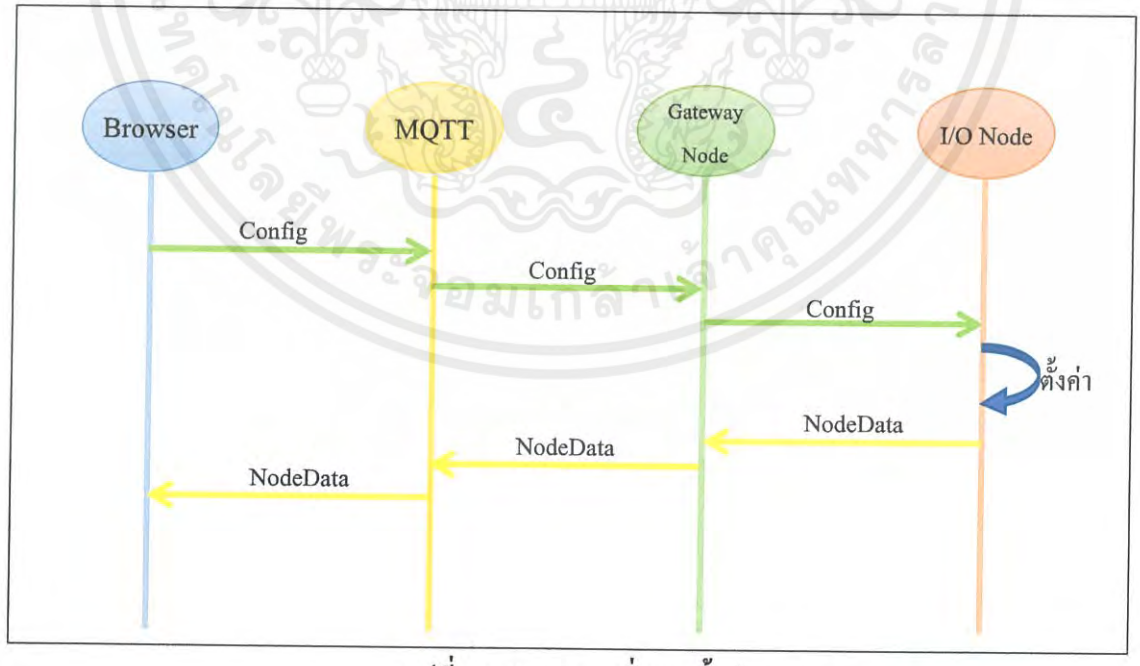

**รูปที่ 3.2 Dataflow เปิด Browser**

2) สั่งการควบคุมทาง Browser



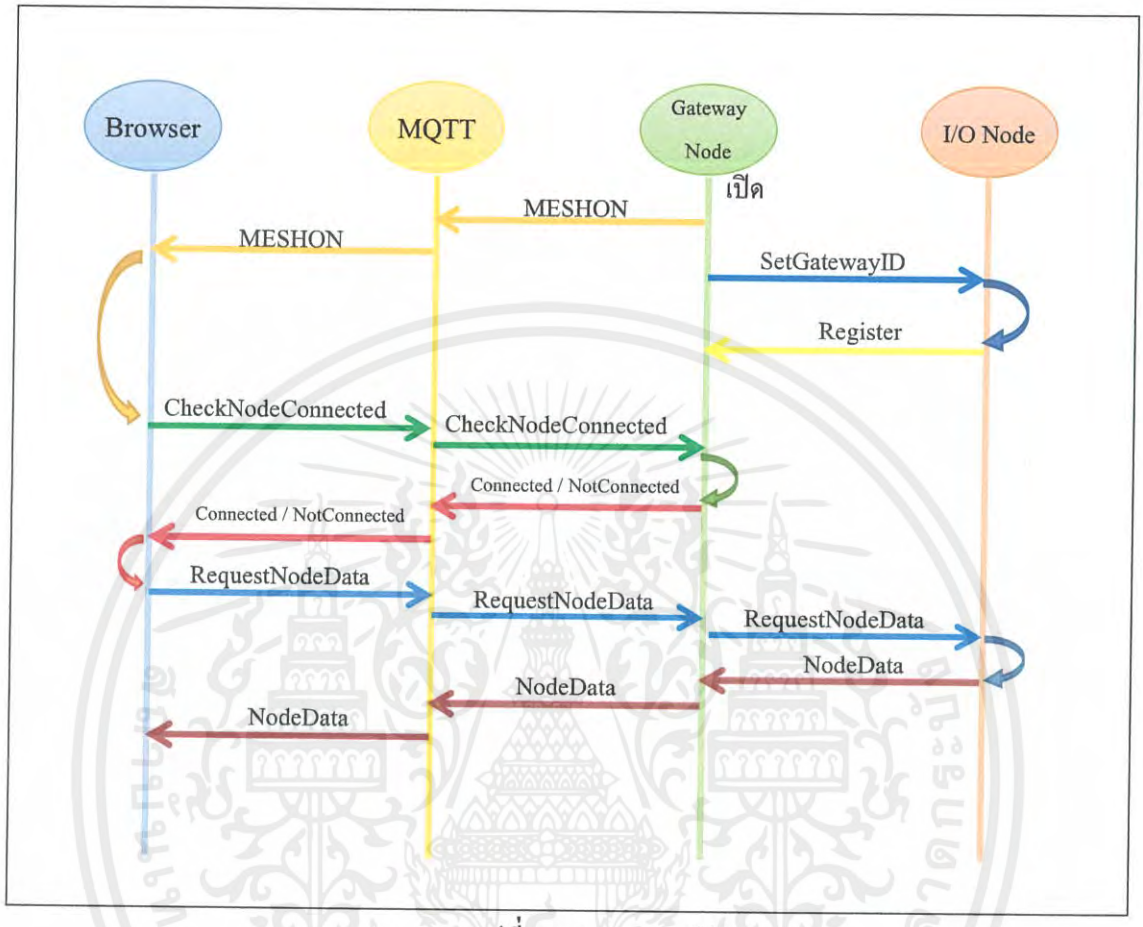

**รูปที่ 3.3 Dataflow สั่งการควบคุมทาง Browser**

3) สั่งการตั้งค่าทาง Browser



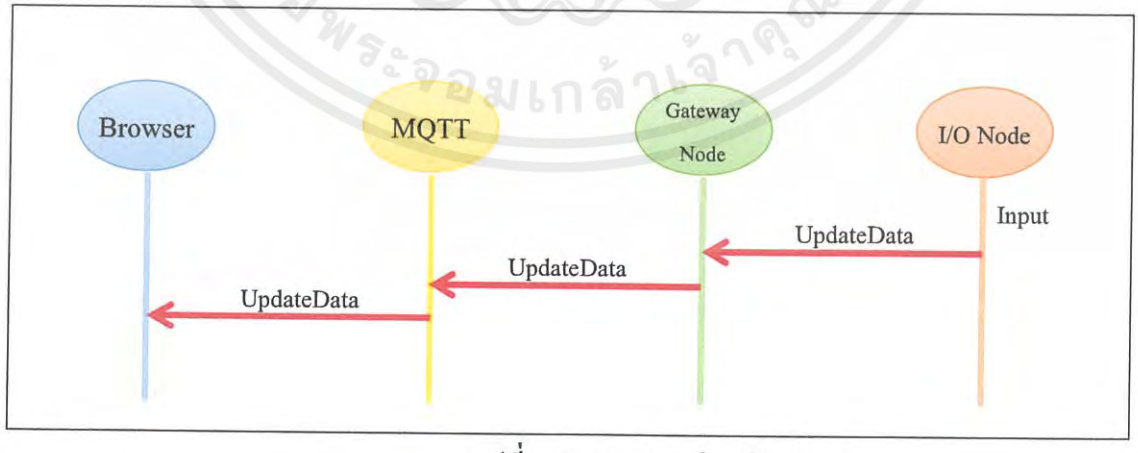

**รูปที่ 3.4 Dataflow สั่งการตั้งค่าทาง Browser**

4) เปิด Gateway Node




**รูปที่ 3.5 Dataflow เปิด Browser**

5) อ่านค่า Input




**รูปที่ 3.6 Dataflow อ่านค่า Input**

## 3.4 โครงสร้างโปรแกรมการทำงานในส่วนต่างๆ

### 3.4.1 โปรแกรมในส่วน I/O Node

#### 3.4.1.1 ส่วนตั้งค่าการทำงานและการเชื่อมต่อ

ตั้งค่าให้สร้าง WIFI Mesh Network กำหนดช่องทางในรับ/ส่งข้อมูล กำหนดค่าเริ่มต้นของ Node นั้นๆ โดยจะมี ชื่อเริ่มต้นของ Node หมายเลขเฉพาะของ Node นั้นๆ การอนุญาตใช้งาน Input/Output ทั้ง 2 ช่อง ชื่อของ Input/Output ช่องนั้นๆหากต้องการใช้ ชนิดของ Input/Output ช่องนั้นๆ หากใช้งาน ค่าเริ่มต้นของ Output แต่ละช่องหากใช้งาน และ address ของ Input/Output หากจำเป็นต้องใช้งาน

#### 3.4.1.2 ส่วนการทำงานเมื่อได้รับข้อความจาก Node ใดๆ

ข้อความจาก Node ใดๆใน WIFI Mesh Network จะมี Topic ติดมาด้วยเพื่อกำหนดการทำงาน Topic ที่ I/O Node จะได้รับมี Config Control RequestNodeData และ SetGatewayID

เมื่อได้รับข้อความ Topic Config จะทำการบันทึกและตั้งค่าตามข้อมูลที่ได้รับมา จากนั้นจะส่ง ข้อความที่มี Topic เป็น NodeData ไปที่ Gateway Node เพื่อบอกค่าข้อมูลที่เปลี่ยนไป

เมื่อได้รับข้อความ Topic Control จะทำการแปลี่ยนค่า Output ตามที่ข้อมูลได้รับมาและบันทึกข้อมูล จากนั้นจะส่งข้อความที่มี Topic เป็น UpdateData ไปที่ Gateway Node เพื่อบอกค่าข้อมูลที่เปลี่ยนไป

เมื่อได้รับข้อความ Topic RequestNodeData จะทำการส่งข้อความที่มี Topic เป็น NodeData เพื่อส่งข้อมูลทั้งหมดของโหนดนั้นๆ กลับไปยังโหนดที่ส่งข้อความมา

เมื่อได้รับข้อความ Topic SetGatewayID จะทำการบันทึก ID ของ Gateway Node ตามที่ได้รับมา จากนั้นจะทำการส่งข้อความที่มี Topic เป็น Register ไปที่ Gateway Node เพื่อบอกให้รู้ว่า ID ของโหนดนั้นๆคืออะไร

#### 3.4.1.3 ส่วนที่ทำงานทุกๆรอบเวลาหนึ่ง

จะมีการอ่านข้อมูลจาก Input แล้วส่งข้อความที่มี Topic เป็น UpdateData ไปที่ Gateway Node เพื่อบอกค่าข้อมูลที่อ่านได้

**3.4.2 โปรแกรมในส่วน Gateway Node**

**3.4.2.1 ส่วนตั้งค่าการทำงานและการเชื่อมต่อ**

ตั้งค่าให้สร้าง WIFI Mesh Network , ตั้งค่าการเชื่อมต่อ internet WIFI และ MQTT จากนั้นส่งข้อความที่มี Topic เป็น MESHON ไปทาง MQTT เพื่อบอกว่า WIFI Mesh Network กำลังใช้งานอยู่

**3.4.2.2 ส่วนการทำงานเมื่อได้รับข้อความจาก MQTT**

ข้อความจาก MQTT จะมี Topic ติดมาด้วยเพื่อกำหนดการทำงาน Topic ที่ Gateway Node จะได้รับมี CheckMesh , CheckNodeConnected , Config , Control และ RequestNodeData

เมื่อได้รับข้อความ Topic CheckMesh จะทำการส่งข้อความที่มี Topic เป็น MESHON ไปที่ MQTT เพื่อบอกให้รู้ว่า WIFI Mesh Network กำลังทำงานและเชื่อมต่ออยู่

เมื่อได้รับข้อความ Topic CheckNodeConnected จะทำการตรวจสอบว่าขณะนี้มีโหนดใดสามารถติดต่อได้บ้าง หากติดต่อได้จะส่งข้อความที่มี Topic เป็น Connected ไปที่ MQTT เพื่อบอกให้รู้ว่าโหนดนั้นสามารถเชื่อมต่อได้ แต่หากไม่สามารถเชื่อมต่อได้จะส่งข้อความที่มี Topic เป็น NotConnected ไปที่ MQTT เพื่อบอกให้รู้ว่าโหนดนั้นไม่สามารถเชื่อมต่อได้

เมื่อได้รับข้อความ Topic Config จะตรวจสอบว่าคำสั่งนี้ต้องการตั้งค่าโหนดใดจากนั้นทำการส่งข้อความที่ได้รับไปที่โหนดนั้น

เมื่อได้รับข้อความ Topic Control จะตรวจสอบว่าคำสั่งนี้ต้องการควบคุมโหนดใดจากนั้นทำการส่งข้อความที่ได้รับไปที่โหนดนั้น

เมื่อได้รับข้อความ Topic RequestNodeData จะตรวจสอบว่าคำสั่งนี้ต้องการข้อมูลของโหนดใดจากนั้นทำการส่งข้อความที่ได้รับไปที่โหนดนั้น

**3.4.2.3 ส่วนการทำงานเมื่อได้รับข้อความจาก Node อื่นๆ**

ข้อความจาก Node อื่นๆใน WIFI mesh network จะมี Topic ติดมาด้วยเพื่อกำหนดการทำงาน Topic ที่ Gateway Node จะได้รับมี NodeData , Register และ UpdateData

เมื่อได้รับข้อความ Topic Register จะทำการบันทึก ID ของโหนดที่ส่งมาเพื่อเอาไว้ใช้ในการส่งข้อความที่ได้รับไปยังโหนดนั้น

เมื่อได้รับข้อความ Topic NodeData จะส่งต่อข้อความที่ได้รับไปทาง MQTT

เมื่อได้รับข้อความ Topic UpdateData จะส่งต่อข้อความที่ได้รับไปทาง MQTT

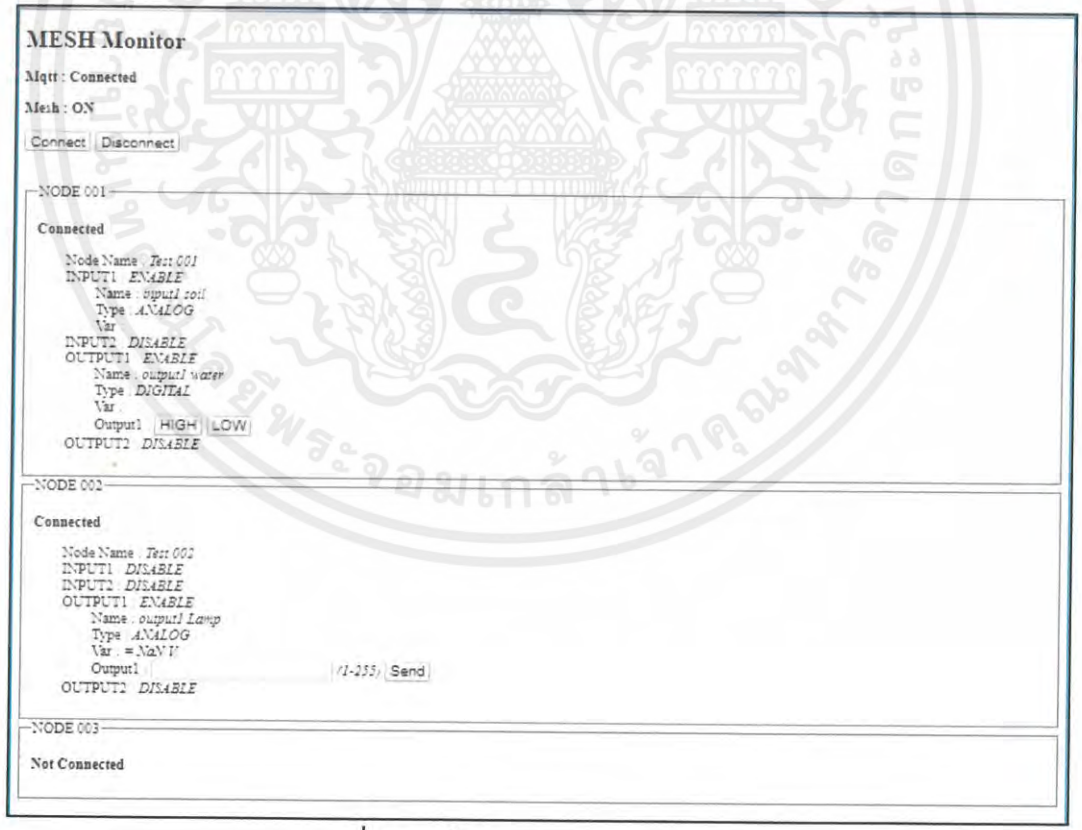
#### 3.4.2.4 ส่วนที่ทำงานทุกๆรอบเวลาหนึ่ง

ส่งข้อความที่มี Topic เป็น SetGatewayID ไปยังทุกโหนดเพื่อบอกว่าโหนดนี้เป็น Gateway Node

### 3.4.3 โปรแกรมในส่วน Web Browser

#### 3.4.3.1 หน้า Monitoring and Control

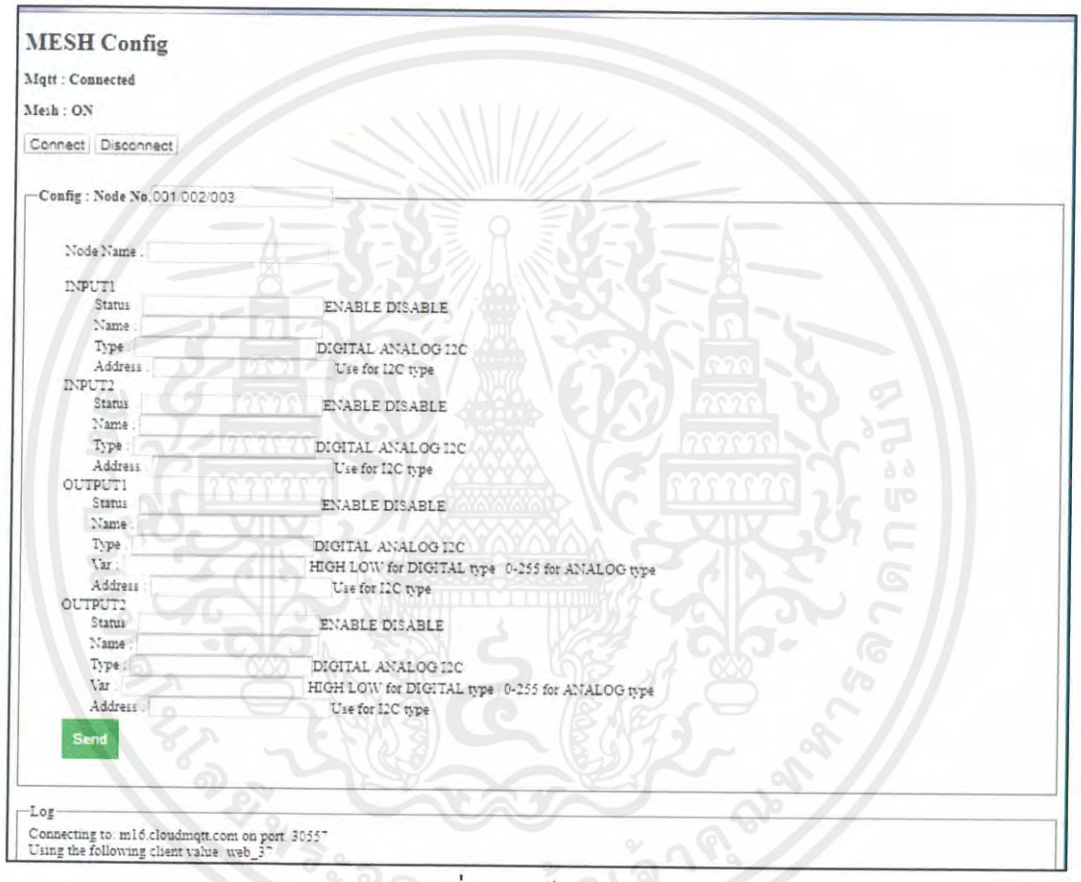
แสดงค่าสถานะของ mesh network , ค่าของโหนดต่างๆตามที่ได้รับมา และ ส่งข้อมูลที่มี Topic เป็น Control ผ่าน MQTT ไปที่ mesh network เพื่อควบคุม output ของ I/O Node ที่อยู่ใน mesh network




รูปที่ 3.7 หน้า Monitoring and Control

### 3.4.3.2 หน้า Config

แสดงค่าสถานะของ mesh network และ ส่งข้อมูลที่มี Topic เป็น Config ผ่าน MQTT ไปที่ mesh network เพื่อตั้งค่า I/O Node ที่อยู่ใน mesh network




รูปที่ 3.8 หน้า Config

# บทที่ 4

# การทดลอง

การบดลองในบทนี้จะเป็นการทดลองใช้เครื่องมือและอุปกรณ์ต่างๆ เช่นการเขียนโปรแกรมควบคุม ESP32 และการตั้งค่าต่างๆ ดังนี้

1) การทดลองสร้าง WIFI Mesh Network
2) การทดลองใช้งาน MQTT cloud ผ่าน Gateway Node
3) การทดลองใช้งาน MQTT cloud ควบคุมระบบ WIFI Mesh Network

## 4.1 การทดลองสร้าง WIFI Mesh Network

### 4.1.1 วัตถุประสงค์

เพื่อทดลองสร้าง WIFI Mesh Network และทดสอบการส่งข้อมูลภายในเครือข่าย

### 4.1.2 วิธีการทดลอง

เขียนโปรแกรมควบคุม ESP32 จำนวนไม่ต่ำกว่า 2 ตัว ทำการสร้างสร้าง WIFI Mesh Network โดยใช้ไลบรารี่ painlessMesh ผ่าน Arduino IDE ทำการทดลองส่งข้อมูลระหว่างโหนดต่างๆ เพื่อทดสอบว่าสามารถส่งข้อมูลหากันได้หรือไม่

### 4.1.3 ขั้นตอนการทดลอง

1) ติดตั้ง Arduino IDE
2) ดาวน์โหลดข้อมูลสำหรับเขียนบอร์ดทดลอง ESP32 บน Arduino IDE โดย
    1. เลือกแท็บ File -> เลือก Preferences จะมีหน้าต่าง Preferences ขึ้นมา
    2. ในหน้าต่าง Preferences ให้พิมพ์ https://dl.espressif.com/dl/package_esp32_index.json ใส่เข้าไปในช่อง Addditional Boards Manager URLs:
    3. เลือกแท็บ Tools -> เลือก Board: -> เลือก Boards Manager... จะมีหน้าต่าง Boards Manager ขึ้นมา
    4. ในหน้าต่าง Boards Manager พิมพ์ esp32 ในช่อง *Filter your search...*
    5. ติดตั้ง esp32 by Espressif Systems

3) ดาวน์โหลดและติดตั้ง Library "painlessMesh.h" โดย

1. ไปที่ https://github.com/gmag11/painlessMesh จากนั้น คลิกที่ปุ่ม Clone or download -> คลิก Download ZIP
2. ติดตั้ง library โดยเลือกแท็บ Sketch -> เลือก include Library -> เลือก Add .ZIP Library... -> ค้นหาและเลือก painlessMesh-master.zip -> คลิกที่ปุ่ม Open

4) เขียนโปรแกรมใน Arduino IDE ตามโปรแกรมที่ 4.1 ด้านล่าง

**โปรแกรมที่ 4.1 hello Node**

```
#include "painlessMesh.h"

//Mesh setting
#define   MESH_PREFIX     "whateverYouLike"
#define   MESH_PASSWORD   "somethingSneaky"
#define   MESH_PORT       5555
//Node default
#define   NODENAME        "NodeLED1"

Scheduler     userScheduler; // to control your personal task
painlessMesh  mesh;

// Functions/Prototype
void receivedCallback( uint32_t from, String &msg );

// Send message every 1 seconds
Task myLoggingTask(TASK_SECOND * 10, TASK_FOREVER, []() {
      DynamicJsonBuffer jsonBuffer;
      JsonObject& msg = jsonBuffer.createObject();
      String str;
      msg["topic"] = "HELLO";
      msg["MyName"] = NODENAME;
      msg["fromnode"] = mesh.getNodeId();

      msg.printTo(str);
      mesh.sendBroadcast(str);
      Serial.print("send Brodcast : ");
      Serial.println(str);
});
```

**โปรแกรมที่ 4.1 hello Node (ต่อ)**

```
void setup() {
  Serial.begin(115200);

  // setup mesh
  mesh.setDebugMsgTypes( ERROR | STARTUP | MESH_STATUS |
CONNECTION );

  mesh.init( MESH_PREFIX, MESH_PASSWORD, &userScheduler,
MESH_PORT );

  mesh.onReceive([] ( uint32_t from, String &msg ) {
    Serial.printf("Received from %u msg=%s\n", from,
msg.c_str());
  });
  mesh.onNewConnection([](size_t nodeId) {
    Serial.printf("New Connection, nodeId = %u\n", nodeId);
  });
  mesh.onChangedConnections([]() {
    Serial.printf("Changed connection\n");
  });
  mesh.onDroppedConnection([](size_t nodeId) {
    Serial.printf("Dropped Connection, nodeId = %u\n",
nodeId);
  });

  // Add the task to the your scheduler
  userScheduler.addTask(myLoggingTask);
  myLoggingTask.enable();
}
void loop() {
  mesh.update();
  userScheduler.execute();

}
```

5) เสียบสายเชื่อมต่อกับบอร์ดทดลอง ESP32

6) เลือกแท็บ Tools -> เลือก Board: -> เลือกชื่อบอร์ดทดลอง ESP32 ที่ใช้ทดลอง (ผู้จัดทำใช้ NodeMCU-32S)

7) เลือกแท็บ Tools -> เลือก Port -> เลือก COM Port ของบอร์ดทดลอง ESP32 ที่ใช้ทดลอง

8) คลิกปุ่ม upload เพื่อลงโปรแกรมบนบอร์ด

9) เสียบสายเชื่อมต่อกับบอร์ดทดลอง ESP32 อีกบอร์ด

10) ทำซ้ำข้อ 6) -7)

11) เปลี่ยน “NodeLED1” ในโปรแกรมเป็นชื่ออื่น

12) คลิกปุ่ม upload เพื่อลงโปรแกรมบนบอร์ด

13) เลือกแท็บ Tools -> เลือก Serial Monitor

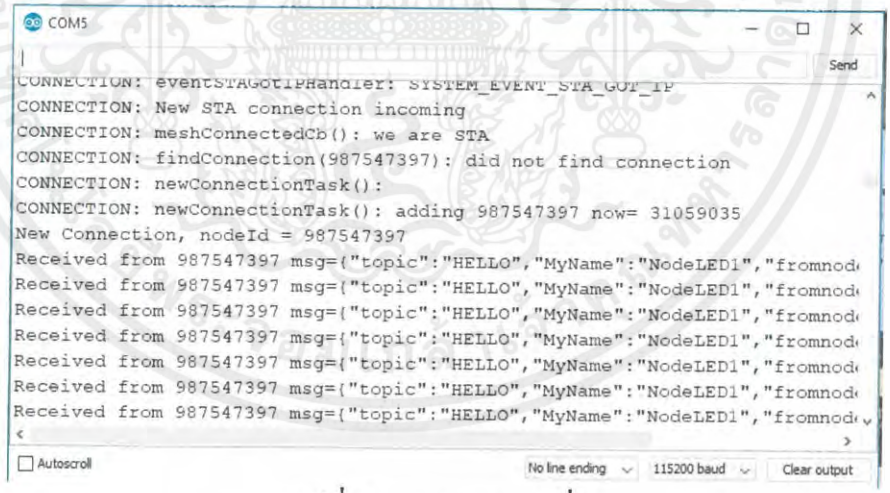
14) ดูข้อความใน Serial Monitor หากได้รับปรากฏข้อความ Received from _____

msg={"Topic":"HELLO","MyName":"NodeLED1","fromNode":_____}

แสดงว่าสามารถส่งข้อความหากันได้

### 4.1.4 ผลการทดลอง

ผลที่ได้จะดูผ่าน Serial Monitor ของ Arduino IDE แสดงการรับส่งข้อมูลระหว่างโหนดในการทดลองจะรับส่งข้อมูลในรูปแบบ JSON




**รูปที่ 4.1 ผลการทดลองที่ 1**

จากการทดลองสามารถรับส่งข้อมูลระหว่างโหนดได้

## 4.2 การทดลองใช้งาน MQTT cloud ผ่าน Gateway Node

### 4.2.1 วัตถุประสงค์

เพื่อทดลองการรับส่งข้อมูลผ่าน MQTT cloud สำหรับการควบคุม ESP32

### 4.2.2 วิธีการทดลอง

เขียนโปรแกรมควบคุม ESP32 รับส่งข้อมูลผ่าน MQTT cloud โดยทำการอ่านข้อมูลผ่าน MQTT และทำตามคำสั่งที่อ่านได้

### 4.2.3 ขั้นตอนการทดลอง

1) ติดตั้ง Arduino IDE
2) ดาวน์โหลดข้อมูลสำหรับเขียนบอร์ดทดลอง ESP32 บน Arduino IDE โดย
    1. เลือกแท็บ File -> เลือก Preferences จะมีหน้าต่าง Preferences ขึ้นมา
    2. ในหน้าต่าง Preferences ให้พิมพ์ https://dl.espressif.com/dl/package_ esp32 _index.json ใส่เข้าไปในช่อง Addditional Boards Manager URLs:
    3. เลือกแท็บ Tools -> เลือก Board: -> เลือก Boards Manager... จะมีหน้าต่าง Boards Manager ขึ้นมา
    4. ในหน้าต่าง Boards Manager พิมพ์ esp32 ในช่อง *Filter your search...*
    5. ติดตั้ง esp32 by Espressif Systems
3) ดาวน์โหลดและตัดตั้ง Library "painlessMesh.h" โดย
    1. ไปที่ https://github.com/gmag11/painlessMesh จากนั้น คลิกที่ปุ่ม Clone or download -> คลิก Download ZIP
    2. ติดตั้ง library โดยเลือกแท็บ Sketch -> เลือก include Library -> เลือก Add .ZIP Library... -> ค้นหาและเลือก painlessMesh-master.zip -> คลิกที่ปุ่ม Open
4) ดาวน์โหลดและตัดตั้ง Library อื่นๆ โดย
    1. เลือกแท็บ Sketch -> เลือก include Library -> เลือก Manage Libraries... จะมีหน้าต่าง Library Manager ขึ้นมา

2. ในหน้าต่าง Library Manager ให้ค้นหาและติดตั้ง Library PubSubClient.h และ WIFI.h

5) สมัครใช้งาน MQTT cloud ที่ www.cloudMQTT.com

6) ตั้งค่าการใช้งาน MQTT cloud

1. คลิกที่ปุ่ม Create New Instance
2. ตั้งชื่อ ในช่อง Name แล้ว คลิกปุ่ม Select Region
3. เลือก Data center แล้วคลิกปุ่ม Review
4. คลิกปุ่ม Create Instance
5. เลือกไปที่ Instance ที่พึ่งสร้างเสร็จ
6. เลือกแท็บ DETAILS จำ/จดค่า Server , User ,Password , Port และ Websockets Port ไว้
7. เลือกแท็บ USERS & ACL ให้พิมพ์ /MESH ลงไปในช่อง Pattern เลือก √ ในช่องหน้า Read Access? และช่องหน้า Write Access? จากนั้น คลิกปุ่ม Add

7) เขียนโปรแกรมใน Arduino IDE ตามโปรแกรมที่ 4.2 ด้านล่าง โดยแทนค่าต่างๆ ด้วยค่าที่ผู้ทดลองใช้งาน

**โปรแกรมที่ 4.2 Board LED**

```
#include <WIFI.h>
#include <PubSubClient.h>
#include "painlessMesh.h"

#define   LED          (LED pin บนบอร์ดของคุณ)
#define WIFI_SSID      "(ssid ของ WIFI ที่คุณใช้)"
#define WIFI_PASSWORD "(password ของ WIFI ที่คุณใช้)"
// MQTT
#define MQTT_server    "( Server ของ MQTT ที่คุณจำไว้)"
#define MQTT_port      (Port ของ MQTT ที่คุณจำไว้)
#define MQTT_user      "(User ของ MQTT ที่คุณจำไว้)"
#define MQTT_password "(Password ของ MQTT ที่คุณจำไว้)"
```

**โปรแกรมที่ 4.2 Board LED (ต่อ)**

```
WiFiClient espClient;
PubSubClient client(espClient);

long lastReconnectAttempt = 0;

// Prototype & Function
void mqttCallback(char* topic, byte* payload, unsigned int
length);
boolean reconnect();
char* string2char(String);
const char* wl_status_to_string(wl_status_t);
void wifiCheck();
void wifiConnect(){
  Serial.print("Connecting to ");
  Serial.println(WIFI_SSID);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Wifi connecting");
  int count = 1;
  while (WiFi.status() != WL_CONNECTED) {
    if(count<5){
      Serial.print(".");
      count++;
    }else{
      Serial.println(".");
      WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
      count=1;
    }
    delay(500);
  }
  Serial.println("\nWifi connected");
  Serial.print("WiFi.macAddress() = ");
  Serial.println(WiFi.macAddress());
  Serial.print("WiFi.localIP()    = ");
  Serial.println(WiFi.localIP());
  Serial.print("WiFi.subnetMask() = ");
  Serial.println(WiFi.subnetMask());
  Serial.print("WiFi.gatewayIP()  = ");
  Serial.println(WiFi.gatewayIP());
}
```

**โปรแกรมที่ 4.2 Board LED (ต่อ)**

```
void setup() {
  Serial.begin(115200);
  pinMode(LED, OUTPUT);
  wifiConnect();
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(mqttCallback);
  delay(1000);
}
void loop() {
  wifiCheck();
  if (!client.connected()) {
    //client.setServer(mqtt_server, mqtt_port);
    //client.setCallback(mqttCallback);
    long now = millis();
    if (now - lastReconnectAttempt > 5000) {
      lastReconnectAttempt = now;
      // Attempt to reconnect
      if (reconnect()) {
        lastReconnectAttempt = 0;
      }
    }
  } else {
    client.loop();
  }
  client.subscribe("/MESH");
}

void mqttCallback(char* topic, byte* payload, unsigned int
length) {
  Serial.print("-----Mqtt Message arrived [");
  Serial.print(topic);
  Serial.print("] ");

  String msg = "";
  int i=0;
  while (i<length) msg += (char)payload[i++];
  Serial.println(msg);
  DynamicJsonBuffer jsonBuffer;
  JsonObject& root = jsonBuffer.parseObject(msg);
  if (root.containsKey("topic")) {
    if (String("CONTROL").equals(root["topic"].as<String>()))
{
```

**โปรแกรมที่ 4.2 Board LED (ต่อ)**

```
if (root.containsKey("LED1")){
        DynamicJsonBuffer jsonBuffer;
        JsonObject& msg2 = jsonBuffer.createObject();
        msg2["topic"] = "UPDATE";
        if (String("ON").equals(root["LED1"].as<String>())) {
          digitalWrite(LED, HIGH);
          msg2["LED1"] = "ON";
        } else if
(String("OFF").equals(root["LED1"].as<String>())) {
          digitalWrite(LED, LOW);
          msg2["LED1"] = "OFF";
        }
        String str;
        msg2.printTo(str);
        mqttSend(str);
      }
    }
  }
}

boolean reconnect(){
  wifiCheck();
  Serial.print("Reconnect ");
  client.setServer(mqtt_server, mqtt_port);
    if (!client.connected()) {
      Serial.print("Attempting MQTT connection...");
      // Attempt to connect
      if (client.connect("ESPClient", mqtt_user,
mqtt_password)) {
        Serial.println("connected");
        client.subscribe("/MESH");
      } else {
        Serial.print("failed, rc=");
        Serial.print(client.state());
      }
    }
  return client.connected();
}
```

**โปรแกรมที่ 4.2 Board LED (ต่อ)**

```
char* string2char(String command){
    if(command.length()!=0){
        char *p = const_cast<char*>(command.c_str());
        return p;
    }
}
void mqttSend(String str){
  Serial.println("Send : " + str);
   do{
      wifiCheck();
      if (!client.connected()) {
        reconnect();
      }
      Serial.print("Attempting MQTT connection...");
      // Attempt to connect
      if (client.connect("ESP8266Client", mqtt_user,
mqtt_password)) {
       client.publish("/MESH",string2char(str) );
       Serial.println("\tpublish : " + str);
      } else {
       Serial.print("failed, rc=");
       Serial.println(client.state());
       delay(100);
      }
      client.subscribe("/MESH");
  } while (!client.connected());
  Serial.println("Send Finished");
}

const char* wl_status_to_string(wl_status_t status) {
  switch (status) {
    case WL_NO_SHIELD: return "WL_NO_SHIELD";
    case WL_IDLE_STATUS: return "WL_IDLE_STATUS";
    case WL_NO_SSID_AVAIL: return "WL_NO_SSID_AVAIL";
    case WL_SCAN_COMPLETED: return "WL_SCAN_COMPLETED";
    case WL_CONNECTED: return "WL_CONNECTED";
    case WL_CONNECT_FAILED: return "WL_CONNECT_FAILED";
    case WL_CONNECTION_LOST: return "WL_CONNECTION_LOST";
    case WL_DISCONNECTED: return "WL_DISCONNECTED";
  }
}
```

**โปรแกรมที่ 4.2 Board LED (ต่อ)**

```
void wifiCheck(){
  if(WiFi.status() != WL_CONNECTED){
    Serial.print("WiFi Not Connected : ");
    Serial.print("WiFi.status = ");
    Serial.println(wl_status_to_string(WiFi.status()));
    wifiConnect();
  }
}
```

8) เสียบสายเชื่อมต่อกับบอร์ดทดลอง ESP32

9) เลือกแท็บ Tools -> เลือก Board: -> เลือกชื่อบอร์ดทดลอง ESP32 ที่ใช้ทดลอง (ผู้จัดทำใช้ NodeMCU-32S)

10) เลือกแท็บ Tools -> เลือก Port -> เลือก COM Port ของบอร์ดทดลอง ESP32 ที่ใช้ทดลอง

11) คลิกปุ่ม upload เพื่อลงโปรแกรมบนบอร์ด

12) สร้างไฟล์นามสกุล .html แล้วเขียนโปรแกรมลงไป ตามโปรแกรมที่ 4.3 ในหน้าถัดไป โดยแทนค่าต่างๆด้วยค่าที่ผู้ทดลองใช้งาน

**โปรแกรมที่ 4.3 LED Controller**

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8" />
<title>CONTROLER</title>
<script src="https://cdnjs.cloudflare.com/ajax/libs/paho-
MQTT/1.0.2/MQTTws31.min.js" type="text/javascript"></script>
<script>
    var config = {
      MQTT_server: "(Server ของ MQTT ที่คุณจำไว้)",
      MQTT_websockets_port: (Websockets Port ของ MQTT ที่คุณจำไว้),
      MQTT_user: "(Server ของ MQTT ที่คุณจำไว้)",
      MQTT_password: "(Server ของ MQTT ที่คุณจำไว้)" };
    function startConnect() {
      document.getElementById("status").innerHTML =
"Connecting...";
      // Generate a random client ID
      clientID = "web_" + parseInt(Math.random() * 100, 10);

      // Fetch the hostname/IP address and port number from
the form
      host = config.mqtt_server;
      port = config.mqtt_websockets_port;

      // Print output for the user in the messages div
      document.getElementById("messages").innerHTML +=
'<span>Connecting to: ' + host + ' on port: ' + port +
'</span><br/>';
      document.getElementById("messages").innerHTML +=
'<span>Using the following client value: ' + clientID +
'</span><br/>';

      // Initialize new Paho client connection
      client = new Paho.MQTT.Client(host, Number(port),
clientID);

      // Set callback handlers
      client.onConnectionLost = onConnectionLost;
      client.onMessageArrived = onMessageArrived;
      // Connect the client, if successful, call onConnect
function
```

**โปรแกรมที่ 4.3 LED Controller (ต่อ)**

```
      client.connect({
        useSSL: true,
        userName: config.mqtt_user,
        password: config.mqtt_password,
        timeout: 3,
        onSuccess: onConnect,
        onFailure: onFail
      });
    }
    // Called when the client connects
    function onConnect() {
      console.log("onConnect");
      // Fetch the MQTT topic from the form
      topic = "/MESH";
      // Print output for the user in the messages div
      document.getElementById("messages").innerHTML +=
'<span>Subscribing to: ' + topic + '</span><br/>';
      document.getElementById("status").innerHTML =
"Connected";
      var messageSend = { topic: "CheckMesh" };
      meshSend(JSON.stringify(messageSend));
      // Subscribe to the requested topic
      client.subscribe(topic);
    }

    function onFail(event) {
      document.getElementById("status").innerHTML = "Error : "
+ event;
    }
    // Called when the client loses its connection
    function onConnectionLost(responseObject) {
      console.log("onConnectionLost: Connection Lost");
      if (responseObject.errorCode !== 0) {
        console.log("onConnectionLost: " +
responseObject.errorMessage);
        document.getElementById("status").innerHTML =
"ConnectionLost: " + responseObject.errorMessage;
      }
    }
```

**โปรแกรมที่ 4.3 LED Controller (ต่อ)**

```
    // Called when a message arrives
    function onMessageArrived(message) {
      console.log("onMessageArrived: " +
message.payloadString);
      document.getElementById("messages").innerHTML +=
'<span>Topic: ' + message.destinationName + '  | ' +
message.payloadString + '</span><br/>';
      updateScroll(); // Scroll to bottom of window
      var inMessage = message.payloadString;
      var messageObj = JSON.parse(inMessage);
      if (messageObj.hasOwnProperty("topic")) {
        if (messageObj.topic == "UPDATE") {
          if (messageObj.hasOwnProperty("LED1")) {
            document.getElementById("LED1_id").innerHTML =
messageObj.LED1;
          }
          if (messageObj.hasOwnProperty("LED2")) {
            document.getElementById("LED2_id").innerHTML =
messageObj.LED2;
          }
        }
      }
    }

    // Called when the disconnection button is pressed
    function startDisconnect() {
      display();
      client.disconnect();
      document.getElementById("status").innerHTML =
"Disconnected";
      document.getElementById("messages").innerHTML +=
'<span>Disconnected</span><br/>';
      updateScroll(); // Scroll to bottom of window
    }

    // Updates #messages div to auto-scroll
    function updateScroll() {
      var element = document.getElementById("messages");
      element.scrollTop = element.scrollHeight;
    }
```

**โปรแกรมที่ 4.3 LED Controller (ต่อ)**

```
    function meshSend(msg) {
      var message = new Paho.MQTT.Message(msg);
      message.destinationName = "/MESH";
      client.send(message);
      client.subscribe("/MESH");
      document.getElementById("messages").innerHTML +=
'<span>Mesh Send: ' + msg + '</span><br/>';
      updateScroll(); // Scroll to bottom of window
    }

    function led1(n) {
      if (n == 0) {
        var l = {
          topic: "CONTROL" ,
          LED1: "OFF"              };
        meshSend(JSON.stringify(l));
      } else if (n == 1) {
        var l = {
          topic: "CONTROL",
          LED1: "ON"               };
        meshSend(JSON.stringify(l));
      }
    }
</script>
</head>
<body>
    <h1>mqtt LED Controller</h1>
    <h3>Mqtt : <span id="status">Click Connect</span></h3>
    <input type="button" onclick="startConnect()"
value="Connect">
    <input type="button" onclick="startDisconnect()"
value="Disconnect">
    <br /><br />
    <h3>LED1 : <span id="LED1_id">---</span></h3>
    <input type="button" onclick="led1(1)" value="ON">
    <input type="button" onclick="led1(0)" value="OFF">
    <br /><br />
    <fieldset>
        <legend>Log</legend>
        <div id="messages"></div>
    </fieldset>
</body>
</html>
```

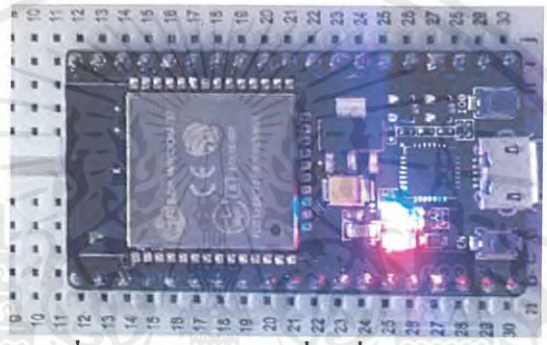
13) บันทึกไฟล์ แล้วเปิดไฟล์ที่เขียนเสร็จบน Web Browser

14) คลิกปุ่ม Connect

15) ทดลองกดปุ่ม ON / OFF เพื่อเปิดหรือปิดไฟ LED บนบอร์ดทดลอง

### 4.2.4 ผลการทดลอง

ผลที่ได้จะดูที่ LED ของบอร์ดทดลอง ESP32 (ผู้จัดทำใช้ LED สีฟ้า)



**รูปที่ 4.2 ผลการทดลองที่ 2 เมื่อคลิดปุ่ม ON**

**รูปที่ 4.3 ผลการทดลองที่ 2 เมื่อคลิดปุ่ม OFF**

จากการทดลองเมื่อ ESP32 รับข้อมูลคำสั่งจาก MQTT ผ่าน www.cloudMQTT.com จะทำการเปิด/ปิด LED ตามคำสั่งที่ได้รับ

## 4.3 การทดลองใช้งาน MQTT cloud ควบคุมระบบ WIFI Mesh Network

### 4.3.1 วัตถุประสงค์

เพื่อทดลองการสร้าง WIFI Mesh Network และควบคุมผ่าน MQTT

### 4.3.2 วิธีการทดลอง

เขียนโปรแกรมควบคุม ESP32 สร้าง WIFI Mesh Network รับส่งข้อมูลผ่าน MQTT cloud โดยทำการอ่านข้อมูลผ่าน MQTT และส่งคำสั่งที่อ่านได้ไปควบคุมโหนดอื่น

### 4.3.3 ขั้นตอนการทดลอง

1) ติดตั้ง Arduino IDE
2) ดาวน์โหลดข้อมูลสำหรับเขียนบอร์ดทดลอง ESP32 บน Arduino IDE โดย
    1. เลือกแท็บ File -> เลือก Preferences จะมีหน้าต่าง Preferences ขึ้นมา
    2. ในหน้าต่าง Preferences ให้พิมพ์ https://dl.espressif.com/dl/package_ esp32 _index.json ใส่เข้าไปในช่อง Addditional Boards Manager URLs:
    3. เลือกแท็บ Tools -> เลือก Board: -> เลือก Boards Manager... จะมีหน้าต่าง Boards Manager ขึ้นมา
    4. ในหน้าต่าง Boards Manager พิมพ์ esp32 ในช่อง *Filter your search*...
    5. ติดตั้ง esp32 by Espressif Systems
3) ดาวน์โหลดและตัดตั้ง Library "painlessMesh.h" โดย
    1. ไปที่ https://github.com/gmag11/painlessMesh จากนั้น คลิกที่ปุ่ม Clone or download -> คลิก Download ZIP
    2. ติดตั้ง library โดยเลือกแท็บ Sketch -> เลือก include Library -> เลือก Add .ZIP Library… -> ค้นหาและเลือก painlessMesh-master.zip -> คลิกที่ปุ่ม Open
4) ดาวน์โหลดและตัดตั้ง Library อื่นๆโดย
    1. เลือกแท็บ Sketch -> เลือก include Library -> เลือก Manage Libraries… จะมีหน้าต่าง Library Manager ขึ้นมา

2. ในหน้าต่าง Library Manager ให้ค้นหาและติดตั้ง Library PubSubClient.h และ WIFI.h

5) สมัครใช้งาน MQTT cloud ที่ www.cloudMQTT.com

6) ตั้งค่าการใช้งาน MQTT cloud

1. คลิกที่ปุ่ม Create New Instance
2. ตั้งชื่อ ในช่อง Name แล้ว คลิกปุ่ม Select Region
3. เลือก Data center แล้วคลิกปุ่ม Review
4. คลิกปุ่ม Create Instance
5. เลือกไปที่ Instance ที่พึ่งสร้างเสร็จ
6. เลือกแท็บ DETAILS จำ/จดค่า Server , User ,Password , Port และ Websockets Port ไว้
7. เลือกแท็บ USERS & ACL ให้พิมพ์ /MESH ลงไปในช่อง Pattern เลือก √ ในช่องหน้า Read Access? และช่องหน้า Write Access? จากนั้น คลิกปุ่ม Add

7) เขียนโปรแกรมใน Arduino IDE ตามโปรแกรมที่ 4.4 ด้านล่าง โดยแทนค่าต่างๆ ด้วยค่าที่ผู้ทดลองใช้งาน

**โปรแกรมที่ 4.4 Gateway LED**

```
#include <WIFI.h>
#include <PubSubClient.h>
#include "painlessMesh.h"

#define   LED          (LED pin บนบอร์ดของคุณ)
#define WIFI_SSID      "(ssid ของ WIFI ที่คุณใช้)"
#define WIFI_PASSWORD "(password ของ WIFI ที่คุณใช้)"
// MQTT
#define MQTT_server    "( Server ของ MQTT ที่คุณจำไว้)"
#define MQTT_port      (Port ของ MQTT ที่คุณจำไว้)
#define MQTT_user      "(User ของ MQTT ที่คุณจำไว้)"
#define MQTT_password "(Password ของ MQTT ที่คุณจำไว้)"
```

**โปรแกรมที่ 4.4 Gateway LED (ต่อ)**

```
//mesh
#define   MESH_PREFIX     "whateverYouLike"
#define   MESH_PASSWORD   "somethingSneaky"
#define   MESH_PORT       5555
#define   NODENAME        "NodeLED1"

Scheduler     userScheduler; // to control your personal task
painlessMesh  mesh;
WiFiClient    espClient;
PubSubClient  client(espClient);

long lastReconnectAttempt = 0;

// Prototype
void mqttCallback(char* topic, byte* payload, unsigned int
length);
void receivedCallback( uint32_t from, String &msg );
void changedConnectionCallback();
SimpleList<uint32_t> nodes;
// Function
boolean reconnect();
char* string2char(String);
void mqttSend(String);
const char* wl_status_to_string(wl_status_t);
void wifiCheck();
void wifiConnect(){
  Serial.print("Connecting to ");
  Serial.println(WIFI_SSID);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Wifi connecting");
  int count = 1;
  while (WiFi.status() != WL_CONNECTED) {
    if(count<5){
      Serial.print(".");
      count++;
    }else{
      Serial.println(".");
      WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
      count=1;
    }
    delay(500);
  }
  Serial.println("\nWifi connected");
```

**โปรแกรมที่ 4.4 Gateway LED (ต่อ)**

```
  Serial.print("WiFi.macAddress() = ");
  Serial.println(WiFi.macAddress());
  Serial.print("WiFi.localIP()    = ");
  Serial.println(WiFi.localIP());
  Serial.print("WiFi.subnetMask() = ");
  Serial.println(WiFi.subnetMask());
  Serial.print("WiFi.gatewayIP()  = ");
  Serial.println(WiFi.gatewayIP());
}

Task logServerTask((TASK_SECOND * 10), TASK_FOREVER, []() {
      DynamicJsonBuffer jsonBuffer;
      JsonObject& msg = jsonBuffer.createObject();
      String str;
      msg["topic"] = "HELLO";
      msg["MyName"] = NODENAME;
      msg["fromnode"] = mesh.getNodeId();

      msg.printTo(str);
      mesh.sendBroadcast(str);
      Serial.print("send Brodcast : ");
      Serial.println(str);
});

void setup() {
  Serial.begin(115200);
  pinMode(LED, OUTPUT);

  //Mesh setting
  mesh.setDebugMsgTypes( ERROR | STARTUP | MESH_STATUS |
CONNECTION );

  mesh.init( MESH_PREFIX, MESH_PASSWORD, &userScheduler,
MESH_PORT );

  mesh.onReceive(&receivedCallback);

  mesh.onNewConnection([](size_t nodeId) {
    Serial.printf("***** New Connection, nodeId = %u\n",
nodeId);
  });
  mesh.onChangedConnections(&changedConnectionCallback);
```

**โปรแกรมที่ 4.4 Gateway LED (ต่อ)**

```
  mesh.onDroppedConnection([](size_t nodeId) {
    Serial.printf("***** Dropped Connection, nodeId = %u\n",
nodeId);
  });
  mesh.onNodeTimeAdjusted([](int32_t offset) {
    Serial.printf("***** Adjusted time %u. Offset = %d\n",
mesh.getNodeTime(),offset);
  });

  // connect to wifi.
  wifiConnect();

  //Set mqtt
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(mqttCallback);

  // Add the task to the your scheduler
  userScheduler.addTask(logServerTask);
  logServerTask.enable();
}

void loop() {
  wifiCheck();
  if (!client.connected()) {
    //client.setServer(mqtt_server, mqtt_port);
    //client.setCallback(mqttCallback);
    long now = millis();
    if (now - lastReconnectAttempt > 5000) {
      lastReconnectAttempt = now;
      // Attempt to reconnect
      if (reconnect()) {
        lastReconnectAttempt = 0;
      }
    }
  } else {
    client.loop();
  }
  client.subscribe("/MESH");
  mesh.update();
  userScheduler.execute(); // it will run mesh scheduler as
well
}
```

**โปรแกรมที่ 4.4 Gateway LED (ต่อ)**

```
void receivedCallback( uint32_t from, String &msg ) {
  Serial.printf("Gateway : Received from %u msg=%s\n", from,
msg.c_str());
  DynamicJsonBuffer jsonBuffer;
  JsonObject& root = jsonBuffer.parseObject(msg);
  if (root.containsKey("topic")) {
      if( String("UPDATE").equals( root["topic"].as<String>()
)){
        Serial.print("\tReceive : ");
        Serial.println( root["topic"].as<String>() );
        root.remove("fromnode");
        String str;
        root.printTo(str);
        mqttSend(str);
      }
  }
}

void mqttCallback(char* topic, byte* payload, unsigned int
length) {
  Serial.print("-----Mqtt Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  String msg = "";
  int i=0;
  while (i<length) msg += (char)payload[i++];
  Serial.println(msg);
  DynamicJsonBuffer jsonBuffer;
  JsonObject& root = jsonBuffer.parseObject(msg);
  if (root.containsKey("topic")) {
    if (String("CONTROL").equals(root["topic"].as<String>()))
{
      if (root.containsKey("LED1")){
        DynamicJsonBuffer jsonBuffer;
        JsonObject& msg2 = jsonBuffer.createObject();
        msg2["topic"] = "UPDATE";
        if (String("ON").equals(root["LED1"].as<String>())) {
          digitalWrite(LED, HIGH);
          msg2["LED1"] = "ON";
        } else if
(String("OFF").equals(root["LED1"].as<String>())) {
          digitalWrite(LED, LOW);
          msg2["LED1"] = "OFF";
```

**โปรแกรมที่ 4.4 Gateway LED (ต่อ)**

```
        }
        String str;
        msg2.printTo(str);
        mqttSend(str);
      }
      if (root.containsKey("LED2")){
        DynamicJsonBuffer jsonBuffer;
        JsonObject& msg3 = jsonBuffer.parseObject(msg);
        root["fromnode"] = mesh.getNodeId();
        String str;
        msg3.printTo(str);
        mesh.sendBroadcast(str);
      }
    }
  }
}

void changedConnectionCallback() {
  Serial.printf("Changed connections %s\n",
mesh.subConnectionJson().c_str());
  nodes = mesh.getNodeList();

  Serial.printf("Num nodes: %d\n", nodes.size());
  Serial.printf("Connection list:");
  SimpleList<uint32_t>::iterator node = nodes.begin();
  while (node != nodes.end()) {
    Serial.printf(" %u", *node);
    node++;
  }
  Serial.println();
}

boolean reconnect(){
  wifiCheck();
  Serial.print("Reconnect ");
  client.setServer(mqtt_server, mqtt_port);
    if (!client.connected()) {
      Serial.print("Attempting MQTT connection...");
      // Attempt to connect
      if (client.connect("ESPClient", mqtt_user,
mqtt_password)) {
        Serial.println("connected");
        client.subscribe("/MESH");
```

**โปรแกรมที่ 4.4 Gateway LED (ต่อ)**

```
      } else {
        Serial.print("failed, rc=");
        Serial.print(client.state());
      }
    }
  return client.connected();
}

char* string2char(String command){
    if(command.length()!=0){
        char *p = const_cast<char*>(command.c_str());
        return p;
    }
}

void mqttSend(String str){
  Serial.println("Send : " + str);
   do{
      wifiCheck();
      if (!client.connected()) {
        reconnect();
      }
      Serial.print("Attempting MQTT connection...");
      // Attempt to connect
      if (client.connect("ESP8266Client", mqtt_user,
mqtt_password)) {
        client.publish("/MESH",string2char(str) );
        Serial.println("\tpublish : " + str);
      } else {
        Serial.print("failed, rc=");
        Serial.println(client.state());
        delay(100);
      }
      client.subscribe("/MESH");
  } while (!client.connected());
  Serial.println("Send Finished");
}
```

**โปรแกรมที่ 4.4 Gateway LED (ต่อ)**

```
void wifiCheck(){
  if(WiFi.status() != WL_CONNECTED){
    Serial.print("WiFi Not Connected : ");
    Serial.print("WiFi.status = ");
    Serial.println(wl_status_to_string(WiFi.status()));
    wifiConnect();
  }
}

const char* wl_status_to_string(wl_status_t status) {
  switch (status) {
    case WL_NO_SHIELD: return "WL_NO_SHIELD";
    case WL_IDLE_STATUS: return "WL_IDLE_STATUS";
    case WL_NO_SSID_AVAIL: return "WL_NO_SSID_AVAIL";
    case WL_SCAN_COMPLETED: return "WL_SCAN_COMPLETED";
    case WL_CONNECTED: return "WL_CONNECTED";
    case WL_CONNECT_FAILED: return "WL_CONNECT_FAILED";
    case WL_CONNECTION_LOST: return "WL_CONNECTION_LOST";
    case WL_DISCONNECTED: return "WL_DISCONNECTED";
  }
}
```

8) เสียบสายเชื่อมต่อกับบอร์ดทดลอง ESP32

9) เลือกแท็บ Tools -> เลือก Board: -> เลือกชื่อบอร์ดทดลอง ESP32 ที่ใช้ทดลอง (ผู้จัดทำใช้ NodeMCU-32S)

10) เลือกแท็บ Tools -> เลือก Port -> เลือก COM Port ของบอร์ดทดลอง ESP32 ที่ใช้ทดลอง

11) คลิกปุ่ม upload เพื่อลงโปรแกรมบนบอร์ด

12) เขียนโปรแกรมใน Arduino IDE ตามโปรแกรมที่ 4.5 ในหน้าถัดไป โดยแทนค่าต่างๆด้วยค่าที่ผู้ทดลองใช้งาน

**โปรแกรมที่ 4.5 Mesh LED**

```
#include "painlessMesh.h"

#define   LED             (LED pin บนบอร์ดของคุณ)
#define   MESH_PREFIX     "whateverYouLike"
#define   MESH_PASSWORD   "somethingSneaky"
#define   MESH_PORT       5555
#define   NODENAME        "NodeLED2"

Scheduler     userScheduler; // to control your personal task
painlessMesh  mesh;

// Prototype
void receivedCallback( uint32_t from, String &msg );
void changedConnectionCallback();
SimpleList<uint32_t> nodes;

Task logServerTask((TASK_SECOND * 10), TASK_FOREVER, []() {
      DynamicJsonBuffer jsonBuffer;
      JsonObject& msg = jsonBuffer.createObject();
      String str;
      msg["topic"] = "HELLO";
      msg["MyName"] = NODENAME;
      msg["fromnode"] = mesh.getNodeId();

      msg.printTo(str);
      mesh.sendBroadcast(str);
      Serial.print("send Brodcast : ");
      Serial.println(str);
});

void setup() {
  Serial.begin(115200);
  pinMode(LED, OUTPUT);

  //Mesh setting
  mesh.setDebugMsgTypes( ERROR | STARTUP | MESH_STATUS |
CONNECTION );

  mesh.init( MESH_PREFIX, MESH_PASSWORD, &userScheduler,
MESH_PORT );

  mesh.onReceive(&receivedCallback);
```

**โปรแกรมที่ 4.5 Mesh LED (ต่อ)**

```
  mesh.onNewConnection([](size_t nodeId) {
    Serial.printf("***** New Connection, nodeId = %u\n",
nodeId);
  });
  mesh.onChangedConnections(&changedConnectionCallback);
  mesh.onDroppedConnection([](size_t nodeId) {
    Serial.printf("***** Dropped Connection, nodeId = %u\n",
nodeId);
  });
  mesh.onNodeTimeAdjusted([](int32_t offset) {
    Serial.printf("***** Adjusted time %u. Offset = %d\n",
mesh.getNodeTime(),offset);
  });

  // Add the task to the your scheduler
  userScheduler.addTask(logServerTask);
  logServerTask.enable();
}

void loop() {
  mesh.update();
  userScheduler.execute(); // it will run mesh scheduler as
well
}

void receivedCallback( uint32_t from, String &msg ) {
  Serial.printf("Gateway : Received from %u msg=%s\n", from,
msg.c_str());
  DynamicJsonBuffer jsonBuffer;
  JsonObject& root = jsonBuffer.parseObject(msg);
  if (root.containsKey("topic")) {
    if (String("CONTROL").equals(root["topic"].as<String>()))
{
      if (root.containsKey("LED2")){
        DynamicJsonBuffer jsonBuffer;
        JsonObject& msg2 = jsonBuffer.createObject();
        msg2["topic"] = "UPDATE";
        msg2["fromnode"] = mesh.getNodeId();
        if (String("ON").equals(root["LED2"].as<String>())) {
          digitalWrite(LED, HIGH);
          msg2["LED2"] = "ON";
        } else if
(String("OFF").equals(root["LED2"].as<String>())) {
```

**โปรแกรมที่ 4.5 Mesh LED (ต่อ)**

```
            digitalWrite(LED, LOW);
            msg2["LED2"] = "OFF";
          }
          String str;
          msg2.printTo(str);
          mesh.sendBroadcast(str);
          Serial.print("send Brodcast : ");
          Serial.println(str);
        }
      }
    }
}

void changedConnectionCallback() {
  Serial.printf("Changed connections %s\n",
mesh.subConnectionJson().c_str());
  nodes = mesh.getNodeList();

  Serial.printf("Num nodes: %d\n", nodes.size());
  Serial.printf("Connection list:");

  SimpleList<uint32_t>::iterator node = nodes.begin();
  while (node != nodes.end()) {
    Serial.printf(" %u", *node);
    node++;
  }
  Serial.println();
}
```

13) เสียบสายเชื่อมต่อกับบอร์ดทดลอง ESP32 อีกบอร์ด

14) ทำซ้ำข้อ 9) -11)

15) สร้างไฟล์นามสกุล .html แล้วเขียนโปรแกรมลงไป ตามโปรแกรมที่ 4.6 ในหน้าถัดไป โดยแทนค่าต่างๆด้วยค่าที่ผู้ทดลองใช้งาน

**โปรแกรมที่ 4.6 LED Mesh Controller**

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8" />
<title>CONTROLER</title>
<script src="https://cdnjs.cloudflare.com/ajax/libs/paho-
MQTT/1.0.2/MQTTws31.min.js" type="text/javascript"></script>
<script>
    var config = {
      MQTT_server: "( Server ของ MQTT ที่คุณจำไว้)",
      MQTT_websockets_port: (Websockets Port ของ MQTT ที่คุณจำไว้),
      MQTT_user: "( Server ของ MQTT ที่คุณจำไว้)",
      MQTT_password: "( Server ของ MQTT ที่คุณจำไว้)"
    };

    function startConnect() {
      document.getElementById("status").innerHTML =
"Connecting...";
      // Generate a random client ID
      clientID = "web_" + parseInt(Math.random() * 100, 10);

      // Fetch the hostname/IP address and port number from
the form
      host = config.mqtt_server;
      port = config.mqtt_websockets_port;

      // Print output for the user in the messages div
      document.getElementById("messages").innerHTML +=
'<span>Connecting to: ' + host + ' on port: ' + port +
'</span><br/>';
      document.getElementById("messages").innerHTML +=
'<span>Using the following client value: ' + clientID +
'</span><br/>';

      // Initialize new Paho client connection
      client = new Paho.MQTT.Client(host, Number(port),
clientID);

      // Set callback handlers
      client.onConnectionLost = onConnectionLost;
      client.onMessageArrived = onMessageArrived;
```

**โปรแกรมที่ 4.6 LED Mesh Controller (ต่อ)**

```
      // Connect the client, if successful, call onConnect
function
      client.connect({
        useSSL: true,
        userName: config.mqtt_user,
        password: config.mqtt_password,
        timeout: 3,
        onSuccess: onConnect,
        onFailure: onFail          });
    }

    // Called when the client connects
    function onConnect() {
      console.log("onConnect");

      // Fetch the MQTT topic from the form
      topic = "/MESH";

      // Print output for the user in the messages div
      document.getElementById("messages").innerHTML +=
'<span>Subscribing to: ' + topic + '</span><br/>';
      document.getElementById("status").innerHTML =
"Connected";

      var messageSend = { topic: "CheckMesh" };
      meshSend(JSON.stringify(messageSend));

      // Subscribe to the requested topic
      client.subscribe(topic);
    }

    function onFail(event) {
      document.getElementById("status").innerHTML = "Error : "
+ event;
    }

    // Called when the client loses its connection
    function onConnectionLost(responseObject) {
      console.log("onConnectionLost: Connection Lost");
      if (responseObject.errorCode !== 0) {
        console.log("onConnectionLost: " +
responseObject.errorMessage);
        document.getElementById("status").innerHTML =
"ConnectionLost: " + responseObject.errorMessage;
      }
    }
```

**โปรแกรมที่ 4.6 LED Mesh Controller (ต่อ)**

```
    // Called when a message arrives
    function onMessageArrived(message) {
      console.log("onMessageArrived: " +
message.payloadString);
      document.getElementById("messages").innerHTML +=
'<span>Topic: ' + message.destinationName + '  | ' +
message.payloadString + '</span><br/>';
      updateScroll(); // Scroll to bottom of window

      var inMessage = message.payloadString;
      var messageObj = JSON.parse(inMessage);

      if (messageObj.hasOwnProperty("topic")) {
        if (messageObj.topic == "UPDATE") {
          if (messageObj.hasOwnProperty("LED1")) {
            document.getElementById("LED1_id").innerHTML =
messageObj.LED1;
          }
          if (messageObj.hasOwnProperty("LED2")) {
            document.getElementById("LED2_id").innerHTML =
messageObj.LED2;
          }
        }
      }
    }

    // Called when the disconnection button is pressed
    function startDisconnect() {
      display();
      client.disconnect();
      document.getElementById("status").innerHTML =
"Disconnected";
      document.getElementById("messages").innerHTML +=
'<span>Disconnected</span><br/>';
      updateScroll(); // Scroll to bottom of window
    }

    // Updates #messages div to auto-scroll
    function updateScroll() {
      var element = document.getElementById("messages");
      element.scrollTop = element.scrollHeight;
    }
```

**โปรแกรมที่ 4.6 LED Mesh Controller (ต่อ)**

```
    function meshSend(msg) {
      var message = new Paho.MQTT.Message(msg);
      message.destinationName = "/MESH";
      client.send(message);

      client.subscribe("/MESH");

      document.getElementById("messages").innerHTML +=
'<span>Mesh Send: ' + msg + '</span><br/>';
      updateScroll(); // Scroll to bottom of window
    }

    function led1(n) {
      if (n == 0) {
        var l = {
          topic: "CONTROL" ,
          LED1: "OFF"      };
        meshSend(JSON.stringify(l));
      } else if (n == 1) {
        var l = {
          topic: "CONTROL",
          LED1: "ON"       };
        meshSend(JSON.stringify(l));
      }
    }
    function led2(n) {
      if (n == 0) {
        var l = {
          topic: "CONTROL",
          LED2: "OFF"      };
        meshSend(JSON.stringify(l));
      } else if (n == 1) {
        var l = {
          topic: "CONTROL",
          LED2: "ON"       };
        meshSend(JSON.stringify(l));
      }
    }
</script>
</head>
<body>
    <h1>mqtt LED Controller</h1>

    <h3>Mqtt : <span id="status">Click Connect</span></h3>
    <input type="button" onclick="startConnect()"
value="Connect">
    <input type="button" onclick="startDisconnect()"
value="Disconnect">
    <br /><br />
```

**โปรแกรมที่ 4.6 LED Mesh Controller (ต่อ)**

```
    <h3>LED1 : <span id="LED1_id">---</span></h3>
    <input type="button" onclick="led1(1)" value="ON">
    <input type="button" onclick="led1(0)" value="OFF">
    <br /><br />

    <h3>LED2 : <span id="LED2_id">---</span></h3>
    <input type="button" onclick="led2(1)" value="ON">
    <input type="button" onclick="led2(0)" value="OFF">
    <br /><br />
    <fieldset>
        <legend>Log</legend>
        <div id="messages"></div>
    </fieldset>
</body>
</html>
```

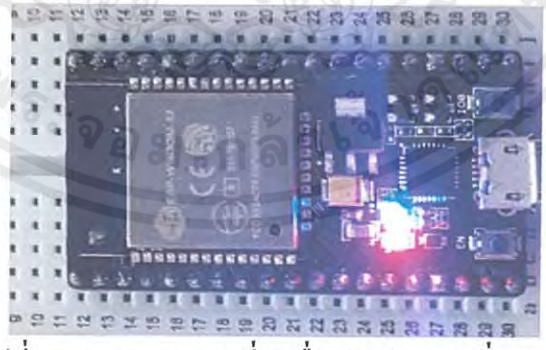
16) บันทึกไฟล์ แล้วเปิดไฟล์ที่เขียนเสร็จบน Web Browser

17) คลิกปุ่ม Connect

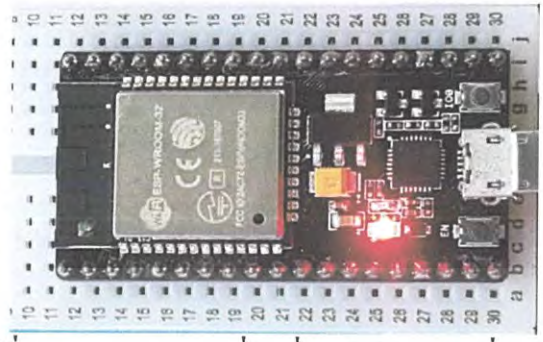
18) ทดลองกดปุ่ม ON / OFF เพื่อเปิดหรือปิดไฟ LED บนบอร์ดทดลอง

### 4.3.4 ผลการทดลอง

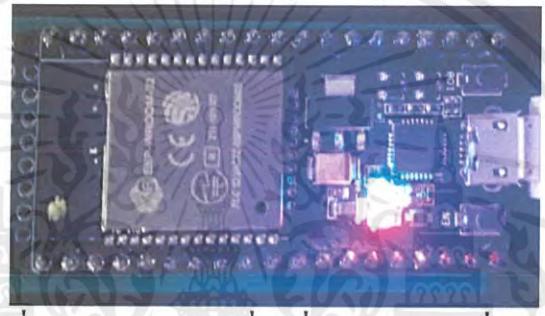
ผลที่ได้จะดูที่ LED (ผู้จัดทำใช้ LED สีฟ้า)ของบอร์ดทดลอง ESP32 ทั้ง 2 บอร์ด



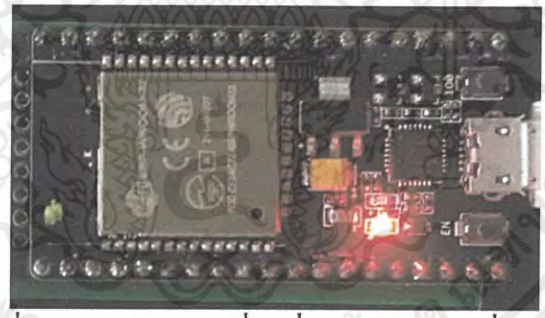
**รูปที่ 4.4 ผลการทดลองที่ 3 เมื่อคลิดปุ่ม ON ที่ LED1**



**รูปที่ 4.5 ผลการทดลองที่ 3 เมื่อคลิดปุ่ม OFF ที่ LED1**



**รูปที่ 4.6 ผลการทดลองที่ 3 เมื่อคลิดปุ่ม ON ที่ LED2**



**รูปที่ 4.7 ผลการทดลองที่ 3 เมื่อคลิดปุ่ม OFF ที่ LED2**

จากการทดลองเมื่อ ESP32 รับข้อมูลคำสั่งจาก MQTT ผ่าน www.cloudMQTT.com จะทำการเปิด/ปิด LED ทั้ง 2 ตัวได้

# บทที่ 5
# บทสรุปและข้อเสนอแนะ

## 5.1 บทสรุปของโครงงาน

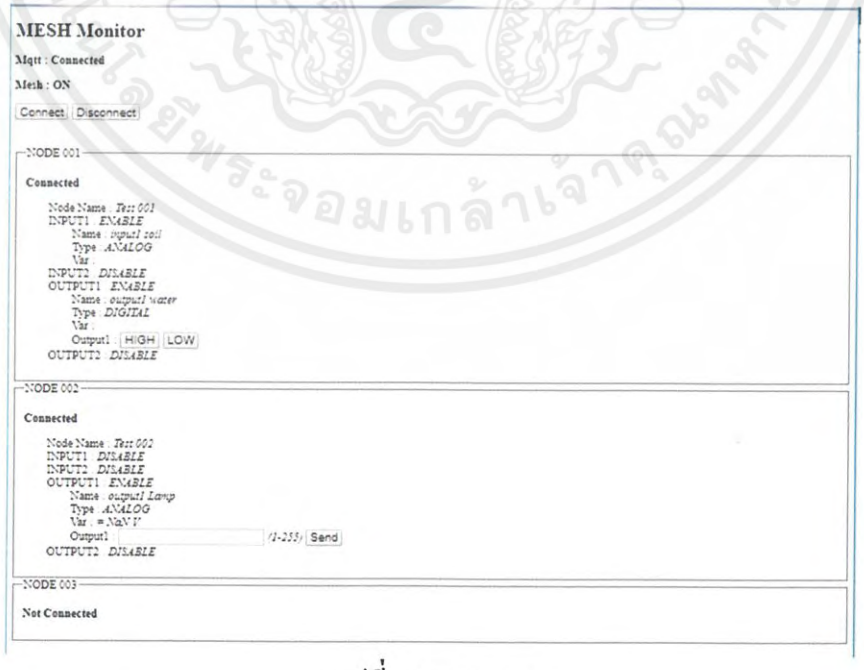
ภายในโครงงานนี้ประกอบไปด้วย 3 ส่วน คือ I/O Node , Gateway Node และ Web browser ซึ่งทำงานร่วมกัน และรับส่งข้อมูลให้กัน โดยระหว่าง Gateway Node และ Web browser มี MQTT เป็นตัวกลางในการรับส่งข้อมูล ทว่าด้วยปัญหาด้านปริมาณการรับส่งข้อมูลของ MQTT โครงสร้างที่ออกแบบไว้ไม่สามารถรองรับการทำงานสำหรับ Topic Config ได้

### 5.1.1 ตัวอย่างชิ้นงาน

**รูปที่ 5.1 Demo I/O Node 001**

รูปที่ 5.2 Demo I/O Node 002&003


รูปที่ 5.3 Demo Gateway Node




รูปที่ 5.4 Brower

## 5.2 ปัญหาและอุปสรรค

### 5.2.1 ส่วน I/O Node

1) เนื่องจาก MQTT ที่ใช้งานไม่สามารถส่งข้อความที่มีความยาวมากได้ ในการรับหรือส่งข้อมูลปริมาณมากจึงต้องแบ่งส่งหลายครั้งทำให้เกิดความยุ่งยากในการทำงาน
2) ไม่สามารถสร้างส่วนการทำงาน Topic Config ได้เนื่องจาก MQTT ที่ใช้งานไม่สามารถส่งข้อความที่มีความยาวมากได้ ในการทำงานส่วนนี้ต้องการข้อมูลทั้งหมดก่อนจึงทำการได้ โครงสร้างโปรแกรมที่ออกแบบไว้ไม่สามารถรองรับได้

### 5.2.2 ส่วน Gateway Node

1) การเชื่อมต่อ Internet WIFI ไม่เสถียร ทำให้ไม่สามารถ รับข้อความจาก MQTT ได้ตลอดเวลา

### 5.2.3 ส่วน Web browser

1) เนื่องจาก MQTT ที่ใช้งานไม่สามารถส่งข้อความที่มีความยาวมากได้ ในการรับหรือส่งข้อมูลปริมาณมากจึงต้องแบ่งส่งหลายครั้งทำให้เกิดความยุ่งยากในการทำงาน

## 5.3 แนวทางการพัฒนาต่อ

1) ใช้บริการจาก MQTT Server ที่สามารถรองรับการรับส่งข้อความปริมาณมากกว่านี้ได้ หรือสร้าง Topic ใหม่สำหรับใช้ยืนยันว่ารับข้อมูลจาก Topic Config ได้ครบแล้วจากนั้นส่งข้อความสั่งให้ I/O Node ทำการตั้งค่าตัวเองใหม่
2) สร้าง MQTT server ที่เป็น Database เก็บข้อมูลเพื่อสร้างสถิติเก็บไว้วิเคราะห์เพิ่มเติม
3) สร้าง Node สำหรับควบคุมภายใน WIFI Mesh Network
4) เพิ่มช่องทางสัญญาณ WIFI ให้ Gateway Node เพื่อให้สามารถรับข้อความจาก MQTT ได้ตลอดเวลา
5) เพิ่มชนิดของ Input/Output ที่สามารถใช้ได้

# บรรณานุกรม


กอบเกียรติ สระอุบล. 2561. **พัฒนา IOT บนแพลตฟอร์ม Arduino และ Raspberry Pi**. กรุงเทพฯ : หสม สำนักพิมพ์ อินเตอร์มีเดีย

**การทดลองใช้งาน NodeMCU กับ MQTT โดยเปิดปิดไฟ LED IN บน NODEMCU ผ่านหน้าเว็บ.** 2560. [Online]. Available : https://havespirit.blogspot.com/2017/03/Nodemcu-MQTT.html

เจ้าของร้าน. 2561. **ESP32 เบื้องต้น :: บทที่ 1 แนะนำ ESP32.** [Online]. Available : https://www.ioxhop.com/article/62/esp32-เบื้องต้น-บทที่-1-แนะนำ-esp32

เจ้าของร้าน. 2561. **ESP32 เบื้องต้น :: บทที่ 10 การใช้งาน WIFI.** [Online]. Available : https://www
.ioxhop.com/article/71/esp32-เบื้องต้น-บทที่-10-การใช้งาน-WIFI

เจ้าของร้าน. 2561. **ESP32 เบื้องต้น :: บทที่ 13 เชื่อมต่อกับ MQTT.** [Online]. Available : https://www.ioxhop.com/article/74/esp32-เบื้องต้น-บทที่-13-เชื่อมต่อกับ-MQTT

เจ้าของร้าน. 2561. **ESP32 เบื้องต้น :: บทที่ 2 บอร์ดพัฒนา ESP32.** [Online]. Available : https://www.ioxhop.com/article/63/esp32-เบื้องต้น-บทที่-2-บอร์ดพัฒนา-esp32

เจ้าของร้าน. 2561. **ESP32 เบื้องต้น :: บทที่ 4 พัฒนา ESP32 ด้วย Arduino.** [Online]. Available : https://www.ioxhop.com/article/65/esp32-เบื้องต้น-บทที่-4-พัฒนา-esp32-ด้วย-arduino

เจ้าของร้าน. 2561. **ESP32 เบื้องต้น :: บทที่ 8 การใช้งานเซ็นเซอร์ต่าง ๆ.** [Online]. Available : https: //www.ioxhop.com/article/69/esp32-เบื้องต้น-บทที่-8-การใช้งานเซ็นเซอร์ต่าง-ๆ

**ทำความรู้จักกับ MQTT และ CoAP โปรโทคอลสำหรับรับส่งข้อมูลบนเครือข่าย IoT.** 2560. [Online]. Available : http://www.adslthailand.com/post/MQTT-coap-comparison-iot-protocol

วิสิทธิ์ เวียงนาค. 2561. **ใส่ใจ 2. ESP32 สมองของระบบใส่ใจ.** [Online]. Available : https://medium.com/@visitwnk/ใส่ใจ-2-esp32-สมองของระบบใส่ใจ-3fbfb8504acf

Admin. 2559. **คู่มือวิธีการติดตั้งระบบ Wireless LAN ขั้นต้น 3 แบบ.** [Online]. Available : https://www.enterpriseitpro.net/บทความ-คู่มือวิธีการติ/

ck blog. ม.ป.ป. **ESP32 ADC & DAC.** [Online]. Available : https://ckblog2016.wordpress.com/2018/03/03/esp32-adc-dac/

**DAC's on ESP32.** ม.ป.ป. [Online]. Available : https://www.xtronical.com/basics/audio/dacs-on-esp32/

Dave Roos. ม.ป.ป. **How Wireless Mesh Networks Work.** [Online]. Available : https://computer.howstuffworks.com/how-wireless-mesh-networks-work.htm

einstronic. 2560. **INTRODUCTION TO NodeMCU ESP32 DevKIT v1.1.** [Online]. Available : https://einstronic.com/wp-content/uploads/2017/06/NodeMCU-32S-Catalogue.pdf

Espressif Systems. ม.ป.ป. **ESP32.** [Online]. Available : https://www.espressif.com/en/products/hardware/esp32/overview

Green Path Technologies Co.,ltd. ม.ป.ป. **What is Mesh Network.** [Online]. Available : http://cloudbreaker.biz/library/pdf/What_is_Mesh_Network.pdf

Louis E. Frenzel. 2548. **A Dozen Top Applications For Mesh Networks.** [Online]. Available : https://www.electronicdesign.com/energy/dozen-top-applications-mesh-networks

Margaret Rouse. 2552. **wireless mesh network (WMN).** [Online]. Available : https://searchnetworking.techtarget.com/definition/wireless-mesh-network

**MQTT (Message Queuing Telemetry Transport) คืออะไร.** 2558. [Online]. Available : https://www.mindphp.com/บทความ/31-ความรู้ทั่วไป/3343-MQTT.html

saixiii. 2560. **JSON คืออะไร.** [Online]. Available : https://saixiii.com/what-is-json/

Sara Santos. 2561. **ESP32 vs ESP8266 – Pros and Cons.** [Online]. Available : https://makeradvi

sor.com/esp32-vs-esp8266/

The HiveMQ Team. 2556. **Build your own Javascript MQTT Web Application.** [Online]. Available : https://www.hivemq.com/blog/build-javascript-MQTT-web-application/

Thomas Laurenson. 2561. **MQTT Web Application Using JavaScript and Paho MQTT Library - How to subscribe to data from LoRaWAN IoT devices.** [Online]. Available : https://www.thomaslaurenson.com/blog/2018/07/10/MQTT-web-application-using-javascript-and-websockets/

W3Schools. ม.ป.ป. **HTML5 Tutorial.** [Online]. Available : https://www.w3schools.com/html/default.asp

W3Schools. ม.ป.ป. **JavaScript Tutorial.** [Online]. Available : https://www.w3schools.com/js/default.asp

Wikipedia. 2562. **Wireless mesh network.** [Online]. Available : https://en.wikipedia.org/wiki/Wireless_mesh_network

Zerynth. ม.ป.ป. **NodeMCU ESP-32S.** [Online]. Available : https://docs.zerynth.com/latest/official/board.zerynth.Nodemcu_esp32/docs/index.html