แพลตฟอร์มให้บริการ IoT Cloud บนโพรโตคอล MQTT ด้วยการออกแบบผังการทำงานผ่าน Node-RED

นางสาวอภัสราภรณ์ ฉิมชัย
นางสาวอมรทิพย์ สุขแก้ว

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตรบัณฑิต
สาขาวิชาเทคโนโลยีอิเล็กทรอนิกส์ (คอมพิวเตอร์)
ภาควิชาเทคโนโลยีวิศวกรรมอิเล็กทรอนิกส์
วิทยาลัยเทคโนโลยีอุตสาหกรรม
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ
พ.ศ.2561

A MQTT-based IoT Cloud Platform with Flow Design by Node-RED

Miss Apatsaraporn Chimchai
Miss Amorntip Sookkaew

Project Report Submitted in Partial Fulfillment of the Requirements
for the Bachelor's Degree of Industrial Technology in
Electronics Technology (Computer)
Department of Electronics Engineering Technology
College of Industrial Technology
King Mongkut's University of Technology North Bangkok
2018

หัวข้อปริญญานิพนธ์	: แพลตฟอร์มให้บริการ IoT Cloud บนโพรโตคอล MQTT ด้วยการออกแบบผังการทำงานผ่าน Node-RED
โดย	: นางสาวอภัสราภรณ์ ฉิมชัย
	นางสาวอมรทิพย์ สุขแก้ว
ที่ปรึกษาปริญญานิพนธ์	: รองศาสตราจารย์ ดร.ชูพันธุ์ รัตนโภคา
สาขาวิชา	: เทคโนโลยีอิเล็กทรอนิกส์ (คอมพิวเตอร์)
ภาควิชา	: เทคโนโลยีวิศวกรรมอิเล็กทรอนิกส์
ปีการศึกษา	: 2561

วิทยาลัยเทคโนโลยีอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ อนุมัติให้นับปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

..... คณบดีวิทยาลัยเทคโนโลยีอุตสาหกรรม
(รองศาสตราจารย์ ดร. สมิตร ส่งพิริยะกิจ)

คณะกรรมการสอบปริญญานิพนธ์

..... ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.พิสิทธิ์ วิสุทธิเมธีกร)

..... กรรมการ
(รองศาสตราจารย์ ดร.ชูพันธุ์ รัตนโภคา)

..... กรรมการ
(อาจารย์ดำรงเกียรติ แซ่ลิ้ม)

Project Title : A MQTT-based IoT Cloud Platform with Flow Design by Node-RED

By : Miss Apatsarapron Chimchai

Miss Amorntip Sookkaew

Project Advisor : Assoc. Prof. Dr. Choopan Rattanapoka

Major Field : Electronics Technology (Computer)

Department : Electronics Engineering Technology

Academic Year : 2018

Accepted by the College of Industrial Technology, King Mongkut's University of Technology North Bangkok in Partial Fulfillment of the Requirements for the Bachelor's Degree.

……. Dean of College of Industrial Technology

(Assoc. Prof. Dr. Smith Songpiriyakij)

Project Committee

……. Chairperson

(Asst. Prof. Dr. Pisit Wisutmetheekorn)

……. Member

(Assoc. Prof. Dr. Choopan Rattanapoka)

……. Member

(Mr. Damrongkiat Lim)

กิตติกรรมประกาศ

บทความวิจัยการออกแบบและพัฒนาแพลตฟอร์มให้บริการ IoT Cloud บนโพรโตคอล MQTT ด้วยการออกแบบผังการทำงานผ่าน Node-RED สำเร็จลุล่วงไปได้ด้วยดีเนื่องด้วยได้รับความช่วยเหลือเป็นอย่างดีจาก รองศาสตราจารย์ ดร. ชูพันธุ์ รัตนโภคา อาจารย์ที่ปรึกษาบทความวิจัยที่ให้คำแนะนำและข้อคิดเห็นต่าง ๆ ของการจัดทำบทความวิจัย และการแก้ไขข้อบกพร่องต่าง ๆ มาโดยตลอด

ขอกราบขอบพระคุณบุพการีเป็นอย่างสูง ซึ่งให้การสนับสนุนในทุก ๆ ด้านเป็นแรงผลักดัน และให้กำลังใจคอยสนับสนุนแก่ผู้จัดทำเสมอมาจนสำเร็จการศึกษา ขอขอบพระคุณคณะอาจารย์สาขาเทคโนโลยีวิศวกรรมอิเล็กทรอนิกส์ แขนงคอมพิวเตอร์ ทุกท่านที่ได้ประสิทธิ์ประสาทวิชาความรู้ และให้ความช่วยเหลือด้านเทคนิคหลาย ๆ อย่างเป็นอย่างดี และขอขอบพระคุณทุกท่านและผู้ที่มีส่วนเกี่ยวข้องกับความสำเร็จแต่มิได้เอ่ยนามทุกท่าน มา ณ ทีนี้ด้วย

สุดท้ายนี้ทางคณะผู้จัดทำ ต้องขอขอบพระคุณท่านกรรมการสอบบทความวิจัยทุกท่านเป็นอย่างสูง ที่ได้ช่วยพิจารณาและให้คำแนะนำในการตรวจทานแก้ไข อนุมัติจนบทความวิจัยนี้สำเร็จเป็นไปตามวัตถุประสงค์และขอบเขตที่ตั้งไว้ทุกประการ ซึ่งผู้จัดทำหวังว่า บทความวิจัยฉบับนี้ จะเป็นประโยชน์ต่อผู้ที่ใช้งานแพลตฟอร์มให้บริการ IoT Cloud บนโพรโตคอล MQTT ด้วยการออกแบบผังการทำงานผ่าน Node-RED

คณะผู้จัดทำ

แพลตฟอร์มให้บริการ IoT Cloud บนโพรโตคอล MQTT ด้วยการออกแบบผังการทำงานผ่าน Node-RED

อภัสราภรณ์ ฉิมชัย อมรทิพย์ สุขแก้ว* และ ชูพันธุ์ รัตนโภคา

บทคัดย่อ

บทความนี้เสนอการออกแบบ และพัฒนาแพลตฟอร์มการให้บริการ IoT ในรูปแบบของเว็บแอพพลิเคชั่นที่พัฒนาด้วย MERN Stack โดยภายในแพลตฟอร์มจะให้บริการ 3 ส่วนหลัก ๆ ได้แก่ MQTT Broker สำหรับการรับส่งข้อมูล Node-RED สำหรับการออกแบบและควบคุมการประมวลผลข้อมูล รวมถึงแสดงข้อมูลออกมาอยู่ในรูปแบบกราฟิกผ่านทาง Dashboard และ ระบบฐานข้อมูล InfluxDB และ MongoDB สำหรับการเก็บข้อมูล โดยโครงสร้างพื้นฐานของแพลตฟอร์มจะทำงานบน Docker ซึ่งอยู่ในรูปแบบ Docker Container จากการทดสอบพบว่าแพลตฟอร์มสามารถรองรับการพัฒนาระบบ IoT ได้เป็นอย่างดี โดยผู้ใช้ไม่จำเป็นต้องติดตั้งบริการต่าง ๆ เอง

คำสำคัญ : แพลตฟอร์มบริการ IoT, Node-RED, Docker, คลาวด์

ภาควิชาเทคโนโลยีวิศวกรรมอิเล็กทรอนิกส์, วิทยาลัยเทคโนโลยีอุตสาหกรรม, มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

* ผู้ติดต่อ, อีเมล์: s6003052422129@email.kmutnb.ac.th@kmutnb.ac.th

A MQTT-based IoT Cloud Platform with Flow Design by Node-RED

Apatsrapron Chimchai[1], Amorntip Sookkaew[2*] and Choopan Rattanapoka

Abstract

Keywords :

Department of Electronic Engineering Technology, College of Industrial Technology, King Mongkut's University of Technology North Bangkok.

[*] Corresponding author, E-mail: s6003052422129@email.kmutnb.ac.th

1. บทนำ

Internet of Things หรือ IoT กำลังได้รับความนิยมมากในปัจจุบัน เนื่องจากเป็นแนวคิดที่ให้อุปกรณ์สามารถเชื่อมต่อเข้าด้วยกัน และสามารถเพิ่มการควบคุมอุปกรณ์ได้อย่างอัจฉริยะ ทำให้เสมือนว่าอุปกรณ์สามารถเชื่อมต่อและพูดคุยกันเองได้ ทำให้การใช้ชีวิตประจำวันของเราสะดวกสบายและปลอดภัยมากขึ้น ในช่วงเวลาไม่กี่ปีที่ผ่านมามีงานวิจัยและนวัตกรรมต่าง ๆ ที่ประยุกต์ใช้งานในหลายด้าน จาก [1] ได้แบ่งด้านการประยุกต์ใช้งาน IoT ออกเป็น 5 ด้านใหญ่ๆ คือ ด้านขนส่ง ด้านสุขภาพ ด้านสิ่งแวดล้อมอัจฉริยะ ด้านสังคม และด้านการประยุกต์ใช้งานในอนาคต ตัวอย่างงานวิจัยที่เกิดขึ้นภายในปีที่ผ่านมาเช่น [2] ได้นำเสนอระบบเรียนรู้การทำกิจกรรมในชีวิตประจำวันภายในชุมชนที่เป็นบ้านอัจฉริยะ (Smart home) [3] ได้นำเสนอกรอบการทำงานของระบบขนส่งอัจฉริยะ (Smart transportation) โดยมีการนำข้อมูลที่อยู่ในรูปแบบ Big data มาช่วยในการประมวลผลการตัดสินใจ และ [4] ได้นำเสนอการออกแบบสถาปัตยกรรมของระบบ IoT เพื่องานด้านสุขภาพ

การออกแบบและพัฒนาระบบ IoT ขึ้นมาเองนั้นจำเป็นจะต้องมีโครงสร้างพื้นฐาน เช่น เครื่องแม่ข่ายและต้องติดตั้งบริการต่าง ๆ สำหรับการใช้งานด้วยตัวเอง บริษัทใหญ่ ๆ เลยมีการนำเสนอแพลตฟอร์มการสร้างระบบ IoT ให้ผู้ใช้สามารถนำไปใช้งานได้ง่าย เช่น Microsoft Azure, Amazon AWS และ Google Cloud IoT แต่อย่างไรก็ตามเนื่องจากเป็นการให้บริการของบริษัทเอกชนที่ตั้งอยู่ต่างประเทศ อาจส่งผลให้มีค่าใช้จ่ายที่สูง และอาจจะมีปัญหาเรื่องความรวมเร็วในการรับ-ส่งข้อมูล

สำหรับในประเทศไทยเองนั้น ก็มีแพลตฟอร์ม IoT ที่ได้รับความนิยมคือ NetPIE [5] ซึ่งผู้ใช้งานสามารถเปิดบริการช่องทางรับ-ส่งข้อมูลระหว่างอุปกรณ์ได้ ผ่านไลบรารี Microgear รวมถึงมีหน้าแสดงข้อมูลที่เรียกว่า Dashboard ให้ผู้ใช้งานสามารถดูข้อมูลต่าง ๆ ของระบบในรูปแบบแผนภูมิได้อย่างสะดวก แต่อย่างไรก็ตาม NetPIE ยังขาดการให้บริการในส่วนพื้นที่เก็บข้อมูลเพื่อนำมาวิเคราะห์หรือประมวลผลต่อไป ทำให้ผู้ใช้งานยังคงต้องไปหาบริการพื้นที่เก็บข้อมูลที่อื่น หรือตั้งติดตั้งด้วยตนเองอยู่ดี

บทความวิจัยนี้ต้องการนำเสนอการออกแบบ และพัฒนาแพลตฟอร์มการให้บริการ IoT ในรูปแบบของเว็บแอปพลิเคชันที่พัฒนาด้วย MERN Stack โดยภายในแพลตฟอร์มจะให้บริการ 3 ส่วนหลักๆ คือ (1) MQTT Broker สำหรับการรับส่งข้อมูล, (2) Node-RED สำหรับการออกแบบและควบคุมการประมวลผลข้อมูล รวมถึงแสดงข้อมูลออกมาอยู่ในรูปแบบกราฟิกผ่านทาง Dashboard และ (3) ระบบฐานข้อมูล InfluxDB และ MongoDB สำหรับการเก็บข้อมูล

2. วรรณกรรมที่เกี่ยวข้อง

2.1 MERN Stack

MERN Stack เป็นเฟรมเวิร์คที่ถูกออกแบบมาเพื่อเป็นตัวช่วยในการพัฒนาเว็บไซต์และรวมถึงเว็บแอปพลิเคชัน MERN Stack ประกอบด้วย 4 ส่วนการทำงาน คือ MongoDB, Express JS, React JS และNode JS โดยมี React JS ทำงานในส่วนของ Client ในขณะที่ Express, Node JS และ MongoDB ทำงานที่ในส่วนของ Server

MongoDB เป็นระบบฐานข้อมูลเชิงเอกสาร ซึ่งได้ถูกจำแนกให้เป็นระบบฐานข้อมูลประเภท NoSQL มีการเก็บข้อมูลเป็นในรูปแบบเอกสารที่มีโครงสร้างคล้ายกับเอกสาร JSON การบันทึกข้อมูลทุก ๆ Record ใน

MongoDB จะเรียกว่า Document ซึ่งจะเก็บค่าเป็น Key และ Value และการเก็บข้อมูล Document ใน MongoDB จะถูกเก็บไว้ใน Collections

Express JS คือ Web Application Framework ที่ได้รับความนิยมมาก สำหรับทำงานบนแพลตฟอร์มของ Node.js ซึ่งเป็น Server ตัวหนึ่ง โดยทั้ง Express.js และ Node.js ต่างก็ใช้ภาษา JavaScript ในการพัฒนา

React JS คือ JavaScript Library เป็นแค่ UI โดยสร้างมาจากพื้นฐานแนวความคิดแบบ MVC (Model View Controller) รองรับการเขียนด้วย JSX (JavaScript Syntax Extension) โดยภายในเว็บจะมองส่วนต่าง ๆ เป็น Component และข้อมูลที่อยู่ใน Component แต่ละชั้นจะเรียกว่า State ส่วนข้อมูลที่ถูกส่งต่อจาก Component ชั้นบนลงไปชั้นล่าง เรียกว่า Props

Node JS คือ Cross-Platform Runtime Environment สำหรับฝั่ง Server ซึ่งถูกพัฒนาขึ้นด้วยภาษา JavaScript โดยปกติแล้ว JavaScript จะทำงานในส่วน ของฝั่ง Client ซึ่งจะทำงานในเชิงโต้ตอบ Web Browser จากนั้นได้มีการพัฒนา NodeJS เพื่อมาเป็น Runtime ใน ส่วนของฝั่ง Server โดย NodeJS จะใช้รูปแบบ Event-driven, Non-blocking I/O ทำให้กินทรัพยากรระบบน้อย มีประสิทธิภาพ และสามารถประมวลผลได้อย่างรวดเร็ว

2.2 Docker

Docker [6] คือ Engine ตัวหนึ่งที่มีการทำงานในลักษณะจำลองสภาพแวดล้อมขึ้นมาบนเครื่อง Server เพื่อใช้ในการเรียกใช้งาน Service ที่ต้องการ มีการทำงานคล้ายกับ Virtual Machine แต่ข้อแตกต่างที่ชัดเจนคือ Virtual Machine เป็นการจำลองทั้ง OS เพื่อใช้งานและหากต้องการใช้งาน Service ใด ๆ จึงทำการติดตั้งเพิ่มเติมบน OS นั้น ๆ แต่สำหรับ Docker จะใช้ Container ในการจำลองสภาพแวดล้อมขึ้นมา เพื่อใช้งานสำหรับ 1 Service ที่ต้องการใช้งาน โดยไม่ต้องมีส่วนของ OS เข้าไปเกี่ยวข้องเหมือน Virtual Machines

2.3 Node-RED

Node-RED [7] เป็นซอฟต์แวร์เปิด ซึ่งเป็นเครื่องมือสำหรับนักพัฒนาโปรแกรมในการเชื่อมต่ออุปกรณ์ฮาร์ดแวร์เข้ากับ APIs (Application Programming Interface) และ Online Service โดยการนำไปใช้จะทำให้ไม่ต้องเขียน API ฝั่ง Server เอง โดย Node-RED เป็นการพัฒนาโปรแกรมแบบ Flow-Based Programming สามารถออกแบบ API ในการรับค่า คำนวณ แปลงข้อมูล เก็บข้อมูล หรือเชื่อมต่อกับบริการอื่น ๆ นอกจากนี้ Node-RED จะเป็นลักษณะของ Browser-based Flow Editor มี Browser ในการเลือก Node มาวางแล้วเชื่อมต่อเพื่อควบคุม I/O ต่าง ๆ ได้ โดย Node-RED จะมี Node ให้เลือกใช้งานซึ่งจะทำให้เกิด Flow แล้วทำการกำหนดค่าการการทำงาน จากนั้นก็ลากสาย (pipe) สำหรับเชื่อมโยงข้อมูลโดยที่ไม่จำเป็นที่จะเขียนต้อง Code ในการพัฒนาโปรแกรม

2.4 MQTT

MQTT [8] เป็น โพรโตคอล ที่ออกแบบมาเพื่อการเชื่อมต่อแบบ M2M (Machine-to-Machine) คืออุปกรณ์กับอุปกรณ์ผ่านเครือข่ายอินเทอร์เน็ต ซึ่งจะทำให้สามารถ ควบคุมอุปกรณ์ต่าง ๆ จากที่อื่นได้ในรูปแบบของ publish/subscribe องค์ประกอบที่สำคัญของ MQTT ประกอบไปด้วย

MQTT Broker ทำหน้าที่เป็นตัวกลางคอยจัดการกับข้อความ โดยจะรับข้อความจาก Publisher และส่งต่อไปยัง Subscriber โดยอ้างอิงจาก Topic ของข้อความ

Publisher ทำหน้าที่ส่งข้อความตาม Topic ที่กำหนดไปยัง MQTT Broker

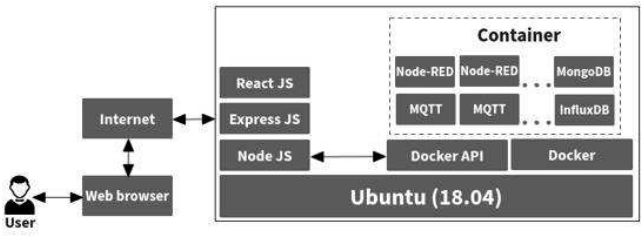
Subscriber ทำหน้าที่รับข้อความตาม Topic ที่ได้ subscribe ไว้กับ MQTT Broker

2.5 InfluxDB

InfluxDB [9] เป็นระบบฐานข้อมูลที่เป็นซอฟต์แวร์เปิด ถูกออกแบบมาสำหรับการเก็บข้อมูลในรูปแบบ Time Series ซึ่ง InfluxDB ถูกพัฒนาขึ้นด้วยภาษา Go เหมาะสำหรับการจัดเก็บข้อมูลที่มีประสิทธิภาพสูงและการเรียกค้นข้อมูลชุดข้อมูลแบบเรียลไทม์ ซึ่งจะใช้เป็นที่เก็บข้อมูลสำหรับกรณีการใช้งานใด ๆ ที่เกี่ยวข้องกับข้อมูลที่มีการ Timestamp เป็นจำนวนมาก เช่น การวัดอัตราการเต้นหัวใจ การทำงานของข้อมูล การทำงานของข้อมูลเซ็นเซอร์ Internet of Things (IoT) และการวิเคราะห์แบบเรียลไทม์ การใช้ InfluxDB ในการเก็บข้อมูลจะทำให้ประหยัดเนื้อที่ ได้มากยิ่งขึ้นด้วยการกำหนดค่า InfluxDB เพื่อเก็บข้อมูลไว้เป็นระยะเวลาที่กำหนดโดยอัตโนมัติ เมื่อหมดอายุจะสามารถลบข้อมูลที่ไม่ต้องการออกจากระบบได้

3. วิธีการวิจัย

แพลตฟอร์มให้บริการ IoT Cloud บนโพรโตคอล MQTT ด้วยการออกแบบผังการทำงานผ่าน Node-RED ที่ถูกออกแบบในบทความวิจัยนี้ แสดงดังรูปที่ 1 โดยผู้ใช้งานจะสามารถเปิดใช้บริการ MQTT Broker, Node-RED และระบบฐานข้อมูล MongoDB และ InfluxDB ผ่านทางเว็บแอปพลิเคชันที่พัฒนาด้วย MERN Stack ซึ่งภายในแพลตฟอร์มบริการทั้ง 4 บริการ (MQTT Broker, Node-RED, MongoDB และ InfluxDB) อยู่ในรูปแบบของ Docker Container ที่ทำงานบน Docker โดยได้ติดตั้งซอฟต์แวร์ทั้งหมดบน Ubuntu เวอร์ชัน 18.04



รูปที่ 1 สถาปัตยกรรมของแพลตฟอร์ม IoT Cloud

ผู้ใช้แต่ละคนสามารถสร้างบริการ MQTT Broker และ Node-RED ได้อิสระออกจากกัน ในขณะที่บริการระบบฐานข้อมูล MongoDB และ InfluxDB จะมีเพียง Container เดียว ที่ใช้งานร่วมกันของทุกผู้ใช้งาน แต่จะมีการกำหนดสิทธิในการใช้งานไม่ให้เข้าถึงข้อมูลของผู้ใช้งานคนอื่นด้วยรหัสผ่าน การที่ Container ของระบบฐานข้อมูลเป็นลักษณะของการใช้งานร่วมกันของทุกผู้ใช้งาน มีวัตถุประสงค์เพื่อลดการใช้ทรัพยากรของระบบ

เว็บแอปพลิเคชันที่พัฒนาด้วย MERN Stack จะมีส่วนที่ติดต่อกับ Docker ผ่านทาง Docker API โดยในบทความวิจัยนี้ได้ใช้ไลบรารี Dockerode [10] สำหรับการสั่งงานเปิด ปิด และดูสถานะของ Container

การพัฒนาแพลตฟอร์มสำหรับงานวิจัยนี้ แบ่งออกเป็นขั้นตอนหลัก ๆ ได้ 4 ขั้นตอน ดังนี้

3.1 การจัดเตรียม Docker Images

หลังจากติดตั้ง Docker ลงบนระบบปฏิบัติการ Ubuntu 18.04 เรียบร้อยแล้ว จะต้องจัดเตรียม Docker Images สำหรับให้บริการผู้ใช้งาน โดย Docker Image ที่จัดเตรียมจะมีทั้งหมด 4 Images ประกอบด้วย Node-RED, MQTT Broker (mosquitto), MongoDB และ InfluxDB ในการติดตั้งแต่ละ Image นั้น จะใช้คำสั่ง docker pull และตามด้วยชื่อของ Images ที่ต้องการติดตั้ง สำหรับ Docker Image ของ MQTT, MongoDB

และ InfluxDB นั้นสามารถใช้งานได้เลยโดยไม่ต้องตั้งค่าอะไรเพิ่มเติม อย่างไรก็ตาม Docker Image ของ Node-RED นั้นจำเป็นต้องมาตั้งค่าเพื่อเพิ่มโมดูลในการแสดง Dashboard รวมถึงโมดูลในการเชื่อมต่อกับระบบฐานข้อมูล InfluxDB และ MongoDB

โดยการปรับแต่ง Docker Image ของ Node-RED นั้น จะเริ่มจากการเปิด Docker Container ของ Node-RED ด้วยคำสั่ง docker run จากนั้นจึงเข้าไปหน้าจัดการ Node-RED ผ่านทางเว็บเบราว์เซอร์ เพื่อเลือกเมนู Manage Palette จากนั้นจึงค้นหาและติดตั้ง Palette ชื่อ Dashboard, InfluxDB และ MongoDB เมื่อติดตั้ง Palette ที่ต้องการเสร็จสิ้นแล้ว จะใช้คำสั่ง docker commit เพื่อบันทึก Docker Image ที่ปรับแต่งเรียบร้อยแล้ว เป็น Docker Image ตัวใหม่สำหรับใช้งานบนแพลตฟอร์มที่พัฒนา

3.2 การเชื่อมต่อระหว่างเว็บไซต์ และ Docker

หน้าที่การทำงานหลักของเว็บแอปพลิเคชัน คือ การให้ผู้ใช้งานสามารถบริหารจัดการบริการทั้ง 4 อย่าง ได้แก่ MQTT Broker, Node-RED, MongoDB และ InfluxDB ภายในแพลตฟอร์มได้ตนเอง โดยการบริหารจัดการบริการหมายถึง การเปิดใช้งานบริการ การปิดใช้งานบริการ การดูสถานะของบริการ และการลบบริการ ดังนั้นเว็บแอปพลิเคชันจึงต้องสามารถเชื่อมต่อกับ Docker ได้ โดยงานวิจัยนี้ได้พัฒนาให้เว็บแอปพลิเคชันสามารถติดต่อสื่อสารกับ Docker ได้ผ่านทาง Docker API ซึ่งได้ใช้ไลบรารี Dockerode เข้ามาช่วยเพื่อให้การพัฒนาโปรแกรมสะดวกสบายมากยิ่งขึ้น

รูปที่ 2 แสดงตัวอย่างโค้ดในการเชื่อมต่อกับ Docker API เพื่อสร้าง Docker Container จาก Docker Image ที่ทำหน้าที่เป็น MQTT Broker ชื่อ mosquitto โดยมีการกำหนดให้ผูกหมายเลข Port 1883 ที่ทำงานบนโพรโตคอล TCP ของ Container เข้ากับหมายเลข Port จริงของเครื่องกายภาพที่สุ่มขึ้น (HostPort : “0”) ถ้าการสร้าง Container สำเร็จแล้วจะได้ข้อมูลตอบกลับจาก Docker ในรูปแบบเอกสาร JSON [{“IP”:”0.0.0.0”, ”PrivatePort”:1883,”PublicPort:32773,”Type”:”tcp”}] ซึ่งหมายถึงหมายเลข Port ที่เครื่องกายหมายเลข 32773 ได้ถูกผูกเข้ากับ Port หมายเลข 1883 ของ Container

```
var Docker = require('dockerode');
var dockerHostIP = "172.20.10.2"
var dockerHostPort = 2375
    var docker = new Docker({ host: dockerHostIP, port: dockerHostPort });
     docker.createContainer({
       name: namecon,
       Image: 'toke/mosquitto',
       AttachStdin: false,
       AttachStdout: false,
       AttachStderr: false,
       Tty: false,
       Cmd: [],
       OpenStdin: false,
       StdinOnce: false,
       HostConfig: {
                   PortBindings: {
                       "1883/tcp": [
                       {
                           HostIp: "0.0.0.0",
                           HostPort: "0"
                       }
                     ]
                   }
                 }

     }).then(function(container) {
       return container.start();
     }).catch(function(err) {
       console.log(err);
     });
```

รูปที่ 2 ส่วนโปรแกรมสำหรับสร้าง Docker Container

3.3 จัดการความปลอดภัยในการใช้งาน Node-Red

Node-RED ตามการตั้งค่าปกติ จะไม่เปิดการป้องกันความปลอดภัย ซึ่งทำให้ผู้ใช้งานบางคนอาจจะสุ่มหมายเลข Port เพื่อเข้าหน้าจัดการ Node-RED ของผู้ใช้งานคนอื่นได้ ดังนั้นเพื่อเพิ่มความปลอดภัยในการใช้งาน Node-RED นั้น จึงจำเป็นต้องเปิดระบบรักษาปลอดภัยอย่างง่ายของ Node-RED โดยผู้ใช้จะต้องใส่ Username และ Password เพื่อเข้าหน้าจัดการของ Node-RED ได้ ซึ่งสามารถทำได้โดยแก้ไขที่ไฟล์ /data/settings.js

อย่างไรก็ตาม แต่ละ Container ของผู้ใช้ มีการตั้งค่า Username และ Password ที่ต่างกัน ซึ่งผู้ใช้งานจะเป็นผู้กำหนดเอง ดังนั้นภายในแพลตฟอร์มจึงได้เตรียมไฟล์ชื่อ template.js ซึ่งก็คือไฟล์ settings.js ของ Node-RED

เอาไว้ และกำหนดข้อความ {{username}} และ {{password}} ในตำแหน่งที่ต้องใส่ Username และ Password เมื่อผู้ใช้สร้างบริการ Node-RED ขึ้นมา เว็บจะให้ใส่ Username และ Password สำหรับเข้าหน้าบริหาร Node-RED จากนั้นเว็บจะอ่านข้อมูลจากไฟล์ template.js แล้วแทนที่ข้อความ {{username}} และ {{password}} ด้วยค่า Username และ Password ที่ผู้ใช้กรอกผ่านหน้าเว็บ และบันทึกเก็บไว้ในแพลตฟอร์มเพื่อใช้งานในครั้งถัดไปส่วนของโปรแกรมที่ใช้ในการจัดการไฟล์ settings.js ของ Node-RED แสดงดังรูปที่ 3

เมื่อผู้ใช้เปิดบริการ Node-RED ส่วนของโปรแกรมที่สร้าง Container จะต้องเพิ่มในส่วนของการนำไฟล์ที่สร้างขึ้นมา เข้าไปแทนที่ไฟล์ /data/settings.js ใน Container ด้วยการเพิ่มเติมแอททริบิวท์ Binds ซึ่งสามารถเขียนได้ดังรูปที่ 4

```
var fs = require('fs')
  fs.readFile('nodereditemplate/template.js', "utf8", function (err,data) {
    if (err) {
      return console.log(err);
    }
  var result = data.replace(/{{username}}/g, userred);
  var result2 = result.replace(/{{password}}/g, hash);
    fs.writeFile('usepassnodered/'+username+namered+'.js', result2, 'utf8', function (err) {
      if (err) return console.log(err);
    });
});
```

รูปที่ 3 ส่วนโปรแกรมในการเตรียมไฟล์ settings.js

```
var noderedConfig = "/home/pat/usepassnodered/"+namecon+".js";
  docker.createContainer({
    name: namecon,
    Image: 'project-nodered',
    AttachStdin: false,
    AttachStdout: false,
    AttachStderr: false,
    Tty: false,
    Cmd: [],
    OpenStdin: false,
    StdinOnce: false,
    HostConfig: {
                Binds: [
                    noderedConfig+":/data/settings.js"
                ],
                PortBindings: {
                    "1880/tcp": [
                    {
                        HostIp: "0.0.0.0",
                        HostPort: "0"
                    }
                  ]
                }
            }

  }).then(function(container) {
    return container.start();
  }).catch(function(err) {
    console.log(err);
  });
```

รูปที่ 4 ส่วนโปรแกรมสำหรับสร้าง Docker Container และมีการแทนที่ไฟล์ใน Container

3.4 การสร้างบัญชีผู้ใช้งานระบบฐานข้อมูล

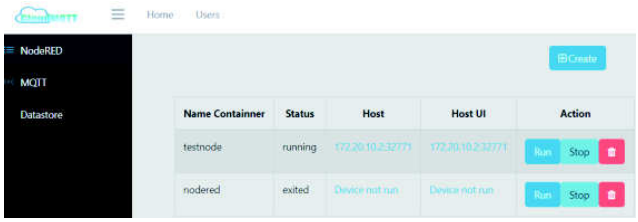
เนื่องจากการออกแบบให้ระบบฐานข้อมูลเป็นการใช้งานร่วมกันของทุกผู้ใช้งาน ดังนั้นจึงเป็นการให้ 1 ฐานข้อมูลต่อ 1 บริการที่ผู้ใช้เปิดใช้บริการ ซึ่งชื่อฐานข้อมูลที่ผู้ใช้งานต้องการสร้างนั้น จะต้องไม่ซ้ำกับผู้ใช้งานคนอื่น โดยหน้าเว็บจะรับค่าจากผู้ใช้งาน 3 ค่า คือ Username ที่จะใช้งานฐานข้อมูล, Password ที่จะเข้าถึงฐานข้อมูล และชื่อฐานข้อมูล รูปที่ 5 แสดงตัวอย่างส่วนของโปรแกรมในการสร้างฐานข้อมูลในระบบฐานข้อมูล InfluxDB

```
const Influx = require('influx');
const influx = new Influx.InfluxDB('http://172.20.10.2:8086/')
    let dataname = dbname;
    let dbuser = name;
    let dbpassword = password
influx.createDatabase(dataname)
influx.createUser(dbuser, dbpassword)
influx.grantPrivilege(dbuser, 'ALL', dataname)
```

รูปที่ 5 ส่วนโปรแกรมในการสร้างฐานข้อมูล InfluxDB

4. ผลการทดลอง

แพลตฟอร์ม IoT Cloud ที่พัฒนาขึ้นมา หลังจากเข้าสู่ระบบแล้ว จะมีเมนูให้จัดการบริการหลักทั้ง 3 บริการ คือ NodeRED, MQTT และ Datastore ดังรูปที่ 6



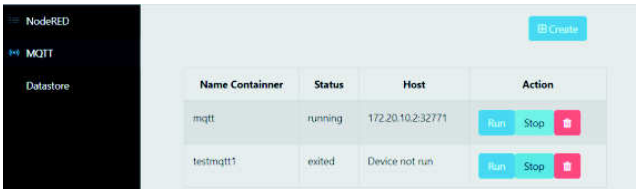
รูปที่ 6 หน้าจอหลักของแพลตฟอร์ม

เพื่อทดสอบการทำงานของแพลตฟอร์มที่พัฒนาขึ้น จึงนำมาประยุกต์ใช้เป็นระบบพื้นฐานของการทำระบบ IoT ในการวัดอุณหภูมิ ซึ่งข้อมูลของอุณหภูมิจะมาจาก

เซ็นเซอร์ที่ติดตั้งบน NodeMCU และจะนำค่าอุณหภูมินั้นมาบันทึกลงฐานข้อมูล InfluxDB และแสดงค่าอุณหภูมิแบบเวลาจริงบนแผนภูมิแบบเกจ และนำค่าจากฐานข้อมูล InfluxDB ย้อนหลัง 1 วัน มาแสดงในรูปแผนภูมิเชิงเส้น ขั้นตอนในการพัฒนาระบบ IoT ในการวัดอุณหภูมิผ่านแพลตฟอร์ม IoT Cloud มีดังนี้

4.1 เปิดบริการ MQTT Broker

ทำการเปิดบริการ MQTT Broker โดยการเข้าเมนู MQTT และกดปุ่ม Create จากนั้นใส่ชื่อของบริการ MQTT เมื่อสร้างบริการเสร็จสิ้นแล้ว ผู้ใช้จะได้ Container ที่ทำหน้าที่เป็น MQTT Broker ส่วนตัว และเว็บจะแสดงหมายเลข IP (172.20.10.2) และ Port (32771) ของ MQTT Broker ที่เปิดให้บริการพร้อมกับสถานะ Running หมายถึง MQTT Broker พร้อมทำงานแล้ว ดังรูปที่ 7



รูปที่ 7 หน้าแสดงสถานะของ MQTT Broker

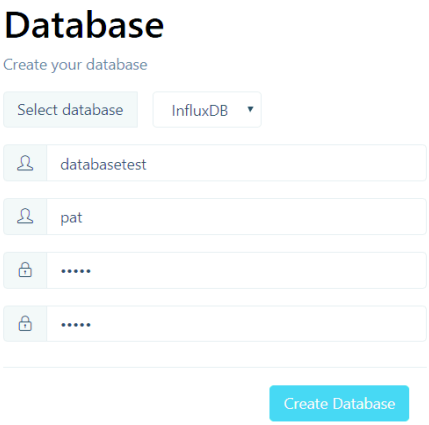
4.2 ตั้งค่าให้อุปกรณ์ส่งค่าอุณภูมิ

ในการทดลองนี้ ได้ประกอบฮาร์ดแวร์ โดยใช้ NodeMCU และเซนเซอร์วัดค่าอุณภูมิ ซึ่งค่าของอุณหภูมิที่อ่านได้มาจากเซ็นเซอร์นั้นจะถูกส่งไปยัง MQTT Broker ดังนั้นการเขียนโปรแกรมบน NodeMCU จึงต้องใช้ไลบรารี MQTT เพื่อส่งค่าเซ็นเซอร์อุณหภูมิไปยัง MQTT Broker ที่หมายเลข IP 172.20.10.2 และ หมายเลข Port 32771 และได้ใช้ Topic สำหรับการส่งข้อมูลในรูปแบบ Publish ไปยัง MQTT Broker ชื่อ temp

4.3 เปิดบริการฐานข้อมูล InfluxDB

เนื่องจากการออกแบบระบบ IoT ในการวัดอุณภูมิต้องการเก็บข้อมูลอุณหภูมิลงฐานข้อมูล InfluxDB ด้วย ดังนั้นจึงต้องเปิดบริการฐานข้อมูล InfluxDB การเปิดบริการฐานข้อมูล InfluxDB นั้น เข้าไปยังเมนู Datastore และกดปุ่ม Create จากนั้น ผู้ใช้สามารถเลือกได้ว่าจะเปิดบริการฐานข้อมูล MongoDB หรือ InfluxDB ตามด้วยชื่อฐานข้อมูล ชื่อบัญชีเข้าใจงานระบบฐานข้อมูล และรหัสผ่านเข้าใช้งานระบบฐานข้อมูล

จากรูปที่ 8 คือ เลือกเปิดใช้บริการฐานข้อมูล InfluxDB โดยให้สร้างฐานข้อมูลชื่อ databasetest และบัญชีผู้ใช้งานระบบฐานข้อมูลคือ pat เมื่อสร้างเสร็จสิ้นแล้ว เว็บจะแสดงข้อมูลที่สำคัญต่าง ๆ ดังรูปที่ 9

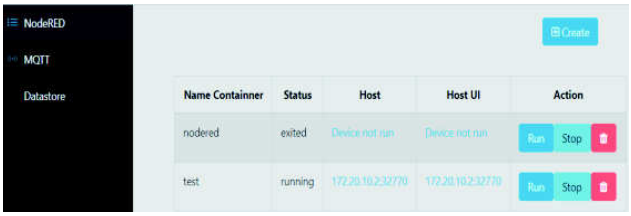


รูปที่ 8 หน้าเปิดบริการใช้งานฐานข้อมูล InfluxDB

รูปที่ 9 หน้าแสดงสถานะของฐานข้อมูล

4.4 เปิดบริการ Node-RED

Node-RED ถูกใช้สำหรับประมวลผลข้อมูลแบบ Flow สามารถเปิดบริการโดยเข้าไปที่เมนู NodeRED และกดปุ่ม Create จากนั้นให้ใส่ชื่อบริการ Node-RED รวมทั้งชื่อบัญชีเข้าใช้งาน และรหัสผ่านในการเข้าหน้าเว็บของ Node-RED เมื่อเปิดบริการเสร็จสิ้นแล้วหน้าเว็บจะแสดงชื่อบริการ สถานะการทำงาน และ ลิงค์เพื่อเข้าไปหน้าเว็บของ Node-RED ดังรูปที่ 10

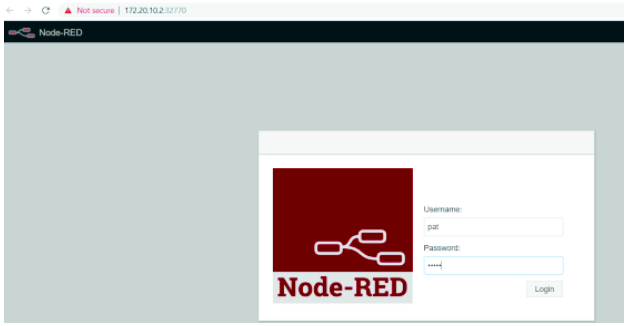


รูปที่ 10 หน้าแสดงสถานะของ Node-RED

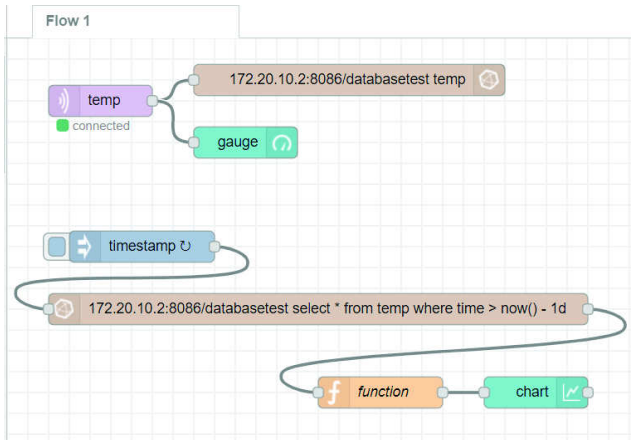
4.5 ออกแบบแผนผังการทำงานบน Node-RED

หลังจากการเปิดบริการ Node-RED เรียบร้อยแล้ว เมื่อผู้ใช้กดลิงค์เพื่อเข้าสู่ เว็บ Node-RED จะต้องใส่ Username และ Password ที่ได้กำหนดไว้ตั้งแต่ตอนเปิดบริการดังรูปที่ 11 เพื่อเพิ่มความปลอดภัยในการใช้งาน

จากนั้นออกแบบแผนผังการทำงานดังรูปที่ 12 ซึ่งการทำงานจะแบ่งออกเป็น 2 ส่วน คือ (1) แผนผังส่วนบน เป็นส่วนที่รับข้อมูลจาก MQTT Broker แล้วนำข้อมูลที่รับได้ไปบันทึกไว้ในฐานข้อมูล พร้อมทั้งส่งไปแสดงบนหน้า Dashboard ในรูปแบบแผนภูมิเกจ และ (2) แผนผังส่วนล่าง เป็นการนำข้อมูลที่เก็บไว้ในฐานข้อมูล โดยนำเฉพาะข้อมูลย้อนหลังจากเวลาปัจจุบัน 1 วัน จนถึงปัจจุบันเพื่อนำไปแสดงบน Dashboard ในรูปแบบของแผนภูมิเส้น



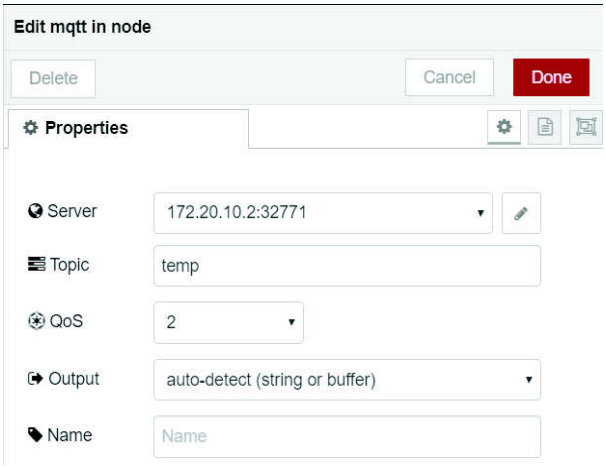
รูปที่ 11 หน้าเข้าสู่ระบบของ Node-RED



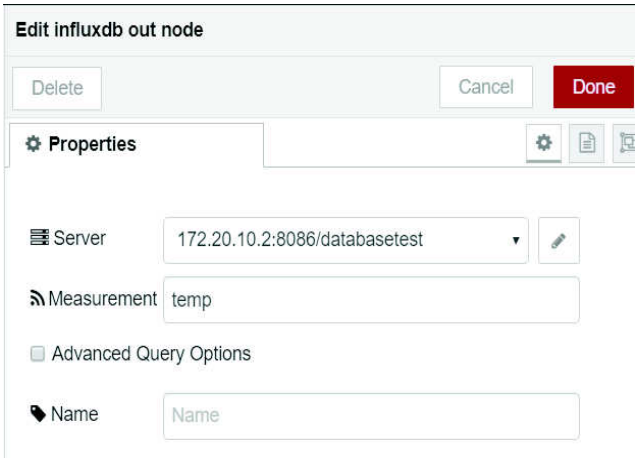
รูปที่ 12 หน้าออกแบบการทำงานของระบบ IoT บน Node-RED

โหนด temp เป็นโหนด MQTT In ของ Node-RED โดยจะ subscribe Topic ที่ชื่อ temp จาก MQTT Broker ที่หมายเลข IP 172.20.10.2 และหมายเลข Port คือ 32771 ในรูปแบบ 172.20.10.2:32771 ซึ่งก็คือ MQTT Broker ที่ได้เปิดบริการไปในหัวข้อที่ 4.1 การตั้งค่าของโหนดมีรายละเอียดดังรูปที่ 13

โหนดสำหรับเขียนข้อมูลลง InfluxDB คือ โหนด InfluxDB Out โดยตั้งค่าให้เชื่อมต่อไปยังฐานข้อมูล 172.20.10.2:8086/databasetest ที่ได้เปิดบริการในหัวข้อที่ 4.3 และกำหนดให้เก็บข้อมูลที่ได้รับลงตารางที่ชื่อว่า temp ดังแสดงในรูปที่ 14

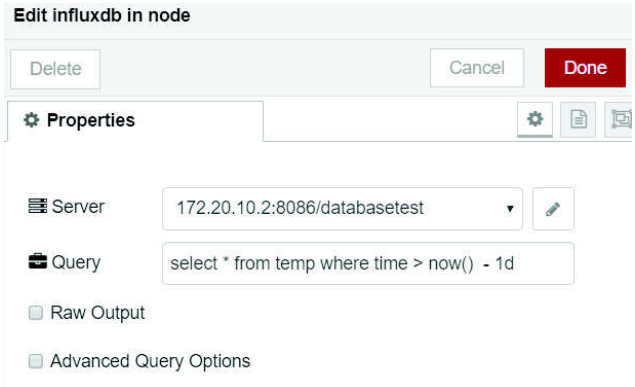


รูปที่ 13 หน้าตั้งค่าของโหนด MQTT In

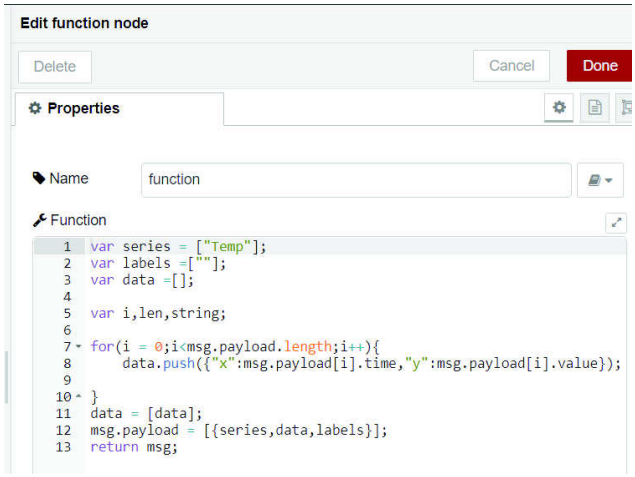


รูปที่ 14 หน้าตั้งค่าของโหนด InfluxDB Out

สำหรับแผนผังส่วนล่างโหนดที่ใช้อ่านข้อมูลจากฐานข้อมูลคือโหนด Influx In โดยตั้งค่าให้เชื่อมต่อกับฐานข้อมูลที่ 172.20.10.2:8086/databasetest ที่ได้เปิดบริการในหัวข้อที่ 4.3 และกำหนดคำสั่งในการสืบค้นข้อมูลคือ select * from temp where time > now() – 1d ซึ่งก็คือนำข้อมูลจากตาราง temp ที่มีข้อมูลเวลาในการบันทึกตั้งแต่เวลาปัจจุบันย้อนหลัง 1 วัน จนถึงเวลาปัจจุบันออกมา โดยรายละเอียดในการตั้งค่าแสดงดังรูปที่ 15



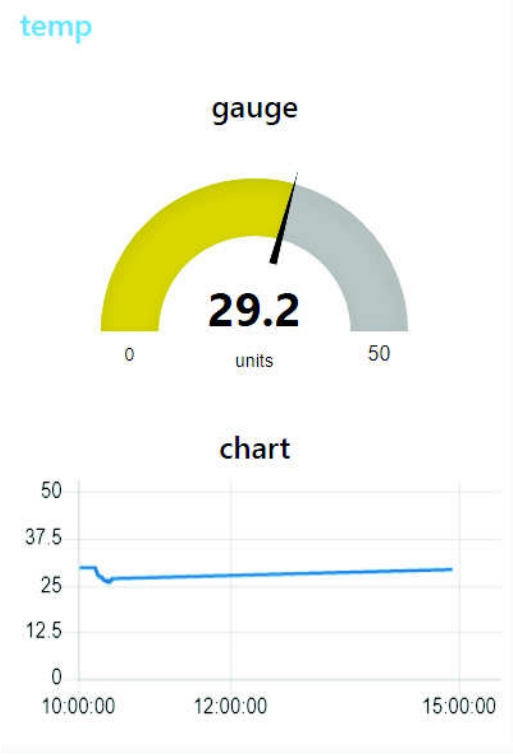
รูปที่ 15 หน้าตั้งค่าของโหนด InfluxDB In



รูปที่ 16 หน้าตั้งค่าของโหนด Function

ค่าที่ดึงออกมาจากฐานข้อมูลจะอยู่ในรูปแบบของ Object ซึ่งไม่สามารถนำไปแสดงบน Dashboard ของ Node-RED ได้โดยตรง จำเป็นต้องแปลงรูปแบบของข้อมูลให้อยู่ในรูปแบบที่เหมาะสม โดยเขียนโปรแกรมด้วยภาษา Javascript ผ่านโหนด Function ของ Node-RED ดังแสดงในรูปที่ 16

เมื่อสร้างแผนผังการทำงานของ Node-RED เรียบร้อยแล้ว จะสามารถดูข้อมูลในรูปแบบกราฟิก ผ่าน Dashboard ได้ดังรูปที่ 17



รูปที่ 17 หน้า Dashboard สำหรับแสดงข้อมูล

5. สรุปผล

บทความวิจัยนี้ได้นำเสนอการออกแบบและพัฒนาแพลตฟอร์มให้บริการ IoT Cloud บนโพรโตคอล MQTT ด้วยการออกแบบผังการทำงานผ่าน Node-RED โครงสร้างพื้นฐานของแพลตฟอร์มทำงานบน Docker และบริการที่ให้บริการคือ MQTT Broker, Node-RED และระบบฐานข้อมูล MongoDB และ InfluxDB จะอยู่ในรูปแบบ Docker Container โดยแพลตฟอร์มมีส่วนติดต่อกับผู้ใช้เป็นเว็บแอปพลิเคชันพัฒนาด้วย MERN Stack ที่ผู้ใช้สามารถเปิดใช้บริการต่าง ๆ ได้ด้วยตนเอง

จากการทดสอบพบว่าแพลตฟอร์มสามารถรองรับและช่วยเหลือผู้ใช้งานทั่วไปในการพัฒนาระบบ IoT ได้เป็นอย่างดี โดยผู้ที่ต้องการพัฒนาระบบ IoT ไม่จำเป็นต้องติดตั้งบริการต่าง ๆ ด้วยตนเอง

6. เอกสารอ้างอิง

[1] P. Datta and B. Sharma, "A survey on IoT architectures, protocols, security and smart city based applications," 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Delhi, 2017, pp. 1-5.

[2] T. Perumal, Y. L. Chui, M. A. B. Ahmadon and S. Yamaguchi, "IoT based activity recognition among smart home residents," 2017 IEEE 6th Global Conference on Consumer Electronics (GCCE), Nagoya, 2017, pp. 1-2.

[3] S. Shukla, Balachandran K and Sumitha V S, "A framework for smart transportation using Big Data," 2016 International Conference on ICT in Business Industry & Government (ICTBIG), Indore, 2016, pp. 1-3.

[4] N. Kumar, "IoT architecture and system design for healthcare systems," 2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon), Bangalore, 2017, pp. 1118-1123.

[5] NETPIE | Network Platform for Internet of Everything, [online] Available: https://netpie.io, 21 May 2019.

[6] Merkel Dirk, "Docker: Lightweight linux containers for consistent development and deployment", Linux J., vol. 2014, no. 239, March 2014.

[7] Node-RED guide, [online] Available: http://noderedguide.com, 21 May 2019.

[8] MQTT, [online] Available: http://mqtt.org, 21 May 2019

[9] InfluxDB 1.7 documentation, [online] Available: https://docs.influxdata.com/influxdb/v1.7, 21 May 2019.

[10] Dockerode, [online] Available: https://github.com/apocas/dockerode, 21 May 2019.

ประวัติผู้จัดทำ

ชื่อ-นามสกุล : นายชูพันธุ์ รัตนโภคา

อีเมล : choopanr@kmutnb.ac.th

ประวัติการศึกษา

พ.ศ. 2539 ปริญญาบัตรวิชาชีพ สาขาไฟฟ้าและอิเล็กทรอนิกส์

วิทยาลัยเทคโนโลยีอุตสาหกรรม

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

พ.ศ. 2543 วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเกษตรศาสตร์

ประวัติผู้จัดทำ

ชื่อ-นามสกุล : นายชูพันธุ์ รัตนโภคา

อีเมล : choopanr@kmutnb.ac.th

ประวัติการศึกษา

พ.ศ. 2539 ปริญญาบัตรวิชาชีพ สาขาไฟฟ้าและอิเล็กทรอนิกส์
วิทยาลัยเทคโนโลยีอุตสาหกรรม
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

พ.ศ. 2539 ปริญญาบัตรวิชาชีพชั้นสูง สาขาไฟฟ้าและอิเล็กทรอนิกส์
วิทยาลัยเทคโนโลยีอุตสาหกรรม
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

พ.ศ. 2543 อุตสาหกรรมศาสตรบัณฑิต สาขาเทคโนโลยีอิเล็กทรอนิกส์
วิทยาลัยเทคโนโลยีอุตสาหกรรม
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ