ตัวเก็บข้อมูลสำหรับอุปกรณ์ MODBUS

DATA LOGGER FOR MODBUS DEVICES

ณัฐวุฒิ กัลยาณสุต

เทพบดินทร์ นิลละออ

ศุภศิลป์ อุ่นจิตร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2560

ตัวเก็บข้อมูลสำหรับอุปกรณ์ MODBUS
DATA LOGGER FOR MODBUS DEVICES

ณัฐวุฒิ กัลยาณสุต
เทพบดินทร์ นิลละออ
ศุภศิลป์ อุ่นจิตร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมการวัดคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2560

DATA LOGGER FOR MODBUS DEVICES

NATTAWUT KANLAYANASUT

THEPBADIN NINLAOR

SUPASIN AUNJIT

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUE OF TECHNOLOGY LARDKRABANG

ACADEMIC YEAR 2017

ปริญญานิพนธ์ปีการศึกษา 2560

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

.....

หัวข้อปริญญานิพนธ์ ตัวเก็บข้อมูลสำหรับอุปกรณ์ Modbus

DATA LOGGER FOR MODBUS DEVICES

นักศึกษาผู้จัดทำ นายณัฐวุฒิ กัลยาณสุต **รหัสนักศึกษา** 57010464

นายเทพบดินทร์ นิลละออ **รหัสนักศึกษา** 57010526

นายศุภศิลป์ อุ่นจิตร **รหัสนักศึกษา** 57011279

ปริญญา วิศวกรรมศาสตรบัณฑิต

สาขาวิชา วิศวกรรมการวัดคุม

ปีการศึกษา 2560

อาจารย์ผู้ควบคุมปริญญานิพนธ์	ลายมือชื่อ
รองศาสตราจารย์อาจินต์ น่วมสำราญ รองศาสตราจารย์ ดร.วิทยา ทิพย์สุวรรณพร	

หัวข้อปริญญานิพนธ์	ตัวเก็บข้อมูลสำหรับอุปกรณ์ Modbus (Data Logger For Modbus Devices)		
นักศึกษาผู้จัดทำ	นายณัฐวุฒิ	กัลยาณสุต	รหัสนักศึกษา 57010464
	นายเทพบดินทร์	นิลละออ	รหัสนักศึกษา 57010526
	นายศุภศิลป์	อุ่นจิตร	รหัสนักศึกษา 57011279
อาจารย์ที่ปรึกษา	รองศาสตราจารย์ อาจินต์		น่วมสำราญ
	รองศาสตราจารย์ ดร.วิทยา		ทิพย์สุวรรณพร
ปีการศึกษา	2560		

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอ เกี่ยวกับการประยุกต์ใช้อุปกรณ์บันทึกข้อมูลหรือ data logger และการพัฒนาอุปกรณ์โดยใช้บอร์ดคอมพิวเตอร์ขนาดเล็ก raspberry pi ใช้งานร่วมกับโปรแกรม codesys และบันทึกผลลงการ์ดหน่วยความจำถาวร (SD Card) โดยโครงงานมีวัตถุประสงค์เพื่อเป็นการศึกษาการบันทึกข้อมูลทางด้านอุตสาหกรรมเช่น อุณหภูมิ ความดัน การไหล ระดับ เป็นต้น และเป็นการเพิ่มมูลค่าให้แก่อุปกรณ์ที่มีด้วยการพัฒนาให้ใช้งานได้เทียบเท่ากับอุปกรณ์ราคาแพง นอกจากนี้ยังสามารถนำไปบูรณาการกับศาสตร์ด้านอื่นๆได้ เช่น ด้านสิ่งแวดล้อมโดยเก็บข้อมูลมลพิษในอากาศเป็นสถิติเพื่อนำมาวิเคราะห์ และแก้ไขปัญหามลพิษในอากาศ รวมถึงการจัดเก็บข้อมูลลงในฐานข้อมูลและแสดงผลผ่านเว็บไซต์ โดยจะมีการนำเสนอข้อมูลที่ได้จาก Data logger 3 รูปแบบ คือ รูปแบบตาราง รูปแบบกราฟ และใช้โปรแกรมในการนำเสนอ เช่น Excel โดยข้อมูลที่ได้สามารถกำหนดเวลาในการบันทึกผลและนำมาแสดงผลเพื่อทำการวิเคราะห์ผลทางสถิติได้

Thesis Title	Data Logger For Modbus Devices
Authors	Mr. Nattawut Kanlayanasut
	Mr. Thepbadin Ninlaor
	Mr. Supasin Aunjit
Thesis Advisor	Assoc Prof. Arjin Numsomran
	Assoc. Prof. Dr. Vittaya Tipsuwanporn
Year	2560

Abstract

This thesis designs a data logger (also data logger or data recorder). by using raspberry pi 3, codesys program and save the result to memory card (SD card). The purpose is to study the recording of industrial information such as temperature, pressure, flow, etc. it can also be integrated with other fields, such as the environment, by collecting air pollutants into statistics for analysis. And solve the problem of air pollution. The data is stored in the database and display on website. The data from the data logger is present in three formats: table, graph, and use the presentation program, such as Excel to record and display results for statistical analysis

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดี เนื่องจากผู้จัดทำได้รับคำปรึกษาและความอนุเคราะห์จากรศ. อาจินต์ น่วมสำราญ และ รศ.ดร. วิทยา ทิพย์สุวรรณพร ซึ่งเป็นอาจารย์ที่ปรึกษาและผู้ควบคุมปริญญานิพนธ์และคณาจารย์สาขาวิศวกรรมการวัดและควบคุม

ขอขอบพระคุณ รศ. อาจินต์ น่วมสำราญ ที่คอยช่วยเหลือและให้คำปรึกษาในการทำโครงงานนี้ และยังช่วยตรวจสอบและแก้ไขข้อผิดพลาดต่างๆ รวมถึงต้องขอขอบคุณรุ่นพี่ปริญญาโท คุณกิตติพล ก้านขุนทด ที่ได้ให้ความช่วยเหลือในหลายๆด้าน และต้องขอขอบคุณคณาจารย์สาขาวิศวกรรมการวัดและควบคุม คณะวิศวกรรม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกๆท่านที่ได้ให้คำปรึกษาและช่วยเหลือจนสามารถทำปริญญานิพนธ์ฉบับนี้เสร็จสิ้นลงตามวัตถุประสงค์

คณะผู้จัดทำ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญตาราง

ตารางที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความสำคัญของปริญญานิพนธ์

เนื่องจากในปัจจุบันการเก็บบันทึกข้อมูลนั้นมีความสำคัญอย่างมากในรูปแบบงานด้านอุตสาหกรรม โดยเฉพาะการเก็บข้อมูลที่มีการเปลี่ยนแปลงตลอดเวลา เช่น ความชื้น อุณหภูมิ แสงแดดในงานเกษตร เป็นต้น จำเป็นต้องมีอุปกรณ์บันทึกผลและสามารถนำข้อมูลมาวิเคราะห์ได้ ซึ่งเครื่องบันทึกข้อมูล หรือ Data logger จะเป็นตัวนำข้อมูลที่เก็บไว้แล้วมาถ่ายโอนเข้าคอมพิวเตอร์เพื่อสะดวกต่อการขนย้ายและข้อมูลไม่สูญหาย

ดังนั้นผู้วิจัยจึงได้มีแนวคิดในการศึกษาและพัฒนาตัวเก็บข้อมูลสำหรับอุปกรณ์ Modbus เพื่อใช้ในการเก็บข้อมูลตัวแปรพื้นฐานทางด้านอุตสาหกรรม เช่น อุณหภูมิ ความดัน การไหล ระดับ เป็นต้น และสามารถนำไปบูรณาการกับศาสตร์ด้านอื่นๆได้ เช่น ด้านสิ่งแวดล้อมโดยเก็บข้อมูลมลพิษในอากาศ เป็นต้น และเพื่อให้งบประมาณไม่สูงจนเกินไป ผู้วิจัยจึงมีแนวคิดที่จะออกแบบ Data logger สำหรับอุปกรณ์ Modbus ด้วยบอร์ด Raspberry pi 3 โดยเขียนโปรแกรมควบคุมด้วยโปรแกรม Codesys ซึ่งเป็นโปรแกรมที่สามารถดาวน์โหลดมาใช้ได้ฟรี นอกจากนี้สามารถบันทึกผลของข้อมูลลงใน SD Card และบนฐานข้อมูล โดยสามารถดูข้อมูลย้อนหลังและนำผลของข้อมูลมาวิเคราะห์เพื่อแก้ไขหรือพัฒนาในกระบวนการต่อไป

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1.ศึกษาพื้นฐานการวัดและบันทึกข้อมูลกายภาพโดยตัวเก็บข้อมูลสำหรับอุปกรณ์ Modbus

2. ออกแบบและพัฒนาตัวเก็บข้อมูลสำหรับอุปกรณ์ Modbus

3. ทดสอบและวิเคราะห์ประสิทธิภาพตัวเก็บข้อมูลสำหรับอุปกรณ์ Modbus

1.3 ขอบเขตของปริญญานิพนธ์

1. ศึกษาการทำงานของอุปกรณ์เก็บข้อมูลหรือ Data logger
2. ศึกษาการใช้งานโปรแกรม codesys และบอร์ด raspberry รวมถึงการจัดการฐานข้อมูลเพื่อจะนำค่าข้อมูลมาแสดงผล

1.4 ขั้นตอนการศึกษา

1. ศึกษาการทำงานของอุปกรณ์เก็บข้อมูลรูปแบบต่างๆข้อดีข้อเสียของอุปกรณ์เพื่อเป็นพื้นฐานในการกำหนดสเปคของอุปกรณ์ Data Logger ที่ต้องการพัฒนา
2. ศึกษาการสื่อสารแบบ Modbus Protocol และโปรแกรมที่ใช้ทำให้อุปกรณ์สื่อสารกันในลักษณะ Master/Slave
3. ศึกษาการเขียนโปรแกรม Codesys, Arduino การเขียนโดยใช้ภาษา ladder และ structure เบื้องต้น
4. ศึกษาออกแบบเมนูแสดงผลและระยะเวลาในการเก็บข้อมูลรวมถึงกำหนดข้อมูลพื้นฐานของอุปกรณ์
5. ศึกษาการทำงานของบอร์ด Arduino Uno r3 และ Ethernet Shield W5500

1.5 ประโยชน์ที่คาดว่าจะได้รับ

มีความรู้พื้นฐานการบันทึกข้อมูลกายภาพโดยตัวเก็บข้อมูลสำหรับอุปกรณ์ Modbus สามารถออกแบบและพัฒนาตัวเก็บข้อมูลสำหรับอุปกรณ์ Modbus รวมถึงสามรถทดสอบและวิเคราะห์ประสิทธิภาพตัวเก็บข้อมูลสำหรับอุปกรณ์ Modbus

1.6 ผลที่คาดหวังของโครงการ

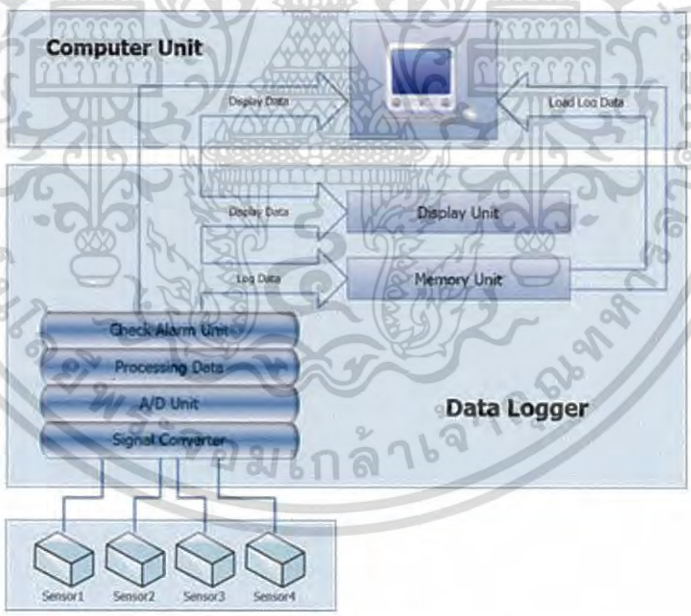
ได้ความรู้ที่เกี่ยวข้องกับเนื้อหาของโครงงาน ได้ศึกษาและพัฒนาตัวเก็บข้อมูลสำหรับอุปกรณ์ Modbus มีความรู้และความเข้าใจในการใช้งานโปรแกรม codesys และบอร์ด raspberry pi รวมถึงการได้อุปกรณ์เก็บข้อมูลหรือ Data logger ที่มีประสิทธิภาพดีแต่มีราคาถูก และการนำเอาเทคโนโลยีสารสนเทศสมัยใหม่มาใช้อย่างมีคุณค่าและสร้างสรรค์ และได้ศึกษาวิธีการเก็บบันทึกข้อมูลกายภาพต่างๆ เช่น อุณหภูมิ ความชื้น ปริมาณก๊าซคาร์บอนไดออกไซด์ โดยใช้บอร์ดคอมพิวเตอร์ขนาดเล็กและการ์ดหน่วยความจำถาวร รวมถึงการจัดเก็บข้อมูลลงในฐานข้อมูลเพื่อแสดงผลผ่านเว็บไซต์ นอกจากนี้ยังสามารถจัดทำสื่อนำเสนอได้ด้วยตนเองและประยุกต์ใช้ให้เข้ากับการเรียนรู้ของตนเองมากยิ่งขึ้น

บทที่ 2
ทฤษฎีที่เกี่ยวข้องกับงานวิจัย

2.1 อุปกรณ์บันทึกข้อมูล (Data Logger)

Data Logger คือ อุปกรณ์ที่ใช้สำหรับเก็บบันทึกข้อมูลที่เป็นสัญญาณชนิดต่างๆโดย Data logger จะมีหน่วยความจำสำหรับเก็บค่าที่วัดได้ของสัญญาณตามการใช้งานที่เกี่ยวข้องกับการบันทึกข้อมูลของอุณหภูมิความชื้น, แรงดันไฟฟ้า, กระแสไฟฟ้า, ความดัน, การไหล และอื่นๆ ช่วงเวลาการบันทึกที่กำหนดไว้โดยอัตโนมัติสามารถใช้เครื่องคอมพิวเตอร์ในการอ่านข้อมูลจากหน่วยความจำของ Data logger สามารถนำรายงาน มาวิเคราะห์หาปัญหาที่เกิดขึ้นได้รวดเร็วและแน่นอนไม่มีปัญหาที่เกิดจากคนบันทึกผิดพลาดและการหยุดงาน สามารถตอบสนองความต้องการเกี่ยวกับการใช้งานในการเก็บข้อมูลเพื่อทำการบันทึกแสดงรายละเอียดการใช้งานดังรูปที่ 2.1

2.1.1 ฟังก์ชั่นของ Data Logger



รูปที่ 2.1 ภาพรวมฟังก์ชั่นของ Data Logger

Data Logge จะรับค่าที่จะบันทึกจากตัวเซนเซอร์นำมาผ่านทางสัญญาณ Convertor เพื่อทำการแปลงสัญญาณที่รับมาให้เป็นสัญญาณที่ A/D ของ Data Logger นำมาใช้ในการแปลงให้เป็นข้อมูลดิจิตอลหลังจากนั้น Data Logger อาจนำข้อมูลดิจิตอลนั้นมาประมวลผลหรือนำข้อมูลมาเช็คเพื่อทำการส่ง Alarm ไปเตือนผู้ใช้ว่าข้อมูลมีค่ามากไปหรือ

น้อยไปได้แล้วจึงนำข้อมูลที่ได้ไปเก็บบันทึกในหน่วยความจำของ Data Logger หรือนำมาแสดงผลบนหน้าของ Data Logger หรือบนหน้าจอคอมพิวเตอร์ได้

2.1.2 เครื่องคอมพิวเตอร์กับการใช้งาน **Data logger**

1) เพื่อใช้กำหนดการสื่อสารกับ Data logger
2) เพื่อใช้กำหนดรูปแบบการทำงาน (Configuration) ของ Data loggerเพื่อใช้ในการแสดงค่าของข้อมูลแบบ Real Time หรืออ่านข้อมูลที่เก็บบันทึกไว้ใน Data logger มานำเสนอในภายหลัง
3) เพื่อใช้ในการวิเคราะห์และประมวลผลข้อมูลสร้างรายงาน,รูปกราฟ, สำหรับนำมาแสดงผลบนหน้าจอหรือพิมพ์ออกมาได้

2.1.3 ขนาดของหน่วยความจำที่ใช้ในการเก็บข้อมูล

1) ช่วงเวลาทั้งหมดที่เราต้องการให้Datalogger เก็บบันทึกข้อมูล (Recording Duration)
2) ค่า Sampling Time ที่เราต้องการใช้ในการบันทึกข้อมูล โดย Recording Duration / Sampling Time = จำนวนข้อมูลที่สามารถบันทึกได้ของหน่วยความจำ

2.1.4 ส่วนที่ใช้ในการติดต่อกับผู้ใช้งาน (User Interface)

เลือกตามรูปแบบการใช้งานที่ต้องการ ซึ่งมีหลายรูปแบบดังนี้

1) มีหน้าจอแสดงผลและปุ่มกดสำหรับใช้งานที่ตัว Data Logger (Front Panel & Display)
2) จอสัมผัส (Touch Screen)
3) อุปกรณ์ควบคุมระยะไกลแบบมือถือ (Hand-held/Remote Programmer) ใช้เครื่องคอมพิวเตอร์ในการติดต่อและทำงานกับ Data Logger

2.1.5 การติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์กับ Data Logger

การสื่อสารมีหลายรูปแบบหรือมาตรฐานที่ใช้ในการติดต่อดังนี้RS232, RS422, RS485, USB, IEEE1394, GPIB, SCSI, TTL, Parallel, อีเทอร์เน็ต, Modem, Radio/Telemetry ซึ่งแต่ละแบบก็เหมาะสมกับการใช้งานคนละรูปแบบกัน เช่น ถ้าเราสามารถต่อสาย จากเครื่องคอมพิวเตอร์เพื่อติดต่อกับ Data Logger ในระยะใกล้ๆได้และทำการติดต่อกับ Data Logger เพียงตัวเดียวก็อาจใช้ Data Logger ที่ใช้มาตรฐาน RS232 หรือ

ถ้าต่อสายระยะใกล้ๆได้ แต่ต้องการติดต่อกับ Data Logger หลายๆตัวพร้อมกันได้ ก็อาจใช้ Data Logger ที่ใช้มาตรฐาน RS485 หรือถ้าเครื่องคอมพิวเตอร์ที่เราจะใช้งานกับ Data Logger อยู่ไกลจาก Data Logger มากๆก็อาจจะใช้ Data Logger ที่สามารถใช้ Modem ในการติดต่อสื่อสารแทน เป็นต้น

2.1.6 Data Presentation

การนำเสนอข้อมูลที่ได้จาก Data Logger นั้น สามารถนำเสนอในรูปแบบต่างๆ เช่น รูปแบบตาราง (Table) หรือ รูปกราฟ (Graph) โดยใช้โปรแกรมในการนำเสนอ เช่น Microsoft Excel เป็นต้น

2.2 แบบจำลองเครือข่าย (Network Model)

ในระบบเครือข่ายจะประกอบไปด้วยทั้งฮาร์ดแวร์และซอฟท์แวร์เพื่อที่จะทำหน้าที่ในการส่งข้อมูลจากต้นทางไปยังปลายทางได้อย่างถูกต้อง และมีประสิทธิภาพในส่วนของฮาร์ดแวร์นั้นจะทำหน้าที่ในการประมวลผลและนำสัญญาณทางไฟฟ้าต่างๆจากต้นทางไปยังปลายทางส่วนซอฟท์แวร์นั้นเป็นส่วนสำคัญอีกส่วนหนึ่งที่จะคอยควบคุมการทำงานของฮาร์ดแวร์อีกทั้งยังเป็นส่วนที่จะติดต่อกับผู้ใช้ด้วย

เนื่องจากระบบเครือข่ายนั้นมีการทำงานที่ซับซ้อนดังนั้นในการอธิบายถึงการทำงานของระบบเครือข่ายนั้นจึงต้องแบ่งการทำงานออกเป็นระดับชั้นต่างๆเพื่อที่ให้ง่ายต่อการทำความเข้าใจในบทนี้จะอธิบายถึงระดับชั้นของเครือข่าย(NetworkLayer)และการทำงานในแต่ละระดับชั้น

2.2.1 การทำงานแบบเป็นลำดับชั้น (Layers Tasks)

ก่อนจะอธิบายถึงแบบจำลองของเครือข่าย จะขอยกตัวอย่างลำดับขั้นตอนของการส่งจดหมายทางไปรษณีย์ เพื่อให้เข้าใจถึงการทำงานแบบลำดับชั้นเสียก่อน

ในการส่งจดหมายนั้นสามารถแบ่งลำดับชั้นการทำงานได้ 3 ลำดับ การที่จะส่งจดหมายได้นั้นจะต้องมีทั้งผู้ส่ง ผู้รับ และผู้นำส่งจดหมาย โดยที่แต่ละคนจะมีขั้นตอนดังนี้

2.2.1.1 ผู้ส่ง

มีขั้นตอนการทำงานเรียงลำดับจากชั้นที่ 3 ไปชั้นที่ 2 และชั้นที่ 1 ดังนี้

ชั้นที่ 3 ผู้ส่งจดหมายจะต้องเขียนจดหมาย นำจดหมายใส่ซอง เขียนที่อยู่ของผู้ส่งและผู้รับ นำจดหมายหย่อนลงในตู้รับจดหมาย

ชั้นที่ 2 จดหมายจะถูกนำออกจากตู้ไปยังที่ทำการไปรษณีย์ต้นทาง

ชั้นที่ 1 ที่ทำการไปรษณีย์ต้นทางจะทำการคัดแยกจดหมาย และให้ผู้นำส่งจดหมายนำจดหมายไปยังไปรษณีย์ปลายทาง

2.2.1.2 ผู้รับ

เมื่อจดหมายไปถึงยังไปรษณีย์ปลายทางแล้ว ผู้รับจะมีลำดับขั้นตอนจากชั้นที่ 1 ไป ชั้นที่ 2 และชั้นที่ 3 ดังนี้

ชั้นที่ 1 ไปรษณีย์ปลายทางทำการรับจดหมาย

ชั้นที่ 2 จดหมายจะถูกนำมาจัดเรียง และส่งไปยังผู้รับ

ชั้นที่ 3 เมื่อผู้รับได้รับจดหมายแล้ว เปิดจดหมาย อ่านจดหมาย

เมื่อพิจารณาแล้วจะเห็นได้ว่าในแต่ละชั้นของการทำงานนั้นจะมีหน้าที่ที่แตกต่างกันออกไป ที่ต้นทางจะต้องทำงานจากระดับชั้นบนสุดไปยังระดับชั้นล่างสุด ส่วนที่ปลายทางจะทำงานจากระดับชั้นล่างสุดไปยังระดับชั้นบนสุด โดยที่ในแต่ละระดับชั้นจะต้องทำให้เสร็จก่อน จึงจะสามารถทำงานในระดับชั้นถัดไปได้

2.2.2 แบบจำลองอินเทอร์เน็ต (Internet Model)

จากหัวข้อข้างต้นจะทำให้เห็นภาพของการส่งข้อมูลจากที่หนึ่งไปยังอีกที่หนึ่ง ซึ่งจะต้องผ่านกระบวนการต่างๆมากมาย เพื่อให้ง่ายต่อการอธิบายและสามารถเข้าถึงการทำงานได้ดีขึ้น จึงจำเป็นที่จะต้องแบ่งการทำงานเหล่านั้นออกเป็นลำดับชั้น ในระแบบเครือข่ายก็เหมือนกันจะต้องมีการแบ่งการทำงานออกเป็นลำดับชั้น เนื่องจากการทำงานของเครือข่ายมีความซับซ้อนกว่าการส่งจดหมายทางไปรษณีย์เป็นอย่างมาก ดังนั้นจึงต้องมีการออกแบบมาตรฐานของแบบจำลองเครือข่าย(Network model) เพื่อใช้ในการลดความซับซ้อนและให้ง่ายในการทำความเข้าใจ

เนื่องจากอินเทอร์เน็ตเป็นเครือข่ายที่ใหญ่ที่สุดและมีผู้ใช้งานเครือข่ายนี้กันอยู่ทั่วโลก ดังนั้นในการอธิบายแบบจำลองของเครือข่ายจะใช้แบบจำลองของอินเทอร์เน็ตเป็นหลัก แบบจำลองของอินเทอร์เน็ตหรือสามารถเรียกได้อีกชื่อหนึ่งที่รู้จักโดยทั่วไปคือ ชุดโพรโตคอลทีซีพี/ไอพี (TCP/IP Protocol suite) ซึ่งจะประกอบไปด้วยลำดับชั้นหรือ เลเยอร์ (Layers) ที่ซ้อนทับกันอยู่ทั้งหมด 5 เลเยอร์ คือ ฟิสิคัล (Physical layer), เดทาลิงค์ (Data link layer), เน็ตเวิร์ก (Network layer), ทรานสปอร์ต (Transport layer) และแอพพลิเคชั่น (Application layer)

5 Application

4 Transport

3 Network

2 Data link

1 Physical

อุปกรณ์ที่ใช้ภายในเครือข่ายอินเทอร์เน็ตแต่ละตัวนั้นไม่จำเป็นที่จะต้องทำงานให้ครบ 5 เลเยอร์ ขึ้นอยู่กับลักษณะของการใช้งาน เช่น ในการส่งข้อมูลจากสถานีต้นทาง A ไปยังสถานีปลายทาง B ซึ่งอยู่กันคนละซีกโลกนั้น จำเป็นที่ข้อมูลนั้นจะต้องผ่านอุปกรณ์ต่างๆ มากมายกว่าที่จะถึงจุดหมายปลายทางแต่อุปกรณ์บางตัวที่ข้อมูลจะต้องวิ่งผ่านไปนั้นอาจจะทำงานแค่ 2 หรือ 3 เลเยอร์เท่านั้น

2.2.2.1 หน้าที่ของแต่ละเลเยอร์

1) ฟิสิคัลเลเยอร์ (Physical Layer) เป็นเลเยอร์ระดับล่างสุดที่ทำการส่งข้อมูลในระดับบิตไปยังสื่อที่ใช้ในการส่งข้อมูลภายในเลเยอร์นี้จะเกี่ยวข้องกับการกำหนดคุณสมบัติทางกลและทางไฟฟ้าให้กับอินเทอร์เฟซและสื่อที่ใช้ในการส่งข้อมูล
2) เดทาลิงค์เลเยอร์ (Data link Layer) เดทาลิงค์เลเยอร์จะรับข้อมูลมาจากฟิสิคัลเลเยอร์ข้อมูลที่ได้รับมานั้นอาจมีความผิดพลาดซึ่งเกิดจากการเดินทางของข้อมูลมาจากต้นทางดังนั้นในเดทาลิงค์เลเยอร์จะต้องทำการแก้ไขข้อผิดพลาดของข้อมูลเพื่อที่จะให้เน็ตเวิร์กเลเยอร์ได้รับข้อมูลที่ปราศจากข้อผิดพลาด
3) เน็ตเวิร์กเลเยอร์ (Network Layer) ในเลเยอร์นี้จะรับผิดชอบในการส่งข้อมูลจากต้นทางไปยังปลายทาง (source-to-destination delivery) ให้เป็นไปได้อย่างถูกต้องถึงแม้ว่าการส่งข้อมูลจะเป็นการส่งข้ามเครือข่ายกันเนื่องจากในเดทาลิงค์เลเยอร์จะเน้นการส่งข้อมูลภายในเครือข่ายเดียวกัน
4) ทรานสปอร์ตเลเยอร์ (Transport Layer) ในเน็ตเวิร์กเลเยอร์จะทำการส่งข้อมูลจากต้นทาง (source) ไปปลายทาง (destination) ให้ได้อย่างถูกต้องถ้าทั้งต้นทางและปลายทางมีโพรเซส (process) ในการรับส่งข้อมูลเพียงโพรเซสเดียวแต่ทั้งต้นทางและปลายทางนั้นสามารถมีการรับส่งข้อมูลได้หลายโพรเซส ดังนั้น ในทรานสปอร์ตเยอร์จึงต้องมีข้อกำหนดในการรับส่งข้อมูลกันระหว่างโพเซส (process-to-process delivery) ด้วย
5) แอพพลิเคชั่นเลเยอร์ (Application Layer) ในเลเยอร์นี้จะเน้นในส่วนของการติดต่อกับผู้ใช้ (User interface)และบริการ service ต่างๆของ เครือข่ายที่จะมีให้ใช้ เช่น จดหมาย อิเล็กทรอนิกส์

(Electronic mail) การโอนย้ายไฟล์ (File transfer)

2.2.3 แบบจำลอง OSI (OSI Model)

แบบจำลอง OSI (Open System Interconnection) ได้ถูกออกมาโดยองค์กร ISO (International Organization for Standardzation) เพื่อเป็นมาตรฐานในการพัฒนาเครือข่ายการสื่อสารข้อมูลและคอมพิวเตอร์ โดยจะแบ่งโครงสร้างของเครือข่ายออกเป็นเลเยอร์ และกำหนดการทำงานของแต่ละเลเยอร์ แต่แบบจำลอง OSI นั้นเป็นแบบจำลองที่เป็นเพียงทฤษฎีเพื่อจะทำให้ง่ายต่อการเข้าใจและเห็นถึงการทำงานเป็นเลเยอร์

7 Application

6 Presentation

5 Session

4 Transport

3 Network

2 Data link

1 Physical

จะเห็นว่าแบบจำลอง OSI จะแบ่งออกเป็น 7 เลเยอร์ โดยที่จะมี เซสชั่นเลเยอร์ (Session layer) และพรีเซนเทชั่นเลเยอร์ (Presentation layer) เพิ่มเข้ามา

1) เซสชั่นเลเยอร์ (Session layer) จะเป็นเลเยอร์ที่มีการสร้างเซสชั่นกันระหว่างเครื่อง เพื่อให้ผู้ใช้สามารถที่จะเชื่อมโยงกับเครื่องอื่นๆได้ เช่น การล็อกอินเข้าใช้งานเครื่องระยะไกลในแต่ละครั้งเป็นต้นเมื่อมีการสร้างเซสชั่นกันแล้วการรับส่งข้อมูลจะ ใช้บริการจากทรานสปอร์ตเลเยอร์
2) เซนเทชั่นเลเยอร์ (Presentation layer) เป็นเลเยอร์ที่ช่วยแปลงรูปแบบของข้อมูล และแปลข้อมูลเพื่อที่จะให้การแลกเปลี่ยนข้อมูลนั้นๆ เป็นไปในรูปแบบเดียวกัน เช่น การเข้ารหัสข้อมูล การถอดรหัสข้อมูล และการบีบอัดข้อมูล เป็นต้น

อย่างไรก็ตามในปัจจุบันนี้การทำงานของทั้งเซสชั่นเลเยอร์ และพรีเซนเทชั่นเลเยอร์ได้นำมาสร้างเป็นโพรโตคอลที่อยู่ในเลเยอร์ต่างๆ ของแบบจำลองอินเทอร์เน็ตแล้ว เช่น การเข้ารหัสและการถอดรหัสข้อมูล หรือการบีบข้อมูล เป็นต้น ดันนั้นจึงได้เน้นถึงแบบจำลองอินเทอร์เน็ตเป็นหลัก

2.3 เครือข่ายคอมพิวเตอร์และระบบ Cloud

2.3.1 ระบบเครือข่ายคอมพิวเตอร์

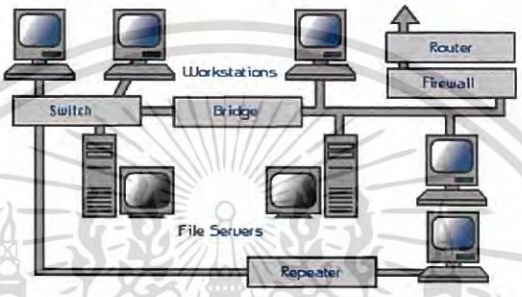
ระบบเครือข่ายคอมพิวเตอร์ หรือระบบเน็ตเวิร์ค คือ กลุ่มของคอมพิวเตอร์และอุปกรณ์ต่างๆ ที่ถูกนำมาเชื่อมต่อกันเพื่อให้ผู้ใช้ในเครือข่ายสามารถติดต่อสื่อสาร แลกเปลี่ยนข้อมูล และใช้อุปกรณ์ต่างๆ ในเครือข่ายร่วมกันได้"เครือข่ายนั้นมีหลายขนาด ตั้งแต่ขนาดเล็กที่เชื่อมต่อกันด้วยคอมพิวเตอร์เพียงสองสามเครื่อง เพื่อใช้งานในบ้านหรือในบริษัทเล็กๆ ไปจนถึงเครือข่ายขนาดใหญ่ที่เชื่อมต่อกันทั่วโลก ส่วน Home Network หรือเครือข่ายภายในบ้าน ซึ่งเป็นระบบ LAN (Local Area Network) เป็นระบบเครือข่ายคอมพิวเตอร์ขนาดเล็กๆ หมายถึง การนำเครื่องคอมพิวเตอร์และอุปกรณ์ มาเชื่อมต่อกันในบ้าน สิ่งที่เกิดตามมาก็คือประโยชน์ในการใช้คอมพิวเตอร์ด้านต่างๆ เช่น

1) การใช้ทรัพยากรร่วมกัน หมายถึง การใช้อุปกรณ์ต่างๆ เช่น เครื่องพิมพ์ร่วมกัน กล่าวคือ มีเครื่องพิมพ์เพียงเครื่องเดียว ทุกคนในเครือข่ายสามารถใช้เครื่องพิมพ์นี้ได้ ทำให้สะดวกและประหยัดค่าใช้จ่าย เพราะไม่ต้องลงทุนซื้อเครื่องพิมพ์หลายเครื่อง (นอกจากจะเป็นเครื่องพิมพ์คนละประเภท)
2) การแชร์ไฟล์ เมื่อคอมพิวเตอร์ถูกติดตั้งเป็นระบบเน็ตเวิร์กแล้ว การใช้ไฟล์ข้อมูลร่วมกันหรือการแลกเปลี่ยนไฟล์ทำได้อย่างสะดวกรวดเร็วไม่ต้องอุปกรณ์เก็บข้อมูลใดๆทั้งสิ้นในการโอนย้ายข้อมูลตัดปัญหาเรื่องความจุของสื่อบันทึกยกเว้นอุปกรณ์ในการจัดเก็บข้อมูลหลักอย่างฮาร์ดดิสก์ หากพื้นที่เต็มก็คงต้องหามาเพิ่ม
3) การติดต่อสื่อสาร โดยคอมพิวเตอร์ที่เชื่อมต่อเป็นระบบเน็ตเวิร์ก สามารถติดต่อพูดคุยกับเครื่องคอมพิวเตอร์อื่น โดยอาศัยโปรแกรมสื่อสารที่มีความสามารถใช้เป็นเครื่องคอมพิวเตอร์ได้เช่นเดียวกันหรือการใช้อีเมล์ภายในก่อให้เครือข่าHome Network หรือ Home Office จะเกิดประโยชน์นี้อีกมากมาย
4) การใช้อินเทอร์เน็ตร่วมกัน คอมพิวเตอร์ทุกเครื่องที่เชื่อมต่อในระบบเน็ตเวิร์กสามารถใช้งานอินเทอร์เน็ตได้ทุกเครื่อง โดยมีโมเด็มตัวเดียว ไม่ว่าจะเป็นแบบอนาล็อกหรือแบบดิจิตอลอย่าง ADSL ยอดฮิตในปัจจุบันระบบเครือข่ายสามารถแบ่งออกเป็น 3 ประเภท ดังนี้

2.3.1.1 เครือข่ายเฉพาะที่ หรือ (Local Area Network : LAN)

เป็นเครือข่ายขนาดเล็กซึ่งเชื่อมโยงคอมพิวเตอร์ และอุปกรณ์สื่อสารที่อยู่ในห้อง ที่บริเวณเดียวกันเข้าด้วยกัน เช่น ภายในอาคาร หรือภายในองค์กรที่มีระยะทางไม่ไกลมากนัก เป็นต้น โดยคอมพิวเตอร์แต่ละเครื่องจะต่อเข้ากับอุปกรณ์เครือข่าย เช่น ฮับ, สวิตช์ เป็นต้น ซึ่ง

อุปกรณ์เครือข่ายแต่ละตัวจะเชื่อมต่อกันโดยใช้สายตีเกลียวคู่ สายใยแก้วนำแสงหรือคลื่นวิทยุ และเครือข่ายแลนจะเชื่อมต่อถึงกันด้วยอุปกรณ์จัดเส้นทาง (router) การสร้างเครือข่ายแลนนี้แต่ละองค์กร สามารถดำเนินการเองได้ โดยการวางสายสัญญาณสื่อสารภายในอาคารหรือภายในพื้นที่ของตนเอง เครือข่ายแลนมีตั้งแต่เครือข่ายขนาดเล็กที่เชื่อมโยงคอมพิวเตอร์ตั้งแต่สอง เครื่องขึ้นไปภายในห้องเดียวกัน จนถึงเชื่อมโยงระหว่างห้องหรือองค์กรขนาดใหญ่ เช่น ภายในสำนักงาน ภายในโรงเรียนหรือมหาวิทยาลัย เป็นต้น ทำให้เครื่องคอมพิวเตอร์หลายๆ เครื่องที่เชื่อมต่อกัน สามารถส่งข้อมูลแลกเปลี่ยนกันได้อย่างสะดวก รวดเร็ว และยังสามารถใช้ทรัพยากรร่วมกันได้อีกด้วย **ดังรูปที่** 2.2



รูปที่ 2.2 ระบบเครือข่าย LAN

2.3.1.2 เครือข่ายนครหลวง หรือแมน

(Metropolitan Area Network : MAN) เป็นเครือข่ายที่เชื่อมโยงแลนที่อยู่ห่างกัน เช่น ระหว่างสำนักงานที่อยู่คนละอาคาร ระบบเคเบิลทีวีตามบ้านในปัจจุบัน เป็นต้น โดยมีลักษณะการเชื่อมโยงคอมพิวเตอร์ที่มีระห่างไกลกันในช่วง 5-40 Km ผ่านสายสื่อสารประเภทสายใยแก้วนำแสงสายโคแอกเชียล หรืออาจใช้คลื่นไมโครเวฟ

2.3.1.3 เครือข่ายวงกว้าง หรือแวน

(Wide Area Network : WAN) เป็นเครือข่ายคอมพิวเตอร์ขนาดใหญ่ที่เชื่อมโยงระบบคอมพิวเตอร์ในระยะห่างไกล มีการติดต่อสื่อสารกันในบริเวณกว้าง เช่น เชื่อมโยงระหว่างจังหวัด ระหว่างประเทศ เป็นต้น

2.3.2 อุปกรณ์สื่อสารในระบบเครือข่ายคอมพิวเตอร์

การเชื่อมต่อเครื่องคอมพิวเตอร์ให้กลายเป็นระบบเครือข่ายได้นั้น จะต้องอาศัยอุปกรณ์สื่อสารในระบบเครื่องคอมพิวเตอร์ (network device) ซึ่ง ทำหน้าที่รับและส่งข้อมูลโดยผ่านทางสื่อกลาง ไม่ว่าจะเป็นสื่อกลางแบบใช้สาย และสื่อกลางแบบไร้สาย ดังรูปที่ 2.3 ซึ่งอุปกรณ์สื่อสารในระบบเครือข่ายคอมพิวเตอร์มีดังนี้

รูปที่ 2.3 อุปกรณ์สื่อสารในระบบเครือข่ายคอมพิวเตอร์

2.3.2.1 เครื่องทวนสัญญาณ (repeater)

เป็นอุปกรณ์ที่ทำหน้าที่รับสัญญาณดิจิทัล แล้วส่งต่อออกไปยังอุปกรณ์ตัวอื่นเหตุที่ต้องใช้อุปกรณ์ทวนสัญญาณเนื่องจากการส่งสัญญาณไปในตัวกลางที่เป็นสายสัญญาณนั้น เมื่อระยะทางมากขึ้นแรงดันของสัญญาณจะลดลงเรื่อยๆ ทำให้ไม่สามารถส่งสัญญาณในระยะทางไกลๆได้ ดังนั้น การใช้อุปกรณ์ทวนสัญญาณจะทำให้สามารถส่งสัญญาณไปได้ไกลขึ้น โดยสัญญาณไม่สูญหาย

2.3.2.2 ฮับ (hub)

เป็นอุปกรณ์ที่ทำหน้าที่รวมสัญญาณที่มาจากอุปกรณ์รับส่ง หรือเครื่องคอมพิวเตอร์หลายๆ เครื่องเข้าด้วยกัน สัญญาณที่ส่งมาจากฮับจะกระจายไปยังทุกเครื่องที่ต่ออยู่กับฮับซึ่งแต่ละเครื่องจะเลือกรับเฉพาะข้อมูลที่ส่งมาถึงตนเองเท่านั้น

2.3.2.3 บริดจ์ (bridge)

ใช้ในการเชื่อมต่อเครือข่ายหลายเครือข่ายเข้าด้วยกัน โดยจะต้องเป็นเครือข่ายที่ใช้โพรโทคอลตัวเดียวกัน ซึ่งมีความสามารถมากกว่าฮับและอุปกรณ์ทวนสัญญาณ คือ สามารถกรองข้อมูลที่จะส่งต่อได้ โดยการตรวจสอบว่า ข้อมูลที่ส่งนั้นมีปลายทางอยู่ที่ใด หากเครื่องปลายทางอยู่ภายในเครือข่ายเดียวกันกับเครื่องส่ง ก็จะส่งข้อมูลนั้นไปในเครือข่ายเดียวกันเท่านั้น ไม่ส่งไปยังเครือข่ายอื่น แต่หากข้อมูลมีปลายทางอยู่ที่เครือข่ายอื่น ก็จะส่งข้อมูลไปในเครือข่ายที่มีเครื่องปลายทางอยู่เท่านั้น ทำให้สามารถจัดการกับความหนาแน่นของข้อมูลได้อย่างมีประสิทธิภาพมากขึ้น

2.3.2.4 อุปกรณ์จัดเส้นทาง (router)

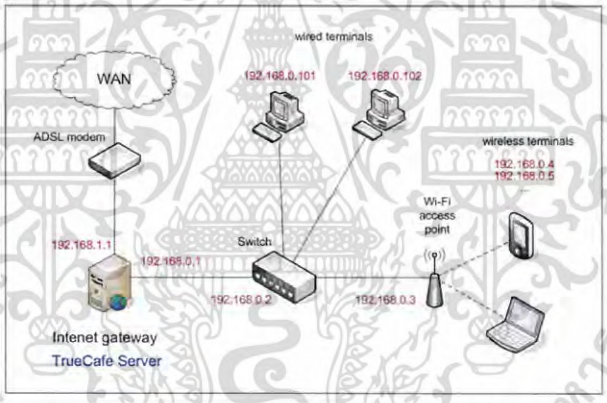
สามารถกรองข้อมูลได้เช่นเดียวกับบริดจ์แต่จะมีความสามารถมากกว่าตรงที่สามารถหาเส้นทางในการส่งกลุ่มข้อมูล (data packer) ไปยังเครื่องปลายทางในระยะทางที่สั้นที่สุดได้

2.3.2.5 สวิตช์ (switch)

นำความสามารถของฮับกับบริดจ์มารวมกันแต่การส่งข้อมูลจากคอมพิวเตอร์ตัวหนึ่งจะไม่กระจายไปยังคอมพิวเตอร์ทุก เครื่องเหมือนกับฮับ เพราะสวิตช์จะทำหน้าที่รับกลุ่มข้อมูลมาตรวจสอบก่อนว่าเป็นของคอมพิวเตอร์ เครื่องใด แล้วนำข้อมูลนั้นส่งต่อไปยังคอมพิวเตอร์เป้าหมาย ซึ่งช่วยลดปัญหาการชนหรือความคับคั่งของข้อมูล

2.3.2.6 เกตเวย์ (gateway)

เป็นอุปกรณ์ที่ทำหน้าที่เชื่อมต่อเครือข่ายต่างๆเข้าด้วยกันไม่ว่าเครือข่ายนั้นจะใช้โพรโทคอลตัวใดก็ตาม เนื่องจากเกตเวย์สามารถแปลงรูปแบบแพ็กเก็ตของโพรโทคอลหนึ่งไปเป็นรูป แบบของอีกโพรโทคอลหนึ่งได้ เพื่อให้เหมาะสามกับการใช้งานในเครือข่าย ทำให้สามารถเชื่อมต่อกับเครือข่ายอื่นๆ ได้อย่างไม่มีข้อจำกัด ดังรูปที่ 2.4 แต่ในปัจจุบันนี้ได้รวมการทำงานของเกตเวย์ไว้ในอุปกรณ์จัดเส้นทาง (router) แล้ว ทำให้อุปกรณ์จัดเส้นทางสามารถทำงานเป็นเกตเวย์ได้ จึงไม่จำเป็นต้องซื้อเกตเวย์อีก



รูปที่ 2.4 เครือข่ายเกตเวย์ (Gateway)

2.3.3 ระบบ Cloud

Cloud คือการทำงานร่วมกันของเซิร์ฟเวอร์จำนวนมากโดยแบ่งชั้นการประมวลผลออกจากชั้นเก็บข้อมูล คำว่า Cloud หรือ Cloud Datacenter ในวงการไอทีนั้นเป็นคำเปรียบเปรยสำหรับศูนย์ข้อมูลยุคใหม่ โดย “Cloud คือเทคโนโลยีที่เริ่มต้นมาจากการทำ Virtualization เป็นพื้นฐาน และถูกพัฒนาขึ้นจนให้ผู้ใช้งานสามารถให้บริการตัวเองได้ (Self-service)

ชั้นการประมวลผล (Computing layer) เป็นการร่วมกันทำงานของเซิร์ฟเวอร์จำนวนมาก แม้มีเซิร์ฟเวอร์ใดเสียหาย ก็จะไม่มีผลกับการใช้งานของลูกค้า เพราะจะสวิทช์การทำงานไปยังเซิร์ฟเวอร์ตัวอื่นแทนโดยอัตโนมัติในทันที เว็บหรือเซิร์ฟเวอร์เสมือนของท่านจะทำงานประมวลผลในชั้นนี้ ซึ่งระบบจะแบ่งทรัพยากร CPU, หน่วยความจำ ให้ตามจำนวน

ที่ท่านใช้งาน และแยกทรัพยากรกับผู้อื่นอย่างชัดเจน พร้อม Firewall ป้องกันระบบของท่านจากผู้ใช้อื่น

ชั้นเก็บข้อมูล (Storage layer) เป็นการทำงานร่วมกันของระบบเก็บข้อมูลแบบ SAN ที่มีความเสถียร และความเร็วสูง โดยสามารถย้ายไปใช้งาน SAN สำรองได้ทันทีที่เกิดเหตุขัดข้องเสียหายของอุปกรณ์หลัก โดยส่วนใหญ่จะใช้ SAN อย่างน้อย 2 ตัว ซึ่งมีข้อมูลที่เหมือนกัน (Replicate) ตลอดเวลา ข้อมูลต่างๆ ของลูกค้าจะถูกเก็บไว้ที่ชั้นนี้

เครือข่ายเน็ตเวิร์คความเร็วสูงจะเป็นตัวเชื่อมระหว่างชั้นการประมวลผล และชั้นเก็บข้อมูล เพื่อให้มั่นใจว่าสามารถรับส่งข้อมูลได้อย่างทันใจตลอดเวลา ระบบ Cloud บางแบบยังรองรับการขยายหรือหดตัวโดยอัตโนมัติสำหรับเซิร์ฟเวอร์เสมือน หรือเว็บของลูกค้าเมื่อการใช้งานเพิ่มหรือลดตามที่ได้กำหนดไว้

ดังนั้น ด้วยระบบ Cloud แท้จริง โดยการทำงานร่วมกันของเซิร์ฟเวอร์จำนวนมาก และการแยกส่วนของการทำงานแบบเป็นระบบนี้ ทำให้การทำงานของเว็บหรือเซิร์ฟเวอร์เสมือนของผู้ใช้ไม่ติดขัดและมั่นใจได้ตลอดเวลา แตกต่างจากเว็บโฮสติ้งหรือเซิร์ฟเวอร์ธรรมดาทั่วไป ที่หากเกิดการติดขัดเสียหายของอุปกรณ์นั้นๆก็จะทำให้การทำงานหยุดลงโดยไม่มีระบบทดแทน

2.4 โปรโตคอล Modbus (Modbus Protocol)

โพรโตคอล คือ กฎหรือข้อกำหนดของการสื่อสาร โดยก่อนที่ผู้ส่งและผู้รับจะสามารถติดต่อกัน ได้นั้น จะต้องสร้างข้อตกลงของการสื่อสารกันก่อน เพื่อที่จะให้สามารถที่จะเข้าใจกันได้ เช่น สมมติว่าคนไทยต้องการพูดกับคนต่างชาติ จะต้องมีการตกลงกันก่อนว่าในการสื่อสารกันนั้นจะใช้ภาษาอะไรเพื่อที่จะให้พูดจาเป็นภาษาเดียวกันและเข้าใจตรงกัน

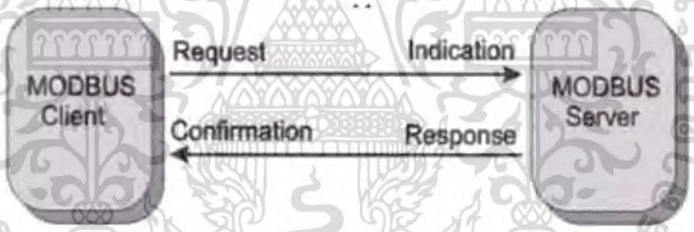
โพรโตคอลมีอยู่ด้วยกันหลายแบบ เช่น โพรโตคอล PROFIBUS, โพรโตคอล TCP/IP, โพรโตคอล Modbus เป็นต้น ซึ่งในการทำวิจัยครั้งนี้จะใช้โพรโตคอลแบบ Modbus TCP/IP

โปรโตคอล Modbus เป็นโปรโตคอลเพื่อใช้สื่อสารข้อมูลอินพุต เอาต์พุตและรีจีสเตอร์ภายใน PLC ได้รับการพัฒนาขึ้นในปี พ.ศ. 2522 โดย Modicon (ปัจจุบันคือบริษัท Schneider Electric) โปรโตคอลModbusได้เป็นที่ยอมรับกันอย่างแพร่หลายในการติดต่อสื่อสารที่เป็นแบบเน็ตเวิร์คโปรโตคอล เนื่องจาก Modbus เป็นระบบเปิด ไม่มีค่าใช้จ่ายเชื่อมต่อง่ายและพัฒนาง่ายอีกทั้งยังสามารถนำโปรโตคอลนี้ไปใช้งานในอุปกรณ์อื่นๆได้ เช่น Digital Power Meter, RTU (Remote Terminal Unit), Remote I/O และ PLC เป็นต้น นอกจากนี้ MODBUS และยังสามารถใช้งานร่วมกับแอพพลิเคชั่นจำพวก SCADA และ HMI Software ได้อีกด้วย มาตรฐาน Modbus ประกอบด้วยโปรโตคอลในระดับแอพลิ

เคชั่น (OSI ชั้นที่ 7) ที่ใช้การสื่อสารแบบไคลเอนต์/เซิร์ฟเวอร์ (Client/Server) ระหว่างอุปกรณ์ที่ถูกเชื่อมต่อกับระบบบัสของเครือข่าย

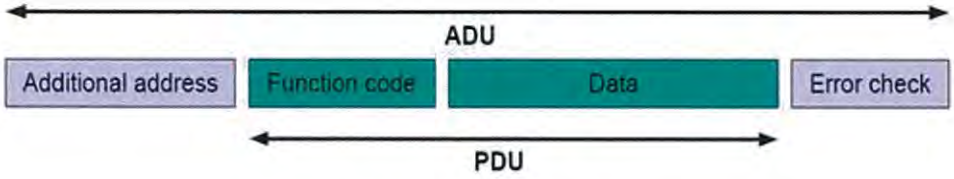
โปรโตคอล Modbus ใช้โครงสร้างการสื่อสารแบบไคลเอนต์/เซิร์ฟเวอร์ โดยมีโหมดการทำงานแบบ รีเควสต์/เรสปอนส์ (Request/Response) หรือการร้องขอและการตอบสนอง โดยไม่มีวิธีการควบคุมการเข้าถึงสื่อ (Media Access Control) ที่ถูกใช้ในเลเยอร์ที่ 2 โมเดลไคลเอนต์/เซิร์ฟเวอร์มีรูปแบบเมสเสจ (Message) อยู่ 4 ชนิดหลัก จากมุมมองของไคลเอนต์และเซิร์ฟเวอร์

1) Modbus Request เมสเสจร้องขอส่งลงบนเครือข่ายที่ฝั่งไคลเอนต์เพื่อที่จะเริ่มต้นทำทรานแซคชั่น (Transaction)
2) Modbus Confirmation เมสเสจตอบสนองเพื่อยืนยันการทำงานที่ฝั่งไคลเอนต์
3) Modbus Indication เมสเสจร้องขอที่ถูกรับที่ฝั่งเซิร์ฟเวอร์
4) Modbus Response เมสเสจตอบสนองที่ถูกส่งจากเซิร์ฟเวอร์การส่งรับเมสเสจของโมเดลไคลเอนต์/เซิร์ฟเวอร์จะถูกใช้เพื่อแลกเปลี่ยนข้อมูลระหว่างสองอุปกรณ์หรือระหว่างซอฟต์แวร์ เช่น HMI/SCADA กับอุปกรณ์ที่ใช้โปรโตคอล Modbus ดังรูปที่ 2.5



รูปที่ 2.5 การแลกเปลี่ยนเมสเสจของ Modbus

ในกรณีที่ทุกอย่างเป็นปกติ การแลกเปลี่ยนข้อมูลระหว่างไคลเอนต์และเซิร์ฟเวอร์จะเป็นรูปที่ 5 ไคลเอนต์ (อุปกรณ์ที่เป็นมาสเตอร์) จะเริ่มทำการร้องขอข้อมูล ตัวโปรโตคอล Modbus ส่วนใหญ่จะทำงานอยู่ที่ชั้นที่ 7 จะสร้างฟอร์แมตข้อมูลที่เรียกว่า POU (Protocol Data Unit) ซึ่งประกอบด้วยรหัสฟังก์ชั่น หรือ Function Code และข้อมูลที่ทำการร้องขอ ณ ที่ระดับ OSI ชั้นที่ 2 ข้อมูล PDU จะถูกเพิ่มเติมจนเป็น ADU (Application Data Unit) ดังรูปที่ 2.6 โดยการเพิ่มฟิลด์ที่เกี่ยวกับการระบุอุปกรณ์ เช่น หมายสเลฟ และค่าสำหรับใช้ในการตรวจสอบว่าข้อมูลผิดพลาด (Error Detection)



รูปที่ 2.6 รูปแบบเฟรม Modbus

เซิร์ฟเวอร์ (อุปกรณ์สเลฟ) จะทำงานตามคำขอและเริ่มตอบสนองการร้องขอ การปฏิสัมพันธ์ระหว่างไคลเอนต์และเซิร์ฟเวอร์ถูกแสดงดังรูปที่ 2.7

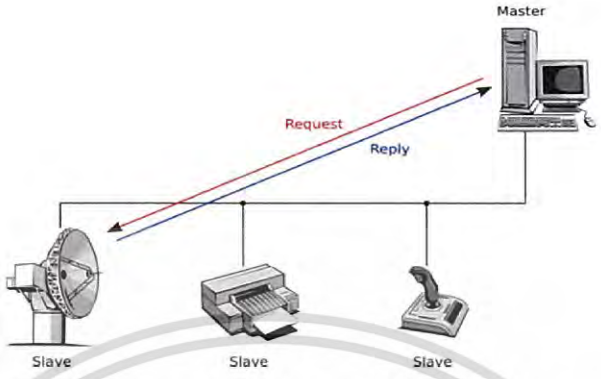
รูปที่ 2.7 ปฏิสัมพันธ์แบบ ไคลเอนต์/เซิร์ฟเวอร์ ของ Modbus

2.4.1 หลักการมาสเตอร์/สเลฟ ของ Modbus

โปรโตคอล Modbus เป็นการสื่อสารข้อมูลในลักษณะมาสเตอร์/สเลฟ ซึ่งเป็นการสื่อสารจากอุปกรณ์แม่หรืออุปกรณ์มาสเตอร์เครื่องเดียว ส่วนใหญ่มักเป็นซอฟต์แวร์คอมพิวเตอร์หรืออุปกรณ์แสดงผล HMI ไปยังอุปกรณ์ลูกหรืออุปกรณ์สเลฟได้หลายๆเครื่อง สเลฟอาจเป็นอุปกรณ์ต่อพ่วงใด ๆ (I/O transducer, วาล์ว, ไดร์ฟเครือข่ายหรืออุปกรณ์วัดอื่นๆ) ซึ่งประมวลผลและส่งข้อมูลไปยังอุปกรณ์มาสเตอร์ การสื่อสารจะต้องเริ่มต้นที่ตัวมาสเตอร์เสมอ ตัวสเลฟจะไม่สามารถตอบสนองหรือส่งข้อมูลใดๆได้ ถ้าไม่มีการร้องขอจากมาสเตอร์และระหว่างสเลฟด้วยกันเองจะไม่มีการสื่อสารระหว่างกัน มาสเตอร์สามารถส่งการร้องขอไปยังสเลฟได้ 2 วิธีดังต่อไปนี้

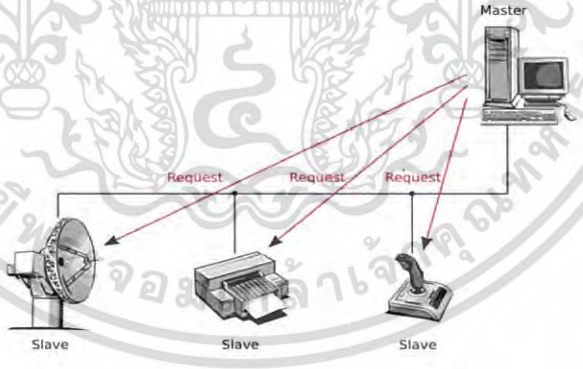
1) โหมดยูนิคาสต์ (Unicast Mode) ในโหมดนี้มาสเตอร์จะใช้แอดเดรสหรือหมายเลขแบบระบุตัวสเลฟ หลังจากที่สเลฟรับและประมวลผลการร้องขอสเลฟจะตอบกลับโดยเฟรมจะมีแอดเดรสของตัวมันตอบกลับไปยังมาสเตอร์ในโหมดนี้การสื่อสารจะมี2เมสเสจคือเมสเสจการร้องขอจากมาสเตอร์ และเมสเสจการตอบสนองจากสเลฟ โดยที่แต่ละเมสเสจต้องมี

หมายเลขแอดเดรสอยู่ในช่วงจาก 1 ถึง 247 และไม่ซ้ำกันจึงจะมีความอิสระจากสเลฟตัวอื่นๆ



รูปที่ 2.8 การร้องขอแบบยูนิคาสต์

2) โหมดบรอดคาสต์ (Broadcast Mode) ในโหมดนี้มาสเตอร์สามารถส่งการร้องขอไปยังทุกสเลฟในเวลาเดียวกัน แต่จะไม่มีเมสเสจตอบกลับมาจากสเลฟใดๆ การร้องขอในโหมดนี้มักเป็นคำสั่งประเภทเขียนทุกสเลฟต้องยอมรับคำขอนี้สำหรับฟังก์ชั่นประเภทเขียนแอดเดรสหมายเลข 0 ถูกใช้ในกำหนดเมสเสจให้เป็นเมสเสจบรอดคาสต์



รูปที่ 2.9 การร้องขอแบบบรอดคาสต์

โปรโตคอล Modbus สามารถใช้หมายเลขได้จำนวน 256 หมายเลข แอดเดรสหมายเลข 0 ถูกจองไว้สำหรับบรอดคาสต์ สเลฟทุกตัวต้องรับรู้และตอบสนองต่อแอดเดรสนี้ยกเว้นฟังก์ชั่นประเภทอ่านข้อมูล

2.4.2 ไดอะแกรมสถานะของมาสเตอร์/สเลฟ

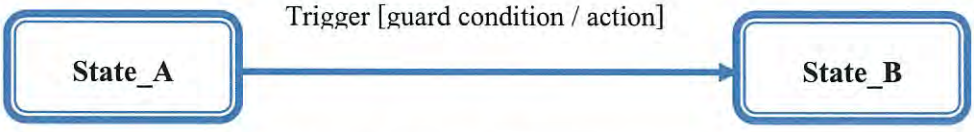
ในชั้นดาต้าลิงก์ของ Modbus จะประกอบด้วยสองส่วนดังต่อไปนี้

1) โปรโตคอล มาสเตอร์/สเลฟ

2) โหมดการส่ง (RTU และ ASCII)

2.4.2.1 ซินแทกซ์ของไดอะแกรมสถานะ

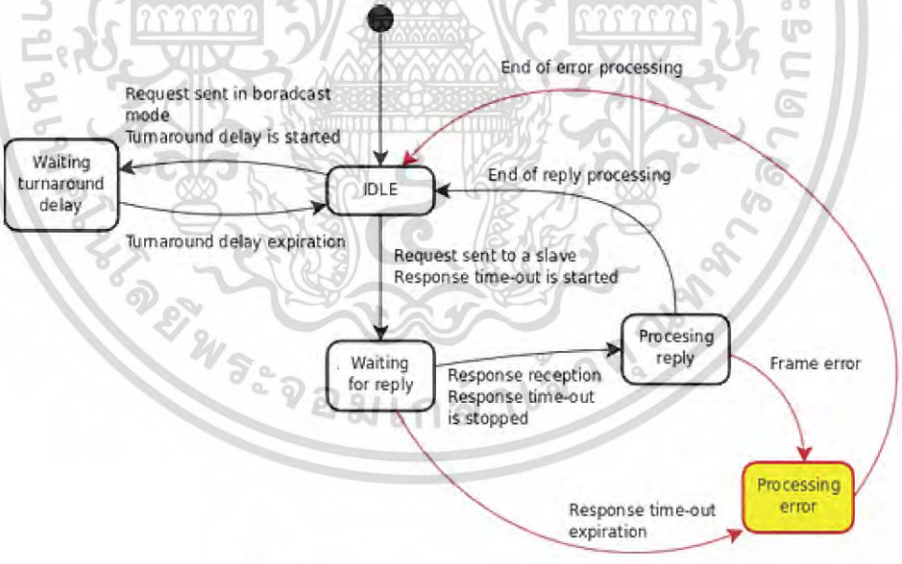
เพื่อความเข้าใจในไดอะแกรมสถานะ ซินแทกซ์นี้จะช่วยให้เข้าใจยิ่งขึ้น



รูปที่ 2.10 ซินแทกซ์ของไดอะแกรมสถานะ

เมื่อ Trigger หรือเกิดการเปลี่ยนแปลงขึ้นในระบบโดยที่ระบบอยู่ในสถานะ A แล้วสถานะ A ของระบบจะเปลี่ยนไปยังสถานะ B ถ้า Guard Conditionหรือเงื่อนไขถูกต้องเป็นจริงและระบบจะทำดำเนินการตามฟังก์ชั่นที่ถูกตั้งไว้ ดังรูปที่ 2.10

2.4.2.2 ไดอะแกรมสถานะของมาสเตอร์



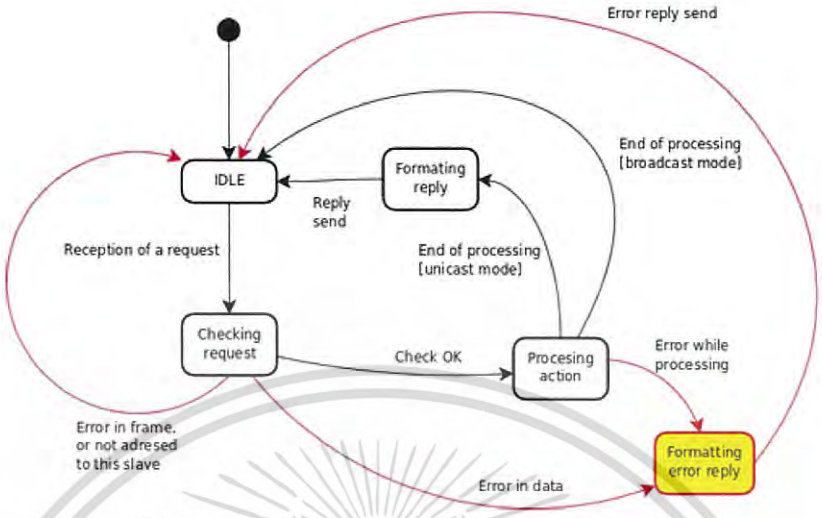
รูปที่ 2.11 ไดอะแกรมสถานะของมาสเตอร์

1) สถานะว่าง คือไม่มีการร้องขอ สถานะนี้จะเริ่มหลังจากเริ่มเปิดใช้งานอุปกรณ์หลังจากที่ได้ส่งการร้องขอมาสเตอร์จะออกจากสถานะว่าง และจะไม่ส่งการ ร้องขอที่สองในเวลาเดียวกัน
2) เมื่อการร้องขอแบบยูนิคาสต์ได้ถูกส่งไปยังสเลฟ มาสเตอร์จะไปอยู่

ที่สถานะรอ การตอบกลับและตัวจับเวลาเพื่อรอการตอบสนองก็จะเริ่มนับเวลาซึ่งมันจะป้องกันไม่ให้มาสเตอร์รอการตอบกลับอย่างไม่มีที่สิ้นสุด เวลาในการรอการ ตอบสนองจะขึ้นอยู่กับการตั้งค่าของผู้ใช้หรือผู้ผลิต

3) เมื่อได้รับการตอบกลับ มาสเตอร์จะตรวจสอบการตอบกลับก่อนที่จะประมวลผลข้อมูลภายในการตรวจสอบอาจจะตรวจเจอข้อผิดพลาดได้ยกตัวอย่างการตอบกลับอาจจะมาจากสเลฟที่ไม่ได้ถูกร้องขอ หรือมีความ ผิดพลาดในเฟรมที่รับมา ในกรณีที่การตอบกลับมาจากตัวสเลฟที่ไม่ต้องการตัวจับเวลารอการตอบกลับก็ยังวิ่งจับเวลารอต่อไป ในกรณีตรวจเจอความ ผิดพลาดในเฟรมอาจจะมีการส่งการร้องขอซ้ำอีกครั้ง
4) ถ้าไม่มีการตอบกลับตัวจับเวลารอการตอบกลับจนหมดเวลา พร้อมทั้งแจ้งความผิดพลาดให้ระบบ แล้วมาสเตอร์จะเปลี่ยนไปอยู่ในสถานะว่าง และส่ง การร้องขอซ้ำ จำนวนการส่งซ้ำจะขึ้นอยู่กับการตั้งค่าของผู้ใช้หรือผู้ผลิต
5) เมื่อมีการร้องขอแบบบรอดคาสต์ส่งลงบนบัสจะต้องไม่มีการตอบกลับใดๆ จากสเลฟ ดังนั้นเวลาการรอจะขึ้นอยู่กับมาสเตอร์เพื่อรอให้ทุกสแลฟประมวลผลการร้องขอล่าสุดจนเสร็จสิ้น ก่อนที่จะส่งการร้องขอใหม่ ระยะเวลาหน่วงนี้ถูกเรียกว่า Turnaround Delay ดังนั้นมาสเตอร์จะไปอยู่ที่สถานะรอ การหน่วงเวลาจำนวน 1 รอบ รานแซคชั่นก่อนที่จะกลับไปอยู่ที่สถานะว่าง และส่งการร้องขออื่นต่อไป
6) ในโหมดยูนิคาสต์ระยะเวลารอการตอบสนองต้องนานเพียงพอสำหรับรอให้ สเลฟประมวลผลการร้องขอพร้อมระยะเวลาในการส่งการตอบรับ ในโหมด บรอดคาสต์เวลา Turnaround ต้องนานเพียงพอสำหรับสแลฟใน การประมว ผลพร้อมทั้งการรอรับการร้องขอใหม่ ดังนั้นเวลา Turnaround ต้องสั้นกว่าเวลาการรอการตอบสนองปกติ โดยทั่วไปแล้วเวลาการรอการตอบสนองจะอยู่ที่ 1 วินาทีที่บรอดเรต 9,600 bps และเวลาTurnaround ควรอยู่ที่ 100 ms. ถึง 200 ms.

2.4.2.3 ไดอะแกรมสถานะของสเลฟ



รูปที่ 2.12 ไดอะแกรมสถานะของสแลฟ

1) สถานะว่างคือไม่มีการร้องขอจากมาสเตอร์
2) เมื่อได้รับการร้องขอสเลฟจะตรวจสอบว่าเมสเสจก่อนที่จะดำเนินการตาม การร้องขอในเมสเสจมีหลายความผิดพลาดที่เกิดขึ้นได้ เช่น ฟอร์แมตผิดพลาดคำสั่งไม่ถูกสนับสนุนในกรณีผิดพลาดเหล่านี้สเลฟต้องแจ้งความผิดพลาดไปยังมาสเตอร์
3) เมื่อสเลฟดำเนินการเสร็จเรียบร้อยถ้าเป็นเมสเสจแบบยูนิคาสต์สเลฟต้องตอบกลับ หรือยืนยันกลับไปยังมาสเตอร์

2.4.3 การส่งข้อมูลแบบอนุกรม

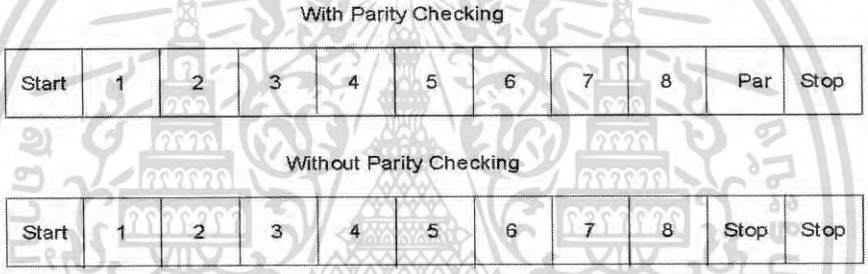
มีสองโหมดในการส่งข้อมูลแบบอนุกรมที่ถูกนิยามใน Modbus นั้นคือ โหมด RTU และ ASCII แต่ละโหมดได้ถูกนิยามการเข้ารหัสในเมสเสจและการส่งบิตข้อมูลบนสายสื่อสารแบบอนุกรม โหมดการส่งข้อมูลต้องเหมือนกันทั้งสองด้านของการสื่อสารแบบอนุกรม

Modbus RTU สมควรเป็นโหมดที่ทุกอุปกรณ์ที่สนับสนุนโปรโตคอล Modbus ต้องสนับสนุน ส่วน Modbus ASCII เป็นเพียงโหมดทางเลือกที่ใช้ในระบบงานบางระบบเท่านั้น อุปกรณ์ควรสามารถเลือกโหมดตามที่ผู้ใช้ต้องการไม่ว่าจะเป็น RTU และ ASCII โดยที่โหมดดีฟอลต์ต้องเป็นโหมด RTU

2.4.4 โหมด RTU

เมื่ออุปกรณ์สื่อสารใช้โหมด Modbus RTU แต่ละ 8 บิตในเมสเสจจะถูกแบ่งเป็น 2 ส่วน ส่วนละ 4 บิตเพื่อแทนที่หรือแสดงด้วยอักขระ ASCII 2 อักขระ ข้อดีของโหมด RTU คือสามารถลดจำนวนไบต์ที่ใช้ส่งข้อมูลได้เกือบครึ่งหนึ่งเมื่อเทียบกับโหมด ASCII แต่ละเมสเสจต้องส่งข้อมูลเป็นไบต์ต่อเนื่องกัน ในโหมด RTU การรับส่งข้อมูล 1 ไบต์ ไม่ว่าจะเป็นข้อมูลส่วนใดภายในเฟรมจะต้องทาการส่งบิตข้อมูลรวม 11 บิต คือ บิตเริ่มต้น 1 บิต, บิตข้อมูล 8 บิต, บิตตรวจสอบ Parity ของข้อมูล 1 บิตและบิตหยุด 1 บิต

โดยปกติจะใช้โหมด RTU จะใช้อีเวนพาริตี้ (Even Parity) แต่พาริตี้อื่นก็สามารถใช้ได้ เพื่อที่ให้แน่ใจว่าอุปกรณ์สามารถทำงานกับอุปกรณ์อื่นได้ อุปกรณ์ควรสนับสนุนการส่งแบบไม่มีพาริตี้ (No Parity) ถ้าไม่มีพาริตี้จะต้องมีบิตหยุดจำนวน 2 บิตในการส่งข้อมูลจำนวนหนึ่งไบต์ ดังรูปที่ 2.13

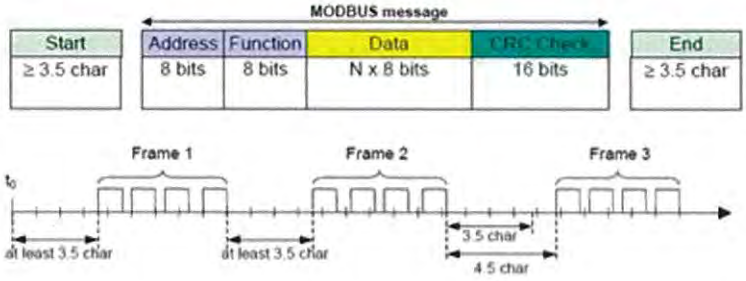


รูปที่ 2.13 หนึ่งไบต์ของ Modbus RTU

2.4.4.1 การส่งเฟรมของ Modbus RTU

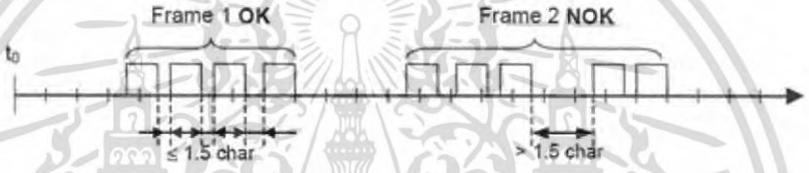
เมสเสจของ Modbus RTU จะถูกจัดว่าเป็นเฟรมที่มีจุดเริ่มและจุดจบของเฟรมชัดเจน คุณสมบัตินี้ทำให้อุปกรณ์ที่รับเฟรมสามารถรับรู้จุดเริ่มต้นของเมสเสจและสามารถรับรู้ว่าเมสเสจจบเมื่อไร เมสเสจครึ่งๆกลางๆ จะสามารถตรวจสอบเจอและจะถูกทิ้งไป

ใน Modbus RTU แต่ละเมสเสจจะต้องถูกคั้นด้วยสถานะว่างอย่างน้อยเป็นระยะเวลาในการส่งจำนวน 3.5 อักขระ ดังรูปที่ 2.14



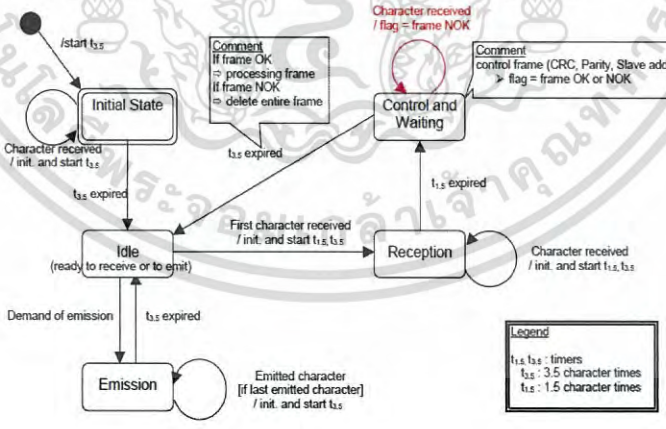
รูปที่ 2.14 การส่งเฟรมของ Modbus RTU

ตลอดทั้งเมสเสจ เฟรมต้องส่งอักขระอย่างต่อเนื่องกัน ถ้ามีช่วงว่างมากกว่า 1.5 อักขระเกิดขึ้นระหว่าง 2 อักขระ เมสเสจนั้นจะต้องถูกตีความว่าไม่สมบูรณ์และถูกเพิกเฉยและทิ้งโดยตัวรับเมสเสจ ดังรูปที่ 2.15



รูปที่ 2.15 ช่วงว่างภายในเมสเสจของ Modbus RTU

2.4.4.2 ไดอะแกรมสถานะของ Modbus RTU



รูปที่ 2.16 ไดอะแกรมสถานะของ Modbus RTU

1) การเปลี่ยนจากสถานะเริ่มต้นไปยังสถานะว่าง ต้องรอหน่วงเวลา t3.5 เพื่อทำ ให้แน่ใจระยะช่วงระหว่างเฟรม
2) สถานะว่างคือสถานะปกติที่ไม่มีการส่งและรับ

3) ในโหมด RTU การสื่อสารต้องจะกลับมาที่สถานะว่าง เมื่อไม่มีการส่งข้อมูล ใดๆนานมากกว่าช่วงเวลา 3.5 อักขระ
4) เมื่อการเชื่อมต่ออยู่ในสถานะว่าง แต่ละอักขระที่ถูกส่งบนบัสจะถูกตรวจสอบว่าเป็นจุดเริ่มต้นของเฟรมหลังจากตรวจเจอสถานะการเชื่อมต่อจะย้ายไปอยู่แอกทีฟจุดจบของเฟรมสามารคาดการณ์
5) เมื่อฟิลด์หมายเลขแอดเดรสได้ถูกวิเคราะห์ว่าเป็นเฟรมของอุปกรณีนั้นๆถ้าเฟรมไม่มีข้อผิดพลาดจนถูกทิ้ง หลังจากตรวจเจอจุดจบของเฟรมการคำนวณCRCและการตรวจสอบก็จะถูกดำเนินการเพื่อที่จะลดเวลาในการประมวลผล ของการรับเฟรม หมายเลขแอดเดรสของอุปกรณ์จะถูกตรวจสอบทันทีโดยไม่ จำเป็นต้องรอจนจบเฟรมส่วนตัว CRC จะถูกคำนวณก็ต่อเมื่อเฟรมนั้นเป็นของอุปกรณ์หรือเป็นเมสเสจประเภทบรอดคาสต์

2.4.4.3 การตรวจสอบ CRC

Modbus RTU มีฟิลด์สำหรับตรวจสอบความผิดพลาดที่ใช้หลักการของ CRC (Cyclical Redundancy Checking) โดยค่าจะคำนวณจากข้อมูลภายในเมสเสจ ตัว CRC มีไว้ตรวจสอบเนื้อหาจริงๆภายในเมสเสจทั้งหมดโดยอิสระจากการตรวจสอบของพาริตี้ที่ใช้ตรวจสอบเบื้องต้นในแต่ละอักขระ ตัว CRC จะมีทั้งหมด 16 บิตโดยแบ่งเป็นจำนวน 2 ไบต์ CRC จะถูกเติมต่อท้ายของเมสเสจโดยที่ไบต์ต่ำจะถูกเติมก่อนตามด้วยไบต์สูง ดังนั้นไบต์สูงของ CRC จะถูกส่งเป็นลำดับสุดท้ายของเมสเสจของ Modbus RTU

ค่าของ CRC จะถูกคำนวณโดยอุปกรณ์ที่ส่งเมสเสจ อุปกรณ์ที่รับเมสเสจนี้จะทำการคำนวณซ้ำอีกครั้งระหว่างที่รับเมสเสจ และจะเปรียบเทียบกับค่า CRC ที่มากับเมสเสจ ถ้าทั้งสองค่าไม่เท่ากันแสดงว่ามีความผิดพลาดเกิดขึ้น อีกวิธีการหนึ่งที่ใช้ในการตรวจสอบเมสเสจที่ได้รับนั้นคือการคำนวณหาค่า CRC จากเนื้อหารวมทั้งค่า CRC ที่ส่งมา ผลลัพธ์ที่ได้ก็คือ ถ้าข้อมูลทุกค่าถูกต้องนั้น CRC ที่คำนวณได้ต้องมีค่าเท่ากับศูนย์

การคำนวณ CRC เริ่มที่การตั้งค่าเริ่มต้นขนาด 16 บิตเข้าไปที่ตัวรีจิสเตอร์ที่มีทุกบิตมีค่าเท่ากับหนึ่ง หลังจากนั้นเริ่มนำข้อมูลแต่ละไบต์ของเมสเสจเข้ามาคำนวณกับค่าภายในรีจิสเตอร์ ระหว่างการคำนวณ CRC แต่ละไบต์จะถูกดำเนินการ XOR (Exclusive OR) กับค่าในรีจิสเตอร์ แล้วผลรับที่ได้ก็จะถูกชิฟต์ (Shift) หนึ่งบิตไปยังทิศทางทาง LSB พร้อมทั้งเติมบิตค่าเท่ากับศูนย์ที่ตำแหน่ง MSB

ตัวบิตที่ถูกชิฟต์ออกจะถูกนำมาตรวจสอบ ถ้ามีค่าเท่ากับหนึ่งแล้วค่าในรีจิสเตอร์จะถูกทำ XOR กับรหัสโพลีโนเมียล (Polynomial Code) โดยผลลัพธ์ก็ถูกนำไปเก็บที่รีจิสเตอร์ตัวเดิม แต่ถ้ามีค่าเท่ากับศูนย์ จะไม่ทำการ XOR กระบวนการนี้จะถูกทำซ้ำจน

กระทั้งมีการชิฟต์จำนวนแปดครั้งหรือแปดบิต หลังจากการชิฟต์ครั้งที่แปด ไบต์ถัดไปจะถูกทำ XOR กับค่าในรีจิสเตอร์ปัจจุบัน และทำตามขั้นตอนเดิมจำนวนแปดครั้ง ค่าสุดท้ายของรีจิสเตอร์หลังจากที่ทุกไบต์ของเมสเสจได้ถูกนำมาคำนวณนั้นก็คือค่า CRC

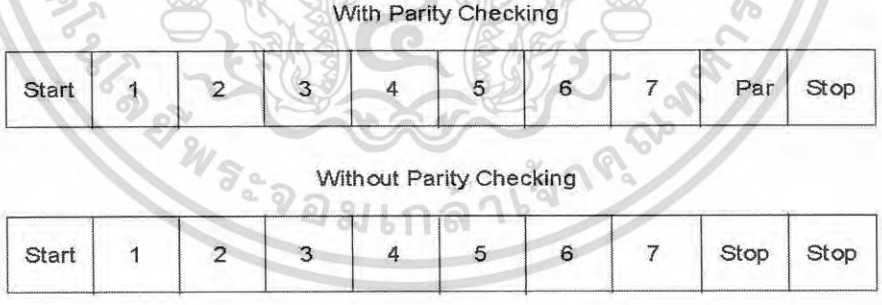
2.4.5 โหมด ASCII

เมื่ออุปกรณ์ถูกตั้งค่าให้ใช้โหมด Modbus ASCII (American Standard Code for Information Interchange) แต่ละไบต์ในเมสเสจจะถูกส่งโดยใช้ 2 อักขระของรหัส ASCII โหมดนี้ถูกใช้ในการเชื่อมต่อถ้าอุปกรณ์ไม่สามารถใช้โหมด RTU อันเนื่องประสิทธิภาพฮาร์ดแวร์และการจัดการตัวจับเวลามีไม่เพียงพอ โหมดนี้โดยภาพรวมมีประสิทธิผลต่ำกว่าโหมด RTU เพราะแต่ละไบต์ข้อมูลต้องการ 2 อักขระ ASCII

ยกตัวอย่าง ไบต์ข้อมูล 0x5B จะถูกเข้ารหัสเป็นสองอักขระ ASCII คือ 0x35 และ 0x42 (0x35="5" และ 0x42="B" ในตาราง ASCII)

ฟอร์แมตที่ใช้ส่งแต่ละไบต์ (จำนวน 11 บิต) ในโหมด ASCII ระบบการเข้ารหัสเลขฐาน 16 ใช้อักขระ ASCII 0-9, A-F

จำนวนบิตต่อไบต์ 1 บิตเริ่ม, 7 บิตข้อมูล ส่งบิต LSB ก่อน, 1 บิตสำหรับพาริตี้, 1 บิตหยุดโดยปกติโหมด ASCII จะใช้ อีเวนพาริตี้ (Even Parity) แต่พาริตี้อื่น ๆ ก็สามารถใช้ได้ เพื่อที่ให้แน่ใจว่าอุปกรณ์สามารถทำงานกับอุปกรณ์อื่นได้ อุปกรณ์สมควรสนับสนุนโนพาริตี้ (No Parity) หรือไม่มีพาริตี้ ถ้าไม่มีพาริตี้จะต้องมีบิตหยุดจำนวน 2 บิต ดังรูปที่ 2.17



รูปที่ 2.17 ฟอร์แมตในการส่งข้อมูลหนึ่งไบต์ของ Modbus ASCII

2.4.5.1 การส่งเฟรมของ Modbus ASCII

เมสเสจของ Modbus ASCII จะถูกจัดเป็นเฟรมที่มีจุดเริ่มและจุดจบของเฟรม เช่นเดียวกับ Modbus RTU คุณสมบัตินี้ทำให้อุปกรณ์ที่รับเฟรมสามารถรับรู้จุดเริ่มต้นของเมสเสจและสามารถรับรู้ว่าเมสเสจจบเมื่อไร เมสเสจครึ่งๆกลางๆจะสามารถถูกตรวจสอบเจอใน Modbus ASCII แต่ละเมสเสจจะต้องถูกแบ่งแยกด้วยอักขระที่ความหมาย

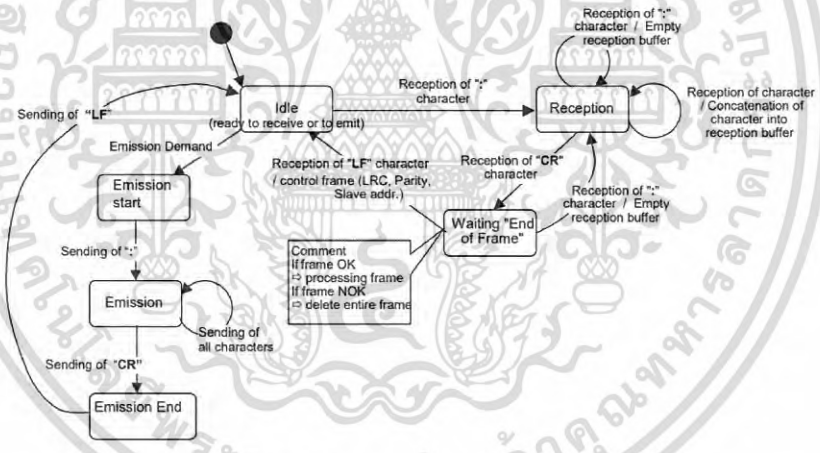
พิเศษที่ได้กำหนดไว้ก่อนแล้ว เช่น เป็นจุดเริ่มต้นของเฟรมและจุดจบของแฟรม เมสเสจทุกเมสเสจต้องเริ่มต้นอักขระโคลอน ":" รหัส ASCII 0x3A) และจบด้วยอักขระเป็นคู่นั้นคือ CRLF (รหัส ASCII 0xD และ 0xA) หรือการขึ้นบรรทัดใหม่นั้นเอง

อักขระที่ใช้ได้ในการส่งคืออักขระ ASCII ที่ใช้สื่อถึงเลขเดี่ยวในฐาน 16 (0-9, A-F) อุปกรณ์จะคอยตรวจอักขระ ":" เมื่อมันตรวจเจอมันจะพยายามแปลความหมายทุกอักขระจนกระทั่งเจอจุดจบของเฟรม ดังรูปที่ 2.18

Start	Address	Function	Data	LRC	End
1 char :	2 chars	2 chars	0 up to 2x252 char(s)	2 chars	2 chars CR,LF

รูปที่ 2.18 การส่งเฟรมของ Modbus ASCII

2.4.5.2 ไดอะแกรมสถานะของ Modbus ASCII



รูปที่ 2.19 ไดอะแกรมสถานะของ Modbus ASCII

1) สถานะว่างคือสถานะปกติไม่มีการส่งและรับ
2) แต่ละการรับอักขระ ":" หมายความถึงจุดเริ่มต้นของเมสเสจใหม่ ถ้าเมสเสจกำลังประมวลผลการรับอยู่และได้รับอักขระ ":" อีก เมสเสจปัจจุบันที่กำลัง แปลจะถูกตีความว่าไม่สมบูรณ์และถูกทิ้งไปรวมทั้งบัฟเฟอร์จะถูกเคลียร์เริ่ม ใหม่อีกด้วย
3) หลังจากตรวจเจอจุดจบของเฟรมLRCจะถูกคำนวณและตรวจสอบว่าถูกต้อง หรือไม่ เพื่อที่จะลดเวลาในการประมวลผล แอดเดรสจะถูกวิเคราะห์ตั้งแต่แรกโดยไม่ต้องรอถึงจุดจบของเฟรม

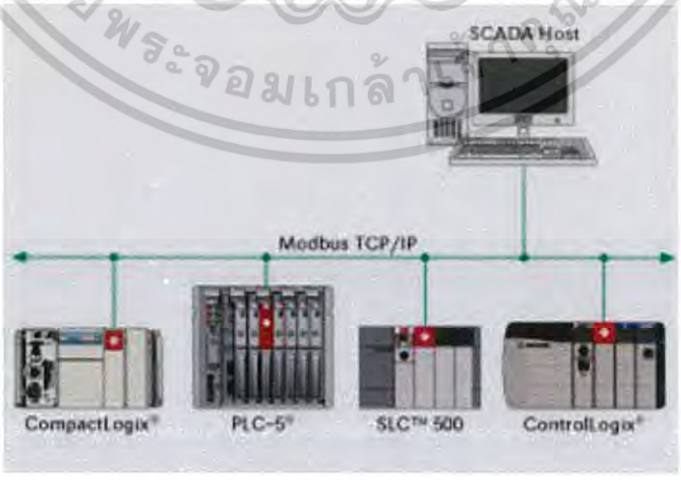
2.4.5.3 การตรวจสอบ LRC

ในโหมด Modbus ASCII เมสเสจจะมีการตรวจสอบความผิดพลาดโดยใช้วิธีการ LRC (Longitudinal Redundancy Checking) โดยจะคำนวณบนเนื้อหาของเมสเสจโดยไม่รวมตัว ":" และ "CRLF" ตัว LRC จะมีขนาดเพียงหนึ่งไบต์ ค่า LRC จะถูกคำนวณโดยอุปกรณ์ที่ส่งเมสเสจและต่อท้ายเข้าไปในเมสเสจ อุปกรณ์ที่รับเมสเสจจะคำนวณ LRC ระหว่างการรับเมสเสจ และเปรียบเทียบกับค่ากับค่า LRC ที่มากับเมสเสจ ถ้าค่าไม่เท่ากันแสดงว่ามีความผิดปกติเกิดขึ้น อีกวิธีการหนึ่งคือการคำนวณ LRC ใหม่รวมกับค่า LRC ที่มากับเมสเสจค่าที่ได้จะต้องเป็นศูนย์จึงหมายความว่าเมสเสจนั้นไม่ผิดพลาด

ค่า LRC ถูกคำนวณโดยวิธีการบวกค่าแต่ละไบต์ในเมสเสจเข้าด้วยกันโดยไม่สนใจค่าที่ทดเกินขนาดแปดบิต และทำการ 2's Complement ผลบวกให้กลายเป็นทางด้านลบ ในโหมด ASCII ค่า LRC จะถูกเปลี่ยนเป็นรหัส ASCII โดยใช้สองไบต์แล้วจัดวางไว้ที่ท้ายเมสเสจก่อน "CRLF"

2.4.6 MODBUS TCP/IP

MODBUS TCP/IP ถูกพัฒนาขึ้นโดยมีวัตถุประสงค์เพื่อจะนำการสื่อสารแบบอินเตอร์เน็ต มาใช้กับอุปกรณ์จำพวก Ethernet Device ระยะในการใช้งานสำหรับการเดินสาย (สาย LAN) คือ 100 m โดยสามารถขยายระยะในการสื่อสารได้โดยการใช้อุปกรณ์รีพีตเตอร์หรือในระบบ LAN จะเรียกอุปกรณ์นี้ว่าฮับหรือสวิตช์ก็จะสามารถลากสายได้อีก 100 เมตร และยังสามารถต่อรีพีตเตอร์ขยายระยะทางได้โดยไม่จำกัด ในการสื่อสารโดยทั่วไปมีความเร็ว 100,000,000 bps (100 Mbps) และเชื่อมต่ออุปกรณ์ได้ไม่จำกัดจำนวน ดังรูปที่ 2.20

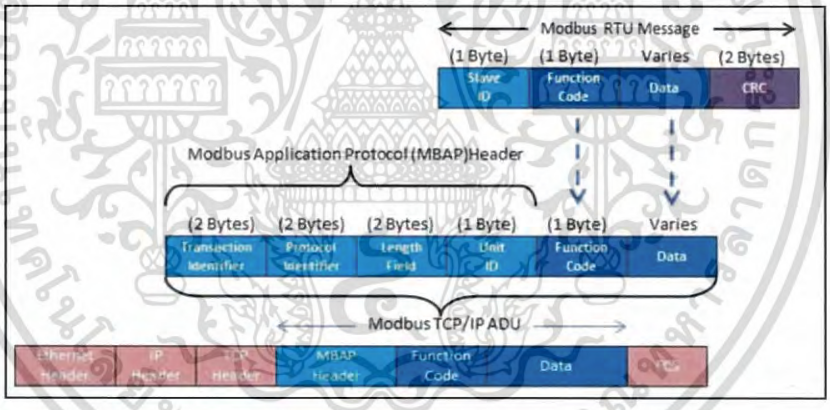


รูปที่ 2.20 การใช้ Modbus TCP/IP กับอุปกรณ์จาพวก Ethernet Device

MODBUS ASCII/RTU ที่จะติดต่อสื่อสารกับ MODBUS TCP เพื่อให้ใช้งานในเครือข่ายอีเทอร์เน็ตจะใช้เกตเวย์ติดต่อและแปลงรูปแบบการสื่อสารข้อมูล โดยการสื่อสารของ MODBUS RTU/ASCII จะเป็นการสื่อสารผ่านทาง RS-232/422/485 นั้นจะถูกเกตเวย์แปลงให้เป็น MODBUS TCP เพื่อใช้ในการติดต่อสื่อสารในเครือข่ายอีเทอร์เน็ตต่อไป

รูปที่ 2.21 การแปลง MODBUS Serial เป็น MODBUS Ethernet

Modbus Application Protocol (MBAP) จะประกอบด้วยข้อมูล 7 ไบต์ซึ่งจะวางหน้า Modbus RTU message ดังรูปที่ 2.22 และ 2.23 โดยมีรายละเอียดดังนี้



รูปที่ 2.22 ลักษณะเฟรมข้อมูลของ MODBUS TCP/IP

Fields	Length	Description -	Client	Server
Transaction Identifier	2 Bytes	Identification of a MODBUS Request / Response transaction.	Initialized by the client	Recopied by the server from the received request
Protocol Identifier	2 Bytes	0 = MODBUS protocol	Initialized by the client	Recopied by the server from the received request
Length	2 Bytes	Number of following bytes	Initialized by the client (request)	Initialized by the server (Response)
Unit Identifier	1 Byte	Identification of a remote slave connected on a serial line or on other buses.	Initialized by the client	Recopied by the server from the received request

รูปที่ 2.23 ลักษณะเฟรมข้อมูลของ MODBUS TCP/IP ในส่วน MBAP Header

2.4.6.1 MBAP Header

1) Transaction/invocation Identifier (2 ไบต์) ใช้จับคู่การแลกเปลี่ยนข้อมูลเมื่อมี Message หลายๆ ชุด ถูกส่งออกมาด้วยTCP เดียวกัน ด้วย Client ตัวใดตัวหนึ่ง โดยไม่ต้องรอลำดับการ Response
2) Protocol Identifier (2 ไบต์) ใช้สำหรับการมัลติเพล็กซ์ภายใน ระบบ MODBUS ในส่วนใหญ่จะมีค่าเป็น 0 เสมอ
3) Length (2 ไบต์) เป็นการระบุจำนวน Byte ที่รวมจานวน Byte ของ unit identifier, function code, และ data fields
4) Unit Identifier (1 byte) เป็นการระบุ ID ของเซิร์ฟเวอร์ที่อยู่ในระบบสื่อสารอาจตั้งเป็น00 ถึง FF ใช้สำหรับจุดประสงค์ในการกำหนดเส้นทางระบบ ภายใน

2.4.7 โครงสร้างของโปรโตคอล Modbus

ตารางที่ 2.1 แสดงรูปแบบทั่วไปของฟอร์แมต Modbus

Address Field	Function Field	Data Field	Error Check Field
1 Byte	1 Byte	Variable	2 Byte

ฟิลด์แรกในแต่ละเมสเสจจะเป็นฟิลด์แอดเดรส หรือฟิลด์หมายเลขของอุปกรณ์ซึ่งจะมีขนาดความยาวเพียง 1 ไบต์ ในเฟรมร้องขอไบต์นี้จะใช้ระบุถึงตัวคอนโทรลเลอร์ที่คำร้องขอจะส่งไปถึง ส่วนในเฟรมการตอบสนองจะเริ่มต้นด้วยฟิลด์หมายเลขแอดเดรสที่บอกถึงตัวอุปกรณ์ที่ตอบสนองกลับแต่ละสเลฟสามารถมีค่าของแอดเดรสอยู่ในช่วงระหว่าง 1 ถึง 247 ในทางปฏิบัติแล้วนั้นคือการจำกัดจำนวนของสเลฟ โดยที่จริงแล้วในระบบเครือข่ายจะมีมาสเตอร์หนึ่งตัว และสเลฟสองถึงสามตัวเท่านั้นเพราะถ้ามีจำนวนสเลฟมาก ประสิทธิภาพของการสื่อสารจะลดถอยลง

ฟิลด์ที่สองในแต่ละเมสเสจคือ ฟิลด์ฟังก์ชั่นซึ่งมีขนาดเพียงหนึ่งไบต์เช่นเดียวกันในการร้องขอ ไบต์นี้จะระบุฟังก์ชั่นที่ให้ตัวอุปกรณ์ปลายทางดำเนินการ ถ้าอุปกรณ์ปลายทางสามารถดำเนินการตามฟังก์ชั่นที่ร้องขอ ฟิลด์ฟังก์ชั่นในเมสเสจตอบสนองจะใช้หมายเลขฟังก์ชั่นเดียวกันกับเฟรมร้องขอให้ดำเนินการ มิฉะนั้นฟิลด์ฟังก์ชั่นจะตอบกลับโดยหมายเลขฟังก์ชั่นที่ถูกดัดแปลงโดยการเซตให้บิตตำแหน่ง MSB (Most Significant Bit) มีค่าเท่ากับหนึ่งซึ่งเป็นตัวบ่งบอกว่ามีข้อผิดพลาดเกิดขึ้น

ฟิลด์ที่สามในเมสเสจ คือ ฟิลด์ข้อมูลที่ความยาวของฟิลด์สามารถเปลี่ยนแปลงได้ตามฟังก์ชั่นที่ใช้ในฟิลด์ฟังก์ชั่น ในเฟรมร้องขอจากมาสเตอร์ ฟิลด์นี้อาจจะบรรจุชื่อข้อมูลที่มาสเตอร์ร้องขอ ส่วนในเฟรมตอบกลับของสแลฟจะบรรจุข้อมูลที่มาสเตอร์ต้องการหรือตอบกลับในรูปแบบยืนยัน

สองไบต์สุดท้ายของเฟรมจะประกอบกันเป็นฟิลด์ตรวจสอบข้อผิดพลาดค่าที่เป็นตัวเลขถูกคำนวณโดยใช้วิธีการ CRC-16 หรือ LRC การตรวจสอบข้อผิดพลาดนั้นจะทำให้แน่ใจว่าอุปกรณ์จะไม่ตอบสนองคำขอที่มีความผิดพลาด

ตารางที่ 2.2 Modbus Address และ Function Code

- **Main Modbus function code**

Function Code	Action	Table Name	For address
01 (01 hex)	Read	Discrete Output Coils	0x
05 (05 hex)	Write single	Discrete Output Coil	0x
15 (0F hex)	Write multiple	Discrete Output Coils	0x
02 (02 hex)	Read	Discrete Input Contacts	1x
04 (04 hex)	Read	Analog Input Registers	3x
03 (03 hex)	Read	Analog Output Holding Registers	4x
06 (06 hex)	Write single	Analog Output Holding Register	4x
16 (10 hex)	Write multiple	Analog Output Holding Registers	4x

- **Function code : R/W Modbus address table**

	address	Point type	Coil/Register Numbers	Data Addresses	Type	Table Name
DO	0x	01:coil status	1-9999	0000 to 270E	Read-Write	Discrete Output Coils
DI	1x	02:input status	10001-19999	0000 to 270E	Read-Only	Discrete Input Contacts
AI	3x	04:input register	30001-39999	0000 to 270E	Read-Only	Analog Input Registers
AO	4x	03:holding register	40001-49999	0000 to 270E	Read-Write	Analog Output Holding Registers

โดยทั่วไปค่าแอดเดรสข้อมูลที่ใช้ในเมสเสจจะเป็นค่าออฟเซตหรือรีเลทีฟ แต่ก็ไม่แน่เสมอไป ให้ตรวจสอบคู่มือการใช้งานของอุปกรณ์นั้นๆเป็นสำคัญ

2.4.8 หมายเลขฟังก์ชั่น (Function Codes)

แต่ละเฟรมร้องขอจะบรรจุหมายเลขฟังก์ชั่นที่นิยามลักษณะการทำงานที่ถูกต้องการให้สเลฟดำเนินการ ความหมายของฟิลด์ข้อมูลขึ้นอยู่กับหมายเลขหรือชนิดฟังก์ชั่นที่ใช้ การร้องขอในบางครั้งอาจเรียกอีกอย่างว่า คิวรี (Query) ซึ่งเป็นศัพท์ทางเทคนิคที่ใช้กันมากในระบบฐานข้อมูล

2.4.8.1 Read Coil หรือ Digital Output Status (Function Code 01)

หมายเลขฟังก์ชั่นนี้อนุญาตให้มาสเตอร์สามารถดึงสถานะ ON/OFF ของคอยล์ในตัวสเลฟซึ่งปกติแล้วจะใช้บอกสถานะการคอนโทรลของสเลฟว่าควบคุมอะไร อย่างไรในขณะนั้น ฟิลด์ข้อมูลของเฟรมร้องขอจะมีความสัมพันธ์กับหมายเลขแอดเดรสของคอยล์ตัวแรกตามด้วยจำนวนของคอยล์ที่ถูกต้องการ ฟิลด์ข้อมูลของเฟรมตอบสนองจะประกอบจำนวนคอยล์นับเป็นจำนวนไบต์ซึ่งอาจจะมีหลายไบต์ แต่ละไบต์จะแสดงค่าคอยล์ได้จำนวนแปดคอยล์ นั้นหมายความว่าคอยล์จะถูกจัดทีละ 8 คอยล์ให้เป็นหน่วยไบต์

โดยมีหมายเลขแอดเดรสที่เรียงต่อเนื่องกันตามลำดับบิต (1=ON, 0=OFF) บิตตำแหน่ง LSB (Least Significant Bit) จะแสดงถึงคอยล์ตัวแรกตามคำร้องขอ และถ้าจำนวนของคอยล์ที่ร้องขอไม่เป็นจำนวนเท่าของเลขแปด ข้อมูลในไบต์สุดท้ายที่เหลือจะถูกเติมด้วยค่าศูนย์จนเต็ม จำไว้ว่า ถ้ามีการร้องขอจำนวนคอยล์หลายไบต์ โลว์ออเดอร์บิต (Low Order Bit) ของไบต์แรกในเฟรมตอบสนองจะเป็นของคอยล์หมายเลขแอดเดรสแรกที่ร้องขอ

มาสเตอร์ได้ร้องของสถานะของคอยล์แอดเดรสที่ 20-56 จากสเลฟหมายเลข 17 สเลฟได้ตอบกลับดังเมสเสจตอบกลับข้างล่าง พิจารณาที่สถานะคอยล์ 27-20 โดยอยู่ในรูปเลขฐานสิบหก CD หรือฐานสอง 1100 1101 คอยล์ 27 จะอยู่ที่ตำแหน่ง MSB ส่วนคอยล์ 20 อยู่ที่ตำแหน่ง LSB ของไบต์ ดังนั้นจากซ้ายไปขวา คอยล์ 27 ถึงคอยล์ 20 จะมีค่า ON–ON–OFF–OFF–ON–ON–OFF–ON

2.4.8.2 Read Digital Input Status (Function Code 02)

ฟังก์ชั่นนี้ทำให้มาสเตอร์สามารถอ่านค่าอินพุตแบบดิสครีต (Discrete Input) หรือดิจิตอลอินพุตในอุปกรณ์สเลฟ ฟิลด์ข้อมูลของเฟรมร้องขอจะประกอบด้วยหมายเลขแอดเดรสของอินพุตแรกตามด้วยจำนวนดิจิตอลอินพุตที่ต้องการอ่าน ฟิลด์ข้อมูลของเฟรมตอบสนองจะประกอบจำนวนข้อมูลนับเป็นหน่วยไบต์เช่นกัน ซึ่งอาจจะมีจำนวนหลายไบต์ ข้อมูลที่ได้หน่วยย่อยจะเป็นบิตโดยการอ้างอิงค่าของดิจิตอลอินพุตที่ต้องการอ่านตามตำแหน่ง

ข้อมูลดิสครีตอินพุตจะถูกจัดในรูปแบบหนึ่งบิตเรียงกันไป (1-ON, 0=OFF) บิตตำแหน่ง LSB จะเป็นค่าแรกของดิสครีตอินพุตในไบต์แรก ถ้าจำนวนของดิสครีตอินพุตไม่ลงตัวเป็นหน่วยไบต์ บิตที่เหลือจะถูกเติมด้วยบิตศูนย์ให้เต็มไบต์สุดท้าย

มาสเตอร์จะทำการร้องขอค่าดิสครีตอินพุตจาก 10197 ถึง 10218 จากสเลฟหมายเลข 17 สเลฟได้ตอบกลับดังเมสเสจตอบกลับ พิจารณาที่สถานะอินพุต 10204–10197 โดยอยู่ในรูปเลขฐานสิบหก AC หรือฐานสอง 1010 1100 อินพุต 10204 จะ

อยู่ที่ตำแหน่ง MSB ส่วนอินพุต 10197 อยู่ที่ตำแหน่ง LSB ของไบต์ ดังนั้นจากซ้ายไปขวา อินพุต 10204 ถึงอินพุต 10197 จะมีค่า ON–OFF–ON–OFF–ON–ON–OFF–OFF

2.4.8.3 Read Holding Register (Function Code 03)

ฟังก์ชั่นนี้ทำให้มาสเตอร์สามารถดึงค่าในรีจิสเตอร์ของสเลฟได้ โดยทั่วไปคือค่าเซตติ้งหรือพารามิเตอร์ของอุปกรณ์นั้นๆฟิลด์ดาต้าของเฟรมร้องขอจะประกอบด้วยที่อยู่แอดเดรสที่อ้างอิงถึงตำแหน่งรีจิสเตอร์ตัวแรกตามด้วยจำนวนรีจิสเตอร์ที่ต้องการอ่าน ฟิลด์ดาต้าของเฟรมตอบสนองจะประกอบด้วย จำนวนไบต์ของข้อมูลของรีจิสเตอร์ที่ถูกอ่านตามด้วยค่าที่อ่านได้เป็นจำนวนหลาย ๆ ไบต์

เนื้อหาข้อมูลของแต่ละรีจิสเตอร์ที่ร้องขอ (ขนาด 16 บิต) จะถูกตอบกลับในรูปของสองไบต์ติดต่อกัน (โดยไบต์บนจะถูกส่งก่อน) มาสเตอร์ร้องขอค่าโฮลดิ้งรีจิสเตอร์ที่ 40108–40110 จากสเลฟหมายเลขที่ 17 สเลฟตอบกลับโดยรีจิสเตอร์ 40108 มีขนาดสองไบต์ 0x02 และ 0x2B หรือ 555 ส่วนรีจิสเตอร์ 40109-40110 มีค่าเท่ากับ 0 และ 100 ตามลำดับ

2.4.8.4 Reading Input Register (Function Code 4)

ฟังก์ชั่นนี้อนุญาตให้มาสเตอร์ สามารถอ่านค่าอินพุตรีจิสเตอร์จากหลาย ๆ รีจิสเตอร์ในอุปกรณ์สเลฟ โดยทั่วไปอินพุตรีจิสเตอร์จะมีไว้เก็บค่าวัดอะนาลอก

ฟิลด์ข้อมูลของเฟรมร้องขอจะประกอบด้วยแอดเดรสของอินพุตรีจิสเตอร์ตัวแรกตามด้วยจำนวนรีจิสเตอร์ที่ต้องการอ่านฟิลด์ข้อมูลของเฟรมตอบสนองจะประกอบด้วยจำนวนไบต์ของข้อมูลรีจิสเตอร์ที่ถูกอ่านประกอบด้วยกันหลายๆ ไบต์ที่เป็นค่าภายในรีจิสเตอร์ ข้อมูลในแต่ละรีจิสเตอร์จะถูกส่งในรูปแบบสองไบต์ติดต่อกัน (ไบต์บนจะถูกส่งก่อน) ช่วงค่าตำแหน่งของรีจิสเตอร์ที่เป็นไปได้จะอยู่ระหว่าง 0-4095

2.4.8.5 Force Single Coil (Function Code 5)

ฟังก์ชั่นนี้อนุญาตให้มาสเตอร์สามารถทำการเปลี่ยนสถานะของคอยล์ภายในตัวอุปกรณ์สเลฟฟิลด์ดาต้าของเฟรมร้องขอจะประกอบด้วยแอดเดรสของคอยล์และสถานะที่ต้องการจะเปลี่ยนสำหรับคอยล์นั้น ๆ ค่าฐานสิบหก 0xFF00 นั้นจะทำการแอกตีฟหรือจ่ายไฟให้คอยล์ ในขณะที่ค่า 0x0000 จะทำการดีแอกทีฟหรือยกเลิกจ่ายไฟคอยล์ ส่วนค่าอื่นๆนอกจากที่กล่าวจะไม่มีความหมายใดๆ

ถ้าตัวอุปกรณ์สเลฟสามารถกระทำการบนคอยล์ที่ถูกร้องขอได้ มันจะตอบกลับด้วยเฟรมที่เหมือนเฟรมร้องขอจากมาสเตอร์ทุกประการ มิฉะนั้นมันจะส่งเฟรมที่บ่งบอกข้อผิดพลาดไปแทน ตัวอย่างคอยล์แอดเดรส 173 ของสเลฟหมายเลข 17 ถูกแอกทีฟ ถ้าคำสั่งทำสำเร็จสเลฟจะตอบกลับเหมือนเดิมทุกประการ

2.4.8.6 Preset Single Register (Function Code 06)

ฟังก์ชั่นนี้ทำให้มาสเตอร์สามารถเปลี่ยนแปลงข้อมูลภายในของโฮลดิ้งรีจิสเตอร์ในตัวอุปกรณ์สเลฟ ฟิลด์ดาต้าของเฟรมร้องขอจะประกอบด้วยแอดเดรสของโฮลดิ้งรีจิสเตอร์ตามด้วยค่าใหม่ที่ต้องการเขียนลงบนตัวรีจิสเตอร์นั้น (เขียนไบต์บนก่อน)

ถ้าสเลฟสามารถเขียนค่าใหม่ลงบนรีจิสเตอร์ที่ต้องการได้ เฟรมตอบกลับจะเหมือนกับเฟรมร้องขอทุกประการ มิฉะนั้นการตอบสนองจะส่งเฟรมที่มีตัวบ่งบอกความผิดพลาดกลับมา

เมื่อสแลฟแอดเดรสที่ถูกระบุเป็น 00 จะหมายถึงการส่งคำสั่งแบบบรอดคาสต์ ดังนั้นทุกสแลฟจะโหลดค่ารีจิสเตอร์ด้วยค่าที่กำหนดพร้อมกันและไม่ทำการตอบกลับ

2.4.8.7 Read Exception Status (Function Code 7)

เฟรมขนาดสั้นๆ ที่ร้องขอสถานะภายในอุปกรณ์สเลฟจำนวนแปดค่า สถานะจำนวนแปดสถานะนั้นได้ถูกกำหนดความหมายเป็นที่เรียบร้อยโดยมาตรฐาน ในกรณีพิเศษผู้ใช้หรือผู้ผลิตสามารถกำหนดความหมายของสถานะได้ สำหรับตัวอย่างนี้ควรเป็นสถานะระบบ เช่น สถานะของแบตเตอรี่ หรือแม้กระทั่งหน่วยความจำถูกป้องกันหรือไม่ หรือสถานะของระบบออนไลน์หรือไม่

2.4.8.8 Force Multiple Coil หรือ Digital Output (Function Code 16)

ฟังก์ชั่นนี้จะบังคับกลุ่มของคอยล์ที่เรียงติดกันไปตามสถานะ ON หรือ OFF ตามที่มาสเตอร์ต้องการ ดังตัวอย่างต่อไปนี้มีการบังคับตั้งค่า 10 คอยล์ เริ่มที่คอยล์ 20 แอดเดรส 19 หรือ 0x13 (ที่สเลฟ 17) ให้มีสถานะ ดังค่า 0xCD01 โดยตำแหน่งบิตที่ไม่ใช้จะถูกเติมด้วยค่าศูนย์ ส่วนเมสเสจที่ตอบกลับนั้นจะตอบกลับคล้ายกับเมสเสจร้องขอแต่ไม่มีส่วนของจำนวนไบต์ของข้อมูลตำแหน่งบิต ดังรูปที่ 2.33

ตำแหน่งบิต: 1 1 0 0 1 1 0 1 0 0 0 0 0 0 0 1

คอยล์ : 27 26 25 24 23 22 21 20 – – – – – – 29 28

ถ้าหมายเลขสเลฟเท่ากับศูนย์ระบุในเฟรมร้องขอโหมดบรอดคาสต์จะทำงานส่งผลให้สเลฟทุกตัวเปลี่ยนสถานะของชุดคอยล์ตามที่กำหนด

2.4.8.9 Force Multiple Register (Function Code 10)

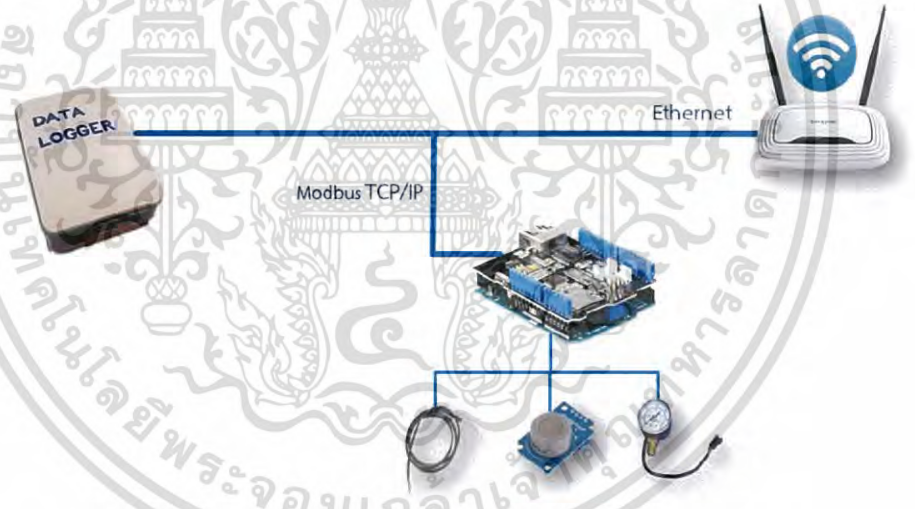
ฟังก์ชั่นนี้มีความคล้ายคลึงกับคำสั่งพรีเซตค่าบนรีจิสเตอร์ตัวเดียวและการบังคับหรือเปลี่ยนสถานะของคอยล์จำนวนหลายคอยล์ อุปกรณ์สเลฟแอดเดรส 17 มี 2 รีจิสเตอร์เริ่มที่หมายเลข 40002 โดยตำแหน่งแรกที่แอดเดรส 01 โดยจะเปลี่ยนค่าภายใน

รีจิสเตอร์ให้เป็นค่า 0x000A และ 0x0102 ตามลำดับ ส่วนเมสเสจที่ตอบกลับนั้นจะตอบกลับคล้ายกับเมสเสจร้องขอแต่ไม่มีส่วนของจำนวนไบต์และข้อมูลที่จะเปลี่ยน

บทที่ 3
วิธีการดำเนินงาน

3.1 ภาพรวมขององค์ประกอบหลักของระบบ

กระบวนการประยุกต์ใช้งานอุปกรณ์บันทึกข้อมูลหรือ data logger พัฒนาอุปกรณ์โดยใช้บอร์ดคอมพิวเตอร์ขนาดเล็ก Raspberry Pi ใช้งานร่วมกับโปรแกรม Codesys และบันทึกผลลงการ์ดหน่วยความจำถาวร (SD Card) โดยโครงงานมีวัตถุประสงค์เพื่อเป็นการศึกษาการบันทึกข้อมูลทางด้านอุตสาหกรรม เช่น อุณหภูมิ ความดัน การไหล ระดับ เป็นต้นดังนั้นเพื่อให้การทำงานมีประสิทธิภาพเราจึงจำเป็นต้องศึกษาองค์ประกอบต่างๆ ซึ่งในบทนี้ผู้วิจัยจะกล่าวถึงองค์ประกอบทั้งหมดที่เกี่ยวข้องกับกระบวนการการออกแบบการทำงานของระบบ และการแสดงผลการทำงานของอุปกรณ์ผ่านเครือข่ายอินเตอร์เน็ต ซึ่งภาพรวมการทำงานแสดงดังรูปที่ 3.1



รูปที่ 3.1 ภาพรวมระบบการทำงานของอุปกรณ์ Data Logger

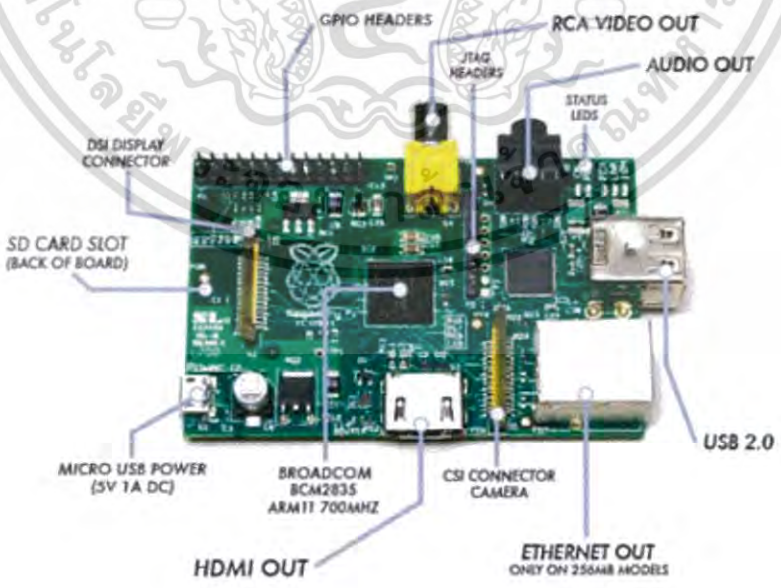
3.2 การประยุกต์ใช้งานฮาร์ดแวร์และซอฟต์แวร์

ในหัวข้อนี้ จะกล่าวถึงองค์ประกอบทั้งหมดที่เกี่ยวข้องกับกระบวนการสร้าง Data Logger สำหรับอุปกรณ์ Modbus ประกอบด้วยฮาร์ดแวร์และซอฟต์แวร์โดยในส่วนฮาร์ดแวร์ได้อธิบายถึงอุปกรณ์ไมโครคอนโทรลเลอร์ Raspberry Pi อย่างละเอียด และในส่วนซอฟต์แวร์จะมีการอธิบายซอฟต์แวร์ที่เกี่ยวข้องทั้งหมดที่เกี่ยวข้องกับการวิจัย ซึ่งรายละเอียดต่างๆจะอธิบายในแต่ละหัวข้อดังต่อนี้

3.2.1 องค์ประกอบของฮาร์ดแวร์

3.2.1.1 ไมโครคอนโทรลเลอร์ Raspberry Pi

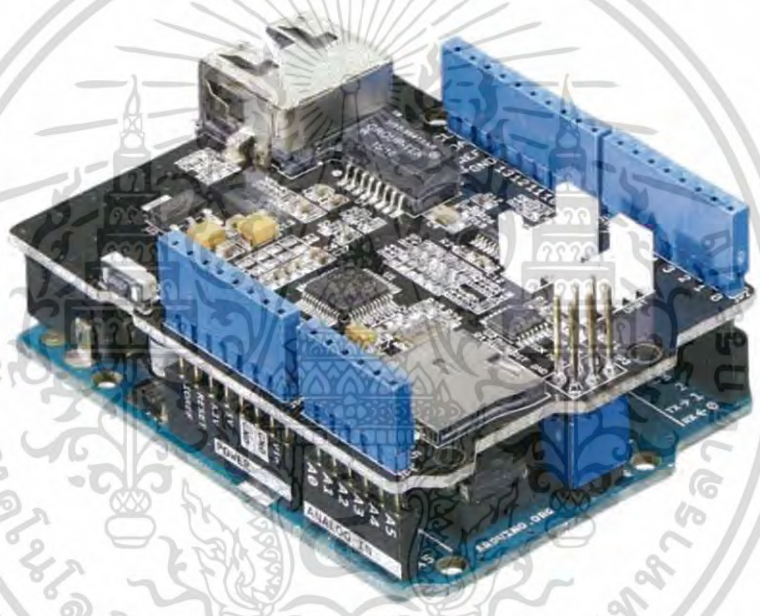
RaspberryPi คือ บอร์ดคอมพิวเตอร์ขนาดเล็กที่สามารถเชื่อมต่อกับจอมอนิเตอร์,คีย์บอร์ดและเมาส์ได้สามารถนำมาประยุกต์ใช้ในการทำโครงงานทางด้านอิเล็ก-ทรอนิกส์ การเขียนโปรแกรมหรือเป็นเครื่องคอมพิวเตอร์ตั้งโต๊ะขนาดเล็กไม่ว่าจะเป็นการทำงาน Spreadsheet Word Processing ท่องอินเทอร์เน็ต ส่งอีเมล หรือเล่นเกมส์ อีกทั้งยังสามารถเล่นไฟล์วีดีโอความละเอียดสูง(High-Definition)Hได้อีกด้วยดังรูปที่ 3.2 ด้วยเหตุนี้ผู้วิจัยจึงได้เลือกใช้บอร์ดRaspberry Pi ร่วมกับโปรแกรม Codesys ในการพัฒนาเพื่อสร้างอุปกรณ์บันทึกข้อมูล หรือ Data Logger โดยนำข้อมูลที่เก็บได้บันทึกลงใน SD Card ของบอร์ด Raspberry Pi และสามารถนำมาข้อมูลมาใช้งานในและแสดงผลในรูปแบบต่างๆได้



รูปที่ 3.2 ไมโครคอนโทรลเลอร์ Raspberry Pi

3.2.1.2 บอร์ด Arduino Uno r3 + Ethernet Shield W5500

Arduino Uno r3 เป็นบอร์ดไมโครคอนโทรเลอร์ตระกูล AVR ที่มีการพัฒนาแบบ Open Source คือมีการเปิดเผยข้อมูลทั้งด้าน ฮาร์ดแวร์ และ ซอร์ฟแวร์ สำหรับตัวบอร์ด Arduino ถูกออกแบบมาให้ใช้งานได้ง่ายด้วยสาเหตุนี้ผู้วิจัยจึงเลือกใช้งาน และบอร์ดยังสามารถพัฒนาต่อยอดทั้งตัวบอร์ดและโปรแกรมได้ง่ายโดยสามารถต่อวงจรอิเล็กทรอนิคส์จากภายนอกแล้วเชื่อมต่อเข้ามาที่ขา I/O ของบอร์ดได้เลย นอกจากนี้ยังสามารถเลือกต่อกับบอร์ดเสริม (Arduino Shield) ประเภทต่างๆได้ง่ายโดยผู้วิจัยได้เลือกใช้ Ethernet Shield W5500 ดังรูปที่ 3.3 เพื่อให้สามารถใช้ในการจำลองการสื่อสารระหว่างเซ็นเซอร์กับ Raspberry Pi ให้เป็นแบบ Modbus ได้



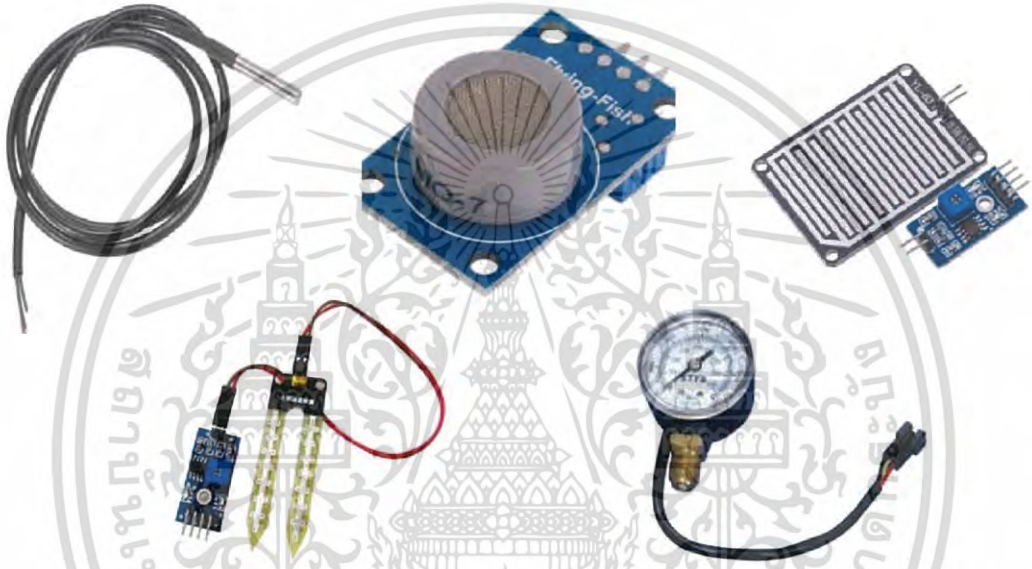
รูปที่ 3.3 บอร์ด Arduino Uno r3 + Ethernet Shield W5500

คุณสมบัติของ บอร์ด Arduino Uno r3

1) ง่ายต่อการพัฒนา มีรูปแบบคำสั่งพื้นฐาน ไม่ซับซ้อนเหมาะสำหรับผู้เริ่มต้นมี Arduino Community กลุ่มคนที่ร่วมกันพัฒนาที่แข็งแรง
2) Open Hardware ทำให้ผู้ใช้สามารถนำบอร์ดไปต่อยอดใช้งานได้หลายด้าน,ราคาไม่แพง
3) Cross Platform สามารถพัฒนาโปรแกรมบน OS ใดก็ได้

3.2.1.3 อุปกรณ์ตรวจจับสัญญาณชนิดต่างๆ

เนื่องจากในการทดสอบการทำงานของอุปกรณ์ Data Logger For Modbus ไม่สามารถนำอุปกรณ์ไปเชื่อมต่อกับระบบการทำงานจริงในอุตสาหกรรมได้ตลอดการทดลอง จึงได้เลือกใช้เซ็นเซอร์ในการเก็บพารามิเตอร์ดังนี้ Thermocouple, Carbon monoxide, Raindrop, Flow และ Humidity Sensor ดังรูปที่ 3.4 ซึ่งจะส่งข้อมูลผ่านบอร์ด Arduino Uno r3 โดยใช้โปรโตคอลแบบ Modbus



รูปที่ 3.4 อุปกรณ์ตรวจจับสัญญาณชนิดต่างๆ

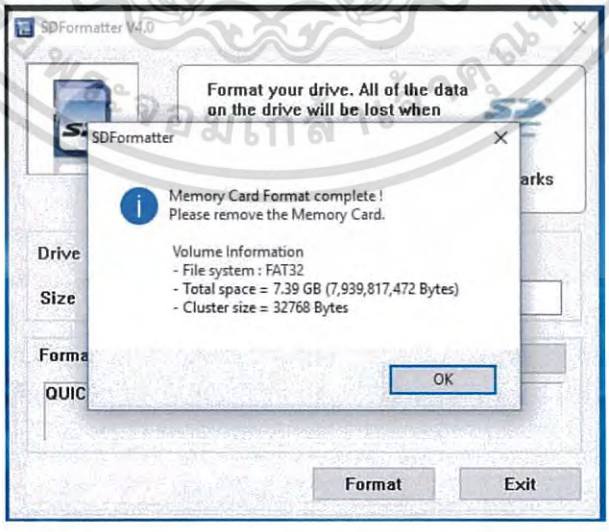
3.2.2 องค์ประกอบของซอฟต์แวร์

3.2.2.1 ระบบปฏิบัติการ NOOBS

สามารถดาวน์โหลดระบบปฏิบัติการ NOOBS หรือ Raspbian จากเว็บในหน้าเว็บนี้ https://www.raspberrypi.org โดยถ้าดาวน์โหลด NOOBS จะมี Raspbian ติดมาให้ไม่ต้องดาวน์โหลดทีหลัง แต่ NOOBS Lite ทุก OS ต้องโหลดผ่านอินเทอร์เน็ต โดยสามารถติดตั้งลงใน SD Card ของบอร์ด Raspberry Pi เมื่อดาวน์โหลดจะได้ไฟล์ .zip ดังรูป ดังรูปที่ 3.5

รูปที่ 3.5 หน้าเว็บดาวน์โหลดระบบปฏิบัติการ NOOBS

เตรียม SD Card ดาวน์โหลดโปรแกรม SD Formatter มาติดตั้งในคอมพิวเตอร์ แล้วทำการ Format SD Card โดยใช้ SD Formatter ก่อน

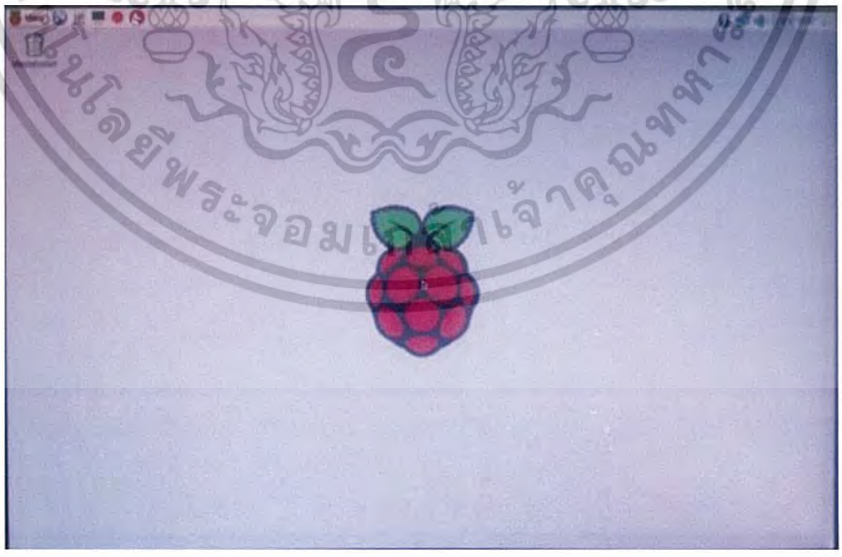


รูปที่ 3.6 การ Formatter SD card โดยใช้โปรแกรม SD Formatter

ติดตั้งระบบปฏิบัติการ NOOB บน SD Card โดยไฟล์ NOOBS ที่ได้จะเป็นไฟล์ .zip สามารถแตกไฟล์ .zip ออกจะได้ไฟล์ต่างๆดังรูปที่ 3.7

รูปที่ 3.7 การติดตั้ง NOOB บน SD Card

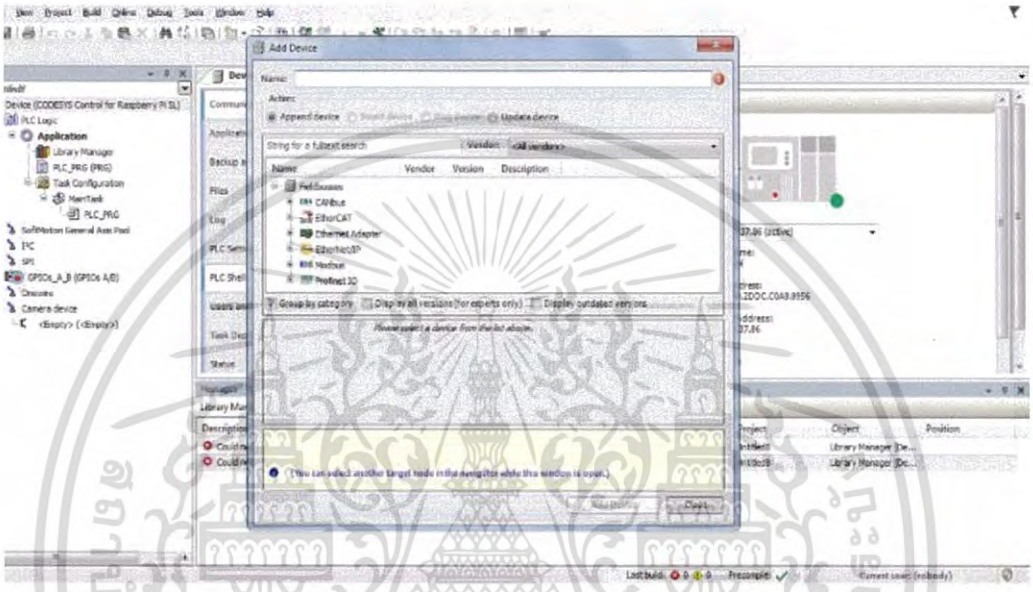
นำ SD card ที่ลง NOOBS แล้วใส่ไปในช่องเสียบ SD Card ของ Raspberry Pi ต่อพอร์ตต่างๆ คือ Ethernet , เมาส์, คีย์บอร์ด, HDMI ต่อกับจอมอนิเตอร์ จากนั้นจึงต่อกับ Power Supply



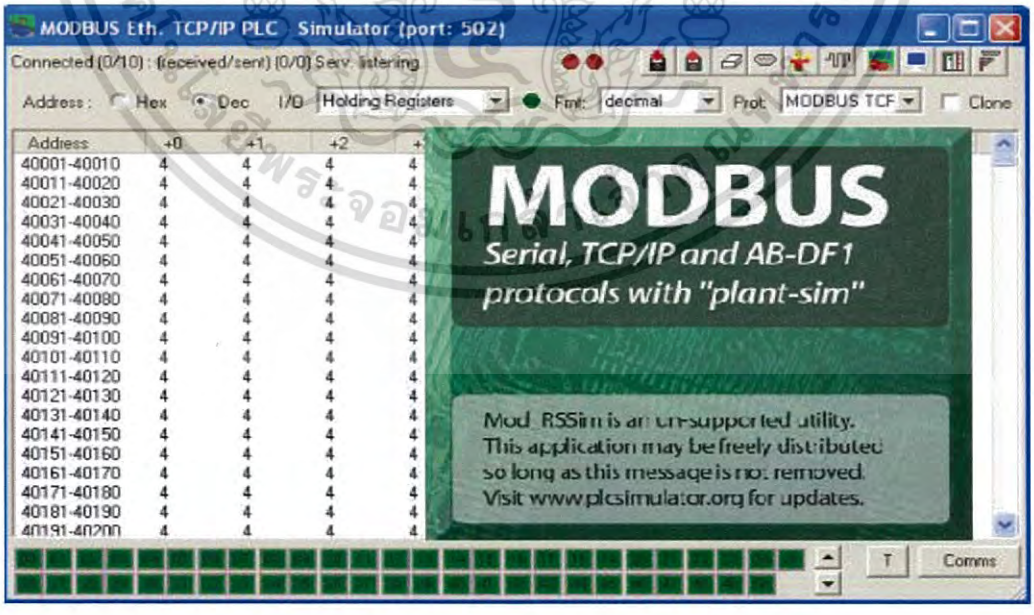
รูปที่ 3.8 เมื่อรีบูทขึ้นมาใหม่แล้ว ก็จะเข้าสู่หน้า Desktop ของ Raspbian เป็นอันเสร็จ

3.2.2.2 โปรแกรม Codesys และ โปรแกรม Modbus_Rssim

ใช้ในการเขียนควบคุมการทำงานและการเชื่อมต่ออุปกรณ์ให้สื่อสารกันได้ในลักษณะ มาสเตอร์ / สเลฟ และแสดงผลข้อมูล โดยให้อุปกรณ์ raspberry pi 3 เป็น มาสเตอร์ และโน้ตบุ๊กเป็น สเลฟ (โดยในเบื้องต้นใช้โปรแกรม Mod_Rssim จำลองเป็นเซ็นเซอร์) ร่วมกับโปรแกรม Codesys ดังรูปที่ 3.9 และรูปที่ 3.10 โดยมีวิธีการใช้งานดังนี้



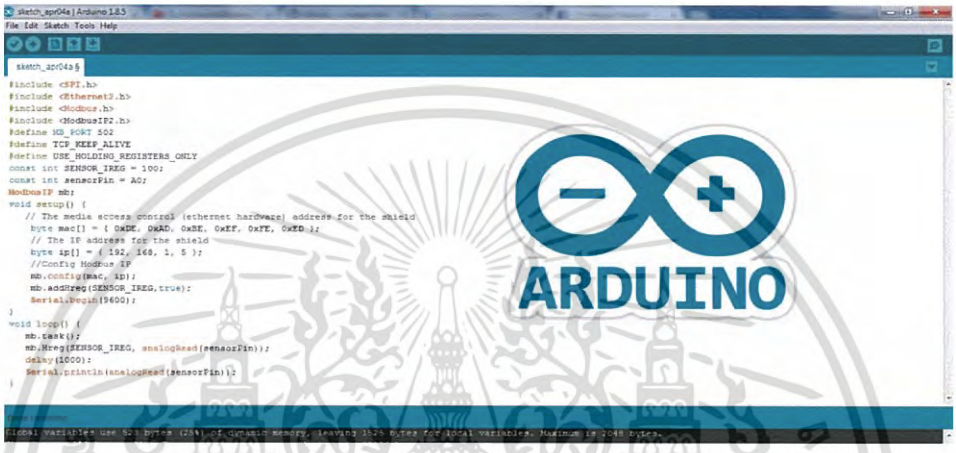
รูปที่ 3.9 แสดงการใช้งานโปรแกรม Codesys



รูปที่ 3.10 แสดงการใช้งานโปรแกรม Mod_Rssim

3.2.2.3 โปรแกรม Arduino IDE

โปรแกรม Arduino ได้พัฒนาส่วนของระบบการเขียนโปรแกรมหรือที่เรียกว่า IDE (Integrated Development Environment) ใช้ในการจำลองเป็นอุปกรณ์สเลฟ ส่งข้อมูลจาก Sensor ชนิดต่างๆซึ่งในส่วนของ software สามารถดาวน์โหลด library เพิ่มได้จาก อินเตอร์เน็ต ซึ่งทำให้สามารถเขียนโปรแกรมกับคอนโทรลเลอร์ชนิดอื่นๆได้ ดังรูปที่ 3.11



รูปที่ 3.11 แสดงการใช้งาน Arduino

3.2.2.4 เว็บ Anyviz Cloud

เว็บฐานข้อมูล Anyviz ซึ่งเป็นระบบ Cloud ฟรีสำหรับผู้ที่ต้องการทดลองใช้ โดยสามารถเรียกดูข้อมูลย้อนหลังได้และยังสามารถดูข้อมูลแบบ กราฟหรือตารางได้เพียงแค่สมัครเข้าใช้งาน ดังรูปที่ 3.12

รูปที่ 3.12 หน้าเว็บ Anyviz Cloud

3.3 การออกแบบและการดำเนินงาน

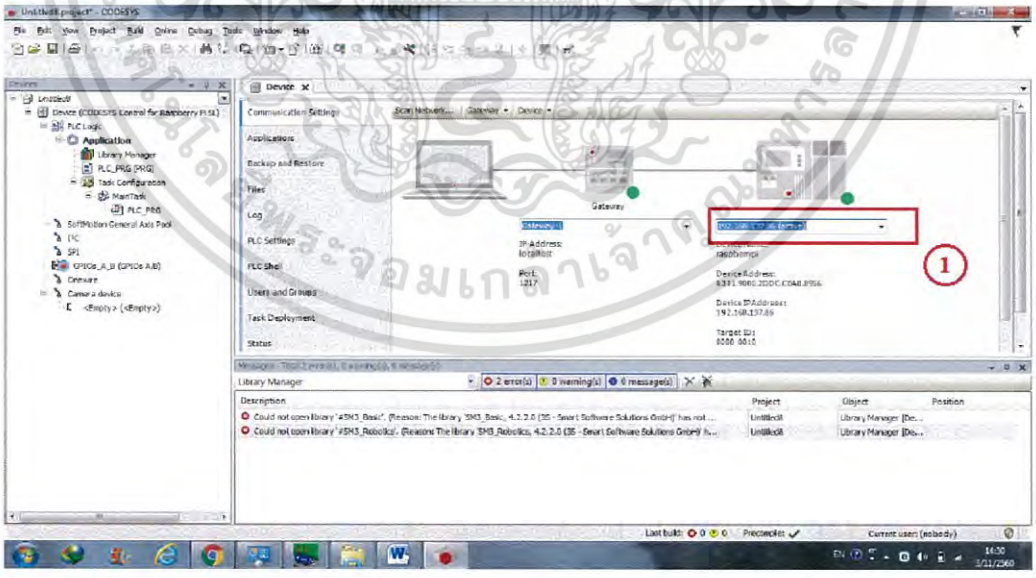
การสร้างอุปกรณ์ Data logger สำหรับโปรโตคอล Modbus ออกแบบโดยพัฒนาบอร์ด Raspberry Pi 3 ให้สามารถบันทึกข้อมูลลง SD Card ได้และพัฒนาตัวเก็บข้อมูลสำหรับอุปกรณ์ Modbus เพื่อใช้ในการเก็บข้อมูลตัวแปรพื้นฐานทางด้านอุตสาหกรรม เช่น อุณหภูมิ ความดัน การไหล ระดับ เป็นต้น โดยทำการควบคุมด้วยโปรแกรม Codesys เพราะฉะนั้นจึงได้มีการออกแบบลักษณะการทำงานและการรูปแบบการเชื่อมต่อ ดังนี้

3.3.1 การพัฒนาบอร์ด Raspberry pi 3 ให้เป็นอุปกรณ์ Data logger for Modbus โดยเขียนโปรแกรมควบคุมการทำงานด้วยโปรแกรม Codesys

การออกแบบโปรแกรม Codesys สำหรับการควบคุมการทำงานของอุปกรณ์ Data logger มีขั้นตอนในการออกแบบและเขียนโค้ดดังนี้

3.3.1.1 การเชื่อมต่อหว่างบอร์ด Raspberry Pi 3 กับ โปรแกรม Codesys

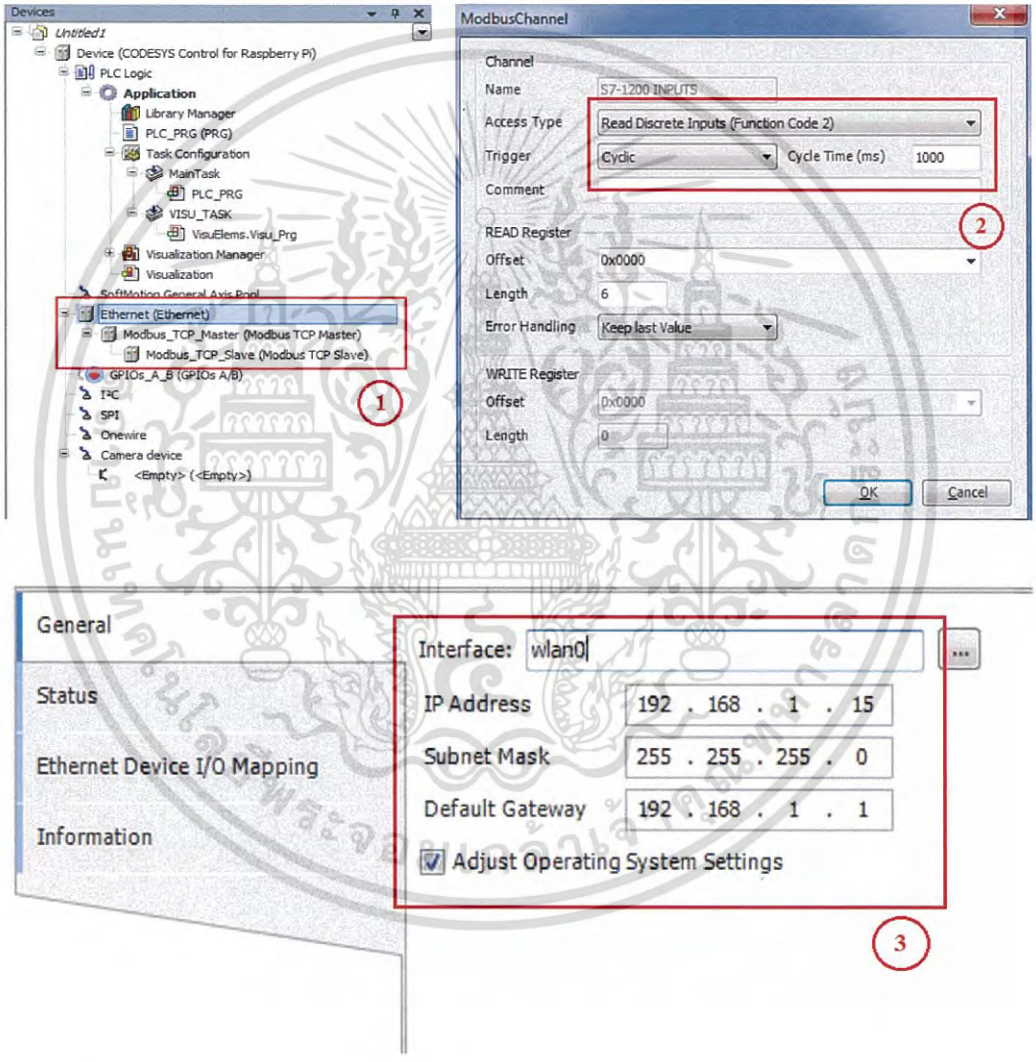
การเชื่อมต่อโดยใช้ Ethernet โดยสามารถเลือกใช้งานฟังก์ชั่นที่สามารถเขียนควบคุมบอร์ด Raspberry Pi ได้คือกำหนด Device เป็น (CODESYS Control for Raspberry pi 3) และต่อผ่าน Wi-Fi หรือ สายแลนโดยใส่ IP ของบอร์ดใน Device (ตามรูป 3.12 ที่ช่องหมายเลข 1) ของโปรแกรม Codesys



รูปที่ 3.13 แสดงการเชื่อมต่อหว่างบอร์ด Raspberry Pi 3 กับ โปรแกรม Codesys

3.3.1.2 การตั้งค่าการเชื่อมต่อ Codesys กับ Raspberry Pi และการกำหนดค่า Modbus

ในการตั้งค่าจะต้องป้อนที่อยู่ IP ของ Modbus TCP สเลฟ สามารถตั้งค่าดังรูป 3.13 โดยไปที่ Modbus_TCP_สเลฟ device (หมายเลข 1) และป้อน สเลฟ IP Address ซึ่งเป็น IP Address(หมายเลข 3) ของอุปกรณ์ที่เราต้องการจะเก็บข้อมูลและต้องกำหนดค่าที่อยู่ที่ต้อง read/write (หมายเลข 2) ไปยัง Modbus สเลฟ ไปที่ "Modbus สเลฟ Channel" แล้วเลือก Add Channel กำหนดค่าช่องแรกตามที่ปรากฏในรูปที่ 3.14



รูปที่ 3.14 แสดงการการตั้งค่าการเชื่อมต่อ Codesys กับ Raspberry Pi และการกำหนดค่า Modbus

3.3.1.3 การเขียนโค้ดควบคุมการทำงานของ Data logger ด้วย โปรแกรม Codesys

เนื่องจากในการสร้างมีข้อกำหนดของอุปกรณ์คือสามารถรับอินพุทได้ถึง 8 Channel และสามารถบันทึกข้อมูลได้แบบเรียลไทม์ได้ สามารถแสดง วัน เวลา รวมถึงข้อมูลต่างๆที่ต้องการบันทึกได้อย่างถูกต้อง นอกจากนี้ยังสามารถแสดงผลข้อมูลในรูปแบบ ตาราง และกราฟ ผู้วิจัยจึงได้ออกแบบลักษณะการทำงานและการควบคุม ดังนี้

1. การเขียนโค้ดสำหรับการรับและส่งข้อมูลโดยใช้โปรโตคอล Modbus ระหว่าง บอร์ด Arduino Uno r3 กับบอร์ด Raspberry pi ควบคุมโปรแกรม Codesys โดยภาษาที่เขียนมี 2 ภาษาคือ Structure text (ST) และ Ladder(LD) โดยใช้รูปแบบฟังก์ชั่นและ library ที่มีในโปรแกรม Codesys
2. ออกแบบการเขียนโค้ดควบคุมการทำงาน ของอุปกรณ์ Data Logger For Modbus โดยจะต้องทำให้อุปกรณ์ สามารถเก็บบันทึกข้อมูลได้ แสดงข้อมูลในรูปแบบต่างๆได้อย่างถูกต้องและสามารถตั้งค่าการควบคุมได้ตามฟังก์ชั่นที่ต้องการ
3. ทดลองบันทึกผลการสื่อสารแบบ Modbus 2 รูปแบบคือ ใช้โปรแกรม Modbus_RSsimและรับข้อมูลจากเซ็นเซอร์จริงโดยผ่านบอร์ด Arduino Uno r3 ต่อใช้งานร่วมกับ Ethernet Shield W5500

ในการเขียนโค้ดสำหรับการรับและส่งข้อมูลขั้นตอนแรกจะต้องดาวน์โหลด library ที่ทำให้โปรแกรมสามารถใช้งานได้ เช่น การรับ-ส่งไฟล์ การแสดงวัน-เวลา เป็นต้น โดยสามารถโหลดได้จาก Codesys store ซึ่ง libraryที่นำมาใช้ในการเขียนโค้ดมี ดังรูป 3.15

Name	Namespace	Effective version
3SLicense = 3SLicense, 3.5.12.0 (3S - Smart Software Solutions GmbH)	_3S_LICENSE	3.5.12.0
AlarmManager = AlarmManager, 3.5.12.0 (Intern)	AlarmManager	3.5.12.0
AnyViz Cloud Adapter, 1.4.1.0 (Mirasoft GmbH & Co. KG)	**AnyViz**	**1.4.1.0**
BreakpointLogging = Breakpoint Logging Functions, 3.5.5.0 (3S - Smart Software Solutions GmbH)	BPLog	3.5.5.0
CAA File = CAA File, 3.5.12.0 (CAA Technical Workgroup)	**FILE**	**3.5.12.0**
CAA Types = CAA Types Extern, 3.5.10.0 (CAA Technical Workgroup)	**CAA**	**3.5.10.0**
CmpErrors = CmpErrors, 3.3.1.40 (System)	**CmpErrors**	**3.3.1.40**
CmpTraceMgr = CmpTraceMgr, 3.5.11.0 (System)	CmpTraceMgr	3.5.11.0
Collections, 3.5.11.0 (System)	**Collections**	**3.5.11.0**
Element Collections, 3.5.12.0 (3S - Smart Software Solutions GmbH)	**COL**	**3.5.12.0**
File Access, 3.5.7.0 (3S - Smart Software Solutions GmbH)	**File_Access**	**3.5.7.0**
IecVarAccess = IecVarAccess, 3.5.11.0 (System)	IecVarAccessLibrary	3.5.11.0
IoDrvEthernet = IoDrvEthernet, 3.5.11.0 (3S - Smart Software Solutions GmbH)	IoDrvEthernet	3.5.11.0
IoDrvGPIO, 3.5.11.0 (3S - Smart Software Solutions GmbH)	IoDrvGPIO	3.5.11.0
IoDrvModbusTCP = IoDrvModbusTCP, 3.5.12.0 (3S - Smart Software Solutions GmbH)	IoDrvModbusTCP	3.5.12.0
IoStandard = IoStandard, 3.5.10.0 (System)	IoStandard	3.5.10.0
SM3_Basic = SM3_Basic, 4.3.0.0 (3S - Smart Software Solutions GmbH)	SM3_Basic	4.3.0.0
SM3_CNC = SM3_CNC, 4.3.1.0 (3S - Smart Software Solutions GmbH)	SM3_CNC	4.3.1.0
SM3_Robotics = SM3_Robotics, 4.3.0.0 (3S - Smart Software Solutions GmbH)	SM3_Robotics	4.3.0.0
SM3_Robotics_Visu = SM3_Robotics_Visu, 4.3.0.0 (3S - Smart Software Solutions GmbH)	SM3_Robotics_Visu	4.3.0.0
SM3_Transformation = SM3_Transformation, 4.3.1.0 (3S - Smart Software Solutions GmbH)	TRAFO	4.3.1.0
Standard = Standard, 3.5.12.0 (System)	**Standard**	**3.5.12.0**
SysDir = SysDir, 3.5.12.0 (System)	**SysDir**	**3.5.12.0**

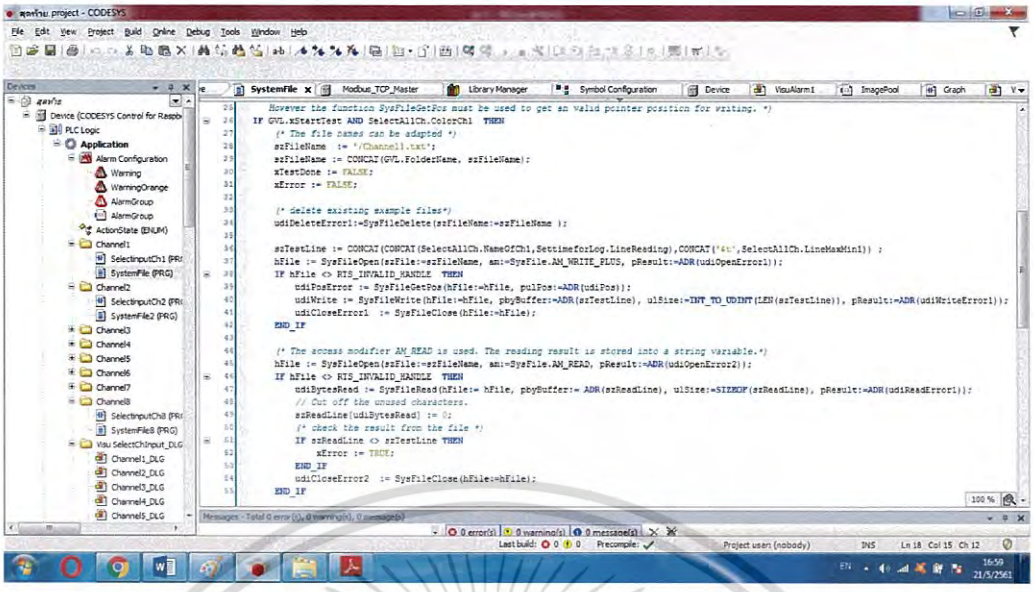
(ก)

Name	Namespace	Effective version
System_VisuElemMeter = VisuElemMeter, 3.5.12.0 (System)	VisuElemMeter	3.5.12.0
System_VisuElems = VisuElems, 3.5.12.10 (System)	VisuElems	3.5.12.10
System_VisuElemsAlarm = VisuElemsAlarm, 3.5.12.0 (System)	VisuElemsAlarm	3.5.12.0
System_VisuElemsDateTime = VisuElemsDateTime, 3.5.11.0 (System)	VisuElemsDateTime	3.5.11.0
System_VisuElemsSpecialControls = VisuElemsSpecialControls, 3.5.11.0 (System)	VisuElemsSpecialControls	3.5.11.0
System_VisuElemsWinControls = VisuElemsWinControls, 3.5.12.10 (System)	VisuElemsWinControls	3.5.12.10
System_VisuElemTextEditor = VisuElemTextEditor, 3.5.11.0 (System)	VisuElemTextEditor	3.5.11.0
System_VisuElemTrace = VisuElemTrace, 3.5.12.0 (System)	VisuElemTrace	3.5.12.0
System_VisuElemXYChart = VisuElemXYChart, 3.5.12.0 (System)	VisuElemXYChart	3.5.12.0
system_visuinputs = visuinputs, 3.5.12.0 (system)	visuinputs	3.5.12.0
System_VisuNativeControl = VisuNativeControl, 3.5.12.0 (System)	VisuNativeControl	3.5.12.0
SysTime, 3.5.9.0 (System)	**SysTime**	**3.5.9.0**
SysTypes Interfaces, * (System)	**SysTypes**	**3.5.2.0**
Time and Date, 3.5.7.0 (3S - Smart Software Solutions GmbH)	**Time_and_Date**	**3.5.7.0**
Unit Conversion Interfaces, * (Intern)	**UC**	**3.5.4.0**
Util = Util, 3.5.11.0 (System)	**Util**	**3.5.11.0**
Visu Interfaces, 3.5.12.0 (System)	**Visu_Interfaces**	**3.5.12.0**
VisuDialogs = VisuDialogs, 3.5.12.0 (System)	**VisuDialogs**	**3.5.12.0**
VisuElemBase, 3.5.12.10 (System)	**VisuElemBase**	**3.5.12.10**
VisuTrendStorageAccess = VisuTrendStorageAccess, 3.5.12.10 (System)	VisuTrendStorageAccess	3.5.12.10
VisuUserManagement = VisuUserMgmt, 3.5.12.0 (System)	VisuUserManagement	3.5.12.0
VisuUserMgmt, 3.5.12.0 (System)	**VisuUserManagement**	**3.5.12.0**

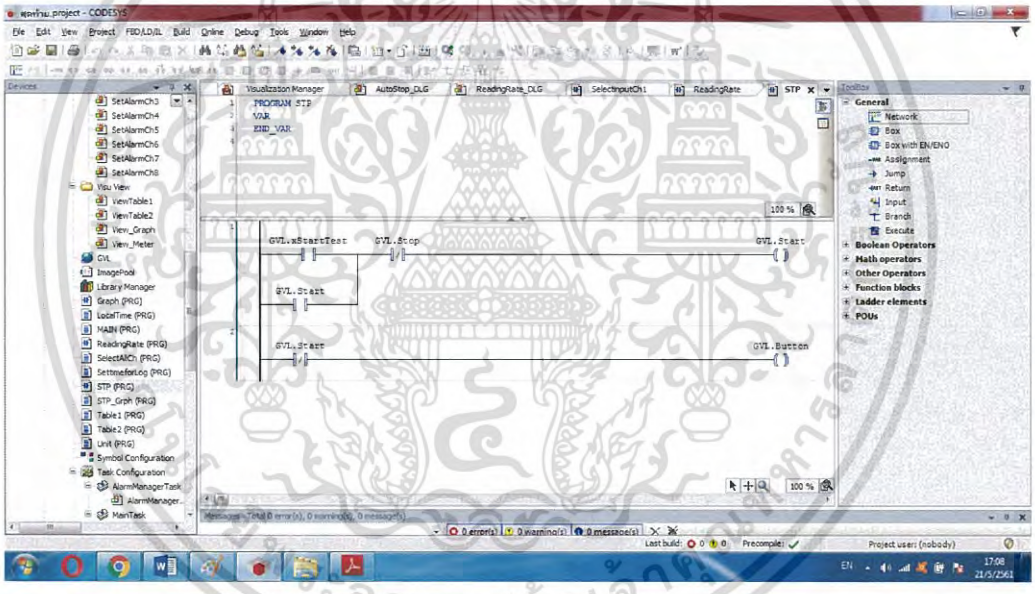
(ข)

รูปที่ 3.15 (ก) และ (ข) คือ library ที่ใช้ในการเขียนโปรแกรม

ภาษาที่ใช้เขียนมี 2 แบบคือ Structure text (ST) และ Ladder (LD) โดย Structure text ใช้ในการเขียนโค้ดในการสร้างและควบคุมให้อุปกรณ์ทำงานตามฟังก์ชั่นที่ต้องการ ในส่วน Ladder ใช้ในการออกแบบกราฟฟิค เช่น ปุ่มกด แสดงไฟ เป็นต้น ดังตัวอย่างที่แสดงดังรูปที่ 3.16 (สามารถศึกษาโค้ดโปรแกรมทั้งหมดเพิ่มเติมได้ที่ภาคผนวก)



(ก)



(ข)

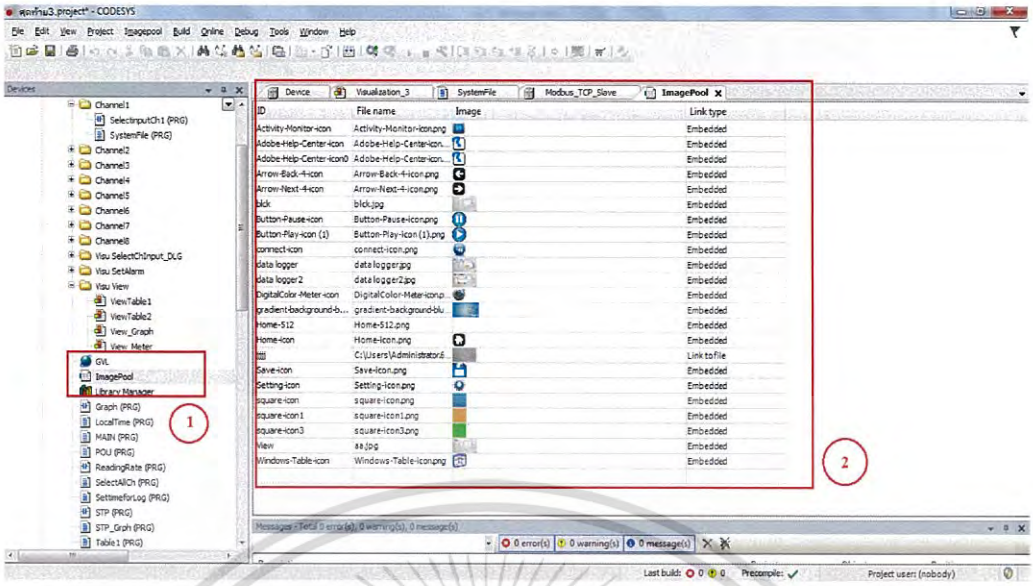
รูปที่ 3.16 ตัวอย่างการเขียนโค้ดโปรแกรม

(ก) การเขียนโค้ดโดยใช้ภาษา Structure text (ST)

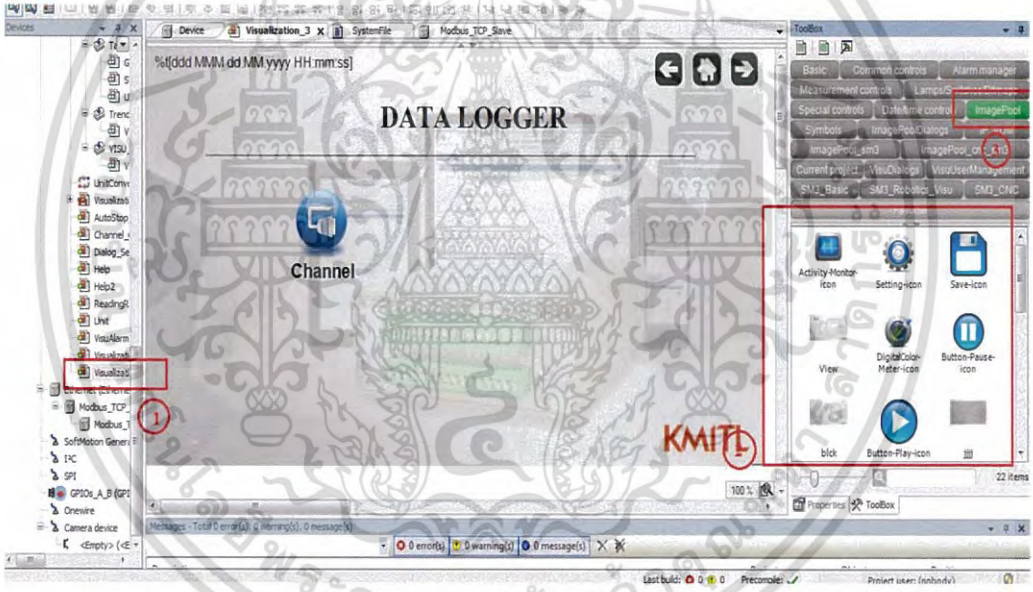
(ข) การเขียนโค้ดโดยใช้ภาษา Ladder (LD)

3.3.1.4 การออกแบบหน้าจอแสดงผล

การออกแบบหน้าจอการแสดงผลของอุปกรณ์สามารถใส่รูปภาพที่ต้องการในโปรแกรม Codesys โดยใช้ฟังก์ชั่น Image pool (หมายเลข 1) โดยสามารถออกแบบโดยเลือกรูปที่ต้องการใส่ในช่องหมายเลข2 ได้ดังรูป 3.17 (ก)

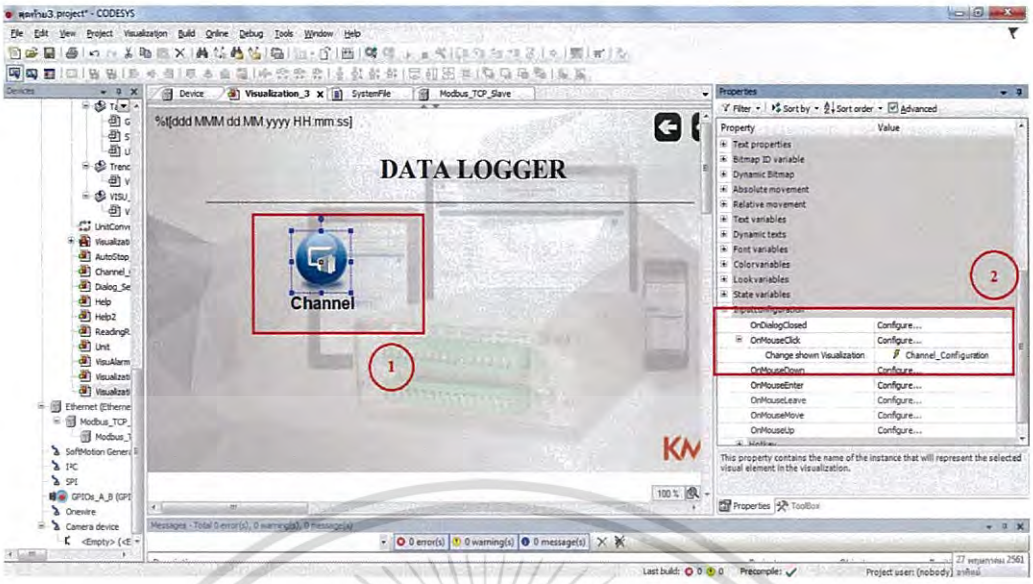


(ก)



(ข)

จากนั้นเลือก Visualization และเลือกรูปแบบที่ต้องการได้จากคำสั่ง Toolbox ดังรูป 3.17 (ข) เมื่อต้องการต้องการตั้งค่าให้ปุ่มกดใช้งานตามต้องการโดยสามารถเลือกใช้คำสั่ง Properties ดังรูป 3.17 (ค)



(ค)

รูปที่ 3.17 (ก),(ข) และ (ค) แสดงตัวอย่างการใช้งาน Visualization

การแสดงผลทำได้โดยสามารถนำคอมพิวเตอร์หรือโทรศัพท์มาเป็นอุปกรณ์ที่ใช้ในการติดต่อระหว่างผู้ใช้งานกับอุปกรณ์ Data logger for Modbus เพื่อควบคุมและเป็นจอแสดงผล ดังนั้นในขั้นตอนแรกจะต้องมีการกำหนดการนำเสนอก่อน โดย HMI ที่ออกแบบมีแนวความคิด ดังนี้

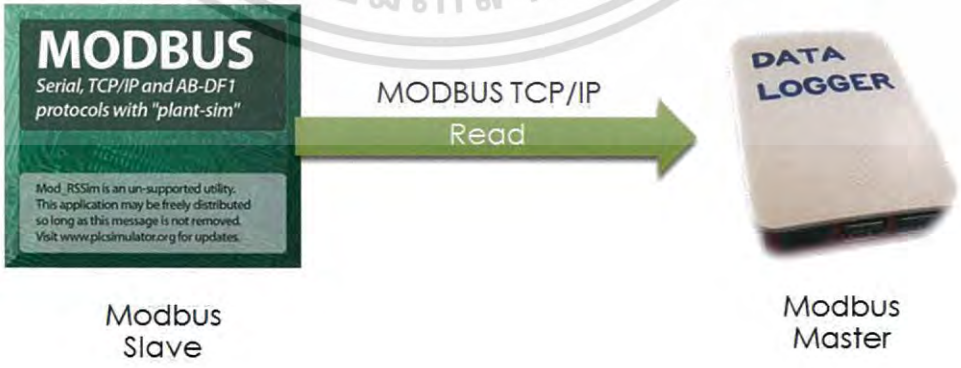
1. สามารถใช้งานง่าย มีคำอธิบายและรูปในทุกฟังก์ชั่น
2. สามารถแสดงผลข้อมูลแบบ เรียลไทม์ สามารถแสดง วัน-เดือน-ปี และเวลารวมถึง ชื่อพารามิเตอร์ที่ต้องการวัดได้อย่างถูกต้องและแม่นยำ
3. สามารถแสดงผลข้อมูลได้หลากหลายในรูปแบบกราฟ ตาราง และเปรียบเทียบข้อมูลพารามิเตอร์ต่างๆที่บันทึกจากหลายๆ Channel ได้
4. มีระบบจัดการให้ผู้ใช้งานได้แก้ไขข้อมูล
5. มีคู่มือการใช้งานอุปกรณ์ ในหน้าต่าง Help

รูปที่ 3.18 ตัวอย่างการออกแบบหน้าจอแสดงผลด้วยโปรแกรม Codesys

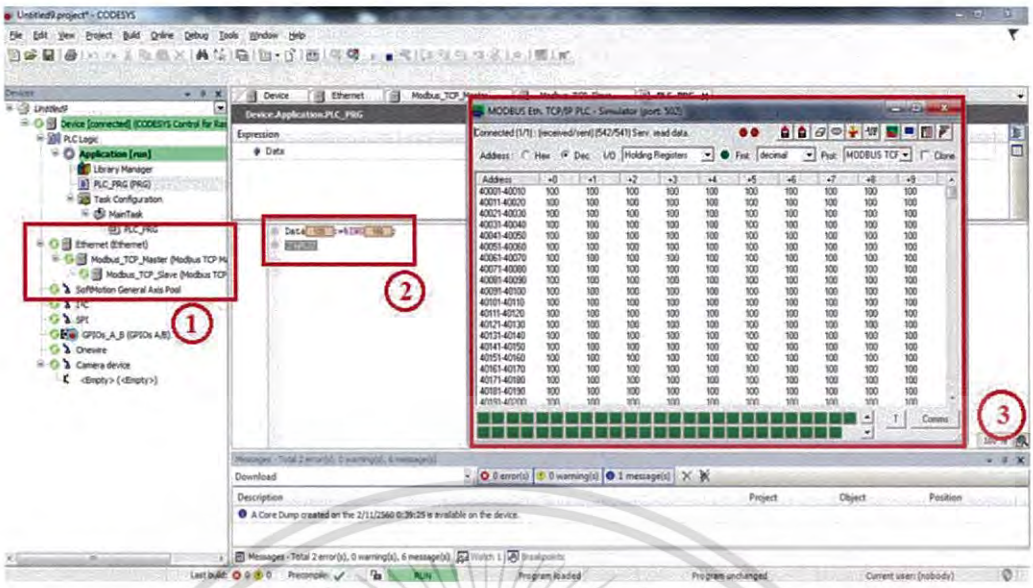
3.3.2 การจำลองการส่งข้อมูลแบบ Modbus โดยใช้โปรแกรม Modbus_RSsim และบันทึกข้อมูล

การจำลองวิธีเชื่อมต่ออุปกรณ์ให้สื่อสารกันในโปรโตคอล Modbus สามารถทำได้โดยกำหนดอุปกรณ์ มาสเตอร์ / สเลฟ (หมายเลข 1) และยังสามารถแสดงโค้ดตัวอย่างตาม (หมายเลข 2) ซึ่งสามารถแสดงผลข้อมูลโดยให้อุปกรณ์ Raspberry pi 3 เป็น มาสเตอร์ และ โน้ตบุ๊กเป็น สเลฟ และเปิดโปรแกรม Mod_Rssim (หมายเลข 3) ใช้ร่วมกับโปรแกรม Codesys ในการจำลองรับข้อมูล

จากการสืบค้นข้อมูลและศึกษาการทำงานของอุปกรณ์ raspberry pi 3 รวมถึงการใช้งานโปรแกรมที่เกี่ยวข้อง จึงได้ทำการปฏิบัติจริงโดยมีผลการทดลอง ดังรูป 3.19



(ก)



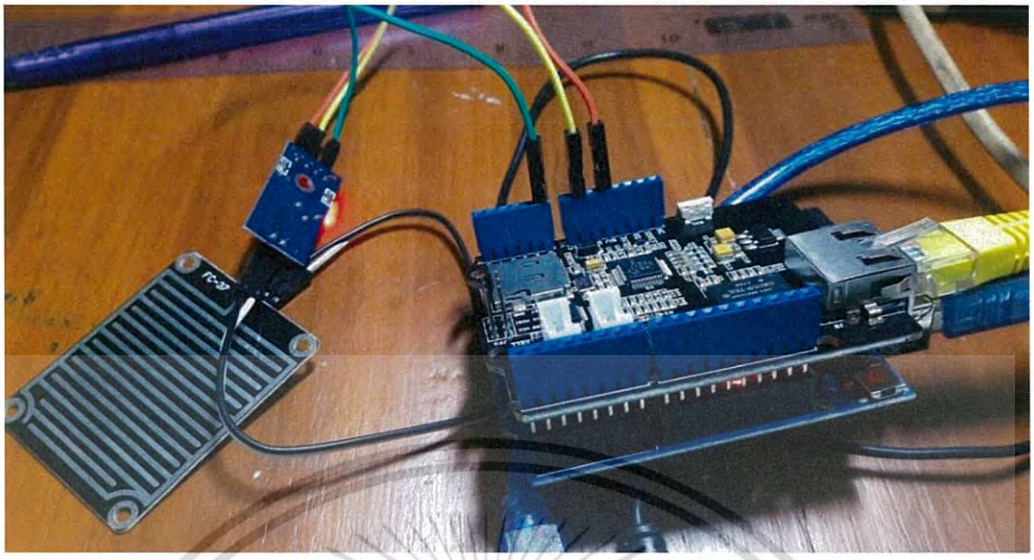
(ข)

รูปที่ 3.19 (ก) และ (ข) ผลการจำลองการสื่อสารโดยใช้โปรโตคอล Modbus รับข้อมูลจาก Mod_Rssim

3.3.3 การส่งข้อมูลแบบ Modbus จากเซ็นเซอร์โดยผ่านบอร์ด Arduino Uno r3 ใช้งานร่วมกับ Ethernet Shield W5500 และบันทึกข้อมูล

การรับส่งข้อมูลโดยใช้โปรโตคอล Modbus ในการจำลองเลือกใช้บอร์ด Raspberry pi เป็นอุปกรณ์ มาสเตอร์ และ Arduino Uno r3 เป็นอุปกรณ์ สเลฟ ดังรูปที่ 3.20 (ก) เนื่องจากมี library ที่รองรับการสื่อสารได้หลากหลายและใช้งานง่าย และเนื่องจากต้องการให้บอร์ด Arduino Uno r3 สามารถเชื่อมต่ออินเตอร์เน็ตและรองรับอินพุตได้ 8 Chanel จึงเลือกใช้ Ethernet Shield W5500 ต่อใช้งานร่วมกับเซ็นเซอร์ตรวจจับพารามิเตอร์ต่างๆ เช่น Temperature, Carbon monoxide, Flow เป็นต้น ดังรูปที่ 3.20 (ข) เขียนควบคุมด้วยโปรแกรม Codesys

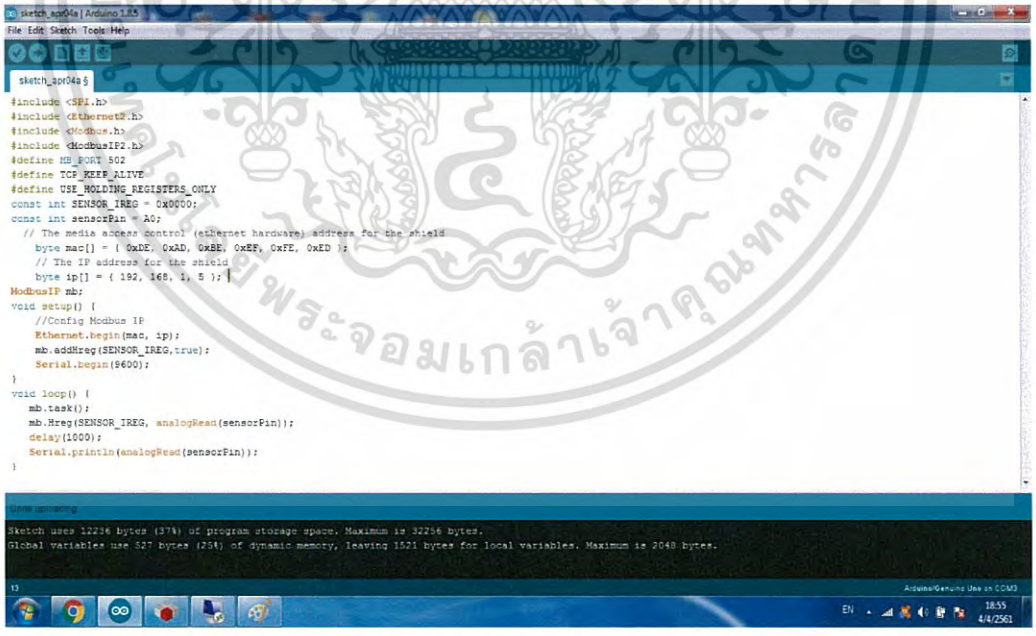
(ก)



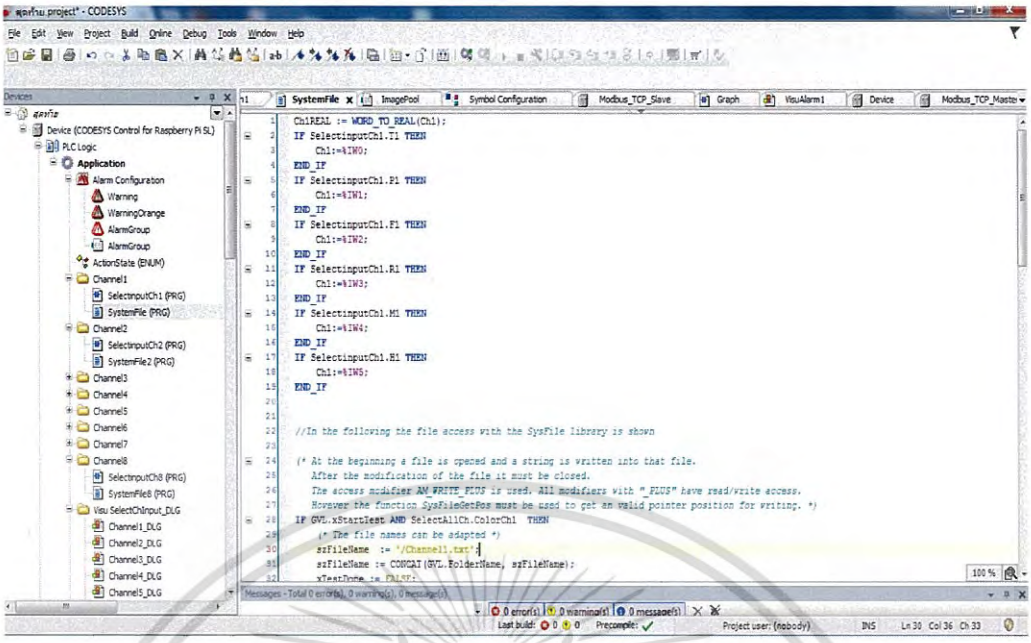
(ข)

รูปที่ 3.20 (ก) และ (ข) ตัวอย่างการเชื่อมต่อเซ็นเซอร์ Raindrop กับบอร์ด Arduino Uno r3

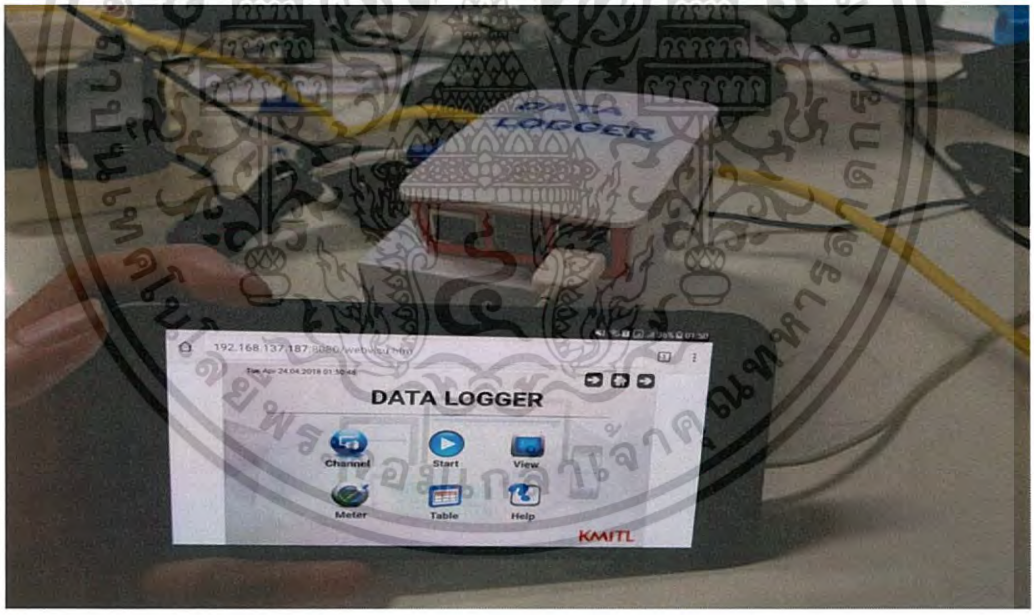
การต่อเซ็นเซอร์สามารถเลือกใช้ library ของ Modbus ที่มีในโปรแกรม Arduino IDE และสามารถเลือกฟังก์ชั่นโค้ดตามรูปที่ 3.21



รูปที่ 3.21 รูปแบบโค้ดการส่งข้อมูล 1 Channel จาก Arduino



รูปที่ 3.22 ตัวอย่างโค้ดการรับข้อมูล Temperature 1 Channel จาก Arduino โดยโปรแกรม Codesys



รูปที่ 3.23 ตัวอย่างการใช้งาน Data Logger บันทึกข้อมูล

3.3.4 การเก็บบันทึกข้อมูลโดยใช้ Anyviz Cloud

การบันทึกข้อมูลของอุปกรณ์ออกแบบให้สามารถเก็บข้อมูลได้ 2 ลักษณะ ดังนี้ บันทึกข้อมูลลงใน SD Card และบันทึกข้อมูลไว้บนเว็บฐานข้อมูล Anyviz ซึ่งเป็นระบบ Cloud ฟรีสำหรับผู้ที่ต้องการทดลองใช้ โดยสามารถเรียกดูข้อมูลย้อนหลังได้และยังสามารถดู

ข้อมูลแบบกราฟหรือตารางได้ เพียงแค่สมัครเข้าใช้งานและสามารถ login เข้าสู่ระบบของหน้าเว็บ Anyviz Cloud ดังรูปที่ 3.24 โดยสามารถเรียกใช้ดังนี้

1.) เข้าหน้าเว็บไซด์ Anyviz เพื่อสมัครเป็นสมาชิกเพื่อใช้บริการ Cloud

2.) เขียนโค้ดในโปรแกรม Codesys และกำหนดเลข Projectid และ password

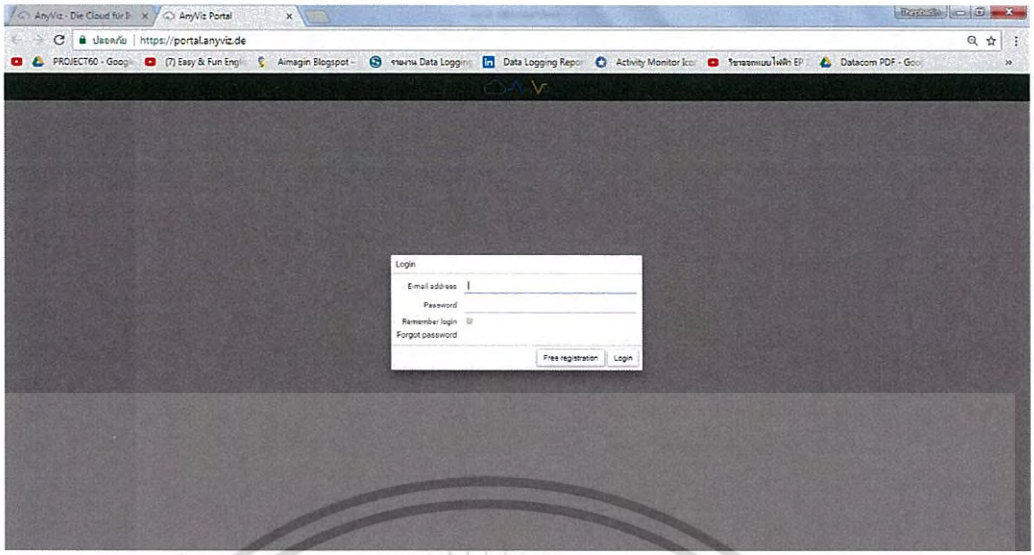
3.) เมื่อรันโปรแกรมสามารถเข้าใช้งานที่เว็บ Anyviz โดยใส่ password ตามที่ตั้งค่า

(ก)

```
PROGRAM PLC_PRG
VAR
        anyviz : Anyviz.AnyVizClient := (Projectid := 422 ,password := 'Bbbb');
END_VAR
```

```
anyviz ();
```

(ข)



(ค)

รูปที่ 3.24 (ก),(ข) และ (ค) ตัวอย่างโค้ดการส่งข้อมูลไปเก็บบนเว็บ Anyviz Cloud

บทที่ 4
การนำเสนอผลของข้อมูลและการวิเคราะห์ข้อมูล

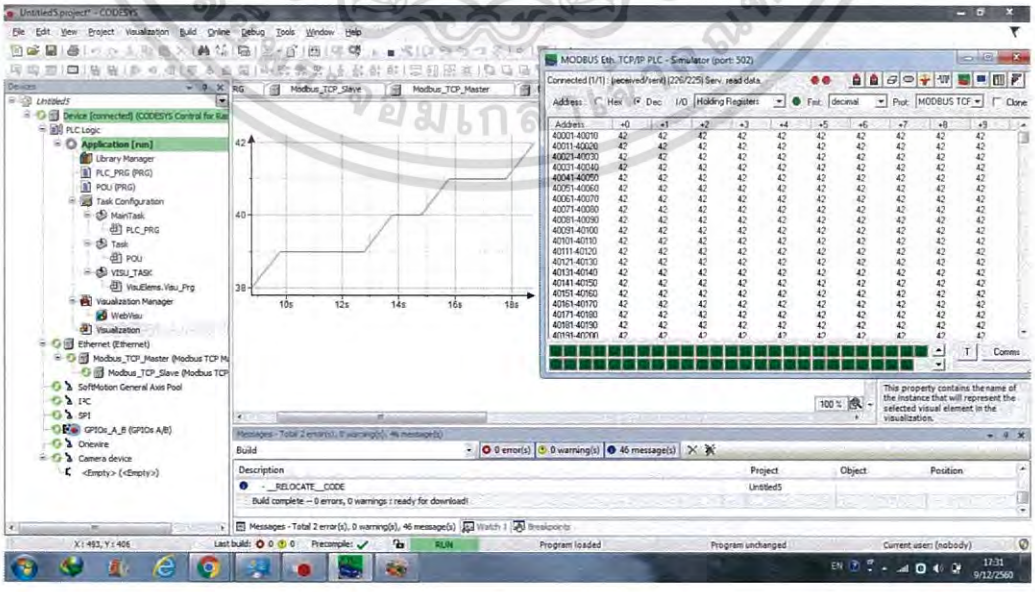
4.1 กล่าวนำ

เนื่องจากในการสร้างอุปกรณ์ Data Logger For Modbus เบื้องต้นผู้วิจัยได้เลือกใช้บอร์ด Raspberry pi และเขียนโปรแกรมควบคุมด้วยโปรแกรม Codesys ในขั้นการทดลองและการวิเคราะห์ข้อมูลผู้วิจัยไม่สามารถนำอุปกรณ์ไปเชื่อมต่อกับเซ็นเซอร์จริงได้ตลอดทั้งการทดลอง ผู้วิจัยจึงได้แบ่งการทดลองออกเป็น 2 ลักษณะ คือ ใช้การจำลองการเชื่อมต่อรับส่งข้อมูลแบบ Modbus โดยโปรแกรม Modbus_RSsim และการรับส่งข้อมูลจากเซ็นเซอร์โดยผ่านบอร์ด Arduino Uno r3 ใช้งานร่วมกับ Ethernet Shield W5500 เพื่อทดสอบการใช้งานของอุปกรณ์กับเซ็นเซอร์จริง

4.2 การนำเสนอผลของข้อมูล

4.2.1 การจำลองการรับส่งข้อมูลแบบ Modbus โดยโปรแกรม Modbus_RSsimและบันทึกข้อมูล

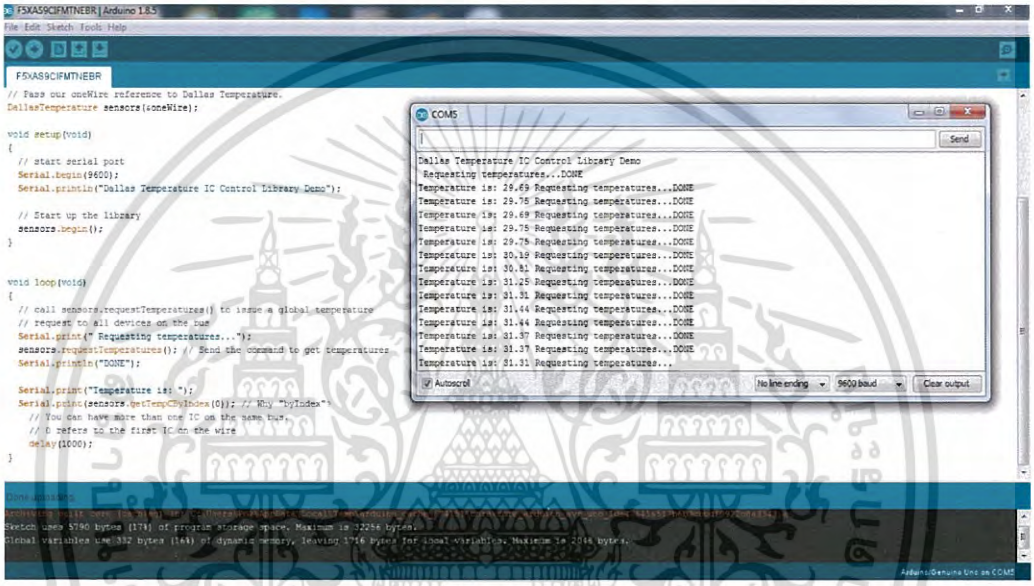
โปรแกรม Modbus_RSsim ผู้วิจัยนำมาใช้ในการจำลองเป็นอุปกรณ์ สเลฟ เนื่องจากเป็นโปรโตคอลที่ต้องการใช้งาน และสะดวกในการเขียนโปรแกรมในช่วงเริ่มแรกโดยไม่ต้องต่ออุปกรณ์จากภายนอกซึ่งผลของการใช้งานเป็นดังรูปที่ 4.1



รูปที่ 4.1 การเชื่อมต่อระหว่างโปรแกรม Codesys กับ Mod_Rssim

4.2.2 การรับส่งข้อมูลแบบ Modbus โดยเชื่อมต่อกับบอร์ด Arduino Uno r3 และบันทึกข้อมูล

สำหรับการรับ- ส่งข้อมูลจากบอร์ด Arduino ในรูปแบบ Modbus ทำได้โดยทดลองรับ-ส่งข้อมูลจากเซ็นเซอร์โดยผ่านบอร์ด Arduino ใช้งานร่วมกับ Ethernet Shield W5500 โดยผู้วิจัยได้ได้กำหนดช่องสัญญาณ 8 ช่องสำหรับต่อเซ็นเซอร์ โดยทดลองเก็บค่าพารามิเตอร์หลายแบบ เช่น Temperature, Pressure, Gas เป็นต้น โดยผลการทดลองเป็นดังรูปที่ 4.2 และ 4.3

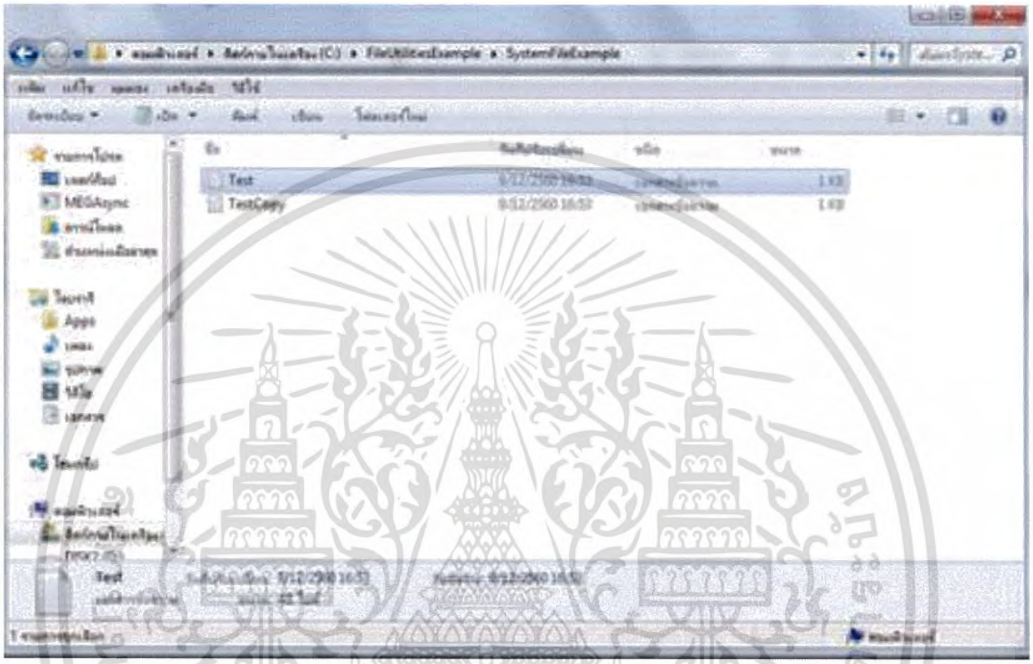


รูปที่ 4.2 ข้อมูลการเชื่อมต่อเซ็นเซอร์ Temperature กับบอร์ด Arduino Uno r3

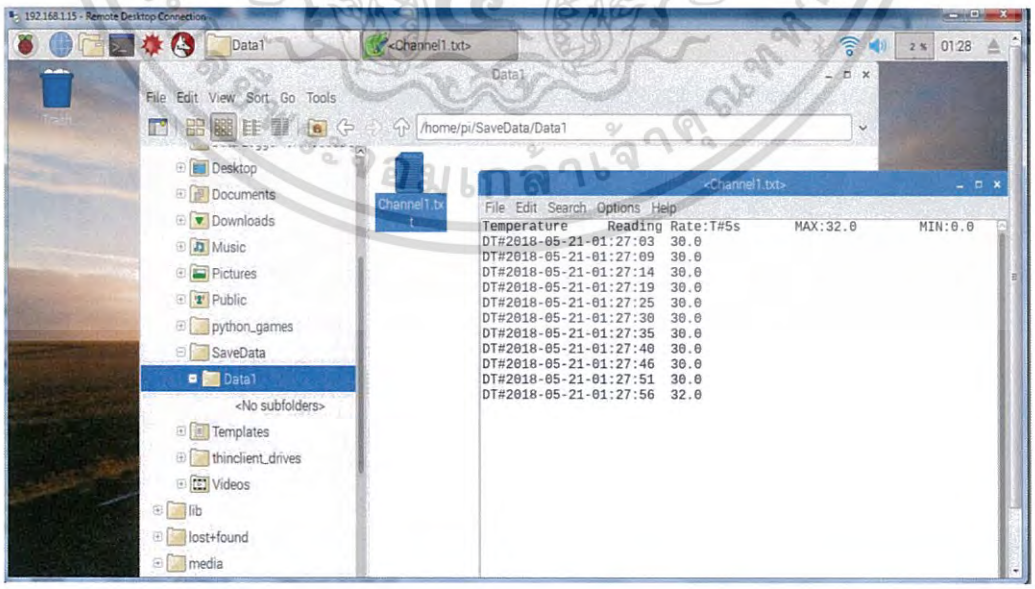
รูปที่ 4.3 การสื่อสารโดยใช้โปรโตคอล Modbus ระหว่างบอร์ด Raspberry pi และ Arduino

4.2.3 รูปแบบการแสดงผลข้อมูลที่เก็บบันทึกใน SD Card และฐานข้อมูล

การบันทึกข้อมูลของ Data logger สำหรับอุปกรณ์ Modbus สามารถบันทึกข้อมูลแบบ Real-time โดยแสดงข้อมูล วัน-เดือน-ปี, ระยะเวลาเก็บ, และพารามิเตอร์ที่ต้องการเก็บบันทึกเป็นไฟล์ .txt ลงใน SD Card ภายในบอร์ด Raspberry Pi 3 โดยแสดงข้อมูลดังรูปที่ 4.4 และ 4.5

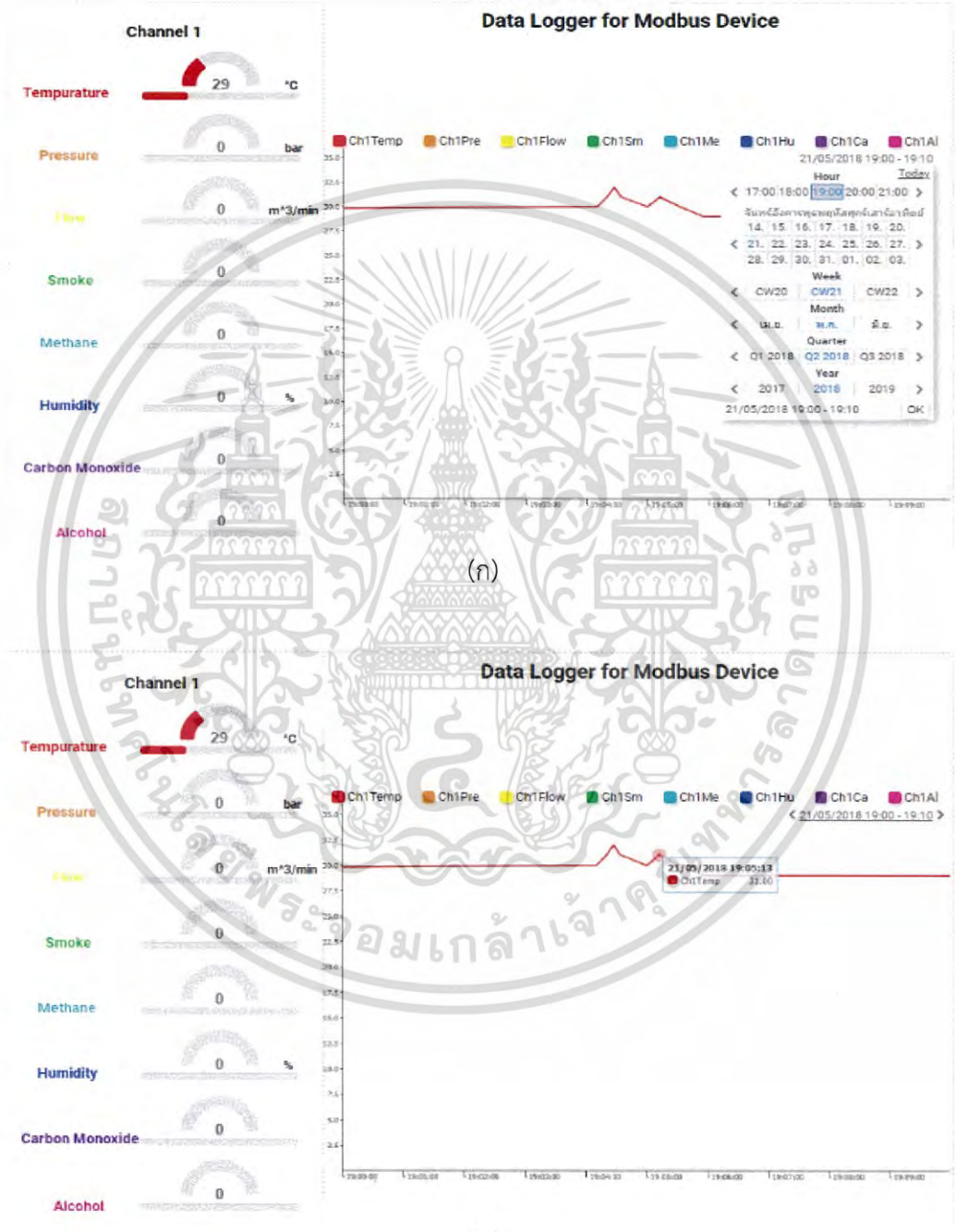


รูปที่ 4.4 ไฟล์ที่เก็บอยู่ในรูปของไฟล์ .txt



รูปที่ 4.5 ไฟล์ที่เก็บโดยอุปกรณ์ Data Logger For Modbus

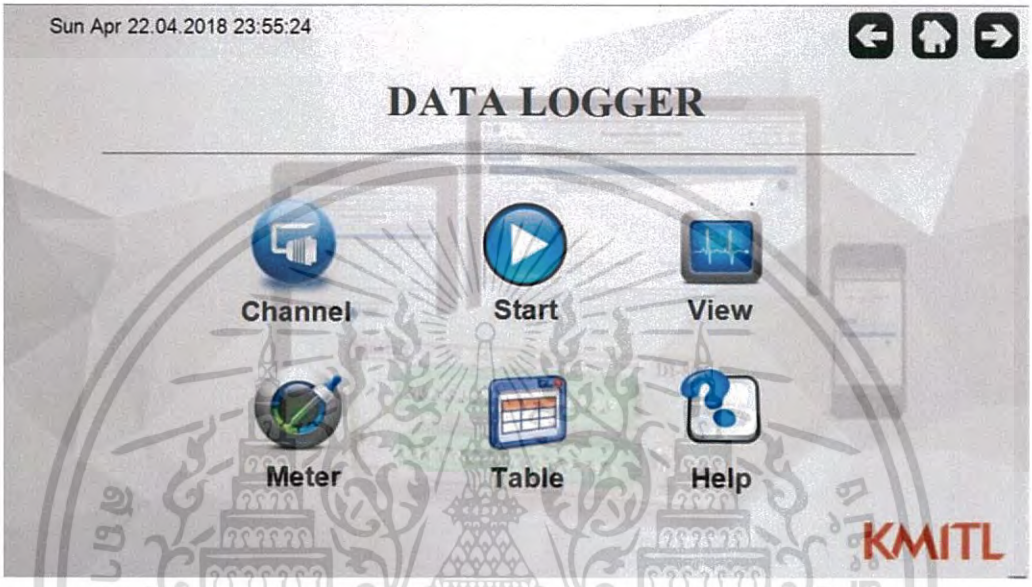
โดยการแสดงข้อมูลบนเว็บ Anyviz Cloud สามารถออกแบบการแสดงผลได้ไม่มาก เนื่องจากเป็นเว็บที่ให้ใช้งานฟรีจึงมีบางฟังก์ชั่นที่ใช้งานไม่ได้และมีเวลาจำกัดในการใช้งาน แต่ยังสามารถนำตัวแปรที่ต้องการบันทึกผลมาแสดงเป็นกราฟหรือตารางได้ และยังสามารถดูข้อมูลย้อนหลังได้ตลอดการบันทึก ดังรูปที่ 4.6



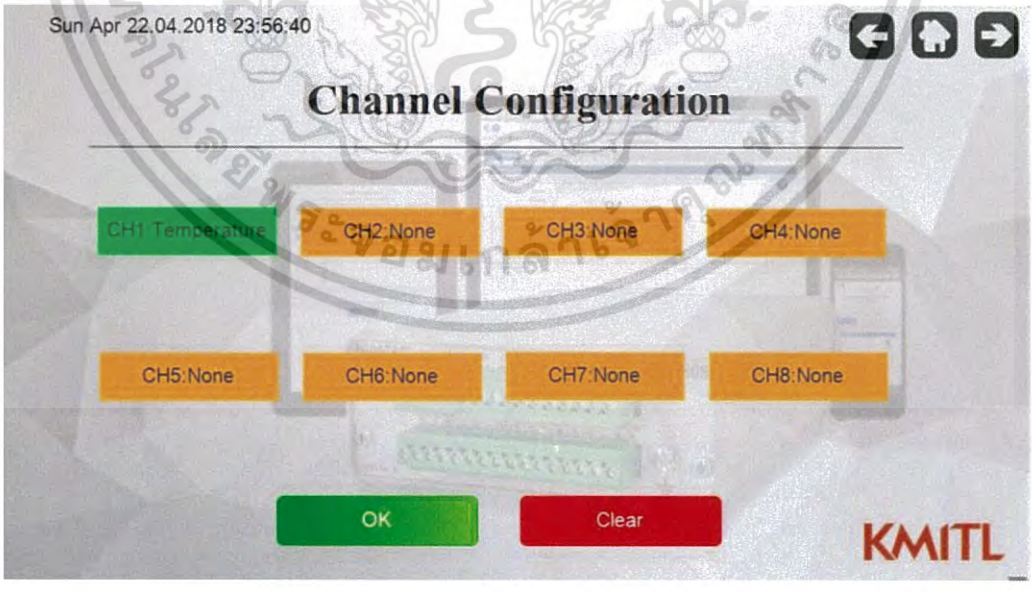
รูปที่ 4.6 (ก) และ (ข) ตัวอย่างการแสดงผลบนเว็บ Anyviz Cloud

4.2.4 การทำงานของหน้าจอแสดงผล

ในการแสดงผล ผู้วิจัยได้ออกแบบให้มีหน้าหลักและมีฟังก์ชั่นการใช้งานที่คล้ายคลึงกับอุปกรณ์ Data Logger ชนิดอื่นๆแต่มีฟังก์ชั่นที่มีความแตกต่างเช่น การเลือกช่องสัญญาณอินพุทได้ถึง 8 ช่อง การตั้งเวลาบันทึก และการแสดงผลของข้อมูลที่หลากหลาย เป็นต้น โดยตัวอย่างการแสดงผลของอุปกรณ์มีลักษณะ ดังรูปที่ 4.7 และ 4.8

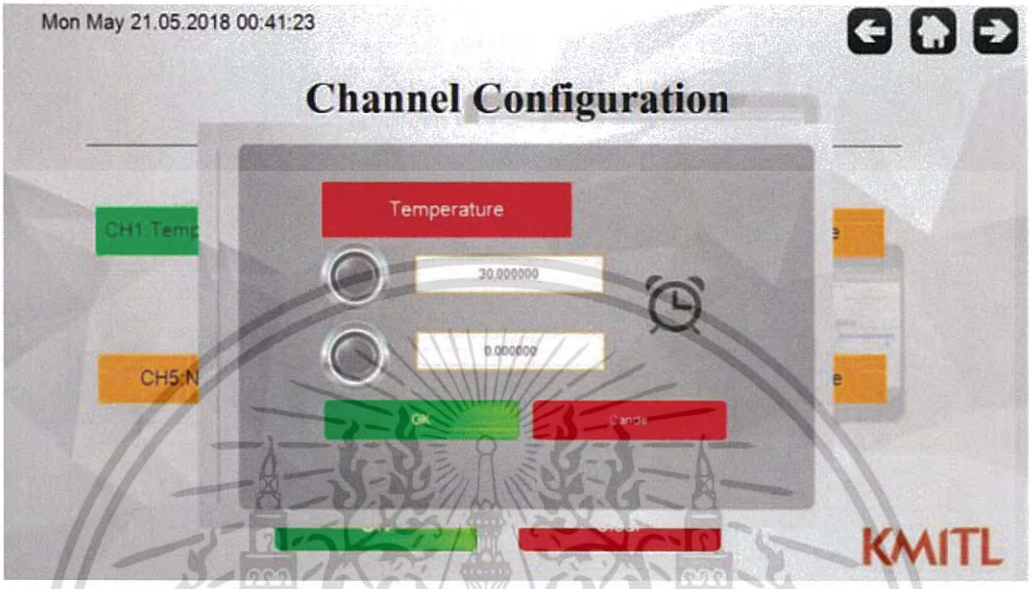


รูปที่ 4.7 หน้าหลักของอุปกรณ์ Data Logger For Modbus



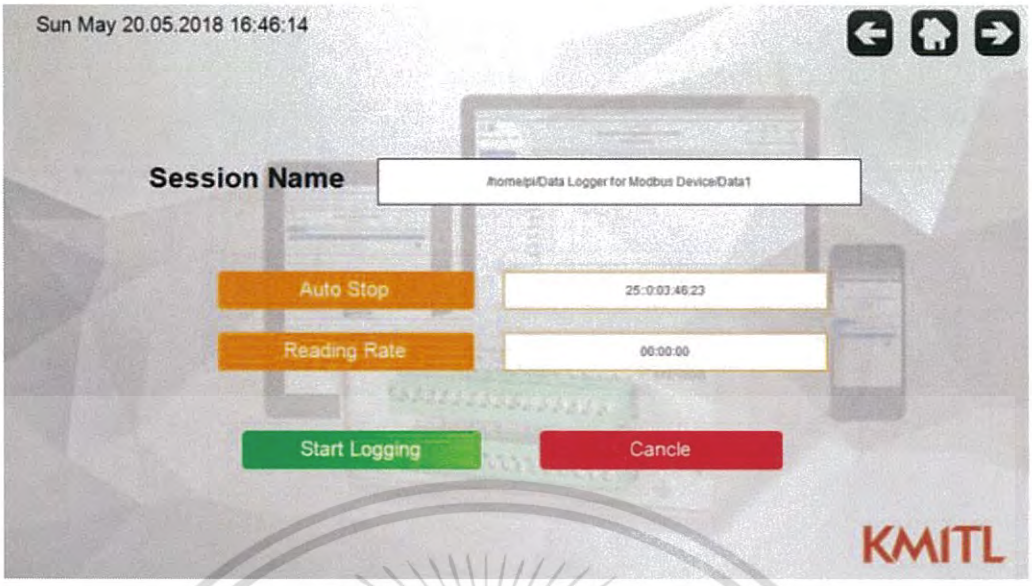
รูปที่ 4.8 ตัวอย่างการเลือก Channel ในการบันทึกค่าอุณหภูมิ

Data Logger สำหรับอุปกรณ์ Modbus สามารถตั้งค่าเวลาในการบันทึกและเวลาหยุดบันทึกได้และสามารถตั้งค่าเวลาในการแจ้งเตือนผู้ใช้ (Alarm) แสดงวิธีการใช้งานดังรูปที่ 4.9 และ 4.10



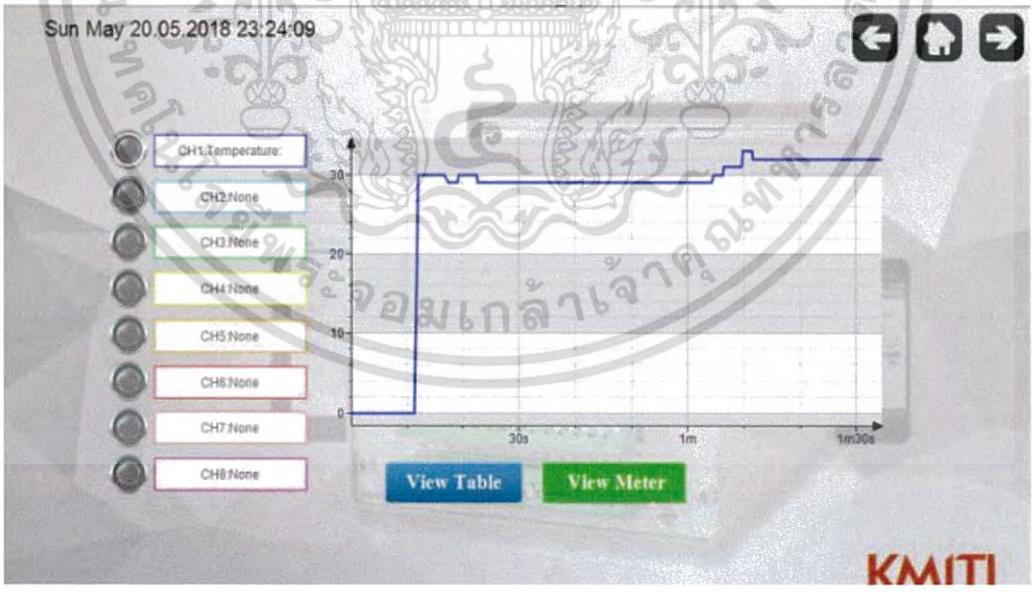
รูปที่ 4.9 การตั้งค่าเวลาในการแจ้งเตือนผู้ใช้

รูปที่ 4.10 หน้าต่างแสดงการแจ้งเตือนเมื่อข้อมูลถึงค่าอันตราย



รูปที่ 4.11 หน้าต่างการตั้งค่ารูปแบบต่างๆ

การทดสอบการทำงานและการแสดงผลจะต้องควบคู่ไปกับการตรวจสอบค่าที่บันทึกและแสดงในโปรแกรม Codesys ดังนั้นในการออกแบบและการนำค่าพารามิเตอร์ต่างๆมาแสดงนั้นจะต้องมีความถูกต้อง มีความคลาดเคลื่อนน้อยที่สุด และยังสามารถเลือกดูหน่วยที่ต้องการได้ดังรูปที่ 4.12, 4.13 และ 4.14

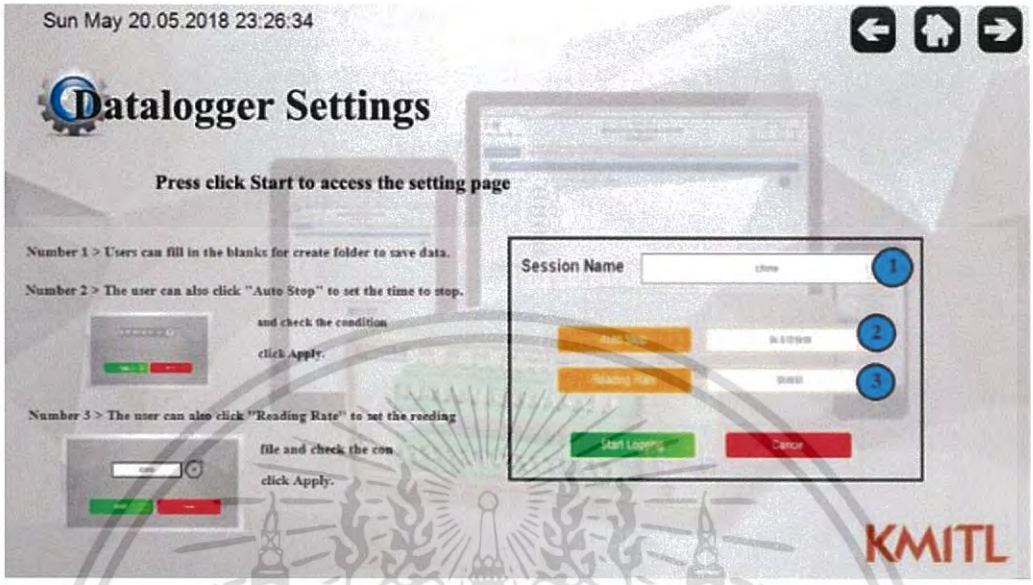


รูปที่ 4.12 การแสดงผลข้อมูลในรูปแบบกราฟในการบันทึกค่าอุณหภูมิ

รูปที่ 4.13 การแสดงผลข้อมูลในรูปแบบมิเตอร์ในการบันทึกค่าอุณหภูมิ

รูปที่ 4.14 หน้าต่างแสดงหน่วยอื่นๆ

Data Logger สำหรับอุปกรณ์ Modbus ยังมีฟังก์ชั่น Help ในการช่วยเหลือผู้ใช้งาน ซึ่งสามารถแสดงวิธีใช้งานและวิธีการตั้งค่าต่างๆของอุปกรณ์ ดังรูปที่ 4.15



รูปที่ 4.15 แสดงหน้า Help สำหรับผู้ใช้งาน

4.3 การวิเคราะห์ข้อมูล

ในหัวข้อนี้จะเป็นการเปรียบเทียบค่าที่เก็บได้จาก Data Logger สำหรับอุปกรณ์ Modbus ทั้ง 2 ลักษณะที่ทำการทดลองคือ ใช้การจำลองการเชื่อมต่อรับส่งข้อมูลแบบ Modbus โดยโปรแกรม Modbus_RSsim และการรับส่งข้อมูลจากเซ็นเซอร์โดยผ่านบอร์ด Arduino Uno r3 ใช้งานร่วมกับ Ethernet Shield W5500 เพื่อให้เห็นถึงประสิทธิภาพของอุปกรณ์ว่าสามารถเก็บข้อมูลมีความคลาดเคลื่อนเกิดขึ้นเท่าใด

4.3.1 การใช้วิธีจำลองด้วยโปรแกรม Modbus_RSsim ในการบันทึกข้อมูล

จากรูปผลการทดลองที่ 4.1 โดยจำลองการสื่อสารจากโปรแกรม Modbus_RSsim โดยให้โปรแกรมเป็น Slave โดยผลการทดลองแสดงให้เห็นว่าค่าที่บันทึกจากอุปกรณ์ Data Logger for Modbus บันทึกค่าได้โดยค่าความคลาดเคลื่อนน้อยมาก ซึ่งสามารถใช้งานได้

4.3.2 การต่อกับเซ็นเซอร์จริง ในการบันทึกข้อมูล

จากการทดลองต่อกับเซ็นเซอร์จริงพบว่าค่าที่บันทึกยังมีค่าความคลาดเคลื่อนจากค่าที่เซ็นเซอร์จริงวัดได้ ซึ่งอาจเกิดจากการใช้งานโปรแกรม Codesys ที่ติดตั้งฟรี เพราะมีฟังก์ชั่นบางฟังก์ชั่นที่ไม่สามารถใช้งานได้ และในการส่งข้อมูลจากบอร์ด Arduino มาที่เก็บที่ SD Card อาจจะมีสิ่งรบกวนบ้างในการสื่อสารเช่น ปัญหาจากกการตรวจจับข้อมูลของเซ็นเซอร์ ความเร็วในการแสดงผลต่างกัน เป็นต้น

อย่างไรก็ตาม อุปกรณ์สามารถบันทึกบันทึกข้อมูลได้ และมีค่าความคลาดเคลื่อนอยู่ในเกณฑ์ที่ยอมรับได้ โดยสามารถใช้เป็นอุปกรณ์เก็บข้อมูลและใช้ข้อมูลในการวิเคราะห์หาแนวโน้มเพื่อใช้พัฒนาต่อไปได้

บทที่ 5
สรุปผลการวิจัย และข้อเสนอแนะจากการวิจัย

5.1 สรุปผลการทดลอง

ในการทดลองนี้เป็นการสร้าง Data logger สำหรับอุปกรณ์ Modbus จากบอร์ด Raspberry Pi 3 โดยในการทดลองผู้วิจัยได้แบ่งการบันทึกข้อมูลออกเป็น 2 แบบ คือ การบันทึกข้อมูลโดยใช้โปรแกรม Mod_RSsim (โปรแกรมจำลอง) และจากเซ็นเซอร์ต่อกับ Ethernet Shield W5500 (ส่งข้อมูลแบบ Modbus โดยใช้บอร์ด Arduino) จากการทดลองจะเห็นว่าการบันทึกข้อมูลจากโปรแกรม Mod_RSsim สามารถบันทึกข้อมูลได้ดีกว่าการเก็บข้อมูลจากเซ็นเซอร์จริงที่ส่งมาจากบอร์ด Arduino โดยค่าที่ได้จะมีค่าที่เที่ยงตรงกว่า ความคลาดเคลื่อนน้อยกว่า มีการส่งข้อมูลถึงกันเร็วกว่า แต่ค่าที่รับจากบอร์ด Arduino นั้นมีความคลาดเคลื่อนอยู่ในเกณฑ์ที่ยอมรับได้ ทั้งนี้ Data logger สำหรับอุปกรณ์ Modbus สามารถบันทึกข้อมูลแบบ Real-time โดยแสดงข้อมูล วัน-เดือน-ปี, ระยะเวลาเก็บ,ตั้งการแจ้งเตือนค่าสูงสุดต่ำสุดได้ และพารามิเตอร์ที่ต้องการจะบันทึกในรูปแบบไฟล์ .txt ลงใน SD Card ภายในบอร์ด Raspberry Pi 3 และสามารถเรียกดูข้อมูลย้อนหลังได้ที่เว็บ Anyviz cloud ในเบื้องต้นอุปกรณ์ยังมีข้อจำกัดในการใช้งานบางประการ เนื่องจากเป็นการสร้างอุปกรณ์ใช้ต้นทุนต่ำและซอร์ฟแวร์ฟรีเกือบทั้งหมด อย่างไรก็ตาม Data logger สำหรับอุปกรณ์ Modbus นี้สามารถใช้งานและบันทึกผลข้อมูลได้อยู่ในเกณฑ์ดีและสามารถนำข้อมูลที่ได้มาใช้ในการวิเคราะห์กระบวนการต่างๆเพื่อพัฒนาต่อไปได้

5.2 ปัญหาที่พบในการทดลองและแนวทางการแก้ปัญหา

1. ในการพัฒนาให้บอร์ด Raspberry Pi 3 เป็น Data logger สำหรับอุปกรณ์ Modbus เขียนควบคุมการทำงานโดยใช้โปรแกรม Codesys พบว่าโปรแกรมยังมีบั๊คเกิดขึ้นในการเขียนโปรแกรมมากพอสมควร วิธีแก้ไขปัญหาคือพยายามหลีกเลี่ยงฟังก์ชั่นที่ใช้งานไม่ได้และเลือกใช้ฟังก์ชั่นที่มีบั๊คน้อยที่สุด

2. เมื่อเขียนโปรแกรมเยอะ จะทำให้การอัพโหลดลง Raspeberry pi 3 เกิดปัญหาหรืออัพโหลดไม่ได้ จึงควรหลีกเลี่ยงการเขียนโปรแกรมให้กินพื้นที่ข้อมูลมากเกินความจำเป็น

3. ในการศึกษาโปรแกรม Codesys ค่อนข้างทำได้ยากเนื่องจากมีคนใช้น้อย จึงทำให้การสืบค้นข้อมูลหรือหาตัวอย่างในการทำนั้นใช้เวลานาน วิธีแก้ปัญหาคือปรึกษาอาจารย์ที่ปรึกษาและช่วยกันสืบค้นข้อมูลและสอบถามผู้รู้จนได้ข้อมูลที่ต้องการใช้ในการทดลอง

4. การส่งข้อมูลจาก Arduino Ethernet W5500 ไปยัง Data logger for Modbus

device ทำได้ยากเนื่องจากเป็นอุปกรณ์ที่มีราคาถูก วิธีแก้ไขปัญหาคืออาจจะใช้อุปกรณ์ที่มีคุณภาพดีกว่าเพื่อการส่งข้อมูลที่ดีกว่าขึ้นและความคลาดเคลื่อนน้อยลง

5. ในการส่งข้อมูลไปเก็บบนฐานข้อมูลผู้วิจัยใช้เลือกใช้งานเว็บ Anyviz cloud เนื่องจากเป็นเว็บที่สามารถใช้งานได้ฟรี แต่ปัญหาที่ตามมาคือไม่สามารถใช้งานได้ทุกฟังก์ชั่นและมีเวลาจำกัดในการใช้งาน

6. งบประมาณในการซื้ออุปกรณ์การทดลองไม่พอสำหรับซื้ออุปกรณ์ที่มีคุณภาพที่ดีซึ่งแพงเกินไป วิธีแก้ไขปัญหาคือซื้ออุปกรณ์ที่มีราคาถูกลงมาอาจจะไม่ดีเท่าอุปกรณ์ที่แพงแต่สามารถใช้งานได้

5.3 ข้อเสนอแนะ

ซอฟต์แวร์ที่ใช้ในการทดลองเป็นซอฟต์แวร์ฟรี ทำให้มีเวลาจำกัดในการใช้งานต่างๆ อย่างเช่น Library CODESYS Control for Raspberry Pi SL ใช้งานได้ไม่เกิน 2 ชั่วโมง เมื่อเกินต้องรีเซ็ตใหม่เพื่อให้ใช้งานต่อได้ รวมถึงเว็บ Anyviz cloud เนื่องจากเป็นเว็บที่สามารถใช้งานได้ฟรีแต่ก็มีปัญหาที่ตามมาคือไม่สามารถใช้งานได้ทุกฟังก์ชั่นและมีเวลาจำกัดในการใช้งาน ทำให้ไม่สะดวกต่อการวิจัย เพื่อการพัฒนาที่รวดเร็วควรซื้อซอฟต์แวร์บางอย่างที่ทำให้การใช้งานนั้นสะดวกขึ้น ไม่ต้องเสียเวลาในการหาใช้ซอฟแวร์ฟรี ซึ่งในบางส่วนก็มีอยู่แล้วแต่ต้องมาทำเพิ่มเพื่อใช้ฟรีหรือใช้งานได้ทุกฟังก์ชั่น

บรรณานุกรม

[1] พิชิต จินตโกศลวิทย, "การสื่อสารข้อมูลในงานอุตสาหกรรม", กรุงเทพฯ : ซีเอ็ดยูเคชั่น, 2552

[2] "การสื่อสารแบบ Modbus Protocol", 2554. [ระบบออนไลน์] แหล่งที่มา : https://riverplusblog.com/2011/08/18/plc-protocol (18 พฤศจิกายน 2560).

[3] "บทความ What is a Data Logger", 2553. [ระบบออนไลน์] แหล่งที่มา : https://medium.com/project-mar/data-logger-with-rest-service-149c4a45a3f3 (20 ตุลาคม 2560).

[4] "ชนิดของ Data logger", 2556. [ระบบออนไลน์] แหล่งที่มา : http://www.wisco.co.th main/sites/default/files/articles/Logger%20Knowledge_0.pdf (20 ตุลาคม 2560).

[5] "การใช้งานเบื้องต้น Raspberry Pi 3", 2559. [ระบบออนไลน์] แหล่งที่มา : http://http://cpre.kmutnb.ac.th/esl/learning/index.php?article=rpi3_quickstart (23 ตุลาคม 2560).

[6] "คู่มือและการตั้งค่า Raspberry Pi 3 ", 2557. [ระบบออนไลน์] แหล่งที่มา : https://www.thaieasyelec.com/downloads/ETEE056A/8inch%20TFT%20 Touchscreen %20for%20Raspberry.pdf (24 ตุลาคม 2560).

[7] "การต่อเซ็นเซอร์กับบอร์ด Arduino", 2560. [ระบบออนไลน์] แหล่งที่มา : https://thaieasyelec.com/article-wiki/review-product-article/ (13 เมษายน 2561)

[8] "Cloud server คือ", 2556. [ระบบออนไลน์] แหล่งที่มา : http://www.thaicreate.com/web-host/web-host-cloud-server.html (10 เมษายน 2561)

[9] "Modbus TCP/IP protocol", May 18,2002. [Online] Available : https://www.rtaautomation.com/technologies/modbus-tcpip (Nov 18, 2018)

[10] "Master/Slave", May 18,2002. [Online] Available : https://www.ibm.com/support /knowledgecenter (Nov 22, 2018)

[11] "Modbus RS_sim", May 20,2010. [Online] Available : http://www.plcsimulator.org / (Nov 22, 2018)

[12] "Codesys download software", Mar 20,2017. [Online] Available : https://www.codesys.com/ (Oct 22, 2017)

[13] "Codesys connect raspberry pi 3", Sep 9,2017. [Online] Available : https://www.youtube.com/watch?v=6FPf3RHWyeU (Oct 25, 2017)

[14] "Codesys Store", Mar 20,2017. [Online] Available : https://store.codesys.com/ (Oct 25, 2017)

[15] "Codesys read write file", Oct 17,2016. [Online] Available : https://www402.abbext.com/DownloadFile.aspx?file=UserFiles (Oct 17, 2017)

[16] "Trace codesys", May 1,2016. [Online] Available : https://www.youtube.com/watch?v=EdBltV4NE3k (Oct 19, 2017)

[17] "Ethernet shield W5500", 2015. [Online] Available : http://wiki.seeedstudio.com/W5500_Ethernet_Shield_v1.0/ (Apr 13, 2018)

[18] "Temperature Sensor Arduino", Sep 29,2014. [Online] Available : https://learn.adafruit.com/tmp36-temperature-sensor/using-a-temp-sensor (Apr 20, 2017)

[19] "Anyviz cloud", May 15,2016. [Online] Available : https://www.anyviz.de/en/ (Apr 10, 2018)

[20] "AnyViz Cloud Adapter", May 15,2016. [Online] Available : https://store.codesys.com/anyviz-cloud-adapter.html https://www.anyviz.de/en/ (Apr 12, 2018)

ภาคผนวก

ภาคผนวก ก.
ชุดคำสั่งที่ใช้ในการทดลอง

1. โค้ดที่ใช้ในการบันทึกข้อมูลลงใน SD Card (1 ช่องสัญญาณ)

```
PROGRAM MAIN
VAR
    xInit   : BOOL;
    udiCreateError: UDINT;
    Num:INT:=0;
    Load1:BOOL;
    Load2:BOOL;
END_VAR
```

```
(* Create the base directory *)
IF NOT xInit THEN
    udiCreateError := SysDirCreate(szDir:= GVL.FolderName);
END_IF
xInit := TRUE;
//Folder Name
GVL.FolderName := '/home/pi/Data Logger for Modbus Device/Data1' ;
//Synchronous file modification.
SystemFile();
SystemFile2();
SystemFile3();
SystemFile4();
SystemFile5();
SystemFile6();
SystemFile7();
SystemFile8();
ReadingRate();
Table1();
Table2();
```

```
IF GVL.xStartTest AND SelectAllCh.ColorCh8  THEN
    (* The file names can be adapted *)
    szFileName  := 'Channel8.txt';
    szFileName := CONCAT(GVL.FolderName, szFileName);
    xTestDone := FALSE;
    xError := FALSE;

    (* delete existing example files*)
    udiDeleteError1:=SysFileDelete(szFileName:=szFileName );

    szTestLine := SelectAllCh.NameOfCh8 ;
    hFile := SysFileOpen(szFile:=szFileName, am:=SysFile.AM_WRITE_PLUS, pResult:=ADR(udiOpenError1));
    IF hFile <> RTS_INVALID_HANDLE  THEN
        udiPosError := SysFileGetPos(hFile:=hFile, pulPos:=ADR(udiPos));
        udiWrite := SysFileWrite(hFile:=hFile, pbyBuffer:=ADR(szTestLine), ulSize:=INT_TO_UDINT(LEN(szTestLine)), pResult:=ADR(udiWriteError1));
        udiCloseError1  := SysFileClose(hFile:=hFile);
    END_IF

    (* The access modifier AM_READ is used. The reading result is stored into a string variable.*)
    hFile := SysFileOpen(szFile:=szFileName, am:=SysFile.AM_READ, pResult:=ADR(udiOpenError2));
    IF hFile <> RTS_INVALID_HANDLE  THEN
        udiBytesRead := SysFileRead(hFile:= hFile, pbyBuffer:= ADR(szReadLine), ulSize:=SIZEOF(szReadLine), pResult:=ADR(udiReadError1));
        // Cut off the unused characters.
        szReadLine[udiBytesRead] := 0;
        (* check the result from the file *)
        IF szReadLine <> szTestLine THEN
            xError := TRUE;
        END_IF
        udiCloseError2  := SysFileClose(hFile:=hFile);
    END_IF
```

```
END_IF

IF GVL.Q1 AND SelectAllCh.ColorCh8 THEN
    (* The modifier AM_APPEND can be used to append a string at the end of the file.*)
    szTestLine := CONCAT(CONCAT('$n',DT_TO_STRING(GVL.Date_and_time_format)),CONCAT('$t',REAL_TO_STRING(Ch8)));
    hFile := SysFileOpen(szFile:=szFileName, am:=SysFile.AM_APPEND, pResult:=ADR(udiOpenError3));
    IF hFile <> RTS_INVALID_HANDLE  THEN
        udiWrite := SysFileWrite(hFile:= hFile, pbyBuffer:= ADR(szTestLine), ulSize:=INT_TO_UDINT(LEN(szTestLine)), pResult:=ADR(udiWriteError2))
        udiCloseError3  := SysFileClose(hFile:=hFile);
    END_IF
END_IF
```

```
IF GVL.Stop AND SelectAllCh.ColorCh8 THEN
    // Some file information
    SysFileGetTime(szFileName:=szFileName, ptFileTime:=ADR(udcFileTime));
    //creation time of the file in local time
    tCreation    := UDINT_TO_DT(udcFileTime.tCreation);
    // last access of the file
    tLastAccess  := UDINT_TO_DT(udcFileTime.tLastAccess);
    // last modification of the file
    tLastModification    := UDINT_TO_DT(udcFileTime.tLastModification);
    udiFileSize  := SysFileGetSize(szFileName:=szFileName, pResult:=ADR(udiSizeError));
    SysFileGetPath(szFileName:=szFileName, szPath:=szPathName, diMaxLen:=255);

     IF udiPosError = 0 AND udiCopyError = 0 AND udiWriteError1 = 0 AND udiWriteError2 = 0 AND udiWriteError3 = 0 AND udiOpenError1 = 0
     AND udiOpenError2 = 0 AND udiOpenError3 = 0 AND udiReadError1 = 0 AND udiReadError2 = 0
     AND udiReadError3 = 0 AND udiCloseError1 = 0 AND udiCloseError2 = 0 AND udiCloseError3 = 0 AND udiSizeError = 0 AND  xError = FALSE THEN
        xTestDone := TRUE;
    ELSE
        xError := TRUE;
    END_IF
END_IF
(*MAX*)
IF Ch8REAL > SelectAllCh.MaxCh8LR AND SelectAllCh.MaxSwitch8 THEN
    Alarm1Ch8 := TRUE;
END_IF
(*MIN*)
IF Ch8REAL < SelectAllCh.MinCh8LR AND SelectAllCh.MaxSwitch8 THEN
    Alarm2Ch8 := TRUE;
END_IF
```

2. โค้ดแสดง วัน-เวลาปัจจุบัน

```
PROGRAM LocalTime
VAR
END_VAR
```

```
// 1981
// Get time in seconds since 1970 :
GVL.Date_and_time_in_seconds := SysTimeRtcGet(GVL.Date_and_Time_result);
IF GVL.Date_and_Time_result <> 0 THEN
RETURN;
END_IF

// Convert UTC seconds to local time seconds, regarding Timezone and Summer time :
GVL.Date_and_Time_result := SysTimeRtcConvertUtcToLocal(GVL.Date_and_time_in_seconds, GVL.Local_date_time_seconds);
IF GVL.Date_and_Time_result <> 0 THEN
RETURN;
END_IF

// Convert the UTC seconds to a DATE_AND_TIME variable dt#yyyy-mm-dd-hh:mm:ss :
GVL.Date_and_time_format := UDINT_TO_DT(GVL.Local_date_time_seconds);
// Convert DATE_AND_TIME to a String :
GVL.Date_and_time_string := DT_TO_STRING(GVL.Date_and_time_format);
```

3. โค้ดในการเลือกช่องสัญญาณตั้งเวลาและตั้ง Alarm (โดยแสดงโค้ดแค่ 1 ช่องสัญญาณ)

```
//Clear
GVL.Clear := GVL.ClearCh1 AND GVL.ClearCh2 AND GVL.ClearCh3 AND GVL.ClearCh4
          AND GVL.ClearCh5 AND GVL.ClearCh6 AND GVL.ClearCh7 AND GVL.ClearCh8;
// Deactivate Display Start/Stop Button
GVL.OKAll := NOT(ColorCh1)AND NOT(ColorCh2)AND NOT(ColorCh3);
//Input Chaanel1
    ColorCh1 := SelectinputCh1.T1 OR SelectinputCh1.P1 OR SelectinputCh1.F1
         OR SelectinputCh1.R1 OR SelectinputCh1.A1 OR SelectinputCh1.M1
         OR SelectinputCh1.Cb1 OR SelectinputCh1.H1 ;
    IF NOT ColorCh1 THEN
        NameOfCh1 := 'None';
    END_IF
    IF SelectinputCh1.T1 THEN
        NameOfCh1 := 'Temperature';
    END_IF
    IF SelectinputCh1.P1 THEN
        NameOfCh1 := 'Pressure';
    END_IF
    IF SelectinputCh1.F1 THEN
        NameOfCh1 := 'Flow';
    END_IF
    IF SelectinputCh1.R1 THEN
        NameOfCh1 := 'Raindrop';
    END_IF
    IF SelectinputCh1.M1 THEN
        NameOfCh1 := 'Methen';
    END_IF
    IF SelectinputCh1.H1 THEN
        NameOfCh1 := 'Humidity';
    END_IF


    //Alarm Ch1
MaxBlank1:= NOT MaxSwitch1;
MinBlank1:= NOT MinSwitch1;
LineMax1:= CONCAT('MAX:',REAL_TO_STRING(MaxCh1LR));
LineMin1:= CONCAT('MIN:',REAL_TO_STRING(MinCh1LR));
LineMaxMin1:= CONCAT(LineMax1,LineMin1);
```

```
PROGRAM SettimeforLog
VAR
    //Set Time Code
    DateAndTimeOfStop       : DATE_AND_TIME;
    AutoStopDate            : DATE_AND_TIME;
    DelayedTimeDate         :DATE_AND_TIME;
    ReadingRate             :TIME;
    Timer                   :TIME;
```

```
//Set Time
//Convert DT to UDINT
AutoStopDate1 := DT_TO_UDINT(AutoStopDate);
AutoStopDate2 := AutoStopDate1 - GVL.Local_date_time_seconds;
ResultTime:=UDINT_TO_TIME(AutoStopDate2);
TimeToUse := ResultTime * 1000;

LineReading:=CONCAT('$t',CONCAT('Reading Rate:',(TIME_TO_STRING(ReadingRate))));
//Start
STP();

//End of time
 IF AutoStopDate2 = Stoptime THEN
    GVL.Stop:= TRUE;
 END_IF;
 IF GVL.Stop THEN
     AutoStopDate2 := Stoptime ;
 END_IF;
```

4. โค้ดเริ่มและหยุดแสดงกราฟ

```
PROGRAM STP_Grph
VAR
    //Return STOP to FALSE
    TONStop: TON;
    ResetStop: BOOL;
    //Trace
    bStart:BOOL;
    bStop:BOOL:=TRUE;
END_VAR
```

```
// Graph
bStart:=GVL.xStartTest;
  IF GVL.Stop THEN
    bStop:= TRUE;
  END_IF
//Return STOP to FALSE
TONStop(IN := GVL.Stop, PT:= T#1S);
ResetStop := TONStop.Q;
    IF ResetStop THEN
     GVL.Stop:= FALSE;
     GVL.xStartTest:=FALSE;
    END_IF
```

5. โค้ดแสดงการแปลงหน่วย ยกตัวอย่าง Temperature, pressures

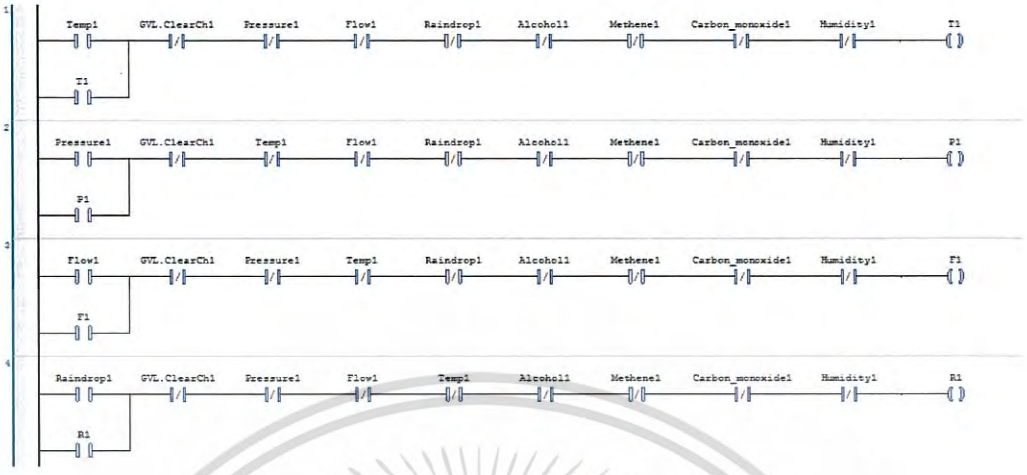
```
//temp
TempK1 := C2K.Convert(TempC1);
TempF1 := C2F.Convert(TempC1);
TempC1 := C2F.Reverse(TempF1);
TempC1 := C2K.Reverse(TempK1);

//pressure
psi := atm2psi.Convert(atm);
kPa := atm2kpa.Convert(atm);
bar := atm2bar.Convert(atm);
Hg  := atm2Hg.Convert(atm);
atm := atm2psi.Reverse(psi);
atm := atm2kpa.Reverse(kpa);
atm := atm2bar.Reverse(bar);
atm := atm2Hg.Reverse(Hg);
```

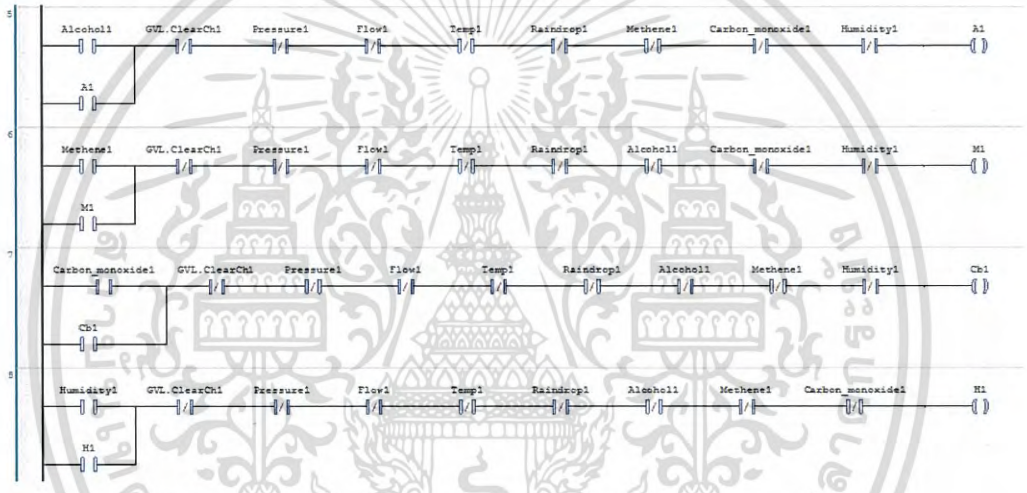
6. โค้ดแสดงค่าแบบ Real-time โดยแสดงผลในตาราง

```
//Table 1
Ch1ST := REAL_TO_STRING(SystemFile.Ch1REAL);
Ch2ST := REAL_TO_STRING(SystemFile2.Ch2REAL);
Ch3ST := REAL_TO_STRING(SystemFile3.Ch3REAL);
Ch4ST := REAL_TO_STRING(SystemFile4.Ch4REAL);
IF SelectAllCh.ColorCh1 THEN
CON1 := CONCAT('$t',Ch1ST);
ELSE CON1 := CONCAT('$t','');
END_IF
IF SelectAllCh.ColorCh2 THEN
CON2 := CONCAT('$t',Ch2ST);
ELSE CON2 := CONCAT('$t','');
END_IF
IF SelectAllCh.ColorCh3 THEN
CON3 := CONCAT('$t',Ch3ST);
ELSE CON3 := CONCAT('$t','');
END_IF
IF SelectAllCh.ColorCh4 THEN
CON4 := CONCAT('$t',Ch4ST);
ELSE CON4 := CONCAT('$t','');
END_IF
TimeString := CONCAT('$n',GVL.Date_and_time_string);
Ch1to4 := CONCAT(CONCAT(CON1,CON2),CONCAT(CON3,CON4));
sValue := CONCAT(TimeString,Ch1to4);
```

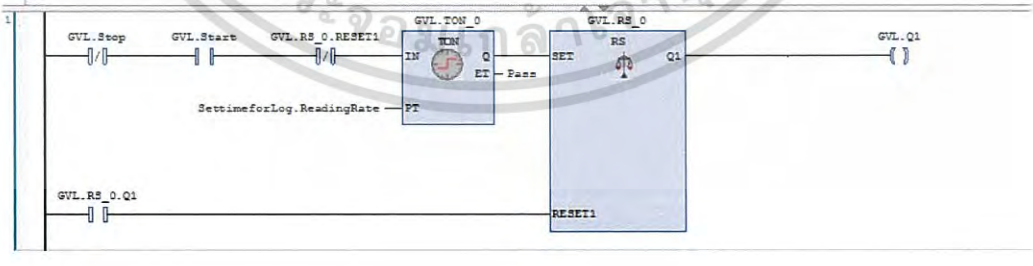
```
IF GVL.Q1 AND (SelectAllCh.ColorCh1 OR SelectAllCh.ColorCh2 OR SelectAllCh.ColorCh3 OR SelectAllCh.ColorCh4) THEN
    state:=ActionState.ADD_ELEMENT;
END_IF
CASE state OF
    ActionState.IDLE:
        // Idle

    ActionState.ADD_ELEMENT:
        // Add element
        eError := itfList.AddElement(itfElement := StringElementFactory.Create(sValue := sValue));
        IF eError = COL.COLLECTION_ERROR.NO_ERROR THEN
            state := ActionState.UPDATE_VISU;
        ELSE
            state := ActionState.IDLE;
        END_IF;

    ActionState.REMOVE_FIRST_ELEMENT:
        // Remove first element
        eError := itfList.RemoveElementAt(udiPosition := 0);
        IF eError = COL.COLLECTION_ERROR.NO_ERROR THEN
            state := ActionState.UPDATE_VISU;
        ELSE
            state := ActionState.IDLE;
        END_IF;

    ActionState.UPDATE_VISU:
        // Clear visu array
        FOR i := 0 TO 90 DO
            aStringValues[i] := '';
        END_FOR
        eError := itfList.ElementIterator(itfIterator := iterator);
        i := 0;
        WHILE iterator.HasNext() DO
            iterator.Next(itfElement => itfElement);
            // Cast IElement to IStringElement
            __QUERYINTERFACE(itfElement, itfStringElement);
            aStringValues[i] := itfStringElement.StringValue;
            i := i + 1;
        END_WHILE
        state := ActionState.IDLE;
END_CASE
```

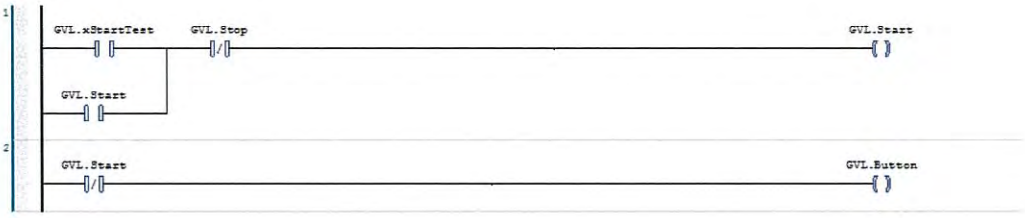
7. โค้ด LD ที่ใช้ในการกดปุ่มเลือกฟังก์ชั่น และแสดงกราฟฟิกต่างๆ

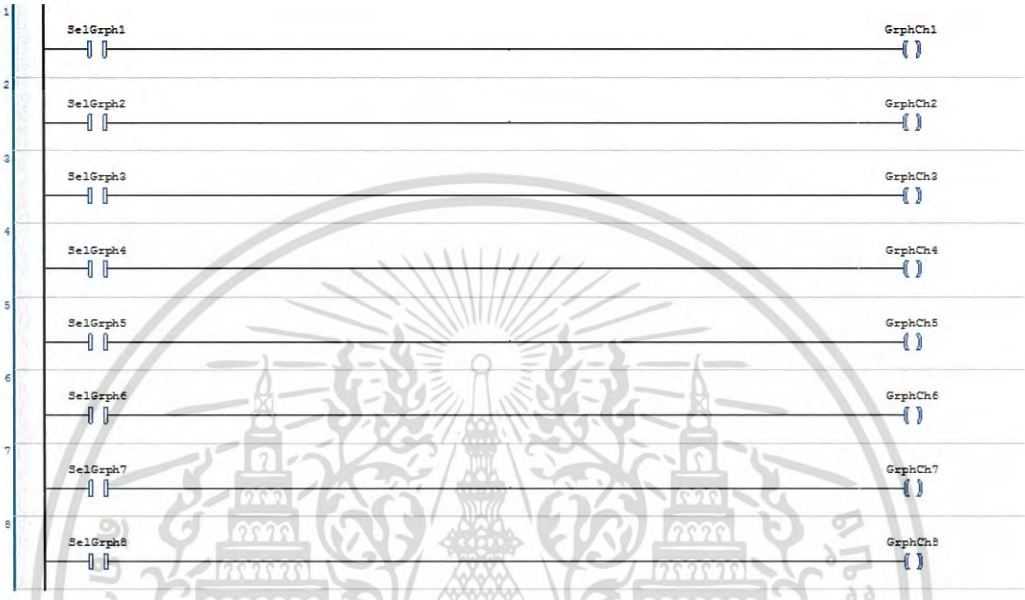





```
PROGRAM ReadingRate
VAR
    Pass : TIME;
END_VAR
```







8. โค้ดการส่งข้อมูลไปเก็บบนเว็บ Anyviz Cloud

```
PROGRAM PLC_PRG
VAR
        anyviz : Anyviz.AnyVizClient := (Projectid := 422 ,password := 'Bbbb');
END_VAR
```

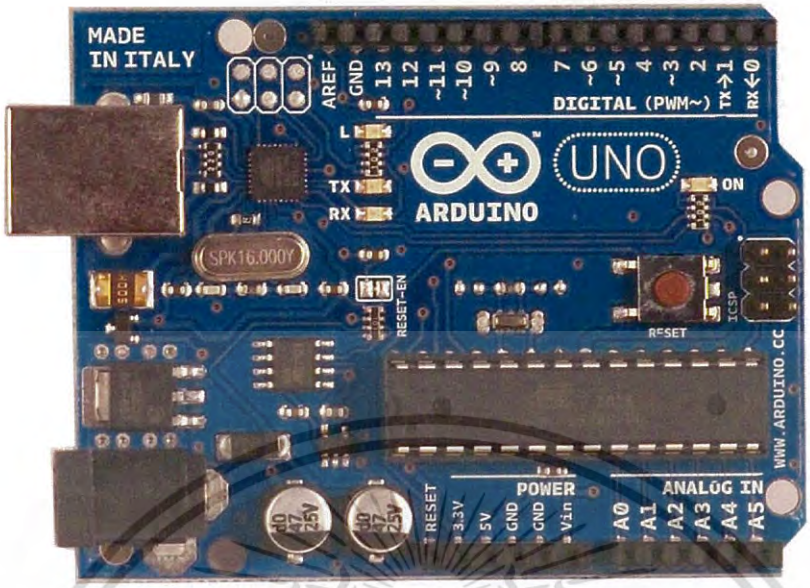
```
anyviz ();
```

ภาคผนวก ข.

เอกสารที่เกี่ยวข้อง

(อ้างอิงจาก https://static.raspberrypi.org)

Arduino UNO



CE

Product Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduno, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Index

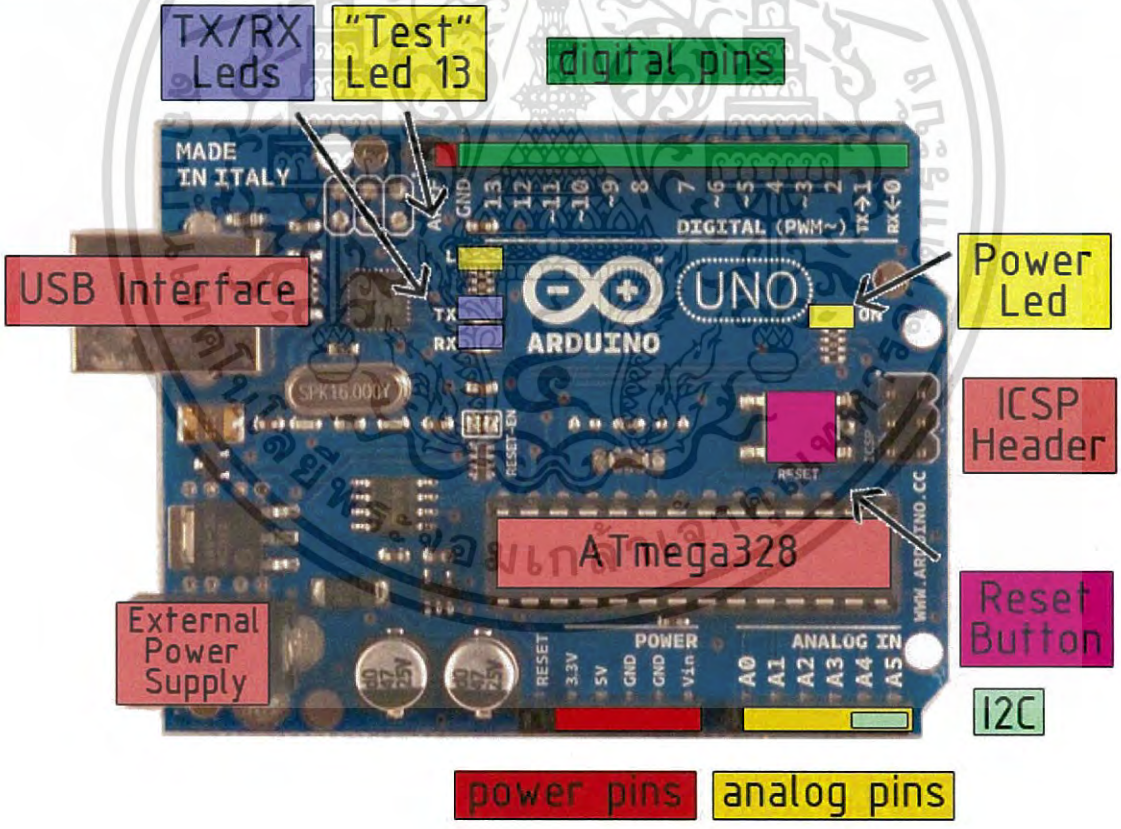
Technical Specification

EAGLE files: arduino-duemilanove-uno-design.zip Schematic: arduino-uno-schematic.pdf

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB of which 0.5 KB used by bootloader
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

the board



Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The Atmega328 has 32 KB of flash memory for storing code (of which 0,5 KB is used for the bootloader); It has also 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. TThese pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.

- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

- **I²C: 4 (SDA) and 5 (SCL).** Support I²C (TWI) communication using the Wire library.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and Atmega328 ports.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an *.inf file is required..

The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Uno's digital pins.

The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. To use the SPI communication, please see the ATmega328 datasheet.

Programming

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno w/ ATmega328" from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

The ATmega8U2 firmware source code is available . The ATmega8U2 is loaded with a DFU bootloader, which can be activated by connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader).

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

How to use Arduino

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).

Arduino is a cross-platoform program. You'll have to follow different instructions for your personal OS. Check on the Arduino site for the latest instructions. *http://arduino.cc/en/Guide/HomePage*

Linux Install

Windows Install

Mac Install

Once you have downloaded/unzipped the arduino IDE, you can Plug the Arduino to your PC via USB cable.

Blink led

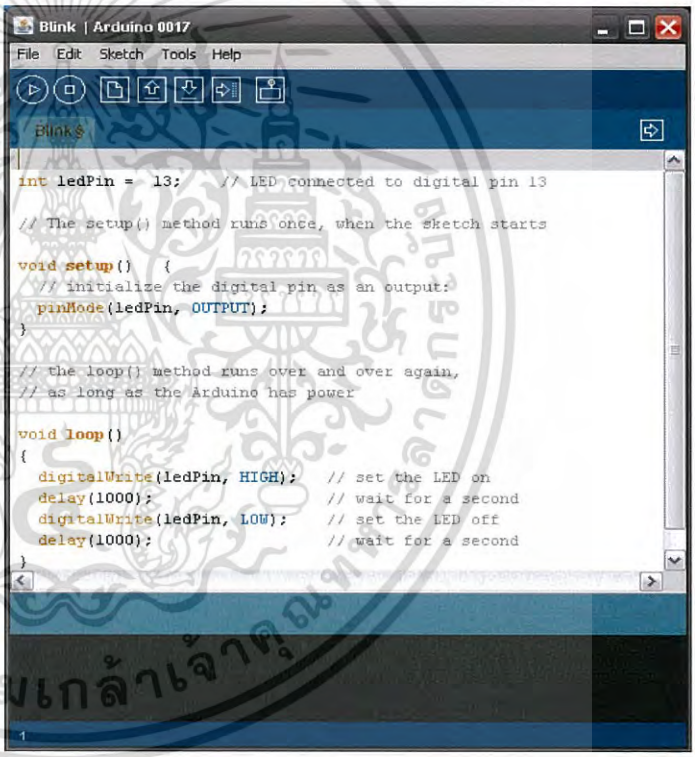
Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select

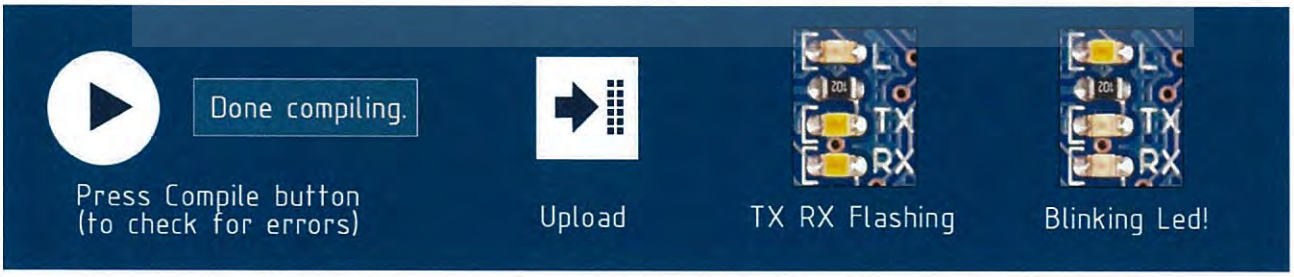
File>Sketchbook>
Arduino-0017>Examples>
Digital>Blink

Once you have your skecth you'll see something very close to the screenshot on the right.

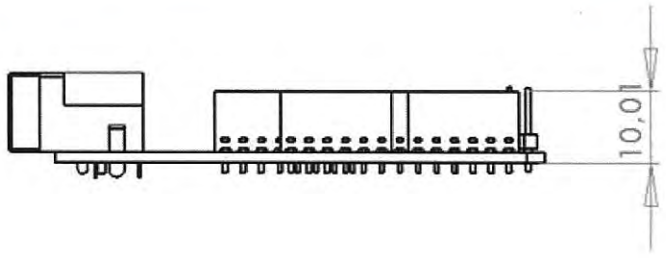
In **Tools>Board** select

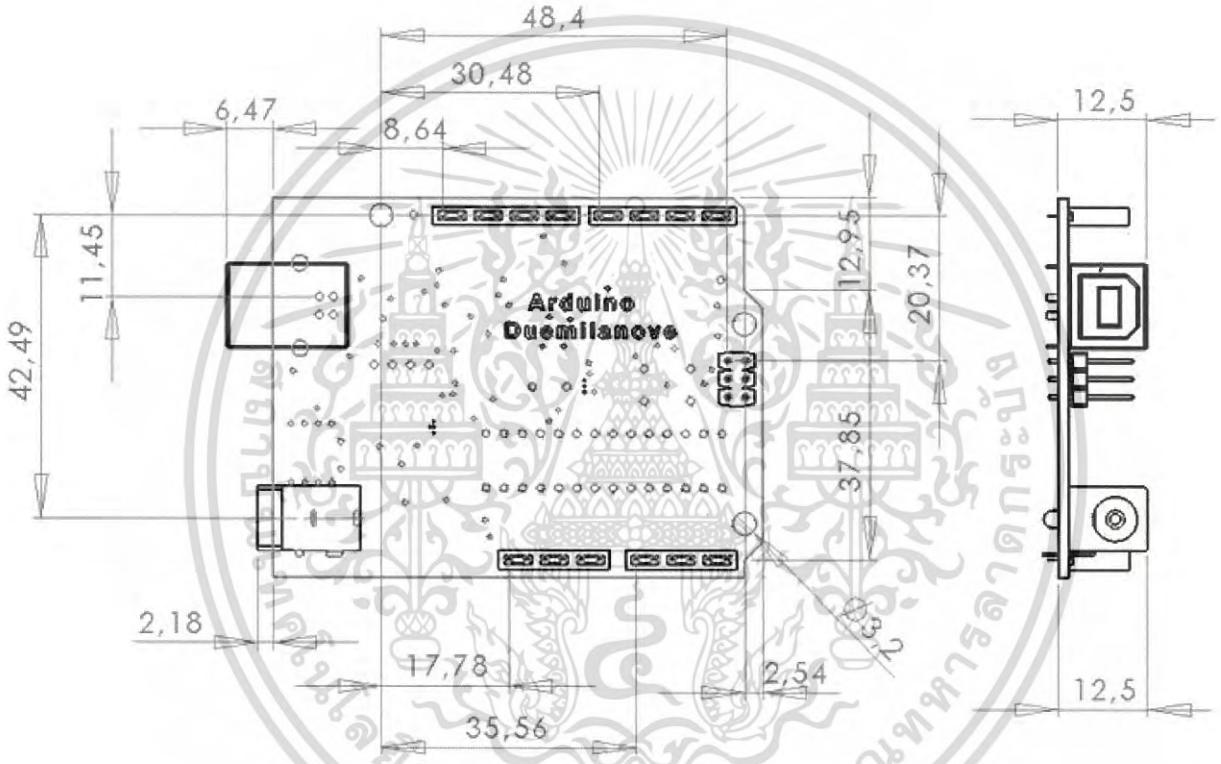
Now you have to go to
Tools>SerialPort
and select the right serial port, the one arduino is attached to.

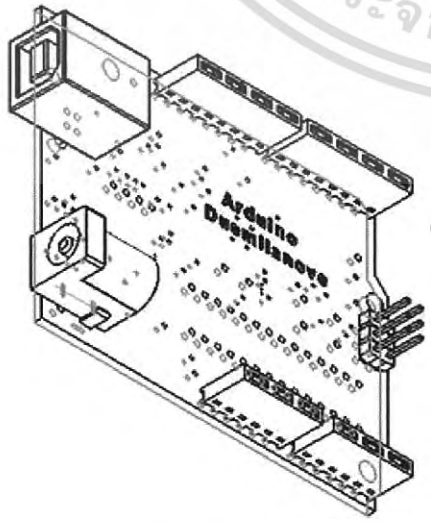




Dimensioned Drawing







Terms & Conditions

1. Warranties

1.1 The producer warrants that its products will conform to the Specifications. This warranty lasts for one (1) years from the date of the sale. The producer shall not be liable for any defects that are caused by neglect, misuse or mistreatment by the Customer, including improper installation or testing, or for any products that have been altered or modified in any way by a Customer. Moreover, The producer shall not be liable for any defects that result from Customer's design, specifications or instructions for such products. Testing and other quality control techniques are used to the extent the producer deems necessary.

1.2 If any products fail to conform to the warranty set forth above, the producer's sole liability shall be to replace such products. The producer's liability shall be limited to products that are determined by the producer not to conform to such warranty. If the producer elects to replace such products, the producer shall have a reasonable time to replacements. Replaced products shall be warranted for a new full warranty period.

1.3 EXCEPT AS SET FORTH ABOVE, PRODUCTS ARE PROVIDED "AS IS" AND "WITH ALL FAULTS." THE PRODUCER DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE

1.4 Customer agrees that prior to using any systems that include the producer products, Customer will test such systems and the functionality of the products as used in such systems. The producer may provide technical, applications or design advice, quality characterization, reliability data or other services. Customer acknowledges and agrees that providing these services shall not expand or otherwise alter the producer's warranties, as set forth above, and no additional obligations or liabilities shall arise from the producer providing such services.

1.5 The Arduino™ products are not authorized for use in safety-critical applications where a failure of the product would reasonably be expected to cause severe personal injury or death. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Arduino™ products are neither designed nor intended for use in military or aerospace applications or environments and for automotive applications or environment. Customer acknowledges and agrees that any such use of Arduino™ products which is solely at the Customer's risk, and that Customer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

1.6 Customer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products and any use of Arduino™ products in Customer's applications, notwithstanding any applications-related information or support that may be provided by the producer.

2. Indemnification

The Customer acknowledges and agrees to defend, indemnify and hold harmless the producer from and against any and all third-party losses, damages, liabilities and expenses it incurs to the extent directly caused by: (i) an actual breach by a Customer of the representation and warranties made under this terms and conditions or (ii) the gross negligence or willful misconduct by the Customer.

3. Consequential Damages Waiver

In no event the producer shall be liable to the Customer or any third parties for any special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of the products provided hereunder, regardless of whether the producer has been advised of the possibility of such damages. This section will survive the termination of the warranty period.

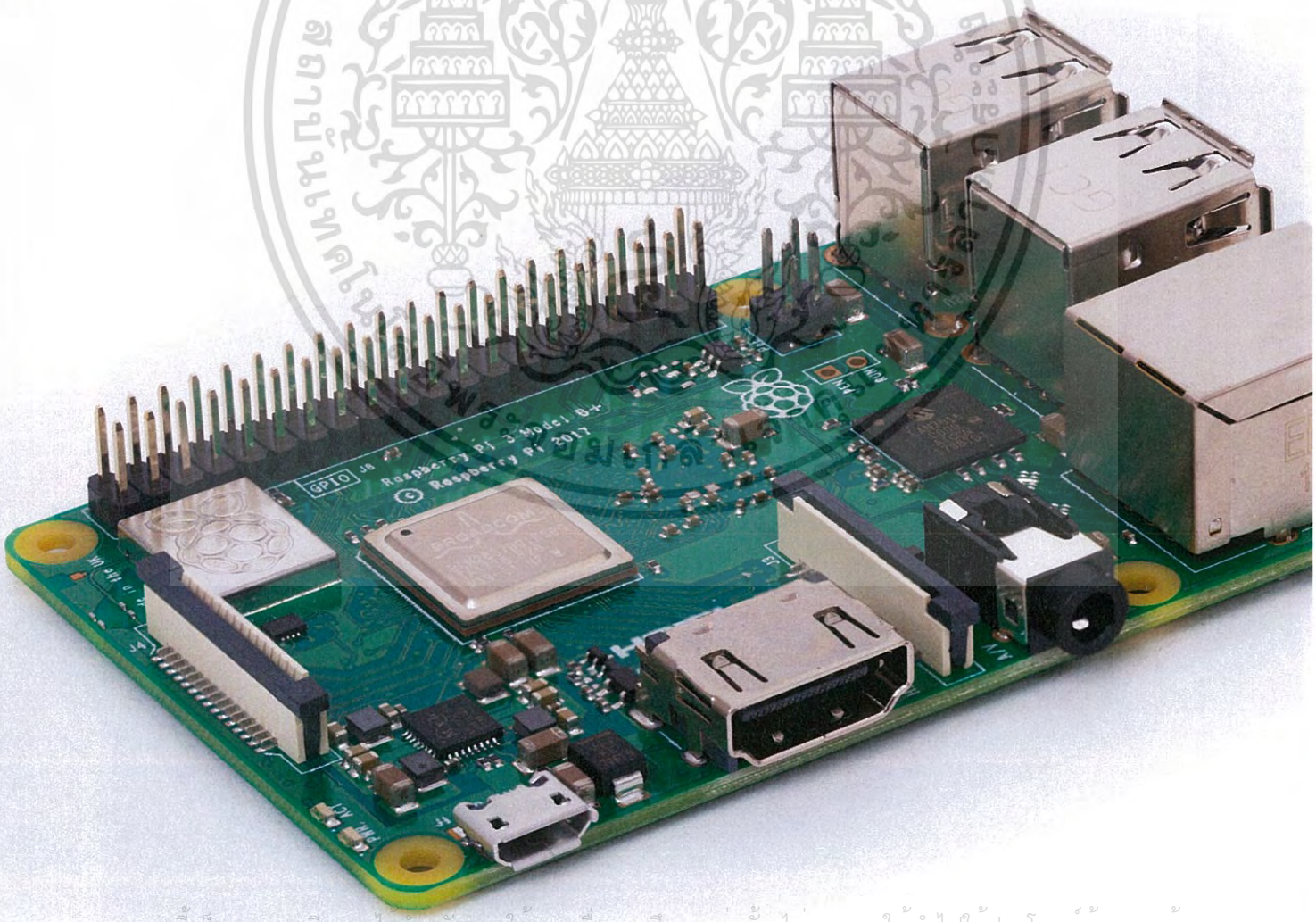
4. Changes to specifications

The producer may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." The producer reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.

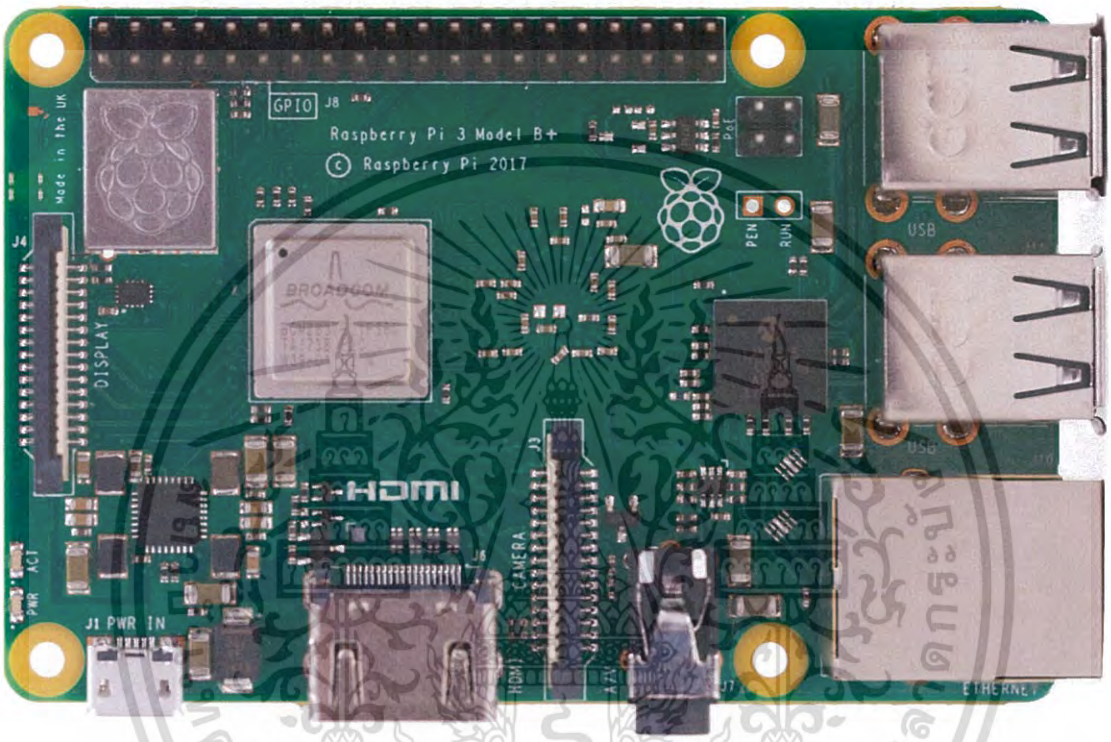
Enviromental Policies

The producer of Arduino™ has joined the Impatto Zero® policy of LifeGate.it. For each Arduino board produced is created / looked after half squared Km of Costa Rica's forest's.

Raspberry Pi 3 Model B+



Overview



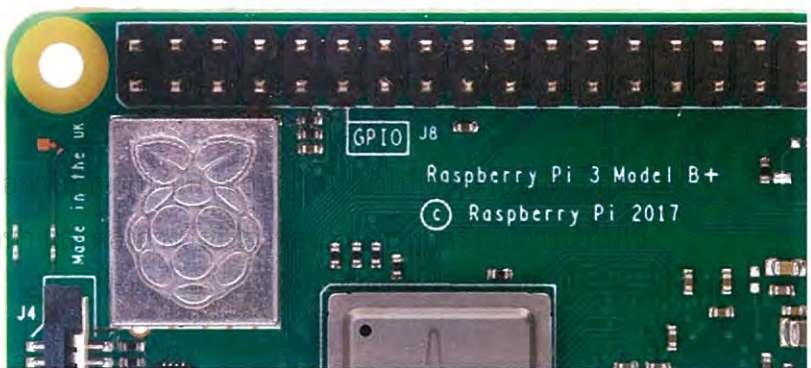
The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range, boasting a 64-bit quad core processor running at 1.4GHz, dual-band 2.4GHz and 5GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT

The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market.

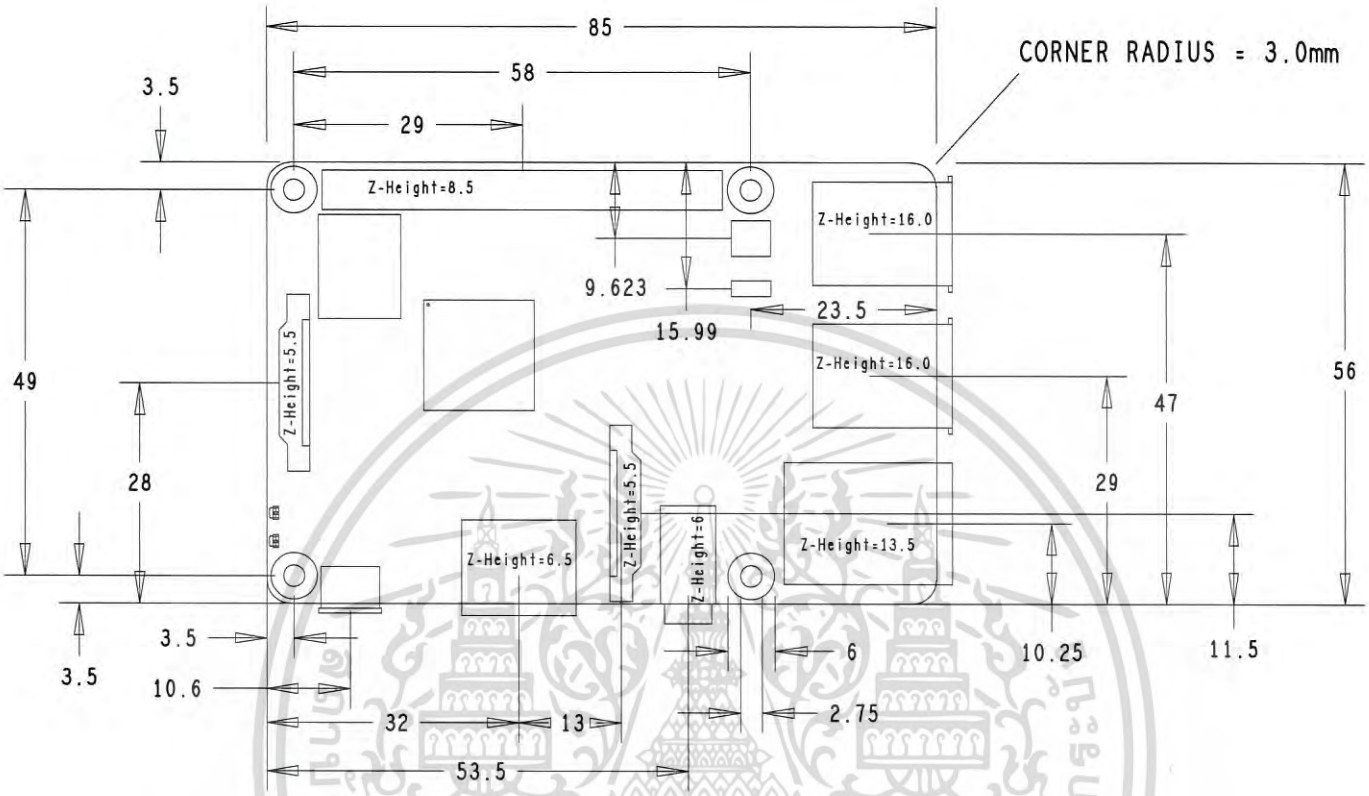
The Raspberry Pi 3 Model B+ maintains the same mechanical footprint as both the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B.

Specifications

Processor:	Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4GHz
Memory:	1GB LPDDR2 SDRAM
Connectivity:	■ 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE ■ Gigabit Ethernet over USB 2.0 (maximum throughput 300Mbps) ■ 4 × USB 2.0 ports
Access:	Extended 40-pin GPIO header
Video & sound:	■ 1 × full size HDMI ■ MIPI DSI display port ■ MIPI CSI camera port ■ 4 pole stereo output and composite video port
Multimedia:	H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics
SD card support:	Micro SD format for loading operating system and data storage
Input power:	■ 5V/2.5A DC via micro USB connector ■ 5V DC via GPIO header ■ Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
Environment:	Operating temperature, 0–50°C
Compliance:	For a full list of local and regional product approvals, please visit www.raspberrypi.org/products/raspberry-pi-3-model-b+
Production lifetime:	The Raspberry Pi 3 Model B+ will remain in production until at least January 2023.



Physical specifications



Warnings

- This product should only be connected to an external power supply rated at 5V/2.5A DC. Any external power supply used with the Raspberry Pi 3 Model B+ shall comply with relevant regulations and standards applicable in the country of intended use.
- This product should be operated in a well-ventilated environment and, if used inside a case, the case should not be covered.
- Whilst in use, this product should be placed on a stable, flat, non-conductive surface and should not be contacted by conductive items.
- The connection of incompatible devices to the GPIO connection may affect compliance, result in damage to the unit, and invalidate the warranty.
- All peripherals used with this product should comply with relevant standards for the country of use and be marked accordingly to ensure that safety and performance requirements are met. These articles include but are not limited to keyboards, monitors, and mice when used in conjunction with the Raspberry Pi.
- The cables and connectors of all peripherals used with this product must have adequate insulation so that relevant safety requirements are met.

Safety instructions

To avoid malfunction of or damage to this product, please observe the following:

- Do not expose to water or moisture, or place on a conductive surface whilst in operation.
- Do not expose to heat from any source; the Raspberry Pi 3 Model B+ is designed for reliable operation at normal ambient temperatures.
- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
- Whilst it is powered, avoid handling the printed circuit board, or only handle it by the edges to minimise the risk of electrostatic discharge damage.