

ระบบควบคุมอาร์เอฟแมกนีตรอนสเปตเตอริงผ่านหน้าจอสัมผัส

RF MAGNETRON SPUTTERING CONTROL SYSTEM VIA
TOUCH SCREEN DISPLAY



โครงการพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาวิทยาศาสตรบัณฑิต (ฟิสิกส์ประยุกต์)

ภาควิชาฟิสิกส์ คณะวิทยาศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2561

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

RF MAGNETRON SPUTTERING CONTROL SYSTEM VIA TOUCH SCREEN DISPLAY



A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR
THE DEGREE OF BACHELOR OF SCIENCE (APPLIED PHYSICS)
DEPARTMENT OF PHYSICS, FACULTY OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2018

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อโครงการพิเศษ ระบบควบคุมอาร์เอฟแมกนีตรอนสเปตเตอริงผ่านหน้าจอสัมผัส
RF MAGNETRON SPUTTERING CONTROL SYSTEM VIA
TOUCH SCREEN DISPLAY

ชื่อนักศึกษา นายภูมินทร์ รอดพัน รหัสนักศึกษา 58051123
นายรัฐพล หมั่นการ รหัสนักศึกษา 58051129
นางสาวอารีนันท์ แซ่จึ้ง รหัสนักศึกษา 58051179

ปริญญา วิทยาศาสตร์บัณฑิต (ฟิสิกส์ประยุกต์)

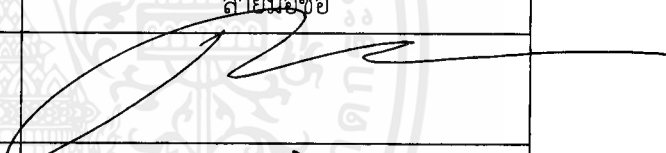
ภาควิชา ฟิสิกส์

ปีการศึกษา 2561

อาจารย์ที่ปรึกษา รศ.ดร.วราวุฒิ เถาลัดดา

อาจารย์ที่ปรึกษาร่วม ผศ.ดร.ศ.ทิพวรรณ คล้ายบุญมี

คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง (สจล.) อนุมัติให้
โครงการพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรบัณฑิต (ฟิสิกส์ประยุกต์)
ปีการศึกษา 2561

คณะกรรมการสอบ	ลายมือชื่อ
ดร.ภาณุพล โขลนกระโทก	
ประธานกรรมการ	
ดร.พิศาล สุขวิสูตร	พ.ศ.ค. สุขวิสูตร
กรรมการ	
รศ.ดร.วราวุฒิ เถาลัดดา	
กรรมการและอาจารย์ที่ปรึกษา	
ผศ.ดร.ศ.ทิพวรรณ คล้ายบุญมี	ผ.ทิพวรรณ คล้ายบุญมี
กรรมการและอาจารย์ที่ปรึกษาร่วม	

ลิขสิทธิ์ของคณะวิทยาศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อโครงการพิเศษ	ระบบควบคุมอาร์เอฟแมกนีตรอนสปีดเตอร์ริงผ่านหน้าจอสัมผัส
ชื่อนักศึกษา	นายภูมินทร์ รอดพัน รหัสนักศึกษา 58051123 นายรัฐพล หมั่นการ รหัสนักศึกษา 58051129 นางสาวอารินันท์ แซ่จิ่ง รหัสนักศึกษา 58051179
ปริญญา	วิทยาศาสตร์บัณฑิต (ฟิสิกส์ประยุกต์)
ภาควิชา	ฟิสิกส์
คณะ	วิทยาศาสตร์
มหาวิทยาลัย	สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา	2561
อาจารย์ที่ปรึกษา	รศ.ดร.วราวุฒิ เถาถัดดา
อาจารย์ที่ปรึกษาร่วม	ผศ.ดร.ศ.ทิพวรรณ คล้ายบุญมี

บทคัดย่อ

โครงการพิเศษนี้นำเสนอการพัฒนาระบบควบคุมการปลูกฟิล์มบางด้วยเทคนิคอาร์เอฟแมกนีตรอนสปีดเตอร์ริงโดยคอมพิวเตอร์ผ่านทางหน้าจอสัมผัส จุดประสงค์เพื่อให้เกิดความสะดวกสำหรับผู้ใช้งาน ทำให้ผู้ใช้งานสามารถควบคุมและรับรู้การทำงานของทั้งระบบจากที่เดียวคือหน้าจอคอมพิวเตอร์แบบสัมผัส การพัฒนาแบ่งออกเป็น 2 ส่วนหลัก ได้แก่ ส่วนควบคุมการทำงานของอุปกรณ์ในระบบปลูกฟิล์มบาง และส่วนติดต่อกับผู้ใช้ สำหรับส่วนระบบปลูกฟิล์มบางนั้น อุปกรณ์หลักที่ต้องควบคุมประกอบด้วย ระบบปั๊มสุญญากาศ แหล่งกำเนิดคลื่นวิทยุและแม่เหล็กเหนี่ยวนำและระบบหล่อเย็นของปั๊มสุญญากาศและแหล่งกำเนิดคลื่นวิทยุ สิ่งที่ต้องควบคุมระบบปั๊มสุญญากาศ ได้แก่ การเปิด-ปิดปั๊ม การตรวจวัดความดันสุญญากาศและการควบคุมอัตราการไหลของแก๊สเข้าสู่ห้องสุญญากาศ ส่วนของแหล่งกำเนิดคลื่นวิทยุและแม่เหล็กเหนี่ยวนำ จะต้องสามารถควบคุมกำลังงานของคลื่นวิทยุที่ต้องการส่งออกไปและตรวจวัดคลื่นวิทยุที่สะท้อนกลับได้ ในส่วนระบบหล่อเย็นสามารถตรวจวัดอัตราการไหลและอุณหภูมิของน้ำได้ ทั้ง 3 ส่วนที่กล่าวนี้จะถูกควบคุมและตรวจวัดโดยไมโครคอนโทรลเลอร์ Arduino UNO R3 ระบบทำงานได้โดยโปรแกรมที่พัฒนาขึ้นด้วย Labview ผู้ใช้สามารถตั้งค่าควบคุมและอ่านผลการทำงานผ่านทางหน้าจอคอมพิวเตอร์แบบสัมผัส

คำสำคัญ : โปรแกรม Labview ไมโครคอนโทรลเลอร์ ระบบอาร์เอฟแมกนีตรอนสปีดเตอร์ริง
Arduino UNO R3

Title	RF Magnetron Sputtering Control System Via Touch Display	
Students	Mr. Phumin Rodpun	Student ID 58051123
	Mr. Rathapol Mankan	Student ID 58051129
	Miss. Areenun Sae-Chueng	Student ID 58051179
Degree	Bachelor of Science (Applied Physics)	
Department	Physics	
Faculty	Science	
University	King Mongkut's Institute of Technology Ladkrabang (KMITL)	
Academic Year	2018	
Advisor	Assoc. Prof. Dr. Warawoot Thowlada	
Co-advisor	Asst. Prof. Dr. S.Tippawan Klayboonme	

Abstract

This special project proposes the development of RF magnetron sputtering control system via a touch screen display. The develop control system allows users control the entire system from one place. The system is divided into two main parts: RF magnetron sputtering system equipment control and the user interface for certain system. The main equipment of the sputtering system consists of a vacuum pump system, rf power generator and matching network and water-cooling system of vacuum pump and rf generator. For the vacuum pump system: pump power on-off, vacuum pressure measurement and gas flow rate to the vacuum chamber are control by computer. For the rf generator, rf power can be control and the reflection of rf signal can be monitored. In the cooling system section, the flow rate and water temperature can be monitored. The developed system uses Arduino UNO R3 as a microcontroller. The system is operating by a program developed with Labview. The user can set parameters, control and monitor the performance of operating condition through the touch-screen display.

Keywords: Labview Program, Microcontroller, RF Magnetron Sputtering System, Arduino UNO R3

กิตติกรรมประกาศ

โครงการพิเศษนี้สำเร็จลุล่วงไปได้ด้วยดีนั้น เนื่องจากการสนับสนุน และ ความอนุเคราะห์เป็นอย่างดีจาก รศ.ดร.วราวุฒิ เถาสัตตา อาจารย์ภาควิชาฟิสิกส์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อาจารย์ที่ปรึกษา ซึ่งคอยให้คำแนะนำอย่างใกล้ชิดพร้อมทั้งเสนอแนวทางและวิธีแก้ไขปัญหาดังต่าง ๆ

ขอขอบคุณ ผศ.ดร.ศ.ทิพวรรณ คล้ายบุญมี อาจารย์ภาควิชาฟิสิกส์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อ.ที่ปรึกษาร่วม ที่มอบความรู้เกี่ยวกับระบบควบคุมอาร์เอฟแมกนีตรอนสปัตเตอริงผ่านหน้าจอสัมผัส การทำงาน และการแก้ปัญหา

ขอขอบพระคุณ บิดา-มารดา ที่ให้ได้รับการศึกษา ตลอดจนคอยเลี้ยงดูและอบรมสั่งสอนเป็นกำลังใจสำคัญ เป็นแรงผลักดัน รวมถึงเพื่อนๆ และบุคคลอื่น ที่ไม่ได้กล่าวถึง ผู้จัดทำโครงการพิเศษ ขอขอบพระคุณเป็นอย่างสูงมา ณ โอกาสนี้

ภูมินทร์ รอดพัน
รัฐพล หมั่นการ
อารีนันท์ แซ่จึ้ง

สารบัญ

	หน้า
บทคัดย่อภาษาไทย.....	ก
บทคัดย่อภาษาอังกฤษ.....	ข
กิตติกรรมประกาศ.....	ค
สารบัญ.....	ง
สารบัญตาราง.....	ช
สารบัญรูป.....	ซ
คำย่อ/สัญลักษณ์.....	ฅ
บทที่ 1 บทนำ.....	1
1.1ความเป็นมาและความสำคัญ.....	1
1.2วัตถุประสงค์ของงานวิจัย.....	1
1.3ขอบเขตของงานวิจัย.....	1
1.4ขั้นตอนดำเนินงาน.....	2
1.5ประโยชน์ที่คาดว่าจะได้รับ.....	2
บทที่ 2 ทฤษฎีที่เกี่ยวข้อง.....	3
2.1ไมโครคอนโทรเลอร์.....	3
2.1.1หน่วยประมวลผลกลางหรือซีพียู (CPU : Central Processing Unit)	3
2.1.2หน่วยความจำ (Memory)	3
2.1.3ส่วนติดต่อกับอุปกรณ์ภายนอก หรือพอร์ต (Port)	3
2.1.4ช่องทางเดินของสัญญาณ หรือบัส (BUS).....	4
2.1.5วงจรกำเนิดสัญญาณนาฬิกา	4
2.2Arduino.....	5
2.2.1จุดเด่นที่ทำให้บอร์ด Arduino เป็นที่นิยม.....	6
2.2.2รูปแบบการเขียนโปรแกรมบน Arduino.....	6
2.2.3องค์ประกอบของArduino.....	8
2.3LabVIEW.....	9
2.3.1หลักการทำงานของLabView.....	9

สารบัญ (ต่อ)

	หน้า
2.3.2ส่วนประกอบต่าง ๆ ใน LabVIEW.....	10
2.3.3ข้อดีในการใช้โปรแกรม LabVIEW.....	18
บทที่ 3ขั้นตอนการดำเนินงาน.....	19
3.1อุปกรณ์ที่เราจะทำการควบคุมรับค่าและแสดงผล.....	19
3.1.1Massflow Controller Argon , Nitrogen , Oxygenand and Hydrogen.....	19
3.1.2Rotary Pump and Heater (Diffusion Pump)	21
3.1.3RF Generator.....	21
3.1.4Rf Generator (CPS)	23
3.1.5Water Cooler Flow.....	24
3.1.6Chiller Temperature.....	25
3.2Arduino With LabView.....	25
3.2.1Rf Generator and Relay.....	27
3.2.2Rf Generator (CPS).....	29
3.2.3Flow Rate and temperature.....	32
3.2.4Mass Flow.....	33
3.3ระบบควบคุมระบบ Rf Magnetron Sputtering.....	36
3.4ขั้นตอนการทดลอง.....	37
3.4.1ขั้นตอนที่ 1.....	37
3.4.2ขั้นตอนที่ 2.....	38
3.4.3ขั้นตอนที่ 3.....	38
3.4.4ขั้นตอนที่ 4.....	39
บทที่ 4 ผลการทดลองและการอภิปรายผล.....	40

สารบัญ (ต่อ)

	หน้า
4.1ผลการทดลองที่ 1.....	40
4.1.1วิเคราะห์ผลการทดลอง.....	41
4.2ผลการทดลองที่ 2.....	42
4.2.1วิเคราะห์ผลการทดลอง.....	43
4.3ผลการทดลองที่ 3.....	45
4.4ผลการทดลองที่ 4.....	4.4
บทที่ 5 สรุปผลการวิจัยและข้อเสนอแนะ.....	48
5.1สรุปผลการวิจัย.....	49
เอกสารอ้างอิง.....	50
ภาคผนวก.....	51
ภาคผนวก ก.....	55
ภาคผนวก ข.....	59
ภาคผนวก ค.....	

สารบัญตาราง

ตารางที่	หน้า
2.1เปรียบเทียบค่าศัพท์ที่ใช้ในการเขียนโปรแกรม.....	18
3.1Pin Name Type Description.....	21
3.2Pin Name Type Description.....	23
4.1ผลการทดลองการควบคุมอัตราการไหลของก๊าซอาร์กอน (Ar) (โดยใช้ Mass Flow ของ Ar กับเครื่องควบคุมอัตราการไหลที่สร้างขึ้นและใช้ Mass Flow ของ H ₂ กับเครื่องควบคุมอัตราการไหล รุ่น ROD – 4).....	40
4.2ผลการทดลองของ Rf Generator Power (W) และ Rf Generator Control Box(W)	42



สารบัญรูป

รูปที่	หน้า
2.1องค์ประกอบของ Microcontroller.....	4
2.2Arduino UNO Board.....	6
2.3การเชื่อมต่อสื่อสารระหว่างคอมพิวเตอร์กับ Microcontroller.....	6
2.4การเลือกชนิดของ Board ใน Arduino IDE.....	7
2.5เลือกหมายเลข Comport ของบอร์ด.....	7
2.6การ Compiled และ การ Upload.....	7
2.7Layout & Pin out Arduino Board (Model: Arduino UNO R3).....	8
2.8แสดงตัวอย่างเครื่องมือวัดเสมือนที่สร้างจาก LabVIEW.....	9
2.9Block Diagram เครื่องมือวัดที่สร้างจาก LabVIEW.....	9
2.10Front Panel ของ LabVIEW.....	11
2.11Object ที่อยู่บน Front Panel ของ LabVIEW.....	11
2.12Controls Palette ที่ใช้ในการออกแบบ Front Panel.....	12
2.13Tools Palette ที่ใช้ในการออกแบบ Front Panel.....	12
2.14ตัวอย่าง Block Diagram.....	13
2.15ตัวอย่าง Block Diagram Node.....	14
2.16เครื่องมือ Tools Palette.....	14
2.17แสดงลักษณะทั่วไปของ Icon และ Connector.....	15
2.18แสดงข้อมูลประเภท Numeric.....	16
2.19แสดงข้อมูลประเภท Boolean.....	17
3.1MCP4902.....	20
3.2Mass Flow Controller Block Diagram	20
3.3Solid State Relay.....	21
3.4Rf Generator (CPS).....	24
3.5YF-S201 Water Flow Sensor.....	24
3.6Data Sheet ของ YF-S201.....	25
3.7วงจรการเชื่อมต่อ YF-S201.....	25

สารบัญรูป (ต่อ)

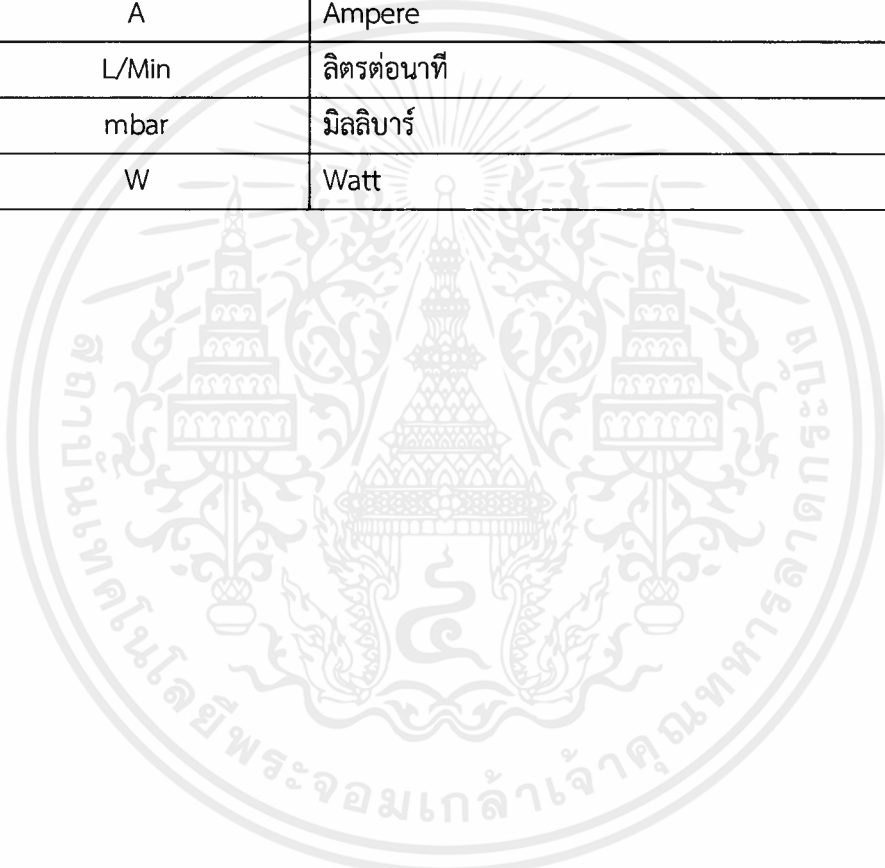
รูปที่	หน้า
3.8DS18B20 Temperature Sensor.....	26
3.9Data Sheet ของ DS18B20.....	26
3.10วงจรการเชื่อมต่อ DS18B20.....	27
3.11แสดงแผนผังระหว่าง Arduino และ LabView.....	27
3.12โค้ดที่ใช้ในการทำงานของ Rf Generator and Relay.....	28
3.13โปรแกรม LabView ในส่วนของ Rf Generator.....	29
3.14ไดอะแกรมการทำงานของ Rf Generator.....	30
3.15ไดอะแกรมการทำงานของ Rf Generator (CPS).....	31
3.16โค้ดที่ใช้ในการทำงานของ Flow Rate and temperature.....	32
3.17โปรแกรม LabView ในส่วนของ DS18D20.....	33
3.18โปรแกรม LabView ในส่วนของ Flow Rate.....	33
3.19โค้ดที่ใช้ในการทำงานของ Mass Flow.....	34
3.20โปรแกรม LabView ในส่วนของ Mass Flow.....	34
3.21ไดอะแกรมการทำงานของ Mass Flow.....	35
3.22โครงสร้างภายในของเครื่องควบคุมระบบ Rf Magnetron Sputtering.....	36
3.23ด้านหลังของเครื่องควบคุมระบบ Rf Magnetron Sputtering.....	36
3.24ด้านหน้าของเครื่องควบคุมระบบ Rf Magnetron Sputtering.....	37
3.25วิธีการเลือก Com Port.....	37
3.26อุณหภูมิน้ำใน Chiller.....	38
3.27อัตราการไหลออกของน้ำใน Chiller.....	38
3.28ป้อนข้อมูลของ Mass Flow.....	39
3.29การเพิ่ม Rf Magnetron เข้าในระบบ.....	39
4.1กราฟความสัมพันธ์ระหว่างอัตราการไหล (sccm) กับ ความดัน (mbar).....	41

สารบัญรูป (ต่อ)

รูปที่	หน้า
4.2กราฟแสดงความสัมพันธ์ระหว่าง Rf Generator Power (W) กับ Rf Generator Control Box (W).....	43
4.3ภายในระบบสุญญากาศ.....	44
4.4กราฟ Rf Generator และ Reflect Rf Generator.....	44
4.5หน้า Front Panel ของ ระบบ Rf Generator และ Reflect Rf Generator.....	45
4.6เปรียบเทียบระหว่างหน้าจอแสดงผล กับ เทอร์โมคอปเปอร์.....	45
4.7ไฟแจ้งเตือนของระบบ การวัดอุณหภูมิ.....	46
4.8อัตราไหลของน้ำที่หน้า Front Panel.....	46
4.9ฟิล์มที่ปลูกจากระบบ Rf Magnetron Sputtering Control System Via Touch Screen Display.....	47

คำย่อ/สัญลักษณ์

คำย่อ/สัญลักษณ์	คำอธิบาย
sccm	Standard Cubic Centimeters (per) Minute
V	Voltage
GND	Ground
D/A	Digital to Analog
A	Ampere
L/Min	ลิตรต่อนาที
mbar	มิลลิบาร์
W	Watt



บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญ

เนื่องจากระบบแมกนีตรอนสปัตเตอริง (Magnetron Sputtering) เป็นระบบที่มีส่วนประกอบในการทำงานหลายส่วนและชิ้นงานที่ทำผ่านระบบนี้จะออกมาถูกต้อง มีคุณภาพดีหรือไม่ ก็ขึ้นอยู่กับการทำงานของระบบของผู้ใช้งานซึ่งต้องควบคุมให้มีค่าพารามิเตอร์ของอุปกรณ์ต่าง ๆ ตรงตามที่คำนวณไว้ ซึ่งเป็นเรื่องยุ่งยากที่ผู้ใช้งานจะต้องคอยตรวจสอบอุปกรณ์ทุกส่วนว่ายังทำงานตรงตามที่ตั้งค่าหรือไม่ หรือมีอุปกรณ์ขึ้นโดยหยุดทำงานไป

โครงการนี้เป็นการพัฒนาการควบคุมระบบแมกนีตรอนสปัตเตอริง (Magnetron Sputtering) โดยจะให้แสดงค่าพารามิเตอร์ทั้งหมดของระบบที่ผู้ใช้งานจำเป็นต้องตรวจเช็คระหว่างการใช้งาน อยู่บนหน้าจอสัมผัส ผู้ใช้งานสามารถมองเห็น กำหนด และสามารถเปลี่ยนแปลงค่า parameter ทุกส่วนของระบบได้โดยผ่านหน้าจอสัมผัสนี้

1.2 วัตถุประสงค์ของงานวิจัย

- 1) เพื่อศึกษาการใช้งานอาร์ดูโน้ (Arduino) และเซ็นเซอร์ (Sensor)
- 2) เพื่อพัฒนาโปรแกรมสำหรับผู้ใช้งานระบบแมกนีตรอนสปัตเตอริง (RF Magnetron sputtering โดยใช้ LabView)
- 3) เพื่อสร้างชุดควบคุมระบบแมกนีตรอนสปัตเตอริง (RF Magnetron Sputtering)

1.3 ขอบเขตของงานวิจัย

- 1) ศึกษาการเขียนโปรแกรมโดยใช้อาร์ดูโน้ (Arduino)
- 2) ศึกษาการเขียนโปรแกรมโดยใช้แลปวิว (LabVIEW)
- 3) ศึกษาวิธีการควบคุมและหลักการทำงานของอุปกรณ์ที่ใช้ในระบบแมกนีตรอนสปัตเตอริง (RF Magnetron Sputtering)

1.4 ขั้นตอนดำเนินงาน

กิจกรรม	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค.
1.ศึกษาการใช้งานระบบ RF Magnetron Sputtering							
2.ศึกษาการทำงานและควบคุม เครื่องมือผ่าน RS232 และ LabVIEW							
3.สร้างระบบและเขียนโปรแกรมควบคุมระบบ RF Magnetron Sputtering							
4.ทดสอบและปรับปรุง โปรแกรมควบคุมระบบ RF Magnetron Sputtering							
5.จัดทำรูปเล่มโครงงานพิเศษ							

1.5 ประโยชน์ที่คาดว่าจะได้รับ

- 1) สามารถเขียนโปรแกรมเพื่อควบคุมระบบโดยใช้แลปวิว (LabView) ได้
- 2) สามารถควบคุมระบบแมกนีตรอนสปัตเตอริง (RF Magnetron sputtering) ได้อย่างมีประสิทธิภาพ

บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

2.1 ไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ (Microcontroller) คือ อุปกรณ์ควบคุมขนาดเล็ก ซึ่งบรรจุความสามารถที่คล้ายคลึงกับระบบคอมพิวเตอร์โดยในไมโครคอนโทรลเลอร์ได้รวมเอาซีพียู, หน่วยความจำ และพอร์ต ซึ่งเป็นส่วนประกอบหลักสำคัญของระบบคอมพิวเตอร์เข้าไว้ด้วยกัน โดยทำการบรรจุเข้าไว้ในตัวถังเดียวกัน

โครงสร้างโดยทั่วไป ของไมโครคอนโทรลเลอร์นั้น สามารถแบ่งออกมาได้เป็น 5 ส่วนใหญ่ๆ ดังต่อไปนี้

2.1.1 หน่วยประมวลผลกลางหรือซีพียู (CPU : Central Processing Unit)

CPU ทำหน้าที่เป็นศูนย์กลางควบคุมการทำงานของระบบคอมพิวเตอร์ทั้งหมด โดยนำข้อมูลจากอุปกรณ์รับข้อมูลมาทำงาน ประมวลผลข้อมูลตามคำสั่งของโปรแกรม และส่งผลลัพธ์ออกไปหน่วยแสดงผล

2.1.2 หน่วยความจำ (Memory)

Memory สามารถแบ่งออกเป็น ส่วน คือ หน่วยความจำที่มีเพื่อไว้ใช้สำหรับเก็บโปรแกรมหลัก (Program Memory) เปรียบเสมือนฮาร์ดดิสก์ของ เครื่องคอมพิวเตอร์ตั้งโต๊ะ คือ ข้อมูลใด ๆ ที่ถูกเก็บไว้ในนี้จะไม่สูญหายไปแม้ไม่มีไฟเลี้ยง อีกส่วนหนึ่งคือหน่วยความจำข้อมูล (Data Memory) ใช้เป็นเหมือนกระดานทดในการคำนวณของซีพียู และเป็นที่พักข้อมูลชั่วคราวขณะทำงาน แต่หากไม่มีไฟเลี้ยง ข้อมูลก็จะหายไปคล้ายกับหน่วยความแรม (RAM) ในเครื่องคอมพิวเตอร์ทั่ว ๆ ไป แต่สำหรับไมโครคอนโทรลเลอร์สมัยใหม่ หน่วยความจำข้อมูลจะมีทั้งที่เป็นหน่วยความจำแรม ซึ่งข้อมูลจะหายไปเมื่อไม่มีไฟเลี้ยง และเป็นอีอีพรอม (EEPROM : Erasable Electrically Read-Only Memory) ซึ่งสามารถเก็บข้อมูลได้แม้ไม่มีไฟเลี้ยง

2.1.3 ส่วนติดต่อกับอุปกรณ์ภายนอก หรือพอร์ต (Port)

Port มี 2 ลักษณะคือ พอร์ตอินพุต (Input Port) และพอร์ตเอาต์พุต (Output Port) ส่วนนี้จะใช้ในการเชื่อมต่อกับอุปกรณ์ภายนอก ถือว่าเป็นส่วนที่สำคัญมาก ใช้ร่วมกันระหว่างพอร์ตอินพุต เพื่อรับสัญญาณ อาจจะใช้การกดสวิตช์ เพื่อนำไปประมวลผลและส่งไปพอร์ตเอาต์พุต เพื่อแสดงผลเช่น การติดสว่างของหลอดไฟ เป็นต้น

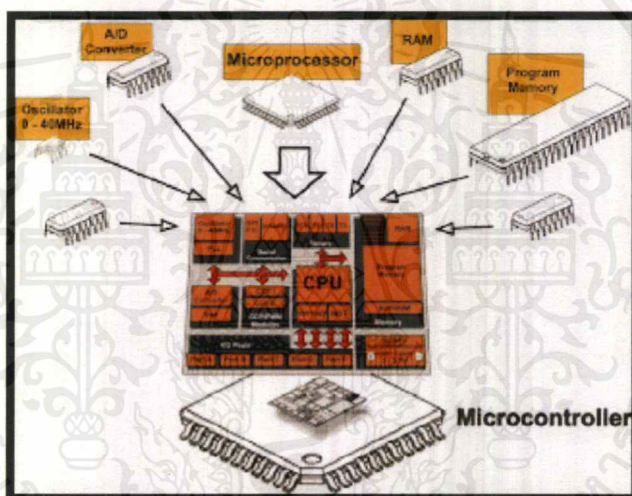
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1.4 ช่องทางเดินของสัญญาณ หรือบัส (BUS)

BUS คือเส้นทางการแลกเปลี่ยนสัญญาณข้อมูลระหว่าง ซีพียู หน่วยความจำ และพอร์ต เป็นลักษณะของสายสัญญาณ จำนวนมากอยู่ในตัวไมโครคอนโทรลเลอร์ โดยแบ่งเป็นบัสข้อมูล (Data Bus), บัสแอดเดรส (Address Bus) และบัสควบคุม (Control Bus)

2.1.5 วงจรกำเนิดสัญญาณนาฬิกา

วงจรกำเนิดสัญญาณนาฬิกานับเป็นส่วนประกอบที่สำคัญมากอีกส่วนหนึ่ง เนื่องจากการทำงานที่เกิดขึ้นในตัวไมโครคอนโทรลเลอร์จะขึ้นอยู่กับสัญญาณนาฬิกา หากสัญญาณนาฬิกาที่มีความถี่สูงการทำงานก็จะสามารถทำงานได้ดีขึ้นส่งผลให้ ไมโครคอนโทรลเลอร์ ตัวนั้นมีความเร็วในการประมวลผลสูงตามไปด้วย



รูปที่ 2.1 องค์ประกอบของ Microcontroller

นอกจากนี้ยังมีส่วนพิเศษอื่น ๆ จะขึ้นอยู่กับกระบวนการผลิตของแต่ละบริษัทที่จะผลิตขึ้นมาใส่คุณสมบัติพิเศษลงไป เช่น

- ADC (Analog to Digital) ส่วนภาครับสัญญาณอนาล็อกแปลงไปเป็นสัญญาณดิจิทัล
- DAC (Digital to Analog) ส่วนภาคส่งสัญญาณดิจิทัลแปลงไปเป็นสัญญาณอนาล็อก
- I2C (Inter Integrate Circuit Bus) เป็นการสื่อสารอนุกรม แบบซิงโครนัส (Synchronous) เพื่อใช้ ติดต่อสื่อสาร ระหว่าง ไมโครคอนโทรลเลอร์ (MCU) กับอุปกรณ์ภายนอก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ซึ่งถูกพัฒนาขึ้นโดยบริษัท Philips Semiconductors โดยใช้สายสัญญาณเพียง 2 เส้นเท่านั้น คือ serial data (SDA) และสาย serial clock (SCL) ซึ่งสามารถ เชื่อมต่ออุปกรณ์ จำนวนหลายๆ ตัว เข้าด้วยกันได้ ทำให้ MCU ใช้พอร์ตเพียง 2 พอร์ตเท่านั้น

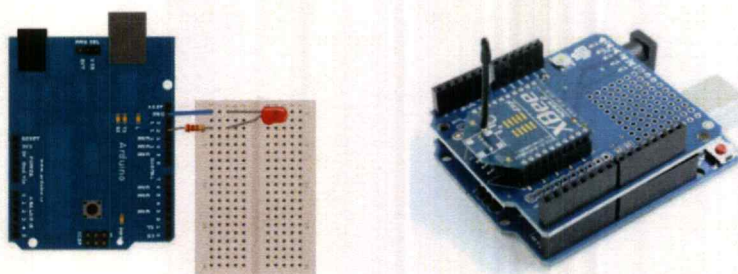
-SPI (Serial Peripheral Interface) เป็นการเชื่อมต่อกับอุปกรณ์เพื่อรับส่ง ข้อมูลแบบซิงโครไนส์ (Synchronize) มีสัญญาณนาฬิกาเข้ามาเกี่ยวข้องกับระหว่าง ไมโครคอนโทรลเลอร์ (Microcontroller) หรือจะเป็นอุปกรณ์ภายนอกที่มีการรับส่งข้อมูลแบบ SPI อุปกรณ์ที่ทำหน้าที่เป็นมาสเตอร์ (Master) โดยปกติแล้วจะเป็นไมโครคอนโทรลเลอร์ หรือ อาจกล่าวได้ว่าอุปกรณ์ Master จะต้องควบคุมอุปกรณ์ Slave ได้ โดยปกติตัว Slave มักจะเป็น ไอซี (IC) หน้าที่พิเศษต่าง ๆ เช่น ไอซีอุณหภูมิ, ไอซีฐานเวลานาฬิกาจริง (Real-Time Clock) หรือ อาจเป็นไมโครคอนโทรลเลอร์ที่ทำหน้าที่ในโหมด Slave ก็ได้เช่นกัน

- PWM (Pulse Width Modulation) การสร้างสัญญาณพัลส์แบบสแควร์เวฟ ที่ สามารถปรับเปลี่ยนความถี่และ Duty Cycle ได้เพื่อนำไปควบคุมอุปกรณ์ต่าง ๆ เช่น มอเตอร์

- UART (Universal Asynchronous Receiver Transmitter) ทำหน้าที่รับส่ง ข้อมูลแบบอะซิงโครไนส์สำหรับมาตรฐานการรับส่งข้อมูลแบบ RS-232

2.2 Arduino

Arduino อ่านว่า (อา-ดู-อิ-โน้ หรือ อาดูยโน้) เป็นบอร์ดไมโครคอนโทรลเลอร์ตระกูล AVR ที่มีการพัฒนาแบบ Open Source คือมีการเปิดเผยข้อมูลทั้งด้าน Hardware และ Software ตัว บอร์ด Arduino ถูกออกแบบมาให้ใช้งานได้ง่าย ดังนั้นจึงเหมาะสำหรับผู้เริ่มต้นศึกษา ทั้งนี้ ผู้ใช้งานยังสามารถดัดแปลง เพิ่มเติม พัฒนาต่อยอดทั้งตัวบอร์ด หรือโปรแกรมต่อได้อีกด้วยความ ง่ายของบอร์ด Arduino ในการต่ออุปกรณ์เสริมต่าง ๆ คือผู้ใช้งานสามารถต่อวงจรอิเล็กทรอนิกส์ จากภายนอกแล้วเชื่อมต่อเข้ามาที่ขา I/O ของบอร์ด หรือเพื่อความสะดวกสามารถเลือกต่อกับ บอร์ดเสริม (Arduino Shield) ประเภทต่าง (ดูตัวอย่างรูปที่ 2.2)

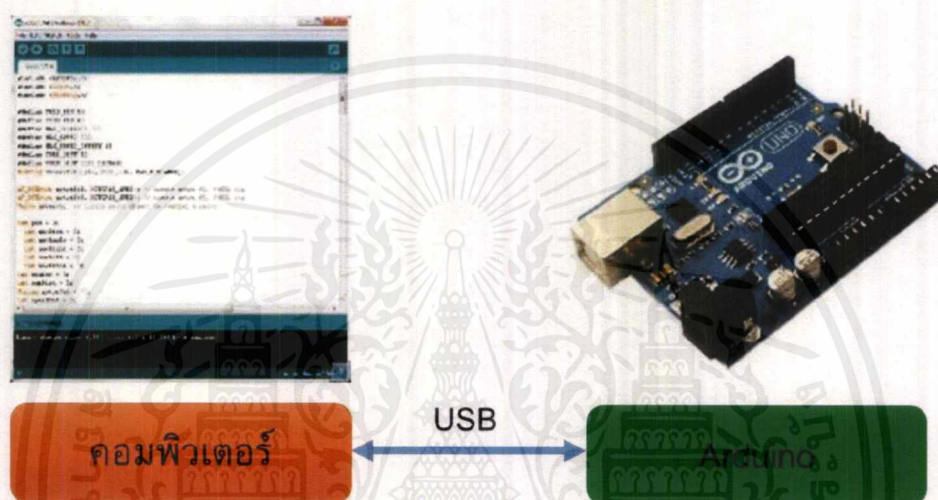


รูปที่ 2.2 Arduino UNO Board

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.1 จุดเด่นที่ทำให้บอร์ด Arduino เป็นที่นิยม

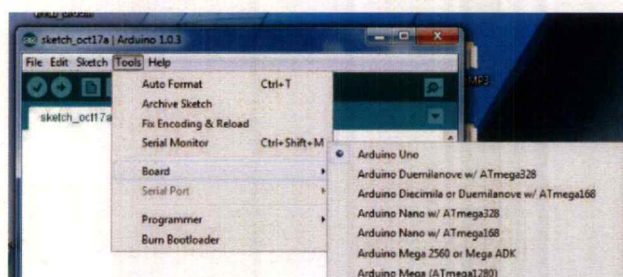
- 1) ง่ายต่อการพัฒนา มีรูปแบบคำสั่งพื้นฐาน ไม่ซับซ้อนเหมาะสำหรับผู้เริ่มต้น
- 2) มี Arduino Community กลุ่มคนที่ร่วมกันพัฒนาที่แข็งแรง
- 3) Open Hardware ทำให้ผู้ใช้สามารถนำบอร์ดไปต่อยอดใช้งานได้หลายด้าน
- 4) ราคาไม่แพง
- 5) Cross Platform สามารถพัฒนาโปรแกรมบน OS ใดก็ได้



รูปที่ 2.3 การเชื่อมต่อสื่อสารระหว่างคอมพิวเตอร์กับ Microcontroller

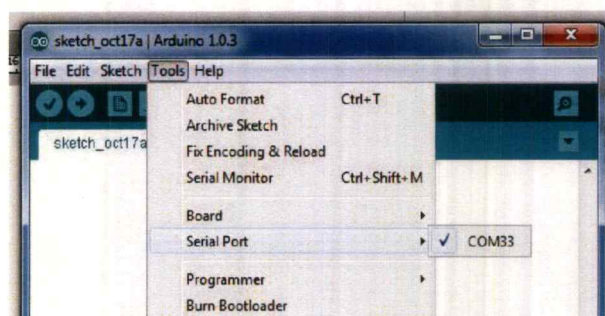
2.2.2 รูปแบบการเขียนโปรแกรมบน Arduino

- 1) เขียนโปรแกรมบนคอมพิวเตอร์ ผ่านทางโปรแกรม Arduino IDE ซึ่งสามารถดาวน์โหลดได้
- 2) หลังจากเขียนโค้ดโปรแกรมเรียบร้อยแล้ว ให้ผู้ใช้งานเลือกรุ่นบอร์ด Arduino ที่ใช้และหมายเลข Com port



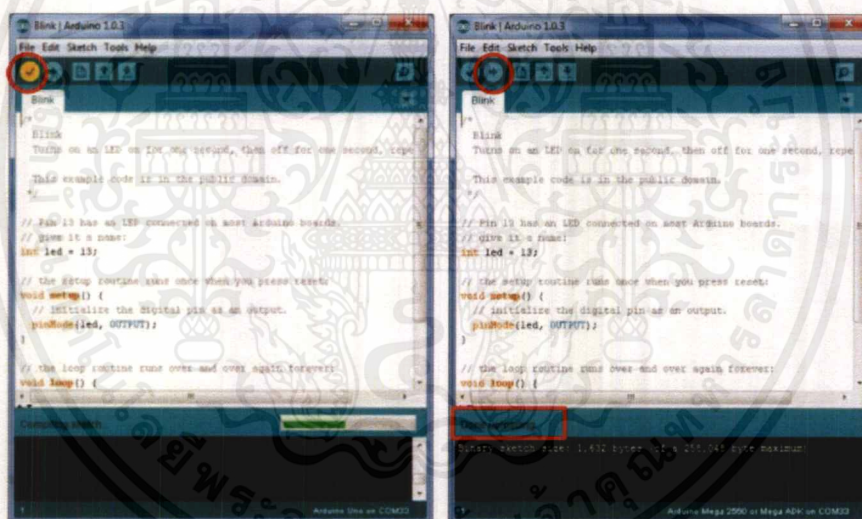
รูปที่ 2.4 การเลือกชนิดของ Board ใน Arduino IDE

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



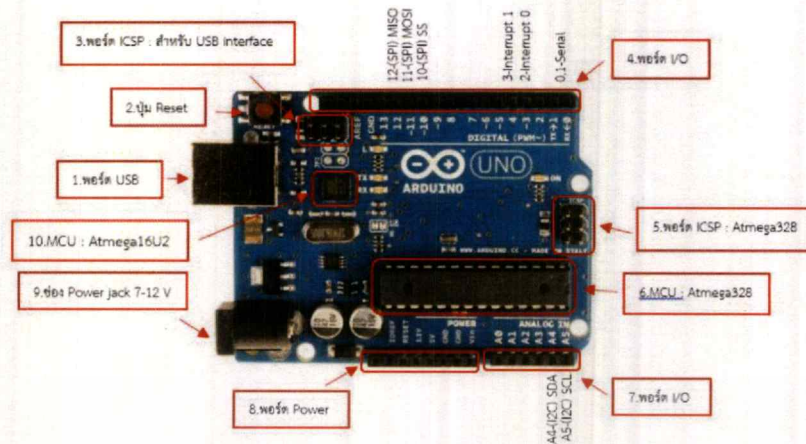
รูปที่ 2.5 เลือกหมายเลข Comport ของบอร์ด

3) กดปุ่ม Verify เพื่อตรวจสอบความถูกต้องและ Compile โค้ดโปรแกรม จากนั้นกดปุ่ม Upload โค้ด โปรแกรมไปยังบอร์ด Arduino ผ่านทางสาย USB เมื่ออัปโหลดเรียบร้อยแล้ว จะแสดงข้อความแถบข้างล่าง “Done uploading” และบอร์ดจะเริ่มทำงานตามที่เขียนโปรแกรมไว้ได้ทันที



รูปที่ 2.6 การ Compiled และ การ Upload

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.7 Layout & Pin out Arduino Board (Model: Arduino UNO R3)

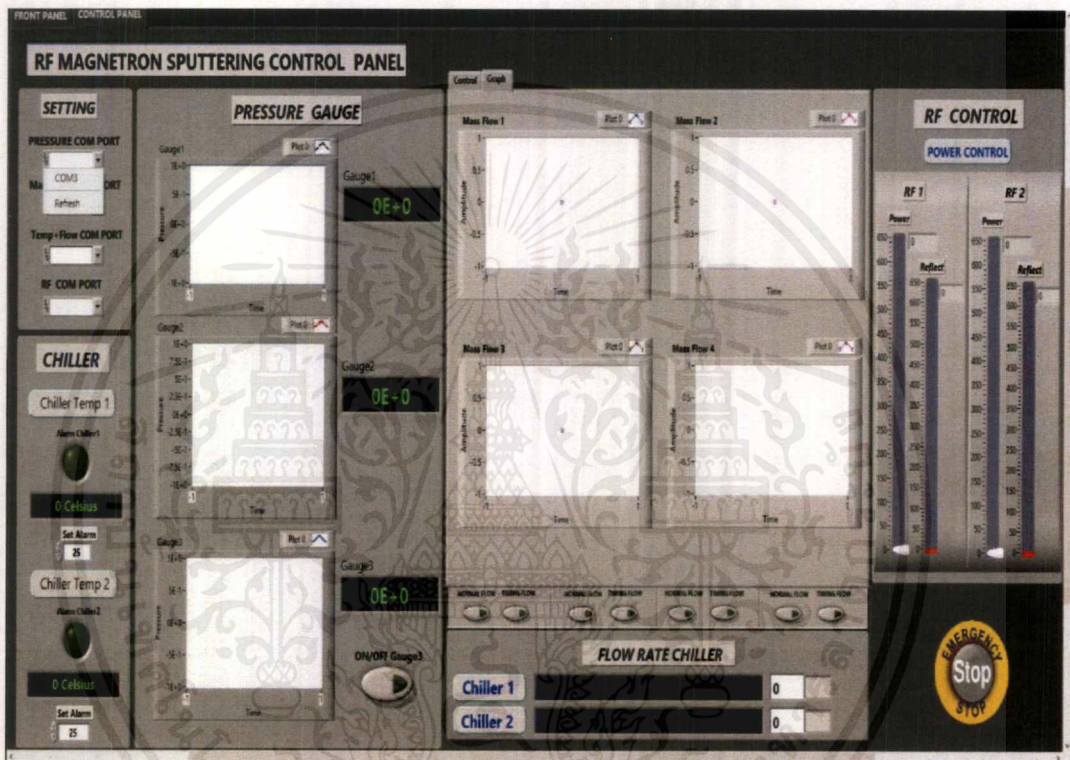
2.2.3องค์ประกอบของArduino

- 1) USB Port ใช้สำหรับต่อกับ Computer เพื่ออัปโหลดโปรแกรมเข้า MCU และจ่ายไฟให้กับบอร์ด
- 2) Reset Button : เป็นปุ่ม Reset ใช้กดเมื่อต้องการให้ MCU เริ่มการทำงานใหม่
- 3) ICSP Port ของ Atmega16U2 เป็นพอร์ตที่ใช้โปรแกรม Visual Com port บน Atmega16U2
- 4) I/O Port : Digital I/O ตั้งแต่ขา D0 ถึง D13 นอกจากนี้ บาง Pin จะทำหน้าที่อื่น ๆ เพิ่มเติมด้วย เช่น Pin0,1 เป็นขา Tx , Rx Serial, Pin3,5,6,9,10 และ 11 เป็นขา PWM
- 5) ICSP Port : Atmega328 เป็นพอร์ตที่ใช้โปรแกรม Bootloader
- 6) MCU : Atmega328 เป็น MCU ที่ใช้บนบอร์ด Arduino
- 7) I/O Port : นอกจากจะเป็น Digital I/O แล้ว ยังเปลี่ยนเป็น ช่องรับสัญญาณอนาล็อก ตั้งแต่ขา A0-A5
- 8) Power Port : ไฟเลี้ยงของบอร์ดเมื่อต้องการจ่ายไฟให้กับวงจรภายนอก ประกอบด้วยขาไฟเลี้ยง +3.3 V, +5V, GND, V_{in}
- 9) Power Jack : รับไฟจาก Adapter โดยที่แรงดันอยู่ระหว่าง 7-12 V
- 10) MCU ของ Atmega16U2 เป็น MCU ที่ทำหน้าที่เป็น USB to Serial โดย Atmega328 จะติดต่อกับ Computer ผ่าน Atmega16U

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3 LabVIEW

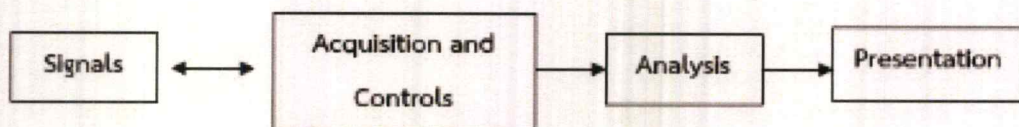
LabVIEW ย่อมาจากคำว่า Laboratory Virtual Instrument Engineering Workbench โปรแกรมที่พัฒนาขึ้นโดยใช้ LABVIEW จะเรียกว่า Virtual Instrument หรือจะเรียกย่อ ๆ ว่า VI ซึ่งหมายถึงเครื่องมือวัดเสมือน ดังตัวอย่าง จากรูปนี้เป็น Oscilloscope ที่ได้ทำการสร้างขึ้นบนหน้าจอคอมพิวเตอร์



รูปที่ 2.8 แสดงตัวอย่างเครื่องมือวัดเสมือนที่สร้างจาก LabVIEW

2.3.1 หลักการทำงานของLabVIEW

LabVIEW อาศัยหลักการทำงานของเครื่องมือวัดหรือการวัดคุมทำให้ผู้ใช้สามารถออกแบบตามที่ใช้ต้องการ หลักการดังกล่าวแบ่งออกเป็น 3 ส่วนใหญ่ๆ คือ



รูปที่ 2.9 Block Diagram เครื่องมือวัดที่สร้างจาก LabVIEW

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1) Acquisition: ซึ่งเป็นส่วนที่รับข้อมูล (Input) จากสิ่งแวดล้อมภายนอกเข้าสู่ระบบในที่นี้คือ คอมพิวเตอร์ โดยข้อมูลที่เข้าสู่ระบบนี้อาจมาจากการวัด DAQ (สำหรับสัญญาณทางไฟฟ้า)

2) Analysis: หลังจากที่ได้รับข้อมูลแล้วอาจจะผ่านฟังก์ชันในการวิเคราะห์ข้อมูล ซึ่งจะแสดงผลในรูปแบบที่สื่อความหมายในสิ่งที่ผู้ใช้งานสามารถนำไปแสดงแทนสื่อที่วัดได้และใช้งานได้

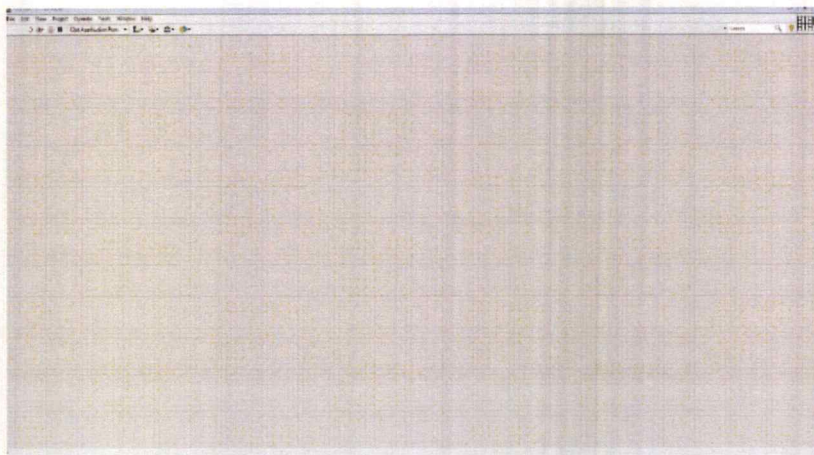
3) Presentation: คือ การแสดงผลในรูปแบบที่เป็นประโยชน์ต่อผู้ใช้งาน โดยอาจแสดงบนหน้าจอคอมพิวเตอร์ เช่น DMM (Digital Multimeter) แสดงผลเฉพาะที่วัดได้โดยไม่ต้องจำเป็นต้องรู้ความสัมพันธ์กับเวลา หรือ Spectrum Analysis จะแสดงสัญญาณในรูปความถี่หรือการพิมพ์ออกมาเป็นรายงานหรือเก็บข้อมูลในฮาร์ดดิสก์

2.3.2 ส่วนประกอบต่าง ๆ ใน LabVIEW

โปรแกรมที่เขียนขึ้นมาโดย LabVIEW เราจะเรียกว่า Virtual Instrument (VI) เพราะลักษณะที่ปรากฏทางจอภาพเมื่อผู้ใช้ใช้งานจะเหมือนกับเครื่องมือหรืออุปกรณ์ทางวิศวกรรม ในขณะที่เดียวกันหลังจากของอุปกรณ์เสมือนจริงเหล่านั้นจะเป็นการทำงานของ ฟังก์ชัน , Subroutines และโปรแกรมหลักเหมือนกับภาษาทั่วไป สำหรับ VI หนึ่งๆ จะประกอบด้วยส่วนประกอบ 3 ส่วนคือ

1) Front Panel

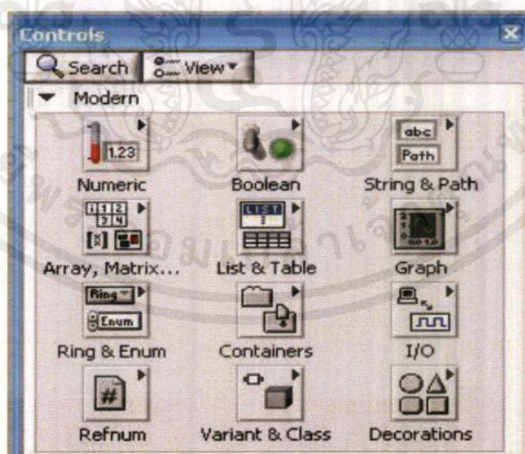
Front Panel หรือหน้าปัทม์ จะเป็นส่วนที่ใช้สื่อความกันระหว่างผู้ใช้กับโปรแกรม (หรือที่นิยมเรียก User Interface) โดยทั่วไปจะมีลักษณะเหมือนกับหน้าปัทม์ของเครื่องมือหรืออุปกรณ์ที่ใช้งานด้านการวัดทั่ว ๆ ไป โดยทั่วไปจะประกอบด้วย สวิตช์ปิดเปิด, ปุ่มบิด, ปุ่มกด จอแสดงผลหรือแม้แต่ค่าที่ผู้ใช้สามารถกำหนด สำหรับผู้ที่คุ้นเคยกับการเขียนโปรแกรมประเภท Visual ทั้งหลายคงจะเข้าใจกันดีว่า Front Panel นี้จะเปรียบเสมือนเป็น GUI ของโปรแกรมหรือ VI นั้นเอง



รูปที่ 2.10 Front Panel ของ LabVIEW

1.1) เครื่องมือที่ใช้ออกแบบ Front Panel

เครื่องมือที่ใช้ในการออกแบบ Front Panel จะประกอบไปด้วย Control Palette และ Tools Palette ซึ่ง LabVIEW มี Controls Palette ที่ใช้ในการออกแบบ Front Panel แสดงดังรูป 2.12 ซึ่งเป็นส่วนที่ติดต่อกับผู้ใช้งาน (User Interface) โดยจะจัดเป็นกลุ่มต่าง ๆ เช่น กลุ่มของตัวเลข (Numeric) ซึ่งภายในกลุ่มจะมี Control และ Indicator ต่าง ๆ ที่เกี่ยวกับตัวเลข



รูปที่ 2. 11 Controls Palette ที่ใช้ในการออกแบบ Front Panel

Tools Palette คือ เครื่องมือที่ใช้ในการพัฒนาโปรแกรม ซึ่งจะใช้ทั้งการออกแบบ Front Panel และ Block Diagram ในส่วนนี้จะกล่าวถึง Tools Palette สำหรับออกแบบ Front Panel

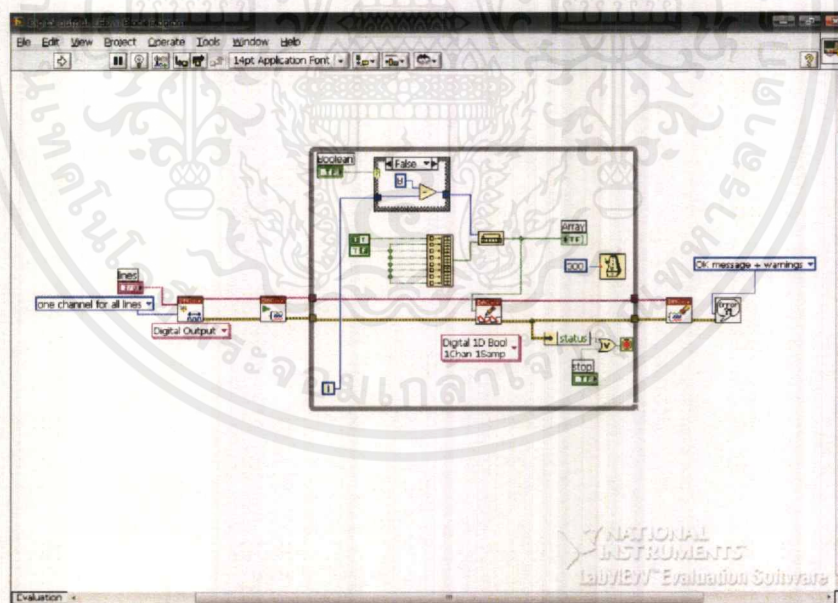
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.12 Tools Palette ที่ใช้ในการออกแบบ Front Panel

1.2) Block Diagram

Block Diagram นี้จะประกอบด้วย ฟังก์ชัน ค่าคงที่ โปรแกรมควบคุมการทำงานหรือโครงสร้าง จากนั้นในแต่ละส่วนเหล่านี้ ซึ่งจะปรากฏในรูปของ Block เราจะได้รับ การต่อสาย (Wire) สำหรับ Block ที่เหมาะสมเข้าด้วยกัน เพื่อกำหนดลักษณะ การไหลของข้อมูลระหว่าง Block เหล่านั้น ทำให้ข้อมูลได้รับการประมวลผลตามที่ต้องการและ แสดงผลออกมาให้แก่ผู้ใช้ต่อไป



รูปที่ 2.13 ตัวอย่าง Block Diagram

2) Block Diagram Node

Node คือรูป Icon ที่อยู่บน Block Diagram ซึ่งมี Input และ/หรือ Output และจะทำงานตามหน้าที่เมื่อมีการรันโปรแกรม โดยแบ่งเป็นสามชนิดหลัก ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1) Function คือ Node ที่มีหน้าที่ในการทำงานพื้นฐานของคอมพิวเตอร์ ซึ่งเราไม่สามารถที่จะเจาะเข้าไปดูรายละเอียดภายในได้อีก เช่น การบวก การคูณ

2) SubVIs หรือ Subroutine หรือ Subprogram คือ โปรแกรมย่อยที่ถูกเขียนขึ้นมาเพื่อถูกนำมาเรียกใช้ในอีกโปรแกรมหนึ่ง เราสามารถเปิดเข้าไปดู front panel และ block diagram ได้เมื่อ double click ที่ Icon ของมัน

3) Express VIs เป็น subVIs ประเภทพิเศษโดยที่เมื่อเราเลือก Express VI มาวางบน Block Diagram มันจะปรากฏหน้าต่าง Configuration ขึ้นมาเพื่อนให้เราเข้าไปป้อนค่า Parameters ต่าง ตามต้องการและเมื่อเราป้อนค่าเสร็จ มันก็จะสร้างโค้ดไว้ภายในอัตโนมัติ ตามที่เราได้ตั้งค่าไว้ ซึ่งความสามารถของ Express VI นี้ทำให้เราแทบไม่จำเป็นต้องต่อสาย Input เลยเพราะ Parameter ทั้งหมดได้ถูกสร้างขึ้นมาแล้วถูกเก็บไว้ภายในเรียบร้อยแล้ว จึงทำให้การเขียน LabVIEW ง่ายและเร็วขึ้นมาก สังเกตง่ายๆ Express VI จะมี Icon ขนานใหญ่ที่มีพื้นหลังเป็นสีฟ้า

Function

SubVI

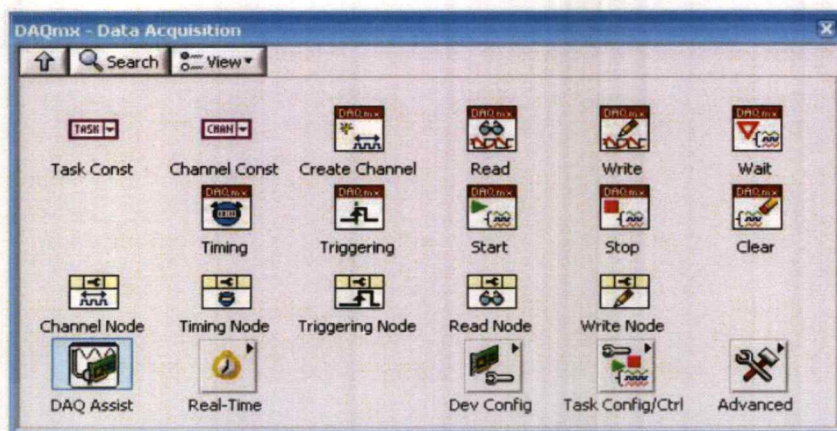
ExpressVI



รูปที่ 2.14 ตัวอย่าง Block Diagram Node

2.1 เครื่องมือที่ใช้ในการเขียนโปรแกรมบน Block Diagram

LabVIEW ใช้ Functions Palette ซึ่งจะมี Function และ SubVI ต่าง ๆ ที่มีอยู่แล้วให้ผู้ใช้เลือกใช้ โดย Function และ SubVI จัดเป็นกลุ่ม ๆ เช่น Numeric Function จะมี Function ต่าง ๆ เกี่ยวกับตัวเลข เช่น บวก ลบ คูณ หาร



รูปที่ 2.15 เครื่องมือสำหรับ Dam – Data Acquisition

2.2 Tools Palette สำหรับ Block diagram

เปิด Tools Palette โดยการ Click ที่ Window >> Show Tools Palette หรือกด Shift Right-Click แล้ว Click เลือก Tool ที่ต้องการใช้ (ถ้า Automatic Tools Selection เปิดอยู่แล้ว (LED) เป็นสีเขียว ให้ Click เพื่อปิด Automatic Tool Selection ซึ่งมีเฉพาะใน Version 6.1 ขึ้นไป) Tool ที่ใช้มีดังต่อไปนี้



รูปที่ 2.16 เครื่องมือ Tools Palette

- 1) Operation Tool ใช้ในการเปลี่ยนแปลงค่าหรือเลือกค่าคงที่ใน Block Diagram
- 2) Position/Size/Select ใช้ในการเลือก/เคลื่อนย้าย/จัดขนาดของสิ่งที่สร้างขึ้นบน Block Diagram
- 3) Edit Text Tool ใช้ในการแก้ไขข้อความที่เป็นตัวอักษร หรือเพิ่มข้อความลงบน Front Panel
- 4) Wiring Tool ใช้ในการโยงสาย (Wiring) ระหว่าง Terminal หรือ Node ซึ่งสายที่โยงนี้จะเป็นเส้นทางเดินของข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3) Icon และ Connector

Icon และ Connector เปรียบเสมือนโปรแกรมย่อย Subroutine ในโปรแกรมปกติทั่ว ๆ ไป โดย Icon จะหมายถึง Block Diagram ตัวหนึ่งที่มีการส่งข้อมูลเข้าและออกผ่านทาง Connector ซึ่งใน LabVIEW เราจะเรียก Subroutine นี้ว่า SubVI ข้อดีของการเขียนโปรแกรมด้วยภาษา G นี้ก็คือเราสามารถสร้าง VI ที่ละส่วนขึ้นมาให้ทำงานด้วยตัวเองได้อย่างอิสระ จากนั้นในภายหลังหากเราต้องการเราก็สามารถเขียน โปรแกรมอื่นขึ้นมาเพื่อเรียกใช้งาน VI ที่เราเคยสร้างขึ้นก่อนหน้านี้ทีละตัว ซึ่งทำให้ VI ที่เราเขียนขึ้นก่อนกลายเป็น SubVI ไป



รูปที่ 2.17 แสดงลักษณะทั่วไปของ Icon และ Connector

คำศัพท์ต่าง ๆ ที่ใช้กันใน LabVIEW นี้ออกจะแตกต่างจากที่เราใช้กันในภาษาการเขียนโปรแกรม ตัวหนังสือทั่ว ๆ ไปในหลาย ๆ ด้าน ดังนั้นเพื่อให้ผู้ที่เริ่มใช้ LabVIEW เข้าใจถึงศัพท์ต่าง ๆ ที่ใช้ในโปรแกรม เราจึงขอเปรียบเทียบศัพท์ที่ใช้ใน LabVIEW กับโปรแกรมพื้นฐานทั่ว ๆ ไปตามตารางที่ได้แสดงต่อไปนี้

ตารางที่ 2. 2 เปรียบเทียบคำศัพท์ที่ใช้ในการเขียนโปรแกรม

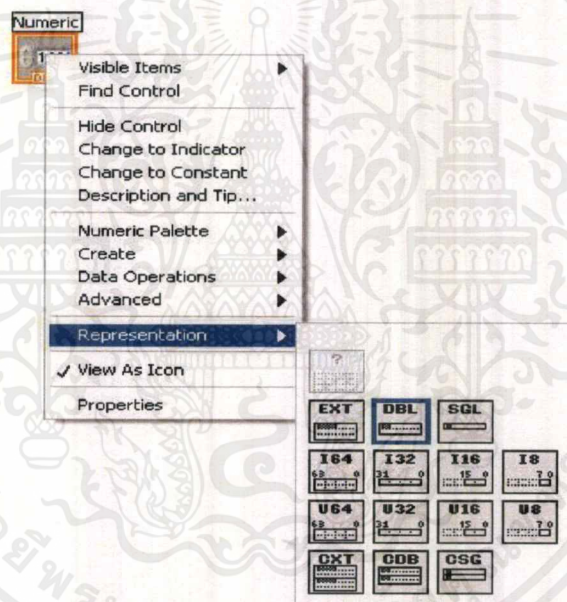
LabVIEW	โปรแกรมพื้นฐาน	หน้าที่
VI	Program	ตัวโปรแกรมหลัก
Function	function	ฟังก์ชันสำเร็จรูปที่สร้างขึ้นมากับโปรแกรมนั้นเช่น sin, log เป็นต้น
SubVI	Subroutine	โปรแกรมย่อยที่ถูกเรียกใช้โดยโปรแกรมหลัก
Front Panel	user interface	ส่วนที่ติดต่อกับผู้ใช้
Block Diagram	Program code	การเขียนตามขั้นตอนของทีแต่ละโปรแกรมกำหนดขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3) ประเภทของข้อมูล

ในการเขียนโปรแกรมทั่ว ๆ ไปจะต้องมีการประกาศตัวแปรก่อนที่จะใช้ตัวแปรนั้น แต่สำหรับโปรแกรม LabVIEW มันจะจัดการให้เองหมดโดยผู้ใช้ไม่จำเป็นต้องทำเอง เพียงแค่เลือกประเภทของข้อมูลที่มาวางบนโค้ดให้ถูกต้องเท่านั้น ประเภทของข้อมูลภายใน LabVIEW ก็มีหลายอย่างเหมือนกับโปรแกรมในภาษาอื่น ๆ และยังมีอีกบางประเภทที่ใช้ใน LabVIEW เท่านั้น โปรแกรม LabVIEW แบ่งข้อมูลเป็น 6 ชนิดดังนี้คือ

3.1) Numeric คือ ข้อมูลประเภทตัวเลขที่มีทั้งจำนวนเต็มซึ่งใน Block Diagram จะเห็นเป็นสีน้ำเงิน และจำนวนทศนิยมจะเห็นเป็นสีส้ม และสามารถเปลี่ยนไปมาได้ โดยการคลิกขวาที่ตัวเลขนั้น แล้วเลือก representation และเลือกประเภทตัวเลขได้เลย



รูปที่ 2.18 แสดงข้อมูลประเภท Numeric

3.2) Boolean คือ ข้อมูลประเภทที่มีสองค่า คือ True และ false บน Block Diagram จะแสดงข้อมูลเป็นสีเขียว และสำหรับ Front Panel ตัว Boolean จะมีลักษณะเป็น ตัวControl หรือ สวิตช์ ถ้าเป็น Output ก็จะเป็น LED หรือหลอดไฟประเภทต่าง ๆ แสดงดังรูป 2.40

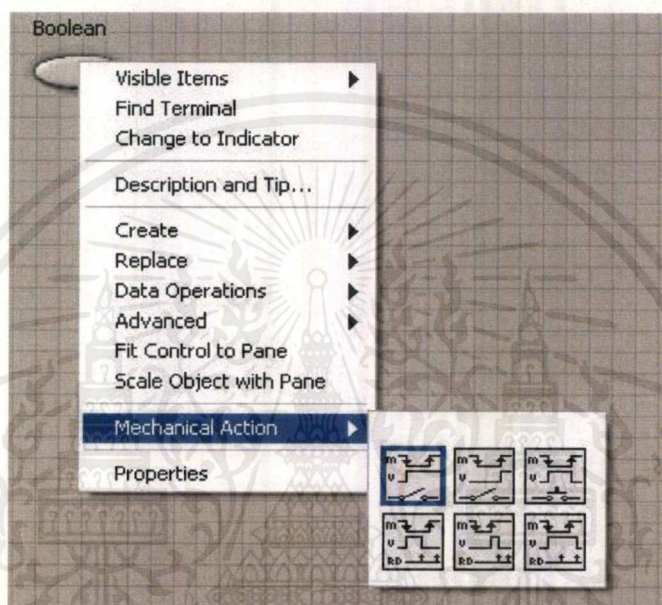
1) String คือ ข้อมูลประเภทที่เป็นตัวอักษร Icon จะแสดงเป็นสี่เหลี่ยม สำหรับการแสดงผล จะมีอยู่ 4 แบบ คือ

1.1) Normal Display คือการแสดงปกติ

1.2) Code Display คือการแสดงแบบ โค้ดมีประโยชน์สำหรับแสดงตัวอักษรที่ตาเปล่ามองไม่เห็น การเว้นวรรค แทบ หรือการขึ้นบรรทัดใหม่

1.3) Password Display คือการแทนตัวอักษรด้วย *

1.4) Hex Display แสดงผลเป็นรหัสเลขฐานสิบหก



รูปที่ 2. 19 แสดงข้อมูลประเภท Boolean

2) Enum คือข้อมูลประเภทแสดงให้ผู้ใช้เห็นเป็นตัวหนังสือ แต่ค่าจริงของมันคือตัวเลข ดังนั้นบน Block Diagram เราจึงมองเห็นข้อมูลประเภทนี้เป็นสีน้ำเงิน ซึ่งเหมือนกับจำนวนเต็ม

3) Dynamic เป็นข้อมูลที่อยู่ในรูปของ Waveform บน Block Diagram ถูกแสดงด้วยสีน้ำเงินเข้มซึ่งภายในจะประกอบด้วย Array ของเวฟฟอร์ม Time Stamp ชื่อของสัญญาณ ข้อมูลประเภท Dynamic นี้ส่วนใหญ่ใช้ใน Express VI จากพวกการอ่าน กำเนิดและวิเคราะห์สัญญาณ

4) Time Stamp เป็นข้อมูลที่ประกอบด้วยวันที่ และเวลาที่มีความละเอียดถึงมิลลิวินาที Time Stamp บน Block diagram จะมีหน้าต่างที่เป็นสีน้ำตาลเส้นหน้าสามารถนำมาแปลงให้เป็น วันที่ เวลา แบบ String ได้

2.3.3 ข้อดีในการใช้โปรแกรม LabVIEW

- ใช้ภาษา G (Graphical Language) หรือภาษารูปภาพแทนการเขียนโค้ด
- สามารถเปลี่ยนเครื่องคอมพิวเตอร์ส่วนบุคคลของเราให้กลายเป็นเครื่องมือวัดในหลายรูปแบบเช่นไม่ว่าจะเป็น Oscilloscope, Multi-meter, Function Generator, Strain meter Thermometer หรือเครื่องมือวัดอื่น ๆ ตามที่เราต้องการ



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3

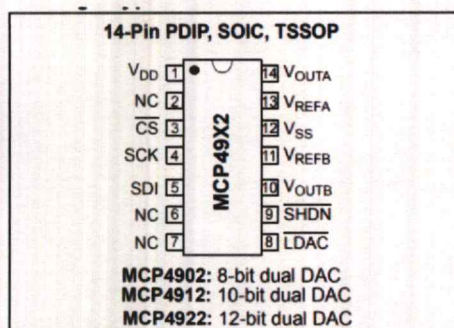
ขั้นตอนการดำเนินงาน

3.1 อุปกรณ์ที่เราจะทำการควบคุมรับค่าและแสดงผล

- 1) Massflow Controller Argon
- 2) Massflow Controller Nitrogen
- 3) Massflow Controller Oxygen
- 4) Massflow Controller Hydrogen
- 5) Rotary Pump
- 6) Heater (Diffusion Pump)
- 7) RF Generator
- 8) Water cooler flow
- 9) Chiller Temperature
- 10) Touch Screen Display
- 11) Relay

3.1.1 Massflow Controller Argon, Nitrogen, Oxygen and Hydrogen

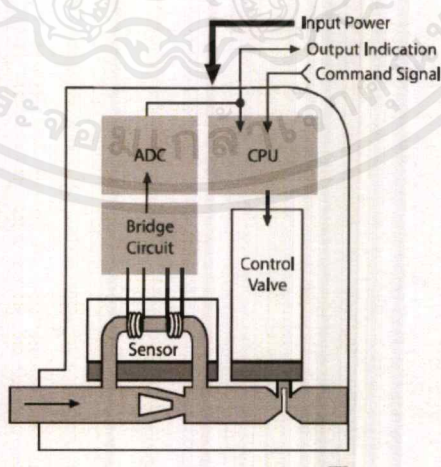
พอร์ตสำหรับการเชื่อมต่อของ Massflow module มีทั้งสิ้น 9 ขา แต่ในที่นี้เราเลือกใช้ 5 ขาคือขา 4 GND, 3 +15 V, 5 -15 V, 7 Signal Common และ 6 Setpoint ลักษณะของ Flow Signal และ Set Point คือเป็นสัญญาณอนาล็อก 0-5 V DC Hydrogen Module มีอัตราการไหล 0 -500 sccm Oxygen Module มีอัตราการไหล 0-10 sccm Methane Module มีอัตราการไหล 0-10 sccm Argon Module มีอัตราการไหล 0-20 sccm ส่วน Nitrogen Module มีอัตราการไหล 0-50 sccm วิธีที่เราจะเลือกใช้คือการใช้ IC Digital to Analog Converter รับค่าสัญญาณส่งไปที่ Arduino และจ่ายแรงดันให้กับขา SetPoint รับค่าสัญญาณ Voltage การไหลที่แท้จริงไปให้กับ Arduino



รูปที่ 3.1 MCP4902

MCP4922 เป็น IC Digital to Analog Converter ความละเอียดขนาด 12 Bit ทำงานที่ไฟเลี้ยง 5 โวลต์ โดยมีการเชื่อมต่อแบบ SPI interface มี OUTPUTA และ OUTPUTB ระดับโวลต์สูงสุดที่ 4095 bit สามารถเลือกระดับโวลต์ output ได้เป็น 2.048 โวลต์ หรือ 4.096 โวลต์ ตามการเลือกใน Bit Selector การส่งงาน 1 ครั้งจะมีการรับส่งข้อมูลทั้งหมด 16 Bit โดยแต่ละ bit คือ

- Bit ที่ 15 0 เลือก Output A 1 เลือก Output B
- Bit ที่ 14 เป็น Don't Care
- Bit ที่ 13 เลือกระดับแรงดัน Output สูงสุด 0 เลือกเป็น 2.048 V 1 เลือกเป็น 4.096 V
- Bit ที่ 12 เลือกการทำงาน 0 เป็น Non-available mode 1 เป็น Active Mode
- Bit ที่ 11-0 เป็นข้อมูลจำนวนจริง จำนวน 12 Bit 0 – 4095



รูปที่ 3.2 Mass Flow Controller Block Diagram

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1.2 Rotary Pump and Heater (Diffusion Pump)

อุปกรณ์ใช้ไฟขนาด 220 โวลต์ การออกแบบการควบคุม 2 สิ่งนี้โดยใช้ Relay ที่มีขนาดของคอยล์ 24 โวลต์เพื่อเปิดปิดไฟบ้าน ในที่นี้จะใช้ Relay ตัวเล็กขนาดคอยล์ 5V DC เพื่อจ่ายไฟ 5 โวลต์ให้กับ Solid State Relay



รูปที่ 3.3 Solid State Relay

3.1.3 RF Generator

รายชื่อ DB-25 Pin ด้านหลังของเครื่อง RF OEM 650A

ตารางที่ 3.1 Pin Name Type Description

Pin of Mass Flow Controller	Function
1	Max. Power (E)
2	Reflected power (+)
3	Forward power (+)
4	RF ON/OFF Control (+)
5	Power set point (+)
6	+28VDC
7	RF ON (E)
8	Analog remote
9	Overheat (E)
10	Interlock

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Pin of Mass Flow Controller	Function
11	Leveling select
12	Load power (+)
13	+15VDC
14	Max. power (C)
15	Reflected power return (-)
16	Forward power return (-)
17	RF ON/OFF Control (-)
18	Power set point return (-)
19	GND Signal / Chassis ground.
20	20 RF ON (C)
21	GND Signal / Chassis ground.
22	Overheat (C)
23	Interlock
24	Option V User voltage for interface purposes
25	Load power return (-)

Note 1: For all isolated transistor outputs.

Transistor OFF (switch open) - $V_{CEmax} = 40VDC$ ($I_C < 500\mu A$)

Transistor ON (switch closed) - $I_{Cmax} = 10mA$ ($V_{CE} < 1V$)

Note 2: For all ground referenced logic level inputs.

HIGH = 2VDC min to 30VDC max.

LOW = -0.2VDC min to 1VDC max.

ส่วนแรก คือ การเปิดปิดเครื่อง RF ทำได้โดยการ Jump ขา 10 กับขา 23 เข้าด้วยกัน เครื่อง RF จึงจะสามารถเริ่มทำงานได้ และการสั่ง RF ON ต้องป้อนไฟ 15 โวลต์เข้าที่ ขา 4 ของ DB-25

ส่วนที่สุดท้าย คือ Power Setpoint และ Reflect Power Power Setpoint คือขาที่เราต้องจ่ายแรงดันเพื่อควบคุม Power RF โดยใช้แรงดันสูงสุดคือ 10 โวลต์ สำหรับ Power 650 วัตต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การควบคุมในส่วนนี้ใช้วงจรเดียวกับที่ออกแบบให้กับ Massflow Controller โดยเพิ่มวงจรขยายใน
ตอนท้ายให้ขยายแรงดันสูงสุดจาก 5 โวลต์ ให้กลายเป็น 10 โวลต์ Reflect Power บอกถึง Power
ที่ถูกสะท้อนกลับมายังแหล่งกำเนิดการรับค่าสัญญาณ สามารถรับได้โดย Arduino โดยตรง

3.1.4 Rf Generator (CPS)

ตารางรายชื่อ DB-25 Pin ด้านหลังของเครื่อง Rf Generator (CPS) ดังรูป 3.4

ตารางที่ 3.2 Pin Name Type Description

Pin of Mass Flow Controller	Function
1	Remote Setpoint negative
2	Remote Setpoint positive
3	Forward power
4	Reflected power
5	Remote RF On
6	Pulse mode
7	Ground
8	Ground
9	Auxiliary feedback negative
10	Auxiliary feedback positive
11	-
12	-
13	External interlock
14	RF Off light
15	Enable Remote ON
16	Enable Auxiliary Feedback
17	Enable Remote Setpoint
18	-
19	RF On light

Pin of Mass Flow Controller	Function
20	+5V out
21	Enable True Power mode
22	CEX mode
23	-
24	-
25	+12V out for interlock



รูปที่ 3.4 Rf Generator (CPS)

3.1.5 Water Cooler Flow

YF-S201 เป็น Sensor ตรวจจับการไหลของน้ำโดยมีสายไฟ 3 เส้น คือ สายสีดำ เป็น GND สายสีเหลืองเป็น สายส่งสัญญาณออก และสายสีแดงเป็นสายไฟเลี้ยง 5-24 V DC สัญญาณ ที่ได้ออกมาเป็น ความถี่(pulse) และความถี่ที่ออกมาขึ้นกับอัตราการไหลของน้ำ



รูปที่ 3.5 YF-S201 Water Flow Sensor

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Description:

Water flow sensor consists of a plastic valve body, a water rotor, and a hall-effect sensor. When water flows through the rotor, rotor rolls. Its speed changes with different rate of flow. The hall-effect sensor outputs the corresponding pulse signal. This one is suitable to detect flow in water dispenser or coffee machine. We have a comprehensive line of water flow sensors in different diameters. Check them out to find the one that meets your need most.

Features:

Compact, Easy to Install
High Sealing Performance
High Quality Hall Effect Sensor
RoHS Compliant

Specifications:

Working Voltage: DC 4.5V-24V
Normal Voltage: DC 5V-18V
Max. Working Current: 15mA (DC 5V)
Load capacity: $\leq 10\text{ mA}$ (DC 5V)
Flow Rate Range: 1-30L/min
Load Capacity: $\leq 10\text{mK}$ (DC 5V)
Operating Temperature: $\leq 80^\circ\text{C}$
Liquid Temperature: $\leq 120^\circ\text{C}$
Operating Humidity: 35%~90%RH
Allowing Pressure: $\leq 1.7\text{MPa}$
Storage Temperature: $-25\sim+80^\circ\text{C}$
Storage Humidity: 25%~95%RH
Electric strength: 1250V/min
Insulation resistance: $\geq 100\text{M}\Omega$
External threads: 1/2"
Outer diameter: 20mm
Intake diameter: 9mm
Outlet diameter: 13mm

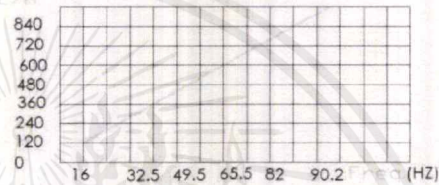
**Application:**

Water heaters, credit card machines, water vending machine, flow measurement device

Circuit:

Red: Positive
Black: GND
Yellow: Output signal

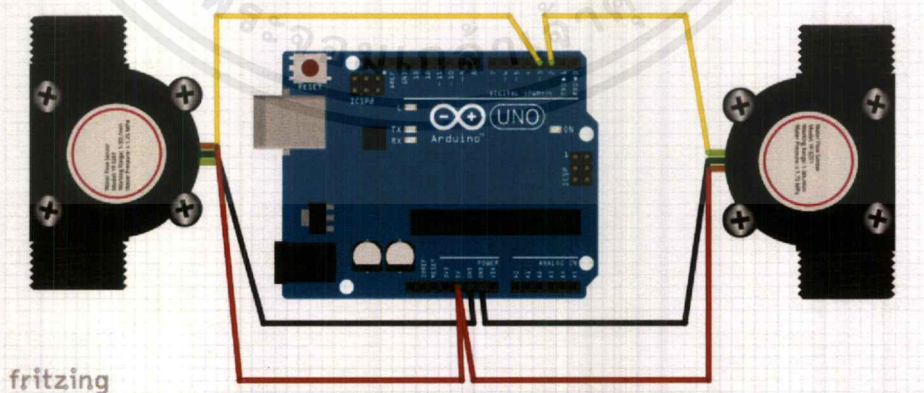
Flow Range: 100L/H~1800L/H		
Flow (L/H)	Pulse (Hz)	Error range
120	10	$\pm 1\%$ $\pm 5\%$
240	20	
360	30	
480	40	
600	50	
720	60	



รูปที่ 3.6 Data Sheet ของ YF-S201

3.1.6 Chiller Temperature

การต่อ sensor YF-S201 กับ Arduino ในที่นี้เราใช้ sensor 2 ตัว ในการวัดสัญญาณที่ทางน้ำไหลกลับของ Chiller นำสายสีเหลืองของ Water Flow ตัวแรกเสียบที่ Port 2 และตัวที่ 2 เสียบที่ Port 3 นำสีแดงเสียบที่ Port 5 V และนำสายสีดำเสียบที่ Port GND ตามรูปที่ 3.10



รูปที่ 3.7วงจรการเชื่อมต่อ YF-S201

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.8 DS18B20 Temperature Sensor

DS18B20 ใช้ตรวจวัดอุณหภูมิของน้ำใน Chiller ที่ปากทางน้ำออก ในการตรวจวัดอุณหภูมิมี 3 สายในการตรวจคือ สายสีแดงเป็นไฟเลี้ยง สายสีดำเป็น GND สายสีแดงเป็นสายสัญญาณ

General Description

The DS18B20 digital thermometer provides 1-wire, 1-Digital Celsius temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18B20 communicates over a 1-wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. In addition, the DS18B20 can derive power directly from the data line ("parasitic power"), eliminating the need for an external power supply.

Each DS18B20 has a unique 64-bit serial code, which allows multiple DS18B20s to function on the same 1-Wire bus. Thus, it is simple to use one microprocessor to control many DS18B20s distributed over a large area. Applications that can benefit from this feature include HVAC, environmental controls, temperature monitoring systems inside buildings, equipment, or machinery, and process monitoring and control systems.

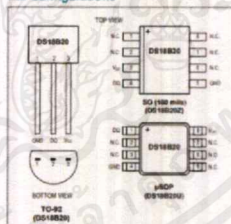
Applications

- Thermostatic Controls
- Industrial Systems
- Consumer Products
- Thermometers
- Thermally Sensitive Systems

Benefits and Features

- Unique 1-Wire® Interface Requires Only One Pin for Communication
- Reduce Component Count with Integrated Temperature Sensor and EEPROM
- Measures Temperatures from -55°C to $+125^{\circ}\text{C}$ (-67°F to $+257^{\circ}\text{F}$)
- $\pm 0.5^{\circ}\text{C}$ Accuracy from -10°C to $+50^{\circ}\text{C}$
- Programmable Resolution from 9 Bits to 12 Bits
- No External Components Required
- Parasitic Power Mode Requires Only 2 Pins for Operation (DQ and GND)
- Simplifies Distributed Temperature-Sensing Applications with Multidrop Capability
 - Each Device Has a Unique 64-Bit Serial Code Stored in On-Chip ROM
- Flexible User-Definable Nonvolatile (NV) Alarm Settings with Alarm Search Command Identifies Devices with Temperature Outside Programmed Limits
- Available in 5-Pin SO (150 mils), 5-Pin μSOP , and 3-Pin TO-92 Packages

Pin Configurations



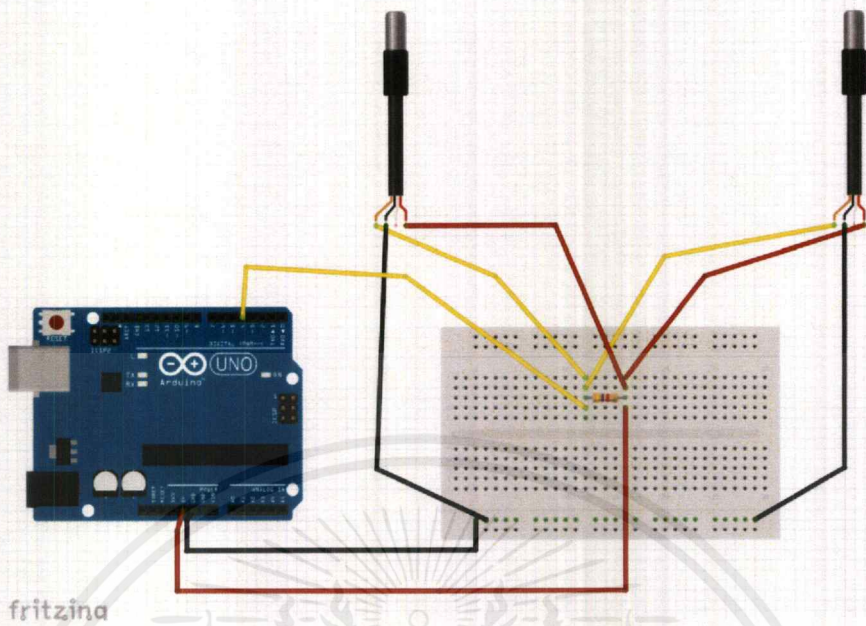
Ordering Information appears at end of data sheet.

1-Wire is a registered trademark of Dallas Integrated Products, Inc.

รูปที่ 3.9 Data Sheet ของ DS18B20

การต่อ Sensor DS18B20 คือ การนำสายสีดำต่อเข้ากับ Port GND สายสีแดงต่อลงที่บอร์ดไขปลา และนำสายสีเหลืองต่อลงบอร์ดไขปลา หลังจากนั้นนำตัวต้านทาน ค่า 4.7 กิโลโอห์มมาต่อขึ้นระหว่าง 2 สายนี้(การต่อแบบนี้เรียกว่าการต่อแบบ Pull Up) และนำสายสีแดงไปต่อลงที่ Port 5V สายสีเหลืองไปต่อเข้ากับ Port 4 ดังรูปที่ 3.12

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้คัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

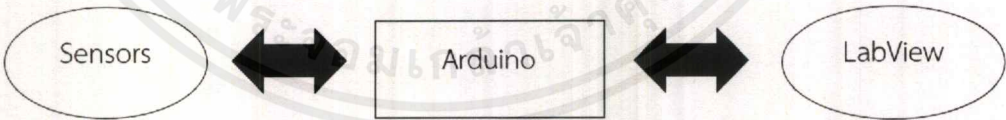


รูปที่ 3.10 วงจรการเชื่อมต่อ DS18B20

ทั้งหมดนี้คือการออกแบบวงจรเพื่อใช้ในการส่งและรับข้อมูลจาก Device และ Sensor ต่าง ๆ ซึ่ง
จังหวะการรับและการส่งข้อมูลนั้น จะถูกควบคุมโดย Arduino

3.2 Arduino With LabView

LabView ทำหน้าที่เป็นหน้าต่าง Interface ที่คอยสั่งการส่วน Arduino เป็นตัว Controller
ได้รับค่ามาจาก LabView และมา ควบคุม ตัว sensor และอุปกรณ์ต่าง ๆ



รูปที่ 3.11 แสดงแผนผังระหว่าง Arduino และ LabView

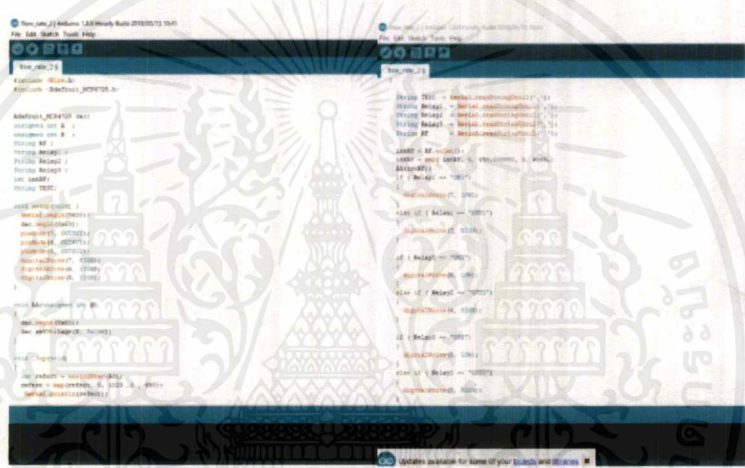
3.2.1 Rf Generator and Relay

ส่วนแรก แสดงการทำงานของโปรแกรม Arduino คือส่วนที่ควบคุมและรับค่าจาก
Rf Generator และ Relay โดยที่ Rf Generator ใช้ 0-10 V ในการควบคุมการทำงาน และมีค่า
Reflect Rf Generator แสดงออกมา โดยในโค้ดที่แสดงในรูปที่ 3.11 คือโค้ดแสดงการสั่งงาน IC D/A

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เนื่องจากต้องการส่งค่าจากโปรแกรม Arduino ที่เป็นค่าดิจิทัล 0-4950 ให้แปลงเป็นค่าอนาล็อก 0-5 V ขึ้นต่อไปคือป้อนค่าไปยังออปแอมป์เพื่อขยายค่าแรงดันจาก 0-5 V ให้แปลงค่าเป็น 0-10 V และ ส่วนของ Reflect Rf Generator ค่าที่โปรแกรม Arduino สามารถรับได้คือ 0-5 V แต่ค่าที่ Reflect Rf Generator ป้อนเข้ามาในโปรแกรมคือจะป้อนค่าตั้งแต่ 0-10 V เพื่อเทียบกับ 0-650 วัตต์ ดังนั้น เราจึงต้องแปลงค่าก่อนที่จะนำค่าที่ได้จาก Reflect Rf Generator เข้าโปรแกรม Arduino โดยการ เพิ่มวงจรแบ่งแรงดันไฟฟ้าเพื่อลดแรงดันจาก 0-10 V ให้กลายเป็น 0-5 V

ส่วนของ Relay เลือกใช้ Relay แบบ Active Low ชนิด 5 V 10 A โดยสามารถเชื่อมต่อกับ โปรแกรม Arduino โดยตรง และนำ Solid State Relay ขึ้นเพื่อให้สามารถทนกระแสไฟได้มากขึ้น

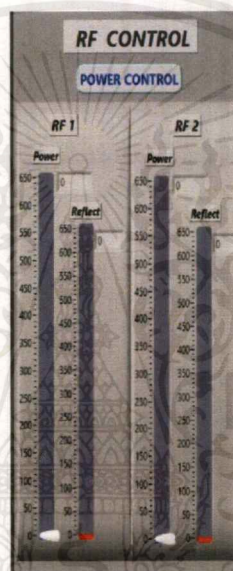


รูปที่ 3.12 โค้ดที่ใช้ในการทำงานของ Rf Generator and Relay

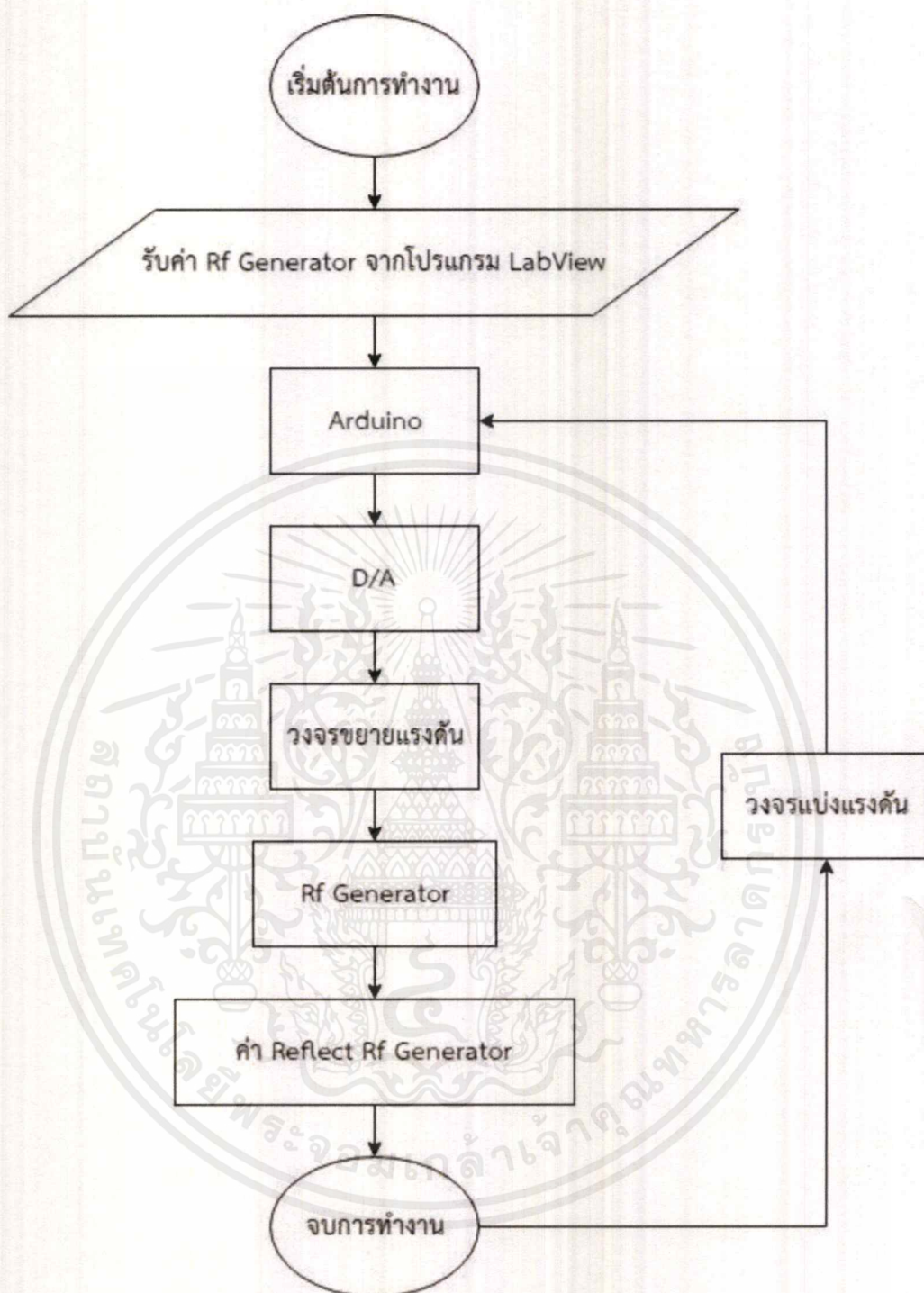
3.2.2 Rf Generator (CPS)

ในส่วนของการแสดงการทำงานของโปรแกรม Arduino ทำงานเหมือนหัวข้อ 3.2.1 แต่แตกต่างที่ในเครื่อง Rf Generator (CPS) เครื่องนี้ใช้แรงดัน 0-5 V จึงไม่จำเป็นต้องผ่านตัวออปแอมป์ สามารถเชื่อมต่อกับโปรแกรม Arduino ได้โดยตรง

ในส่วนของโปรแกรม LabView คือส่วนของการแสดงค่า Reflect Rf Generator และป้อนค่า Rf Generator ของทั้งสองส่วนเพื่อส่งไปยังโปรแกรม Arduino

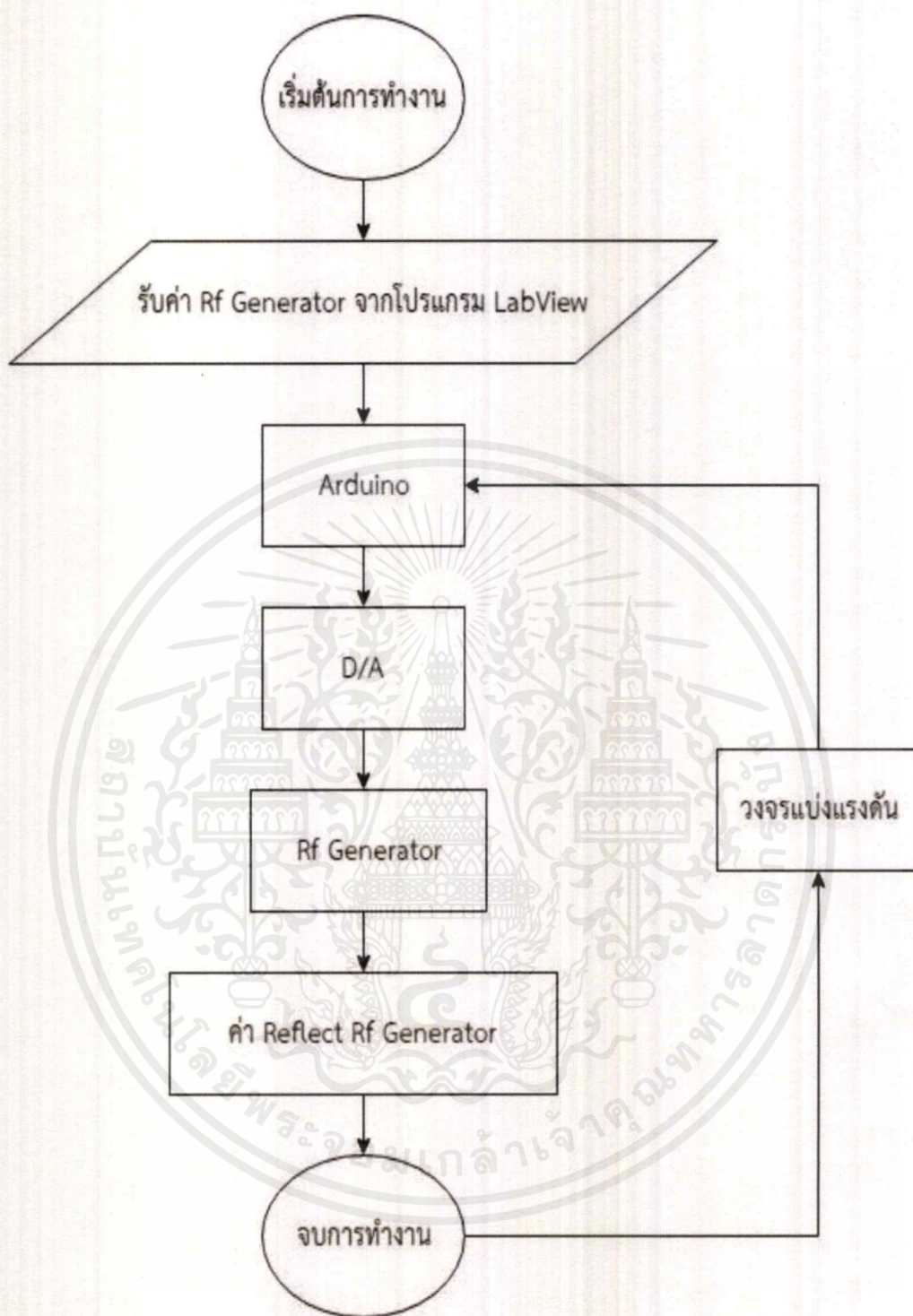


รูปที่ 3.13 โปรแกรม LabView ในส่วนของ Rf Generator



รูปที่ 3.14 ไตอะแกรมการทำงานของ Rf Generator

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

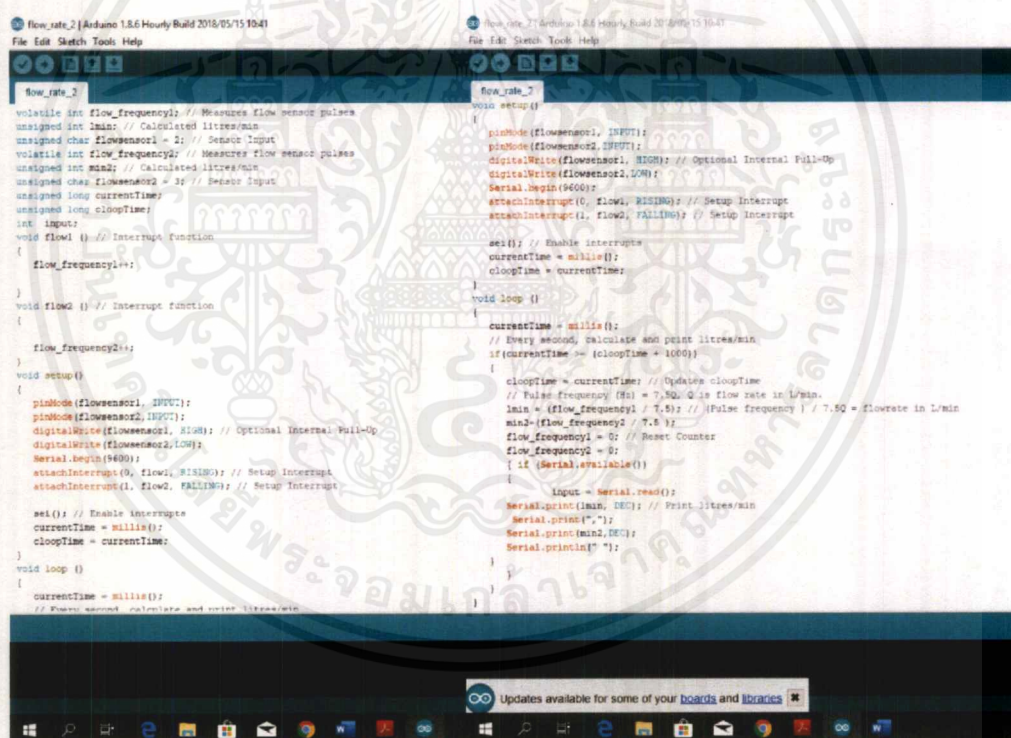


รูปที่ 3.15 ไดอะแกรมการทำงานของ Rf Generator (CPS)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.3 Flow Rate and temperature

ในส่วนของโปรแกรม Arduino ดังรูปที่ 3.14 แสดงโค้ดการทำงานของ Flow Rate โดยใช้หลักการของ Hall Effect Sensor และ อัตราการไหลแปรผันตรงกับความเร็ว โดยการส่งสัญญาณดิจิตอลออกมา ซึ่งวัดอัตราการไหลได้จากความเร็วของพัลส์สัญญาณ จากนั้นนำสัญญาณความเร็วที่ได้มาเข้าสมการแปลงหน่วยในโปรแกรม Arduino และการทำงานของ Sensor Temperature ใช้ IC DS18D20 ซึ่งเป็น IC ที่สามารถส่งค่าอุณหภูมิออกมาเป็นดิจิตอล ซึ่งมีความละเอียด 12 บิต โดยที่ใช้สัญญาณของขาออกเข้ามาเป็น Input แล้วสามารถทำให้ประหยัดพื้นที่ของไมโครคอนโทรลเลอร์โดยการเลือก Address ของ DS18D20 แต่ละตัวได้



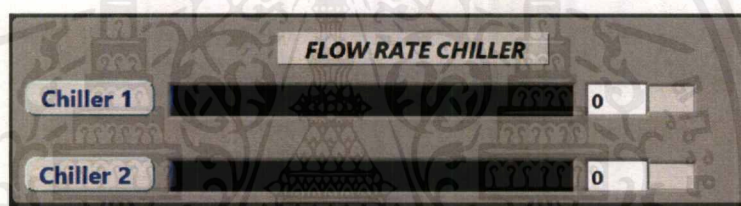
รูปที่ 3.16 โค้ดที่ใช้ในการทำงานของ Flow Rate และ temperature

ในส่วนของโปรแกรม LabView มีหน้าที่แสดงผลการทำงานของ Flow Rate ในหน่วย L/Min และมีส่วนที่เพิ่มเติมมาคือ DS18D20 เขียนให้โปรแกรม LabView เตือนด้วยเสียงและมีแสงสีแดงแจ้งเตือนที่ปุ่ม Alarm ตามอุณหภูมิที่สามารถกำหนดเอง ในที่นี้กำหนดอยู่ที่ 25 องศาเซลเซียส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



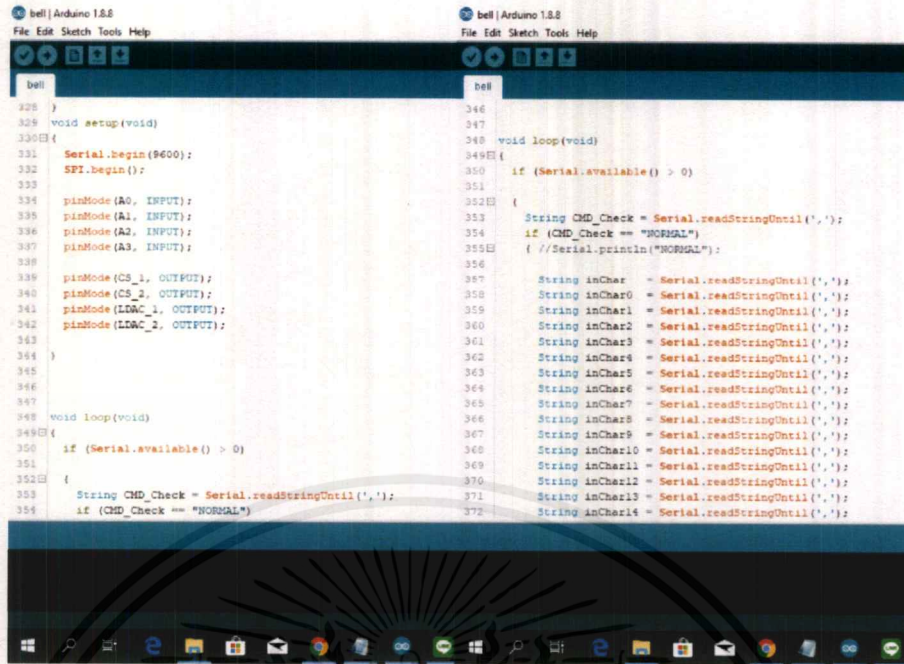
รูปที่ 3.17 โปรแกรม LabView ในส่วนของ DS18D20



รูปที่ 3.18 โปรแกรม LabView ในส่วนของ Flow Rate

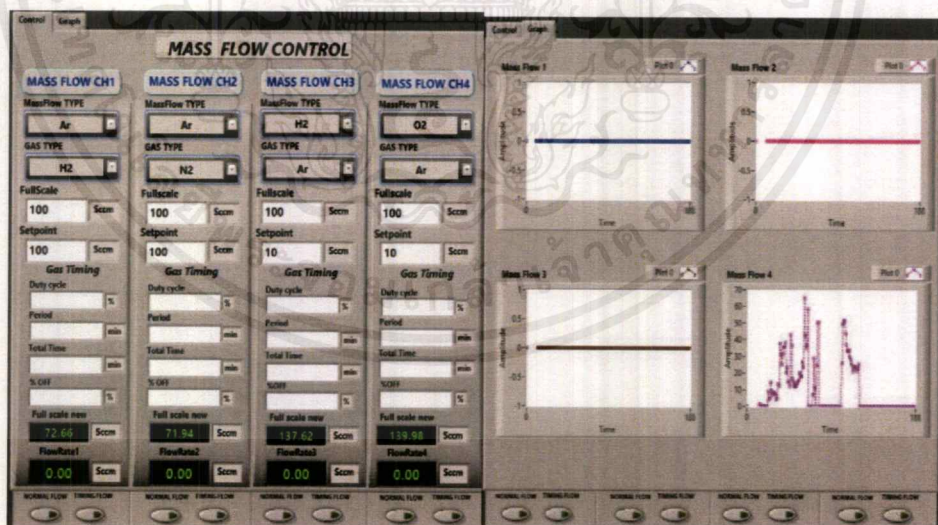
3.2.4 Mass Flow

ในส่วนของโปรแกรม Arduino คือการรับ-ส่งค่าและคำนวณค่าที่ป้อนเข้ามาในระบบ หลักการทำงานคือเริ่มต้นจากการรับค่าจากโปรแกรม LabView และส่งมายังโปรแกรม Arduino เพื่อมาเข้าสมการคำนวณค่า Set Point ส่งต่อไปยัง MCP 4922 แปลงแรงดัน 0-5 V อนุาล็อกส่งกลับไปที่ Mass Flow และจาก Mass Flow ส่งกลับไปที่โปรแกรม Arduino จะแปลงค่าสัญญาณอนุาล็อกเป็นดิจิตอลและคำนวณเป็นค่าอัตราการไหล



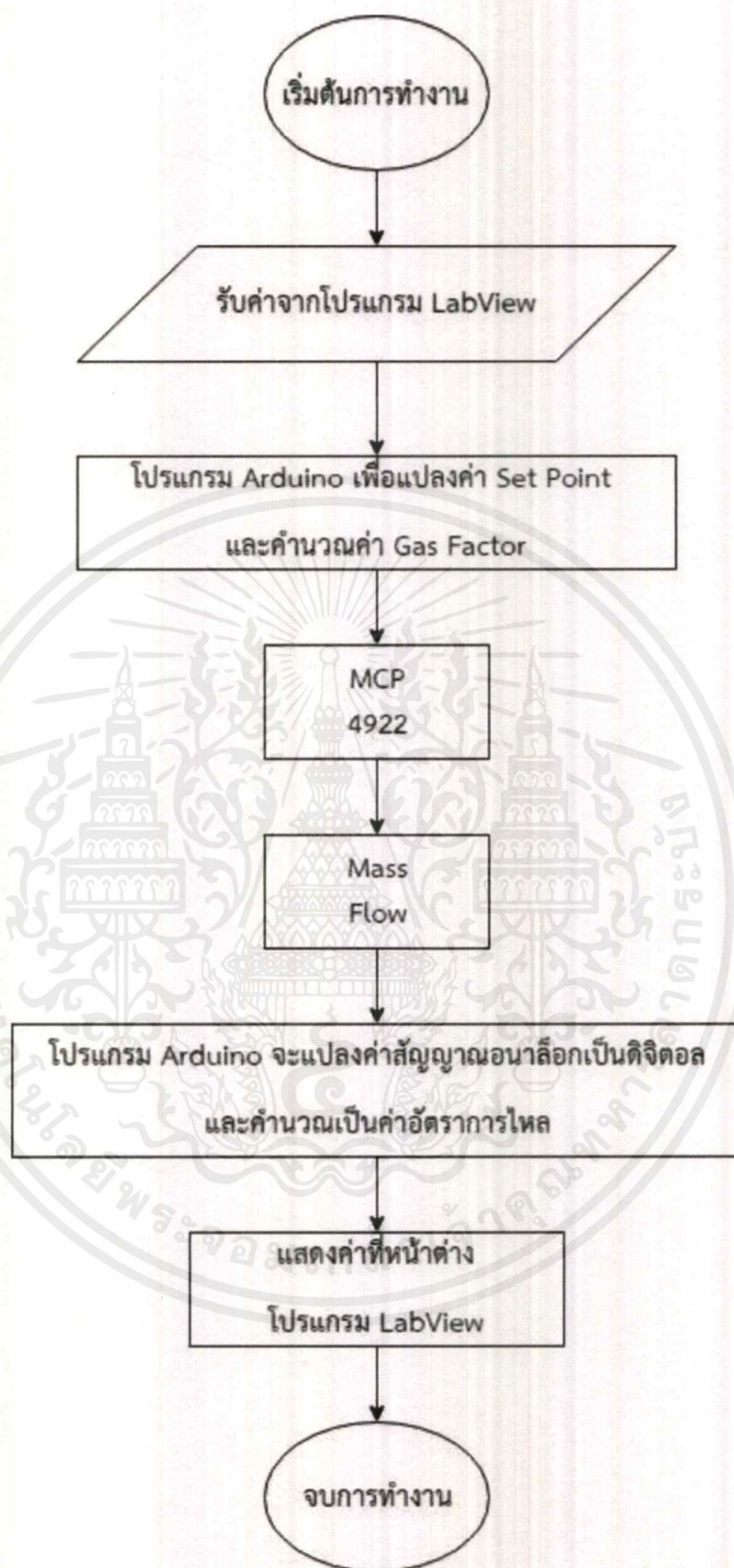
รูปที่ 3.19 โค้ดที่ใช้ในการทำงานของ Mass Flow

ในส่วนของโปรแกรม LabView มีหน้าที่รับค่าของชนิด Mass Flow ค่า Full Scale ค่า Set Point และแสดงค่าดังรูปที่ 3.18



รูปที่ 3.20 โปรแกรม LabView ในส่วนของ Mass Flow

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

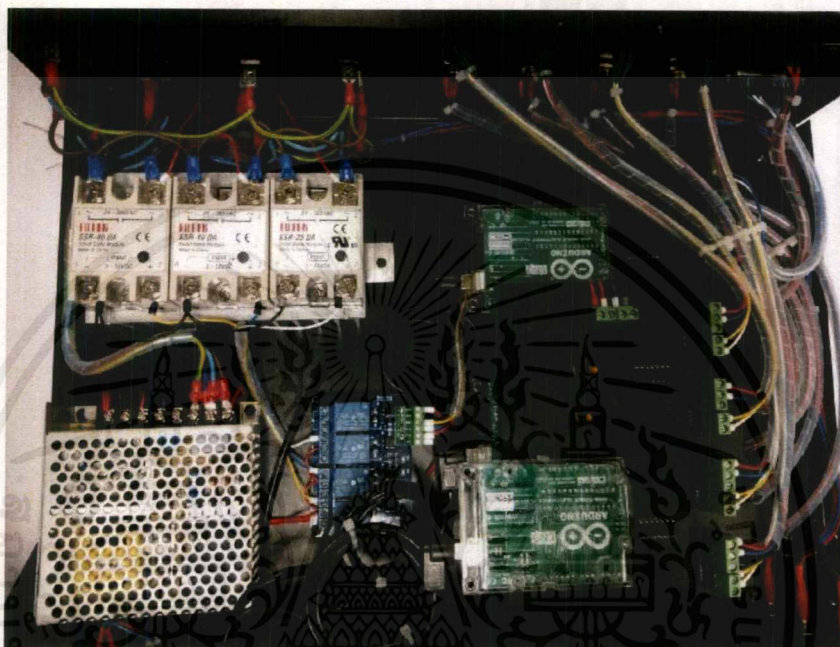


รูปที่ 3. 21 ไตอะแกรมการทำงานของ Mass Flow

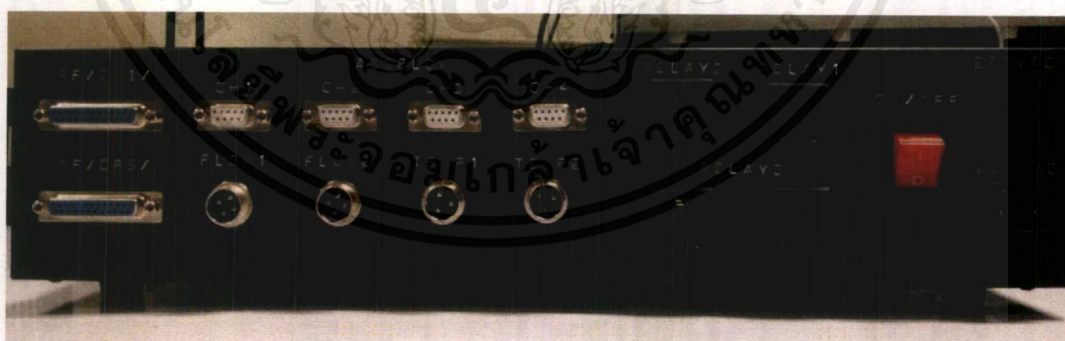
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3 ระบบควบคุมระบบ Rf Magnetron Sputtering

เครื่องควบคุมระบบ RF Magnetron Sputtering ที่ใช้ในการทดลองนี้เป็นเครื่องมือที่ได้รับการออกแบบและสร้างขึ้นมา โดยสามารถควบคุมระบบทั้งหมดได้ผ่านทางโปรแกรม LabView ทั้งหมดและแสดงผลผ่านทางหน้าจอสัมผัส ซึ่งเครื่องนี้มีลักษณะดังนี้



รูปที่ 3.22 โครงสร้างภายในของเครื่องควบคุมระบบ Rf Magnetron Sputtering



รูปที่ 3.23 ด้านหลังของเครื่องควบคุมระบบ Rf Magnetron Sputtering

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

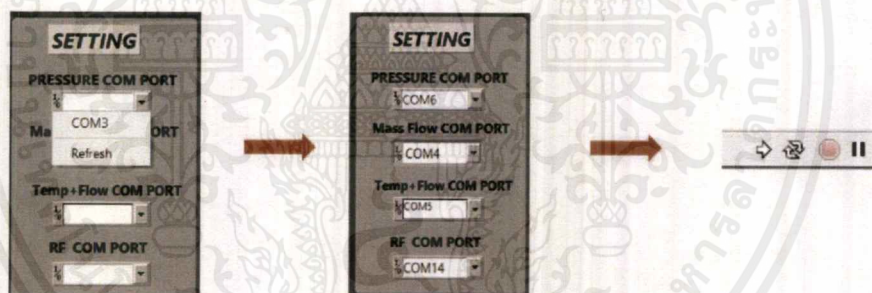


รูปที่ 3.24 ด้านหน้าของเครื่องควบคุมระบบ Rf Magnetron Sputtering

3.4 ขั้นตอนการทดลอง

3.4.1 ขั้นตอนที่ 1

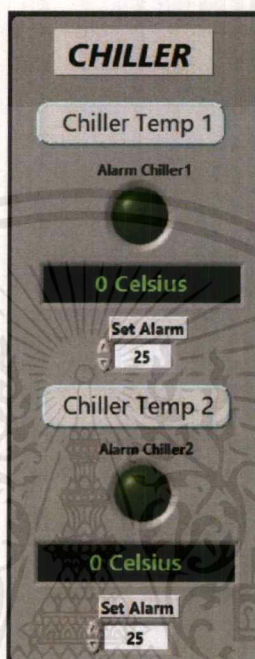
เมื่อเปิดโปรแกรม LabView เสร็จเรียบร้อย ให้ทำการเลือก Comport ของแต่ละชนิดให้ตรงตามตัวของบอร์ด Arduino แล้วจึงกดเริ่มโปรแกรม



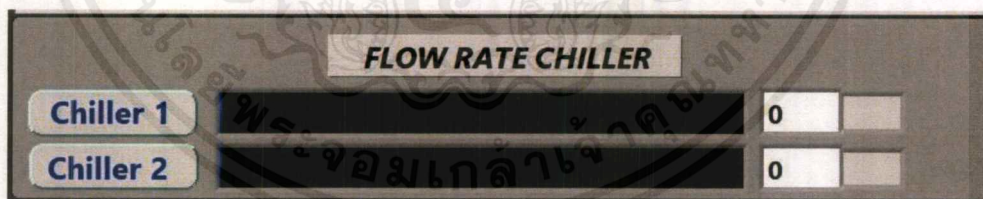
รูปที่ 3.25 วิธีการเลือก Com Port

3.4.2 ขั้นตอนที่ 2

จากรูปที่ 3.26 แสดงถึงการตรวจสอบอุณหภูมิของน้ำของแต่ละ Chiller ที่มีไฟแสดงสถานะเมื่ออุณหภูมิของ Chiller ถึงค่าที่กำหนดและจากรูปที่ 3.27 แสดงถึงอัตราการไหลของน้ำของแต่ละ Chiller บนหน้าจอแสดงผล



รูปที่ 3.26 อุณหภูมิน้ำใน Chiller

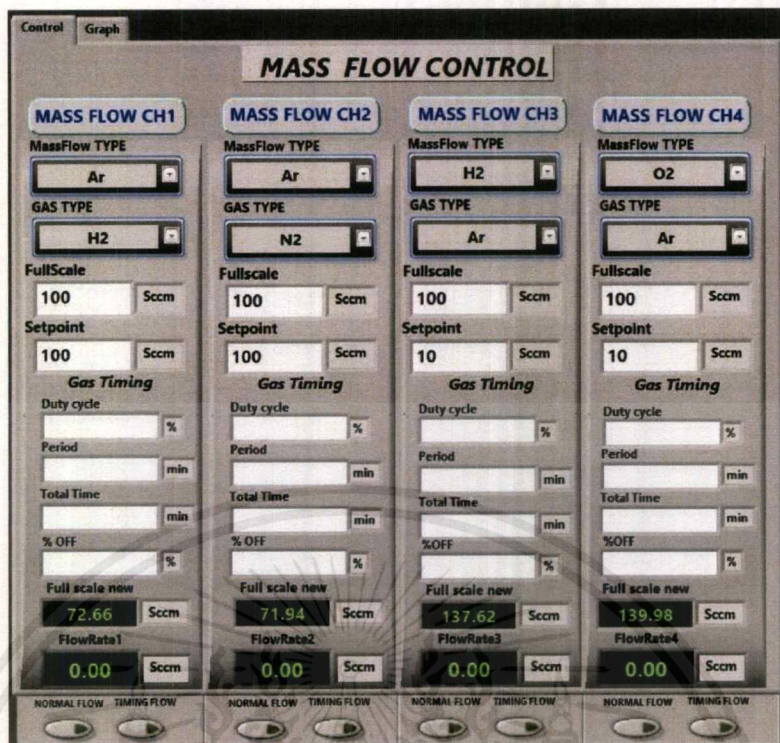


รูปที่ 3.27 อัตราการไหลออกของน้ำใน Chiller

3.4.3 ขั้นตอนที่ 3

ในส่วนของ Mass Flow Control สามารถควบคุมได้ 2 แบบ คือ แบบปกติและแบบ Gas Timing ซึ่งแบบปกติสามารถควบคุมโดยการกดเลือก MassFlow Type , Gas Type และทำการป้อนค่า Full Scale Setpoint ของ Mass Flow แต่ละตัวและในส่วนของ Gas Timing สามารถกำหนดค่า Duty Cycle , Period , Total Time และ %Off ตามลำดับดังรูปที่ 3.28

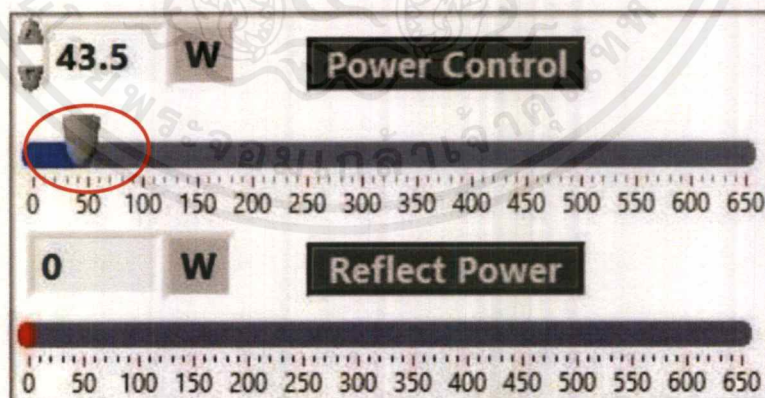
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.28 ป้อนข้อมูลของ Mass Flow

3.4.4 ขั้นตอนที่ 4

ตรวจสอบความดันบนหน้าจอแสดงผล เมื่อมีความดันในระดับที่ต้องการแล้ว ให้ทำการกดปุ่มเพื่อเลื่อนเปิดเครื่อง Rf Magnetron หรือกำหนดค่า Power ที่ต้องการดังรูปที่ 3.29



รูปที่ 3.29 การเพิ่ม Rf Magnetron เข้าในระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4

ผลการทดลองและการอภิปรายผล

จากงานโครงการพิเศษนี้ได้ทำการทดสอบระบบโดยการ Operate ทั้งระบบพร้อมกัน และเคลือบฟิล์มขึ้นมาโดยใช้ระบบควบคุมที่สร้างขึ้น

4.1 ผลการทดลองที่ 1

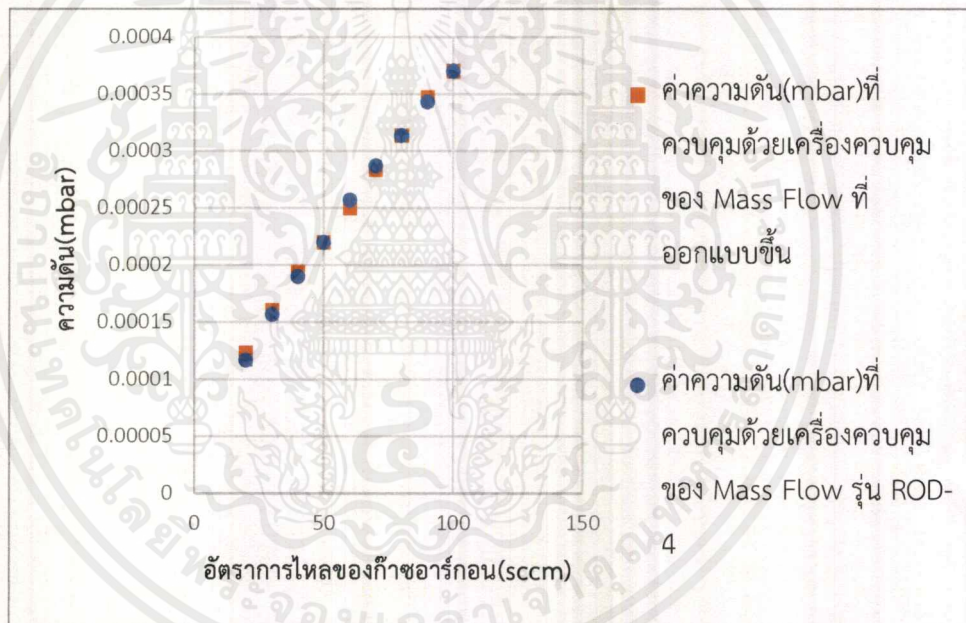
เมื่อนำเครื่องควบคุมอัตราการไหลของก๊าซที่สร้างขึ้นไปทดสอบควบคุมอัตราการไหลของก๊าซเข้าระบบสุญญากาศของ RF Magnetron Sputtering แล้วดูค่าความดันที่ได้เปรียบเทียบกับค่าความดันที่ได้จากการควบคุมอัตราการไหลด้วยเครื่องควบคุมอัตราการไหล รุ่น ROD-4 ได้ผลการทดลองดังนี้

ตารางที่ 4.1 ผลการทดลองการควบคุมอัตราการไหลของก๊าซอาร์กอน (Ar) โดยใช้ Mass Flow ของ Ar กับเครื่องควบคุมอัตราการไหลที่สร้างขึ้นและใช้ Mass Flow ของ H_2 กับเครื่องควบคุมอัตราการไหล รุ่น ROD - 4)

ค่า อัตรา การไหล (sccm)	ค่าความดันที่ควบคุมด้วยเครื่องควบคุม ของ Mass Flow รุ่น ROD-4 ($\times 10^{-3}$ mbar)				ค่าความดันที่ควบคุมด้วยเครื่องควบคุม ของ Mass Flow ที่ออกแบบ ($\times 10^{-3}$ mbar)			
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	$\bar{X} \pm \text{Uncertainty}$	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	$\bar{X} \pm \text{Uncertainty}$
20	0.12	0.11	0.12	0.12 ± 0.00	0.12	0.13	0.12	0.12 ± 0.00
30	0.16	0.16	0.15	0.16 ± 0.00	0.16	0.16	0.16	0.16 ± 0.00
40	0.19	0.19	0.19	0.19 ± 0.00	0.2	0.19	0.19	0.19 ± 0.00
50	0.22	0.22	0.22	0.22 ± 0.00	0.22	0.22	0.22	0.22 ± 0.00
60	0.25	0.26	0.26	0.26 ± 0.00	0.25	0.25	0.25	0.25 ± 0.00

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ค่า อัตรา การไหล (sccm)	ค่าความดันที่ควบคุมด้วยเครื่องควบคุม ของ Mass Flow รุ่น ROD-4 ($\times 10^{-3}$ mbar)				ค่าความดันที่ควบคุมด้วยเครื่องควบคุม ของ Mass Flow ที่ออกแบบ ($\times 10^{-3}$ mbar)			
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	$\bar{X} \pm \text{Uncertainty}$	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	$\bar{X} \pm \text{Uncertainty}$
70	0.28	0.29	0.29	0.29 ± 0.00	0.28	0.28	0.29	0.28 ± 0.00
80	0.31	0.31	0.32	0.31 ± 0.00	0.31	0.31	0.32	0.31 ± 0.00
90	0.34	0.34	0.35	0.34 ± 0.00	0.34	0.35	0.35	0.35 ± 0.00
100	0.37	0.37	0.37	0.37 ± 0.00	0.37	0.37	0.37	0.37 ± 0.00



รูปที่ 4.1 กราฟความสัมพันธ์ระหว่างอัตราการไหลของก๊าซอาร์กอน (sccm) กับ ค่าความดัน (mbar)

4.1.1 วิเคราะห์ผลการทดลอง

จากการทดลองเพื่อดูค่าความดันในระบบพบว่าเมื่อเปรียบเทียบกับระหว่างกล่องที่สร้างขึ้นกับเครื่องควบคุมอัตราการไหล รุ่น ROD - 4 เมื่อเพิ่มอัตราการไหลในระบบค่าที่ได้มีค่าใกล้เคียงกัน

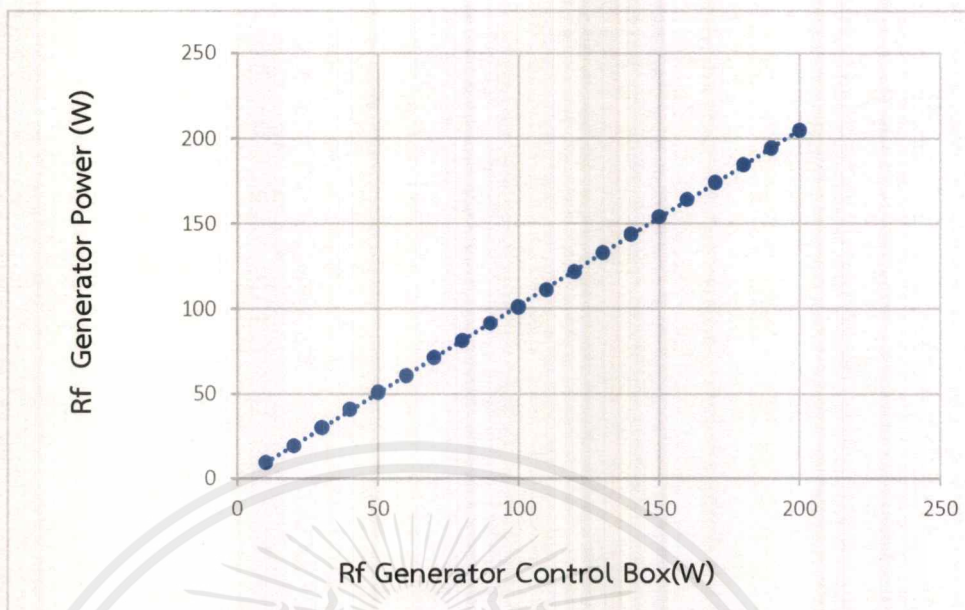
4.2 ผลการทดลองที่ 2

เมื่อป้อนค่า Power = 10,20,30,...,200 W แล้วเปรียบเทียบค่าจากหน้าจอของเครื่อง Rf Generator Power

ตารางที่ 4.2 ผลการทดลองของ Rf Generator Power (W) และ Rf Generator Control Box(W)

Rf Generator Control Box (W)	Rf Generator Power (W)				เปอร์เซ็นต์ความ แตกต่าง (%)
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	$\bar{X}$	
10	9.00	10.00	9.00	9.33	6.89
20	19.00	20.00	19.00	19.33	3.38
30	30.00	30.00	30.00	30.00	0.00
40	41.00	40.00	42.00	41.00	2.46
50	51.00	50.00	52.00	51.00	1.98
60	61.00	60.00	61.00	60.67	1.10
70	72.00	70.00	72.00	71.33	1.80
80	82.00	80.00	82.00	81.33	1.65
90	92.00	90.00	93.00	91.67	1.83
100	102.00	100.00	102.00	101.33	1.32
110	112.00	110.00	112.00	111.33	1.20
120	123.00	120.00	123.00	122.00	1.65
130	134.00	130.00	135.00	133.00	2.28
140	146.00	140.00	146.00	144.00	2.81
150	157.00	150.00	156.00	154.33	2.84
160	167.00	160.00	167.00	164.67	2.87
170	177.00	170.00	177.00	174.67	2.70
180	188.00	180.00	187.00	185.00	2.73
190	198.00	190.00	196.00	194.67	2.42
200	209.00	200.00	206.00	205.00	2.46

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

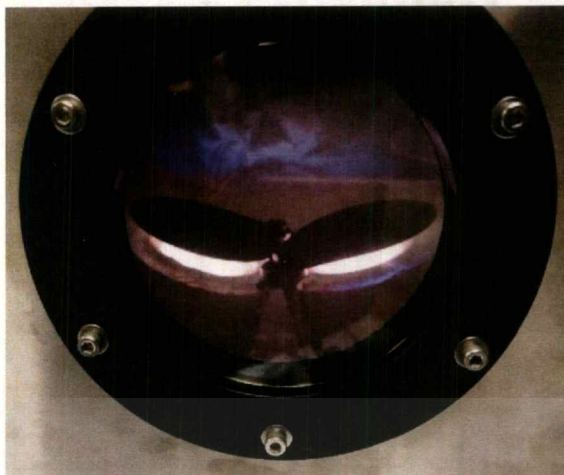


รูปที่ 4.2 กราฟแสดงความสัมพันธ์ระหว่าง Rf Generator Power (W) กับ Rf Generator Control Box(W)

4.2.1 วิเคราะห์ผลการทดลอง

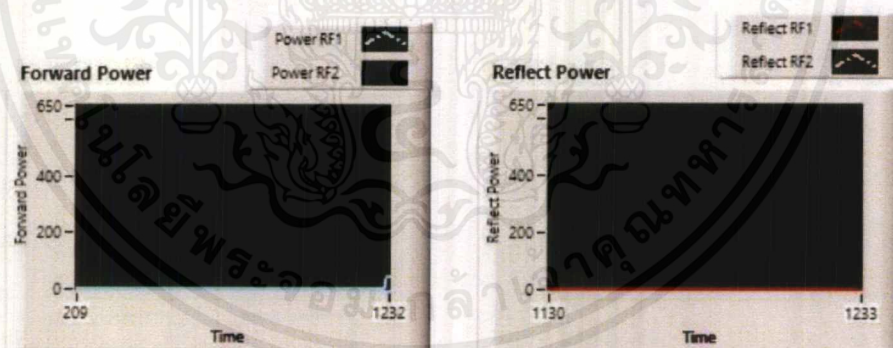
Rf Generator Control Box กับ Rf Generator Power (W) ซึ่งมีค่าสูงสุด 6.89 เปอร์เซ็นต์ และมีค่าต่ำสุด 0.00 เปอร์เซ็นต์

จากรูปที่ 4.3 ระบบ Rf Magnetron Sputtering Control System Via Touch Screen Display สามารถจู่หัว Magnetron ทั้งสองหัวได้พร้อมกันภายในระบบสุญญากาศได้สำเร็จ



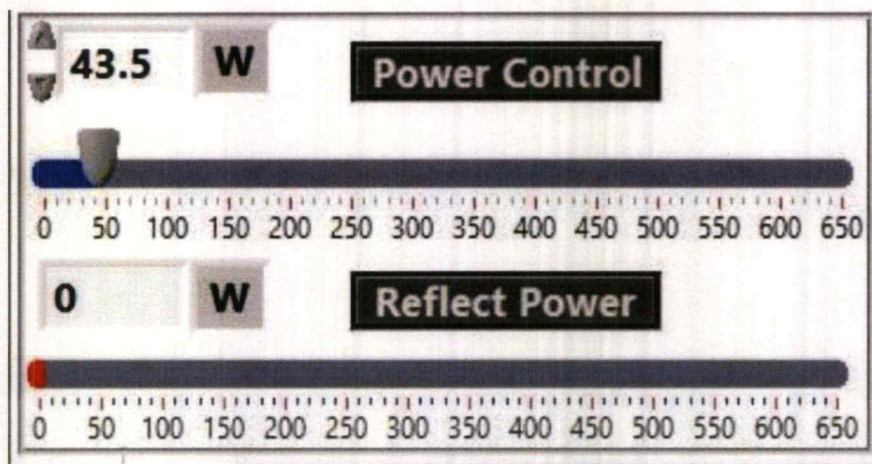
รูปที่ 4.3 ภายในระบบสุญญากาศ

จากรูปที่ 4.4 และรูปที่ 4.5 จอแสดงผลหน้า Front Panel ของโปรแกรม LabView ในส่วน Rf Magnetron สามารถควบคุม Rf Generator Power ได้ตามที่ต้องการมีอัตราการเพิ่มและลดไม่ช้าเกินไป ค่า Reflect Rf Generator สามารถแสดงได้ถูกต้องแม่นยำ เมื่อเปรียบเทียบกับมิเตอร์ที่แสดงผลอยู่หน้าเครื่อง Rf Magnetron



รูปที่ 4.4 กราฟ Rf Generator และ Reflect Rf Generator

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.5 หน้า Front Panel ของ ระบบ Rf Generator และ Reflect Rf Generator

4.3 ผลการทดลองที่ 3

ในส่วนความสามารถในการวัดอุณหภูมิ จากการทดสอบเมื่อเราเปรียบเทียบค่าที่แสดงผลทางหน้าจอสัมผัสกับเทอร์โมคอปเปอร์ ดังรูป 4.6 ค่ามีความใกล้เคียงกัน และสามารถแจ้งเตือนเมื่อมีอุณหภูมิมีค่าสูงกว่าอุณหภูมิที่กำหนด ดังรูป 4.7



รูปที่ 4.6 เปรียบเทียบระหว่างหน้าจอแสดงผล กับ เทอร์โมคอปเปอร์



รูปที่ 4.7 ไฟแจ้งเตือนของระบบ การวัดอุณหภูมิ

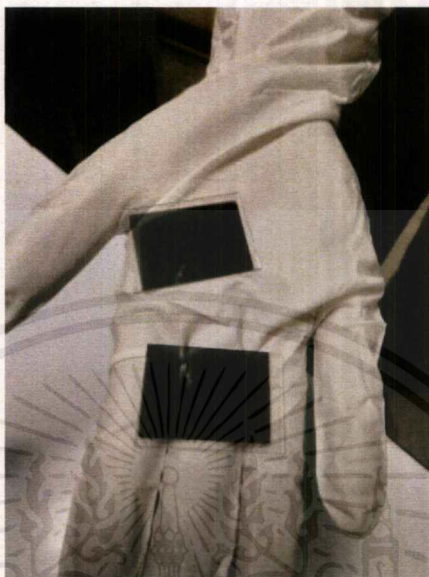
4.4 ผลการทดลองที่ 4

ในส่วนของการวัด Flow Rate จากการทดลองพบว่าสามารถแสดงผลอัตราการไหลของน้ำได้แบบทันทีทันใด ดังรูป 4.7 แสดงถึงอัตราการไหลของน้ำภายในเครื่อง Chiller ในแต่ละตัว



รูปที่ 4.8 อัตราไหลของน้ำที่หน้า Front Panel

จากรูปที่ 4.8 ระบบ Rf Magnetron Sputtering Control System Via Touch Screen Display สามารถปลูกฟิล์มออกมาได้จริงตามเงื่อนไขของผู้ทดลองได้ตามที่ต้องการ



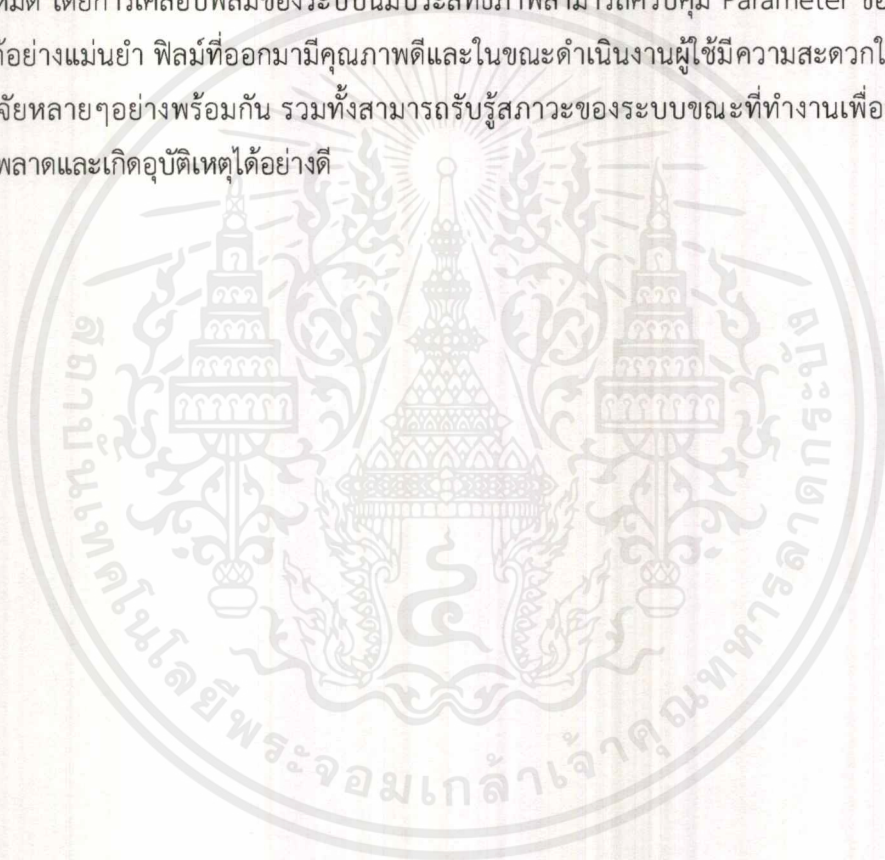
รูปที่ 4.9 ฟิล์มที่ปลูกจากระบบ Rf Magnetron Sputtering Control System Via Touch Screen Display

บทที่ 5

สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

ในโครงการพิเศษนี้ได้ศึกษา ทดลองเพื่อสร้างระบบควบคุมอาร์เอฟแมกนีตรอนสเปกโตรริงผ่านหน้าจอสัมผัส ทำให้ง่ายต่อการใช้งานมากขึ้นโดยสิ่งที่สร้างขึ้นมาคือ ระบบควบคุม Rf Magnetron ระบบควบคุม Mass Flow ระบบแสดงค่า Flow Rate และระบบแสดงค่า Temperature ระบบทั้งหมดนี้ควบคุมและแสดงผลผ่านโปรแกรม Arduino และโปรแกรม LabView ทั้งหมด โดยการเคลือบฟิล์มของระบบนี้มีประสิทธิภาพสามารถควบคุม Parameter ของอุปกรณ์ต่างๆได้อย่างแม่นยำ ฟิล์มที่ออกมามีคุณภาพดีและในขณะดำเนินงานผู้ใช้มีความสะดวกในการควบคุมปัจจัยหลายๆอย่างพร้อมกัน รวมทั้งสามารถรับรู้สภาวะของระบบขณะที่ทำงานเพื่อป้องกันความผิดพลาดและเกิดอุบัติเหตุได้อย่างดี



เอกสารอ้างอิง

บริษัท วินัส ซัพพลาย จำกัด.//2017.//บทความ Arduino คืออะไร ตอนที่1แนะนำเพื่อนใหม่ที่ชื่อ Arduino.

[Online]. // Available: <https://www.thaieasyelec.com/article-wiki/basic-electronics/%E0%B8%9A%E0%B8%97%E0%B8%84%E0%B8%A7%E0%B8%B2%E0%B8%A1-arduino-arduino.html>.

LabView.//2017.// บทที่ 2 ทฤษฎีและหลักการ 2.1 โปรแกรม LabVIEW

[Online].// Available: http://www.research-system.siam.edu/images/coop/DESIGN_AND_CONSTRUCTION_OF_ELECTRICAL_MEASUREMENT_USING_LABVIEW_PROGRAM/ch2.pdf

พระสา บุญดีขวัญกุล และ ลักษณะ ตั้งอภิรักษ์กุล. 2560. “การพัฒนาระบบการวัดและควบคุม

อัตราการไหลของก๊าซด้วยไมโครคอนโทรลเลอร์” วิทยาศาสตร์บัณฑิต สาขาวิชาฟิสิกส์
ประยุกต์ ภาควิชาฟิสิกส์ คณะวิทยาศาสตร์, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร
ลาดกระบัง.

นายศุภวัฒน์ ยิ่งแจ่มศิริ และ นางสาวศิริญา สมลือแสน.2557. “ การพัฒนาระบบควบคุม

อาร์เอฟแมกนีตรอนสปีดเตอร์ริง ผ่านหน้าจอสัมผัส”วิทยาศาสตร์บัณฑิต สาขาวิชาฟิสิกส์
ประยุกต์ ภาควิชาฟิสิกส์ คณะวิทยาศาสตร์, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร
ลาดกระบัง.



ภาคผนวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#include <Wire.h>
#include <Adafruit_MCP4725.h>
```

```
Adafruit_MCP4725 dac;
```

```
unsigned int A ;
```

```
unsigned int B ;
```

```
String RF ;
```

```
String Relay1 ;
```

```
String Relay2 ;
```

```
String Relay3 ;
```

```
int intRF;
```

```
String TEST;
```

```
void setup(void) {
```

```
    Serial.begin(9600);
```

```
    dac.begin(0x60);
```

```
    pinMode(7, OUTPUT);
```

```
    pinMode(6, OUTPUT);
```

```
    pinMode(5, OUTPUT);
```

```
    digitalWrite(7, HIGH);
```

```
    digitalWrite(6, HIGH);
```

```
    digitalWrite(5, HIGH);
```

```
}
```

```
void AA(unsigned int B)
```

```
{
```

```
    dac.begin(0x60);
```

```
    dac.setVoltage(B, false);
```

```
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

void loop(void)
{
    int relect = analogRead(A0);
    relect = map(relect, 0, 1023 ,0 , 650);
    Serial.println(relect);

    if (Serial.available() > 0)
    {

        String TEST  = Serial.readStringUntil(';');
        String Relay1 = Serial.readStringUntil(';');
        String Relay2 = Serial.readStringUntil(';');
        String Relay3 = Serial.readStringUntil(';');
        String RF      = Serial.readStringUntil(' ');

        intRF = RF.toInt();
        intRF = map( intRF, 0, 650.000000, 0, 4095);
        AA(intRF);
        if ( Relay1 == "ON1")
        {
            digitalWrite(7, LOW);
        }
        else if ( Relay1 == "OFF1")
        {
            digitalWrite(7, HIGH);
        }

        if ( Relay2 == "ON2")
        {
            digitalWrite(6, LOW);
        }
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

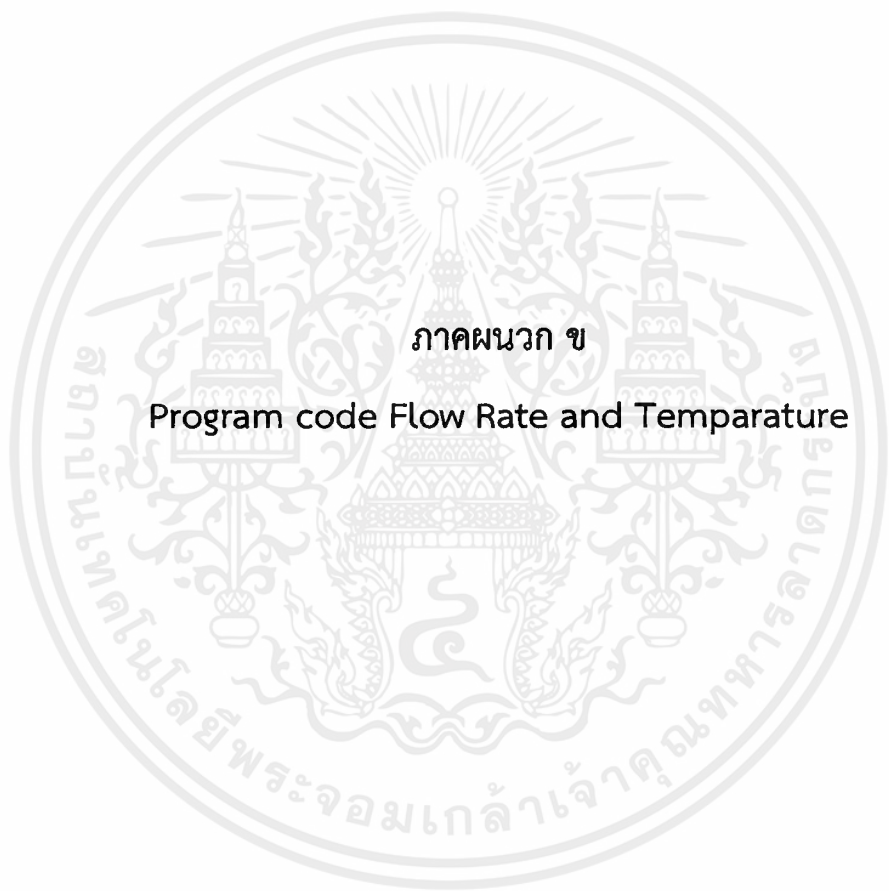
```

else if ( Relay2 == "OFF2")
{
    digitalWrite(6, HIGH);
}

if ( Relay3 == "ON3")
{
    digitalWrite(5, LOW);
}
else if ( Relay3 == "OFF3")
{
    digitalWrite(5, HIGH);
}
}
}

```





ภาคผนวก ข

Program code Flow Rate and Temperature

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

//Temp///
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
float tempC;
int input;

////flow///
int flowPin = 2;
int flowPin1 = 3;
double flowRate;
double flowRate1;
volatile int count;
volatile int count1;

////Temp address///
DeviceAddress T1 = { 0x28, 0xB2, 0xEC, 0x46, 0x92, 0x08, 0x02, 0xC8 };
DeviceAddress T2 = { 0x28, 0x59, 0x14, 0x46, 0x92, 0x0D, 0x02, 0x7E };

void setup() {
  Serial.begin(9600); //Start Serial

  sensors.setResolution(T1, 10);
  sensors.setResolution(T2, 10);

  pinMode(flowPin, INPUT);
  attachInterrupt(0, Flow, RISING);
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

pinMode(flowPin1, INPUT);
attachInterrupt(1, Flow1, RISING);

}

void printTemperature(DeviceAddress deviceAddress)
{
    float tempC = sensors.getTempC(deviceAddress); //อ่านtemp
    Serial.print (tempC);
}

void loop() {
    if (Serial.available()) {
        input = Serial.read();
        sensors.requestTemperatures();

        count = 0;
        count1 = 0;
        interrupts();
        delay (1000);
        noInterrupts();

        flowRate = (count * 2.25);
        flowRate = flowRate * 60;
        flowRate = flowRate / 1000;
        flowRate1 = (count1 * 2.25);
        flowRate1 = flowRate1 * 60;
        flowRate1 = flowRate1 / 1000;
        Serial.print(flowRate);
        Serial.print(",");
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

Serial.print(flowRate1);
Serial.print(",");
printTemperature(T1);
Serial.print(",");
printTemperature(T2);
Serial.println("");
}
}

```

```

void Flow()
{
    count++;
}
void Flow1()
{
    count1++;
}

```



ภาคผนวก ค

Program code Mass Flow

```

#include <SPI.h>
#include "Arduino.h"
#define CS_1 9
#define CS_2 8
#define LDAC_1 3
#define LDAC_2 2
float  bitweighttovoltage1;
float  bitweighttovoltage2;
float  bitweighttovoltage3;
float  bitweighttovoltage4;
float  fullscalenew1;
float  fullscalenew2;
float  fullscalenew3;
float  fullscalenew4;
float  voltunit1;
float  voltunit2;
float  voltunit3;
float  voltunit4;
float  flowrate1;
float  flowrate2;
float  flowrate3;
float  flowrate4;
float  FS1;
float  SP1;
float  GF1;
float  GM1;
int    volt1;
int    volt2;
int    volt3;
int    volt4;
float  FS2;
float  SP2;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
float GF2;
float GM2;
float FS3;
float SP3;
float FS4;
float SP4;
float GF3;
float GM3;
float GF4;
float GM4;
float analogmassflow1;
float analogmassflow2;
float analogmassflow3;
float analogmassflow4;
float spn1;
float spn2;
float spn3;
float spn4;
float AVGANALOGMASSFLOW1;
float AVGANALOGMASSFLOW2;
float AVGANALOGMASSFLOW3;
float AVGANALOGMASSFLOW4;
float x1;
float x2;
float x3;
float x4;
float x5;
float x6;
float x7;
float x8;
float x9;
float x10;
float FIXVOLTAGE1 = 0.04;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

float OFFSET1 = 0.00;
float FIXVOLTAGE2 = 0.04;
float OFFSET2 = 0.00;
float FIXVOLTAGE3 = 0.04;
float OFFSET3 = 0.00;
float FIXVOLTAGE4 = 0.04;
float OFFSET4 = 0.00;
float ofset = 0.00;
float ofset1 = 0;
String fsn1 ;
String fr2 ;
String fsn3 ;
String fr4 ;
String fsn5 ;
String fr6 ;
String fsn7 ;
String fr8 ;
String DATAALL ;
//*****TIMING*****
*****
int Duty1 ;
int FLOW_ON1;
int FLOW_OFF1;
int STEP_TIME1 = 0;
unsigned long TIME_RUN1 ;
unsigned long TIME_CURRENT1 = 0 ;
unsigned long Period1;
double Period_ON1;
double Period_OFF1 ;
bool State1 = false ;
int Duty2 ;
int FLOW_ON2;
int FLOW_OFF2;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

int STEP_TIME2 = 0;
unsigned long TIME_RUN2 ;
unsigned long TIME_CURRENT2 = 0 ;
unsigned long Period2;
double Period_ON2;
double Period_OFF2 ;
bool State2 = false ;
int Duty3 ;
int FLOW_ON3;
int FLOW_OFF3;
int STEP_TIME3 = 0;
unsigned long TIME_RUN3 ;
unsigned long TIME_CURRENT3 = 0 ;
unsigned long Period3;
double Period_ON3;
double Period_OFF3 ;
bool State3 = false ;
int Duty4 ;
int FLOW_ON4;
int FLOW_OFF4;
int STEP_TIME4 = 0;
unsigned long TIME_RUN4 ;
unsigned long TIME_CURRENT4 = 0 ;
unsigned long Period4;
double Period_ON4;
double Period_OFF4 ;
bool State4 = false ;
int STEP_TIME_ALL = 0;
bool State_While = false;
bool State_While1 = true;
bool State_While2 = true;
bool State_While3 = true;
bool State_While4 = true;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

bool state_OFF = true;
}
String FLOW_1 ;
String FLOW_2 ;
String FLOW_3;
String FLOW_4 ;
void DAC_1( int A)
{
    int channelA = A | 0b0111000000000000;
    digitalWrite(CS_1, LOW);
    SPI.transfer(highByte(channelA));
    SPI.transfer(lowByte(channelA));
    digitalWrite(CS_1, HIGH);
    digitalWrite(LDAC_1, LOW);
    delay(1);
    digitalWrite(LDAC_1, HIGH);
}
void DAC_2( int B)
{
    int channelB = B | 0b1111000000000000;
    digitalWrite(CS_1, LOW);
    SPI.transfer(highByte(channelB));
    SPI.transfer(lowByte(channelB));
    digitalWrite(CS_1, HIGH);
    digitalWrite(LDAC_1, LOW);
    delay(1);
    digitalWrite(LDAC_1, HIGH);
}
void DAC_3( int C)
{
    int channelC = C | 0b0111000000000000;
    digitalWrite(CS_2, LOW);
    SPI.transfer(highByte(channelC));
    SPI.transfer(lowByte(channelC));

```

```

digitalWrite(CS_2, HIGH);
digitalWrite(LDAC_2, LOW);
delay(1);
digitalWrite(LDAC_2, HIGH);
}
void DAC_4( int D)
{
    int channelD = D | 0b1111000000000000;
    digitalWrite(CS_2, LOW);
    SPI.transfer(highByte(channelD));
    SPI.transfer(lowByte(channelD));
    digitalWrite(CS_2, HIGH);
    digitalWrite(LDAC_2, LOW);
    delay(1);
    digitalWrite(LDAC_2, HIGH);
}
void AnalogRead()
{ for (int i = 1; i <= 10; i++)
    {
        analogmassflow1 = analogRead(A0);
        x10 = x9;
        x9 = x8;
        x8 = x7;
        x7 = x6;
        x6 = x5;
        x5 = x4;
        x4 = x3;
        x3 = x2;
        x2 = x1;
        x1 = analogmassflow1;
        AVGANALOGMASSFLOW1 = (x10 + x9 + x8 + x7 + x6 + x5 + x4 + x3 + x2 + x1) / 10;
    }
    for (int i = 1; i <= 10; i++)

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

{
    analogmassflow2 = analogRead(A1);
    x10 = x9;
    x9 = x8;
    x8 = x7;
    x7 = x6;
    x6 = x5;
    x5 = x4;
    x4 = x3;
    x3 = x2;
    x2 = x1;
    x1 = analogmassflow2;
    AVGANALOGMASSFLOW2 = (x10 + x9 + x8 + x7 + x6 + x5 + x4 + x3 + x2 + x1) /
10;
}
for (int i = 1; i <= 10; i++)
{
    analogmassflow3 = analogRead(A2);
    x10 = x9;
    x9 = x8;
    x8 = x7;
    x7 = x6;
    x6 = x5;
    x5 = x4;
    x4 = x3;
    x3 = x2;
    x2 = x1;
    x1 = analogmassflow3;
    AVGANALOGMASSFLOW3 = (x10 + x9 + x8 + x7 + x6 + x5 + x4 + x3 + x2 + x1) / 10;
}
for (int i = 1; i <= 10; i++)
{
    analogmassflow4 = analogRead(A3);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

x10 = x9;
x9 = x8;
x8 = x7;
x7 = x6;
x6 = x5;
x5 = x4;
x4 = x3;
x3 = x2;
x2 = x1;
x1 = analogmassflow4;
AVGANALOGMASSFLOW4 = (x10 + x9 + x8 + x7 + x6 + x5 + x4 + x3 + x2 + x1) / 10;
}
// สมการmassflowตัวที่1
bitweighttovoltage1 = (5.0 / 1023) * AVGANALOGMASSFLOW1; //สมการแปลงค่าจาก 0-
1023 ให้เป็น 0-5 V // flowrate1
if (bitweighttovoltage1 < FIXVOLTAGE1)
{
    bitweighttovoltage1 = OFFSET1;
}
else if (bitweighttovoltage1 > FIXVOLTAGE1 )
{
    bitweighttovoltage1 = bitweighttovoltage1 + ofset;
}
volttounit1 = (FS1 / 5) * bitweighttovoltage1; //แปลงจากvเป็นscm
fullscalenew1 = (FS1 / GM1) * GF1; //fullscaleตัวใหม่
flowrate1 = volttounit1 / FS1 * fullscalenew1; //แก้สตัวใหม่
//สมการmassflowตัวที่2
bitweighttovoltage2 = (5.0 / 1023) * AVGANALOGMASSFLOW2; //สมการแปลงค่าจาก 0-
1023 ให้เป็น 0-5 V // flowrate2
if (bitweighttovoltage2 < FIXVOLTAGE2)
{
    bitweighttovoltage2 = OFFSET2;
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

else if (bitweighttovoltage2 > FIXVOLTAGE2 )
{
    bitweighttovoltage2 = bitweighttovoltage2 + ofset;
}

volttounit2 = (FS2 / 5) * bitweighttovoltage2;    //แปลงจากvเป็นsccm
fullscalenew2 = (FS2 / GM2) * GF2;                //fullscaleตัวใหม่
flowrate2 = volttounit2 / FS2 * fullscalenew2;    //แก้สตัวใหม่
//สมการmassflowตัวที่3

bitweighttovoltage3 = (5.0 / 1023) * AVGANALOGMASSFLOW3; //สมการแปลงค่าจาก 0-
1023 ให้เป็น 0-5 V // flowrate3
if (bitweighttovoltage3 < FIXVOLTAGE3)
{
    bitweighttovoltage3 = OFFSET3;
}
else if (bitweighttovoltage3 > FIXVOLTAGE3 )
{
    bitweighttovoltage3 = bitweighttovoltage3 + ofset;
}
volttounit3 = (FS3 / 5) * bitweighttovoltage3;    //แปลงจากvเป็นsccm
fullscalenew3 = (FS3 / GM3) * GF3;                //fullscaleตัวใหม่
flowrate3 = volttounit3 / FS3 * fullscalenew3;    //แก้สตัวใหม่
//สมการmassflowตัวที่4

bitweighttovoltage4 = (5.0 / 1023) * AVGANALOGMASSFLOW4; //สมการแปลงค่าจาก 0-
1023 ให้เป็น 0-5 V // flowrate4
if (bitweighttovoltage4 < FIXVOLTAGE4)
{
    bitweighttovoltage4 = OFFSET4;
}
else if (bitweighttovoltage4 > FIXVOLTAGE4 )
{
    bitweighttovoltage4 = bitweighttovoltage4 + ofset;
}
volttounit4 = (FS4 / 5) * bitweighttovoltage4;    //แปลงจากvเป็นsccm

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

fullscalenew4 = (FS4 / GM4) * GF4;          //fullscaleตัวใหม่
flowrate4 = voltunit4 / FS4 * fullscalenew4; //แก้สตัวใหม่
fsn1 = String(fullscalenew1 );
fr2  = String(flowrate1 );
fsn3 = String(fullscalenew2 );
fr4  = String(flowrate2 );
fsn5 = String(fullscalenew3 );
fr6  = String(flowrate3 );
fsn7 = String(fullscalenew4);
fr8  = String(flowrate4);
}

void setup(void)
{
  Serial.begin(9600);
  SPI.begin();
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  pinMode(A2, INPUT);
  pinMode(A3, INPUT);
  pinMode(CS_1, OUTPUT);
  pinMode(CS_2, OUTPUT);
  pinMode(LDAC_1, OUTPUT);
  pinMode(LDAC_2, OUTPUT);
  DAC_1(0);
  DAC_2(0);
  DAC_3(0);
  DAC_4(0);
}

void loop(void)
{
  while (Serial.available() > 0)
  {
    String CMD_Check = Serial.readStringUntil(',');

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

if (CMD_Check == "NORMAL")
{
    String CMD_Check_Flow = Serial.readStringUntil(',');
    if (CMD_Check_Flow == "FLOW_1")
    {
        String inChar  = Serial.readStringUntil(',');
        String inChar0 = Serial.readStringUntil(',');
        String inChar1 = Serial.readStringUntil(',');
        String inChar2 = Serial.readStringUntil(',');
        FS1 = inChar.toInt(); // fullscale // massflow1
        SP1 = inChar0.toFloat(); // setpoint
        GF1 = inChar1.toFloat(); // gasfactor
        GM1 = inChar2.toFloat(); // gasfactormassflow
        spn1 = SP1 * (GM1 / GF1); // รับค่า setpoint มาแปลงค่าก่อน // massflow ตัวที่ 1
        if (spn1 > FS1)
        {
            spn1 = FS1;
        }
        FS1 = FS1 * 100;
        spn1 = spn1 * 100;
        volt1 = map( spn1, 0, FS1, 0, 4095); // ฟังก์ชันเทียบบรรทัดไดรยง massflow ตัวที่ 1
        if (volt1 == -1)
        {
            volt1 = 0;
        }
        DAC_1(volt1);
        FS1 = FS1 / 100;
        spn1 = spn1 / 100;
        fsn1 = String(fullscalenew1 );
        fr2 = String(flowrate1 );
    }
    else if (CMD_Check_Flow == "FLOW_2")
    {

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

String inChar3 = Serial.readStringUntil(',');
String inChar4 = Serial.readStringUntil(',');
String inChar5 = Serial.readStringUntil(',');
String inChar6 = Serial.readStringUntil(',');
FS2 = inChar3.toInt(); // fullscale // massflow2
SP2 = inChar4.toFloat(); // setpoint
GF2 = inChar5.toFloat(); // gasfactor
GM2 = inChar6.toFloat(); // gasfactormassflow
spn2 = SP2 * (GM2 / GF2); // รับค่า setpoint มาแปลงค่าก่อน // massflow ตัวที่ 2
if (spn2 > FS2)
{
    spn2 = FS2;
}
FS2 = FS2 * 100;
spn2 = spn2 * 100;
volt2 = map(spn2, 0, FS2, 0, 4095); // ฟังก์ชันเทียบบรรทัดไตรยาง massflow ตัวที่ 2
if (volt2 == -1)
{
    volt2 = 0;
}
DAC_2(volt2);
FS2 = FS2 / 100;
spn2 = spn2 / 100;
fsn3 = String(fullscalenew2);
fr4 = String(flowrate2);
}
else if (CMD_Check_Flow == "FLOW_3")
{
    String inChar7 = Serial.readStringUntil(',');
    String inChar8 = Serial.readStringUntil(',');
    String inChar9 = Serial.readStringUntil(',');
    String inChar10 = Serial.readStringUntil(',');
    FS3 = inChar7.toInt(); // fullscale // massflow3

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

SP3 = inChar8.toFloat();//setpoint
GF3 = inChar9.toFloat();//gasfactor
GM3 = inChar10.toFloat();//gasfactormassflow
spn3 = SP3 * (GM3 / GF3);//รับค่าsetpointมาแปลงค่าก่อน//massflowตัวที่3
if (spn3 > FS3)
{
    spn3 = FS3;
}
FS3 = FS3 * 100;
spn3 = spn3 * 100;
volt3 = map(spn3, 0, FS3, 0, 4095); //ฟังก์ชันเทียบบรรทัดไตรยางmassflowตัวที่3
if (volt3 == -1)
{
    volt3 = 0;
}
DAC_3(volt3);
FS3 = FS3 / 100;
spn3 = spn3 / 100;
fsn5 = String(fullscalenew3 );
fr6 = String(flowrate3 );
}
else if (CMD_Check_Flow == "FLOW_4")
{
    String inChar11 = Serial.readStringUntil(',');
    String inChar12 = Serial.readStringUntil(',');
    String inChar13 = Serial.readStringUntil(',');
    String inChar14 = Serial.readStringUntil(',');
    FS4 = inChar11.toInt(); // fullscale // massflow4
    SP4 = inChar12.toFloat();//setpoint
    GF4 = inChar13.toFloat();//gasfactor
    GM4 = inChar14.toFloat();//gasfactormassflow
    spn4 = SP4 * (GM4 / GF4);//รับค่าsetpointมาแปลงค่าก่อน//massflowตัวที่4
    if (spn4 > FS4)

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

{
    spn4 = FS4;
}
FS4 = FS4 * 100;
spn4 = spn4 * 100;
volt4 = map(spn4, 0, FS4, 0, 4095); // ฟังก์ชันเทียบบรรทัดไดรยงmassflowตัวที่4
if (volt4 == -1)
{
    volt4 = 0;
}
DAC_4(volt4);
FS4 = FS4 / 100;
spn4 = spn4 / 100;
fsn7 = String(fullscalenew4 );
fr8 = String(flowrate4 );
}
// ***** TIMING GAS *****
}
else if (CMD_Check == "TIMING")
{
    String CMD_Check_Flow = Serial.readStringUntil(',');
    if (CMD_Check_Flow == "FLOW_1")
    {
        String COMMANDDuty1 = Serial.readStringUntil(','); //เปอร์เซ็นต์ดีวตี้
        String COMMANDPeriod1 = Serial.readStringUntil(','); //หน่วยที่พิมพ์เป็นมิลลิวินาที
        String TIMERUNALL1 = Serial.readStringUntil(','); //หน่วยที่พิมพ์เป็นนาฬิกา
        String VOLUMEVALECLOSE1 = Serial.readStringUntil(',');
        Duty1 = COMMANDDuty1.toInt();
        Period1 = 60000 * COMMANDPeriod1.toDouble();
        TIME_RUN1 = 60000 * TIMERUNALL1.toInt();
        FLOW_ON1 = volt1;
        FLOW_OFF1 = ( VOLUMEVALECLOSE1.toInt() * FLOW_ON1) / 100;
        STEP_TIME1 = TIME_RUN1 / Period1;
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

Period_OFF1 = map(Duty1, 0, 100, 0.00, Period1);
Period_ON1 = Period1 - Period_OFF1;
State1 = true ;
DAC_1( FLOW_ON1);
TIME_CURRENT1 = millis();
}
if (CMD_Check_Flow == "FLOW_2")
{
    String  COMMANDDuty2 = Serial.readStringUntil(',');    //เปอร์เซ็นต์ดีวตี้
    String  COMMANDPeriod2 = Serial.readStringUntil(',');  //หน่วยที่พิมพ์เป็นมิลลิวินาที
    String  TIMERUNALL2 = Serial.readStringUntil(',');     //หน่วยที่พิมพ์เป็นนาฬิกา
    String  VOLUMEVALECLOSE2 = Serial.readStringUntil(',');
    Duty2 = COMMANDDuty2.toInt();
    Period2 = 60000 * COMMANDPeriod2.toDouble();
    TIME_RUN2 = 60000 * TIMERUNALL2.toInt();
    FLOW_ON2 = volt2;
    FLOW_OFF2 = ( VOLUMEVALECLOSE2.toInt() * FLOW_ON2) / 100;
    STEP_TIME2 = TIME_RUN2 / Period2;
    Period_OFF2 = map(Duty2, 0, 100, 0.00, Period2);
    Period_ON2 = Period2 - Period_OFF2;
    State2 = true ;
    DAC_2( FLOW_ON2);
    TIME_CURRENT2 = millis();
}
if (CMD_Check_Flow == "FLOW_3")
{
    String  COMMANDDuty3 = Serial.readStringUntil(',');    //เปอร์เซ็นต์ดีวตี้
    String  COMMANDPeriod3 = Serial.readStringUntil(',');  //หน่วยที่พิมพ์เป็นมิลลิวินาที
    String  TIMERUNALL3 = Serial.readStringUntil(',');     //หน่วยที่พิมพ์เป็นนาฬิกา
    String  VOLUMEVALECLOSE3 = Serial.readStringUntil(',');
    Duty3 = COMMANDDuty3.toInt();
    Period3 = 60000 * COMMANDPeriod3.toDouble();
    TIME_RUN3 = 60000 * TIMERUNALL3.toInt();

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

FLOW_ON3 = volt3;
FLOW_OFF3 = ( VOLUMEVALECLOSE3.toInt() * FLOW_ON3) / 100;
STEP_TIME3 = TIME_RUN3 / Period3;
Period_OFF3 = map(Duty3, 0, 100, 0.00, Period3);
Period_ON3 = Period3 - Period_OFF3;
State3 = true ;
DAC_3(FLOW_ON3);
TIME_CURRENT3 = millis();
}

if (CMD_Check_Flow == "FLOW_4")
{
    String COMMANDDuty4 = Serial.readStringUntil(',');    //เปอร์เซ็นต์ตัว
    String COMMANDPeriod4 = Serial.readStringUntil(','); //หน่วยที่พิมพ์เป็นมิลลิวินาที
    String TIMERUNALL4 = Serial.readStringUntil(',');    //หน่วยที่พิมพ์เป็นนาฬิกา
    String VOLUMEVALECLOSE4 = Serial.readStringUntil(',');
    Duty4 = COMMANDDuty4.toInt();
    Period4 = 60000 * COMMANDPeriod4.toDouble();
    TIME_RUN4 = 60000 * TIMERUNALL4.toInt();
    FLOW_ON4 = volt4;
    FLOW_OFF4 = ( VOLUMEVALECLOSE4.toInt() * FLOW_ON4) / 100;
    STEP_TIME4 = TIME_RUN4 / Period4;
    Period_OFF4 = map(Duty4, 0, 100, 0.00, Period4);
    Period_ON4 = Period4 - Period_OFF4;
    State4 = true ;
    DAC_4( FLOW_ON4);
    TIME_CURRENT4 = millis();
}

/* State_While = !State_While;
State_While1 = true;
State_While2 = true;
State_While3 = true;
State_While4 = true;*/
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

AnalogRead();

DATAALL = fsn1 + "," + fr2 + "," + fsn3 + "," + fr4 + "," + fsn5 + "," + fr6 + "," + fsn7 +
"," + fr8;

Serial.println(DATAALL);
}

if (STEP_TIME1 > 0)
{
    if ((millis() - TIME_CURRENT1 >= Period_ON1) && State1 == true)
    {
        DAC_1(FLOW_OFF1);
        TIME_CURRENT1 = millis();
        State1 = !State1;
        STEP_TIME1 = STEP_TIME1 - 1;
    }
    else if ((millis() - TIME_CURRENT1 >= Period_OFF1) && State1 == false )
    {
        DAC_1(FLOW_ON1);
        TIME_CURRENT1 = millis();
        State1 = !State1;
    }
}

if (STEP_TIME2 > 0)
{
    if ((millis() - TIME_CURRENT2 >= Period_ON2) && State2 == true)
    {
        DAC_2(FLOW_OFF2);
        TIME_CURRENT2 = millis();
        State2 = !State2;
        STEP_TIME2 = STEP_TIME2 - 1;
    }
    else if ((millis() - TIME_CURRENT2 >= Period_OFF2) && State2 == false )
    {
        DAC_2(FLOW_ON2);
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

TIME_CURRENT2 = millis();
State2 = !State2;
}
}
if (STEP_TIME3 > 0)
{
if ((millis() - TIME_CURRENT3 >= Period_ON3) && State3 == true)
{
DAC_3(FLOW_OFF3);
TIME_CURRENT3 = millis();
State3 = !State3;
STEP_TIME3 = STEP_TIME3 - 1;
}
else if ((millis() - TIME_CURRENT3 >= Period_OFF3) && State3 == false )
{
DAC_3(FLOW_ON3);
TIME_CURRENT3 = millis();
State3 = !State3;
}
}
if (STEP_TIME4 > 0)
{
if ((millis() - TIME_CURRENT4 >= Period_ON4) && State4 == true)
{
DAC_4(FLOW_OFF4);
TIME_CURRENT4 = millis();
State4 = !State4;
STEP_TIME4 = STEP_TIME4 - 1;
}
else if ((millis() - TIME_CURRENT4 >= Period_OFF4) && State4 == false )
{
DAC_4(FLOW_ON4);
TIME_CURRENT4 = millis();

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        State4 = !State4;
    }
}

/*Serial.println( (millis() - TIME_CURRENT1 >= Period_OFF1));
Serial.println( (millis() - TIME_CURRENT2 >= Period_OFF2));
Serial.println( (millis() - TIME_CURRENT3 >= Period_OFF3));
Serial.println( (millis() - TIME_CURRENT4 >= Period_OFF4));
//Serial.println( ((STEP_TIME1 == STEP_TIME2 == STEP_TIME3 == STEP_TIME4 ==
0) && (millis() - TIME_CURRENT1 >= Period_OFF1) && (millis() - TIME_CURRENT2 >=
Period_OFF2) && (millis() - TIME_CURRENT3 >= Period_OFF3) && (millis() -
TIME_CURRENT4 >= Period_OFF4)));
if ((STEP_TIME1 == STEP_TIME2 == STEP_TIME3 == STEP_TIME4 == 0) && (millis() -
TIME_CURRENT1 >= Period_OFF1) && (millis() - TIME_CURRENT2 >= Period_OFF2) &&
(millis() - TIME_CURRENT3 >= Period_OFF3) && (millis() - TIME_CURRENT4 >=
Period_OFF4))
{
    State_While1 = false;
    State_While2 = false;
    State_While3 = false;
    State_While4 = false;
}

//Serial.println(!(State_While1 || State_While1 || State_While1 || State_While1 ||
false));
if ( !(State_While1 || State_While1 || State_While1 || State_While1 || false))
{
    Serial.println("KUY");
    State_While = !State_While;
    break;
}*/
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้