แบบจำลองการควบคุมออกซิเจนในน้ำอัตโนมัติสำหรับบ่อเลี้ยงกุ้ง

DISSOVED OXYGEN AUTOMATIC CONTROL MODEL FOR THE SHRIMP POND

ชวฤทธิ์ ภักดิ์สรสิทธิ์

ชัชวาล จันทร์แก่น

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2557

แบบจำลองการควบคุมออกซิเจนในน้ำอัตโนมัติสำหรับบ่อเลี้ยงกุ้ง
DISSOLVED OXYGEN AUTOMATIC CONTROL MODEL FOR THE SHRIMP POND

ชวฤทธิ์ ภักดิ์สรสิทธิ์
ชัชวาล จันทร์แก่น

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิศวกรรมการวัดคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2557

DISSOLVED OXYGEN AUTOMATIC CONTROL MODEL FOR THE SHRIMP POND

Chawarit Paksorrasit
Chatchawan Junkan

A THESIS SUBMITTED INPARTIAL FULFILLMENT
OF THE REQUIREMENT FOE THE DEGREE OF
BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING
FACULTY OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC 2014

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ใบรับรองปริญญานิพนธ์

.....

หัวข้อปริญญานิพนธ์ แบบจำลองการควบคุมออกซิเจนในน้ำอัตโนมัติสำหรับบ่อเลี้ยงกุ้ง
DISSOLVED OXYGEN AUTOMATIC CONTROL MODEL FOR THE SHRIMP POND

นักศึกษาผู้จัดทำ นายชวฤทธิ์ ภักดิ์สรสิทธ์ **รหัสนักศึกษา** 54010292
นายชัชวาล จันทร์แก่น **รหัสนักศึกษา** 54010301

ปริญญา วิศวกรรมศาสตร์บัณฑิต

สาขาวิชา วิศวกรรมการวัดคุม

ปีการศึกษา 2557

อาจารย์ควบคุมปริญญานิพนธ์	ลายมือชื่อ
อาจารย์นริทร์ ธรรมารักษ์วัฒนะ	

หัวข้อปริญญานิพนธ์	แบบจำลองการควบคุมออกซิเจนในน้ำอัตโนมัติสำหรับบ่อเลี้ยงกุ้ง DISSOLVED OXYGEN AUTOMATIC CONTROL MODEL FOR THE SHRIMP POND			
นักศึกษาผู้จัดทำ	นายชวฤทธิ์	ภักดิ์สรสิทธ์	**รหัสนักศึกษา**	54010292
	นายชัชวาล	จันทร์แก่น	**รหัสนักศึกษา**	54010301
อาจารย์ที่ปรึกษา	อาจารย์นริทร์	ธรรมารักษ์วัฒนะ		
ปีการศึกษา	2557			

บทคัดย่อ

โครงการนี้จัดทำขึ้นเพื่อทำการวิจัยและพัฒนาการควบคุมออกซิเจนในน้ำอัตโนมัติสำหรับบ่อเลี้ยงกุ้ง และเพื่อลดปริมาณการใช้พลังงานไฟฟ้าหรือน้ำมันเชื้อเพลิงลง ซึ่งจะช่วยให้ต้นทุนในการเพาะเลี้ยงกุ้งลดลง เช่นกัน การที่จะเพิ่มประสิทธิภาพการใช้พลังงาน ในการเพาะเลี้ยงกุ้งเพื่อให้กระบวนการผลิตสามารถประหยัดพลังงานได้สูงสุด และสร้างความยั่งยืนให้กับอุตสาหกรรมกุ้ง โดยไม่กระทบต่อสิ่งแวดล้อม รวมทั้งลดการใช้พลังงานส่วนเกินโดย ในการพัฒนาระบบได้อาศัย การใช้เทคโนโลยีด้านคอมพิวเตอร์มาประยุกต์การปิด-เปิด เครื่องตีน้ำให้เป็นอัตโนมัติ และใช้ค่าออกซิเจนละลายน้ำเป็นตัวควบคุมการทำงานของเครื่องตีน้ำใน บ่อเลี้ยงกุ้ง

Thesis Title	DISSOLVED OXYGEN AUTOMATIC CONTROL MODEL FOR THE SHRIMP POND
Authors	Mr. Chawarit Paksorrasit
	Mr. Chatchawan Junkan
Thesis Advisor	Mr. Narin Tammarugwattana
Year	2014

Abstract

This project is intended to continue research and development of dissolved oxygen automatic control for in shrimp pond. To reduce electrical energy consumption or fuel that would allow the cost of shrimp farming had decreased as well. To increase energy efficiency in the shrimp so that it can process a maximum of energy savings.

And establish sustainability for the shrimp industry without affecting the environment, including the reduction of excess energy. In the development of computer technology would be applied to the on-off water wheel machine turned into water by using up oxygen as water solubility behavior of a hit in water. Automatic control system model for water quality management in shrimp pond.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดีเนื่องจากได้รับความช่วยเหลือและความร่วมมือจากหลายๆฝ่าย บุคคลที่เป็นส่วนสำคัญในการผลักดันให้โครงการนี้สำเร็จลุล่วงได้ด้วยดี ก็คือ อาจารย์นรินทร์ ธรรมารักษ์วัฒนะ อาจารย์ที่ปรึกษาปริญญานิพนธ์ที่ให้ทั้งความใส่ใจ แนะนำ และช่วยเหลือเสมอมาต้องขอขอบพระคุณเป็นอย่างมาก

และต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ได้ คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดีพร้อมให้โอกาสทางการศึกษาที่ดี และสนับสนุนอย่างเต็มที่ และยังคอยให้กำลังใจ เอาใจใส่เสมอในทุกๆด้านอันที่เปรียบหามิได้ ข้าพเจ้าขอระลึกพระคุณอันสูงสุดและขอกราบพระคุณมา ณ ที่นี้ด้วย

คุณค่าและประโยชน์ที่พึงได้รับจากปริญญานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

คณะผู้จัดทำ

สารบัญ

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

สารบัญภาพ(ต่อ)

สารบัญภาพ(ต่อ)

สารบัญโปรแกรม

สารบัญโปรแกรม(ต่อ)

บทที่ 1
บทนำ

1.1 ความสำคัญของปริญญานิพนธ์

ปัจจุบันประเทศไทยเป็นผู้นำในอุตสาหกรรมการเพาะเลี้ยงกุ้งทะเลของโลก โดยมีการพัฒนากระบวนการเพาะเลี้ยงให้มีคุณภาพตามมาตรฐานสากลอย่างต่อเนื่องตลอดมา ณ ตอนนี้ไทยได้ก้าวขึ้นสู่การเป็นผู้นำทั้งในด้านการเพาะเลี้ยงและการส่งออกกุ้งของโลก แต่อย่างไรก็ตาม เป็นที่ทราบกันดีว่าในกระบวนการเพาะเลี้ยงกุ้งทะเลนั้น ความจำเป็นในการใช้พลังงานทั้งไฟฟ้าหรือน้ำมันเชื้อเพลิงเป็นสิ่งที่หลีกเลี่ยงไม่ได้ เพราะเป็นปัจจัยสำคัญในการควบคุมคุณภาพน้ำให้มีความเหมาะสมต่อการเจริญเติบโต ของกุ้ง ซึ่งสวนทางกับราคาพลังงานซึ่งเป็นต้นทุนการเพาะเลี้ยงที่ปรับตัวสูงขึ้นอย่างต่อเนื่องจึงส่งผลให้เกษตรกรต้องแบกรับภาระค่าใช้จ่ายที่สูงขึ้นตามไปด้วย

ซึ่งโครงการนี้จัดทำขึ้นเพื่อทำการวิจัยและพัฒนาการควบคุมออกซิเจนในน้ำสำหรับบ่อเลี้ยงกุ้ง เพื่อลดปริมาณการใช้พลังงานไฟฟ้าหรือการน้ำมันเชื้อเพลิงลง ซึ่งจะช่วยให้ต้นทุนในการเพาะเลี้ยงกุ้งลดลงเช่นกัน การที่จะเพิ่มประสิทธิภาพการใช้พลังงาน ในการเพาะเลี้ยงกุ้งเพื่อให้กระบวนการผลิตสามารถประหยัดพลังงานได้สูงสุด โดยใช้ระบบควบคุมการปั่นมอเตอร์อัตโนมัติ และเครื่องเซนเซอร์มาตรวจวัดปริมาณออกซิเจนละลายน้ำ อุปกรณ์เซนเซอร์จะทำหน้าที่วัดและส่งข้อมูลไปคอมพิวเตอร์เพื่อให้แสดงค่าปริมาณที่ออกซิเจนละลายน้ำและบันทึกผล เมื่อใดที่ค่าออกซิเจนละลายน้ำต่ำกว่าค่ามาตรฐานที่เหมาะสมกับการเลี้ยงกุ้ง ระบบจะทำการเปิดมอเตอร์แบบอัตโนมัติเพื่อปั่นน้ำเพื่อให้ค่าของออกซิเจนละลายน้ำอยู่ในเกณฑ์ที่เหมาะสม

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1.2.1 เพื่อศึกษาหลักการทำงานของเซนเซอร์วัดออกซิเจนละลายน้ำ
1.2.2 เพื่อศึกษาการเขียนโปรแกรมคำสั่งของบอร์ด Arduino
1.2.3 เพื่อศึกษาการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอก
1.2.4 เพื่อศึกษาการเขียนโปรแกรม Visual studio 2012
1.2.5 เพื่อฝึกทักษะการติดต่อ การทำงานร่วมกับผู้อื่น

1.3 ขอบเขตของปริญญานิพนธ์

1.3.1 สามารถเขียนโปรแกรมของ Arduino เพื่อติดต่อกับคอมพิวเตอร์ได้
1.3.2 สามารถเขียนโปรมแกรมของ Visual studio 2012
1.3.3 สามารถต่อวงจรไฟฟ้ากับอุปกรณ์ เซนเซอร์ ไอซี และ บอร์ดคอลโทรลได้
1.3.4 สามารถใช้เซนเซอร์วัดออกซิเจนละลายน้ำและทราบถึงปัจจัยของการเปลี่ยนแปลงออกซิเจนในน้ำ
1.3.5 สามารถควบคุมการ เปิด-ปิด มอเตอร์ให้เป็นแบบอัตโนมัติได้

1.4 ขั้นตอนการศึกษา

1.4.1 ศึกษาการเขียนโปรแกรม Arduino

1.4.2 ศึกษาการทำงานของเซนเซอร์วัดออกซิเจนละลายน้ำ

1.4.3 ศึกษาการเขียนโปรแกรม Visual studio 2012

1.4.4 ศึกษาการต่อวงจรไฟฟ้ากับอุปกรณ์ เซนเซอร์ ไอซี และ บอร์ดคอลโทรล

1.4.5 ศึกษาปัจจัยที่มีผลต่อออกซิเจนละลายน้ำ

บทที่ 2
ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 รู้จักกับ Arduino

2.1.1 Arduino

Arduino เป็นภาษาอิตาลี ซึ่งใช้เป็นชื่อโครงการการพัฒนาไมโครคอนโทรเลอร์ตระกูล AVR แบบ Open Source ที่ได้รับการปรับปรุงมาจากโครงการพัฒนา Open source ของ AVR อีกโครงการหนึ่งที่ชื่อว่า " Wiring " แต่เหนื่องจากโครงการของ Wiring เลือกใช้AVR เบอร์ ATmega128 ซึ่งเป็นไมโครคอนโทรเลอร์ที่มีจำนวนของหน่วยความจำ และ I/O ค่อนข้างมาก และที่สำคัญ ATmega 128 เป็นชิพแบบตัวถังแบบ SMD จึงทำให้เป็นอุปสรรคสำหรับผู้เริ่มต้นในการสร้างบอร์ดหรือต่อวงจรขึ้นมาใช้งานกันเอง และบอร์มีขนาดใหญ่ ซึ่งอาจดูว่าเกินความจำเป็นสำหรับผู้เริ่มต้น จึงไม่ค่อยได้รับความนิยมเท่าที่ควร แต่หลังจากที่ทางทีมงาน arduino นำ Source code ของ "Wiring" มาพัฒนาปรับปรุงใหม่ให้สามารถใช้งานกับไมโครคอนโทรลเลอร์AVR ขนาดเล็กอย่างMega8 และ Mega 168ได้ จึงทำให้ระบบวงจรของบอร์ดมีขนาดเล็กกว่า "Wiring" มากและยังใช้อุปกรณ์น้อยชิ้น ทำให้ง่ายต่อการต่อวงจรใช้งานกันเองและยังประหยัดต้นทุนในการสร้างบอร์ดได้มาก ด้วยเหตุนี้เองที่ทำให้ "Arduino" ได้รับความนิยมจากผู้ใช้งานทั่วโลกเป็นอย่างมาก ในระยะเวลาอันรวดเร็ว

เนื่องจาก Arduino เป็นภาษาอิตาลี ซึ่งมีสำเนียงการอ่านออกเสียงที่เป็นรูปแบบเฉพาะ และไม่มีการกำหนดเป็นภาษาไทยอย่างเป็นทางการ ถึงแม้ว่า Arduino เป็นที่รู้จักของคนไทยมาในระยะเวลาหนึ่งแล้วก็ตามที แต่ก็ยังไม่มีคำอ่านที่เป็นภาษาไทยอย่างเป็นทางการ อย่างเป็นทางการนี้ควรออกเสียงเป็นไทยว่าอย่างไร บางคนอ่านออกเสียง อาร์-ดู-วี-โน่ บางคนอ่านว่า อา-เดีย-โน่ บางคนอ่านว่า เอ-อา-ดู-ไอ-โน และอื่นๆอีกมากมาย ซึ่งผู้เขียนไม่รู้จักเรื่องภาษาอิตาลีเลยจึงไม่ทราบว่า จะเรียกชื่อของ Arduino เป็นภาษาไทยว่าอย่างไรจึงจะถูกต้อง ดังนั้นเพื่อไม่ให้เกิดความสับสน จึงขอใช้การทับศัพท์ตามชื่อเรียกที่เป็นภาษาอังกฤษเป็น Arduino ไปเลย

Arduino มีจุดเด่นในเรื่องของความง่ายในการเรียนรู้และใช้งานเนื่องจากมีการออกแบบคำสั่งต่างๆขึ้นมาสนับสนุนในการใช้งาน ด้วยรูปแบบที่ง่ายไม่ซับซ้อน ซึ่งถึงแม้ว่า Arduino เองจะมีรูปแบบการใช้งานคล้ายๆกับไมโครคอนโทรลเลอร์อย่าง Basic Stamp ของ Parallax ,BX-24 ของNetmedias และ Handy Board ของ MIT แต่ก็มีจุดเด่นกว่าของรายการอื่นๆหลากหลายอย่างเช่น

- ราคาไม่แพง เนื่องจากมี Source Code และวงจร แจกให้ฟรีสามารถต่อวงจรขึ้นเองได้
- โปรแกรมที่ใช้พัฒนาจอง Arduino รองรับการทำงานทั้งWindows,Linuxและ Macintosh OSX
- มีรูปแบบคำสั่งที่ง่ายต่อการใช้งาน แต่สามารถนำไปใช้งานจริงๆ ที่มีความซับซ้อนมากๆ ได้และยังสามารถสร้างคำสั่งและLibrary ใหม่ๆขึ้นมาใช้งานเองได้เมื่อมีความชำนาญมากขึ้นแล้ว
- มีการเปิดเผยวงจรและSource Code ทั้งหมดทำให้สามรถนำไปพัฒนาต่อยอดเพิ่มเติมได้ตามความต้องการทั้ง Hardware และ software

Arduino เป็นบอร์ดไมโครคอนโทรลเลอร์โดยใช้แ AVR ขนาดเล็กเป็นตัวปรัมวลผลและสั่งงาน เหมาะสมสำหรับนำไปใช้ในการศึกษาเรียนรู้ในระบบไมโครคอนโทรลเลอร์ และนำไปประยุกต์ใช้งานเกี่ยวกับการควบคุมอุปกรณ์ Input/output ต่างๆได้มากมาย ทั้งในแบบที่เป็นการทำงานตัวเดียวอิสระหรือการเชื่อมต่อสั่งงานร่วมกับอุปกรณ์อื่นๆ เช่น คอมพิวเตอร์ PC ทั้งนี้เนื่องมาจากว่า Arduino สนับสนุนการเชื่อมต่อกับอุปกรณ์ Input/output ต่างๆได้มากมายทั้งแบบ Digital และ Analog เช่น การรับค่าจากสวิตช์ หรือ อุปกรณ์ตรวจจับ (Sensor) แบบต่างๆรวมไปถึงการควบคุมอุปกรณ์ Output ต่างๆตั้งแต่ LED,หลอดไฟ,มอเตอร์,รีเลย์ โดยระบบฮาร์ดแวร์ของ Arduino สามารถสร้างและประกอบขึ้นใช้งานได้เอง ในกรณีที่ผู้ใช้พอมีความรู้ด้านอิเลคทรอนิกส์อยู่บ้างหรือสามารถซื้อแผนวงจรสำเร็จรูปที่มีการผลิตออกจำหน่ายกันในราคาไม่แพง สำหรับเรื่องของโปรแกรมทีค่จะใช้เป็นเป็นเครื่องมือในการพัฒนานั้นสามารถ Download มาใช้งานกันได้ฟรีโดยไม่ต้องเสียค่าใช้จ่ายใดๆโดย Arduino มีความง่ายต่อการพัฒนาต่อการพัฒนาโปรแกรมและมีรเอกสารข้อมูลรวมทั้งตัวอย่างต่างๆให้ใช้เป็นแนวทางในการในการศึกษาเรียนรู้เป็นจำนวนมาก เนื่องจาก Arduino เป็นระบบการพัฒนาไมโครคอนโทรลเลอร์แบบ Open source ซึ่งมีการตีพิมพ์เอกสารต่างๆที่เกี่ยวข้องออกมาเผยแพร่ให้ได้รับรู้เป็นระยะๆ รวมทั้งการเปิดเผย Source code และตัวอย่างๆ ให้ผู้ใช้งานนำไปใช้งานหรือพัฒนาดัดแปลงต่อยอดได้โดยไม่เสียค่าใช้จ่ายด้วยเหตุนี้จึงมีคนทั่วไปให้ความสนใจและนำไปศึกษาทดลองใช้งานกันมากมาย มีการนำไปดัดแปลงและสร้างเป็นโครงงาน แบบต่างๆ กันเป็นจำนวนมาก จึงเป็นประโยชน์อย่างยิ่งสำหรับผู้ที่เริ่มต้นที่สามารถนำเอาตัวอย่างและโครงงานต่างๆที่คนอื่นทำไว้แล้วมาใช้อ้างอิงเป็นแนวทางการศึกษาเรียนรู้ได้โดยง่ายและที่สำคัญคือ ฟรี ไม่เสียค่าใช้จ่าย

2.1.2 เปรียบเทียบ ภาษาซี กับ Arduino

โปรแกรมภาษาของ Arduino จะใช้ภาษา C++ ซึ่งเป็นรูปแบบของโปรแกรมภาษาซีประยุกต์ แบบหนึ่ง ที่มีโครงสร้างของตัวภาษาโดยรวมใกล้เคียงกันกับ ภาษาซีมาตรฐาน (ANSI-C) อื่นๆ เพียงแต่ได้มีการปรับปรุงรูปแบบในการเขียนโปรแกรมบางส่วนที่ผิดเพี้ยนไปจาก ANSI-C เล็กน้อย เพื่อช่วยลดความยุ่งยากในการเขียนโปรแกรมและให้ผู้เขียนโปรแกรมสามารถเขียนโปรแกรมได้ง่ายและสะดวกมากขึ้นกว่าการเขียนภาษาซีตามแบบมาตรฐานของ ANSI-C โดยตรง

เมื่อกล่าวถึงภาษาซีนั้น คิดว่าหลายคนคงเคยได้ยินชื่อของภาษาซีกันมาบ้างแล้ว และบางคนอาจเคยเขียนโปรแกรมด้วยภาษาซีกันมาแล้ว บางคนอาจกำลังศึกษาและใช้งานกันอยู่ สำหรับภาษาซีที่มีการใช้งานกันอยู่ในแวดวงของ ไมโครคอนโทรลเลอร์ ปัจจุบันนั้นจะมีอยู่มากมากหลายยี่ห้อทั้งแบบที่แจกจ่ายให้ใช้งานกันฟรีๆ และแบบเสียเงินซื้อด้วยระดับราคาต่างๆกันไป อาจกล่าวได้ว่ามันมีมากจนนับกันไม่ถ้วนเลยทีเดียว มักมีคำถามประจำว่า ถ้าคิดจะเริ่มต้นเขียนภาษาซี จะเลือกใช้งานภาษาซีตัวไหนดี ภาษาซีแต่ละตัวมันแตกต่างกันอย่างไร คำถามเหล่านี้ มักเกิดขึ้นเสมอๆกับผู้ที่กำลังคิดจะเริ่มต้น หรือแม้แต่คนที่เคยเรียนโปรแกรมภาษาซีมาบ้างแล้วแต่อยู่บนโลกของคอมพิวเตอร์ เมื่อคิดจะเปลี่ยนหรือย้ายมาอยู่ในโลกของไมโครคอนโทรลเลอร์บ้าง กลับเริ่มต้นไม่ถูก และไม่สามารถเขียนโปรแกรมใช้งานได้ทันที ทั้งๆที่การเขียนโปรแกรมก็ใช้ภาษา ที่เรียกว่า ภาษาซีเหมือนๆกัน ดังนั้นในที่นี้ผู้เขียนจึงจะขอกล่าวนำให้รู้จัก ภาษาซี กันก่อนซักเล็กน้อย ก่อนที่จะเริ่มต้นเข้าสู่เรื่องราวที่เป็นภาษาซีของ Arduino กันโดยตรง

โดยประวัติความเป็นมาของภาษาซีนั้น ถูกบันทึกไว้ว่า ภาษาซี (C Programming Language)นั้นถูกพัฒนาขึ้นมาในปี ค.ศ. 1970 โดย Dennis Ritchie แห่ง Bell Laboratories ซึ่งในช่วงแรกๆ ภาษาซีถูกใช้งานเฉพาะแต่ในห้องปฏิบัติการของ Bell จนกระทั่งปี 1978 Brian Kerninghan กับ Dennis Ritchie จึงได้ออกหนังสือ กำหนดมาตรฐานของภาษาซี ออกมาเผยแพร่ ข้อกำหนดนี้ ถูกเรียกว่า K&R C ทำให้ภาษาซีเริ่มเป็นที่รู้จักของคนทั่วไปกันมากขึ้น จนกระทั่งปี 1980 ภาษาซี ก็ได้รับความนิยมมากขึ้นเป็นลำดับ จนได้มีการพัฒนาโปรแกรมที่ใช้ในการแปลภาษาซี (C-Compiler) ออกมาใช้งานกันอย่างแพร่หลาย ซึ่งสิ่งที่ทำให้ภาษาซีได้รับความนิยมในเวลาอันรวดเร็วก็เนื่องจากว่า ภาษาซีมีความได้เปรียบและมีความอ่อนตัวในการใช้งานที่เหนือกว่าภาษาอื่นๆ คือ ภาษาซี สามารถนำไปใช้งานบนระบบฮาร์ดแวร์ที่มีความแตกต่างกันได้หลากหลาย โดยผู้ใช้เพียงแต่เลือกใช้ตัวแปลคำสั่ง (C-Compiler) ให้ตรงกับระบบฮาร์ดแวร์ที่ใช้งานอยู่ ส่วนเรื่องรูปแบบในการเขียนโปรแกรมจะเป็นมาตรฐานอันเดียวกัน จนในบางครั้ง อาจสามารถย้ายโปรแกรมจากระบบฮาร์ดแวร์หนึ่งไปใช้งานกับอีกระบบหนึ่งได้ ทำให้ไม่ต้องเสียเวลาในการศึกษาโปรแกรมใหม่ๆให้เสียเวลา เนื่องจากว่าภาษาซี เป็นภาษาที่มีรูปแบบการใช้งานที่ง่าย คือ มีแต่ข้อกำหนดในการใช้งานหรือSyntax แต่ไม่มีฟังก์ชันสำเร็จรูป (Built-in Function) ใดๆรวมอยู่ในตัวของภาษาด้วย โดยส่วนที่เป็นฟังก์ชั่นการใช้งานต่างๆ เช่น การดำเนินการเกี่ยวกับ Input / Output การจองหน่วยความจำ (Memory Allocation) เป็นหน้าที่ของผู้ใช้ที่จะต้องสร้างขึ้นใช้งานเอง หรือ ในบางครั้งก็อาจใช้วิธีการเรียกใช้ฟังก์ชั่นที่ผู้ผลิตตัวแปลคำสั่ง (C-Compiler) สร้างเตรียมไว้ให้ใช้งานในรูปแบบของ Library Function **ซึ่งคำสั่งหรือฟังก์ชั่นส่วนที่เป็น Library Function นี้เอง ที่เป็นสาเหตุทำให้ภาษาซี มีความแตกต่างกัน** เพียงแต่ว่าในบางครั้งผู้ใช้จะแบ่งแยกไม่ออกว่า อันไหนเป็นคำสั่งส่วนมาตรฐานของภาษาซี อันไหนเป็นส่วนที่สร้างขึ้นมาใหม่ด้วยเหตุที่ภาษาซี เป็นที่นิยมใช้งานกันอย่างแพร่หลายมากขึ้น จึงทำให้มีผู้ผลิต Compiler ของภาษาซีเกิดขึ้นมามากมาย ทั้งแบบที่แจกจ่ายให้ใช้งานกันฟรีๆ และแบบที่จำหน่ายเพื่อการค้า สาเหตุของการแข่งขันกันนี้เองที่ทำให้เริ่มมีการเพิ่มเติมความสามารถและลูกเล่นต่างๆให้กับภาษาซีมากขึ้น เพื่อหวังดึงดูดใจผู้ซื้อและเป็นจุดขาย ทำให้ภาษาซีเริ่มมีความแตกต่างกันในแต่ละผู้ผลิต จนทำให้เหล่าผู้ใช้เกิดความสับสนกันเป็นอย่างมาก ดังนั้นทาง **American National Standard Institute (ANSI)** จึงได้ทำการตั้งข้อกำหนดมาตรฐานของภาษาซีขึ้น โดยเรียกกันว่า **"ANSI-C"** เพื่อใช้เป็นข้อบังคับและคงมาตรฐานของภาษาซีไว้ไม่ให้เปลี่ยนแปลงไปจากเดิม โดยทุกผู้ผลิตจะต้องสร้างตัวแปลภาษาให้มีคุณสมบัติการใช้งานขั้นพื้นฐานที่เป็นสิ่งเดียวกัน ตามข้อกำหนดของ ANSI สำหรับส่วนของคำสั่งที่จะมีการสร้างเพิ่มเติมขึ้นมาใช้งานใหม่นั้นให้เป็นสิทธิของผู้ผลิตแต่ละรายที่จะสร้างขึ้นมาใหม่ แต่จะไม่ถือเอาส่วนคำสั่งที่มีการสร้างขึ้นมาเพิ่มเติมนั้นว่าเป็นส่วนของ ANSI-C ด้วย ดังนั้นเมื่อจะใช้งานภาษาซีตัวใด ก็ขอให้ผู้อ่านศึกษารายละเอียดทางด้านคุณสมบัติของตัวภาษานั้นๆด้วย ว่ามีคุณสมบัติการใช้งานที่รองรับคำสั่งของ ANSI-C หรือไม่ มีข้อยกเว้นอย่างไรบ้าง และมีการสร้างคำสั่งหรือเพิ่มเติมความสามารถมากกว่า ANSI-C อย่างไรบ้าง ซึ่งถ้าโปรแกรมที่เขียนขึ้นนั้น เลือกใช้เฉพาะคำสั่งที่เป็น ANSI-C ทั้งหมด จะทำให้สามารถย้าย Code โปรแกรม จากระบบฮาร์ดแวร์แบบหนึ่ง เพื่อไปใช้งานกับอีกระบบฮาร์ดแวร์แบบหนึ่งได้ทันที แต่ถ้าใน Code โปรแกรมนั้นมีการใช้คำสั่งที่นอกเหนือไปจากคำสั่งของ ANSI-C แล้ว ก็จะไม่สามารถใช้ได้ทันที ต้องมีการปรับแก้ Code กันใหม่เป็นบางส่วน หรือต้องสร้างคำสั่งบางส่วนขึ้นมาทดแทน จึงจะสามารถใช้งานได้ ซึ่งส่วนของคำสั่งที่เป็นภาษาซีตามมาตรฐานของ ANSI-C จะมี 32 คำสั่ง คือ

auto	break	case	char	const
continue	default	do	double	else
enum	extern	float	for	goto
if	int	long	register	return
short	signed	sizeof	static	struct
switch	typedef	union	unsigned	void
volatile	while			

คำสั่งทั้ง 32 คำสั่งนี้ ไม่ว่าจะอยู่บน C-Compiler ตัวใดก็จะต้องมีข้อกำหนดและรูปแบบการใช้งานของคำสั่งที่เหมือนกันทั้งหมด ยกเว้นว่า C-Compiler ตัวนั้น ไม่รองรับคำสั่งของ ANSI-C ทั้งหมด ซึ่งส่วนนี้ต้องดูจากคุณสมบัติของภาษาซีทีจะใช้ว่าเป็นอย่างไรมาถึงตรงนี้ก็คงพอจะทำให้ผู้อ่านได้รู้จักภาษาซีกันดีขึ้นบ้างพอสมควรแล้ว ดังนั้น ในการศึกษาภาษาซีนั้น ขอให้ศึกษาโครงสร้าง ข้อกำหนด และรูปแบบการใช้งาน ของคำสั่งที่เป็น ANSI-C ทั้ง 32 คำสั่งนี้ไว้ก่อนเป็นอันดับแรก ซึ่งจะเป็นพื้นฐานและเป็นประโยชน์อย่างมาก และยังสามารถที่จะนำความรู้ที่ได้ไปใช้งานได้กับ C-Compiler ทุกยี่ห้อ ที่รองรับมาตรฐานของ ANSI-C ได้ทั้งหมด แล้วค่อยไปศึกษาเพิ่มเติมในเรื่องของลูกเล่น คำสั่งเพิ่มเติม และ ความความสามารถอื่นๆในส่วนที่แตกต่างกันของ C-Compilerสำหรับการเขียนโปรแกรมของ Arduino นั้นจะใช้ภาษา C++ ซึ่งเป็นรูปแบบภาษาซีประยุกต์แบบหนึ่ง ที่มีโครงสร้างการทำงานของตัวภาษาโดยรวม คล้ายกับ ภาษาซีมาตรฐาน (ANSI-C) ทั่วๆไป เพียงแต่ได้มีการปรับปรุงเพื่อลดความยุ่งยากในการใช้งานลง เพื่อให้ผู้ใช้สามารถใช้งานและเขียนโปรแกรมได้ง่ายและสะดวกมากกว่าเขียนภาษาซีแบบมาตรฐานโดยตรง ซึ่งในความเป็นจริงแล้วในการเขียนโปรแกรมของArduino เราสามารถใช้คำสั่งต่างๆที่เป็นคำสั่งตามมาตรฐานของ ANSI-C เข้ามาใช้ในการเขียนโปรแกรมได้ทันที โดยรูปแบบการเขียนโปรแกรม และ การใช้งานคำสั่งต่างๆนั้น สามารถอ้างอิงจากหนังสือ ตำรา ของภาษาซี มาตรฐาน ANSI-C ได้โดยตรง

ซึ่งหนังสือเล่มนี้จะเขียนขึ้นโดยอ้างอิงข้อมูลบางส่วนมาจากข้อกำหนดต่างๆของภาษาซีตามข้อกำหนดของ ANSI-C ผสมรวมเข้ากับข้อกำหนดเฉพาะและรายละเอียดเพิ่มเติมในส่วนที่เป็น C++ ของArduino เนื่องจาก Arduino ได้ทำการปรับปรุงและดัดแปลงรูปแบบการเขียนโปรแกรมที่ผิดเพี้ยนไปจากภาษาซีมาตรฐาน ซึ่งในบางส่วนก็เป็นส่วนที่ถูกออกแบบและสร้างขึ้นมาใหม่ โดยเนื้อหาบางส่วนก็ได้ ตัดต่อเรียบเรียงและแปลมาจากเค้าโครงของคู่มืออ้างอิงของ Arduino ที่เป็นภาษาอังกฤษ โดยได้มีการสอดแทรกเนื้อหาอื่นๆและตัวอย่างโปรแกรมต่างๆเพิ่มเติมเข้าไปตามความเหมาะสม

ซึ่งจากการที่ผู้เขียนได้ทำการศึกษาค้นคว้าทดลองและใช้งานภาษาซี ของ Arduino มาในระยะเวลาหนึ่ง พบว่าในความเป็นจริงแล้ว Arduino นั้นไม่ใช่ C-Compiler โดยตรง แต่ Arduino จะมีลักษณะการทำงานเช่นเดียวกันกับ Text Editor ของภาษา C++ ตัวหนึ่ง โดยจะทำงานร่วมกับ Utilityบางส่วนที่ Arduino สร้างขึ้นมารองรับ โดย Arduino จะใช้รูปแบบการทำงานของ Text Editor เป็นฉากหน้าในการติดต่อสื่อสารกับผู้ใช้เท่านั้น ส่วนเบื้องหลังจริงๆนั้น Arduino จะไปเรียกใช้ตัวแปลภาษาซีและUtilityอื่นๆที่ใช้เป็นเครื่องมือพัฒนาโปรแกรมของ ไมโครคอนโทรลเลอร์ ตระกูล AVR อีกทีหนึ่ง โดย Arduino จะเลือกใช้ C-Compiler ของ “GNU AVR-GCC Toolchain” ร่วมกับ Library Function ของ “avr-libc” ส่วนUtility ที่ใช้ในการ Upload Code ให้กับ AVR นั้นก็จะใช้

ของ “AVRDude” ดังนั้นผู้ที่เขียนภาษาซีของ AVRเป็นอยู่แล้ว และต้องการประยุกต์ใช้งาน Arduino ให้ได้ประสิทธิภาพการทำงานมากยิ่งขึ้นไปอีก ก็สามารถศึกษาข้อกำหนด และ หน้าที่ในการใช้งาน Library และคำสั่งอื่นๆที่บรรจุไว้ใน Library ต่างๆทั้งจากของ“GNU AVR-GCC Toolchain” และ “avr-libc” เพิ่มเติมอีก เพื่อใช้เป็นแนวทางในการปรับปรุงและประยุกต์ใช้งาน Arduino ในรูปแบบที่สลับซับซ้อนมากๆขึ้นไปได้อีก

2.1.3 โครงสร้างการเขียนโปรแกรม ภาษาซี ของ Arduino

ภาษาซีของ Arduino จะจัดแบ่งรูปแบบโครงสร้างของการเขียนโปรแกรมออกเป็นส่วนย่อยๆหลายๆส่วน โดยเรียกแต่ละส่วนว่า ฟังก์ชั่น และ เมื่อนำฟังก์ชั่น มารวมเข้าด้วยกัน ก็จะเรียกว่าโปรแกรมโดยโครงสร้างการเขียนโปรแกรมของ Arduino นั้น ทุกๆโปรแกรมจะต้องประกอบไปด้วยฟังก์ชั่นจำนวนเท่าใดก็ได้ แต่อย่างน้อยที่สุดต้องมีฟังก์ชั่น จำนวน 2 ฟังก์ชั่น คือ setup() และ loop() ดังตัวอย่าง

โปรแกรมที่ 2.1 รูปแบบโครงสร้างของการเขียนโปรแกรม

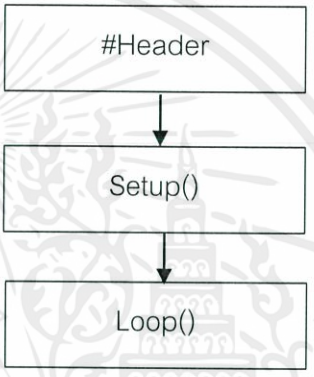
```
#include <Servo.h>          // สั่งผนวกไฟล์ ชื่อ Servo.h เข้ามาใช้ในโปรแกรม

int Servo1 = 9;             // กำหนดให้ Servo1 แทน Pin Digital-9

Servo myservo;              // สร้าง object ชื่อ myservo เพื่อควบคุม Servo

void setup()

{

myservo.attach(Servo1);     // กำหนดให้ใช้ขา Digital-9 สร้างสัญญาณควบคุม Servo

}
```

- Header ได้แก่ ส่วนที่เป็น Compiler Directive ต่างๆ รวมไปถึงส่วนของการประกาศตัวแปร และค่าคงที่ต่างๆที่จะใช้ในโปรแกรม
- setup() ในส่วนนี้เปน็ ฟังกช์ั่นบังคับที่ต้องกำหนดใหมี้ในทุกๆโปรแกรม ถึงแมว้่า ในบางโปรแกรมจะไม่ต้องการใช้งานก็ยังจำเป็นต้องประกาศไว้ด้วยเสมอ เพียงแต่ไม่ต้องเขียนคำสั่งใดๆไว้ในระหว่าง
- วงเล็บปีกกา { } ที่ใช้เป็นตัวกำหนดขอบเขตของฟังก์ชั่น โดยฟังก์ชั่นนี้จะใช้สำหรับบรรจุคำสั่งในส่วนที่ต้องการให้โปรแกรมทำงานเพียงรอบเดียวตอนเริ่มต้นทำงานของโปรแกรมครั้งแรกเท่านั้น ซึ่ง
- ได้แก่คำสั่งเกี่ยวกับการ Setup ค่าการทำงานต่างๆ เช่น การกำหนดหน้าที่การใช้งานของ

- PinMode และการกำหนดค่า Baudrate สำหรับใช้งานพอร์ตสื่อสารอนุกรม เป็นต้น
- **loop()** เป็นส่วนฟังก์ชั่นบังคับที่ต้องกำหนดให้มีในทุกๆโปรแกรมเช่นเดียวกันกับฟังก์ชั่น setup( )
  โดยฟังก์ชั่น loop( ) นี้จะใช้บรรจุคำสั่งที่ต้องการให้โปรแกรมทำงานเป็นวงรอบซ้ำๆกันไปไม่รู้จบ ซึ่ง
  ถ้าเปรียบเทียบกับรูปแบบของ ANSI-C ส่วนนี้ก็คือ ฟังก์ชั่น main( ) นั่นเอง

โปรแกรมที่ 2.2 แสดงโครงสร้างโปรแกรมของ Arduino

```
#include <Servo.h>

int Servo1 = 9;

Servo myservo;

Setup()

{

myservo.attach(9);

}
```



จะเห็นได้ส่วนแรกซึ่งถือเป็นส่วนเริ่มต้นของโปรแกรม ซึ่งจะเรียกว่า Header โดยประกอบด้วยคำสั่ง#include ซึ่งเป็นคำสั่งพิเศษที่เรียกว่า Compiler Directive ซึ่งมัน ไม่ใช่คำสั่งสำหรับสั่งงานในโปรแกรมดังนั้นคำสั่งนี้จึงไม่ต้องมีเครื่องหมายเซมิโคลอนปิดท้ายคำสั่งเหมือนคำสั่งอื่นๆ โดย Compiler

Directiveจะใช้ทำหน้าที่สำหรับบอกให้ Compiler รับรู้เงื่อนไขในการแปลคำสั่งเท่านั้น ซึ่งในกรณีคำสั่ง #include จะใช้สำหรับบอกให้ Compiler รับรู้ว่าในการแปลคำสั่งของโปรแกรมนี้ มีไฟล์ภายนอกใดบ้างที่จำเป็นต้องใช้ร่วมในการแปลคำสั่งให้กับโปรแกรมนี้ โดย

จากตัวอย่าง ข้างต้นจะเป็นการบอกให้ Compiler ทำการผนวกไฟล์ ชื่อ “Servo.h” เข้ามาใช้เพื่อเรียกใช้คำสั่งต่างๆที่บรรจุไว้ เข้ามาใช้งานในโปรแกรม โดยใช้รูปแบบ

โปรแกรมที่ 2.3 ตำแหน่ง Directory ที่เก็บรวบรวม Library ของโปรแกรม Arduino

```
#include <header.h>
```


โดยเมื่อพบคำสั่ง #include ตัวแปลภาษาของ Arduino จะไปค้นหาไฟล์ที่ระบุไว้ในเครื่องหมาย<>หลังคำสั่ง #include จากตำแหน่ง Directory ที่เก็บรวบรวม Library ของ

โปรแกรม Arduino ไว้ ซึ่งก็คือ“..\arduino-0012\hardware\libraries\” เช่น เมื่อทำการติดตั้งโปรแกรมของ Arduino ไว้ที่ Directory ที่ชื่อว่า “c:\arduino-0012” ไฟล์ภายนอกที่เป็น Library Function และ Header ต่างๆ จะถูกรวบรวมเก็บไว้ที่“c:\arduino-0012\hardware\libraries\” เมื่อโปรแกรมพบคำสั่ง #include โปรแกรมของ Arduino จะไปค้นหาไฟล์ต่างๆจากตำแหน่งของ Directory ที่ชื่อ “c:\arduino-0012\hardware\libraries\” นั่นเอง

โดยส่วนของ Header จะนับรวมไปถึง คำสั่งส่วนที่ใช้ประกาศสร้าง ตัวแปร (Variable Declaration)และค่าคงที่ (Constant Declaration) รวมทั้ง ฟังก์ชั่นต่างๆ (Function Declaration) ด้วย ซึ่งจากตัวอย่างได้แก่ส่วนที่เป็นคำสั่ง

โปรแกรมที่ 2.4 ส่วนของ Header

```
int Servo1 = 9;

Servo myservo;
```

สำหรับส่วนที่มีความสำคัญและจำเป็นที่สุดของโปรแกรม Arduino ที่จำเป็นต้องมีและจะขาดไม่ได้ในการเขียนโปรแกรมของ Arduino คือ ฟังก์ชั่น setup() และ ฟังก์ชั่น loop() ซึ่งฟังก์ชั่น ทั้ง 2 ส่วนนี้มีรูปแบบโครงสร้างที่เหมือนกัน แต่ถูกกำหนดด้วยชื่อของฟังก์ชั่นเป็นการเฉพาะ คือ setup() และ loop() โดยsetup() จะเขียนไว้ก่อน loop() ซึ่งทั้ง 2 ฟังก์ชั่นนี้ มีขอบเขต เริ่มต้นและสิ้นสุด อยู่ภายใต้เครื่องหมาย { }

โปรแกรมที่ 2.5 ฟังก์ชั่น setup() ใน Arduino

```
void setup()

{

.คำสั่งต่างๆ ที่ต้องการเขียนไว้ภายใต้ฟังก์ชั่น setup()

.}
```

หน้าที่ของฟังก์ชั่น setup() ใน Arduino คือ ใช้ทำหน้าที่เป็นส่วนของโปรแกรมย่อยสำหรับใช้บรรจุคำสั่งต่างๆ ที่ใช้สำหรับกำหนดการทำงานของระบบ หรือ กำหนดคุณสมบัติการทำงานให้กับอุปกรณ์ต่างๆซึ่งคำสั่งทั้งหมดที่บรรจุไว้ภายใต้ฟังก์ชั่นของ setup() นี้ จะถูกเรียกขึ้นมาทำงานเพียงรอบเดียวคือตอนเริ่มต้นการทำงานของโปรแกรม (หลังการรีเซ็ตให้ MCU เริ่มต้นทำงาน) เท่านั้น โดยคำสั่งที่นิยมบรรจุไว้ในฟังก์ชั่นส่วนนี้ ได้แก่ คำสั่งสำหรับกำหนดโหมดการทำงานของ Digital Pin หรือ คำสั่งสำหรับ กำหนดคุณสมบัติของพอร์ตสื่อสารอนุกรม เป็นต้น

โปรแกรมที่ 2.6 ฟังก์ชั่น loop() ใน Arduino

```
void loop()

{.

คำสั่งต่างๆที่ต้องการให้ทำงานภายใต้ฟังก์ชั่น loop().

.}
```

หน้าที่ของฟังก์ชั่น loop() ใน Arduino คือใช้ทำหน้าที่เป็นส่วนของโปรแกรมหลัก สำหรับใช้บรรจุคำสั่งควบคุมการทำงานต่างๆของโปรแกรม ที่ต้องการใช้โปรแกรมทำงาน โดยคำสั่งที่บรรจุไว้ในฟังก์ชั่นนี้จะถูกเรียกขึ้นมาทำงานซ้ำๆกันตามลำดับและเงื่อนไขที่กำหนดไว้

ข้อสังเกตุ จะเห็นได้ว่าโปรแกรมนี้จะประกอบไปด้วย คำสั่งที่เขียนขึ้นเองส่วนหนึ่ง และอีกส่วน หนึ่งเป็น คำสั่งจากภายนอก ที่มีการสร้างและเก็บรวบรวมเป็นไฟล์ ในรูปแบบของ Library Function เก็บอยู่ภายนอกโปรแกรม เมื่อต้องการใช้งานก็สั่งผนวกไฟล์นั้นเข้ามาใช้งานในโปรแกรม จากนั้นก็สามารถเรียกใช้งานคำสั่งต่างๆที่สร้างเก็บไว้ในไฟล์นั้นได้ตามต้องการ

2.1.4 ตัวแปรใน Arduino

ตัวแปร หมายถึง กลุ่มของ ตัวอักษร ตัวเลข และ เครื่องหมายใดๆ ที่รวมกันเป็นชื่อเพื่อใช้กำหนดเป็นตัวแทนของค่าข้อมูลที่เราต้องการจะอ้างถึงในโปรแกรม ทั้งนี้ก็เนื่องจากว่าในการทำงานของโปรแกรมจริงๆนั้นจะใช้ค่าตัวเลขที่ผู้ใช้กำหนดให้ มาทำการประมวลผล ซึ่งในการเขียนโปรแกรมถ้าเราต้องเขียนโปรแกรมโดยกำหนดเป็นค่าตัวเลขให้กับโปรแกรมตรงๆเลย ก็จะทำให้โปรแกรมที่เราเขียนขึ้นเต็มไปด้วยค่าตัวเลขต่างๆมากมาย ซึ่งยากต่อการอ่าน ยากต่อการทำความเข้าใจ และยากต่อการตรวจสอบความถูกต้องและอาจทำให้เกิดความผิดพลาดได้ง่ายด้วย ดังนั้นทุกภาษา จึงยอมให้มีการกำหนดชื่อ ขึ้นมาใช้งานแทนค่าตัวเลข เพื่อให้เขียนโปรแกรมได้สะดวกและง่ายต่อการอ่าน ทำความเข้าใจ ได้มากยิ่งขึ้น

ซึ่งลักษณะของข้อมูล อาจมีทั้งแบบที่เป็นค่าซึ่งสามารถเปลี่ยนแปลงได้ (variable) หรือ อาจเป็นแบบที่มีค่าคงที่ไม่สามารถเปลี่ยนแปลงได้ (constant) ในการประกาศใช้งานตัวแปรจำเป็นต้องประกาศชนิดของตัวแปร หรือบางครั้งอาจมีการกำหนดค่าเริ่มต้นให้กับตัวแปรด้วยก็ได้

การตั้งชื่อตัวแปรของ Arduino จะยึดหลักของ ANSI-C ทุกประการนั่นก็คือ

- ชื่อของตัวแปรต้องประกอบไปด้วยตัวอักษร ตัวเลข และ ยอมให้ใช้เครื่องหมายพิเศษอีก 2 ตัว คือเครื่องหมาย Under Line (_) และ Dollar Sign ($) มาใช้ในการตั้งชื่อได้ โดยชื่อต้องเรียงติดทั้งหมดห้ามมีการเว้นวรรค และ มีความยาวไม่เกิน 32 ตัวอักษร

• ชื่อของตัวแปรที่กำหนด จะใช้เฉพาะตัวอักษรทั้งหมด หรือ ตัวอักษรผสมกับตัวเลข หรือ อาจผสมกับเครื่องหมาย _ และ $ ด้วยก็ได้ แต่ต้องเริ่มต้นชื่อด้วยตัวอักษรเป็นลำดับแรกเสมอ ห้ามตั้งชื่อโดยเริ่มต้นด้วยตัวเลข หรือ เครื่องหมาย

• ชื่อของตัวแปร ต้องกำหนดด้วยตัวอักษรที่เป็นตัวอักษรพิมพ์เล็กเท่านั้น และในภาษาซีจะถือว่าตัวอักษรที่เป็นตัวอักษรพิมพ์ใหญ่ และ ตัวอักษรพิมพ์เล็ก มีความหมายต่างกัน เช่น “LED” และ“led” และ “Led” ภาษาซี จะถือว่าเป็นคนละชื่อกัน

• ในการตั้งชื่อตัวแปร ห้ามนำคำสงวน (Reserve Word) มาใช้ตั้งชื่อ ซึ่งคำสงวน ได้แก่ ชื่อคำสั่ง และชื่อ Internal Function ต่างๆที่สร้างไว้แล้วในตัว ภาษา

2.1.5 คำสงวนต่างๆของ Arduino

ในโปรแกรมภาษาซีของ Arduino นั้นจะมี ประโยค หรือ คำสั่ง ที่กำหนดขึ้นมาใช้งานไว้แล้วจำนวนหนึ่ง เมื่อผู้ใช้ต้องการจะประกาศ สร้างตัวแปร หรือ สร้างคำสั่ง หรือ ฟังก์ชั่นต่างๆ ขึ้นมาใช้งานเอง ก็ต้องไม่ตั้งชื่อให้ซ้ำกับชื่อของคำสั่งที่มีอยู่แล้วของ Arduino อีก ซึ่งคำสั่งที่ Arduino กำหนดขึ้นเพื่อใช้งานเป็นการเฉพาะไว้แล้วนั้นจะถือเป็น คำสงวน ที่ห้ามไม่ให้ผู้ใช้ทำการประกาศขึ้นมาใช้งานใหม่อีก เพราะจะไปซ้ำกับคำสั่ง เดิมที่มีอยู่แล้วนั่นเอง ซึ่งคำสงวนของ Arduino ได้แก่

คำสงวนที่เป็น Constant ของ Arduino

HIGH	LOW	INPUT	OUTPUT	SERIAL
DISPLAY	PIHALF_PI	TWO_PI	LSBFIRST	MSBFIRST
CHANGE	FALLING	RISING	false	true

คำสงวนที่เป็น Port Variables & Constants ของ Arduino

DDRB	PINB	PORTB	PB0	PB1	PB2
PB3	PB4	PB5	PB6	PB7	DDRC
PINC	PORTC	PC0	PC1	PC2	PC3
PC4	PC5	PC6	PC7	DDRD	PIND
PORTD	PD0	PD1	PD2	PD3	PD4
PD5	PD6	PD7			

คำสงวนที่เป็น Data types ของ Arduino

Boolean	byte	char	class	default
double	Int	long	private	protected
public	return	short	signed	static
throw	try	unsigned	void	switch

คำสงวนที่เป็นคำสั่ง ของ Arduino

Abs	acos	+=	+	[]	asin
=	atan	atan2	&	|	boolean
Byte	case	ceil	char	class	,
//	?:	constrain	cos	{}	--
default	delay	delayMicroseconds		/	/**
.	else	==	exp	false	float
floor	for	<	<=	if	++
!=	int	<<	<	<=	log
&&	!	||	loop	max	millis
min	-	%	/*	*	
new	()	PI	return	>>	;
Serial	Setup	sin	sq	sqrt	-=
witch	tan	this	true	void	while begin
read	write	print	println	Available	
digitalWrite	digitalRead	analogRead	analogWrite	attachInterrupts	
detach	Interrupts	beginSerial	serialWrite	serialRead	
serialAvailable	printString	printInteger	printByte	printHex	
printOctal	printBinary	printNewline	pulseIn	shiftOut	pinMode

2.1.6 ชนิดและประเภทของตัวแปร

ถ้าหากว่าเราจะเปรียบเทียบว่า ตัวแปร คือ ภาชนะสำหรับบรรจุสิ่งของ และ ข้อมูล คือ สิ่งของที่เราต้องการจะเก็บ จะเห็นได้ว่า สิ่งของต่างๆรอบๆตัวเรานั้น จะมีคุณสมบัติที่แตกต่างกันไป ดังนั้นในการเลือกภาชนะสำหรับใช้บรรจุสิ่งของ เราก็จำเป็นต้องเลือกชนิดของภาชนะให้มีความเหมาะสมที่จะใช้เก็บสิ่งของด้วย ซึ่งสิ่งแรกที่ต้องพิจารณาคือ เราจะต้องรู้จักคุณสมบัติของสิ่งของที่ต้องการจะจัดเก็บ และ จุดประสงค์การใช้งาน ก่อน จากนั้นจึงจัดหาภาชนะที่มี ขนาด และ รูปทรงของภาชนะ เหมาะสมที่จะใช้เก็บสิ่งของเพื่อให้สามารถจัดเก็บ และ นำสิ่งของออกมาใช้งาน ได้อย่างง่ายประหยัดมากที่สุด

ในภาษาซีนั้น มีการกำหนด และ จำแนก ชนิดของตัวแปร ไว้เป็น 5 ชนิดด้วยกัน โดยแต่ละชนิดจะมีคุณสมบัติการใช้งานที่ต่างกัน เพื่อใช้ในการเก็บข้อมูลที่มีรูปแบบแตกต่างกัน คือ

- **char** ใช้เก็บข้อมูลที่เป็นตัวอักษร (character) ใช้เก็บข้อมูลที่เป็นเลขจำนวนเต็มได้ 256ค่า
- **int** ใช้เก็บข้อมูลที่เป็นเลขจำนวนเต็ม (integer) ใช้เก็บข้อมูลที่เป็นเลขจำนวนเต็มได้ 65536 ค่า
- **float** ใช้เก็บข้อมูลที่เป็นเลขทศนิยมแบบ Single Precision
- **double** ใช้เก็บข้อมูลที่เป็นเลขทศนิยมแบบ Double Precision ซึ่ง สามารถเก็บค่าตัวเลขทศนิยมที่มีความละเอียดและถูกต้องของทศนิยมมากว่าแบบ float ถึง 2 เท่า
- **void** ใช้เก็บตัวแปรที่ไม่มีค่า

ตารางที่ 2.1 ตารางแสดงคุณสมบัติของตัวแปรของภาษาซี

ชนิดตัวแปร	จำนวนบิต	ค่าข้อมูลที่เก็บได้
char	8	-128 ถึง +127
int	16	-32768 ถึง +32767
float	32	3.4E-38 ถึง 3.4E+38
double	64	1.7E-308 ถึง 1.7E+308
void	0	ไม่มีค่า

โดยในการประกาศสร้างตัวแปรขึ้นมาใช้งานของภาษาซีนั้นจะใช้รูปแบบดังนี้

โปรแกรมที่ 2.7 ตัวอย่างชนิดและประเภทของตัวแปร

```
type name
```

- type หมายถึง ชนิดของตัวแปร

2.1.7 คุณสมบัติเฉพาะของตัวแปร

สำหรับตัวแปรชนิดที่ใช้เก็บค่าเลขจำนวนเต็ม (char และ int) นั้น ในภาษาซี ไม่ได้มีการจำแนก ชนิดของตัวแปรเพื่อใช้เก็บค่าตัวเลขที่เป็น ค่าบวก หรือ ค่าลบ เป็นการเฉพาะ แต่ภาษาซีจะใช้วิธีการเพิ่ม คำสั่งสำหรับกำหนดคุณสมบัติเฉาพะให้กับตัวแปรไว้อีก 4 คำสั่ง สำหรับใช้กำหนดคุณสมบัติของตัวแปรแบบนี้ให้มีคุณสมบัติที่เฉพาะเจาะจงลงไปอีกเป็นต้นว่า จะใช้ตัวแปรในการเก็บค่าตัวเลขที่เป็นค่าบวกอย่างเดียวหรือต้องการเก็บค่าตัวเลขแบบคิดเครื่องหมายด้วย เพื่อให้ผู้ใช้สามารถปรับแก้คุณสมบัติในการใช้งานของตัวแปรให้มีคุณสมบัติใกล้เคียงกับความต้องการใช้งานมากขึ้นไปอีก และเพื่อจำกัดขอบเขตการใช้งานของตัวแปรให้ตรงกับจุดประสงค์มากยิ่งขึ้นและยังเป็นการช่วยให้ประหยัดจำนวนของหน่วยความจำที่ใช้สร้างตัวแปรด้วยและทำให้โปรแกรมทำงานได้เร็วขึ้นกว่าเดิมอีกด้วย โดยในภาษาซีจะมีคำสั่ง ที่ใช้สำหรับระบุคุณสมบัติเฉพาะของตัวแปรที่ใช้เก็บค่าเลขจำนวนเต็ม 4 คำสั่ง คือ

- unsigned ใช้ระบุให้เก็บค่าเลขจำนวนเต็มในตัวแปรเฉพาะค่าที่เป็นบวกเท่านั้น
- signed ใช้ระบุให้เก็บค่าเลขจำนวนเต็มในตัวแปรทั้งค่า บวก และ ลบ
- short ใช้ระบุให้เก็บค่าเลขจำนวนเต็มในตัวแปรที่มีค่าน้อยกว่า int
- long ใช้ระบุให้เก็บค่าเลขจำนวนเต็มในตัวแปรที่มีค่ามากกว่า int เป็น 2 เท่าโดยคำสั่งทั้ง 4 นี้จะใช้นำหน้าชนิดของตัวแปร เพื่อบอกให้ Compiler รับรู้ว่าตัวแปรที่เราประกาศสร้างขึ้นมาใช้งานนั้น มีคุณสมบัติเป็นอย่างไร เช่น

โปรแกรมที่ 2.8 ชนิดและประเภทของตัวแปร

```
char ch1;                    //กำหนดให้ ch1 เก็บค่า -128 ถึง +127
sign char ch2;               //กำหนดให้ ch2 เก็บค่า -128 ถึง +127
unsigned char ch3;           //กำหนดให้ ch3 เก็บค่า 0 ถึง 255
int num1;                    //กำหนดให้ num1 เก็บค่าได้ -32768 ถึง +32767
signed int num2;             //กำหนดให้ num2 เก็บค่าได้ -32768 ถึง +32767
```

2.1.8 การกำหนดชนิดตัวแปรขึ้นมาใช้งานเอง

นอกจากตัวแปรตามมาตรฐานของ ANSI-C ทั้ง 5 ชนิดที่กล่าวไปแล้วนั้น ภาษาซี ยังยอมให้ผู้ใช้สามารถกำหนดชนิดตัวแปร ขึ้นมาใช้งานได้เองด้วย โดยใช้คำสั่ง **typedef** โดยมีรูปแบบการใช้งานดังนี้

typedef type name

- **type** คือ ชื่อของชนิดตัวแปรที่ต้องการสร้างขึ้นใหม่
- **name** คือ ชื่อที่ใช้เรียกชนิดตัวแปรจริง(ตัวแปรมาตรฐาน)

โดยวิธีการนี้จะช่วยให้เราสามารถกำหนด ชื่อชนิดของตัวแปร แบบใหม่ๆขึ้นมาใช้งานได้เอง ในแบบที่เราต้องการ ซึ่งในกรณี C-Compiler ของ AVR และ Arduino เอง ก็มักใช้วิธีการนี้ เพื่อประกาศ

สร้างชื่อชนิดของตัวแปร แบบใหม่ขึ้นมาใช้งาน ซึ่งจุดประสงค์หลักก็เพื่อ ลดความยาวในการพิมพ์คำสั่ง ในการเขียนโปรแกรมให้สั้นลงเท่านั้นเอง ตัวอย่างของ ชื่อชนิดของตัวแปร แบบใหม่ๆที่พบเห็นในภาษาซีของ AVRและ Arduino ที่จะพบเห็นได้บ่อยๆ ได้แก่ int8_t, uint8_t, int16_t, uint16_t, int32_t, uint32t,....

โดยถ้าต้องการจะทราบว่า ชื่อชนิดของตัวแปร แบบใหม่ๆที่พบเห็นนี้ เมื่อเทียบกับชื่อของชนิดตัวแปรของ ภาษาซี ตามมาตรฐานของ ANSI-C แล้วมันคือ อะไร ในกรณีของ Arduino สามารถตรวจสอบได้จากไฟล์“stdint.h”ที่อยู่ใน c:\arduino0012\hardware\tools\avr\avr\include” ซึ่งเมื่อตรวจสอบดูในไฟล์ของ “stdint.h” ดูก็จะพบ บรรทัด คำสั่งที่ใช้ประกาศสร้าง ชื่อของชนิดตัวแปร ใหม่ๆขึ้นมาใช้งาน เช่น

ตารางที่ 2.2 ตารางสรุปคุณสมบัติของตัวแปรแบบต่างๆตามมาตรฐานของ ANSI-C

ชนิดตัวแปร	จำนวนบิต	ค่าข้อมูลที่เก็บได้
char	8	-128 ถึง +127
signed **char**	8	-128 ถึง +127
unsigned **char**	8	0 ถึง +255
int	16	-32768 ถึง +32767
signed **int**	16	-32768 ถึง +32767
unsigned **int**	16	0 ถึง 65535
short **int**	16	-32768 ถึง +32767
signed short **int**	16	-32768 ถึง +32767
unsigned short **int**	16	0 ถึง 65535
long **int**	32	-2147483648 ถึง +2147483647
signed long **int**	32	-2147483648 ถึง +2147483647
unsigned long **int**	32	0 ถึง +4294967295
float	32	3.4E-38 ถึง 3.4E+38
double	64	1.7E-308 ถึง 1.7E+308
long **double**	80	3.4E-4932 ถึง 3.4E+4932

โปรแกรมที่ 2.9 แสดงชื่อชนิดของตัวแปร

```
typedef signed char int8_t;
typedef unsigned char uint8_t;
typedef signed int int16_t;
typedef unsigned int uint16_t;
typedef signed long int int32_t;
typedef unsigned long int uint32_t;
```

จากตัวอย่างข้างต้นจะเห็นได้ว่า ในความเป็นจริงแล้ว ชื่อชนิดของตัวแปร แบบใหม่ๆ ที่มีการประกาศสร้างขึ้นมาใช้งานของ Arduino นั้น ในความเป็นจริงแล้ว มันก็เป็นแค่การกำหนดชื่อใหม่ๆขึ้นมาเท่านั้น แต่คุณสมบัติการใช้งานก็จะยังคงเหมือนกับตัวแปรที่ใช้ใน ภาษาซีมาตรฐานตามแบบของ ANSI-Cอยู่เหมือนเดิม ซึ่งจุดประสงค์จริงๆไม่ได้ต้องการสร้างชนิดของตัวแปรแบบใหม่ขึ้นมาใช้งาน แต่ใช้เพื่อลดความยาวของชื่อ ให้สั้นลง เพื่อที่เวลาเขียนโปรแกรมจะใช้พิมพ์คำสั่งได้สั้นกระชับขึ้น คือ แทนที่จะต้องพิมพ์คำสั่งยาวๆ อย่าง **unsigned long int** ก็ใช้ชื่อ **int32_t** แทน ซึ่งมีความหมายเหมือนกันทุกประการแต่เขียนในรูปแบบที่สั้นและกระชับกว่ากันเท่านั้นเอง

2.1.9 สรุปชนิดของตัวแปรใน Arduino ที่ใช้บ่อยๆ

• **boolean** ใช้เก็บค่าข้อมูล เพียง 2 จำนวน คือ TRUE (จริง) และ FALSE (เท็จ)

• **char** ใช้เก็บค่าข้อมูลขนาด 8 บิต ใช้สำหรับเก็บค่ารหัสของตัวอักษร ซึ่งสามารถกำหนดเป็นค่าหรือ เขียนตัวอักษรไว้ภายใต้เครื่องหมาย ฟันเดี่ยวก็ได้ เช่น 'A' หรือ 0x41 หรือ 65

• **byte** ใช้เก็บค่าข้อมูลขนาด 8 บิตที่เป็นค่าจำนวนเต็มแบบไม่คิดเครื่องหมาย(เหมือนกันกับ unsigned char ในภาษาซี) ซึ่งสามารถเก็บค่าข้อมูลได้ 256 ค่า คือ 0-255

• **int** หรือ Integer ใช้เก็บค่าข้อมูลขนาด 16บิต ที่เป็นค่าจำนวนเต็ม แบบคิดเครื่องหมาย โดยอ

สามารถใช้ เก็บข้อมูลได้ 65536 ค่า คือ -32768 ถึง +32767

• **unsigned int** ใช้เก็บค่าข้อมูลขนาด 16บิต ที่เป็นค่าจำนวนเต็ม แบบไม่คิดเครื่องหมาย โดยสามารถใช้เก็บข้อมูลได้ 65536 ค่า คือ 0-65535

• **long** ใช้เก็บค่าข้อมูลขนาด 32บิต ที่เป็นค่าเลขจำนวนเต็มแบบคิดเครื่องหมาย โดยสามารถใช้เก็บข้อมูลได้ 4294967296 ค่า คือ -2,147,483,648 ถึง 2,147,483,647

• **unsigned long** ใช้เก็บค่าข้อมูลขนาด 32บิต ที่เป็นค่าเลขจำนวนเต็มแบบไม่คิดเครื่องหมายโดยสามารถใช้เก็บข้อมูลได้ 4294967296 ค่า คือ 0 ถึง 4,294,967,295

• **float** ใช้เก็บค่าข้อมูลที่เป็นเลขทศนิยมแบบคิดเครื่องหมายขนาด 32 บิต โดยสามารถเก็บค่าได้ระหว่าง 3.4E-38 ถึง 3.4E+38 (-3.4028235E+38 ถึง 3.4028235E+38)

• **double** ใช้เก็บค่าข้อมูลที่เป็นเลขทศนิยมเช่นเดียวกันกับ float แต่มีค่าความละเอียดกว่าfloatถึง 2 เท่า สามารถเก็บค่าได้มากถึง 1.7E+308

• **void** เป็นตัวแปรแบบที่ไม่มีการเก็บค่าใดๆ คือ ไม่มีค่านั่นเอง

• **arrays** เป็นตัวแปรที่ใช้เก็บข้อมูลหลายๆค่าไว้ในตัวแปรตัวเพียงชื่อเดียว แต่มีตัวเลขสำหรับชี้แหน่งการเก็บข้อมูลต่างกัน โดยตัวเลขที่ใช้ทำหน้าที่เป็นตัวชี้ตำแหน่งของข้อมูล เรียกว่า Index Number โดยค่าลำดับของข้อมูลในตัวแปร array ตำแหน่งแรกจะมีค่าเป็น ศูนย์เสมอ

• **string** เป็นตัวแปรใช้เก็บข้อความ หรือ ตัวอักษรหลายๆตัว ซึ่ง string ก็คือ array ของตัวแปรแบบ char นั่นเอง

• **pointer** เป็นตัวแปรที่ไม่ได้ใช้เก็บข้อมูล แต่ใช้เก็บค่าตำแหน่งแอดเดรสของหน่วยความจำที่ใช้สร้างเป็นตัวแปรสำหรับเก็บข้อมูลซึ่งตัวแปรแบบนี้จะใช้ทำหน้าที่เป็นตัวชี้ไปยังตำแหน่งแอดเดรสของตัวแปรอื่นๆอีกทีหนึ่ง

2.1.10 การแปลงค่าตัวแปรโดยการ cast

ในภาษาซีนั้น ถ้าหากว่ามีการนำค่าในตัวแปร ที่ต่างชนิดกัน มากระทำกัน จะทำให้ได้ผลลัพธ์ที่ผิดไป ดังนั้นจึงมีการจำเป็นต้องมีการแปลงชนิดของตัวแปรให้เป็นชนิดเดียวกันก่อน แล้วจึงนำตัวแปรนั้นมากระทำกันได ซึ่งในภาษาซี จะยอมให้มีการสั่งแปลงค่าข้อมูลในตัวแปรจากประเภทหนึ่งไปเป็นอีกประเภทหนึ่งเป็นการชั่วคราว เพื่อนำไปกระทำกัน โดยที่ค่าในตัวแปรต้นแบบไม่ถูกเปลี่ยนแปลงไปด้วย ซึ่งเราเรียกว๊ะนี้ว่า การ cast ซึ่งมีรูปแบบ ดังนี้

(type)variable

- type หมายถึง ชนิดของตัวแปรที่ต้องการเปลี่ยนค่า
- variable หมายถึง ตัวแปรที่ต้นแบบที่ต้องการนำมาเปลี่ยนค่า

โปรแกรมที่ 2.10 ตัวอย่างการใช้งาน

```
Viod setup () {

Serial.Begin(19200);                    //กำหนดใช้งานพอร์ตอนุกรม Buad =19200

}

  void loop () {
float num1 =5.0 ;                       //กำหนดให้ num1เป็นแบบ float มีค่า =5.0
float num2= 0.5;                        //กำหนดให้ num2เป็นแบบ float มีค่า =0.5
float num3;                             //กำหนดให้ num3 เป็นแบบ float
num3=num1/num2;                         //กำหนดให้ num3=num1หารด้วย num2
serial .println("Example ccast data") //พิมพ์ข้อความ"Example cast data"
serial .print("value of 5.0/0.5")       //พิมพ์ข้อความ" value of 5.0/0.5"
serial.println(int)num3;                //พิมพ์ค่า num3 โดยแปลงค่าnum3 เป็นint
ก่อน

While(1);

}
```

จากตัวอย่างตจะเห็นได้ว่า num1 num2 num3 เป็นตัวแปรแบบ float ทั้งหมด โดยnum1 และ num2 จะถูกกำหนดคาเริ่มต้นให้ตอนสั่งประกาศสร้างตัวปแรเลย ส่วน num3 จะยังไมถูกกำหนดค่าให้กับตัวแปรตอนที่ประกาศสร้างตัวแปร แต่จะสั่งให้เอาค่าของ num 1 หารด้วย num 2 และนำผลลัพธ์ที่ได้ไปเก็บไว้ใน num 3 ส่วนที่น่าสนใจคือ บรรทัดคำสั่ง serial.pribntl(int)num3; ซึ่งเป็นการสั่ง cast ของค่าnum3 ซึ่งเป็นแบบ float ให้เป็นint ก่อนแล้วนำค่าที่ได้ส่งให้กับคำสั่ง serial.println ซึ่งถ้าไม่มีการสั่ง cast ก่อนโปรแกรมนี้จะไม่สามารถสั่ง compile ได้ เนื่องจากคำสั่ง serial.println ของ Arduino ไม่สามารถรับค่าตัวข้อมูลที่เป็นเลขทศนิยมแบบ float ได้ แต่จะรับเฉพาะค่าตัวเลขที่เป็นจำนวนเต็มเท่านั้น

2.2 ตัวแปรต่างๆในภาษาซี

2.2.1 ตัวแปรแบบ array

ตัวแปรแบบ Array จะมีลักษณะแตกต่างจากตัวแปรอื่นๆ คือ ตัวแปรอื่นๆนั้น เมื่อประกาศขึ้นมาใช้งานแล้ว ตัวแปร 1 ตัวก็จะสามารถใช้เก็บข้อมูลได้ 1 ค่า เท่านั้น เมื่อต้องการเก็บข้อมูลหลายๆค่าก็จะต้องสร้างตัวแปรแบบนั้นขุ้นมาหลขายๆตัว ซึ่งถ้ามีจำนวนตัวแปรในโปรแกรมมากๆก็จะส่งผลให้เกิดความยุ่งยากในการกำหนดชื่อและเรียกใช้งาน ภาษาซี จึงได้มีการสร้างตัวแปรแบบย array ขึ้นมาเพื่อใช้แก้ปัญหานี้โดยที่ array เป็นตัวแปรที่ใช้เก็บข้อมูลหลายๆค่าไว้ในตัวแปรตัวเพียงชื่อเดียว แต่มทีตัวเลขสำหรับชี้ตำแหน่งข้อมูลที่ต่างกัน โดยตัวเลขที่ใช้ทำหน้าที่เป็นตัวชี้ตำแหน่งขแงข้อมูลตัวแปรแบบ Array เรียกว่า Index Number โดยค่าลำดับของข้อมูลตัวแปร array ตำแหน่งแรกจะมีค่าเป็น ศูนย์ เสมอ

ตัวแปรแบบ Array สามารถกำหนดให้ใช้สำหรับเก็บข้อมูลแบบใดก็ได้ เพียงแต่มีข้อแม้งว่า ข้อมูลที่เก็บในตัวแปร Array นั้นต้องเป็นแบบเดียวกันเสมอ เช่น Array ของ char,array ของ int,array ของ float, array ของ double เป็นต้น โดยรูปแบบการประกาศสร้างตัวแปรแบบ Array คือ

โปรแกรมที่ 2.11 แสดงรูปแบบตัวแปรแบบ Array

```
ชนิดตัวแปร     ชนิดตัวแปร [size];

ชนิดตัวแปร     ชื่อตัวแปร[size1,size2,size3,...,sizeN];
```

- ชนิดของตัวแปร หมายถึง ชนิดของข้อมูลที่จะเก็บใน Array ของตัวแปร ซึ่งใช้ได้กับทุกตัวแปรทุกประเภทของภาษาซี เช่น char, int, float, double
- ชื่อของตัวแปร[size]หมายถึง ชื่อของตัวแปร array ที่ต้องการจะสร้างขึ้นโดยมีขนาดเท่ากับ m
- Size หมายถึง ขนาดของ array ซึ่งเป็นจำนวนที่จะใช้เก็บข้อมูล โดยการเข้าถึงข้อมูลจะใช้ตัวบ่งชี้ index ซึ่งมีค่า ระหว่าง 0 ถึง size-1 ในการชี้ตำแหน่งข้อมูลใน array
- size1,size2,size3,...,sizeNs หมายถึงขนาดของArray ในกรณที่กำหนด Array เป็นแบบหลายมิติ

2.2.2 ตัวแปรแบบ string

ตัวแปรแบบ String คือตัวแปรที่สำหรับใช้เก็บตัวอกษรหลายๆตัว ซึ่งในความเป็นจริงแล้วตัวแปรแบบ String ก็จัดเป็นตัวแปรแบบ Array ประเภทหนึ่ง เพียงแต่เป็น array ของตัวอักษรแบบ char เท่านั้นเอง ซึ่งข้อแตกต่างระหว่าง string กับ array ก็คือ string จะต้องปิดท้ายข้อมูลด้วยค่าข้อมูลที่เป็นศูนย์เสมอ เพียงแต่ว่าในตอนกำหนดค่า String ไม่ต้องกำหนดค่าศูนย์ด้วย โดยภาษาซีจะเติมค่าศูนย์ปิดท้ายให้เองแบบอัตโนมัติ ดังนั้นในการกำหนดขนาดของ Array ของ string นั้นต้องกำหนดขนาดของArray ให้มีค่ามากกว่าจำนวนตัวอักษรที่ต้องการจัดเก็บ 1 ค่าเสมอ ซึ่งภาษาซี ของ Arduino สามารถประกาศตัวแปร string ได้หลายรูปแบบ

2.2.3 ตัวแปรแบบ Pointer

Pointer เป็นตัวแปรที่ไม่ได้ใช้เก็บข้อมูล แต่ใช้เก็บค่าตำแหน่งแอดเดรสของหน่วยความจำที่ใช้สร้างเป็นตัวแปรสำหรับข้อมูลแทน โดยตัวแปรแบบนี้จะใช้ทำหน้าที่เป็นตัวบ่งชี้ไปยังตำแหน่งแอดเดรสของตัวแปรอื่นๆอีกทีหนึ่งโดยมีรูปแบบการใช้งานดังนี้

โปรแกรมที่ 2.12 แสดงรูปแบบตัวแปรแบบ Pointer

```
ชนิดของตัวแปร*ชื่อตัวแปร
```

- ชนิดของตัวแปร หมายถึง ชนิดของตัวแปรที่ต้องการให้ pointer ชี้ไปหา
- *ชื่อของตัวแปร หมายถึง ชื่อของตัวแปรแบบ pointer ซึ่งกำหนดให้ต้องใช้เครื่องหมาย*นำหน้าชื่อและห้ามเว้นวรรคด้วย

2.2.4 ขอบเขตของตัวแปร

ในเรื่องของตัวแปรนั้น นอกจากเรื่องของคุณสมบัติในการใช้เก็บข้อมูลของตัวแปรยังมีสิ่งหนึ่งที่เราต้องเรียนรู้เพิ่มเติมเกี่ยวกับคุณสมบัติของตัวแปร ก็คือ เรื่องของขอบเขตในการใช้งานของตัวแปร ซึ่ง ถึงแม้ว่าจะเป็นตัวแปรชนิดเดียวกันแต่ถ้ามีการประกาศใช้งาน ต่างกัน ก็จะมีคุฯสมบัติเรื่องขอบเขตการใช้งานที่แตกต่างกันออกไปด้วย โดยภาษาซี ได้จำแนก ชนิดของตัวแปรตามขอบเขตการเรียกใช้งานออกเป็นดังนี้

- Local หรือ automatic หมายถึง ตัวแปรที่ประกาศขึ้นในส่วนเริ่มต้นของโปรแกรมใน block {} โดยตัวโปรแกรมนี้จะเรียกใช้ได้เฉพาะภายใน Block ที่ประกาศไว้เท่านั้น เมื่อโปรแกรมจบการทำงานจาก block หรือก ออกนอก block ไป หน่วยความจำที่ใช้สร้างตัวแปรนั้นจะถูกยกเลิกไป การประกาศตัวแปรแบบนี้ควรกำหนดค่าเริ่มต้นให้กับตัวแปรด้วย เนื่องจาก complier จะไม่กำหนดค่าเริ่มต้นให้กับตัวแปร แต่จะยืถือเอาค่าข้อมูลเดิมที่ค้างอยู่ในหน่วยความจำที่ถูกนำมาใช้สร้างตัวแปร เป็นค่าเริ่มต้นแทน
- Global หรือ external หมายถึง ตัวแปรที่สามารถใช้งานเริ่มกันทุกฟังก์ชั่นในโปรแกรม โดยมีขอบเขตการใช้งานตั้งแต่จุดที่เริ่มต้นประกาศตัวแปร ไปจนสิ้นสุดของขอบเขตโปรแกรม โดยตัวแปรแบบนี้จะประกาศก่อนฟังก์ชั่นหลัก main() เพื่อทำให้ทุกฟังก์ชั่นในโปรแกรมสามารถเรียกใช้งานได้
- Static มีขอบเขตการใช้งานเฉพาะภายใน block ที่กำหนดไว้เท่านั้น เพียงแต่มีความพิเศษตรงที่เมื่อโปรแกรมสออกนอก block ไปแล้วค่าของตัวแปรจะยังไม่หมดสภาพไป ค่าต่างๆยังคงอยู่ และสามารถเรียกใช้งานได้ครั้งต่อไป
- Volatile เป็น keyword ใช้ประกาศเพื่อบอกให้ complier รู้ว่าตัวแปรนี้ มีการเปลี่ยนแปลงค่าอยู่ตลอดเวลา และทุกๆโปรกแกรม ทั้งจากโปรแกรมหลัก โปรแกรมย่อย หรือ interrupt เพื่อป้องกันไม่ให้ complier มาทำการลดขนาด code (Optimize Code)ส่วนที่จะมากระทำกับตัวแปรที่ประกาศไว้นี้
- Const ใช้ประกาศเพื่อบอกให้ complier รู้ว่าตัวแปรที่ประกาศไว้นั้นเป็นค่าคงที่

- PROGMEM เป็น keyword เพื่อบอกให้ complier รู้ว่าตัวแปรที่ประกาศนี้ต้องการสร้างและเก็บไว้ในหน่วยความจำที่ใช้เก็บโปรแกรมของ MCU ตระกูล AVR ซึ่งใช้กับตัวแปรที่ต้องการสร้างไว้เพื่ออ่านออกมาใช้งานเพียงอย่างเดียว ไม่มีการเปลี่ยนแปลงค่าในขณะที่โปรแกรมทำงานอยู่ ซึ่งควรประกาศ PROGMEM เพื่อประหยัดหน่วยความจำ SRAM ในการนำมาใช้สร้างตัวแปร

2.2.5 ค่างคงที่ใน Arduino

ในภาษา Arduino มีการกำหนดค่าตัวแปรที่แบบค่าคงที่ (constant) ไว้จำนวนหนึ่ง ให้ผู้เรียกใช้เพื่ออำนวยความสะดวกและช่วยให้สามารถอ่านความหมายของคำสั่งในโปรแกรมได้ง่ายและตรงความหมายมากยิ่งขึ้น

- FALSE เป็นค่าคงที่แบบ Boolean มีค่าเป็น ศูนย์ หรือ เท็จ หรือ FALSE
- TRUE เป็นค่าคงที่แบบ Boolean มีค่าเป็นค่าใดๆที่ไม่ใช่ศูนย์ ซึ่งอาจจะเป็น 1 หรือ -1 หรือค่าอื่นใดก็ได้ที่ไม่ใช่ศูนย์ จะถือเป็น จริง หรือ FALSE ทั้งหมด
- HIGH ใช้แทนสถานะโลจิก HIGH หรือ Logic “1”
- LOW ใช้แทนสถานะโลจิกLOW หรือ Logic “0”
- INPUT ใช้ในการกำหนดค่าสถานะ Input ให้กับฟังก์ชั่น pinModel()
- OUTPUT ใช้ในการกำหนดค่าสถานะ Output ให้กับฟังก์ชั่น pinModel()

นอกจากค่าคงที่ในส่วนที่มีการ จองหรือกำหนดไว้แล้วจาก Complier ใน Arduino ยังค่าคงที่อีกแบบหนึ่ง ซึ่งก็คือ ค่าคงที่แบบตัวเลขจำนวนเต็มที่ถูกกำหนดจากผู้ใช้เอง ซึ่งได้แก่ค่าคงที่อยู่ในรูปของค่าตัวเลขจำนวนเต็มแบบต่างๆ แต่เนื่องจาก คุณสมบัติของตัวแปร สำหรับใช้ในการเก็บค่าตัวเลขจำนวนเต็มตัวนั้นจะมีอยู่มากมายหลายแบบ เช่น signed , unsigned ,sort, long ดังนั้นในการกำหนดค่าตัวแปรให้กับ complier ก็ต้องมีการระบุให้ complier ทราบด้วยว่า ตัวเลขที่กำหนดให้ไปนั้นเป็นค่าตัวเลขแบบใดโดย Arduino กำหนดให้ใช้ รหัสตัวอักษร ต่อท้ายตัวเลขเพื่อกำหนดค่าคุณสมบัติเฉพาะของตัวเลข ตามต้องการ โดยใช้รหัสตัวอักษร U และ L ต่อท้ายตัวเลขเพื่อบ่งบอกถึงคุณสมบัติเฉพาะของตัวเลขในแบบที่เราต้องการ ซึ่งจะใช้ตัวอักษรตัวเล็กหรือตัวใหญ่ก็ได้มีความหมายเหมือนกัน คือ

- U หรือ u ใช้บอกให้ complier รู้ตัวเลขที่กำหนดเป็นแบบ unsigned เช่น 33U
- l หรือ L ใช้บอกให้ complier รู้ตัวเลขที่กำหนดเป็นแบบ Long เช่น 100000L
- ul หรือUL ใช้บอกให้ complier รู้ตัวเลขที่กำหนดเป็นแบบ unsigned long เช่น 32767UL

2.2.6 นิพจน์และตัวดำเนินการของArduino

นิพจน์ (Expression) คือ ประโยคคำสั่งซึ่งถือเป็นองค์ประกอบพื้นฐานที่ใช้สำหรับสร้างประโยคแบบที่เป็นเชิงซ้อนของภาษาซี โดยนิพจน์ใช้สำหรับใช้ในการสั่งให้เอาจำนวน 2 จำนวนมากระทำกันโดยมีเครื่องหมาย (operator) เป็นตัวแสดงการกระทำกันของจำนวนทั้ง2 โดยค่าค่าของจำนวนทั้ง2 นั้นอาจเป็นค่าคงคี่ ค่าตัวแปร หรือ ข้อมูลใดๆ ก็ได้

ตัวดำเนินการ (Operator) คือ เครื่องหมายใดๆ ที่ใช้ทำหน้าที่แสดงการกระทำกันของจำนวน2 จำนวน ซึ่งอาจเป็น การกระทำกันระหว่างข้อมูลกับข้อมูล ข้อมูลกับตัวแปร หรือ ตัวแปร

กับตัวแปรก็ได้ โดยตัวดำเนินการ ที่ใช้ในภาษาซี จะมีอยู่ด้วยกันหลายประเภท เพื่อใช้ทำหน้าที่ที่แตกต่างกัน อันได้แก่

- เครื่องหมายทางคณิตศาสตร์ (Arithmetric operator)
- เครื่องหมาย การกำหนดค่า (Assignment operator)
- เครื่องหมาย แบบยูนารี (Unary operator)
- เครื่องหมาย เปรียบเทียบ (Comparative operrator)
- เครื่องหมาย เชิงตรรก หรือ การกระทำทางโลจิก (Logical operator)
- เครื่องหมาย การกระทำแบบบิต
- เครื่องหมาย การกำหนดเงื่อนไข (Conditional operator)

2.2.6.1 เครื่องหมายทางคณิตศาสตร์

- + ใช้แทนการบวก
- - ใช้แทนการลบ
- * ใช้แทนการคูณ
- / ใช้แทนการหาร
- % ใช้แทนการ mod หรือ การหารแบบเอาเศษที่ได้จากการหารมาใช้

2.2.6.2 เครื่องหมายสำหรับกำหนดค่า

- = ใช้ในการกำหนดค่าให้ตัวแปร
- ++ การเพิ่มค่า
- -- การลดค่า
- +=การเพิ่มค่าเท่ากับค่าทางขวา
- -= การลดค่าเท่ากับค่าทางขวา
- *= นำตัวเองคูณกับค่าทางขวา
- /= นำตัวเองหารกับค่าทางขวา
- %= นำตัวเอง mod กับค่าทางขวา

ในภาษาซี สามารถกำหนดค่าให้กับตัวแปรได้อีกแบบหนึ่ง ซึ่งเรียกว่าการให้ค่าแบบ ยูนารี โดยใช้เครื่องหมาย++ หรือ – ในการกำหนดค่า ตัวอย่างเช่น

- X++ เป็นการเพิ่มค่าให้ i ซึ่งมีความหมายเหมือนกับ x=x+1
- X--เป็นการลดค่าให้ i ซึ่งมีความหมายเหมือนกับ x=x-1

2.2.6.3 เครื่องหมายที่ใช้ในการเปรียบเทียบ

- == มีความหมาย หมายถึง เท่ากับ
- != มีความหมาย หมายถึง ไม่เท่ากับ
- < มีความหมาย หมายถึง น้อยกว่า
- > มีความหมาย หมายถึง มากกว่า
- <= มีความหมาย หมายถึง น้อยกว่าหรือเท่ากับ
- >= มีความหมาย หมายถึง มากกว่าหรือเท่ากับ

2.2.6.4 เครื่องหมาการกระทำทางโลจิก

- && มีความหมาย หมายถึง Logical AND ซึ่งเปรียบเทียบได้กับ และ
- || มีความหมาย หมายถึง Logical OR ซึ่งเปรียบเทียบได้กับ หรือ

- ! มีความหมาย หมายถึง Logical NOT ซึ่งเปรียบเทียบได้กับ ไม่

2.2.6.5 เครื่องหมายการกระทำแบบบิต

- & มีความหมาย หมายถึง Bitwise AND ซึ่งเป็นการ AND ของบิต
- | มีความหมาย หมายถึง Bitwise OR ซึ่งเป็นการOR ของบิต
- ^ มีความหมาย หมายถึง Bitwise XOR ซึ่งเป็นการ XOR ของบิต
- ~มีความหมาย หมายถึง Bitwise NOT ซึ่งเป็นการ NOT ของบิต
- >> มีความหมาย หมายถึง Bit Shift Right ซึ่งเป็นการเลื่อนบิตไปทางขวา
- <<มีความหมาย หมายถึง Bit Shift Right ซึ่งเป็นการเลื่อนบิตไปทางซ้าย

2.2.6.6 เครื่องหมายสัญลักษณ์อื่นๆที่ใช้

- ; เครื่องหมาย เซมิโคลอน (;) ใช้สำหรับบอกการสิ้นสุดของประโยคคำสั่ง
- { } เครื่องหมายวงเล็บปีกกา (Curly Braces) ใช้กำหนดขอบเขตของประโยคคำสั่งโปรแกรมย่อยๆต่างๆ
- // เครื่องหมายแสดงจุดเริ่มต้นของคำอธิบาย (comment) โดยข้อความใดๆที่ตามหลังเครื่องหมายนี้ภายในบรรทัดเดียวกัน จะถือว่าเป็นคำอธิบาย และจะไม่ถูกนำไปแปลด้วย
- /* */ เครื่องหมายแสดงขอบเขตของการเขียนอธิบาย (comment) โดยข้อความใดๆที่อยู่ระหว่างเครื่องหมาย /* ไปจนถึงเครื่องหมาย */ โปรแกรมจะถือว่าเป็นคำอธิบายทั้งหมด และจะไม่นำมาแปลด้วย

2.3 คำสั่งในภาษาซีของ Arduino

2.3.1 การตรวจสอบเงื่อนไขของภาษาซีของ Arduino

ในการเขียนโปรแกรมทุกภาษานั้นมีสิ่งหนึ่งที่ขาดไม่ได้คือ การสร้างเงื่อนไขให้กับโปรแกรมเพื่อให้ได้ผลลัพธ์การทำงานตามที่เราต้องการ ในภาษาซี ของ Arduino ก็เช่นกัน ที่ได้มีการสร้างคำสั่งสำหรับตรวจสอบเงื่อนไขต่างๆไว้รองรับการเขียนโปรแกรม

- If
- If...else
- For
- Switch case
- While
- Do...while
- Break
- Continuous
- Goto
- Return

โดยคำสั่งสำหรับตรวจสอบเงื่อนไขเหล้านี้จะใช้ควบคู่กับเครื่องหมาย เปรียบเทียบ และเครื่องหมายทางโลจิก สำหรับใช้กำหนดเงื่อนไขการทำงานให้กับคำสั่ง ซึ่งได้แก่

- == มีความหมายว่า เท่ากับ
- != มีความหมายว่า ไม่เท่ากับ
- < มีความหมายว่า น้อยกว่า
- > มีความหมายว่า มากกว่า
- <= มีความหมายว่า น้อยกว่าหรือเท่ากับ
- >= มีความหมายว่า มากกว่าหรือเท่ากับ
- && มีความหมายว่า Logical AND เปรียบเทียบได้กับ และ
- || มีความหมายว่า Logical OR เปรียบเทียบได้กับ หรือ
- ! มีความหมายว่า Logical NOT เปรียบเทียบได้กับ ไม่

2.3.2 คำสั่ง if

คำสั่ง if เป็นคำสั่งสำหรับใช้ตรวจสอบเงื่อนไข เพื่อสั่งให้โปรแกรมเลือกทำงาน ตามผลลัพธ์ที่ได้จากการตรวจสอบเงื่อนไขของคำสั่ง โดยมีรูปแบบคำสั่งดังนี้คือ

โปรแกรมที่ 2.13 แสดงรูปแบบคำสั่ง if

```
If(เงื่อนไข) {
คำสั่งที่ต้องการกระทำเมื่อเงื่อนไขเป็นจริง
 }
```

การทำงานของโปรแกรม เมื่อใช้การตรวจสอบเงื่อนไขแบบนี้ คือ ถ้าเงื่อนไขเป็นจริงก็จะทำงานตามคำสั่งอยู่หลังเงื่อนไข แต่ถ้าเงื่อนไขเป็นเท็จจะข้ามคำสั่งที่อยู่หลังเงื่อนไข

โปรแกรมที่ 2.14 แสดงตัวอย่างการใช้งานคำสั่ง if

```
Viod setup
{
Serial.begin(19200);              //เปิดใช้งานพอร์ตอนุกรม baud 19200
Serial.begin("Press any Key")  //พิมพ์ข้อความ "Press any Key"
  }
 void loop () {
if(Serial.aviable()>0)            //ถ้าข้อมูลใน Buffer มากกว่า ศูนย์
}
 int var =Serial.read();          //อ่านข้อมูลจาก Buffer มาไว้ในตัวแปรชื่อ
vaSerial.println(var,BYTE)              //สั่งพิมพ์ข่อมูลในตัวแปร var
  }
 }
```

การทำงานขอโปรแกรมตัวอย่าง จะเห็นได้ว่า ในโปรแกรมใช้คำสั่ง if เพื่อตรวจสอบเงื่อนไขว่าจำนวนของข้อมูลในBuffer มีค่ามากกว่า ศูนย์หรือไม่ ซึ่งถ้าพบว่ามีจำนวนข้อมูลใน Buffer มากกว่า ศูนย์หรือไม่ ซึ่งถ้าพบว่ามีจำนวนใน Buffer แล้ว โปรแกรมก็จะทำการสั่งการอ่านข้อมูลออกมาจาก Buffer แล้วสั่งพิมพ์ข้อมูลที่รับได้นั้นกลับมาให้เห็นทางพอร์ตสื่อสารอนุกรมด้วย

แต่ถ้าผลการตรวจสอบเงื่อนไขไม่เป็นจริง คือ จำนวนข้อมูลใน Buffer ยังเป็นศูนย์ อยู่ในโปรแกรมก็จะวนรอบตรวจสอบเงื่อนไขใหม่โดยไม่ทำอะไร

2.3.3 คำสั่ง if...else แบบ2 ทางเลือก

คำสั่ง if...else เป็นการสั่งตรวจสอบเงื่อนไขเช่นเดียวกัน if แต่ใช้สำหรับการตรวจสอบเงื่อนไขที่มีหลายเพิ่มขึ้นอีก 1 ทางเลือก โดยมีรูปแบบคำสั่งดังนี้

โปรแกรมที่ 2.15 แสดงรูปแบบคำสั่ง if...else แบบ2 ทางเลือก

```
If (เงื่อนไข){
  คำสั่งที่ต้องการให้ทำเมื่อเงื่อนไขเป็นจริง
}
 else
 {
คำสั่งที่ต้องการให้ทำเมื่อเงื่อนไขเป็นเท็จ
  }
```

ซึ่งจากรูปแบบการใช้คำสั่ง if...else แบบนี้ มีความหมายเหมือนกับคำว่า ถ้าเงื่อนไขเป็นจริงให้ทำอย่างนี้ ไม่เช่นนั้นให้ทำอย่างนั้น ให้ทำอย่างนั้นซึ่งจะเห็นว่า โปรแกรมจะมีทางเลือกในการทำงานเพิ่มขึ้นมากกว่าการใช้คำสั่ง if อีก 1 ทางเลือก รวมเป็น2 ทางเลือกที่จะทำให้โปรแกรมทำงานเมื่อเงื่อนไขเป็นเท็จ

โปรแกรมที่ 2.16 ตัวอย่างการใช้งานคำสั่ง if แบบ 2 ทางเลือก

```
Viod setup () {
serial.begin(19200);                    //เปิดใช้งานพอร์ตอนุกรมด้วย baud 19200
serial.println("Press Any Key");        //พิมพ์ข้อความ"Press Any Key"
 }
 void loop() {
 if (serial.available()>0)              //ถ้าจำนวนข้อมูลใน Buffer มากกว่า ศูนย์
 }
int var = serial.read();                //อ่านข้อมูลจาก Buffer มาไว้ในตัวแปรชื่อ var
serial.println(var,BYTE);               //สั่งพิมพ์ค่าข้อมูลในตัวแปรvar
  }
 else
{} }
```

2.3.4 คำสั่ง if...else แบบหลายเงื่อนไข

คำสั่ง if...else แบบหลายเงื่อนไขเช่นเดียวกันกับ if...else แต่ใช้สำหรับการตรวจสอบเงื่อนไขที่มีมากกว่า 1 เงื่อนไข โดยมีรูปแบบคำสั่งดังนี้

โปรแกรมที่ 2.17 คำสั่ง if...else แบบหลายเงื่อนไข

```
If(เงื่อนไขที่ 1)
{
คำสั่งที่ต้องการให้ทำเมื่อเงื่อนไขที่ 1 เป็นจริง
}
else if (เงื่อนไขที่ 2)
{
คำสั่งที่ต้องการให้ทำเมื่อเงื่อนไขที่ 2 เป็นจริง
}
.
.
else if (เงื่อนไขที่ n)
{
คำสั่งที่ต้องการให้ทำเมื่อเงื่อนไขที่ n เป็นจริง
{
else
{
คำสั่งที่ต้องการให้เงื่อนไขเป็นเท็จ
}
```

ซึ่งจากรูปแบบการใช้คำสั่ง if...else แบบนี้ มีความหมายเหมือนกับประโยคที่ว่า ถ้าเงื่อนไขที่ 1 เป็นจริง ให้ทำงานที่ 1 ไม่เช่นนั้นให้ตรวจสอบเงื่อนไขที่2 และถ้าเงื่อนไขที่2เป็นจริงให้ทำงานที่2 ไม่เช่นนั้นให้ตรวจสอบเงื่อนไขที่3 เป็นจริงให้ทำงานที่3 ไม่เช่นนั้นให้ตรวจสอบเงื่อนไขที่ n และถ้าเงื่อนไขที่ n เป็นจริงก็ให้ทำงานที่ n

ซึ่งจะเห็นได้ว่า เราสามารถจะทำการเพิ่มเติมเงื่อนไขให้กับโปรแกรมเพื่อใช้เป็นทางเลือกในการเลือกทำงานตามคำสั่งต่างๆตามความเหมาะสมได้หลายทางเลือกเลยทีเดียว ตามความต้องการ ทำให้เขียนโปรแกรมสามารถทำได้ง่ายและสะดวกมากขึ้น

โปรแกรมที่ 2.18 ตัวอย่างการใช้คำสั่ง if...else แบบหลายเงื่อนไข(1)

```
Viod menu ()                    //โปรแกรมย่อยพิมพ์เมนู
 {
serial.printl();                //ขึ้นบรรทัดใหม่
serial.printl("Number1=10");     //พิมพ์ข้อความ "Number1=10"
serial.printl("Number2=5");      //พิมพ์ข้อความ "Number1=5
serial.printl("1=Number1+Number2");//พิมพ์ข้อความ "1=Number1+Number2"
serial.printl("2=Number1-Number2"); //พิมพ์ข้อความ "2=Number1-Number2"
serial.printl("3=Number1×Number2");//พิมพ์ข้อความ
"3=Number1×Number2"serial.printl("4=Number1/Number2"); //พิมพ์ข้อความ
"4=Number1/Number2"
 serial.print("select...");                //พิมพ์ข้อความ "select..."
 }
 viod setup(){
 serial.begin(19200);           //เปิดใช้พอร์ตอนุกรมด้วยbaud 19200
menu();                         //พิมพ์เมนู
}
 viod loop () {
while(serial.aviable()>0)       //รอจนกว่าจำนวนข้อมูลใน Buffer มากกว่าศูนย์
{
int var=serial.read();          //อ่านข้อมูลมาจาก Buffer
serial.println(var,BYTE);
if(var=='1')                    //ถ้าค่าใน val เท่ากับรหัส ASCII ของ '1'
 }
serial.print("10+5=");           //พิมพ์ข้อความ "10+5="
serial.printLn(10+5);           //พิมพ์ค่า10+5
 }
```

ต่อส่วนของหน้าที่แล้ว

โปรแกรมที่ 2.19 ตัวอย่างการใช้คำสั่ง if...else แบบหลายเงื่อนไข(2)

```
else(var=='2')   {
serial.print("10+5=");                    //พิมพ์ข้อความ "10-5="
serial.printLn(10+5);                     //พิมพ์ค่า10-5
 }
else(var=='3') {
serial.print("10×5=");                    //พิมพ์ข้อความ "10×5="
serial.printLn(10×5);                     //พิมพ์ค่า10×5
 }
else(var=='4') {
serial.print("10/5=");                    //พิมพ์ข้อความ "10/5="
serial.printLn(10/5);                     //พิมพ์ค่า10/5
}
else {
serial.println("select Number Error"); //พิมพ์ข้อความ"select Number Error"
}
menu();                                   //พิมพ์เมนู
}
}
```

โปรแกรมที่ 2.20 ผลลัพธ์การทำงานขอตัวอย่างการใช้คำสั่ง if...else แบบหลายเงื่อนไข

```
Number=10
Number=5
1=Number1+Number2
2=Number1×Number2
3=Number1/Number2
4=Number1+Number2
select ...1
10+5=15
Number1=10
Number2=5
1=Number1+Number2
2=Number1×Number2
3=Number1/Number2
4=Number1+Number2
select ...5
Select Number Error
Number1=10
Number2=5
1=Number1+Number2
2=Number1×Number2
3=Number1/Number2
4=Number1+Number2
select ...
```

จากตัวอย่างโปรแกรมข้างต้น เป็นตัวอย่างที่แสดงให้เห็นการใช้คำสั่ง if ...else แบบหลายทางเลือกในการทำงาน โดยการทำงานของโปรแกรมจะเริ่มต้นด้วยการ กำหนดค่าการใช้งานพอร์ตสื่อสารอนุกรม จากนั้นก็พิมพ์ข้อความแสดงเมนู เพื่อบอกให้ผู้ใช้เลือกตัวเลข 1 ถึง 4 แล้วก็ทำงานในวงรอบเพื่อรับข้อมูลจากผู้ใช้ โดยถ้าพบว่าการรับข้อมูลมาเก็บรอไว้ใน Buffer ก็ทำการอ่านข้อมูลออกมาแล้วใช้คำสั่ง if เพื่อตรวจสอบเงื่อนไขว่าค่าที่ได้มีค่าเท่ากับ'1'หรือ 2 หรือ3 หรือ4 หรือไม่ ถ้าตรงกับเงื่อนไขใดก็ให้ทำงานตามคำสั่งในเงื่อนไขนั้น คือ

- ถ้ารับค่าได้เท่ากับ1 ก็นำค่า10 บวกกับ5 แล้วพิมพ์ออกมาให้เห็น
- ถ้ารับค่าได้เท่ากับ2 ก็นำค่า10 ลบกับ 5 แล้วก็พิมพ์ออกมาให้เห็น
- ถ้ารับค่าได้เท่ากับ3 ก็นำค่า10 คูณกับ 5 แล้วก็พิมพ์ออกมาให้เห็น
- ถ้ารับค่าได้เท่ากับ4 ก็นำค่า10 หารกับ 5 แล้วก็พิมพ์ออกมาให้เห็น
- ถ้ารับค่าอื่นที่ไม่ใช่ 1,2,3,4 ก็สั่งพิมพ์เป็น select number error

2.3.5 คำสั่ง for

คำสั่ง for เป็นคำสั่งสำหรับสั่งโปรแกรมวนรอบทำงาน โดยมีการกำหนดค่าเริ่มต้นและเงื่อนไขการสิ้นสุดที่แน่นอน โดยมีรูปแบบดังนี้

โปรแกรมที่ 2.21 รูปแบบคำสั่ง for

```
For(ค่าเริ้มต้น;เงื่อนไขการวนรอบ;การเพิ่มค่าหรือลดค่าตัวแปรในแต่ละรอบ)
{
คำสั่งที่ต้องการให้ทำ ถ้าเงื่อนไขวนรอบยังไม่เสร็จ
}
```

โปรแกรมที่ 2.22 ตัวอย่างการใช้งานคำสั่ง for

```
Viod setup () {
serial.begin()19200;                    //เปิดใช้งานพอร์ตอนุกรมด้วย baud 19200
                }
void loop() {
serial.println("Test for");             //พิมพ์คำว่า "Test for"
for (int x=0; x<10; x++)                //ให้วนรอบโดยกำหนดให้ x=0 ถึง 9
{
serial.print("x=")                      //พิมพ์ข้อความ "x ="
serial.println(x);                      //พิมพ์ค่าของ x
delay(50);                              //หน่วงเวลา 50mS
}
 }
```

โปรแกรมที่ 2.23 ผลลัพธ์การทำงาน

```
Test for
x=0
x=1
x=2
x=3
x=4
x=5
x=6
x=7
x=8
x=9
Test for
x=0
```

2.3.6 Switch/case

คำสั่ง Switch/case ใช้สำหรับสั่งตรวจสอบเงื่อนไข เพื่อเลือกให้โปรแกรมทำงานตามเงื่อนไขที่ต้องการเพียงเงื่อนไขเดียว ซึ่งผลของการทำงานของคำสั่ง จะเหมือนกับ if...else แบบหลายเงื่อนไขเพียงแต่มีรูปแบบการใช้งานที่เป็นระเบียบและใช้งานได้ง่ายกว่า ซึ่งการทำงานของคำสั่งนี้ จะเป็นการนำค่าในตัวแปรที่กำหนดให้ไปทำการเปรียบเทียบกับค่าคงที่ที่กำหนดไว้ ถ้าตรงกันก็จะให้เงื่อนไขเป็นจริงและโปรแกรมจะทำงานตามคำสั่งที่อยู่หลังเงื่อนไขการเปรียบเทียบค่านั้นๆ เมื่อตรวจสอบเสร็จก็จะไปตรวจสอบเงื่อนไขต่อไป จนถึงลำดับสุดท้าย ถ้าไม่ผลการเปรียบเทียบไม่ตรงกับเงื่อนไขใดเลย ก็จะไปทำงานตามคำสั่งที่อยู่ส่วน default รูปแบบการใช้งานคำสั่ง

โปรแกรมที่ 2.24 แสดงรูปแบบคำสั่งSwitch/case

```
Switch(ตัวแปรที่จะนำมาตรวจสอบ)
 { case ค่าคงที่ 1 :คำสั่งที่ต้องการให้ทำงานเมื่อการตรวจสอบค่าคงที่ 1 เป็นจริง
case ค่าคงที่2 :คำสั่งที่ต้องการให้ทำงานเมื่อการตรวจสอบค่าคงที่ 1 เป็นจริง

 case ค่าคงที่n :คำสั่งที่ต้องการให้ทำงานเมื่อการตรวจสอบค่าคงที่ n เป็นจริง
default  : คำสั่งที่ต้องการให้ทำงานเมื่อไม่ตรงกับเงื่อนไขใดๆ
}
```

2.3.7 คำสั่ง while

คำสั่ง While เป็นคำสั่งให้โปรแกรมวนรอบทำงานซ้ำๆกัน (loop) เช่นเดียวกับคำสั่ง for แต่มีความแตกต่างกันที่ คำสั่ง for จะมีจำนวนรอบทำงานที่แน่นนอน แต่คำสั่ง while จะทำงานในวงรอบไม่รู้จบจนกว่าเงื่อนไขจะเป็นเท็จจึงจะสิ้นสุดการทำงานในวงรอบ โดยมีรูปแบบดังนี้

โปรแกรมที่ 2.25 แสดงรูปแบบคำสั่ง while

```
While(เงื่อนไข)
{
คำสั่งที่ต้องการให้ทำงานเมื่อเงื่อนไขยังเป็นจริงอยู่
}
```

โปรแกรมที่ 2.26 ตัวอย่างการใช้งานคำสั่ง while

```
Viod setup ()
{
serial.begin(19200);                    //เปิดใช้งานพอร์ตอนุกรมาด้วย baud 19200
                        }
 viod loop() {
serial.println("Test while");           //พิมพ์ข้อความ "Test while"
int x=0;                                //สร้างตัวแปรชื่อ x เป็นแบบ int
 while(x<=5)                            //วนรอบจนกว่า x มีค่ามากกว่า 5
 {
serial.print("x=");                     //พิมพ์ข้อความ"x="
serial.println(x);                      //พิมพ์ค่าของ x
x++;                                    //เพิ่มค่า x=1
}
}
```

โปรแกรมที่ 2.27 ผลลัพธ์การทำงานตัวอย่าง while

```
Test do while
x=0
x=1
x=2
x=3
x=4
x=5
Test do while
x=0
```

2.3.8 คำสั่ง do...while

คำสั่ง do...while ใช้สำหรับให้โปรแกรมวนรอบทำงานเหมือนกันกับ for และ while แต่ลักษณะการทำงานจะแตกต่างกัน โดยคำสั่งนี้จะทำงานตามคำสั่งหลัง do ก่อน แล้วจึงตรวจสอบเงื่อนไขแต่while ตรวจสอบเงื่อนไขก่อนแล้วจึงทำงานตามคำสั่งที่อยู่หลังเงื่อนไข โดยถ้าผลการตรวจสอบเงื่อนไขยังเป็นจริงอยู่ก็จะกลับไปเริ่มต้นทำงานตามคำสั่งที่อยู่หลัง do อีก จนกว่าจะพบว่าเงื่อนไขเป็นเท็จจึงจะจบจากคำสั่งโดยรูปแบบของคำสั่งเป็นดังนี้

โปรแกรมที่ 2.28 แสดงรูปแบบคำสั่ง do...while

```
Do (เงื่อนไข)
{
คำสั่งที่ต้องการให้ทำงาน
} while (เงื่อนไข);
```

โปรแกรมที่ 2.29 ตัวอย่างการใช้งานคำสั่ง do...while

```
Viod setup()
{
serial.begin(19200);                    //เปิดใช้พอร์ตอนุกรมด้วย baud 19200
}
viod loop()
{
serial.println("Test do...while");      //พิมพ์ข้อความ "Test do...while"
int x=0;                                //สร้างตัวแปรชื่อ xเป็นแบบ int
do
{
serial.print("x=");                     //พิมพ์ข้อความ "x="
serial.println(x);                      //พิมพ์ค่าของ x
x++;                                    //เพิ่มค่า x=1
} while(x<=5);                          //วนรอบจนกว่าxมีค่ามากกว่า 5
}
```

โปรแกรมที่ 2.30 ผลลัพธ์ตัวอย่างการใช้งานคำสั่ง do...while

```
Test do while
x=0
x=1
x=2
x=3
x=4
x=5
Test do while
x=0
.
```

จากตัวอย่างการใช้คำสั่ง while และ do..while ข้างต้นจะเห็นว่าได้ผลการทำงานที่เหมือนกันคือ ค่าของ x ถูกพิมพ์ ด้วยค่า 0 ถึง 5 แต่ถ้าลองเปลี่ยนค่าเริ่มต้นของ x จาก0 เป็น 6 ดูจะเห็นได้ว่าคำสั่ง do...while จะทกงานผิดพลาด คือมีการสั่งพิมพ์ค่า x=6 ออกมาด้วย1 ครั้ง แต่คำสั่ง while จะทำงานได้ถูกต้องโดยไม่มีการพิมพ์ค่าของ x เลย ที่เป็นเช่นนี้ก็เพราะว่า คำสั่ง while จะตรวจสอบเงื่อนไขก่อนเข้าทำงานในวงรอบ แต่คำสั่ง do...while จะทำงานในวงรอบก่อน แล้วจึงตรวจสอบเงื่อนไขภายหลัง นั่นเอง

โปรแกรมที่ 2.31 ผลลัพธ์การทำงานของคำสั่ง while เมื่อกำหนดค่าเริ่มต้นให้ x=6

```
Test while
Test while
```

โปรแกรมที่ 2.32 ผลลัพธ์การทำงานของคำสั่ง do...while เมื่อกำหนดค่าเริ่มต้นให้ x=6

```
Test do...while
x=6
Test while
```

2.3.9 คำสั่ง break

คำสั่ง break จะใช้ร่วมกับคำสั่งประเภทที่ทำงานแบบวนรอบ เช่น do, for, while เพื่อให้โปรแกรมหยุดการทำงานหรือจบการทำงานจากวงรอบโดยไม่สนใจเงื่อนไข นอกจากนี้แล้วยังใช้คำสั่ง break สำหรับสั่งให้โปรแกรมจบการทำงานของคำสั่ง switch เพื่อข้ามการตรวจสอบเงื่อนไขต่อไปใน

โปรแกรมที่ 2.33 ตัวอย่างการใช้งานคำสั่ง break

```
Viod setup()
{
serial.begin(19200);                    //เปิดใช้พอร์ตอนุกรมด้วย baud 19200
}
viod loop()
{
serial.println("Test break");          //พิมพ์ข้อความ  "Test break"
for(int x=0;x<10:x++)                   //ให้วนรอบโดยกำหนดให้ x=0 ถึง 9
 {
 if(x>5)                                //ถ้ามากกว่า 5
{
break                                   //จบการทำงานใน for โดยไม่สนใจเงื่อนไข
}
 serial.print("x=");                    //พิมพ์ข้อความ "x="
serial.println(x);                      //พิมพ์ค่าของ x
delay();                                //หน่วงเวลา 50 mS
}
}
```

โปรแกรมที่ 2.34 ผลลัพธ์การทำงานตัวอย่างการใช้งานคำสั่ง break

```
Test break
x=0
x=1
x=2
x=3
x=4
x=5
Test break
x=0
```

จากตัวอย่างนี้จะเห็นว่าค่าของ X จะถูกพิมพ์ออกมาแค่0 ถึง 5 เท่านั้นเนื่องจากพอ X = 6 ก็จะทำให้โปรแกรมทำงานตามคำสั่ง break ซึ่งก็จะเกิดผลให้โปรแกรมจบการทำงานในวงรอบของคำสั่ง for เริ่มต้นทำงานในวงรอบ loop () ใหม่อีก แต่ถ้าจะลบคำสั่งbreak ทิ้งไปค่าของ x ก็จะถูกพิมพ์จนครบตามเงื่อนไขของคำสั่ง for คือ 0 ถึง 9 ตามปกติ

2.3.10 คำสั่ง continue

คำสั่ง continue จะใช้สำหรับสั่งให้โปรแกรมข้ามการทำงานของคำสั่งที่อยู่ถัดไป จำนวน 1 คำสั่ง แต่ ถ้าเขียนคำสั่งนี้ไว้ภายใต้วงรอบการทำงานของคำสั่งที่ทำงานแบบวงรอบ เช่น do, for, while จะเป็นการข้ามการทำงานไป 1 ระดับชั้น ซึ่งเป็นการจบการทำงานจากวงรอบโดยไม่ต้องรอตรวจสอบเงื่อนไข

โปรแกรมที่ 2.35 ตัวอย่างการใช้งานคำสั่ง continue

```
Viod setup ()
{
serial.begin (19200);                    //เปิดใช้พอร์ตอนุกรมด้วย baud 19200
}
viod loop ()
{
serial.println("Test continue");        //พิมพ์ข้อความ "Test continue"
for(int x=0; x<10:x++)                   //ให้วนรอบโดยกำหนดให้ x=0 ถึง 9
{
If (x>5)                                 //ถ้ามากกว่า 5
{
continue;                                //จบการทำงานใน for โดยไม่สนใจเงื่อนไข
}
serial.print ("x=");                     //พิมพ์ข้อความ "x="
serial.println(x);                       //พิมพ์ค่าของ x
delay ();                                //หน่วงเวลา 50 mS
}
}
```

โปรแกรมที่ 2.36 ผลลัพธ์การทำงานตัวอย่างการใช้งานคำสั่ง continue

```
Test continue
x=0
x=1
x=2
x=3
x=4
x=5
Test continue
x=0
.
```

จากตัวอย่างนี้จะเห็นว่าค่าของ X จะถูกพิมพ์ออกมาแค่0 ถึง 5 เท่านั้นเนื่องจากพอ X = 6 ก็จะทำให้โปรแกรมทำงานตามคำสั่ง continue ซึ่งก็จะข้ามการทำงานในวงรอบของคำสั่ง for ไปเริ่มต้นทำงานวงรอบของ loop() ใหม่อีก แต่ถ้าลบคำสั่ง continue ทิ้งไป ค่าของ x ก็จะถูกพิมพ์ครบตามเงื่อนไขของคำสั่ง for คือ 0 ถึง 9ตามปกติ

2.3.11 คำสั่ง goto

คำสั่ง goto เป็นคำสั่งสำหรับให้โปรแกรมกระโดดไปทำงานยังตำแหน่งต่างๆ ในโปรแกรมที่อ้างอิงถึงด้วยตำแหน่ง Label โดยไม่สนใจเงื่อนไข ซึ่งการกระโดดของโปรแกรมสามารถไปได้ทุกทิศทางทั้งเดินหน้าและถอยหลัง โดยมีรูปแบบคำสั่งใช้งานของคำสั่งเป็นดังนี้

โปรแกรมที่ 2.37 แสดงรูปแบบคำสั่ง goto

```
label:
.
คำสั่งอื่นๆ
.
goto label;
```

โปรแกรมที่ 2.38 แสดงรูปแบบคำสั่ง goto กรณีการกระโดดถอยหลังกลับ

```
goto label;
.
คำสั่งอื่นๆ
.
label:
```

Label หมายถึงตำแหน่ง label ที่จะใช้เป็นจุดอ้างอิงในการให้โปรแกรมกระโดดข้ามไปทำงาน โดยการตั้งชื่อ label จะยึดหลักและข้อกำหนดเช่นเดียวกันกับการตั้งชื่อตัวแปร แต่ใช้เครื่องหมายโคลอน(:)ปิดท้ายชื่อ โดยไม่ถือว่าเครื่องหมายโคลอน เป็นส่วนหนึ่งของชื่อด้วย เป็นเพียงเครื่องหมายสำหรับบอกให้ complier รับรู้เท่านั้นว่าชื่อที่ประกาศไว้นั้นเป็น Label ไม่ใช่ตัวแปร

แต่อย่างไรก็ตามถึงแม้ว่าภาษาซี จะยอมให้ใช้คำสั่ง goto ในการกระโดดข้ามไปทำงานยังตำแหน่งต่างๆของโปรแกรมได้อย่างอิสระ แต่ในการเขียนโปรแกรมเพื่อใช้งานจริงๆนั้น ควรหลีกเลี่ยงการใช้คำสั่งนี้ แต่แนะนำให้พยายามใช้คำสั่งอื่น ที่มีรูปแบบการทำงานเป็นโครงสร้างจะเหมาะสมกว่า เพราะการใช้คำสั่งgoto ในการสั่งให้โปรแกรมกระโดด กลับไป กลับมา จะทำให้โปรแกรมไม่เป็นระเบียบเท่าที่ควร โดยเฉพาะถ้ามีการสั่งใช้คำสั่ง goto เพื่อสั่งให้กระโดดหลายๆที่ในโปรแกรม และจะทำให้คนเขียนโปรแกรมเคยชินกับการเขียนโปรแกรม แบบไม่เป็นแบบแผน ไม่รู้จัก

การกำหนดเงื่อนไขเพื่อแก้ปัญหา มุ่งแต่จะหาทางออกแบบง่ายๆ ยิ่งเมื่อโปรแกรมมีจำนวนคำสั่งมากๆ ก็จะยิ่งส่งผล ให้ทำการตรวจสอบการทำงานของโปรแกรม และจุดผิดพลาดของโปรแกรมได้ยาก ซึ่งควรใช้คำสั่งนี้ในกรณีที่เกิดทางตันหาทางออกอื่นที่ดีกว่าไม่ได้หรือเกิดความจำเป็นจริงๆเท่านั้น

โปรแกรมที่ 2.39 ตัวอย่างการใช้งานคำสั่ง goto

```
Viod setup()
{
serial.begin(19200);                    //เปิดใช้พอร์ตอนุกรมด้วย baud 19200
}
viod loop()
{
serial.println("Test goto");           //พิมพ์ข้อความ "Test goto"
int=0;                                  //สร้างตัวแปรชื่อ x เป็นแบบ int
RepeatPrint:                            //ตำแหน่ง Label ReopeatPrint
serial.print("x=");
serial.println(x);
 x++
if(x>5)goto Stopprint                   //ถ้า x มากกว่า 5 ให้กระโดดไปที่ StopPrint
goto RepeatPrint;                       //กระโดดกลับไปที่ RepeatPrint
StopPrint                               //ตำแหน่ง label StopPrint
serial.println("Stopprint");            //พิมพ์ข้อความ "Stopprint"
while(1);                               //วนรอบอยู่กับที่(หยุดการทำงานของโปรแกรม)
 }
```

โปรแกรมที่ 2.40 ผลลัพธ์การทำงานตัวอย่างการใช้งานคำสั่ง goto

```
Test goto
x-0
x=1
x=2
x=3
x=4
x=5
stop print
```

จากข้างต้นเป็นการแสดงให้เห็นการใช้คำสั่ง goto ในการกระโดดไปทำงานยังตำแหน่งต่างๆในโปรแกรม ซึ่งสามารถสั่งให้โปรแกรมกระโดดข้ามไปยังตำแหน่งต่างๆในโปรแกรมได้ตามต้องการ โดยใช้ตำแหน่ง Label เป็นจุดอ้างอิงในการกระโดด

การกทำงานของโปรแกรมเริ่มจากการสั่งปิดใช้งานพอร์ตอนุกรม ด้วยค่า baud rate =19200 จากนั้นก็สั่งพิมพ์ข้อความข้อความแสดงการทำงานของโปรแกรมพร้อมกับการประกาศสร้างตัวแปร ชื่อ X โดยกำหนดค่าเริ่มต้นให้กับตัวแปร x มีค่าเท่ากับศูนย์ จากนั้นก็สั่งพิมพ์ค่าของ X ให้เห็นพร้อมกับเพอ่มค่าของ X ขึ้น 1ค่าทุกครั้งที่สั่งพิมพ์ค่าของ X เรียบร้อยแล้ว โดยจะมีการใช้คำสั่ง if ในการตรวจสอบค่าของ X ว่ามีค่ามากกว่า 5 หรือยัง ถ้า X มีค่ามากกว่า 5 ก็ให้โปรแกรมกระโดดข้ามไปทำงานยัง label ที่ชื่อ Stopprint ซึ่งคำสั่งที่อยู่หลังตำแหน่ง label stopprint คือคำสั่งที่สั่งพิมพ์ข้อความ "stop print" แล้วหยุดการทำงาน

แต่ถ้าค่าของ X ยังไม่เกิน 5 โปรแกรมก็จะถูกสั่งให้กระโดดไปทำงานยังตำแหน่ง label ที่มีชื่อ ว่า RepeatPrint ซึ่งก็จะเป็นการวนสั่งพิมพ์ค่าของ X และทำการเพิ่มค่าของ X ใหม่อีกรอบ

2.3.12 คำสั่ง return

คำสัง return จะใช้สำหรับจบการทำงานจากโปรแกรมย่อยนอกจากนี้แล้วยังสามารถใช้ในการศึกษาค่าจากโปรแกรมย่อยกลับไปยังโปรแกรมหลักอีกด้วย

โปรแกรมที่ 2.41 แสดงตัวอย่างรูปแบบคำสั่ง return

```
int checkSensor()
{
if(anlogRead(0)>400)                    //ถ้าค่าของ Analog-0มีค่าเกิน 400 แสดงว่าError
{
return1;                                //คืนค่าผลลัพธ์กลับไปยังโปรแกรมหลัก = Error
}
else
{
return0;                                //คืนค่าผลลัพธ์กลับไปยังโปรแกรมหลัก =OK
}
}
```

จากตัวอย่างการใช้งานของคำสั่ง return ในการส่งค่าผลลัพธ์จากโปรแกรมย่อยกลับมายังโปรแกรมหลัก โดยการกำหนดให้โปรแกรมย่อย ชื่อ checkSensor() ทำหน้าที่ในการตรวจสอบค่า Sensor กำหนดเงื่อนไขไว้ว่า ถ้าค่าของ Sensor มากกว่า 400 ให้ส่งค่ากลับไปยังโปรแกรมหลัก เท่ากับ 1

2.3.13 กลุ่มคำสั่งที่เกี่ยวกับ Digital I/O

คำสั่งในกลุ่มนี้เป็นกลุ่มคำสั่ง สำหรับใช้งาน PIN I/O ของ Arduino ในแบบของ digtal I/O ซึ่งปรกติแล้วในการจะกำหนดหน้าที่ใช้งานขาสัญญาณ ของไมโครคอนโทรลเลอร์ AVR ซึ่งนิยมเรียกกันว่า PinI/O นั้นเราจะต้องไปกหนดให้กับรีจิสเตอร์ต่างๆ ในตัว MCU โดยตรง เพื่อเลือกกำหนด รูปแบบการทำงานของขาสัญญาณ Pin I/O ต่างๆให้มีคุณสมบัติตามที่เราต้องการ ซึ่งกรรมวิธีและ ขั้นตอนดังกล่าวข้างต้น จะมีความยุ่งยากซับซ้อนพอสมควร และอาจจะดูเป็นเรื่องยากลำบากสำหรับผู้เริ่มต้นใช้งานไมโครคอนโทรลเลอร์ด้วยซ้ำไป แต่ใน Arduino นั้น ได้มีการสร้างฟังก์ชั่น หรือคำสั่งสำหรับช่วยลดความยุ่งยากซับซ้อนตรงนี้ไว้ให้แล้ว ทำให้ผู้ใช้สามรถสั่งกำหนดหน้าที่การทำงานและสั่งงาน Pin I/O ต่างๆของMCU ได้โดยง่าย

สำหรับ Digital I/O Pin ใน Arduino นั้น จะมีทั้งหมด 14 Pin โดย Arduino ได้กำหนดรหัสของตัวเลขซึ่งเป็นจำนวนเต็มที่มีค่าระหว่าง0ถึง13 สำหรับอ้างถึง Digital I/O ทั้ง 14 Pin แต่ตามปรกติและจะมี Digital I/O Pin จำนวน 2 Pin ซึ่งถูกสงวนไว้สำหรับใช้เป็นพอร์ตสื่อสารอนุกรม RS232 สำหรับใช้ Upload Code ของโปรแกรม ให้กับ Broad จึงเหลือ Digital I/O pin สำหรับใช้งานเป็น Digital I/O จริงๆจำนวน 12 Pin คือDigital I/O หมายเลข 2 ถึง 13 เท่านั้น โดยคำสั่งใน กลุ่มนี้จะมีอยู่ด้วยกัน 3 คำสั่ง

- viod pinMode(pin,mode)
- viod digitalWrite(pin,value)
- int digitalRead(pin)

2.4 คุณสมบัติของออกซิเจนละลายน้ำ

ออกซิเจนเป็นปัจจัยที่นับว่ามีความสำคัญต่อการดำรงชีวิตเนื่องจากสัตว์น้ำทุกชนิดเป็นต้องใช้ออกซิเจนในขบวนการต่าง ๆ ภายในร่างกาย เพื่อการเจริญเติบโต กุ้งทะเลมีความต้องการออกซิเจนที่ละลายในน้ำตั้งแต่ 5 มิลลิกรัมออกซิเจนต่อลิตร ขึ้นไป ถือว่าเป็นสภาวะที่เหมาะสมต่อการเจริญเติบโตของกุ้ง การเปลี่ยนแปลงปริมาณออกซิเจนที่ละลายน้ำในรอบวัน ปริมาณออกซิเจนที่ละลายมีแนวโน้มสูงขึ้นตั้งแต่ 8:00 นาฬิกา ไปจนถึง 15:00 นาฬิกา ซึ่งเป็นค่าสูงสุด และมีแนวโน้มลดลงตั้งแต่เวลา 18:00 นาฬิกา ไปเรื่อย ๆ จนถึง 6:00 นาฬิกา ซึ่งจะเป็นค่าต่ำสุด ปริมาณออกซิเจนละลายในน้ำบริสุทธิ์ที่อุณหภูมิของน้ำ 25 องศาเซลเซียส จะมีปริมาณออกซิเจนอิ่มตัวประมาณ 8.24 มิลลิกรัมออกซิเจนต่อลิตร แต่เมื่ออุณหภูมิน้ำเพิ่มขึ้นเป็น 30 องศาเซลเซียส จะมีปริมาณออกซิเจนอิ่มตัวในน้ำ 7.54 มิลลิกรัมออกซิเจนต่อลิตร ส่วนในน้ำทะเลที่ความเค็ม 30 ppt และอุณหภูมิ 30 องศาเซลเซียส จะมีปริมาณออกซิเจนละลายอิ่มตัวในน้ำประมาณ 6.39 มิลลิกรัมออกซิเจนต่อลิตร ถ้าอุณหภูมิของน้ำสูงขึ้นหรือค่าความเค็มเพิ่มขึ้น ปริมาณออกซิเจนละลายในน้ำจะมีค่าลดลง ปริมาณออกซิเจนละลายน้ำมีผลต่อการดำรงชีวิตของกุ้งมาก ถ้าปริมาณออกซิเจนละลายในน้ำมีค่าต่ำกว่า 5 มิลลิกรัมออกซิเจนต่อลิตร กุ้งจะมีการเจริญเติบโตช้า กุ้งทะเลจะมีการเจริญเติบโตดีถ้ามีปริมาณออกซิเจนละลายสูงกว่า 5 มิลลิกรัมต่อลิตร ปัจจัยที่มีผลต่อการเปลี่ยนแปลงของปริมารออกซิเจนละลายในน้ำ ได้แก่ แสงแดด การไหลเวียนของน้ำ แพลงก์ตอนพืชและสัตว์ พืชน้ำ ความโปร่งแสง ความลึกของบ่อ ชนิดและปริมาณจุลินทรีย์ สิ่งขับถ่ายของกุ้ง รวมทั้งปริมาณอาหารที่เหลือจากการกินของกุ้ง

2.4.1 ปัจจัยที่มีผลต่อการละลายออกซิเจน

1. ความดันอากาศ ถ้าความดันอากาศสูงออกซิเจนละลายน้ำได้มากขึ้น
2. อุณหภูมิของน้ำ ถ้าอุณหภูมิของน้ำสูงออกซิเจนละลายน้ำได้น้อยลง
3. ความเข้มข้นของเกลือในน้ำ ถ้าความเข้มข้นของเกลือในน้ำสูง ออกออกซิเจนละลายน้ำได้น้อยลง

2.4.2 การวัดปริมาณออกซิเจนที่ละลายในน้ำ

เนื่องจากออกซิเจนที่ละลายอยู่ในน้ำสามารถเปลี่ยนแปลงได้โดยง่าย การวิเคราะห์หาปริมาณออกซิเจนละลายน้ำจึงต้องทำทันทีที่เก็บตัวอย่าง การหาปริมาณออกซิเจนละลายในน้ำมีวิธีการหาที่สามารถทำได้ 2 วิธี คือ วิธีทางเคมี และการใช้เครื่องมือวัด DO meter

2.4.3 หลักการพื้นฐานของการวัดค่า DO ของหัววัด

วิธีการวัดปริมาณออกซิเจนละลายน้ำของหัววัดของเครื่อง DO meter นั้น ใช้หลักการทางไฟฟ้าเคมี(electrochemical) ซึ่งมีองค์ประกอบที่สำคัญ(เป็นองค์ประกอบที่อยู่ภายในหัววัด) คือ ขั้ว แคโทด ขั้วแอโนด อิเล็กโทรไลต์ และเยื่อหุ้มหัววัด (เป็นวัสดุที่มีความสามารถในการเลือกให้เฉพาะออกซิเจนผ่านได้)

2.4.4 ตัวตรวจวัดค่าออกซิเจนละลายในน้ำ

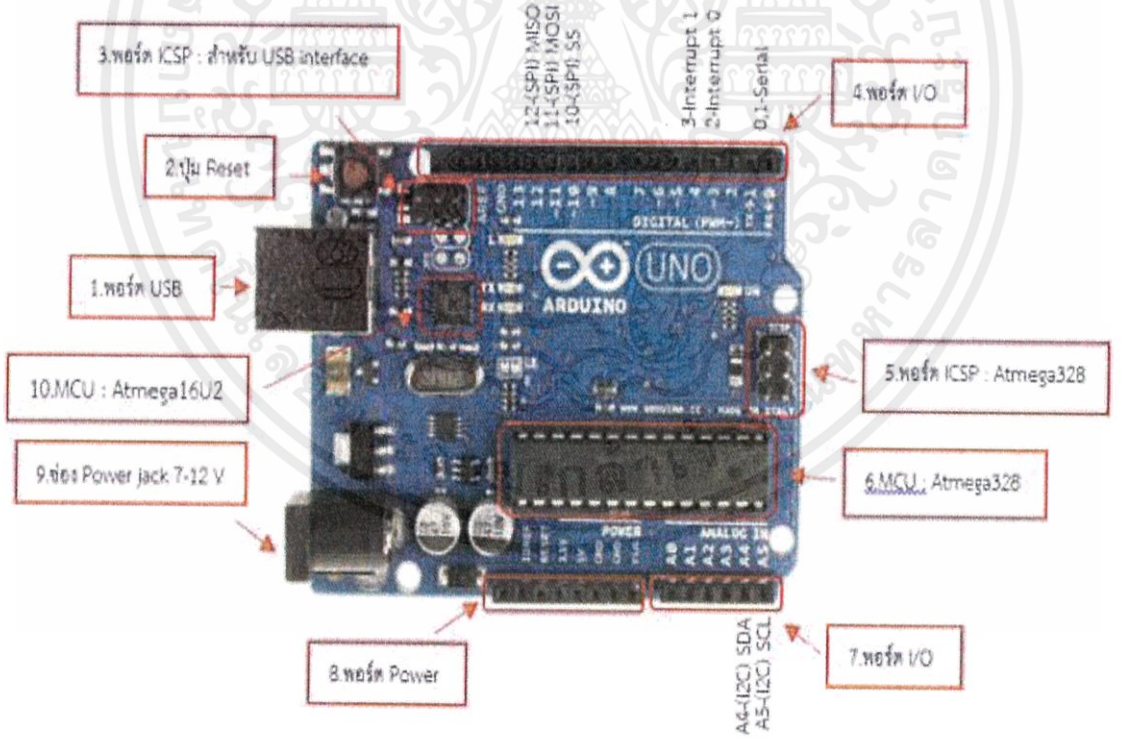
จะใช้ตัวตรวจวัดออกซิเจนในน้ำของ Atlas Scientific™ โดยจะมีหลักการ คือ วัดค่าออกซิเจนในน้ำแล้วให้ผลออกมาเป็นแรงดันไฟฟ้า โดยแรงดันเอาต์พุตที่ได้จะเป็นอนาล็อกอยู่ในช่วง 5.6 mV - 15.0 mV และเมื่อนำไปต่อวัดค่าออกซิเจนในน้ำจะมีผลตอบสนอง 90% ภายใน 10 นาที โดยสามารถทำงานได้อย่างเต็มที่ ที่อุณหภูมิ 24 -26 องศาเซลเซียส สำหรับเงื่อนไขการทำงานของเซตอุณหภูมิของน้ำจะอยู่ในช่วง 5 - 35 องศาเซลเซียส แรงดันภายในน้ำ -0.2 ถึง 1.0 kgf/cm2 gage (ที่ความลึกระดับน้ำระดับ 10 cm.) โดยการเก็บรักษาตัวตรวจวัดนี้จะต้องเก็บในแนวนอน แรงดันของการวันไฟฟ้านี้จะเปลี่ยนไปเมื่อเกิดการสะเทือนขึ้น

รูปที่ 2.1 ภาพชุดวัดออกซิเจนละลายน้ำ

2.5 บอร์ด Arduino Uno R3

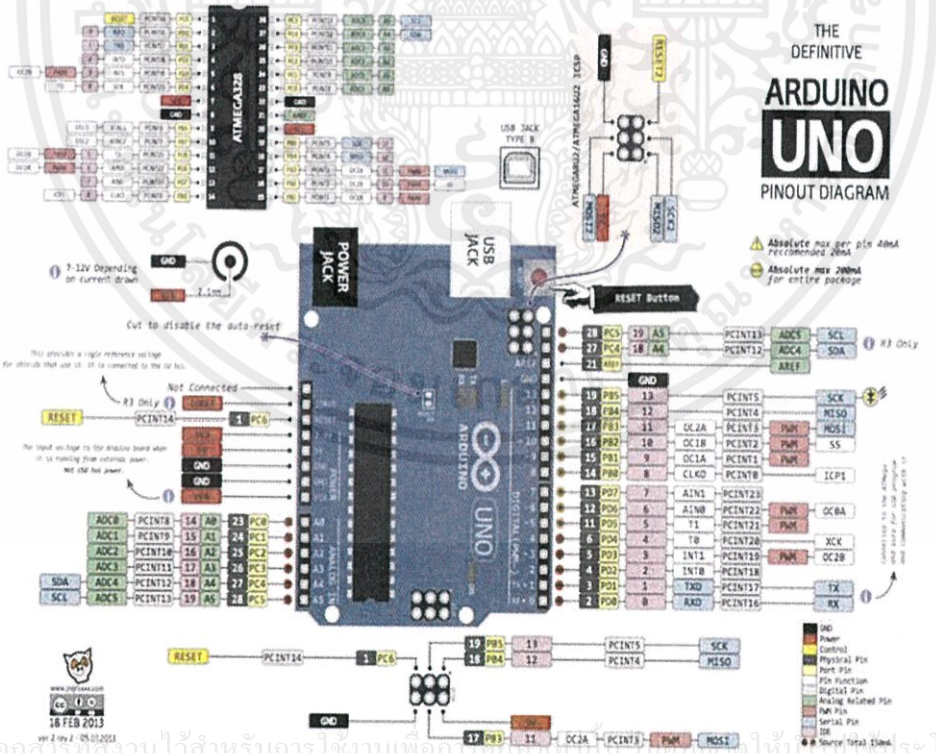
Arduino เป็นภาษาอิตาลี เป็นชื่อที่สาหรับใช้เพื่อโครงการพัฒนาไมโครคอนโทรลเลอร์ตระกูลของ AVR (Alf (Egil Bogen) and Vegard (Wollan)'sRISC processor) แบบรหัสเปิด (open source) ที่ได้รับการปรับปรุงมาจากอีกโครงการ “Wiring” ที่ใช้ ATmega128 ที่มีหน่วยความจาและ I/O ค่อนข้างมากและตัวถังของ ATmega128 เป็นอุปสรรคกับผู้เริ่มใช้งานในเบื้องต้น จึงไม่เป็นที่นิยม จากนั้นทีมงานของ Arduino ได้นามาพัฒนาต่อยอดโดยให้สามารถใช้งาน AVR ขนาดเล็กได้ จึงทาให้วงจรของบอร์ดมีขนาดที่เล็กลง อีกทั้งยังใช้อุปกรณ์จานวนน้อยอีกด้วย ทาให้ง่ายในการต่อวงจรและประหยัดในการสร้างบอร์ดมาก จึงทาให้ Arduino ได้รับความนิยมอย่างมาก

Arduino เป็นบอร์ดไมโครคอนโทรลเลอร์ที่ใช้ AVR ขนาดเล็กเป็นตัวประมวลผล เหมาะสาหรับการศึกษาเรียนรู้ไมโครคอนโทรลเลอร์ และการนาไปประยุกต์ใช้งานในการควบคุมอุปกรณ์ทั้งอินพุทและเอ้าท์พุท ต่างๆได้ โปรแกรมภาษาของ Arduino จะใช้รูปแบบของภาษา C++ โดยในการทดลองนี้จะเป็นการเริ่มต้นศึกษาการใช้งานบอร์ดไมโครคอนโทรลเลอร์ Arduino Model: Arduino UNO R3 ใช้ชิพ ATmega328 รันที่ความถี่ 16 MHz หน่วยความจาแฟลช 32 KB แรม 2 KB บอร์ดใช้ไฟเลี้ยง 7 ถึง 12 V มีระดับแรงดันไฟฟ้าในการทางานและขาสัญญาณอยู่ที่ 5 V (TTL) มี Digital Input / Output 14 ขา (เป็น PWM ได้ 6 ขา) มี Analog Input 6 ขา Serial UART 1 ชุด I2C 1 ชุด SPI 1 ชุด เขียนโปรแกรมบนซอฟท์แวร์ Arduino IDE และโปรแกรมผ่านพอร์ต USB1



รูปที่ 2.2 ภาพแสดงส่วนประกอบของบอร์ด ArduinoUNO R3

1. **USBPort:** ใช้สาหรับต่อกับ Computer เพื่ออับโหลดโปรแกรมเข้า MCU และจ่ายไฟให้กับบอร์ด
2. **Reset Button:** เป็นปุ่ม Reset ใช้กดเมื่อต้องการให้ MCU เริ่มการทางานใหม่
3. **ICSP Port** ของ Atmega16U2 เป็นพอร์ตที่ใช้โปรแกรม Visual Com port บน Atmega16U2
4. **I/OPort:**Digital I/O ตั้งแต่ขา D0 ถึง D13 นอกจากนี้ บาง Pin จะทาหน้าที่อื่นๆ เพิ่มเติมด้วย เช่น Pin0,1 เป็นขา Tx,Rx Serial, Pin3,5,6,9,10 และ 11 เป็นขา PWM
5. **ICSP Port:** Atmega328 เป็นพอร์ตที่ใช้โปรแกรม Bootloader
6. **MCU:** Atmega328 เป็น MCU ที่ใช้บนบอร์ด Arduino
7. **I/OPort:** นอกจากจะเป็น Digital I/O แล้ว ยังเปลี่ยนเป็น ช่องรับสัญญาณอนาล็อก ตั้งแต่ขา A0- A5
8. **Power Port:** ไฟเลี้ยงของบอร์ดเมื่อต้องการจ่ายไฟให้กับวงจรภายนอก ประกอบด้วยขาไฟเลี้ยง +3.3 V, +5V, GND, Vin
9. **Power Jack:** รับไฟจาก Adapter โดยที่แรงดันอยู่ระหว่าง 7-12 V
10. **MCU** ของ Atmega16U2 เป็น MCU ที่ทาหน้าที่เป็น USB to Serial โดย Atmega328 จะติดต่อกับ Computer ผ่าน Atmega16U2



รูปที่ 2.3 ภาพแสดงตำแหน่งขาตำแหน่งไมโครเคอนทรเลอร์ Arduino UNO

2.6 ทำความรู้จักกับภาษา C# เบื้องต้น

ภาษา C# (C# Programming Language) จะเขียนว่า C Sharp (ซี-ชาร์ป) ก็ได้ไม่ผิด บางทีเราอาจจะเคยได้ยินภาษาอื่นๆที่คล้าย C# เช่น ภาษา C, Java, C++ ซึ่งภาษาเหล่านี้เป็นที่มาของ C# นั่นเอง (ตัว C เป็นตัวบ่งบอกให้รู้ว่า C# มีต้นกำเนิดมาจาก C นั่นเอง) เครื่องหมาย # คือ เป็นสิ่งที่แสดงถึงความก้าวหน้ากว่า C++ ไปอีกระดับหนึ่ง (มอง # ให้เป็นเครื่องหมาย + ซ้อนกันสี่มุม อาจมองแบบ C++++ ก็ได้ไม่ผิด) ถ้าจะให้พูดกันเข้าใจง่ายๆอีกก็คือ C# ได้รวบรวมข้อดีของภาษาต่างๆเช่น Java, Delphi, C++ เข้าไว้ด้วยกัน อีกทั้งยังมีความเรียบง่ายกว่า ใครเคยใช้ Java จะรู้ว่ามีความคล้ายกับ C# มากที่สุดอีกทั้งยังมีเครื่องมือดีๆ อย่าง Visual C# 2012ของทางไมโครซอฟท์ซึ่งได้พัฒนาเป็น Version ปัจจุบันในขณะนี้ จึงลดความยุ่งยากในการโปรแกรมได้มาก สามารถพัฒนาโปรแกรมระดับสูงได้ด้วย

ภาษา C# เป็นเครื่องมือพัฒนาโปรแกรมที่อยู่ภายใต้โปรแกรม Microsoft Visual Studio ที่ทางบริษัท Microsoft ได้พัฒนาและออก Version ต่างๆดังนั้น ถ้าหากต้องการพัฒนาโปรแกรมด้วยภาษานี้ จึงต้องทำการดาวน์โหลดโปรแกรม Microsoft Visual Studio มาติดตั้งในเครื่อง ซึ่งโปรกรม Microsoft Visual Studio นั้นก็มีอยู่หลาย Version ให้เลือกใช้ และ Version ล่าสุดในปัจจุบันนี้เป็น Microsoft Visual Studio2012 ซึ่งในหนังสือเล่มนี้จะใช้ Microsoft Visual Studio2012 Express เป็นหลัก ซึ่งMicrosoft Visual Studio 2012 Express ทาง Microsoft ได้เปิดให้ดาวน์โหลดให้ทดลองใช้งานได้ฟรีแต่เครื่องมือบางอย่างอาจไม่ครบถ้วนอย่างVersion เต็มที่ต้องเสียค่าใช้จ่ายในการซื้อ License ของแท้ โปรแกรม Microsoft Visual Studio 2012 Express สามารถดาวน์โหลดได้จากเว็บไซต์ของทาง Microsoft

2.6.1 ติดตั้งโปรแกรม Microsoft Visual Studio 2012

ก่อนอื่นต้องไปดาวน์โหลดโปรแกรมMicrosoft Visual Studio 2012 Express จาก http://www.microsoft.com/visualstudio/eng/downloads (ตุลาคม 2555) ซึ่งจะมีVersion ให้เลือกใช้งาน ให้เลือกที่ Microsoft Visual Studio 2012 Express for Windows Desktop ซึ่งจะรันบน Windowsเมื่อทำการดาวน์โหลดมาเรียบร้อยแล้วจะได้ไฟล์ที่เป็นนามสกุล .iso ในที่นี้ให้ใช้โปรแกรมเขียนแผนทำการWrite ข้อมูลลงแผน เพื่อที่จะได้นำข้อมูลที่อยู่ในไฟล์ .iso ออกมาใช้งานต่อไปในขั้นตอนการติดตั้งความต้องการพื้นฐานของระบบที่จะติดตั้งโปรแกรม Microsoft Visual Studio 2012 Express

<u>ความต้องการด้านระบบปฏิบัติการ</u>

- Windows 7 (x86 หรือ x64)
- Windows 8 Consumer Preview (x86 หรือ x64) Windows 8 Release Preview
- Windows Server 2008 R2 (x64)
- Windows Server 2012 (x64)
- ความต้องการทางด้านอุปกรณ์ Hardware
- หน่วยประมวลผลกลาง (CPU) 1.6 GHz ขึ้นไป
- หน่วยความจำ RAM อย่างน้อย 1 GB

- ฮาร์ดดิสก์ขนาดความจุข้อมูลอย่างน้อย 10 GB

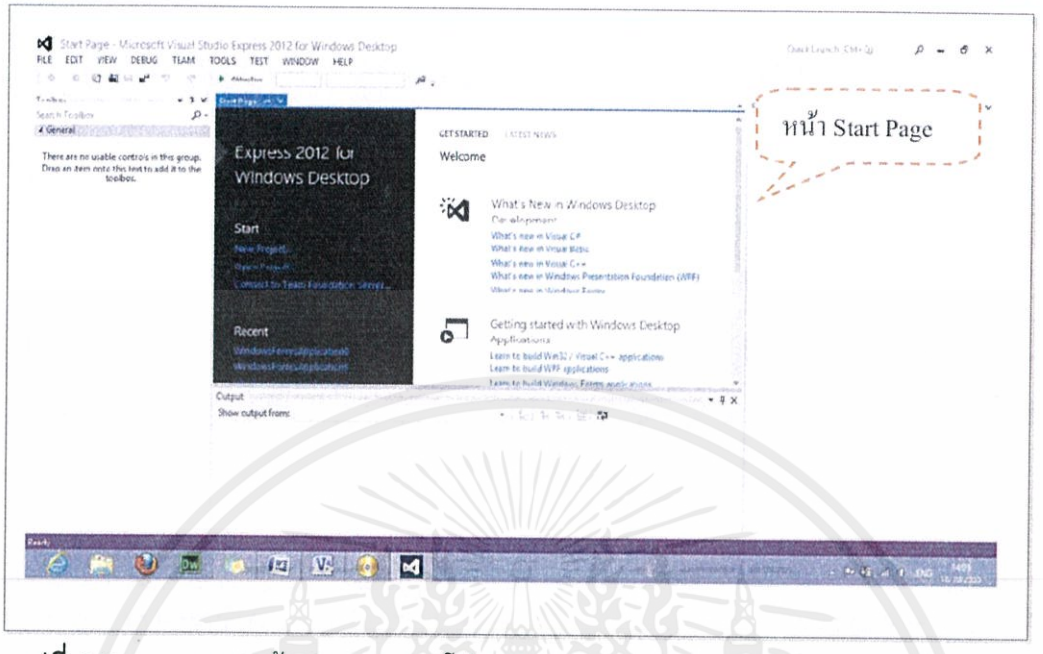
2.6.2 การใช้งานโปรแกรม Microsoft Visual C# 2012 เบื้องต้น

2.6.2.1 การเข้าสู่โปรแกรม

1). เมื่อได้ทำการติดตั้งโปรแกรม Microsoft Visual Studio 2012 Express ลงไปในเครื่องคอมพิวเตอร์เสร็จเรียบร้อยแล้วการเรียกใช้งานโปรแกรมให้เข้าไปที่ Start>AllProgram>MicrosoftVisual Studio 2012 Express>VS Express for Desktop เมื่อเลือกแล้วโปรแกรมจะแสดงหน้าต่างให้ใส่Product Key ซึ่งในที่นี้ให้กด Cancel ไปก่อนเพื่อปิดหน้าต่าง Product key ในส่วนของหน้าต่างนี้ถ้าไม่ใส่Product Key ตอนนี้ โปรแกรมจะสามารถใช้งานได้ 30 วัน หลังจากนั้นโปรแกรมจะให้ลงทะเบียนผ่านเว็บไซต์ของ Microsoft ซึ่งทาง Microsoft จะให้ชุด Product Key มาชุดหนึ่งเพื่อทำการ Activate โปรแกรมและโปรแกรมจะสามารถใช้ได้ตลอดไป แต่โปรแกรมจะยังไม่ใช้Version เต็ม เนื่องจากเป็น Version Express

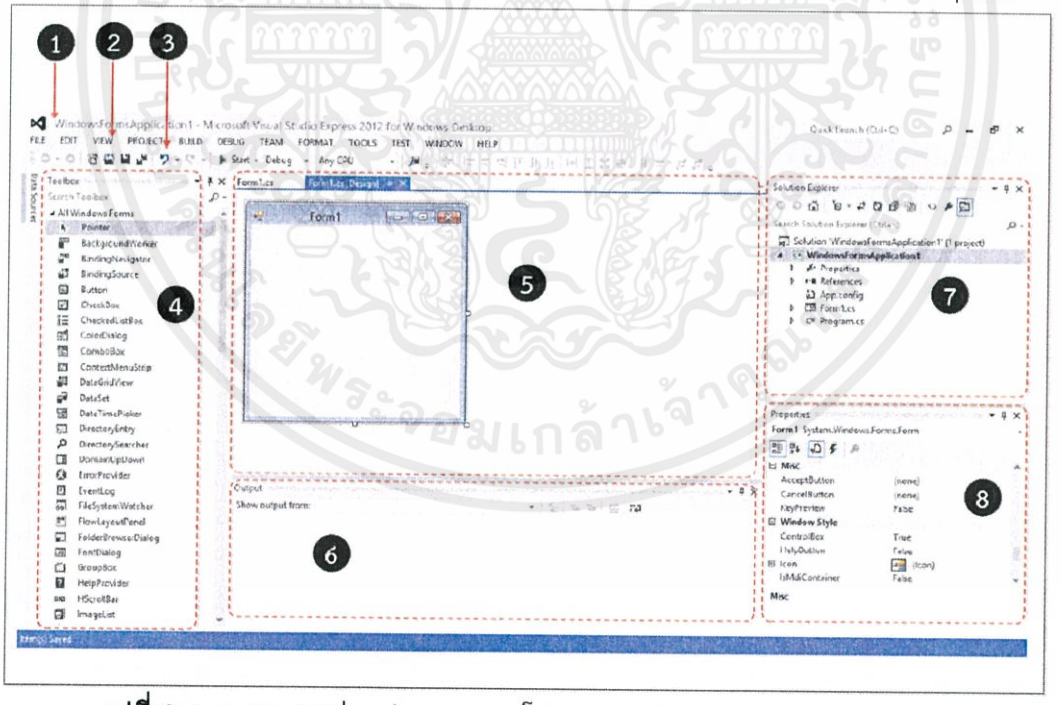
รูปที่ 2.4 ภาพแสดงหน้าต่างให้กรอก Product Key สำหรับการเปิดโปรแกรมครั้งแรก

2). ถ้าหากการรันโปรแกรมไม่ติดขัดอะไร โปรแกรม Microsoft Visual Studio 2012 Express จะแสดงหน้าหลัก Start Page สำหรับการเริ่มใช้งานโปรแกรม ดังแสดงในภาพที่ 2.4



รูปที่ 2.5 ภาพแสดงหน้า Start Page โปแกรม Microsoft Visual Studio 2012 Express

3). ส่วนประกอบหน้าต่างของโปรแกรม Microsoft Visual C#2012 Express



รูปที่ 2.6 ภาพแสดงส่วนประกอบของโปรแกรม Microsoft Visual C#2012

คำอธิบายภาพส่วนประกอบหน้าต่างของโปรแกรม Microsoft Visual C#2012 Express แสดงดังรูปที่ 2.6

หมายเลข 1 เรียกว่า Title Bar เป็นแถบแสดงชื่อเรื่องที่กำลังเปิดอยู่หรือเป็นส่วนบอกชื่อโปรแกรมและชื่อโปรเจ็กต์

หมายเลข 2 เรียกว่า Menu Bar เป็นแถบเมนูที่แสดงรายการคำสั่งต่างๆของโปรแกรม เช่น เปิดโปรเจ็กต์ สร้างโปรเจ็กต์ใหม่ การบันทึกโปรเจ็กต์ เป็นต้น

หมายเลข 3 เรียกว่า Tools Bar เป็นแถบเครื่องมือสำหรับจัดการกับ Project ที่กำลังเปิดอยู่หรือใช้เรียกแทน Menu Bar

หมายเลข 4 เรียกว่า Tool Box เป็นกลุ่มของเครื่องมือที่มี Control ต่างๆใช้ในการออกแบบวัตถุลงในส่วนที่เป็นแบบฟอร์ม ตัวอย่างเช่น Button , Text Box, Label เป็นต้น

หมายเลข 5 เป็นพื้นที่สำหรับการเขียนโปรแกรมและการออกแบบ เช่น การนำปุ่มมาวาง

หมายเลข 6 เรียกว่า Output เป็นส่วนสำหรับแสดงผลการรันโปรแกรมว่า Build โปรแกรมสำเร็จหรือไม่สำเร็จ และมีข้อผิดพลาดเท่าไหร่ และเมื่อการรันโปรแกรมพบจุดที่โค้ดผิดโปรกรมจะแสดง Error Listขึ้นมาแทนซึ่งเป็นการแจ้งเตือนผลการ Compile โปรแกรมที่ทำการพัฒนา เวลาที่เขียนโค้ดผิดว่าอยู่ที่บรรทัดใดบ้าง

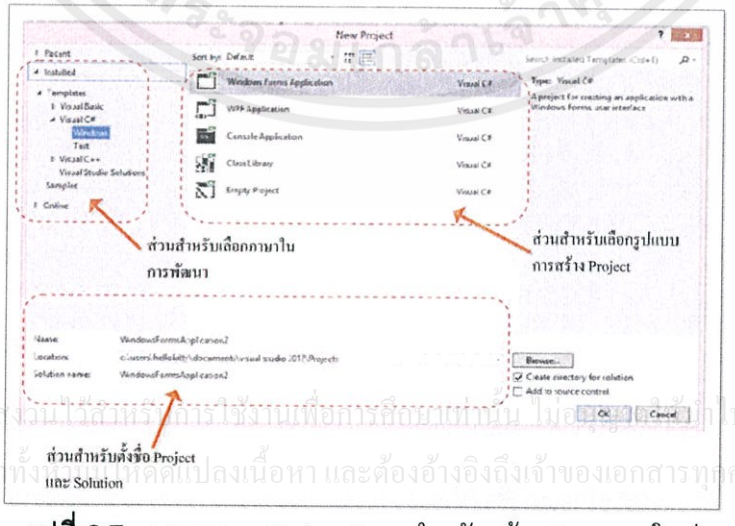
หมายเลข 7 Solution Explorer เป็นส่วนสำหรับแสดง Solution และ Project ต่างๆที่กำลังเปิดอยู่และเป็นส่วนที่ควบคุมการทำงานของหน้าจอของส่วนที่ 5

หมายเลข 8 Properties Windows เป็นส่วนสำหรับการกำหนดคุณสมบัติให้กับ Control ต่างๆที่นำมาวาง เช่น การกำหนดสีให้กับปุ่ม Button การกำหนดสีพื้นหลังให้กับ Form เป็นต้น

2.6.2.2 การสร้าง การบันทึก การ Add Project การ Add Windows Form การปิด และการเปิดProject ภาษา C#

1).การสร้าง Project ใหม่

1.1) ที่หน้า Start Page ให้คลิกเลือกที่ New Project... โปรแกรมก็จะแสดง DialogBox สำหรับการสร้าง Project ใหม่ ซึ่งในที่นี้ในส่วนทางด้านซ้ายมือของ Dialog Box จะเป็นส่วนสำหรับให้เลือกภาษาที่จะพัฒนา ให้เลือกเป็นภาษาVisual C# โดยคลิกที่รูปสามเหลี่ยมด้านหน้า Visual C# จะมีรายการย่อยแสดงออกมาแล้วเลือกที่ Windows ดังแสดงในรูปที่ 2.7



รูปที่ 2.7 ภาพแสดง Dialog Box สำหรับสร้าง Project ใหม่

1.2) หลังจากเลือกภาษา เป็น Visual C# และเลือกในส่วนของ Windows แล้วโปรแกรมก็จะแสดงรูปแบบการสร้าง Project ให้เลือก ซึ่งประกอบไปด้วย Windows Form Application,WPF Application, Console Application, Class Library และ Empty Project สำหรับโปรแกรมMicrosoft Visual Studio 2012 Version เต็มอาจจะมีมากกว่านี้ ซึ่งรูปแบบต่างๆสามารถ อธิบายได้ดังนี้

- Windows Form Application เป็นการสร้าง Project แบบ Windows Formหรือแบบ GraphicUser Interface (GUI) ซึ่งเป็นการพัฒนาแบบสร้างฟอร์มติดต่อกับผู้ใช้ ที่มีปุ่ม มี Label มี Text Box เป็นต้น

- WPF Application ย่อมาจาก Windows Presentation Foundation ซึ่งคือเทคโนโลยีหนึ่งของMicrosoft ในการสร้างซอฟท์แวร์ออกมาได้อย่างสวยงามมีระบบการทำงานคล้ายๆ Windows FormApplication แต่ว่ามีอิสระมากกว่าทำงานได้หลากหลาย ง่ายต่อการแก้ไข แต่จำกัดการใช้งานที่ .NETFramework 3.0 ขึ้นไป

- Console Application เป็นการสร้าง Project แบบ Console Application ซึ่งจะเป็นการเขียนคำสั่งภาษา C# ในลักษณะของ Class ในรูปแบบของ Text Modeและเวลารันโปรแกรมจะรันผ่าน Dos ซึ่งอาจเห็นผลการรันเป็นจอดำ

- Class Library เป็นการสร้าง Project ในลักษณะของ Class ด้วยภาษา C# สำหรับให้Projectอื่นๆสามารถมาเรียกใช้งานได้ เช่น Class สำหรับเชื่อมต่อกับ Database เป็นต้น

- Empty Project เป็นการสร้าง Project เปล่าๆ ซึ่งภายในไม่มีส่วนประกอบใดๆเลย ซึ่งเป็นProjectเปล่าๆสำหรับให้ผู้พัฒนาไปเพิ่มเอาส่วนประกอบต่างๆเอง เช่น การ Add Formเข้าสู่ Project เป็นต้น

1.3) ที่ด้านล่างของ Dialog Box จะสังเกตเห็นว่ามีช่องให้ใส่ข้อมูลต่างๆ เช่น Name,Location, Solution Name และจะมี Check Box ให้คลิกเลือก ในส่วนนี้จะเป็นการกำหนดค่าต่างๆให้กับProject ที่จะสร้างใหม่ เช่น ชื่อ Project ที่จัดเก็บ Project เป็นต้น ซึ่งสามารถอธิบายได้ดังนี้Name เหมายถึง ช่องสำหรับให้ใส่ชื่อ Project ที่จะสร้างใหม่Location หมายถึง ตำแหน่งที่จัดเก็บ Project ใหม่ที่จะสร้างขึ้นโดยค่า Default ของ MicrosoftVisual Studio 2012 Express นี้ จะ เก็บ Project ไว้ที่ Documents\visual studio 2012\Projects แต่สามารถเปลี่ยนแปลงที่จัดเก็บ Project ไปไว้ที่อื่นได้โดยคลิกที่ปุ่ม Browse... เพื่อเลือกที่จัดเก็บใหม่ Solution หมายถึงการเลือกรูปแบบการสร้าง Solution ของ Project โดยจะมีให้เลือกอยู่ 3 แบบคือ

- Create new solution หมายถึง การสร้าง Project ใหม่ และสร้าง Solution ใหม่

- Add to solution หมายถึง การสร้าง Project ใหม่ แต่ไม่ได้สร้าง Solution ใหม่แต่เป็นการสร้าง Project ใหม่โดยการเพิ่ม Project ที่กำลังสร้างเข้าไปใน Solution ที่กำลังเปิดใช้งานอยู่ในปัจจุบัน

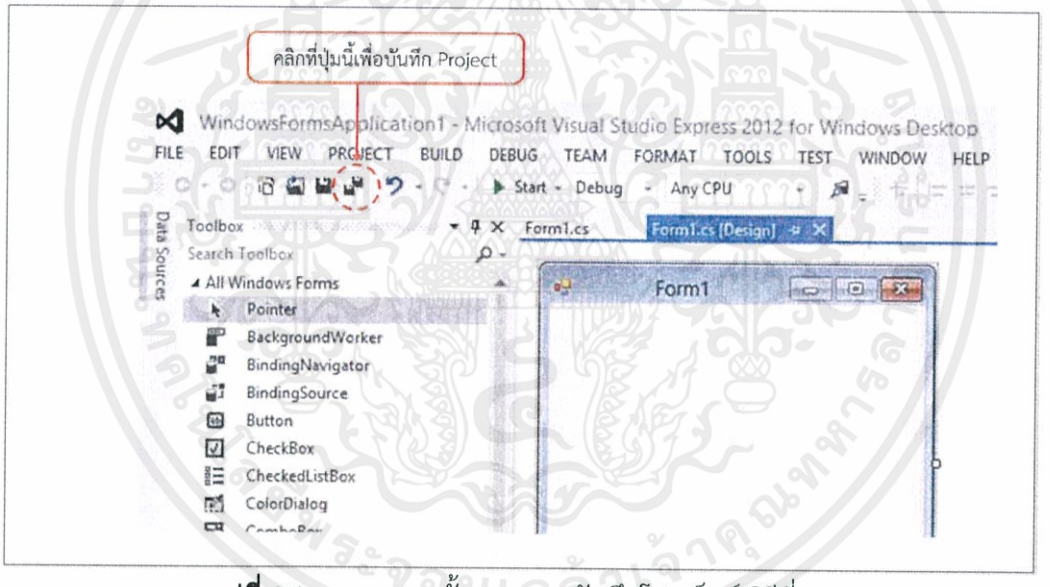
- Create in new instance หมายถึง การสร้าง Project ใหม่ และสร้าง Solution ใหม่ แต่โปรแกรมจะเปิดหน้าต่างโปรแกรม Microsoft Visual Studio หน้าต่างใหม่ขึ้นมาอีกSolution Name หมายถึง การตั้งชื่อให้กับ Solution ซึ่ง Solution จะเป็นไฟล์ๆหนึ่งที่ทำหน้าที่รวมเอา Project ไว้ด้วยกัน หรือกล่าวได้อีกนัยหนึ่งว่าไฟล์สำหรับเปิด Project ซึ่งในหนึ่ง Solution มีได้หลายProject เพื่อความเข้าใจ ยกตัวอย่าง เช่น Solution เปรียบเสมือนตู้ใบหนึ่ง และ Project ก็เปรียบเสมือนลิ้นชักต่างๆที่อยู่ภายในตู้ซึ่งทำหน้าที่เก็บสิ่งของต่างๆแตกต่างกันออกไปCreate directory for solution

เป็น Check Box สำหรับให้เลือกว่าจะสร้าง Folder สำหรับเก็บProject ไว้ภายใน Folder ที่เก็บ Solution หรือไม่ถ้าไม่คลิกที่ช่องนี้ เวลาสร้าง Project ไฟล์ต่างๆของSolution และ Project จะอยู่ในระดับเดียวกันหรือไม่ได้เก็บอย่างเป็นระเบียบ แต่ถ้าเลือก Create directory for solution จะเป็นการสร้าง Folder ย่อยให้กับ Project แต่ละตัวแยกเป็น Folder อย่างเป็นระเบียบตามชื่อ Project ที่ตั้งไว้

1.4) หลังจากเลือกรูปแบบการสร้าง Project และทำการตั้งชื่อ Projectเรียบร้อยแล้วให้คลิกที่ปุ่ม OK เพื่อทำการสร้าง Project ในที่นี้ขอยกตัวอย่างเป็น Project แบบ Windows Application จะได้ผลลัพธ์ ดังภาพที่ 2.47โดยปกติแล้ว Microsoft Visual Studio จะมีการตั้งชื่อ Default ของ Project และ Solutionเป็นWindowsFormsApplication แล้วตามด้วยลำดับของ Solution ที่เคยสร้างมาก่อน ซึ่งถ้าหากไม่เคยมีการสร้าง Project และ Solution มาก่อน ชื่อของProject และ Solution จะเป็นWindowsFormsApplication1

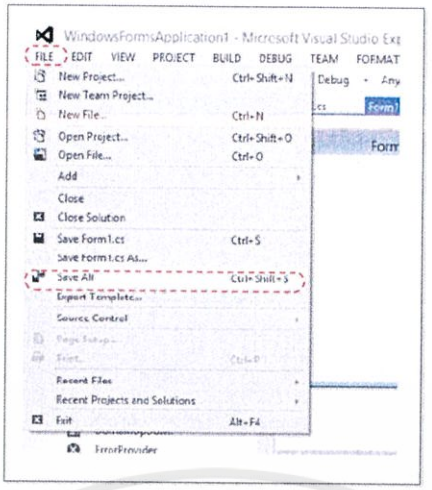
2). การบันทึกโปรเจ็กต์

2.1) การบันทึกโดยใช้เครื่องมือในแถบ Tools Bar ดังในรูปที่ 2.8



รูปที่ 2.8 ภาพแสดงขั้นตอนการบันทึกโปรเจ็กต์ วิธีที่ 1

2.2) การบันทึกโดยใช้วิธีเข้า Menu Bar ให้เข้าไปที่เมนู FILE และเลือSave All ผลลัพธ์ที่ได้ก็เหมือนกันกับวิธีที่ 1 ที่ผ่านมา ดังแสดงในรูปที่ 2.9

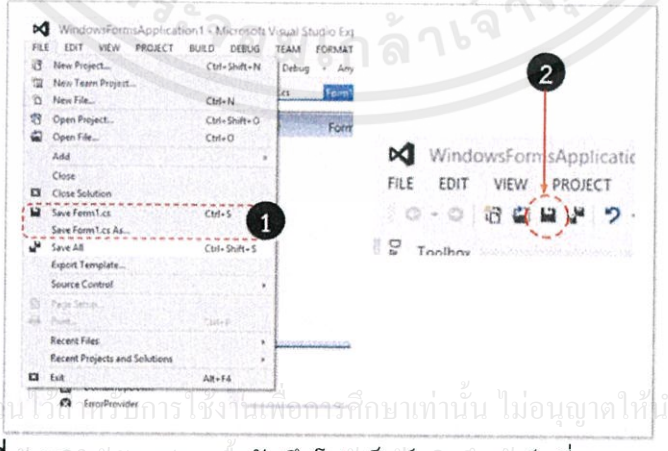


รูปที่ 2.9 ภาพแสดงวิธีการบันทึกโปรเจ็กต์ วิธีที่ 2

2.3) การบันทึกโดยวิธีการใช้คีย์ลัดที่คีย์บอร์ด โดยให้กดปุ่ม Ctrl+Shift+Sผลลัพธ์ที่ได้ก็เหมือนกันกับวิธีอื่นๆ

2.4) ใน Microsoft Visual Studio 2012 การบันทึกโปรเจ็กต์จะไม_มีDialog Box แสดงขึ้นมาให้ใส่ชื่อหรือกำหนดค่าต่างๆเหมือนกับ Microsoft Visual Studio เวอร์ชั่นก่อนๆ เพราะว่าตอนที่สร้างProject ใหม่โปรแกรมก็ได้สร้างProject และSolution ไว้ใน Path ที่เรากำหนดแล้ว ดังนั้นเมื่อกดบันทึกจึงเป็นการบันทึกการเปลี่ยนแปลงต่างๆของฟอร์ม การบันทึกทั้ง 3 วิธีที่ผ่านมาเป็นการบันทึกแบบทั้งหมดหรือSave All ซึ่งหมายถึง การบันทึกทุกอย่าง และทุกฟอร์มที่เปิดอยู่อย่างเช่น เราได้ทำการสร้างฟอร์มไว้หลายฟอร์มใน 1 โปรเจ็กต์ และทำการเปิดฟอร์มต่างๆขึ้นมาแก้ไขพร้อมกัน เมื่อทำการสั่งให้บันทึกด้วยวิธีการทั้ง 3 แบบแล้วโปรแกรมก็จะบันทึกให้ทุกฟอร์มที่เปิดแก้ไขอยู่ในขณะนั้น วิธีและขั้นตอนการ Add Form หลายๆฟอร์มใน 1 โปรเจ็กต์ สามารถศึกษาได้ในหัวข้อต่อไป

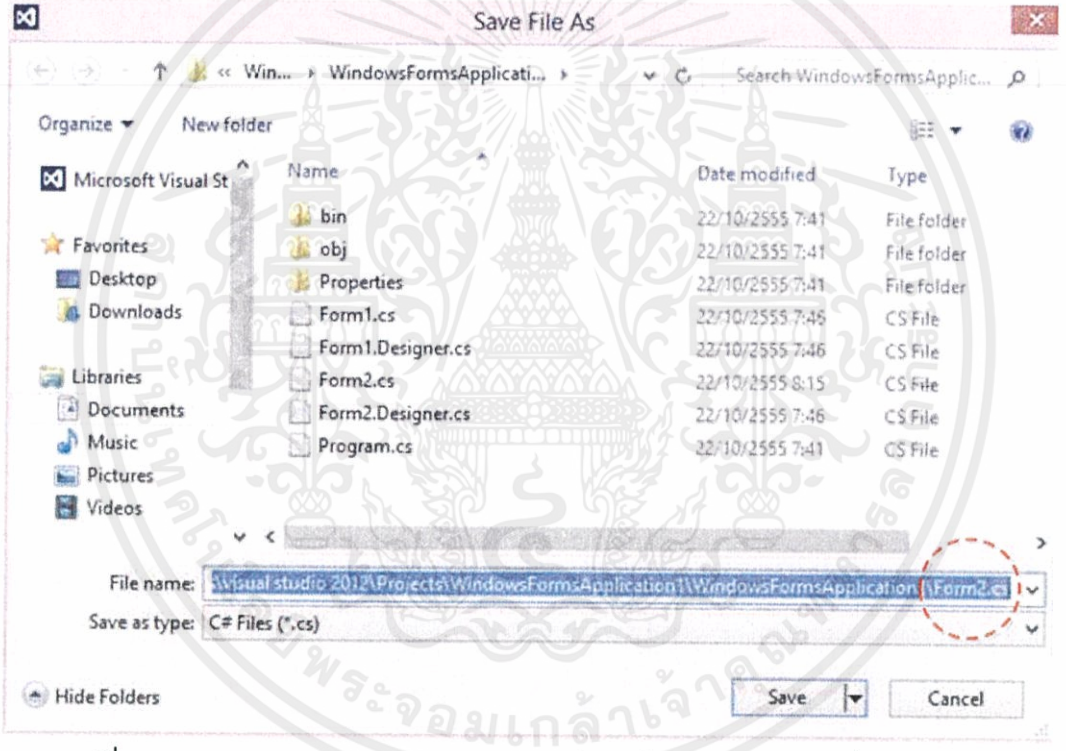
2.5) นอกเหนือจากวิธีการบันทึกทั้ง 3 วิธีดังกล่าวมาแล้ว ยังมีการบันทึกแบบ บันทึกเฉพาะฟอร์มที่กำลัง Active อยู่ในขณะนั้น วิธีการบันทึกเฉพาะฟอร์มดังแสดงในรูปที่ 2.10



รูปที่ 2.10 ภาพแสดงการบันทึกโปรเจ็กต์เฉพาะฟอร์มที่ Active อยู่

คำอธิบายการบันทึกโปรเจ็กต์เฉพาะฟอร์มที่กำลัง Active อยู่ในขณะนั้น รูปที่ 2.11 สามารถอธิบายได้ดังนี้

หมายเลข 1 เป็นการบันทึกโดยใช้วิธีการเข้าเมนู FILE ซึ่งในหมายเลขที่ 1 จะมีให้เลือกอยู่ 2 รูปแบบ คือ Save Form1.cs และ Save Form1.cs As ซึ่งทั้งสองอย่างมีข้อแตกต่างกัน สามารถอธิบายได้ดังนี้Save Form1.cs คำว่า Form1 เป็นชื่อของฟอร์มที่กำลัง Active อยู่ในขณะนั้น การบันทึกแบบนี้โปรแกรมจะบันทึกการเปลี่ยนแปลงเฉพาะ Form1 เท่านั้นถึงแม้ว่าจะเปิด Form มาแก* ไขหลายๆ Form ก็ตามSave Form1.cs As เป็นการบันทึก Form1 ในชื่อใหม่หรือเป็น Form ใหม่ ซึ่งจะเอาคุณสมบัติต่างๆที่กำหนดไว้ติดไปด้วย เมื่อเลือกการบันทึกแบบนี้แล้วโปรแกรมจะแสดง Dialog Box ขึ้นมาให้กำหนดชื่อของForm ใหม่ โดยโปรแกรม Microsoft Visual Studio จะมีการกำหนดชื่อ Default มาให้แล้วชื่อ Form2.csซึ่งเราสามารถเปลี่ยนใหม่เป็นชื่ออะไรก็ได้ ตัวอย่าง Dialog Box แสดงดังรูปที่ 2.11



รูปที่ 2.11 ภาพแสดง Dialog Box การบันทึกโปรเจ็กต์เฉพาะ Form ที่ Active อยู่

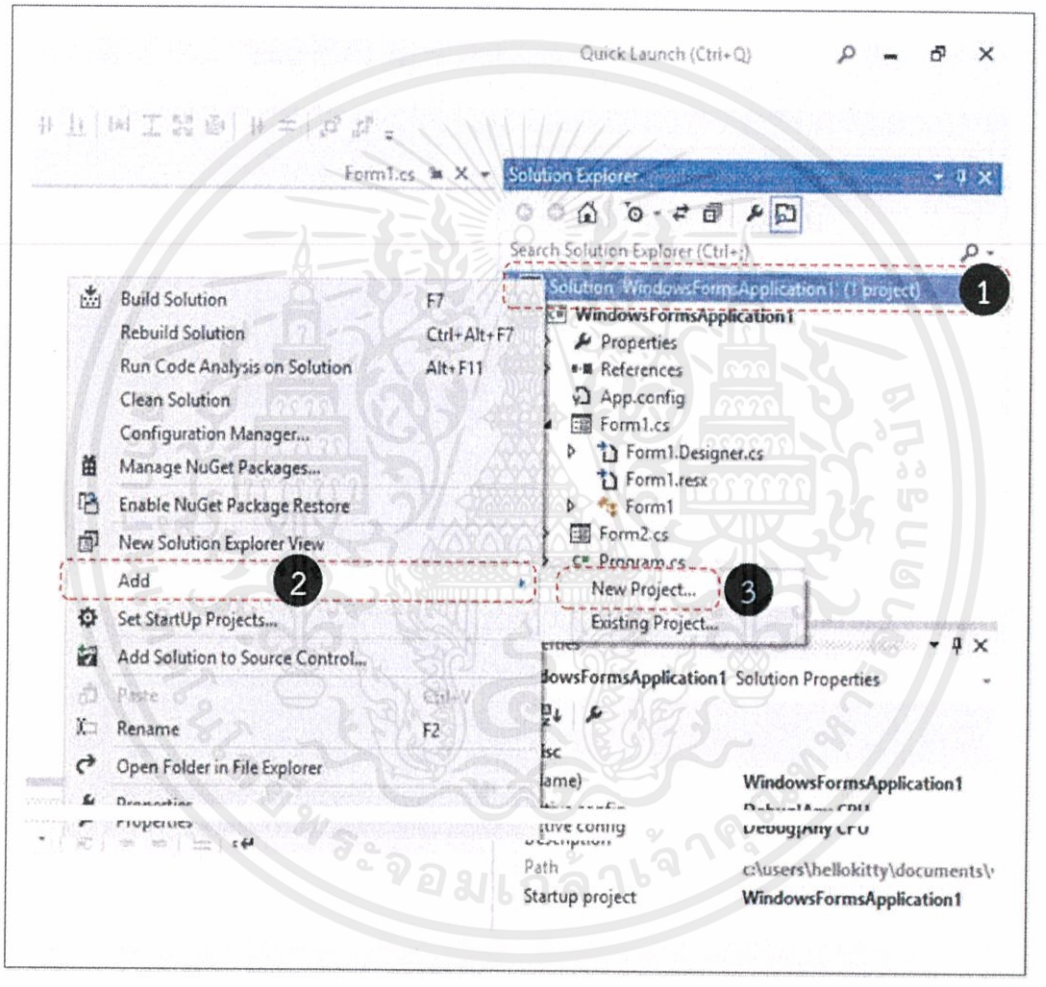
จากภาพสามารถอธิบายได้ว่า ในส่วนที่เป็นวงกลมสีแดงหมายถึงชื่อของ Form ที่จะบันทึกเป็นชื่อใหม่ในที่นี้สามารถเปลี่ยนแปลงเป็นชื่ออะไรก็ได้ และส่วนที่อยู่ด้านหน้าวงกลมสีแดงจะเป็น Path ที่สำหรับเก็บไฟล์ที่กำลังจะบันทึกว่าให้อยู่ที่ใด โดยปกติแล้วโปรแกรมจะตั้งค่า Default ให้อยู่ที่เดียวกับ Solution อยู่แล้ว แต่ก็สามารถเปลี่ยนแปลง Path ที่ใหม่เก็บได้ แต่ไม่ขอแนะนำเพราะจะทำให้การเขียนโปรแกรมไม่เป็นระเบียบ

หมายเลข 2 เป็นการบันทึกเช่นเดียวกับวิธีการของหมายเลข 1 แต่จะใช้วิธีการในส่วนของการใช้เครื่องมือในแถบ Tools Bar แทน ซึ่งผลลัพธ์ก็เหมือนกัน

3). การเพิ่มโปรเจ็กต์เข้ามายัง Solution

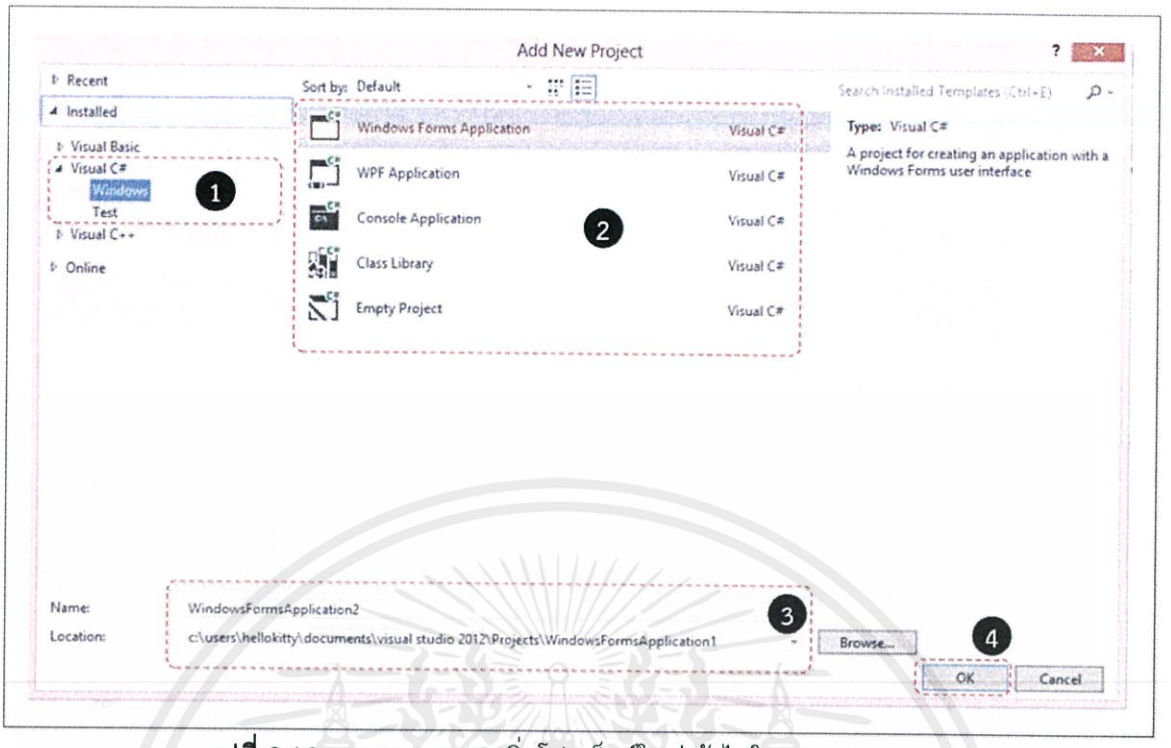
จากที่ได้เคยกล่าวมาแล้วว่า ใน 1 Solution มีได้หลายโปรเจ็กต์ ดังนั้นเพื่อความเข้าใจยิ่งขึ้นจึงขอยกตัวอย่างการเพิ่มโปรเจ็กต์ใหม่เข้ามาใน Solution ดังต่อไปนี้

3.1) ในส่วนของ Solution Explorer ให้ทำการคลิกขวาที่ Solutionสังเกตให้ดีๆ ว่าได้เลือกที่ Solution หรือที่ Project เพราะถ้าเลือกผิดจะทำให้ผลลัพธ์ไม่เหมือนกัน จะสังเกตได้ว่า ถ้าเป็น Solution จะมีคำว่า Solution อยู่ด้านหน้า เมื่อคลิกขวาที่ Solution แล้วให้เลือกที่ Add > New Project ดังแสดงในรูปที่ 2.12



รูปที่ 2.12 ภาพแสดงการเพิ่มโปรเจ็กต์ใหม่เข้ามายัง Solution

3.2) เมื่อเลือก Add New Project แล้ว จะปรากฏ Dialog Box ขึ้นมาให้เลือกประเภทของโปรเจ็กต์ซึ่งจะคล้ายๆกับการสร้างโปรเจ็กต์ใหม่ทุกประการซึ่งสามารถย้อนกลับไปดูได้ในหัวข้อก่อนหน้านี้ การเพิ่มโปรเจ็กต์เข้าไปใน Solution ดังแสดงในรูปที่ 2.12



รูปที่ 2.13 ภาพแสดงการเพิ่มโปรเจ็กต์ใหม่เข้าไปใน Solution

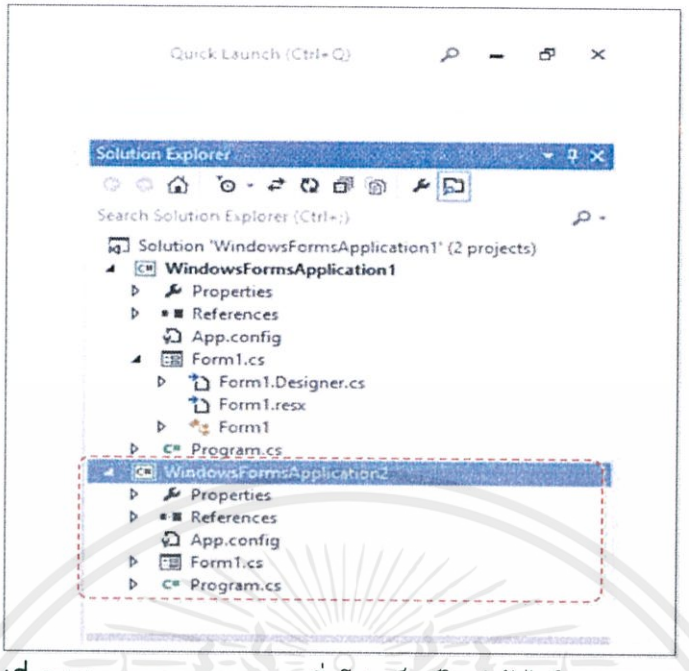
จากภาพเป็นการแสดงการเพิ่มโปรเจ็กต์ใหม่เข้าไปใน Solution สามารถอธิบายได้ดังต่อไปนี้

หมายเลข 1 เป็นการเลือกภาษาที่จะพัฒนาหรือภาษาที่จะใช้สร้างโปรเจ็กต์ ในที่นี้ให้เลือกเป็นภาษา C#

หมายเลข 2 เป็นการเลือกประเภทของโปรเจ็กต์ซึ่งสามารถดูรายละเอียดอย่างละเอียดได้ในหัวข้อการสร้างโปรเจ็กต์ใหม่ ที่ผ่านมาแล้วในหัวข้อก่อนๆ

หมายเลข 3 จะเป็นการกำหนดชื่อโปรเจ็กต์และตำแหน่งที่จัดเก็บโปรเจ็กต์ ซึ่งถ้าไม่กำหนดโปรแกรม Microsoft Visual Studio จะมีชื่อ Default ให้เป็น WindowsFormApplication 2 เนื่องจากก่อนหน้านี้เราได้สร้างโปรเจ็กต์ที่เป็น WindowsFormApplication 1 ไปแล้ว และในส่วนของตำแหน่งLocation โปรแกรม Microsoft Visual Studio จะกำหนดค่าให้โปรเจ็กต์นี้อยู่ในที่เดียวกับที่ Solution อยู่ในตอนแรกที่เลือกสร้าง
โปรเจ็กต์

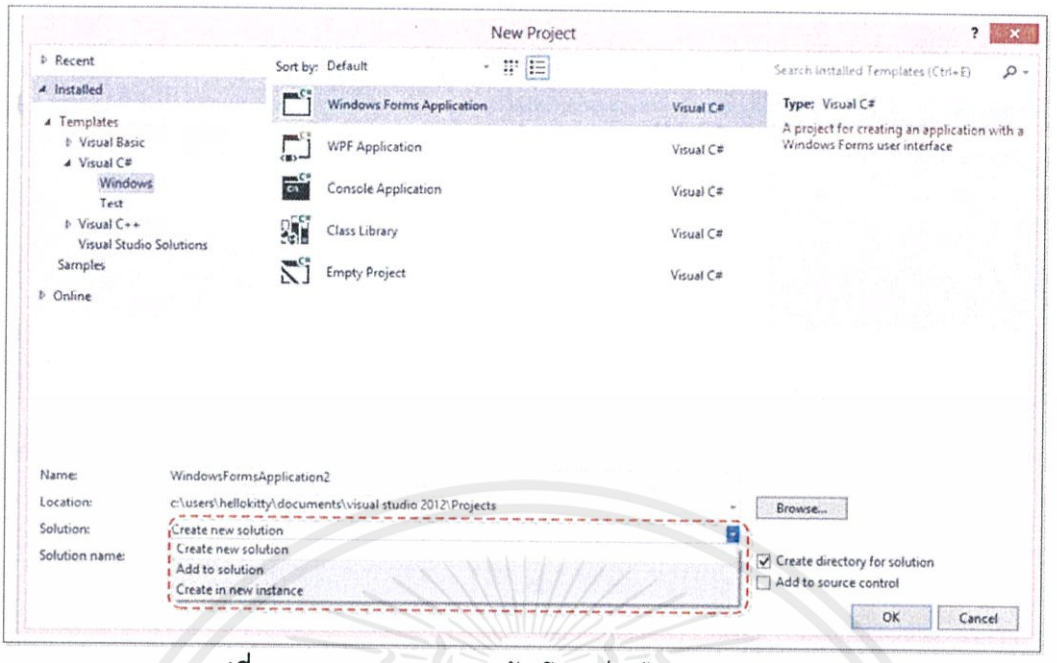
หมายเลข 4 เมื่อได้ทำการเลือกและกำหนดชื่อของโปรเจ็กต์เรียบร้อยแล้วให้กดที่ปุ่ม OK เพื่อทำการสร้างโปรเจ็กต์ใหม่เมื่อคลิกที่ปุ่ม OK แล้วจะได้โปรเจ็กต์ใหม่ที่ชื่อว่า Windows Form Application 2 ดังแสดงในรูปที่ 2.14



รูปที่ 2.14 ภาพแสดงผลการเพิ่มโปรเจ็กต์ใหม่เข้าไปใน Solution

จากภาพจะเห็นว่ามีโปรเจ็กต์เพิ่มเข้ามาใน Solution ชื่อว่า WindowsFormApplication2 ซึ่งโปรเจ็กต์นี้อยู่ภายใต้ Solution ที่ชื่อว่า Solution WindowsApplication1 ในที่นี้ชื่อของโปรเจ็กต์และชื่อของSolution ของผู้อ่านอาจจะแตกต่างออกไปจากหนังสือเล่มนี้ เพราะขึ้นอยู่กับการตั้งชื่อตอนสร้างว่าตอนสร้างได้ตั้งชื่อ Project และชื่อ Solution เป็นอะไร ในหนังสือเล่มนี้ขอยึดเอาค่า Default ที่โปรแกรม Microsoft Visual Studio ให้มาเป็นหลัก

ในการเพิ่มโปรเจ็กต์เข้ามาใน Solution อีกวิธีหนึ่งคือการใช้เมนู FILE และเลือก New Project แต่ในที่นี้ต้องสังเกตุให้ดีว่าในการสร้างโปรเจ็กต์นี้จะสร้างเป็นแบบไหน ซึ่งถ้าหากเลือกที่ New Project แล้ว กดปุ่มOK ไปเลยจะทำให้โปรแกรม Microsoft Visual Studio สร้าง Project และ Solution ใหม่ให้ด้วยเนื่องจากว่า การเลือกสร้างโปรเจ็กต์ผ่านทางเมนู FILE โปแกรม Microsoft Visual Studio ได้ตั้งค่า Default ให้สร้าง Solution ใหม่ด้วยดังนั้นเพื่อป้องกันการผิดพลาดเมื่อคลิกที่ New Project จะมี Dialog Box ให้สร้างโปรเจ็กต์ขึ้นมาให้สังเกตที่ช่อง Solution ให้ดี เพราะจะเป็นตัวกำหนดว่าเราจะทำการสร้างโปรแจ็กต์ใน Solution ใหม่ หรือการสร้างโปรเจ็กต์เข้าไปใน Solution ที่กำลังเปิดใช้งานอยู่ในขณะนั้น เพื่อให้มองเห็นภาพจะขอยกตัวอย่างดังรูปที่ 2.15



รูปที่ 2.15 ภาพแสดงการสร้างโปรเจ็กต์ใหม่ผ่านเมนู FILE

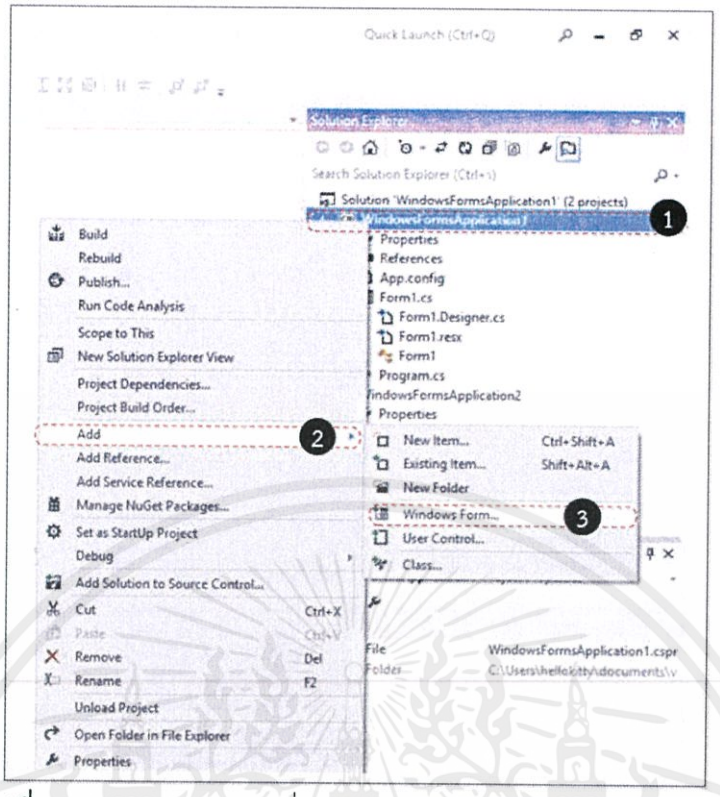
จากรูปที่ 2.15 ในส่วนที่วงสีแดงไว้คือส่วนที่สำคัญว่าจะสร้างโปรเจ็กต์ใน Solution แบบใด ซึ่งเมื่อคลิกที่สามเหลี่ยมที่มุมของช่อง Solution จะมีรูปแบบการสร้างโปรเจ็กต์ 3 รูปแบบคือ Create new solution, Add to solution และ Create in new instance ซึ่งทั้ง 3 รูปแบบมีความหมายที่แตกต่างกันสามารถอธิบายได้ดังต่อไปนี้

Create new solution หมายถึงการสรางโปรเจ็กต์ขึ้นมาใหม่พร้อมทั้งสร้าง Solution ใหม่สำหรับโปรเจ็กต์นี้ด้วย ดังนั้น ถึงแม้เราจะเปิดโปรเจ็กต์และ Solution ใดๆไว้ก็ตาม โปรเจ็กต์ใหม่ที่สร้างก็ไม่เข้าไปอยู่ใน Solution นั้นๆที่เปิดไว้Add to solution หมายถึง การสร้างโปรเจ็กต์ใหม่และเพิ่มโปรเจ็กต์ใหม่นั้นเข้าไปอยู่ภายใต้Solution ที่กำลังเปิดใช้งานอยู่ในขณะนั้นCreate in new instance หมายถึง การสร้างโปรเจ็กต์และ Solution ใหม่ในอีกหน้าต่างหนึ่งของโปรแกรม Microsoft Visual C# หรือเป็นการเปิดโปรกรม Microsoft Visual Studio ขึ้นมาอีก 1 หน้าต่าง

4). การเพิ่ม Windows Form เข้ามาในโปรเจ็กต์

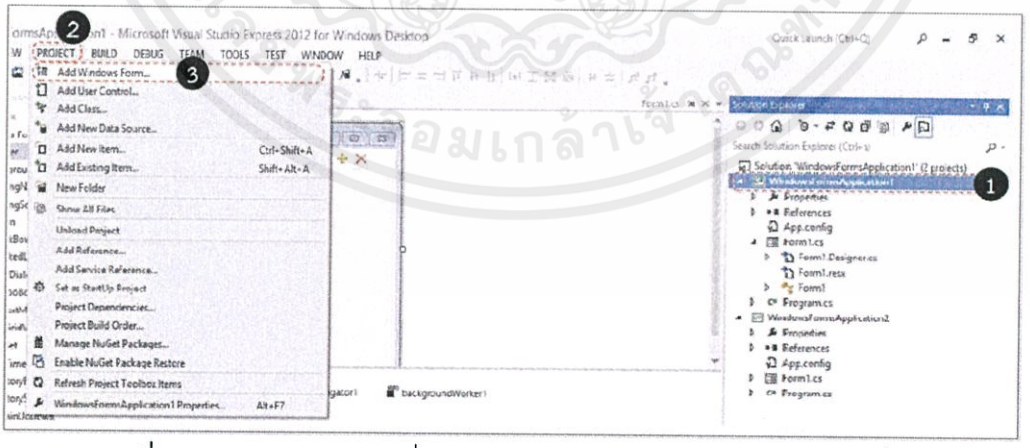
การเพิ่ม Form เข้ามาในโปรเจ็กต์คือการเพิ่มหน้าออกแบบโปรแกรมเข้ามาในโปรเจ็กต์เนื่องจากว่าเป็นไปไม่ได้ว่าในโปรแกรม 1 โปรแกรมจะมีแค่หน้าจอเดียวเพื่อติดต่อกับผู้ใช้ ดังนั้นใน 1โปรแกรมหรือ 1 โปรเจ็กต์จึงมีมากกว่า 1 Form ซึ่งวิธีการเพิ่ม Form เข้ามาในโปรเจ็กต์มีวิธีการดังนี้

4.1). ในส่วนของ Solution Explorer ให้คลิกขวาที่โปรเจ็กต์ที่ต้องการจะเพิ่ม Windows Formเข้าไป (ไม่ใช่คลิกขวาที่ Solution) เมื่อคลิกขวาที่โปรเจ็กต์แล้วให้เลือกที่ Add และเลือก Windows Form ดังแสดงในรูปที่ 2.16



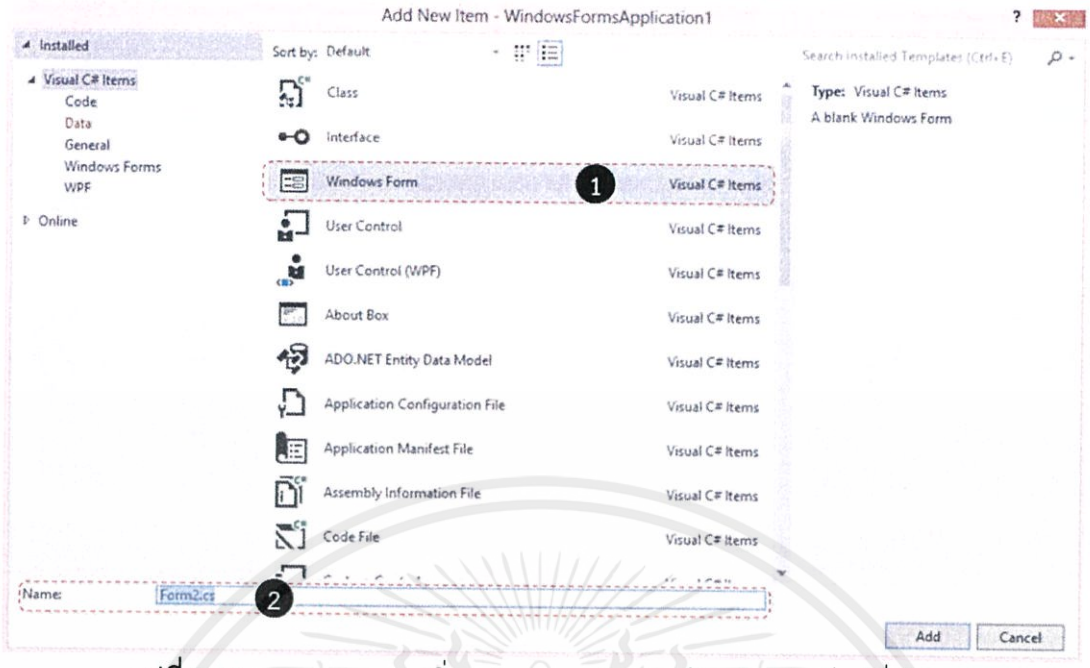
รูปที่ 2.16 ภาพแสดงการเพิ่ม Windows Form เข้ามาในโปรเจ็กต์

4.2) การเพิ่ม Windows Form เข้ามาในโปรเจ็กต์อีกวิธีหนึ่งคือ การใช้ Menu Bar ที่ชื่อว่าProject วิธีการก็คือ ในส่วนของ Solution Explorer ให้คลิกเลือกที่โปรเจ็กต์ที่ต้องการจะเพิ่ม Windows Form เข้าไป แล้วคลิกเลือกที่ Project ใน Menu Bar จากนั้นให้เลือก Windows Form ซึ่งผลลัพธ์ก็จะได้เหมือนกัน วิธีการเพิ่ม Windows Form เข้ามาในโปรเจ็กต์ดังแสดงในรูปที่ 2.17



รูปที่ 2.17 ภาพแสดงการเพิ่ม Windows Form เข้าไปในโปรเจ็กต์วิธีที่ 2

4.3) เมื่อเลือกที่ Windows Form แล้ว ก็จะปรากฏหน้าต่างสำหรับเพิ่ม Windows Form เข้าไปในโปรเจ็กต์ให้เลือกชนิดของการสร้างเป็น Windows Form แล้วตั้งชื่อของ Windows Form แล้วคลิกที่ปุ่ม Add ดังแสดงในรูปที่ 2.18



รูปที่ 2.18 ภาพแสดงการเพิ่ม windows Form เข้ามาในโปรเจ็กต์ที่เลือก

จากภาพเป็น Dialog Box ที่ได้จากการเลือกจากข้อ 1 และข้อ 2 ซึ่งจะเป็น Dialog Box ให้เลือกชนิดของการสร้าง ในส่วนของหมายเลข 1 ในที่นี้ให้เลือกเป็น Windows Form และส่วนของหมายเลข 2 ให้ตั้งชื่อของ Windows Form และหลังจากคลิกที่ปุ่ม Add เรียบร้อยแล้ว ก็จะได้ Windows Form เข้ามาในโปรเจ็กต์ที่เลือกไว้

5). การปิดโปรเจ็กต์

การปิดโปรเจ็กต์เป็นการปิดหน้าต่างการทำงานของโปรแกรมซึ่งมีด้วยกันหลายวิธีและแต่ละวิธีมีผลลัพธ์ที่แตกต่างกัน ซึ่งสามารถอธิบายได้ดังนี้

5.1) การปิดโดยใช้คำสั่งที่เมนู FILE -->Exit หรือคลิกที่ปุ่ม Close หรือรูปกากบาทสีแดงที่มุมขวาของหน้าต่างโปรแกรม Microsoft Visual Studio การปิดโดยวิธีนี้จะเป็นการปิดโปรเจ็กต์รวมทั้งปิดโปรแกรม Microsoft Visual Studio ไปด้วย

5.2) การปิดโดยใช้คำสั่งที่เมนู FILE --> Close วิธีนี้เป็นการปิดเฉพาะ Windows Form ที่กำลังเปิดใช้งานอยู่ในขณะนั้นแต่โปรแกรม Microsoft Visual Studio ยังคงเปิดใช้งานอยู่ถ้าหากว่าในขณะนั้นไม่มีการเปิดหน้าต่างออกแบบของ Windows Form ใดๆอยู่เลย จะปิดด้วยวิธีการนี้ไม่ได้

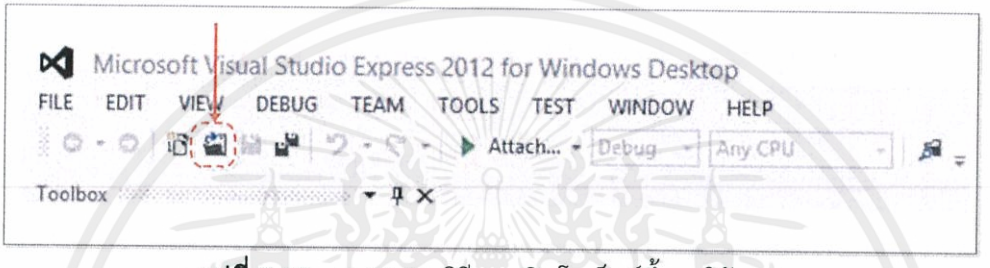
5.3) การปิดโดยใช้คำสั่งที่เมนู FILE --> Close Solution วิธีนี้เป็นการปิด Solution ที่กำลังเปิดใช้งานอยู่ในขณะนั้นแต่โปรแกรม Microsoft Visual Studio ยังคงเปิดใช้งานอยู่ เมื่อใช้คำสั่งนี้จะเห็นได้ว่าในส่วนของ Solution Explorer จะว่างเปล่าไม่มีโปรเจ็กต์หรือ Solution ใดๆแสดงอยู่อีก และถึงแม้ว่าจะมีการเปิดใช้งาน Windows Form ค้างไว้หลายๆอัน Windows Form เหล่านั้นก็จะถูกปิดไปพร้อมกับSolution ด้วย

6). การเปิดใช้งานโปรเจ็กต์ภาษา C# ที่มีอยู่แล้ว

หลังจากที่ได้เรียนรู้วิธีการสร้างโปรเจ็กต์การบันทึกโปรเจ็กต์เรียบร้อยแล้วต่อไปจะเป็นการเปิดโปรเจ็กต์ที่ได้ทำการบันทึกไว้ขึ้นมาใช้งาน ซึ่งมีอยู่ด้วยกันหลายวิธีดังต่อไปนี้

6.1) การเปิดโปรเจ็กต์โดยใช้คำสั่งที่เมนู FILE --> Open Project ซึ่งจะปรากฏ Dialog Boxขึ้นมาให้เลือกตำแหน่งที่จัดเก็บไฟล์ Project หรือ Solution ที่ได้เลือกไว้ในขั้นตอนการสร้างโปรเจ็กต์ครั้งแรก

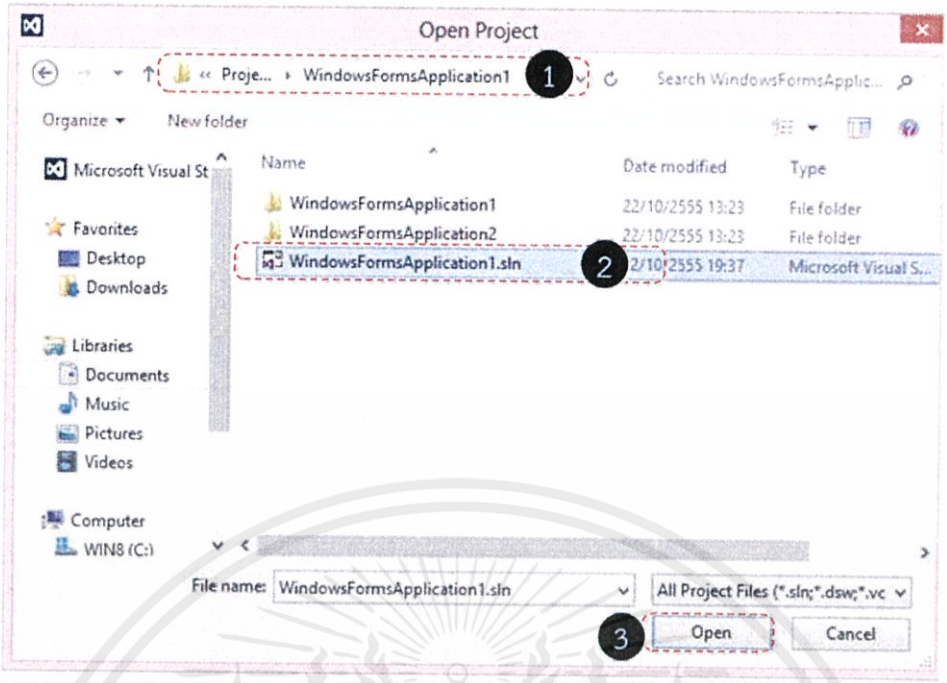
6.2) การเปิดโปรเจ็กต์โดยใช้คำสั่งที่แถบ Tools Bar ซึ่งผลลัพธ์ก็จะได้เหมือนกันกับการเปิดด้วยวิธีที่ได้กล่าวมาแล้ว วิธีการเปิดโปรเจ็กต์ด้วยวิธีนี้แสดงดังรูปที่ 2.19



รูปที่ 2.19 ภาพแสดงวิธีการเปิดโรเจ็กต์ขึ้นมาใช้งาน

6.3) การเปิดโปรเจ็กต์โดยใช้คีย์ลัดในการเปิด ให้กดปุ่ม Ctrl+O ที่คีย์บอร์ด ซึ่งผลลัพธ์ก็จะได้เหมือนกันกับวิธีการเปิดอื่นๆที่ผ่านมา

6.4) หลังจากที่เลือกวิธีการเปิดโปรเจ็กต์ วิธีใดวิธีหนึ่งแล้ว โปรแกรม Microsoft Visual Studio จะแสดง Dialog Box เพื่อให้เลือกเปิดไฟล์ Solution หรือ ไฟล์โปรเจ็กต์ที่ได้เลือกตำแหน่งที่จัดเก็บไว้ในครั้งแรกในขั้นตอนการสร้าง ซึ่ง Path หรือ ตำแหน่งที่เก็บไฟล์ Solution นั้น จะขึ้นอยู่กับว่าได้ทำการเลือกจัดเก็บไว้ที่ใดในขั้นตอนการสร้างซึ่งถ้าหากในขั้นตอนการสร้างโปรเจ็กต์ใหม่ไม่ได้ทำการเปลี่ยนค่าตำแหน่งที่จัดเก็บใดๆ โปรแกรม Microsoft Visual Studio จะจัดเก็บไฟล์ Solution และโปรเจ็กต์ไว้ที่Document\Visual Studio 2012\Project\ เป็นค่า Default ซึ่งภายในจะประกอบด้วย Folder ที่จัดเก็บSolution และโปรเจ็กต์ต่างๆ ถ้าหากได้ทำ ตามหนังสือเล่มนี้ จะพบ Folder ชื่อว่า WindowsFormApplication 1 ซึ่งเป็น Folder ที่จัดเก็บ Solution ให้ทำการเปิดเข้าไปข้างในและให้หาไฟล์ที่มีนามสกุลเป็น .sln ซึ่งเป็นนามสกุลของไฟล์ Solution ที่ใช้สำหรับเปิดโปรเจ็กต์ หลังจากนั้นให้เลือกที่ไฟล์ที่นามสกุล .sln แล้วเลือกที่ปุ่ม Open ดังแสดงวิธีการในรูปที่ 2.19



รูปที่ 2.20 ภาพแสดง Dialog Box การเปิดโปรเจ็กต์

จากภาพ ในหมายเลข 1 เป็นส่วนสำหรับให้เลือกตำแหน่งที่จัดเก็บไฟล์ Solution หมายเลข 2 ให้เลือกไฟล์ที่มีนามสกุล .sln ซึ่งเป็นนามสกุลของ Solution หลังจากนั้นคลิกที่ปุ่ม Open หมายเลข 3 และโปรแกรม Microsoft Visual Studio จะทำการเปิดโปรเจ็กต์ที่เคยสร้างไว้ขึ้นมาเพื่อดำเนินการต่อไป

6.5) การเปิดโปรเจ็กต์โดยไม่ผ่านโปรแกรม Microsoft Visual Studio ซึ่งวิธีนี้เป็นวิธีที่ง่ายที่สุด เป็นวิธีการเปิดโดยเข้าไปยังตำแหน่งที่จัดเก็บไฟล์ Solution ไว้โดยตรงผ่านทางหน้า Windows Explorer และหาไฟล์ที่มีนามสกุล .sln หลังจากนั้นให้ดับเบิ้ลคลิกที่ไฟล์นั้น ก็จะเป็นการเปิดโปรเจ็กต์เช่นกัน

2.6.3 การเรียกส่วนประกอบหน้าต่างของโปรแกรม Microsoft Visual C# 2012

ส่วนประกอบต่างๆสามารถดูได้จากหัวข้อส่วนประกอบของโปรกรม Microsoft Visual C# 2012 Express ที่ผ่านมา อีกปัญหาหนึ่งที่พบเห็นอยู่เป็นประจำในการใช้โปรแกรม Microsoft Visual C# 2012 Express คือหน้าต่างส่วนประกอบของหน้าต่างโปรแกรมไม่ครบหรือถูกปิดไปทั้งที่ตั้งใจและไม่ได้ตั้งใจ แต่ที่จริงแล้วหน้าต่างเหล่านั้นสามารถเรียกใช้ให้แสดงได้เหมือนเดิมได้ ซึ่งจะขออธิบายแยกตามแต่ละส่วนเป็น ข้อๆ ดังต่อไปนี้

2.6.3.1 การเรียกใช้หน้าต่าง Solution Explorer

หน้าต่าง Solution Explorer เป็นหน้าต่างสำหรับแสดงชื่อของ Solution และโปรเจ็กต์ต่างๆ ซึ่งเป็นส่วนที่ได้ใช้งานค่อนข้างบ่อย ดังนั้นบางทีเมาส์อาจเผลอไปคลิกโดนปุ่ม Close ที่มุมของหน้าต่าง Solution Explorer ซึ่งจะทำให้หน้าต่างส่วนนี้หายไป วิธีการเรียกให้หน้าต่างนี้กลับมาสามารถทำได้ดังนี้

1) การเรียกใช้โดยวิธีการใช้คำสั่งจากเมนู View ใน Menu Bar แล้วเลือกที่ Solution Explorer จะเห็นว่าหน้าต่าง Solution Explorer กลับมาแสดงเหมือนเดิมแล้ว

2) การเรียกใช้โดยใช้คีย์ลัดที่คีย์บอร์ด ให้กดที่ปุ่ม Ctrl+Alt+L ที่คีย์บอร์ด ก็เป็นการเรียกใช้งานหน้าต่างนี้ได้เหมือนกัน

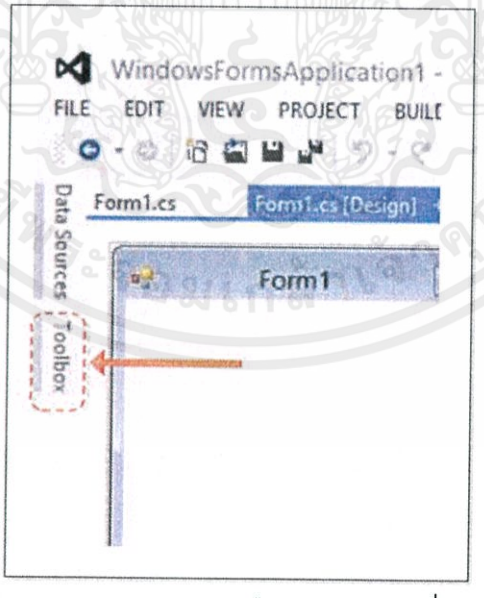
2.6.3.2 การเรียกใช้หน้าต่าง Toolbox

หน้าต่าง Toolbox เป็นหน้าต่างสำหรับเก็บกลุ่มของ Control ต่างๆ สำหรับการออกแบบและพัฒนาโปรแกรม เมื่อหน้าต่าง Toolbox หายไปสามารถเรียกใช้งานได้ดังนี้

1) การเรียกใช้โดยใช้คำสั่งที่เมนู View ใน Menu Bar แล้วเลือกที่ Toolbox จะทำให้หน้าต่าง Toolbox กลับมาแสดงเหมือนเดิม

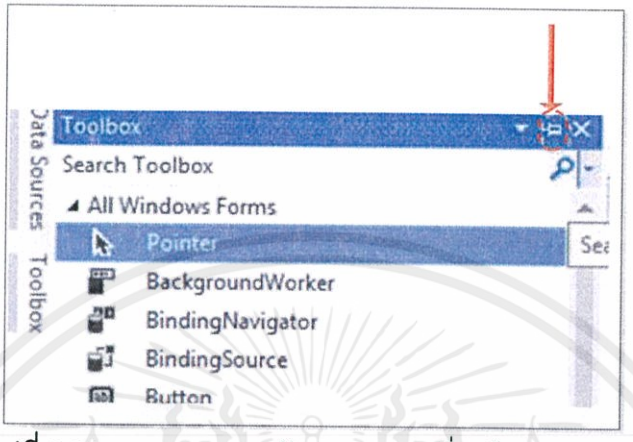
2) การเรียกใช้หน้าต่าง Toolbox โดยใช้คีย์ลัดที่คีย์บอร์ด ให้กดที่ปุ่ม Ctrl+Alt+Xที่คีย์บอร์ด ก็เป็นการเรียกใช้งานหน้าต่างเหมือนกัน

หมายเหตุ Toolbox สามารถปักหมุดให้แสดงอย่างถาวรได้ ซึ่งปกติแล้ว Toolbox จะแสดงผลในรูปแบบ Auto Hide ซึ่งจะ Slide เข้าออก ซึ่งบางทีอาจจะไม่ค่อยสะดวกนักในการดึงเอา Control ต่างๆมาใช้งาน การจัดการรูปแบบการแสดง Toolbox สามารถอธิบายได้ดังนี้



รูปที่ 2.21 ภาพแสดงการเลือก Toolbox ที่ซ่อนอยู่

จากภาพในวงกลมสีแดง Toolbox ที่มีการซ่อนกลุ่มของ Control ต่างๆไว้ภายใน เมื่อเอาเมาส์ไปคลิกแล้ว กลุ่ม Control ต่างๆก็จะแสดงออกมา แต่เมื่อนำเมาส์ไปทำงานที่ส่วนอื่นๆของโปรแกรมแล้ว กลุ่ม Control เหล่านั้นก็จะถูกซ่อนไว้อีก ในที่นี้ที่ Title Bar ของหน้าต่าง Toolbox จะมีให้ปักหมุดเอา Toolbox ไว้ ซึ่งมีวิธีการดังรูปที่ 2.22



รูปที่ 2.22 ภาพแสดงการปักหมุดแถบเครื่องมือ Toolbox

จากภาพเมื่อคลิกที่ส่วนที่วงสีแดงไว้ จะทำให้หน้าต่าง Toolbox แสดงอยู่อย่างถาวร ไม่เป็น AutoHide เหมือนเดิม แต่เมื่อคลิกซ้ำอีกครั้งก็เป็น

การตั้งให้กลับไปเป็น Auto Hide เหมือนเดิมการตั้งค่าในรูปที่ 2.22 สามารถนำ ไปลองใช้จัดการกับส่วนประกอบอื่นๆของหน้าต่างของMicrosoft Visual C# 2012 Express ได้เช่นเดียวกัน

2.6.3.3 การเรียนใช้หน้าต่าง Properties Windows

Properties Windows เป็นหน้าต่างสำหรับกำหนดคุณสมบัติให้กับ Control ต่างๆที่นำมาวางบน Windows Form และรวมไปถึงคุณสมบัติของ Windows Form ด้วย และเมื่อหน้าต่างนี้หายไปสามารถเรียกกลับมาใช้งานได้ดังนี้

1) ใช้เมนูคำสั่ง View ที่ Menu Bar แล้วเลือกที่ Properties Windows ซึ่งจะทำให้หน้าต่างProperties Windows กลับมาแสดงเหมือนเดิม

2) การเรียกใช้โดยใช้คีย์ลัดที่คีย์บอร์ด ให้กดที่ปุ่ม Alt+Enter ที่คีย์บอร์ด ก็เป็นการเรียกใช้Properties Windows ได้เช่นกัน

2.6.3.4 การเรียกใช้งานกลุ่มของ Output และ Error List

Output เป็นส่วนที่แสดงผลการรันโปรแกรมหรือ Build โปรแกรมซึ่งถ้าโปรกรมรันไม่ผ่านจะมีหน้าต่าง Error List แสดงข้อผิดพลาดออกมาว่าผิดที่บรรทัดใดบ้าง ซึ่ง Output และ Error List จะแสดงที่ตำแหน่งเดียวกัน ที่จริงแล้ว Output กับ Error List เป็นคนละหน้าต่างกัน แต่ในที่นี้เนื่องจากแสดงผลอยู่ในที่เดียวกันจึงขออธิบายในหัวข้อเดียวกัน การเรียกใช้งานสามารถทำได้ดังนี้

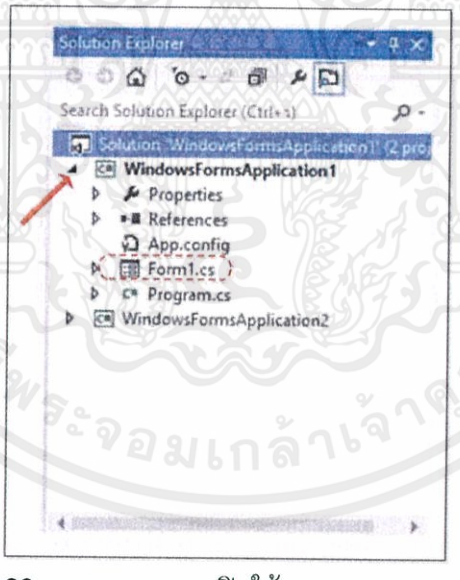
1) การเรียกใช้ Output สามารถเรียกใช้ได้จากเมนู View -->Output จาก Menu Bar

2) การเรียกใช้ Error List สามารถเรียกใช้ได้จากเมนู View -->Error List จาก Menu Bar

เกร็ดความรู้ : ในบางครั้งหน้าต่างในส่วนต่างๆที่ Microsoft Visual C# 2012 Express ไม่หายไปแต่มือพลาดเผลอไปลากโดน Title Bar ของ ส่วนประกอบต่างๆ เช่น Solution Explorer หรือ Properties Windows แล้วดึงออกมาจากตำแหน่งเดิมที่ควรอยู่ซึ่งทำให้การเขียนโปรแกรมไม่สะดวกเพราะกลุ่มของหน้าต่างเหล่านั้นได้หลุดออกจากตำแหน่งเดิมมาแสดงทับส่วนต่างๆ เช่น Windows Form หรือบางทีทับกันเอง วิธีแก้ไขให้หน้าต่างเหล่านั้นกลับเข้าที่เดิมให้เข้าไปที่เมนู Window ในส่วนของ Menu Bar แล้วเลือก Reset Window Layout ซึ่งจะทำให้ทุกอย่างจัดเรียงเหมือนเดิมกับตอนที่ลงโปรแกรมเสร็จใหม่ๆ

2.6.4 การเรียกใช้งาน Windows Form

Windows Form เป็นส่วนสำหรับการออกแบบโปแกรมและเขียนโค้ด เมื่อเราเปิดโปรเจ็กต์ที่บันทึกไว้ขึ้นมาครั้งแรกจะไม่มี Windows Form ใดๆแสดงในส่วนของการออกแบบ ดังนั้นเพื่อเปิดเอา Windows Form ขึ้นมาใช้งาน ในส่วนของ Solution Explorer ให้ทำการคลิกที่ 3 เหลี่ยมที่หน้า Solution แล้วเลือกเปิด Windows Form ที่ต้องการซึ่งจะมีนามสกุล .cs ดังแสดงในรูปที่ 2.23



รูปที่ 2.23 ภาพแสดงการเปิดใช้งาน Windows Form

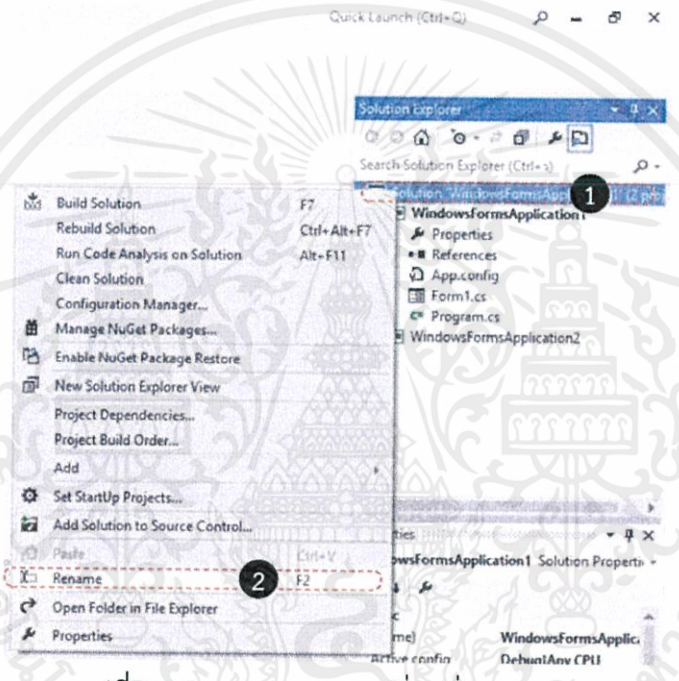
จากภาพเป็นการเปิด Form1.cs คำว่า Form1 คือชื่อของ Windows Form ซึ่งจะแตกต่างกันออกไปตามที่ได้กำหนดไว้ในตอนต้น หรือถ้าไม่ได้กำหนด โปรแกรม Microsoft Visual C# 2012 Express จะตั้งต้นให้เป็น Form1 เป็นค่า Default ไว้ซึ่งสามารถเปลี่ยนแปลงได้ทีหลัง และในส่วนของ .cs คือนามสกุลของWindows Form ที่สร้างจากภาษา C# จึงมีนามสกุลเป็น .cs ให้ทำการดับเบิลคลิกที่ไฟล์นี้ หลังจากนั้นโปแกรมก็จะทำการเปิด Windows Form ขึ้นมารวมทั้งคุณสมบัติต่างๆที่ได้กำหนดไว้ทุกประการ

2.6.5 การเปลี่ยนชื่อของ Solution , Project และ Windows Form

เพื่อให้แยกแยะระหว่าง Solution, Project, และ Windows Form จึงควรมีการตั้งชื่อให้กับองค์ประกอบเหล่านี้ เพื่อแยกแยะว่านี่คือ Solution นี่คือ Project และ Windows Form ส่วนนี้เป็นWindows Form เกี่ยวกับอะไร เป็นต้น การเปลี่ยนชื่อ Solution , Project และ Windows Form สามารถอธิบายได้ดังนี้

2.6.5.1 การเปลี่ยนชื่อ Solution

1). วิธีที่ 1 คลิกขวาที่ชื่อของ Solution แล้วเลือก Rename แล้วใส่ชื่อของ Solution ใหม่เข้าไป ดังแสดงในรูปที่ 2.24



รูปที่ 2.24 ภาพแสดงการเปลี่ยนชื่อ Solution

2) วิธีที่ 2 ถ้าไม่ถนัดใช้วิธีการคลิกขวาให้ใช้วิธีคลิกซ้ายที่ชื่อของ Solution แล้วกดที่ปุ่ม F2 ที่คีย์บอร์ด แล้วพิมพ์ชื่อของ Solution ใหม่เข้าไปก็ได้เช่นกัน

3) วิธีที่ 3 ใช้วิธีการคลิกซ้ายที่ชื่อของ Solution ที่ต้องการเปลี่ยนชื่อ เว้นระยะในการคลิกไว้สักพักหนึ่งแล้วคลิกซ้ายซ้ำลงไปอีกครั้ง โปรแกรมก็จะให้พิมพ์ชื่อของ Solution ใหม่เข้าไป

2.6.5.2 การเปลี่ยนชื่อของโปรเจ็กต์

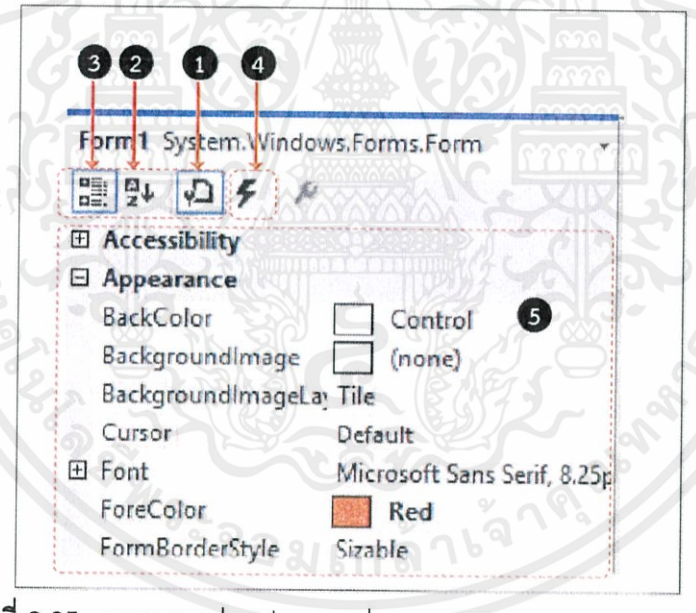
ขั้นตอนการเปลี่ยนชื่อของโปรเจ็กต์จะเหมือนกันกับขั้นตอนการเปลี่ยนชื่อของ Solution ทุกประการ เพียงแต่เปลี่ยนจากการทำงานจาก Solution มาเป็นโปรเจ็กต์แทน เช่น การคลิกที่ชื่อของโปรเจ็กต์ที่ต้องการเปลี่ยนชื่อแล้วกดปุ่ม F2 แล้วพิมพ์ชื่อของโปรเจ็กต์ใหม่เข้าไป เป็นต้น

2.6.5.3 การเปลี่ยนชื่อให้กับ Windows Form

ขั้นตอนการเปลี่ยนชื่อของ Windows Form จะเหมือนกันกับขั้นตอนการเปลี่ยนชื่อของ Solution ทุกประการ เพียงแต่เปลี่ยนจากการทำงานจาก Solution มาเป็น Windows Form แทน เช่น การคลิกที่ชื่อของ Windows Form ที่ต้องการเปลี่ยนชื่อแล้วกดปุ่ม F2 แล้วพิมพ์ชื่อของ Windows From ใหม่เข้าไป เป็นต้น แต่มีข้อแม้สำคัญ คือ ในตอนเปลี่ยนชื่อของ Windows Form ให้ระวังนามสกุลของ Windows Form เอาไว้ ซึ่งในที่นี้คือนามสกุล .cs กล่าวคือให้คงนามสกุลของ Windows From เอาไว้อย่าลบออก หรือถ้าลบออกแล้วต้องพิมพ์เข้าไปแทนให้ครบเหมือนเดิมตามรูปแบบของ Microsoft Visual C# 2012 Express กำหนดไว้

2.6.6 การใช้งาน Control พื้นฐานในโปรกรม Microsoft Visual C# 2012

ในการใช้งาน Control ต่างๆ จะต้องมีการกำหนดค่าต่างๆซึ่งเรียกว่า Properties แต่ในส่วนของProperties Windows มีหลาย Tab และหลายมุมมองในการจัดการกับ Control ดังนั้นเพื่อให้เข้าใจเป็นแนวเดียวกันจึงจะขออธิบายแต่ละ Tab ของ Properties Windows ที่สำคัญเสียก่อน ในส่วนของ Title Bar ของProperties Windows จะมีมุมมองต่างๆในการกำหนดค่าซึ่งแบ่งเป็นกลุ่มๆ ดังรูปที่ 2.25



รูปที่ 2.25 ภาพแสดงส่วนประกอบต่างๆของ Properties Windows

ส่วนประกอบต่างๆของ Properties Windows จากภาพสามารถอธิบายได้ดังต่อไปนี้

หมายเลข 1 หมายถึง Properties ซึ่งเป็นค่า Default ที่โปแกรม Microsoft Visual C# 2012 Express ตั้งให้ Focus อยู่ที่ Tab นี้อยู่แล้ว เป็น Tab สำหรับกำหนดคุณสมบัติต่างๆให้กับ Control ต่างๆ เช่น เปลี่ยนสีพื้นหลัง ใส่ข้อความ Text เปลี่ยน Font เป็นต้น ซึ่ง Properties มีมุมมองให้ปรับแสดงอยู่ 2 มุมมองใน หมายเลข 2 และ หมายเลข 3

หมายเลข 2 หมายถึง Alphabetical เป็นการจัดเรียง Properties ของหมายเลข 1 เรียงตามตัวอักษร เพื่อให้ง่ายต่อการเลือกกำหนดค่า Properties

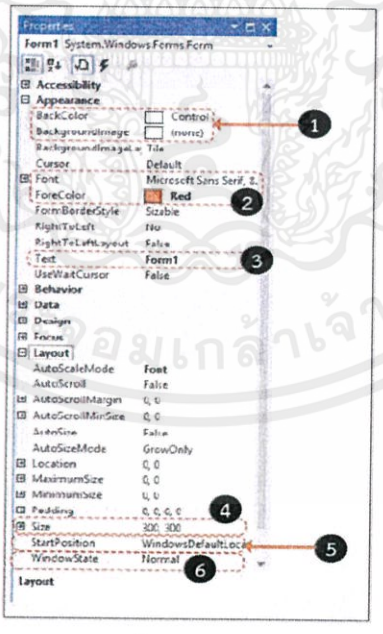
หมายเลข 3 หมายถึง Categorized เป็นการจัดเรียง Properties หมายเลข 1 แยกตามหมวดหมู่การใช้งาน

หมายเลข 4 เป็นส่วนที่ควบคุมเกี่ยวกับ Event หรือเหตุการณ์ต่างๆที่สามารถเกิดขึ้นได้กับ Control ต่างๆที่เลือกในขณะนั้น เช่น Event เมื่อ คลิกเมาส์จะให้เกิดอะไรขึ้น เมื่อกดที่ปุ่มคีย์บอร์ดจะเกิดอะไรกับ Control นั้นๆ เป็นต้น

หมายเลข 5 คือส่วนสำหรับกำหนดค่าต่างๆให้กับ Control ต่างๆเช่น กำหนดสี กำหนดขนาดของFont เป็นต้น ซึ่งรายการที่แสดงในส่วนนี้จะเปลี่ยนแปลงไปตามการเลือกหมายเลข 1 หรือหมายเลข 4 หมายเหตุ เพื่อให้เป็นไปในทางเดียวกันกับผู้เขียน หากต้องการปรับคุณสมบัติต่างๆให้กับ Control ที่เลือกให้เลือกในส่วนของหมายเลข 1 ให้ Active ไว้ หรือถ้าต้องการจัดการเกี่ยวกับ Event หรือเหตุการณ์ต่างๆ ให้เลือกในส่วนของหมายเลข 4 ให้ Active ไว้ ในที่นี้การใช้งาน Control ต่างๆ และการกำหนดProperties ต่างๆให้กับ Control ผู้เขียนจะขอยกเอาเฉพาะ Control ที่ใช้งานบ่อยๆ และ Properties ที่สำคัญเท่านั้น

2.6.6.1 ฟอร์มหรือ Windows Form

ฟอร์มเป็นพื้นที่สำหรับออกแบบและเขียนโค้ด เป็นส่วนสำหรับนำเอา Control อื่นๆมาวางโดยปกติแล้วเมื่อสร้างโปรเจ็กต์ใหม่โปรแกรม Microsoft Visual C# 2012 Express จะสร้างฟอร์มที่ชื่อForm1 มาให้เป็นค่า Default อยู่แล้ว ซึ่งฟอร์มจะมี Properties ที่สำคัญอยู่หลายส่วนด้วยกัน ซึ่งการใช้งานProperties ของฟอร์มทำได้โดยคลิกเมาส์ขวาที่พื้นที่ฟอร์มแล้วเลือก Properties หรือถ้าหน้าต่าง Properties Windows เปิดใช้งานอยู่ก็สามารถปรับแต่งได้เลยซึ่ง Properties ที่สำคัญของฟอร์มมีดังนี้



รูปที่ 2.26 ภาพแสดง Properties ของ Form ที่สำคัญ

หมายเลข 1 คือ Properties สำหรับกำหนดสีพื้นหลังของ Form ซึ่งมีอยู่ 2 รูปแบบคือ Black Color คือการใช้สีพื้นหลัง และ BackgroundImage เป็นการใส่ภาพพื้นหลังให้กับ Form การใช้ให้เลือกใช้ Properties ตัวใดตัวหนึ่ง ซึ่งหากกำหนดทั้งสองอย่างโปรแกรม Microsoft Visual C#2012 Express จะใช้ BackgroundImage เป็นค่า Default

หมายเลข 2 เป็นการกำหนด Font และสีของตัวอักษรของ Control ที่อยู่ภายใน Form นั้นๆ Font คือ การเลือกรูปแบบ Font และขนาดของตัวอักษร และ ForeColor คือการกำหนดสีให้กับ Control หรือ ตัวอักษรใดๆก็ตามที่อยู่ภายใน Form นั้น เช่น ตั้งค่า ForeColor เป็นสีแดง เมื่อลาก Label เข้ามาใน Formจะทำให้สีของตัวอักษรของ Label มีสีแดงเป็นค่า Default

หมายเลข 3 เป็นการกำหนดข้อความที่จะปรากฏบนหัวของ Form หรือ Title Bar ของ Form นิยมใส่เป็นชื่อฟอร์มหรือชื่อของโปรแกรม แต่ถ้าไม่ใส่โปรแกรม Microsoft Visual C# 2012 Express จะกำหนดเป็นลำดับของ Form ในโปรเจ็กต์นั้นๆ เป็นค่า Default เช่น Form1, Form2 เป็นต้น

หมายเลข 4 คือการกำหนดขนาดความกว้างและสูงของ Form โดยตัวเลขชุดแรกจะเป็นความกว้างและชุดที่ 2 จะเป็นความสูง โดยปกติจะมีค่า Default เป็น 300x300

หมายเลข 5 StartPosition คือการกำหนดตำแหน่งของการแสดงผลของฟอร์มเมื่อทำการรันโปรแกรมว่าจะให้แสดงที่ตำแหน่งใดของจอภาพ ในที่นี้จะมีให้ปรับอยู่หลายแบบ แต่ที่แนะนำคือให้เลือก CenterScreen ซึ่งเมื่อรันโปรแกรมแล้วฟอร์มจะแสดงที่ตรงกลางหน้าจอตลอดไม่ว่าจอภาพจะมีขนาดใดก็ตามส่วนการแสดงผลในรูปแบบอื่นๆให้ผู้อ่านลองกำหนดค่าแล้วรันดู

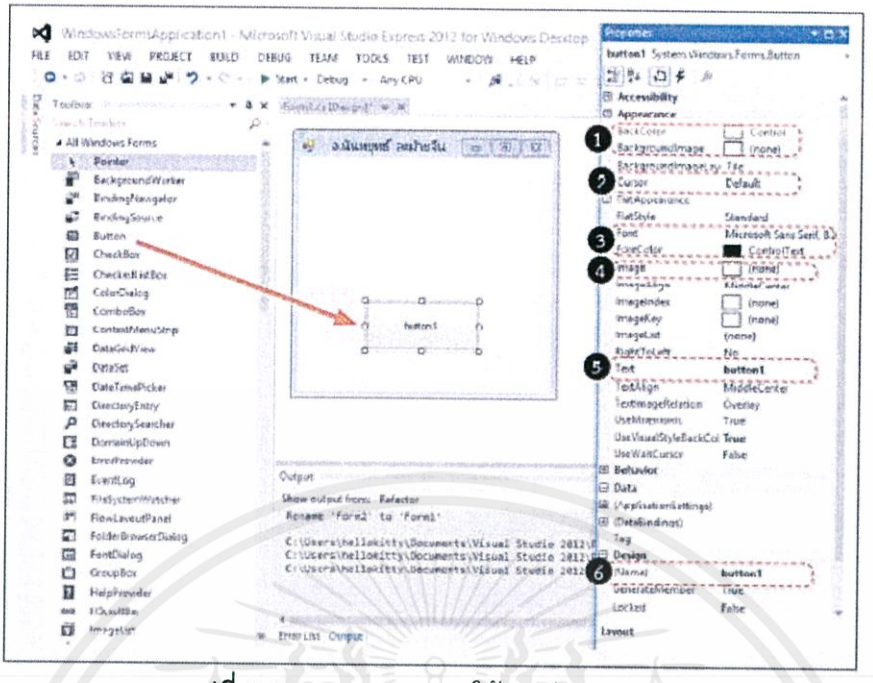
หมายเลข 6 WindowState เป็น Properties สำหรับกำหนดว่า เมื่อรันโปรเจ็กต์แล้วจะแสดงฟอร์มแบบเต็มจอหรือย่อลงไปเก็บไว้ใน Taskbar ของ Start Menu ซึ่งจะมีให้เลือกอยู่ 3 โหมดด้วยกัน คือ Normal หมายถึง แสดงผลปกติตามที่ตั้งไว้ในหมายเลข 4 และ Maximized หมายถึง เมื่อรันโปรเจ็กต์แล้วให้ขยายพื้นที่ฟอร์มให้เต็มจอ และโหมดสุดท้ายคือ Minimized หมายถึง เมื่อรันโปรเจ็กต์แล้วให้ย่อหน้าต่างไปเก็บไว้ใน Taskbar ของ Start Menu

2.6.6.2 ปุ่ม Button

ปุ่มButtonเป็นปุ่มคำสั่งสำหรับเขียนโปรแกรมควบคุมส่วนต่างๆตามวัตถุประสงค์ของโปรแกรมเมอร์ ซึ่งการใช้งานคือ ลาก Button ใน Toolbox มาวางในฟอร์ม หากต้องการเขียนโค้ดให้ทำการดับเบิ้ลคลิกที่ปุ่ม โปรแกรม Microsoft Visual C# จะแสดงหน้าสำหรับให้เขียนโค้ดเฉพาะปุ่มนั้นๆ สัญลักษณ์ของ Control Button มีลักษณะดังภาพ

รูปที่ 2.27 ภาพแสดงสัญลักษณ์ของปุ่ม Button

การใช้งาน Button ให้ลาก Button ตามสัญลักษณ์รูปที่ 2.28 มาวางบนฟอร์มในตำแหน่งที่ต้องการ ซึ่งProperties ของปุ่ม Button ที่สำคัญสามารถแสดงได้ดังภาพต่อไปนี้



รูปที่ 2.28 ภาพแสดงการใช้งานปุ่ม Button

หมายเลข 1 คือ Properties สำหรับเลือก Background ของปุ่ม Button ซึ่ง BackColor คือการใส่สีให้กับปุ่ม และ BackgroundImage เป็นการใส่รูปภาพเป็นพื้นหลังให้กับปุ่ม

หมายเลข 2 คือ Properties สำหรับเลือกลักษณะของ Cursor เมื่อชี้เมาส์ไป Active ที่ปุ่ม Button ว่าจะให้มีลักษณะเป็นอย่างไร

หมายเลข 3 Font และ ForeColor เป็นการเลือก Font และขนาดของ Font ของตัวอักษรที่แสดงในปุ่ม Button ในส่วนของ ForeColor เป็นการเลือกสีให้กับตัวอักษรที่อยู่ในปุ่ม Button

หมายเลข 4 คือ Properties สำหรับการเลือกรูปภาพประกอบปุ่ม Button เพื่อให้สวยงามและเหมาะกับหน้าที่ของปุ่ม ไม่ใช่ภาพพื้นหลังของปุ่มแต่เป็นภาพสำหรับตกแต่ง เช่น การใส่รูปภาพเครื่องหมายบวกเพื่อเป็นการแสดงว่าปุ่มนี้คือการ Add ข้อมูล เป็นต้น

หมายเลข 5 Text คือ Properties สำหรับใส่ข้อความที่ปรากฏบนปุ่ม Button

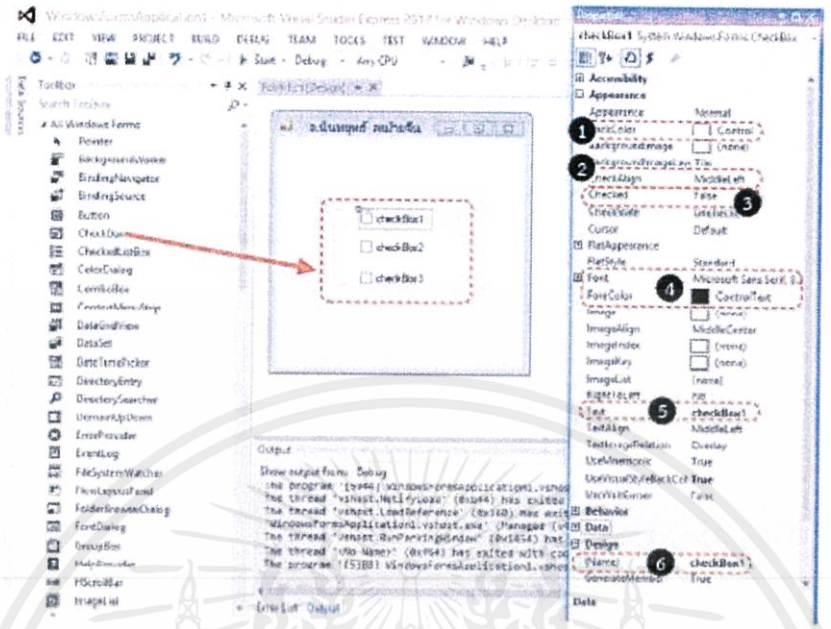
หมายเลข 6 Name คือ Properties สำหรับการตั้งชื่อให้ปุ่ม ซึ่งมีความสำคัญมาก ควรตั้งชื่อเป็นภาษาอังกฤษเพราะชื่อนี้จะนำไปอ้างอิงในการเขียนโปรแกรมสำหรับควบคุมปุ่ม

2.6.6.3 Checkbox

Checkbox คือ Control ในกลุ่มของตัวเลือกที่สามารถเลือกได้มากกว่า 1 ค่า วิธีการใช้งานคือ ลาก Control Checkbox มาวางที่ฟอร์มเหมือนปุ่ม Button หากต้องการมากกว่า 1 ตัวเลือกก็ลากมาวางหลายๆอัน ซึ่งสัญลักษณ์ของ Control Checkbox ดังแสดงในรูปที่ 2.29

รูปที่ 2.29 ภาพแสดงสัญลักษณ์ของ Control Checkbox

ลักษณะการใช้งานคือให้ลากเอา CheckBox ตามสัญลักษณ์ในรูปที่ 2.29 มาวางไว้บนฟอร์ม ซึ่งProperties ของ CheckBox และการใช้งานดังแสดงในรูปที่ 2.30



รูปที่ 2.30 ภาพแสดงการใช้งาน CheckBox

หมายเลข 1 BlackColor เป็นการกำหนดสีพื้นให้กับข้อความที่อยู่ด้านข้างของ CheckBox

หมายเลข 2 CheckAlign คือการเลือกตำแหน่งของช่อง Check ว่าให้อยู่ที่ตำแหน่งใด อยู่ซ้ายขวาตรงกลาง ด้านบน หรือด้านล่างของข้อความ เป็นต้น

หมายเลข 3 Checked คือ Properties สำหรับเลือกว่าให้ Checkbox มีค่า Default เป็น Checked
ที่ Checkbox ไว้ ซึ่งถ้าเลือก Properties นี้ให้เป็น True หมายถึง Check ช่องที่กำหนดไว้ หรือถ้าเลือกเป็น False หมายถึงไม่ได้ Checked ไว้เลย

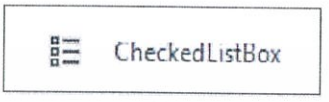
หมายเลข 4 Font และ ForeColor คือการกำหนด Font ขนาดของ Font และสีของตัวอักษรที่อยู่ด้านข้าง Checkbox

หมายเลข 5 Text คือ Properties สำหรับใส่ข้อความที่อยู่ด้านข้าง Checkbox

หมายเลข 6 Name คือชื่อของ Checkbox ชื่อในส่วนนี้จะนำไปอ้างอิงในการเขียนโปรแกรมควบคุม ดังนั้นควรตั้งให้มีความสอดคล้องกับลักษณะข้อมูลที่จะให้เลือก

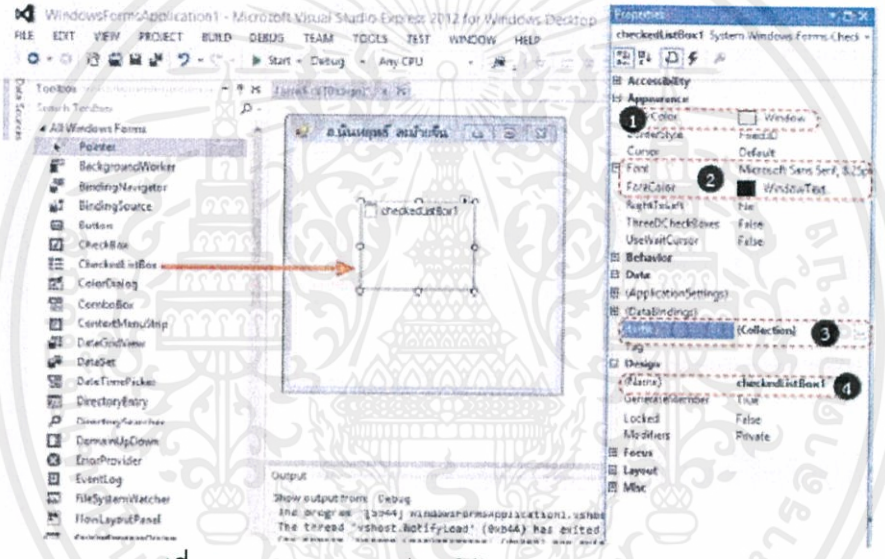
2.6.6.4 CheckedListBox

CheckedListBox คือ กลุ่มของ Control ที่สามารถเลือกได้มากกว่า 1 ค่า แต่เป็นการรวบรวมเอากลุ่มของ Checkbox เข้าไว้ด้วยกันรวมเป็น Object เพื่อให้ง่ายต่อการใช้งาน โดยปกติถ้าเราใช้Checkbox ถ้าต้องการมากกว่า 1 ตัวเลือกต้องลาก Checkbox มาวางหลายๆอัน แต่ถ้าเป็น CheckedListbox ลากมาครั้งเดียวสามารถกำหนดตัวเลือกได้ไม่จำกัดโดยไม่ต้องไปลาก Control มาใหม่ สัญลักษณ์ของ Control CheckedListBox ดังแสดงในรูปที่ 2.30



รูปที่ 2.31 ภาพแสดงสัญลักษณ์ของ Control CheckedListBox

Properties ที่สำคัญของ CheckedListBox สามารถอธิบายดังรูปที่ 2.32



รูปที่ 2.32 ภาพแสดงวิธีการใช้งาน CheckedListBox

หมายเลข 1 BackColor คือการกำหนดสีพื้นหลังกับกลุ่มของ CheckedListBox

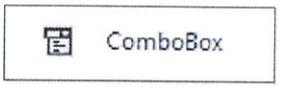
หมายเลข 2 Font และ ForeColor คือการกำหนด Font ขนาดของตัวอักษรที่มีอยู่ใน CheckedListBox และ ForeColor คือ สีของตัวอักษรที่อยู่ใน CheckedListBox

หมายเลข 3 Item คือ Properties ที่สำคัญของ CheckedListBox เป็น Properties สำหรับเพิ่มตัวเลือกเข้าไปใน CheckedListBox เมื่อคลิกเลือกที่ Properties นี้ จะมี Dialog Box พร้อมทั้งช่องให้กรอก Item หรือตัวเลือก แถวละตัวเลือก

หมายเลข 4 Name เป็น Properties สำหรับตั้งชื่อให้กับ CheckedListBox เพื่อนำไปใช้งานในการอ้างอิงในการเขียนโปรแกรมต่อไป

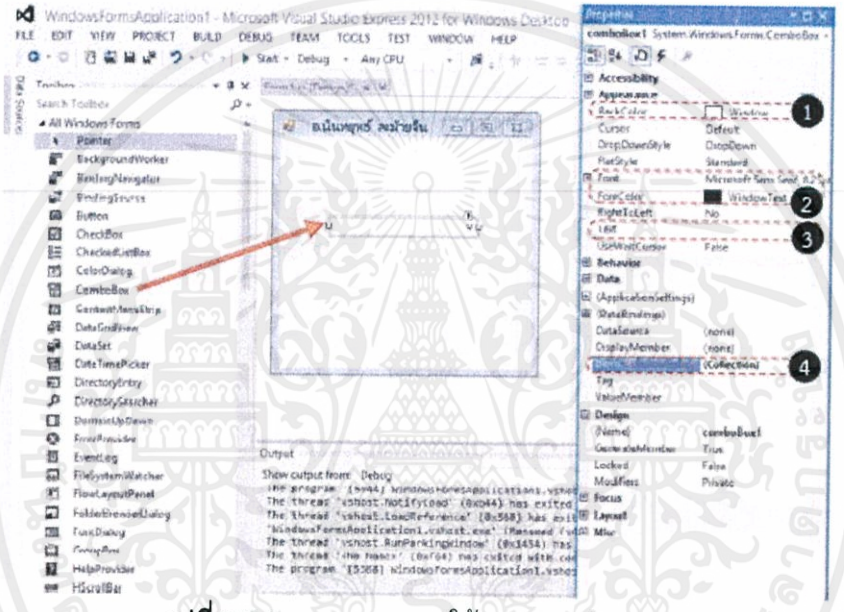
2.6.6.5 Combo box

ComboBox เป็น Control สำหรับตัวเลือกเช่นเดียวกับ CheckedListBox แต่มีข้อแตกต่างกันคือ ComboBox จะมีลักษณะเป็น List โดยมีสามเหลี่ยมเล็กๆอยู่ด้านข้างเลือกให้เลือกแสดงข้อมูลที่อยู่ในList ทั้งหมด สัญลักษณ์ของ Control ComboBox ดังแสดงในรูปที่ 2.33



รูปที่ 2.33 ภาพแสดงสัญลักษณ์ของ Control ComboBox

Properties ที่สำคัญของ ComboBox ดังแสดงในภาพต่อไปนี้



รูปที่ 2.34 ภาพแสดงการใช้งาน ComboBox

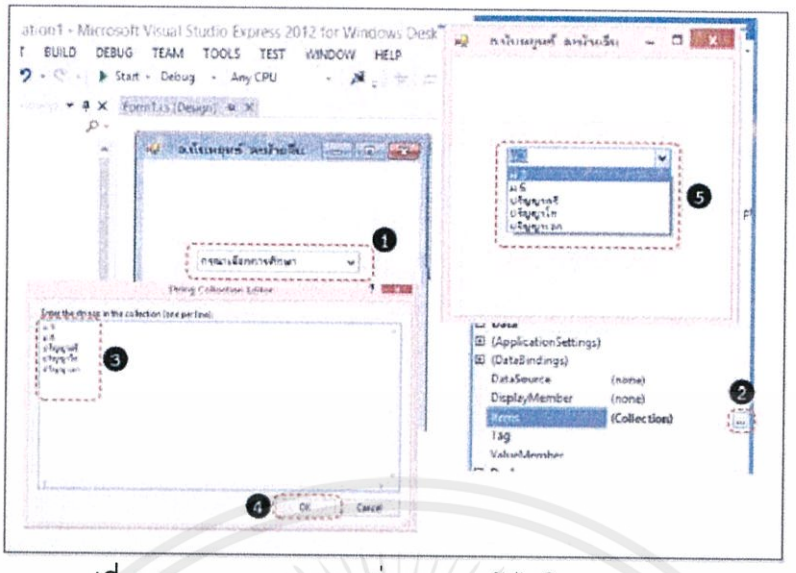
หมายเลข 1 คือ Properties สำหรับกำหนดสีพื้นหลังให้กับ ComboBox

หมายเลข 2 คือ Properties สำหรับกำหนด Font และขนาดของตัวอักษร หรือ Item ที่อยู่ภายในComboBox และส่วนของ ForeColor คือการกำหนดสีตัวอักษรให้กับ Item

หมายเลข 3 Text คือ Properties สำหรับใส่ข้อความเข้าไปใน ComboBox แต่ไม่ใช่ Item หรือตัวเลือก เป็นเพียงข้อความที่ปรากฏให้เห็นเฉยๆ

หมายเลข 4 Item คือ Properties สำหรับใส่ Item หรือตัวเลือกเข้าไปใน ComboBox วิธีการใส่Item เข้าไปใน ComboBox สามารถศึกษาได้ในหัวข้อถัดไป

1). การใส่ Item เข้าไปใน ComboBox ให้คลิกเลือกที่ Properties ที่ชื่อว่า Items เมื่อคลิกแล้วจะเห็น สัญลักษณ์จุดสามอันอยู่ภายในกล่องสี่เหลี่ยม ให้คลิกเลือก แล้วจะมี Dialog Box แสดงขึ้นมาให้ใส่ Item ต่างๆ ซึ่งการใส่ Item แต่ละอันให้เว้นด้วยการกดปุ่ม Enter เพื่อให้มองเห็นภาพ จะขอยกตัวอย่างการใช้งานดังรูปที่ 2.35



รูปที่ 2.35 ภาพแสดงการเพิ่ม Item เข้าไปใน ComboBox

จากภาพสามารถอธิบายได้ดังต่อไปนี้

หมายเลข 1 ให้เลือกที่ ComboBox ที่ต้องการจะใส่ตัวเลือกเข้าไปข้างในก่อน

หมายเลข 2 หา Properties ที่ชื่อว่า Items แล้วคลิกที่ปุ่มตามหมายเลข 2 จะปรากฏ Dialog Boxสำหรับให้ใส่ Item เข้าไป

หมายเลข 3 พิมพ์ Item ที่ต้องการ ซึ่งการกด Enter 1 ครั้ง หมายถึง 1 Item การเว้นระหว่าง Itemให้เว้นด้วยการกด Enter ที่คีย์บอร์ด ดังแสดงในหมายเลข 3

หมายเลข 4 เมื่อทำการใส่ Item เข้าไปครบแล้วกดปุ่ม OK เพื่อยืนยันการใส่ Item

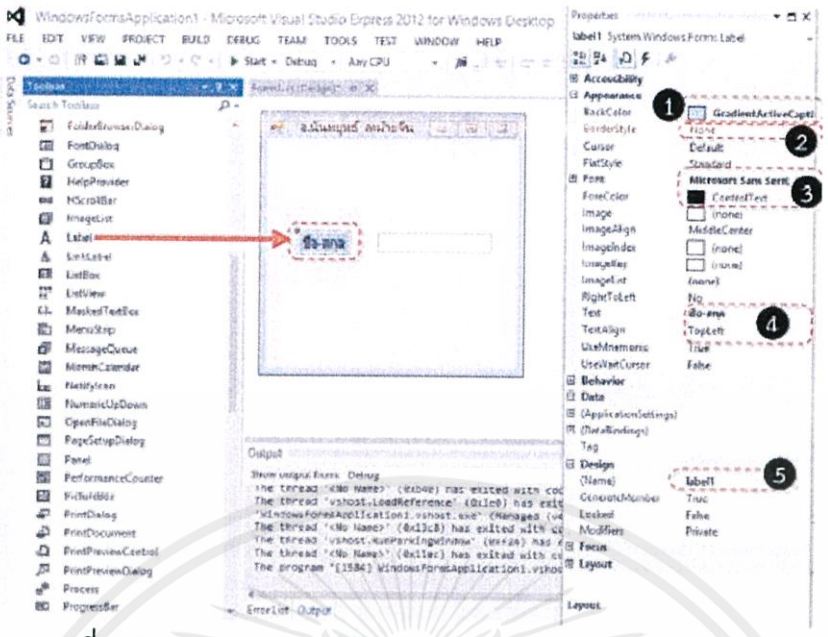
หมายเลข 5 เป็นผลลัพธ์จากการรันโปรแกรมที่ได้จากการใส่Item

2.6.6.6 Label

Label เป็นป้ายหรือคำอธิบายที่นิยมใช้ร่วมกันกัน Control อื่นๆ เพื่อบอกว่า Control นั้นๆมีหน้าที่อะไร เช่น วาง Label ไว้หน้า TextBox เพื่ออธิบายว่า TextBox นั้นมีไว้สำหรับให้ใส่ข้อมูลอะไรสัญลักษณ์ของ Control Label ดังแสดงในรูปที่ 2.36

รูปที่ 2.36 ภาพแสดงสัญลักษณ์ของ Control Label

การใช้งานและ Properties ที่สำคัญของ Label สามารถอธิบายได้ดังรูปที่ 2.37 ดังนี้



รูปที่ 2.37 ภาพแสดงการใช้ Label และการกำหนด Properties

จากภาพสามารถอธิบายได้ดังนี้

หมายเลข 1 เป็น Properties สำหรับกำหนดสีพื้นหลังของ Label

หมายเลข 2 เป็น Properties สำหรับกำหนดลักษณะของขอบให้กับ Label

หมายเลข 3 เป็น Properties สำหรับกำหนด Font และขนาดของตัวอักษรที่อยู่ภายใน Label และForeColor คือการกำหนดสีของตัวอักษรที่อยู่ภายใน Label

หมายเลข 4 Text และ TextAlign ในส่วนของ Properties Text เป็น Properties สำหรับใส่ข้อความใน Label ในส่วนของ Properties TextAlign เป็น Properties สำหรับกำหนดตำแหน่งการแสดงของตัวอักษรว่าให้ชิดซ้าย ชิดขวา เป็นต้น

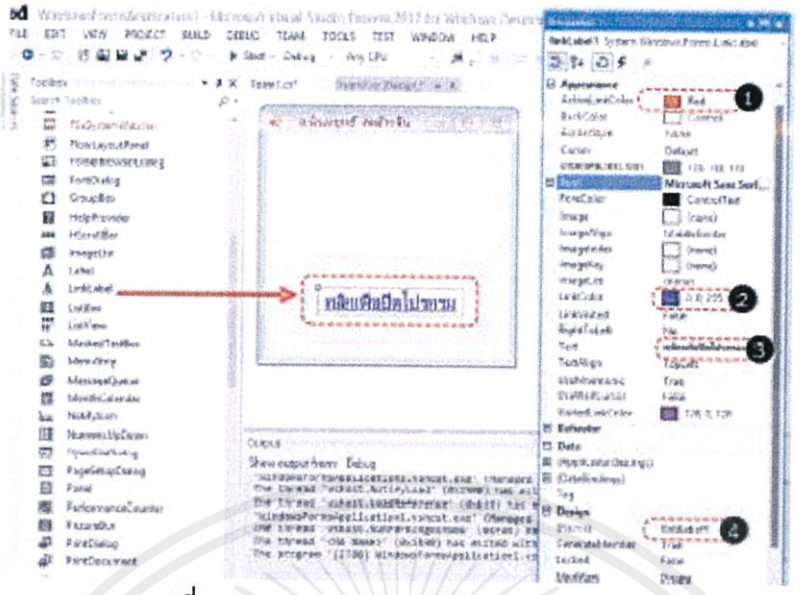
หมายเลข 5 Name เป็น Properties สำหรับใส่ชื่อของ Label เวลาเขียนโปรแกรมอ้างถึงจะต้องระบุชื่อของ Label ก่อนจึงจะสามารถใช้งานคุณสมบัติอื่นๆได้ในขั้นตอนการเขียนโปแกรม

2.6.6.7 LinkLabel

LinkLabel เป็น Control ที่คล้ายกับ Label ทุกประการ แต่มีความพิเศษคือ มีลักษณะเป็นคล้ายๆลิงค์ ซึ่งมีขีดเส้นได้ มีการกำหนดค่าเมื่อเมาส์ไปชี้ ซึ่งคล้ายๆกับลิงค์ที่พบเห็นตามเว็บไซต์ต่างๆ ซึ่งLabel แบบปกติจะเป็นแค่ป้ายธรรมดา สัญลักษณ์ของ Control LinkLabel ดังแสดงในรูปที่ 2.38

รูปที่ 2.38 ภาพแสดงสัญลักษณ์ของ Control Link Label

Properties ที่สำคัญของ LinkLabel มีดังภาพต่อไปนี้



รูปที่ 2.39 ภาพแสดงการใช้งาน LinkLabel

จากภาพสามารถอธิบายได้ดังต่อไปนี้

หมายเลข 1 ActiveLindColor เป็นสีของตัวอักษรของ LinkLabel เมื่อคลิกเมาส์

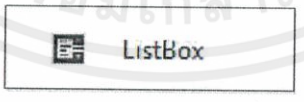
หมายเลข 2 Link Color คือสีของตัวอักษรบน LinkLabel

หมายเลข 3 Text คือ Properties สำหรับใส่ตัวอักษรที่แสดงบน LinkLabel

หมายเลข 4 Name คือชื่อของ LinkLabel ปกติแล้วโปรแกรมจะตั้งให้อัตโนมัติชื่อ linklabel1

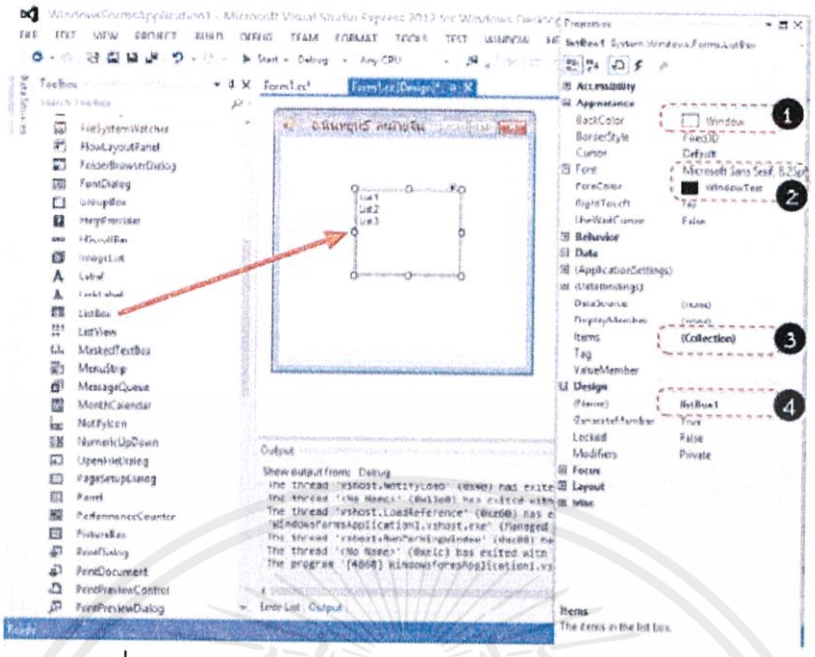
2.6.6.8 ListBox

ListBox คือ Control สำหรับเป็นตัวเลือกเช่นเดียวกับ ComboBox แต่ลักษณะการแสดงผลแตกต่างกันเท่านั้นเอง ListBox จะแสดงรายการ Item ทั้งหมดให้เห็นในกรอบสี่เหลี่ยม ซึ่งจะมี Itemให้เลือก และสามารถเพิ่ม Item ได้ไม่จำกัด สัญลักษณ์ของ Control ListBox ดังแสดงในรูปที่ 2.40



รูปที่ 2.40 ภาพแสดงสัญลักษณ์ของ ListBox

Properties ที่สำคัญและลักษณะการใช้งาน ListBox เนื่องจากมีความคล้ายกันกับ ComboBox จึงไม่ขออธิบายในส่วนของ Properties ที่เหมือนกัน ซึ่งการใช้งานดังแสดงในภาพต่อไปนี้



รูปที่ 2.41 ภาพแสดง Properties และการใช้งาน ListBox

จากภาพสามารถอธิบายได้ดังต่อไปนี้

หมายเลข 1 BackColor เป็น Properties สำหรับการกำหนดสีพื้นหลังให้กับ ListBox

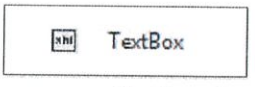
หมายเลข 2 Font และ ForeColor เป็น Properties สำหรับกำหนด Font และขนาดของตัวอักษรให้กับข้อความที่อยู่ภายใน ListBox และ Properties ForeColor เป็น Properties สำหรับการกำหนดสีของตัวอักษรที่อยู่ใน ListBox

หมายเลข 3 Item คือ Properties สำหรับใส่ตัวเลือก หรือ Item เข้าไปใน ListBox ซึ่งขั้นตอนการใส่คล้ายๆกับการใส่ Item ใน ComboBox ดังนั้นสามารถย้อนกลับไปดูในหัวข้อการใส่ Item ให้กับ ComboBox ที่ผ่านมา

หมายเลข 4 เป็นชื่อของ ListBox โดยโปรแกรมจะตั้งให้โดยอัตโนมัติชื่อว่า ListBox1 ถ้าหากภายในForm นั้นๆยังไม่เคยมี ListBox มาก่อน

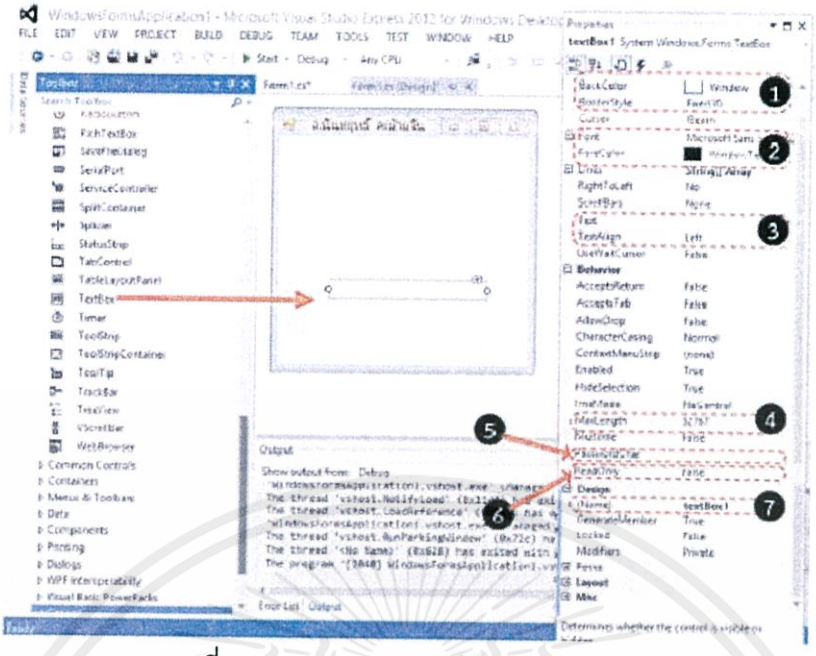
2.6.6.9 TextBox

TextBox เป็น Control ตัวหนึ่งที่มีไว้สำหรับรับค่าข้อความจากผู้ใช้งาน ซึ่งอาจจะเป็น Control ที่ใช้งานมากที่สุดก็ว่าได้ในการเขียนโปรแกรมที่ต้องมีการป้อนข้อมูลเข้ามาจำนวนมากๆ สัญลักษณ์ของ Control TextBox ดังแสดงในรูปที่ 2.42



รูปที่ 2.42 ภาพแสดงสัญลักษณ์ของ TextBox

วิธีการใช้TextBox คือให้ลาก TextBox ตามสัญลักษณ์ในรูปที่ 2.43 มาวางไว้บนฟอร์ม ซึ่ง TextBox จะมีProperties ที่สำคัญดังแสดงในภาพต่อไปนี้



รูปที่ 2.43 ภาพแสดงการใช้งาน TextBox

จากภาพสามารถอธิบายได้ดังนี้

หมายเลข 1 BackColor และ BorderStyle ส่วนของ Properties BackColor เป็นการกำหนดสีพื้นหลังให้กับ TextBox และส่วนของ BorderStyle คือ Properties สำหรับการกำหนดลักษณะของเส้นของของTextBox

หมายเลข 2 Font และ ForeColor ส่วนของ Properties Font เป็น Properties สำหรับการกำหนดรูปแบบของ Font ขนาดของตัวอักษรที่อยู่ใน TextBox และในสวนของ Properties ForeColor คือการกำหนดสีของตัวอักษรที่อยู่ภายใน TextBox

หมายเลข 3 Text และ TextAlign ส่วนของ Properties Text คือการกำหนดข้อความที่จะแสดงบนTextBox ปกติแล้วโปแกรม Microsoft Visual C# 2012 Express จะปล่อยเว้นว่างไว้ และใน PropertiesTextAlignคือ Properties สำหรับกำหนดตำแหน่งการแสดงข้อความว่าให้ชิดขอบซ้ายขวาหรือกึ่งกลางของของ TextBox

หมายเลข 4 MaxLenght เป็น Properties สำหรับกำหนดความยาวของตัวอักษรสูงสุดว่าใน TextBox นั้นสามารถกรอกตัวอักษรได้กี่อักขระ

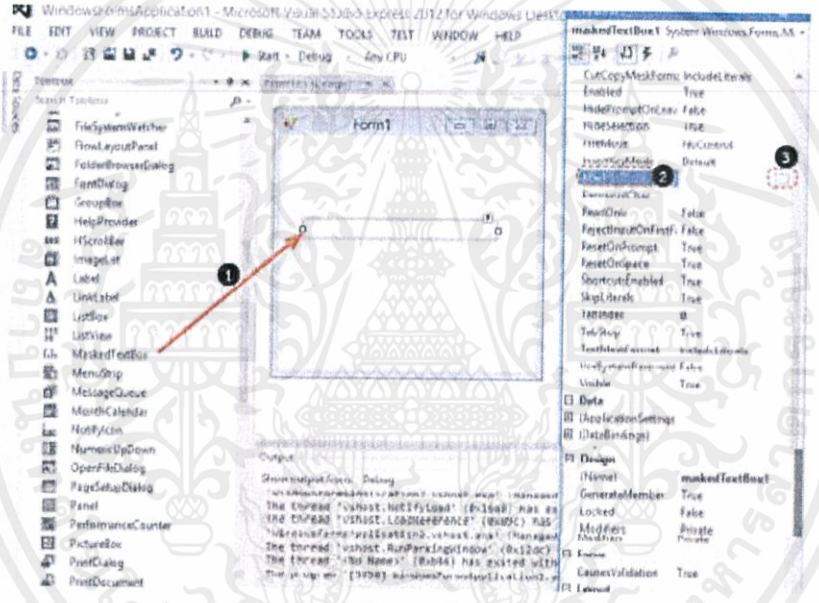
หมายเลข 5 PasswordChar คือ Properties สำหรับกำหนดรูปแบบในการแสดงผลเมื่อกรอก Password เช่น ถ้ากำหนดค่า Properties นี้โดยใส่เครื่องหมาย * และหลังจากที่รันโปรเจ็กต์ทดสอบดูแล้วเมื่อพิมพ์ข้อความเข้าไปใน TextBox นั้น จะทำให้ตัวอักษรเป็น * ซึ่งนิยมใช้ในช่องสำหรับกรอก Password

หมายเลข 6 ReadOnly หมายถึง การกำหนดค่าให้ TextBox สามารถอ่านได้อย่างเดียวแต่คีย์อะไรเข้าไปไม่ได้ ซึ่งสามารถเปลี่ยนได้อยู่ 2 ค่าคือ True และ False ซึ่งปกติแล้วโปแกรมจะตั้งค่า Default ไว้ที่ False ซึ่งหมายถึง สามารถกรอกข้อมูลเข้าไปได้ แต่ถ้าตั้งเป็น True จะหมายถึงไม่สามารถกรอกข้อความใดๆเข้าไปใน TextBox นั้นๆได้

หมายเลข 7 Nameเป็นPropertiesที่สำคัญซึ่งในการเขียนโปรแกรมแล้วถ้าต้องการเขียนโปรแกรมควบคุม Properties ของ TextBox ต้องรู้จักชื่อขง TextBox ก่อน ดังนั้นจึงแนะนำให้ตั้งเป็นภาษาอังกฤษและเหมาะและมีชื่อที่เหมาะสมสอดคล้องกับลักษณะงาน

2.6.6.10 MaskedTextBox

MaskedTextBox เป็น Control ตัวหนึ่งที่มีลักษณะคล้ายกับ Text Box ทุกประการ แต่MaskedTextBox จะมีส่วนช่วยในการจัดรูปแบบการกรอกข้อมูลลงใน Text Box ให้เป็นไปในรูปแบบเดียวกัน เช่น เบอร์โทร หมายเลขบัตรประชาชน เป็นต้น กล่าวคือ MaskedTextBox ก็คือ Text Box ตัวหนึ่งที่สามารถกำหนดรูปแบบการกรอกข้อมูลได้โดยการกำหนดจะกำหนดที่ Properties ของ MaskedTextBox ในที่นี้เนื่องจากมี Properties ที่คล้ายๆกับ Text Box ดังนั้นจะขอยกตัวอย่างเฉพาะ Properties ที่แตกต่างเท่านั้นสามารถอธิบายได้ดังนี้



รูปที่ 2.44 ภาพแสดงการใช้งาน MaskedTextBox

จากภาพสามารถอธิบายการใช้งานได้ดังนี้

ใน Properties ส่วนอื่นๆ จะคล้ายๆกับการใช้งาน Text Box จึงไม่ขอกล่าวถึง แต่ใน MaskedTextBox จะมีอีก Properties หนึ่งที่แตกต่างจาก Text Box นั่นคือ Properties Mask ซึ่งเป็น Properties สำหรับกำหนดรูปแบบการป้อนข้อมูลเข้า MaskedTextBox ซึ่งการใช้งานจะอธิบายตามหมายเลขได้ดังนี้

หมายเลข 1 ให้ลากเอา Control ที่ชื่อว่า MaskedTextBox มาวางที่ Form จากนั้นให้คลิกเลือกที่ตัวของ MaskedTextBox ที่ด้านขวามือก็จะเป็น Properties ของ MaskedTextBox แต่ถ้าไม่ขึ้นให้คลิกขวาที่MaskedTextBox แล้วเลือก Properties

หมายเลข 2 คือ Properties ที่สำคัญในการกำหนดรูปแบบการป้อนข้อมูลใน MaskedTextBox ในที่นี้สามารถกำหนดเองก็ได้ หรือถ้าจะใช้ที่โปรแกรม Microsoft Visual C# 2012

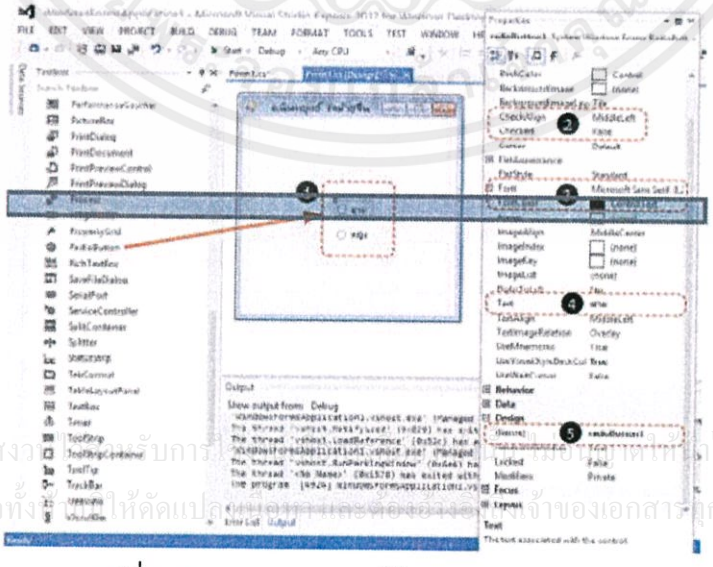
Express กำหนดมาให้เบื้องต้นให้คลิกเลือกที่หมายเลข 3 ซึ่งจะปรากฏหน้าต่างให้เลือกรูปแบบที่มีอยู่แล้วและนอกจากนี้ยังสามารถกำหนดรูปแบบเองได้อีกด้วย

หมายเลข 3 เป็นการเลือกรูปแบบของการใส่ข้อมูลที่โปรแกรม Microsoft Visual C# 2012 Express เตรียมมาให้ หรือ นอกจากนี้ยังสามารถกำหนดรูปแบบเพิ่มเติมเข้าไปได้ในส่วนของหมายเลข 3 นี้ ได้อีกด้วยที่ Properties Mask ของ MaskedTextBox สามารถกำหนดรูปแบบการพิมพ์ได้เอง โดยการพิมพ์รูปแบบเข้าไปที่ช่องว่างของ Properties ได้เลยโดยไม่ต้องคลิกที่หมายเลข 3 ก็ได้ ตัวอย่างเช่น ให้พิมพ์(000)-000000 เข้าไปที่ช่อง เมื่อทดลองรันโปรแกรมจะเห็นว่า เมื่อป้อนข้อมูลเข้าไปใน MaskedTextBox ข้อมูล 3ตัวแรกจะอยู่ภายในเครื่องหมายวงเล็บ และที่นอกวงเล็บจะมีขีดขั้นหลังจากนั้นก็จะเป็นข้อมูล 6 หลัก ซึ่งจะพิมพ์มากกว่านี้ไม่ได้ เพื่อให้มองเห็นภาพสามารถดูผลรันได้จากภาพต่อไปนี้

รูปที่ 2.45 ภาพแสดงผลการกำหนดรูปแบบการพิมพ์ข้อมูลใน MaskedTexBox

2.6.6.11 RadioButton

RadioButton คือ Control ในลักษณะของตัวเลือกซึ่งสามารถเลือกได้เพียงค่าเดียว นิยมใช้งานกับการเลือกเพศ เลือกตำแหน่ง ซึ่งวิธีการใช้งานสามารถอธิบายได้ดังรูปที่ 2.46 ดังนี้



รูปที่ 2.46 ภาพแสดงการใช้งาน RadioButton

จากภาพเป็นกานแสดงวิธีการใช้งาน RadioButton ซึ่งสามารถอธิบายได้ดังต่อไปนี้

หมายเลข 1 ให้ลาก RadioButton มาวางที่ Windows Form ในที่นี้จะให้เลือกเป็นเพศชายและเพศหญิงจึงลากเอา RadioButton มา 2 อัน และเปลี่ยน Properties Text ให้เป็นชายและหญิง

หมายเลข 2 เป็น Properties ที่ชื่อว่า Checked เป็น Propertiesสำหรับกำหนดค่า Default ในการChecked ที่ RadioButton ตัวใด และสวนของ CheckeAlign เป็นการกำหนดตำแหน่งการแสดง RadioButton ว่าอยู่ที่ด้านหน้าข้อความ ด้านหลังข้อความ เป็นต้น

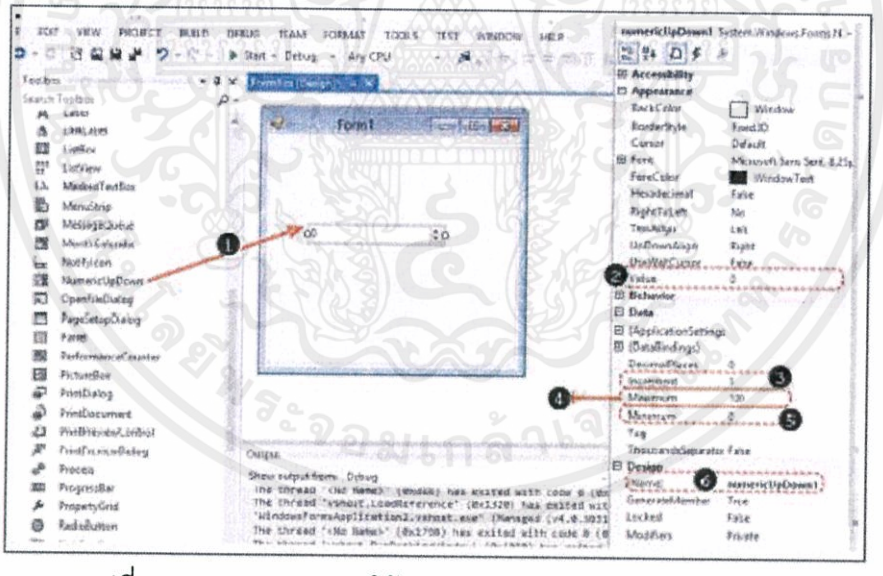
หมายเลข 3 Properties Font เป็นการกำหนดแบบอักษรและขนาดของตัวอักษรที่อยู่ข้างๆ หรือคำอธิบายของ RadioButton และในส่วนของ Properties ForeColor คือ Properties สำหรับกำหนดสีของตัวอักษรที่อยู่ข้างๆ RadioButton

หมายเลข 4 Text คือ Properties สำหรับกำหนดข้อความที่ปรากฏอยู่ข้างๆ RadioButton

หมายเลข 5 Name เป็น Properties สำหรับกำหนดชื่อให้กับ RadioButton เพื่อที่จะใช้ในการอ้างถึงในขั้นตอนการเขียนโปรแกรมต่อไป

2.6.6.12 numeric UpDown

Numeric UpDown เป็น Control สำหรับให้คลิกเลือกตัวเลขแบบแบบศรขึ้นลงโดยไม่ต้องกรอกตัวเลขเข้าไปเองเพื่อป้องกันการกรอกตัวอักษรหรือข้อมูลอย่างอื่น การใช้งานสามารถอธิบายได้ดังภาพต่อไปนี้



รูปที่ 2.47 ภาพแสดงการใช้งาน Control numeric UpDown

จากภาพอธิบายได้ว่า

หมายเลข 1 ให้ลาก Control numeric UpDown มาวางที่ Form จากนั้นเมื่อคลิกที่ numericUpDown ให้ Active จะได้ Properties แสดงที่ด้านขวา ซึ่ง Properties ที่สำคัญอธิบายในหัวข้อถัดไป

หมายเลข 2 เป็น Properties สำหรับกำหนดค่าเริ่มต้นหรือ ค่า Default ให้กับ numeric UpDown

หมายเลข 3 เป็น Properties สำหรับกำหนดระยะห่างของช่วงตัวเลขเมื่อกดที่ลูกศรขึ้นลง

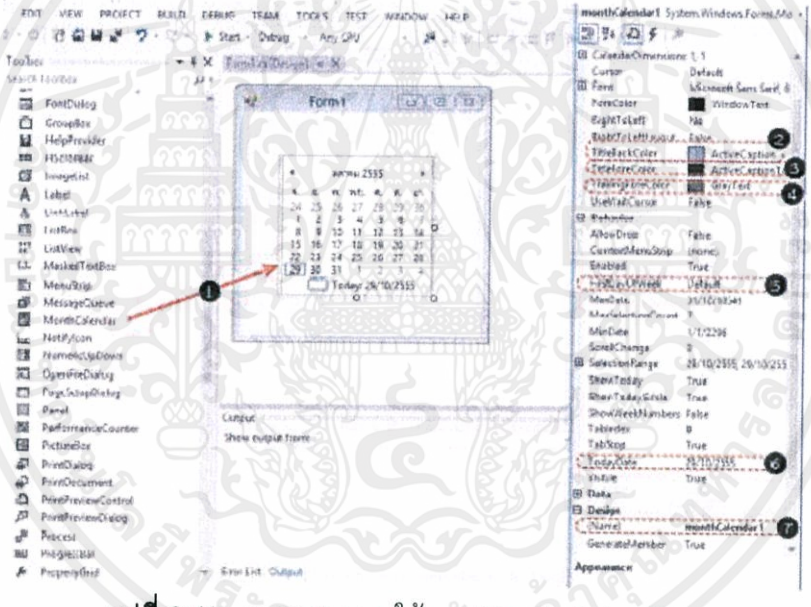
หมายเลข 4 เป็น Properties สำหรับกำหนดค่าสูงสุดที่ numeric UpDown สามารถเลื่อนขึ้นไปได้

หมายเลข 5 เป็น Properties สำหรับกำหนดค่าต่ำสุดที่ numeric UpDown สามารถเลื่อนลงไปได้

หมายเลข 6 เป็น Properties สำหรับกำหนดชื่อเพื่อนำไปอ้างอิงในการใช้งานตัว numeric UpDown ต่อไปในขั้นตอนการเขียนโค้ด

2.6.6.13 Month Calendar

Month Calendar เป็น Control ตัวหนึ่งที่มีไว้สำหรับให้เลือกเกี่ยวกับวันที่ รายละเอียดและ Properties ที่สำคัญสามารถอธิบายได้จากภาพต่อไปนี้



รูปที่ 2.48 ภาพแสดงการใช้งาน Month Calendar

จากภาพ เป็นการแสดงวิธีใช้งาน Control ที่ชื่อว่า Month Calendar ซึ่งรายละเอียดต่างๆ สามารถ อธิบายได้ดังนี้

หมายเลข 1 ลาก Control ชื่อว่า Month Calendar มาวางที่ Form ที่ด้านขวามือจะปรากฏ
Properties ให้ปรับแต่งเกี่ยวกับ Month Calendar

หมายเลข 2 คือ Properties สำหรับกำหนดสีของส่วนหัวของปฏิทิน

หมายเลข 3 คือ Properties สำหรับกำหนดสีของตัวอักษรในส่วนหัวของปฏิทิน

หมายเลข 4 คือ Properties สำหรับกำหนดสีวันที่ของเดือนก่อนและหลัง

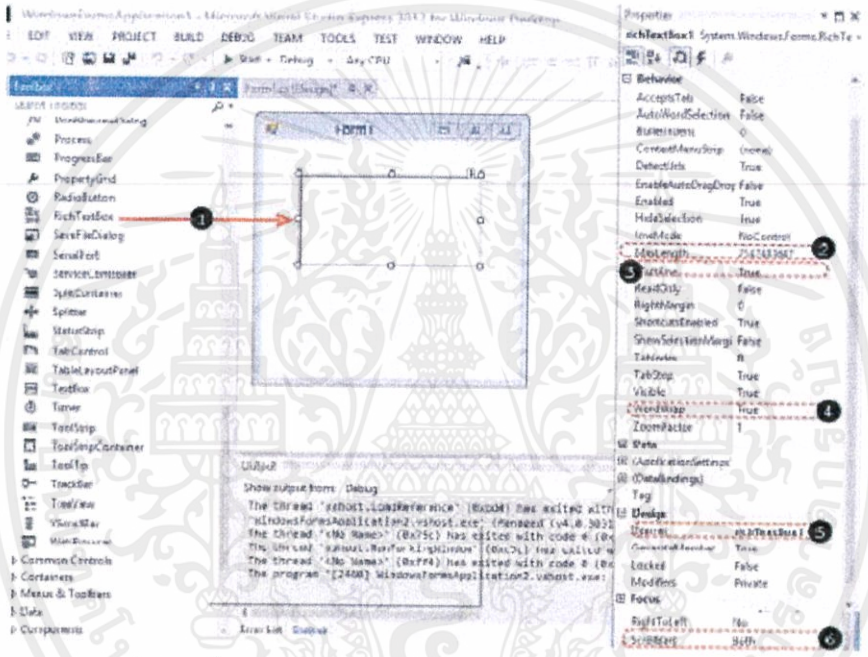
หมายเลข 5 คือ Properties สำหรับกำหนดวันเริ่มต้นของสัปดาห์ โดยปกติแล้วจะมีค่า Default เป็นวันจันทร์

หมายเลข 6 คือ Properties สำหรับกำหนดค่าของวันที่ปัจจุบัน

หมายเลข 7 คือ Properties สำหรับกำหนดชื่อให้กับ Month Calendar เพื่อที่จะนำไปใช้ในการอ้างอิงในการเขียนโปรแกรมหรือนำเอาค่าของ Month Calendar ไปใช้งาน

2.6.6.14 RichTextBox

RichTextBox คือ Control ตัวหนึ่งที่มีไว้สำหรับการกรอกข้อความคล้ายๆ กับ TextBoxทั่วไป แต่ RichTextBox จะมีความพิเศษคือสามารถพิมพ์ได้หลายๆแถว ซึ่งเมื่อข้อความลัน RichTextBox แล้ว Control RichTextBox จะแสดงแถบเลื่อนขึ้นลง หรือ Score bar ให้เลื่อนดูข้อความที่ซ่อนอยู่ การใช้งานและ Properties ที่สำคัญของ RichTextBox มีดังต่อไปนี้



รูปที่ 2.49 ภาพแสดงการใช้งาน RichTextBox

จากภาพสามารถอธิบายได้ดังนี้

หมายเลข 1 ให้ลาก RichTextBox มาวางที่ Form ขนาดของ RichTextBox สามารถย่อขยายได้ตามต้องการโดยคลิกที่ตัวของ RichTextBox เมื่อคลิกแล้วจะปรากฏสี่เหลี่ยมเล็กๆทั้งสี่ด้านของ RichTextBox ให้ทำการคลิกเมาส์ค้างแล้วลากสี่เหลี่ยมแต่ละด้านให้ได้ขนาดตามที่ต้องการ ส่วนวิธีการกำหนด Propertiesสามารถทำได้โดยคลิกขวาที่ตัว RichTextBox แล้วเลือกที่ Properties ซึ่งจะปรากฏหน้าต่างสำหรับกำหนด Properties ที่ด้านขวาของโปรแกรม ซึ่งโดยปกติถ้าไม่ได้ปิดส่วนประกอบของหน้าต่างโปรแกรม Microsoft Visual C# ใดๆ หน้าต่างสำหรับกำหนด Properties จะแสดงที่ด้านขวาอยู่แล้ว

หมายเลข 2 เป็น Properties สำหรับกำหนดค่าจำนวนตัวอักษรที่สามารถพิมพ์ได้สูงสุดใน RichTextBox ซึ่งปกติจะมีค่า Default มาให้เป็น 2,147,483,647 ตัวอักษร

หมายเลข 3 เป็น Properties สำหรับกำหนดว่าให้ Control RichTextBox มีลักษณะเป็น Multiline หรือสามารถพิมพ์ข้อความได้ในลักษณะของหลายแถว ซึ่งสามารถกำหนดได้ 2 ค่าคือ True

คือ สามารถพิมพ์ข้อความได้หลายแถว และ False คือ สามารถพิมพ์ข้อความได้เพียงแถวเดียวเรียงต่อไปเรื่อยๆโดยไม่ขึ้นบรรทัดใหม่ให้

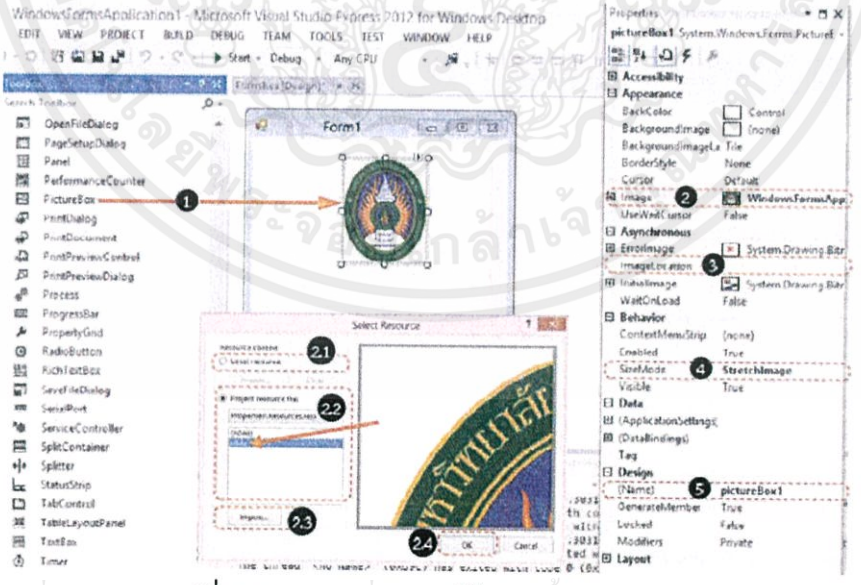
หมายเลข 4 เป็น Properties สำหรับกำหนดค่าให้มีการทำงานแบบ Word wrap หรือ การขึ้นบรรทัดใหม่ให้โดยอัตโนมัติเมื่อการพิมพ์ข้อความมาชนขอบด้านขวาของ RichTextBox ถ้ากำหนดค่าให้เป็น True หมายถึง ให้ขึ้นบรรทัดใหม่เมื่อสิ้นสุดขอบของ RichTextBox และหากกำหนดค่าเป็น False จะเป็นการปิดการใช้งาน Word wrap ซึ่งหมายถึง การพิมพ์ข้อความจะพิมพ์ไปเรื่อยๆถึงแม้จะชนขอบด้านขวาของ RichTextBox ก็ตาม จะไม่ขึ้นบรรทัดใหม่ให้ จนกว่าผู้ใช้จะกดที่ปุ่ม Enter จึงจะขึ้นบรรทัดใหม่

หมายเลข 5 เป็น Properties สำหรับกำหนดชื่อให้กับ RichTextBox เพื่อนำไปอ้างถึงในการใช้งานในขั้นตอนการเขียนโค้ด

หมายเลข 6 เป็น Properties สำหรับกำหนดรูปแบบการแสดง Score bar แนวตั้งหรือแนวนอนเมื่อจำนวนตัวอักษรล้น RichTextBox ซึ่งในที่นี้สามารถเลือกกำหนดค่าได้ ซึ่งมีหลายค่าแต่ค่าที่นิยมใช้งานคือ Vertical หมายถึงแสดง Score bar ในแนวตั้ง และ Horizontal หมายถึงแสดง Score bar ในแนวนอนแต่รูปแบบ Horizontal จะไม่เห็นผลถ้าเปิดใช้งาน Properties WordWrap ให้เป็น true ในหมายเลข 4 แต่ถ้าตั้งค่า Properties ในหมายเลข 6 เป็น None หมายถึง ไม่ใช้งาน Score bar

2.6.6.15 PictureBox

PicturBox เป็น Control สำหรับใส่รูปภาพหรือแทรกรูปภาพเข้าไปใน Form โดยสามารถเลือกภาพได้อย่างอิสระ การใช้งานและ Properties ที่สำคัญสามารถแสดงได้จากภาพต่อไปนี้



รูปที่ 2.50 ภาพแสดงการใช้งาน PictureBox

จากภาพสามารถอธิบายได้ดังนี้

หมายเลข 1 ลาก Control ที่ชื่อ PictureBox มาวางที่ Form จากนั้นถ้าต้องการกำหนดค่า Properteis ให้คลิกขวาที่ PictureBox ที่ Form แล้วเลือก Properties จะได้หน้าต่างสำหรับกำหนด Properties ให้กับ Picture Box ซึ่งโดยปกติถ้าไม่ได้ปิดส่วนประกอบของหน้าต่างโปรแกรม Microsoft Visual C# ใดๆ หน้าต่างสำหรับกำหนด Properties จะแสดงที่ด้านขวาอยู่แล้ว

หมายเลข 2 เป็นการเลือกรูปภาพจากเครื่องคอมพิวเตอร์จากตำแหน่งที่เก็บเพื่อนำ มาใส่ใน PictureBox เมื่อคลิกเลือกที่ Properties ที่ชื่อว่า Image หมายเลข 2 จะมีปุ่มสี่เหลี่ยมเล็กๆที่มุมขวาของ Properties Image สำหรับให้คลิกเลือกรูปภาพ ให้ทำการคลิกที่สี่เหลี่ยมเล็กๆนั้น หลังจากนั้นจะได้หน้าต่างสำหรับให้ใส่รูปภาพซึ่งจะอธิบายในข้อย่อย ต่อไปนี้

หมายเลข 2.1 Local resource เป็นการเลือกรูปภาพแบบใส่ภาพเข้าไปใน PictureBox เลยโดยไม่ต้องเก็บภาพให้สามารถใช้ได้กับส่วนอื่นๆของโปรเจ็กต์ ซึ่งถ้าคลิกเลือกที่หมายเลขนี้ หากต้องการใส่รูปภาพให้เลือกที่ปุ่ม Import แล้วเลือกรูปภาพ

หมายเลข 2.2 Project resource file เป็นการเลือกรูปภาพในรูปแบบการนำภาพมาเก็บไว้ใน Library (ส่วนที่ลูกศรสีแดงชี้) เพื่อให้สามารถเรียกใช้ภาพนั้นได้อีกเมื่อใช้ PictureBox ใน Form อื่นๆในโปรเจ็กต์เดียวกัน

หมายเลข 2.3 ในที่นี้ผู้เขียนเลือกใช้รูปแบบการเลือกรูปภาพแบบข้อ 2.2 ดังนั้นจึงใช้งานปุ่มImport ที่ด้านล่างของส่วนนี้ เพื่อเลือกรูปภาพมาใส่ไว้ใน Library

หมายเลข 2.4 ในกรณีที่เลือกรูปแบบการใส่รูปภาพแบบข้อ 2.1 คลิกที่ปุ่ม OK ได้เลย เมื่อเลือกภาพที่ต้องการได้แล้ว แต่ถ้าเลือกใช้งานในรูปแบบที่ 2.2 ในกรณีที่ได้ Import รูปภาพมาเป็นจำนวนมากให้เลือกรูปภาพที่ต้องการในส่วนที่ลูกศรสีแดงชี้ ก่อน แล้วค่อยคลิกที่ปุ่ม OK

หมายเลข 3 เป็น Properties สำหรับระบุตำแหน่งที่อยู่ของรูปภาพ จะใช้ในกรณีที่ทราบตำแหน่งของรูปภาพในเครื่องคอมพิวเตอร์แน่นอนแล้ว แต่วิธีการนี้ไม่ค่อยนิยมใช้ จะใช้วิธีในหมายเลข 2 แทน

หมายเลข 4 เป็น Properties สำหรับตั้งค่าลักษณะการแสดงผลรูปภาพใน PictureBox ว่าให้แสดงในลักษณะหรือโหมดใด เช่น แบบเต็ม PictureBox แบบอยู่กึ่งกลาง หรือแบบย่อขยายตามขนาดของ PictureBox เป็นต้น ซึ่งมีอยู่ด้วยกันหลายโหมดซึ่งสามารถเลือกทดสอบดูได้ในที่นี้ผู้เขียนยกตัวอย่างเป็นโหมดของ tretchImage ซึ่งเป็นโหมดที่ทำให้ขนาดรูปภาพที่แสดงขยายเต็มพื้นที่ของ PictureBoxซึ่งจะยืดหรือหดตามขนาดของPictureBox ที่กำหนดไว้

หมายเลข 5 เป็น Properties สำหรับกำหนดชื่อให้กับ PictureBox เพื่อนำไปใช้งานหรืออ้างอิงในการเขียนโปรแกรมเมื่อจะใช้งาน PictureBox

บทที่ 3
วิธีการดำเนินงาน

3.1 หลักการทำงาน

เมื่อเซนเซอร์ตรวจจับค่าออกซิเจนละลายน้ำได้ เอาต์พุตที่ได้จะออกมาเป็น mV ส่งผ่านตัว Dissolved Oxygen Circuit เพื่อชดเชย อุณหภูมิ, ความดัน และ ความเค็มและยังทำการขยายสัญญาณแล้ว ส่งค่าออกมาเป็น mg/L ในช่วงตั้งแต่ 0.01 ถึง 35.99 mg/L บอร์ด Arduino จะทำการรับค่าออกซิเจนละลายน้ำมาเพื่อทำการแปลงให้อยู่ในแบบการส่ง Serial port ไปยังคอมพิวเตอร์เพื่อแสดงผล และประมวลผล เพื่อทำการควบคุมกลับแบบอัตโนมัติ ถ้าค่า mg/L ที่ได้รับมามีค่าน้อยค่าคงที่ ที่ตั้งไว้ โปรแกรมจะทำการสั่งมอเตอร์ให้ทำงานเพื่อปั่นน้ำ เพื่อเพิ่มค่าออกซิเจนละลายน้ำให้สูงขึ้น จนเกินกว่าค่าที่กำหนดไว้ ในทางกลับกัน เมื่อค่าออกซิเจนละลายน้ำมีค่าสูงเกินกว่าที่กำหนดไว้ โปรมแกรมจะทำการสั่งให้มอเตอร์หยุดทำงาน

3.2 ติดตั้งโปรแกรม Arduino 1.0.6, Visual Studio 2012 และ Measurement Studio 2013

เริ่มแรกในการดำเนินโครงงานจะต้องติดตั้ง 3 โปรมแกรมที่จำเป็นนี้เพื่อใช้ในการเขียนโปรแกรมและอำนวยความสะดวกในการติดต่อกันระหว่างอุปกรณ์ถายนอกและคอมพิวเตอร์ แต่ละโปรแกรมนั้นก็มีหน้าที่ต่างกันไปโดยการเป็นตัวประมวลผลกลางและควบคุมกลับนั้น จะเป็นหน้าที่ของ Visual Studio 2012 ส่วนการควบคุมมอเตอร์และการรับข้อมูลในขั้นแรกเพื่อส่งต่อไปยังตัวประมวลผลกลางจะเป็นหน้าที่ของบอร์ด Arduino ซึ่งทั้ง 2 โปรแกรมนี้จะต้องเขียนโปรแกรมให้สองคล้องกันแต่ไม่เหมือนกัน ส่วนโปรแกรมสุดท้าย Measurement Studio 2013 เป็นโปรแกรมขยายในส่วนของโปรแกรม Visual Studio 2012 เนื่องจากต้องการใช้งานบางฟังก์ชันที่ Visual Studio 2012 ไม่มี ขั้นตอนการติดตั้งจะอยู่ในภาคผนวก

รูปที่ 3.1 สัญลักษณ์โปรแกรม Arduino, Visual Studio 2012 และ Measurement Studio 2013

3.3 การออกแบบโปรแกรมอ่านค่าและส่งข้อมูล ของบอร์ด Ardiuno

1. ส่วนแรกจะเป็นการประกาศการใช้งานของฟังก์ชัน SoftwareSerial.h และการใช้การส่งข้อมูลแบบ อนุมกรม โดยการเปิดการใช้งานพอร์ต rx และ tx รวมถึงการกำหนดชนิดค่าของตัวแปรต่างๆ

โปรแกรมที่ 3.1 การกำหนดฟังก์ชันการใช้งานและการกำหนดตัวแปร

```
#include <SoftwareSerial.h>
#define rx 2
#define tx 3

SoftwareSerial myserial(rx, tx);

char DO_data[20];
char computerdata[20];
byte received_from_computer=0;
byte received_from_sensor=0;
byte arduino_only=0;
byte startup=0;
byte string_received=0;
float DO_float=0;
float sat_float=0;
char *DO;
char *sat;
int incomingByte;
```

2. Void Setup ส่วนนี้จะเป็นการเปิดการใช้งานพอร์ตอนุกรมและตั้งค่า Baud Rate รวมไปถึงการกำหนดพอร์ตอินพุตและเอาต์พุต

โปรแกรมที่ 3.2 การตั้ง Baud Rate และการกำหนดพอร์ตอินพุตและเอาต์พุต

```
void setup(){
    Serial.begin(38400);
    myserial.begin(38400);
    pinMode(8, OUTPUT);

          }
```

3. โปรแกรมที่คอยตรวจรับว่ามีการส่งคำสั่งหรือข้อมูลมาจากคอมพิวเตอร์หรือไม่ ถ้าไม่มีก็จะผ่านโปรแกรมส่วนนี้ไปเลย แต่ถ้ามีก็จะทำการอ่านคำสั่งหรือข้อมูลนั้นจนกระทั้งสิ้นสุดที่ <CR> แล้วนำข้อมูลหรือคำสั่งนั้นกระทำ หรือแสดงผลต่อไป จากนั้นจะทำการล้างข้อมูลนั้นเพื่อทำการรอคำสั่งหรือข้อมูลถัดไป

โปรแกรมที่ 3.3 โปรแกรมตรวจรับคำสั่งหรือข้อมูลมาจากคอมพิวเตอร์

```
void serialEvent(){
        if(arduino_only!=1){
           received_from_computer=Serial.readBytesUntil(13,computerdata,20);
           computerdata[received_from_computer]=0;
           myserial.print(computerdata);
           myserial.print('\r');
          }
        }
```

4. โปรมแกรมตั้งแต่ส่วนนี้เป็นต้นไป จะเป็นโปรแกรมที่อยู่ในส่วนของ Void Loop คือ โปรมแกรมที่อยู่ในส่วนนี้จะทำงานวนซ้ำไปซ้ำมาเป็น loop จนกว่าจะหยุดการเชื่อมต่อ โปรมแกรมนี้คือส่วนรอรับคำสั่งให้เปิด-ปิดมอเตอร์ โปรแกรมจะคอยอ่าน incomingByte (ถูกกำหนดให้เป็นตัวแปรชนิด int) ตลอดเวลาตามเงื่อนไข Serial.available() > 0 นั้นเอง ถ้าอ่านเข้ามาเป็น H จะทำการเปิดมอเตอร์ ที่พอร์ต 8 แต่ถ้าเป็น L จะทำการสั่งปิดมอเตอร์ที่พอร์ต 8 เช่นกัน

โปรแกรมที่ 3.4 โปรแกรมคอยรับคำสั่งจากคอมพิวเตอร์เพื่อเปิดปิดมอเตอร์

```
void loop(){

    if (Serial.available() > 0) {
       incomingByte = Serial.read();
     if (incomingByte == 'H') {
     digitalWrite(8, HIGH);
     }
     if (incomingByte == 'L') {
      digitalWrite(8, LOW);
     }

  }
```

5. โปรแกรมนี้จะอยู่ใน Void Loop เช่นกัน จะคอยตรวจว่ามีการส่งข้อมูลในพอร์ตอนุกรมหรือไม่ ถ้ามีก็จะทำการอ่านข้อมูลนั้นจนกระทั้งสิ้นสุดที่ <CR> แล้วนำข้อมูลนั้นแสดงผลต่อไป จากนั้นจะทำการล้างข้อมูลนั้นเพื่อทำการรอคำสั่งหรือข้อมูลถัดไป

โปรแกรมที่ 3.5 โปรแกรมตรวจรับข้อมูลที่มาจาก DO EZO circuit

```
if(myserial.available() > 0){
   received_from_sensor=myserial.readBytesUntil(13,DO_data,20);
   DO_data[received_from_sensor]=0;
   string_received=1;

   if((DO_data[0] >= 48) && (DO_data[0] <=57)){
      pars_data();
      }
   else
     Serial.println(DO_data);
```

6. Void pars_data() คือส่วนหนึ่งของโปรแกรมด้านบนที่จะกระทำถ้าตรงตามเงื่อนไข (DO_data[0] >= 48) && (DO_data[0] <=57) คือ ข้อมูลที่ได้ต้องเป็นตัวเลขไม่ใช่ตัวหนังสือ

โปรแกรมที่ 3.6 ส่วนย่อยในการรับข้อมูล DO EZO circuit

```
void pars_data()
{
    byte i;
  byte pars_flag=0;

  for(i=0;i<=received_from_sensor;i++)
      {
        if(DO_data[i]==','){pars_flag=1;}
      }
  if(pars_flag){
    DO=strtok(DO_data, ",");
  sat=strtok(NULL, ",");
  Serial.print("DO:");
  Serial.println(DO);
  //DO_float=atoi(DO);

  Serial.print("%sat:");
  Serial.println(sat);
  //sat_float=atoi(sat);
  }
  else
  {
   Serial.print("DO:");
   Serial.println(DO_data);
  }
}
```

3.4 การออกแบบโปรแกรมอ่านค่าแสดงผล และควบคุมมอเตอร์โดย Visual Studio

1. ส่วน Head เป็นการกำหนดการใช้งานฟังก์ชันที่ใช้เขียนโปรแกรม โดยฟังก์ชันที่ขึ้นด้วย System คือ ฟังก์ชันเดิมของโปรแกรม Visual Studio 2012 และ ฟังก์ชันที่ขึ้นด้วย NationalInstruments คือ ฟังก์ชันของส่วนขยายที่ได้จากการติดตั้ง Measurement Studio 2013 ตัวอย่างฟังก์ชันที่สำคัญได้แก่ System.Windows.Forms และ NationalInstruments.UI.WindowsForms

โปรแกรมที่ 3.7 ฟังก์ชันหลักและฟังก์ชันส่วนขยายในโปรแกรม Visual Studio 2012

```
using NationalInstruments.Net;
using NationalInstruments.Analysis;
using NationalInstruments.Analysis.Conversion;
using NationalInstruments.Analysis.Dsp;
using NationalInstruments.Analysis.Dsp.Filters;
using NationalInstruments.Analysis.Math;
using NationalInstruments.Analysis.Monitoring;
using NationalInstruments.Analysis.SignalGeneration;
using NationalInstruments.Analysis.SpectralMeasurements;
using NationalInstruments;
using NationalInstruments.UI;
using NationalInstruments.DAQmx;
using NationalInstruments.NetworkVariable;
using NationalInstruments.NetworkVariable.WindowsForms;
using NationalInstruments.Tdms;
using NationalInstruments.UI.WindowsForms;
using NationalInstruments.Controls;
using NationalInstruments.Controls.Rendering;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;
using System.Media;
```

2. ส่วนนี้เป็นการประกาศและกำหนดชนิดตัวแปร

โปรแกรมที่ 3.8 แสดงการประกาศตัวแปร

```
namespace dissolvedOxygen
{
    public partial class Form1 : Form
    {

        private SerialPort myport;
        private DateTime datetime;
        private string in_data;

        string specifier;
        float value_O2 = 0;
        int relayON = 0 ;
        int relayOFF = 0;
        private SoundPlayer _soundPlayer;
```

3. ส่วนนี้เป็นการกำหนดตัวแปรให้ _soundPlayer เป็นฟังก์ชันเล่นไฟล์เสียงโดยการเล่นไฟล์ชื่อ Alert3.wav

โปรแกรมที่ 3.9 การกำหนดตัวแปรให้กับไฟล์เสียง

```
public Form1()
{
    InitializeComponent();
    _soundPlayer = new SoundPlayer("Alert3.wav");
}
```

4. ส่วนนี้เป็นการโหลดชื่อพอร์ตจากคอมพิวเตอร์ ว่า ณ ขณะนี้มีพอร์ตไหนบ้างที่เชื่อมต่อกับคอมพิวเตอร์อยู่แล้วให้นำไปเก็บเป็นตัวเลือกที่ comboBox1 เพื่อใช้เป็นตัวเลือกพอร์ตที่จะติดต่อกับบอร์ด Arduino

โปรแกรมที่ 3.10 แสดงการโหลดชื่อพอร์ตที่เชื่อมต่อกับคอมพิวเตอร์

```
private void Form1_Load(object sender, EventArgs e)
{
    string[] list = SerialPort.GetPortNames();
    foreach (string port in list)
    {
        comboBox1.Items.Add(port);
    }
}
```

5. การคลิ้กปุ่ม Start จะทำให้เกิดการทำคำสั่งต่างๆภายในนี้ เช่น การกำหนดตัวแปร myport ให้เป็นการติดต่อทาง SerialPort , การกำหนด Baud Rate ให้ทำกับ 38400 baud/sec และการกำหนดชื่อพอร์ตที่จะใช้ติดต่อ ซึ่งได้มาจาก comBox1 นั้นเอง ที่สำคัญของการคลิ้กปุ่ม Start คือ การสั่งให้ SerialPort เปิดแล้วเริ่มการส่งถ่ายข้อมูล

โปรแกรมที่ 3.11 แสดงคำสั่งต่างๆของปุ่ม Start

```
private void start_btn_Click(object sender, EventArgs e)
{

    myport = new SerialPort();
    myport.BaudRate = 38400;
    myport.PortName = port_name_tb.Text;
    myport.Parity = Parity.None;
    myport.DataBits = 8;
    myport.StopBits = StopBits.One;
    myport.DataReceived += myport_DataReceived;

    try
    {
        myport.Open();
        data_tb.Text = "";
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Error");
    }

}
```

6. ส่วนนี้เป็นการสั่งให้ in_data ที่ถูกกำหนดเป็นตัวแปรชนิด string อ่านข้อมูลจาก SerialPort แล้วนำไปแสดงผลที่ displaydata_event

โปรแกรมที่ 3.12 แสดงการเขียนคำสั่งอ่านข้อมูลจาก SerialPort

```
private void myport_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
    in_data = myport.ReadLine();

    this.Invoke(new EventHandler(displaydata_event));

}
```

7. เมื่อนำข้อมูลเข้ามายังคอมพิวเตอร์ได้แล้ว ส่วนต่อไปนี้คือการแสดงผลข้อมูลที่ได้ออกมา บนจอแสดงผล จะประกอบไปด้วย 1). ค่าออกซิเจนละลายน้ำที่วัดเข้ามา โดยเพิ่มวันเวลาแบบ Real time เข้าไปด้วย 2). ส่วนของกราฟนั้นจะถูกพรอตกราฟโดยตัวแปร Value_O2 ซึ่งเป็นตัวแปรชนิด int ถูกแปลงมาจาก in_data ที่เป็นตัวแปรชนิด string 3). ส่วนสุดท้ายคือปุ่มคอนโทรลต่างๆซึ่งจะทำการอธิบายต่อไป

โปรแกรมที่ 3.13 แสดงการเขียนคำสั่งแสดงผลข้อมูลบนจอ

```
private void displaydata_event(object sender, EventArgs e)
{
    datetime = DateTime.Now;
    string time = datetime.Day + "/" + datetime.Month + "/" + datetime.Year + "\t\t" +
        datetime.Hour + ":" + datetime.Minute + ":" + datetime.Second;
    data_tb.AppendText(time + "\t\t" + in_data + "\n");

        if (in_data.Length > 0)
    {
        value_O2 = float.Parse(in_data.Substring(3, 4));
        waveformGraph1.PlotYAppend(value_O2);
    }
```

8. ส่วนนี้จะเกิดการกระทำเมื่ออยู่ในโหมด Auto เท่านั้น มีคำสั่งควบคุมการ เปิด-ปิด มอเตอร์ และ การแจ้งเตือนเสียง โดยคำสั่งการเปิดปิดนั้นจะขึ้นอยู่กับการกำหนด Lower Cursor และการแจ้งเตือนขึ้นอยู่กับ Alarm Cursor

โปรแกรมที่ 3.14 แสดงการเขียนคำสั่งโหมด Auto

```
if (sys_btn.Text == "Auto")
{

                    if (value_O2 < Lower.YPosition)
                    {

                        myport.Write("H");

                    }


                    else
                    {
                        myport.Write("L");

                    }


                    if (value_O2 <= Alarm.YPosition && value_O2 >= Lower.YPosition)
                    {
                        _soundPlayer.Play();
                        label19.Text = "Warning!";
                    }
                    else
                    {
                        _soundPlayer.Stop();
                        label19.Text = "Normal";
                    }
}
```

9. เมื่อมีการคลิ้กปุ่ม Exit จะมี Message Box เด้งขึ้นมาถามว่า “Do you want to exit the program” มี 3 ปุ่มให้เลือก Yes, No, Cancel ถ้ากด Yes การปิดโปรแกรม

โปรแกรมที่ 3.15 แสดงการเขียนคำสั่งปุ่ม Exit

```
private void exit_btn_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Do you want to exit the program?", "Please confirm",
        MessageBoxButtons.YesNoCancel, MessageBoxIcon.Information) == DialogResult.Yes)
    {
        Application.Exit();

    }
}
```

10. เมื่อมีการคลิ้กปุ่ม Stop จะทำให้ SerialPort ถูกปิด

โปรแกรมที่ 3.16 แสดงการเขียนคำสั่งปิด SerialPort

```
private void stop_btn_Click(object sender, EventArgs e)
{
    try
    {
        myport.Close();

    }
    catch (Exception ex2)
    {
        MessageBox.Show(ex2.Message, "Error");
    }
}
```

11. ปุ่ม Save ทำหน้าที่ในการบันทึกข้อมูล วัน เวลา และ ค่าออกซิเจนละลายน้ำ เป็นไฟล์ *.txt

โปรแกรมที่ 3.17 แสดงการเขียนคำสั่งการบันทึกข้อมูล

```
private void save_btn_Click(object sender, EventArgs e)
{
    try
    {

        string pathfile = @"C:\Users\Chatchawan\Desktop\DATA\";
        string filename = "Dissolved_Data.txt";
        System.IO.File.WriteAllText(pathfile + filename, data_tb.Text);
        MessageBox.Show("Data has been saved" + pathfile, "Save File");
    }
    catch (Exception ex3)
    {
        MessageBox.Show(ex3.Message, "Error");
    }
}
```

12. ปุ่มควบคุมมอเตอร์ ต้องอยู่ในโหมด Manual เท่านั้น โดยการทำงานในคำสั่งนี้จะทำการส่ง “H” ออกมา 50 ครั้งเพื่อทำการ เปิดมอเตอร์ การปิดก็เช่นกัน (การส่ง 50 ครั้งนั้นใช้เวลาเพียงเสี้ยววินาที ไม่ต่างจากการส่งเพียงครั้งเดียว) เนื่องจากการอ่านโปรแกรมในแต่ละรอบนั้นไวมากหาก ส่งคำสั่งออกมาแค่ครั้งเดียวโปรแกรมที่ค่อยรอรับคำสั่งก็จะผ่านเลยไปโดยที่ไม่ได้รับคำสั่งเลย ส่วนตัวแปร relayON และ relayOFF นั้นจะทำหน้าที่ในการนับรอบให้ครบ 50 รอบโดยการใช้คำสั่ง Loop While เมื่อครบ 50 รอบ ก็จะทำการเคลียค่าทิ้ง เพื่อรอการคลิ้กครั้งต่อไป

โปรแกรมที่ 3.18 แสดงการเขียนคำสั่งโหมด Manual

```
private void on_btn_Click(object sender, EventArgs e)
{
    if (relayON == 50) { relayON = 0; }
    if (relayOFF == 50) { relayOFF = 0; }
    if (sys_btn.Text == "Manual")
    {
        if (on_btn.Text == "ON")

            {
                while (relayON < 50)
                {
                    on_btn.Text = "OFF";
                    myport.Write("H");
                    on_btn.BackColor = System.Drawing.Color.FromName("Green");

                    relayON++;
                }
            }
        else
        {

            while(relayOFF < 50)
            {
            myport.Write("L");
            on_btn.Text = "ON";
            on_btn.BackColor = System.Drawing.Color.FromName("Control");

            relayOFF++;
            }
        }
    }
}
```

13. กลุ่มคำสั่ง Option การคลิ้กปุ่มเหล่านี้จะทำการส่ง “R/r”, “C/r”, “E/r” คือการสั่งให้บอร์ด Arduino สั่ง DO EZO circuit อีกที่ให้ทำการ อ่านข้อมูลกลับมาในเร็ว 500 ms/ครั้ง , ให้เปิดโหมด Calibration และ ให้หยุดการอ่านข้อมูล ตามลำดับของคำสั่งที่กล่าวไว้ข้างต้น

โปรแกรมที่ 3.19 การเขียนคำสั่งกลุ่ม Option

```
private void button3_Click(object sender, EventArgs e)
{
    myport.Write("R\r");
}

private void button2_Click(object sender, EventArgs e)
{
    myport.Write("C\r");
}

private void button4_Click(object sender, EventArgs e)
{
    myport.Write("E\r");
}
```

14. ปุ่มคำสั่งเลือกระบบในการควบคุมมี 2 ระบบ คือ Auto และ Manual ปุ่มนี้จะเป็นการคลิ้กแบบ Button ธรรมดา แต่การเขียนคำสั่งทำให้กลายเป็น Toggle Button

โปรแกรมที่ 3.20 การเขียนคำสั่งปุ่ม System Control

```
private void sys_btn_Click(object sender, EventArgs e)
{
    on_btn.BackColor = System.Drawing.Color.FromName("Control");

    if (sys_btn.Text == "Manual")
    {

        sys_btn.Text = "Auto";

    }
    else
    {
        sys_btn.Text = "Manual";

    }
}
```

15. เมื่อมีการเลือก พอร์ตใน comboBox1 ที่กล่าวไว้ข้างต้นแล้ว ชื่อพอร์ตนั้นก็จะไปแสดงใน port name textbox เช่นกัน เพื่อนำไปใช้ในการกดปุ่ม Start

โปรแกรมที่ 3.21 แสดงการนำชื่อ port ที่เลือกไปใช้ต่อ

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    port_name_tb.Text = comboBox1.Text;
}
```

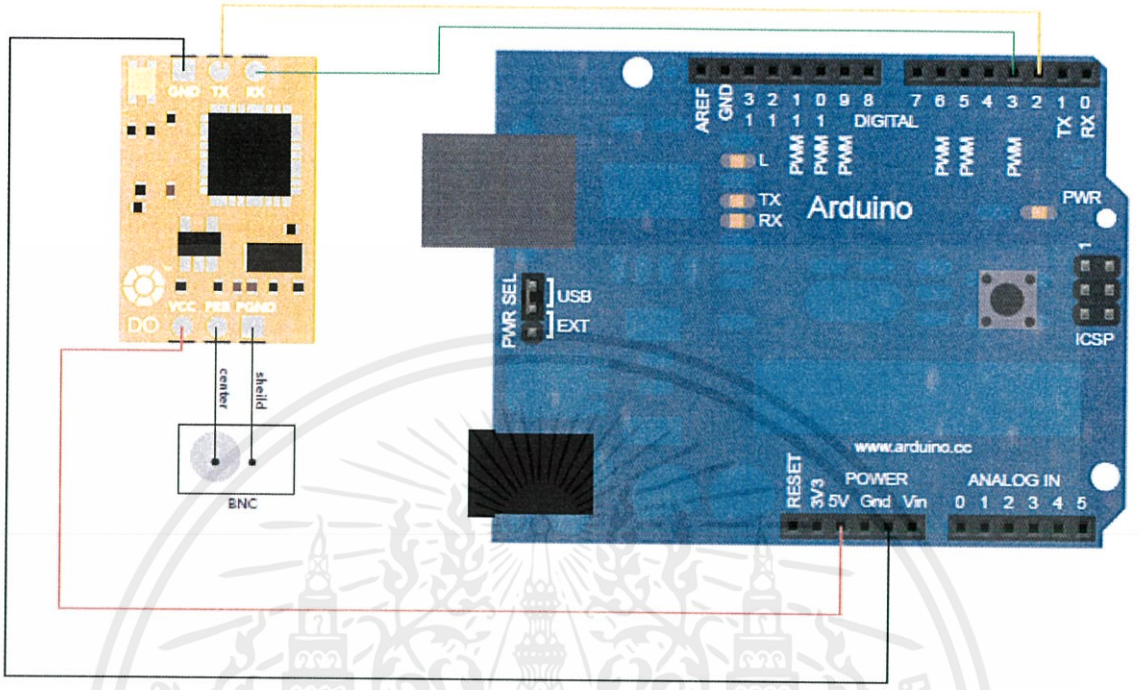
16. เป็นคำสั่งในการกำหนดค่า Lower และ Upper ให้อยู่ในระดับที่ต้องการควบคุม แล้วทำการแสดงค่าที่เปลี่ยนนั้นที่ label14 และ label15 ตัวอย่างการทำงาน เช่น ถ้าค่า Lower ต่ำกว่า 4 mg/L มอเตอร์ก็จะทำงาน

โปรแกรมที่ 3.22 แสดงคำสั่งการกำหนดค่า Lower และ Upper

```
private void waveformGraph1_CursorChanged(object sender, EventArgs e)
{
    specifier = "G";

    label15.Text = Lower.YPosition.ToString(specifier);
    label14.Text = Upper.YPosition.ToString(specifier);


}
```

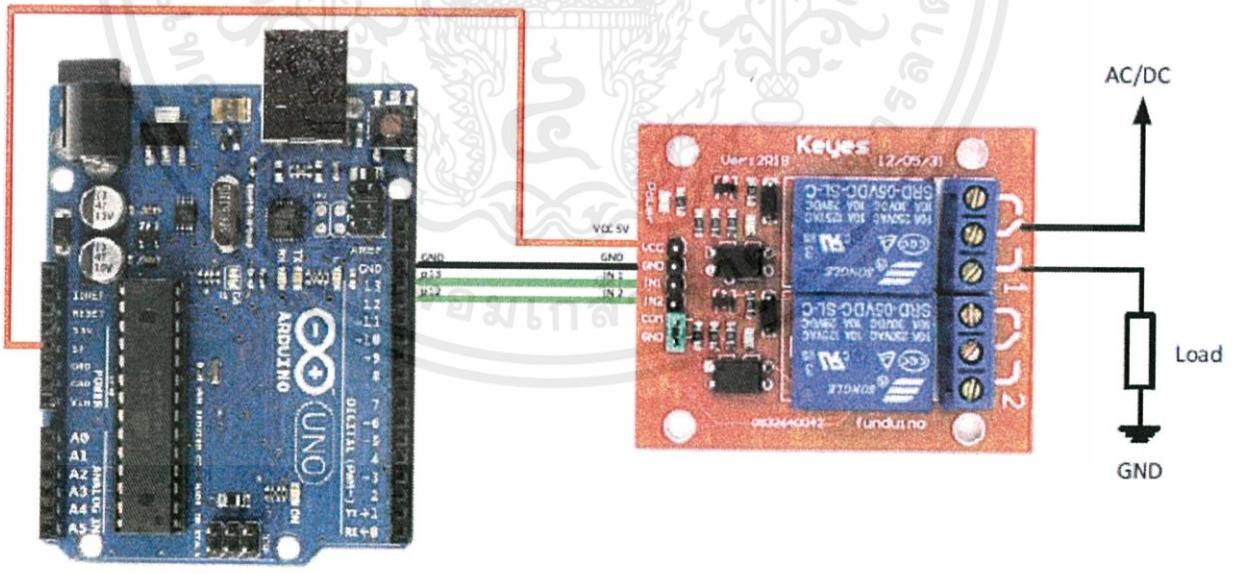
3.5 การต่อวงจร

3.5.1 การต่อวงจรบอร์ด Arduino กับ Sensor



รูปที่ 3.2 การต่อวงจรบอร์ด Arduino กับ Sensor

3.5.2 การต่อวงจรบอร์ด Arduino กับ Relay

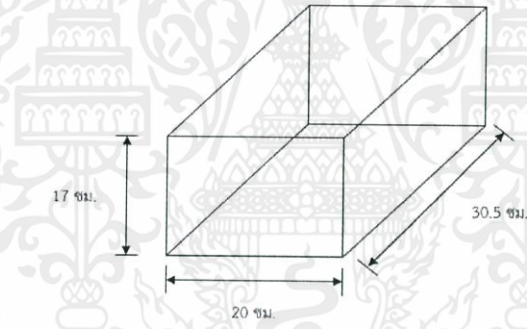


รูปที่ 3.3 การต่อวงจรบอร์ด Arduino กับ Relay

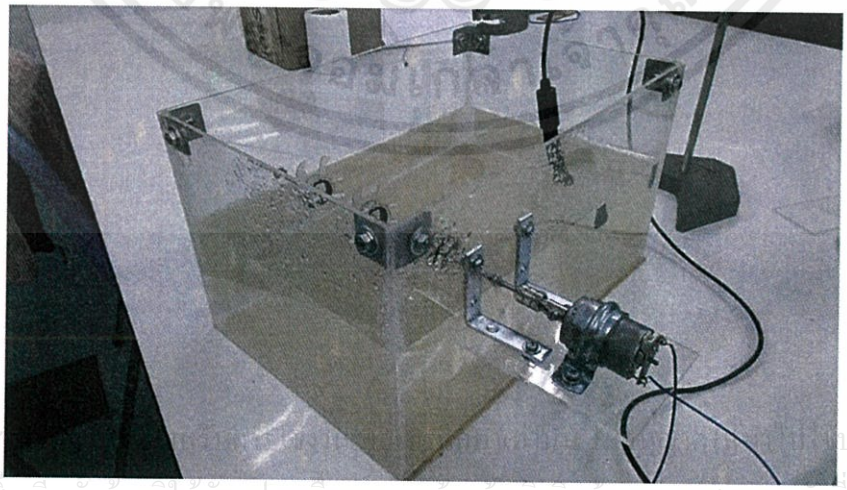
3.6 อุปกรณ์การทดลอง

- Dissolved Oxygen Sensor
- บอร์ด Arduino
- สายไฟ
- Relay
- มอเตอร์
- แผ่นอาคลิลิค
- ซิลิโคน
- น้ำยาประสานแผ่นอาคลิลิค,
- น็อตตัวเมีย ตัวผู้ แหวน เหล็กฉาก
- ใบพัด, แกนใบพัด

3.7 แบบจำลองบ่อที่ใช้ในการทดลอง



รูปที่ 3.4 ภาพแสดง Dimension บ่อ

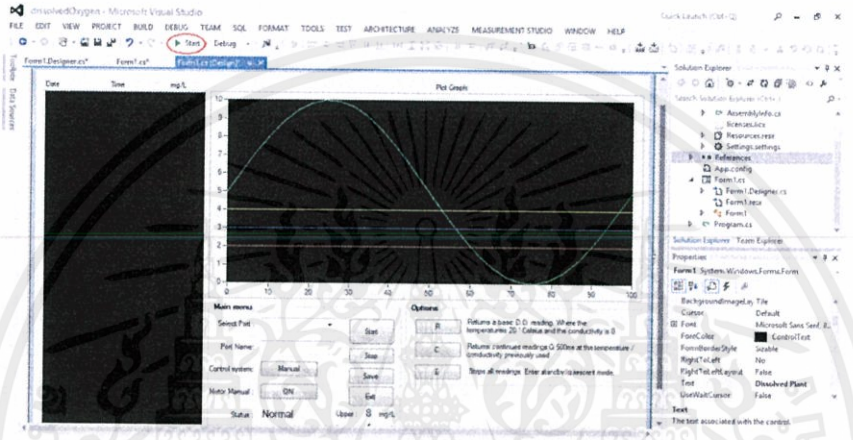


รูปที่ 3.5 ภาพแสดงบ่อทดลองจริง

บทที่ 4
การทดลองและแสดงผล

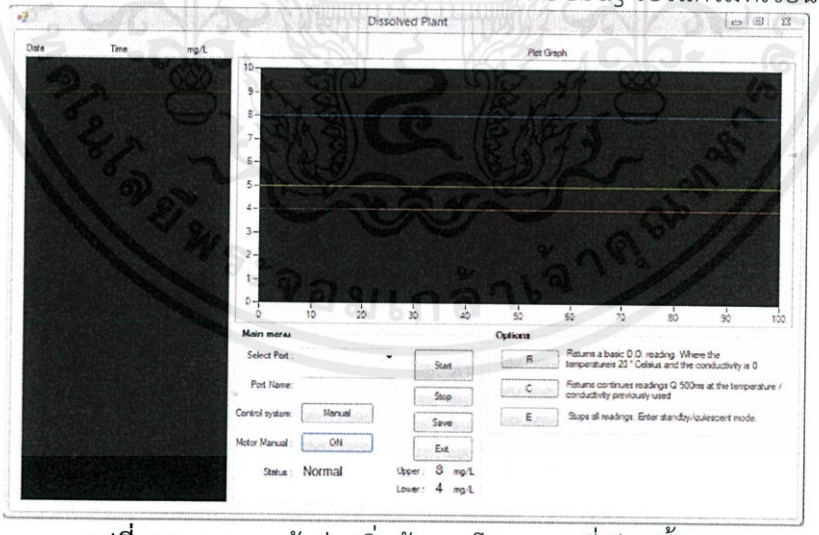
4.1 การทดลองใช้งานโปรแกรมอ่านค่าและควบคุมกลับเขียนโดย Visual Studio

1. เปิด folder “dissolved2” > dissolvedOxygen > dissolvedOxygen.sln จากนั้น โปรแกรมที่เขียนเสร็จแล้วจะขึ้นมาเป็นดังรูปที่ 4.1 จากนั้นเริ่ม Run โปรแกรมโดยการคลิ้กที่ Start



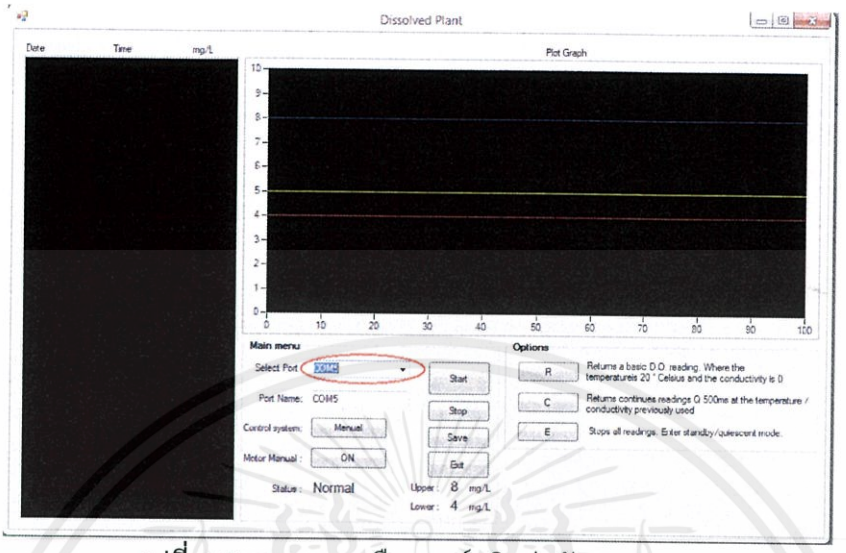
รูปที่ 4.1 แสดงหน้าต่างโปรแกรมอ่านค่าและควบคุมกลับ

2. เมื่อคลิ้กที่ Start โปรแกรม Visual Studio จะทำการ Debug โปรแกรมที่เขียนขึ้นมา



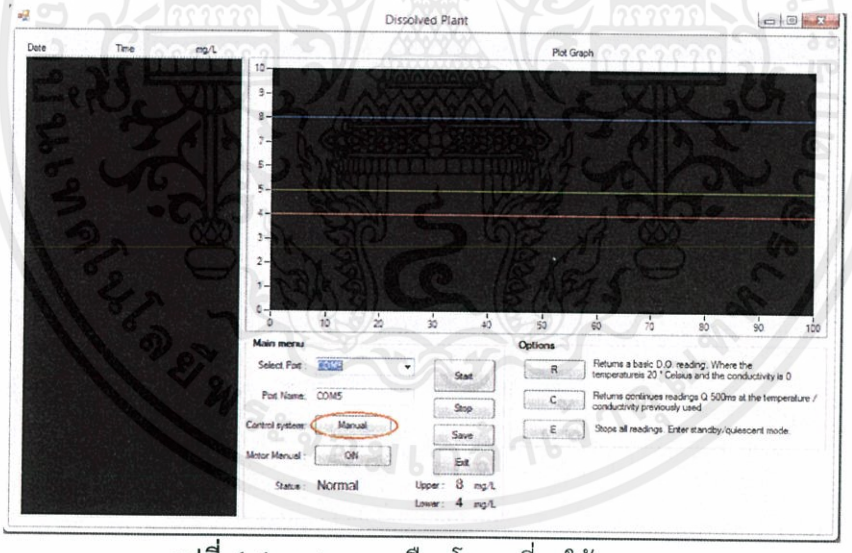
รูปที่ 4.2 แสดงหน้าต่างเริ่มต้นของโปรแกรมที่เขียนขึ้นมา

3. ทำการเลือกพอร์ตที่เชื่อมต่อกับบอร์ด Arduino ในที่นี้โปรแกรมจะทำการแสกนพอร์ตที่เชื่อมต่อกับคอมพิวเตอร์ออกมาทั้งหมดอยู่ที่ comboBox1 ซึ่งพอร์ตที่เชื่อมต่อกับ Arduino คือ พอร์ต 5



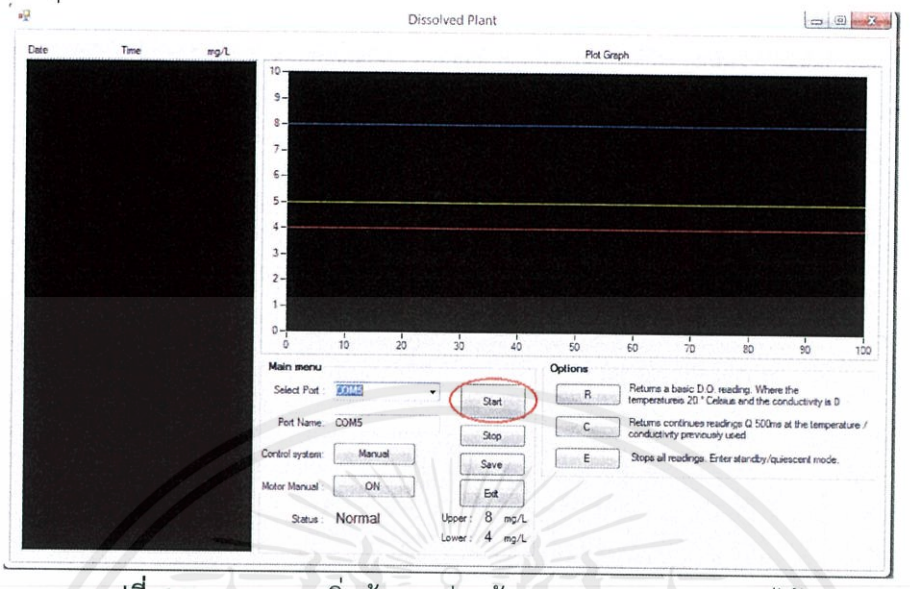
รูปที่ 4.3 แสดงการเลือกพอร์ตติดต่อกับ Arduino

4. จากนั้นทำการเลือกโหมดที่ต้องการควบคุม โดยถ้าไม่คลิ้ก Default ของโหมดคือโหมด Manual



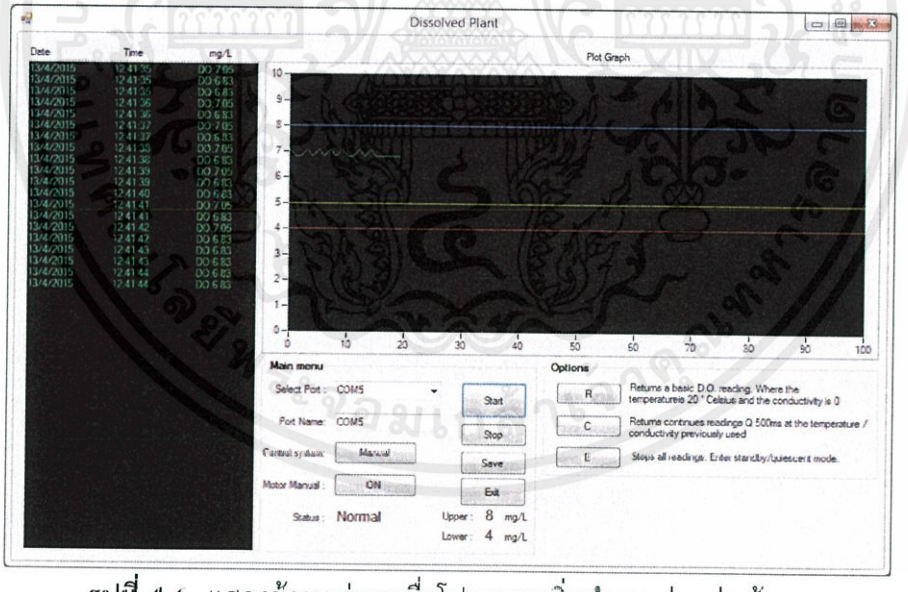
รูปที่ 4.4 แสดงการเลือกโหมดที่จะใช้ควบคุม

5. จากนั้นทำการคลิ้กที่ Start ดังรูปที่ 4.5 โปรแกรมจะเริ่มรับค่าที่อ่านมาและสามารถเริ่มควบคุมเปิดปิดในโหมด Manual ได้



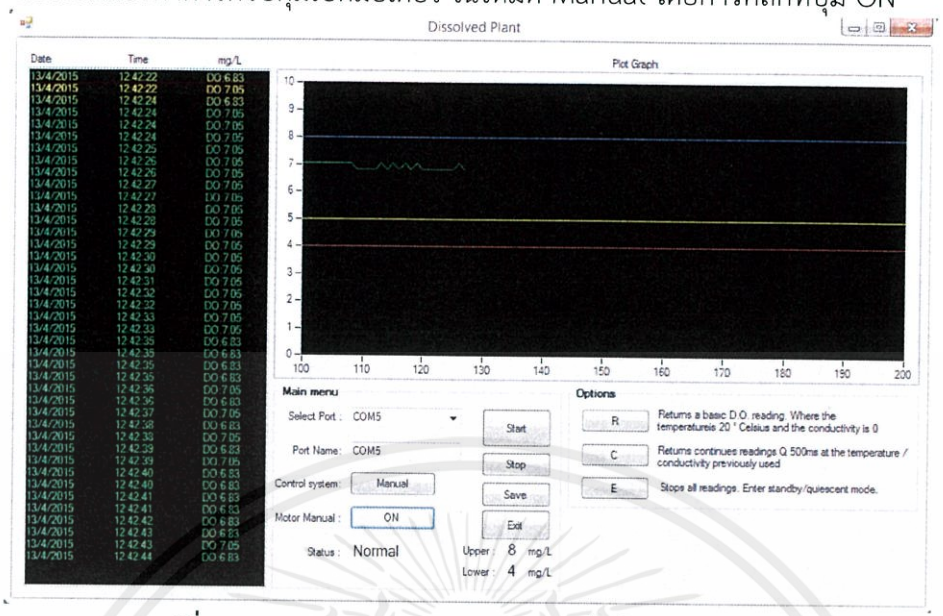
รูปที่ 4.5 แสดงการเริ่มต้นการอ่านข้อมูลและสามารถควบคุมได้

6. หน้าต่างทางด้านขวาจะเริ่มทำการแสดงข้อมูลแบบ Real time โดยกบอก วัน เวลา และค่าออกซิเจนละลายน้ำตามลำดับ ด้านซ้ายจะเป็นก็จะเริ่มพรอตกราฟในแกน Y ตามค่าออกซิเจนละลายน้ำที่อ่านมา



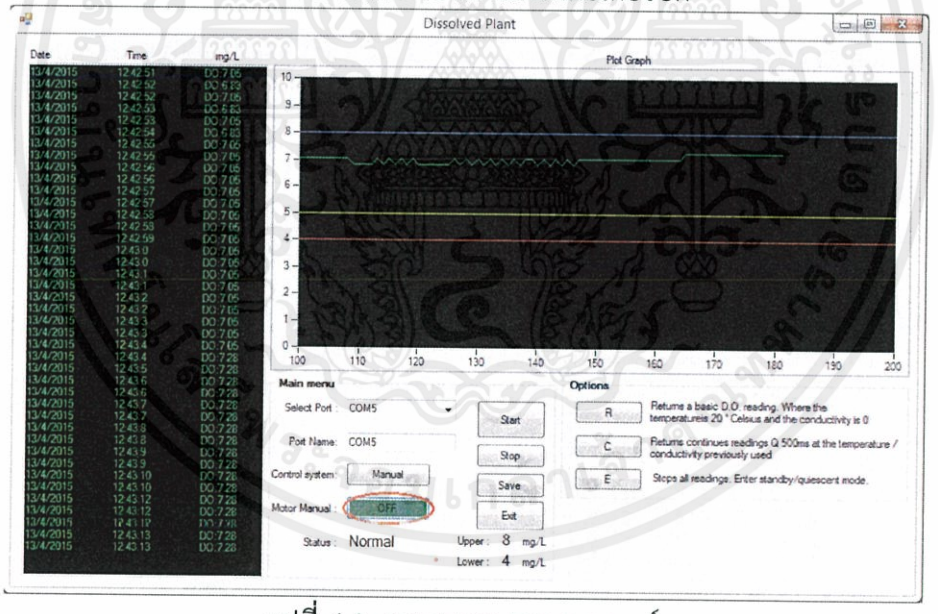
รูปที่ 4.6 แสดงข้อมูลต่างๆเมื่อโปรแกรมเริ่มทำงานอ่านค่าเข้ามา

7. ต่อไปให้ลองทำการควบคุมเปิดมอเตอร์ ในโหมด Manual โดยการคลิ้กที่ปุ่ม ON



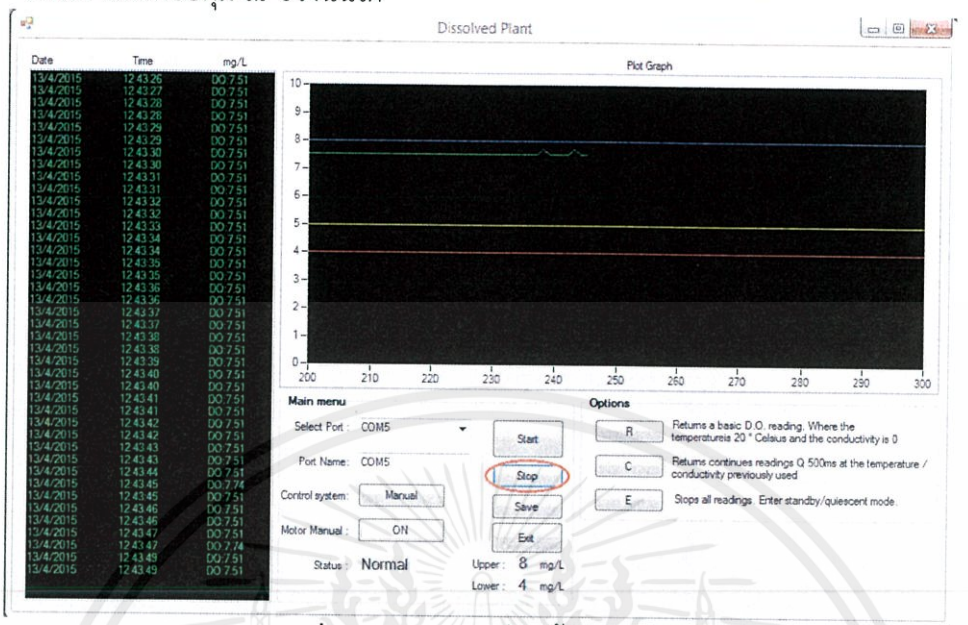
รูปที่ 4.7 แสดงการควบคุมเปิดมอเตอร์ในโหมด Manual

8. เมื่อคลิ้กแล้วปุ่มนั้นแล้วมอเตอร์ก็จะเริ่มทำงาน ปุ่มจะเป็นสีเขียวและขึ้นข้อความบนปุ่มว่า “OFF” เพื่อว่าเมื่อครั้งต่อไปที่คลิ้กจะเป็นการสั่งให้มอเตอร์ปิด



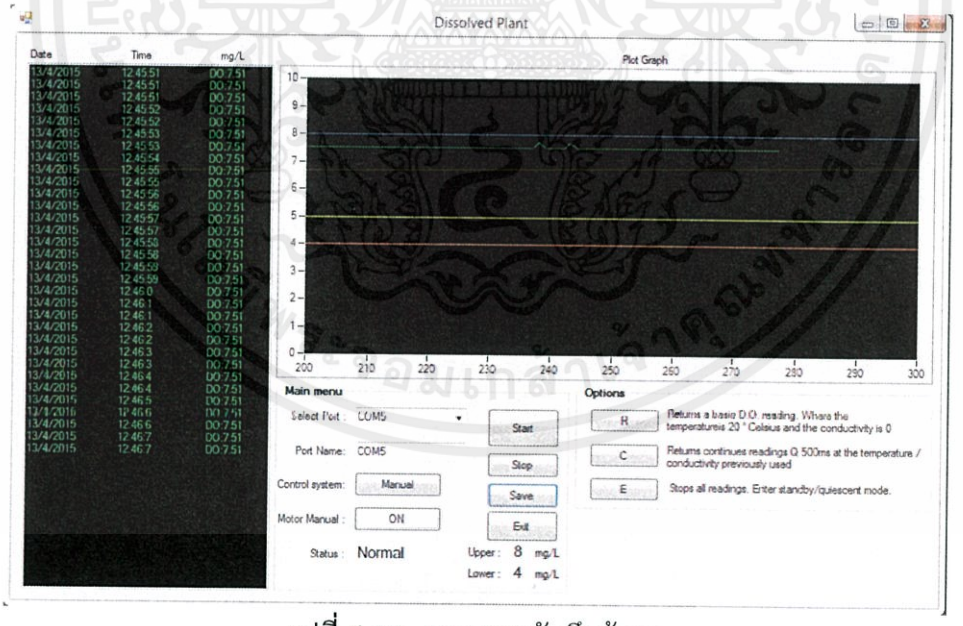
รูปที่ 4.8 แสดงสถานะของมอเตอร์

9. การคลิ้กปุ่ม Stop จะทำให้พอร์ตที่ติดต่อรับส่งข้อมูลนั้นถูกตัดขาดทันที ทำให้ไม่สามารถอ่านค่าและควบคุม ณ ช่วงนั้นได้



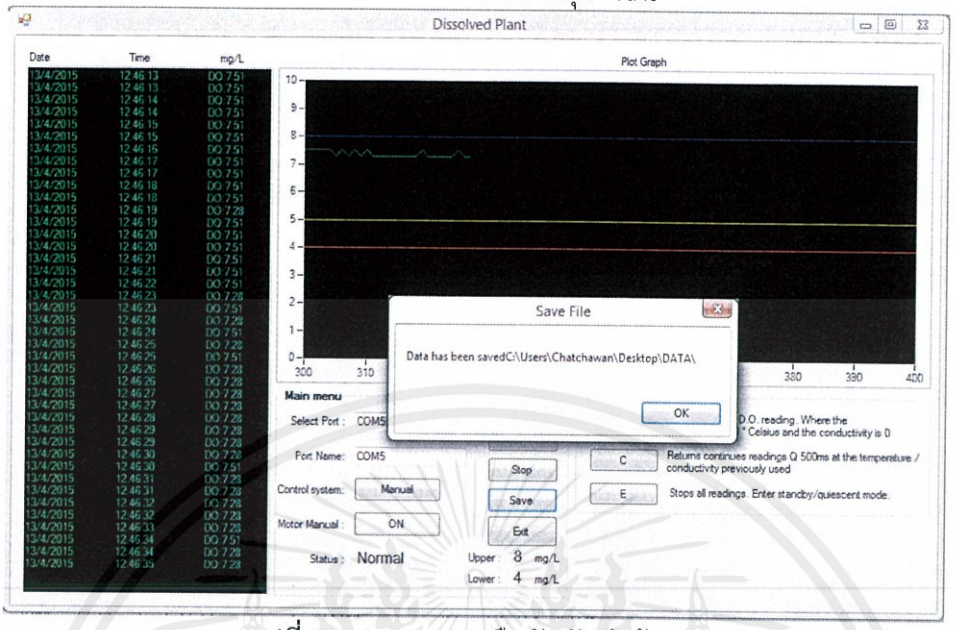
รูปที่ 4.9 แสดงผลเมื่อคลิ้กปุ่ม Stop

10. การบันทึกข้อมูลทำได้โดยการคลิ้กปุ่ม Save จะบันทึกข้อมูลเฉพาะ วัน เวลา และค่าออกซิเจนละลายน้ำเท่านั้น



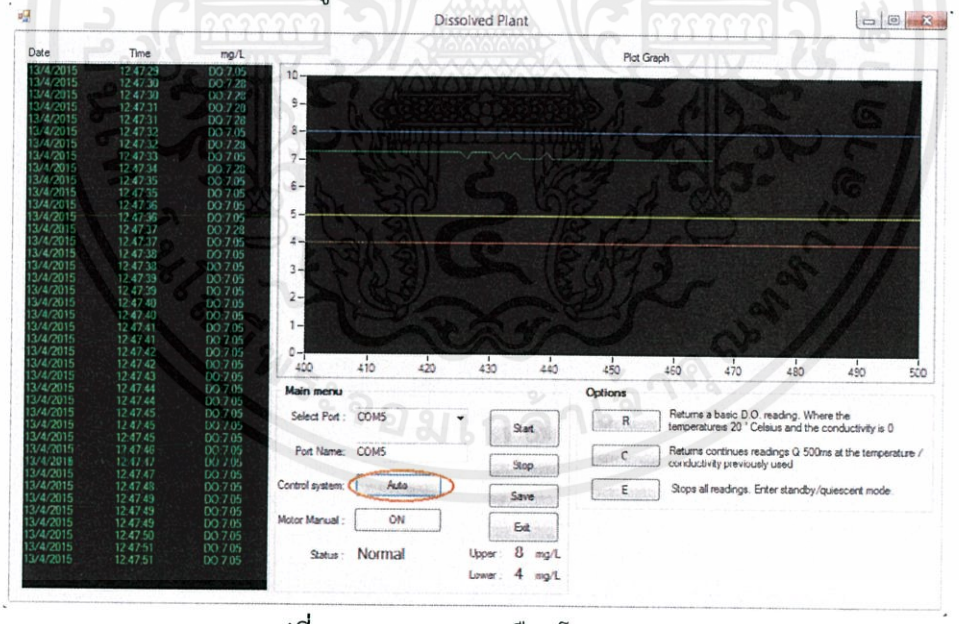
รูปที่ 4.10 แสดงการบันทึกข้อมูล

11. เมื่อคลิ้กที่ปุ่ม Save แล้วจะมีกล่องข้อความขึ้นมา เพื่อบอกว่าไฟล์นั้นได้บันทึกไว้แล้วและได้บันทึกไว้ที่ไหน โดยไฟล์ที่บันทึกนั้นเป็นไฟล์นามสกุล *.txt



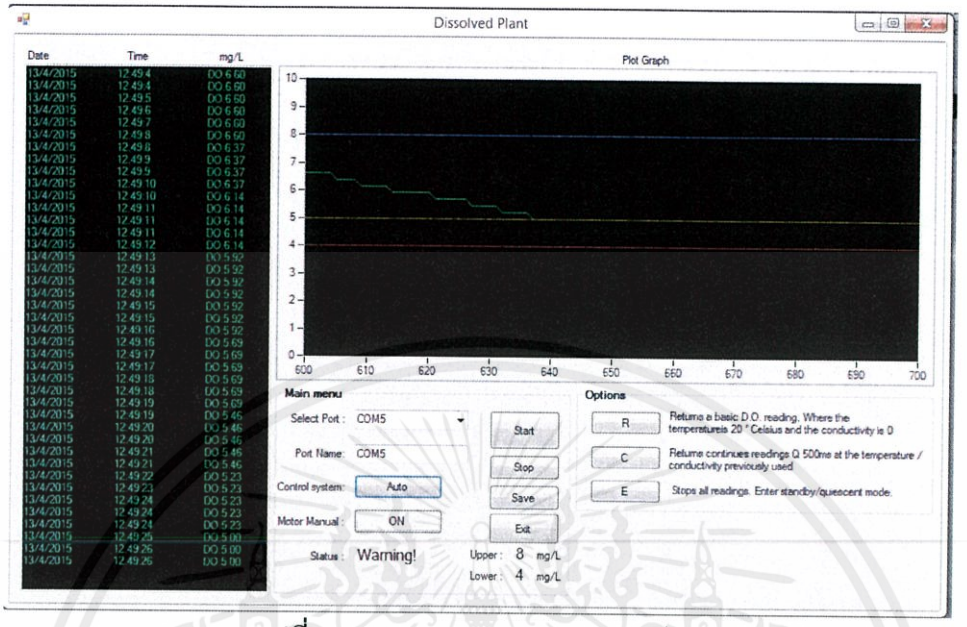
รูปที่ 4.11 แสดงการยืนยันบันทึกข้อมูล

12. ต่อไปลองทำการควบคุมในโหมด Auto โดยการไปคลิ้กปุ่ม Manual จะทำให้ปุ่มนั้นเปลี่ยนเป็นโหมด Auto ดังรูปที่ 4.12



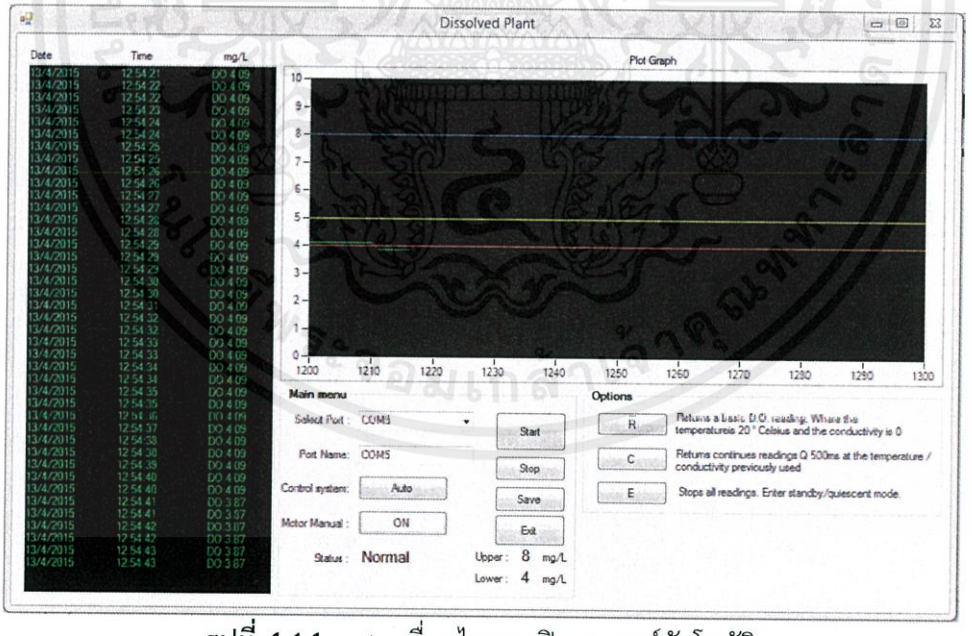
รูปที่ 4.12 แสดงการเลือกโหมด Auto

13. จะเห็นว่าในรูปที่ 4.13 ค่าออกซิเจนละลายน้ำได้มีแนวโน้มลดลงเรื่อยๆ จนมาถึงเส้นสีเหลือง เส้นนี้เรียกว่าเส้น Alarm โปรแกรมจะทำการแจ้งเตือนผู้สังเกตกาณ์โดยเปิดเสียง Alarm อัตโนมัติ



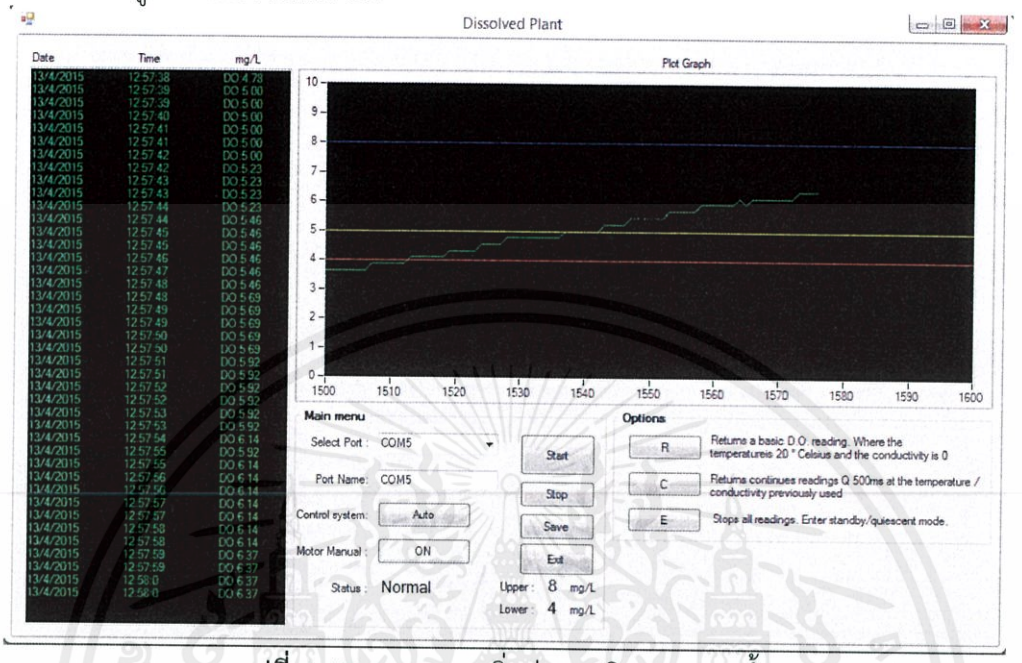
รูปที่ 4.13 แสดงการ Alarm แบบเสียง

14. เมื่อค่าออกซิเจนยังคงลดต่อเนื่องจนถึงเส้นสีแดง เรียกว่าเส้น Lower โปรแกรมจะทำการปิดเสียง Alarm และทำการสั่งเปิดมอเตอร์อัตโนมัติ



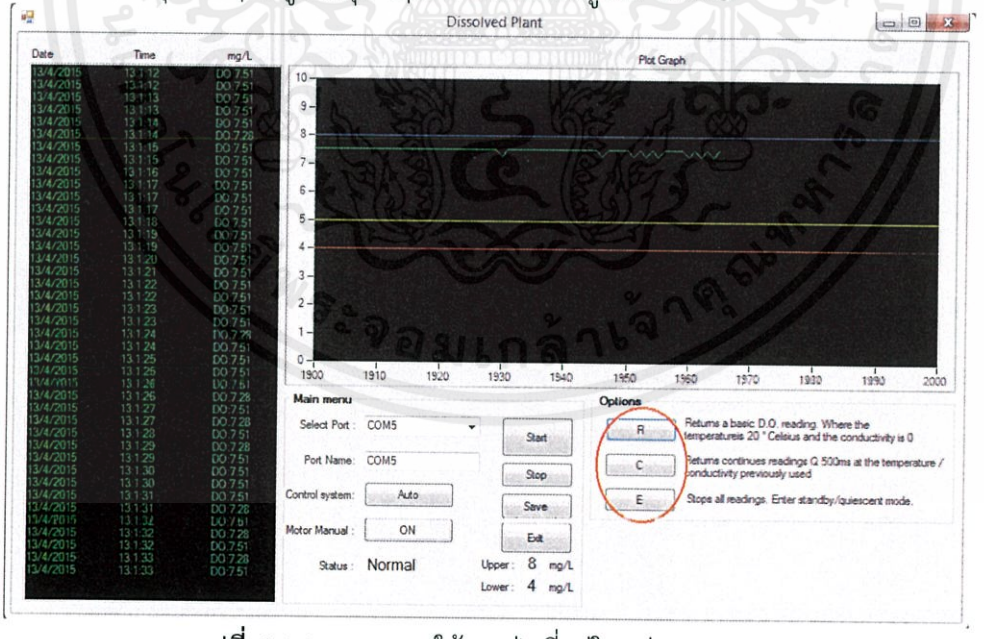
รูปที่ 4.14 แสดงเงื่อนไขการเปิดมอเตอร์อัตโนมัติ

15. จากรูปที่ 4.14 เมื่อมอเตอร์ทำการเปิดก็จะทำให้ออกซิเจนละลายน้ำเพิ่มขึ้นเรื่อยๆ และเมื่อค่าออกซิเจนละลายน้ำสูงกว่าเส้น Lower โปรแกรมก็จะทำการปิดมอเตอร์ และ เปิดเสียง Alarm ให้ทราบว่าออกซิเจนละลายน้ำยังอยู่ในเกณฑ์น้อยอยู่ ใช้เวลาสักพักออกซิเจนละลายน้ำก็จะสูงกว่าเส้น Alarm เอง



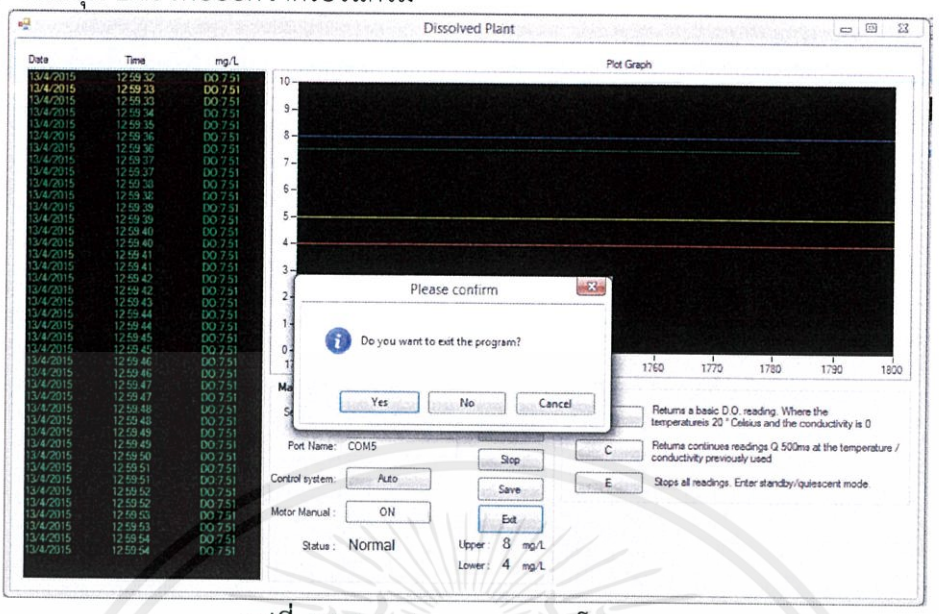
รูปที่ 4.15 แสดงการเพิ่มค่าออกซิเจนละลายน้ำ

16. การใช้งานปุ่มต่างๆที่อยู่ในกลุ่ม Options สามารถดูได้ที่ บทที่ 3



รูปที่ 4.16 แสดงการใช้งานปุ่มที่อยู่ในกลุ่ม Options

17. คลิ้กปุ่ม Exit เพื่อออกจากโปรแกรม



รูปที่ 4.17 แสดงการออกจากโปรแกรม

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผล

ผลจากการทดลองการวัดออกซิเจนละลายน้ำเพื่อควบคุมมอเตอร์ใบพัดตีน้ำในบ่อกุ้งนั้น พบว่าการควบคุมแบบอัตโนมัติให้ความสะดวกสบายแก่เจ้าของบ่อกุ้งเนื่องจากการใช้คอมพิวเตอร์ในการควบคุมทำให้ไม่ต้องจ้างแรงงานดูแลในเรื่องนี้ และข้อมูลที่ได้ยังสามารถนำมาวิเคราะห์ได้ว่าช่วงไหนของวันออกซิเจนละลายน้ำน้อย หรือมากเพียงพอสำหรับในเวลานั้นๆ ทำให้ประหยัดพลังงานไฟฟ้า ตรงกันข้ามกับการเปิดมอเตอร์ในระบบที่ไม่มีการวัดออกซิเจนละลายน้ำมาควบคุม ทำให้ต้องเปิดอยู่ตลอดเวลาหรือเปิดมากเกินไปจนออกซิเจนละลายน้ำเกินความจำเป็นนั้นเอง

5.2 ข้อเสนอแนะ

ในโครงการนี้ได้ทำการออกแบบจำลองบ่อกุ้งเพียงบ่อเดียว แต่ในความเป็นจริงแล้วเจ้าของบ่อกุ้งนั้นมีบ่อกุ้งเป็นของตัวเองมากกว่าหนึ่งบ่อ หากแต่เซนเซอร์ที่ใช้ตรวจวัดนั้นมีเพียงชิ้นเดียวจึงทำให้ไม่สามารถเขียนโปรแกรมให้รองรับการวัดค่าหลายๆบ่อได้เป็นรูปธรรม เนื่องจากเซนเซอร์วัดออกซิเจนละลายน้ำหายากมากและมีราคาค่อนข้างแพง ตัวเลือกน้อย การใช้งานค่อนข้างละเอียดอ่อนและต้องการการดูแลเป็นอย่างมาก เพราะหาก โพรบที่ใช้ตรวจวัดนั้นแห้งทำให้เซนเซอร์เสียทันที จึงเป็นข้อจำกัดอย่างหนึ่งในการออกแบบจำลองระบบควบคุมนี้ไปใช้งานในสถานที่จริง

บรรณานุกรม

ศุภชัย สมภานิช, **Professional Data base Programing VB 2010 & VC# 2010.** นนทบุรี : ไอดีซีฯ, 2553

กิตินันท์ พลสวัสดิ์, **เริ่มต้น Visual Basic 2008 ฉบับโปรแกรมเมอร์** นนทบุรี : ไอดีซีฯ, 2552

พฤทธินันท์ ลิ่มวรวิวัฒน์, พัชรินทร์ อินโสม และ ภาษิต ภัทรากุล **"ระบบเซนเซอร์ไร้สายตรวจวัดคุณภาพน้ำ"** วิทยานิพนธ์วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชา วิศวกรรมโทรคมนาคม บัณฑิตวิทยาลัย, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2551

ธีระศักดิ์ สุโชตินันท์, ประยุทธ อินแบน **โปรแกรมเมอร์ มือใหม่ หัดเขียนโปรแกรม Visual Basic 6.0** กรุงเทพฯ : สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย 2549

เอกชัย มะการ, **เรียนรู้และเข้าใจใช้งานไมโครคอนโทรลเลอร์ตระกูล AVR ด้วย Arduino** กรุงเทพฯ : บริษัท อีทีที จำกัด. 2552

Microsoft Visual Studio Professional 2012 .[Online]. Available https://msdn.microsoft.com/

Measurement Studio 2013 .[Online]. Available http://www.ni.com/mstudio/

Aduino IDE.[Online]. Available http://arduino.cc/en/Main/Software

Dissolved Oxygen Data Sheet .[Online]. Available http://atlas-scientific.com/_files/_datasheets/_circuit/DO_EZO_Datasheet.pdf?

ThaiEasyElec **Relay Data Sheet**

http://www.thaieasyelec.net/archives/Manual/EFDV237/EFDV237_2_Channels_Relay_Quick_Start_Guide.pdf

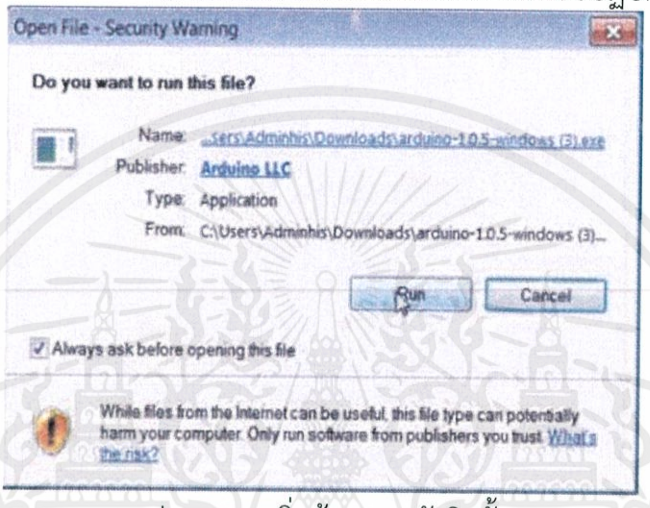
ภาคผนวก

การติดตั้งโปรแกรม

Arduino 1.0.6 Installing

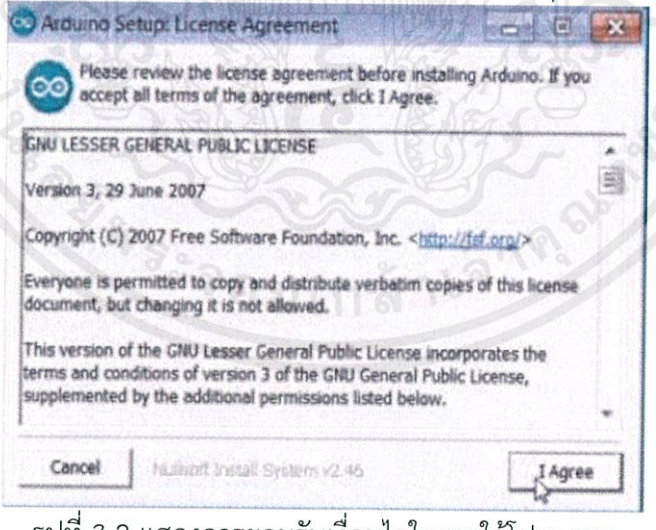
มีขั้นตอนดังนี้

1. เข้าไปดาวน์โหลดโปรแกรมได้ที่ http://arduino.cc/en/Main/Software
เลือกตัวโปรแกรมให้เหมาะสมกับระบบปฏิบัติการ Run>



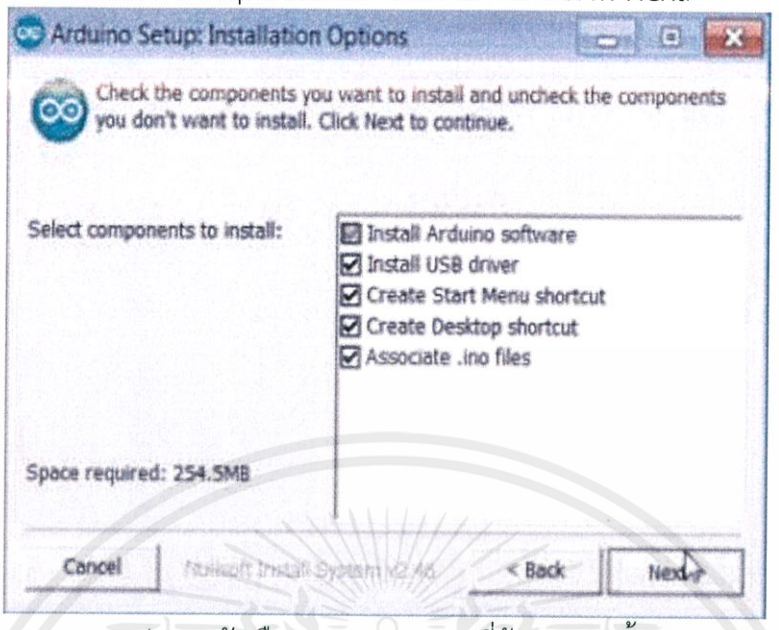
รูปแสดงการเริ่มต้น Run ตัวติดตั้ง

2. กด I Agree> เพื่อยอมรับเงื่อนไขต่างๆ



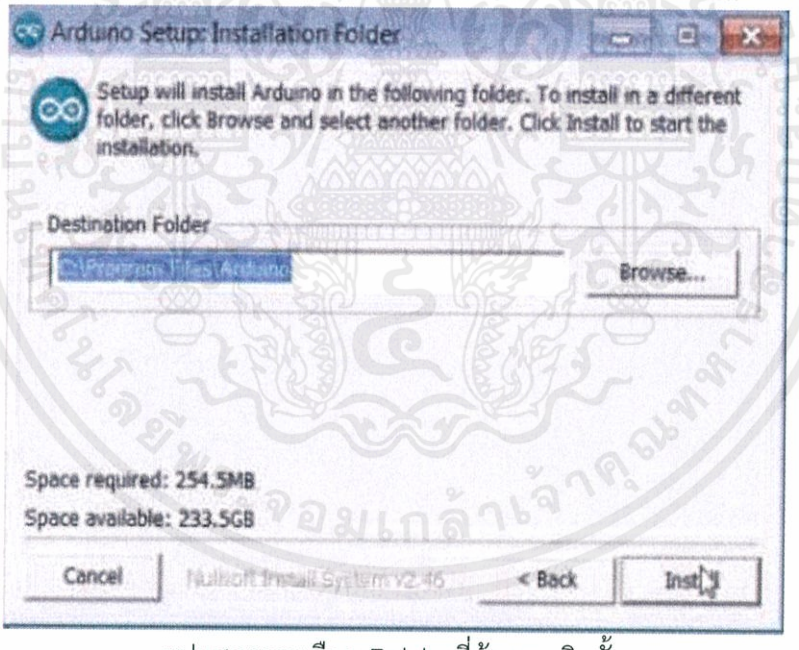
รูปที่ 3.2 แสดงการยอมรับเงื่อนไขในการใช้โปรแกรม

3. เลือก Component ที่ต้องใช้งานการตามภาพ Next>



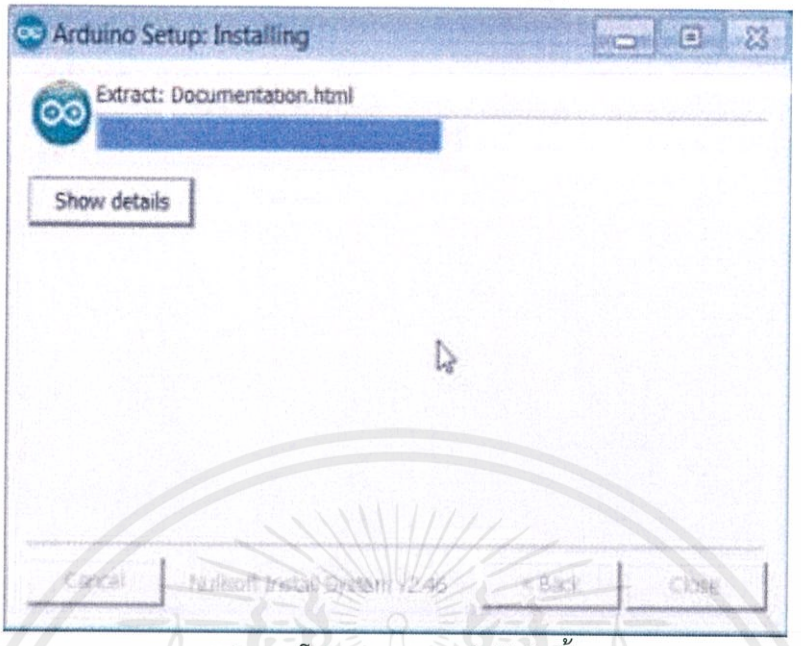
รูปแสดงตัวเลือก Component ที่ต้องการติดตั้ง

4. เลือก Destination Folder ที่ต้องการติดตั้ง จากนั้น Install>



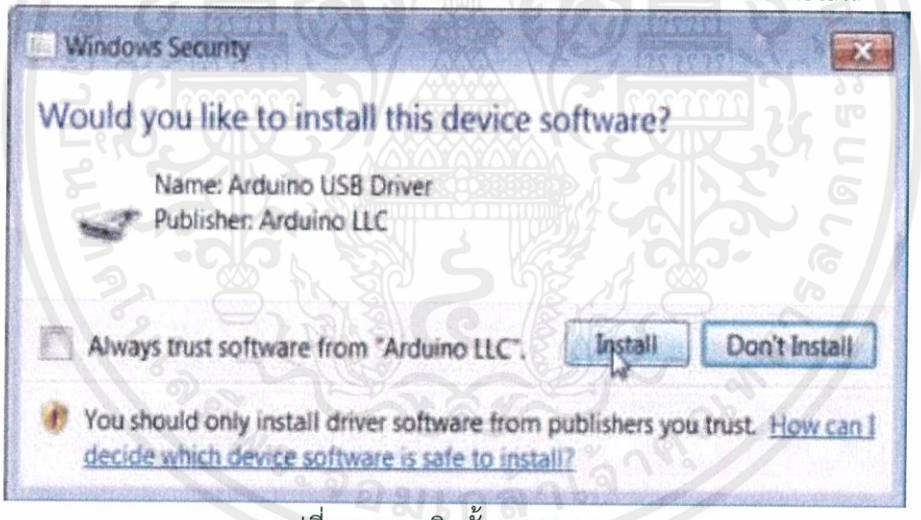
รูปแสดงการเลือก Folderที่ต้องการติดตั้ง

5. โปรมแกรมเริ่มทำการติดตั้ง



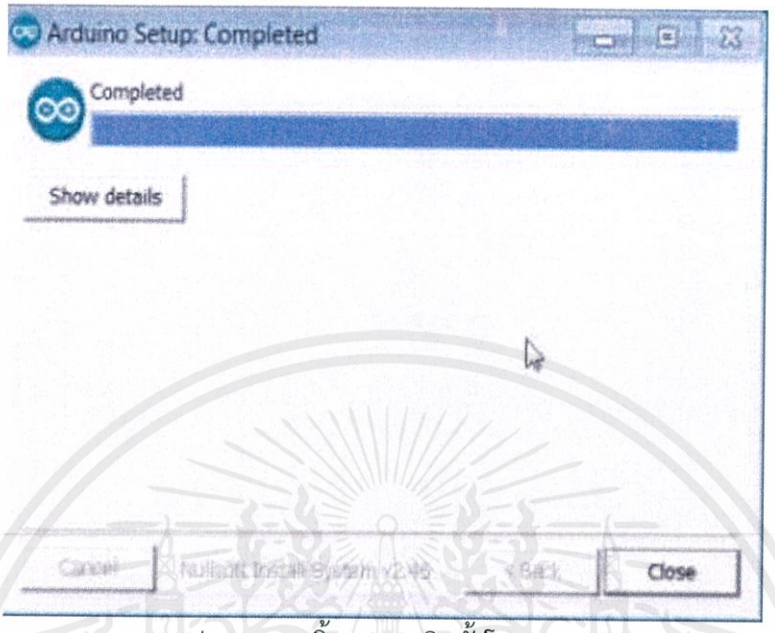
รูปแสดงโปรแกรมขณะกำลังติดตั้ง

6. ขณะติดตั้งจะมีหน้าต่างเด้งขึ้นมาถาม ให้ติดตั้ง USB driver กด Install>



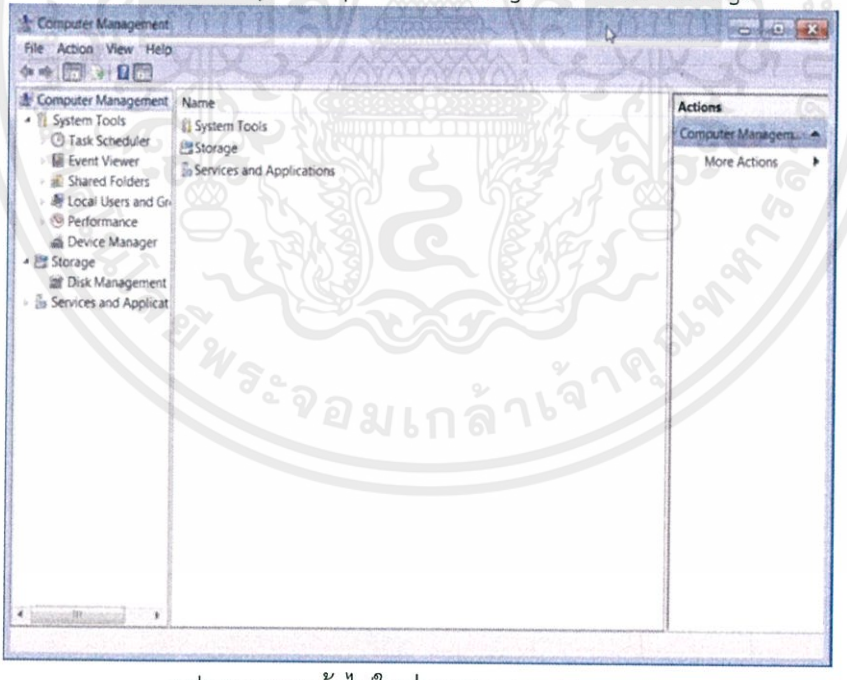
รูปที่แสดงการติดตั้ง USB driver

7. เมื่อติดตั้งเสร็จสิ้นกด Close> แล้วลองเข้าโปรแกรมดู เป็นอันเสร็จสินในการติดตั้ง แต่หากพบว่าโปรแกรมนั้นไม่สามารถหาพอร์ตของบอร์ด Arduino เจอให้ทำการติดตั้ง Port Driver ตามขั้นตอนต่อไป



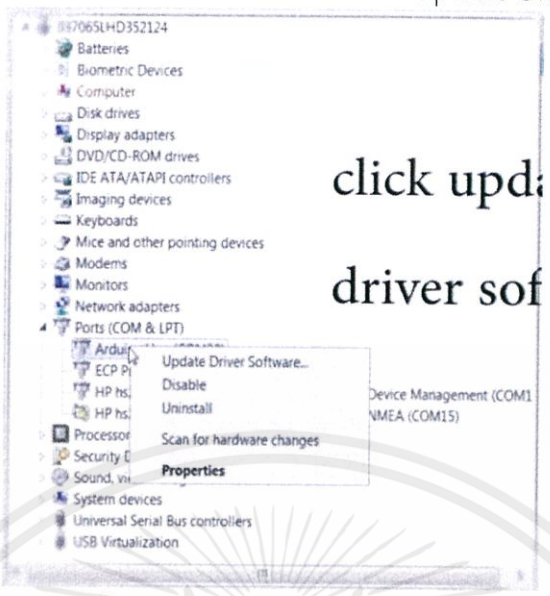
รูปแสดงการสิ้นสุดการติดตั้งโปรแกรม

8. คลิกขวาที่ My Computer> Manage> Device Manager



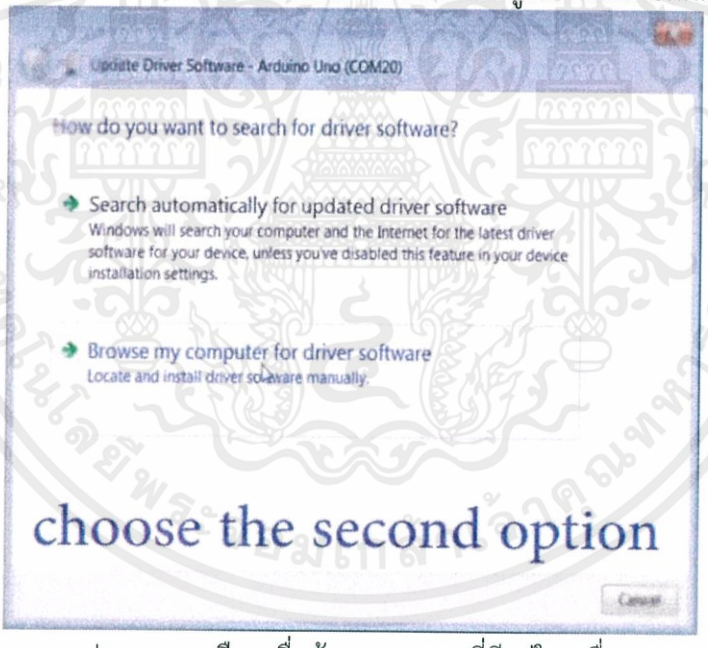
รูปแสดงการเข้าไปในส่วน Device Manager

9. ไปที่ Ports คลิกขวาที่ Port ของ Arduino เลือก Update Driver Software



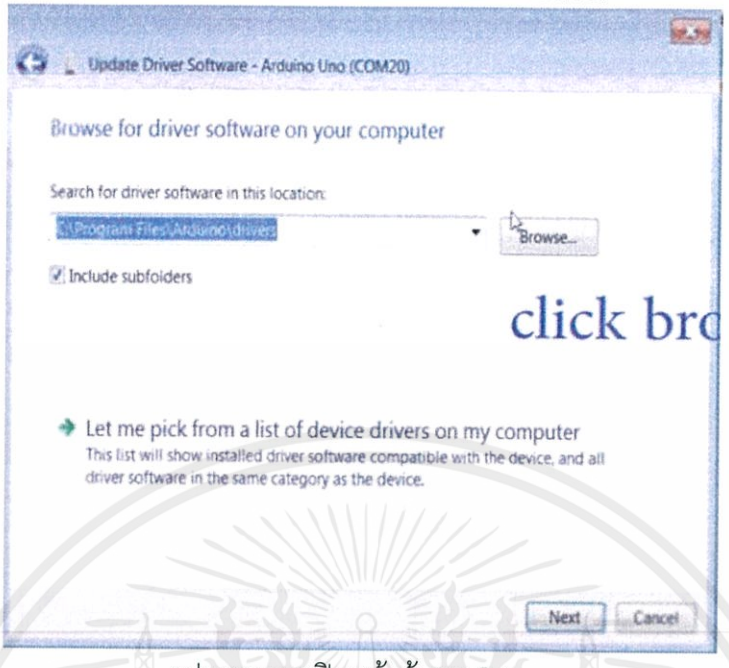
แสดงการเลือก Update Driver ของ Arduino Port

10. เลือกตัวเลือกที่สอง เพื่อค้นหา Driver ที่มีอยู่ในเครื่องเพื่อติดตั้ง



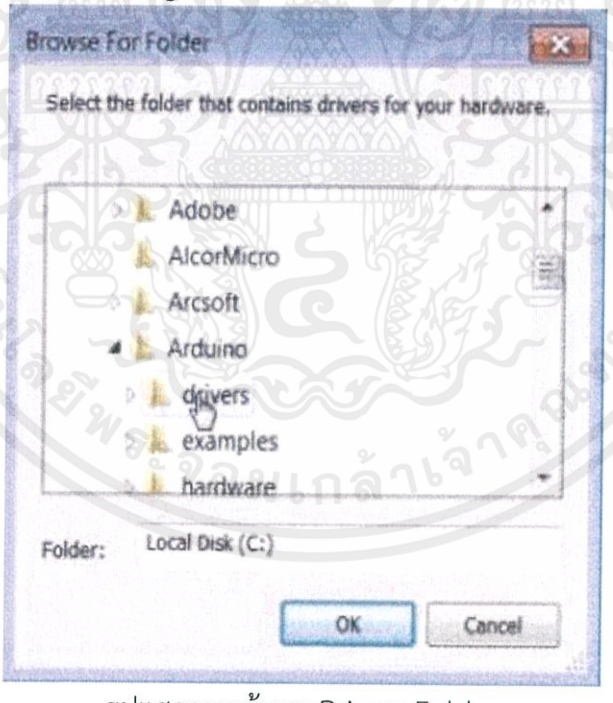
รูปแสดงการเลือกเพื่อค้นหา Driver ที่มีอยู่ในเครื่อง

11. คลิกที่ Brose เปิดหน้าต่างค้นหา Driver



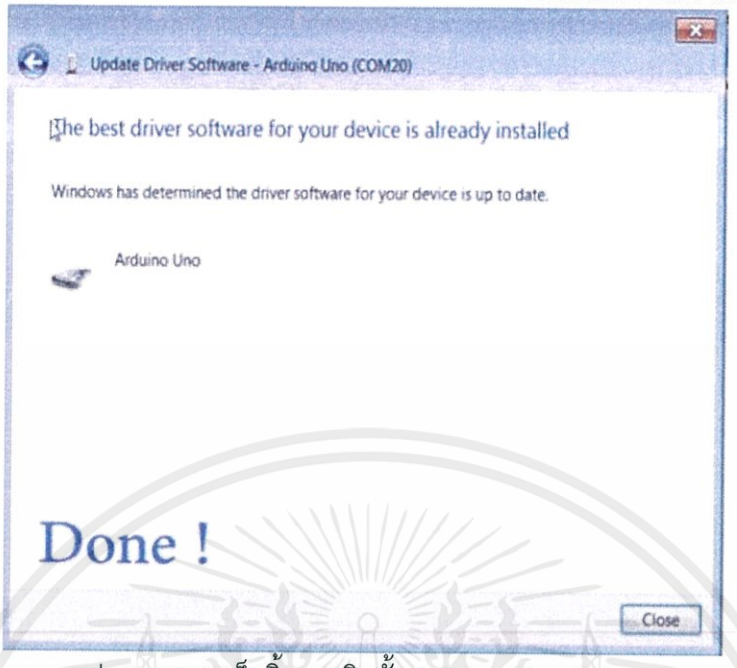
รูปแสดงการเปิดหน้าค้นหา Driver

12. ไปที่ Local disk> Program Files> Arduino > Drivers Folder และ OK>



รูปแสดงการค้นหา Drivers Folder

13. จากนั้นจะกลับมาที่หน้าต่างค้นหา Driver กด Next> เป็นอันติดตั้งเสร็จสิ้น Close>

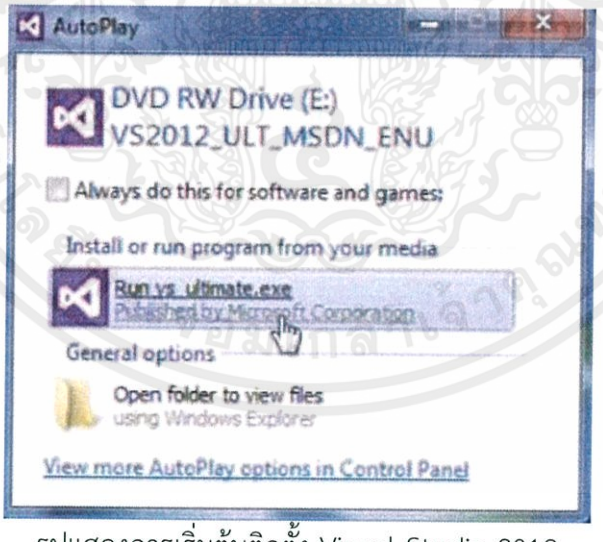


รูปแสดงการเสร็จสิ้นการติดตั้ง Aduino Port Driver

Visual Studio 2012 Installing

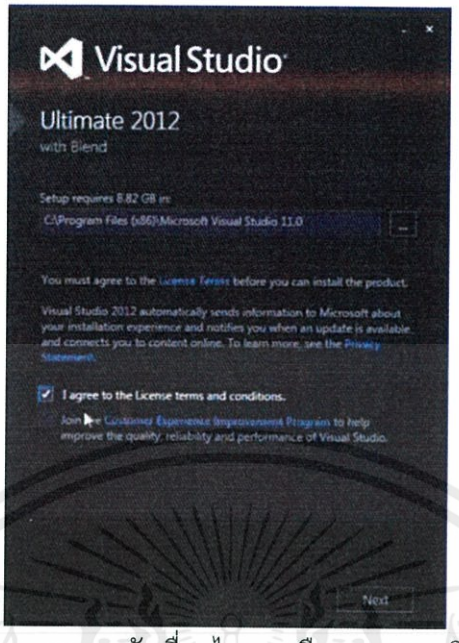
มีขั้นตอนดังนี้

1. ใส่แผ่น แล้วรอ Auto run ทำงาน จากนั้นกด Run vs ultimate.exe



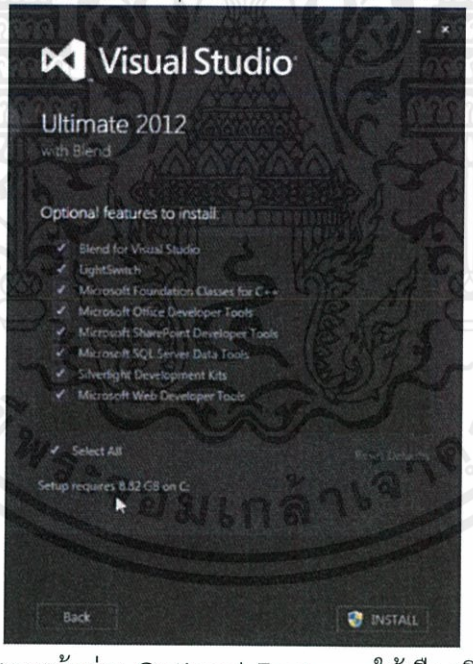
รูปแสดงการเริ่มต้นติดตั้ง Visual Studio 2012

2. กดยอมรับเงื่อนไขการใช้โปรแกรม และสามารถเลือก Folder ที่จะติดตั้งโปรแกรม Next>



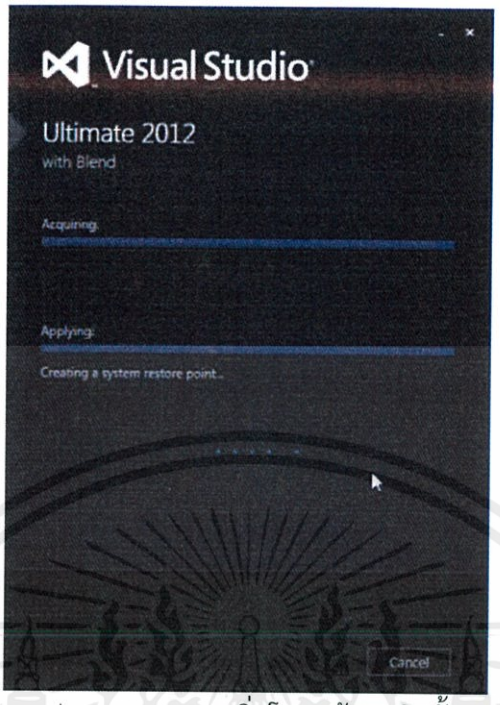
รูปแสดงการยอมรับเงื่อนไขและเลือก Folder ติดตั้ง

3. เลือกติดตั้ง Option ทั้งหมด กด Install>



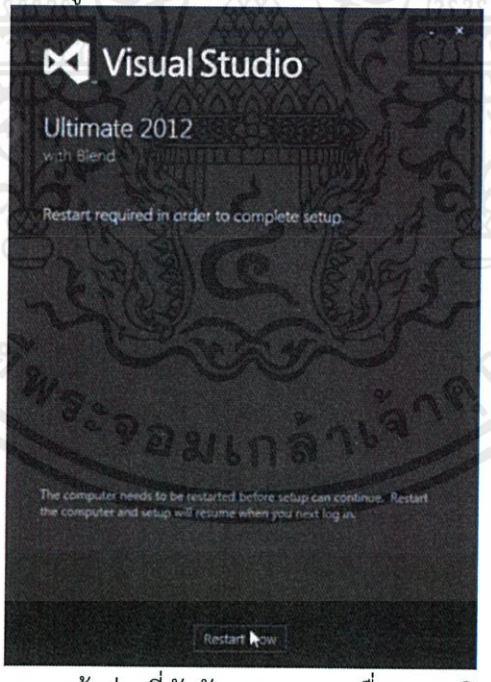
รูปแสดงหน้าต่าง Optional Features ให้เลือกติดตั้ง

4. เริ่มโหลดทำการติดตั้ง



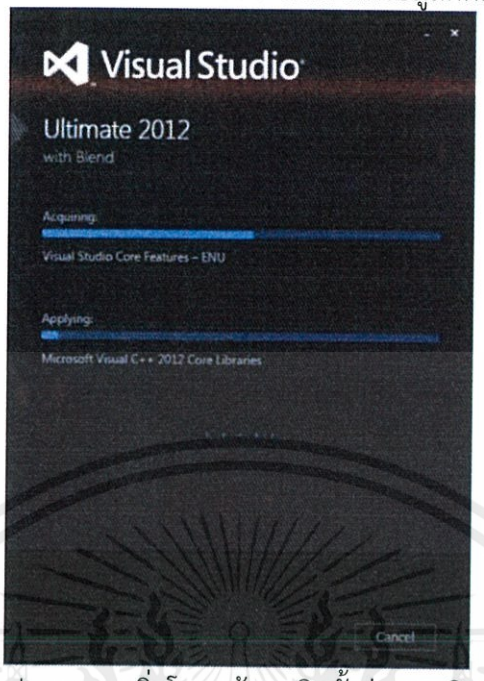
รูปแสดงภาพขณะเริ่มโหลดข้อมูลติดตั้ง

5. ขณะเริ่มโหลดข้อมูลติดตั้งจะมีหน้าต่างเด้งขึ้นมาให้กด Restart Now>



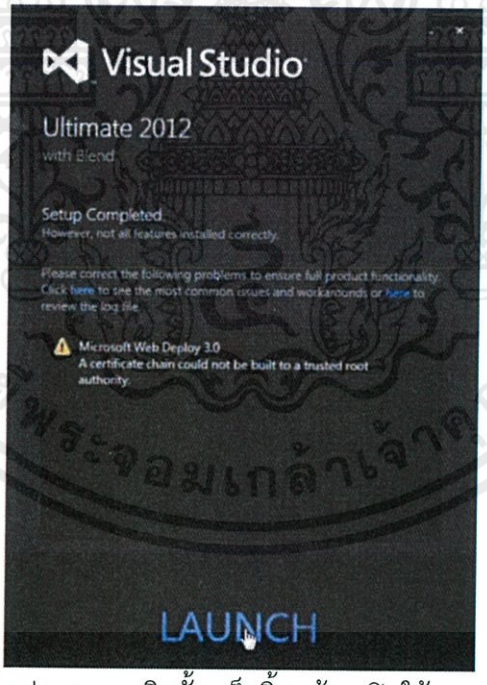
รูปแสดงหน้าต่างที่บังคับ Restart เครื่องขณะติดตั้ง

6. เมื่อเครื่อง Restart เสร็จจะเริ่มโหลดข้อมูลติดตั้งต่อ



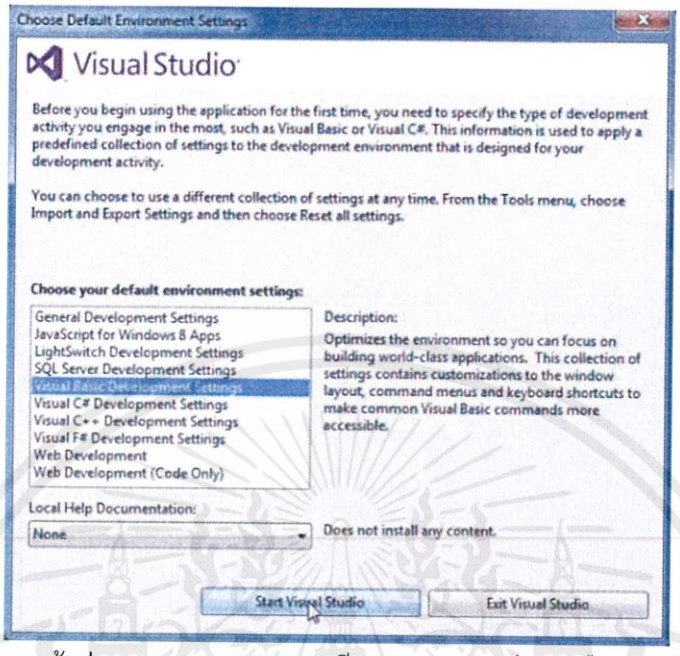
รูปแสดงการเริ่มโหลดข้อมูลติดตั้งต่อจากเดิม

7. เมื่อติดตั้งเสร็จจะขึ้นหน้าต่าง ถามว่าจะเปิดโปรแกรมเลยไหม กด LAUNCH> เพื่อทดสอบใช้งาน



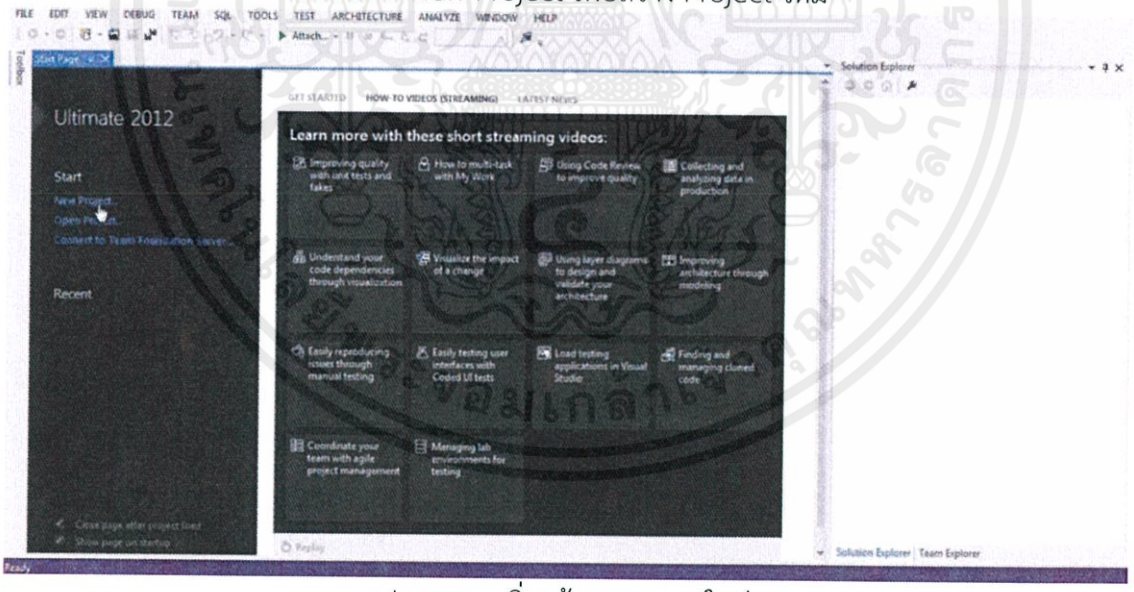
รูปแสดงการติดตั้งเสร็จสิ้นพร้อมเปิดใช้งาน

8. จะมีหน้าต่างขึ้นมาถามว่า จะเลือกใช้ระบบการเขียนแบบไหนให้เลือก Visual Basic และทำการเลือก None เมื่อเราไม่ต้องการความช่วยเหลือตอนเริ่มต้น



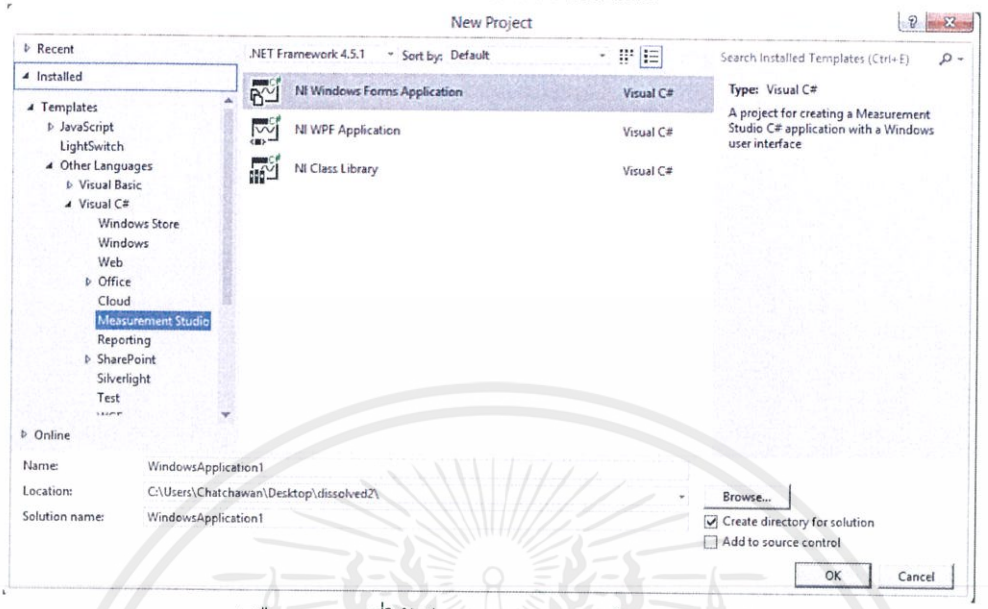
รูปแสดงหน้าต่างสอบถามระบบการเขียน และความช่วยเหลือตอนเริ่มต้น

9. คลิกที่ New Project เพื่อสร้าง Project ใหม่



รูปแสดงการเริ่มสร้าง Project ใหม่

10. เลือก Visual c#> Measurement Studio > NI Windows Form Application เริ่มทำงานได้เลย



รูปเลือกภาษาที่ใช้เขียน และ Form ในการทำงาน

โปรแกรม Ardiuno

```
#include <SoftwareSerial.h>      //we have to include the SoftwareSerial library, or else we
can't use it.
#define rx 2                     //define what pin rx is going to be.
#define tx 3                     //define what pin Tx is going to be.
SoftwareSerial myserial(rx, tx); //define how the soft serial port is going to work.
char DO_data[20];               //we make a 20 byte character array to hold incoming data
from the D.O.
char computerdata[20];           //we make a 20 byte character array to hold incoming data
from a pc/mac/other.
byte received_from_computer=0;    //we need to know how many characters have been
received.
byte received_from_sensor=0;       //we need to know how many characters have been
received.
byte arduino_only=0;               //if you would like to operate the D.O. Circuit with the
Arduino
                                    only and not use the serial monitor to send it
                                  commands set this
                                    to 1. The data will still come out on the serial monitor,
                                  so you can
                                    see it working.
byte startup=0;                   //used to make sure the Arduino takes over control of
the D.O.
                                   Circuit properly.
byte string_received=0;          //used to identify when we have received a string from the
D.O. circuit.
float DO_float=0;                //used to hold a floating point number that is the D.O.
float sat_float=0;               //used to hold a floating point number that is the percent
saturation.
char *DO;                       //char pointer used in string parsing
char *sat;                      //char pointer used in string parsing
int incomingByte;              //number pointer used in integer parsing


void setup(){
    Serial.begin(38400);          //enable the hardware serial port
    myserial.begin(38400);        //enable the software serial port
    pinMode(8, OUTPUT);           //define port 8 is output port
          }
void serialEvent(){              //this interrupt will trigger when the data coming from the
serial monitor(pc/mac/other) is received.
```

```
        if(arduino_only!=1){       //if Arduino_only does not equal 1 this function will be bypassed.
          received_from_computer=Serial.readBytesUntil(13,computerdata,20); //we read the data sent from the serial monitor(pc/mac/other) until we see a <CR>. We also count how many characters have been received.
          computerdata[received_from_computer]=0; //we add a 0 to the spot in the array just after the last character we received. This will stop us from transmitting incorrect data that may have been left in the buffer.
          myserial.print(computerdata);          //we transmit the data received from the serial monitor(pc/mac/other) through the soft serial port to the D.O. Circuit.
          myserial.print('\r');                  //all data sent to the D.O. Circuit must end with a <CR>.
        }
       }
void loop(){
      if (Serial.available() > 0) {
       incomingByte = Serial.read();
      if (incomingByte == 'H')
      {
      digitalWrite(8, HIGH);
       }
      if (incomingByte == 'L')
 {
      digitalWrite(8, LOW); }
 }
 if(myserial.available() > 0){        //if we see that the D.O. Circuit has sent a character.
    received_from_sensor=myserial.readBytesUntil(13,DO_data,20); //we read the data sent from D.O. Circuit until we see a <CR>. We also count how many character have been received.
    DO_data[received_from_sensor]=0;  //we add a 0 to the spot in the array just after the last character we received. This will stop us from transmitting incorrect data that may have been left in the buffer.
    string_received=1;                //a flag used when the Arduino is controlling the D.O. Circuit to let us know that a complete string has been received.

    if((DO_data[0] >= 48) && (DO_data[0] <=57)){   //if DO_data[0] is a digit and not a letter
      pars_data();
      }
    else     Serial.println(DO_data);          //if the data from the D.O. circuit does not start with a number transmit that data to the serial monitor.
      }
}
void pars_data()
{
```

```
    byte i;
  byte pars_flag=0;
    for(i=0;i<=received_from_sensor;i++)
      {
        if(DO_data[i]==','){pars_flag=1;}
      }
    if(pars_flag){
    DO=strtok(DO_data, ",");          //let's pars the string at each comma.
  sat=strtok(NULL, ",");            //let's pars the string at each comma.
     Serial.print("DO:");             //We now print each value we parsed sepratly.
  Serial.println(DO);               //this is the DO value.
  //DO_float=atoi(DO);                //Uncomment this line to turn the string into to floating
pint value.
     Serial.print("%sat:");           //We now print each value we parsed sepratly.
  Serial.println(sat);              //this is the TDS value.
  //sat_float=atoi(sat);              //Uncomment this line to turn the string into to floating
pint value.
  }
  else                              //if the output is ony DO and not DO + % sat
  {
   Serial.print("DO:");             //print out "DO:"
   Serial.println(DO_data);          //printout that DO in Mg/L
  }
}
```

โปรแกรม Visual Studio 2012

```
using NationalInstruments.Net;
using NationalInstruments.Analysis;
using NationalInstruments.Analysis.Conversion;
using NationalInstruments.Analysis.Dsp;
using NationalInstruments.Analysis.Dsp.Filters;
using NationalInstruments.Analysis.Math;
using NationalInstruments.Analysis.Monitoring;
using NationalInstruments.Analysis.SignalGeneration;
using NationalInstruments.Analysis.SpectralMeasurements;
using NationalInstruments;
using NationalInstruments.UI;
using NationalInstruments.DAQmx;
using NationalInstruments.NetworkVariable;
using NationalInstruments.NetworkVariable.WindowsForms;
using NationalInstruments.Tdms;
using NationalInstruments.UI.WindowsForms;
using NationalInstruments.Controls;
using NationalInstruments.Controls.Rendering;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;
using System.Media;
namespace dissolvedOxygen
{
    public partial class Form1 : Form
    {
        private SerialPort myport;
        private DateTime datetime;
        private string in_data;
        string specifier;
        float value_O2 = 0;
        int relayON = 0 ;
        int relayOFF = 0;
        private SoundPlayer _soundPlayer;
```

```
        public Form1()
          {
           InitializeComponent();
           _soundPlayer = new SoundPlayer("Alert3.wav");
          }
        private void Form1_Load(object sender, EventArgs e)
          {
           string[] list = SerialPort.GetPortNames();
           foreach (string port in list)
           {
               comboBox1.Items.Add(port);
           }
}
        private void start_btn_Click(object sender, EventArgs e)
        {
           myport = new SerialPort();
           myport.BaudRate = 38400;
           myport.PortName = port_name_tb.Text;
           myport.Parity = Parity.None;
           myport.DataBits = 8;
           myport.StopBits = StopBits.One;
           myport.DataReceived += myport_DataReceived;
           try
           {   myport.Open();
               data_tb.Text = "";
           }
           catch (Exception ex)
           {
              MessageBox.Show(ex.Message, "Error");
           }
        }
        private void myport_DataReceived(object sender, SerialDataReceivedEventArgs e)
        {
           in_data = myport.ReadLine();
           this.Invoke(new EventHandler(displaydata_event));
          }

        private void displaydata_event(object sender, EventArgs e)
        {
           datetime = DateTime.Now;
           string time = datetime.Day + "/" + datetime.Month + "/" + datetime.Year + "\t\t" +
    datetime.Hour + ":"
           + datetime.Minute + ":" + datetime.Second;
           data_tb.AppendText(time + "\t\t" + in_data + "\n");
```

```
            if (in_data.Length > 0)
            {
                value_O2 = float.Parse(in_data.Substring(3, 4));
                waveformGraph1.PlotYAppend(value_O2);
            }
    if (sys_btn.Text == "Auto")
                {
                  if (value_O2 < Lower.YPosition)
                                {
                                    myport.Write("H");
                                }
                                else
                                {
                                    myport.Write("L");
                                }
                  if (value_O2 <= Alarm.YPosition && value_O2 >= Lower.YPosition)
                                {
                                    _soundPlayer.Play();
                                    label19.Text = "Warning!";
                                }
                                else
                                {  _soundPlayer.Stop();
                                    label19.Text = "Normal";
                                }
                }
        }

        private void exit_btn_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Do you want to exit the program?", "Please confirm",
MessageBoxButtons.YesNoCancel, MessageBoxIcon.Information) == DialogResult.Yes)
            {
                Application.Exit();
            }
        }
        private void stop_btn_Click(object sender, EventArgs e)
        {
            try
            {
            myport.Close();
            }
```

```
    catch (Exception ex2)
    {
        MessageBox.Show(ex2.Message, "Error");
    }
}
private void save_btn_Click(object sender, EventArgs e)
{
    try


    {   string pathfile = @"C:\Users\Chatchawan\Desktop\DATA\";
        string filename = "Dissolved_Data.txt";
        System.IO.File.WriteAllText(pathfile + filename, data_tb.Text);
        MessageBox.Show("Data has been saved" + pathfile, "Save File");
    }
    catch (Exception ex3)
    {
        MessageBox.Show(ex3.Message, "Error");
    }
}
private void on_btn_Click(object sender, EventArgs e)
{
    if (relayON == 50) { relayON = 0; }
    if (relayOFF == 50) { relayOFF = 0; }
    if (sys_btn.Text == "Manual")
    {
        if (on_btn.Text == "ON")
          {
              while (relayON < 50)
              {
                  on_btn.Text = "OFF";
                  myport.Write("H");
                  on_btn.BackColor = System.Drawing.Color.FromName("Green");
                 relayON++;
              }
          }
        else
        {
          while(relayOFF < 50)
          {  myport.Write("L");
          on_btn.Text = "ON";
          on_btn.BackColor = System.Drawing.Color.FromName("Control");
          relayOFF++;
```

```
            }
        }
    }
}
private void button3_Click(object sender, EventArgs e)
{
    myport.Write("R\r");
}
private void button2_Click(object sender, EventArgs e)
{
    myport.Write("C\r");
}
private void button4_Click(object sender, EventArgs e)
{
    myport.Write("E\r");
}
private void sys_btn_Click(object sender, EventArgs e)
{
    on_btn.BackColor = System.Drawing.Color.FromName("Control");

    if (sys_btn.Text == "Manual")
    {
        sys_btn.Text = "Auto";
    }
    else
    {
        sys_btn.Text = "Manual";
    }
}
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    port_name_tb.Text = comboBox1.Text;
}
private void waveformGraph1_CursorChanged(object sender, EventArgs e)
{
    specifier = "G";
    label15.Text = Lower.YPosition.ToString(specifier);
    label14.Text = Upper.YPosition.ToString(specifier);
    }
}
}
```