ระบบควบคุมอุปกรณ์ไฟฟ้าแบบไร้สายภายในโรงเรือน

WIRELESS ELECTRICAL DEVICE CONTROLLER SYSTEM IN HOUSING

โดย

นายฐาปกรณ์ จันทะวงษ์ศรี

นายณัฐพันธุ์ แสงสุรีย์วัชชรา

นายณัฐวุฒิ ภู่สวาท

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2558

ระบบควบคุมอุปกรณ์ไฟฟ้าแบบไร้สายภายในโรงเรือน
WIRELESS ELECTRICAL DEVICE CONTROLLER SYSTEM IN HOUSING

โดย

นายฐาปกรณ์	จันทะวงษ์ศรี	55010308
นายณัฐพันธุ์	แสงสุรีย์วัชชรา	55010388
นายณัฐวุฒิ	ภู่สวาท	55010405

อาจารย์ที่ปรึกษา
ผศ.ดร.สมเกียรติ ฤกษ์วรัญญู
ผศ.สมภพ แก้วมีชัย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2558

2554 ผ่านการตรวจรูปเล่มแล้ว
(.....)
อาจารย์ที่ปรึกษา
18/5/59

2554 ผ่านการตรวจชิ้นงานแล้ว
(.....)
กรรมการผู้ตรวจชิ้นงาน
24/4/59

KMITL วิศวกรรมโทรคมนาคม Telecommunications Engineering

KMITL วิศวกรรมโทรคมนาคม Telecommunications Engineering

ปริญญานิพนธ์ปีการศึกษา 2558
ภาควิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง ระบบควบคุมอุปกรณ์ไฟฟ้าแบบไร้สายภายในโรงเรือน
WIRELESS ELECTRICAL DEVICE CONTROLLER SYSTEM IN HOUSING

ผู้จัดทำ

1.	นายฐาปกรณ์	จันทะวงษ์ศรี	55010308
2.	นายณัฐพันธุ์	แสงสุรีย์วัชรา	55010388
3.	นายณัฐวุฒิ	ภู่สวาท	55010405

..... อาจารย์ที่ปรึกษา
(ผศ.ดร.สมเกียรติ ฤกษ์วรัญญู)

..... อาจารย์ที่ปรึกษาร่วม
(ผศ.สมภพ แก้วมีชัย)

กิตติกรรมประกาศ

การดำเนินโครงงาน “ระบบควบคุมอุปกรณ์ไฟฟ้าแบบไร้สายภายในโรงเรือน” จะไม่สามารถสำเร็จลุล่วงไปได้ด้วยดีได้เลย หากขาดการสนับสนุน และกำลังใจจากหลายๆ ฝ่าย เช่น

การได้รับทุนอุดหนุนโครงการจากสาขาวิชาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ผศ.ดร.สมเกียรติ ฤกษ์วรัญญู อาจารย์ที่ปรึกษาโครงงาน และผศ.สมภพ แก้วมีชัย อาจารย์ที่ปรึกษาร่วม สำหรับคำปรึกษา คำแนะนำและแนวทางการแก้ไขเมื่อเกิดปัญหาในการจัดทำโครงงาน รวมทั้งสนับสนุนสถานที่ เครื่องมือ และอุปกรณ์ต่างๆ ที่จำเป็นต้องใช้ในระหว่างการจัดทำโครงงาน

คุณพ่อ คุณแม่ และครอบครัวที่คอยห่วงใย และมอบกำลังใจให้ในยามที่เกิดปัญหา รวมทั้งคำแนะนำเพิ่มเติมที่มีประโยชน์ต่อการจัดทำโครงงาน

พี่ๆ ปริญญาโทและทุกๆ คนในห้อง T110 ที่ได้คอยให้คำแนะนำ คำปรึกษาและให้ความรู้เพิ่มเติมเกี่ยวกับการจัดทำโครงงาน

เพื่อนๆ สาขาวิชาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่คอยให้คำปรึกษา ช่วยเหลือและกำลังใจในทุกด้านๆ

ผู้จัดทำขอขอบพระคุณทุกๆ ท่านเป็นอย่างสูง ณ ที่นี้ ที่ได้ช่วยให้การจัดทำโครงงานในครั้งนี้สำเร็จลุล่วงไปได้ด้วยดี

นายฐาปกรณ์ จันทะวงษ์ศรี
นายณัฐพันธุ์ แสงสุรีย์วัชชรา
นายณัฐวุฒิ ภู่สวาท

ผู้จัดทำ

ระบบควบคุมอุปกรณ์ไฟฟ้าแบบไร้สายภายในโรงเรือน
WIRELESS ELECTRICAL DEVICE CONTROLLER SYSTEM IN HOUSING

โดย นายฐาปกรณ์ จันทะวงษ์ศรี 55010308
นายณัฐพันธุ์ แสงสุรีย์วัชชรา 55010388
นายณัฐวุฒิ ภู่สวาท 55010405

อาจารย์ที่ปรึกษา ผศ.ดร.สมเกียรติ ฤกษ์วรัญญู
อาจารย์ที่ปรึกษาร่วม ผศ.สมภพ แก้วมีชัย

บทคัดย่อ

อุตสาหกรรมการเลี้ยงสัตว์ได้ผลิตอาหารซึ่งเป็นปัจจัยสำคัญในการดำรงชีวิตของมนุษย์ โดยการเจริญเติบโตของสัตว์นั้นขึ้นอยู่กับอาหารและสิ่งแวดล้อมที่ต้องทำการดูแลให้เหมาะสมกับแต่ละช่วงวัยของสัตว์เพื่อทำให้สัตว์มีการเจริญเติบโตเต็มที่ เราจึงนำเสนอเกี่ยวกับระบบควบคุมอุปกรณ์ไฟฟ้าในโรงเรือนเลี้ยงสัตว์ที่ประกอบไปด้วยพัดลมระบายอากาศที่มีการตั้งเวลาควบคุมการทำงานและมอเตอร์ปั๊มน้ำที่มีการควบคุมปริมาณการผสมสารสำหรับการเลี้ยงสัตว์เช่น วิตามิน หรือยาบำรุง กับน้ำให้มีอัตราส่วนที่เหมาะสม สามารถตรวจสอบการทำงานของอุปกรณ์ทั้งหมดได้จากข้อมูลซึ่งมาจากการวัดกระแสไฟฟ้าและแรงดันไฟฟ้าด้วยเซ็นเซอร์ โดยควบคุมด้วยระบบไร้สายผ่านซิกบี ซึ่งผู้ใช้สามารถควบคุมการทำงานจากระยะไกลได้ผ่านโปรแกรมที่เชื่อมต่อกับเซิร์ฟเวอร์และระบบฐานข้อมูลซึ่งสามารถแจ้งเตือนให้ผู้ใช้ทราบถึงความผิดปกติในการทำงานของอุปกรณ์ไฟฟ้า

ABSTRACT

Animal food industry produces food, which are important for all of us in daily life. As you know, the growth of these animals depends on feeding and surrounding, which have to be paid attention for suitability at each age. So we develop the wireless electrical device controller system in housing including ventilation fans, which are controlled by setting time and water pump motors, which control the amount of blending foods such as blending Vitamins and

medicines, with water the system can be monitored about voltage and current from everywhere by using Zigbee modules, which user can control by programming data into server system and database system to notify about working status of each electrical device.

สารบัญ

สารบัญ(ต่อ)

สารบัญรูป

สารบัญรูป(ต่อ)

หน้า

รูปที่

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

หน้า

รูปที่

สารบัญตาราง

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันเทคโนโลยีการสื่อสารข้อมูลแบบไร้สายนั้นได้เข้ามามีบทบาทอย่างมากในชีวิตประจำวันของเรา ซึ่งเทคโนโลยีเหล่านี้มีบทบาทในการพัฒนาของอุตสาหกรรมต่างๆในประเทศอย่างมากมาย แต่เมื่อเรามองในอีกด้านหนึ่งซึ่งก็คือของอุตสาหกรรมทางด้านการเกษตรแล้วกลับพบว่า ยังคงไม่ค่อยพัฒนามากเท่าที่ควรนัก โดยเฉพาะการนำเทคโนโลยีเข้ามาใช้นั้นยังถือว่ามีน้อยเกินไป

จากปัญหาดังกล่าวจึงได้ศึกษาและนำเสนอแนวทางการแก้ไขปัญหาโดยใช้ความรู้เกี่ยวกับการเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์และการสื่อสารข้อมูลไร้สายผ่านซิกบี ให้เข้ามามีบทบาทในการจัดการและช่วยพัฒนาระบบควบคุมภายในโรงเรือน ให้มีความสะดวกสบายมากยิ่งขึ้น โดยระบบนี้จะช่วยจัดการด้านต่างๆ เกี่ยวกับการเลี้ยงดู การให้อาหาร ตลอดจนการฉีดพ่นยารักษาความสะอาดของโรงเรือน ทำให้เกษตรกรทำงานได้อย่างสะดวกสบายและมีประสิทธิภาพยิ่งขึ้น โดยสามารถควบคุมระบบได้อย่างถูกต้องและแม่นยำโดยผ่านระบบการสื่อสารไร้สาย

1.2 วัตถุประสงค์

1) เพื่อสร้างระบบควบคุมไฟฟ้าในโรงเรือนผ่านระบบไร้สาย
2) เพื่อศึกษาการจัดเก็บบันทึกข้อมูลในระบบฐานข้อมูล
3) เพื่อศึกษาการทำงานและการรับส่งข้อมูลด้วยโมดูลไร้สายสายซิกบี
4) เพื่อศึกษาถึงทฤษฎีและหลักการทำงานของวงจรเฟสล็อคลูป
5) เพื่อศึกษาและออกแบบการสร้างเว็บเพื่อสำหรับสังเกตการณ์การทำงานระบบไฟฟ้า
6) เพื่อศึกษาการเขียนโปรแกรมและออกแบบระบบควบคุมด้วยโปรแกรมภาษา C#
7) เพื่ออำนวยความสะดวกและเพิ่มประสิทธิภาพในการดูแลสัตว์ภายในโรงเรือน

1.3 ขอบเขตของปริญญานิพนธ์

ปริญญานิพนธ์นี้มีจุดประสงค์เพื่อออกแบบและพัฒนาระบบควบคุมอุปกรณ์ไฟฟ้าภายในโรงเรือน เพื่อทำการสังเกตการณ์การทำงานของอุปกรณ์ไฟฟ้าโดยจะมีเซ็นเซอร์วัดกระแสและแรงดันวัดค่าเพื่อแสดงการทำงานและบันทึกค่าข้อมูลที่อ่านได้จากเซ็นเซอร์ลงในระบบฐานข้อมูลผ่านระบบเครือข่าย โดยผู้ใช้สามารถดูข้อมูลและสั่งการได้จากเว็บบราวเซอร์และโปรแกรมที่เขียนขึ้นด้วยภาษา C# โดยการเชื่อมต่อนั้นจะมีการใช้ข้อมูลในระบบฐานข้อมูลร่วมด้วย

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

ในบทนี้จะอธิบายเกี่ยวกับไมโครคอนโทรลเลอร์จะส่งค่าจากตัวเซ็นเซอร์ต่างๆ คือเซ็นเซอร์วัดกระแสและแรงดันผ่านซิกบี โดยโปรแกรม C# จะนำค่าที่ได้เก็บในฐานข้อมูลพร้อมกับเวลา วันที่และหมายเลขอุปกรณ์ พร้อมแสดงกราฟกระแสและแรงดันเทียบกับเวลาแบบเรียลไทม์ผ่านหน้าโปรแกรม C# และกราฟแบบเลือกวันที่ เวลาและหมายเลขอุปกรณ์แสดงผลผ่านหน้าเว็บโดยแสดงถึงการทำงานของอุปกรณ์ไฟฟ้าต่างๆ โปรแกรม C# จะอ่านคำสั่งการควบคุมอุปกรณ์ไฟฟ้าจากฐานข้อมูลและประมวลผลส่งคำสั่งผ่านซิกบีไปยังไมโครคอนโทรลเลอร์สั่งจ่ายลอจิกทางดิจิตอลเพื่อควบคุมอุปกรณ์ไฟฟ้าต่อไป โดยสามารถกำหนดคำสั่งควบคุมอุปกรณ์ไฟฟ้าและบันทึกลงในฐานข้อมูลผ่านหน้าโปรแกรม C# และหน้าเว็บ

2.1 ไมโครคอนโทรลเลอร์ (Microcontroller)

ไมโครคอนโทรลเลอร์เป็นอุปกรณ์ควบคุมขนาดเล็กที่มีความสามารถคล้ายๆกับคอมพิวเตอร์เพราะภายในไมโครคอนโทรลเลอร์ได้รวมหน่วยประมวลผลกลาง (CPU) หน่วยความจำ (Memory) และพอร์ต (Port) ซึ่งเป็นส่วนประกอบหลักที่สำคัญของระบบคอมพิวเตอร์เข้าไว้ด้วยกัน

โครงสร้างโดยทั่วไปของไมโครคอนโทรลเลอร์นั้น สามารถแบ่งออกมาได้เป็น 5 ส่วนใหญ่ๆ ดังต่อไปนี้

1. หน่วยประมวลผลกลางหรือซีพียู (CPU : Central Processing Unit)

2. หน่วยความจำ (Memory) สามารถแบ่งออกเป็น 2 ส่วน คือ หน่วยความจำที่มีไว้สำหรับเก็บโปรแกรมหลัก (Program Memory) เปรียบเสมือนฮาร์ดดิสก์ของเครื่องคอมพิวเตอร์ตั้งโต๊ะ คือข้อมูลใดๆ ที่ถูกเก็บไว้ในนี้จะไม่สูญหายไปแม้ไม่มีไฟเลี้ยง อีกส่วนหนึ่งคือหน่วยความจำข้อมูล (Data Memory) ใช้เป็นเหมือนกระดาษทดในการคำนวณของซีพียู และเป็นที่พักข้อมูลชั่วคราวขณะทำงาน แต่หากไม่มีไฟเลี้ยง ข้อมูลก็จะหายไปคล้ายกับหน่วยความจำ (RAM) ในเครื่องคอมพิวเตอร์ทั่วๆ ไป แต่สำหรับไมโครคอนโทรลเลอร์สมัยใหม่ หน่วยความจำข้อมูลจะมีทั้งที่เป็นหน่วยความจำแรม ซึ่งข้อมูลจะหายไปเมื่อไม่มีไฟเลี้ยง และเป็นอีอีพรอม (EEPROM : Erasable Electrically Read-Only Memory) ซึ่งสามารถเก็บข้อมูลได้แม้ไม่มีไฟเลี้ยงก็ตาม

3. ส่วนติดต่อกับอุปกรณ์ภายนอกหรือพอร์ต (Port) มี 2 ลักษณะคือ พอร์ตอินพุต (Input Port) และพอร์ตส่งสัญญาณหรือพอร์ตเอาท์พุต (Output Port) ส่วนนี้จะใช้ในการเชื่อมต่อกับอุปกรณ์ภายนอกถือว่าเป็นส่วนที่สำคัญมาก ใช้ร่วมกันระหว่างพอร์ตอินพุตเพื่อรับสัญญาณอาจจะด้วยการกดสวิตช์เพื่อนำไปประมวลผลและส่งไปพอร์ตเอาท์พุตเพื่อแสดงผลเช่น การติดสว่างของหลอดไฟ เป็นต้น

4. ช่องทางเดินของสัญญาณหรือบัส (BUS) คือเส้นทางการแลกเปลี่ยนสัญญาณข้อมูลระหว่างซีพียู หน่วยความจำและพอร์ต เป็นลักษณะของสายสัญญาณจำนวนมากอยู่ภายในตัวไมโครคอนโทรลเลอร์ โดยแบ่งเป็นบัสข้อมูล (Data Bus), บัสแอดเดรส (Address Bus) และบัสควบคุม (Control Bus)

5. วงจรกำเนิดสัญญาณนาฬิกานับเป็นส่วนประกอบที่สำคัญมากอีกส่วนหนึ่ง เนื่องจากการทำงานที่เกิดขึ้นในตัวไมโครคอนโทรลเลอร์ จะขึ้นอยู่กับการกำหนดจังหวะ หากสัญญาณนาฬิกามีความถี่สูง จังหวะการทำงานก็จะสามารถทำได้ถี่ขึ้นส่งผลให้ไมโครคอนโทรลเลอร์ตัวนั้น มีความเร็วในการประมวลผลสูงตามไปด้วย

2.1.1 Arduino

Arduino เป็นบอร์ดไมโครคอนโทรเลอร์ตระกูล AVR ที่มีการพัฒนาแบบ Open Source คือมีการเปิดเผยข้อมูลทั้งด้าน Hardware และ Software ตัวบอร์ด Arduino ถูกออกแบบมาให้ใช้งานได้ง่าย ดังนั้นจึงเหมาะสำหรับผู้เริ่มต้นศึกษา ทั้งนี้ผู้ใช้งานยังสามารถดัดแปลงเพิ่มเติมพัฒนาต่อยอดทั้งตัวบอร์ดหรือโปรแกรมต่อได้อีกด้วย โดยนิยมใช้ AVR ในตระกูล ATMEGA88, ATMEGA168, ATMEGA328 แสดงดังในรูปที่ 2.1

รูปที่ 2.1 บอร์ดไมโครคอนโทรลเลอร์ Arduino [1]

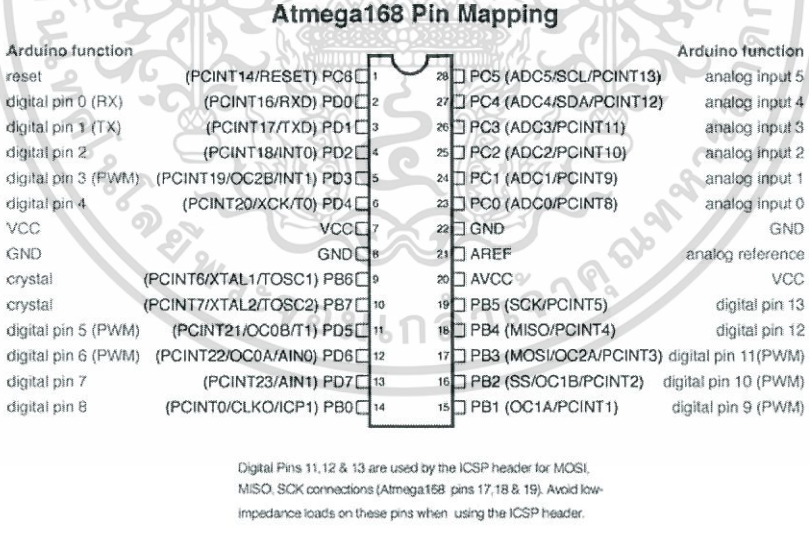
2.1.2 ไมโครคอนโทรลเลอร์ตระกูล AVR ATMega168

ไมโครคอนโทรลเลอร์ตระกูล AVR เป็นหนึ่งในไมโครคอนโทรลเลอร์ที่ผลิตโดยบริษัท ATMEL AVR ทำงานใช้สัญญาณนาฬิกาเพียง 1 ลูก นับเป็นไมโครคอนโทรลเลอร์ที่มีประสิทธิภาพและความสามารถสูง แบ่งออกเป็นหลายอนุกรม ในแต่ละอนุกรมยังแบ่งออกเป็นหลายเบอร์ เพื่อรองรับความต้องการที่แตกต่างของผู้ใช้งาน ในขณะที่ยังคงประสิทธิภาพที่เท่ากัน สำหรับไมโครคอนโทรลเลอร์ AVR ที่จะใช้ในโครงงานนี้จะเป็นเบอร์ ATMega168 โดยที่รายละเอียดและคุณสมบัติภายในไมโครคอนโทรลเลอร์ ATMega168 มีดังต่อไปนี้

2.1.2.1 คุณสมบัติที่สำคัญคือบรรจุอยู่ภายในตัวถังแบบ 28 พิน ทำงานได้ด้วยความถี่สูงสุด 20 เมกะเฮิรตซ์ ทำงานใช้สัญญาณนาฬิกาเพียง 1 ลูก มีรีจิสเตอร์ใช้งานทั่วไปขนาด 8 บิต จำนวน 32 ตัว หน่วยความจำรอม (ROM: Read Only Memory) แบบแฟลช (Flash) (มีโหมดป้องกันหน่วยความจำ) ขนาด 16 กิโลไบต์ (เขียน/ลบได้ 10,000 ครั้ง) หน่วยความจำข้อมูลแบบอีพีรอม (EPROM: Erasable Programmable Read Only Memory) มีโหมดป้องกันหน่วยความจำขนาด 512 กิโลไบต์ (เขียน/ลบได้ 100,000 ครั้ง) หน่วยความจำข้อมูลแบบ เอสแรม (SRAM: Static Random-Access Memory) 1 กิโลไบต์ พร้อมปรีสเกล (Pre-Scale) ที่มีระบบตรวจสอบความผิดพลาดในการทำงานของซอฟต์แวร์ (Software) มีโมดูลแปลงสัญญาณอนาล็อกเป็นดิจิตอล (ADC: Analog-to-Digital Converter) ขนาด 10 บิต มากถึง 8 ช่อง โมดูลเปรียบเทียบแรงดันอนาล็อก (Analog) การสื่อสารข้อมูลอนุกรมมีทั้งแบบยูเออาร์ที (UART:Universal Asynchronous Receiver Transmitters) หรือแบบมาตรฐานการเชื่อมต่อแบบอนุกรม RS232

2.1.2.2 การพัฒนาโปรแกรมของไมโครคอนโทรลเลอร์ ด้วย Arduino ซึ่งเป็นฟรีซอฟท์แวร์สำหรับใช้พัฒนาการใช้งานไมโครคอนโทรลเลอร์ ในตระกูล AVR โดยภาษาที่ใช้กับบอร์ดนี้จะเป็นลักษณะของภาษาซีและซีพลัสพลัส (C/C++) และมีแหล่งข้อมูลต่างๆ พร้อมให้เรียกใช้งานได้ทันทีมากมาย ครอบคลุมการติดต่อกับอินพุต/เอาท์พุต (Input/Output) ต่างๆได้อย่างกว้างขวาง

2.1.2.3 ขาพอร์ตอินพุตเอาท์พุตกับ Arduino มีการใช้งานขาพอร์ตอินพุตเอาท์พุตกับ Arduino แสดงดังรูปที่ 2.2



รูปที่ 2.2 ขาต่างๆของไมโครคอนโทรลเลอร์ Atmega168 [2]

โดยที่มีรายละเอียดในแต่ละขาพอร์ตดังนี้

Port D (PD0..PD7) : เป็นขาสัญญาณของดิจิตอล [0-7]

Port B (PB0..PB5) : เป็นขาสัญญาณของดิจิตอล [8-13]

Port C (PC0..PC5) : เป็นขาสัญญาณของอนาล็อก [0-5]

VCC : เป็นขาแรงดันไฟตรง

GND : เป็นขากราวด์ (Ground)

PC6 /RESET : เป็นขารีเซตวงจร

AVCC : เป็นขาแรงดันสำหรับพอร์ต A

AREF : เป็นขาแรงดันอนาล็อกอ้างอิงสำหรับโมดูลแปลงสัญญาณอนาล็อกเป็นดิจิตอล

2.1.3 การสื่อสารข้อมูล

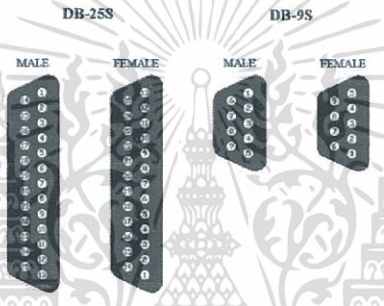
การสื่อสารข้อมูลเป็นการส่งสัญญาณออกจากอุปกรณ์ส่งและรับสัญญาณเข้าไปยังอุปกรณ์รับ การถ่ายโอนข้อมูลสามารถจัดจำแนกได้ 2 แบบ คือการถ่ายโอนข้อมูลแบบขนานและการถ่ายโอนข้อมูลแบบอนุกรม

2.1.3.1 ช่องทางสื่อสารแบบขนาน (Parallel Port) คือการถ่ายโอนข้อมูลแบบขนานเป็นการส่งข้อมูลทีละ 1 ไบต์ หรือ 8 บิต จากอุปกรณ์ส่งไปยังอุปกรณ์รับ ตัวกลางในการส่งข้อมูลจึงต้องมีช่องทางในการส่งข้อมูลอย่างน้อย 8 ช่องทาง โดยมากจะเป็นสายสัญญาณแบบขนาน ระยะทางของสายสัญญาณแบบขนานระหว่างสองเครื่องไม่ควรยาวเกิน 100 ฟุต เพราะอาจทำให้เกิดปัญหาสัญญาณสูญหายไปกับความต้านทานของสาย นอกจากการส่งข้อมูลหลักแล้วอาจจะมีทางเดินของสัญญาณควบคุมอื่นๆ อีก

2.1.3.2 ช่องทางสื่อสารแบบอนุกรม (Serial Port) ในนั้นจะมีความเร็วในการสื่อสารช้ากว่าแบบขนานเนื่องจากการเคลื่อนย้ายข้อมูลแบบอนุกรมนั้นเป็นการส่งข้อมูลครั้งละ 1 บิต แต่พอร์ตขนานนั้นสามารถส่งข้อมูลทีละหลายๆ บิตพร้อมๆ กันได้ ข้อดีของการสื่อสารข้อมูลแบบอนุกรมคือ สามารถส่งข้อมูลได้ในระยะทางที่ไกลกว่าแบบขนาน ซึ่งพอร์ตสื่อสารอนุกรมสามารถสื่อสารแบบไม่เข้าจังหวะ (Asynchronous) ซึ่งใช้สัญญาณนาฬิกาในการควบคุมจังหวะของการรับส่งสัญญาณ หรือแบบเข้าจังหวะ (Synchronous) จะใช้รูปแบบของการส่งข้อมูล (Bit Pattern) เป็นตัวกำหนดว่าส่วนไหนเป็นส่วนเริ่มต้นข้อมูล ส่วนไหนเป็นตัวข้อมูล ส่วนไหนจะเป็นตัว

ตรวจสอบความถูกต้องของข้อมูล และส่วนไหนเป็นส่วนปิดท้ายของข้อมูล โดยต้องกำหนดให้สัญญาณนาฬิกาตรงกันทั้งภาคส่งและภาครับ ซึ่งถ้าเป็นการสื่อสารแบบไม่เข้าจังหวะจะสามารถทำการส่งข้อมูลและรับข้อมูลไปพร้อมๆ กันได้ ซึ่งการสื่อสารแบบนี้เรียกว่าฟูลดูเพล็กซ์ (Full Duplex) ซึ่งในการสื่อสารแบบนี้จะใช้สายสัญญาณอินพุต / เอาท์พุตทั้งหมด 2 เส้น ประกอบไปด้วย สายในการส่งข้อมูล (Tx: Transmitter) และสายในการรับข้อมูล (Rx: Receiver)

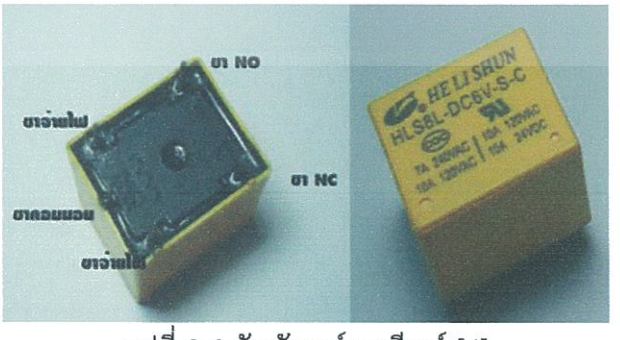
2.1.3.3 มาตรฐาน RS232 DB9 และ DB25 คือ โดยปกติไมโครคอมพิวเตอร์จะมีพอร์ตอนุกรมที่เรียกว่า RS 232 อยู่ในตัว โดยพอร์ต (Port) นี้ทำหน้าที่รับและส่งข้อมูลในแบบอนุกรม หรือเป็นชนิดของคอนเน็คเตอร์ (Connecter) คือ DB9 เป็นคอนเน็คเตอร์แบบ 9 ขา และ DB25 เป็นคอนเน็คเตอร์แบบ 25 ขา ทั้งนี้ลักษณะของคอนเน็คเตอร์แบบ DB9 และ DB25 แสดงได้ดังรูปที่ 2.3



รูปที่ 2.3 ลักษณะคอนเน็คเตอร์แบบ DB9 และ DB25 [3]

2.2 รีเลย์ (Relay)

รีเลย์เป็นอุปกรณ์ไฟฟ้าชนิดหนึ่ง ซึ่งทำหน้าที่ตัดต่อวงจรแบบเดียวกับสวิตช์ โดยควบคุมการทำงานด้วยไฟฟ้าRelay มีหลายประเภท ตั้งแต่ Relay ขนาดเล็กที่ใช้ในงานอิเล็กทรอนิกส์ทั่วไป จนถึง Relay ขนาดใหญ่ที่ใช้ในงานไฟฟ้าแรงสูง โดยมีรูปร่างหน้าตาแตกต่างกันออกไป แต่มีหลักการทำงานที่คล้ายคลึงกัน สำหรับการนำ Relay ไปใช้งาน จะใช้ในการตัดต่อวงจร ทั้งนี้ Relay ยังสามารถเลือกใช้งานได้หลากหลายรูปแบบแสดงได้ดังรูปที่ 2.4

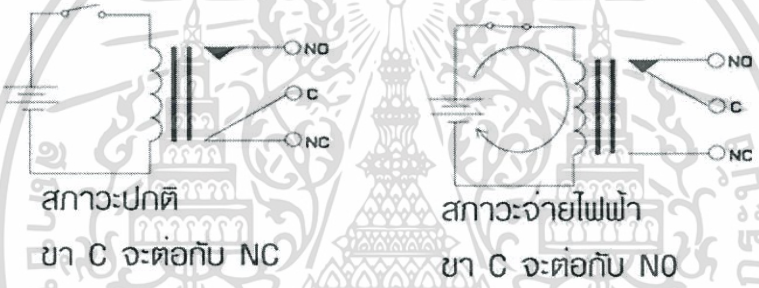


รูปที่ 2.4 สัญลักษณ์ของรีเลย์ [4]

ภายใน Relay จะประกอบไปด้วยขดลวดและหน้าสัมผัส NC (Normally Close) เป็นหน้าสัมผัสปกติปิด โดยในสภาวะปกติหน้าสัมผัสนี้จะต่อเข้ากับขา COM (Common) และจะลอยหรือไม่สัมผัสกันเมื่อมีกระแสไฟฟ้าไหลผ่านขดลวด

หน้าสัมผัส NO (Normally Open) เป็นหน้าสัมผัสปกติเปิด โดยในสภาวะปกติจะลอยอยู่ ไม่ถูกต่อกับขา COM (Common) แต่จะเชื่อมต่อกันเมื่อมีกระแสไฟไหลผ่านขดลวด

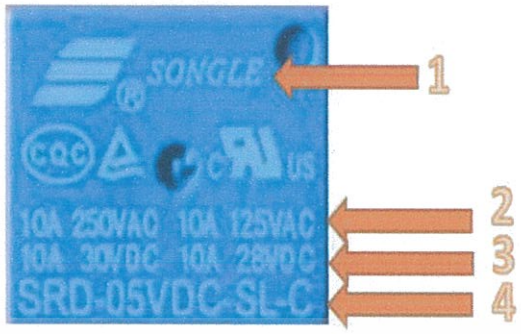
ขา COM (Common) เป็นขาที่ถูกใช้งานร่วมกันระหว่าง NC และ NO ขึ้นอยู่กับว่า ขณะนั้นมีกระแสไฟฟ้าไหลผ่านขดลวดหรือไม่ หน้าสัมผัสใน Relay 1 ตัวอาจมีมากกว่า 1 ชุด ขึ้นอยู่กับผู้ผลิตและลักษณะของงานที่ถูกนำไปใช้การทำงานของรีเลย์ คือ เมื่อมีกระแสไฟฟ้าไหลผ่านขดลวด จะทำให้ขดลวดเกิดสนามแม่เหล็กไปดึง แผ่นหน้าสัมผัส C ดึงลงมา แตะหน้าสัมผัส NO ทำให้มีกระแสไหลผ่านหน้าสัมผัสไปได้แสดงได้ดังรูปที่ 2.5



รูปที่ 2.5 การทำงานของรีเลย์ [5]

เราจะใช้งาน Relay แบบ SPDT (Single Pole Double Throw) หลักการทำงานของ Relay นั้น ในส่วนของขดลวด เมื่อมีกระแสไฟฟ้าไหลผ่าน จะทำให้ขดลวดเกิดการเหนี่ยวนำและทำหน้าที่เสมือนแม่เหล็กไฟฟ้า ส่งผลให้ขา COM ที่เชื่อมต่ออยู่กับหน้าสัมผัส NC (ในสภาวะที่ยังไม่เกิดการเหนี่ยวนำ) ย้ายกลับเชื่อมต่อกับหน้าสัมผัส NO แทน และปล่อยให้ขา NC ลอย เมื่อมองที่ขา NC กับ COM และ NO กับ COM แล้วจะเห็นว่ามีการทำงานติด-ดับลักษณะคล้ายการทำงานของสวิตช์ เราสามารถอาศัยคุณสมบัตินี้ไปประยุกต์ใช้งานได้

วิธีการนำ Relay Module ไปประยุกต์ใช้งานจริง เรามาดูวิธีอ่านคุณสมบัติของ Relay ว่าสามารถรองรับการทำงานที่แรงดันและกระแสไฟฟ้าเท่าไร ใช้แรงดันไฟฟ้าในการทำงานอย่างไรแสดงได้ดังรูปที่ 2.6

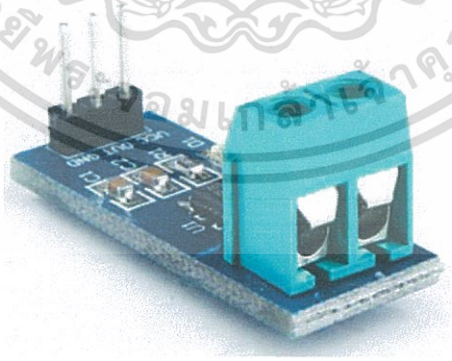


รูปที่ 2.6 คุณสมบัติของ Relay [6]

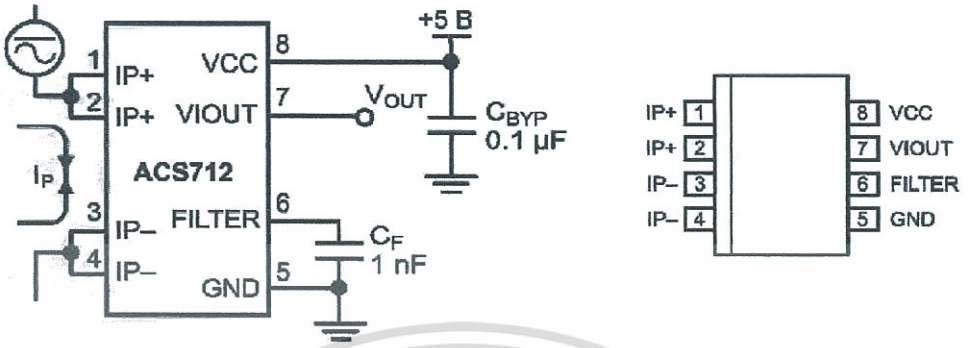
1. ยี่ห้อ รุ่นของผู้ผลิต (แบรนด์) รวมถึงสัญลักษณ์มาตรฐานต่างๆ
2. รายละเอียดของไฟฟ้ากระแสสลับที่รองรับการทำงานได้ (VAC)
3. รายละเอียดของไฟฟ้ากระแสตรงที่รองรับการทำงานได้ (VDC)
4. โมเดล ระดับแรงดันฝั่งขดลวด ชนิดและโครงสร้าง และข้อมูลด้าน Coil Sensitivity

2.3 เซ็นเซอร์วัดกระแส (Current Sensor)

โมดูลนี้เป็นอุปกรณ์ใช้ต่อพ่วงกับ Arduino ในช่อง Analog เพื่ออ่านค่ากระแสที่ไหลผ่าน โดยอาศัยไอซี ACS712-05 Current Sensor ที่วัดกระแสโดยใช้หลักการของ Hall Effect ใช้ได้ทั้งการวัดไฟฟ้ากระแสตรงและไฟฟ้ากระแสสลับแสดงได้ดังรูปที่ 2.7 และการต่อใช้งาน (ซ้าย) และขาต่างๆ (ขวา) ของ ACS712 แสดงได้ดังรูปที่ 2.8



รูปที่ 2.7 เซ็นเซอร์วัดกระแส ACS712 [7]



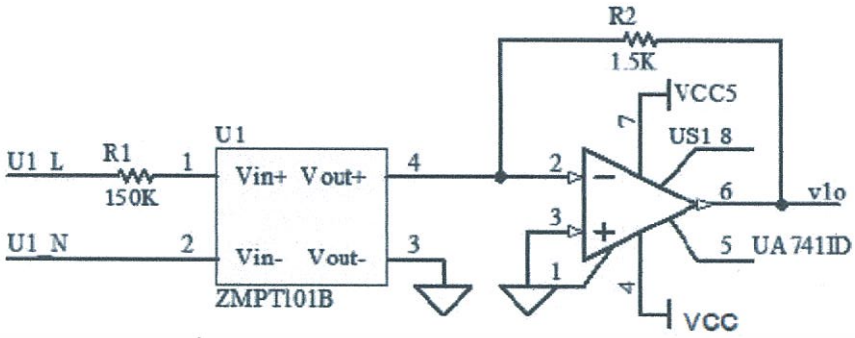
รูปที่ 2.8 การต่อใช้งาน (ซ้าย) และขาต่างๆ (ขวา) ของ ACS712 [8]

2.4 เซ็นเซอร์วัดแรงดันไฟฟ้า (Voltage Sensor)

2.4.1 เซ็นเซอร์วัดแรงดันไฟฟ้ากระแสสลับ

โมดูลเซ็นเซอร์แรงดันไฟฟ้ากระแสสลับ ZMPT101B ใช้ตรวจวัดค่าแรงดันไฟฟ้ากระแสสลับ AC 220V วัดแรงดันไฟ AC สูงสุดที่ 250VAC สัญญาณที่ออกจากโมดูลเป็นสัญญาณอนาล็อก สามารถนำไปต่อเข้ากับขา ADC ของไมโครคอนโทรเลอร์ที่ใช้ Vref +5V ได้ทันทีเช่น Arduino Uno, Mega, Leonardo มีวงจรขยายสัญญาณ สามารถปรับขนาดแอมพลิจูดของสัญญาณเอาท์พุตได้ จากการปรับตัวต้านทานปรับค่าบนบอร์ดแสดงได้ดังรูปที่ 2.9 และการต่อใช้งานและขาต่างๆของ ZMPT101B แสดงได้ดังรูปที่ 2.10

รูปที่ 2.9 เซ็นเซอร์วัดแรงดัน AC [9]



รูปที่ 2.10 การต่อใช้งานและขาต่างๆของ ZMPT101B [10]

2.4.2 เซ็นเซอร์วัดแรงดันไฟฟ้ากระแสตรง

โมดูลวัดแรงดันไฟฟ้า 0-24 โวลต์สำหรับ Arduino Standard Voltage Sensor Module For Arduino โมดูลนี้ใช้หลักการวัดแรงดันไฟฟ้าทางขา Analog ของ Arduino ที่สามารถอ่านค่าแรงดันไฟฟ้าที่ 0-5 โวลต์ออกมาเป็นค่าดิจิตอล 0-1023 โมดูลนี้ใช้วงจรแบ่งแรงดันทำให้สามารถวัดไฟได้สูงสุดถึง 24.9 โวลต์ โดยใช้ไฟเลี้ยงที่ 5 โวลต์ ถ้าบอร์ด arduino ใช้ไฟเลี้ยงที่ 3.3 โวลต์จะวัดไฟได้สูงสุดที่ 16.5 โวลต์แสดงได้ดังรูปที่ 2.11

รูปที่ 2.11 เซ็นเซอร์วัดแรงดัน DC [11]

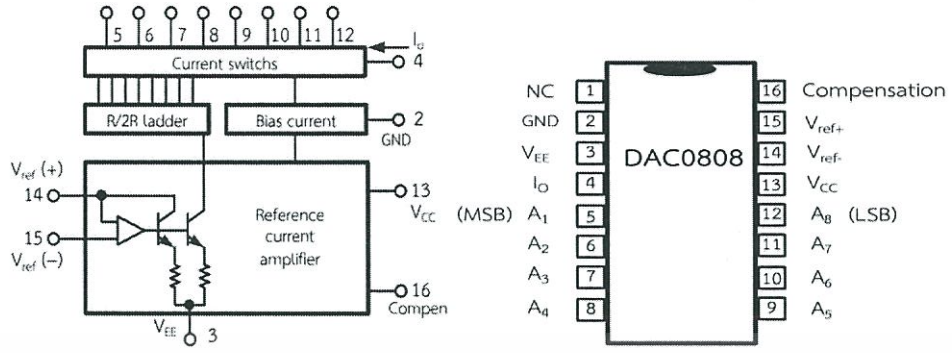
2.5 การแปลงสัญญาณดิจิตอลเป็นอนาล็อก

วงจรดิจิตอลสามารถทำงานร่วมกับอุปกรณ์ภายนอกและระบบปฏิบัติงานแบบอื่นๆ ได้ จะต้องมีการแปลงสัญญาณดิจิตอลที่ได้จากระบบประมวลผลให้เป็นสัญญาณอนาล็อก (D/A converter) โดยนำสัญญาณประมวลผลจากระบบคอมพิวเตอร์หรืออุปกรณ์ดิจิตอลส่งค่าสัญญาณทางเอาท์พุตของระบบให้เป็น ปริมาณการเปลี่ยนแปลงของค่าแบบต่อเนื่องเพื่อให้ระบบสามารถทำงานกับอุปกรณ์ที่เป็นแบบอนาล็อกได้ เป็นลักษณะของอุปกรณ์ในการเชื่อมต่อซึ่งวงจรการแปลงค่าสัญญาณนั้นมีความสำคัญในการทำหน้าที่แปลงสัญญาณดิจิตอลของกลุ่มจำนวน

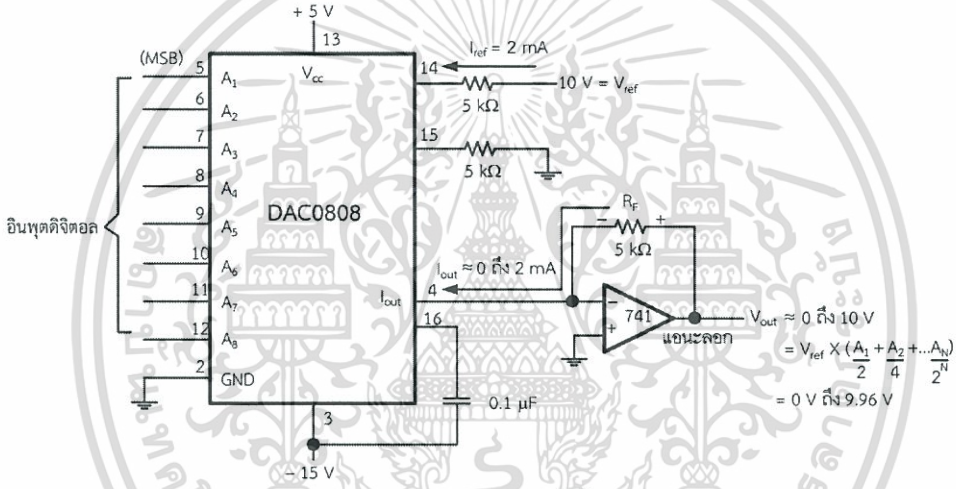
เลขฐานสองให้เป็นสัญญาณแบบอนาล็อก โดยค่าความละเอียดของเอาท์พุตที่ได้จะมีความสัมพันธ์โดยตรงต่อจำนวนบิตทางอินพุต โดยวงจรการแปลงค่าดิจิตอลเป็นอนาล็อกขนาด 4 บิต จะมีค่าเท่ากับ 2^n เมื่อ n คือจำนวนบิตทางอินพุตจะได้ 2^4 ค่าของระดับความแตกต่างของสัญญาณเอาท์พุตอนาล็อกและถ้าเป็นวงจรแปลงสัญญาณดิจิตอลเป็นอนาล็อกขนาด 8 บิต จะได้ค่าของระดับความแตกต่างของสัญญาณเอาท์พุตอนาล็อกที่เท่ากับ 2^8=256 ค่า ซึ่งจะเห็นได้ว่าเมื่อจำนวนบิตอินพุตมากขึ้นก็จะได้ความละเอียดของระดับสัญญาณทางด้านเอาท์พุตมากขึ้นเท่านั้น ในเนื้อหาของหน่วยนี้ จะเป็นการศึกษาเรื่องของหลักการทำงานของอุปกรณ์ที่นำมาสร้างเป็นวงจรแปลงสัญญาณดิจิตอลเป็นอนาล็อก คือ ออปแอมป์ และการใช้งานของวงจรรวมที่ใช้เป็นวงจรแปลงสัญญาณดิจิตอลเป็นอนาล็อก ค่าเอาท์พุตของข้อมูลที่ได้จากการทำงานหรือการประมวลผลจากวงจรดิจิตอล เมื่อต้องการนำมาใช้กับ อุปกรณ์อนาล็อกในรูปแบบต่าง ๆ ที่ไม่ใช่ระบบดิจิตอลหรืออุปกรณ์ที่มีการทำงานและใช้ค่าของสัญญาณแบบต่อเนื่อง ดังนั้นจึงต้องมีการแปลงค่าข้อมูลที่อยู่ในรูปแบบสัญญาณดิจิตอลให้อยู่ในรูปแบบของสัญญาณอนาล็อกเพื่อให้อุปกรณ์สามารถทำงานร่วมกันได้ สำหรับอุปกรณ์ดิจิตอลจะมีวงจรรวมที่ทำหน้าที่ในการแปลงสัญญาณดังกล่าวเรียกว่า วงจรแปลงสัญญาณดิจิตอลเป็นอนาล็อก (D/A Converter) เนื่องจากค่าปริมาณทางกายภาพที่อยู่ในธรรมชาติ เช่น เวลา แรง อุณหภูมิ เสียง ความเร็ว จะเป็นปริมาณทางอนาล็อก เมื่อมีการแปลงรูปแบบของสัญญาณจากอุปกรณ์ที่ทำหน้าที่แปลงปริมาณทางกายภาพเป็นค่าแรงดันไฟฟ้า และมีการแปลงเป็นสัญญาณดิจิตอลเข้าสู่วงจรในระบบดิจิตอลทำการประมวลผลตามความต้องการของงาน เช่น การวัดค่าปริมาณเมื่อได้ผลลัพธ์ซึ่งจะเป็นกลุ่มสัญญาณรหัสตัวเลขฐานสอง และต้องการนำค่าดังกล่าวกลับไปใช้ในทางกายภาพ จึงต้องทำการแปลงสัญญาณกลับไปเป็นปริมาณทางอนาล็อกอีกครั้ง หรือมีการแปลงน้ำหนักของเลขฐานสองผ่านวงจรแปลงสัญญาณให้เป็นแรงดันอนาล็อก โดยการแปลงสัญญาณดิจิตอลเป็นอนาล็อกพื้นฐานสามารถใช้การทำงานของออปแอมป์และการใช้วงจรรวมที่ทำหน้าที่เฉพาะในการทำงานได้

2.5.1 วงจรรวมที่ใช้เป็นวงจรแปลงสัญญาณดิจิตอลเป็นอนาล็อก

วงจรรวมที่นิยมใช้กันมากและราคาไม่แพงสำหรับวงจรแปลงค่า D/A ขนาด 8 บิต (DAC) คือ เบอร์ MC1408 และ DAC0808 ซึ่งมีสมรรถนะของการทำงานสูงถูกออกแบบมาเพื่อใช้สร้างเอาท์พุตกระแสอย่างต่อเนื่องของค่าเวิลด์ดิจิตอลขนาด 8 บิต และการอ้างอิงค่าแรงดันอนาล็อก มีคุณลักษณะการทำงานที่มี ความเร็วสูงในการเซทค่าเวลาทำงานที่ 70 ns และมีค่าความแม่นยำสูงความผิดพลาดที่ $\pm$ 0.19 % สามารถ เชื่อมต่อสัญญาณอินพุตได้ทั้งทีทีแอลและซีมอส อัตราความเร็วในการทำงาน 4.0 mA/μS (อินพุท สลูว์) ค่า แรงดันเอาท์พุตอยู่ในช่วง +0.5 V ถึง -5.0 V ใช้กับแหล่งจ่ายแรงดันมาตรฐานที่ +5.0 V และ -5.0 V ถึง -15.0 V (ข้อมูลจากคู่มือบริษัทฟิลลิปส์) มีเป็นบล็อกไดอะแกรมของ MC1408 ตำแหน่งขาการต่อใช้งานและรูปแบบการต่อประยุกต์ใช้งานแสดงได้ดังรูปที่ 2.12



(ก) บล็อคไดอะแกรม (ข) ตำแหน่งขาการต่อใช้งาน

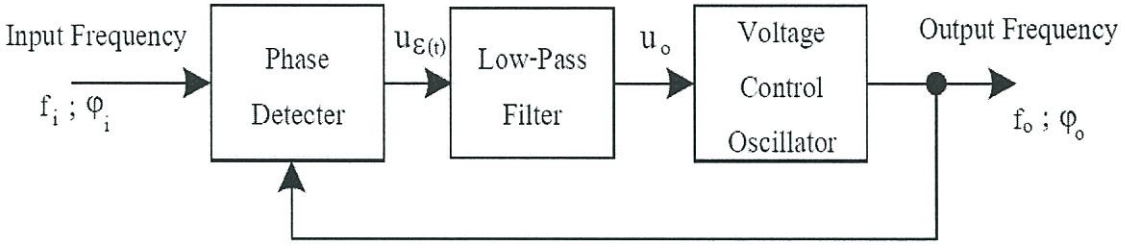


(ค) วงจรการประยุกต์ใช้งาน

รูปที่ 2.12 (ก) บล็อคไดอะแกรมของ DAC 0808 (ข) ตำแหน่งขาการต่อใช้งานและ (ค) รูปแบบการต่อประยุกต์ใช้งาน [12]

2.6 เฟสล็อคลูป (Phase Lock Loop : PLL)

PLL เป็นระบบควบคุมความถี่ โดยใช้วิธีเปรียบเทียบเฟส (Phase) ของความถี่ทางด้านเอาต์พุตกับเฟสของความถี่อ้างอิง (Reference Frequency) ซึ่งถูกป้อนเขาทางด้านอินพุตของระบบ แสดงได้ดังรูปที่ 2.13



รูปที่ 2.13 เป็นแผนผังของ PLL เบื้องต้น

จากแผนผังพบว่า PLL เบื้องต้น จะประกอบด้วย 3 สวนใหญ่ๆ คือ

ภาคตรวจจับเฟสหรือตัวเปรียบเทียบสัญญาณ (Phase Detector หรือ Comparator)

มีหน้าที่เปรียบเทียบความแตกต่างของเฟสระหว่าง φ_i และ φ_o ให้กำเนิดแรงดันคลาดเคลื่อน (Error Voltage) ; u_ε ออกมาทางเอาต์พุต

$$u_\varepsilon(t) = K_\varphi\left(\varphi_i - \varphi_o\right) = K_\varphi \Delta_\varphi \tag{2.1}$$

ภาคกรองความถี่ต่ำผ่านหรือภาคกรองความถี่ลูป (Low – Pass Filter หรือ Loop Filter)

ทำหน้าที่กำจัดส่วนประกอบทางไฟสลับที่ปะปนมากับแรงดันคลาดเคลื่อนและปล่อยให้ส่วนประกอบทางไฟตรงของแรงดันคลาดเคลื่อนผ่านไปยังเอาต์พุต

$$u_o = \overline{u_\varphi(t)} = \frac{1}{T}\int_o^T u_\varepsilon(t)\,dt \tag{2.2}$$

ภาคกำเนิดสัญญาณควบคุมด้วยแรงดัน (Voltage Controlled Oscillator, VCO)

ความถี่ f_o ของ VCO จะเปลี่ยนไปตามแรงดัน; u_o ทางอินพุต ดังนั้นเมื่อ u_o เปลี่ยนไปก็จะมีผลทำให้ f_o และ φ_o เปลี่ยนแปลงตามไปด้วย

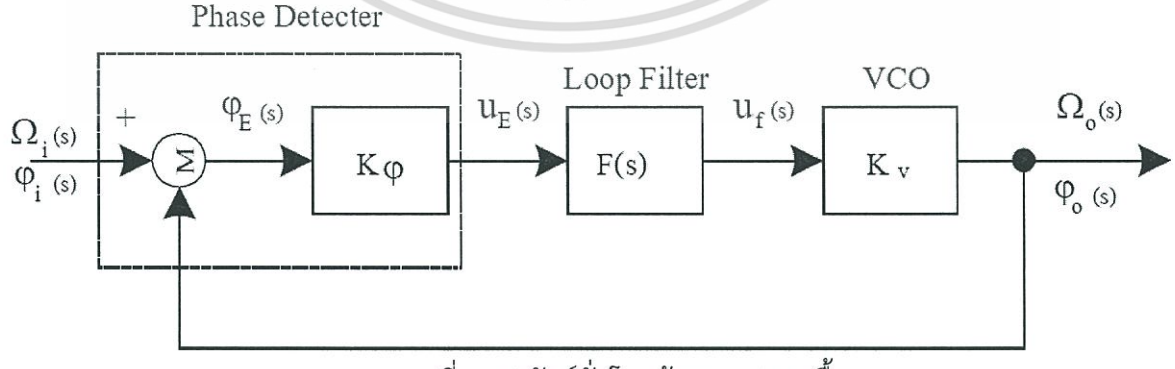
$$f_o = K_f u_o \tag{2.3}$$

ในระบบ PLL ขนาดของแรงดันคลาดเคลื่อนที่ได้จากภาคตรวจจับเฟสจะแปรผันเป็นสัดส่วนโดยตรงกับผลต่างของ $f_i - f_o$ และ $\varphi_i - \varphi_o$ แรงดันคลาดเคลื่อนนี้จะไปควบคุมให้การเปลี่ยนแปลงของ f_o ของ VCO เป็นไปในทิศทางที่ทำให้ผลต่างของความถี่; $f_i - f_o$ มีขนาดลดลง นั่นคือ f_o จะมีค่าเข้าใกล้ f_i มากขึ้น เราเรียกภาวะของลูป (Loop) ในขณะที่ VCO เริ่มเปลี่ยนความถี่ f_o วา "สภาวะแคปเตอร (Capture State)" เมื่อ f_o มีค่าเท่ากับ f_i การเปลี่ยนแปลงของ f_o ก็สิ้นสุดลง เราเรียกภาวะนี้ว่า "เฟสล็อค (Phase Locked)"

โดยทั่วไปแล้วเราสามารถบอกได้ว่า PLL ประกอบด้วย 3 ภาวะด้วยกัน คือ

1. ภาวะทำงานเป็นอิสระ (Free – Running) ; ความถี่ของ VCO ถูกกำหนดจากโครงสร้างวงจร ของ VCO เอง
2. ภาวะแคปเตอร (Capture) ; ความถี่ f_o ของ VCO กำลังวิ่งเข้าหาความถี่อินพุต; f_i
3. ภาวะเฟสล็อค (Phase Locked) ; ความถี่ f_o ของ VCO เท่ากับความถี่อินพุต; f_i

ย่านความถี่ ซึ่งระบบลูปสามารถติดตามการเปลี่ยนแปลงของความถี่อินพุตได้เราเรียกว่า "ย่านล็อค (Lock Range)" ย่านความถี่ ซึ่งระบบลูปสามารถเข้าถึงภาวะเฟสล็อคได เรียกว่า "ย่านแคปเตอร (Capture Range)" ซึ่งจะมีย่านแคบกว่า Lock Range สำหรับลักษณะสมบัติทางไดนามิกส์ (Dynamic Characteristics) ของ PLL จะถูกกำหนดโดยคุณสมบัติของตัวกรองความถี่ลูป ซึ่งเป็นวงจรกรองความถี่ต่ำผ่าน ในขณะที่ PLLอยู่ในภาวะเฟสล็อค ความเร็วในการติดตามการเปลี่ยนแปลงของความถี่อินพุต; f_i ของ PLL จะถูกจำกัดโดยตัวกรองความถี่ลูปโดยมีฟังก์ชั่นโอนย้ายของ PLL พื้นฐานแสดงได้ดังรูปที่ 2.14



รูปที่ 2.14 ฟังก์ชั่นโอนย้ายของ PLL พื้นฐาน

เนื่องจาก ตัวกรองความถี่ลูป เป็นวงจรกรองความถี่ต่ำผ่าน ซึ่งมีความถี่ตัด (Cut off Frequency) ต่ำกว่าส่วนประกอบทางไฟสลับของแรงดันคลาดเคลื่อนจากภาคตรวจจับเฟสมากๆทำให้ส่วนประกอบทางไฟสลับ ไม่สามารถส่งผ่านลูปได้ ดังนั้น $u_E(S)$ ที่ปรากฏในแผนผัง จึงหมายถึง แรงดันคลาดเคลื่อนที่ไม่มีส่วนประกอบทางไฟสลับสำหรับที่ VCO เราสามารถหาความสัมพันธ์ระหว่าง $\varphi_o(S)$ และ $u_f(S)$ ได้ดังนี้

$$\Omega_o(S) = K_V u_f(S) \tag{a}$$

เนื่องจาก $$\Omega_o(S) = S\varphi_o(S) \tag{b}$$

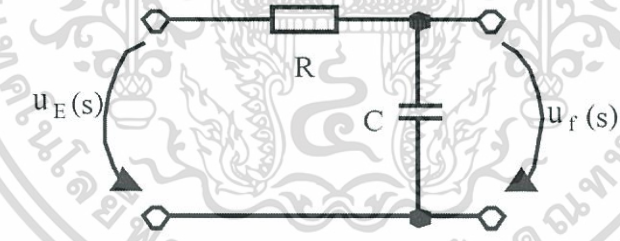
แทนค่า (b) ใน (a) จะได้

$$\varphi_o(S) = \frac{K_V u_f(S)}{S} \tag{2.4}$$

ฟังก์ชั่นโอนย้าย; $T(s)$ ของ PLL ซึ่งแสดงความสัมพันธ์ระหว่าง $\varphi_i(S)$ และ $\varphi_o(S)$ สามารถหาได้จากแผนผังข้างต้น ดังนี้

$$T(S) = \frac{\varphi_o(S)}{\varphi_i(S)} = \frac{K_\varphi K_V F(S)}{S + K_\varphi K_V F(S)} \tag{2.5}$$

PLL ที่มีตัวกรองความถี่ชนิด A แสดงได้ดังรูปที่ 2.15



รูปที่ 2.15 ตัวกรองความถี่ต่ำผ่านแบบพาสซีฟ

จากวงจร ตัวกรองความถี่ต่ำผ่านแบบพาสซีฟมีฟังก์ชั่นโอนย้าย

$$F_A(S) = \frac{u_f(S)}{u_E(S)} = \frac{1}{1+\tau s} \tag{2.6}$$

ค่าเวลาคงที่ (Time Constant) ; τ หาได้จาก

$$\tau = RC \tag{2.7}$$

ความถี่ตัด; ω_A

$$\omega_A = \frac{1}{\tau} = \frac{1}{RC} \tag{2.8}$$

เมื่อแทน $F_A(S)$ จาก(6) ลงใน $F(S)$ จาก (5) จะได้ฟังก์ชั่นโอนย้ายของ PLL ดังนี้

$$T_A(S) = \frac{K_\varphi K_V / \tau}{s^2 + (1/\tau)s + K_\varphi K_V / \tau} \tag{2.9}$$

สมการ (9) เป็นระบบลำดับที่สอง (2^{nd} – Order System) ซึ่งสามารถเขียนในรูปแบบมาตรฐานดังนี้

$$T_A(S) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{2.10}$$

ความถี่ธรรมชาติของระบบ (System Natural Frequency (Swing)) ; ω_n

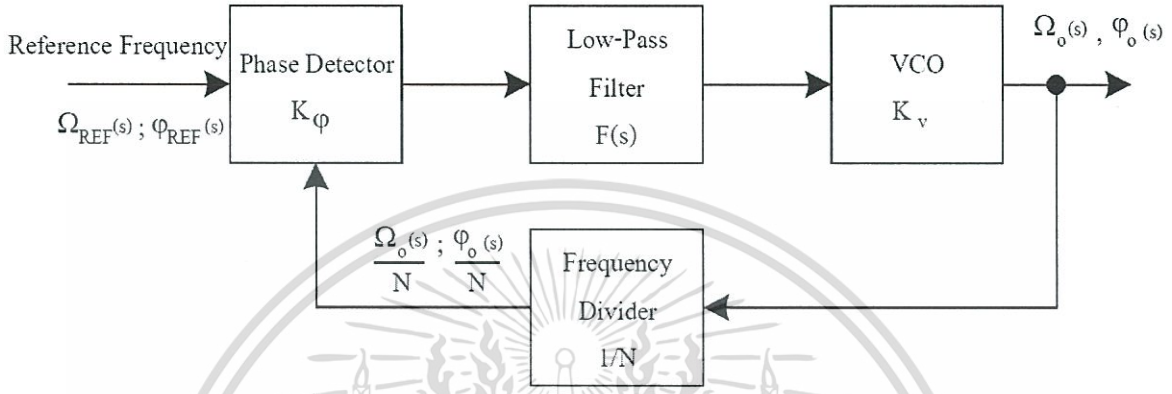
$$\omega_n = \sqrt{\frac{K_\varphi K_V}{T}} = \sqrt{K_\varphi K_V \omega_A} \tag{2.11}$$

ปัจจัยหน่วง (Damping Factor); ζ

$$\zeta = \frac{1}{2}\sqrt{\frac{1}{K_\varphi K_v \tau}} = \frac{1}{2}\sqrt{\frac{\omega_A}{K_\varphi K_v}} \tag{2.12}$$

2.7 วงจรสังเคราะห์ความถี่ (Frequency Synthesizers)

วงจรสังเคราะห์ความถี่เป็นแหล่งจ่ายความถี่ ซึ่งเอาต์พุตสามารถให้ความถี่เป็นทวีคูณ (Multiple) ของความถี่อ้างอิงที่อินพุต (Input Reference Frequency) แสดงได้ดังรูปที่ 2.16



รูปที่ 2.16 วงจรสังเคราะห์ความถี่ โดยใช้ระบบ PLL

การติดตั้งวงจรนับแบบหาร N ในทางเดินป้อนกลับ ทำให้เราได้ความถี่เอาต์พุต; $\Omega_o(S)$ มีขนาดเป็นความถี่ N เท่าของความถี่อ้างอิงที่อินพุต: $\Omega_{REF}(S)$ เมื่อ PLL อยู่ในภาวะ "เฟสล็อค"

สำหรับฟังก์ชั่นโอนย้าย ซึ่งแสดงถึงความสัมพันธ์ระหว่าง $\varphi_o(S)$ และ $\varphi_{REF}(S)$ ของวงจรสังเคราะห์ความถี่ สามารถเขียนได้ดังนี้

$$T(S)=\frac{\varphi_o(S)}{\varphi_{REF}(S)}=\frac{K_V K_\varphi F(S)}{s+K_V K_\varphi F(S)\frac{1}{N}} \tag{2.13}$$

วงจรสังเคราะห์ความถี่ที่ใช้ตัวกรองความถี่ชนิด A

ฟังก์ชั่นวงจรกรองชนิด A

$$F_A(S)=\frac{1}{1+\tau s} \tag{2.14}$$

ปัจจัยหน่วงของระบบ: ζ และความถี่ธรรมชาติ: ω_n หาได้จาก

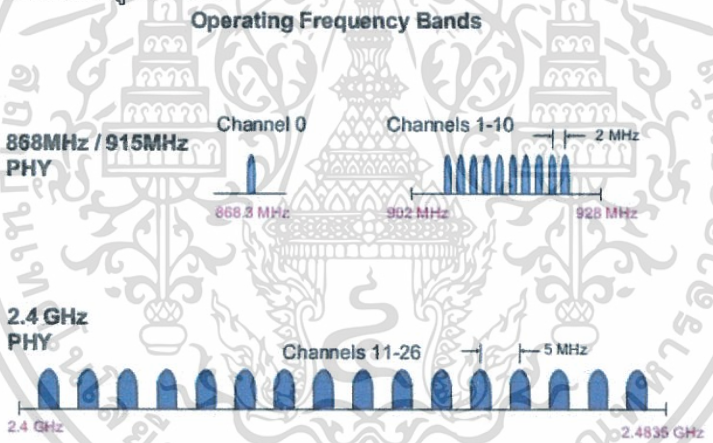
$$\omega_n=\sqrt{\frac{K_V K_\varphi}{N\tau}} \tag{2.15}$$

$$\zeta=\frac{1}{2}\sqrt{\frac{N}{K_V K_\varphi \tau}} \tag{2.16}$$

2.8 หลักการพื้นฐานของเทคโนโลยี Zigbee

เป็นเทคโนโลยีไร้สายที่ร่วมกันสื่อสารข้อมูลผ่านเซ็นเซอร์ขนาดเล็กจิ๋ว โดยชื่อ Zigbee ได้มาจากพฤติกรรมของผึ้งโดยผึ้งจะบินซิกแซ็กและจะให้ข้อมูลข่าวสารระหว่างผึ้งด้วยกันที่เกี่ยวข้องกับ ตำแหน่ง ระยะทาง และทิศทางของอาหารที่พวกกันกำลังหาอยู่ Zigbee ถูกสร้างขึ้นภายใต้มาตรฐาน IEEE 802.15.4 เป็นมาตรฐานที่ใช้สำหรับการสื่อสารความเร็วต่ำ และมีการใช้พลังงานน้อย การทำงานของ Zigbee จะเป็นการรับส่งคลื่นสัญญาณข้อมูลผ่านชิปเล็กจิ๋วจุดต่อจุดไปเรื่อยๆจนถึงปลายทางโดยใช้พลังงานแสงอาทิตย์หรือแบตเตอรี่ขนาดเล็กที่กินไฟน้อยมาก มีทางเข้าช่องสัญญาณหลายช่องทางเพื่อหลีกเลี่ยงการชนกัน ระยะทางทั่วไป 50 เมตรมี Topology แบบ Star, Peer-to-Peer, Mesh

Zigbee กำหนดย่านความถี่ใช้งานตามมาตรฐานไว้ 3 ย่านความถี่คือ ย่าน 2.4 GHz, ย่าน 915 MHz และย่าน 868 MHz โดยแต่ละย่านจะมีช่องสัญญาณ 16 ช่อง, 10 ช่อง และ 1 ช่อง ตามลำดับ ส่วน อัตรารับส่งข้อมูล (ทางอากาศ) จะอยู่ที่ 250 Kbps, 40 Kbps, 20 Kbps ตามลำดับเช่นกันแสดงได้ดังรูปที่ 2.17



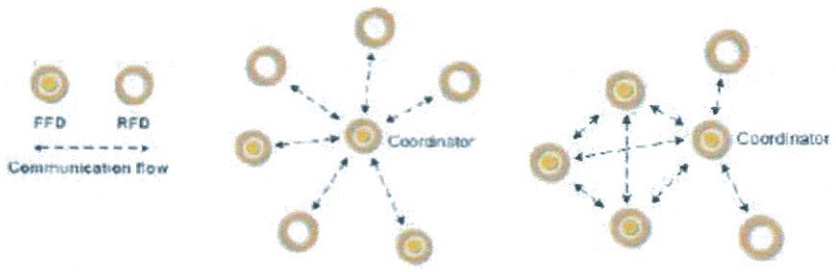
รูปที่ 2.17 ย่านความถี่ของการใช้ Zigbee และ Chanel ต่างๆ [13]

ZigBee นำ Physical Layer และ MAC Layer ของ IEEE 802.15.4 ซึ่งเป็นมาตรฐานการกำหนดการสื่อสารไร้สายแบบ WPAN (Wireless Personal Area Network) มาทำงานใน Layer ที่ต่ำกว่า (2 Layer ล่างสุด) เช่น เรื่องของ ระดับกำลังสัญญาณ, Link Quality, Access control, Security ฯลฯ แต่ใน Layer ถัดไปจะเป็นรูปแบบของ Zigbee

2.8.1 ชนิดอุปกรณ์ของ Zigbee

อุปกรณ์ของซิกบีมี 2 ชนิดคือ

1. Physical Device แบ่งได้ 2 ประเภท แสดงดังรูปที่ 2.18



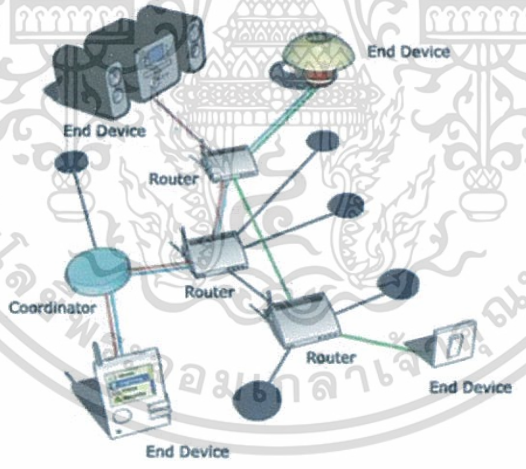
Star Mesh

รูปที่ 2.18 การเชื่อมต่อเครือข่ายแบบ Physical Device [14]

-Full Function Device (FFD) : เป็น Router ที่เป็นสื่อกลางในการส่งข้อมูลจากอุปกรณ์อื่นๆ โดยจะใช้พลังงานจาก Power line สามารถทำงานได้ในทุก Topology และสามารถเป็นจุดเชื่อมต่อได้

-Reduced Function Device (RFD) : จะใช้พลังงานแบตเตอรี่ไม่สามารถถ่ายทอดข้อมูลจากอุปกรณ์อื่นๆได้ ทำได้ง่ายในเครือข่ายแบบ Star

2. Logical Device โดยแบ่งออกเป็น 3 ประเภท แสดงได้ดังรูปที่ 2.19



รูปที่ 2.19 การเชื่อมต่อกันเป็นเครือข่ายแบบ Logical Device [15]

- Zigbee Coordinator เป็นจุดที่ประสานเชื่อมต่อกันทำหน้าที่จัดเก็บข้อมูลในเครือข่าย

-Zigbee Routers ทำหน้าที่จัดเรียงเส้นทางข้อมูลส่งผ่านเครือข่ายระหว่างโหนดใดๆ

-Zigbee End Device เป็นโหนดที่อยู่ในส่วนของผู้ใช้งานโดยเป็นได้ทั้งแบบ FFD หรือ RFD

2.8.2 โครงสร้างของโปรโตคอล Zigbee

โปรโตคอล Zigbee ถูกออกแบบมาเฉพาะในส่วนของ Application layer, Application support layer และ Network layer เท่านั้น แต่ใช้ MAC layer และ Physical layer ตามมาตรฐาน IEEE 802.15.4 รูปแบบเครือข่ายตามมาตรฐาน IEEE 802.15.4 แบ่งออกเป็น 2 แบบ ได้แก่ แบบ star และแบบ peer-to-peer

2.8.2.1 Application layer เป็นชั้นที่มีของ End point อยู่เรียกว่า Application framework โดยมี Zigbee Device object (ZDO) ทำหน้าที่ในการจัดการเข้าถึงและใช้งาน Application layer

2.8.2.2 Application support sub-layer ทำหน้าที่ในการสร้างเฟรมของ Application layer และทำหน้าที่ในการรับส่งข้อมูลรวมถึงการจัดการด้านต่างๆ ที่เกี่ยวข้องกับ Application layer โดยในแต่ละเครือข่ายจะต้องมี FFD 1 ตัวทำหน้าที่เป็นศูนย์กลางของเครือข่าย เรียกว่า PAN Coordinator และ RFD จะเข้าร่วมเครือข่ายกับ PAN Coordinator ประจำเครือข่ายนั้นๆ

2.8.2.3 Network layer ทำหน้าที่ในการ routing ข้อมูลจากต้นทางไปยังปลายทางที่อาจอยู่ในเครือข่ายเดียวกันหรือต่างเครือข่ายกัน

2.8.3 ขั้นตอนการทำงานของโปรโตคอล Zigbee

2.8.3.1 ขั้นตอนการทำงานของ Zigbee Coordinator

Zigbee Coordinator จะเริ่มต้นเครือข่าย โดยการตรวจสอบการใช้ช่องสัญญาณวิทยุภายในรอบๆ ถ้ามีช่องสัญญาณที่ไม่ถูกใช้โดย coordinator ตัวอื่น ก็สามารถเริ่มต้นเครือข่ายได้หลังจากนั้น Coordinator จะทำหน้าที่เป็นศูนย์กลางของเครือข่ายรองรับการเข้าร่วมเครือข่ายของ Zigbee end-device และรองรับการร้องขออื่นๆตามมาตรฐานด้วยเช่นกัน ในโครงงานนี้ coordinator รองรับการเข้าร่วมเครือข่าย การออกจากเครือข่าย และการร้องขอการ binding เท่านั้น

2.8.3.2 ขั้นตอนการทำงานของ Zigbee end-device

Zigbee end-device จะเริ่มต้นการทำงานโดยการร้องขอการเข้าร่วมเครือข่ายไปยัง coordinator ประจำเครือข่ายนั้นๆ โดยการตรวจสอบผ่านช่องสัญญาณต่างๆว่า coordinator ใช้ช่องสัญญาณใดอยู่เมื่อเข้าร่วมเครือข่ายแล้ว end-device จึงสามารถทำการร้องขอคำสั่งอื่นๆ ผ่านทาง coordinator ได้เช่น การส่งข้อความทั่วไป (Message), การร้องขอ Binding (Binding Request) การขอออกจากเครือข่าย

2.8.4 Xbee

Xbee เป็นอุปกรณ์โมดูลไร้สาย Zigbee ในโครงงานนี้ได้เลือกใช้ Xbee รุ่น XBee 2mW Wire Antenna - Series 2 (ZigBee Mesh) ผลิตโดยบริษัท Digi โดย Xbee เป็นโมดูลรับส่งสัญญาณไร้สายย่านความพี่ 2.4 GHz ตามมาตรฐานโปรโตคอล Zigbee/IEEE 802.15 โดยใช้พลังงานต่ำ (3.3v) รับส่งข้อมูลด้วยอัตราเร็ว 250 Kbps สายอากาศแบบ Whip antenna (Wire Ant) รองรับเครือข่ายแบบ mesh Xbee ที่ใช้แสดงดังรูปที่ 2.20

รูปที่ 2.20 Xbee รุ่น Xbee 2mw wire antenna – Series 2 [16]

2.9 มาตรฐาน IEEE 802.15.4

IEEE 802.15.4 เป็นมาตรฐานของระดับชั้น Physical (PHY) และ Media Access Control (MAC) สำหรับอุปกรณ์ไร้สายที่ต้องการความเร็วต่ำ ออกแบบมาเพื่อใช้พลังงานน้อย ราคาไม่สูงมาก การส่งข้อมูลมีความเชื่อถือได้เพราะมีความถูกต้องสูงและสามารถใช้ได้ทั้งในเครือข่ายแบบ Star และ peer to peer มีระยะส่งข้อมูลได้ประมาณ 10-75 เมตร ขึ้นอยู่กับกำลังที่ใช้ในการส่งข้อมูลและสภาพแวดล้อม มีการเข้ารหัสข้อมูลที่ส่งเพื่อความปลอดภัยของข้อมูล

โดยการรับส่งข้อมูลเบื้องต้นในวงจรเครื่องรับส่งวิทยุ (Physical Layer) จะใช้คลื่นวิทยุความถี่ 2.4–2.4835 GHz, Bit rate 250 kbps มี 16 ช่องสัญญาณคือช่อง 11-26 ใช้งานได้ทั่วโลก, 902-928 MHz, Bit rate 40kbps มี 10 ช่องสัญญาณ คือ 1-10 และ 868-870 MHz, Bit rate 20 kbps มี 1 ช่องสัญญาณคือช่อง 0 ใช้งานได้ในพื้นที่ ยุโรป อเมริกาเหนือ ออสเตรเลีย และนิวซีแลนด์

มาตรฐานเครือข่ายแบบไร้สาย IEEE 802.15.4 ถูกนำมาประยุกต์ใช้แบบยูบิควิตัส (Ubiquitous) โดยเป็นการสื่อสารระหว่างอุปกรณ์ กับอุปกรณ์ หรืออุปกรณ์ กับมนุษย์ ที่ผ่านระบบเครือข่ายไร้สาย เช่น เทคโนโลยีซิกบี ซึ่งการนำไปประยุกต์ใช้งานเพื่อเกิดประโยชน์เช่นระบบการควบคุมอัตโนมัติ ที่บ้าน โรงงาน และโกดังเก็บสินค้า , ระบบการติดตามสำหรับ ความปลอดภัย ชีวะอนามัย และสิ่งแวดล้อม, การตรวจหาตำแหน่งที่นำไปใช้ใน การปฏิบัติการทางทหาร การทำงานของนักผจญเพลิง เป็นต้น

2.10 การเขียนโปรแกรมด้วยภาษา c#

ภาษา C# (ซี-ชาร์ป) [17] เป็นภาษาโปรแกรมคอมพิวเตอร์ระดับสูงที่ใช้สาหรับเขียนโปรแกรมคอมพิวเตอร์ ที่ได้รับความนิยมเพิ่มมากขึ้นเรื่อยๆ ในปัจจุบัน ซึ่งภาษา C# ถูกพัฒนามาจากภาษา C++ และภาษาอื่นๆ และมีโครงสร้างแบบเชิงวัตถุ (object-oriented programming) พัฒนาเริ่มแรกโดยบริษัทไมโครซอฟท์เพื่อทำงานบนดอตเน็ตเฟรมเวิร์ก โดยใช้ Visual Studio (วิชวล-สตูดิโอ) เป็นเครื่องมือสำหรับพัฒนาโปรแกรมคอมพิวเตอร์ ซึ่ง Visual Studio เป็นเครื่องมือที่คอยอำนวยความสะดวกในการพัฒนาโปรแกรมคอมพิวเตอร์ ทำให้ผู้เขียนโปรแกรมสามารถพัฒนาโปรแกรมคอมพิวเตอร์ได้ไม่ยากนัก

2.10.1 ข้อดีของภาษา c#

1) เป็นภาษาที่เขียนง่าย ไม่ซับซ้อนและเรียบง่าย เพราะคล้ายภาษา Java ภาษา C และ ภาษา C++

2) เป็นภาษาโปรแกรมคอมพิวเตอร์ยุคใหม่ที่ถูกสร้างขึ้นมา สำหรับการพัฒนาโปรแกรมคอมพิวเตอร์ ภายใต้แนวคิด .NET Framework ซึ่งเป็นแนวคิดที่ได้รับความนิยมสูงที่สุดในปัจจุบัน

3) เป็นภาษาที่ถูกออกแบบมาให้ทางานบน .NET Framework (ดอตเน็ต-เฟรมเวิร์ก) โดย .NET Framework เป็นรูปแบบในการพัฒนาโปรแกรมคอมพิวเตอร์สมัยใหม่ ซึ่งบริษัทไมโครซอฟท์เป็นผู้พัฒนา ซึ่งคุณสมบัติที่สำคัญของ .NET Framework ก็คือ ผู้ใช้งานสามารถใช้งานบนระบบฮาร์ดแวร์ (Hardware) หรือ ระบบปฏิบัติการ (Operating System) ที่แตกต่างกันได้อย่างไม่มีปัญหา เช่น เครื่องพีซีกับเครื่องแมคหรือ ระบบปฏิบัติการวินโดว์กับระบบปฏิบัติการแมคอินทอช เป็นต้น ดังนั้น ผู้เขียนโปรแกรมจึงสามารถเขียนโปรแกรมคอมพิวเตอร์ใหม่ๆ ได้โดยง่าย รวดเร็ว และไม่ต้องติดข้อจำกัดต่างๆ อย่างเช่นการเขียนโปรแกรมคอมพิวเตอร์ในสมัยก่อนอีกต่อไป

4) เป็นภาษาที่แข็งแกร่ง เพราะเป็นภาษาที่ได้มีการแก้ไขข้อบกพร่องบางอย่างของภาษา Java ภาษา C และ ภาษา C++ เหล่านั้น ทำให้ ภาษา C# เป็นภาษาที่มีความสมบูรณ์ตามแบบฉบับของโครงสร้างแบบเชิงวัตถุ (object-oriented programming)

2.10.2 เครื่องมือสำหรับเขียนโปรแกรมภาษา c#

การเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษา C# นั้น จะมีเครื่องมือที่ช่วยคอยอำนวยความสะดวกสบายให้ผู้เขียนโปรแกรมสามารถเขียนโปรแกรมคอมพิวเตอร์ได้อย่างง่ายดาย รวดเร็ว และ ผู้เขียนโปรแกรมสามารถแก้ไขข้อผิดพลาดในการเขียนโปรแกรมได้ง่ายขึ้นอีกด้วย ซึ่งเครื่องมือดังกล่าวก็คือ โปรแกรม Visual Studio นั่นเอง

Visual Studio เป็นซอฟต์แวร์ประเภท IDE (Integrated Development Environment) ซึ่งเป็นการนำแนวความคิดการทางานแบบรวมศูนย์มาใช้ คือ การทำให้วงจรการพัฒนาระบบทั้งหมดทางานได้อย่างสะดวก รวดเร็ว และ ง่ายดาย เริ่มตั้งแต่การวิเคราะห์ ออกแบบ จนถึงการนาไปปรับใช้ให้เหมาะสมกับวัตถุประสงค์ของการเขียนโปรแกรมคอมพิวเตอร์นั้นๆ

2.11 เครือข่ายและการใช้งานเครือข่าย

2.11.1 เน็ตเวิร์ค (Network)

เน็ตเวิร์ค หมายถึงกลุ่มของคอมพิวเตอร์ หรืออุปกรณ์ซึ่งเชื่อมโยงและสื่อสารกันได้ผ่านการเชื่อมโยงรูปแบบใดรูปแบบหนึ่ง โดยที่อุปกรณ์ใดๆ ที่เชื่อมต่ออยู่กับโครงข่ายจะเรียกว่าโหนด (Node) โดยที่โหนดแต่ละโหนดจะเชื่อมต่อกันด้วยลิงค์ (Link) แต่ละโหนดในโครงข่ายนั้นสามารถสื่อสารได้โดยการส่งข้อมูลถึงกันผ่านทางลิงค์เหล่านี้ สำหรับโครงข่ายเองแล้วแบ่งออกได้ดังนี้

- LAN (Local Area Network)

ระบบเครือข่ายระดับท้องถิ่น มีระยะไม่ไกล 10 กิโลเมตร ส่วนใหญ่จะเป็นภายอาคารเดียวกัน หรือต่างอาคารที่มีระยะไม่ไกลมากนัก เช่น เครือข่ายภายในมหาวิทยาลัย

- MAN (Metropolitan Area Network)

ระบบเครือข่ายระดับเมืองเป็นเน็ตเวิร์คที่มีขนาดใหญ่กว่า LAN โดยต้องทำการเชื่อมต่อกับเครือข่ายผ่านผู้ให้บริการ เช่น ภายในจังหวัด ภูมิภาค เป็นต้น

-WAN (Wide Area Network)

ระบบเครือข่ายที่เรียกได้ว่าเป็นเวิลด์ไวด์ (World Wide) เป็นเครือข่ายที่มีขนาดใหญ่ที่สุด เช่นการส่งข้อมูลภายในประเทศหรือข้ามประเทศ หรืออาจจะข้ามทวีปก็ได้

ในการส่งข้อมูลแต่ละโหนดในโครงข่ายสามารถสื่อสารระหว่างกันได้ นอกจากการเชื่อมต่อแต่ละโหนดเข้าด้วยกันทางกายภาพแล้ว ยังมีความจำเป็นที่ต้องกำหนดรูปแบบและขั้นตอนของการส่งสัญญาณ โดยรูปแบบแต่ละขั้นตอนดังกล่าวจะถูกเรียกว่าโปรโตคอล (Protocol) ในปัจจุบันโปรโตคอลที่ได้รับความนิยมเชื่อมต่อกับ LAN ได้แก่ อีเทอร์เน็ท (Ethernet) โดยอีเทอร์เน็ทยังได้รับการพัฒนาต่อไป ทำให้เกิดโปรโตคอลอีกหลากหลายมาตรฐานที่มีข้อแตกต่างกันในด้านของความเร็วในการรับส่งข้อมูล หรือสายสัญญาณที่ใช้

2.11.2 โครงสร้างของโปรโตคอลทีซีพี/ไอพี (TCP/IP)

โปรโตคอลทีซีพี/ไอพี [18] มีการจัดแบ่งกลไกการทำงานออกเป็นชั้นๆ เรียกว่า เลเยอร์ เหมือนกับมาตรฐาน โอเอสไอโมเดล (OSI Model: Open System Interconnection Model) โดยสามารถเทียบเคียงกับมาตรฐานของโอเอสไอได้ ซึ่งในแต่ละเลเยอร์ของโปรโตคอล TCP/IP จะประกอบด้วยชั้นต่างๆดังนี้

- Process Layer or Application Layer
- Host-to-Host Layer or Transport Layer

- Internetwork Layer

เมื่อเทียบกับมาตรฐานโอเอสไอ จะเห็นว่าบางเลเยอร์ของโปรโตคอลทีซีพี/ไอพี เทียบได้กับมาตรฐานโอเอสไอถึง 2 เลเยอร์ และบางเลเยอร์ก็จะทำงานคาบเกี่ยวกับมาตรฐานโอเอสไอ

ในแต่ละกลไกของโปรโตคอลทีซีพี/ไอพี จะมีโปรโตคอลอื่นๆในชุดของทีซีพี/ไอพีร่วมทำงานอยู่ด้วย จึงทำให้เป็นที่มาของชื่อเรียก Protocol Stack เนื่องจากมีโปรโตคอลซ้อนทับกันอยู่เพื่อช่วยทำงาน จะเห็นได้ว่ามีโปรโตคอลในแต่ละระดับซ้อนทับกันอยู่หลายตัวด้วยกัน การซ้อนกันเป็นชั้นๆหรือแต่ละเลเยอร์นี้ หากเป็นมาตรฐานโอเอสไอ จะมีข้อบังคับแต่ละชั้นติดต่อกันได้เฉพาะชั้นที่ติดกับตนเองเท่านั้น แต่สำหรับแสต็ก ทีซีพี/ไอพี แล้วจะเห็นว่าบางชั้นสามารถละเลยหรือข้ามไปติดต่อกับชั้นอื่นที่ไม่ติดกับตนได้

2.11.2.1 โพรเซสเลเยอร์หรือแอพพลิเคชั่นเลเยอร์

โปรโตคอลทีซีพี/ไอพี เทียบกับมาตรฐานโอเอสไอนั้น ในชั้นบนสุดเรียกว่า โพรเซสเลเยอร์ (Process Layer) ทำงาน 2 หน้าที่เทียบได้กับแอพพลิเคชั่นเลเยอร์(Application Layer) และ พรีเซนต์เตชั่นเลเยอร์ (Presentation Layer) ในชั้นนี้จะรองรับการทำงานของแอพพลิเคชั่นต่างๆที่ทำงานเป็นโพรเซสอยู่ในเครื่องเซิร์ฟเวอร์ (Server) ที่ให้บริการและเครื่องที่ขอใช้บริการ หรือไคลเอนต์ ซึ่งจะติดต่อกันผ่านโปรโตคอลเฉพาะแอพพลิเคชั่นอีกทีหนึ่ง

การทำงานของแอพพลิเคชั่นต่างๆจะอยู่ที่โพรเซสเลเยอร์นี้ และมีการติดต่อกันตามแต่ละโปรโตคอลเฉพาะแล้วแต่แอพพลิเคชั่นที่ใช้งาน จากการที่โพรเซสเลเยอร์ของทีซีพี/ไอพีรองรับให้โปรโตคอลอื่นทำงานได้หลายโพรเซสและหลายโปรโตคอลพร้อมกัน ทำให้ผู้ใช้สามารถเปิดโปรแกรมใช้งานได้หลายๆอย่างพร้อมกัน เช่น เปิดโปรแกรมเว็บบราวเซอร์เพื่อเรียกดูเว็บเพจพร้อมกับใช้งานโปรแกรมรับส่งอีเมลล์ไปพร้อมกัน โดยไม่ต้องรอให้ทำงานอย่างใดอย่างหนึ่งเสร็จก่อน

โปรโตคอลหลักๆที่ทำงานในโพรเซสเลเยอร์ซึ่งผู้ใช้มักคุ้นเคยกันดี ได้แก่ เอฟทีพี (FTP : File Transfer Protocol), เทลเนท (Telnet), เอชทีทีพี (HTTP : Hypertext Transfer Protocol) และเอสเอ็มทีพี (SMTP : Simple Mail Transfer Protocol) นอกจากนี้ยังมีโปรโตคอลอื่นที่อยู่เบื้องหลัง ซึ่งการทำงานโดยที่ผู้ใช้ไม่สามารถมองเห็นได้จากโปรแกรมหรือไม่ได้มีการใช้งานโดยตรง เช่น

- โปรโตคอลดีเอ็นเอส (DNS:Domain Name System) ที่ทำหน้าที่แปลงข้อมูลชื่อโดเมน หรือชื่อเว็บไซท์ให้เป็นหมายเลข IP (IP Address)

- โปรโตคอลเอสเอ็นเอ็มพี (SNMP : Simple Network Management Protocol) ใช้ในการควบคุมและตรวจสอบอุปกรณ์ที่อยู่ในเครือข่าย

- โปรโตคอลดีเอชซีพี (DHCP : Dynamic Host Configuration Protocol) ทำหน้าที่แจกจ่ายข้อมูลพารามิเตอร์ของเครือข่ายให้กับเครื่องลูกข่ายที่เชื่อมต่ออยู่

2.11.2.2 โฮสทูโฮสหรือทรานสปอร์ตเลเยอร์

การทำงานที่ชั้นนี้จะมีบทบาทในการจัดการต่อจากแอพพลิเคชั่นเลเยอร์ บางครั้งเราจะเรียกโฮสทูโฮส (Host-to-Host) ว่าเป็น ทรานสปอร์ตเลเยอร์ (Transport Layer) ซึ่งไม่ใช่ชั้นของทรานสปอร์ตเลเยอร์ในโอเอสไอโมเดล การทำงานของโฮสทูโฮสนี้จะทำการสร้างการเชื่อมต่อระหว่างแอพพลิเคชั่นกับโฮสทูโฮส โดยจุดเชื่อมกันเพื่อรับส่งข้อมูลเรียกว่าพอร์ต (Port) หรือ ซ็อกเก็ต (Socket) และในแต่ละแอพพลิเคชั่นก็จะสร้างการเชื่อมต่อผ่านพอร์ต ได้พร้อมกันหลายๆ แอพพลิเคชั่น ซึ่งการใช้งานพอร์ตของแต่ละแอพพลิเคชั่นที่อยู่ชั้นโพรเซสเลเยอร์ จะแตกต่างกันตามหมายเลขที่กำหนดไว้ และแต่ละโปรโตคอลจะมีการใช้งานพอร์ตหมายเลขต่างๆไม่ซ้ำกัน

เมื่อแอพพลิเคชั่นทำงานผ่านโปรโตคอลในชั้นโพรเซสเลเยอร์ จะมีการส่งผ่านข้อมูลไปยังโฮสทูโฮสเลเยอร์ที่ชั้นนี้จะมีการเชื่อมต่อผ่านพอร์ตที่กำหนด ทำให้การรับส่งข้อมูลในแต่ละโปรโตคอลทำได้ถูกต้อง ถึงแม้ว่าในเครื่องเซิร์ฟเวอร์ที่ให้บริการจะมีการทำงานอยู่หลายโพรเซสที่แตกต่างกันก็ตาม หรือมีผู้เข้ามาใช้บริการเป็นจำนวนมาก และหลายแอพพลิเคชั่นในเวลาเดียวกัน ในชั้นโฮสทูโฮสนี้จะมีโปรโตคอลทำงาน 2 โปรโตคอลที่แตกต่างกันไป คือ โปรโตคอลทีซีพี/ไอพี และโปรโตคอลยูดีพี (UDP) ในการส่งผ่านข้อมูลไปชั้นถัดๆไป จะเห็นว่าโปรโตคอลทีซีพี/ไอพี และโปรโตคอลยูดีพี จะถูกผนึกเข้าไปในโปรโตคอลไอพีอีกทีหนึ่งแล้วส่งไปยังเครือข่ายอินเทอร์เน็ตต่อไป

ตัวโปรโตคอลทีซีพี/ไอพี และโปรโตคอลยูดีพี จะมีแอพพลิเคชั่นเฉพาะเพื่อเรียกใช้งานแยกกัน คือ แอพพลิเคชั่นเรียกใช้โปรโตคอล เอฟดีพี (FTP), เทลเนท, เอชทีทีพี (HTTP) และเอสเอ็มทีพี (SMTP) จะมีการส่งผ่านข้อมูลโดยเรียกใช้โปรโตคอลทีซีพี (TCP) ส่วนแอพพลิเคชั่นที่ใช้โปรโตคอลเอสเอ็นเอ็มพี (SNMP) และดีเอชซีพี (DHCP) จะส่งผ่านข้อมูลโดยเรียกใช้โปรโตคอลยูดีพี (UDP) และสำหรับโปรโตคอลดีเอ็นเอส (DNS) นั้นจะสามารถเรียกใช้โปรโตคอลได้ทั้งทีซีพี (TCP)

และยูดีพี (UDP) ซึ่งเหตุผลที่มีการเรียกใช้โปรโตคอล ทีซีพี และยูดีพี แตกต่างกัน เนื่องจากวิธีการทำงานของทั้งสองโปรโตคอลต่างกันนั่นเอง

2.11.3 โปรโตคอลทีซีพี (TCP)

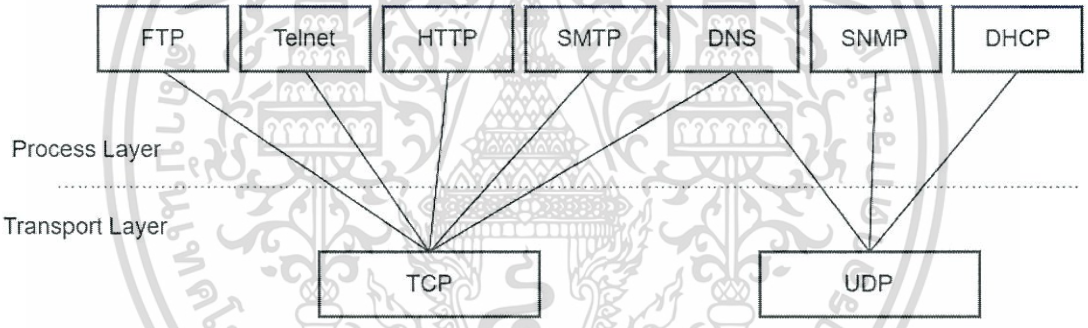
โปรโตคอล TCP (Transmission Control Protocol) เป็นโปรโตคอลที่มีการรับส่งข้อมูลแบบ Stream Orient protocol หมายความว่า การรับส่งข้อมูลจะไม่คำนึงถึงปริมาณข้อมูลที่จะส่งไปแต่จะแบ่งข้อมูลออกเป็นส่วนย่อยๆก่อน แล้วจึงส่งออกไปยังปลายทางอย่างต่อเนื่องเป็นลำดับข้อมูล ในกรณีที่ข้อมูลส่วนใดส่วนหนึ่งสูญหายไป ก็จะส่งข้อมูลส่วนนั้นใหม่อีกครั้งสำหรับปลายทางก็จะทำหน้าที่จัดเรียงข้อมูลออกเป็น datagram ใหม่ให้ต่อเนื่องกันและประกอบกลับเป็นข้อมูลทั้งหมดได้ ซึ่งจะแยกข้อมูลที่ไม่ถูกต้องออก ดังนั้นแอพพลิเคชั่นหรือโพรเซสใดที่อาศัยการส่งข้อมูลด้วยโปรโตคอลTCP จะต้องใช้หน่วยความจำและขนาดของสัญญาณ (Bandwidth) มากกว่า UDP

การติดต่อกันระหว่างกันจะต้องเป็นแบบ Connection-Oriented คือต้องมีการสร้างเส้นทางติดต่อกันเป็น Session ทั้ง 2 ด้านเสียก่อนแล้วจึงจะรับส่งข้อมูลไปได้พร้อมกัน (Full Duplex) เหมือนกับการใช้โทรศัพท์ติดต่อกัน เมื่อผู้ติดต่อต้นทางเรียกให้ฝ่ายตรงข้ามรับสายแล้วจึงเริ่มการสนทนา เช่น คำพูดว่า “สวัสดี” หรือ “Hello” ก่อนเพื่อให้แน่ใจว่าฝ่ายตรงข้ามพร้อมจะติดต่อด้วย จากนั้นจึงเริ่มต้นติดต่อกัน และเมื่อต้องการจะยกเลิกการติดต่อจะมีการพูดว่า “สวัสดี” ให้ฝ่ายตรงข้ามทราบว่าจะเลิกการติดต่อและวางสายไป ซึ่งในระหว่างการติดต่อกันนั้นแม่ว่าฝ่ายใดฝ่ายหนึ่งหรือทั้งสองฝ่ายจะมีช่วงที่เงียบไป คือไม่พูดเป็นเวลานานๆแต่การเชื่อมต่อของทั้งสองก็ยังคงอยู่เช่นเดิมจนกว่าจะมีการยกเลิกการติดต่อระหว่างกัน เช่นเดียวกับกลไกโปรโตคอล TCP เมื่อแอพพลิเคชั่นต้องการส่งข้อมูลผ่านจะใช้โปรโตคอลที่เหมาะสมในชั้น Process Layer ติดต่อไปและมีการสร้างช่องส่งข้อมูลผ่านพอร์ตที่กำหนดเพื่อส่งผ่านข้อมูลไปยังโปรโตคอล TCP

ในระหว่างการรับส่งข้อมูล TCP จะเพิ่มขบวนการสอบทานข้อมูลเพื่อให้ข้อมูลมีความถูกต้องไม่ผิดพลาดไปจากเดิม โดยการส่งสัญญาณสอบทานข้อมูล (Acknowledgement) และส่งข้อมูลให้ใหม่อีกครั้ง ถ้าปลายทางไม่ได้รับหรือเกิดความเสียหายขึ้น

2.11.4 โปรโตคอล UDP

ใน Host-to-Host layer นอกจากจะมีโปรโตคอล TCP แล้วยังมีโปรโตคอลที่ชื่อว่า UDP (User Datagram Protocol) ด้วย ซึ่งมีคุณสมบัติแตกต่างกันอยู่ด้วย ในการรับส่งข้อมูลผ่านโปรโตคอล UDP จะเป็นแบบทั้ง 2 ด้านไม่จำเป็นต้องอาศัยการเชื่อมต่อกันก่อน (Connectionless) ระหว่างเครื่องเซิร์ฟเวอร์ให้บริการกับเครื่องที่ขอใช้บริการ โดยไม่ต้องแจ้งให้ฝ่ายรับเตรียมรับข้อมูลเหมือนกับ TCP และไม่มีการตรวจสอบความถูกต้องครบถ้วนในการรับส่งข้อมูลนั้นๆด้วย เนื่องจากโปรโตคอล UDP ไม่มีสัญญาณสอบทานข้อมูล (Acknowledgement) ในการส่งข้อมูลแต่ละครั้งและไม่มีการส่งข้อมูลใหม่อีกในกรณีที่เกิดความผิดพลาดของข้อมูล เมื่อเป็นเช่นนี้ แอพพลิเคชั่นหรือโพรเซสใดที่ต้องอาศัยโปรโตคอล UDP ในการส่งผ่านข้อมูลก็อาจต้องสร้างขบวนการตรวจสอบข้อมูลขึ้นมาเองมีโปรโตคอล TCP และ UDP แสดงได้ดังรูปที่ 2.21



รูปที่ 2.21 โปรโตคอล TCP และ UDP

จากรูปที่ 2.21 จะเห็นว่าโปรโตคอลชั้นบนขึ้นไป ที่ใช้การส่งผ่านข้อมูลโดยโปรโตคอล UDP เช่น โปรโตคอล SNMP (ใช้ควบคุมและจัดการอุปกรณ์ในเครือข่าย), หรือโปรโตคอล DHCP (ใช้ส่งข้อมูลพารามิเตอร์ของเครือข่ายให้กับลูกข่ายใช้งาน) การส่งข้อมูลเหล่านั้นไม่ต้องรับทราบหรือตรวจสอบว่าข้อมูลไปถึงปลายทางหรือไม่ แต่กลไกตรวจสอบข้อมูลที่มีการรับส่งจะไปทำในขั้นตอนของโปรโตคอลชั้นที่สูงกว่าแทน

อินเตอร์เน็ทเวิร์คเลเยอร์ (Internetwork Layer)

เลเยอร์ชั้น Internetwork Layer ได้แก่ส่วนของโปรโตคอล IP ซึ่งทำหน้าที่คล้ายกับชั้นของ Network Layer ของ OSI – Model คือเชื่อมต่อคอมพิวเตอร์เข้ากับระบบเครือข่ายที่อยู่ชั้นล่างลงไป และทำหน้าที่เลือกเส้นทางการรับส่งข้อมูล ผ่านอุปกรณ์เครือข่ายต่างๆ จนไปถึงผู้รับข้อมูล ในชั้นนี้จะจัดการกับกลุ่มข้อมูลเป็นลักษณะ Frame ในรูปแบบของ TCP/IP

Internetwork Layer มีหน้าที่ส่งผ่านข้อมูลในระหว่างเครือข่าย โดยมีโปรโตคอลที่ทำงานเป็นกลไกสำคัญในการส่งผ่านข้อมูลไปยังเครือข่ายใดๆบนอินเทอร์เน็ตนั่นคือ โปรโตคอล IP นอกจากนี้ในชั้น Internetwork Layer ยังมีการทำงานด้วยอีก 2 ชนิดคือ โปรโตคอล Internetwork Control Message Protocol (ICMP) และโปรโตคอล Address Resolution Protocol (ARP)

2.11.5 HTTP

HTTP ย่อมาจาก Hypertext Transfer Protocol เป็นโปรโตคอลสื่อสารที่ทำงานบนระบบโปรโตคอล TCP HTTP ใช้ระบบเครือข่ายใยแมงมุม (World Wide Web) ทำหน้าที่ในการจำหน่าย, แจกจ่ายรวมไปถึงการรับข้อมูล จากระบบสื่อกลางชั้นสูง (Hypermedia System) ที่ประกอบด้วยเครื่องให้บริการ (Server) ที่มีให้บริการมากมายทั่วโลก

2.11.6 PHP

PHP [19] ย่อมาจากคำว่า "Personal Home Page Tool" เป็น Server Side Script ที่มีการทำงานที่ฝั่งของเครื่องคอมพิวเตอร์ Server ซึ่งรูปแบบในการเขียนคำสั่งการทำงานนั้นจะมีลักษณะคล้ายกับภาษา Perl หรือภาษา C และสามารถที่จะใช้ร่วมกับภาษา HTML ได้อย่างมีประสิทธิภาพ ซึ่งจะทำให้รูปแบบเว็บเพจมีความสามารถเพิ่มขึ้นในด้านของการเขียนโปรแกรม ในการสร้างเว็บจะใช้ Script อยู่ 2 แบบด้วยกันคือ

- Server-Side Script เป็นลักษณะของภาษาที่ทำงานบนเครื่องServer

- Client-Side Script เป็นลักษณะของภาษาที่ทำงานบนเครื่องผู้ใช้

ความสามารถของ PHP นั้น สามารถที่จะทำงานเกี่ยวกับ Dynamic Web ได้ทุกรูปแบบ เหมือนกับ CGI หรือ ASP ไม่ว่าจะเป็นการดูแลจัดการระบบฐานข้อมูล ระบบรักษาความปลอดภัยของเว็บเพจ การรับ - ส่ง Cookies เป็นต้น

2.11.7 Server

เซิร์ฟเวอร์ (Server) คือเครื่องคอมพิวเตอร์หรือระบบปฏิบัติการหรือโปรแกรมคอมพิวเตอร์ซึ่งทำหน้าที่ให้บริการอย่างใดอย่างหนึ่งหรือหลายๆอย่าง แก่คอมพิวเตอร์หรือโปรแกรมคอมพิวเตอร์ที่เป็นลูกข่าย

ระบบปฏิบัติการคอมพิวเตอร์ที่ทำหน้าที่ให้บริการอะไรบางอย่างแก่คอมพิวเตอร์หรือโปรแกรมคอมพิวเตอร์อื่น

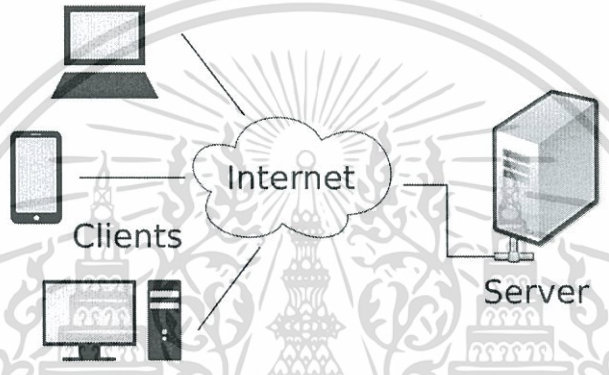
โปรแกรมคอมพิวเตอร์ที่ทำหน้าที่ให้บริการแก่คอมพิวเตอร์หรือโปรแกรมคอมพิวเตอร์อื่นโดยปกติแล้ว โปรแกรมคอมพิวเตอร์ที่เป็น Server จะทำงานบนระบบปฏิบัติการอาจจะเป็น Linux หรือ Windows หรือ Unix ก็ได้ ดังนั้นคำว่า Server จึงมิได้หมายถึงคอมพิวเตอร์ เพียงอย่างเดียวแต่ยังหมายถึงระบบปฏิบัติการคอมพิวเตอร์ หรือโปรแกรมคอมพิวเตอร์อีกด้วย

ตัวอย่างโปรแกรมคอมพิวเตอร์ที่ทำหน้าที่เป็นเซิร์ฟเวอร์ โดยยกตัวอย่างเป็นกลุ่มๆดังนี้

- Web Server คือโปรแกรมที่ให้บริการเว็บ เช่น Apache Web Server
- Mail Server คือโปรแกรมที่ให้บริการ E-mail เช่น Postfix, Gmail
- DNS Server คือโปรแกรมที่ทำหน้าที่ให้บริการโดเมนเนมอาทิเช่น bind9
- Database server คือโปรแกรมที่ทำหน้าที่ให้บริการฐานข้อมูล เช่น MySQL

สำหรับระบบปฏิบัติการที่นิยมใช้เป็น Server ได้แก่

- Linux สำหรับ Linux Distribution ที่ได้รับความนิยมได้แก่ Debian, Ubuntu
- Windows สำหรับ Windows ที่นิยมใช้เป็น Server ได้แก่ Windows Server
- Unix สำหรับ Unix ถือเป็นระบบปฏิบัติการที่เก่าระบบหนึ่งแต่ยังใช้งานอยู่จนถึงทุกวันนี้ได้แก่ BSD



รูปที่ 2.22 การเชื่อมต่อระหว่าง Server และ Clients [20]

บทที่ 3

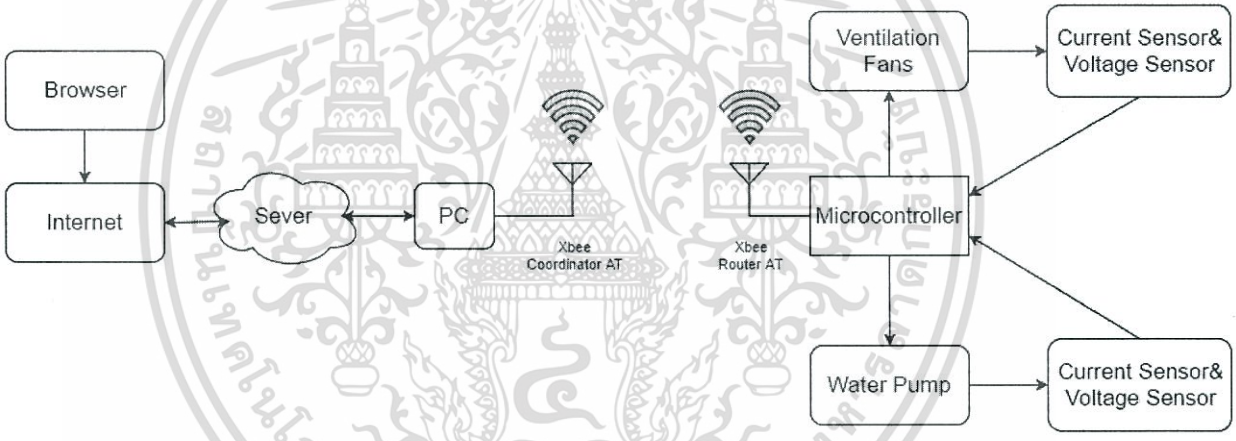
การออกแบบและการจัดทำโครงงาน

โครงงานนี้เป็นการออกแบบและพัฒนาระบบควบคุมมอเตอร์ไฟฟ้าในโรงเรือน เพื่อสังเกตการณ์การทำงานของมอเตอร์โดยจะมีเซ็นเซอร์วัดกระแสและแรงดันเพื่อบันทึกค่าข้อมูลที่อ่านได้จากเซ็นเซอร์ลงในระบบฐานข้อมูลผ่านระบบการสื่อสารแบบไร้สายด้วยซิกบี โดยผู้ใช้สามารถสั่งการและควบคุมการทำงานของระบบได้จากโปรแกรม GUI ที่ออกแบบด้วยภาษา C# รวมทั้งเว็บเบราว์เซอร์

3.1 การออกแบบ

3.1.1 การออกแบบระบบรวม

Block diagram ของระบบ แสดงดังรูปที่ 3.1



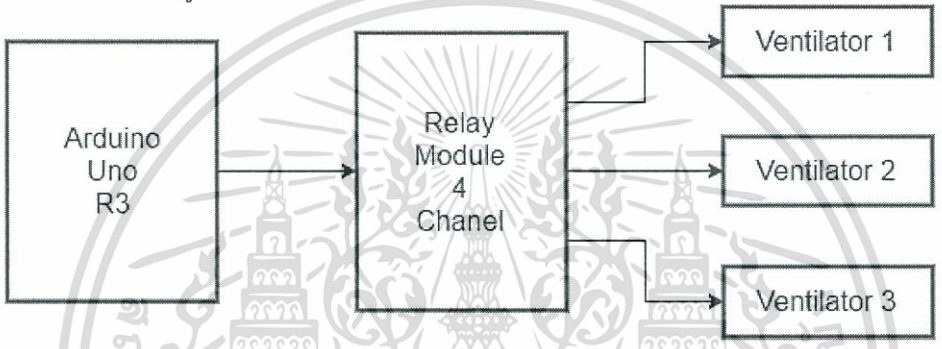
รูปที่ 3.1 แผนภาพการทำงานโดยรวมของระบบ

หลักการทำงานของระบบเริ่มจากฝั่งเซิร์ฟเวอร์ (PC) ที่เชื่อมต่อกับ Xbee ที่ตั้งค่าเป็น Coordinator AT ส่งค่าจากโปรแกรม GUI ที่เขียนด้วยภาษา C# จากโปรแกรม MS Visual Studio โดยจะส่งค่าเป็นตัวเลขต่างๆเช่น 0,1,2 ... เป็นต้น และที่ฝั่งไมโครคอนโทรลเลอร์ที่เชื่อมต่อกับ Xbee Router AT ด้วย Xbee Shield จะรับค่าจากที่ส่งเข้ามาแล้วทำการเช็คเงื่อนไขตามที่ได้ออกแบบไว้ว่าส่งค่านี้มาจะตอบสนองอย่างไร สั่งให้ตัวไหนเปิดหรือปิด เมื่ออุปกรณ์ไฟฟ้าทำงานจะทำการส่งค่าพารามิเตอร์เช่น กระแส แรงดัน กลับมายัง GUI และเก็บเข้าเซิร์ฟเวอร์ ผู้ใช้สามารถที่จะกำหนดการแจ้งเตือนได้จาก GUI นอกจากการสั่งงานผ่าน GUI ที่อยู่บน PC แล้วผู้ใช้สามารถสั่งควบคุมระบบอุปกรณ์ได้จากแอปพลิเคชั่นหรือเว็บที่เชื่อมต่อกับเซิร์ฟเวอร์ด้วย

3.1.2 การออกแบบและจัดทำในส่วนของฮาร์ดแวร์

3.1.2.1 ส่วนของพัดลมระบายอากาศ

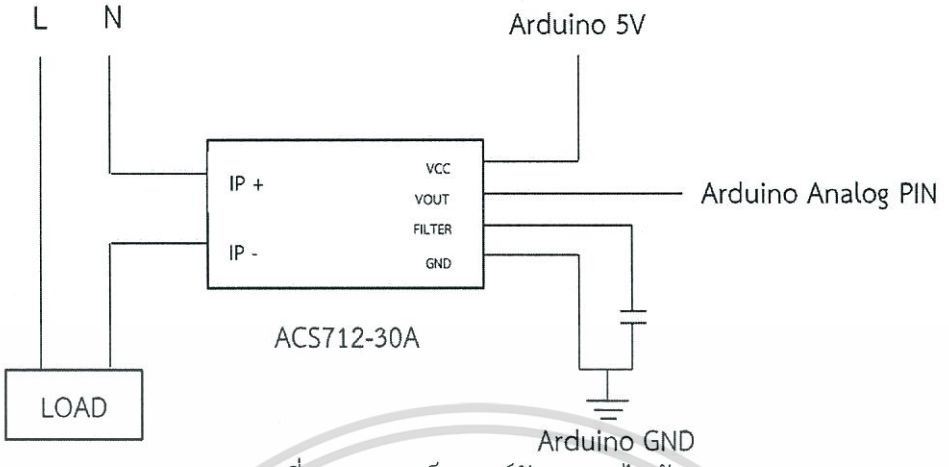
ในส่วนของพัดลมระบายอากาศนั้นจะเชื่อมต่ออยู่กับรีเลย์โมดูล 4 แชนแนล โดยจะใช้พัดลม 3 ตัวเชื่อมต่อกับรีเลย์โมดูลที่ทำหน้าที่เป็นสวิตช์ทางไฟฟ้าถูกควบคุมโดยไมโครคอนโทรลเลอร์ Arduino ที่ทำหน้าที่ในการควบคุมการเปิดและปิดของอุปกรณ์ไฟฟ้าหรือก็คือพัดลมทั้ง 3 ตัว และนอกจากนี้พัดลมยังต้องเชื่อมต่อกับเซ็นเซอร์วัดกระแสและแรงดันที่เชื่อมต่อกับไมโครคอนโทรลเลอร์ที่ทำหน้าที่ในการวัดค่ากระแส แรงดัน และส่งผ่าน Xbee กลับไปเข้าสู่ระบบฐานข้อมูล โดยเซ็นเซอร์ก็จะเชื่อมต่อกับ ไมโครคอนโทรลเลอร์อีกหนึ่งตัวเพื่อแยกการทำงานออกจากกัน ดังแสดงในรูปที่ 3.2



รูปที่ 3.2 การต่อวงจรเพื่อควบคุมการสั่งงานเปิด-ปิดพัดลมระบายอากาศ

3.1.2.2 วงจรเซ็นเซอร์วัดกระแสไฟฟ้า AC

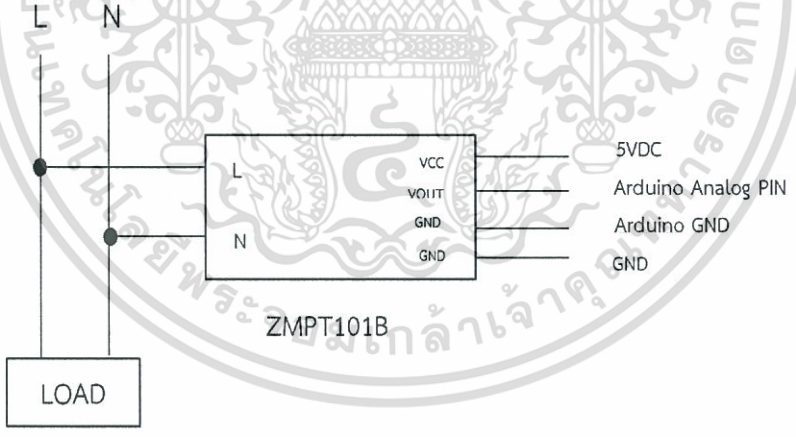
ในการวัดสัญญาณกระแสไฟฟ้านั้นเราจำเป็นจะต้องเปลี่ยนจากกระแสไฟฟ้าที่เกิดขึ้นจริงไปเป็นสัญญาณที่ไมโครคอนโทรลเลอร์สามารถจัดการได้คือสัญญาณของแรงดันไฟฟ้าแอมพลิจูดต่ำ เราจึงเลือกใช้โมดูลสำเร็จรูป ACS 712 ซึ่งใช้หลักการของ Hall Effect ในการเปลี่ยนกระแสไฟฟ้าไปเป็นสัญญาณแรงดันไฟฟ้าแอมพลิจูดต่ำเพื่อป้อนเข้าสู่ไมโครคอนโทรลเลอร์ต่อไป เนื่องจากทำการวัดกระแสไฟฟ้าที่เป็นกระแสสลับ ถ้าหากใช้ Hall Effect เซ็นเซอร์ทั่วไปค่าสัญญาณแรงดันเอาต์พุตที่ออกมาจากเซ็นเซอร์จะเป็นกระแสสลับด้วย นั่นหมายถึงมีทั้งด้านที่เป็นบวกและด้านที่เป็นลบ แต่ไมโครคอนโทรลเลอร์สามารถรับค่าอินพุตได้แต่ค่าบวกเพียงอย่างเดียว จึงอาจจะต้องใช้แรงดันมาไบแอสยกระดับแรงดันไฟฟ้าให้เป็นบวกเพียงอย่างเดียว แต่ในกรณีของเซ็นเซอร์ ACS 712 นั้นมีความสามารถในการยกระดับแรงดันได้ด้วยค่าแรงดันไฟฟ้ากระแสตรง 2.5 โวลต์ ทำให้ค่าเอาต์พุตที่ออกมานั้นเป็นซีกบวกทั้งหมดและสามารถส่งไปยังไมโครคอนโทรลเลอร์ได้ทันทีเพื่ออ่านค่ากระแสที่ไหลผ่านโหลด โดยมีวงจรเซ็นเซอร์วัดกระแสไฟฟ้าดังแสดงในรูปที่ 3.3



รูปที่ 3.3 วงจรเซ็นเซอร์วัดกระแสไฟฟ้า

3.1.2.3 วงจรเซ็นเซอร์วัดแรงดันไฟฟ้า AC

ในการวัดสัญญาณแรงดันไฟฟ้าโดยโมดูลเซ็นเซอร์วัดแรงดันไฟ AC สูงสุดที่ 250VAC สัญญาณที่ออกจากโมดูลเป็นสัญญาณอนาล็อก สามารถนำไปต่อเข้ากับขา ADC ของไมโครคอนโทรเลอร์ Arduino ที่ใช้ Vref +5V ได้ทันที มีวงจรขยายสัญญาณ สามารถปรับขนาดแอมพลิจูดของสัญญาณเอาท์พุตได้ จากการปรับตัวต้านทานปรับค่าบนบอร์ด โดยมีวงจรเซ็นเซอร์วัดแรงดันไฟฟ้าดังแสดงในรูปที่ 3.4

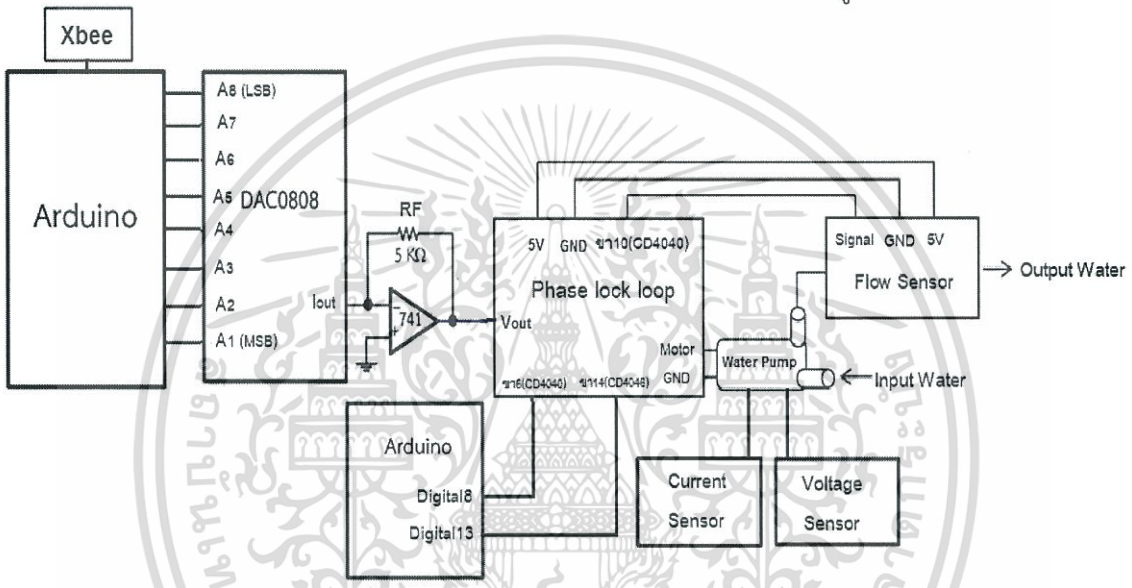


รูปที่ 3.4 วงจรเซ็นเซอร์วัดแรงดันไฟฟ้า

3.1.2.4 ส่วนของมอเตอร์ปั๊มน้ำ

หลักการทำงานของระบบเริ่มจากฝั่งเซิร์ฟเวอร์ที่เชื่อมต่อกับ Xbee ส่งค่าจากโปรแกรม GUI ที่เขียนด้วยภาษา C# จากโปรแกรม MS Visual Studio ควบคุมวงจรรวมที่ใช้เป็นวงจรแปลงสัญญาณดิจิตอลเป็นอนาล็อกผ่านไมโครคอนโทรลเลอร์ที่มี Xbee ภาครับติดอยู่เพื่อตั้งค่าการจ่ายน้ำของตัวมอเตอร์ปั๊มน้ำ โดยแรงดันเอาต์พุตของวงจรรวมที่ใช้เป็นวงจรแปลงสัญญาณ

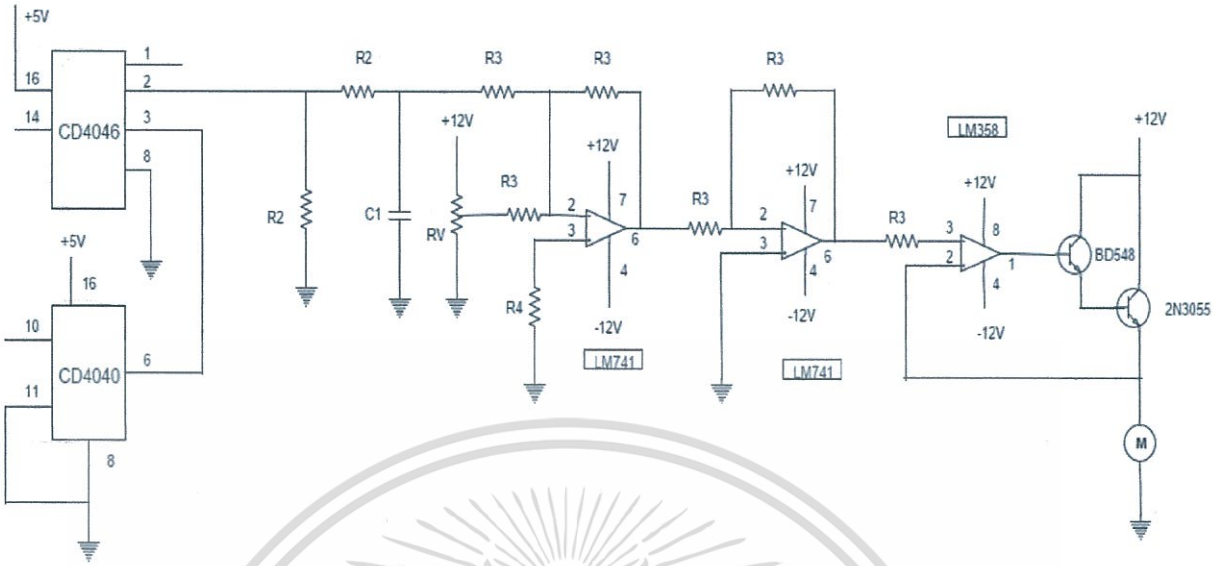
ดิจิตอลเป็นอนาล็อกจะถูกป้อนให้กับวงจร Phase Lock Loop เพื่อสามารถที่จะปรับเปลี่ยนระดับปริมาณการจ่ายน้ำของมอเตอร์ปั๊มน้ำได้ โดยในแต่ละระดับปริมาณเรายังควบคุมให้มีอัตราการไหลที่คงที่สม่ำเสมอโดยใช้ไมโครคอนโทรลเลอร์ตรวจจับความถี่ที่ออกมาจากวงจร Phase Lock Loop หลังจากนั้นก็ใช้ไมโครคอนโทรลเลอร์ตัวเดิมป้อนความถี่เข้าไปให้เท่ากับความถี่ที่ตรวจจับได้ทำให้ระบบการจ่ายน้ำเพื่อผสมสารสำหรับการเลี้ยงสัตว์มีประสิทธิภาพเหมาะสมกับการใช้งานในโรงเรือน และเรายังวัดค่าปริมาณทางไฟฟ้าต่างๆได้แก่ แรงดันไฟฟ้า, กระแสไฟฟ้า เพื่อแจ้งเตือนให้ผู้ใช้ทราบถึงความผิดปกติในการทำงานของมอเตอร์ปั๊มน้ำโดยแสดงผลเป็นกราฟแบบเรียลไทม์ และกราฟที่สามารถเลือกช่วงเวลาได้ มีบล็อกไดอะแกรมการทำงานของระบบแสดงได้ดังรูปที่ 3.5



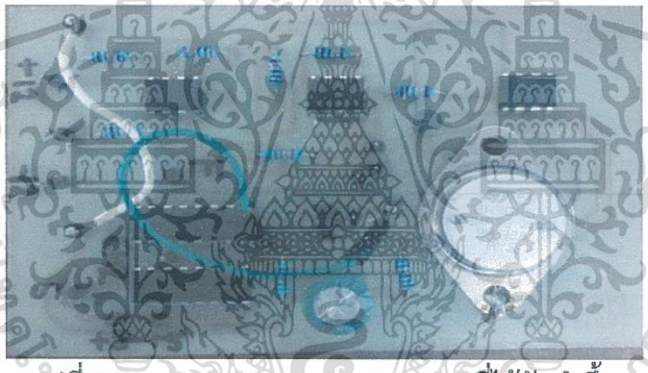
รูปที่ 3.5 บล็อกไดอะแกรมการทำงานของระบบ

3.1.2.5 วงจร Phase Lock Loop (Phase lock loop)

วงจร Phase Lock Loop ใช้สำหรับควบคุมการไหลของน้ำประกอบไปด้วย 4 ส่วนคือ ก. Phase detector ข. วงจรกรองความถี่ต่ำ ค.วงจร VCO และ ง. วงจรหารใช้สำหรับหารความถี่ให้ต่ำลง โดยหลักการทำงานเริ่มจากป้อนสัญญาณ TTL เข้า CD 4046 เป็นสัญญาณอ้างอิงเพื่อเปรียบเทียบกับความถี่น้ำที่ไหลผ่านเข้าวงจรหาร CD 4040 จากนั้นความถี่ของสัญญาณทั้งสองจะเปรียบเทียบกัน ผลลัพธ์ที่ได้คือเอาท์พุตของ Phase detector ต่อมาทำการกรองสัญญาณความถี่ต่ำ เพื่อให้ได้ความถี่เข้าใกล้ความถี่ของ TTL มากที่สุด จากนั้นจะนำความถี่ที่ผ่านการกรองแล้วมาขับเคลื่อนมอเตอร์เพื่อทำการดูดน้ำ การทำงานจะวนลูปอยู่อย่างนี้จนความถี่เข้าใกล้ความถี่ TTL ในที่สุด เรียกว่า Phase lock loop หรือ PPL โดยมีการต่อวงจรแสดงดังรูปที่ 3.6 และวงจรที่ได้จัดทำขึ้นแสดงดังรูปที่ 3.7



รูปที่ 3.6 วงจร Phase Lock Loop



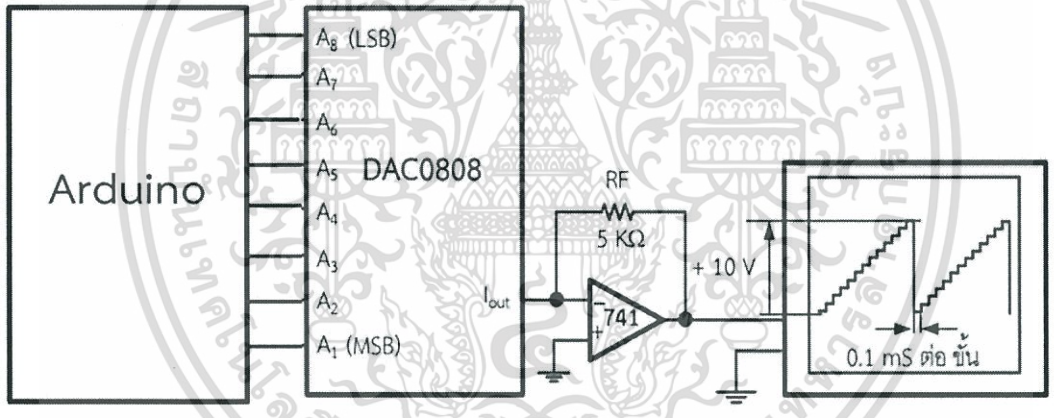
รูปที่ 3.7 ลายวงจรPhase Lock Loop ที่ได้จัดทำขึ้น

3.1.2.6 วงจรรวมที่ใช้เป็นวงจรแปลงสัญญาณดิจิตอลเป็นอนาล็อก

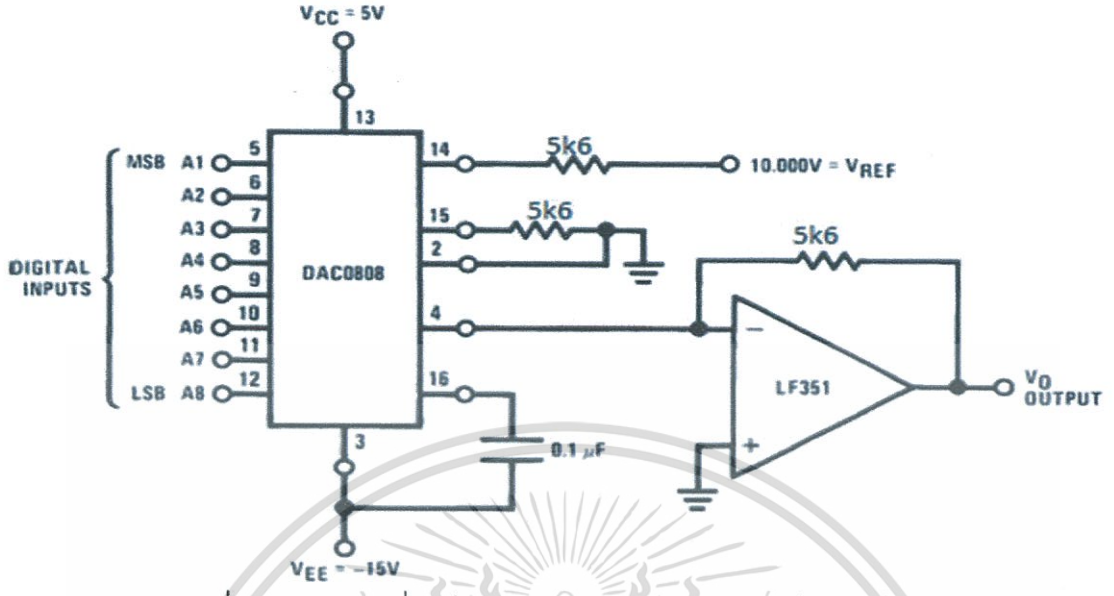
เป็นการประยุกต์ใช้งานด้วยอินพุตดิจิตอล 8 บิต A1 ถึง A8 และโดยที่ อินพุต A1 เป็นบิตนัยสำคัญสูงสุด และให้เอาต์พุตเป็นแบบอนาล็อกที่ 0 V ถึง 10 V โดยต้องมีกระแสอ้างอิง (Iref) สำหรับวงจร D/A ซึ่งให้ค่าแรงดันเอาต์พุตที่ 10 V ที่ค่าความต้านทาน 5 kW ส่วนในการอ้างอิงแรงดันลบ ทำได้จากกการใช้ขา 15 ต่อกับกราวด์ผ่านตัวต้านทานขนาด 5 kW ค่ากระแสอ้างอิงที่ 2 mA เป็นตัวบังคับให้ค่ากระแสเอาต์พุตเต็มพิกัดมีค่าโดยประมาณ 2 mA และ การคำนวณกระแสเอาต์พุตที่เกิดขึ้น โดยการใช้สูตรดังสมการที่ 3.1

$$I_{out} = I_{ref}\left(\frac{A_1}{2}+\frac{A_2}{4}+\frac{A_3}{8}+\frac{A_4}{16}+\frac{A_5}{32}+\frac{A_6}{64}+\frac{A_7}{128}+\frac{A_8}{256}\right) \qquad (3.1)$$

ตัวอย่างเช่น เมื่ออินพุต A1 ถึง A8 มีสภาวะเป็น HIGH ทั้งหมด กระแสเอาต์พุต I_{OUT} จะมีค่าเท่ากับ I_{ref} x 0.996 การแปลงค่ากระแสเอาต์พุตเป็นค่าแรงดันเอาต์พุต โดยการต่อตัวต้านทานอนุกรมที่ขา 4 กับกราวด์ของวงจร จะได้เอาต์พุตจากแรงดันตกคร่อมที่ตัวต้านทานซึ่งเป็นวิธีการที่ง่าย แต่อาจจะมีความไม่แน่นอนเมื่อมีการนำวงจรรวมไปต่อกับโหลดที่มีขนาดต่าง ๆ วิธีการที่มีความถูกต้องมากด้วยการใช้ออปแอมป์ ดังเช่น 741 ซึ่งแสดงในรูปที่ 3.8 กระแสเอาต์พุตจะไหลผ่านไปที่ตัวต้านทาน R_F ซึ่งจะให้ค่าเอาต์พุตเท่ากับ I_{ref} x R_F ช่วงของค่า แรงดันเอาต์พุตสามารถเปลี่ยนได้โดยการเปลี่ยนค่าความต้านทาน R_F และถูกจำกัดค่าแรงดันจากคุณสมบัติของออปแอมป์ที่ใช้งาน การทดสอบวงจรดังกล่าวโดยการใช้ออสซิลโลสโคปและการต่อวงจรนับแบบ 8 บิต ใช้เป็นอินพุตสัญญาณดิจิตอลจากการใช้ออสซิลโลสโคปวัดสัญญาณเอาต์พุต ดังในรูปที่ 3.8 ซึ่งวงจรนับมีการนับค่า จาก 0000 0000 นับขึ้นไปที่ค่า 1111 1111 และแรงดันเอาต์พุตอนาล็อกจะเพิ่มขึ้นจาก 0 V ขึ้นไปจนเกือบ +10 V ใน 256 ขั้น ค่าของระยะเวลาต่อ 1 ขั้น จะมีค่าเท่ากับการกลับส่วนค่าความถี่ของอินพุตสัญญาณนาฬิกา ดังแสดงในวงจรการทดสอบการประยุกต์ใช้วงจรรวมแปลงสัญญาณดิจิตอลเป็นอนาล็อกรูปที่ 3.8 และ มีการต่อวงจรดังแสดงในรูปที่ 3.9 และลายวงจรที่ได้จัดทำขึ้นแสดงดังรูปที่ 3.10



รูปที่ 3.8 วงจรการทดสอบการประยุกต์ใช้วงจรรวมแปลงสัญญาณดิจิตอลเป็นอนาล็อก

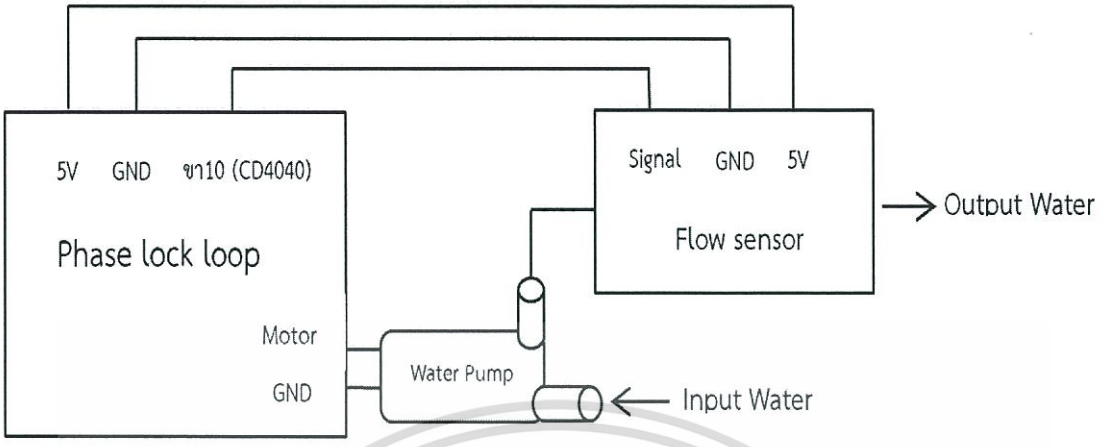


รูปที่ 3.9 วงจรรวมที่ใช้เป็นวงจรแปลงสัญญาณดิจิตอลเป็นอนาล็อก

รูปที่ 3.10 ลายวงจรรวมที่ใช้เป็นวงจรแปลงสัญญาณดิจิตอลเป็นอนาล็อกที่ได้จัดทำขึ้น

3.1.2.7 เซ็นเซอร์วัดอัตราการไหลของน้ำ

ในส่วนของเซ็นเซอร์วัดอัตราการไหลของน้ำนั้น ในโครงงานนี้ได้ใช้เซ็นเซอร์วัดอัตราการไหลของน้ำชนิดวาล์วพลาสติกขนาดหน้าตัดท่อ 0.75 นิ้ว โดยเซ็นเซอร์จะวัดค่าสัญญาณจากรอบการหมุนของน้ำที่ไหลผ่านใบพัด ทั้งนี้แสดงแผนภาพการเชื่อมต่อของเซ็นเซอร์วัดอัตราการไหลของน้ำกับวงจร Phase Lock Loop แสดงได้ดังรูปที่ 3.11



รูปที่ 3.11 แผนผังการเชื่อมต่อระหว่างเซ็นเซอร์วัดอัตราการไหลของน้ำกับวงจร Phase Lock Loop

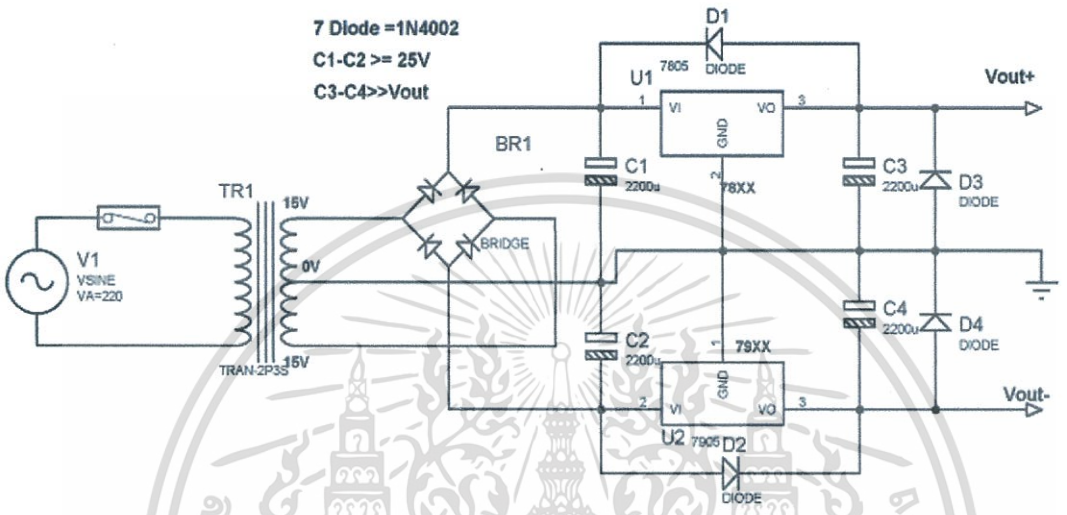
จากรูปที่ 3.11 เป็นแผนผังวงจรการเชื่อมต่อระหว่างเซ็นเซอร์วัดอัตราการไหลของน้ำกับวงจร Phase Lock Loop โดยมีหลักการทำงาน คือ เซ็นเซอร์จะวัดค่าสัญญาณจากรอบการหมุนของใบพัดเมื่อมีน้ำไหลผ่านแล้วทำการส่งสัญญาณขาออกรูปพัลส์ไปยังวงจร Phase Lock Loop จากนั้นวงจร Phase Lock Loop จะมีส่วนหารความถี่ของสัญญาณพัลส์โดยไมโครคอนโทรลเลอร์จะตรวจจับความถี่หลังจากการหารแล้วป้อนความถี่อ้างอิงให้เท่ากับความถี่ที่ตรวจจับได้โดยไมโครคอนโทรลเลอร์ตัวเดิมเพื่อทำให้เฟสนั้นล็อคทำให้อัตราการไหลของน้ำนั้นคงที่สม่ำเสมอ โดยเราสามารถนำค่าสัญญาณพัลส์ดังกล่าวไปคำนวณหาอัตราการไหลของน้ำตามสมการที่ 3.2

$$Q = \frac{F(Hz)}{5.5} \langle l / \min \rangle \qquad (3.2)$$

จากสมการสามารถอธิบายได้ว่า Q คือค่าอัตราการไหลของน้ำ และ F คือความถี่ของสัญญาณที่ได้ ทั้งนี้ความถี่ดังกล่าวคำนวณได้จากสัญญาณขาออกรูปพัลส์ของเซ็นเซอร์

3.1.2.8 วงจรแปลงไฟ 220VAC เป็นไฟบวกลบ 12VDC

ใช้เป็นวงจรวงจรแปลงไฟ 220VAC เป็นไฟบวกลบ 12VDC มี schematic แสดงได้ดังรูปที่ 3.12 และลายวงจรที่ได้จัดทำขึ้นแสดงได้ดังรูปที่ 3.13

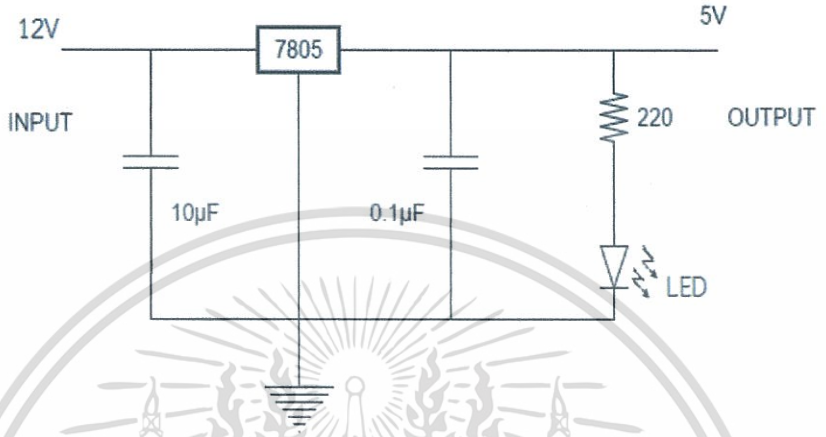


รูปที่ 3.12 วงจรแปลงไฟ 220VAC เป็นไฟบวกลบ 12VDC

รูปที่ 3.13 ลายวงจรแปลงไฟ 220VAC เป็นไฟบวกลบ 12VDC ที่ได้จัดทำขึ้น

3.1.2.9 วงจรแปลงแรงดัน 12V เป็น 5V

ใช้เป็นวงจรวงจรแปลงแรงดัน 12V เป็น 5V แสดงได้ดังรูปที่ 3.14 และลายวงจรที่ได้จัดทำขึ้นแสดงได้ดังรูปที่ 3.15



รูปที่ 3.14 วงจรแปลงแรงดัน 12V เป็น 5V

รูปที่ 3.15 ลายวงจรแปลงแรงดัน 12V เป็น 5V ที่ได้จัดทำขึ้น

3.1.3 การออกแบบและตั้งค่าให้กับ X-Bee

3.1.3.1 การเชื่อมต่อ Xbee กับ PC

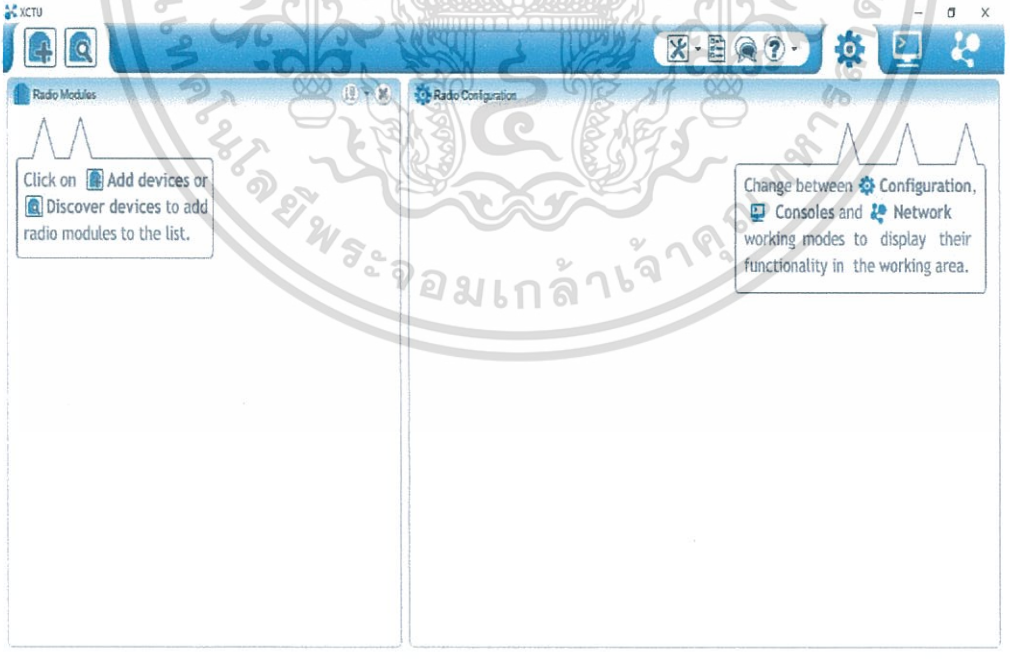
การเชื่อมต่อ Xbee กับ PC นั้นจะกำหนดให้ Xbee ที่เชื่อมต่อกับ PC เซตค่าเป็น Coordinator AT ต่ออยู่กับ Mini Dongle USB ผ่าน USB Port ได้โดยตรง ซึ่งคอมพิวเตอร์จะมองเป็น Com Port (Serial UART) สามารถใช้ได้กับ Xbee ทุกรุ่น

3.1.3.2 การเชื่อมต่อ Xbee กับ Arduino

การเชื่อมต่อ Xbee กับ Arduino จะใช้อุปกรณ์เพิ่มเติมขึ้นมาอีก 1 ชิ้นเรียกว่า Xbee Shield v 2.0 for Arduino ซึ่งสามารถใช้ร่วมกับ Arduino Uno R3 ได้เสียบเข้ากับด้านบนของตัว Arduino สามารถใช้งานรับส่งข้อมูลผ่าน Xbee ได้โดยใช้ฟังก์ชั่นคำสั่ง Software Serial ใน Arduino กำหนดการสื่อสารแบบ Serial Port

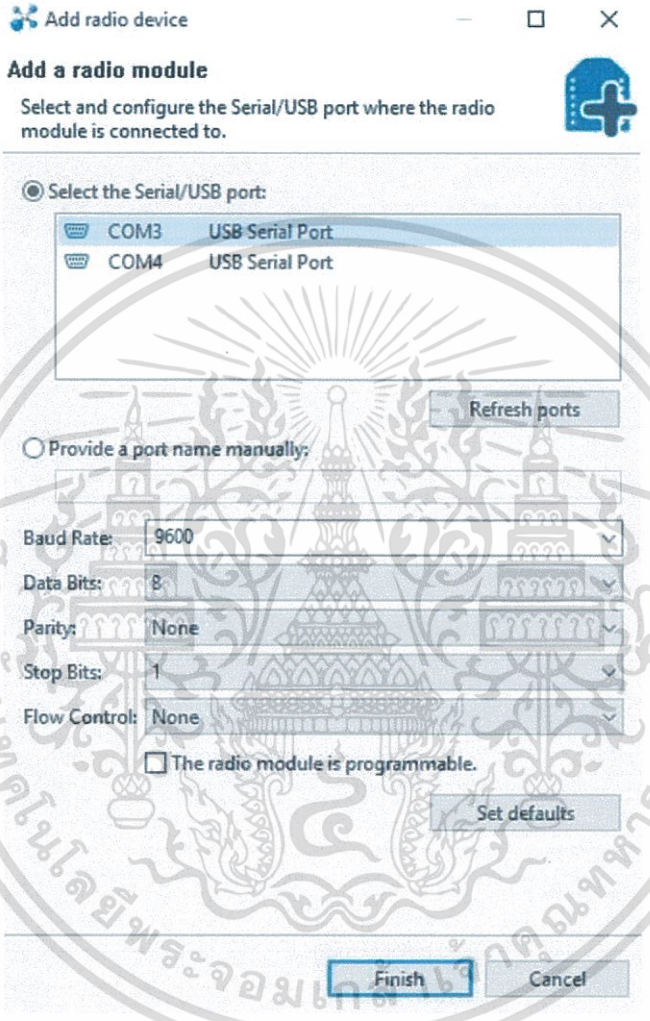
3.1.3.3 การตั้งค่า Xbee

การตั้งค่าให้กับ Xbee เพื่อทำการรับส่งข้อมูลไร้สายนั้นจะใช้โปรแกรมที่ชื่อว่า X-CTU ในการตั้งค่า หน้าต่างโปรแกรม X-CTU จะแสดงดังรูปที่ 3.16



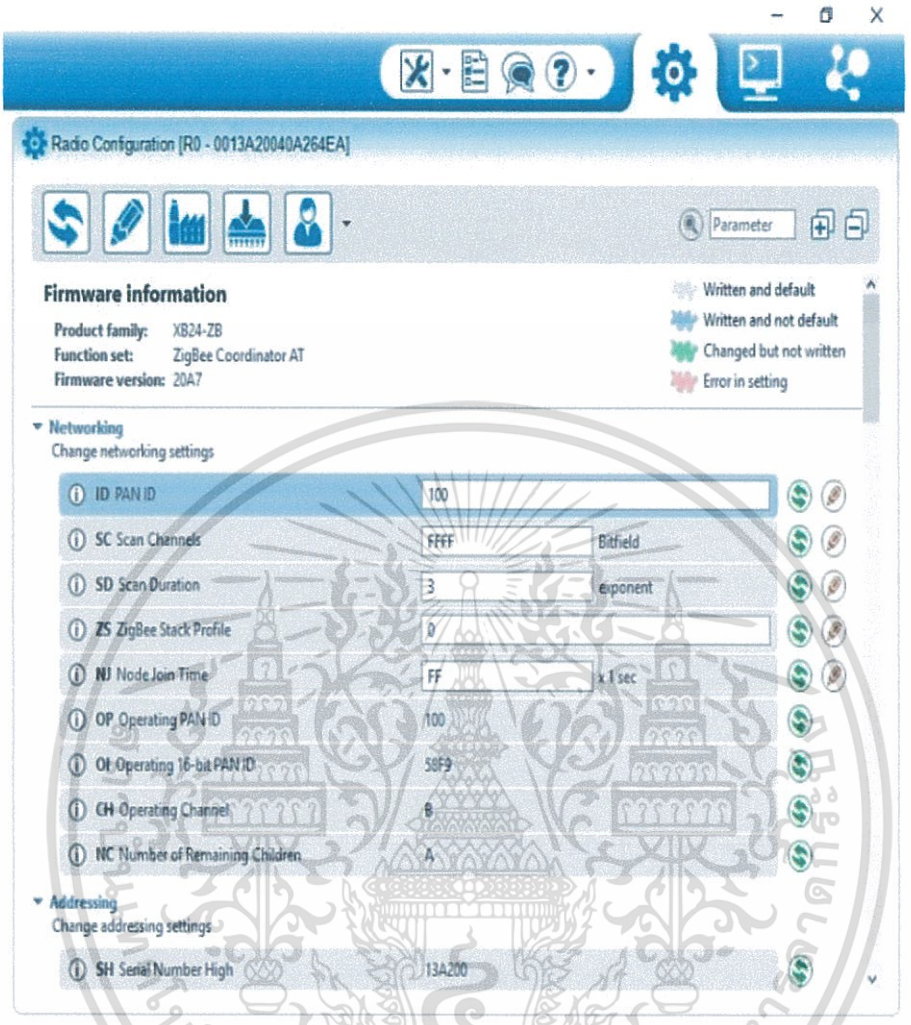
รูปที่ 3.16 หน้าตาของโปรแกรม X-CTU

จากนั้นคลิกที่ Add Radio Device ด้านซ้ายบนจะทำการแสดงพอร์ตที่มีการเชื่อมต่ออุปกรณ์ดังรูปให้ ให้ทำการเลือกคอมพอร์ตของอุปกรณ์นั้นแล้วทำการตั้งค่าแสดงได้ดังรูปที่ 3.17 โดยสามารถเพิ่มได้มากกว่า 1 อุปกรณ์



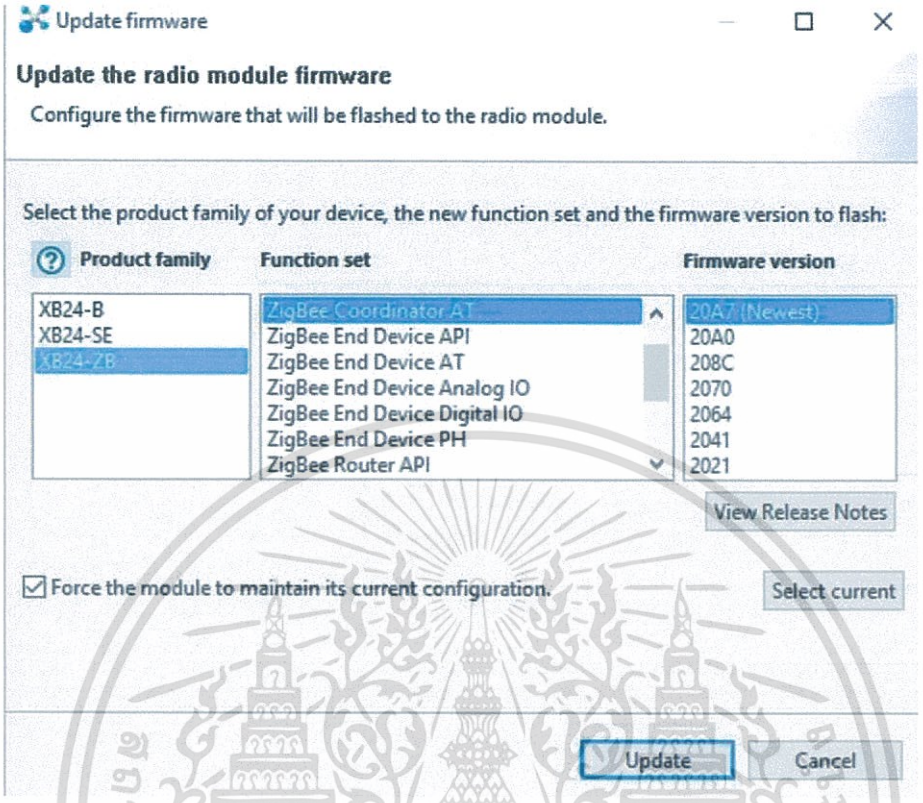
รูปที่ 3.17 การเลือกเพิ่มอุปกรณ์เพื่อทำการตั้งค่าพารามิเตอร์ต่างๆ

จากนั้นจะเห็นว่ามีอุปกรณ์ขึ้นมาที่ด้านซ้ายพร้อมกับรายละเอียดต่างๆ แสดงได้ดังรูปที่ 3.18 เช่น Port, MAC address, Function เป็นต้นและด้านขวาที่ Tap Configuration จะเห็นค่าพารามิเตอร์ต่างๆตัวอุปกรณ์ให้ทำการตั้งค่า PAN ID ให้ตรงกันสำหรับเครือข่ายที่เราจะใช้ เช่น 100 ในหัวข้อ Address ให้ทำการตั้งค่า DH และ DL ให้ตรงกับค่าของ SH และ SL ของ Xbee อีกตัว

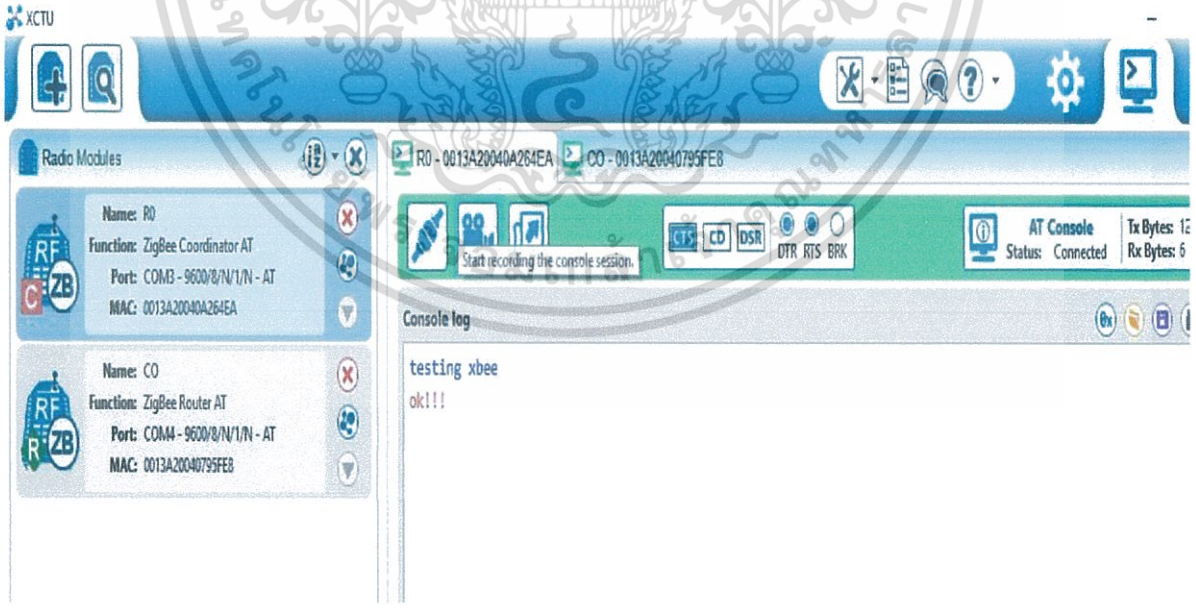


รูปที่ 3.18 รายละเอียดของพารามิเตอร์ต่างๆในอุปกรณ์ Xbee

สุดท้ายก็ทำการอัพเดท Firmware ของ Xbee แสดงได้ดังรูปที่ 3.19 โดยเลือก Function การทำงานของ Xbee ตามที่ออกแบบไว้ โดยให้ตัวหนึ่งเป็น Coordinator AT และอีกตัวหนึ่งเป็น Router AT และทดสอบการรับส่งโดยคลิกไปที่แถบ Console ด้านบนขวาของโปรแกรม X-CTU จากรูปที่ 3.20 จะเห็นว่าสามารถรับส่งข้อความถึงกันได้แล้ว



รูปที่ 3.19 การอัพเดท Firmware ของ Xbee

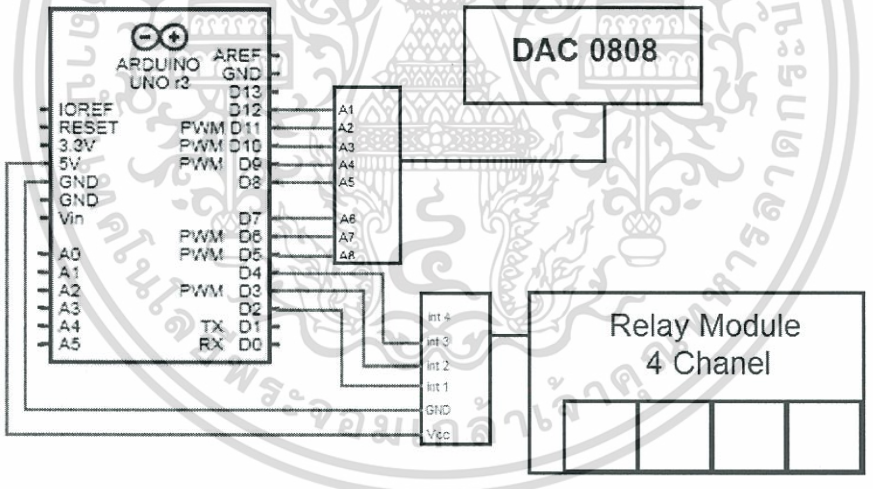


รูปที่ 3.20 ทดสอบการรับส่งข้อมูลของ Xbee

3.1.4 การออกแบบและเขียนโปรแกรมเพื่อควบคุมการทำงานด้วยไมโครคอนโทรลเลอร์

3.1.4.1 ออกแบบส่วนรับค่าเพื่อสั่งงานควบคุมการทำงานอุปกรณ์ไฟฟ้า

ในส่วนของการออกแบบระบบรับคำสั่งเพื่อควบคุมการทำงานนั้นจะต้องทำการเชื่อมต่อไมโครคอนโทรลเลอร์เข้ากับ Xbee โดยเชื่อมต่อขา xbee Tx จาก Xbee Shield เข้ากับขา 0 หรือขา Rx ของไมโครคอนโทรลเลอร์ จากนั้นจะเขียนโปรแกรมกำหนดตัวแปรหนึ่งตัวให้มารับค่าที่ได้รับมาทาง Xbee โดยตัวแปรจะทำการเก็บค่านั้นและเช็คเงื่อนไขการทำงานว่าตรงกับเงื่อนไขใด ตัวอย่างเช่น หากรับค่า '1' มาหมายความว่าให้ทำการเปิดพัดลมตัวที่ 1 หรือ 'A' ให้ทำการปิดพัดลมตัวที่ 1 เป็นต้น โดยที่ประเภทของคำสั่งที่ใช้ในส่วนนี้ได้แบ่งออกเป็น 3 ส่วนหลักๆคือ ควบคุมการเปิด ควบคุมการปิดของพัดลม และสุดท้ายคือควบคุมระดับความเร็วของมอเตอร์ตั้งแต่ 0 จนถึง 12 V โดยค่าที่รับมา โดยกำหนดให้ขา output ของรีเลย์คือ 2, 3, 4 และ output ที่ใช้จ่ายแรงดันเข้า DAC0808 เพื่อใช้ควบคุมระดับแรงดันคือขา 5-12 ลักษณะการต่อวงจรสามารถแสดงได้ดังรูปที่ 3.21 โดยการทำงานของค่าที่รับมาต่างๆจะแสดงได้ดังตารางที่ 3.1

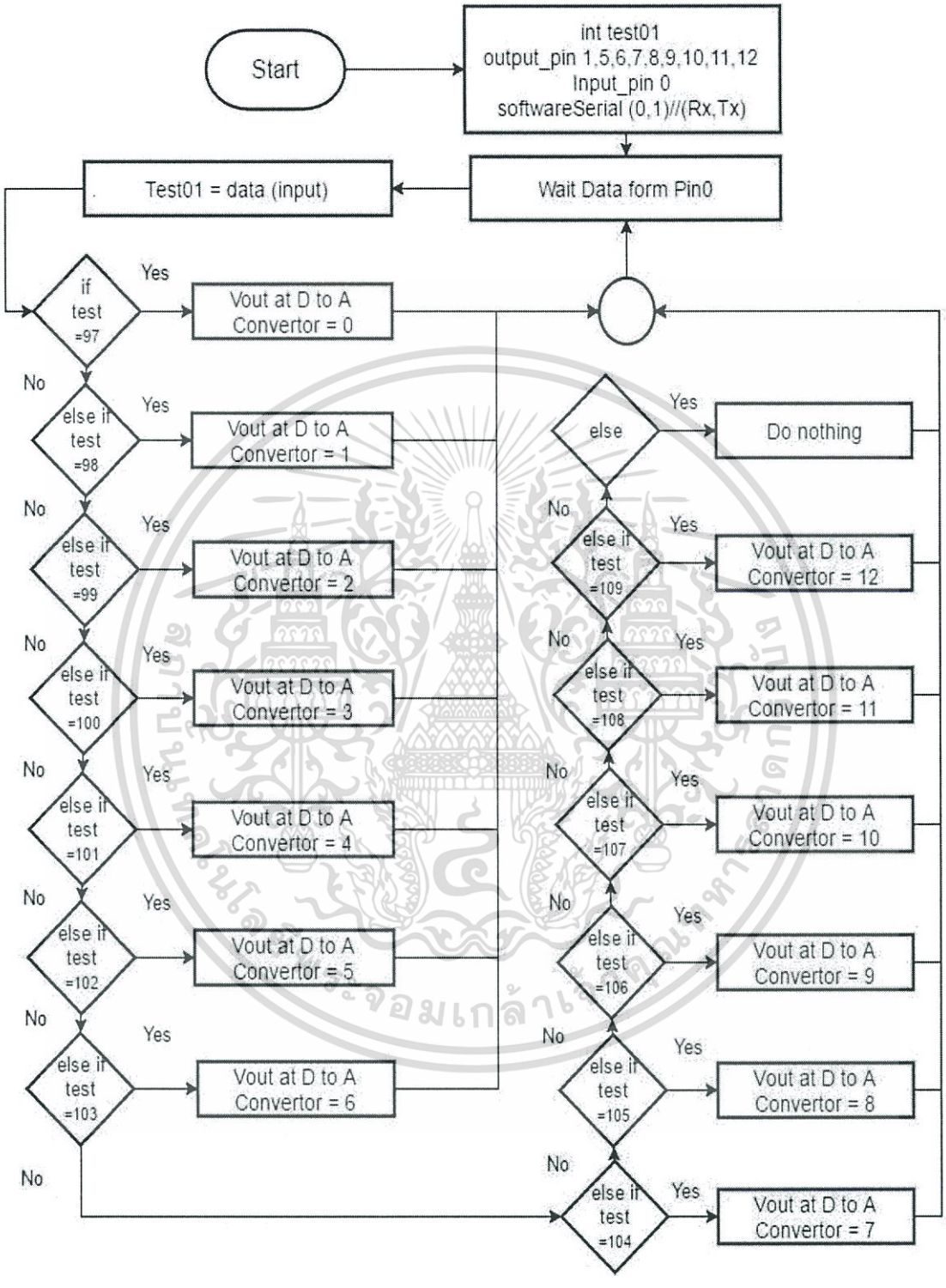


รูปที่ 3.21 ลักษณะการต่อวงจร

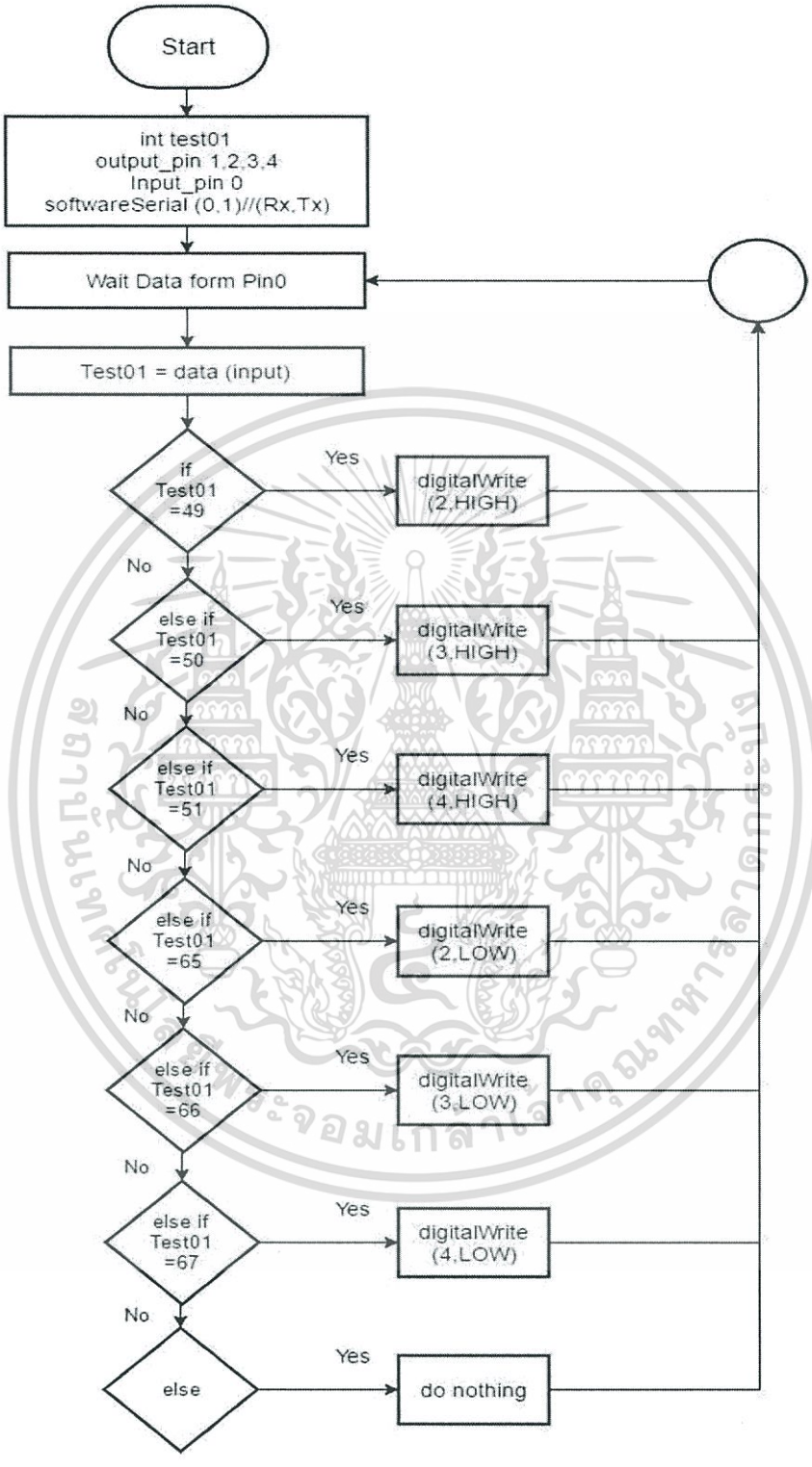
ตารางที่ 3.1 คำสั่งที่ไมโครคอนโทรลเลอร์รับและการทำงาน

คำสั่ง		การทำงาน
Char	DEC	
1	49	รีเลย์ตัวที่ 1 (พัดลมตัวที่ 1) เปิด
2	50	รีเลย์ตัวที่ 2 (พัดลมตัวที่ 2) เปิด
3	51	รีเลย์ตัวที่ 3 (พัดลมตัวที่ 3) เปิด
4	52	รีเลย์ตัวที่ 4 (พัดลมตัวที่ 4) เปิด
A	65	รีเลย์ตัวที่ 1 (พัดลมตัวที่ 1) ปิด
B	66	รีเลย์ตัวที่ 2 (พัดลมตัวที่ 2) ปิด
C	67	รีเลย์ตัวที่ 3 (พัดลมตัวที่ 3) ปิด
D	68	รีเลย์ตัวที่ 4 (พัดลมตัวที่ 4) ปิด
A	97	จ่ายแรงดัน 8 ขา เพื่อให้จ่ายแรงดันเป็น 0 V
B	98	จ่ายแรงดัน 8 ขา เพื่อให้จ่ายแรงดันเป็น 1 V
C	99	จ่ายแรงดัน 8 ขา เพื่อให้จ่ายแรงดันเป็น 2 V
D	100	จ่ายแรงดัน 8 ขา เพื่อให้จ่ายแรงดันเป็น 3 V
E	101	จ่ายแรงดัน 8 ขา เพื่อให้จ่ายแรงดันเป็น 4 V
F	102	จ่ายแรงดัน 8 ขา เพื่อให้จ่ายแรงดันเป็น 5 V
G	103	จ่ายแรงดัน 8 ขา เพื่อให้จ่ายแรงดันเป็น 6 V
H	104	จ่ายแรงดัน 8 ขา เพื่อให้จ่ายแรงดันเป็น 7 V
I	105	จ่ายแรงดัน 8 ขา เพื่อให้จ่ายแรงดันเป็น 8 V
Jd	106	จ่ายแรงดัน 8 ขา เพื่อให้จ่ายแรงดันเป็น 9 V
K	107	จ่ายแรงดัน 8 ขา เพื่อให้จ่ายแรงดันเป็น 10 V
L	108	จ่ายแรงดัน 8 ขา เพื่อให้จ่ายแรงดันเป็น 11 V
M	109	จ่ายแรงดัน 8 ขา เพื่อให้จ่ายแรงดันเป็น 12 V

และแผนผังแสดงการทำงานของโปรแกรมรับค่าเพื่อสั่งงานควบคุมอุปกรณ์ต่างของไมโครคอนโทรลเลอร์สามารถแสดงการทำงานได้แสดงดังรูปที่ 3.22 เป็นการควบคุมระดับแรงดันที่นำไปใช้จ่ายให้แก่วงจร Phase Lock Loop เพื่อควบคุมความเร็วของมอเตอร์ และแสดงการทำงานในส่วนของการควบคุมการเปิดและปิดอุปกรณ์ไฟฟ้าแสดงได้ดังรูปที่ 3.23



รูปที่ 3.22 แผนผังการควบคุมระดับแรงดันของโปรแกรมไมโครคอนโทรลเลอร์

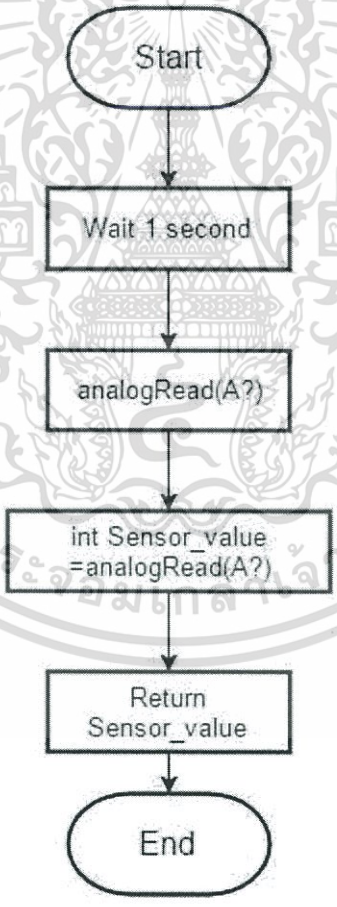


รูปที่ 3.23 แผนผังแสดงการทำงานของวิธีการควบคุมการเปิดและปิดอุปกรณ์ต่างๆ

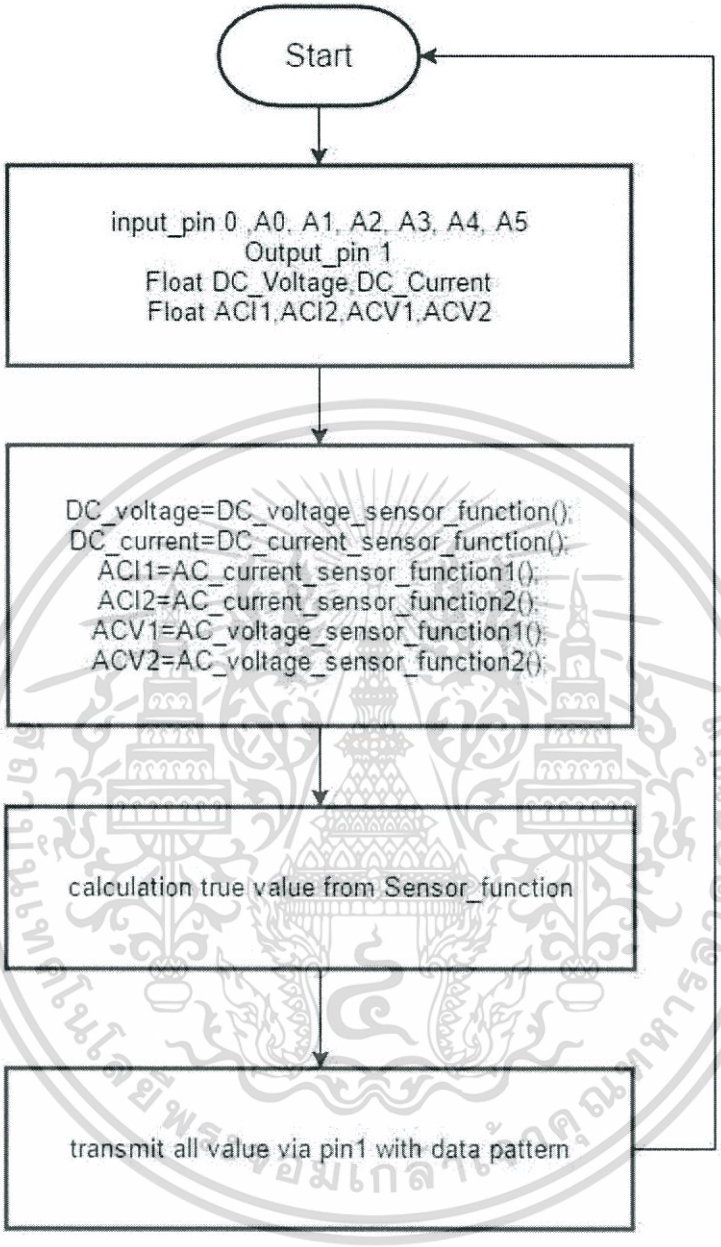
3.1.4.2 ออกแบบส่วนสำหรับการส่งค่าข้อมูลที่วัดได้

นอกจากการควบคุมการเปิด-ปิดอุปกรณ์ไฟฟ้า และส่วนของการควบคุมระดับแรงดันแล้วยังมีส่วนของการส่งข้อมูลกลับไปอีกด้วย ซึ่งก็คือค่าของกระแสและแรงดัน ทั้งแบบ AC (พัดลม) และ DC (มอเตอร์ปั๊มน้ำ) ด้วย ในส่วนนี้จะทำการเขียนโปรแกรมการทำงานใส่ไมโครคอนโทรลเลอร์อีกตัวเพื่อแยกการทำงานออกจากกัน โดยจะต้องทำการส่งทั้งหมด 6 ค่า ด้วยกันโดยในบอร์ด Arduino Uno R3 จะใช้ขา A0-A5 ในการวัดค่า และค่าที่วัดมานี้จะต้องนำมาทำการปรับหรือคำนวณตามสูตรเพื่อให้ได้แค่ที่ควรจะเป็น

ในส่วนของโค้ดโปรแกรมนั้นจะเขียนโปรแกรมโดยแบ่งเป็นส่วนของฟังก์ชันสำหรับเซ็นเซอร์วัดค่าซึ่งจะต้องมีทั้งหมด 6 ฟังก์ชันและส่วนการทำงานหลักหรือ void loop() ในส่วนของฟังก์ชันจะมีรูปแบบการทำงานที่คล้ายๆกันทั้ง 6 ฟังก์ชัน แสดงได้ดังรูปที่ 3.24 และการทำงานของโปรแกรมหลักจะแสดงได้ดังรูปที่ 3.25



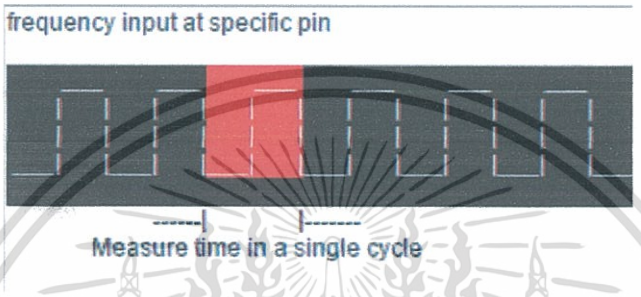
รูปที่ 3.24 รูปแบบการทำงานของฟังก์ชันสำหรับเซ็นเซอร์ต่างๆ



รูปที่ 3.25 การทำงานของโค้ดวัดกระแสและแรงดันของอุปกรณ์ทุกตัวรวมกัน

3.1.4.3 การออกแบบและเขียนโปรแกรมสำหรับวัดค่าและป้อนความถี่

ในส่วนนี้จะใช้ไมโครคอนโทรลเลอร์อีก 1 ตัวสำหรับการวัดและป้อนความถี่เป็นตัวที่ 3 โดยไมโครคอนโทรลเลอร์ตัวนี้จะใช้ Library ที่ชื่อว่า FreqMeasure หลักการทำงานของ FreqMeasure Library ในโหมดวัดจะใช้การวัดช่วงเวลาของ 1 รอบคลื่นที่เข้ามา จากนั้นนำเวลาที่ได้ไปคำนวณเป็นความถี่แสดงได้ดังรูปที่ 3.26 และในไลบรารี่นี้กำหนดขาที่ใช้งานในการวัดแสดงได้ดังรูปที่ 3.27



รูปที่ 3.26 หลักการทำงานของ FreqMeasure Library

Board	Frequency Input Pin	Pins Unusable with analogWrite()
Arduino Uno	8	9, 10
Arduino Duemilanove	8	9, 10
Arduino Mega	49	6, 7, 8
Sanguino	14	12, 13

รูปที่ 3.27 ขาที่ใช้กับไลบรารี่นี้สำหรับ Arduino เวอร์ชันต่างๆ

นอกจากนี้ในไลบรารี่นี้ มีฟังก์ชันสำคัญๆหลายอย่างที่ใช้กันบ่อยๆ เช่น FreqMeasure.begin(); ใช้สำหรับกำหนดเปิดใช้งานไลบรารี่ FreqMeasure.available(); ใช้สำหรับตรวจสอบว่ามีความถี่เข้ามาและพร้อมที่จะอ่านค่าหรือไม่ หากค่าที่อ่านได้เป็น 0 (false) หมายถึงไม่มีความความถี่เข้ามาหรือไม่พร้อมที่จะอ่าน FreqMeasure.read(); ใช้สำหรับอ่านค่าความถี่ และ FreqMeasure.end(); สำหรับปิดการใช้งานไลบรารี่ FreqMeasure Library และคืนค่า PIN กลับไปสู่ PWM (analogWrite) function

โดยในโปรแกรมที่ออกแบบนี้จะใช้ขา 8 ในการวัดความถี่ซึ่งอ่านได้จาก Flow Rate Sensor และป้อนความถี่ที่มีความถี่เดียวกันนี้ออกจากขา 13 เข้าสู่ขา 14 CD4046 ในวงจร Phase Lock Loop

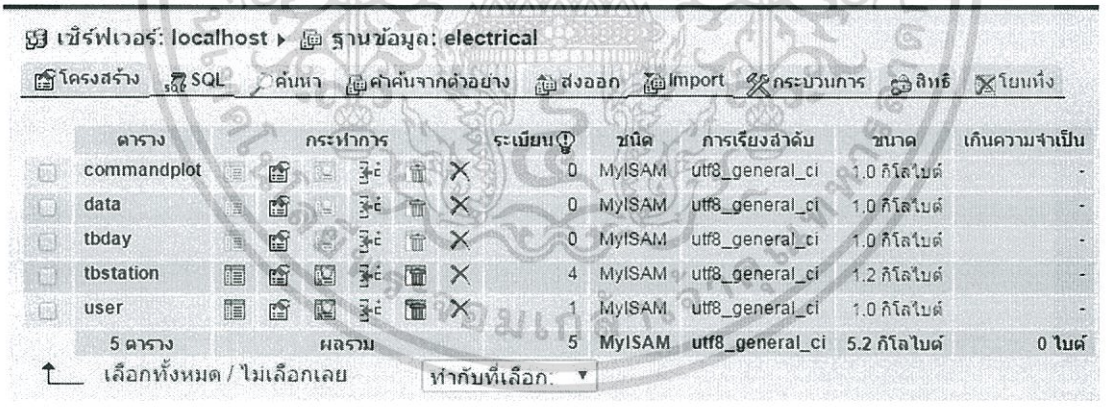
3.1.5 การออกแบบและจัดทำในส่วนของซอร์ฟแวร์

ในส่วนของซอร์ฟแวร์นั้นจะประกอบไปด้วยการออกแบบ 3 ส่วนคือ ระบบฐานข้อมูล ระบบควบคุมการทำงานผ่านเว็บไซต์ และการควบคุมผ่านโปรแกรมสำหรับผู้ใช้งานที่เขียนด้วยภาษา C# ในโปรแกรม Microsoft Visual Studio

ในส่วนของการทำงานนั้นตัวโปรแกรมที่เขียนด้วย C# หรือ GUI (Graphic User Interface) นั้นจะทำการเชื่อมโยงการทำงานระหว่างการส่งข้อมูลที่รับมาทาง Serial Port ส่งเข้าไปเก็บที่ระบบฐานข้อมูล รวมทั้งการบันทึกการทำงานต่างๆ และดึงค่าหรือคำสั่งมาใช้ในการควบคุมระบบไฟฟ้าที่ได้ออกแบบเอาไว้ โดยโปรแกรม C# นอกจากจะทำหน้าที่ดังที่กล่าวมาแล้วยังสามารถเชื่อมต่อกับเว็บซึ่งสามารถสั่งงานระบบได้เช่นเดียวกับสั่งผ่าน GUI โดยตรง ผู้ใช้สามารถที่จะสังเกตผลของค่าที่เซ็นเซอร์วัดได้โดยแสดงเป็นกราฟค่ากระแสและแรงดันในหน้าต่างที่ออกแบบไว้ในโปรแกรม

3.1.5.1 การออกแบบระบบฐานข้อมูล

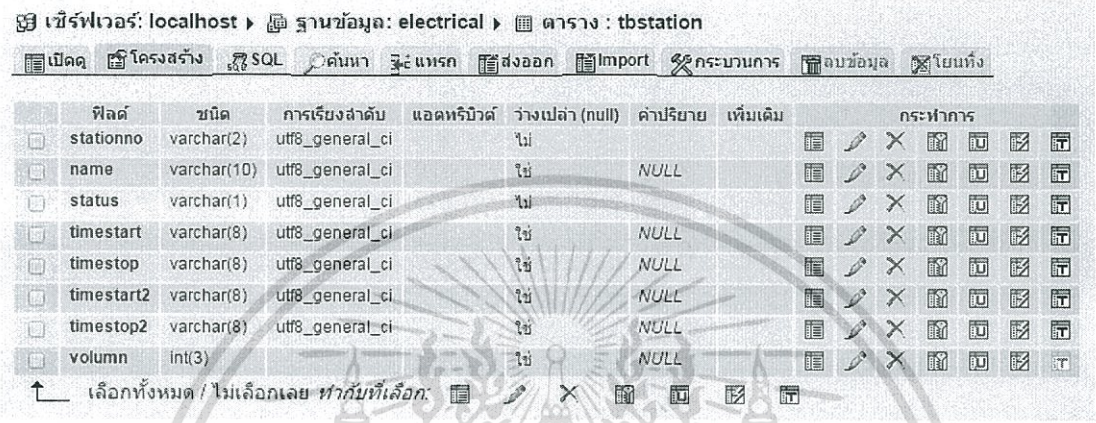
ทำการออกแบบฐานข้อมูล MySQL ชื่อ electrical แสดงได้ดังรูปที่ 3.28 เพื่อใช้เก็บข้อมูลต่างๆของระบบ และออกแบบตารางจัดเก็บข้อมูล 5 ตาราง ตารางที่ 1 commandplot ตารางที่ 2 data ตารางที่ 3 tbstation ตารางที่ 4 user ตารางที่ 5 tbday แสดงรายละเอียดของตารางต่างๆดังนี้



รูปที่ 3.28 ฐานข้อมูล electrical

1) ตารางเก็บคำสั่งควบคุมอุปกรณ์ไฟฟ้า

สร้างตารางเก็บคำสั่งควบคุมอุปกรณ์ไฟฟ้า (tbstation) เพื่อเก็บข้อมูลคำสั่งควบคุมการทำงานอุปกรณ์ไฟฟ้าภายในระบบ แสดงได้ดังรูปที่ 3.29 ข้อมูลชื่อฟิลด์ ชนิด ข้อมูลขนาดข้อมูล คำอธิบาย แสดงในตารางที่ 3.2


เซิร์ฟเวอร์: localhost ▸ ฐานข้อมูล: electrical ▸ ตาราง : tbstation

เปิดดู | โครงสร้าง | SQL | ค้นหา | แทรก | ส่งออก | Import | กระบวนการ | ลบข้อมูล | โยนทิ้ง

ฟิลด์	ชนิด	การเรียงลำดับ	แอตทริบิวต์	ว่างเปล่า (null)	ค่าปริยาย	เพิ่มเติม	กระทำการ
stationno	varchar(2)	utf8_general_ci		ไม่			
name	varchar(10)	utf8_general_ci		ใช่	NULL		
status	varchar(1)	utf8_general_ci		ไม่			
timestart	varchar(8)	utf8_general_ci		ใช่	NULL		
timestop	varchar(8)	utf8_general_ci		ใช่	NULL		
timestart2	varchar(8)	utf8_general_ci		ใช่	NULL		
timestop2	varchar(8)	utf8_general_ci		ใช่	NULL		
volumn	int(3)			ใช่	NULL		

เลือกทั้งหมด / ไม่เลือกเลย *ทำกับที่เลือก:*

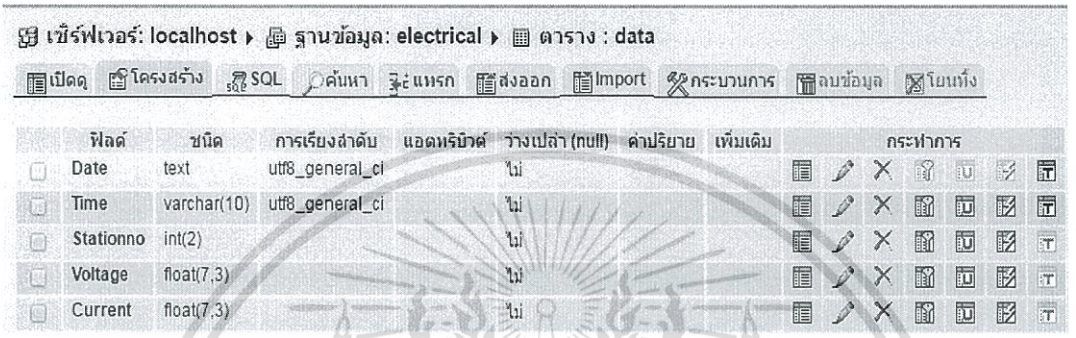
รูปที่ 3.29 การเก็บข้อมูลคำสั่งควบคุมอุปกรณ์ไฟฟ้า tbstation

ตารางที่ 3.2 รายละเอียดของตารางเก็บคำสั่งควบคุมอุปกรณ์ไฟฟ้า (tbstation)

ชื่อฟิลด์	ชนิดข้อมูล	ขนาดข้อมูล	คำอธิบาย
stationno	Varchar	2	หมายเลขอุปกรณ์ไฟฟ้า
name	Varchar	10	รายชื่ออุปกรณ์ไฟฟ้า
status	Varchar	1	สถานะ Station 1 = ทำงาน 2 = หยุดทำงาน
timestart	Varchar	8	เวลาทำงาน1
timestop	Varchar	8	เวลาหยุดทำงาน1
timestart2	Varchar	8	เวลาทำงาน2
timestop2	Varchar	8	เวลาหยุดทำงาน2
volumn	Int	3	เก็บค่าเปอร์เซ็นต์อัตราไหล 0 – 100

2) ตารางเก็บข้อมูลกระแสและแรงดัน

สร้างตารางเก็บข้อมูลกระแสและแรงดัน (data) เพื่อเก็บข้อมูลกระแสและแรงดันของอุปกรณ์ไฟฟ้าต่างๆ ภายในระบบ แสดงได้ดังรูปที่ 3.30 ข้อมูลชื่อฟิลด์ ชนิด ข้อมูล ขนาดข้อมูล คำอธิบาย แสดงในตารางที่ 3.3



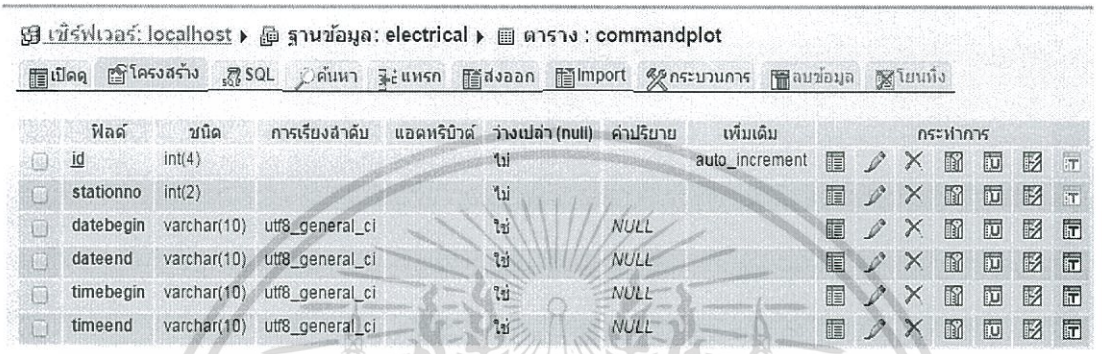
รูปที่ 3.30 การเก็บข้อมูลกระแสและแรงดัน (data)

ตารางที่ 3.3 รายละเอียดของตารางเก็บข้อมูลกระแสและแรงดัน (data)

ชื่อฟิลด์	ชนิดข้อมูล	ขนาดข้อมูล	คำอธิบาย
Date	text		วันที่รูปแบบ DD/MM/YYYY
Time	varchar	10	เวลา HH:MM:SS
Stationno	int	2	หมายเลขอุปกรณ์ไฟฟ้า
Voltage	float	(7,3)	แรงดัน
Current	float	(7,3)	กระแส

3) ตารางเก็บคำสั่งพล็อตกราฟ

สร้างตารางเก็บคำสั่งพล็อตกราฟ (commandplot) เพื่อเก็บข้อมูลคำสั่ง วันที่ และ ช่วงเวลาที่ต้องการจะ plot กราฟ แสดงได้ดังรูปที่ 3.31 ข้อมูลชื่อฟิลด์ ชนิด ข้อมูล ขนาดข้อมูล คำอธิบาย แสดงในตารางที่ 3.4



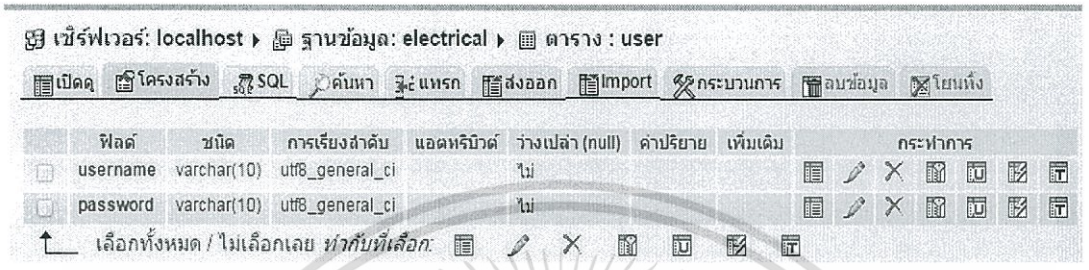
รูปที่ 3.31 การเก็บข้อมูลคำสั่งพล็อตกราฟ (commandplot)

ตารางที่ 3.4 รายละเอียดของตารางเก็บคำสั่งพล็อตกราฟ (commandplot)

ชื่อฟิลด์	ชนิดข้อมูล	ขนาดข้อมูล	คำอธิบาย
Id	int	4	หมายเลขคำสั่ง
stationno	int	2	หมายเลขอุปกรณ์ไฟฟ้า
datebegin	varchar	10	วันเริ่มรูปแบบ DD/MM/YYYY
dateend	varchar	10	วันสุดท้ายรูปแบบ DD/MM/YYYY
timebegin	varchar	10	เวลาเริ่ม HH:MM:SS
timeend	varchar	10	เวลาหยุด HH:MM:SS

4) ตารางเก็บชื่อผู้ใช้

สร้างตารางเก็บชื่อผู้ใช้ (user) เพื่อเก็บรายชื่อและรหัสผู้ใช้งานภายในระบบ แสดงได้ดังรูปที่ 3.32 ข้อมูลชื่อฟิลด์ ชนิดข้อมูล ขนาดข้อมูล คำอธิบายแสดงในตารางที่ 3.5



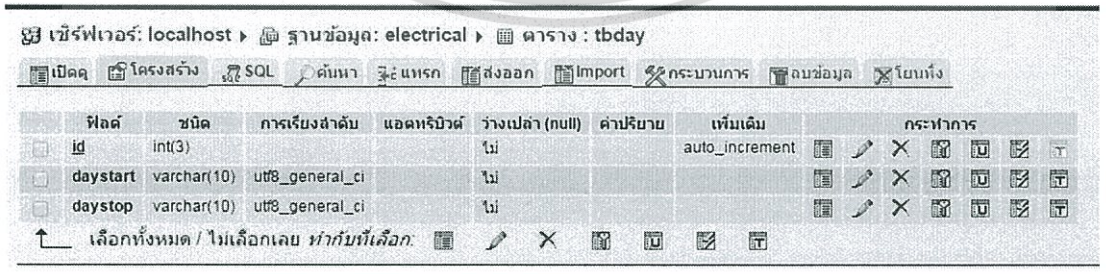
รูปที่ 3.32 การเก็บข้อมูลชื่อผู้ใช้(user)

ตารางที่ 3.5 รายละเอียดของตารางเก็บชื่อผู้ใช้งาน (user)

ชื่อฟิลด์	ชนิดข้อมูล	ขนาดข้อมูล	คำอธิบาย
username	varchar	10	ชื่อผู้ใช้
password	varchar	10	รหัสผู้ใช้

5) ตารางเก็บคำสั่งช่วงวันทำงาน

สร้างตารางเก็บคำสั่งช่วงวันทำงาน เพื่อเก็บคำสั่งวันทำงานและวันหยุดทำงาน แสดงได้ดังรูปที่ 3.33 ข้อมูลชื่อฟิลด์ ชนิดข้อมูล ขนาดข้อมูล คำอธิบายแสดงในตารางที่ 3.6



รูปที่ 3.33 การเก็บข้อมูลคำสั่งช่วงวันทำงาน

ตารางที่ 3.6 รายละเอียดของตารางเก็บคำสั่งช่วงวันทำงาน

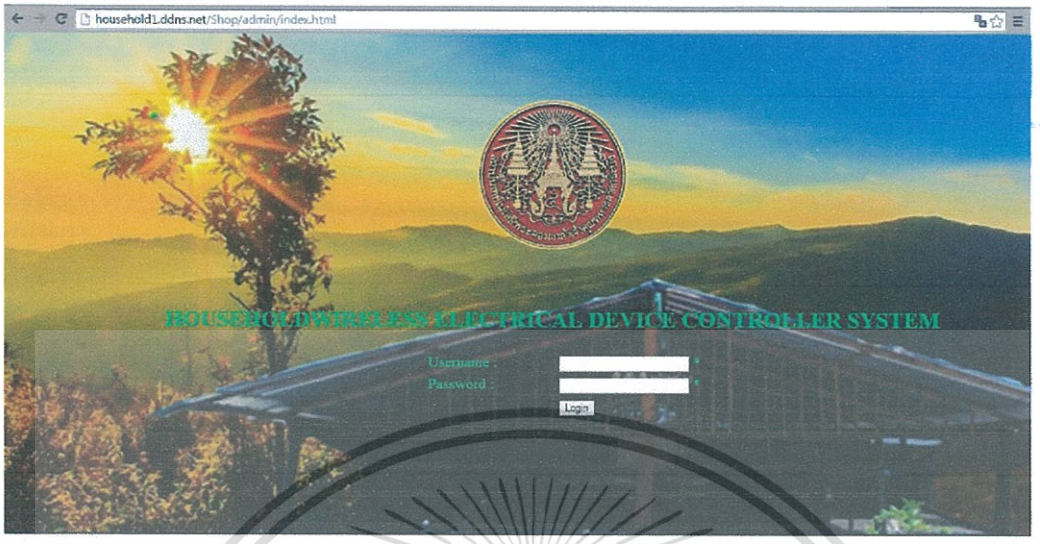
ชื่อฟิลด์	ชนิดข้อมูล	ขนาดข้อมูล	คำอธิบาย
id	int	3	หมายเลขคำสั่ง
daystart	varchar	10	วันทำงาน
daystop	varchar	10	วันหยุดทำงาน

3.1.5.2 การออกแบบเว็บไซต์

ทำการออกแบบหน้าเว็บต่างๆ ไว้เพื่อให้ผู้ใช้งานในระบบเท่านั้นที่ล็อกอินใช้งานระบบได้ ใช้งานอย่างสะดวก ภายในระบบมีหน้าเว็บควบคุมอุปกรณ์ไฟฟ้าภายในโรงเรือน อาทิเช่น พัดลม ปั๊มน้ำ และมีหน้าเว็บที่สามารถดูข้อมูลกระแส แรงดันของอุปกรณ์ไฟฟ้าต่างๆย้อนหลังได้ แสดงผลออกมาในรูปแบบกราฟเวลาเทียบกับแรงดันหรือกระแส ซึ่งจะมีหน้าเว็บแสดงรายละเอียดอุปกรณ์ไฟฟ้าโดยรูปแบบหน้าเว็บต่างๆมีดังต่อไปนี้

1) หน้าเว็บล็อกอิน

ทำการออกแบบหน้า login สำหรับให้ผู้ดูแลระบบเท่านั้นที่สามารถเข้าใช้งานภายในระบบได้ เมื่อเริ่มเข้าใช้งานระบบต้องผ่านหน้าเว็บนี้ก่อนทุกครั้งไม่สามารถไปหน้าเว็บอื่นโดยที่ไม่ผ่านหน้านี้ได้ ซึ่งจะมีไฟล์ checksession.php เป็นไฟล์ตรวจสอบ โดยต้องทำการ กรอก username และ password แล้วกด Login แสดงได้ดังรูปที่ 3.34 ไฟล์ที่เกี่ยวข้องประกอบไปด้วยไฟล์ index.html ซึ่งเป็นไฟล์แสดงหน้าล็อกอินประกอบไปด้วยช่อง Username กับ Password ให้ผู้ใช้กรอกแล้วคลิกปุ่ม Login จะส่งทั้งสองค่าไปไฟล์ checklogin.php เพื่อตรวจสอบเทียบกับค่าที่อยู่ในตาราง user ถ้าตรงก็นำทางไปสู่หน้าเว็บ home



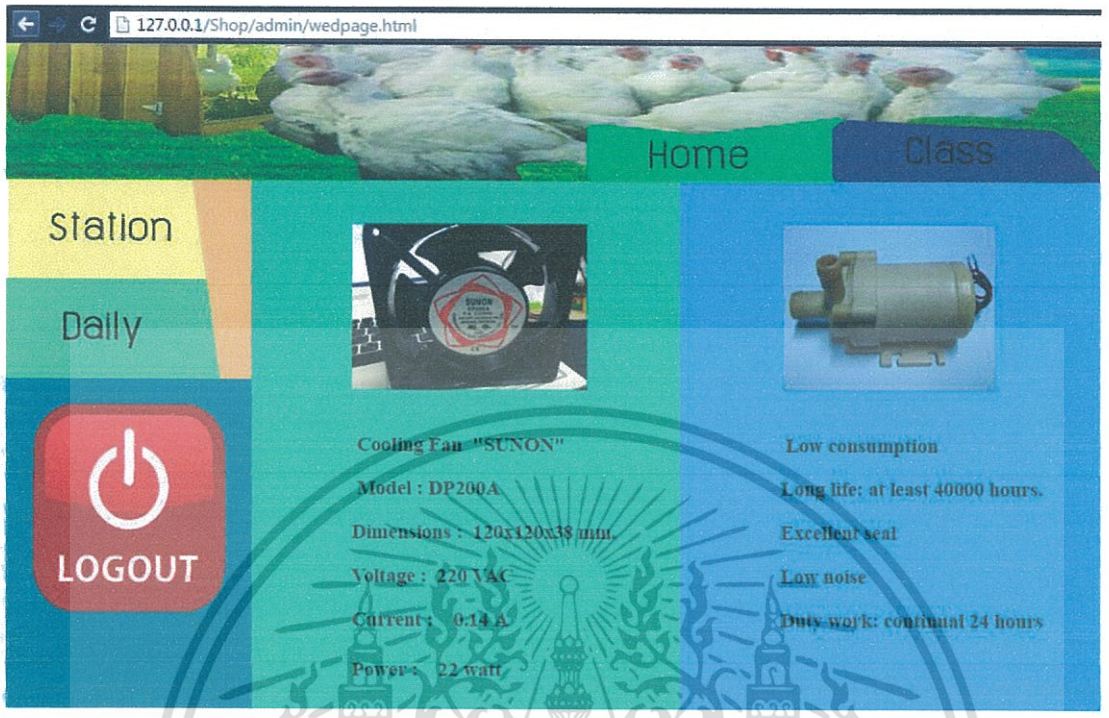
รูปที่ 3.34 หน้าเว็บล็อกอิน

2) หน้าเว็บ home

หน้าเว็บ home เป็นกลุ่มหน้าเว็บที่มีลักษณะเหมือนกัน ประกอบไปด้วยปุ่มลิงค์ไปหน้าเว็บอื่นๆ อาทิเช่น หน้าเว็บควบคุมอุปกรณ์ไฟฟ้า ไฟล์ที่เกี่ยวข้องประกอบไปด้วยไฟล์ webpage1.html webpage2.html ซึ่งทั้งสองไฟล์มีลักษณะองค์ประกอบที่เหมือนกัน

ก.ส่วนหน้าเว็บแสดงข้อมูลอุปกรณ์ไฟฟ้า

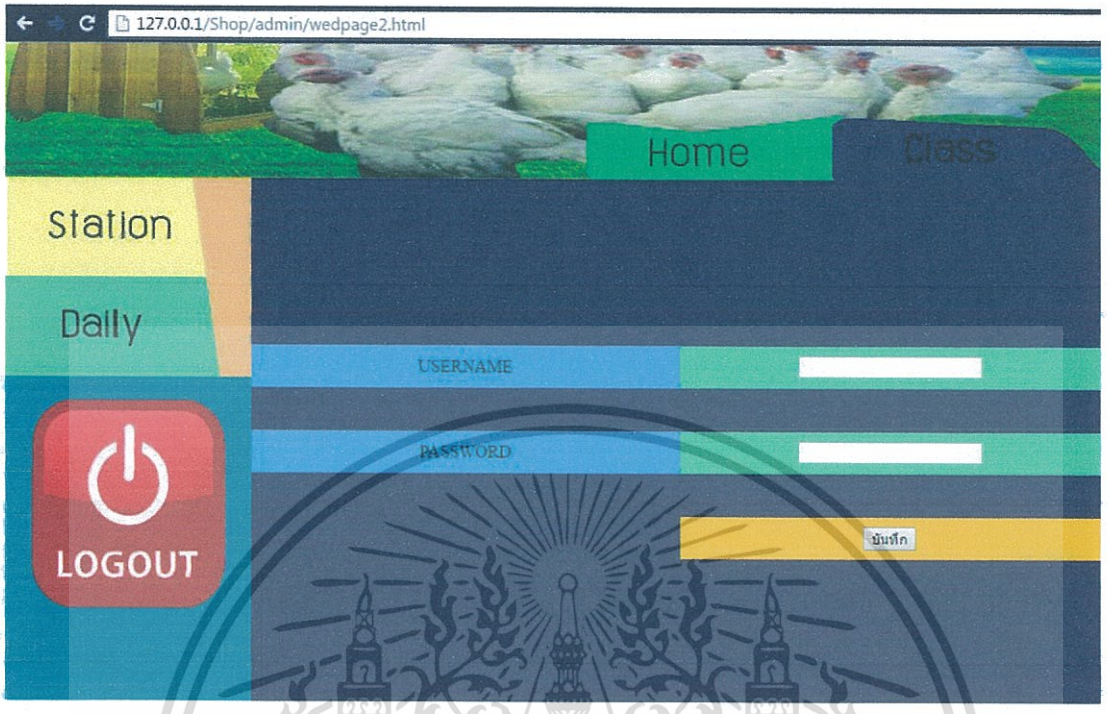
webpage1.html หน้าเว็บนี้แสดงได้ดังรูปที่ 3.35 แสดงข้อมูลอุปกรณ์ไฟฟ้าภายในระบบ เช่น พัดลม ปั๊มน้ำ ซึ่งจะแสดงข้อมูลเบื้องต้นต่างๆ



รูปที่ 3.35 หน้าเว็บแสดงข้อมูลอุปกรณ์ไฟฟ้า

ข.ส่วนหน้าเว็บสำหรับเพิ่มผู้ดูแลระบบ

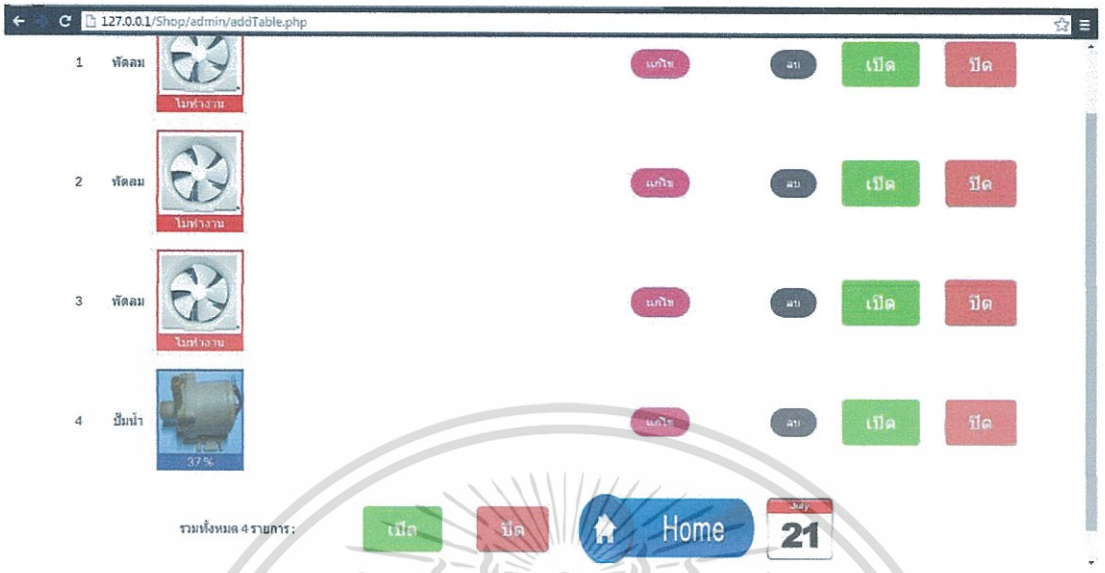
webpage2.html หน้าเว็บนี้แสดงได้ดังรูปที่ 3.36 มีช่องไว้สำหรับเพิ่มผู้ดูแลระบบ ซึ่งใช้สำหรับล็อกอินเข้าใช้ระบบ ประกอบไปด้วยช่อง username ชื่อผู้ใช้ และ password รหัส เมื่อทำการกรอกครบแล้วคลิกปุ่มบันทึกจะ save ข้อมูลเข้าสู่ตาราง user



รูปที่ 3.36 หน้าเว็บเพิ่มผู้ดูแลระบบ

3) ส่วนหน้าเว็บสำหรับควบคุมอุปกรณ์ไฟฟ้า

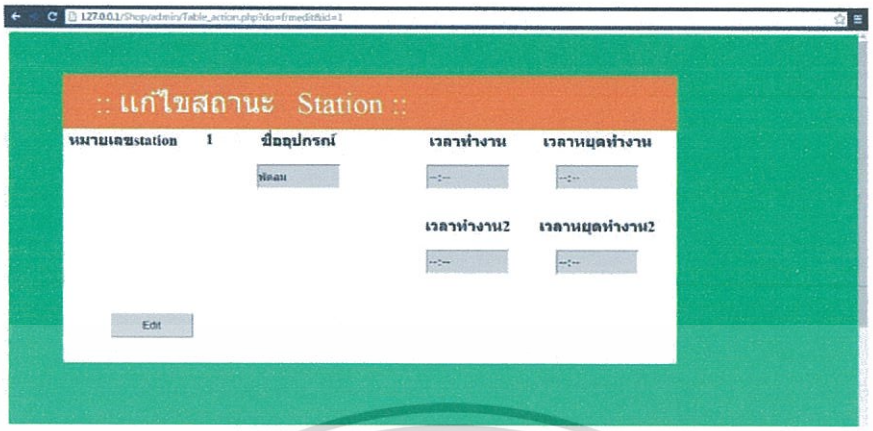
เมื่อคลิกปุ่ม Station ในหน้าเว็บ home จะนำทางไปสู่หน้าเว็บควบคุมอุปกรณ์ไฟฟ้าแสดงได้ดังรูปที่ 3.37 หน้าเว็บแสดงอุปกรณ์ไฟฟ้าที่มีอยู่ในระบบ ผู้ใช้สามารถเพิ่ม ลบได้และและตั้งช่วงวันที่ทำงานโดยการตั้งวันทำงานและวันหยุดทำงาน (ทำการคลิกปุ่มรูปวันที่) ใช้ควบคุมอุปกรณ์ไฟฟ้าทั้งพัดลมและปั๊มน้ำซึ่ง หมายเลข 1 – 3 จะเป็นพัดลม และ 4 ปั๊มน้ำ ซึ่งการควบคุมพัดลมมีสามแบบ คือควบคุมโดยการตั้งเวลาเปิดปิด (ทำการคลิกปุ่มแก้ไขในแต่ละแถวของอุปกรณ์) และ เปิดปิด (ทำการคลิกปุ่มเปิดหรือปิดในแต่ละแถวของอุปกรณ์) เปิดปิดพัดลมทุกตัว (ทำการคลิกปุ่มเปิดหรือปิดด้านล่างสุด) และมีรูปภาพพัดลมแสดงสถานะทำงานของพัดลม ส่วนการควบคุมปั๊มน้ำมีแบบเดียวคือการเลื่อนปรับ volume (ทำการกดปุ่มแก้ไขในแถวที่ 4) ซึ่งมี ค่าเป็น 0 – 100% โดยไฟล์ที่เกี่ยวข้องประกอบไปด้วย ไฟล์ addTable.php ไฟล์แสดงหน้าเว็บควบคุมอุปกรณ์ไฟฟ้า ไฟล์ Table_action.php ไว้ใช้ประมวลผลเพื่อเก็บคำสั่งควบคุมอุปกรณ์ไฟฟ้าเข้าสู่ Database



รูปที่ 3.37 หน้าเว็บสำหรับควบคุมอุปกรณ์ไฟฟ้า

4) หน้าแก้ไขข้อมูลอุปกรณ์ไฟฟ้า

เมื่อทำการคลิกปุ่มแก้ไขก็จะนำทางไปสู่หน้าแก้ไขคำสั่งควบคุมอุปกรณ์ไฟฟ้าไว้สำหรับควบคุมอุปกรณ์ไฟฟ้า มีสองแบบ แบบแรกของอุปกรณ์พัดลมควบคุมการทำงานโดยตั้งเวลาเปิด-ปิด มีให้ตั้งสองช่วงเวลา จากรูปที่ 3.38 ช่วงเวลาแรกแถวบนตั้งเวลาได้ภายในช่วง 00:00 น. ถึง 12:00 น. ช่วงเวลาสองแถวล่างตั้งเวลาได้ภายในช่วง 12:01 น. ถึง 23:59 น. และแบบสองของอุปกรณ์ปั๊มน้ำควบคุมการทำงานโดยปรับเปอร์เซ็นต์อัตราไหล มีช่วงในการปรับ 0 – 100% แสดงดังรูปที่ 3.39 เมื่อทำการตั้งเวลาหรือปรับ เปอร์เซ็นต์อัตราไหลแล้ว ทำการคลิกปุ่ม edit คำสั่งจะบันทึกลงในตารา tbstation ณ บรรทัดของอุปกรณ์นั้นๆ

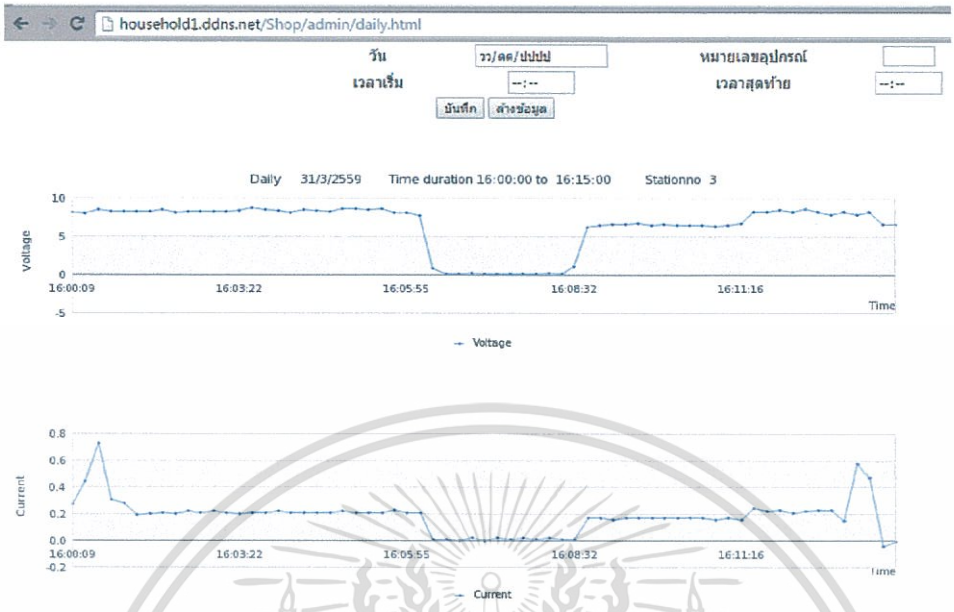


รูปที่ 3.38 หน้าแก้ไขข้อมูลอุปกรณ์ไฟฟ้าแบบแรก

รูปที่ 3.39 หน้าแก้ไขข้อมูลอุปกรณ์ไฟฟ้าแบบสอง

5) หน้าเว็บแสดงข้อมูลกราฟ

เมื่อนำการคลิกปุ่ม daily ในกลุ่มหน้าเว็บ home จะนำทางไปสู่หน้าเว็บแสดงข้อมูลกราฟแสดงได้ดังรูปที่ 3.40 หน้าเว็บนี้แสดงข้อมูลเป็นกราฟ ประกอบไปด้วยสองส่วน ส่วนแรก ไว้สำหรับป้อนคำสั่งพล็อตกราฟ เมื่อทำการกรอกข้อมูลครบถ้วนแล้วคลิกปุ่มบันทึก คำสั่งจะเก็บในตาราง commandplot ส่วนที่สอง แสดงข้อมูลในลักษณะกราฟแรงดันและกระแสเทียบกับเวลา ซึ่งจะพล็อตตามคำสั่งล่าสุดที่อยู่ในตาราง commandplot ไฟล์ที่เกี่ยวข้องประกอบไปด้วย ไฟล์ daily.html ไว้แสดงหน้าเว็บนี้ ไฟล์plot.php เก็บคำสั่ง plot กราฟเข้าสู่ตาราง commandplot ไฟล์ example1.php เรียกข้อมูลแรงดันพล็อตเทียบกับเวลาตามคำสั่งล่าสุดในตาราง commandplot ไฟล์ example2.php เรียกข้อมูลกระแสพล็อตเทียบกับเวลาตามคำสั่งล่าสุดในตาราง commandplot



รูปที่ 3.40 หน้าเว็บแสดงข้อมูลกราฟ

6) หน้าเว็บตั้งช่วงวันทำงาน

เมื่อนำการคลิกปุ่มรูปวันที่ในหน้าเว็บควบคุมอุปกรณ์ไฟฟ้า จะนำทางไปสู่หน้าเว็บตั้งช่วงวันทำงานแสดงได้ดังรูปที่ 3.41 หน้าเว็บนี้ใช้ตั้งช่วงวันทำงานโดยการตั้งวันทำงานและวันหยุดทำงานเมื่อทำการคลิกปุ่ม edit คำสั่งบันทึกลงในตาราง tbday

รูปที่ 3.41 หน้าเว็บตั้งช่วงวันทำงาน

3.1.5.3 การออกแบบโปรแกรม c#

ได้ทำการออกแบบโปรแกรม เพื่อทำการอ่านคำสั่งการทำงานของระบบจากฐานข้อมูล และประมวลผลส่งคำสั่งผ่านโมดูลไร้สาย Zigbee ไปยัง Zigbee ที่ต่อกับไมโครคอลโทรลเลอร์ สั่งอุปกรณ์ไฟฟ้าให้ทำงานต่อไป การทำงานของเริ่มจากการอ่านคำสั่งช่วงวันทำงานและหยุดทำงานจากตาราง tbday แล้วเปรียบเทียบว่าวันปัจจุบันอยู่ในช่วงวันนั้นหรือไม่ ถ้าอยู่จะทำการเชื่อมต่อ โมดูลไร้สาย Zigbee จากนั้นจะทำการอ่านคำสั่งควบคุมอุปกรณ์ไฟฟ้าแต่ละตัวในตาราง tbstation โดยทำการอ่านคำสั่งทีละบรรทัด และตรวจสอบว่าพัดลมตัวที่ 1 – 3 (บรรทัดที่ 1 – 3) อยู่ในโหมดตั้งเวลาเปิด-ปิด (มีค่าเวลาเปิด-ปิดในบรรทัดนั้นๆ) หรือ เปิด-ปิด (ไม่มีค่าเวลาเปิด-ปิดในบรรทัดนั้นๆ) และทำการส่งคำสั่งผ่านโมดูลไร้สายดังตารางที่ 3.7 ตรวจสอบว่าปั๊มน้ำทำงานอยู่ที่เปอร์เซ็นต์ (volumn) เท่าไร (บรรทัดที่ 4) และส่งคำสั่งผ่านโมดูลไร้สายต่อไปดังตารางที่ 3.8 และทำการรับค่าที่ได้จากการอ่านเซนเซอร์ทั้งกระแสและแรงดันผ่านโมดูลไร้สาย บันทึกค่าพร้อมวันเวลา ลงตาราง data นอกจากนั้นยังมีหน้าต่าง GUI ต่างๆดังนี้

ตารางที่ 3.7 การส่งคำสั่งควบคุมพัดลมตัวที่ 1 ตัวที่ 2 และตัวที่ 3 ผ่านโมดูลไร้สาย Zigbee

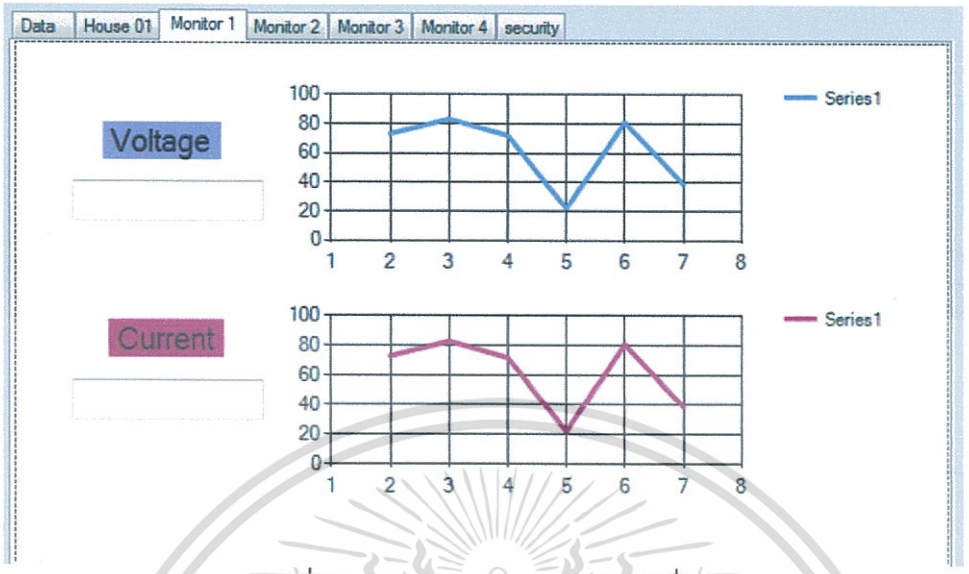
ค่า status ในตาราง tbstation	คำสั่งที่ส่งผ่าน Zigbee			
	พัดลมตัวที่ 1 stationno =“1”	พัดลมตัวที่ 2 stationno =“2”	พัดลมตัวที่ 3 stationno =“3”	สถานะ
0	A	B	C	หยุดทำงาน (ปิด)
1	1	2	3	ทำงาน (เปิด)

ตารางที่ 3.8 การส่งคำสั่งควบคุมปั๊มน้ำ (stationno ="4") ผ่านโมดูลไร้สาย Zigbee

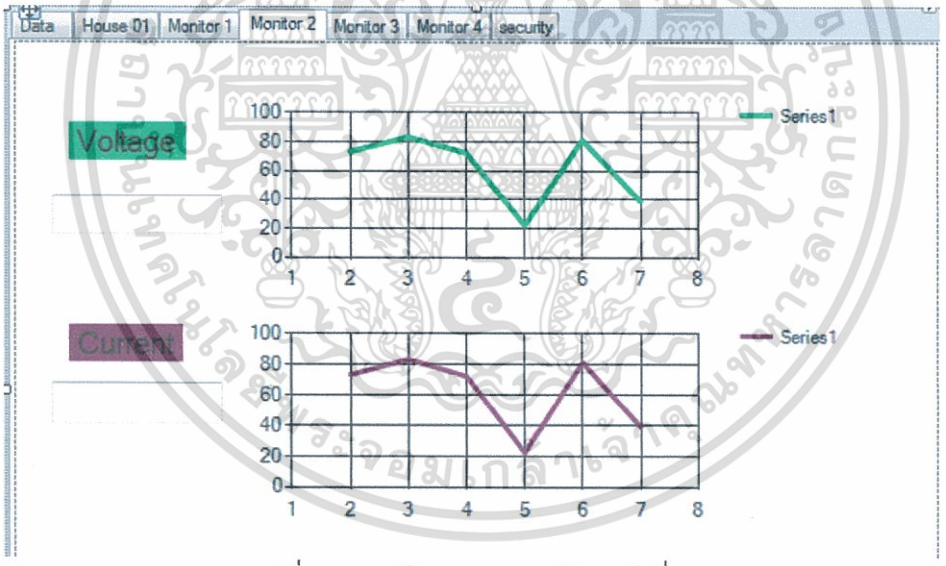
ค่า volumn ในตาราง tbstation	คำสั่งที่ส่งผ่าน Zigbee
0 – 8	A
8 – 16	B
16 – 24	C
24 – 32	D
32 – 40	E
40 – 48	F
48 – 54	G
54 – 62	H
62 – 70	I
70 – 78	J
78 – 86	K
86 – 92	L
92 – 100	M

1) หน้าแสดงกราฟ

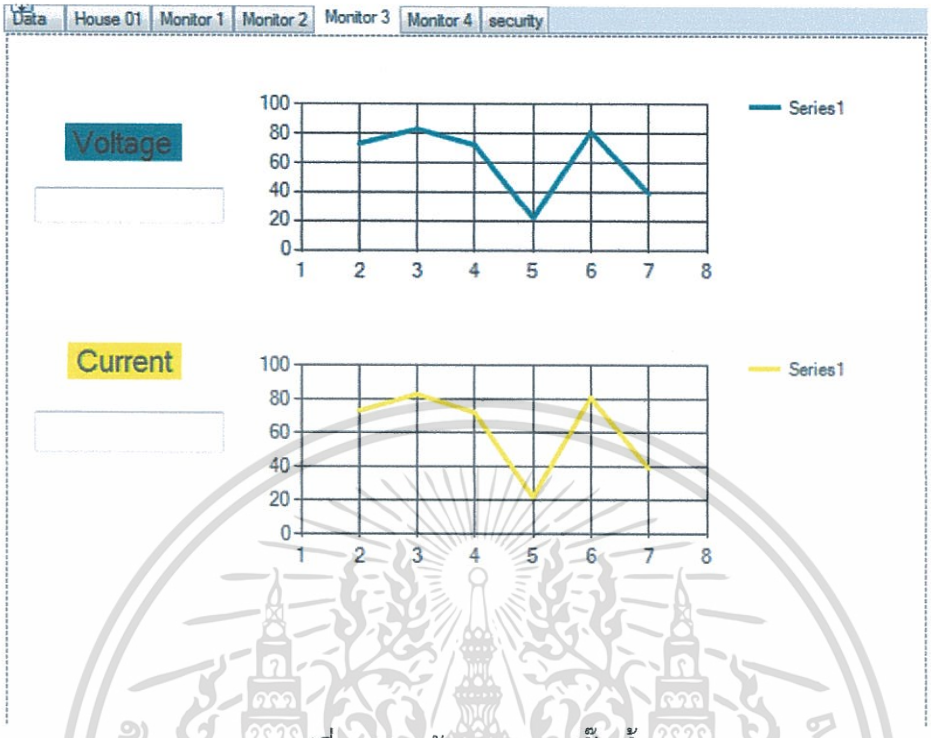
หน้าแสดงกราฟประกอบไปด้วยสามหน้า จะแสดงกราฟกระแสและแรงดันเทียบกับเวลาแบบเรียลไทม์ทุกวินาที เวลาย้อนหลัง 7 วินาที พร้อมทั้งแสดงค่ากระแสและแรงดันล่าสุด มีทั้งหมดสามอุปกรณ์ พัดลมตัวที่ 1 ดังรูปที่ 3.42 และ พัดลมตัวที่ 2 ดังรูปที่ 3.43 ปั๊มน้ำ ดังรูปที่ 3.44 ทั้งสามรูปจะพบว่ากราฟที่แสดงอยู่ด้านบนคือกราฟแรงดันเทียบกับช่วงเวลาย้อนหลัง 7 วินาที พร้อมแสดงค่าแรงดันล่าสุดในช่องกราฟที่แสดงอยู่ด้านล่างคือกราฟกระแสเทียบกับช่วงเวลาย้อนหลัง 7 วินาที พร้อมแสดงค่ากระแสล่าสุดในช่อง



รูปที่ 3.42 หน้าแสดงกราฟพัดลมตัวที่ 1



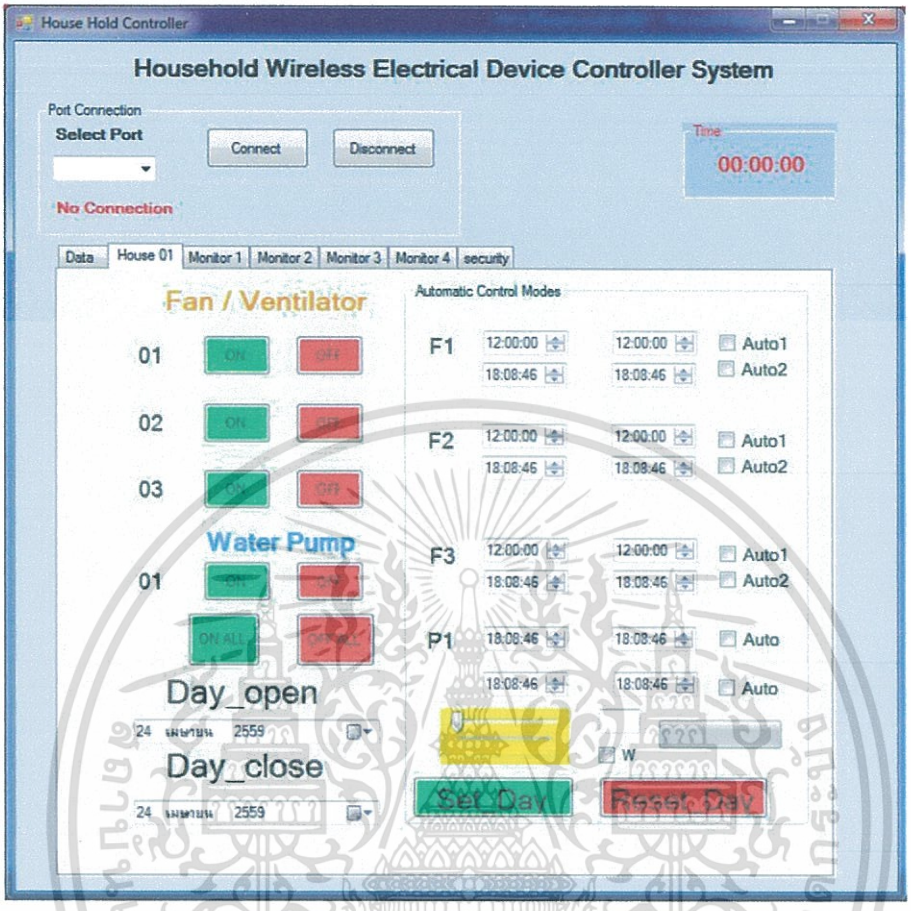
รูปที่ 3.43 หน้าแสดงกราฟพัดลมตัวที่ 2



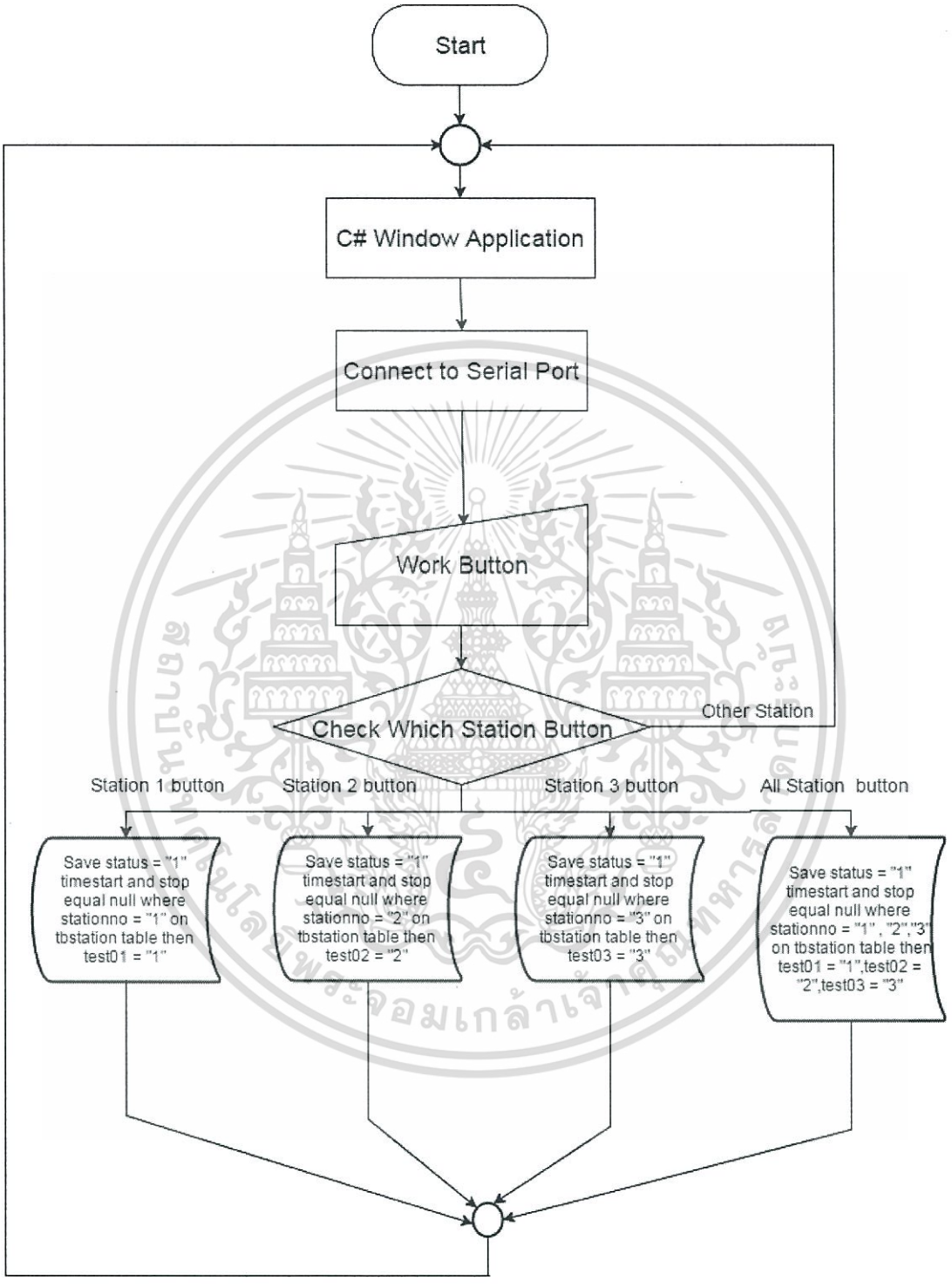
รูปที่ 3.44 หน้าแสดงกราฟปั๊มน้ำ

2) หน้าควบคุมอุปกรณ์ไฟฟ้า

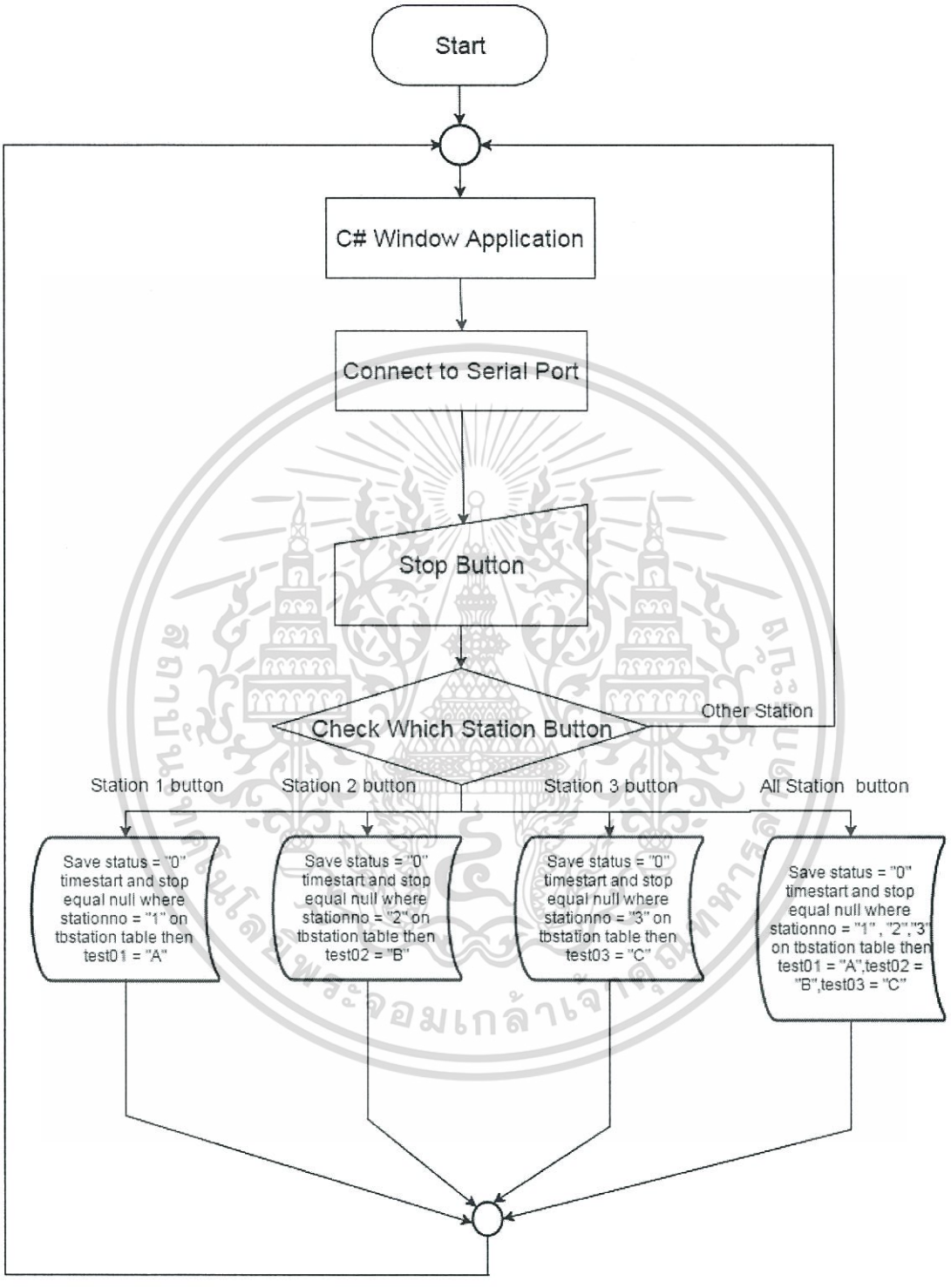
หน้าควบคุมอุปกรณ์ไฟฟ้าดังรูปที่ 3.45 ตัวที่ 1 – 3 เป็นพัดลม ตัวที่ 4 เป็น ปั๊มน้ำ มีลักษณะการควบคุมการทำงานอุปกรณ์ไฟฟ้าคล้ายกับหน้าเว็บดังนี้ ตั้งวันทำงานและวันหยุดทำงาน พัดลมควบคุมเปิดปิดตั้งเวลา เปิดปิด และเปิดปิดทั้งหมด ส่วนปั๊มน้ำจะควบคุมอัตราไหล 0 – 100% แต่ต่างตรงที่ในโปรแกรม c# หน้าควบคุมจะอยู่ภายในหน้าเดียว หน้าควบคุมมีลักษณะดังรูป ซึ่งจะมี flow chart แสดงการทำงานเมื่อ คลิกปุ่มต่างๆ ดังต่อไปนี้ flow chart การทำงานของการคลิกปุ่ม ON แต่ละตัวและทุกตัวดังรูปที่ 3.46 flow chart การทำงานโปรแกรมเมื่อทำการคลิกปุ่ม Off ทั้งแบบ 1 ตัวและทุกตัวดังรูปที่ 3.47 flow chart การทำงานโปรแกรมของการเลื่อน Tracker bar เพื่อควบคุมระดับแรงดันดังรูปที่ 3.48



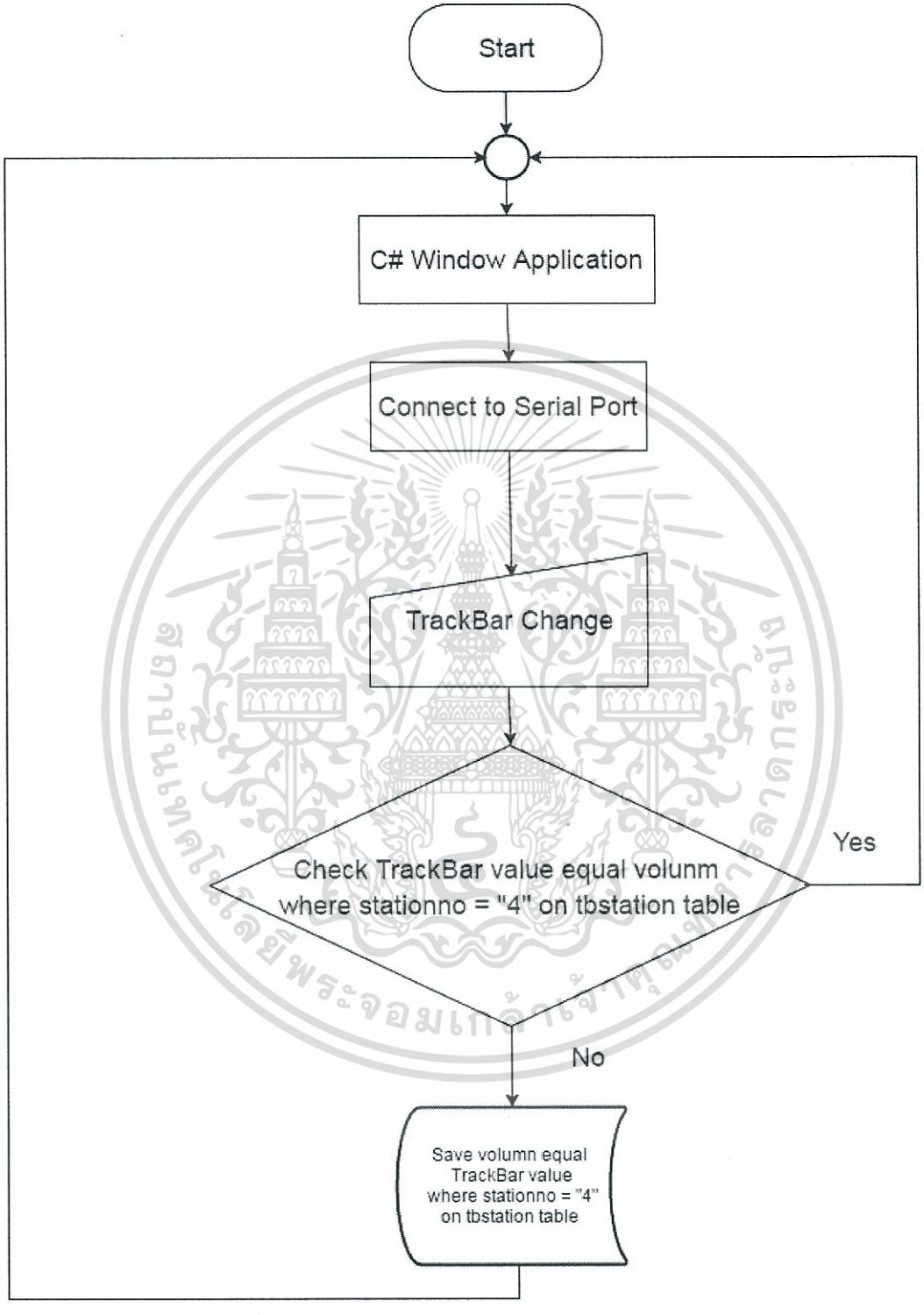
รูปที่ 3.45 หน้าควบคุมอุปกรณ์ไฟฟ้า



รูปที่ 3.46 Flow Chart การทำงานของการคลิกปุ่ม ON แต่ละตัวและทุกตัว



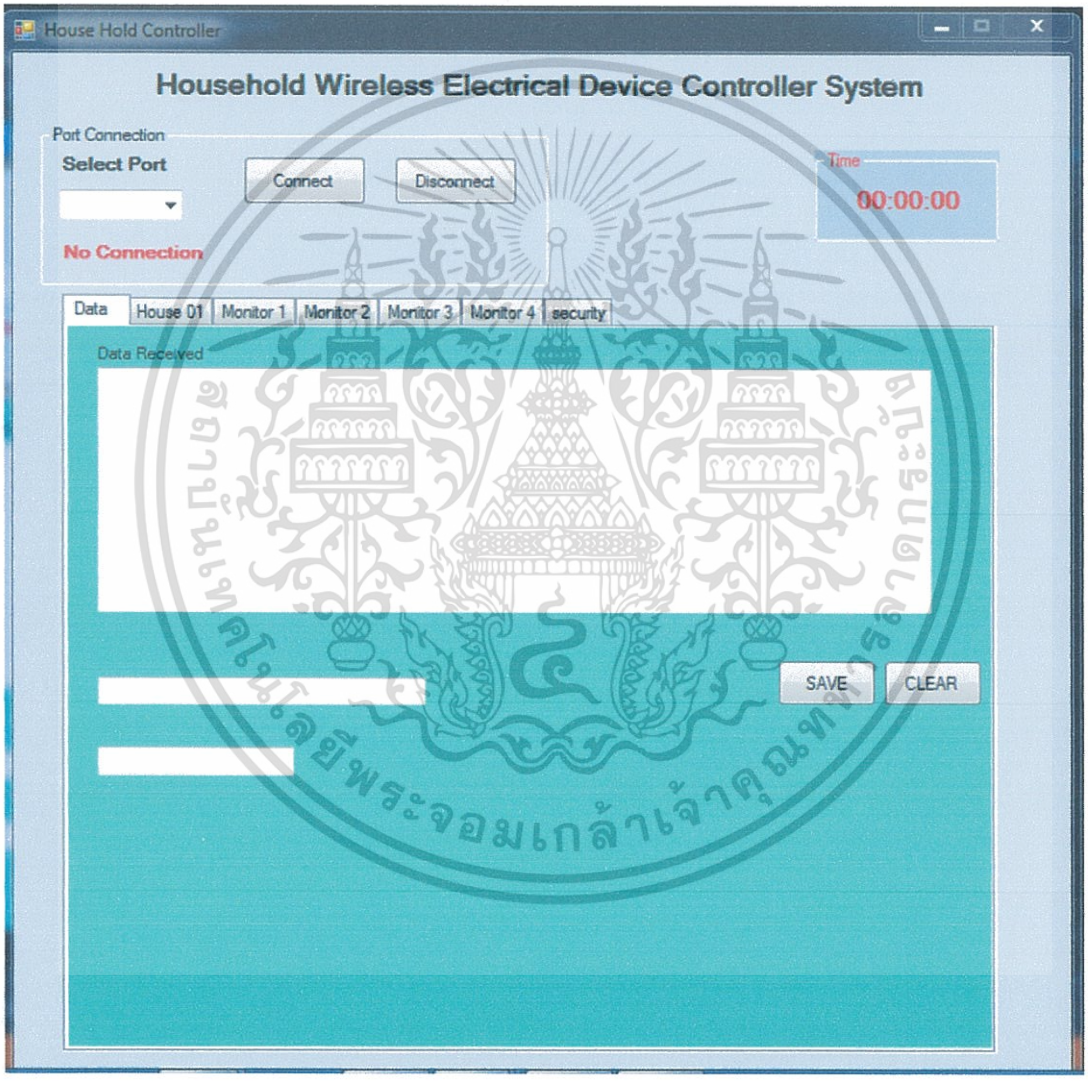
รูปที่ 3.47 Flow Chart การทำงานโปรแกรมเมื่อทำการคลิกปุ่ม Off ทั้งแบบ 1 ตัวและทุกตัว



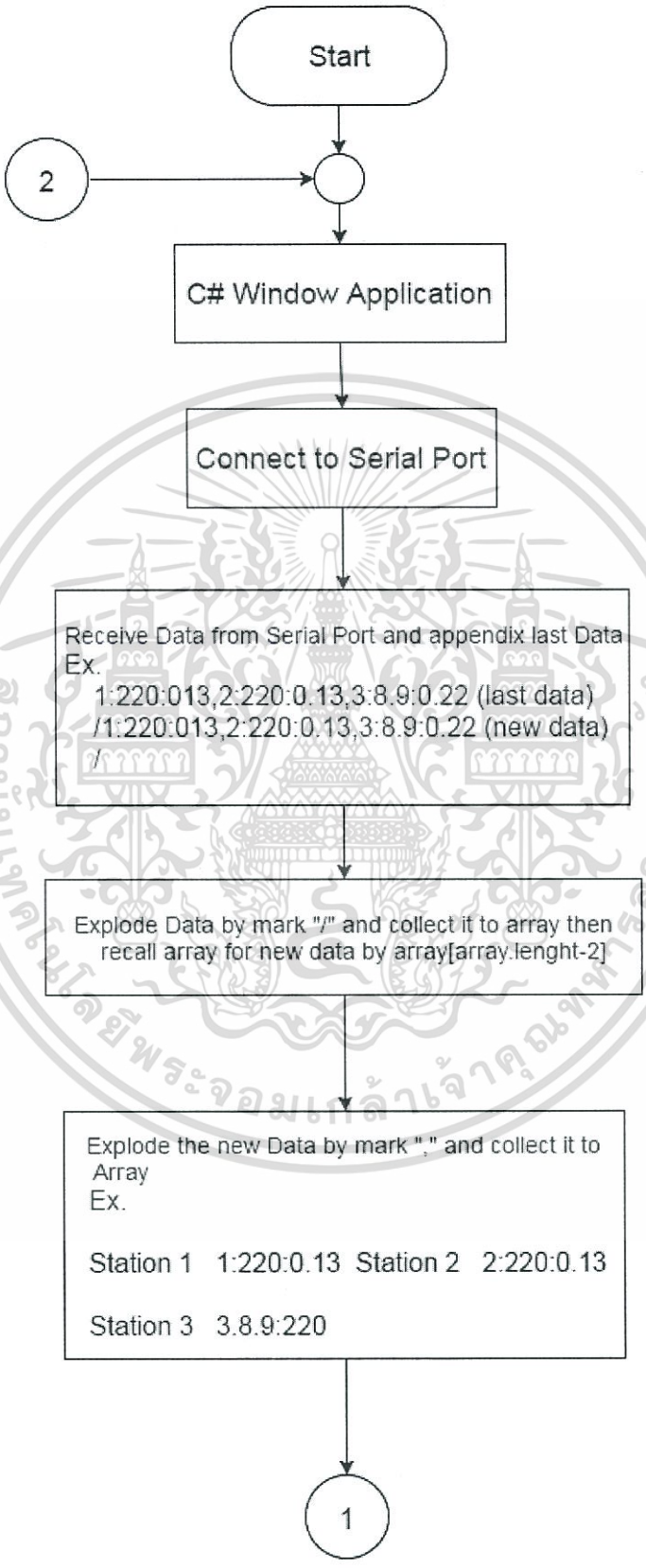
รูปที่ 3.48 Flow Chart การทำงานโปรแกรมของการเลื่อน Tracker bar เพื่อควบคุมระดับแรงดัน

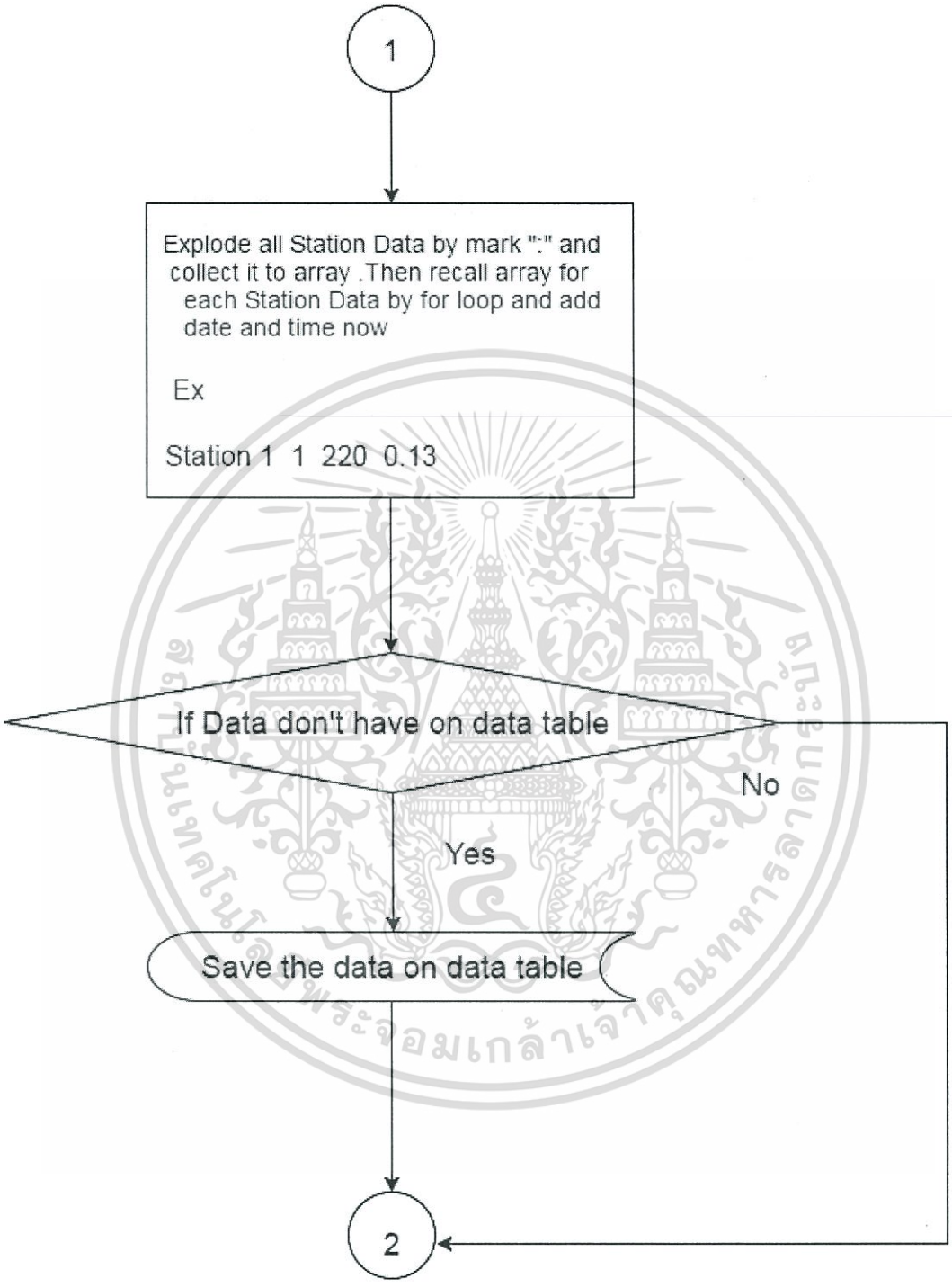
3) หน้าแสดงข้อมูลกระแสและแรงดัน

หน้านี้ดังรูปที่ 3.49 จะแสดงข้อมูลกระแสและแรงดันของอุปกรณ์ไฟฟ้าที่รับมาจาก Zigbee ผ่าน Serial Port ในรูปแบบดังนี้ 1:220:0.13,2:220:0.13,3:8.9:0.23 พัดลมตัวที่ 1(1):แรงดัน:กระแส, พัดลมตัวที่ 2(2) :แรงดัน:กระแส, ปั๊มน้ำ(3) :แรงดัน:กระแส หลังจากแสดงจะนำค่าต่างเหล่านี้ไปจัดเก็บในตาราง data มี flow chart แสดงการรับและจัดเก็บข้อมูลดังรูปที่ 3.50



รูปที่ 3.49 หน้าแสดงข้อมูลกระแสและแรงดัน





รูปที่ 3.50 Flow Chart แสดงการรับค่าและบันทึกลง Database

3.2 เครื่องมือที่ใช้ในการทดลอง

3.2.1 โมดูลสื่อสารไร้สายซิกบี หรือ Xbee Series 2 (ZB) 2 ตัว ใช้สำหรับหรับส่งข้อมูล

3.2.2 รีเลย์โมดูล 4 แชนแนล

3.2.3 ไมโครคอนโทรลเลอร์ Arduino Uno R3 จำนวน 3 ตัว

3.2.4 Xbee Shield 1 ตัว

3.2.5 Xbee Dongle และสาย USB 1 ชุด

3.2.5 Ventilator 220VAC 0.14A SUNON DP200A จำนวน 3 ตัว

3.2.6 มอเตอร์ปั๊มน้ำ จำนวน 1 ตัว

3.2.7 Flow Rate Sensor รุ่น SEN02141B จำนวน 1 ตัว

3.3 วิธีการจัดเก็บผลการทดลอง

3.3.1 ทดสอบการรับส่งค่าระหว่างซิกบีโมดูล

3.3.2 ทดสอบการสั่งงานเปิด-ปิดอุปกรณ์หลายตัวโดยรับส่งข้อมูลผ่าน Xbee

3.3.3 การทดสอบมอเตอร์ปั๊มน้ำด้วยวงจร Phase Lock Loop

3.3.4 การทดสอบวงจรรวมที่ใช้เป็นวงจรแปลงสัญญาณดิจิตอลเป็นอนาล็อก

3.3.5 การทดสอบเซ็นเซอร์วัดอัตราการไหลของน้ำ

3.3.6 การทดสอบเซ็นเซอร์วัดกระแสและแรงดันของมอเตอร์ปั๊มน้ำ

3.3.7 การทดสอบการวัดและป้อนความถี่ด้วยไมโครคอนโทรลเลอร์ ARDUINO

3.3.8 ทดสอบควบคุมอุปกรณ์ไฟฟ้า

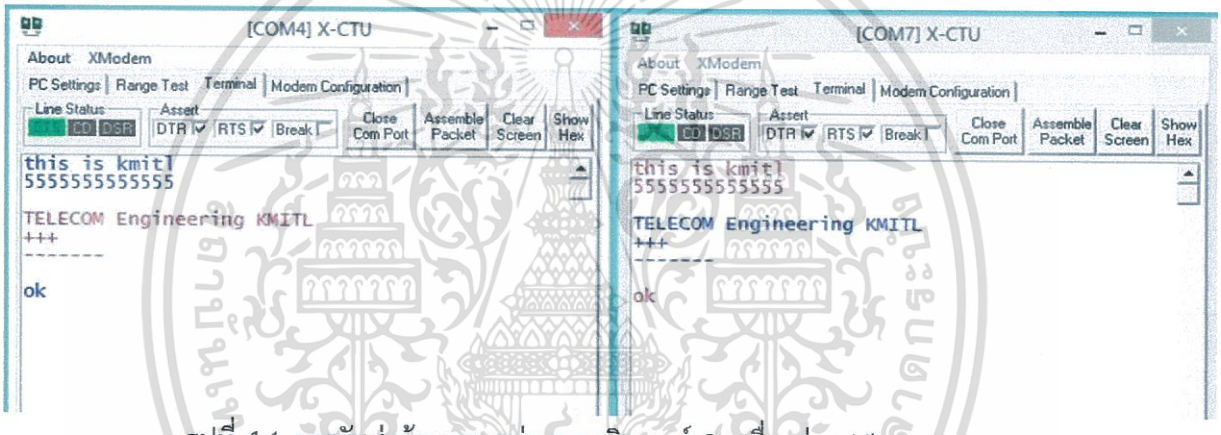
3.3.9 ทดสอบพล็อตกราฟ

บทที่ 4

ผลการทดลอง

4.1 ผลการทดสอบการรับส่งค่าระหว่างซิกบีโมดูล

ทำการทดสอบการรับส่งข้อมูลระหว่างคอมพิวเตอร์ 2 เครื่องผ่านการสื่อสารไร้สายโดยใช้ Xbee ทำการพิมพ์ข้อมูลลงในโปรแกรม X-CTU (ในรูปใช้เวอร์ชันเก่า) ซึ่งผลการทดลองแสดงได้ดังรูปที่ 4.1 เมื่อทำการพิมพ์ข้อความใน Terminal ที่ฝั่งหนึ่งก็จะปรากฏอีกฝั่งด้วยเช่นกัน



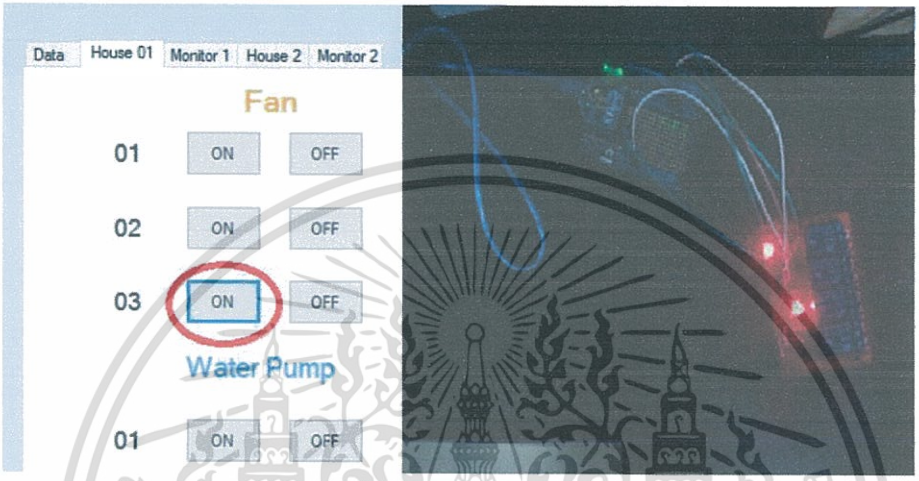
รูปที่ 4.1 การรับส่งข้อมูลระหว่างคอมพิวเตอร์ 2 เครื่องผ่าน Xbee

จากรูปที่ 4.1 ข้อมูลตัวอักษรสีน้ำเงินคือข้อมูลที่พิมพ์และต้องการส่ง ส่วนข้อมูลตัวอักษรสีแดงคือข้อมูลที่ได้รับจากคอมพิวเตอร์อีกเครื่องหนึ่ง

4.2 ผลการทดสอบสั่งงานเปิด-ปิดอุปกรณ์หลายตัวโดยรับส่งข้อมูลผ่าน Xbee

ในการทดสองการสั่งเปิดปิด ได้เขียนโค้ดใน Arduino เพื่อควบคุมอุปกรณ์ไฟฟ้าหลายๆตัว ในที่นี้แทนด้วย รีเลย์ 4 Channel แต่ละช่องแทนด้วยอุปกรณ์ 1 ตัว 3 ตัวแรกแทนพัดลมระบายอากาศ และอีกตัวแทนปั๊มน้ำ 1 ตัว สำหรับ 1 โรง โดย GUI ที่ใช้สั่งการทำงานเขียนบนโปรแกรม MS Visual Studio 2013 โดยมีการทำงานคือ ปุ่ม on ส่งเลข 1-4 และปุ่ม Off จะส่ง A, B, C และ D ไปทาง Serial Port ที่โปรแกรมเลือกเชื่อมต่ออยู่ โดยส่งออกไปผ่าน Xbee Coordinator AT ที่ต่อกับ Mini Dongle จากนั้นข้อมูลจะไปถึงฝั่งรับ ซึ่งมี Xbee ตัวรับ Router AT ต่อด้วย Xbee shield อยู่บนตัว Arduino เมื่อได้รับค่าเลข “3” ที่ส่งมา (จะต้องเอาไปเทียบกับ ASCII ซึ่ง 3=51 ใน DEC และต้องกำหนดที่โค้ดที่รันอยู่ใน Arduino)

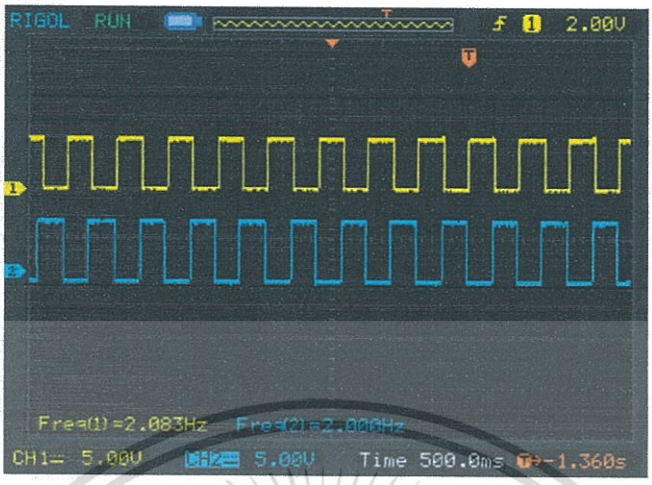
จากรูปที่ 4.2 จะเห็นได้ว่ารีเลย์ช่อง 3 มีไฟติดอยู่ หมายถึง Arduino ได้รับค่า 3 เข้ามาและมันจะทำการสั่งให้รีเลย์ตัวนั้นติด อุปกรณ์ที่ต่อกับรีเลย์ตัวนั้นก็จะทำงานได้อัตโนมัติ ในขณะเดียวกันถ้าส่ง 1 ก็หมายความว่า ให้อุปกรณ์ตัวที่ 1 ทำงาน หรือส่ง A ก็หมายความว่าให้ปิดอุปกรณ์ตัวที่ 1



รูปที่ 4.2 การสั่งเปิดปิดอุปกรณ์ไฟฟ้าผ่านตัว Relay 4 Channel

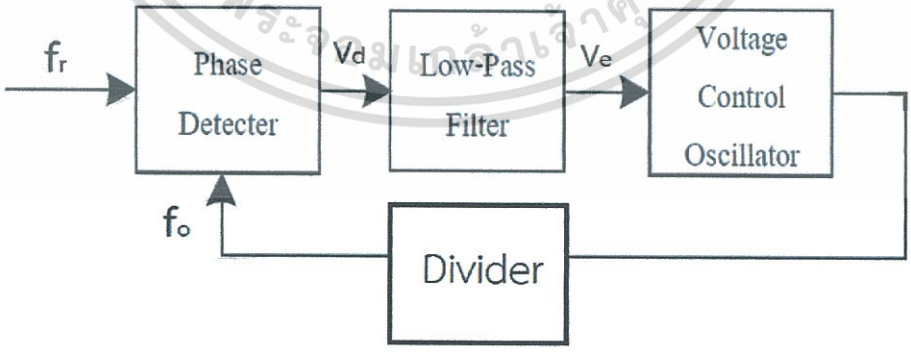
4.3 ผลการทดสอบมอเตอร์ปั๊มน้ำด้วยวงจร Phase Lock Loop

การทดลองโดยการต่อออสซิลโลสโคป Ch.1 เข้าที่ขา 14 ของ CD4046 เพื่อวัดสัญญาณอินพุตที่ป้อนจากไมโครคอนโทรลเลอร์ Arduino โดยจะป้อนความถี่ตามความถี่ที่วัดได้จากขา 6 ของ CD4040 และต่อออสซิลโลสโคป Ch.2 เข้าที่ขา 6 ของ CD4040 เพื่อวัดความถี่ของน้ำที่ไหลโดยเราจะนำเซ็นเซอร์อัตราไหลมาต่อเข้ากับท่อมอเตอร์ปั๊มน้ำด้านน้ำออกเพื่อนำสัญญาณพัลส์ของเซ็นเซอร์นี้ป้อนเข้าสู่วงจรหารความถี่จึงได้สัญญาณความถี่เพื่อไปเปรียบเทียบกับสัญญาณความถี่อินพุต และเราสามารถทำการปรับเปอร์เซ็นต์การทำงานของมอเตอร์ปั๊มน้ำได้โดยใช้วงจรรวมที่ใช้เป็นวงจรแปลงสัญญาณดิจิตอลเป็นแอนะลอกเพื่อนำค่าแรงดันเอาต์พุตระดับต่างๆไปป้อนให้กับวงจรเฟสล็อคลูปโดยวงจรเฟสล็อคลูปจะอยู่ในสภาวะล็อคตลอดเวลาทำให้อัตราการไหลของน้ำคงที่ความถี่ของอินพุต Ch.1 จะเข้าใกล้ความถี่หลังจากผ่านวงจรหารความถี่ Ch.2 ดังแสดงในรูปที่ 4.3



รูปที่ 4.3 สัญญาณอินพุต (Ch.1) และ เอาต์พุตจาก CD 4040 (Ch.2)

ระบบเฟสล็อคลูปจะสามารถทำงานอย่างมีเสถียรภาพในช่วงกว้างช่วงหนึ่งที่เรียกว่าเฟสล็อคลูปแบนด์วิดท์ (PLL bandwidth) การทำงานของระบบในช่วงนี้ พารามิเตอร์ของระบบจะทำงานอย่างสัมพันธ์กันภายใต้เงื่อนไขที่ต้องให้ความถี่อินพุตทั้งสองของเฟสดีเทคเตอร์เท่ากันเสมอ การทำงานที่สัมพันธ์กันดังกล่าวนี้จะอธิบายจากบล็อกไดอะแกรมการทำงานของ PLL ในสภาวะล็อคดังรูปที่ 4.4 และแสดงค่าความสัมพันธ์ระหว่างแรงดันกับความถี่ภายในวงจรเฟสล็อคลูปดังตารางที่ 4.1



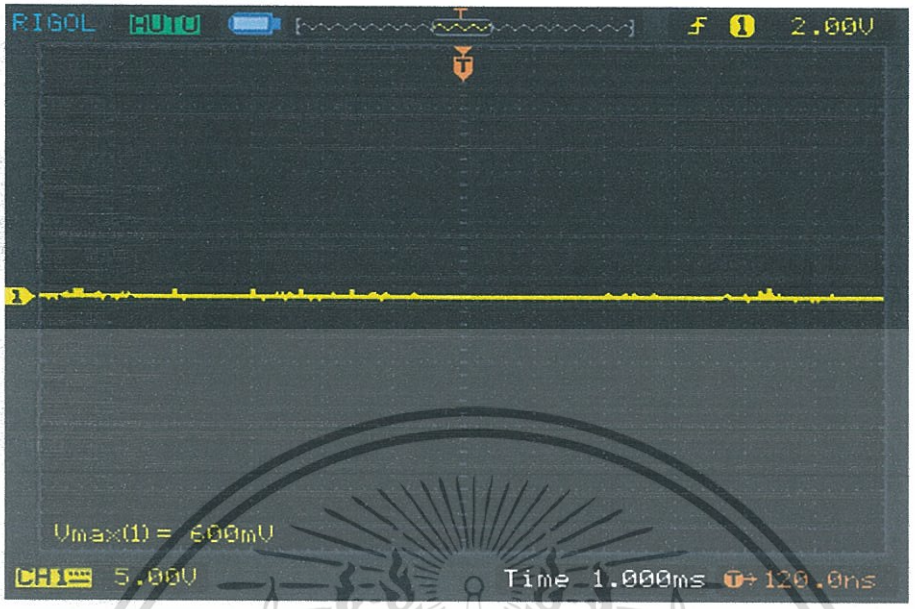
รูปที่ 4.4 บล็อกไดอะแกรมการทำงานของ PLL ในสภาวะล็อค

ตารางที่ 4.1 ความสัมพันธ์ระหว่างแรงดันกับความถี่ภายในวงจรเฟสล็อคลูป

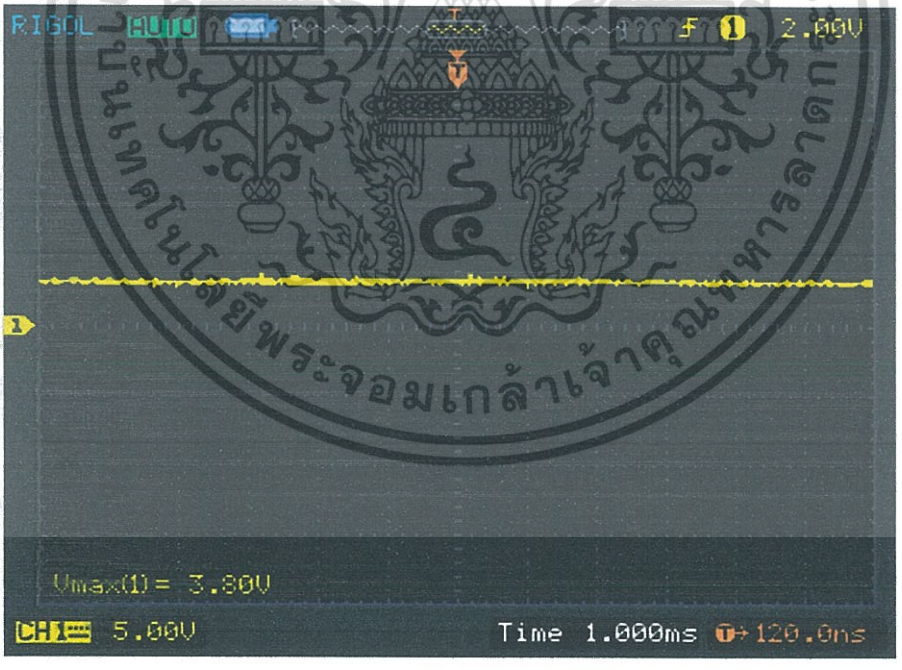
Freq (Hz)	f ขา 10 CD4040 (Hz)	V ขา6 CD4040 (V)	f ขา 6 CD4040 (Hz)	Vd (V)	fd (Hz)	Ve (V)	Fe (Hz)
2	15.5	5.28	1.984	2.88	1.953	0.72	1.984
2	14.5	5.28	1.812	2.88	1.812	0.72	1.812
2	13.5	5.28	1.667	2.88	1.689	0.72	1.667
2	12.5	5.28	1.563	2.88	1.563	0.88	1.623
2	11.5	5.28	1.437	2.88	1.437	0.96	1.437
2	10.5	5.28	1.302	2.96	1.302	1.04	1.33
2	9.5	5.28	1.179	3.04	1.179	1.12	1.179
2	8.5	5.28	1.068	3.12	1.059	1.2	1.059
2	7.5	5.28	0.9328	3.12	0.9328	1.36	0.9398
2	6.5	5.28	0.8117	3.28	0.8117	1.6	0.8117
2	5.5	5.28	0.6868	3.36	0.6868	1.68	0.6757
2	4.5	5.28	0.5618	3.52	0.5556	2.16	0.5618
2	3.5	5.28	0.4386	3.68	0.4348	2.56	0.4348
2	2.5	5.28	0.3125	4.08	0.3125	3.2	0.3125
2	2	5.28	0.25	4.32	0.25	3.68	0.25
2	1.5	5.28	0.188	4.64	0.1838	4.24	0.1894
2	1	5.28	0.125	4.88	0.1238	4.8	0.1225

4.4 ผลการทดสอบวงจรรวมที่ใช้เป็นวงจรแปลงสัญญาณดิจิตอลเป็นอนาล็อก

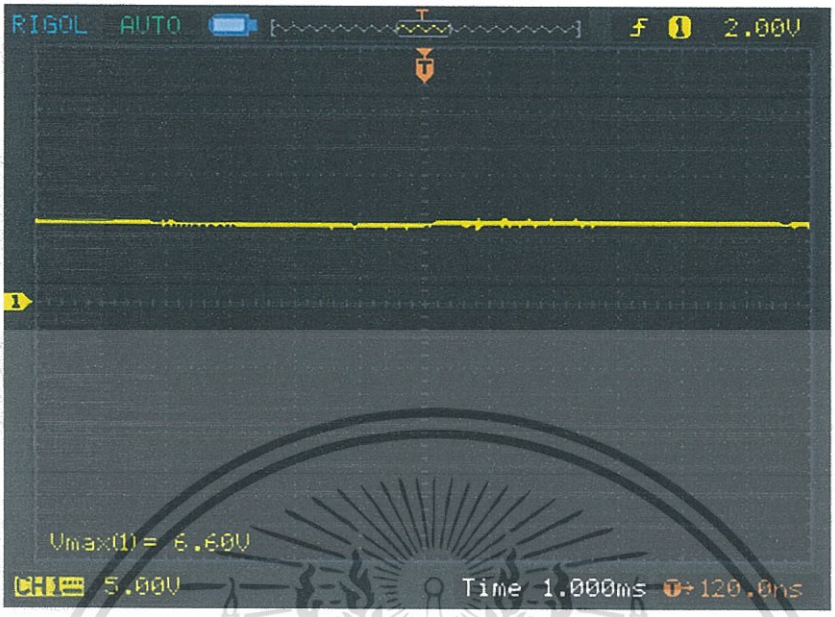
การทดสอบวงจรโดยการปรับเปอร์เซ็นต์การทำงานของมอเตอร์ปั๊มน้ำผ่านหน้าต่าง C# หรือหน้าเว็บ ซึ่งวงจรนับมีการนับค่า จาก 0000 0000 นับขึ้นไปที่ค่า 1111 1111 และแรงดันเอาต์พุตอนาล็อกจะเพิ่มขึ้นจาก 0 V ขึ้นไปจนเกือบ +12 V สามารถใช้ออสซิลโลสโคปวัดค่าได้ดังแสดงในรูปที่ 4.5 (ก) (ข) (ค) (ง) (จ) และแสดงค่าแรงดันเอาต์พุตที่วัดโดยออสซิลโลสโคปและมิเตอร์ดังในตารางที่ 4.2



รูปที่ 4.5 (ก) ค่าแรงดันเอาต์พุต 0.6 V

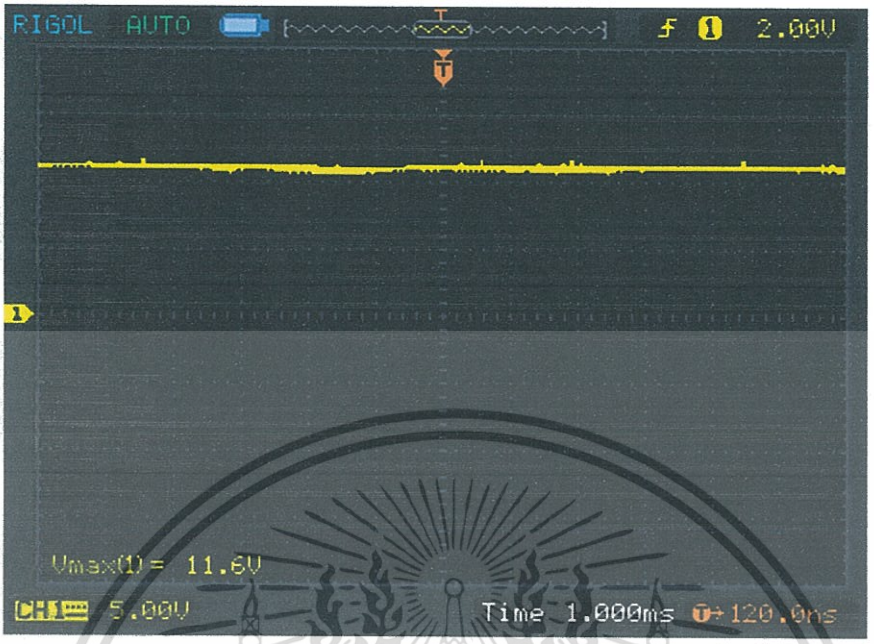


รูปที่ 4.5 (ข) ค่าแรงดันเอาต์พุต 3.8 V



รูปที่ 4.5 (ค) ค่าแรงดันเอาต์พุต 6.6 V

รูปที่ 4.5 (ง) ค่าแรงดันเอาต์พุต 9.6 V

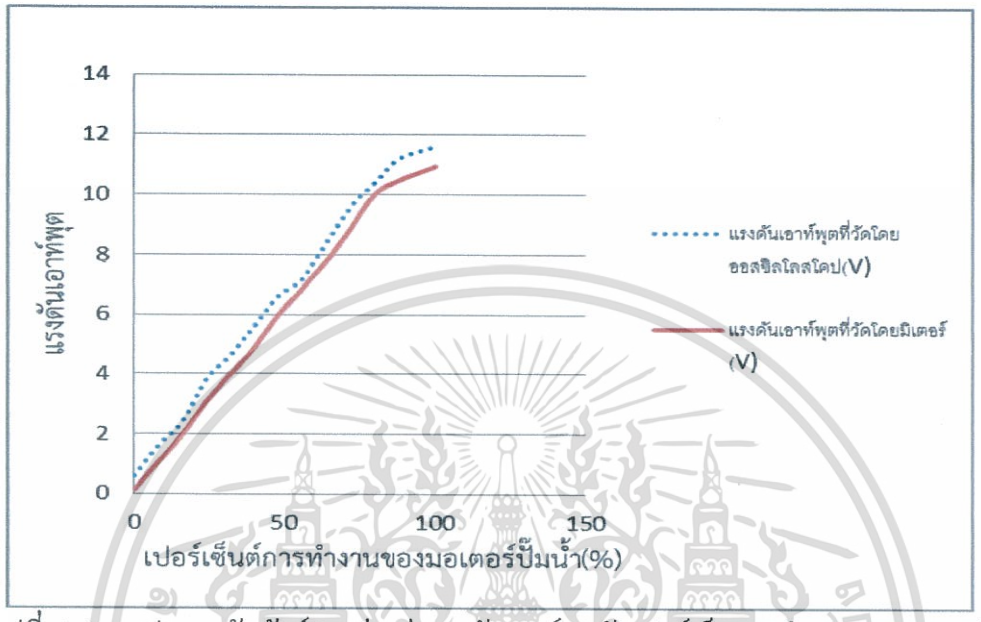


รูปที่ 4.5 (จ) ค่าแรงดันเอาต์พุต 11.6 V

ตารางที่ 4.2 ค่าแรงดันเอาต์พุตที่วัดโดยออสซิลโลสโคปและมิเตอร์ที่ระดับการทำงานต่างๆ

เปอร์เซ็นต์การทำงานของมอเตอร์ปั๊มน้ำ (%)	แรงดันเอาท์พุตที่วัดโดยออสซิลโลสโคป (V)	แรงดันเอาท์พุตที่วัดโดยมิเตอร์ (V)
0	0.6	0.13
8	1.6	1.04
16	2.4	1.96
24	3.8	3.05
32	4.6	3.96
40	5.6	4.86
48	6.6	5.96
56	7.2	6.87
64	8.4	7.78
72	9.6	8.88
80	10.4	10.01
88	11.2	10.47
100	11.6	10.94

จากผลการทดลองวัดค่าแรงดันเอาต์พุตมีกราฟความสัมพันธ์ระหว่างค่าแรงดันเอาต์พุตที่วัดโดยออสซิลโลสโคปและมิเตอร์แสดงดังรูปที่ 4.6



รูปที่ 4.6 กราฟความสัมพันธ์ระหว่างค่าแรงดันเอาต์พุตกับเปอร์เซ็นการทำงานของมอเตอร์

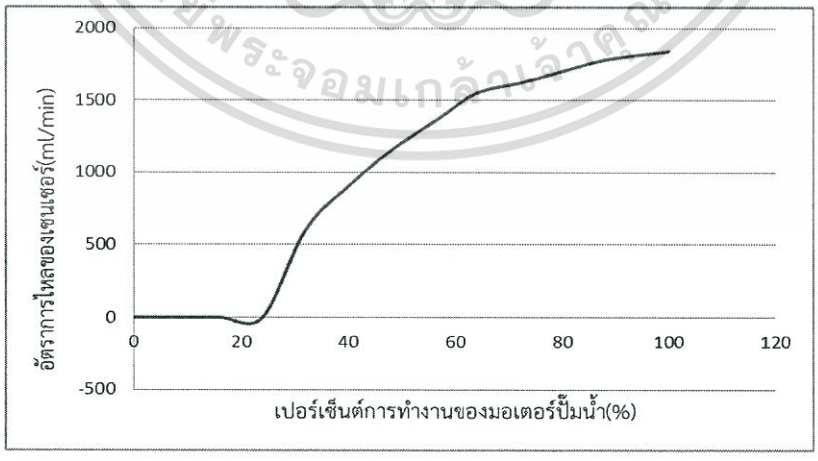
4.5 ผลการทดสอบเซ็นเซอร์วัดอัตราการไหลของน้ำ

การทดสอบวงจรโดยการปรับเปอร์เซ็นต์การทำงานของมอเตอร์ปั๊มน้ำผ่านหน้าต่าง C# หรือหน้าเว็บ เซ็นเซอร์จะวัดค่าสัญญาณจากรอบการหมุนของน้ำที่ไหลผ่านใบพัดโดยเราเก็บผลระดับอัตราการไหลในหน่วย ml/min โดยใช้กระบอกตวง ดังแสดงความสัมพันธ์ระหว่างค่าเปอร์เซ็นต์การทำงานของมอเตอร์ปั๊มน้ำกับอัตราการไหลของเซ็นเซอร์ในตารางที่ 4.3

ตารางที่ 4.3 ความสัมพันธ์ระหว่างค่าเปอร์เซ็นต์การทำงานของมอเตอร์ปั๊มน้ำกับอัตราการไหลของเซ็นเซอร์

เปอร์เซ็นต์การทำงานของมอเตอร์ปั๊มน้ำ (%)	อัตราการไหลของเซ็นเซอร์ (ml/min)
0	0
8	0
16	0
24	0
32	600
40	900
48	1150
56	1350
64	1550
72	1620
80	1700
88	1780
100	1840

จากค่าอัตราการไหลในตารางที่ 4.3 จะช่วยให้เรารู้ถึงอัตราส่วนในการผสมสารสำหรับการเลี้ยงสัตว์เช่น วิตามิน หรือยาบำรุง กับน้ำให้มีอัตราส่วนที่เหมาะสม โดยมีกราฟความสัมพันธ์ระหว่างเปอร์เซ็นต์การทำงานของมอเตอร์ปั๊มน้ำกับอัตราการไหลของเซ็นเซอร์แสดงดังรูปที่ 4.7



รูปที่ 4.7 กราฟความสัมพันธ์ระหว่างเปอร์เซ็นต์การทำงานมอเตอร์ปั๊มน้ำกับอัตราการไหลของน้ำ

4.6 ผลการทดสอบเซ็นเซอร์วัดกระแสและแรงดันของมอเตอร์ปั๊มน้ำ

การทดสอบเซ็นเซอร์โดยการปรับเปอร์เซ็นต์การทำงานของมอเตอร์ปั๊มน้ำผ่านหน้าต่าง C# หรือหน้าเว็บเพื่อวัดค่ากระแสและแรงดันของมอเตอร์ปั๊มน้ำโดยเปรียบเทียบกับค่าที่วัดกับมิเตอร์ดังแสดงในตารางที่ 4.4

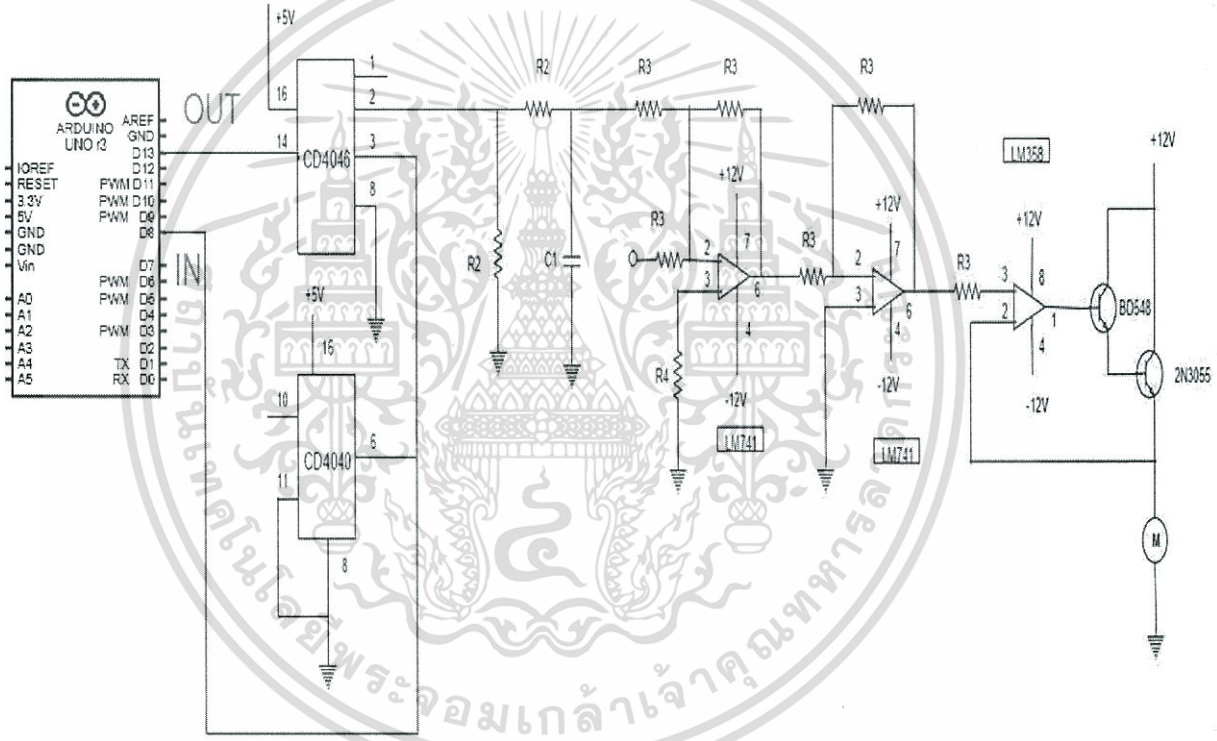
ตารางที่ 4.4 ความสัมพันธ์ระหว่างเปอร์เซ็นต์การทำงานของมอเตอร์ปั๊มน้ำกับค่ากระแสและแรงดันของมอเตอร์ปั๊มน้ำ

เปอร์เซ็นต์การทำงานของมอเตอร์ปั๊มน้ำ (%)	กระแสที่วัดจากเซ็นเซอร์ (A)	กระแสที่วัดจากมิเตอร์ (A)	แรงดันที่วัดจากเซ็นเซอร์ (V)	แรงดันที่วัดจากมิเตอร์ (V)
0	0.04	0	0.1	0.16
8	0.04	0	1.1	1.092
16	0.05	0.029	2.0	2.017
24	0.10	0.098	2.6	2.672
32	0.13	0.121	3.8	3.805
40	0.15	0.146	4.5	4.66
48	0.18	0.179	5.7	5.68
56	0.20	0.208	6.5	6.51
64	0.22	0.237	7.4	7.33
72	0.25	0.272	8.2	8.26
80	0.25	0.273	8.4	8.41
88	0.25	0.274	8.4	8.42
100	0.25	0.275	8.4	8.42

จากค่ากระแสและแรงดันของมอเตอร์ปั๊มน้ำเมื่อวัดเทียบกับมิเตอร์ในตารางที่ 4.4 ค่าที่ได้ค่อนข้างใกล้เคียงกับมิเตอร์ทำให้เราทราบถึงระดับการทำงานของมอเตอร์ปั๊มน้ำได้จากค่ากระแสและแรงดัน

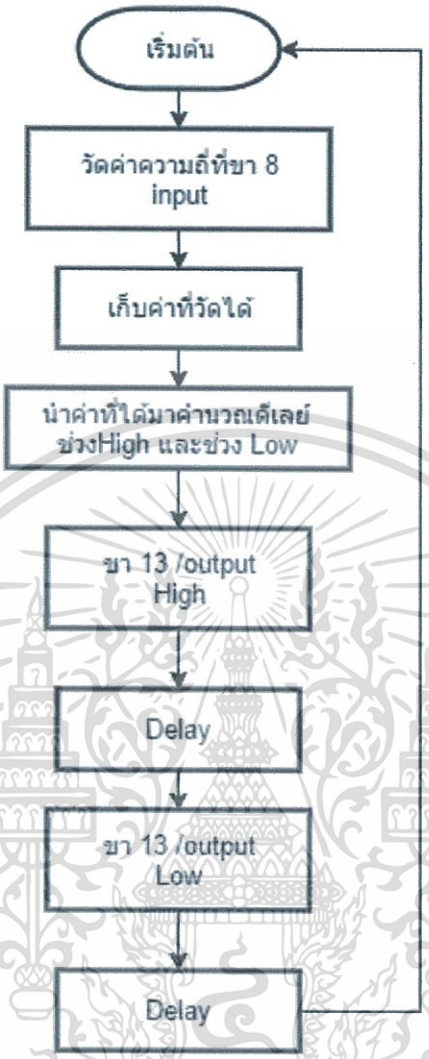
4.7 ผลการทดสอบการวัดและป้อนความถี่ด้วยไมโครคอนโทรลเลอร์ ARDUINO

ใช้ไมโครคอนโทรลเลอร์ตรวจจับความถี่ที่ออกมาจากขา 6 ของ CD4040 ของวงจรเฟสล็อคลูปหลังจากนั้นก็ใช้ไมโครคอนโทรลเลอร์ตัวเดิมป้อนความถี่เข้าที่ขา 14 ของ CD4046 ให้เท่ากับความถี่ที่ตรวจจับได้ทำให้ระบบการจ่ายน้ำเพื่อผสมสารสำหรับการเลี้ยงสัตว์มีอัตราที่คงที่สม่ำเสมอดังแสดงในบล็อกไดอะแกรมรูปที่ 4.8 และมีแผนผังการทำงานของไมโครคอนโทรลเลอร์ดังแสดงในรูปที่ 4.9

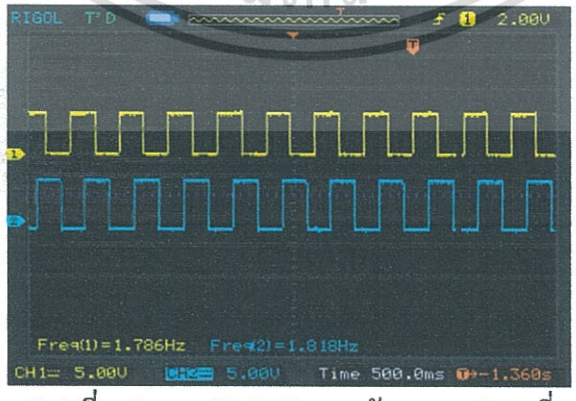


รูปที่ 4.8 การวัดและป้อนความถี่ด้วยไมโครคอนโทรลเลอร์ ARDUINO ในวงจรเฟสล็อคลูป

จากผลการทดสอบเมื่อใช้ออสซิลโลสโคปวัดสัญญาณ Ch1.ที่ขา 14 ของ CD4046 และวัดสัญญาณ Ch2. ที่ขา6 ของ CD4040 ของวงจรเฟสล็อคลูปพบว่าความถี่ของทั้ง 2 สัญญาณใกล้กันจะได้ผลแสดงดังรูปที่ 4.10



รูปที่ 4.9 Flow Chart การทำงานของโปรแกรมการวัดและป้อนความถี่ด้วยไมโครคอนโทรลเลอร์

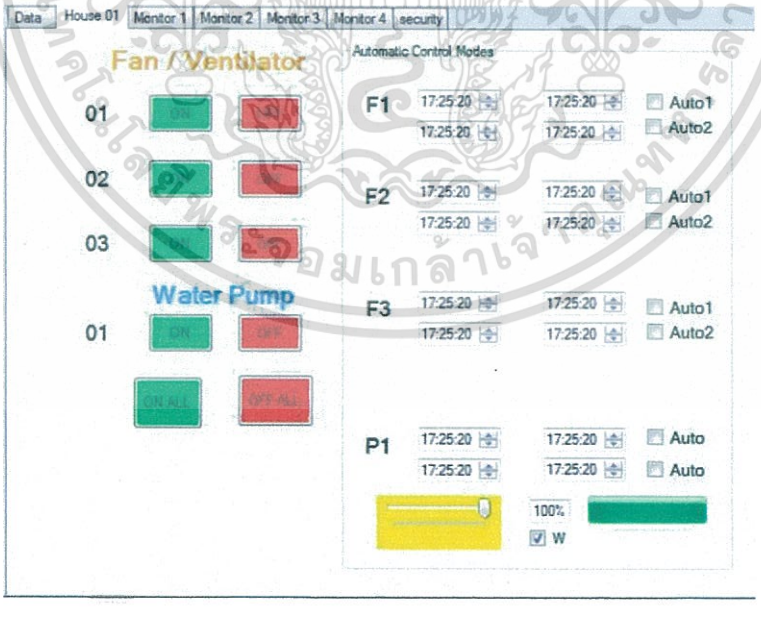


รูปที่ 4.10 สัญญาณ Ch1.ที่ขา6 ของ CD4040 และสัญญาณ Ch2. ที่ขา 14 ของ CD4046

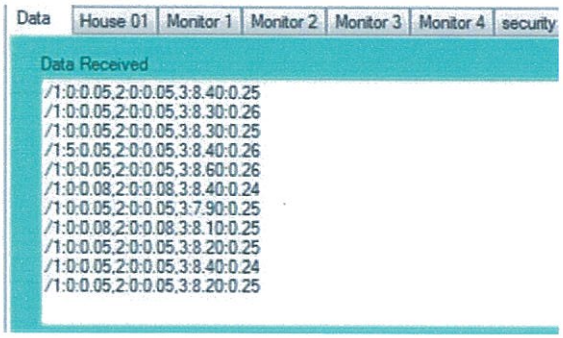
4.8 ทดสอบควบคุมอุปกรณ์ไฟฟ้า

4.1.1 ทดสอบควบคุมปั๊มน้ำ

ทำการทดสอบควบคุมปั๊มน้ำผ่านหน้าเว็บและหน้าโปรเกรม c# เริ่มแรกทำการควบคุมผ่านหน้าโปรแกรม c# โดยทำการปรับ trackbar ไปที่ระดับ 100% ดังรูปที่ 4.11 และปรับเปอร์เซ็นต์หน้าแก้ไขข้อมูล ปรับไปที่ระดับ 100% ดังรูปที่ 4.15 ตามลำดับ แล้วทำการคลิกปุ่ม edit ทั้งการสั่งผ่านโปรแกรม c# และ ผ่านหน้าเว็บ ข้อมูลจะบันทึกลงตารางข้อมูล tbstation ดังรูปที่ 4.15 และโปรแกรม c# จะอ่านข้อมูลจากตารางข้อมูล และประมวลผลส่งคำสั่งผ่าน Serial Port สื่อสาร Zigbee ไปยังไมโครคอนโทรลเลอร์ จะสั่งปั๊มน้ำให้ทำตามคำสั่งที่ได้รับมาในที่นี้ 100% จากนั้นไมโครคอนโทรลเลอร์จะทำการวัดค่า กระแสและแรงดันส่งกลับผ่าน Zigbee สู่โปรแกรม c# แสดงค่าที่รับมาที่ หน้าแสดงข้อมูลกระแสและแรงดัน ดังรูปที่ 4.12 จะสังเกตพบว่าค่าที่รับมาในที่นี้แต่ละบรรทัดประกอบด้วย /1:0:0.05, 2:0:0.05, 3:8.40:0.25 คือ พัดลมตัวที่ 1:แรงดัน:กระแส, พัดลมตัวที่ 2:แรงดัน:กระแส, ปั๊มน้ำ:แรงดัน:กระแส ในที่ที่นี้สังเกตแรงดันและกระแสจะพบว่าค่าจะไม่คงที่อาจจะเกิดจากความผิดพลาดอันเนื่องมาจากการส่งข้อมูลผ่าน Zigbee หรือเป็นที่เซ็นเซอร์วัดแรงดันและกระแสวัดค่าผิดอันเนื่องมาจากการจ่ายไฟเลี้ยงให้เซ็นเซอร์ไม่พอ จากนั้นก็จะเก็บค่าที่ได้พร้อมแยกเป็นแต่ละอุปกรณ์ลงตารางข้อมูล data ดังรูปที่ 4.16 โดยจะทำการแยกข้อมูลแต่ละแถวก่อน และบันทึกข้อมูลในตาราง data ในแต่ละแถวรูปแบบดังนี้ วันที่ เวลา หมายเลขอุปกรณ์ แรงดัน กระแส เช่น 6/4/2559 17:26:31 3 8.2 0.25



รูปที่ 4.11 หน้าควบคุมอุปกรณ์ไฟฟ้า c#



รูปที่ 4.12 หน้าแสดงข้อมูลกระแสและแรงดัน c#

รูปที่ 4.13 หน้าเว็บควบคุมอุปกรณ์ไฟฟ้า

รูปที่ 4.14 หน้าแก้ไขข้อมูลอุปกรณ์ไฟฟ้า

	stationno	name	status	timestart	timestop	timestart2	timestop2	volumn
	1	พัดลม	0					0
	2	พัดลม	0					0
	3	พัดลม	0					0
	4	ปั๊มน้ำ	0					100

รูปที่ 4.15 ข้อมูลในตาราง tbstation

	Date	Time	Stationno	Voltage	Current
	6/4/2559	17:26:29	3	8.500	0.240
	6/4/2559	17:26:31	1	0.000	0.050
	6/4/2559	17:26:31	2	0.000	0.050
	6/4/2559	17:26:31	3	8.200	0.250
	6/4/2559	17:26:34	1	0.000	0.050
	6/4/2559	17:26:34	2	0.000	0.050
	6/4/2559	17:26:34	3	8.600	0.240

รูปที่ 4.16 ข้อมูลในตาราง data

4.1.2 ทดสอบควบคุมพัดลม

4.1.2.1 ทดสอบตั้งเวลาเปิด-ปิด

ทำการทดสอบตั้งเวลาเปิดปิด ผ่านหน้าเว็บควบคุมอุปกรณ์ไฟฟ้าดังรูปที่ 4.17 และหน้าต่างควบคุมอุปกรณ์ไฟฟ้า c# ดังรูปที่ 4.18 โดยตั้งเวลากำหนดให้

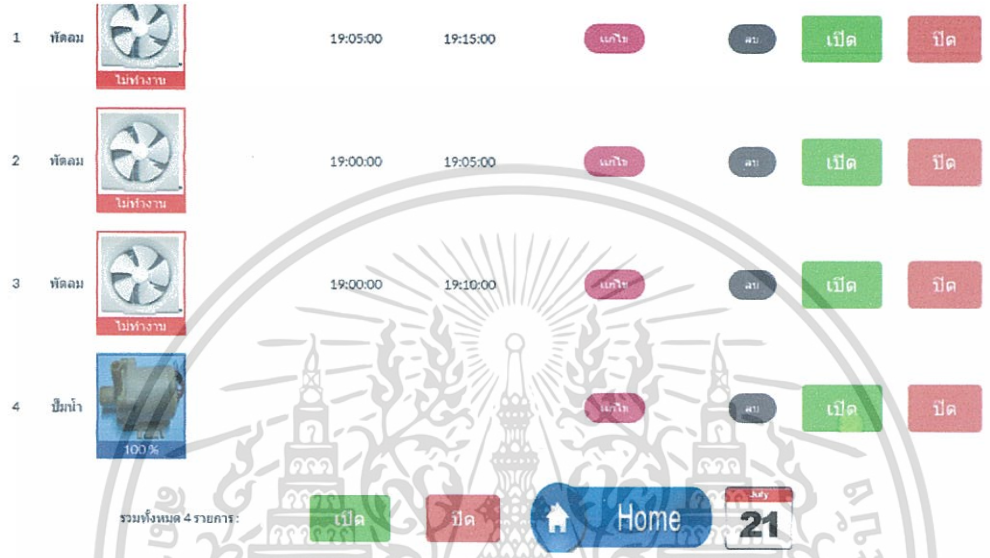
พัดลมตัวที่ 1 เวลาเปิด 19:05:00 น. เวลาปิด 19:15:00 น.

พัดลมตัวที่ 2 เวลาเปิด 19:00:00 น. เวลาปิด 19:05:00 น.

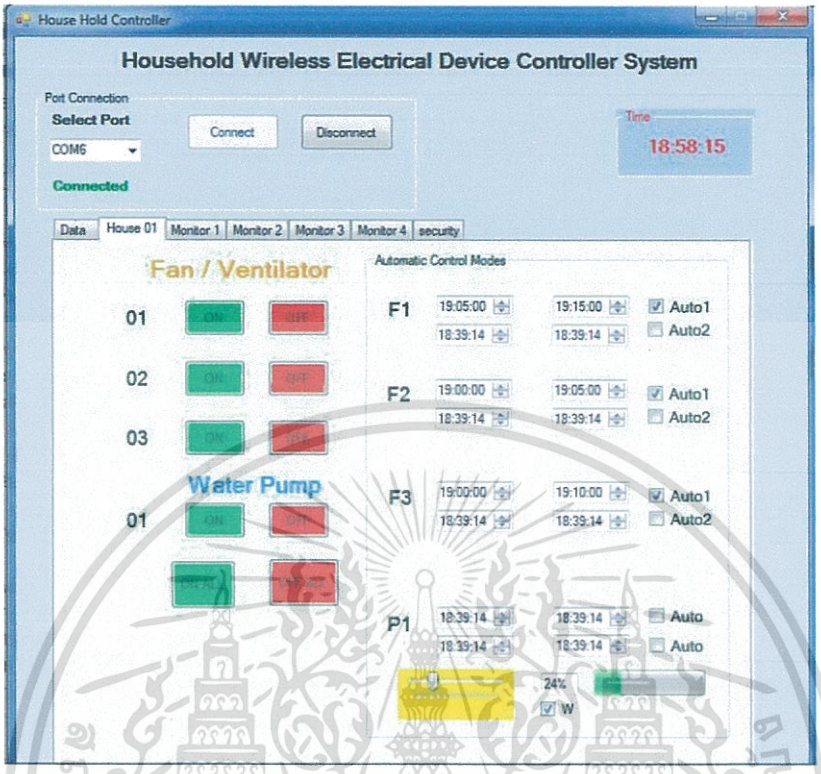
พัดลมตัวที่ 3 เวลาเปิด 19:00:00 น. เวลาปิด 19:10:00 น.

เมื่อทำการตั้งเวลาเปิดปิดแล้วไม่ว่าตั้งในหน้าเว็บหรือหน้าโปรแกรม c# ข้อมูลจะถูกบันทึกลงในตาราง tbstation โปรแกรมc# จะอ่านข้อมูลในตารางนี้แต่ละแถว ประมวลผล และบันทึกค่า status ใหม่ในตาราง เพื่อควบคุมพัดลมแต่ละตัวให้เปิด (ทำงาน status = 1) ปิด (หยุดทำงาน status = 0) ตามช่วงเวลาที่กำหนดไว้ แล้วส่งคำสั่งผ่าน Zigbee ไปให้คอนโทรลเลอร์ควบคุมพัดลม เริ่มแรกก่อนถึงเวลา (19:00:00 น.) ที่ตั้งไว้แต่ละอุปกรณ์ ข้อมูลในตารางดังรูป 4.19 พัดลมทุกตัวหยุดทำงาน (พัดลมทุกตัว status = 0) เมื่อเวลาอยู่ในช่วง 19:00:00 น. ถึง 19:05:00 น. พัดลมตัวที่ 1 หยุดทำงาน พัดลมตัวที่ 2 และ 3 จะทำงาน ดังแสดงในหน้าเว็บดังรูปที่ 4.20 หน้าเว็บจะแสดงรูปภาพพัดลมเคลื่อนไหวสีเขียวในแถวพัดลมตัวที่ 2 และ 3 แล้วข้อมูลในตารางดังรูปที่ 4.21 status ของพัดลมตัวที่ 1 เท่ากับ 0 ตัวที่ 2 และ 3 เท่ากับ 1 เมื่อถึงเวลาในช่วง 19:05:00 น.

ถึง 19:10:00 น.พัดลมตัวที่ 1 กับ3 จะทำงาน ตัวที่ 2 หยุดทำงาน (พัดลมตัวที่ 1 และ 3 status = 1 ตัวที่ 2 status = 0) เมื่อถึงเวลาในช่วง 19:10:00 น. ถึง 19:15:00 น. พัดลมตัวที่หนึ่งจะทำงานเพียงตัวเดียว (พัดลมตัวที่ 1 status = 1 ตัวที่ 2 และ 3 status = 0) และเมื่อเวลาถึง 19:15:00 น. พัดลมทุกตัวหยุดทำงาน (พัดลมทุกตัว status = 0)



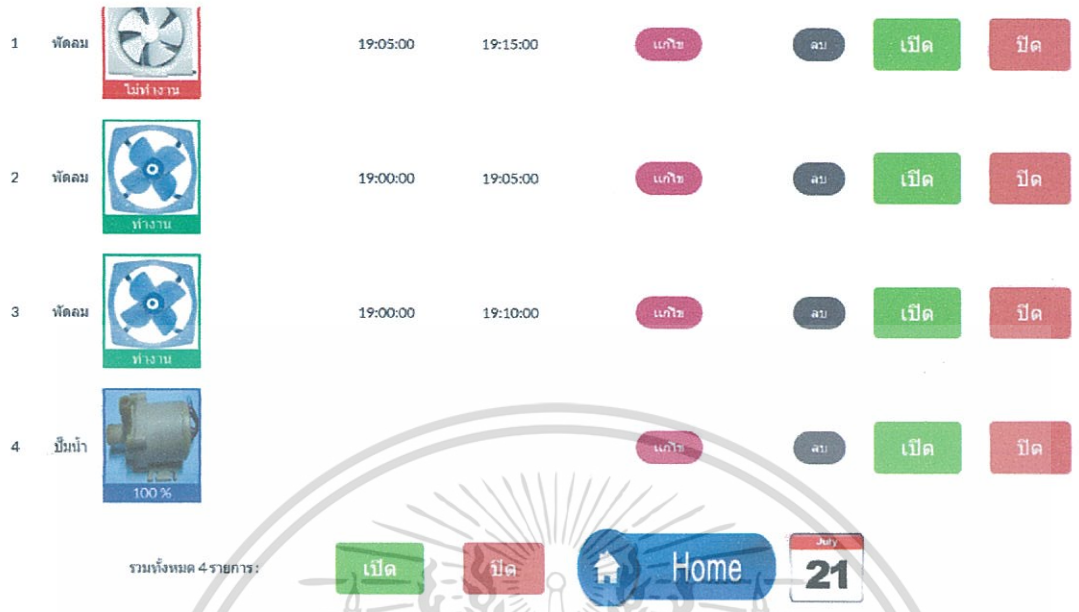
รูปที่ 4.17 หน้าเว็บควบคุมอุปกรณ์ไฟฟ้า (ณ ก่อนเวลา 19:05:00 น.)



รูปที่ 4.18 หน้าควบคุมอุปกรณ์ไฟฟ้าที่เขียนด้วยภาษา C# (ณ เวลา 18:58:15 น.)

	stationno	name	status	timestart	timestop	timestart2	timestop2	volumn
	1	พัดลม	0			19:05:00	19:15:00	0
	2	พัดลม	0			19:00:00	19:05:00	0
	3	พัดลม	0			19:00:00	19:10:00	0
	4	ปั๊มน้ำ	0					100

รูปที่ 4.19 ข้อมูลในตาราง tbstation (ณ ก่อนเวลา 19:05:00 น.)



รูปที่ 4.20 หน้าเว็บควบคุมอุปกรณ์ไฟฟ้า (ช่วงเวลา 19:00:00 น. ถึง 19:05:00 น.)

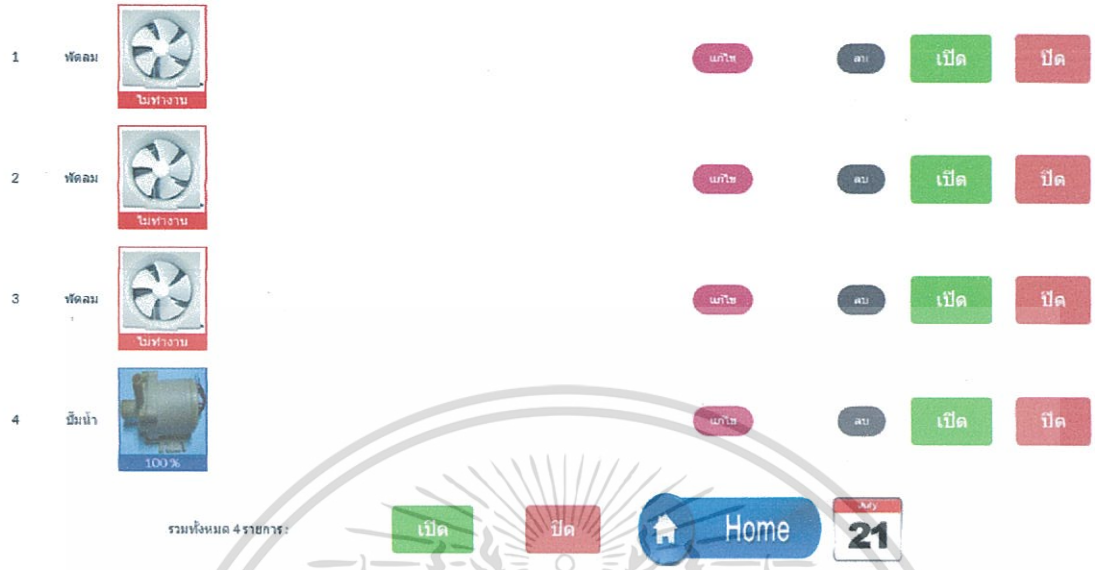
	stationno	name	status	timestart	timestop	timestart2	timestop2	volumn
	1	พัดลม	0			19:05:00	19:15:00	0
	2	พัดลม	1			19:00:00	19:05:00	0
	3	พัดลม	1			19:00:00	19:10:00	0
	4	ปั๊มน้ำ	0					100

รูปที่ 4.21 ข้อมูลในตาราง tbstation (ช่วงเวลา 19:00:00 น. ถึง 19:05:00 น.)

4.1.2.2 ทดสอบเปิดปิดพัดลม

1) ทดสอบเปิดปิดพัดลม 1 ตัว

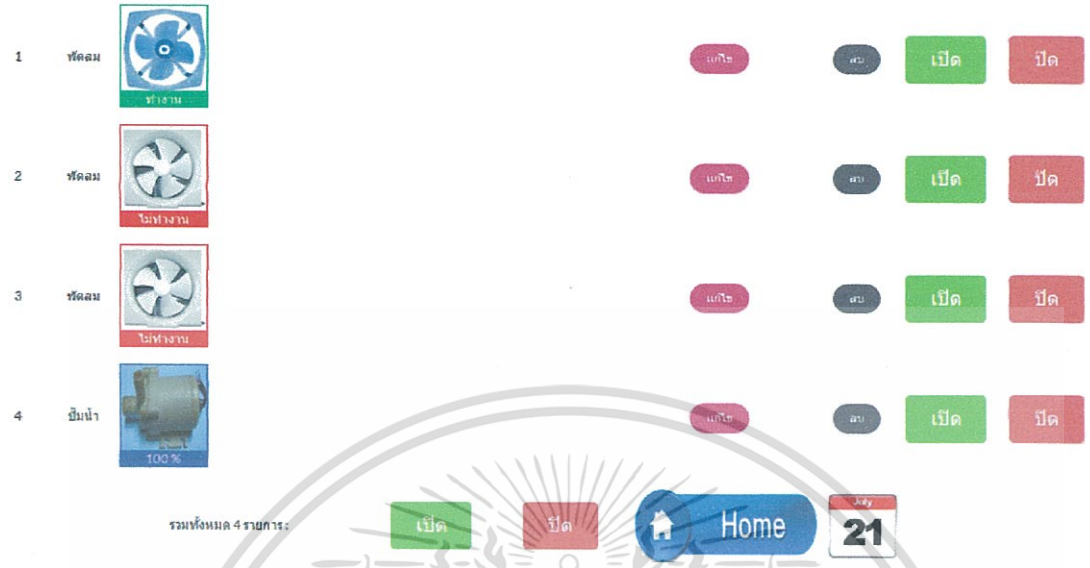
ทำการทดสอบเปิด ปิดพัดลมตัวที่ 1 โดยทำการคลิกที่ ปุ่มเปิด ในรูปที่ 4.22 ในแถวที่ 1 แล้วจะบันทึกข้อมูลในตาราง tbstation ดังรูปที่ 4.23 จากค่า status เท่ากับ 0 ค่าจะบันทึกเปลี่ยนเป็น status เท่ากับ 1 ดังรูปที่ 4.23 รูปพัดลมตัวที่ 1 จะเปลี่ยนไปดังรูปที่ 4.24 จากนั้น โปรแกรม c# จะทำการประมวลผลอ่านข้อมูลในตารางนี้ แล้วทำการส่งคำสั่งผ่าน Zigbee ส่งไปให้ไมโครคอนโทรลเลอร์ควบคุมพัดลมให้ทำงาน ซึ่งการควบคุมผ่านหน้าต่างควบคุมอุปกรณ์ไฟฟ้า c# ก็มีลักษณะคล้ายกันโดยจะบันทึกข้อมูลลงในตารางก่อน



รูปที่ 4.22 หน้าเว็บควบคุมอุปกรณ์ไฟฟ้า

	stationno	name	status	timestart	timestop	timestart2	timestop2	volumn
	1	พัดลม	0	NULL	NULL	NULL	NULL	0
	2	พัดลม	0	NULL	NULL	NULL	NULL	0
	3	พัดลม	0	NULL	NULL	NULL	NULL	0
	4	ปั๊มน้ำ	0	NULL	NULL	NULL	NULL	100

รูปที่ 4.23 ข้อมูลในตาราง tbstation



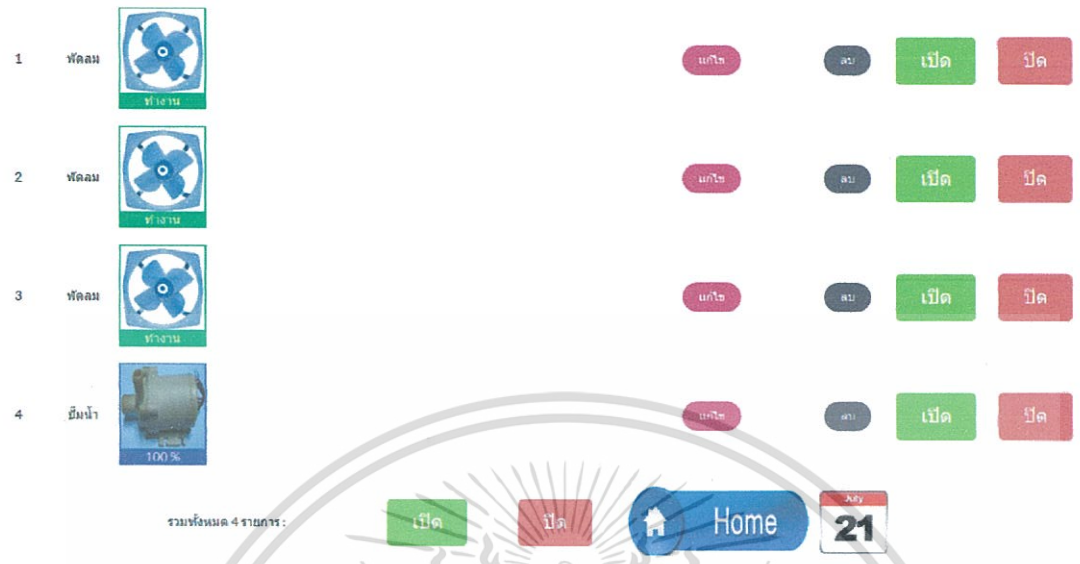
รูปที่ 4.24 หน้าเว็บควบคุมอุปกรณ์ไฟฟ้า

	stationno	name	status	timestart	timestop	timestart2	timestop2	volumn
	1	พัดลม	1	NULL	NULL	NULL	NULL	0
	2	พัดลม	0	NULL	NULL	NULL	NULL	0
	3	พัดลม	0	NULL	NULL	NULL	NULL	0
	4	ปั๊มน้ำ	0	NULL	NULL	NULL	NULL	100

รูปที่ 4.25 ข้อมูลในตาราง tbstation

2) ทดสอบเปิดปิดพัดลมทุกตัว

ทำการทดสอบเปิดพัดลมทุกตัว เมื่อทำการคลิก ปุ่มเปิดที่อยู่ด้านล่างสุดในหน้าเว็บควบคุมอุปกรณ์ไฟฟ้าดังรูปที่ 4.26 จะบันทึกข้อมูลลงในตาราง tbstation ดังรูปที่ 4.27 โดยค่า status ทุกอุปกรณ์มีค่าเท่ากับ 1 จากนั้น โปรแกรม c# จะอ่านข้อมูลในตารางนี้และส่งคำสั่งผ่าน Zigbee ให้ไมโครคอนโทรลเลอร์ ควบคุมพัดลมทุกตัวให้ทำงาน ส่วนหน้าต่างควบคุมอุปกรณ์ไฟฟ้า c# ก็มีการทำงานเหมือนกันโดยต้องทำการคลิกปุ่ม open all



รูปที่ 4.26 หน้าเว็บควบคุมอุปกรณ์ไฟฟ้า

	stationno	name	status	timestart	timestop	timestart2	timestop2	volumn
	1	พัดลม	1	NULL	NULL	NULL	NULL	0
	2	พัดลม	1	NULL	NULL	NULL	NULL	0
	3	พัดลม	1	NULL	NULL	NULL	NULL	0
	4	ปั๊มน้ำ	1	NULL	NULL	NULL	NULL	100

รูปที่ 4.27 ข้อมูลในตาราง tbstation

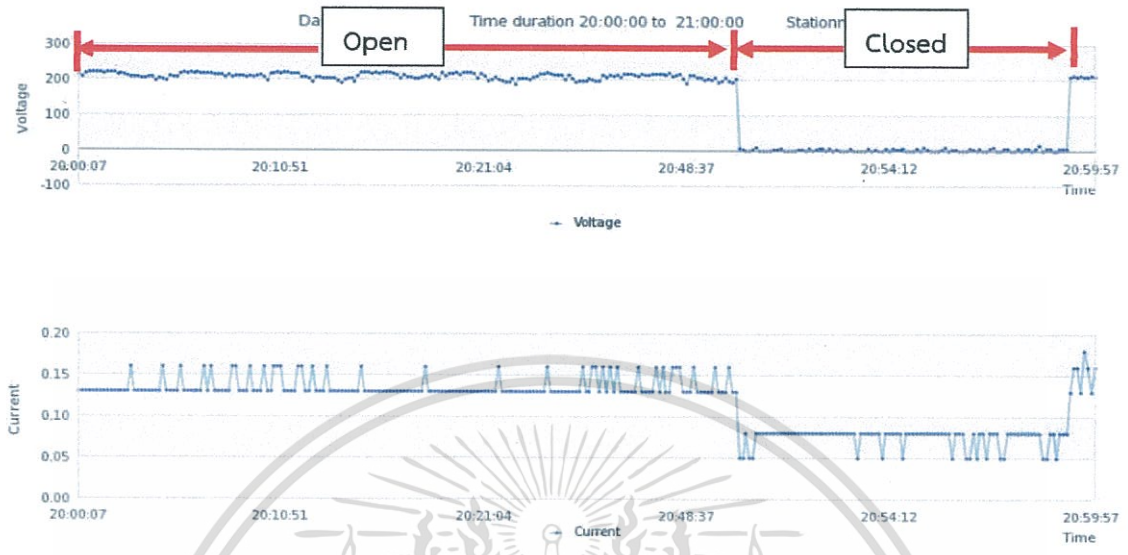
4.9 ทดสอบพล็อตกราฟ

4.9.1 ทดสอบพล็อตกราฟหน้าเว็บแสดงข้อมูลกราฟ

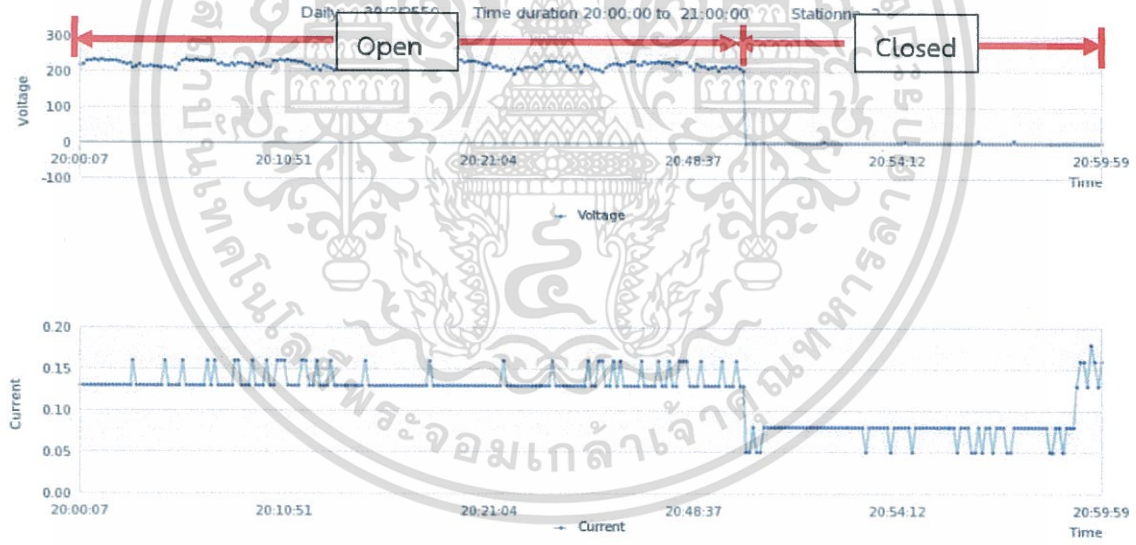
ทำการทดสอบพล็อตกราฟในหน้าเว็บแสดงข้อมูลกราฟ โดยสั่งพล็อตกราฟอุปกรณ์ปั๊มน้ำ กำหนดวันที่ 6/4/2559 เวลา 17:30:00 น. ถึง เวลา 17:50:00 น. เลือกอุปกรณ์หมายเลข 3 แล้วทำการสั่งพล็อต ได้ผลออกมาแสดงเป็นกราฟแรงดันและกระแสเทียบกับเวลาดังรูปที่ 4.28 เมื่อสังเกตกราฟ กราฟแบ่งออกได้เป็น 6 ช่วงตามเปอร์เซ็นต์การทำงานของปั๊มน้ำ ช่วง 1 72 – 100 % ช่วง 2 64% ช่วง 3 56% ช่วง 4 48% ช่วงที่ 5 40% ช่วงที่ 6 32% จะพบอีกว่ากราฟจะเกิดการลดอันเนื่องได้ทำการลดค่าเปอร์เซ็นต์การทำงานของปั๊มน้ำจาก 100 % จนถึง 32% เมื่อดูที่กราฟในแต่ละช่วงจะพบว่าทั้งในกราฟกระแสและแรงดันเทียบกับเวลาค่ากระแสและแรงดันในแต่ละช่วงไม่คงที่ อันเนื่องมาจากความผิดพลาดการส่งค่าผ่าน Zigbee หรือการอ่านค่าที่ผิดพลาดของเซ็นเซอร์อันเนื่องมาจากไฟเลี้ยงไม่เพียงพอ

รูปที่ 4.28 กราฟแรงดันและกระแสของปั๊มน้ำเทียบกับเวลา

ทำการทดสอบพล็อตกราฟของพัดลม 1 และ 2 ตามลำดับ โดยกำหนดค่าต่างๆ เหมือนกันดังนี้ วันที่ 30/3/59 และช่วงเวลา 20:00:00 น. ถึง ช่วงเวลา 21:00:00 น. แล้วตัวที่ 1 เลือกหมายเลขอุปกรณ์ 1 แล้วตัวที่ 2 เลือกหมายเลขอุปกรณ์ 2 ทำการพล็อตทีละกราฟ ได้ผลกราฟแสดงออกมาดังรูปที่ 4.29 และ 4.30 ตามลำดับ จากทั้งสองรูปจะสังเกตเห็นว่ากราฟแบ่งได้เป็น 2 ช่วง ตามลักษณะการทำงานของพัดลม คือ ทำงาน (ช่วงopen) และ ไม่ทำงาน (ช่วงclosed) และพบอีกว่ากราฟของทั้งสองอุปกรณ์เปลี่ยนจาก ทำงาน เป็นไม่ทำงาน เพราะว่าได้ทำการสั่งให้พัดลมทำงาน และไม่ทำงาน ตามลำดับ เมื่อสังเกตดูค่าในแต่ละช่วงทั้งช่วง Open และ Closed จะมีค่าไม่คงที่ อันเนื่องมาจากความผิดพลาดส่งข้อมูลผ่าน Zigbee หรือ การอ่านค่าจากเซ็นเซอร์ที่ผิดพลาด อันเนื่องมาจากไฟเลี้ยงไม่เพียงพอ



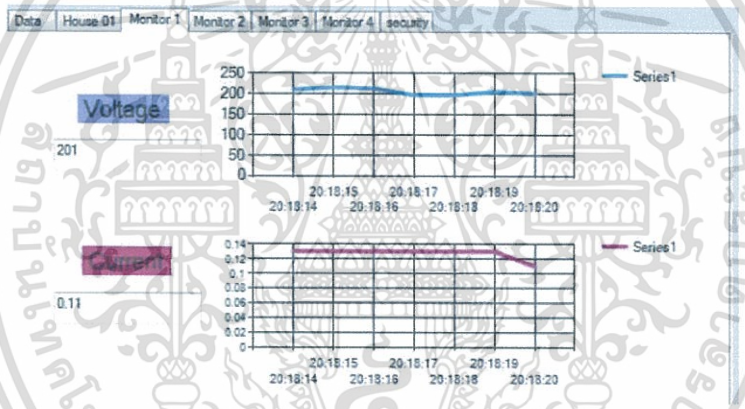
รูปที่ 4.29 กราฟแรงดันและกระแสเทียบกับเวลาของพัดลมตัวที่ 1



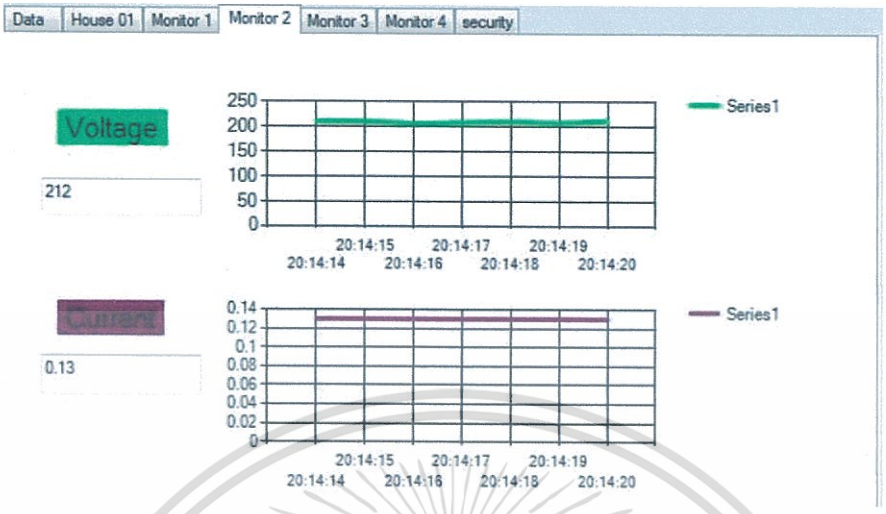
รูปที่ 4.30 กราฟแรงดันและกระแสเทียบกับเวลาของพัดลมตัวที่ 2

4.9.2 ทดสอบพล็อตกราฟหน้าแสดงกราฟ c#

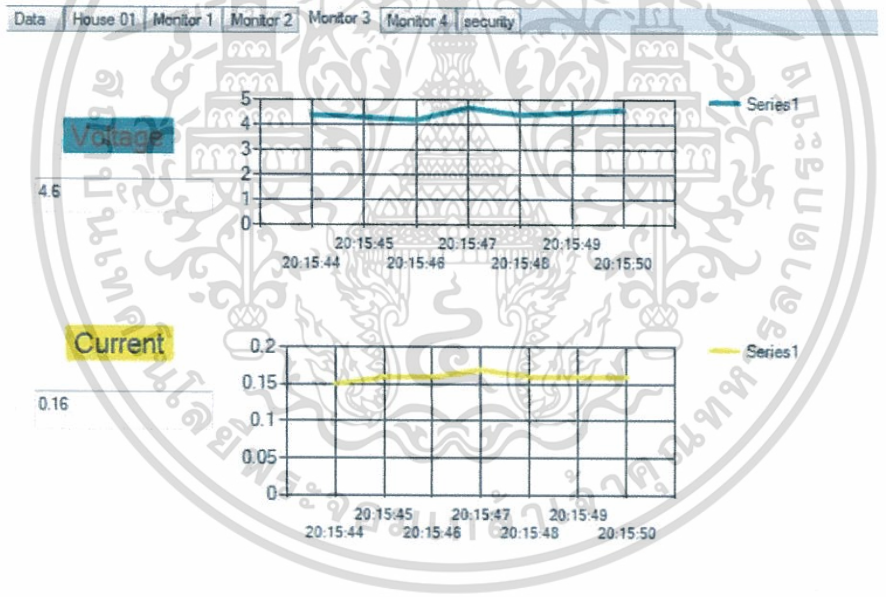
ทำการทดสอบพล็อตกราฟในหน้าแสดงกราฟ c# กราฟในหน้าเหล่านี้จะแสดงค่าอัตโนมัติ โดยกราฟจะแสดงเป็นกราฟแรงดันเทียบกับเวลา และกราฟกระแสเทียบกับเวลา ค่าเวลานั้นจะแสดงย้อนหลังไป 7 วินาที ตามแสดงในรูป กราฟในรูปที่ 4.31 เป็นกราฟของพัดลมตัวที่ 1 แรงดันจะแสดงค่าอยู่ที่ประมาณ 200 โวลต์กระแสอยู่ที่ประมาณ 0.11 แอมป์ จากค่าที่สังเกตเห็นแสดงว่าพัดลมตัวที่ 1 ทำงาน จากรูปที่ 4.32 เป็นกราฟของพัดลมตัวที่ 2 ลักษณะกราฟและค่าใกล้เคียงกับคล้ายกับรูปที่ 4.29 แสดงว่าพัดลมตัวที่ 2 ทำงาน ส่วนกราฟในรูปที่ 4.33 เป็นกราฟของปั๊มน้ำ แรงดันแสดงค่าอยู่ที่ประมาณ 4.5 โวลต์ กระแสแสดงค่าอยู่ที่ประมาณ 0.16 แอมป์ จากค่าที่สังเกตเห็นแสดงว่าปั๊มน้ำทำงานอยู่ที่ 40% เมื่อสังเกตค่ากระแสและแรงดันของกราฟของทั้งสามอุปกรณ์ ค่าที่ได้จะไม่คงที่ อันเนื่องมาจากการส่งค่าที่ผิดพลาดอันเนื่องมาจากการส่งผ่าน Zigbee หรือการอ่านค่ากระแสและแรงดันที่ผิดพลาดอันเนื่องมาจากไฟเลี้ยงไม่เพียงพอ



รูปที่ 4.31 หน้าแสดงกราฟพัดลมตัวที่ 1



รูปที่ 4.32 หน้าแสดงกราฟพัดลมตัวที่ 2



รูปที่ 4.33 หน้าแสดงกราฟปั๊มน้ำ

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

ระบบควบคุมอุปกรณ์ไฟฟ้าแบบไร้สายในโรงเรือนนี้ สามารถที่จะสั่งงานควบคุมอุปกรณ์ต่างๆในระบบนี้ได้ ด้วยโปรแกรมที่เขียนด้วยภาษา C# โดยเขียนบน Microsoft Visual Studio และสามารถเฝ้าสังเกตการณ์ทำงานของอุปกรณ์ได้จากการตรวจสอบค่ากระแสและแรงดันที่วัดได้ของอุปกรณ์แต่ละตัวแสดงในรูปของกราฟและตัวเลข ซึ่งจะวัดค่าต่างๆของอุปกรณ์ด้วยเซ็นเซอร์ และทำการส่งผ่านโมดูลไร้สายซิกบี เช่นเดียวกันกับการสั่งงาน ซึ่งต้องส่งค่าผ่านซิกบีออกไปหาไมโครคอนโทรลเลอร์เช่นกัน นอกจากนั้นผู้ใช้สามารถตั้งเวลาเพื่อให้อุปกรณ์ทำงานเองอัตโนมัติเมื่อถึงเวลาที่ได้ทำการตั้งค่าเอาไว้ นอกจากการสั่งงานผ่านโปรแกรมแล้วยังทำทั้งหมดหมดนี้ได้แบบเดียวกันกับการสั่งผ่านหน้าเว็บ โดยที่ตัวโปรแกรมและเว็บไซด์จะเชื่อมข้อมูลกับระบบเซิร์ฟเวอร์และฐานข้อมูลร่วมกัน ดังนั้นแม้ผู้ใช้จะอยู่จากที่ไกลๆก็สามารถที่จะตรวจสอบการทำงานและสั่งการระบบได้

5.2 ปัญหาที่พบในการทดลอง

1. โมดูลไร้สายซิกบีมีข้อจำกัดในเรื่องระยะทาง ทำให้การรับส่งข้อมูลควรมีระยะไม่ไกลกันจนเกินไป และไม่ควรมีสิ่งบดบังระหว่างทาง

2. การรวมโค้ดโปรแกรมทั้งการส่งค่ากลับและรอค่ารับเพื่อควบคุมอุปกรณ์เอาไว้ในไมโครคอนโทรลเลอร์ตัวเดียวกัน สามารถทำได้ยาก

5.3 การแก้ปัญหา

1. จัดวางตำแหน่งของอุปกรณ์ทั้งภาคส่งและรับให้อยู่ในตำแหน่งที่ดีที่สุด

2. แบ่งการทำงานของตัวไมโครคอนโทรลเลอร์โดยให้ตัวหนึ่งทำหน้าที่รอรับคำสั่งควบคุมและอีกตัวทำหน้าที่คอยส่งค่าที่วัดได้จากเซ็นเซอร์ โดยทั้งสองรับส่งข้อมูลผ่านซิกบีร่วมกัน

5.4 ข้อเสนอแนะ

ในระบบนี้ยังมีความช้าในเรื่องการรับส่งข้อมูลพอสมควร โดยอาจจะนำระบบที่มีความเร็วหรือประสิทธิภาพมากกว่านี้เข้ามาใช้งานแทนก็จะสามารถทำให้ระบบนี้เป็นระบบที่น่าใช้งานมากกว่าเดิม

ภาคผนวก ก

ASCII Table and Description

ASCII Table and Description

เนื่องจากคอมพิวเตอร์เข้าใจเพียงตัวเลข การใช้ ASCII Table เป็นวิธีการเชิงตัวเลขที่จะทำให้คอมพิวเตอร์เข้าใจคำสั่ง แอสเป็นรหัสอักขระที่ประกอบด้วยอักษรละติน เลขอารบิก เครื่องหมายวรรคตอน และสัญลักษณ์ต่างๆ โดยแต่ละรหัสจะแทนด้วยตัวอักขระหนึ่งตัว เช่น รหัส 65 (เลขฐานสิบ) ใช้แทนอักษรเอ (A) พิมพ์ใหญ่ เป็นต้น

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

ภาคผนวก ข

Arduino Code

1. Code Arduino สำหรับควบคุมการเปิดปิดพัดลมระบายอากาศและความแรงของมอเตอร์ปั๊มน้ำ

```
#include <SoftwareSerial.h>
#define RxD 0 //
#define TxD 1 //
SoftwareSerial mySerial(RxD,TxD); // RX, TX

int Relay01=2;
int Relay02=3;
int Relay03=4;
//int Relay04=7;
int pin[8]={5,6,7,8,9,10,11,12};

int test01 ;
int test02 ;
int test;

void setup(){
Serial.begin(9600);
pinMode(RxD, INPUT);
pinMode(TxD, OUTPUT);
mySerial.begin(9600);
pinMode(Relay01,OUTPUT);
pinMode(Relay02,OUTPUT);
pinMode(Relay03,OUTPUT);
//pinMode(Relay04,OUTPUT);

int pin[8]={3,4,5,6,7,8,9,10};
    for(int i=0; i <8 ; i++){
  pinMode(pin[i],OUTPUT);
  }
}
```

```
void loop() {

  if (mySerial.available()>0){
  test01=mySerial.read();
  if(test01==49){  digitalWrite(Relay01,HIGH);      }
  if(test01==50){  digitalWrite(Relay02,HIGH);      }
  if(test01==51){  digitalWrite(Relay03,HIGH);      }
  if(test01==52){  digitalWrite(Relay04,HIGH);      }

  if(test01==65){  digitalWrite(Relay01,LOW);       }
  if(test01==66){  digitalWrite(Relay02,LOW);       }
  if(test01==67){  digitalWrite(Relay03,LOW);       }
  if(test01==68){  digitalWrite(Relay04,LOW);       }

if(test01 ==97)  { for(int n=0;n<7;n++){  digitalWrite(pin[n],LOW);   }  }
if(test01 ==98)
  {
    for(int n=0;n<3;n++){  digitalWrite(pin[n],LOW);    }
    digitalWrite(pin[3],HIGH);    digitalWrite(pin[4],LOW);
    digitalWrite(pin[5],HIGH);    digitalWrite(pin[6],LOW);
    digitalWrite(pin[7],HIGH);
  }
   if(test01 ==99)
  {
    for(int n=0;n<2;n++){    digitalWrite(pin[n],LOW);    }
    digitalWrite(pin[2],HIGH);    digitalWrite(pin[3],LOW);
    digitalWrite(pin[4],HIGH);    digitalWrite(pin[5],LOW);
    digitalWrite(pin[6],HIGH);    digitalWrite(pin[7],HIGH);
  }
   if (test01 ==100)
  {
    digitalWrite(pin[0],LOW);    digitalWrite(pin[1],HIGH);
    for(int n=2;n<7;n++){    digitalWrite(pin[n],LOW);    }
  }
```

```
if (test01 ==101)
  {
 digitalWrite(pin[0],LOW);   digitalWrite(pin[1],HIGH);
 digitalWrite(pin[2],LOW);    digitalWrite(pin[3],HIGH);
 digitalWrite(pin[4],LOW);   digitalWrite(pin[5],HIGH);
 digitalWrite(pin[6],LOW);   digitalWrite(pin[7],HIGH);
}
  if (test01 ==102)
{
  digitalWrite(pin[0],LOW);    digitalWrite(pin[1],HIGH);
  digitalWrite(pin[2],HIGH);    digitalWrite(pin[3],LOW);
  digitalWrite(pin[4],HIGH);    digitalWrite(pin[5],LOW);
  digitalWrite(pin[6],HIGH);    digitalWrite(pin[7],HIGH);
}
  if (test01 ==103)
{   for(int m=0;m<1;m++){   digitalWrite(pin[m],HIGH);    }
  for(int n=1;n<7;n++){   digitalWrite(pin[n],LOW);    }
}
 if (test01 ==104)
  {
  digitalWrite(pin[0],HIGH);    digitalWrite(pin[1],LOW);
  digitalWrite(pin[2],LOW);    digitalWrite(pin[3],HIGH);
  digitalWrite(pin[4],LOW);    digitalWrite(pin[5],HIGH);
  digitalWrite(pin[6],LOW);    digitalWrite(pin[7],HIGH);
}
   if (test01 ==105)
{
  digitalWrite(pin[0],HIGH);      digitalWrite(pin[1],LOW);
  digitalWrite(pin[2],HIGH);      digitalWrite(pin[3],LOW);
  digitalWrite(pin[4],HIGH);      digitalWrite(pin[5],LOW);
  digitalWrite(pin[6],HIGH);      digitalWrite(pin[7],HIGH);
}

if (test01 ==106)
  {   for(int m=0;m<2;m++){   digitalWrite(pin[m],HIGH);    }
  for(int n=2;n<7;n++){   digitalWrite(pin[n],LOW);   }   }
```

```
if (test01 ==107)
 {   for(int m=0;m<2;m++){   digitalWrite(pin[m],HIGH);  }
  digitalWrite(pin[2],LOW);    digitalWrite(pin[3],HIGH);
  digitalWrite(pin[4],LOW);    digitalWrite(pin[5],HIGH);
  digitalWrite(pin[6],LOW);    digitalWrite(pin[7],HIGH);
}

 if (test01 ==108)
  {   for(int m=0;m<3;m++){   digitalWrite(pin[m],HIGH);   }
  digitalWrite(pin[3],LOW);    digitalWrite(pin[4],HIGH);
  digitalWrite(pin[5],LOW);    digitalWrite(pin[6],HIGH);
  digitalWrite(pin[7],LOW);
}
if (test01 ==109)
  {   for(int m=0;m<8;m++){   digitalWrite(pin[m],HIGH);   }      }



}
}
```

จากโค้ดทำการรับค่าตัวอักษรที่รับมาทาง Xbee แล้วเข้าสู่เงื่อนไข if(serial.available>0) จากนั้นใช้ตัวแปรทำการรับค่าก็คือ test01 โดยจะนำมาเช็คกับเงื่อนไขต่างๆถ้าตรงกับเงื่อนไขอันไหนก็จะทำตามเงื่อนไขนั้น โดยที่ test01 กำหนดให้เป็น interger จะต้องไปทำการเทียบค่าในตาราง ASCII Table (สามารถดูได้ในภาคผนวก)

2. Code โปรแกรมรวมของ Arduino ตัวที่ 2 ซึ่งมีหน้าที่การส่งค่าเซ็นเซอร์กระแสและแรงดัน

```
#include <SoftwareSerial.h> //
#define RxD 0
#define TxD 1
SoftwareSerial mySerial(RxD,TxD); // RX, TX
////////////////////////////////////////////////////////////////////////////
#define CURRENT_SENSOR1 A0  // sensor pin1 A0
#define CURRENT_SENSOR2 A1  // sensor pin1 A0
//#define CURRENT_SENSOR3 A2  // sensor pin1 A0

//float amplitude_current1;float amplitude_current2;float amplitude_current3;
//float effective_value1;float effective_value2;float effective_value3;
////////////////////////////////////////////////////////////////////////////
//this part is AC Voltage
const int VOLTAGE_SENSOR001 = A3;
const int VOLTAGE_SENSOR002 = A4;
//const int VOLTAGE_SENSOR003 = A5;
int sensorValue001 = 0,sensorValue002 = 0,sensorValue003 = 0;        // value read
from the pot
int outputValue001 = 0,outputValue002 = 0,outputValue003 = 0;
////////////////////////////////////////////////////////////////////////////
//DC Part
//Define Voltage and Current sensor pins
#define DC_CURRENT01 A2
#define DC_Voltage01 A5
int VSS;
float VDC_value;
```

```
//AC Current and Voltage Sensor Declaration
void pins_init1()
{  pinMode(CURRENT_SENSOR1, INPUT);  }
void pins_init2()
{  pinMode(CURRENT_SENSOR2, INPUT);  }
void pins_init4()
{  pinMode(VOLTAGE_SENSOR001,INPUT);  }
void pins_init5()
{  pinMode(VOLTAGE_SENSOR002,INPUT);  }
//////////////////////////////////////////////////////
// AC current Sensor Function
float getAC_Current001()
{
  int sensorValue1;     float sensorMax1 = 0
  uint32_t start_time = millis();
  while((millis()-start_time) < 250)
  {
    sensorValue1 = analogRead(CURRENT_SENSOR1);
    if (sensorValue1 > sensorMax1)
      {       sensorMax1 = sensorValue1;     }
  }
     return sensorMax1;
}

float getAC_Current002()
{
  int sensorValue2;   float sensorMax2=0;
  uint32_t start_time = millis();
  while((millis()-start_time) < 250)
  {
     sensorValue2 = analogRead(CURRENT_SENSOR2);
    if (sensorValue2 > sensorMax2)
      {       sensorMax2 = sensorValue2;    }
  }
        return sensorMax2;
}
```

```
//AC voltage 01//
int getAC_voltage01()
{
    int sensorACV01_Value;        int ACV01 = 0;
    uint32_t start_time = millis();
    while((millis()-start_time) < 125) //sample for 1000ms
    {
        sensorACV01_Value = analogRead(VOLTAGE_SENSOR001);
        if (sensorACV01_Value > ACV01)
        {          ACV01 = sensorACV01_Value;        }
    }
  return ACV01; //
}

////AC votage 2//
int getAC_voltage02()
{
    int sensorACV02_Value;        int ACV02 = 0;
      uint32_t start_time = millis();
    while((millis()-start_time) < 125) //sample for 1000ms
    {
       sensorACV02_Value = analogRead(VOLTAGE_SENSOR002);
       if (sensorACV02_Value > ACV02)
       {          ACV02 = sensorACV02_Value;        }
     }
     return ACV02;
}
```

```
//DC Function
void pins_DC1() {  pinMode(DC_CURRENT01,INPUT);  }
void pins_DC2() {  pinMode(DC_Voltage01,INPUT);  }

//DC current
float GetDC_Current()
{
  float SensorDC_I,average = 0;
  for(int i = 0; i < 250; i++)
  {
      average = average + (.0264 * analogRead(DC_CURRENT01) -13.51) / 250;
      delay(1);
  }
    return average;
}

//DC Voltage
float GetDC_Voltage()
{
  float temp;
  VSS=analogRead(DC_Voltage01);
  temp=VSS/4.092;
  VSS=(int)temp;//
  VDC_value=((VSS%100)/10.0);

  return VDC_value;
}
```

```
void setup()
{
  Serial.begin(9600);
  pins_init1();
  pins_init2();
  pins_init4();
  pins_init5();
  pins_DC1();
  pins_DC2();
  pinMode(RxD, INPUT);
  pinMode(TxD, OUTPUT);
  mySerial.begin(9600);
 }

void loop()
{

  float AC_current001;float AC_current002;
  AC_current001 = getAC_Current001();
  AC_current002 = getAC_Current002();

  int VACx01,VACx02;
  VACx01=getAC_voltage01();
  VACx02=getAC_voltage02();

  float Val_DCI;float Val_DCV;
  Val_DCI = GetDC_Current();
  Val_DCV = GetDC_Voltage();
```

```
  //Calculation
  float amplitude_current1;       float effective_value1;
  float amplitude_current2;       float effective_value2;

  amplitude_current1=(float)(AC_current001-512)/1024*5/185*1000000/1000;
  amplitude_current2=(float)(AC_current001-512)/1024*5/185*1000000/1000;
  effective_value1=amplitude_current1/1.414;
  effective_value2=amplitude_current2/1.414;

  if(amplitude_current1<=0.04&&amplitude_current1>-5)
    {amplitude_current1=0;}
  if(amplitude_current2<=0.04&&amplitude_current2>-5)
    {amplitude_current2=0;}
      //
  int amplitude_voltage01=0; //int effective_voltage01=0;
  int amplitude_voltage02=0; //int effective_voltage02=0;
  amplitude_voltage01=(float)(VACx01-512)/1024*1000;
  amplitude_voltage02=(float)(VACx02-512)/1024*1000;

//Print Value of Current and Voltage
 Serial.print("/");

  Serial.print("1:");Serial.print(amplitude_voltage01);//voltage
  Serial.print(":"); Serial.print(amplitude_current1);//current
  Serial.print(",");

  Serial.print("2:");Serial.print(amplitude_voltage02);//voltage
   Serial.print(":"); Serial.print(amplitude_current2);
   Serial.print(",");

  Serial.print("3:");Serial.print(Val_DCV);//voltage
  Serial.print(":");Serial.println(Val_DCI);

}
```

จาก Code เหล่านี้จะเห็นว่าเริ่มแรกเราให้ sensor อ่านค่าออกมาจะได้เป็นค่า Sensor value จากนั้นเอาค่าที่ได้มาทำการคำนวณแล้วก็ปริ้นค่าออกมา เราต้องทำแบบนี้กับทั้ง 6 ค่าโดยใช้ fuction เป็นตัวดำเนินการ sensor แต่ละตัว โดยค่าจะอ่านทุกๆ 1 วินาที(1000 ms) คิดจากผลรวมของ delay ในลูปต่างๆ

3. code Arduino ตัวที่ 3 สำหรับการการอ่านค่าความถี่และป้อนความถี่สำหรับวงจรPLL

```
#include <FreqMeasure.h>
double frz = 0;
void setup() {
  Serial.begin(57600);
  FreqMeasure.begin();
  pinMode(13, OUTPUT);
}
 double sum=0; int count=0;
 void loop() {
  if (FreqMeasure.available()) {
    sum = sum + FreqMeasure.read();
    count = count + 1;
    if (count > 0) {
      double frequency = F_CPU / (sum / count);
      Serial.println(frequency);
      sum = 0;
      count = 0;
      frz = (500/frequency);
    }
    digitalWrite(13, HIGH);
    delay(frz);
    digitalWrite(13, LOW);
    delay(frz);
  }
}
```

Code นี้ ดัดแปลงมาจากตัวอย่างของ Code การวัดความถี่ด้วยไลบรารี่ที่ชื่อ FreqMeasure ซึ่งทำการวัดค่าความถี่ (input) ที่ขา8 (กำหนดมาให้ใช้ขานี้) และทำการสร้างสัญญาณใหม่ที่มีความถี่เท่ากันที่ขา 13 (output) โดย Code นี้จะนำไปใช้ร่วมกับวงจร PLL (Phase Lock Loop) อ่านค่าจาก Flowrate และป้อนเข้าไปใน PLL ทำให้ควบคุมมอเตอร์ปั๊มน้ำได้ด้วยอัตราที่คงที่แม้ระดับน้ำจะเปลี่ยนแปลงไป

บรรณานุกรม

[1] Arduino Board

http://www.makerzoo.co/wp-content/uploads/2014/12/11021-01a.jpg

(สืบค้นวันที่ 4 กันยายน 2558)

[2] ATmega168/328-Arduino Pin Mapping

http://www.arduino.cc/en/Hacking/PinMapping168 (สืบค้นวันที่ 6 กันยายน 2558)

[3] ลักษณะ connector แบบ DB9 และ DB25

http://www.machinetoolhelp.com/Applications/RS232Communications.html

(สืบค้นวันที่ 6 กันยายน 2558)

[4] รีเลย์

http://www.semi-shop.com/knowledge/knowledge_detail.php?sk_id=28

(สืบค้นวันที่ 7 กันยายน 2558)

[5] การทำงานของ Relay

http://www.semi-shop.com/knowledge/knowledge_detail.p (สืบค้นวันที่ 15 กันยายน 2558)

[7] Sensor ACS712

http://www.arduitronics.com/product/24/5a-current-sensor-module-2

(สืบค้นวันที่ 15 กันยายน 2558)

บรรณานุกรม(ต่อ)

[8] การต่อใช้งาน ACS712

http://www.arduitronics.com/product/24/5a-current-sensor-module-2

(สืบค้นวันที่ 15 กันยายน 2558)

[9] Sensors : Single Phase Voltage Sensor

http://www.thaieasyelec.com/en/home/single-phase-voltage-sensor-detail.html?tmpl=component&flexiblelayout=print

(สืบค้นวันที่ 11 กันยายน 2558)

[10] การต่อใช้งาน Single Phase Voltage Sensor

http://www.emartee.com/product/42082/

(สืบค้นวันที่ 11 กันยายน 2558)

[11] เซ็นเซอร์วัดแรงดัน DC

http://www.arduinoall.com/p/598

(สืบค้นวันที่ 11 กันยายน 2558)

[12] DAC 0808 การใช้งาน

http://www.alldatasheet.com/view.jsp?Searchword=Dac0808

(สืบค้นวันที่ 2 กุมภาพันธ์ 2558)

[13] รูปแสดงย่านความถี่ของการใช้งานซิกบี

http://fosiao.com/system/files/2013/zigbee.jpg (สืบค้นวันที่ 2 กันยายน 2558)

[14] การเชื่อมต่อเครือข่ายแบบ Physical Device ของ Zigbee

http://www.thaieasyelec.com/article-wiki/basic-electronics/what-is-zigbee.html

(สืบค้นวันที่ 2 กันยายน 2558)

บรรณานุกรม(ต่อ)

[15] การเชื่อมต่อกันเป็นเครือข่ายแบบ Logical Device

http://www.thaieasyelec.com/article-wiki/basic-electronics/what-is-zigbee.html

(สืบค้นวันที่ 2 กันยายน 2558)

[16] XBee 2mW Wire Antenna - Series 2

https://www.sparkfun.com/products/10414

(สืบค้นวันที่ 15 กันยายน 2558)

[17] ภาษา c#

https://sites.google.com/site/programmingm42/phasa-c

(สืบค้นวันที่ 12 กันยายน 2558)

[18] TCP/IP

http://cptd.chandra.ac.th/selfstud/datacom/CAI/part3-4.htm

(สืบค้นวันที่ 20 กันยายน 2558)

[19] PHP

https://www.gotoknow.org/posts/428663

(สืบค้นวันที่ 25 กันยายน 2558)

[20] รูปแสดงการเชื่อมต่อระหว่าง Clients และ Server

https://upload.wikimedia.org/wikipedia/commons/thumb/c/c9/Client-server-model.svg/2000px-Client-server-model.svg.png

(สืบค้นวันที่ 25 กันยายน 2558)