ระบบควบคุมและจัดการข้อมูลอัจฉริยะ
INTELLIGENT CONTROL AND DATA SURVEILLANCE SYSTEM

ธนกร พานิช
ศุภเดช กิ่งนอก
สมาธิ อนันต์เจริญวัฒน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมแมคคาทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2560

ระบบควบคุมและจัดการข้อมูลอัจฉริยะ

INTELLIGENT CONTROL AND DATA SURVEILLANCE SYSTEM

ธนกร พานิช
ศุภเดช กิ่งนอก
สมาธิ อนันต์เจริญวัฒน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมแมคคาทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2560

INTELLIGENT CONTROL AND DATA SURVEILLANCE SYSTEM

THANAKORN PANICH

SUPHADET KINGNOK

SAMATHI ANANCHAROENWAT

THIS THESIS IS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN MECHATRONICS ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2017

ปริญญานิพนธ์ปีการศึกษา 2560

ภาควิชาวิศวกรรมการวัดและควบคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบควบคุมและจัดการข้อมูลอัจฉริยะ

INTELLIGENT CONTROL AND DATA SURVEILLANCE SYSTEM

ผู้จัดทำ นายธนกร พานิช 57010533

นายศุภเดช กิ่งนอก 57011267

นายสมาธิ อนันต์เจริญวัฒน์ 57011304

.....อาจารย์ที่ปรึกษา

(ผู้ช่วยศาสตราจารย์ ดร.คงศักดิ์ อนันตหิรัญรัตน์)

ระบบควบคุมและจัดการข้อมูลอัจฉริยะ

โดย

นายธนกร พานิช 57010533

นายศุภเดช กิ่งนอก 57011267

นายสมาธิ อนันต์เจริญวัฒน์ 57011304

อาจารย์ที่ปรึกษา

ผู้ช่วยศาสตราจารย์ ดร.คงศักดิ์ อนันตหิรัญรัตน์

ปีการศึกษา 2560

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอระบบควบคุมแบบไร้สาย ที่ถูกสร้างขึ้นจากการทำงานร่วมกันของอุปกรณ์ 2 ส่วน ส่วนที่หนึ่งคือ มินิคอมพิวเตอร์ Raspberry Pi ทำหน้าที่รับคำสั่งจาก User Interface (UI) บนหน้าเวปไซต์ และทำหน้าที่เป็นตัวประมวลผลกลางของระบบด้วยซอฟแวร์ Node-RED และส่วนที่สองคือ ไมโครคอนโทรลเลอร์ NodeMCU ทำหน้าที่รับคำสั่งจากฮาร์ดแวร์ เช่น สวิตซ์ปุ่มกด, เซนเซอร์วัดอุณหภูมิและความชื้น และควบคุมการทำงานของอุปกรณ์ต่างๆ เช่น หลอดไฟ, พัดลม และมอเตอร์เซอร์โว เป็นต้น ทั้ง 2 ส่วน ติดต่อสื่อสารกันผ่านโครงข่ายอินเตอร์เน็ตไร้สาย Wi-Fi ด้วยการเข้ารหัสแบบ ASCII ทำให้สามารถประยุกต์ใช้กับไมโครคอนโทรเลอร์รูปแบบอื่นๆ เพิ่มเติมได้ นอกจากนี้ระบบยังเก็บรวบรวมข้อมูลการใช้งานไว้ในระบบฐานข้อมูล ที่สามารถนำไปพัฒนาต่อยอดได้ในอนาคต ด้วยการควบคุมจากทั้งบนเวปไซต์ด้วยโทรศัพท์มือถือหรือคอมพิวเตอร์ ไปจนถึงการสั่งการด้วยสวิตซ์แบบดั้งเดิม และสามารถติดตั้งอุปกรณ์ต่างๆ เข้าไปเพิ่มเติมได้อีก เช่น ติดตั้งกล้องวิดีโอเพื่อแสดงภาพภายในห้องหรือเซนเซอร์ตรวจจับการเคลื่อนไหว เป็นต้น ทำให้การประยุกต์ใช้งานนั้นทำได้อย่างหลากหลาย จากการทดสอบการทำงานของระบบควบคุมไร้สาย ตัวระบบนั้นสามารถตอบสนองต่อคำสั่งจากทั้งซอฟแวร์และฮาร์ดแวร์ได้อย่างรวดเร็ว และแสดงสถานะของอุปกรณ์ที่ถูกควบคุมได้อย่างถูกต้อง ด้วยเหตุนี้ระบบควบคุมไร้สายสามารถช่วยอำนวยความสะดวก ยกระดับคุณภาพชีวิตและลดเหตุการอัคคีภัยจากการลืมปิดการทำงาน ของอุปกรณ์ไฟฟ้าภายในบ้านเรือนหรืออาคารได้เป็นอย่างมาก

INTELLIGENT CONTROL AND DATA SURVEILLANCE SYSTEM

By

Mr. Thanakorn Panich 57010533

Mr. Suphadet Kingnok 57011267

Mr. Samathi Anancharoenwat 57011304

Advisor

Asst.Prof.Dr. Kongsak Anuntahirunrat

Academic Year 2017

ABSTRACT

This thesis presents the wireless devices control system that consists of combination of 2 main section. The first part is Raspberry Pi 3 Model B as microcomputer that process any information from User Interface (UI) on website by Node-RED software. The second part is NodeMCU as microcontroller that send and receive data from switch and sensor. Also, able to control various electrical appliance such as lighting, fan and servo motor. Both parts are communicated by wireless network and encrypted by ASCII that make this system easy to develop in various environment. Database in this system collect usage data and able to develop in the future. This system able to control from both on website with a smart phone or computer yet the traditional switchgear. Additional equipment can be install depend on user need, such as a video camera or motion sensor. So, it makes the system have a wide applicable. The testing result was satisfied. The system responds quickly to signal from both software and hardware control. Also, the status of the device correctly displayed. Therefore, this system can facilitate, Improve the quality of life and reduce the risk of fire from forgetting to shut down electrical appliances in homes or buildings.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ดำเนินการสำเร็จลุล่วงไปได้ด้วยดี เนื่องจากได้รับความกรุณาเป็นอย่างสูงจากท่าน ผู้ช่วยศาสตราจารย์ ดร.คงศักดิ์ อนันตหิรัญรัตน์ อาจารย์ที่ปรึกษา ที่ได้ช่วยเหลือในการให้คำแนะนำชี้แนะแนวทางในการแก้ปัญหาที่เกิดขึ้นในการทำวิทยานิพนธ์อย่างทุ่มเท รวมทั้งฝึกฝน พวกข้าพเจ้าให้มีความรู้ความสามารถในการทำปริญญานิพนธ์ชิ้นนี้ได้อย่างมีประสิทธิภาพ

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์สุมิตร พนาอุดมทรัพย์ สังกัดภาควิชาวิศวกรรมการวัดและควบคุม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ได้ให้คำแนะนำ คำปรึกษาและคอยช่วยเหลือ สนับสนุนการแก้ปัญหาทางด้านไมโครคอนโทรลเลอร์ (Microcontroller) และการเขียนโปรแกรมด้วยภาษา C++ ทำให้พวกข้าพเจ้าได้รับความรู้ วิธีการแก้ปัญหาและแนวทางด้านการเขียนโปรแกรมเป็นอันมาก

ขอขอบพระคุณ ดร.เฉลิมภัณฑ์ ฟองสมุทร สังกัดภาควิชาวิศวกรรมไฟฟ้า มหาวิทยาลัยบูรพา ที่ได้สละเวลาส่วนตัวอันมีค่า มาให้คำปรึกษาและคอยช่วยเหลือสนับสนุนการแก้ปัญหาเกี่ยวกับเทคโนโลยี Internet of Things (IoT) การสร้างโครงข่ายไร้สาย และการจัดทำเว็บไซต์ รวมถึงการใช้ภาษาไพธอน (Python) ทำให้พวกข้าพเจ้าได้รับความรู้ ความเข้าใจ วิธีการแก้ปัญหาและแนวคิดภาพรวมของการทำงานแบบไร้สายเป็นอันมาก

ขอขอบพระคุณคณะอาจารย์ และเจ้าหน้าที่ภาควิชาวิศวกรรมการวัดและควบคุม รวมถึงเพื่อนๆ และพี่น้อง สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ให้ความช่วยเหลือและให้คำแนะนำที่ดีมาโดยตลอด

สุดท้ายนี้คณะผู้จัดทำขอกราบขอบพระคุณผู้ปกครองและครอบครัว ซึ่งให้การสนับสนุนในด้านต่างๆ และคอยเป็นกำลังใจมาโดยตลอดจนสำเร็จการศึกษา คณะผู้จัดทำรู้สึกซาบซึ้งเป็นอย่างยิ่งสำหรับทุกความช่วยเหลือ และขอขอบคุณทุกท่านที่มีส่วนเกี่ยวข้องมา ณ ที่นี้ด้วย

คณะผู้จัดทำ

นายธนกร พานิช

นายศุภเดช กิ่งนอก

นายสมาธิ อนันต์เจริญวัฒน์

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญรูป

สารบัญรูป(ต่อ)

สารบัญตาราง

บทที่ 1

บทนำ

1.1 ความสำคัญและที่มาของปริญญานิพนธ์

ปัจจุบันนี้ถือได้ว่าโลกได้เข้าสู่ยุคโลกาภิวัตน์แบบเต็มตัว กลายเป็นสังคมแห่งการพัฒนา และการเรียนรู้แบบเปิด ซึ่งมีหลากหลายภาคแขนงวิชา เป็นการบูรณาการองค์ความรู้ทุกอย่างเข้าด้วยกันทั้งด้านเทคโนโลยี การสื่อสาร การคมนาคม ฯลฯ เพื่อทำให้เกิดสิ่งใหม่ๆ ที่เป็นประโยชน์ต่อการเรียนรู้และพัฒนาต่อไปในภายหน้า ซึ่งปัจจุบันเทคโนโลยีทางด้านการควบคุมสั่งการระยะไกลผ่านอินเตอร์เน็ตและฐานข้อมูล นั้นเจริญก้าวหน้าอย่างรวดเร็ว และเริ่มเข้ามามีบทบาทกับชีวิตของมนุษย์ในด้านต่างๆ มากขึ้น เช่น ด้านการแพทย์ ด้านอุตสาหกรรมการผลิต ด้านงานสำรวจ ด้านการเกษตร ด้านการขนส่งและสาธารณูปโภค และในอนาคตเทคโนโลยีเหล่านี้ ไม่เพียงอำนวยความสะดวกแก่ผู้ปฏิบัติงานแต่ยังสามารถลดบุคลากรต่างๆที่ไม่จำเป็น ซึ่งอาจไปถึงขั้นเกิดระบบการผลิตในอุตสาหกรรมที่ปราศจากมนุษย์ ทั้งหมดเพื่อเพิ่มความแม่นยำ ลดระยะเวลาในการดำเนินงาน และเพื่อลดการทำงานที่หนักเกินกำลังของมนุษย์

การควบคุมสั่งการภายในบ้านนั้นถือเป็นสิ่งที่เกิดขึ้นบ่อยครั้งในปัจจุบัน แต่ผู้คนนั้นใช้เวลาอยู่ที่บ้านน้อยลง ต้องออกไปทำงานเช้าขึ้นและกลับจากการทำงานช้าลง อันเนื่องมากจากความเร่งรีบและการแข่งขันในการทำงาน ปัญหาการจราจรที่ติดขัด เป็นต้น การทำงานของอุปกรณ์บางอย่างนั้นไม่สามารถเริ่มทำงานได้หากผู้ใช้ไม่ได้อยู่ภายในตัวบ้าน ในบางครั้งเกิดการต้องการสั่งงานอุปกรณ์ไฟฟ้าก่อนกลับบ้านหรือหลงลืมปิดอุปกรณ์หลังออกจากบริเวณบ้านไปแล้ว จากการใช้ชีวิตที่เปลี่ยนไปและมองเห็นปัญหาในจุดนี้ จึงเกิดเป็นปริญญานิพนธ์นี้ขึ้นมาโดยการใช้ มินิคอมพิวเตอร์ทำงานร่วมกับไมโครคอนโทรลเลอร์ที่มีการเชื่อมต่ออินเตอร์เน็ต เพื่อแก้ปัญหาต่างๆ ที่เกิดขึ้น อำนวยความสะดวกให้กับชีวิตประจำวันของผู้คนให้มีความสะดวกสบาย และลดการเกิดเหตุเพลิงไหม้จากการลืมปิดการทำงานของอุปกรณ์ไฟฟ้าในครัวเรือน

1.2 วัตถุประสงค์ในการทำปริญญานิพนธ์

1. เพื่อศึกษาและทำความเข้าใจเกี่ยวกับการสร้างโค้ด เพื่อควบคุมอุปกรณ์ต่างๆ ให้เป็นไปตามที่ต้องการ โดยใช้ภาษา C++ ในการโปรแกรม

2. เพื่อศึกษาและทำความเข้าใจเกี่ยวกับการใช้งานมินิคอมพิวเตอร์ Raspberry Pi 3 รวมไปถึงการใช้ฟังก์ชันต่างๆ ที่มีอยู่ เพื่อให้เกิดประโยชน์สูงสุด

3. เพื่อศึกษาและทำความเข้าใจเกี่ยวกับการใช้ฐานข้อมูล MySQL เพื่อรวบรวม และจัดเก็บข้อมูลให้เข้าถึงได้ง่าย และลดขนาดของข้อมูลที่มีความซับซ้อน

4. เพื่อสร้างระบบต้นแบบของกล่องควบคุมอัตโนมัติที่สามารถควบคุม และสั่งการอุปกรณ์ภายในบ้านด้วยระยะไกล ซึ่งสามารถทำงานได้โดยอัตโนมัติและยังสามารถเก็บข้อมูลต่างๆ ที่เกิดขึ้นได้ด้วย อีกทั้งสามารถต่อยอดไปใช้กับโรงงานอุตสาหกรรมขนาดเล็กและกลางได้ในอนาคต

1.3 ขอบเขตการศึกษา

1. ปริญญานิพนธ์นี้ใช้ Raspberry Pi ในการควบคุมและเก็บค่าต่างๆ เพียงตัวเดียวเท่านั้น ทำให้ลดต้นทุนและอุปกรณ์ระหว่างทำปริญญานิพนธ์ได้

2. ในส่วนของการเก็บข้อมูลเข้าระบบฐานข้อมูลนั้น จะพัฒนาผ่านคอมพิวเตอร์ส่วนบุคคล (PC) โดยใช้โปรแกรม MySQL ซึ่งจะทำให้ระหว่างการพัฒนาไม่ต้องใช้งาน Raspberry Pi เพื่อให้ส่วนการควบคุมสามารถใช้ Raspberry Pi ได้อย่างเต็มที่

3. ปริญญานิพนธ์นี้เป็นขนาดจำลองของระบบควบคุมภายในบ้าน

1.4 รายละเอียดของปริญญานิพนธ์

เนื้อหาที่จะกล่าวในปริญญานิพนธ์ฉบับนี้ประกอบด้วย

บทที่ 1 บทนำ กล่าวถึง วัตถุประสงค์ ขั้นตอนการศึกษา ขอบเขตของการศึกษา รายละเอียดของปริญญานิพนธ์ในแต่ละบท ประโยชน์ที่คาดว่าจะได้รับ

บทที่ 2 ทฤษฎีและความรู้ที่เกี่ยวข้องของหลักการ และทฤษฎีที่เกี่ยวข้องกับอุปกรณ์ที่ใช้ในปริญญานิพนธ์ การคำนวณ และโปรแกรมที่ใช้ในปริญญานิพนธ์

บทที่ 3 การออกแบบวงจรและการเขียนโปรแกรม

บทที่ 4 การทดลองและผลการทดลองของระบบย่อยและระบบโดยรวม

บทที่ 5 สรุปและวิจารณ์ผลการดำเนินงาน ปัญหาที่เกิดขึ้นจริง แนวทางในการปรับปรุงและการพัฒนาต่อยอด

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้ใช้ความคิดในการทำปริญญานิพนธ์ เพื่อเป็นการฝึกให้มีการทำงานเป็นระบบ มีความสามัคคี และมีความรับผิดชอบมากขึ้น

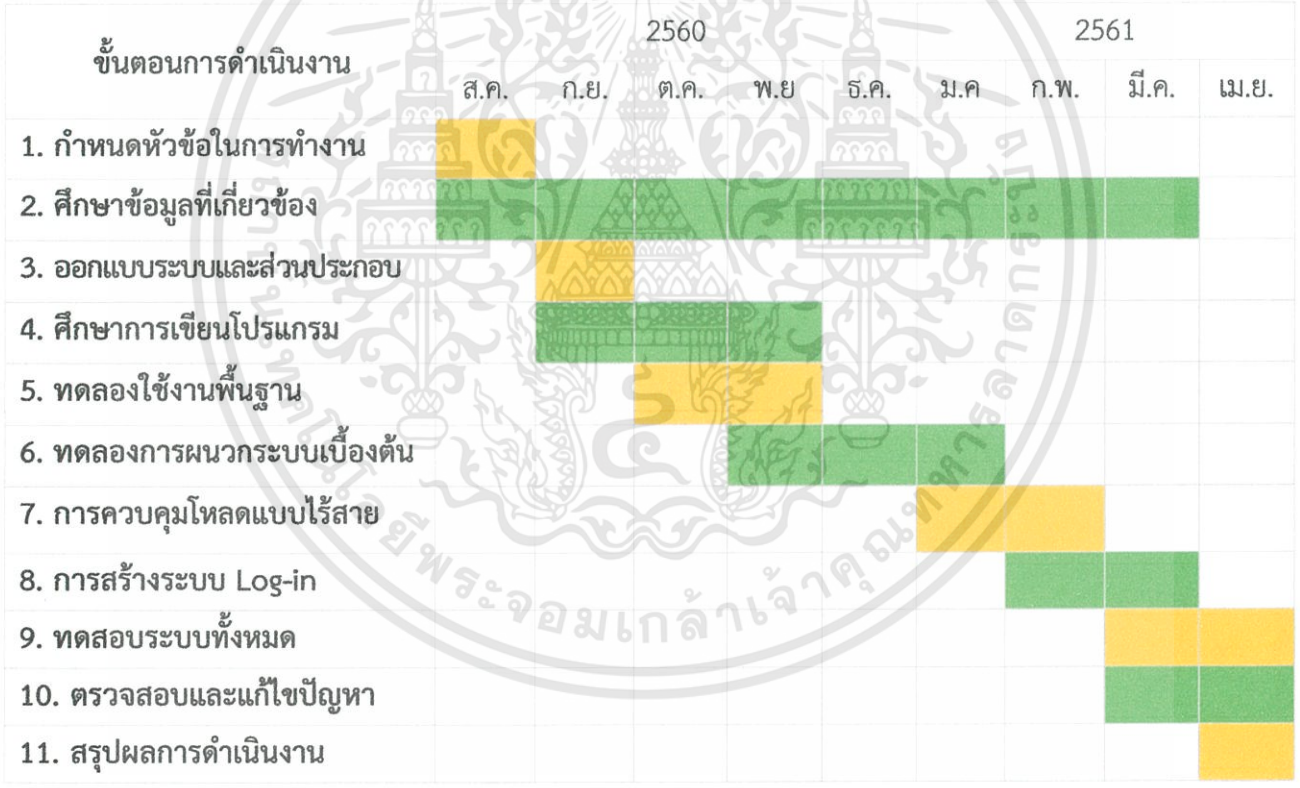
2. เข้าใจการเขียนโปรแกรมด้วยคำสั่งภาษา Python และการควบคุมอุปกรณ์ต่างๆ ผ่าน Raspberry Pi มากยิ่งขึ้น

3. เข้าใจการใช้งานระบบฐานข้อมูล (Database) และคำสั่งใน MySQL เพื่อใช้ในการเก็บค่าและเรียกดูผ่านหน้าเว็บไซต์

4. รู้ปัญหาและแนวทางในการพัฒนาระบบควบคุมในบ้านเพื่อให้สอดคล้องกับการใช้งานจริง

1.6 แผนการดำเนินงาน

ตารางที่ 1.1 แผนการดำเนินโครงงานระหว่างเดือนสิงหาคม 2560 ถึงเดือนเมษายน 2561



บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 อินเตอร์เน็ตในทุกสิ่ง Internet of Things (IOT)

อินเตอร์เน็ตในทุกสิ่ง หมายถึง การที่อุปกรณ์หรือสิ่งต่างๆ ที่ได้ถูกเชื่อมโยงเข้าสู่โลกอินเตอร์เน็ต ทำให้มนุษย์สามารถสั่งการควบคุมการใช้งานอุปกรณ์ต่างๆ ผ่านทางเครือข่ายอินเตอร์เน็ต เช่น การเปิด-ปิด อุปกรณ์เครื่องใช้ไฟฟ้า (การสั่งการเปิดไฟฟ้าภายในบ้านด้วยการเชื่อมต่ออุปกรณ์ควบคุม เช่น รถยนต์ โทรศัพท์มือถือ เครื่องมือสื่อสาร เครื่องมือทางการเกษตร อาคาร บ้านเรือน เครื่องใช้ไฟฟ้าในชีวิตประจำวันต่างๆ ผ่านเครือข่ายอินเตอร์เน็ต เป็นต้น

2.2 ไมโครคอนโทรลเลอร์ (Microcontroller)

2.2.1 โมดูลไวไฟ ESP8266

โมดูลไวไฟ ESP8266 เป็นการรวมกันของชิป Microcontroller กับ Wi-Fi Module ผลิตโดยบริษัท Espressif System โดย ESP8266 เป็นชื่อของชิปไอซีบนบอร์ดของโมดูล ซึ่งไอซี ESP8266 ไม่มีพื้นที่โปรแกรม (Flash Memory) ในตัว ต้องใช้ไอซีภายนอก (External Flash Memory) ในการเก็บโปรแกรม ที่ใช้เชื่อมต่อผ่านโปรโตคอล SPI ทำให้โมดูล ESP8266 มีพื้นที่ในการโปรแกรมมากกว่าไอซีไมโครคอนโทรลเลอร์อื่นๆ โครงสร้างภายนอกและตำแหน่งขาของ ESP-1 ดังรูปที่ 2.1

รูปที่ 2.1 โครงสร้างภายนอกและตำแหน่งขาของ ESP-1

ESP8266 ทำงานที่แรงดันไฟกระแสตรง 3.3 ถึง 3.6V หากต้องการนำไปใช้กับอุปกรณ์อื่นๆ ที่ใช้แรงดันไฟกระแสตรง 5V จำเป็นต้องใช้วงจรแบ่งแรงดันมาช่วย เพื่อป้องกันไม่ให้ตัวโมดูลเสียหาย กระแสไฟฟ้าที่โมดูลใช้งานสูงสุดคือ 200mA มีความถี่ในการทำงาน (Clock) 40MHz ทำให้สามารถใช้งานกับอุปกรณ์ที่ใช้ความถี่สูง เช่น จอ LCD ได้ดีกว่าไมโครคอนโทรลเลอร์ชนิดอื่นๆ

ขาพิน (PIN) ของ โมดูลไวไฟ ESP8266 ดังรูปที่ 2.2 แบ่งได้ดังนี้

VCC เป็นขาสำหรับจ่ายไฟเข้าเพื่อให้โมดูลทำงานได้ ซึ่งแรงดันไฟกระแสตรงที่ใช้งานคือ 3.3 ถึง 3.6V และขากราวด์ (GND)

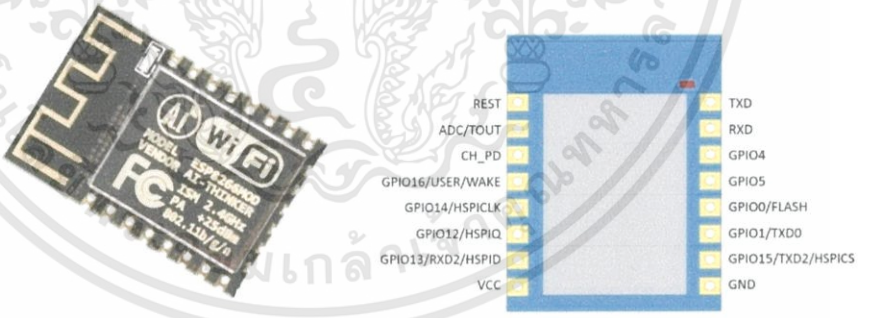
Reset และ CH_PD (หรือ EN) เป็นขาที่ต้องต่อไฟบวก เพื่อเรียกใช้งานทั้ง 2 ขานี้สามารถนำมาใช้รีเซ็ตโมดูลได้ แต่แตกต่างตรงที่ Reset สามารถปล่อยลอยไว้ได้ แต่ CH_PD (หรือ EN) จำเป็นต้องต่อเข้ากับไฟบวกเท่านั้น เมื่อขานี้ไม่ต่อเข้าไฟบวกโมดูลจะไม่ทำงานทันที

GPIO เป็นขาดิจิตอล Input/Output ทำงานที่แรงดัน 3.3V

GPIO15 เป็นขาที่ต้องต่อลงกราวด์ (GND) เท่านั้น เพื่อให้โมดูลทำงานได้

GPIO0 เป็นขาสำหรับการเลือกโหมดทำงาน หากนำขานี้ลงกราวด์ (GND) จะเข้าโหมดโปรแกรม หากลอยไว้หรือนำเข้าไฟบวกจะเข้าโหมดการทำงานปกติ

ADC เป็นขาอนาล็อกอินพุต รับแรงดันได้สูงสุดที่ 1V ขนาด 10-bit การนำไปใช้งานกับแรงดันที่สูงกว่าต้องใช้วงจรแบ่งแรงดันเข้ามาช่วย



รูปที่ 2.2 โครงสร้างภายนอกและตำแหน่งขาของ ESP-12E

ESP8266 มีอยู่ด้วยกันหลายรุ่น โดยรุ่นที่ได้รับความนิยมและได้รับการนำไปติดตั้งบน NodeMCU Platform คือ ESP-12E ซึ่งเป็นรุ่นที่มีการเพิ่มขาตรงส่วนท้ายของแผ่นปริ้นจำนวน 6 ขา ได้แก่ SCLK MOSI GPIO10 GPIO09 MISO และ CS0 เพื่อเป็นขาที่ใช้เชื่อมต่อผ่านโปรโตคอล SPI เนื่องจากในรุ่นอื่นๆ ต้องใช้ขา GPIO อื่นในการใช้โปรโตคอล SPI ทำให้ประหยัดขาที่ใช้งานได้

2.2.2 คุณสมบัติของ ESP8266

SDIO 2.0, SPI, UART

32-pin QFN package

Integrated RF switch, balun, 24dBm PA, DCXO, and PMU

Integrated RISC processor, on-chip memory and external memory interfaces

Integrated MAC/baseband processor

Quality of Service management

12S interface for high fidelity audio application

On-chip low-dropout linear regulators for all internal supplies

Proprietary spurious-free clock generation architecture

Integrated WEP, TKIP, AES, and WAPI engines

Wi-Fi 802.11 b/g/n and Wi-Fi Direct (P2P), soft-AP

Integrated TCP/IP protocol stack

Integrated TR switch, balun, LNA, power amplifier and matching network

Integrated PLLs, regulators, DCXO and power management units

+19.5dBm output power in 802.11b mode

Power down leakage current of <10μA

Integrated low power 32-bit CPU could be used as application processor

SDIO 1.2/2.2, SPI, UART

STBC, 1x1 MIMO, 2x1 MIMO

A-MPDU & A-MSDU aggregation & 0.4ms guard interval

Wake up and transmit packets in <2ms

Standby power consumption of <1.0mW (DTM3)

2.2.3 NodeMCU

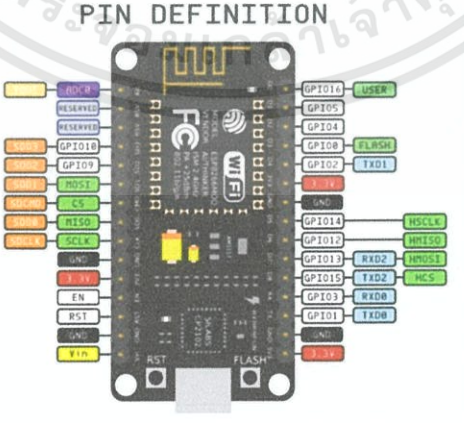
NodeMCU คือ แพลตฟอร์มหนึ่งที่ประกอบไปด้วย Development Kit (Hardware) ซึ่งรวมเอาชิปประมวลผล ESP รุ่นต่างๆ กับอุปกรณ์อื่นๆ เช่น พอร์ตจ่ายไฟเลี้ยง พอร์ตเชื่อมต่อต่างๆ (Connector) วงจรไฟเลี้ยง เป็นต้น มาไว้บน PCB เดียวกัน ดังรูปที่ 2.3 เพื่อให้ผู้ใช้สามารถใช้งานทันทีและสะดวกต่อการพัฒนามากขึ้น และ Firmware (Software) ที่เป็น Open Source เขียนโปรแกรมด้วยภาษา Lua เป็นพื้นฐาน มีการติดตั้งโมดูลไวไฟ ESP8266 ซึ่งเป็นทั้งตัว CPU หลักและ

ชิปเชื่อมต่ออินเตอร์เน็ตแบบไร้สาย (Wi-Fi) ซึ่งใน NodeMCU V.1 นั้นจะใช้งานโมดูลที่เป็นชิปรุ่น ESP-12 เลือกใช้ NodeMCU V.2 เป็น NodeMCU รุ่นพัฒนาต่อยอดจาก V.1 ใน NodeMCU V.2 จะใช้โมดูลที่เป็นชิป ESP-12E ซึ่งเป็นรุ่นใหม่ล่าสุดแทน มีการปรับปรุง PCB ให้แคบลงและยาวขึ้น ทำให้สามารถเสียบใช้งาน Protoboard ได้ทันที เปลี่ยนชิปแปลง USB2Serial จาก CH340 ในตัว V.1 เป็น CP2102 เพื่อลดปัญหาที่มีกับระบบปฏิบัติการ (OS) ของคอมพิวเตอร์

รูปที่ 2.3 โครงสร้างภายนอกของ NodeMCU V.2

2.2.4 คุณสมบัติและการต่อใช้งาน NodeMCU

ลักษณะเด่นของ NodeMCU คือ มีความคล้ายคลึงกับไมโครคอลโทรลเลอร์ Arduino ที่มีพอร์ต GPIO ติดมาในตัว ดังรูปที่ 2.4 ทำให้สามารถเขียนโปรแกรมเพื่อใช้งาน GPIO ได้โดยตรงไม่ต้องผ่านอุปกรณ์อื่นๆ มีการพัฒนาจนสามารถใช้งานร่วมกับโปรแกรม Arduino IDE ทำให้สามารถพัฒนาโปรแกรมด้วยภาษา C/C++ เพิ่มขีดความสามารถในการพัฒนามากขึ้น โดยเฉพาะด้าน IoT เช่น Web Server ขนาดเล็ก, การเปิด-ปิดไฟผ่าน Wi-Fi เป็นต้น

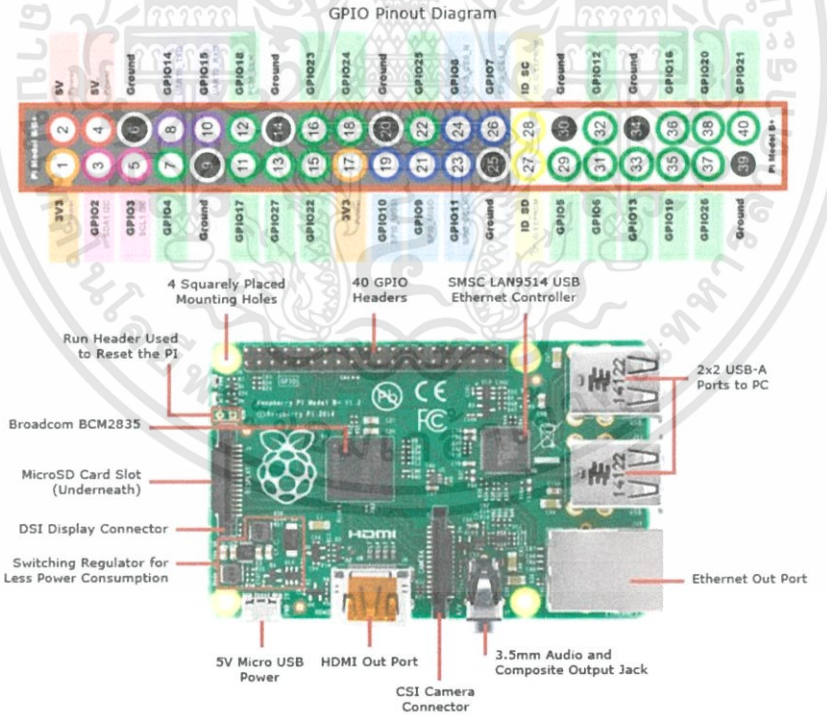


รูปที่ 2.4 ตำแหน่งขาของ NodeMCU V.2

NodeMCU นั้นมี GPIO จำนวน 10 ขา (Pin) ที่สามารถเป็นได้ทั้ง PMW, I2C และ 1-wire ส่วนขา ADC จะเป็นขาแยกเฉพาะ มี USB-TTL ในตัว มี PCB Antenna สำหรับรับส่งสัญญาณไร้สาย ใช้งานคอนเนกเตอร์ (Connector) แบบ Micro-USB สำหรับจ่ายแรงดันไฟเลี้ยงกระแสตรง +5V และดาวโหลดเฟิร์มแวร์หรือโปรแกรม โดยคุณสมบัติทางเทคนิคจะอ้างอิงตามชิปของโมดูล ESP8266 เป็นหลัก สำหรับ NodeMCU V.2 ที่เลือกใช้คือ ชิป ESP-12E

2.2.5 Mini Computer Raspberry Pi 3 Model B

Raspberry Pi เป็นบอร์ดคอมพิวเตอร์ขนาดเล็กถูกพัฒนาโดย Raspberry Pi Foundation มีความสามารถในการทำงานทั่วไป เช่น งานเอกสาร, ดูหนังฟังเพลง, ใช้ทำ Web Server เป็นต้น ทั้งยังมีความสามารถในการติดต่อสื่อสาร และควบคุมอุปกรณ์อิเล็กทรอนิกส์ต่างๆ ผ่าน GPIO ของตัวบอร์ด ดังรูปที่ 2.5 Raspberry Pi ใช้ระบบปฏิบัติการ Linux เป็นพื้นฐาน มีระบบปฏิบัติการที่นิยม และนำมาใช้คือ Raspbian ซึ่งเป็นระบบปฏิบัติการอย่างเป็นทางการ ภาษาพื้นฐานที่ใช้เขียนโปรแกรมบน Raspberry Pi คือ Python แต่ยังสามารถเขียนโปรแกรมต่างๆ ได้อีกทั้ง C/C++ , Shell Script เป็นต้น โดยอาศัยโปรแกรม Complier แปลงเป็นภาษา Python



รูปที่ 2.5 โครงสร้างภายนอกและพอร์ตการเชื่อมต่อของ Raspberry Pi 3 Model B

2.2.6 คุณสมบัติและการต่อใช้งาน Raspberry Pi 3

Raspberry Pi 3 Model B ใช้งาน CPU แบบ 4 แกนประมวลผล สามารถเชื่อมต่อ Wi-Fi 802.11n 2.4GHz และ Bluetooth 4.1 (BLE) ทำให้สามารถเชื่อมต่อแบบไร้สายกับอุปกรณ์อื่นๆ ได้ เช่น NodeMCU เป็นต้น ใช้งานแหล่งเก็บข้อมูลแบบ MicroSD Card สูงสุด 64GB รองรับการเชื่อมต่อ USB, HDMI, 3.5mm Audio, Ethernet และมี Connector สำหรับจอแสดงผลและกล้องดิจิตัลแยกอีก 1 ชุด รับไฟเลี้ยงจากพอร์ต (Connector) แบบ Micro USB +5V 2.5mA มีขาทั้งหมด 40 ขา เป็น GPIO 26 ขา สำหรับการเชื่อมต่อกับอุปกรณ์ภายนอก

SoC: Broadcom BCM2837

CPU: 4core ARM Cortex-A53, 1.2GHz

GPU: Broadcom VideoCore IV

RAM: 1GB LPDDR2 900MHz

Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless

Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy

Storage: microSD Card

GPIO: 40-pin Header, populated

Ports: HDMI, 3.5mm analog audio-video jack, 4x USB, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)

2.3 ภาษา Python

2.3.1 ข้อมูลเบื้องต้นของภาษา Python

ภาษา Python เป็นภาษาเขียนโปรแกรมระดับสูงที่ใช้กันอย่างกว้างขวางในการเขียนโปรแกรมสำหรับวัตถุประสงค์ทั่วไปภาษา Python นั้นเป็นภาษาแบบ Interprete ที่ถูกออกแบบโดยมีปรัชญาที่จะทำให้โค้ดอ่านได้ง่ายขึ้น และโครงสร้างของภาษานั้นจะทำให้โปรแกรมเมอร์สามารถเข้าใจแนวคิดการเขียนโค้ดโดยใช้บรรทัดที่น้อยลงกว่าภาษาอย่าง C++ และ Java ซึ่งภาษานั้นถูกกำหนดให้มีโครงสร้างที่ตั้งใจให้การเขียนโค้ดเข้าใจง่ายทั้งในโปรแกรมเล็กไปจนถึงโปรแกรมขนาดใหญ่

ภาษา Python เป็นภาษาที่สามารถสร้างงานได้หลากหลายกระบวนทัศน์ (Multi-Paradigm Language) โดยจะมองอะไรที่มากกว่าการ Coding เพื่อนำมาใช้ในงานตามรูปแบบเดิม แต่จะเป็นการนำเอาหลักการของกระบวนทัศน์แบบ Object-oriented Programming, Structured Programming, Functional Programming และ Aspect-oriented Programming นำเอามาใช้ทั้ง

แบบเดียว และนำมาใช้ร่วมกัน ซึ่ง Python นั้นเป็นภาษาที่มีการตรวจสอบชนิดตัวแปรแบบยืดหยุ่น (Dynamically Type-Checked) และใช้ Garbage Collection ในการจัดการหน่วยความจำ

2.3.2 ข้อเด่นของภาษา Python

ง่ายต่อการเรียนรู้ โดยภาษา Python มีโครงสร้างของภาษาไม่ซับซ้อนเข้าใจง่าย ซึ่งโครงสร้างภาษา Python จะคล้ายกับภาษาซีมาก เพราะภาษา Python สร้างขึ้นมาโดยใช้ภาษาซี ทำให้ผู้ที่คุ้นเคยภาษาซีอยู่แล้วใช้งานภาษา Python ได้ไม่ยาก นอกจากนี้โดยตัวภาษาเองมีความยืดหยุ่นสูง ทำให้การจัดการกับงานด้านข้อความ และ Text File ได้เป็นอย่างดี

ไม่ต้องเสียค่าใช้จ่ายใดๆ ทั้งสิ้น เพราะตัวแปรภาษา Python อยู่ภายใต้ลิขสิทธิ์ Python Software Foundation License (PSFL) ซึ่งเป็นของ Python Software Foundation (PSF) ซึ่งมีลักษณะคล้ายกับลิขสิทธิ์แม่แบบอย่าง General Public License (GPL) ของ Free Software Foundation (FSF)

ใช้ได้หลายแพลตฟอร์ม ในช่วงแรกภาษา Python ถูกออกแบบใช้งานกับระบบ Unix อยู่ก็จริง แต่ในปัจจุบันได้มีการพัฒนาตัวแปลภาษา Python ให้สามารถใช้กับระบบปฏิบัติการอื่นๆ อาทิเช่น Linux Platform, Windows Platform, OS/2, Amiga, Mac OS X และรวมไปถึงระบบปฏิบัติการที่ .NET Framework, Java Virtual Machine ทำงานได้ ซึ่งใน Nokia Series 60 ก็สามารถทำงานได้เช่นกัน

ภาษา Python ถูกสร้างขึ้นโดยได้รวบรวมเอาส่วนดีของภาษาต่างๆ เข้ามาไว้ด้วยกัน อาทิเช่น ภาษา ABC, Modula-3, Icon, ANSI C, Perl, Lisp, Smalltalk และ Tcl

Python สามารถรวมการพัฒนาของระบบเข้ากับ COM, .NET และ CORBA Objects

สำหรับ Java Libraries แล้วสามารถใช้ Jython เพื่อทำการพัฒนาซอฟต์แวร์จากภาษา Python สำหรับ Java Virtual Machine

สำหรับ .NET Platform แล้ว สามารถใช้ IronPython ซึ่งเป็นการพัฒนาของ Microsoft เพื่อจะทำให้ Python นั้นสามารถทำงานได้บน .Net Framework ซึ่งใช้ชื่อว่า Python for .NET

Python นั้นสนับสนุน Internet Communications Engine (ICE) และการรวมกันของเทคโนโลยีอื่นๆ อีกมากมายในอนาคต

บางครั้งนักพัฒนาอาจจะพบว่า Python ไม่สามารถทำงานบางอย่างได้ แต่นักพัฒนาต้องการให้มันทำงานได้ ก็สามารถพัฒนาเพิ่มได้ในรูปแบบของ Extension Modules ซึ่งอยู่ในรูปแบบของโปรแกรม C หรือ C++ หรือใช้ SWIG หรือ Broost.Python

ภาษา Python สามารถพัฒนาเป็นภาษาประเภท Server Side Script คือ การทำงานของภาษา Python จะทำงานด้านฝั่ง Server แล้วส่งผลลัพธ์กลับมายัง Client ทำให้มีความปลอดภัยสูง และยังสามารถใช้ภาษา Python นำมาพัฒนา Web Service ได้อีกด้วย

ใช้พัฒนาระบบบริหารการสร้างเว็บไซต์สำเร็จรูปที่เรียกว่า Content Management Systems (CMS) ซึ่ง CMS ที่มีชื่อเสียงมาก และเบื้องหลังทำงานด้วยภาษา Python คือ Plone

2.4 phpMyAdmin

phpMyAdmin เป็นส่วนต่อประสานที่สร้างโดยภาษา PHP ซึ่งใช้จัดการฐานข้อมูล MySQL ผ่าน Web Browser โดยสามารถที่จะทำการสร้างฐานข้อมูลใหม่ หรือทำการสร้าง TABLE ใหม่ๆ และยังมี Function ที่ใช้สำหรับการทดสอบการ Query ข้อมูลด้วยภาษา SQL พร้อมกันนั้น ยังสามารถทำการ Insert Delete Update หรือแม้กระทั่งใช้คำสั่งต่างๆ เหมือนกันกับการใช้ภาษา SQL ในการสร้างตารางข้อมูล (Table)

phpMyAdmin เป็นโปรแกรมประเภท MySQL Client ตัวหนึ่งที่ใช้ในการจัดการข้อมูล MySQL ผ่าน Web Browser ได้โดยตรง phpMyAdmin ตัวนี้จะทำงานบน Web Server เป็น PHP Application ที่ใช้ควบคุมจัดการ MySQL Server ความสามารถของ phpMyAdmin คือ

1. สร้างและลบ Database
2. สร้างและจัดการ Table เช่น แทรก, ลบและแก้ไข Record, ลบ Table, แก้ไข Field
3. โหลด Text File เข้าไปเก็บเป็นข้อมูลใน Table ได้
4. หาผลสรุป (Query) ด้วยคำสั่ง SQL

2.5 MySQL

MySQL คือ Open Source Relational Database Management System (RDBMS) ซึ่งตอนแรก MySQL นั้นเป็นของบริษัท MySQL AB แต่ในปัจจุบันผู้ที่เป็นเจ้าของ MySQL คือ บริษัท Oracle โดย MySQL นั้นถือว่าเป็นฐานข้อมูลที่ได้รับความนิยมในการนำมาใช้งานบน Web Application เป็นอย่างมาก ซึ่งเป็นส่วนหนึ่งในสิ่งที่เรียกว่า LAMP (Linux, Apache, MySQL และ PHP) โดยตัวอย่าง Web Application ที่มีการใช้ MySQL เช่น TYPO3, Joomla, WordPress, phpBB, MyBB, Drupal รวมไปทั้งเว็บไซต์ขนาดใหญ่ที่มีการใช้ MySQL ในส่วนหนึ่งของ Production เช่น Wikipedia, Google (ไม่ใช่ส่วนของการค้นหา), Facebook, Twitter, Flickr, Nokia.com และ YouTube เป็นต้น

2.5.1 คุณสมบัติของ MySQL

1. สนับสนุน Cross-Platform Support
2. รองรับ Stored Procedures
3. รองรับ Triggers และ Cursors
4. สนับสนุน Information Schema
5. สนับสนุน SSL
6. รองรับการทำ Query Caching
7. รองรับการทำ Sub-SELECTs
8. รองรับการทำ Replication ทั้งแบบ Master-Master และ Master-Slave
9. Full-Text Indexing และ Searching Using MyISAM Engine
10. รองรับ Unicode

2.6 Relay

Relay หรือรีเลย์ คือ อุปกรณ์ที่เปลี่ยนพลังงานไฟฟ้าเป็นพลังงานแม่เหล็ก เพื่อใช้ดึงดูดหน้าสัมผัสของรีเลย์ให้เปลี่ยนสถานะ ทำหน้าที่เป็นสวิตช์ ตัด-ต่อ วงจร โดยป้อนกระแสไฟฟ้าให้กับขดลวด เพื่อทำการปิดหรือเปิดหน้าสัมผัส เมื่อหยุดจ่ายไฟ หน้าสัมผัสภายในรีเลย์จะกลับเป็นสถานะเริ่มต้น NC/NO ซึ่งสามารถนำรีเลย์ไปประยุกต์ใช้ในการควบคุมวงจรต่างๆ ให้ทำงานตามที่ต้องการ

โดยรีเลย์มักจะถูกนำมาใช้ในงานที่ต้องการควบคุม หรือตัดต่อวงจรไฟฟ้าที่มีแรงดันหรือกระแสสูง ด้วยตัวควบคุมที่มีแรงดันต่ำกว่า เช่น การเปิดปิดพัดลมแรงดันไฟกระแสสลับ 220V ด้วย Microcontroller ที่ใช้แรงดันไฟกระแสตรง 5V

2.6.1 ข้อควรระวังในการใช้งาน

1. แรงดันใช้งาน หรือแรงดันที่ทำให้รีเลย์ทำงานได้ ที่ตัวรีเลย์จะระบุค่าแรงดันใช้งานไว้ เช่น 12VDC คือต้องใช้แรงดันไฟกระแสตรง 12V เท่านั้น หากใช้มากกว่านี้ขดลวดภายในตัวรีเลย์อาจจะขาดได้ หรือหากใช้แรงดันต่ำกว่ามาก รีเลย์จะไม่ทำงาน

2. การใช้งานกระแสของหน้าสัมผัส ที่ตัวรีเลย์จะระบุไว้ เช่น 10A 220VAC คือ หน้าสัมผัสของรีเลย์นั้นสามารถทนแรงดันไฟกระแสสลับ 220V ได้ที่ 10A ในการใช้งานจริงควรใช้ไม่เกินจากค่าที่กำหนดหรือต่ำกว่าเล็กน้อยเพื่อป้องกันความเสียหายที่จะเกิดขึ้นกับหน้าสัมผัส

3. จำนวนหน้าสัมผัสการใช้งาน ต้องคำนึงถึงจำนวนและชนิดของหน้าสัมผัสที่ต้องการใช้งาน เพราะรีเลย์แต่ละตัวนั้นมีจำนวนและลักษณะของหน้าสัมผัสไม่เหมือนกัน

2.7 Relay Module 8 Channel

เป็นชุดอุปกรณ์ที่รวมเอารีเลย์จำนวน 8 ตัวไว้บนแผ่นวงจรเดียวกัน เพื่อให้สามารถใช้งานกับตัว Microcontroller ได้สะดวกและปลอดภัยมาขึ้น โดยมีลักษณะการรวมเอากราวด์ ไฟเลี้ยงและสัญญาณขาเข้ามารวมไว้ในจุดเดียวเพื่อเชื่อมต่อได้โดยง่าย มีการส่งสัญญาณไปยังรีเลย์ด้วย Opto-Isolator และแยกลายวงจรออกจากกันช่วยป้องกันไม่ให้เกิดกระแสไหลย้อนกลับไปยังตัว Microcontroller ซึ่งจะทำให้อุปกรณ์เกิดความเสียหายได้

2.7.1 ข้อควรระวังในการใช้งาน

1. การรองรับการทำงานกับ Microcontroller ชนิดต่างๆ

2. แรงดันใช้งาน หรือแรงดันที่ทำให้รีเลย์ทำงานได้ ที่ตัวรีเลย์จะระบุค่าแรงดันใช้งานไว้ เช่น 12VDC คือ ต้องใช้แรงดันไฟกระแสตรงที่ 12V เท่านั้น หากใช้มากกว่านี้ขดลวดภายในตัวรีเลย์อาจจะขาดได้ หรือหากใช้แรงดันต่ำกว่ามากรีเลย์จะไม่ทำงาน

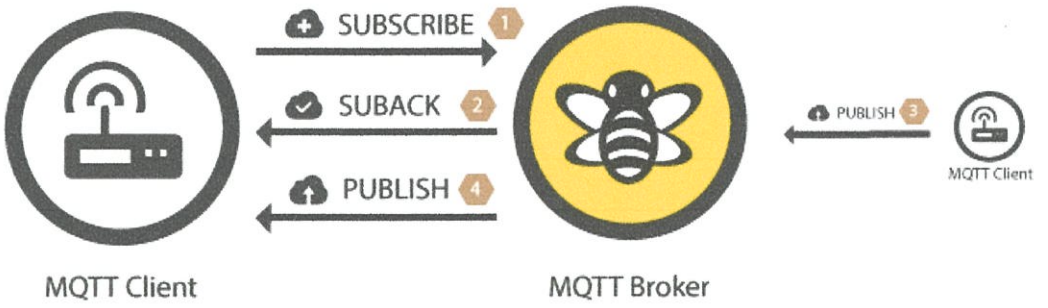
3. การใช้งานกระแสของหน้าสัมผัส ที่ตัวรีเลย์จะระบุไว้ เช่น 10A 220VAC คือ หน้าสัมผัสของรีเลย์นั้นสามารถทนแรงดันไฟกระแสสลับ 220V ได้ที่ 10A ในการใช้งานจริงควรใช้ไม่เกินจากค่าที่กำหนดหรือต่ำกว่าเล็กน้อยเพื่อป้องกันความเสียหายที่จะเกิดขึ้นกับหน้าสัมผัส

4. จำนานหน้าสัมผัสการใช้งาน ต้องคำนึงถึงจำนวนและชนิดของหน้าสัมผัสที่ต้องการใช้งาน เพราะรีเลย์แต่ละตัวนั้นมีจำนวนและลักษณะของหน้าสัมผัสไม่เหมือนกัน

2.8 Message Queuing Telemetry Transport (MQTT)

Message Queuing Telemetry Transport (MQTT) เป็น Protocol ที่ออกแบบมาเพื่อการเชื่อมต่อแบบ M2M (Machine-to-Machine) คือ อุปกรณ์กับอุปกรณ์ สนับสนุนเทคโนโลยี IOT (Internet of Things) คือ เทคโนโลยีที่อินเตอร์เน็ตเชื่อมต่อกับอุปกรณ์ต่างๆ เช่น โทรศัพท์มือถือ รถยนต์ โทรทัศน์ ตู้เย็น เข้ากับอินเตอร์เน็ต ทำให้สามารถเชื่อมโยงสื่อสารกับอุปกรณ์ต่างๆ ได้ โดยผ่านเครือข่ายอินเตอร์เน็ต ซึ่งจะทำให้มนุษย์สามารถควบคุมอุปกรณ์ต่างๆ จากที่อื่นได้ เช่น การสั่งเปิด-ปิด ไฟในบ้านจากสถานที่อื่นๆ

เนื่องจากโปรโตคอลตัวนี้มีความรวดเร็ว ออกแบบมาเพื่อใช้งานกับอุปกรณ์อิเล็กทรอนิกส์ขนาดเล็ก การรับส่งข้อมูลในเครือข่ายที่มีขนาดเล็ก แบนด์วิดธ์ต่ำ ใช้หลักการแบบ Publisher และ Subscriber คล้ายกับหลักการที่ใช้ใน Web Service ที่ต้องใช้ Web Server เป็นตัวกลางระหว่างคอมพิวเตอร์ของผู้ใช้ แต่ MQTT จะใช้ตัวกลางที่เรียกว่า Broker เพื่อทำหน้าที่ จัดการคิว รับ-ส่งข้อมูลระหว่างอุปกรณ์ และทั้งในส่วนที่เป็น Publisher และ Subscriber



รูปที่ 2.6 หลักการทำงานของ MQTT

จากรูปที่ 2.6 จะเห็นได้ว่า Topic จะเป็นตัวอ้างอิงหลัก ข้อมูลที่จะ Publisher ออกไปยัง Broker จะต้องมี Topic กำกับไว้เสมอ ทางฝ่าย Subscriber ก็จะอ้างถึง Topic เพื่อเรียกข้อมูลที่ต้องการ เหมือนกับการสมัครเป็นสมาชิกของหนังสือพิมพ์ฉบับหนึ่ง ชื่อของหนังสือก็เปรียบเหมือน Topic และผู้ผลิตก็คือ Publisher เมื่อถึงเวลาที่หนังสือเสร็จ ผู้ส่ง Broker ก็จะนำหนังสือพิมพ์มาส่ง

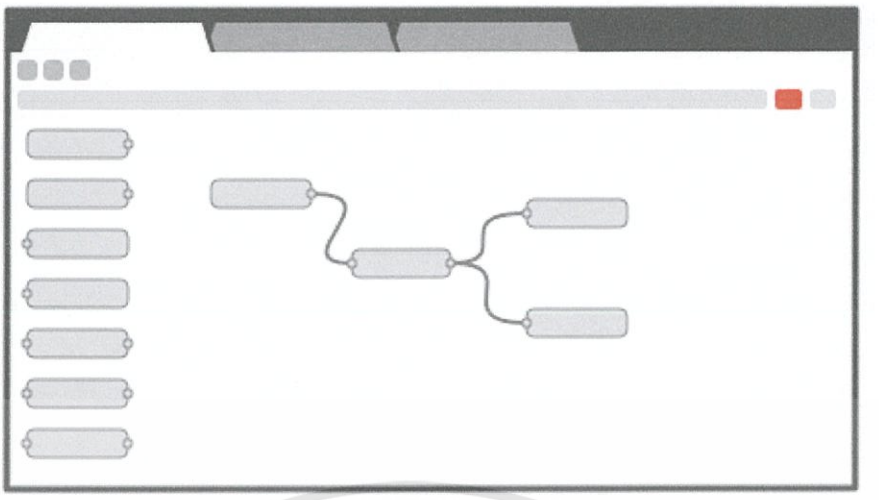
องค์ประกอบของ MQTT Protocol จะประกอบไปด้วย Broker Publisher และ Subscriber แต่ละอย่างก็จะทำหน้าที่แตกต่างกันออกไป โดย Broker ทำหน้าที่เป็นตัวกลางคอยจัดการกับข้อความ โดยอ้างอิงจาก Topic Publisher จะทำหน้าที่คอยส่งข้อมูลไปยังหัวข้อนั้นๆ Subscriber จะทำหน้าที่คอยดูการเปลี่ยนแปลงของข้อความ ที่อ้างอิงด้วย Topic เช่น ถ้ามีหัวข้อที่หน้าสนใจและมีการเปลี่ยนแปลงก็จะทำการดึงข้อมูลนั้นๆ มาใช้งาน

2.9 Node-RED

Node-RED เป็นเครื่องมือสำหรับนักพัฒนาโปรแกรม ในการเชื่อมต่ออุปกรณ์ฮาร์ดแวร์เข้ากับ APIs (Application Programming Interface) ซึ่งเป็นการพัฒนาโปรแกรมแบบ Flow-Based Programming ที่มีหน้า UI สำหรับนักพัฒนาให้ใช้งานผ่าน Web Browser ดังรูปที่ 2.7 ทำให้การเชื่อมต่อเส้นทางการไหลของข้อมูลนั้นเป็นเรื่องง่าย เนื่องจาก Node-RED เป็น Flow-Based Programming นั้นทำให้แทบจะไม่ต้องเขียน Code ในการพัฒนาโปรแกรมเลย เพียงแค่เลือก Node มาวางแล้วเชื่อมต่อก็สามารถควบคุม I/O ได้ โดยจะมี Node ให้เลือกใช้งานอย่างหลากหลาย

สามารถสร้าง Function JavaScript ได้โดยใช้ Text Editor ที่มีอยู่ใน Node-RED และยังสามารถบันทึก Function, Templates, Flows เพื่อไปใช้งานกับงานอื่นต่อไป

Node-RED นั้นทำงานบน Node.js ทำให้เหมาะสำหรับการใช้งานกับ Raspberry Pi เนื่องจากใช้ทรัพยากรน้อย ขนาดไฟล์ไม่ใหญ่ และ Node.js ยังทำหน้าที่เป็นตัวกลางให้ Raspberry Pi สามารถติดต่อกับ Web Browser และอุปกรณ์อื่นๆ ได้



รูปที่ 2.7 โครงสร้างโปรแกรม Node-RED

2.10 รหัสมาตรฐานของสหรัฐอเมริกาเพื่อการแลกเปลี่ยนสารสนเทศ (ASCII)

แอสกี้ (ASCII) หรือรหัสมาตรฐานของสหรัฐอเมริกาเพื่อการแลกเปลี่ยนสารสนเทศ (American Standard Code for Information Interchange) เป็นรหัสอักขระที่ประกอบด้วยอักษรละติน เลขอารบิก เครื่องหมายวรรคตอนและสัญลักษณ์ต่างๆ โดยแต่ละรหัสจะแทนด้วยตัวอักขระหนึ่งตัว เช่น รหัส 65 (เลขฐานสิบ) ใช้แทนอักษรเอ (A) พิมพ์ใหญ่ เป็นต้น รหัสแอสกี้ถูกใช้เป็นรหัสกลางในการสื่อสารของระบบคอมพิวเตอร์ และเครื่องมือสื่อสารแบบดิจิทัลต่างๆ ภายใต้การดูแลของสถาบันมาตรฐานแห่งชาติสหรัฐอเมริกา (American National Standard Institute : ANSI) ปัจจุบันมีอักขระทั้งหมด 256 ตัว จากการจัดเก็บข้อมูลแบบ 8 บิต จึงเลือกใช้การสื่อสารด้วยการเข้ารหัสแบบ ASCII ระหว่าง Raspberry Pi และ NodeMCU

2.11 ฟังก์ชัน Interrupt

Interrupt คือ การขัดจังหวะการทำงานของโปรแกรมหลัก เมื่อเกิดเหตุการณ์บางอย่างขึ้น ทำให้ตัว CPU ไปทำงานโปรแกรมที่กำหนดไว้เมื่อเกิดการ Interrupt ทันที เมื่อเสร็จสิ้นโปรแกรมที่กำหนดไว้ CPU จะกลับมาทำงานที่โปรแกรมหลักตามปกติ โดย Interrupt มี 2 ประเภท คือ

1. External Interrupt เป็นการตอบสนองต่อเหตุการณ์ภายนอก เช่น การเปลี่ยนแปลงสัญญาณที่ขา Input

2. Internal Interrupt เป็นการตอบสนองต่อเหตุการณ์ภายใน เช่น การทำงานของ ฟังก์ชัน Timer/Counter

ในการใช้งานได้เลือกใช้ External Interrupt เพื่อรับค่าการเปลี่ยนแปลงของสวิตช์แบบปุ่มกด โดยใช้คำสั่ง AttachInterrupt() ในการสร้าง มีรูปแบบการใช้งานดังนี้

AttachInterrupt(Interrupt, ISR, Mode)

Interrupt คือ ตำแหน่งของขาที่ต้องการใช้งาน โดย NodeMCU สามารถใช้งานขา GPIO ได้ทุกๆ ขา เช่น

GPIO04 จะใส่เลข 4 หรือ D2 ก็ได้

ISR คือ ชื่อของฟังก์ชันที่สร้างไว้และต้องการเรียกใช้งานเมื่อเกิดการ Interrupt ที่กำหนดขึ้น

Mode คือ รูปแบบการเกิด Interrupt มี 5 รูปแบบ

LOW จะเกิดการ Interrupt เมื่อขาที่กำหนดไว้มีสถานะเป็น LOW

HIGH จะเกิดการ Interrupt เมื่อขาที่กำหนดไว้มีสถานะเป็น HIGH

CHANGE จะเกิดการ Interrupt เมื่อขาที่กำหนดไว้มีการเปลี่ยนแปลงสถานะ เช่น HIGH เป็น LOW หรือ LOW เป็น HIGH

RISING จะเกิดการ Interrupt เมื่อขาที่กำหนดไว้มีการเปลี่ยนสถานะจาก LOW เป็น HIGH

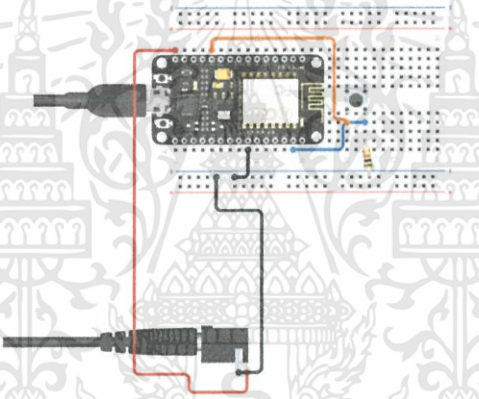
FALLING จะเกิดการ Interrupt เมื่อขาที่กำหนดไว้มีการเปลี่ยนสถานะจาก HIGH เป็น LOW

รูปที่ 2.8 การทำงานของฟังก์ชัน Interrupt

บทที่ 3
การออกแบบวงจรและการเขียนโปรแกรม

3.1 วงจรรับสัญญาณขาเข้าจากสวิตซ์ปุ่มกด

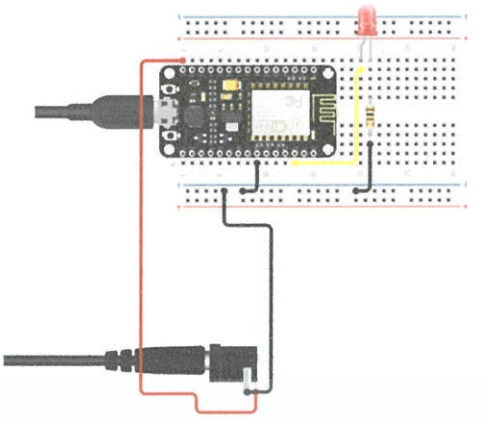
ในขั้นตอนแรกของการทำงานต้องการสัญญาณขาเข้า (Input) เพื่อสั่งการอุปกรณ์ต่างๆ เนื่องจาก NodeMCU ทำงานที่แรงดันไฟกระแสตรง 3.3 ถึง 3.6V จึงสร้างวงจรแรงดัน 3.3V ขึ้น โดยใช้ไฟเลี้ยงจากขา +3.3V ต่อกับสวิตซ์ปุ่มกดหน้าแบบ Normally Open (NO) และตัวต้านทานขนาด 10kΩ ไปยังขากราวด์ (GND) จากนั้นต่อขาที่ต้องการใช้รับสัญญาณขาเข้าที่ตำแหน่งระหว่างสวิตซ์กับตัวต้านทาน เพื่อรับสัญญาณลักษณะขาขึ้น (RISING) ดังรูปที่ 3.1



รูปที่ 3.1 การต่อวงจรรับสัญญาณขาเข้าจากสวิตซ์

3.2 วงจรควบคุมหลอด LED

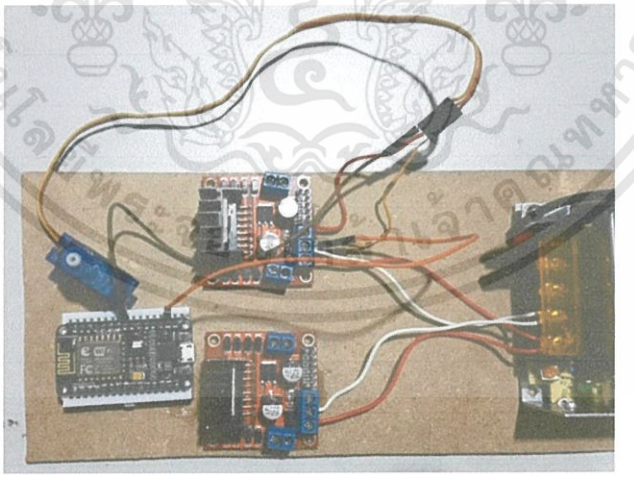
หลอดแอลอีดี (LED) มีแรงดันไฟกระแสตรงที่ใช้งานไม่เกิน 5V จึงสามารถต่อใช้งานกับ NodeMCU ได้โดยตรง ดังนี้ ต่อตัวต้านทานกับขาที่ใช้ควบคุมหลอดแอลอีดีเข้ากับตัวต้านทานขนาด 330Ω เพื่อป้องกันกระแสไฟฟ้าเกิน ถัดไปเป็นตัวหลอดแอลอีดีที่ต่อขาลบลงขา GND ของตัว NodeMCU ดังรูปที่ 3.2



รูปที่ 3.2 วงจรควบคุมหลอด LED

3.3 วงจรควบคุม Servo

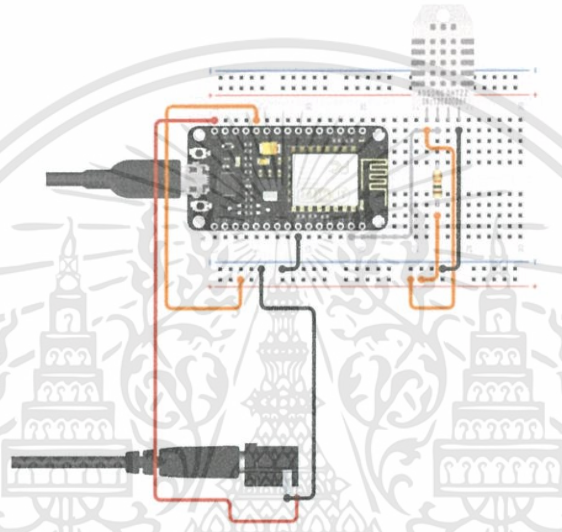
การใช้งานมอเตอร์ Servo นั้นใช้สัญญาณ PMW เพื่อควบคุมตำแหน่งการหมุน แต่เนื่องจากตัวมอเตอร์ Servo นั้นต้องการไฟเลี้ยงกระแสตรงที่ +5V เพื่อเริ่มทำงาน จึงต้องมีการใช้แหล่งจ่ายไฟจากภายนอกคือ บอร์ด L298N โดยต่อขาสัญญาณควบคุมเข้ากับสายไฟสีเหลือง สายไฟ +5V เข้ากับสายสีแดง และต่อสายสีดำเข้ากับกราวด์ (GND) เดียวกันทั้งหมด ดังรูปที่ 3.3



รูปที่ 3.3 วงจรควบคุมมอเตอร์ Servo

3.4 วงจรรับค่า DHT22

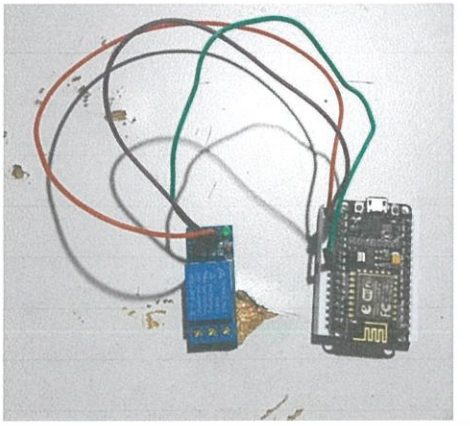
DHT22 ที่เป็นโมดูลจะมีการแปลงขาจาก 4 ขาให้เหลือเพียง 3 ขาอยู่แล้ว คือ VCC, OUTPUT, GND ทำงานได้ที่แรงดันไฟกระแสตรง 3.3 ถึง 5V ทำให้ต่อกับ NodeMCU ได้โดยตรง ดังนี้ ต่อสาย VCC เข้ากับขา 3.3V ต่อสาย OUTPUT เข้ากับขาที่ใช้รับสัญญาณข้อมูลและต่อสาย GND เข้าด้วยกัน ดังรูปที่ 3.4



รูปที่ 3.4 วงจรรับค่า DHT22

3.5 วงจรควบคุมรีเลย์

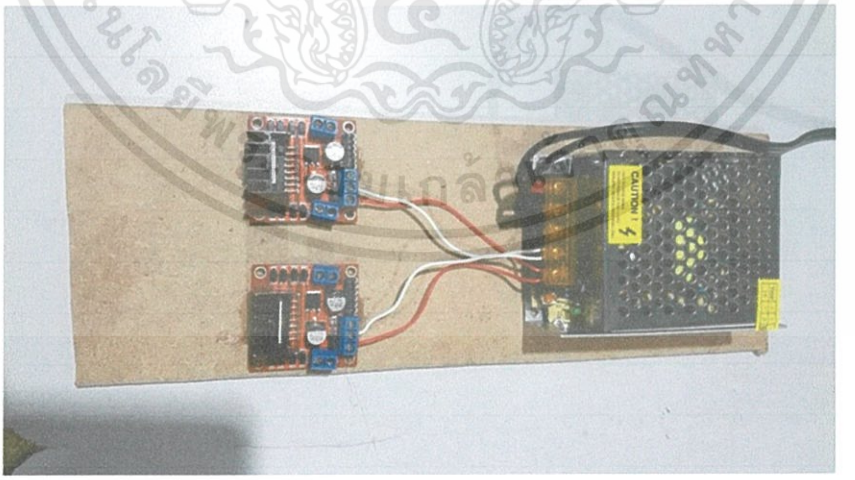
ในการควบคุมอุปกรณ์ไฟฟ้าที่ใช้งานแรงดันไฟกระแสสลับ 220V นั้นไม่สามารถทำได้โดยตรงด้วย NodeMCU ได้ ต้องใช้การควบคุมทางอ้อมด้วย Relay Module ซึ่งสั่งการด้วยสัญญาณไฟกระแสตรง 3.3V ได้ ใช้ไฟเลี้ยงกระแสตรง +5V เพื่อควบคุมหน้าสัมผัสที่รองรับไฟฟ้ากระแสสลับขนาด 220V ได้ การต่อวงจรควบคุมทำได้ ดังนี้ ต่อสายไฟเลี้ยง VCC เข้ากับขา Vin ของ NodeMCU ซึ่งมีแรงดันไฟกระแสตรง 5V ต่อสาย IN เข้ากับขา GPIO ที่ใช้ควบคุม และต่อ GND ร่วมกัน ในส่วนของวงจรแรงดันไฟกระแสสลับ 220V นั้นให้ตัดสาย Line นำปลายด้านแหล่งจ่ายต่อกับหน้าสัมผัส COM และปลายด้านอุปกรณ์เข้ากับหน้าสัมผัส NO หรือ NC แล้วแต่การใช้งาน ในที่นี้เลือกใช้หน้าสัมผัส NO เพราะต้องการให้สถานะเริ่มต้นเป็นการตัดวงจรไฟฟ้ากระแสสลับ ดังรูปที่ 3.5



รูปที่ 3.5 การต่อวงจรควบคุมรีเลย์

3.6 วงจรแปลงไฟเลี้ยงกระแสตรง 5V

ในการใช้งานทั่วไปแหล่งจ่ายไฟหลักจะมาจากไฟฟ้ากระแสสลับ 220V แต่ระบบนั้นทำงานที่แรงดันไฟกระแสตรง 3.3V และ 5V เลือกใช้ Switching Supply 220VAC to 12VDC ขนาดพิกัด 5A เป็นอุปกรณ์จ่ายไฟหลัก ซึ่งต่อเข้ากับบอร์ด L298N ซึ่งสามารถขับอุปกรณ์แรงดันไฟกระแสตรง 12V แบบ PMW ได้ในอนาคต และสามารถจ่ายแรงดันไฟกระแสตรง +5V ให้กับตัว NodeMCU และอุปกรณ์อื่นๆ เช่น มอเตอร์ Servo ได้ การต่อวงจรทำได้โดยเชื่อมต่อขา +12V และขา GND ของ Switching Supply และ L298N เข้าด้วยกัน ดังรูปที่ 3.6



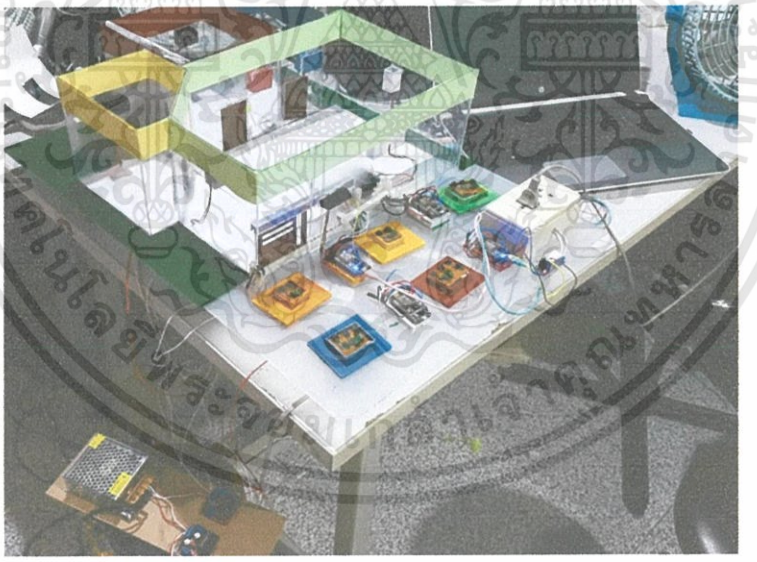
รูปที่ 3.6 การต่อวงจรไฟเลี้ยง 5V และ 12V

3.7 หลักการสร้างแบบจำลองบ้าน

ในการออกแบบตัวบ้านนั้นจะเน้นแสดงให้เห็นถึงการทำงานในส่วนต่างๆ ของบ้าน ที่ทำงานแยกกันอย่างอิสระตาม NodeMCU ที่ควบคุมในพื้นที่นั้นๆ โดยตัวบ้านสร้างจากแผ่น Acrylic ใส และมีการล้อมกรอบด้วยสีต่างๆ เพื่อแสดงพื้นที่ต่างๆ ที่แยกออกจากกันอย่างชัดเจน และมีหลอดแอลอีดีแทนหลอดไฟ มี Servo มอเตอร์แทนมอเตอร์ขนาดจริง มีปุ่มกดเพื่อควบคุมอุปกรณ์ต่างๆ ภายในตัวบ้าน ซึ่งติดตั้งอยู่ด้านหน้าของแบบจำลอง ดังรูปที่ 3.7

วัสดุที่ใช้ในการสร้างบ้านจำลอง

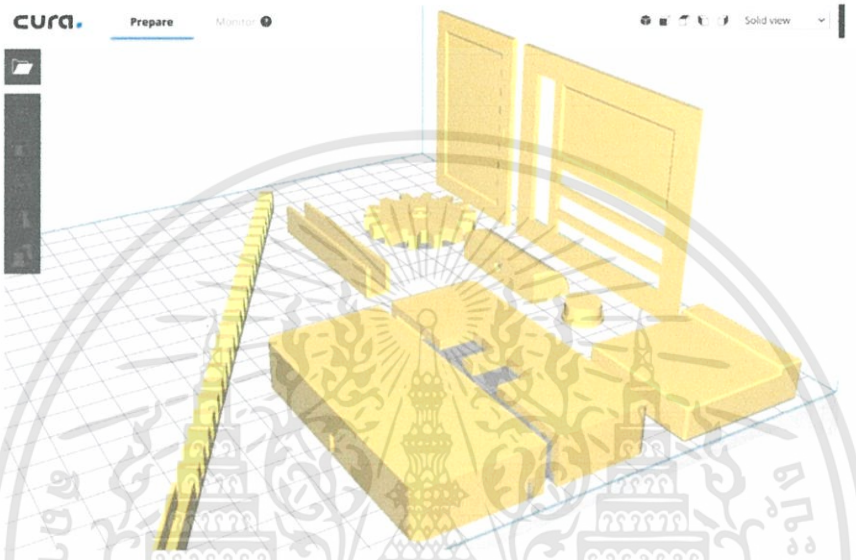
1. แผ่น Acrylic ใส
2. กระดาษสี
3. หมึกพลาสติก
4. กาวร้อน
5. ปืนกาว
6. ผ้าสักกะหลาด
7. สายไฟ



รูปที่ 3.7 แบบจำลองบ้านโดยรวม

3.8 การออกแบบอุปกรณ์ภายในบ้าน

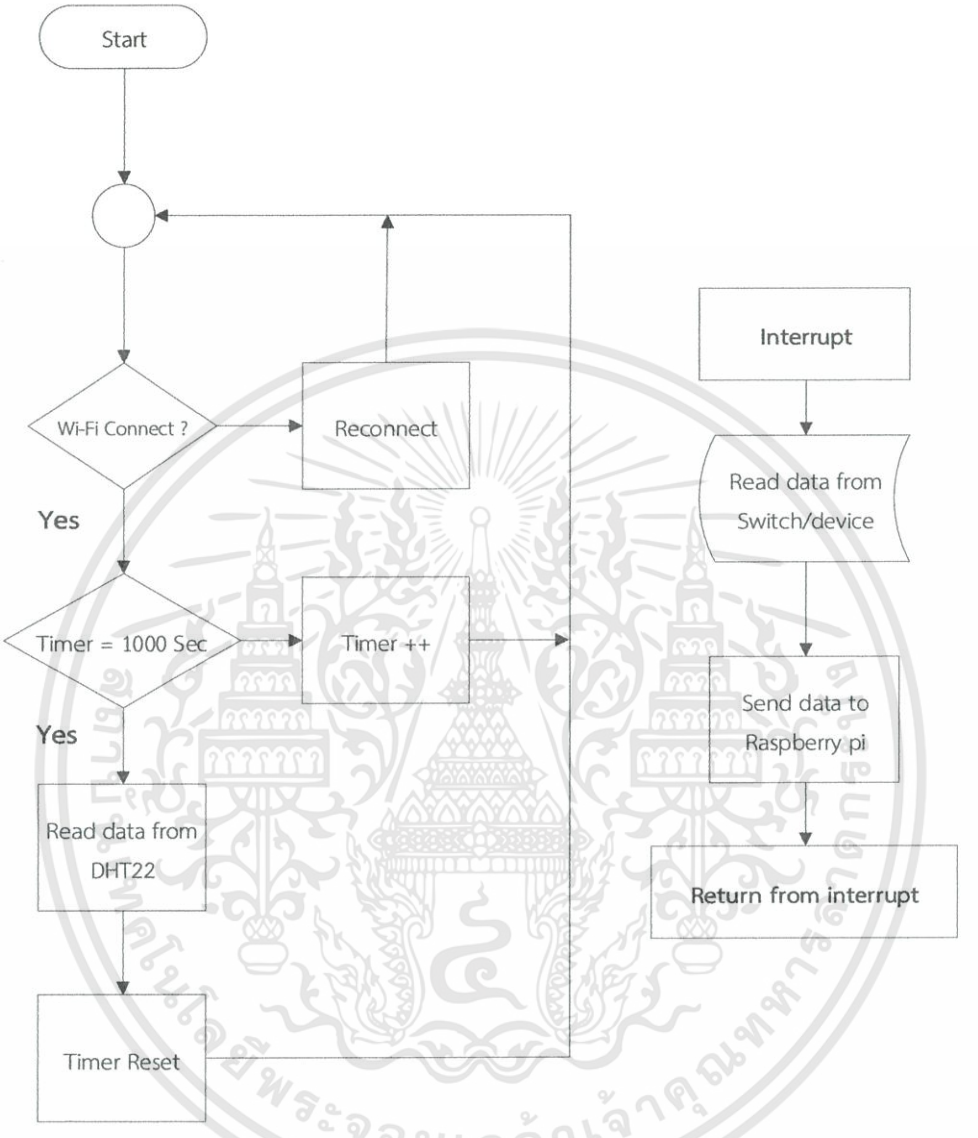
อุปกรณ์ภายในบ้านใช้การออกแบบด้วยโปรแกรม SOLIDWORKS และนำไปสร้างขึ้นด้วยเครื่องพิมพ์ 3 มิติ (3D Printer) ประกอบด้วย ประตูเลื่อน, เฟือง, รางเลื่อน, ประตูบานพับ, บานพับ, โคมไฟ, ฝาครอบมอเตอร์ Servo และกล่องสวิตช์ควบคุม ดังรูปที่ 3.8



รูปที่ 3.8 ชิ้นงานที่ถูกออกแบบด้วยโปรแกรม SOLIDWORKS

3.9 การเขียนโปรแกรม

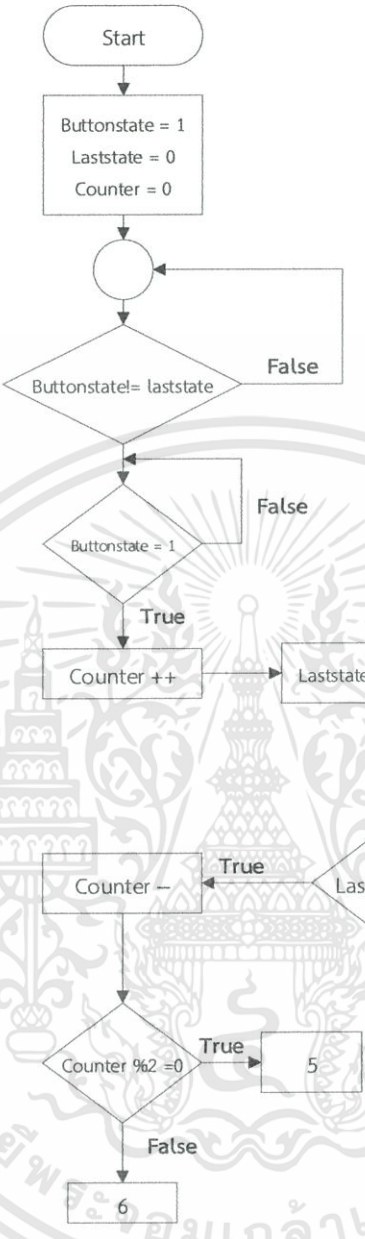
3.9.1 โปรแกรมการรับและส่งค่าจากเซนเซอร์และสวิตช์



รูปที่ 3.9 Flow Chart โปรแกรมการรับและส่งค่าจากเซนเซอร์และสวิตช์

จากรูปที่ 3.9 เป็นแผนภาพแสดงโปรแกรมการรับและส่งค่าจากเซนเซอร์และสวิตช์ โดยเมื่อเริ่มการทำงาน NodeMCU จะทำการเชื่อมต่อสัญญาณ Wi-Fi จากนั้นตัวนับเวลาจะเริ่มนับเวลา เมื่อเวลาผ่านไป 1000 วินาที Node-MCU จะอ่านค่าจากเซนเซอร์ DHT22 ซึ่งเป็นเซนเซอร์ที่วัดค่าอุณหภูมิและความชื้น เมื่ออ่านค่าแล้วจะเริ่มนับเวลาใหม่ แต่เมื่อมีการกดปุ่มหรืออ่านค่าจากเซนเซอร์ขึ้น NodeMCU จะส่งค่าที่อ่านไปยัง Raspberry Pi ซึ่งเป็น MQTT Broker

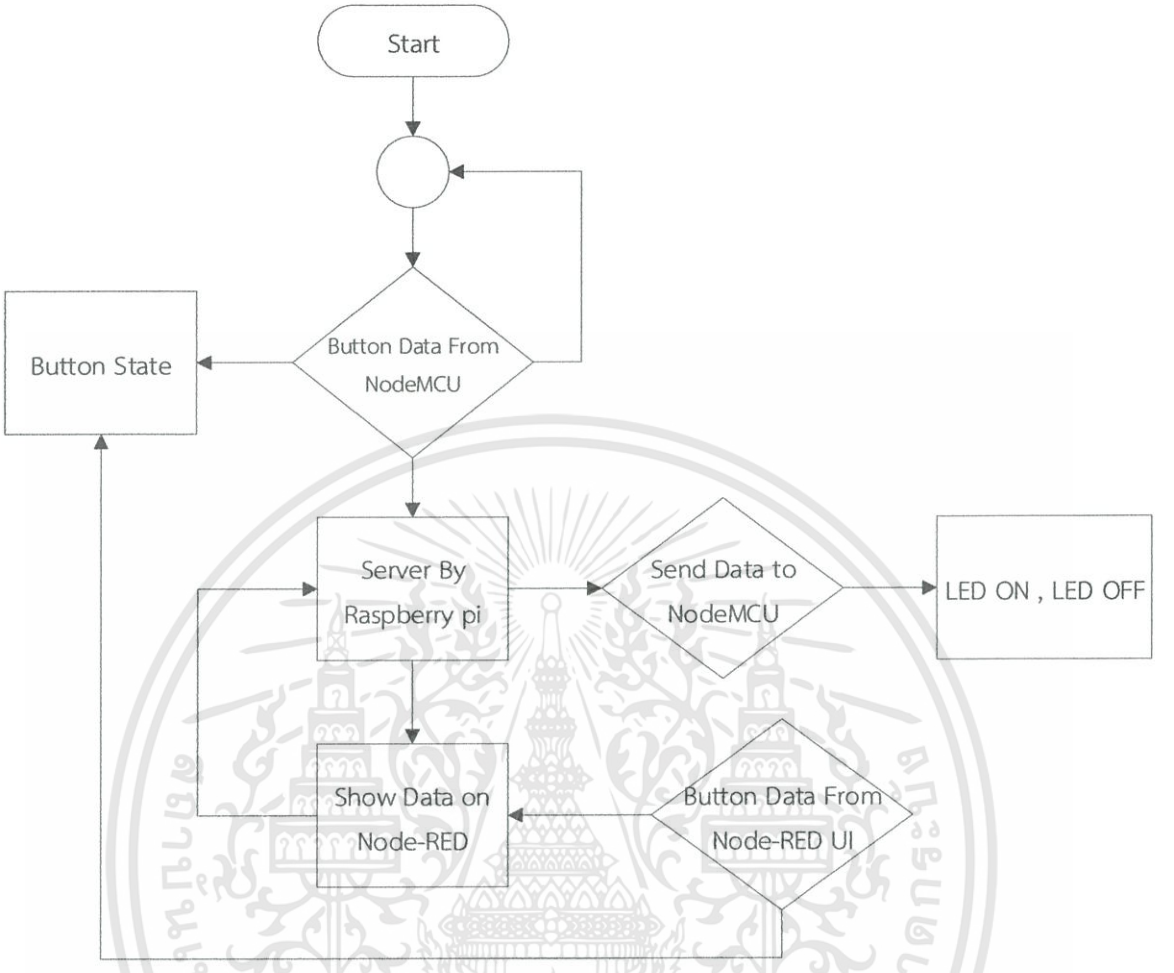
3.9.2 โปรแกรมตรวจสอบสถานะล่าสุดของสวิตซ์เพื่อใช้ในการส่งค่าข้อมูล



รูปที่ 3.10 Flow Chart โปรแกรมตรวจสอบสถานะล่าสุดของสวิตซ์เพื่อใช้ในการส่งค่าข้อมูล

จากรูปที่ 3.10 เป็นแผนภาพการตรวจสอบสถานะล่าสุดของสวิตซ์เพื่อใช้ในการส่งค่าข้อมูล โดยเมื่อเริ่มการทำงาน โปรแกรมจะมีการตั้งค่าพื้นฐานโดย Bottonstate = 1, Laststate = 0 และ Counter = 0 โปรแกรมจะทำการตรวจสอบค่าของ Bottonstate ว่าถูกกดไปแล้วหรือไม่ หากถูกกดไปแล้วจะทำให้มีการเปลี่ยนค่าใน Laststate ให้เหมือนกับ Bottonstate เพื่อให้โปรแกรม Update สถานะของสวิตซ์ในปัจจุบัน

3.9.3 โปรแกรมเปิด-ปิด LED โดยใช้สวิตช์แบบทางกล และสวิตช์แบบซอฟแวร์



รูปที่ 3.11 Flow Chart โปรแกรมเปิด-ปิด LED โดยใช้สวิตช์แบบทางกล และสวิตช์แบบซอฟแวร์

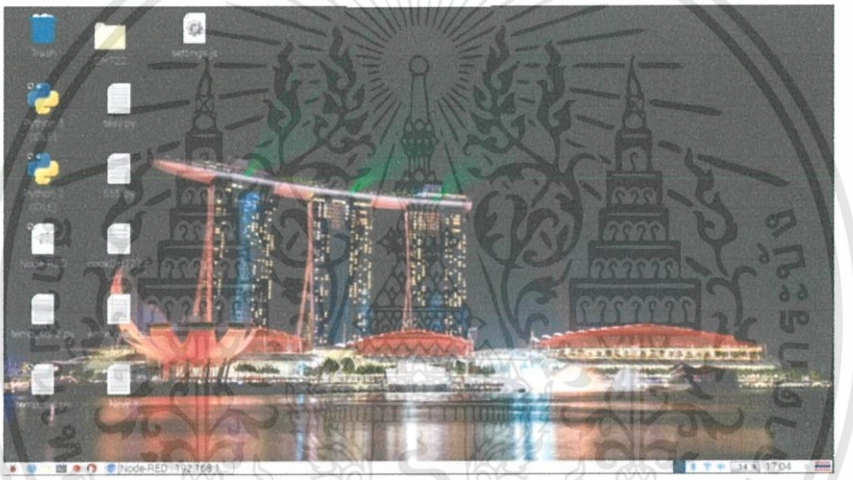
จากรูปที่ 3.11 เป็นแผนภาพการเปิด-ปิด LED โดยใช้สวิตช์แบบทางกล และสวิตช์แบบซอฟแวร์ เริ่มต้นโดยการตรวจสอบสถานะของปุ่มกดใน NodeMCU และใน UI ของ Node-RED เพื่อ Update สถานะของสวิตช์ในปัจจุบันให้เหมือนกัน จากนั้นส่งข้อมูลสถานะล่าสุดของสวิตช์เข้าไปแสดงผลที่หน้า UI ของ Node-RED พร้อมทั้งส่งไปที่ NodeMCU เพื่อให้สั่งการเปิด-ปิด LED โดยอิงข้อมูลจากค่าล่าสุดของสวิตช์ที่บันทึกไว้ใน UI ของ Node-RED และ NodeMCU

บทที่ 4

การทดลองและผลการทดลอง

4.1 การทดสอบติดตั้งระบบปฏิบัติการใน Raspberry Pi 3

การติดตั้งระบบปฏิบัติการ Raspbian ใน Raspberry Pi 3 เพื่อใช้ในการติดตั้งโปรแกรมหลักและโปรแกรมย่อยที่จะใช้ในโครงงาน โดยในโครงงานนี้จะใช้ความสามารถในระบบปฏิบัติการ Raspbian มาเป็นตัวช่วยในการอำนวนความสะดวกเรื่องของการใช้งานด้านต่างๆ เช่น การใช้ Graphical Programming ใน Node-RED, การสร้างระบบฐานข้อมูลใน phpMyAdmin, และทดสอบระบบต่างๆ ผ่าน Web Browser ที่ระบบปฏิบัติการมีให้มา ดังในรูปที่ 4.1



รูปที่ 4.1 หน้าต่าง UI (User Interface) ของ Raspbian

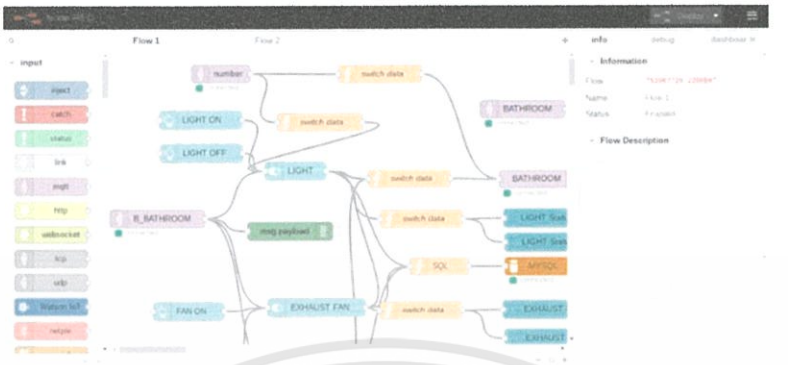
สรุปผลการทดลองที่ 4.1

เมื่อทำการติดตั้งระบบปฏิบัติการ Raspbian ใน Raspberry Pi 3 แล้ว ปรากฏว่าสามารถใช้ฟังก์ชันต่างๆ ได้อย่างสมบูรณ์

4.2 การทดสอบเขียนโปรแกรม Graphical โดยผ่านโปรแกรม Node-RED

การออกแบบโปรแกรม Graphical โดยผ่านโปรแกรม Node-RED เพื่อให้สามารถกำหนดรูปแบบการทำงานให้ได้ตามที่ต้องการ จากรูปที่ 4.2 จะเห็นได้ว่าในโปรแกรม Node-RED จะประกอบไปด้วย Node ต่างๆ และคำสั่งใน Node นั้นๆ ซึ่งจะแบ่งเป็น Node ประเภทต่างๆ เช่น Input, Output และ Function โดยเมื่อทำการเชื่อมโยง Node ต่างๆ เข้าด้วยกันจะกลายเป็น Flow

หรือรูปแบบการทำงานของระบบ ซึ่งจากการทดลองพบว่า Flow ที่เขียนขึ้นมานั้นสามารถทำการรับค่า เก็บค่า และแสดงสถานะของสวิตช์ในส่วนต่างๆ ได้อย่างรวดเร็วและแม่นยำ



รูปที่ 4.2 หน้าต่างของโปรแกรมและ Flow ของ Node-RED

สรุปผลการทดลองที่ 4.2

เมื่อเขียนโปรแกรมลงใน Node-RED แล้วทดลองสั่งการหลอด LED ผ่านหน้า UI บนคอมพิวเตอร์ ปรากฏว่าหลอด LED สามารถเปิด-ปิด ตามที่สั่งการไว้ได้

4.3 การทดลองควบคุมการเปิดปิด LED ผ่าน Wi-Fi โดยใช้ Node-RED

จากรูปที่ 4.3 เป็นการใช้งาน User Interface (UI) อย่างง่ายผ่านโปรแกรม Node-RED โดยการต่อวงจร LED พื้นฐาน แล้วทำการเขียนโปรแกรมลงใน Node-RED เพื่อให้ติดต่อสื่อสารกับบอร์ด Raspberry Pi ผ่าน Web Browser ได้

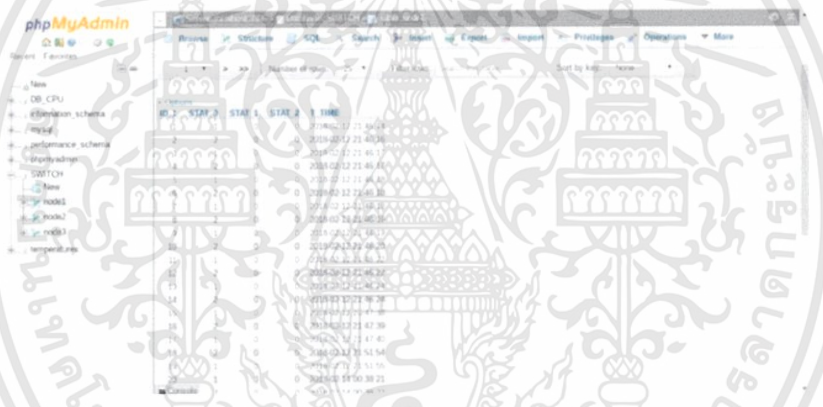
รูปที่ 4.3 หน้าต่าง UI (User Interface) ของ Node-RED

สรุปผลการทดลองที่ 4.3

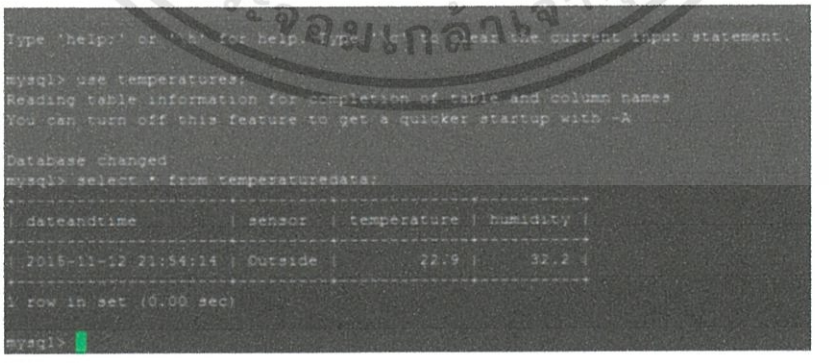
เมื่อเขียนโปรแกรมลงใน Node-RED แล้วทดลองเปิด-ปิด หลอด LED ผ่านหน้า Web Browser ปรากฏว่า หลอด LED เปิด-ปิด ตามที่สั่งการไว้

4.4 การทดลองการนำค่าที่ได้มาเก็บไว้ในระบบฐานข้อมูล

การออกแบบระบบฐานข้อมูล (Database) โดยผ่านโปรแกรม phpMyAdmin ซึ่งสามารถจัดสรรรูปแบบข้อมูลที่ต้องการเก็บว่าจะเก็บข้อมูลที่สนใจไว้ในส่วนใดของ Database เพื่อให้ง่ายต่อการตรวจสอบและเรียกดูข้อมูล จากรูปที่ 4.4 จะเห็นได้ว่าในโปรแกรม phpMyAdmin จะใช้ภาษา SQL ในการเขียน ซึ่งประกอบไปด้วยส่วนของ Workspace และ Tree ซึ่งสามารถแบ่งเป็น Branch ต่างๆ เพื่อสร้างหัวข้อที่จะเก็บข้อมูลที่ต้องการ และจากการทดลองพบว่าโปรแกรมที่เขียนขึ้นมานั้นสามารถทำการรับค่าจาก Node-RED และเก็บค่าลงไปในตารางที่สร้างไว้ในรูปแบบตัวเลข พร้อมทั้งระบุวันที่และเวลาได้อย่างถูกต้องแม่นยำ ดังรูปที่ 4.5



รูปที่ 4.4 หน้าต่าง Table ที่เก็บค่าจาก Node-RED



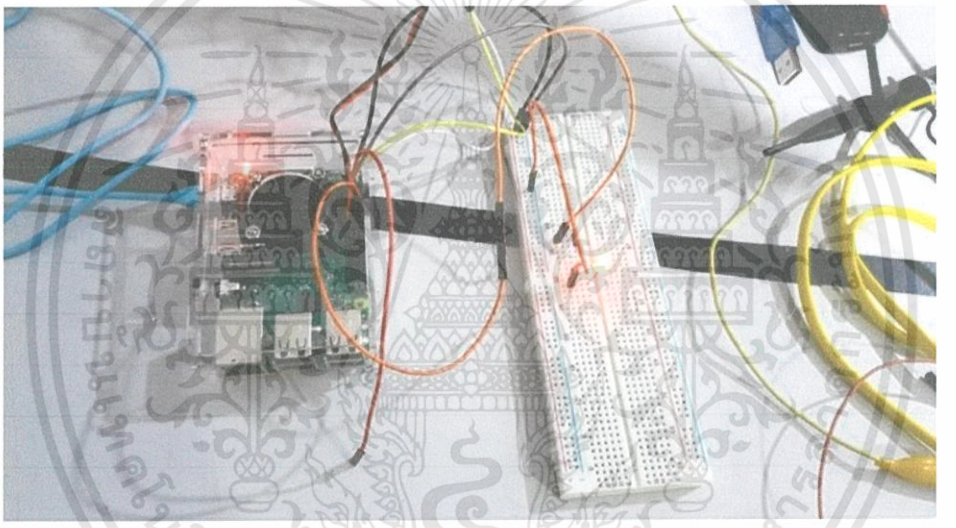
รูปที่ 4.5 นำข้อมูลจากเซนเซอร์มาเก็บไว้ในระบบฐานข้อมูล

สรุปผลการทดลองที่ 4.4

ในการทดลองขั้นตอนนี้ จะเป็นการทดลองนำค่าที่ได้จากเซนเซอร์ DHT22 ซึ่งเป็นเซนเซอร์วัดความชื้น และอุณหภูมิ มาลงในฐานข้อมูลผ่าน phpMyAdmin การทดลองนี้เป็นการทดสอบการรับส่ง และการเรียกใช้ข้อมูลโดยผ่านระบบฐานข้อมูล พบว่าสามารถนำค่าจากเซนเซอร์มาเก็บไว้ในระบบฐานข้อมูล และสามารถเรียกดูข้อมูลที่บันทึกในระบบฐานข้อมูลได้

4.5 การทดลองส่งสัญญาณขาออก

ทดลองเขียนโปรแกรมด้วยภาษา Python เพื่อสั่งเปิดการใช้งาน GPIO และส่งสัญญาณขาออกที่ขา (Pin) ต่างๆ โดยใช้ LED เป็นตัวบ่งชี้ หากหลอด LED ติดแสดงว่า Raspberry Pi มีการเปลี่ยนสถานะของ GPIO เป็น 1 (3.3V)



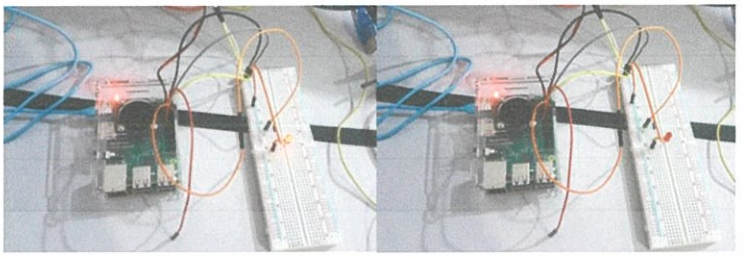
รูปที่ 4.6 ผลการทดลองส่งสัญญาณขาออก

สรุปผลการทดลองที่ 4.5

จากรูปที่ 4.6 หลอด LED มีการติดเมื่อใช้งานโปรแกรมที่สั่งให้ GPIO จ่ายสัญญาณออกไป

4.6 การทดลองหน่วงเวลาติดดับของ LED

ทดลองเขียนโปรแกรมด้วยภาษา Python เพื่อสั่งให้หลอด LED มีการติดดับตามการหน่วงเวลาที่กำหนดไว้ที่ 1 วินาที



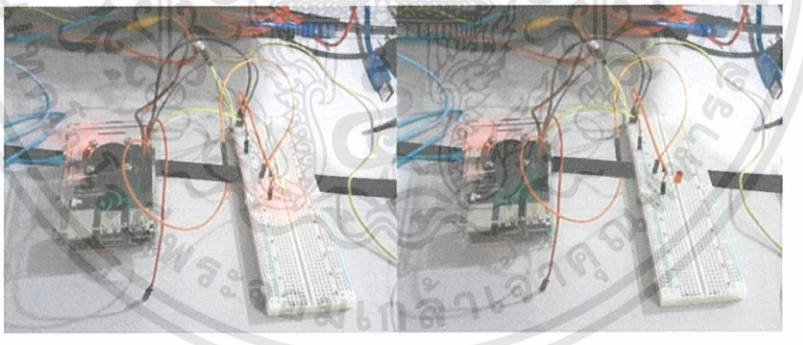
รูปที่ 4.7 ผลการทดลองหน่วงเวลาติดของ LED และหน่วงเวลาดับของ LED

สรุปผลการทดลองที่ 4.6

จากรูปที่ 4.7 หลอด LED ติดและดับสลับกันเป็นเวลาช่วงละ 1 วินาที

4.7 การทดลองหน่วงสลับช่วงเวลาติดดับของ LED

ทดลองเขียนโปรแกรมด้วยภาษา Python เพื่อสั่งให้หลอด LED มีการติดดับตามการหน่วงเวลาที่กำหนดไว้ที่ 5 วินาทีและติดดับเป็นช่วงเวลา 1 วินาที และวนการทำงานกลับไปเป็นลูป



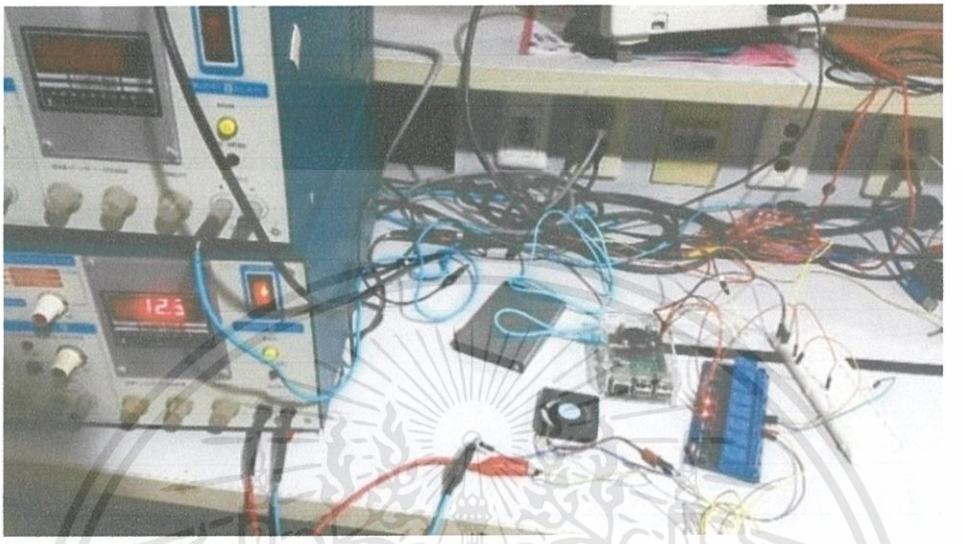
รูปที่ 4.8 ทดลองหน่วงสลับเวลาขณะติดของ LED และหน่วงสลับเวลาขณะดับของ LED

สรุปผลการทดลองที่ 4.7

จากรูปที่ 4.8 หลอด LED สามารถติดดับสลับช่วงเวลาได้ตามการทดลอง

4.8 การทดลองควบคุมโหลดที่แรงดันสูงกว่าผ่านชุด 8 Channel Relay Module

ทดลองเขียนโปรแกรมด้วยภาษา Python เพื่อสั่งให้รีเลย์เปิดหน้าสัมผัสให้พัดลมแรงดันไฟกระแสตรง 12V ทำงาน



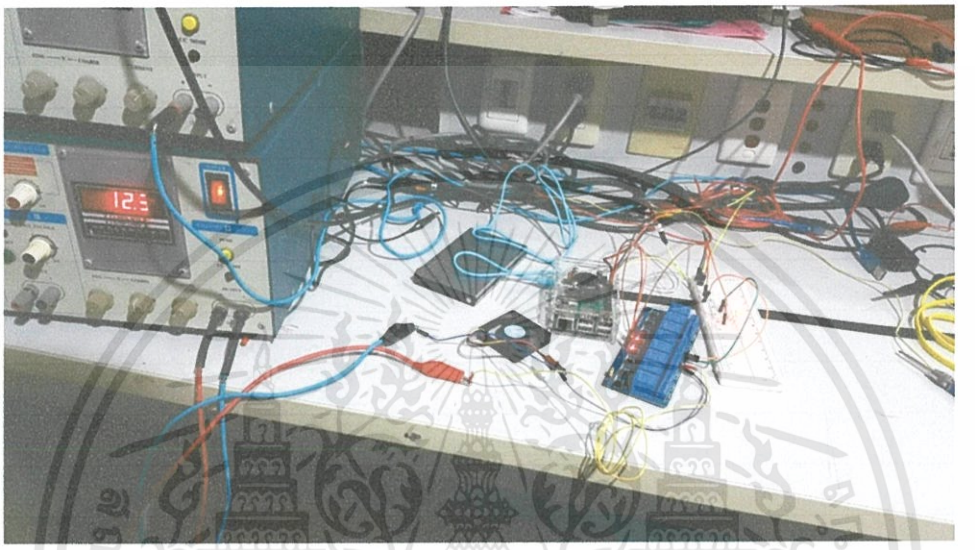
รูปที่ 4.9 ผลการทดลองควบคุมโหลดที่แรงดันสูงกว่าผ่านชุด 8 Channel Relay Module

สรุปผลการทดลองที่ 4.8

จากรูปที่ 4.9 รีเลย์ทำงานได้ตามโปรแกรมคำสั่งที่อยู่ใน Raspberry Pi และพัดลมที่ใช้แรงดันไฟกระแสตรง 12V ทำงาน

4.9 การทดลองควบคุมโหลดที่หลายแรงดันผ่านชุด 8 Channel Relay Module

ทดลองเขียนโปรแกรมด้วยภาษา Python เพื่อสั่งให้รีเลย์ทำการเปิดหน้าสัมผัสให้พัดลมแรงดันไฟกระแสตรง 12V ทำงานเป็นเวลา 5 วินาที พร้อมกับหลอด LED ดับและพัดลมหยุดทำงาน 1 วินาที พร้อมกับหลอด LED ติด สลับกันไปเรื่อยๆ



รูปที่ 4.10 ผลการทดลองควบคุมโหลดที่หลายแรงดันผ่านชุด 8 Channel Relay Module

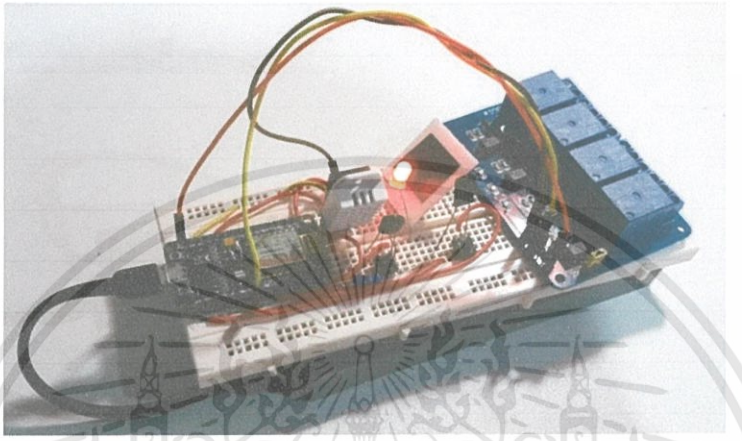
สรุปผลการทดลองที่ 4.9

จากรูปที่ 4.10 รีเลย์ทำงานได้ตามโปรแกรมคำสั่ง พัดลมแรงดันไฟกระแสตรง 12V ทำงานเป็นเวลา 5 วินาที พร้อมหลอด LED ที่ติดสลับกับหยุดทำงาน 1 วินาทีพร้อม LED ที่ดับได้ตามคำสั่ง

4.10 การทดสอบควบคุมโหลดแบบไร้สายโดยผ่าน NodeMCU

4.10.1 การควบคุมโหลดแบบไร้สาย

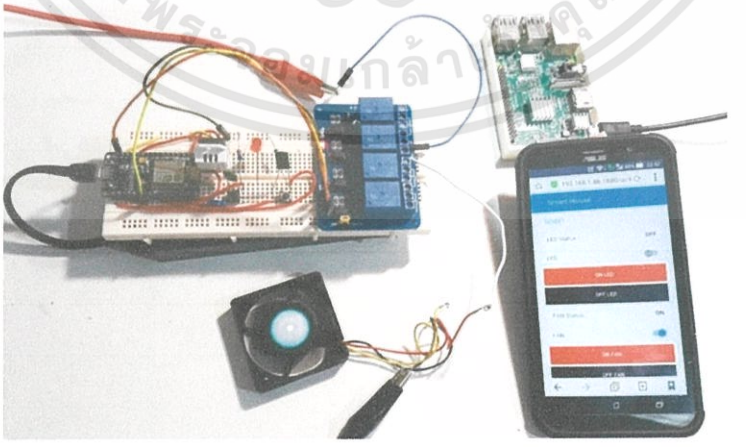
จากรูปที่ 4.11 พบว่าโปรแกรมสำหรับควบคุมมอเตอร์ Servo, LED ที่เขียนขึ้นในโปรแกรม Arduino สามารถควบคุมโดยใช้ Smart Phone และคอมพิวเตอร์ ผ่าน Raspberry Pi 3 ให้เปิดปิดและแสดงสถานะตามจริง โดยผ่านระบบ Wi-Fi ซึ่งการทดลองนี้จะสามารถบ่งบอกประสิทธิภาพของการรับส่งค่าคำสั่งต่างๆ ผ่านระบบไร้สาย ซึ่งค่าคำสั่งที่ส่งไปจะส่งไปในรูปแบบ Digital



รูปที่ 4.11 การสั่งเปิดปิด LED ผ่านระบบ Wi-Fi

4.10.2 การควบคุมโหลดแรงดันไฟต่างระดับ แบบไร้สายโดยผ่านรีเลย์

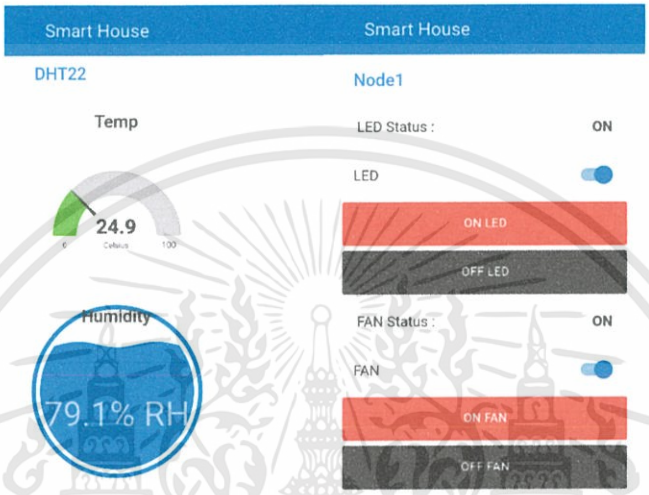
จากรูปที่ 4.12 พบว่าโปรแกรมสำหรับควบคุมรีเลย์ ที่เขียนขึ้นในโปรแกรม Arduino สามารถควบคุมโดยใช้ Smart Phone และคอมพิวเตอร์ผ่าน Raspberry Pi 3 ให้เปิดปิดและแสดงสถานะตามจริงได้ ซึ่งค่าคำสั่งที่ส่งไปจะส่งไปในรูปแบบ Digital โดยผ่านระบบ Wi-Fi และการทดลองนี้จะสามารถบ่งบอกความสามารถในการควบคุมโหลดที่ใช้งานจริงในบ้านได้ผ่านระบบไร้สาย



รูปที่ 4.12 การสั่งเปิดปิดพัดลมแรงดันไฟกระแสตรง 12V ด้วยรีเลย์ ผ่านระบบ Wi-Fi

4.11 การสร้าง UI (User Interface) เพื่อใช้ตรวจสอบสถานะและควบคุมระบบ

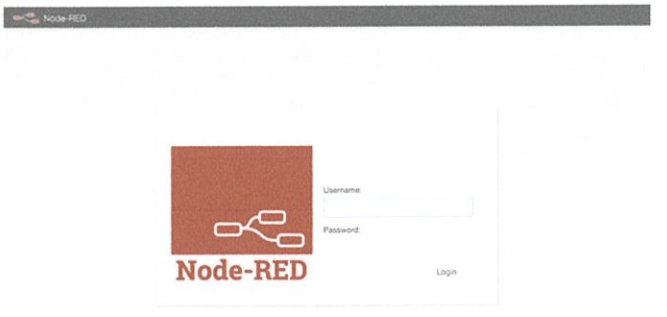
การสร้าง UI (User Interface) หรือส่วนต่อประสานกับผู้ใช้ทำหน้าที่เชื่อมประสานระหว่างผู้ใช้และระบบ ในโครงงานนี้สร้างขึ้นมาเพื่อให้สามารถระบุตำแหน่งสวิตช์ที่ต้องการควบคุมและรู้สถานะของสวิตช์ได้ โดยสร้างสวิตช์ให้ตรงกับตำแหน่งที่ต้องการ พร้อมทั้งบอกถึงสถานะของตัวสวิตช์ด้วย ดังรูปที่ 4.13 จากการทดลองพบว่า UI ที่เขียนขึ้นมานั้นสามารถทำการควบคุมและแสดงผลได้อย่างถูกต้อง



รูปที่ 4.13 UI (User Interface) บน Smart Phone

4.12 การสร้างมาตรการความปลอดภัยให้กับระบบ

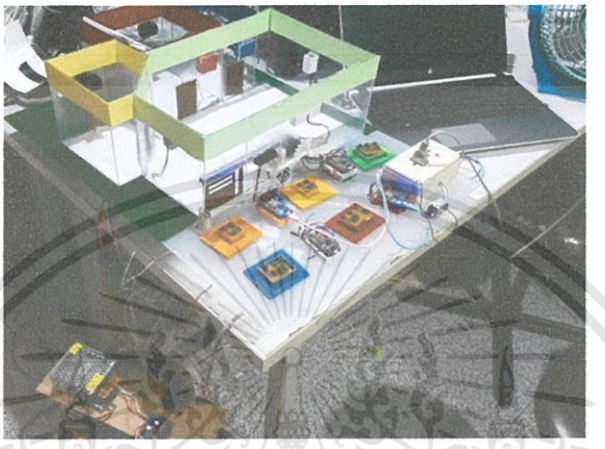
การสร้างมาตรการความปลอดภัยให้กับระบบหรือ Security Policy ในโครงงานนี้สร้างขึ้นมาเพื่อให้สามารถป้องกันการเข้ามาแก้ไขระบบโดยไม่ได้รับอนุญาต โดยสร้างให้มีหน้าต่าง Log-in พร้อม User และ Password ซึ่ง Password จะใช้การเข้ารหัสแบบ MD5 ซึ่งจะสามารถทำให้การเจาะเข้าระบบนั้นเป็นไปได้ยากขึ้น อีกทั้งจำกัดผู้ใช้ที่ไม่เกี่ยวข้องเพื่อไม่ให้ไปแก้ไขระบบจนเสียหายได้ ดังรูปที่ 4.14



รูปที่ 4.14 หน้า Log-in สำหรับเข้าสู่ระบบเพื่อทำการแก้ไข

4.13 การทดสอบระบบทั้งหมด

ในการทดลองได้ทดสอบนำทุกส่วนของทั้งระบบมารวมกัน และทำการควบคุมโดยผ่าน Smart Phone และคอมพิวเตอร์ พร้อมทั้งทดลองควบคุมผ่านสวิตช์จริง ดังรูปที่ 4.15 ซึ่งทั้งระบบสามารถควบคุมและแสดงสถานะหลอดไฟ รวมถึงอุปกรณ์แรงดันไฟกระแสสลับ 220V ได้อย่างถูกต้อง มีการตอบสนองได้รวดเร็ว แต่ระบบนี้ถูกจำกัดเพียงการใช้ในระบบวง LAN ส่วนบุคคลเท่านั้น



รูปที่ 4.15 ภาพรวมของระบบ

บทที่ 5
สรุปและวิจารณ์ผลการทดลอง

5.1 สรุปผลการทดลอง

จากการทดสอบการเข้าใช้งานระบบ พบว่าระบบสามารถควบคุมและแสดงสถานะอุปกรณ์พื้นฐานและอุปกรณ์ไฟฟ้ากระแสสลับแรงดัน 220V ในบ้าน ผ่านชุด Relay ได้อย่างถูกต้อง การตอบสนองทำได้อย่างรวดเร็ว และสามารถติดตามการใช้งานต่างๆ ผ่านระบบฐานข้อมูล ส่วน UI (User Interface) ที่สร้างไว้ทำให้ผู้ใช้สามารถเข้าใจ และควบคุมระบบได้ง่ายโดยไม่จำเป็นต้องรู้ลึกในการทำงานของระบบ และในเรื่องของความปลอดภัยสามารถป้องกันการเข้าถึงระบบแบบไม่พึงประสงค์ได้ ระบบการทำงานของส่วนประกอบย่อยต่างๆ นั้นสามารถทำงานได้จริง และสามารถนำมาทำงานร่วมกันได้ โดยมีตัวควบคุมหลัก Raspberry Pi และอินเตอร์เน็ตเป็นหัวใจของระบบนี้

5.2 ปัญหาที่พบและแนวทางการแก้ไขปัญหา

5.2.1 ปัญหาที่พบ

จากการทำโครงงานพบว่า ปัญหาหลักๆ ที่พบเจอคือ ปัญหาที่เกี่ยวข้องกับ Hardware เช่น การบัดกรีเชื่อมจุดทำได้ไม่ดี จึงทำให้การส่งสัญญาณผิดเพี้ยน อีกทั้งบางครั้งจุดบางจุดเชื่อมติดกันแบบไม่ได้ตั้งใจทำให้ Microcontroller รวน และสวิตช์ที่นำมาใช้ในครั้งแรกนั้นพบว่าหน้าสัมผัสไม่ดี ทำให้ทุกครั้งที่กดลงไปเกิดสัญญาณที่ไม่พึงประสงค์ อีกปัญหาที่พบคือ สัญญาณการรับส่งข้อมูลผ่านระบบไร้สายนั้น ถ้าอยู่ในสภาพแวดล้อมที่มีสัญญาณรบกวนมากหรือการควบคุม NodeMCU หลายๆ ตัวพร้อมกันจะทำให้ระบบตอบสนองช้าและไม่แม่นยำเท่าที่ควร ซึ่งสาเหตุอาจเกิดการที่สัญญาณที่ใช้งานนั้นมีความถี่เหมือนหรือใกล้เคียงกับสัญญาณรบกวน ทำให้ต้องมีการส่งสัญญาณที่ถี่ขึ้นเพื่อให้รับสัญญาณที่ถูกต้องได้ ปัญหาที่พบเพิ่มเติมคือ ระบบ LAN ไร้สายจะต้องทำการตั้งค่าใหม่ทุกครั้งหากมีการเปลี่ยน Router ซึ่งทำให้ไม่สะดวกหาก Router เดิมเกิดความเสียหาย ในส่วนของปัญหาที่เหลือจะเป็นปัญหาในระหว่างการประกอบโมเดล อาทิเช่น กาวยึดไม่สนิท, การเกิดไอกาวบนแผ่น Acrylic, ขนาดชิ้นส่วนที่สร้างขึ้นจาก 3D Printer ไม่ตรงกับความต้องการ

5.2.2 แนวทางการแก้ไขปัญหา

การแก้ปัญหาในโครงงานนี้ต้องกล่าวก่อนว่าในกรณีข้างต้น ปัญหาได้แบ่งออกเป็นสามส่วน ซึ่งการแก้ปัญหานั้นมีดังนี้

1. ในส่วนของปัญหาเกี่ยวข้องกับ Hardware มีการแก้ไขโดยการใช้ Breadboard แทนการบัดกรี ซึ่งทำให้ง่ายต่อเปลี่ยนวงจรแต่หากต้องเคลื่อนย้ายตัวโมเดลงานอาจจะทำให้สายไฟหลุดได้ง่าย

2. ส่วนของการส่งสัญญาณไร้สายพบว่าบางครั้งที่ไม่สามารถรับส่งสัญญาณได้ดีนั้น เนื่องจากมีสัญญาณรบกวนอื่นๆ หรือการควบคุม NodeMCU หลายๆ ตัวพร้อมกัน ซึ่งจะทำให้ระบบตอบสนองช้าและไม่แม่นยำเท่าที่ควร ทำการแก้ไขโดยใช้ฟังก์ชันแยกช่องสัญญาณของ MQTT (Message Queuing Telemetry Transport) Protocol ซึ่งจะช่วยให้มีการแยกช่องสัญญาณที่จะใช้สื่อสารกับ NodeMCU แต่ละตัว ทำให้การส่งสัญญาณนั้นเป็นไปอย่างรวดเร็วและแม่นยำ ถึงแม้จะมีการควบคุม NodeMCU หลายตัวพร้อมๆ กัน สำหรับสัญญาณรบกวนนั้น สามารถทำการแก้ไขได้โดยการเปลี่ยนสถานที่ทดลองหรือใช้การส่งสัญญาณด้วยช่วงความถี่อื่นๆ

3. ทางด้านตัวโมเดลพบว่าการใช้กาวตราช้างหรือกาวร้อนทำให้เกิดไอกาวติดอยู่บนแผ่น Acrylic ซึ่งทำให้ตัวโมเดลไม่สวยงาม จึงทำการแก้ไขโดยใช้น้ำยาทำความสะอาด ทำความสะอาดตรงที่เกิดไอกาว และเปลี่ยนไปใช้กาวจากปืนกาวแทน เพื่อให้ไม่เกิดไอกาวติดที่แผ่น Acrylic

5.3 ข้อเสนอแนะและแนวทางการพัฒนา

โครงงานนี้ตั้งใจไว้ว่าจะเป็นโมเดลของระบบควบคุม และตรวจสอบสถานะในบ้านผ่านระบบไร้สาย ที่สามารถควบคุมผ่าน Smartphone และคอมพิวเตอร์ โดยที่ผู้ใช้ไม่จำเป็นต้องมีความรู้ในการสร้างระบบ ซึ่งระบบนี้ยังสามารถพัฒนาระบบให้มีความเสถียรและแม่นยำมากขึ้น โดยการใช้ Microcontroller ที่มีความแม่นยำสูงขึ้น โดยยังสามารถพัฒนาให้ระบบออนไลน์ได้ ซึ่งจะทำให้ผู้ใช้สามารถควบคุมระบบได้จากทุกๆ พื้นที่บนโลกผ่านอินเตอร์เน็ต อีกทั้งข้อมูลที่เก็บไว้ในระบบฐานข้อมูล (Database) สามารถนำไปใช้ในการพัฒนา Machine Learning ซึ่งจะสามารถเรียนรู้นิสัยการใช้งานและคาดเดาพฤติกรรมเพื่ออำนวยความสะดวกแก่ผู้ใช้งานเป็นรายบุคคลได้ การทดลองนี้เป็นแนวคิดในการต่อยอดการสร้างระบบควบคุมในรูปแบบต่างๆ เช่น ระบบ Scada ในอุตสาหกรรม และการทำ Smart City ที่รัฐบาลกำลังให้ความสนใจ ซึ่งในอนาคตระบบควบคุมอัตโนมัติระยะไกลจะมีความสำคัญอย่างยิ่งในวงการอุตสาหกรรมและครัวเรือนอย่างแน่นอน

เอกสารอ้างอิง

[1] **"Raspberry Pi 3 Model B Hardware description"** [Online]. Available:

https://www.raspberrypi.org/products/raspberry-pi-3-model-b

[2] **"MySQL client download and introduction"** [Online]. Available:

www.oracle.com/technetwork/database/mysql/index.html

[3] **"Node-RED : A flow-based programming for Iot"** [Online]. Available:

https://nodered.org

[4] **"NodeMcu hardware description"** [Online]. Available:

http://www.nodemcu.com/index_en.html

[5] **"Arduino examples from libraries"** [Online]. Available:

https://www.arduino.cc/en/Tutorial/Homepage

[6] **"MQTT Server"** [Online]. Available:

https://medium.com/@tanakornpiamsin/ติดตั้ง-mqtt-server-d31bcae85d0d

[7] **"Microcontroller Schematic Diagram"** [Online]. Available:

https://www.circuito.io/

ภาคผนวก

ภาคผนวก ก

โปรแกรมในการควบคุมของ NodeMCU

ก1. ห้องน้ำของบ้านจำลอง

```
#include<ESP8266WiFi.h>
#include<PubSubClient.h>
#include <Servo.h>
Servo myservo;
////////////////////////////callback/////////////////////////
int myCounter = 0;
int buttonPushCounter = 1;
int buttonPushCounter2 = 1;
int buttonPushCounter3 = 1;
void callback(char* topic,byte* payload,unsigned int length1){
Serial.print("message arrived[");
Serial.print(topic);
Serial.println("]");
for(int i=0;i<length1;i++)
{
  Serial.print(payload[i]);
 }
if(payload[0]==49) {digitalWrite(16,HIGH); buttonPushCounter=0; }    //1
if(payload[0]==50){digitalWrite(16,LOW); buttonPushCounter=1; myCounter = 0; }//2
if(payload[0]==51){ digitalWrite(5,HIGH);  buttonPushCounter2=0; } //3
if(payload[0]==52){ digitalWrite(5,LOW);   buttonPushCounter2=1;}//4
 if(payload[0]==53){  myservo.write(180); buttonPushCounter3=0; }//5
 if(payload[0]==54){   myservo.write(0);  buttonPushCounter3=1;}//6

Serial.println();
```

```
}
const char* mqtt_server="192.168.1.88";
WiFiClient espclient;
PubSubClient client(mqtt_server,1883,callback,espclient);
////////////////////////////DEFINE/////////////////////////
/*const byte sensor_1 = 12;
const byte sensor_2 = 13;
int value_1 = digitalRead(sensor_1);
int value_2 = digitalRead(sensor_2);
int flag1 = 0;
int flag2 = 0;*/
int buttonState = 0;         // B_LED
int lastButtonState = 0;     //
const int  buttonPin = 0;    //
const byte interruptPin = 0;
int buttonState2 = 0;        // B_FAN
int lastButtonState2 = 0;    //
const int  buttonPin2 = 2;   //
const byte interruptPin2 = 2;
int buttonState3 = 0;        // B_DOOR
int lastButtonState3 = 0;    //
const int  buttonPin3 = 14;  //
const byte interruptPin3 = 14;
//////////////SETUP//////////////
void setup() {
 Serial.begin(115200);
 Serial.print("connecting");
 WiFi.begin("MYNETWORK","helloecc");         //SSID,PASSWORD
 while(WiFi.status()!=WL_CONNECTED){
    delay(500);
```

```
  Serial.print(".");
 }
 Serial.println();
 reconnect();
////////////////////////////////////////////////////////////
 myservo.attach(4);  //servo pin
 myservo.write(0);
 pinMode(sensor_1,INPUT); //sensor_1
  pinMode(sensor_2,INPUT); //sensor_2
 pinMode(0,INPUT); //B1 LED
 //pinMode(2,INPUT); //B2 FAN
 pinMode(14,INPUT); //B3 DOOR
 //pinMode(5,OUTPUT);//FAN
 pinMode(16,OUTPUT);//LED
 pinMode(interruptPin, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(interruptPin), B_LED , RISING);
pinMode(interruptPin2, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(interruptPin2), B_FAN , RISING);
 pinMode(interruptPin3, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(interruptPin3), B_DOOR , RISING); }
//////////////////////RECONNECT////////////////////////////////////////////////
void reconnect(){
  while(WiFi.status()!=WL_CONNECTED){
     delay(500);
    Serial.print(".");
  }
  while(!client.connected()){
  if(client.connect("ESP8266Client1452")){
   Serial.println("connected");
    client.subscribe("BATHROOM");
```

```
  }
    else{
      Serial.print("failed,rc=");
      Serial.println(client.state());
       delay(500);
    }
  }
}
//////////////////////////Button//////////////////////
void B_LED(){
  buttonState = digitalRead(buttonPin);
  if (buttonState != lastButtonState) {
    if (buttonState == HIGH  ) {
      buttonPushCounter++;
    }
    else {
            }
       delay(50);}
  lastButtonState = buttonState;
 if(lastButtonState  == 1){
   buttonPushCounter--;
    if (buttonPushCounter %2 == 0) {
   client.publish("B_BATHROOM","1");
   Serial.println("1");
  } else {
    client.publish("B_BATHROOM","2");
     Serial.println("2");
  }
 }
}
```

```
//////////////////////////Button_2//////////////////////
void B_FAN(){
  buttonState2 = digitalRead(buttonPin2);


  if (buttonState2 != lastButtonState2) {
    if (buttonState2 == HIGH  ) {
      buttonPushCounter2++;
    }
    else {
            }
        delay(50);
  }
  lastButtonState2 = buttonState2;
 if(lastButtonState2  == 1){
   buttonPushCounter2--;
    if (buttonPushCounter2 %2 == 0) {
   client.publish("B_BATHROOM","3");
   Serial.println("3");
  } else {
    client.publish("B_BATHROOM","4");
     Serial.println("4");
  }
 }
}
//////////////////////////Button_3//////////////////////
void B_DOOR(){
  buttonState3 = digitalRead(buttonPin3);
  if (buttonState3 != lastButtonState3) {
    if (buttonState3 == HIGH  ) {
      buttonPushCounter3++;
```

```
    }
    else {
            }
        delay(50);
  }
  lastButtonState3 = buttonState3;
 if(lastButtonState3  == 1){
   buttonPushCounter3--;
    if (buttonPushCounter3 %2 == 0) {
   client.publish("B_BATHROOM","5");
   Serial.println("5");
  } else {
    client.publish("B_BATHROOM","6");
     Serial.println("6");
  }
 }
}
//////////////////////////Loop////////////////////////////
void loop() {
  // put your main code here, to run repeatedly:
    if(!client.connected()){
      reconnect();
    }
    client.loop();
/////////////////////////////////////////////////////////
/*int value_1 = digitalRead(sensor_1);
int value_2 = digitalRead(sensor_2);
  if(value_1 ==1 )
  {
    flag1 = 1;
```

```
  }
   if(value_2 ==1)
  {
    flag2 = 1;
  }
  if(flag1 == 1){
   if(value_1 ==0 &&value_2 ==1)
  {
    myCounter++;
    delay(300);
    flag1 = 0;
    char num[5];
String str;
str=String(myCounter);
String (str).toCharArray(num,5);
   client.publish("number",num);
  }
  }
  if(flag2 == 1){
  if(value_2 ==0 &&value_1 ==1)
  {
    myCounter--;
    if(myCounter<0){
      client.publish("number","0");
      }
       delay(300);
    flag2 = 0;
    char num[5];
String str;
str=String(myCounter);
```

```
String (str).toCharArray(num,5);
   client.publish("number",num);
  }}
if(alarm==1){
  tone(0, 2000); // Send 1KHz sound signal...
  delay(100);        // ...for 1 sec
  noTone(0);     // Stop sound...
  delay(100);
  } }
```

ก2. ห้องนอนของบ้านจำลอง

```
#include<ESP8266WiFi.h>
#include<PubSubClient.h>
#include <Servo.h>
Servo myservo;
////////////////////////////callback//////////////////////////
int buttonPushCounter = 1;
int buttonPushCounter2 = 1;
int buttonPushCounter3 = 1;
void callback(char* topic,byte* payload,unsigned int length1){
Serial.print("message arrived[");
Serial.print(topic);
Serial.println("]");
for(int i=0;i<length1;i++)
{
  Serial.print(payload[i]);
 }
if(payload[0]==49) {digitalWrite(16,HIGH); buttonPushCounter=0; }    //1
if(payload[0]==50){digitalWrite(16,LOW); buttonPushCounter=1; }//2
if(payload[0]==51){ myservo.write(180);  buttonPushCounter2=0; } //3
```

```
if(payload[0]==52){  myservo.write(0);  buttonPushCounter2=1;}//4
if(payload[0]==53){  digitalWrite(4,HIGH);  buttonPushCounter3=0; }//5
if(payload[0]==54){  digitalWrite(4,LOW);  buttonPushCounter3=1;}//6
Serial.println();
}
const char* mqtt_server="192.168.1.88";
WiFiClient espclient;
PubSubClient client(mqtt_server,1883,callback,espclient);
////////////////////////////////////DEFINE/////////////////////////////////
//const int ledPin = 16;      // the pin that the LED is attached to
int buttonState = 0;         // B_LED
int lastButtonState = 0;     //
const int  buttonPin = 12;    //
const byte interruptPin = 12;
int buttonState2 = 0;         // B_DOOR
int lastButtonState2 = 0;     //
const int  buttonPin2 = 2;    //
const byte interruptPin2 = 2;
int buttonState3 = 0;         // B_PLUG
int lastButtonState3 = 0;     //
const int  buttonPin3 = 14;    //
const byte interruptPin3 = 14;
////////////////////////////////////////////SETUP/////////////////////////////////////////////////
/
void setup() {
 Serial.begin(115200);
 Serial.print("connecting");
 WiFi.begin("MYNETWORK","helloecc");          //SSID,PASSWORD
 while(WiFi.status()!=WL_CONNECTED){
    delay(500);
```

```
    Serial.print(".");
   }
   Serial.println();
   reconnect();
myservo.attach(5);  //DOOR
 myservo.write(0);

 pinMode(12,INPUT); //B1
 pinMode(2,INPUT); //B2
 pinMode(14,INPUT); //B3
 pinMode(4,OUTPUT); //PLUG
 pinMode(16,OUTPUT); //LED
 pinMode(interruptPin, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(interruptPin), B_LED , RISING);
 pinMode(interruptPin2, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(interruptPin2), B_DOOR, RISING);
 pinMode(interruptPin3, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(interruptPin3), B_PLUG, RISING);
}
////////////////////////////////////RECONNECT//////////////////////////////////////////////
void reconnect(){
  while(WiFi.status()!=WL_CONNECTED){
     delay(500);
    Serial.print(".");
  }
  while(!client.connected()){
  if(client.connect("ESP8266Client1451")){
   Serial.println("connected");
    client.subscribe("BEDROOM");
  }
```

```
    else{
      Serial.print("failed,rc=");
      Serial.println(client.state());
       delay(500);
    }
  }
}
//////////////////////////Button/////////////////////
void B_LED(){
  buttonState = digitalRead(buttonPin);
  if (buttonState != lastButtonState) {
    if (buttonState == HIGH  ) {
      buttonPushCounter++;
    }
    else {
            }
       delay(50);
  }
  lastButtonState = buttonState;
 if(lastButtonState  == 1){
   buttonPushCounter--;
    if (buttonPushCounter %2 == 0) {
   client.publish("B_BEDROOM","1");
   Serial.println("1");
  } else {
    client.publish("B_BEDROOM","2");
     Serial.println("2");
  }
 }
}
```

```
//////////////////////////Button_2/////////////////////
void B_DOOR(){
   buttonState2 = digitalRead(buttonPin2);
   if (buttonState2 != lastButtonState2) {
    if (buttonState2 == HIGH  ) {
      buttonPushCounter2++;
    }
    else {
            }
       delay(50);
  }
  lastButtonState2 = buttonState2;
 if(lastButtonState2  == 1){
   buttonPushCounter2--;
    if (buttonPushCounter2 %2 == 0) {
   client.publish("B_BEDROOM","3");
   Serial.println("3");
  } else {
    client.publish("B_BEDROOM","4");
     Serial.println("4");
  }
 }
}
//////////////////////////Button_3/////////////////////
void B_PLUG(){
    buttonState3 = digitalRead(buttonPin3);
   if (buttonState3 != lastButtonState3) {
    if (buttonState3 == HIGH  ) {
      buttonPushCounter3++;
    }
```

```
    else {
            }
        delay(50);
  }
  lastButtonState3 = buttonState3;
 if(lastButtonState3  == 1){
   buttonPushCounter3--;
    if (buttonPushCounter3 %2 == 0) {
   client.publish("B_BEDROOM","5");
   Serial.println("5");
  } else {
    client.publish("B_BEDROOM","6");
     Serial.println("6");
  }
 }
}
//////////////////////////Loop//////////////////////////
void loop() {
  // put your main code here, to run repeatedly:
    if(!client.connected()){
      reconnect();
    }
    client.loop();
}
```

ก3. ห้องเก็บของของบ้านจำลอง

```
#include<ESP8266WiFi.h>
#include<PubSubClient.h>
#include <Servo.h>
Servo myservo;
```

```
//////////////////////////////callback//////////////////////////
int buttonPushCounter = 1;
int buttonPushCounter2 = 1;
int buttonPushCounter3 = 1;
int buttonPushCounter4 = 1;
void callback(char* topic,byte* payload,unsigned int length1){
Serial.print("message arrived[");
Serial.print(topic);
Serial.println("]");
for(int i=0;i<length1;i++)
{
  Serial.print(payload[i]);
 }
if(payload[0]==49) {digitalWrite(16,HIGH); buttonPushCounter=0; }    //1
if(payload[0]==50){digitalWrite(16,LOW); buttonPushCounter=1; }//2
if(payload[0]==51){  digitalWrite(5,HIGH);  buttonPushCounter2=0; }//3
if(payload[0]==52){  digitalWrite(5,LOW);  buttonPushCounter2=1;}//4
if(payload[0]==53){  digitalWrite(4,HIGH);  buttonPushCounter3=0; }//5
if(payload[0]==54){  digitalWrite(4,LOW);  buttonPushCounter3=1;}//6
if(payload[0]==55){ myservo.write(115);  buttonPushCounter4=0; } //7
if(payload[0]==56){  myservo.write(25);  buttonPushCounter4=1;}//8
Serial.println();
}
const char* mqtt_server="192.168.1.88";
WiFiClient espclient;
PubSubClient client(mqtt_server,1883,callback,espclient);
//////////////////////////////////////DEFINE/////////////////////////////////
const int ledPin = 16;      // the pin that the LED is attached to
int buttonState = 0;        // B_LED1
int lastButtonState = 0;    //
```

```
const int  buttonPin = 2;    //
const byte interruptPin = 2;
int buttonState2 = 0;         // B_LED2
int lastButtonState2 = 0;     //
const int  buttonPin2 = 14;    //
const byte interruptPin2 = 14;
int buttonState3 = 0;         // B_LED3
int lastButtonState3 = 0;     //
const int  buttonPin3 = 12;    //
const byte interruptPin3 = 12;
int buttonState4 = 0;         // B_DOOR
int lastButtonState4 = 0;     //
const int  buttonPin4 = 13;    //
const byte interruptPin4 = 13;
//////////////////////////////////////SETUP////////////////////////
void setup() {
 Serial.begin(115200);
 Serial.print("connecting");
 WiFi.begin("MYNETWORK","helloecc");         //SSID,PASSWORD
 while(WiFi.status()!=WL_CONNECTED){
     delay(500);
   Serial.print(".");
  }
  Serial.println();
  reconnect();
 ////////////////////////////////////////////////////////////
 myservo.attach(0);  //DOOR
 myservo.write(0);
 pinMode(2,INPUT); //B1
 pinMode(14,INPUT); //B2
```

```
 pinMode(12,INPUT); //B3
 pinMode(13,INPUT); //B4
 pinMode(16,OUTPUT); //LED1
 pinMode(5,OUTPUT); //LED2
 pinMode(4,OUTPUT); //LED3
  pinMode(interruptPin, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(interruptPin), B_LED1 , RISING);
 pinMode(interruptPin2, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(interruptPin2), B_LED2, RISING);
  pinMode(interruptPin3, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(interruptPin3), B_LED3, RISING);
 pinMode(interruptPin4, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(interruptPin4), B_DOOR, RISING);
}
////////////////////////////////////RECONNECT//////////////////////////////////////////////////
void reconnect(){
  while(WiFi.status()!=WL_CONNECTED){
     delay(500);
    Serial.print(".");
  }
  while(!client.connected()){
  if(client.connect("ESP8266Client1453")){
   Serial.println("connected");
   client.subscribe("STOREROOM");
  }
    else{
      Serial.print("failed,rc=");
      Serial.println(client.state());
       delay(500);
    }
```

```
  }
}
//////////////////////////Button//////////////////////
void B_LED1(){
    buttonState = digitalRead(buttonPin);
   if (buttonState != lastButtonState) {
    if (buttonState == HIGH  ) {
      buttonPushCounter++;
    }
    else {
            }
        delay(50);
  }
  lastButtonState = buttonState;
 if(lastButtonState  == 1){
   buttonPushCounter--;
    if (buttonPushCounter %2 == 0) {
   client.publish("B_STOREROOM","1");
   Serial.println("1");
  } else {
    client.publish("B_STOREROOM","2");
     Serial.println("2");
  }
 }
}
//////////////////////////Button_2//////////////////////
void B_LED2(){
    buttonState2 = digitalRead(buttonPin2);
   if (buttonState2 != lastButtonState2) {
    if (buttonState2 == HIGH  ) {
```

```
      buttonPushCounter2++;
    }
    else {
          }
       delay(50);
  }
  lastButtonState2 = buttonState2;
 if(lastButtonState2  == 1){
   buttonPushCounter2--;
    if (buttonPushCounter2 %2 == 0) {
   client.publish("B_STOREROOM","3");
   Serial.println("3");
  } else {
    client.publish("B_STOREROOM","4");
     Serial.println("4");
  }
 }
}
//////////////////////////Button_3/////////////////////////
void B_LED3(){
    buttonState3 = digitalRead(buttonPin3);
   if (buttonState3 != lastButtonState2) {
    if (buttonState3 == HIGH  ) {
      buttonPushCounter3++;
    }
    else {
          }
       delay(50);
  }
  lastButtonState3 = buttonState2;
```

```
 if(lastButtonState3  == 1){
   buttonPushCounter3--;
    if (buttonPushCounter3 %2 == 0) {
   client.publish("B_STOREROOM","5");
   Serial.println("5");
  } else {
    client.publish("B_STOREROOM","6");
     Serial.println("6");
  }
 }
}
//////////////////////////Button_4/////////////////////////
void B_DOOR(){
   buttonState4 = digitalRead(buttonPin4);
   if (buttonState4 != lastButtonState4) {
    if (buttonState4 == HIGH  ) {
      buttonPushCounter4++;
    }
    else {
            }
       delay(50);
  }
  lastButtonState4 = buttonState4;
 if(lastButtonState4  == 1){
   buttonPushCounter4--;
    if (buttonPushCounter4 %2 == 0) {
   client.publish("B_STOREROOM","7");
   Serial.println("7");
  } else {
    client.publish("B_STOREROOM","8");
```

```
    Serial.println("8");
  }
 } }
//////////////////////////Loop//////////////////////////
void loop() {
  // put your main code here, to run repeatedly:
   if(!client.connected()){
     reconnect();
   }
   client.loop();
}
```

ก4. ห้องนั่งเล่นของบ้านจำลอง

```
#include<ESP8266WiFi.h>
#include<PubSubClient.h>
#include <Servo.h>
Servo myservo;
int limit =0;
int count=1;
////////////////////////////callback//////////////////////////
int buttonPushCounter = 1;
int buttonPushCounter2 = 1;
int buttonPushCounter3 = 1;
int buttonPushCounter4 = 1;
void callback(char* topic,byte* payload,unsigned int length1){
Serial.print("message arrived[");
Serial.print(topic);
Serial.println("]");
for(int i=0;i<length1;i++)
{
```

```
  Serial.print(payload[i]);
 }
if(payload[0]==49) {digitalWrite(16,HIGH); buttonPushCounter=0; }    //1
if(payload[0]==50){digitalWrite(16,LOW); buttonPushCounter=1; }//2
if(payload[0]==51){  digitalWrite(5,HIGH);  buttonPushCounter3=0; }//3
if(payload[0]==52){  digitalWrite(5,LOW);  buttonPushCounter3=1;}//4
if(payload[0]==53){ count=1; limit=0; buttonPushCounter4=0; } //5
if(payload[0]==54){ count=0; limit=0; buttonPushCounter4=1;}//6
 Serial.println();
}
const char* mqtt_server="192.168.1.88";
WiFiClient espclient;
PubSubClient client(mqtt_server,1883,callback,espclient);
//////////////////////////////////////////////////DEFINE/////////////////////////
const int ledPin = 16;      // the pin that the LED is attached to
int buttonState = 0;         // B_LED1
int lastButtonState = 0;     //
const int  buttonPin = 13;    //
const byte interruptPin = 13;
int buttonState2 = 0;         // B_LED2
int lastButtonState2 = 0;    //
const int  buttonPin2 = 2;    //
const byte interruptPin2 = 2;
int buttonState3 = 0;         // B_DOOR
int lastButtonState3 = 0;     //
const int  buttonPin3 = 14;    //
const byte interruptPin3 = 14;
int buttonState4 = 0;         // B_LIMIT
int lastButtonState4 = 0;     //
const int  buttonPin4 = 12;    //
```

```
const byte interruptPin4 = 12;
//////////////////////////////////////////////////SETUP////////////////////////
void setup() {
 Serial.begin(115200);
 Serial.print("connecting");
 WiFi.begin("MYNETWORK","helloecc");          //SSID,PASSWORD
 while(WiFi.status()!=WL_CONNECTED){
    delay(500);
  Serial.print(".");
 }
 Serial.println();
 reconnect();
 ////////////////////////////////////////////////////////////
 myservo.attach(4);  //DOOR
 myservo.write(0);
  pinMode(13,INPUT); //B1
 pinMode(2,INPUT); //B2
 pinMode(14,INPUT); //B3
 pinMode(12,INPUT); //LIMIT
pinMode(16,OUTPUT); //LED1
 pinMode(5,OUTPUT); //LED2
 pinMode(interruptPin, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(interruptPin), B_LED1 , RISING);
 pinMode(interruptPin2, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(interruptPin2), B_LED2, RISING);
 pinMode(interruptPin3, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(interruptPin3), B_DOOR, RISING);
 pinMode(interruptPin4, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(interruptPin4), B_LIMIT, RISING);
}
```

```
//////////////////////////////////////////RECONNECT//////////////////////////
void reconnect(){
  while(WiFi.status()!=WL_CONNECTED){
     delay(500);
    Serial.print(".");
  }
  while(!client.connected()){
  if(client.connect("ESP8266Client1454")){
   Serial.println("connected");
      client.subscribe("LIVINGROOM");
  }
    else{
      Serial.print("failed,rc=");
      Serial.println(client.state());
       delay(500);
    }
  }
}
//////////////////////////Button//////////////////////
void B_LED1(){
    buttonState = digitalRead(buttonPin);
   if (buttonState != lastButtonState) {
    if (buttonState == HIGH  ) {
      buttonPushCounter++;
    }
    else {
            }
        delay(50);
  }
  lastButtonState = buttonState;
```

```
 if(lastButtonState  == 1){
   buttonPushCounter--;
    if (buttonPushCounter %2 == 0) {
   client.publish("B_LIVINGROOM","1");
   Serial.println("1");
  } else {
    client.publish("B_LIVINGROOM","2");
     Serial.println("2");
  }
 }
}
//////////////////////////Button_2//////////////////////
void B_LED2(){
    buttonState2 = digitalRead(buttonPin2);
   if (buttonState2 != lastButtonState2) {
    if (buttonState2 == HIGH  ) {
      buttonPushCounter2++;
    }
    else {
           }
       delay(50);
  }
  lastButtonState2 = buttonState2;
 if(lastButtonState2  == 1){
   buttonPushCounter2--;
    if (buttonPushCounter2 %2 == 0) {
   client.publish("B_LIVINGROOM","3");
   Serial.println("3");
  } else {
    client.publish("B_LIVINGROOM","4");
```

```
      Serial.println("4");
    }
   }
  }
  ////////////////////////Button_3/////////////////////
  void B_DOOR(){
     buttonState3 = digitalRead(buttonPin3);
    if (buttonState3 != lastButtonState3) {
     if (buttonState3 == HIGH  ) {
       buttonPushCounter3++;
     }
     else {
            }
        delay(50);
   }
   lastButtonState3 = buttonState3;
  if(lastButtonState3  == 1){
    buttonPushCounter3--;
     if (buttonPushCounter3 %2 == 0) {
    client.publish("B_LIVINGROOM","5");
    Serial.println("5");
   } else {
     client.publish("B_LIVINGROOM","6");
      Serial.println("6");
   }
  }
 }
 ////////////////////////Button_4/////////////////////
 void B_LIMIT(){
     buttonState4 = digitalRead(buttonPin4);
```

```
  if (buttonState4 != lastButtonState4) {
   if (buttonState4 == HIGH  ) {
     buttonPushCounter4++;
   }
   else {
          }
      delay(50);
 }
 lastButtonState4 = buttonState4;
if(lastButtonState4  == 1){
  buttonPushCounter4--;
   if (buttonPushCounter4 %2 == 0) {
  limit=0;
  Serial.println("7");
 } else {
limit=0;
    Serial.println("8");
 }
}
}
//////////////////////Loop/////////////////////////
void loop() {
 // put your main code here, to run repeatedly:
   if(!client.connected()){
     reconnect();
   }
   client.loop();
//////////////////////DOOR/////////////////////
  if(count==1){
   myservo.write(180);
```

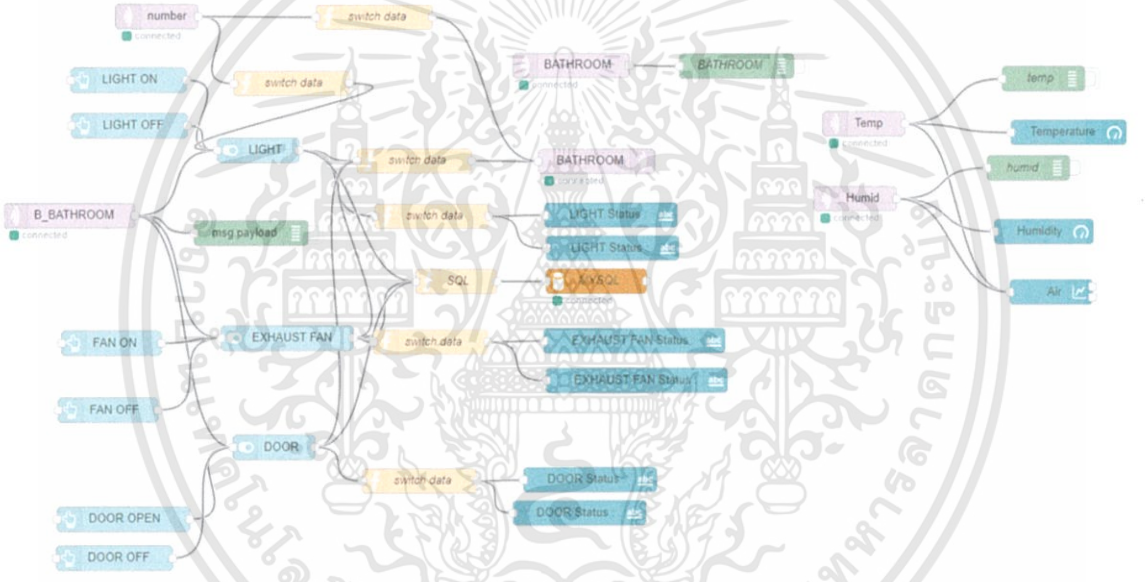
```
    }
  if(count==0){
  myservo.write(0);
    }
     if(limit==0){
    myservo.writeMicroseconds(1500);
    delay(100);
   }
////////////////////////////////////////////////////
}
```

ภาคผนวก ข

โปรแกรมรูปภาพในการควบคุมของ Node-RED

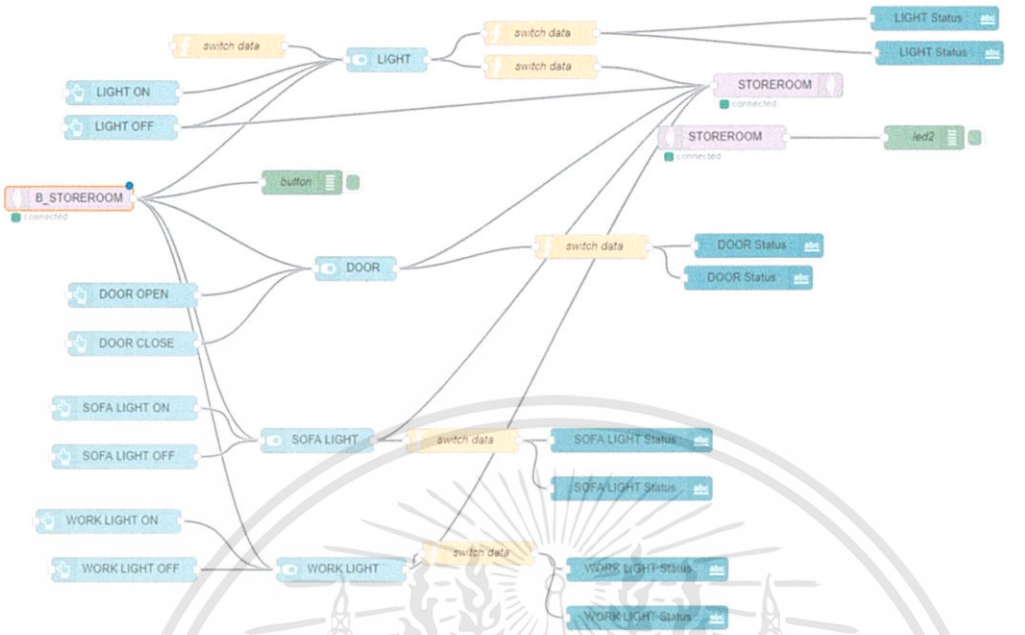
วิธีการเขียนโปรแกรมในส่วนการควบคุมใน Node-RED จะเป็นการเขียนโปรแกรมรูปแบบของ Graphical Programming หรือโปรแกรมแบบรูปภาพ ซึ่งจะมีความง่ายและสะดวกสะบายในการใช้งาน ผู้ใช้ที่เริ่มต้นสามารถใช้งานได้เลย

ข1. ห้องน้ำของบ้านจำลอง



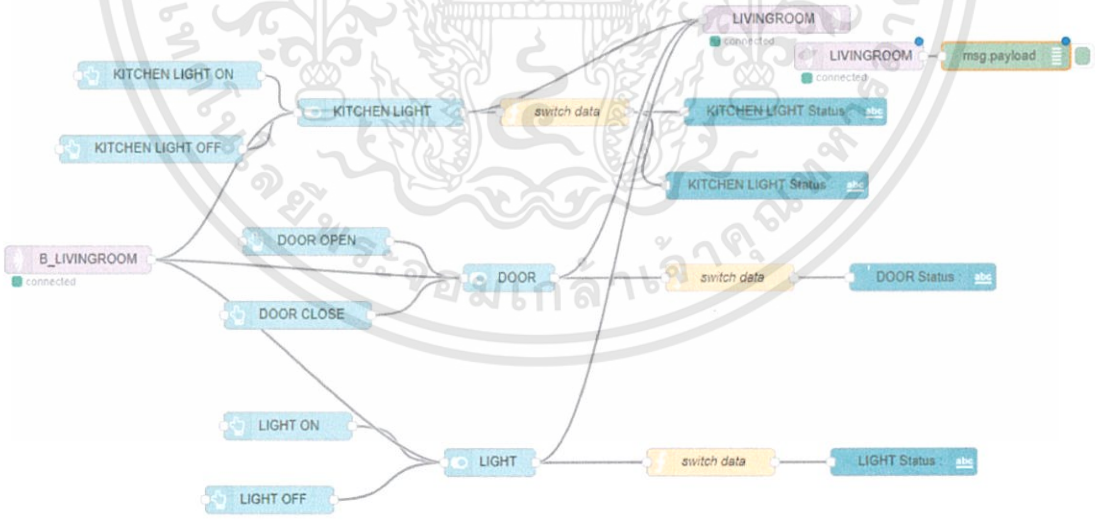
รูปที่ ข.1 โปรแกรมรูปภาพของห้องน้ำของบ้านจำลอง

ข2. ห้องเก็บของของบ้านจำลอง



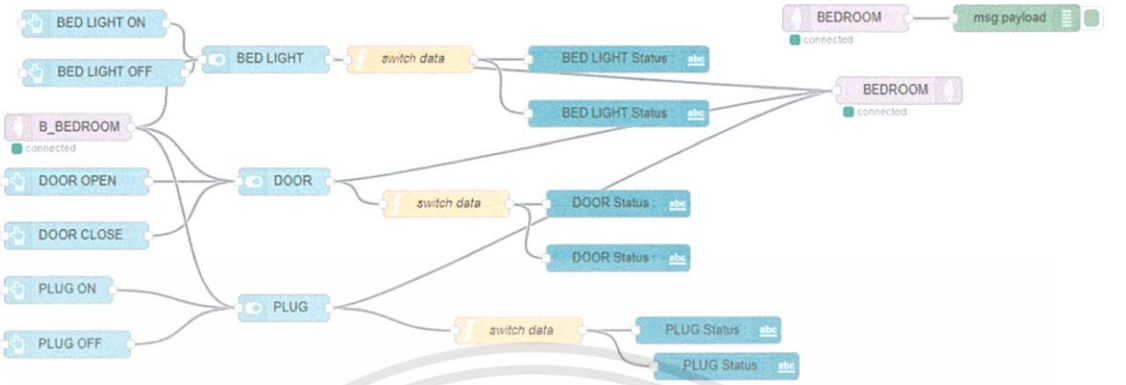
รูปที่ ข.2 โปรแกรมรูปภาพของห้องเก็บของของบ้านจำลอง

ข3. ห้องนั่งเล่นของบ้านจำลอง



รูปที่ ข.3 โปรแกรมรูปภาพของห้องนั่งเล่นของบ้านจำลอง

ข4. ห้องนอนของบ้านจำลอง

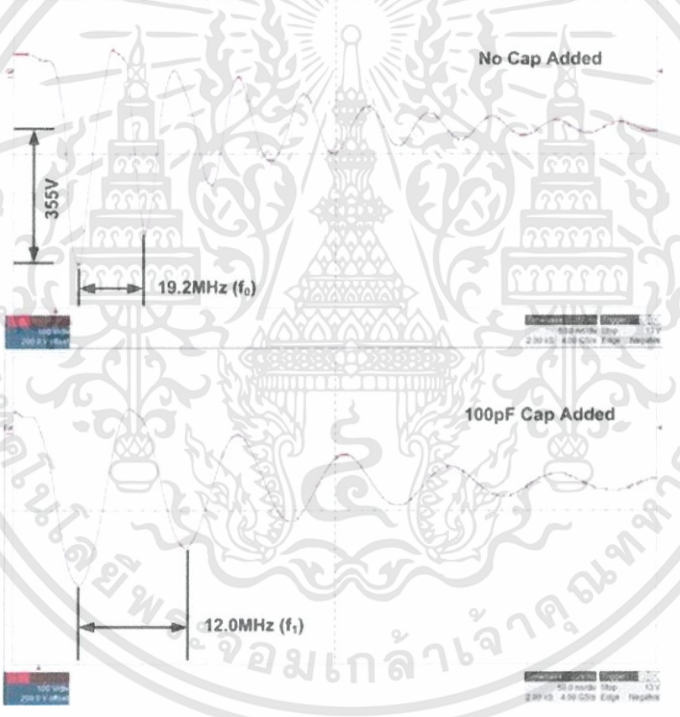


รูปที่ ข.4 โปรแกรมรูปภาพของห้องนอนของบ้านจำลอง

ภาคผนวก ค

การเลือกใช้ตัวเก็บประจุในการแก้ไขการเกิดสัญญาณรบกวนที่เกิดในวงจร

การเปิดปิดสวิตช์อย่างฉับพลันของวงจรจะทำให้เกิดสัญญาณรบกวนขึ้นในวงจร ส่งผลให้ไมโครคอนโทรลเลอร์ตรวจจับสัญญาณขาเข้า (Input) ได้ผิดเพี้ยน จึงทำการแก้ไขด้วยหลักการ RC-Snubber โดยการต่อตัวเก็บประจุ (Capacitor) และตัวต้านทาน (Resistor) อนุกรมกัน และคร่อมตัวสวิตช์ปุ่มกด สามารถคำนวณหาค่าขนาดของตัวเก็บประจุและตัวต้านทานได้ดังนี้



รูปที่ ค.1 เรกติไฟเออ (Rectifier) ที่ไม่ได้ต่อ RC-Snubber (บน) และ การเลื่อนของความถี่ (ล่าง)

Seven steps to calculate an R-C snubber	
Step 1	Measure the circuit's oscillation frequency (f_0). See Figure 1 (top).
Step 2	Add a capacitor (C_1) in parallel with the rectifier or FET and measure the shifted oscillation frequency (f_1). Select a C_1 value that is several times larger than the rectifier's stated typical capacitance at full-reverse voltage in the data sheet. In this example, the rectifier's reverse capacitance is 22pF, so I chose a 100pF value for C_1. A frequency shift of at least 50% is reasonable. See Figure 1 (bottom).
Step 3	Calculate the frequency shift ratio: $m = \frac{f_0}{f_1}$.
Step 4	Calculate the circuit's parasitic capacitance: $C_0 = \frac{C_1}{(m^2-1)}$.
Step 5	Calculate the circuit's parasitic inductance: $L = \frac{(m^2-1)}{(2\pi f_0)^2 C_1}$.
Step 6	Calculate the starting snubber capacitor value: $C_{snub} = 3 * C_0$.
Step 7	Calculate the starting snubber-resistor value: $R_{snub} = \sqrt{\frac{L}{C_0}}$.

รูปที่ ค.2 ขั้นตอนการคำนวนหาขนาดของตัวเก็บประจุและตัวต้านทานของ RC-Snubber

รูปที่ ค.3 แสดงการลดลงของยอดแหลม (Spike) และ Damping Effect จากการคำนวณ

การใช้ RC-Snubber จะช่วยลดการเกิดของยอดแหลม (Spike) และ Damping Effect โดยเราสามารถเลือกใช้ขนาดของตัวเก็บประจุให้มากกว่าหรือน้อยกว่าค่าจากการคำนวณได้ ตัวเก็บประจุ (C_{snub}) ที่มีขนาดใหญ่กว่าจะสามารถลดการเกิดยอดแหลมของแรงดันไฟฟ้า (Voltage-Spike) ได้ดีกว่า แต่จะทำให้เกิดการสูญเสียพลังงานที่ตัวต้านทาน (R_{snub}) มากขึ้น