ระบบการให้น้ำผ่านโครงข่ายการสื่อสาร NB-IOT

IRRIGATION SYSTEM THROUGH NB-IOT NETWORK

โดย

นายธนพล เขียวหวาน

นายธนศาสตร์ อารีรอบ

นายพงศ์พสธร สกุลพระรักษา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2561

ระบบการให้น้ำผ่านโครงข่ายการสื่อสาร NB-IOT

IRRIGATION SYSTEM THROUGH NB-IOT NETWORK

โดย

นายธนพล	เขียวหวาน	58010501
นายธนศาสตร์	อารีรอบ	58010524
นายพงศ์พสธร	สกุลพระรักษา	58010818

อาจารย์ที่ปรึกษา

รศ.ดร.พิสิฐ บุญศรีเมือง

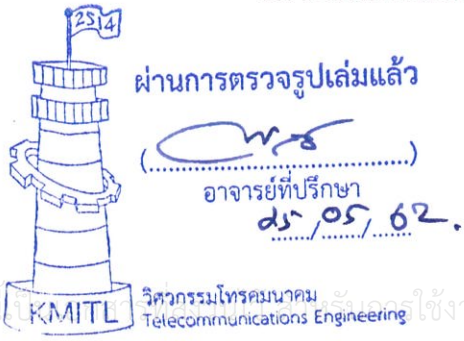
ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2561



ปริญญานิพนธ์ปีการศึกษา 2561

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบการให้น้ำผ่านโครงข่ายการสื่อสาร NB-IOT

IRRIGATION SYSTEM THROUGH NB-IOT NETWORK

ผู้จัดทำ

1. นายธนพล เขียวหวาน 58010501
2. นายธนศาสตร์ อารีรอบ 58010524
3. นายพงศ์พสธร สกุลพระรักษา 58010818

..... อาจารย์ที่ปรึกษา

(รศ.ดร.พิสิฐ บุญศรีเมือง)

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี เนื่องจากได้รับความกรุณาและช่วยเหลือในด้านต่างของ รศ.ดร.พิสิฐ บุญศรีเมือง อาจารย์ที่ปรึกษาปริญญานิพนธ์ ซึ่งท่านได้ให้คำแนะนำและข้อคิดเห็นต่างๆ อันเป็นประโยชน์อย่างยิ่งในการทำปริญญานิพนธ์ในครั้งนี้ อีกทั้งยังช่วยแก้ปัญหาต่างๆ ที่เกิดขึ้นระหว่างการดำเนินงานอีกด้วย ขอขอบคุณนักศึกษาปริญญาโทคุณอารีฟ ดาแตสาตู สำหรับข้อแนะนำและความช่วยเหลือในทุกๆด้านในการทำปริญญานิพนธ์

ขอขอบพระคุณคณาจารย์ภาควิชาวิศวกรรมโทรคมนาคม คณะวิศวกรรมสาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกๆท่านที่ได้ประสิทธิ์ประสาทวิชาความรู้ คำแนะนำต่างๆให้ข้าพเจ้า

สุดท้ายนี้ ผู้วิจัยขอขอบพระคุณบิดามารดา และครอบครัว ซึ่งเปิดโอกาสให้ได้รับการศึกษาเล่าเรียน ตลอดจนคอยช่วยเหลือและให้กำลังใจเสมอมา

นายธนพล เขียวหวาน
นายธนศาสตร์ อารีรอบ
นายพงศ์พสธร สกุลพระรักษา
ผู้จัดทำ

ระบบการให้น้ำผ่านโครงข่ายการสื่อสาร NB-IOT

IRRIGATION SYSTEM THROUGH NB-IOT NETWORK

โดย นายธนพล เขียวหวาน 58010501

นายธนศาสตร์ อารีรอบ 58010524

นายพงศ์พสธร สกุลพระรักษา 58010818

อาจารย์ที่ปรึกษา รศ.ดร.พิสิฐ บุญศรีเมือง

บทคัดย่อ

ปริญญานิพนธ์นี้ได้พัฒนาระบบการให้น้ำสำหรับพื้นที่การเกษตรซึ่งสามารถควบคุมการเปิด/ปิด วาล์วน้ำด้วยโปรแกรมที่ทันสมัย โดยโปรแกรมที่ทันสมัยสำหรับวาล์วน้ำจะถูกควบคุมโดยเว็บแอปพลิเคชันที่สามารถควบคุมการให้น้ำจากส่วนไหนก็ได้ของโลกโดยผ่านเครือข่ายสื่อสาร NB-IOT ซึ่งเป็นเทคโนโลยีการสื่อสารแบบใหม่ โดยผู้ใช้จะสามารถเชื่อมกับเว็บแอปพลิเคชันได้โดยใช้อุปกรณ์ใดๆก็ได้ที่สามารถเชื่อมต่อกับเวปเซิฟเวอร์โดยผ่านอินเทอร์เน็ต การเชื่อมต่อระหว่างวาล์วน้ำกับตัวควบคุมใช้การส่งข้อมูลแบบ POWER LINE COMMUNICATION เพื่อส่งข้อมูลได้จากสายไฟบ้านปกติเพื่อประหยัดค่าใช้จ่ายและแรงงาน ตามลำดับ

ABSTRACT

Recently, the irrigation system for large-scale agricultural areas requires a large number of water pipes, water pumps, water valves, and the labor costs. So, this thesis would like to solve these problems.

This project develops the irrigation system for large-scale agricultural areas, which can separately control the on/off water valves following the advanced

programs. The advanced programs for water valves are controlled by using Web application to control the flow of water from any part of the world via the NB-IOT communications network; the user can connect to Web Application by any equipment that can connect to Web Sever for remote controls function for the irrigation system via internet connection. The connection between water valve controllers uses the power line communication and low voltage electricity for cost savings and safety, respectively.

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

หน้า

รูปที่

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญตาราง

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องจากปัจจุบันการทำการเกษตรที่มีแปลงเกษตรขนาดใหญ่มีความยุ่งยากและประสบปัญหาในการใช้แรงงานจำนวนมากเพื่อดูแลการรดน้ำแปลงเกษตรที่มีจำนวนหลายแปลงซึ่งต้องอาศัยแรงงานเดินรดหรือควบคุมการเปิด-ปิดระบบน้ำทีละแปลง ซึ่งเป็นเหตุให้เสียค่าใช้จ่ายและเวลาโดยไม่จำเป็น ซึ่งด้วยสาเหตุมาจากที่แรงดันของปั๊มน้ำไม่เพียงพอที่จะจ่ายน้ำให้กับแปลงเกษตรพร้อมกันทุกแปลง

จากความสำคัญข้างต้นจึงเป็นที่มาของปริญญานิพนธ์ ระบบควบคุมการให้น้ำผ่านโครงข่ายการสื่อสาร NB-IoT เพื่อให้ระบบการให้น้ำทางการเกษตรมีการควบคุมที่มีประสิทธิภาพมากขึ้น โดยการพัฒนาเว็ปแอปพลิเคชันสำหรับควบคุมคำสั่งการให้น้ำขึ้นไปยังคลาวด์เซิร์ฟเวอร์เพื่อติดต่อกับ NB-IoT ให้ทำการให้น้ำตามที่ต้องการ เพื่อให้ง่ายและประหยัดเวลาในการควบคุมการจ่ายน้ำแต่ละแปลงโดยอุปกรณ์แต่ละตัวจะเชื่อมกันผ่านเทคโนโลยี Power Line Communication ซึ่งสามารถส่งข้อมูลไปกับไฟบ้านผ่านสายไฟโดยข้อดีคือการที่สามารถส่งข้อมูลไปได้ในระยะทางไกลซึ่งเหมาะกับพื้นที่ที่มีขนาดกว้าง

1.2 วัตถุประสงค์

1.เพื่อควบคุมระบบการให้น้ำให้เหมาะสมกับพืชแต่ละชนิดผ่านคลาวด์ ด้วยแอปพลิเคชันบนสมาร์ทโฟน

2.ศึกษาการทำงานและการใช้อุปกรณ์สื่อสาร NB-IoT ร่วมกับบอร์ด Arduino ในการเชื่อมต่อกับคลาวด์เซิร์ฟเวอร์

3.เพื่อศึกษาการรับส่งข้อมูลผ่านโครงข่าย Power Line Communication (PLC)

1.3 ขอบเขตของโครงงาน

1.ใช้อุปกรณ์ NB-IoT ในการเชื่อมต่อกับเซิร์ฟเวอร์ซึ่งสามารถเชื่อมต่อได้ระยะทางไกลและใช้พลังงานต่ำ

2.สามารถกำหนดปริมาณการให้น้ำเป็นปริมาตรซึ่งจะเป็นไปตามความเหมาะสมของพืชที่จะทำการให้น้ำ

3.สามารถใช้ Power Line Communication ในการสื่อสารระหว่างบอร์ด Arduino โดยจะสามารถสื่อสารกันได้ในระยะทางไกลผ่านสายไฟ 220 โวลต์

4.มีระบบตรวจสอบสถานะการเปิด-ปิดของวาล์วและสถานะตรวจสอบวาล์ววาล์วมีการทำงานขัดข้องหรือไม่

บทที่ 2
ทฤษฎีและบทความที่เกี่ยวข้อง

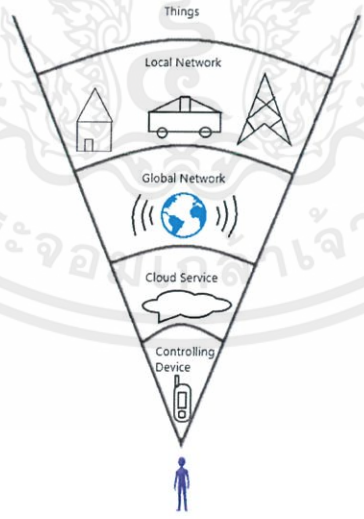
2.1 Internet of Things (IoT)

Internet of Things (IoT) เป็นการประยุกต์นำเอาอุปกรณ์ต่างๆมาเชื่อมต่อกันผ่านอินเตอร์เน็ต ทำให้สามารถตรวจสอบสถานะและควบคุมการทำงานของอุปกรณ์เหล่านั้นได้อย่างสะดวกผ่านอินเตอร์เน็ตดังรูปที่ 2.1

สำหรับเทคโนโลยีแบบไร้สายที่ใช้ในการเชื่อมต่ออุปกรณ์ IoT เข้าสู่ระบบอินเตอร์เน็ตพอจะแบ่งออกเป็นกลุ่มใหญ่ๆ ได้ 2 กลุ่มคือ

1. เทคโนโลยีไร้สายแบบระยะใกล้ (Short-range communication) ครอบคลุมในพื้นที่ที่มีระยะไม่กว้างมากนัก ได้แก่ เทคโนโลยี WIFI, Bluetooth, Zigbee, Z-wave, etc.

2. เทคโนโลยีไร้สายแบบระยะไกล (Long-range หรือ Wide area communication) ได้แก่ เทคโนโลยีที่ใช้ในระบบมือถือ (Cellular networks) เช่น GSM/GPRS (2G), UMTS (3G) และ LTE (4G)



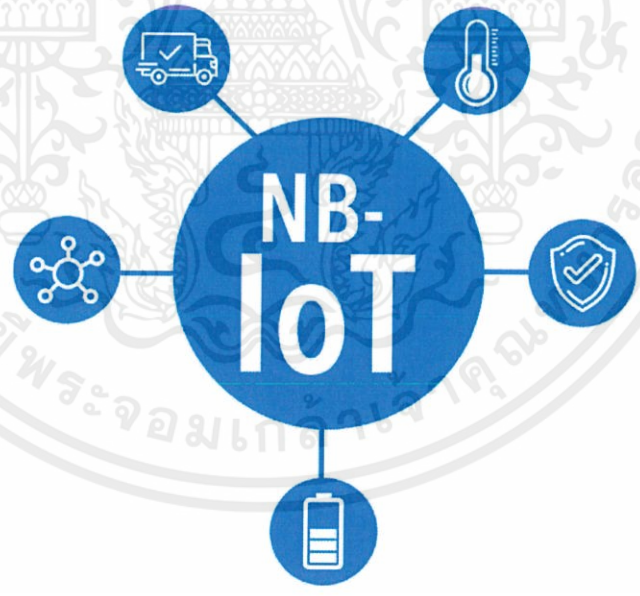
รูปที่ 2.1 Internet of Things

2.1.1 NB-IoT

Narrow Band IoT (NB-IoT) เป็นเทคโนโลยีการสื่อสารระยะไกลแบบใช้พลังงานต่ำที่มีการพัฒนาต่อยอดมาจากระบบ LTE (4G) เหมาะกับ application ที่ไม่ต้องการความเร็วในการส่งข้อมูลมากนัก เช่น Smart Parking หรือ Smart metering ดังรูปที่ 2.2

NB-IoT มีคุณสมบัติดังนี้

1. รองรับการสื่อสารโดยใช้พลังงานต่ำ รับการใช้งานแบตเตอรี่ของอุปกรณ์ IoT ได้นานถึง 10 ปี
2. เพิ่มประสิทธิภาพในการรับสัญญาณให้ดีขึ้น (รองรับสัญญาณได้ดีขึ้น 20 dB หรือเพิ่มพื้นที่ให้บริการได้ 10 เท่า) ทำให้สามารถครอบคลุมพื้นที่ให้บริการได้มากขึ้นรวมถึงบริเวณที่สัญญานมือถือปกติเข้าไปไม่ถึงเช่น ใต้ดิน ผนัง กำแพงก็สามารถติดตั้งใช้งานอุปกรณ์ IoT ได้
3. รองรับการเชื่อมต่อกับอุปกรณ์ IoT ได้เป็นจำนวนมาก (มากกว่าแสนอุปกรณ์ต่อสถานีฐาน)
4. เหมาะสำหรับการใช้งานอุปกรณ์ IoT ที่ต้องการความเร็วในการส่งสัญญาณไม่เกิน 200 kbps
5. ในระยะยาวอุปกรณ์จะมีราคาถูกกว่าอุปกรณ์ที่ใช้ 2G/3G/4G และ Cat-M1

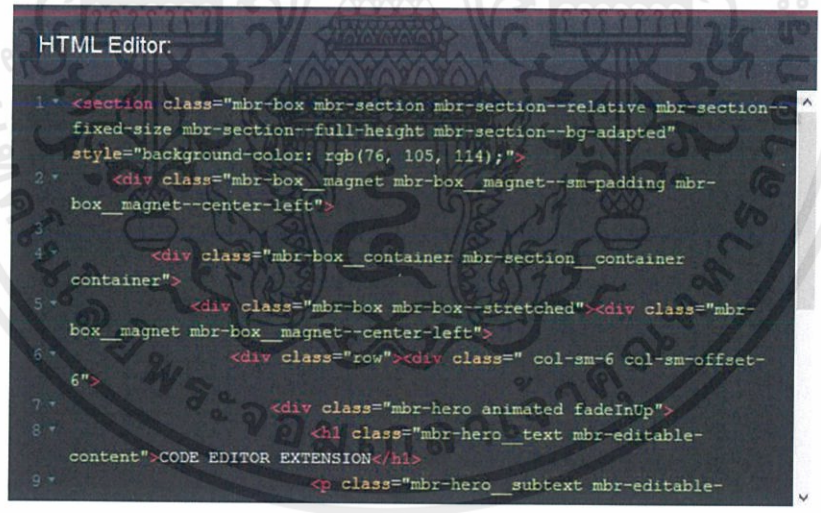


รูปที่ 2.2 NB-IoT

2.2 HTML

HTML ย่อมาจาก Hyper Text Markup Language คือภาษาคอมพิวเตอร์ที่ใช้ในการแสดงผลของเอกสารบน website หรือที่เราเรียกกันว่าเว็บเพจ ถูกพัฒนาและกำหนดมาตรฐานโดยองค์กร World Wide Web Consortium (W3C) และจากการพัฒนาทางด้าน Software ของ Microsoft ทำให้ภาษา HTML เป็นอีกภาษาหนึ่งที่ใช้เขียนโปรแกรมได้ หรือที่เรียกว่า HTML Application

HTML เป็นภาษาประเภท Markup สำหรับการการสร้างเว็บเพจ โดยใช้ภาษา HTML สามารถทำโดยใช้โปรแกรม Text Editor ต่างๆ เช่น Notepad, EditPlus หรือจะอาศัยโปรแกรมที่เป็นเครื่องมือช่วยสร้างเว็บเพจ เช่น Microsoft FrontPage, Dream Weaver ซึ่งอำนวยความสะดวกในการสร้างหน้า HTML ส่วนการเรียกใช้งานหรือทดสอบการทำงานของเอกสาร HTML จะใช้โปรแกรม web browser เช่น IE Microsoft Internet Explorer (IE), Mozilla Firefox, Safari, Opera, และ Netscape Navigator เป็นต้น ดังรูปที่ 2.3

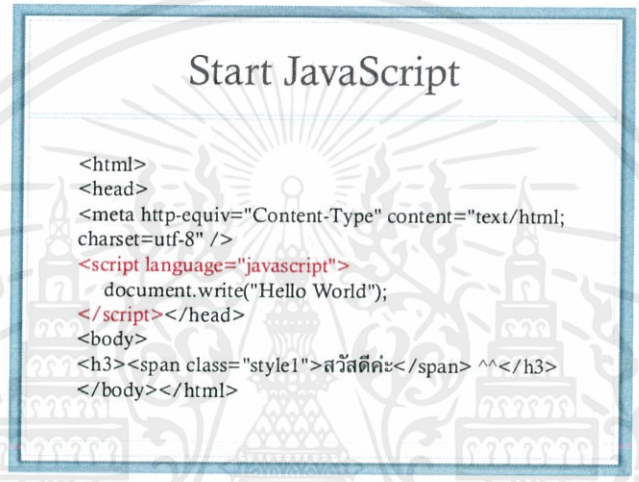


รูปที่ 2.3 HTML

2.3 javaScript

JavaScript คือ ภาษาคอมพิวเตอร์สำหรับการเขียนโปรแกรมบนระบบอินเทอร์เน็ต ที่กำลังได้รับความนิยมอย่างสูง Java JavaScript เป็น ภาษาสคริปต์เชิงวัตถุ (ที่เรียกกันว่า "สคริปต์"

(script) ซึ่งในการสร้างและพัฒนาเว็บไซต์ (ใช้ร่วมกับ HTML) เพื่อให้เว็บไซต์ของเราดูมีการเคลื่อนไหว สามารถตอบสนองผู้ใช้งานได้มากขึ้น ซึ่งมีวิธีการทำงานในลักษณะ "แปลความและดำเนินงานไปทีละคำสั่ง" (interpret) หรือเรียกว่า อ็อบเจ็กโอเรียลเต็ด (Object Oriented Programming) ที่มีเป้าหมายในการ ออกแบบและพัฒนาโปรแกรมในระบบอินเทอร์เน็ต สำหรับผู้เขียนด้วยภาษา HTML สามารถทำงานข้ามแพลตฟอร์มได้ โดยทำงานร่วมกับ ภาษา HTML และภาษา Java ได้ทั้งทางฝั่งไคลเอนต์ (Client) และ ทางฝั่งเซิร์ฟเวอร์ (Server) ดังรูปที่ 2.4



รูปที่ 2.4 JavaScript

2.3.1 Node.js

node.js คือ Programming language ที่ใช้โครงสร้างภาษา JavaScript ในการเขียนและมีการรันด้วย Chrome's V8 JavaScript engine (ตัวรัน java script ที่ Web Browser google chrome ใช้งานด้วย) โดยตอนเริ่มต้นก็คือ V8 โดยที่ Google พัฒนาเอาไว้ แล้วเปิดให้นำไปใช้กันได้เสรี เอาไปแปะกับงานส่วนไหนก็ได้ โดยที่มันจะรัน JavaScript ได้ และเร็วมาก ก็เลยมีคนพัฒนาเอามาทำเป็น server interpreter เพื่อที่จะได้มีภาษาที่เขียนและใช้งานบน server แบบเร็วๆ จึงเป็นที่มาของ node.js นั่นเอง ดังรูปที่ 2.5

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

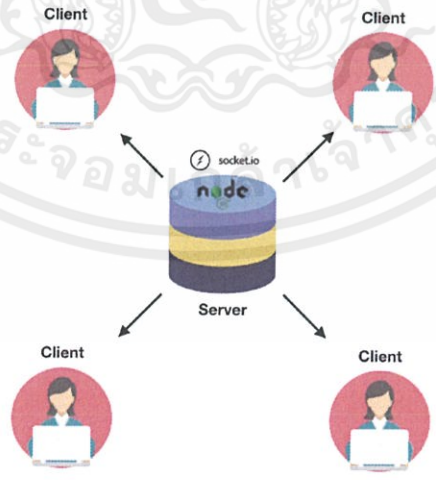
รูปที่ 2.5 Node.Js

2.4 Soket.io

Socket.io ดังรูปที่ 2.6 เป็น JavaScript frameworks ที่เอาไว้เรียกใช้งาน Websocket เพื่อคอยรับส่งข้อมูลจาก client และ server โดยที่ทำงานแบบ Real-time ตัวอย่างของการใช้งาน Websocket เช่น Chat room , Document collaboration คำสั่งหลักๆจะมีอยู่ด้วยกัน 2 แบบคือ

1.socket.on (ชื่อท่อ, callback) เมื่อได้รับข้อมูลมาทางท่อนี้ให้ทำอะไร

2.socket.emit (ชื่อท่อ, ข้อมูลที่จะส่ง) ส่งข้อมูลไปยังทุกๆ client ที่เชื่อมต่อกับท่อนี้



รูปที่ 2.6 Socket.io

2.5 CSS

CSS ย่อมาจาก Cascading Style Sheet ดังรูปที่ 2.7 คือ ภาษาที่ใช้ในการจัดรูปแบบเอกสาร HTML ให้มีความสวยงาม ได้แก่ สีของข้อความ สีพื้นหลัง ประเภทตัวอักษร และการจัดวางข้อความ

ประโยชน์ของ CSS

1.CSS มีคุณสมบัติมากกว่า tag ของ html เช่น การกำหนดกรอบให้ข้อความ รวมทั้งสี รูปแบบของข้อความ

2.CSS นั้นกำหนดที่ต้นของไฟล์ html หรือตำแหน่งอื่น ๆ ก็ได้ และสามารถมีผล กับเอกสารทั้งหมด หมายถึงกำหนด ครั้งเดียวจุดเดียวก็มีผลกับการแสดงผลทั้งหมด ทำให้เวลาแก้ไขหรือปรับปรุงทำได้สะดวก ไม่ต้องไล่ตามแก้ tag ต่างๆ ทั่วทั้งเอกสาร

3.CSS สามารถกำหนดแยกไว้ต่างหากจาก ไฟล์เอกสาร html และสามารถนำมาใช้ร่วม กับเอกสารหลายไฟล์ได้ การแก้ไขก็แก้เพียงจุดเดียวก็มีผลกับเอกสารทั้งหมด

รูปที่ 2.7 CSS

2.6 Ionic Framework

Ionic Framework ดังรูปที่ 2.8 คือเครื่องมือในการสร้าง HTML , CSS และ JavaScript เพื่อใช้ในการสร้าง Mobile Application ซึ่งสามารถใช้งานได้ค่อนข้างง่าย Ionic Framework เป็นเครื่องมือสร้างแอพมือถือที่สามารถสร้างทีเดียว สามารถใช้งานได้บนระบบปฏิบัติการ iOS, Android และ Windows ซึ่งก็จะใช้งานร่วมกับ Framework ตัวอื่น ๆ ได้ คือ Angular และ Apache Cordova

รูปที่ 2.8 Ionic Framework

2.7 Cloud Computing

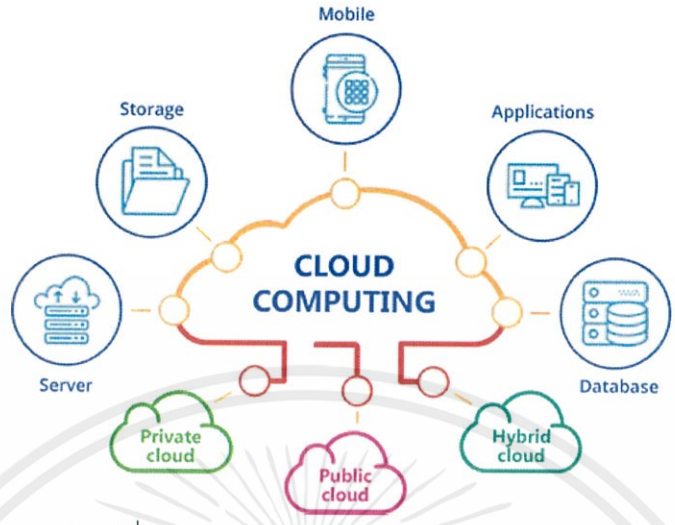
เป็นเสมือนคอมพิวเตอร์เครื่องหนึ่งที่เราใช้งานกันอยู่คือมีทั้งฮาร์ดแวร์และซอฟต์แวร์ที่ใช้ในการประมวลผลและเก็บข้อมูล แต่ Cloud Computing เป็นระบบคอมพิวเตอร์ที่ใหญ่มากๆ รองรับการใช้งาน การประมวลผล ตลอดจนการจัดเก็บข้อมูลได้อย่างมหาศาล Cloud Computing สามารถจำแนกตามการใช้งานได้ 3 ประเภทคือ

1. Infrastructure as a Service (IaaS) เป็นโครงสร้างพื้นฐานเหมือนกับระบบคอมพิวเตอร์คือ มีทั้งหน่วยประมวลผล ระบบเครือข่าย และพื้นที่จัดเก็บข้อมูลให้ใช้งาน

2. Platform as a Service (PaaS) เป็นการใช้งานเกี่ยวกับแพลตฟอรม์ต่างๆ เช่น การพัฒนาเว็บแอปพลิเคชั่นหรือโมบายแอปพลิเคชั่น เป็นต้น

3. Software as a Service (SaaS) เป็นการใช้งานทางด้านซอฟต์แวร์ ซึ่งภายใต้ระบบหรือเทคโนโลยี Cloud Computing จะมีซอฟต์แวร์เฉพาะด้าน เช่น ซอฟต์แวร์ทางบัญชี ซอฟต์แวร์การบริหารจัดการโรงแรม และอื่นๆ ซึ่งจำเป็นสำหรับธุรกิจประเภทต่างๆ ให้ใช้งาน

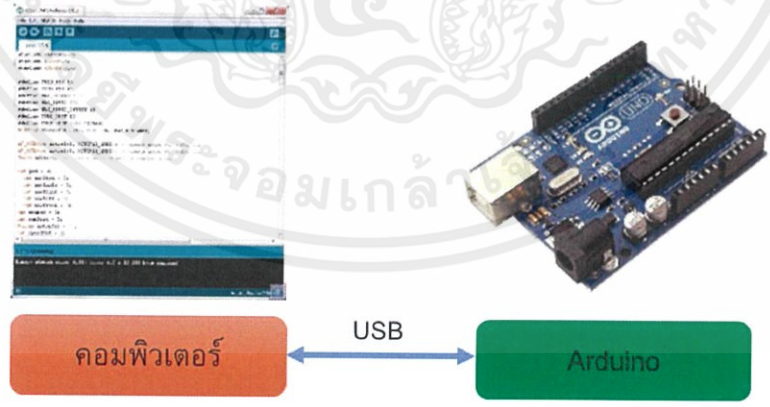
สรุปก็คือ Cloud Computing เป็นระบบคอมพิวเตอร์ที่พร้อมรองรับการทำงานของผู้ใช้งานในทุกๆ ด้านไม่ว่าจะเป็นระบบเครือข่าย การจัดเก็บข้อมูล การทดสอบระบบหรือติดตั้งฐานข้อมูล หรือการใช้งานซอฟต์เฉพาะด้านในธุรกิจต่างๆ โดยที่ผู้ใช้งานไม่ต้องติดตั้งระบบทั้งฮาร์ดแวร์และซอฟต์แวร์ไว้ที่สำนักงานให้ยุ่งยาก แต่สามารถใช้งานในสิ่งที่ต้องการได้ด้วยการเชื่อมต่อกับระบบ Cloud Computing ผ่านอินเทอร์เน็ตซึ่งมีประโยชน์ดังรูปที่ 2.9



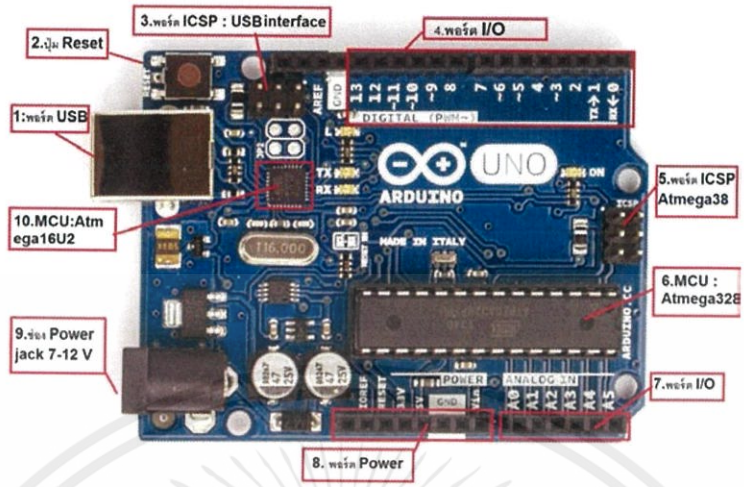
รูปที่ 2.9 ประโยชน์ของ Cloud Computing

2.8 บอร์ด Arduino

Arduino เป็นบอร์ดไมโครคอนโทรเลอร์ตระกูล AVR ที่มีการพัฒนาแบบ Open Source คือมีการเปิดเผยข้อมูลทั้งด้าน Hardware และ Software ตัวบอร์ด Arduino ถูกออกแบบมาให้ใช้งานได้ง่าย สามารถเชื่อมต่อกับคอมพิวเตอร์เพื่ออัพโหลดโปรแกรมและจ่ายไฟให้กับบอร์ดได้ด้วยสาย USB ดังรูปที่ 2.10 ดังนั้นจึงเหมาะสำหรับผู้เริ่มต้นศึกษา โดยผู้ใช้งานยังสามารถดัดแปลง พัฒนาต่อยอดได้ และบอร์ด Arduino ยังสามารถต่อกับอุปกรณ์และวงจรอิเล็กทรอนิกส์จากภายนอกแล้วเชื่อมต่อเข้ามาที่ขา I/O ของบอร์ด แสดงดังรูปที่ 2.11



รูปที่ 2.10 การเชื่อมต่อ Arduino กับ Computer



รูปที่ 2.11 Layout & Pin out Arduino Board (Model: Arduino UNO R3)

1. USBPort : ใช้สำหรับต่อกับ Computer เพื่ออัพโหลดโปรแกรมเข้า MCU และจ่ายไฟให้กับบอร์ด

2. Reset Button : เป็นปุ่ม Reset ใช้กดเมื่อต้องการให้ MCU เริ่มการทำงานใหม่

3. ICSP Port : ของ Atmega16U2 เป็นพอร์ตที่ใช้โปรแกรม Visual Com port บน Atmega16U2

4. I/Oport : Digital I/O ตั้งแต่ขา D0 ถึง D13 นอกจากนี้ บาง Pin จะทำหน้าที่อื่นๆ เพิ่มเติมด้วย เช่น Pin0,1 เป็นขา Tx,Rx Serial, Pin3,5,6,9,10 และ 11 เป็นขา PWM

5. ICSP Port : Atmega328 เป็นพอร์ตที่ใช้โปรแกรม Bootloader

6. MCU Atmega328 เป็น MCU ที่ใช้บนบอร์ด Arduino

7. I/O Port : นอกจากจะเป็น Digital I/O แล้ว ยังเปลี่ยนเป็น ช่องรับสัญญาณอนาล็อก ตั้งแต่ขา A0-A5

8. Power Port : ไฟเลี้ยงของบอร์ดเมื่อต้องการจ่ายไฟให้กับวงจรภายนอก ประกอบด้วยขาไฟเลี้ยง +3.3 V, +5V, GND, V_{IN}

9. Power Jack : รับไฟจาก Adapter โดยที่แรงดันอยู่ระหว่าง 7-12 V

10. MCU ของ Atmega16U2 เป็น MCU ที่ทำหน้าที่เป็น USB to Serial โดย Atmega328 จะติดต่อกับ Computer ผ่านAtmega16U2

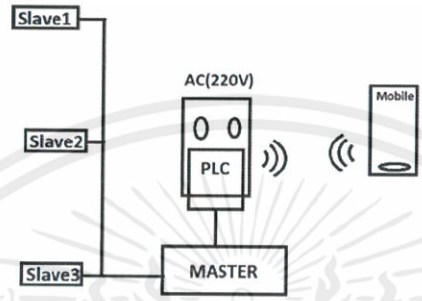
โปรแกรมภาษาของ Arduino จะใช้ภาษา C++ ซึ่งเป็นรูปแบบของโปรแกรมภาษาซีประยุกต์แบบหนึ่ง ที่มีโครงสร้างตัวภาษาโดยรวมใกล้เคียงกันกับ ภาษาซีมาตรฐาน (ANSI-C) อื่นๆ

เพียงแต่ได้มีการปรับปรุงรูปแบบในการเขียนโปรแกรมบางส่วนที่ผิดเพี้ยนไปจากANSI-C เล็กน้อย เพื่อช่วยลดความยุ่งยากในการเขียนโปรแกรมและให้ผู้เขียนโปรแกรมสามารถได้ง่ายและ สะดวกมากขึ้นกว่าการเขียนภาษาซีตามแบบมาตรฐานของ ANSI-C โดยตรงภาษาซี สามารถ นำไปใช้งานบนระบบฮาร์ดแวร์ที่มีความแตกต่างกันได้หลากหลาย โดยผู้ใช้เพียงแต่เลือกใช้ ตัวแปลคำสั่ง (C-Compiler) ให้ตรงกับระบบฮาร์ดแวร์ที่ใช้งานอยู่ ส่วนเรื่องรูปแบบในการเขียน โปรแกรมจะเป็นมาตรฐานเดียวกัน American National Standard Institute (ANSI) จงได้ทำการ ตั้งข้อกำหนดมาตรฐานของภาษาซีขึ้น โดยเรียกกันว่า "ANSI-C" เพื่อใช้เป็นข้อบังคับและคง มาตรฐานของภาษาซีไว้ไม่ให้เปลี่ยนแปลงไปจากเดิม โดยทุกผู้ผลิตจะต้องสร้างตัวแปลภาษาให้มี คุณสมบัติการใช้งานขั้นพื้นฐานเดียวกัน

2.9 เทคโนโลยี Power line Communication

เทคโนโลยี Power Line Communications (PLC) ดังที่แสดงในรูปที่ 2.12 เป็นเทคโนโลยีการติดต่อสื่อสาร ที่ทำให้สามารถส่งสัญญาณเสียง ข้อมูล และมัลติมีเดีย โดยผ่านระบบนำจ่ายกระแสไฟฟ้า หรือสายไฟฟ้าที่มีใช้ตามบ้านเรือนทั่วไป ทั้งที่เป็นระบบจ่ายไฟฟ้าแรงต่ำ (LV distribution cable) หรือระบบจ่ายไฟฟ้าแรงปานกลาง (MV distribution cable) โดยอาจมีการเรียกชื่อที่แตกต่างกันไป ไม่ว่าจะเป็น Power Line Communications (PLC), Power Line Telecommunications (PLT), Broadband over Power line (BPL) หรือ Ethernet over Power line และอาจมีการให้คำนิยามและรายละเอียดของเทคโนโลยีที่แตกต่างกันไปด้วย แต่ในที่นี้จะเรียก เทคโนโลยีในลักษณะนี้ทั้งหมดว่า เทคโนโลยีการสื่อสารผ่านสายไฟฟ้า(Power Line Communications - PLC) เทคโนโลยี PLC เป็นเทคโนโลยีที่ใช้สายไฟฟ้าในระบบจ่ายไฟฟ้าที่มีอยู่เดิม เพื่อให้บริการ รับ ส่ง ข้อมูลไม่ว่าจะเป็นข้อมูลความเร็วต่ำ (narrowband PLC) เช่น การควบคุมอุปกรณ์ไฟฟ้าภายในบ้าน การเฝ้าระวังรักษาความปลอดภัยภายในบ้าน และใช้ในการควบคุม สั่งการของหน่วยงานให้บริการสาธารณูปโภคด้านไฟฟ้าเองเช่น การควบคุมการทำงานของ switch gear (เพื่อปิด-เปิด อุปกรณ์ป้องกันระบบจ่ายไฟฟ้า) การอ่านมาตรวัดไฟฟ้าอัตโนมัติ (automatic meter reading - AMR) หรือการแจ้งอัตราค่าไฟฟ้า (tariff broadcast) เป็นต้น โดยพัฒนามาจากในระยะเริ่มแรกที่หน่วยงานให้บริการสาธารณูปโภคด้านไฟฟ้า (power utility providers) ใช้สายส่งแรงสูง เพื่อติดต่อสื่อสาร และใช้ในการควบคุมสถานีจ่ายไฟฟ้า (substation)

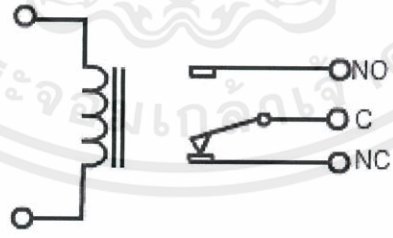
ระหว่างกัน และในปัจจุบัน ได้พัฒนาขีดความสามารถให้รับส่งข้อมูลความเร็วสูง (broadband PLC) เช่น high speed Internet, video streaming, VoIP ผ่านระบบจ่ายไฟฟ้าแรงต่ำได้ด้วย จึงสามารถใช้เป็นโครงข่ายส่วนเข้าถึงผู้ใช้บริการ (access network) ทดแทนคู่สายโทรศัพท์ได้



รูปที่ 2.12 เทคโนโลยี Power Line Communication

2.10 หลักการทำงานของ Relay

รีเลย์ (relay) คือ อุปกรณ์อิเล็กทรอนิกส์ที่ทำหน้าที่ ตัด-ต่อวงจร คล้ายกับสวิตซ์ โดยใช้หลักการหน้าสัมผัส และการที่จะให้รีเลย์ทำงานต้องจ่ายไฟให้อุปกรณ์ตามที่กำหนด เพราะเมื่อจ่ายไฟให้กับตัวรีเลย์ จะทำให้หน้าสัมผัสติดกัน กลายเป็นวงจรปิด และตรงข้ามทันทีที่ไม่ได้จ่ายไฟให้มัน รีเลย์จะกลายเป็นวงจรเปิด ไฟที่ใช้ป้อนให้กับตัวรีเลย์จะเป็นไฟที่มาจาก เพาเวอร์ฯ ของเครื่อง ดังนั้นทันทีที่เปิดเครื่อง จะทำให้รีเลย์ทำงาน โดยรีเลย์ที่ใช้งานเป็นรีเลย์ที่มีสัญลักษณ์ดังรูปที่ 2.13



รูปที่ 2.13 สัญลักษณ์ของรีเลย์

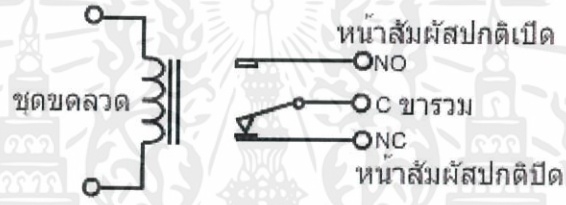
ประเภทของรีเลย์ แบ่งออกตามลักษณะการใช้งานได้เป็น 2 ประเภทคือ

1. รีเลย์กำลัง (power relay) หรือมักเรียกกันว่าคอนแทกเตอร์ (Contactor) ใช้ในการควบคุมไฟฟ้ากำลัง มีขนาดใหญ่กว่ารีเลย์ธรรมดา

2. รีเลย์ควบคุม (control Relay) มีขนาดเล็กกำลังไฟฟ้าต่ำ ใช้ในวงจรควบคุมทั่วไปที่มีกำลังไฟฟ้าไม่มากนัก หรือเพื่อการควบคุมรีเลย์หรือคอนแทกเตอร์ขนาดใหญ่

2.10.1 โครงสร้างของรีเลย์

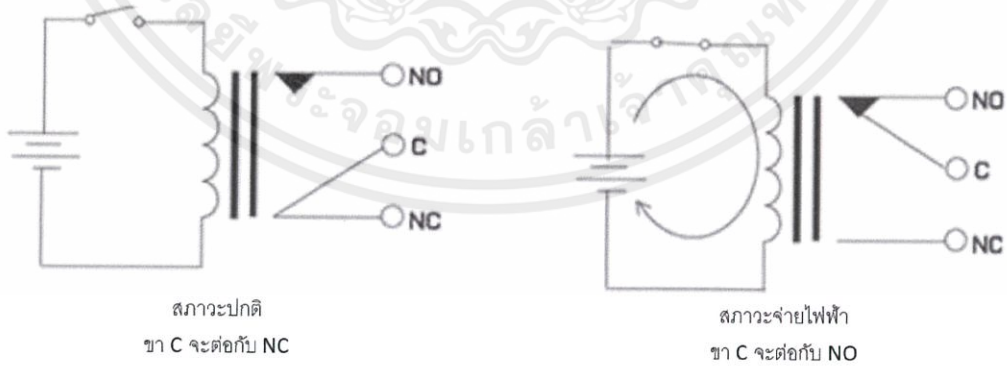
รูปที่ 2.14 ภายในโครงสร้างของ รีเลย์ จะประกอบไปด้วยขดลวด 1 ชุด และหน้าสัมผัส ซึ่งในหน้าสัมผัส 1 ชุด ซึ่งจะประกอบไปด้วย หน้าสัมผัสแบบปกติปิด (Normally Close หรือ NC.) ซึ่งในสภาวะปกติ ขานี้จะต่ออยู่กับขาร่วม (C) และ หน้าสัมผัสแบบปกติเปิด (Normally Open หรือ NO.) ขานี้จะต่อเข้ากับขาร่วม (C) เมื่อขดลวดมีแรงดันตกคร่อม หรือกระแสไหลผ่าน (ในปริมาณที่เพียงพอ) ใน รีเลย์ 1 ตัว อาจมีหน้าสัมผัสมากกว่า 1 ชุด ซึ่งขึ้นอยู่กับผู้ผลิต



รูปที่ 2.14 สัญลักษณ์ของรีเลย์แทนโครงสร้างรีเลย์

2.10.2 หลักการทำงานของรีเลย์

เมื่อมีกระแสไฟฟ้าไหลผ่านขดลวด จะทาให้ขดลวดเกิดสนามแม่เหล็กไปดึง แผ่นหน้าสัมผัสให้ดึงลงมา แตะหน้าสัมผัสอีกอันทาให้มีกระแสไหลผ่านหน้าสัมผัสไปได้ โดยรูปที่ 2.15 แสดงถึงตำแหน่งขาและรายละเอียดการใช้งาน



รูปที่ 2.15 สภาวะการทำงานของรีเลย์

องค์ประกอบตำแหน่งขาต่างๆของรีเลย์

1. ขาจ่ายแรงดันใช้งาน ซึ่งจะมีอยู่ 2 ขา จากรูปจะเห็นสัญลักษณ์ขดลวดแสดงตำแหน่งขา coil หรือ ขาต่อแรงดันใช้งาน

2. ขา C หรือ COM หรือ ขาคอมมอน จะเป็นขาต่อระหว่าง NO และ NC

3. ขา NO (Normally opened หรือ ปกติเปิด) โดยปกติขานี้จะเปิดเอาไว้ จะทำงานเมื่อเราป้อนแรงดันให้รีเลย์

4. ขา NC (Normally closed หรือ ปกติปิด) โดยปกติขานี้จะต่อกับขา C ในกรณีที่เราไม่ได้จ่ายแรงดัน หน้าสัมผัสของ C และ NC จะต่อถึงกัน

ข้อคำนึงในการใช้งานรีเลย์

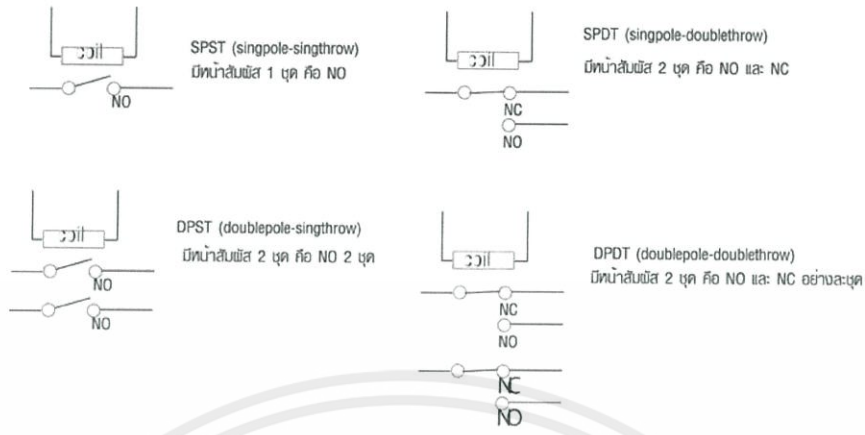
1. แรงดันใช้งาน หรือแรงดันที่ทำให้รีเลย์ทำงานได้ หากเราดูที่ตัวรีเลย์จะระบุค่าแรงดันใช้งานไว้ (หากใช้ในงานอิเล็กทรอนิกส์ ส่วนมากจะใช้แรงดันกระแสตรงในการใช้งาน) เช่น 12VDC คือต้องใช้แรงดันที่ 12 VDC เท่านั้นหากใช้มากกว่านี้ ขดลวดภายใน ตัวรีเลย์อาจจะขาดได้ หรือหากใช้แรงดันต่ำกว่ามาก รีเลย์จะไม่ทำงาน ส่วนในการต่อวงจรนั้นสามารถต่อขั้วใดก็ได้ เพราะตัวรีเลย์ จะไม่ระบุขั้วต่อไว้ (นอกจากชนิดพิเศษ)

2. การใช้งานกระแสผ่านหน้าสัมผัส ซึ่งที่ตัวรีเลย์จะระบุไว้ เช่น 10A 220AC คือหน้าสัมผัสของรีเลย์นั้นสามารถทนกระแสได้ 10 แอมแปร์ที่ 220VAC ครับ แต่การใช้ก็ควรจะใช้งานที่ระดับกระแสต่ำกว่านี้จะเป็นการดีกว่า เพราะถ้ากระแสมากหน้าสัมผัส ของรีเลย์จะละลายเสียหายได้

3. จำนวนหน้าสัมผัสการใช้งาน ควรดูว่ารีเลย์นั้นมีหน้าสัมผัสให้ใช้งานกี่อัน และมีขั้วคอมมอนด้วยหรือไม่

จำนวนหน้าสัมผัสของรีเลย์

ปกติแล้วรีเลย์จะมีหน้าสัมผัสและการเรียกจำนวนหน้าสัมผัสดังรูปที่ 2.16



รูปที่ 2.16 หน้าสัมผัสของรีเลย์

ประโยชน์ของรีเลย์

1. ทำให้ระบบส่งกำลังมีเสถียรภาพ (Stability) สูงโดยรีเลย์จะตัดวงจรเฉพาะส่วนที่เกิดผิดปกติ ออกเท่านั้น ซึ่งจะเป็นการลดความเสียหายให้แก่ระบบน้อยที่สุด

2. ลดค่าใช้จ่ายในการซ่อมแซมส่วนที่เกิดผิดปกติ

3. ลดความเสียหายไม่เกิดลุกลามไปยังอุปกรณ์อื่นๆ

4. ทำให้ระบบไฟฟ้าไม่ดับทั้งระบบเมื่อเกิดฟอลต์ขึ้นในระบบ

2.11 หลักการทำงานของ Solenoid Valves

โซลินอยด์วาล์ว (Solenoid Valve) คือ วาล์วที่ทำงานด้วยไฟฟ้าซึ่งมีทั้งชนิด 2/2, 3/2, 4/2, 5/2 และ 5/3 ในบทความนี้จะกล่าวถึงเฉพาะวาล์วชนิด 2/2 ซึ่งใช้ควบคุมการเปิดปิดของเหลวและก๊าซเท่านั้น ส่วนวาล์วชนิด 3/2, 4/2, 5/2 และ 5/3 ซึ่งส่วนใหญ่ใช้กับระบบนิวแมติคและระบบไฮดรอลิค

เมื่อกล่าวถึงชนิดของวาล์วเป็นตัวเลขเช่น 2/2, 4/2 หรือ 5/2 นั้น ตัวเลขหน้าบอกถึงจำนวนทางเข้าออกของวาล์ว นั้นๆ ว่ามีกี่ทางหรือมีกี่รู (port) ส่วนตัวเลขที่ตามหลังเครื่องหมายทับ (/) นั้นบอกถึงจำนวนสถานะ หรือ จำนวนตำแหน่ง (position) ของวาล์ว เช่น วาล์ว 2/2 ก็คือ วาล์วที่มี 2 ทาง และ มี 2 สถานะ คือ ปิดและเปิด ส่วนวาล์ว 5/2 ก็คือวาล์วที่มี 5 ทาง และมี 2 สถานะ เป็นต้น

การทำงานของโซลินอยด์วาล์ว 2/2 มีการควบคุมให้เปิดปิดได้ด้วย 3 ระบบคือ

1. ระบบเปิดปิดโดยตรง (Direct Acting หรือ Direct Operated)
2. ระบบเปิดปิดทางอ้อม (Indirect Acting หรือ Pilot Operated)
3. ระบบลูกผสม (Combined Acting หรือ Combined Operated)

2.11.1 ระบบเปิดปิดโดยตรง (Direct Acting หรือ Direct Operated)

ระบบเปิดปิดโดยตรงโซลินอยด์วาล์ว 2 ทางแบบปกติปิด (N/C) ดังรูปที่ 2.17 คือ ระบบของวาล์วที่นักศึกษาได้เลือกใช้งานซึ่งมีทางเข้าหนึ่งทางและทางออกหนึ่งทางทุ่น (plunger) จะมีซีลอยู่ปลายด้านล่างทำหน้าที่ เปิดและปิดรูทางผ่านของของไหลเมื่อจ่ายไฟฟ้าเข้าหรือตัดไฟฟ้าออกจากคอยล์

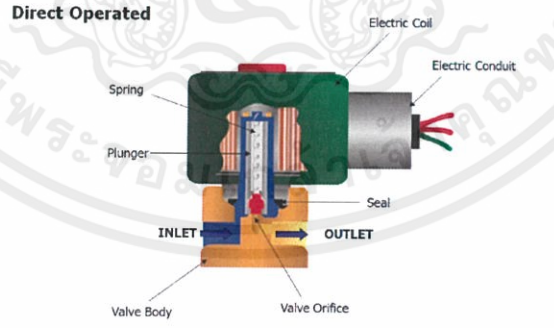
ข้อควรระวังในการใช้วาล์วที่ทำงานด้วยระบบนี้คือ

เมื่อมีการเพิ่มความดัน (pressure) ของของไหลในระบบจะทำให้ต้องใช้แรงมากขึ้นในการเปิดวาล์ว หากความดันของของไหลสูงกว่าที่กำลังของคอยล์จะเปิดวาล์วได้วาล์วนั้นก็จะไม่ทำงานถึงแม้จะมีการจ่ายไฟฟ้าแล้วก็ตาม แสดงดังรูปที่ 2.18

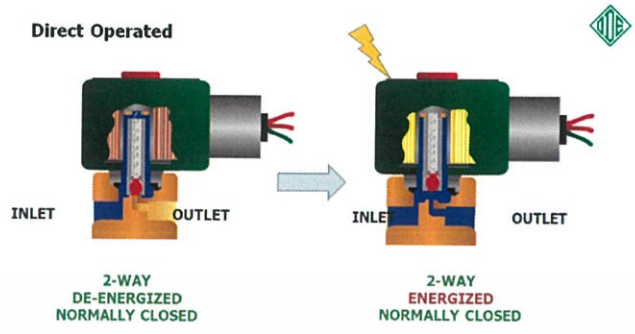
หลักการ: วาล์วเปิด-ปิดโดยอาศัยแรงจาก Coil และสปริงเพียงอย่างเดียว

ข้อดี: ไม่จำเป็นต้องอาศัยความดันของของไหลในการช่วยเปิด-ปิด

ข้อจำกัด: มักจะใช้กับวาล์วที่มีขนาดไม่ใหญ่นักส่วนมากจะอยู่ขนาด 1/8”-1/4”



รูปที่ 2.17 องค์ประกอบของวาล์วระบบเปิดปิดโดยตรง



รูปที่ 2.18 การทำงานของระบบวาล์วแบบปิดปิดโดยตรง

2.12 หลักการทำงานของ Water Flow Meter

การทำงานของ Water flow meter จะให้น้ำไหลผ่านใบพัดภายในให้เกิดการหมุน โดยมี hall effect sensor เป็นตัวตรวจจับการหมุน และส่งสัญญาณ pulse ออกมาทุกๆ รอบหมุน

2.12.1 การวัดอัตราการไหลน้ำ (volumetric flow rate)

ข้อมูลที่ได้จากสายสัญญาณ จะอยู่ในรูปของ digital pulse ที่ duty cycle คงที่ (50% ± 10%) ความถี่สัญญาณที่ออกมาจะแปรเปลี่ยนตามความเร็วการไหลของน้ำ โดยตัวเซนเซอร์ปล่อยสัญญาณ pulse ออกมา 1 ลูกเมื่อใบพัดภายในตัวมิเตอร์หมุนไป 1 รอบ ดังนั้นหลักการวัดอัตราการไหลน้ำจึงใช้การนับ จำนวน pulse ที่ถูกส่งออกมาใน 1 หน่วยเวลา

สำหรับรุ่น FS300A ที่อัตราการไหล 1 L/min สัญญาณที่ออกมามีความถี่ 5.5 Hz

$$Q = freq / coeff \quad (2.1)$$

- Q แทนอัตราการไหลในหน่วย (L/min)
- freq แทนจำนวน pulse ที่นับได้ใน 1 วินาที
- coeff แทนค่าคงที่ของความถี่ที่ปล่อยออกมาเทียบกับอัตราการไหล (ในที่นี้คือ 5.5 Hz per L/min)

สำหรับการหา freq นั้น จะใช้จำนวน pulse ที่นับได้ใน 1 วินาที ในการหา (หน่วย pulse/sec) สามารถกำหนด stepTime การนับ pulse เองได้ (ขึ้นกับความละเอียดที่ต้องการ)โดยเวลาจริงที่บอร์ดนับ pulse ได้ (delTime) จะคาดเคลื่อนจาก stepTime ที่ตั้งไว้เล็กน้อยตามการ

ประมวลของบอร์ด ดังนั้นจำนวน pulse ที่อ่านได้จะต้องนำมาเทียบบัญญัติไตรยางศ์ให้ได้จำนวน pulse เทียบเท่าที่นับได้ใน 1 วินาที (จะได้หน่วย pulse/sec พอดี) ดังนี้

$$freq = 1000 * pulseCount / delTime \quad (2.2)$$

ในการทดลองว่า stepTime ที่เหมาะสม (จากการทดสอบ) 1.5-1.7 วินาที ต่อการนับ pulse แต่ละครั้ง

2.13 Adapter

AC หมายถึง ไฟฟ้ากระแสสลับ และ DC หมายถึง ไฟฟ้ากระแสตรง แรงดันและสัญญาณไฟฟ้าได้อ้างอิงถึง AC และ DC ด้วยถึงแม้จะไม่ใช่กระแส สัญญาณไฟฟ้า คือแรงดันหรือกระแสซึ่งเป็นพาหะของข้อมูลข่าวสาร โดยทั่วไปจะเป็นแรงดัน แต่ก็สามารถใช้เรียกได้ทั้งกระแสและ แรงดันในวงจร

2.13.1 ไฟฟ้ากระแสสลับ (AC)

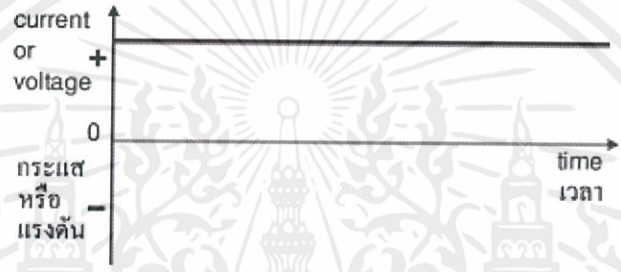
ไฟฟ้ากระแสสลับ (AC) มีลักษณะการไหลทางเดียวแต่สลับทิศอย่างต่อเนื่อง แรงดันกระแสสลับมีการเปลี่ยนอย่างต่อเนื่องระหว่างบวกและลบอัตราการเปลี่ยนทิศทางเรียกว่าความถี่ของไฟกระแสสลับ มีหน่วยวัดเป็นเฮิร์ท (Hz) ซึ่งก็คือจำนวนรอบคลื่นต่อหนึ่งวินาที ไฟฟ้าหลักในประเทศไทยใช้ความถี่ 50Hz แหล่งจ่ายไฟกระแสสลับเหมาะสำหรับจ่ายกำลังให้อุปกรณ์บางอย่าง เช่น หลอดไฟและเครื่องกำเนิดความร้อน แต่วงจรอิเล็กทรอนิกส์ส่วนใหญ่ต้องการเลี้ยงด้วยไฟกระแสตรงคงที่ รูปที่ 2.19 แสดงถึงสัญญาณคลื่นไซน์ของไฟกระแสสลับจาก power supply

เอซี(AC)จากแหล่งจ่ายกำลัง

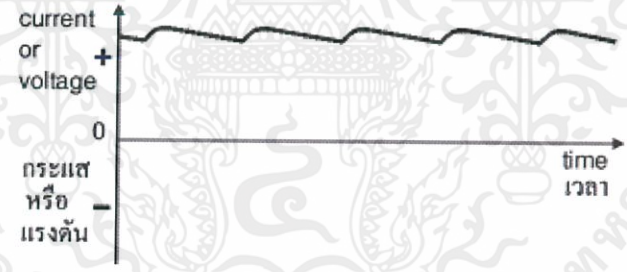
รูปที่ 2.19 สัญญาณคลื่นไซน์ของไฟกระแสสลับจาก power supply

2.13.2 ไฟฟ้ากระแสตรง (DC)

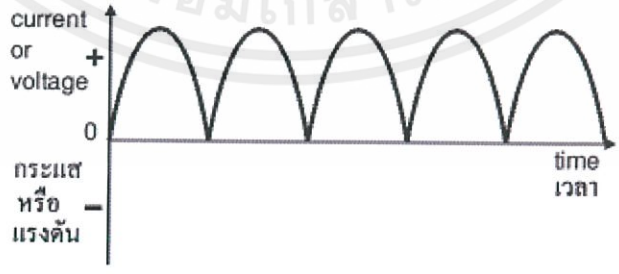
ไฟฟ้ากระแสตรง (DC) ไหลไปทิศทางเดียวแต่อาจจะเพิ่มขึ้นหรือลดลง แรงดันกระแสตรงเป็นบวกหรือเป็นลบก็ได้ แต่อาจจะเพิ่มขึ้นหรือลดลงโดยวงจรอิเล็กทรอนิคส์ปกติต้องเลี้ยงด้วยไฟกระแสตรงสม่ำเสมอและคงที่ที่ค่าหนึ่งหรือไฟกระแสตรงที่เรียบมีค่าเปลี่ยนแปลงที่เรียกว่าริบเปิ้ลเพียงเล็กน้อย โดยไฟกระแสตรงแบ่งเป็น 3 ลักษณะ คือ ไฟกระแสตรงแบบสม่ำเสมอ (Steady), ไฟกระแสตรงเรียบ (Smooth) และไฟกระแสตรงแบบไม่เรียบ (Varying) ดังรูปที่ 2.20 รูปที่ 2.21 รูปที่ 2.22 ตามลำดับ



รูปที่ 2.20 ไฟกระแสตรงแบบสม่ำเสมอ (Steady)



รูปที่ 2.21 ไฟกระแสตรงแบบเรียบ (Smooth)

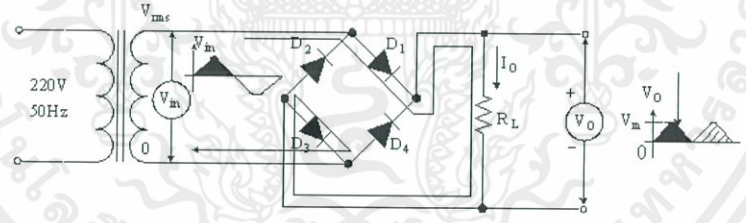


รูปที่ 2.22 ไฟกระแสตรงแบบไม่เรียบ (Varying)

เซลล์แบตเตอรีและแหล่งจ่ายกำลังแบบคุมค่าให้ไฟกระแสตรงแบบสม่ำเสมอ ซึ่งเป็น DC ในอุดมคติสำหรับวงจรอิเล็กทรอนิคส์แหล่งจ่ายกำลังประกอบด้วยหม้อแปลง ซึ่งทำหน้าที่แปลงไฟกระแสสลับหลักให้ได้แรงดันกระแสสลับที่เหมาะสม จากนั้นก็แปลงไฟกระแสสลับให้เป็นไฟกระแสตรงด้วยตัวเรียงกระแสแบบบริดจ์ แต่ไฟที่ได้ยังไม่เรียบและไม่เหมาะที่จะใช้กับวงจรอิเล็กทรอนิกส์ แหล่งจ่ายกำลังบางแบบจะมีตัวเก็บประจุเพื่อกรองไฟให้เรียบ ซึ่งเหมาะสำหรับใช้กับวงจรอิเล็กทรอนิกส์ที่มีความไวน้อยรวมทั้งใช้กับโครงงานส่วนใหญ่ของเราหลอดไฟ ตัวทำความร้อนและมอเตอร์ ทำงานด้วยไฟเลี้ยงกระแสตรงได้

2.13.3 วงจรเรียงกระแสเต็มคลื่นแบบบริดจ์ (Bridge rectifier)

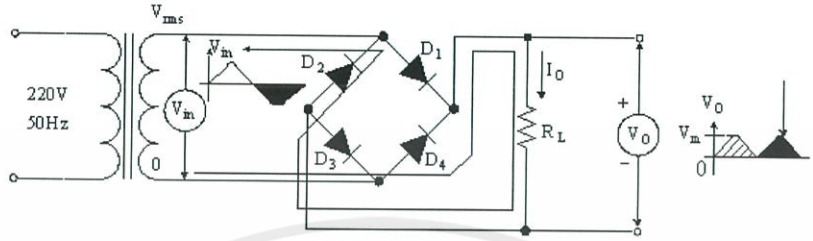
วงจรเรียงกระแสเต็มคลื่นแบบบริดจ์เป็นวงจรที่นิยมใช้กันมากอีกแบบหนึ่งโดยจะใช้ไดโอดสี่ตัวต่อกันในลักษณะบริดจ์โดยไดโอดสองตัวจะทำหน้าที่ในการเรียงกระแสครึ่งไซเกิลบวกและอีกสองตัวเรียงกระแสในครึ่งไซเกิลลบ ค่าแรงดันไฟตรงที่ได้ทางเอาต์พุตจะมีลักษณะเหมือนกับวงจรเรียงกระแสแบบเต็มคลื่นใช้หม้อแปลงมีแท็ปกลางจะต่างกันก็แต่วิธีการในการต่อวงจรเพื่อใช้งาน อีกทั้งการเลือกไดโอดมาต่อใช้งานสามารถเลือกใช้ไดโอดแต่ละตัวทนแรงดันได้เพียงแค่ครึ่งหนึ่งของแรงดันไฟฟ้าที่จ่ายให้กับวงจร ทั้งนี้เนื่องจากไดโอดสองตัวจะแบ่งแรงดันกันตัวละครึ่งหนึ่งของแหล่งจ่าย



รูปที่ 2.23 วงจรเรียงกระแสเต็มคลื่นแบบบริดจ์สัญญาณครึ่งไซเกิลบวกผ่าน

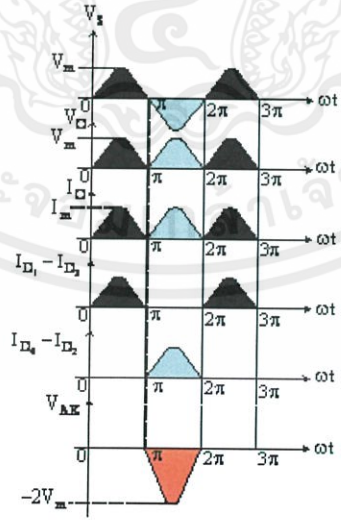
การทำงานของวงจรในรูปที่ 2.23 เมื่อสัญญาณอินพุตครึ่งไซเกิลด้านบวกปรากฏที่ด้าน V_{rms} สัญญาณครึ่งไซเกิลบวกจะผ่านไดโอด D_1 ไปตกคร่อมโหลดความต้านทาน (R_L) และมีทิศทางไหลจากขั้วด้านบน (ขั้วบวก) ผ่านความต้านทาน R_L ไปหาขั้วด้านล่าง (ขั้วลบ) จากนั้นก็จะไหลผ่านไดโอด D_3 และครบวงจรที่ขั้วอีกด้านหนึ่งของหม้อแปลง (ด้าน 0) ซึ่งในขณะนี้ไดโอด D_1 และ D_3 จะมีแรงดันตกคร่อมน้อยมากเพราะว่าอยู่ในลักษณะไบแอสตรง ดังนั้นไดโอด D_1 และ D_3 จึงทำ

หน้าที่เสมือนเป็นสวิตช์ต่อวงจร รูปคลื่นสัญญาณที่ตกคร่อมโหลดตัวต้านทาน R_L (V_o) จึงถูกจัดเรียงไว้ทางด้านบวกของเอาต์พุต นั่นคือเกิดการไหลของกระแสผ่านได้ทิศทางเดียว



รูปที่ 2.24 วงจรเรียงกระแสเต็มคลื่นแบบบริดจ์สัญญาณครึ่งไซเกิลลบผ่าน

พิจารณาวงจรรูปที่ 2.24 เมื่อสัญญาณด้านครึ่งไซเกิลบวกหมดไปและปรากฏสัญญาณด้านครึ่งไซเกิลลบ หมายความว่าขณะนี้ ขั้วหม้อแปลงที่เป็นอินพุตของวงจรเรียงกระแสด้านที่เป็นศูนย์จะปรากฏมีแรงดันไฟเป็นบวกขณะที่ขั้วด้านบน (V_{rms} เดิม) จะมีค่าเป็นศูนย์แทนสลับกันกับรูปที่ 2.23 ดังนั้นไดโอด D_4 จะเสมือนได้รับไบแอสตรง สัญญาณรูปคลื่นของกระแสและแรงดันจะไหลผ่านไดโอด D_4 ไปตกคร่อมโหลดตัวต้านทาน R_L และสังเกตได้ว่าจะมีทิศทางไหลจากขั้วด้านบน (ขั้วบวก) ผ่านความต้านทาน R_L ไปหาขั้วด้านล่าง (ขั้วลบ) เหมือนกันกับเมื่อครั้งครึ่งไซเกิลบวกผ่าน หลังจากนั้นจะไหลผ่านไดโอด D_2 ครบวงจรที่ขั้วอีกด้านหนึ่งของหม้อแปลง รูปคลื่นสัญญาณที่ตกคร่อมโหลดตัวต้านทาน R_L(V_o) จึงถูกจัดเรียงไว้ด้านเดียวกัน คือให้กระแสผ่านได้ทิศทางเดียว



รูปที่ 2.25 รูปคลื่นสัญญาณที่ส่วนต่างๆของวงจรเรียงกระแสเต็มคลื่นแบบบริดจ์

จากรูปที่ 2.25 เป็นการแสดงรูปคลื่นสัญญาณที่ส่วนต่างๆของวงจรเรียงกระแสเต็มคลื่นแบบบริดจ์ตลอดทั้งรอบของรูปคลื่นสัญญาณ ซึ่งประกอบด้วยสัญญาณอินพุตและสัญญาณเอาต์พุตโดยพิจารณาในหนึ่งรอบ คือตั้งแต่ระยะ 0 - 2π เพราะในทุกๆรอบจะมีระยะและสัญญาณที่เท่ากัน จะเห็นว่าที่ระยะ 0 – π สัญญาณไซน์ด้านอินพุตเป็นสัญญาณด้านบวก ซึ่งที่ผ่านมาในรูปที่ 2.24 นั้น ไดโอด D_1 และ D_3 จะทำหน้าที่เสมือนสวิตช์ต่อวงจรให้สัญญาณผ่านไปยังโหลด R_L และสัญญาณด้านเอาต์พุต (V_o) ก็จะไปปรากฏที่โหลด R_L ตั้งแต่ระยะ 0 – π เช่นกันเพราะถือว่าแรงดันตกคร่อมที่ไดโอดมี D_1 และ D_3 ในขณะนี้มีค่าน้อยมากเหมือนสวิตช์ทั่วไปจนอาจตัดทิ้งออกไปได้ และเมื่อพิจารณาที่โหลดเป็นตัวต้านทาน สัญญาณรูปคลื่นของกระแสที่ได้ทางด้านเอาต์พุต (I_o) ซึ่งขณะนี้ก็คือ ($I_{D1} - I_{D3}$) ก็จะมีรูปร่างและมีระยะเดียวกันกับสัญญาณแรงดันเอาต์พุตหรืออินเฟสกันนั่นเอง

ต่อมาเมื่อสัญญาณไซน์ด้านอินพุตปรากฏด้านลบซึ่งที่ผ่านมาในรูปที่ 2.24 นั้น ไดโอด D_4 และ D_2 จะเปลี่ยนมาทำหน้าที่เสมือนสวิตช์ต่อวงจรให้สัญญาณผ่านไปยังโหลด R_L และสัญญาณด้านเอาต์พุต (V_o) ก็จะไปปรากฏที่โหลด R_L ตั้งแต่ระยะ 0 - 2π เช่นกันและค่าแรงดันตกคร่อมที่ไดโอดมี D_4 และ D_2 ขณะนี้ก็มีค่าน้อยมากเหมือนสวิตช์ทั่วไปเช่นกันจนตัดทิ้งออกไปได้ อีกทั้งโหลดยังคงเป็นตัวต้านทานตัวเดิม สัญญาณรูปคลื่นของกระแสที่ได้ทางด้านเอาต์พุต (I_o) ซึ่งขณะนี้ก็คือ ($I_{D4} - I_{D2}$) ก็จะมีรูปร่างและอินเฟสกันเหมือนกับช่วง 0 – π เช่นกัน สัญญาณที่เกิดในระยะ 0 – π และระยะ π - 2π จะถูกกำหนดทิศทางโดยลักษณะการต่อวงจรให้ไปปรากฏต่อเนื่องทางด้านเดียวกันซึ่งพิจารณาตามรูปที่ 2.25 ก็คืออยู่ด้านบวกนั่นเอง

ค่าแรงดันไฟตรงเฉลี่ยที่ได้ทางด้านเอาต์พุตจะมีลักษณะเหมือนกับของวงจรเรียงกระแสเต็มคลื่นแบบใช้หม้อแปลงมีแท็ปกลาง นั่นคือ

$$\begin{aligned} V_{O_{avg}} &= \frac{2}{2\pi}\int_0^{\pi} V_m \sin\omega t.\, d\omega t \\ &= \frac{V_m}{\pi}\int_0^{\pi} \sin\omega t.\, d\omega t \\ &= \frac{V_m}{\pi}[-\cos\omega t]_0^{\pi} \\ &= \frac{V_m}{\pi}[-(\cos\pi - \cos 0)] \\ &= \frac{V_m}{\pi}[-(-1-1)] \end{aligned}$$

$$V_{O_{avg}} = \frac{2V_m}{\pi} = 0.637V_m \qquad (2.3)$$

$$I_{O_{avg}} = \frac{V_{O_{avg}}}{R_L} = \frac{0.637V_m}{R_L} \quad (2.4)$$

$$V_{O_{rms}} = \sqrt{\frac{2}{2\pi}\int_0^{\pi} V_m^2 \sin^2\omega t.\, d\omega t}$$

$$\begin{aligned} V_{O_{rms}}^2 &= \frac{V_m^2}{\pi}\int_0^{\pi} \sin \omega t.\, d\omega t \\ &= \frac{V_m^2}{2\pi}\int_0^{\pi} \frac{1}{2}(1-\cos 2\omega t)\, d\omega t \\ &= \frac{V_m^2}{2\pi}\left(\int_0^{\pi} d\omega t - \int_0^{\omega} \cos 2\omega t\, .\, d\omega t\right) \\ &= \frac{V_m^2}{2\pi}\left[\omega t\int_0^{\pi} -\frac{1}{2}\int_0^{\pi} \cos 2\omega t\, .\, d\omega t\right] \\ &= \frac{V_m^2}{2\pi}\left[\pi - \frac{1}{2}(\sin 2\omega t)_0^{\pi}\right] \\ &= \frac{V_m^2}{2\pi}\left[\pi - \frac{1}{2}(\sin 2\pi - \sin 0)\right] \end{aligned}$$

$$\therefore V_{O_{rms}}^2 = \frac{V_m^2}{\pi}$$

$$V_{O_{rms}} = \frac{V_m}{\sqrt{2}} = 0.707V_m \quad (2.5)$$

$$I_{O_{rms}} = \frac{V_{O_{rms}}}{R_L} = \frac{0.707V_m}{R_L} \quad (2.6)$$

V_{Oavg} = ค่าแรงดันไฟตรงเฉลี่ยทางด้านเอาต์พุต มีหน่วยเป็นโวลต์ (V)

I_{Oavg} = ค่ากระแสไฟตรงเฉลี่ยที่ไหลผ่านตัวต้านทาน มีหน่วยเป็นแอมแปร์ (A)

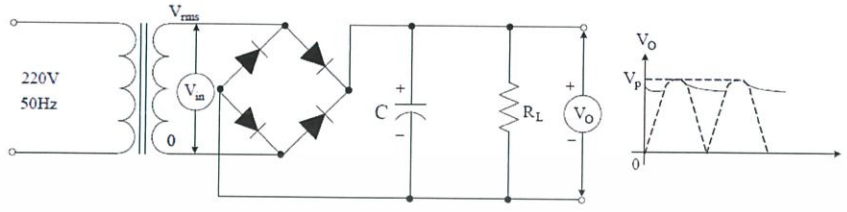
V_{Orms} = ค่าแรงดันไฟฟ้าใช้งานด้านเอาต์พุต มีหน่วยเป็นโวลต์ (V)

I_{Orms} = ค่ากระแสไฟฟ้าใช้งานที่ไหลผ่านตัวต้านทาน มีหน่วยเป็นแอมแปร์ (A)

$V_m = V_p$ = ค่าแรงดันสูงสุด มีหน่วยเป็นโวลต์ (V) = 1.414 V_{rms}

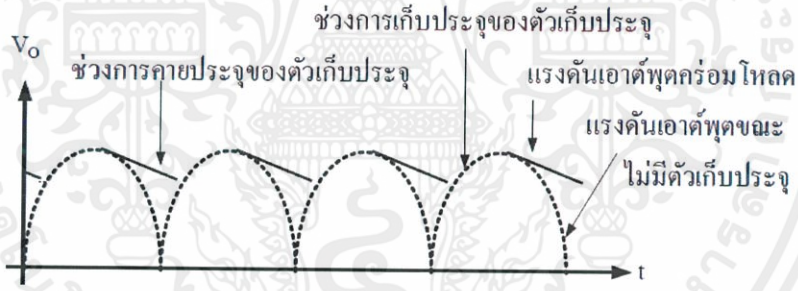
V_{rms} = ค่าแรงดันใช้งานด้านเอาต์พุตของหม้อแปลง มีหน่วยเป็นโวลต์ (V)

2.13.4 การกรองสัญญาณด้วยตัวเก็บประจุ (Capacitor filter)



รูปที่ 2.26 การกรองสัญญาณเอาต์พุตด้วยตัวเก็บประจุ

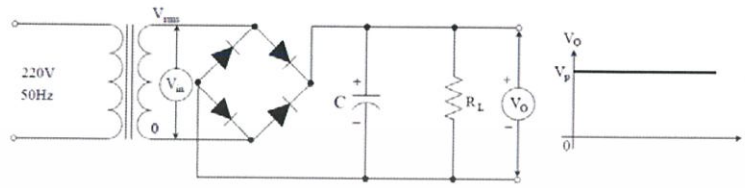
การกรองสัญญาณด้านเอาต์พุตเพื่อให้ได้แรงดันไฟที่เรียบสม่ำเสมอเป็นไฟตรงที่สมบูรณ์ ตามรูปที่ 2.26 เป็นการแสดงให้เห็นสัญญาณที่ได้ทางเอาต์พุตในภาพรวมทั้งในขณะไม่ได้ต่อตัวเก็บประจุและในขณะเมื่อต่อตัวเก็บประจุเพื่อทำการกรองสัญญาณทางเอาต์พุต ซึ่งยังคงนิยมใช้ตัวเก็บประจุแบบอิเล็กโตรไลต์มาทำ การกรองสัญญาณ เนื่องจากติดตั้งใช้งานได้ง่ายและกินพื้นที่น้อยกว่าการกรองสัญญาณด้วยวิธีอื่น



รูปที่ 2.27 การกรองสัญญาณเอาต์พุตด้วยตัวเก็บประจุที่มีขนาดยังไม่เหมาะสม

พิจารณารูปที่ 2.27 ถ้าหากเลือกใช้ตัวเก็บที่มีขนาดไม่เหมาะสม ค่าความจุอาจน้อยเกินไป ลักษณะสัญญาณที่ได้ทางเอาต์พุตจะยังไม่เรียบเป็นเส้นตรงยังคงมีการกระเพื่อมของสัญญาณอยู่ อย่างไรก็ตามจะเห็นว่าการคายประจุจะเป็นไปอย่างช้าๆ เปรียบเทียบกับสัญญาณแรงดันเอาต์พุต ขณะไม่มีตัวเก็บประจุแล้วแรงดันเอาต์พุตคร่อมโหลดจะไม่ลดลงมาถึงค่าศูนย์ในขณะที่แรงดันเอาต์พุตขณะไม่มีตัวเก็บประจุจะลดลงมาถึงค่าศูนย์ ถ้าทำการเพิ่มค่าความจุของตัวเก็บประจุให้มีค่า มากขึ้น สัญญาณเอาต์พุตที่ได้ก็จะยิ่งมีความเรียบมากขึ้น ค่าความชันในช่วงของการคายประจุของ ตัวเก็บประจุก็จะมีค่าลดลง และถ้าเพิ่มค่าความจุให้เหมาะสมจะได้สัญญาณ

เอาต์พุตเป็นเส้นตรง อย่างไรก็ตามถ้าเพิ่มค่าความจุมากเกินไปก็อาจเป็นอันตรายต่อวงจรเรียงกระแส เพราะจะมีกระแส ไหลในวงจรมากขึ้นจนอาจทำให้เกิดการลัดวงจรขึ้นได้



รูปที่ 2.28 วงจรกรองสัญญาณที่ใช้ตัวเก็บประจุเหมาะสม

จากรูปที่ 2.28 จะเห็นว่าการกรองสัญญาณให้เรียบ ตัวเก็บประจุ (Capacitor) จะทำหน้าที่กรองสัญญาณที่ค่าแรงดันสูงสุดหรือที่ค่ายอดคลื่น (Peak voltage , VP หรือ Maximum voltage , Vm) ถ้าใช้ตัวเก็บประจุถูกขนาดและเหมาะสม ค่าสัญญาณแรงดันไฟตรงที่ ได้ทางเอาต์พุตจะมีความเรียบเป็นเส้นตรง ทำให้การกระเพื่อม (Ripple) มีค่าน้อยมากหรือแทบไม่มี

ดังนั้น สิ่งที่จะละทิ้งไม่ได้ในการเลือกใช้ตัวเก็บประจุ (Capacitor) เพื่อกรองสัญญาณ คือ ขนาดของตัวเก็บประจุต้องถูกต้องเหมาะสมเพื่อไม่ให้เกิดอันตรายต่อวงจรหลักในการพิจารณาเลือกใช้ตัวเก็บประจุก็ยังเหมือนในวงจรเรียงกระแสต่างๆที่ผ่านมาคือ

1. ต้องทราบขนาดค่าแรงดันใช้งานด้านเอาต์พุตของหม้อแปลงและแรงดันไฟตรงด้านเอาต์พุตที่ต้องการใช้งาน

2. ต้องทราบกระแสพิกัดสูงสุดด้านเอาต์พุตของหม้อแปลงที่สามารถจ่ายได้

2.14 วงจรรักษาระดับแรงดัน (Voltage Regulator Circuit)

2.14.1 วงจรเรกกูเลเตอร์โดยใช้ไอซี 3 ขาแบบแรงดันเอาต์พุตคงที่

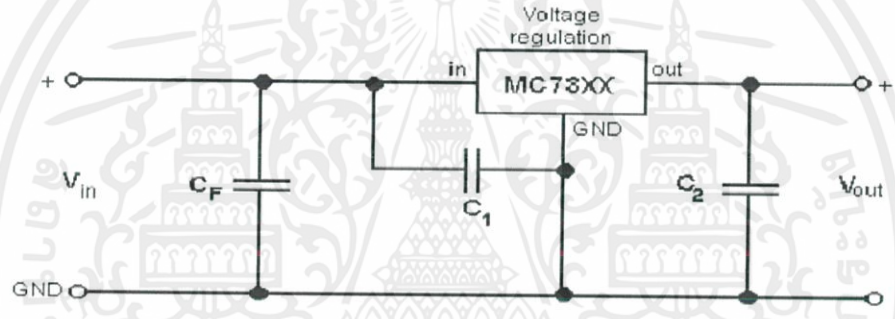
ไอซี 3 ขาแบบแรงดันเอาต์พุตคงที่นิยมใช้กันมากคือตระกูล MC78xx และตระกูล MC79xx โดยตระกูล 78xx จะใช้แรงดันแบบบวกที่คงที่ ส่วนตระกูล 79xx จะให้แรงดันแบบลบที่คงที่โดยที่ xx จะบอกขนาดแรงดัน ตัวอย่างเช่น MC7805 คือไอซีตระกูล MC78xx ที่ให้แรงดันแบบบวกคงที่ ขนาด 5 V และ MC7912 คือไอซีตระกูล MC79xx ที่ให้แรงดันแบบลบคงที่ ขนาด 12 V

IC ตระกูล MC78xx และ MC79xx จะมีลักษณะที่ใกล้เคียงกันมากจะแตกต่างกันเพียงการให้แรงดันที่บวกหรือลบเท่านั้น ฉะนั้นในหัวข้อนี้จะกล่าวเกี่ยวกับการออกแบบไอซี MC78xx

2.14.2 วงจรเรกกูเลเตอร์พื้นฐานโดยใช้ไอซีตระกูล MC78xx

ในรูปที่ 2.29 แสดงวงจรมาตรฐานของแหล่งจ่ายไฟแบบแรงดันเอาต์พุตคงที่โดยใช้เป็นวงจรเรกกูเลเตอร์ จะเห็นได้ว่าขา in ของ MC78xx จะต่อกับไฟบวกส่วนขา GND จะต่อกับไฟลบ ใส่ไว้เพื่อ C_1 ลดความเหนี่ยวนำภายในไอซี ซึ่งมักใช้ค่า 1 µF แบบแทนทาลัมหรือ 0.1 µF แบบเซรามิก ส่วน มีไว้ป้องกันสัญญาณรบกวน ซึ่งมักใช้ค่า 1 µF แบบแทนทาลัม หรือ 0.1 µF แบบเซรามิก

การป้อนแรงดันที่ขา in และขา out ของ MC78xx จะต้องมีค่าต่างกันพอสมควรโดยดูจากค่า $V_{in} - V_{out}$ ใน Data Sheet เช่น MC7805 ค่า $V_{in} - V_{out} = 2$ V ฉะนั้นแรงดันที่ขา in ต้องป้อนมากกว่า 7V ขึ้นไปแต่ต้องน้อยกว่า $V_{in(max)}$ ซึ่งจาก Data Sheet มีค่าเท่ากับ 35 V

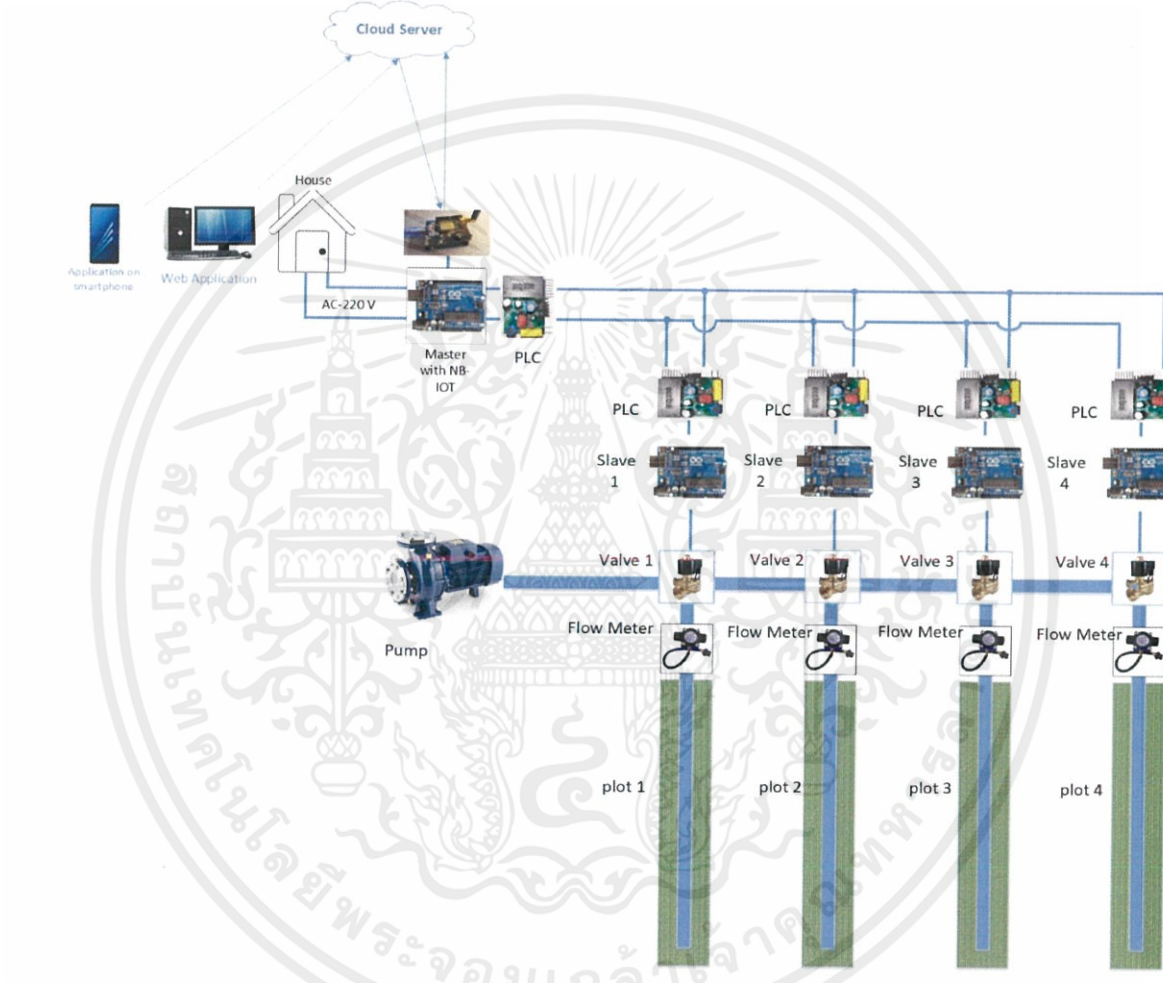


รูปที่ 2.29 วงจรแหล่งจ่ายไฟโดยใช้ MC78xx

บทที่ 3

การออกแบบและการจัดทำปริญญานิพนธ์

3.1 การออกแบบ



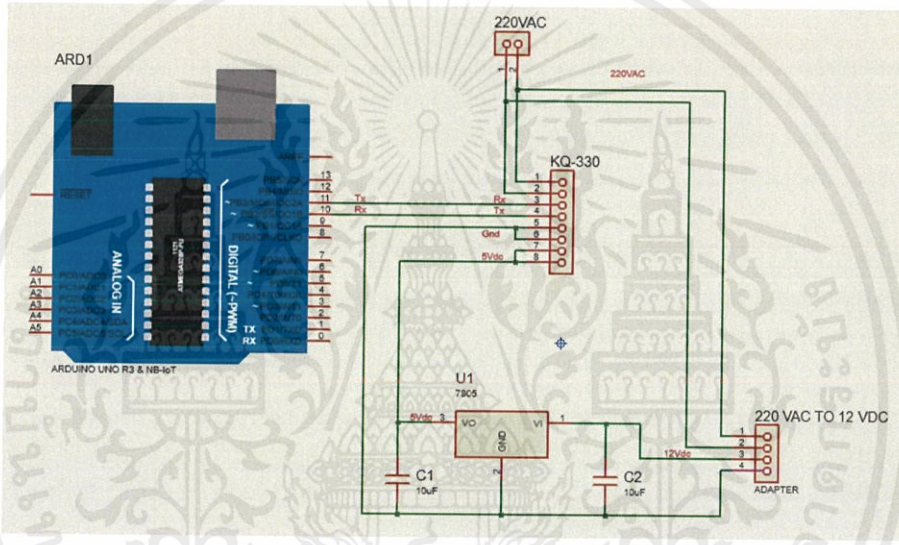
รูปที่ 3.1 ระบบการให้น้ำผ่านโครงข่ายการสื่อสาร NB-IOT

จากรูปที่ 3.1 บล็อกไดอะแกรมการทำงานของระบบการให้น้ำผ่านโครงข่าย NB-IOT โดยผู้ใช้สามารถควบคุมปริมาณการให้น้ำแก่พืชแต่ละชนิดผ่านคลาวด์ด้วยคำสั่งจากแอปพลิเคชันบนสมาร์ทโฟนและเว็บแอปพลิเคชัน โดยจะใช้อุปกรณ์สื่อสาร NB-IOT ร่วมกับบอร์ด Arduino ในการเชื่อมต่อกับเซิร์ฟเวอร์ อีกทั้งยังใช้การสื่อสารผ่านไฟบ้าน 220 โวลต์ด้วยเทคโนโลยีการสื่อสาร

แบบ Power Line Communication ในการสั่งการจากบอร์ดมาสเตอร์สู่บอร์ดสลาฟ โดยจะควบคุมให้ทำการจ่ายน้ำตามปริมาณน้ำที่กำหนด ซึ่งจะมี Flow Meter ในการตรวจดูการไหลของน้ำว่าวาล์วน้ำที่ได้สั่งการไปนั้นได้ทำงานตามคำสั่งจริงหรือไม่ (กรณีเกิดความเสียหายต่ออุปกรณ์) และแจ้งเตือนกลับมาสู่ผู้ใช้งาน

3.1.1 การเชื่อมต่อวงจร

1) การเชื่อมต่อวงจรของตัวมาสเตอร์ในโปรแกรมโพรติอุสเป็นดังรูปที่ 3.2



รูปที่ 3.2 การเชื่อมต่อวงจรของตัวมาสเตอร์

- ขา Vin ของ Arduino Uno ต่อกับแรงดัน 12V จากอแดปเตอร์เพื่อเป็นไฟเลี้ยงให้กับบอร์ด

- ขา GND ของ Arduino Uno ต่อลง GND ในวงจรเรกกูเลเตอร์ และขา Mode ของโมดูล KQ330

- ขา +5V ของ Arduino Uno ต่อกับ +5V ของโมดูล KQ330

- ขาที่ 7 ของ Arduino Uno ต่อกับไฟ LED เพื่อเอาไว้เป็นส่วนตรวจเช็คการส่งข้อมูล

- ขาที่ 10 ของ Arduino Uno ต่อกับขา Tx ของโมดูล KQ330 เพื่อคอยรับคำสั่งจากโมดูล

- ขาที่ 11 ของ Arduino Uno ต่อกับขา Rx ของโมดูล KQ330 เพื่อคอยส่งคำสั่งจากบอร์ดไปที่โมดูล

- ขาที่ 14 (GND) ของ Arduino Uno ต่อกับขา GND ของโมดูล KQ330

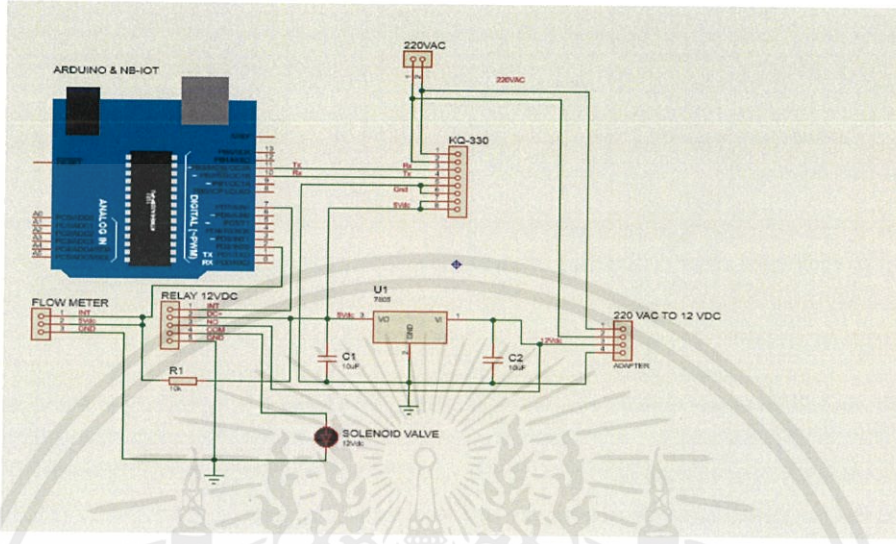
- ขา AC ของโมดูล KQ330 ต่อกับไฟบ้าน 220V

- ขา GND จากวงจรเรกกูเลเตอร์ต่อเข้ากับขา GND ของโมดูล KQ330

ซึ่งการเชื่อมต่ออุปกรณ์ตามการออกแบบข้างต้นเมื่อนำมาประกอบจะได้อุปกรณ์มาสเตอร์ดังรูปที่ 3.3

รูปที่ 3.3 อุปกรณ์มาสเตอร์ที่ได้ทำการเชื่อมต่อแล้ว

2) การเชื่อต่อวงจรของตัวเสลฟในโปรแกรมโพรติอุสเป็นดังรูปที่ 3.4



รูปที่ 3.4 การเชื่อมต่อวงจรของตัวสเลฟ

- ขา Vin ของ Arduino Uno ต่อกับแรงดัน 12V จากอแดปเตอร์เพื่อเป็นไฟเลี้ยงให้กับบอร์ด

- ขา GND ของ Arduino Uno ต่อเข้า GND ของ Flow Meter และขา Mode ของโมดูล KQ330

- ขา +5V ของ Arduino Uno ต่อกับ +5V ของโมดูล KQ330

- ขาที่ 2 ต่อกับตัวต้านทาน

- ขาที่ 7 ของ Arduino Uno ต่อเข้ากับขา IN ของอุปกรณ์ Relay

- ขาที่ 10 ของ Arduino Uno ต่อกับขา Tx ของโมดูล KQ330 เพื่อคอยรับคำสั่งจากโมดูล

- ขาที่ 11 ของ Arduino Uno ต่อกับขา Rx ของโมดูล KQ330 เพื่อคอยส่งคำสั่งจากบอร์ดไปที่โมดูล

- ขาที่ 14 (GND) ของ Arduino Uno ต่อกับขา GND ของโมดูล KQ330

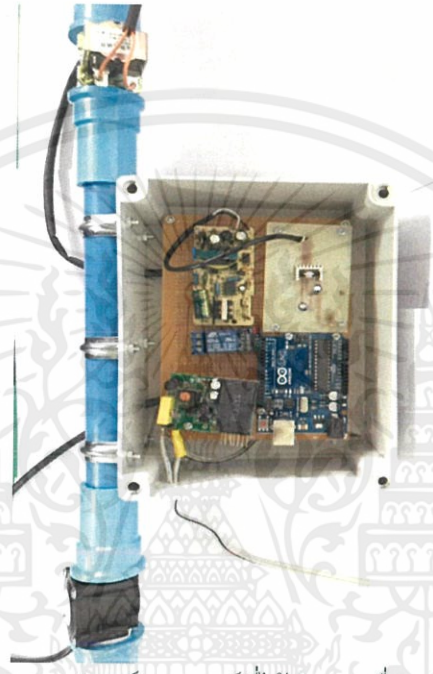
- ขา VCC ของอุปกรณ์ Relay ต่อเข้ากับแรงดัน +12V ของวงจรเรกกูเลเตอร์

- ขา GND ของอุปกรณ์ Relay ต่อเข้ากับแรงดัน GND ของวงจรเรกกูเลเตอร์

- ขา NO ของอุปกรณ์ Relay ต่อเข้ากับแรงดัน +12V ของวงจรเรกกูเลเตอร์

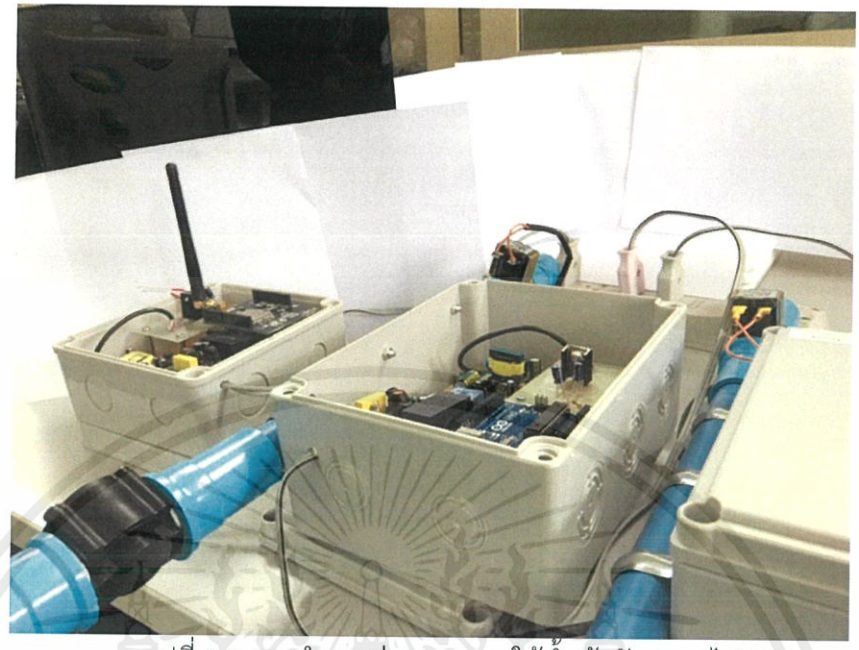
- ขา COM ของอุปกรณ์ Relay ต่อเข้ากับ Solenoid Valve
- ขาอีกด้านของ Solenoid Valve ต่อเข้ากับขา GND ของวงจรเรกกูเลเตอร์

ซึ่งการเชื่อมต่ออุปกรณ์ตามการออกแบบข้างต้นเมื่อนำมาประกอบจะได้อุปกรณ์เสลฟ ดังรูปที่ 3.5



รูปที่ 3.5 อุปกรณ์มาสเตอร์ที่ได้ทำการเชื่อมต่อแล้ว

เมื่อทำการเชื่อมต่ออุปกรณ์มาสเตอร์เข้ากับอุปกรณ์เสลฟจะได้โดยการเชื่อมต่อระบบด้วยสายไฟ 220v โดยในที่นี้จะทำการจำลองเชื่อมต่ออุปกรณ์มาสเตอร์ 1 ตัวและอุปกรณ์เสลฟ 1 ตัวเข้ากับเต้าเสียบเข้าเป็นระบบเดียวกัน จะได้เป็นดังภาพที่ 3.6

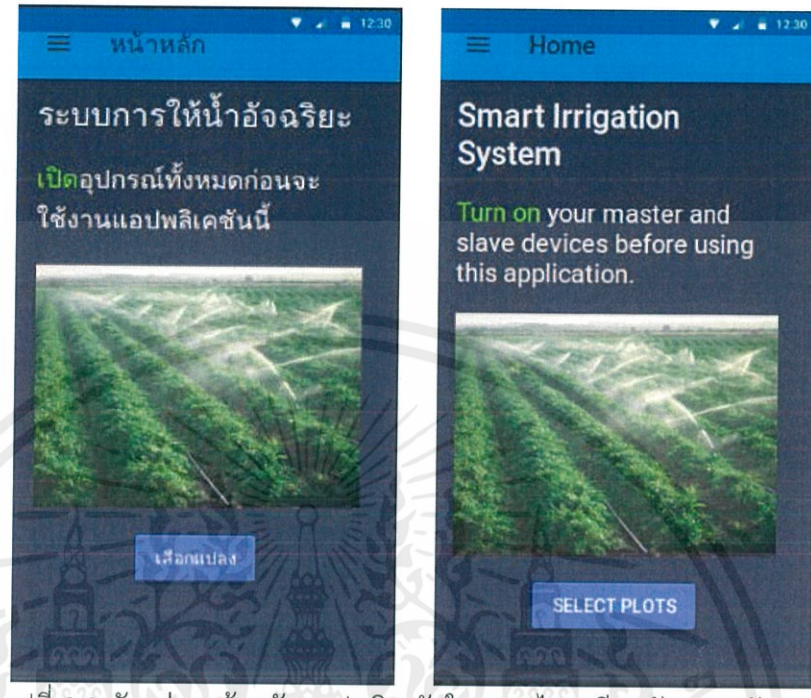


รูปที่ 3.6 การจำลองต่อระบบการให้น้ำเข้ากับระบบไฟ 220v

3.1.2 การออกแบบแอปพลิเคชันบนสมาร์ทโฟน

ในการออกแบบแอปพลิเคชันบนสมาร์ทโฟนจะมีรูปไอคอนของแอปพเคชันดังรูปที่ 3.7 เมื่อทำการกดเริ่มใช้งานจะเข้าสู่หน้าหลักแอปพลิเคชันที่ผู้ใช้สามารถทำการเลือกอุปกรณ์ในการสั่งการโดยในส่วนของหน้าหลักนั้นจะมีแถบเมนูให้ผู้ใช้สามารถเลือกเปลี่ยนภาษาได้สองภาษาคือภาษาอังกฤษและภาษาไทยเพื่อความสะดวกตามลักษณะของผู้ใช้งานโดยภาษาเริ่มต้นจะเป็นภาษาไทยมีหน้าหลักเป็นดังรูปที่ 3.8 และแถบเมนูแสดงดังรูปที่ 3.9

รูปที่ 3.7 ไอคอนแอปลิเคชัน



รูปที่ 3.8 ตัวอย่างหน้าหลักแอปพลิเคชันในภาษาไทยเทียบกับภาษาอังกฤษ

รูปที่ 3.9 แถบเมนูการใช้งาน

ในส่วนของการเลือกแปลงที่ต้องการใช้งานนั้นจะมีหน้าเริ่มต้นเป็นดังรูปที่ 3.10 ซึ่งจะแสดงถึงจำนวนของอุปกรณ์ที่ได้ทำการเชื่อมต่อไว้พร้อมสำหรับการใช้งานโดยกรณีนี้มีอุปกรณ์ที่พร้อมใช้งานสองอุปกรณ์ที่ติดตั้งไว้ตามแปลงเกษตรสองแปลงคือแปลง 1 และแปลง 2

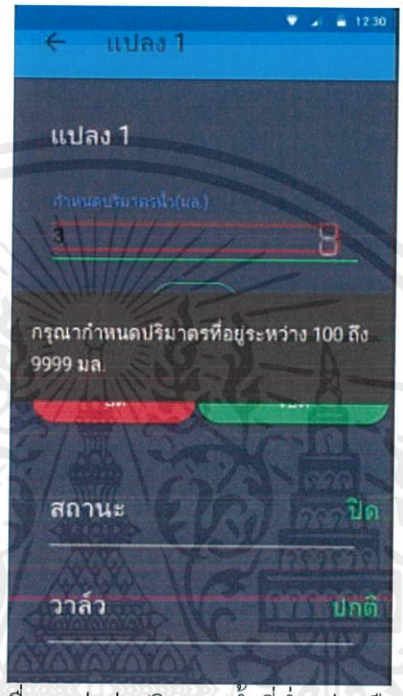
รูปที่ 3.10 หน้าใช้งานเลือกอุปกรณ์

เมื่อทำการเลือกอุปกรณ์ที่ต้องการใช้งาน หน้าแอปพลิเคชันจะมีลักษณะเป็นดังรูปที่ 3.11 ซึ่งถ้าผู้ใช้ไม่ต้องการกำหนดปริมาณน้ำผู้ใช้สามารถเปิดการทำงานของวาล์วน้ำได้ทันที และในกรณีที่ผู้ใช้ต้องการกำหนดปริมาณน้ำจะมีส่วนให้ผู้ใช้กำหนดปริมาณน้ำในหน่วยมิลลิลิตร โดยต้องทำการกดส่งค่าให้อุปกรณ์ที่ปุ่ม send หลังจากที่กดส่งแล้วจะมีการแจ้งเตือนแก่ผู้ใช้ว่าได้ทำการกำหนดปริมาณไปเท่าใด ดังรูปที่ 3.12

รูปที่ 3.11 หน้าควบคุมอุปกรณ์แปลงที่ 1

รูปที่ 3.12 แจ้งเตือนเมื่อส่งค่าปริมาตรน้ำที่กำหนด

ถ้าหากผู้ใช้ทำการกำหนดปริมาณน้ำที่น้อยเกินไปจะมีการแจ้งเตือนบอกถึงปริมาณขั้นต่ำที่ผู้ใช้ควรกำหนดคือปริมาณระหว่าง 100-9999 มิลลิลิตรเพื่อคำนึงถึงความเหมาะสมในการให้น้ำแก่แปลงเกษตร ดังรูปที่ 3.13

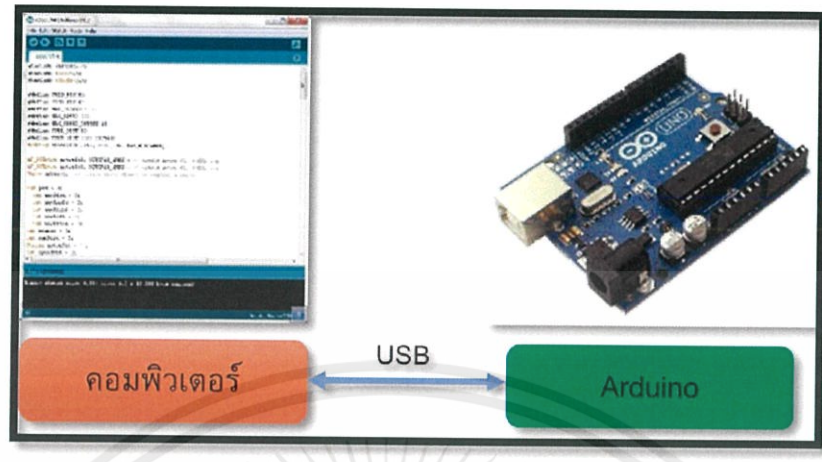


รูปที่ 3.13 แจ้งเตือนเมื่อกดส่งค่าปริมาตรน้ำที่ต่ำกว่าหรือมากกว่าค่าที่กำหนด

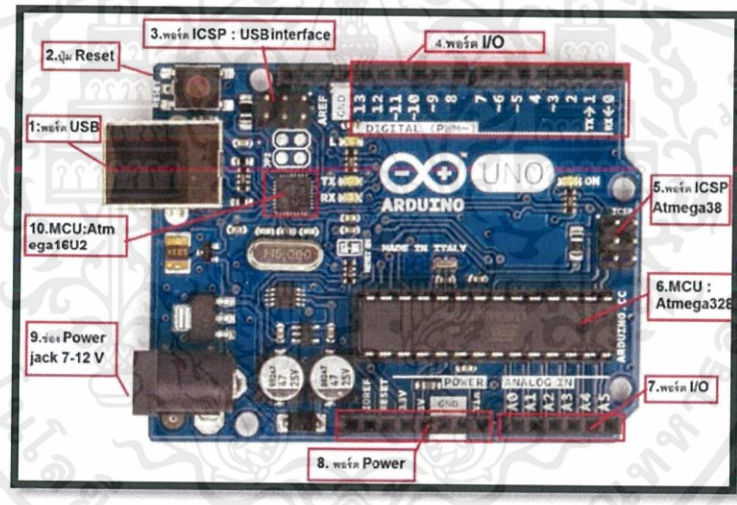
3.2 เครื่องมือที่ใช้ในการทดลอง

3.2.1 บอร์ด Arduino

Arduino เป็นบอร์ดไมโครคอนโทรเลอร์ตระกูล AVR ที่มีการพัฒนาแบบ Open Source คือมีการเปิดเผยข้อมูลทั้งด้าน Hardware และ Software ตัวบอร์ด Arduino ถูกออกแบบมาให้ใช้งานได้ง่าย สามารถเชื่อมต่อกับคอมพิวเตอร์เพื่ออัพโหลดโปรแกรมและจ่ายไฟให้กับบอร์ดได้ด้วยสาย USB ดังรูปที่ 3.14 ดังนั้นจึงเหมาะสำหรับผู้เริ่มต้นศึกษา โดยผู้ใช้งานยังสามารถดัดแปลง พัฒนาต่อยอดได้ และบอร์ด Arduino ยังสามารถต่อกับอุปกรณ์และวงจรอิเล็กทรอนิกส์จากภายนอกแล้วเชื่อมต่อเข้ามาที่ขา I/O ของบอร์ด แสดงดังรูปที่ 3.15



รูปที่ 3.14 การเชื่อมต่อ Arduino กับ Computer



รูปที่ 3.15 Layout & Pin out Arduino Board (Model: Arduino UNO R3)

1. USBPort : ใช้สำหรับต่อกับ Computer เพื่ออัพโหลดโปรแกรมเข้า MCU และจ่ายไฟให้กับบอร์ด

2. Reset Button : เป็นปุ่ม Reset ใช้กดเมื่อต้องการให้ MCU เริ่มการทำงานใหม่

3. ICSP Port : ของ Atmega16U2 เป็นพอร์ตที่ใช้โปรแกรม Visual Com port บน Atmega16U2

4. I/Oport : Digital I/O ตั้งแต่ขา D0 ถึง D13 นอกจากนี้ บาง Pin จะทำหน้าที่อื่นๆ เพิ่มเติมด้วย เช่น Pin0,1 เป็นขา Tx,Rx Serial, Pin3,5,6,9,10 และ 11 เป็นขา PWM

5. ICSP Port : Atmega328 เป็นพอร์ตที่ใช้โปรแกรม Bootloader

6. MCU Atmega328 เป็น MCU ที่ใช้บนบอร์ด Arduino

7. I/O Port : นอกจากจะเป็น Digital I/O แล้ว ยังเปลี่ยนเป็น ช่องรับสัญญาณอนาล็อก ตั้งแต่ขา A0-A5

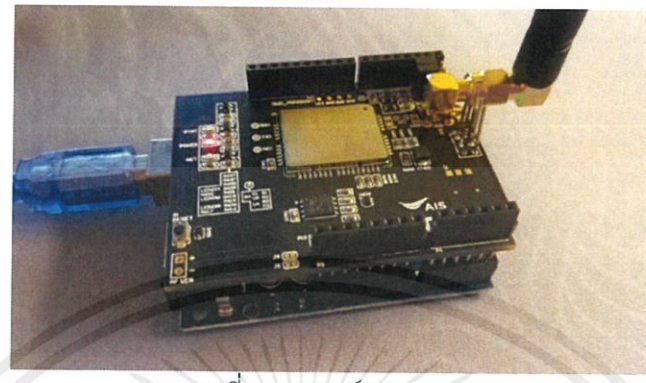
8. Power Port : ไฟเลี้ยงของบอร์ดเมื่อต้องการจ่ายไฟให้กับวงจรภายนอก ประกอบด้วยขาไฟเลี้ยง +3.3 V, +5V, GND, VIN

9. Power Jack : รับไฟจาก Adapter โดยที่แรงดันอยู่ระหว่าง 7-12 V

10. MCU ของ Atmega16U2 เป็น MCU ที่ทำหน้าที่เป็น USB to Serial โดย Atmega328 จะติดต่อกับ Computer ผ่าน Atmega16U

โปรแกรมภาษาของ Arduino จะใช้ภาษา C++ ซึ่งเป็นรูปแบบของโปรแกรมภาษาซีประยุกต์แบบหนึ่ง ที่มีโครงสร้างตัวภาษาโดยรวมใกล้เคียงกันกับ ภาษาซีมาตรฐาน (ANSI-C) อื่นๆ เพียงแต่ได้มีการปรับปรุงรูปแบบในการเขียนโปรแกรมบางส่วนที่ผิดเพี้ยนไปจากANSI-C เล็กน้อย เพื่อช่วยลดความยุ่งยากในการเขียนโปรแกรมและให้ผู้เขียนโปรแกรมสามารถได้ง่ายและ สะดวกมากขึ้นกว่าการเขียนภาษาซีตามแบบมาตรฐานของ ANSI-C โดยตรงภาษาซี สามารถ นำไปใช้งานบนระบบฮาร์ดแวร์ที่มีความแตกต่างกันได้หลากหลาย โดยผู้ใช้เพียงแต่เลือกใช้ ตัวแปลคำสั่ง (C-Compiler) ให้ตรงกับระบบฮาร์ดแวร์ที่ใช้งานอยู่ ส่วนเรื่องรูปแบบในการเขียน โปรแกรมจะเป็นมาตรฐานเดียวกัน American National Standard Institute (ANSI) จงได้ทำการ ตั้งข้อกำหนดมาตรฐานของภาษาซีขึ้น โดยเรียกกันว่า “ANSI-C” เพื่อใช้เป็นข้อบังคับและคง มาตรฐานของภาษาซีไว้ไม่ให้เปลี่ยนแปลงไปจากเดิม โดยทุกผู้ผลิตจะต้องสร้างตัวแปลภาษาให้มี คุณสมบัติการใช้งานขั้นพื้นฐานเดียวกัน

3.2.2 NB-IOT Module



รูปที่ 3.16 บอร์ด NB-IOT

จากรูปที่ 3.16 คืออุปกรณ์ NB-IOT ที่ใช้โดยรายละเอียดดังนี้

1. USBPort : ใช้สำหรับต่อกับ Computer เพื่ออัพโหลดโปรแกรมเข้า MCU และจ่ายไฟให้กับบอร์ด

2. Reset Button : เป็นปุ่ม Reset ใช้กดเมื่อต้องการให้ MCU เริ่มการทำงานใหม่

3. ICSP Port : ของ Atmega16U2 เป็นพอร์ตที่ใช้โปรแกรม Visual Com port บน Atmega16U2

4. I/Oport : Digital I/O ตั้งแต่ขา D0 ถึง D13 นอกจากนี้ บาง Pin จะทำหน้าที่อื่นๆ เพิ่มเติมด้วย เช่น Pin0,1 เป็นขา Tx,Rx Serial, Pin3,5,6,9,10 และ 11 เป็นขา PWM

5. ICSP Port : Atmega328 เป็นพอร์ตที่ใช้โปรแกรม Bootloader

6. MCU Atmega328 เป็น MCU ที่ใช้บนบอร์ด Arduino

7. I/O Port : นอกจากจะเป็น Digital I/O แล้ว ยังเปลี่ยนเป็น ช่องรับสัญญาณอนาล็อก ตั้งแต่ขา A0-A5

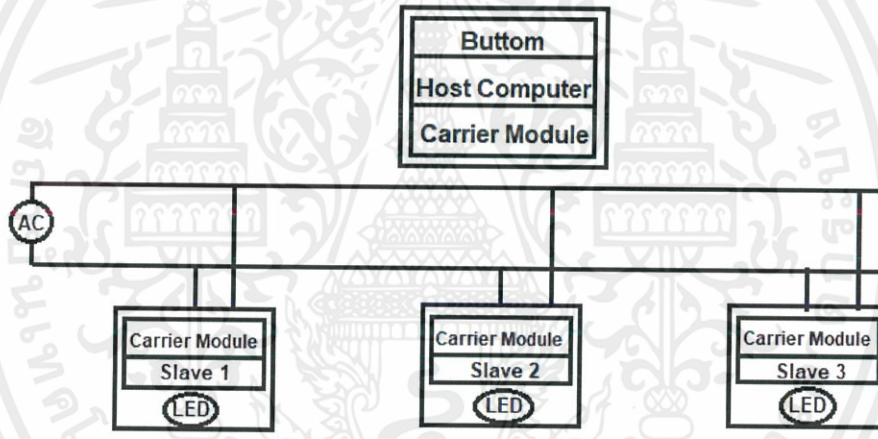
8. Power Port : ไฟเลี้ยงของบอร์ดเมื่อต้องการจ่ายไฟให้กับวงจรภายนอก ประกอบด้วยขาไฟเลี้ยง +3.3 V, +5V, GND, VIN

9. Power Jack : รับไฟจาก Adapter โดยที่แรงดันอยู่ระหว่าง 7-12 V

10. MCU ของ Atmega16U2 เป็น MCU ที่ทำหน้าที่เป็น USB to Serial โดย Atmega328 จะติดต่อกับ Computer ผ่าน Atmega16U2

3.2.3 KQ330 (Power Line Communication)

ระบบประกอบด้วยไมโครคอนโทรลเลอร์ STC (ตัวประมวลผลหลัก), โมดูล KQ330 (โมดูลโมเด็ม) วงจรการส่งข้อมูล วงจรรับข้อมูล วงจรตรวจจับ zero-crossing วงจรขยายไตรโอด วงจรเรโซแนนซ์และวงจรแยกหม้อแปลง ระบบ Master-Slave สามารถส่งสัญญาณข้อมูลผ่านสายไฟเพื่อให้สามารถควบคุมได้จากระยะไกลของเครื่องโฮสต์ได้ หลักของระบบประกอบด้วยสองส่วนคือหนึ่งในนั้นคือไมโครคอนโทรลเลอร์ STC ควบคุมโมดูล KQ330 ของผู้ให้บริการสายส่งเพื่อส่งและรับข้อมูล (วงจรการตรวจจับเรโซแนนซ์และวงจรขยายสัญญาณ) รูปที่ 3.17 แสดงรูปไดอะแกรมภาพรวมของระบบ



รูปที่ 3.17 ไดอะแกรมภาพรวมของระบบ

การออกแบบระบบสื่อสารสายส่งไฟฟ้าสามารถแบ่งออกเป็นการรับส่งข้อมูลและรับข้อมูลตามการไหลของข้อมูลซึ่งการรับส่งข้อมูลมีดังนี้

1. ระบบใช้อินเทอร์เฟซแบบตรง (การสื่อสารแบบอะซิงโครนัสแบบอนุกรม) ของโมดูล SCM และ KQ330

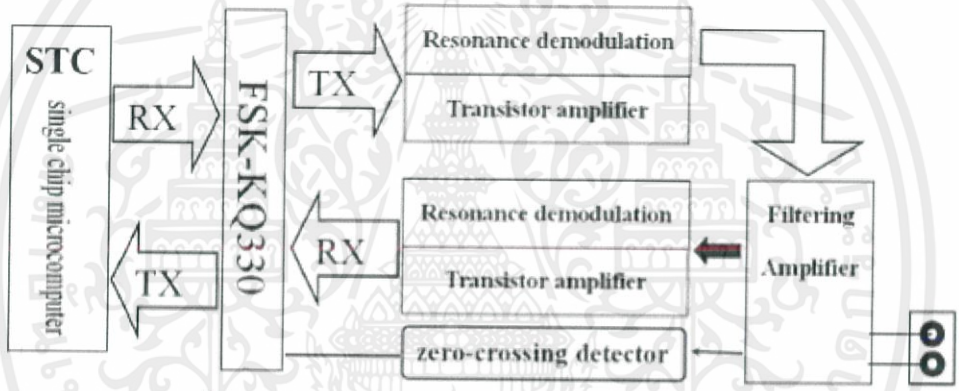
2. หลังจากที่ข้อมูลการรับส่งข้อมูลถูกมอดูเลตโดยโมดูล KQ330 ผ่านวงจรภายนอก (วงจรเครื่องขยายและวงจรการตรวจจับเรโซแนนซ์) สัญญาณสแควร์เวฟจะเปลี่ยนเป็นสัญญาณไซน์ (sinusoidal signal) หลังจากแยกสัญญาณรบกวนสัญญาณจะถูกส่งไปกับสายไฟ

การไหลของข้อมูลที่ได้รับมีดังนี้

1. วงจรตรวจจับเรโซแนนซ์จะตรวจจับสัญญาณที่ได้รับ

2. หลังจากที่สัญญาณถูกแยกจากหม้อแปลงรูปคลื่นจะมีรูปร่างตามวงจรเรโซแนนซ์ ในที่สุดสัญญาณรูปร่างจะถูกดีมอดูเลตโดยโมดูลข้อมูลของ KQ330 หลังจากที่ข้อมูลถูกดีมอดูเลต ข้อมูลจะถูกส่งกลับไปยังไมโครคอนโทรลเลอร์ผ่านทางพอร์ตอนุกรม

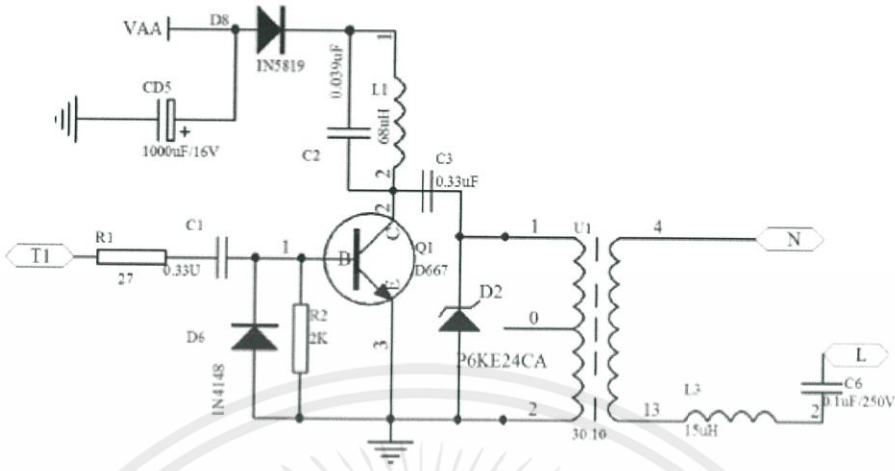
ในส่วนข้อมูลที่ได้รับข้อมูลจะมีวงจรตรวจจับ zero-crossing โดยฟังก์ชั่นของวงจรคือ การตรวจจับความถี่ของสัญญาณคลื่นซายน์บนสายไฟ เมื่อระดับของสัญญาณผ่านที่จุดศูนย์ ไมโครคอนโทรลเลอร์จึงจะทำการส่งหรือรับข้อมูล จากรูปที่ 3.18 แสดงแผนภาพการไหลของสัญญาณของโมดูลสายส่ง



รูปที่ 3.18 แผนภาพการไหลของสัญญาณของโมดูลสายส่ง

3.2.3.1.วงจรการส่งข้อมูล

วงจรส่วนใหญ่ประกอบด้วยวงจรขยายของไตรโอดส์ วงจรเรโซแนนและวงจรแยกตัวหม้อแปลง วงจรเรโซแนนซ์ช่วยให้สัญญาณคลื่นมีเสถียรภาพมากขึ้นและไม่มีสัญญาณรบกวน มีแผนภาพหลักการทำงานวงจรด้านส่งดังรูปที่ 3.19

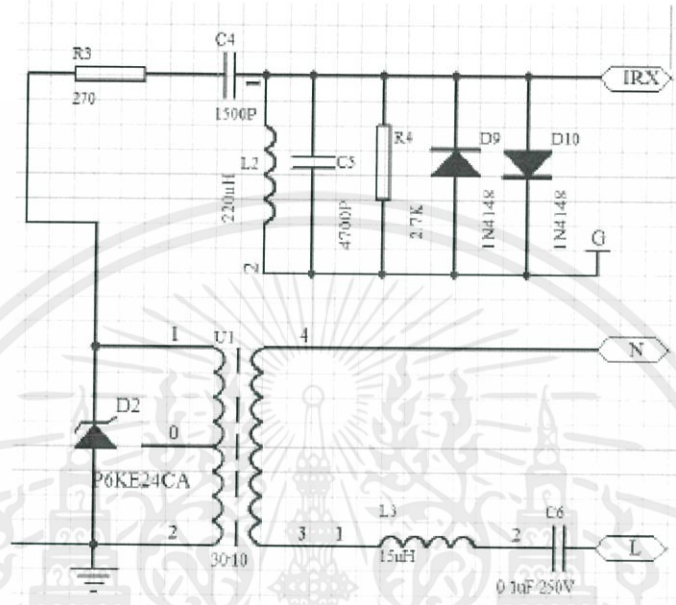


รูปที่ 3.19 วงจรด้านส่ง

การวิเคราะห์วงจรในระบบการสื่อสารพาห์ของ Power Line การทำงานของวงจรด้านส่งนั้นจะมี ขา 8 ของโมดูล KQ330 ส่งทำการส่งสัญญาณสแควร์เวฟจากนั้นจะส่งต่อไปกับสายไฟหลังจากที่ผ่านวงจรขยายและตรวจจับแล้ว โดยหน้าที่ของ R1 และ C1 คือการจำกัดกระแสที่ไหลผ่าน ทรานซิสเตอร์ Q1 เป็นตัวขยายสัญญาณ L1 และ C2 เป็นวงจรเรโซแนนซ์ ซึ่งบทบาทของวงจรนี้สามารถเปลี่ยนรูปสัญญาณสแควร์เวฟขาที่ขาที่ 8 ของ KQ330 ให้แป็นคลื่นไซน์ จากนั้นจะขยายสัญญาณจากทรานซิสเตอร์ Q1 ซึ่งฟังก์ชั่นของหม้อแปลงคือการแยกสัญญาณรบกวนออก

3.2.3.2. วงจรการรับข้อมูล

วงจรนี้ประกอบด้วยวงจรขยายไตรโอดส์ วงจรตรวจจับเรโซแนนซ์ วงจรแยกหม้อแปลงและวงจรตรวจจับ zero-crossing วงจรรับสัญญาณเชื่อมต่อกับสายไฟและส่งสัญญาณข้อมูลไปยังโมดูล หน้าที่หลักของวงจรรับคือการตรวจจับ เมื่อตรวจพบสัญญาณคลื่นและขยายสัญญาณโมดูล KQ330 สามารถระบุสภาวะปกติได้ แผนภาพหลักการทำงานของวงจรด้านรับเป็นดังรูปที่ 3.20

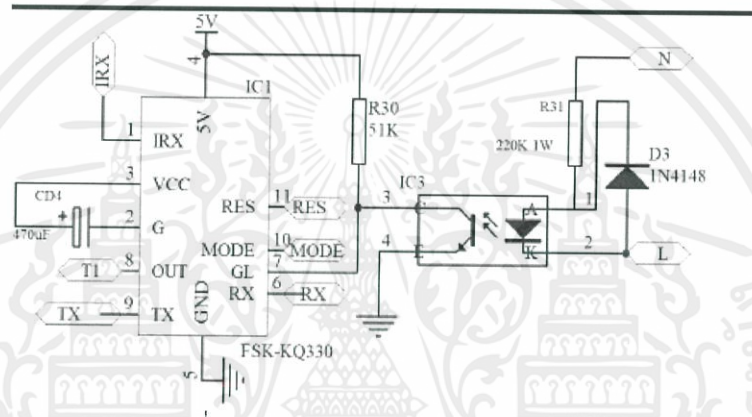


รูปที่ 3.20 วงจรด้านรับ

การวิเคราะห์วงจรขั้นตอนการทำงานของวงจรรับสัญญาณมีดังต่อไปนี้ ประการแรก สัญญาณในสายไฟเข้าขาด้านรับ (ขา1) ของ KQ330 หลังจากข้ามวงจรอุปกรณ์ต่อพ่วง ฟังก์ชั่นของวงจรรับสัญญาณที่อยู่รอบข้างคือวงจรเรโซแนนซ์เลือกความถี่ กล่าวคือใช้ KQ330 เพื่อระบุความถี่ของสัญญาณหลังจากที่หม้อแปลงแยกสัญญาณขัดจังหวะและเรโซแนนซ์เลือกความถี่สัญญาณและขยายสัญญาณจะใช้ไดโอดสองตัวกรองสัญญาณที่มีขนาดใหญ่เกินไปเพื่อป้องกันโมดูล KQ330 ไม่ให้เสียหาย ถัดมา L3 และ C6 ที่เป็นวงจรเรโซแนนซ์จะสามารถให้สัญญาณที่ต้องการเข้ามา โดยบทบาทของวงจรเรโซแนนซ์คือการขยายสัญญาณและการตรวจจับสัญญาณ เหมือนกับฟังก์ชั่นของวงจรเรโซเนเตอร์ของวิทยุที่สามารถแยกรูปคลื่นของสัญญาณได้ โดย L2 และ C5 เป็นวงจรก็เป็นวงจรเรโซแนนซ์เช่นเดียวกัน โดยมีหน้าที่คือการขยายสัญญาณและการตรวจจับสัญญาณเพื่อกรองสัญญาณสัญญาณผิดเพี้ยน ไดโอดย้อนกลับสองอัน (1N4148) มีหน้าที่คือการจำกัดแอมพลิจูดของคลื่นโดยให้มีค่าน้อยกว่า 0.7 โวลต์ ดังนั้นวงจรสามารถระบุรูปคลื่นที่ต้องการและป้องกันโมดูล KQ330 ได้อย่างมีประสิทธิภาพ

3.2.3.3 วงจรตรวจจับ zero-crossing

บทบาทของวงจรตรวจจับสัญญาณ zero-crossing คือการตรวจจับคลื่นซายน์ในสายไฟ เมื่อคลื่นไซน์ผ่านจุดศูนย์สัญญาณข้อมูลจะสามารถส่งได้อย่างเสถียรและความกว้างของคลื่นจะไม่เปลี่ยนแปลงไปมาก วงจรตรวจจับ zero-crossing แสดงดังรูปที่ 3.21



รูปที่ 3.21 วงจรตรวจจับ zero-crossing

3.2.3.4 โมดูล KQ330

โมดูลใช้สายไฟแรงดันต่ำเป็นสื่อของการส่งสัญญาณ มีความถี่คลื่นพาห์ของสัญญาณที่มอดูเลตโดยโมดูลคือ 50 kHz ~ 350 kHz สัญญาณความถี่สูงในสายไฟแรงดันต่ำสามารถส่งผ่านไปตามเส้นทางได้ แผนภาพขาของโมดูลแสดงในรูปที่ โดย pin1 เป็นจุดเริ่มต้นของข้อมูลขาเข้า ขา 8 คือส่วนของข้อมูลขาออก แรงดันไฟฟ้าในการทำงานของโมดูลคือ +5 โวลต์

3.2.4 Power Adapter 12V 1A

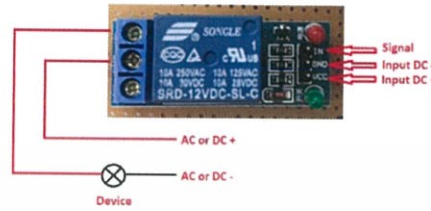
Power Adapter 12V 1A ดังรูปที่ 3.22 คือ อะแดปเตอร์ แหล่งจ่ายไฟ 12V 1 แอมป์ สามารถใช้กับบอร์ด Arduino ได้ โดยข้อมูลของอุปกรณ์ดังนี้

รูปที่ 3.22 อะแดปเตอร์ 12V 1A

1. แรงดันไฟฟ้าขาเข้า : AC100V-240V ~ 50 / 60Hz
2. แรงดันขาออก: DC 12v 1A
3. ขั้วเอาต์พุต: ช่องเสียบด้านในอยู่ด้านนอกบวกลบ
4. การเชื่อมต่อเอาท์พุท: เส้นผ่าศูนย์กลาง 5.5 มม. เจาะ 2.5 มม. (ปลั๊กใส่ 2.1mm innerspring ใส่สากล)

3.2.5 12V 1 Channel Relay High-Level Trigger Relay Module (with Optocoupler)

ใช้งานในการควบคุมอุปกรณ์ไฟฟ้า รับกระแสได้ถึง 10 A ใช้งานได้ทั้งไฟฟ้ากระแสตรงและกระแสสลับ รับแรงดันระดับ 12V ตรงจาก Arduino board ออกแบบให้ป้องกันวงจรด้านควบคุมออกจากด้านกำลังโดยการใช้การส่งผ่านด้วยแสง (Optocoupler) ในตัวรีเลย์ซึ่งประกอบด้วยขา 3 ขาคือ Normally Close(NC), Normally Open(NO) และ Common(COM) ดังรูปที่ 3.23



รูปที่ 3.23 อุปกรณ์ Relay ที่ใช้งาน

3.2.5.1 คุณสมบัติของโมดูลรีเลย์ 1 ช่อง

1. รับกระแสได้ 10A และ 250VAC หรือ 30VDC
2. แต่ละช่องมี LED แสดงสถานะ
3. แรงดันคอยล์ 12V ต่อช่อง
4. ชุดแรงดันไฟฟ้าทำงาน 5-12 V
5. สัญญาณอินพุต 3-5 V สำหรับแต่ละช่องสัญญาณ
6. ขา IN สำหรับเชื่อมต่อกับบอร์ด Arduino
7. มีสามพินสำหรับ Normally Open และ Normally Close สำหรับแต่ละช่อง
8. VCC(DC+) เชื่อมไฟเลี้ยง 12v

3.2.6 โซลินอยด์วาล์วพลาสติก 6หุน 12VDC

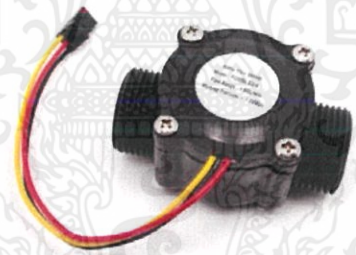
รูปที่ 3.24 โซลินอยด์วาล์วที่ใช้

โซลินอยด์วาล์ว พลาสติก ชนิด N/C (Normally Closed) แบบปกติปิด ดังรูปที่ 3.25 เมื่อทำการจ่ายไฟฟ้าให้โซลินอยด์วาล์ว หรือวาล์วไฟฟ้าจะทำการเปิดออก น้ำจะสามารถไหลผ่านไปได้ ซึ่งจะมีลูกศรบอกที่โซลินอยด์วาล์ว ถ้าลูกศรชี้ไปทางไหนคือเป็นทางออก โดยลักษณะอุปกรณ์เป็นดังนี้

1. เป็นวาล์วเปิด-ปิดอัตโนมัติ ที่สั่งการด้วยไฟฟ้า
2. ฝาเกลียว ขนาด 6 หุน
3. วาล์ว เปิด เมื่อมีการจ่ายไฟฟ้าเข้า
4. ไฟเลี้ยง 12 VDC 0.3A
5. ตัวเครื่องทำด้วยวัสดุพลาสติกทางน้ำเข้า-ออกมีลักษณะเป็นเกลียว
6. ทนแรงดัน 0.01Mpa

3.2.7 G3/4 Water Flow Sensor

เป็นเซนเซอร์วัดอัตราการไหลของน้ำที่มีแรงดันต่ำกว่า 1.75 MPa โดยมีอัตราการไหลสูงสุด รูปที่ 3.25 แสดงถึง Water Flow Sensor ที่ใช้งาน



รูปที่ 3.25 Flow Meter

G3/4 Water Flow Sensor เป็น Sensor สำหรับวัดอัตราการไหลของน้ำ (น้ำกรดด่าง และน้ำมันทุประเภท ไม่สามารถใช้ได้) สามารถต่อกับท่อ PVC ขนาด 3/4 นิ้วได้ โดยที่สามารถวัด Flow rate ได้ช่วง 1~60 L/min โดยมี Hall-Effect Sensor ส่งสัญญาณ Pulse Signal ออกมา โดยรายละเอียดอุปกรณ์เป็นดังนี้

1. แรงดันที่อุปกรณ์ทำงาน : 5V-24V
2. แรงดันค่าน้อยที่สุดที่อุปกรณ์ทำงาน : DC 4.5V
3. แรงดันค่ามากที่สุดที่อุปกรณ์ทำงาน : 15mA (DC 5V)

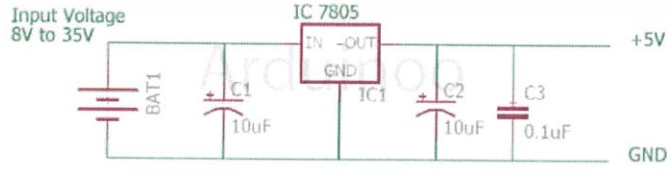
4. เส้นผ่าศูนย์กลางภายนอก : 26mm
5. ย่านอัตราการไหล : 1~60 L/min
6. อุณหภูมิที่ทำงาน : 0°C ~80°C
7. อุณหภูมิน้ำ : <120°C
8. ความชื้นที่สามารถทำงาน : 35%~90%RH
9. ความดันที่สามารถทำงาน : under 1.75Mpa
10. อุณหภูมิที่เก็บรักษา : -25°C~+80°C

3.2.8 วงจรแปลงไฟ 12 Vdc เป็น 5 Vdc

รูปที่ 3.26 IC7805 ที่ใช้งาน

การใช้งาน IC Voltage Regulator (7805) IC Voltage Regulator เป็น IC ที่แปลงจากแรงดันที่สูงกว่า (Vin) ให้เป็นแรงดันที่ต่ำกว่าและเรียบคงที่ (Vout) โดยในบทความนี้จะกล่าวถึง IC 78xx Series ซึ่งเป็น Fixed LinearVoltage Regulator คือไม่สามารถเปลี่ยนแรงดันเอาต์พุตได้ (มี Linear Voltage Regulator บางตัวที่สามารถเปลี่ยนค่า Vout ได้ เช่น LM317) โดยแต่ละรุ่นใน 78xx Series ก็จะมีค่า แรงดันเอาต์พุตที่ต่างกันไป โดยการดูจากเลข 2 หลักท้ายของชื่อ IC เช่น 7805 ดังรูปที่ 3.26 ก็จะมีค่า แรงดันเอาต์พุต 5V

วงจรแปลงไฟจาก 8-35V เป็น 5V ดังรูปที่ 3.27 สามารถจ่ายให้กับ Sensor ต่างๆที่ต้องการแรงดันไฟ 5V ได้ (C1,C2 เป็น Electrolytic Capacitor และ C3 เป็น Ceramic Capacitor)



รูปที่ 3.27 วงจรแปลงไฟจาก 8-35V เป็น 5V

3.3 การจัดเก็บผลการทดลอง

1) ตรวจสอบข้อมูลปริมาณน้ำที่ส่งมาบน Serial Monitor ว่าค่าที่รับมาตรงกับค่าที่ส่งมาจากมาสเตอร์หรือไม่

2) การสั่งการทำงานและตรวจสอบสถานะการทำงานของระบบน้ำผ่านเว็บแอปพลิเคชันและแอปพลิเคชันบนสมาร์ทโฟน

3) การทดลองหาค่าเฉลี่ยปริมาณน้ำที่ระบบได้กำหนดไว้

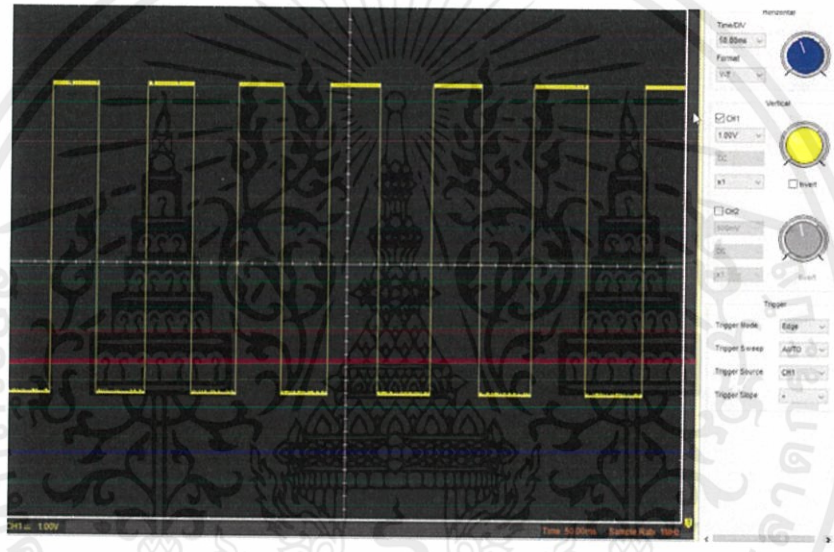
4) การแสดงผลทาง SSH ใน Server

บทที่4

ผลการทดลอง

4.1 วัดสัญญาณพัลส์ที่เกิดจากการใช้ Water flow meter วัดปริมาตรน้ำ

จากการทดสอบวัดสัญญาณที่เกิดจากการสั่งให้ระบบทำการจ่ายน้ำปริมาตร 300 มิลลิลิตร โดยสัญญาณที่วัดได้จาก Water Flow Sensor เป็นสัญญาณพัลส์ที่มีความถี่ตามอัตราการไหลของน้ำซึ่งเป็นดังรูปที่ 4.1



รูปที่ 4.1 สัญญาณพัลส์ที่วัดได้จาก Water Flow Sensor

4.2 ทดสอบการวัดปริมาตรน้ำ

4.2.1 การวัดปริมาตรน้ำโดยกำหนดให้ระบบจ่ายปริมาตรน้ำที่ 500 มิลลิลิตร

การทดลองสั่งงานระบบโดยสั่งงานผ่านแอพพลิเคชั่น โดยกำหนดปริมาตรน้ำ 500 มิลลิลิตรและทดลองวัดปริมาตรน้ำที่ระบบได้จ่ายออกมาจริงจำนวน 5 ครั้ง และหาปริมาตรน้ำเฉลี่ยของทั้ง 5 ครั้ง เป็นไปตามตารางที่ 4.1

ตารางที่ 4.1 การวัดปริมาตรน้ำที่วัดได้จากการทดลองเมื่อสั่งงานให้ระบบจ่ายน้ำจำนวน 500 มิลลิลิตร

การทดลองครั้งที่	ปริมาตรน้ำที่วัดได้(มิลลิลิตร)
1	500
2	530
3	510
4	500
5	500
เฉลี่ย	508

4.2.2 การวัดปริมาตรน้ำโดยกำหนดให้ระบบจ่ายปริมาตรน้ำที่ 1000 มิลลิลิตร

การทดลองสั่งงานของระบบโดยสั่งงานผ่านแอพพริเคชั่น โดยกำหนดปริมาตรน้ำจำนวน 1000 มิลลิลิตรและทดลองวัดปริมาตรน้ำที่ระบบได้จ่ายออกมาจริงจำนวน 5 ครั้ง และหาปริมาตรน้ำเฉลี่ยของทั้ง 5 ครั้ง เป็นไปตามตารางที่ 4.2

ตารางที่ 4.2 การวัดปริมาตรน้ำที่วัดได้จากการทดลองเมื่อสั่งงานให้ระบบจ่ายน้ำจำนวน 1000 มิลลิลิตร

การทดลองครั้งที่	ปริมาตรน้ำที่วัดได้(มิลลิลิตร)
1	1015
2	1025
3	1030
4	1005
5	1010
เฉลี่ย	1017

4.2.3 การวัดปริมาตรน้ำโดยกำหนดให้ระบบจ่ายปริมาตรน้ำที่ 5000 มิลลิลิตร

การทดลองสั่งงานของระบบโดยสั่งงานผ่านแอพพริเคชั่น โดยกำหนดปริมาตรน้ำจำนวน 5000 มิลลิลิตรและทดลองวัดปริมาตรน้ำที่ระบบได้จ่ายออกมาจริงจำนวน 5 ครั้ง และหาปริมาตรน้ำเฉลี่ยของทั้ง 5 ครั้ง เป็นไปตามตารางที่ 4.3

ตารางที่ 4.3 การวัดปริมาตรน้ำที่วัดได้จากการทดลองเมื่อสั่งงานให้ระบบจ่ายน้ำจำนวนคงที่ 5000 มิลลิลิตร

การทดลองครั้งที่	ปริมาตรน้ำที่วัดได้(มิลลิลิตร)
1	5020
2	5015
3	5000
4	5015
5	5030
เฉลี่ย	5016

4.2.4 ทดสอบการวัดปริมาตรน้ำโดยกำหนดให้ระบบจ่ายน้ำในปริมาณที่เพิ่มขึ้น

การทดลองสั่งงานของระบบโดยสั่งงานผ่านแอพพลิเคชั่น โดยกำหนดปริมาตรน้ำจำนวน 10000 มิลลิลิตร, 20000 มิลลิลิตร, 30000 มิลลิลิตร, 40000 มิลลิลิตร และ 50000 มิลลิลิตรและทดลองวัดปริมาตรน้ำที่ระบบได้จ่ายออกมาจริง เป็นไปตามตารางที่ 4.4

ตารางที่ 4.4 การวัดปริมาตรน้ำที่วัดได้จากการทดลองเมื่อสั่งงานให้ระบบจ่ายน้ำ 10000 – 50000 มิลลิลิตร

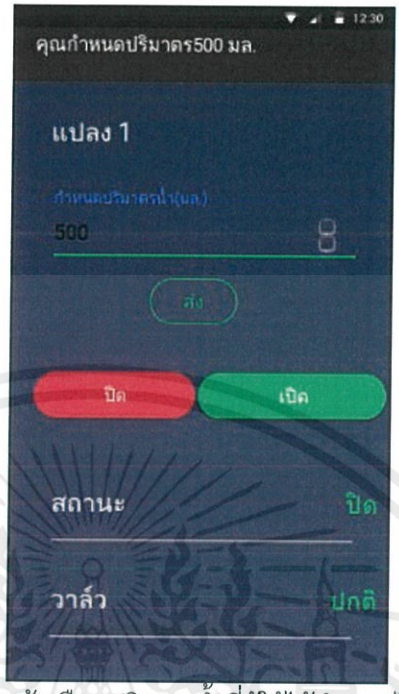
การทดลองครั้งที่	ปริมาตรน้ำที่วัดได้(มิลลิลิตร)
1	10024
2	20030
3	30020
4	40030
5	50030
เฉลี่ย	10024

4.3 การสั่งงานผ่านแอปพลิเคชันสมาร์ทโฟน

เมื่อทำการเลือกอุปกรณ์ที่ต้องการใช้งาน หน้าแอปพลิเคชันจะมีลักษณะเป็นดังรูปที่ 4.2 ซึ่งถ้าผู้ใช้ไม่ต้องการกำหนดปริมาณน้ำผู้ใช้สามารถเปิดการทำงานของวาล์วน้ำได้ทันที และในกรณีที่ผู้ใช้ต้องการกำหนดปริมาณน้ำจะมีส่วนให้ผู้ใช้กำหนดปริมาณน้ำในหน่วยมิลลิลิตรซึ่งในที่นี้ได้กำหนดไว้เป็น 500 มิลลิลิตร ดังรูปที่ 4.3 โดยต้องทำการกดส่งค่าให้อุปกรณ์สามารถควบคุมปริมาณน้ำได้ซึ่งจะมีการแจ้งเตือนแก่ผู้ใช้ว่าได้ทำการกำหนดปริมาณไปเท่าใด ดังรูปที่ 4.4 และถ้าหากผู้ใช้ทำการกำหนดปริมาณน้ำที่น้อยเกินไประบบจะไม่สามารถเปิดน้ำได้ซึ่งมีการแจ้งเตือนบอกถึงปริมาณขั้นต่ำที่ผู้ใช้ควรกำหนดคือปริมาณระหว่าง 100-9999 มิลลิลิตรเพื่อคำนึงถึงความเหมาะสมในการให้น้ำแก่แปลงเกษตร ดังรูปที่ 4.5

รูปที่ 4.2 หน้าควบคุมอุปกรณ์แปลงที่ 1

รูปที่ 4.3 การกำหนดปริมาณน้ำ 500 มิลลิลิตร

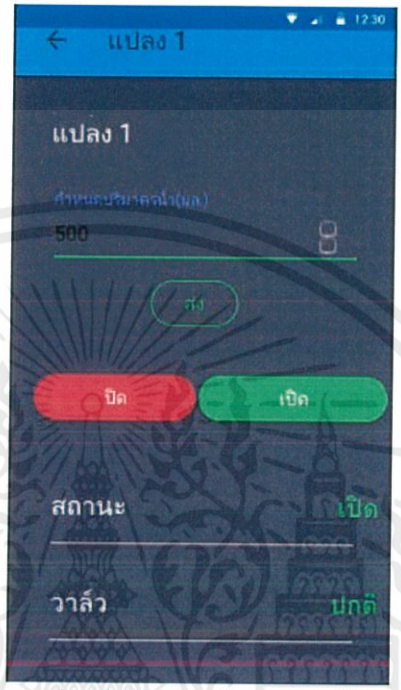


รูปที่ 4.4 การแจ้งเตือนปริมาณน้ำที่ผู้ใช้ได้กำหนดไว้ก่อนหน้านี้

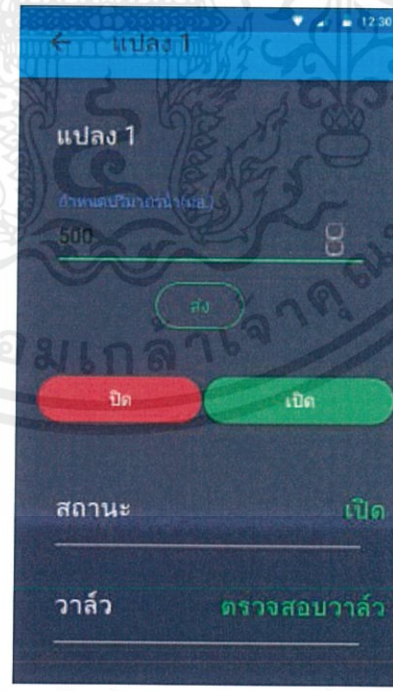
รูปที่ 4.5 การแจ้งเตือนถึงปริมาณน้ำขั้นต่ำที่เหมาะสมแก่แปลงเกษตร

ในการสั่งเปิดการทำงานของวาล์วน้ำจะมีสถานะคอยแจ้งแก่ผู้ใช้ว่าวาล์วได้เปิดหรือปิดหรือไม่โดยจะแสดงค่า “เปิด-ปิด” ในช่องของสถานะและแจ้งบอกการตรวจสอบวาล์วว่าว่าระบบได้

มีการให้น้ำแก่แปลงเกษตรจริงๆหรือไม่โดยจะแสดงว่า “ปกติ” ในกรณีที่ระบบทำงานเป็นปกติ ดังรูปที่ 4.6 และแสดงว่า “ตรวจสอบวาล์ว” ในกรณีที่ระบบไม่สามารถจ่ายน้ำได้ตามปกติ ดังรูปที่ 4.7



รูปที่ 4.6 สถานะกรณีวาล์วเปิดการทำงานและมีการจ่ายน้ำเป็นปกติ



รูปที่ 4.7 กรณีวาล์วเปิดการทำงานแต่ไม่มีการจ่ายน้ำ

4.5 การแสดงผลทาง SSH ใน Server

```
root@instance-1:/home/nongnot232323/myproject/dashboard# node udp.js
Running at Port 4001
My udp protocol start listen on port 7899
Client connected...
110.49.201.226:7899 - incoming message = hello_01
```

รูปที่ 4.8 แสดง Port ที่ใช้รัน server และข้อความจาก NB-IoT

รัน TCP server ที่ port 4001 และ รัน UDP server ที่ port 7899 โดยเริ่มต้นอุปกรณ์ NB-IoT จะติดต่อมายัง server ก่อนโดยส่งข้อความมาว่า hello_01 ดังรูปที่ 4.8

```
Password:
root@instance-1:/home/nongnot232323# cd myproject/dashboard
root@instance-1:/home/nongnot232323/myproject/dashboard# node udp.js
Running at Port 4001
My udp protocol start listen on port 7899
Client connected...
110.49.201.243:7899 - incoming message = hello_01
Client connected...
00v500
Data sent !!!
001
001
Data sent !!!
Data sent !!!
Client connected...
00v500
Data sent !!!
001
000
001
Data sent !!!
Data sent !!!
Data sent !!!
110.49.201.243:7899 - incoming message = 001
110.49.201.243:7899 - incoming message = 00F
check
000
Data sent !!!
000
Data sent !!!
110.49.201.243:7899 - incoming message = 00A
normal
110.49.201.243:7899 - incoming message = 00A
normal
110.49.201.243:7899 - incoming message = 00A
normal
Client connected...
000
Data sent !!!
110.49.201.243:7899 - incoming message = 000
Client connected...
110.49.201.243:7899 - incoming message = 00A
normal
001
Data sent !!!
110.49.201.243:7899 - incoming message = 001
110.49.201.243:7899 - incoming message = 00F
check
```

รูปที่ 4.9 ข้อมูลที่รับส่งบน udp server

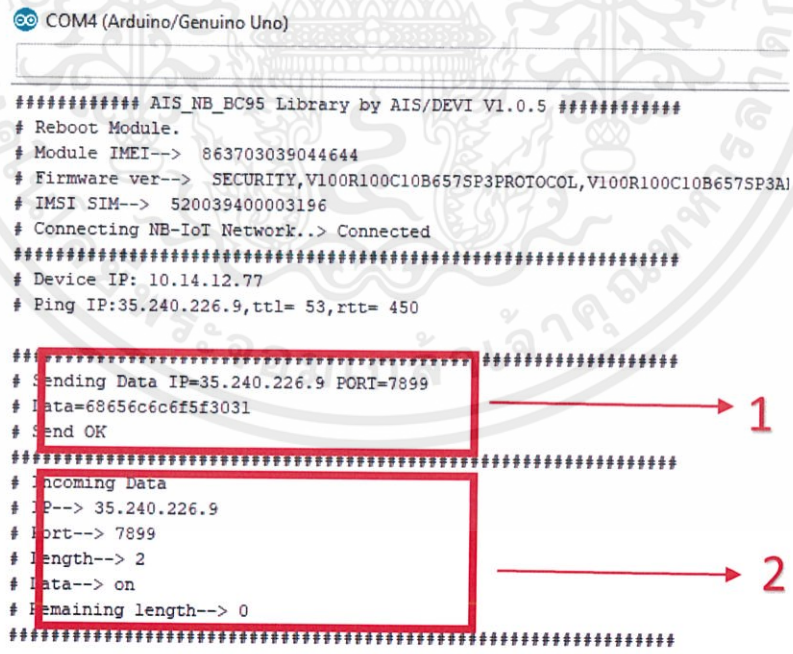
ในรูปที่ 4.9 แสดงตัวอย่างเมื่อกดคำสั่ง ON/OFF บนแอปพลิเคชันข้อความคำว่า xx1/xx0 จะถูกส่งไปยังอุปกรณ์ NB-IoT โดยที่ xx แทนหมายเลขอุปกรณ์สเลฟและ 1 แทนเปิดส่วน 0 แทนปิด หลังจากที่อุปกรณ์ NB-IoT ได้รับข้อมูลแล้วก็จะส่งไปที่สเลฟผ่านทาง power line

220 Vac เพื่อสั่งงานเปิดปิดโซลินอยด์วาล์วอีกทีหนึ่ง สถานะเปิดปิดของวาล์วจะถูกส่งกลับไปยัง server เพื่อแสดงผลบนหน้าแอปพลิเคชันและทุก ๆ 60 วินาทีอุปกรณ์ NB-IoTจะส่งผลการตรวจสอบการทำงานของวาล์วซึ่งข้อมูลที่ส่งมาจะเป็น xxA/xxF โดยที่ xx แทนหมายเลขอุปกรณ์สเลฟและ A แทนวาล์วทำงานปกติส่วน F แทนวาล์วทำงานไม่ปกติอาจเกิดการชำรุด

4.6 การแสดงผลทาง Serial monitor

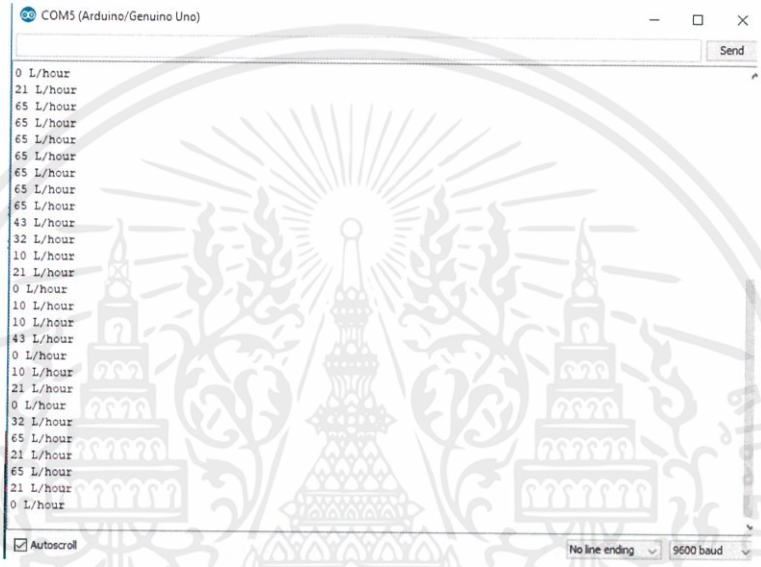
รูปที่ 4.10 ข้อมูลอุปกรณ์ NB-IoT

จากรูปที่ 4.10 เป็นการแสดงค่าที่บอกเกี่ยวกับอุปกรณ์ NB-IoT มีดังนี้ Module IMEI, Firmware ver., IMS SIM, ผลทดสอบการเชื่อต่อต่อโครงข่าง NB-IoT, Device IP และผลทดสอบการเชื่อมต่อกับ server



รูปที่ 4.11 การรับ-ส่งข้อมูลระหว่างอุปกรณ์ NB-IoTกับ server

รูปที่ 4.11 ในกรอบหมายเลข 1 คืออุปกรณ์ NB-IoT ได้ส่งข้อความไปยัง server ก่อน เพื่อให้ server รู้จักกับอุปกรณ์ตัวนี้ ในส่วนของกรอบหมายเลข 2 เมื่อมีการกดปุ่ม ON/OFF ที่หน้า dashboard ฝั่ง server จะส่งคำว่า on/off มายังอุปกรณ์นี้และข้อมูลที่รับมาจะมีรายละเอียดดังนี้ IP,Port ของ server, ความยาวของ Data และ Data ที่รับมา



รูปที่ 4.12 อัตราการไหลของน้ำที่วัดจาก Flow meter

รูปที่ 4.12 คืออัตราการไหลของน้ำที่ฝั่ง slave ซึ่งเอาไว้ให้ทราบถึงการทำงานของวาล์วดังนี้ ถ้ากดเปิดวาล์วแล้วน้ำไหลหรืออัตราการไหลมากกว่า 0 แสดงว่าวาล์วยังทำงานได้ปกติแต่ถ้าน้ำไม่ไหลหรืออัตราการไหลเป็น 0 L/hour แสดงว่าวาล์วอาจเกิดชำรุดในทางกลับกัน ถ้ากดปิดวาล์วแล้วน้ำไหลหรืออัตราการไหลมากกว่า 0 L/hour หมายความว่าวาล์วอาจเกิดชำรุดได้แต่ถ้าน้ำไม่ไหลหรืออัตราการไหลเป็น 0 L/hour แสดงว่าวาล์วทำงานได้ปกติ

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

ระบบการจำลองระบบการให้น้ำผ่านโครงข่ายการสื่อสาร NB-IoT มีการทำงานดังนี้ เมื่อทำการกำหนดปริมาตรน้ำผ่านเว็บแอปพลิเคชันหรือแอปพลิเคชันบนสมาร์ทโฟนและสั่งการทำงานให้เปิด-ปิดระบบน้ำ คำสั่งจะถูกสั่งผ่านไปยังเซิร์ฟเวอร์สู่บอร์ด NB-IoT ที่เชื่อมอยู่กับบอร์ดอะดูโนมาสเตอร์ ซึ่งจะมีไฟ LED ในการตรวจสอบสถานะของคำสั่งแต่ละครั้งว่าคำสั่งได้ถูกส่งผ่านมาหรือไม่ โดยคำสั่งจากบอร์ดอะดูโน่มาสเตอร์จะถูกส่งไปอะดูโน่สลาฟผ่านเทคโนโลยี Power Line Communication ด้วยโมดูล KQ330 ที่ทำการมอดูเลตสัญญาณแบบ FSK ผ่านสายไฟ 220 Vac ส่งข้อมูลไปยังบอร์ดสลาฟให้ทำการสั่งจ่ายน้ำตามปริมาตรที่กำหนด โดยบอร์ดสลาฟนั้นจะเชื่อมต่อกับโซลินอยด์วาล์วและ Water Flow Sensor คอยตรวจจับการไหลของน้ำเพื่อคำนวณปริมาตรของน้ำที่ไหลผ่านท่อเพื่อประโยชน์ในการควบคุมปริมาณการให้น้ำที่เหมาะสมกับการทำเกษตรกรรมซึ่งระบบสามารถแจ้งเตือนผู้ใช้งานได้ในกรณีที่วาล์วไม่ทำงานหรือกำหนดปริมาณน้ำขั้นต่ำไม่เหมาะสมแก่การใช้งาน

ในส่วนของการคำนวณปริมาตรน้ำที่วัดจาก water flow meter พบว่ามีความคลาดเคลื่อนจากที่กำหนดไว้ไม่เกิน ±0.03 ลิตรหรือ ±30 มิลลิลิตร

5.2 ข้อเสนอแนะ

1) เนื่องจากอุปกรณ์ NB-IoT นั้นเป็นอุปกรณ์ที่เชื่อมต่อด้วยสัญญาณผ่านโครงข่ายโทรศัพท์จึงจำเป็นต้องอยู่ในพื้นที่ที่มีสัญญาณเข้าถึงได้เพียงพอจึงควรตั้งไว้ภายนอกอาคารที่มีสถานที่กำบังจากสภาพอากาศต่างๆเพื่อให้สัญญาณเข้าถึงตัวอุปกรณ์ได้

2) โมดูล KQ-330 การส่งข้อมูลมีระยะทางจำกัดคือสามารถส่งได้ไกลสูงสุด 1.5 กิโลเมตร

3) เนื่องจากไฟที่ใช้ในโมดูล Power Line Communication คือแรงดันไฟที่ค่อนข้างอันตรายการจะติดตั้งนั้นควรจะมีความระมัดระวังสูงมากโดยตองมีการวางแผนการดำเนินการอย่างรัดกุมและคำนึงถึงความปลอดภัยในการใช้งาน

บรรณานุกรม

[1] มงคล พรหมเทศ. “รีเลย์และคอนแทกเตอร์.”
http://edu.e-tech.ac.th/mdec/learning/e-web/ sara010.html, 2009.

[2] มงคล พรหมเทศ. “รีเลย์และคอนแทกเตอร์.”
http://www.lpc.rmutl.ac.th/elcen/elearning/motorcontrol/module4/contactor1.html, 2009.

[3] Andrew Errity. “Getting started with Node, Express and Socket.io.”
https://github.com/aerrity/socket-click-example.

[4] Arduino. “Arduino Download.”
https://www.arduino.cc/en/Main/Software.

[5] ARTIYA. “วิธีรับส่งข้อมูลผ่าน NB-IoT Shield กับ UDP Server.”
https://medium.com/@artiya4u/ais-nb-iot-udp-connect-f86f803b61a.

[6] CSLOXINFO, “รู้จักเทคโนโลยี Cloud Computng.”
http://dccloud.csloxinfo.com/th/wecloud01/

[7] Factomart Admin, “Flow Meter มีหลักการทำงานยังไง ถึงได้วัดการไหลได้แม่นยำ?.”
https://www.factomart.com/th/factomartblog/principle-flow-meter/

[8] Factomart Admin. “หลักการทำงานของ Solenoid Valve.”
https://www.factomart.com/th/factomartblog/principle-of-solenoid-valve/.

[9] Isaranu. “สร้าง IoT protocol, database และ Dashboard บน Google Cloud Platform ด้วย Node.js และ Chart.js.”
http://code.isaranu.com/node/2017/nodejs-iot-simple-protocol-dashboard.php.

[10] Kitjabhorn Kaensuk. “Voltage Regulator Circuit) วงจรรักษาระดับแรงดัน (Voltage Regulator Circuits.”
https://www.academia.edu/4329833/Voltage_Regulator_Circuit_วงจรรักษาระดับแรงดัน_Voltage_Regulator_Circuits.

[11] PSP TECH CO, “Relay คืออะไร.”
http://bedroomlearning.blogspot.com/2016/10/relay.html.

[12] Septian Alif Farhansyah. “Design of Power Line Carrier Communication System basedon FSK-KQ330 Module.”
https://www.scribd.com/document/354142151/KQ330-Datasheet.

[13] Silver Moon. “2012.UDP socket programming in php.”

https://www.binarytides.com/udp-socket-programming-in-php/.

[14] Yin QUN, Zhang JIANBO. "Design of Power Line Communication System based on FSK-KQ330 Module."

https://www.scribd.com/document/354142151/KQ330-Datasheet.

ภาคผนวก

โค้ดโปรแกรมที่ใช้ในปริญญานิพนธ์

Arduino

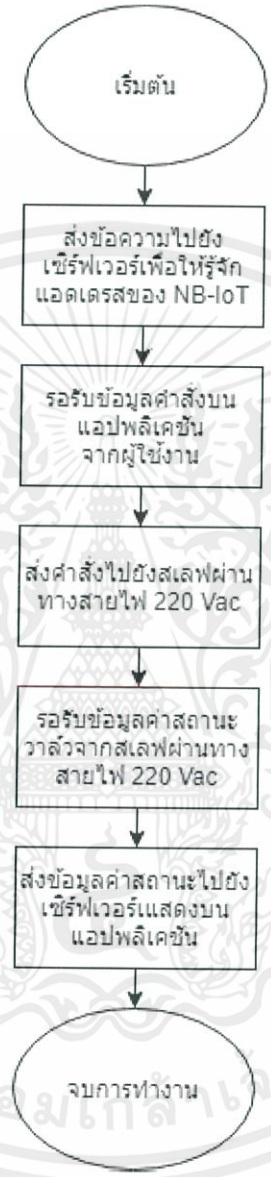
Master.ino สำหรับอะดูโน่ที่ใช้เป็นมาสเตอร์

```
#include <AIS_NB_BC95.h>
#include<SoftwareSerial.h>
SoftwareSerial my(10,11);                    // Rx,Tx for Power Line Communication
String serverIP = "35.240.226.9";            // Server IP
String serverPort = "7899";                  // Server Port
char x[9],z[5];
int led=7;
AIS_NB_BC95 AISnb;
void setup()
{
/* set up NB-IoT device */
  AISnb.debug = true;
  Serial.begin(9600);
  my.begin(9600);
  AISnb.setupDevice(serverPort);
  String ip1 = AISnb.getDeviceIP();
  delay(1000);
  pingRESP pingR = AISnb.pingIP(serverIP);
  pinMode(led, OUTPUT);
 // master send data to udp server //
  UDPSend udp = AISnb.sendUDPmsgStr(serverIP, serverPort,"hello_01");
}
```

```
void loop()
{
UDPReceive resp = AISnb.waitResponse();     // master get data from server
String y = AISnb.toString(resp.data);       // data to string
 if(y.length()==3){                         // if get data is 000 (off) or 001(on)
        digitalWrite(led,HIGH);             // led on
        delay(500);
        digitalWrite(led,LOW);              // led off
        my.listen();
        my.write(0x03);                     // send 3 string (001 or 000)
        my.print(y);                        // send data to slave
        }
else if(y.length()==6){                     // if get water volume (ex.00v100)
        my.listen();
        my.write(0x06);
        my.print(y);
        }
else if(y.length()==7){                     // if get water volume (ex.00v1000)
        my.listen();
        my.write(0x07);
        my.print(y);
        }
else if(y.length()==8){                     // if get water volume (ex.00v10000)
        my.listen();
```

```
        my.write(0x08);
        my.print(y);
        }
    else if(my.available()>0){                    // if master get data(states) from slave
        String getSt = my.readString();           // read string
        Serial.println(getSt);
            if(getSt.length()==4) {       // if data is state (x00A normal,x00F check)
                getSt.toCharArray(z,5);
                String state = String(z[1])+String(z[2])+String(z[3]);
                Serial.println(state);
    // master send state of slaves to server
    UDPSend udp = AISnb.sendUDPmsgStr(serverIP, serverPort,state);
            }
        }
}
```

บล็อกไดอะแกรมการทำงานของมาสเตอร์



Slave.ino สำหรับอะดูโน่ที่ใช้เป็นสเลฟ

```
#include<SoftwareSerial.h>
#define hallsensor 3                    // pin Flow meter
#define stepTime 1700                   // Steptime from experiment
SoftwareSerial myserial(12,13);         // Rx,Tx for Power line communication
volatile uint16_t pulseCount;
float flowRate;
float flowRateML;
float flowVol;
float inpvol;
unsigned long prevTime;
unsigned long curTime;
unsigned long delTime;
String st,y,v;
char z[10];
int led = 8;                            // pin for Control relay
int i=0,j=0;
int k;
const long interval = 60;               // set delay loop second
unsigned long previousMillis = 0;
void pulseCounter ()                    // This is the function that the interupt calls
{
pulseCount++;           // This function measures the rising and falling edge of the
}
void measureFlow ()
```

```
// this function calculate water volume and send valve state to Master
{
  detachInterrupt(1);
  flowRate=(1000*pulseCount/delTime)/5.5;          // Calculate Flowrate in L
  flowRateML = flowRate*1000/60;                   // Calculate Flowrate in mL
  flowVol += (flowRate/60)*(delTime/1000);         // Calculate Volume in L
  prevTime=curTime;
  show();                                          // display on serial monitor
  pulseCount = 0;
  attachInterrupt(1,pulseCounter ,FALLING);
if(z[3]=='1'&& k==0){                              // valve on
    if(i==3 && flowRate>0){
     Serial.println("nomal");
      myserial.listen();
      myserial.write(0x03);                        // send state to master
      myserial.print("00A");                       // state for normal
     i=0;
     k++;
     flowVol=0;
     }
   else if(i==3 && flowRate == 0){
     Serial.println("check");
     myserial.listen();
       myserial.write(0x03);                       // send state to master
      myserial.print("00F");                       // state for unusual
```

```
      i=0;
      k++;
      flowVol=0;
      }
    i++;
}
  else if(z[3]=='0' && k==0){                    // valve off
            if(j==3 && flowRate==0){
             Serial.println("nomal");
        myserial.listen();
        myserial.write(0x03);                    // send state to master
       myserial.print("00A");                    // state for normal
        j=0;
        k++;
        flowVol=0;
     }
  else if(j==3 && flowRate>0){
     Serial.println("check");
     myserial.listen();
     myserial.write(0x03);                       // send state to master
     myserial.print("00F");                      // state for unusual
     j=0;
     k++;
      flowVol=0;
     }
```

```
    j++;
   }
  if(inpvol >= 0.1){                          // if get water volume from user
// if volume is between input volume from user +- 0.03 L //
    if(flowVol >= (inpvol-0.03)  && flowVol<= (inpvol+0.03)  ){
     Serial.println(flowVol);
     digitalWrite(led,LOW);                   // vale off
     myserial.listen();
     myserial.write(0x03);
     myserial.print("000");                   // send state to master
      z[3]='0';
     inpvol=0;
     }
   }
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval*1000)
// every 1 minute send valve state to master //
  {
   if(z[3]=='1' && k>0){
      if(flowRate == 0 ){
          myserial.listen();
          myserial.write(0x03);
          myserial.print("00F"); }
    else if(flowRate > 0 ){
         myserial.listen();
```

```
            myserial.write(0x03);
            myserial.print("00A");
    }
   else if(z[3]=='0' && k>0){
        if(flowRate == 0 ){
          Serial.println("ok00");
           myserial.listen();
          myserial.write(0x03);
          myserial.print("00A");    }
  else if(flowRate > 0 ){
     myserial.listen();
     myserial.write(0x03);
     myserial.print("00F");   }
    }
       previousMillis = currentMillis;
  }
}
void show(){                              // display results on serial monitor
  Serial.print("Flow Rate: ");
  Serial.print(flowRate,2);
  Serial.print("L/min ");
  Serial.print("Flow RateML: ");
  Serial.print(flowRateML,2);
  Serial.print("ML/s ");
  Serial.print("Vol: ");
```

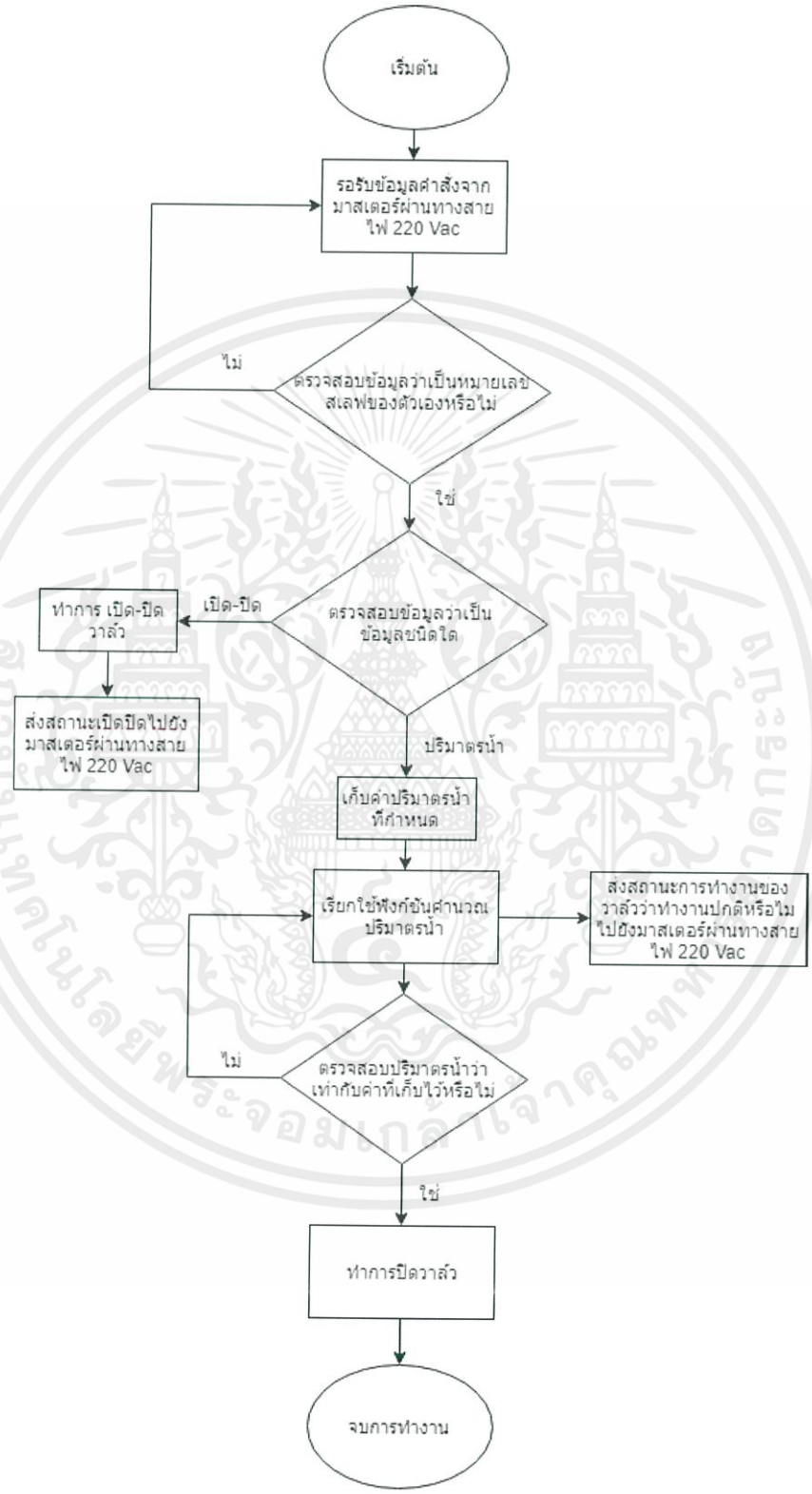
```
  Serial.print(flowVol,2);
  Serial.println("L");
  }
void setup() //
{
myserial.begin(9600);
pinMode(led, OUTPUT);
pinMode(hallsensor, INPUT);                    // initializes digital pin 2 as an input
digitalWrite(hallsensor,HIGH);
Serial.begin(9600);
attachInterrupt(1,pulseCounter,FALLING);          // Call interrupt
Serial.println("Flow Meter");
delay(1000);
prevTime=0;
flowRate=0;
flowVol=0;
pulseCount=0;
}
void loop ()
{
curTime = millis();
delTime=curTime-prevTime;
if(delTime>=stepTime){                              // every 1 min call measureFlow()
  measureFlow();
}
```

```
if(myserial.available()>0){                    // if slave get data from master
   String p=myserial.readString();
   p.toCharArray(z,10);
   Serial.println(p);
   String sl = String(z[1])+String(z[2]);
   if(sl=="00"){                               // if slave number 00
      if(z[3]=='0'){                           // off
        Serial.println("LOW");
        digitalWrite(led,LOW);                 // off valve
         myserial.listen();
         myserial.write(0x03);
        myserial.print("000");                 // send off state to master
        k=0;
        inpvol=0;
        }
      else if(z[3]=='1'){                      // on
         Serial.println("HIGH");
         digitalWrite(led,HIGH);               // on valve
         myserial.listen();
         myserial.write(0x03);
         myserial.print("001");                // send on state to master
         k=0;
         flowVol=0;
        }
```

```
else if(z[3] == 'v'){                          // define volume
        if(p.length()==5){                          // data is ?00v1   (x Lit)
          v=String(z[4]);
         Serial.println(v);
        inpvol= v.toFloat();                        // string to float (number)
        Serial.println(inpvol);
         }
        else if(p.length()==6){                     // data is ?00v10   (xx Lit)
         v=String(z[4])+String(z[5]);
        Serial.println(v);
       inpvol= v.toFloat();
       Serial.println(inpvol);
        }
       else if(p.length()==7){                      // data is ?00v0.1  (x.x or xxx Lit)
        v=String(z[4])+String(z[5])+String(z[6]);
        Serial.println(v);
      inpvol= v.toFloat();
      Serial.println(inpvol);
       }
      else if(p.length()==8){                       // data is ?00v1000   (x,xxx Lit )
       v=String(z[4])+String(z[5])+String(z[6])+String(z[7]);
      Serial.println(v);
     inpvol= v.toFloat();
     Serial.println(inpvol);
     }
```

```
            else if(p.length()==9){                    // data is ?00v10000  ( xx,xxx Lit )
              v=String(z[4])+String(z[5])+String(z[6])+String(z[7])+String(z[8]);
             Serial.println(v);
            inpvol= v.toFloat();
            Serial.println(inpvol);
            }
          }
      }
   }
}
```

บล็อกไดอะแกรมการทำงานของสเลฟ



Mobile Appilcation

ไฟล์ทั้งหมดเขียนบน Ionic framework V.3

App.component.ts

```
/* import modules */
import { Component, ViewChild } from '@angular/core';
import { Nav, Platform} from 'ionic-angular';
import { StatusBar } from '@ionic-native/status-bar';
import { SplashScreen } from '@ionic-native/splash-screen';
import { HomePage } from '../pages/home/home';
import { ListPage } from '../pages/list/list';
import { LgPage } from '../pages/lg/lg';
import { TranslateService } from '@ngx-translate/core';
import { Storage } from '@ionic/storage';
@Component({
  templateUrl: 'app.html'
})
export class MyApp {
  @ViewChild(Nav) nav: Nav;
  rootPage: any = HomePage;
  public tokenButton:boolean=true;
  pages: Array<{title: string, component: any}>;
  public a :boolean;
  public Lang:string="language";
constructor(public storage:Storage, public platform: Platform, public statusBar:
StatusBar, public splashScreen: SplashScreen,public translate:TranslateService) {
```

```
        this.initializeApp();
        translate.setDefaultLang('th');                    // set Thai language default
        this.storage.set(this.Lang,'th');                  // storage Lang parameter
        this.pages = [
        { title: 'Home', component: HomePage},
        { title: 'Devices', component: ListPage },
        { title: 'Language', component: LgPage}
        ];
  }
  initializeApp() {
        this.platform.ready().then(() => {
        this.statusBar.styleDefault();
        this.splashScreen.hide();
        });
  }
  openPage(page) {                                         // open pages function
   if(page.title=="Home")
        this.nav.setRoot(page.component);
   else
        this.nav.push(page.component);
  }
}
```

App.html

```
/* html file for menu page */
<ion-menu [content]="content" >
  <ion-header >
        <ion-toolbar>
        <ion-title >{{"Menu" | translate}}</ion-title>
        </ion-toolbar>
  </ion-header>
  <ion-content class="bg" >
        <ion-list>
        <button class="op" menuClose ion-item *ngFor="let p of pages"
(click)="openPage(p)">
        <ion-icon  color="secondary" ios="ios-home" md="md-home" *ngIf="p.title
== 'Home'"></ion-icon>
        <ion-icon color="secondary" name="switch"  *ngIf="p.title ==
'Devices'"></ion-icon>
        <ion-icon color="secondary" name="settings" *ngIf="p.title ==
'Language'"></ion-icon>
        {{p.title | translate}}
        </button>
        </ion-list>
  </ion-content>
</ion-menu>
<ion-nav [root]="rootPage" #content swipeBackEnabled="false"></ion-nav>
```

App.module.ts

```
/* file for manage modules */
import { BrowserModule } from '@angular/platform-browser';
import { ErrorHandler, NgModule } from '@angular/core';
import { IonicApp, IonicErrorHandler, IonicModule } from 'ionic-angular';
import { MyApp } from './app.component';
import { HomePage } from '../pages/home/home';
import { ListPage } from '../pages/list/list';
import { IonicStorageModule } from '@ionic/storage';
import { StatusBar } from '@ionic-native/status-bar';
import { SplashScreen } from '@ionic-native/splash-screen';
import { List2Page } from '../pages/list2/list2';
import { List3Page } from '../pages/list3/list3';
import { LgPage } from '../pages/lg/lg';
import {SocketIoModule, SocketIoConfig} from 'ng-socket-io';
import { TranslateModule, TranslateLoader } from '@ngx-translate/core';
import { HttpClientModule, HttpClient } from '@angular/common/http';
import { TranslateHttpLoader } from '@ngx-translate/http-loader';
/* socket.io server is my IP */
const config: SocketIoConfig = {url: 'http://35.240.226.9:4001',options:{}};
export function HttpLoaderFactory(http: HttpClient) {
 return new TranslateHttpLoader(http, './assets/i18n/', '.json');
 }
```

```
@NgModule({
  declarations: [
        MyApp,
        HomePage,
        ListPage,
        List2Page,
        List3Page,
        LgPage
  ],
  imports: [
        BrowserModule,
        IonicModule.forRoot(MyApp),
        SocketIoModule.forRoot(config),
        IonicStorageModule.forRoot(),
        HttpClientModule,
        TranslateModule.forRoot({
        loader: {
        provide: TranslateLoader,
        useFactory: (HttpLoaderFactory),
        deps: [HttpClient]
        }
        })
  ],
  bootstrap: [IonicApp],
  entryComponents: [
```

```
        MyApp,
        HomePage,
        ListPage,
        List2Page,
        List3Page,
        LgPage
    ],
    providers: [
        StatusBar,
        SplashScreen,
        {provide: ErrorHandler, useClass: IonicErrorHandler}
    ]
})
export class AppModule {}
```

App.scss

```
/* CSS file to create styles like fonts and colors */
.op{
        font-size: 18px;
        background-color:  #34495e;
        color: aliceblue  ;
}
```

```
.bg{
        background-color:  #34495e;
}

.toolbar-background{
        background-color: deepskyblue;
}
$toolbar-title-color:#ec03cd;
$toolbar-text-color:#e21c2c;
$toolbar-md-title-text-color:#488aff;
```

en.json

```
/* data file for transfer English language */
{
        "Home":"Home",
        "HeadHome":"Smart Irrigation System",
        "Suggest":"your master and slave devices before using this application.",
        "TurnOn":"Turn on",
        "SelectDe":"Select Plots",
        "Menu":"Menu",
        "Devices":"Devices",
        "Language":"Language",
        "LanguageTitle":"Language",
        "English":"English",
        "Thai":"ภาษาไทย",
```

```
    "Slave No.1":"Plot No.1",
    "Slave No.2":"Plot No.2",
    "SelectDevices":"Select Plots",
    "Set volume(mL)":"Set volume(mL)",
    "On-Off":"On-Off",
    "Valve":"Valve",
    "State":"State",
    "Send":"Send",
    "on":"on",
    "off":"off",
    "normal":"normal",
    "check":"check"
}
```

Th.json

```
/* data file for transfer to Thai language */
{
    "Home":"หน้าหลัก",
    "HeadHome":"ระบบการให้น้ำอัจฉริยะ",
    "Suggest":"อุปกรณ์ทั้งหมดก่อนจะใช้งานแอปพลิเคชันนี้",
    "TurnOn":"เปิด",
    "SelectDe":"เลือกแปลง",
    "Devices":"เลือกแปลง",
    "Language":"ภาษา (Language)",
    "LanguageTitle":"ภาษา",
```

```
        "Menu":"เมนู",
        "English":"English",
        "Thai":"ภาษาไทย",
        "Slave No.1":"แปลง 1",
        "Slave No.2":"แปลง 2",
        "SelectDevices":"เลือกแปลง",
        "Set volume(mL)":"กำหนดปริมาตรน้ำ(มล.)",
        "On-Off":"เปิด-ปิด",
        "Valve":"วาล์ว",
        "State":"สถานะ",
        "Send":"ส่ง",
        "on":"เปิด",
        "off":"ปิด",
        "normal":"ปกติ",
        "check":"ตรวจสอบวาล์ว"
}
```

Home.html

```
/* home page of application */
<ion-header>
  <ion-navbar >
        <button ion-button menuToggle>
        <ion-icon name="menu"></ion-icon>
        </button>
        <ion-title class="home">{{"Home" | translate}}</ion-title>
```

```
  </ion-navbar>
</ion-header>
<ion-content padding class="bg">
  <h3 class="head">{{ "HeadHome" | translate}}</h3>
  <p class="det">
        <a class="to">{{"TurnOn" | translate}}</a>{{ "Suggest" | translate}} </p>
  <img src="assets/imgs/plothome.jpg">
  <br><br>
  <button ion-button primary  class="bt" ion-button (click) ="goSelect()" >
        {{ "SelectDe" | translate}}</button>
</ion-content>
```

Home.scss

```
/* CSS file of home page */
page-home {
        .bg{
        background-color:  #34495e  ;
        }
        .bt{
        display: block;
        margin: 0 auto;
        }
        .head{
        color: aliceblue  ;
        font-size: 25px;
```

```
        }
        .det{
        color: snow;
        font-size: 20px;
        }
        .to{
        color: chartreuse;
        }
        .home{
        color: aliceblue;
        }
}
```

Home.ts

```
/* mange functions file of home page */
import { Component } from '@angular/core';
import { NavController,NavParams } from 'ionic-angular';
import { ListPage } from '../list/list';
@Component({
  selector: 'page-home',
  templateUrl: 'home.html'
})
export class HomePage {
  constructor(public navCtrl: NavController,public navParams: NavParams) {
  }
```

```
    goSelect(){                          // if click select plots button call this function
        this.navCtrl.push(ListPage);     // go to ListPage
        }
}
```

Lg.html

```
/* Page of select language */
<ion-header>
  <ion-navbar>
        <ion-title>{{"LanguageTitle" | translate}}</ion-title>
  </ion-navbar>
</ion-header>
<ion-content padding class="Bg">
        <ion-item class="Bt">
        <button   class="Bt" (click)= "Tr('en')" >{{"English" | translate}}</button>
        </ion-item>
        <ion-item class="Bt">
        <button   class="Bt" (click)= "Tr('th')" >{{"Thai" | translate}}</button>
        </ion-item>
</ion-content>
```

Lg.scss

```
/* CSS file of select language page */
.Bg{
        background-color:  #34495e  ;
}
.Bt{
        background-color:  #34495e  ;
        color: aliceblue;
        font-size: 20px;
 }
```

Lg.ts

```
/* manage function file of select language page */
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams } from 'ionic-angular';
import {TranslateService} from '@ngx-translate/core';
import { Storage } from '@ionic/storage';
@IonicPage()
@Component({
  selector: 'page-lg',
  templateUrl: 'lg.html',
})
export class LgPage {
  pages: Array<{Ltitle: string}>;
  public lg:any;
```

```
  public Lang:string="language";
  constructor(public storage:Storage,public navCtrl: NavController, public navParams:
NavParams,public translateService:TranslateService) {
        this.pages = [
        { Ltitle: 'English'},
        { Ltitle: 'Thai'}
        ];
 }


Tr(lg) {
        this.translateService.use(lg);              // translate language
        this.storage.set(this.Lang,lg);             //storage variable language
        }
}
```

List.html

```
/* html file of List page */
<ion-header>
 <ion-navbar>
        <button ion-button menuToggle>
        <ion-icon name="menu"></ion-icon>
        </button>
        <ion-title>{{"SelectDevices" | translate}}</ion-title>
 </ion-navbar>
```

```
</ion-header>
<ion-content class="bg">
 <div>
  <ion-list>
        <button ion-item *ngFor="let item of items"  class="bgbt"
(click)="itemTapped($event, item)">                          // List  plots
        <img src="assets/imgs/plot.jpg"><br/>              // display image
        {{item.title | translate}}
        </button>
  </ion-list> </div>
</ion-content>
```

List.scss

```
/* CSS file of List page */
page-list {
        .bg{
        background-color:  #34495e  ;
        }
        .bgbt{
        background-color:  #34495e  ;
        color: aliceblue;
        font-size: 20px;
        }
}
```

List.ts

```
/* Manage function of List page */
import { Component } from '@angular/core';
import { NavController, NavParams } from 'ionic-angular';
import { List2Page } from '../list2/list2';
import { List3Page } from '../list3/list3';

@Component({
  selector: 'page-list',
  templateUrl: 'list.html'
})
export class ListPage {
  selectedItem: any;
  icons: string[];
  public title: string;
  public items: Array<{title: string}>;
  public tk:boolean;
  constructor(public navCtrl: NavController, public navParams: NavParams) {
        this.selectedItem = navParams.get('item');
        this.items = [];
        for (let i = 1; i < 3; i++) {                    // show  plot (plot1 & plot2)
        this.items.push({
        title: 'Slave No.' + I
        });
        }
```

```
  }
  itemTapped(event, item) {
        if(item.title =='Slave No.1'){
        this.navCtrl.push(List2Page);                    // open page
        }

else if(item.title =='Slave No.2'){
        this.navCtrl.push(List3Page);
        }
 }
}
```

Slave.html

```
/* html file of slave page */
<ion-header>
 <ion-navbar>
        <button ion-button menuToggle>
        <ion-icon name="menu"></ion-icon>
        </button>
        <ion-title>{{"Slave No.1" | translate}}</ion-title>
 </ion-navbar>
</ion-header>
<ion-content padding class="bg" >
        <ion-label  class="head" >{{"Slave No.1" | translate}}</ion-label>
        <ion-item class="inp">
```

```
        <ion-label color="primary" class="stack" stacked>{{"Set volume(mL)" |
translate}}</ion-label>
        <ion-input [(ngModel)]="WaVol" class="number" type="number"
max="99999" min="100"></ion-input>                    // input for water volume
</ion-item>
        <br>
        <button   ion-button (click) ="presentToast()" class="send" color="secondary"
ion-button round outline >{{"Send" | translate}}</button>
        <br><br>
  <div class="tg">
        <button   ion-button (click) ="ButtonClickedON()"   class="on"
color="secondary" ion-button round >{{"on" | translate}}</button>
        <!--<ion-label class="tgonoff" >{{"On-Off" | translate}}</ion-label>-->
  <!-- <ion-toggle color="secondary" class="togglec"  [(ngModel)]="toggle_1"
(ngModelChange)="ButtonClicked()"></ion-toggle>-->
  <button   ion-button (click) ="ButtonClickedOFF()"  class="off"  color="danger" ion-
button round >{{"off" | translate}}</button>
  </div>
  <br><br>
  <ion-item class="tg">
        <ion-label  class="st" >{{"State" | translate}}</ion-label>
        <ion-label class="statec" color="secondary">{{state | translate}}</ion-label>
  </ion-item><br>
 <ion-item class="valveState">
        <ion-label  class="che">{{"Valve" | translate}}</ion-label>
```

```
        <ion-label class="checkc"  color="secondary">{{check | translate}}</ion-
label>
</ion-item>
</ion-content>
```

slave.module.ts

```
/* manage module file of slave */
import { NgModule } from '@angular/core';
import { IonicPageModule } from 'ionic-angular';
import { List2Page } from './list2';
import { ListPage } from '../list/list';
@NgModule({
  declarations: [
        List2Page,
        ListPage
  ],
  imports: [
        IonicPageModule.forChild(List2Page),
  ],
})
export class List2PageModule {}
```

slave.scss

```
/* CSS file of slave */
page-list2 {
        .bg{
        background-color:  #34495e  ;
        }
        .tg{
        background-color: #34495e ;
        }
        .valveState{
        background-color: #34495e ;
        }
        .inp{
        background-color: #34495e ;
        }
        .number:focus{
        background-color:lightblue;
        }
        .stack{
        font-size: 10;
        }
        .send{
        display: block;
        margin: 0 auto;
```

```
}
.send:active {
transform: translateY(4px);
}
 .statec{
position: fixed;
right: 0px;
font-size: 20px;
font-weight: bolder;
 }
 .checkc{
position: fixed;
right: 0px;
font-size: 20px;
font-weight: bolder;
 }
 .st{
font-size: 20px;
color: aliceblue;
  }
 .tgonoff{
color: aliceblue;
font-size: 20px;
 }
 .head{
```

```
    color: aliceblue;
    font-size: 20px;
    margin-left:5%;
     }
    .che{
    font-size: 20px;
    color: aliceblue; }
    .toggle_1{
    margin-left: 100%;
    }
   .off{
    width: 50%;
    }
     .off:active {
    transform: translateY(4px);
     }
    .on{
   position: fixed;
   right: 0px;
   width: 50%;
     }
    .on:active {
   transform: translateY(4px);
     }
}
```

slave.ts

```
/* manage function file of slave */
import { Component } from '@angular/core';
import { NavController,NavParams, Icon  } from 'ionic-angular';
import { Socket } from 'ng-socket-io';
import { Observable } from 'rxjs/Observable';
import { ToastController } from 'ionic-angular';
import { Storage } from '@ionic/storage';
@Component({
  selector: 'page-list2',
  templateUrl: 'list2.html',
})
export class List2Page {
  WaVol:any;
  public state: any = 'off';
  public check: any = 'normal';
  key2:string="stateOnOff";
  key3:string="stateCheck";
  Lang:string="language";
  constructor(public storage: Storage,public navCtrl: NavController,public
socket:Socket,public toastCtrl: ToastController,public navParams: NavParams) {
        this.getMessages().subscribe(data => {
        console.log(data);
        if(data == 'off' || data == 'on'){
```

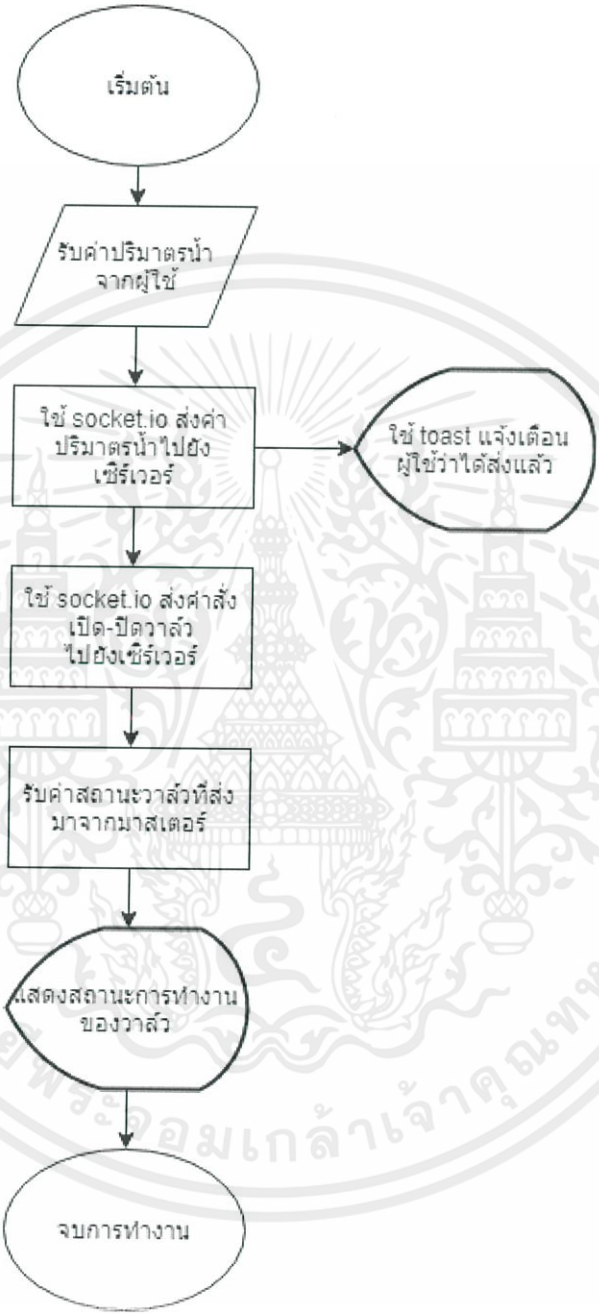
```
        this.storage.set(this.key2,data);                    // storage salve state
        this.state=data;                                     // display slave state
        }
        else if(data =='normal' || data == 'check'){
        this.storage.set(this.key3,data);
        this.check=data;                                     // display check state of slave
        }
        });
}
ionViewCanEnter(){                                           // storage state when  leave page
        console.log("L2Canenter");
        this.storage.get(this.key2).then((val) => {
        console.log('*State',val);
        if(val==null){
        this.state='off';
        val=this.state;
        this.storage.set(this.key2,this.state);              // function for storage states
        }
        else{
        this.state=val;
        this.storage.set(this.key2,val);
        }
        });
        this.storage.get(this.key3).then((val) => {
        console.log('*Check',val);
```

```
        if(val==null){
        this.check='normal';
        val=this.check;
        this.storage.set(this.key3,this.check);
        }
        else{
        this.check=val;
        this.storage.set(this.key3,val);
        }
        });
}
  ionViewDidEnter(){                    // when enter this page
        console.log("L2didenter");
        this.storage.get(this.Lang).then((val) => {
        console.log(val);
        });
  }
ButtonClickedON() {                     // when click on
 this.socket.connect();
 this.socket.emit('clicked','001');     // socket send 001 to server
  }
  ButtonClickedOFF() {                  // when click on
        this.socket.connect();
        this.socket.emit('clicked','000');      // socket send 000 to server
}
```

```
getMessages() {                                    // when socket get data
        let observable = new Observable(observer => {
        this.socket.on('UpdateState',(data) => {
        observer.next(data);
        });
        })
        return observable;
}
presentToast() {                                   // show text when click send button
        this.storage.get(this.Lang).then((val) => {
        let toast = this.toastCtrl.create({
        message: "You set "+this.WaVol+"  mL",
        duration: 3000,
        position: 'top'
        });
        // English Language
        let toast2 = this.toastCtrl.create({
        message: "Please select a value that is between 100 to 9999 ml.",
        duration: 3000,
        position: 'middle'
        });
        // Thai Language
        let toastTH = this.toastCtrl.create({
        message: "คุณกำหนดปริมาตร"+this.WaVol+"  มล.",
        duration: 3000,
```

```
        position: 'top'
        });
        let toast2TH = this.toastCtrl.create({
        message: "กรุณากำหนดปริมาตรที่อยู่ระหว่าง 100 ถึง 9999 มล.",
        duration: 3000,
        position: 'middle'
        });
 if(val=='th'){                                          // if language is Thai
        if(this.WaVol>=100 && this.WaVol<=99999  ){
        toastTH.present();                               // show text
        this.socket.emit('clicked',"00v"+this.WaVol);   // socket send volume to server
        }
        else{
        toast2TH.present();
        }
        }
else{                                                   // if language is English
        if(this.WaVol>=100 && this.WaVol<=99999  ){
        toast.present();
        this.socket.emit('clicked',"00v"+this.WaVol);
        }
        else{
        toast2.present();
        }
        } }); } }
```

บล็อกไดอะแกรมการทำงานของ Mobile Application



Server

Server.js ไฟล์ที่ใช้รันบน cloud server

```
const express = require('express');
const app = express();
var nbiotdb = mongojs('nbiotdb');
var server = require('http').createServer(app);
var io = require('socket.io')(server);
/* Setup UDP protocol connection */
var dgram = require("dgram");
var server_udp = dgram.createSocket("udp4");
var PORT = 7899;
var remote_port, remote_address;

app.use(express.static('myproject/dashboard'));
/* Start event "listening" connection from IoT */
 server_udp.on("listening", function(){
    var address = server_udp.address();
    console.log("My udp protocol start listen on port " + address.port);
});
server_udp.on("message", function (message, remote) {
/* storage NB-IoT address */
    remote_port = remote.port;
    remote_address = remote.address;
    console.log(remote.address + ":" + remote.port +' - incoming message = ' +
message);
```

```
/* socket send state on-off */
   if(message == '001'){
        io.emit('UpdateState',"on");          // socket send data to clients
}
else if(message == '000'){
        io.emit('UpdateState',"off");
}
 else if(message == '011'){
        io.emit('UpdateState',"on2");
}
else if(message == '010'){
        io.emit('UpdateState',"off2");
}
/* socket send state normal-check valve */
 else if(message == '00A'){
  console.log('normal');
        io.emit('UpdateState',"normal");
}
 else if(message == '00F'){
  console.log('check');
        io.emit('UpdateState',"check");
}
 else if(message == '01A'){
  console.log('normal');
        io.emit('UpdateState',"normal2");
```

```
    }
     else if(message == '01F'){
       console.log('check');
           io.emit('UpdateState',"check2");
    }
});
/* listening socket */
io.on('connection', function(client) {
        console.log('Client connected...');
/* when the server receives clicked message, do this */
        client.on('clicked', function(data) {
         console.log(data);]
/* server send data to NB-IoT */
  server_udp.send(data,remote_port, remote_address,function(error){
  if(error){
    client.close();
  }else{
    console.log('Data sent !!!');  }
}); }); });
server.listen(4001,function(){                    //TCP server run on port 4001
console.log("Running at Port 4001");
});
server_udp.bind(PORT);                            // UDP server bind port 7899
```

บล็อกไดอะแกรมการทำงานบนเซิร์ฟเวอร์