ระบบควบคุมระดับน้ำแบบไร้สายและการมอนิเตอร์
WIRELESS LEVEL CONTROL SYSTEM AND MONITOR

ณัฐ เตโชสกลดี
ตรียุทธ วรรณพิรุณ
ธนัชพร บุญพาชื่น

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมระบบควบคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2560

ระบบควบคุมระดับน้ำแบบไร้สายและการมอนิเตอร์

WIRELESS LEVEL CONTROL SYSTEM AND MONITOR

ณัฐ เตโชสกลดี

ตรียุทธ วรรณพิรุณ

ธนัชพร บุญพาชื่น

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2560

WIRELESS LEVEL CONTROL SYSTEM AND MONITOR

NUT	TECHOSAKONDEE
TREEYUT	WANNAPIRUN
THANATPORN	BOONPACHUEN

THIS THESIS IS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN CONTROL ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTTTUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2017

ปริญญานิพนธ์ปีการศึกษา 2560

ภาควิชาวิศวกรรมการวัดและควบคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบควบคุมระดับน้ำแบบไร้สายและการมอนิเตอร์

WIRELESS LEVEL CONTROL SYSTEM AND MONITOR

ผู้จัดทำ นายณัฐ เตโชสกลดี 57010405

นายตรียุทธ วรรณพิรุณ 57010488

นางสาวธนัชพร บุญพาชื่น 57010576

..... อาจารย์ที่ปรึกษา

(ศาสตราจารย์ ดร.วันชัย ริ้วรุจา)

..... อาจารย์ที่ปรึกษา

(ผู้ช่วยศาสตราจารย์ ดร.วรรณดี เพชรมณีล้ำค่า)

ระบบควบคุมระดับน้ำแบบไร้สายและการมอนิเตอร์

โดย

นายณัฐ	เตโชสกลดี	57010405
นายตรียุทธ	วรรณพิรุณ	57010488
นางสาวธนัชพร	บุญพาชื่น	57010576

อาจารย์ที่ปรึกษา

ศาสตราจารย์ ดร.วันชัย ริ้วรุจา

ผู้ช่วยศาสตราจารย์ ดร.วรรณดี เพชรมณีล้ำค่า

ปีการศึกษา 2560

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอการควบคุมระดับน้ำแบบไร้สายและการมอนิเตอร์ จัดทำขึ้นเพื่ออธิบายถึงขั้นตอนการทำงานของระบบควบคุมระดับน้ำให้เป็นไปตามค่าที่ต้องการ การเขียนโปรแกรมเพื่อควบคุมระบบและการออกแบบหน้าแสดงผลสำหรับการใช้งาน จุดประสงค์หลักของการจัดทำโครงงานนี้คือ เพื่อจำลองการทำงานในระบบอุตสาหกรรมขนาดใหญ่ โดยใช้เทคโนโลยี IOT และอุปกรณ์ที่รองรับ เพื่อศึกษาการทำงานว่ามีประสิทธิภาพเพียงพอต่อการใช้งานจริงได้หรือไม่ หรือมีเสถียรภาพในการทำงานมากน้อยเพียงใด

WIRELESS LEVEL CONTROL SYSTEM AND MONITOR

By

Mr. Nut Techosakondee 57010405

Mr. Treeyut Wannapirun 57010488

Miss Thanatporn Boonpachuen 57010576

Advisors

Prof. Dr. Vanchai Riewruja

Asst. Prof. Dr. Wandee Petchmaneelumka

Academic year 2017

ABSTRACT

This report presents a wireless level control system and monitor. The operation of this system provides the control of the desired level via LabVIEW program with wireless communication. Programming for control the system and design of user interface are implemented. The purpose of this project is to simulate the large industrial systems by using IOT technology and compatible devices. The performance and stability of this system are studied.

กิตติกรรมประกาศ

การจัดทำปริญานิพนธ์ฉบับนี้ สามารถสำเร็จลุล่วงไปได้ด้วยดี เนื่องจากได้รับการสนันสนุนและความช่วยเหลือเป็นอย่างดีจากศาสตราจารย์ ดร. วันชัย ริ้วรุจา และ ผู้ช่วยศาสตราจารย์ ดร. วรรณดี เพชรมณีล้ำค่า ที่ให้คำปรึกษาเป็นอย่างดีมาโดยตลอดรวมทั้งยังเอื้อเฟื้ออุปกรณ์สำหรับการทำโครงงานและความช่วยเหลืออื่นๆ ที่เป็นประโยชน์ต่อโครงงาน ทางคณะผู้จัดทำจึงขอขอบพระคุณไว้ ณ โอกาสนี้

ขอขอบพระคุณศาสตราจารย์ ดร. วันชัย ริ้วรุจา และ ผู้ช่วยศาสตราจารย์ ดร. วรรณดี เพชรมณีล้ำค่า ที่ทำหน้าที่เป็นอาจารย์ที่ปรึกษา และคอยดูแลมาโดยตลอด

ขอบคุณพี่ เพื่อน ทุกคนที่ให้ความช่วยเหลือหาทางแก้ปัญหา และช่วยสนับสนุนอุปกรณ์ที่ขาดเหลือยู่เสมอ รวมทั้งคอยถามไถ่ความคืบหน้าของโครงงาน

สุดท้ายนี้ทางคณะผู้จัดทำขอกราบขอบพระคุณบิดา มารดา และครอบครัวที่เป็นกำลังใจที่ดีตลอดมา รวมถึงสนับสนุนในเรื่องของงบประมาณที่ขาดเหลือ และยังสร้างแรงบันดาลใจที่ดีในการทำโครงงานนี้จนสำเร็จลุล่วงและเสร็จสมบูรณ์ลงได้

ผู้จัดทำ

นายณัฐ เตโชสกลดี

นายตรียุทธ วรรณพิรุณ

นางสาวธนัชพร บุญพาชื่น

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญรูป

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

สารบัญตาราง

บทที่ 1

บทนำ

1.1 ความเป็นมาของปริญญานิพนธ์

ในปัจจุบัน IOT (Internet of things) ได้เข้ามามีบทบาทในการดำเนินชีวิตเรามากขึ้น รวมทั้งการทำงานในระบบอุตสาหกรรมด้วย มีการนำเทคโนโลยี IOT เข้ามาใช้และพัฒนาระบบในการส่งข้อมูลเพื่อใช้ในการมอนิเตอร์ จากระยะไกล ซึ่งในระบบอุตสาหกรรมขนาดใหญ่เอง แต่เดิมก็มีระบบในการรับส่งข้อมูลและการควบคุมจากระยะไกลที่เรียกว่า SCADA (Supervisory control and data acquisition) แต่เนื่องจากระบบนี้มีขนาดใหญ่และราคาแพง ทำให้ระบบอุตสาหกรรมหันมาให้ความสนใจเกี่ยวกับ IOT มากขึ้น เนื่องจากราคาถูกกว่า มีประสิทธิภาพในการทำงานที่สามารถทดแทนเครื่องมือขนาดใหญ่ได้ และสามารถพัฒนาให้ตอบสนองกับความต้องการในการใช้งานที่หลากหลายได้ อีกทั้งยังสามารถส่งข้อมูลขึ้นสู่ระบบ Cloud เพื่อใช้มอนิเตอร์จากระยะไกลผ่านอุปกรณ์อื่นๆ นอกจากคอมพิวเตอร์ภายในห้องควบคุม

ระบบควบคุมระดับน้ำแบบไร้สายและการมอนิเตอร์ จึงได้ถูกจัดทำขึ้นเพื่อจำลองการทำงานในระบบอุตสาหกรรมขนาดใหญ่โดยใช้เทคโนโลยี IOT และอุปกรณ์ที่รองรับ เพื่อศึกษาการทำงานว่ามีประสิทธิภาพเพียงพอต่อการใช้งานจริงได้หรือไม่ หรือมีเสถียรภาพในการทำงานมากน้อยเพียงใด โดยผู้จัดทำเลือกบอร์ด NodeMCU ESP8266 develope kit V1.0 เป็นตัวควบคุม เนื่องจากมีคุณสมบัติเพียงพอต่อการใช้งาน มีชิป Wi-Fi ในตัวเพื่อใช้ประมวลผลค่าที่ได้จากเซนเซอร์ และเชื่อมต่อแบบไร้สายไปยัง Server โดยโปรแกรมที่ใช้เป็น Server รองรับค่าจากบอร์ดตัวควบคุมคือ โปรแกรม LabVIEW 2017 student edition ซึ่งเป็นโปรแกรมที่ใช้ออกแบบหน้าจอมอนิเตอร์ เพื่อดูค่าพารามิเตอร์ที่รับมาผ่านกราฟฟิกที่เข้าใจง่าย และสามารถใช้ Function block เขียนโปรแกรมควบคุมระดับน้ำโดยใช้ตัวปรับปรุง PID ในการควบคุมระดับน้ำให้ได้ตามต้องการผ่านการปรับค่า Setpoint จากหน้าจอมอนิเตอร์ สามารถวิเคราะห์เสถียรภาพของระบบได้จากการแสดงผลกราฟในหน้าจอมอนิเตอร์ที่แสดงผลแบบ Real time นอกจากนี้ยังเพิ่มขีดความสามารถในการทำงาน โดยผู้จัดทำได้เพิ่มระบบการมอนิเตอร์จากระยะไกล โดยการส่งข้อมูลเข้าสู่ระบบ Cloud platform ของ NECTEC ที่ชื่อ NETPIE เพื่อเก็บข้อมูลและแสดงผลค่าพารามิเตอร์ต่างๆ ผ่านอุปกรณ์ตัวอื่นนอกเหนือจากเครื่อง Server ได้อีกด้วย

1.2 วัตถุประสงค์ในการทำปริญญานิพนธ์

1. เพื่อศึกษาการทำงานของบอร์ด NodeMCU ESP8266 develope kit V1.0
2. เพื่อศึกษาการควบคุมแบบ PID Controller
3. เพื่อสร้างหน้าจอสำหรับติดตามและแสดงผล และหน้าควบคุมสำหรับผู้ใช้งาน
4. เพื่อศึกษาการทำงานของ Pressure sensor, Flow sensor และ Actuator
5. เพื่อศึกษาการทำงานของโปรแกรม LabVIEW 2017
6. เพื่อออกแบบและสร้างแบบจำลองควบคุมถังน้ำอัตโนมัติ
7. เพื่อศึกษาและเขียนโปรแกรม Arduino
8. เพื่อศึกษาการรับส่งข้อมูลและการทำงานได้จากระยะไกล
9. เพื่อศึกษาอุปกรณ์และเทคโนโลยี IOT

1.3 ขอบเขตของโครงงาน

1. สร้างระบบจำลองไร้สายผ่าน TCP-IP ที่สามารถควบคุมระบบควบคุมระดับน้ำ ได้สำเร็จและสามารถใช้งานได้จริง

2. ฝึกเขียนโปรแกรมในการสั่งงาน Arduino เพื่อควบคุมระบบควบคุมระดับน้ำร่วมกับโปรแกรม LabVIEW 2017 ในการรับ-ส่งข้อมูล แบบไร้สายรวมทั้งแสดงผลผ่านจอมอนิเตอร์

3. สามารถนำ Pressure sensor, Flow sensor และ Actuator มาประยุกต์ใช้งานกับระบบควบคุมระดับน้ำแบบไร้สายได้

4. สามารถควบคุมความเร็วของมอเตอร์ผ่านระบบ Wi-Fi โดยควบคุมผ่านจอมอนิเตอร์

5. สามารถมอนิเตอร์ได้จากระยะไกลผ่านอุปกรณ์พกพา

1.4 ขั้นตอนการดำเนินโครงงาน

1. ศึกษาทฤษฎีที่เกี่ยวข้องกับการทำงานของระบบทั้งหมด ได้แก่ หลักการทำงานของ NodeMCU การเขียนโปรแกรม Arduino การใช้โปรแกรม LabVIEW 2017 การรับ-ส่งข้อมูลผ่าน TCP/IP การทำงานของเซนเซอร์ การรับ-ส่งข้อมูลขึ้น NETPIE รวมถึงการออกแบบโครงสร้างของระบบจำลอง

2. วางแผนการทำงาน

3. ออกแบบระบบ วงจร และโครงสร้างของระบบควบคุมน้ำแบบไร้สาย

4. ศึกษาส่วนประกอบ และอุปกรณ์ที่จำเป็นจะต้องใช้ภายในระบบ

5. สร้างระบบ ติดตั้งอุปกรณ์ และเขียนโปรแกรมการควบคุม

6. ทดลองและจดบันทึกผลการทำงาน หากมีข้อผิดพลาดก็ทำการแก้ไข และทดลองใหม่

7. สรุปผลการทำงาน

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถนำระบบนี้ไปประยุกต์ใช้กับอุตสาหกรรมได้ในอนาคต

2. สามารถเขียนโปรแกรมควบคุมการทำงานของระบบ และควบคุมการทำงานผ่านมอนิเตอร์ โดยใช้โปรแกรม LabVIEW 2017

3. สามารถควบคุมระดับน้ำอัตโนมัติแบบไร้สายได้

4. สามารถควบคุมการทำงานของมอเตอร์ได้

5. มีความรู้ความเข้าใจในการเขียนโปรแกรม Arduino

6. สามารถดูการทำงานของระบบผ่านจอมอนิเตอร์ได้จากระยะไกลผ่าน NETPIE

1.6 รายละเอียดของปริญญานิพนธ์

เนื้อหาที่จะกล่าวในปริญญานิพนธ์ฉบับนี้ประกอบด้วย 5 บท กับอีก 2 ภาคผนวกดังนี้

บทที่ 1 บทนำ เป็นการกล่าวถึงที่มาและความสำคัญของปริญญานิพนธ์ วัตถุประสงค์ของการทำโครงงาน ขอบเขตของโครงงาน ขั้นตอนการศึกษา และจัดทำโครงงาน ประโยชน์ที่คาดว่าจะได้รับและรายละเอียดของปริญญานิพนธ์

บทที่ 2 ทฤษฎี หลักการ และความรู้ที่เกี่ยวข้องในส่วนของซอฟต์แวร์ เพื่อใช้ในการจำลอง Server เพื่อรับค่าจากบอร์ดตัวควบคุม และการส่งค่าขึ้นระบบ Cloud

บทที่ 3 ขั้นตอนการดำเนินงาน

บทที่ 4 ผลการทดลอง

บทที่ 5 สรุปผลการดำเนินงานปัญหา และแนวทางการปรับปรุงโครงงาน

ภาคผนวก ก การติดตั้งโปรแกรมที่เกี่ยวข้อง

ภาคผนวก ข โปสเตอร์

บทที่ 2

ทฤษฎีและข้อมูลที่เกี่ยวข้อง

2.1 เทคโนโลยี Wi-Fi

2.1.1 Wi-Fi

Wi-Fi หรือ Wireless-Fidelity เป็นเทคโนโลยีไร้สาย ภายใต้การสื่อสาร มาตรฐาน IEEE 802.11 ซึ่งเป็นมาตรฐานที่ถูกอนุมัติให้ใช้จาก IEEE (The institute of electrical and electronics engineers) ซึ่งเป็นผู้พัฒนาและจัดตั้งมาตรฐานของ Wi-Fi

Wi-Fi เป็นเทคโนโลยีเครือข่ายไร้สาย เพื่อให้อุปกรณ์คอมพิวเตอร์สามารถสื่อสารกันได้บนมาตรฐานการทำงานแบบเดียวกัน Wi-Fi จะใช้คลื่นความถี่วิทยุและคลื่นความถี่อินฟาเรดในการรับส่งข้อมูล จึงสามารถทะลุผ่านสิ่งกีดขวางได้ทำให้การใช้งานบนเคลือข่ายไร้สายมีความคล่องตัว และสะดวกสบายมากขึ้น โดยข้อมูลจะมีการรับส่งผ่านคลื่นวิทยุที่คลื่นความถี่ 2.4 GHz ด้วยความเร็ว 11 Mbps ระยะห่างได้ประมาณ 300 ฟุต ในปัจุบันนิยมใช้ Wi-Fi เพื่อต่อกับอินเตอร์เน็ต ผ่านแอคเซสพอยต์ (Access Point, AP) หรือฮอตสปอต (Hotspot) ซึ่งสามารถครอบคลุมประมาณ 20 เมตร ภายในอาคาร

2.1.2 รูปแบบการเชื่อมต่อของ Wi-Fi

1. การเชื่อมต่อแบบ Infrastructure mode

เป็นระบบเครือข่ายไร้สาย Client/Server จะมีคอมพิวเตอร์หลักเครื่องหนึ่งเป็น Server ซึ่งทำหน้าที่เสมือนเป็นตัวเก็บข้อมูลระยะไกล และประมวลผลบางอย่างให้กับ Client สามารถให้บริการเครื่องลูกข่ายได้ถึง 15-50 อุปกรณ์ เหมาะสำหรับการนำไปขยายเครือข่าย

2. การเชื่อมต่อแบบ Ad-Hoc mode

เป็นระบบเครือข่ายไร้สายที่สามารถทำการเชื่อมต่อกัน โดยไม่จำเป็นต้องผ่านอุปกรณ์กระจายสัญญาณ ก็สามารถสื่อสารหรือแลกเปลี่ยนข้อมูลได้ แต่ไม่สามารถติดต่อสื่อสารหรือแลกเปลี่ยนข้อมูลกับอุปกรณ์ที่มีสัญญาณได้

2.1.3 มาตรฐานของเทคโนโลยี Wi-Fi

1. IEEE 802.11b

ถูกเผยแพร่เมื่อปี พ.ศ. 2542 ซึ่งมีการนิยมใช้อย่างแพร่หลาย ใช้เทคโนโลยีที่เรียกว่า CCK (Complimentary code keying) ผนวกกับ DSSS (Direct sequence spread spectrum) สามารถรับส่งข้อมูลด้วยความเร็วสูงสุดที่ 11 Mbps และจะทำงานที่คลื่นความถี่ 2.4 GHz ซึ่งมีความสามารถทำงานร่วมกับอุปกรณ์อื่นๆ ได้หลากหลาย

2. IEEE 802.11a

ถูกเผยแพร่เมื่อปี พ.ศ. 2542 เป็นระบบเครือข่ายไร้สายที่มีประสิทธิภาพสูง มีความเร็วในการรับส่งข้อมูลที่ 54 Mbps ทำงานที่ย่านความถี่ 5 GHz ดังนั้นปัญหาการรบกวนคลื่นความถี่จึงไม่เกิดหรือเกิดได้น้อย แต่ระยะทางการเชื่อมต่อของสัญญาณค่อนข้างสั้นที่ประมาณ 30 เมตร จากจุดกระจายสัญญาณ ในขณะที่ประเทศไทยไม่อนุญาตให้ใช้คลื่นความถี่ 5 GHz ทำให้มาตรฐานนี้ไม่เป็นที่นิยมมากนัก

3. IEEE 802.11g

ถูกเผยแพร่เมื่อปี พ.ศ. 2546 เป็นระบบเครือข่ายไร้สายที่ความถี่ 2.4 GHz และรับส่งข้อมูลที่ความเร็ว 36-54 Mbps สามารถใช้ร่วมกับ IEEE 802.11b ได้ จึงทำให้ไม่จำเป็นที่จะต้องเปลี่ยนแปลงโครงสร้างของระบบ เพียงแต่อัพเกรดเฟิร์มแวร์เท่านั้น ดังนั้นมาตรฐานนี้จึงเป็นที่ยอมรับจากผู้ใช้เป็นจำนวนมากหากมีราคาไม่แพงจนเกินไป

4. IEEE 802.11n

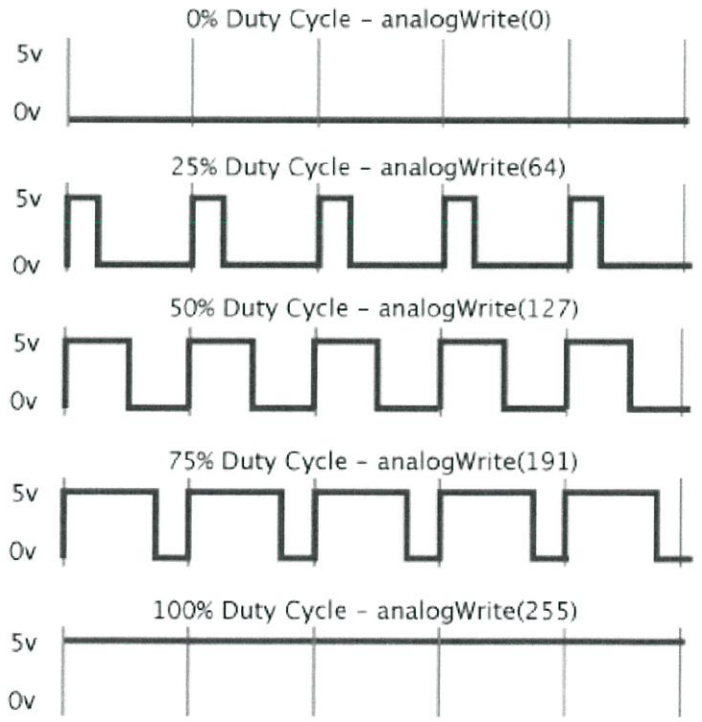
ในปัจจุบันมีการพัฒนาจากมาตรฐานเดิมเพื่อเพิ่มความเร็วในการรับส่งข้อมูลให้ได้สูงสุด 300-600 Mbps โดยมาตรฐานนี้จะมีการเพิ่มการสื่อสารทั้งรับและส่งแบบหลายช่องสัญญาณ (MIMO : Multiple-input Multiple-output) และทำงานด้วยช่วงความห่างของสัญญาณอยู่ที่ 40 MHz ซึ่งการนำ MIMO มาใช้จะช่วยเพิ่มความสามารถในการรับส่งและกู้คืนสัญญาณได้ดีกว่า รวมถึงการแยกแยะสัญญาณและแปลงสัญญาณให้เป็นช่องสัญญาณเดียว (SDM : Spatial division multiplexing)

2.2 สัญญาณ PWM

การมอดูเลชันความกว้างของสัญญาณพัลส์ (Pulse width modulation) หรือ PWM คือ การนำสัญญาณมาเปรียบเทียบสัดส่วนและความกว้างของสัญญาณพัลส์ โดยจะนำสัญญาณสามเหลี่ยมและสัญญาณที่ต้องการปรับมาเปรียบเทียบกัน การนำ PWM มาใช้ในระบบดิจิตอล จะใช้เป็นค่าของดิวตี้ไซเคิล (Duty cycle) ซึ่งค่าของดิวตี้ไซเคิลคือ ช่วงความกว้างของพัลส์ที่มีสถานะลอจิกสูง โดยคิดสัดส่วนเป็นเปอร์เซ็นต์จากความกว้างของพัลส์ทั้งหมด ยกตัวอย่างเช่น ถ้าค่าดิวตี้ไซเคิลมีค่าเท่ากับ 50% หมายความว่าใน 1 รูปสัญญาณพัลส์ จะมีช่วงที่เป็นสถานะลอจิกสูงอยู่ครึ่งหนึ่ง และสถานะลอจิกต่ำอยู่ครึ่งหนึ่ง และหากค่าดิวตี้ไซเคิลมากหมายความว่า ค่าความกว้างของพัลส์ที่สถานะลอจิกสูงก็จะมีความกว้างมาก หากค่าดิวตี้ไซเคิลมีค่าเท่ากับ 100% หมายความว่าจะไม่มีสถานะลอจิกต่ำเลย ดังรูปที่ 2.1 เป็นรูปแสดงค่าดิวตี้ไซเคิลแบบ 8 Bit ค่าความละเอียดจะอยู่ที่ 0-255 โดยค่าดิวตี้ไซเคิลสามารถหาได้จากสมการที่ (2.1)

ค่าดิวตี้ไซเคิล = (ช่วงของสัญญาณพัลส์/คาบเวลาทั้งหมดของสัญญาณ) x 100 (2.1)

ซึ่งโมดูล PWM แต่ละตัวจะมีค่าลอจิกไม่เหมือนกัน ถ้าเป็นชนิด 10 Bit จะมีความละเอียด 0-1023 ดังนั้นการจะนำอุปกรณ์ไปใช้จะต้องทำการปรับค่าให้เหมาะสมก่อนซึ่งจะอธิบายในส่วนของบทที่ 3



รูปที่ 2.1 กราฟแสดงค่า PWM ในรูปของดิวตี้ไซเคิล

2.3 การวัดอัตราการไหล

อัตราการไหล (Flow rate) คือ ปริมาณของของไหลที่ไหลผ่านพื้นที่หน้าในช่วงเวลาหนึ่งมีสมการทั่วไปดังนี้สมการที่ (2.2)

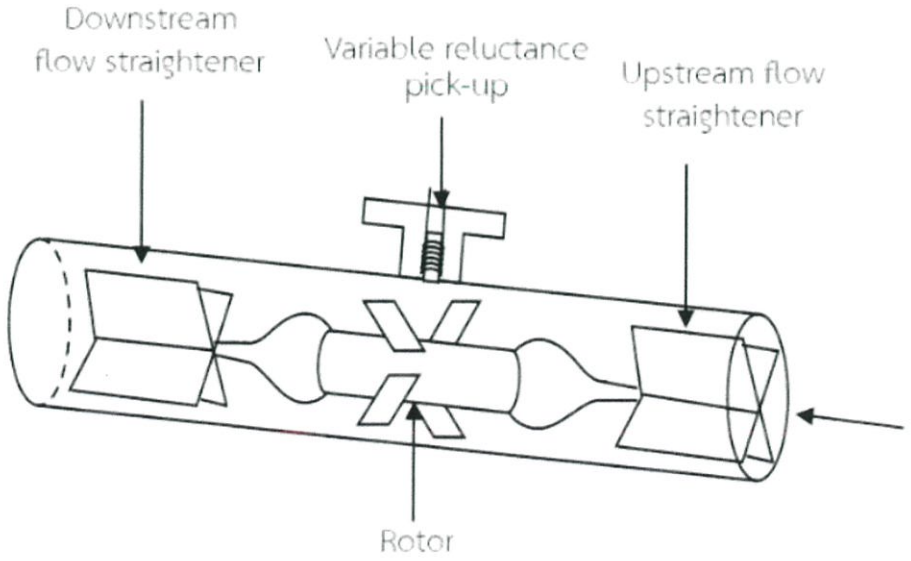
$$Q = AV \tag{2.2}$$

เมื่อ A คือ พื้นที่หน้าตัดการไหล

V คือ ความเร็วเฉลี่ยของการไหล

ในปริญานิพนธ์นี้เครื่องมือวัดอัตราการไหล (Flow sensor) ที่ถูกนำมาใช้เป็นประเภท เทอร์ไบน์ (Turbine) เป็นรูปแบบที่สามารถใช้วัดอัตราการไหลโดยตรง และการทำงานของเทอร์ไบน์นั้นจะเปลี่ยนค่าอัตราการไหลให้อยู่ในรูปแบบของสัญญาณไฟฟ้า โดยใช้ร่วมกับใบพัดที่ทำจากวัสดุที่มีคุณสมบัติเป็นสารแม่เหล็กติดตั้งอยู่บนโรเตอร์ ซึ่งเมื่อใบพัดหมุนจะทำให้เกิดสนามแม่เหล็กอุปกรณ์จะใช้ Hall effect sensor แรงดันไฟฟ้าเหนี่ยวนำที่ได้จากคอล์ยตรวจจับแม่เหล็ก (Magnetic pick-up coil) เป็นขดลวดพันรอบแม่เหล็กถาวรติดตั้งอยู่ภายในตัวเครื่องมือวัดดังรูปที่ 2.2

ตรวจจับสนามแม่เหล็กที่เกิดจากการหมุนของใบพัด และส่งสัญญาณเป็นดิจิตอลด้วยการนับจำนวนพัลส์ที่เกิดขึ้น ไปให้ไมโครโปรเซสเซอร์ประมวลผลเป็นอัตราการไหลซึ่งคำนวณจากอัตราการเกิดพัลส์



รูปที่ 2.2 หลักการทำงานของเครื่องมือวัดอัตราการไหล

เครื่องมือวัดอัตราการไหลชนิดนี้จะวัดได้ค่าแม่นยำเมื่อมีอัตราการไหลที่สูงๆ และความแม่นยำจะลดลงในช่วงอัตราการไหลต่ำๆ สาเหตุเพราะแรงเสียดทานธรรมชาติส่งผลให้การวัดมีความคลาดเคลื่อน เครื่องมือวัดอัตราการไหลชนิดนี้มีย่านการวัด (Range) การไหลของของเหลวอยู่ที่ 1–100,000 ลิตรต่อนาที และย่านการวัดอัตราการไหลของก๊าซอยู่ที่ 5–100,000 ลิตรต่อนาที

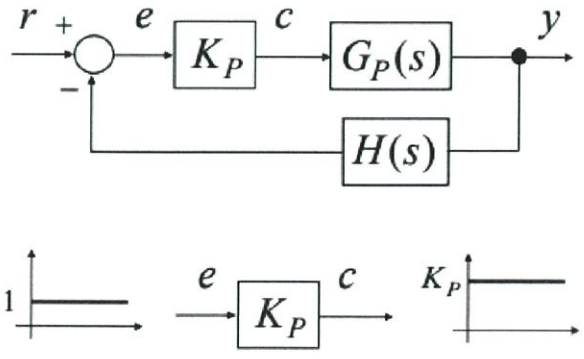
เครื่องมือวัดอัตราการไหลนี้ไม่สามารถใช้วัดของไหลที่มีสารแขวนลอยได้ ในการติดตั้งเครื่องมือวัดอัตราการไหลนี้สามารถติดตั้งได้ทั้งแกนตั้งหรือแกนนอน แต่ต้องให้ห่างจากจุดหักเลี้ยวของท่อพอสมควรทั้งด้านหน้าหลังของเครื่องมือวัด เพื่อไม่ให้ความเร็วของการไหลลดลงจะทำให้การวัดคลาดเคลื่อน (Error)

2.4 ตัวควบคุมแบบสัดส่วน-ปริพันธ์-อนุพันธ์ (Proportional Integral Derivative Control: PID Control)

เป็นการนำตัวควบคุมแบบสัดส่วน ตัวควบคุมแบบปริพันธ์ และตัวควบคุมแบบอนุพันธ์ มาทำงานร่วมกัน หรือเรียกว่าตัวควบคุมแบบพีไอดี เป็นการควบคุมที่ใช้งานได้ง่ายไม่ซับซ้อน และมีการใช้อย่างแพร่หลายในระบบโรงงานอุตสาหกรรม การควบคุมแบบพีไอดีจะมีการควบคุมย่อย 3 ตัว โดยรายละเอียดการทำงานของตัวควบคุมมีดังนี้

2.4.1 ตัวควบคุมแบบสัดส่วน (Proportional control)

ตัวควบคุมแบบสัดส่วนหรือตัวควบคุมแบบตัวพี ตัวควบคุมนี้เป็นการนำสัญญาณค่าความผิดพลาด (e) ระหว่างสัญญาณอ้างอิงกับสัญญาณเอาต์พุตของตัวควบคุมมาเป็นอินพุตของตัวควบคุม แล้วตัวควบคุมจะทำการสร้างสัญญาณเอาต์พุต โดยการขยายสัญญาณดังกล่าวด้วยค่าเกนของตัวควบคุม รูปแบบของบล็อกไดอะแกรมและลักษณะของการประมวลผลสัญญาณจะเป็นดังรูปที่ 2.3

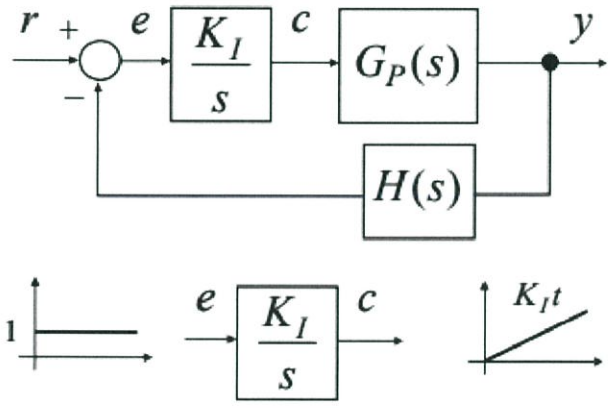


รูปที่ 2.3 ตัวควบคุมแบบพี

การควบคุมแบบนี้เมื่อมีการปรับค่าเกนให้สูงจะทำให้มีผลตอบสนองที่เร็วขึ้น จะเกิดปัญหาหากนำไปใช้งานกับระบบชนิด Type 0 ซึ่งตัวควบคุมแบบพีจะไม่สามารถขจัดค่าความผิดพลาดที่เกิดขึ้นกับระบบออกไปได้ แต่สามารถทำให้ค่าความผิดพลาดนี้มีค่าน้อยลงได้โดยการปรับค่าเกนให้มีค่าสูงขึ้น ทำให้เอาต์พุตของระบบมีค่าที่สูงขึ้นตาม (ระบบมีผลตอบสนองเร็วขึ้น) แต่หากมากเกินไปจะทำให้ระบบเกิดความไม่เสถียร และทำให้ระบบมีค่าพุ่งเกินที่สูง ซึ่งจะเป็นอัตรายต่อระบบ ในทางกลับกันหากค่าเกนมีค่าน้อยเกินไปจะทำให้ระบบนั้นตอบสนองช้า และไม่สามารถสู้ต่อสิ่งรบกวนได้

2.4.2 ตัวควบคุมแบบปริพันธ์ (Integral control)

ตัวควบคุมแบบปริพันธ์ หรือตัวควบคุมแบบไอ ตัวควบคุมนี้เป็นการนำเอาค่าความผิดพลาดของสัญญาณอ้างอิงกับสัญญาณเอาต์พุตมาเป็นอินพุตของตัวควบคุม แล้วตัวควบคุมจะอินทิเกรตสัญญาณความผิดพลาดดังกล่าวแล้วคูณด้วยค่าเกนของตัวควบคุมของบล็อกไดอะแกรม เพื่อที่จะได้สัญญาณเอาต์พุตของตัวควบคุมแบบไอ โดยลักษณะของการประมวลผลสัญญาณเป็นดังรูปที่ 2.4

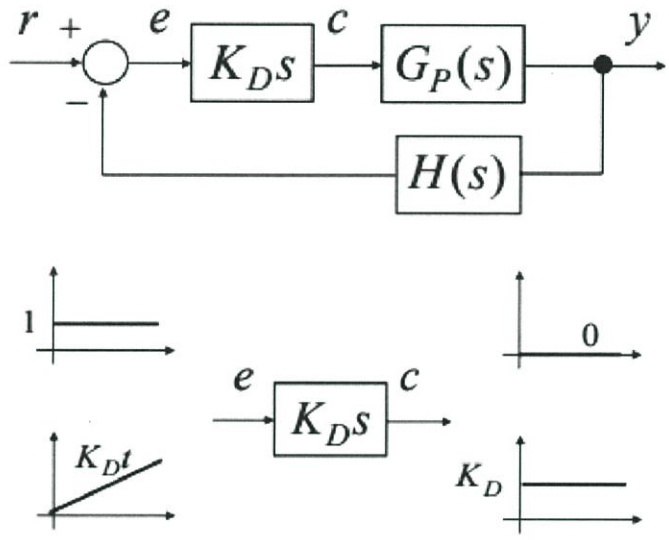


รูปที่ 2.4 ตัวควบคุมแบบไอ

การควบคุมแบบไอหากนำไปใช้งานกับระบบ Type 0 จะสามารถขจัดค่าความผิดพลาดได้ในสภาวะคงตัว แต่ยังมีข้อเสียคือ ตัวควบคุมแบบไอไม่สามารถลดผลของการพุ่งเกินจากระบบได้ และหากเกนมีค่าสูงเกินไปจะทำให้ได้ผลตอบสนองที่ไม่ต้องการ เช่น ระบบเกิดการแกว่ง

2.4.3 ตัวควบคุมแบบอนุพันธ์ (Derivative control)

ตัวควบคุมแบบอนุพันธ์หรือตัวควบคุมแบบดี เป็นการเปรียบเทียบสัญญาณอ้างอิงกับสัญญาณเอาต์พุตเพื่อนำสัญญาณนี้มาเป็นอินพุตของตัวควบคุม โดยตัวควบคุมแบบดีจะทำอนุพันธ์สัญญาณความผิดพลาดแล้วคูณด้วยค่าเกนของตัวควบคุมของบล็อกไดอะแกรม เพื่อที่จะได้สัญญาณเอาต์พุตของตัวควบคุมแบบดี โดยลักษณะของการประมวลผลสัญญาณเป็นดังรูปที่ 2.5

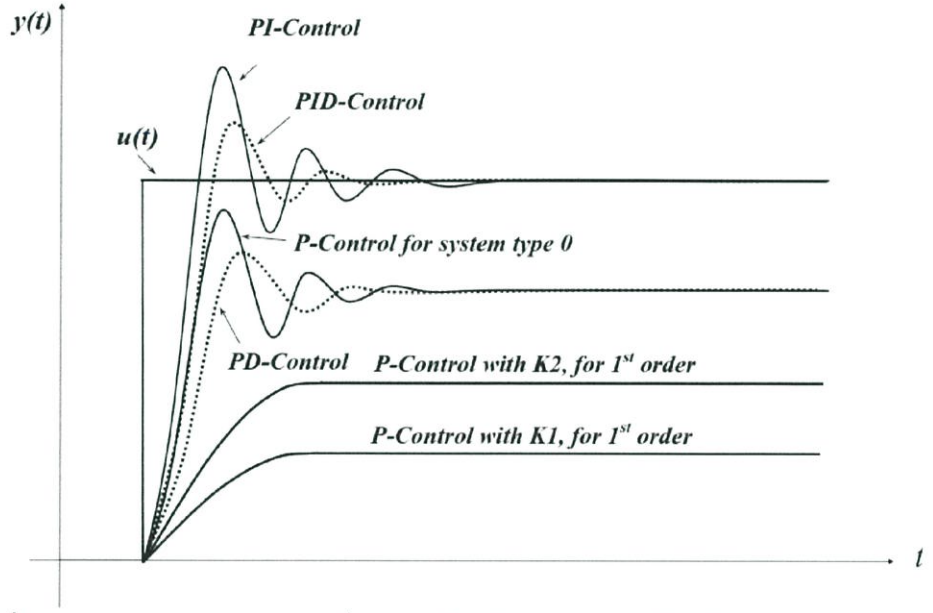


รูปที่ 2.5 ตัวควบคุมแบบดี

ตัวควบคุมแบบพีจะนำไปใช้งานเพื่อลดผลจากการพุ่งเกินของระบบ และลดผลตอบสนองที่มีการเปลี่ยนแปลงไปมา แต่ต้องปรับเกนให้เหมาะสม ไม่เช่นนั้นอาจทำให้เกิดผลตอบสนองที่ไม่พึงประสงค์ ตัวควบคุมแบบดีจะไม่สามารถใช้กับระบบที่มีสัญญาณรบกวนได้ เพราะตัวควบคุมแบบดีจะเป็นตัวขยายสัญญาณรบกวนทำให้ระบบทำงานเพี้ยนได้

จากรูปที่ 2.6 จะเห็นว่าถ้านำตัวควบคุมแบบพีไปใช้กับระบบที่เป็นอันดับหนึ่งผลตอบสนอง ที่ได้จะมีค่าความผิดพลาดในสภาวะคงตัว ซึ่งสามารถลดได้ด้วยการเพิ่มค่าเกนให้สูงขึ้น แต่หากนำไปใช้กับระบบที่มีอันดับสูงขึ้นหรือระบบ Type 0 ค่าความผิดพลาดก็ยังคงมีค่าอยู่ และหากเพิ่มค่าเกนให้สูงขึ้นเพื่อลดค่าความผิดพลาดก็จะทำให้ระบบมีค่าพุ่งเกินสูงขึ้นด้วย ถ้าหากใช้ตัวควบคุมร่วมกันระหว่างพีกับไอ หรือเรียกว่าตัวควบคุมแบบพีไอให้กับระบบนี้แล้ว ตัวควบคุมแบบพีก็จะช่วยลดค่าความผิดพลาดในสถาวะคงตัวได้ แต่ผลตอบสนองยังมีค่าพุ่งเกินเหมือนเดิม ถ้าหากใช้ตัวควบคุมร่วมกันระหว่างพีกับดี หรือเรียกว่าตัวควบคุมแบบพีดี ระบบนี้ค่าพุ่งเกินของผลตอบสนองจะลดลง แต่ค่าความผิดพลาดในสภาวะคงตัวยังคงมีอยู่ ดังนั้นหากใช้ตัวควบคุมร่วมกันระหว่างพีไอและดี หรือเรียกว่า ตัวควบคุมแบบพีไอดี

โดยการปรับค่าเกนให้เหมาะสมกับระบบนั้นๆ ก็จะได้ผลตอบสนองแบบหน่วงต่ำกว่าวิกฤตที่มีค่าพุ่งเกินที่เหมาะสมกับระบบนั้นๆ



รูปที่ 2.6 ผลตอบสนองของระบบที่ควบคุมด้วยตัวควบคุมแบบพีไอดีในรูปแบบต่างๆ

2.5 Message queuing telemetry transport

Message queuing telemetry transport หรือ MQTT คือ โปรโตคอลชนิดหนึ่งที่ถูกออกแบบและพัฒนามาเพื่อการเชื่อมต่อแบบ M2M (Machine-to-machine) สำหรับการเชื่อมต่อระหว่างอุปกรณ์กับอุปกรณ์ให้ทำงานร่วมกันได้ ทั้งยังสนับสนุนเทคโนโลยี IOT ซึ่งก็คือ อินเตอร์เน็ตที่สามารถเชื่อมต่อกับอุปกรณ์ต่างๆ ได้ เช่น ต่อโทรศัพท์มือถือเข้ากับ รถยนต์ ทีวี เครื่องซักผ้า และอุปกรณ์เครื่องใช้ไฟฟ้าภายในบ้านต่างๆ โดยผ่านเครือข่ายไร้สายบนอินเตอร์เน็ต ทำให้สามารถควบคุมอุปกรณ์ต่างๆ จากที่อื่นได้

รูปที่ 2.7 รูปแบบการทำงานของ MQTT

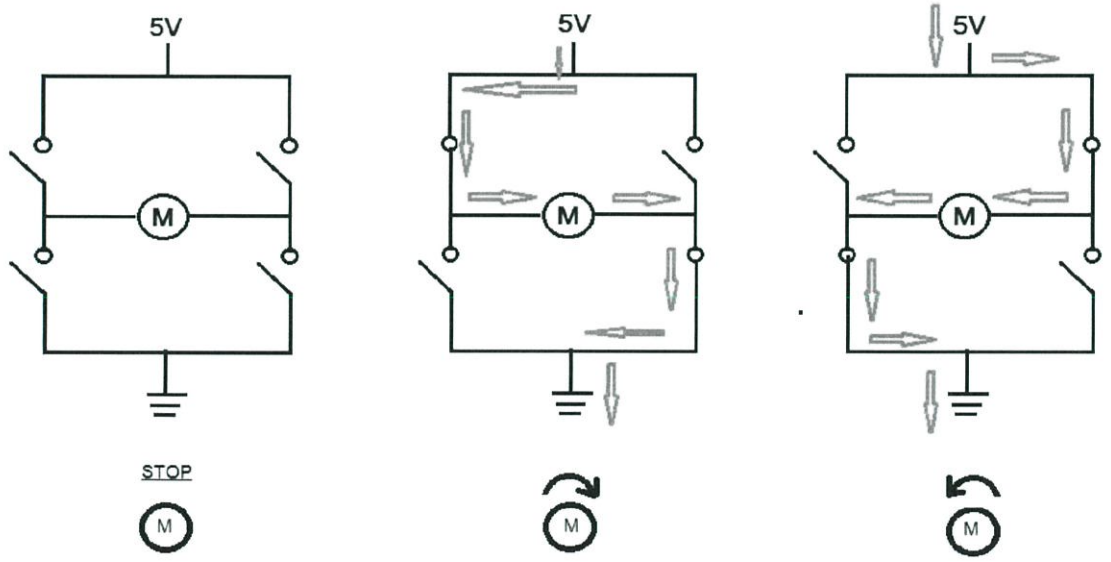
โดย MQTT จะมีตัวกลางที่เรียกว่า โบรกเกอร์ (Broker) ซึ่งจะทำหน้าที่ในการจัดการรับ-ส่งข้อมูลระหว่างอุปกรณ์ โดยตัวโบรกเกอร์จะทำการรับส่งระหว่าง Publisher ซึ่งทำหน้าที่ส่งข้อมูลไปยังหัวเรื่องต่างๆ (Topic) และ Subscriber จะทำหน้าที่ดูแลการเปลี่ยนแปลงของข้อมูลที่อ้างอิงโดยหัวเรื่อง ดังรูปที่ 2.7 จะเห็นได้ว่า หัวเรื่องจะเป็นตัวอ้างอิงหลักของข้อมูลที่จะออกจาก Publisher ไปยังโบรกเกอร์เสมอ ทางด้าน Subscriber ก็จะอ้างถึงหัวเรื่องเพื่อร้องขอข้อมูลที่ต้องการ

2.6 มอเตอร์กระแสตรง

มอเตอร์กระแสตรง (DC motor) จะทำงานโดยวิธีการให้กระแสกับขดลวดที่วางอยู่ในสนามแม่เหล็ก ทำให้เกิดแรงแม่เหล็ก แรงแม่เหล็กที่เกิดขึ้นนี้จะมากหรือน้อยขึ้นอยู่กับปริมาณกระแสไฟฟ้าและพลังของสนามแม่เหล็ก

2.6.1 การขับมอเตอร์กระแสตรง

ในการขับมอเตอร์กระแสตรงต้องใช้วงจรที่เรียกว่า วงจรขับมอเตอร์ (Driver) เป็นวงจรที่ใช้ในการควบคุมการหมุนของมอเตอร์กระแสตรง



รูปที่ 2.8 หลักการควบคุมทิศทางการขับมอเตอร์

โดยหลักการทำงานคือ ใช้วงจร H-bridge แล้วควบคุมการเปิดและปิดของหน้าสัมผัส ซึ่งหน้าสัมผัสอาจจะเป็นทรานซิสเตอร์ (Transistor) หรือรีเลย์ (Relay) ก็ได้ จะทำให้สามารถควบคุมทิศทางการหมุนของมอเตอร์กระแสตรงดังรูปที่ 2.8

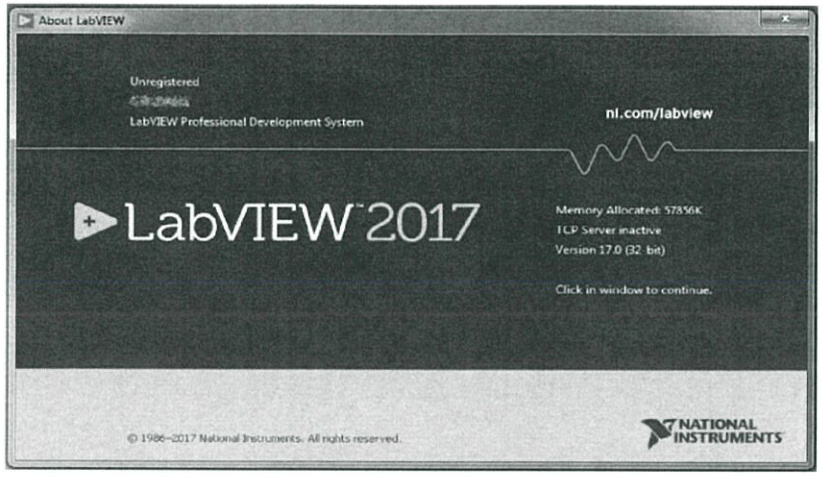
2.6.2 การควบคุมความเร็วมอเตอร์กระแสตรง

การควบคุมความเร็วมอเตอร์กระแสตรงนั้น สามารถทำได้โดยการปรับลดแรงดันไฟฟ้าที่จ่ายให้กับมอเตอร์ หากปรับลดแรงดันไฟฟ้าที่จ่ายให้มอเตอร์จะทำให้ความเร็วรอบของมอเตอร์กระแสตรงลดลง การปรับลดแรงดันที่จ่ายให้มอเตอร์นั้นมีหลากหลายวิธี เช่น ต่อตัวต้านทานปรับค่าได้แบบอนุกรมกับมอเตอร์กระแสตรง การควบคุมการจ่ายแรงดันไฟฟ้าให้กับมอเตอร์กระแสตรง ซึ่งวิธีดังกล่าวจะมีข้อเสีย คือ เมื่อแรงดันไฟฟ้าลดต่ำลงและส่งผลให้กระแสไฟฟ้าลดต่ำลงด้วย ทำให้แรงบิดของมอเตอร์กระแสตรงลดลง ดังนั้นจึงเลือกวิธีควบคุมการจ่ายกระแสไฟฟ้าให้กับมอเตอร์เป็นช่วงๆ โดยแต่ละช่วงค่าเฉลี่ยที่จ่ายให้มอเตอร์กระแสตรง จะทำให้มอเตอร์กระแสตรงมีความเร็วรอบในการหมุนที่ต่างกัน เรียกวิธีการมอดูเลชันความกว้างของกว้างของพัลส์หรือ PWM ซึ่งอธิบายไว้ในหัวข้อที่ 2.2

2.7 โปรแกรม LabVIEW

โปรแกรม LabVIEW ย่อมาจาก Laboratory Virtual Instrument Engineering Workbench เป็นโปรแกรมที่ถูกพัฒนาขึ้นเพื่อใช้เสมือนเป็นเครื่องมือวัด LabVIEW เริ่มต้นโดยบริษัท National Instrument ซึ่งได้ค้นคว้าเพื่อหาวิธีที่จะลดเวลาหรือขั้นตอนในการเขียนโปรแกรม เพื่อใช้งานด้านระบบเครื่องมือวัด

โดยบริษัทมีการพัฒนาโปรแกรมอย่างต่อเนื่องเพื่อให้เขียนโปรแกรมได้สะดวกมากขึ้น จนสามารถเชื่อมต่อกับอุปกรณ์ต่างๆ ได้มากขึ้น พร้อมกับฟังก์ชันการทำงานที่หลากหลายเพื่อให้เหมาะสมกับการใช้งานแต่ละรูปแบบ ดังนั้นจุดประสงค์หลักของการทำงานของ LabVIEW คือ การจัดการในด้านการวัดและเครื่องมือวัดอย่างมีประสิทธิภาพ สิ่งที่ LabVIEW นั้นแตกต่างจากโปรแกรมอื่นๆ คือ LabVIEW เป็นโปรแกรมประเภท GUI (Graphic User Interface) นั้นก็คือ ไม่จำเป็นต้องเขียนโค้ดหรือคำสั่งใดๆ โดยภาษาที่ใช้ในตัว LabVIEW นั้นจะเป็นภาษารูปภาพหรือเรียกอีกอย่างว่าภาษา G (Graphical Language) ซึ่งจะมาแทนการเขียนโปรแกรมที่ทั่วไป เช่น ภาษาซี โดยตัว LabVIEW สามารถที่จะนำข้อมูลจากภายนอกมาทำการวิเคราะห์ ประมวลผล และแสดงค่า พร้อมทั้งมีการเก็บข้อมูล เพื่อสามารถดูข้อมูลได้ย้อนหลัง



รูปที่ 2.9 โปรแกรม LabVIEW

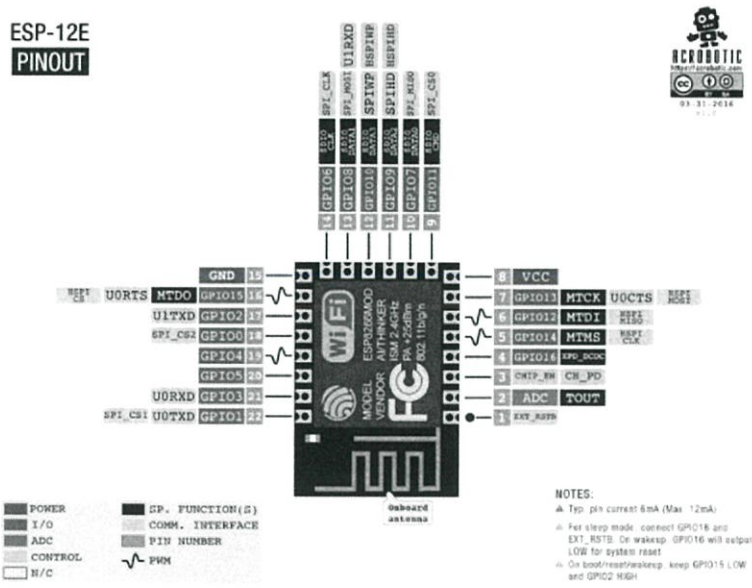
2.8 โมดูล Wi-Fi ESP8266

โมดูล Wi-Fi ESP8266 จะประกอบไปด้วย ไมโครคอนโทรลเลอร์และโมดูล Wi-Fi ดังนั้นจึงสามารถโปรแกรมข้อมูลลงไปได้ ทำให้สามารถนำไปใช้งานแทนไมโครคอนโทรลเลอร์ได้ และมีพื้นที่ประมาณ 4 MB ในการเขียนโปรแกรมลงไปในตัวโมดูล โดยที่ ESP8266 เป็นชื่อของชิปไอซีบนบอร์ดของโมดูล โดย ESP8266 ไม่มีพื้นที่โปรแกรม (Flash memory) ในตัวทำให้ต้องใช้ไอซีจากแหล่งอื่น (External flash memory) ในการเก็บข้อมูลของโปรแกรม โดยเชื่อมต่อผ่านโปรโตคอล SPI จึงทำให้โมดูล ESP8266 มีพื้นที่ในการเก็บโปรแกรมดังรูปที่ 2.10

รูปที่ 2.10 โมดูล Wi-Fi ESP8266-12E

ESP8266 ทำงานที่แรงดันไฟฟ้าอยู่ที่ 3.3V-3.6V หากนำไปใช้งานร่วมกับอุปกรณ์อื่นๆ ที่ใช้แรงดัน 5V จำเป็นต้องใช้วงจรแบ่งแรงดันมาช่วยเพื่อไม่ให้โมดูลพังเสียหาย กระแสที่โมดูลใช้งานสูงสุดคือ 200mA ความถี่คริสตอล 40MHz ทำให้ทำงานได้รวดเร็วตามความถี่

ESP8266-12E โมดูลรุ่นนี้นิยมใช้มากในการทดลองหรือพัฒนา เนื่องจากไม่จำเป็นต้องต่อเสาอากาศเพิ่มขึ้นมา มีความเสถียรมากขึ้น และมีความรวดเร็วในการดำเนินการโปรแกรมดังรูปที่ 2.11

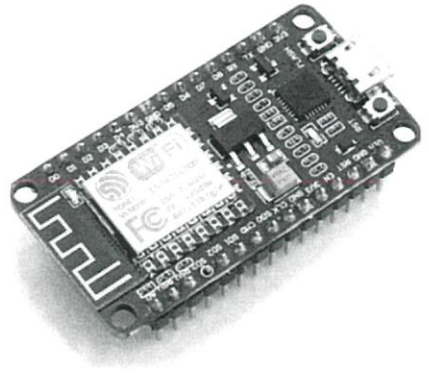


รูปที่ 2.11 รายละเอียด Pinout ของ ESP8266-12E

2.9 NodeMCU

NodeMCU คือ รูปแบบที่มีการพัฒนาจากโปรเจค IOT โดยจะประกอบไปด้วย Development kit (ตัวบอร์ด) และ Firmware (Software บนบอร์ด) ที่เป็น Open source โดยสามารถเขียนโปรแกรมได้ด้วยภาษา Lau ทำให้ง่ายต่อการใช้งาน โดยมีโมดูล Wi-Fi ในตัว ซึ่งเป็นสิ่งสำคัญในการเชื่อมต่อกับอินเตอร์เน็ต โดยโมดูล Wi-Fi จะฝังอยู่ในตัวของ NodeMCU

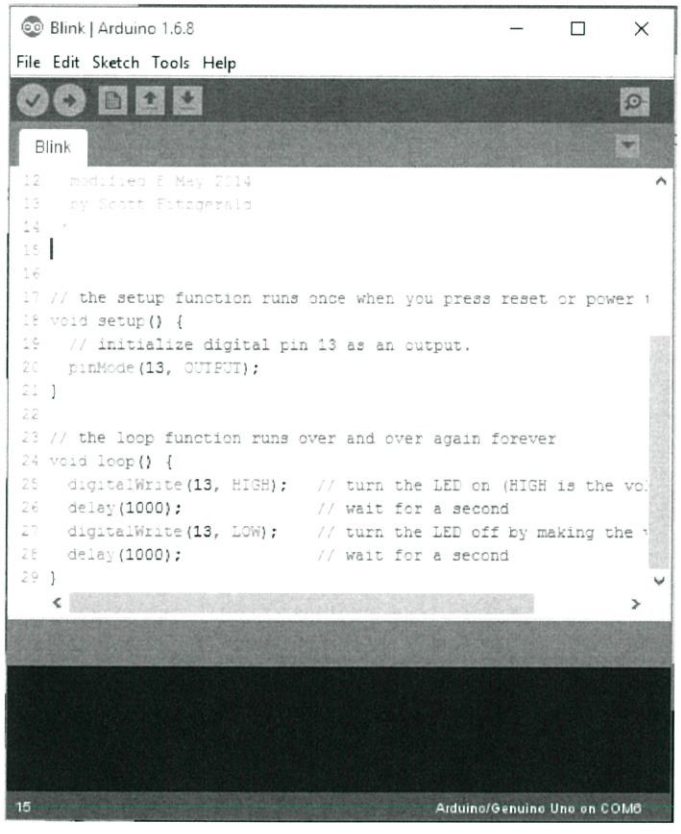
NodeMCU นั้นจะมีลักษณะคล้าย Arduino ตรงที่มีอินพุตเอาต์พุตมาให้ภายในตัว และสามารถเขียนโปรแกรมคอนโทรลอุปกรณ์ในลักษณะ I/O ได้โดยไม่ต้องผ่านตัวกลางหรืออุปกรณ์อื่นๆ ทั้งนี้ยังได้มีการพัฒนา NodeMCU ให้สามารถใช้งานร่วมกับโปรแกรม Arduino IDE จึงสามารถใช้ภาษา C/C++ ได้ในการเขียนโปรแกรม ทำให้การใช้งานหลากหลายมากยิ่งขึ้น โดยเฉพาะในเรื่องของ IOT เช่น การเปิดปิดไฟผ่าน Wi-Fi ดังรูปที่ 2.12



รูปที่ 2.12 NodeMCU ESP8266 V1.0

2.10 Arduino IDE

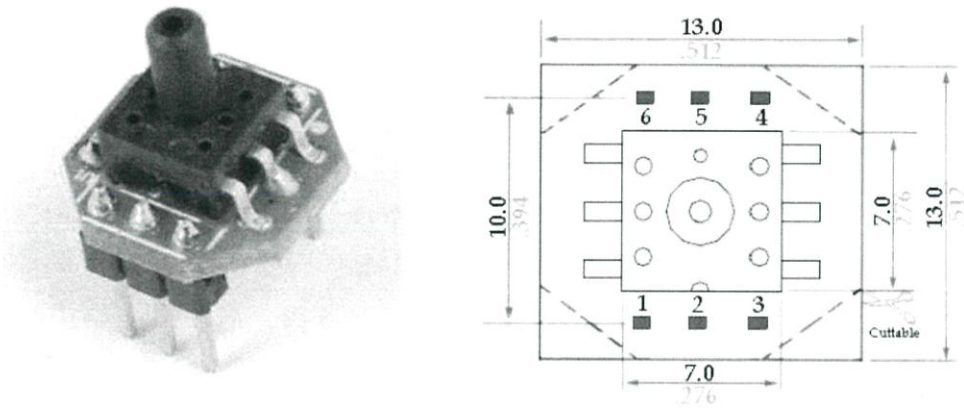
Arduino IDE ย่อมาจาก Arduino integrated development environment ซึ่งก็คือ ซอฟต์แวร์ที่ใช้ในการพัฒนางานสำหรับบอร์ด Arduino หรืออุปกรณ์อื่นๆ ที่รองรับโปรแกรม Arduino IDE โดยที่โปรแกรม Arduino IDE จะใช้ในการเขียนโปรแกรมและคอมไพล์ลงในบอร์ด IDE คือ ส่วนเสริมของระบบการพัฒนาหรือตัวช่วยในเรื่องต่างๆ ที่จะคอยช่วยเหลือคนที่พัฒนา Application ต่างๆ เพื่อให้เกิดความรวดเร็ว ถูกต้อง แม่นยำ ตรวจสอบระบบได้ และง่ายต่อการแก้ไขปรับปรุงดังรูปที่ 2.13



รูปที่ 2.13 หน้าแสดงการทำงานของโปรแกรม Arduino IDE

2.11 โมดูลเซนเซอร์วัดความดัน

XGZP6847 โมดูลเซนเซอร์วัดความดันที่มีการเชื่อมต่อแบบ Ratiometric analog สำหรับอ่านค่าความดันที่มากกว่าช่วงปกติ มีความแม่นยำ ตอบสนองได้ไว ส่วนมากนิยมใช้กับอุปกรณ์การแพทย์ อุปกรณ์ออกกำลังกาย มีประสิทธิภาพ ดังรูปที่ 2.14



รูปที่ 2.14 โมดูล XGZ6847

ตารางที่ 2.1 พารามิเตอร์ประสิทธิภาพของโมดูล XGZ6847

Item	Data	Unit
Output	0.5-4.5	V
Accuracy	±1	% Span
Zero Temp.Coefficient	±0.3	%FS/°C
Long term Stability	±1	% Span
Over Pressure	3x(≤500kPa)	Rated
	1.5x(≥500kPk)	
Compensation	-20 ~85/-4~176	°C/°F
Operating Temp	-20 ~100/-4~212	°C/°F
Storage Temp	-40 ~125/-40~257	°C/°F

ตารางที่ 2.2 ตำแหน่งขาของโมดูล

1	2	3	4	5	6
N/C	Vdd	GND	Vdd	OUT	GND

2.11.1หลักการทำงานของสเตรนเกจ

สเตรนเกจจะมีขดลวดขดไปมาวางอยู่บนวัสดุที่เป็นฉนวนหรือวัสดุกึ่งตัวนำ เมื่อได้รับแรงกดจะทำให้ค่าความต้านทานมีการเปลี่ยนแปลง

สเตรนเกจแบ่งได้เป็น 2 แบบคือ 1. แบบยึดติด (Bonded strain gauge) 2. แบบไม่ยึดติด (Unbonded strain gauge) ทั้งสองแบบมีหลักการทำงานคล้ายกันคือ นำขดลวดเส้นเล็กๆ ไปติดกับวัสดุที่ต้องการตรวจวัดความเครียดจะมีความสัมพันธ์กับสมการที่ (2.4)

$$R = \frac{pL}{A} \tag{2.3}$$

เมื่อ

R	คือ ความต้านทานของสเตรนเกจ	หน่วยโอห์ม
p	คือ ค่าความต้านทานคงที่ของขดลวด	หน่วยโอห์มเมตร
L	คือ ความยาวของเส้นลวด	หน่วยเมตร
A	คือ พื้นที่หน้าตัดของตัวนำ	หน่วยตารางเมตร

การตรวจวัดความเครียดของวัสดุจะพิจารณา เกจแฟกเตอร์ (Gauge factor) คือ ค่าความต้านทานของเกจที่เปลี่ยนแปลงส่วนค่าความยาวที่เปลี่ยนแปลง

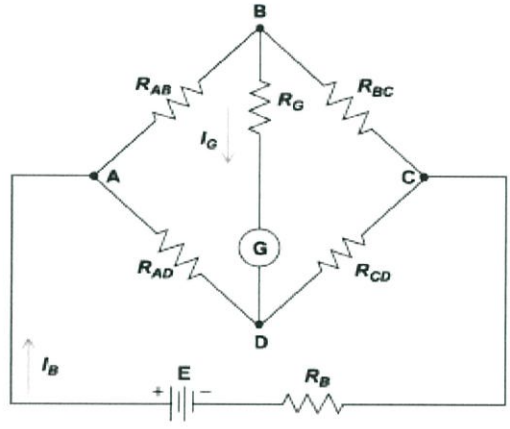
$$K = \frac{\Delta R/R}{\Delta L/L} \tag{2.4}$$

เมื่อ

K	คือ เกจเฟกเตอร์	
R	คือ ความต้านทานตั้งต้น	หน่วยโอมห์ม
ΔR	คือ ความต้านทานที่เปลี่ยนแปลง	หน่วยโอมห์ม
L	คือ ความยาวตั้งต้น	หน่วยเมตร
ΔL	คือ ความยาวที่เปลี่ยนแปลง	หน่วยเมตร

2.11.2 วงจรบริดจ์แบบวีทสโตน

ในเซนเซอร์ที่ให้ค่าออกมาเป็นค่าความต้านทานไฟฟ้า จะนิยมเอาวงจรบริดจ์แบบวีทสโตนมาใช้ร่วมกันในการแปลงสัญญาณที่ได้ไปเป็นสัญญาณทางไฟฟ้า จะแสดงวงจรวีทสโตนบริดจ์ในรูปที่ 2.15



รูปที่ 2.15 วงจรวีทสโตนบริดจ์

บทที่ 3

ขั้นตอนการดำเนินงาน

ในการทำโครงงานการควบคุมระดับน้ำแบบไร้สายและการมอนิเตอร์นั้น ได้วางแผนขั้นตอนการดำเนินงานออกเป็นหัวข้อต่างๆ โดยเรียงลำดับความสำคัญและความต่อเนื่องของการทำงานออกเป็นขั้นตอนดังนี้

1. การเลือกอุปกรณ์
2. การทดสอบการใช้งานอุปกรณ์
3. การทดสอบการเชื่อมต่อ Server
4. การเขียน Function block ของโปรแกรม LabVIEW เพื่อสร้าง Server
5. การรวมโค้ด
6. การออกแบบวงจรตู้ควบคุม

3.1 การเลือกอุปกรณ์

ขั้นตอนในการเลือกอุปกรณ์เป็นขั้นตอนแรกของการทำงาน โดยการเลือกใช้อุปกรณ์นั้นเลือกที่จะใช้โมดูลเซนเซอร์ควบคู่กับอุปกรณ์อื่นๆ ที่เกี่ยวข้องกับการใช้งานที่เหมาะสมโดยอุปกรณ์ที่เลือกมีดังนี้

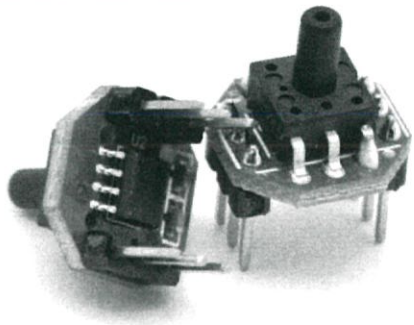
3.1.1 เซนเซอร์วัดระดับน้ำ

สำหรับโครงงานนี้ได้เลือกเซนเซอร์วัดความดันน้ำ XGZP6847 pressure sensor module เนื่องจากค่าแรงดันสูงสุดของถังน้ำที่วัดได้ มีค่าสูงสุดที่ 4.369 kpa ดังนั้นจึงเลือกเซนเซอร์รุ่นนี้ ซึ่งมีช่วงของการวัดค่าแรงดันอยู่ที่ 0-10 kpa จะมีความใกล้เคียงกับค่าจริงที่เกิดขึ้น และทำให้ค่าจากโปรแกรมที่ NodeMCU อ่านได้มีค่าละเอียด ดังรูปที่ 3.1

คุณสมบัติของเซนเซอร์วัดความดัน

- Ranges: 0-10 kpa
- MEMs technology
- Gauge
- DIP package
- For non-corrosive gas or liquid

- Amplified and calibrated.
- Temp. compensated: 0°C~+85°C (32°F~+185°F)
- Directly application, Low cost



รูปที่ 3.1 โมดูลวัดความดัน XGZP6847

3.1.2 เซนเซอร์วัดอัตราการไหลของน้ำ

สำหรับโครงงานนี้ได้เลือกเซนเซอร์วัดอัตรการไหลของน้ำ (Electronic flow sensor electronic flow meter) 1-30 ลิตรต่อนาที โดยใช้กับท่อขนาด ½ นิ้ว ดังรูปที่ 3.2

คุณสมบัติของเซนเซอร์วัดความดัน

- The level of the test pulse frequency (Hz) = [8.1Q – 3] ±10 (Q = flow rate L/min.)
- Minimum rated working voltage: DC 3.5, 5–24V
- Maximum current: 15 mA DC 5V
- Voltage range: DC 5-18V
- Operating temp: ≤ 80°C
- Allow compression: water pressure 1.75 mpa below

รูปที่ 3.2 Electronic flow sensor

3.1.3 ชุดขับมอเตอร์ BTS7960 43A

คุณสมบัติของชุดขับมอเตอร์

- Operating voltage 5.5 to 27V (B+)
- Path resistance of typ. 16mΩ at 25°C
- Low quiescent current of typ. 7μA at 25°C
- PWM capability of up to 25kHz combined with active freewheeling
- Switched mode current limitation for reduced power dissipation in overcurrent
- Current limitation level of 43A typ.
- Status flag diagnosis with current sense capability
- Over temperature shut down with latch behavior
- Under voltage shut down
- Driver circuit with logic level inputs
- Adjustable slew rates for optimized EMI
- 74AHC244 Schmitt-trigger octal buffer/line driver for ESD protection(inputs accepts voltages higher than VCC)

4.3.6 Electrical Characteristics - Protection Functions

$-40\ °C < T_j < 150\ °C$; $8\ V < V_S < 18\ V$ (unless otherwise specified)

Pos.	Parameter	Symbol	Limit Values			Unit	Test Conditions
			min.	typ.	max.		
Under Voltage Shut Down							
4.3.1	Switch-ON voltage	$V_{UV(ON)}$	–	–	5.5	V	V_S increasing
4.3.2	Switch-OFF voltage	$V_{UV(OFF)}$	4.0	–	5.4	V	V_S decreasing
4.3.3	ON/OFF hysteresis	$V_{UV(HY)}$	–	0.2	–	V	–
Over Voltage Lock Out							
4.3.4	Switch-ON voltage	$V_{OV(ON)}$	27.5	–	–	V	V_S decreasing
4.3.5	Switch-OFF voltage	$V_{OV(OFF)}$	27.6	–	30	V	V_S increasing
4.3.6	ON/OFF hysteresis	$V_{OV(HY)}$	–	0.2	–	V	–

รูปที่ 3.3 Datasheet ของชุดขับมอเตอร์ BTS7960 43A

โมดูลขับมอเตอร์เป็นรูปแบบ Full bridge ใช้ไอซีเบอร์ BTS7960 เพื่อใช้สำหรับขับมอเตอร์กระแสตรงที่ต้องการค่าดีซีสูง ใช้สัญญาณพัลส์ในการควบคุมความเร็ว สามารถควบคุมความเร็วให้หมุนทวนเข็มหรือตามเข็มได้ และรองรับความเร็วของพัลส์ได้ที่ 25kHz โดยมีแรงดันไฟเลี้ยงมอเตอร์ 6-27VDC กระแสเอาต์พุตสูงสุดจะอยู่ที่ 40A แต่ในทางปฏิบัติจริงไม่ควรใช้กระแสเกิน 20A เพื่อความปลอดภัยแรงดันอินพุต (PWM) สำหรับใช้ควบคุมควรเป็น 5VDC โดยไอซี BTS7960 จะมีระบบป้องกันดังต่อไปนี้

- Under voltage shutdown ถ้าแหล่งจ่ายไฟให้กับมอเตอร์ต่ำกว่า 5.5V ไอซีจะหยุดการทำงานจนกระทั่งแรงดันจะเพิ่มขึ้นจนเป็น 5.5V ขึ้นไป
- Over temperature protection ถ้าอุณหภูมิภายในตัวไอซีสูงเกินค่าที่ถูกกำหนดไว้ไอซีจะหยุดทำงานทันที
- Current limitation มีระบบป้องกันกระแสเกิน 43A

การต่อใช้งานทางด้านเอาต์พุต

- ขา B+ : ขั้วบวกแหล่งจ่ายไฟสำหรับมอเตอร์
- ขา B - : กราวด์ของแหล่งจ่ายไฟสำหรับมอเตอร์
- ขา M+ : ขั้วบวกของมอเตอร์
- ขา M - : ขั้วลบของมอเตอร์

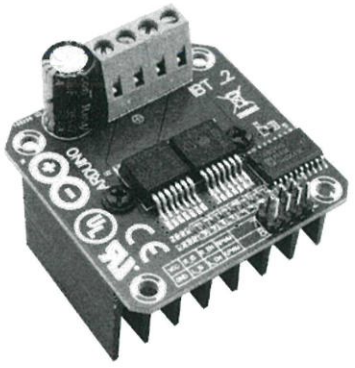
การต่อใช้งานทางด้านอินพุต (ขาควบคุม)

- VCC : +5V
- GND : GND
- R_IS และ L_IS เป็นขาเอาต์พุตแสดงสถานะที่ผิดพลาด (Error signal) กรณีที่กระแสทางเอาต์พุตไหลเกินหรือเกิดการลัดวงจร และตัว IC จะหยุดทำงานเสมอ
- R_EN และ L_EN จะเป็นขาควบคุม Enable (เปิดปิดการทำงานของเอาต์พุตทางขวาและทางซ้ายตามลำดับ) : Active high (ต่อ 5V)
- RPWM และ LPWM เป็นขาอินพุตสำหรับต่อสัญญาณพัลส์เพื่อมาควบคุมความเร็วของมอเตอร์

หลักการทำงาน

- จำเป็นต้องต่อขา R_EN และ L_EN ด้วย 5V ไว้ (เป็นการ Enable เอาต์พุต)
- ถ้าต้องการให้หมุนไปทางซ้ายจ่ายพัลส์ (หรือจ่าย 5V) ไปที่ขา LPWM โดยที่ขา RPWM ให้ต่อลงกราวด์ (หรือจ่าย 0V)

- ถ้าต้องการให้หมุนไปทางขวาจ่ายพัลส์ (หรือจ่าย 5V) ไปที่ขา RPWM โดยที่ขา LPWM ให้ต่อลงกราวด์ (หรือจ่าย 0V)
- ถ้าจ่าย +5V ไปพร้อมกันที่ขา LPWM และ RPWM มอเตอร์จะหยุดหมุน



รูปที่ 3.4 High-power motor drive BTS7960 43A

3.1.4 ปั๊มน้ำจุ่ม/แช่

รูปที่ 3.5 ปั๊มน้ำแบบจุ่ม/แช่ DC 12V รุ่น BL-2512SI

คุณสมบัติของปั๊มน้ำ DC 12V รุ่น BL-2512SI

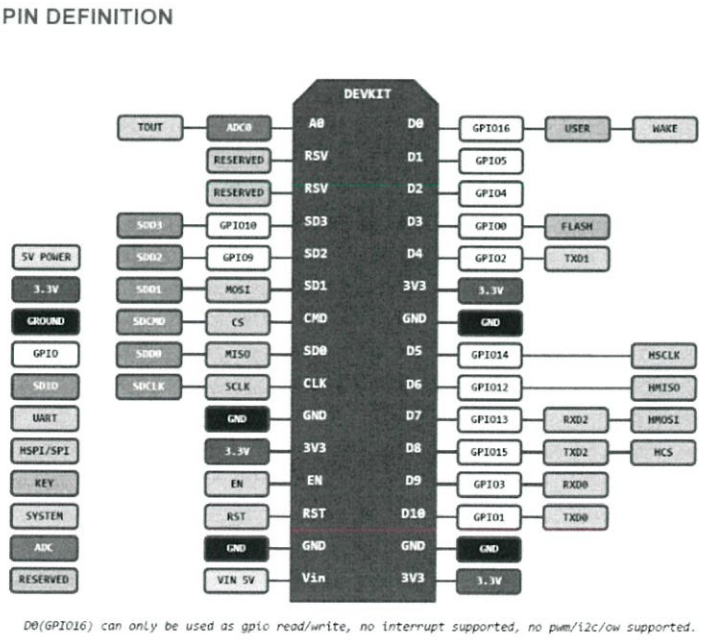
- Power: 12VDC
- Current: 5.4A
- Outlet dia: 1 inch (25mm)
- Delivery head: 4 m
- Delivery volume: 70 L/min, 4200 L/hr
- Motor power: 45W/5400/rpm

3.1.5 NodeMCU ESP8266

เลือกใช้ NodeMCU/ESP8266 เพราะสามารถเขียนโปรแกรมที่ใช้งานได้ง่าย โดยใช้โปรแกรม Arduino ในการเขียนโค้ดคำสั่งการทำงาน จึงทำให้มีแหล่งข้อมูลให้ศึกษาค้นคว้าเพิ่มเติมในอินเตอร์เน็ต และบอร์ด NodeMCU/ESP8266 เป็นบอร์ดไมโครคอนโทรลเลอร์ที่มีจำนวนขาพอร์ตอินพุตและเอาต์พุตมากพอสำหรับการนำไปใช้งานจริง สามารถต่อกับเซนเซอร์ได้ทั้งแบบดิจิตอลและแอนาล็อก ทั้งยังต่อเพื่อขับอุปกรณ์เอาต์พุตให้ทำงาน โดยที่จะต้องเขียนโปรแกรมเพื่อสั่งงานให้บอร์ด NodeMCU/ESP8266 สามารถควบคุมอุปกรณ์ต่างๆ ได้ เนื่องจากมีโมดูล Wi-Fi ในตัว จึงสามารถเชื่อมต่อเพื่อรับส่งข้อมูลผ่านทางอินเตอร์เน็ตได้ นอกจากนั้นยังมีราคาถูก เพื่อใช้ในการพัฒนาอุปกรณ์ IOT

คุณสมบัติของ NodeMCU ESP8266

- Based on โมดูล Wi-Fi ที่ชื่อ ESP8266
- มี GPIO, PWM, I2C, 1-Wire และ ADC รวมอยู่ในบอร์ดเดียว
- มี USB-TTL มาในตัว ใช้ IC CP2102 ของบริษัท Slabs
- มี PCB antenna สำหรับรับ-ส่ง สัญญาณแบบไร้สาย
- ใช้ Micro-USB สำหรับจ่ายไฟเลี้ยงและดาวน์โหลดเฟิร์มแวร์
- มีขา A0 รับอินพุตแรงดันแบบ Analog (ADC ขนาด 10 บิต) ผ่านวงจรแบ่งแรงดันด้วยตัวต้านทาน 100k/220k (ลดแรงดันอินพุตจาก 0V–3.3V ลงมาให้อยู่ในช่วง 0V-1V)
- มีวงจรควบคุมแรงดัน 3.3V บนบอร์ด



รูปที่ 3.6 Pinout ของ NodeMCU ESP8266

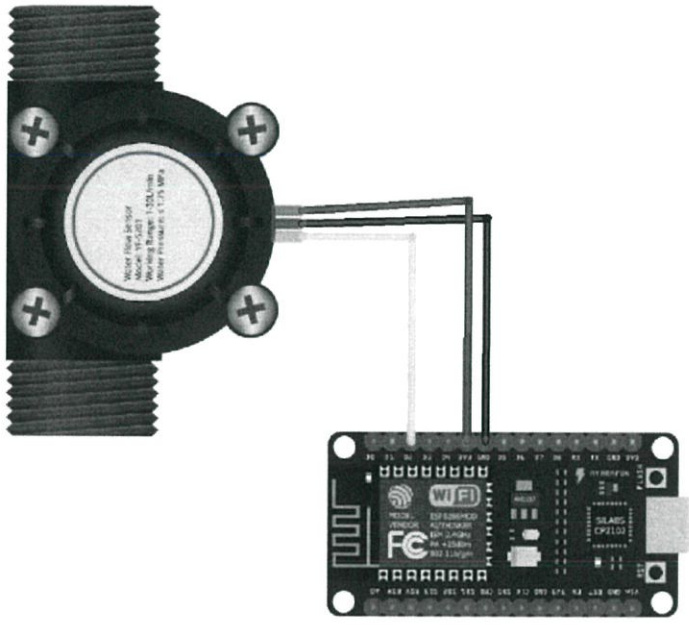
3.2 การทดสอบการใช้งานของอุปกรณ์

3.2.1 โค้ดทดสอบการใช้งาน Flow sensor กับ NodeMCU ESP8266

```
volatile int NbTopsFan;
int Calc;
int hallsensor = 4;
int outputValue;
void rpm()
{
  NbTopsFan++;

 }

void setup()
{
  // put your setup code here, to run once:
  pinMode(hallsensor,INPUT);
  Serial.begin(115200);
 attachInterrupt(hallsensor,rpm,RISING);
}

void loop()
{

   NbTopsFan = 0; //Set NbTopFan to 0 ready for calculate
   sei();  //Enables interrupt
   delay(200); // wait 1 second
   cli(); // Disable interrupt
   Calc = (NbTopsFan*1.2307); //(Pulse frequency x 60)/7.5Q = flow rate in L/hr
   outputValue = map(Calc,0,1,0,100);
   Serial.print("flowrate = ");
   Serial.print(Calc,DEC); // Prints the number calculate
   Serial.print("L/min"); //Print L/hour and returns to new line
   Serial.print("\t%output = ");
   Serial.print(outputValue);
   Serial.print("\tPULSE = ");
   Serial.println(NbTopsFan);

  }
```

1. เลือกขา DI2 (GPIO4) ซึ่งรองรับการ Interrupt (สามารถดู Pinmap ที่รองรับการ Interrupt จาก Datasheet ของอุปกรณ์ ซึ่งตัวควบคุมแต่ละตัวจะไม่เหมือนกัน)

2. การคำนวณค่าออกมาเป็น L/min (ลิตรต่อนาที) ต้องหารค่าจากเซนเซอร์ด้วย Calibrate factor ซึ่งปกติจะเป็น 7.5 แต่ตัวอุปกรณ์จะคำนวณทุกๆ 1 วินาที หากต้องการให้เซนเซอร์งานเร็วขึ้น ต้องเปลี่ยนค่า Factor ใหม่



รูปที่ 3.7 การต่อ Flow sensor กับ NodeMCU

COM5

```
flowrate = 8L/min       %output = 50    PULSE = 61
flowrate = 8L/min       %output = 50    PULSE = 61
flowrate = 8L/min       %output = 50    PULSE = 61
flowrate = 8L/min       %output = 50    PULSE = 62
flowrate = 8L/min       %output = 50    PULSE = 61
flowrate = 8L/min       %output = 50    PULSE = 61
flowrate = 8L/min       %output = 50    PULSE = 61
flowrate = 8L/min       %output = 50    PULSE = 61
flowrate = 8L/min       %output = 50    PULSE = 61
flowrate = 9L/min       %output = 56    PULSE = 69
flowrate = 10L/min      %output = 62    PULSE = 82
flowrate = 13L/min      %output = 81    PULSE = 99
flowrate = 15L/min      %output = 93    PULSE = 113
flowrate = 16L/min      %output = 100   PULSE = 122
flowrate = 16L/min      %output = 100   PULSE = 122
flowrate = 16L/min      %output = 100   PULSE = 122
flowrate = 16L/min      %output = 100   PULSE = 122
flowrate = 16L/min      %output = 100   PULSE = 123
flowrate = 16L/min      %output = 100   PULSE = 123
flowrate = 16L/min      %output = 100   PULSE = 122
```

รูปที่ 3.8 ผลการทำงานของ Flow sensor

3.2.2 โค้ดทดสอบการใช้งาน Pressure sensor กับ NodeMCU ESP8266

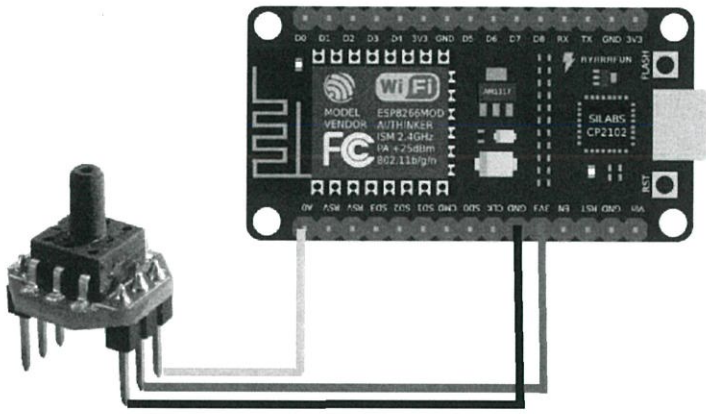
```
#include <Ticker.h>
Ticker flipper;
const uint32_t ANALOG_PIN = A0;
int sensorValue = 0;
int OutputValue = 0;
void readValue()
  {
     sensorValue = analogRead(ANALOG_PIN);
     OutputValue = map(sensorValue,288,1015,0,100);

   }

void setup()
{  Serial.begin(115200);
  //flipper.attach(0.001,*****);
   flipper.attach_ms(1, readValue);
}

void loop()
{     Serial.print("\tsensor = ");
      Serial.print(sensorValue);
      Serial.print("\toutput = ");
      Serial.println(OutputValue);
      delay(100);
}
```

1

2

1. ค่าที่จะนำมา map จะต้องวัดค่า Analog ของเซนเซอร์กับระดับน้ำก้นถังและปากถัง เพื่อเทียบเป็นเปอร์เซ็นต์ 0-100% ซึ่งค่าอาจจะไม่เท่าเดิมทุกครั้ง ดังนั้นควรตรวจเช็คก่อนทำงาน

2. ฟังก์ชัน Ticker สามารถกำหนดความเร็วในการ Interrupt ข้อมูลจากเซนเซอร์ได้ โดยถ้าใช้ .attach_ms จะกำหนดเวลาเป็นมิลลิวินาที ถ้าใช้ .attach จะกำหนดเป็นวินาที



รูปที่ 3.9 การต่อ Pressure sensor กับ NodeMCU

COM6 (Arduino/Genuino Uno)

```
sensor = 437    output = 29     output2 = 2263
sensor = 436    output = 29     output2 = 2263
sensor = 435    output = 29     output2 = 2263
sensor = 436    output = 29     output2 = 2263
sensor = 436    output = 29     output2 = 2263
sensor = 435    output = 29     output2 = 2263
sensor = 436    output = 29     output2 = 2263
sensor = 435    output = 29     output2 = 2263
sensor = 436    output = 29     output2 = 2263
sensor = 436    output = 29     output2 = 2263
sensor = 436    output = 29     output2 = 2263
sensor = 437    output = 29     output2 = 2263
sensor = 437    output = 29     output2 = 2263
sensor = 437    output = 29     output2 = 2263
sensor = 437    output = 29     output2 = 2263
sensor = 437    output = 29     output2 = 2263
sensor = 436    output = 29     output2 = 2263
sensor = 436    output = 29     output2 = 2263
sensor = 435    output = 29     output2 = 2263
sensor = 437    output = 29     output2 = 2263
```

รูปที่ 3.10 ผลการทำงาน Pressure sensor

3.2.3 โค้ดทดสอบการใช้งานตัวขับมอเตอร์และปั๊มกับ NodeMCU ESP8266

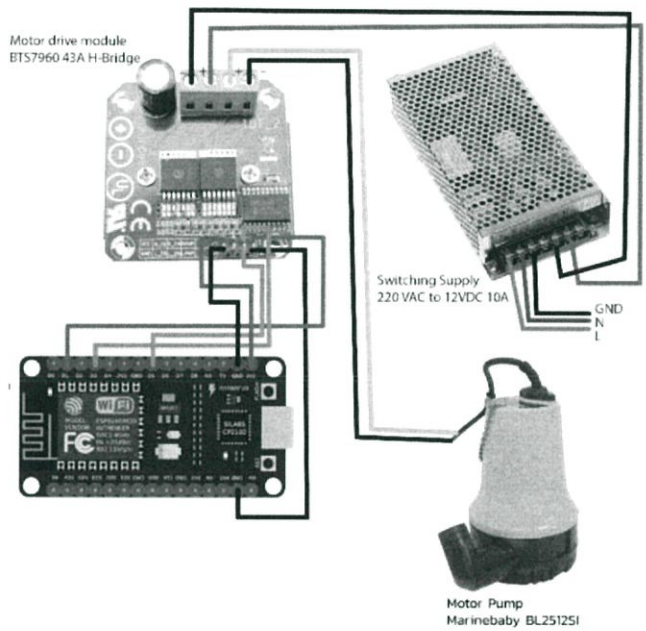
```
int RPWM = 5;
float motorSpeedPWM;
int motorSpeedDUT;
float Speed;
char rec;
float mapf(float x, float in_min, float in_max, float out_min, float out_max)
  {
     float result;
     result = (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
     return result;
   }


void setup()
{
  pinMode(RPWM,OUTPUT);
  Serial.begin(115200);

}

void loop()
{
    while(Serial.available())
  {
      rec = Serial.read();

    if(rec == 'a')  motorSpeedDUT +=1;
    else if(rec == 's')  motorSpeedDUT +=10;
    else if (rec == 'd')  motorSpeedDUT -=1;
    else if (rec =='f')  motorSpeedDUT -=10;

      if (motorSpeedDUT > 0)
      {
      digitalWrite(RPWM,HIGH);
      motorSpeedPWM = mapf(motorSpeedDUT, 0, 100, 0, 1020);
      analogWrite(RPWM,motorSpeedPWM);
      Serial.print("Motor Start");
      Serial.print("\t");
      }

   if(motorSpeedDUT == 0)
  {
    digitalWrite(RPWM,LOW);
    Serial.print("Motor Stop");
    Serial.print("\t");
    motorSpeedDUT = 0;
    motorSpeedPWM = 0;
  }

  Serial.print("Duty Circle = ");
  Serial.print(motorSpeedDUT);
  Serial.print("\t");
  Serial.print("Motor Speed = ");
  Serial.println(motorSpeedPWM);
  Serial.flush();
  delay(150);
  }
```

1

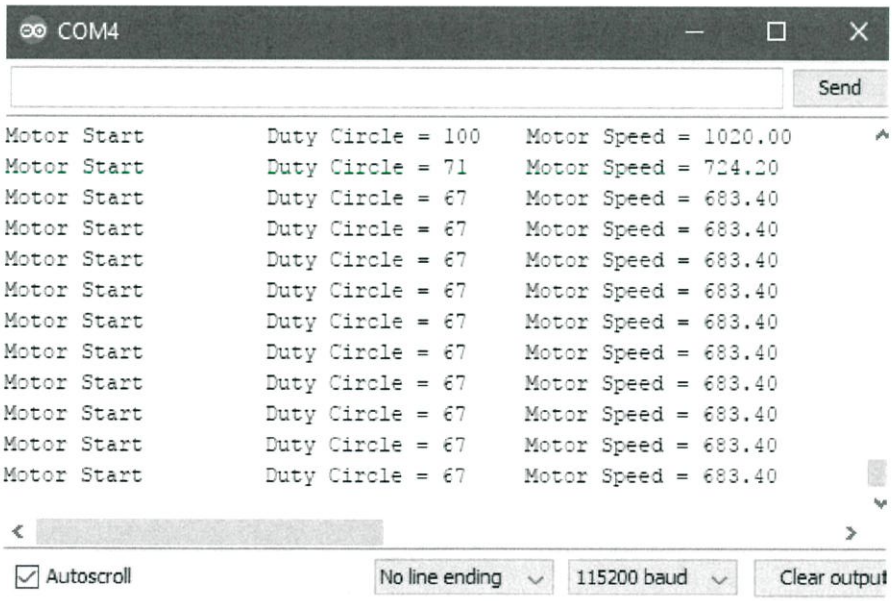
2

3

1. ปกติการ map จะได้ค่าเป็นจำนวนเต็ม ฟังก์ชัน mapf จะทำให้การ map ค่าเป็นทศนิยม
2. ทดลองรับค่าจาก Serial เพื่อดูระดับการทำงาน
3. เนื่องจาก NodeMCU มีการสั่งการแบบ 10 bit (2^{10}) ดังนั้นต้อง map ค่า 0-100% กับ 0-1020



รูปที่ 3.11 การต่อตัวขับมอเตอร์และปั๊มกับ NodeMCU



รูปที่ 3.12 ผลการทำงานของมอเตอร์

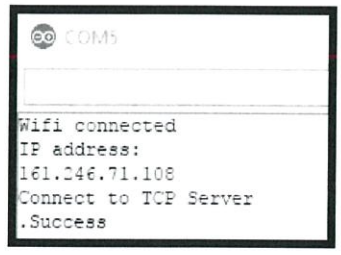
3.3 การทดสอบการเชื่อมต่อ Server

3.3.1 โค้ดทดสอบการเชื่อมต่อ Server (LabVIEW) กับ NodeMCU ESP8266

```
#include <ESP8266WiFi.h>
#define SERVER_PORT 80  //ค่า port ที่ต้องการเชื่อมต่อ
String  m_respond = "";
IPAddress server_ip = {161,246,71,102};
const char* ssid = "[redacted]"; //ค่า ssid
const char* password = "[redacted]"; // ค่า password
WiFiServer server(SERVER_PORT); //สร้าง object และกำหนด port ที่ต้องการเชื่อมต่อกับ server
WiFiClient client;              // สร้าง object client


void setup()
{
  Serial.begin(115200); //เปิดใช้ serial
  WiFi.begin(ssid,password); // เชื่อมต่อกับ AP
  while (WiFi.status() != WL_CONNECTED) // รอการเชื่อมต่อ
  {
    delay(500);
    Serial.print(".");
  }
    Serial.println("");
    Serial.println("Wifi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP()); // แสดงหมายเลข IP
    Serial.println("Connect to TCP Server");
    while (!client.connect(server_ip,SERVER_PORT)) //เชื่อมต่อกับ Server
    {
      Serial.print(".");
      delay(100);
     }
    Serial.println("Success");

}
```

1. โค้ดในส่วนของ IP Address เป็นเลข IP ของคอมพิวเตอร์ที่ LabVIEW เป็น Server รอรับค่าอยู่ ส่วน SSID และ Password เป็นค่าที่ได้จากเราท์เตอร์

2. เมื่อเชื่อมต่อกันแล้ว ที่หน้า Serial จะขึ้นดังรูปที่ 3.13 โดย IP ที่แสดงดังรูปเป็นของ NodeMCU ที่เราท์เตอร์แจกให้



รูปที่ 3.13 ผลการเชื่อมต่อกับ Server

ในส่วนของการรับ – ส่งข้อมูลจาก Server (LabVIEW) ของ NodeMCU ESP8266

```
34  void loop()
35  {
36    m_respond = "";
37    while (client.available()) //ตรวจเช็คว่ามี Data ส่งมาจาก Server หรือไม่
38    {
39      char c = client.read();   //ใช้ char รอรับค่าจากการ read
40       m_respond += c;          //เปลี่ยนเป็น String
41    }
42
43    Serial.println(m_respond);
44    client.println([redacted]);
45    delay(100);
46  }
```

3

4

3. การรับข้อมูลผ่าน TCP/IP จะเป็นแบบข้อมูลแบบ String เท่านั้น แต่เพื่อความเร็วในการรับค่า จึงใช้วิธีรับค่าทีละตัวอักษร แล้วนำมารวมเป็นข้อมูลใหม่ โดยใช้คำสั่ง client.read

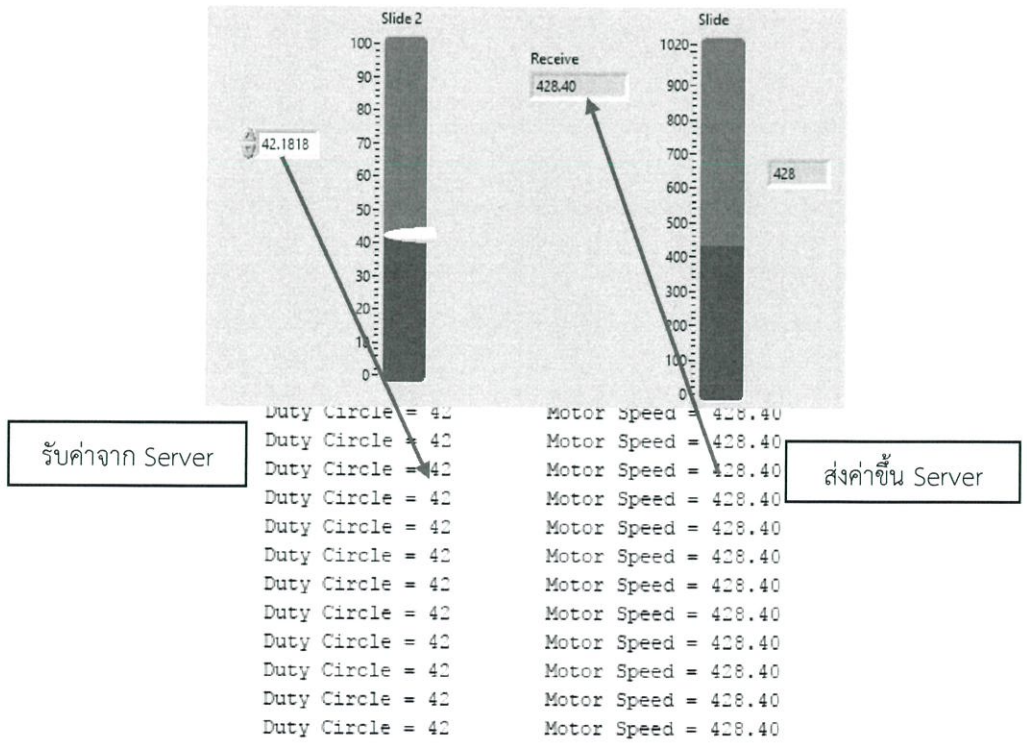
4. การส่งข้อมูลกลับไปที่ Server ใช้คำสั่ง client.println(*****); หมายถึงทุกครั้งที่ส่งข้อมูลจะขึ้นบรรทัดใหม่ และในกรณีที่ต้องการส่งข้อมูลมีหลายชุดในครั้งเดียว สามารถเขียนได้ 2 แบบคือ แบบรวมให้อยู่ในตัวแปรเดียวก่อนส่ง หรือแยกตัวแปรแล้วส่งทีละตัวแปร

แบบรวมตัวแปร

```
String dataClient = ((String)[redacted] + "|"+(String)[redacted]);
client.println(dataClient);
```

แบบแยกตัวแปร

```
client.print(outputValue);
client.print("|");
client.println(motorSpeedDUT);
```



รูปที่ 3.14 ผลการรับส่งค่าจาก Server

3.3.2 โค้ดทดสอบการเชื่อมต่อ Server (NETPIE) กับ NodeMCU ESP8266

```
/* connect to NETPIE with microgear function */
#include <MicroGear.h>
#define APPID   "[redacted]"  //"YOUR_APPID"
#define KEY     "[redacted]"  //"YOUR_KEY"
#define SECRET  "[redacted]"  //"YOUR_SECRET"
#define ALIAS   "[redacted]" //name of controller
#define FEEDID  "[redacted]"
#define APIKEY  "[redacted]"
#define MAX_TEMPC 100
#define MAX_TEMPF 150
#define MAX_HUMID 100
#define INTERVAL 15000
#define T_INCREMENT 200
#define T_RECONNECT 5000
int timer = 0;
char str[32];
/*.....................................................*/
/* connect to wifi */
#include <ESP8266WiFi.h>
const char* ssid = "[redacted]"; //ค่า ssid
const char* password = "[redacted]"; // ค่า password
WiFiClient client;
MicroGear microgear(client);
/*.....................................................*/

void onMsghandler(char *topic, uint8_t* msg, unsigned int msglen) {
    Serial.print("Incoming message -->");
    msg[msglen] = '\0';
    Serial.println((char *)msg);
}
void onConnected(char *attribute, uint8_t* msg, unsigned int msglen){
    Serial.println("Connected to NETPIE...");
      microgear.setAlias(ALIAS);
}

void setup()
{
  Serial.begin(115200); //เปิดใช้ serial
  WiFi.begin(ssid,password); // เชื่อมต่อกับ AP

  while (WiFi.status() != WL_CONNECTED) // รอการเชื่อมต่อ
  {
    delay(500);
    Serial.print(".");
  }
    Serial.println("");
    Serial.println("Wifi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP()); // แสดงหมายเลข IP
    microgear.on(MESSAGE,onMsghandler);
    microgear.on(CONNECTED,onConnected);
    microgear.init(KEY,SECRET,ALIAS);
    microgear.connect(APPID);
    Serial.println("Connect to TCP Server");
}
```

1

2

3

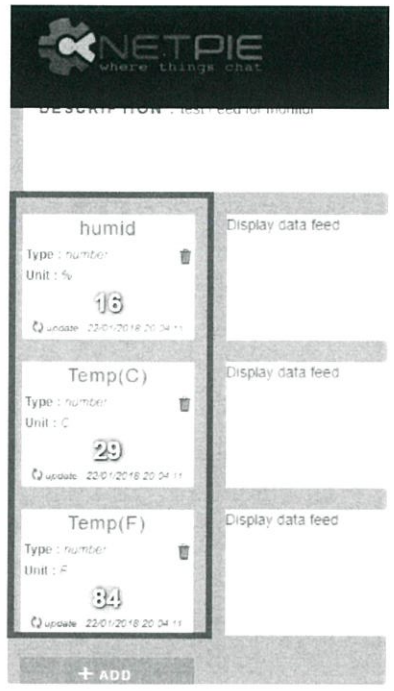
4

1. ข้อมูลตรงส่วนนี้ เป็น Key ที่ใช้เชื่อมต่อกับ Server ได้จากการสมัคร NETPIE
2. โค้ดในส่วนนี้จะคล้ายกับโค้ด Server LabVIEW ต่างกันตรงที่ไม่ต้องกำหนด IP และ Port
3. Void ทั้ง 2 ชุดนี้ใช้สำหรับการตรวจสอบข้อมูลที่รับมาและสถานะการเชื่อมต่อ
4. โค้ดในส่วนนี้คล้ายกับโค้ด Server ของ LabVIEW แต่เพิ่มเติมส่วนของฟังก์ชัน Microgear

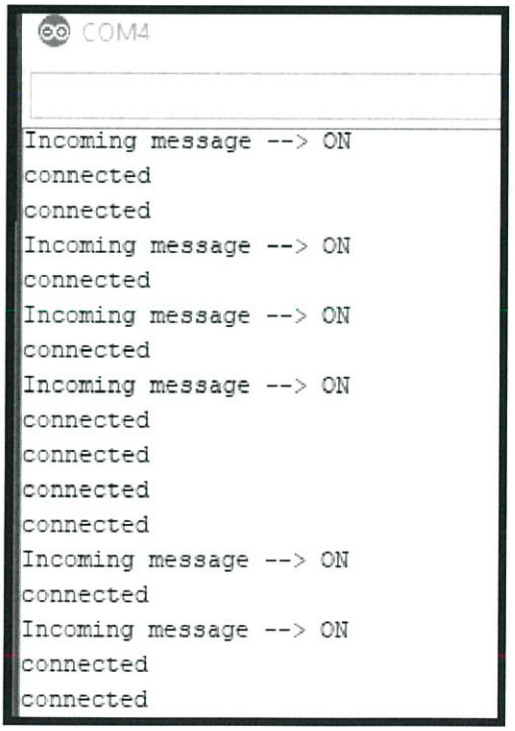
```
void loop()
{
        if (microgear.connected()) {
        Serial.println("connected");
        microgear.loop();
        if (timer >= INTERVAL) {
            sprintf(str,"%d,%d,%d",h,t,f);
            microgear.publish("/node1/dht",str);
            Serial.print("\nHumidity: ");

            String data = "{\"humid\":";
            data += h;
            data += ", \"Temp(C)\":";
            data += t ;
            data += ", \"Temp(F)\":";
            data += f ;
            data += "}";
            if (isnan(h) || isnan(t) || isnan(f) ||
            h >= MAX_HUMID || t >= MAX_TEMPC || f >= MAX_TEMPF) {
              Serial.println("Failed to read from DHT sensor!");
            }else{
              Serial.print("Sending -->");
              Serial.println((char*) data.c_str());
              microgear.writeFeed(FEEDID,data); //YOUR  FEED ID, API KEY
            }
           timer = 0;
        }
     else timer += T_INCREMENT;
    }
    else {
        Serial.println("connection lost, reconnect...");
        if (timer >= T_RECONNECT) {
            microgear.connect(APPID);
            timer = 0;
        }
        else timer += T_INCREMENT;
    }

  delay(T_INCREMENT);
    }
```

5. ในการส่งข้อมูลขึ้น Server ใช้คำสั่ง microgear.publish(“/ชื่อALIAS/Topic”, ชื่อข้อมูล) โดยสามารถส่งข้อมูลหลายชุดเป็น Array ที่มีเครื่องหมาย “,” คั่น เมื่อข้อมูลอยู่ที่ Server สามารแบ่งข้อมูลไปใช้แสดงผลโหมด Freeboard ในแต่ละ Widget ได้เลย

6. ในการส่งข้อมูลเพื่อแสดงผลในโหมด Feed ของ NETPIE จะใช้คำสั่ง microgear.writeFeed (ชื่อ FeedID, ชื่อชุดข้อมูล) โดยในที่นี้ส่งข้อมูลทั้งหมด 3 อย่าง โดยเก็บไว้ที่ตัวแปรชื่อ data ตัวเดียว ซึ่งชื่อตัวแปรแต่ละตัวที่จะนำไปใช้จะต้องตั้งให้ตรงกับตัวแปรที่ Server รับด้วยในหน้า Feed

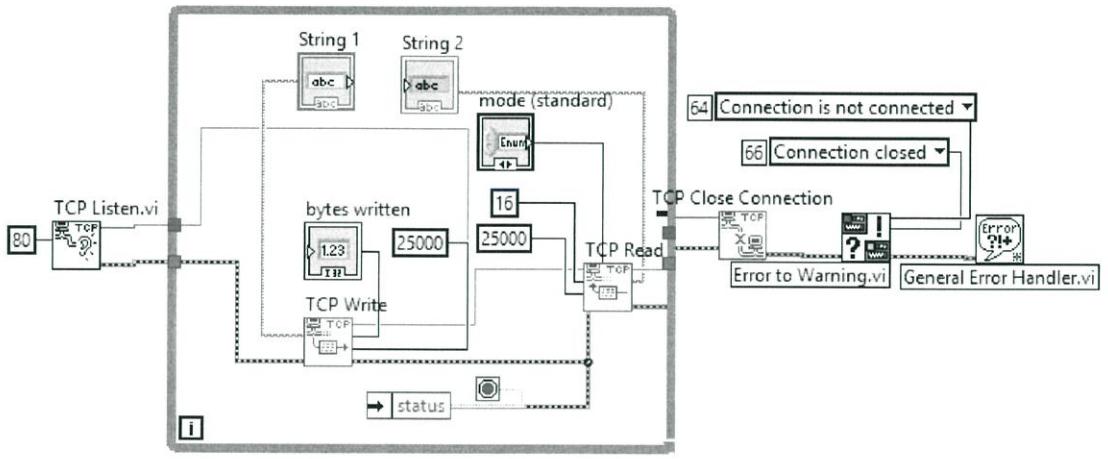


รูปที่ 3.15 การตั้งชื่อตัวแปรใน NETPIE

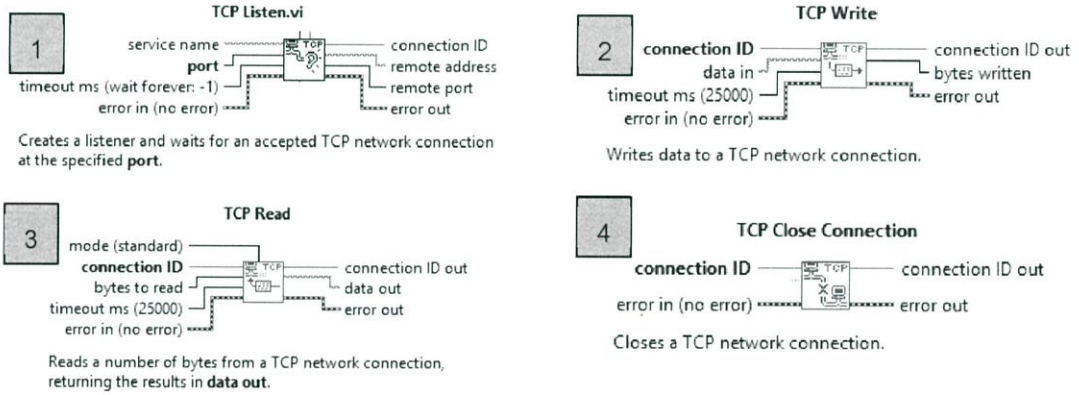


รูปที่ 3.16 ผลการเชื่อมต่อกับ Server NETPIE

3.4 การเขียน Function block ของโปรแกรม LabVIEW เพื่อสร้าง Server



รูปที่ 3.17 การเขียน Function block รับส่งข้อมูล



รูปที่ 3.18 Function block ที่เกี่ยวข้อง

การเชื่อมต่อระหว่าง NodeMCU กับโปรแกรม LabVIEW แบบไร้สายสามารถเขียน Block diagram เพื่อรับค่าที่ส่งเข้ามาและสั่งค่ากลับโดยใช้ฟังก์ชันจากรูปที่ 3.18

1. TCP Listen เป็นฟังก์ชันในการตั้งค่าเริ่มต้นในการสื่อสาร โดยโครงงานนี้ใช้วิธีการกำหนด Port ในการเชื่อมต่อระหว่าง NodeMCU ซึ่งเป็นตัว Client และ LabVIEW ซึ่งเป็นตัว Server สำหรับส่งข้อมูลและรับค่ากลับเพื่อควบคุมการทำงานของปั๊ม โดย Port ที่ใช้คือ 80

2. TCP Write เป็นฟังก์ชันในการส่งค่ากลับ โดยรับค่าจากช่อง Data in

3. TCP Read เป็นฟังก์ชันในการอ่านค่าที่รับมาจาก Client โดยค่าที่อ่านได้จะถูกกำหนดด้วย byte to read ก่อนออกทางช่อง Data out ไปแสดงผล โดยมีโหมดในการอ่านแตกต่างกัน ดังรูปที่ 3.19

4. TCP Close connect ใช้สำหรับปิดการเชื่อมต่อเพื่อให้การทำงานครบวงจรการสื่อสาร

mode indicates the behavior of the read operation.

0	**Standard** (default)—Waits until all bytes you specify in **bytes to read** arrive or until **timeout ms** runs out. Returns the number of bytes read so far. If fewer bytes than the number of bytes you requested arrive, returns the partial number of bytes and reports a timeout error.
1	**Buffered**—Waits until all bytes you specify in **bytes to read** arrive or until **timeout ms** runs out. If fewer bytes than the number you requested arrive, returns no bytes and reports a timeout error.
2	**CRLF**—Waits until all bytes you specify in **bytes to read** arrive or until the function receives a CR (carriage return) followed by a LF (linefeed) within the number of bytes you specify in **bytes to read** or until **timeout ms** runs out. The function returns the bytes up to and including the CR and LF if it finds them in the string.
3	**Immediate**—Waits until the function receives any bytes from those you specify in **bytes to read**. Waits the full timeout only if the function receives no bytes. Returns the number of bytes so far. Reports a timeout error if the function receives no bytes.

รูปที่ 3.19 โหมดในการรับค่าของ TCP Read

1. Standard (Default)

เป็นโหมดมาตรฐาน โดยจะรอรับข้อมูล byte ทั้งหมดจนกว่าจะถึง byte ที่กำหนด (byte to read) หรือรอจนกว่าจะถึงค่าเวลาที่ตั้งไว้ (timeout ms) โดยจะแสดงจำนวน byte ที่อ่านแล้ว ถ้าจำนวน byte ที่เข้ามามีน้อยกว่าที่กำหนดไว้ จะส่ง byte กลับบางส่วนและส่งค่าผิดพลาดไปยัง timeout error

2. Buffered

จะรอรับข้อมูลใน byte ทั้งหมดจนกว่าจะถึง byte ที่กำหนด (byte to read) หรือ รอจนกว่าจะถึงค่าเวลาที่ตั้งไว้ (timeout ms) ถ้าจำนวน byte ที่เข้ามามีน้อยกว่าที่กำหนดไว้ จะไม่ส่ง byte กลับและไม่ส่งค่าผิดพลาดไปยัง timeout error

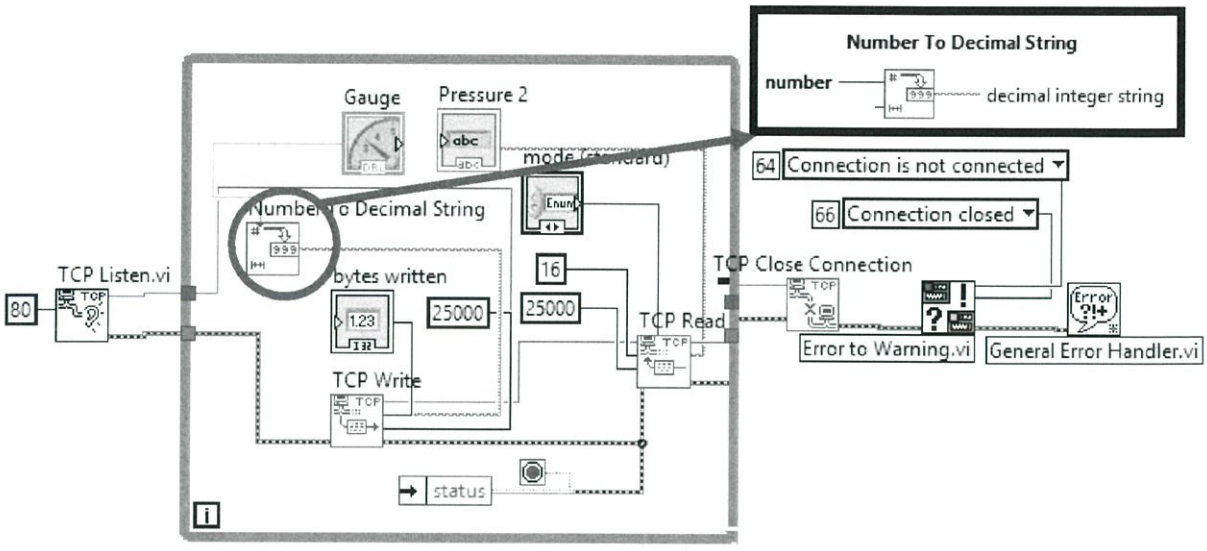
3. CRLF

จะรอรับข้อมูล byte ทั้งหมดจนกว่าจะถึง byte ที่กำหนด (byte to read) หรือรอจนกว่าจะได้รับค่า CR (Carriage return หมายถึง อักขระพิเศษที่กำหนดขึ้นเพื่อใช้เป็นข้อตกลงในการทำงาน) หรือด้วยค่า LF (Linefeed หมายถึง การขึ้นบรรทัดใหม่ของข้อมูล) ภายในจำนวน byte ที่กำหนดไว้หรือกระทั่งหมดเวลาที่กำหนดไว้ (timeout ms) ฟังก์ชันจะส่งค่า byte กลับคืนพร้อมทั้ง CR และ LF โดยสามารถดูได้จากค่า String

4. Immediate

จะรอรับข้อมูล byte ทั้งหมด จนกระทั่งฟังก์ชันได้รับ byte จากระบุเป็น byte ที่จะอ่านแล้ว ถ้าไม่มี byte เข้ามาแล้ว ฟังก์ชันจะรอจนกว่าจะถึงค่าเวลาที่ตั้งไว้ (timeout ms) เท่านั้น ถ้าจำนวน byte ที่เข้ามามีน้อยกว่าที่กำหนดไว้ จะส่ง byte กลับและส่งรายงานไปยัง timeout error หากฟังก์ชันไม่มี byte เข้ามา

ในโครงงานนี้เลือกใช้โหมด CRLF ในการอ่านข้อมูลเนื่องจากข้อมูลที่ส่งมาจากบอร์ด มีการขึ้นบรรทัดใหม่ทุกครั้งหลังจากส่งเสร็จในแต่ละครั้ง เพื่อให้การแสดงผลปรากฏตัวเลขเพียงชุดเดียว

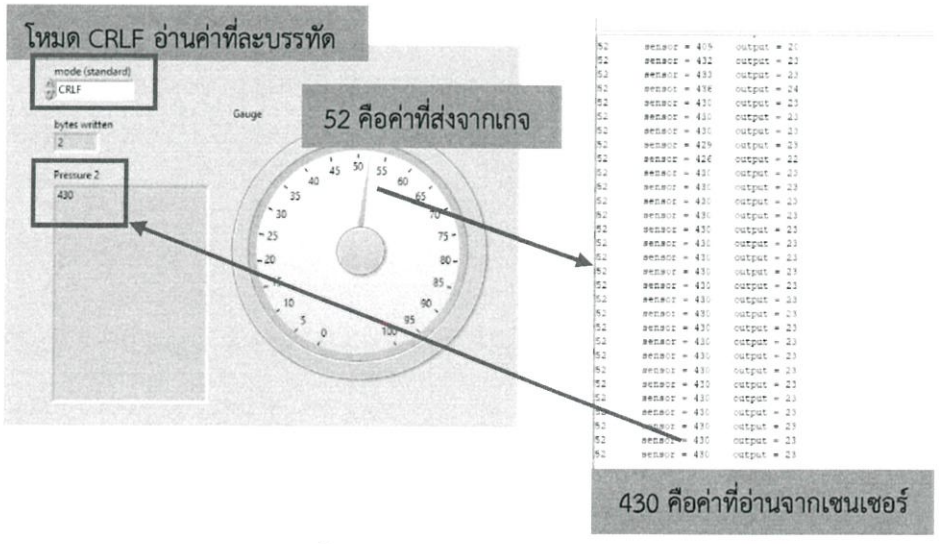


รูปที่ 3.20 การแปลงข้อมูล Decimal เป็น String

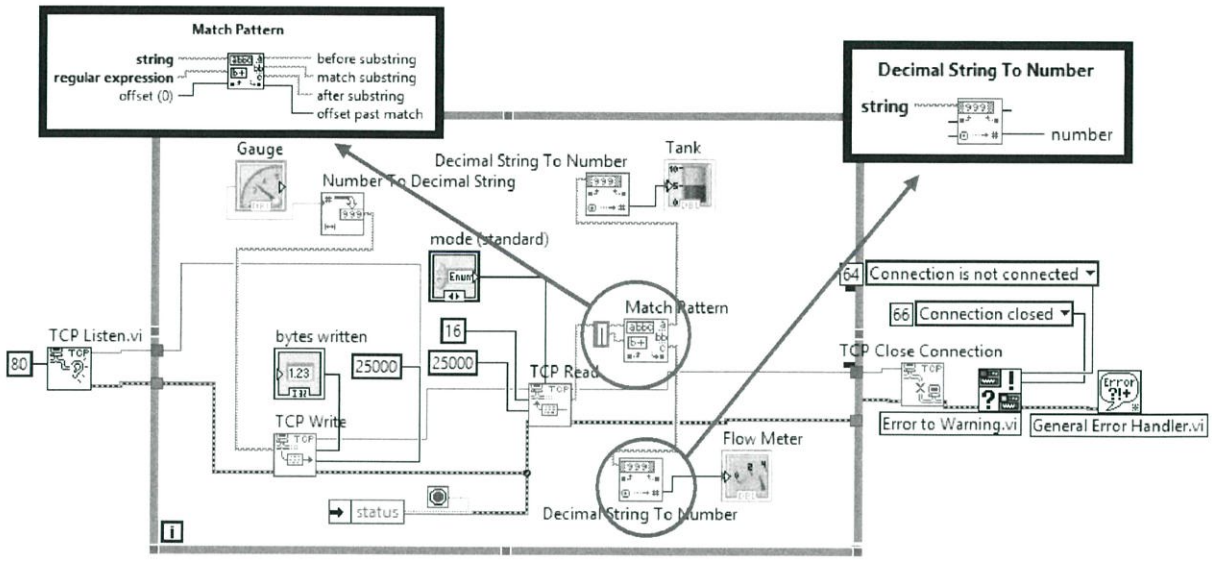
หลังจากเขียนฟังก์ชันในการรับค่าและส่งค่าด้วย TCP/IP แล้ว จากนั้นทดลองรับส่งค่า โดยการสื่อสารแบบ TCP/IP ข้อมูลที่ใช้รับส่งนั้นจะเป็น String ทั้งหมด ดังนั้นจึงต้องแปลงค่าเพื่อนำไปใช้งานอื่นๆ ดังนี้

1. ทดลองส่งค่าจากเกจใน Front panel พบว่าค่าที่ได้จะเป็น Decimal number ทำให้ไม่สามารถส่งข้อมูลได้ เนื่องจากโปรโตคอล TCP/IP รับส่งข้อมูลแบบ String เท่านั้น จึงต้องทำการแปลงข้อมูลเป็น String ด้วยฟังก์ชัน Number to decimal string ก่อนนำข้อมูลเข้าสู่ Data in ของ TCP Write เพื่อทำการส่งค่ากลับไปยัง Client

2. ทดลองรับค่าจากบอร์ด โดยข้อมูลที่ส่งมายังฟังก์ชัน TCP Read จะเป็น String จึงใช้ฟังก์ชันที่อ่านค่า String เพื่อดูค่าที่ส่งมา

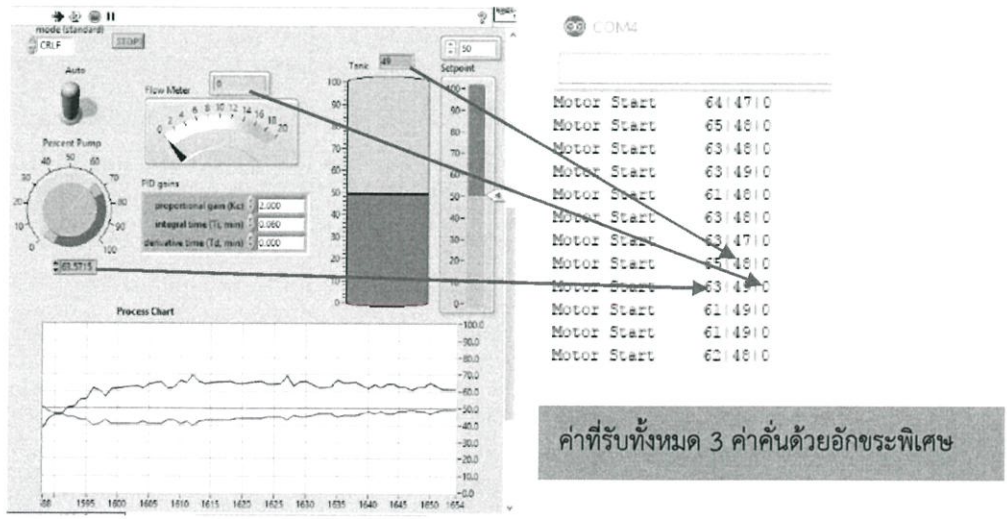


รูปที่ 3.21 ผลการรับและส่งข้อมูล

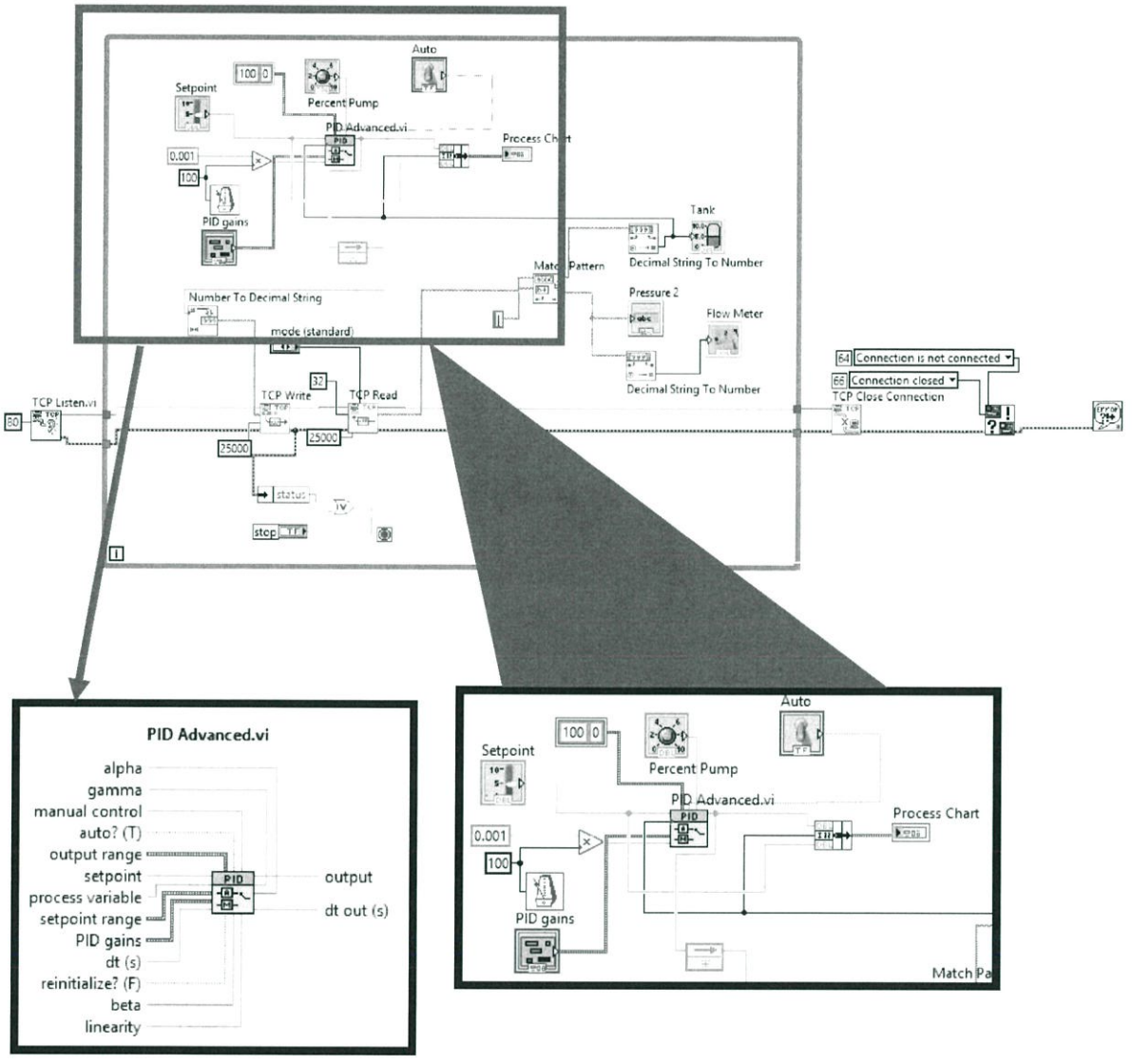


รูปที่ 3.22 การใช้ Match pattern และการแปลงค่า String เป็น Decimal

หลังจากทดลองรับส่งค่าได้แล้ว ขั้นต่อไปคือ ทดลองรับค่าแล้วนำไปแสดงผลด้วยเกจต่างๆ ในหน้า Front panel แต่เนื่องจากค่าที่ต้องการส่งขึ้นมาแสดงผลที่หน้าจอมอนิเตอร์มี 2 ค่า คือ ค่าระดับน้ำจาก Pressure sensor และอัตรการไหลจาก Flow sensor ดังนั้นการส่งข้อมูลจะต้องมีการแบ่งข้อมูลเพื่อนำไปใช้ประมวลผลคนละส่วน จึงใช้ฟังก์ชัน Match pattern ในการจัดการ โดยฟังก์ชันนี้จะแบ่งข้อมูลออกเป็นสองส่วนโดยใช้อักขระที่กำหนดเป็นตัวแบ่ง โดยกำหนดอักขระนั้นไว้ในช่อง Match substring และข้อมูลจะถูกแบ่งออกเป็นสองส่วน ส่วนแรกคือ Before substring และส่วนหลังคือ After substring โดยในโครงงานนี้ใช้อักขระ | เป็นตัวแบ่งข้อมูล จากนั้นข้อมูลที่ได้จะเป็นค่า String แต่เกจและมิเตอร์ที่จะใช้วัดค่าแสดงออกมาที่หน้ามอนิเตอร์รับค่าเป็น Decimal หรือ Fract/Exp (เลขทศนิยม) ดังนั้นจึงต้องใช้ฟังก์ชันในการแปลงข้อมูล String เป็นเลข Decimal ด้วยฟังก์ชัน Decimal string to number



รูปที่ 3.23 ผลการทำงานของ Match pattern



รูปที่ 3.24 การใช้งานฟังก์ชัน PID advance

หลังจากแบ่งข้อมูลได้แล้วก็เข้าสู่การควบคุมโดยใช้ฟังก์ชัน PID โดยโปรแกม LabVIEW มีฟังก์ชัน PID หลายตัวให้เลือกใช้ แต่ในโครงงานนี้เลือกใช้ฟังก์ชัน PID advance เนื่องจากสามารถเข้าใจการทำงานได้ง่ายและไม่ซับซ้อนจนเกินไป โดยกำหนดค่าเริ่มต้นดังนี้

- Manual control ใช้เป็นตัวควบคุมแบบปกติโดยไม่ใช้ตัวควบคุม
- Auto(T) เป็นการกำหนดการทำงานว่าจะใช้ PID ในการควบคุม หรือจะเป็นโหมด Manual
- Output range เป็นการกำหนดช่วงการทำงานของค่า Output โดยค่า Default คือ 100 ถึง -100 แต่ในการควบคุมปั๊มน้ำจะเป็นการทำงานตั้งแต่ค่า 0 ขึ้นไปเพราะค่าติดลบจะเป็นการทำงานตรงข้ามหรือสั่งปั๊มน้ำหมุนกลับทิศ ซึ่งปั๊มที่ใช้ไม่สามารถทำงานได้ ดังนั้นจึงกำหนดค่าเป็น 0 ถึง 100

- Setpoint เป็นการกำหนดค่าให้ระบบทำตามจนถึงระดับที่ตั้งไว้ โดยใช้ฟังก์ชัน Slide ในการกำหนดเป็น Input เข้าสู่ฟังก์ชัน PID

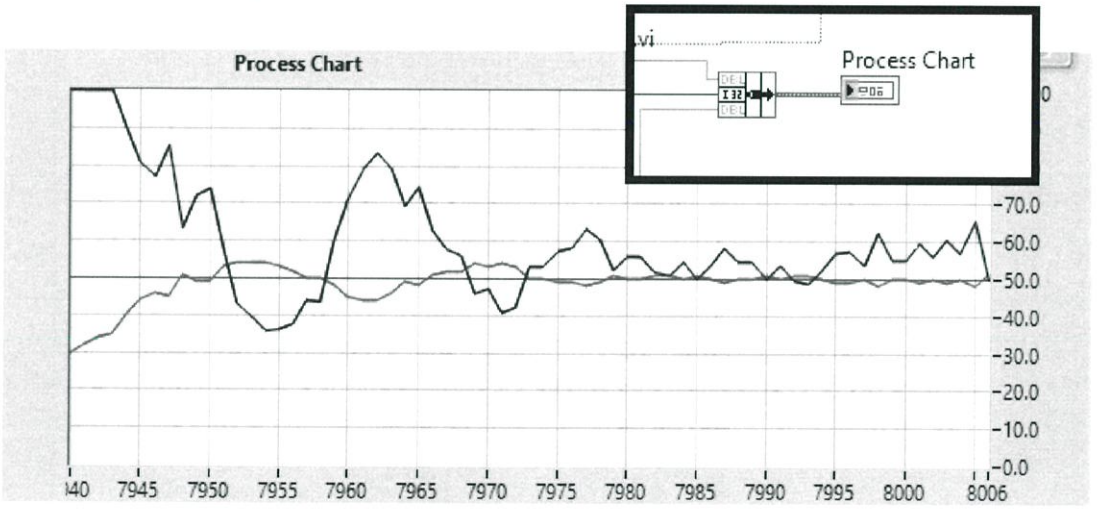
- Process Value เป็นค่าผลลัพธ์ที่ได้จากระบบการทำงาน ในที่นี้คือค่าระดับน้ำ ดังนั้นหลังจากแบ่งข้อมูลและแปลงค่าระดับน้ำจากเซนเซอร์ที่วัดค่าเป็น Analog เป็นระดับเปอร์เซ็นต์แล้ว จึงนำค่ามาเป็น Input ใส่ฟังก์ชัน PID ได้เลย

- PID gain เป็นการกำหนดค่าพารามิเตอร์ในการควบคุมการทำงานให้มีความเสถียร โดยมีทั้งหมด 3 ค่าคือ Kp, Ti และ Td

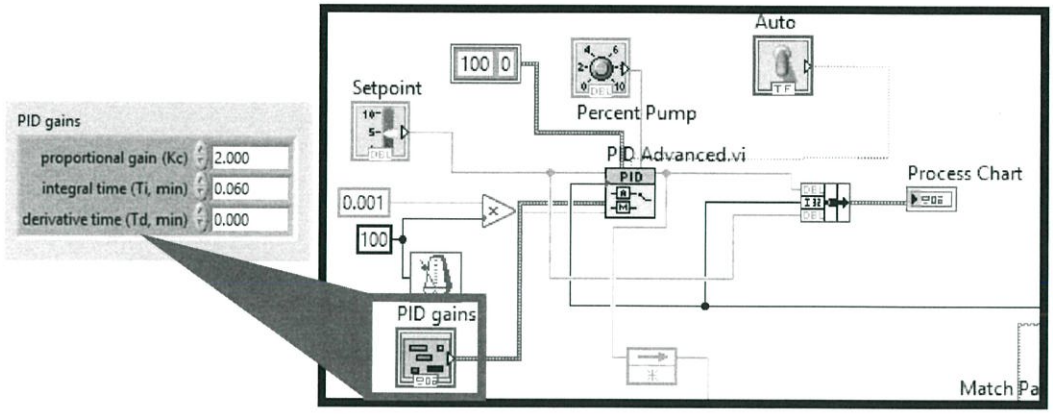
- dt(s) เป็นการกำหนดระยะเวลาในการทำงาน ในที่นี้ใช้ Timer หน่วงระยะเวลาการทำงานและใช้ตัวคูณ 100 และ 0.001 เนื่องจากโปรแกรม LabVIEW มีการทำงานเป็น µs

- Output เป็นค่าที่ได้จากการปรับปรุงข้อมูลด้วย PID แล้ว ค่านี้จึงใช้ส่งกลับไปยังตัวควบคุมเพื่อสั่งปั๊ม โดยทำการ Feedback กลับไปยัง Data in ของฟังก์ชัน TCP Write

ในส่วนของการมอนิเตอร์ค่าต่างๆ สามารถนำข้อมูลมาแสดงเป็นกราฟเพื่อดูการตอบสนองของระบบได้แบบ Real time ได้จากฟังก์ชัน Waveform chart และสามารถปรับรูปแบบการแสดงผลได้หลากหลาย ทั้งขนาด สี รูปแบบกราฟ

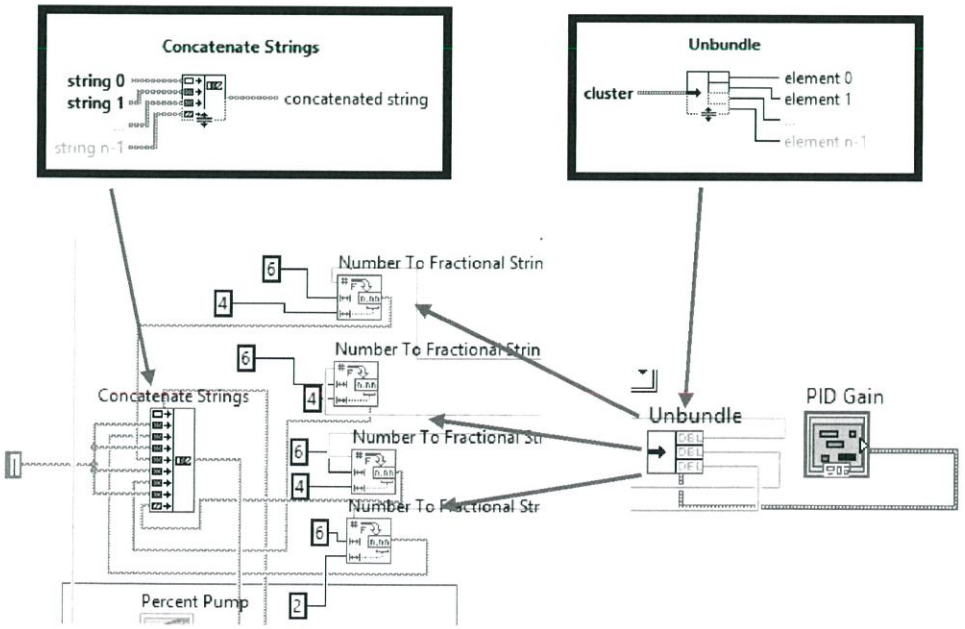


รูปที่ 3.25 การใช้งานการแสดงผลแบบกราฟ

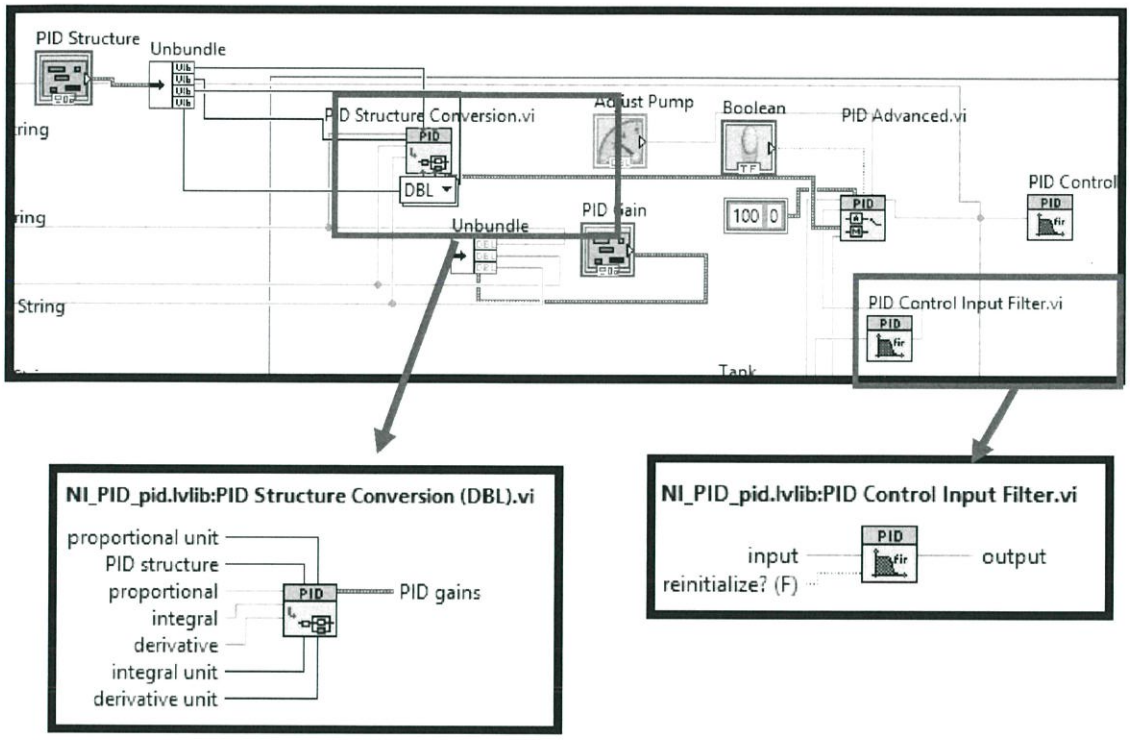


รูปที่ 3.26 การใช้งานพารามิเตอร์ PID

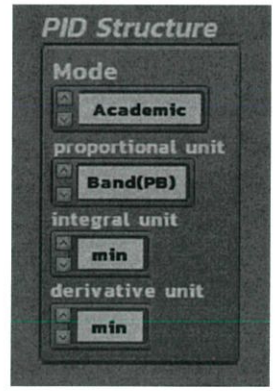
หลังจากทดลองควบคุมระดับน้ำด้วย PID แล้ว ขั้นต่อไปคือ การส่งค่า PID ที่ตั้งค่าไว้กลับไปยังบอร์ดเพื่อจะให้บอร์ดส่งค่านี้ขึ้นไปแสดงผลใน Cloud platform เพื่อมอนิเตอร์จากระยะไกล แต่เนื่องจากฟังก์ชัน PID มีพารามิเตอร์ต่างๆ อยู่ในกรอบที่เรียกว่า Cluster ทำให้เส้นข้อมูลที่ออกจากฟังก์ชันมีเพียงค่าเดียวเพื่อความง่ายในการใช้งาน ซึ่งเป็นรูปแบบเริ่มต้นจากการเรียกใช้งานฟังก์ชัน ทำให้การจะนำข้อมูลแต่ละตัวไปใช้ต้องทำการแยกข้อมูลแต่ละพารามิเตอร์ออกมาทีละตัว เพื่อแปลงค่าก่อนจะนำมารวมกันอีกครั้งเพื่อส่งกลับ โดยใช้ฟังก์ชัน Unbundle รับค่าจาก Cluster PID เพื่อแยกข้อมูลแต่ละพารามิเตอร์ ก่อนเข้าฟังก์ชันแปลงเป็น String โดยใช้ฟังก์ชัน Number to string แล้วนำค่า String ทั้งหมดไปรวมโดยมีอักขระพิเศษคั่นเพื่อใช้เป็นตัวตัดข้อมูลภายในบอร์ดภายหลัง โดยใช้ฟังก์ชัน Concatenate string ก่อนจะส่งไปยัง Data in



รูปที่ 3.27 การใช้งาน Concatenate string และ Unbundle



รูปที่ 3.28 การใช้งานฟังก์ชัน PID structure และ Filter

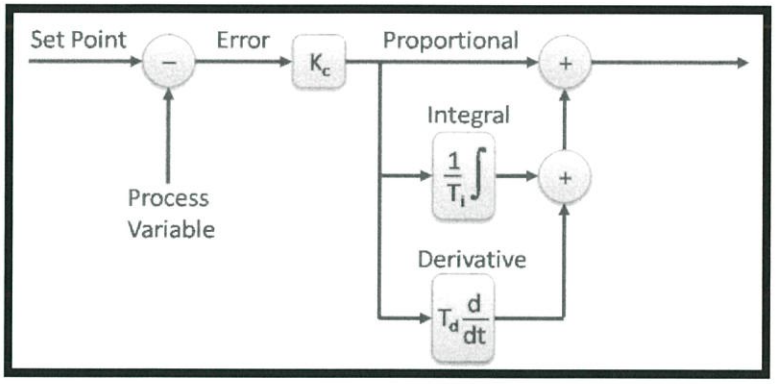


รูปที่ 3.29 หน้าแสดงผล PID structure

ในส่วนของ PID structure นั้นเป็นส่วนเสริมที่ทำขึ้นเพื่อศึกษาความแตกต่างของโหมด การควบคุมแบบ PID ที่ตั้งค่าหน่วย (Unit) และโหมดแตกต่างกัน โดยใช้ฟังก์ชัน PID structure conversion ซึ่งค่าที่ใช้กับฟังก์ชันนี้ สามารถนำค่า Proportional, Integral และ Derivative ที่แปลงค่าออกจาก Cluster แล้วมาใช้ได้เลย โดยโหมดที่ใช้มีให้เลือกทั้งหมด 3 โหมดคือ Academic, Series และ Parallel ส่วน Filter นั้นใช้เพื่อกรองสัญญาณรบกวนที่รับค่ามาออก ทำให้สัญญาณที่ได้นิ่งขึ้น

0	**Academic (Standard)** (default)—Specifies that the controller is in PID Academic form.
1	**Parallel**—Specifies that the controller is in PID Parallel form.
2	**Series**—Specifies that the controller is in PID Series form.

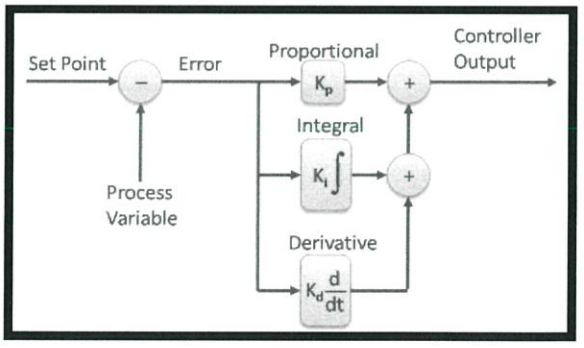
รูปที่ 3.30 ประเภทของโหมด PID structure



รูปที่ 3.31 Academic form

Academic form เป็นอัลกอริทึมควบคุมแบบเก่า สามารถเรียกได้หลายชื่อ เช่น Non-interacting, Ideal, Standard หรือ ISA อัลกอริทึม โดยกฎของ Cohen-coon และ Lambda PID มีพื้นฐานการออกแบบมาจากอัลกอริทึมนี้และจากรูปที่ 3.31 สามารถเขียนเป็นสมการที่ (3.1)

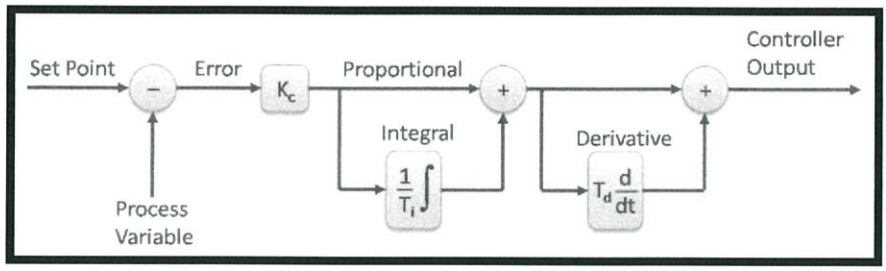
$$\frac{U(s)}{Y(s)} = K_c\left(1 + \frac{1}{sT_i} + sT_d\right) \tag{3.1}$$



รูปที่ 3.32 Parallel form

เป็นอัลกอริทึมที่ไม่มีเกนและการปรับแต่งค่ามีผลต่อสัดส่วนเฉพาะโหมด ในการใช้งานจริงมักไม่นิยมใช้ เนื่องจากปรับแต่งค่ายาก แต่ในอุปกรณ์ควบคุมบางตัวมักใส่มาเป็นตัวเลือก จากรูปที่ 3.32 สามารถเขียนให้อยู่ในรูปของสมการได้ดังสมการที่ (3.2)

$$\frac{U(s)}{Y(s)} = K_p + \frac{K_i}{s} + sK_d \tag{3.2}$$



รูปที่ 3.33 Series form

Series form สามารถเรียกได้หลายชื่อ เช่น Classical, Real หรือ Interactive อัลกอริทึม โดยกฎของ Ziegler-nichols ได้รับการพัฒนามาจากอัลกอริทึมนี้ ในปัจจุบันมักพบได้ในตัวควบคุมแบบนิวเมติกและอิเล็กทรอนิกส์ จากรูปที่ 3.33 สามารถเขียนให้อยู่ในรูปของสมการได้ดังสมการที่ (3.3)

$$\frac{U(s)}{Y(s)} = K_c(1 + \frac{1}{sT_i})(1 + sT_d) \tag{3.3}$$

0	**Gain (Kc)** (default)—Specifies that the proportional component is expressed in terms of proportional gain (K_c).
1	**Band (PB)**—Specifies that the proportional component is expressed in terms of proportional band (*PB*).

รูปที่ 3.34 ประเภทหน่วยของ Proportional

หน่วยของ Proportional มีแบบเกน (Kc) เป็นตัวคูณเพื่อเพิ่มค่า หรือกำหนดช่วงสูงสุดด้วย Band (PB) แสดงในสมการที่ (3.4) และสมการที่ (3.5)

$$U_p(t) = K_c e(t) \tag{3.4}$$

$$K_p = \frac{100\%}{PB} \tag{3.5}$$

0	**min** (default)—Specifies that the integral component is expressed in minutes.
1	**s**—Specifies that the integral component is expressed in seconds.

รูปที่ 3.35 ประเภทหน่วยของ Integral และ Derivative

หน่วยของ Integral และ Derivative มีให้เลือกเป็นหน่วยนาที (min) และหน่วยวินาที (s)

3.5 การรวมโค้ด

โค้ดรวมใช้วิธีการเขียนแบบ Multitasking ซึ่งเป็นการเขียนแบบแยกไฟล์การทำงานออกจากกัน เพื่อป้องกันความแตกต่างของเวลาในการทำงานของเซนเซอร์แต่ละตัว แต่สามารถรวมไฟล์ให้การทำงานสามารถทำงานพร้อมกันได้ โดยสร้างไฟล์ในแทบด้านบน แล้วตั้งชื่อไฟล์ให้มีนามสุกล .h แล้ว save จะมีชื่อไฟล์ย่อยที่สร้างไว้อยู่ในโฟลเดอร์เดียวกับไฟล์ .ino ซึ่งโค้ดส่วนใหญ่จะคล้ายกับของเดิม แต่จะมีการปรับค่าบางอย่างที่แตกต่างไปจากเดิม

รูปที่ 3.36 แทบการสร้าง Task

Name	Date modified	Type	Size
FlowTask.h	16/3/2561 13:39	H File	1 KB
LightTask.h	11/3/2561 16:06	H File	1 KB
Multi_test1.ino	12/5/2561 15:11	Arduino file	3 KB
NETPIEtask.h	16/3/2561 13:37	H File	3 KB
OTAtask.h	26/2/2561 1:48	H File	1 KB
PresTask.h	2/5/2561 9:29	H File	1 KB
PumpTask.h	7/4/2561 16:04	H File	4 KB
test.h	12/5/2561 15:11	H File	1 KB

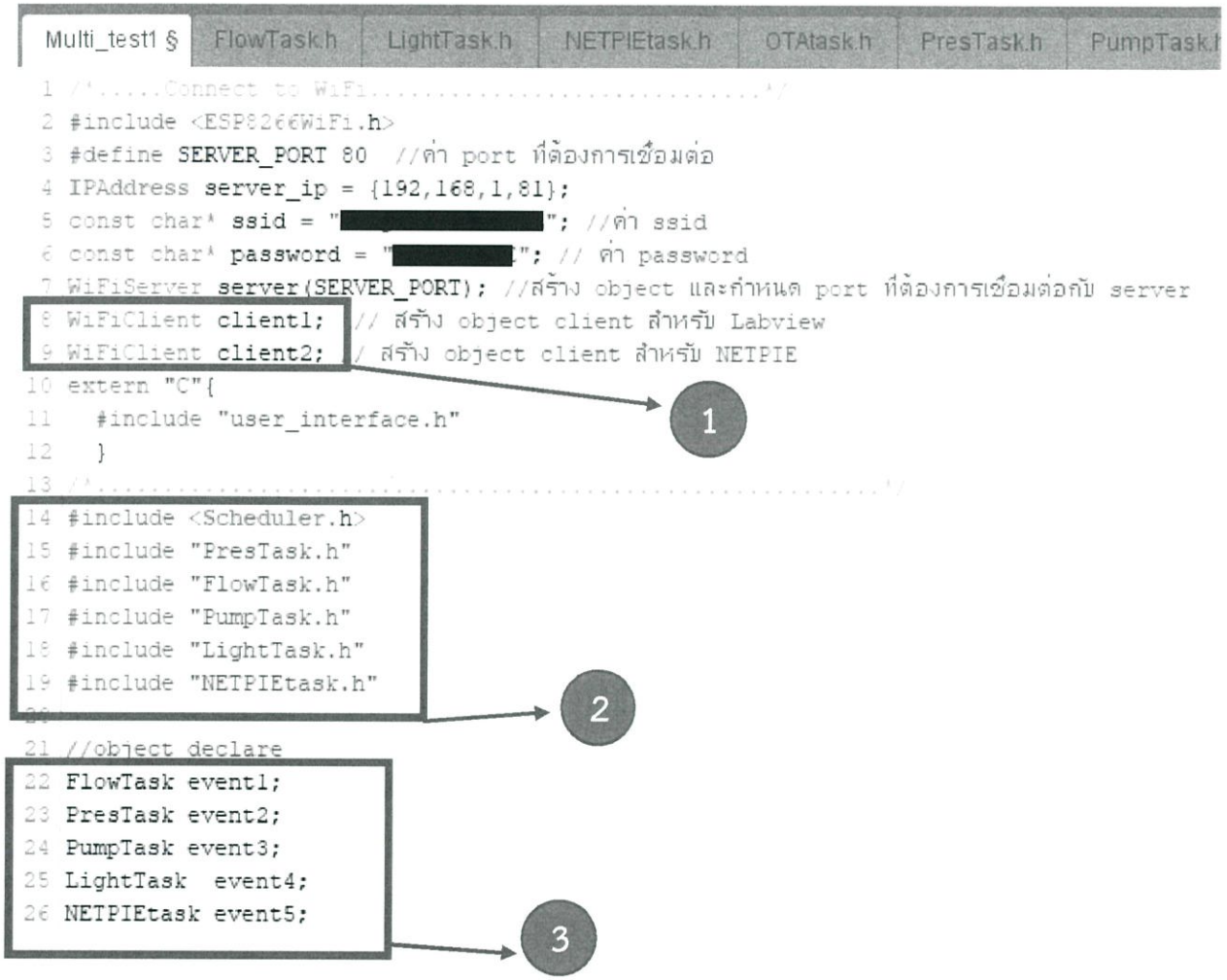
รูปที่ 3.37 ที่เก็บไฟล์ Task ที่ถูกสร้างไว้

โดยใน Task ที่เป็น Main จะทำการดึงไฟล์ Task ย่อยๆ มาและกำหนดค่า Event ของแต่ละ Task และใน Task main นั้น ไม่สามารถใช้ void loop() ได้ ส่วนใน Task ย่อยๆ นั้น ต้องเขียนตามรูปแบบดังรูปที่ 3.38 โดยต้องตั้งชื่อ Class ซึ่งชื่อนี้จะถูกนำไปกำหนดค่า Event ใน Task main

```
1  class ******* : public Task {
2  private:
3
4  protected:
5
6      void setup()
7      {
8      }
9      void loop()
10     {
11     }
12 };
```

รูปที่ 3.38 การสร้างชื่อ Class

3.5.1 โค้ด Main ของโค้ดรวมโดยเขียนแบบ Multitasking



1. เพิ่มตัวแปร Client เพราะต้องเชื่อมต่อ ทั้ง 2 Server ระหว่าง LabVIEW และ NETPIE
2. #include “ชื่อไฟล์ Task ย่อย” เป็นการดึงค่าของ Task อื่นมาทำงานร่วมกัน
3. กำหนด Event โดยใช้ชื่อ Class ที่ได้จากไฟล์ Task ย่อย

```
27
28 /*.....*/
29 void setup() {
30   Serial.begin(115200); //เปิดใช้ serial
31   wifi_status_led_install(2,  PERIPHS_IO_MUX_GPIO2_U, FUNC_GPIO2);
32    //system_update_cpu_freq(80);
33   system_update_cpu_freq(160);
34   WiFi.begin(ssid,password); // เชื่อมต่อกับ AP
35    while (WiFi.status() != WL_CONNECTED) // รอการเชื่อมต่อ
36   {
37     delay(500);
38     Serial.print(".");
39
40   }
41     Serial.println("");
42     Serial.println("Wifi connected");
43     Serial.println("IP address: ");
44     Serial.println(WiFi.localIP()); // แสดงหมายเลข IP
45
46     Serial.println("Connect to TCP Server");
47     while (!client1.connect(server_ip,SERVER_PORT)) //เชื่อมต่อกับ Server
48     {
49       Serial.print(".");
50       delay(100);
51      }
52   Serial.println("Success");
53   Scheduler.start(&event1);
54   Scheduler.start(&event2);
55   Scheduler.start(&event3);
56   Scheduler.start(&event4);
57   Scheduler.start(&event5);
58   Scheduler.begin();
59
60 }
61
62 void loop() {
63
64   }
65
```

4

5

4. คำสั่งไฟกระพริบที่ขา DI4 (GPIO2) โดยเมื่อยังไม่เชื่อมต่อกับ Wi-Fi ไฟจะกระพริบถี่ เพื่อใช้ตรวจสอบสถานะในการเชื่อมต่อกับ Server

5. กำหนดการทำงานของแต่ละ Event

3.5.2 โค้ด Task ของ Flow sensor ในโค้ดรวมโดยเขียนแบบ Multitasking

Multi_test1 | FlowTask.h § | LightTask.h | NETPIEtask.h | OTAtask.h | PresTask.h | PumpTask.h

```
volatile int NbTopsFan;
float Calc;
#define hallsensor  4 //flow sensor ขา DI2
void rpm()
    {
      NbTopsFan++;
     }
class FlowTask : public Task {
private:

protected:

    void setup() {
        pinMode(hallsensor,INPUT);
        attachInterrupt(hallsensor,rpm,RISING);
    }
    void loop() {
         NbTopsFan = 0; //Set NbTopFan to 0 ready for calculate
         sei();  //Enables interrupt
         delay(100); // wait 0.1 second
         cli(); // Disable interrupt
         Calc = (NbTopsFan*1.45);
         String dataClient = {(String)OutputValue + "|" + (String)Calc};
         client1.println(dataClient);
         client1.flush();
   }
};
```

1

1. การเขียนแบบ Multitasking นั้น สามารถดึงตัวแปรคำสั่งข้าม Task ได้ โดยในที่นี้ดึงคำสั่งให้ส่งข้อมูลกลับไปที่ Server LabVIEW จำนวน 2 ตัวโดยส่งแบบมีอักขระ “|” คั่น เพื่อนำไปแบ่งในข้อมูลต่อใน LabVIEW มาใช้ใน Task ของ Flow sensor

3.5.3 โค้ด Task ของ Pressure sensor ในโค้ดรวมโดยเขียนแบบ Multitasking

Multi_test1 § | FlowTask.h § | LightTask.h | NETPIEtask.h | OTAtask.h | PresTask.h | PumpT

```
#include <Ticker.h>
Ticker flipper;
const uint32_t ANALOG_PIN = A0;
int sensorValue = 0;
int OutputValue = 0;

 void readValue()
  {
      sensorValue = analogRead(ANALOG_PIN);
      OutputValue = map(sensorValue,340,1020,0,100);
  }
class PresTask : public Task {
private:

protected:

    void setup() {
     flipper.attach(0.1, readValue);
    }
    void loop()
    {

     }
};
```

3.5.4 โค้ด Task ของไฟกระพริบเพื่อบอกสถานะ

โดยตั้งค่าไฟกระพริบที่ขา DI0 (GPIO16) ซึ่ง Task นี้จะทำงานหลังจากเชื่อมต่อกับ Server แล้ว (ดูจาก Event ในหน้า Main) ไฟที่ตัวควบคุมจะกระพริบทุกๆ 1 วินาที เป็นสถานะเพื่อตรวจสอบว่าตอนนี้มีการเชื่อมต่อกับ Server แล้ว

Multi_test1 | FlowTask.h | LightTask.h § | NETPIEtask.h § | OTAtask.h | PresTask.h | PumpT

```
#define ledBlink 16 //ขา DI0
class LightTask : public Task {
private:

protected:

    void setup() {
        pinMode(ledBlink,OUTPUT);
        digitalWrite(ledBlink,HIGH);
    }
    void loop() {
  digitalWrite(ledBlink, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                    // wait for a second
  digitalWrite(ledBlink, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                    // wait for a second
     }

};
```

3.5.5 โค้ด Task ของการรับค่าและส่งข้อมูลการทำงานของปั๊ม ในโค้ดรวมโดยเขียนแบบ Multitasking

Multi_test1 | FlowTask.h | LightTask.h | NETPIEtask.h § | OTAtask.h | PresTask.h | PumpTask.h §

```
#define RPWM  5 // motor ขา DI1
#define enL  0 // enable left ขา DI3
#define enR  12 // enable right ขา DI6
float motorSpeedPWM;
float motorSpeedDUT;
float Motor;
float Setpoint;
float Kp;
float Ti;
float Td;
String m_respond = ""; //ใช้รับค่า motor speed
float mapf(float x, float in_min, float in_max, float out_min, float out_max)
  {
     float result;
     result = (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
     return result;
   }
String data1 = "";
String data2 = "";
String data3 = "";
String data4 = "";
String data5 = "";
String getValue(String data, char separator, int index)  //ฟังก์ชันแบ่งข้อมูล
{
  int found = 0;
  int strIndex[] = {0, -1};
  int maxIndex = data.length()-1;

  for(int i=0; i<=maxIndex && found<=index; i++){
    if(data.charAt(i)==separator || i==maxIndex){
        found++;
        strIndex[0] = strIndex[1]+1;
        strIndex[1] = (i == maxIndex) ? i+1 : i;
    }
  }

  return found>index ? data.substring(strIndex[0], strIndex[1]) : "";
}
class PumpTask : public Task {
private:

protected:
    void setup() {
        pinMode(RPWM,OUTPUT);   //   กำหนดขา PWM
        pinMode(enL, OUTPUT);
        pinMode(enR, OUTPUT);
        digitalWrite(enL, HIGH);
        digitalWrite(enR, HIGH);

    }
```

1. โดยปกติโปรแกรมจะไม่มีฟังก์ชันแบ่งข้อมูลมาให้ จึงกำหนดขึ้นมาเองโดยตั้งชื่อว่า getValue

```
51      void loop() {
52         m_respond = "";           // ให้เป็นค่าว่างไว้ก่อน
53      while(clientl.available()) //ตรวจเช็คว่ามี Data ส่งมาจาก Server หรือไม่
54       {
55         char c = clientl.read();   //ใช้ char รอรับค่าจากการ read
56         m_respond += c;           //เปลี่ยนเป็น String
57
58         datal = getValue(m_respond,'|',0); //แบ่งข้อมูลออกเป็นส่วนๆ
59         data2 = getValue(m_respond,'|',1);
60         data3 = getValue(m_respond,'|',2);
61         data4 = getValue(m_respond,'|',3);
62         data5 = getValue(m_respond,'|',4);
63       }
64         motorSpeedDUT= datal.toFloat(); //เปลี่ยน String เป็น Float
65         Setpoint = data2.toFloat();
66         Kp = data3.toFloat();
67         Ti = data4.toFloat();
68         Td = data5.toFloat();
69
70    if(motorSpeedDUT <= 0)    // ถ้ารับค่ามาเป็น 0 ให้มอเตอร์หยุดทำงาน
71    {
72      digitalWrite(RPWM,LOW);
73      Serial.print("Motor Stop");
74      Serial.print("\t");
75      motorSpeedDUT = 0;
76      motorSpeedPWM = 0;
77
78    }
79    else if(motorSpeedDUT > 0)        //ถ้ารับค่ามากกว่า 0 ให้มอเตอร์ทำงาน
80      {
81        digitalWrite(RPWM,HIGH);
82        motorSpeedPWM = mapf(motorSpeedDUT, 0, 100, 395, 1020);
83        analogWrite(RPWM,motorSpeedPWM);
84        Serial.print("Motor Start");
85        Serial.print("\t");
86       }
87
88       Serial.printf("%.2f | %d | %.2f | %d | %d"
89                     ,motorSpeedDUT,OutputValue,Calc,sensorValue,NbTopsFan);
90       Serial.print("\n");
91       Serial.printf("\t\t%.2f | %.3f | %.3f | %.3f ",Setpoint,Kp,Ti,Td);
92       Serial.print("\n");
93       delay(100);
94
95      }
96  };
```

2. แบ่งชุดข้อมูลที่รับมาจาก Server LabVIEW โดยตัดข้อมูลทุกครั้งที่มีอักขระ “|”

3. ข้อมูลที่รับมาจะเป็น String ต้องแปลงเป็นตัวเลขก่อนจึงจะนำไปคำนวณต่อได้ โดยการใช้ .toInt เพื่อแปลงเป็นจำนวนเต็ม และ .toFlaot แปลงเป็นจำนวนทศนิยม

4. แสดงค่าทั้งหมดที่หน้า Serial เพื่อดูค่าการรับและการส่งทั้งหมด

3.5.6 โค้ด Task ของ NETPIE ในโค้ดรวมโดยเขียนแบบ Multitasking

Multi_test1 | FlowTask.h | LightTask.h | NETPIEtask.h | OTAtask.h | PresTask.h | PumpTask.h

```
/*........ connect to NETPIE with microgear function......... */
#include <MicroGear.h>
MicroGear microgear(client2);
#define ALIAS   "[redacted]" //name of controller
#define APPID   "[redacted]"  //"YOUR_APPID"
#define KEY     "[redacted]"  //"YOUR_KEY"
#define SECRET  "[redacted]"  //"YOUR_SECRET"
#define FEEDID  "[redacted]"
#define APIKEY  "[redacted]"
#define MAX_MOTOR 100
#define MAX_OUT 100
#define MAX_SET 100
#define INTERVAL 15000
#define T_INCREMENT 200
#define T_RECONNECT 5000
int timer = 0;
char str[32];
char str1[32];
char str2[32];

void onMsghandler(char *topic, uint8_t* msg, unsigned int msglen) {
    Serial.print("Incoming message -->");
    msg[msglen] = '\0';
    Serial.println((char *)msg);}

void onConnected(char *attribute, uint8_t* msg, unsigned int msglen){
    Serial.println("Connected to NETPIE...");
      microgear.setAlias(ALIAS);}
/*.................................................*/
class NETPIEtask : public Task {
private:
protected:

    void setup() {
      while (WiFi.status() != WL_CONNECTED) // รอการเชื่อมต่อ
  {
    delay(500);
    Serial.print(".");
  }
    Serial.println("");
    Serial.println("Wifi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP()); // แสดงหมายเลข IP
    microgear.on(MESSAGE,onMsghandler);
    microgear.on(CONNECTED,onConnected);
    microgear.init(KEY,SECRET,ALIAS);
    microgear.connect(APPID);
    }
```

1

1. กำหนดตัวแปร Array เพิ่มเป็น 3 ตัว เพื่อใช้รองรับข้อมูล 3 ชุด

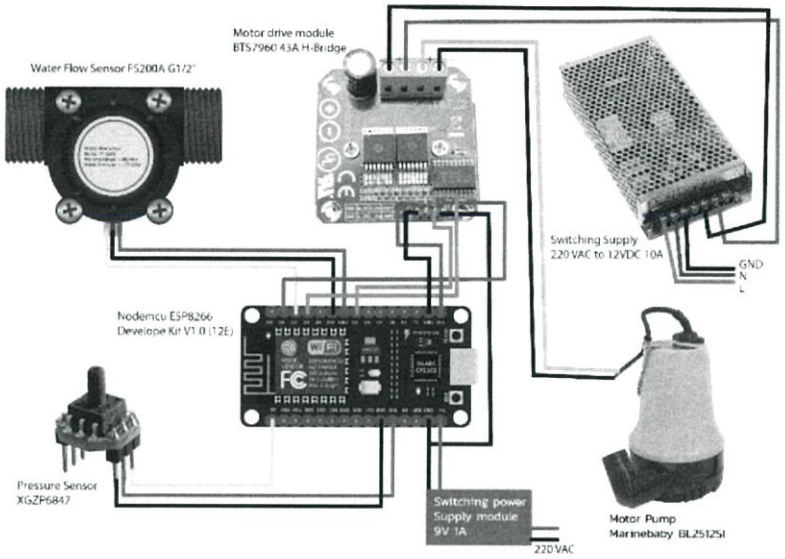
```
49    void loop() {
50      if (microgear.connected()) {
51        Serial.println("connected");
52          microgear.loop();
53      if (timer >= INTERVAL) {
54          sprintf(str1,"%.2f,%.2f,%d,%.2f",motorSpeedDUT,Setpoint,OutputValue,Calc);
55          sprintf(str2,"%.3f,%.3f,%.3f",Kp,Ti,Td);
56          microgear.publish("/node1/plant",str1);
57          microgear.publish("/node1/para",str2);
58
59            String data = "{\"Motor\":";
60            data += motorSpeedDUT;
61            data += ", \"Output\":";
62            data += OutputValue ;
63            data += ", \"Setpoint\":";
64            data += Setpoint ;
65            data += "}";
66            if (isnan(motorSpeedDUT) || isnan(OutputValue) || isnan(Setpoint)
67            || motorSpeedDUT >= MAX_MOTOR
68            || OutputValue >= MAX_OUT
69            || Setpoint >= MAX_SET)
70            {
71              Serial.println("Failed to read from DHT sensor!");
72            }else{
73              Serial.print("Sending -->");
74              Serial.println((char*) data.c_str());
75              microgear.writeFeed(FEEDID,data); //YOUR  FEED ID, API KEY
76            }
77           timer = 0;
78        }
79    else timer += T_INCREMENT;
80    }
81    else {
82        Serial.println("connection lost, reconnect...");
83        if (timer >= T_RECONNECT) {
84            microgear.connect(APPID);
85            timer = 0;
86        }
87        else timer += T_INCREMENT;
88    }
89
90  delay(200);
91    }
92
93  };
```

2. ข้อมูลตัวแรกรับค่า Motor speed, Setpoint, Output และอัตรการไหลของ Flow sensor ส่วนข้อมูลชุดสองรับค่า Kp, Ti และ Td ที่รับมาจาก LabVIEW

3. ข้อมูลตัวแรกส่งค่าโดยกำหนด Topic ชื่อ plant ส่วนข้อมูลชุดที่ 2 ส่งค่าโดยกำหนด Topic ชื่อ para

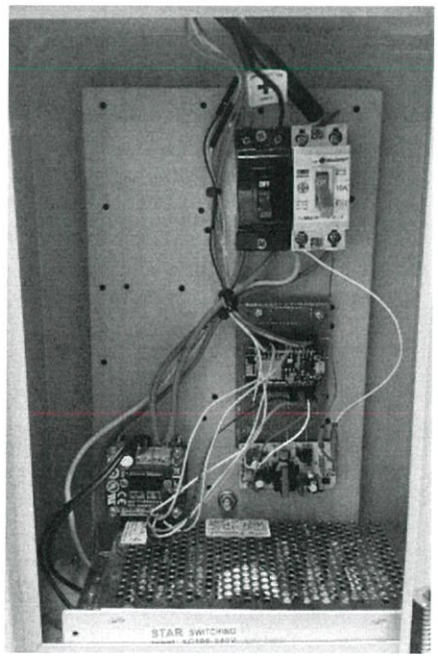
4. ข้อมูลชุดที่ 3 นำค่า Motor speed, Setpoint และระดับน้ำ ไปพลอตเป็นกราฟที่หน้า Feed

3.6 การออกแบบตู้วงจรควบคุม



รูปที่ 3.39 การต่ออุปกรณ์ทั้งหมด

จากการทดสอบและทดลองใช้งานอุปกรณ์ต่างๆ และทดสอบการส่งและรับค่าจาก Server แล้ว ขั้นตอนสุดท้ายคือ การวางระบบการเชื่อมต่อและแผงวงจรลงในตู้ควบคุมที่ติดอยู่กับตัวชิ้นงาน ได้ทำการออกแบบการเชื่อมต่อระหว่างอุปกรณ์เซนเซอร์ และตัวบอร์ดควบคุมพร้อมทั้งแหล่งจ่ายไฟให้อยู่ภายตู้เดียวกันได้ดังนี้ จะเห็นว่าบอร์ด NodeMCU เพียงตัวเดียวก็เพียงพอต่อการใช้งานร่วมกับอุปกรณ์ทั้งหมดแล้ว และยังสามารถเชื่อมต่อกับ Server ได้พร้อมกัน 2 Server จากการเขียนแบบ Multitasking ทำให้ประหยัดการใช้อุปกรณ์และการใช้สายลงไปได้มาก ทำให้พื้นที่ภายในสามารถเพิ่มเติมฟังก์ชันอื่นได้ในอนาคต



รูปที่ 3.40 ภายในตู้ควบคุม

บทที่ 4

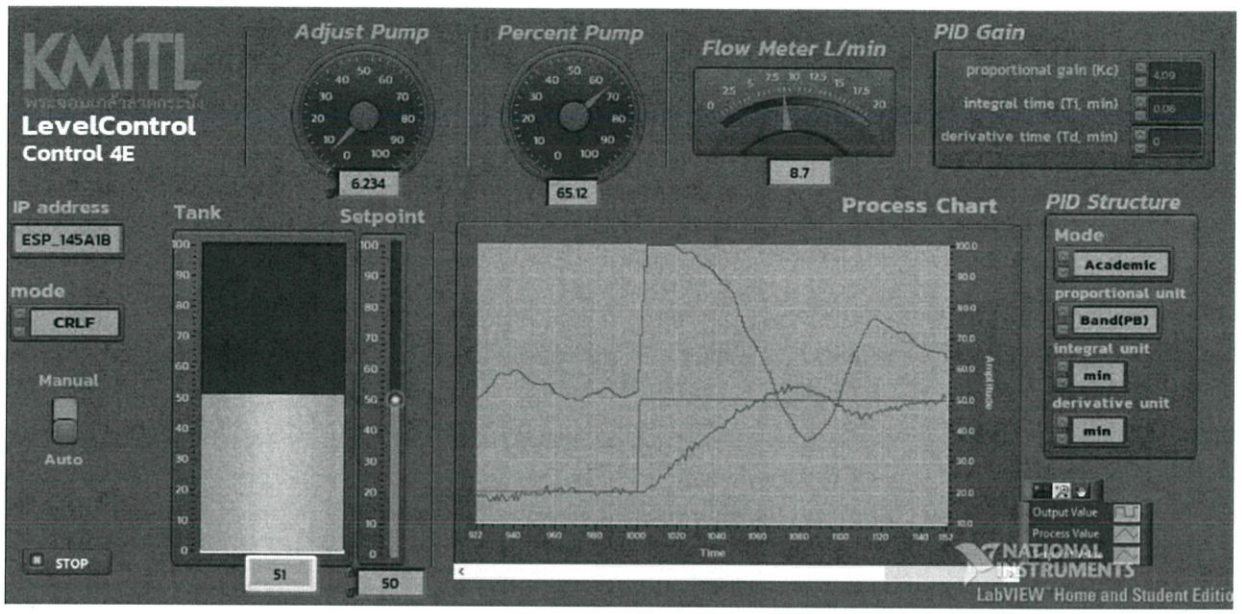
ผลการดำเนินงาน

4.1 ผลการดำเนินงานในส่วนของการเลือกอุปกรณ์และการทดสอบอุปกรณ์

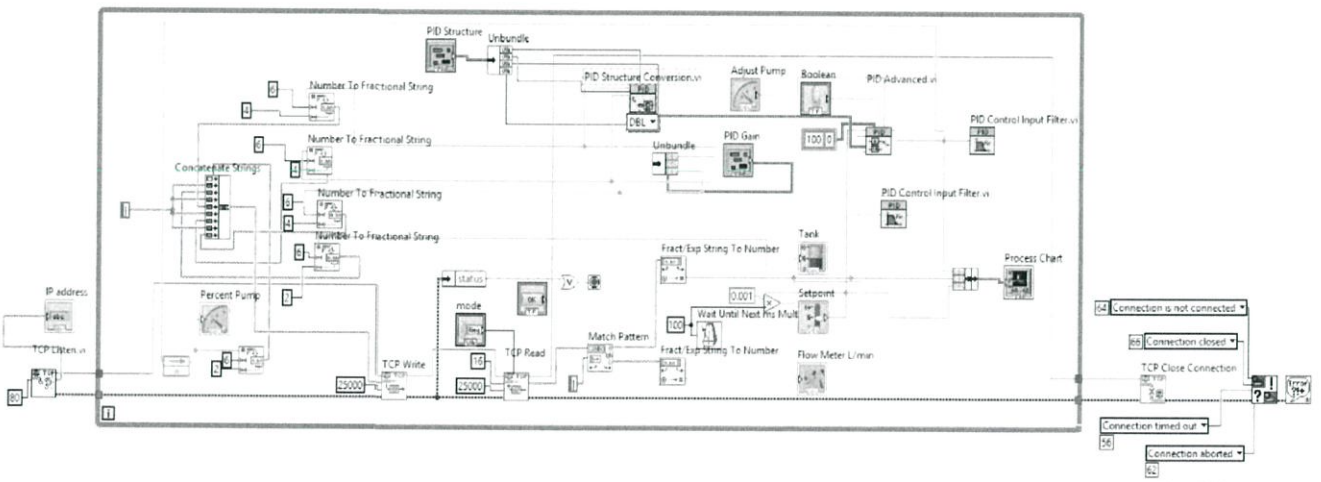
ในส่วนของการเลือกใช้อุปกรณ์ถือเป็นขั้นตอนแรกการทำโครงงาน และเป็นขั้นตอนที่สำคัญมาก เพราะหากการเลือกใช้อุปกรณ์ที่ไม่เหมาะสมในการทำงาน จะทำให้การทำงานยุ่งยากและต้องเสียเวลาในการปรับแก้และหลังจากเลือกอุปกรณ์ให้เหมาะสมกับการทำงานแล้ว จะต้องทำการทดสอบการใช้งานก่อนนำไปใช้จริง เพื่อป้องกันความผิดพลาดของอุปกรณ์ในระหว่างการทำงาน ป้องกันการเสียหายจากการใช้งานร่วมกับอุปกรณ์อื่นๆ และตรวจสอบการทำงานของอุปกรณ์ว่าเป็นไปตามข้อกำหนดของอุปกรณ์หรือไม่ หลังจากการลือกและทดสอบ พบว่าอุปกรณ์สามารถทำงานได้เพียงพอต่อความต้องการในโครงงานนี้ เนื่องจากเงื่อนไขทางด้านราคาและคุณภาพของอุปกรณ์ ทำให้ประสิทธิภาพของอุปกรณ์อาจจะไม่สูงมากนัก แต่ก็สามารถตอบสนองได้รวดเร็ว เพียงพอที่จะทำให้เห็นผลการทดลองจากการปรับค่าพารามิเตอร์ในแต่ละรูปแบบได้

4.2 ผลการดำเนินงานในส่วนของการติดตั้งโปรแกรมและการเขียน Function block

ในส่วนของการเลือกใช้โปรแกรมในการจำลอง Server เพื่อรับและส่งข้อมูล ระหว่างคอมพิวเตอร์และบอร์ดตัวควบคุม เลือกใช้ของโปรแกรม LabVIEW เนื่องจากสามารถเขียนคำสั่งโดยใช้ Function block แทนการเขียนโปรแกรม สามารถรับข้อมูลของเซนเซอร์ จากตัวควบคุมมาประมวลผลผ่านคำสั่ง Block ได้เลย ทำให้ประหยัดเวลาในการทำงาน และสามารถออกแบบหน้าจอในการมอนิเตอร์ได้ หลังจากติดตั้งโปรแกรมและทดลองเขียน Function block จำลอง Server แล้ว พบว่าโปรแกรมสามารถทำงานได้เป็นอย่างดี สามารถประมวลผลและแสดงออกมาในหน้ามอนิเตอร์ได้แบบ Real time สามารถเชื่อมต่อเพื่อรับค่าจากตัวควบคุมได้อย่างรวดเร็ว และมีเสถียรภาพเพียงพอต่อการใช้งานในระดับหนึ่ง ซึ่งทำให้งานใช้งานจริง สามารถเห็นความแตกต่างของการปรับค่าพารามิเตอร์อื่นๆ ได้เป็นอย่างดี แต่อาจจะมีปัญหาเรื่องสัญญาณรบกวนจากข้อมูลที่รับมาผ่านสัญญาณ Wi-Fi บ้างเพียงเล็กน้อย



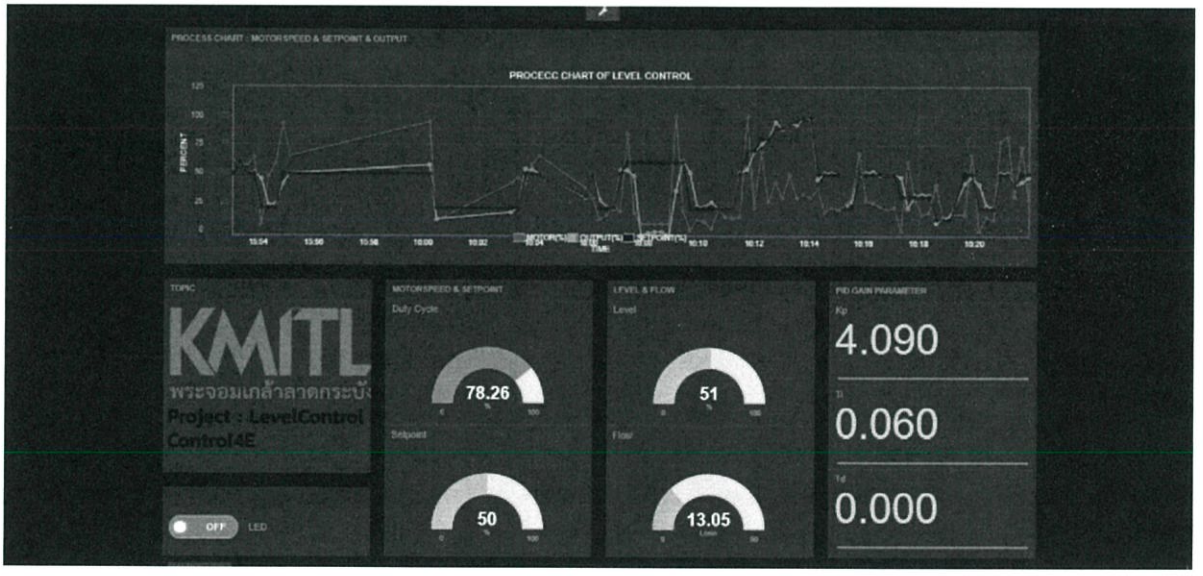
รูปที่ 4.1 หน้าจอมอนิเตอร์ที่ออกแบบด้วยโปรแกรม LabVIEW



รูปที่ 4.2 Function block ทั้งหมดที่เขียนด้วยโปรแกรม LabVIEW

4.3 ผลการทดสอบในการรับส่งค่าผ่าน Cloud server

ในส่วนของการรับส่งข้อมูลผ่านระบบ Cloud server นั้นได้เลือกใช้ NETPIE ซึ่งพัฒนาโดย NECTEC และเป็น Free platform ให้เปิดให้ผู้พัฒนาใช้งานได้โดยไม่เสียค่าใช้จ่าย และมีคู่มือการใช้งานเป็นภาษาไทย ทำให้ง่ายต่อการเรียนรู้และพัฒนาต่อยอดมาปรับใช้กับโครงงาน ซึ่งหลังจากทดลองศึกษาและใช้งานโดยรับและส่งข้อมูลจากบอร์ดตัวควบคุมกับ Cloud server เพื่อดูเสถียรภาพในการรับส่งและความรวดเร็วในการตอบสนอง พบว่าระบบสามารถทำงานได้เป็นอย่างดี มีเสถียรภาพในการเชื่อมต่อสูง ไม่หลุดการเชื่อมต่อจากการใช้งานในระยะเวลานานๆ แต่ความเร็วในการรับส่งอาจจะมีการหน่วงเล็กน้อย เนื่องจากเป็นการส่งผ่านสัญญาณ Wi-Fi และข้อจำกัดของ Platform ในเรื่องการแสดงผลเป็นรูปแบบกราฟซึ่งไม่สามารถเก็บข้อมูลแบบ Real time ได้ แต่สามารถเก็บข้อมูลได้เร็วสุดที่ 15 วินาทีและเป็นระยะเวลาได้นานสูงสุดถึง 3 วัน แต่ถึงแม้จะมีข้อจำกัดเหล่านี้ การทำงานของระบบก็สามารถตอบสนองได้เพียงพอต่อการใช้งานและจำลองระบบการส่งข้อมูลและมอนิเตอร์จากระยะไกลแล้ว

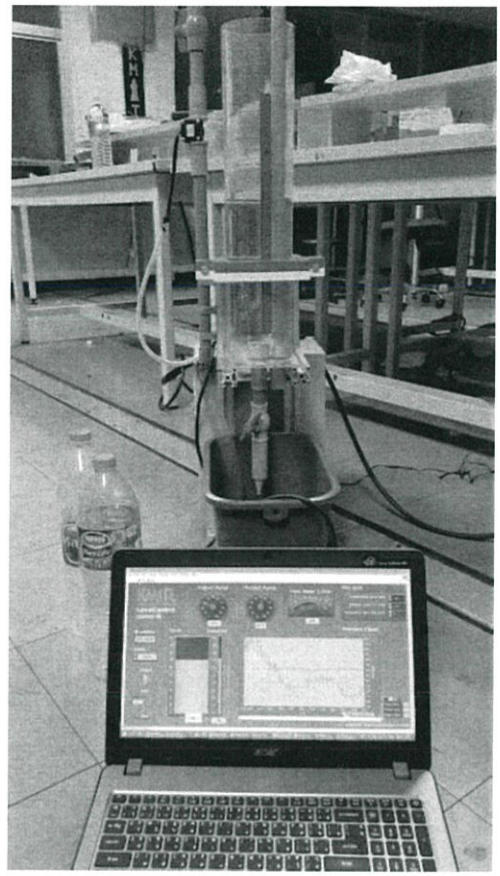


รูปที่ 4.3 หน้ามอนิเตอร์ของ NETPIE

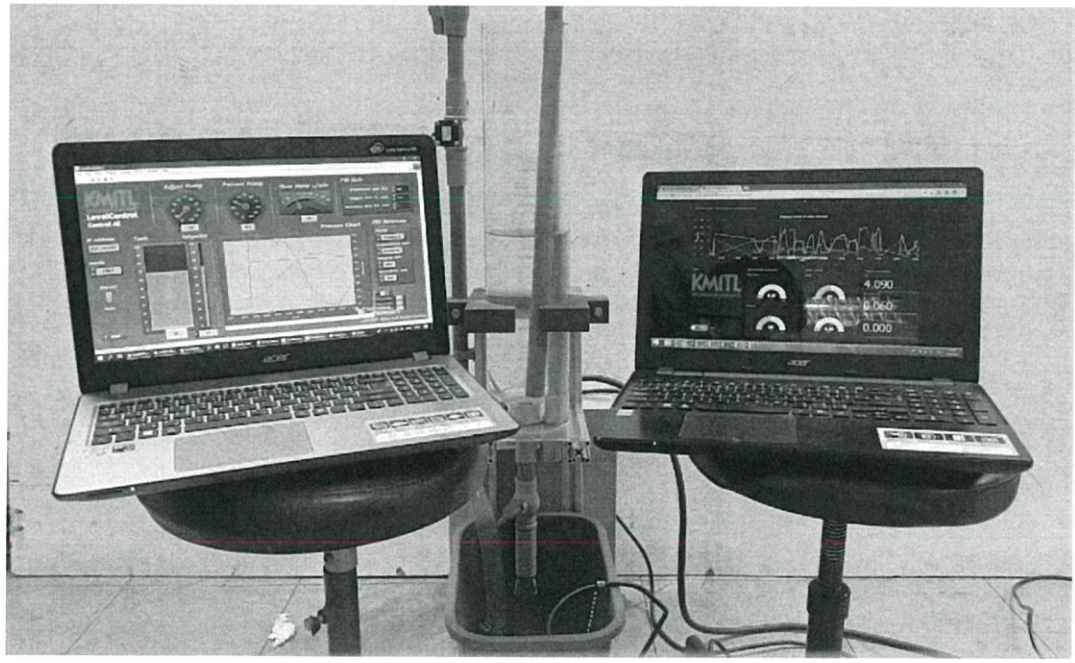
รูปที่ 4.4 หน้าแสดงผลกราฟของ NETPIE ในโหมด Feed

4.4 การรวมโค้ด

ในส่วนของการรวมโค้ด นั้นเป็นขั้นตอนสุดท้ายและเป็นขั้นตอนที่ยุ่งยากที่สุดของการทำโครงงาน เนื่องจากเซนเซอร์แต่ละตัวมีโค้ดการทำงานที่ต้องใช้การหน่วงเวลาที่แตกต่างกัน (Delay) และ Server NETPIE ก็มีข้อจำกัดในเรื่องการกำหนดค่าหน่วงเวลาไว้ที่ค่าหนึ่งซึ่งไม่สามารถปรับได้ ดังนั้นการรวมโค้ดโดยใช้วิธีปกติจึงไม่สามารถทำได้ เนื่องจากโปรแกรม Arduino IDE ที่ใช้ภาษา C ในการเขียน จะประมวลผลพร้อมกันทั้งโค้ด ทำให้การตั้งค่าหน่วงเวลาของแต่ละอุปกรณ์กระทบซึ่งกันและกัน และทำให้ความเร็วในการทำงานของอุปกรณ์และการส่งข้อมูลไปยัง Server ทั้ง LabVIEW และ NETPIE ช้าลงด้วย ดังนั้นจึงต้องใช้การเขียนแบบ Multitasking ซึ่งเป็นการเขียนการทำงานแยกออกเป็น Task ย่อยๆ แล้วดึงให้มาทำงานร่วมกัน ซึ่งการเขียนแบบ Multitasking นั้นการตั้งค่าการหน่วงเวลาของแต่ละอุปกรณ์จะแยกออกจากกันโดยสิ้นเชิงแต่สามารถทำงานร่วมกันได้ ด้วยเหตุนี้การรวมโค้ดของแต่ละอุปกรณ์ และการเชื่อมต่อ Server ทั้ง LabVIEW และ NETPIE สามารถทำได้โดยใช้บอร์ดตัวควบคุมเพียงแค่ตัวเดียว และจากการใช้งานจริงพบว่า ระบบสามารถทำงานประสานกันได้เป็นอย่างดี และสามารถทำงานให้เป็นไปตามวัตถุประสงค์ที่ตั้งไว้ได้ แต่จะมีปัญหาเล็กน้อยในส่วนของหน่วยความจำแบบ Global variable เนื่องจากการเขียนแบบ Multitasking มีการดึงตัวแปรข้าม Task เพื่อนำไปประมวลผลในส่วนอื่น ซึ่งการทำงานแบบนี้จะกินพื้นที่ในส่วนของ Global variable และเมื่อโค้ดที่ใช้มีจำนวนมากจนกินพื้นที่ไปเกิน 75% ของหน่วยความจำ บอร์ดตัวควบคุมจะแจ้งเตือนในเรื่องของเสถียรภาพที่อาจจะน้อยลง



รูปที่ 4.5 การทดสอบการชิ้นงานจริง



รูปที่ 4.6 การทดสอบการเชื่อมต่อและการใช้งาน

บทที่ 5

สรุปผลการดำเนินงานและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

เนื่องจากการทำงานในอุตสาหกรรมขนาดใหญ่ บางส่วนการผลิตอาจมีสารพิษหรืออันตรายจากการทำงาน การทำงานส่วนใหญ่จึงเป็นการควบคุมจากระยะไกลจากห้องควบคุม โดยการมอนิเตอร์หรือดูแนวโน้มจากข้อมูล แล้วนำมาวิเคราะห์หาสาเหตุของปัญหา โครงงานควบคุมระดับน้ำแบบไร้สายและการมอนิเตอร์ จึงถูกทำขึ้นเพื่อใช้จำลองระบบการทำงานในกระบวนการผลิตในอุตสาหกรรมขนาดใหญ่ ให้ใกล้เคียงกับระบบที่ใช้งานจริงมากที่สุด มีการส่งข้อมูลเพื่อดูค่าพารามิเตอร์ต่างๆ และการเก็บแนวโน้มข้อมูลเพื่อใช้ในการวิเคราะห์ปัญหา

จากผลการดำเนินงานในส่วนของการปรับปรุงที่ตัวเครื่องทางด้าน Hardware การทดสอบโปรแกรมของบอร์ดตัวควบคุม และการส่งข้อมูลขึ้นระบบ Cloud platform เพื่อการมอนิเตอร์จากระยะไกล สามารถใช้งานได้จริงตามที่คาดหวังไว้ สามารถทำงานได้อย่างมีประสิทธิภาพและสามารถประยุกต์ต่อยอดไปใช้ร่วมกับงานอื่นๆ ได้ในอนาคต

5.2 ปัญหาที่พบและแนวทางแก้ไข

5.2.1 ปัญหาที่พบ

1. เนื่องจากบอร์ด NodeMCU เป็นบอร์ดที่เพิ่งได้รับการพัฒนาให้รองรับภาษา C เมื่อไม่กี่ปีมานี้ ทำให้ Library บางตัวของโปรแกรม Arduino IDE ยังไม่รองรับ ตัวอย่างเช่น Library timer ที่ใช้ร่วมกับการอ่านค่า Analog จาก Pressure sensor

2. บอร์ด NodeMCU มีการประมวลผลทางลอจิกแบบ 10 bit (0-1023) ซึ่งโค้ดในการควบคุมมอเตอร์ปั๊มชุดเก่าเป็นแบบ 8 bit (0-255) ทำให้การทดสอบปั๊มครั้งแรกสูบน้ำไปยังแท้งน้ำไม่ขึ้น

3. การสูบน้ำเข้าถังในตอนแรกทำให้เกิดปัญหาน้ำกระเพื่อมซึ่งเกิดจากการกระแทกของน้ำกับก้นถัง ซึ่งมีระยะห่างพอสมควร ทำให้การวัดระดับน้ำด้วย Pressure sensor ที่ก้นถังคลาดเคลื่อน เป็นปัญหาต่อการควบคุมระดับน้ำ

4. การเขียนโค้ดควบคุมการทำงานให้บอร์ด NodeMCU โดยการรับและส่งค่าจาก LabVIEW ซึ่งเป็น Server และการส่งข้อมูลขึ้นระบบ Cloud platform ต้องผ่านระบบ TCP/IP โดยมีเราท์เตอร์เป็น

ตัวกลาง ซึ่งในการทดลองรวมโค้ดทั้งสองอย่างอยู่ในบอร์ดเดียวกัน ในตอนแรกไม่สามารถรวมกันได้เพราะมีปัญหาเรื่องการชนกันของข้อมูล เนื่องจากตัวบอร์ดเป็น Client เพียงตัวเดียวแต่ต้องเชื่อมต่อกับ Server 2 ตัวโดยใช้ IP ชุดเดียวกัน

5. Pressure sensor ที่ใช้งานมีปัญหาเกี่ยวกับการสเกลค่าระดับน้ำที่สูงกว่าระดับ 80% ของแท้งน้ำ ค่าที่อ่านได้จะให้ค่าคงที่ตลอด (Saturate) เนื่องจากเซนเซอร์ถูกใช้งานมาก่อนเป็นเวลานาน ทำให้เกิดการสึกหรอ อีกทั้งตัวเซนเซอร์เองก็มีราคาแพงและหาซื้อไม่ค่อยได้ในประเทศไทย

5.2.2 แนวทางแก้ไข

1. ใช้ฟังก์ชันอื่นในการอ่านค่า Analog แทน ใช้ฟังก์ชัน Ticker ในการ Interrupt ข้อมูลมาใช้งาน

2. ทำการเขียนโค้ดสเกลเปอร์เซ็นต์การทำงานของปั๊มใหม่ให้เป็นแบบ 10 bit ทำให้สามารถใช้งานได้เหมือนเดิม

3. ทำการต่อท่อให้ยาวขึ้นเพื่อให้ใกล้ก้นถัง เพื่อลดระยะห่างไม่ให้น้ำกระแทกกับก้นถังมากเกินไป ช่วยลดปัญหาการกระเพื่อมของน้ำได้ แต่การติดท่อยาวขึ้นจะทำให้เกิดเหตุการณ์ค้อนน้ำ (Water hammer) หรือการไหลย้อนกลับของน้ำได้เมื่อปั๊มทำงานน้อยลงหรือหยุดการทำงาน ทำให้เกิดการกระตุกของท่อส่งน้ำ และอาจส่งผลเสียต่อใบพัดของปั๊มน้ำอีกด้วย จึงได้ทำการติด Swing check valve แบบฝาทองเหลืองเพิ่มเพื่อป้องกันการไหลย้อนกลับของน้ำ

4. ทำการเขียนโค้ดเพิ่มเติมโดยการแบ่ง Client ออกเป็น Client ย่อยๆ เพื่อแบ่งการทำงานในการเชื่อมต่อ Server ออกเป็นสองวง ระหว่างบอร์ด Client กับ Server LabVIEW และบอร์ด Client กับ Server NETPIE

5. การซื้อเซนเซอร์ใหม่นั้นต้องใช้วลาในการจัดหาและสั่งซื้อ อาจทำให้การทำงานล่าช้า จึงใช้วิธีปรับสเกลแทนการซื้อ โดยการเลื่อนขีดสเกลบอกระดับน้ำบนตัวแท้งน้ำลงมาให้อยู่ใช่วงที่ Pressure sensor สามารถวัดค่าได้โดยไม่เกิดการ Saturate

5.3 ข้อเสนอแนะ

1. จากการศึกษาการเชื่อมต่อระหว่างบอร์ด NodeMCU กับ LabVIEW พบว่ามีแหล่งข้อมูลให้ศึกษาน้อย และส่วนใหญ่จะใช้สาย RS232 ในการเชื่อมต่อ แต่ใช้ระบบสื่อสารแบบไร้สาย เพื่อสะดวกในการเคลื่อนย้ายและทดลองระบบ ทำให้ต้องใช้เวลาในการศึกษาข้อมูลเพิ่มเติมทำให้เกิดความล่าช้าในการทำงาน

2. เนื่องจากปัจจุบันมีการพัฒนาทางด้าน Cloud platform ขึ้นมารองรับอุปกรณ์ที่สามารถเชื่อมต่อ Wi-Fi มากขึ้น ทำให้มีทางเลือกหลากหลายมากขึ้นในการเลือกใช้ แต่วิธีการทำงานใช้ส่วนใหญ่ยังให้ไม่มีในรูปแบบเป็นหนังสือการสอนภาษาไทย ทำให้ต้องศึกษาจากภาษาอังกฤษ และจากเว็บไซต์ต่างๆ ที่มีวีดีโอการสอนใช้งาน

3. เนื่องจากการทำงานจริงมีสัญญาณรบกวนเยอะพอสมควร ทั้งสัญญาณรบกวนจาก Wi-Fi อื่นๆ ที่มารบกวนการรับส่งข้อมูลและผลของการกระชากไฟจากการสั่งให้ปั๊มทำงานในรอบสูงๆ ดังนั้นการทำงานควรจะมีบอร์ด Sheild เพื่อป้องกันการกระชากของมอเตอร์และมีการเดินระบบสายไฟที่ดีกว่าโดยการลงแผ่นปริ้นเพื่อป้องกันการรบกวนจากสัญญาณภายนอก

เอกสารอ้างอิง

[1] **"มาตรฐาน IEEE"** เข้าถึงได้จาก: http://yie-it224.blogspot.com/ (วันที่สืบข้อมูล: 20 กันยายน 2560)

[2] **"เทคโนโลยี Wi-Fi"** เข้าถึงได้จาก: https://tech-noblogg.blogspot.com/2013/09/wi-fi_8896.html (วันที่สืบข้อมูล: 20 กันยายน 2560)

[3] **"สัญญาณ PWM"** เข้าถึงได้จาก: http://narong.ece.engr.tu.ac.th/ei444/document/11-Servo_motor_59.pdf (วันที่สืบข้อมูล: 21 กันยายน 2560)

[4] **"ค่าดิวตี้ไซเคิล"** เข้าถึงได้จาก: https://www.facebook.com/YETC.net/photos/a.282807968539073.1073741827.149611001858771/282808978538972/?type=3&Theater (วันที่สืบข้อมูล: 21 กันยายน 2560)

[5] **"การวัดอัตราการไหล"** เข้าถึงได้จาก: http://somsak.me.engr.tu.ac.th/download/flow%20rate.pdf (วันที่สืบข้อมูล: 25 กันยายน 2560)

[6] **"เครื่องมือวัดอัตราการไหล"** เข้าถึงได้จาก: http://www.foodnetworksolution.com/wiki/wordcap/turbine%20flow%20meter (วันที่สืบข้อมูล: 25 กันยายน 2560)

[7] **"การควบคุม"** การเข้าถึงได้จาก: http://eng.sut.ac.th/me/box/2_55/425440/Controller_Design2_55.pdf (วันที่สืบข้อมูล: 30 กันยายน 2560)

[8] **"การควบคุมแบบ PID"** เข้าถึงได้จาก: http://suchart.rmutl.ac.th/04-220-308/Control.pdf (วันที่สืบข้อมูล: 30 กันยายน 2560)

[9] **"MQTT"** เข้าถึงได้จาก: https://www.hivemq.com/blog/how-to-get-started-with-mqtt (วันที่สืบข้อมูล: 10 ตุลาคม 2560)

[10] **"มอเตอร์กระแสตรง"** เข้าถึงได้จาก: http://kpp.ac.th/elearning/elearning3/book-11.html (วันที่สืบข้อมูล: 10 ตุลาคม 2560)

[11] **"โปรแกรม LabVIEW เบื้องต้น"** เข้าถึงได้จาก: http://www.research-system.siam.edu/images/coop/DESIGN_AND_CONSTRUCTION_OF_ELECTRICAL_MEASUREMENT_USING_LABVIEW___PROGRAM/ch2.pdf (วันที่สืบข้อมูล: 11 ตุลาคม 2560)

[12] **"ESP8266"** เข้าถึงได้จาก: http://narong.ece.engr.tu.ac.th/ei444/document/ESP8266.pdf (วันที่สืบข้อมูล: 11 ตุลาคม 2560)

[13] **"Arduino IDE"** เข้าถึงได้จาก: http://www.sbt.ac.th/new/sites/default/files/TNP_Unit_2.pdf (วันที่สืบข้อมูล: 12 ตุลาคม 2560)

เอกสารอ้างอิง(ต่อ)

[14] **"เซนเซอร์ XGZ6847"** เข้าถึงถึงได้จาก: https://www.aliexpress.com/item/2pcs-0-40kPa-electronic-sphygmomanometer-sensor-XGZP6847-pressure-transmitter-module-0-5V-gas-pressure-sensor/32711968737.html (วันที่สืบข้อมูล: 15 ตุลาคม 2560)

[15] **"หลักการทำงานของสเตรนเกจ"** เข้าถึงถึงได้จาก: http://www.kyowa-ei.co.th/tha/-technical/strain_gages/index.html (วันที่สืบข้อมูล: 15 ตุลาคม 2560)

[16] **"High-Power Motor Drive"** เข้าถึงได้จาก: https://www.arduitronics.com/product/983/motor-drive-module-bts7960-43a-with-h-bridge (วันที่สืบข้อมูล: 18 ตุลาคม 2560)

[17] **"ปั๊มน้ำ DC 12V"** เข้าถึงได้จาก: http://www.ปั๊มน้ำดีซี.com/ปั๊มน้ำDC-12V/ปั๊มน้ำDC-12V-BL2512SI (วันที่สืบข้อมูล: 23 ตุลาคม 2560)

[18] **"NETPIE"** เข้าถึงได้จาก: https://netpie.gitbooks.io/nodemcu-esp8266-on-netpie/content/chapter1.html (วันที่สืบข้อมูล: 23 ตุลาคม 2560)

[19] **"เทคโนโลยี MQTT"** เข้าถึงได้จาก: https://medium.com/@tanakornpiamsin/ติดตั้ง-mqtt-server-d31bcae85d0d (วันที่สืบข้อมูล: 28 ตุลาคม 2560)

[20]**"การเชื่อมต่อแบบ TCP/IP บน LabVIEW"** เข้าถึงได้จาก: http://www.ni.com/white-paper/2710/en/ (วันที่สืบข้อมูล: 30 ตุลาคม 2560)

[21]**"การติดตั้งและการลง Library เพื่อรองรับ NodeMCU ESP8266 ของโปรแกรม Arduino IDE"** เข้าถึงได้จาก: https://thaieasyelec.com/article-wiki/embedded-electronics-application/getting-started-with-esp8266-nodemcu.html (วันที่สืบข้อมูล: 5 พฤศจิกายน 2560)

[22]**"คู่มือการติดตั้งและใช้งาน VI บน LabVIEW"** เข้าถึงได้จาก: http://ftp.qwavesys.com/repository/EspVIEW/User%20Manual%20-%20ESP8266%20 (LabVIEW%20Version)%20-Q-Wave%20Systems%201.0.4.pdf (วันที่สืบข้อมูล: 6 พฤศจิกายน 2560)

[23]**"NodeMCU ESP8266 เชื่อมต่อกับ NETPIE เบื้องต้น"** เข้าถึงได้จาก: https://medium.com/@zCaesar/มือใหม่หัดรู้จัก-iot-review-การใช้งาน-netpie-anto-และ-aws-iot- dbd0680af696(วันที่สืบข้อมูล: 8 พฤศจิกายน 2560)

[24]**"NETPIE: Internet of Things"** เข้าถึงได้จาก: ttps://www.nectec.or.th/innovation/innovation-software/netpie.html (วันที่สืบข้อมูล: 9 พฤศจิกายน 2560)

เอกสารอ้างอิง(ต่อ)

[25]"**ตัวอย่างโปรแกรมการทดสอบ Flow sensor**" เข้าถึงได้จาก: https://www. 9arduino.com /article/69/สอนใช้งาน-water-flow-sensor-กับ-arduino (วันที่สืบข้อมูล: 15 พฤศจิกายน 2560)

[26]"**บทความการใช้งานเริ่มต้น ESP8266 NodeMCU**" เข้าถึงได้จาก: https://www.thaieasyelec .com/article-wiki/embedded-electronics-application/getting-started-with-esp8266-nodemcu-ch5.html (วันที่สืบข้อมูล: 10 มกราคม 2560)

[27]"**วิธีเชื่อมต่อ NodeMCU เข้ากับระบบ Network**" เข้าถึงได้จาก: https://thaieasyelec.com /article-wiki/embedded-electronics-application/getting-started-with-esp8266-nodemcu-ch2.html (วันที่สืบข้อมูล: 10 มกราคม 2560)

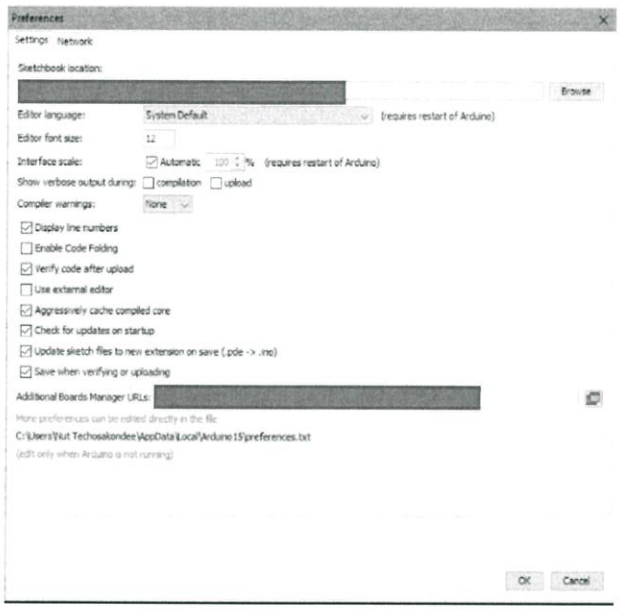
[28]"**PID structure**" เข้าถึงได้จาก: http://www.acsysteme.com/en/serial-or-parallel-pid (วันที่สืบข้อมูล: 10 มกราคม 2560)

ภาคผนวก

ภาคผนวก ก

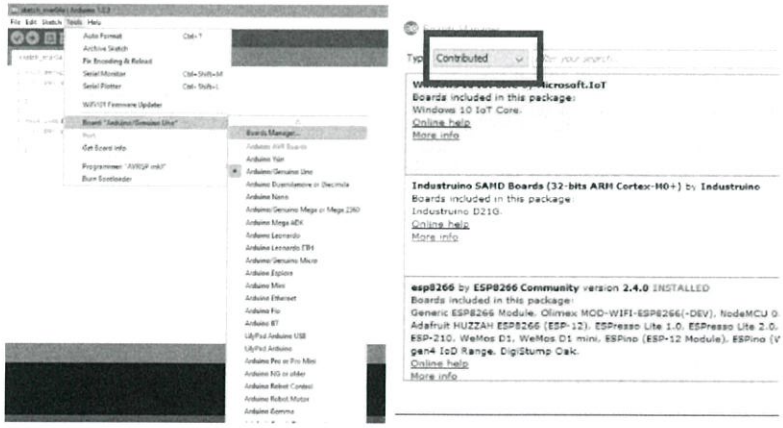
การติดตั้งโปแกรมที่เกี่ยวข้อง

ก1 การติดตั้งและการลง Library เพื่อรองรับ NodeMCU ESP8266 ของโปรแกรม Arduino IDE



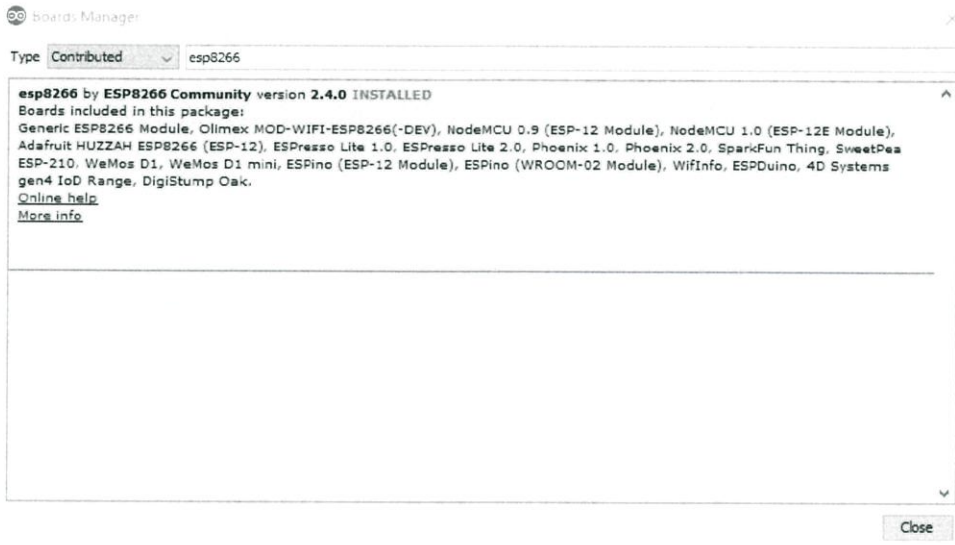
รูปที่ ก1.1 การตั้งค่า Preference

- ไปที่ File >> Preferences เพื่อกำหนดค่าเริ่มต้นในการติดตั้งบอร์ด และใส่ URL ลงใน Addition Board Manage เป็น http://arduino.esp8266.com//stable/ package_esp8266com_index.json



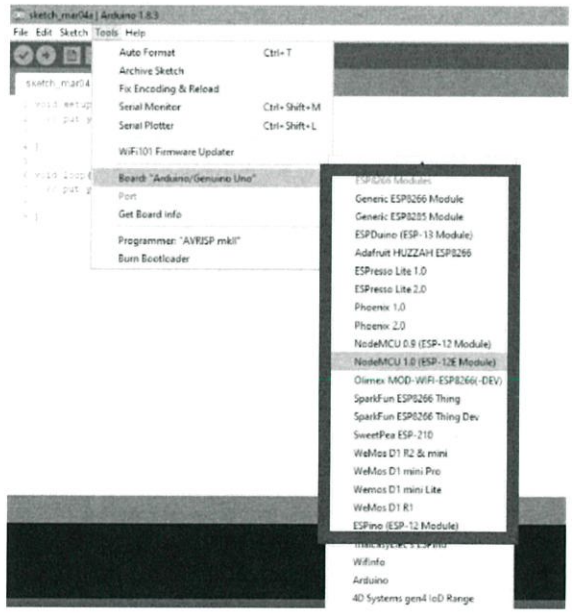
รูปที่ ก1.2 การติดตั้ง Board (1)

- ไปที่ Tools >> Board >> Board Manager และเลือก Type เป็น Contributed



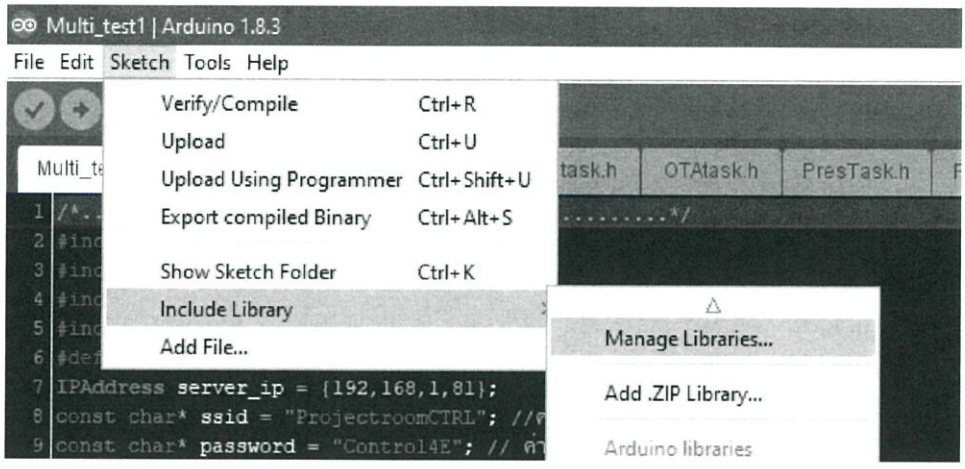
รูปที่ ก1.3 การติดตั้ง Board (2)

- พิมพ์ในช่องค้นหาว่า ESP8266 แล้วทำการติดตั้งโดยกด install เมื่อติดตั้งเสร็จแล้วจะขึ้นคำว่า installed ที่ด้านหลังของบอร์ด



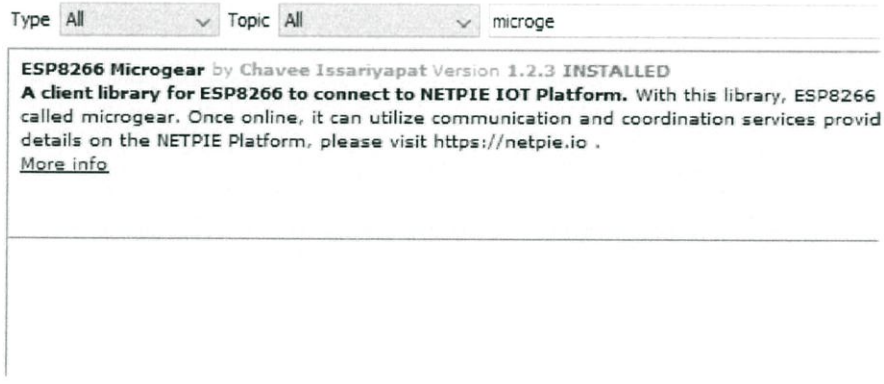
รูปที่ ก1.4 หลังจากติดตั้ง Board

- เมื่อติดตั้งบอร์ดเสร็จแล้ว สามารถตรวจสอบได้ที่ Tools >> Board เมื่อเลื่อนลงมาจะพบว่ามีบอร์ดเพิ่มเข้ามา โดยโปรเจคนี้จะใช้บอร์ดที่ชื่อ NodeMCU 1.0 (ESP-12E Module)



รูปที่ ก1.5 การติดตั้ง Library NETPIE (1)

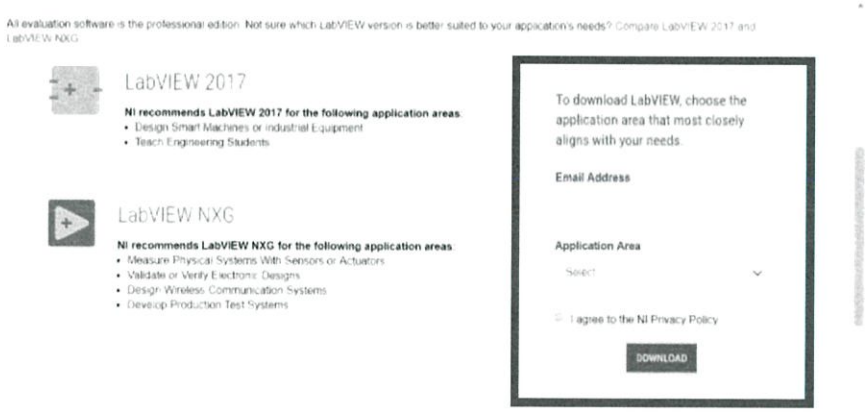
- ส่วนการใช้งาน NETPIE นั้นจะต้องติดตั้ง Library เพิ่มเติม เพื่อใช้ในการติดต่อสื่อสารกันได้ระหว่าง Cloud server กับบอร์ดตัวควบคุม โดยไปที่ Sketch >> Include Library >> Manage Libraries



รูปที่ ก1.6 การติดตั้ง Library NETPIE (2)

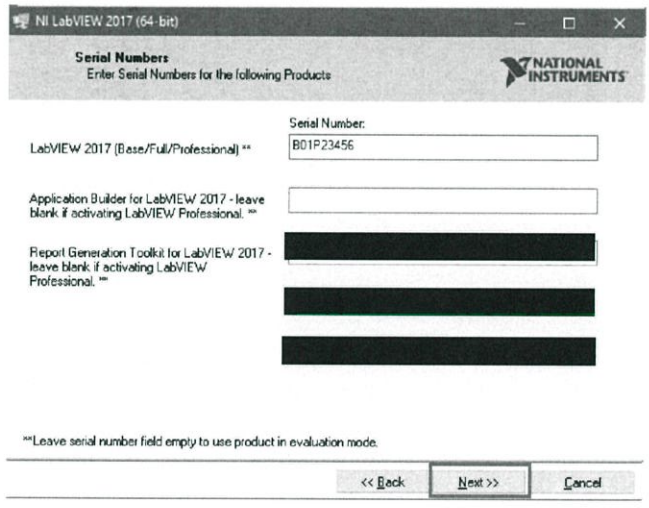
- จากนั้นพิมพ์คำว่า Microgear และทำการติดตั้ง เมื่อติดตั้งเสร็จแล้ว สามารถดูโค้ดได้จากโค้ดตัวอย่างของโปรแกรม

ก2 การติดตั้งโปรแกรม LabVIEW 2017 และติดตั้งส่วนเสริม



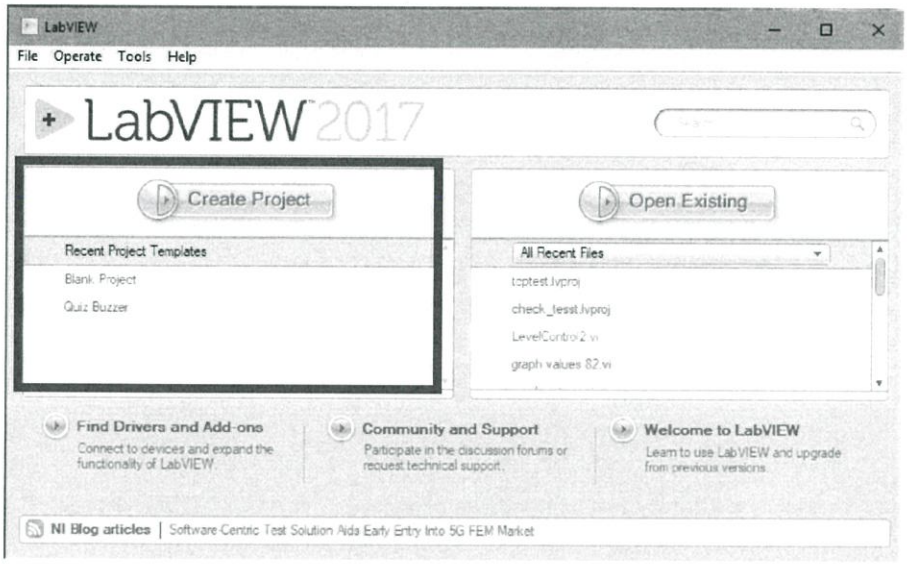
รูปที่ ก2.1 การติดตั้งโปรแกรม LabVIEW 2017 (1)

- ทำการ download โปรแกรมได้จากเว็บ http://www.ni.com/en- th/shop/labview/download.html โดยสมัครอีเมลและทำตามขั้นตอนในการติดตั้งให้เรียบร้อย สามารถดูวิธีการติดตั้งแบบละเอียดได้ที่ http://www.studica.com/blog/download-install-labview-2017



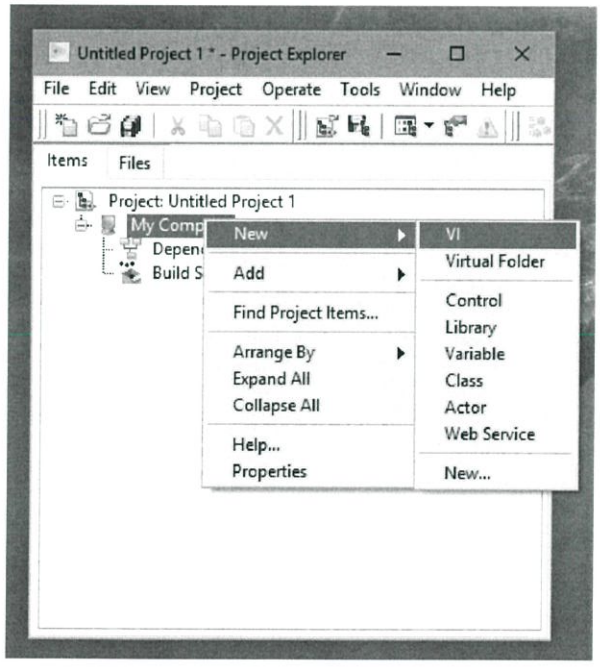
รูปที่ ก2.2 การติดตั้งโปรแกรม LabVIEW 2017 (2)

- ระหว่างการติดตั้งโปรแกรมจะการแจ้งให้ใส่ Serial number หากผู้ใช้ไม่ใส่ ตัวโปรแกรมจะเป็นเวอร์ชัน Trial สามารถใช้ได้ตามอายุเวลาที่ระบุไว้ แต่หากมี Serial number จะสามารถใช้ได้แบบไม่มีเวลาจำกัด



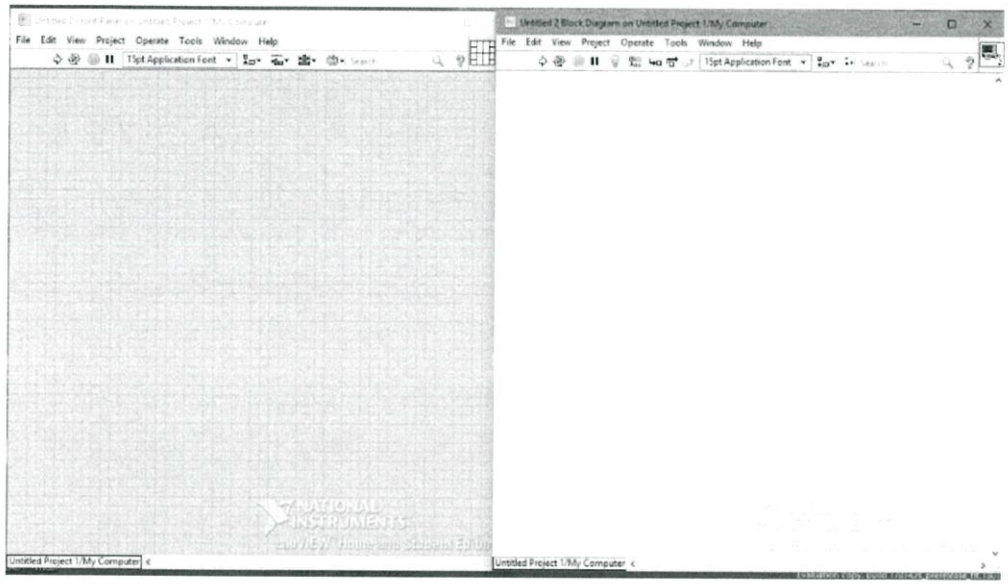
รูปที่ ก2.3 การเข้าใช้งานโปรแกรม LabVIEW

- เมื่อทำการติดตั้งเสร็จแล้ว ทำการสร้างโปรเจคใหม่ โดยกดไปที่ Create Project



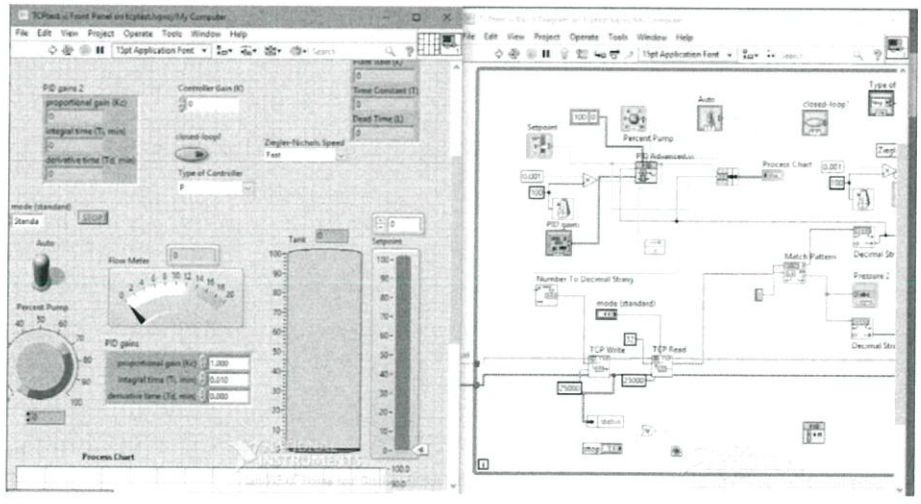
รูปที่ ก2.4 การสร้างโปรเจคใหม่

- จะมีหน้าต่าง Project Explorer ขึ้นมา คลิกขวาที่ My Computer >> New >> VI เพื่อสร้างไฟล์งานขึ้นมาใหม่



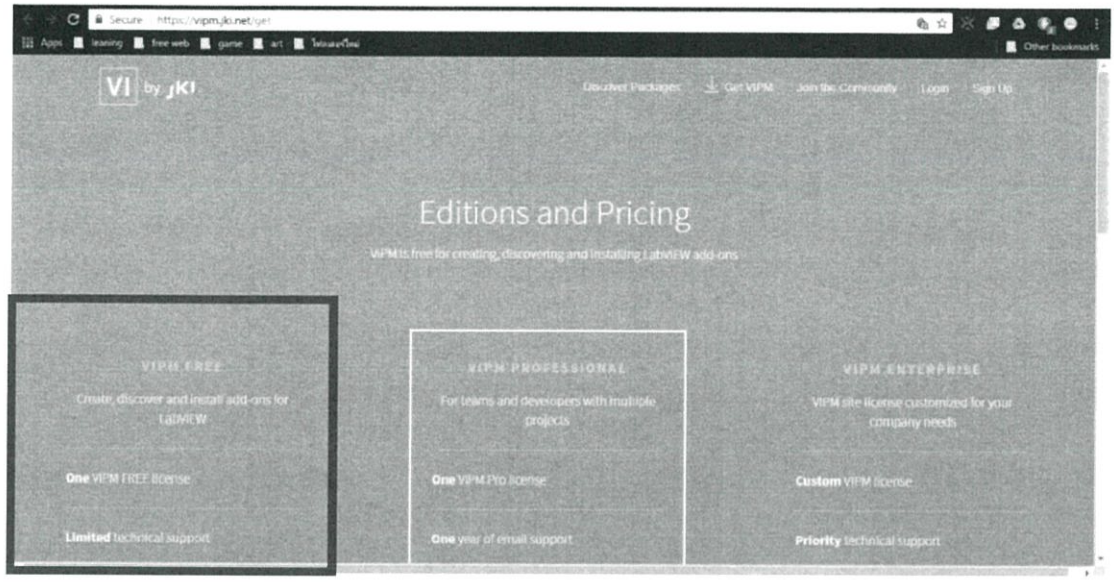
รูปที่ ก2.5 หน้าตาโปรแกรม LabVIEW

- เมื่อสร้างไฟล์งานขึ้นมาแล้ว จะมีหน้าต่างขึ้นมาสองอันคือ Front panel และ Block diagram สามารถกด Ctrl T เพื่อเปิดหน้าต่างพร้อมกันทั้ง 2 หน้า



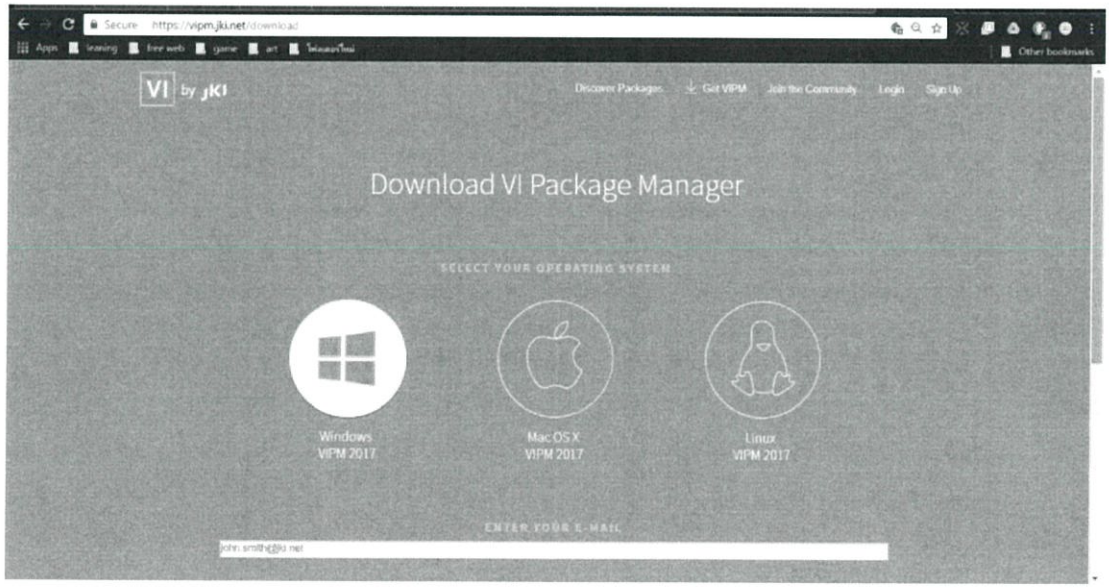
รูปที่ ก2.6 หน้าการใช้งานของโปรแกรม

- โดยการทำงานนั้น หน้าต่าง Front panel จะเป็นการใช้งานด้านการออกแบบหน้าจอแสดงผลและการมอนิเตอร์ ส่วนด้าน Block diagrams จะเป็นการเขียนฟังก์ชัน ควบคุมการทำงานและประมวลผลโดยใช้ Function block



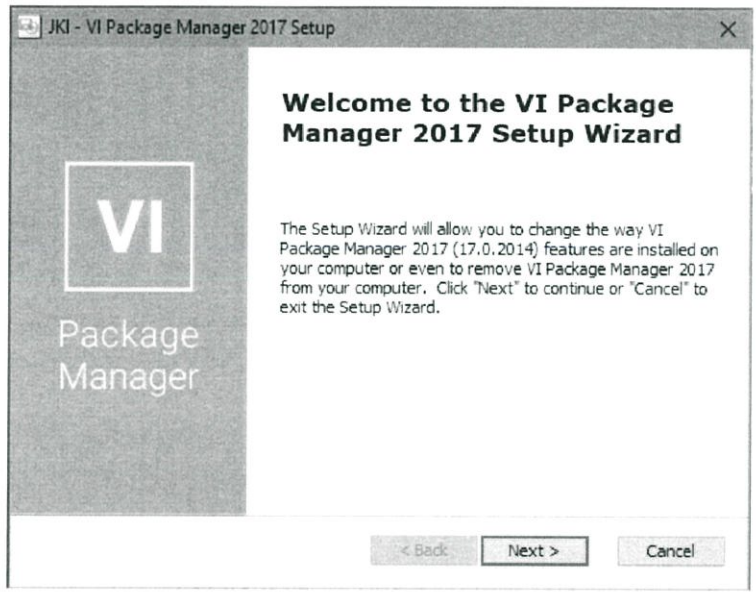
รูปที่ ก2.7 การติดตั้งส่วนเสริม (1)

- ดาวน์โหลดโปรแกรม VI Package Manager (VIPM) แล้วเลือกแบบ Free edition



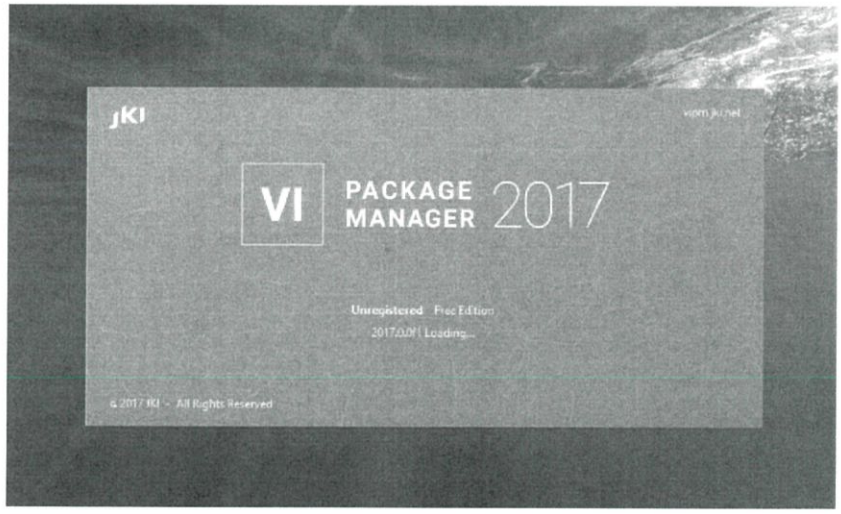
รูปที่ ก2.8 การติดตั้งส่วนเสริม (2)

- เลือกเวอร์ชันที่ตรงกับเครื่องคอมพิวเตอร์ และใส่อีเมลยืนยันเพื่อรับไฟล์ตัวลงโปรแกรม



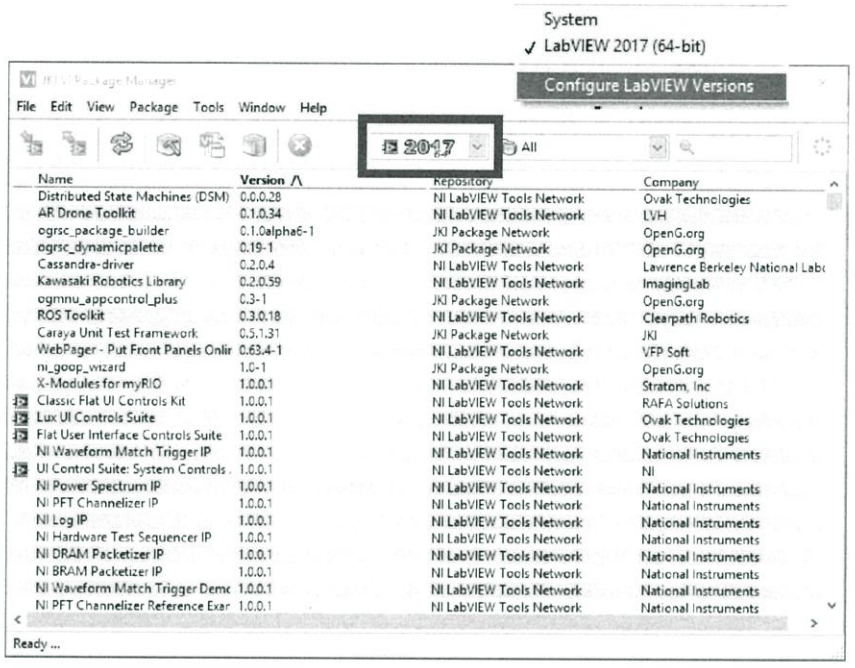
รูปที่ ก2.9 การติดตั้งส่วนเสริม (3)

- ทำการติดตั้งโปรแกรมตามปกติ และเนื่องจากเป็นเวอร์ชันฟรี จึงไม่ต้องใส่ Serial number



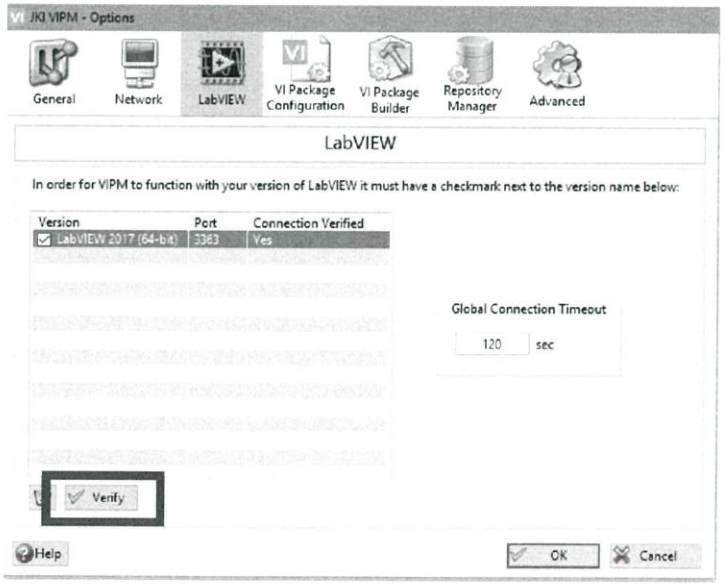
รูปที่ ก2.10 การติดตั้งส่วนเสริม (4)

- หลังจากติดตั้งโปรแกรมเสร็จแล้วเปิดโปแกรมขึ้นมา จะขึ้นหน้าต่างแสดงเวอร์ชั่นที่ติดตั้ง



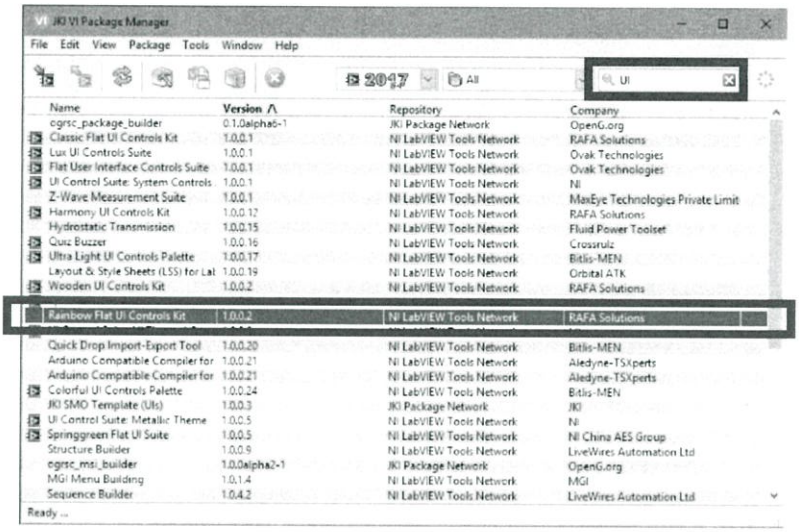
รูปที่ ก2.11 การตั้งค่า Configure เพื่อยืนยันรุ่นการใช้งาน

- ภายในโปรแกรมจะรวบรวม Add-on ต่างๆ ไว้เพื่อให้เรานำมาใช้ แต่ก่อนอื่นต้องทำการตรวจสอบเวอร์ชันของโปรแกรมที่จะรองรับส่วนเสริมนี้ก่อน โดยคลิกที่รูปเวอร์ชัน >> Configure LabVIEW Versions



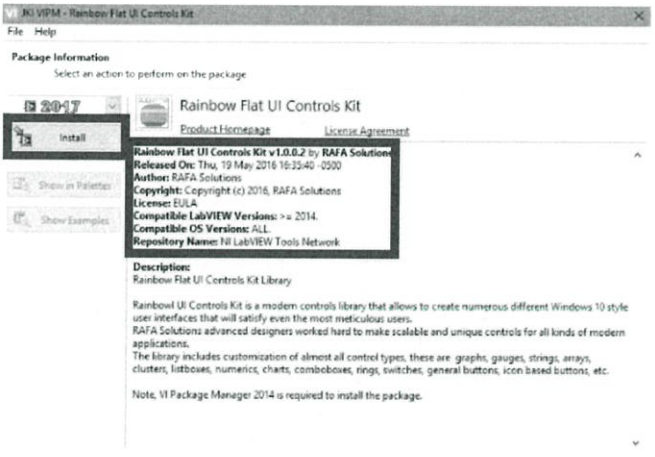
รูปที่ ก2.12 การ Verify รุ่นการใช้งาน

- ทำการยืนยันโดยกด Verify เมื่อยืนยันเสร็จแล้ว กด OK เพื่อออกจากหน้าต่าง



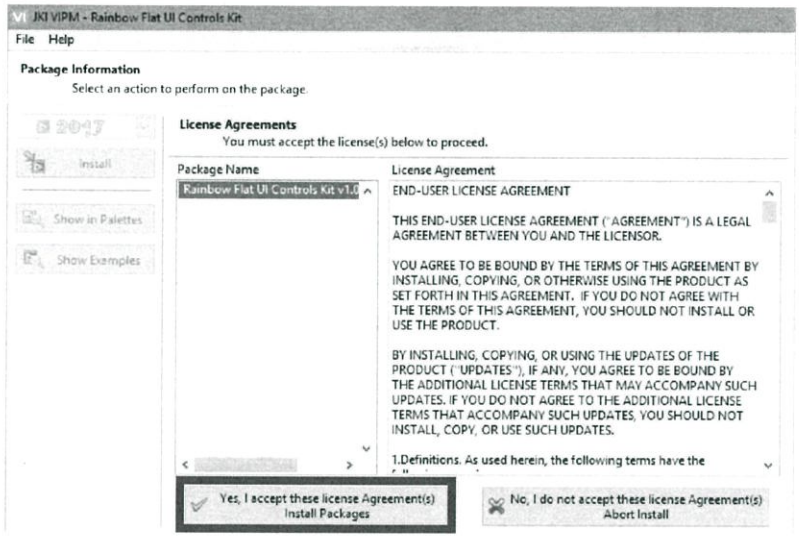
รูปที่ ก2.13 การติดตั้งส่วนเสริม UI Interface (1)

- จากนั้น พิมพ์หา Add-on ที่ต้องการในช่องค้นหา โดยโปรเจคนี้เราต้องการ Add-on สำหรับตกแต่งหน้าจอแสดงผลเพื่อใช้ในการมอนิเตอร์ จึงค้นหาคำที่เกี่ยวข้องกับคำว่า UI โปรแกรมก็จะนำชื่อที่เกี่ยวข้องมาให้ จากนั้นทำการเลือก Add-on โดยผู้จัดทำเลือก Rainbow flat UI controls kit



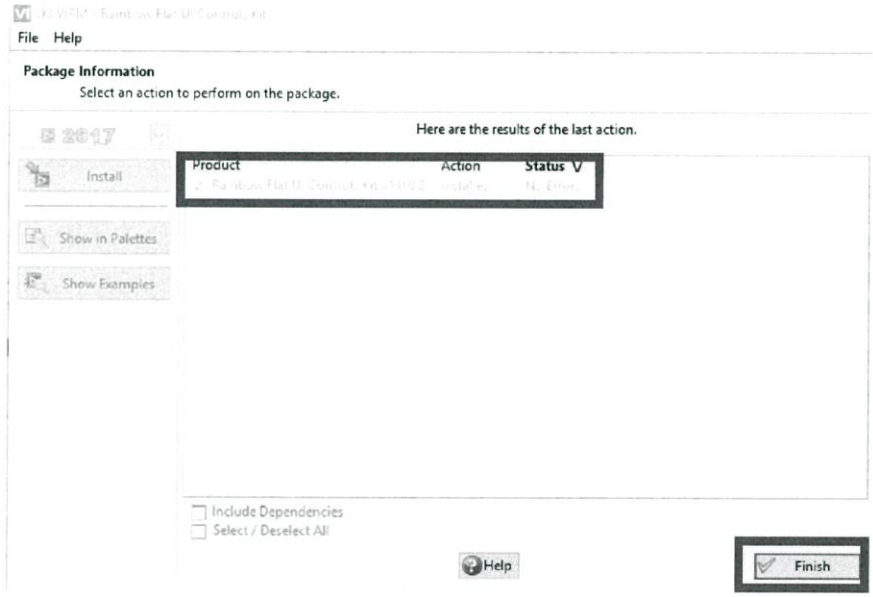
รูปที่ ก2.14 การติดตั้งส่วนเสริม UI Interface (2)

- เมื่อทำการคลิกเข้ามา จะมีข้อมูลเกี่ยวกับ Add-on ระบุอยู่ บางตัวอาจเป็นเวอร์ชัน Trial ใช้ได้เวลาจำกัด ในกรณีที่ต้องการใช้งานแบบไม่จำกัดเวลาจะต้องเสียเงินเพิ่ม หรือบางตัวเป็นแบบฟรี จากนั้นกด Install เพื่อลง Add-on



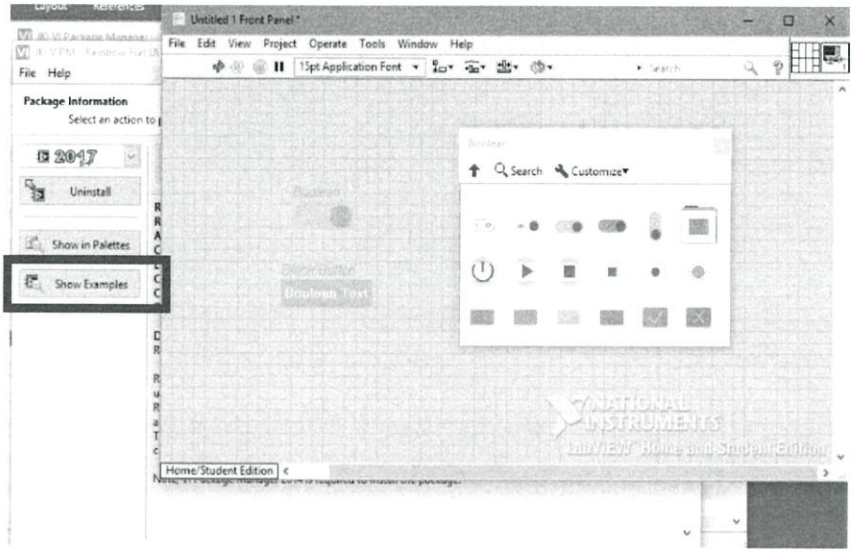
รูปที่ ก2.15 การติดตั้งส่วนเสริม UI Interface (3)

- จากนั้นจะมีการระบุ License ต่างๆ เป็นข้อบังคับ กด Yes เพื่อยืนยันการติดตั้ง



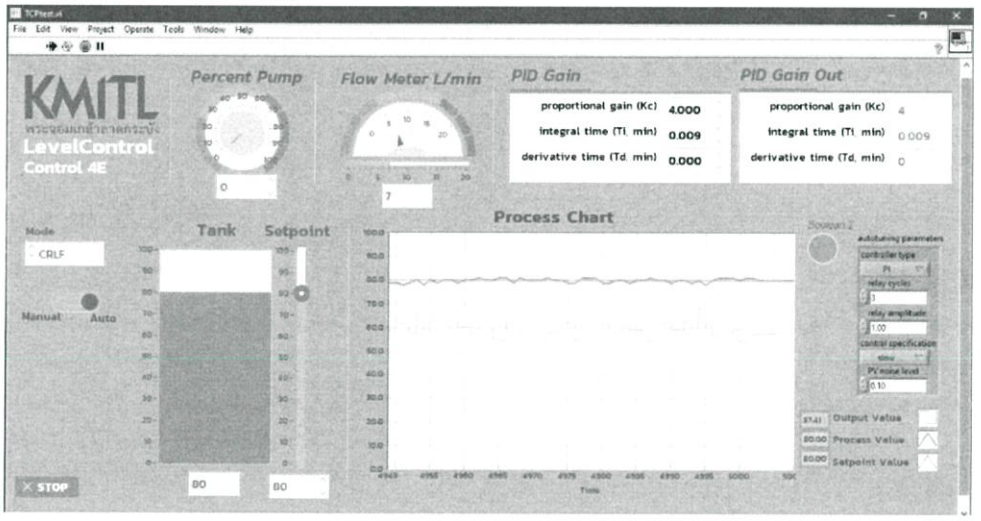
รูปที่ ก2.16 การติดตั้งส่วนเสริม UI Interface (4)

- เมื่อติดตั้งเสร็จแล้วจะขึ้นหน้านี้ เพื่อบอกว่าการติดตั้งเสร็จแล้ว กด Finish เพื่อออกจากหน้าต่าง



รูปที่ ก2.17 การทดลองใช้งานส่วนเสริม UI Interface

- จากนั้นทดลองใช้งาน Add-on โดยคลิกไปที่ Show Example จะขึ้นหน้าต่างให้ทดลองใช้ Add-on



รูปที่ ก2.18 การออกแบบหน้า Interface

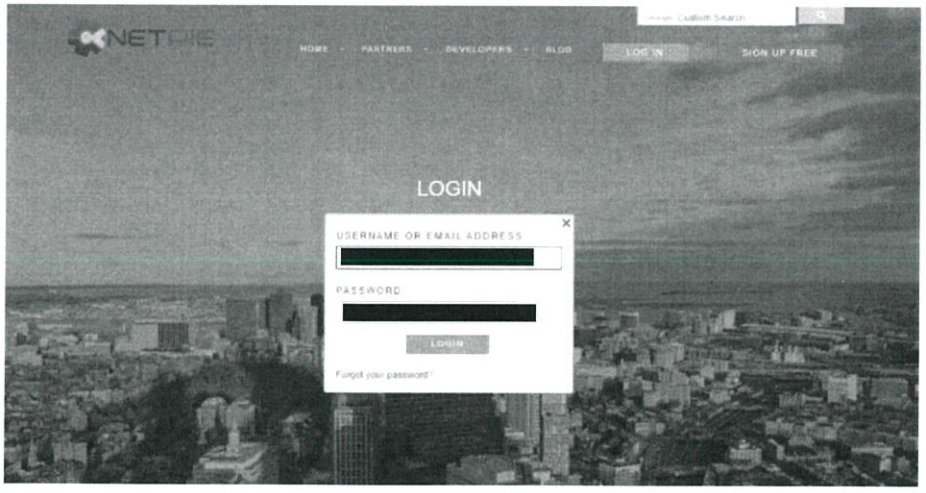
- เมื่อติดตั้งเสร็จแล้วสามารถนำ Add-on มาใช้กับตัวไฟล์โปรเจคได้ โดยการค้นหา Widget ในหน้า Front panel ด้วยชื่อจาก Add-on นั้นๆ

ก3 การสมัครเข้าใช้งาน NETPIE

ในส่วนของการใช้งานการส่งข้อมูลขึ้นระบบ Cloud server โดยใช้ NETPIE นั้น มีขั้นตอนดังนี้

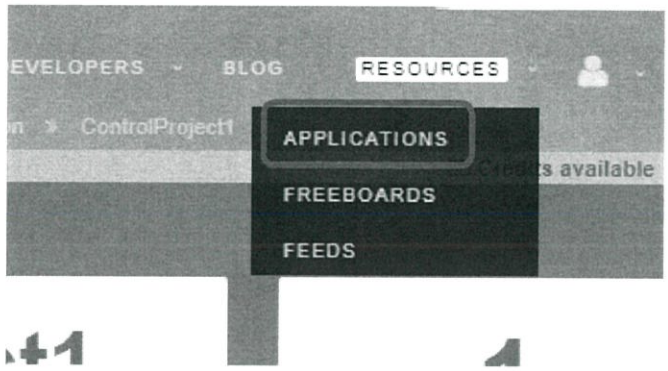
รูปที่ ก3.1 การสมัครเข้าใช้งาน NETPIE (1)

- ทำการสมัครสมาชิกโดยไปที่เว็บ https://netpie.io กด Sign up free และกรอกข้อมูลที่สำคัญให้เรียบร้อย ระบบจะทำการส่งอีเมลกลับมาเพื่อให้ยืนยันตัวตนอีกครั้งก่อนใช้งาน



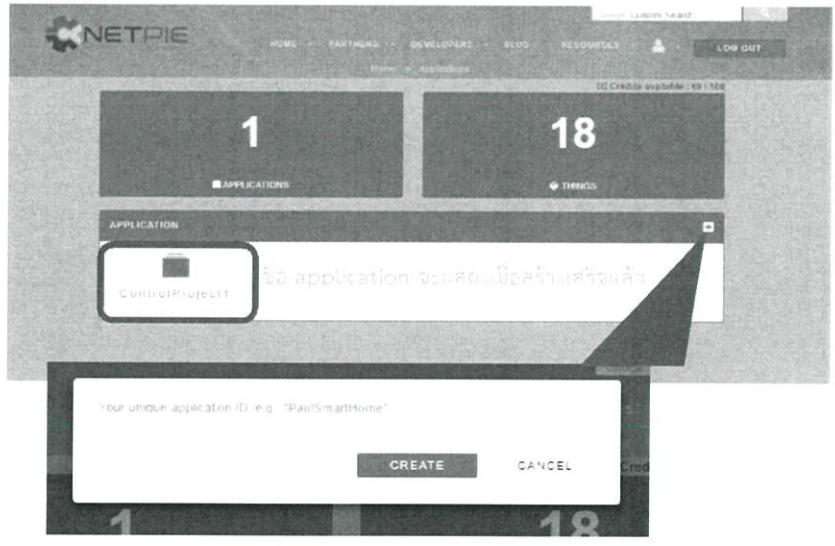
รูปที่ ก3.2 การสมัครเข้าใช้งาน NETPIE (2)

- เมื่อทำการยืนยันตัวตนเรียบร้อยแล้ว กลับมาที่หน้าเว็บอีกครั้ง กด Sign in เพื่อเข้าใช้งาน และกรอก Username และ Password ให้เรียบร้อย



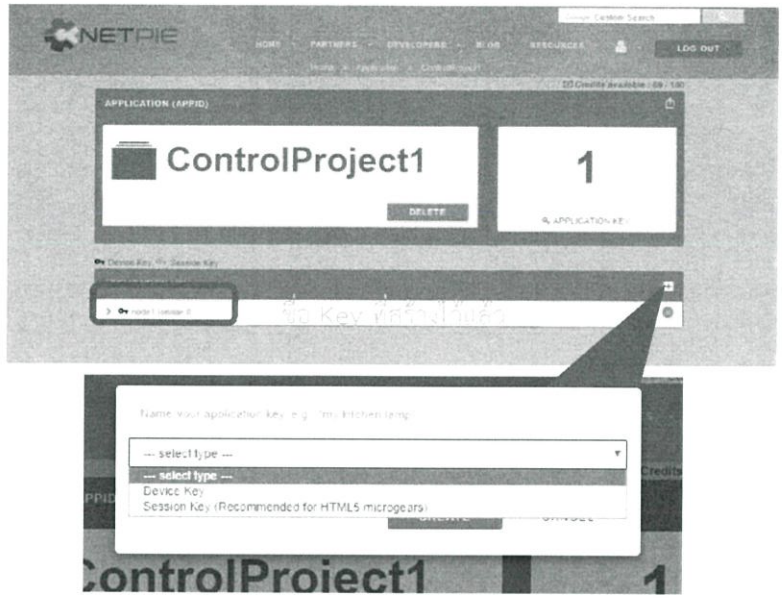
รูปที่ ก3.3 การใช้งานในส่วน Application

- เมื่อเข้าสู่หน้าใช้งาน ไปที่เมนู Resources จะมีโหมดการใช้งานให้เลือกใช้ 3 หน้า คือ Applications Freeboards และ Feeds โดยขั้นแรกต้องไปกำหนดค่าเริ่มต้นที่หน้า Applications ก่อน



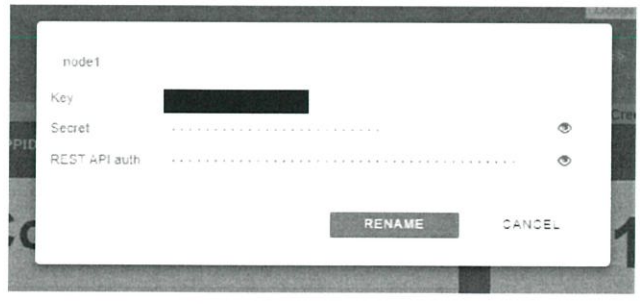
รูปที่ ก3.4 การสร้าง Application ก่อนการใช้งาน

- ที่โหมด Applications เป็นส่วนที่ใช้กำหนดค่าตั้งต้นของ Application ที่เราจะใช้งานในส่วนอื่นๆต่อไป โดยเริ่มแรก ให้กดไปที่เครื่องหมาย + และสร้างชื่อ Application ของเรา ซึ่งจะต้องไม่ซ้ำกับชื่ออื่น เมื่อสร้างเสร็จแล้ว ก็จะมีชื่อ Application แสดงอยู่ในแถบ



รูปที่ ก3.5 การสร้าง Key และเลือกประเภท

- จากนั้นทำการคลิกเข้าไปใน Application จะปรากฏหน้าที่ใช้สร้าง Key ซึ่งเป็นส่วนสำคัญในการเชื่อมต่อและระบบป้องกันความปลอดภัย เมื่อกดเข้าไปแล้ว ให้สร้างชื่อ Key ของตนเอง และเลือกประเภทเป็น Device key เนื่องจาก Key ที่เราใช้ จะต้องใช้ร่วมกับบอร์ดตัวควบคุมที่ไม่ได้มีการเปลี่ยนแปลงข้อมูลบ่อยๆ และต้องการให้ระบบทำความรู้จักกันเพียงครั้งเดียว เพื่อการเชื่อมต่อครั้งต่อๆ ไปสามารถเชื่อมต่อกันได้อย่างรวดเร็ว เพราะอุปกรณ์และระบบรู้จักกันแล้ว

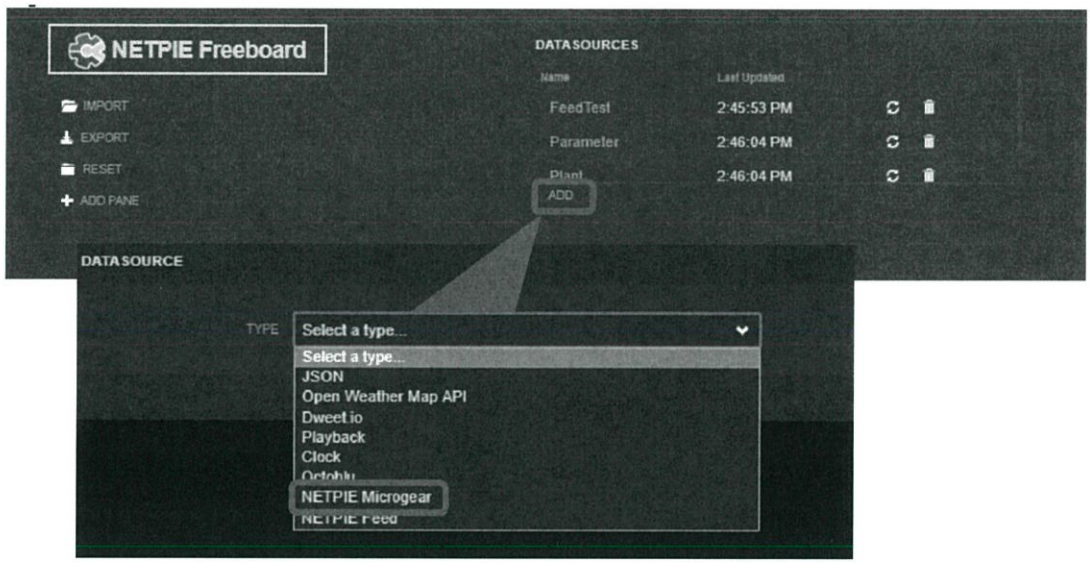


รูปที่ ก3.6 ข้อมูลภายใน Key

- เมื่อกดเข้าไปที่ Key ที่สร้างเสร็จแล้ว จะปรากฏข้อมูลของ Key ที่ระบบสร้างไว้ให้ ซึ่งข้อมูลทั้ง 3 นี้เป็นข้อมูลที่จะต้องนำไปใส่ในโค้ดของบอร์ดตัวควบคุมด้วย ซึ่งจะอธิบายในส่วนการใช้งานโค้ด Microgear

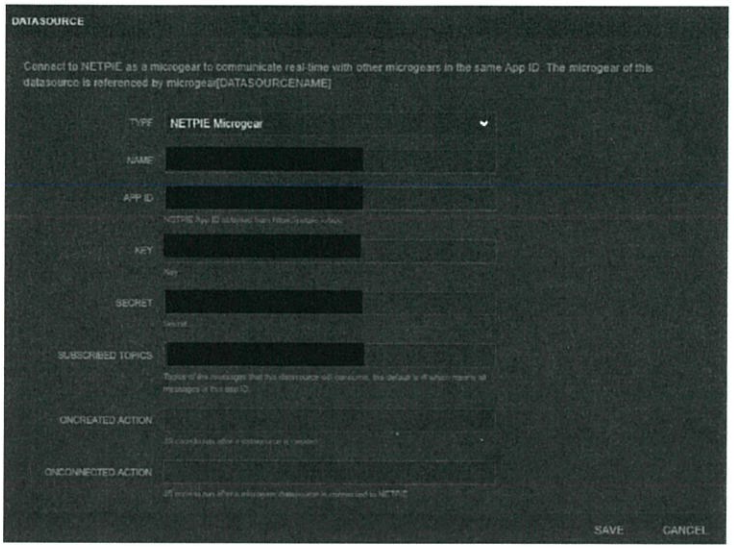
รูปที่ ก3.7 การใช้งานในส่วน Freeboard และการสร้าง Freeboard

- ต่อไปในโหมดของ Freeboards เป็นการสร้างหน้าจอสำหรับมอนิเตอร์ด้วย Widget ต่างๆ โดยขั้นแรกต้องสร้าง Freeboards ของเราก่อน โดยคลิกที่เครื่องหมาย + แล้วสร้างชื่อ เมื่อสร้างเสร็จแล้วจะปรากฏชื่อ Freeboards ทำการคลิกเข้าไป



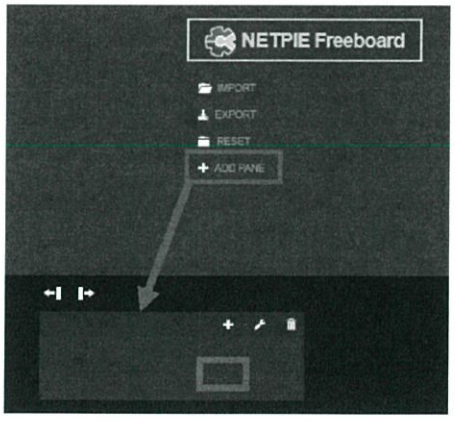
รูปที่ ก3.8 การส้ราง datasource ในการรับข้อมูล

- ภายในหน้า Freeboards จะมีหน้าตาดังรูปที่ ก3.8 โดยทุกครั้งที่เราจะรับข้อมูลจากบอร์ดตัวควบคุม ที่ส่งค่ามา จะต้องสร้าง Datasource เพื่อรับค่าก่อนเสมอ โดยกด Add เพื่อสร้าง โดยประเภทของ Datasource ที่ใช้คือ NETPIE Microgear จะใช้สำหรับ Widget ทั่วไปเช่น Gauge, Toggle ส่วน NETPIE Feed จะใช้สำหรับการรับข้อมูลมาพลอตกราฟ ซึ่งจะอธิบายในส่วนของโหมด Feeds อีกทีหนึ่ง



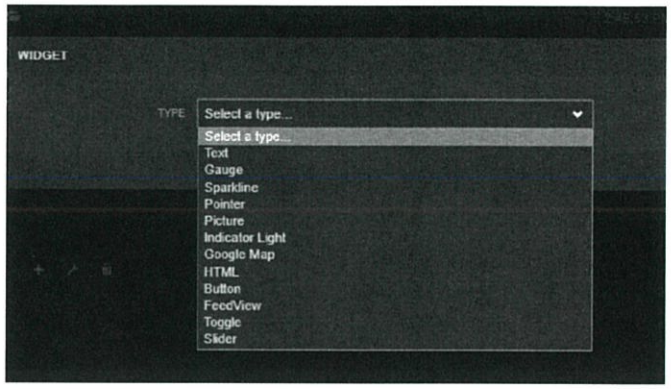
รูปที่ ก3.9 การตั้งค่าภายใน Datasource

- จากนั้นจะปรากฏหน้าต่างให้ใส่ค่าต่างๆ ให้เราตั้งชื่อ Datasource ที่ช่อง Name และใส่ข้อมูลในช่องต่างๆ ซึ่งเป็นค่าที่ได้จากหน้า Applications เช่น App ID (ชื่อของ Application ที่เราตั้งไว้) Key และ Secret จากนั้นในช่องของ Subscribe topic การใส่ค่าจะต้องมี Topic เดียวกันกับในส่วนของโค้ดที่บอร์ดตัวควบคุมส่งมาจึงจะรับค่าได้ ถ้าไม่กำหนด Topic มาต้องใส่ค่าเป็น /# จะเป็นค่า Default ของระบบ คือ เป็นการรอรับค่าจากข้อมูลที่ส่งมาแบบไม่มี Topic



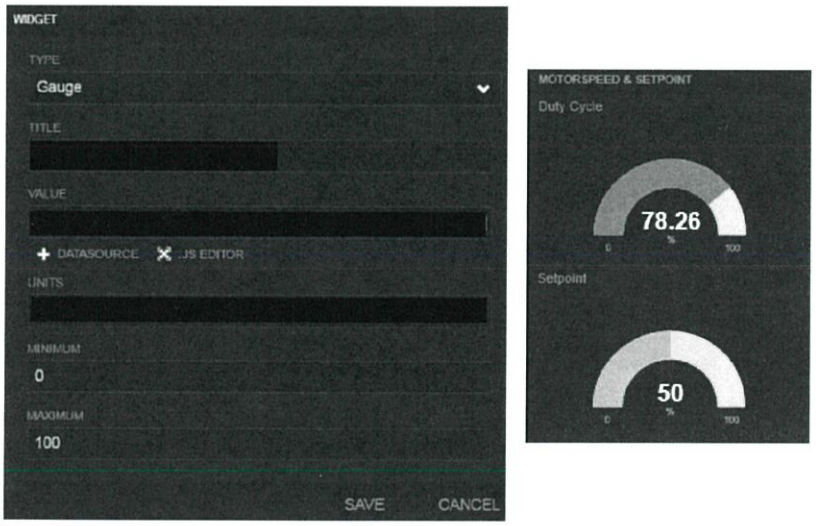
รูปที่ ก3.10 การสร้าง Pane สำหรับ Widget

- เมื่อสร้าง Datasource แล้ว ขั้นต่อไปคือการสร้าง Widget ที่ใช้แสดงค่า โดยกดที่ Add Pane จะมีช่องว่างปรากฏขึ้น กดเครื่องหมาย + เพื่อเพิ่ม Widget ลงบนช่องว่างที่สร้างไว้



รูปที่ ก3.11 การเลือกใช้ Widget

- โดย Widget ที่มีนั้น มีให้เลือกใช้หลากหลาย ผู้ใช้สามารถเลือกได้ตามความต้องการ ซึ่ง Widget แต่ละชนิดนั้น จะมีการตั้งค่าบางอย่างแตกต่างกัน ซึ่งสามารถศึกษาได้จากคู่มือการใช้งาน โดยสามารถอ่านได้ที่เว็บไซต์ https://netpie.io/tutorials

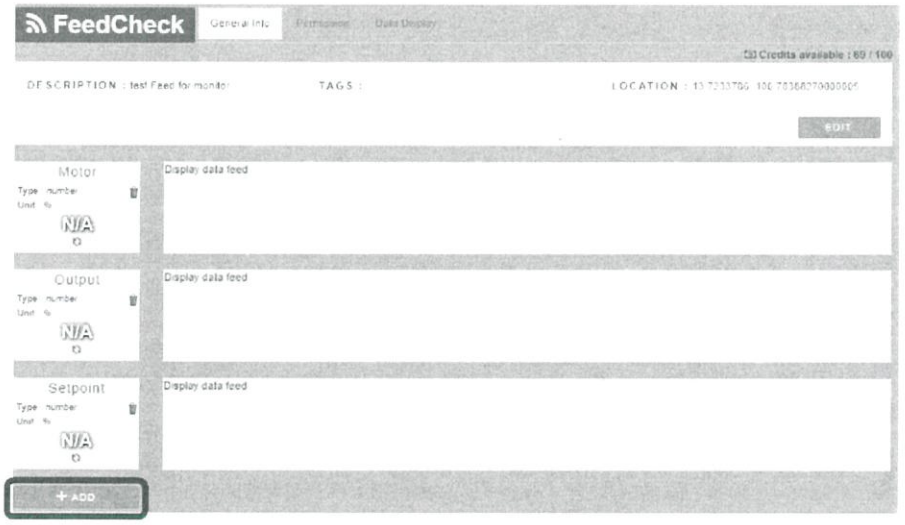


รูปที่ ก3.12 การตั้งค่าภายใน Widget ประเภท Gauge

- ตัวอย่างการใช้งาน Widget gauge โดยช่อง Title เป็นการตั้งชื่อ Gauge Value เป็นการรับค่า ต้องตั้งตาม form ดังนี้คือ datasources ["ชื่อ datasource"]["/ชื่อ AppID/ชื่อ Topic"].split(",")[0] โดย.split เป็นการแบ่งข้อมูลด้วยอักขระ “,” ในกรณีการส่งข้อมูลที่มี Topic เดียวกันแต่มีหลายชุดข้อมูล โดย Widget นี้สามารถตั้งค่าให้แบ่งข้อมูลเพื่อนำมาแสดงผลได้จากการตั้งค่าในช่อง [ ] โดย 0 คือชุดข้อมูลแรก Units เป็นการตั้งค่า หน่วยให้กับ Widget และ Minimum & Maximim เป็นการตั้งค่าต่ำสุด และสูงสุดให้กับ Widget ในการแสดงผล

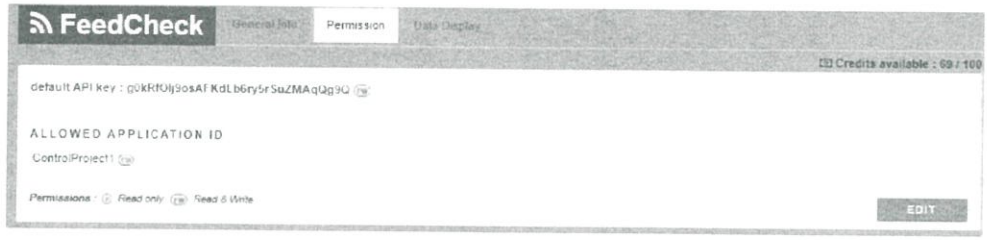
รูปที่ ก3.13 การใช้งานในส่วน Feed

- ต่อไปในโหมดของ Feeds เป็นการสร้างหน้าสำหรับการแสดงผลแบบกราฟ โดยขั้นแรกต้องสร้าง Feed ของเราก่อน โดยคลิกที่เครื่องหมาย + แล้วสร้างชื่อ เมื่อสร้างเสร็จแล้วจะปรากฏชื่อ Feed ทำการคลิกเข้าไปเพื่อสร้างรูปแบบกราฟ



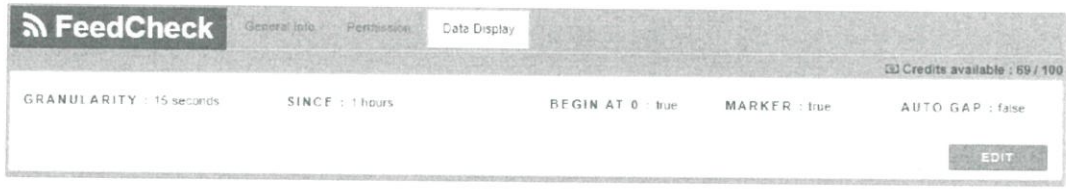
รูปที่ ก3.14 หน้าใช้งาน Feed และการสร้างตัวแปร

- โดยหน้าตาของหน้า Feed จะแสดงเป็นหน้าว่างๆ สำหรับรอรับค่าเพื่อมาพลอตกราฟ โดยขั้นตอนแรกต้องเพิ่มตัวแปรข้อมูลที่จะใช้พลอต โดยกดที่ Add แล้วตั้งชื่อตัวแปรและหน่วย โดยชื่อและจำนวนตัวแปร จะต้องตรงกับที่บอร์ดตัวควบคุมส่งมา



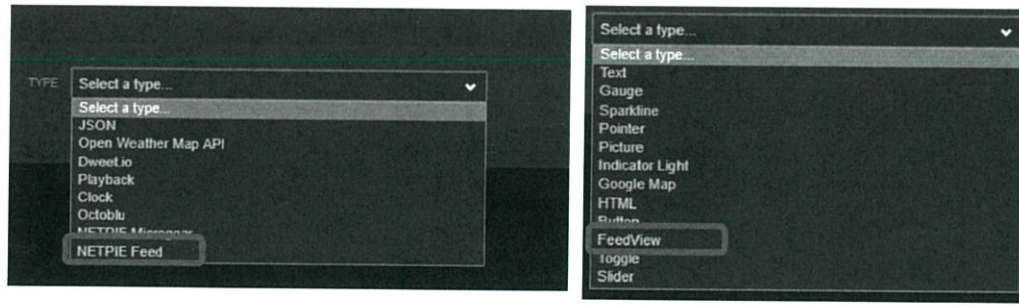
รูปที่ ก3.15 การกำหนดค่า Permission

- การตั้งค่าในโหมด Feed จะมี อยู่ 3 ส่วนด้วยกันคือ General info จะแสดงข้อมูลต่างๆที่เราตั้งค่าไว้ ส่วน Permission เป็นการกำหนดการเข้าถึงข้อมูล โดยใช้ API Key โดยเราสามารถตั้งค่าสิทธิ์ในการให้ผู้ใช้งาน Application อื่นๆสามารถเข้ามาดูในส่วนของ Feed ได้



รูปที่ ก3.16 การกำหนดค่าในการแสดงผล

- ในส่วนของ Data Display เป็นการกำหนดค่าในการพลอตข้อมูลเป็นกราฟ ว่าให้พลอตกราฟลงทุกๆกี่วินาที สามารถเซตรูปกราฟให้แสดงผลในช่วงเวลาได้ เช่น 1 ชั่วโมง หรือ 10 นาที ซึ่งผู้ใช้สามารถตั้งค่าเองได้ ตามความต้องการรูปแบบของกราฟข้อมูลที่ใช้ แต่กราฟที่ได้จะแยกข้อมูลกัน



รูปที่ ก3.17 การนำข้อมูล Feed ไปแสดงในหน้า Freeboard

- หากต้องการรวมข้อมูลลงเป็นกราฟเดียวกัน ต้องใช้ Widget ในหน้า Freeboards โดยสร้าง Datasource เป็น NETPIE Feed และทำใช้ Widget ที่ชื่อ Feedview โดยการตั้งค่าแบบละเอียดสามารถศึกษาได้จากคู่มือ

ภาคผนวก ข

โปสเตอร์

Department of Instrumentation and Control Engineering
(Control Engineering)
BEST 6007 Wireless Level Control System and Monitor

KMITL ENGINEERING PROJECT DAY 2018

Wireless Level Control System and Monitor

Nut Techosakondee , Treeyut Wannapirun and Thanatporn Boonpachuen

Abstract

This project presents a wireless level control system and monitor. The operation of this system provides the control of the desired level via LabVIEW program with wireless communication. Programming for control the system and design of user interface are implemented. The purpose of this project is to simulate the large industrial systems by using IOT technology and compatible devices. The performance and stability of this system are studied.

Introduction

In the present, IOT (internet of things) equipment and technology are played the greater role in our lives including work in the industry. IOT technology has been introduced and developed to transmit data for remote monitoring. It has a remote control and data acquisition system called SCADA (Supervisory Control and Data Acquisition). However, this system is large and expensive. The industry has turned its attention to the IOT because it is cheaper and more efficient workplace. It can replace the large tools and be developed to adapt for a variety of applications. They can also send data to the cloud system for remote monitoring through other devices except with computers in the control room.

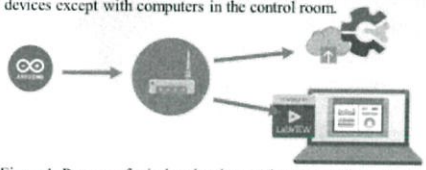


Figure 1. Process of wireless level control system and monitor

Methodology

The control broad of the system and programming are studied. Data from sensors is scaled to the appropriate range. Then scaled data is sent over WiFi to the computer. Function block in LabVIEW program is programmed for receiving information from the client. The process control is improved using PID controller. Finally, data is transfer to Cloud system for remote monitor.

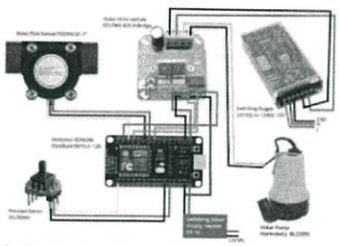


Figure 2. Circuit of wireless level control system and monitor

Results

From the experiment of wireless level control system, the system can connect between a NodeMCU client and a LabVEIW server program on a computer. It can control the water level as needed by adjusting the setpoint from the user interface. The system stability is controlled using PID controller and analyzed via the graphic display in the realtime. It can also be connected to another cloud server for sending the data to the cloud system for monitoring from a remote device such as a mobile phone or tablet can work as well.

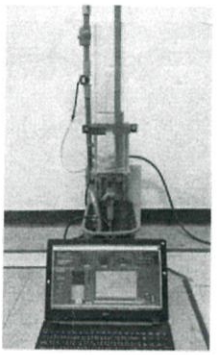


Figure 3. Experimental setup

Conclusion

The wireless level control system and monitor in this project integrates many branches of knowledge including the modern technology. IOT technology is one of knowledge used in this project. From the experimental results, it can be seen that the wireless level control system can operate effectively.

Figure 4. User Interface

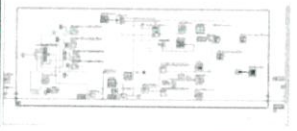


Figure 5. Code of LabVIEW

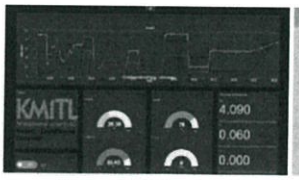


Figure 6. Monitor (NETPIE)

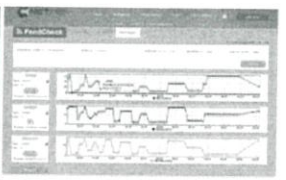


Figure 7. Data for display

References

[1] "NETPIE Training Course Materials" [Online] Available : https://netpie.io/tutorials

[2] "NodeMCU ESP8266 and Application" [Online] Available : http://thaieasyelec.com/article-wiki/embedded-electronics-application/getting-started-with-esp8266-nodemcu.html

[3] "Basic TCP/IP" [Online] Available : http://www.ni.com/white-paper/2710/en/

[4] "Multitasking and connecting for Nodemcu ESP8266" [Online] Available : https://www.facebook.com/groups/arduino.thai/

ENGINEERS LADKRABANG

E-mail: vanchai.ri@kmitl.ac.th, wandee.pe@kmitl.ac.th