เครื่องวัดไร้สายสำหรับฟาร์มสัตว์ปีก

WIRELESS SENSOR FOR POULTRY FARM

นางสาวเสาร์วรินท์ วงศ์ภิวัฒนา

MISS SAOWARIN WONGPIWATTANA

นางสาวอนัญญา เกศวรกุล

MISS ANUNYA KETVORAKUL

นางสาวอัจฉราภา ชลเขต

MISS AUTCHARAPA CHONLAKHET

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมสารสนเทศ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2553

WIRELESS SENSOR FOR POULTRY FARM

MISS SAOWARIN WONGPIWATTANA

MISS ANUNYA KETVORAKUL

MISS AUTCHARAPA CHONLAKHET

THIS THESIS IS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INFORMATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2010

หัวข้อปริญญานิพนธ์	เครื่องวัดไร้สายสำหรับฟาร์มสัตว์ปีก	
รายชื่อนักศึกษา	นางสาวเสาร์วรินท์ วงศ์ภิวัฒนา	รหัสนักศึกษา 50011791
	นางสาวอนัญญา เกศวรกุล	รหัสนักศึกษา 50011833
	นางสาวอัจฉราภา ชลเขต	รหัสนักศึกษา 50011917
ปริญญา	วิศวกรรมศาสตรบัณฑิต	
สาขาวิชา	วิศวกรรมสารสนเทศ	
พ.ศ.	2553	
อาจารย์ที่ปรึกษาปริญญานิพนธ์	อาจารย์มนชนก ศรีเสือขาม	
	รศ.ดร.อรรถสิทธิ์ หล่าสกุล	

ปริญญานิพนธ์ฉบับนี้ ได้รับการอนุมัติให้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรวิศวกรรมศาสตรบัณฑิต คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง

.....

(อาจารย์มนชนก ศรีเสือขาม)

.....

(รศ.ดร.อรรถสิทธิ์ หล่าสกุล)

อาจารย์ผู้ควบคุมปริญญานิพนธ์

หัวข้อปริญญานิพนธ์	เครื่องวัดไร้สายสำหรับฟาร์มสัตว์ปีก	
รายชื่อนักศึกษา	นางสาวเสาร์วรินท์ วงศ์ภิวัฒนา	รหัสนักศึกษา 50011791
	นางสาวอนัญญา เกศวรกุล	รหัสนักศึกษา 50011833
	นางสาวอัจฉราภา ชลเขต	รหัสนักศึกษา 50011917
ปริญญา	วิศวกรรมศาสตรบัณฑิต	
สาขาวิชา	วิศวกรรมสารสนเทศ	
พ.ศ.	2553	
อาจารย์ที่ปรึกษาปริญญานิพนธ์	อาจารย์มนชนก ศรีเสือขาม	
	รศ.ดร.อรรถสิทธิ์ หล่าสกุล	

บทคัดย่อ

โครงงานนี้เป็นการนำเสนอเครื่องวัดไร้สายและเครื่องควบคุมอุปกรณ์ไฟฟ้าแบบไร้สายสำหรับฟาร์มสัตว์ปีก โดยนำเอาเทคโนโลยีของสัญญาณ Wireless ซึ่งเป็นเครือข่ายไร้สาย จากโมดูล ZigBee 2.4 GHz มาประยุกต์ใช้ในการส่งผ่านข้อมูลโดยใช้ computer ส่งคำสั่งให้โมดูล ZigBee ตัวส่ง จากนั้นสัญญาณไร้สายจะส่งไปหา ZigBee ตัวรับ แล้วส่งคำสั่งควบคุมการทำงานโดยใช้ไมโครคอนโทรลเลอร์ในการควบคุมระบบเซนเซอร์วัดอุณหภูมิและอุปกรณ์ไฟฟ้าผ่านรีเลย์สวิตช์ ส่วนผลที่ได้จากเซนเซอร์เครื่องวัดจะส่งข้อมูลอุณหภูมิในช่วงเวลาต่างๆ ตามที่ต้องการมาแสดงผ่านทางหน้าจอ GUI และเก็บบันทึกไว้ในฐานข้อมูลผ่านโปรแกรมประยุกต์ซึ่งผู้ใช้สามารถสั่งการเครื่องควบคุมอุณหภูมิและอุปกรณ์ไฟฟ้าแบบไร้สายได้อย่างสะดวกโดยไม่ต้องเข้าไปในฟาร์มเอง จุดเด่นของโครงงานนี้ ก็คือการใช้งาน ZigBee ทำงานแบบทวนสัญญาณให้แก่กัน ซึ่งจะทำให้สามารถส่งผ่านข้อมูลระหว่างผู้ใช้กับตัวโมดูลเซนเซอร์ตัวที่ต้องการติดต่อได้แม้จะอยู่อย่างห่างไกลกันเกินสัญญาณระหว่างทั้งคู่ได้ ทำให้สามารถใช้งานได้ในพื้นที่ฟาร์มทั่วไปแม้จะเป็นพื้นที่ๆ กว้างขวางก็ตาม ซึ่งการทำเช่นนี้ได้ทำให้ผู้ดูแลฟาร์มสามารถตรวจสอบการทำงานของอุปกรณ์ต่างๆ ในฟาร์มขนาดใหญ่ได้สะดวกขึ้น

Thesis Title	Wireless Sensor for Poultry Farm	
Student	Ms. Saowarin Wongpiwattana	Student ID. 50011791
	Ms. Ananya Kesvorakul	Student ID. 50011833
	Ms. Autcharapa Chonlakhet	Student ID. 50011917
Degree	Bachelor of Engineering	
Program	Information Engineering	
Year	2010	
Thesis Advisor	Miss Monchanok SrisuaKham	
	Assoc. Prof. Dr. Attasit Lasakul	

ABSTRACT

This Project presents the wireless sensor and the control wireless system of electric equipments for poultry farm. Supplying the wireless technology from ZigBee module 2.4 GHz to transfer the control data from the computer to ZigBee module Transmitter. The command transit to ZigBee module receiver. The microcontroller received command to control temperature sensor and electric equipments by relay switch. The results from the wireless sensor sends temperature data through time. According to the display screen via GUI and stored in a database. The advantage point is that, long ranges communicate between User and module sensor (over the specification of Zigbee). By setting all modules (Zigbee) operated as repeater to each other. So, it can be used for wider area. Users can command a temperature controller and a wireless electric device without having to visit the farm itself. Users can check the operation of equipment according to the standards easier. Features of this project. Zigbee is to use a repeater to work together. It enables data transmission between the user with a sensor module to connect even when they are very far apart over the signal between both. Made available in an extensive farm area.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดีทั้งนี้ต้องขอขอบพระคุณท่านอาจารย์ในสาขาวิชาวิศวกรรมสารสนเทศทุกท่านที่ได้อบรมสั่งสอนรวมทั้งให้ความรู้ และขอขอบพระคุณ รศ.ดร.อรรถสิทธิ์ หล่าสกุล และ อาจารย์มนชนก ศรีเสือขาม อาจารย์ที่ปรึกษาปริญญานิพนธ์ ที่ผลักดันและให้โอกาสโครงงานนี้ได้เป็นจริงขึ้นมา โดยคอยช่วยเหลือชี้แนวทางและให้คำปรึกษาในการทำงาน รวมถึงขั้นตอนในการทำงานตลอดจนทุนทรัพย์ที่ใช้ในการทำโครงงาน

ขอขอบคุณเพื่อนนักศึกษาและผู้มีส่วนรวมในโครงการนี้ทุกท่านที่มิได้เอ่ยนามในที่นี้ ซึ่งได้มีโอกาสทำงานร่วมกันให้กำลังใจในการทำงาน รวมทั้งคอยให้ความช่วยเหลือในเวลาที่เกิดปัญหา

ขอกราบขอบพระคุณบิดา มารดา ของผู้ดำเนินการวิจัยทุกท่านผู้คอยเป็นกำลังใจ เข้าใจให้โอกาสและสนับสนุนในทุก ๆ ด้าน โดยเฉพาะด้านทุนทรัพย์ ขอขอบพระคุณทุก ๆ ท่านไว้ ณ ที่นี้

คณะผู้จัดทำ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันมีการนำเทคโนโลยีต่างๆ มาใช้กับการเกษตรเพื่อช่วยเพิ่มความสะดวกสบายในการควบคุมดูแลผลผลิตมากขึ้น เช่น ควบคุมสภาพอากาศ อุณหภูมิ ความชื้น เป็นต้น ซึ่งปัจจัยเหล่านี้เป็นปัจจัยหลัก ที่มีผลกระทบโดยตรงต่อคุณภาพของผลผลิตนั้นๆ โดยเฉพาะการเจริญเติบโตในผลผลิตที่เป็นพืช และสัตว์เลี้ยง เนื่องจากสิ่งต่างๆ ที่แวดล้อมอยู่นั้นสามารถเกิดการเปลี่ยนแปลงได้ตลอดเวลาอีกทั้งยังไม่รู้ว่าจะเกิดขึ้นเมื่อใดหรือเวลาใด ดังนั้นการรู้ถึงค่าสภาพสิ่งแวดล้อมต่างๆ จึงเป็นสิ่งที่ช่วยให้การควบคุมดูแลผลผลิตง่ายยิ่งขึ้น โดยค่าที่ได้นั้นต้องมีความแม่นยำถูกต้อง เพื่อควบคุมการผลิตให้ได้คุณภาพและเหมาะสมมากที่สุด ซึ่งในปัจจุบันพบว่ามีอุปกรณ์สำหรับใช้ตรวจวัดที่มีขายสำเร็จรูปอยู่ในท้องตลาดมากมายให้เลือกหลายชนิด แต่ก็มีต้นทุนสูง ราคาแพง และส่วนใหญ่ก็จะใช้พลังงานสิ้นเปลือง อีกทั้งยังไม่สามารถเคลื่อนย้ายได้

จากปัญหาดังกล่าว จึงเล็งเห็นถึงการนำประโยชน์ของเทคโนโลยีระบบเครือข่ายไร้สาย ซึ่งเป็นเครื่องมือที่สามารถนำมาใช้ในการรับส่งข้อมูลจากระบบเครื่องวัดเซนเซอร์ต่างๆที่ได้ติดตั้งไว้ ให้มาแสดงผลยังหน้าจอสำหรับควบคุม ซึ่งจะช่วยให้ผู้ใช้สามารถดูข้อมูลได้จากทุกจุดทุกเวลาตามต้องการ โดยที่ไม่ต้องเข้าไปในฟาร์มเอง

โครงงานนี้จึงนำแนวคิดในการใช้เทคโนโลยีระบบเครือข่ายไร้สายมาประยุกต์ใช้กับฟาร์มสัตว์ปีก โดยเลือกใช้ระบบสื่อสารไร้สายแบบ ZigBee ซึ่งเป็นเทคโนโลยีใหม่และมีข้อดีในหลายๆ ด้าน ที่สำคัญคือ มีราคาถูก ใช้พลังงานต่ำ และสามารถเคลื่อนย้ายเซนเซอร์ไปวัดตามที่ต่างๆ ได้อย่างอิสระ โดยไม่ต้องใช้สายต่างๆ เป็นตัวเชื่อมโยงในระบบเครือข่าย ส่วนค่าที่ได้ก็มีความถูกต้องตามความเป็นจริงแบบ Real-Time และ Non-Real-Time ที่เป็นประโยชน์ต่อการพัฒนาระบบควบคุมสภาพอากาศ ระบบแจ้งเตือน อีกทั้งยังเป็นแนวทางในการพัฒนาระบบเซนเซอร์อื่นๆ ให้ประยุกต์ใช้กับเทคโนโลยีระบบเครือข่ายไร้สายอย่างอุปกรณ์ ZigBee ได้ต่อไปในอนาคตอีกด้วย

1.2 วัตถุประสงค์

1.2.1 เพื่อศึกษา วิเคราะห์และออกแบบเครื่องวัดไร้สายสำหรับฟาร์มสัตว์ปีก โดยนำ ZigBee ซึ่งเป็นระบบเครือข่ายไร้สายมาประยุกต์ใช้ในการรับส่งข้อมูลที่ได้จากเซนเซอร์เครื่องวัดที่ติดตั้งไว้ภายในฟาร์ม ให้สามารถรับส่งข้อมูลได้

1.2.2 เพื่อศึกษาการใช้งานเทคโนโลยีไมโครคอนโทรลเลอร์ dsPIC30F4011 ในการส่งคำสั่งให้รับส่งข้อมูลตามที่ต้องการได้

1.2.3 ออกแบบโปรแกรมส่วน GUI เพื่อทำหน้าที่ติดต่อกับผู้ใช้ ในการสั่งงานควบคุมต่างๆ เช่น การเรียกอ่านผลอุณหภูมิ การสั่งรีเลย์ให้ทำงาน และในการทำงานโหมดต่างๆ (Auto/Manual) โดยมีการจัดรูปแบบคำสั่งเพื่อติดต่อกับตัววัดที่ต้องการติดต่อด้วยอย่างอัตโนมัติ

1.3 ขอบเขตของโครงงาน

1.3.1 สร้างเครื่องวัดไร้สายสำหรับฟาร์มสัตว์ปีกที่เป็นโมดูลต้นแบบหนึ่งตัวให้มีส่วนต่างๆดังนี้

- ส่วนของไมโครคอนโทรลเลอร์ dsPIC4011
- ส่วนของการติดต่อ XBee-PRO
- ส่วนของการติดต่อตัววัดอุณหภูมิจำนวนหนึ่งช่อง
- ส่วนของการติดต่ออุปกรณ์ตัดต่อไฟฟ้ารีเลย์จำนวนสองช่อง

1.3.2 สามารถส่งคำสั่งควบคุมการปฏิบัติงานจากเครื่อง Main ผ่านเครือข่ายไร้สายไปยังเครื่อง Client ได้ และสามารถส่งข้อมูลค่าที่วัดได้จากเครื่อง Client ไปยังเครื่อง Main ได้

1.3.3 สร้างโปรแกรมควบคุม Microcontroller ด้วย โปรแกรม MPLAB

1.3.4 สร้าง User Interface ซึ่งเป็นลักษณะของ GUI ด้วย โปรแกรม Visual C#2010

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 สามารถนำชุดอุปกรณ์ที่ทำขึ้นไปใช้งานจริงในฟาร์มลักษณะต่างๆ ได้

1.4.2 นำโปรแกรมที่เชื่อมโยงเครือข่ายของ XBee-PRO แต่ละตัวมาทำหน้าที่เป็นตัวกลางส่งผ่านคำสั่งได้ โดยแต่ละตัวของ client มีหน้าที่ 3 อย่าง คือ ทำหน้าที่ส่งต่อคำสั่ง , ส่งต่อข้อมูลกลับ , อ่านค่าอุณหภูมิและ อ่าน/สั่ง สถานะรีเลย์ ของตนเอง

1.4.3 ช่วยเพิ่มความสะดวกสบายในการควบคุมดูแลฟาร์มให้กับผู้ผลิต

1.4.4 นำระบบเครือข่ายการสื่อสารไร้สายไปประยุกต์ใช้กับงานด้านสิ่งแวดล้อม และด้านการรักษาความปลอดภัยได้

1.5 ขั้นตอนการดำเนินโครงงาน

ตารางที่ 1.1 ขั้นตอนการดำเนินงาน

ลำดับ	ขั้นตอนการดำเนินงาน	ปี 2553							ปี 2554		
		มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.
1.	กำหนดขอบเขต วิเคราะห์ปัญหาและออกแบบโครงงาน										
2.	ศึกษาการทำงานของ ZigBee เพื่อหาความเหมาะสมที่จะใช้ในโครงงาน										
3.	ออกแบบ Software ให้ RS232 ติดต่อกับ เครื่อง PC ได้และออกแบบ GUI ให้ติดต่อ dsPIC30F4011 กับ RS232 ได้										
4.	ต่อไอซี DS1820 ที่ใช้วัดอุณหภูมิกับวงจร dsPIC30F4011 ให้สามารถแสดงผลบนจอ GUI ได้และสั่งเปิด-ปิดอุปกรณ์รีเลย์ได้										
5.	ออกแบบและสร้างโมดูลต้นแบบ										
6.	ออกแบบและทดสอบ Software ให้รับ-ส่งข้อมูลผ่านโมดูลต้นแบบได้ทั้ง 3 โหมด										
7.	ทดสอบการทำงานจริงและปรับปรุงแก้ไขข้อบกพร่อง										
8.	สรุปผลและจัดทำรูปเล่มรายงาน										

บทที่ 2
ทฤษฎีพื้นฐานที่ใช้

2.1 เทคโนโลยี ZigBee

เทคโนโลยี ZigBee เป็นการสื่อสารที่ออกแบบขึ้นสำหรับการสื่อสารในเครือข่ายเซ็นเซอร์แบบไร้สาย (Wireless Sensor Network) โดยชื่อ ZigBee ได้มาจากพฤติกรรมการสื่อสารของผึ้งโดยผึ้งจะบินแบบซิกแซ็ก และจะให้ข้อมูลข่าวสารระหว่างผึ้งด้วยกันที่เกี่ยวกับตำแหน่ง ระยะทาง และทิศทางของอาหารที่พวกมันกำลังหาอยู่โดยเริ่มจากการกำหนดมาตรฐานการรับ-ส่งข้อมูลแบบ IEEE 802.15.4 ที่เน้นการสื่อสารแบบประหยัดพลังงาน ความเร็วการรับส่งข้อมูลต่ำและมีราคาถูก การสื่อสารลักษณะนี้ได้ถูกนำมาใช้สำหรับการสื่อสารระหว่างเครื่องตรวจวัดหรือเซ็นเซอร์ที่ต้องการสื่อสารแบบไร้สายเพื่อลดความยุ่งยากซับซ้อนสำหรับการติดตั้ง เช่น บริเวณโรงงานหนึ่งๆ อาจจะต้องใช้จำนวนเซ็นเซอร์ปริมาณมากๆ และเครื่องรับส่งที่มีราคาถูกและประหยัดพลังงาน โดยการสื่อสารระยะใกล้แบบ ZigBee แตกต่างจากการสื่อสารแบบ Bluetooth ดังนี้

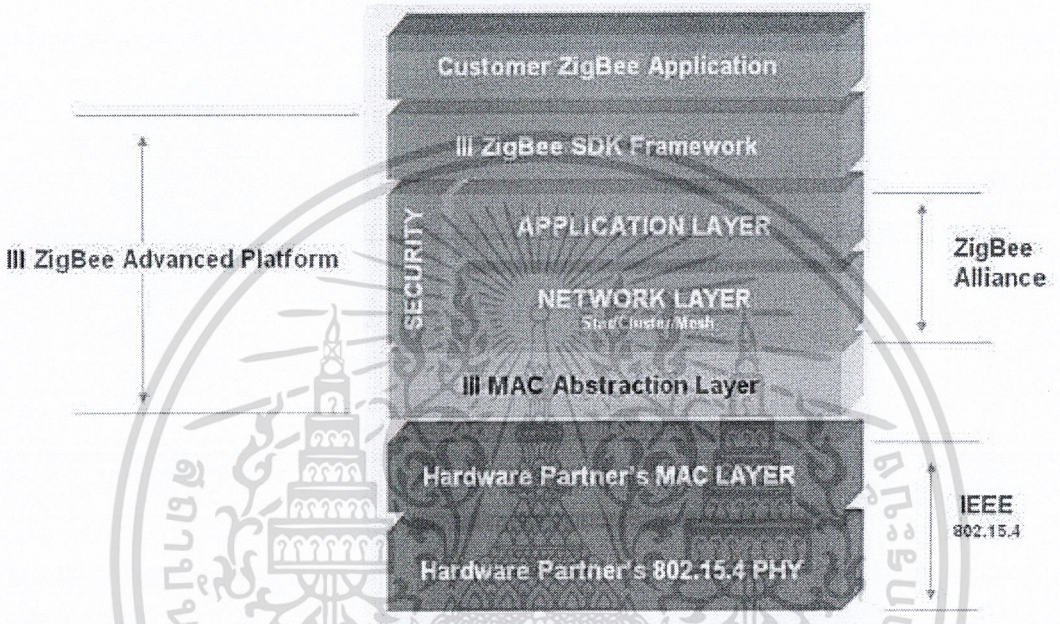
- มีการเชื่อมต่ออย่างซับซ้อนเพื่อรองรับการเชื่อมต่อสำหรับเครือข่ายขนาดใหญ่
- การใช้งานแบบประหยัดพลังงานเพื่อการใช้งานได้ยาวนานจากพลังงานแบตเตอรี่
- การสื่อสารระยะใกล้ในระยะ 10 - 100 เมตร
- เหมาะสำหรับการเฝ้าระวัง (Monitor) และการควบคุม (Control) ใช้งานอุตสาหกรรม งานสิ่งแวดล้อม งานก่อสร้างและงานทางการแพทย์
- เน้นการสื่อสารข้อมูลที่ความเร็วประมาณ 125 - 250 กิโลบิตต่อวินาที (kbps)

2.2.1 มาตรฐาน IEEE 802.15.4

มาตรฐาน IEEE 802.15.4 กำหนดขึ้นสำหรับการรับส่งข้อมูลเบื้องต้นในวงจรเครื่องรับส่งวิทยุ (Physical Layer) และการควบคุมการรับส่ง (Link Layer) ดังต่อไปนี้ การสื่อสารใช้คลื่นวิทยุความถี่ 2.4 กิกะเฮิรตซ์ (GHz) แบ่งออกเป็น 16 ช่องสัญญาณ สัญญาณละ 5 เมกะเฮิรตซ์ (MHz) สำหรับความถี่ 900 เมกะเฮิรตซ์ (MHz) แบ่งออกเป็น 10 ช่องสัญญาณ สัญญาณละ 2 เมกะเฮิรตซ์ (MHz) ใช้การผสมสัญญาณ (Modulation) แบบ Offset Quadrature Phase Shift Keying (Offset-QPSK) และใช้การแก้ปัญหาสัญญาณรบกวนแบบ Direct Sequence Spread Spectrum (DSSS) ที่มีอัตราการสเปรดดิ่ง (Spreading) 2 ล้าน chip/sec ควบคุมการรับส่งข้อมูล โดยใช้โปรโตคอลแบบ Carrier Sense Multiple Access / Collision Avoidance

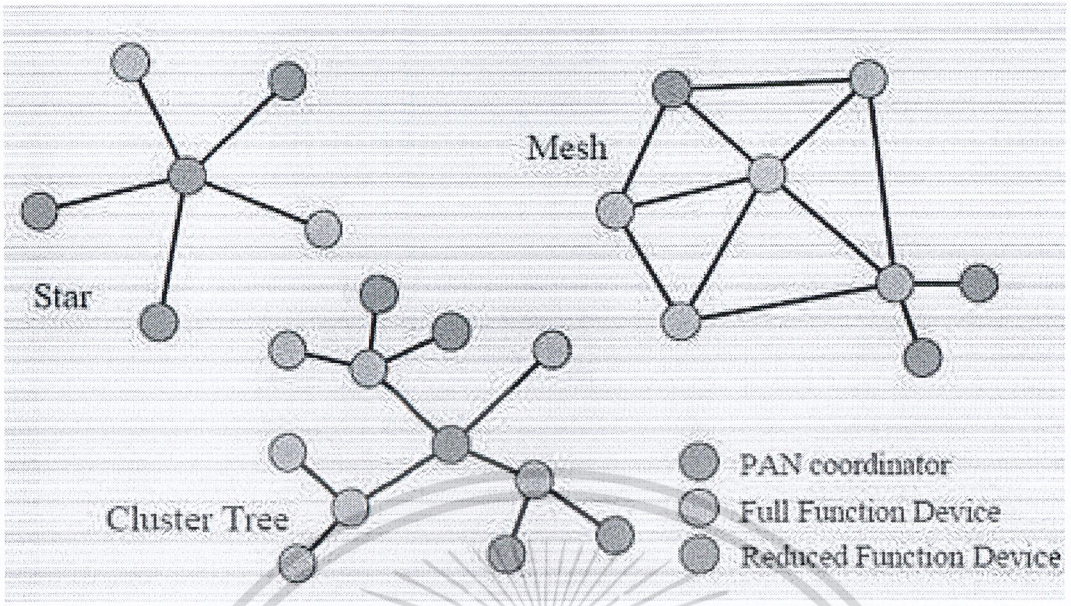
(CSMA/CA) และเพื่อให้การสื่อสารเครือข่ายเซ็นเซอร์ไร้สายเป็นมาตรฐานเดียวกัน จึงกำหนดมาตรฐานเพิ่มสำหรับการเชื่อมต่อเป็นเครือข่าย (Network Layer) และการนำไปใช้งาน (Application Layer) ร่วมกับมาตรฐาน IEEE 802.15.4 เป็นมาตรฐานใหม่ที่กำหนดโดยองค์กร ZigBee Alliance

2.1.2 โครงสร้างโปรโตคอล ZigBee



รูปที่ 2.1 แสดง Layer ชั้นต่างๆ ของโปรโตคอล ZigBee

การเชื่อมต่อเป็นโครงข่ายของเครือข่ายเซ็นเซอร์ไร้สาย กำหนดโดยมาตรฐาน ZigBee มีอยู่ 3 รูปแบบ ได้แก่ แบบดาว (Star) แบบระดับเดียว (Peer-to-Peer) และแบบโครงตาข่าย (Mesh) ซึ่งการเชื่อมต่อแบบ Star เหมือนการเชื่อมต่อแบบโครงข่ายจิ๋ว (Piconet) สำหรับเครือข่าย Bluetooth โดยการสื่อสารระหว่างอุปกรณ์เซ็นเซอร์ไร้สายสามารถทำได้โดยผ่านอุปกรณ์ Personal Area Network (PAN) Coordinator หรือ Gateway Node สำหรับการเชื่อมต่อแบบระดับเดียวเป็นการเชื่อมต่อเพื่อขยายโครงข่ายให้กว้างออกไป โดยในการเชื่อมต่อนั้นจะต้องมี PAN Coordinator ซึ่งเชื่อมต่ออยู่กับอุปกรณ์เซ็นเซอร์ที่มีความสามารถเต็ม (Full-Function Device) ในการหาเส้นทาง (Routing) และอุปกรณ์เซ็นเซอร์แบบที่มีความสามารถลดลง (Reduced-Function Device) จะเชื่อมต่อกับอุปกรณ์ที่มีความสามารถเต็มอีกทีหนึ่ง ซึ่งอุปกรณ์ที่มีความสามารถเต็มเทียบเท่าได้กับอุปกรณ์ Bluetooth แบบ Master ที่มีหน้าที่เชื่อมต่อกับ Master ตัวอื่นๆ เพื่อให้เกิดเครือข่ายที่ใหญ่ขึ้น



รูปที่ 2.2 รูปแบบต่างๆ ของโทโปโลยีในการเชื่อมต่อของเครือข่าย

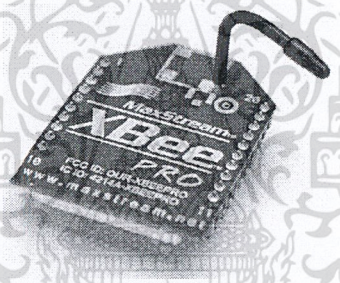
ส่วนสำคัญสำหรับเครือข่ายเซ็นเซอร์ไร้สายคือกลไกการประหยัดพลังงานซึ่งตัวอุปกรณ์เซ็นเซอร์สามารถกำหนดระยะเวลาในการเข้าสู่การหลับหรือพักการทำงาน (Sleep Mode) โดยตัวอุปกรณ์จะทำหน้าที่ ลดการใช้พลังงานให้เหลือน้อยที่สุดเพื่อเป็นการประหยัดพลังงาน การกำหนดการหลับนี้ทำได้โดยการร้องขอไปที่อุปกรณ์ Full Function Device เพื่อบอกระยะเวลาที่จะทำการหลับ และเมื่อมีการส่งข้อมูลจากอุปกรณ์ตัวอื่นมายังอุปกรณ์ที่หลับอยู่ อุปกรณ์ Full Function Device จะเก็บข้อมูลไว้ให้ชั่วคราวและถามหาอุปกรณ์ตัวนั้นเป็นระยะเมื่ออุปกรณ์ตัวนั้นตื่นหรือพร้อมทำงานต่อจะได้รับการถามหาอุปกรณ์ตัวนั้นจึงส่งการร้องขอข้อมูลที่ได้เก็บไว้ให้ และจึงทำการรับส่งข้อมูลจนได้รับข้อมูลครบสมบูรณ์ต่อไป

การประยุกต์ใช้งาน ZigBee นั้นจะแบ่งแยกตามประเภทของข้อมูลข่าวสาร ที่มีอยู่ 3 แบบ คือ ข้อมูลแบบ Periodic ข้อมูลเป็นช่วงเวลา โปรแกรมสามารถควบคุมอัตราการส่งและตัวตรวจจับสัญญาณกระตุ้น เช็คข้อมูลและทำให้ข้อมูลไม่เคลื่อนไหว ใช้สำหรับเซนเซอร์และมิเตอร์ข้อมูลแบบ Intermittent เป็นลักษณะที่มีการส่งผ่านข้อมูลเมื่อมีการใช้งาน เช่น สวิตซ์ไฟ และข้อมูลแบบ Repetitive low latency ใช้ในงานที่ต้องการ Latency น้อยๆ โดยการสื่อสารจะใช้วิธีจัดสรรช่องเวลาและสามารถใช้กลไกแบบ GTS เพื่อรับประกันคุณภาพของการบริการนำไปใช้ในงาน เช่น เมาส์ไร้สาย เป็นต้น

ตารางที่ 2.1 แถบความถี่ และอัตราข้อมูล

PHY (MHz)	Frequency band (MHz)	Spreading parameters		Data parameters		
		Chip rate (kchip/s)	Modulation	Bit rate (kb/s)	Symbol rate (ksymbol/s)	Symbols
868/915	868–868.6	300	BPSK	20	20	Binary
	902–928	600	BPSK	40	40	Binary
868/915 (optional)	868–868.6	400	ASK	250	12.5	20-bit PSSS
	902–928	1600	ASK	250	50	5-bit PSSS
868/915 (optional)	868–868.6	400	O-QPSK	100	25	16-ary Orthogonal
	902–928	1000	O-QPSK	250	62.5	16-ary Orthogonal
2450	2400–2483.5	2000	O-QPSK	250	62.5	16-ary Orthogonal

2.2 โมดูลสื่อสารข้อมูลไร้สาย XBee - PRO



รูปที่ 2.3 โมดูล XBee-PRO แบบ Chip Antenna

2.2.1 คุณสมบัติทางเทคนิค

- คุณสมบัติโดยทั่วไป

  - ความถี่ในการทำงาน : 2.4 GHz
  - สายอากาศ : มีสายอากาศแบบ Whip
  - ระยะทำการในร่ม : สูงสุด 300 ฟุต หรือประมาณ 100 เมตร
  - ระยะทำการกลางแจ้ง (แบบ line-of-sight) : สูงสุดถึงประมาณ 1,500 เมตร
  - กำลังสูง : 60 mW (18 dBm)
  - ความไวในการรับสัญญาณ : -100 dBm (1% packet error rate)
  - การทำงานของขาพอร์ต : สามารถกำหนดผ่านทางซอฟต์แวร์ X-CTU เพื่อให้

ทำงานเป็น

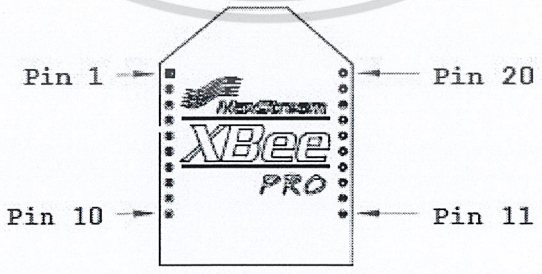
1) อินพุตอนาล็อกสำหรับวงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอล ความละเอียด 10 บิต

2) อินพุตเอาท์พุตดิจิตอล

- ขนาด : 0.96 x 1.297 นิ้ว หรือ 2.438 x 3.294 เซนติเมตร
- ไฟเลี้ยง : 2.8- 3.4 V.
- กระแสไฟฟ้า : เมื่อส่งข้อมูล 215 mA,รับข้อมูล 55 mA,น้อยกว่า 10 mA ในโหมดลดพลังงานที่ไฟเลี้ยง +3.3 V.
- อุณหภูมิใช้งาน : -40 องศาเซลเซียส ถึง 85 องศาเซลเซียส

- **คุณสมบัติด้านการสื่อสารข้อมูล**
  - สามารถทำงานเป็นอุปกรณ์มาสเตอร์และสเลฟได้
  - อัตราถ่ายทอดข้อมูลผ่านคลื่นวิทยุ : 250,000 บิตต่อวินาที
  - อัตราการถ่ายทอดข้อมูลอนุกรม (บอร์ดเรต) : 1,200 - 115,2000 บิตต่อวินาที
  - รูปแบบโครงสร้างข้อมูลที่รองรับ : จุดต่อจุด (Point to Point) , จุดต่อหลายจุด (Point to MultiPoint) และเข้ากันได้กับอุปกรณ์ตามมาตรฐาน IEEE 802.15.4
  - ทางเลือกแอดเดรส : PAN ID,ช่อง(channel) และแอดเดรส (Address) สำหรับแอดเดรสสามารถกำหนดรหัสแอดเดรสได้มากถึง 65,000 รหัส
  - เทคโนโลยีในการกระจายคลื่น : DSSS (Direct Sequence Spread Spectrum)
  - รองรับการทำงานทั้งแบบ API และ AT command สามารถกำหนดผ่านทางซอฟต์แวร์ X-CTU

2.2.2 การจัดขา



รูปที่ 2.4 การจัดขาของโมดูล XBee-PRO

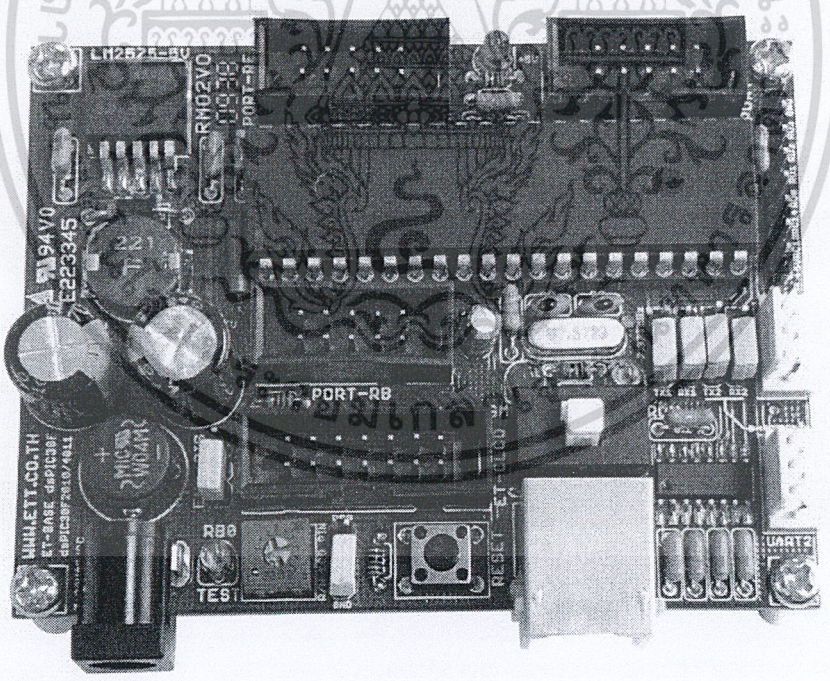
ตารางที่ 2.2 การจัดขาของโมดูล XBee-PRO และฟังก์ชันในการทำงาน

ขาที่	ชื่อขา	การทำงาน
1	Vcc	ขาต่อไฟเลี้ยง +3.3V
2	DOUT	ขาเอาท์พุตส่งข้อมูลอนุกรม
3	DIN	ขาอินพุตรับข้อมูลอนุกรม
4	DO8	ขาเอาท์พุตดิจิตอลช่อง8
5	RESET	ขารีเซตหลัก(แอกทีฟ “0”)
6	PWM0/RSSI	ขาเอาท์พุต PWM ช่อง0 และขาเอาท์พุตแสดงความแรงของการรับสัญญาณ
7	PWM1	ขาเอาท์พุต PWM ช่อง 1
8	NC	ไม่ใช้งาน
9	DTR/SLEEP_RQ/DI8	ขาอินพุตรับสัญญาณให้หยุดทำงานเข้าสู่โหมดSLEEP หรือ เป็นขาอินพุตดิจิตอลช่อง8
10	GND	ขาต่อกราวด์
11	AD4/DIO4	ขาอินพุตอนาล็อก 4 หรือ ขาอินพุต/เอาท์พุตดิจิตอล 4
12	CTS/DIO7	อินพุตรับสัญญาณแจ้งการส่งข้อมูลจากโฮสต์ (clear-2-send) ใช้ในการควบคุมจังหวะการรับส่งข้อมูลหรือเป็นขาอินพุต เอาท์พุตดิจิตอล 7
13	ON/SLEEP	ขาแสดงสถานการณ์ทำงาน “ 1 ” :อยู่ในโหมดทำงานปกติ “ 0 ” :อยู่ในโหมด SLEEP
14	VRE	ขาต่อแรงดันอ้างอิงสำหรับโมดูลแปลงสัญญารอนาล็อกเป็น ดิจิตอลภายใน XBee Series 2
15	ASSOCIATED/AD5/DIO5	ขาแสดงสถานะการเชื่อมต่อหรือขาอินพุตอนาล็อก 5 หรือขา อินพุตเอาท์พุตดิจิตอล 5
16	RTS/AD6/DIO6	ขาเอาท์พุตแจ้งความพร้อมในส่งข้อมูล (Ready-To-Send) ใช้ ควบคุมจังหวะการรับส่งข้อมูลหรือเป็นขาอินพุตอนาล็อก 6 หรือขาอินพุตเอาท์พุตดิจิตอล 6
17	AD3/DIO3	ขาอินพุตอนาล็อก 3 หรือขาอินพุตเอาท์พุตดิจิตอล 3
18	AD2/DIO2	ขาอินพุตอนาล็อก 2 หรือขาอินพุตเอาท์พุตดิจิตอล 2
19	AD1/DIO1	ขาอินพุตอนาล็อก 1 หรือขาอินพุตเอาท์พุตดิจิตอล 1
20	AD0/DIO0	ขาอินพุตอนาล็อก 0 หรือขาอินพุตเอาท์พุตดิจิตอล 0

2.3 ไมโครคอนโทรเลอร์ dsPIC30F4011

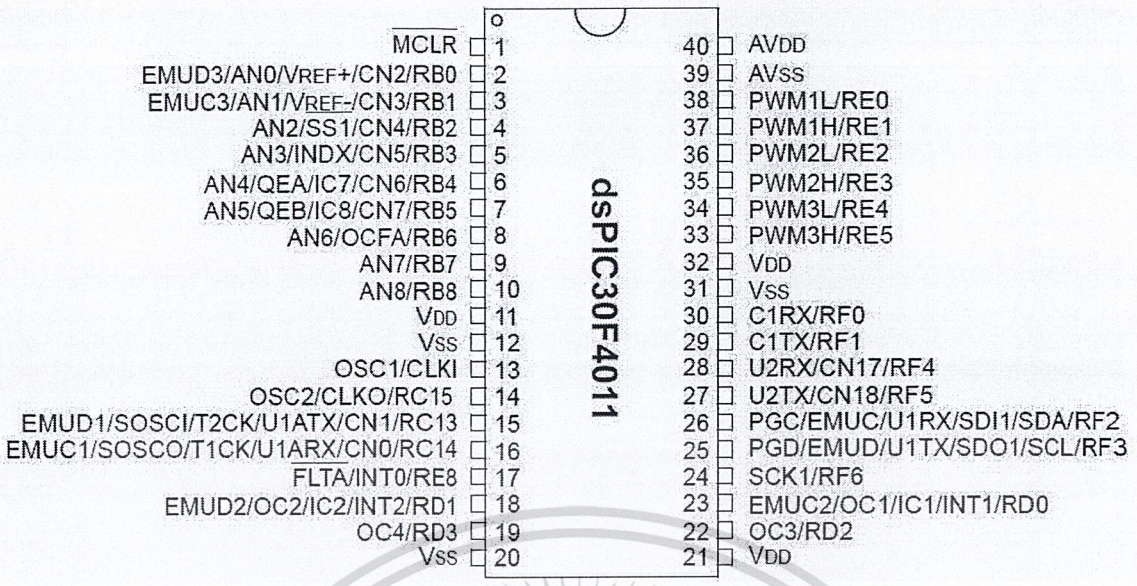
dsPIC30F4011 เป็นไมโครคอนโทรเลอร์ในตระกูล dsPIC30F ขนาด 40 Pin ของ Microchips เป็น MCU ประจำบอร์ด โดย dsPIC30F4011 เป็น MCU ซึ่งใช้การประมวลผลข้อมูลแบบ 16 บิต ซึ่งมีจุดเด่นในด้านของความสามารถในการประมวลผลข้อมูลสัญญาณแบบดิจิตอล เหมาะอย่างยิ่งสำหรับนำไปประยุกต์ใช้ในงานควบคุมต่างๆ โดยโครงสร้างภายในจะเป็นการผสมผสานระหว่างไมโครคอนโทรลเลอร์ (MCU) และวงจร DSP (Digital Signal Processing) รวมเข้าไว้ด้วยกัน หรืออาจเรียก MCU ตระกูล dsPIC30F ว่าเป็น DSC หรือ Digital Signal Controller ก็ได้ โดยโครงสร้างของบอร์ดไมโครคอนโทรเลอร์ dsPIC30F4011 ได้รับการออกแบบให้บอร์ดมีขนาดเล็ก เหมาะต่อการนำไปประยุกต์ใช้งานเป็นหลัก โดยภายในบอร์ดได้บรรจุเอาวงจรที่จำเป็นต่อการใช้งาน และสะดวกต่อการพัฒนาโปรแกรม มีความยืดหยุ่น สามารถปรับเปลี่ยนสัญญาณ I/O เพื่อนำไปประยุกต์ใช้งานในลักษณะต่างๆ ให้สอดคล้อง และเหมาะสมกับความต้องการใช้งานได้ในหลายๆ ลักษณะตามต้องการ

2.3.1 โครงสร้างบอร์ด dsPIC30F4011



รูปที่ 2.5 บอร์ดไมโครคอนโทรลเลอร์ dsPIC30f4011

- ขั้วต่อแหล่งจ่ายไฟเลี้ยงวงจรของบอร์ด ใช้กับแหล่งจ่ายไฟ 7-20VAC/DC
- IC Regulate แบบ Switching ขนาด 5V/1A
- LED TEST สำหรับทดสอบการทำงานของบอร์ด โดยควบคุมจาก RB0
- Jumper สำหรับ ตัด ต่อ สัญญาณ RB0 กับ LED TEST
- VR ปรับค่า สำหรับใช้ปรับความสว่างของหน้าจอแสดงผล LCD
- ขั้วต่อ 14PIN IDE สำหรับเชื่อมต่อกับ LCD แบบ Character
- Jumper สำหรับเลือกรูปแบบการควบคุมขา R/W ของ LCD
- สวิตช์ Reset สำหรับ Reset การทำงานของ MCU เมื่ออยู่ในโหมด Run
- ขั้วต่อ ICD2 สำหรับใช้เชื่อมต่อกับเครื่องโปรแกรมและดีบักตามมาตรฐาน ICD2
- สวิตช์ สำหรับเลือกโหมดการทำงานระหว่าง Run (RUN) และ Program (PGM)
- LED สีแดง แสดงสถานะ PGM เมื่อบอร์ดทำงานใน Program Mode
- LED สีเขียว แสดงสถานะ RUN เมื่อบอร์ดทำงานใน Run Mode
- ขั้วต่อ UART2 โดยเป็นสัญญาณแบบ RS232 โดยใช้ Pin ของ RF4 (RX2) และ RF5 (TX2) เป็นสัญญาณเชื่อมต่อ
- ขั้วต่อ UART1 โดยเป็นสัญญาณแบบ RS232 ซึ่งใช้ Pin ของ RC13(TX1),RC14(RX1) เป็นสัญญาณเชื่อมต่อ
- Jumper สำหรับเลือกการเชื่อมต่อสัญญาณ RC13,RC14,RF4,RF5 ว่าจะใช้ขาสัญญาณดังกล่าวทำหน้าที่เป็นขาสัญญาณรับส่งของ RS232 หรือ GPIO สำหรับใช้งานทั่วไป
- ขั้วต่อสัญญาณ RC13,RC14,RD0,RD1,RD2 และ RD3 สำหรับใช้งาน
- ขั้วต่อสัญญาณ Port-RF ซึ่งถ้าเป็น MCU รุ่น 40 Pin จะมี 7 บิต คือ RF [0..6]
- LED สำหรับแสดงสถานะ ของแหล่งจ่ายไฟ +5V ของบอร์ด
- ขั้วต่อสัญญาณ Port-RE ซึ่งจะมี 7 บิต คือ RE [0..6 และ 8]
- ขั้วต่อสัญญาณ Port-RB ซึ่งถ้าเป็น MCU 40 Pin จะมี 8 บิต คือ RB [0..7]
- MCU ประจำบอร์ด เป็น รุ่น 40 Pin ใช้เบอร์ dsPIC30F4011



รูปที่ 2.6 แสดงการจัดขาสัญญาณของ dsPIC30F4011

2.3.2 คุณสมบัติของบอร์ด

- เลือกใช้ MCU ตระกูล dsPIC30F4011 ของ Microchips เป็น MCU ประจำบอร์ด โดยคุณสมบัติเด่นๆของ MCU ได้แก่

- มีหน่วยความจำ Flash 48 KByte

- มีหน่วยความจำ RAM ขนาด 2 KByte

- มีหน่วยความจำ EEPROM ขนาด 1 KByte สำหรับเก็บข้อมูลใช้งาน

- มีพอร์ต I/O ขนาด 29 Bit

- มี 16 Bit Timer/Counter จำนวน 5 ชุด

- มี Input Capture จำนวน 4 ช่อง

- มี Output Compare จำนวน 4 ช่อง

- มี ADC 10 Bit/500 Ksps จำนวน 9 ช่อง

- มี PWM Motor Control จำนวน 6 ช่อง พร้อม Quadrature Encode Interface (QEI)

- มี UART จำนวน 2 ช่อง

- มี SPI จำนวน 1 ช่อง และ มี I2C จำนวน 1 ช่อง

- มีวงจร Watchdog, Power-ON Reset, PWM

- ใช้ Crystal ความถี่ 7.3728MHz สามารถใช้ PLL คูณความถี่เพื่อ Run ความถี่ 29.4912MHz ได้

- มีพอร์ตสื่อสารอนุกรม UART แบบ RS232 จำนวน 2 ช่อง พร้อม Jumper สำหรับเลือกใช้งาน UART หรือ GPIO ได้ตามต้องการ โดยใช้ขั้วต่อ UART แบบ CPA-4 Pin มาตรฐาน อีทีที

- มีขั้ว ICSP มาตรฐาน ICD2 แบบ RJ11 สำหรับใช้ร่วมกับชุดพัฒนาโปรแกรม และ Debugger ที่รองรับการทำงานตามมาตรฐาน ICD2 ของ Microchips เช่น ICD2 หรือ Pickit2 ได้

- มี Switch สำหรับสลับสัญญาณระหว่าง Program/Debug (PGM) และ ใช้งานปรกติ (RUN) พร้อม LED แสดงโหมดการทำงานของบอร์ด

- มีขั้วต่อสัญญาณ I/O แบบ Header ขนาด 2x5 จำนวน 3 ชุด และ Header 1x8 Pin อีก 1 ชุด

- Header 14Pin สำหรับ Character LCD พร้อม VR ปรับความสว่าง

- มี Switch Reset สำหรับสั่ง Reset การทำงานของ MCU ภายในบอร์ด

- มี LED สำหรับทดสอบการทำงาน โดยใช้ RB0 ในการควบคุม พร้อม Jumper ตัดต่อสัญญาณ

- Power AC/DC Input พร้อม Regulate แบบ Switching เบอร์ LM2575 ขนาด 5V/1A ลดปัญหาความร้อนจากวงจร Regulate และ LED แสดงสถานะแหล่งจ่าย Power

- ขนาด PCB Size เล็กเพียง 8 x 6 cm.

2.4 พอร์ตอนุกรม RS-232

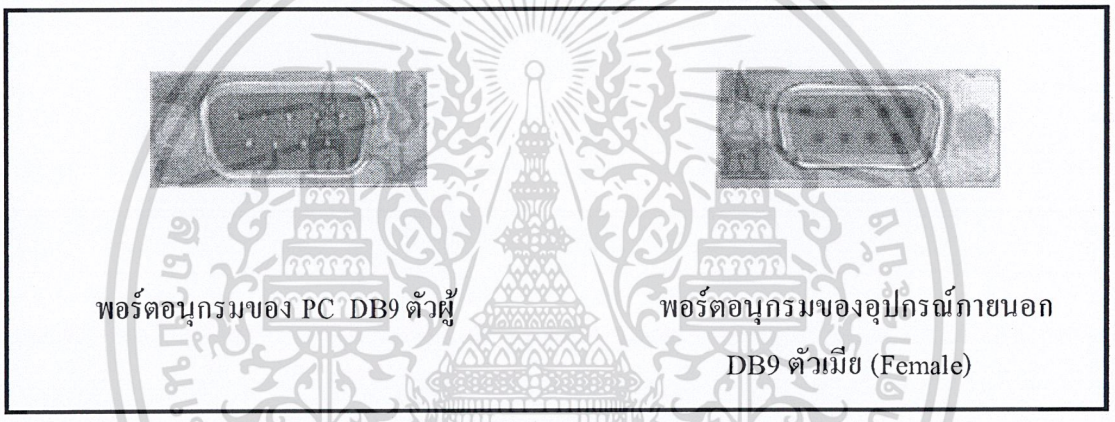
RS-232 ย่อมาจาก Recommended Standard-232 มาตรฐาน RS-232 เป็นมาตรฐานที่ได้รับการออกแบบมาเพื่อที่จะทำให้อุปกรณ์ต่อพ่วงจากผู้ผลิตต่างกันสามารถทำงานร่วมกันได้ มาตรฐานหลายชนิดได้รับการออกแบบขึ้นมา แต่มาตรฐานที่ได้รับความนิยมและใช้กันกว้างขวาง คือ มาตรฐาน RS-232 ซึ่งถูกประกาศใช้ในปี 1969 โดยสมาคมอุตสาหกรรมอิเล็กทรอนิกส์ ในยุคแรกการอินเตอร์เฟสแบบ RS-232 ถูกออกแบบสำหรับเชื่อมต่อเทอร์มินอลกับโมเด็ม เพื่อป้องกันไม่ให้เกิดการส่งข้อมูลบนสายเส้นเดียวกัน มาตรฐาน RS-232 ได้แบ่งอุปกรณ์ออกเป็น 2 ประเภท ซึ่งอุปกรณ์ทั้งสองประเภทนี้ก็คือ

1) DTE (Data Terminal Equipment) คือ อุปกรณ์สำหรับส่งข้อมูล เช่น คอมพิวเตอร์

2) DCE (Data Communication Equipment) คือ อุปกรณ์สำหรับทำการติดต่อ เช่น Modem ตามมาตรฐาน RS-232 อุปกรณ์ DTE ควรใช้หัวต่อตัวผู้ และอุปกรณ์ DCE ควรใช้หัวต่อตัวเมีย ซึ่งหัวต่อนิยมใช้กันอยู่จะเป็นชนิด D-Type ชนิด 9 ขาและ 25 ขา (บางครั้งเรียก DB-25 และ DB-9) ระดับแรงดันจะมีค่าระหว่าง –3V ถึง –15V สำหรับลอจิก High (Mask) และลอจิก Low จะมีระดับแรงดันระหว่าง +3V ถึง +15V (Space)

2.4.1 ลักษณะของคอนเน็กเตอร์แบบ D-Type

หัวต่อแบบ D-Type ที่ใช้ในการสื่อสารแบบอนุกรมของเครื่องคอมพิวเตอร์นั้น จะมีอยู่ 2 ลักษณะคือ แบบ 9 ขา และแบบ 25 ขา บางครั้งเรียกว่า DB9 และ DB25 ซึ่งหัวต่อทั้งสองชนิดจะมีลักษณะการทำงานของสัญญาณเหมือนกัน แต่การจัดเรียงไม่เหมือนกัน



รูปที่ 2.7 หัวต่อพอร์ตอนุกรมอาร์เอส-232

ตารางที่ 2.3 หน้าที่ของสัญญาณต่างๆ

Signal	Full Name	Originator	Function
TxD	Transmit Data	DTE	ส่งข้อมูลทีละบิตจาก DTE ไปยัง DCE
RxD	Receive Data	DCE	รับข้อมูลทีละบิตจาก DCE ไปยัง DTE
CTS	Clear to Send	DCE	ตรวจจับสัญญาณจาก DCE ว่าพร้อมจะรับข้อ-มูลจาก DTE
CD	Carrier Detect	DCE	เมื่อไรที่ตรวจสัญญาณเจอที่ปลายทางของสาย สัญญาณจะทำให้สายสัญญาณ Active
DSR	Data Set Ready	DCE	บอก DTE ว่า DCE พร้อมที่จะทำงานแล้ว
DTR	Data Terminal Ready	DTE	สัญญาณจาก DTE บอกให้ DCE เตรียมพร้อม
RTS	Request to Send	DTE	สัญญาณจาก DTE บอกให้ DCE เตรียมพร้อมที่จะรับข้อมูล
RI	Ring Indicator	DCE	ตรวจจับสัญญาณของสายโทรศัพท์

รูปที่ 2.8 DB9 ตัวผู้ เมื่อมองจากด้านหลัง

2.4.2 แสดงการจัดขา ของคอนเน็กเตอร์อนุกรมแบบ DB9 และหน้าที่การใช้งานต่างๆ

Pin	Description	Type
1	Data Carrier Detect (DCD)	Input
2	Received Data (RXD)	Input
3	Transmitted Data (TXD)	Output
4	Data Terminal Ready (DTR)	Output
5	Signal Ground (GND)	Input
6	Data Set Ready (DSR)	Input
7	Request To Send (RTS)	Output
8	Clear to Send (CTS)	Input
9	Ring Indicator (RI)	Input

รายละเอียดของสายสัญญาณต่างๆประกอบไปด้วย

- Transmit Data : TD ใช้สำหรับส่งข้อมูลอนุกรมออกจากคอมพิวเตอร์
- Receive Data : RD ใช้สำหรับรับข้อมูลอนุกรมเข้ามายังคอมพิวเตอร์
- Request To Send : RTS ใช้สำหรับส่งข้อมูลไปยังอุปกรณ์ปลายทาง เพื่อร้องขอให้อุปกรณ์ปลายทางส่งข้อมูลกลับมา
- Clear To Send : CTS ใช้สำหรับตรวจสอบว่าอุปกรณ์ที่เชื่อมต่อด้วย พร้อมที่จะรับข้อมูลหรือไม่โดยจะคอยรับสัญญาณ RTS เมื่อทุกอย่างพร้อมก็จะทำการส่งข้อมูลออกทางขา RT
- Data Set Ready : DSR ใช้สำหรับตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง จะใช้คู่กับขา DTR
- Signal Ground : SG เป็นกราวด์ของระบบ
- Carrier Detect : CD ขานี้จะ Active เมื่อมีการส่งสัญญาณ Carrier จากโมเด็ม
- Data Terminal Ready : DTR ใช้สำหรับบอกให้อุปกรณ์ปลายทางรับรู้ว่าต้องการติดต่อด้วยโดยขา DTR นี้ ต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทาง
- Ring Indicator : RI ขานี้จะ Active เมื่อโมเด็มได้รับสัญญาณเรียกเข้าจากสายโทรศัพท์

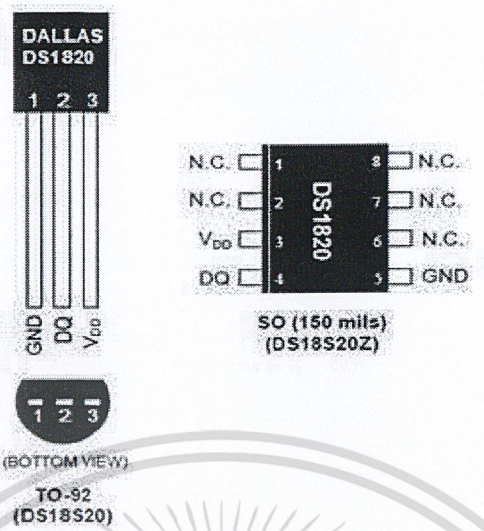
2.5 ไอซีวัดอุณหภูมิ DS1820

DS1820 เป็น IC วัดอุณหภูมิแบบดิจิตอล ของ Dallas Semiconductor สามารถวัดอุณหภูมิเป็นหน่วยองศา C ในช่วง -55C ถึง 125C ที่ความละเอียด 9-12 บิต และมีความแม่นยำอยู่ที่ 0.5C ในช่วง -10C ถึง 85C โดยจะให้สัญญาณเอาท์พุตออกมาเป็นแบบดิจิตอล และยังสามารถที่จะทำการโปรแกรมเข้าไปยังหน่วยความจำและควบคุมฟังก์ชันภายในไอซีได้ ซึ่งมีหน่วยความจำรอมภายในขนาด 64 บิต แบบเลเซอร์รอม จึงสามารถที่จะทำการอ่านและเขียนข้อมูลเกี่ยวกับหน้าที่การทำงาน ในการตรวจวัดอุณหภูมิได้อย่างมากตามการประมวลผลของไมโครโปรเซสเซอร์ นอกจากนั้นแล้วยังสามารถติดตั้ง DS1820 เพื่อการตรวจวัดอุณหภูมิได้ในหลายลักษณะและหลายสถานที่ ตำแหน่งการติดตั้งที่มีความแตกต่างอย่างมากกับอุปกรณ์ทั่วๆ ไปไม่ว่าจะเป็นการติดตั้งภายในอาคาร อุปกรณ์เครื่องใช้ต่างๆ หรือภายในเครื่องจักรและเอาท์พุตที่เป็นอนุกรมตัวเลขของ DS1820 สามารถต่อเอาท์พุตบนสายสัญญาณเพียงเส้นเดียวได้หลายๆ ชุดโดยไม่สับสนข้อมูลซึ่งกันและกัน

2.5.1 คุณสมบัติเด่นของ DS1820

- อินเตอร์เฟสสัญญาณผ่านขาเอาท์พุตเพียงพอร์ตเดียวแบบ 1 สายข้อมูล (1-Wire)
- ขยายจุดตรวจจับอุณหภูมิได้หลายๆ จุดบนสายข้อมูลเพียง 1 สายข้อมูล
- ไม่ต้องใช้อุปกรณ์ภายนอกมาร่วม
- สามารถควบคุมการทำงาน POWER ON ได้ผ่านทางสายข้อมูล
- POWER ON ขณะสแตนด์บายเป็นศูนย์
- ย่านการวัดอุณหภูมิตั้งแต่ -55 องศาเซลเซียสถึง +125 องศาเซลเซียสที่ 0.5 องศาเซลเซียสต่อสเต็ป หรือตั้งแต่ย่าน -67 องศาฟาเรนไฮน์ถึง +257 องศาฟาเรนไฮน์ที่ 0.9 องศาฟาเรนไฮน์ต่อสเต็ป
- อัตราความเร็วในการแปลงจากอุณหภูมิมาเป็นค่าตัวเลขดิจิตอลเท่ากับ 200 µsec
- ผู้ใช้งานสามารถกำหนดการเซตค่าเตือนย่านอุณหภูมิได้ในแบบ Non-Volatile
- การเตือนย่านอุณหภูมินั้นสามารถกำหนดรหัสผ่านการสั่งการและแอดเดรสของอุปกรณ์ได้จากภายนอกพื้นที่ตรวจวัดอุณหภูมิผ่านทางโปรแกรมภายนอก
- เหมาะกับการประยุกต์ใช้งานตรวจวัดอุณหภูมิและติดตั้งไว้ในอุปกรณ์ควบคุมเทอร์โมสแตติก ระบบโรงงานอุตสาหกรรม ผลิตภัณฑ์ เทอร์โมมิเตอร์ หรือระบบอื่นๆ ที่มีส่วนตรวจจับอุณหภูมิทำงานร่วมอยู่

2.5.2 การจัดการขา



รูปที่ 2.9 โครงสร้าง และขาของ DS1820 ตัวถังแบบ TO-92

ตางรางที่ 2.4 การจัดขาของ DS1820

ขาที่	ชื่อขา	รายละเอียด
1	GND	ขากราวด์
2	DQ	ขา Data / Input / Output
3	VDD	ขาไฟเลี้ยง +3V - +5V

การสื่อสารและควบคุม DS18B20 นั้นสามารถทำได้โดยใช้บัสข้อมูลแบบ 1-wire ของ Dallas Semiconductor ซึ่งใช้สายสัญญาณเพียงแค่เส้นเดียวเท่านั้น ภายใน DS18B20 แต่ละตัวมีโค๊ดประจำตัวขนาด 64 บิต ทำให้สามารถใช้งาน DS18B20 หลายตัวทำงานบนบัสแบบ 1 wire พร้อมกันได้ นอกจากนี้ DS18B20 ยังสามารถทำงานในโหมดพาราสิต (Parasite Power Mode) ซึ่งเป็นการทำงานโดยไม่ใช้ไฟเลี้ยง แต่ใช้พลังงานจากสายสัญญาณ 1-wire ซึ่งมีประโยชน์มากสำหรับการวัดอุณหภูมิระยะไกล หรือในการใช้งานในที่ ๆ มีเนื้อที่จำกัด

ข้อมูลอุณหภูมิที่วัดได้จะถูกเก็บอยู่ในรีจิสเตอร์ Temperature ซึ่งมีขนาด 16 บิตดังแสดงในรูป ถ้าข้อมูลอุณหภูมิเป็นบวก S จะเป็น “1” แต่ถ้าข้อมูลอุณหภูมิเป็นลบ S จะเป็น “0” ในกรณีที่ DS18B20 ทำงานในโหมดความละเอียด 12 บิต บิตทุกบิตในรีจิสเตอร์ Temperature จะถูกใช้หมด แต่ในกรณีที่ทำงานในโหมด 9-11 บิต บิตล่าง (บิต 0 – บิต 2) จะไม่ถูกใช้งาน ซึ่งในการกำหนดโหมดความละเอียดการทำงานของ DS18B20 นั้นสามารถกำหนดได้ที่รีจิสเตอร์ Configuration ซึ่งโดยปกติเริ่มต้น DS18B20 จะทำงานในโหมด 12 บิต

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
LS Byte	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	bit15	**bit14**	**bit13**	**bit12**	**bit11**	**bit10**	**bit9**	**bit8**
MS Byte	S	S	S	S	S	2^6	2^5	2^4

รูปที่ 2.10 โครงสร้างภายในรีจิสเตอร์ Temperature LSB และ MSB

ตารางที่ 2.5 ความสัมพันธ์ระหว่างอุณหภูมิกับค่าที่อ่านได้

Temperature	Digital Output (Binary)	Digital Output (HEX)
+85°C	0000 0101 0101 0000	0550h
+125°C	0000 0000 1111 1010	00FAh
+25.0°C	0000 0000 0011 0010	0032h
+0.5°C	0000 0000 0000 0001	0001h
+0.0°C	0000 0000 0000 0000	0000h
-0.5°C	1111 1111 1111 1111	FFFFh
-25.0°C	1111 1111 1100 1110	FFCEh
-55°C	1111 1111 1001 0010	FF92h

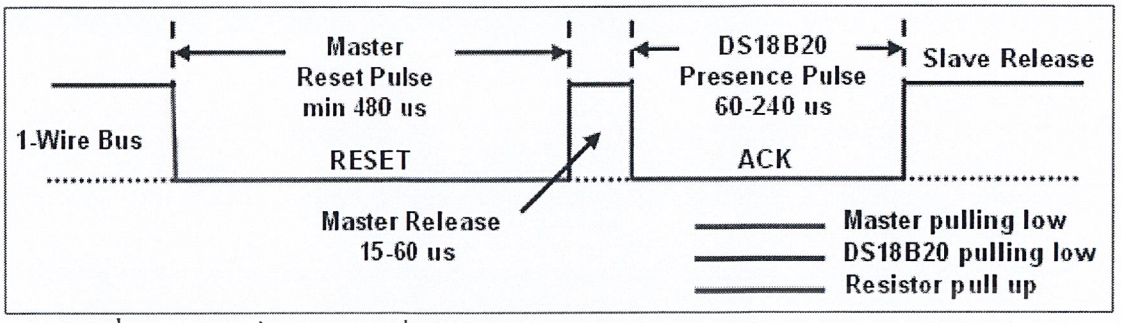
การสื่อสารแบบ 1-wire เป็นระบบบัสข้อมูลแบบ Half-duplex นั้นคือสามารถสื่อสารได้ 2 ทิศทาง แต่ไม่สามารถรับและส่งข้อมูลพร้อมกันในช่วงเวลาเดียวกันได้ ระบบบัสมีการทำงานเป็นแบบ Master/Slave โดยอุปกรณ์ Master จะเป็นตัวควบคุมสถานะ และจังหวะการรับส่งของบัสข้อมูล ในขณะที่อุปกรณ์ Slave จะทำงานตามการควบคุมของอุปกรณ์ Master เท่านั้น

ในการใช้งานบัสแบบ 1 wire นี้ สายสัญญาณข้อมูล DQ จะต้องมีสภาวะปกติที่ลอจิกสูง สามารถทำได้โดยการต่อตัวต้านทานประมาณ 5 กิโลโอห์มพูลอัปไว้กับไฟเลี้ยง

รูปแบบของสัญญาณบนบัส 1-wire สามารถแบ่งออกได้เป็น 6 รูปแบบ คือ Reset Pulse, Presence Pulse, write 0, write 1, read 0, read 1

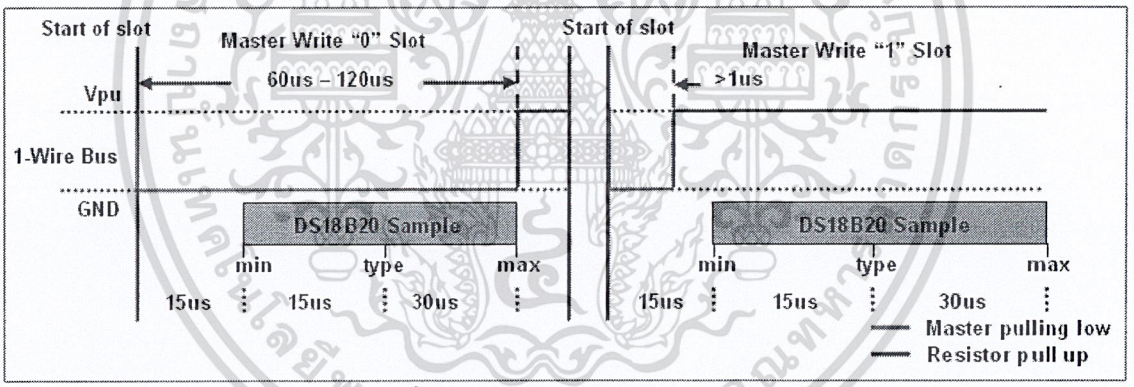
ในกระบวนการเริ่มต้นการสื่อสารแบบ 1-wire ทั้งหมดนั้น อุปกรณ์ Master ต้องขอเริ่มการสื่อสารด้วยการสร้าง Reset Pulse ก่อน เมื่ออุปกรณ์ Slave ได้รับ Reset Pulse ก็จะสร้าง

Presence Pulse เพื่อตอบรับการขอเริ่มการสื่อสารนั้น ซึ่งมีรายละเอียดของช่วงเวลาต่าง ๆ ดังแสดงในรูปที่ 2.11



รูปที่ 2.11 การเริ่มการติดต่อสื่อสารแบบ 1-wire ด้วย Reset Pulse และ Presence Pulse

ในการเขียนข้อมูลแบ่งออกเป็น 2 ชนิดคือการเขียนข้อมูล "1" และการเขียนข้อมูล "0" ดังแสดงในรูปที่ 2.12 การเขียนข้อมูลลง DS18B20 ต้องใช้ช่วงเวลาของไทม์สล๊อตอย่างต่ำ 60 µsec และต้องมีช่วงเวลาระหว่างไทม์สล๊อตอย่างต่ำ 1 µsec

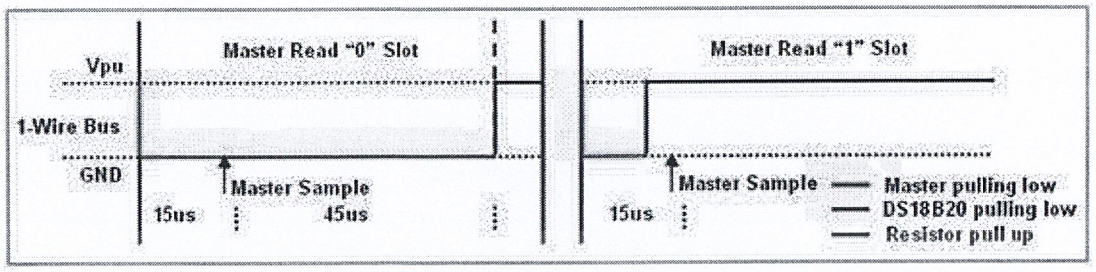


รูปที่ 2.12 การเขียนข้อมูลลง DS18B20

การเขียนข้อมูลทั้ง 2 ชนิด เริ่มแรกอุปกรณ์ Master ต้องดึงสัญญาณบนบัส 1-wire ลงมาให้อยู่ในสถานะลอจิกต่ำก่อน ในกรณีที่ต้องการเขียนข้อมูล "0" ลงใน DS18B20 อุปกรณ์ Master ต้องดึงสัญญาณบนบัสให้เป็นลอจิกต่ำต่อ จนกว่าจะครบช่วงเวลาไทม์สล๊อต (อย่างต่ำ 60 µsec) ส่วนในกรณีที่ต้องการเขียนข้อมูล "1" ลง DS18B20 อุปกรณ์ Master ต้องปล่อยบัส เพื่อให้บัสกลับไปอยู่ในสถานะลอจิกสูงก่อนการ Sampling ของ DS18B20 ซึ่งจะอยู่ในช่วง 15 µsec-60 µsec หลังจากที่อุปกรณ์ Master ดึงสัญญาณบัส 1-wire ลงมา

ในการอ่านค่าภายใน SRAM ของ DS18B20 สามารถทำได้ก็ต่อเมื่ออุปกรณ์ Master ได้เขียนข้อมูลเพื่อขอทำการอ่านค่าใน SRAM (Read Scratchpad) ซึ่งมีค่าเป็น 0xBE ลงไปที่

DS18B20 เสียก่อน จากนั้นจึงเริ่มอ่านข้อมูลจากบัส 1-wire โดยไทม์สล๊อตของการอ่านต้องมีช่วงเวลาอย่างต่ำ 60 μsec และต้องมีช่วงเวลาระหว่างไทม์สล๊อตอย่างต่ำ 1 μsec ดังแสดงในรูปที่ 2.13



รูปที่ 2.13 การอ่านข้อมูลจาก DS18B20

การอ่านข้อมูลจากบัส 1-wire เริ่มแรกอุปกรณ์ Master จะต้องดึงบัส 1-wire ลงให้อยู่ในสถานะลอจิกต่ำเป็นช่วงเวลาอย่างน้อย 1 μsec จากนั้นจึงค่อยปล่อยบัส ในกรณีที่ DS18B20 ส่งข้อมูล "0" DS18B20 จะดึงบัสให้เป็นลอจิกต่ำจนจนกว่าจะสิ้นสุดไทม์สล๊อตถึงจึงจะปล่อยบัสให้กลับไปอยู่ในสถานะลอจิกสูง ส่วนในกรณีที่ DS18B20 ส่งข้อมูล "1" DS18B20 จะปล่อยบัสให้อยู่ในสภานะลอจิกสูงตลอด ในการ Sample เพื่อรับข้อมูลจาก DS18B20 ควรทำภายใน 15 μsec หลังจากจุดเริ่มของไทม์สล๊อตดังแสดงในรูปที่ 2.13

ขั้นตอนการเข้าใช้งาน DS18B20 มี 3 ขั้นตอนด้วยกันคือ

1. Initialization
2. ROMCommand
3. DS18B20 Function Command

การ Initialization ประกอบไปด้วยการส่ง Reset Pulse จากอุปกรณ์ Master ตามด้วย Presence Pulse ซึ่งตอบรับโดย DS18B20 เพื่อบ่งบอกว่าอุปกรณ์พร้อมทำงาน

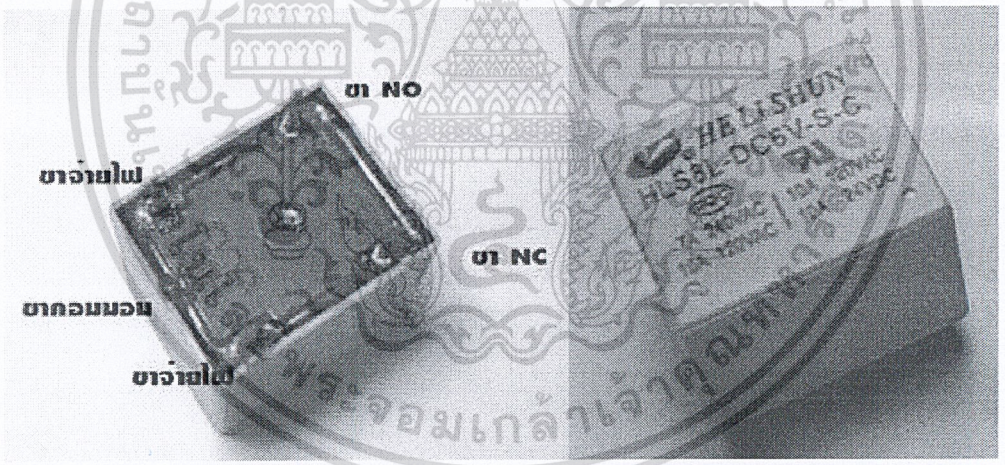
ตารางที่ 2.6 Rom Function Commands

Name	Command	Description
Read Rom	33h	ใช้อ่านค่า 64 บิต (Lasers ROM) แต่ command นี้จะใช้ได้ก็ต่อเมื่อต่อ DS1820 ไว้ในตัวเดียวเท่านั้น ถ้าต่อหลายๆ ตัวแล้วใช้command นี้ DS1820 แต่ละตัวก็จะส่งข้อมูลออกมาชนกัน
Match Rom	55h	Command นี้แล้วตามด้วย 64 บิต (Lasers ROM) เพื่อให้มาสเตอร์ระบุว่าจะให้ DS1820 ตัวไหนทำงาน จะใช้ก็ต่อเมื่อต่อ DS1820 หลายตัวในสายเส้นเดียวกัน
Skip Rom	CCh	ถ้าเราต่อ DS1820 เพียงตัวเดียวใน BUS เราก็ไม่ต้องระบุ 64 บิต (Lasers ROM) เราสามารถใช้ command นี้เพื่อลดขั้นตอนในการสื่อสาร
Search Rom	F0h	เมื่อเริ่มทำการ Initial ไปแล้วตัวมาสเตอร์อาจจะยังไม่รู้ว่ามี DS1820 อยู่กี่ตัว command Search ROM ทำให้มาสเตอร์สามารถแยกแยะ DS1820 แต่ละตัวได้
Write Scratchpad	4Eh	ใช้ในการเขียนข้อมูลลง DS1820 เริ่มต้นที่ตำแหน่งของ TH register การเขียนจะสิ้นสุดลงเมื่อไหร่ก็ได้ ในการส่งสัญญาณรีเซต
Read Scratchpad	48h	ใช้ในการ copy ข้อมูล scratch pad ไปยัง EE Memory
Convent T	44h	ใช้ในการเริ่มต้นแปลงค่าอุณหภูมิ
Recall E2	B8h	Command นี้จะอ่านค่า Temperature Trigger ที่เก็บไว้ใน EE Memory ขึ้นมา
Read Power Supply	B4h	หลังจากส่ง command นี้ไปแล้ว DS1820 จะส่งสัญญาณกลับมา ส่งศูนย์เท่ากับ Parasite Power, ส่งหนึ่งเท่ากับ External power supply

หลังจากการทำ Initialization เสร็จเรียบร้อยแล้ว อุปกรณ์ Master ต้องส่ง ROM Command ไปยัง DS1820 ROM Command นั้นแบ่งออกได้เป็น 5 คำสั่งด้วยกันคือ SEARCH ROM [F0h], READ ROM [33h], MATCH ROM [55h], SKIP ROM [CCh], ALARM SEARCH [ECh] ซึ่งในกรณีที่ต่อใช้งาน DS1820 เพียงตัวเดียวนั้นจะใช้ ROM Command ได้แค่ 2 คำสั่ง นั้นคือ READ ROM ซึ่งเป็นการอ่านค่า ROM Code ขนาด 64 บิต บนตัว DS1820 อีกคำสั่งคือ SKIP ROM ซึ่งเป็นคำสั่งที่ใช้ในกรณีที่อุปกรณ์ Master ต้องการส่งคำสั่งควบคุม DS1820 ทุกตัว ซึ่งไม่จำเป็นต้องระบุ ROM Code

หลังจากที่อุปกรณ์ Master ส่ง Rom Command ไปยัง DS18B20 แล้ว อุปกรณ์ Master จะสามารถใช้ Function Command เพื่อเข้าไปควบคุมการทำงานของ DS1820 ได้ Function Command ประกอบไปด้วย CONVERT T [44h], WRITE SCRATCHPAD[4Eh], READ SCRATCHPAD[BEh], COPY SCRATCHPAD [48h], RECALL E2[B8h], READ POWER SUPPLY [B4h]

2.6 รีเลย์



รูปที่ 2.14 รีเลย์ควบคุม

รีเลย์ (Relay) เป็นอุปกรณ์แม่เหล็ก Magnatics device ที่ใช้กันมานาน ปัจจุบันก็ยังคงนิยมใช้กันอยู่ แต่ในปัจจุบันนี้ รีเลย์ ถูกพัฒนาให้มีคุณภาพดีกว่าสมัยก่อนมาก แต่ยังคงหลักการ และโครงสร้างเดิม เอาไว้

2.6.1 ชนิดของวงจรรีเลย์

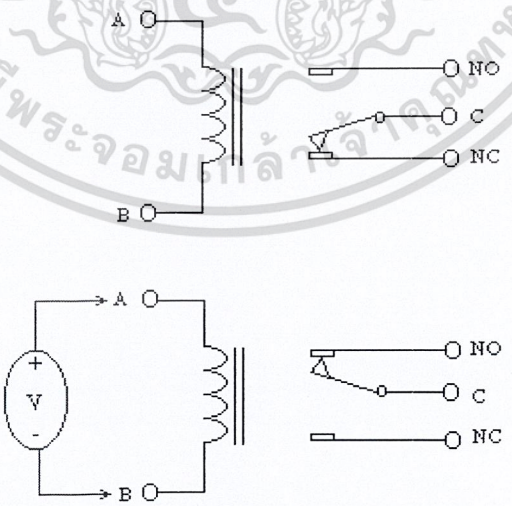
รีเลย์ (Relay) เป็นอุปกรณ์ทำหน้าที่เป็นสวิตช์มีหลักการทำงานคล้ายกับ ขดลวดแม่เหล็กไฟฟ้าหรือโซลินอยด์ (solenoid) รีเลย์ใช้ในการควบคุมวงจร ไฟฟ้าได้อย่างหลากหลาย รีเลย์ เป็นสวิตช์ควบคุมที่ทำงานด้วยไฟฟ้า แบ่งออกตามลักษณะการใช้งานได้เป็น 2 ประเภทคือ

1. รีเลย์กำลัง (Power relay) หรือมักเรียกกันว่า คอนแทกเตอร์ (Contactor or Magnetic contactor) ใช้ในการควบคุมไฟฟ้ากำลัง มีขนาดใหญ่กว่ารีเลย์ธรรมดา

2. รีเลย์ควบคุม (Control Relay) มีขนาดเล็กกำลังไฟฟ้า ต่ำ ใช้ในวงจรควบคุมทั่วไปที่มีกำลังไฟฟ้าไม่มากนัก หรือเพื่อ การควบคุมรีเลย์หรือคอนแทกเตอร์ขนาดใหญ่ รีเลย์ควบคุม บางทีเรียกกันง่าย ๆ ว่า "รีเลย์"

หน้าที่ของคอนแทกเตอร์ คือ การใช้กำลังไฟฟ้าจำนวนน้อยเพื่อไปควบคุมการตัดต่อกำลังไฟฟ้าจำนวนมาก คอนแทกเตอร์ทำให้เราสามารถควบคุมกำลังไฟฟ้าในตำแหน่งอื่นๆ ของระบบไฟฟ้าได้ สายไฟควบคุมให้รีเลย์กำลังหรือคอนแทกเตอร์ทำงานเป็นสายไฟฟ้าขนาดเล็กต่อเข้ากับสวิตช์ควบคุมและคอล์ยของคอนแทกเตอร์ กำลังไฟฟ้าที่ป้อนเข้าคอล์ยอาจจะเป็นไฟฟ้ากระแสตรง หรือไฟฟ้ากระแสสลับก็ได้ขึ้นอยู่กับการออกแบบ การใช้คอนแทกเตอร์ทำให้สามารถควบคุมวงจรจากระยะไกล (Remote) ได้ ซึ่งทำให้เกิดความปลอดภัยกับผู้ปฏิบัติงานในการควบคุมกำลังไฟฟ้า

2.6.2 โครงสร้างภายในของรีเลย์



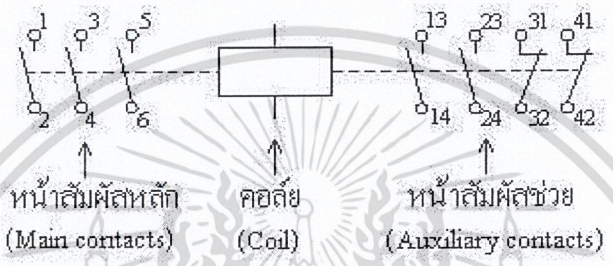
รูปที่ 2.15 โครงสร้างภายในของรีเลย์

ภายในโครงสร้างของ รีเลย์ จะประกอบไปด้วยขดลวด (Coil) 1 ชุด และ หน้าสัมผัส (Contactor) ซึ่งในหน้าสัมผัส 1 ชุด จะประกอบไปด้วย

1. หน้าสัมผัสแบบปกติปิด (Normally Close หรือ NC.) ซึ่งในสภาวะปกติ ขานี้จะต่ออยู่กับขาร่วม (Common)

2. หน้าสัมผัสแบบปกติเปิด (Normally Open หรือ NO.) ขานี้จะต่อเข้ากับขาร่วม (Common) เมื่อขดลวดมีแรงดันตกคร่อม หรือกระแสไหลผ่าน ในปริมาณที่เพียงพอ

ใน รีเลย์ 1 ตัว อาจมีหน้าสัมผัสมากกว่า 1 ชุด เช่น 2 ชุด, 4 ชุด แล้วแต่ผู้ผลิต



รูปที่ 2.16 โครงสร้างหน้าสัมผัสภายในของรีเลย์

คอนแทกเตอร์ (Contactors) นอกจากจะมีหน้าสัมผัสทั้งส่วนเคลื่อนที่ และหน้าสัมผัสส่วนที่อยู่กับที่แล้วหน้าสัมผัสภายในของคอนแทกเตอร์ยังแบ่งออกเป็น 2 ส่วนตามลักษณะของการทำงาน ซึ่งแบ่งออกเป็น 2 ส่วนดังนี้ คือ

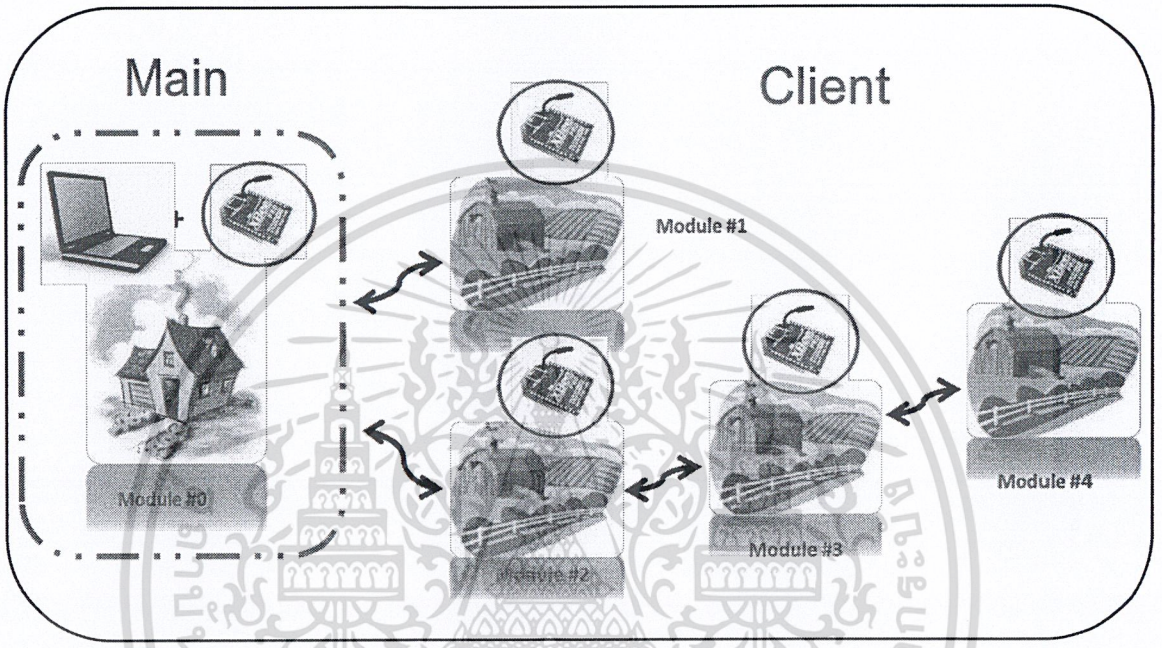
1. หน้าสัมผัสหลัก (Main Contacts) โดยปกติแล้วหน้าสัมผัสหลักมี 3 อัน สำหรับส่งผ่านกำลังไฟฟ้า 3 เฟสเข้าไปสู่มอเตอร์ หรือโหลดที่ใช้แรงดันไฟฟ้า 3 เฟส หน้าสัมผัสหลักของคอนแทกเตอร์มีขนาดใหญ่ทนแรงดันและกระแสได้สูง หน้าสัมผัสหลักเป็นชนิดปกติเปิด (Normally open;N.O. contact) อักษรกำกับ หน้าสัมผัสด้านแหล่งจ่ายคือ 1, 3, 5 หรือ L1, L2, L3 และด้านโหลดคือ 2, 4, 6 หรือ T1, T2, T3 ดังรูป

2. หน้าสัมผัสช่วย (Auxiliary Contacts) หน้าสัมผัสชนิดนี้ติดตั้งอยู่ด้านข้างทั้งสองด้านของตัวคอนแทกเตอร์ มีขนาดเล็กทนกระแสได้ต่ำทำหน้าที่ช่วยการทำงานของวงจร เช่น เป็นหน้าสัมผัสที่ทำให้คอนแทกเตอร์ทำงานได้ตลอดเวลา หรือเรียกว่า "holding" หรือ "maintaining contact" หน้าสัมผัสช่วยนี้จะเป็นหน้าสัมผัสแบบโยกได้สองทาง โดยจะถูกดึงขึ้น-ลงไปตามจังหวะการดูด-ปล่อยของคอนแทกเตอร์ อักษรกำกับหน้าสัมผัสช่วย จะเป็น 13, 14 สำหรับ

คอนแทกเตอร์ที่มีหน้าสัมผัสช่วยแบบปกติเปิด 1 ชุด ถ้ามี N.O. ชุดที่ 2จะเป็น 23, 24 และหน้าสัมผัสช่วยแบบปกติปิดจะมีอักษรกำกับเป็น 31, 32 และ 41, 42

บทที่ 3
การออกแบบระบบการทำงาน

3.1 การทำงานของระบบ

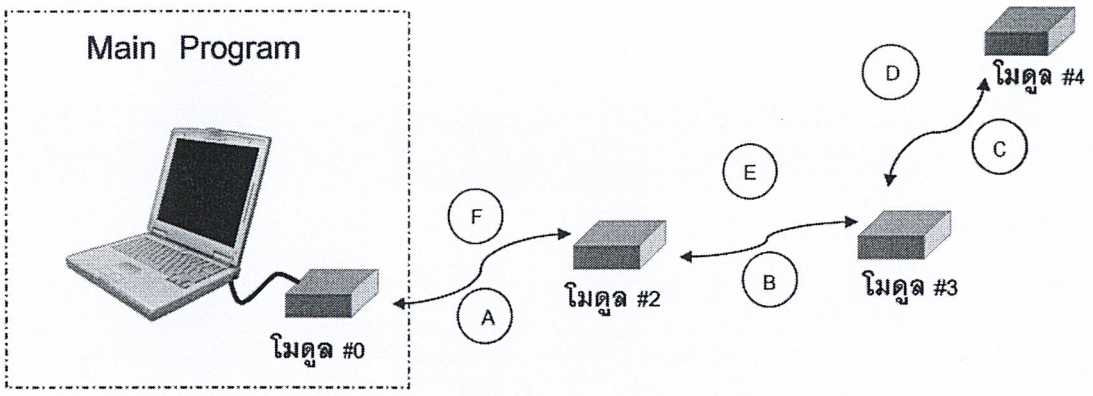


รูปที่ 3.1 แสดงรูปแบบสัญญาณการติดต่อระหว่าง PC กับโมดูล

การออกแบบจะแบ่งออกเป็นเครื่อง Main และเครื่อง Client ในด้านของ Client จะใช้เซนเซอร์ตัววัดอุณหภูมิ DS1820 เมื่อวัดค่าได้แล้วจะส่งค่านั้นมาที่บอร์ดภาครับ ซึ่งจะแสดงผลผ่านทางหน้าจอคอมพิวเตอร์โดยผ่านพอร์ต RS-232

3.2 หลักการทำงานของเครื่อง Main

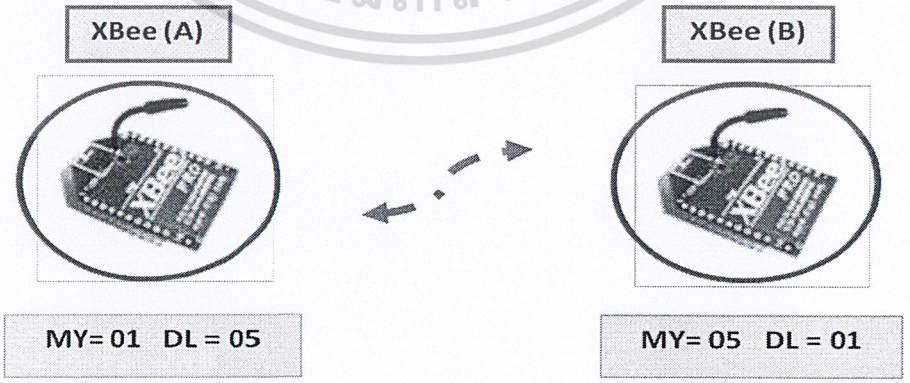
ส่วนนี้จะเป็นฮาร์ดแวร์ที่ใช้ในการติดต่อรับ-ส่งข้อมูล หรือคำสั่งจากตัวเครื่องคอมพิวเตอร์ (Main) กับตัวเครื่องที่บรรจุเซนเซอร์วัดอุณหภูมิไว้ หรือที่เรียกว่า Client เมื่อผู้ใช้ (User) ทำการเลือกระบบวัดค่าว่าเป็นแบบ Auto/Manual ผ่านทางหน้าจอคอมพิวเตอร์ หลังจากนั้นเครื่อง Main จะทำการส่งคำสั่งไปยังอุปกรณ์ XBee-PRO โดยผ่าน RS-232



รูปที่ 3.2 แสดงผลการเดินทางของคำสั่งในแต่ละช่วง

โปรแกรมหลัก (Main) (C# บน PC) โปรแกรมส่วนนี้ทำเป็น GUI เพื่อทำหน้าที่ติดต่อกับผู้ใช้ในการสั่งงานควบคุมต่างๆ ต่อโมดูลตัวลูก เช่น การอ่านผลอุณหภูมิ สั่งรีเลย์ทำงาน และในการทำงานโหมดต่างๆ (Auto/Manual) โดยมีการจัดรูปแบบคำสั่งเพื่อติดต่อกับตัววัดที่ต้องการติดต่อด้วยอย่างอัตโนมัติ

อนึ่ง ในส่วนของ XBee-PRO ที่ได้จัดหามานั้นเป็นแบบ XBee-PRO Series 1 ซึ่งไม่สามารถทำงานในโหมด Mesh หรือเน็ตเวิร์คระดับสูงได้ จึงได้เลือกให้ทำงานแบบ Peer to Peer คือติดต่อแบบตัวต่อตัวเลย โดยกำหนดค่าการติดต่อแต่ละครั้งเองโดยไมโครคอนโทรลเลอร์ รูปแบบการติดต่อกันแบบตัวต่อตัวคือ หนึ่งตัวของ XBee-PRO จะต้องกำหนดไว้ก่อนว่าตัวเองเบอร์อะไรและตัว XBee ที่จะติดต่อด้วยเบอร์อะไร ทั้งสองตัวต้องกำหนดค่าไว้สลับกันดังนี้ เช่น ตัว A มีการกำหนดเบอร์ตัวเองคือ MY = 01 และตัวที่จะติดต่อด้วยคือ DL = 05 ฉะนั้นตัว B ก็ต้องกำหนดเป็น MY = 05 และ DL = 01 นั่นเอง การทำเช่นนี้หากมีตัวอื่นอยู่ในรัศมีการส่งก็จะไม่กระทบกระเทือนตัวอื่นเลย จะมีการรับส่งได้เฉพาะคู่ A-B นี้เท่านั้น



รูปที่ 3.3 แสดงการกำหนดค่าของเบอร์ของ XBee-PRO

ส่วนการกำหนดค่าของเบอร์คือ MY, DL นั้นสามารถทำได้โดยการส่งคำสั่งผ่าน RS-232 จาก PC หรือ Microcontroller เป็นคำสั่งแบบ AT-COMMAND ได้ ดังนั้นในส่วนของโปรแกรมทุกตัวก็จะมีการสร้างฟังก์ชั่นการกำหนด MY, DL นี้ไว้ใช้งานเองด้วย เพราะเส้นทางการส่งข้อมูลระหว่าง PC ถึง ตัวโมดูลที่ต้องการติดต่อด้วยนั้น จะต้องสามารถกำหนดได้อย่างอิสระจากทางด้าน PC

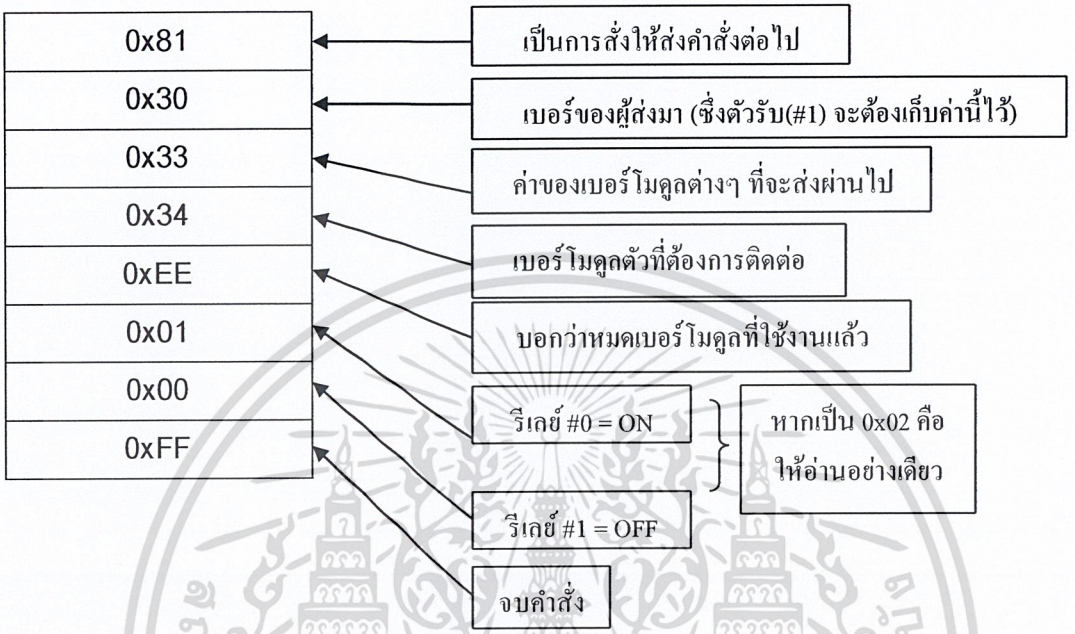
3.2.1 รูปแบบของคำสั่งในแต่ละไบท์ที่ใช้งาน

การส่งคำสั่งจาก PC เพื่ออ่านอุณหภูมิและสั่งงานรีเลย์ จะกำหนดให้มีการอ่านค่าของอุณหภูมิจากตัว โมดูลเบอร์ 4 (#4) และมีการสั่งให้ Relay0, Relay1 ให้ ON และ OFF ตามลำดับ

ตารางที่ 3.1 รูปแบบของคำสั่งในแต่ละไบท์ที่ใช้งาน

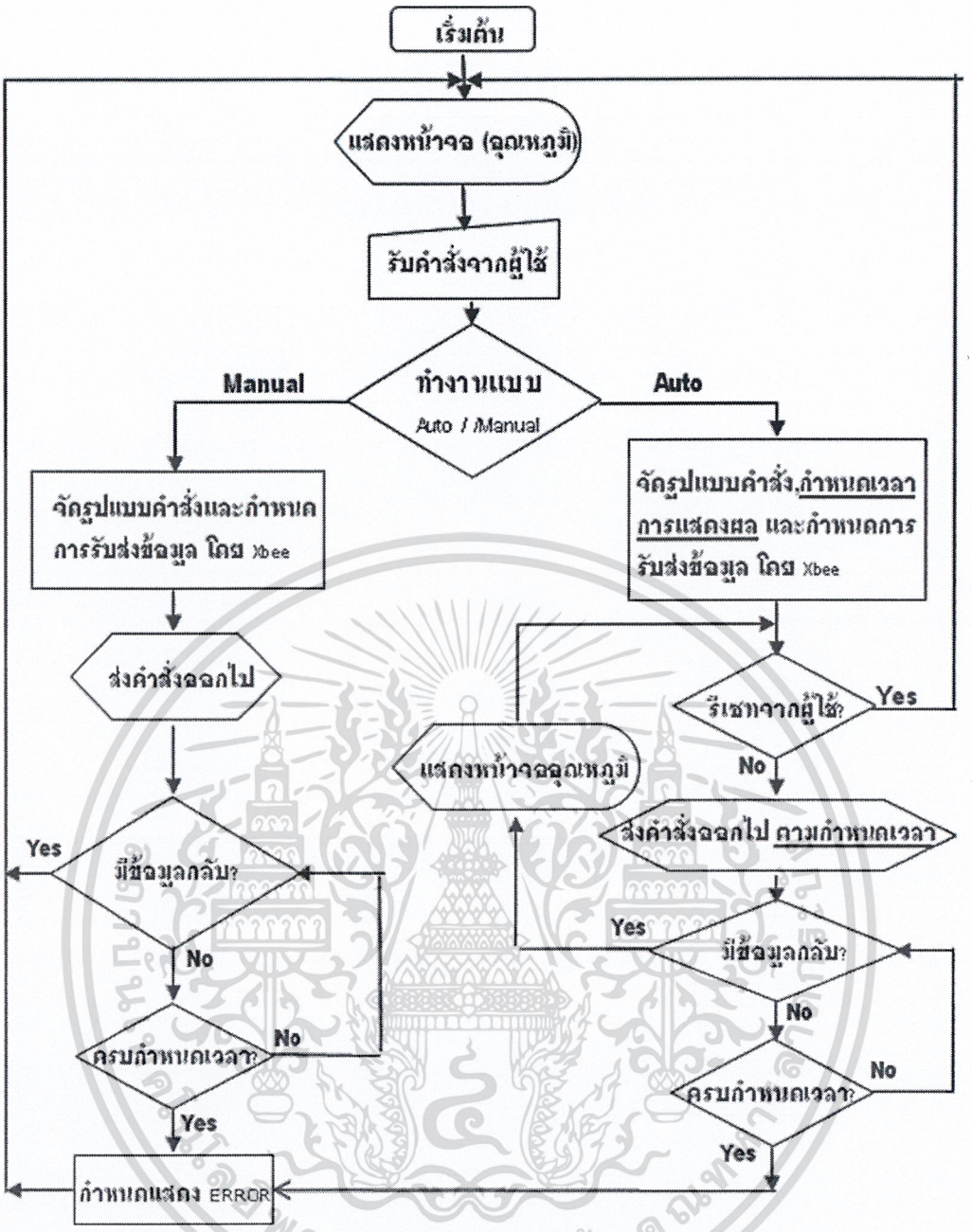
ซึ่งจะได้เส้นทาง คือ #2 → #3 → #4 ดังนั้นคำสั่งเริ่มต้นที่ตัว PC ก็จะเป็นดังต่อไปนี้ สังเกตว่า จะไม่มีหมายเลขโมดูลที่รับคำสั่งนี้คือ 0x32 (ตัวที่สอง)

ตารางที่ 3.2 ตัวอย่างของรูปแบบของคำสั่งในแต่ละไบท์

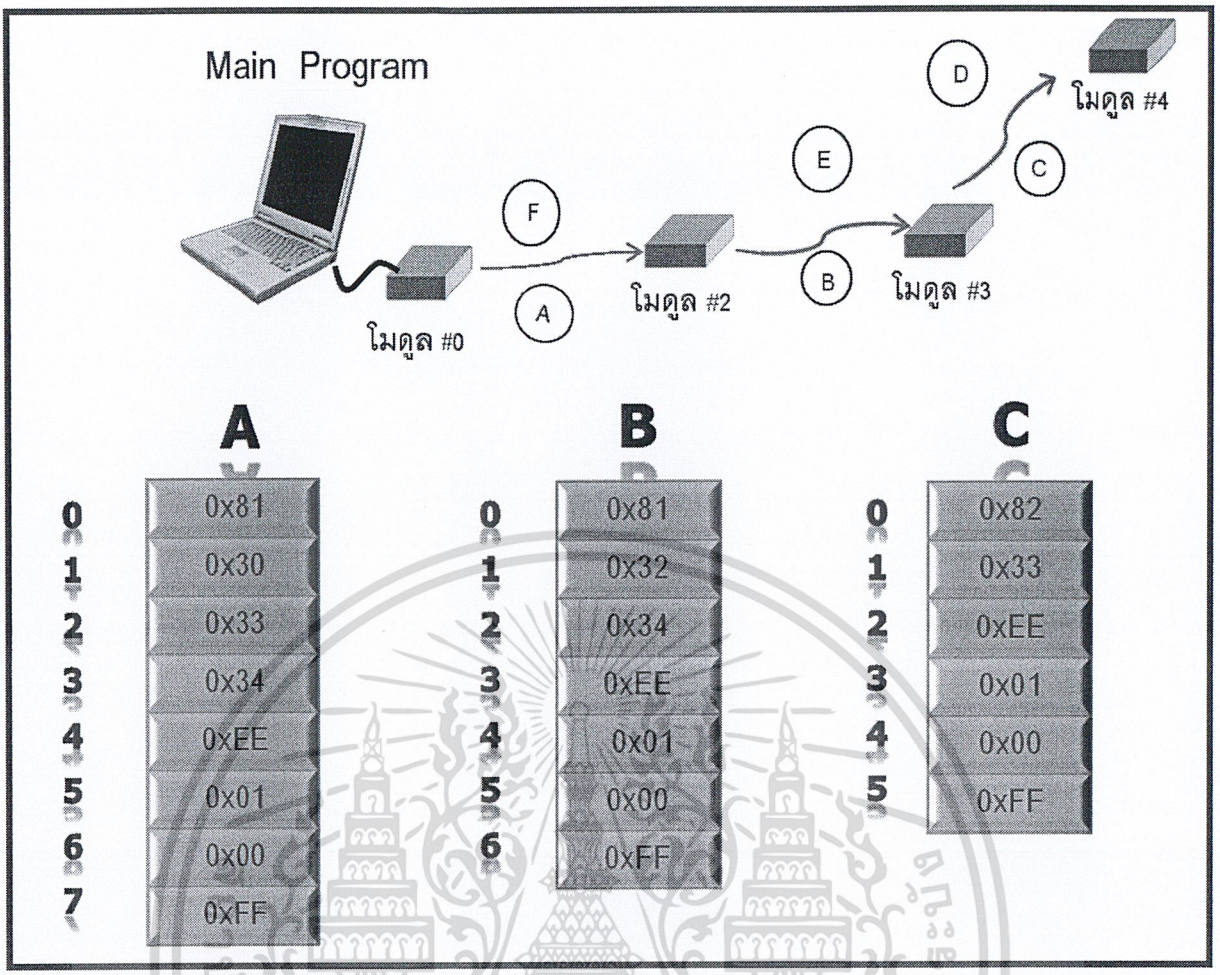


โปรแกรมส่วนของ PC (เขียนด้วย C# , GUI)

ในรูปแบบของคำสั่งจากด้านบนก็จะถูกจัดการสร้างขึ้นเองจากส่วนของโปรแกรมบน PC ซึ่งมีการเชื่อมต่อกับโมดูลเบอร์ศูนย์ (#0) อยู่แล้ว ดังนั้นโปรแกรมส่วนนี้ ก็จะต้องจัดการตั้งเบอร์ XBee-PRO บนโมดูล #0 ให้มีค่าเป็น MY = 00, DL = 02 เอง จากนั้นจึงส่งคำสั่งดังกล่าวออกมา ดังนั้น Flowchart ที่ควรจะเป็นจึงแสดงดังนี้



รูปที่ 3.4 FlowChart ของโปรแกรมหลัก (Main) บน PC



รูปที่ 3.5 รูปแสดงการส่งคำสั่งจาก PC เพื่ออ่านอุณหภูมิและสั่งงานรีเลย์

หมายเหตุ หากเป็นการเดินทางตามเส้นทางแสดงข้างบน ก่อนอื่นต้อง set ตัว XBee #0 ให้มี MY = 30 และ DL = 32 เพื่อเป็นการกำหนดการติดต่อให้ติดต่อกับตัวที่ XBee#2 เท่านั้น ส่วนตัวอื่นๆ ก็ set เบอร์ตัวเอง ส่วน DL ไปติดต่อกับ XBee #0 ก็ได้

ที่จุด A โมดูล #2 รับคำสั่งมาก็จะเก็บเลขเบอร์ผู้ส่งมาคือ 0x30 ไว้ แล้วก็จะเซทตัวเอง (XBee-PRO) ให้มีการส่งต่อไปที่โมดูล #3 (คือ DL = 0x33) และใส่เบอร์ตัวเองในไบท์ที่ 1 แทน ส่วนไบท์ที่ 3 เดิมก็ขยับมาเป็นไบท์ที่ 2 แทน (ขยับขึ้นมาทุกตัวจนหมดไบท์ที่บรรจุเลข 0xFF ดังรูป)

ที่จุด B โมดูล #3 ก็จะทำเช่นเดียวกันกับโมดูล #2 จะเห็นได้ว่าคำสั่งจะสั้นลง (เหลืออยู่ 7 ไบท์)

ที่จุด C ที่โมดูล #3 ส่งไปหาโมดูล #4 ก็ทำเช่นเดียวกัน จะเห็นได้ว่าไบท์ที่ 0 จะเปลี่ยนเป็น 0x82 เพราะได้ตรวจพบว่าเป็นการตัวโมดูล #4 เป็นตัวสุดท้ายหรือตัวที่จะถูกอ่านอุณหภูมิ (ตรวจดูที่ รหัส 0xEE) นั่นเอง

ดังนั้นจะเห็นได้ว่าไม่ว่าจะมีโมดูลกี่ตัว หากเรากำหนดเส้นทางวิ่งได้ถูกต้องก็สามารถอ่านค่าอุณหภูมิได้ ทุกตัวอย่างแม่นยำ

3.3 หลักการทำงานของเครื่อง Client

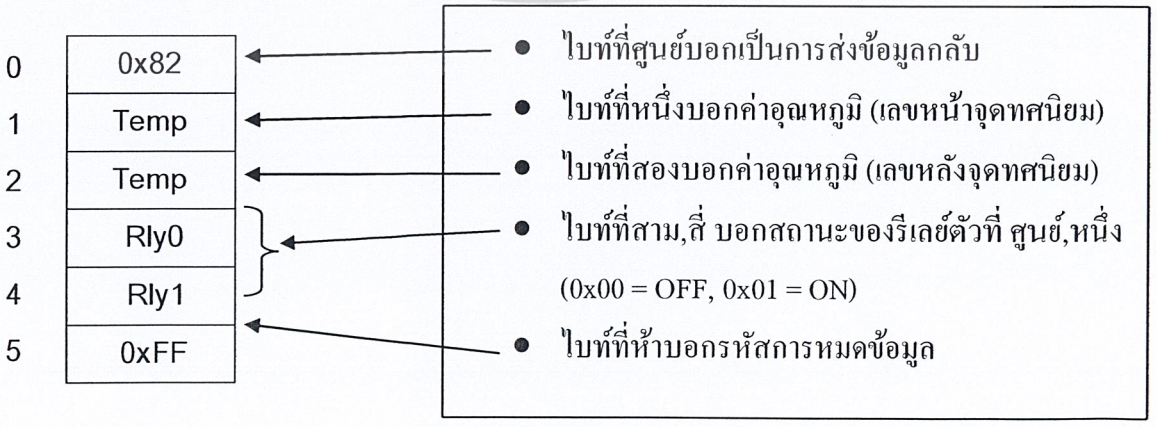
โปรแกรมรอง (client) #1,#2,#3,#4,... (C บน Microcontroller dsPIC30F4011) โปรแกรมส่วนนี้ทำหน้าที่อยู่ 3 อย่าง

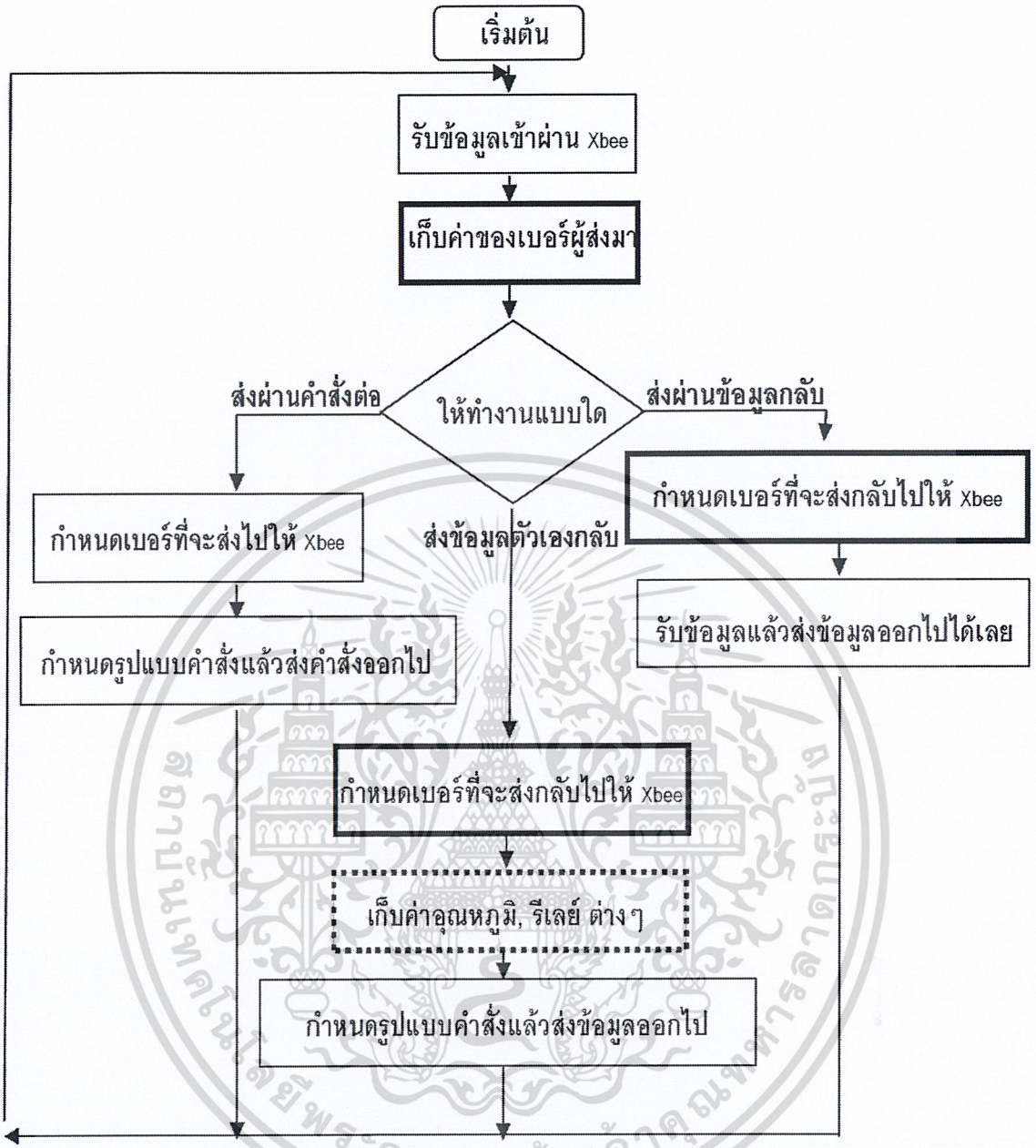
1. ทำหน้าที่ส่งต่อคำสั่ง เมื่อรับคำสั่งมาแล้วรู้ว่าไม่ใช่คำสั่งที่จะทำการอ่านค่าจากโมดูลตัวเอง ก็จะจัดรูปแบบของคำสั่งใหม่แล้วส่งคำสั่งนี้ต่อไปสู่โมดูลตัวอื่นตามที่ระบุมากับคำสั่งนั้น (มีการเก็บเบอร์โมดูลที่รับคำสั่งไว้ด้วยเพื่อใช้ในกรณีตอนส่งข้อมูลกลับ)
2. ทำหน้าที่ส่งต่อข้อมูลกลับ ในช่วงตอนส่งข้อมูลกลับไปสู่ PC เมื่อรับข้อมูลที่จะส่งกลับมา (จากตัวโมดูลอื่น) ถึงตัวเอง ก็จะทำการส่งข้อมูลนั้นต่อให้กลับไปตามเส้นทางตอนขามา เพราะได้มีการเก็บเบอร์ของโมดูลไว้แล้ว
3. ทำหน้าที่อ่านค่าอุณหภูมิและ อ่าน/สั่ง สถานะรีเลย์ของตนเอง ในส่วนนี้จะทำการอ่านค่าอุณหภูมิของตอนเองตามคำสั่งที่ระบุมา โดยอาจมีการอ่านสถานะหรือสั่งรีเลย์ให้ทำงานด้วยก็ได้ (ขึ้นอยู่กับคำสั่ง) จากนั้นก็จะจัดข้อมูลใหม่ส่งกลับไป (เพราะได้บันทึกเบอร์โมดูลที่จะส่งกลับไว้แล้ว ทำให้ส่งกลับได้อย่างถูกต้อง)

3.3.1 รูปแบบของข้อมูลที่ส่งกลับ

ในการส่งข้อมูลกลับนั้น เนื่องจากการทดลองครั้งนี้ได้มีการส่งข้อมูลอยู่สองอย่างคือ อุณหภูมิจำนวน 1 ช่อง และสถานะของรีเลย์จำนวน 2 ตัว (ซึ่งในทางปฏิบัติเราสามารถเพิ่มตัววัดอะไรเข้าไปก็ได้ไม่จำกัด) ดังนั้นจึงมีช่องสำหรับส่งค่าดังกล่าวจำนวน 6 ไบท์ ดังรูปข้างล่าง

ตารางที่ 3.3 รูปแบบของข้อมูลที่ส่งกลับ



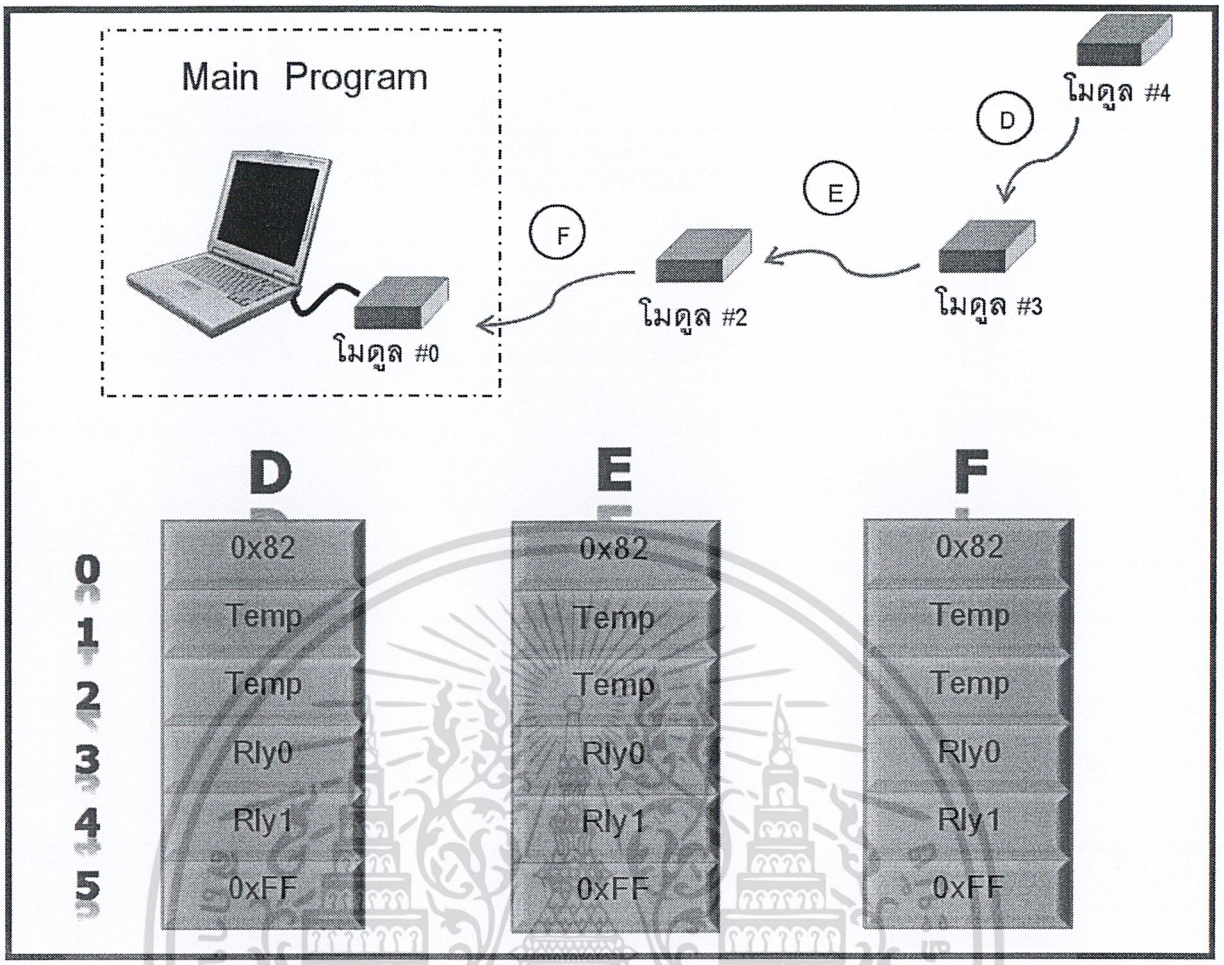


รูปที่ 3.6 Flowchart ของโปรแกรมรอง (Client) บน MCU

โปรแกรมส่วนของโมดูล (ทุกตัวเหมือนกัน)

ทำงาน 3 โหมด คือ

1. ส่งผ่านคำสั่งต่อ
2. ส่งผ่านข้อมูลกลับ
3. ส่งข้อมูลของตัวเองกลับ



รูปที่ 3.7 แสดงผลการเดินทางของข้อมูลกลับมาในแต่ละช่วง

จะเห็นได้ว่าข้อมูลไม่ได้มีการปรับปรุงเปลี่ยนแปลงอะไร เพราะโมดูลในเส้นทางกลับทุกตัวสามารถส่งกลับได้ถูกเส้นทาง (ถูกโมดูล) เพราะในตอนช่วงส่งต่อคำสั่งก็ได้มีการบันทึกเก็บตัวที่จะรับข้อมูลในตอนส่งข้อมูลกลับไว้แล้วทุกโมดูลดังได้กล่าวมาแล้วตอนต้น

บทที่ 4

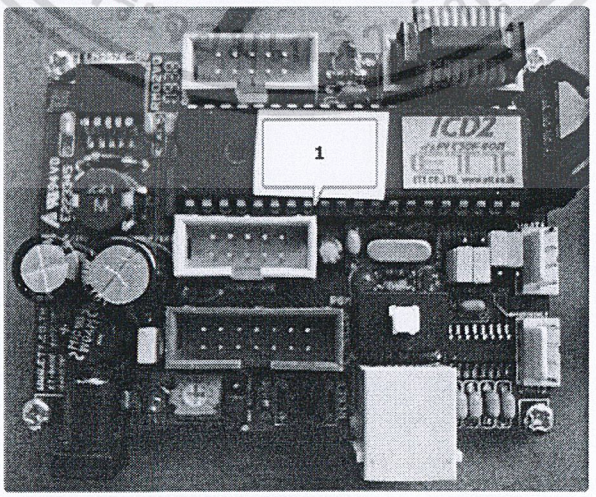
การทดลองและผลการทดลอง

4.1 อุปกรณ์ที่ใช้ในการการทดลอง

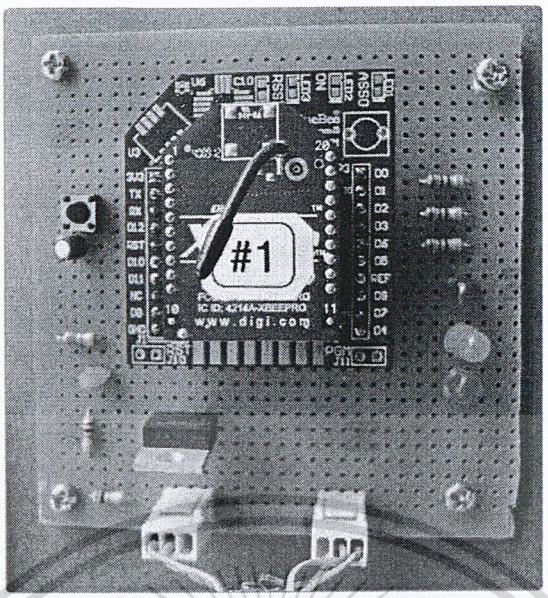
4.1.1 อุปกรณ์ภาคส่ง (Main)

รูปที่ 4.1 บอร์ด XBee-PRO Main กับ พอร์ตอนุกรม RS-232

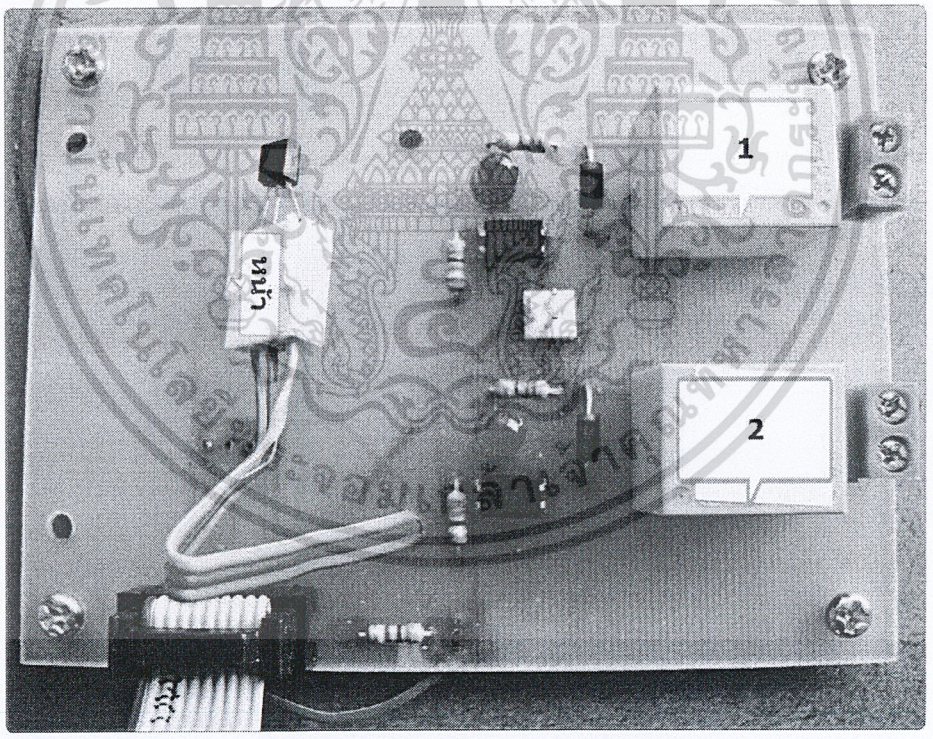
4.1.2 อุปกรณ์ภาครับ (Client)



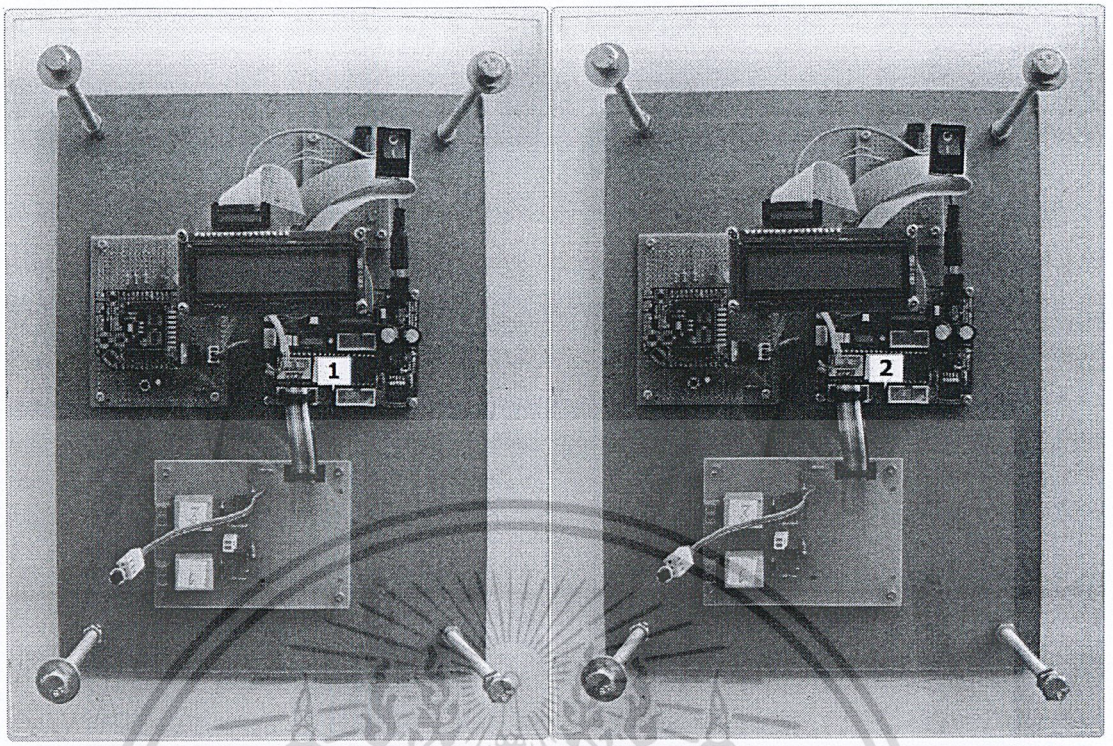
รูปที่ 4.2 บอร์ด dsPIC30F4011



รูปที่ 4.3 โมดูล XBee-PRO

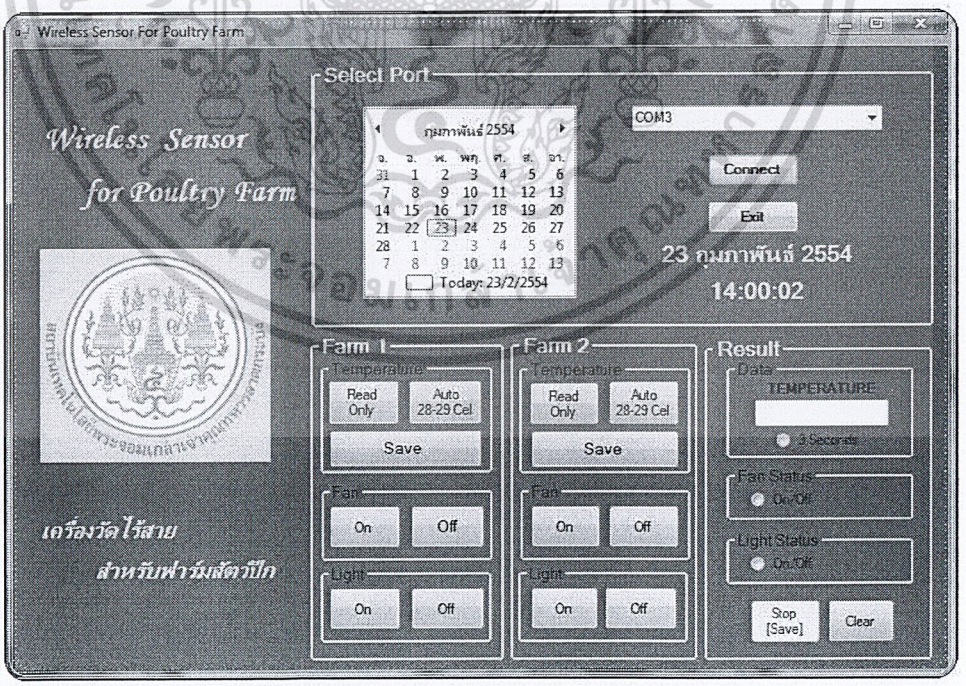


รูปที่ 4.4 บอร์ดเซ็นเซอร์วัดอุณหภูมิ และอุปกรณ์รีเลย์



รูปที่ 4.5 โมเดลอุปกรณ์ชุดที่ 1 และชุดที่ 2

4.1.3 หน้าจอโปรมแกรม GUI

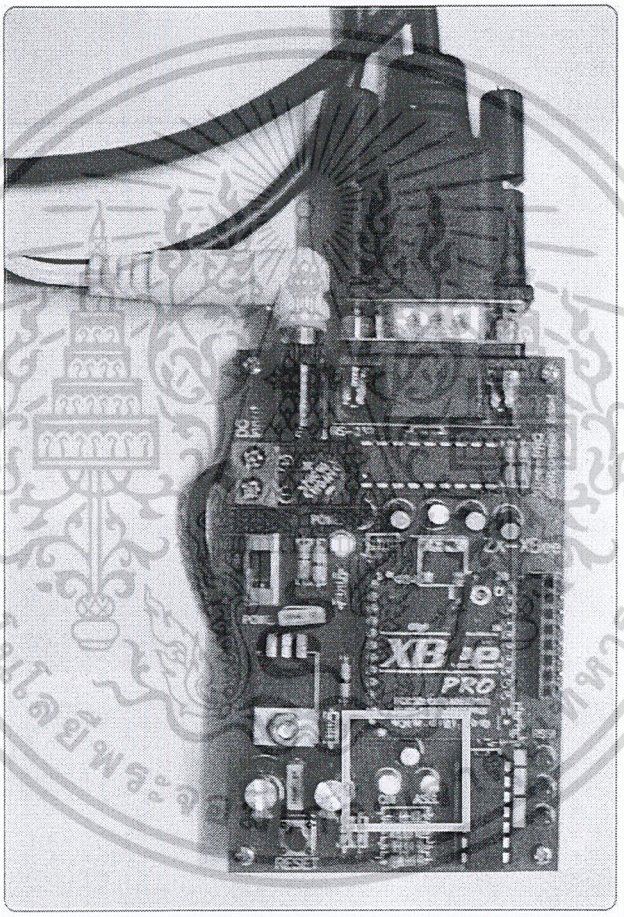


รูปที่ 4.6 หน้าจอโปรแกรม GUI

4.2 การตั้งค่าโมดูล XBee-PRO

4.2.1 การติดต่อโมดูล XBee-PRO กับคอมพิวเตอร์

1) เชื่อมต่อบอร์ดไมโครคอนโทรลเลอร์ dsPIC30F4011 กับพอร์ตอนุกรมอาร์เอส-232 (RS-232) จากนั้นจ่ายไฟให้กับบอร์ดไมโครคอนโทรลเลอร์และเมื่อใส่โมดูล XBee-PRO ลงไปแล้วไฟ LED ที่ตำแหน่ง ON จะติด และที่ตำแหน่ง ASS. จะกระพริบ แสดงสถานะพร้อมใช้งานของโมดูล XBee-PRO

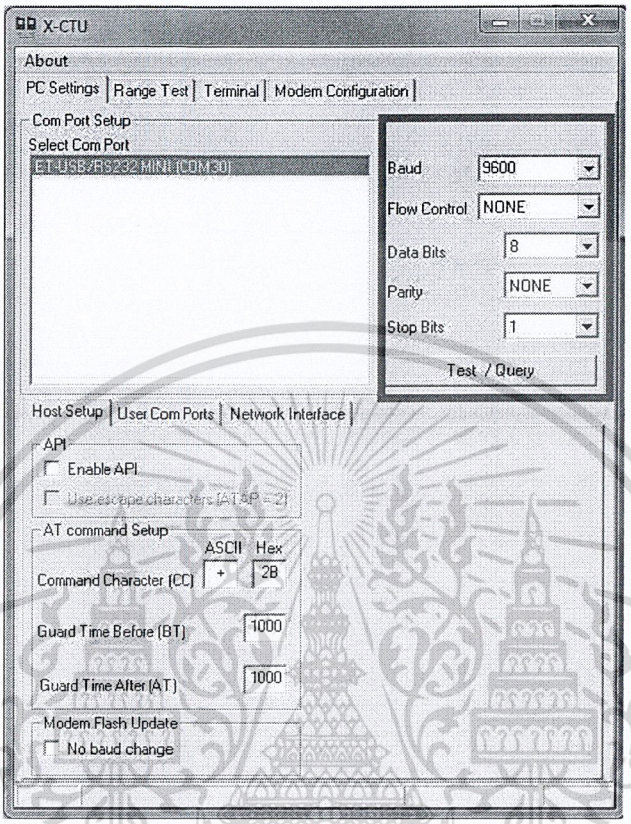


รูปที่ 4.7 แสดงไฟสถานะของโมดูล XBee-PRO

2) กำหนดการเชื่อมต่อพอร์ตคอมพิวเตอร์กับโมดูล XBee-PRO โดยใช้โปรแกรม X-CTU ทำการเลือกคอมพอร์ต และกำหนดค่าต่างๆ ดังนี้

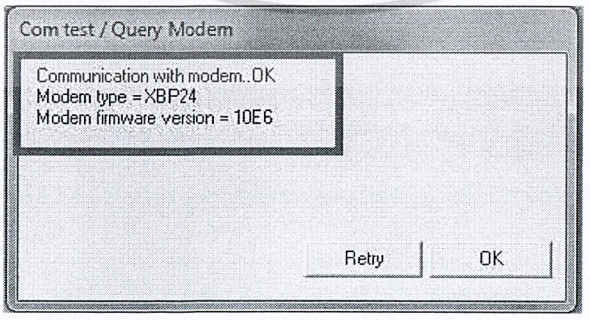
- อัตราบอร์ด (Baudrate) ที่ 9600 bps (ซึ่งค่านี้ต้องตรงกับค่าที่กำหนดไว้ในไมโครคอนโทรลเลอร์ ถ้ากำหนดไม่ตรงตัว Coordinator กับอุปกรณ์ End Device จะไม่สามารถทำการติดต่อสื่อสารกันได้)

- Data Bits เป็น 8 บิต
- Parity หรือบิตตรวจสอบ เป็น NONE (ไม่มีการตรวจสอบ)
- Stop Bits เป็น 1 บิต



รูปที่ 4.8 กำหนดการเชื่อมต่อพอร์ตคอมพิวเตอร์กับโมดูล XBee-PRO โดยใช้โปรแกรม X-CTU

3) จากนั้นกด Test เพื่อทดสอบการติดต่อระหว่าง XBee-PRO กับโปรแกรมว่าสามารถติดต่อกันได้หรือไม่ เมื่อเชื่อมต่อกันได้จะปรากฏหน้าต่างดังรูป



รูปที่ 4.9 แสดงให้เห็นว่าสามารถติดต่อกันได้ระหว่าง ZigBee กับโปรแกรม X-CTU

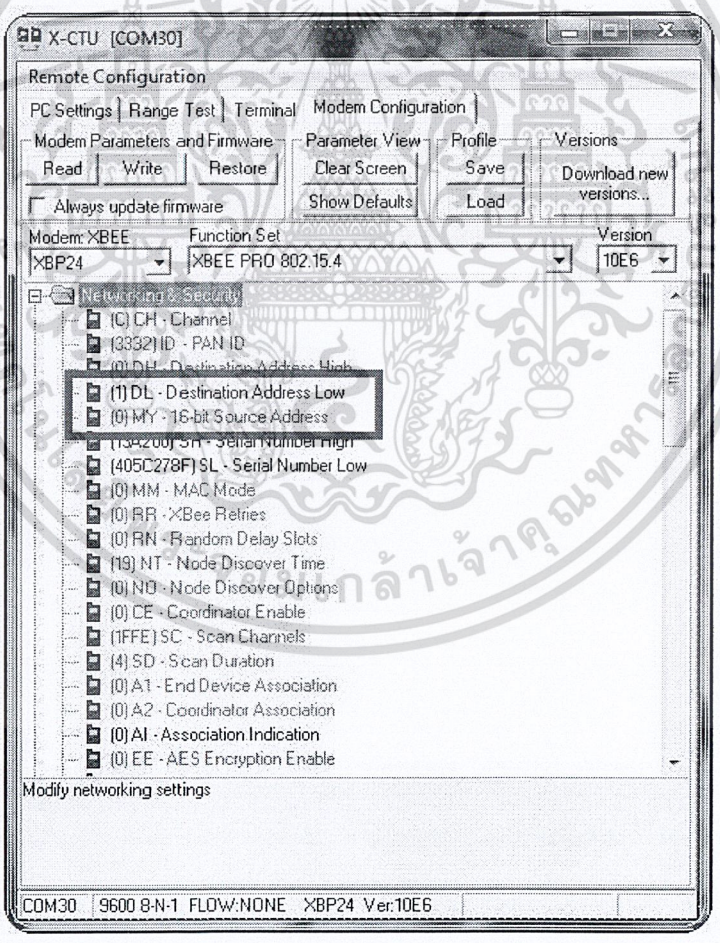
4.2.2 การตั้งค่าคอนฟิกูเรชั่นให้แก่โมดูล XBee-PRO

4.2.2.1 การกำหนดพารามิเตอร์ให้กับ Coordinator

ทำการกำหนดค่าพารามิเตอร์ให้กับ XBee-PRO ภาครับ ซึ่งทำงานเป็น Coordinator โดยที่โมดูล XBee-PRO ไม่จำเป็นต้องกำหนดในส่วนของ Function Set แต่หากเป็น XBee ธรรมดา ต้องกำหนดการทำงานด้วยว่าจะให้เป็น Coordinator หรือ End Device

ในการติดต่อสื่อสารระหว่างโมดูล XBee-PRO นั้นมีค่าพารามิเตอร์ที่ใช้ในการจัดการเกี่ยวกับเครือข่าย (Networking) ซึ่งมีพารามิเตอร์สำคัญๆ ที่ต้องกำหนดค่า ดังนี้

- DL (Destination Address Low) กำหนดเป็น 1 (เป็นค่า SL ของโมดูลตัวส่ง) เพราะมี client ทั้งหมด 2 ชุด
- MY (16-bit Source Address) กำหนดเป็น 0



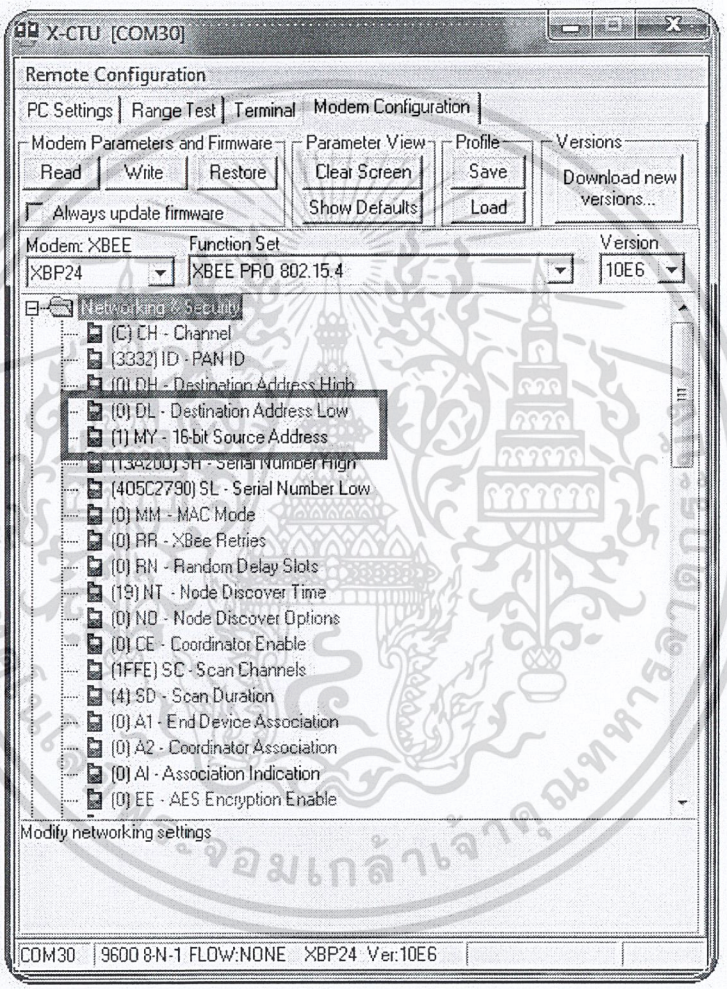
รูปที่ 4.10 การกำหนดพารามิเตอร์ให้กับ Coordinator ที่ภาครับให้เป็นเครื่อง Main

4.2.2.2 การกำหนดพารามิเตอร์ให้กับ End Device

สำหรับ XBee-PRO ที่ภาคส่งจะมีทั้งหมด 2 Client ทำงานเป็น End Device โดย มี พารามิเตอร์ที่ต้องกำหนดคล้ายคลึงกับ ZigBee ที่ภาครับ ดังนี้

Client 1 (Farm 1)

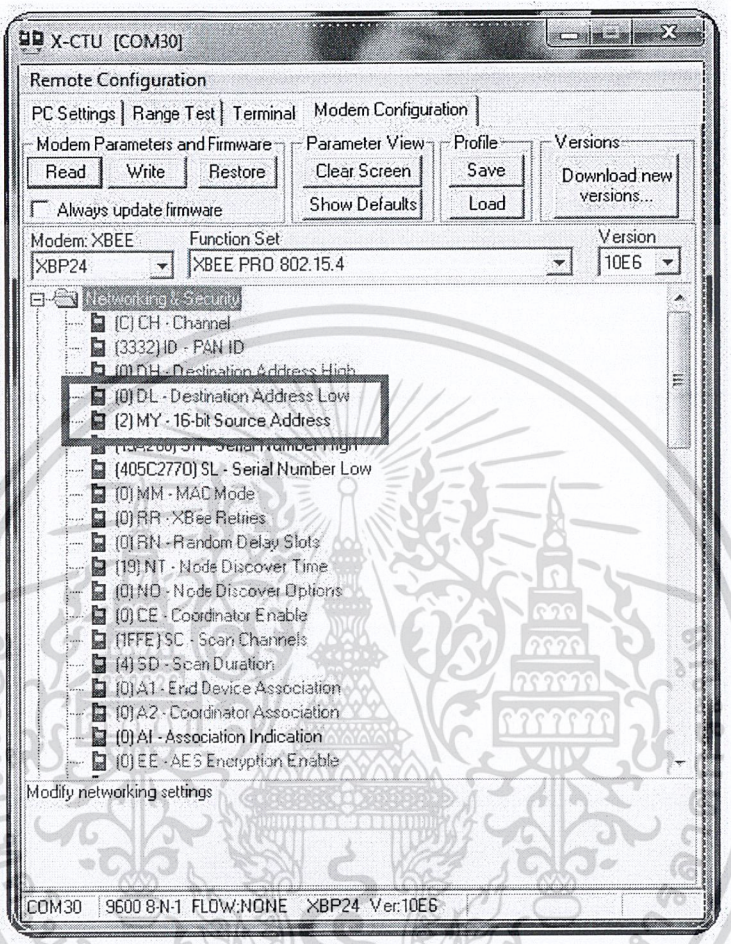
- DL (Destination Address Low) กำหนดเป็น 0 (เป็นค่า SL ของโมดูลตัวรับ)
- MY (16-bit Source Address) กำหนดเป็น 1



รูปที่ 4.11 การกำหนดพารามิเตอร์ให้กับ End Device ที่ภาคส่งให้เป็นเครื่อง Client 1

Client 2 (Farm 2)

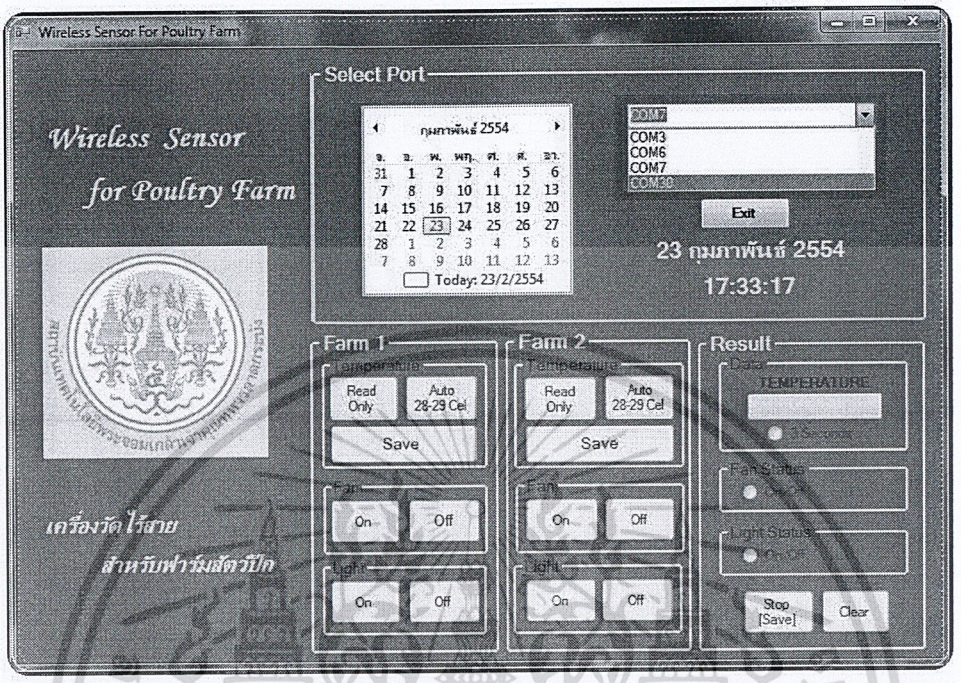
- DL (Destination Address Low) กำหนดเป็น 0 (เป็นค่า SL ของโมดูลตัวรับ)
- MY (16-bit Source Address) กำหนดเป็น 2



รูปที่ 4.12 การกำหนดพารามิเตอร์ให้กับ End Device ที่ภาคส่งให้เป็นเครื่อง Client 2

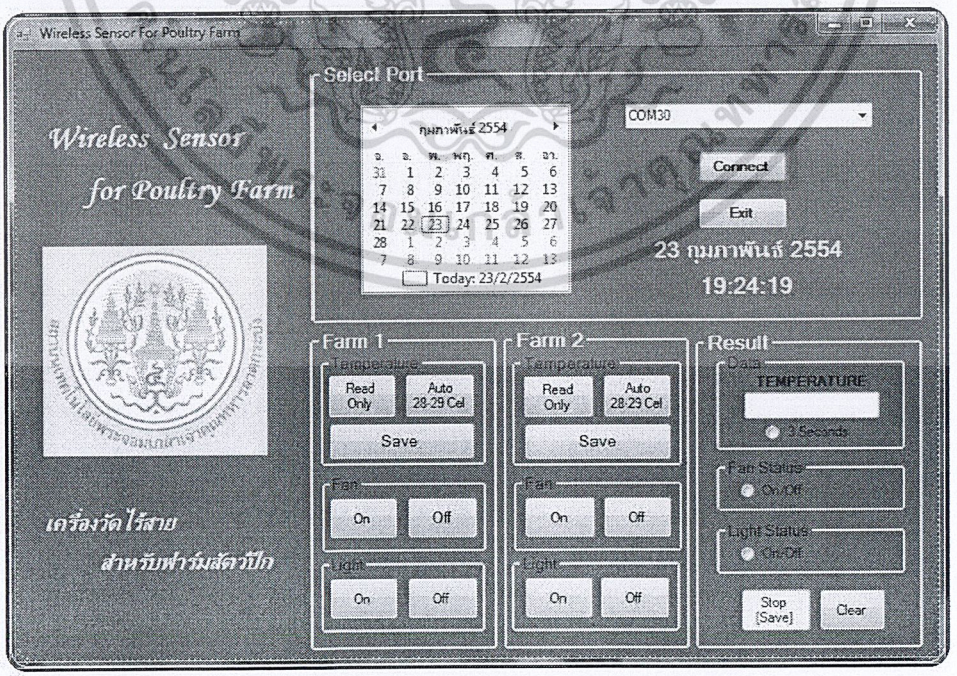
4.3 ขั้นตอนการใช้งาน

1) เปิดหน้าจอโปรแกรม GUI ไปที่ช่อง Select Port ให้เลือกหมายเลขพอร์ตให้ตรงกับพอร์ตอนุกรมอาร์เอส-232 (RS-232) ที่เชื่อมต่อไว้



รูปที่ 4.13 แสดงการเลือกพอร์ตอนุกรมเพื่อเชื่อมต่อ

2) กดปุ่ม Connect เพื่อเชื่อมต่อ ระบบนี้จะทำงานก็ต่อเมื่อได้ทำการเชื่อมต่อระหว่างโปรแกรมการใช้งานกับชุดเครื่องวัดไร้สาย ซึ่งจะมีลักษณะดังรูปที่ 4.14

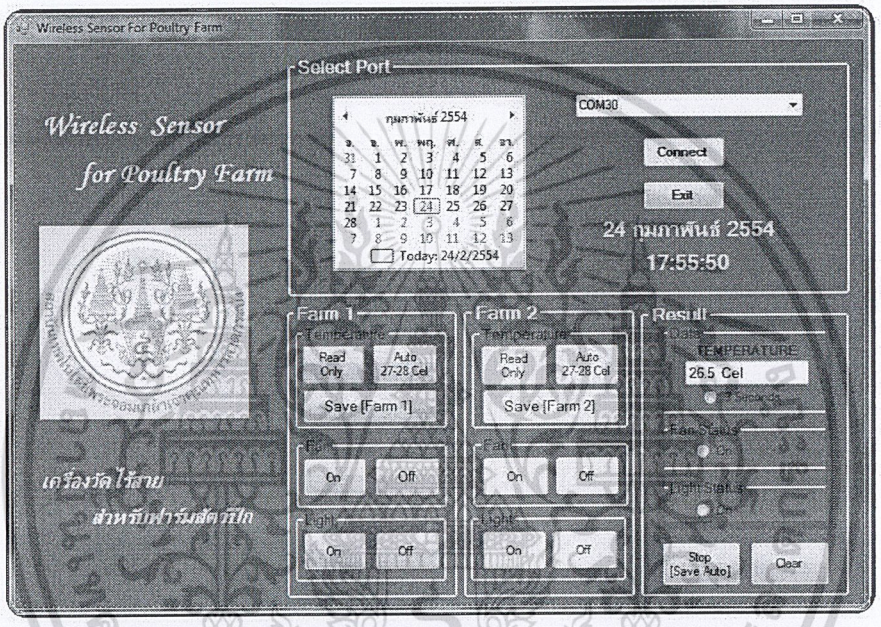


รูปที่ 4.14 แสดงการเชื่อมต่อเพื่อสั่งงานไมโครคอนโทรลเลอร์

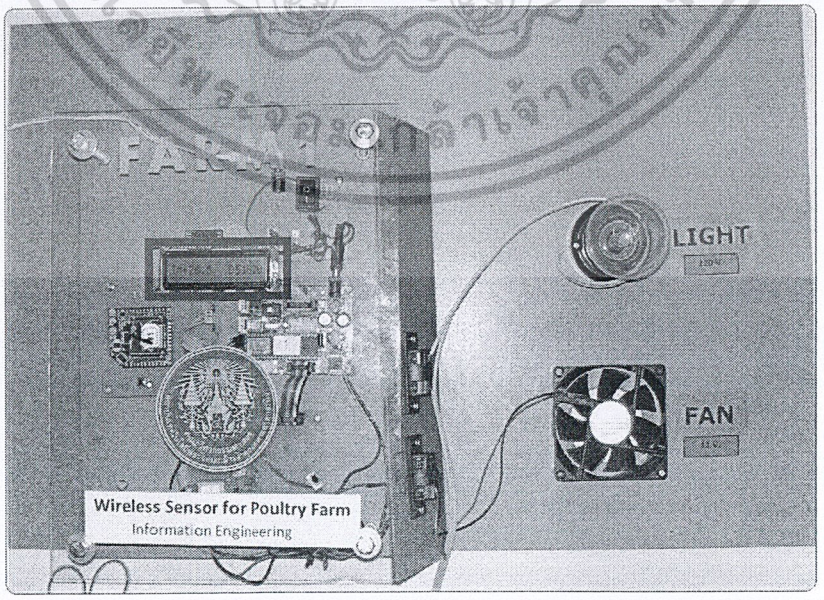
3) ทำการทดลองรับข้อมูลอุณหภูมิ สั่งการรีเลย์ และตรวจดูสถานะรีเลย์ โดยทำการทดลองทีละฟาร์ม

- Farm 1

3.1) ทดลองส่งรับข้อมูลอุณหภูมิและตรวจดูสถานะรีเลย์ จากฟาร์มที่ 1 โดยไปที่ช่อง Farm 1 ให้เลือกกดปุ่ม Read Only โมดูล XBee-PRO ภาคส่ง จะส่งคำสั่งไปยังโมดูล XBee-PRO ภาครับให้รับค่าอุณหภูมิและสถานะรีเลย์กลับมา โดยที่ช่องแสดงผล TEMPERATURE จะแสดงค่าอุณหภูมิที่วัดได้ และที่ช่อง Fan Status และ Light Status ก็จะแสดงสถานะรีเลย์ในขณะนั้นกลับมาให้ผู้ใช้ทราบ

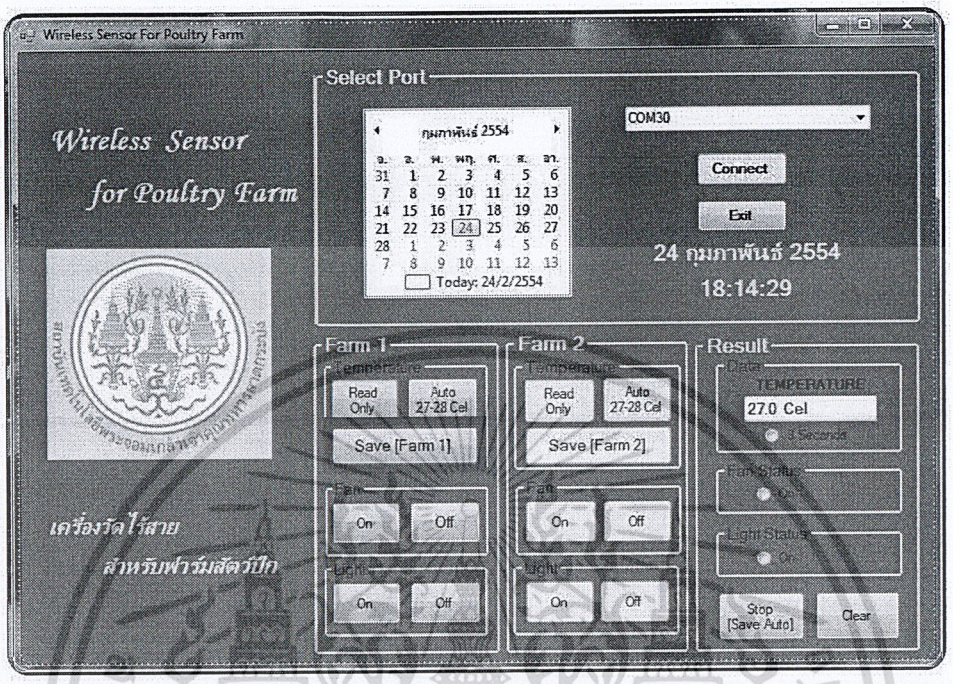


รูปที่ 4.15 แสดงอุณหภูมิและสถานะรีเลย์จากฟาร์มที่ 1 จากหน้าจอโปรแกรม GUI

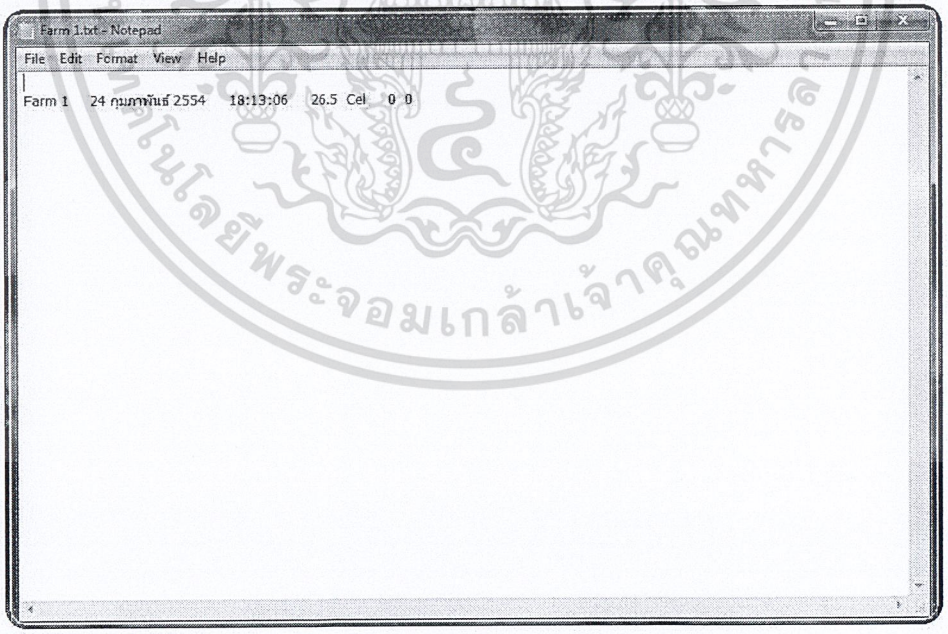


รูปที่ 4.16 แสดงอุณหภูมิและสถานะรีเลย์ของฟาร์มที่ 1

3.2) ทำการบันทึกข้อมูลอุณหภูมิและสถานะรีเลย์ที่ตรวจวัดได้ จากฟาร์มที่ 1 ณ เวลาและวันที่ที่กดบันทึก โดยไปที่ช่อง Farm 1 ให้เลือกกดปุ่ม Save [Farm 1] โดยข้อมูลที่บันทึกจะถูกเก็บ เป็น Text file อยู่ใน Folder ที่สร้างเอาไว้

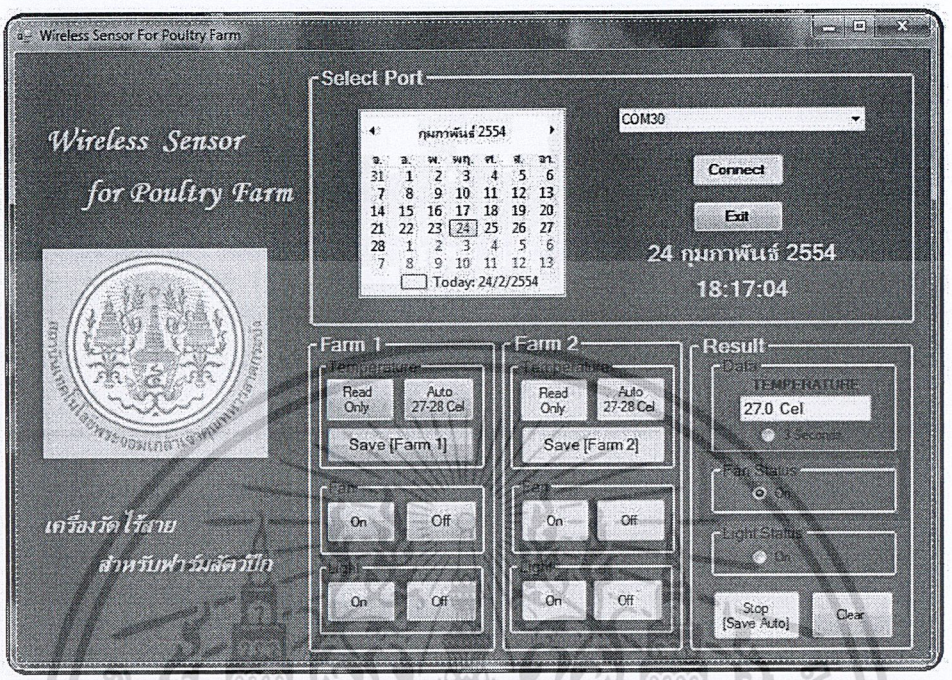


รูปที่ 4.17 แสดงการเลือกบันทึกข้อมูลจากฟาร์มที่ 1

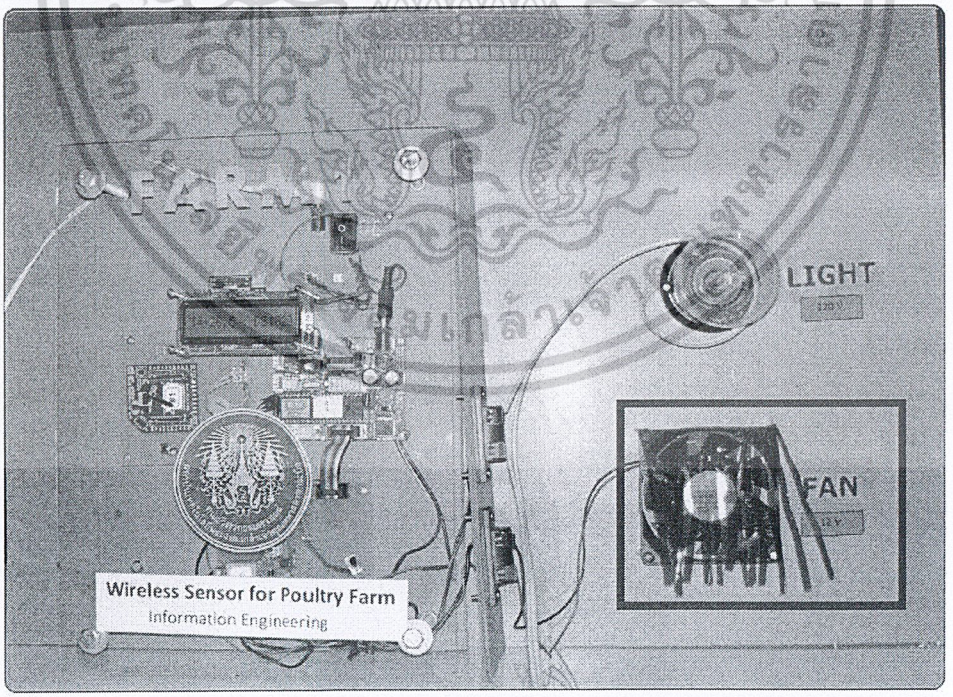


รูปที่ 4.18 แสดงข้อมูลอุณหภูมิและสถานะรีเลย์ที่บันทึกได้จากฟาร์มที่ 1

3.3) ทดลองเปิด-ปิดอุปกรณ์ไฟฟ้ารีเลย์ 1 ในการทดลองเลือกใช้อุปกรณ์พัดลม 12V โดยจากฟาร์มที่ 1 ไปที่ช่อง Fan ให้เลือกกดปุ่ม On สถานะรีเลย์ที่ช่อง Status Relay1 จะมีไฟติดแสดงสถานะว่ารีเลย์เปิดอยู่

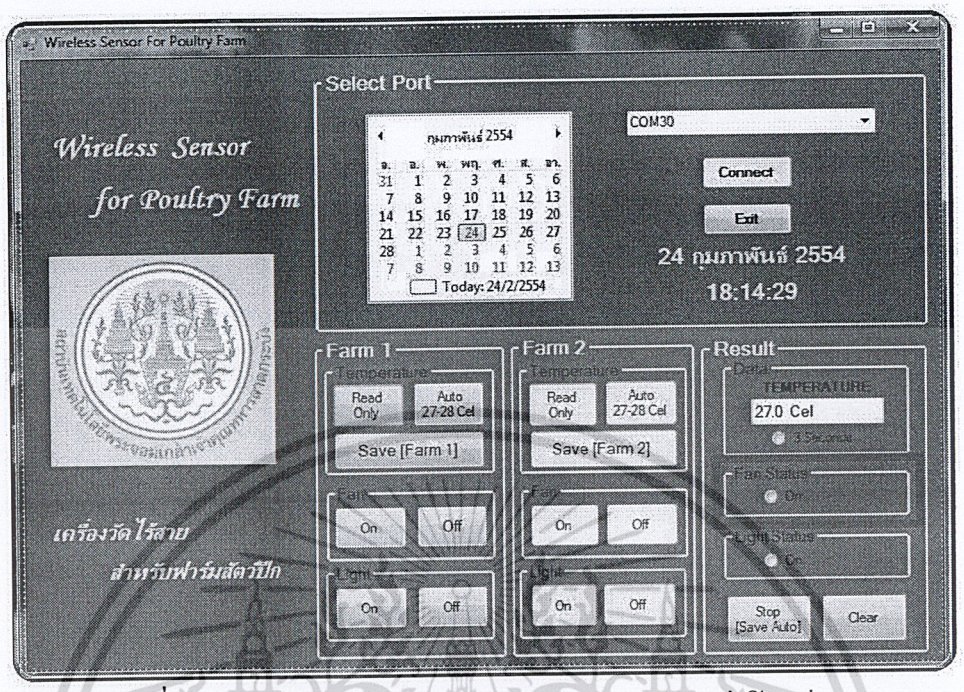


รูปที่ 4.19 หน้าจอโปรแกรม GUI แสดงสถานะพัดลมว่าเปิดอยู่

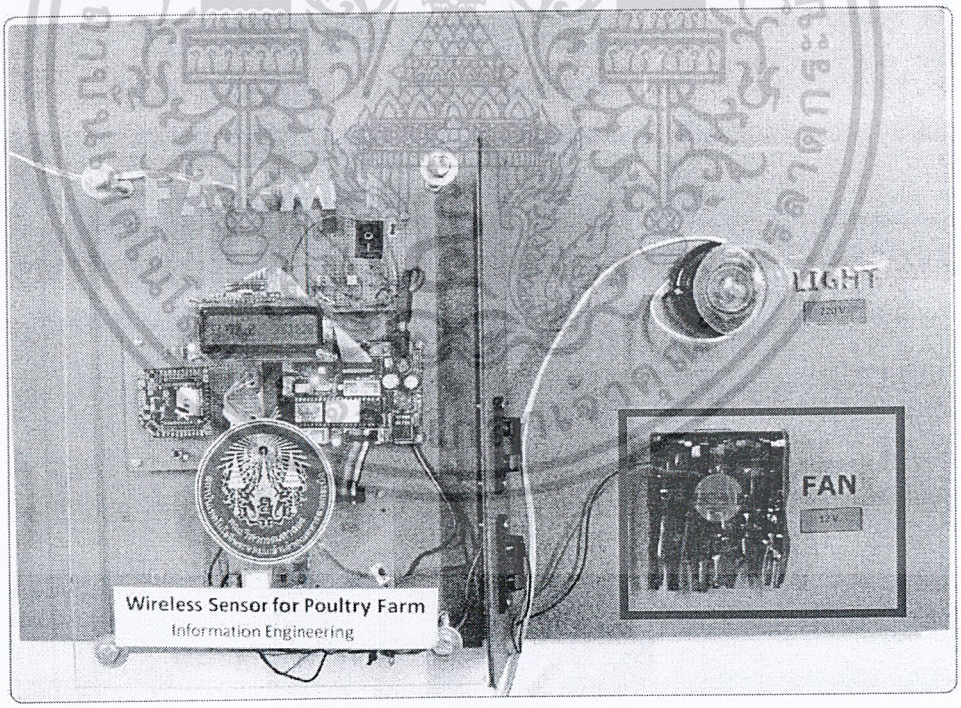


รูปที่ 4.20 แสดงการเปิดอุปกรณ์ไฟฟ้ารีเลย์ 1 ไปยังฟาร์มที่ 1

3.4) ทดลองปิดอุปกรณ์ไฟฟ้ารีเลย์ 1 โดยจากฟาร์มที่ 1 ไปที่ช่อง Fan ให้เลือกกดปุ่ม Off สถานะรีเลย์ที่ช่อง Status Relay1 จะไม่มีไฟติดแสดงสถานะว่ารีเลย์ปิดแล้ว

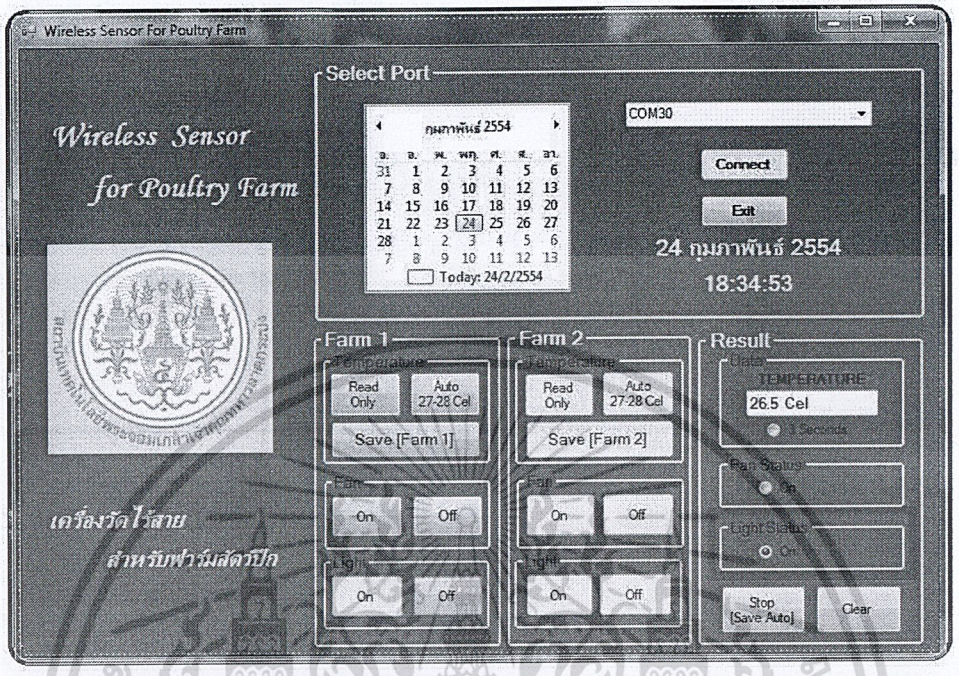


รูปที่ 4.21 หน้าจอโปรแกรม GUI แสดงสถานะพัดลมว่าปิดอยู่

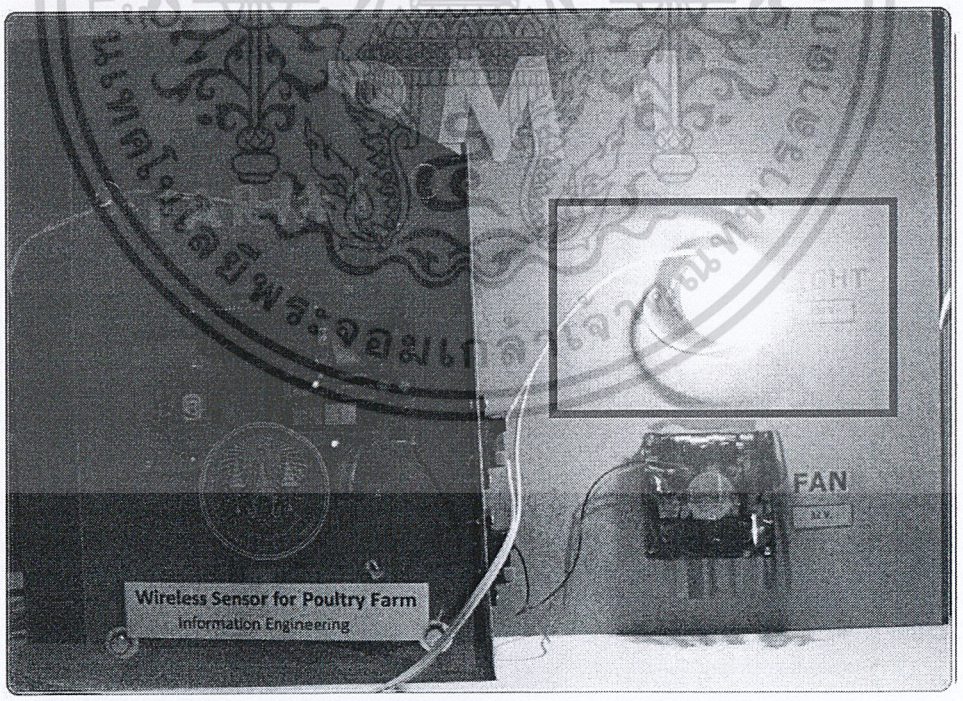


รูปที่ 4.22 แสดงการปิดอุปกรณ์ไฟฟ้ารีเลย์ 1 ไปยังฟาร์มที่ 1

3.5) ทดลองเปิด-ปิดอุปกรณ์ไฟฟ้ารีเลย์ 2 ในการทดลองเลือกใช้อุปกรณ์หลอดไฟฟ้า 60 Watts โดยจากฟาร์มที่ 1 ไปที่ช่อง Light ให้เลือกกดปุ่ม On สถานะรีเลย์ที่ช่อง Status Relay2 จะมีไฟติดแสดงสถานะว่ารีเลย์เปิดอยู่

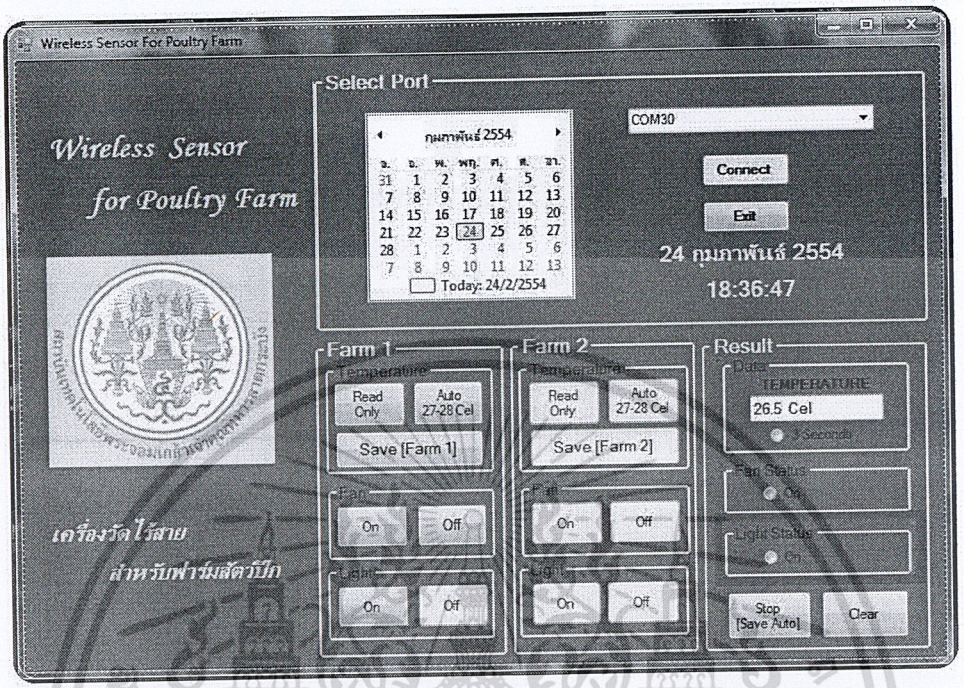


รูปที่ 4.23 หน้าจอโปรแกรม GUI แสดงสถานะไฟว่าเปิดอยู่

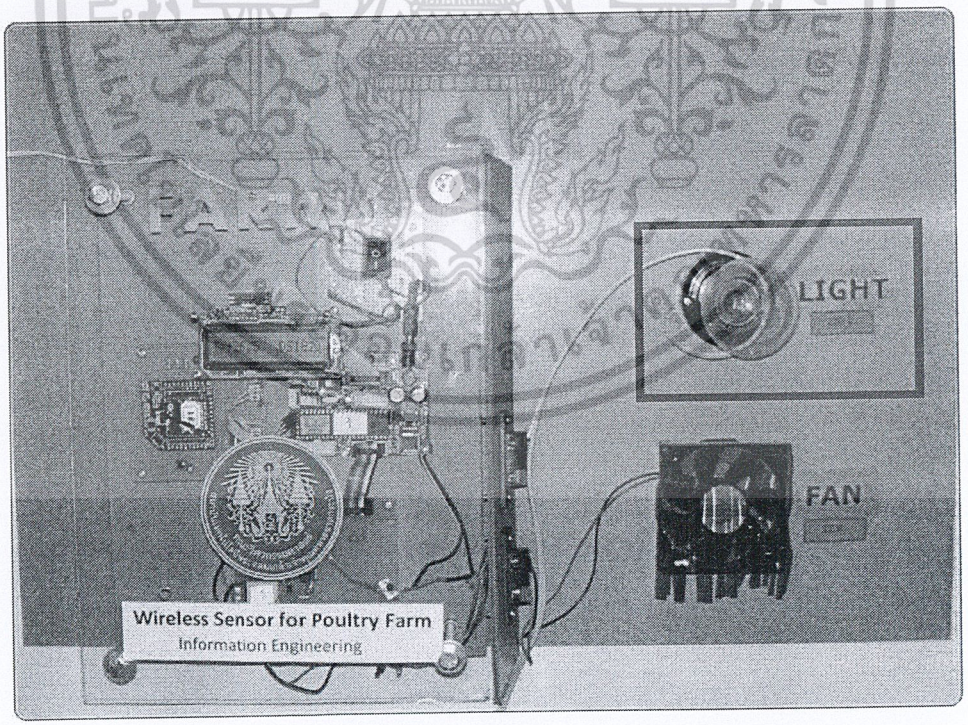


รูปที่ 4.24 แสดงการเปิดอุปกรณ์ไฟฟ้ารีเลย์ 2 ไปยังฟาร์มที่ 1

3.6) ทดลองปิดอุปกรณ์ไฟฟ้ารีเลย์ 2 ในการทดลองเลือกใช้อุปกรณ์หลอดไฟฟ้า 60 Watts โดยจากฟาร์มที่ 1 ไปที่ช่อง Light ให้เลือกกดปุ่ม Off สถานะรีเลย์ที่ช่อง Status Relay2 จะมีไม่มีไฟติดแสดงสถานะว่ารีเลย์ปิดแล้ว

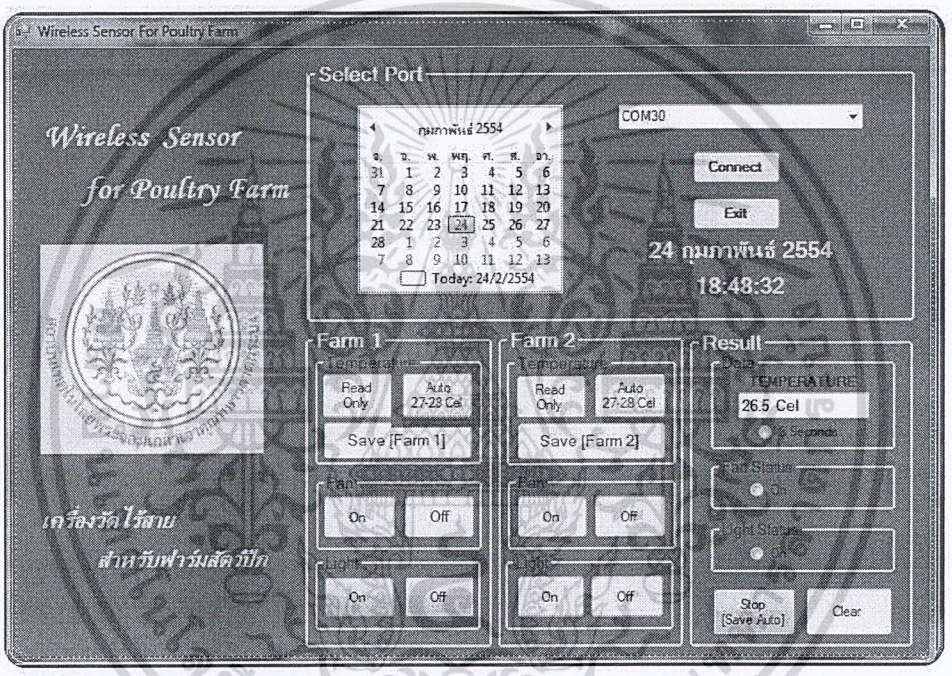


รูปที่ 4.25 หน้าจอโปรแกรม GUI แสดงสถานะไฟว่าปิดอยู่

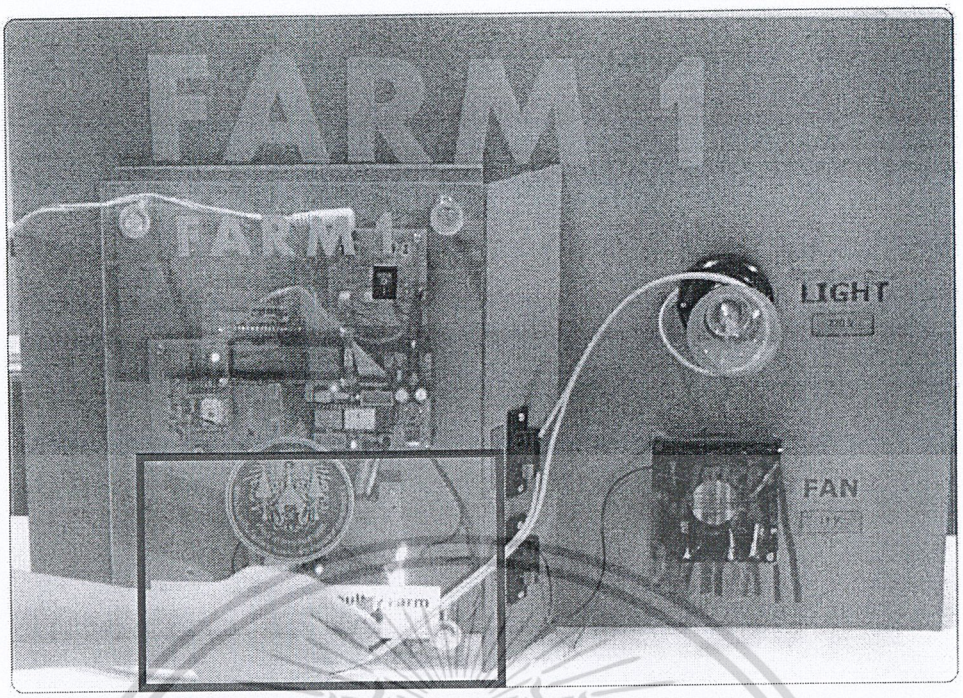


รูปที่ 4.26 แสดงการปิดอุปกรณ์ไฟฟ้ารีเลย์ 2 ไปยังฟาร์มที่ 1

3.7) ทดลองรับส่งข้อมูลอุณหภูมิและตรวจดูสถานะรีเลย์ จากฟาร์มที่ 1 โดยให้โปรแกรมทำการตรวจวัดอุณหภูมิแบบอัตโนมัติ ซึ่งในการทดลองตั้งโปรแกรมเวลาให้ตรวจวัดทุกๆ 3 วินาที โดยไปที่ช่อง Farm 1 ให้เลือกกดปุ่ม Auto 27.0-28.0 จากนั้นที่ช่องแสดงผล TEMPERATURE จะแสดงค่าอุณหภูมิที่วัดได้ทุกๆ 3 วินาที และที่ช่อง Fan Status , Light Status ก็จะแสดงสถานะรีเลย์ในขณะนั้นกลับมาแสดงทุกๆ 3 วินาที และมีการโปรแกรมให้อุปกรณ์ไฟฟ้ารีเลย์ทำการเปิด-ปิด อัตโนมัติ เมื่ออุณหภูมิสูงกว่า 27.5 องศาเซลเซียส ก็จะสั่งให้อุปกรณ์ไฟฟ้ารีเลย์ ซึ่งในการทดลองได้เลือกใช้อุปกรณ์พัดลม 12V ทำการเปิดจนกว่าอุณหภูมิจะต่ำกว่า 27.5 องศาเซลเซียส จึงจะทำการปิด ซึ่งเป็นอุปกรณ์ไฟฟ้ารีเลย์ 1 ส่วนอุปกรณ์ไฟฟ้ารีเลย์ 2 ได้เลือกใช้เป็นหลอดไฟฟ้า 60 Watts ซึ่งจะส่งคำสั่งไปอ่านค่ากลับมาอย่างเดียว

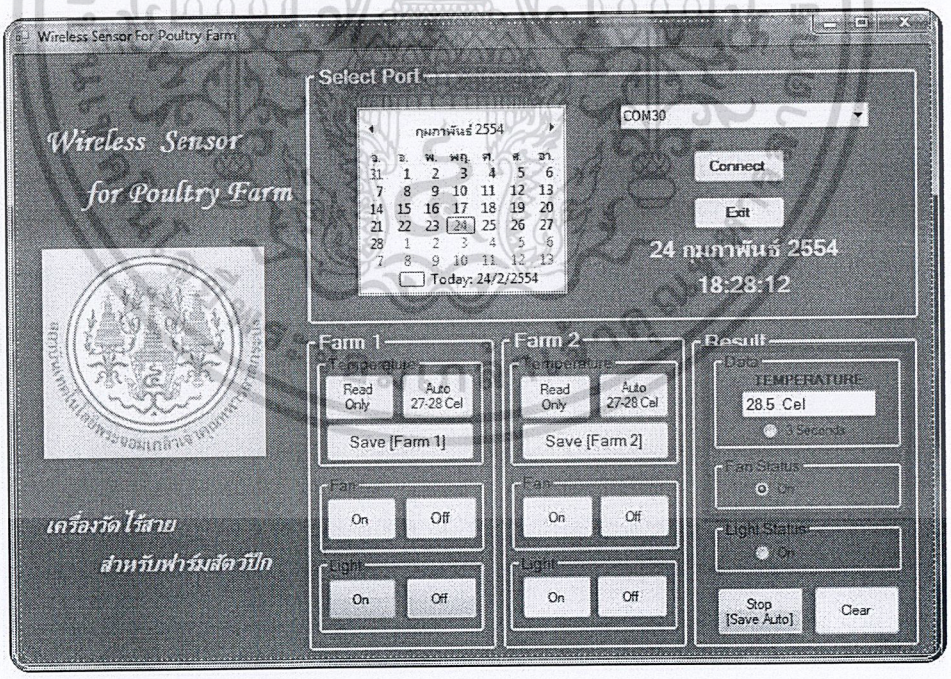


รูปที่ 4.27 หน้าจอโปรแกรม GUI แสดงลักษณะปุ่มในโปรแกรมเมื่อกดปุ่ม Auto

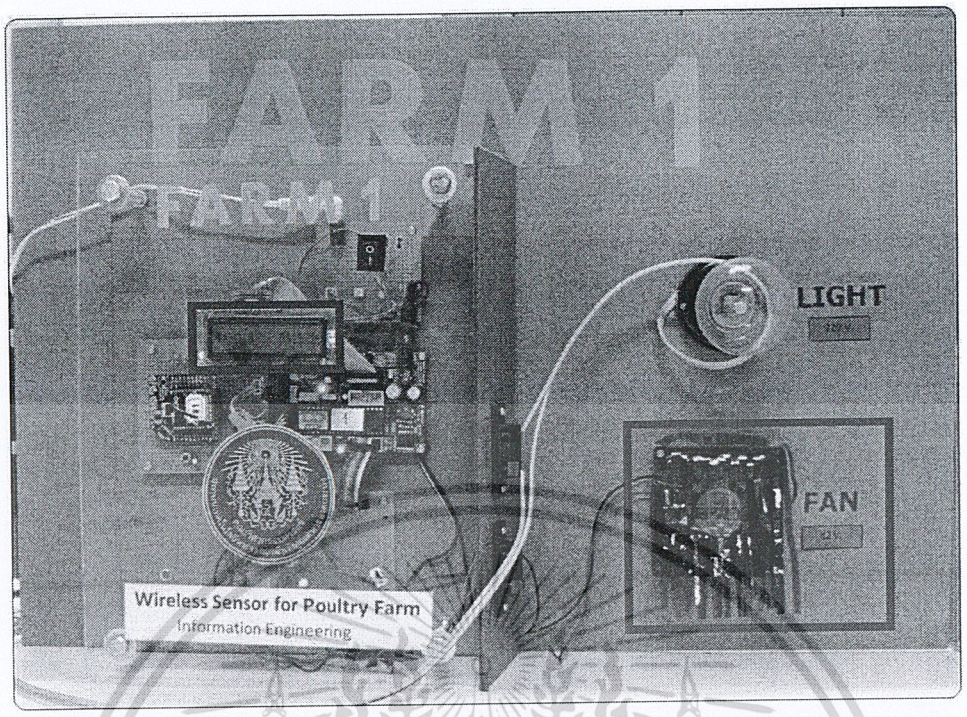


รูปที่ 4.28 แสดงการตรวจวัดข้อมูลอุณหภูมิและตรวจดูสถานะรีเลย์จากฟาร์มที่ 1 แบบอัตโนมัติ และทำการเพิ่มอุณหภูมิ

เมื่ออุณหภูมิเพิ่มขึ้นจนถึง 28.0 องศาเซลเซียส จะเห็นได้ว่าพัดลม 12V จะทำงาน

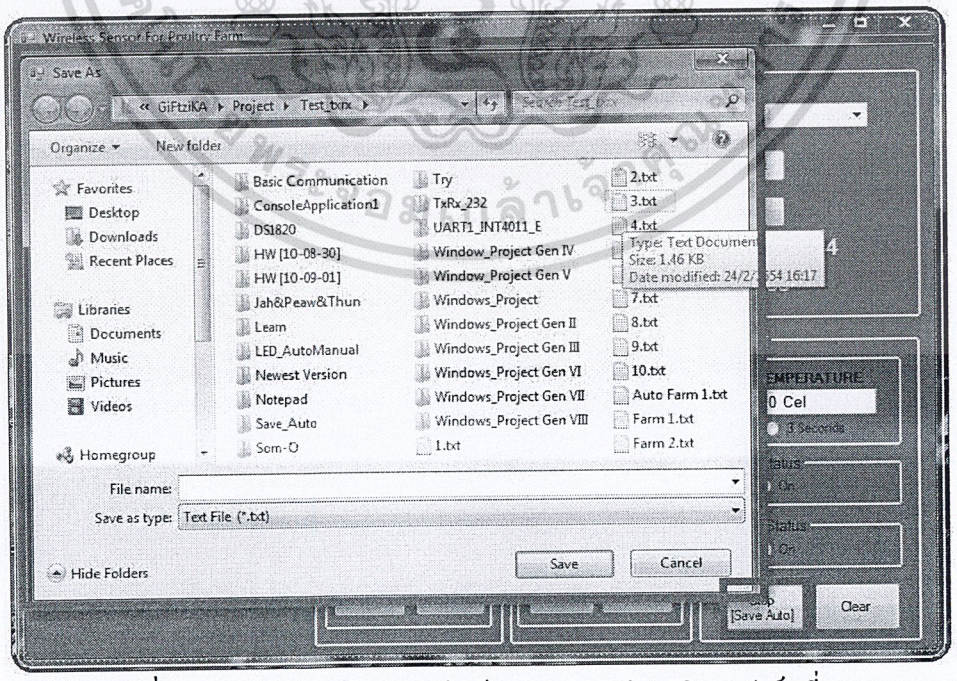


รูปที่ 4.29 หน้าจอโปรแกรม GUI แสดงสถานะรีเลย์ว่าเปิดใช้งานอยู่แบบอัตโนมัติ

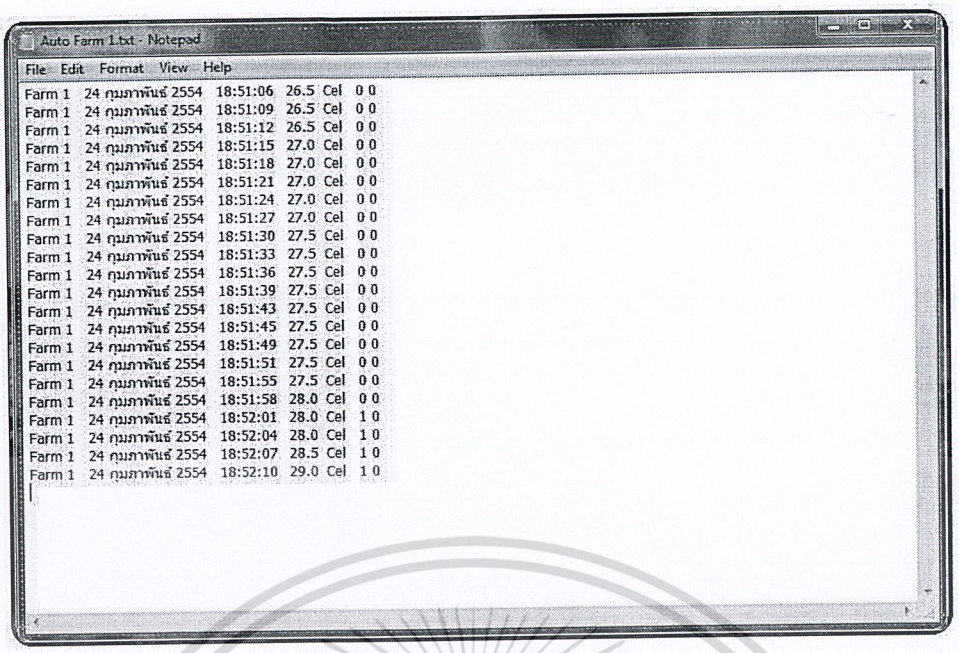


รูปที่ 4.30 แสดงการตรวจวัดข้อมูลอุณหภูมิและตรวจดูสถานะรีเลย์จากฟาร์มที่ 1 แบบอัตโนมัติ

3.8) เมื่อต้องการหยุดทำการตรวจวัดอุณหภูมิแบบอัตโนมัติและต้องการบันทึกข้อมูลอุณหภูมิและสถานะรีเลย์ที่ตรวจวัดได้แบบอัตโนมัติจากฟาร์มที่ 1 ให้ไปที่ช่องแสดงผล ให้เลือกกดปุ่ม Stop [Save Auto] โปรแกรมจะเรียก File ให้เราเลือก Folder ปลายทางที่เราต้องการบันทึก ข้อมูลจะถูกบันทึกอัตโนมัติเป็น Text file อยู่ใน Folder ที่เราเลือกเอาไว้



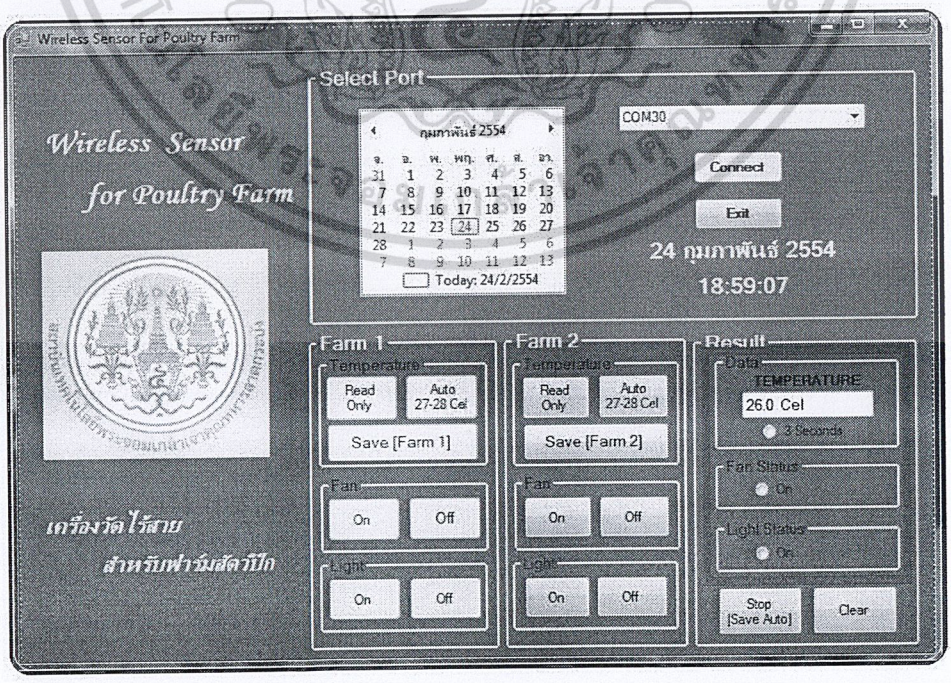
รูปที่ 4.31 แสดงการเลือกบันทึกข้อมูลแบบอัตโนมัติจากฟาร์มที่ 1



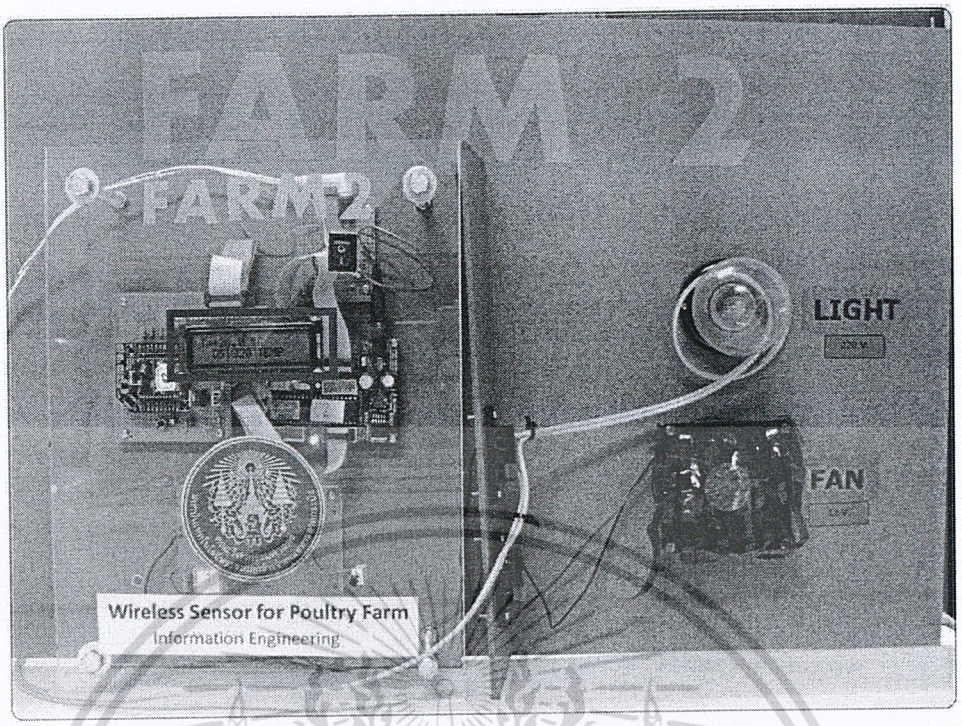
รูปที่ 4.32 แสดงข้อมูลอุณหภูมิและสถานะรีเลย์ที่บันทึกได้จากฟาร์มที่ 2 แบบอัตโนมัติ

- Farm 2

3.9) ทดลองส่งรับข้อมูลอุณหภูมิและตรวจดูสถานะรีเลย์ จากฟาร์มที่ 2 โดยไปที่ช่อง Farm 2 ให้เลือกกดปุ่ม Read Only โมดูล XBee-PRO ภาคส่ง จะส่งคำสั่งไปยังโมดูล XBee-PRO ภาครับให้รับค่าอุณหภูมิและสถานะรีเลย์กลับมา โดยที่ช่องแสดงผล TEMPERATURE จะแสดงค่าอุณหภูมิที่วัดได้ และที่ช่อง Fan Status และ Light Status ก็จะแสดงสถานะรีเลย์ในขณะนั้นกลับมาให้ผู้ใช้ทราบ

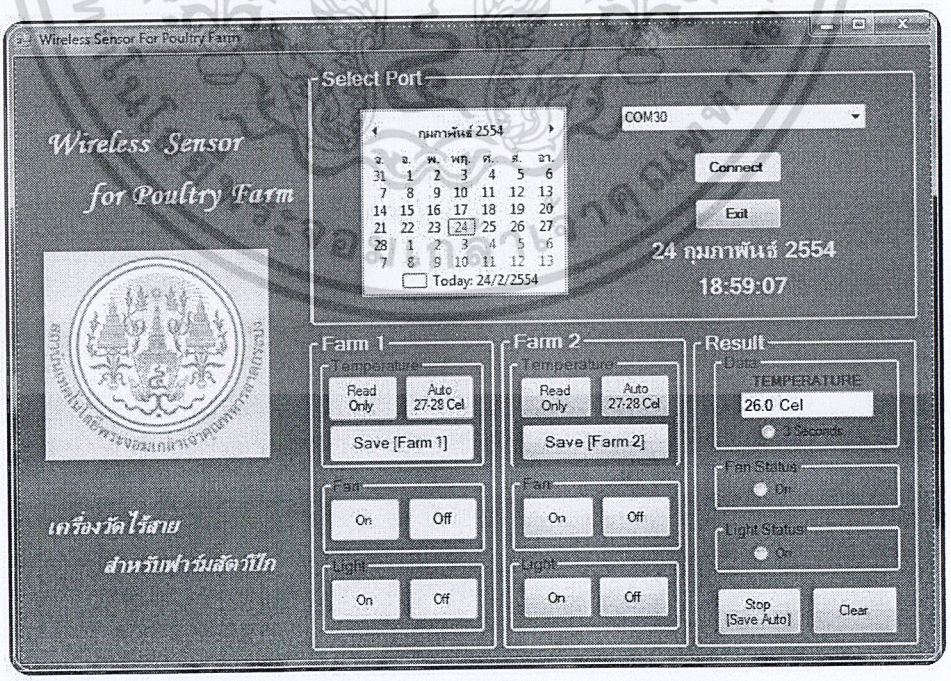


รูปที่ 4.33 แสดงอุณหภูมิและสถานะรีเลย์จากฟาร์มที่ 2 จากหน้าจอโปรแกรม GUI

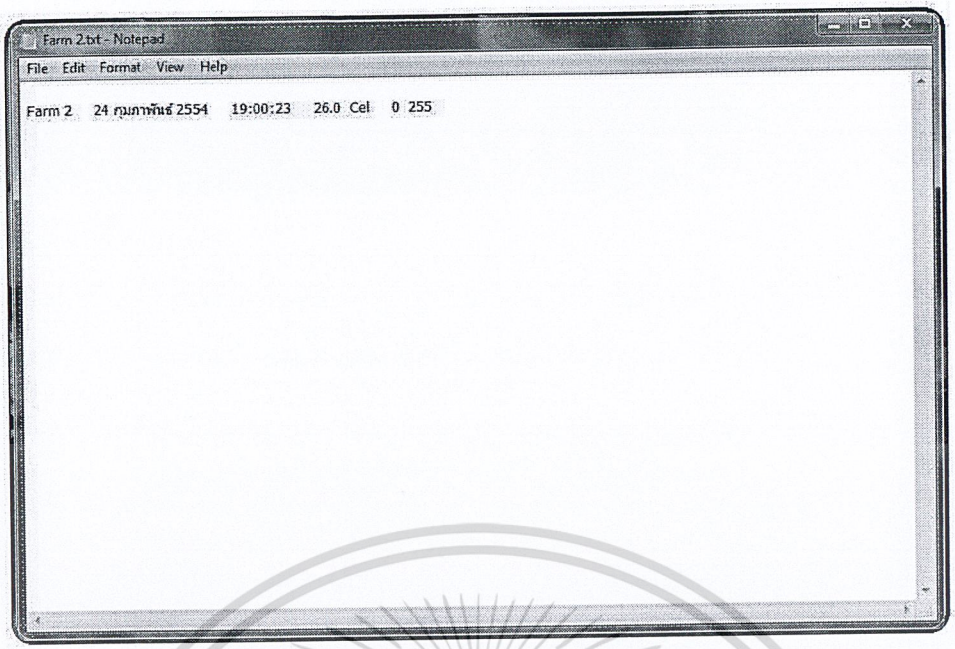


รูปที่ 4.34 แสดงอุณหภูมิและสถานะรีเลย์ของฟาร์มที่ 2

3.10) ทำการบันทึกข้อมูลอุณหภูมิและสถานะรีเลย์ที่ตรวจวัดได้ จากฟาร์มที่ 2 ณ เวลาและวันที่ที่กดบันทึก โดยไปที่ช่อง Farm 2 ให้เลือกกดปุ่ม Save [Farm 2] โดยข้อมูลที่บันทึกจะถูกเก็บ เป็น Text file อยู่ใน Folder ที่สร้างเอาไว้

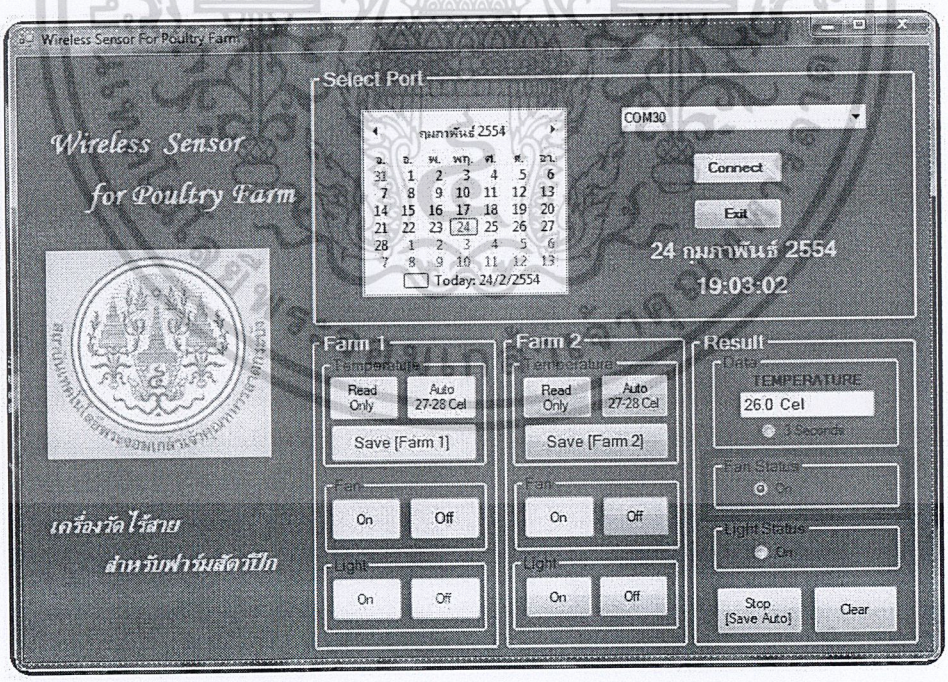


รูปที่ 4.35 แสดงการเลือกบันทึกข้อมูลจากฟาร์มที่ 2

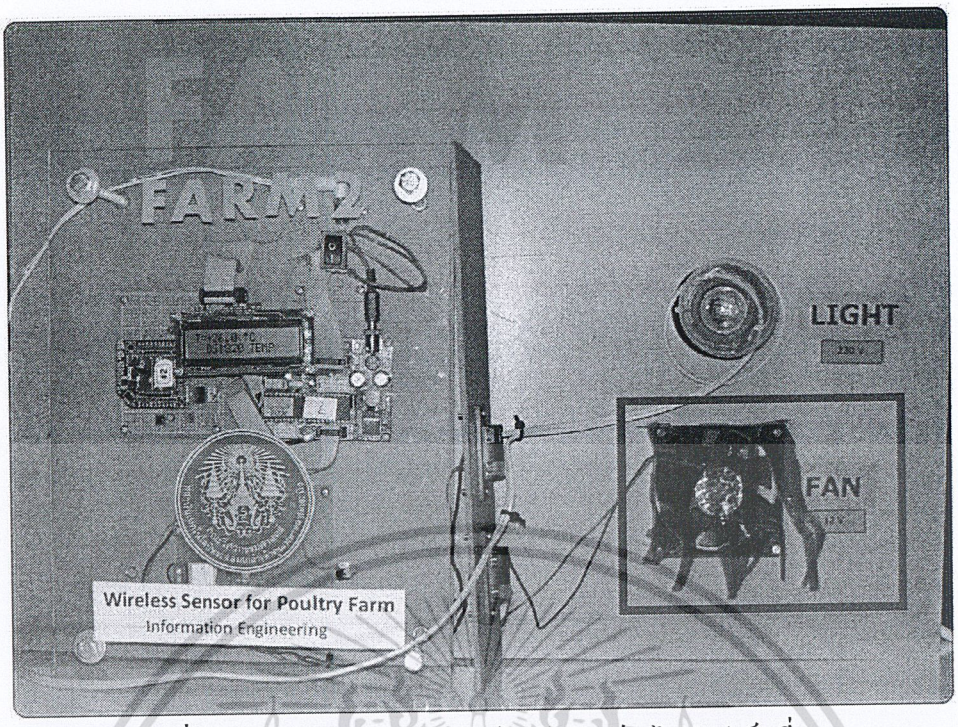


รูปที่ 4.36 แสดงข้อมูลอุณหภูมิและสถานะรีเลย์ที่บันทึกได้จากฟาร์มที่ 2

3.11) ทดลองเปิด-ปิดอุปกรณ์ไฟฟ้ารีเลย์ 2 ในการทดลองเลือกใช้อุปกรณ์พัดลม 12V โดยจากฟาร์มที่ 1 ไปที่ช่อง Fan ให้เลือกกดปุ่ม On สถานะรีเลย์ที่ช่อง Status Relay1 จะมีไฟติดแสดงสถานะว่ารีเลย์เปิดอยู่

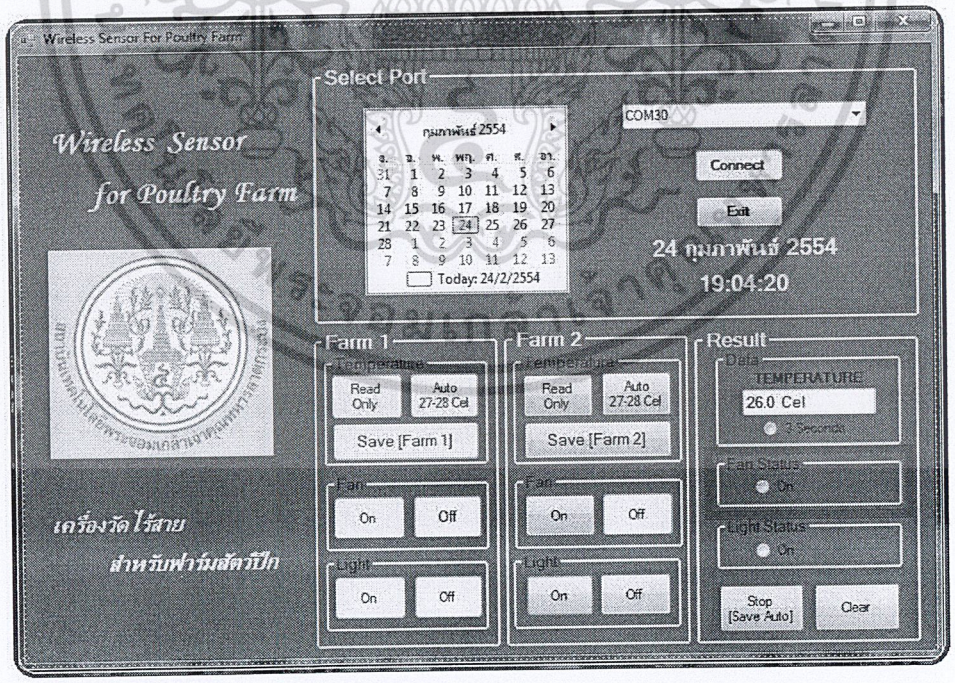


รูปที่ 4.37 หน้าจอโปรแกรม GUI แสดงสถานะพัดลมว่าเปิดอยู่

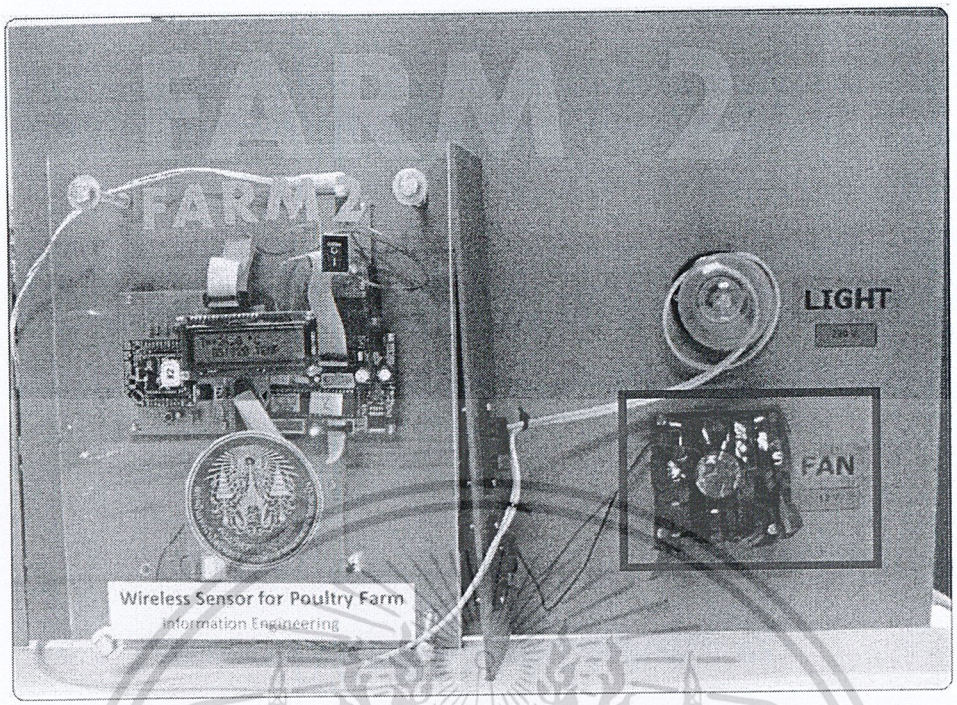


รูปที่ 4.38 แสดงการเปิดอุปกรณ์ไฟฟ้ารีเลย์ 1 ไปยังฟาร์มที่ 2

3.12) ทดลองปิดอุปกรณ์ไฟฟ้ารีเลย์ 1 โดยจากฟาร์มที่ 2 ไปที่ช่อง Fan ให้เลือกกดปุ่ม Off สถานะรีเลย์ที่ช่อง Status Relay1 จะไม่มีไฟติดแสดงสถานะว่ารีเลย์ปิดแล้ว

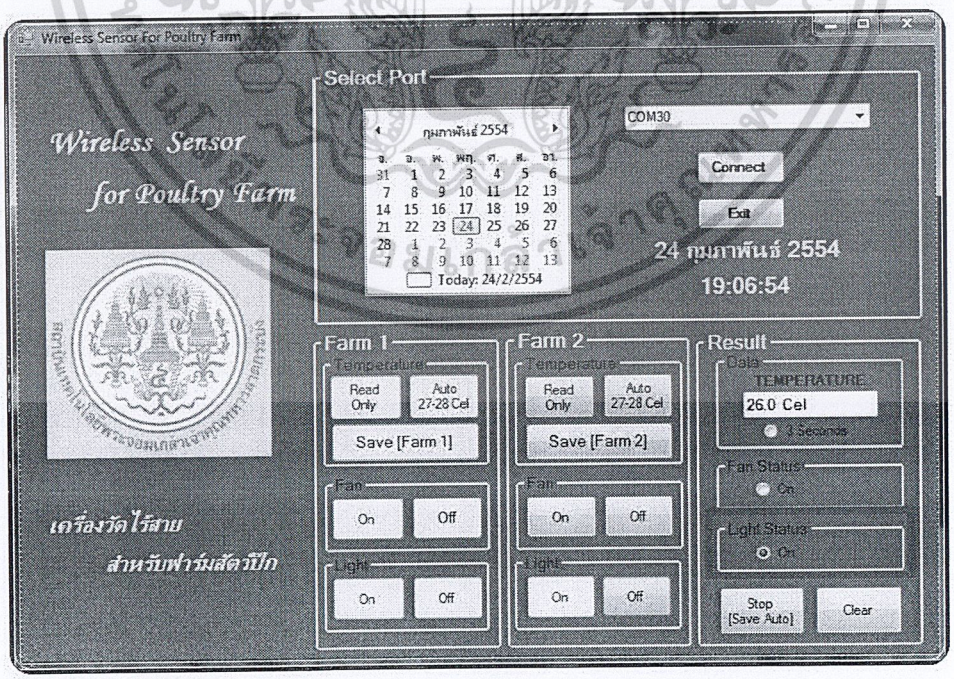


รูปที่ 4.39 หน้าจอโปรแกรม GUI แสดงสถานะพัดลมว่าปิดอยู่

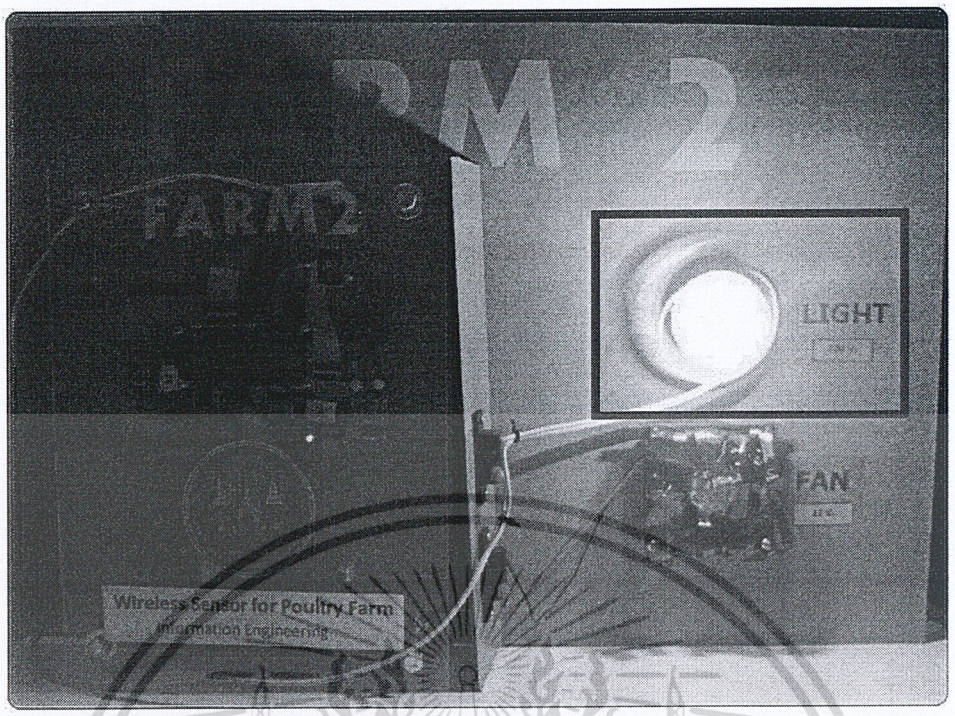


รูปที่ 4.40 แสดงการปิดอุปกรณ์ไฟฟ้ารีเลย์ 1 ไปยังฟาร์มที่ 2

3.13) ทดลองเปิด-ปิดอุปกรณ์ไฟฟ้ารีเลย์ 2 ในการทดลองเลือกใช้อุปกรณ์หลอดไฟฟ้า 60 Watts โดยจากฟาร์มที่ 2 ไปที่ช่อง Light ให้เลือกกดปุ่ม On สถานะรีเลย์ที่ช่อง Status Relay2 จะมีไฟติดแสดงสถานะว่ารีเลย์เปิดอยู่

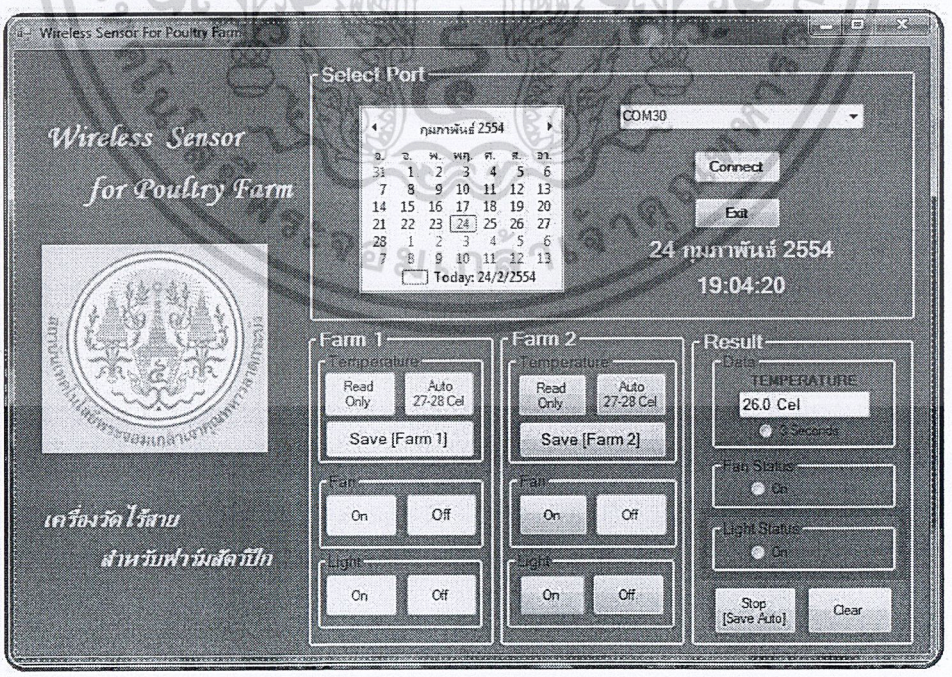


รูปที่ 4.41 หน้าจอโปรแกรม GUI แสดงสถานะไฟว่าเปิดอยู่

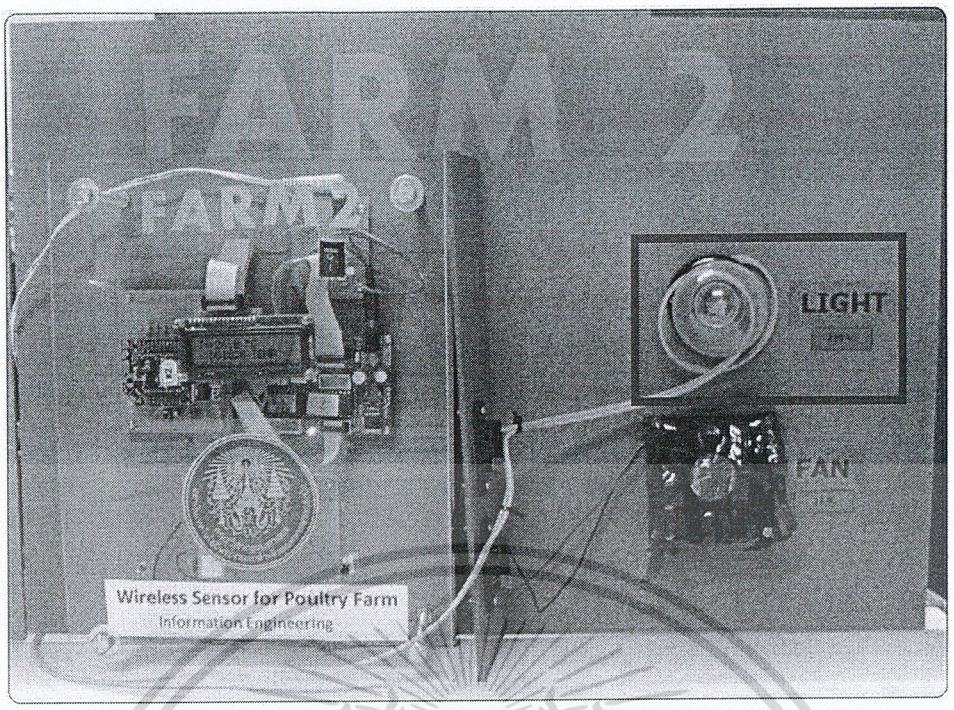


รูปที่ 4.42 แสดงการเปิดอุปกรณ์ไฟฟ้ารีเลย์ 2 ไปยังฟาร์มที่ 2

3.14) ทดลองปิดอุปกรณ์ไฟฟ้ารีเลย์ 2 ในการทดลองเลือกใช้อุปกรณ์หลอดไฟฟ้า 60 Watts โดยจากฟาร์มที่ 2 ไปที่ช่อง Light ให้เลือกกดปุ่ม Off สถานะรีเลย์ที่ช่อง Status Relay2 จะมีไม่มีไฟติดแสดงสถานะว่ารีเลย์ปิดแล้ว

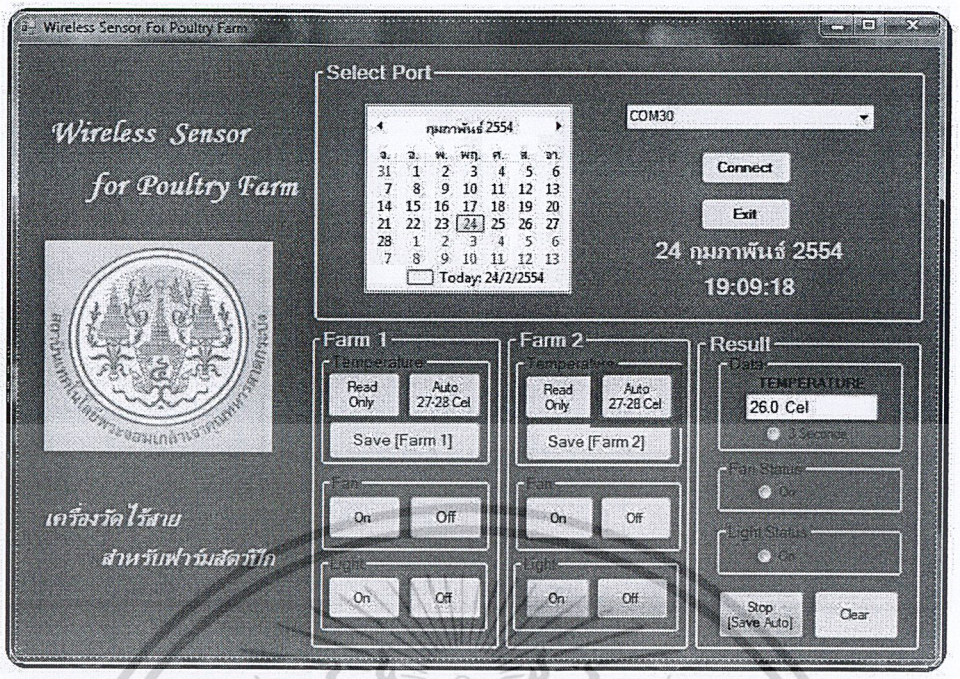


รูปที่ 4.43 หน้าจอโปรแกรม GUI แสดงสถานะไฟว่าปิดอยู่

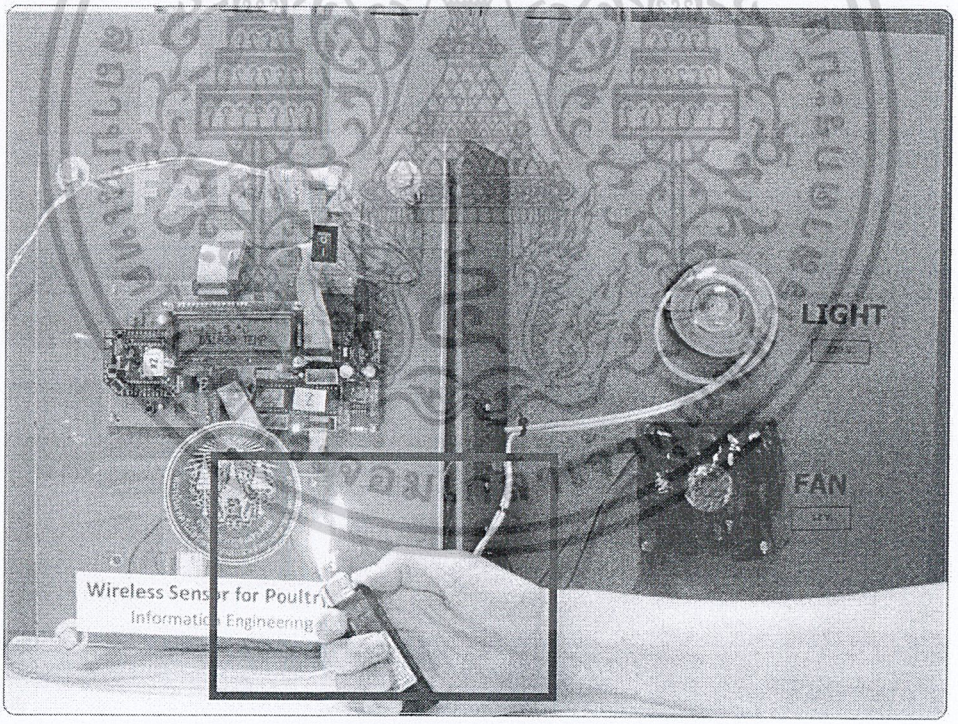


รูปที่ 4.44 แสดงการปิดอุปกรณ์ไฟฟ้ารีเลย์ 2 ไปยังฟาร์มที่ 2

3.15) ทดลองรับส่งข้อมูลอุณหภูมิและตรวจดูสถานะรีเลย์ จากฟาร์มที่ 2 โดยให้โปรแกรมทำการตรวจวัดอุณหภูมิแบบอัตโนมัติ ซึ่งในการทดลองตั้งโปรแกรมเวลาให้ตรวจวัดทุกๆ 3 วินาที โดยไปที่ช่อง Farm 2 ให้เลือกกดปุ่ม Auto 27.0-28.0 จากนั้นที่ช่องแสดงผล TEMPERATURE จะแสดงค่าอุณหภูมิที่วัดได้ทุกๆ 3 วินาที และที่ช่อง Fan Status , Light Status ก็จะแสดงสถานะรีเลย์ในขณะนั้นกลับมาแสดงทุกๆ 3 วินาที และมีการโปรแกรมให้อุปกรณ์ไฟฟ้ารีเลย์ทำการเปิด-ปิด อัตโนมัติ เมื่ออุณหภูมิสูงกว่า 27.5 องศาเซลเซียส ก็จะสั่งให้อุปกรณ์ไฟฟ้ารีเลย์ ซึ่งในการทดลองได้เลือกใช้อุปกรณ์พัดลม 12V ทำการเปิดจนกว่าอุณหภูมิจะต่ำกว่า 27.5 องศาเซลเซียส จึงจะทำการปิด ซึ่งเป็นอุปกรณ์ไฟฟ้ารีเลย์ 1 ส่วนอุปกรณ์ไฟฟ้ารีเลย์ 2 ได้เลือกใช้เป็นหลอดไฟฟ้า 60 Watts ซึ่งจะส่งคำสั่งไปอ่านค่ากลับมาอย่างเดียว

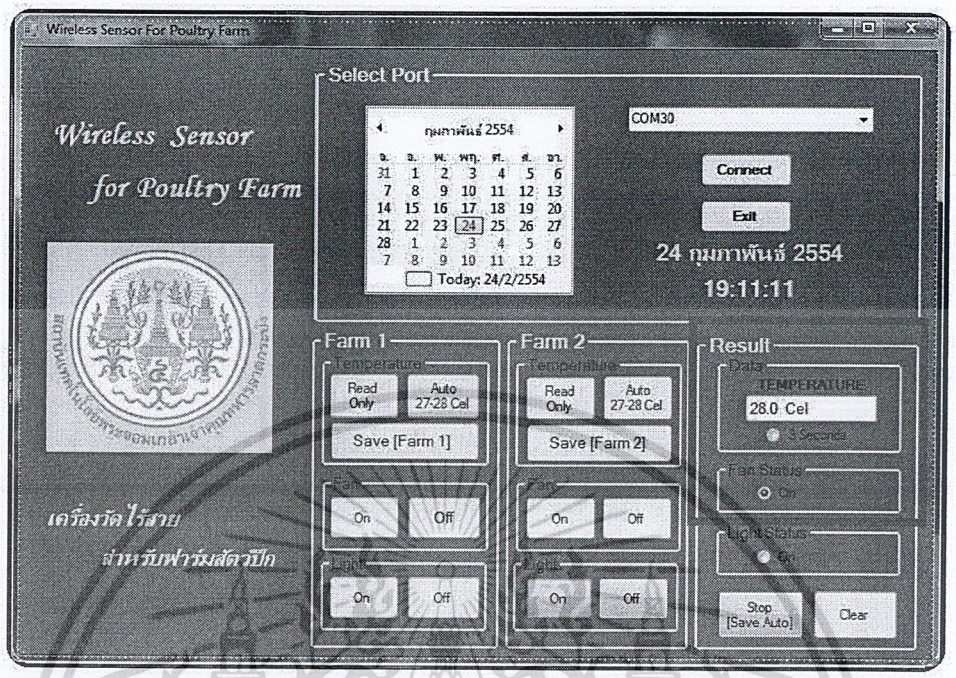


รูปที่ 4.45 หน้าจอโปรแกรม GUI แสดงลักษณะปุ่มในโปรแกรมเมื่อกดปุ่ม Auto

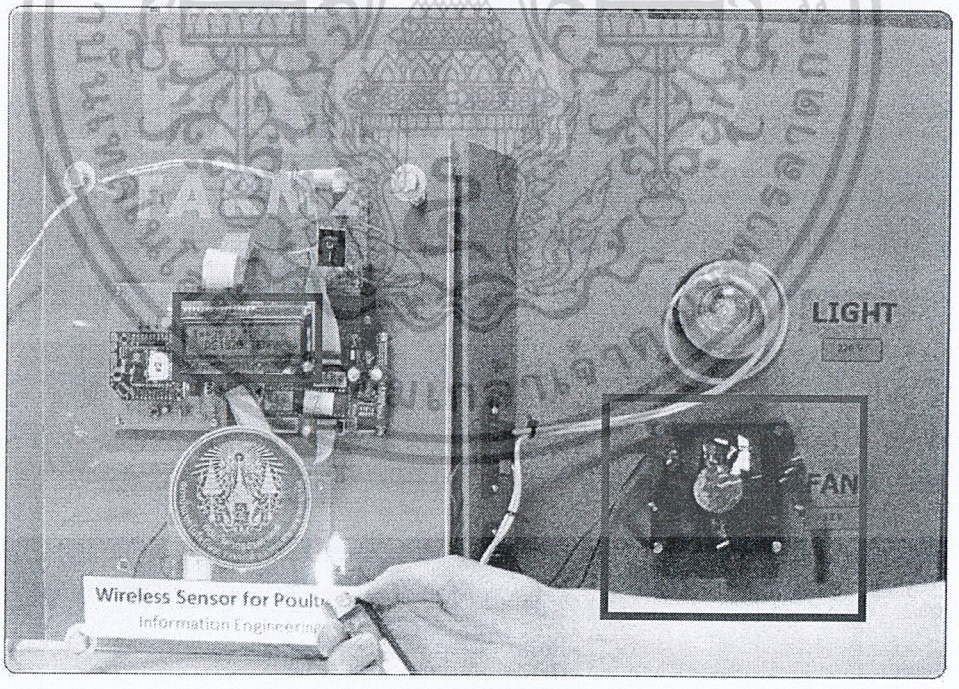


รูปที่ 4.46 แสดงการตรวจวัดข้อมูลอุณหภูมิและตรวจดูสถานะรีเลย์จากฟาร์มที่ 2 แบบอัตโนมัติ และทำการเพิ่มอุณหภูมิ

เมื่ออุณหภูมิเพิ่มขึ้นจนถึง 28.0 องศาเซลเซียส จะเห็นได้ว่าพัดลม 12V จะทำงาน

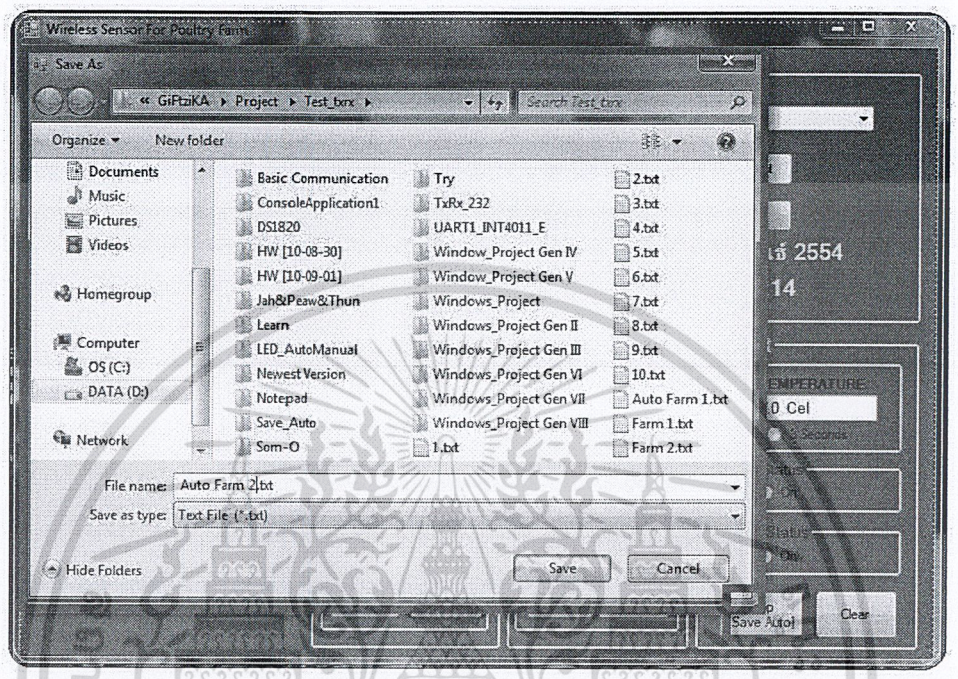


รูปที่ 4.47 หน้าจอโปรแกรม GUI แสดงสถานะรีเลย์ว่าเปิดใช้งานอยู่แบบอัตโนมัติ

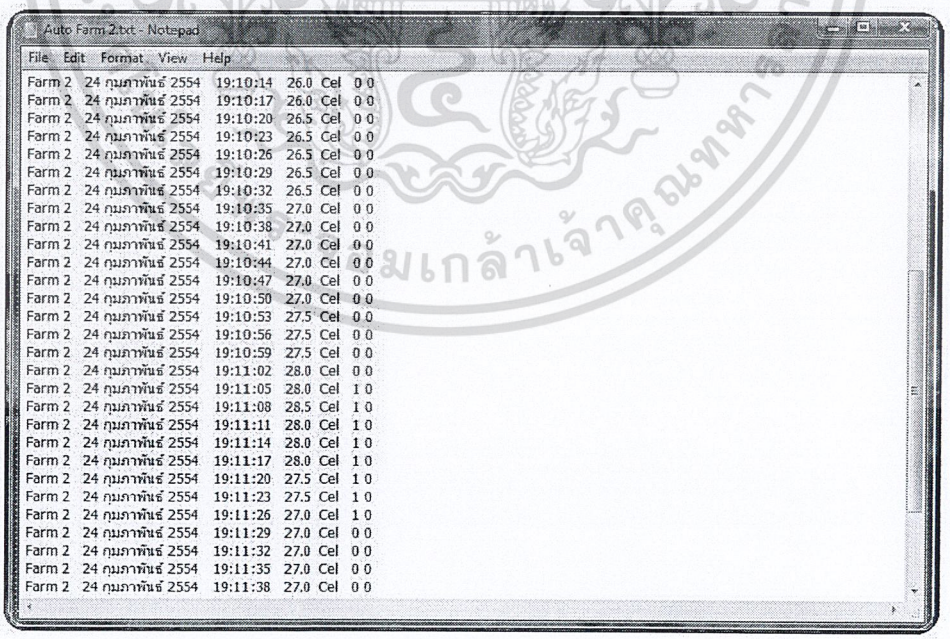


รูปที่ 4.48 แสดงการตรวจวัดข้อมูลอุณหภูมิและตรวจดูสถานะรีเลย์จากฟาร์มที่ 2 แบบอัตโนมัติ

3.16) เมื่อต้องการหยุดทำการตรวจวัดอุณหภูมิแบบอัตโนมัติและต้องการบันทึกข้อมูลอุณหภูมิและสถานะรีเลย์ที่ตรวดวัดได้แบบอัตโนมัติจากฟาร์มที่ 2 ให้ไปที่ช่องแสดงผล ให้เลือกกดปุ่ม Stop [Save Auto] โปรแกรมจะเรียก File ให้เราเลือก Folder ปลายทางที่เราต้องการบันทึก ข้อมูลจะถูกบันทึกอัตโนมัติเป็น Text file อยู่ใน Folder ที่เราเลือกเอาไว้



รูปที่ 4.49 แสดงการเลือกบันทึกข้อมูลแบบอัตโนมัติจากฟาร์มที่ 1



รูปที่ 4.50 แสดงข้อมูลอุณหภูมิและสถานะรีเลย์ที่บันทึกได้จากฟาร์มที่ 2 แบบอัตโนมัติ

4.4 การทดลองวัดระยะการส่ง

เมื่อสามารถรับ – ส่งข้อมูลไร้สายได้แล้ว ในการใช้งานจริงจะมีปัจจัยสำคัญที่ทำให้การส่งข้อมูลเกิดความผิดพลาด นั่นคือระยะในการส่งที่มีสิ่งกีดขวาง ซึ่งตามความเป็นจริงการตรวจวัดภายในฟาร์ม หรือสถานที่ใดๆ ตัวรับและตัวส่งไม่ได้ส่งข้อมูลเป็นแบบเส้นตรง แต่อาจจะอยู่ในที่ทึบและมีสิ่งกีดขวาง เช่น วัตถุที่สัญญาณไม่สามารถทะลุผ่านได้

เราจึงการทดลองวัดระยะที่สามารถส่งข้อมูลได้ในที่ร่ม โดยการวัดค่าในระยะที่ต่างๆ กันตามคุณสมบัติของ XBee-PRO ที่สามารถส่งได้สูงสุดประมาณ 100 เมตร ระยะทำการในร่ม โดยมีสิ่งกีดขวางเป็นป้าย เสา ต้นไม้ ได้ผลการทดลอง ดังตารางที่ 4.1

ตารางที่ 4.1 ผลทดลองวัดระยะทางการส่งข้อมูลในที่ร่ม

ระยะทาง (เมตร)	การรับส่งข้อมูล
10	ปกติ
20	ปกติ
30	ปกติ
40	ปกติ
50	ปกติ
60	ปกติ
70	ปกติ
80	ปกติ
90	ปกติ
100	ปกติ
110	ปกติ
120	เริ่มเกิดการดีเลย์
130	ขาดการส่งข้อมูลเป็นระยะ
140	ข้อมูลส่งมาได้น้อยครั้ง ขาดการติดต่อเป็นเวลานาน
150	ไม่สามารถติดต่อได้

บทที่ 5
บทสรุปและวิจารณ์ผลการทดลอง

5.1 สรุปผลการทดลอง

โครงงานเครื่องวัดไร้สายสำหรับฟาร์มสัตว์ปีกที่ได้ทำการพัฒนาขึ้นนี้ มีผลการทดลองเป็นไปตามวัตถุประสงค์ของโครงงาน โดยมีรายละเอียดในการพัฒนาแยกออกเป็นส่วนต่างๆ ดังนี้

1) ส่วนควบคุมและประมวลผลกลาง ทำหน้าที่ในการควบคุม สั่งการ และตอบสนองต่อเหตุการณ์ต่างๆให้ได้ตามที่ต้องการ โดยเขียนโปรแกรมเพื่อเชื่อมต่อการทำงานร่วมกันระหว่างส่วนควบคุม ส่วนประมวลผล และส่วนแสดงผล ให้สามารถรับส่งข้อมูลผ่าน ZigBee ซึ่งเป็นระบบเครือข่ายไร้สายได้ ซึ่งผลการทดลองที่ได้เป็นไปตามวัตถุประสงค์ของโครงงานคือ ส่วนประมวลผลกลางสามารถควบคุมโปรแกรมให้รับส่งข้อมูลที่ตรวจวัดได้จากเซนเซอร์เครื่องวัดที่ติดตั้งไว้ภายในฟาร์ม ให้มาแสดงผลบนหน้าจอ GUI ที่ได้ออกแบบไว้บนคอมพิวเตอร์ได้ถูกต้อง

2) ส่วนวงจร สร้างวงจรชุดเครื่องวัดไร้สายซึ่งประกอบด้วย เซนเซอร์สำหรับตรวจวัดอุณหภูมิและอุปกรณ์ไฟฟ้ารีเลย์ได้แก่ พัดลม และหลอดไฟฟ้า ซึ่งควบคุมโดยไมโครคอนโทรลเลอร์ dsPIC30F4011 โดยสามารถโปรแกรมให้ไมโครคอนโทรลเลอร์ติดต่อกับเซนเซอร์และรีเลย์ แล้วแสดงผลผ่านหน้าจอLCD และหน้าจอ GUI บนเครื่อง PC โดยใช้โมดูล XBee-PRO เป็นตัวเชื่อมต่อกับระบบเครือข่าย โดยโมดูล XBee-PRO ภาคส่ง จะส่งคำสั่งไปยังโมดูล XBee-PRO ภาครับให้รับค่าอุณหภูมิและสถานะรีเลย์กลับมา ซึ่งมีผลการทดลองเป็นไปตามที่คาดหวังไว้คือ วงจรเซนเซอร์สามารถตรวจวัดค่าอุณหภูมิ และแสดงรีเลย์ ผ่านหน้าจอLCD แล้วส่งค่ากลับมาที่หน้าจอ GUI บนเครื่อง PC ซึ่งผลข้อมูลที่ได้มีความถูกต้องตรงกันทั้งสองส่วน และสามารถทำงานตามคำสั่งได้ตามต้องการโดยใช้รูปแบบการแสดงผลที่สามารถเข้าใจได้ง่าย

3) ส่วนแสดงผลให้สามารถสั่งการเปิด-ปิด อุปกรณ์รีเลย์จากส่วนควบคุมของหน้าจอ GUI ซึ่งตัววงจรสามารถทำงานได้อย่างดีและมีรูปแบบการแสดงผลที่สามารถเข้าใจได้ง่ายซึ่งจะทำหน้าที่ติดต่อกับผู้ใช้ในการสั่งงานควบคุมต่างๆคือ การเปิด-ปิดพัดลม การเปิด-ปิดหลอดไฟฟ้า ทั้งแบบ Manual และ Auto และสามารถแสดงผลการเก็บบันทึกข้อมูลอุณหภูมิที่วัดได้ และแสดงสถานะของรีเลย์ได้ทุกช่วงเวลาตามต้องการทำงานได้อย่างมีประสิทธิภาพ และรวดเร็ว

4) ส่วนการบันทึกผล โดยสามารถเก็บบันทึกข้อมูลอุณหภูมิที่วัดได้ และเก็บบันทึกสถานะของรีเลย์ได้ทุกช่วงเวลาตามต้องการ ซึ่งทำงานตอบสนองได้อย่างมีประสิทธิภาพและรวดเร็ว

5.2 ปัญหาที่พบ

1) เนื่องจากการทำโครงงานนี้ต้องอาศัยความรู้ในหลายๆ ด้าน ทั้งฮาร์ดแวร์และซอร์ฟแวร์ รวมถึงเทคโนโลยีอื่นๆ ที่ใช้ในโครงงานนี้ ผู้ทำโครงงานมีความรู้ไม่เพียงพอ จึงเสียเวลามากในการค้นคว้าหาความรู้

2) อุปกรณ์ที่ใช้บางอย่าง เช่น XBee-PRO บอร์ดไมโครคอนโทรลเลอร์ dsPIC30F4011 และเซนเซอร์วัดอุณหภูมิ DS1820 มีราคาแพง

3) ปัญหาความไม่คงทนและชำรุดได้ง่ายของอุปกรณ์อิเล็กโทรนิกส์ เมื่อทำการทดลองมากๆ ทำให้เสียเวลาซ่อมแซมและจัดหาอุปกรณ์ใหม่

4) ปัญหาการเขียนโปรแกรมควบคุมเซนเซอร์ และควบคุมอุปกรณ์ไฟฟ้ารีเลย์ผ่านไมโครคอนโทรเลอร์ลงบอร์ดภาคส่งแล้วไม่สามารถทำงานได้

5) ปัญหาในการออกแบบและเขียนโค้ดโปรแกรมหน้าจอ GUI ด้วยโปรแกรม C#2010 เมื่อทำการทดลอง ค่าที่รับมาจากบอร์ดภาคส่งไม่แสดงผลออกทางหน้าจอตามที่ต้องการ

6) ทำการออกแบบให้วงจรเซนเซอร์มีขนาดเล็กและมีอุปกรณ์มาก จึงทำให้วงจรเซนเซอร์นั้นมีความซับซ้อน ยากในการเชื่อมต่อสายไฟระหว่างอุปกรณ์ เป็นผลให้สายไฟที่ใช้เชื่อมต่อไม่แน่นทำให้ได้ผลการการทดลองคลาดเคลื่อน

7) อุปกรณ์เกิดความเสียหายจากการได้รับความร้อนมากเกินไประหว่างการต่อวงจร หรือจาก การต่อไฟเลี้ยงเพิ่มให้กับอุปกรณ์ ทำให้การทำงานมีความผิดพลาดและไม่เสถียร ทำให้ต้องแก้ไข เปลี่ยนแปลงตัวอุปกรณ์หลายครั้ง

5.3 แนวทางการแก้ไขปัญหาและพัฒนาโครงงาน

1) สามารถนำชุดเครื่องวัดไร้สายไปประยุกต์ใช้กับเซนเซอร์เครื่องวัดและเซนเซอร์ต่างๆ หรืออุปกรณ์ไฟฟ้ารีเลย์ชนิดอื่นๆได้

2) สามารถพัฒนาโดยสร้างชุดเครื่องวัดไร้สายให้สามารถรับส่งข้อมูลได้ในระยะทางไกลโดยผ่านโมดูล XBee-PRO ซึ่งเป็นระบบเครือข่ายไร้สายให้กว้างไกลยิ่งขึ้นได้

3) พัฒนาส่วนหน้าจอ GUI ให้สะดวกต่อการใช้งานให้สามารถทำงานตามคำสั่งได้ตามต้องการ โดยใช้รูปแบบการแสดงผลที่สามารถเข้าใจได้ง่ายต่อผู้ใช้มากยิ่งขึ้น

4) อุปกรณ์โมดูล XBee-PRO และอุปกรณ์เซนเซอร์มีข้อจำกัดในการใช้งานอยู่บ้าง ในการพัฒนาต่อจึงควรพัฒนาหรือเลือกชนิดของอุปกรณ์ให้มีความเหมาะสมกับการใช้งานภายในงบประมาณที่มีอยู่

5) อุปกรณ์ที่มีการใช้งาน ควรรองรับกับสถานการณ์ต่างๆ ที่อาจเกิดขึ้น จึงควรพัฒนาให้มีความคงทน แข็งแรงและสามารถใช้งานได้ในสภาพอากาศต่างๆ ด้วย

บรรณานุกรม

[1] ZigBee IEEE 802.15.4 Standard datasheet at
http://www.thaitelecomkm.org/TTE/topic/attach/Bluetooth_and_Zigbee/index.php

[2] ZigBee Technology and IEEE 802.15.4 Standard datasheet at
http://zigbeeyoyo.blogspot.com/2007/08/zigbee.html

[3] DS1820 Digital Thermometer Replacements datasheet at
www.datasheetcatalog.org/datasheet/maxim/DS1820-DS1820S.pdf

[4] DS1820 Embedded at
www.thaiembedded.com/blog/?tag=ds1820

[5] Zigbee and Xbee BASIC
http://www.thaieasyelec.com/index.php?lay=show&ac=article&Id=538707975&Ntype=1

[6] Xbee Basic Configuration in Network Application
http://www.thaieasyelec.com/Embedded-Electronics-Application/Xbee-Basic-Configuration-in-Network-Application.html

[7] สัจจะ จรัสรุ่งรวีวร. 2550. **คู่มือ Visual C# 2005 ฉบับสมบูรณ์.** ครั้งที่ 1. นนทบุรี: ไอดีซี อินโฟ ดิสทริบิวเตอร์ เซ็นเตอร์

[8] นคร ภักดีชาติ และคณะ. **คู่มือการทดลอง dsPIC Microcontroller เบื้องต้นด้วยโปรแกรมภาษาซี** . กรุงเทพ : บริษัท อินโนเวตีฟ เอ็กพอริเมนต์ จำกัด

ภาคผนวก ก.

SOURCE CODE MAIN AND CLIENT

SOURCE CODE MAIN

Code Visual C#2010

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.IO.Ports;                                  //ใส่เพื่อใช้งาน RS-232 Port ได้
using System.IO;
using System.Windows.Forms;

namespace Test_txrx
{
   public partial class Form1 : Form
   {
      int counter = new int();
      int end_counter = new int();
      byte[] bbyte = new byte[10];                      //สำหรับไว้รับข้อมูล
      byte[] tbyte = new byte[10];                      // สำหรับไว้ส่งข้อมูล
      string data_mess;

      public Form1()
      {
         InitializeComponent();
      }

      private void Form1_Load(object sender, EventArgs e)
      {
       serialPort1 .DataReceived +=new SerialDataReceivedEventHandler(serialPort1_DataReceived);
        //กำหนดช่องเลือกการติดต่อ
       string[] comStr = SerialPort.GetPortNames();
        int i = 0;
        foreach (string port in comStr)
        {
          comboBox1.Items.Add(comStr[i]);
          i++;
        }
        //comboBox1.SelectedIndex = 2;
        textBox1.Text = "";
        textBox2.Text = "";
        textBox3.Text = "";
```

```
textBox4.Text = "";
textBox5.Text = "";
textBox6.Text = "";
textBox7.Text = "";
textBox8.Text = "";
textBox9.Text = "";
textBox10.Text = "";
textBox11.Text = "";
textBox12.Text = "";
textBox13.Text = "";
textBox14.Text = "";
textBox15.Text = "";
textBox16.Text = "";
textBox17.Text = "";
textBox18.Text = "";
textBox19.Text = "";
textBox20.Text = "";
textBox21.Text = "";
textBox22.Text = "";
textBox23.Text = "";
textBox24.Text = "";
textBox25.Text = "";
textBox26.Text = "";

button2.Enabled = true;
button3.Enabled = false;
button4.Enabled = false;
button5.Enabled = false;
button6.Enabled = false;
button7.Enabled = false;
button8.Enabled = false;
button9.Enabled = false;
button10.Enabled = false;
button11.Enabled = false;
button12.Enabled = false;
button13.Enabled = false;
button14.Enabled = false;
button15.Enabled = false;
button16.Enabled = false;
button17.Enabled = false;
button18.Enabled = false;

textBox1.Enabled = false;
```

```
textBox2.Enabled = false;
textBox3.Enabled = false;
textBox4.Enabled = false;
textBox5.Enabled = false;
textBox6.Enabled = false;
textBox7.Enabled = false;
textBox8.Enabled = false;
textBox9.Enabled = false;
textBox10.Enabled = false;
textBox11.Enabled = false;
textBox12.Enabled = false;
textBox13.Enabled = false;
textBox14.Enabled = false;
textBox15.Enabled = false;
textBox16.Enabled = false;
textBox17.Enabled = false;
textBox18.Enabled = false;
textBox19.Enabled = false;
textBox20.Enabled = false;
textBox21.Enabled = false;
textBox22.Enabled = false;
textBox23.Enabled = false;
textBox24.Enabled = false;
textBox25.Enabled = false;
textBox26.Enabled = false;

radioButton1.Enabled = false;
radioButton2.Enabled = false;
radioButton3.Enabled = false;

  counter = 0;
end_counter = 6;
radioButton1.Checked = false;
radioButton2.Checked = false;
timer1.Enabled = false;
timer2.Enabled = false;
timer3.Enabled = true;
timer4.Enabled = false;
timer5.Enabled = false;
}
```

```
private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
   while (counter < end_counter)
   {
      bbyte[counter] = (byte)serialPort1.ReadByte();
      counter = counter + 1;
   }
}

private void button1_Click(object sender, EventArgs e)                //ปุ่มคอนเน็ก
{
   if (comboBox1.Items.Count > 0)
   {
      if (!serialPort1 .IsOpen )
      {
         serialPort1 .PortName = comboBox1.Text;
         serialPort1 .Open ();

         tbyte[0] = 0x82;
         textBox1.Text = tbyte[0].ToString();
         tbyte[1] = 0x30;
         textBox2.Text = tbyte[1].ToString();
         tbyte[2] = 0xEE;
         textBox3.Text = tbyte[2].ToString();
         tbyte[3] = 0x00;
         textBox4.Text = tbyte[3].ToString();
         tbyte[4] = 0x00;
         textBox5.Text = tbyte[4].ToString();
         tbyte[5] = 0xFF;
         textBox6.Text = tbyte[5].ToString();

         tbyte[0] = 0x81;
         textBox14.Text = tbyte[0].ToString();
         tbyte[1] = 0x30;
         textBox15.Text = tbyte[1].ToString();
         tbyte[2] = 0x32;
         textBox16.Text = tbyte[2].ToString();
         tbyte[3] = 0xEE;
         textBox17.Text = tbyte[3].ToString();
         tbyte[4] = 0x00;
         textBox18.Text = tbyte[4].ToString();
         tbyte[5] = 0x00;
```

```
textBox19.Text = tbyte[5].ToString();
tbyte[6] = 0xFF;
textBox20.Text = tbyte[6].ToString();

button1.Enabled = true;
button2.Enabled = true;
button3.Enabled = true;
button4.Enabled = true;
button5.Enabled = true;
button6.Enabled = false;
button7.Enabled = true;
button8.Enabled = true;
button9.Enabled = true;
button10.Enabled = true;
button11.Enabled = true;
button12.Enabled = true;
button13.Enabled = true;
button14.Enabled = true;
button15.Enabled = true;
button16.Enabled = true;
button17.Enabled = true;
button18.Enabled = true;

textBox1.Enabled = true;
textBox2.Enabled = true;
textBox3.Enabled = true;
textBox4.Enabled = true;
textBox5.Enabled = true;
textBox6.Enabled = true;
textBox7.Enabled = true;
textBox8.Enabled = true;
textBox9.Enabled = true;
textBox10.Enabled = true;
textBox11.Enabled = true;
textBox12.Enabled = true;
textBox13.Enabled = true;
textBox14.Enabled = true;
textBox15.Enabled = true;
textBox16.Enabled = true;
textBox17.Enabled = true;
textBox18.Enabled = true;
textBox19.Enabled = true;
textBox20.Enabled = true;
```

```
            textBox21.Enabled = true;
            textBox22.Enabled = true;
            textBox23.Enabled = true;
            textBox24.Enabled = true;
            textBox25.Enabled = true;
            textBox26.Enabled = true;

            radioButton1.Enabled = true;
            radioButton2.Enabled = true;
            radioButton3.Enabled = true;

        }

    }
}

private void button3_Click(object sender, EventArgs e)          //ปุ่มกดส่ง
{
    tbyte[0] = 0x82;
    textBox1.Text = tbyte[0].ToString();
    tbyte[1] = 0x30;
    textBox2.Text = tbyte[1].ToString();
    tbyte[2] = 0xEE;
    textBox3.Text = tbyte[2].ToString();
    tbyte[3] = 0x02;
    textBox4.Text = tbyte[3].ToString();
    tbyte[4] = 0x02;
    textBox5.Text = tbyte[4].ToString();
    tbyte[5] = 0xFF;
    textBox6.Text = tbyte[5].ToString();

    tbyte[0] = byte.Parse(textBox1.Text);
    tbyte[1] = byte.Parse(textBox2.Text);
    tbyte[2] = byte.Parse(textBox3.Text);
    tbyte[3] = byte.Parse(textBox4.Text);
    tbyte[4] = byte.Parse(textBox5.Text);
    tbyte[5] = byte.Parse(textBox6.Text);
    timer1.Enabled = true;
    serialPort1.Write(tbyte, 0, 6);
    counter = 0;

    button1.Enabled = false;
    button2.Enabled = true;
```

```
    button3.Enabled = true;
    button4.Enabled = true;
    button5.Enabled = true;
    button6.Enabled = false;
    button7.Enabled = false;
    button8.Enabled = false;
    button9.Enabled = true;
    button10.Enabled = true;
    button11.Enabled = true;
    button12.Enabled = true;
    button13.Enabled = false;
    button14.Enabled = false;
    button15.Enabled = false;
    button16.Enabled = false;
    button17.Enabled = true;
    button18.Enabled = false;
}

private void button2_Click(object sender, EventArgs e)          //ปุ่มออก
{
    timer1.Enabled = false;
    serialPort1.Close();
    this.Close();
}

private void timer1_Tick(object sender, EventArgs e)            //แสดงผล
{
    textBox7.Text = bbyte[0].ToString();
    textBox8.Text = bbyte[1].ToString();
    textBox9.Text = bbyte[2].ToString();
    textBox10.Text = bbyte[3].ToString();
    textBox11.Text = bbyte[4].ToString();
    textBox12.Text = bbyte[5].ToString();

    textBox13.Text = bbyte[1].ToString() + "." + bbyte[2].ToString() + "  Cel";
    if (bbyte[3] == 0) radioButton1.Checked = false;
    else radioButton1.Checked = true;
    if (bbyte[4] == 0) radioButton2.Checked = false;
    else radioButton2.Checked = true;

    if (counter == 6) counter = 0;
    button6.Enabled = true;
    //data_mess += textBox13.Text + "    " + bbyte[3].ToString() + " " + bbyte[4].ToString() + "  ||  ";
```

```
    data_mess += groupBox7.Text + "   " + date.Text + "   " + time.Text + "   " + textBox13.Text + "   " + bbyte[3].ToString() + " " + bbyte[4].ToString() +
"\r\n";
        timer1 .Enabled =false ;

    }

    private void button4_Click(object sender, EventArgs e)              //ปุ่มเคลียร์
    {
      //ล้างข้อมูลที่รับมาทั้งหมด
      for (int i = 0; i < 10; i++)
      {bbyte[i] = 0x00; }

      //ล้างส่วนแสดงผลหน้าจอ
      textBox4.Text = "";
      textBox5.Text = "";
      textBox7.Text = "";
      textBox8.Text = "";
      textBox9.Text = "";
      textBox10.Text = "";
      textBox11.Text = "";
      textBox12.Text = "";
      textBox13.Text = "";
      //textBox14.Text = "";
      //textBox15.Text = "";
      //textBox16.Text = "";
      //textBox17.Text = "";
      textBox18.Text = "";
      textBox19.Text = "";
      textBox20.Text = "";
      textBox21.Text = "";
      textBox22.Text = "";
      textBox23.Text = "";
      textBox24.Text = "";
      textBox25.Text = "";
      textBox26.Text = "";

      data_mess = "";                                                   // ลบข้อมูลที่บันทึกไว้ออกไป

      button1.Enabled = true;
      button2.Enabled = true;
      button3.Enabled = true;
      button4.Enabled = true;
      button5.Enabled = true;
```

```
    button6.Enabled = false;
    button7.Enabled = true;
    button8.Enabled = true;
    button9.Enabled = true;
    button10.Enabled = true;
    button11.Enabled = true;
    button12.Enabled = true;
    button13.Enabled = true;
    button14.Enabled = true;
    button15.Enabled = true;
    button16.Enabled = true;
    button17.Enabled = true;
    button18.Enabled = true;

    radioButton1.Checked = false;
    radioButton2.Checked = false;
    radioButton3.Checked = false;
}

private void button6_Click(object sender, EventArgs e)          // หยุด Auto และ Save
{
    timer2.Enabled = false;
    timer5.Enabled = false;

    button2.Enabled = true;
    button3.Enabled = true;                                     // ปุ่มส่ง
    button4.Enabled = true;                                     // ปุ่มเคลียร์
    button5.Enabled = true;                                     // ปุ่ม Auto
    button7.Enabled = true;
    button8.Enabled = true;
    button9.Enabled = true;
    button10.Enabled = true;
    button11.Enabled = true;
    button12.Enabled = true;
    button13.Enabled = true;
    button14.Enabled = true;
    button15.Enabled = true;
    button16.Enabled = true;
    button17.Enabled = true;
    button18.Enabled = true;
```

```
    saveFileDialog1.Filter = "Text File |*.txt";
    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
    {
        string path = saveFileDialog1.FileName;
        StreamWriter sw = new StreamWriter(path);
        //sw.WriteLine();
        //sw.Write(date.Text);
        //sw.Write("    " + time.Text);
        //sw.Write("    " + data_mess);                    //เขียนข้อมูลที่รับได้ทั้งหมด
        sw.Write(data_mess);                               //เขียนข้อมูลที่รับได้ทั้งหมด
        sw.Close();
    }
}

private void button5_Click(object sender, EventArgs e)    // Auto 28-29
{
    tbyte[0] = 0x82;
    textBox1.Text = tbyte[0].ToString();
    tbyte[1] = 0x30;
    textBox2.Text = tbyte[1].ToString();
    tbyte[2] = 0xEE;
    textBox3.Text = tbyte[2].ToString();
    tbyte[3] = 0x02;
    textBox4.Text = tbyte[3].ToString();
    tbyte[4] = 0x02;
    textBox5.Text = tbyte[4].ToString();
    tbyte[5] = 0xFF;
    textBox6.Text = tbyte[5].ToString();

    data_mess = "";
    tbyte[0] = byte.Parse(textBox1.Text);
    tbyte[1] = byte.Parse(textBox2.Text);
    tbyte[2] = byte.Parse(textBox3.Text);
    tbyte[3] = byte.Parse(textBox4.Text);
    tbyte[4] = byte.Parse(textBox5.Text);
    tbyte[5] = byte.Parse(textBox6.Text);
    timer2.Enabled = true;
    serialPort1.Write(tbyte, 0, 6);
    counter = 0;

    button1.Enabled = false;
    button2.Enabled = false;
    button3.Enabled = false;
```

```
    button4.Enabled = false;
    button5.Enabled = false;
    button6.Enabled = true;
    button7.Enabled = false;
    button8.Enabled = false;
    button9.Enabled = false;
    button10.Enabled = false;
    button11.Enabled = true;
    button12.Enabled = true;
    button13.Enabled = false;
    button14.Enabled = false;
    button15.Enabled = false;
    button16.Enabled = false;
    button17.Enabled = false;
    button18.Enabled = false;
}

private void timer2_Tick(object sender, EventArgs e)            //แสดงผลโหมด Auto
{
    if (counter == 6) counter = 0;

    textBox7.Text = "";
    textBox8.Text = "";
    textBox9.Text = "";
    textBox10.Text = "";
    textBox11.Text = "";
    textBox12.Text = "";

    textBox13.Text = "";
    textBox13.Refresh();
    radioButton3.Checked = true;
    for (double  count  = 5000000; count > 0; count--) ;          //หน่วงเวลานิดหนึ่ง


    textBox7.Text = bbyte[0].ToString();
    textBox8.Text = bbyte[1].ToString();
    textBox9.Text = bbyte[2].ToString();
    textBox10.Text = bbyte[3].ToString();
    textBox11.Text = bbyte[4].ToString();
    textBox12.Text = bbyte[5].ToString();

    textBox13.Text = bbyte[1].ToString() + "." + bbyte[2].ToString() + "  Cel";
    if (bbyte[3] == 0) radioButton1.Checked = false;
```

```
    else radioButton1.Checked = true;
    if (bbyte[4] == 0) radioButton2.Checked = false;
    else radioButton2.Checked = true;

    button6.Enabled = true;
    data_mess += groupBox7.Text + "   " + date.Text + "   " + time.Text + "   " + textBox13.Text + "   " + bbyte[3].ToString() + " " + bbyte[4].ToString() + "\r\n";

    double   i = bbyte[1];
    if (bbyte[2] == 5) { i += 0.5; }
    if (i>27.5)
    {
       tbyte[3] = 1;
       serialPort1.Write(tbyte, 0, 6);
       counter = 0;
    }
    if(i<27.5)
    {
       tbyte[3] = 0;
       serialPort1.Write(tbyte, 0, 6);
       counter = 0;
    }
    /*if (i == 28.5)
    {
       tbyte[3] = 2;
       serialPort1.Write(tbyte, 0, 6);
       counter = 0;
    }*/

    serialPort1.Write(tbyte, 0, 6);
    counter = 0;
    radioButton3.Checked = false;
    for (double count = 5000000; count > 0; count--) ;                    //หน่วงเวลานิดหนึ่ง
}

private void timer3_Tick(object sender, EventArgs e)
{
    date.Text = DateTime.Now.ToLongDateString();
    time.Text = DateTime.Now.ToLongTimeString();
}

private void button7_Click(object sender, EventArgs e)
{
    tbyte[0] = 0x81;
```

```
textBox14.Text = tbyte[0].ToString();
tbyte[1] = 0x30;
textBox15.Text = tbyte[1].ToString();
tbyte[2] = 0x32;
textBox16.Text = tbyte[2].ToString();
tbyte[3] = 0xEE;
textBox17.Text = tbyte[3].ToString();
tbyte[4] = 0x02;
textBox18.Text = tbyte[4].ToString();
tbyte[5] = 0x02;
textBox19.Text = tbyte[5].ToString();
tbyte[6] = 0xFF;
textBox20.Text = tbyte[6].ToString();

tbyte[0] = byte.Parse(textBox14.Text);
tbyte[1] = byte.Parse(textBox15.Text);
tbyte[2] = byte.Parse(textBox16.Text);
tbyte[3] = byte.Parse(textBox17.Text);
tbyte[4] = byte.Parse(textBox18.Text);
tbyte[5] = byte.Parse(textBox19.Text);
tbyte[6] = byte.Parse(textBox20.Text);
timer4.Enabled = true;
serialPort1.Write(tbyte, 0, 7);
counter = 0;

button1.Enabled = false;
button2.Enabled = true;
button3.Enabled = false;
button4.Enabled = true;
button5.Enabled = false;
button6.Enabled = false;
button7.Enabled = true;
button8.Enabled = true;
button9.Enabled = false;
button10.Enabled = false;
button11.Enabled = false;
button12.Enabled = false;
button13.Enabled = true;
button14.Enabled = true;
button15.Enabled = true;
button16.Enabled = true;
button17.Enabled = false;
button18.Enabled = true;
```

```
}

private void timer4_Tick(object sender, EventArgs e)
{
    textBox21.Text = bbyte[0].ToString();
    textBox22.Text = bbyte[1].ToString();
    textBox23.Text = bbyte[2].ToString();
    textBox24.Text = bbyte[3].ToString();
    textBox25.Text = bbyte[4].ToString();
    textBox26.Text = bbyte[5].ToString();

    textBox13.Text = bbyte[1].ToString() + "." + bbyte[2].ToString() + "  Cel";
    if (bbyte[3] == 0) radioButton1.Checked = false;
    else radioButton1.Checked = true;
    if (bbyte[4] == 0) radioButton2.Checked = false;
    else radioButton2.Checked = true;

    if (counter == 7) counter = 0;
    button6.Enabled = true;
    data_mess += groupBox8.Text + "   " + date.Text + "  " + time.Text + textBox13.Text + "   " + bbyte[3].ToString() + " " + bbyte[4].ToString() + "\r\n";
    timer4.Enabled = false;
}

private void button8_Click(object sender, EventArgs e)
{
    tbyte[0] = 0x81;
    textBox14.Text = tbyte[0].ToString();
    tbyte[1] = 0x30;
    textBox15.Text = tbyte[1].ToString();
    tbyte[2] = 0x32;
    textBox16.Text = tbyte[2].ToString();
    tbyte[3] = 0xEE;
    textBox17.Text = tbyte[3].ToString();
    tbyte[4] = 0x02;
    textBox18.Text = tbyte[4].ToString();
    tbyte[5] = 0x02;
    textBox19.Text = tbyte[5].ToString();
    tbyte[6] = 0xFF;
    textBox20.Text = tbyte[6].ToString();

    data_mess = "";
    tbyte[0] = byte.Parse(textBox14.Text);
    tbyte[1] = byte.Parse(textBox15.Text);
```

```
    tbyte[2] = byte.Parse(textBox16.Text);
    tbyte[3] = byte.Parse(textBox17.Text);
    tbyte[4] = byte.Parse(textBox18.Text);
    tbyte[5] = byte.Parse(textBox19.Text);
    tbyte[6] = byte.Parse(textBox20.Text);
    timer5.Enabled = true;
    serialPort1.Write(tbyte, 0, 7);
    counter = 0;

    button1.Enabled = false;
    button2.Enabled = false;
    button3.Enabled = false;
    button4.Enabled = false;
    button5.Enabled = false;
    button6.Enabled = false;
    button7.Enabled = false;
    button8.Enabled = false;
    button9.Enabled = false;
    button10.Enabled = false;
    button11.Enabled = false;
    button12.Enabled = false;
    button13.Enabled = false;
    button14.Enabled = false;
    button15.Enabled = true;
    button16.Enabled = true;
    button17.Enabled = false;
    button18.Enabled = false;
}

private void timer5_Tick(object sender, EventArgs e)
{
    if (counter == 7) counter = 0;

    textBox21.Text = "";
    textBox22.Text = "";
    textBox23.Text = "";
    textBox24.Text = "";
    textBox25.Text = "";
    textBox26.Text = "";

    textBox13.Text = "";
    textBox13.Refresh();
    radioButton3.Checked = true;
```

```
for (double count = 5000000; count > 0; count--) ;                    //หน่วงเวลานิดหนึ่ง

textBox21.Text = bbyte[0].ToString();
textBox22.Text = bbyte[1].ToString();
textBox23.Text = bbyte[2].ToString();
textBox24.Text = bbyte[3].ToString();
textBox25.Text = bbyte[4].ToString();
textBox26.Text = bbyte[5].ToString();

textBox13.Text = bbyte[1].ToString() + "." + bbyte[2].ToString() + "  Cel";
if (bbyte[3] == 0) radioButton1.Checked = false;
else radioButton1.Checked = true;
if (bbyte[4] == 0) radioButton2.Checked = false;
else radioButton2.Checked = true;

button6.Enabled = true;
data_mess += groupBox8.Text + "   " + date.Text + "   " + time.Text + "   " + textBox13.Text + "   " + bbyte[3].ToString() + " " + bbyte[4].ToString() + "\r\n";

double i = bbyte[1];
if (bbyte[2] == 5) { i += 0.5; }
if (i > 27.5)
{
   tbyte[4] = 1;
   serialPort1.Write(tbyte, 0, 7);
   counter = 0;
}
if (i < 27.5)
{
   tbyte[4] = 0;
   serialPort1.Write(tbyte, 0, 7);
   counter = 0;
}
/*if (i == 28.5)
{
   tbyte[3] = 2;
   serialPort1.Write(tbyte, 0, 6);
   counter = 0;
}*/

serialPort1.Write(tbyte, 0, 7);
counter = 0;
radioButton3.Checked = false;
for (double count = 5000000; count > 0; count--) ;  //หน่วงเวลานิดหนึ่ง
```

```
}

private void button9_Click(object sender, EventArgs e)
{
   /*tbyte[0] = 0x82;
   textBox1.Text = tbyte[0].ToString();
   tbyte[1] = 0x30;
   textBox2.Text = tbyte[1].ToString();
   tbyte[2] = 0xEE;
   textBox3.Text = tbyte[2].ToString();*/
   tbyte[3] = 0x01;
   textBox4.Text = tbyte[3].ToString();
   tbyte[4] = 0x02;
   textBox5.Text = tbyte[4].ToString();
   tbyte[5] = 0xFF;
   textBox6.Text = tbyte[5].ToString();

   tbyte[0] = byte.Parse(textBox1.Text);
   tbyte[1] = byte.Parse(textBox2.Text);
   tbyte[2] = byte.Parse(textBox3.Text);
   tbyte[3] = byte.Parse(textBox4.Text);
   tbyte[4] = byte.Parse(textBox5.Text);
   tbyte[5] = byte.Parse(textBox6.Text);
   timer1.Enabled = true;
   serialPort1.Write(tbyte, 0, 6);
   counter = 0;

   button1.Enabled = false;
   button2.Enabled = true;
   button3.Enabled = true;
   button4.Enabled = true;
   button5.Enabled = true;
   button6.Enabled = false;
   button7.Enabled = false;
   button8.Enabled = false;
   button9.Enabled = false;
   button10.Enabled = true;
   button11.Enabled = true;
   button12.Enabled = true;
   button13.Enabled = false;
   button14.Enabled = false;
   button15.Enabled = false;
```

```
    button16.Enabled = false;
    button17.Enabled = true;
    button18.Enabled = false;
}

private void button10_Click(object sender, EventArgs e)
{
    /*tbyte[0] = 0x82;
    textBox1.Text = tbyte[0].ToString();
    tbyte[1] = 0x30;
    textBox2.Text = tbyte[1].ToString();
    tbyte[2] = 0xEE;
    textBox3.Text = tbyte[2].ToString();*/
    tbyte[3] = 0x00;
    textBox4.Text = tbyte[3].ToString();
    tbyte[4] = 0x02;
    textBox5.Text = tbyte[4].ToString();
    tbyte[5] = 0xFF;
    textBox6.Text = tbyte[5].ToString();

    tbyte[0] = byte.Parse(textBox1.Text);
    tbyte[1] = byte.Parse(textBox2.Text);
    tbyte[2] = byte.Parse(textBox3.Text);
    tbyte[3] = byte.Parse(textBox4.Text);
    tbyte[4] = byte.Parse(textBox5.Text);
    tbyte[5] = byte.Parse(textBox6.Text);
    timer1.Enabled = true;
    serialPort1.Write(tbyte, 0, 6);
    counter = 0;

    button1.Enabled = false;
    button2.Enabled = true;
    button3.Enabled = true;
    button4.Enabled = true;
    button5.Enabled = true;
    button6.Enabled = false;
    button7.Enabled = false;
    button8.Enabled = false;
    button9.Enabled = true;
    button10.Enabled = false;
    button11.Enabled = true;
    button12.Enabled = true;
    button13.Enabled = false;
```

```
    button14.Enabled = false;
    button15.Enabled = false;
    button16.Enabled = false;
    button17.Enabled = true;
    button18.Enabled = false;
}

private void button11_Click(object sender, EventArgs e)
{
    /*tbyte[0] = 0x82;
    textBox1.Text = tbyte[0].ToString();
    tbyte[1] = 0x30;
    textBox2.Text = tbyte[1].ToString();
    tbyte[2] = 0xEE;
    textBox3.Text = tbyte[2].ToString();*/
    tbyte[3] = 0x02;
    textBox4.Text = tbyte[3].ToString();
    tbyte[4] = 0x01;
    textBox5.Text = tbyte[4].ToString();
    tbyte[5] = 0xFF;
    textBox6.Text = tbyte[5].ToString();

    tbyte[0] = byte.Parse(textBox1.Text);
    tbyte[1] = byte.Parse(textBox2.Text);
    tbyte[2] = byte.Parse(textBox3.Text);
    tbyte[3] = byte.Parse(textBox4.Text);
    tbyte[4] = byte.Parse(textBox5.Text);
    tbyte[5] = byte.Parse(textBox6.Text);
    timer1.Enabled = true;
    serialPort1.Write(tbyte, 0, 6);
    counter = 0;

    button1.Enabled = false;
    button2.Enabled = true;
    button3.Enabled = true;
    button4.Enabled = true;
    button5.Enabled = true;
    button6.Enabled = false;
    button7.Enabled = false;
    button8.Enabled = false;
    button9.Enabled = true;
    button10.Enabled = true;
    button11.Enabled = false;
```

```
    button12.Enabled = true;
    button13.Enabled = false;
    button14.Enabled = false;
    button15.Enabled = false;
    button16.Enabled = false;
    button17.Enabled = true;
    button18.Enabled = false;
}

private void button12_Click(object sender, EventArgs e)
{
    /*tbyte[0] = 0x82;
    textBox1.Text = tbyte[0].ToString();
    tbyte[1] = 0x30;
    textBox2.Text = tbyte[1].ToString();
    tbyte[2] = 0xEE;
    textBox3.Text = tbyte[2].ToString();*/
    tbyte[3] = 0x02;
    textBox4.Text = tbyte[3].ToString();
    tbyte[4] = 0x00;
    textBox5.Text = tbyte[4].ToString();
    tbyte[5] = 0xFF;
    textBox6.Text = tbyte[5].ToString();

    tbyte[0] = byte.Parse(textBox1.Text);
    tbyte[1] = byte.Parse(textBox2.Text);
    tbyte[2] = byte.Parse(textBox3.Text);
    tbyte[3] = byte.Parse(textBox4.Text);
    tbyte[4] = byte.Parse(textBox5.Text);
    tbyte[5] = byte.Parse(textBox6.Text);
    timer1.Enabled = true;
    serialPort1.Write(tbyte, 0, 6);
    counter = 0;

    button1.Enabled = false;
    button2.Enabled = true;
    button3.Enabled = true;
    button4.Enabled = true;
    button5.Enabled = true;
    button6.Enabled = false;
    button7.Enabled = false;
    button8.Enabled = false;
    button9.Enabled = true;
```

```
    button10.Enabled = true;
    button11.Enabled = true;
    button12.Enabled = false;
    button13.Enabled = false;
    button14.Enabled = false;
    button15.Enabled = false;
    button16.Enabled = false;
    button17.Enabled = true;
    button18.Enabled = false;
}

private void button13_Click(object sender, EventArgs e)
{
    /*byte[0] = 0x81;
    textBox14.Text = tbyte[0].ToString();
    tbyte[1] = 0x30;
    textBox15.Text = tbyte[1].ToString();
    tbyte[2] = 0x32;
    textBox16.Text = tbyte[2].ToString();
    tbyte[3] = 0xEE;
    textBox17.Text = tbyte[3].ToString();*/
    tbyte[4] = 0x01;
    textBox18.Text = tbyte[4].ToString();
    tbyte[5] = 0x02;
    textBox19.Text = tbyte[5].ToString();
    tbyte[6] = 0xFF;
    textBox20.Text = tbyte[6].ToString();

    tbyte[0] = byte.Parse(textBox14.Text);
    tbyte[1] = byte.Parse(textBox15.Text);
    tbyte[2] = byte.Parse(textBox16.Text);
    tbyte[3] = byte.Parse(textBox17.Text);
    tbyte[4] = byte.Parse(textBox18.Text);
    tbyte[5] = byte.Parse(textBox19.Text);
    tbyte[6] = byte.Parse(textBox20.Text);
    timer4.Enabled = true;
    serialPort1.Write(tbyte, 0, 7);
    counter = 0;

    button1.Enabled = false;
    button2.Enabled = true;
    button3.Enabled = false;
    button4.Enabled = true;
```

```
    button5.Enabled = false;
    button6.Enabled = false;
    button7.Enabled = true;
    button8.Enabled = true;
    button9.Enabled = false;
    button10.Enabled = false;
    button11.Enabled = false;
    button12.Enabled = false;
    button13.Enabled = false;
    button14.Enabled = true;
    button15.Enabled = true;
    button16.Enabled = true;
    button17.Enabled = false;
    button18.Enabled = true;
}

private void button14_Click(object sender, EventArgs e)
{
    /*tbyte[0] = 0x81;
    textBox14.Text = tbyte[0].ToString();
    tbyte[1] = 0x30;
    textBox15.Text = tbyte[1].ToString();
    tbyte[2] = 0x32;
    textBox16.Text = tbyte[2].ToString();
    tbyte[3] = 0xEE;
    textBox17.Text = tbyte[3].ToString();*/
    tbyte[4] = 0x00;
    textBox18.Text = tbyte[4].ToString();
    tbyte[5] = 0x02;
    textBox19.Text = tbyte[5].ToString();
    tbyte[6] = 0xFF;
    textBox20.Text = tbyte[6].ToString();

    tbyte[0] = byte.Parse(textBox14.Text);
    tbyte[1] = byte.Parse(textBox15.Text);
    tbyte[2] = byte.Parse(textBox16.Text);
    tbyte[3] = byte.Parse(textBox17.Text);
    tbyte[4] = byte.Parse(textBox18.Text);
    tbyte[5] = byte.Parse(textBox19.Text);
    tbyte[6] = byte.Parse(textBox20.Text);
    timer4.Enabled = true;
    serialPort1.Write(tbyte, 0, 7);
    counter = 0;
```

```
    button1.Enabled = false;
    button2.Enabled = true;
    button3.Enabled = false;
    button4.Enabled = true;
    button5.Enabled = false;
    button6.Enabled = false;
    button7.Enabled = true;
    button8.Enabled = true;
    button9.Enabled = false;
    button10.Enabled = false;
    button11.Enabled = false;
    button12.Enabled = false;
    button13.Enabled = true;
    button14.Enabled = false;
    button15.Enabled = true;
    button16.Enabled = true;
    button17.Enabled = false;
    button18.Enabled = true;
}

private void button15_Click(object sender, EventArgs e)
{
    /*tbyte[0] = 0x81;
    textBox14.Text = tbyte[0].ToString();
    tbyte[1] = 0x30;
    textBox15.Text = tbyte[1].ToString();
    tbyte[2] = 0x32;
    textBox16.Text = tbyte[2].ToString();
    tbyte[3] = 0xEE;
    textBox17.Text = tbyte[3].ToString();*/
    tbyte[4] = 0x02;
    textBox18.Text = tbyte[4].ToString();
    tbyte[5] = 0x01;
    textBox19.Text = tbyte[5].ToString();
    tbyte[6] = 0xFF;
    textBox20.Text = tbyte[6].ToString();

    tbyte[0] = byte.Parse(textBox14.Text);
    tbyte[1] = byte.Parse(textBox15.Text);
    tbyte[2] = byte.Parse(textBox16.Text);
    tbyte[3] = byte.Parse(textBox17.Text);
    tbyte[4] = byte.Parse(textBox18.Text);
```

```
    tbyte[5] = byte.Parse(textBox19.Text);
    tbyte[6] = byte.Parse(textBox20.Text);
    timer4.Enabled = true;
    serialPort1.Write(tbyte, 0, 7);
    counter = 0;

    button1.Enabled = false;
    button2.Enabled = true;
    button3.Enabled = false;
    button4.Enabled = true;
    button5.Enabled = false;
    button6.Enabled = false;
    button7.Enabled = true;
    button8.Enabled = true;
    button9.Enabled = false;
    button10.Enabled = false;
    button11.Enabled = false;
    button12.Enabled = false;
    button13.Enabled = true;
    button14.Enabled = true;
    button15.Enabled = false;
    button16.Enabled = true;
    button17.Enabled = false;
    button18.Enabled = true;
}

private void button16_Click(object sender, EventArgs e)
{
    /*tbyte[0] = 0x81;
    textBox14.Text = tbyte[0].ToString();
    tbyte[1] = 0x30;
    textBox15.Text = tbyte[1].ToString();
    tbyte[2] = 0x32;
    textBox16.Text = tbyte[2].ToString();
    tbyte[3] = 0xEE;
    textBox17.Text = tbyte[3].ToString();*/
    tbyte[4] = 0x02;
    textBox18.Text = tbyte[4].ToString();
    tbyte[5] = 0x00;
    textBox19.Text = tbyte[5].ToString();
    tbyte[6] = 0xFF;
    textBox20.Text = tbyte[6].ToString();
```

```
    tbyte[0] = byte.Parse(textBox14.Text);
    tbyte[1] = byte.Parse(textBox15.Text);
    tbyte[2] = byte.Parse(textBox16.Text);
    tbyte[3] = byte.Parse(textBox17.Text);
    tbyte[4] = byte.Parse(textBox18.Text);
    tbyte[5] = byte.Parse(textBox19.Text);
    tbyte[6] = byte.Parse(textBox20.Text);
    timer4.Enabled = true;
    serialPort1.Write(tbyte, 0, 7);
    counter = 0;

    button1.Enabled = false;
    button2.Enabled = true;
    button3.Enabled = false;
    button4.Enabled = true;
    button5.Enabled = false;
    button6.Enabled = false;
    button7.Enabled = true;
    button8.Enabled = true;
    button9.Enabled = false;
    button10.Enabled = false;
    button11.Enabled = false;
    button12.Enabled = false;
    button13.Enabled = true;
    button14.Enabled = true;
    button15.Enabled = true;
    button16.Enabled = false;
    button17.Enabled = false;
    button18.Enabled = true;
}

private void groupBox14_Enter(object sender, EventArgs e)
{

}

private void button17_Click(object sender, EventArgs e)
{
    string f = "D:\\GiFtziKA\\Project\\Test_txrx\\";                    //กำหนดที่เซฟ
    int i;
    string fb = "Farm 1.txt";                                           //ชื่อ

    StreamWriter sw = File.AppendText(@f + fb);
```

```
    sw.WriteLine();
    sw.Write(groupBox7.Text);
    sw.Write("    " + date.Text);
    sw.Write("    " + time.Text);
    sw.Write("    " + textBox13.Text);
    sw.Write("    " + bbyte[3].ToString());
    sw.Write("  " + bbyte[4].ToString());
    sw.Close();                                          //ปิดไฟล์แต่ละครั้ง

    button1.Enabled = false;
    button2.Enabled = true;
    button3.Enabled = true;
    button4.Enabled = true;
    button5.Enabled = true;
    button6.Enabled = false;
    button7.Enabled = false;
    button8.Enabled = false;
    button9.Enabled = true;
    button10.Enabled = true;
    button11.Enabled = true;
    button12.Enabled = true;
    button13.Enabled = false;
    button14.Enabled = false;
    button15.Enabled = false;
    button16.Enabled = false;
    button17.Enabled = true;
    button18.Enabled = false;
}

private void button18_Click(object sender, EventArgs e)
{
    string f = "D:\\GiFtziKA\\Project\\Test_txrx\\";          //กำหนดที่เซฟ
    int i;
    string fb = "Farm 2.txt";  //ชื่อ

    StreamWriter sw = File.AppendText(@f + fb);

    sw.WriteLine();
    sw.Write(groupBox8.Text);
    sw.Write("    " + date.Text);
    sw.Write("    " + time.Text);
    sw.Write("    " + textBox13.Text);
```

```
            sw.Write("     " + bbyte[4].ToString());
            sw.Write("  " + bbyte[5].ToString());
            sw.Close();                                          //ปิดไฟล์แต่ละครั้ง

            button1.Enabled = false;
            button2.Enabled = true;
            button3.Enabled = false;
            button4.Enabled = true;
            button5.Enabled = false;
            button6.Enabled = false;
            button7.Enabled = true;
            button8.Enabled = true;
            button9.Enabled = false;
            button10.Enabled = false;
            button11.Enabled = false;
            button12.Enabled = false;
            button13.Enabled = true;
            button14.Enabled = true;
            button15.Enabled = true;
            button16.Enabled = true;
            button17.Enabled = false;
            button18.Enabled = true;
        }
    }
}
```

SOURCE CODE CLIENT

Code MPLAB

คำสั่ง

```
  ------------
 | 0x81/82 | <---- ค่าเริ่มต้นจะเป็นตัวบอกว่าจะเป็นการส่งคำสั่งต่อหรือสั่งกลับข้อมูล
  ------------
 | ID ผู้ส่ง มา      | <------- ID ของผู้ที่ส่งคำสั่งมา
  ------------
 |Next ID   | <------- ID ของผู้ที่จะถูกส่งต่อไป
  ------------
 | ID     | <------- ID ของตัวที่ต้องการอ่านค่าอุณหภูมิ
  ------------
 | 0XEE   | <----- บอกว่าหมดตัวส่งต่อแล้ว(ตัวหน้านี้เป็นตัวที่ต้องการอ่านอุณหภูมิ)
  ------------
 | Relay0  | <----- 0x01 = ON, 0x00 = OFF,0x02=Read only
  ------------
 | Relay1  | <----- 0x01 = ON, 0x00 = OFF,0x02=Read only
  ------------
 | 0xFF   | <----- จบ
  ------------
```

ข้อมูลกลับ

```
  ------------
 | 0x81/82 | <---- ค่าเริ่มต้นจะเป็นตัวบอกว่าจะเป็นการส่งคำสั่งต่อหรือสั่งกลับข้อมูล
  ------------
 | Temp(f) | <------- อุณหภูมิเลขหน้าจุด
  ------------
 | Temp(b) | <----- อุณหภูมิเลขหลังจุด
  ------------
 | Relay0  | <----- 0x01 = ON, 0x00 = OFF
  ------------
 | Relay1  | <----- 0x01 = ON, 0x00 = OFF
  ------------
 | 0xFF   | <----- จบ
  ------------
```

*/

```
#include <p30f4011.h>                    // dsPIC30F4011 MPU Register
#include "stdio.h"                                                                    // Used
"sprintf" Function
#include "uart.h"

/* Setup Configuration For ET-BASE dsPIC30F4011 */
_FOSC(CSW_FSCM_OFF & XT_PLL16);          // Disable Clock Switching,Enable Fail-Salf Clock
                                         // Clock Source = Primary XT + (PLL x 16)
_FWDT(WDT_OFF);                          // Disable Watchdog
_FBORPOR(PBOR_OFF & PWRT_64 & MCLR_EN);  // Disable Brown-Out ,Power ON = 64mS,Enable MCLR
_FGS(CODE_PROT_OFF);                     // Code Protect OFF

/* End Configuration For ET-BASE dsPIC30F4011 */

#define CodeStop    0xFF     // Key Code บอกจบคำสั่งหรือข้อมูล
//#define MY_ID      0x32     //กำหนดเลขหมายตัวเองหมายเลข '2'
#define MY_ID       0x31     //กำหนดเลขหมายตัวเองหมายเลข '1'

//---- LED (Relay#0)
#define TRIS_Rly0 TRISFbits.TRISF1
#define Relay0  LATFbits.LATF1
#define TRIS_Rly1 TRISFbits.TRISF2
#define Relay1  LATFbits.LATF2

// Character LCD Interface Pins
#define TRIS_DATA_PIN_4   TRISBbits.TRISB0     // Direction D4
#define TRIS_DATA_PIN_5   TRISBbits.TRISB1     // Direction D5
#define TRIS_DATA_PIN_6   TRISBbits.TRISB2     // Direction D6
#define TRIS_DATA_PIN_7   TRISBbits.TRISB3     // Direction D7
#define TRIS_RS       TRISBbits.TRISB4         // Direction RS
#define TRIS_E        TRISBbits.TRISB5         // Direction E

#define DATA_PIN_4      LATBbits.LATB0         // RB0 = D4 LCD
#define DATA_PIN_5      LATBbits.LATB1         // RB1 = D5 LCD
#define DATA_PIN_6      LATBbits.LATB2         // RB2 = D6 LCD
#define DATA_PIN_7      LATBbits.LATB3         // RB3 = D7 LCD
#define RS_PIN        LATBbits.LATB4           // RB4 = RS LCD
#define E_PIN         LATBbits.LATB5           // RB5 = E  LCD
```

```
/* Display ON/OFF Control */
#define DON             0x0F        // Display on
#define DOFF            0x0B        // Display off
#define CURSOR_ON       0x0F        // Cursor on
#define CURSOR_OFF      0x0D        // Cursor off
#define BLINK_ON        0x0F        // Cursor Blink
#define BLINK_OFF       0x0E        // Cursor No Blink


/* Cursor or Display Shift */
#define SHIFT_CUR_LEFT    0x13      // Cursor shifts to the left
#define SHIFT_CUR_RIGHT   0x17      // Cursor shifts to the right
#define SHIFT_DISP_LEFT   0x1B      // Display shifts to the left
#define SHIFT_DISP_RIGHT  0x1F      // Display shifts to the right

/* Function Prototypes */
void Initial_4bitLCD(void);                 // Initial LCD Interface
void SetCGRamAddr(unsigned char);
void SetDDRamAddr(unsigned char);           // Set Cursor Address
void WriteCmdLCD(unsigned char);            // Write Command
void WriteDataLCD(unsigned char);           // Write Data
void PutsLCD(unsigned char*);               // Print Message
void Delay_tW_LCD(void);                    // Enable Pulse Delay
void Busy_LCD(void);                        // Wait LCD Busy
void Delay(unsigned long int);              // Delay Time Function

//---- For DS1820
//#define TRIS_DS1820   TRISEbits.TRISE0    // Direction E0
//#define DOUT_DS1820   LATEbits.LATE0      // OUT
//#define DIN_DS1820    PORTEbits.RE0       // IN

#define TRIS_DS1820   TRISFbits.TRISF0      // Direction E0
#define DOUT_DS1820   LATFbits.LATF0        // OUT
#define DIN_DS1820    PORTFbits.RF0         // IN

unsigned char tempresbit(void);
void tempwrbyte(unsigned char dat);
```

```
char temprdbyte(void);
char rdtemp(void);

void Forward(void);
void Backward(void);


char buf_Tempc[20]={'\0','\0','\0','\0','\0'                    // Set buffer for display Temp
                                                                ,'\0','\0','\0','\0','\0'
                                                                ,'\0','\0','\0','\0','\0'
                                                                ,'\0','\0','\0','\0','\0'};


//-------------------------:Global variables
unsigned char Buf[80];          // Received Command/data is stored in array Buf
unsigned char BBuf[20];
unsigned char data_tmp[80];
unsigned char ch;
unsigned char c_buf=0;
unsigned char addr_back;        //ไว้เก็บ My เพื่อส่งกลับ

//-----------------------------------------------------:U1RX ISR
// UART1 Receiver Interrupt Service Routine
void _ISR _U1RXInterrupt(void)
{
  _U1RXIF = 0;   // Clear interrupt flag RX

  // Read the receive buffer till atleast one or more character can be read
  while (DataRdyUART1()) {
    ch = ReadUART1();     // Read buffer rx
    //putcUART1(ch);       // print character rx
    Buf[c_buf++] = ch;    // wirte to buf
  }
}

//-----------------------------------------------------:U1TX ISR
// UART1 Transmitter Interrupt Service Routine
```

```
void _ISR _U1TXInterrupt(void)
{
  _U1TXIF = 0;  // Clear interrupt flag TX
}

//-----------------------------------------------------:Delay MS
// Delay 1 ms (XT w/PLL 4x)
void Delay_MS(unsigned int ms)
{
  unsigned int i;

  for (; ms>0; ms--)
    for (i=0; i<728; i++)
      Nop();              // delay 1 mch cycle
}

//-----------------------------------------------------:Uart1_Init
// Initialize UART1
void Uart1_Init(void)
{
  unsigned int U1MODEvalue, U1STAvalue, BaudRate;

  CloseUART1();                                  // Close UART1

  // Config Interrupt UART1
  ConfigIntUART1(UART_RX_INT_EN &                // Receive interrupt enabled
           UART_RX_INT_PR6 &                     // Priority RX interrupt 6
           UART_TX_INT_EN &                      // transmit interrupt enabled
           UART_TX_INT_PR2 );                    // Priority TX interrupt 2

  U1MODEvalue = UART_EN &                        // Module enable
          UART_IDLE_CON &                        // Work in IDLE mode
          //UART_RX_TX &                         // Communication through the normal pins
          //UART_ALTRX_ALTTX &                   // Communication through ALT pins

          UART_DIS_WAKE &                        // Disable Wake-up on START bit Detect during SLEEP Mode bit
```

```
        UART_DIS_LOOPBACK &             // Loop back disabled
        UART_DIS_ABAUD &                // Input to Capture module from ICx pin
        UART_NO_PAR_8BIT &              // no parity 8 bit
        UART_1STOPBIT;                  // 1 stop bit

        U1STAvalue = UART_INT_TX_BUF_EMPTY &  // Interrupt on TXBUF becoming empty
        UART_TX_PIN_NORMAL &            // UART TX pin operates normally
        UART_INT_RX_CHAR &              // Interrupt on every char received
        UART_ADR_DETECT_DIS &           // address detect disable
        UART_RX_OVERRUN_CLEAR;          // Rx buffer Over run status bit clear

 BaudRate = 191;                        // BaudRate 9600 pbs

 // Open UART1
 OpenUART1(U1MODEvalue, U1STAvalue, BaudRate);
}

//-----------------------------------------------------:Uart1 PrintStr
// print string to uart1
void Uart1_PrintStr(unsigned char *str_uart)
{
 putsUART1 ((unsigned int *)str_uart);
 while (BusyUART1());                   // wait for transmission to complete
}

//---------- SUBrountine ----------
void Initial_4bitLCD(void)
{
 TRIS_DATA_PIN_4 = 0;                                                    // Set Port Direction =
Output
 TRIS_DATA_PIN_5 = 0;
 TRIS_DATA_PIN_6 = 0;
 TRIS_DATA_PIN_7 = 0;
 TRIS_RS = 0;
 TRIS_E = 0;

 RS_PIN = 0;                            // Instruction Select
 E_PIN = 0;                             // Disable Strobe
```

```
  Delay(50000);                          // Power-ON Delay (minimum of 15ms)

  WriteCmdLCD(0x33);                     // Initial 4 Bit Interface Format
  WriteCmdLCD(0x32);
  WriteCmdLCD(0x28);                     // Function Set (DL=0 4-Bit,N=1 2 Line,F=0 5X7)


  WriteCmdLCD(0x0C);                     // Display on/off Control (Entry Display,Cursor off,Cursor not Blink)
  WriteCmdLCD(0x06);                     // Entry Mode Set (I/D=1 Increment,S=0 Cursor Shift)
  WriteCmdLCD(0x01);                     // Clear Display  (Clear Display,Set DD RAM Address=0)
}

void SetCGRamAddr(unsigned char address)
{
  RS_PIN = 0;                            // Select Instruction
  DATA_PIN_7 = 0;                        // DB7 Must be 0 For Setting CGRam ADDR
  DATA_PIN_6 = 1;                        // DB6 Must be 1 For Setting CGRam ADDR
  DATA_PIN_5 = ((address & 0x20)>>5);
  DATA_PIN_4 = ((address & 0x10)>>4);
  E_PIN = 1;                             // Strobe High Nibble
  Delay_tW_LCD();                        // Enable Pulse Delay
  E_PIN = 0;
  Busy_LCD();


  RS_PIN = 0;                            // Select Instruction
  DATA_PIN_7 = ((address & 0x08)>>3);
  DATA_PIN_6 = ((address & 0x04)>>2);
  DATA_PIN_5 = ((address & 0x02)>>1);
  DATA_PIN_4 = ((address & 0x01)>>0);
  E_PIN = 1;                             // Strobe Low Nibble
  Delay_tW_LCD();                        // Enable Pulse Delay
  E_PIN = 0;
  Busy_LCD();
}

void SetDDRamAddr(unsigned char address)
{
```

```
  RS_PIN = 0;                                   // Select Instruction
  DATA_PIN_7 = 1;                               // DB7 Must be 1 For Setting DDRam ADDR
  DATA_PIN_6 = ((address & 0x40)>>6);
  DATA_PIN_5 = ((address & 0x20)>>5);
  DATA_PIN_4 = ((address & 0x10)>>4);
  E_PIN = 1;                                    // Strobe High Nibble
  Delay_tW_LCD();                               // Enable Pulse Delay
  E_PIN = 0;
  Busy_LCD();

  RS_PIN = 0;                                   // Select Instruction
  DATA_PIN_7 = ((address & 0x08)>>3);
  DATA_PIN_6 = ((address & 0x04)>>2);
  DATA_PIN_5 = ((address & 0x02)>>1);
  DATA_PIN_4 = ((address & 0x01)>>0);
  E_PIN = 1;                                    // Strobe Low Nibble
  Delay_tW_LCD();                               // Enable Pulse Delay
  E_PIN =0;
  Busy_LCD();
}

void WriteCmdLCD(unsigned char cmd)
{
  RS_PIN = 0;                                   // Select Instruction
  DATA_PIN_7 = ((cmd & 0x80)>>7);
  DATA_PIN_6 = ((cmd & 0x40)>>6);
  DATA_PIN_5 = ((cmd & 0x20)>>5);
  DATA_PIN_4 = ((cmd & 0x10)>>4);
  E_PIN = 1;                                    // Strobe High Nibble
  Delay_tW_LCD();                               // Enable Pulse Delay
  E_PIN = 0;
  Busy_LCD();

  RS_PIN = 0;                                   // Select Instruction
  DATA_PIN_7 = ((cmd & 0x08)>>3);
  DATA_PIN_6 = ((cmd & 0x04)>>2);
  DATA_PIN_5 = ((cmd & 0x02)>>1);
  DATA_PIN_4 = (cmd & 0x01);
```

```
  E_PIN = 1;                                  // Strobe Low Nibble
  Delay_tW_LCD();                             // Enable Pulse Delay
  E_PIN = 0;
  Busy_LCD();
  Delay(50000);                               // 1.64mS Delay
 }

 void WriteDataLCD(unsigned char data)
 {
  RS_PIN = 1;                                 // Select Data
  DATA_PIN_7 = ((data & 0x80)>>7);
  DATA_PIN_6 = ((data & 0x40)>>6);
  DATA_PIN_5 = ((data & 0x20)>>5);
  DATA_PIN_4 = ((data & 0x10)>>4);
  E_PIN = 1;                                  // Strobe High Nibble
  Delay_tW_LCD();                             // Enable Pulse Delay
  E_PIN = 0;
  Busy_LCD();

  RS_PIN = 1;                                 // Select Data
  DATA_PIN_7 = ((data & 0x08)>>3);
  DATA_PIN_6 = ((data & 0x04)>>2);
  DATA_PIN_5 = ((data & 0x02)>>1);
  DATA_PIN_4 = (data & 0x01);
  E_PIN = 1;                                  // Strobe Low Nibble
  Delay_tW_LCD();                             // Enable Pulse Delay
  E_PIN = 0;
  Busy_LCD();
 }

 void PutsLCD(unsigned char* buffer)
 {
  while(*buffer != '\0')                      // Loop Until End of Message
  {
   WriteDataLCD(*buffer);                     // Write Character to LCD
   buffer++;
  }
}
```

```
void Delay_tW_LCD(void)                    // Enable Pulse Delay
{
  int i;
  for(i=0;i<18;i++);
}

void Busy_LCD(void)                        // 40uS Delay
{
  unsigned int i;
  for(i=0;i<1800;i++);
}

/***********************/
/* Delay Time Function */
/*    1-4294967296     */
/***********************/
void Delay(unsigned long int count1)
{
  while(count1 > 0) {count1--;}            // Loop Decrease Counter
}
//***Function for DS1820 **********************************

unsigned char tempresbit()
{
        int i,j;

 TRIS_DS1820 = 0;                          // set output
 DOUT_DS1820 = 0;                          // Send LOW
 for(j=1764;j>0;j--)                       //Delay 600 uS
    Nop();
 DOUT_DS1820 = 1;                          /* end reset pulse */

 TRIS_DS1820 = 1;                          // กำหนดเป็น INPUT
 for(j=176;j>0;j--)
    Nop();                                 // Delay 60 uS
 if(DIN_DS1820)
  return 0;
```

```
  for(j=1764;j>0;j--)                          //Delay 600 uS
     Nop();
   return 1;
}

void tempwrbyte(unsigned char dat)
{
 unsigned char i,j;

 for(i=0;i<8;i++)
 {
  TRIS_DS1820 = 0;                            // set OUTPUT
  DOUT_DS1820 = 0;
  for(j=0;j<44;j++)
     Nop();                                   // Delay 15 uS

 if(dat & 0x01)
  DOUT_DS1820=1;
  else
  DOUT_DS1820=0;

 for(j=176;j>0;j--)
     Nop();                                   // Delay 60 uS
 DOUT_DS1820 = 1;
 dat >>= 1;

 for(j=0;j<30;j++)
     Nop();                                   // Delay 10 uS
 }
}

char temprdbyte()
{
 unsigned char dat,i,j;

 for(i=0;i<8;i++)
 {
  dat >>= 1;
```

```
    TRIS_DS1820 = 0;
    DOUT_DS1820 = 0;
    for(j=0;j<44;j++)
     Nop();                                  // Delay 15 uS
    DOUT_DS1820 = 1;

    TRIS_DS1820 = 1;
    if(DIN_DS1820)
     dat |= 0x80;                            // input = 1
     else
     dat &= 0x7F;                            // input = 0
     for(j=176;j>0;j--)
     Nop();                                  // Delay 60 uS

  }
  return dat;
}

char rdtemp()
{
  char dat;
  int j;

  if(tempresbit())
  {
    tempwrbyte(0xCC);                        //Skip rom
    tempwrbyte(0x44);                        //Temp Convert
    for(j=1470;j>0;j--)
     Nop();                                  // Delay 500 uS
     tempresbit();
     tempwrbyte(0xCC);
     tempwrbyte(0xBE);

     dat = temprdbyte();
     return dat;
  }
  return 0;
}
```

```
void Forward()
{
        unsigned char rx_data[5];
        unsigned char rx_dat, i;

        //เก็บ MY ของตัวส่งคำสั่งมาเพื่อเอาไว้ส่งกลับได้ถูกต้อง
        addr_back = Buf[1];

        if(Buf[3]==0xEE)
          data_tmp[0]=0x82;                    //กำหนดหัวข้อมูลให้ตัวต่อไปเป็นการส่งกลับ
        else data_tmp[0]=0x81;                 //กำหนดหัวข้อมูลให้ตัวต่อไปเป็นการส่งต่อ
          data_tmp[1]= MY_ID;

    //กำหนดเส้นทางส่งต่อไป
     ch = 0;
     c_buf=0;
    while(BusyUART1());
            WriteUART1('+');
    while(BusyUART1());
            WriteUART1('+');
    while(BusyUART1());
            WriteUART1('+');

   rx_dat='+';
   //--- รอคำตอบเป็น 'ok'
     while(ch!=0x4B);
    ch = 0;
    c_buf=0;
    while(ch!=0x0D);
    ch = 0;
    c_buf=0;

        Delay(10000);

//ส่งคำสั่งเปลี่ยนค่าของ DL
```

```
while(BusyUART1());
                    WriteUART1('A');
while(BusyUART1());
                    WriteUART1('T');
while(BusyUART1());
                    WriteUART1('D');
while(BusyUART1());
                    WriteUART1('L');
while(BusyUART1());
                    WriteUART1(Buf[2]);          //ค่า MY ของตัวที่จะส่งต่อไป
while(BusyUART1());                              // Enter
                    WriteUART1(0x0D);


rx_dat='+';
//--- รอคำตอบเป็น 'ok'
        while(ch!=0x4B);
 ch = 0;
 c_buf=0;
 while(ch!=0x0D);
 ch = 0;
 c_buf=0;

//ปิดการเขียน
while(BusyUART1());
                    WriteUART1('A');
        while(BusyUART1());
                    WriteUART1('T');
        while(BusyUART1());
                    WriteUART1('C');
        while(BusyUART1());
                    WriteUART1('N');
while(BusyUART1());
                    WriteUART1(0x0D);            // Enter


rx_dat='+';
 i = 0;
//--- รอคำตอบเป็น 'ok'
 while(ch!=0x4B);
```

```
    ch = 0;
    c_buf=0;
    while(ch!=0x0D);
    ch = 0;
    c_buf=0;

      Delay(100000);      //หยุดสักครู่เพื่อให้ Xbee set ตัวเอง

    //เตรียมข้อมูลส่วนที่เหลือและส่งต่อได้==========================
   i=2;
   while(But[i]!=0xFF)
   {
     data_tmp[i]=But[i+1];
     i++;
   }
     data_tmp[i]=0xFF;  //ใส่รหัสตัวสุดท้าย

   //ส่งข้อมูลต่อไปได้==========================================
   i=0;
    //คำสั่งทั้งหมด
    while(data_tmp[i]!=0xFF)
    {
              while(BusyUART1());
                       WriteUART1(data_tmp[i]);
     i++;
    }
   while(BusyUART1());
                       WriteUART1(data_tmp[i]);   //ส่งตัวสุดท้าย 0xFF ออกไป
}



void Backward()
{
          unsigned char rx_dat;
  int i,j;

   data_tmp[0]=0x82;  //รหัสส่งกลับ
  //ตรวจสอบว่าเป็นตัวที่ถูกอ่านอุณหภูมิหรือถูกสั่งรีเลย์ทำงานหรือไม่
```

```
if(Buf[2]==0xEE)
  { //เป็นตัวสุดท้าย ดังนั้นจึงทำการตรวจสอบว่ามีความต้องการสั่งรีเลย์อย่างไร
            //และทำการใส่ค่าอุณหภูมิ

            j = rdtemp();
if(j&0x01)
  {
                  i =5;
                  j >>= 1;
                  }
        else
                  {
                        i = 0;
                        j >>= 1;
                  }

            data_tmp[1]=(unsigned char) j;  //ค่าอุณหภูมิตัวหน้าจุดทศนิยม
            data_tmp[2] =(unsigned char) i; //ค่าทศนิยมอุณหภูมิ
                        //data_tmp[1]=0xAA;  //ค่าอุณหภูมิตัวหน้าจุดทศนิยม
            //data_tmp[2] =0xBB; //ค่าทศนิยมอุณหภูมิ

            //ตรวจการอ่านหรือสั่งรีเลย์ทำงาน
            //หาตำแหน่งรีเลย์ตัวที่หนึ่ง
             i=0;
             while(Buf[i]!= 0xEE)
             { i++;}

            //ตรวจการอ่านหรือสั่งรีเลย์ทำงาน
            if(Buf[i+1]==0x01)                  //สั่ง On รีเลย์ตัวที่หนึ่ง
              { Relay0 = 1; }
             else if (Buf[i+1]==0x00)           //สั่ง OFF รีเลย์ตัวที่หนึ่ง
              { Relay0 = 0;}

               //อ่านสถานะตนเองแล้วกลับไป
            data_tmp[3]= Relay0;                //เก็บค่าของรีเลย์ตัวที่หนึ่งไว้ส่งกลับ
```

```
            //รีเลย์ตัวที่สอง
            if(Buf[i+2]==0x01)                  //สั่ง On รีเลย์ตัวที่สอง
              // Relay1 = 1;
{  Relay1 = 1; }
            else if (Buf[i+2]==0x00)            //สั่ง OFF รีเลย์ตัวที่สอง
              // Relay1 = 0;
              { Relay1 = 0;
        }
              //อ่านสถานะตนเองแล้วกลับไป

                    data_tmp[4]= Relay1;      /อ่านค่าของ Relay ตัวที่สอง
                  data_tmp[5]= 0xFF;  //จบข้อมูล


//กำหนดเส้นทางส่งกลับโดยเปลี่ยนค่าของ DL
  ch = 0;
  c_buf=0;
  while(BusyUART1());
            WriteUART1('+');
  while(BusyUART1());
            WriteUART1('+');
  while(BusyUART1());
            WriteUART1('+');

 rx_dat='+';
 //--- รอคำตอบเป็น 'ok'

  while(ch!=0x4B);
  ch = 0;
  c_buf=0;
  while(ch!=0x0D);
  ch = 0;
  c_buf=0;

        Delay(10000);

          //ส่งคำสั่งเปลี่ยนค่าของ DL
  while(BusyUART1());
```

```
                WriteUART1('A');
              while(BusyUART1());
                WriteUART1('T');
              while(BusyUART1());
                WriteUART1('D');
              while(BusyUART1());
                WriteUART1('L');
  while(BusyUART1());
                WriteUART1(addr_back);          //ค่า MY ของตัวที่จะส่งกลับไป
  while(BusyUART1());
                WriteUART1(0x0D);               // Enter

 rx_dat='+';
 //--- รอคำตอบเป็น 'ok'

 while(ch!=0x4B);
  ch = 0;
  c_buf=0;
  while(ch!=0x0D);
  ch = 0;
  c_buf=0;

 //ปิดการเขียน
  while(BusyUART1());
                WriteUART1('A');
              while(BusyUART1());
                WriteUART1('T');
             while(BusyUART1());
                WriteUART1('C');
              while(BusyUART1());
                WriteUART1('N');
              while(BusyUART1());
                WriteUART1(0x0D);               // Enter

rx_dat='+';
 i = 0;
 //--- รอคำตอบเป็น 'ok'
```

```
  while(ch!=0x4B);
  ch = 0;
  c_buf=0;
  while(ch!=0x0D);
  ch = 0;
  c_buf=0;


 Delay(100000);                                  //หยุดสักครู่เพื่อให้ Xbee set ตัวเอง

 //ส่งข้อมูลกลับต่อไป ==========================================
 i=0;
  while(data_tmp[i]!=0xFF)
 {
            while(BusyUART1());
             WriteUART1(data_tmp[i]);
   i++;
 }
 while(BusyUART1());
             WriteUART1(data_tmp[i]);            //ส่งตัวสุดท้าย 0xFF ออกไป
 Delay(100000);
} else
 {
            //ก็อปปี้ ข้อมูลไว้
            for(i=0;i<20;i++)
            BBuf[i]=Buf[i];


            //เปลี่ยนค่าของ DL เป็นการส่งกลับไปยังตัวส่งคำสั่งมาตอนแรก
//กำหนดเส้นทางส่งกลับโดยเปลี่ยนค่าของ DL
 ch = 0;
 c_buf=0;
 while(BusyUART1());
             WriteUART1('+');
 while(BusyUART1());
             WriteUART1('+');
 while(BusyUART1());
             WriteUART1('+');
```

```
// rx_dat='+';
 //--- รอคำตอบเป็น 'ok'

   while(ch!=0x4B);
   ch = 0;
   c_buf=0;
   while(ch!=0x0D);
   ch = 0;
   c_buf=0;

            Delay(10000);

             //ส่งคำสั่งเปลี่ยนค่าของ DL
  while(BusyUART1());
               WriteUART1('A');
             while(BusyUART1());
               WriteUART1('T');
             while(BusyUART1());
               WriteUART1('D');
             while(BusyUART1());
               WriteUART1('L');
  while(BusyUART1());
               WriteUART1(addr_back);          //ค่า MY ของตัวที่จะส่งกลับไป
  while(BusyUART1());
               WriteUART1(0x0D);               // Enter

rx_dat='+';
//--- รอคำตอบเป็น 'ok'

while(ch!=0x4B);
 ch = 0;
 c_buf=0;
 while(ch!=0x0D);
 ch = 0;
 c_buf=0;

//ปิดการเขียน
```

```
   while(BusyUART1());
                    WriteUART1('A');
                  while(BusyUART1());
                    WriteUART1('T');
                 while(BusyUART1());
                    WriteUART1('C');
                  while(BusyUART1());
                    WriteUART1('N');
                   while(BusyUART1());
                    WriteUART1(0x0D);                    // Enter

rx_dat='+';
 i = 0;
 //--- รอคำตอบเป็น 'ok'

 while(ch!=0x4B);
  ch = 0;
  c_buf=0;
 while(ch!=0x0D);
  ch = 0;
  c_buf=0;


Delay(100000);                                   //แขุดสักครู่เพื่อให้ Xbee set ตัวเอง


            //ส่งกลับต่อไปแบบไม่ใช่ตัวที่ถูกอ่านอุณหภูมิหรือรีเลย์ ==========
//ก็อปปี้ ข้อมูลกลับคืน
            for(i=0;i<20;i++)
            Buf[i]=BBuf[i];


i=1;
while(BusyUART1());
            WriteUART1(data_tmp[0]);                 //ส่งตัวแรกรหัส 0x82 ออกไป
while(Buf[i]!=0xFF)
{
            while(BusyUART1());
            WriteUART1(Buf[i]);
```

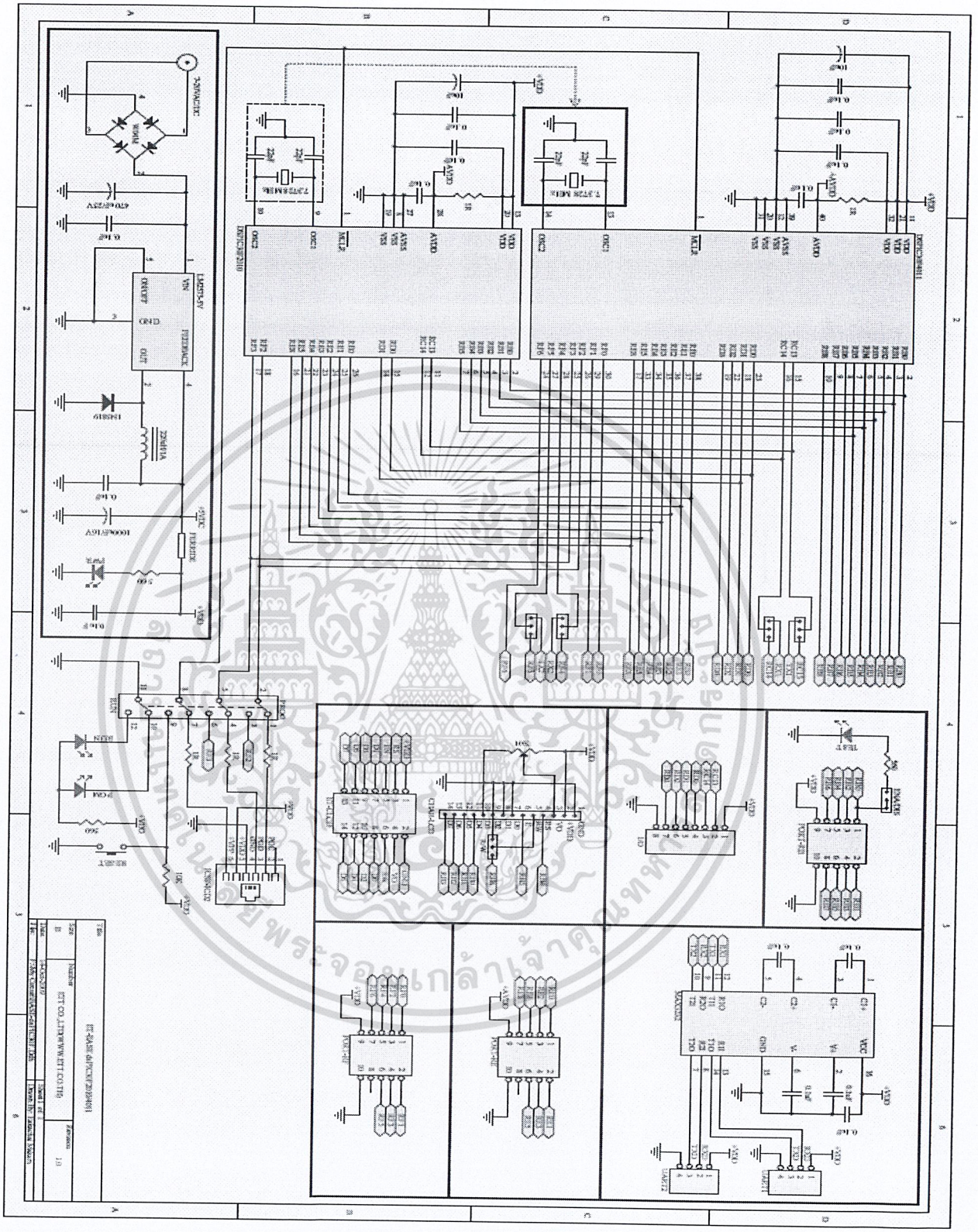
```
                i++;
                }
                while(BusyUART1());
                WriteUART1(Buf[i]);                    //ส่งตัวสุดท้าย 0xFF ออกไป
    }
}

//=================================================================
//----------------------:Main Program:----------------------
//=================================================================

int main(void)
{
        int i,j;

 Initial_4bitLCD();                                    // Initial LCD 4 Bit Interface
 Uart1_Init();                                         // Initialize the UART1 9600 bps

//กำหนด รีเลย์ตัวที่หนึ่งใช้ช่องพอร์ท F0,F1
 TRIS_Rly0 = 0;                                        // Set direction control RF0 Output
 Relay0 = 0;                                           // Clear RF0 กำหนดให้ตัวรีเลย์อยู่ในสภาวะ OFF ก่อน
 TRIS_Rly1 = 0;                                        // Set direction control RF1 Output
 Relay1 = 0;                                           // Clear RF0 กำหนดให้ตัวรีเลย์อยู่ในสภาวะ OFF ก่อน

  SetDDRamAddr(0x00);
  PutsLCD((unsigned char *)"  PROGRAM TEST  ");
  SetDDRamAddr(0x40);
  PutsLCD((unsigned char *)"  DS1820 TEMP   ");
  Delay(2000000);

_LATF1 = 1;  //กำหนดสถานะเริ่มต้นให้ รีเลย์ตัวที่0 เป็น On ก่อน
_LATF2 = 1;  //กำหนดสถานะเริ่มต้นให้ รีเลย์ตัวที่1 เป็น On ก่อน

/*
    while(BusyUART1());
            WriteUART1('C');
    while(BusyUART1());
            WriteUART1('D');
```

```
 */
for(;;) {



 j = rdtemp();
              if(j&0x01)
             {
                        i =5;
                        j >>= 1;
                        }
             else
             {
                        i = 0;
                        j >>= 1;
             }

 //สองบรรทัด
// sprintf(buf_Tempc,"Temp = +%d.%d %cC    ",j,i,0xDF);
// SetDDRamAddr(0x40);

 //บรรทัดเดียว
 sprintf(buf_Tempc,"T=+%d.%d%cC",j,i,0xDF);
 SetDDRamAddr(0x00);

          PutsLCD((unsigned char *)buf_Tempc);
          Delay(20000);


      //มีคำสั่งหรือข้อมูลเข้ามา
if (ch == CodeStop) {

                ch = 0;    //ลบตัวสุดท้ายที่รับไว้
      c_buf=0;   //ตั้งให้เริ่มต้นรับข้อมูลหรือคำสั่งใหม่

      //เก็บ My ตัวที่ส่งข้อมูลมาไว้เพื่อส่งกลับได้ถูกต้อง
          //addr_back = 0x30;  //กำหนดค่าเริ่มต้นทุกตัวเป็นการกลับไปสู่ตัวแม่เบอร์ 0
```

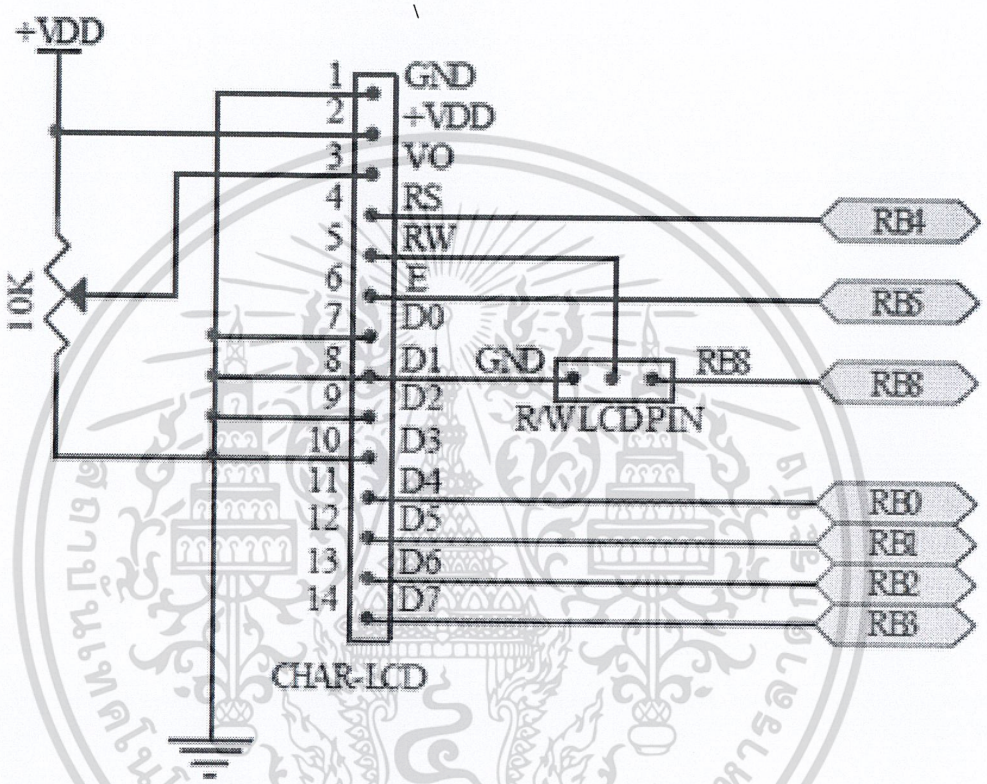
```
            addr_back = Buf[1]; //ตัวที่ศูนย์ คือ 0x30, ตัวที่หนึ่งคือ 0x31 อย่าลืม
            //ทำการตรวจสอบว่าเป็นการส่งต่อหรือส่งกลับ  ( 0x81 = ส่งต่อ (Forward), 0x82 = ส่งกลับ (Backward))


 Delay(80000);
   /*  Buf[0]=0x81;
     Buf[1]=0x30;
     Buf[2]=0x32;
     Buf[3]=0xEE;
     Buf[4]=0x00;
     Buf[5]=0x00;
     Buf[6]=0xFF;

 sprintf(buf_Tempc,"SetDATA ");
    SetDDRamAddr(0x00);

            PutsLCD((unsigned char *)buf_Tempc);
 */
            switch (Buf[0])
            {
             case 0x81 : Forward();
                    break;
             case 0x82 : Backward();
                         break;
             default   : break;
            }
 /*
             sprintf(buf_Tempc," Send OK  ");
    SetDDRamAddr(0x00);

            PutsLCD((unsigned char *)buf_Tempc);
  */
     }
            }

 return 0;
}
```

ภาคผนวก ข.

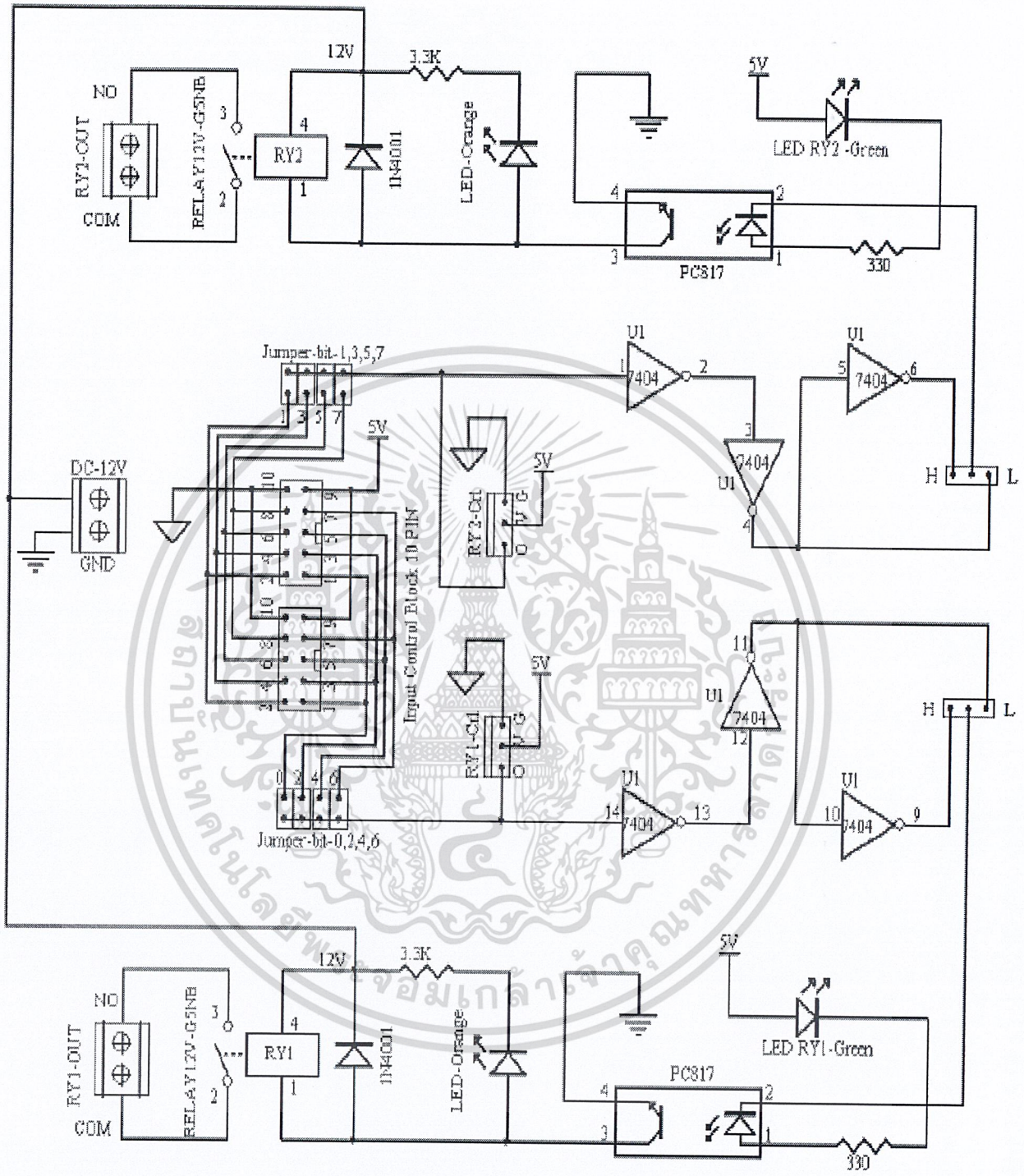
วงจรรวมของโครงงานทั้งหมด



รูปที่ ข.1 แสดงวงจร ET-BASE dsPIC30F4011



รูปที่ ข.2 วงจรอุปกรณ์หน้าจอ LED



รูปที่ ข.3 วงจรรีเลย์

ภาคผนวก ค.

DATA SHEET

1. XBee/XBee-PRO OEM RF Modules

XBee and XBee-PRO RF Modules were engineered to meet IEEE 802.15.4 standards and support the unique needs of low-cost, low-power wireless sensor networks. The modules require minimal power and provide reliable delivery of data between devices.

The modules operate within the ISM 2.4 GHz frequency band and are pin-for-pin compatible with each other.

1.1. Key Features

High Performance, Low Cost

XBee
- Indoor/Urban: up to 100' (30 m)
- Outdoor line-of-sight: up to 300' (100 m)
- Transmit Power: 1 mW (0 dBm)
- Receiver Sensitivity: -92 dBm

XBee-PRO
- Indoor/Urban: up to 300' (100 m)
- Outdoor line-of-sight: up to 1 mile (1500 m)
- Transmit Power: 100 mW (20 dBm) EIRP
- Receiver Sensitivity: -100 dBm

RF Data Rate: 250,000 bps

Advanced Networking & Security

Retries and Acknowledgements

DSSS (Direct Sequence Spread Spectrum)

Each direct sequence channels has over 65,000 unique network addresses available

Point-to-point, point-to-multipoint and peer-to-peer topologies supported

Coordinator/End Device operations supported

128-bit Encryption (downloadable firmware version coming soon)

Transparent and API operation

Low Power

XBee
- TX Current: 45 mA (@3.3 V)
- RX Current: 50 mA (@3.3 V)
- Power-down Current: < 10 µA

XBee-PRO
- TX Current: 215 mA (@3.3 V)
- RX Current: 55 mA (@3.3 V)
- Power-down Current: < 10 µA

Easy-to-use

No configuration necessary for out-of box RF communications

Free X-CTU Software (Testing and configuration software)

Small form factor

Network compatible with other 802.15.4 devices

AT and API Command Modes for configuring module parameters

Free & Unlimited Technical Support

1.1.1. Worldwide Acceptance

FCC Approval (USA) Refer to Appendix A [p47] for FCC Requirements. Systems that contain XBee/XBee-PRO RF Modules inherit MaxStream Certifications.

ISM (Industrial, Scientific & Medical) 2.4 GHz frequency band

Manufactured under **ISO 9001:2000** registered standards

XBee/XBee-PRO RF Modules are optimized for use in **US, Canada, Australia, Israel and Europe** (contact MaxStream for complete list of agency approvals).

1.2. Specifications

Table 1-01. Specifications of the XBee/XBee-PRO OEM RF Modules

Specification	XBee	XBee-PRO
Performance		
Range: Indoor/Urban (w/ 1.9 dB Whip or 2.1 dB Dipole antenna)	up to 100 ft. (30m)	up to 300' (100m)
Range: Outdoor RF line-of-sight (w/ 1.9 dB Whip or 2.1 dB Dipole antenna)	up to 300 ft. (100m)	up to 1 mile (1500m)
Transmit Power Output (software selectable)	1mW (0 dBm)	60 mW (18 dBm) conducted, 100 mW (20 dBm) EIRP*
RF Data Rate	250,000 bps	250,000 bps
Interface Data Rate (software selectable)	1200 - 115200 bps (non-standard baud rates also supported)	1200 - 115200 bps (non-standard baud rates also supported)
Receiver Sensitivity	-92 dBm (1% packet error rate)	-100 dBm (1% packet error rate)
Power Requirements		
Supply Voltage	2.8 – 3.4 V	2.8 – 3.4 V
Transmit Current (typical)	45mA (@ 3.3 V)	If PL=0 (10dBm): 137mA(@3.3V), 139mA(@3.0V) PL=1 (12dBm): 155mA (@3.3V), 153mA(@3.0V) PL=2 (14dBm): 170mA (@3.3V), 171mA(@3.0V) PL=3 (16dBm): 188mA (@3.3V), 195mA(@3.0V) PL=4 (18dBm): 215mA (@3.3V), 227mA(@3.0V)
Idle / Receive Current (typical)	50mA (@ 3.3 V)	55mA (@ 3.3 V)
Power-down Current (@ 3.0 V)	< 10 µA	< 10 µA
General		
Operating Frequency	ISM 2.4 GHz	ISM 2.4 GHz
Dimensions	0.960" x 1.087" (2.438cm x 2.761cm)	0.960" x 1.297" (2.438cm x 3.294cm)
Operating Temperature	-40 to 85° C (industrial)	-40 to 85° C (industrial)
Antenna Options	Integrated Whip, Chip or U.FL Connector	Integrated Whip, Chip or U.FL Connector
Networking & Security		
Supported Network Topologies	Point-to-Point, Point-to-Multipoint, Peer-to-Peer	Point-to-Point, Point-to-Multipoint, Peer-to-Peer
Number of Channels (software selectable)	16 Direct Sequence Channels	12 Direct Sequence Channels
Addressing Options	PAN ID, Channel and Source/Destination Addresses	PAN ID, Channel and Source/Destination Addresses
Agency Approvals		
United States (FCC Part 15.247)	OUR-XBEE	OUR-XBEEPRO
Industry Canada (IC)	4214A-XBEE	4214A-XBEEPRO
Europe (CE)	ETSI	ETSI (Max. 10 dBm transmit power output)*

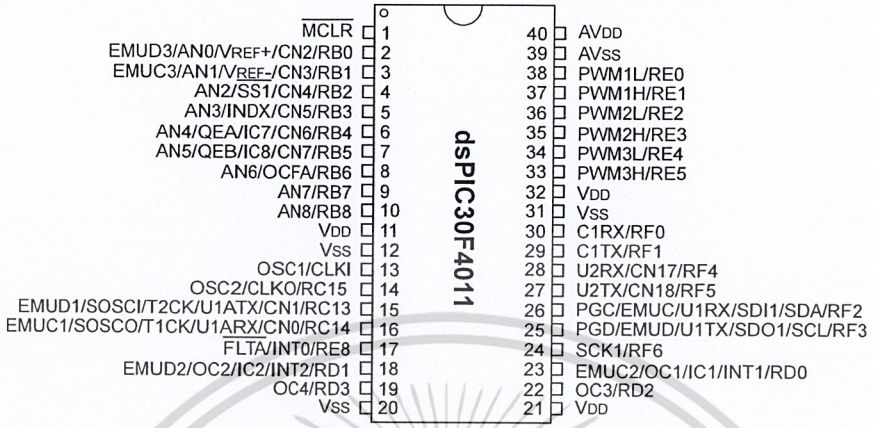
* When operating in Europe: XBee-PRO RF Modules must be configured to operate at a maximum transmit power output level of 10 dBm. The power output level is set using the PL command. The PL parameter must equal "0" (10 dBm).

Additionally, European regulations stipulate an EIRP power maximum of 12.86 dBm (19 mW) for the XBee-PRO and 12.11 dBm for the XBee when integrating high-gain antennas.

Regarding Antenna Options: The ranges specified are typical when using the integrated Whip (1.5 dBi) and Dipole (2.1 dBi) antennas. The Chip antenna option provides advantages in its form factor; however, it typically yields shorter range than the Whip and Dipole antenna options (especially outdoors). For more information, refer to the "XBee Antenna" application note located on MaxStream's web site (http://www.maxstream.net/support/knowledgebase/article.php?kb=153).

Pin Diagrams

40-Pin PDIP



44-Pin TQFP

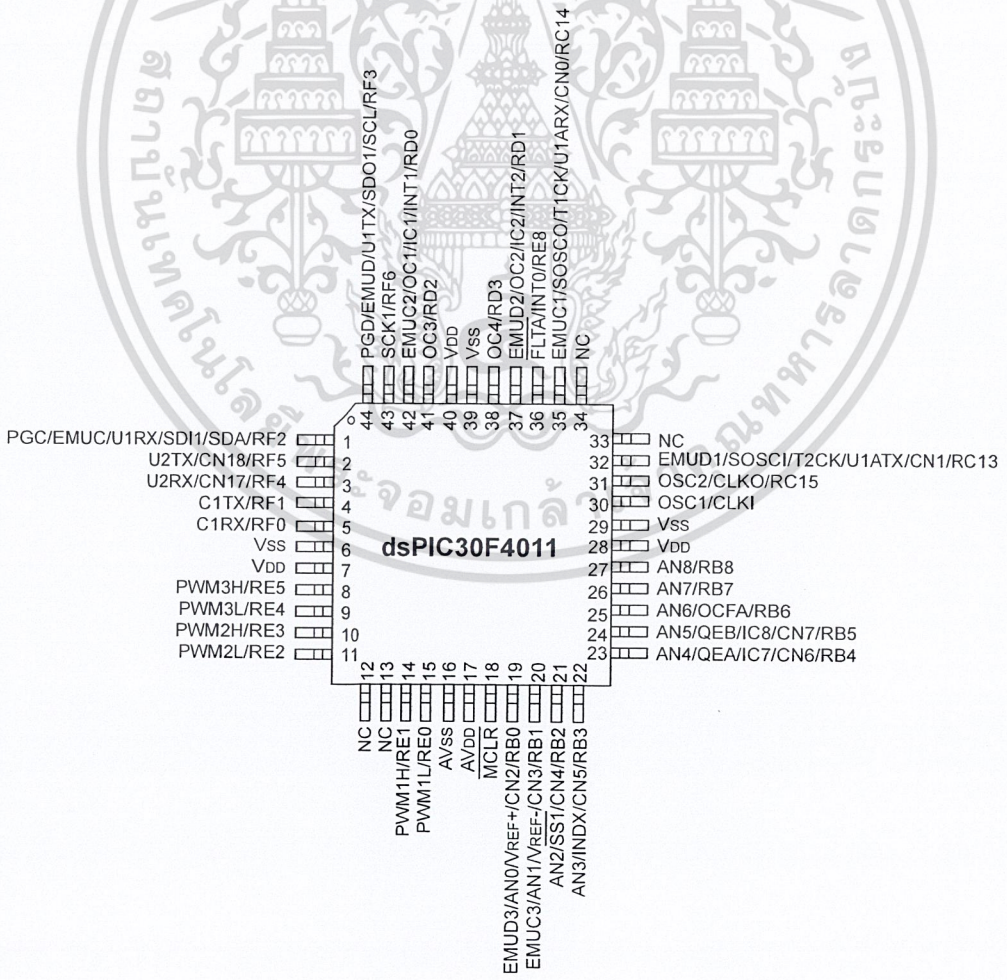


Table 1-1 provides a brief description of the device I/O pinout and the functions that are multiplexed to a port pin. Multiple functions may exist on one port pin. When multiplexing occurs, the peripheral module's functional requirements may force an override of the data direction of the port pin.

TABLE 1-1: dsPIC30F4011 I/O PIN DESCRIPTIONS

Pin Name	Pin Type	Buffer Type	Description
AN0-AN8	I	Analog	Analog input channels. AN0 and AN1 are also used for device programming data and clock inputs, respectively.
AVDD	P	P	Positive supply for analog module. This pin must be connected at all times.
AVSS	P	P	Ground reference for analog module.
CLKI CLKO	I O	ST/CMOS —	External clock source input. Always associated with OSC1 pin function. Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. Optionally functions as CLKO in RC and EC modes. Always associated with OSC2 pin function.
CN0-CN7 CN17-CN18	I	ST	Input change notification inputs. Can be software programmed for internal weak pull-ups on all inputs.
C1RX C1TX	I O	ST —	CAN1 bus receive pin. CAN1 bus transmit pin.
EMUD EMUC EMUD1 EMUC1 EMUD2 EMUC2 EMUD3 EMUC3	I/O I/O I/O I/O I/O I/O I/O I/O	ST ST ST ST ST ST ST ST	ICD Primary Communication Channel data input/output pin. ICD Primary Communication Channel clock input/output pin. ICD Secondary Communication Channel data input/output pin. ICD Secondary Communication Channel clock input/output pin. ICD Tertiary Communication Channel data input/output pin. ICD Tertiary Communication Channel clock input/output pin. ICD Quaternary Communication Channel data input/output pin. ICD Quaternary Communication Channel clock input/output pin.
IC1, IC2, IC7, IC8	I	ST	Capture inputs 1, 2, 7 and 8.
INDX QEA QEB	I I I	ST ST ST	Quadrature Encoder Index Pulse input. Quadrature Encoder Phase A input in QEI mode. Auxiliary Timer External Clock/Gate input in Timer mode. Quadrature Encoder Phase A input in QEI mode. Auxiliary Timer External Clock/Gate input in Timer mode.
INT0 INT1 INT2	I I I	ST ST ST	External interrupt 0. External interrupt 1. External interrupt 2.
$\overline{FLTA}$ PWM1L PWM1H PWM2L PWM2H PWM3L PWM3H	I O O O O O O	ST — — — — — —	PWM Fault A input. PWM1 low output. PWM1 high output. PWM2 low output. PWM2 high output. PWM3 low output. PWM3 high output.
$\overline{MCLR}$	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active-low Reset to the device.
OCFA OC1-OC4	I O	ST —	Compare Fault A input (for Compare channels 1, 2, 3 and 4). Compare outputs 1 through 4.

Legend: CMOS = CMOS compatible input or output; Analog = Analog input; ST = Schmitt Trigger input with CMOS levels; O = Output; I = Input; P = Power

TABLE 1-1: dsPIC30F4011 I/O PIN DESCRIPTIONS (CONTINUED)

Pin Name	Pin Type	Buffer Type	Description
OSC1	I	ST/CMOS	Oscillator crystal input. ST buffer when configured in RC mode; CMOS otherwise.
OSC2	I/O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. Optionally functions as CLKO in RC and EC modes.
PGD	I/O	ST	In-Circuit Serial Programming™ data input/output pin.
PGC	I	ST	In-Circuit Serial Programming clock input pin.
RB0-RB8	I/O	ST	PORTB is a bidirectional I/O port.
8RC13-RC15	8I/O	8ST	PORTC is a bidirectional I/O port.
RD0-RD3	I/O	ST	PORTD is a bidirectional I/O port.
RE0-RE5, RE8	I/O	ST	PORTE is a bidirectional I/O port.
RF0-RF6	I/O	ST	PORTF is a bidirectional I/O port.
SCK1	I/O	ST	Synchronous serial clock input/output for SPI1.
SDI1	I	ST	SPI1 data in.
SDO1	O	—	SPI1 data out.
$\overline{SS1}$	I	ST	SPI1 slave synchronization.
SCL	I/O	ST	Synchronous serial clock input/output for I^2C™.
SDA	I/O	ST	Synchronous serial data input/output for I^2C.
SOSCO	O	—	32 kHz low-power oscillator crystal output.
SOSCI	I	ST/CMOS	32 kHz low-power oscillator crystal input. ST buffer when configured in RC mode; CMOS otherwise.
T1CK	I	ST	Timer1 external clock input.
T2CK	I	ST	Timer2 external clock input.
U1RX	I	ST	UART1 receive.
U1TX	O	—	UART1 transmit.
U1ARX	I	ST	UART1 alternate receive.
U1ATX	O	—	UART1 alternate transmit.
U2RX	I	ST	UART2 receive.
U2TX	O	—	UART2 transmit.
VDD	P	—	Positive supply for logic and I/O pins.
VSS	P	—	Ground reference for logic and I/O pins.
VREF+	I	Analog	Analog voltage reference (high) input.
VREF-	I	Analog	Analog voltage reference (low) input.

Legend: CMOS = CMOS compatible input or output Analog = Analog input
ST = Schmitt Trigger input with CMOS levels O = Output
I = Input P = Power

19-5474; Rev 8/10

DALLAS SEMICONDUCTOR MAXIM

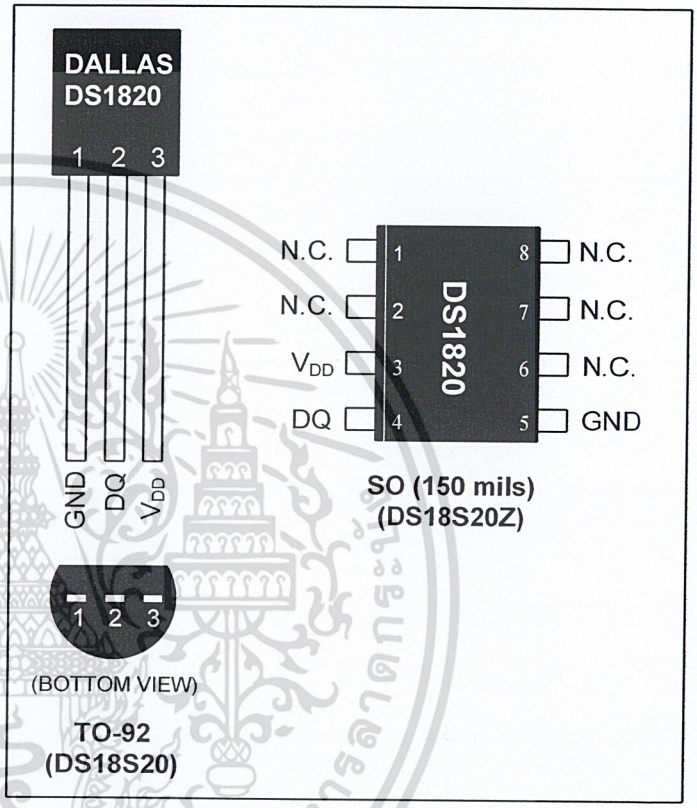
DS18S20
High-Precision 1-Wire Digital Thermometer

www.maxim-ic.com

FEATURES

- Unique 1-Wire® Interface Requires Only One Port Pin for Communication
- Each Device has a Unique 64-Bit Serial Code Stored in an On-Board ROM
- Multidrop Capability Simplifies Distributed Temperature Sensing Applications
- Requires No External Components
- Can Be Powered from Data Line. Power Supply Range is 3.0V to 5.5V
- Measures Temperatures from -55°C to +125°C (-67°F to +257°F)
- ±0.5°C Accuracy from -10°C to +85°C
- 9-Bit Thermometer Resolution
- Converts Temperature in 750ms (max)
- User-Definable Nonvolatile (NV) Alarm Settings
- Alarm Search Command Identifies and Addresses Devices Whose Temperature is Outside Programmed Limits (Temperature Alarm Condition)
- Applications Include Thermostatic Controls, Industrial Systems, Consumer Products, Thermometers, or Any Thermally Sensitive System

PIN CONFIGURATIONS



DESCRIPTION

The DS18S20 digital thermometer provides 9-bit Celsius temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18S20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. It has an operating temperature range of –55°C to +125°C and is accurate to ±0.5°C over the range of –10°C to +85°C. In addition, the DS18S20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply.

Each DS18S20 has a unique 64-bit serial code, which allows multiple DS18S20s to function on the same 1-Wire bus. Thus, it is simple to use one microprocessor to control many DS18S20s distributed over a large area. Applications that can benefit from this feature include HVAC environmental controls, temperature monitoring systems inside buildings, equipment, or machinery, and process monitoring and control systems.

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

ORDERING INFORMATION

PART	TEMP RANGE	PIN-PACKAGE
DS18S20	–55°C to +125°C	3 TO-92
DS18S20+	–55°C to +125°C	3 TO-92
DS18S20/T&R	–55°C to +125°C	3 TO-92 (2000 Piece)
DS18S20+T&R	–55°C to +125°C	3 TO-92 (2000 Piece)
DS18S20-SL/T&R	–55°C to +125°C	3 TO-92 (2000 Piece)*
DS18S20-SL+T&R	–55°C to +125°C	3 TO-92 (2000 Piece)*
DS18S20Z	–55°C to +125°C	8 SO
DS18S20Z+	–55°C to +125°C	8 SO
DS18S20Z/T&R	–55°C to +125°C	8 SO (2500 Piece)
DS18S20Z+T&R	–55°C to +125°C	8 SO (2500 Piece)

+Denotes a lead(Pb)-free/RoHS-compliant package. A "+" appears on the top mark of lead(Pb)-free packages.
T&R = Tape and reel.
**TO-92 packages in tape and reel can be ordered with straight or formed leads. Choose "SL" for straight leads. Bulk TO-92 orders are straight leads only.*

PIN DESCRIPTION

PIN		NAME	FUNCTION
TO-92	SO		
1	5	GND	Ground
2	4	DQ	Data Input/Output. Open-drain 1-Wire interface pin. Also provides power to the device when used in parasite power mode (see the *Powering the DS18S20* section.)
3	3	V_{DD}	Optional V_{DD}. V_{DD} must be grounded for operation in parasite power mode.
—	1, 2, 6, 7, 8	N.C.	No Connection

OVERVIEW

Figure 1 shows a block diagram of the DS18S20, and pin descriptions are given in the *Pin Description* table. The 64-bit ROM stores the device's unique serial code. The scratchpad memory contains the 2-byte temperature register that stores the digital output from the temperature sensor. In addition, the scratchpad provides access to the 1-byte upper and lower alarm trigger registers (T_H and T_L). The T_H and T_L registers are nonvolatile (EEPROM), so they will retain data when the device is powered down.

The DS18S20 uses Maxim's exclusive 1-Wire bus protocol that implements bus communication using one control signal. The control line requires a weak pullup resistor since all devices are linked to the bus via a 3-state or open-drain port (the DQ pin in the case of the DS18S20). In this bus system, the microprocessor (the master device) identifies and addresses devices on the bus using each device's unique 64-bit code. Because each device has a unique code, the number of devices that can be addressed on one bus is virtually unlimited. The 1-Wire bus protocol, including detailed explanations of the commands and "time slots," is covered in the *1-Wire Bus System* section.

Another feature of the DS18S20 is the ability to operate without an external power supply. Power is instead supplied through the 1-Wire pullup resistor via the DQ pin when the bus is high. The high bus signal also charges an internal capacitor (C_{PP}), which then supplies power to the device when the bus is low. This method of deriving power from the 1-Wire bus is referred to as "parasite power." As an alternative, the DS18S20 may also be powered by an external supply on V_{DD}.