ระบบเฝ้าระวังบ่อเลี้ยงสัตว์น้ำด้วยเซ็นเซอร์ไร้สายผ่านระบบ GSM

โดย

นางสาวพิมจันทร์ ชัยชนะ รหัสนักศึกษา B5012008

นางสาวพัชรียา พวงมาลัย รหัสนักศึกษา B5030309

รายงานนี้เป็นส่วนหนึ่งของการศึกษาวิชา 427499 โครงงานวิศวกรรมโทรคมนาคม และ

วิชา 427494 โครงงานศึกษาวิศวกรรมโทรคมนาคม

ประจำภาคการศึกษาที่ 1 และ 2 ปีการศึกษา 2553

หลักสูตรวิศวกรรมศาตรบัณฑิต สาขาวิชาวิศวกรรมโทรคมนาคม หลักสูตรปรับปรุง พ.ศ. 2545

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

ระบบเฝ้าระวังบ่อเลี้ยงสัตว์น้ำด้วยเซ็นเซอร์ไร้สายผ่านระบบ GSM

คณะกรรมการสอบโครงงาน

(ผู้ช่วยศาสตราจารย์ ดร. วิภาวี หัตถกรรม)
กรรมการ/อาจารย์ที่ปรึกษาโครงงาน

(ผู้ช่วยศาสตราจารย์ ดร. พีระพงษ์ อุฑารสกุล)
กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร. มนต์ทิพย์ภา อุฑารสกุล)
กรรมการ

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับรายงานโครงงานฉบับนี้ เป็นส่วนหนึ่งของการศึกษาระดับปริญญาตรี สาขาวิชาวิศวกรรมโทรคมนาคม วิชา 427499 โครงงานวิศวกรรมโทรคมนาคม และวิชา 427494 โครงงานศึกษาวิศวกรรมโทรคมนาคม ประจำปีการศึกษา 2553

โครงงาน	ระบบเฝ้าระวังบ่อเลี้ยงสัตว์น้ำด้วยเซ็นเซอร์ไร้สายผ่านระบบ GSM
จัดทำโดย	นางสาวพิมจันทร์ ชัยชนะ
	นางสาวพัชรียา พวงมาลัย
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร.วิภาวี หัตถกรรม
สาขาวิชา	วิศวกรรมโทรคมนาคม
ภาคการศึกษาที่	1/ 2553, 2/2553

บทคัดย่อ

(Abstract)

ปัจจุบันเทคโนโลยีเครือข่ายเซ็นเซอร์ไร้สาย ซึ่งเป็นเทคโนโลยีที่ได้รับความนิยมอย่างมาก ได้เข้ามามีบทบาทในการดำรงชีวิต และมีแนวโน้มนำเทคโนโลยีมาประยุกต์ใช้ทางด้านเกษตรกรรม ไม่ว่าจะเป็นการทำไร่ ทำสวนหรือเพาะเลี้ยงสัตว์น้ำ เพื่อช่วยเพิ่มประสิทธิภาพผลผลิตให้มีคุณภาพมากขึ้น ดังนั้นจึงได้นำเสนอโครงงาน ระบบเซ็นเซอร์ตรวจวัดคุณภาพของน้ำเพื่อไม่ให้เป็นอันตรายแก่สิ่งมีชีวิตในน้ำ ซึ่งระบบเซ็นเซอร์ทำการวัดค่า อุณหภูมิ(Temperature) ความเป็นกรด-ด่าง (pH) และออกซิเจนที่ละลายในน้ำ (DO) ซึ่งข้อมูลทั้งหมดถูกส่งผ่านเซ็นเซอร์ไร้สายมายังเครื่องรับปลายทางที่เชื่อมต่อกับสถานีฐานและแสดงผลผ่านทางคอมพิวเตอร์ที่จะทำหน้าที่ประมวลผล เพื่อแจ้งค่าที่วัดได้แก่ผู้ใช้งานผ่าน SMS โดยโครงงานนี้จะทำการติดตั้ง ทดสอบระบบเซ็นเซอร์ไร้สาย และ GUI เพื่อง่ายต่อการใช้งานระบบ

กิตติกรรมประกาศ

(Acknowledgement)

จากการจัดทำโครงงานเรื่องระบบเฝ้าระวังบ่อเลี้ยงสัตว์น้ำด้วยเซ็นเซอร์ไร้สายผ่านระบบ GSM ส่งผลให้คณะผู้จัดทำได้รับความรู้และประสบการณ์ต่างๆ มากมาย โครงงานฉบับนี้สามารถสำเร็จลุล่วงไปได้ด้วยดี เนื่องจากได้รับความกรุณาจากอาจารย์ที่ปรึกษาโครงงาน ผู้ช่วยศาสตราจารย์ ดร. วิภาวี หัตถกรรม ที่ได้ให้ความช่วยเหลือ ให้คำปรึกษาแนะนำในทุกๆ ด้าน ตลอดจนชี้แนะข้อบกพร่องต่างๆ รวมถึงการให้แนวคิด การดูแลเอาใจใส่ติดตามงาน ให้แก่คณะผู้จัดทำมาโดยตลอด และขอขอบพระคุณคณาจารย์ บุคลากร และพี่นักศึกษาบัณฑิตศึกษาสาขาวิชาวิศวกรรมโทรคมนาคมทุกท่านที่คอยแนะนำและให้ความช่วยเหลือ

คณะผู้จัดทำใคร่ขอขอบพระคุณทุกๆ ท่านที่ได้กล่าวไปแล้วไว้ ณ ที่นี้ ซึ่งเป็นผู้ให้โอกาสทางการศึกษาและคอยสนับสนุน สำหรับส่วนดีของโครงงานชิ้นนี้ ขออุทิศให้แก่อาจารย์ทุกท่านที่ได้ประสิทธิ์ประสาทวิชาความรู้ให้แก่คณะผู้จัดทำ

นางสาวพิมจันทร์ ชัยชนะ

นางสาวพัชรียา พวงมาลัย

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญรูป

สารบัญรูป(ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

เนื่องจาก ในปัจจุบันได้มีการแข่งขันกันทางด้านคุณภาพของสินค้ามากมาย รวมทั้งทางด้านเกษตรกรรมด้วยเช่นกัน โดยเฉพาะทางด้านการเพาะเลี้ยงสัตว์น้ำ อาทิเช่น กุ้ง ปลาฯ ซึ่งเกษตรกรชาวประมงจำเป็นต้องพัฒนาผลิตภัณฑ์สัตว์น้ำให้มีคุณภาพและมีมาตรฐาน เพื่อสร้างความน่าเชื่อถือให้กับผู้บริโภค คุณสมบัติของน้ำที่เหมาะสมในการเพาะเลี้ยงก็เป็นส่วนสำคัญในการพัฒนาคุณภาพของสัตว์น้ำให้มีมูลค่าเพิ่มทางเศรษฐกิจ ปัจจัยสำคัญที่ทำให้น้ำมีคุณสมบัติที่เหมาะสมแก่การเพาะเลี้ยงสัตว์น้ำ คือ อุณหภูมิ(temperature) ความเป็นกรด-ด่าง(pH) และปริมาณออกซิเจนที่ละลายในน้ำ (Dissolved oxygen : DO) ซึ่งปัจจัยทั้ง3นี้มีผลต่อการเจริญเติบโต การสืบพันธุ์ และกิจกรรมต่างๆในการดำเนินชีวิตของสัตว์น้ำ เพราะต้องมีค่าคงที่อยู่ใช่วงที่เหมาะสมสำหรับการดำเนินชีวิตของสัตว์น้ำ หากเกิดการเปลี่ยนแปลงจะส่งผลโดยตรงต่อการเจริญเติบโตและขยายพันธุ์ของสัตว์น้ำ จึงควรมีการระมัดระวังป้องกันอย่างใกล้ชิด แต่เนื่องจากความเป็นกรด-ด่าง(pH) และปริมาณออกซิเจนที่ละลายในน้ำ(Dissolved oxygen : DO) จำเป็นต้องวัดอย่างสม่ำเสมอ ทำให้เสียเวลา ค่าใช้จ่ายและแรงงานที่ต้องทำการวัดค่าอยู่บ่อยๆ เกิดความยุ่งยากและแก้ไขไม่ทันท่วงที หากเกิดสภาวะผิดปกติขึ้นในแหล่งน้ำ ฉะนั้น **โครงงานระบบเฝ้าระวังบ่อเลี้ยงสัตว์น้ำด้วยเซ็นเซอร์ไร้สายผ่านระบบ GSM** จึงได้เล็งเห็นว่าจะใช้ระบบการส่ง SMS มาช่วยในการแก้ปัญหาเนื่องจากว่า ผู้ประกอบการก็มีโทรศัพท์ประจำอยู่แล้ว เราก็ให้โครงงานชิ้นนี้ส่ง SMS เข้าไปยังโทรศัพท์มือถือ ของผู้ประกอบการ เมื่อเซ็นเซอร์ทำการวัดค่า ก็จะส่งข้อมูลมาเก็บไว้ที่สถานีฐาน แล้วทำการส่ง SMS ตามเวลาที่กำหนดเช่น ให้ส่งทุกๆ สามชั่วโมง และมี GUI (Graphic User Interface) เพื่อให้ผู้ประกอบการสามารถตรวจสอบข้อมูลได้สะดวก ซึ่งจากการแก้ปัญหาโดยใช้การส่ง SMS จะเห็นได้ว่าผู้ประกอบการสามารถรับรู้ข้อมูลได้ตลอดแม้จะอยู่สถานที่ไหนก็ตาม ซึ่งโครงงานนี้ได้ใช้ MODULE GSM CPU เป็นตัวรับคำสั่งและทำการส่ง SMS โดยการส่ง SMS นี้ได้ใช้ระบบ truemove GSM ในการส่งเป็นหลักเนื่องจากเป็นการประหยัดต้นทุน เพราะค่าใช้จ่ายในการรับส่ง SMS ค่อนข้างถูกและระบบการส่ง SMS ก็เป็นที่นิยมแพร่หลายในปัจจุบัน

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อเป็นการศึกษาในการใช้โปรแกรม Microsoft Visual C++ 6.0
2. เพื่อเป็นการศึกษาและพัฒนาฝีมือในการใช้โปรแกรม MATLAB R2009a
3. นำเอาระบบการส่ง SMS ไปช่วยในการแจ้งค่า อุณหภูมิ(temperature) ค่าความเป็นกรด-ด่าง (pH)และออกซิเจนที่ละลายในน้ำ (DO) ในบ่อเลี้ยงสัตว์น้ำ แทนการจ้างแรงงาน
4. เพื่อติดตั้งและทดสอบระบบเซ็นเซอร์ ให้สามารถตรวจจับอุณหภูมิ(temperature) ค่าความเป็นกรด-ด่าง (pH) และออกซิเจนที่ละลายในน้ำ (DO) ได้

1.3 ขอบเขตการดำเนินงาน

1. ศึกษาการทำงานของโปรแกรม Microsoft Visual C++ 6.0 และ Matlab R2009a
2. ศึกษาการทำงานของ Module Wireless CPU รุ่น Q26 (GSM)
3. เขียนโปแกรมควบคุมการทำงานของ Module Wireless CPU รุ่น Q26 (GSM) เพื่อส่ง SMS โดยใช้ Microsoft Visual C++ 6.0
4. เขียนโปรแกรมสร้าง GUI (Graphic user interface) โดยใช้ Matlab R2009a
5. จัดเตรียมอุปกรณ์สำหรับติดตั้ง
6. ทดสอบอุปกรณ์ทั้งระบบให้ได้ตามวัตถุประสงค์
7. นำไปติดตั้งยังสถานที่จริง

1.4 ขั้นตอนการดำเนินงาน

1. ศึกษาโปรแกรม Microsoft Visual C++ 6.0 และ Matlab 2009
2. ศึกษาการทำงานของ Module Wireless CPU รุ่น Q26 (GSM)
3. เขียนโปรแกรมให้สามารถส่ง SMS ได้ตามเงื่อนไขที่กำหนด
4. เขียนโปรแกรมสร้าง GUI (Graphic user interface) เพื่อง่ายต่อการตรวจสอบข้อมูล
5. ทดสอบการทำงานโปรแกรมส่ง SMS และ GUI (Graphic user interface)
6. จัดทำอุปกรณ์ทั้งหมด
7. ทดสอบการรับส่งข้อมูลระหว่าง เซ็นเซอร์กับสถานีฐาน
8. นำอุปกรณ์ทั้งหมดไปติดตั้งและทดสอบเพื่อให้ได้ตามวัตถุประสงค์
9. สรุปผลการทดลอง เขียนรายงาน และนำเสนอผลงาน

บทที่ 2
ทฤษฎีพื้นฐานที่เกี่ยวข้อง

2.1 บทนำ

ในบทนี้จะกล่าวถึงทฤษฎีพื้นฐานที่เกี่ยวข้องกับระบบเฝ้าระวังบ่อเลี้ยงสัตว์น้ำด้วยเซ็นเซอร์ไร้สายผ่านระบบ GSM ซึ่งเนื้อหาจะเกี่ยวกับระบบGSM รวมไปถึงเครือข่ายเซ็นเซอร์ไร้สาย และพื้นฐานการใช้งานเซ็นเซอร์ pH, DO, Temperature

2.2 เครือข่ายเซ็นเซอร์ไร้สาย (Wireless Sensor Networks)

ระบบเครือข่ายเซ็นเซอร์แบบไร้สาย (Wireless Sensor Networks หรือ WSNs) คือการใช้อุปกรณ์เซ็นเซอร์เล็กๆ จำนวนมากเพื่อตรวจวัดคุณสมบัติต่างๆ ของสิ่งแวดล้อมที่เราสนใจและประมวลผลข้อมูลเหล่านั้นเพื่อสร้างองค์ความรู้ใหม่เกี่ยวกับสิ่งแวดล้อมรอบตัวเราหรือตอบสนองกับการเปลี่ยนแปลงของสภาพแวดล้อมได้โดยอัตโนมัติ ซึ่งอุปกรณ์พื้นฐานของ WSN คือ Mote เป็นคอมพิวเตอร์ขนาดเล็กสำหรับวัดอุณหภูมิความชื้นหรือสภาวะแวดล้อมอื่นๆ ทำงานได้โดยใช้แบตเตอรี่ธรรมดาและสื่อสารกับ Mote ที่อยู่ใกล้เคียงโดยใช้ลักษณะการส่งข้อมูลระบบในเครือข่ายไร้สาย ซึ่งข้อมูลจะถูกส่งผ่านระหว่าง Mote จนกระทั่งถึงจุดหมายปลายทางซึ่งอาจจะเป็นคอมพิวเตอร์หรืออุปกรณ์อื่นๆ สำหรับรวบรวมข้อมูลเพื่อวัดได้

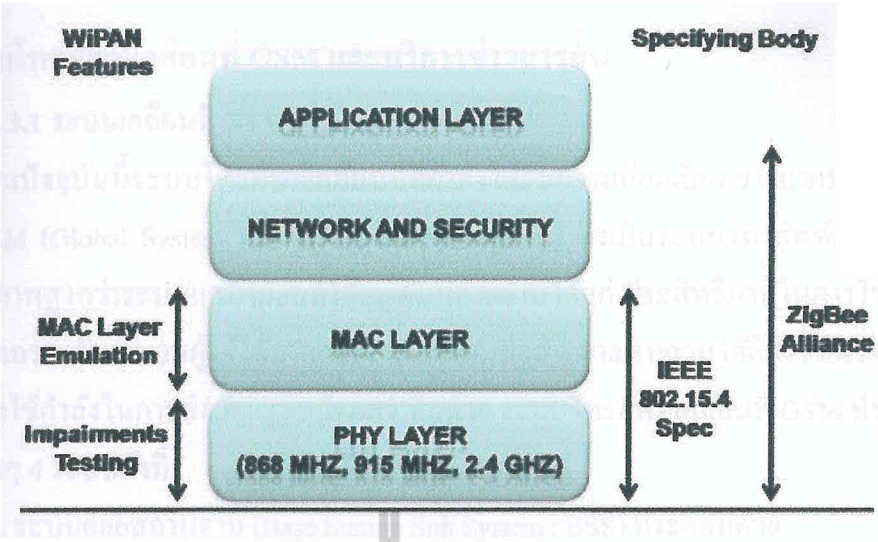
ตัวอย่างการใช้งานของเครือข่ายตรวจรู้ไร้สายนี้ ได้แก่การฝังอุปกรณ์เซนเซอร์ไว้ในรังนกหายากบางชนิดเพื่อตรวจจับการเปลี่ยนแปลงอุณหภูมิที่มีผลต่อการย้ายถิ่นฐานของนกเหล่านั้น การติดตั้งเซนเซอร์ไว้ในอุปกรณ์ผสมสารเคมีขนาดใหญ่หรือท่อส่งเคมีในโรงงานอุตสาหกรรมเพื่อตรวจจับการรั่วซึมของสารเคมี การใช้เซนเซอร์ตรวจวัดการสั่นไหวของอุปกรณ์หลายๆ อย่างที่ใช้สำหรับการสร้างคอมพิวเตอร์ชิพ เพื่อตรวจจับความผิดปกติของเครื่องมือเหล่านั้นเพื่อ ให้สามารถเข้าไปดูแลได้ก่อนที่มันจะเสียหาย การติดตั้งเซนเซอร์ไว้รอบ ๆ สนามบินเพื่อตรวจจับการบุกรุกในบริเวณที่ห้ามคนเข้า เป็นต้น จากตัวอย่างเหล่านี้จะเห็นได้ว่านักวิทยาศาสตร์และวิศวกรสามารถใช้ประโยชน์จากเครือข่ายตรวจรู้ไร้สายได้มากมายหลายรูปแบบ นอกจากนี้การออกแบบเซ็นเซอร์โหนดให้มีขนาดที่เล็กและใช้พลังงานน้อยทำให้สามารถติดตั้งได้ในสภาพแวดล้อมที่หลากหลาย เทคโนโลยีเครือข่ายเซ็นเซอร์จึงได้ถูกคาดการณ์ว่าจะเป็นเทคโนโลยีหลักในการขับดันสู่ยุคของคอมพิวเตอร์ทุกแห่งหน(Ubiquitous computing, pervasive computing) ด้วยการสร้างสภาพแวดล้อมประดิษฐ์ในรอบๆตัวของเราทุกคน

อย่างไรอย่างไรก็ตามการทำให้ WSN ทำงานได้จริงอย่างมีประสิทธิภาพนั้น มีอุปสรรคหลายอย่างเนื่องจาก mote มีขีดจำกัดในความเร็วของ processor, ความจุของ data storage, และ bandwidth ในการสื่อสาร ดังนั้น อายุการใช้งานของมันจึงขึ้นอยู่กับการบริหารการใช้พลังงานของมันเองซึ่ง มีอยู่จำกัดด้วยเหตุนี้ผู้พัฒนา mote จึงต้องออกแบบระบบ hardware และ software รวมถึงระบบการสื่อสารของ mote ให้ทำงานโดยใช้พลังงานน้อยที่สุด นอกจากนี้ในแง่ของการใช้งานผู้พัฒนา WSN ต้องสร้างเครื่องมือที่ทำผู้ใช้ซึ่งไม่จำเป็นจะต้องมีความรู้ขั้นสูงทางด้าน computer engineering สามารถใช้งานและสร้าง WSN applications โดยง่ายด้วย

2.2.1 ZigBee

ZigBee เป็นการสื่อสารที่ออกแบบขึ้นสำหรับการสื่อสารในเครือข่ายเซ็นเซอร์แบบไร้สาย (Wireless Sensor Network) โดยเริ่มจากการกำหนดมาตรฐานการรับ-ส่งข้อมูลแบบ IEEE 802.15.4 ที่เน้นการสื่อสารแบบประหยัดพลังงาน ความเร็วการรับส่งข้อมูลต่ำและมีราคาถูก การสื่อสารลักษณะนี้ได้ถูกนำมาใช้สำหรับการสื่อสาร ระหว่างเครื่องตรวจวัดหรือเซ็นเซอร์ ที่ต้องการสื่อสารแบบไร้สาย เพื่อลดความยุ่งยากซับซ้อนสำหรับการติดตั้ง เช่น บริเวณโรงงานหนึ่งๆ อาจจะต้องใช้จำนวนเซ็นเซอร์ปริมาณมากๆ และเครื่องรับส่งที่มีราคาถูก และประหยัดพลังงาน

มาตรฐาน IEEE 802.15.4 กำหนดขึ้นสำหรับการรับส่งข้อมูลเบื้องต้นในวงจรเครื่องรับส่งวิทยุ (Physical Layer) และการควบคุมการรับส่ง (Link Layer) ดังต่อไปนี้ การสื่อสารใช้คลื่นวิทยุความถี่ 2.5 กิกะเฮิรตซ์ (GHz) แบ่งออกเป็น 16 ช่องสัญญาณๆ ละ 5 เมกะเฮิรตซ์ (MHz) สำหรับความถี่ 900 เมกะเฮิรตซ์(MHz) แบ่งออกเป็น 10 ช่องสัญญาณๆ ละ 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 ควบคุมการรับส่งข้อมูลโดยใช้โปรโตคอลแบบ Carrier Sense Multiple Access/ Collision Avoidance (CSMA/CA) และเพื่อให้การสื่อสารเครือข่ายเซ็นเซอร์ไร้สายเป็นมาตรฐานเดียวกัน จึงกำหนดมาตรฐานเพิ่มสำหรับการเชื่อมต่อเป็นเครือข่าย (Network Layer) และการนำไปใช้งาน (Application Layer) ร่วมกับมาตรฐาน IEEE 802.15.4 เป็นมาตรฐานใหม่ที่กำหนดโดยองค์กร ZigBee Alliance



รูป 2.1 แสดง Protocol Zigbee

Zigbee ได้แบ่งตามลักษณะการทำงาน 3 แบบ คือ

1. Coordinator มีหน้าที่สร้างการสื่อสารเชื่อมโยงเครือข่ายระหว่าง End device กับ Router หรือ Coordinator กับ Coordinator ด้วยกัน หรือ Coordinator กับ Router กำหนด address ให้กับ Device ที่อยู่ในวงเครือข่าย ไม่ให้ซ้ำกัน ดูแลจัดการเรื่องการ Routing เส้นทาง ซึ่งเทียบได้กับ FFD

2. End-device เป็นอุปกรณ์ปลายทางสุด ซึ่งจะใช้รับสัญญาณจาก Sensor ที่ปลายทาง โดยที่ใช้พลังงานต่ำในการทำงานเทียบได้กับ RFD หรือ FFD บางกรณี ขึ้นอยู่กับ sensor ที่ใช้

3. Router มีหน้าที่รับส่งข้อมูลในเส้นทางต่าง ๆของเครือข่าย ซึ่งเทียบได้กับ FFD

2.2.2 ขั้นตอนการทำงานของโพรโทคอล ZigBee

1.) ขั้นตอนการทำงานของ ZigBee coordinator

ZigBee coordinator จะเริ่มต้นเครือข่าย โดยการตรวจสอบการใช้ช่องสัญญาณวิทยุภายในบริเวณรอบๆ ถ้ามีช่องสัญญาณที่ไม่ถูกใช้โดย coordinator ตัวอื่น ก็สามารถเริ่มต้นเครือข่ายได้ หลังจากนั้น coordinator ก็จะทำหน้าที่เป็นศูนย์กลางของเครือข่าย รองรับการเข้าร่วมเครือข่ายของ ZigBee end-device และรองรับการร้องขออื่นๆ ตามมาตรฐานด้วยเช่นกัน

2.) ขั้นตอนการทำงานของ ZigBee end-device

ZigBee end-device จะเริ่มต้นการทำงานโดยการร้องขอการเข้าร่วมเครือข่ายไปยัง coordinator ประจำเครือข่ายนั้นๆ โดยการตรวจสอบผ่านช่องสัญญาณต่างๆ ว่า coordinator ใช้ช่องสัญญาณใดอยู่เมื่อเข้าร่วมเครือข่ายแล้ว end-device จึงสามารถทำการร้องขอคำสั่งอื่นๆ ผ่านทาง coordinator ได้ เช่น การส่งข้อความทั่วไป (Message), การร้องขอการ Binding (Binding request), การขอออกจากเครือข่าย

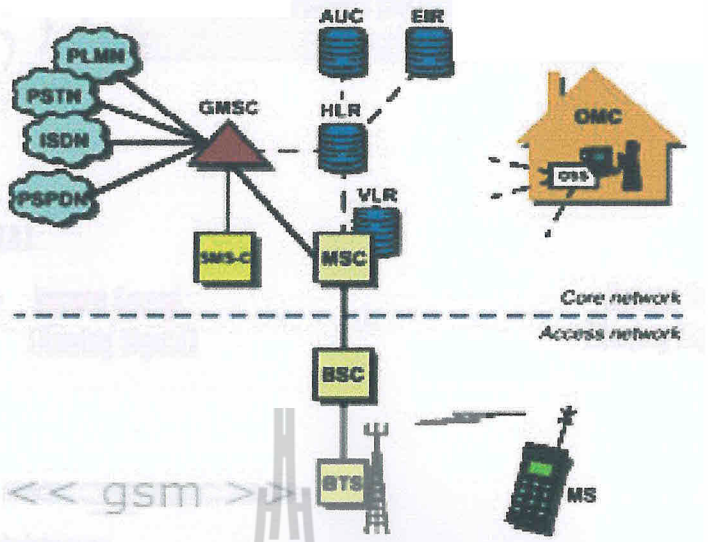
2.3 ระบบโทรศัพท์เคลื่อนที่ GSM และบริการข่าวสารสั้น

2.3.1 ระบบเคลื่อนที่ GSM

ในปัจจุบันนี้ระบบโทรศัพท์เคลื่อนที่ที่กำลังได้รับความนิยมเป็นอย่างมากระบบหนึ่งก็คือระบบ GSM (Global System for Mobile communication) ซึ่งเป็นระบบโทรศัพท์แบบดิจิทัล ที่มีประสิทธิภาพสูงกว่าระบบแอนาลอกที่ใช้อยู่เดิมหลายด้าน ได้แก่ ประสิทธิภาพในการใช้สเปกตรัม โดยสามารถรองรับจำนวนผู้ใช้ได้มากกว่า สามารถทนต่อสัญญาณรบกวนได้ดีกว่า มีความปลอดภัยสูง และยังใช้กำลังในการส่งสัญญาณน้อยกว่าอีกด้วย ระบบโทรศัพท์เคลื่อนที่ GSM ประกอบด้วยระบบย่อยๆ 4 ระบบดังนี้

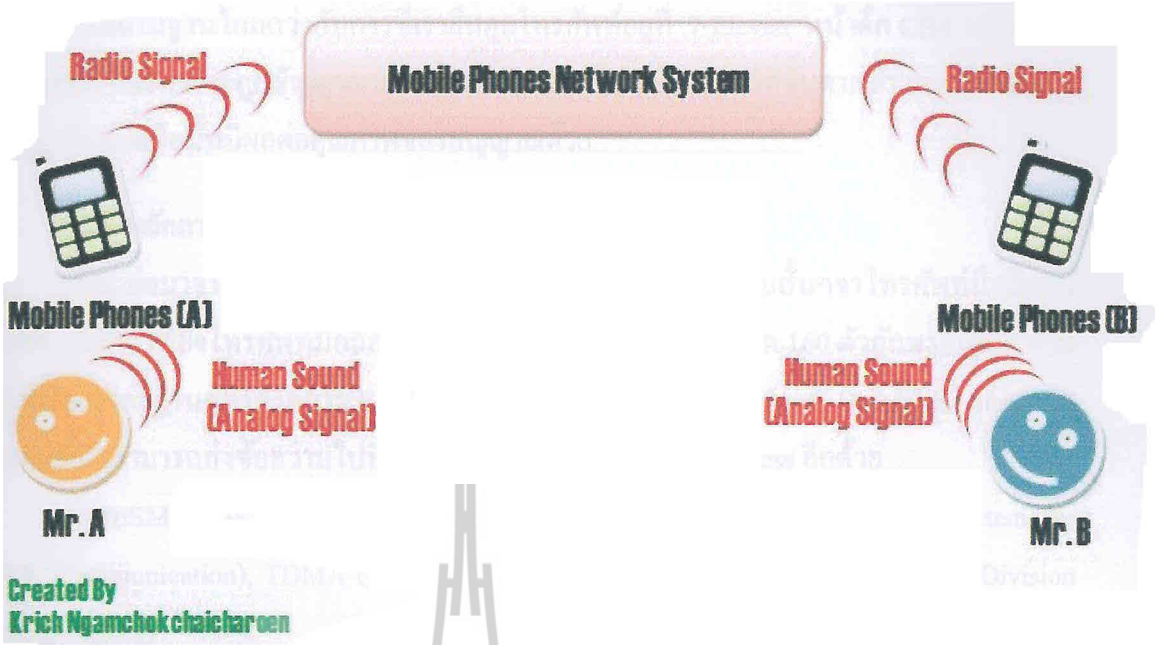
1. ระบบย่อยสถานีฐาน (Base Station Sub System : BSS) ประกอบด้วย
   - สถานีฐานรับ-ส่งสัญญาณ (Base Transceiver station: BTS)
   - ตัวควบคุมสถานีฐาน (Base Station Controller: BSC)
2. ระบบย่อยสวิตชิง (Switching Sub System: SSS) ประกอบด้วย
   - ชุมสายโทรศัพท์เคลื่อนที่ (Mobile Switching Center: MSC)
   - ฐานข้อมูลทะเบียนผู้ใช้ (Home Location Register: HLR)
   - ฐานข้อมูลผู้มาเยือน (Visitor Location Register: VLR)
   - ฐานข้อมูลตรวจสอบความถูกต้องของผู้ใช้ (Authentication Center: AC)
   - ฐานข้อมูลเครื่องโทรศัพท์มือถือ (Equipment Identity Register: EIR)
   - ศูนย์บริการเสริม เช่น ศูนย์บริการส่งข่าวสารสั้น (Short Message Service Center: SMS-C) เป็นต้น
3. ศูนย์ปฏิบัติการและบำรุงรักษา (Operation and Maintenance Center: OMC)
   - ศูนย์ปฏิบัติการและบำรุงรักษาสำหรับระบบย่อยสถานีฐาน (Operation and Maintenance Center for Base Station Sub System: OMC-B)
4. สถานีเคลื่อนที่หรือโทรศัพท์เคลื่อนที่ (Mobile Station: MS)

2.3.2 การใช้โทรศัพท์ในการติดต่อสื่อสาร (Mobile Phone for Human Communications)



รูป 2.2 แสดง รูปแบบการติดต่อสื่อสารแบบ GSM

การติดต่อสื่อสารของโทรศัพท์มือถือนั้น สามารถติดต่อกันได้ทั้งระหว่าง PLMN (Public Land Mobile Network) กับ PSTN (Public Switched Telephone Network) โดยมี Gateway MSC (GMSC) เป็นอุปกรณ์ในการติดต่อประสาน หรือติดต่อภายในระบบเครือข่าย PLMN ด้วยกันเอง ดังนั้น ถ้านาย A ทำการโทรเข้าโทรศัพท์มือถือของนาย B โดยใช้โทรศัพท์ที่บ้านของเค้าเอง เมื่อนาย A เริ่มการโทรออก ข้อมูลจากโทรศัพท์ต้นทางจะถูกนำไปใช้ในการหาที่อยู่ ณ ขณะนั้นของโทรศัพท์ปลายทาง โดยส่งไปยัง PSTN เพื่อค้นหาเส้นทางในการเชื่อมต่อเครือข่ายปลายทาง (GMSC) และไปที่ HLR (Home Location Register) ส่วนที่เก็บข้อมูล รายละเอียด และเครือข่าย SIM ของเครื่องเอาไว้ และเช็คต่อไปยัง VLR (Visitors Location Register) ในแต่ละเครือข่ายว่าอยู่ใน MSC ไหน (VLR จะมีกระบวนการ Location Update เพื่อบันทึกเครื่องที่เข้ามาภายใน Location Area ของชุมสายนั้นๆ) โดยใช้ IMSI (International Mobile Subscriber Identity) รหัสเฉพาะเครื่องที่ได้มาจาก HLR เปรียบเทียบ หากพบว่าอยู่ที่ MSC ไหน VLR จะส่ง MSRN (Mobile Station Roaming Number) เลขหมายชั่วคราวที่กำหนดขึ้นมาเป็นเส้นทางให้กับ GMSC ใช้ในการ route ไปยัง MSC ปลายทาง เมื่อ route เส้นทางได้ก็จะทำการ paging แจ้งไปยังโทรศัพท์เครื่องนั้นๆ สัญญาณ paging จะถูกส่ง ออกจาก BS (Base Station) ที่ควบคุมบริเวณ Location Area นั้นอยู่ และเมื่อโทรศัพท์ปลายทางรับสายตอบรับกลับมาการเชื่อมต่อก็จะเสร็จสมบูรณ์ ทำให้นาย A สามารถติดต่อสนทนากับนาย B ได้



รูป 2.3 แสดง การสนทนาของ A และ B

เนื่องด้วยเสียงของมนุษย์ถือเป็นสัญญาณแบบอนาล็อค (Analog Signal) เมื่อเราใช้เสียงของเราพูดผ่านโทรศัพท์มือถือ เสียงของเราจะถูกแปลงผสมสัญญาณเสียงกับคลื่นพาหะ (Modulate) เพื่อลดการถูกรบกวนจากสัญญาณอื่น และขยายสัญญาณด้วยเครื่องส่ง เพื่อส่งออกอากาศด้วยคลื่นวิทยุไปให้สถานีฐาน (BS) จากนั้นสัญญาณวิทยุจะถูกส่งจากเครือข่ายไปสู่บุคคลปลายทางที่เราได้ทำการติดต่อ คลื่นวิทยุจะถูกแปลงกลับมาเป็นสัญญาณเสียงอีกครั้งหนึ่งด้วยเครื่องรับภายในโทรศัพท์ปลายสาย เพื่อให้อีกบุคคลหนึ่งสามารถรับฟังและเกิดความเข้าใจได้

ขอบเขตในการติดต่อสื่อสาร

การติดต่อของโทรศัพท์มือถือกับสถานีฐานต้องอยู่ในขอบเขต หรือพื้นที่สัญญาณที่สถานีฐานส่งไปถึง ซึ่งขอบเขตดังกล่าวมีปัจจัยร่วมด้วยกัน 2 ปัจจัย คือ

1. กำลังส่งของโทรศัพท์มือถือ โทรศัพท์มือถือที่มีกำลังส่งต่างกันจะมีผลทำให้ขอบเขตของการติดต่อกับสถานีฐาน (BS) ต่างกันด้วย โดยที่กำลังส่งของโทรศัพท์มือถือจะแปรผันไปตามระยะห่างจากสถานีฐาน ถ้าสถานีฐานวัดระดับสัญญาณที่ได้รับจากโทรศัพท์มือถือแล้วไม่อยู่ในระดับที่กำหนด สถานีฐานจะส่งสัญญาณไปยังส่วนควบคุมของโทรศัพท์มือถือ เพื่อให้เพิ่มหรือลดกำลังของโทรศัพท์มือถือ

2. ตำแหน่งของโทรศัพท์มือถือ ตำแหน่งที่เราคุยโทรศัพท์มือถือจะมีผลต่อขอบเขตในการติดต่อกับสถานีฐาน (BS) เช่น ถ้าเรายืนคุยโทรศัพท์อยู่ที่ชั้นดาดฟ้าของตึก CB4 จะมีขอบเขตในการ

ติดต่อไปยังสถานีฐานไกลกว่ากับการที่เรายืนคุยโทรศัพท์อยู่ที่ 7-Eleven หน้าตึก CB4 แต่การคุยโทรศัพท์ที่ 7-Eleven จะถูกสัญญาณรบกวนน้อยกว่าการไปคุยโทรศัพท์ที่ชั้นดาดฟ้า เพราะตำแหน่งของโทรศัพท์มือถือนั้นมีผลต่อคุณภาพของสัญญาณด้วย

2.3.3 หลักการรับส่ง SMS ของโทรศัพท์มือถือ

SMS ย่อมาจาก Shor/ Message Service เป็นบริการส่งข้อความสั้นๆจาโทรศัพท์มือถือต้นทางผ่านชุมสายไปยังโทรศัพท์มือถือปลายทาง โดยสามารถส่งได้สูงสุด 160 ตัวอักษรต่อครั้ง ตามข้อกำหนดมาตรฐานขององค์การ ETSI (European Telecommunications Standards Institute) นอกจากนี้ยังสามารถส่งข้อความไปที่เครื่อง Fax, PC หรือ Internet address อีกด้วย

ระบบSMSในระบบเครือข่ายโทรศัพท์มือถือรองรับโดยระบบGSM (Global System for Mobile Communication), TDMA (Time Division Multiple Access) และ CDMA (Code Division Multiple Access)

เมื่อ SMS ถูกส่งจากโทรศัพท์มือถือเครื่องหนึ่ง ข้อความนั้นจะถูกส่งไปที่ Short Message Service Center (SMSC) จากนั้นจึงจะส่งไปยังโทรศัพท์มือถือเครื่องรับอีกทอดหนึ่ง โดยมีกระบวนการดังนี้

1. SMSC จะส่ง SMS Request ไปยัง Home Location Register (HLR) เพื่อหาตำแหน่งของผู้รับ

2. เมื่อ HLR ได้รับสัญญาณ Request ก็จะส่งสถานะของผู้รับ (Subscriber's status) กลับมายัง SMSC คือ สถานะของเครื่องรับ Inactive หรือ Active, ตำแหน่งของเครื่องรับถ้าสถานะของเครื่องรับเป็น Inactive แล้ว SMSC จะเก็บข้อความไว้ช่วงเวลาหนึ่ง และเมื่อใดที่เครื่องรับมีสถานะ Active แล้ว HLR จะส่ง SMS Notification ไปยัง SMSC และ SMSC ก็จะตอบรับข้อความนั้นไว้ จากนั้น SMSC จะส่งผ่านข้อความในรูปแบบ Short Message Delivery Point-to-Point ไปยังระบบบริการ โดยระบบจะทำการเรียกไปยังเครื่องรับและถ้าเครื่องรับมีการตอบรับกลับมา ข้อความก็จะถูกส่งตามไปและ SMSC จะได้รับการตอบยืนยันว่า ข้อความได้ถูกรับโดยปลายทางเรียบร้อยแล้ว หลังจากนั้นข้อความจะมีสถานะเป็น SENT และจะไม่ถูกส่งอีก

โหมดของการรับส่งข้อมูล SMS

แบ่งออกเป็น 2 โหมด คือ Text Mode และ PDU Mode (Protocol Description Unit Mode)

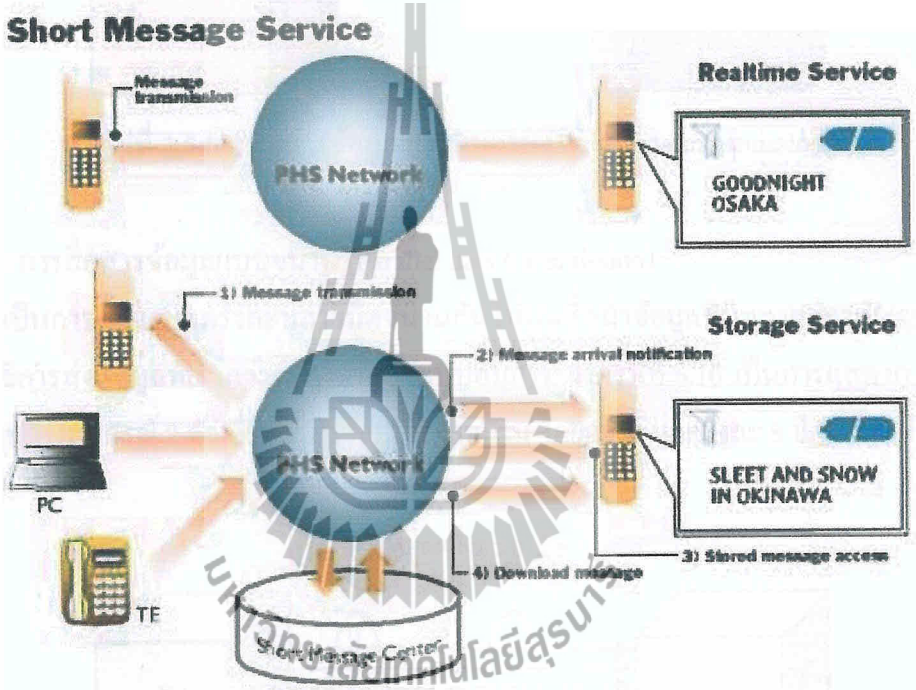
การส่งข้อความใน Text Mode นั้นจะเป็นการนำข้อความที่ต้องการส่งมาเข้ารหัสก่อน (โดยตัวเครื่องเอง) แล้วจึงส่งข้อมูลในรูป PDU Mode อีกครั้งหนึ่ง แต่ในบางเครื่องก็ไม่สนับสนุนการส่งแบบ Text Mode ผ่านทาง AT Command แต่หากเป็น PDU Mode จะสามารถส่งได้ นอกจากเครื่องจะไม่ต้องทำอาศัยการเปลี่ยนแปลงข้อมูลอีกชั้น

รูปแบบในการส่งข้อมูลในรูป SMS ผ่านทาง AT Command

มี 2 รูปแบบ คือ Text Mode และ PDU Mode

Text Mode เป็นการส่งข้อมูลในรูปของตัวอักษรได้โดยตรง ซึ่งตัวเครื่องส่วนใหญ่ไม่รองรับการส่งข้อมูลรูปแบบนี้ผ่านทาง AT Command จึงไม่สามารถใช้งานได้สมบูรณ์

PDU Mode PDU ย่อมาจาก PACKET DATA UNIT เป็นรูปแบบการส่งข้อความ SMS อีกรูปแบบหนึ่งที่ต้องมีการเข้ารหัสข้อมูลที่สลับซับซ้อนแต่ตัวเครื่องจะสามารถรับรู้ได้ทุกเครื่องที่รับคำสั่ง AT Command ได้



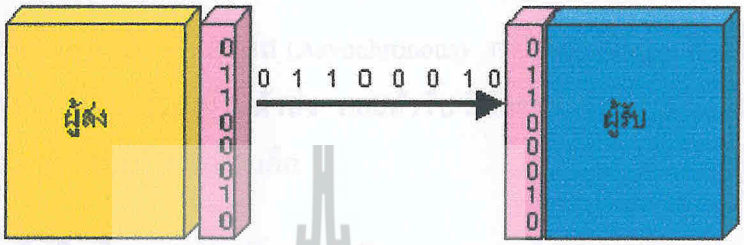
รูป 2.4 แสดงขอบเขตของการส่ง SMS

2.4 การสื่อสารข้อมูล

ระบบคอมพิวเตอร์สามารถส่งข้อมูลได้สองรูปแบบคือ แบบขนาน(parallel) ซึ่งจะส่งข้อมูลทุกบิตออกไปพร้อมกัน และแบบอนุกรม(serial) ซึ่งจะส่งข้อมูลออกไปครั้งละบิต การส่งข้อมูลแบบขนานนี้จะต้องใช้สายในการส่งข้อมูลจำนวนมากซึ่งไม่เหมาะกับการส่งระยะไกลโดยมากแล้วจะใช้ในการส่งระยะใกล้ เช่น ระหว่างไมพรเซบเซอร์กับฮาร์ดดีสก์ เครื่องพิมพ์ เป็นต้น ส่วนการส่งระยะไกลๆควรใช้การส่งข้อมูลแบบอนุกรมโดยจะส่งข้อมูลออกไปครั้งละบิต การส่งข้อมูลแบบอนุกรมนั้นจะส่งได้ช้ากว่าการส่งแบบขนานแต่ค่าใช้จ่ายจะต่ำกว่า ในไมโครโพรเซสเซอร์ 8051 จะมีพอร์ตอนุกรมอยู่ภายในซึ่งจะรับส่งข้อมูลได้สองทิศทางในพอร์ตเดียวกัน

- การสื่อสารข้อมูลแบบอนุกรม (Serial data transmission)

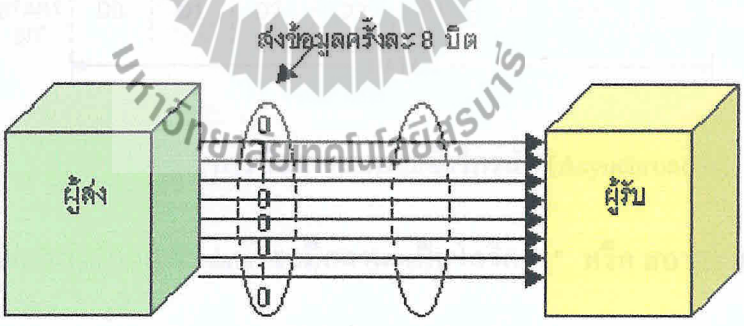
เป็นการส่งข้อมูลครั้งละ 1 บิต ไปบนสัญญาณจนครบจำนวนข้อมูลที่มีอยู่ สามารถนำไปใช้กับสื่อนำข้อมูลที่มีเพียง 1 ช่องสัญญาณได้ สื่อนำข้อมูลที่มี 1 ช่องสัญญาณนี้จะมีราคาถูกกว่าสื่อนำข้อมูลที่มีหลายช่องสัญญาณ และเนื่องจากการสื่อสารแบบอนุกรมมีการส่งข้อมูลได้ครั้งละ 1 บิต เท่านั้น การส่งข้อมูลประเภทนี้จึงช้ากว่าการส่งข้อมูลครั้งละหลายบิต



รูปที่ 2.5 การสื่อสารข้อมูลแบบอนุกรม (Serial data transmission)

- การสื่อสารข้อมูลแบบขนาน (Parallel data transmission)

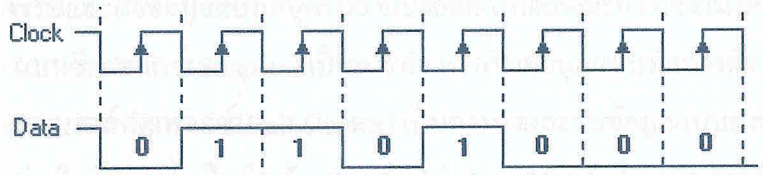
เป็นการส่งข้อมูลครั้งละหลายบิตขนานกันไปบนสื่อนำข้อมูลที่มีหลายช่องสัญญาณ วิธีนี้จะเป็นวิธีการส่งข้อมูลที่เร็วกว่าการส่งข้อมูลแบบอนุกรม จากรูปที่ 6.10 เป็นการแสดงการสื่อสารข้อมูลระหว่างอุปกรณ์ 2 ตัว ที่มีการส่งข้อมูลแบบขนาน โดยส่งข้อมูลครั้งละ 8 บิตพร้อมกัน



รูปที่ 2.6 การสื่อสารข้อมูลแบบขนาน (Parallel data transmission)

1.) รูปแบบการสื่อสารแบบอนุกรม มีด้วยกันอยู่ 2 แบบ คือแบบซิงโครนัส (Synchronous) และแบบอะซิงโครนัส (Asynchronous)

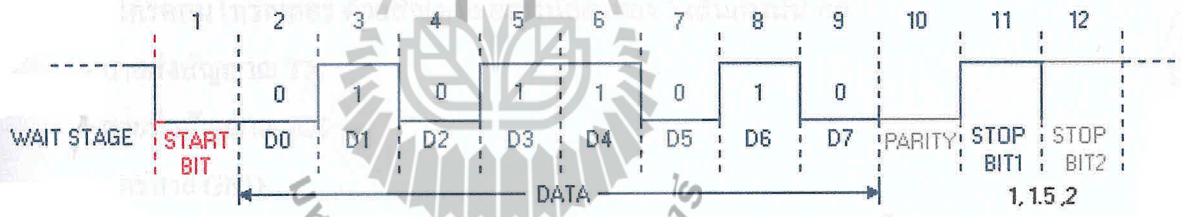
- การสื่อสารแบบซิงโครนัส (Synchronous) การรับส่งข้อมูล จะมีสัญญาณนาฬิกา ซึ่งเป็นตัวกำหนด จังหวะเวลา การส่งข้อมูล ร่วมอยู่ด้วยอีกเส้นหนึ่ง ใช้คู่กับสัญญาณข้อมูล ตัวอย่างเช่น การส่งสัญญาณจากคีย์บอร์ด



รูปที่ 2.7 การสื่อสารแบบซิงโครนัส (Synchronous)

- การสื่อสารแบบอะซิงโครนัส (Asynchronous) การรับส่งข้อมูล โดยที่ไม่จำเป็นต้อง มีสัญญาณนาฬิกา ร่วมด้วย แต่จะใช้ให้ตัวส่ง และตัวรับ มีอัตราส่งข้อมูล ที่เท่ากัน รูปแบบข้อมูลแบบอะซิงโครนัส ประกอบด้วย 4 ส่วนคือ

1. บิตเริ่มต้น(Start bit) มีขนาด 1 บิต
2. บิตข้อมูล(Data) มีขนาด 5,6,7 หรือ 8 บิต
3. บิตตรวจสอบพาริตี้ (Parity bit) มีขนาด 1 บิตหรือไม่มี
4. บิตหยุด(Stop bit) มีขนาด 1,1.5,2 บิต



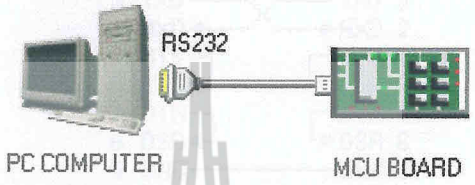
รูปที่ 2.8 การสื่อสารแบบอะซิงโครนัส (Asynchronous)

- เมื่อไม่มีการส่งข้อมูล ขา data จะมีสถานะเป็นโลจิก "1" หรือ สถานะหยุดรอ (Waiting stage)
- เมื่อเริ่มต้นส่งข้อมูลจะให้ขา data เป็นโลจิก "0" เป็นจำนวน 1 บิต เรียกว่าบิตเริ่มต้น (Start bit)
- จากนั้นก็จะเริ่มต้นส่งข้อมูล โดยส่งบิตต่ำไปก่อน (LSB)
- แล้วตามด้วยพาริตี้บิต (จะมีหรือไม่ก็ได้ ขึ้นอยู่กับการติดตั้งค่า ของทั้งสองฝ่าย)
- สุดท้ายตามด้วยโลจิก "1" อย่างน้อย 1 บิต (มีขนาด 1, 1.5, หรือ 2 บิต) เพื่อแสดงว่าสิ้นสุดข้อมูล

2.) การรับและส่งข้อมูลแบบอนุกรมยังแบ่งออกเป็นลักษณะการใช้งานได้ 3 แบบคือ

- แบบซิมเพลกซ์(Simplex) เป็นการส่ง หรือรับข้อมูลแบบทิศทางเดียวเท่านั้น
- แบบฮาล์ฟดูเพลกซ์(Half Duplex) เป็นการส่งและรับข้อมูลแบบสลับกันคือ เมื่อด้านหนึ่งส่ง อีกด้านหนึ่งเป็นฝ่ายรับ สลับกันไม่สามารถรับ-ส่งในเวลาเดียวกันได้
- แบบฟลูดูเพลกซ์(Full Duplex) สามารถรับ-ส่งข้อมูลในเวลาเดียวกัน

3.) การใช้งานพอร์ต RS-232



รูปที่ 2.9 การสื่อสารข้อมูล RS-232

การสื่อสารแบบอนุกรม นับว่ามีความสำคัญ ต่อการใช้งาน ไมโครคอนโทรลเลอร์มาก เพราะสามารถใช้แป้นพิมพ์ และจอภาพของ PC เป็น อินพุต และ เอาต์พุต ในการติดต่อ หรือควบคุม ไมโครคอนโทรลเลอร์ ด้วยสัญญาณอย่างน้อย เพียง 3 เส้นเท่านั้น คือ

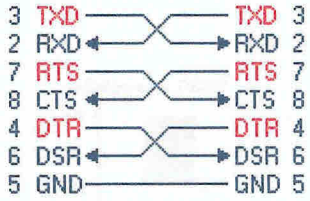
- สายส่งสัญญาณ TX
- สายรับสัญญาณ RX
- และสาย GND

โดยปกติพอร์ตอนุกรม RS-232C จะสามารถต่อสายได้ยาว 50 ฟุตโดยประมาณ ขึ้นอยู่กับ ชนิดของสายสัญญาณ, ระยะทาง, และ ปริมาณ สัญญาณ รบกวน

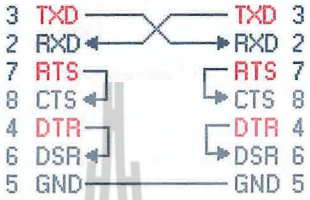
รูปที่ 2.10 พอร์ตอนุกรมของ PC DB9 ตัวผู้ (Male)

รูปที่ 2.11 พอร์ตอนุกรมของอุปกรณ์ภายนอก DB9 ตัวเมีย (Female)

- พอร์ตอนุกรมของ PC จะเป็นคอนเน็คเตอร์แบบ DB9 ตัวผู้ (Male)
- พอร์ตอนุกรม ของอุปกรณ์ภายนอก จะเป็นคอนเน็คเตอร์แบบ DB9 ตัวเมีย(FeMale)



รูปที่ 2.12 การเชื่อมต่ออุปกรณ์ภายนอกผ่าน DB9 แบบ Null modem



รูปที่ 2.13 การต่ออุปกรณ์ภายนอกผ่าน DB9 แบบ 3 เส้น

4.) การทำงานของขาสัญญาณ DB9

TXD เป็นขาที่ใช้ส่งข้อมูล

RXD เป็นขาที่ใช้รับข้อมูล

DTR แสดงสภาวะพอร์ตว่าเปิดใช้งาน, **DSR** ตรวจสอบว่าพอร์ต ที่ติดต่อด้วย เปิดอยู่หรือไม่

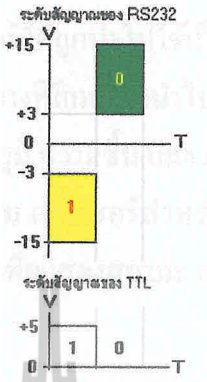
- เมื่อเปิดพอร์ตอนุกรม ขา DTR จะ ON เพื่อให้อุปกรณ์ได้รับทราบว่าต้องการติดต่อด้วย
- ในขณะเดียวกันก็จะตรวจสอบขา DSR ว่าอุปกรณ์พร้อมหรือไม่

RTS แสดงสภาวะพอร์ตว่าต้องการส่งข้อมูล, **CTS** ตรวจสอบว่าพอร์ตที่ติดต่ออยู่ ต้องการส่งข้อมูลหรือไม่

- เมื่อต้องการส่งข้อมูลขา RTS จะ ON และจะส่งข้อมูลออกที่ขา TXD เมื่อส่งเสร็จก็จะOFF
- ในขณะเดียวกันก็จะตรวจสอบขา CTS ว่าอุปกรณ์ต้องการที่จะส่งข้อมูลหรือไม่

GND ขา ground

5.) ระดับสัญญาณของ RS232



รูปที่ 2.14 แสดงระดับสัญญาณของ RS232C และระดับสัญญาณของ TTL

- สัญญาณรบกวนที่เกิดขึ้น ในสายนำสัญญาณ มักจะมีแรงดันเป็นบวก เมื่อเทียบกับกราวด์
- เพื่อป้องกันสัญญาณรบกวนนี้ จึงออกแบบแรงดัน ของโลจิก "1" เป็นลบ คืออยู่ในช่วง 3V ถึง -15V ส่วนแรงดัน ของโลจิก "0" อยู่ในช่วง +3V ถึง +15V

- และเหตุที่ ระดับสัญญาณ ของ RS232 อยู่ในช่วง +15V ถึง -15V ก็เพื่อให้ต่อสายสัญญาณ ไปได้ไกลขึ้น

- ดังนั้นจึงจำเป็นจะต้องมีวงจรเปลี่ยนระดับแรงดันของ RS232 มาเป็นระดับแรงดันของ TTL

6.) อัตราการส่งข้อมูล (Baud rate)

คือความเร็วของการรับ-ส่งข้อมูล เป็นจำนวนบิทต่อวินาทีเช่น 300, 1,200, 2,400, 4,800, 9,600, 14,400, 19,200, 38,400, 56,000 เป็นต้น การเลือกอัตราการส่งข้อมูลขึ้นอยู่กับ ชนิดของสายสัญญาณ, ระยะทาง, และปริมาณสัญญาณรบกวน

2.5 เซ็นเซอร์(sensor

เซนเซอร์เป็นตัวที่ใช้ตรวจจับสภาวะใดๆ เช่น อุณหภูมิ แสง สี เสียง หรือวัตถุ ต่างๆ โดยอาศัยหลักการที่แตกต่างกันไปแต่ละตัวเพื่อเปลี่ยนจากคุณสมบัติทางฟิสิกส์มาเป็นคุณสมบัติทางไฟฟ้า โดยทั่วไปเทคโนโลยีของเซนเซอร์ได้ถูกนำไปใช้เป็นองค์ประกอบหลักที่สำคัญในลักษณะงานสองประเภทคือ ใช้ตรวจวัดปริมาณทางฟิสิกส์เพื่อนำไปแสดงผลการตรวจวัดหรือจัดเก็บบันทึกข้อมูลในระบบการวัด เช่น การวัดอุณหภูมิ ความชื้น แสง เป็นต้น และอย่างที่สองคือใช้ตรวจสอบสภาพกระบวนการในระบบการควบคุม เซนเซอร์สำหรับการตรวจวัดข้อมูลที่เป็นตัวแปรทางฟิสิกส์ โดยมากจะถูกนำไปใช้เป็นข้อมูลเพื่อแสดงสถานะ สภาพของระบบในขณะนั้น

2.5.1 Temperature sensor

การตรวจวัดอุณหภูมิใช้รูปแบบการเปลี่ยนแปลงระดับแรงดันไฟฟ้าจากสัญญาณอนาล็อกไปสู่สัญญาณดิจิตอล โดยสัมพันธ์กับอุณหภูมิ

เทอร์โมคัปเปิล คืออุปกรณ์วัดอุณหภูมิโดยใช้หลักการเปลี่ยนแปลงอุณหภูมิหรือความร้อนเป็นแรงเคลื่อนไฟฟ้า (emf) เทอร์โมคัปเปิลทำมาจากโลหะตัวนำที่ต่างชนิดกัน 2 ตัว (แตกต่างกันทางโครงสร้างของอะตอม) นำมาเชื่อมต่อปลายทั้งสองเข้าด้วยกันที่ปลายด้านหนึ่ง เรียกว่าจุดวัดอุณหภูมิ ส่วนปลายอีกด้านหนึ่งปล่อยเปิดไว้ เรียกว่าจุดอ้างอิง หากจุดวัดอุณหภูมิและจุดอ้างอิงมีอุณหภูมิต่างกันก็จะทำให้มีการนำกระแสในวงจรเทอร์โมคัปเปิลทั้งสองข้าง

American Limits of Error ASTM E230-ANSI MC 96.1

ANSI Code		Standard Limits†		Special Limits†	
J	Temp Range	>0 to 750°C	>32 to 1382°F	0 to 753°C	32 to 1382°F
	Tolerance Value	2.2°C or 0.75%	4.0°F or 0.75%	1.1°C or 0.4%	2.0°F or 0.4%
K	Temp Range	>0 to 1250°C	>32 to 2282°F	0 to 1250°C	32 to 2282°F
	Tolerance Value	2.2°C or 0.75%	4.0°F or 0.75%	1.1°C or 0.4%	2.0°F or 0.4%
	Temp. Range*	-200 to 0°C	-328 to 32°F		
	Tolerance Value	2.2°C or 2.0%	4.0°F or 2.0%		
T	Temp Range	>0 to 350°C	>32 to 662°F	0 to 353°C	32 to 662°F
	Tolerance Value	1.0°C or 0.75%	1.8°F or 0.75%	0.5°C or 0.4%	1°F or 0.4%
	Temp. Range*	-200 to 0°C	-328 to 32°F		
	Tolerance Value	1.0°C or 1.5%	1.8°F or 1.5%		
E	Temp Range	>0 to 900°C	>32 to 1652	0 to 903°C	32 to 1652°F
	Tolerance Value	1.7°C or 0.5%	3°F or 0.5%	1.0°C or 0.4%	1.8°F or 0.4%
	Temp. Range*	-200 to 0°C	-328 to 32°F		
	Tolerance Value	1.7°C or 1.0%	3°F or 1.0%		
N	Temp Range	>0 to 1300°C	>32 to 2372°F	0 to 1300°C	32 to 2372°F
	Tolerance Value	2.2°C or 0.75%	4.0°F or 0.75%	1.1°C or 0.4%	2.0°F or 0.4%
	Temp. Range*	-270 to 0°C	-454 to 32°F		
	Tolerance Value	2.2°C or 2.0%	4.0°F or 2.0%		
R S	Temp Range	0 to 1450°C	32 to 2642°F	0 to 1450°C	32 to 2642°F
	Tolerance Value	1.5°C or 0.25%	2.7°F or 0.25%	0.6°C or 0.1%	1°F or 0.1%
B	Temp Range	600 to 1700°C	1472 to 3092°F	Not Established	
	Tolerance Value	0.5%	0.9°F		
G*C*D*	Temp Range	0 to 2320°C	32 to 4208°F	Not Established	
	Tolerance Value	4.5°C or 1.0%	8.0°F		

* Not official symbol or standard designation † Whichever value is greater

Note: Material is normally selected to meet tolerances above 0°C. If thermocouples are needed to meet tolerances below 0°C, the purchaser shall state this as selection of material is usually required.

IEC Tolerance Class EN 60584-2; JIS C 1602

IEC Code		Class 1	Class 2	Class 3†
J	Temp Range	-40 to 375°C	-40 to 333°C	
	Tolerance Value	±1.5°C	±2.5°C	Not Established
	Temp. Range	375 to 750°C	333 to 750°C	
	Tolerance Value	±0.4% Reading	±0.75% Reading	
K N	Temp Range	-40 to 375°C	-40 to 333°C	-167 to 40°C
	Tolerance Value	±1.5°C	±2.5°C	±2.5°C
	Temp. Range	375 to 1000°C	333 to 1200°C	-200 to -167°C
	Tolerance Value	±0.4%	±0.75% Reading	±1.5% Reading
T	Temp Range	-40 to 125°C	-40 to 133°C	-67 to 40°C
	Tolerance Value	±0.5°C	±1°C	±1°C
	Temp. Range	125 to 350°C	133 to 350°C	-200 to -67°C
	Tolerance Value	±0.4% Reading	±0.75% Reading	±1.5% Reading
E	Temp Range	-40 to 375°C	-40 to 333°C	-167 to 40°C
	Tolerance Value	±1.5°C	±2.5°C	±2.5°C
	Temp. Range	375 to 800°C	333 to 900°C	-200 to -167°C
	Tolerance Value	±0.4% Reading	±0.75% Reading	±1.5% Reading
R S	Temp Range	0 to 1100°C	0 to 600°C	
	Tolerance Value	±1°C	±1.5°C	Not Established
	Temp. Range	1100 to 1600°C	600 to 1600°C	
	Tolerance Value	±[1 + 0.3%x (Rdg-1100)]°C	±0.25% Reading	
B	Temp Range			600 to 800°C
	Tolerance Value	Not Established		±4°C
	Temp. Range		600 to 1700°C	800 to 1700°C
	Tolerance Value		±0.25% Reading	±0.5% Reading

† Material is normally selected to meet tolerances above –40°C. If thermocouples are needed to meet limits of Class 3, as well as those of Class 1 or 2, the purchaser shall state this, as selection of material is usually required.

รูป 2.15 แสดงชนิดต่างๆ ของ *Thermocouple*

ชนิดของเซ็นเซอร์ที่ใช้ในโครงงาน คือ ชนิด T (copper–constantan) เหมาะกับการวัดอุณหภูมิในช่วง -200 ถึง 350 °องศาเซลเซียส มีความไวประมาณ 43 μV/°C.

รูป 2.16 Thermocouple probe

2.5.2 pH sensor

pH เซ็นเซอร์ คืออุปกรณ์ไฟฟ้าเคมี (electrochemical) ที่สร้างแรงเคลื่อนไฟฟ้าซึ่งเป็นสัดส่วนกันค่า pH ของสารละลายที่นำไปวัด

ส่วน pH มิเตอร์ก็ใช้หลักการที่กล่าวมา คือใช้หลักการวัดค่าความเป็นกรดหรือด่างที่มีน้ำเป็นตัวทำละลาย ซึ่งก็ใช้หลักการทางเคมีไฟฟ้าเหมือนกัน หลังจากนั้นก็นำค่าความต่างศักย์ที่เกิดขึ้นมาระหว่างอิเล็กโทรดอ้างอิง (reference electrode)กับอิเล็กโทรดวัด(sensing electrode)ไปขยาย ปรับสภาพ และแสดงผลต่อไป

หลักการวัดค่า pH

เมื่อสารละลาย 2 ตัวที่แตกต่างกันถูกใส่สงในด้านทั้งสองของเมมเบรนแก้ว จะทำให้เกิดแรงเคลื่อนไฟฟ้าที่เป็นสัดส่วนกับความแตกต่างของ pH ระหว่างสารละลายทั้งสอง และตกคร่อมทั้งสองด้านของเมมเบรน

ข้อดีของการวัดด้วยวิธีนี้ได้แก่

1. ย่านการวัดกว้าง (ไม่มีปัญหาช่วง pH 0 – 14)
2. ให้ผลตอบสนองที่รวดเร็ว (วัดในช่วงเวลาสั้น ๆ)
3. การปฏิบัติงานง่าย การวัดต่อเนื่องทำได้ง่าย
4. การผลิตซ้ำดี พร้อมกับมีความผิดพลาดโดยผู้ปฏิบัติงานน้อย
5. ความผิดพลาดเนื่องจากผลของเกลือและโปรตีนต่ำกว่าวิธีอื่นๆ (วิธีการวัดแบบอื่น ๆ ไม่สามารถวัดสารตัวอย่างที่ประกอบด้วยสสารแบบออกซิไดซิงและรีดิวซิงได้)

ข้อเสียของการวัดด้วยวิธีนี้ได้แก่

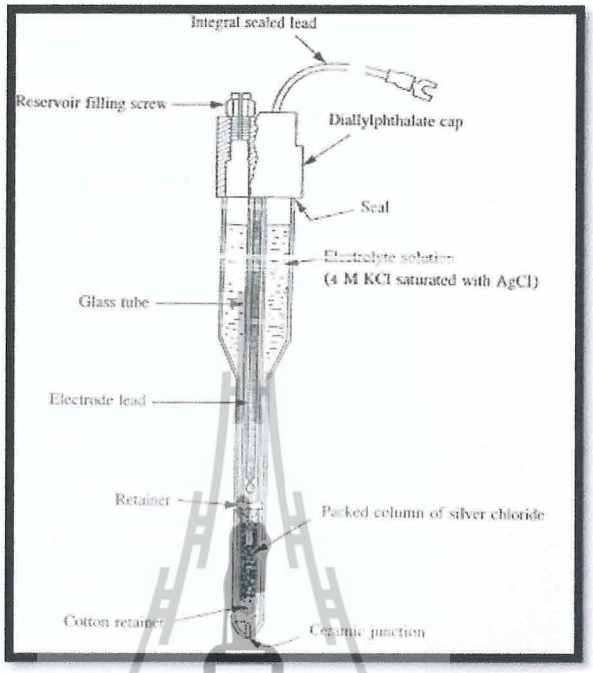
1. เมมเบรนเปราะ แตกได้ง่าย
2. ความต้านทานภายในของอิเล็กโทรดสูงทำให้ต้องใช้มิลลิโวลต์มิเตอร์ที่มีความต้านทานอินพุตสูง

ส่วนประกอบและการทำงานของอิเล็กโทรดวัดและอิเล็กโทรดอ้างอิง

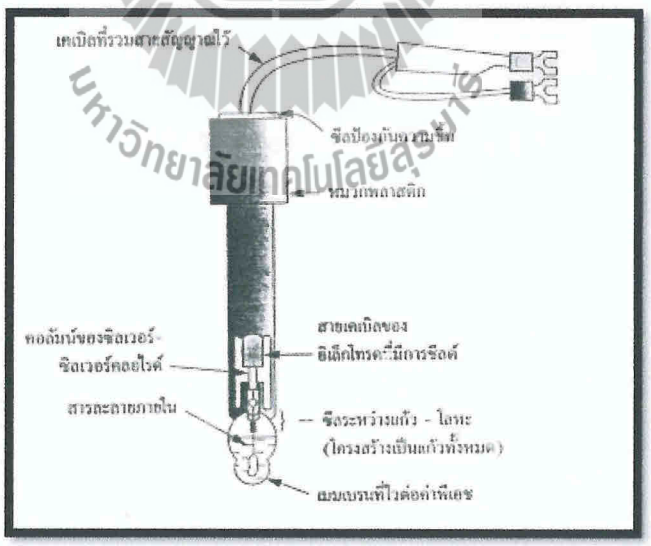
1. ขั้ววัด (Sensors Electrode) โดยปกติอิเล็กโทรดวัดจะทำจากแก้ว ขั้ววัดจะมีกระเปาะแก้วบางมาก ๆเพื่อให้ไวต่อออิออน H^+ ภายในอิเล็กโทรดจะบรรจุสารละลายกันชนที่เรียกว่าบัฟเฟอร์ (buffer) ซึ่งมีค่า pH คงที่ (ประมาณ pH 7) สารละลายนี้ได้แก่ KCl อิ่มตัว ในสารละลายดังกล่าวจะมีขั้วไฟฟ้าซึ่งทำด้วยเงิน (silver) ฉาบด้วยซิลเวอร์คลอไรด์จุ่มอยู่ ดังแสดงในรูปที่ 2.16

2. ขั้วอ้างอิง (Reference Electrode) หรือเรียกว่าขั้วเปรียบเทียบ มีไว้เพื่อให้ศักย์ไฟฟ้าที่ขั้ววัดครบวงจร ขั้วอ้างอิงนี้จะมีลวดนำ (lead wire) ต่อกับซิลเวอร์คลอไรด์และจุ่มซิลเวอร์คลอไรด์ลงใน KCl ชนิดอิ่มตัว KCl นี้จะซึมผ่านรูเล็ก ๆ ที่ปลายของหลอดแก้วเพื่อสัมผัสกับ

สารละลายที่จะวัด การเชื่อมต่อระหว่างสาร ละลายทั้งสองนี้เรียกว่า “salt bridge” รูที่เชื่อมต่อนี้มักจะมีสารเอสเบสตอสหรือเซรามิกวางคั่นเอาไว้ เพื่อทำให้การไหลของ KCl คงที่ตลอดเวลาขณะที่ทำการวัด ดังรูปที่ 2.17



รูปที่ 2.17 แสดงตัวอย่างของอิเล็กโทรดวัดค่าพีเอชแบบแก้ว



รูปที่ 2.18 ตัวอย่างอิเล็กโทรดอ้างอิงที่วัดค่าพีเอช(อิเล็กโทรดแบบซิลเวอร์/ซิลเวอร์-คลอไรด์)

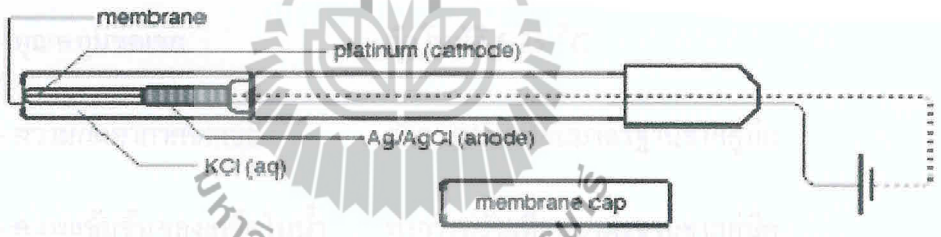
สาร	pH
กรดสารพิษจากเหมืองร้าง	-3.6 - 1.0
กรดจากแบตเตอรี	-0.5
กรดในกระเพาะอาหาร	1.5 - 2.0
เลมอน	2.4
Coke	2.5
น้ำส้มสายชู	2.9
ส้ม หรือ แอปเปิล	3.5
เบียร์	4.5
ฝนกรด	< 5.0
กาแฟ	5.0
ชา	5.5
นม	6.5
น้ำบริสุทธิ์	7.0
น้ำลายมนุษย์	6.5 - 7.4
เลือด	7.34 - 7.45
น้ำทะเล	8.0
สบู่ล้างมือ	9.0 - 10.0
แอมโมเนีย (ยาสามัญประจำบ้าน)	11.5
น้ำยาปรับผ้านุ่ม	12.5
โซดาไฟ	13.5

รูป 2.19 ตัวอย่างค่า pH ของสารต่าง ๆ

2.5.3 Dissolved Oxygen sensor (DO Sensor)

การทำงานของ Dissolved Oxygen probe

Vernier Dissolved Oxygen Probe เป็น Probe ชนิด Clark-cell มีหัววัดแบบ polarographic electrodeที่ใชในการสัมผัสกับความเข้มข้นของออกซิเจนที่ละลายในน้ำ โดยมี ทองคำขาวเป็นขั้ว cathode และมี silver/silver chloride เป็นขั้วอ้างอิง (anode เป็นขั้วอ้างอิง) ส่วนอิเล็กโทรไลต์ (electrolyte) ที่ใช้คือ KCLเป็นตัวคัดเลือกเอาเฉพาะสารละลายออกซิเจนที่ละลายในน้ำเท่านั้นที่สามารถซึมผ่านmembrane เข้ามาได้เพื่อทำปฏิกิริยากับขั้วไฟฟ้า หัววัดชนิด polarographic นั้นเป็นหัววัดที่ต้องได้รับแรงดันไฟฟ้าจากภายนอกก่อน ซึ่งโดยทั่วไปจะใช้แบตเตอรี่ (เช่น ถ่าน) เป็นแหล่งกำเนิดแรงดันไฟฟ้าจากภายนอก จึงจะสามารถนำไปใช้วัดได้ ซึ่งแรงดันไฟฟ้าที่ใช้ในการ polarize ก็จะส่งผ่านมาเข้าสู่ขั้วแคโทดและแอโนด เมื่อนำไปใช้งาน ออกซิเจนที่ละลายในน้ำจะเข้าสู่กระบวนการผ่านทางช่องของ membrane เข้าสู่สารละลายอิเล็กโทรไลต์และจะถูกรีดิวซ์ที่ขั้วแคโทด แล้วจึงวัดผลที่ได้ออกมา



รูป 2.20 ส่วนประกอบของ Dissolved Oxygen probe

การเกิดปฏิกิริยา

ขั้วไฟฟ้า	ทำปฏิกิริยา	ผลตอบสนอง
cathode	reduced	$\frac{1}{2}O_2 + H_2O + 2e^- \longrightarrow 2\,OH^-$
reference electrode (anode)	oxidation	$Ag + Cl^- \longrightarrow AgCl + e^-$

จากตารางจะเห็นได้ว่า กระแสจะทำสัดส่วนกับการกระจายของออกซิเจน ในทางกลับกันจะได้ค่าความเข้มข้นของออกซิเจนที่ละลายในน้ำ ซึ่งกระแสนี้จะเปลี่ยนไปอยู่ในรูปของแรงดันไฟฟ้า

คุณสมบัติของ Dissolved Oxygen probe

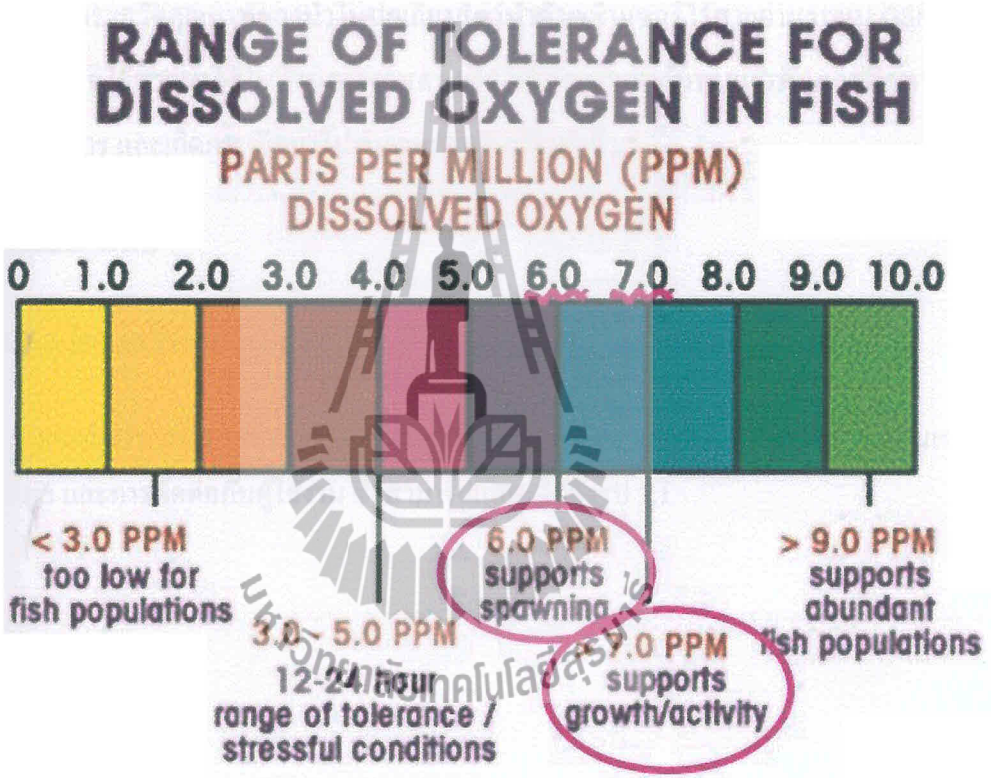
- ช่วงการวัด	0 – 15 mg/L (or ppm)
- ความถูกต้องในการวัด	± 0.2 mg/L
- ความละเอียดทางการวัด	13-bit resolution (SensorDAQ): 0.007 mg/L 12-bit resolution (LabPro, LabQuest,LabQuest Mini, Go!Link, ULI II, SBI): 0.014 mg/L 10-bit resolution (CBL 2): 0.056 mg/L
- ผลตอบสนองทางเวลา	95 % ภายใน 30 วินาที, 98 % ภายใน 45 วินาที
- ตัวอย่างการไหลขั้นต่ำ	20 cm/second
- การเก็บค่าข้อมูลปรับเทียบมาตรฐาน	Slope = 3.27 Intercept = –0.327
- อุณหภูมิชดเชย	อัตโนมัติ 5-35 $^{\circ}C$
- ความกดอากาศชดเชย	ทำการปรับเทียบมาตรฐานจากคู่มือ
- ความเข้มข้นของเกลือในน้ำ	ทำการปรับเทียบมาตรฐานจากคู่มือ

ค่าของออกซิเจนที่ละลายในน้ำ (Dissolved Oxygen : DO)

ออกซิเจนมีความสำคัญต่อการดำรงชีวิตของสัตว์น้ำ การขาดแคลนออกซิเจนในน้ำถึงแม้ว่าจะไม่ต่ำลงจนถึงระดับที่ทำให้สัตว์น้ำตาย แต่มีผลต่อการดำรงชีวิตหลายประการ โดยพบว่าหากมีปริมาณออกซิเจนที่ต่ำกว่า 3 มิลลิกรัมต่อลิตร ทำให้ระยะเวลาในการฟักตัวจากไข่ช้ากว่าปกติและความแข็งแรงของตัวอ่อนน้อยลง ทำให้ตัวอ่อนมีลักษณะผิดปกติ นอกจากนี้ปริมาณออกซิเจนที่น้อยลงทำให้ประสิทธภาพการย่อยอาหาร การต้านทานสารพิษลดลง เป็นสาเหตุทำให้ปลาอ่อนแอติดโรคง่ายขึ้น ปริมาณออกซิเจนที่มีผลกระทบต่อปลามีดัง ตาราง 2.1

ตาราง 2.1 ปริมาณออกซิเจนที่มีผลต่อปลา

0-0.30 มิลลิกรัมต่อลิตร	ปลาขนาดเล็กมีชีวิตรอดในระยะเวลาสั้น ๆ
0.31-2.00 มิลลิกรัมต่อลิตร	เป็นอันตรายต่อปลาหากอยู่ในสภาวะนี้นาน ๆ
2.10-4.00 มิลลิกรัมต่อลิตร	เจริญเติบโตช้าและติดเชื้อโรคได้ง่าย
5.00 มิลลิกรัมต่อลิตรขึ้นไป	เหมาะสมต่อการเจริญเติบโตของปลา



รูป 2.21 ปริมาณออกซิเจนที่ผลต่อปลา

บทที่ 3

ระบบเฝ้าระวังบ่อเลี้ยงสัตว์น้ำด้วยเซ็นเซอร์ไร้สายผ่านระบบ GSM

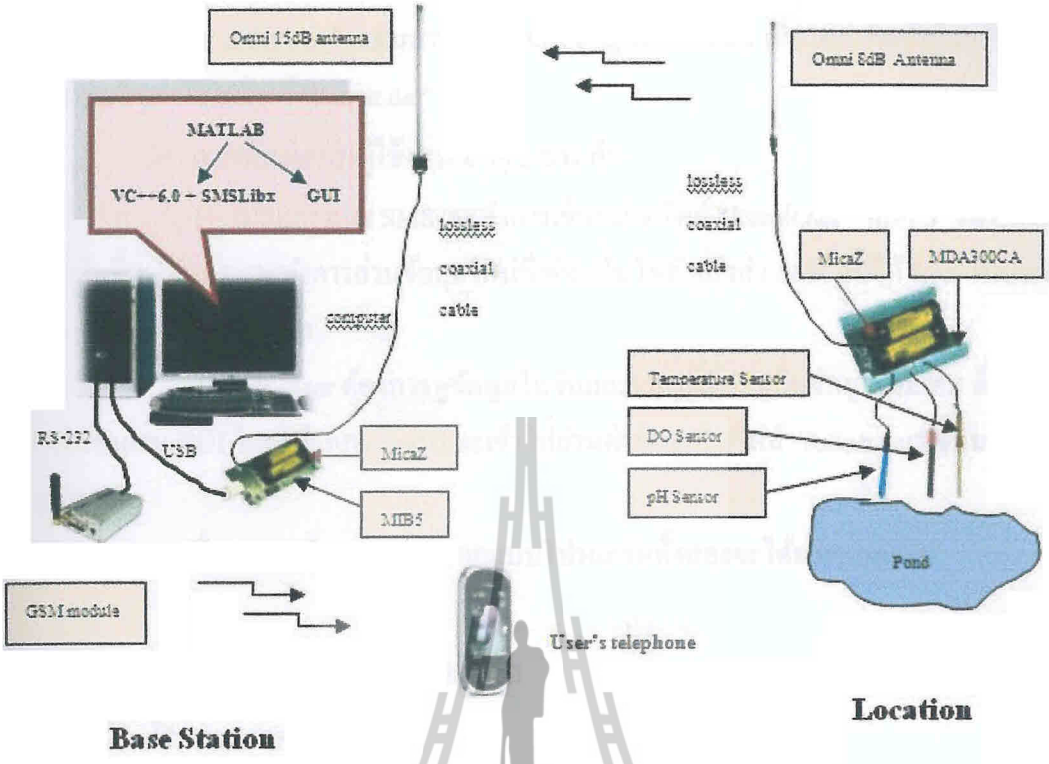
3.1 บทนำ

โครงงานเรื่อง **ระบบเฝ้าระวังบ่อเลี้ยงสัตว์น้ำด้วยเซ็นเซอร์ไร้สายผ่านระบบ GSM** นี้ จัดทำขึ้นเพื่อทดสอบการตรวจวัดคุณภาพของน้ำในบ่อเลี้ยงสัตว์น้ำด้วยเซ็นเซอร์ไร้สายผ่านระบบ GSM ซึ่งเป็นการนำเทคโนโลยีไร้สายมาใช้ในด้านการเกษตร เพื่อให้เกิดการลดต้นทุนแต่เพิ่มผลผลิตทางการเกษตรให้กับเกษตรกร และเกิดการพัฒนาโปรแกรมสำหรับผู้จัดทำด้วย

3.2 รูปแบบระบบ

3.2.1 โครงสร้างระบบ

การทำงานของระบบจะแบ่งออกเป็นสามส่วนหลัก ได้แก่ ระบบเครือข่ายเซ็นเซอร์ไร้สาย, ฐานข้อมูล และการติดต่อกับผู้ใช้งาน ซึ่งสามารถแสดงได้ดังรูป 3.1



รูป 3.1 โครงสร้างของระบบ

3.2.2 หลักการทำงานในแต่ละส่วนของระบบ

➢ **ระบบเครือข่ายเซ็นเซอร์ไร้สาย**

เครือข่ายเซ็นเซอร์ไร้สาย ประกอบด้วยอุปกรณ์ที่ทำหน้าที่เป็นโหนดเซ็นเซอร์(Sensor node) และโหนดสถานีฐาน (Base station node) โหนดเซ็นเซอร์ทำหน้าที่ในการส่งข้อมูลที่วัดจากเซ็นเซอร์ไปยังโหนดสถานีฐานผ่านทางคลื่นความถี่วิทยุ ส่วนโหนดสถานีฐานจะทำหน้าที่ในการติดต่อสื่อสารระหว่างเครือข่ายเซ็นเซอร์ไร้สาย กับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม (Serial port)

➢ **ฐานข้อมูล**

การเก็บข้อมูลจะทำผ่านโปรแกรม MATLAB R2009a โดยการรัน code โปรแกรม อ่านค่าจาก serial port ที่ต่อกับโหนดสถานีฐาน ซึ่งเป็นค่าของแรงดันไฟฟ้า (ADC) จึงต้องทำการแปลงค่าจากแรงดันไฟฟ้า (ADC) ไปเป็นค่าของ อุณหภูมิ(Temperature), กด-ด่าง (pH)และออกซิเจนที่ละลายในน้ำ

(DO) ที่แท้จริง โดย calibrate ด้วยสมการเฉพาะของข้อมูลแต่ละชนิด เมื่อได้ค่าที่แท้จริงแล้ว โปรแกรมจะทำการเซ็บค่าไว้ในไฟล์ "Result.dat"

➢ **การติดต่อกับผู้ใช้งาน** มีอยู่ 2 ทาง คือ

1) โปรแกรมส่ง SMS จะทำการเช็คขนาดไฟล์ "Result.dat" ทุกๆ 5 วินาทีหากขนาดไฟล์เพิ่มขึ้น โปแกรมจะทำการอ่านข้อมูลใหม่ที่เข้ามาในไฟล์ แล้วส่ง SMS ไปให้ User ผ่านทาง GSM Module เข้าโทรศัพท์มือถือ

2) เมื่อ User ต้องการดูข้อมูลในวันและเวลาที่มีการเก็บข้อมูลเพิ่มเติม ดีสามารถดูได้จากโปรแกรม GUI โดยโปรแกรม GUI จะเข้าไปอ่านค่าข้อมูลในไฟล์ "Result.dat" ขึ้นมาแสดงในรูปของกราฟ

ซึ่งรายละเอียดของการออกแบบโปรแกรมทั้งสองจะได้อธิบายต่อไป

3.2.3 หน้าที่ของอุปกรณ์ในแต่ละส่วนของระบบ

➢ **ระบบเครือข่ายเซ็นเซอร์ไร้สาย**

- pH sensor, Temperature sensor และ DO sensor ทำหน้าที่ในการวัดค่า ความเป็นกรด-ด่าง(pH) ,อุณหภูมิ(Temperature) และออกซิเจนที่ละลายในน้ำ(DO) ตามลำดับค่าที่ได้อยู่ในรูปของแรงดันไฟฟ้า

- MDA300CA ทำหน้าที่เป็นตัวอินเตอร์เฟซระหว่าง pH sensor, Temperature sensor และ DO sensor กับ MicaZ(ตัวส่ง)

- MicaZ (ตัวส่ง) คือโหนดเซ็นเซอร์ ทำหน้าที่ในการส่งค่าที่วัดได้จากเซ็นเซอร์ไปยังโหนดสถานีฐาน

- Lossless Coaxial Cable ทำหน้าที่ในการนำส่งสัญญาไฟฟ้าณระหว่าง MicaZ กับสายอากาศ

- 8dB omni Antenna คือ สายอากาศทางด้านส่ง ทำหน้าที่ในการแปลงสัญญาณไฟฟ้าไปเป็นสัญญาณคลื่นความถี่วิทยุส่งไปในอวกาศ

- 15dB omni Antenna คือ สายอากาศทางด้านรับ ทำหน้าที่แปลงสัญญาณคลื่นความถี่วิทยุไปเป็นสัญญาณไฟฟ้า

- MicaZ (ตัวรับ) คือโหมดเซ็นเซอร์ ทำหน้าที่ในการรับค่าที่วัดได้จากเซ็นเซอร์จากโหนดเซ็นเซอร์

- MIB5 ทำหน้าที่เป็นตัวอินเตอร์เฟซระหว่าง MicaZ (ตัวรับ) กับ คอมพิวเตอร์ผ่านพ็อร์ตอนุกรม

- USB ทำหน้าที่ในการเชื่อมต่อพ็อร์ทอนุกรมระหว่างคอมพิวเตอร์กับ MIB5

➢ **ฐานข้อมูล**

ด้านซอฟต์แวร์

- โปรแกรม MATLAB R2009a ทำหน้าที่ในการแปลงค่าแรงดันไฟฟ้าที่วัดได้จากเซ็นเซอร์แต่ละชนิด เป็นค่าจริงของข้อมูลชนิดนั้นๆ แล้วเก็บไว้ในไฟล์ "Result.dat"

ด้านฮาร์ดแวร์

- คอมพิวเตอร์ ทำหน้าที่เป็น Server

➢ **การติดต่อกับผู้ใช้งาน**

ด้านซอฟต์แวร์

- โปรแกรมส่ง SMS ทำหน้าที่ในการเข้าไปอ่านข้อมูลใหม่จากไฟล์ "Result.dat" แล้วส่ง SMS ไปให้ User

- โปรแกรม GUI ทำหน้าที่ในการเข้าไปอ่านค่าจากในไฟล์ "Result.dat" แล้วมาแสดงในรูปของกราฟ

ด้านฮาร์ดแวร์

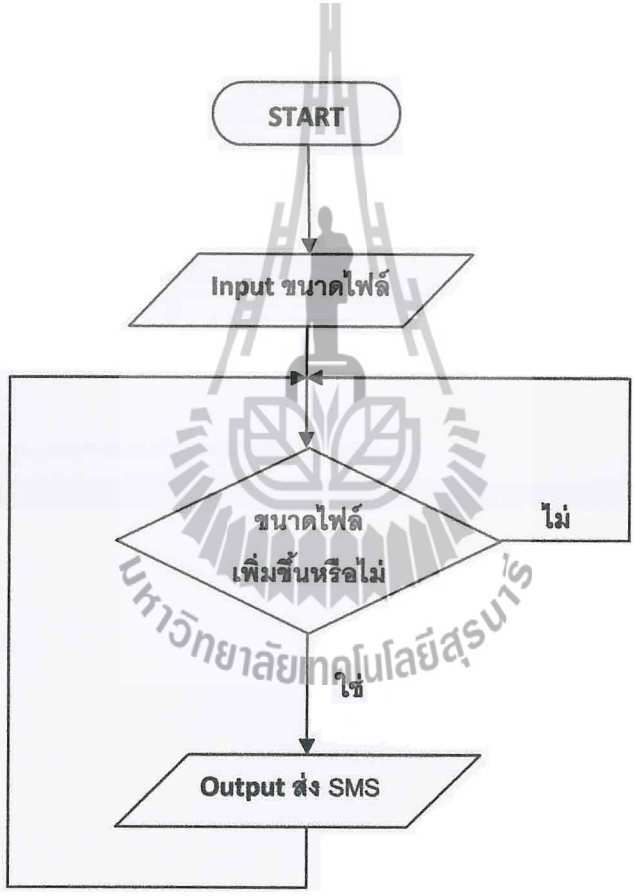
- RS-232 ทำหน้าที่ในการเชื่อมต่อระหว่าง GSM module กับ คอมพิวเตอร์

- GSM module ทำหน้าที่ในการรับคำส่งจากโปรแกรมส่ง SMS ให้ส่ง SMS ไปให้ User ผ่านเครื่อข่าย GSM

3.3 การสร้าง code โปรแกรมเพื่อส่ง SMS

3.3.1 การทำงานของโปรแกรม

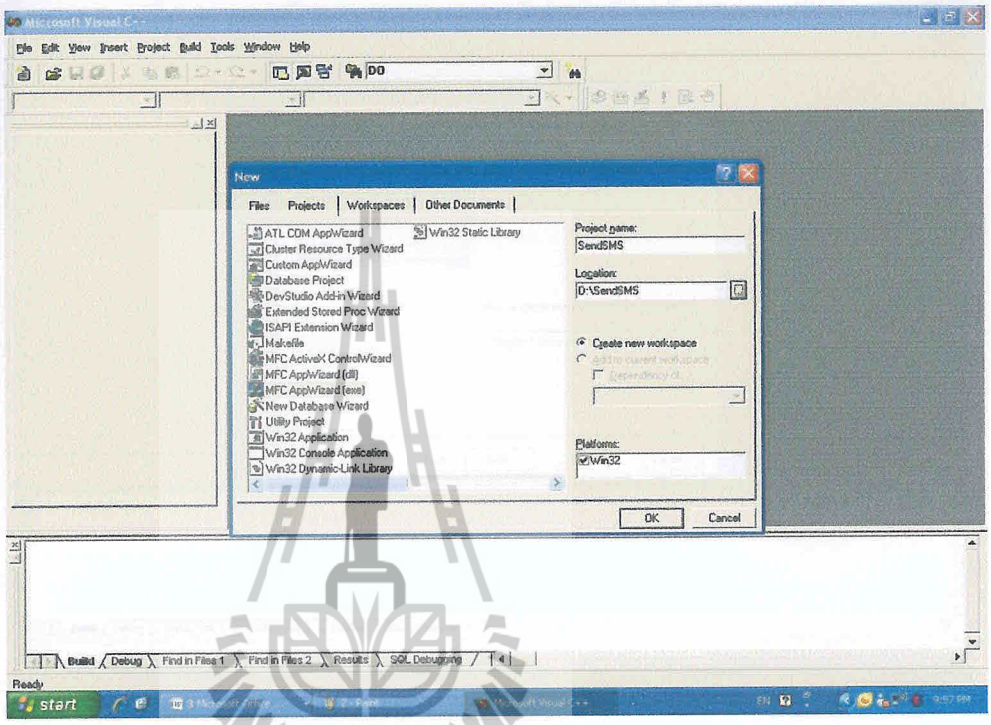
การทำงานของโปรแกรม คือ คอยตรวจเช็คข้อมูลใหม่ที่ส่งจากเซ็นเซอร์ไร้สายมายังสถานีฐาน โดยการเช็คขนาดของไฟล์หากไฟล์มีขนาดเพิ่มขึ้นก็จะทำการส่ง SMS ไปยังโทรศัพท์มือถือ ซึ่งสามารถอธิบายแผนภาพได้ดังรูปที่ 3.2 ส่วนรายละเอียดของ code โปรแกรมจะได้อธิบายในขั้นตอนถัดไป



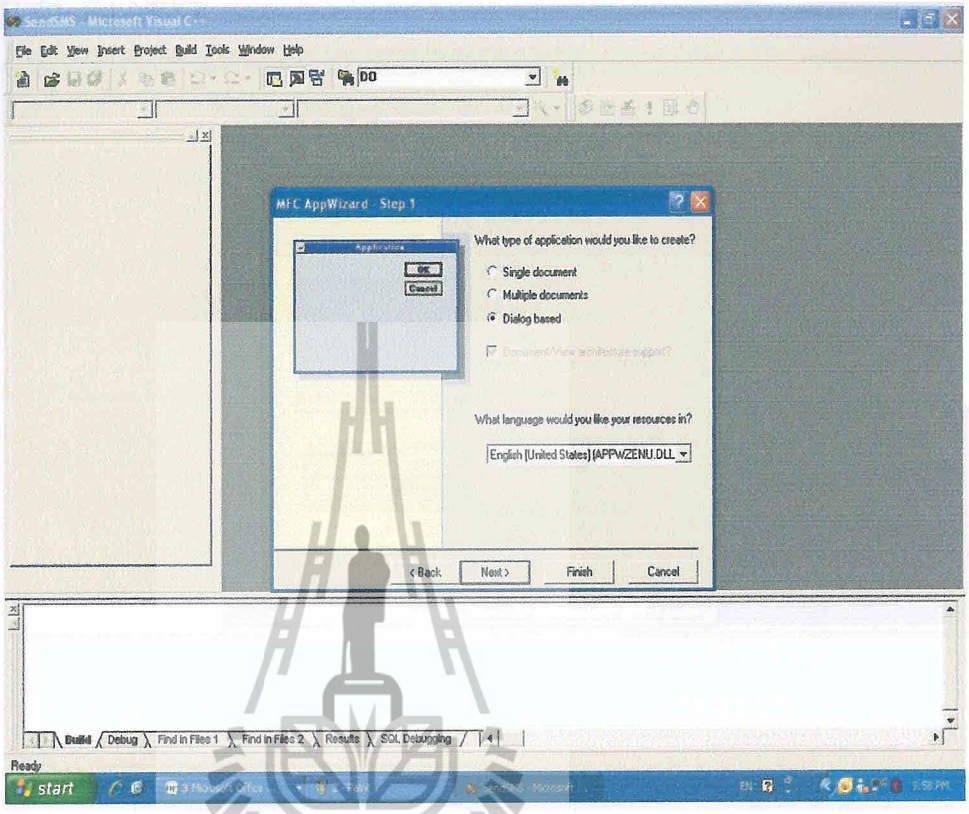
รูป 3.2 Flowchart การทำงานของโปรแกรมส่ง SMS

3.3.2 ขั้นตอนการออกแบบ

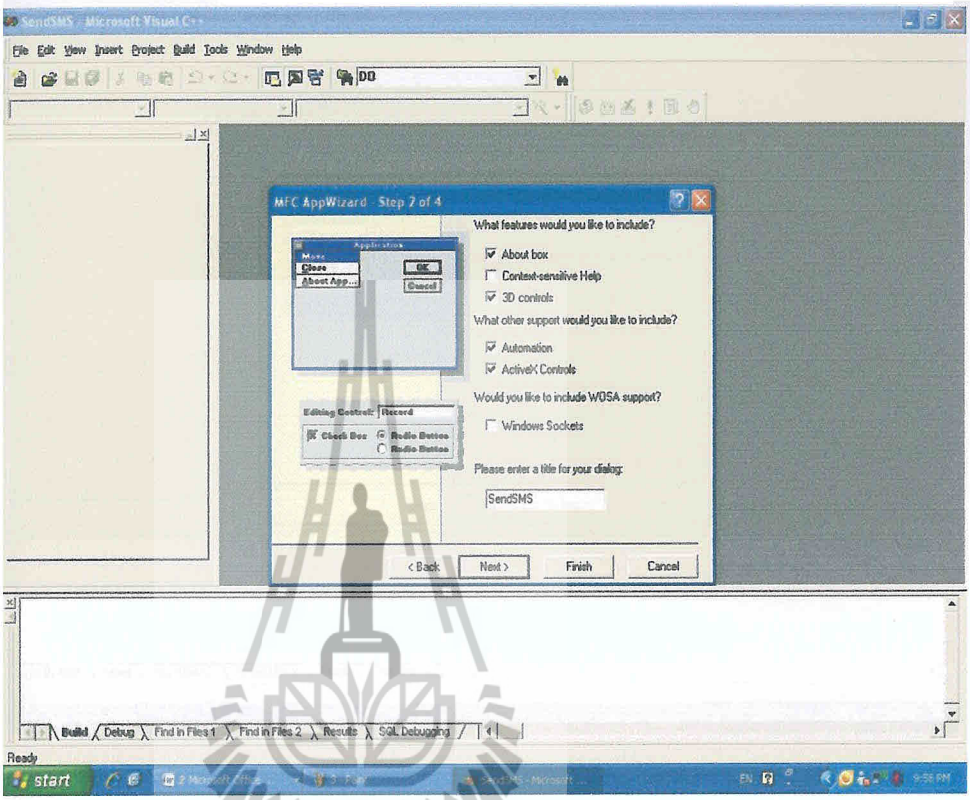
1) เปิดโปรแกรม Microsoft Visual C++ 6.0 ขึ้นมา จากนั้นไปที่ File > New > Project เลือก MFCAppWizard (exe) ตั้งชื่อโปรเจ็คใหม่ว่า "SendSMS" จากนั้น click OK ด้านล่าง



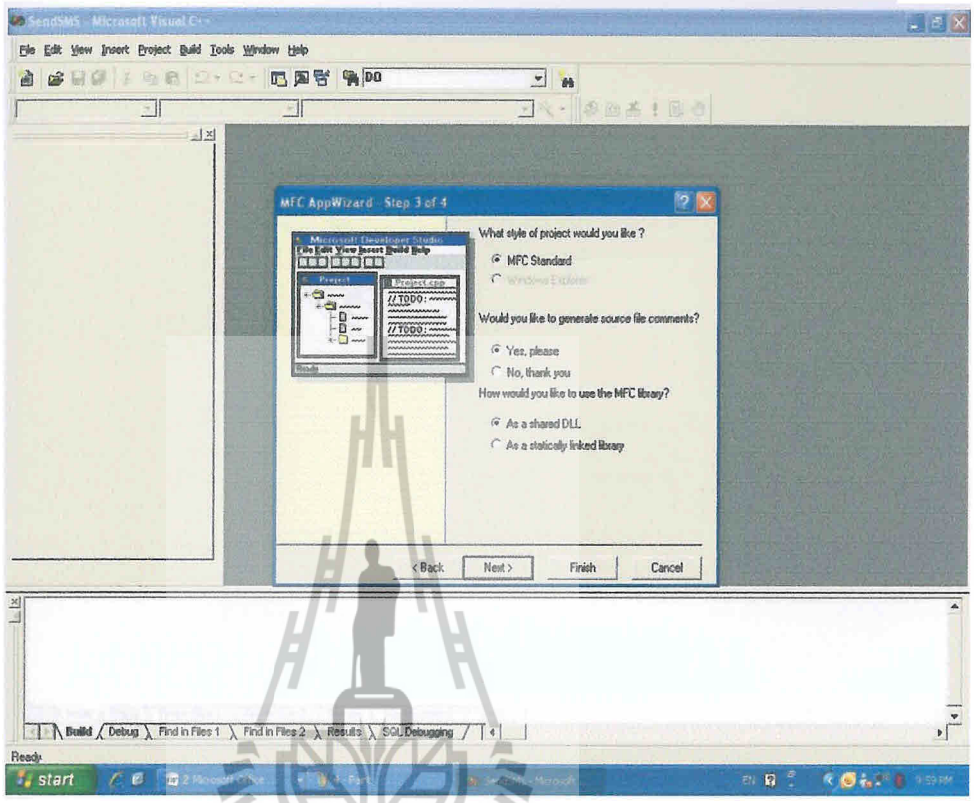
2) เลือก ชนิดของ application เป็น dialog based แล้ว click NEXT



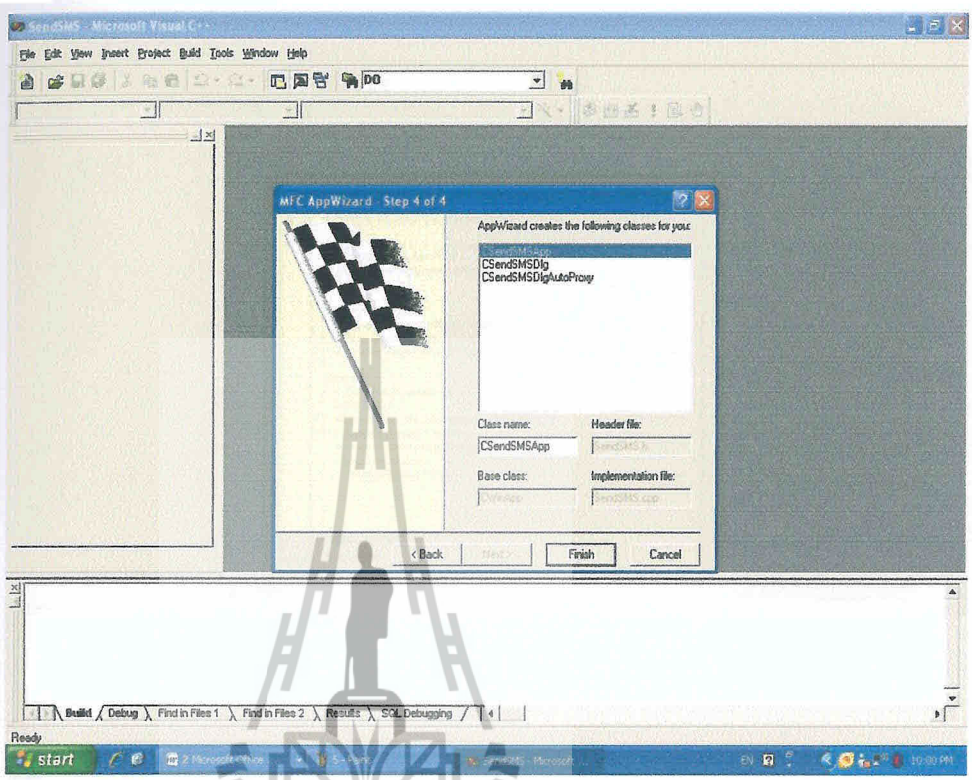
3) เลือก Automation (ถ้าลืมการดำเนินการนี้ จะไม่สามารถรัน code ใน objects ที่สร้างขึ้นได้) แล้ว click NEXT



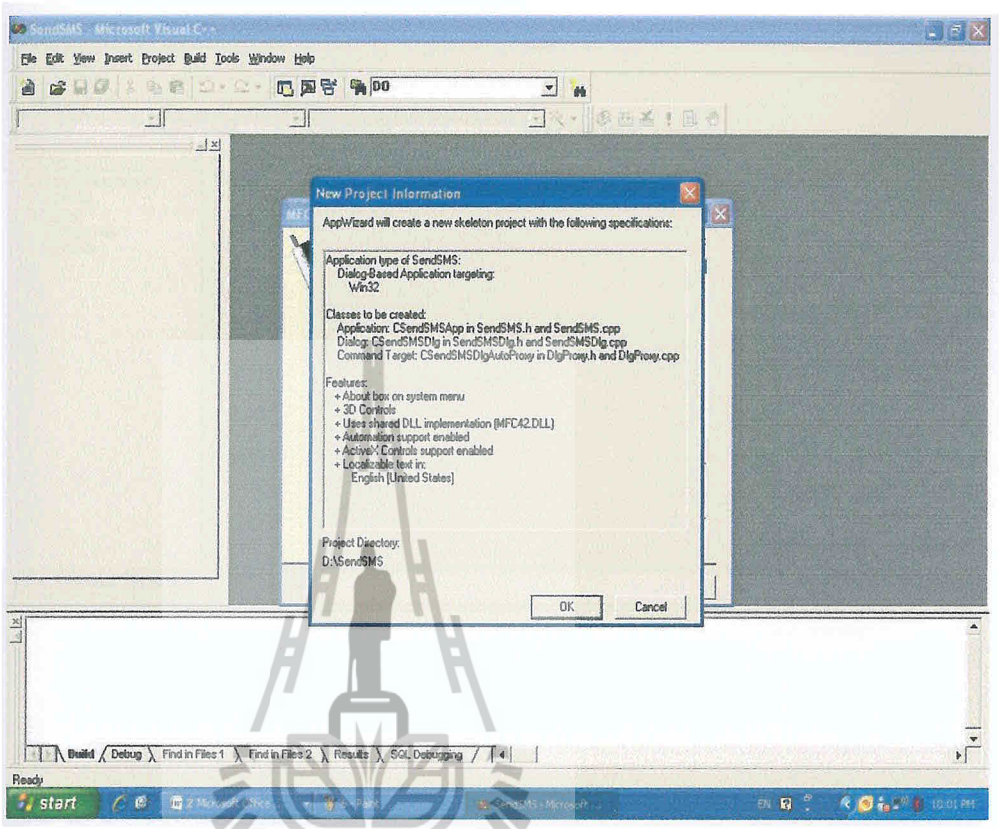
4) เลือก MFC Standard, Yes,Please และ As a Shared DLL แล้ว click NEXT



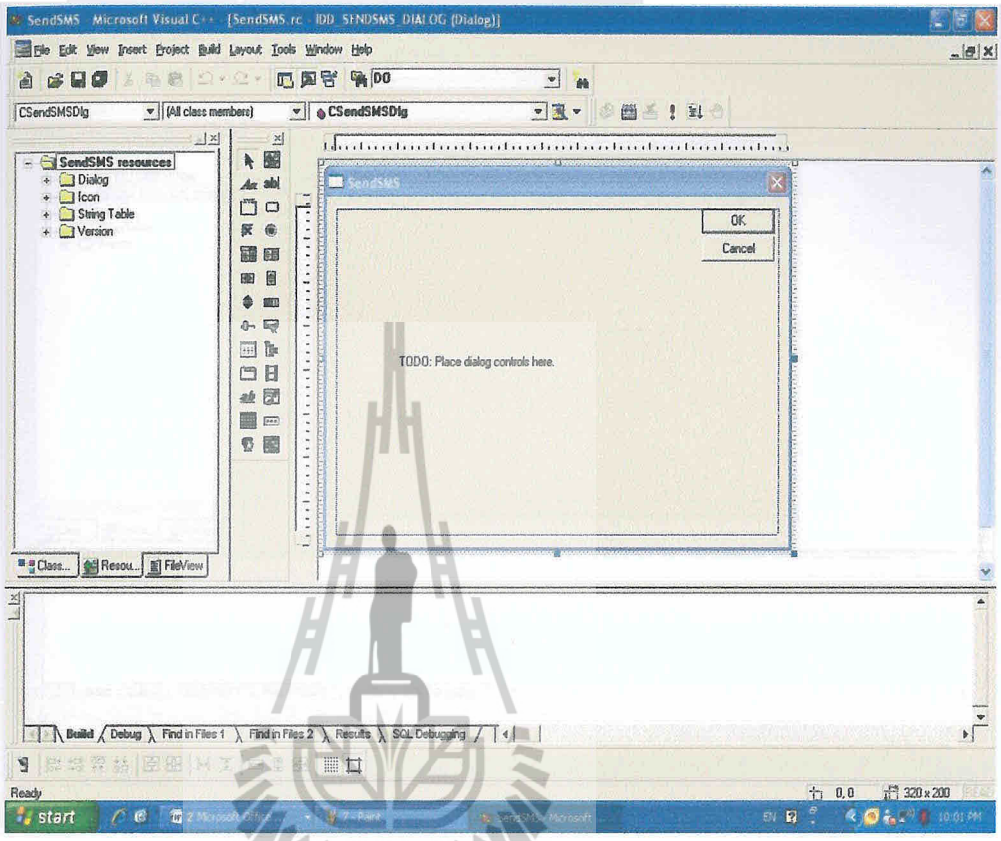
5) Click FINISH



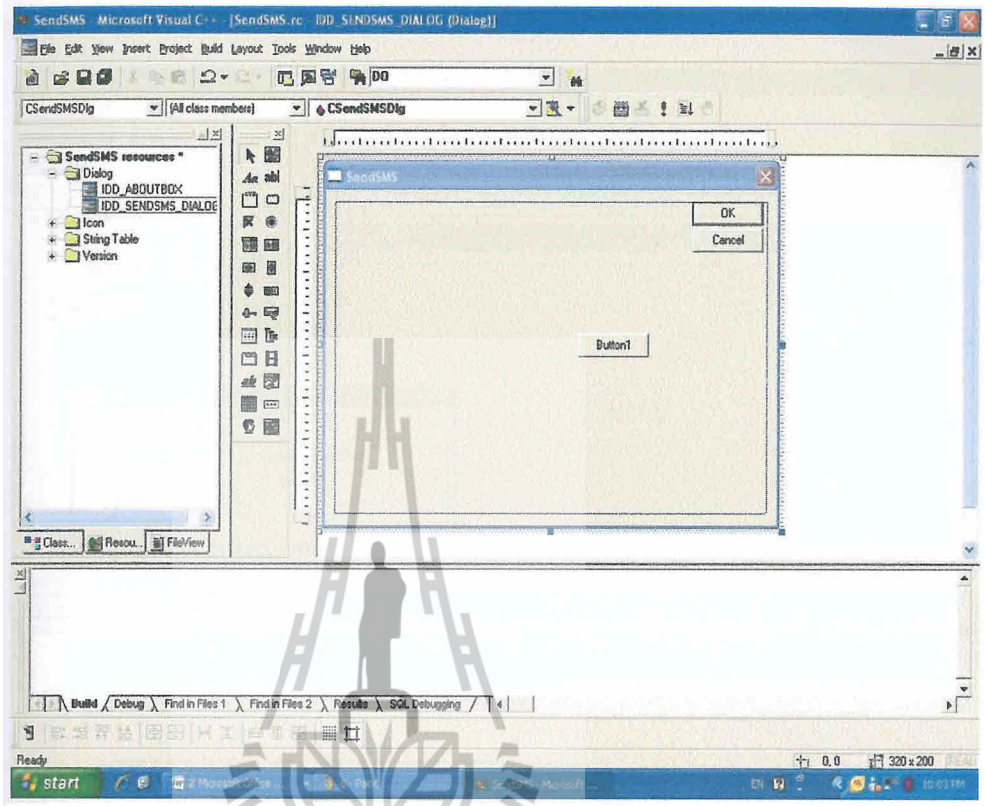
6) Click OK เพื่อนยืนยันการสร้างโปรเจ็ค



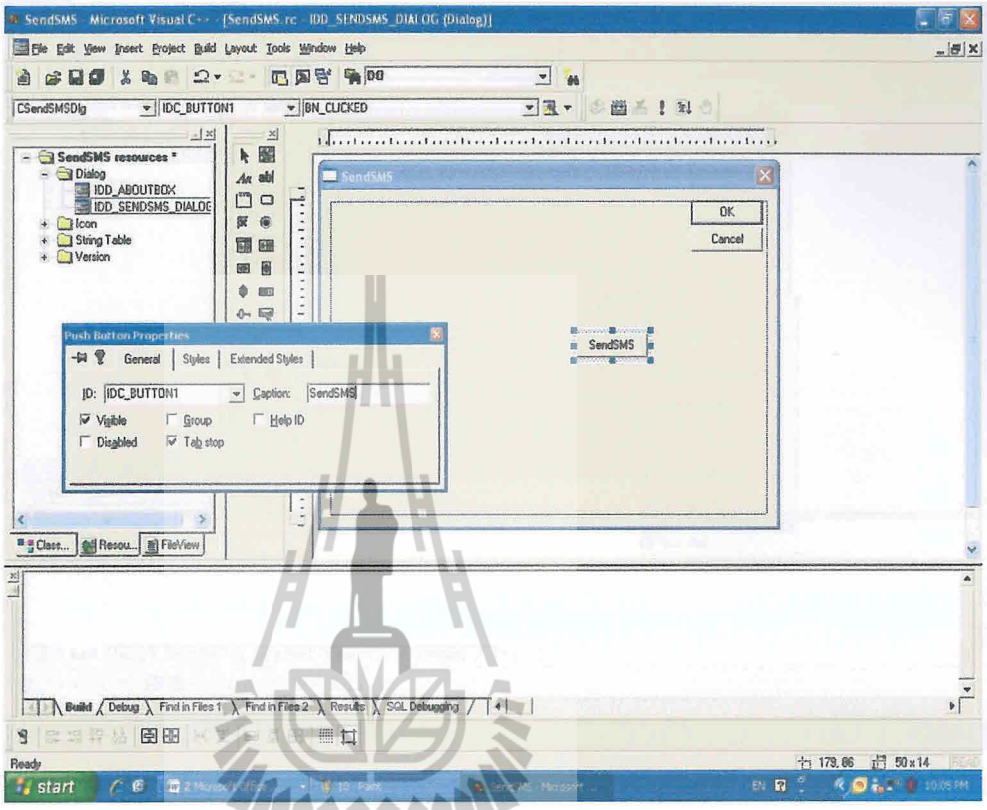
7) จะได้ dialog box สำหรับสร้างโปรแกรมส่ง SMS



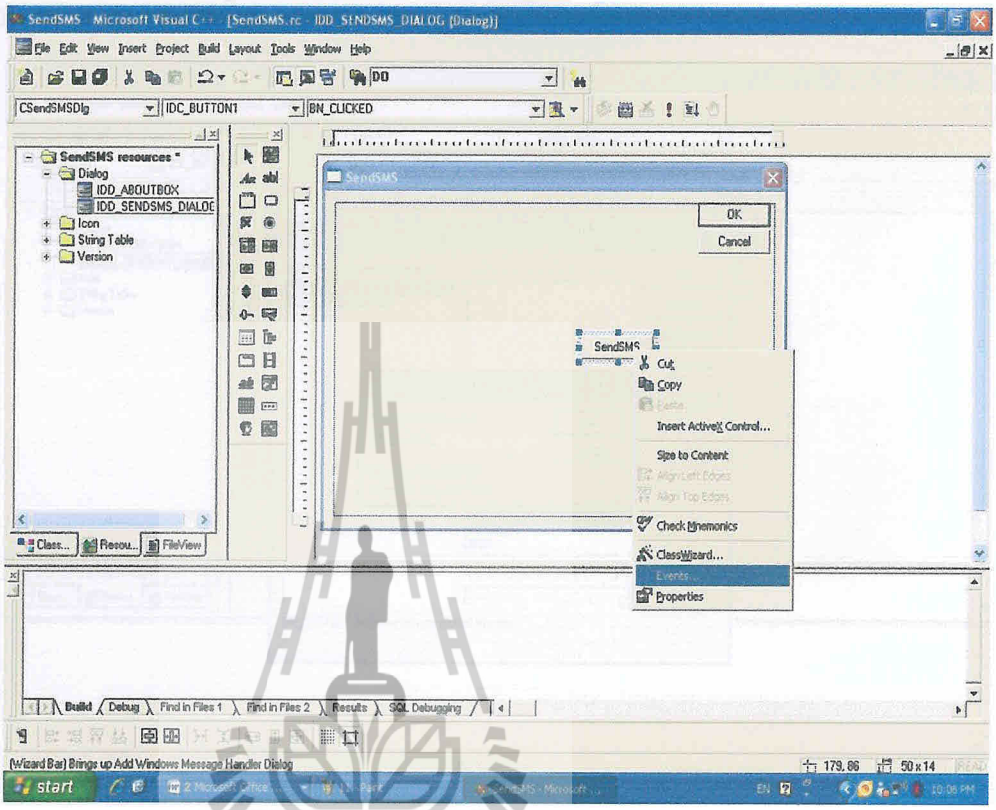
8) เลือกคอมโพเนนต์ Button จากแถบเครื่องมือ มาวางบน dialog box



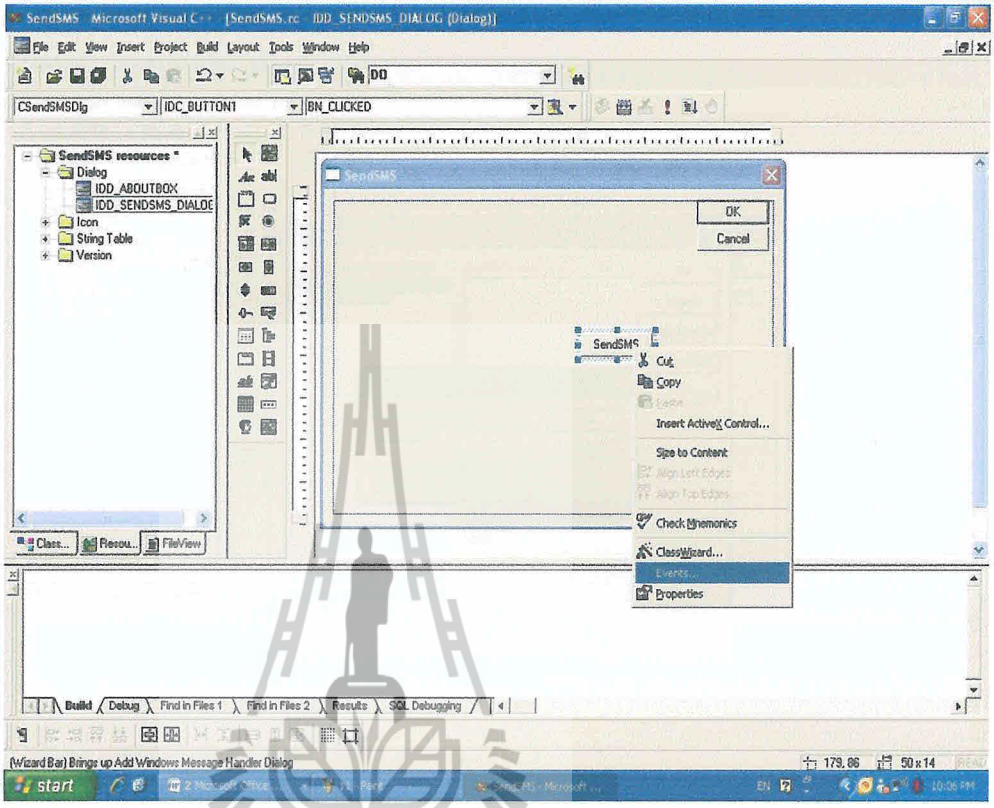
9) Click ขวาที่ Button แล้วเลือก Properties จากนั้นเปลี่ยนข้อความในส่วน Caption เป็น “SendSMS”



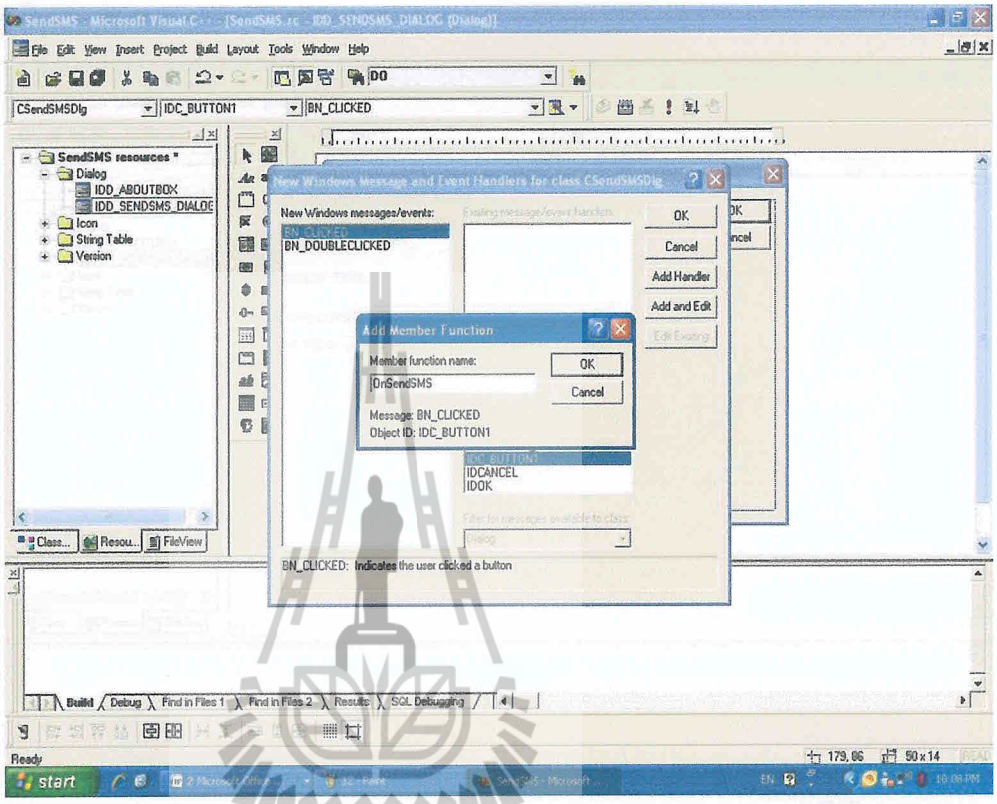
10) Click ขวาที่ Button อีกครั้ง เลือก Events...



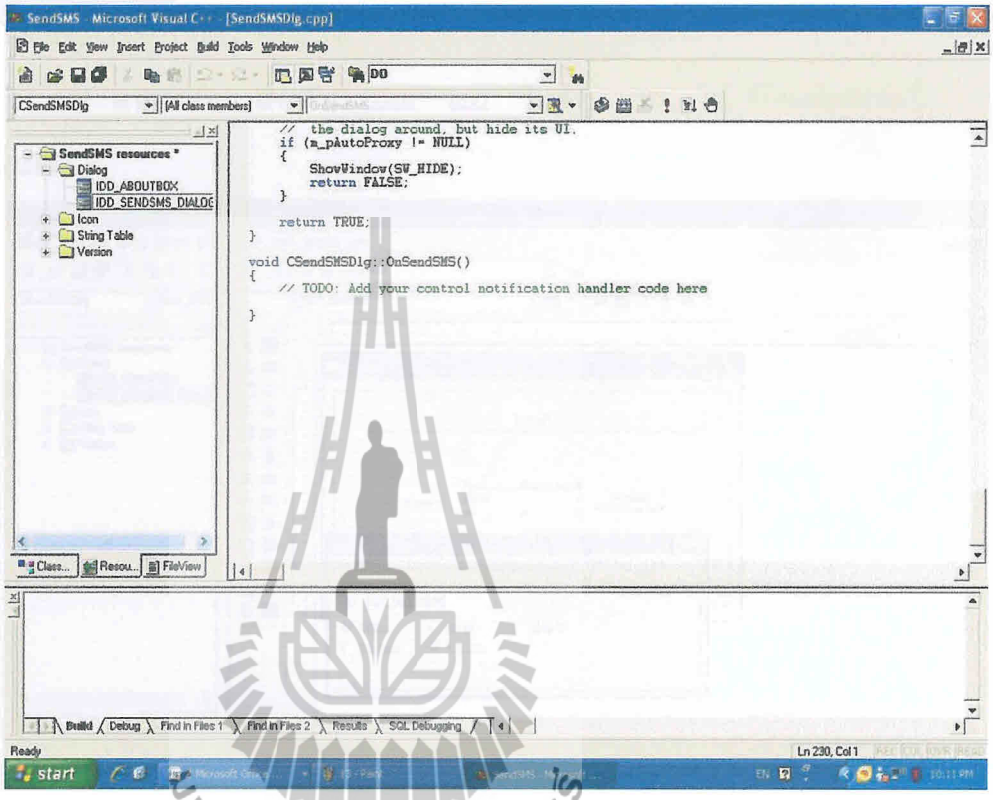
10) Click ขวาที่ Button อีกครั้ง เลือก Events...



11) เลือก “BN_CLICKED” และ “IDC_BUTTON1” จากนั้นกดปุ่ม Add Handler แล้วใส่ข้อความ “OnSendSMS” ที่ member function name และกดปุ่ม OK

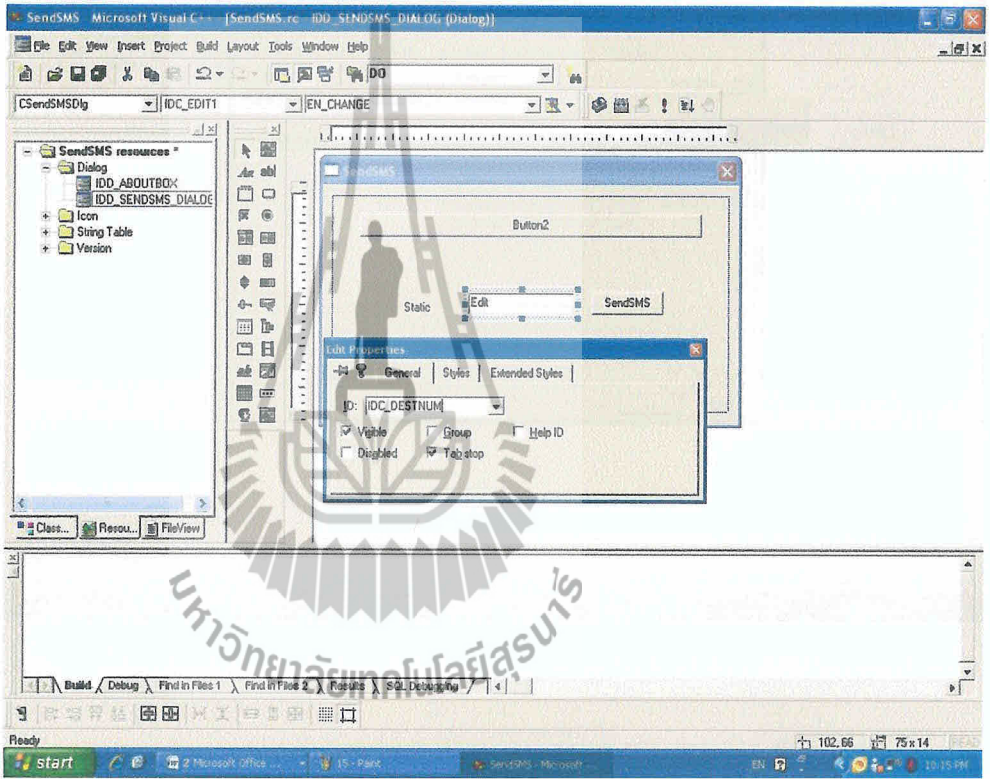


12) Dueble click ที่ Button (ชื่อ SendSMS) จะได้ฟังก์ชัน void CSendSMS :: OnSendSMS( ) สำหรับเขียน code โปรแกรมควบคุมการส่ง SMS ซึ่งรายละเอียดของ code โปรแกรมจะกล่าวถึงในขั้นตอนถัดไป

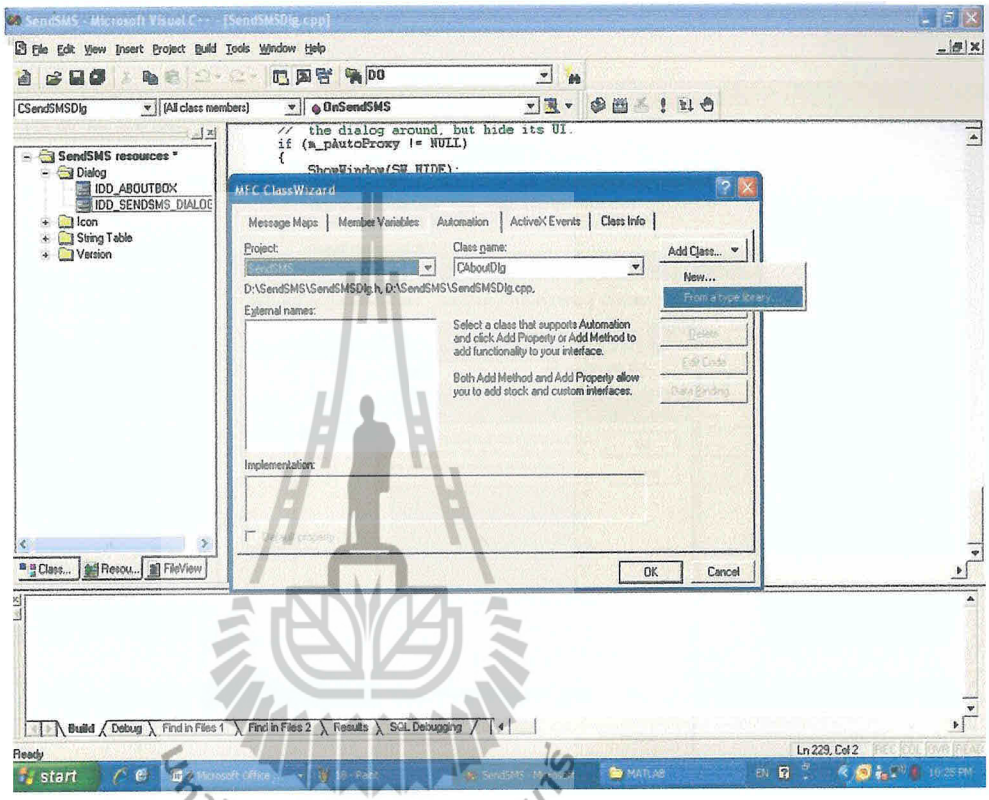


13) เพิ่มคอมโพเนนต์ Button , Static Text และ Edit text ตามลำดับจากแถบเครื่องมือลงบน dialog box แล้วกำหนดคุณสมบัติโดยการ click ขวา เลือก Properties ของคอมโพเนนต์นั้นๆ Button และ Static Text ในส่วนของ Caption ใส่ข้อความ " Pond Monitoring System Using Wireless Sensor Network and GSM " และ " Send To "

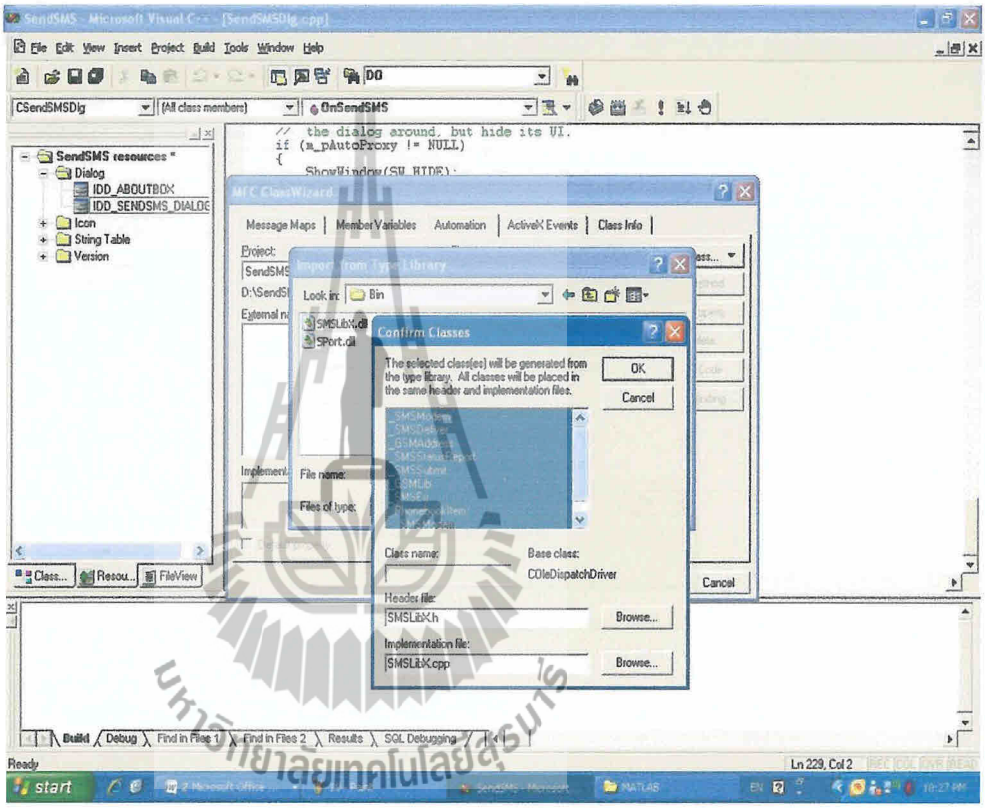
14) ตามลำดับ สำหรับ Edit text ในส่วน ID ให้เปลี่ยนข้อความเป็น " IDC_DESTNUM "



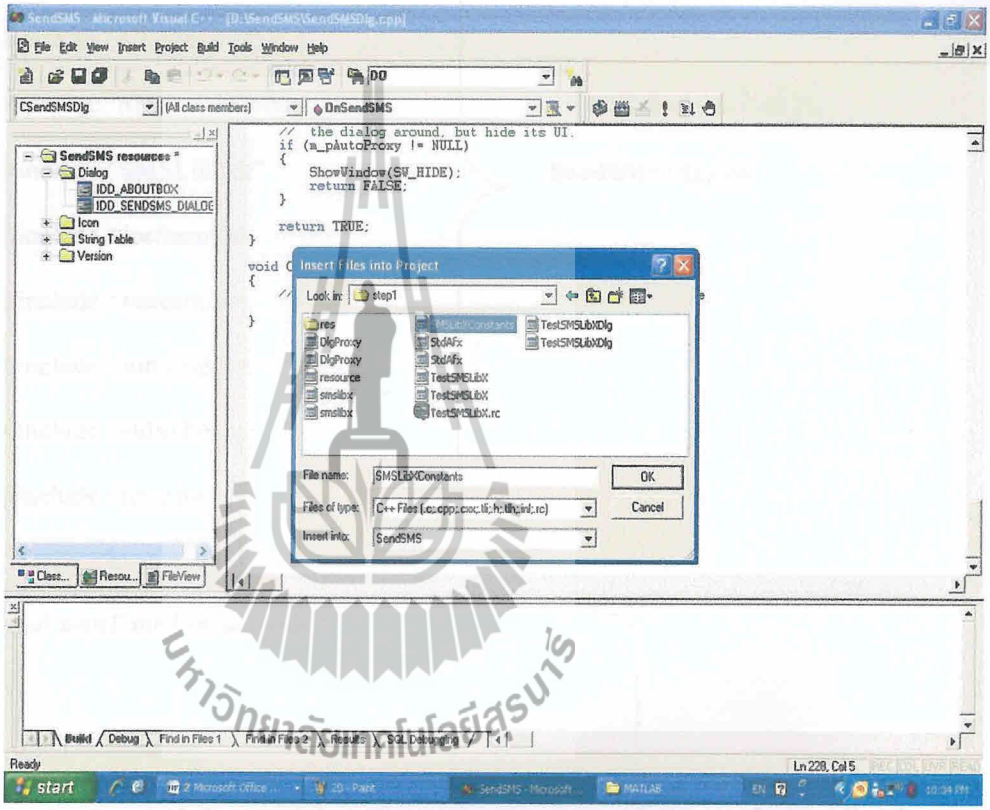
15) เป็นการเพิ่ม SMSLibX Library เข้าไปใน project โดยการเลือกที่ view > ClassWizard... > Automation > Add Class > From type library…



16) ทำการ Browse ไฟล์ SMSLibX.dll จากไฟล์การติดตั้ง SMSLibX Library ในคอมพิวเตอร์ จากนั้น กด OK สำหรับการเลือกไฟล์ SMSLibX.dll และกด OK อีกครั้งเพื่อยืนยันการเลือกทุก classes



17) ทำการเพิ่ม header file ชื่อ “ SMSLibXConstants.h”โดยไปที่ Project > Add To Project > Files ... (ไฟล์นี้สามารถหาได้จาก tutorial file zip ที่โหลดมาจากเว็บไซต์ http://www.smsco.it/tomcat/en/sms/smslibx.jsp) และทำการคัดลอกไฟล์ “ SMSLibXConstants.h ” ไปไว้ในโฟลเดอร์ Project SendSMS ด้วย



18) Code โปรแกรม

```
#include "stdafx.h"
#include "SendSMS.h"
#include "SendSMSDlg.h"
#include "DlgProxy.h"
#include "SMSLibXConstants.h"
#include "SMSLibX.h"
#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>
#include <stdio.h>
#include <time.h>

void waitTime ( int seconds )
{
 clock_t endwait;
 endwait = clock () + seconds * CLOCKS_PER_SEC ;
 while (clock() < endwait) {}
}
```

ประกาศ include ในส่วน SendSMSDlg.cpp

ประกาศฟังก์ชัน void ชื่อ waitTime

```
void CTestSMSLibX3Dlg::OnSendSMS()
{
     CString oper;
     char str[128];
     memset(str,0,128);
```

ประกาศตัวแปร

เช็คการเชื่อมต่อ Module

```
_SMSModem modem;
     if( !modem.CreateDispatch("SMSLibX.SMSModem") ) {
               AfxMessageBox("Cannot link to SMSLibX.SMSModem object: SMSLibX
library may be not registered!");
               return;
     }
```

เลือกชนิด Module ที่ใช้

```
long modemType =
SMSLibXConstants::GSMModemTypeConstants::gsmModemWavecom;
short comPort = 9;
```

ตั้งค่า port ของ module

```
CString smscNumber = "";
```

ประกาศตัวแปรชนิด CString

เปิด / ปิดการใช้งานการแจ้งเตือนข้อความที่เข้ามา

```
long notifyMode =
SMSLibXConstants::SMSModemNotificationConstants::smsNotifyNone;
```

```
bool unregisteredMode = false;          ลงทะเบียนเครื่อข่าย GSM

short timeout = 60;                     ตั้งค่าเวลาสำหรับการ startup module ( 60 วินาที )
```

รอ event (click)จากปุ่ม Button ชื่อ SendSMS

```
CWaitCursor wait;

        try {

        FILE *fp;           ประกาศตัวแปรชนิด pointer

        int n;
        int size1=0;
        int size2=0;
        int year1,year;                 ประกาศตวแปรชนด integer
        int month1,month;
        int day1,day;
        int hour1,hour;
        int min1,min;
        double temp1,temp;
        double pH1,pH;                  ประกาศตัวแปรชนิด double
        double DO1,DO;
```

เปิดการสื่อสาร Module

```
oper = "Opening modem communication";
modem.OpenComm(modemType, comPort, smscNumber, notifyMode, timeout,
unregisteredMode);
oper = "Creating SMS message"
CString destNumber;          ประกาศตัวแปรชนิด Cstring
GetDlgItemText(IDC_DESTNUM, destNumber);  รับค่าจาก Keybord มาเก็บไว้ในตัว
```

แปร destNumber

<u>วน Loop เช็คขนาดไฟล์</u>

```
for (n=1; n>0; n++)
    {
        fp = fopen("Result.dat","rt");      เปิดไฟล์เพื่อเช็คขนาดไฟล์
        fseek(fp,0,SEEK_END);               เช็คขนาดไฟล์ตั้งแต่ต้นไฟล์จนถึงท้ายไฟล์
        size1 = ftell(fp);                  เก็บค่าขนาดไฟล์ใน ตัวแปร  size1
        fclose(fp);                         ปิดไฟล์
```

<u>เช็คขนาดไฟล์ว่ามีขนาดเพิ่มขึ้นหรือไม่</u>

```
if (size1>size2) {
        ifstream inClientFile("Result.dat",ios::in);  เปิดไฟล์เพื่อนอ่านค่าในไฟล์
```

อ่านค่าในไฟล์ที่ละบรรทัดจนถึงท้ายไฟล์

```
while(inClientFile>>year1>>month1>>day1>>hour1>>min1>>temp1>>pH1>>DO1)
        {
                year = year1;
                month = month1;
                day = day1;
                hour = hour1;
                min = min1;
                temp = temp1;
                pH = pH1;
                DO = DO1;
        }
```

เก็บค่าจากไฟล์

แปลงค่าตัวแปรเป็นตัว แปรชนิด Csting

```
CString stryear,strmonth,strday,strhour,strmin,strtemp,strpH,strDO;
stryear.Format("%d", year);
strmonth.Format("%d", month);

strday.Format("%d", day);
strhour.Format("%d", hour);

strmin.Format("%d", min);
strtemp.Format("%.3f", temp);
strpH.Format("%.3f", pH);
strDO.Format("%.3f", DO);
```

จัดรูปแบบข้อความและส่ง SMS ไปยังหมายเลขปลายที

```
        oper = "Sending SMS message";

        CString msgTxt ="Date: "+stryear+"-"+strmonth+"-"+strday+"\nTime
"+strhour+":"+strmin+"\nTemp: "+strtemp+"\npH: "+strpH+"\nDO: "+strDO+"";
        int msgId = modem.SendTextMessage(destNumber, msgTxt, false);
        size2=size1;
}
```

วน Loop save "Hello" หากขนาดไฟล์เท่าเดิม

```
else {

        ofstream inClientFile11("testt.text",ios::out);

        CString msgTxt1 = "Hello\t";
        inClientFile11 << msgTxt1;
        inClientFile11.close();
}
waitTime (5);     รอเวลาทุก 5 วินาที แล้วเช็คขนาดไฟล์
}

}
```

```
catch (CException* e) {
        e->GetErrorMessage(str, 128);
        AfxMessageBox("Failure on "+oper+":\n"+str);
        e->Delete();
    }
try{ modem.CloseComm(); }catch(...){;}
    modem.ReleaseDispatch();
}
```

เช็ค errorในการส่ง SMS

ปิด Module

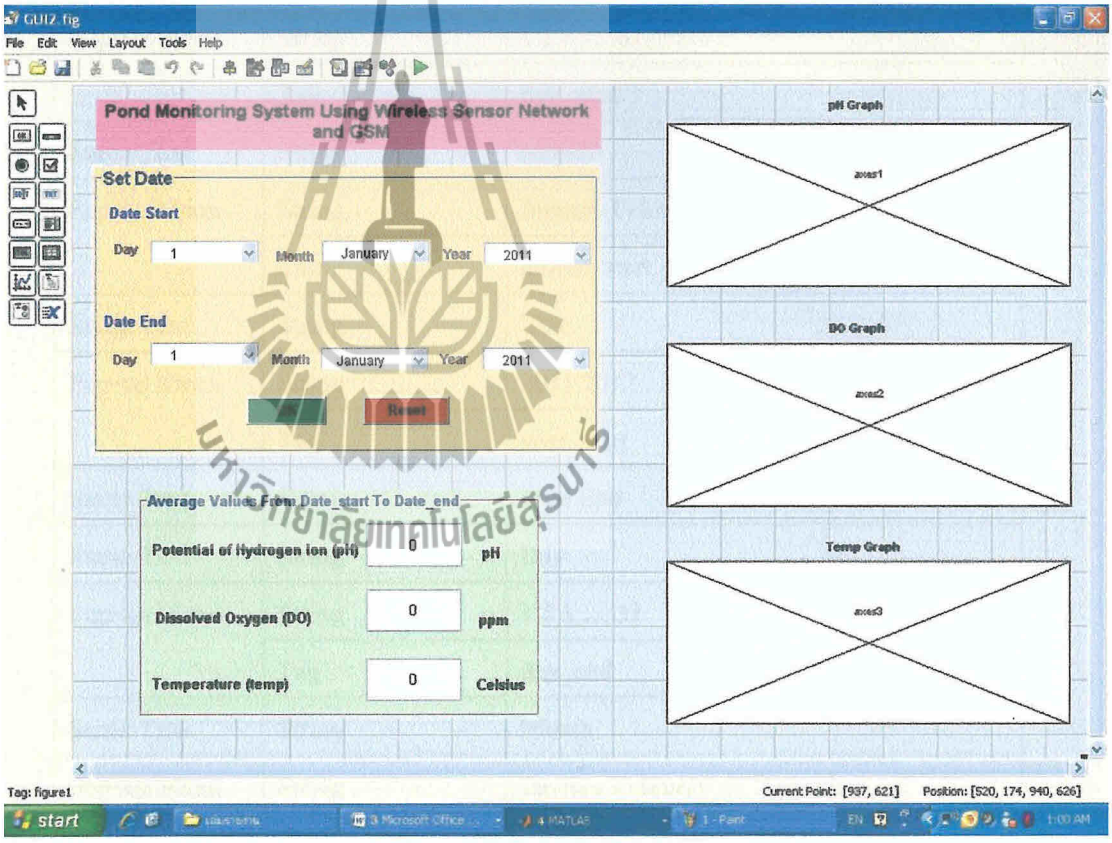
3.4 การออกแบบโปรแกรม GUI (Graphic User Interface)

3.4.1 ขั้นตอนการออกแบบ

1) เปิดโปรแกรม MATLAB R2009a ขึ้นมา แล้วพิมพ์ " Guide " ที่ Command Window จะได้ GUIDE Quick Start ขึ้นมา

2) ไปที่ Create New GUI > Blank GUI (Default) แล้ว click OK

3) เลือกคอมโพเนนต์ 18 Static Text, 6 Pop-up Menu, 2 Push Button, 2 Panel และ 3Axes จากแถบเครื่องมือด้านซ้าย แล้วจัดว่างคอมโพเนนต์ดังภาพข้างล่าง



4) กำหนดค่าพร็อพเพอร์ตี้ให้กับคอมโพเนนต์ต่างๆ ดังตาราง

ตาราง 3.1 กำหนดค่าพร็อพเพอร์ตี้ให้กับคอมโพเนนต์

คอมโพเนนต์	พร็อพเพอร์ตี้	ค่าที่กำหนด
Static Text	String	Pond Monitoring System Using Wireless Sensor Network and GSM
Panel	String	Set Date
Static Text	String	Date Start
Static Text	String	Day
Pop-up Menu	String	1 2 3 … 31
	Tag	day_start
Static Text	String	Month
Pop-up Menu	String	January February … December
	Tag	month_start
Static Text	String	Year
Pop-up Menu	String	2011 2012 …
	Tag	year_start
Static Text	String	Date End
Static Text	String	Day
Pop-up Menu	String	1 2 3 … 31
	Tag	day_end
Static Text	String	Month
Pop-up Menu	String	January February … December
	Tag	month_end
Static Text	String	Year
Pop-up Menu	String	2011 2012 …
	Tag	year_end

คอมโพเนนต์	พร็อพเพอร์ตี้	ค่าที่กำหนด
Push Button	String	OK
	Tag	OK
Push Button	String	Reset
	Tag	Reset
Panel	String	Average Values From Date_start To Date_end
Static Text	String	Potential of Hydrogen ion (pH)
Edit Text	String	0
	Tag	average_PH
Static Text	String	pH
Static Text	String	Dissolved Oxygen (DO)
Edit Text	String	0
	Tag	average_DO
Static Text	String	ppm
Static Text	String	Temperature (temp)
Edit Text	String	0
	Tag	average_temp
Static Text	String	Celsius
Static Text	String	pH Graph
Axes	Tag	axes1
Static Text	String	DO Graph
Axes	Tag	axes2
Static Text	String	Temp Graph
Axes	Tag	axes3

* **หมายเหตุ** หากต้องการเปลี่ยนสี backgroung และตัวอักษร ของคอมโพเนนต์ก็สามารถทำได้ที่ Property Inspector > BackgroundColor และ Property Inspector > ForegroundColor ตามลำดับ

5) save ไฟล์ (ชื่อไฟล์).fig แล้ว ไปที่ M-file Editor จะได้ function ของคอมโพเนนต์ต่างๆตามที่ได้จัดรูปแบบไว้แล้วข้างต้น

6) เปลี่ยนฟังก์ชันเหล่านี้

```
function day_start_Callback(hObject, eventdata, handles)
function month_start_Callback(hObject, eventdata, handles)
function year_start_Callback(hObject, eventdata, handles)
function day_end_Callback(hObject, eventdata, handles)
function month_end_Callback(hObject, eventdata, handles)
function year_end_Callback(hObject, eventdata, handles)
```

เป็น

```
function [day_start] = day_start_Callback(hObject, eventdata, handles)
function [month_start] = month_start_Callback(hObject, eventdata, handles
function [year_start] = year_start_Callback(hObject, eventdata, handles)
function [day_end] = day_end_Callback(hObject, eventdata, handles)
function [month_end] = month_end_Callback(hObject, eventdata, handles)
function [year_end]= year_end_Callback(hObject, eventdata, handles)
```

ตามลำดับ เพื่อเก็บค่าจากการรับค่าจากภายนอกของ ฟังก์ชัน เหล่านี้ไว้ในตัวแปรแล้วส่งค่าไปใช้ในฟังก์ชัน อื่น

7) เขียน code โปรแกรม

ฟังก์ชัน day_start

```
function [day_start] = day_start_Callback(hObject, eventdata, handles)
day_start=get(handles.day_start, 'Value');
```

รับค่าเป็นตัวเลขมาเก็บไว้ในตัวแปร day_start

ฟังก์ชัน month_start

```
function [month_start] = month_start_Callback(hObject, eventdata, handles)
month_start=get(handles.month_start, 'Value');
```
รับค่าเป็นตัวเลขมาเก็บไว้ในตัวแปร month_start

ฟังก์ชัน year_start

```
function [year_start] = year_start_Callback(hObject, eventdata, handles)
year_start1=get(handles.year_start, 'Value');
```
รับค่าเป็นตัวเลขมาเก็บไว้ในตัวแปร year_start1

```
year_start=2010+year_start1;
```

ฟังก์ชัน day_end

```
function [day_end] =  day_end_Callback(hObject, eventdata, handles)
day_end=get(handles.day_end, 'Value');
```
รับค่าเป็นตัวเลขมาเก็บไว้ในตัวแปร day_end

ฟังก์ชัน month_end

```
function [month_end] = month_end_Callback(hObject, eventdata, handles)
month_end=get(handles.day_end, 'Value');
```
รับค่าเป็นตัวเลขมาเก็บไว้ในตัวแปร month_end

ฟังก์ชัน year_end

```
function [year_end]= year_end_Callback(hObject, eventdata, handles)
year_end1=get(handles.year_start, 'Value');
```
รับค่าเป็นตัวเลขมาเก็บไว้ในตัวแปร year_end1

```
year_end=2010+year_end1;
```

*** หมายเหตุ** ในฟังก์ชัน year_start และ year_end การที่ year_start=2010+year_start1 และ year_end=2010+year_end1 เพราะว่าค่า Value ที่รับมาจาภายนอกจะเรียงลำดับตามจำนวนนับ คือ 1,2,…,n เช่น หากกำหนด 2011 เป็นลำดับที่ 1 ในพร็อพเพอร์ตี้ String ของคอมโพเนนต์ Pop-up Menu จึงจำเป็นต้องบวกค่าคงที่ 2010 กับค่าในตัวแปร year_start1 และ year_end1 ตามลำดับ เพื่อให้ได้ค่า 2011 ตามต้องการ เป็นต้น

ฟังก์ชัน OK

```
function OK_Callback(hObject, eventdata, handles)
day_start = day_start_Callback(hObject, eventdata, handles);
month_start = month_start_Callback(hObject, eventdata, handles);
year_start = year_start_Callback(hObject, eventdata, handles);
day_end = day_end_Callback(hObject, eventdata, handles);
month_end = month_end_Callback(hObject, eventdata, handles) ;
year_end = year_end_Callback(hObject, eventdata, handles) ;
```

รับค่าตัวแปรจาก ฟังก์ชัน day_start, month_start, year_start, day_end, month_end และ year_end

```
flag_start = false;
flag_end = false;
flag_end1 = false;
sum_PH = 0;
sum_DO = 0;
sum_temp = 0;
count_PH = 0;
count_DO = 0;
count_temp = 0;
count_YS = 0;
```

ประกาศตัวแปร


```
fid_q=fopen('Result.dat','r');
```

เปิดไฟล์

วน loop อ่านค่าจากไฟล์ตาม วัน เดือน และปี ที่กำหนด

```
while flag_end~=true
[num,count]=fscanf(fid_q,'%8f', [1,8]);
```

```
if((num(1,1)==year_start)&&(num(1,2)==month_start)&&(num(1,3)==day_start))
        flag_start=true;
end

if((flag_start==true)&&(flag_end==false))

        if num(1,1)
                count_YS = count_YS+1;
                year(count_YS)=num(1,1);
                month(count_YS)=num(1,2);
                day(count_YS)=num(1,3);
                hour(count_YS)=num(1,4);
                mn(count_YS)=num(1,5);
                sec(count_YS)=0;
        end
        if num(1,6)
                sum_temp = sum_temp + num(1,6);
                count_temp = count_temp + 1;
                temp( count_temp ) = num(1,6);
        end

        if num(1,7)
                sum_PH = sum_PH + num(1,7);
                count_PH = count_PH + 1;
                PH( count_PH ) = num(1,7);

        end
```

```
            if num(1,8)
                sum_DO = sum_DO + num(1,8);
                count_DO = count_DO + 1;
                DO( count_DO ) = num(1,8);
            end
        end

        if((num(1,1)==year_end)&&(num(1,2)==month_end)&&(num(1,3)==day_end))
            flag_end1=true;
        end

if(((num(1,1)~=year_end)||(num(1,2)~=month_end)||(num(1,3)~=day_end))&&(flag_end1==true))
        flag_end=true;
    end
    end
```

ค่าเฉลี่ยของ temp, pH และ DOตามวัน เดือน และปี ที่กำหนด

```
avg_temp = (sum_temp-num(1,6))/(count_temp-1);
avg_PH = (sum_PH-num(1,7))/(count_PH-1);
avg_DO  = (sum_DO-num(1,8))/(count_DO-1);
```

นำค่าจากในไฟล์ไปพล็อตกราฟ

```
xdate = datenum(year,month,day,hour,mn,sec);
```

พล็อตค่า pH ใน axes1

```
axes(handles.axes1);
if (day_start==day_end && month_start==month_end && year_start==year_end)
        s = plot(xdate(1:count_PH-1) , PH(1:count_PH-1) ,'-b*');
        datetick('x',15,'keeplimits')
        grid on;
        hold on;
        xlabel('time'),ylabel('pH value (pH)');
elseif (year_start~=year_end)
        s = plot(xdate(1:count_PH-1) , PH(1:count_PH-1) ,'-b*');
        datetick('x','mmm-yy','keeplimits')
        grid on;
        hold on;
        xlabel('time'),ylabel('pH value (pH)');
else
        s = plot(xdate(1:count_PH-1) , PH(1:count_PH-1) ,'-b*');
        datetick('x',19,'keeplimits')
        grid on;
        hold on;
        xlabel('time'),ylabel('pH value (pH)');
end
```

พล็อตค่า DO ใน axes2

```
axes(handles.axes2);
if (day_start==day_end && month_start==month_end && year_start==year_end)
        s = plot(xdate(1:count_DO-1) , DO(1:count_DO-1) ,'-g*');
        datetick('x',15,'keeplimits')
        grid on;
        hold on;
        xlabel('time'),ylabel('DO value (ppm)');
elseif (year_start~=year_end)
        s = plot(xdate(1:count_DO-1) , DO(1:count_DO-1) ,'-g*');
        datetick('x','mmm-yy','keeplimits')
        grid on;
        hold on;
        xlabel('time'),ylabel('DO value (ppm)');
else
        s = plot(xdate(1:count_DO-1) , DO(1:count_DO-1) ,'-g*');
        datetick('x',19,'keeplimits')
        grid on;
        hold on;
        xlabel('time'),ylabel('DO value (ppm)');
end
```

พล็อตค่า Temp ใน axes3

```
axes(handles.axes3);
if (day_start==day_end && month_start==month_end && year_start==year_end)
        s = plot(xdate(1:count_temp-1) , temp(1:count_temp-1) ,'-r*');
        datetick('x',15,'keeplimits')
        grid on;
        hold on;
        xlabel('time'),ylabel('temp value (°C)');
elseif (year_start~=year_end)
        s = plot(xdate(1:count_temp-1) , temp(1:count_temp-1) ,'-r*');
        datetick('x','mmm-yy','keeplimits')
        grid on;
        hold on;
        xlabel('time'),ylabel('temp value (°C)');
else
        s = plot(xdate(1:count_temp-1) , temp(1:count_temp-1) ,'-r*');
        datetick('x',19,'keeplimits')
        grid on;
        hold on;
        xlabel('time'),ylabel('temp value (°C)');
end
```

```
set(handles.average_PH,'String', avg_PH);
set(handles.average_DO,'String', avg_DO);
set(handles.average_temp,'String', avg_temp);
```

ส่งค่าเฉลี่ยไปแสดงที่หน้าจอ GUI ในส่วนของ Average Values From Date_start To Date_end

ฟังก์ชัน Reset

```
function Reset_Callback(hObject, eventdata, handles)
initialize_gui(gcbf, handles, true);          เรียกฟังก์ชันชื่อ initialize_gui
clear_axes(gcbf, handles, true);   เรียกฟังก์ชันชื่อ clear_axes
```

สร้างฟังก์ชันขึ้นมาใหม่ ชื่อ initialize_gui และ clear_axes

ฟังก์ชัน initialize_gui

```
function initialize_gui(fig_handle, handles, isreset)
if isfield(handles, 'metricdata') && ~isreset
return;
end
        set ค่าไปที่ค่า เริ่มต้นการเปิดโปรแกรม
        handles.metricdata.average_PH = 0;
        handles.metricdata.average_DO = 0;
        handles.metricdata.average_temp = 0;
        handles.metricdata.day_end = 1;
        set(handles.average_PH, 'String', handles.metricdata.average_PH);
        set(handles.average_DO, 'String', handles.metricdata.average_DO);
        set(handles.average_temp, 'String', handles.metricdata.average_temp);
set(handles.year_start, 'Value',handles.metricdata.day_end);
        set(handles.month_start, 'Value',handles.metricdata.day_end);
        set(handles.day_start, 'Value',handles.metricdata.day_end);
        set(handles.year_end, 'Value',handles.metricdata.day_end);
        set(handles.month_end, 'Value',handles.metricdata.day_end);
        set(handles.day_end, 'Value',handles.metricdata.day_end);
```

```
function clear_axes(fig_handle, handles, isreset)
axes(handles.axes1);
cla;
axes(handles.axes2);
cla;
axes(handles.axes3);
cla;
```

Clear กราฟ

8) ทำการแปลงไฟล์จาก .m เป็นไฟล์ .exe

ไปยังหน้า Command Window แล้วพิมพ์คำสั่ง mcc –m ตามด้วยชื่อไฟล์ ในที่นี่ตั้งชื่อไฟล์ว่า GUI2 จะได้ “ mcc –m GUI2 ”

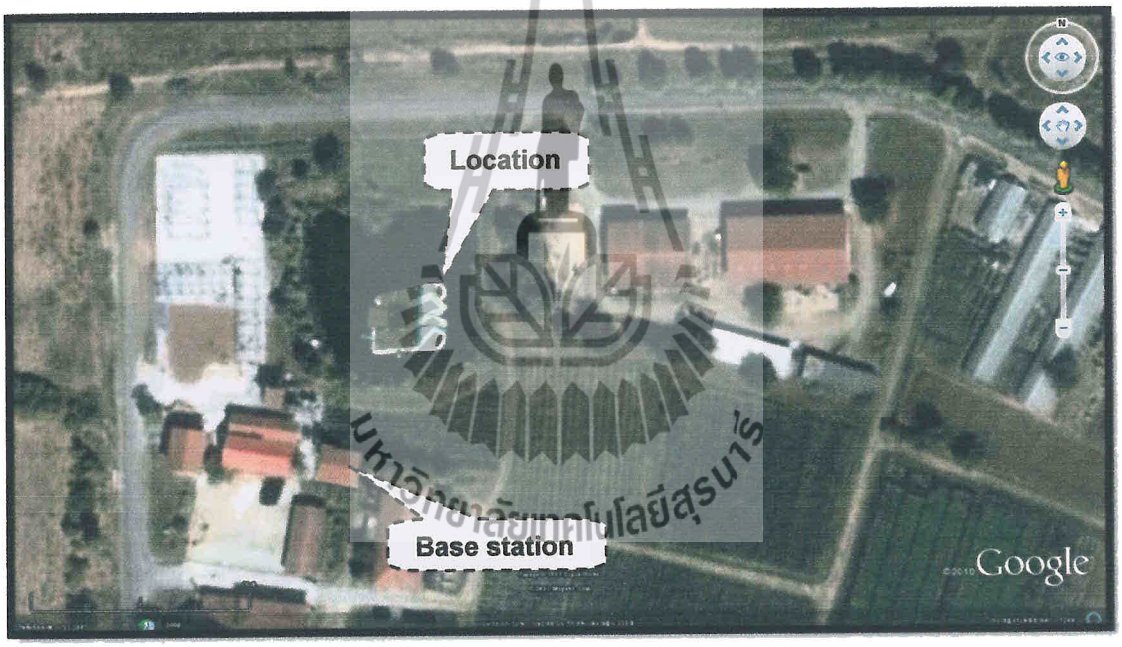
บทที่ 4

ผลการทดสอบระบบ

4.1 บทนำ

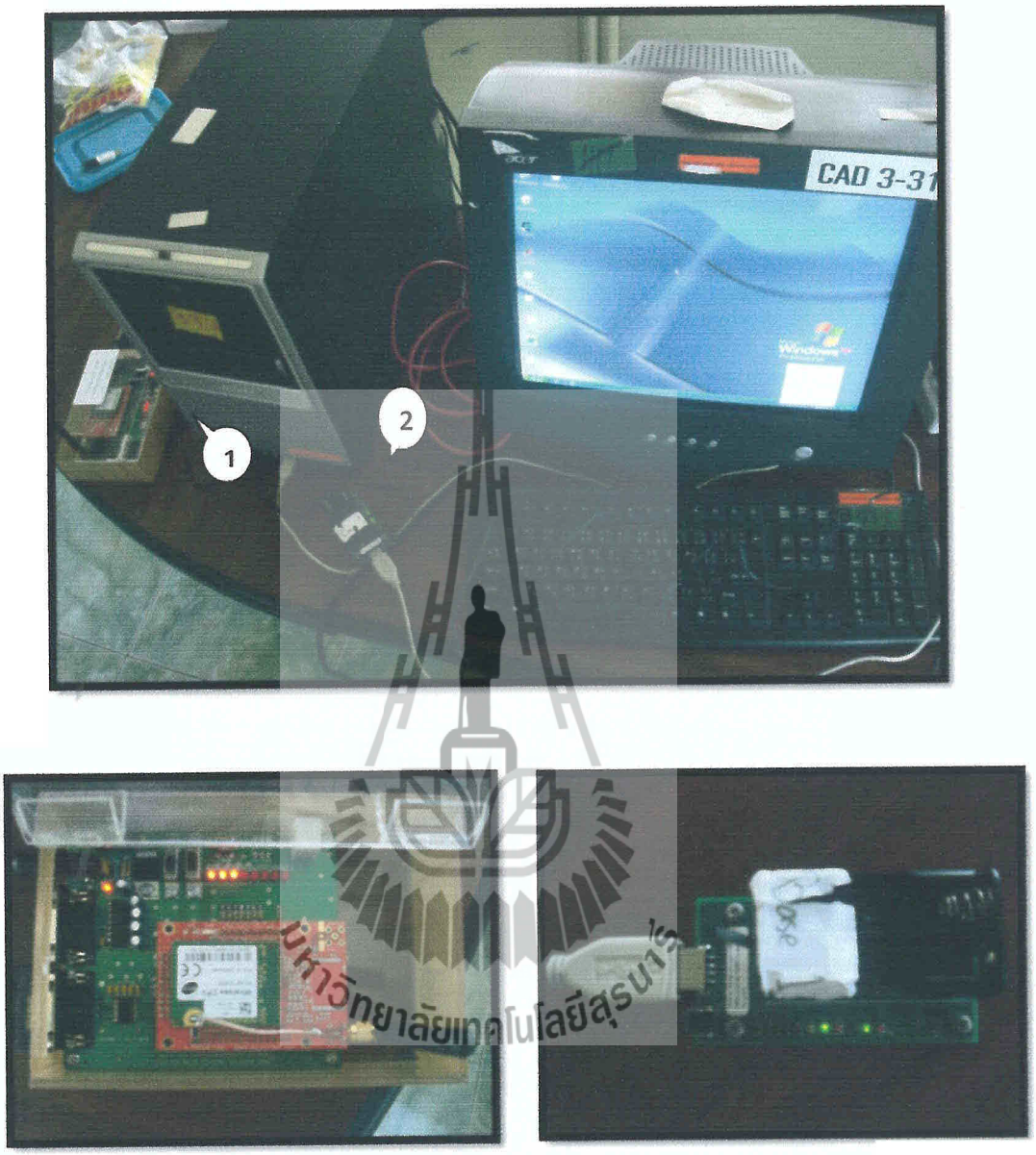
จากการศึกษาและทำความเข้าใจเกี่ยวกับทฤษฎีพื้นฐานในบทที่ 2 และ 3 นั้น ทำให้สามารถออกแบบโปรแกรมที่เสร็จสมบูรณ์พร้อมที่จะนำไปทดสอบการใช้งานจริง เพื่อให้บรรลุวัตถุประสงค์ของโครงงาน

4.2 โครงสร้างระบบทั้งหมดในการติดตั้ง



รูป 4.1 แผนที่แสดงที่ติดตั้ง ณ บริเวณศูนย์ความเป็นเลิศทางด้านชีวมวล มหาวิทยาลัยเทคโนโลยีสุรนารี

4.2.1 ส่วนของภาคส่ง



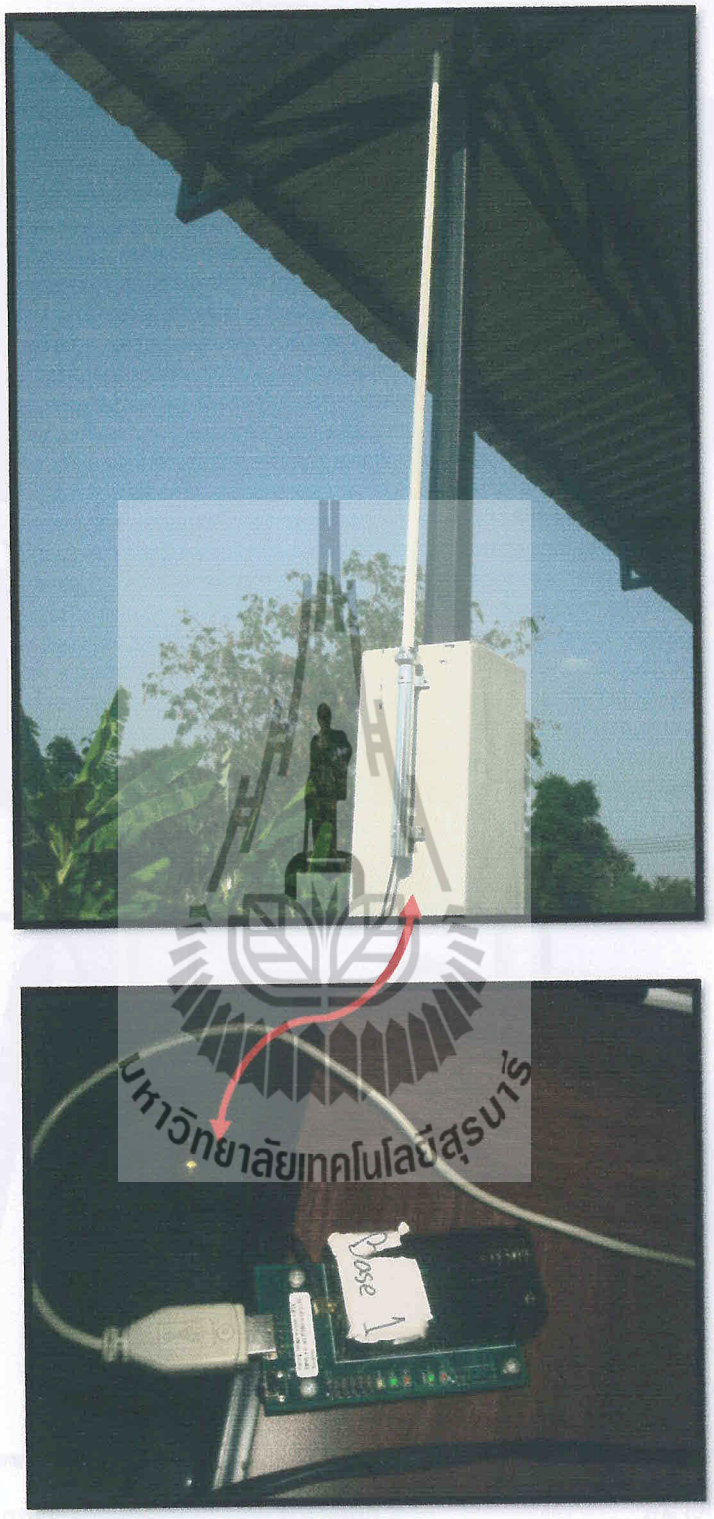
หมายเลข 1

บอร์ด GSM Module Wireless CPU

หมายเลข 2

ตัวรับ

รูป 4.2 แสดงส่วนประกอบโดยรวมของสถานีฐาน

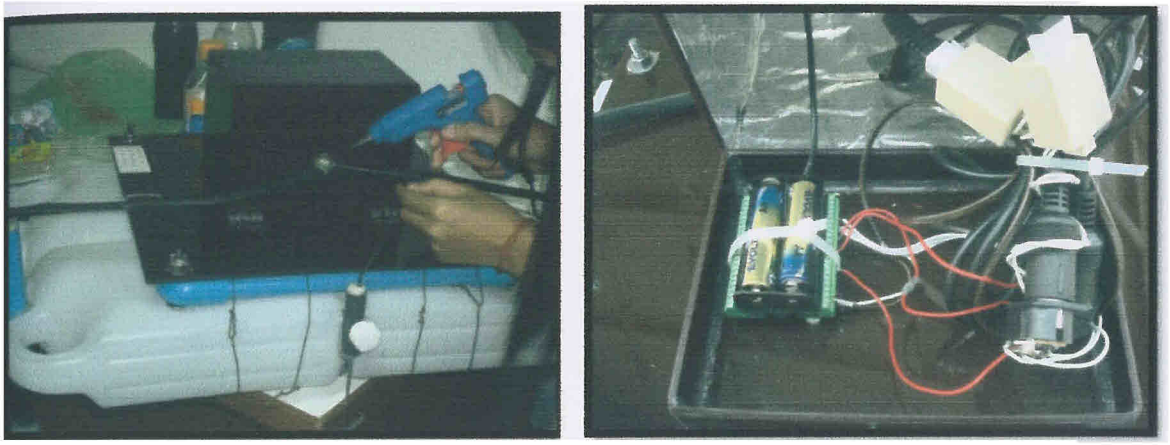


รูป 4.3 การเชื่อมต่อระหว่างสายอากาศกับโหมดของสถานีฐาน

4.2.2 ส่วนของภาครับ

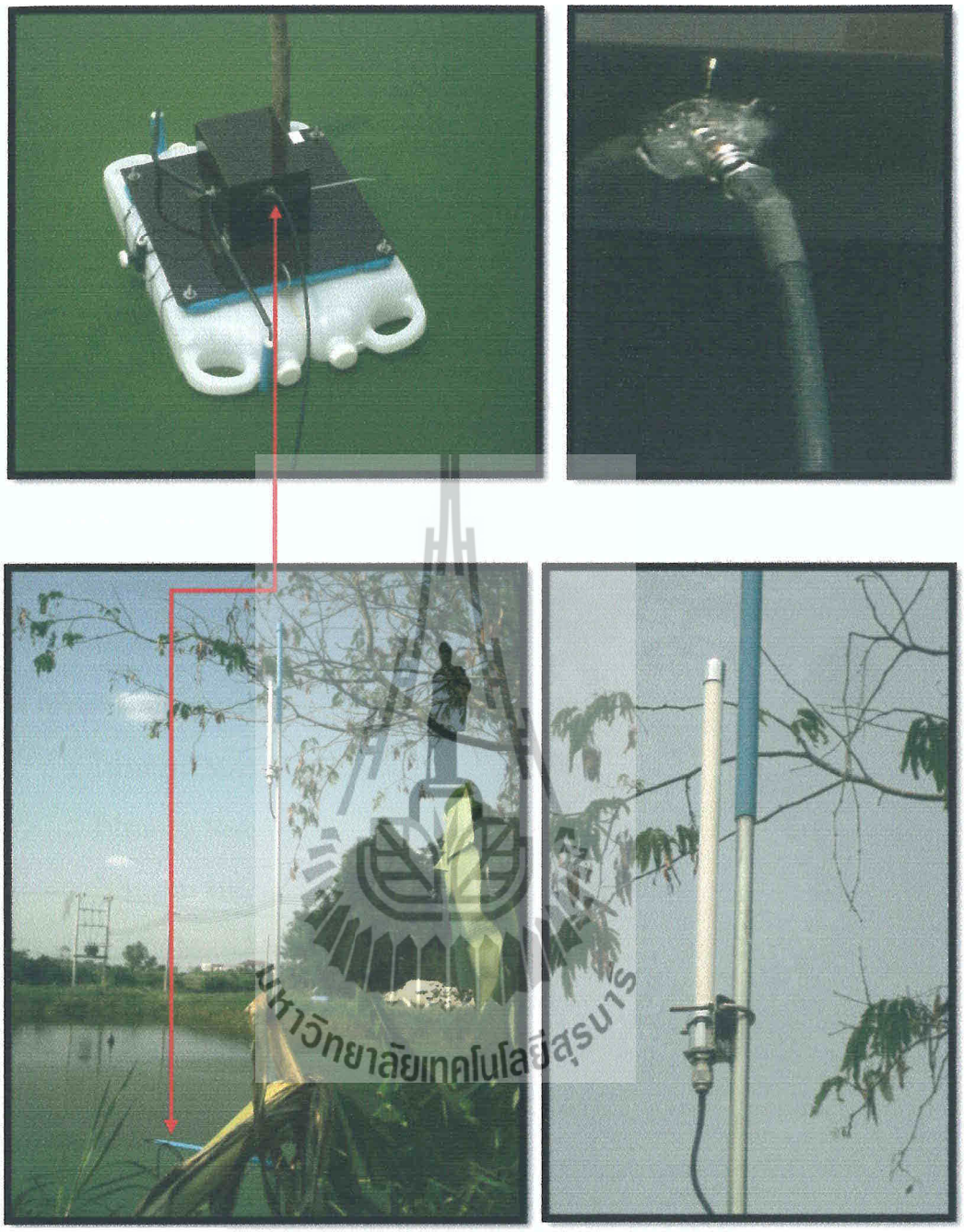
หมายเลข 3	หมายเลย 2	หมายเลข 1
Dissolved O_2	Temperature	pH sensor

รูป 4.4 ส่วนประกอบโดยรวมภายนอกของทุ่นลอย



รูป 4.5 ส่วนประกอบวงจรภายในของทุ่นลอย

รูป 4.6 เมื่อทุ่นนำไปวางในน้ำเรียบร้อยแล้ว



รูป 4.7 เชื่อมต่อระหว่างโหมดภาคส่งกับสายอากาศ

4.2.3 คอมพิวเตอร์ที่ใช้ในการติดตั้ง

1. PC Acer รุ่น Pentium IV 2.8 GHz 1 เครื่อง

- Main Processor: Processor Pentium IV 2.8 GHz
- RAM: DDR2 512 Mb
- Harddisk 80 Gb SATA
- CD-RW 52X32X52X
- NIC Onboard/Sound card Onboard/VAG-Onboard
- Monitor 17 นิ้ว
- CRT Acer
- Keyboard Hp
- Mouse Dell/USB

.2.4 โปรแกรมที่ใช้ในการติดตั้ง

1. โปรแกรม MATLAB R2009a
2. โปรแกรม Nesc
3. โปรแกรม Microsoft Visual C++ 6.0 และ SMSLibX

.2.5 ฮาร์ดแวร์ที่ใช้ในการติดตั้ง

1. บอร์ด GSM Module Wireless CPU
2. MDA300CA
3. MicaZ 2 บอร์ด
4. MIB5

.2.6 สายนำส่งสัญญาณที่ใช้ในการติดตั้ง

1. Lossless Coaxial Cable 50 Ω 2 เส้น

.2.7 สายแปลงสัญญาณที่ใช้ในการติดตั้ง

1. USB serial port
2. RS-232

4.2.8 เซ็นเซอร์ที่ใช้ในการติดตั้ง

1. Dissolved Oxigen Sensor (DO-BTA)

2. pH Sensor (PH-BTA)

3. EA high-frequency soil mota true sensor

- รุ่น EC5

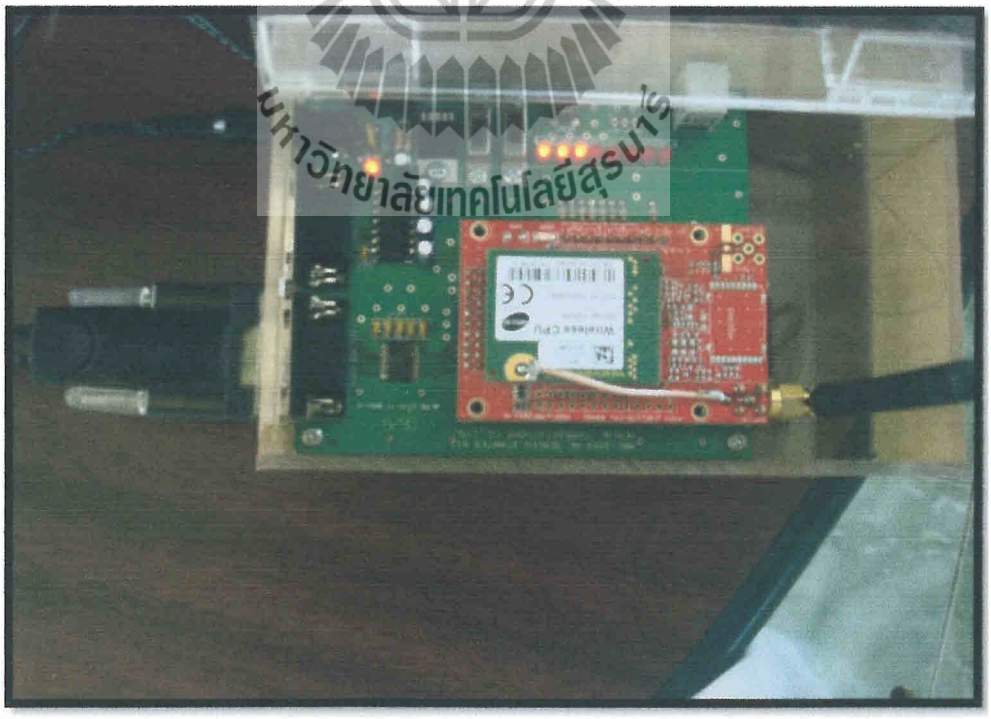
- ยี่ห้อ Decagon Devices

4.2.9 สายอากาศที่ใช้ในการติดตั้ง

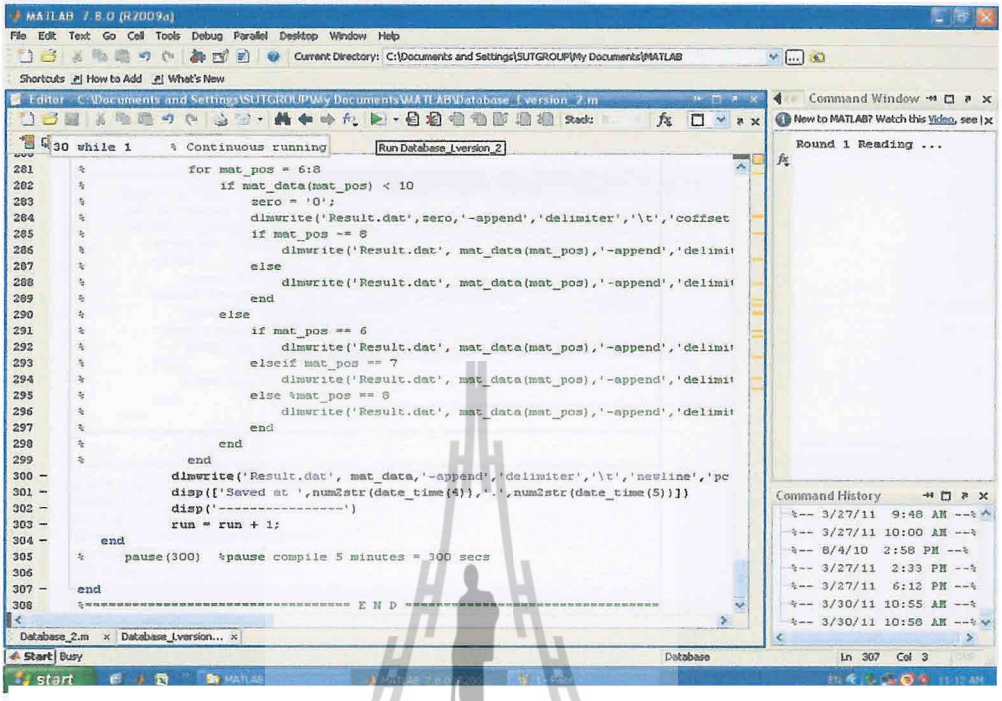
1. SYS2U ANT-OM09-2.4GHz Outdoor Antenna 9 dBi-omni Type Hot!

4.3 ขั้นตอนการทดสอบโปรแกรมส่ง SMS

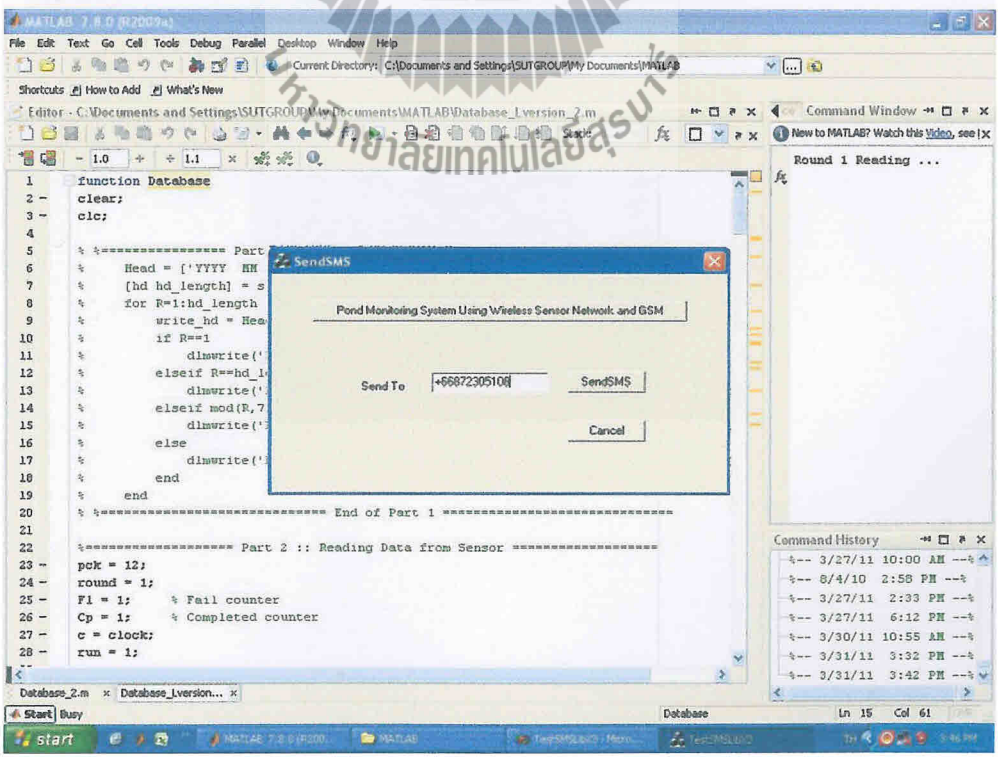
1) เชื่อมต่อ Module เข้ากับคอมพิวเตอร์ ผ่านทาง Serial Port



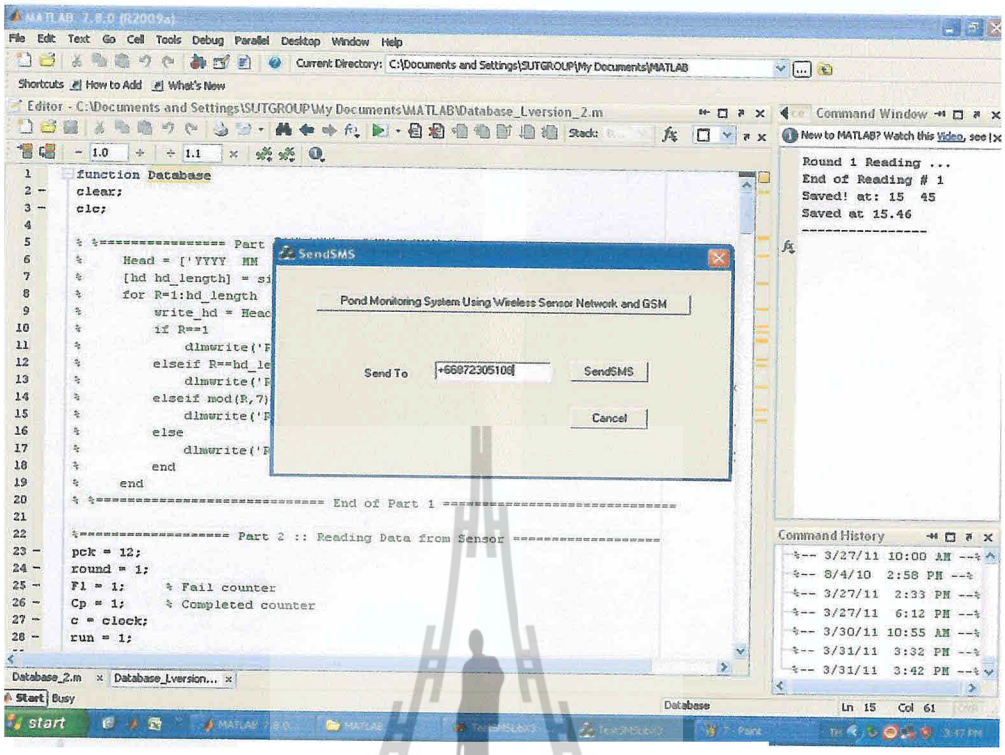
2) เปิดโปรแกรม MATLAB R2009a ขึ้นมา แล้วเปิดไฟล์ Database_Lversion2 ทำการรันโปรแกรมเพื่อเก็บข้อมูลจากเซ็นเซอร์ไร้สาย



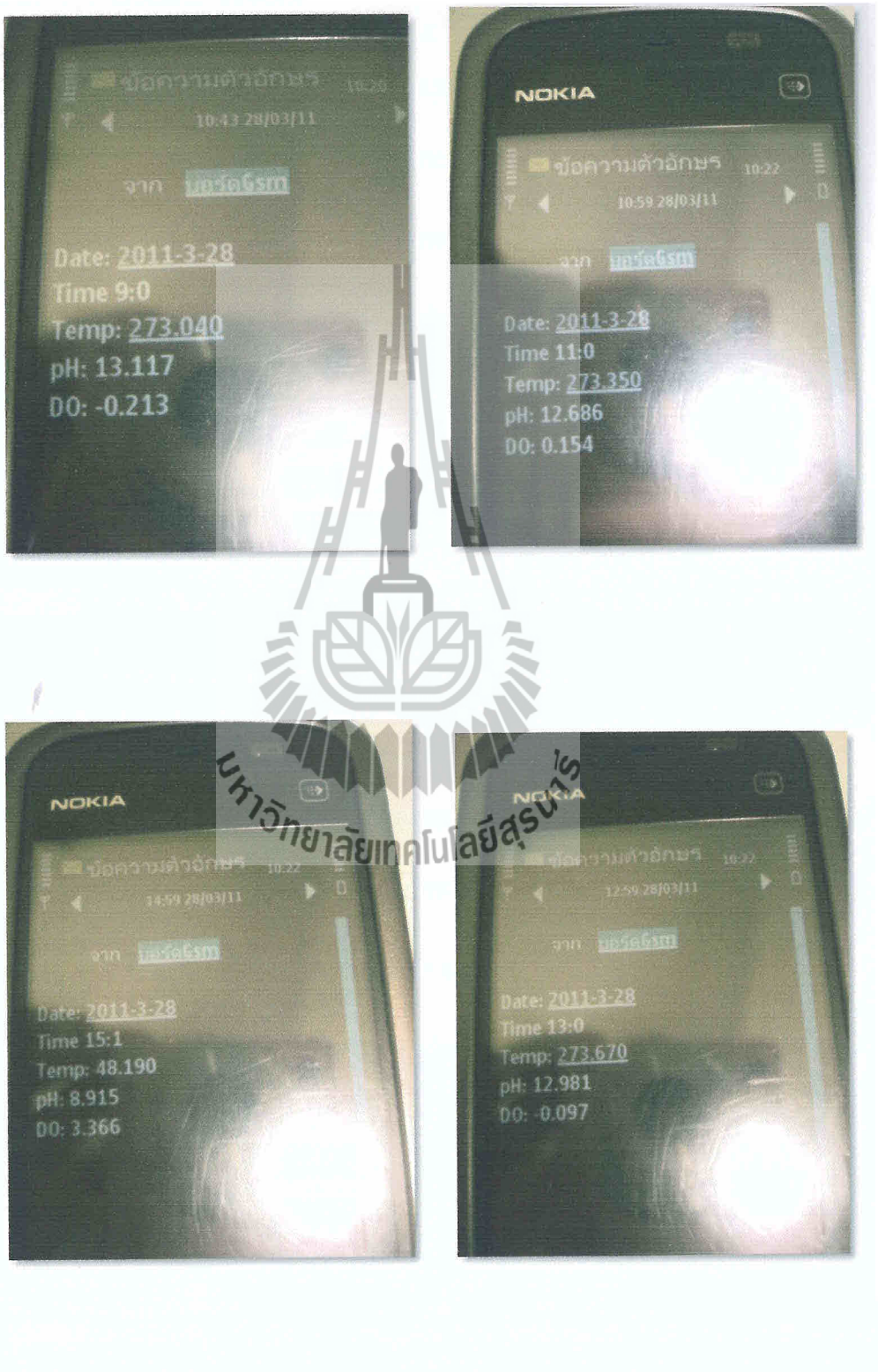
3) เปิดไฟล์โปรแกรม "SendSMS.dsw" Type Project Workspace ขึ้นมา แล้วทำการรันป้อนหมายเลขโทรศัพท์ปลายทาง กดปุ่ม SendSMS



4) เมื่อโปรแกรมรันเก็บข้อมูลเสร็จแล้ว จะมีการส่ง SMS มายังโทรศัพท์มือถือ

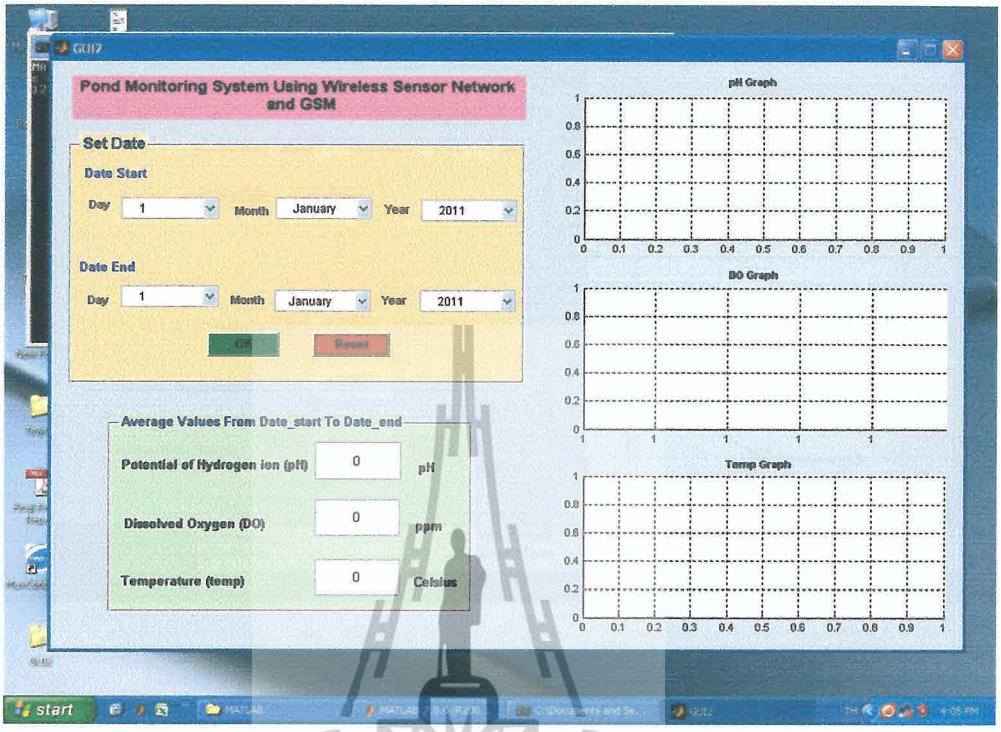


4.1) ผลการทดสอบโดยการตั้งค่าโปแกรมเก็บข้อมูลทุกสองชั่วโมง ตังแต่ 9.00-15.00น. (วันที่ 28 มี.ค. 2554)\

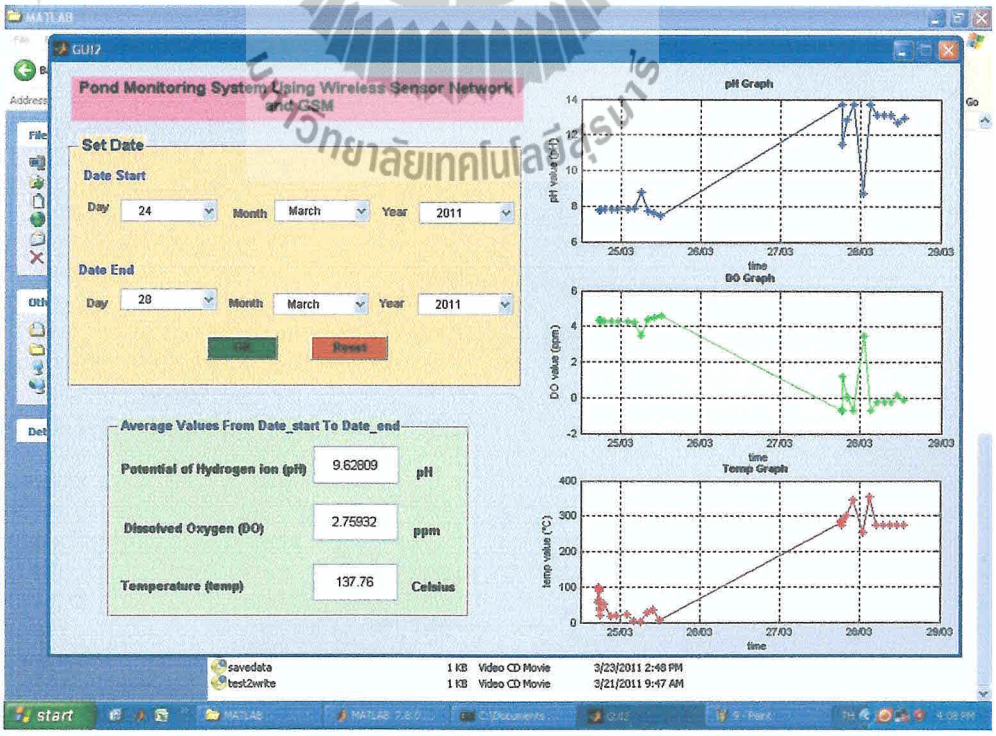


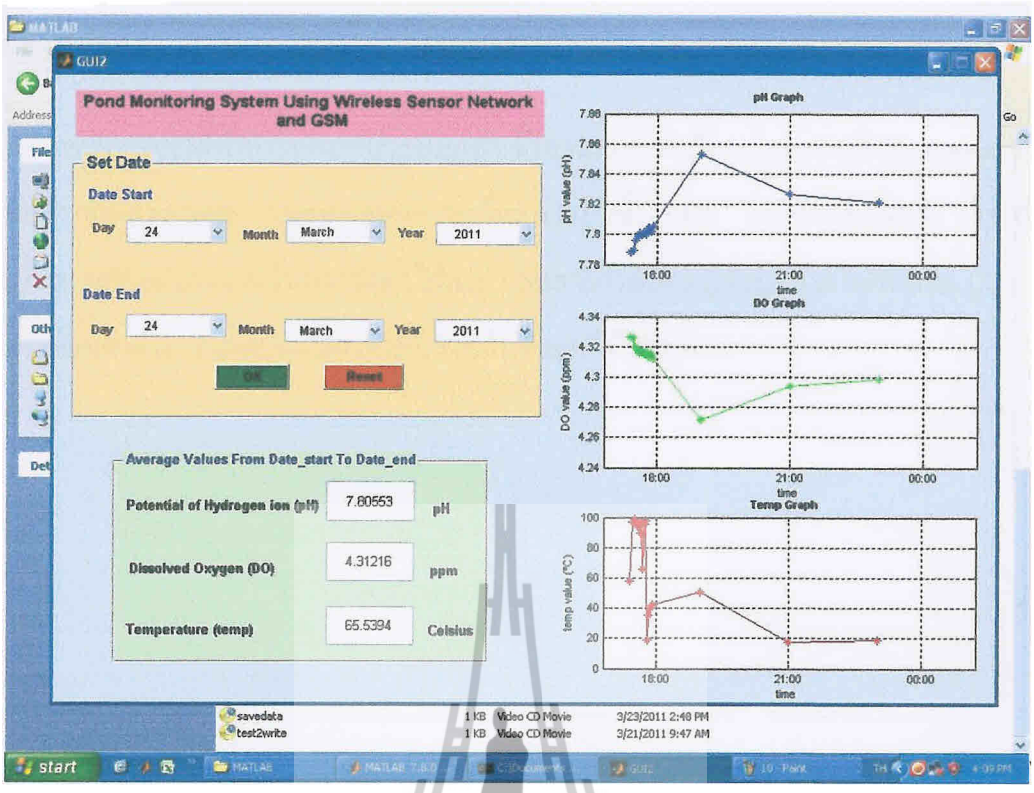
4.4 การทดสอบโปรแกรม GUI

1) เปิดโปรแกรม GUI2 Type Application ขึ้นมา



2) ป้อนวัน เดือน และปี ที่ต้องการดูข้อมูลแล้วกดปุ่ม OK





4.5 สรุปผล

หลังจากที่ได้ทำการทดสอบระบบแล้ว ซึ่งสามารถแจ้งเตือนค่าอุณหภูมิ (Temperature) ค่าความเป็นกด-ด่าง (pH) และค่า ออกซิเจนที่ละลายในน้ำ (DO) โดยการส่ง SMS ได้ ทำให้ประหยัดเวลาและแรงงานในการวัดค่าได้มาก และสามารถเรียกดูข้อมูลด้วยโปรแกรม GUI เพื่อเปรียบเทียบค่าต่างๆในแต่วันและเวลาที่มีการเก็บข้อมูลไว้ได้สะดวก

บทที่ 5

บทสรุปของโครงงาน

5.1 บทนำ

เนื้อหาในบทนี้จะกล่าวถึงบทสรุปของโครงงานระบบเฝ้าระวังบ่อเลี้ยงสัตว์น้ำด้วยเซ็นเซอร์ไร้สายผ่านระบบGSM ซึ่งประกอบไปด้วยข้อสรุปของโครงงาน ปัญหาที่พบขณะดำเนินงาน วิธีแก้ปัญหา ข้อเสนอแนะและวิธีการพัฒนาโครงงานต่อไป

5.2 สรุปโครงงาน

โครงงานระบบเฝ้าระวังบ่อเลี้ยงสัตว์น้ำด้วยเซ็นเซอร์ไร้สายผ่านระบบGSM ได้บรรลุตามวัตถุประสงค์ คือระบบเซ็นเซอร์ไร้สายสามารถรับ-ส่งข้อมูลได้อย่างสมบูรณ์ ทำให้มีข้อมูลมาเก็บไว้ที่สถานีฐาน โปรแกรมส่ง SMS ก็สามารถดึงข้อมูลจากสถานีฐานมาส่งSMS ได้ เพื่อแจ้งเตือนต่อผู้ใช้งานได้ และในส่วนของโปรแกรม GUI ก็สามารถดึงข้อมูลจากสถานีฐาน มาแสดงได้ ซึ่งทำให้เกิดเป็นระบบเฝ้าระวังบ่อเลี้ยงสัตว์น้ำด้วยเซ็นเซอร์ไร้สายผ่านระบบ GSM ที่สามารถนำไปติดตั้งใช้งาน ณ สถานที่จริงได้

5.3 ปัญหาและแนวทางในการแก้ไข

ในการทำโครงงานระบบเฝ้าระวังบ่อเลี้ยงสัตว์น้ำด้วยเซ็นเซอร์ไร้สายผ่านระบบ GSM ปัญหาที่พบบ่อยๆ แสดงดังตารางที่ 5.1 ซึ่งประกอบด้วยตัวปัญหาที่พบ สาเหตุของปัญหา และรวมถึงวิธีการแก้ไขปัญหา

ตารางที่ 5.1 ปัญหาและสาเหตุที่พบในขณะดำเนินงานและวิธีการแก้ไข

ปัญหาที่พบขณะดำเนินงาน	สาเหตุและวิธีการแก้ไข
1. การเลือกใช้ภาษาในการเขียนโปรแกรมให้เหมาะสมกับโครงงาน	**สาเหตุ:** เนื่องจากผู้จัดทำมีความรู้ในการเขียนโปรแกรมไม่มากนัก **วิธีการแก้ไข:** ทดลองศึกษาเขียนโปรแกรมด้วยภาษาต่างๆ โดยค้นคว้าข้อมูลจากหนังสือและ Internet จนพบภาษาของโปรแกรมที่เหมาะกับโครงงาน คือ โปรแกรม Microsoft Visual C++ 6.0
2. เกิดข้อผิดพลาดในการเขียนโปรแกรมส่งSMS	**สาเหตุ:** เนื่องจากทางผู้จัดทำยังไม่มีความชำนาญในการเขียนโปรแกรมด้วยภาษา C และ C ++ **วิธีการแก้ไข:** ได้ทำการศึกษาหาความรู้เพิ่มเติมของการเขียนโปรแกรมด้วยภาษา C และ C++ จากหนังสือและInternet
3. การใช้งานบอร์ด GSM	**สาเหตุ:** เกิดจากทางบริษัท ไม่ได้ส่งคู่มือการใช้งานมาให้ จึงทำให้ผู้จัดทำไม่สามารถใช้งาน **วิธีแก้ไข:** ติดต่อกลับไปที่บริษัทให้ส่งคู่มือแนะนำการใช้งานมาให้

ตารางที่ 5.1 ปัญหาและสาเหตุที่พบในขณะดำเนินงานและวิธีการแก้ไข (ต่อ)

4. เกิดปัญหาการเขียน GUI แสดงกราฟข้อมูล	**สาเหตุ:** ทางผู้จัดทำได้เลือกใช้โปรแกรม Microsoft Visual C++ 2008 ในการสร้าง GUI แต่เกิดปัญหา library ที่ใช้ในการพล็อตกราฟ หาได้ยาก และที่หาได้ส่วนใหญ่ไม่มีความถูกต้อง ผู้จัดทำต้องนำมาแก้ไขอีกจำนวนมาก **วิธีการแก้ไข:** เปลี่ยนโปรแกรมที่ใช้ในการเขียนมาเป็น MATLAB R2009a ซึ่งใช้งานได้ง่ายกว่า และภาษาที่ใช้คล้ายกับภาษาซีที่ผู้ใช้ได้ทำมาก่อนแล้ว เพียงนำมาดัดแปลงแก้ไขเพิ่มเติมในส่วนของการแสดงผล ซึ่งโปรแกรม MATLAB R2009a รองรับการใช้งานทางด้านกราฟิกอยู่แล้ว
5. ปัญหาเกิดเมื่อนำไปติดตั้ง ณ สถานที่จริง	**สาเหตุ:** การหาสัญญาณที่ใช้ในการส่งข้อมูลระหว่างเซ็นเซอร์กับสถานีฐาน เนื่องจากการส่งข้อมูลเป็นแบบ line-of-sight **วิธีแก้ไข:** หาสัญญาณ ณ บริเวณใกล้เคียงไปเรื่อยๆ จนพบตำแหน่งที่เหมาะสมและเป็นบริเวณที่มีสัญญาณดีที่สุดในระดับที่สามารถส่งข้อมูลสื่อสารกันระหว่างเซ็นเซอร์กับสถานีฐานได้

5.4 ข้อเสนอแนะ

5.3.1 ควรศึกษาวิธีการใช้โปรแกรมอย่างละเอียด ก่อนใช้งานเนื่องจากอาจเกิดข้อผิดพลาดระหว่างใช้งานได้

5.3.2 การใช้โปรแกรมควรตรวจสอบการตั้งค่าระบบของโปรแกรมให้เหมาะสม เช่น การตั้งค่าของ port

5.3.3 การทดสอบการส่งสัญญาณระหว่างเซ็นเซอร์กับสถานีฐานควรหลีกเลี่ยงสิ่งกีดขวาง เช่น อาคาร ต้นไม้ เป็นต้น ดังนั้นถ้าต้องการหลีกเลี่ยงปัจจัยนี้ควรจะเลือกเป็นสถานที่โล่ง ซึ่งจะทำงานได้ดีในลักษณะ line-of-sight

5.3.4 การดูแลรักษาเซ็นเซอร์ต้องเป็นไปตามข้อแนะนำที่มากับเซ็นเซอร์เนื่องจากเซ็นเซอร์มีความเปราะบาง ถ้าใช้ไม่ถูกวิธีอาจทำให้เกิดความเสียหายได้ ซึ่งจะทำให้ค่าที่วัดได้เกิดความผิดเพี้ยน

5.5 แนวทางในการพัฒนาต่อไป

เนื่องจากโครงงานระบบเฝ้าระวังบ่อเลี้ยงสัตว์น้ำด้วยเซ็นเซอร์ไร้สายผ่านระบบ GSM สามารถบอกค่าคุณภาพของน้ำผ่านทางSMSตามเวลาที่ตั้งไว้ ซึ่งอาจเป็นการสิ้นเปลืองจำนวนเงินในการส่งข้อความ ฉะนั้นในการพัฒนาต่อจะให้สามารถแจ้งเตือนเฉพาะกรณีที่น้ำเกิดความผิดปกติขึ้นเท่านั้น และสามารถพัฒนาใช้ได้กับโรงงานอุตสาหกรรม เพื่อใช้ตรวจสอบคุณภาพของบ่อบำบัดน้ำเสียในโรงงานก่อนปล่อยออกสู่แม่น้ำลำคลอง ทำให้ประหยัดทั้งเวลาและบุคลากรที่ต้องคอยตรวจสอบอยู่เสมอ

5.6 กล่าวสรุป

โครงงานระบบเฝ้าระวังบ่อเลี้ยงสัตว์น้ำด้วยเซ็นเซอร์ไร้สายผ่านระบบ GSM มีส่วนประกอบหลักดังนี้

1. ภาษาที่ใช้ในการพัฒนาเขียนโปรแกรม
   - ภาษา C และ C++
2. Hardware ที่นำมาใช้
   - Computer PC
   - บอร์ด GSM Module Wireless CPU
   - Sims care โทรศัพท์เคลื่อนที่
   - สายอากาศ
3. เซ็นเซอร์
   - pH
   - DO (Dissolved Oxygen)
   - Temperature Probes
4. Software ที่นำมาใช้
   - Microsoft Visual C++ 6.0
   - MATLAB R2009a

ประวัติผู้เขียน

นางสาวพิมจันทร์ ชัยชนะ เกิดเมื่อวันที่ 22 มกราคม พ.ศ. 2531 ภูมิลำเนาอยู่ที่ ตำบลสร้างก่อ อำเภอกุดจับ จังหวัดอุดรธานี สำเร็จการศึกษาระดับมัธยมปลายจากโรงเรียนกุดจับประชาสรรค์ อำเภอกุดจับ จังหวัดอุดรธานี เมื่อปี พ.ศ. 2549 ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4 สาขาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

นางสาวพัชรียา พวงมาลัย เกิดเมื่อวันที่ 29 ธันวาคม พ.ศ. 2531 ภูมิลำเนาอยู่ที่ ตำบลหนองหญ้า อำเภอเมือง จังหวัดกาญจนบุรี สำเร็จการศึกษาระดับมัธยมปลายจากโรงเรียนกาญจนานุเคราะห์ อำเภอเมือง จังหวัดกาญจนบุรี เมื่อปี พ.ศ. 2549 ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

บรรณานุกรม

- ผศ.ดร. พีระพงษ์ อุฑารสกุล, 427459 ระบบสื่อสารโทรศัพท์เคลื่อนที่ (Mobile Communication System). สาขาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี.

- นายณัฐพงศ์ แซ่หู, นางสาวชฎารัตน์ หาดทวายกาญจน์, นายณัฐวุฒิ ภูงามเงิน โปรแกรมซอฟต์แวร์การกระจายข่าวสารด้วยระบบ SMS. โครงงานวิศวกรรมโทรคมนาคม สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี, 2553.

- นางสาวนีรนุช มหาสมุทร, นางสาวมะลิวรรณ ผิวทอง ระบบการวัดอุณหภูมิโดยผ่านเครือข่ายไร้สาย Zigbee. โครงงานวิศวกรรมโทรคมนาคม สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี, 2552.

- เกษมสันต์ พานิชการ. C++ และหลักการของ OOP ฉบับเริ่มต้น. กรุงเทพฯ : ซีเอ็ดยูเคชั่น , 2537.

- Bates, Jonathan. Practical Visual C++ 6. Indianapolis, Ind. : Que, c1999

- Biran, Adrian. MATLAB for engineers. Wokingham, England ;Reading, Mass. : Addison-Wesley Pub. Co., c1995.

- Deitel, Paul J. Visual C++ 2008 : how to program. 2nd ed. Upper Saddle River, N.J. : Pearson Prentice Hall : Deitel, c2008.

- Chapman, Stephen J. MATLAB programming for engineers. Australia ;Pacific Grove, CA : Brooks/Cole, c2000.

- Higham, D. J. MATLAB guide. 2nd ed. Philadelphia : Society for Industrial and Applied Mathematics, c2005.

- Holzner, Steven. Fast track Visual C++ 6.0 programming. New York : Wiley, c1998.

- http://www.thaitelecomkm.org/TTE/topic/attach/Bluetooth_and_Zigbee/ index.php

- http://student.nu.ac.th/electronic/00005.doc

- http://www.cpe.ku.ac.th/~yuen/204471/sensor/temp_pres/

- http://www.codeguru.com/forum/archive/index.php/t-74303.html

- http://www.google.co.th/

- http://www.mathworks.com/help/techdoc/ref/clabel.html

- http://www.vernier.com/probes/

ภาคผนวก

ภาคผวก ก

Code โปรแกรมที่เกี่ยวข้อง

1. โปรแกรมส่ง SMS

```
// SendSMSDlg.cpp : implementation file

#include "stdafx.h"

#include "SendSMS.h"

#include "SendSMSDlg.h"

#include "DlgProxy.h"

#include "SMSLibXConstants.h"

#include "SMSLibX.h"

#include <iostream.h>

#include <fstream.h>

#include <iomanip.h>

#include <stdio.h>

#include <time.h>

#ifdef _DEBUG

#define new DEBUG_NEW

#undef THIS_FILE

static char THIS_FILE[] = __FILE__;
```

```
#endif

// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
        CAboutDlg();

// Dialog Data
        //{{AFX_DATA(CAboutDlg)
        enum { IDD = IDD_ABOUTBOX };
        //}}AFX_DATA
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CAboutDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
        //}}AFX_VIRTUAL
// Implementation
protected:
        //{{AFX_MSG(CAboutDlg)
        //}}AFX_MSG
```

```
	DECLARE_MESSAGE_MAP()
};
CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}
void waitTime ( int seconds )
{
 clock_t endwait;
 endwait = clock () + seconds * CLOCKS_PER_SEC ;
 while (clock() < endwait) {}
}
void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}
```

```
BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
        //{{AFX_MSG_MAP(CAboutDlg)
                // No message handlers
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()
// CSendSMSDlg dialog
IMPLEMENT_DYNAMIC(CSendSMSDlg, CDialog);
CSendSMSDlg::CSendSMSDlg(CWnd* pParent /*=NULL*/)
        : CDialog(CSendSMSDlg::IDD, pParent)
{
        //{{AFX_DATA_INIT(CSendSMSDlg)
        // NOTE: the ClassWizard will add member initialization here
        //}}AFX_DATA_INIT
        // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
        m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
        m_pAutoProxy = NULL;
}
CSendSMSDlg::~CSendSMSDlg()
{
```

```
        // If there is an automation proxy for this dialog, set
        //  its back pointer to this dialog to NULL, so it knows
        //  the dialog has been deleted.
        if (m_pAutoProxy != NULL)
                m_pAutoProxy->m_pDialog = NULL;
}

void CSendSMSDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CSendSMSDlg)
                // NOTE: the ClassWizard will add DDX and DDV calls here
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CSendSMSDlg, CDialog)
        //{{AFX_MSG_MAP(CSendSMSDlg)
        ON_WM_SYSCOMMAND()
        ON_WM_PAINT()
        ON_WM_QUERYDRAGICON()
        ON_WM_CLOSE()
```

```
    ON_BN_CLICKED(IDC_BUTTON1, OnSendSMS)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

// CSendSMSDlg message handlers

BOOL CSendSMSDlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    // Add "About..." menu item to system menu.
    // IDM_ABOUTBOX must be in the system command range.
    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);
    CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != NULL)
    {
        CString strAboutMenu;
        strAboutMenu.LoadString(IDS_ABOUTBOX);
        if (!strAboutMenu.IsEmpty())
        {
            pSysMenu->AppendMenu(MF_SEPARATOR);
```

```
                    pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX,
strAboutMenu);
                }
        }
        // Set the icon for this dialog.  The framework does this automatically
        //  when the application's main window is not a dialog
        SetIcon(m_hIcon, TRUE);                 // Set big icon
        SetIcon(m_hIcon, FALSE);                // Set small icon
        // TODO: Add extra initialization here
        return TRUE;  // return TRUE  unless you set the focus to a control
}
void CSendSMSDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
        if ((nID & 0xFFF0) == IDM_ABOUTBOX)
        {
                CAboutDlg dlgAbout;
                dlgAbout.DoModal();
        }
        else
```

```
        {
                CDialog::OnSysCommand(nID, lParam);
        }
}
// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.
void CSendSMSDlg::OnPaint()
{
        if (IsIconic())
        {
                CPaintDC dc(this); // device context for painting
                SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(),
0);
                // Center icon in client rectangle
                int cxIcon = GetSystemMetrics(SM_CXICON);
                int cyIcon = GetSystemMetrics(SM_CYICON);
                CRect rect;
                GetClientRect(&rect);
```

```
            int x = (rect.Width() - cxIcon + 1) / 2;
            int y = (rect.Height() - cyIcon + 1) / 2;
            // Draw the icon
            dc.DrawIcon(x, y, m_hIcon);
        }
        else
        {
            CDialog::OnPaint();
        }
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CSendSMSDlg::OnQueryDragIcon()
{
        return (HCURSOR) m_hIcon;
}
// Automation servers should not exit when a user closes the UI
//  if a controller still holds on to one of its objects.  These
```

```
// message handlers make sure that if the proxy is still in use,
// then the UI is hidden but the dialog remains around if it
// is dismissed.
void CSendSMSDlg::OnClose()
{
        if (CanExit())
                CDialog::OnClose();
}
void CSendSMSDlg::OnOK()
{
        if (CanExit())
                CDialog::OnOK();
}
void CSendSMSDlg::OnCancel()
{
        if (CanExit())
                CDialog::OnCancel();
}
BOOL CSendSMSDlg::CanExit()
```

```
{
        // If the proxy object is still around, then the automation
        //  controller is still holding on to this application.  Leave
        //  the dialog around, but hide its UI.
        if (m_pAutoProxy != NULL)
        {
                ShowWindow(SW_HIDE);
                return FALSE;
        }
        return TRUE;
}
void CSendSMSDlg::OnSendSMS()
{
                CString oper;
        char str[128];
        memset(str,0,128);
        _SMSModem modem;
        if( !modem.CreateDispatch("SMSLibX.SMSModem") ) {
```

```
                AfxMessageBox("Cannot link to SMSLibX.SMSModem object: SMSLibX
library may be not registered!");
                return;
        }
long modemType = SMSLibXConstants::GSMModemTypeConstants::gsmModemDummy;
short comPort = 9;
        CString smscNumber = "";
        long notifyMode =
SMSLibXConstants::SMSModemNotificationConstants::smsNotifyNone;
        bool unregisteredMode = false;
        short timeout = 60;
CWaitCursor wait;
        try {
          FILE *fp;
                int n;
                int size1=0;
                int size2=0;
                int year1,year;
                int month1,month;
                int day1,day;
```

```
                int hour1,hour;
                int min1,min;
                double temp1,temp;
                double pH1,pH;
                double DO1,DO;
                oper = "Opening modem communication";
                modem.OpenComm(modemType, comPort, smscNumber, notifyMode,
        timeout, unregisteredMode);
        oper = "Creating SMS message";
        CString destNumber;
        GetDlgItemText(IDC_DESTNUM, destNumber);
        for (n=1; n>0; n++)
                {
                fp = fopen("Result.dat","rt");
                fseek(fp,0,SEEK_END);
                size1 = ftell(fp);
                fclose(fp);
        if (size1>size2) {
ifstream inClientFile("Result.dat",ios::in)
        while(inClientFile>>year1>>month1>>day1>>hour1>>min1>>temp1>>pH1>>DO1)
```

```
{
        year = year1;
        month = month1;
        day = day1;
        hour = hour1;
        min = min1;
        temp = temp1;
        pH = pH1;
        DO = DO1;
            }

CString stryear,strmonth,strday,strhour,strmin,strtemp,strpH,strDO;
        stryear.Format("%d", year);
        strmonth.Format("%d", month);
        strday.Format("%d", day);
        strhour.Format("%d", hour);
        strmin.Format("%d", min);
        strtemp.Format("%.3f", temp);
        strpH.Format("%.3f", pH);
```

```
                strDO.Format("%.3f", DO);

        CString msgTxt ="Date: "+stryear+"-"+strmonth+"-"+strday+"\nTime
"+strhour+":"+strmin+"\nTemp: "+strtemp+"\npH: "+strpH+"\nDO: "+strDO+"";

                oper = "Sending SMS message";

                int msgId = modem.SendTextMessage(destNumber, msgTxt, false);

        size2=size1;

                }

                else {

                        ofstream inClientFile11("testt.text",ios::out);

                        CString msgTxt1 = "Hello\t";

                        inClientFile11 << msgTxt1;

                        inClientFile11.close();

                        }
                waitTime (5);

        }


} catch (CException* e) {

                e->GetErrorMessage(str, 128);

                AfxMessageBox("Failure on "+oper+":\n"+str);
```

```
                e->Delete();
            }
        try{ modem.CloseComm(); }catch(...){;}
            modem.ReleaseDispatch();
        }
```

2. โปรแกรม GUI

```
function varargout = GUI2(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI2_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
```

```
end
function GUI2_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);

function varargout = GUI2_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function [year_start]= year_start_Callback(hObject, eventdata, handles)
year_start1=get(handles.year_start, 'Value');
year_start=2010+year_start1;

function year_start_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function [month_start] = month_start_Callback(hObject, eventdata, handles)
month_start=get(handles.month_start, 'Value');

function month_start_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function [day_start] = day_start_Callback(hObject, eventdata, handles)
day_start=get(handles.day_start, 'Value');
```

```
function day_start_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function [year_end] = year_end_Callback(hObject, eventdata, handles)
year_end1=get(handles.year_end, 'Value');
year_end=2010+year_end1;

function year_end_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function [month_end] = month_end_Callback(hObject, eventdata, handles)
month_end=get(handles.month_end, 'Value');
.
function month_end_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function [day_end] = day_end_Callback(hObject, eventdata, handles)
day_end=get(handles.day_end, 'Value');
```

```
function day_end_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function average_PH_Callback(hObject, eventdata, handles)

function average_PH_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function average_DO_Callback(hObject, eventdata, handles)

function average_DO_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function average_temp_Callback(hObject, eventdata, handles)

function average_temp_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function Reset_Callback(hObject, eventdata, handles)
initialize_gui(gcbf, handles, true);
clear_axes(gcbf, handles, true);

function initialize_gui(fig_handle, handles, isreset)
if isfield(handles, 'metricdata') && ~isreset
   return;
end
handles.metricdata.average_PH = 0;
handles.metricdata.average_DO = 0;
handles.metricdata.average_temp = 0;
handles.metricdata.day_end = 1;
set(handles.average_PH, 'String', handles.metricdata.average_PH);
set(handles.average_DO, 'String', handles.metricdata.average_DO);
set(handles.average_temp, 'String', handles.metricdata.average_temp);
set(handles.year_start, 'Value',handles.metricdata.day_end);
set(handles.month_start, 'Value',handles.metricdata.day_end);
set(handles.day_start, 'Value',handles.metricdata.day_end);
set(handles.year_end, 'Value',handles.metricdata.day_end);
set(handles.month_end, 'Value',handles.metricdata.day_end);
set(handles.day_end, 'Value',handles.metricdata.day_end);
function clear_axes(fig_handle, handles, isreset)
axes(handles.axes1);
cla;
axes(handles.axes2);
cla;
axes(handles.axes3);
```

```
cla;
function OK_Callback(hObject, eventdata, handles)
year_start = year_start_Callback(hObject, eventdata, handles);
month_start = month_start_Callback(hObject, eventdata, handles);
day_start = day_start_Callback(hObject, eventdata, handles);
year_end = year_end_Callback(hObject, eventdata, handles) ;
month_end = month_end_Callback(hObject, eventdata, handles) ;
day_end = day_end_Callback(hObject, eventdata, handles);
flag_start = false;
flag_end = false;
flag_end1 = false;
sum_PH = 0;
sum_DO = 0;
sum_temp = 0;
count_PH = 0;
count_DO = 0;
count_temp = 0;
count_YS = 0;
fid_q=fopen('Result.dat','r');
while flag_end~=true
[num,count]=fscanf(fid_q,'%8f', [1,8]);
  if((num(1,1)==year_start)&&(num(1,2)==month_start)&&(num(1,3)==day_start))
     flag_start=true;
  end
  if((flag_start==true)&&(flag_end==false))
        if num(1,1)
           count_YS = count_YS+1;
           year(count_YS)=num(1,1);
```

```
        month(count_YS)=num(1,2);
        day(count_YS)=num(1,3);
        hour(count_YS)=num(1,4);
        mn(count_YS)=num(1,5);
        sec(count_YS)=0;
      end
      if num(1,6)
        sum_temp = sum_temp + num(1,6);
        count_temp = count_temp + 1;
        temp( count_temp ) = num(1,6);
      end
      if num(1,7)
        sum_PH = sum_PH + num(1,7);
        count_PH = count_PH + 1;
        PH( count_PH ) = num(1,7);
      end
      if num(1,8)
        sum_DO = sum_DO + num(1,8);
        count_DO = count_DO + 1;
        DO( count_DO ) = num(1,8);
     end
end
if((num(1,1)==year_end)&&(num(1,2)==month_end)&&(num(1,3)==day_end))
      flag_end1=true;
end
if(((num(1,1)~=year_end)||(num(1,2)~=month_end)||(num(1,3)~=day_end))&&(flag_end1==true))
   flag_end=true;
end
```

```
end
 avg_temp = (sum_temp-num(1,6))/(count_temp-1);
avg_PH = (sum_PH-num(1,7))/(count_PH-1);
avg_DO  = (sum_DO-num(1,8))/(count_DO-1);
 xdate = datenum(year,month,day,hour,mn,sec);


axes(handles.axes3);
if (day_start==day_end && month_start==month_end && year_start==year_end)
   s = plot(xdate(1:count_temp-1) , temp(1:count_temp-1) ,'-r*');
   datetick('x',15,'keeplimits')
   grid on;
   hold on;
   xlabel('time'),ylabel('temp value (°C)');
elseif (year_start~=year_end)
   s = plot(xdate(1:count_temp-1) , temp(1:count_temp-1) ,'-r*');
   datetick('x','mmm-yy','keeplimits')
   grid on;
   hold on;
   xlabel('time'),ylabel('temp value (°C)');


else
   s = plot(xdate(1:count_temp-1) , temp(1:count_temp-1) ,'-r*');
   datetick('x',19,'keeplimits')
   grid on;
   hold on;
   xlabel('time'),ylabel('temp value (°C)');
end
```

```
axes(handles.axes1);
if (day_start==day_end && month_start==month_end && year_start==year_end)
   s = plot(xdate(1:count_PH-1) , PH(1:count_PH-1) ,'-b*');
   datetick('x',15,'keeplimits')
   grid on;
   hold on;
   xlabel('time'),ylabel('pH value (pH)');
elseif (year_start~=year_end)
   s = plot(xdate(1:count_PH-1) , PH(1:count_PH-1) ,'-b*');
  datetick('x','mmm-yy','keeplimits')
   grid on;
   hold on;
   xlabel('time'),ylabel('pH value (pH)');

else
   s = plot(xdate(1:count_PH-1) , PH(1:count_PH-1) ,'-b*');
   datetick('x',19,'keeplimits')
   grid on;
   hold on;
   xlabel('time'),ylabel('pH value (pH)');
end

axes(handles.axes2);
if (day_start==day_end && month_start==month_end && year_start==year_end)
   s = plot(xdate(1:count_DO-1) , DO(1:count_DO-1) ,'-g*');
   datetick('x',15,'keeplimits')
   grid on;
```

```
    hold on;
    xlabel('time'),ylabel('DO value (ppm)');
elseif (year_start~=year_end)
    s = plot(xdate(1:count_DO-1) , DO(1:count_DO-1) ,'-g*');
    datetick('x','mmm-yy','keeplimits')
    grid on;
    hold on;
    xlabel('time'),ylabel('DO value (ppm)');
else
    s = plot(xdate(1:count_DO-1) , DO(1:count_DO-1) ,'-g*');
    datetick('x',19,'keeplimits')
    grid on;
    hold on;
    xlabel('time'),ylabel('DO value (ppm)');
end
set(handles.average_PH,'String', avg_PH);
set(handles.average_DO,'String', avg_DO);
set(handles.average_temp,'String', avg_temp);
```


3. โปรแกรมบันทึกข้อมูลไว้ในไฟล์ .dat

```
% function Database
clear;
clc;
```

```
% %==================== Part 1 :: Reading Data from Sensor ====================
pck = 12;
round = 1;
Fl = 1;    % Fail counter
Cp = 1;    % Completed counter


c = clock;
min_time = c(1,5);  %Select minute
sec_time = c(1,6);  %Select second
run = 1;
while 1
   c1 = clock;
   %if (min_time == 20) || (min_time == 30) || (min_time == 40) || (min_time == 50) || (min_time ==
00)|| (min_time == 10)|| (min_time == 5)|| (min_time == 15)|| (min_time == 25)|| (min_time == 35)||
(min_time == 45)|| (min_time == 55)
   if (min_time == c1(1,5)) || (c1(1,5)- c(1,5) == 1)
      disp ('Reading Now...')
      % ================================================
      s = serial('COM6','BaudRate',57600);    % Set port@COM4 and BaudRate
      set(s,'TimeOut',90);                % Determine Timeout    %Set timeout for s
      fopen(s)                          % Open port
      save1 = fread(s,37*pck);             % Read and Save value to save1
      fclose(s)                         % Close port
      delete(s)
      clear s


      disp(['End of Reading # ',num2str(Cp)])
```

```
    Cp = Cp+1;
    % ========================================


    Pos_1 = find(save1 == 126);    PD1 = Pos_1(1,1);
    Pos_2 = find(save1 == 66);     PD2 = Pos_2(1,1);
    Pos_3 = find(save1 == 255);    PD3 = Pos_3(1,1);


    if (PD2-PD1 == 1) && (PD3-PD2 == 1)
       %disp(['Saved! at: ',num2str(c(4:5))])
    % for rd = 1:round
       Data_Base(:,1) = save1;
       %save('DataBase_201009.txt','Data_Base','-ascii','-tabs','-append');
       save('DataBase_012011.txt','Data_Base','-ascii','-tabs');
    % end
       round = round+1;
    else disp(['Fail # ',num2str(Fl)])
        Fl = Fl+1;
    end
% %========================== End of Part 1 ==========================
%========================== Part 2:: Extract Data ==========================
    load DataBase_012011.txt;
    DTB = DataBase_012011;
    clear DataBase_012011;
    % pck = 12;
    % round = 1;
    PP3 = 1;   % Used in line 20
    PP1 = 1;   % Used in line 31
    % Choose the channel %
```

```
    CH = 7;
%       temp_CH = 8;
%       pH_CH = 2;
%       DO_CH = 5;
Pos_1 = find(DTB(:,:) == 126);     %Find position of 126
Pos_2 = find(DTB(:,:) == 66);      %Find position of 66
[rw_p1 cl_p1] = size(Pos_1);       %Given  rw_p1 = row of Pos_1
%===Extract only header-126===     %Find Header Position
for P1 = 1:2:rw_p1
   Pos_H1(PP1,:) = Pos_1(P1,1);
   PP1 = PP1+1;
end
%=================================%
[rw_pH1 cl_pH1] = size(Pos_H1);    %Given  rw_pH1 = row of Pos_H1
[rw_p2 cl_p2] = size(Pos_2);       %Given  rw_p2 = row of Pos_2
[rw_tm cl_tm] = size(DTB);         %Given  rw_tm = row of DTB
% Packet classification ==========================
pt2 = 1;
% pt_temp = 1;
% pt_pH = 1;
% pt_DO = 1;
for tm = 1:cl_tm
% tm = 3;
%    for pt = 1:pck
   for pt = 1:(rw_tm/37)-10
      Ex_pData(:,pt) = DTB((Pos_H1(pt,1):Pos_H1(pt,1)+36),tm);
      % extract only chose channel
      if (Ex_pData(13,pt) == CH)          %Channel = line 13
```

```
            Ex_Data(:,pt2) = Ex_pData(:,pt);
            pt2 = pt2 + 1;
          end
        end
        % ==========================================
         [dta pckCH] = size(Ex_Data);
    %    [dta1 count_pck_temp] = size(Ex_Data_temp);
    %    [dta2 count_pck_pH]   = size(Ex_Data_pH);
    %    [dta3 count_pck_DO]   = size(Ex_Data_DO);
        % % Extract packet ============================== Data only no header **
        n=1;
        for k = 1:pckCH;
          for m = 15:1:34;  % Data zone @ #15 - #34
            data(n) = Ex_Data(m,k);       %Data set [1x60]
            n = n+1;
          end
        end
%============================== End of Part 2 ==========================
%====================== Part 3 :: Generate ADC Data ======================
        Gen = repmat(256,1,n-1);          %Gen = Column of 256 = Column of data()
        MostBit = data.*Gen;              %Gen256x3 [1x60]    %data * 256
        [o Mbit] = size(MostBit);
        % Extract odd row from data-matrix
        y = 1;
        for q = 1:2:Mbit;
          RdataL(y) = data(q);
          y = y + 1;
        end
```

```
        % Extract even row from 256-matrix (MostBit)
        x = 1;
        for p = 2:2:Mbit;
          RdataH(x) = MostBit(p);
          x = x+1;
        end

        % === Sensor calibration ========== %
        Finaldata = RdataH+RdataL;
        save('ADCdata.txt','Finaldata','-ascii','-tabs');
%------------- Have ADC file only (not reading from sensor ------------%
%         load ADCdata_30_2.txt                                        %
%         Finaldata = ADCdata_30_2;                                    %
%         clear ADCdata_30_2;                                          %
%-------------------------------------------------------------------%
%================================ End of Part 3 ========================
%====================== Part 4 :: Sensor calibration ===================
        v_Temp = abs((-12.5*(Finaldata/2048-1))-3.7075);
        v_DO = 2.5 * Finaldata/4090;
        v_pH = 2.5 * Finaldata/4090;                  % For Ch A0-A6

        Temp = v_Temp/0.03878;
        DO = 3.27*v_DO - 0.727;
        pH = -3.838*v_pH + 13.72;
    end
%================================ End of Part 4 ========================
%====================== Part 5 :: Filter Data & Save ===================
        [o FinData] = size(Finaldata);
```

```
if mod(FinData,3) ~= 0
   FinData_2  = FinData - mod(FinData,3);
end
row_data = FinData_2/3;
TempData = zeros(row_data,1);
pHData   = zeros(row_data,1);
DOData   = zeros(row_data,1);
TP = 1;
PP = 1;
DP = 1;
%------------------ Find first Temp Position ----------------
temp_pos = 0;
pH_pos = 0;
M = 0;
while temp_pos == 0 && pH_pos == 0
   M = M + 1;
   if Temp(M) >= 0 && Temp(M) <= 80
      temp_pos = M;
      pH_pos = temp_pos + 1;
      DO_pos = temp_pos + 2;
   else
      if pH(M)>= 5 && pH(M) <= 9
         pH_pos = M;
         DO_pos = pH_pos + 1;
         temp_pos = pH_pos + 2;
      end
   end
end
```

```
%-------------------------------------------------------------
%------------------------ Filter Data ------------------------
for P=1:row_data;
   if temp_pos <= FinData && pH_pos <= FinData && DO_pos <= FinData
      % Choose temp data
      if Temp(temp_pos) >= 0 && Temp(temp_pos) <= 100
         TempData(TP,1) = Temp(temp_pos);
         TP = TP + 1;
      end
      % Choose pH data
      if pH(pH_pos) >= 0 && pH(pH_pos) <= 14
         pHData(PP,1) = pH(pH_pos);
         PP = PP + 1;
      end
      % Choose DO data
      if DO(DO_pos) >= 0 && DO(DO_pos) <= 15
         DOData(DP,1) = DO(DO_pos);
         DP = DP + 1;
      end
   end
   temp_pos = temp_pos + 3;
   pH_pos   = pH_pos + 3;
   DO_pos   = DO_pos + 3;
end
%------------------------ Save Data -------------------------
mean_temp = mean(TempData);
mean_pH = mean(pHData);
mean_DO = mean(DOData);
```

```
        date_time = clock;
        mat_data = [date_time(1:5),mean_temp,mean_pH,mean_DO];
        %write year
        dlmwrite('Result.dat',mat_data(1),'-append','delimiter','\t');
        %check MM DD HH mm
        for mat_pos = 2:5
          if mat_data(mat_pos) < 10
            zero = '0';
            dlmwrite('Result.dat',zero,'-append','delimiter','\t','coffset',1);
            dlmwrite('Result.dat',mat_data(mat_pos),'-append','delimiter','\t','coffset',-1);
          else
            dlmwrite('Result.dat',mat_data(mat_pos),'-append','delimiter','\t','coffset',1);
          end
        end

        for mat_pos = 6:8
          if mat_data(mat_pos) < 10
            zero = '0';
            dlmwrite('Result.dat',zero,'-append','delimiter','\t','coffset',2);
            if mat_pos ~= 8
              dlmwrite('Result.dat', mat_data(mat_pos),'-append','delimiter','\t','precision',
'%.4f','coffset',-1);
            else
              dlmwrite('Result.dat', mat_data(mat_pos),'-append','delimiter','\t','precision',
'%.4f','coffset',-1,'newline','pc');
            end
          else
            if mat_pos ~= 8
```

```
                dlmwrite('Result.dat', mat_data(mat_pos),'-append','delimiter','\t','precision',
'%.4f','coffset',1.5);
              else
                dlmwrite('Result.dat', mat_data(mat_pos),'-append','delimiter','\t','precision',
'%.4f','coffset',1.5,'newline','pc');
              end
            end
          end
          disp(['Saved at ',num2str(date_time(4)),':',num2str(date_time(5))])
          disp('----------------')
%     end
%     pause(300)  %pause compile 5 minutes = 300 secs
end
%==========================================-- E N D ==============================
```


ภาคผนวก ข

SMSLibx Library

SMSLibX is an SMS ActiveX component (DLL library) to send and receive SMS from PC via GSM modem/phone.

SMSLibX is a product for **software developers and integrators**; it can be easily integrated into Windows applications source code.

Commercial versions

- **Software only**

  For users who already own a **compatible GSM modem or phone.**

- **Software + modem**

  For users needing a complete solution.

  The following modems are available:

  - **Cinterion MC55i** (ex Siemens) - GPRS serial modem
  - **Digicom Wave GPRS** - GPRS modem USB

License policy

- You must buy one **software license** for each GSM device you have to interact with. Each license is bound to the **IMEI code** of the GSM device in use.
- See further details about **software license policy** here.

Features

- **Bi-directional SMS communication** (SMS sending and receival)
- **Tracking SMS delivery** (delivery receipt / SMS status report)
- **Text or Unicode SMS**
- **Binary SMS** (e.g. logos, ringing tones etc.)
- **Concatenated SMS** (longer than 160 characters)
- **Flash SMS** (immediately displayed on phone)
- **SMS options**: validity period, class, alphabet...
- SIM messages management
- **SIM phonebook management**
- Alert on **incoming calls** (and subsequent reject)
- **Call forwarding** to a different phone number
- GSM signal quality measuring

- Modem information retrieval
- **Multi-modem support** (up to 8 at a time)

Compatibility

- **Operating system**:

  Windows 98, ME, 2000, XP, Vista, 7; Server NT 4.0, 2000, 2003, 2008.
- **Programming environment**: MS Visual Basic, Visual C++, .NET C#, ASP, MS Access, Excel, Borland Delphi/C++... and any other environment supporting **ActiveX** integration.
- **Any SIM card** for GSM mobile phones.
- **Any GSM operator**.
- **Requires a GSM device** compliant with ETSI GSM 07.05/07 specification (3GPP 27.005/27.007).
- **Successfully tested with** the following GSM modem/phones:

  Audiotel Industrial, Modex, Celline,
  Digicom Wave GPRS,
  Falcom A2D,
  Huawei E220,
  Lightspeed 180M,
  Nokia 6210, 7110,
  Siemens MC35, MC35i, TC35, TC35i,
  Wavecom WMOD2B
  and others.

Documentation and resources

- **User manual & reference**
- Software tools for testing puroses

- **Download resources**:
  - **Free software demo**
  - **SMS integration tutorials & examples** on several languages

SMSLibX can send/receive from 6 to 20 sms/minute.

This means **thousands messages per day** (approx. 5,000 to 25,000).

Actual performances can vary according to several factors as: hardware device in use, environment factors (GSM network) and usage mode.

Performances

SMSLibX can send/receive from 6 to 20 sms/minute.

This means **thousands messages per day** (approx. 5,000 to 25,000).

Actual performances can vary according to several factors as: hardware device in use, environment factors (GSM network) and usage mode.

Most important factors are:

- quality of GSM signal coverage on your modem location
- efficiency of SMS Service Centers (SMSC) provided by your GSM operator
- sending time: SMS traffic on peak hours may be slower
- traffic direction: uni-directional traffic (just sending or receiving SMS) is faster than bi-directional traffic (sending+receiving SMS at a time).

ภาคผนวก ค

คู่มือเซ็นเซอร์

Dissloved Oxygen Sensor

Dissolved Oxygen Probe

(Order Code DO-BTA)

The Dissolved Oxygen Probe can be used to measure the concentration of dissolved oxygen in water samples tested in the field or in the laboratory. You can use this sensor to perform a wide variety of tests or experiments to determine changes in dissolved oxygen levels, one of the primary indicators of the quality of an aquatic environment:

- Monitor dissolved oxygen in an aquarium containing different combinations of plant and animal species.
- Measure changes in dissolved oxygen concentration resulting from photosynthesis and respiration in aquatic plants.
- Use this sensor for an accurate on-site test of dissolved oxygen concentration in a stream or lake survey, in order to evaluate the capability of the water to support different types of plant and animal life.
- Measure Biological Oxygen Demand (B.O.D.) in water samples containing organic matter that consumes oxygen as it decays.
- Determine the relationship between dissolved oxygen concentration and temperature of a water sample.

Inventory of Items Included with the Dissolved Oxygen Probe

Check to be sure that each of these items is included in your Dissolved Oxygen Probe box:

- Dissolved Oxygen Probe (dissolved oxygen electrode with membrane cap)
- One replacement membrane cap
- Sodium Sulfite Calibration Standard (2.0 M Na_2SO_3) and MSDS sheet
- D.O. Electrode Filling Solution, MSDS sheet, and filling pipet
- Calibration bottle (empty, lid with hole)
- D.O. Polishing Strips (1 pkg)
- Dissolved Oxygen Probe booklet

Collecting Data with the Conductivity Probe

This sensor can be used with the following interfaces to collect data:

- Vernier LabQuest® as a standalone device or with a computer
- Vernier LabQuest® Mini with a computer
- Vernier LabPro® with a computer, TI graphing calculator, or Palm® handheld
- Vernier Go!®Link
- Vernier EasyLink®
- Vernier SensorDAQ®
- CBL 2™

Do I Need to Calibrate the Dissolved Oxygen Probe?

It is not always necessary to perform a new calibration when using the Dissolved Oxygen Probe in the classroom. If your experiment or application is looking only at a change in dissolved oxygen, then the software's stored calibration is all you need. If you are making discrete measurements, such as taking readings in a stream or lake, and you want to improve the accuracy of your measurements, then it is best to perform a new calibration.

Preparing the Dissolved Oxygen Probe for Use

Part A: Probe Preparation

1. Prepare the Dissolved Oxygen Probe for use.
   a. Remove the blue protective cap from the tip of the probe. This protective cap can be discarded once the probe is unpacked.
   b. Unscrew the membrane cap from the tip of the probe.

   c. Using a pipet, fill the membrane cap with 1 mL of DO Electrode Filling Solution.
   d. Carefully thread the membrane cap back onto the electrode.
   e. Place the probe into a beaker filled with about 100 mL of distilled water.

Part B: Probe Warm-up

2. Connect the Dissolved Oxygen Probe to the interface. [1]
3. It is necessary to warm up the Dissolved Oxygen Probe for 10 minutes before taking readings. To warm up the probe, leave it in the water and connected to the interface with the data collection program running for 10 minutes. The probe must stay connected at all times to keep it warmed up. If disconnected for a few minutes, it will be necessary to warm up the probe again. [2]

[1] If your system does not support auto-ID, open an experiment file in Logger *Pro* or set up the sensor manually.

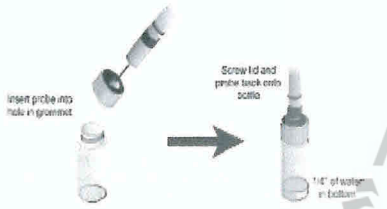
[2] The polarization process is slightly different for EasyLink because the Dissolved Oxygen Probe receives power only when the calculator is on and EasyData is running. If the calculator goes to sleep, the Dissolved Oxygen Probe does not receive power. To work around this, navigate to the Live Calibration screen, where the sensor will receive constant power, and leave it there for the duration of the warm-up period.

Part C: Probe Calibration

4. You are now ready to choose the calibration method for the Dissolved Oxygen Probe.
   - If you wish to use the stored calibration, proceed to Part D.
   - If you wish to perform a new calibration for the Dissolved Oxygen Probe, follow this procedure:

   a. Enter the calibration routine for your data collection program.[1]
   b. **First Calibration Point:** Remove the probe from the water and place the tip of the probe into the Sodium Sulfite Calibration Solution.
   c. When the displayed voltage reading stabilizes, enter 0 (the known dissolved oxygen value in mg/L).
   d. **Second Calibration Point:** Rinse the probe with distilled water and gently blot dry.
   e. Unscrew the lid of the calibration bottle provided with the probe. Slide the lid and the grommet about ½ inch onto the probe body.

   f. Add water to the bottle to a depth of about ¼ inch and screw the bottle into the cap, as shown. **Important:** Do not touch the membrane or get it wet during this step. Keep the probe in this position for about a minute.
   g. When the displayed voltage reading stabilizes, enter the correct saturated dissolved oxygen value (in mg/L) from Table 1 found on page 5–6 using the current barometric pressure and air temperature values. If you do not have the current air pressure, use Table 2 found on page 7 to estimate the air pressure at your altitude.



[1] If using an EasyLink and EasyData, you will need to press a button on the calculator every few minutes in order to keep the calculator and EasyData active and providing power to the sensor.

Part D: Collecting Data

5. You are now ready to collect dissolved oxygen concentration data.
   a. Place the tip of the probe into the water being tested (submerge 4–6 cm). Do not completely submerge. The handle is not waterproof.
   b. Gently stir the probe in the water sample. Monitor the dissolved oxygen concentration in the live readouts. **Note:** It is important to keep stirring the probe in the water sample. There must always be water flowing past the probe tip when you are taking measurements. As the probe measures the concentration of dissolved oxygen, it removes oxygen from the water at the junction of the probe membrane. If the probe is left still in calm water, reported DO readings will appear to be dropping.

Specifications

Range:	0 to 15 mg/L (or ppm)
Accuracy:	±0.2 mg/L
Resolution	
13-bit resolution (SensorDAQ):	0.007 mg/L
12-bit resolution (LabPro, LabQuest, LabQuest Mini, Go!Link, ULI II, SBI):	0.014 mg/L
10-bit resolution (CBL 2):	0.056 mg/L
Response Time:	95% of final reading in 30 seconds, 98% in 45 seconds
Temperature Compensation:	automatic from 5–35°C
Pressure Compensation:	manual, accounted for during calibration
Salinity Compensation:	manual, accounted for during calibration
Minimum sample flow:	20 cm/second
Stored Calibration Values	
Slope =	3.27
Intercept =	–0.327

This sensor is equipped with circuitry that supports auto-ID. When used with LabPro, LabQuest, LabQuest Mini, Go! Link, SensorDAQ, EasyLink, or CBL 2, the data-collection software identifies the sensor and uses pre-defined parameters to configure an experiment appropriate to the recognized sensor.

Table 1: Dissolved oxygen (mg/L) in air-saturated distilled water (at various temperature and pressure values)

	770 mm	760 mm	750 mm	740 mm	730 mm	720 mm	710 mm	700 mm
0°C	14.76	14.57	14.38	14.19	13.99	13.80	13.61	13.42
1°C	14.38	14.19	14.00	13.82	13.63	13.44	13.26	13.07
2°C	14.01	13.82	13.64	13.46	13.28	13.10	12.92	12.73
3°C	13.65	13.47	13.29	13.12	12.94	12.76	12.59	12.41
4°C	13.31	13.13	12.96	12.79	12.61	12.44	12.27	12.10
5°C	12.97	12.81	12.64	12.47	12.30	12.13	11.96	11.80
6°C	12.66	12.49	12.33	12.16	12.00	11.83	11.67	11.51
7°C	12.35	12.19	12.03	11.87	11.71	11.55	11.39	11.23
8°C	12.05	11.90	11.74	11.58	11.43	11.27	11.11	10.96
9°C	11.77	11.62	11.46	11.31	11.16	11.01	10.85	10.70
10°C	11.50	11.35	11.20	11.05	10.90	10.75	10.60	10.45
11°C	11.24	11.09	10.94	10.80	10.65	10.51	10.36	10.21
12°C	10.98	10.84	10.70	10.56	10.41	10.27	10.13	9.99
13°C	10.74	10.60	10.46	10.32	10.18	10.04	9.90	9.77
14°C	10.51	10.37	10.24	10.10	9.96	9.83	9.69	9.55
15°C	10.29	10.15	10.02	9.88	9.75	9.62	9.48	9.35
16°C	10.07	9.94	9.81	9.68	9.55	9.42	9.29	9.15
17°C	9.86	9.74	9.61	9.48	9.35	9.22	9.10	8.97
18°C	9.67	9.54	9.41	9.29	9.16	9.04	8.91	8.79
19°C	9.47	9.35	9.23	9.11	8.98	8.86	8.74	8.61
20°C	9.29	9.17	9.05	8.93	8.81	8.69	8.57	8.45
21°C	9.11	9.00	8.88	8.76	8.64	8.52	8.40	8.28
22°C	8.94	8.83	8.71	8.59	8.48	8.36	8.25	8.13
23°C	8.78	8.66	8.55	8.44	8.32	8.21	8.09	7.98
24°C	8.62	8.51	8.40	8.28	8.17	8.06	7.95	7.84
25°C	8.47	8.36	8.25	8.14	8.03	7.92	7.81	7.70
26°C	8.32	8.21	8.10	7.99	7.89	7.78	7.67	7.56
27°C	8.17	8.07	7.96	7.86	7.75	7.64	7.54	7.43
28°C	8.04	7.93	7.83	7.72	7.62	7.51	7.41	7.30
29°C	7.90	7.80	7.69	7.59	7.49	7.39	7.28	7.18
30°C	7.77	7.67	7.57	7.47	7.36	7.26	7.16	7.06
31°C	7.64	7.54	7.44	7.34	7.24	7.14	7.04	6.94
32°C	7.51	7.42	7.32	7.22	7.12	7.03	6.93	6.83
33°C	7.39	7.29	7.20	7.10	7.01	6.91	6.81	6.72
34°C	7.27	7.17	7.08	6.98	6.89	6.80	6.70	6.61
35°C	7.15	7.05	6.96	6.87	6.78	6.68	6.59	6.50

Table 1 (cont.) : Dissolved oxygen (mg/L) in air-saturated distilled water (at various temperature and pressure values)

	690 mm	680 mm	670 mm	660 mm	650 mm
0°C	13.23	13.04	12.84	12.65	12.46
1°C	12.88	12.70	12.51	12.32	12.14
2°C	12.55	12.37	12.19	12.01	11.82
3°C	12.23	12.05	11.88	11.70	11.52
4°C	11.92	11.75	11.58	11.40	11.23
5°C	11.63	11.46	11.29	11.12	10.95
6°C	11.34	11.18	11.01	10.85	10.68
7°C	11.07	10.91	10.75	10.59	10.42
8°C	10.80	10.65	10.49	10.33	10.18
9°C	10.55	10.39	10.24	10.09	9.94
10°C	10.30	10.15	10.00	9.86	9.71
11°C	10.07	9.92	9.78	9.63	9.48
12°C	9.84	9.70	9.56	9.41	9.27
13°C	9.63	9.49	9.35	9.21	9.07
14°C	9.42	9.28	9.14	9.01	8.87
15°C	9.22	9.08	8.95	8.82	8.68
16°C	9.02	8.89	8.76	8.63	8.50
17°C	8.84	8.71	8.58	8.45	8.33
18°C	8.66	8.54	8.41	8.28	8.16
19°C	8.49	8.37	8.24	8.12	8.00
20°C	8.33	8.20	8.08	7.96	7.84
21°C	8.17	8.05	7.93	7.81	7.69
22°C	8.01	7.90	7.78	7.67	7.55
23°C	7.87	7.75	7.64	7.52	7.41
24°C	7.72	7.61	7.50	7.39	7.28
25°C	7.59	7.48	7.37	7.26	7.15
26°C	7.45	7.35	7.24	7.13	7.02
27°C	7.33	7.22	7.12	7.01	6.90
28°C	7.20	7.10	6.99	6.89	6.78
29°C	7.08	6.98	6.87	6.77	6.67
30°C	6.96	6.86	6.76	6.66	6.56
31°C	6.85	6.75	6.65	6.55	6.45
32°C	6.73	6.63	6.54	6.44	6.34
33°C	6.62	6.53	6.43	6.33	6.24
34°C	6.51	6.42	6.32	6.23	6.13
35°C	6.40	6.31	6.22	6.13	6.03

Elevation Barometric Pressure Table

If you do not have a barometer available to read barometric pressure, you can estimate the barometric pressure reading at your elevation (in feet) from Table 2. The values are calculated based on a barometric air pressure reading of 760 mm Hg at sea level.

Table 2: Approximate Barometric Pressure at Different Elevations

Elevation (feet)	Pressure (mm Hg)	Elevation (feet)	Pressure (mm Hg)	Elevation (feet)	Pressure (mm Hg)
0	760	2000	708	4000	659
250	753	2250	702	4250	653
500	746	2500	695	4500	647
750	739	2750	689	4750	641
1000	733	3000	683	5000	635
1250	727	3250	677	5250	629
1500	720	3500	671	5500	624
1750	714	3750	665	5750	618

How the Dissolved Oxygen Probe Works

The Vernier Dissolved Oxygen Probe is a Clark-type polarographic electrode that senses the oxygen concentration in water and aqueous solutions. A platinum cathode and a silver/silver chloride reference anode in KCl electrolyte are separated from the sample by a gas-permeable plastic membrane.

Figure 1

A fixed voltage is applied to the platinum electrode. As oxygen diffuses through the membrane to the cathode, it is reduced:

$$\tfrac{1}{2}\,O_2 + H_2O + 2e^- \longrightarrow 2\,OH^-$$

The oxidation taking place at the reference electrode (anode) is:

$$Ag + Cl^- \longrightarrow AgCl + e^-$$

Accordingly, a current will flow that is proportional to the rate of diffusion of oxygen, and in turn to the concentration of dissolved oxygen in the sample. This current is converted to a proportional voltage, which is amplified and read by any of the Vernier lab interfaces.

Storage and Maintenance of the Dissolved Oxygen Probe

Following the procedure outlined in this section will enhance the lifetime of your Dissolved Oxygen Probe and its membrane cap. Follow these steps when storing the electrode:

- **Long-term storage (more than 24 hours):** Remove the membrane cap and rinse the inside and outside of the cap with distilled water. Shake the membrane cap dry. Also rinse and dry the exposed anode and cathode inner elements (blot dry with a lab wipe). Reinstall the membrane cap loosely onto the electrode body for storage. Do not screw it on tightly.
- **Short-term storage (less than 24 hours):** Store the Dissolved Oxygen Probe with the membrane cap submerged in about 1 inch of distilled water.
- **Polishing the metal electrodes:** If the cathode (the small, shiny, metal contact in the center of the glass stem shown in Figure 1) and anode (the silver, metal foil surrounding the lower portion of the inner body) become discolored or appear corroded, polish them with the polishing strip that is provided with the probe. Perform this operation only as needed to restore electrode performance—it should be necessary only once every year or so. Remove the membrane cap from your Vernier Dissolved Oxygen Probe. Thoroughly rinse the inner elements of the probe with distilled water to remove all filling solution. Cut a one-inch piece from the D.O. Electrode Polishing Strip provided. Wet the dull (abrasive) side of the polishing strip with distilled water. Using a circular motion, polish the center glass element of the cathode (on the very end of the electrode). Use gentle finger pressure during this polishing operation. Polish only enough to restore a bright, clean surface to the center element. Next, polish the silver anode located around the base of the electrode inner element. Polish only enough to restore a silver appearance. Note: Aggressive polishing will damage the probe inner elements. *Be sure* to use only gentle pressure when performing the polishing of the anode and cathode. When you have completed the polishing, rinse the cathode and anode elements thoroughly and dry with a lab wipe.

With normal use, the Vernier Dissolved Oxygen Probe will last for years. The membrane cap will, however, require replacement after about 6 months of continuous use. Replacement of the membrane is recommended when your Dissolved Oxygen Probe will no longer respond rapidly during calibration or when taking D.O. readings. Use of your Dissolved Oxygen Probe in samples that are non-aqueous or in those that contain oil, grease, or other coating agents will result in shortened membrane life. Replacement membranes can be obtained from Vernier (order code MEM).

Automatic Temperature Compensation

Your Vernier Dissolved Oxygen Probe is automatically temperature compensated, using a thermistor built into the probe. The temperature output of this probe is used to automatically compensate for changes in permeability of the membrane with changing temperature. If the probe was not temperature compensated, you would notice a change in the dissolved oxygen reading as temperature changed, even if the actual concentration of dissolved oxygen in the solution did not change. Here are two examples of how automatic temperature compensation works:

- If you calibrate the Dissolved Oxygen Probe in the lab at 25°C and 760 mm Hg barometric pressure (assume salinity is negligible), the value you enter for the saturated oxygen calibration point would be 8.36 mg/L (see Table 1). If you were to take a reading in distilled water that is saturated with oxygen by rapid stirring, you would get a reading of 8.36 mg/L. *If* the water sample were then cooled to 10°C with no additional stirring, the water would no longer be saturated (cold water can hold more dissolved oxygen than warm water). So, the reading of the temperature-compensated Dissolved Oxygen Probe would still be 8.36 mg/L.
- If, however, the solution was cooled to 10°C *and* continually stirred so it remained saturated by dissolving additional oxygen, the temperature-compensated probe would give a reading of 11.35 mg/L—the value shown in Figure 2. Note: Temperature compensation *does not mean* that the reading for a saturated solution will be the same at two different temperatures—the two solutions have different concentrations of dissolved oxygen, and the probe reading should reflect this difference.

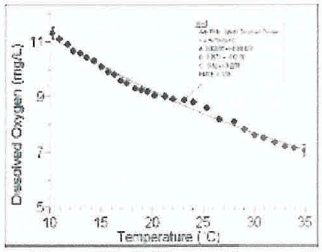


Figure 2: Saturated Dissolved Oxygen vs. Temperature Data

Sampling in Ocean Salt Water or Tidal Estuaries
(at salinity levels greater than 1000 mg/L)

Dissolved Oxygen concentration for air saturated water at various salinity values, DO_{salt}, can be calculated using the formula:

$$DO_{salt} = DO - (k \cdot S)$$

- DO_{salt} is the concentration of dissolved oxygen (in mg/L) in salt-water solutions.
- DO is the dissolved oxygen concentration for air-saturated distilled water as determined from Table 1.
- S is the salinity value (in ppt). Salinity values can be determined using the Vernier Chloride Ion-Selective Electrode or Conductivity Probe as described in the *Water Quality with Vernier* lab manual.
- k is a constant. The value of k varies according to the sample temperature, and can be determined from Table 3.

Table 3: Salinity Correction Constant Values

Temp. (°C)	Constant, k	Temp. (°C)	Constant, k	Temp. (°C)	Constant, k	Temp. (°C)	Constant, k
1	0.08796	8	0.06916	15	0.05602	22	0.04754
2	0.08466	9	0.06697	16	0.05456	23	0.04662
3	0.08164	10	0.06479	17	0.05328	24	0.04593
4	0.07911	11	0.06286	18	0.05201	25	0.04499
5	0.07646	12	0.06124	19	0.05073	26	0.04425
6	0.07391	13	0.05931	20	0.04964	27	0.04361
7	0.07135	14	0.05757	21	0.04854	28	0.04296

Example: Determine the saturated DO calibration value at a temperature of 23°C and a pressure of 750 mm Hg, when the Dissolved Oxygen Probe is used in seawater with a salinity value of 35.0 ppt.

First, find the dissolved oxygen value in Table 1 (DO = 8.55 mg/L). Then find k in Table 3 at 23°C (k = 0.04662). Substitute these values, as well as the salinity value, into the previous equation:

$$DO_{salt} = DO - (k \cdot S) = 8.55 - (0.04662 \times 35.0) = 8.55 - 1.63 = 6.92 \text{ mg/L}$$

Use the value 6.92 mg/L when performing the saturated DO calibration point (water-saturated air), as described in Step 6. The Dissolved Oxygen Probe will now be calibrated to give correct DO readings in salt-water samples with a salinity of 35.0 ppt.

Important: For most dissolved oxygen testing, it is not necessary to compensate for salinity; for example, if the salinity value is 0.5 ppt, using 25°C and 760 mm Hg, the calculation for $DO_{(s)}$ would be:

$$DO_{salt} = DO - (k \cdot S) = 8.36 - (0.04498 \times 0.5) = 8.36 - 0.023 = 8.34 \text{ mg/L}$$

At salinity levels less than 1.0 ppt, neglecting this correction results in an error of less than 0.2%.

Maintaining and Replenishing the Sodium Sulfite Calibration Solution

Having an oxygen-free solution to perform a zero-oxygen calibration point is essential for accurate readings with your Dissolved Oxygen Probe. The Sodium Sulfite Calibration Solution that was included with your probe will last a long time, but not indefinitely. Here are some suggestions for maintaining and replacing this solution:

- After your first use of the solution for calibration, the solution will no longer be brim full (some overflow results when the probe is inserted into the solution). If you cap the solution with an air space above the probe, oxygen gas in the space will dissolve in the sodium sulfite solution—as a result, the solution may not be oxygen free. To prevent this from occurring, before putting on the lid, gently squeeze the bottle so the level of the solution is at the very top of the bottle neck; with the solution at this level, screw on the lid. The bottle will remain in this "collapsed" position. Using this procedure, the 2.0 M Na_2SO_3 should remain oxygen free for a long period of time. If the calibration voltage reading displayed during the first calibration point is higher than in previous calibrations, it may be time to replace the solution, as described below.
- The 2.0 M sodium sulfite (Na_2SO_3) solution can be prepared from solid sodium sulfite crystals: Add 25.0 g of solid anhydrous sodium sulfite crystals (Na_2SO_3) to enough distilled water to yield a final volume of 100 mL of solution. The sodium sulfite crystals do not need to be reagent grade; laboratory grade will work fine. Many high school chemistry teachers will have this compound in stock. Prepare the solution 24 hours in advance of doing the calibration to ensure that all oxygen has been depleted. If solid sodium sulfite is not available, you may substitute either 2.0 M sodium hydrogen sulfite solution, (sodium bisulfite, 20.8 g of $NaHSO_3$ per 100 mL of solution) or 2.0 M potassium nitrite (17.0 g of KNO_2 per 100 mL of solution).

Replacement Parts

Vernier Software & Technology

Replacement Membrane Caps	MEM
Polishing Strips (pkg of 2)	PS
D.O. Probe Filling Solution	FS
D.O. Probe Calibration Solution	DO-CAL

Flinn Scientific (P.O. Box 219, Batavia, IL 60510, Tel: 800-452-1261)

sodium sulfite standard solution, 2.0 M, 65 mL bottle	Order Number: SO426
sodium sulfite, anhydrous solid, 500 g bottle	Order Number: SO111

Using the Dissolved Oxygen Probe with Other Vernier Sensors

It is very important for you to know that the Dissolved Oxygen Probe will interact with some other Vernier sensors and probes, *if* they are placed in the same solution (in the same aquarium or beaker, for example), *and* they are connected to the same interface box (e.g., the same LabPro interface). This situation arises because the Dissolved Oxygen Probe outputs a signal in the solution, and this signal can affect the reading of another probe.

The following probes *cannot* be connected to the same interface as a Dissolved Oxygen Probe and placed in the same solution:

- Conductivity Probe
- pH Sensor
- Direct-Connect Temperature Probe
- Ion Selective Electrodes

If you wish to take simultaneous readings using any of the probe combinations listed above, here are some alternative methods:

- To take simultaneous dissolved oxygen and conductivity or dissolved oxygen and pH readings, you can connect the probes to two different interfaces. If the two probes in question are connected to separate interfaces, the two probes will read correctly in the same solution.
- The Vernier Stainless Steel Temperature Probe (also shipped with CBL 2) can be used in the same container with the Dissolved Oxygen Probe.
- If you are sampling a lake or stream and want to use two of the probes with a single interface, you can connect the two probes in question to the same interface and load their respective calibrations. Place one probe in the water first and take its reading. Then remove it and place the second probe in the solution to take its reading.

Background Information about Dissolved Oxygen

Dissolved oxygen is a vital substance in a healthy body of water. Various aquatic organisms require different levels of dissolved oxygen to survive. Whereas trout require higher levels of dissolved oxygen, fish species like carp and catfish survive in streams with low oxygen concentrations. Water with a high level of dissolved oxygen is generally considered to be a healthy environment that can support many different types of aquatic life.

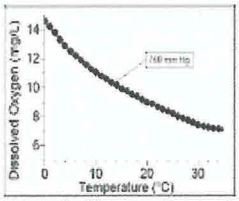


Figure 3: Saturated Dissolved Oxygen vs. Temperature at 760 mm Hg

There are many factors that can affect the level of dissolved oxygen in a body of water. Turbulence from waves on a lake or from a fast-moving stream can greatly increase the amount of water exposed to the atmosphere, resulting in higher levels of dissolved oxygen. Water temperature is another factor that can affect dissolved-oxygen levels; like other gases, the saturated level of dissolved oxygen is less in warm water than in cold water, shown in Figure 3.

Photosynthesis cycles also have a large effect on dissolved oxygen levels of an aquatic environment. Aquatic plants and photosynthetic microorganisms will cause oxygen gas to be produced during daylight hours from photosynthesis:

$$CO_2 + H_2O \longrightarrow C_xH_yO_z + O_2$$

As the afternoon progresses, dissolved-oxygen levels increase as photosynthesis occurs. After sundown, photosynthesis decreases—however, plant and animal organisms continue to respire. Throughout the night and early morning, respiration results in a decrease in dissolved-oxygen levels:

$$C_xH_yO_z + O_2 \longrightarrow CO_2 + H_2O$$

The amount and variety of plant and animal life in a stream affects the degree to which the photosynthesis-respiration cycle occurs.

Levels of organic wastes from manmade sources such as pulp mills, food-processing plants, and wastewater treatment plants can also result in lower levels of dissolved oxygen in streams and lakes. Oxidation of these wastes depletes the oxygen, sometimes at a faster rate than turbulence or photosynthesis can replace it. Thus, use of a Dissolved Oxygen Probe to determine dissolved oxygen concentration and biological oxygen demand of a stream can be important tests in determining the health and stability of an aquatic ecosystem.

Calibrating and Monitoring Using Units of Percent Saturation

Instead of calibrating using units of mg/L (equal to parts per million or ppm), you may also choose to calibrate dissolved oxygen using units of % saturation. When doing a calibration for units of % saturation, the calibration point done in the sodium sulfite solution (zero oxygen) is assigned a value of 0%, and that for water-saturated air (or air-saturated water) is given a value of 100%. It must be noted, however, that 100% represents an oxygen-saturated solution only at that particular temperature, pressure, and salinity level. If you intend to compare your measured dissolved oxygen values with data collected under a different set of conditions, a preferable method would be to use units of mg/L (described earlier in this booklet).

If you have calibrated your Dissolved Oxygen Probe in units of mg/L, you can easily calculate percent saturation using the formula:

% saturation = (actual DO reading / Saturated DO reading from Table 1) x 100

For example, if your Dissolved Oxygen Probe gives a D.O. reading of 6.1 mg/L at a temperature of 20°C and a pressure of 740 mm Hg, look up the saturated dissolved oxygen reading in Table 1 (8.93 mg/L). The value for % saturation is:

% saturation = (6.1 / 8.93) x 100 = 68%

Warranty

All Vernier Dissolved Oxygen Probes are warranted to be free from defects in material and workmanship for a period of five years from purchase of the probe, provided the electrode has been used in accordance with this instruction manual and used under normal laboratory conditions. The warranty does not apply when the electrode has been subjected to accident, alternate use, misuse, or abuse in any manner. Consumables, including the membrane, associated with Vernier products are excluded from this warranty.

pH Sensor

pH Sensor
(Order Code PH-BTA)

Our pH Sensor can be used for any lab or demonstration that can be done with a traditional pH meter. This sensor offers the additional advantages of automated data collection, graphing, and data analysis. Typical activities using our pH sensor include studies of household acids and bases, acid-base titrations, monitoring pH change during chemical reactions or in an aquarium as a result of photosynthesis, investigations of acid rain and buffering, and investigations of water quality in streams and lakes.

Collecting Data with the pH Sensor

This sensor can be used with the following interfaces to collect data:

- Vernier LabQuest® as a standalone device or with a computer
- Vernier LabQuest® Mini with a computer
- Vernier LabPro® with a computer, TI graphing calculator, or Palm® handheld
- Vernier Go!®Link
- Vernier EasyLink®
- Vernier SensorDAQ®
- CBL 2™

Here is the general procedure to follow when using the pH Sensor:

1. Connect the pH Sensor to the interface.
2. Start the data-collection software[1].
3. The software will identify the pH Sensor and load a default data-collection setup. You are now ready to collect data.

Important: Do not fully submerge the sensor. The handle is not waterproof.

Data-Collection Software

This sensor can be used with an interface and the following data-collection software.

- **Logger *Pro* 3** This computer program is used with LabQuest, LabQuest Mini, LabPro, or Go!Link.
- **Logger *Pro* 2** This computer program is used with ULI or Serial Box Interface.
- **Logger Lite** This computer program is used with LabQuest, LabQuest Mini, LabPro, or Go!Link.
- **LabQuest App** This program is used when LabQuest is used as a standalone device.
- **EasyData App** This calculator application for the TI-83 Plus and TI-84 Plus can be used with CBL 2, LabPro, and Vernier EasyLink. We recommend version 2.0 or newer, which can be downloaded from the Vernier web site, www.vernier.com/easy/easydata.html, and then transferred to the calculator. See the Vernier web site, www.vernier.com/calc/software/index.html for more information on the App and Program Transfer Guidebook.
- **DataMate program** Use DataMate with LabPro or CBL 2 and TI-73, TI-83, TI-84, TI-86, TI-89, and Voyage 200 calculators. See the LabPro and CBL 2 Guidebooks for instructions on transferring DataMate to the calculator.
- **Data Pro** This program is used with LabPro and a Palm handheld.
- **LabVIEW** National Instruments LabVIEW™ software is a graphical programming language sold by National Instruments. It is used with SensorDAQ and can be used with a number of other Vernier interfaces. See www.vernier.com/labview for more information.

[1] If you are using Logger *Pro* 2 with either a ULI or SBI, the sensor will not auto-ID. Open an experiment file for the pH Sensor in the Probes & Sensors folder.

pH Electrode Specifications

Type:	Sealed, gel-filled, epoxy body, Ag/AgCl
Response time:	90% of final reading in 1 second
Temperature range:	5 to 80°C
12 mm OD	
Range:	pH 0–14
13-bit Resolution (SensorDAQ):	0.0025 pH units
12-bit Resolution (LabQuest, LabQuest Mini, Go!Link, LabPro, ULI, SBI):	0.005 pH units
10-bit Resolution (CBL 2):	0.02 pH units
Isopotential pH:	pH 7 (point at which temperature has no effect)
Output:	59.2 mV/pH at 25°C
Stored Calibration Values[2]:	
Intercept (k_0):	13.720
Slope (k_1):	–3.838

NOTE: This product is to be used for educational purposes only. It is not appropriate for industrial, medical, research, or commercial applications.

How the pH Sensor Works

The pH Amplifier inside the handle is a circuit which allows a standard combination pH electrode (such as the Vernier 7120B) to be monitored by a lab interface. The cable from the pH Amplifier ends in a BTA plug.

The pH Sensor will produce a voltage of 1.75 volts in a pH 7 buffer. The voltage will increase by about 0.25 volts for every pH number decrease. The voltage will decrease by about 0.25 volts/pH number as the pH increases.

The Vernier gel-filled pH Sensor is designed to make measurements in the pH range of 0 to 14. A polycarbonate body that extends below the glass sensing bulb of the

[2] These are average calibration values. Actual values may vary because sensors are individually calibrated by Vernier before shipping.

electrode makes this probe ideal for the demands of a middle school, high school, or university level science class or for making measurements in the environment. The gel-filled reference half cell is sealed—it never needs to be refilled.

This sensor is equipped with circuitry that supports auto-ID. When used with LabQuest, LabQuest Mini, LabPro, Go! Link, SensorDAQ, EasyLink, or CBL 2, the data-collection software identifies the sensor and uses pre-defined parameters to configure an experiment appropriate to the recognized sensor.

Preparing for Use

To prepare the electrode to make pH measurements, follow this procedure:

- Remove the storage bottle from the electrode by first unscrewing the lid, then removing the bottle and lid. Thoroughly rinse the lower section of the probe, especially the region of the bulb, using distilled or deionized water.
- When the probe is not being stored in the storage bottle, it can be stored for short periods of time (up to 24 hours) in pH-4 or pH-7 buffer solution. It should never be stored in distilled water.
- Connect the pH Sensor to your lab interface, load or perform a calibration (as described in the next section), and you are ready to make pH measurements. **Note:** Do not completely submerge the sensor. The handle is not waterproof.

When you are finished making measurements, rinse the tip of the electrode with distilled water. Slide the cap onto the electrode body, then screw the cap onto the storage bottle. **Note:** When the level of storage solution left in the bottle gets low, you can replenish it with small amounts of tap water the first few times you use the probe (but not indefinitely!). A better solution is to prepare a quantity of pH-4 buffer/KCl storage solution (see the section on Maintenance and Storage) and use it to replace lost solution.

Do I Need to Calibrate the pH Sensor?

We feel that you should not have to perform a new calibration when using the pH Sensor for most experiments in the classroom. We have set the sensor to match our stored calibration before shipping it. You can simply use the appropriate calibration file that is stored in your data-collection program from Vernier in any of these ways:

1. If you ordered the PH-BTA version of the sensor, and you are using it with a LabQuest, LabQuest Mini, LabPro or CBL 2 interface, then a calibration (in pH) is automatically loaded when the pH Sensor is connected. **Note:** Each pH Sensor (PH-BTA version) is calibrated at Vernier. This custom calibration is then stored on the sensor. This means that when you first use it, you will see pH readings that are accurate to +/- 0.10 pH units, without calibration! With time, you may see some minor loss of the initial custom calibration accuracy, but for most purposes (see below), it should not be necessary to calibrate the pH Sensor.
2. If you are using Logger *Pro* software (version 2.0 or newer) on a Macintosh or Windows computer, open an experiment file for the pH Sensor, and its stored calibration will be loaded at the same time. **Note:** If you have an earlier version of Logger *Pro*, a free upgrade is available from our web site.
3. Any version of the DataMate or EasyData program (with LabPro or CBL 2) has stored calibrations for this sensor.
4. Any version of Data Pro has stored calibrations for this sensor.

If you are performing a chemistry experiment, or doing water quality testing that requires a very accurate calibration, you can calibrate the Vernier pH Electrode following this procedure:

- Use the 2-point calibration option of the Vernier data-collection program. Rinse the tip of the electrode in distilled water. Place the electrode into one of the buffer solutions (e.g., pH 4). When the voltage reading displayed on the computer or calculator screen stabilizes, enter a pH value, "4".
- For the next calibration point, rinse the electrode and place it into a second buffer solution (e.g., pH 7). When the displayed voltage stabilizes, enter a pH value, "7".
- Rinse the electrode with distilled water and place it in the sample.

pH Buffer Solutions

In order to do a calibration of the pH Sensor, or to confirm that a saved pH calibration is accurate, you need to have a supply of pH buffer solutions that cover the range of pH values you will be measuring. We recommend buffer solutions of pH 4, 7, and 10.

- Vernier sells a pH buffer kit (order code PHB). The kit has 12 tablets: four tablets each of buffer pH 4, 7, and 10. Each tablet is added to 100 mL of distilled or deionized water to prepare respective pH buffer solutions.
- Flinn Scientific (www.flinnsci.com, Tel. 800-452-1261) sells a wide variety of buffer tablets and prepared buffer solutions.
- You can prepare your own buffer solutions using the following recipes:

pH 4.00	Add 2.0 mL of 0.1 M HCl to 1000 mL of 0.1 M potassium hydrogen phthalate.
pH 7.00	Add 582 mL of 0.1 M NaOH to 1000 mL of 0.1 M potassium dihydrogen phosphate.
pH 10.00	Add 214 mL of 0.1 M NaOH to 1000 mL of 0.05 M sodium bicarbonate.

Maintenance and Storage

Short-term storage (up to 24 hours): Place the electrode in pH-4 or pH-7 buffer solution.

Long-term storage (more than 24 hours): Store the electrode in a buffer pH-4/KCl storage solution in the storage bottle. The pH Electrode is shipped in this solution. Vernier sells 500 mL bottles of replacement pH Storage Solution (order code PH-SS), or you can prepare additional storage solution by adding 10 g of solid potassium chloride (KCl) to 100 mL of buffer pH-4 solution. Flinn Scientific (800- 452-1261) sells a Buffer Solution Preservative (order code B0175) that can be added to this storage solution. By storing the electrode in this solution, the reference portion of the electrode is kept moist. Keeping the reference junction moist adds to electrode longevity and retains electrode response time when the unit is placed back into

service. If the electrode is inadvertently stored dry (we don't recommend this!), immerse the unit in soaking solution for a minimum of eight hours prior to service.

When testing a pH Sensor, it is best to place it into a known buffer solution. This allows you to see if the sensor is reading correctly (e.g., in a buffer pH 7, is the sensor reading close to pH 7). Do not place your sensor into distilled water to check for readings—distilled water can have a pH reading anywhere between 5.5 and 7.0, due to variable amounts of carbon dioxide dissolved from the atmosphere. Furthermore, due to a lack of ions, the pH values reported with the sensor in distilled water will be erratic.

If your pH Sensor is reading slightly off of the known buffer pH (e.g., reads 6.7 in a buffer 7), you may simply need to calibrate the sensor. You can calibrate the sensor in two buffer solutions for two calibration points. If you do not remember or know how to perform a calibration, refer to the booklet that came with the pH sensor.

If your readings are off by several pH values, the pH readings do not change when moved from one buffer solution to another different buffer, or the sensor's response seems slow, the problem may be more serious. Sometimes a method called "shocking" is used to revive pH electrodes. To shock your pH Sensor, perform the following:

1. Let the pH Electrode soak for 4-8 hours in an HCl solution between 0.1 and 1.0 M.
2. Rinse off the electrode and let it sit in some buffer pH 7 for an hour or so.
3. Rinse the electrode and give it another try.

Mold growth in the buffer/KCl storage solution can be prevented by adding a commercial growth inhibitor. This mold will not harm the electrode and can easily be removed using a light detergent solution.

This sensor is designed to be used in aqueous solutions. The polycarbonate body of the sensor can be damaged by many organic solvents. In addition, do not use the sensor in solutions containing: perchlorates, silver ions, sulfide ions, biological samples with high concentrations of proteins, or Tris buffered solutions.[3] Do not use it in hydrofluoric acid or in acid or base solutions with a concentration greater than 1.0 molar. The electrode may be used to measure the pH of sodium hydroxide solutions with a concentration near 1.0 molar, but should not be left in this concentration of sodium hydroxide for periods longer than 5 minutes. Using or storing the electrode at very high temperatures or very low temperatures (near 0°C) can damage it beyond repair.

Warranty

Vernier warrants this product to be free from defects in materials and workmanship for a period of five years from the date of shipment to the customer. This warranty does not cover damage to the product caused by abuse or improper use. Additionally, the warranty does not cover accidental breakage of the glass bulb of the pH Sensor.