ระบบบันทึกและวิเคราะห์ข้อมูลการขายสินค้าในเครื่องจำหน่ายสินค้าอัตโนมัติ

Data Storage and Analysis System for Vending Machine

กิตติพันธ์ รัตนศรีบัวทอง

KITTIPAN RATTANASRIBUATHONG

ปกรณ์ เหล่านภาพร

PAKORN LAONAPAPORN

พงศกร ประทีปรัตน์

PONGSAGORN PRATEEPRAT

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2561

ระบบบันทึกและวิเคราะห์ข้อมูลการขายสินค้าในเครื่องจำหน่ายสินค้าอัตโนมัติ

Data Storage and Analysis System for Vending Machine

โดย

กิตติพันธ์ รัตนศรีบัวทอง

ปกรณ์ เหล่านภาพร

พงศกร ประทีปรัตน์

อาจารย์ที่ปรึกษา

อาจารย์ ชินภัทร นันทจิวากรชัย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2561

ปริญญานิพนธ์ปีการศึกษา 2561

ภาควิชา วิศวกรรมอิเล็กทรอนิกส์

คณะ วิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบบันทึกและวิเคราะห์ข้อมูลการขายสินค้าในเครื่องจำหน่ายสินค้าอัตโนมัติ

Data Storage and Analysis System for Vending Machine

ผู้จัดทำ นายกิตติพันธ์ รัตนศรีบัวทอง รหัสประจำตัว 58010104

นายปกรณ์ เหล่านภาพร รหัสประจำตัว 58010705

นายพงศกร ประทีปรัตน์ รหัสประจำตัว 58010810

ปริญญานิพนธ์นี้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

(อาจารย์ ชินภัทร นันทจิวากรชัย)

หัวข้อปริญญานิพนธ์	ระบบบันทึกและวิเคราะห์ข้อมูลการขายสินค้าในเครื่องจำหน่ายสินค้าอัตโนมัติ
นักศึกษา	นายกิตติพันธ์ รัตนศรีบัวทอง รหัสประจำตัว 58010104
	นายปกรณ์ เหล่านภาพร รหัสประจำตัว 58010705
	นายพงศกร ประทีปรัตน์ รหัสประจำตัว 58010810
ปริญญา	วิศวกรรมศาสตร์บัณฑิต
ภาควิชา	วิศวกรรมอิเล็กทรอนิกส์
ปีการศึกษา	2561
อาจารย์ที่ปรึกษาปริญญานิพนธ์	อาจารย์ชินภัทร นันทจิวากรชัย

บทคัดย่อ

โครงงานนี้นำเสนอ ระบบบันทึกการขายและวิเคราะห์สินค้าที่ขายในเครื่องจำหน่ายสินค้าอัตโนมัติ โดยระบบจะบันทึกว่ามีการสั่งสินค้าอะไร จำนวนเท่าไหร่ และเป็นเงินเท่าไหร่ มีสินค้าเหลือเท่าไหร่เมื่อเทียบกับสินค้าตั้งต้น จำนวนลูกค้าของวันนี้ และข้อมูลอื่น ๆ ตามที่ต้องการ เป็นต้น จากนั้นจะแสดงประวัติการขายสินค้าที่ขายได้ในแต่ละวันในหลายรูปแบบ และนำประวัติการขายสินค้าเหล่านั้น มาวิเคราะห์และจัดลำดับความนิยมของสินค้า ตัวเครื่องจะบรรจุสินค้าสามารถจำหน่ายสินค้า 2 ช่อง แต่ละช่องจะจำหน่ายสินค้าทีละ 1 ชิ้น มีระบบเซนเซอร์ช่วยเช็คสินค้าด้วยว่าสามารถทำให้สินค้าออกจากตัวเครื่องได้จริง ตัวเครื่องออกแบบให้มีความสะดวกในการใช้งาน ซึ่งระบบที่ผู้จัดทำออกแบบปัจจุบันสามารถรองรับการจำหน่ายได้สินค้า 4 สินค้า และสามารถแก้ไขเพื่อเพิ่มเติมจำนวนชนิดของสินค้าได้

Thesis Title	Data Storage and Analysis System for Vending Machine	
Student	Mr. Kittipan Rattanasribuathong	Student ID 58010104
	Mr. Pakorn Laonapaporn	Student ID 58010705
	Mr. Pongsagorn Prateeprat	Student ID 58010810
Degree	Bachelor of Engineering	
Program	Electronics Engineering	
Year	2018	
Thesis Advisor	Professor Chinnapat Nantajiwakornchai	

ABSTRACT

This project present about Data Storage and Analysis System for Vending Machine. System will store orders, number of orders, total payments, number of left goods, number of customers and other information. System will display the sales history of products sold each day in many forms and use sales history of goods to analyze and rank the popularity of the product. The machine can sell two products. Each channel sells one product at a time. There is a sensor system to help check products that can make the product out of the machine. The machine is designed to be convenient to use. Now our system can support 4 goods that can develop to add more goods.

กิตติกรรมประกาศ

ปริญญานิพนธ์สำเร็จลุล่วงได้ด้วยความร่วมมือกันของผู้จัดทำ โดยได้รับคำสั่งสอนและคำแนะนำจากอาจารย์ ชินภัทร นันทจิวากรชัย ในด้านโครงสร้างของเครื่องจำหน่ายสินค้าอัตโนมัติ การออกแบบเซนเซอร์ การสร้างฐานข้อมูลออนไลน์ และคุณสมบัติที่เครื่องจำหน่ายสินค้าควรจะเป็น

ผู้จัดทำขอขอบคุณคณะผู้จัดทุกคนที่ช่วยกันทำโปรเจคนี้ให้สำเร็จได้ดังที่ตั้งเป้าหมายไว้ อาจารย์ทุกท่านที่สั่งสอนและแนะนำ รวมถึงเพื่อนทุกคนที่ทำให้การทำโปรเจคนี้จบลงได้อย่างราบรื่น

กิตติพันธ์ รัตนศรีบัวทอง

ปกรณ์ เหล่านภาพร

พงศกร ประทีปรัตน์

สารบัญ

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป(ต่อ)

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญ

ในปัจจุบันการบริการต่างๆทำได้โดยอัตโนมัติไม่ว่าจะเป็นการซักผ้า การจำหน่ายสินค้า และมีระบบหลายระบบทำงานออนไลน์ ไม่ว่าจะเป็นการเงินออนไลน์ การค้าออนไลน์ ผู้จัดทำได้ทำการศึกษาการทำงานของเครื่องจำหน่าสินค้า และการใช้ข้อมูลในระบบออนไลน์ นำมาจำลองเป็นเครื่องจำหน่ายสินค้าอัตโนมัติที่มีระบบเก็บข้อมูลออนไลน์ได้

เครื่องจำหน่ายสินค้าอัตโนมัติเก็บข้อมูลการจำหน่ายสินค้าและนำข้อมูลการจำหน่ายไปวิเคราะห์เพื่อประกอบการตัดสินใจในการเลือกจำหน่ายสินค้าให้ตรงตามความต้องการของผู้บริโภคที่ใช้บริการเครื่องจำหน่ายสินค้า เพื่อให้เกิดประโยชน์สูงสุดแก่ผู้บริโภคและผู้จำหน่าย

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

1.2.1 เข้าใจการวางแผนงานและการทำงานเป็นกลุ่ม

1.2.2 การออกแบบการทำงานของเครื่องจำหน่ายสินค้า โดยใช้อุปกรณ์ทางอิเล็กทรอนิกส์ต่างๆ รวมถึงการออกแบบโครงสร้างของเครื่องจำหน่ายสินค้า

1.2.3 การออกแบบระบบฐานข้อมูลออนไลน์

1.3 สมมุติฐานการศึกษา

เครื่องจำหน่ายสินค้าสามารถนำสินค้าออกจากตัวเครื่องได้ตามที่ต้องการ สามารถตรวจสอบผลการจำหน่ายสินค้าได้อย่างถูกต้อง และสร้างข้อมูลเพื่อการประกอบการตัดสินใจการจำหน่ายได้

1.4 ขอบเขตการศึกษา

1.4.1 สามารถนำสินค้าออกจากเครื่องจำหน่ายได้โดยไม่มีข้อผิดพลาด

1.4.2 เซ็นเซอร์ที่ใช้ในการตรวจสอบผลการจำหน่ายสินค้าทำงานได้ถูกต้อง

1.4.3 จัดเก็บข้อมูลในฐานข้อมูลออนไลน์และนำข้อมูลไปวิเคาระห์ได้อย่างถูกต้อง

1.5 ประโยชน์ที่คาดว่าจะได้รับ

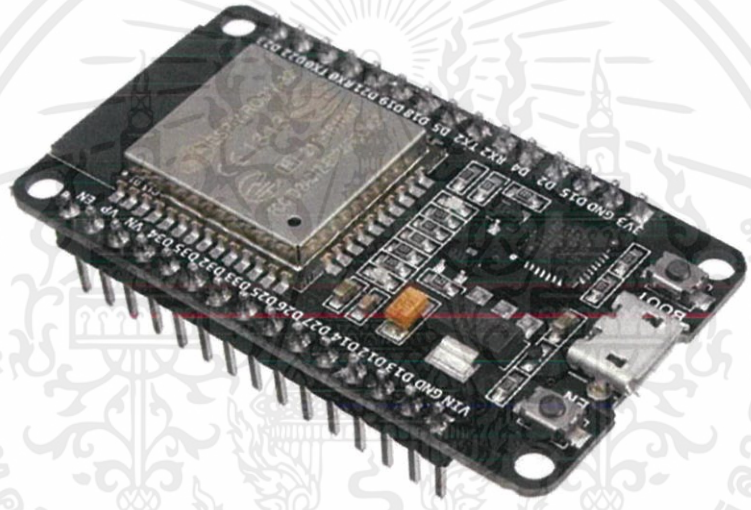
เข้าใจการรับส่งข้อมูลของฐานข้อมูลออนไลน์ การทำงานของอุปกรณ์ในระบบ IoT (Internet of thing) และการนำอุปกรณ์ทางอิเล็กทรอนิกส์มาประยุกต์ใช้ในการสร้างเครื่องจำหน่ายสินค้าอัตโนมัติจำลอง

บทที่ 2

หลักการและทฤษฎี

2.1 ESP32

ใช้ในการควบคุมการทำงานของเครื่องจำหน่ายสินค้าและรับส่งข้อมูลไปยังฐานข้อมูลออนไลน์



รูปที่ 2.1 ESP32

ESP32 เป็นชื่อของไอซีไมโครคอนโทรลเลอร์ประเภทหนึ่ง ที่รองรับการเชื่อมต่อ Wi-Fi และ Bluetooth 4.2 BLE ในตัว ผลิตโดยบริษัท Espressif ในประเทศจีน ESP32 มีคุณสมบัติดังนี้

1.หน่วยประมวลผล Tensilica LX6 Dual-Core @ 160/240MHz

2.แรมในตัว 512 kBytes

3.รอมภายนอกรองรับการเชื่อมต่อสูงสุด 16 MBytes

4.Wi-Fi มาตรฐาน 802.11 b/g/n รองรับการใช้งานโหมด Station softAP และ Wi-Fi Direct

5.Bluetooth ในตัวรองรับการทำงานในหมวด 2.0 และโหมด 4.0 BLE

6.แรงดันไฟฟ้าในการทำงาน 2.6 ถึง 3 โวลต์

7.อุณหภูมิที่ทำงานได้ -40 ถึง 125 องศาเซลเซียส

Specifications	ESP32
MCU	Xtensa® Dual-Core 32-bit LX6 600 DMIPS
802.11 b/g/n Wi-Fi	Yes, HT40
Bluetooth	Bluetooth 4.2 and below
Typical Frequency	160 MHz
SRAM	512 kBytes
Flash	SPI Flash , up to 16 MBytes
GPIO	36
Hardware / Software PWM	1 / 16 Channels
SPI / I2C / I2S / UART	4/2/2/2
ADC	12-bit
CAN	1
Ethernet MAC Interface	1
Touch Sensor	Yes
Temperature Sensor	Yes
Working Temperature	-40℃ – 125℃

รูปที่ 2.2 คุณสมบัติของ ESP32

ระบบต่างๆที่ติดตั้งมากับ ESP32 มีดังนี้ วงจรกรองสัญญาณรบกวนในวงจรขยายสัญญาณ เซ็นเซอร์แม่เหล็ก เซ็นเซอร์สัมผัส(Capacitive touch) เซ็นเซอร์อุณหภูมิ รองรับระบบการเชื่อมต่อคริสตัล 32.768 kHz. สำหรับใช้กับส่วนวงจรนับเวลาโดยเฉพาะ

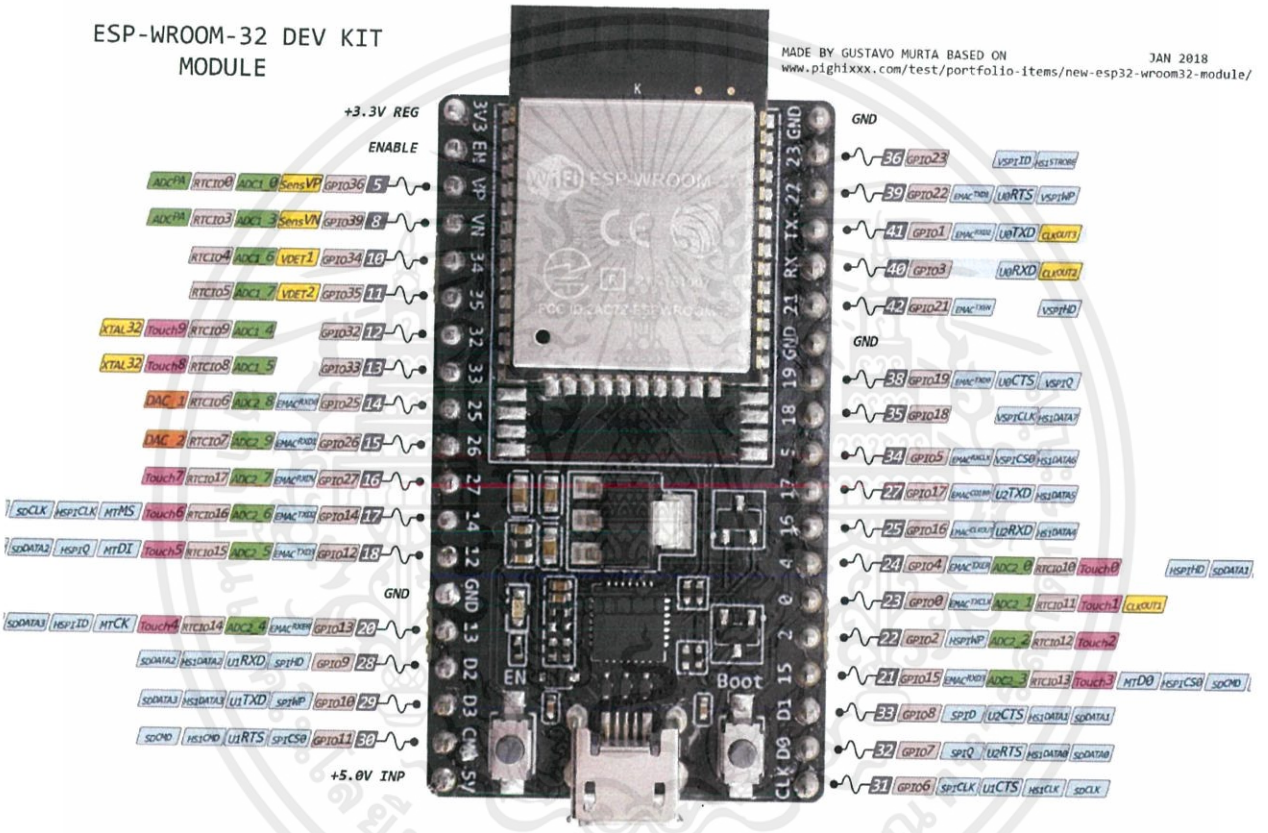
ฟังก์ชั่นเกี่ยวกับความปลอดภัยที่ ESP32 รองรับมีดังนี้ รองรับการเข้ารหัส Wi-Fi แบบ WEP และ WPA/WPA2 PSK/Enterprise มีวงจรเข้ารหัส AES/SHA2/Elliptical Curve Cryptography/RSA-4096 ในตัว

การเชื่อมต่อพอร์ทต่างๆของ ESP32 มีดังนี้

GPIO จำนวน 32 ช่อง UART จำนวน 3 ช่อง SPI จำนวน 3 ช่อง I^2C จำนวน 2 ช่อง

ADC จำนวน 12 ช่อง DAC จำนวน 2 ช่อง I^2S จำนวน 2 ช่อง PWM/Timer ทุกช่อง

SD-Card

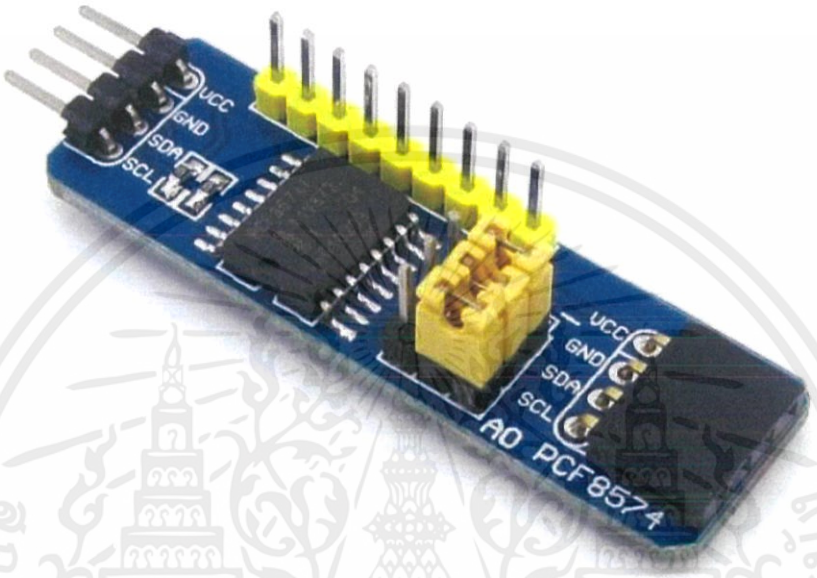


รูปที่ 2.3 พอร์ทการเชื่อมต่อของ ESP32

ด้านประสิทธิภาพของการใช้งาน ESP32 ทำงานได้ดี ในการรับ-ส่งข้อมูลได้ที่ความเร็วสูงสุด 150 Mbps เมื่อเชื่อมต่อแบบ 11n HT40 ได้ความเร็วสูงสุด 72 Mbps เมื่อเชื่อมต่อแบบ 11n HT20 ได้ความเร็วสูงสุดที่ 54 Mbps เมื่อเชื่อมต่อแบบ 11g และได้ความเร็วสูงสุดที่ 11 Mbps เมื่อเชื่อมต่อแบบ 11b การเชื่อมต่อผ่านโปรโตคอล UDP จะสามารถรับ-ส่งข้อมูลได้ที่ความเร็ว 135 Mbps ในSleep Mode ใช้กระแสไฟฟ้าเพียง 2.5 μA.

2.2 PCF8574 IO Expansion Module

ใช้ในการเพิ่มพอร์ทการรับข้อมูลของเครื่องจำหน่ายสินค้า



รูปที่ 2.4 PCF8574 IO Expansion Module

PCF8574 เป็นอุปกรณ์ที่ใช้ในการเพิ่มพอร์ทการเชื่อมต่อแบบ I^2C มีพอร์ทแบบ Pin Header และ Connector เพิ่มความสะดวกและความหลากหลายในการใช้งาน PCF8574 สามารถเชื่อมต่อกันเองได้เพื่อเพิ่มพอร์ทการเชื่อมต่อจาก 8 พอร์ท ได้มากสุดถึง 64 พอร์ท

2.3 Node-RED

2.3.1 Node-RED

Node-RED เป็น flow-based programming tool ตัวหนึ่ง สร้างขึ้นมาเพื่อทำงานร่วมกับ อุปกรณ์ Hardware, APIs, รวมถึงพวกระบบ Online Service ต่าง ๆ ซึ่งรันบน Node.js โดยตัวมันไม่ได้รองรับเฉพาะแค่ HTTP เท่านั้น มันยังรองรับพวก TCP, UDP, WebSocket, MQTT and serial ports ด้วย นอกจากนั้นยังสามารถรันบน Raspberry Pi หรือ Arduino ได้ด้วย จาก IBM's Emerging Technology Services ทีม ตั้งแต่ปี 2013 โดยคุณ Nick O'Leary กับคุณ Dave Conway-Jones และตอนนี้ก็มีอยู่ใน part ของ JS Foundation แล้ว

Node-RED สร้างบน Node.js ทำให้สามารถเชื่อมโยงกับโมดูล ที่กำเนิดบนโครงสร้างเดียวกันได้มากมาย คุณสามารถเพิ่มโมดูล Arduino โดยอาศัย Firmata ติดต่อกับฐานข้อมูลแบบ NoSQL อย่าง Mongodb ติดต่อควบคุมกับ Raspberry Pi และอื่นๆอีกมากมาย

2.3.2 Flow-Based programming (FBP)

เป็น Programming Paradiam หรือ การอธิบายถึงพฤติกรรมของ Application network แบบ "black box" หรือ โหนด โดยที่การทำงานแต่ละโหนดจะต้องทำงานได้ตามจุดประสงค์ที่วางเอาไว้

2.3.3 Browser-based flow editing

Node-RED provides browser-based flow editor ให้เราสามารถเขียน flow การทำงานร่วมกันของ application ต่าง ๆ ของเราได้อย่างง่ายดาย บน browser เลย สามารถทำงานร่วมกันกับ JavaScript function ผ่าน text editor ได้ด้วย ตัว Node-RED มี built-in library หรือ function ที่มีประโยชน์ต่าง ๆ เอาไว้ให้เราใช้อย่างเพียงพอ อีกทั้งยังมี template ที่เอาไว้ให้เราใช้ได้อีกต่างหาก

2.4 MQTT PROTOCAL

MQTT ย่อมาจาก Message Queue Telemetry Transport เป็นโปรโตคอลประยุกต์ที่ใช้โปรโตคอล TCP เป็นรากฐาน ออกแบบมาสำหรับงานที่ต้องการ ๆ สื่อสารแบบเรียลไทม์แบบไม่จำกัดแพลตฟอร์ม หมายถึงอุปกรณ์ทุกชิ้นสามารถสื่อสารกันได้ผ่าน MQTT

MQTT จะแบ่งเป็น 2 ฝั่ง คือฝั่งเซิร์ฟเวอร์มักจะเรียกว่า MQTT Broker ส่วนฝั่งผู้ใช้งานจะเรียกว่า MQTT Client ในการใช้งานด้าน IoT จะเกี่ยวข้องกับ MQTT Client เป็นหลัก โดยจะมี MQTT Broker ทั้งแบบฟรี และเสียเงินไว้รองรับอยู่แล้ว ทำให้การสื่อสารข้อมูลผ่าน MQTT จะใช้เซิร์ฟเวอร์ฟรี หรือ MQTT Broker ฟรี เหล่านั้นเป็นตัวกลาง

ลักษณะการใช้งาน MQTT อาจจะเปรียบเสมือนได้กับการใช้งานห้องแชท Line สำหรับอุปกรณ์ โดยอุปกรณ์แต่ละตัวจะมีชื่อเป็นของตนเอง มี Username Password เป็นของตัวเอง และอาจจะมีห้องลับเฉพาะของตนเอง ดังนั้นการใช้งาน MQTT ผู้เขียนจึงจะขอยกตัวอย่างของ MQTT เทียบกับห้องแชทได้ดังนี้

2.4.1 กลุ่มผู้ใช้ (User)

ใน MQTT จะแบ่งกลุ่มของผู้ใช้งานออกเป็น 2 ระดับ คือ

- ระดับสูงสุด – สามารถที่จะรับ-ส่งข้อมูลกับอุปกรณ์ หรือช่องทางใด ๆ ก็ได้ในระบบ หรือเปรียบได้กับแอดมินที่สามารถเข้าไปดูข้อความได้ทุดห้องแม้จะเป็นห้องลับก็ตาม
- ระดับทั่วไป – สามารถรับ-ส่งข้อมูลกับอุปกรณ์หรือช่องทางที่กำหนดไว้เฉพาะเท่านั้น เปรียบได้กับผู้ใช้งาน Line ที่สามารถแชทในห้องที่ตัวเองสร้างได้ หรือเป็นสมาชิกในห้อง แต่ไม่สามารถเข้าไปแชทในห้องที่ไม่ได้เป็นสมาชิก

ในการใช้งานจริง ในอุปกรณ์ต่าง ๆ ควรจะใช้งานในระดับทั่วไป เพื่อความปลอดภัยกรณีอุปกรณ์เหล่านั้นถูกแฮกแล้วไม่สร้างความเสียหายไปยังอุปกรณ์อื่น ๆ ที่อยู่ในช่องทางเฉพาะของแต่ละอุปกรณ์

2.4.2 เส้นทาง (Topic)

เส้นทาง เปรียบเหมือนกับหัวข้อ หรือห้องแชทที่ต้องการจะคุย และการคุยกันจะมีเฉพาะอุปกรณ์ที่อยู่ในห้องนั้น ๆ (Subscribe) ถึงจะสามารถได้รับข้อมูลที่มีการส่งไปในห้องนั้น ๆ ที่ถูกเรียกว่าเส้นทางเนื่องจากการใช้งานส่งข้อมูลและรับข้อมูลจะเหมือนกับเส้นทางในระบบไฟล์ เช่น /Room1/LED ซึ่งระบบเส้นทางนี้นอกจากอุปกรณ์จะสามารถรอการสนทนาในห้องตามเส้นทาง /Room1/LED ได้แล้ว ยังสามารถรอสนทนาเส้นทาง /Room1 ได้ด้วย หากเป็นการรอฟังในเส้นทาง (Subscribe) /Room1 จะหมายถึงการส่งข้อมูลใด ๆ ที่นำหน้าด้วย /Room1 เช่น /Room1/LED , /Room1/Value ผู้ที่รอฟัง.(Subscribe) /Room1 อยู่จะได้รับข้อมูลเหล่านั้นด้วย

2.4.3 คุณภาพข้อมูล (QoS)

แบ่งออกเป็น 3 ระดับดังนี้

- QoS0 – ส่งข้อมูลเพียงครั้งเดียว ไม่สนใจว่าผู้รับจะได้รับหรือไม่
- QoS1 – ส่งข้อมูลเพียงครั้งเดียว ไม่สนใจว่าผู้รับจะได้รับหรือไม่ แต่ให้จำค่าที่ส่งล่าสุดไว้ เมื่อมีการเชื่อมต่อใหม่จะได้รับข้อมูลครั้งล่าสุดอีกครั้ง
- QoS2 – ส่งข้อมูลหลาย ๆ ครั้งจนกว่าปลายทางจะได้รับข้อมูล มีข้อเสียที่สามารถทำงานได้ช้ากว่า QoS0 และ QoS1

2.4.4 การส่งข้อมูล (Publish)

การส่งข้อมลในแต่ละครั้งจะต้องประกอบไปด้วยเส้นทาง (Topic) ข้อมูล และคุณภาพข้อมูล ซึ่งการส่งข้อมูลจะเรียกว่า Publish

2.4.5 การรับข้อมูล (Subscribe)

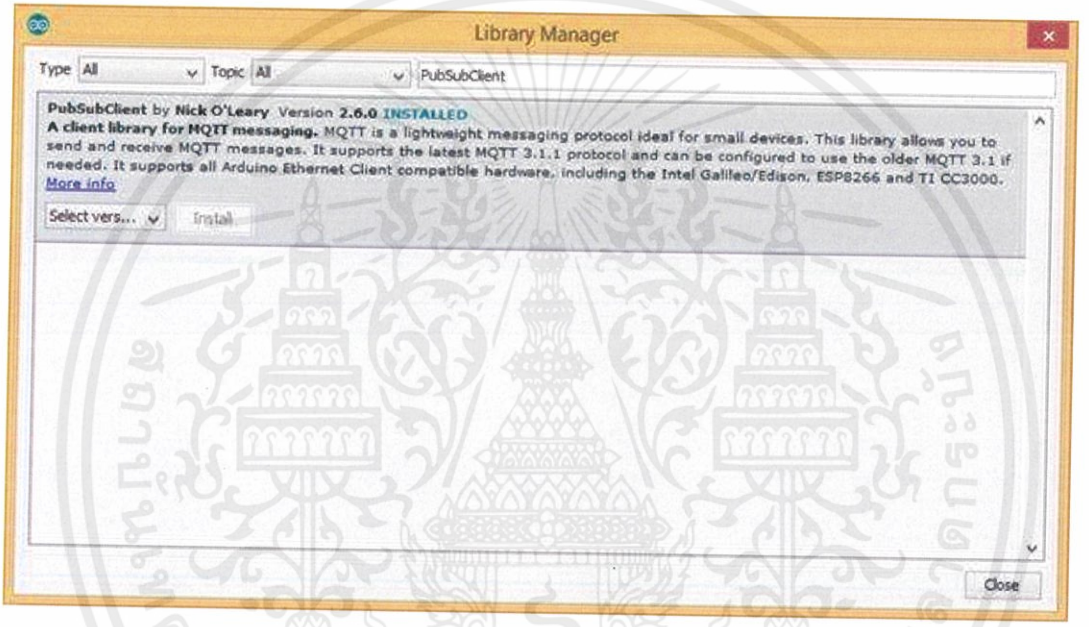
การรับข้อมูลในระบบ MQTT จะรับข้อมูลได้เฉพาะเมื่อมีการเรียกใช้การ Subscribe ไปยัง Topic ที่กำหนด อาจเปรียบได้กับการ Subscribe คือการเข้าไปนั่งรอเพื่อนในกลุ่ม Line ส่งแชทมาหา เมื่อมีการส่งข้อมูลเข้ามาจะเกิดสิ่งที่เรียกว่าเหตุการณ์ (Event) ให้เรากดเข้าไปดูข้อความที่เพื่อน ๆ ส่งเข้ามา

จะเห็นได้ว่า MQTT ก็เปลี่ยนเสมือนห้องแชทของอุปกรณ์ที่จะสนทนาแลกเปลี่ยนข้อมูลกันแบบเรียลไทม์ผ่านเครือข่ายอินเตอร์เน็ต

2.4.5 เชื่อมต่อกับ MQTT ด้วย ESP32

2.4.5.1 ฟังก์ชั่นที่เกี่ยวข้องกับการใช้งาน MQTT

การใช้งาน MQTT บน ESP32 จะใช้งานผ่านไลบารี่ PubSubClient.h จะต้องติดตั้งเพิ่มเติมโดยใช้ Library Manager ค้นหาคำว่า PubSubClient แล้วสามารถกดปุ่ม Install เพื่อติดตั้งได้เลย



รูปที่ 2.5 การเชื่อมต่อกับ MQTT ด้วย ESP32

การในการเชื่อมต่อกับ MQTT จะต้องมีการเรียกใช้ไลบารี่ PubSubClient.h ด้วยเสมอ โดยจะต้องเพิ่มคำสั่ง #include ไว้ด้านบนของไฟล์โปรแกรมเสมอ

ในการสร้างออปเจคเพื่อใช้งานคลาสของไลบารี่ PubSubClient สามารถทำได้โดยมีรูปแบบดังนี้

PubSubClient::PubSubClient(**Client** &client);

ฟังก์ชั่นย่อยที่มีให้ใช้งาน จะแบ่งตามเคสได้ดังนี้

1.เชื่อมต่อกับ Broker – ใช้ฟังก์ชั่นย่อย .setServer() เพื่อตั้งค่า Broker ที่ต้องการเชื่อมต่อ มีรูปแบบการใช้งานดังนี้

```
void PubSubClient::setServer(const char * domain, uint16_t port);
```

มีค่าพารามิเตอร์ดังนี้

- (const char *) domain – ชื่อโดเมน หรือชื่อโฮสของ Broker ที่ต้องการเชื่อมต่อ
- uint16_t port – พอร์ตของ Broker (ไม่ใช่พอร์ต SSL)

และไม่มีค่าที่ส่งกลับ

นอกจากนี้การเชื่อมต่อกับ Broker จะใช้ฟังก์ชั่นย่อย .connect() ซึ่งมีรูปแบบดังนี้

```
bool PubSubClient::connect(const char *id, const char *user, const char *pass);
```

มีรายละเอียดของค่าพารามิเตอร์ดังนี้

- (const char*) id – กำหนดชื่อที่ไม่ซ้ำกับผู้อื่นที่ใช้ Broker เดียวกัน เปรียบได้กับการตั้งชื่อใน Line ที่ไม่ซ้ำกับผู้อื่น
- (const char*) user – ชื่อผู้ใช้ที่ใช้เข้าสู่ระบบ ไม่ควรใช้ Username ของผู้ใช้ระดับสูงสุด ควรใช้ Username ที่สร้างใหม่และกำหนดสิทธิ์แล้ว
- (const char*) pass – รหัสผ่านที่ใช้เข้าสู่ระบบ

และตอบกลับมาเป็นผลการเชื่อมต่อ หากเชื่อมต่อสำเร็จจะให้ค่าเป็น True การตรวจสอบสถานะการเชื่อมต่อจะใช้ฟังก์ชั่นย่อย .connected() ในการตรวจสอบ ซึ่งมีรูปแบบการใช้งานดังนี้

bool PubSubClient::connected(); ไม่มีค่าพารามิเตอร์ และตอบกลับมาเป็นผลการเชื่อมต่อ หากกำลังเชื่อมต่ออยู่ให้ค่าเป็น True

2.4.5.2 การรับข้อมูล (Subscribe)

การรับข้อมูลเข้าจะทำงานด้วยเหตุการณ์ โดยจะมีการสร้างฟังก์ชั่นไว้ 1 ตัวเพื่อรอการเรียก เมื่อมีการส่งข้อมูลมาจะเข้าไปเรียกฟังก์ชั่นนั้นขึ้นมาใช้ ฟังก์ชั่นย่อยที่ใช้กำหนดชื่อฟังก์ชั่นที่จะถูกเรียกเมื่อเกิดเหตุการณ์ขึ้นคือ ฟังก์ชั่นย่อย. setCallback() ซึ่งมีรูปแบบการใช้งานดังนี้

void PubSubClient::setCallback(**void** (*callback)(char*, **uint8_t***, **unsigned int**));

มีค่าพารามิเตอร์ 1 ตัว คือ

- (void *) callback – ฟังก์ชั่นที่จะถูกเรียกเมื่อเกิดเหตุการณ์ขึ้น

และไม่มีค่าตอบกลับ

ฟังก์ชั่นที่ใช้รับข้อมูล (Subscribe) จากเส้นทางที่สนใจ จะใช้ฟังก์ชั่นย่อย .subscribe() มีรูปแบบการใช้งานดังนี้

void PubSubClient::subscribe(**const char** *topic);

มีค่าพารามิเตอร์ 1 ตัว คือ

- (const char*) topic – เส้นทางที่ต้องการรับข้อมูล

และไม่มีการตอบกลับ การใช้งานฟังก์ชั่นย่อย .subscribe() มักจะเรียกใช้เมื่อมีการเชื่อมต่อกับ Broker สำเร็จแล้วเท่านั้น

2.4.5.3 การส่งข้อมูล (Publish)

ใช้ฟังก์ชั่นย่อยที่ชื่อว่า. publish() มีรูปแบบการใช้งานดังนี้

bool PubSubClient::publish(**const char** *topic, **const char** *payload);

มีรายละเอียดของค่าพารามิเตอร์ดังนี้

- (const char*) topic – เส้นทางที่ต้องการส่งข้อมูล
- (const char*) payload – ข้อมูลที่ต้องการส่ง

และมีการตอบกลับเป็นข้อมูลชนิด bool ถ้าส่งข้อมูลสำเร็จจะได้ค่า True

2.4.6 MOSQUITTO

Mosquitto เป็น open source message Broker ที่ใช้งานกับ MQTT protocal โดย Mosquitto มีน้ำหนักเบา และเหมาะสมกับอุปกรณ์หลายๆแบบ ซึ่งในโครงงานนี้ เราจะใช้ Mosquitto เป็น MQTT Broker

รูปที่ 2.6 MOSQUITTO

2.5 Not Only SQL Database

2.5.1NoSQL

เทคโนโลยีฐานข้อมูลที่ถูกออกแบบมาสำหรับงานเฉพาะทางบางอย่างที่ SQL ยังไม่สามารถตอบโจทย์ได้ดีเพียงพอ เมื่อพูดถึง NoSQL จะได้ยินชื่อเว็บไซต์ที่ใหญ่ๆ ติดพ่วงมาด้วย เช่น Facebook,Twitter, FourSquare, Digg และอื่นๆ ทำให้เรารับรู้ว่า NoSQL เป็นระบบฐานข้อมูลสำหรับงานที่ต้องรองรับข้อมูลขนาดใหญ่ๆ รองรับการขยายระบบได้ง่าย เป็นต้น

ข้อดีของ NoSQL

1. NoSQL มักถูกออกแบบมาให้มี Availability สูงมาก และ Scale ระบบเพื่อรองรับผู้ใช้งานจำนวนมากได้ง่าย ถึงแม้ระบบจะทำงานร่วมกันข้าม Data Center ก็ตาม

2. NoSQL หลายๆ ระบบถูกออกแบบมาสำหรับ Unstructured Data โดยเฉพาะ เช่น ประมวลผล Log, XML, JSON และเอกสารต่างๆ ทำให้มีความยืดหยุ่นในการใช้งานเฉพาะทางแต่ละประเภทสูง

ข้อเสียของ NoSQL

1. ส่วนใหญ่แล้ว NoSQL จะทำงานแบบ Non-transactional ดังนั้นถ้าหากข้อมูลมีความละเอียดสูงและผิดพลาดไม่ได้เลย NoSQL หลายๆ ระบบก็อาจจะไม่เหมาะในหลายๆ กรณี

2. การเรียกอ่านข้อมูลขึ้นมาใช้ส่วนใหญ่มักจะมี Cost ที่สูงกว่าการใช้ SQL เพราะไม่สามารถเลือกเจาะจงได้อย่างง่ายๆ ว่าจะเรียกข้อมูลส่วนไหนขึ้นมา ยกเว้นสำหรับงานเฉพาะทางบางอย่างที่จะดีกว่า SQL แบบชัดเจนมาก (ขึ้นอยู่กับงานที่ทำและเทคโนโลยีที่เลือก) แต่การบันทึกข้อมูลลงไปส่วนใหญ่จะง่ายกว่า SQL

3. เทคโนโลยีส่วนใหญ่ไม่มีความเป็นมาตรฐานกลาง ดังนั้นการเปรียบเทียบแต่ละเทคโนโลยีค่อนข้างทำได้ยาก ผู้ใช้งานต้องมีความคุ้นเคยกับการจัดการ Software เหล่านี้ให้ได้ด้วยตัวเอง

4. ผู้เชี่ยวชาญที่สามารถสนับสนุนเทคโนโลยีเหล่านี้ในระดับองค์กรได้นั้นยังมีไม่มาก แต่เทคโนโลยี NoSQL นี้กลับมีความจำเป็นมากในการที่องค์กรจะสร้างความแตกต่างในเชิงเทคโนโลยีจากคู่แข่ง

ปัญหาของ Relational Database

Manual Sharding

การแบ่งตารางฐานข้อมูล (Table) ออกเป็นส่วนๆ แล้วทำการกระจายไปจัดเก็บในหลายๆ Server เพื่อให้แต่ละตาราง (Table) ของฐานข้อมูลไม่จัดเก็บข้อมูลที่เยอะเกินไป เพราะถ้าข้อมูลในแต่ละฐานข้อมูลเยอะเกินไปจะทำให้ระบบฐานข้อมูลช้าไปด้วย แต่ปัญหาก็จะตามมาอีกคือ เมื่อต้องกระจายข้อมูลออกไปในแต่ละ Server การ

จะจัดการกับข้อมูล เช่น เพิ่ม แก้ไข ลบ ดึงข้อมูลมาแสดงต่างๆ จะต้องทำผ่าน Application หรือไม่อย่างนั้น ก็ต้องมี Server บางตัวที่คอยดึงข้อมูลแต่ละ Server มารวมเป็นก้อนเดียว นั้นหมายความว่า เราต้องทำด้วยตัวเราเอง ไม่ใช่ระบบฐานข้อมูลจัดการให้ (Manual Sharding)

Distributed Cache

เมื่อเราต้องการให้ระบบรองรับการเข้ามาใช้งานมากๆได้นั้น ถ้าจะต้องวิ่งเข้ามาอ่านข้อมูลผ่าน Database โดยตรงคงรองรับไม่ไหว หรือทำได้ช้า ดังนั้นจะต้องมีการทำ Cache Layer ขึ้นมา คือแทนที่จะเข้าไปอ่านจากฐานข้อมูลโดยตรง ก็ให้อ่านผ่าน Cache ก่อน ดังนั้นการอ่านข้อมูลจาก Cache เป็นการอ่านจาก Memory โดยตรง ทำให้รองรับปริมาณการเข้ามาใช้งานได้มากขึ้น

แต่ปัญหาคือ การทำ Cache Layer นี้ รองรับเฉพาะการอ่านข้อมูลเท่านั้น ไม่รองรับการเขียนข้อมูลได้ ถ้าต้องการรองรับการเขียนข้อมูลปริมาณมากๆ และอ่านข้อมูลปริมาณมากๆ จึงเป็นสิ่งที่ Relational Database ไม่สามารถรองรับงานในลักษณะ อ่าน-เขียน ข้อมูลปริมาณมากๆ ได้ดีนัก และประการสำคัญ การทำ Cache Layer จะต้องมีการดูแลรักษา และใช้ Server แยกออกไปต่างหากอีกด้วย

จากจุดนี้เอง ทั้งการทำ Sharding และ Caching เป็นสิ่งที่ถูกพัฒนาขึ้นใน NoSQL เทคโนโลยี โดยรองรับ Auto-Sharding และ Integrated Caching ในตัวเอง ดังนั้นเราจึงได้เห็น NoSQL ถูกนำไปใช้งานกับระบบใหญ่ๆ เช่น Facebook,Twitter, FourSquare, Digg และอื่นๆ เพราะ NoSQL ออกแบบมาเพื่อรองรับความต้องการงานใหญ่ๆได้ดีโดยเฉพาะอยู่แล้ว แต่ถึงอย่างนั้นก็ตามยังมีคุณสมบัติอื่นๆ ที่น่าสนใจใน NoSQL เทคโนโลยี

โครงสร้างข้อมูลของ NoSQL Database 4 ชนิด ประกอบไปด้วย

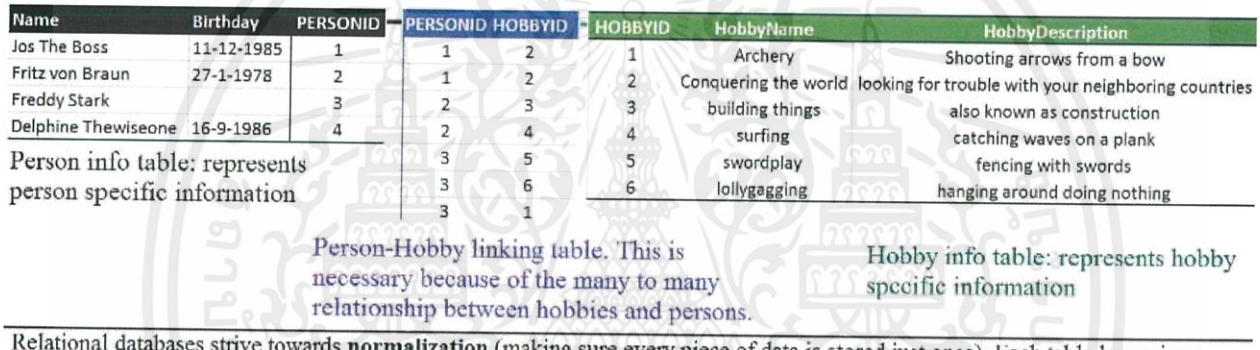
1. Column-Oriented
2. Key-value
3. Document
4. Graph

โดยแต่ละชนิดถูกคิดและสร้างขึ้นมาเพื่อแก้ไขปัญหาที่แตกต่างกันไปซึ่งปัญหาเหล่านั้นไม่สามารถแก้ไขด้วยการใช้งาน RDBMS นั่นเอง และพบว่า NoSQL หนึ่ง ๆ มักจะมีรูปแบบการจัดเก็บข้อมูลที่หลากหลาย หรือเรียกว่า Multi-model database ตัวอย่างที่เห็นได้ชัดคือ OrientDB ซึ่งมันคือ Graph database โดยที่แต่ละ node มันมีรูปแบบเป็น Document แต่ก่อนที่จะไปดูโครงสร้างข้อมูลทั้ง 4 ของ NoSQL เราต้องมาดู RDBMS หรือ Relational Database ก่อนว่ามันเป็นอย่างไร ? เพื่อจะได้เห็นข้อแตกต่างได้ชัดเจนยิ่งขึ้น เราปฏิเสธไม่ได้ว่า

RDBMS นั้นได้รับความนิยมอย่างสูง สามารถรองรับความต้องการต่าง ๆ ได้อย่างมากมาย คำหนึ่งที่เรามักจะได้ยินบ่อย ๆ ในออกแบบโครงสร้างข้อมูลบน RDBMS คือการ Normalization คือ เป็นวิธีการลดความซ้ำซ้อนของข้อมูลนั้นเอง มีโครงสร้างข้อมูลที่อยู่รูปแบบของตารางและแต่ละตารางก็จะมีความสัมพันธ์กัน เช่น

- One-to-one
- One-to-many
- Many-to-many

เมื่อระบบมีขนาดใหญ่ขึ้น จำนวนตารางก็เยอะขึ้น รวมทั้งความสัมพันธ์ต่าง ๆ ก็มากขึ้นเช่นกัน



รูปที่ 2.7 แสดงความสัมพันธ์แบบ Many-to-many ระหว่าง Person และ Hobby

2.5.2โครงสร้างข้อมูลของ NoSQL Database

2.5.2.1 Column-Oriented Database

ถ้าเปรียบเทียบกับ RDBMS จะเห็นว่ามันเป็น Row-based oriented นั่นคือแต่ละ row ของข้อมูลประกอบไปด้วย ID ที่เป็น primary key และ field หรือ column ต่าง ๆ โดยแต่ละ row จะถูกจัดเก็บในตาราง ดังนั้นในการดึงข้อมูลจากตารางจะเป็นแบบ อ่านจากบนลงล่าง และ ซ้ายไปขวา โดยข้อมูลแต่ละ row จะถูก load ไปยัง memory ซึ่งมันทำให้เสียเวลา และ ใช้ memory อย่างมากมาก แสดงดังรูป 2.8

ROWID	Name	Birthday	Hobbies
1	Jos The Boss	11-12-1985	archery, conquering the world
2	Fritz von Braun	27-1-1978	building things, surfing
3	Freddy Stark		swordplay, lollygagging, archery
4	Delphine Thewiseone	16-9-1986	

Row-oriented lookup: from top to bottom and for every entry all columns are taken in memory.

รูปที่ 2.8 Row-based oriented

เพื่อเพิ่มความเร็วในการเข้าถึงข้อมูล เราจึงทำการสร้าง index ให้ตาม column ที่เราต้องการ แต่มันเป็นการเพิ่ม overhead ให้แก่ระบบ ลองคิดดูว่า ถ้าเราทำการ index ทุก ๆ column Column-Oriented Database จึงสร้างมาเพื่อช่วยแก้ไขปัญหาเหล่านี้โดยแต่ละ column จะถูกจัดเก็บแยกกัน ทำให้การเข้าถึงข้อมูลในแต่ละ column เร็วขึ้น รวมทั้งทำให้ง่ายต่อการบีบอัดข้อมูลอีกด้วยเนื่องจากในแต่ละตารางจัดเก็บข้อมูลเพียงชนิดเดียว แสดงดังรูป

Name	ROWID
Jos The Boss	1
Fritz Schneider	2
Freddy Stark	3
Delphine Thewiseone	4

Birthday	ROWID
11-12-1985	1
27-1-1978	2
16-9-1986	4

Hobbies	ROWID
archery	1, 3
conquering the world	1
building things	2
surfing	2
swordplay	3
lollygagging	3

A column-oriented database stores each column separately

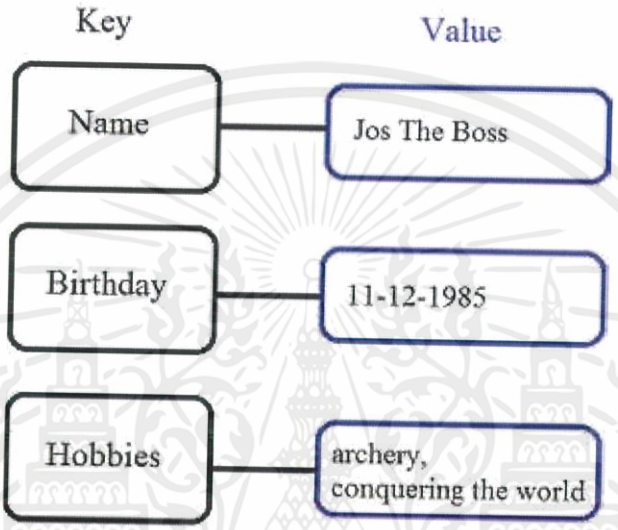
รูปที่ 2.9 การใช้ Column-Oriented Database ในการแก้ไขปัญหา

ตัวอย่าง product ที่มีโครงสร้างข้อมูลเป็น Column-Oriented เช่น

- Apache HBase
- Cassandra
- Hypertable
- Google BigTable

2.5.2.2 Key-Value Database

เป็นโครงสร้างข้อมูลที่เรียบง่าย และ ไม่ซับซ้อนที่สุดแล้ว จากความเรียบง่ายนี่เอง ทำให้ Key-Value มันสามารถรองรับการใช้งานจำนวนมากได้ สามารถรองรับข้อมูลจำนวนมากได้อย่างสบาย ๆ แสดงโครงสร้างดังรูป



รูปที่ 2.10 Key-Value Database

ตัวอย่าง product ที่มีโครงสร้างข้อมูลเป็น Key-Value เช่น

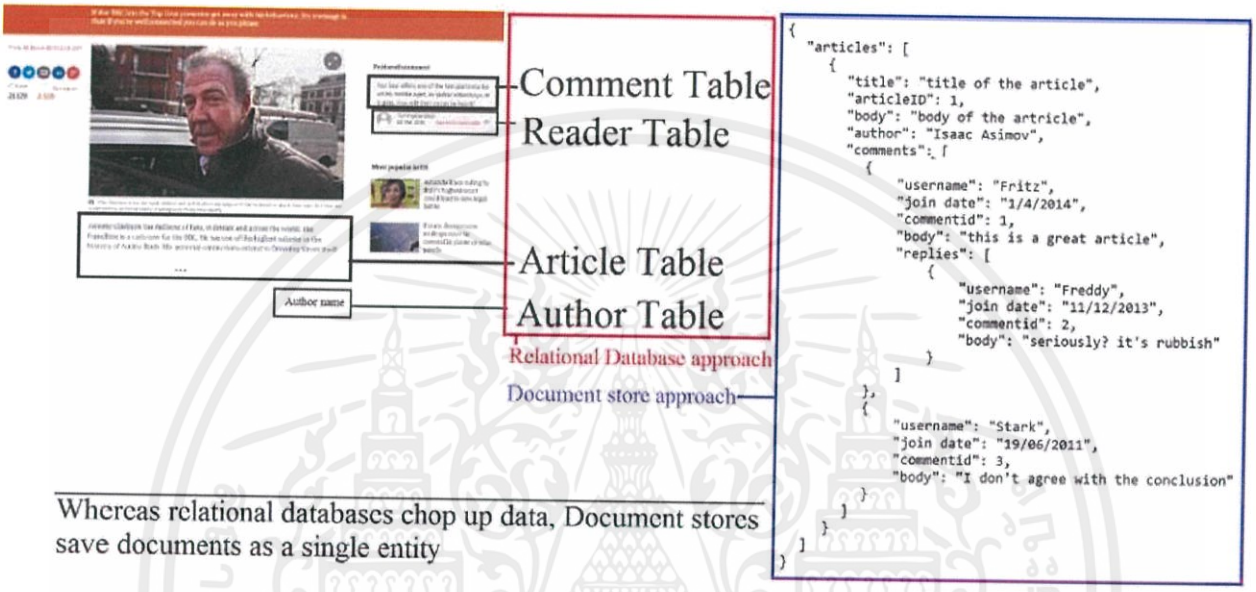
- Memcached
- Redis
- Riak
- Voldemort

Amazon Dynamo

2.5.2.3. Document Database

เป็นโครงสร้างที่ซับซ้อนมาอีกหนึ่งชั้นจาก Key-Value database ทำการเก็บข้อมูลในรูปแบบของเอกสาร แน่นอนว่า ในแต่ละเอกสารต้องมีโครงสร้างข้อมูลเช่นกัน เรามักจะเรียกว่า Schema โดยโครงสร้างแบบนี้จะถูกนำไปใช้อย่างมาก เนื่องจากข้อมูลส่วนใหญ่จะอยู่ในรูปแบบของเอกสารอยู่แล้วหรือถ้าเทียบกับ RDBMS เราอาจจะบอกว่าได้ มันคือข้อมูลที่ถูกทำการ Normalization เพียงเล็กน้อย หรือบางคนบอกว่ามันคือ การ

Denormalization นั่นเอง ทำให้ NoSQL ชนิดนี้เกิดมาเพื่อแก้ไขบางอย่างที่ RDBMS ไม่ตอบโจทย์นั่นเอง ถ้าข้อมูลของเราเป็นหนังสือพิมพ์ หรือ นิตยสารเมื่อนำมาจัดเก็บใน RDBMS แล้ว พบว่าต้องทำการแยกข้อมูลไปจัดเก็บในแต่ละตารางทั้ง ๆ ที่เราสามารถบันทึกข้อมูลในรูปแบบของเอกสารเพียงเอกสารเดียวได้ มันน่าจะช่วยลดงานต่าง ๆ ลงไปได้ แสดงดังรูป 2.11



รูปที่ 2.11 Document Database

ตัวอย่าง product ที่มีโครงสร้างข้อมูลเป็น Document เช่น

- MongoDB
- CouchDB

2.5.2.4. Graph Database

เป็นโครงสร้างข้อมูลที่มีความซับซ้อนสูงที่สุด เนื่องจากใช้จัดเก็บข้อมูลที่มีความสัมพันธ์ระหว่างกัน โดยมักจะใช้งานในเรื่องของ

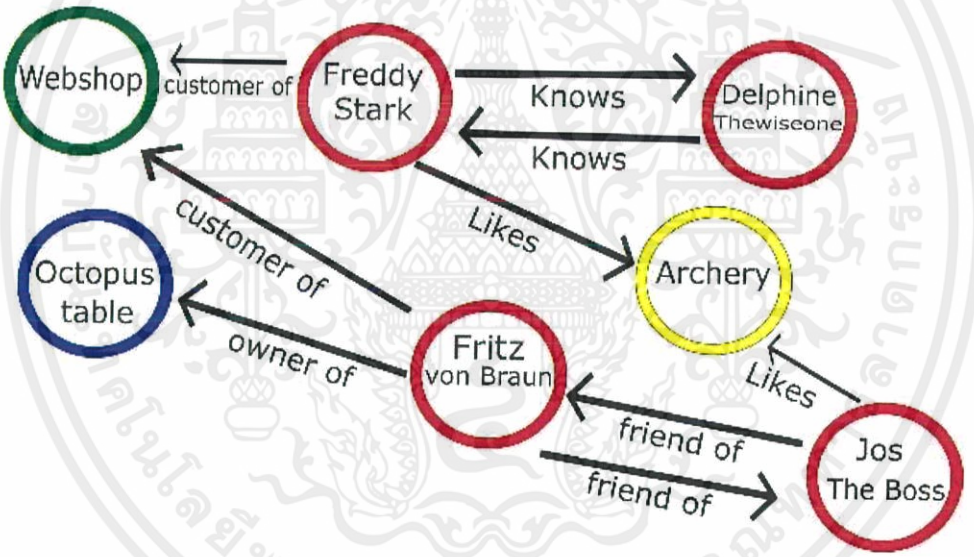
- Social Networking
- Scientific paper citation
- Capital asset cluster
- Direction in map

โครงสร้างข้อมูลแบบ Graph จะประกอบไปด้วย

- **Node** คือ ข้อมูลหรือ entity หนึ่ง ๆ ตัวอย่างเช่นใน Social Network คือ ผู้ใช้งาน
- **Edge** เป็นความสัมพันธ์ระหว่าง entity ซึ่งแสดงอยู่ในรูปแบบเส้น และ มีคุณบัติต่าง ๆ อยู่ด้วย รวมทั้งยังมีทิศทาง หรือ direction อีกด้วยแสดงดังรูป 2.12

ตัวอย่าง product ที่มีโครงสร้างข้อมูลเป็น Graph เช่น

- Neo4j
- OrientDB



รูปที่ 2.12 Graph Database

2.5.3 MongoDB

เป็น database แบบ NoSQL ตัวหนึ่ง ซึ่งแน่นอนว่า ไม่เหมือนกับ MySQL แน่ๆ แต่สิ่งที่ MongoDB หยิบยื่นให้แตกต่างจาก NoSQL ตัวอื่นบางตัว ก็คือการที่มีบริษัท เอกชนเป็น back หลังให้อยู่ ซึ่งคือบริษัท 10gen Inc (หลายตัวก็มีบริษัทสนับสนุนด้วยเหมือนกัน) เนื่องมาจาก product อะไรก็ตามที่เป็น Open Source แบบคนธรรมดาทำ จะมีความเสี่ยงอย่างหนึ่งก็คือ ใช้ๆไปแล้ว เกิดคนที่พัฒนาเบื่อ หรือเปลี่ยนความสนใจ หรือมีเหตุที่จะ

ไม่สามารถทำต่อได้อีก อนาคตเราก็ดับวูบไปเลย แต่ว่าหากมีบริษัทเป็น Backup อย่างน้อยก็มั่นใจได้ว่ามันจะยังไปต่อได้ ตราบใดที่มีบริษัทสนับสนุน และเมื่อถึงวันที่มันแข็งแกร่งพอก็จะเกิด version community ตามออกมาเอง คราวนี้ก็จะเหมือนดาวค้างฟ้า เป็น Open Source คงกระพันเหมือนอย่างที่ MySQL เคยเป็นมาก่อน

อีกเรื่องที่เป็นความแตกต่างก็คือ MongoDB ไม่ได้เรียบง่ายมาก ถึงขนาดกับเป็น database ที่เก็บได้แค่ key กับ value ที่ทำเงื่อนไขอะไรอย่างอื่นไม่ได้ แต่ว่ามันสามารถทำงานที่มีความซับซ้อน และเงื่อนไขเพิ่มขึ้นได้มากกว่า database ตัวอื่น โดยตัว MongoDB เอง ชูจุดเด่นเอาไว้ดังนี้

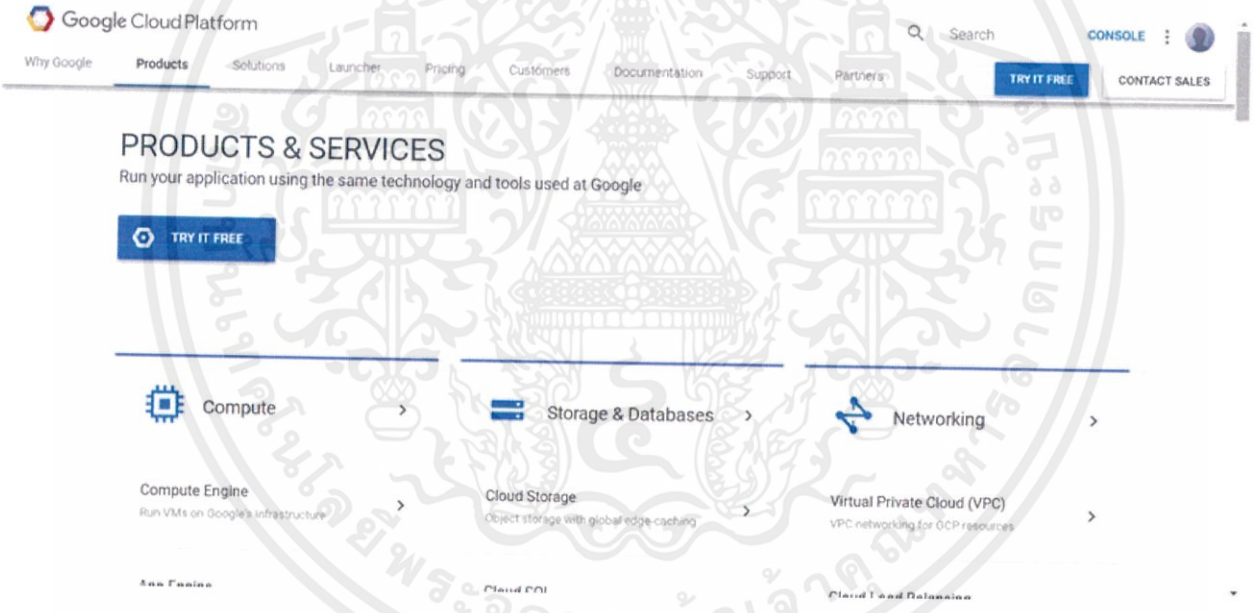
- เก็บข้อมูลแบบ Document - คือการเก็บข้อมูลในรูปแบบที่เป็น Pattern แบบมีโครงสร้าง (จะไม่แบนๆ แบบ MySQL ที่ใน table จะมี field หลายๆ field) โดยมีโครงสร้างแบบทั้งลึกและกว้าง ในแต่ละ record หากนึกไม่ออก ลองนึกถึง array แบบหลายมิติครับ นั่นล่ะครับ 1 record สามารถเก็บเป็นแบบ array หลายมิติได้ ไม่แบนราบเหมือน MySQL 1 record ที่เมื่อแปลงเป็น Array ก็ได้แค่มิติเดียวเท่านั้น
- รองรับการทำ Full Index - มีข้อดีในการ search หาได้อย่างรวดเร็วกับข้อมูลที่มีปริมาณมหาศาล (เรียกว่ามหาศาล เพราะ เยอะกว่าปกติ) และ search ได้จาก ข้อมูลใน ส่วนใดก็ได้
- รองรับ การขยายขนาด และ รองรับการทำงานหนักๆ - เพราะว่าเน้นรองรับงานหนัก และ ปริมาณข้อมูลมากๆ สามารถขยายขนาดได้อย่าวรวดเร็ว ลดข้อจำกัดต่างๆลง
- ทำระบบสำรองได้ง่าย - เราสามารถเพิ่ระบบเพื่อทำงานเป็นตัวหลัก ตัวรอง หรือว่า เป็นหลายๆตัวช่วยกันทำงาน ได้อย่างง่ายๆ ไม่ต้องตั้งค่าอะไรเยอะแยะ
- การเรียกข้อมูลมากแสดง - อย่างที่บอกว่า เป็นการเก็บข้อมูลแบบโครงสร้าง ดังนั้นเวลาเรียกข้อมูลมาแสดงก็จะได้ทั้งโครงสร้างของข้อมูลออกมาเลย
- แก้ไขข้อมูลได้รวดเร็ว - หากเราใช้ MySQL แล้วศึกษาลึกๆ จะพบว่าการ query update จะทำให้ตารางนั้น lock จังหวะที่ update แต่ว่า MongoDB ไม่เป็นอย่างนั้น
- เขียนชุดคำสั่งการทำงานได้ - หากเรามีการทำงานหลายขั้นตอน แบบซ้ำๆ เมื่อเกิดการทำงานในลักษณะแบบใด เราก็จัดกลุ่มคำสั่งที่ทำซ้ำๆแล้วเขียนเหมือนเป็น script เอาไว้เลย เวลานั้นก็รันทั้งก้อนนี้เลย
- เก็บข้อมูลด้วยระบบ GridFS - เป็นระบบการเก็บไฟล์บนพื้นที่ Harddisk ที่เก็บข้อมูลเป็นก้อนๆ และรองรับการเพิ่มหรือลดของปริมาณข้อมูลได้
- มีบริการสอบถามและดูแลเป็นพิเศษ - มี service ดูแลอย่างดีเป็นพิเศษ ให้คำปรึกษาพร้อมดูแลอย่างใกล้ชิด โดยบริษัท 10gen, Inc แต่ก็แน่นอนว่าไม่ฟรีนะสำหรับการบริการพิเศษแบบนี้

และอีกเรื่องที่ทำให้มันน่าสนใจ ก็คือ Performance ถึงแม้ว่าจะช้ากว่า NoSQL ด้วยกันเอง แต่มันก็เร็วกว่า MySQL มากๆ และข้อมูลที่เก็บก็มีความน่าเชื่อถือ ไม่สูญหายดีในระดับหนึ่ง เพราะว่า NoSQL หลายตัวที่ชูเรื่องความแรงมากๆ แต่ถ้าเครื่องดับโดยไม่คาดฝันข้อมูลสูญหาย หรือเสียหายไปเลยก็มี

2.6 Google Cloud Platform

ระบบ Cloud ในปัจจุบันถูกนำมาใช้งานอย่างแพร่หลาย ไม่ว่าจะเป็นใน iCloud, Hotmail, Gmail, Drive และรวมถึง Google Cloud Platform ของกูเกิล ก็ได้มีการนำระบบคลาวด์มาใช้เช่นเดียวกัน แต่เป็นในกลุ่มของ Cloud Platform นั่นเอง

Google Cloud Platform หรือ GCP เป็นระบบคราวด์แพลตฟอร์มที่ให้บริการลักษณะ Web Server ที่ถูกพัฒนาขึ้นโดยกูเกิล มีความสามารถในการวิเคราะห์และจัดการข้อมูล ที่ช่วยตอบโจทย์การทำงานของธุรกิจได้เป็นอย่างดี ข้อดีของบริการ Google Cloud Platform คือ ไม่ต้องทำการซื้อ Hardware เอง, มีผู้ดูแลระบบให้ตลอด 24 ชม., ค่าใช้จ่ายคิดตามจำนวนการใช้งานจริง และนอกจากนี้ยังมีบริการที่แยกย่อยออกไปอีกมากมายให้เลือกใช้งาน ไม่ว่าจะเป็น Compute Engine, Storage & Databases, Big Data, API Platform and Ecosystems, Machine Learning, Identity & Security และอื่น ๆ

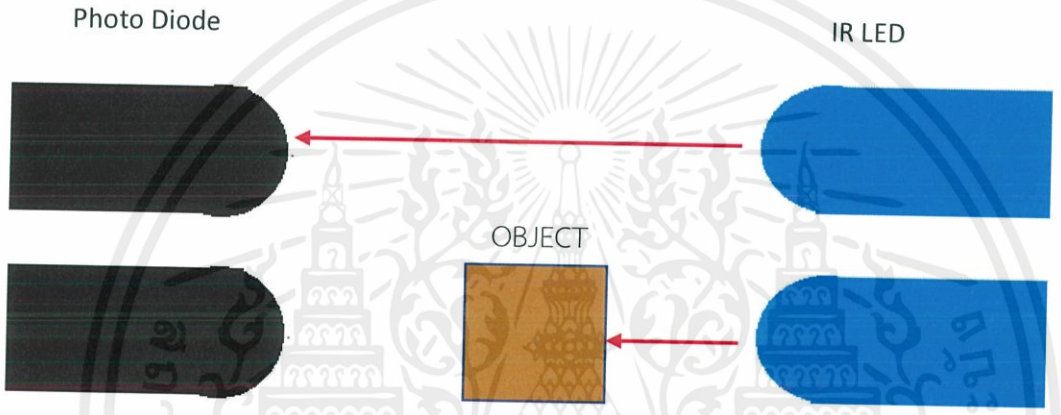


รูปที่ 2.13 Google Cloud Platform

สำหรับผู้ที่สนใจในบริการ Google Cloud Platform สามารถสมัครทดลองใช้งานได้ฟรีพร้อมรับเครดิตฟรีจำนวน $300 และในส่วนของการคิดค่าใช้จ่าย ในปัจจุบัน Google Cloud Platform ได้มีการปรับเปลี่ยนจากวิธีคิดเงินจากนาทีเป็นวินาที ในส่วนของบริการ Compute Engine, Container Engine, Cloud Dataproc และ App Engine

2.7 Infrared Sensor

เซ็นเซอร์ตรวจจับใช้ไดโอดเปล่งอินฟราเรดและโฟโต้ไดโอดให้การทำงาน โดยไดโอดเปล่งแสงอินฟราเรดปล่อยคลื่นอินฟราเรดแล้วโฟโต้ไดโอดรับคลื่นอินฟราเรด เมื่อมีวัตถุผ่านระหว่างไดโอดทั้ง 2 โฟโต้ไดโอดจะไม่สามารถรับคลื่นอินฟาเรดได้



รูปที่ 2.14 IR Sensor

2.7.1 อินฟราเรด

เป็นคลื่นแม่เหล็กไฟฟ้าที่มีความยาวคลื่นอยู่ระหว่างคลื่นวิทยุและแสงมีความถี่ในช่วง 10^{11} – 10^{14} เฮิรตซ์หรือความยาวคลื่นตั้งแต่ 1-1000 ไมโครเมตร มีความถี่ในช่วงเดียวกับไมโครเวฟ มีความยาวคลื่นอยู่ระหว่างแสงสีแดงกับคลื่นวิทยุสสารทุกชนิดที่มีอุณหภูมิอยู่ระหว่าง -200 องศาเซลเซียสถึง 4,000 องศาเซลเซียส จะปล่อยรังสีอินฟราเรดออกมา คุณสมบัติเฉพาะตัวของรังสีอินฟราเรด เช่น ไม่เบี่ยงเบนในสนามแม่เหล็กไฟฟ้า ที่แตกต่างกันก็คือ คุณสมบัติที่ขึ้นอยู่กับความถี่ คือยิ่งความถี่สูงมากขึ้น พลังงานก็สูงขึ้นด้วย

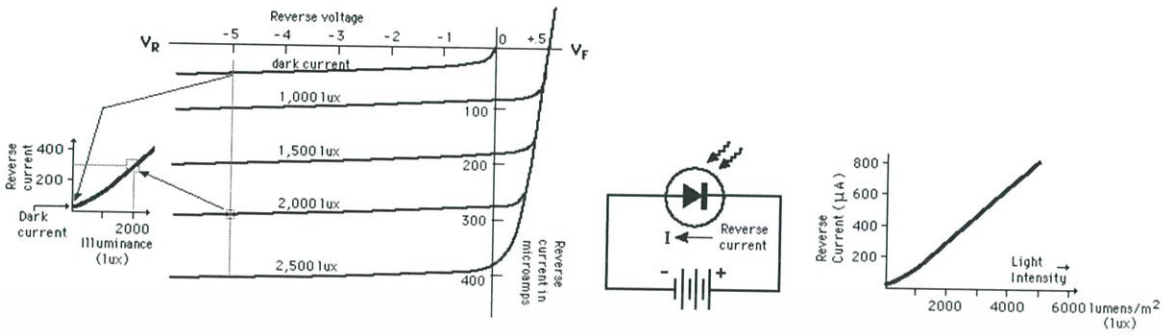
2.7.2 ไดโอดเปล่งแสงอินฟราเรด

ไดโอดเปล่งแสงอินฟราเรดหรือ IR LED เป็นอุปกรณ์สำคัญของตัวส่ง ให้แสงในช่วงคลื่นอินฟราเรด (มองด้วยตาเปล่าไม่เห็น) และให้ความเข้มแสงสูงสุดที่เฉพาะค่าความถี่เท่านั้น LED ประเภทนี้มีลักษณะเหมือน LED ทั่วไป มี 2 ขา คือ แอโนด กับ แคโทด ดังนั้นการต่อใช้งาน ก็เหมือนกรณี LED ทั่วไป LED ที่ให้แสงอินฟราเรดแต่ละชนิด สามารถทนกระแสสูงสุด (mA) ได้แตกต่างกัน

รูปที่ 2.15 ไดโอดอินฟราเรดเปล่งแสง(ซ้าย) โฟโต้ไดโอด(ขวา)

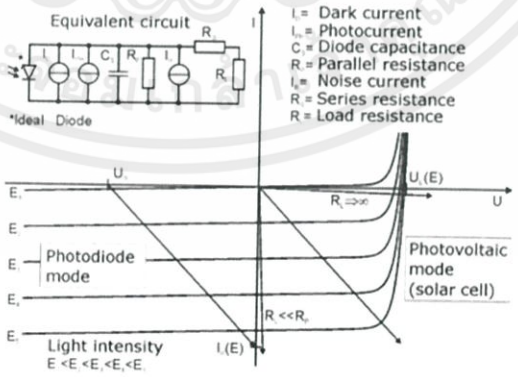
2.7.3 โฟโต้ไดโอด

โฟโต้ไดโอด (Photo diode) อุปกรณ์สารกึ่งตัวนำกระแสได้ก็เนื่องจากการให้พลังงานเพื่อดึงอิเล็กตรอนให้หลุดออกจากบอนด์ เป็นผลทำให้เกิดอิเล็กตรอนอิสระและโฮล และเมื่อให้แรงดันไฟฟ้าจะเกิดสนามไฟฟ้าในแท่งสารนั้นเป็นผลทำให้ประจุอิเล็กตรอนและโฮลเคลื่อนที่ โฟโต้ไดโอดจึงมีหลักการทำงานโดยอาศัยแสงในการเพิ่มพลังงานให้กับอิเล็กตรอนในเนื้อสารกึ่งตัวนำ ยอมให้กระแสไหลผ่านได้มากหรือน้อยนั้นขึ้นอยู่กับปริมาณความเข้มของแสง เมื่อโฟโต้ไดโอดได้รับไบอัสกลับ (Reverse Bias) ด้วยแรงดันค่าหนึ่งและมีแสงมาตกกระทบที่บริเวณรอยต่อ ถ้าแสงที่มาตกกระทบมีความยาวคลื่นหรือแลมดาที่เหมาะสมจะมีกระแสไหลในวงจร โดยกระแสที่ไหลในวงจร จะแปรผกผันกับความเข้มของแสงที่มาตกกระทบ ลักษณะทั่วไปขณะไบอัสตรง (Forward Bias) จะยังคงเหมือนกับไดโอดธรรมดาคือยอมให้กระแสไหลผ่านได้ ในขณะที่ไม่มีแสงตกกระทบจำนวนกระแสที่ไหลผ่านตัวไดโอดนี้เรียกว่า กระแสมืด (dark current)



รูปที่ 2.16 กราฟแสดงสัมพันธ์ระหว่างความเข้มแสงกับแรงดันไฟฟ้าและความเข้มแสงกับกระแสไฟฟ้าของโฟโต้ไดโอด

โหมดการทำงานของโฟโต้ไดโอดมี 2 ประเภท คือ Photovoltaic mode และ Photoconductive mode ในส่วนของโหมด Photovoltaic mode (zero bias) เมื่อไดโอดได้รับแสงที่มาตกกระทบแล้ว จะเกิดการเคลื่อนที่ของอิเล็กตรอนอิสระและโฮลอิสระ ทำให้เกิดไฟฟ้าเป็นหลักการขั้นพื้นฐานของการทำงานของเซลล์แสงอาทิตย์ ในโหมด Photoconductive mode เป็นการไบอัสย้อนกลับให้กับไดโอด ในขณะที่โฟโต้ไดโอดไม่มีแสงตกกระทบจะเกิดกระแสรั่วจากรอยต่อ PN เรียกว่า กระแสมืด (dark current) ขณะที่โฟโต้ไดโอดมีแสงมาตกกระทบที่รอยต่อ PN จะเกิดประจุอิสระที่บริเวณเขตปลอดพาหะจากกระบวนเกิดใหม่ ประจุอิสระที่บริเวณเขตปลอดพาหะจะถูกสนามแม่ไฟฟ้าจากการไบอัสย้อนกลับมากระทำต่อประจุอิสระ ทำให้ประจุอิสระเคลื่อนที่จึงสามารถนำกระแสได้ขณะที่มีแสงตกกระทบ



รูปที่ 2.17 กราฟแสดงโหมดการทำงานของโฟโต้ไดโอด

2.8 เซอร์โวมอเตอร์

ใช้ในการนำสินค้าออกจากเครื่องจำหน่ายสินค้า

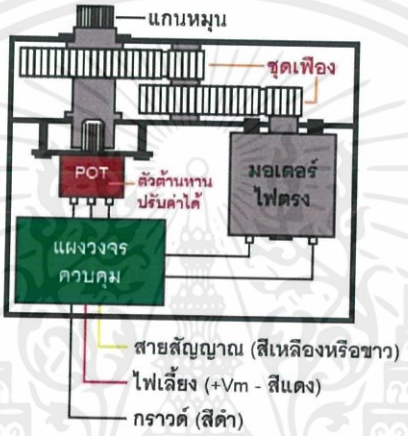
รูปที่ 2.18 เซอร์โวมอเตอร์

เซอร์โวมอเตอร์ (servo motor) เป็นอุปกรณ์ แม่เหล็กไฟฟ้าแบบหนึ่งที่ใช้ในการหมุนตัวขับ (actuator) ไปยังตำแหน่งต่างๆ ด้วยความแม่นยำ โดยใช้สัญญาณพัลส์เพื่อกำหนดตำแหน่งในการหมุน เซอร์โวมอเตอร์มีอัตราทดที่มาก เซอร์โวมอเตอร์มาตรฐานจะมีมุมในการหมุนอยู่ระหว่าง 90 ถึง 180 องศา แล้วแต่ผู้ผลิต แต่ที่นิยมมากที่สุดคือ 0 ถึง 180 องศา และในบางรุ่นของบางผู้ผลิตจะสามารถดัดแปลง ให้หมุนได้ครบ 360 องศาด้วย

ปัจจุบันเซอร์โวมอเตอร์มีด้วยกัน 2 ชนิดหลักๆ คือ ชนิดอะนาลอกและดิจิตอลรูปร่างภายนอกของเซอร์โวมอเตอร์ทั้งสองชนิดจะคล้ายกันมาก ความแตกต่างจะอยู่ที่วงจรควบคุมที่อยู่ภายใน โดยในชนิดอะนาลอกจะใช้วงจรอิเล็กทรอนิกส์ที่ประกอบด้วยอุปกรณ์ สารกึ่งตัวนำจำพวก ทรานซิสเตอร์ มอสเฟต หรือไอซีออปแอมป์เป็นหลัก ในขณะที่ชนิดดิจิตอลจะใช้ ไมโครโปรเซสเซอร์หรือไมโครคอนโทรลเลอร์เป็นตัวควบคุมหลัก

2.8.1 โครงสร้างของเซอร์โวมอเตอร์

ภายในเซอร์โวมอเตอร์ประกอบด้วย มอเตอร์ไฟตรงขนาดเล็ก,ชุดเฟืองทด, แผงวงจรควบคุม และตัวต้านทานปรับค่าได้ (POT : Potentiometer) โดยแผงวงจรควบคุมจะมีวงจรป้อนกลับ เพื่อให้เซอร์โวมอเตอร์รับรู้ตำแหน่งของตัวเองได้ โดยผู้ใช้งานเพียงส่งสัญญาณพัลส์ออกไปควบคุมเท่านั้น ดังแสดงไดอะแกรมการทำงานของเซอร์โวมอเตอร์ในรูปที่ 2.19 แกนของมอเตอร์ไฟตรงจะต่อเข้ากับ ชุดเฟืองเพื่อลดความเร็วรอบลงส่งผลให้แรงบิดที่แกนหมุนมากขึ้น



รูปที่ 2.19 ไดอะแกรมการทำงานของเซอร์โวมอเตอร์

ถ้าหากพลังงานที่จ่ายให้คงที่ เมื่อลดความเร็วรอบลงนั่นย่อมทำให้แรงบิดของมอเตอร์เพิ่มขึ้น การหมุนของมอเตอร์ได้รับการควบคุมจากวงจรควบคุม โดยมีตัวต้านทานปรับค่าได้เป็นตัวกำหนดขอบเขตของแกนหมุน ซึ่งหากไม่มีการปรับแต่งใดๆ แกนหมุนของมอเตอร์จะสามารถหมุนได้ในขอบเขต 0 ถึง 180 องศา (หรือน้อยกว่าขึ้นกับผู้ผลิต) ดังนั้นในการปรับแต่งให้เซอร์โวมอเตอร์สามารถขับแกนหมุนได้รอบตัวจึงมักจะใช้วิธีการถอดตัวต้านทานปรับค่าได้ออก แล้วแทนที่ด้วยตัวต้านทานค่าคงที่ 2 ตัว หรือดัดแปลงให้แกนหมุนของตัวต้านทานปรับค่าได้สามารถหมุนได้รอบตัว แกนหมุนของเซอร์โวมอเตอร์จะมีส่วนปลายเป็นร่องเฟือง (spline) เพื่อให้สามารถติดตั้งอุปกรณ์ที่ใช้ในการเชื่อมโยงไปยังตัวขับหรือกลไกอื่นๆ อุปกรณ์ที่ใช้เชื่อมโยงนั้นเรียกว่า ฮอร์น (horn) ซึ่งมีด้วยกันหลายรูปแบบทั้งแบบเป็นแขน, เป็นแท่ง, กากบาท, แผ่นกลม เป็นต้น สำหรับร่องเฟืองของเซอร์โวมอเตอร์แต่ละยี่ห้อก็มีจำนวนไม่เท่ากัน โดยของ Hitec จะมี 24 ร่องเฟือง ส่วนของ Futaba มี 25 ร่องเฟือง ทำให้ฮอร์นของทั้งสองยี่ห้อไม่สามารถใช้ร่วมกันได้

2.8.2 คุณสมบัติทางเทคนิคของเซอร์โวมอเตอร์

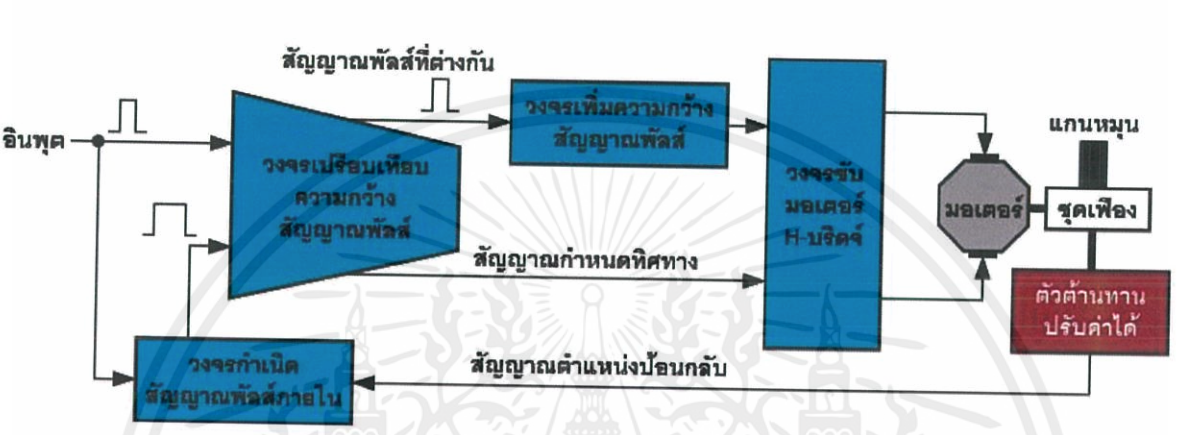
ความเร็ว (speed) และแรงบิดหรือทอร์ค (torque) ความเร็วหมายถึง ระยะเวลาที่ทำให้แกนหมุนของมอเตอร์เคลื่อนที่สู่ตำแหน่งมุมที่กำหนด อาทิ เซอร์โวมอเตอร ตัวหนึ่งมีความเร็ว 0.15 วินาที สำหรับ 60 องศา หมายถึงเซอร์โวมอเตอร์ตัวนี้สามารถขับให้แกนหมุนเคลื่อนที่ไปยังตำแหน่งมุม 60 องศาภายในเวลา 0.15 วินาที ส่วนแรงบิดมักจะปรากฏในหน่วยของออนซ์-นิ้ว (ounce-inches : oz-in) หรือ กิโลกรัม-เซนติเมตร (kg-cm) เป็นคุณสมบัติที่จะบอกต่อผู้ใช้งาน ว่าเซอร์โวมอเตอร์ตัวนี้มีแรงในการขับโหลดที่มีน้ำหนักในหน่วยออนซ์ให้สามารถเคลื่อนที่ไปได้ 1 นิ้ว หรือน้ำหนักในหน่วยกิโลกรัมให้เคลื่อนที่ ไปได้ 1 เซนติเมตร (น้ำหนัก 1 ออนซ์เท่ากับ 0.028 กิโลกรัมโดยประมาณ หรือ 1 กิโลกรัม เท่ากับ 35.274 ออนซ์)

อย่างไรก็ตาม ค่าของความเร็วและแรงบิด ต้องสัมพันธ์กับแรงดันไฟเลี้ยงที่จ่ายให้แก่เซอร์โวมอเตอร์ด้วย ซึ่งมักจะแรงดัน 4.8 หรือ 6V นอกจากนั้นยังมีปัจจัยเกี่ยวกับแรง เสียดทานในระบบเฟืองภายในเซอร์โวมอเตอร์ การหล่อลื่นการเชื่อมโยงระหว่างเฟืองต่อเฟืองในชุดเฟืองทด ที่ส่งผลให้ความเร็วและแรงบิดของ เซอร์โวมอเตอร์เปลี่ยนแปลงไปได้

2.8.3 การทำงานของแผงควบคุมในเซอร์โมเตอร์

การหมุนของเซอร์โวมอเตอร์นั้นจะไม่ได้หมุนเป็นอิสระเหมือนมอเตอร์ทั่วๆ ไปโดยช่วงระยะการหมุนปกติจะอยู่ระหว่าง 90 ถึง 180 องศา ตำแหน่งการหมุนของแกนมอเตอร์ใน เซอร์โวมอเตอร์นี้สามารถควบคุมได้อย่างแม่นยำ เนื่องจากภายในเซอร์โวมอเตอร์มีวงจรอิเล็กทรอนิกส์ทำหน้าที่ตรวจสอบตำแหน่งของเซอร์โวมอเตอร์อยู่ตลอดเวลา ลักษณะการตรวจสอบจะใช้การป้อนกลับค่าตำแหน่งจากตัวต้านทานปรับค่าได้ แล้วนำค่านี้ไป เปรียบเทียบกับค่าพัลส์ที่ป้อนเข้าทางขาควบคุม ค่าของผลต่างที่ได้จะไปปรับตำแหน่งของมอเตอร์ ค่าผลต่างก็จะได้ตำแหน่งของมอเตอร์ที่แม่นยำ ในรูปที่ 2.20 แสดงไดอะแกรมการทำงานของแผงวงจรควบคุมในเซอร์โวมอเตอร์ชนิดอะนาลอก สัญญาณพัลส์ควบคุมที่ส่งเข้ามาทางอินพุต จะถูกส่งไปยังวงจรกำเนิดสัญญาณพัลส์ภายในด้วย โดยมีความกว้างที่เป็น สัดส่วนกับตำแหน่งของแกนหมุนในปัจจุบัน ทั้งสัญญาณพัลส์ที่กำเนิดขึ้นภายในกับสัญญาณพัลส์ควบคุมจะถูกส่งไปยังวงจรเปรียบเทียบเพื่อทำการหักล้างสัญญาณ โดยทิศทางของสัญญาณจะขึ้นอยู่กับว่า ระหว่างสัญญาณพัลส์ควบคุมทางอินพุตกับสัญญาณพัลส์ภายใน สัญญาณพัลส์ใดมีความกว้างมากกว่า โดยเอาต์พุตที่ได้เป็นสัญญาณลอจิก “0” หรือ “1” แล้วส่งไปยังวงจรขับมอเตอร์แบบ H-บริดจ์ เพื่อกำหนดทิศทางการหมุน ด้านค่าความแตกต่างที่เกิดขึ้นระหว่างพัลส์ทั้งสองสัญญาณจะถูกส่งไปยังวงจรเพิ่มความกว้างพัลส์

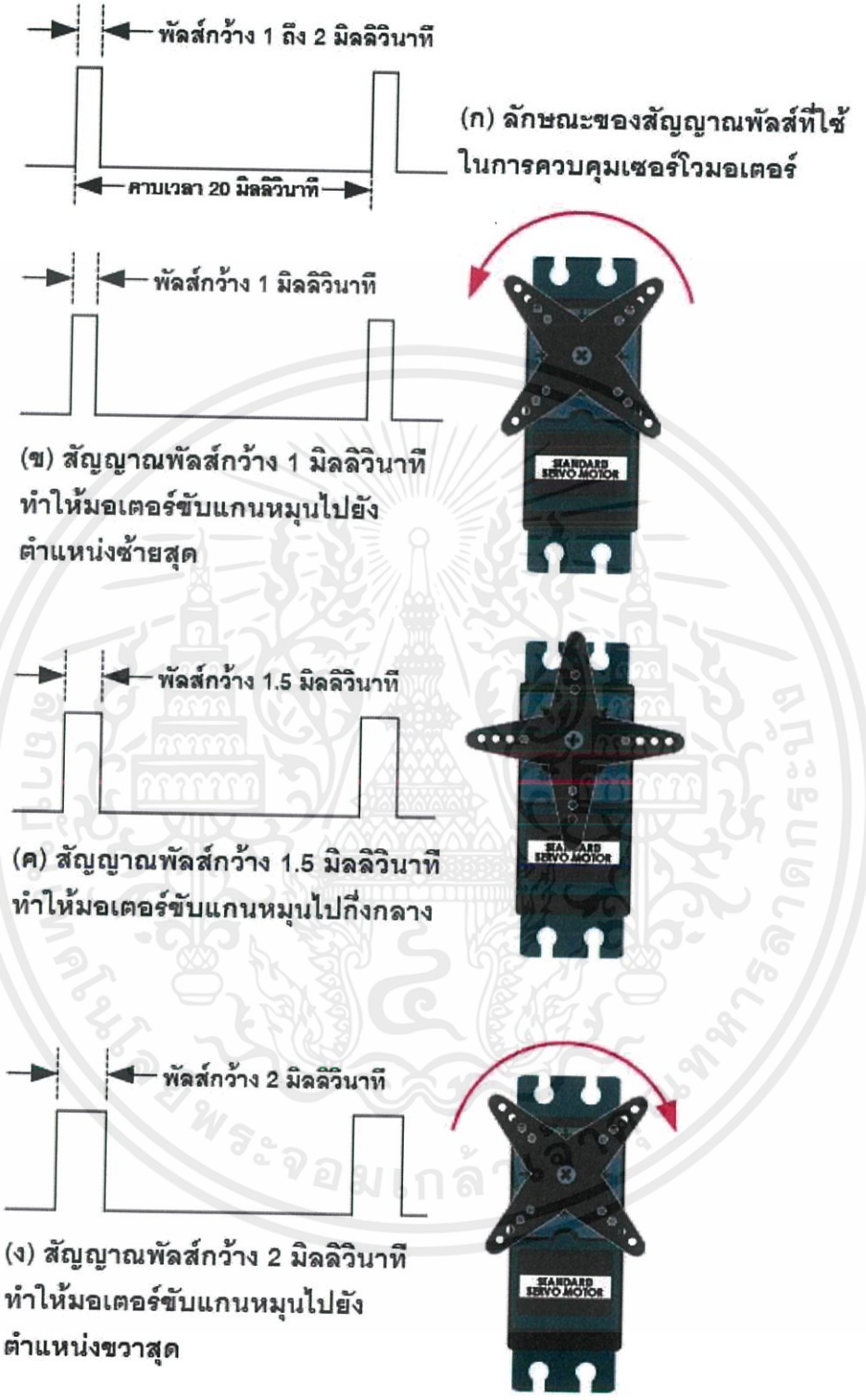
เพื่อสร้างสัญญาณพัลส์สำหรับส่งไปขับมอเตอร์ ผ่านวงจรขับมอเตอร์แบบ H-บริดจ์ โดยความแตกต่างของความกว้างพัลส์ 1% ทำให้เกิดสัญญาณพัลส์สำหรับขับมอเตอร์ในระดับ 50% และความเร็วนี้จะลดลงเมื่อแกนหมุนของมอเตอร์เคลื่อนที่เข้าสู่ตำแหน่งที่กำหนด อันเป็นผลมาจากความแตกต่างของความกว้างสัญญาณพัลส์เริ่มลดลง และหยุดลงเมื่อสัญญาณพัลส์ที่นำมาเปรียบเทียบมีค่าความกว้างเท่ากัน



รูปที่ 2.20 ไดอะแกรมการทำงานของแผงวงจรควบคุมในเซอร์โวมอเตอร์ชนิดอะนาลอก

2.8.4 การควบคุมเซอร์โวมอเตอร์

การควบคุมเซอร์โวมอเตอร์ทำได้โดยสร้างสัญญาณพัลส์ที่มีคาบเวลา 20 มิลลิวินาทีป้อนให้กับวงจรควบคุมภายในเซอร์โวมอเตอร์ดังรูปที่ 2.21 แล้วปรับความกว้างของพัลส์ช่วงบวก ที่พัลส์กว้าง 1 มิลลิวินาที มอเตอร์จะหมุนไปตำแหน่งซ้ายมือสุด ถ้าส่งพัลส์กว้าง 1.5 มิลลิวินาที แกนหมุนของมอเตอร์จะเคลื่อนที่ไปยังตำแหน่งกึ่งกลาง และถ้าส่งพัลส์กว้าง 2 มิลลิวินาที แกนหมุนของมอเตอร์จะเคลื่อนที่ไปยังตำแหน่งขวามือสุด การป้อนสัญญาณพัลส์ที่มีคาบเวลาช่วงบวกตั้งแต่ 1.5 ถึง 2 มิลลิวินาทีจะทำให้เซอร์โวมอเตอร์หมุนทวนเข็มนาฬิกา โดยถ้าค่าความกว้างพัลส์ยิ่งห่างจาก 1.5 มิลลิวินาที มากเท่าใดความเร็วในการหมุนก็จะมากขึ้นเท่านั้น นั่นคือ ความเร็วสูงสุดของการหมุนทวนเข็มนาฬิกาจะเกิดขึ้นเมื่อสัญญาณพัลส์ควบคุมมีความกว้าง 2 มิลลิวินาที การป้อนสัญญาณพัลส์ที่มีคาบเวลาช่วงบวกตั้งแต่ 1 ไปจนถึง 1.5 มิลลิวินาที ทำให้เซอร์โวมอเตอร์หมุนตามเข็มนาฬิกา ซึ่งถ้าค่าความกว้างพัลส์เข้าใกล้ 1 มิลลิวินาทีความเร็วในการหมุนของเซอร์โวมอเตอร์ก็จะมาก นั่นคือ ความเร็วสูงสุดของการหมุนตามเข็มนาฬิกาจะเกิดขึ้นเมื่อสัญญาณพัลส์ควบคุมมีความกว้าง 1 มิลลิวินาที

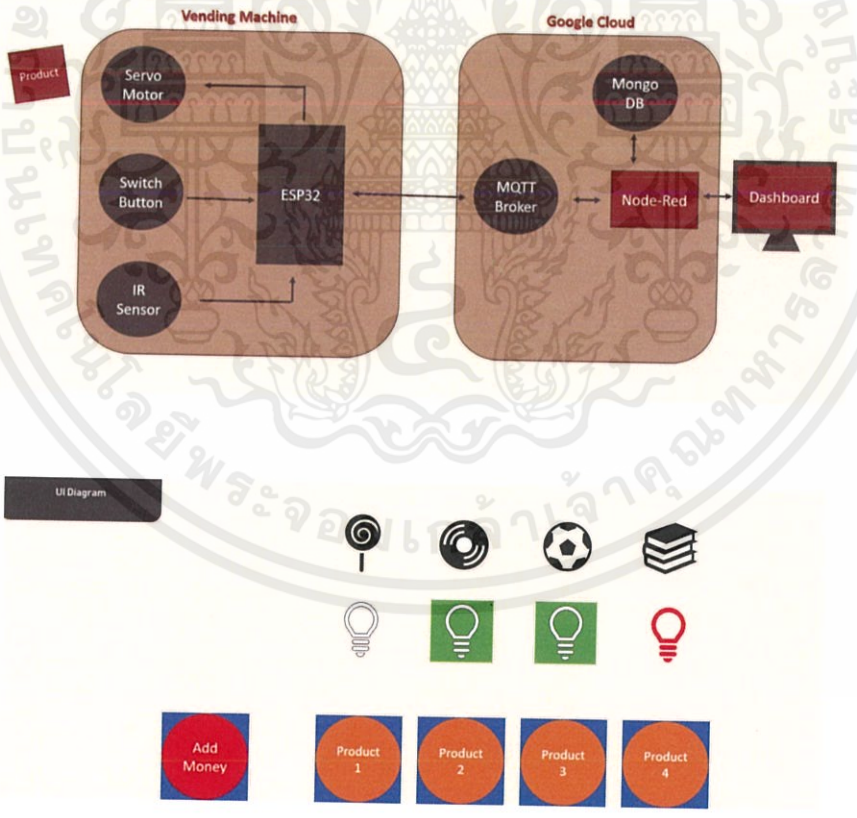


รูปที่ 2.21 แสดงลักษณะของสัญญาณพัลส์ที่ใช้ในการควบคุมเซอร์โวมอเตอร์

บทที่ 3

วิธีการดำเนินงาน

การทำงานของเครื่องจำหน่ายสินค้าควบคุมด้วย ESP32 ด้วยภาษา C++ ใช้ ESP32 ส่งข้อมูลผ่าน MQTT Broker ใช้ Node-RED ติดต่อกับ MongoDB และส่วนของหน้าเว็บ เพื่อสร้างฐานข้อมูลออนไลน์ในการเก็บประวัติการจำหน่ายสินค้า โดยฐานข้อมูลออนไลน์นั้นอยู่บน Google Cloud ระบบจะรับข้อมูลจากสวิทช์ปุ่มกดบนเครื่องจำหน่ายสินค้า โดยมีปุ่มเพิ่มเงินและปุ่มเลือกซื้อสินค้า มีไดโอดแปล่งใช้แสดงสถานะของสินค้า สีเขียว คือ เงินพอที่จะซื้อสินค้า สีแดง คือ สินค้าหมด ไดโอดแปล่งแสงไม่ติด คือ เงินไม่พอซื้อสินค้า เมื่อกดปุ่มที่มีไฟสีเขียวติด เครื่องจะจำหน่ายสินค้าออก



รูปที่ 3.1 Block Diagram การทำงานทั้งหมด(บน) หน้าแสดงผลของเครื่องจำหน่ายสินค้า(ล่าง)

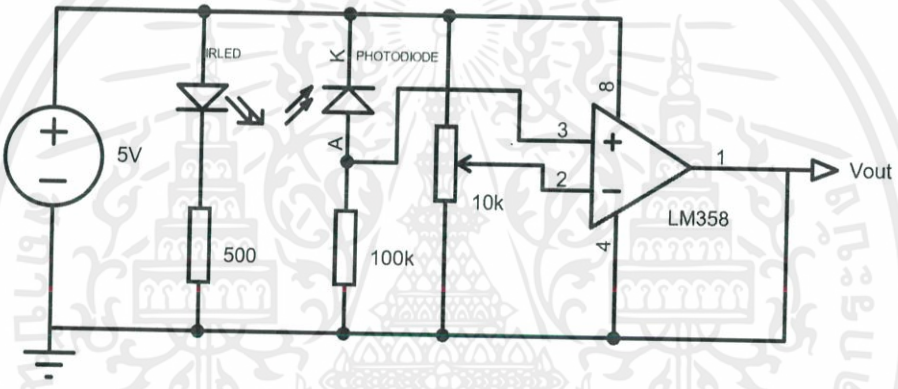
3.1 การออกแบบโครงสร้างของเครื่อง

ในการออกแบบเครื่องจำหน่ายสินค้าอัตโนมัติ โครงสร้างของเครื่องประกอบจากไม้ เนื่องจากออกแบบเพื่อใช้ในการจำลองการทำงาน โดยจะออกแบบดังรูปที่ 3.2 มีหลักการทำงานโดยใช้มอเตอร์ดันกล่องสินค้าจากด้านล่างเพื่อให้กล่องสินค้ายกขึ้นเหนือคานที่กั้นด้านหน้า กล่องสินค้าจะตกลงไปในช่องว่างด้านหน้าเพื่อจำหน่ายแก่ผู้ใช้งาน โดยช่องว่างด้านหน้านั้นจะติดเซนเซอร์เพื่อตรวจสอบการจำหน่วยสินค้าของเครื่อง ใช้สวิทช์และไดโอดเปล่งแสงแทนปุ่มกดซื้อ ปุ่มเลือกสินค้า และไฟแสดงสถานะสินค้า

รูปที่ 3.2 เครื่องจำหน่ายสินค้าอัตโนมัติ

3.2 การออกแบบการทำงานของเซ็นเซอร์

เซ็นเซอร์ที่ใช้ในเครื่องเป็นเซ็นเซอร์ตรวจจับวัตถุ ใช้อินฟราเรดในการตรวจจับวัตถุ สามารถกำหนดระยะที่สามารถทำการตรวจจับวัตถุได้โดยการเปลี่ยนแปลงค่าของตัวต้านทานปรับค่าได้ และใช้วงจรเปรียบเทียบแรงดันโดยใช้ออปแอมป์ เพื่อให้ได้แรงดันเอาพุทที่ออกจากออปแอมป์มีค่าเพียงสองค่าเท่านั้น เมื่อเอาพุทของออปแอมป์มีเพียงทำให้สามารถนำไปใช้งานเป็นดิจิตัลอินพุทของไมโครคอนโทลเลอร์ในการตวรจจับสินค้า

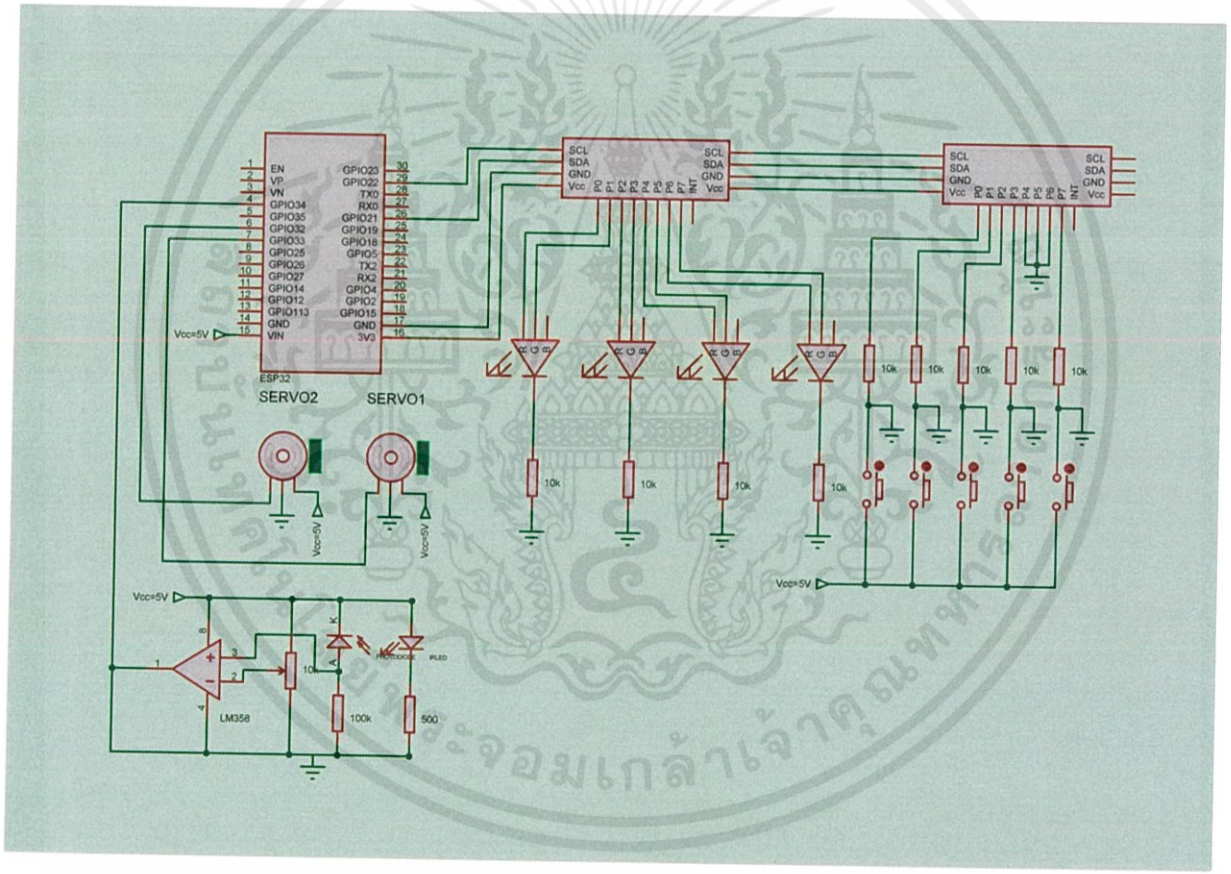


รูปที่ 3.3 วงจรตรวจจับวัตถุด้วยอินฟราเรด

3.3 การออกแบบวงจรรวมของเครื่องจำหน่ายสินค้าอัตโนมัติ

ในการออกแบบวงจรของเครื่องจำหน่ายสินค้าอัตโนมัติ ใช้ไฟเลี้ยงวงจร 5 โวลท์ เพื่อง่ายต่อการเชื่อมต่อกับแหล่งจ่ายไฟ 5 โวลท์ เช่น พาวเวอร์แบงค์ หรืออะแดปเตอร์ชาร์จโทรศัพท์เคลื่อนที่ เป็นต้น

โดยใช้หลอด RGB LED ในการจำลองการแสดงสถานะของสินค้า ปุ่มกดจำลองการจ่ายเงินของผู้ใช้บริการเครื่องจำหน่ายสินค้าอัตโนมัติ และเซ็นเซอร์ตรวจสอบว่าสินค้านั้นออกมาจากเครื่องจำหน่ายหรือไม่



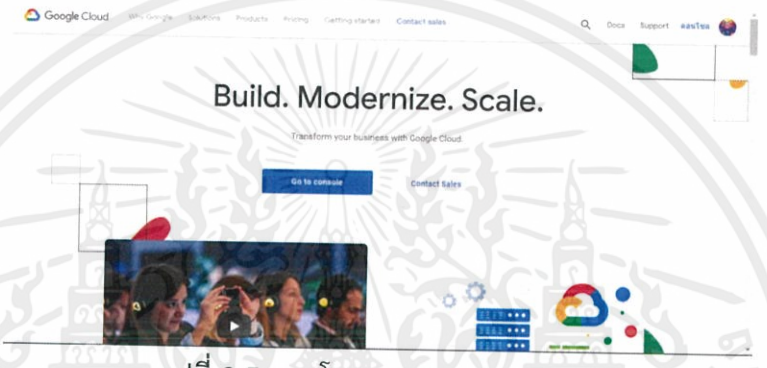
รูปที่ 3.4 วงจรรวมของเครื่องจำหน่ายสินค้าอัตโนมัติ

3.4 การออกแบบคลาวด์และการทำงานของเครื่อง

3.4.1 สร้างคลาวด์

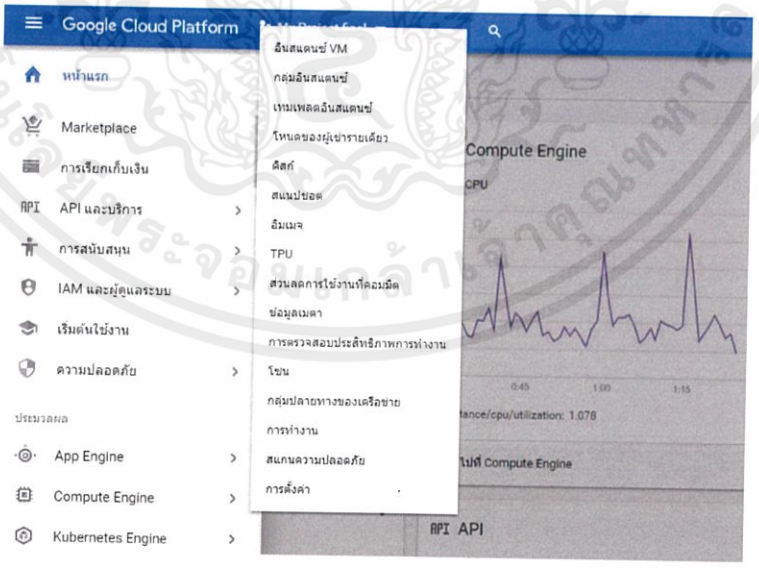
สำหรับใช้ติดตั้ง node.js , node red , Mosquitto , mongodb

1. เข้าเว็บ https://cloud.google.com/
2. เข้าสู่หน้า คอนโซล



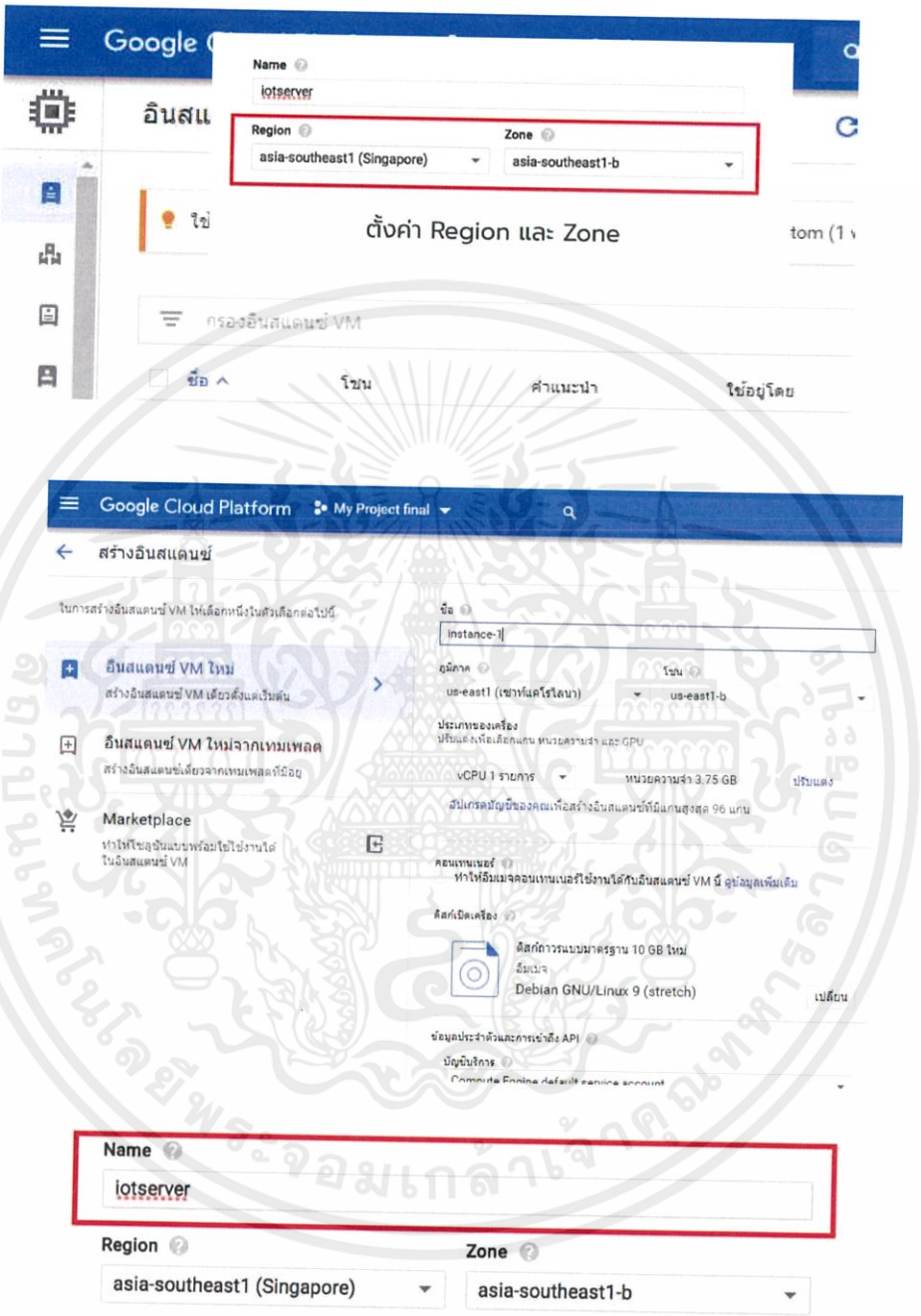
รูปที่ 3.5 คอนโซลของ Google Cloud

3. ไปที่แท็บเมนู เลือก compute engine เลือก VM instance ตามรูป 3.5

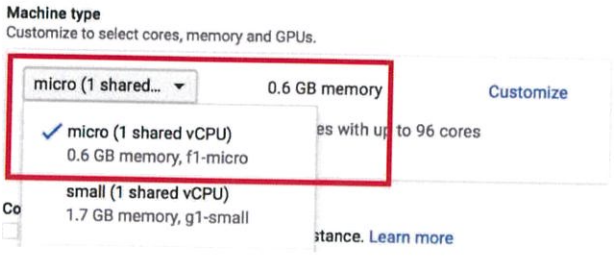


รูปที่ 3.6 VM Instance

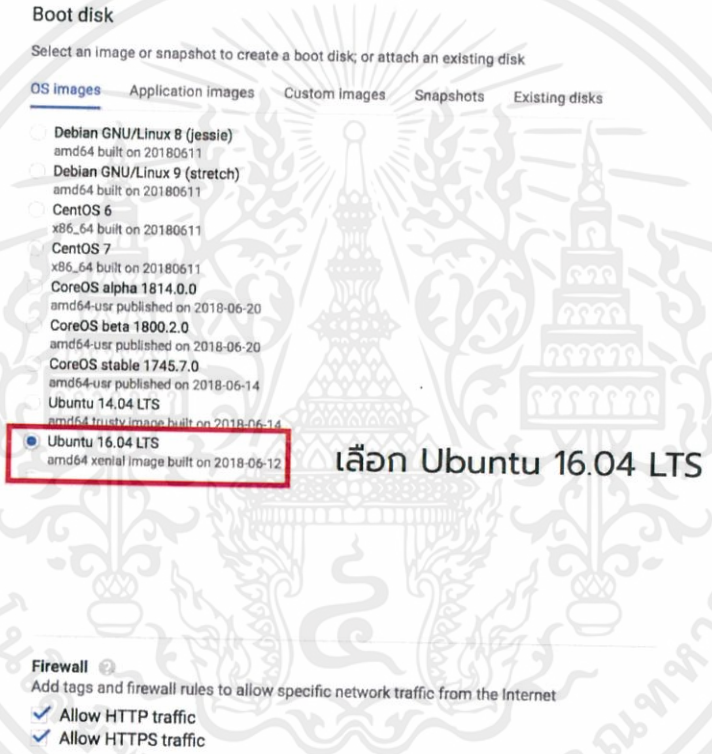
4. คลิก สร้าง vm instance



รูปที่ 3.7 การสร้าง VM Instance



เลือกชนิด micro เพื่อทดสอบใช้งานในราคาต่ำสุด
(แนะนำ small ถ้าใช้สำหรับ node-red ในระยะยาว)

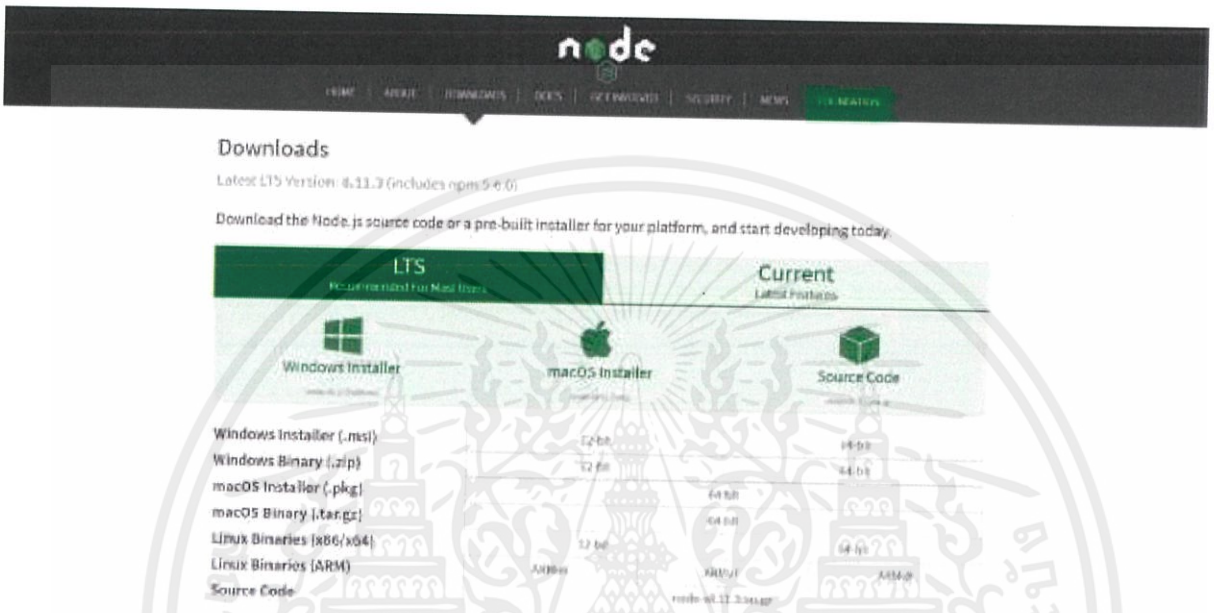


ในเบื้องต้นเราเลือกอนุญาตให้ HTTP, HTTPS ผ่านเข้ามาในระบบ
ก่อนส่วนของ PORT และ Protocol อื่นๆ สามารถสร้างได้ในภายหลัง

รูปที่ 3.8 การสร้าง VM Instance(ต่อ)

3.4.2 ขั้นตอนการติดตั้ง node-red

1.ติดตั้ง NodeJS https://nodejs.org/en/download/

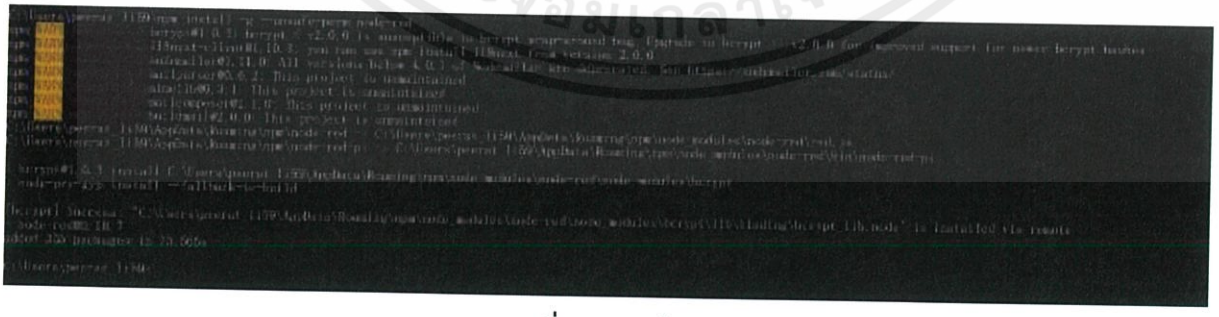


รูปที่ 3.9 เว็ปไซต์ของ Node-RED

2.เปิดหน้า CMD แล้วพิมพ์ชุดคำสั่ง

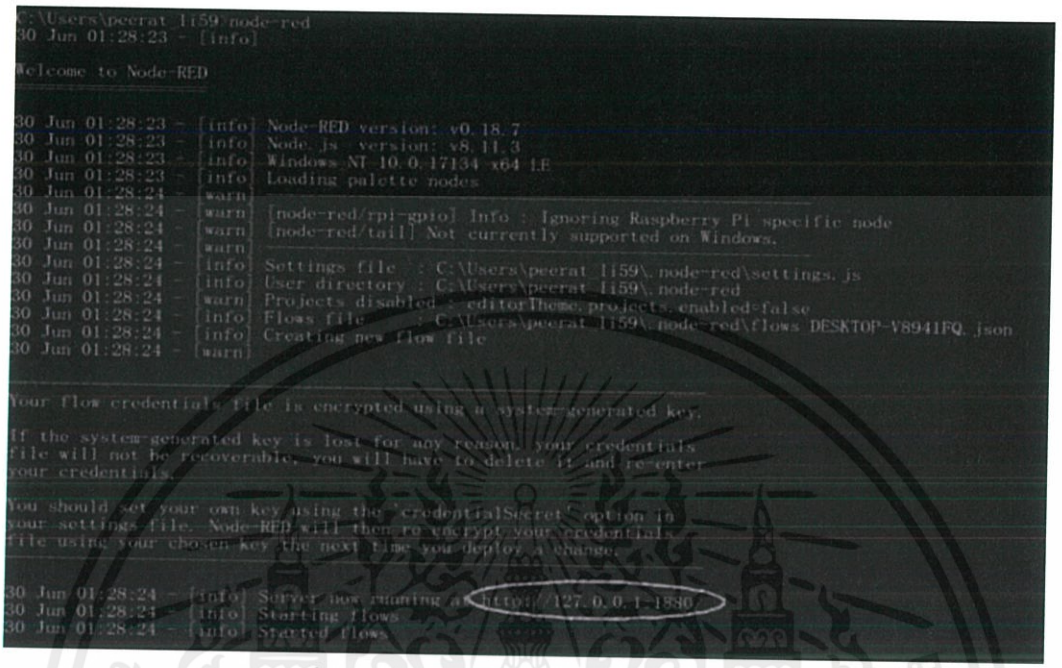
สำหรับ MAC/Linuxsudo npm install -g --unsafe-perm node-red

สำหรับ Windows npm install -g --unsafe-perm node-red



รูปที่ 3.10 หน้า CMD

3. รันคำสั่ง node-red ในหน้า CMD



รูปที่ 3.11 รันคำสั่ง Node-RED ในหน้า CMD

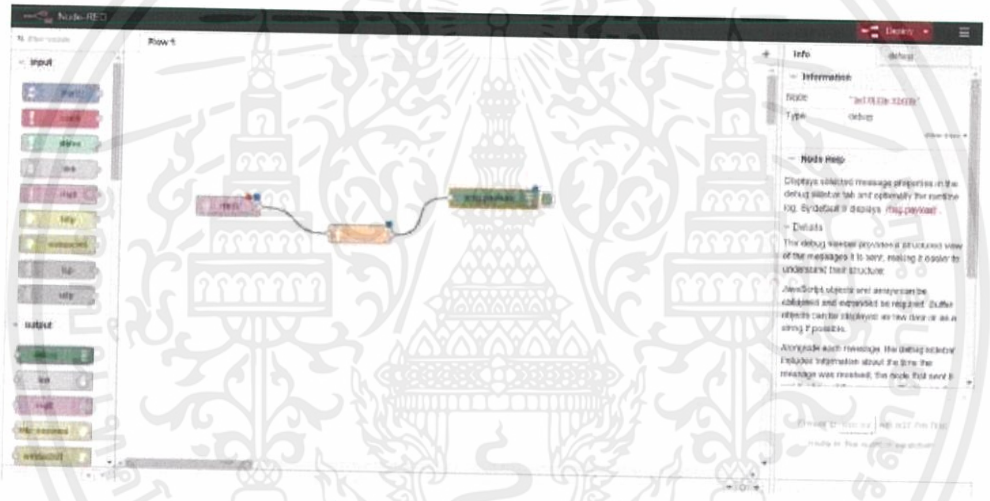
4. เข้าเว็บของเราพิมพ์ URL: http://127.0.0.1:1880 Node-RED พร้อมใช้งาน

3.4.3 ใช้งาน Node-RED

Node ที่เราจะใช้งานมีอยู่ 3 ประเภท

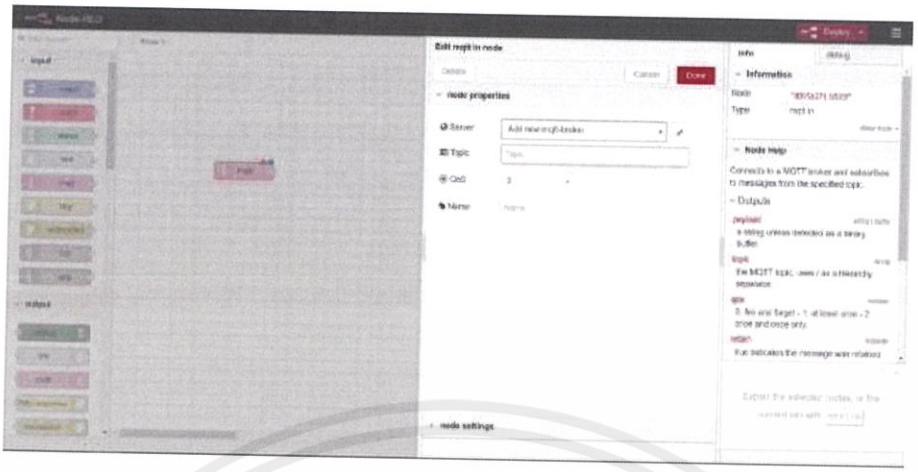
1. Input จะมีจุดให้ลากด้านขวาด้านเดียว node ที่นำข้อมูลเข้า node-red
2. Function มีจุดทั้ง2ข้าง สามารถจัดการข้อมูลทั้งขาเข้า และ ขาออก
3. Output จะมีจุดให้ลากแค่ด้านซ้าย นำข้อมูลออกไปแสดงผล

โดยเราสามารถลาก Node มาวางตรงไหนก็ได้บน Flow ของเรา



รูปที่ 3.12 การลาก Node บน flow

ตัวซ้ายคือประเภทที่ 1 ตัวกลางคือประเภทที่ 2 ตัวขวาคือประเภทที่ 3 โดย Node ประเภท 1 และ 3 จะต้องตั้งค่าเซิฟเวอร์ก่อนถึงจะใช้งานได้ ส่วนประเภท 2แค่ตั้งค่า output กับ input (ลากเส้น)ให้มันก็สามารถใช้งานได้แล้ว เริ่มต้น ลองลาก Node ที่เป็น input ที่ชื่อว่า MQTT มาวางบนจอ แล้วคลิกดูรายละเอียด



รูปที่ 3.13 รายละเอียดของ Node ที่เป็น Input ชื่อ MQTT

การตั้งค่าแบบนี้ขึ้นมา

Server: ตั้งค่า/กำหนด Server MQTT ที่เราจะ connect

Topic: หัวข้อในการพูดคุย

QoS: ลำดับความสำคัญ (กรณีมีหลายๆ Node มาต่อกันแล้วมีข้อความเข้าพร้อมๆกัน)

Name: ชื่อ node ที่เราจะตั้งให้กับอุปกรณ์

กดรูปปากกาตรง Server เพื่อตั้งค่า Server

รูปที่ 3.14 Add new MQTT-Broker

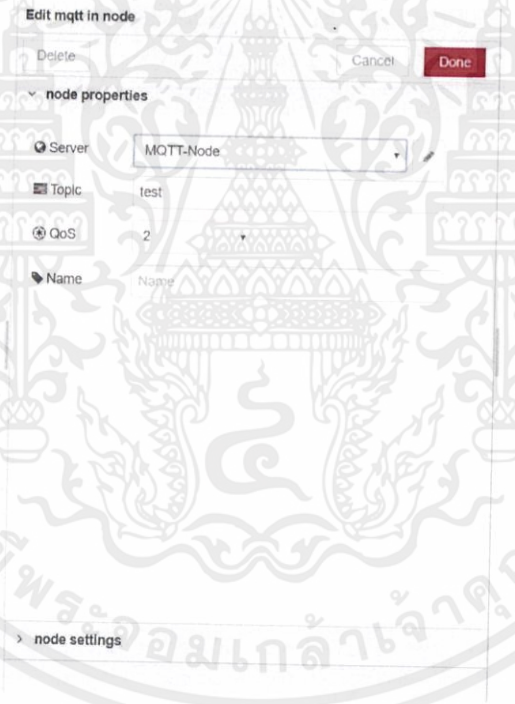
Name: ชื่อเซิฟเวอร์ที่เราต้องการแสดงบนจอของเรา

Server: broker.hivemq.com Port: 1883 (ใช้งานได้ฟรี)

** Port ใช้ใน Node-RED จะเป็น Port ใช้กับ Hardware ส่วนมากจะเป็น 1883**

จากนั้นกด Add เพื่อเซฟการตั้งค่าของเรา

จากนั้นเพิ่ม Topic เข้าไป เป็นคำว่า Test
Topic คือ หัวข้อที่จะพูดคุยกันในเครือข่ายเดียวกัน โดยทั้งผู้รับและผู้ส่งจะต้องตั้งหัวข้อในการส่งข้อความเป็นหัวข้อเดียวกัน จึงจะสื่อสารกันได้



รูปที่ 3.15 การตั้งค่า MQTT

ลาก output ตัวแรกสุดที่ชื่อว่า Debug ลากเส้นจาก MQTT เข้า Debug แล้วกด Deploy มุมขวาบนได้เลย

รูปที่ 3.16 การนำ Output ไป Debug

3.4.4 ขั้นตอนการติดตั้ง MQTT Broker

Install MQTT Broker on Ubuntu

เริ่มต้นจาก install mosquitto และ mosquitto-clients บน Ubuntu ที่สร้างขึ้น

```
sudo apt-get install mosquitto mosquitto-clients
```

สามารถทดสอบการ subscribe ได้ด้วยคำสั่ง

```
mosquitto_sub -h localhost -t test
```

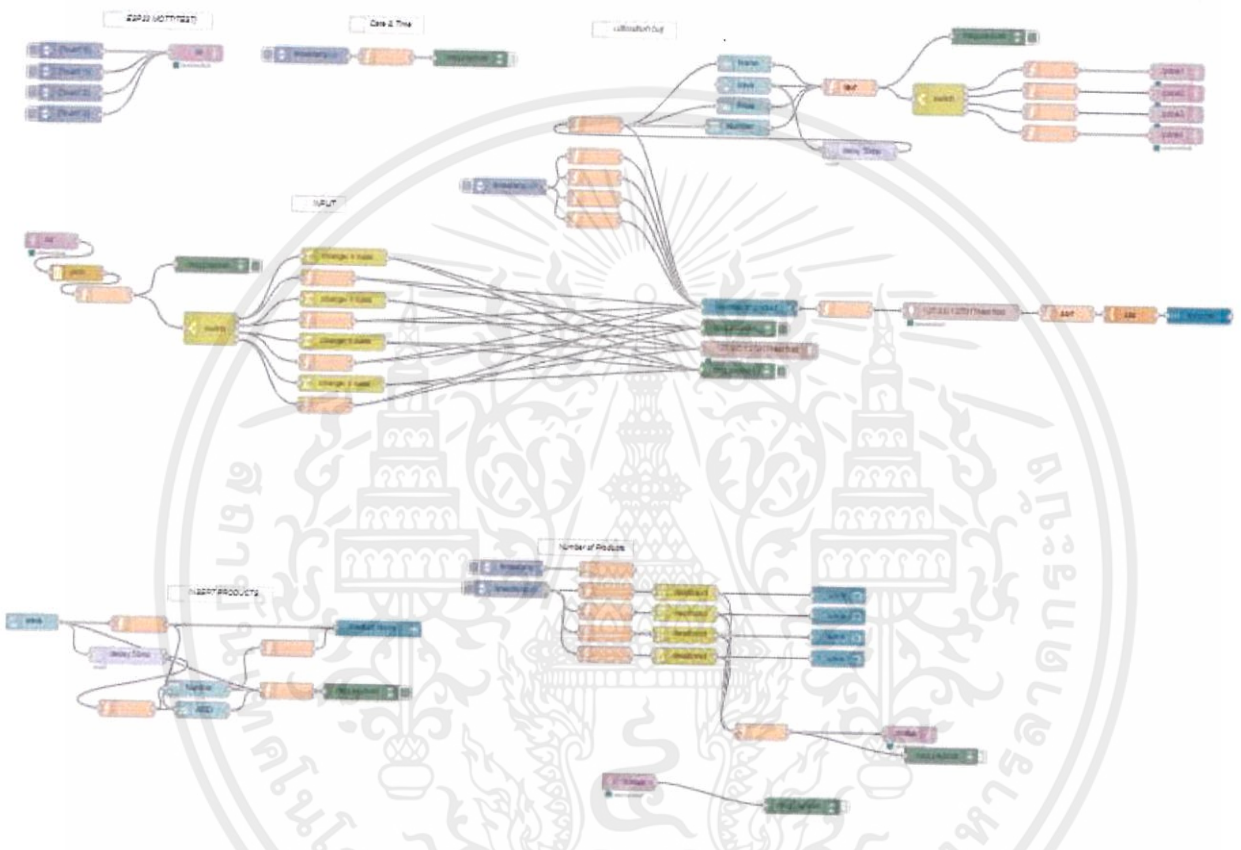
การส่งข้อมูลสามารถทำได้โดยคำสั่ง

```
mosquitto_pub -h localhost -t test -m "hello world"
```

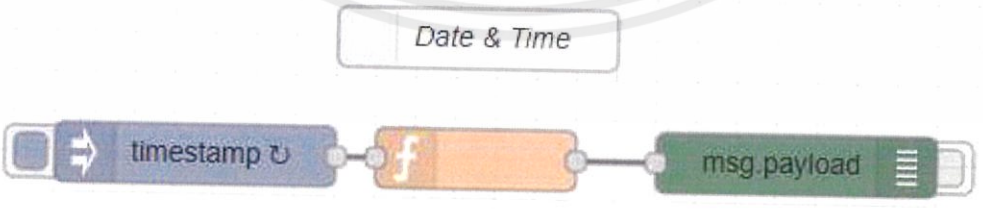
รูปที่ 3.17 ขั้นตอนการติดตั้ง MQTT Broker

3.4.5 สร้าง flow ใน Node-RED

เพื่อให้ทำงานตามระบบที่ได้วางไว้ ตัวอย่างของ flow ทั้งหมด และการ Import flow

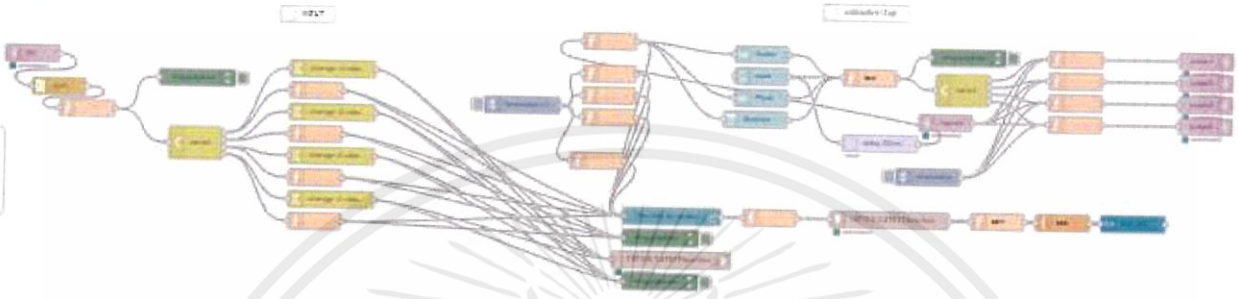


รูปที่ 3.18 flow ทั้งหมด



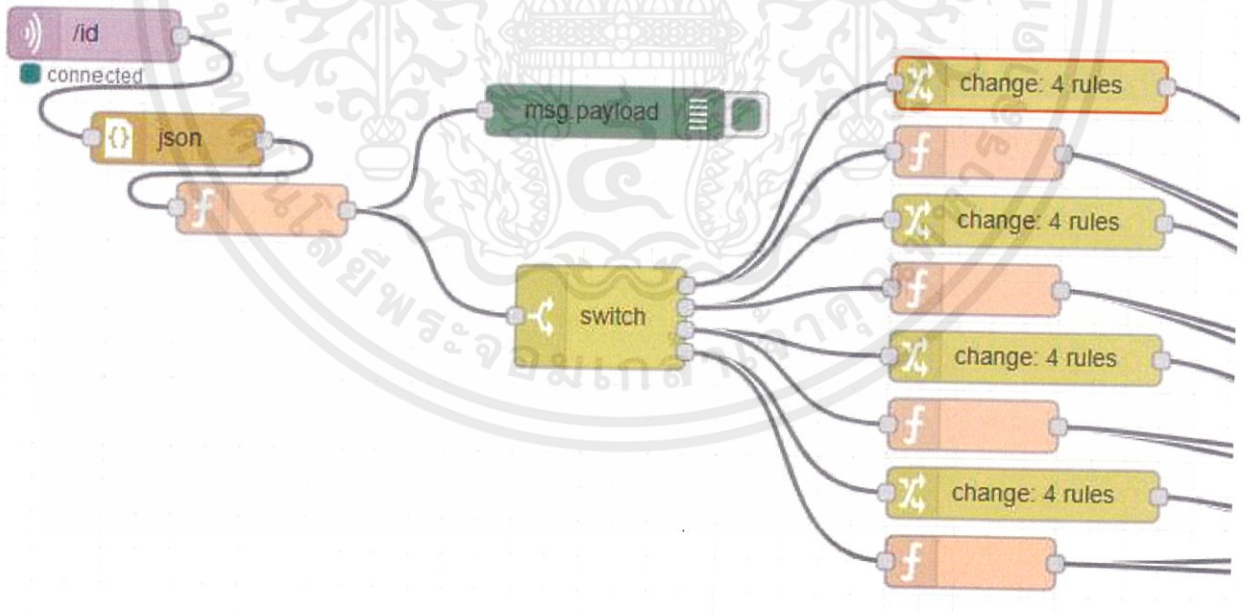
รูปที่ 3.19 flow กำหนด timestamp node

ส่วนกำหนดเวลาใช้ timestamp node ให้ส่งค่าทุก ๆ 1 วินาที เพื่อให้ function node ทำงาน



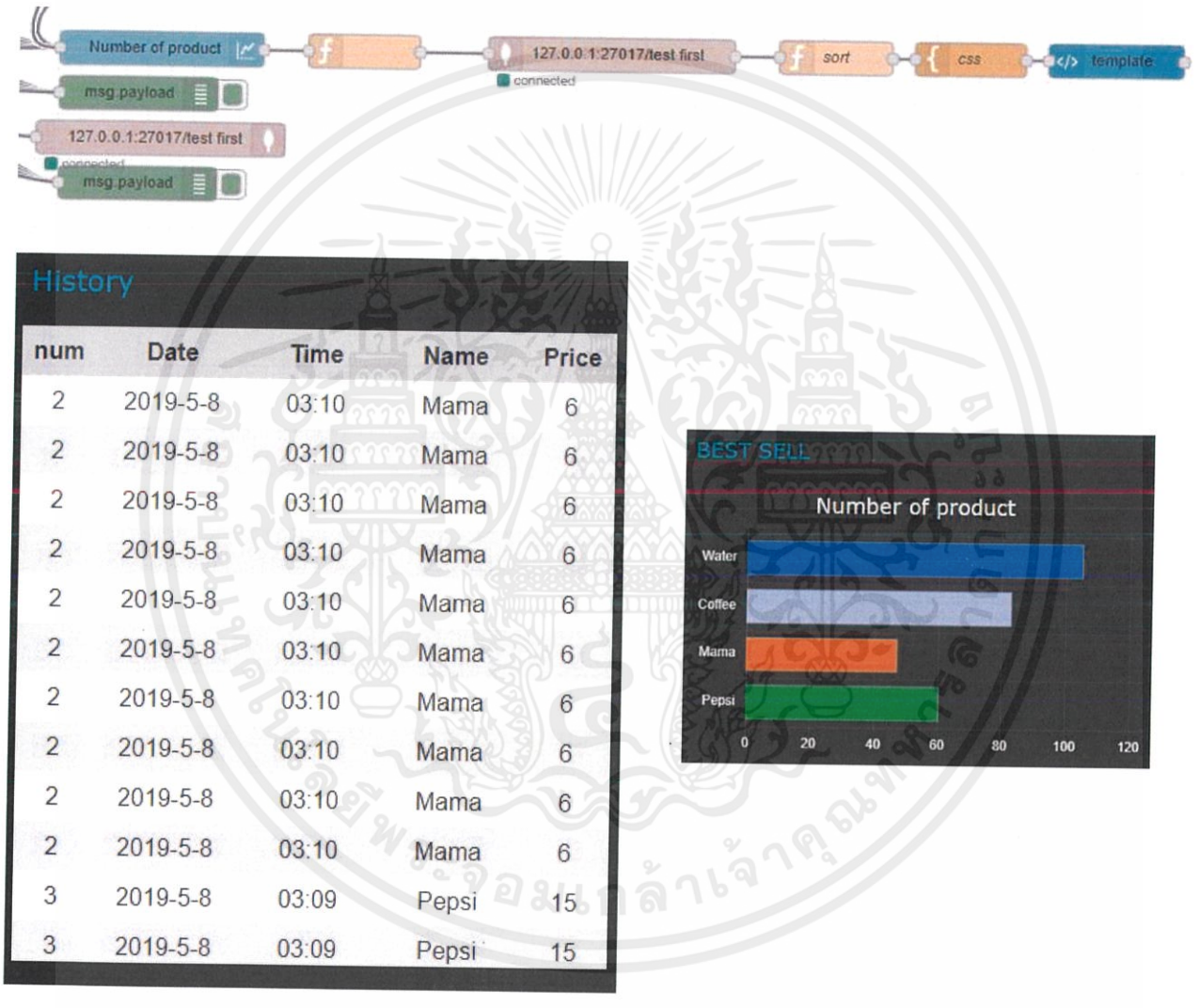
รูปที่ 3.20 flow การทำงานส่วนรับค่า และ เปลี่ยนสินค้าในตู้

ส่วนรับค่าจาก ESP32,บันทึกประวัติ,แสดงประวัติการจำหน่าย,เปลี่ยนแปลงสินค้า,บอกสถานะของสินค้า โดยจะอธิบายเป็นส่วนย่อย ๆ ดังนี้



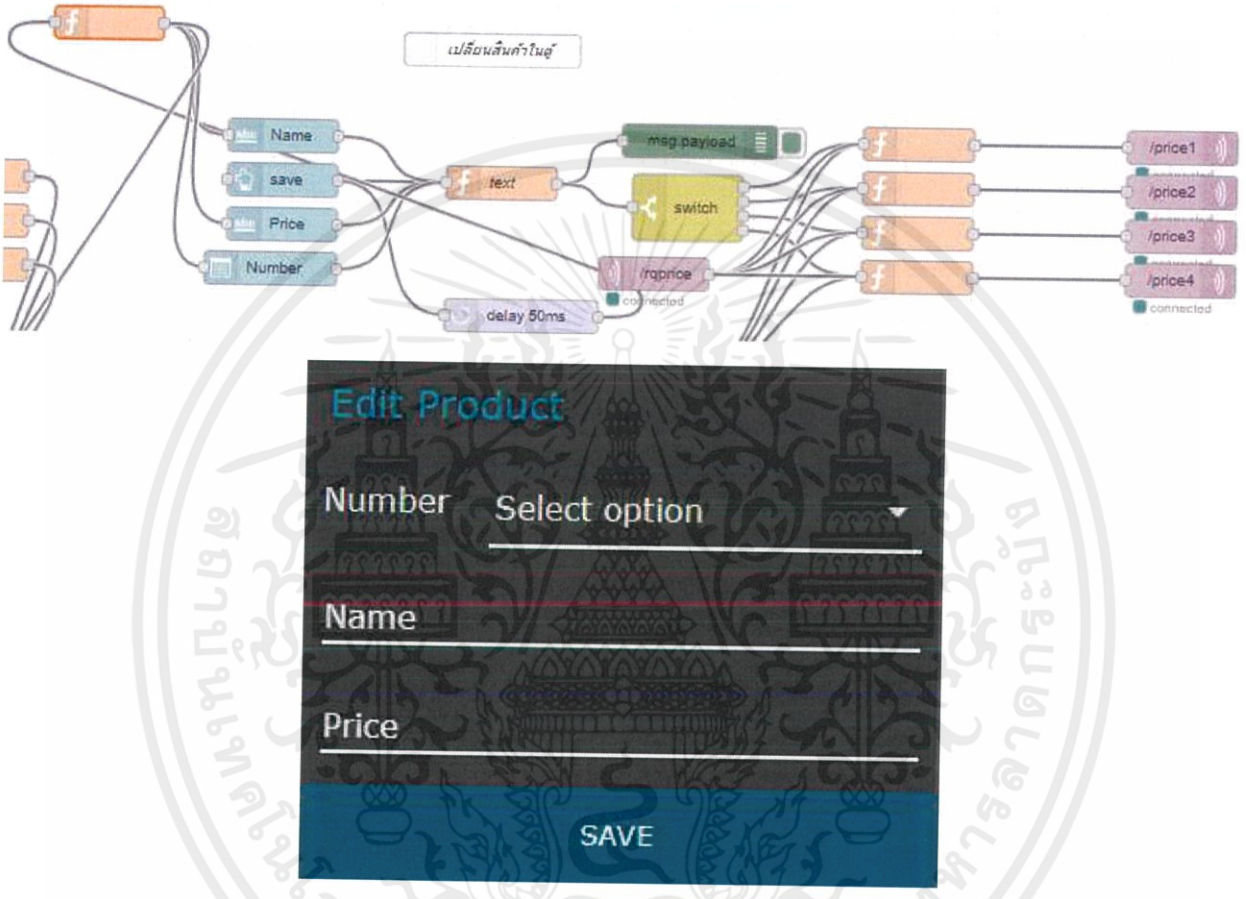
รูปที่ 3.21 flow ส่วนรับค่าจาก ESP32

ส่วนรับค่าจาก ESP32 โดย Subscribe จาก MQTT Broker ค่าที่ได้จะเป็น string ที่อยู่ในรูปแบบของ JSON จากนั้น เราจะใช้ JSON node เพื่อแปลงค่าที่รับมาเป็น json แล้ว function node จะเช็คว่าสินค้าที่สั่งซื้อมีเหลือในตู้หรือไม่ จากนั้นจะแยกสินค้าตามหมายเลขที่รับมา แล้วเพิ่ม วันที่ เวลา ชื่อสินค้า ราคา เข้าไปใน JSON format อีกที



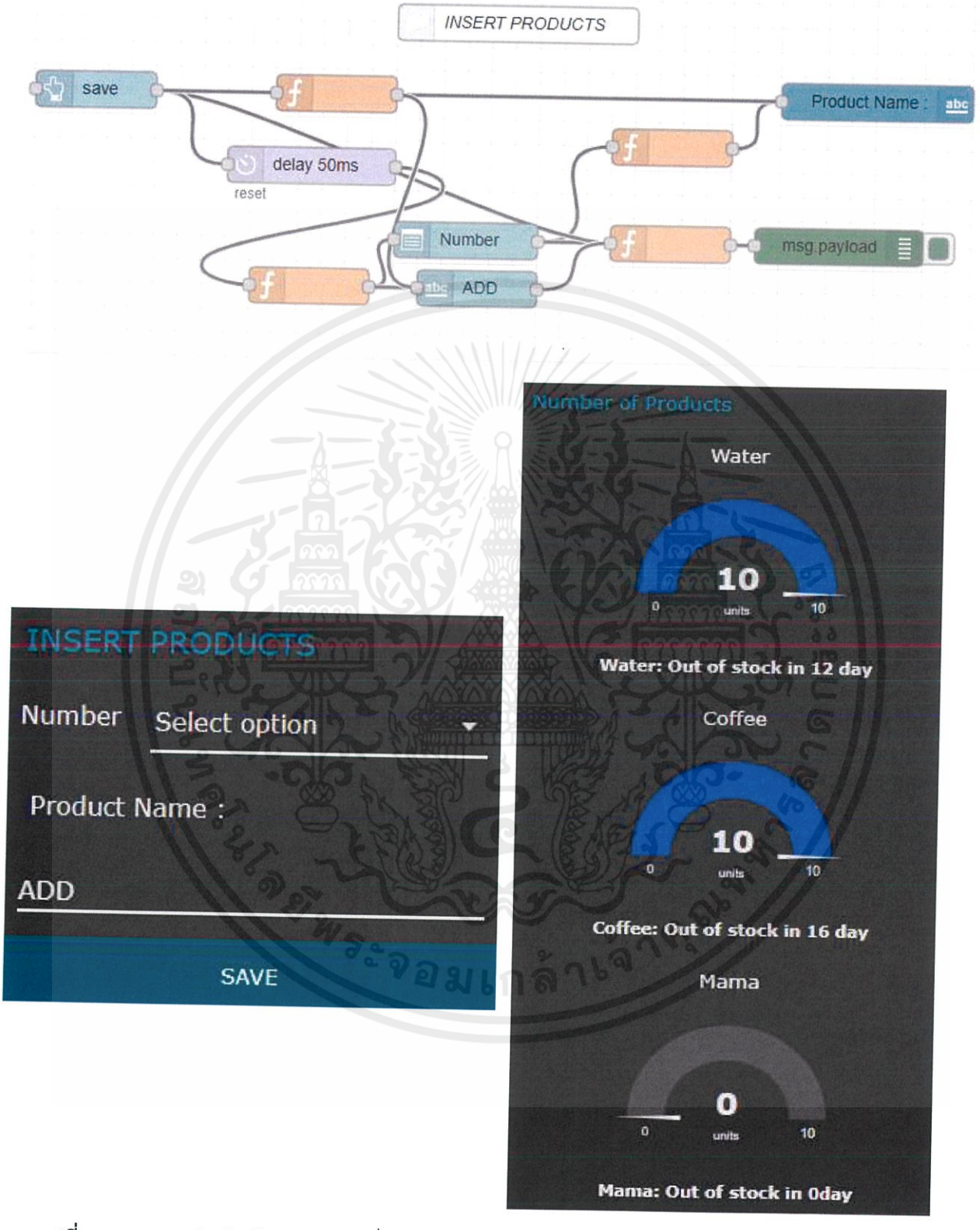
รูปที่ 3.22 flow ส่วนบันทึกลง database , เพิ่มยอดจำหน่ายของสินค้านั้น ๆ แล้วแสดงในหน้าเว็บ(บน) ประวัติการจำหน่ายสินค้า(ล่างซ้าย) กราฟแสดงความนิยมสินค้า(ล่างขวา)

นำข้อมูลจากส่วนก่อนหน้า ไปบันทึกลง database , เพิ่มยอดจำหน่ายของสินค้านั้น ๆ แล้วแสดงในหน้าเว็บ , ทำการเรียกดูข้อมูลทั้งหมดใน database แล้วแสดงออกมาในหน้าเว็บ ผ่าน template node



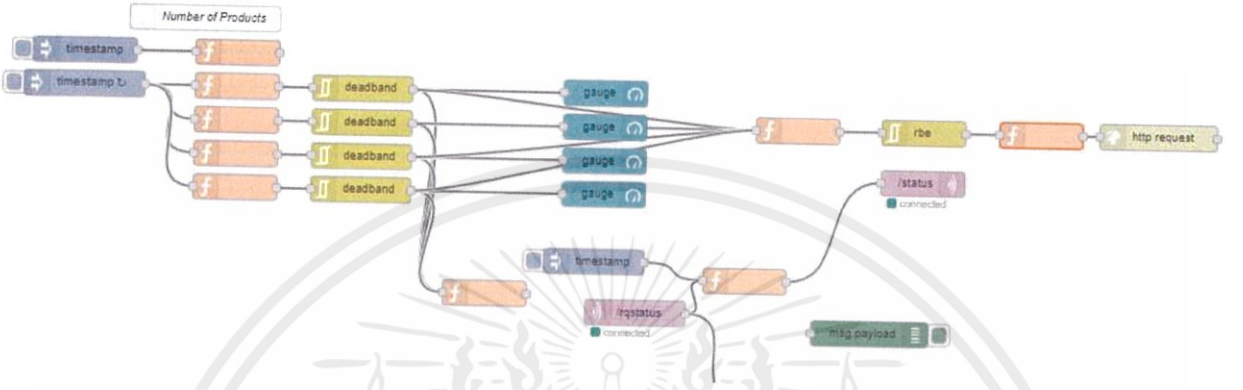
รูปที่ 3.23 flow เปลี่ยนสินค้าในตู้(บน) การตั้งค่าสินค้า(ล่าง)

การสร้างส่วนกรอกข้อมูลบนหน้าเว็บ โดยเมื่อมีการเปลี่ยนชนิดของสินค้าในตู้พนักงานจะต้องมากรอกรายการสินค้าใหม่ที่ได้ทำการเปลี่ยนไป จากนั้นเมื่อกด save ชื่อ และ ราคา ของสินค้า จะถูกเปลี่ยนไป และส่งราคาใหม่ไปอัพเดตที่ ESP32 ในเวลาเดียวกันจะทำการแสดงยอดจำหน่ายสินค้าใหม่ โดยไปแทนที่สินค้าเดิม



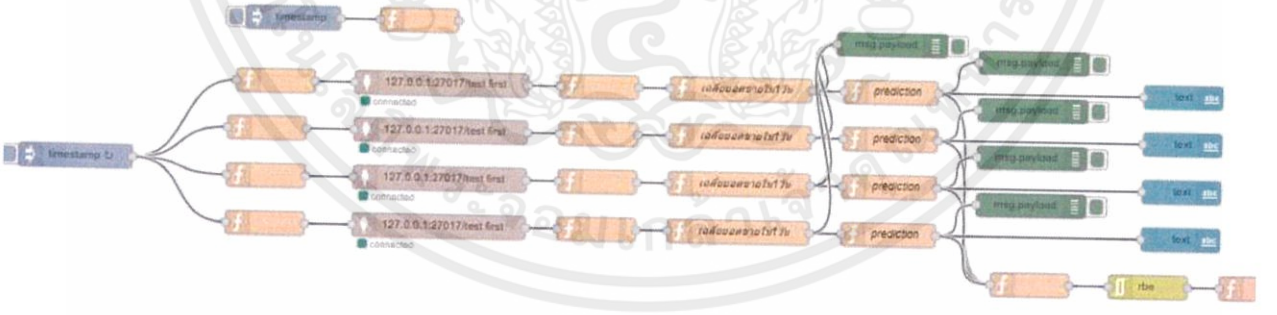
รูปที่ 3.24 flow เติมสินค้า (บน) การเพิ่มสินค้า(ล่างซ้าย) การแสดงผลจำนวนสินค้าบนหน้าเว็บ(ล่างขวา)

ส่วนเพิ่มสินค้าในตู้ เมื่อพนักงานได้ทำการเพิ่มสินค้าในตู้จะต้องมากรอกจำนวนสินค้าที่ใส่เข้าไปบนหน้าเว็บแล้วระบบจะทำการเพิ่มยอดสินค้าคงเหลือในตู้



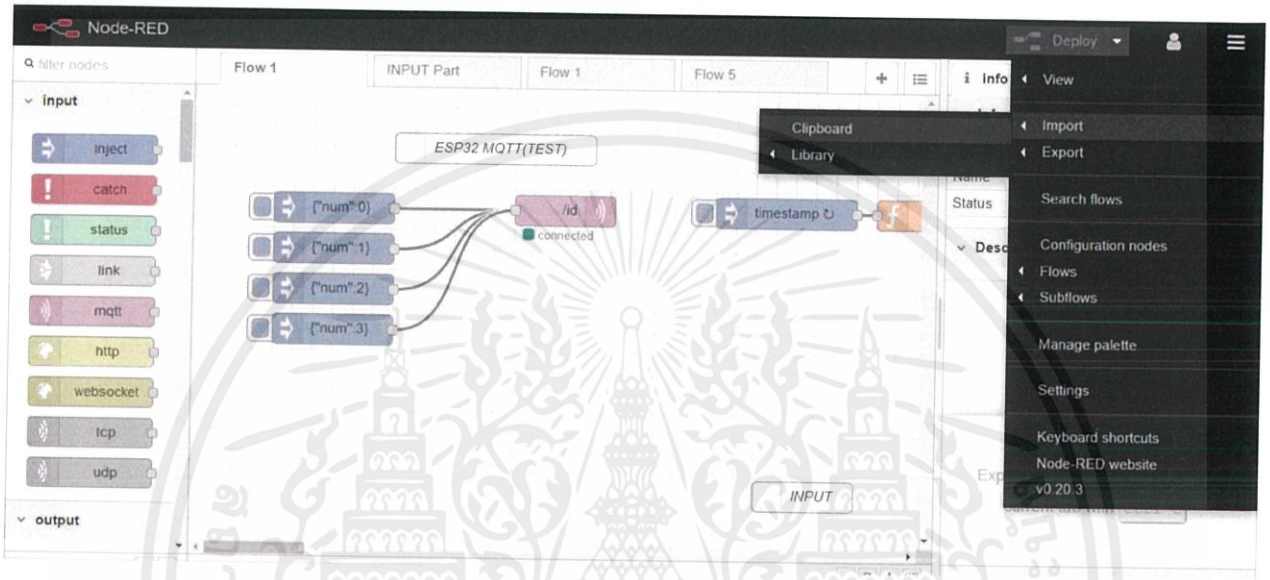
รูปที่ 3.25 flow ส่วนเพิ่มสินค้าในตู้

ส่วนสร้างหน้าสินค้าคงเหลือในตู้



รูปที่ 3.26 flow ส่วนสร้างหน้าสินค้าคงเหลือในตู้

ส่วนแจ้งเตือนสินค้าหมดผ่าน line application เมื่อสินค้าหมดตู้หรือถึงเวลาแจ้งเตือน โดยจะทำการเรียกข้อมูลการจำหน่ายในวันนั้น ๆ มาเฉลี่ยแล้วไปเช็คกับสินค้าในตู้ว่าจะหมดในอีกกี่วัน หากสินค้าในตู้จะหมดในเวลาไม่เกิน 2 วันก็จะส่งข้อความแจ้งเตือนไปยัง line application กด import ตามรูป



รูปที่ 3.27 การ Import flow

จากนั้นวางclipboard จาก

https://drive.google.com/open?id=1fASphYU2cGkP0WhRx4sly553hO7fQJVy

ลงไป ก็จะได้ flow ทั้งหมดใน รูปข้างต้นมา

3.4.6 ออกแบบการทำงานของเครื่องด้วย ESP32

ในส่วนของ ESP32 เราจะเขียนคำสั่งในเครื่องจำหน่ายสินค้าอัตโนมัติ ทำงานตามปกติ แต่เพิ่มเติมคือ เมื่อจำหน่ายสินค้าไปแล้ว จะทำการรายงานไปที่เซิร์ฟเวอร์ว่าจำหน่ายสินค้าหมายเลขอะไรไป จากนั้น flow ใน node red จะทำการเก็บค่าลง Database และคำนวณการจำหน่ายในแต่ละวัน พร้อมแจ้งเตือนเมื่อถึงเวลาที่ต้องไปเติมสินค้า

ตัวอย่างโค้ดคำสั่ง :

```
void setup() {
 // put your setup code here, to run once:
 Serial.begin(115200);
 pinMode(34,INPUT); //sensor infrared
 Wire.begin(); //i2c pcf8574
 Wire.beginTransmission(0x20); //0x20
 Wire.write(0x00);
 Wire.endTransmission();
 delay(500);

 setup_wifi();
 client.setServer(mqtt_server, 1883);
 client.setCallback(callback);

ledcSetup(1, 50, TIMER_WIDTH); // channel 1, 50 Hz, 16-bit width
ledcAttachPin(32, 1);   // GPIO 22 assigned to channel 1

ledcSetup(2, 50, TIMER_WIDTH); // channel 2, 50 Hz, 16-bit width
ledcAttachPin(33, 2);   // GPIO 19 assigned to channel 2
```

```
    /*---------------CHECK PRODUCT STATUS & PRICE----------------*/
          client.subscribe("/status");

          client.subscribe("/price1");

          client.subscribe("/price2");

          client.subscribe("/price3");

          client.subscribe("/price4");

    /*---------------CHECK PRODUCT STATUS---------------*/
         client.publish("/rqprice", "request price");
         client.publish("/rqstatus", "request status");
  xq = xQueueCreate(10, sizeof(uint16_t));
  xTaskCreate( task1, "TaskOne", 4096, NULL, 1, NULL);
}
void loop() {
  // put your main code here, to run repeatedly:
  if (!client.connected())
  {
    reconnect();
  }
  client.loop();
```

```
  /*--------------------CHECK MONEY--------------------*/

  for(int i = 0; i < 4; i++)
  {
    if (money >= price[i] && _status[i] != 0 ) {
      //led[i]==GREEN
      led = led | LED_G_ON[i];

    }else{
      led = led & LED_G_OFF[i];
    }

}
     Wire.beginTransmission(PCF8574_ADDR); //0x20
     Wire.write(led);
     Wire.endTransmission();

/*--------------------CHECK MONEY--------------------*/
/*--------------------PUSH BUTTONS--------------------*/

Wire.requestFrom(0x21, 1);    // request 1 bytes from slave device 001
while (Wire.available()) { // slave may send less than requested
```

```
    c = Wire.read(); // receive a byte as character
  }
  //Serial.println(c);
  switch (c)
  {
    case 254:
      sendproduct(0);
      break;
    case 253:
      sendproduct(1);
      break;
    case 251:
      Serial.println("sendproduct chanel 3");
      Serial.println(_status[2]);
      Serial.println(price[2]);
      Serial.println(money);
  if (money >= price[2] && _status[2] != 0 ) {
   Serial.println("sendproduct");
   Wire.beginTransmission(0x20); //0x20
   Wire.write(0x00);
   Wire.endTransmission();
   delay(1000);
   Serial.println("change = ");
   Serial.println(money-price[2]);
   money = 0;
```

```
    snprintf (msg, 75, "{\"num\":%ld}", 2);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish("/id", msg);
    client.publish("/rqstatus", "request status");

   loop();

}
delay(300);
     break;
  case 247:
     Serial.println("sendproduct chanel 4");
     Serial.println(_status[3]);
     Serial.println(price[3]);
     Serial.println(money);
     if (money >= price[3] && _status[3] != 0 ) {
  Serial.println("sendproduct");
  Wire.beginTransmission(0x20); //0x20
  Wire.write(0x00);
  Wire.endTransmission();
  delay(1000);
  Serial.println("change = ");
```

```
      Serial.println(money-price[3]);
      money = 0;



      snprintf (msg, 75, "{\"num\":%ld}", 3);
      Serial.print("Publish message: ");
      Serial.println(msg);
      client.publish("/id", msg);
      client.publish("/rqstatus", "request status");
      loop();
  }
   delay(300);
       break;
    case 127:
       money +=10;
       Serial.println(money);
       delay(300);
       while(Wire.read()==128){delay(100);}
       loop;
       break;
    default:
       break;
  }
/*--------------------PUSH BUTTONS-------------------*/}
```

ฉบับเต็มที่ภาคผนวก

บทที่ 4

ผลการทดลอง

ผลการทดลองที่ได้รับจากการทดลองแสดงในตาราง โดยแบ่งหัวข้อออกเป็นดังนี้ ส่วนรับข้อมูลจาก ESP32 ส่วนแก้ไขข้อมูล ส่วนเพิ่มสินค้าในตู้ หน้าเว็ปแสดงผล ตู้จำหน่ายสินค้า โดย 1 คือ การทำงานตามที่ต้องการ และ 0 คือ การทำงานผิดพลาด

ตารางที่ 4.1 แสดงการรับข้อมูลของฐานข้อมูลออนไลน์ โดยรับข้อมูลจาก ESP32 ที่อยู่ในเครื่องจำหน่ายสินค้า

ส่วนรับข้อมูลจาก ESP32				
	MQTT Subscribe	Add data & time	Classified type	Insert to DB
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1
5	1	1	1	1

ตารางที่ 4.2 แสดงการแก้ไขข้อมูลของฐานข้อมูลออนไลน์ แสดงส่วนต่างๆของฐานข้อมูลออนไลน์ที่สามารถแก้ไขได้ และการบันทึกผลหลังจากการแก้ไข

ส่วนแก้ไขข้อมูล				
	เลือกชนิดของสินค้า	บันทึกชื่อสินค้า	บันทึกราคา	กด saveจึงจะทำการบันทึกข้อมูล
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1
5	1	1	1	1

ตารางที่ 4.3 แสดงการเพิ่มสินค้าในเครื่องจำหน่ายสินค้า แสดงการเพิ่มสินค้า จำนวนสินค้าที่เพิ่ม และการบันทึกข้อมูลหลังการเพิ่มสินค้า

ส่วนเพิ่มสินค้าในเครื่องจำหน่ายสินค้า				
	เลือกชนิดสินค้าได้	มีชื่อสินค้าขึ้นตรงกับชนิดที่เลือก	ใส่จำนวนที่เติมได้	กด save จึงจะทำการบันทึกข้อมูล
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1
5	1	1	1	1

ตารางที่ 4.4 แสดงส่วนของหน้าเว็บแสดงผล หน้าเว็บจะแสดงข้อมูลต่างๆตามที่แสดงในตาราง

หน้าเว็บแสดงผล			
	แสดงจำนวนสินค้าที่จำหน่ายได้สะสม	แสดงจำนวนสินค้าที่เหลืออยู่ในตู้	แสดงประวัติการจำหน่าย
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1

ตารางที่ 4.5 แสดงการทำงานของตู้จำหน่ายสินค้า แสดงการจ่ายสินค้า การทำงานของเซนเซอร์ และ MQTT Subscribe ที่ควบคุมจากฐานข้อมูลออนไลน์

ตู้จำหน่ายสินค้า				
	จ่ายสินค้าได้ทีละ 1 ชิ้น	เซนเซอร์ตรวจสอบการจ่ายสินค้าทำงานได้	MQTT Subscribe สถานะสินค้าว่าพร้อมจำหน่ายหรือไม่	MQTT Subscribe ราคาสินค้ามาอัพเดท
1	1	1	1	0
2	1	1	0	0
3	1	1	1	0
4	0	0	1	0
5	1	1	0	0

บทที่ 5

สรุปและวิจารณ์ผลการทดลอง

5.1 สรุปผลการทดลอง

จากการทดสอบการใช้งาน ระบบบันทึกแล้ววิเคราะห์ข้อมูลการจำหน่ายสินค้าในเครื่องจำหน่ายสินค้าอัตโนมัติ สรุปผลได้ดังนี้

1. ระบบสามารถเก็บค่าจากการSubscribe message ผ่าน MQTT protocal ได้เป็นปกติ
2. ระบบสามารถบันทึกค่าลงDatabase ได้เป็นปกติ
3. ระบบสามารถแสดง ยอดจำหน่ายสินค้าแต่ละชนิด , ประวัติการจำหน่ายสินค้า และจำนวนสินค้าที่คงเหลือในเครื่องได้
4. ตัวเครื่องยังมีปัญหาการ Publish , Subscribe ผ่าน MQTT protocal แบบ multiple topics อยู่ ส่งผลให้การทำงานไม่เป็นไปตามลำดับที่วางไว้

5.2 วิจารณ์ผลการทดลอง

จากผลการทดลอง เราสังเกตได้ว่า การส่งข้อมูลผ่าน MQTT protocal จะมีDelay จากการส่ง ทำให้การทำงานของเครื่องจำหน่ายสินค้าอัตโนมัติทำงานไม่เป็นลำดับตามที่วางไว้ อีกประเด็นที่เจอคือ เครื่องมือที่เลือกใช้ (Node-red) ทำให้เราพอเข้าใจการทำงานของระบบ แต่ไม่เป็นรูปแบบที่ชัดเจนและเป็นสากล

5.3 ปัญหาและแนวทางการพัฒนา

ในระบบที่วางไว้สามารถเอาไปปรับใช้กันการเก็บสินค้า ระบบการยืมสินค้าได้ทุกประเภท โดยเน้นไปที่ระบบการเก็บประวัติการใช้สินค้า และการนับสินค้าที่เหลืออยู่ในคลัง

บรรณานุกรม

[1] Editorial. (2018). **เซอร์โวมอเตอร์**. เข้าถึงได้จาก :

https://www.inventor.in.th/home/%E0%B9%80%E0%B8%8B%E0%B8%AD%E0%B8%A3%E0%B9%8C%E0%B9%82%E0%B8%A7%E0%B8%A1%E0%B8%AD%E0%B9%80%E0%B8%95%E0%B8%AD%E0%B8%A3%E0%B9%8C/#.XOL6UsgzZPb

[2] Wikipedia. (2019). **Photodiode**. เข้าถึงได้จาก : https://en.wikipedia.org/wiki/Photodiode

[3] Sakul Montha. (2018). **สร้าง REST-API ด้วย Node-RED กันเถอะ**. เข้าถึงได้จาก :

https://medium.com/@iamgique/%E0%B8%AA%E0%B8%A3%E0%B9%89%E0%B8%B2%E0%B8%87-rest-api-%E0%B8%94%E0%B9%89%E0%B8%A7%E0%B8%A2-node-red-%E0%B8%81%E0%B8%B1%E0%B8%99%E0%B9%80%E0%B8%96%E0%B8%AD%E0%B8%B0-9c8eff780dd3

[4] supotsaeea. (2015). **Node-RED คืออะไร**. เข้าถึงได้จาก : https://supotsaeea.wordpress.com

ภาคผนวก

```
//-----------------Header-----------------//

#include <WiFiMulti.h>

#include <PubSubClient.h>

#define MQTT_ID "ESP32424242"


const char* ssid     = "Kitirat";

const char* password = "89MXK5T7";

const char *mqtt_server = "35.247.190.221";


WiFiMulti WiFiMulti;

WiFiClient esp32Client;

PubSubClient client(esp32Client);


#include <Wire.h>

#include <ESP32_Servo.h>


#define TIMER_WIDTH 16


#include "esp32-hal-ledc.h"
```

```
// address of PCF8574 IC
#define PCF8574_ADDR (0x20)

//Servo myservo;
//int ADC_Max = 4096;
uint8_t money = 0 , income = 0 ,c, led = 0x00;
uint8_t price[4] = {20,30,40,50} ;
uint8_t LED_G_ON[4] = {0x02,0x08,0x20,0x80},
        LED_G_OFF[4] = {0xfd,0xf7,0xdf,0x7f},
        LED_R_ON[4] = {0x01,0x04,0x10,0x40},
        LED_R_OFF[4] = {0xfe,0xfb,0xef,0xbf};
uint8_t _status[4];

char msg[50];
//-----------------Header-----------------//
void callback(char *topic, byte *payload, unsigned int length)
{
  //mypayload = 0 ;
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
```

```
for (int i = 0; i < length; i++)
{
  mypayload[i] = (char)payload[i];

}
Serial.print(mypayload);
Serial.println();

if (strcmp(topic,"/status")==0) {

 for (int i = 0; i < 4; i++)
 {
   Serial.print((char)payload[i]);
   if(payload[i] == '0' ){//เปิดไฟแดง

      _status[i] = 0;
      led = led | LED_R_ON[i];
      //cancel_ch(i);

   }
```

```
      else {


        _status[i] = 1 ;
        //use_ch(i);
        led = led & LED_R_OFF[i];//ปิดไฟแดง
      }
      Serial.print(_status[0]);
      Serial.print(_status[1]);
      Serial.print(_status[2]);
      Serial.println(_status[3]);
      Wire.beginTransmission(PCF8574_ADDR); //0x20
      Wire.write(led);
      Wire.endTransmission();


    }
    Serial.println();
    memset(mypayload, 0, sizeof mypayload);
  }
  if (strcmp(topic,"/price1")==0) {
    price[0] = strtol(mypayload,NULL,10);
    Serial.println(price[0]);
```

```
    memset(mypayload, 0, sizeof mypayload);
  }
  if (strcmp(topic,"/price2")==0) {
    price[1] = strtol(mypayload,NULL,10);
    Serial.println(price[1]);
    memset(mypayload, 0, sizeof mypayload);
  }
  if (strcmp(topic,"/price3")==0) {
    price[2] = strtol(mypayload,NULL,10);
    Serial.println(price[2]);
    memset(mypayload, 0, sizeof mypayload);
  }
  if (strcmp(topic,"/price4")==0) {
    price[3] = strtol(mypayload,NULL,10);
    Serial.println(price[3]);
    memset(mypayload, 0, sizeof mypayload);
  }
```

```
}



xQueueHandle xq; // GLOBAL QUEUE OBJECT

void setup() {

 // put your setup code here, to run once:

 Serial.begin(115200);

 pinMode(34,INPUT); //sensor infrared

 Wire.begin(); //i2c pcf8574

 Wire.beginTransmission(0x20); //0x20

 Wire.write(0x00);

 Wire.endTransmission();

 delay(500);


 setup_wifi();

 client.setServer(mqtt_server, 1883);

 client.setCallback(callback);


 ledcSetup(1, 50, TIMER_WIDTH); // channel 1, 50 Hz, 16-bit width
```

```
  ledcAttachPin(32, 1);   // GPIO 22 assigned to channel 1


  ledcSetup(2, 50, TIMER_WIDTH); // channel 2, 50 Hz, 16-bit width
  ledcAttachPin(33, 2);   // GPIO 19 assigned to channel 2
    /*---------------CHECK PRODUCT STATUS & PRICE----------------*/
        client.subscribe("/status");


        client.subscribe("/price1");


        client.subscribe("/price2");


        client.subscribe("/price3");


        client.subscribe("/price4");


    /*---------------CHECK PRODUCT STATUS----------------*/
        client.publish("/rqprice", "request price");
        client.publish("/rqstatus", "request status");
  xq = xQueueCreate(10, sizeof(uint16_t));
  xTaskCreate( task1, "TaskOne", 4096, NULL, 1, NULL);
}
```

```
void sendproduct(int num){


 if (money >= price[num] && _status[num] != 0 ) {
  Serial.println("sendproduct");
  Wire.beginTransmission(0x20); //0x20
  Wire.write(0x00);
  Wire.endTransmission();
  if(num == 0){
    ledcWrite(num+1, 1638);       // sweep servo 1
  }else{
    ledcWrite(num+1, 7864);       // sweep servo 1
  }


  //myservo.write(0);                    // ส่งค่าไปควบคุม servo เป็นมุมที่ต้องการในช่วง 0-179 องศา
  delay(1000);
  while(digitalRead(34)== 1){Serial.println("wait digitalRead(34)== 0");}
  while(digitalRead(34)== 0){Serial.println("wait digitalRead(34)== 1");}
  Serial.println("confirm !!!");
  if(num == 0){
    ledcWrite(num+1, 7864);       // sweep servo 1
```

```
  }else{
    ledcWrite(num+1, 1638);       // sweep servo 1
  }

  //myservo.write(180);                    // ส่งค่าไปควบคุม servo เป็นมุมที่ต้องการในช่วง 0-179 องศา
  delay(1000);

  Serial.println("change = ");
  Serial.println(money-price[num]);
  money = 0;

  snprintf (msg, 75, "{\"num\":%ld}", num);
  Serial.print("Publish message: ");
  Serial.println(msg);
  client.publish("/id", msg);
  client.publish("/rqstatus", "request status");

}
delay(300);
```

```
  loop();

}

void setup_wifi()
{
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFiMulti.addAP(ssid, password);

  delay(500);

  while (WiFiMulti.run() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
```

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

void reconnect()
{
 // Loop until we're reconnected
 while (!client.connected())
 {
  Serial.print("Attempting MQTT connection...");
  // Attempt to connect
  if (client.connect(MQTT_ID, "kitirat", "kitirat"))
  {
   Serial.println("connected");
   // ... and resubscribe
   /*----------CHECK PRODUCT STATUS & PRICE----------*/

    client.subscribe("/status");
```

```
        client.subscribe("/price1");
        client.subscribe("/price2");
        client.subscribe("/price3");
        client.subscribe("/price4");

      /*---------------CHECK PRODUCT STATUS----------------*/
    }
    else
    {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 2 seconds");
      // Wait 2 seconds before retrying
      delay(2000);
    }
  }
}


void task1(void *p) {
  while (1)
```

```
  {
    /* code */
  }


  vTaskDelete(NULL);
}



void loop() {
  // put your main code here, to run repeatedly:
  if (!client.connected())
  {
    reconnect();
  }
  client.loop();

  /*--------------------CHECK MONEY--------------------*/

  for(int i = 0; i < 4; i++)
```

```
{
  if (money >= price[i] && _status[i] != 0 ) {
    //led[i]==GREEN
    led = led | LED_G_ON[i];

  }else{
    led = led & LED_G_OFF[i];
  }

}
    Wire.beginTransmission(PCF8574_ADDR); //0x20
    Wire.write(led);
    Wire.endTransmission();

/*--------------------CHECK MONEY--------------------*/
/*--------------------PUSH BUTTONS--------------------*/

Wire.requestFrom(0x21, 1);    // request 1 bytes from slave device 001
while (Wire.available()) { // slave may send less than requested
```

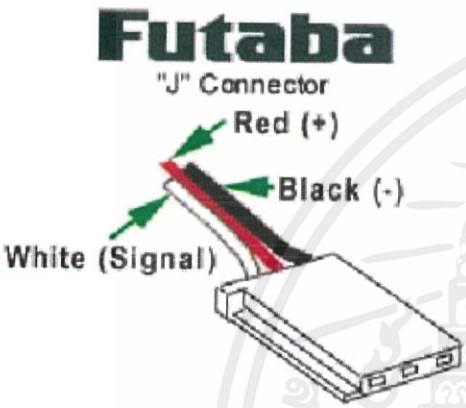
```
    c = Wire.read(); // receive a byte as character
}
//Serial.println(c);
switch (c)
{
  case 254:
    sendproduct(0);
    break;
  case 253:
    sendproduct(1);
    break;
  case 251:
    Serial.println("sendproduct chanel 3");
    Serial.println(_status[2]);
    Serial.println(price[2]);
    Serial.println(money);
if (money >= price[2] && _status[2] != 0 ) {
 Serial.println("sendproduct");
 Wire.beginTransmission(0x20); //0x20
 Wire.write(0x00);
 Wire.endTransmission();
```

```
    delay(1000);
    Serial.println("change = ");
    Serial.println(money-price[2]);
    money = 0;



    snprintf (msg, 75, "{\"num\":%ld}", 2);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish("/id", msg);
    client.publish("/rqstatus", "request status");


   loop();


}
delay(300);
     break;
   case 247:
     Serial.println("sendproduct chanel 4");
     Serial.println(_status[3]);
     Serial.println(price[3]);
```

```
    Serial.println(money);
    if (money >= price[3] && _status[3] != 0 ) {
  Serial.println("sendproduct");
  Wire.beginTransmission(0x20); //0x20
  Wire.write(0x00);
  Wire.endTransmission();
  delay(1000);
  Serial.println("change = ");
  Serial.println(money-price[3]);
  money = 0;



  snprintf (msg, 75, "{\"num\":%ld}", 3);
  Serial.print("Publish message: ");
  Serial.println(msg);
  client.publish("/id", msg);
  client.publish("/rqstatus", "request status");
  loop();


}
```

```
    delay(300);
      break;
    case 127:
      money +=10;
      Serial.println(money);
      delay(300);
      while(Wire.read()==128){delay(100);}
      loop;
      break;
    default:
      break;
  }

/*--------------------PUSH BUTTONS-------------------*/

}
```

S3003 FUTABA SERVO



..S3003 FUTABA SERVO...

Detailed Specifications

Control System:	+Pulse Width Control 1520usec Neutral	Current Drain (4.8V):	7.2mA/idle
Required Pulse:	3-5 Volt Peak to Peak Square Wave	Current Drain (6.0V):	8mA/idle
Operating Voltage:	4.8-6.0 Volts	Direction:	Counter Clockwise/Pulse Traveling 1520-1900usec
Operating Temperature Range:	-20 to +60 Degree C	Motor Type:	3 Pole Ferrite
Operating Speed (4.8V):	0.23sec/60 degrees at no load	Potentiometer Drive:	Indirect Drive
Operating Speed (6.0V):	0.19sec/60 degrees at no load	Bearing Type:	Plastic Bearing
Stall Torque (4.8V):	44 oz/in. (3.2kg.cm)	Gear Type:	All Nylon Gears
Stall Torque (6.0V):	56.8 oz/in. (4.1kg.cm)	Connector Wire Length:	12"
Operating Angle:	45 Deg. one side pulse traveling 400usec	Dimensions:	1.6" x 0.8"x 1.4" (41 x 20 x 36mm)
360 Modifiable:	Yes	Weight:	1.3oz. (37.2g)