การประยุกต์บอร์ดไมโครคอนโทรลเลอร์กับตัวตั้งเวลาแบบโปรแกรมได้

APPLICATION OF MICROCONTROLLER FOR PROGRAMMABLE TIMER

นางสาวธนิกานต์ วรเลิศ
นางสาวศศิมาภรณ์ มงคลพิทักษ์
นางสาวสุภาภรณ์ ใจเสงี่ยม

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิศวกรรมการวัดคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2553

APPLICATION OF MICROCONTROLLER FOR PROGRAMMABLE TIMER

THANIKAN WORALERT

SASIMAPORN MONGKOLPITAK

SUPAPORN JAISA-NGIAM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2010

สาขาวิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

.....

หัวข้อปริญญานิพนธ์ การประยุกต์บอร์ดไมโครคอนโทรลเลอร์กับตัวตั้งเวลาแบบโปรแกรมได้

APPLICATION OF MICROCONTROLLER FOR PROGRAMMABLE TIMER

นักศึกษาผู้จัดทำ

นางสาวธนิกานต์	วรเลิศ	รหัสนักศึกษา	50010654
นางสาวศศิมาภรณ์	มงคลพิทักษ์	รหัสนักศึกษา	50011547
นางสาวสุภาภรณ์	ใจเสงี่ยม	รหัสนักศึกษา	50011744

ปริญญา วิศวกรรมศาสตรบัณฑิต

สาขาวิชา วิศวกรรมการวัดคุม

ปีการศึกษา 2553

อาจารย์ผู้ควบคุมปริญญานิพนธ์	ลายมือชื่อ
รองศาสตราจารย์สุพรรณ กุลพาณิชย์	

หัวข้อปริญญานิพนธ์ การประยุกต์บอร์ดไมโครคอนโทรลเลอร์กับตัวตั้งเวลาแบบโปรแกรมได้
APPLICATION OF MICROCONTROLLER FOR PROGRAMMABLE TIMER

นักศึกษาผู้จัดทำ นางสาวธนิกานต์ วรเลิศ รหัสนักศึกษา 50010654
นางสาวศศิมาภรณ์ มงคลพิทักษ์ รหัสนักศึกษา 50011547
นางสาวสุภาภรณ์ ใจเสงี่ยม รหัสนักศึกษา 50011744

อาจารย์ที่ปรึกษา รองศาสตราจารย์สุพรรณ กุลพาณิชย์

ปีการศึกษา 2553

บทคัดย่อ

ปริญญานิพนธ์นี้ มีแรงจูงใจมาจากความสามารถอันหลากหลาย และมีประสิทธิภาพของไมโครคอนโทรลเลอร์ (Microcontroller) อาดูโน่ (Arduino)โดยเริ่มจากการเลือกรูปแบบการทำงานของไมโครคอนโทรลเลอร์เป็นตัวตั้งเวลาแบบโปรแกรมได้แล้วจึงเขียนโปรแกรม เพื่อให้อาดูโน่สามารถประยุกต์ใช้งานให้เป็นตัวตั้งเวลาแบบโปรแกรมได้ โดยการเชื่อมต่ออาดูโน่เข้ากับเครื่องคอมพิวเตอร์ เขียนโปรแกรมด้วยภาษาซีพลัสพลัส (C^{++}) และทำการป้อนค่าฐานเวลาเพื่อให้มีความสอดคล้องกับค่าเวลาจริง (Real time) เข้าไปในตัวโปรแกรมแล้วจึงออกแบบการแสดงผลที่หน้าจอแอลซีดี (LCD Display) ให้ออกมาในรูปแบบของกราฟบาร์แสดงสถานะของการนับค่าเวลา และออกแบบให้สามารถควบคุมการตั้งเวลาผ่านไฮเปอร์เทอมินอล (Hyper terminal) จึงพบว่าอาดูโน่สามารถประยุกต์ให้เป็นตัวตั้งเวลาได้อย่างมีประสิทธิภาพ

Thesis Title	Application of Microcontroller for Programmable Timer	
Authors	Miss Thanikan	Woralert
	Miss Sasimaporn	Mongkolpitak
	Miss Supaporn	Jaisa-ngiam
Thesis Advisor	Assoc. Prof. Supan	Kulpanich
Year	2010	

ABSTRACT

In this paper, we applied microcontroller development platform, Arduino board, for programmable timer which used C Compiler. The time base is set in order to match the real time of real time clock. The resultant is shown by the LCD display in bar graph form. We also design the system which can be controlled via the Hyper terminal. The result is shown that, the Arduino controller is given a high efficient in time setting.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ สำเร็จได้ด้วยความกรุณาจากรองศาสตราจารย์สุพรรณ กุลพาณิชย์ อาจารย์ที่ปรึกษาโครงการที่ได้ให้ความกรุณาแนะนำช่วยเหลือตลอดการทำปริญญานิพนธ์ รวมถึงตรวจตราและแก้ไขในส่วนของเนื้อหา

คุณขเษมวิศว์ วิทูรรัชฎ์ ที่ให้คำปรึกษาแนะนำในการเขียนโปรแกรม และปัญหาที่เกิดจากการเชื่อมต่อตัวไมโครคอนโทรเลอร์กับอุปกรณ์แสดงผล เพื่อให้สามารถใช้งานควบคุมได้ตามปรกติ คณะผู้จัดทำจึงขอกราบขอบพระคุณทุกท่านมา ณ ที่นี้ด้วย

สำหรับคุณงามความดีอันใดที่เกิดจากปริญญานิพนธ์ฉบับนี้ คณะผู้จัดทำขอมอบให้กับบิดามารดาซึ่งเป็นที่รักและเคารพยิ่ง ตลอดจนครูอาจารย์ที่เคารพทุกท่าน ที่ได้ประสิทธิ์ประสาทวิชาความรู้และถ่ายทอดประสบการณ์ที่ดีให้แก่คณะผู้จัดทำตลอดมา

คณะผู้จัดทำ

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาของปริญญานิพนธ์

ปัจจุบันวงการไมโครคอนโทรลเลอร์ (Microcontroller) ได้มีการแข่งขันและพัฒนาเพื่อให้มีความสามารถครบถ้วนในการตอบสนองกับทุกความต้องการของผู้ใช้งาน หากผู้ใช้เป็นผู้ที่เพิ่งเริ่มศึกษาการใช้งานไมโครคอนโทรลเลอร์อาดูโน่ (Arduino) จึงเหมาะที่จะเป็นทางเลือกที่ควรพิจารณาเป็นอันดับแรก ปริญญานิพนธ์ฉบับนี้มีแรงจูงใจจากความสามารถและประสิทธิภาพอันหลากหลายของอาดูโน่ ซึ่งเป็นไมโครคอนโทรลเลอร์มีรูปแบบที่ง่ายไม่ซับซ้อน และมีการให้ดาวน์โหลดโปรแกรมได้โดยไม่เสียค่าใช้จ่ายทั้งยังเปิดเผยวงจร (Open source) และซอร์สโค๊ต (Source Code) โดยใช้ภาษาซีพลัสพลัส (C^{++})ในการเขียนโปรแกรม สามารถประยุกต์ใช้งานได้หลากหลายฟังก์ชั่นอย่างมีประสิทธิภาพจึงเหมาะสำหรับผู้ที่ต้องการศึกษาและเริ่มใช้งานไมโครคอนโทรลเลอร์

เนื่องจากเวลาเป็นสิ่งที่มีค่าและมีความสำคัญ เกี่ยวข้องกับความเป็นไปของทุกชีวิตบนโลก ถ้าหากจะมีเครื่องมือที่ช่วยรักษาและควบคุมเวลาก็จะทำให้ชีวิตเป็นไปได้อย่างสะดวก มีแบบแผนมากยิ่งขึ้น ดั้งนั้นเราจึงทำการศึกษาเพื่อนำเอาไมโครคอนโทรลเลอร์อาดูโน่มาประยุกต์ให้เป็นตัวตั้งเวลาแบบโปรแกรมค่าได้ โดยการเขียนโปรแกรมควบคุมการทำงานด้วยภาษาซีพลัสพลัส และให้แสดงผลผ่านทางจอแอลซีดี (LCD Display) ที่มีขนาด 16×2 ตัวอักษร เพื่อแสดงสถานะของการตั้งค่าการทำงานของการตั้งเวลาและนับเวลา แล้วเชื่อมโยงเข้ากับเครื่องคอมพิวเตอร์ให้สามารถสั่งงานได้ทั้งจากเครื่องคอมพิวเตอร์และจากบอร์ดอาดูโน่ เพื่อจะได้มีความสะดวกในการทดลอง และเป็นแนวทางในการพัฒนาโปรแกรมให้สามารถทำงานได้อย่างมีประสิทธิภาพต่อไปในอนาคต

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1. ศึกษารายละเอียดการทำงานของตัวตั้งเวลาแบบโปรแกรมค่าได้ เพื่อที่จะนำมาใช้กับไมโครคอนโทรลเลอร์ชนิด เอวีอาร์ (AVR) ชื่อว่าอาดูโน่
2. ออกแบบฮาร์ดแวร์แสดงผลเพื่อที่จะนำมาแสดงผลการทดลองของตัวตั้งเวลาแบบโปรแกรมค่าได้
3. เขียนโปรแกรมสั่งให้อาดูโน่ทำงานเป็นตัวตั้งเวลาแบบโปรแกรมค่าได้
4. ประยุกต์ใช้ไฮเปอร์เทอมินอล (Hyper Terminal) กับอาดูโน่

1.3 ขอบเขตของปริญญานิพนธ์

1. สามารถเขียนโปรแกรมสั่งให้อาดูโน่ทำงานเป็นตัวตั้งเวลาแบบโปรแกรมค่าได้
2. สามารถเขียนโปรแกรมสั่งให้อาดูโน่ สามารถรับค่าและแสดงผลออกมาทางจอแอลซีดีได้
3. สามารถประยุกต์ใช้อาดูโน่โดยใช้ไฮเปอร์เทอมินอลได้
4. สามารถจะแสดงผลเป็นแบบอิสระ มีฟังก์ชั่นการทำงานแบบระบบปฏิบัติการแบบเดี่ยว (Stand alone OS)

1.4 ขั้นตอนการศึกษา

1. ศึกษารายละเอียดและการทำงานของตัวตั้งเวลาแบบโปรแกรมค่าได้
2. ศึกษาการเชื่อมต่อและการแสดงผลของตัวตั้งเวลาแบบโปรแกรมค่าได้ในแบบแอลซีดี
3. ศึกษาคำสั่งดีเลย์ (Delay) รวมไปถึงการ โปรแกรมและการแสดงผลของคำสั่งดีเลย์
4. ศึกษาการเขียนโปรแกรมในฟังก์ชั่นตั้งค่าเวลา การจัดการเวลาและการคำนวณค่าเวลา
5. ศึกษาแบบและเลือกแบบของฮาร์ดแวร์แสดงผลอาดูโน่ ซีเรียล แอลซีดี คีย์แพด ชีลด์ (Arduino Serial LCD Keypad Shield)
6. ศึกษาการเขียนโปรแกรม โดยใช้ภาษาซีพลัสพลัส สั่งให้คอนโทรลเลอร์ทำงานเป็นตัวตั้งเวลาแบบโปรแกรมค่าได้และแสดงผลทางจอแอลซีดี
7. ศึกษาการเขียนโปรแกรมเพื่อให้แสดงผลผ่านไฮเปอร์เทอมินอล

1.5 ประโยชน์ที่คาดว่าจะได้รับ

จากวัตถุประสงค์ และขอบเขตที่กล่าวมา ผลที่คาดว่าจะได้รับคือมีความรู้และมีความเข้าใจในรายละเอียดของตัวตั้งเวลาแบบโปรแกรมค่าได้ที่ใช้อาดูโน่เป็นตัวควบคุม และสามารถที่จะเขียนโปรแกรมให้อาดูโน่ทำงานเป็นตัวตั้งเวลาแบบโปรแกรมค่าได้อย่างเข้าใจและเกิดความชำนาญ รวมไปถึงการแสดงผลการทดลองผ่านทางหน้าจอแอลซีดีได้อย่างมีประสิทธิภาพ แล้วจึงสามารถนำไปประยุกต์ใช้ในชีวิตประจำวันได้

บทที่ 2
ทฤษฎีและหลักการ

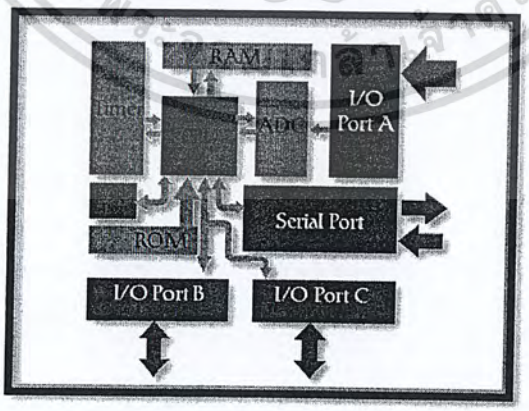
2.1 ไมโครคอนโทรลเลอร์ (Microcontroller)

2.1.1 บทนำ

ไมโครคอนโทรลเลอร์ (Microcontroller) มาจากคำ 2 คำ คำหนึ่งคือ ไมโคร (Micro) หมายถึง ขนาดเล็ก และคำว่าคอนโทรลเลอร์ (controller) หมายถึง ตัวควบคุมหรืออุปกรณ์ควบคุม ดังนั้นไมโครคอนโทรลเลอร์ จึงหมายถึงอุปกรณ์ควบคุมขนาดเล็ก แต่ในตัวอุปกรณ์ควบคุมขนาดเล็กนี้ ได้บรรจุความสามารถที่คล้ายคลึงกับระบบคอมพิวเตอร์ที่คนโดยส่วนใหญ่คุ้นเคย กล่าวคือ ภายในไมโครคอนโทรลเลอร์ได้รวมเอาซีพียู, หน่วยความจำและพอร์ต ซึ่งเป็นส่วนประกอบหลักสำคัญของระบบคอมพิวเตอร์เข้าไว้ด้วยกันโดยทำการบรรจุเข้าไว้ในตัวถังเดียวกัน

นักออกแบบและพัฒนาผลิตภัณฑ์ตลอดจนนักประดิษฐ์ทั้งหลาย ต่างหลีกเลี่ยงไม่ได้เลยที่จะต้องอาศัยวงจรอิเล็กทรอนิกส์เข้าไปมีส่วนเกี่ยวข้องในการควบคุม แต่ครั้นวงจรอิเล็กทรอนิกส์ที่นำมาต่ออนุกรมเพื่อความสามารถที่เราต้องการนั้นก็มีขนาดใหญ่ ซึ่งมีความขัดแย้งกับความต้องการของผู้บริโภคและหลักการออกแบบผลิตภัณฑ์

ดังนั้นไมโครคอนโทรลเลอร์จึงเข้ามาเกี่ยวข้อง เพื่อจะรองรับกับความต้องการและนำไปควบคุมระบบที่เราต้องการโดยให้มีขนาดเล็กที่สุดแต่มิใช่เพียงแค่ขนาดเล็กเท่านั้นยังสามารถป้อนชุดคำสั่งให้ทำงานได้อย่างอัตโนมัติ ด้วยรูปแบบการเขียนโปรแกรมภาษาต่างๆ ตามความถนัดของผู้ใช้งาน



ภาพที่ 2.1 สถาปัตยกรรมพื้นฐานของไมโครคอนโทรลเลอร์

2.1.2 โครงสร้างและส่วนประกอบของไมโครคอนโทรลเลอร์

โครงสร้างโดยทั่วไปของไมโครคอนโทรลเลอร์นั้น สามารถแบ่งออกมาได้เป็น 5 ส่วนใหญ่ๆ ดังต่อไปนี้

1. หน่วยประมวลผลกลางหรือซีพียู (CPU : Central Processing Unit)
2. หน่วยความจำ (Memory)
3. ส่วนติดต่อกับอุปกรณ์ภายนอก หรือพอร์ต (Port)
4. ช่องทางเดินของสัญญาณหรือบัส (BUS)
5. วงจรกำเนิดสัญญาณนาฬิกา

ส่วนประกอบแต่ละส่วนมีการทำงานที่แตกต่างกัน คุณสมบัติและรายละเอียดของส่วนประกอบทั้ง 5 ส่วนได้กล่าวถึงต่อไปนี้

2.1.2.1 หน่วยประมวลผลกลางหรือซีพียู (CPU : Central Processing Unit)

หน่วยประมวลผลกลางหรือซีพียูมีคุณสมบัติหลัก คือการประมวลผลข้อมูลการคำนวณทางคณิตศาสตร์และลอจิก

2.1.2.2 หน่วยความจำ (Memory)

หน่วยความจำ (Memory) สามารถแบ่งออกเป็น 2 ส่วน คือ หน่วยความจำที่มีไว้สำหรับเก็บโปรแกรมหลัก (Program Memory) เปรียบเสมือนฮาร์ดดิสก์ของเครื่องคอมพิวเตอร์ตั้งโต๊ะคือข้อมูลใดๆที่ถูกเก็บไว้ในนี้จะไม่สูญหายไปแม้ไม่มีไฟเลี้ยง อีกส่วนหนึ่งคือหน่วยความจำข้อมูล (Data Memory) ใช้เป็นเหมือนกระดาษทดในการคำนวณของซีพียูและเป็นที่พักข้อมูลชั่วคราวขณะทำงานแต่หากไม่มีไฟเลี้ยงข้อมูลก็จะหายไป คล้ายกับหน่วยความแรม (RAM) ในเครื่องคอมพิวเตอร์ทั่วๆ ไปแต่สำหรับไมโครคอนโทรลเลอร์สมัยใหม่หน่วยความจำข้อมูลจะมีทั้งที่เป็นหน่วยความจำแรมซึ่งข้อมูลจะหายไปเมื่อไม่มีไฟเลี้ยง และเป็นอีอีพรอม (EEPROM : Erasable Electrically Read-Only Memory) ซึ่งสามารถเก็บข้อมูลได้แม้ไม่มีไฟเลี้ยง

2.1.2.3 ส่วนติดต่อกับอุปกรณ์ภายนอกหรือพอร์ต (Port)

ส่วนติดต่อกับอุปกรณ์ภายนอกหรือพอร์ตมี 2 ลักษณะคือ พอร์ตอินพุต (Input Port) และพอร์ตส่งสัญญาณหรือพอร์ตเอาต์พุต (Output Port) ส่วนนี้จะใช้ในการเชื่อมต่อกับอุปกรณ์ภายนอกถือว่าเป็นส่วนที่สำคัญมากใช้ร่วมกันระหว่างพอร์ตอินพุตเพื่อรับสัญญาณ อาจจะ

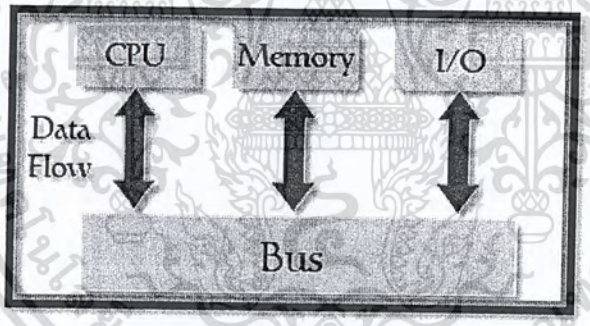
ด้วยการกดสวิตช์เพื่อนำไปประมวลผลและส่งไปพอร์ตเอาต์พุตเพื่อแสดงผลเช่น การติดสว่างของหลอดไฟ เป็นต้น

2.1.2.4 ช่องทางเดินของสัญญาณ หรือบัส (BUS)

ช่องทางเดินของสัญญาณหรือบัส (BUS) คือ เส้นทางการแลกเปลี่ยนสัญญาณข้อมูลระหว่างซีพียู, หน่วยความจำและพอร์ตเป็นลักษณะของสายสัญญาณจำนวนมากอยู่ภายในตัวไมโครคอนโทรลเลอร์โดยแบ่งเป็นบัสข้อมูล (Data Bus), บัสแอดเดรส (Address Bus) และบัสควบคุม (Control Bus)

2.1.2.5 วงจรกำเนิดสัญญาณนาฬิกา

วงจรกำเนิดสัญญาณนาฬิกานับเป็นส่วนประกอบที่สำคัญมากอีกส่วนหนึ่ง เนื่องจากการทำงานที่เกิดขึ้นในตัวไมโครคอนโทรลเลอร์ จะขึ้นอยู่กับการกำหนดจังหวะหากสัญญาณนาฬิกามีความถี่สูง จังหวะการทำงานก็จะสามารถทำได้ถี่ขึ้นส่งผลให้ไมโครคอนโทรลเลอร์ตัวนั้นมีความเร็วในการประมวลผลสูงตามไปด้วย



ภาพที่ 2.2 การส่งข้อมูลของไมโครคอนโทรลเลอร์

2.1.3 ประเภทของไมโครคอนโทรเลอร์

ไมโครคอนโทรลเลอร์มีด้วยกันหลากหลายประเภทแบ่งตามสถาปัตยกรรม (การผลิตและกระบวนการทำงานระบบการประมวลผล) ที่มีใช้ในปัจจุบันยกตัวอย่างดังนี้

1. ไมโครคอนโทรลเลอร์ตระกูล PIC (บริษัทผู้ผลิต Microchip ไมโครชิป)
2. ไมโครคอนโทรลเลอร์ตระกูล MCS51 (บริษัทผู้ผลิต Atmel, Phillips)
3. ไมโครคอนโทรลเลอร์ตระกูล AVR (บริษัทผู้ผลิต Atmel)

4. ไมโครคอนโทรลเลอร์ตระกูล ARM7, ARM9 (บริษัทผู้ผลิต Atmel, Phillips, Analog Device, Sumsung, STMicroelectronics)

5. ไมโครคอนโทรลเลอร์ตระกูล Basic Stamp (บริษัทผู้ผลิต Parallax)

6. ไมโครคอนโทรลเลอร์ตระกูล PSOC (บริษัทผู้ผลิต CYPRESS)

7. ไมโครคอนโทรลเลอร์ตระกูล MSP (บริษัทผู้ผลิต Texas Intruments)

8. ไมโครคอนโทรลเลอร์ตระกูล 68HC (บริษัทผู้ผลิต MOTOROLA)

9. ไมโครคอนโทรลเลอร์ตระกูล H8 (บริษัทผู้ผลิต Renesas)

10. ไมโครคอนโทรลเลอร์ตระกูล RABBIT (บริษัทผู้ผลิต RABBIT SEMICONDUCTOR)

11. ไมโครคอนโทรลเลอร์ตระกูล Z80 (บริษัทผู้ผลิต Zilog)

2.1.4 ภาษาที่ใช้กับไมโครคอนโทรลเลอร์

ภาษาที่ใช้กับไมโครคอนโทรลเลอร์นั้นจะแตกต่างกันตามไมโครคอนโทลเลอร์ของแต่ละตระกูลแต่ประเภทของภาษาที่ใช้สามารถแบ่งออกเป็น

1. ภาษาเครื่อง/ภาษาแอสแซมบลี (Assembly)
2. อินเตอร์พรีเตอร์ (Interpreters)
3. คอมไพล์เลอร์ (Compilers)
4. ปาสคาล (Pascal)

2.1.4.1 ภาษาเครื่อง/ภาษาแอสแซมบลี (Assembly)

ภาษาเครื่อง (Machine Language) คือ โปรแกรมที่ไมโครคอนโทรลเลอร์สามารถเข้าใจมันแต่มันไม่ง่ายสำหรับมนุษย์ที่จะอ่านได้ ภาษาแอสแซมบลี คือ รูปแบบของภาษาเครื่องที่มนุษย์สามารถจะอ่านออกได้ ภาษาแอสแซมบลีเป็นโปรแกรมที่ทำหน้าที่ในการแปลงจากคำสั่งที่มนุษย์อ่านออกได้ไปเป็นภาษาเครื่อง โปรแกรมที่เขียนโดยภาษาแอสแซมบลีจะทำงานเร็ว และมีขนาดเล็กเพราะว่ามันสามารถเข้าถึงฮาร์ดแวร์ (Hardware) ได้โดยตรงแต่ทั้งนี้ขึ้นอยู่กับวิธีการเขียนของผู้เขียนด้วย

2.1.4.2 อินเตอร์พรีเตอร์ (Interpreters)

อินเตอร์พรีเตอร์ คือ ภาษาระดับสูงซึ่งใกล้เคียงกับภาษาของมนุษย์ โดยจะฝังตัวอยู่ในหน่วยความจำ และทำหน้าที่อ่านคำสั่งจากโปรแกรมนั้นขึ้นมาทีละคำสั่งแล้วจึงปฏิบัติตามคำสั่งนั้นๆ ตัวอย่างของอินเตอร์พรีเตอร์ที่รู้จักกันดี คือ ภาษาเบสิค (BASIC Language) และข้อเสียของอินเตอร์พรีเตอร์ คือ ทำงานได้ช้าเนื่องจากต้องแปลคำสั่งทีละคำสั่ง

2.1.4.3 คอมไพล์เลอร์ (Compilers)

คอมไพล์เลอร์ คือ ภาษาระดับสูงซึ่งจะทำหน้าที่แปลโปรแกรมที่เขียนขึ้นให้เป็นภาษาเครื่อง จากนั้นจึงนำเอาโปรแกรมที่แปลเสร็จแล้วเข้าไปเก็บในหน่วยความจำ ทำให้การทำงานเร็วขึ้น ตัวอย่างเช่น ภาษาซี (C Language) เป็นต้น

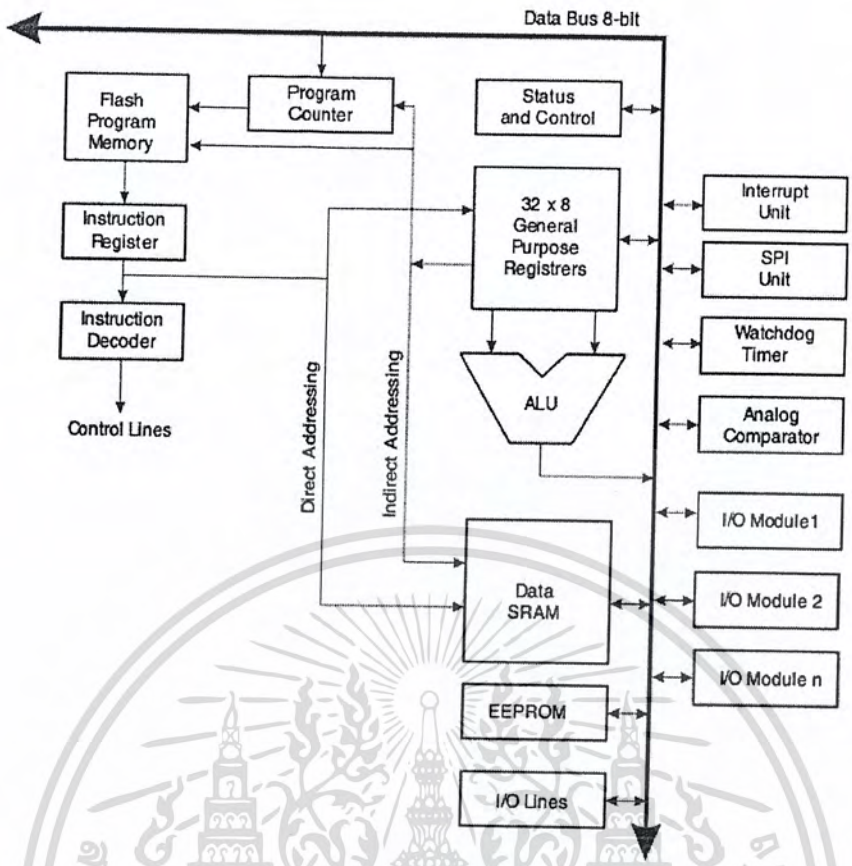
2.1.4.4 ปาสคาล (Pascal)

ปาสคาล จะมีต้นแบบมาจากภาษาแอลกอ (Algorithmic Language) และตัวภาษาปาสคาลเองก็ได้ถูกพัฒนาต่อไปเป็นภาษาที่รู้จักกันในชื่อต่างๆ เช่น ภาษาโมดุลลาส (Modulaz Language) ภาษาอะดา (Ada) ซึ่งเป็นภาษาที่ได้รับการคาดหมายว่าจะได้รับความนิยมในการอนาคต

2.2 ไมโครคอนโทรลเลอร์ตระกูลเอวีอาร์ (AVR Microcontroller)

2.2.1 บทนำ

ไมโครคอนโทรลเลอร์ตระกูลเอวีอาร์ เป็นหนึ่งในไมโครคอนโทรลเลอร์ที่ผลิตโดยบริษัท ATMEL (ซึ่งผู้นำทางด้านไมโครคอนโทรลเลอร์ตระกูล MCS-51) ตระกูลเอวีอาร์จัดเป็นไมโครคอนโทรลเลอร์ตระกูลใหม่จาก บริษัท ATMEL ซึ่งมีสถาปัตยกรรมแบบอาร์ไอเอสซี (Advanced RISC architecture) คือหนึ่งคำสั่งทำงานใช้สัญญาณนาฬิกาเพียง 1 ลูก (instructions in a single clock cycle) เป็นไมโครคอนโทรลเลอร์ที่มีประสิทธิภาพ และมีความสามารถสูง แบ่งออกเป็นหลายอนุกรม ในแต่ละอนุกรมยังแบ่งออกเป็นหลายเบอร์ เพื่อจะรองรับความต้องการที่แตกต่างของผู้ใช้งานในขณะที่ยังคงความประสิทธิภาพที่เท่ากัน



ภาพที่ 2.3 สถาปัตยกรรมพื้นฐานของไมโครคอนโทรลเลอร์ตระกูลเอวีอาร์

เอวีอาร์มีโครงสร้างเป็นแบบฮาร์ดแวร์ ซึ่งจะแยกหน่วยความโปรแกรม (Program memory) ออกจากหน่วยความจำข้อมูล (Data memory), การฟิท์ช (Fetch) และแอกคิวซ์ (Execute) คำสั่งของเอวีอาร์สามารถทำเสร็จภายในหนึ่งไซเคิล (1-cycle) ทำให้เอวีอาร์มีความเร็วหนึ่งเมกกะไอพีเอส (Mega Instruction Per Second) ต่อหนึ่งเมกกะเฮิร์ท (1-MHz)

2.2.2 ไมโครคอนโทรลเลอร์ตระกูลเอวีอาร์

ไมโครคอนโทรลเลอร์ตระกูลเอวีอาร์ได้แบ่งออกเป็น 4 กลุ่มใหญ่ๆ ดังนี้

1. TinyAVR - ATtiny series
2. MegaAVR - ATmega series
3. XMEGA - ATxmega series
4. Application specific AVR

ไมโครคอนโทรลเลอร์ตระกูลเอวีอาร์แต่ละแบบมีคุณสมบัติ และการทำงานที่แตกต่างกัน รายละเอียดคุณสมบัติ และการทำงานไมโครคอนโทรลเลอร์ทั้ง 4 แบบจะได้กล่าวถึงต่อไปนี้

2.2.2.1 TinyAVR - ATtiny series

- มีหน่วยความจำโปรแกรมขนาด 1–8 กิโลไบต์ (KB)
- มีจำนวนขาใช้งาน 6–32 ขา
- มีส่วนของอุปกรณ์เสริมที่ค่อนข้างจำกัด

2.2.2.2 MegaAVR - ATmega series

- มีหน่วยความจำโปรแกรมขนาด 4–256 กิโลไบต์
- มีจำนวนขาใช้งาน 28–100 ขา
- มีชุดคำสั่งที่สามารถจัดการกับหน่วยความจำที่มีขนาดใหญ่มากขึ้น
- มีส่วนของอุปกรณ์เสริมมากในตัวไอซี

2.2.2.3 XMEGA - ATxmega series

- มีหน่วยความจำโปรแกรมขนาด 16–384 กิโลไบต์
- มีจำนวนขาใช้งาน 44, 64, และ100 ขา
- มีชุดคำสั่งที่สามารถจัดการกับระบบดีเอ็มเอ (DMA) และการเข้าถึงเหตุการณ์ได้เร็วมากขึ้น โดยใช้การสื่อสารในแบบต่างๆ ได้หลายรูปแบบ
- มีส่วนของอุปกรณ์เสริมมากในตัวไอซีโดยการใช้งานกับดิจิตอลเป็นอนาล็อก (Digital-to-analog converter (DACs) และยังสามารถเขียนรหัสเฉพาะ โดยเข้ากันได้กับไฟล์แบบเออีเอส (AES) และดีอีเอส (DES) (เช่นใช้สำหรับการกำหนดรหัสส่วนตัว)

2.2.2.4 Application specific AVR เป็นไอซีที่สร้างเพื่อใช้งานเฉพาะ เช่น CAN AVR (ATmega64C1), LCD AVR (ATmega3290P/V), USB AVR (AT90USB1287)...

2.3 อินเตอร์รัปต์ในไมโครคอนโทรลเลอร์ตระกูลเอวีอาร์

2.3.1 บทนำ

การอินเตอร์รัปต์ (Interrupt) คือ กระบวนการขัดจังหวะการทำงานของโปรแกรมปกติหรือโปรแกรมหลักที่กำลังทำงานอยู่ เพื่อจะให้เปลี่ยนมาทำงานในส่วนของโปรแกรมที่ได้กำหนดไว้ในอินเตอร์รัปต์ กระบวนการนี้ช่วยให้ประหยัดเวลาในการทำงานของโปรแกรมหลัก และลดโอกาสผิดพลาดในการตรวจสอบเงื่อนไขการทำงานของโปรแกรม เพราะไม่ต้องคอยตรวจสอบเงื่อนไขใดเงื่อนไขหนึ่งขณะที่โปรแกรมกำลังทำงานหลักอยู่ตลอด โดยสามารถกำหนดหน้าที่การตรวจสอบนี้ให้กับอินเตอร์รัปต์แทน ประเภทของการอินเตอร์รัปต์ในไมโครคอนโทรลเลอร์ แบ่งออกเป็น 2 ประเภทคือ

1. การอินเตอร์รัปต์เนื่องจากสัญญาณภายนอก (External Interrupt) เป็นการตรวจสอบสัญญาณที่รับมาจากภายนอกตัวไมโครคอนโทรลเลอร์

2. การอินเตอร์รัปต์เนื่องจากสัญญาณภายใน (Internal Interrupt) แหล่งกำเนิดสัญญาณนี้จะเกิดจากวงจรภายในของไมโครคอนโทรลเลอร์เอง เช่น การอินเตอร์รัปต์จากสัญญาณของการเกิดโอเวอร์โฟลว์ของไทเมอร์ การอินเตอร์รัปต์เนื่องจากการอ่านเขียนหน่วยความจำอีพรอมเป็นต้น

2.3.2 การอินเตอร์รัปต์ในไมโครคอนโทรลเลอร์ตระกูลเอวีอาร์

เนื่องจากไมโครคอนโทรลเลอร์ตระกูลเอวีอาร์ ประกอบไปด้วยโมดูลการทำงานที่มีความหลากหลาย เช่น Analog to Digital conversion (ADC), อีพรอม, คอมพาเรเตอร์ (Comparator), เคปเชอร์ (Capture), ไทเมอร์ (Timer), เป็นต้น และโมดูลเหล่านี้สนับสนุนการทำงานกับสัญญาณอินเตอร์รัปต์ด้วย จะทำให้มีแหล่งกำเนิดสัญญาณอินเตอร์รัปต์ที่เกี่ยวข้องกับโมดูลดังตารางที่ 2.1 (สำหรับ ATmega 16) มีดังนี้

ตารางที่ 2.1 รีเซตและอินเตอร์รัปต์เวกเตอร์ Atmega 16

หมายเลขเวกเตอร์	แอดเดรสอินเตอร์รัปต์	ที่มาของอินเตอร์รัปต์	รายละเอียดของการอินเตอร์รัปต์
1	$000	RESET	- ขารีเซตภายนอก - เพาเวอร์ออนรีเซต, เบราเอาต์รีเซต - วอตด็อกช์ไทเมอร์รีเซต - JTAG AVR รีเซต
2	$002	INT0	อินเตอร์รัปต์เนื่องจากสัญญาณภายนอก ช่องที่ 0
3	$004	INT1	อินเตอร์รัปต์เนื่องจากสัญญาณภายนอก ช่องที่ 1
4	$006	TIMER2 COMP	อินเตอร์รัปต์เนื่องจาก ไทเมอร์/เคาน์เตอร์ 2 เปรียบเทียบตรงกัน
5	$008	TIMER2 OVF	อินเตอร์รัปต์เนื่องจาก ไทเมอร์/เคาน์เตอร์ 2 เกิดโอเวอร์โฟลว์
6	$00A	TIMER1 CAPT	อินเตอร์รัปต์เนื่องจาก ไทเมอร์/เคาน์เตอร์ 1 เกิด

			การจับสัญญาณได้
7	$00C	TIMER1 COMPA	อินเตอร์รัปต์เนื่องจาก ไทเมอร์/เคาน์เตอร์ 1 เปรียบเทียบตรงกัน แบบ A
8	$00E	TIMER1 COMPB	อินเตอร์รัปต์เนื่องจาก ไทเมอร์/เคาน์เตอร์ 1 เปรียบเทียบตรงกัน แบบ B
9	$010	TIMER1 OVF	อินเตอร์รัปต์เนื่องจาก ไทเมอร์/เคาน์เตอร์ 1 เกิดโอเวอร์โฟลว์
10	$012	TIMER0 OVF	อินเตอร์รัปต์เนื่องจาก ไทเมอร์/เคาน์เตอร์ 0 เกิดโอเวอร์โฟลว์
11	$014	SPI, STC	การส่งข้อมูลอนุกรมเสร็จสมบูรณ์
12	$016	USART, RXC	การรับข้อมูลจากโมดูล USART เสร็จสมบูรณ์
13	$018	USART, UDRE	รีจีสเตอร์ข้อมูล โมดูล USART ว่าง
14	$01A	USRT TXC	การส่งข้อมูลจากโมดูล USART เสร็จสมบูรณ์
15	$01C	ADC	โมดูล ADC แปลงข้อมูลเสร็จสมบูรณ์
16	$01E	EE_RDY	อีพรอมทำงานเสร็จแล้ว
17	$020	ANA_COMP	เกิดการเปรียบเทียบแรงดันอนาล็อก
18	$022	TWI	การรับส่งข้อมูลด้วยสายสัญญาณ 2 เส้น
19	$024	INT2	อินเตอร์รัปต์เนื่องจากสัญญาณภายนอกช่องที่ 2
20	$026	TIMER0 COMP	อินเตอร์รัปต์ เนื่องจากไมเมอร์/เคาน์เตอร์ 0 เปรียบเทียบตรงกัน
21	$028	SPM_RDY	หน่วยความจำโปรแกรมบันทึกเสร็จแล้ว

2.4 ไทม์เมอร์/เคาน์เตอร์0

2.4.1 บทนำ

โมดูลไทม์เมอร์/เคาน์เตอร์ 0 กับสัญญาณพัลส์ (Pluse Width Modulation) มีขนาด 8 บิต (8-bit Timer/Counter0 with PWM) ของเอวีอาร์ นอกจากจะมีคุณสมบัติใช้งานเป็นไทม์เมอร์หรือเคาน์เตอร์ แล้วยังมีคุณสมบัติเพิ่มเติมในการสร้างสัญญาณพัลส์ โดยจะอาศัยโมดูลการเปรียบเทียบข้อมูลในการสร้างสัญญาณ คุณสมบัติที่สำคัญของไทม์เมอร์/เคาน์เตอร์ 0 มีดังนี้

1. ใช้งานเป็นไทม์เมอร์ในการนับสัญญาณนาฬิกาภายใน (ตัวจับเวลา) หรือเป็นเคาน์เตอร์นับสัญญาณนาฬิกาภายนอก (ตัวนับสัญญาณ) ผ่านพอร์ต PB0 (XCK/T0)

2. ใช้งานเป็นฐานเวลาในโหมดเปรียบเทียบข้อมูลเพื่อสร้างสัญญาณที่ PB3 (OC0/AIN1) เพื่อกำหนดหน้าที่ใดหน้าที่หนึ่ง เช่น เมื่อเปรียบเทียบข้อมูลตรงกันจะสลับสถานะขาพอร์ต (Toggle OC0 on compare match) เมื่อเปรียบเทียบข้อมูลตรงกันแล้วจะเซตขาพอร์ต (Set OC0 on compare match)

3. ใช้งานเป็นฐานเวลาในโหมดสร้างสัญญาณพัลส์ผ่านทางขาพอร์ต OC0 ในรูปแบบ Fast PWM Mode หรือ Phase Correct PWM Mode

2.4.2 รีจีสเตอร์ที่เกี่ยวข้องกับการใช้งานไทม์เมอร์/เคาน์เตอร์ 0

รีจีสเตอร์ที่เกี่ยวข้องกับการใช้งานไทม์เมอร์/เคาน์เตอร์ 0 มีดังนี้

1. รีจีสเตอร์ TCNT 0 (Timer/Counter0 Register) รีจีสเตอร์ที่ใช้ในการนับ การนับขึ้นหรือนับลงขึ้นอยู่กับการเซตบิตในรีจีสเตอร์ TCNR0

2. รีจีสเตอร์ OCR0 (Output Compare Register) เป็นรีจีสเตอร์ OCR0 ที่ใช้กำหนดค่าเพื่อเปรียบเทียบกับค่าในรีจีสเตอร์ TCNT0 เมื่อตรงกันแล้วจะให้ผลลัพธ์ตามเงื่อนไขที่กำหนดไว้ในรีจีสเตอร์ TCCR0

3. รีจีสเตอร์ TCCR0 (Timer/Counter Control Register) รีจีสเตอร์กำหนดแหล่งสัญญาณนาฬิกาที่ใช้และกำหนดโหมดการทำงาน

4. รีจีสเตอร์ TIMSK (Timer/Counter Interrupt Mask Register) รีจีสเตอร์เอ็นเอเบิลการใช้งานอินเตอร์รัปต์ของไทม์เมอร์/เคาน์เตอร์0 โดยกำหนดเงื่อนไขการเกิดอินเตอร์รัปต์ได้ 2 รูปแบบ คือ

- บิต OCIE0 (Timer/Counter0 Output Compare Match Interrupt Enable) เอ็นเอเบิลการใช้งานอินเตอร์รัปต์เนื่องจากเปรียบเทียบข้อมูลได้ตรงกัน
- บิต TOIE0 (Timer/Counter0 Overflow Interrupt Enable) เอ็นเอเบิลการใช้งานอินเตอร์รัปต์จากการเกิดโอเวอร์โฟลว์เนื่องจากการนับ

5. รีจีสเตอร์ TIFR (Timer/Counter Interrupt Flag Register) รีจีสเตอร์แสดงการเกิดอินเตอร์รัปต์ของไทม์เมอร์/เคาน์เตอร์0

- แฟลก OCF0 (Output Compare Flag 0) ถูกเซตเมื่อเกิดเหตุการณ์ ข้อมูลเปรียบเทียบตรงกัน
- แฟรก TOV0 (Timer/Counter0 Overflow Flag) ถูกเซตเมื่อข้อมูลเกิดโอเวอร์โฟลว์

2.4.3 โหมดการทำงานของไทม์เมอร์/เคาน์เตอร์ 0

ไทม์เมอร์/เคาน์เตอร์ 0 กำหนดโหมดการทำงานได้ 4 โหมด คือ

1. โหมดไทม์เมอร์/เคาน์เตอร์หรือโหมดปกติ (Normal Mode)
2. โหมดเคลียร์ไทม์เมอร์เมื่อเปรียบเทียบข้อมูลตรงกัน (Clear Timer on Compare Match (CTC) Mode) หรือที่เรียกว่าโหมด CTC
3. โหมดสร้างสัญญาณพัลส์ แบบ Fast PWM Mode
4. โหมดสร้างสัญญาณพัลส์แบบ Phase Correct PWM Mode

2.4.4 การกำหนดโหมดการทำงาน

การกำหนดโหมดการทำงานของไทม์เมอร์/เคาน์เตอร์ 0 จะเกี่ยวข้องกับรีจีสเตอร์ TCCR0 โดยมีรายละเอียดดังนี้

2.4.4.1 รีจีสเตอร์ TCCR0 (Timer/Counter Control Register)

บิตที่	7	6	5	4	3	2	1	0
ชื่อบิต	**FOC0**	**WGM00**	**COM01**	**COM00**	**WGM01**	**CS02**	**CS01**	**CS00**
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ค่าเริ่มต้น	0	0	0	0	0	0	0	0

- บิตที่ 7 : บิต FOC0 (Force Output Compare)

บิต FOC0 จะสามารถเขียนได้อย่างเดียว และทำงานเมื่อไม่ได้อยู่ในโหมดการสร้างสัญญาณพัลส์ หรือ WGM00 เท่ากับ 0 บิตนี้ จะใช้เป็นเครื่องมือเกี่ยวกับสัญญาณสโตรบ (Strobe)

- บิตที่ 3, 6 : บิต WGM01 และ WGM00 (Waveform Generation Mode)

กำหนดโหมดสร้างสัญญาณของไทม์เมอร์/เคาน์เตอร์ 0 กับสัญญาณพัลส์ รายละเอียดดังตารางที่ 2.2

ตารางที่ 2.2 บิตกำหนดรูปแบบสัญญาณของไทม์เมอร์/เคาน์เตอร์

โหมด	**WGM01 (CTC0)**	**WGM00 (PWM0)**	**การทำงานของไทเมอร์/เคาน์เตอร์**	**ค่าสูงสุด (TOP)**	**การปรับปรุงค่าของรีจีสเตอร์ OCR0**	**การเซตค่าแฟลก TOV0**
1	0	0	Normal	0xFF	Immediate	MAX
2	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
3	1	0	CTC	OCR0	Immediate	MAX
4	1	1	Fast PWM	0xFF	TOP	MAX

โดยที่ค่า BOTTOM, MAX และTOP มีรายละเอียดดังนี้

BOTTOM มีค่าเท่ากับ 0x00 (เท่ากับ 0 ในฐานสิบ)

MAX(Maximum) มีค่าเท่ากับ 0xFF (เท่ากับ 255 ในฐานสิบ)

TOP มีค่าสูงสุดของการนับโดยกำหนดให้มีค่าค่าสูงสูงเท่ากับ MAX หรือมีค่าเท่ากับที่กำหนดในรีจีสเตอร์ OCR0 (ค่าที่กำหนดนี้จะขึ้นอยู่กับโหมดการทำงาน)

- บิตที่ 5, 4 : บิต COM01, COM00 (Compare Match Output Mode)

บิตสำหรับกำหนดโหมดการเปรียบเทียบการสร้างสัญญาณเกี่ยวกับบิตที่ 3 และ บิตที่ 6 เมื่อทำงานในโหมดไทม์เมอร์/เคาน์เตอร์หรือโหมดปกติ บิต COM01 และ COM00 จะเซตเป็น “0” หรือเป็นการปิดการใช้งานโหมด CTC และโหมดสร้างสัญญาณ PWM

- บิตที่ 2-0 : บิต CS02, CS01 และ CS00 (Clock Select)

บิตที่กำหนดแหล่งสัญญาณนาฬิกาที่ใช้งานหากกำหนดให้เป็นการใช้งานสัญญาณนาฬิกาภายในก็จะทำงานในโหมดไทม์เมอร์ และหากกำหนดให้ใช้แหล่งสัญญาณนาฬิกาจากภายนอกจะเป็นการกำหนดให้ทำงานในโหมดเคาน์เตอร์ ดังมีรายละเอียดดังตารางที่ 2.3

ตารางที่ 2.3 บิตกำหนดแหล่งสัญญาณนาฬิกาของไทม์เมอร์ 0

CS02	CS01	CS00	รายละเอียด
0	0	0	ไม่ใช่สัญญาณนาฬิกา (ไทม์เมอร์/เคาน์เตอร์หยุดทำงาน)
0	0	1	$clk_{I/O}/1$ (ไม่ใช่ค่าปรีสเกลเลอร์)
0	1	0	$clk_{I/O}/8$ (สัญญาณนาฬิกาหารด้วยค่าปรีสเกลอเลอร์ 8)
0	1	1	$clk_{I/O}/64$ (สัญญาณนาฬิกาหารด้วยค่าปรีสเกลอเลอร์ 64)
1	0	0	$clk_{I/O}/256$ (สัญญาณนาฬิกาหารด้วยค่าปรีสเกลอเลอร์ 256)
1	0	1	$clk_{I/O}/1024$ (สัญญาณนาฬิกาหารด้วยค่าปรีสเกลอเลอร์ 1024)
1	1	0	ใช้สัญญาณนาฬิกาจากภายนอกที่ขา T0 ทำงานที่ขอบขาลงของสัญญาณ

1	1	1	ใช้สัญญาณนาฬิกาจากภายนอกที่ขา T0 ทำงานที่ขอบขาขึ้นของสัญญาณ

หมายเหตุ ปรีสเกลเลอร์ (Prescaler) คือ ตัวหารหรือตัวลดทอนสัญญาณนาฬิกา (สัญญาณนาฬิกาจากคริสตอลที่ขา XTAL1 และ XTAL2)

2.4.4.2 รีจีสเตอร์ TIFR (Timer/Counter Interrupt Flag Register)

บิตที่	7	6	5	4	3	2	1	0
ชื่อบิต	**OCF0**	**TOV2**	**ICF1**	**OCF1A**	**OCF1B**	**TOV1**	**OCF0**	**TOV0**
Read/Write	R/W	R/W	R/W	R	R	R	R	R
ค่าเริ่มต้น	0	0	0	0	0	0	0	0

- บิตที่ 1 : บิต OCF0 (Timer/Counter Compare Flag)

บิต OCF0 จะถูกเซตเมื่อเกิดการเปรียบเทียบข้อมูลระหว่างรีจีสเตอร์ TCNT0 และ OCR0 ที่มีค่าเท่ากัน และการเคลียร์ค่าด้วยกระบวนการทางฮาร์ดแวร์ เมื่อมีการใช้งานฟังก์ชันอินเตอร์รัปต์ (เซตบิต OCIE0 ในรีจีสเตอร์ TIMSK และบิต I ในรีจีสเตอร์ SREG) หรือเคลียร์ด้วยกระบวนการทางซอฟต์แวร์โดยการเขียนค่า "1" ไปที่บิต OCF0

- บิตที่ 0 : บิต TOV0 (Timer/Counter Overflow Flag)

บิตแสดงการเกิดโอเวอร์โฟลว์ เนื่องจากการนับของรีจีสเตอร์ TCNT0 นั้นจะการเกิดโอเวอร์โฟลว์ คือ การเพิ่มค่าจนถึง 0xFF (255) บิต TOV0 จะถูกเซต และถูกเคลียร์ด้วยกระบวนการทางฮาร์ดแวร์เมื่อมีการอินเตอร์รัปต์ฟังก์ชันเนื่องจากการเกิดโอเวอร์โฟลว์ และจะมีการเปิดใช้งานอินเตอร์รัปต์ดังกล่าว (เซตบิต TOIE0 ในรีจีสเตอร์ TIMSK และ บิต I ในรีจีสเตอร์ SREG) หรือเคลียร์ด้วยกระบวนการทางซอฟต์แวร์โดยการเขียนค่า "1" ไปที่บิต TOV0

2.4.5 โหมดไทม์เมอร์/เคาน์เตอร์0

การทำงานในโหมดนี้จะใช้โมดูลไทม์เมอร์/เคาน์เตอร์ 0 ทำงานทั้งแบบไทม์เมอร์ และแบบเคาน์เตอร์โดยการเซตบิตต่างๆ ในรีจีสเตอร์ต่อไปนี้

- รีจีสเตอร์ TCCR0

เซตบิตที่ 2-0 : บิต CS02, CS01 และ CS00 เพื่อกำหนดสัญญาณนาฬิกา

เคลียร์บิตที่ 3-6 : บิต WGM01 และ WGM00 เพื่อกำหนดรูปแบบสัญญาณปกติ โหมด 1 (ตารางที่ 2.2)

เคลียร์บิตที่ 5-4 : บิต COM01, COM00 เพื่อปิดการใช้งานในโหมด CTC และโหมดสร้างสัญญาณพัลส์

- รีจีสเตอร์ TIFR

เซตบิตที่ 0 (TOV0) หรือ 1 (OCF0) เพื่อเอ็นเอเบิลการใช้งานอินเตอร์รัปต์ และต้องเซตบิตอินเตอร์รัปต์โดยรวมบิต I ในรีจีสเตอร์ SREG ด้วย

2.4.6 การหาค่าฐานเวลาเพื่อใช้ในการกำหนดค่าการอินเตอร์รัปต์

$$f = \left[\frac{\text{ความถี่ในการทำงานของ } Microcontroller}{\text{ค่า } Prescaler \; Maximum}\right]$$

$$= \left[\frac{16\ \text{MHz}}{\frac{64}{256}}\right]$$

$$= 976.5625\ Hz$$

$$T = \frac{1}{f}$$

$$= \frac{1}{976.5625}$$

$$= 1.024x10^{-3}\ s$$

$$\approx 1000\ ms$$

$$1000\ \text{ms} = 1\ \text{sec}$$

2.5 โปรแกรมระบบปฏิบัติการ (Operating System)

2.5.1 บทนำ

ระบบปฏิบัติการเป็นโปรแกรมที่ทำหน้าที่ควบคุมการทำงานของเครื่องคอมพิวเตอร์ และอุปกรณ์ที่ต่อพ่วงกับเครื่องคอมพิวเตอร์ โดยจะทำหน้าที่เป็นตัวกลางในการติดต่อกับฮาร์ดแวร์ของเครื่องโดยตรง โปรแกรมใช้งานหรือโปรแกรมประยุกต์ใดๆ ที่ต้องการติดต่อกับเครื่องคอมพิวเตอร์ต้องอาศัยการสั่งงานของโปรแกรมระบบปฏิบัติการ เพื่อควบคุมการทำงานของเครื่องคอมพิวเตอร์ โปรแกรมระบบปฏิบัติการของเครื่องคอมพิวเตอร์แต่ละระบบจะมีความแตกต่างกัน เช่น โปรแกรมระบบปฏิบัติการสำหรับเมนเฟรมคอมพิวเตอร์ระบบหนึ่งจะแตกต่างกับโปรแกรมระบบปฏิบัติการของเมนเฟรมคอมพิวเตอร์ระบบอื่นๆ เป็นต้น โปรแกรมระบบปฏิบัติการที่นิยมใช้อย่างแพร่หลายในเครื่องไมโครคอมพิวเตอร์ ได้แก่ MS-DOS, UNIX, Microsoft WINDOWS 95, 98, NT, XP เป็นต้น

2.5.2 หน้าที่ของโปรแกรมระบบปฏิบัติการ

- เป็นตัวกลางเชื่อมโยงระหว่างผู้ใช้ กับเครื่องคอมพิวเตอร์ เพื่อที่ให้เกิดความสะดวกในการที่จะสั่งงานคอมพิวเตอร์ โดยการจัดเตรียมโปรแกรมระบบปฏิบัติการใส่เอาไว้ในฮาร์ดดิสก์ของเครื่องคอมพิวเตอร์ เพื่อที่จะใช้เป็นเครื่องมือในการติดต่อกับผู้ใช้เช่น ระบบปฏิบัติการดอส (DOS)

ติดต่อกับผู้ใช้โดยให้พิมพ์คำสั่งพร้อมรอรับคำสั่ง (Prompt Singe) ส่วนระบบปฏิบัติการวินโดวส์ติดต่อกับผู้ใช้โดยใช้ภาพกราฟิก เป็นต้น

- ควบคุมการทำงานของฮาร์ดแวร์คอมพิวเตอร์ เช่น ควบคุมการใช้ดิสก์ไดร์ฟ ฮาร์ดดิสก์ คีย์บอร์ด และจอภาพ เป็นต้น

- ทำงานร่วมกับโปรแกรมที่อยู่ในรอมซึ่งเมื่อเริ่มบูตเครื่อง OS จะทำงานต่อจากโปรแกรมประเภท Firmware (เป็นซอฟต์แวร์ที่บริษัทผู้ผลิตคอมพิวเตอร์ได้บันทึกไว้ในหน่วยความจำรอมเพื่อตรวจสอบความพร้อมของฮาร์ดแวร์ในระบบ) ที่จัดเก็บไว้ในรอมจะเริ่มทำงานต่อเมื่อเปิดเครื่องคอมพิวเตอร์ เรามักจะเรียก Firmware นี้ว่า BIOS (Basic Input Output System) โดย BIOS จะทำการตรวจสอบความพร้อมระบบฮาร์ดแวร์ของคอมพิวเตอร์จากนั้นจึงส่งหน้าที่ให้แก่ OS เพื่อให้ควบคุมการทำงานของคอมพิวเตอร์

- จัดตารางการใช้ทรัพยากร การเข้าใช้หน่วยประมวลผลกลางของคำสั่งที่ผู้ใช้สั่งงาน เช่น กำหนดวิธีการจัดคิว (Queue) ของคำสั่งเวลาที่ OS อนุญาตให้ใช้ซีพียูของแต่ละคำสั่ง ทั้งนี้เพื่อให้หน่วยประมวลผลกลางทำงานอย่างมีประสิทธิภาพมากที่สุด

- จัดการข้อมูลและสารสนเทศในหน่วยความจำได้แก่ การนำข้อมูลไปวาง (Placement)ในหน่วยความจำ การแทนที่ข้อมูลในหน่วยความจำ (Replacement) การย้ายข้อมูลในหน่วยความจำ

- จัดการระบบการจัดเก็บไฟล์ข้อมูลลงบนสื่อสำรอง (Secondary Storage Unit)

- นำโปรแกรมประเภทอื่นเข้าประมวลผลในคอมพิวเตอร์ นอกจากประมวลผลแล้วยังคอยให้บริการเมื่อโปรแกรมต่างๆ ต้องการจะใช้ทรัพยากรของระบบคอมพิวเตอร์ ได้แก่หน่วยความจำ ฮาร์ดดิสก์ไดร์ฟ เครื่องพิมพ์ เป็นต้น

- จัดการด้านการรักษาความปลอดภัย

- จัดการเชื่อมต่อ และควบคุมอุปกรณ์ที่อยู่รอบข้างคอมพิวเตอร์ ได้แก่ เครื่องสแกนเนอร์ การ์ดเสียง และโมเด็ม เป็นต้น

2.5.3 คุณลักษณะของโปรแกรมระบบปฏิบัติการ

พิจารณาคุณลักษณะของโปรแกรมระบบปฏิบัติการตามลักษณะต่าง ๆ ดังนี้

2.5.3.1 จำนวนงานที่ทำได้ ถ้าหากมีหลายโปรแกรมทำงานพร้อมกันได้ เรียกว่า Multi - Tasking OS แต่ถ้าหากระบบปฏิบัติการควบคุมให้โปรแกรมทำงานได้ครั้งละ 1 โปรแกรมเท่านั้น เราจะเรียกว่า Single - Tasking OS

2.5.3.2 จำนวนผู้ใช้ จำนวนผู้ใช้ระบบปฏิบัติการสามารถที่จะควบคุมการทำงานให้คอมพิวเตอร์สามารถทำงานพร้อมๆ กันได้ครั้งละหลายเครื่องในระบบเครือข่ายที่มีผู้ใช้หลายคน ถ้าระบบปฏิบัติการสามารถจัดการระบบที่มีผู้ใช้หลายๆ คนพร้อมกันได้ในระบบจะเรียกว่า Multi-User OS แต่หากระบบปฏิบัติการสามารถจัดการระบบได้เพียงเครื่องเดียว หรือมีผู้ใช้ระบบได้เพียงครั้งละ 1 คน จะเรียกว่า Single - User OS

2.5.3.3 ประเภทคอมพิวเตอร์ที่ใช้ได้ ประเภทของคอมพิวเตอร์ จะแบ่งได้ออกเป็น 2 กลุ่ม คือ Generic Operation System (ระบบปฏิบัติการที่ใช้กับเครื่องคอมพิวเตอร์ได้หลายประเภท ไม่ยึดติดกับเครื่องคอมพิวเตอร์ ประเภทใด) กับอีกประเภทหนึ่ง คือ Proprietary Operating System (ระบบปฏิบัติการที่สร้างขึ้นมาเพื่อใช้กับเครื่องคอมพิวเตอร์ระบบหนึ่งระบบใด หรือยี่ห้อหนึ่งยี่ห้อใดเท่านั้น) ตัวอย่างการสร้างระบบปฏิบัติการขึ้นมาเพื่อใช้กับไมโครโปรเซสเซอร์ประเภทเดียว ไม่สามารถนำไปใช้กับคอมพิวเตอร์ประเภทอื่นๆ ได้ เช่น ระบบปฏิบัติการสำหรับเครื่องแมคอินทอช (Macintosh) และเครื่องในตระกูลแอปเปิลII (Apple II) ซึ่งจะใช้ซีพียูยี่ห้อโมโตโรล่า (Motorola) ไม่สามารถนำระบบปฏิบัติการนี้มาใช้กับเครื่องไมโครคอมพิวเตอร์ทั่วๆ ไปได้

2.5.4 ประเภทของระบบปฏิบัติการ

ระบบปฏิบัติการที่เราใช้กันโดยทั่วไปอยู่ในปัจจุบันนั้นอาจนำเอาไปใช้ได้กับคอมพิวเตอร์หลากหลายชนิดตั้งแต่เครื่องคอมพิวเตอร์ระดับใหญ่จนถึงอุปกรณ์คอมพิวเตอร์พกพาขนาดเล็กอาจแบ่งได้ออกเป็น 3 ชนิด คือ

1. ระบบปฏิบัติการแบบเดี่ยว (stand-alone OS)
2. ระบบปฏิบัติการแบบเครือข่าย (network OS)
3. ระบบปฏิบัติการแบบฝัง (embeded OS)

2.5.4.1 ระบบปฏิบัติการแบบเดี่ยว (stand-alone OS) เป็นระบบปฏิบัติการที่เน้นและให้บริการสำหรับผู้ใช้เพียงคนเดียวนิยมใช้สำหรับเครื่องคอมพิวเตอร์ที่ประมวลผล และทำงานแบบทั่วไป เช่น เครื่องคอมพิวเตอร์สำนักงาน จะถูกติดตั้งระบบปฏิบัติการนี้ไว้ใช้รองรับการทำงานบางอย่าง เช่น พิมพ์รายงาน ดูหนัง หรือเชื่อมต่อกับอินเทอร์เน็ต เป็นต้น ปัจจุบันพัฒนาให้มีคุณสมบัติที่เป็นเครื่องลูกข่ายเพื่อขอรับบริการจากเครื่องแม่ข่ายได้ด้วย

2.5.4.2 ระบบปฏิบัติการแบบเครือข่าย (network OS) เป็นระบบปฏิบัติการที่จะมุ่งเน้น และให้บริการสำหรับผู้ใช้หลายๆ คน (multi-user) เรามักนิยมใช้สำหรับงานให้บริการและประมวลผลข้อมูลสำหรับเครือข่ายโดยเฉพาะ ซึ่งมักจะพบเห็นได้กับการนำไปใช้ในองค์กรธุรกิจทั่วไป เครื่องคอมพิวเตอร์ที่ติดตั้งระบบปฏิบัติการเหล่านี้ จะเรียกว่า เครื่องเซิร์ฟวอร์ (sever) ซึ่งเป็นเสมือนเครื่องแม่ข่ายที่ให้บริการข้อมูลต่างๆ ที่จำเป็นสำหรับผู้ใช้นั่นเอง

2.5.4.3 ระบบปฏิบัติการแบบฝัง (embeded OS) เป็นระบบปฏิบัติการที่พบเห็นได้ในอุปกรณ์คอมพิวเตอร์พกพา เช่น พีดีเอ หรือสมาร์ทโฟน (Smart phone) บางรุ่น สามารถช่วยในการทำงานของอุปกรณ์แบบไม่ประจำ เกิดขึ้นมาหลังสุดพร้อมกับอุปกรณ์คอมพิวเตอร์แบบพกพา ซึ่งอุปกรณ์คอมพิวเตอร์แบบพกพาเหล่านี้ได้รับความนิยมมากขึ้น บางระบบมีคุณสมบัติที่ใกล้เคียงกับระบบปฏิบัติการแบบเดี่ยวด้วย เช่น รองรับการทำงานทั่วไป การดูหนัง ฟังเพลง หรือเชื่อมต่ออินเทอร์เน็ตได้

2.6 การทำงานแบบมัลติทาร์กกิ้ง (Multi-Tasking)

2.6.1 บทนำ

การมัลติทาร์กกิ้ง คือ การพัฒนาโปรแกรมให้ทำงานอย่างละนิดอย่างละหน่อยแบ่งกันไปจนครบทุกงานที่มี (task) แต่ซีพียูของไมโครคอนโทรลเลอร์ทำงานเร็วมากจึงมีความรู้สึกว่าทำงานพร้อมกัน หัวใจหลักสำคัญของการเขียนมัลติทาร์กแบบพื้นฐานเลยก็ คือ อย่าทำให้ติดลูป เพราะถ้าติดลูปการทำงานก็จะทำงานแค่ในลูปนั้นๆ (ทำให้ไม่สามารถทำงานอย่างอื่นได้เพราะติดอยู่ในลูป) มัลติทาร์กกิ้ง คือ ระบบหลายภารกิจ การมัลติทาร์กกิ้ง หมายถึง ความสามารถของระบบปฏิบัติการ (Operating System) ที่จะควบคุม และดำเนินการให้เครื่องคอมพิวเตอร์สามารถจัดสรรทรัพยากรของระบบให้ประมวลผลข้อมูล หรือทำงานได้หลายๆ งานพร้อมกัน ซึ่งในการทำงานของเครื่องคอมพิวเตอร์นั้นจะมีงานต่างๆ เข้ามาที่หน่วยประมวลผลกลางมากมาย เช่น การอ่านข้อมูลจากที่เก็บข้อมูลสำรอง หรือการพิมพ์ออกทางเครื่องพิมพ์ เป็นต้น แต่ว่าหน่วยประมวลผลกลางจะสามารถทำงานได้เพียงครั้งละหนึ่งคำสั่งเท่านั้น ทำให้งานอื่นๆ ที่เข้ามานั้นต้องหยุดรองานที่กำลังทำอยู่จนจบก่อน ถึงแม้ว่างานที่กำลังทำอยู่จะอยู่ในสภาวะที่ไม่ได้มีการประมวลผลอะไร ซึ่งเรียกว่า สภาวะ

ไอเดิล (Idle) เป็นการเสียเวลาการทำงานของเครื่องคอมพิวเตอร์ไปโดยเปล่าประโยชน์ ดังนั้นในเครื่องคอมพิวเตอร์ซึ่งมีความสามารถในการประมวลผลด้วยความเร็วสูง เช่น ซูเปอร์คอมพิวเตอร์ เมนเฟรม มินิคอมพิวเตอร์ รวมไปถึงไมโครคอมพิวเตอร์รุ่นใหม่ๆ จะนิยมใช้ระบบปฏิบัติการที่มีความสามารถในการทำงานหลายงานพร้อมกันได้ เรียกว่า การทำมัลติทาร์กกิ้ง

การกล่าวว่าการทำงานหลายงานหรือหลายโปรแกรมพร้อมกันนั้น ไม่ได้หมายความว่าแต่ละโปรแกรมทำงานพร้อมกันไปตลอด แต่โปรแกรมเหล่านั้นจะผลัดกันใช้หน่วยประมวลผลกลาง ในขณะที่ผู้ใช้เข้าใจว่าโปรแกรมของตนนั้นได้รับการประมวลผลจากหน่วยประมวลผลกลางอย่างต่อเนื่องตลอดเวลาเนื่องจากคอมพิวเตอร์สามารถทำงานได้เร็วมากนั่นเอง การใช้งานคอมพิวเตอร์บนระบบมัลติทาร์กกิ้งถึงแม้ว่าเทคโนโลยีในปัจจุบันจะทำให้ความสามารถในการประมวลผลของเครื่องคอมพิวเตอร์สูงมากขึ้น ระบบหรือโปรแกรมต่างๆ ที่ถูกออกแบบมาให้รองรับความต้องการในการใช้งานของผู้ใช้ล้วนมีความซับซ้อนเพิ่มมากขึ้นเช่นกัน แม้ว่าจะมีระบบมัลติทาร์กกิ้งซึ่งทำให้ผู้ใช้รู้สึกสะดวกในการทำงานหลายงานบนเครื่องคอมพิวเตอร์พร้อมกัน แต่ในความเป็นจริงแล้วนั้นมีจุดมุ่งหมายเพื่อให้เราสามารถใช้งานหน่วยประมวลผลกลางได้อย่างมีประสิทธิภาพสูงสุด ซึ่งถ้าเครื่องคอมพิวเตอร์ (Hardware) มีศักยภาพในการรองรับการประมวลผลหลายๆ งานพร้อมกันนั้นไม่เพียงพออาจจะทำให้เกิดความผิดพลาดขึ้นกับการดำเนินการ เช่น อาจเกิดการแฮงค์ (Hang) และสูญเสียข้อมูลที่ยังไม่ได้ทำการเก็บสำรองไว้ เป็นต้น

2.6.2 ประโยชน์ของการนำมาใช้

1. สามารถจัดสรรงานให้ลงตามช่วงเวลาที่ต้องการได้ง่าย เป็นการทำระบบปฏิบัติการในรูปแบบหนึ่ง ซึ่งไม่จำเป็นต้องใช้ระบบปฏิบัติการแบบเรียลไทม์ (Real Time OS) ที่มีขนาดใหญ่และราคาแพง ผู้เขียนสามารถทำความเข้าใจได้ง่ายในการบริหารเวลาของซีพียู (CPU) ได้ถูกต้อง ทำให้เราสามารถกำหนดให้งานแต่ละงานถูกกระทำเป็นเวลาที่คงที่ตามที่เราต้องการ

2. งานถูกทำไปพร้อม ๆ กันหลายๆ งาน ในลักษณะของ ไทม์แชร์ลิ่ง (Time Sharing)

3. ได้คาบเวลาที่แน่นอนเที่ยงตรง เราจึงสามารถนำมาใช้เป็นฐานเวลาภายในแต่ละงานได้ทันที

4. ทำให้เราสามารถเขียนโปรแกรมที่มีความซับซ้อนสูงๆ ได้ง่ายขึ้น

5. ลดความสูญเปล่าของเวลาที่เสียไป จากการใช้คำสั่งประเภทหน่วงเวลา เช่นดีเลย์ (Delay Time)

6. โปรแกรมจะมีโครงสร้างที่ดี

2.7 การติดต่อผ่านแบบอนุกรม

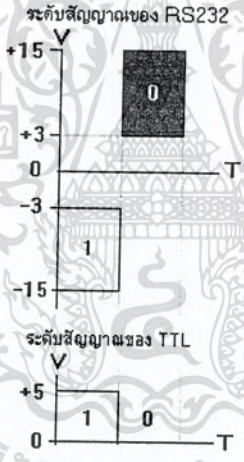
2.7.1 การใช้งานพอร์ตอนุกรมอาร์เอส 232 (RS232)

การสื่อสารแบบอนุกรมนับว่ามีความสำคัญต่อการใช้งานไมโครคอนโทรลเลอร์มาก เพราะสามารถใช้แป้นพิมพ์ และจอภาพของพีซี (PC) เป็นอินพุตและเอาต์พุตในการติดต่อ หรือควบคุมตัวไมโครคอนโทรลเลอร์ด้วยสัญญาณอย่างน้อยเพียง 3 ขาเท่านั้นคือ

- สายส่งสัญญาณ (TX)
- สายรับสัญญาณ (RX)
- สายกราวด์ (GND)

โดยปกติพอร์ตอนุกรมอาร์เอส 232C จะสามารถต่อสายได้ยาว 50 ฟุตโดยประมาณขึ้นอยู่กับชนิดของ สายสัญญาณ, ระยะทาง, และ ปริมาณสัญญาณรบกวน

2.7.2 ระดับสัญญาณของอาร์เอส 232



ภาพที่ 2.4 ระดับสัญญาณของอาร์เอส 232C และระดับสัญญาณของทีทีแอล (TTL)

สัญญาณรบกวนที่เกิดขึ้นในสายนำสัญญาณมักจะมีแรงดันเป็นบวกเมื่อเทียบกับกราวน์เพื่อป้องกันสัญญาณรบกวนนี้จึงออกแบบแรงดันของโลจิก "1" เป็นลบ คืออยู่ในช่วง - 3 ถึง - 15 โวล์ต ส่วนแรงดันของโลจิก "0" อยู่ในช่วง +3 ถึง +15 โวล์ต และเหตุที่ระดับสัญญาณของอาร์เอส 232 อยู่ในช่วง +15 ถึง -15 โวล์ต ก็เพื่อให้ต่อสายสัญญาณไปได้ไกลขึ้นดังนั้นจึงจำเป็นจะต้องมีวงจรเปลี่ยนระดับแรงดันของอาร์เอส 232 มาเป็นระดับแรงดันของทีทีแอล (TTL)

2.7.3 อัตราการส่งข้อมูล (Baud rate)

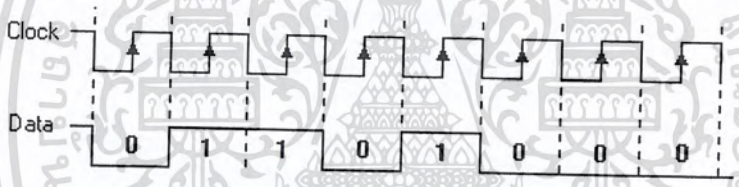
อัตราการส่งข้อมูล (Baud rate) คือ ความเร็วของการรับหรือส่งข้อมูล เป็นจำนวนบิตต่อวินาที เช่น 300, 1,200, 2,400, 4,800, 9,600, 14,400, 19,200, 38,400, 56,000 เป็นต้น การเลือกอัตราการส่งข้อมูลขึ้นอยู่กับชนิดของสายสัญญาณ, ระยะทาง, และปริมาณสัญญาณรบกวน

2.7.4 รูปแบบการสื่อสารแบบอนุกรม

รูปแบบการสื่อสารแบบอนุกรมมีด้วยกันอยู่ 2 แบบ คือแบบซิงโครนัส (Synchronous) และแบบอะซิงโครนัส (Asynchronous)

2.7.4.1 การสื่อสารแบบอนุกรม แบบซิงโครนัส (Synchronous)

การสื่อสารแบบซิงโครนัส (Synchronous) คือการรับส่งข้อมูลจะมีสัญญาณนาฬิกาซึ่งเป็นตัวกำหนดจังหวะเวลาการส่งข้อมูลร่วมอยู่ด้วยอีกขาหนึ่งใช้คู่กับสัญญาณข้อมูล ตัวอย่างเช่น การส่งสัญญาณจากคีย์บอร์ด

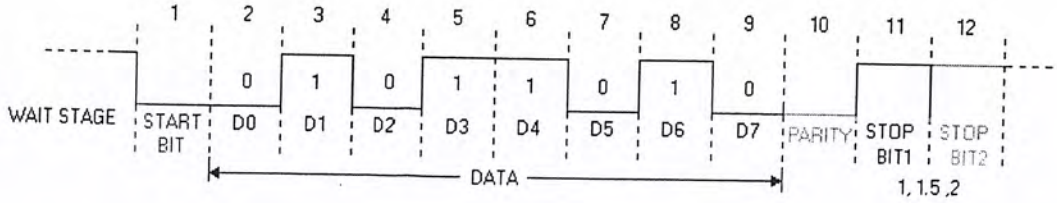


ภาพที่ 2.5 แสดงการสื่อสารแบบอนุกรมแบบซิงโครนัส (Synchronous)

2.7.4.2 การสื่อสารแบบอนุกรม แบบอะซิงโครนัส (Asynchronous)

การสื่อสารแบบอะซิงโครนัส (Asynchronous) คือ การรับส่งข้อมูลได้โดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วยแต่จะใช้ให้ตัวส่งและตัวรับมีอัตราส่งข้อมูลที่เท่ากัน รูปแบบข้อมูลแบบอะซิงโครนัส ประกอบด้วย 4 ส่วนคือ

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



ภาพที่ 2.6 การสื่อสารแบบอนุกรมแบบอะซิงโครนัส (Asynchronous)

- เมื่อไม่มีการส่งข้อมูลขาข้อมูล (data) จะมีสถานะเป็นโลจิก "1" หรือสถานะหยุดรอ (Waiting stage)

- เมื่อเริ่มต้นส่งข้อมูลจะให้ขา data เป็นโลจิก "0" เป็นจำนวน 1 บิต เรียกว่าบิตเริ่มต้น (Start bit) จากนั้นก็จะเริ่มต้นส่งข้อมูล โดยส่งบิตต่ำไปก่อน (LSB)

- แล้วตามด้วยพาริตี้บิต (จะมีหรือไม่ก็ได้ ขึ้นอยู่กับการติดตั้งค่า ของทั้งสองฝ่าย)

- สุดท้ายตามด้วยโลจิก "1" อย่างน้อย 1 บิต (มีขนาด 1, 1.5, หรือ 2 บิต) เพื่อแสดงว่าสิ้นสุดข้อมูล

การรับ และส่งข้อมูลแบบอนุกรมยังแบ่งออกเป็นลักษณะการใช้งานได้ 3 แบบคือ

1. แบบซิมเพลกซ์ (Simplex) เป็นการส่ง หรือรับข้อมูล แบบทิศทางเดียว เท่านั้น
2. แบบฮาล์ฟดูเพลกซ์ (Half Duplex) เป็นการส่ง และรับข้อมูลแบบสลับกันคือ เมื่อด้านหนึ่งส่ง อีกด้านหนึ่ง เป็นฝ่ายรับ สลับกัน ไม่สามารถรับ-ส่งในเวลาเดียวกันได้
3. แบบฟลูดูเพลกซ์ (Full Duplex) สามารถรับ-ส่งข้อมูลในเวลาเดียวกันได้

2.8 ยูเอสบี (USB)

2.8.1 บทนำ

ยูเอสบี (Universal Serial Bus) คือ ระบบเชื่อมต่ออนุกรมความเร็วสูงของคอมพิวเตอร์ซึ่งเป็นช่องทางในการสื่อสารระหว่างคอมพิวเตอร์กับอุปกรณ์อินพุต/เอาต์พุต (Input/output devices) อื่นๆ เพื่อที่นำมาเชื่อมต่อกับคอมพิวเตอร์ไม่ว่าจะเป็นปริ้นเตอร์ (Printer), โมเด็ม (Modem), เมาส์ (Mouse), คีย์บอร์ด (Keyboard), กล้องดิจิตอล (Digital Camera) และอื่นๆ

2.8.2 หลักการทำงานของระบบยูเอสบี

จะทำการตรวจสอบอุปกรณ์ต่อพ่วงทางพอร์ตยูเอสบี (Port USB) แล้วจะกำหนดแอดเดรส (Address) ให้แต่ละอุปกรณ์เรียกว่า กระบวนการอินิวเมอเรชั่น (Enumeration) หรือเมื่อเราทำการปลั๊ก (Plug) อุปกรณ์ไปยังพอร์ตยูเอสบี ระบบก็จะทำการตรวจสอบด้วยกระบวนการอินิวเมอเรชั่น

ทันที เพื่อตรวจสอบชนิดของข้อมูลที่จะทำการรับ หรือจัดส่งให้กับอุปกรณ์ที่ต่อพ่วงเข้ามาซึ่งแบ่งชนิดของข้อมูลได้เป็น 3 แบบ คือ

- ข้อมูลแบบอินเตอร์รัปต์ (Interrupt)
- ข้อมูลแบบบัลค์ (Bulk)
- ข้อมูลแบบไอโซโครนัส (Isochronous)

2.8.2.1 ข้อมูลแบบอินเตอร์รัปต์ (Interrupt)

อินเตอร์รัปต์ คือ การส่งข้อมูลทีละน้อยๆ เช่น อุปกรณ์จำพวกเมาส์ หรือคีย์บอร์ด หรือเกมส์แพด (GamePad) ต่างๆ จะทำการส่งข้อมูลให้กับเครื่องคอมพิวเตอร์คราวละเล็กน้อยและจะส่งแบบไม่ต่อเนื่องตามแต่ลักษณะการใช้งาน

2.8.2.2 ข้อมูลแบบบัลค์ (Bulk)

บัลค์ คือ การส่งข้อมูลคราวละมากๆ รวมกันเป็นก้อน เช่น การพิมพ์งาน ซึ่งเครื่องคอมพิวเตอร์จะส่งข้อมูลให้กับเครื่องพิมพ์คราวละมากๆ เป็นต้น แล้วระบบจะตรวจสอบข้อมูลทั้งความถูกต้องและความครบถ้วนด้วย

2.8.2.3 ข้อมูลแบบไอโซโครนัส (Isochronous)

ไอโซโครนัส คือ การส่งข้อมูลแบบต่อเนื่องเป็นสตรีม (Stream) เช่น พวกสปีคเกอร์ (Speaker) หรือเวปเคม (WebCam) ที่จะมาการส่งข้อมูลอย่างต่อเนื่องแบบเรียลไทม์ (Real-Time) ระหว่างเครื่องคอมพิวเตอร์หรือโฮสต์ (Host) กับอุปกรณ์ต่อพ่วงซึ่งในโหมด (Mode) นี้จะไม่มีการตรวจสอบความถูกต้องของข้อมูลว่าได้รับครบถ้วนถูกต้องหรือไม่ เครื่องคอมพิวเตอร์หรือโฮสต์จะส่งคำสั่ง หรือซักถาม (query) ไปยังอุปกรณ์ผ่านทางคอนโทรลแพคเกต (Control Packet) โดยเครื่องคอมพิวเตอร์นั้นจะทำการกันเนื้อที่ 90% ของแบนด์วิช (Bandwidth) ทั้งหมด

ความเร็วของยูเอสบี 1.1 อยู่ที่ 12 เมกกะบิตต่อวินาที (Mbps) และยูเอสบี 2.0 อยู่ที่ 480 เมกกะบิตต่อวินาที

สำหรับใช้งานการส่งข้อมูลแบบไอโซโครนัสถ้าหากมีการใช้งานถึง 90% ระบบจะปฏิเสธการร้องขอในแบบอินเทอรัพ และมีไอโซโครนัสที่เข้ามาใหม่ทันที โดย 10% ที่กันไว้จะใช้สำหรับการส่งข้อมูลแบบบัลค์ และสำหรับคอนโทรลแพคเกต (Control Packet) ของโฮสต์นั่นเอง

2.8.3 ความสามารถของยูเอสบี

- สามารถลดข้อจำกัดในการต่ออุปกรณ์พ่วงได้มากขึ้นถึง 127 ชิ้น
- ขยายอุปกรณ์มาตรฐานด้วยไดรเวอร์มาตรฐานได้
- สามารถจ่ายไฟฟ้าขนาด 5 โวล์ตให้แก่อุปกรณ์ที่ต่อพ่วงกับยูเอสบี
- ฮอตสแวปปิ้ง (Hot Swapping) สนับสนุนการต่อ, ถอดออก และรีเซตอุปกรณ์ที่ติดต่ออยู่โดยไม่ต้องรีเซตเครื่องคอมพิวเตอร์
- สามารถส่งถ่ายข้อมูลได้สูงสุดถึง 1.5 และ 12 เมกกะบิตต่อวินาที (Mbit/Sec) สัญญาณเสียง และสัญญาณภาพ
- ลดจำนวนสายเคเบิล การเชื่อมต่อนั้นก็ง่ายเนื่องจากสายสัญญาณมีแค่ 4 สายสัญญาณ คือ V+, D+, D- และ V- โดยสายสัญญาณข้อมูล (D+ และ D-) นั้นจะเป็นแบบสายคู่ (Twist pair)
- สายเคเบิลนั้นสามารถนั้นสามารถยาวได้ถึง 5 เมตร
- มีระบบหยุดชั่วคราว (Suspend) เพื่อช่วยในการประหยัดพลังงาน
- มีการกำหนดค่าตำแหน่งแอดเดรสของอุปกรณ์ต่างๆ โดยอัตโนมัติ

2.9 อาดูโน่ (Arduino)

2.9.1 บทนำ

อาดูโน่ เป็นภาษาอิตาลี มีสำเนียงการอ่านออกเสียงที่เป็นรูปแบบเฉพาะ และยังไม่มีการกำหนดเป็นคำภาษาไทยขึ้นมาอย่างเป็นทางการบ้างก็อ่านว่า “อา-ดู-วี-โน่” หรือ “อา-เดีย-โน” หรือ “เอ-อา-ดู-ไอ-โน” และอื่นๆ อีกมากมาย เพื่อไม่ให้เกิดความสับสนจึงขอใช้คำอ่านว่า “อา-ดู-โน่” ซึ่งอาดูโน่เป็นชื่อโครงการพัฒนาไมโครคอนโทรลเลอร์ตระกูลเอวีอาร์ แบบโอเพ่นซอร์ส (Open Source) ที่ได้รับการปรับปรุงมาจากโครงการไวริ่ง (Wiring) เลือกใช้แอทเมก้า 168 (ATmega168) เป็นไมโครคอนโทรลเลอร์ที่มีจำนวนของหน่วยความจำ และพอร์ตค่อนข้างมาก เป็นอุปสรรคในการสร้างบอร์ด และต่อวงจร ทำให้บอร์ดมีขนาดใหญ่เกินความจำเป็นจึงไม่ค่อยได้รับความนิยม ต่อมาพัฒนาปรับปรุงใหม่โดยให้สามารถใช้กับไมโครคอนโทรลเลอร์เอวีอาร์ขนาดเล็ก อย่างเช่น เมก้า 8 (Mega8) และเมก้า 168 (Mega168) ระบบวงจรของบอร์ดจึงมีขนาดเล็กลงและใช้อุปกรณ์น้อยชิ้นขึ้นทำให้ง่ายกับการต่อวงจร และประหยัดต้นทุนในการสร้างบอร์ด อาดูโน่จึงได้รับความนิยมจากผู้ใช้งานทั่วโลกเป็นอย่างมาก นอกจากนี้ยังมีจุดเด่นในเรื่องของความง่ายในการเรียนรู้และใช้งานเนื่องจากมีการออกแบบคำสั่งต่างๆ ขึ้นมาสนับสนุนการใช้งานด้วยรูปแบบที่ง่ายไม่ซับซ้อน แต่สามารถนำไปใช้งานได้จริง และยังสามารถสร้างคำสั่ง/ไลบรารี (Library)ใหม่ๆ ขึ้นมาใช้เองเมื่อมีความชำนาญมากขึ้น ยังรองรับการทำงานทั้งวินโดวส์ (Widows) ลีนุ๊ก (Linux) และแมคอินทอช (Macintosh OSX)

ไมโครคอนโทรลเลอร์ตระกูลเอวีอาร์เป็นไอซีไมโครคอนโทรลเลอร์ของบริษัท Atmel ซึ่งมีสถาปัตยกรรมภายในเป็นแบบอาร์ไอเอสซี (RISC (reduced instruction set computer)) โดยใช้สัญญาณนาฬิกาเพียง 1 ลูกในการปฏิบัติงานใน 1 คำสั่ง จะประกอบด้วยหน่วยความจำโปรแกรมภายในที่เป็นแบบแฟลช์ โปรแกรมข้อมูลได้แบบ In-System programmable และในบางเบอร์ยังสามารถมีการกำหนดตำแหน่งของหน่วยความจำที่สร้างเป็นบูตโหลดเดอร์ กล่าวคือ สามารถเขียนโปรแกรมเพื่อติดต่อกับพีซี หรือไอซีตัวอื่นๆ และยังสามารถโปรแกรมให้กับตัวเองได้ มีขนาดของหน่วยความจำตามเบอร์ของไอซีในแต่ละตัว

ซึ่งโครงการอาดูโน่นี้ถูกพัฒนาเพื่อให้มีจุดเด่นเหนือกว่าไมโครคอนโทรเลอร์รายอื่นๆ เป็นต้นว่า

- ราคาไม่แพง เนื่องจากมีซอร์สโค้ด (Source Cord) และวงจรแจกให้ฟรี สามารถต่อวงจรขึ้นมาใช้เองได้ทำให้ราคาบอร์ดอาดูโน่ไม่แพงเมื่อเทียบกับบอร์ดอื่น
- ทำงานได้หลากหลายแพลตฟอร์มเพราะถูกออกแบบให้อาดูโน่ทำงานได้ทั้งบนวินโดวส์, แมคอินทอชและบนลีนุกซ์ ในขณะที่บอร์ดอื่นทำงานได้เฉพาะบนวินโดวส์
- ใช้งานง่ายมีโปรแกรมพัฒนาที่ไม่ซับซ้อนมีรูปแบบคำสั่งที่ง่ายต่อการใช้งานและสามารถสร้างคำสั่งและไลบรารีใหม่ๆขึ้นมาใช้งานเองเมื่อมีความชำนาญมากขึ้น จึงเหมาะสำหรับมือใหม่และมีความสามารถครบความต้องการของนักพัฒนามืออาชีพ
- เปิดเผยซอร์สโค้ด และนำไปพัฒนาต่อยอดได้ โปรแกรมอาดูโน่ถูกตีพิมพ์แบบเปิดเผยซอร์สโค้ด และสามารถเพิ่มเติมความสามารถผ่านไลบรารีของภาษาซีพลัสพลัส (C++ library)
- เปิดเผยวงจร และนำไปพัฒนาขยายฮาร์ดแวร์ได้ อาดูโน่จะใช้โมโครคอนโทรลเลอร์ของบริษัท Atmel เบอร์ATMEGA8 และ ATMEGA168 วงจรของบอร์ดตีพิมพ์แบบเปิดเผยวงจรภายใต้ Creative Commons License

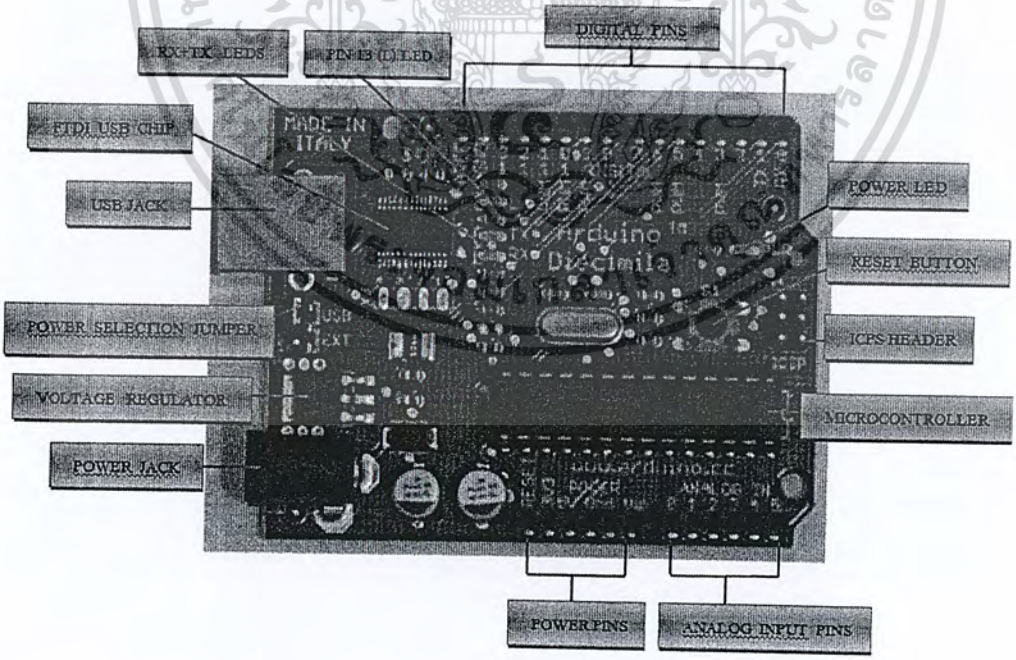
เนื่องจากได้รับความนิยมมากจากจุดเด่นที่เหนือกว่าไมโครคอนโทรเลอร์รายอื่นๆ จึงทำให้มีผู้ศึกษาการใช้งานเป็นจำนวนมากทำให้มีเอกสารข้อมูลรวมทั้งตัวอย่างต่างๆให้ใช้เป็นแนวทางในการศึกษาเรียนรู้มากมายและแพร่หลายเช่นกัน

2.9.2 ข้อมูลฮาร์ดแวร์เบื้องต้นของอาดูโน่

อาดูโน่ชนิด Duemilanove ('2009') เป็นไมโครคอนโทรเลอร์บอร์ดที่ใช้ชิฟแอทเมก้า 168 ซึ่งมี 14 พินเป็นดิจิตอลอินพุต/เอาต์พุต (โดยในนี้มี 6 พินที่สามารถสร้างเอาท์พุตแบบพัลส์ได้), มี 6 อนาล็อกอินพุต, คริสตัลออสซิเลเตอร์ (crystal oscillator) 16 เมกะเฮิร์ซ, เชื่อมต่อผ่านยูเอสบีหรือแจ็ครับไฟ, หัวต่อเป็นไอซีเอสพี (ICSP) และมีปุ่มรีเซต ทำให้บอร์ดนี้มีทุกอย่างที่สามารถรองรับไมโครคอนโทรลเลอร์เพียงต่อสายยูเอสบีพ่วงกับคอมพิวเตอร์หรือใช้เพียงพลังงานดีซี (DC (Direct Current)) จากหม้อไฟตัวบอร์ดก็สามารถทำงานได้ทันที

- Duemilanove หมายถึง 2009 ในภาษาอิตาลี ตั้งชื่อรุ่นตามปีที่บอร์ดนี้เปิดตัว
- ไมโครคอนโทรลเลอร์ (Microcontroller : ATmega168 / ATmega328)
- ระบบจ่ายไฟ (Operating Voltage) : 5 โวล์ต
- อินพุตโวล์ตเตจ (Input Voltage) (ที่ใช้) : 7 – 12 โวล์ต
- ดิจิตอลอินพุต (Digital I/O Pins) : 14 (มี 6 ที่ส่งสัญญาณเอาต์พุตแบบ PWM ได้
- อินพุตโวล์ตเตจ (Input Voltage) (จำกัด) : 6 – 20 โวล์ต
- อนาล็อกอินพุต (Analog Input Pins) : 6 ขา
- DC Current per I/) Pin : 40 มิลลิแอมป์
- DC Current for 3.3V Pin : 50 มิลลิแอมป์
- แฟลชเมมโมรี (Flash Memory) : 16 กิโลไบต์ (ATmega168) / 32กิโลไบต์ (ATmega328) ซึ่ง 2 กิโลไบต์ จะถูกใช้ไปสำหรับบู๊ทโหลดเดอร์ (BootLoader)
- แอสแรม (SRAM): 1 กิโลไบต์ (ATmega168) / 2 กิโลไบต์ (ATmega328)
- อีอีพรอม (EEPROM) : 512 ไบต์ (ATmega168) / 1 กิโลไบต์ (ATmega328)
- สัญญาณนาฬิกา (Clock Speed) : 16 เมกกะเฮิร์ต

2.9.3 ส่วนประกอบด้านหน้าของอาดูโน่



ภาพที่ 2.7 แสดงส่วนประกอบด้านหน้าของอาดูโน่

DIGITAL PINS	: ใช้สำหรับรับและส่งค่าแบบดิจิตอล
POWER SELECTION JUMPER	: ใช้กำหนดชนิดของไฟเลี้ยง
POWER JACK	: ช่องจ่ายไฟภายนอกอุปกรณ์ที่มีอุปกรณ์ต่อพ่วงมาก
USB JACK	: ช่องเชื่อมต่อกับคอมพิวเตอร์ซึ่งเชื่อมต่อโดยอาร์เอส 232
ANALOG INPUT PIN	: ใช้สำหรับรับค่าแบบอนาล็อก
RX+TX LEDS	: หลอดไฟใช้แสดงค่าการเชื่อมต่อของอุปกรณ์และคอมพิวเตอร์
RESET BUTTON	: สำหรับล้างค่าการสั่งการใช้กับอุปกรณ์
FT232	: ไอซีแปลงสัญญาณพอร์ตยูเอสบีเป็นพอร์ตอนุกรม

2.9.4 โปรแกรมอาดูโน่ (Arduino 0021)

ในการเขียนโปรแกรมลงในอาดูโน่เราใช้โปรแกรม Arduino 0021 ในการเขียนโปรแกรมแล้วทำการบันทึกลงในซีพียูของอาดูโน่

การใช้งานโปรแกรม Arduino 0021

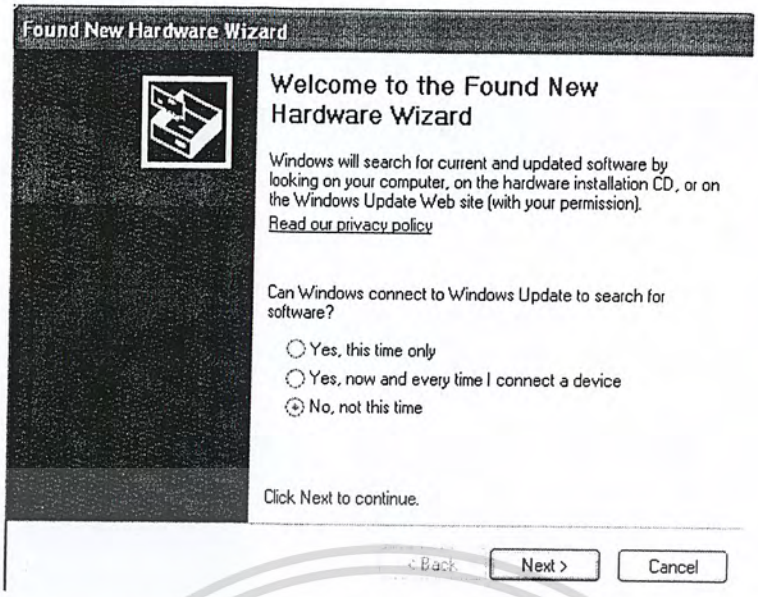
ขั้นตอนที่ 1 เตรียมไดเวอร์ของ Arduino Duemilanove สำหรับวินโดวส์เอ็กซ์พี (Windows XP)

สามารถดาวน์โหลดไดเวอร์ของบอร์ดได้ที่http://www.ftdichip.com/FTDrivers.htm เนื่องจากบอร์ดนี้ใช้ไมโครชิพของ Future Technology Devices International Ltd. (FTDI)

สำหรับท่านที่ใช้วินโดวส์เอ็กซ์พีสามารถดาว์นโหลดได้โดยตรงที่เว็บไซต์นี้ deryoung.com:download ชื่อไฟล์ : CDM 2.04.16 WHQL Certified.zip หลังจากนั้นให้เสียบสายเพื่อเชื่อมต่อระหว่างบอร์ดอาดูโน่กับเครื่องคอมพิวเตอร์ เครื่องจะถามหาไดเวอร์ให้ทำตามขั้นตอนต่างๆ ตามรูปต่อไปนี้

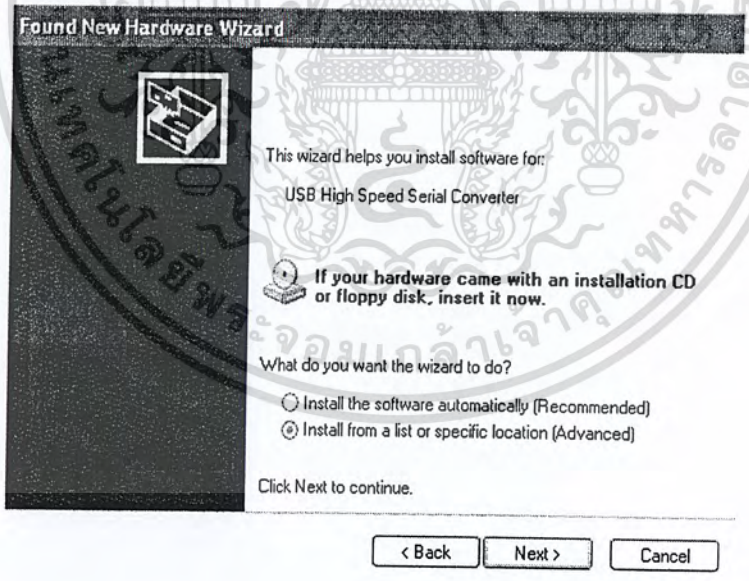
1. เชื่อมต่อสายระหว่างบอร์ดอาดูโน่กับเครื่องคอมพิวเตอร์โดยใช้สายยูเอสบีแล้วรอเครื่องถามหาไดเวอร์

2. ดูหน้าจอป๊อปอัพลิงค์นี้ที่ภาพด้านล่างจากนั้นคุณต้องเลือก " No, not this time "



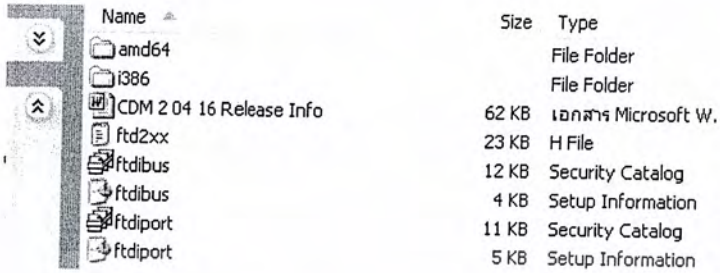
ภาพที่ 2.8 แสดงป๊อบอัพการพบฮาร์ดแวร์ไดเวอร์ใหม่ของคอมพิวเตอร์

3. เลือก "Install from a list or specific location (Advanced)" ตามภาพข้างล่าง



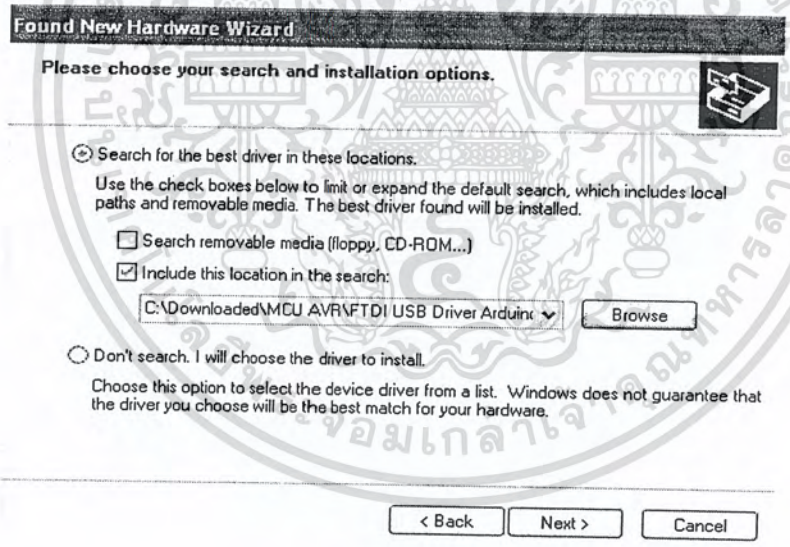
ภาพที่ 2.9 แสดงป๊อบอัพการให้เลือกติดตั้งซอฟต์แวร์ใหม่

4. เปลี่ยนตำแหน่งไดเรกทอรี่ (directory) หรือไปยังที่เก็บไฟล์ไดเวอร์ที่ระเบิดหรือแตกออกแล้ว (ในที่นี้สมมุติอยู่ที่ drive c: path: C:\Downloaded\MCU AVR\Driver UCON-232\Driver UCON-232)



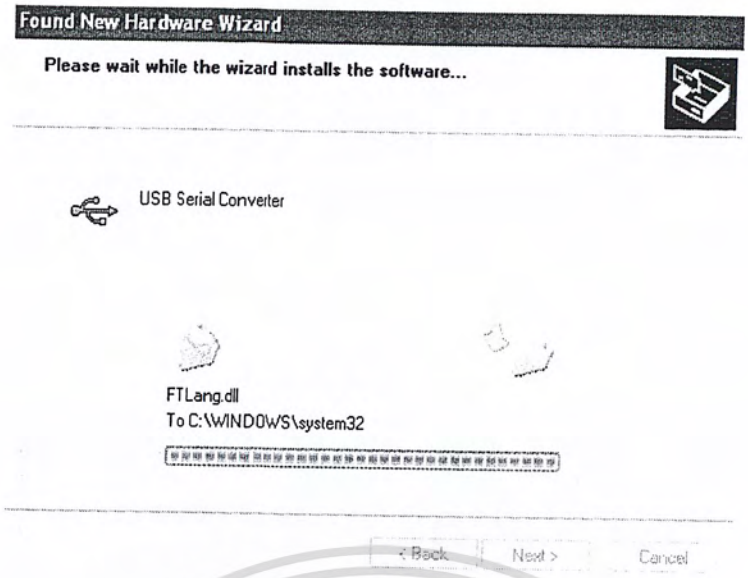
ภาพที่ 2.10 แสดงหน้า directory การเก็บเก็บไฟล์ไดเวอร์ที่ระเบิดหรือแตกออก

แล้วหาตำแหน่งที่เราระเบิดไฟล์ FTDI driver ให้เจอและเลือก (ตัวอย่างตามรูปด้านบน) เมื่อเจอแล้วให้ กด “Next”



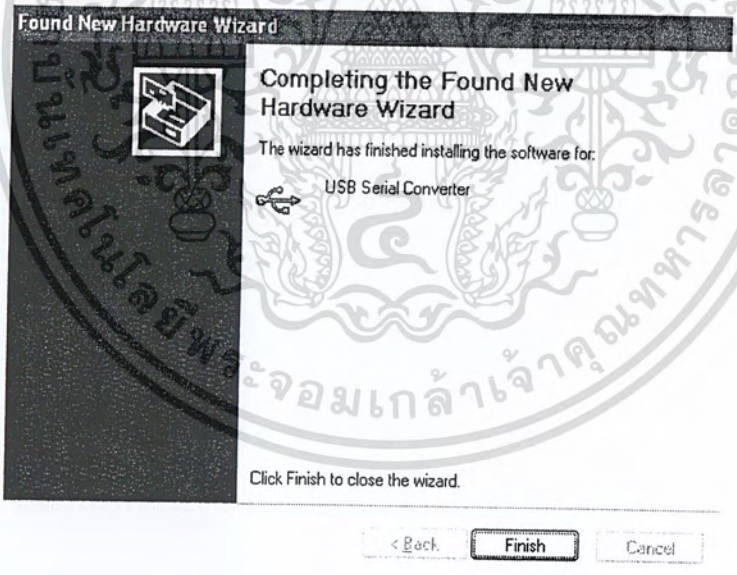
ภาพที่ 2.11 แสดงป๊อบอัพการเลือกเก็บไฟล์

5. รอการประมวลผล... windows XP do configuring....



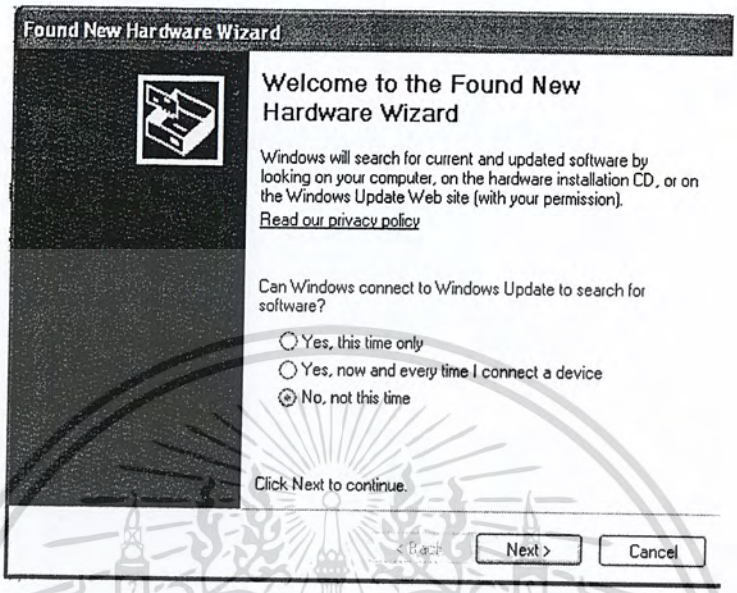
ภาพที่ 2.12 แสดงป๊อบอัพรอการติดตั้งซอฟแวร์ลงเครื่องคอมพิวเตอร์

6. เกือบจะเสร็จแล้วต้องทำอีกรอบ, คลิกที่ "Finish" and Wait...your windows XP



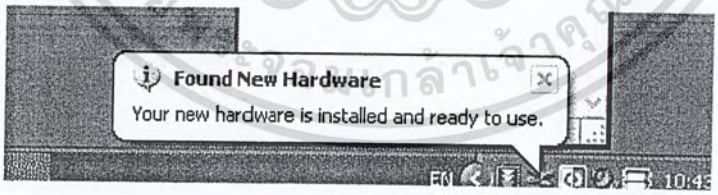
ภาพที่ 2.13 แสดงป๊อบอัพเสร็จสิ้นการติดตั้งซอฟแวร์ลงเครื่องคอมพิวเตอร์

7. ทำอีกครั้งหนึ่ง ... ขั้นที่ 2 และใช้ขั้นตอนเดิมอีกครั้งจนป๊อปอัพหน้าต่างป๊อปอัพและดูข้อความแสดงเช่นนี้ด้านล่าง...



ภาพที่ 2.14 แสดงป๊อบอัพการให้เลือกอัพเดทติดซอฟแวร์

หลังจากรอคอมพิวเตอร์ติดตั้งไดร์ฟเวอร์ถ้าสำเร็จจะต้องขึ้นตามรูปข้างล่างนี้แล้วจะขึ้นคำ "your new hardware is installed and ready to use." หากขึ้นการติดตั้งไดร์ฟเวอร์ก็เสร็จสมบูรณ์



ภาพที่ 2.15 แสดงป๊อบอัพการพร้อมใช้งานฮาร์ดแวร์

ขั้นตอนที่ 2 ตรวจสอบว่า บอร์ดอาดูโน่อยู่คอมพ์พอร์ตไหน

หลังจากที่ได้ติดตั้งไดร์ฟเวอร์ของบริษัท FTDI เรียบร้อยแล้วต้องรู้ก่อนว่าบอร์ดของเรานั้นที่เชื่อมต่อกับยูเอสบีที่เครื่องคอมพิวเตอร์ของเรานั้นได้คอมพ์พอร์ตหมายเลขอะไร

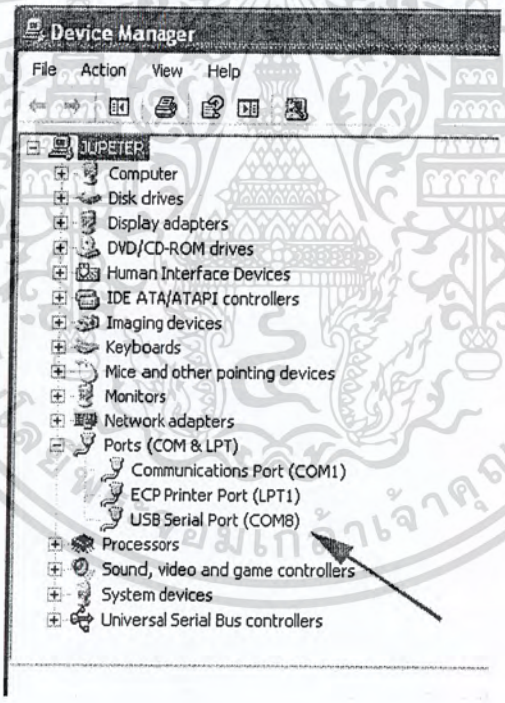
ข้อเท็จจริงคือไมโครชิพบริษัท FTDI ที่ติดมากับบอร์ดนั้นกล่าวโดยย่อ คือ แต่เดิมบอร์ดอาดูโน่จะใช้ซีเรียลพอร์ตเป็นการเชื่อมต่อกับคอมพิวเตอร์ แต่ในยุคปัจจุบันการเชื่อมต่อดังกล่าวง่ายขึ้นมากและเป็นที่นิยมกว่า โดยการหันมาใช้พอร์ตยูเอสบีซึ่งใช้งานง่าย และมีความสะดวกกว่าแถมยังมีไฟเลี้ยงวงจรขนาด 5 โวล์ตไฟกระแสตรงมาให้ด้วยจึงทำให้ในขณะทดลองเขียน และเล่นกับบอร์ดไม่จำเป็นต้องป้อนใช้ไฟเลี้ยงจากภายนอก

แต่อย่างไรก็ดียังคงต้องอ้างอิงกับหมายเลขซีเรียลพอร์ตอยู่ ดังนั้นจึงเป็นสิ่งจำเป็นที่จะมีการต้องตรวจสอบด้วยทุกครั้ง ว่าบอร์ดของเราเชื่อมต่อกับยูเอสบีพอร์ตไหน และได้หมายเลขซีเรียล-พอร์ตอะไรออกมาเพื่อจะได้ไปตั้งค่าในโปรแกรมอาดูโน่ API.

a. ไปที่หน้าจอเดสก์ทอป (Desktop) และคลิกขวาที่ไอคอล (icon) My computer

b. เลือกที่ Device manager.

c. เลือกคอมพ์พอร์ตและLPT... ดังรูป



ภาพที่ 2.16 แสดงการค้นหาพอร์ตใน Device Manager

จากรูปข้างต้นจะเห็นว่ายูเอสบีที่เชื่อมต่อกับบอร์ดอาดูโน่นั้นจะอยู่ที่คอมพ์พอร์ตหมายเลข 8 จำค่านี้ไว้ไปตั้งค่าในโปรแกรมอาดูโน่ต่อไป

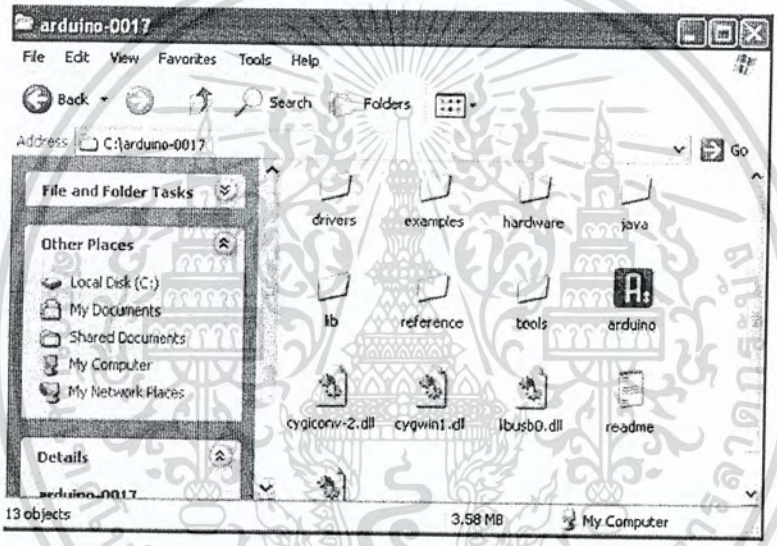
ข้อสังเกต : หากเปลี่ยนไปเชื่อมต่อกับพอร์ตยูเอสบีอื่นๆ ที่เครื่องของเราหมายเลขคอมพ์พอร์ตก็จะเปลี่ยนไปด้วย

ขั้นตอนที่ 3 ติดตั้งโปรแกรมอาดูโน่

สามารถที่จะดาวน์โหลดซอฟต์แวร์เวอร์ชั่นล่าสุดได้ที่ http://arduino.cc/en/Main/Software ซอฟต์แวร์นี้มีไว้ใช้ประโยชน์ในการเขียนโปรแกรมในการควบคุมบอร์ด ให้ทำงานได้ตามที่เราต้องการ โดยจะใช้ภาษาซีในการเขียนโปรแกรมดังกล่าว หรือหากไม่มีความรู้พื้นฐานด้านการเขียนภาษาซีมาก่อนก็ไม่ยาก

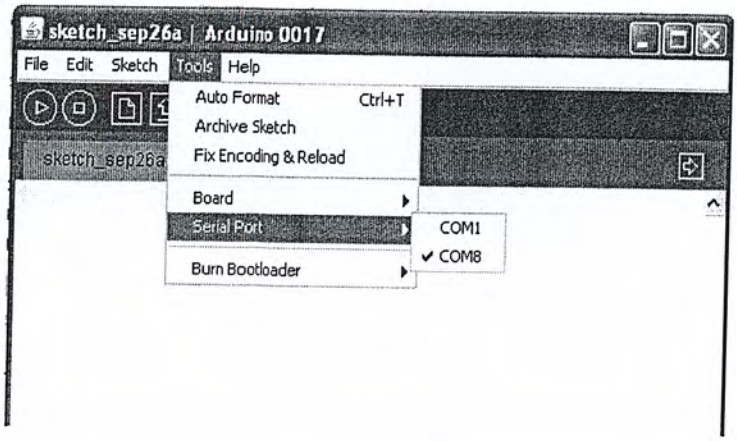
เพราะเนื่องจากอาดูโน่ใช้ได้กับภาษาซีที่มีการประยุกต์ และมีการปรับปรุงให้ออกคำสั่งที่กะทัดรัดมีเพียงไม่กี่คำสั่ง แต่ก็สามารถควบคุมการทำงานของบอร์ดได้อย่างเพียงพอเลยทีเดียว

การติดตั้งนั้นขอให้แตกไฟล์ออกไปไว้ที่ไดร์ฟซี (แนะนำ) เพราะจะจำที่อยู่ของโปรแกรมอาดูโน่ได้ง่าย ส่วนโปรแกรมที่ได้เขียนขึ้นมาจะอยู่ที่ \My Documents\Arduino



ภาพที่ 2.17 แสดงตำแหน่งของไฟล์ที่แตกในไดร์ฟซี

จากนั้นก็ให้รัน (Run) โปรแกรม arduino.exe (ตามรูปไอคอนที่เป็นตัวเอ (A)) โดยการดับเบิลคลิกขึ้นมาก็จะได้ดังรูปแล้วเลือก Tools -> Serial Port

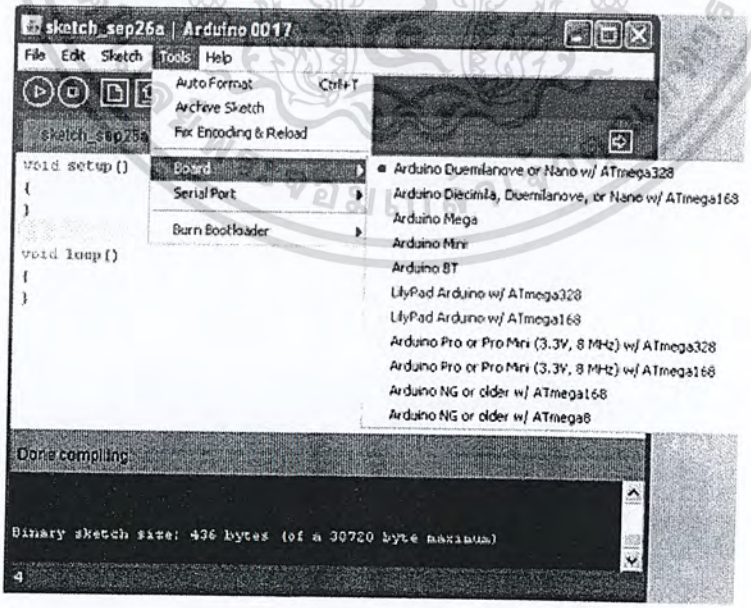


ภาพที่ 2.18 แสดงการเลือกซีเรียลพอร์ตในโปรแกรม arduino.exe

จากนั้นให้เลือกคอมพ์พอร์ตที่เราต้องการ (ขึ้นอยู่กับแต่ละเครื่อง)

ขั้นตอนที่ 4 ตรวจสอบและเลือกชนิดของบอร์ด

หลังจากที่ได้เลือกซีเรียลพอร์ตกันแล้วยังมีอีกสิ่งหนึ่งที่ต้องตั้งค่าไว้ก่อนลงมือเขียนโปรแกรมนั้นคือ การกำหนดเพื่อให้โปรแกรมทราบว่า โปรแกรมกำลังเชื่อมต่อกับบอร์ดชนิดใดอยู่ ในที่นี้หากเราเลือกใช้บอร์ด Duemilanove กับซีพียูเบอร์ ATmega328 ก็ให้เลือกชนิดนั้น



ภาพที่ 2.19 แสดงการกำหนดชนิดของบอร์ดในการเชื่อมต่อกับโปรแกรม

ขั้นตอนที่ 5 ทดลองอัพโหลด (Upload) โปรแกรม

2.9.5 การเชื่อมต่อกับเครื่องคอมพิวเตอร์ส่วนบุคคล

อาดูโน่สามารถทำการโปรแกรมได้จากโปรแกรมอาดูโน่บนเครื่องคอมพิวเตอร์โดยทำการต่อสายยูเอสบี

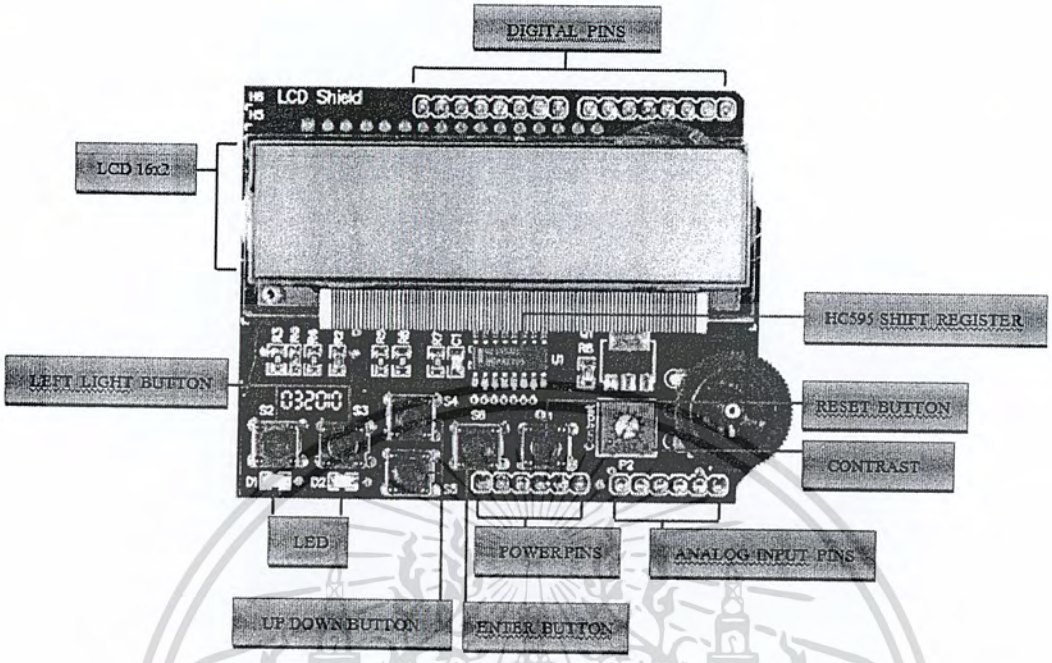
2.9.6 การติดต่อกับพอร์ตภายนอก

การติดต่อกับพอร์ตภายนอก ในการเชื่อมต่อระหว่างอาดูโน่กับคอมพิวเตอร์ส่วนบุคคลได้ง่ายดาย และรวดเร็วโดยผ่านพอร์ตอาร์เอส 232 ซึ่งอาดูโน่จะทำหน้าที่ตรวจจับอินพุตแล้วแปลงเป็นข้อมูลส่งออกไปทางอาร์เอส 232 แล้วก็ใช้กระบวนการรอรับข้อมูลจากอาร์เอส 232 ไปประมวลผลแล้วแสดงค่าอีกต่อหนึ่ง ตรงกันข้ามเมื่อต้องการสั่งควบคุมเอาท์พุตของอาดูโน่จากกระบวนการก็ต้องให้กระบวนการตรวจจับการทำงานบนโปรแกรม เช่น มีการคลิกเมาส์ที่ปุ่ม แล้วจึงแปลงเป็นข้อมูลส่งออกไปยังทางพอร์ตอาร์เอส 232 ซึ่งต้องเขียนโปรแกรมให้อาดูโน่รอรับข้อมูลจากพอร์ตอาร์เอส 232 แล้วแปลความหมายของข้อมูลเพื่อไปสั่งงานเอาท์พุตอีกต่อหนึ่ง

2.10 ฮาร์ดแวร์ในส่วนการแสดงผล (Arduino Serial LCD Keypad Shield)

ฮาร์ดแวร์ในส่วนการแสดงผลอาดูโน่ ซีเรียล แอลซีดี คีย์แพด ชีล (Arduino Serial LCD Keypad Shield) บอร์ดทำขึ้นเพื่อให้ผู้ใช้ใช้งานทางด้านการโปรแกรมค่าปุ่มกดให้ง่ายต่อการใช้งาน ประกอบด้วย จอหน้าแอลซีดีขนาด 16x2 ตัวอักษรและปุ่มกด 6 ปุ่ม

2.10.1 ส่วนประกอบด้านหน้าของอาดูโน่ ซีเรียล แอลซีดี คีย์แพด ชีล



ภาพที่ 2.20 ส่วนประกอบด้านหน้าของอาดูโน่ ซีเรียล แอลซีดี คีย์แพด ชีล

LEFT LIGHT BUTTON : สำหรับเลื่อนซ้ายขวาเพื่อเลือกโหมดการทำงานและเลื่อนหลักตัวเลขที่ต้องการจะกำหนดค่า

UP DOWN BUTTON : สำหรับป้อนตัวเลขที่ต้องการกำหนดค่า

RESET BUTTON : สำหรับล้างการกำหนดค่าเวลาและหน้าจอแอลซีดี

ENTER BUTTON : ใช้เมื่อกำหนดค่าของเวลาที่ต้องการเรียบร้อยเมื่อกดปุ่มจะแสดงเวลาที่กำหนดและบาร์กราฟ

LCD 16x2 : สำหรับแสดงผลการทำงาน

CONTRAST : ใช้สำหรับปรับความสว่างของหน้าจอแอลซีดี

บทที่ 3
หลักการและการออกแบบ

3.1 หลักการการเขียนโปรแกรมของอาดูโน่

3.1.1 โครงสร้างการเขียนโปรแกรมภาษาซีของอาดูโน่

ภาษาซีของอาดูโน่นั้น มีการจัดแบ่งรูปแบบโครงสร้างของการเขียนโปรแกรมเป็นส่วนย่อยหลายๆ ส่วน โดยจะแต่ละส่วนจะเรียกว่าฟังก์ชั่นและเมื่อนำฟังก์ชั่นมารวมกันจะเรียกว่าโปรแกรม โครงสร้างการเขียนโปรแกรมของอาดูโน่นั้นในทุกๆ โปรแกรมต้องประกอบด้วยฟังก์ชั่นจำนวนเท่าใดก็ได้ แต่อย่างน้อยที่สุดต้องมีรูปแบบการใช้งานเป็นฟังก์ชั่นจำนวน 2 ฟังก์ชั่น คือ setup() และ loop()

โครงสร้างพื้นฐานของภาษาซีที่ใช้กับอาดูโน่นั้น จะประกอบไปด้วย 3 ส่วนใหญ่ ๆ ด้วยกัน คือ

- ส่วนหัวของโปรแกรม (header) ในส่วนนี้จะมีหรือไม่มีก็ได้ ถ้ามีต้องกำหนดไว้ในส่วนเริ่มต้นของโปรแกรม ซึ่งในส่วนหัวของโปรแกรมนี้ได้แก่ ส่วนที่เป็นคำสั่งชี้แนะตัวแปลโปรแกรม (Compiler Directive) ต่างๆและรวมไปถึงส่วนของการประกาศค่าตัวแปรและค่าคงที่ต่างๆที่จะใช้ในโปรแกรม

- ส่วนของการกำหนดตัวแปร (set up) ในส่วนนี้เป็นฟังก์ชั่นบังคับที่ต้องกำหนดขึ้นใหม่ในทุกๆ โปรแกรม ถึงแม้ว่าในบางโปรแกรมจะไม่ต้องการใช้งานก็ยังจำเป็นต้องประกาศไว้ด้วยเสมอ เพียงแต่ไม่ต้องเขียนคำสั่งใดๆ ไว้ในระหว่างวงเล็บปีกกา { } ที่จะใช้เป็นตัวกำหนดขอบเขตของฟังก์ชั่นโดยฟังก์ชั่นนี้จะใช้สำหรับบรรจุคำสั่งในส่วนที่ต้องการให้โปรแกรมทำงานเพียงรอบเดียวตอนเริ่มต้นทำงานของโปรแกรมครั้งแรกเท่านั้นซึ่งจะได้แก่ คำสั่งเกี่ยวกับการตั้งค่าการทำงานต่างๆ เช่น การกำหนดหน้าที่การใช้งานของพินโหมด (PinMode) และการกำหนดอัตราการส่งของข้อมูลสำหรับใช้งานพอร์ตสื่อสารอนุกรม เป็นต้น

- ส่วนของการทำงานแบบวนรอบ (loop) จะเป็นส่วนฟังก์ชั่นบังคับที่ต้องกำหนดให้มีในทุก ๆ โปรแกรมเช่นเดียวกันกับฟังก์ชั่น setup( ) โดยฟังก์ชั่น loop( ) นี้จะใช้บรรจุคำสั่งที่ต้องการให้โปรแกรมทำงานเป็นวงรอบซ้ำ ๆ กันไปไม่รู้จบซึ่งถ้าต้องเปรียบเทียบกับรูปแบบของแอนซี-ซี (ANSI-C) ส่วนนี้ก็คือ ฟังก์ชั่นmain( ) นั่นเอง

3.1.2 คำสั่งพื้นฐานในการเขียนโปรแกรมภาษาซีของอาดูโน่

1. คำสั่ง if

คำสั่ง if เป็นคำสั่งสำหรับใช้ตรวจสอบเงื่อนไขเพื่อสั่งให้โปรแกรมเลือกทำงานตามผลลัพธ์ที่ได้จากการตรวจสอบเงื่อนไขของคำสั่งโดยมีรูปแบบคำสั่งดังนี้ คือ

```
if ( เงื่อนไข )
{
คำสั่งที่ต้องการกระทำเมื่อเงื่อนไขเป็นจริง
}
```

การทำงานของโปรแกรมเมื่อใช้การตรวจสอบเงื่อนไขแบบนี้ คือ ถ้าเงื่อนไขเป็นจริงก็จะทำงานตามคำสั่งที่อยู่หลังเงื่อนไขแต่ถ้าเป็นเงื่อนไขเป็นเท็จก็จะข้ามคำสั่งที่อยู่หลังเงื่อนไขไป

2. คำสั่ง if...else แบบ 2 ทางเลือก

คำสั่ง if...else เป็นการสั่งตรวจสอบเงื่อนไข เช่นเดียวกับคำสั่ง if แต่จะใช้สำหรับตรวจสอบเงื่อนไขที่มีหลายเงื่อนไขเพิ่มขึ้นอีก 1 ทางเลือกโดยมีรูปแบบคำสั่งดังนี้ คือ

```
if ( เงื่อนไข )
{
คำสั่งที่ต้องการให้ทำเมื่อเป็นจริง
}
else
{
คำสั่งที่ต้องการให้ทำเมื่อเป็นเท็จ
}
```

3. คำสั่ง if...else แบบหลายเงื่อนไข

คำสั่ง if...else แบบหลายเงื่อนไขเป็นการสั่งตรวจสอบเงื่อนไขเช่นเดียวกับ if...else แต่ใช้สำหรับตรวจสอบเงื่อนไขมากกว่า 1 เงื่อนไข โดยมีรูปแบบคำสั่งดังนี้ คือ

```
if ( เงื่อนไขที่ 1 )
{
คำสั่งที่ต้องการให้ทำเมื่อเงื่อนไขที่ 1 เป็นจริง
}
else if ( เงื่อนไขที่ 2 )
{
```

คำสั่งที่ต้องการให้ทำเมื่อเงื่อนไขที่ 2 เป็นจริง

}

.

.

else if (เงื่อนไขที่ n)

{

คำสั่งที่ต้องการให้ทำเมื่อเงื่อนไขที่ n เป็นจริง

}

Else

{

คำสั่งที่ต้องการให้ทำเมื่อเงื่อนไขเป็นเท็จ

}

4. คำสั่ง for

คำสั่ง for เป็นคำสั่งที่สั่งให้โปรแกรมทำงานแบบวนรอบการทำงานโดยมีการกำหนดค่าเริ่มต้น และเงื่อนไขการสิ้นสุดที่แน่นอนโดยมีรูปแบบดังนี้ คือ

for (ค่าเริ่มต้น; การเพิ่มค่าหรือลดค่าตัวแปรในแต่ละรอบ)

{

คำสั่งที่ต้องการให้ทำ ถ้าเงื่อนไขการวนรอบยังไม่เสร็จ

}

5. คำสั่ง switch/case

คำสั่งswitch/caseใช้สำหรับสั่งตรวจสอบเงื่อนไขเพื่อเลือกให้โปรแกรมทำงานตามเงื่อนไขที่ต้องการเพียงเงื่อนไขเดียว ซึ่งผลของการทำงานของคำสั่งจะเหมือนกันกับ if...else แบบหลายเงื่อนไข เพียงแต่มีรูปแบบการใช้งานที่เป็นระเบียบ และใช้การได้ง่ายกว่าซึ่งการทำงานของคำสั่งนี้จะเป็นการนำค่าในตัวแปรที่กำหนดให้ไปทำการเปรียบเทียบกับค่าคงที่ที่กำหนดไว้แล้วถ้าตรงกันก็จะให้เงื่อนไขเป็นจริง และโปรแกรมจะทำตามคำสั่งที่อยู่หลังเงื่อนไขการเปรียบเทียบค่านั้นๆ เมื่อตรวจสอบเสร็จ ก็จะไปตรวจสอบเงื่อนไขต่อไปจนถึงลำดับสุดท้ายถ้าผลลัพธ์ไม่ตรงกับเงื่อนไขใดเลยก็จะไปทำงานตามคำสั่งที่อยู่ในส่วนโดยปริยาย (default) โดยมีรูปแบบดังนี้ คือ

switch (ตัวแปรที่จะนำมาตรวจสอบ)

{

case ค่าคงที่ 1 : คำสั่งที่ต้องการให้ทำงานเมื่อการตรวจสอบค่าคงที่ 1 เป็นจริง

case ค่าคงที่ 2 : คำสั่งที่ต้องการให้ทำงานเมื่อการตรวจสอบค่าคงที่ 2 เป็นจริง

case ค่าคงที่ n : คำสั่งที่ต้องการให้ทำงานเมื่อการตรวจสอบค่าคงที่ n เป็นจริง

default : คำสั่งที่ต้องการให้ทำงานเมื่อไม่ตรงกับเงื่อนไขใดๆ

}

6. คำสั่ง while

คำสั่ง while เป็นคำสั่งที่จะสั่งให้โปรแกรมทำงานวนรอบการทำงานแบบซ้ำ ๆ กัน (loop) เช่นเดียวกันกับคำสั่ง for แต่มีความแตกต่างกันที่ตรงคำสั่ง for จะมีจำนวนรอบในการทำงานที่แน่นอนแต่คำสั่ง while จะทำงานในการวนรอบไม่รู้จบจนกว่าเงื่อนไขจะเป็นเท็จจึงจะสิ้นสุดการทำงานในการวนรอบโดยมีรูปแบบดังนี้ คือ

while (เงื่อนไข)

{

คำสั่งที่ต้องการให้ทำงานเมื่อเงื่อนไขยังเป็นจริงอยู่

}

การทำงานของคำสั่งจะเริ่มต้นด้วนการตรวจสอบเงื่อนไขในวงเล็บก่อนซึ่งถ้าพบว่าเงื่อนไขเป็นจริงอยู่ก็จะข้ามไปทำงานตามคำสั่งที่อยู่หลังเงื่อนไข เสร็จแล้วก็จะกลับมาตรวจสอบเงื่อนไขตัวใหม่โดยการทำงานจะวนซ้ำๆ อยู่เช่นนี้ไปตลอดจนกว่าเงื่อนไขจะเป็นเท็จจึงจะออกจากคำสั่ง

7. คำสั่ง do...while

คำสั่ง do...while จะใช้สำหรับคำสั่งให้โปรแกรมวนรอบการทำงานเหมือน for และ while แต่ลักษณะการทำงานจะแตกต่างกัน โดยคำสั่งนี้จะทำงานตามคำสั่งหลัง do ก่อนแล้ว จึงตรวจสอบเงื่อนไขแต่ while ซึ่งจะตรวจสอบเงื่อนไขก่อนการทำงานตามคำสั่งที่อยู่หลังเงื่อนไข โดยถ้าผลการตรวจสอบเงื่อนไขยังเป็นจริงอยู่ ก็จะกลับไปเริ่มต้นทำงานตามคำสั่งที่อยู่ด้านหลัง do และรออีกจนกว่าจะพบว่าเงื่อนไขเป็นเท็จจึงจะจบจากคำสั่งโดยรูปแบบของคำสั่งเป็นดังนี้ คือ

do

{

คำสั่งที่ต้องการให้ทำงาน

}while (เงื่อนไข);

8. คำสั่ง break

คำสั่ง break จะใช่ร่วมกับคำสั่งประเภทที่ทำงานแบบวนรอบเช่น for, do, while เพื่อให้โปรแกรมหยุดการทำงานหรือจบการทำงานจากวงรอบโดยไม่สนใจเงื่อนไข นอกจากนี้แล้วยังใช้คำสั่ง break สำหรับสั่งให้โปรแกรมจบการทำงานของคำสั่ง switch เพื่อข้ามการตรวจสอบเงื่อนไขต่อไปนี้ในคำสั่งของ switch

9. คำสั่ง continue

คำสั่ง continue จะใช้สำหรับสั่งให้โปรแกรมข้ามการทำงานของคำสั่งที่อยู่ถัดไปจำนวน 1 คำสั่ง แต่ถ้าเขียนคำสั่งนี้ไว้ภายใต้วงรอบการทำงานของคำสั่งที่ทำงานแบบเป็นวงรอบเช่น for, do, while จะเป็นการข้ามการทำงานไป 1 ระดับชั้นซึ่งเป็นการจบการทำงานจากวงรอบโดยไม่ต้องรอตรวจสอบเงื่อนไข

10. คำสั่ง goto

คำสั่ง goto ซึ่งเป็นคำสั่งสำหรับสั่งให้โปรแกรมกระโดดไปทำงานยังตำแหน่งต่างๆ ในโปรแกรมที่อ้างถึงด้วยตำแหน่งของลาเบล (label) โดยจะไม่สนใจเรื่องของเงื่อนไขซึ่งการกระโดดของโปรแกรมสามารถไปได้ทุกทิศทางทั้งเดินหน้า และถอยหลังโดยมีรูปแบบคำสั่งการใช้งานของคำสั่งดังนี้ คือ

กรณีการกระโดดถอยหลังกลับ

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

กรณีการกระโดข้ามไปข้างหน้า

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

label หมายถึงตำแหน่งลาเบล (label) ที่จะใช้เป็นจุดอ้างอิงในการให้โปรแกรมกระโดดข้ามไปทำงาน โดยการตั้งชื่อลาเบลจะต้องยึดหลัก และข้อกำหนดเช่นเดียวกันกับการตั้งชื่อตัวแปรแต่ใช้เครื่องหมายโคลอน (:) ปิดท้ายชื่อโดยไม่ถือว่าเครื่องหมายโคลอนเป็นส่วนหนึ่งของชื่อด้วยโดยจะเป็นเพียงเครื่องหมายสำหรับบอกให้คอมไพล์เลอร์ (compiler) รับรู้เพียงเท่านั้นว่าชื่อที่ประกาศไว้นั้นเป็นลาเบลไม่ใช่ตัวแปร

11. คำสั่ง return

คำสั่ง return จะใช้สำหรับจบการทำงานจากโปรแกรมย่อยนอกจากนี้แล้วยังสามารถใช้ในการคืนค่าจากโปรแกรมย่อยกลับไปยังโปรแกรมหลักด้วย

โปรแกรมย่อย (Function)

ในการเขียนโปรแกรมนั้น เมื่อโปรแกรมมีขนาดใหญ่ และมีการทำงานที่สลับซับซ้อนมากขึ้นการที่จะใช้คำสั่งจำพวกตรวจสอบเงื่อนไขต่างๆ ดังได้กล่าวมานั้นยังไม่เพียงพอในการแก้ปัญหาจึงมีแนวคิดในการแบ่งการทำงานของโปรแกรมแบ่งออกเป็นส่วนๆ แล้วจะถูกกำหนดให้ทำหน้าที่อย่างใดอย่างหนึ่ง และจะถูกเรียกว่าโปรแกรมย่อยซึ่งในภาษาซีที่เรียกว่าฟังก์ชั่น (Function) โดยมีรูปแบบดังนี้ คือ

รูปแบบฟังก์ชั่น

รูปแบบการส่งค่ากลับ ชื่อฟังก์ชั่น (ค่าที่ส่งให้ฟังก์ชั่น 1,..2,...n)

{

ชนิดตัวแปรชื่อตัวแปร

.

.

คำสั่ง

.

.

Return (ค่าที่ส่งคืนกลับ)

}

รูปแบบการส่งค่ากลับ นั้นหมายถึง ชนิดของตัวแปรที่จะใช้ในการส่งผ่านค่าจากโปรแกรมย่อยกลับไปยังโปรแกรมหลัก หรือโปรแกรมที่ทำหน้าที่เป็นผู้เรียกใช้โปรแกรมย่อยถ้าไม่ต้องการส่งค่ากลับให้กำหนดรูปแบบการส่งค่ากลับเป็น void แทนโดยรูปแบบการส่งค่ากลับต้องประกาศไว้หน้าชื่อของฟังก์ชั่นเสมอ

ชื่อฟังก์ชั่น หมายถึง ชื่อฟังก์ชั่นที่ต้องการตั้งขึ้นซึ่งใช้ข้อกำหนด และหลักเกณฑ์ในการตั้งชื่อเหมือนการตั้งชื่อตัวแปร

ค่าที่ส่งให้ฟังก์ชั่น หมายถึง ข้อมูลที่ต้องการส่งผ่านไปให้ฟังก์ชั่นทำงานซึ่งต้องกำหนดไว้ภายในวงเล็บ ( ) ที่อยู่หลัง ถ้าต้องการผ่านค่าให้กับฟังก์ชั่นมากกว่า 1 ค่า ให้ใช้เครื่องหมายคอมม่า (,) ซึ่งเป็นตัวแบ่งแยกข้อมูลแต่ละชุดด้วย แต่ถ้าไม่ต้องการให้มีการส่งผ่านค่าไปให้กับฟังก์ชั่นควรกำหนดค่าเป็น void แทน

ค่าที่ส่งคืนกลับ หมายถึง ค่าของข้อมูลที่ต้องการส่งกลับไปยังฟังก์ชั่นผู้เรียกโดยในการส่งค่าคืนกลับไปยังฟังก์ชั่นผู้เรียกนั้นจะใช้คำสั่ง return เป็นคำสั่งในการส่งค่ากลับเสมอ

ชนิดของฟังก์ชั่น

รูปแบบของฟังก์ชั่นที่ใช้ในภาษาซีจะมีอยู่ด้วยกันหลายแบบขึ้นอยู่กับการออกแบบของแต่ละโปรแกรม และความจำเป็นในการใช้งานของโปรแกรมซึ่งบางโปรแกรมก็มีการส่งผลการทำงานกลับมาให้กับโปรแกรมที่เป็นฝ่ายเรียกใช้ด้วย บางโปรแกรมก็ไม่มีการส่งค่ากลับคืนมา และบางโปรแกรมก็มีการส่งผ่านค่าข้อมูลไปให้กับโปรแกรม เพื่อใช้เป็นค่าการทำงานด้วยซึ่งสามารถแบ่งออกได้เป็น 2 แบบหลักๆ คือ

1. ฟังก์ชั่นที่ไม่มีการส่งค่ากลับมายังโปรแกรมผู้เรียก

ฟังก์ชั่นแบบนี้จะมีรูปแบบการส่งค่ากลับเป็น void ซึ่งเมื่อเรียกใช้ ก็จะเรียกเฉพาะชื่อของฟังก์ชั่นเท่านั้นซึ่งฟังก์ชั่นแบบนี้ยังมีการแบ่งแยกเป็น 2 แบบ คือ แบบที่มีการส่งผ่านค่าข้อมูลมาให้กับฟังก์ชั่น และแบบที่ไม่มีการส่งผ่านค่าข้อมูลมาให้กับฟังก์ชั่น

2. ฟังก์ชั่นที่มีการส่งค่ากลับคืนมายังโปรแกรมผู้เรียก

ฟังก์ชั่นแบบนี้จะต้องกำหนดรูปแบบการส่งค่ากลับให้กลับฟังก์ชั่นด้วย ซึ่งในการเรียกใช้งานโปรแกรมย่อยแบบนี้ โปรแกรมที่เป็นฝ่ายเรียกใช้ต้องสร้างตัวแปรที่มีข้อมูลตรงกับรูปแบบการส่งค่ากลับของฟังก์ชั่นด้วย ซึ่งฟังก์ชั่นแบบนี้ยังมีแบ่งแยกเป็น 2 แบบ คือ แบบที่มีการส่งผ่านข้อมูลมาให้กับฟังก์ชั่นและแบบที่ไม่มีการส่งผ่านข้อมูลมาให้กับฟังก์ชั่น

การส่งผ่านค่าระหว่างฟังก์ชั่น

สำหรับการส่งผ่านค่าข้อมูลจากฟังก์ชั่น ที่เป็นฝ่ายผู้เรียกไปให้กับฟังก์ชั่นที่เป็นฝ่ายถูกเรียกใช้สามารถใช้ได้กับฟังก์ชั่น แบบที่มีการส่งค่ากลับมายังฟังก์ชั่นผู้เรียก หรือฟังก์ชั่นที่ไม่มีการส่งค่ากลับมายังฟังก์ชั่นผู้เรียกก็ได้โดยวิธีการส่งผ่านค่าให้กับฟังก์ชั่นมี 3 แบบหลักๆ คือ

1. การส่งผ่านค่าด้วยตัวแปร
2. การส่งผ่านค่าด้วยพอยต์เตอร์ (Pointer)
3. การส่งผ่านค่าด้วยอะเรย์ (Array)

3.1.3 คำสั่งหรือฟังก์ชั่นของอาดูโน่

การเขียนโปรแกรมด้วยตัวควบคุมแบบอาดูโน่ นอกจากจะมีคำสั่งสำหรับใช้ในการเขียนโปรแกรมในส่วนที่เป็นโครงสร้าง และเงื่อนไขเพื่อควบคุมการทำงานของตัวโปรแกรมแล้วทางทีมพัฒนาโปรแกรมของอาดูโน่ยังได้สร้างคำสั่งพื้นฐานต่างๆ ที่จำเป็นสำหรับที่จะอำนวยความสะดวกในการเขียนโปรแกรมให้กับผู้ใช้งานไว้ด้วยแล้วจำนวนหนึ่ง ซึ่งคำสั่งต่างๆ เหล่านี้จะถูกบรรจุรวมไว้ในชุดโปรแกรมของอาดูโน่เป็นที่เรียบร้อยแล้ว หลังจากที่ได้ทำการติดตั้งชุดโปรแกรมของตัวควบคุมแบบอาดูโน่ผู้ใช้สามารถอ้างอิง และเรียกคำสั่งเหล่านี้เข้ามาใช้งานในโปรแกรมได้ทันทีซึ่งคำสั่งที่กล่าวมานี้ได้แก่

กลุ่มคำสั่งสำหรับใช้งาน Digital I/O

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

กลุ่มคำสั่งสำหรับใช้งาน Analog I/O

- int analogRead(pin)
- analogWrite(pin,value)

กลุ่มคำสั่งเพิ่มเติมสำหรับใช้งาน I/O (Advance I/O)

- Shiftout(data,clockPin,bitOrder,value)
- Unsigned long pulseIn(pin,value)

กลุ่มคำสั่งสำหรับหน่วงเวลา (Time)

- Unsigned long millis()
- delay(ms)
- delayMicrosecond(us)

กลุ่มคำสั่งทางคณิตศาสตร์ (Math)

- min(x,y)
- max(x,y)
- abs(x,y)
- contrain(x,a,b)
- map(value,fromLow,fromHigh,toLow,toHigh)
- pow(base,exponent)
- sqrt(x)

กลุ่มคำสั่งการคำนวณทางตรีโกณมิติ (Trigonometer)

- sin (rad)

- cos (rad)
- tan (rad)

กลุ่มคำสั่งการสุ่มค่าจำนวน (Random Number)

- random (seed)
- long random (max)
- long random(min,max)

กลุ่มคำสั่งการสื่อสารอนุกรม (Serial Communication)

- Serial.begin(speed)
- int Serial.available()
- int Serial.read()
- Serial.flush()
- Serial.print(data),Serial.print(data,format)
- Serial.print(data),Serial.printIn(data,format)

3.1.4 คำสั่งการเขียนโปรแกรมการติดต่อโปรแกรมซีเรียลมอนิเตอร์ (Serial Monitor) ของอาดูโน่

คำสั่งกลุ่มนี้ใช้สำหรับเป็นการติดต่อสื่อสารระหว่างอาดูโน่กับอุปกรณ์ภายนอก ในรูปแบบของการสื่อสารอนุกรมซึ่งอาจเป็นเครื่องคอมพิวเตอร์ (PC) หรืออุปกรณ์ใด ๆ ก็ได้ โดยเมื่อต้องการรับหรือส่งข้อมูลด้วยการสื่อสารอนุกรมจะต้องสูญเสียขาสัญญาณไปจำนวน 2 ขาด้วยกัน คือ PDD (Digital-0), PD1 (Digital-1) โดยที่

PDD (Digital-0) ใช้ทำหน้าที่เป็นขารับสัญญาณข้อมูลอนุกรม (RXD)

PD1 (Digital-1) ใช้ทำหน้าที่เป็นขาส่งสัญญาณข้อมูลอนุกรม (TXD)

เมื่อต้องการใช้คำสั่งในกลุ่มของการสื่อสารอนุกรมนี้แล้วขาสัญญาณ Digital-0, Digital-1 จะถูกสงวนไว้ใช้งานสำหรับการสื่อสารอนุกรมเป็นการเฉพาะ และจะไม่สามารถนำขาสัญญาณทั้ง 2 ขานี้ไปใช้งานอย่างอื่นได้อีก และภายในโปรแกรมก็จะต้องไม่มีการใช้คำสั่งอื่นๆ เช่น PinMode, DigitalWrite ที่อ้างอิงขาสัญญาณทั้ง 2 ขานี้ ในส่วนอื่นๆ ของโปรแกรมด้วยไม่เช่นนั้นจะทำให้ทำงานของโปรแกรมผิดพลาดได้โดยคำสั่งในกลุ่มของการสื่อสารอนุกรมของอาดูโน่ มีทั้งหมด 8 คำสั่งหลักๆ คือ

- Serial.begin(speed)
- int Serial.available()
- int Serial.read()

- Serial.flush()
- Serial.print(data), Serial.print(data,format)
- Serial.printIn(data), Serial.printIn(data,format)

1. Serial.begin (speed)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ใช้สำหรับเปิดพอร์ตอนุกรม พร้อมทั้งกำหนดค่าความเร็วในการสื่อสาร (Bardrate) ตามที่ผู้ใช้กำหนด ซึ่งผู้ใช้จะต้องกำหนดค่าความเร็วนี้ให้ตรงกับความเร็วของอุปกรณ์อีกฝ่ายหนึ่งที่ต้องการจะสื่อสารด้วย ซึ่งตามปรกติก็ คือ 300, 1200, 2400, 9600, 14400, 19200, 28800, 38400, 57600 และ 115200

รูปแบบของคำสั่ง

Serial.begin(int speed)

ค่าพารามิเตอร์ที่ต้องการ

- Speed คือ ค่าความเร็วในการสื่อสารข้อมูลพอร์ตสื่อสารอนุกรม ซึ่งมีค่าเป็นตัวเลขจำนวนเต็ม และค่าที่แนะนำ คือ 19200

ค่าที่คืนกลับจากฟังก์ชั่น

- ไม่มีการส่งค่ากลับ

2. int Serial.available()

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ใช้ทำหน้าที่สำหรับตรวจสอบจำนวนข้อมูลที่ได้รับได้จากพอร์ตสื่อสารอนุกรม และเก็บรอไว้ในบัฟเฟอร์ (Buffer) มีหน่วยเป็นไบต์ (Byte) ซึ่งก็ คือ จำนวนตัวอักษรที่รับเก็บไว้ในบัฟเฟอร์ขณะนั้นๆ โดยไมโครคอนโทรลเลอร์อาดูโน่ จะสร้างบัฟเฟอร์สำหรับเก็บข้อมูลที่จะรับจากพอร์ตสื่อสารอนุกรมจำนวน 128 ไบต์ โดยเมื่อเก็บข้อมูลเพิ่มเข้าไป 1 ไบต์ ก็จะทำให้พื้นที่เก็บข้อมูลของบัฟเฟอร์เหลือน้อยลง แต่เมื่อผู้ใช้สั่งอ่านค่าออกไปจากบัฟเฟอร์สำหรับเก็บข้อมูลก็จะว่าง (สามารถรับข้อมูลใหม่เข้ามาได้อีก 128 ไบต์) แต่ถ้าข้อมูลถูกรับ และเก็บไว้จนเต็มบัฟเฟอร์แล้ว โปรแกรมไม่สามารถอ่านออกไปประมวลผลได้ทันก็จะทำให้ข้อมูลที่ส่งเข้ามาใหม่เกิดการสูญหายไป

รูปแบบของคำสั่ง

var = Serial.available()

ค่าพารามิเตอร์ที่ต้องการ

- ไม่มีการส่งค่าใด ๆ ให้กับฟังก์ชั่น

ค่าที่คืนกลับจากฟังก์ชั่น

- var คือตัวแปรแบบอินท์ (int) จะสำหรับใช้รับค่าที่ส่งคืนกลับมาจากฟังก์ชั่นซึ่งเป็นค่าจำนวนข้อมูลที่รับได้จากพอร์ตอนุกรมจากพอร์ตสื่อสารอนุกรม ซึ่งจะถูกเก็บไว้ในบัฟเฟอร์โดยสามารถรับและส่งข้อมูลไว้ได้ 128 ไบต์ ถ้าโปรแกรมไม่สามารถอ่านข้อมูลออกไปประมวลผลได้ทันจะทำให้ข้อมูลที่ส่งมาเกิดการสูญหายได้ แต่ถ้าไม่มีข้อมูลเก็บไว้ในบัฟเฟอร์เลย ค่าที่ส่งคืนจากฟังก์ชั่นจะมีค่าเป็นศูนย์เสมอ

3. int Serial.read()

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้จะใช้ทำหน้าที่ สำหรับสั่งอ่านข้อมูลจากบัฟเฟอร์ของพอร์ตสื่อสารอนุกรมออกมาประมวลผลโดยเมื่ออ่านข้อมูลออกมา 1 ไบต์ ก็จะทำให้พื้นที่บัฟเฟอร์ที่เก็บข้อมูลของพอร์ตสื่อสารอนุกรมว่างลงอีก 1 ไบต์เสมอ ซึ่งโดยค่าของข้อมูลที่อ่านได้จะเป็นข้อมูลที่ถูกเก็บรอไว้ในบัฟเฟอร์เป็นลำดับแรกเมื่ออ่านออกไปแล้ว ข้อมูลถัดไปก็จะถูกเลื่อนตำแหน่งมารอไว้เป็นลำดับแรกแทน โดยปรกติ ค่าของข้อมูลจะมีค่าเป็นเลขจำนวนเต็มระหว่าง 0 – 255 แต่ถ้าไม่มีข้อมูลเก็บไว้ภายในบัฟเฟอร์ค่าข้อมูลที่อ่านได้จะมีค่าเป็น -1 แทน

รูปแบบของคำสั่ง

var = Serial.read()

ค่าพารามิเตอร์ที่ต้องการ

- ไม่มีการส่งค่าใด ๆ ให้กับฟังก์ชั่น

ค่าที่คืนกลับจากฟังก์ชั่น

- var คือ ตัวแปรแบบอินท์ สำหรับใช้รับค่าที่ส่งคืนกลับมาจากฟังก์ชั่น ซึ่งเป็นค่า ข้อมูลที่ถูกเก็บรอไว้ในบัฟเฟอร์เป็นลำดับแรก ปรกติจะมีค่าระหว่าง 0 – 255 และฟังก์ชั่นจะคืนค่าข้อมูลเป็น -1 กลับไปให้ ถ้าถูกสั่งอ่านข้อมูลในขณะที่ไม่มีข้อมูลเก็บไว้ในบัฟเฟอร์เลย

4. Serial.flush()

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ใช้ทำหน้าที่สำหรับสั่งล้างข้อมูลทั้งหมดในบัฟเฟอร์ของพอร์ตสื่อสารอนุกรมให้ว่างลงเพื่อรอรับข้อมูลใหม่ที่จะส่งเข้ามาหลังจากจบการทำงานของคำสั่งนี้เรียบร้อยแล้ว

รูปแบบของคำสั่ง

Serial.flush()

ค่าพารามิเตอร์ที่ต้องการ

- ไม่มีการส่งค่าใด ๆ ให้กับฟังก์ชั่น

ค่าที่คืนกลับจากฟังก์ชั่น

- ไม่มีการส่งค่ากลับ

5. Serial.print(data), Serial.print(data,format), Serial.printIn(data) และ Serial.printIn(data,format)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ทำหน้าที่สำหรับสั่งให้ข้อมูลส่งออกไปยังพอร์ตสื่อสารอนุกรมได้อย่างต่อเนื่อง ในลักษณะของการพิมพ์ โดยข้อมูลที่จะสั่งพิมพ์ด้วยคำสั่งนี้ สามารถเป็นได้ทั้ง ข้อมูล ตัวแปร หรือข้อความต่าง ๆ ที่กำหนดไว้ในพารามิเตอร์ภายในวงเล็บ() ออกทางพอร์ตสื่อสารอนุกรม

รูปแบบของคำสั่ง

Serial.print(data)

Serial.print(data,format)

Serial.printIn(data)

Serial.printIn(data,format)

ค่าพารามิเตอร์ที่ต้องการ

Data คือ ข้อมูลที่ต้องการพิมพ์อาจเป็นข้อความ ตัวแปรในรูปแบบต่างๆ ที่ต้องการจะสั่งพิมพ์

Format คือ รูปแบบของการแปลงข้อมูลที่ซึ่งเป็นค่าของข้อมูล (data) แล้วแปลงค่าตัวอักษร (String) ของค่านั้นๆ โดยสามารถกำหนดรูปแบบในการแปลงข้อมูลเป็นค่าตัวเลขในฐานต่างๆ คือ

BIN เป็นการแปลงค่าข้อมูลเพื่อแสดงผลในรูปแบบเลขฐานสอง (Binary) เช่น ถ้าข้อมูลมีค่า 79 จะแปลงผลเป็นตัวอักษรของตัวเลข “1001111”

DEC เป็นการแปลงค่าข้อมูลเพื่อแสดงผลในรูปแบบเลขฐานสิบ (Decimal) เช่น ถ้าข้อมูลมีค่า 79 จะแปลงผลเป็นตัวอักษรของตัวเลข “79”

HEX เป็นการแปลงค่าข้อมูลเพื่อแสดงผลในรูปแบบเลขฐานสิบหก (Hexadecimal) เช่น ถ้าข้อมูลมีค่า 79 จะแปลงผลเป็นตัวอักษรของตัวเลข “4F”

OCT เป็นการแปลงค่าข้อมูลเพื่อแสดงผลในรูปแบบเลขฐานแปด (Octal) เช่น ถ้าข้อมูลมีค่า 79 จะแปลงผลเป็นอักษรของตัวเลข “117”

BYTE เป็นการแปลงค่าข้อมูลเพื่อแสดงผลในรูปของรหัส ASCII เช่น ถ้าข้อมูลมีค่า 79 จะแปลงผลเป็นตัวอักษรของ “O” ซึ่งรหัสแอสกี้ (ASCII) ของตัวอักษรโอ (“O”) คือ 0x4F หรือ 79

ค่าที่คืนกลับจากฟังก์ชั่น

- ไม่มีการส่งค่ากลับ

3.1.5 คำสั่งพื้นฐานในการเขียนโปรแกรมแสดงผลผ่านหน้าจอแอลซีดีของอาดูโน่

คำสั่ง LCD ของ ET_LCD_4BIT Library

LCD_4BITเป็นไลบรารี่สำหรับใช้ติดต่อสั่งงานกับหน้าจอแสดงผลแอลซีดีแบบชาเรคเตอร์ (Character) ทุกรุ่นที่ใช้ไดร์ฟเวอร์ (Driver) ที่มีคำสั่งเข้ากันได้กับ HD44780 ของบริษัทฮิตาชิ (Hitachi) โดยเชื่อมต่อวงจรของหน้าจอ แอลซีดีแบบ 4 บิต โดยใช้ขาสัญญาณ 6 หรือ 7 ขา ซึ่งผู้ใช้สามารถเลือกกำหนด ใช้ขาใด ๆ ก็ได้โดยอิสระจากคำสั่งในโปรแกรม

```
LCD _4BIT(int rs_pin, int rw_pin, int en_pin,
        Int d4_pin, int d5_pin, int d6_pin,int d7_pin );
LCD _4BIT(int rs_pin, int en_pin,
        Int d4_pin, int d5_pin, int d6_pin,int d7_pin );
```

LCD_4BIT Class

LCD_4BIT ประกอบไปด้วยคำสั่งต่าง ๆ สำหรับใช้ติดต่อสั่งงาน และควบคุมการแสดงผลของหน้าจอแอลซีดีทั้งหมด 14 คำสั่ง โดยมีรายละเอียดการใช้งานดังนี้

- void Initial (void);
- void Command (int value);
- void Print (char);
- void Print (const chart[]);
- void Print (unint8_t);
- void Print (int);
- void Print (unsigned int);
- void Print (long);
- void Print (unsigned long);
- void Print (long, int);
- int ReadDisplay (void);

- int ReadAddress (void);
- void SetCursor (int Cursor);
- void ClearScreen (void);
- void ClearLCD (int Cursor, int length);

1. void Initial (void);

เป็นคำสั่งสำหรับเริ่มการทำงานให้กับหน้าจอ แอลซีดี ซึ่งปรกติจะกระทำการเพียงครั้งแรกครั้งเดียวหลังจากเริ่มต้นการทำงานเท่านั้น ดังนั้นคำสั่งนี้จึงนิยมเขียนไว้ในส่วนของตัวของฟังก์ชั่น Setup() ของโปรแกรมอาดูโน่

โดยการทำงานของคำสั่งนี้เป็นการเริ่มของสัญญาณ (initial) ขาสัญญาณของอาดูโน่ตามที่ถูกกำหนดให้จากผู้ใช้เพื่อใช้ในการติดต่อสั่งงานกับหน้าจอแอลซีดีแบบ 4 บิต พร้อมกับกำหนดการทำงานของหน้าจอ แอลซีดีดังนี้

- ฟังก์ชั่น (function) = 0×28 ซึ่งหมายถึงใช้การเชื่อมต่อแบบ 4 บิต โดยใช้รูปแบบการแสดงผลเป็นแบบ 2 บรรทัด และให้ขนาดตัวอักษรเป็น 5×7 จุด
- การควบคุมการเปิด/ปิด ของหน้าจอแสดงผล (Display On/Off Control) = 0×0C ซึ่งหมายถึงการเปิดหน้าจอ (Display ON) และไม่มีการแสดงเคอร์เซอร์ (Cursor)
- เพิ่มค่าเคอร์เซอร์ (Entry Mode Set) = 0×06 ซึ่งหมายถึง ให้เพิ่มค่าตำแหน่งเคอร์เซอร์ทุกครั้งที่มีการเขียนข้อมูล
- เคลียร์คำสั่งที่หน้าจอ (Clear Display) = 0×01 ซึ่งหมายถึงทำให้เคลียร์ (Clear) หน้าจอการแสดงผลทั้งหมดซึ่งการทำงานของฟังก์ชั่นนี้ จะทำให้หน้าจอแสดงผลของหน้าจอแอลซีดีว่างลงและตำแหน่งเคอร์เซอร์ของหน้าจอแอลซีดีจะชี้อยู่ที่ตำแหน่ง 0×00 ซึ่งเป็นตำแหน่งแสดงผลของตัวอักษรตัวแรกของบรรทัดที่ 1

รูปแบบคำสั่ง

Classname.Initial();

พารามิเตอร์ที่ต้องการ

- ไม่มีการผ่านค่าใด ๆ ให้คำสั่ง

ค่าที่คืนกลับจากคำสั่ง

- ไม่มีการส่งค่าใด ๆ กลับคืนมาจากคำสั่ง

2. void Command (int value)

เป็นคำสั่งสำหรับส่งรหัสคำสั่งควบคุมไปให้กับหน้าจอ แอลซีดี โดยรายละเอียดการทำงานของแต่ละคำสั่งได้อธิบายในส่วนของการสั่งงานหน้าจอ แอลซีดี ในส่วนนี้จะขอแสดงรหัสคำสั่งซึ่งถูกใช้งานบ่อย ๆ เช่น คำสั่งสำหรับสั่งเปลี่ยนการทำงานของหน้าจอ แอลซีดี เช่น การแสดงหรือไม่แสดงเคอร์เซอร์ เป็นต้น

ตารางที่ 3.1 สรุปรหัสคำสั่งของหน้าจอ แอลซีดีที่ใช้งานบ่อย

คำสั่ง	รหัสคำสั่ง	การทำงาน
lcd_clear_screen	0X01	ลบข้อความแสดงผลทั้งหมดในหน้าจอ
lcd_cursor_home	0X02	กำหนด cursor ให้เริ่มที่ตำแหน่งแรกของบรรทัด
lcd_display_on	0X0E	สั่งให้ ON แสดงผลหน้าจอ
lcd_display_off	0X08	สั่งให้ OFF หน้าจอแสดงผล
lcd_cursor_blink	0X0F	สั่งให้ Cursor กระพริบ
lcd_cursor_on	0X0E	สั่งให้เปิดการแสดงผล Cursor
lcd_cursor_off	0X0C	สั่งให้ปิดการแสดงผล Cursor
lcd_cursor_left	0X10	สั่งให้ Cursor เลื่อนไปทางซ้ายของหน้าจอ
lcd_cursor_right	0X14	สั่งให้ Cursor เลื่อนไปทางขวาของหน้าจอ
lcd_display_sleft	0X18	สั่งเลื่อนข้อความแสดงผลไปทางซ้าย 1 ตำแหน่ง
lcd_display_sright	0X1C	สั่งเลื่อนข้อความแสดงผลไปทางซ้าย 1 ตำแหน่ง

รูปแบบคำสั่ง

Classname.Command(value)

พารามิเตอร์ที่ต้องการ

- value คือค่ารหัสคำสั่งสำหรับสั่งงานจอแอลซีดี

ค่าที่กลับคืนจากคำสั่ง

- ไม่มีการส่งค่าใด ๆ กลับคืนมาจากคำสั่ง

3. void Print(value), void Print("str"), void Print(value,format)

เป็นคำสั่งสำหรับส่งค่ารหัสแอสกี้หลายๆ ตัว ในรูปแบบของค่าสตริง หรือค่าชาเรคเตอร์อะเรย์ (Character Array) ไปให้กับหน้าจอ แอลซีดีเพื่อแสดงผล

รูปแบบคำสั่ง

classname.Print(value)

classname.Print("str")

classname.Print(value, format)

พารามิเตอร์ที่ต้องการ

- value คือ ค่าที่ต้องการแสงผลที่หน้าจอแอลซีดีซึ่งสามารถเป็นได้ทั้งข้อมูล, ตัวแปร และค่าคงที่โดยค่าจะถูกแสดงผลเป็นตัวอักษรของค่าตัวเลขในรูปแบบของเลขฐานสิบ (Decimal) เช่น ถ้าข้อมูลมีค่า 79 จะแสดงผลเป็นตัวอักษรของตัวเลข "79"

- "str" คือ ค่าสตริงที่ต้องการแสดงผลเช่น เมื่อสั่งแสดงผลด้วย lcd.print("Hello World") ; ที่หน้าจอแสดงผลแอลซีดีก็จะแสดงข้อความเป็น Hello World

- format คือ รูปแบบของการแปรข้อมูลซึ่งเป็นค่าของ value แล้วแปลงเป็นค่าตัวอักษรของค่านั้นๆ โดยสามารถกำหนดรูปแบบในการแสดงข้อมูลเป็นค่าตัวเลขในฐานต่างๆคือ

BIN เป็นการแปลงค่าข้อมูลเพื่อแสดงผลในรูปแบบเลขฐานสอง (Binary) เช่นถ้าข้อมูลมีค่า 79 จะแสดงผลเป็นตัวอักษรของตัวเลข "1001111"

DEC เป็นการแปลงค่าข้อมูลเพื่อแสดงผลในรูปแบบเลขฐานสิบ (Decimal) เช่น ถ้าข้อมูลมีค่า 79 จะแสดงผลเป็นของตัวเลข "79"

HEX เป็นการแปลงค่าข้อมูลเพื่อแสดงผลในรูปแบบเลขฐานสิบหก (Hexadecimal) เช่น ถ้าข้อมูลมีค่า 79 จะแสดงผลเป็นตัวอักษรของ "4F"

OCT เป็นการแปลงค่าข้อมูลเพื่อแสดงผลในรูปแบบเลขฐานแปด (Octal) เช่น ถ้าข้อมูลมีค่า 79 จะแสดงผลเป็นตัวอักษรของ "117"

BYTE เป็นการแปลงค่าข้อมูลเพื่อแสดงผลในรูปของรหัส ASCII เช่น ถ้าข้อมูลมีค่า 79 จะแสดงผลเป็นตัวอักษรของ "O" ซึ่งรหัสแอสกี้ของตัวอักษร โอ ("O") คือ 0×4F หรือ

BCD ชั้นข้อมูลขนาด 1 ไบต์ มีลักษณะคล้ายเลขฐานสิบหกแต่จะแปลงค่าไบต์ (Byte) เป็นแอสกี้ของเลขฐานสิบหก 2 หลัก ในกรณีที่ค่าไม่เกิน 0×0A จะเติมค่า "0" นำหน้าด้วย

ค่าที่คืนกับการคำสั่ง

- ไม่มีการส่งค่าใด ๆ กลับคืนมาจากคำสั่ง

4. Int ReadDisplay(void)

ใช้สำหรับสั่งอ่านข้อมูลจากหน้าจอ แอลซีดี ณ.ตำแหน่งที่เคอร์เซอร์ชี้อยู่ โดยค่าที่อ่านได้จะเป็นรหัสแอสกี้ที่เขียนไปยังหน้าจอ แอลซีดีแล้ว โดยเมื่อใช้คำสั่งนี้จะทำให้ค่าตำแหน่งเคอร์เซอร์ของหน้าจอแอลซีดีถูกเพิ่มค่าขึ้น 1 ตำแหน่งโดยอัตโนมัติ

คำสั่งนี้สามารถใช้ได้กับหน้าจอแอลซีดี ที่ต่อวงจรแบบ 4 บิต โดยใช้ขาสัญญาณ 7 ขา เท่านั้นถ้าใช้คำสั่งนี้กับวงจรที่ต่อโดยใช้ 6 ขา ค่าที่อ่านกลับจากคำสั่งจะได้ค่ารหัสแอสกี้ของ "?" แทน

รูปแบบคำสั่ง

val = classname.ReadDisplay()

พารามิเตอร์ที่ต้องการ

- ไม่มีการส่งค่าใด ๆ ให้กับฟังก์ชั่น

ค่าที่คืนกับจากคำสั่ง

- val เป็นตัวแปรแบบอินท์ ซึ่งต้องสร้างไว้สำหรับรองรับค่าของข้อมูลที่ต้องการอ่านจากหน้าจอแอลซีดี ซึ่งค่าที่อ่านได้จะเป็นค่ารหัสแอสกี้ของหน้าจอแอลซีดี ณ. ตำแหน่งที่เคอร์เซอร์ชี้อยู่ก่อนใช้คำสั่งนี้ โดยคำสั่งนี้สามารถใช้ได้กับหน้าจอแอลซีดีที่ต่อวงจรแบบ 4 บิต โดยใช้สัญญาณในการเชื่อมต่อ 7 ขาเท่านั้น ซึ่งถ้าใช้คำสั่งนี้กับหน้าจอแอลซีดีที่ต่อวงจรแบบ 4 บิต โดยใช้สัญญาณ 6 ขา (ไม่ใช้ R/W) ค่าที่อ่านได้จะเป็นแอสกี้เป็น 0X3F ซึ่งเป็นรหัสแอสกี้ของตัวอักษร "?" แทน

5. int ReadAddress(void)

ใช้สำหรับใช้สำหรับสั่งอ่านค่าแอดแดรส (Address) ของซีจี แรม (CG RAM) และดีดี แรม (DD RAM) (ตำแหน่งเคอร์เซอร์) จากหน้าจอแอลซีดีในขณะนั้นๆ คำสั่งนี้จะสามารถใช้กับหน้าจอแอลซีดีที่ต่อวงจรแบบ 4 บิต โดยจะใช้ขาสัญญาณ 7 ขาเท่านั้นถ้าใช้คำสั่งนี้กับวงจรที่ต่อโดยใช้สัญญาณ 6 เส้น ค่าที่อ่านกลับจากคำสั่งจะได้ค่า 0X00 เสมอ

รูปแบบคำสั่ง

val = classname.ReadAddress()

พารามิเตอร์ที่ต้องการ

- ไม่มีการส่งค่าใด ๆ ให้กับฟังก์ชั่น

ค่าที่คืนกับจากคำสั่ง

- val เป็นตัวแปรแบบอินท์ซึ่งจะต้องสร้างไว้สำหรับรองรับค่าแอดเดรสของ ซีจีแรม หรือดีดีแรม ที่ต้องการอ่านจากหน้าจอแอลซีดี ซึ่งค่าที่อ่านได้จะเป็นค่าตำแหน่งของแอดเดรส (ตำแหน่งเคอร์เซอร์) ของหน้าจอแอลซีดีในขณะที่สั่งอ่าน โดยคำสั่งนี้สามารถนี้สามารถใช้ได้กับหน้าจอแอลซีดีที่ต่อวงจรแบบ 4 บิต โดยใช้สัญญาณในการเชื่อมต่อ 7 ขาเท่านั้นถ้าใช้คำสั่งนี้กับหน้าจอแอลซีดีที่ต่อวงจรแบบ 4 บิตโดยใช้สัญญาณ 6 ขา (ไม่ใช้ R/W) ค่าที่อ่านได้จะเป็นรหัสแอสกี้เป็น 0X00 เสมอ

6. void SetCursor(int Cursor)

ใช้สำหรับกำหนดตำแหน่งเคอร์เซอร์ ของการแสดงผลหน้าจอให้กับหน้าจอแอลซีดี ซึ่งความจริงแล้วคำสั่งนี้ก็ คือ คำสั่ง “SET DD RAM ADDRESS” ของหน้าจอ แอลซีดีนั้นเอง

รูปแบบคำสั่ง

classname.SetCursor (Cursor)

พารามิเตอร์ที่ต้องการ

- Cursor คือ ค่าตำแหน่งเคอร์เซอร์สำหรับเริ่มต้นแสดงผลหน้าจอแอลซีดีซึ่งต้องกำหนดค่าตำแหน่งของหน้าจอแอลซีดีตามค่าตำแหน่งของ แอดเดรสดีดีแรมของหน้าจอแอลซีดีจริงๆ ด้วย โดยการกำหนดค่าของตำแหน่งเคอร์เซอร์นี้เป็นการกำหนดตำแหน่งเคอร์เซอร์ในขณะนั้นๆ ให้กับหน้าจอแอลซีดี โดยเมื่อมีการเขียนข้อมูลให้หน้าจอแอลซีดีแต่ละครั้งค่าเคอร์เซอร์นี้จะเพิ่มค่าขึ้นครั้งละ 1 ค่าเสมอ ซึ่งค่าตำแหน่งของเคอร์เซอร์รุ่นต่างๆ นั้นขอให้ดูจากคู่มือของ หน้าจอแอลซีดีประกอบด้วยดังตัวอย่าง

00	01	02	03	04	05	06	07	40	41	42	43	44	45	46	47

ตำแหน่ง DD RAM ของ LCD 16 ตัวอักษร 1 บรรทัด

00	01	02	03	04	05	06	07	40	41	42	43	44	45	46	47
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

ตำแหน่ง DD RAM ของ LCD 16 ตัวอักษร 2 บรรทัด

00	01	02	03	04	05	06	07	40	41	42	43	44	45	46	47
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F

ตำแหน่ง DD RAM ของ LCD 16 ตัวอักษร 4 บรรทัด

ค่าที่คืนกับจากคำสั่ง

- ไม่มีการส่งค่าใด ๆ กลับคืนมาจากคำสั่ง

7. void ClearScreen(void)

ใช้สำหรับลบข้อความทั้งหมดที่หน้าจอ หรือทำการสั่งเคลียร์หน้าจอการแสดงผลทั้งหมด ในขณะนั้นอันที่จริงแล้วคำสั่งนี้ก็คือคำสั่ง (Command) (0×01) นั้นเอง

รูปแบบคำสั่ง

classname.ClearScreen()

พารามิเตอร์ที่ต้องการ

- ไม่มีการผ่านค่าใด ๆ ให้คำสั่ง

ค่าที่คืนกับจากคำสั่ง

- ไม่มีการส่งค่าใด ๆ กลับคืนมาจากคำสั่ง

8. void ClearLCD(int Cursor, int length)

ใช้สำหรับสั่งลบข้อความที่หน้าจอแอลซีดีตั้งแต่ตำแหน่งของเคอร์เซอร์ที่กำหนด ไปเป็นจำนวนเท่ากับขนาดที่กำหนดในขนาด (Length) พร้อมกำหนดตำแหน่งของเคอร์เซอร์ที่ระบุไว้ด้วย

รูปแบบคำสั่ง

Classname.ClearLCD(Cursor,length)

พารามิเตอร์ที่ต้องการ

- เคอร์เซอร์ คือ ตำแหน่งเริ่มต้นเคอร์เซอร์ที่ต้องการลบหน้าจอแสดงผล

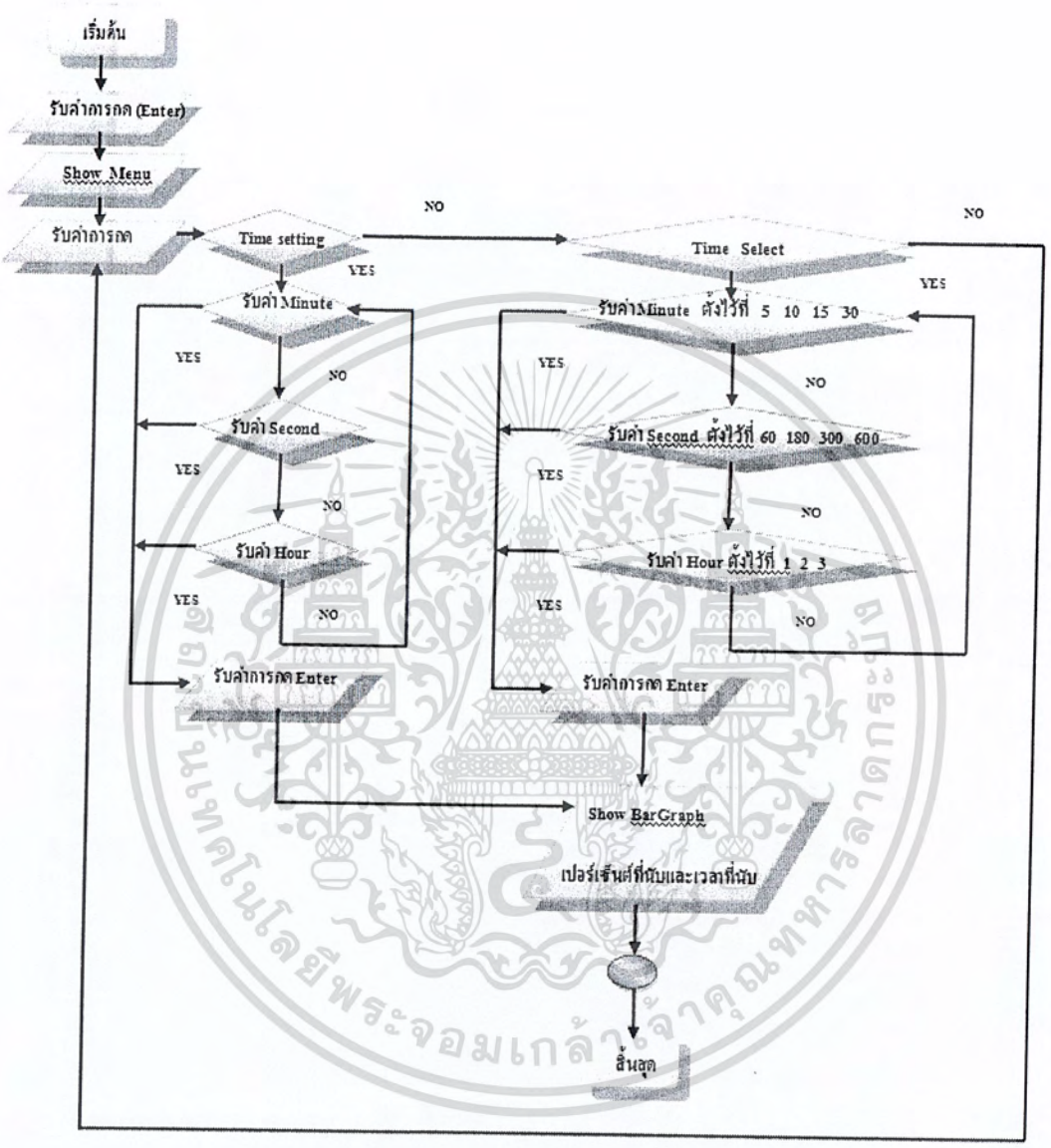
- Length คือ ขนาดจำนวนตัวอักษรที่ต้องการลบ หน้าจอแสดงผล โดยเริ่มนับตำแหน่งแรกที่เคอร์เซอร์เช่น ถ้ากำหนดขนาดของหน้าจอ = 16 โดยกำหนดค่าเคอร์เซอร์ = 0X00 ก็จะหมายถึงให้ลบข้อความหน้าจอจำนวน 16 ตัวอักษร โดยเริ่มที่ตำแหน่งอักษร (0X00) เป็นต้นไป

ค่าที่คืนกับจากคำสั่ง

- ไม่มีการส่งค่าใด ๆ กลับคืนมาจากคำสั่ง

3.2 การออกแบบการแสดงผลผ่านหน้าจอ แอลซีดี

โฟล์วชาร์ต (Flow Chart)



บทที่ 4

การทดลองและผลการทดลอง

4.1 จุดประสงค์ในการทดลอง

1. ทดลองเขียนโปรแกรมเพื่อควบคุมการทำงานของไมโครคอนโทรลเลอร์อาดูโน่

2. ทดลองเขียนโปรแกรมเพื่อติดต่อโปรแกรมซีเรียลมอนิเตอร์ (Serial Moniter) ของไมโครคอนโทรลเลอร์

3. ทดลองเขียนโปรแกรมเพื่อให้แสดงผลผ่านหน้าจอแอลซีดี

4. ทดสอบโปรแกรมที่เขียนขึ้นเพื่อให้สามารถแสดงผลผ่านหน้าจอแอลซีดีให้มีผลตรงตามที่ได้ออกแบบไว้

4.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดลองเขียนโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์อาดูโน่

1. เขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์อาดูโน่โดยให้แอลอีดี (LED) เป็นตัวแสดงผล

2. บันทึกผลการทดลอง

การทดลองที่ 2 ทดลองเขียนโปรแกรมแสดงผลผ่านหน้าจอแอลซีดี

1. เขียนโปรแกรมแสดงผลผ่านหน้าจอแอลซีดีโดยให้ Arduino Serial LCD Keypad Shield เป็นตัวแสดงผล

2. บันทึกผลการทดลอง

การทดลองที่ 3 ทดลองเขียนโปรแกรมการติดต่อโปรแกรมซีเรียลมอนิเตอร์ของไมโครคอนโทรลเลอร์อาดูโน่

1. เขียนโปรแกรมติดต่อโปรแกรมซีเรียลมอนิเตอร์ของไมโครคอนโทรลเลอร์อาดูโน่

2. บันทึกผลการทดลอง

การทดลองที่ 4 ทดสอบโปรแกรมที่เขียนขึ้นเพื่อแสดงผลผ่านหน้าจอแอลซีดีมีผลตรงตามที่ได้ออกแบบ

1. เขียนโปรแกรมแสดงผลผ่านหน้าจอแอลซีดี ตามที่ได้ออกแบบไว้โดยให้ Arduino Serial LCD Keypad Shield เป็นตัวแสดงผล

2. ทดลองโปรแกรมค่าเวลาให้กับ Arduino Serial LCD Keypad Shield

3. บันทึกผลการทดลอง

4.3 ผลการทดลอง

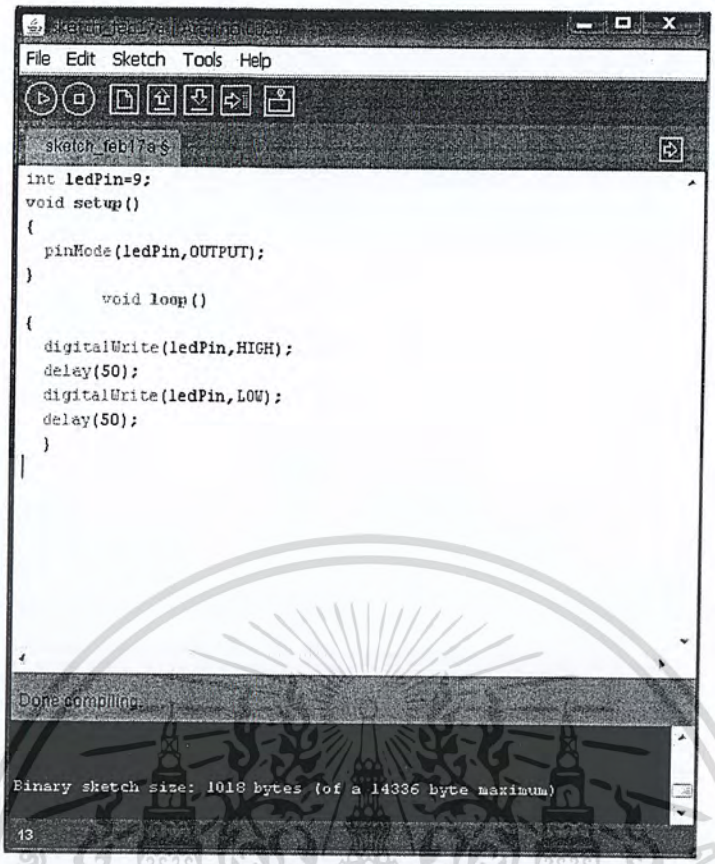
การทดลองที่ 1 ทดลองเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์อาดูโน่ โดยแสดงผลที่แอลอีดี

ผลการทดลอง

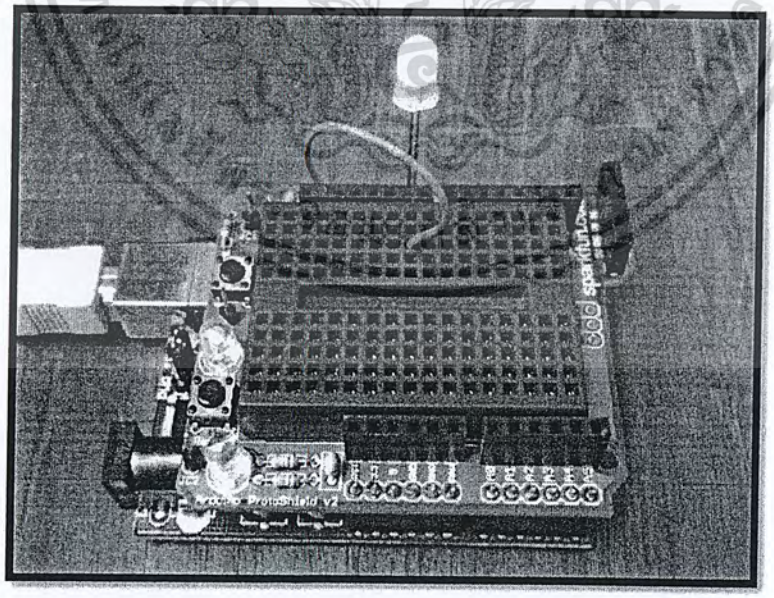
```
int ledPin=9;
void setup()
{
pinMode(ledPin,OUTPUT);
}
void loop()
{
digitalWrite(ledPin,HIGH);
delay(50);
digitalWrite(ledPin,LOW);
delay(50);
}
```

การทดลองนี้ จะเพิ่มเติมรูปแบบคำสั่งดีเลย์ (delay()) ลงไป และให้มีการป้อนอินพุตลอจิก (input logic) "HIGH" และ "LOW" สลับกันไปโดย การติด-ดับจะเร็วช้าขึ้นกับคำสั่งดีเลย์

ผลลัพธ์ : LED จะติดดับห่างกันด้วยความเร็ว 50 มิลลิวินาทีทดลองเปลี่ยนค่าความเร็วของคำสั่งดีเลย์



ภาพที่ 4.1 แสดงโปรแกรมควบคุมไมโครคอนโทรลเลอร์ อาดูโน่โดยแสดงผลที่แอลอีดี



ภาพที่ 4.2 แสดงผลการรันโปรแกรมควบคุมไมโครคอนโทรลเลอร์ อาดูโน่
โดยแสดงผลที่แอลอีดี

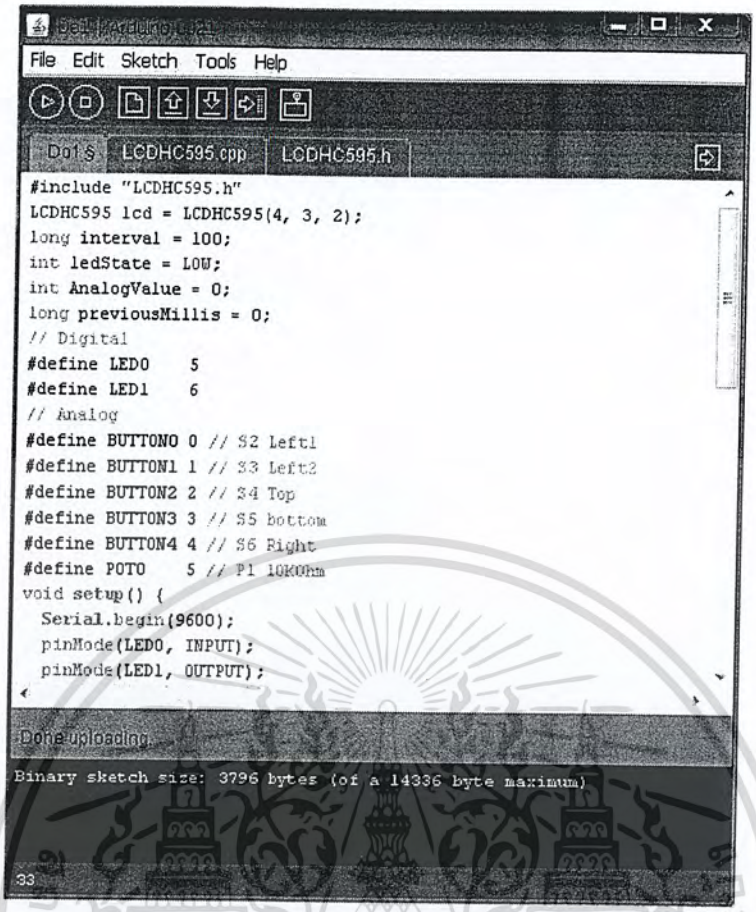
การทดลองที่ 2 ทดลองเขียนโปรแกรมแสดงผลผ่านหน้าจอแอลซีดี

ผลการทดลอง

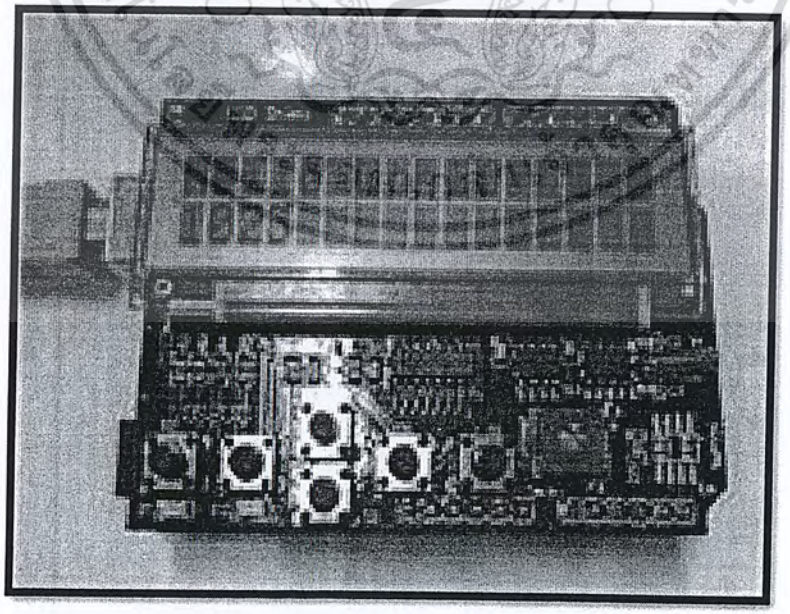
```
#include "LCDHC595.h"
LCDHC595 lcd = LCDHC595(4, 3, 2);
long interval = 100;
int ledState = LOW;
int AnalogValue = 0;
long previousMillis = 0;
// Digital
#define LED0    5
#define LED1    6
// Analog
#define BUTTON0 0 // S2 Left1
#define BUTTON1 1 // S3 Left2
#define BUTTON2 2 // S4 Top
#define BUTTON3 3 // S5 bottom
#define BUTTON4 4 // S6 Right
#define POT0    5 // P1 10KOhm
void setup() {
 Serial.begin(9600);
 pinMode(LED0, INPUT);
 pinMode(LED1, OUTPUT);
digitalWrite(LED1, LOW);
}
void loop()
{
 lcd.Initial();
 lcd.ClearScreen();
```

```
lcd.Print("Hello World");
while(1)
{
 lcd.SetCursor(0xC0);
 AnalogValue = analogRead(POT0);
 lcd.Print((int)AnalogValue);
 if (millis() - previousMillis > interval)
 {
   previousMillis = millis();
   if (ledState == LOW)
 {
   ledState = HIGH;
   digitalWrite(LED1, LOW);
 }
 else
 {
   ledState = LOW;
   digitalWrite(LED1, HIGH);
 }
}
}
}
```

ผลลัพธ์: หน้าจอแอลซีดี จะแสดง คำว่า “Hello World” ที่ตำแหน่ง 0x00 คือบรรทัดที่ 1 และ เลข 1023 ที่บรรทัดที่ 2 นั้นเอง



ภาพที่ 4.3 แสดงโปรแกรมแสดงผลผ่านหน้าจอ แอลซีดี



ภาพที่ 4.4 แสดงผลการรันโปรแกรมแสดงผลผ่านหน้าจอ แอลซีดี

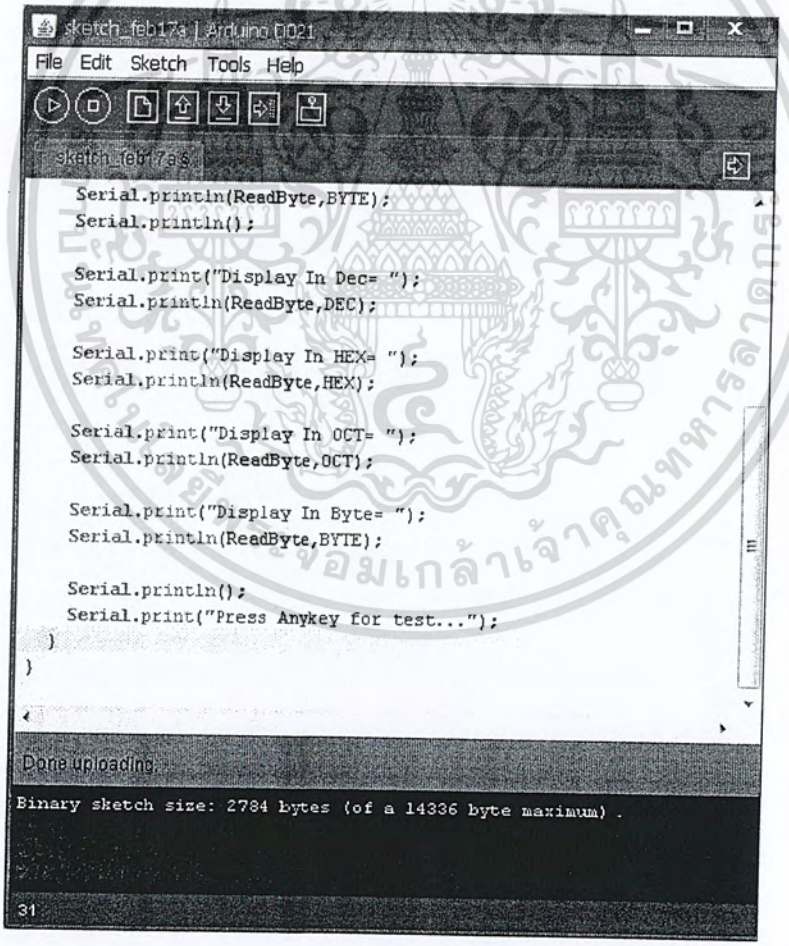
การทดลองที่ 3 ทดลองเขียนโปรแกรมการติดต่อโปรแกรมซีเรียลมอนิเตอร์ของไมโครคอนโทรลเลอร์ อาดูโน่

ผลการทดลอง

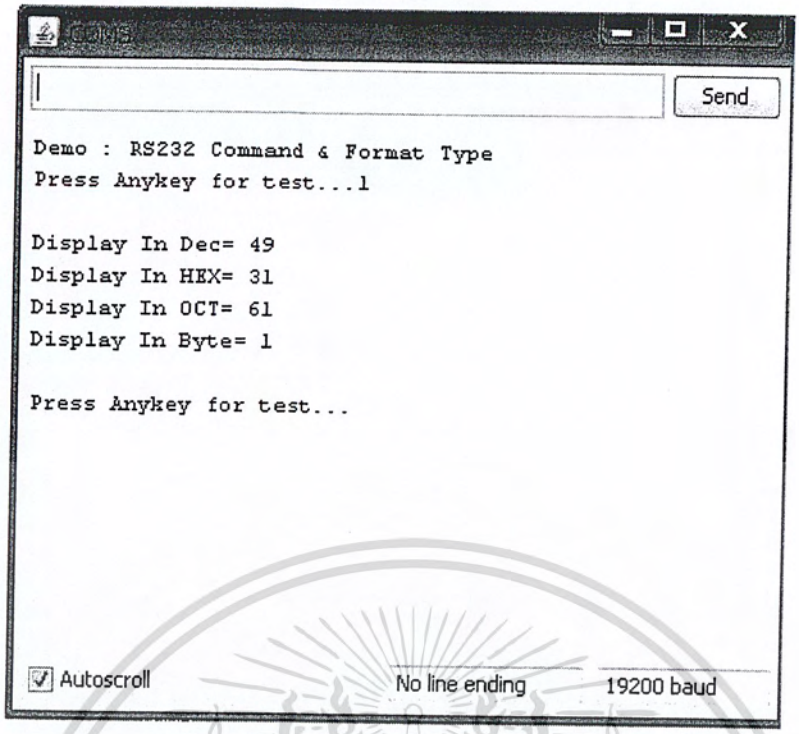
```
int ReadByte = 0;
void setup()
{
 Serial.begin(19200);
 Serial.flush();
 Serial.println("Demo : RS232 Command & Format Type");
 Serial.print("Press Anykey for test...");
}
void loop()
{
 if(Serial.available()>0)
 {
  ReadByte=Serial.read();
  Serial.println(ReadByte,BYTE);
  Serial.println();

  Serial.print("Display In Dec= ");
  Serial.println(ReadByte,DEC);

  Serial.print("Display In HEX= ");
  Serial.println(ReadByte,HEX);

  Serial.print("Display In OCT= ");
  Serial.println(ReadByte,OCT);
```

```
    Serial.print("Display In Byte= ");
    Serial.println(ReadByte,BYTE);

    Serial.println();
    Serial.print("Press Anykey for test...");
  }
}
```

ผลลัพธ์ : เมื่อคีย์ค่าที่คีย์บอร์ดโดยที่ตัวแปรนั้นต้องเป็นตัวเลข ผลที่ได้จะถูกแปลงเป็นค่าเลขฐานสิบ (Decimal), เลขฐานสิบหก (Hexadecimal), เลขฐานแปด (Octal), รหัสแอสกี้



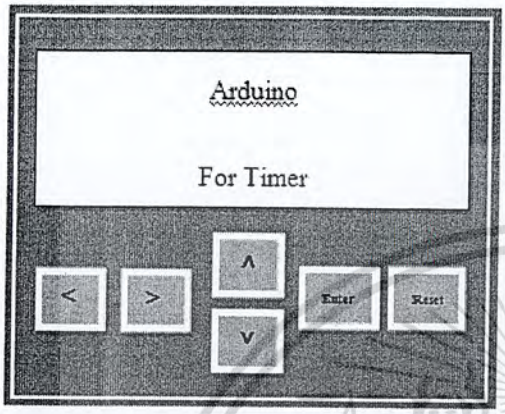
ภาพที่ 4.5 แสดงโปรแกรมการติดต่อโปรแกรมซีเรียลมอนิเตอร์ของไมโครคอนโทรลเลอร์ อาดูโน่



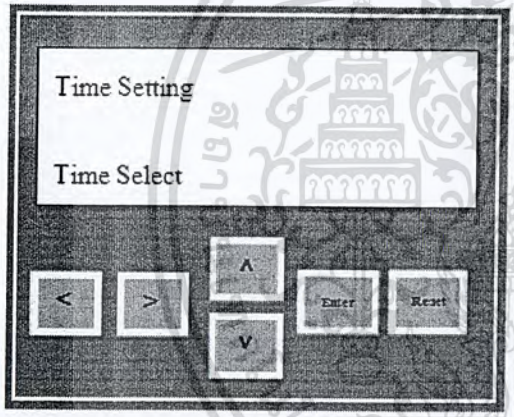
ภาพที่ 4.6 แสดงผลการรันโปรแกรมการติดต่อโปรแกรมซีเรียลมอนิเตอร์ของไมโครคอนโทรลเลอร์อาดูโน่

การทดลองที่ 4 ทดสอบโปรแกรมที่เขียนขึ้นเพื่อแสดงผลผ่านหน้าจอแอลซีดี มีผลตรงตามที่ได้ออกแบบ

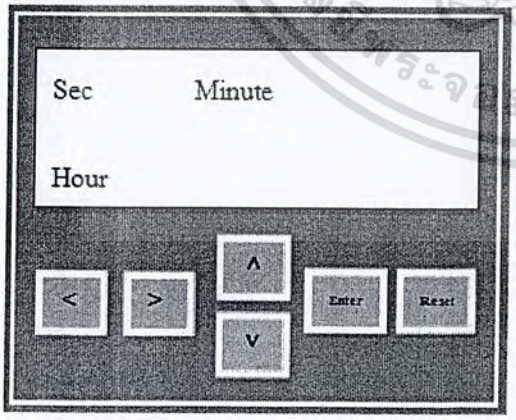
ผลการเขียนโปรแกรมเพื่อแสดงผลผ่านหน้าจอ แอลซีดี ตามที่ได้ออกแบบไว้



หน้าจอเริ่มต้น
แสดงผลเมื่อเปิดโปรแกรมซึ่งจะแสดง ผลอยู่ประมาณ 5 วินาทีและจะเข้าสู่หน้าถัดไปโดยอัตโนมัติ



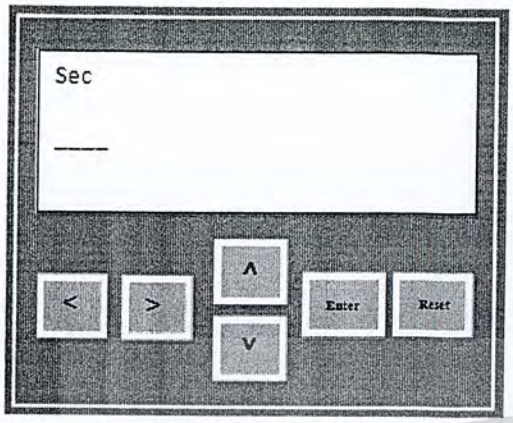
หน้าจอที่ 1 [ซึ่งจะเข้าสู่หน้าจอนี้โดยอัตโนมัติเมื่อเปิดโปรแกรม] แสดงรูปแบบของการตั้งค่าเวลาซึ่งจะมีให้เลือก 2 โหมด โดยการกดปุ่มขึ้น (Λ) หรือปุ่มลง (V) เพื่อเลือกโหมดที่ต้องการใช้ต่อไป ซึ่งมีดังนี้
- โหมดไทม์เซ็ตติ้ง (Time Setting) เป็นการตั้งค่าเวลาโดยผู้ใช้กำหนดค่าเป้าหมายเอง
- โหมดไทม์ซีเล็กต์ (Time Select) เป็นการตั้งค่าเวลาโดยเลือกค่าเป้าหมายที่มีอยู่แล้ว



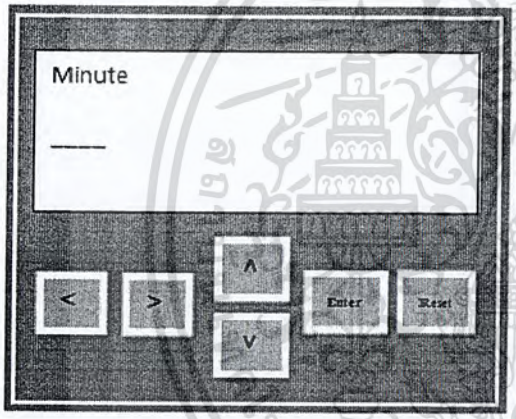
หน้าจอที่ 2 [ซึ่งจะเข้าสู่หน้านี้ได้โดยผ่านการเลือกจากหน้าจอที่ 1 ก่อน] โดยไม่ว่าจะเลือกโหมดไทม์เซ็ตติ้ง (Time Setting) หรือโหมดไทม์ซีเล็กต์ (Time Select) ก็จะต้องเข้าสู่หน้าจอที่ 2 เพื่อเลือกหน่วยของเวลา โดยการกดปุ่มซ้าย (<) หรือปุ่มขวา (>) ซึ่งจะมีหน่วยเวลาให้เลือกดังนี้

- SEC	มีหน่วยเวลาเป็น วินาที
- MINUTE	มีหน่วยเวลาเป็น นาที
- HOUR	มีหน่วยเวลาเป็น ชั่วโมง

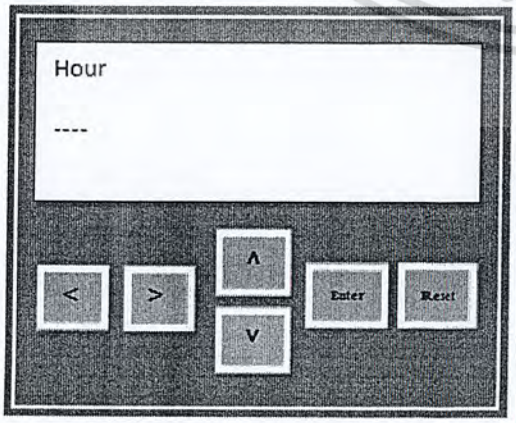
การตั้งค่าแบบไทม์เซ็ตติ่ง (Time Setting)



หน้าจอที่ 3.1 [ซึ่งจะเข้าสู่หน้าจอนี้โดยผ่านการเลือกจากหน้าจอที่ 2 ก่อน] ในกรณีนี้ต้องเลือกที่โหมดไทม์เซ็ตติ่ง (Time Setting) และต่อจากนั้นเลือกโหมดวินาที (Sec) แล้วจึงทำการป้อนค่าเป้าหมายที่ต้องการโดยการกดปุ่มซ้าย (<) หรือปุ่มขวา (>) เพื่อเลือกหลักและกดปุ่มขึ้น (Λ) หรือปุ่มลง (V) เพื่อเลือกค่าเวลาที่ต้องการซึ่งมีย่านจะการใช้งานคือ 0 วินาทีถึง 999 วินาทีเมื่อเลือกค่าที่ต้องการแล้วทำการกดปุ่มเอ็นเตอร์ (Enter)

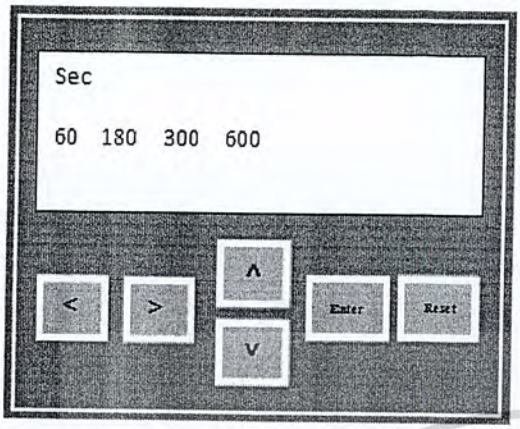


หน้าจอที่ 3.2 [ซึ่งจะเข้าสู่หน้าจอนี้โดยผ่านการเลือกจากหน้าจอที่ 2 ก่อน] ในกรณีนี้ต้องเลือกที่โหมดไทม์เซ็ตติ่ง (Time Setting) และต่อจากนั้นเลือกโหมดนาที (Minute) แล้วจึงทำการป้อนค่าเป้าหมายที่ต้องการโดยการกดปุ่มซ้าย (<) หรือปุ่มขวา (>) เพื่อเลือกหลักและกดปุ่มขึ้น (Λ) หรือปุ่มลง (V) เพื่อเลือกค่าเวลาที่ต้องการซึ่งมีย่านจะการใช้งานคือ 0 นาทีถึง 99 นาทีเมื่อเลือกค่าที่ต้องการแล้วทำการกดปุ่มเอ็นเตอร์

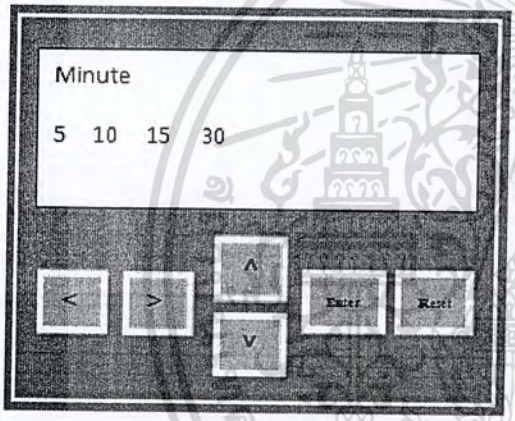


หน้าจอที่ 3.3 [ซึ่งจะเข้าสู่หน้าจอนี้โดยผ่านการเลือกจากหน้าจอที่ 2 ก่อน] ในกรณีนี้ต้องเลือกที่โหมดไทม์เซ็ตติ่ง(Time Setting) เลือกโหมดชั่วโมง (Hour) จากนั้นป้อนค่าเป้าหมายที่ต้องการโดยการกดปุ่มซ้าย (<) หรือปุ่มขวา (>) เพื่อเลือกหลักและกดปุ่มขึ้น (Λ) หรือปุ่มลง (V) เพื่อเลือกค่าเวลาที่ต้องการซึ่งมีย่านจะการใช้งานคือ 0 ชั่วโมงถึง 9 ชั่วโมง เมื่อเลือกค่าที่ต้องการแล้วทำการกดปุ่มเอ็นเตอร์

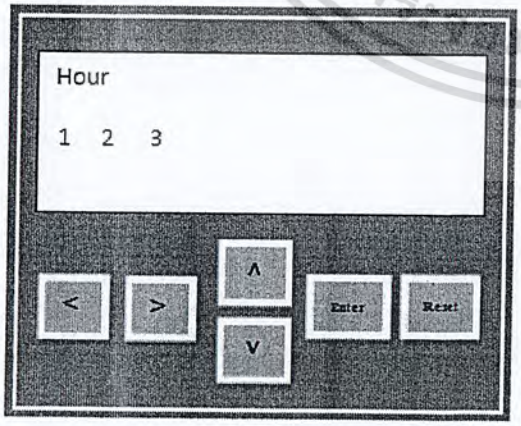
การตั้งค่าแบบไทม์ซีเล็กต์ (Time Select)



หน้าจอที่ 3.4 [ซึ่งจะเข้าสู่หน้าจอนี้โดยผ่านการเลือกจากหน้าจอที่ 2 ก่อน] ในกรณีนี้ต้องเลือกที่โหมดไทม์ซีเล็กต์ (Time Select) และต่อจากนั้นเลือกโหมดวินาที (Sec) จากนั้น ทำการป้อนค่าเป้าหมายที่ต้องการโดยการกดปุ่ม Left (<) หรือปุ่ม Right (>) เพื่อเลือกค่าเวลาที่ต้องการซึ่งมีย่านของการใช้งานคือ 60 180 300 600 วินาที ซึ่งเป็นค่าที่มีการกำหนดไว้แล้ว เมื่อเลือกค่าที่ต้องการแล้วทำการกดปุ่ม

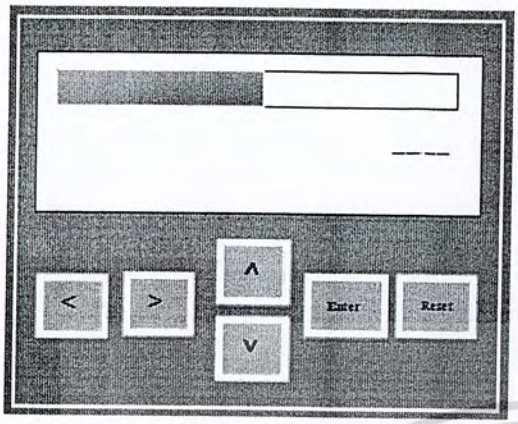


หน้าจอที่ 3.5 [ซึ่งจะเข้าสู่หน้าจอนี้โดยผ่านการเลือกจากหน้าจอที่ 2 ก่อน] ในกรณีนี้ต้องเลือกที่โหมดไทม์ซีเล็กต์ (Time Select) ต่อจากนั้นทำการเลือกโหมดนาที (Minute) และจึงทำการป้อนค่าเป้าหมายที่ต้องการโดยการกดปุ่ม Left (<) หรือปุ่ม Right (>) เพื่อเลือกค่าเวลาที่ต้องการซึ่งมีย่านของการใช้งานคือ 5 10 15 30 นาที ซึ่งเป็นค่าที่มีการกำหนดไว้แล้ว เมื่อเลือกค่าที่ต้องการแล้วทำการกดปุ่มเอ็นเตอร์



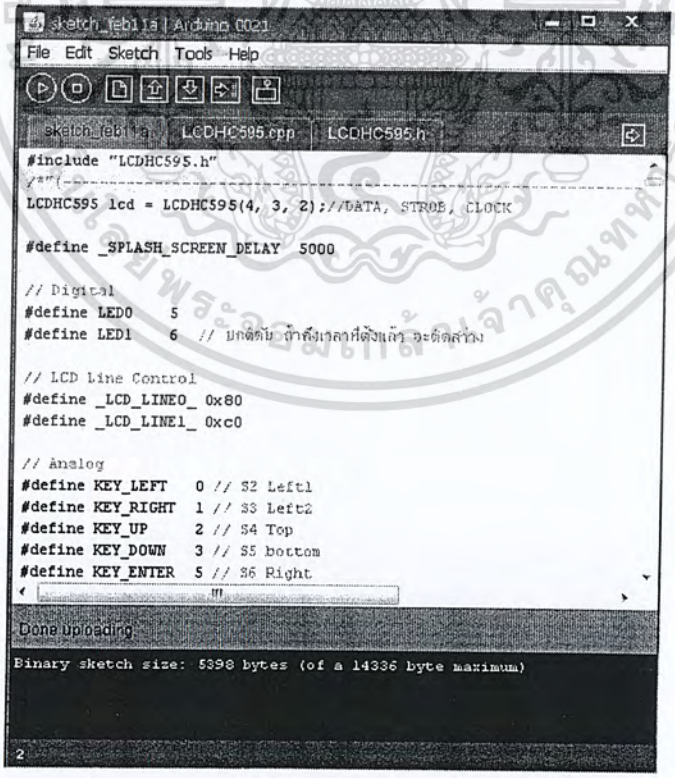
หน้าจอที่ 3.6 [ซึ่งจะเข้าสู่หน้าจอนี้โดยผ่านการเลือกจากหน้าจอที่ 2 ก่อน] ในกรณีนี้ต้องเลือกที่โหมดไทม์ซีเล็กต์ (Time Select) และทำการเลือกโหมดชั่วโมง (Hour) จากนั้นทำการป้อนค่าเป้าหมายที่ต้องการโดยการกดปุ่ม Left (<) หรือปุ่ม Right (>) เพื่อเลือกค่าเวลาที่ต้องการซึ่งมีย่านของการใช้งานคือ 1 2 3 ชั่วโมง ซึ่งเป็นค่าที่มีการกำหนดไว้แล้ว เมื่อเลือกค่าที่ต้องการแล้วทำการกดปุ่มเอ็นเตอร์

การแสดงผลของการนับเวลา



หน้าจอที่ 5 [ซึ่งจะเข้าสู่หน้าจอนี้โดยผ่านการกดปุ่มเอ็นเตอร์ (Enter) จากหน้าจอที่ 3 หรือ หน้าจอที่ 4 ก่อน] ในกรณีนี้เมื่อตั้งค่าเป้าหมายหรือเวลาที่ต้องการแล้วจึงกดปุ่มเอ็นเตอร์ (Enter) เพื่อรัน (Run) โปรแกรมซึ่งจะทำการประมวลผลและนับเวลาตามที่ตั้งค่าไว้ และการแสดงผลจะแบ่งออกเป็น 3 ส่วนด้วยกันคือ

- แสดงผลการนับเวลาเป็นเปอร์เซ็นต์ (%)
- แสดงผลการนับเวลาในรูปแบบของบาร์กราฟ ซึ่งเกิดจากการแปลงการนับเวลาในรูปแบบเปอร์เซ็นต์
- แสดงเป็นตัวเลข คือ เป็นเวลาที่นับได้

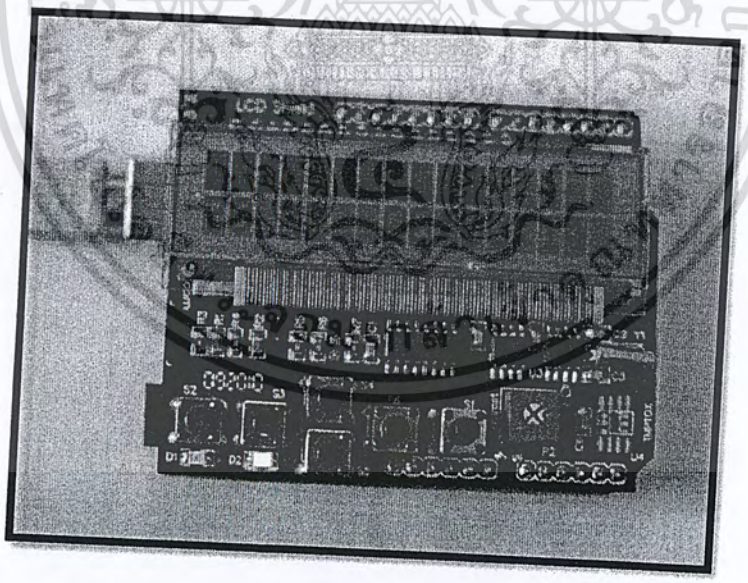


ภาพที่ 4.7 แสดงโปรแกรมที่เขียนขึ้นเพื่อแสดงผลผ่านหน้าจอ แอลซีดี

ผลทดสอบการคีย์ค่าเวลา

ตารางที่ 4.1 แสดงผลการเปรียบเทียบค่าเวลา (วินาที) ระหว่างหน้าจอ (Display) โปรแกรมซีเรียลมอนิเตอร์กับค่าเวลาจริง

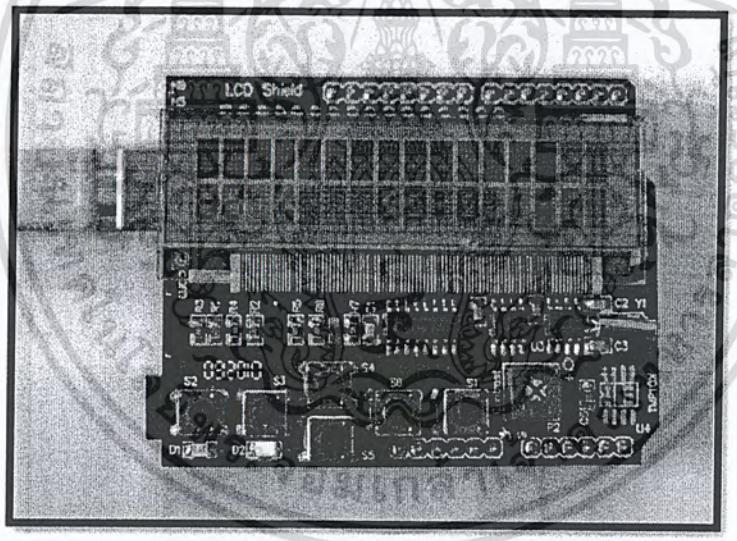
เวลา (วินาที)	เวลา		(Error)	
	เวลา (Display)	เวลา (โปรแกรม Serial Monitor)	Error (Display)	Error (โปรแกรม Serial Monitor)
0	00:00:00	00:00:00	00:00:00	00:00:00
20	00:0020	00:0020	00:00:00	00:00:00
40	00:00:40	00:00:40	00:00:00	00:00:00
60	00:01:00	00:01:00	00:00:00	00:00:00
80	00:01:20	00:01:20	00:00:00	00:00:00
100	00:01:40	00:01:40	00:00:00	00:00:00



ภาพที่ 4.8 แสดงผลการนับเวลาที่ 70 เปอร์เซ็นต์ของ เวลา 20 วินาที

ตารางที่ 4.2 แสดงผลการเปรียบเทียบค่าเวลา (นาที) ระหว่างหน้าจอ (Display) โปรแกรมซีเรียลมอนิเตอร์กับค่าเวลาจริง

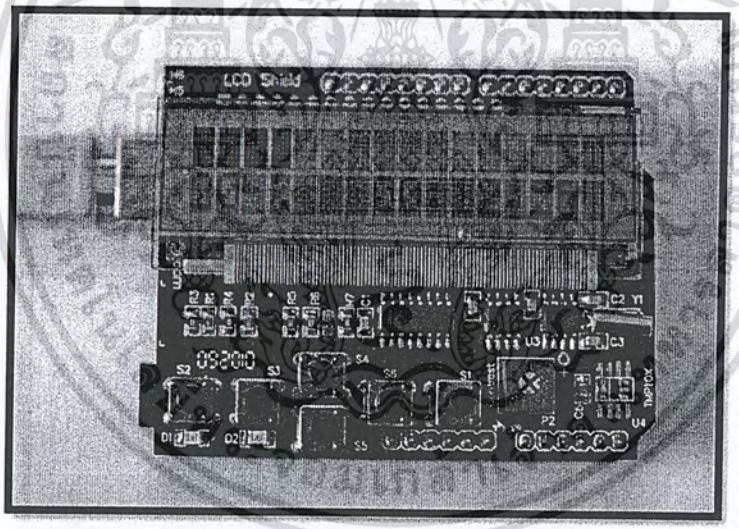
เวลา (นาที)	เวลา		(Error)	
	เวลา (Display)	เวลา (โปรแกรม Serial Monitor)	Error (Display)	Error (โปรแกรม Serial Monitor)
0	00:00:00	00:00:00	00:00:00	00:00:00
12	00:12:00	00:12:00	00:00:00	00:00:00
24	00:24:00	00:24:00	00:00:00	00:00:00
36	00:36:00	00:36:00	00:00:00	00:00:00
48	00:48:00	00:48:00	00:00:00	00:00:00
60	01:00:00	01:00:00	00:00:00	00:00:00



ภาพที่ 4.9 แสดงผลการนับเวลาที่ 88 เปอร์เซ็นต์ของ เวลา 2 นาที

ตารางที่ 4.3 แสดงผลการเปรียบเทียบค่าเวลา (ชั่วโมง) ระหว่างหน้าจอ (Display) โปรแกรมซีเรียลมอนิเตอร์กับค่าเวลาจริง

เวลา (ชั่วโมง)	เวลา		(Error)	
	เวลา (Display)	เวลา (โปรแกรม Serial Monitor)	Error (Display)	Error (โปรแกรม Serial Monitor)
0	00:00:00	00:00:00	00:00:00	00:00:00
1	01:00:00	01:00:00	00:00:00	00:00:00
2	02:00:00	02:00:00	00:00:00	00:00:00
3	03:00:00	03:00:00	00:00:00	00:00:00
4	04:00:00	04:00:00	00:00:00	00:00:00
5	05:00:00	05:00:00	00:00:00	00:00:00

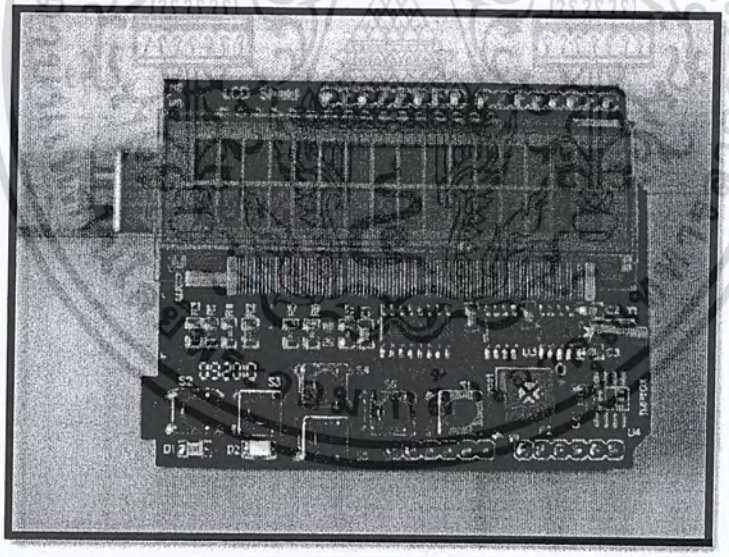


ภาพที่ 4.10 แสดงผลการนับเวลาที่ 20 เปอร์เซ็นต์ของ เวลา 1 ชั่วโมง

ผลการคำนวณเปอร์เซ็นต์ให้แสดงผลเป็นบาร์กราฟ

ตารางที่ 4.4 แสดงผลการคำนวณเปอร์เซ็นต์ให้แสดงผลเป็นบาร์กราฟเปรียบเทียบระหว่างหน้าจอ (Display), โปรแกรมซีเรียลมอนิเตอร์กับค่าที่ได้จากการคำนวณที่หน่วยเวลาเป็นวินาที

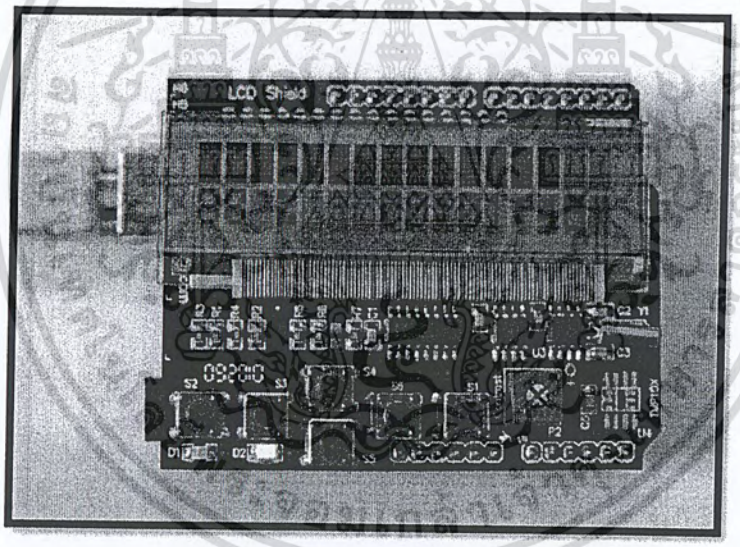
เวลา (วินาที)	Bar graph			Error	
	Bar graph (คำนวณ)	Bar graph (Display)	Bar graph (โปรแกรม Serial Monitor)	Error (Display)	Error (โปรแกรม Serial Monitor)
0				0	0
20	****	****	****	0	0
40	*******	*******	*******	0	0
60	**********	**********	**********	0	0
80	**********	**********	**********	0	0
100	**********	**********	**********	0	0



ภาพที่ 4.11 แสดงผลการคำนวณเปอร์เซ็นต์ให้แสดงผลเป็นบาร์กราฟที่ 70 เปอร์เซ็นต์ของ เวลา 20 วินาที

ตารางที่ 4.5 แสดงผลการคำนวณเปอร์เซ็นต์ให้แสดงผลเป็นบาร์กราฟเปรียบเทียบระหว่างหน้าจอ (Display), โปรแกรมซีเรียลมอนิเตอร์กับค่าที่ได้จากการคำนวณที่หน่วยเวลาเป็นนาที

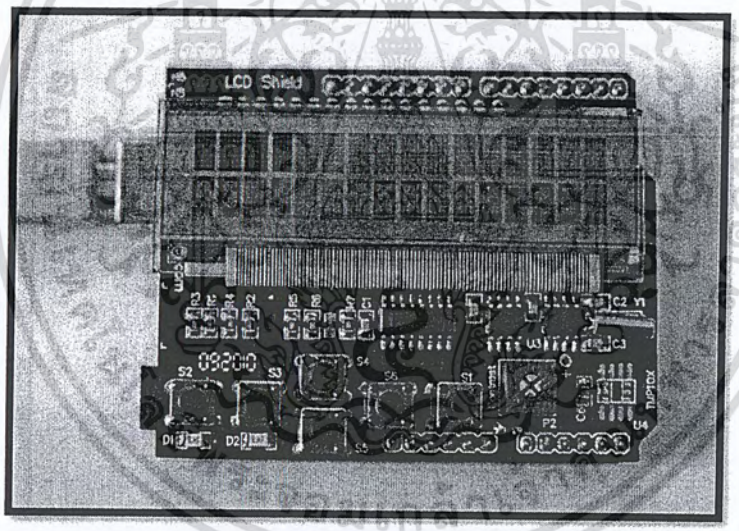
เวลา (นาที)	Bar graph			Error	
	Bar graph (คำนวณ)	Bar graph (Display)	Bar graph (โปรแกรม Serial Monitor)	Error (Display)	Error (โปรแกรม Serial Monitor)
0				0	0
12	****	****	****	0	0
24	*******	*******	*******	0	0
36	**********	**********	**********	0	0
48	**********	**********	**********	0	0
60	**********	**********	**********	0	0



ภาพที่ 4.12 แสดงผลการคำนวณเปอร์เซ็นต์ให้แสดงผลเป็นบาร์กราฟที่ 88 เปอร์เซ็นต์ของ เวลา 2 นาที

ตารางที่ 4.6 แสดงผลการคำนวณเปอร์เซ็นต์ให้แสดงผลเป็นบาร์กราฟเปรียบเทียบระหว่างหน้าจอ (Display), โปรแกรมซีเรียลมอนิเตอร์กับค่าที่ได้จากการคำนวณหน่วยเวลาเป็นชั่วโมง

เวลา (ชั่วโมง)	Bar graph			Error	
	Bar graph (คำนวณ)	Bar graph (Display)	Bar graph (โปรแกรม Serial Monitor)	Error (Display)	Error (โปรแกรม Serial Monitor)
0				0	0
1	****	****	****	0	0
2	*******	*******	*******	0	0
3	**********	**********	**********	0	0
4	**********	**********	**********	0	0
5	**********	**********	**********	0	0



ภาพที่ 4.13 แสดงผลการคำนวณเปอร์เซ็นต์ให้แสดงผลเป็นบาร์กราฟที่ 20 เปอร์เซ็นต์ของ เวลา 1 ชั่วโมง

4.4 สรุปผลการทดลอง

จากการทดลองเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ อาดูโน่ โดยให้แอลอีดี เป็นตัวแสดงผลโดยใช้ฟังก์ชั่นดีเลย์ จะทำให้แอลอีดี ติดดับตามเวลาที่กำหนดค่าการดีเลย์

จากการทดลองเขียนโปรแกรมแสดงผลผ่านหน้าจอ แอลซีดี โดยให้ Arduino Serial LCD Keypad Shield เป็นตัวแสดงผล โดยใช้ฟังก์ชั่น lcd.Print และกำหนดตำแหน่งการแสดงผลบนหน้าจอ แอลซีดี ผลจะปรากฏคำว่า "Hello World" เลข 1023 บนหน้าจอ แอลซีดี ตามตำแหน่งที่กำหนด

จากการทดลองเขียนโปรแกรมติดต่อโปรแกรมซีเรียลมอนิเตอร์ของไมโครคอนโทรเลอร์ อาดูโน่ โดยใช้ฟังก์ชั่น Serial.read เพื่อรับค่าที่คีย์และใช้ฟังก์ชั่น Serial.print เพื่อแสดงผลที่หน้าจอโปรแกรมซีเรียลมอนิเตอร์ ผลจะปรากฏเมื่อคีย์ค่าที่คีย์บอร์ดโดยที่ตัวแปรนั้นต้องเป็นตัวเลข ผลที่ได้จะถูกแปลงเป็นค่า เลขฐานสิบ, เลขฐานสิบหก, เลขฐานแปด, รหัสแอสกี้

จากการทดสอบโปรแกรมที่เขียนขึ้น เพื่อแสดงผลผ่านหน้าจอแอลซีดีโดยให้ Arduino Serial LCD Keypad Shield เป็นตัวแสดงผลมีผลตรงตามที่ได้ออกแบบไว้ และเมื่อโปรแกรมค่าเวลาให้ไมโครคอนโทรลเลอร์ อาดูโน่ ประมวลผลที่ไมโครคอลโทรเลอร์ อาดูโน่ โดยที่ประมวลผลออกมาตรงตามเวลาที่แท้จริง

บทที่ 5
บทสรุปและบทวิจารณ์

5.1 สรุปผลการปฏิบัติงาน

ปริญญานิพนธ์เรื่อง การประยุกต์บอร์ดไมโครคอนโทรลเลอร์กับตัวตั้งเวลาแบบโปรแกรมค่าได้ มีจุดประสงค์เพื่อศึกษาการทำงานของบอร์ดไมโครคอนโทรลเลอร์อาดูโน่ และเขียนโปรแกรมสั่งการบอร์ดโดยได้ออกแบบให้บอร์ดไมโครคอนโทรลเลอร์ อาดูโน่ ทำงานเป็นตัวตั้งเวลาแบบโปรแกรมค่าได้ จากผลการทดลองพบว่าโปรแกรมได้เป็นไปตามที่ได้ออกแบบไว้ ซึ่งการเขียนโปรแกรมให้หน้าจอแอลซีดี แสดงโหมดการทำงาน และสามารถเลือกโหมดการทำงานเพื่อคีย์ค่าข้อมูล (เวลา) ให้โปรแกรมประมวลผลเพื่อโชว์ค่าเวลาที่ทำการนับ และแสดงเป็นเปอร์เซ็นต์ในรูปแบบของบาร์กราฟได้ และสามารถนำผลการแสดงค่าไปแสดงที่โปรแกรมซีเรียลมอนิเตอร์เพื่อตรวจสอบข้อมูล

5.2 ปัญหาในการทำงานและแนวทางแก้ไข

5.2.1 การจับเวลาโดยใช้คำสั่งดีเลย์

การจับเวลาโดยใช้คำสั่งดีเลย์ เมื่อมีการนับเวลาที่นานจะทำให้เกิดการสะสมของดีเลย์ทำให้การนับเวลาเกิดการผิดพลาดไม่แน่นอน

การแก้ปัญหาโดยการเปลี่ยนจากการนับเวลาจากการใช้คำสั่งดีเลย์เป็นการใช้อินเตอร์รัปต์โดยการคำนวณค่าฐานเวลาในการนับเวลาและทำการแปลงเวลาเป็นหน่วยต่างๆส่งผลให้การนับเวลามีค่าที่แน่นอนกว่าการใช้การนับเวลาแบบใช้คำสั่งดีเลย์

5.2.2 การโปรแกรมค่าเวลาจากคีร์บอร์ด (Key Board)

เมื่อมีการคีย์ค่าเวลาจากคีย์บอร์ดจะมีสัญญาณรบกวนทำให้ค่าที่ได้จากการคีย์ค่าไม่แน่นอน

การแก้ปัญหาโดยการเพิ่มการดีเบาสซ์ (debounce) เข้าไปในโปรแกรมโดยโปรแกรมจะมีการตรวจสอบการกดปุ่มและ มีการดีเลย์ส่งผลให้ค่าที่ได้จากการคีย์ค่ามีค่าที่แน่นอนขึ้น

5.2.3 การสื่อสารข้อมูลกับคอมพิวเตอร์ (Computer)

เนื่องจากถ้าต้องการสื่อสารระหว่างไมโครคอนโทรลเลอร์กับคอมพิวเตอร์จะต้องมีการกำหนดค่าความเร็วในการสื่อสารของทั้งสองให้มีค่าตรงกัน

การแก้ปัญหาโดยการกำหนดค่าความเร็วในการสื่อสารให้กำหนดที่โปรแกรมที่เขียนขึ้น และกำหนดที่โปรแกรมซีเรียลมอนิเตอร์ให้มีค่าตรงกันผลจะทำให้การสื่อสารถูกต้อง

5.2.4 การคำนวณเปอร์เซ็นต์ในรูปแบบของบาร์กราฟ

เนื่องจากหน้าจอแอลซีดี มีขนาด 16x2 ตัวอักษร โดยมีขนาด 2 แถว แถวละ 16 ตัวอักษร เราต้องการให้ช่องสี่เหลี่ยมผืนผ้าแสดงสถานะเพียงแค่แถวบน 1 แถว และใช้ช่องสี่เหลี่ยมผืนผ้าแสดงสถานะจำนวน 15 ช่องในการแสดงสถานะ 100 เปอร์เซ็นต์

การแก้ปัญหาโดยทำการคำนวณ จะคำนวณที่ 15 ตัวอักษรเท่ากับ 100 เปอร์เซ็นต์ และถ้า 1 ตัวอักษรจะมีค่า 6.66 เปอร์เซ็นต์ จะทำให้เมื่อนับค่าเวลาที่นับได้เทียบเป็นเปอร์เซ็นต์ที่ทุก ๆ 6.66 เปอร์เซ็นต์จะมีการปริ้นรูปสี่เหลี่ยมผืนผ้า 1 ตัวอักษร

5.2.5 การปริ้นค่าสี่เหลี่ยมผืนผ้าเป็นสัญลักษณ์บาร์กราฟ

เนื่องจากหน้าจอแอลซีดีที่ผู้ทดลองมีลักษณะแตกต่างจากหน้าจอแอลซีดีชนิดอื่น ซึ่งทำให้รหัสแอสกี้ไม่ตรงกับตารางแอสกี้มาตรฐาน จึงเกิดปัญหาในการแสดงข้อมูลสถานะ

การแก้ปัญหาโดยเปลี่ยนไปใช้การกำหนดข้อมูลแบบไบต์ (BYTE) แทน และทดลองหาค่าจนได้ค่าสี่เหลี่ยมผืนผ้าออกมาที่ค่า 205

5.3 ผลที่ได้รับจากโครงงาน

จากการศึกษาทางทฤษฎีเพียงอย่างเดียวนั้นอาจไม่เพียงพอ เพราะเมื่อนำมาทดลองใช้งานจริงย่อมมีปัญหา และเงื่อนไขในสภาวะต่างๆมากมาย ซึ่งการที่จะประสบความสำเร็จนั้นบางครั้งต้องใช้ประสบการณ์ในการทำงานนั้นเป็นอย่างมากหรือบางครั้งอาจได้จากการทดลอง และนำมาวิเคราะห์ควบคู่กันไปกับแนวทางทฤษฎีที่ได้ศึกษา ซึ่งจากการทำโครงงานปริญญานิพนธ์นี้ทำให้ได้เรียนรู้สิ่งต่างๆมากมาย ซึ่งในบางส่วนอาจจะไม่ในตรงกับทฤษฎีเลยหรือบางส่วนของทฤษฎีก็เป็นความรู้ใหม่ที่ผู้ทำไม่เคยศึกษามาก่อนรวมถึงปัญหาต่างๆ ที่จะต้องมีการแก้ไขให้ถูกลุล่วงไปได้ทำให้สามารถไปใช้ในการทำงานจริง หลังจากจบการศึกษา

บรรณานุกรม

คะชา ชาญศิลป์, พิชโยทัย มหัทธนาภิวัฒน์และสมพันธุ์ ชาญศิลป์. 2548. *C Programming for Beginner ภาษาซีสำหรับผู้เริ่มต้น*. พิมพ์ครั้งที่ 2. กรุงเทพมหานคร : เพียร์สัน เอ็นดูเคชั่น อินโดไชน่า.

นคร ภักดีชาติ, ชัยวัฒน์ ลิ้มพรจิตรวิไลและวรพจน์ กรแก้ววัฒนกุล. 2550. *สนุกกับไมโครคอนโทรลเลอร์ด้วย Interactive C*. กรุงเทพมหานคร: อินโนเวตีฟ เอ็กเพอริเมนต์.

ประจิน พลังสันติกุล. *การเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์AVRด้วยภาษาซีกับ WinAVR(C Compiler)*. กรุงเทพมหานคร : แอพชอฟต์เทค.

วัชรินทร์ เดารพ. 2548. *เรียนรู้และเข้าใจ PSoC Microcontroller ด้วย ภาษา Assembly และภาษา C*. กรุงเทพมหานคร: อีทีที.

สมยศ จุณณะปิยะ. 2543. *การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ตระกูล MCS 51*. พิมพ์ครั้งที่ 3. กรุงเทพมหานคร: สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

เอกชัย มะการ. 2552. *เรียนรู้เข้าใจใช้งานไมโครคอนโทรลเลอร์ตระกูลAVRด้วยArduino*. กรุงเทพมหานคร : อีทีที.

โลกิจไทย. 2553. *แนะนำตัว Arduino*. [Online]. Available : http://www.logicthai.net/mode/14

วิกิพีเดีย. 2553. *ไมโครคอนโทลเลอร์*. [Online]. Available : http://th.wikipedia.org/wiki/ไมโครคอนโทลเลอร์

สคูลดอทเน็ท. 2553. *Microcontroller*. [Online]. Available : http://www.school.net/library/ web contest 2003/100team/ Microcontroller

Adisak Chinwong. 2552. *เรื่องต้น AVR Microcontroller*. [Online]. Available : http://www.techninan.ac.th /nan_ntc/adisak51/avr.html

Ayarafun. 2553. *เริ่มต้นเรียนรู้กับไมโครคอลโทลเลอร์*. [Online]. Available : http://www.logicthai.net /mode/14

Brian w. evans. 2009. *Arduino notebook a beginner's reference*. [Online]. Available : http://creativecommons.org/licenses/by-nc-sa/3.0

Engineer001. 2553. *ไมโครคอนโทลเลอร์คืออะไร*. [Online]. Available : http:// www.engineer001.com /index.php?mo=3&art=507518

Inex. 2553. *what is Microcontroller*. [Online]. Available : http://www.inex.co.th/microcontroller/ what is Microcontroller

Kawin (นามแฝง). 2553. *Arduino คืออะไร*. [Online]. Available :
http://www.wara.com/modules.php ?name=News&life=article&sid=711

Massimo Banzi. 2009. *Getting started with ARDUINO*. [Online]. Available :
http://www.arduino.cc

Mce lab. 2553. *ทำความรู้จักกับ Arduino และชุดทดลอง Arduino Project*. [Online].Available :
http://www .mce-lab.com/index.php?mo=3&art=416586

Pcman. 2009. *Start with Arduino*. [Online]. Available : http://www.logicthai.net/node/13

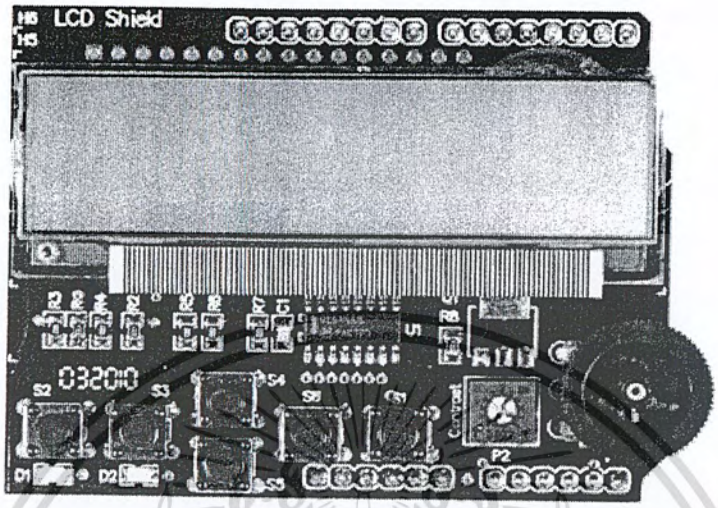
Thaimcu. 2553. *รูปแบบการพัฒนาภาษาโปรแกรมของ Microcontroller*. [Online]. Available:
http://www.themcu.com/article1.htm

ภาคผนวก

ภาคผนวก ก

1. **รายละเอียดของตัวตั้งเวลาแบบโปรแกรมค่าได้ (Specification of Programmable Timer)**
2. **โค้ดของตัวตั้งเวลาแบบโปรแกรมค่าได้ (Code of Programmable Timer)**

1. รายละเอียดของตัวตั้งเวลาแบบโปรแกรมค่าได้ (Specification of Programmable Timer)



Summary :

- Microcontroller ATmega168
- Operating Voltage 5V
- Input Voltage (recommended) 7-12V
- Input Voltage (limits) 6-20V

Features :

- 16x2 Character LCD witch out Black light
- Pushbuttons switch
- Two LED

Mode :

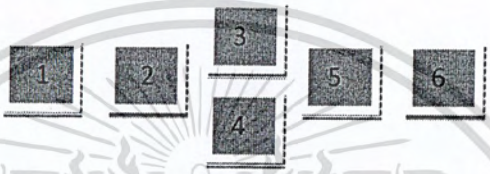
- Time setting
- Time select

Range :

Time units	Mode	
	Time setting	Time select
Sec	0 – 999	5, 10, 15, 30
Minute	0 – 99	60, 180, 300, 600
Hour	0 - 9	1, 2, 3

Detail :

Pushbuttons switch



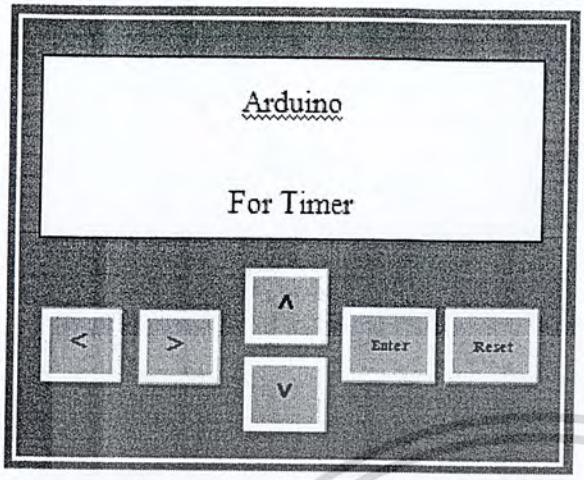
- ปุ่ม 1 , 2 ใช้เลื่อนตำแหน่งในการโปรแกรมค่าเวลาในโหมด Time setting
- ปุ่ม 2 ใช้ในการเลื่อนตำแหน่งเลือกค่าเวลาที่โปรแกรมในโหมด Time select
- ปุ่ม 3 , 4 ใช้ในการปรับเพิ่ม และลดค่าในการโปรแกรมค่าเวลาในโหมด Time setting
- ปุ่ม 5 ใช้ในการเลือกยืนยันการทำงาน
- ปุ่ม 6 ใช้ในการยกเลิกการทำงาน

LED

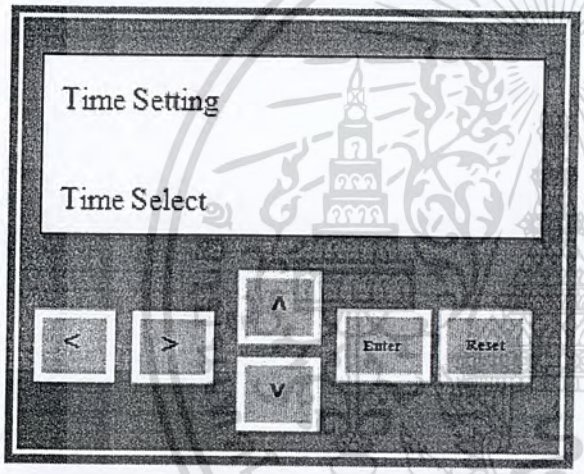
LED 1 LED 2

- LED 1, 2 ใช้แสดงค่าเมื่อการทำงานของ Timer เสร็จสิ้น
- LED 2 ใช้แสดงสถานะขณะที่ Timer ทำงาน

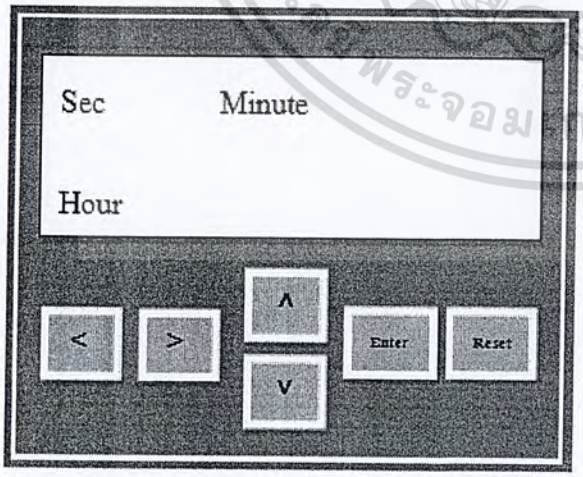
การขั้นตอนการเข้าโหมด



หน้าจอเริ่มต้น
แสดงผลเมื่อเปิดโปรแกรมซึ่งจะแสดงผลอยู่ประมาณ 5 วินาทีและจะเข้าสู่หน้าถัดไปโดยอัตโนมัติ



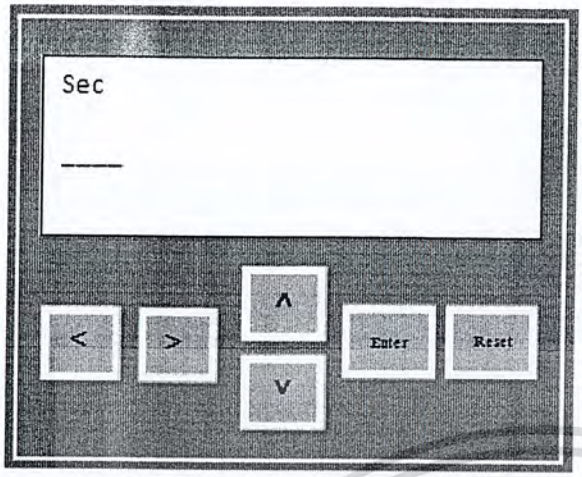
หน้าจอที่ 1 [ซึ่งจะเข้าสู่หน้าจอนี้โดยอัตโนมัติเมื่อเปิดโปรแกรม] แสดงรูปแบบของการตั้งค่าเวลาซึ่งจะมีให้เลือก 2 โหมด โดยการกดปุ่มขึ้น (Λ) หรือปุ่มลง (V) เพื่อเลือกโหมดที่ต้องการใช้ต่อไป ซึ่งมีดังนี้
- โหมดไทม์เซตติ้ง (Time Setting) เป็นการตั้งค่าเวลาโดยผู้ใช้กำหนดค่าเป้าหมายเอง
- โหมดไทม์ซีเร็กต์ (Time Select) เป็นการตั้งค่าเวลาโดยเลือกค่าเป้าหมายที่มีอยู่แล้ว



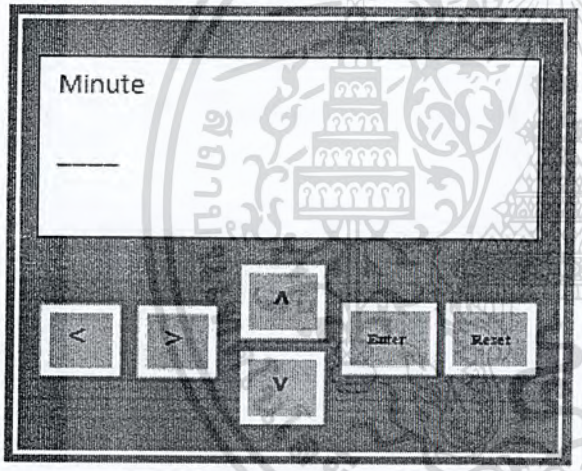
หน้าจอที่ 2 [ซึ่งจะเข้าสู่หน้านี้ได้โดยผ่านการเลือกจากหน้าจอที่ 1 ก่อน] ไม่ว่าจะเลือกโหมดไทม์เซตติ้ง (time setting) หรือโหมดไทม์ซีเร็กต์ (Time Select) ก็จะต้องเข้าสู่หน้าจอที่ 2 เพื่อเลือกหน่วยของเวลา โดยการกดปุ่มซ้าย (<) หรือปุ่มขวา (>) ซึ่งจะมีหน่วยเวลาให้เลือกดังนี้

- SEC	มีหน่วยเวลาเป็นวินาที
- MINUTE	มีหน่วยเวลาเป็นนาที
- HOUR	มีหน่วยเวลาเป็นชั่วโมง

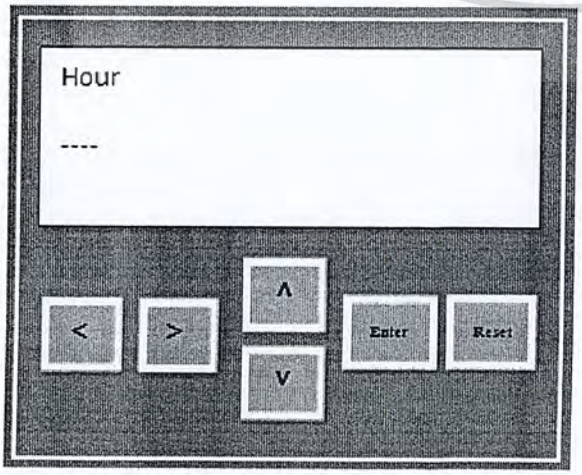
การตั้งค่าแบบทามเซตติ่ง (Time setting)



หน้าจอที่ 3.1 [ซึ่งจะเข้าสู่หน้าจอนี้โดยผ่านการเลือกจากหน้าจอที่ 2 ก่อน] ในกรณีนี้ต้องเลือกที่โหมดไทม์เซตติ่ง (time setting) แล้วเลือกโหมดวินาที (Sec) จากนั้นแล้วทำการป้อนค่าเป้าหมายที่ต้องการโดยการกดปุ่มซ้าย (<) หรือปุ่มขวา (>) เพื่อเลือกหลักและกดปุ่มขึ้น (Λ) หรือปุ่มลง (V) เพื่อเลือกค่าเวลาที่ต้องการซึ่งมีย่านจะการใช้งานคือ 0 วินาทีถึง 999 วินาทีเมื่อเลือกค่าที่ต้องการแล้วทำการกดปุ่มเอ็นเตอร์ (Enter)

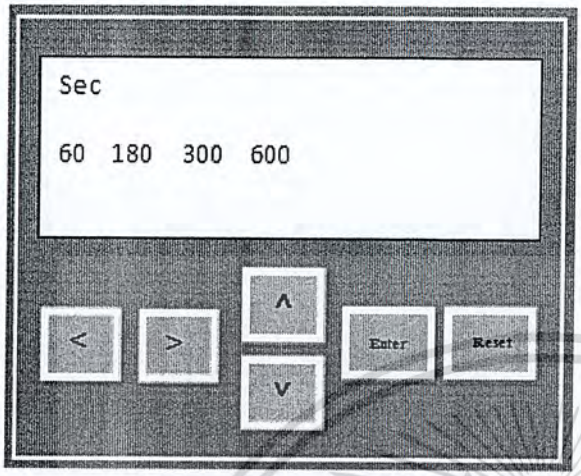


หน้าจอที่ 3.2 [ซึ่งจะเข้าสู่หน้าจอนี้โดยผ่านการเลือกจากหน้าจอที่ 2 ก่อน] ในกรณีนี้ต้องเลือกที่โหมดไทม์เซตติ่ง (Time Setting) แล้วเลือกโหมดนาที (Minute) จากนั้นแล้วทำการป้อนค่าเป้าหมายที่ต้องการ โดยการกดปุ่มซ้าย (<) หรือปุ่มขวา (>) เพื่อเลือกหลักและกดปุ่มขึ้น (Λ) หรือปุ่มลง (V) เพื่อเลือกค่าเวลาที่ต้องการซึ่งมีย่านจะการใช้งานคือ 0 นาทีถึง 99 นาทีเมื่อเลือกค่าที่ต้องการแล้วทำการกดปุ่มเอ็นเตอร์ (Enter)

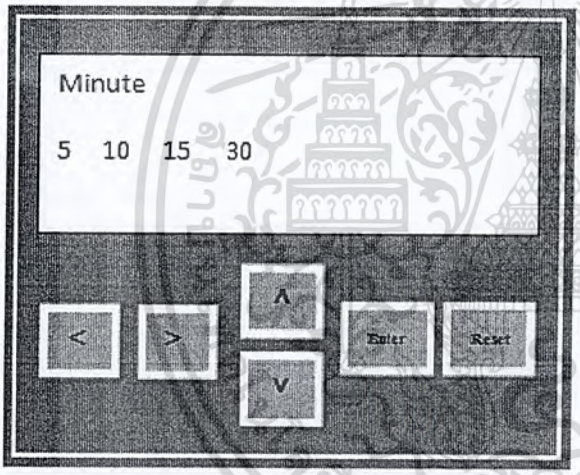


หน้าจอที่ 3.3 [ซึ่งจะเข้าสู่หน้าจอนี้โดยผ่านการเลือกจากหน้าจอที่ 2 ก่อน] ในกรณีนี้ต้องเลือกที่โหมดไทม์เซตติ่ง (Time Setting) แล้วเลือกโหมดชั่วโมง (Hour) จากนั้นแล้วทำการป้อนค่าเป้าหมายที่ต้องการ โดยการกดปุ่มซ้าย (<) หรือปุ่มขวา (>) เพื่อเลือกหลักและกดปุ่มขึ้น (Λ) หรือปุ่มลง (V) เพื่อเลือกค่าเวลาที่ต้องการซึ่งมีย่านจะการใช้งานคือ 0 ชั่วโมงถึง 9 ชั่วโมงเมื่อเลือกค่าที่ต้องการแล้วทำการกดปุ่มเอ็นเตอร์ (Enter)

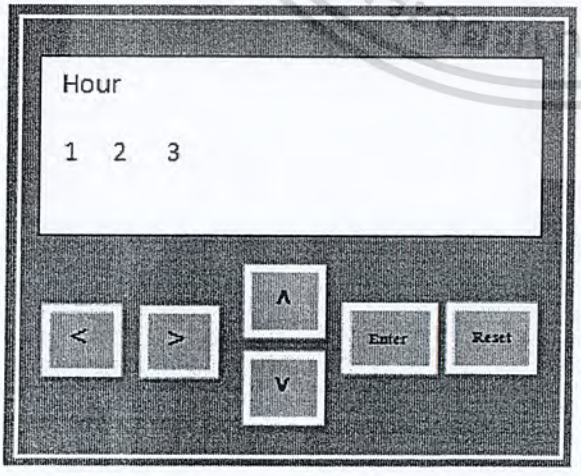
การตั้งค่าแบบทามซีเร็กต์ (Time Select)



หน้าจอที่ 4.1 [ซึ่งจะเข้าสู่หน้าจอนี้โดยผ่านการเลือกจากหน้าจอที่ 2 ก่อน] ในกรณีนี้ต้องเลือกที่โหมดไทม์ซีเร็กต์ (Time Select) แล้วเลือกโหมดวินาที (Sec) จากนั้นแล้วทำการป้อนค่าเป้าหมายที่ต้องการ โดยการกดปุ่มซ้าย (<) หรือปุ่มขวา (>) เพื่อเลือกค่าเวลาที่ต้องการซึ่งมีย่านจะการใช้งานคือ 60 180 300 600 วินาที ซึ่งเป็นค่าที่มีการกำหนดไว้แล้ว เมื่อเลือกค่าที่ต้องการแล้วทำการกดปุ่มเอ็นเตอร์ (Enter)

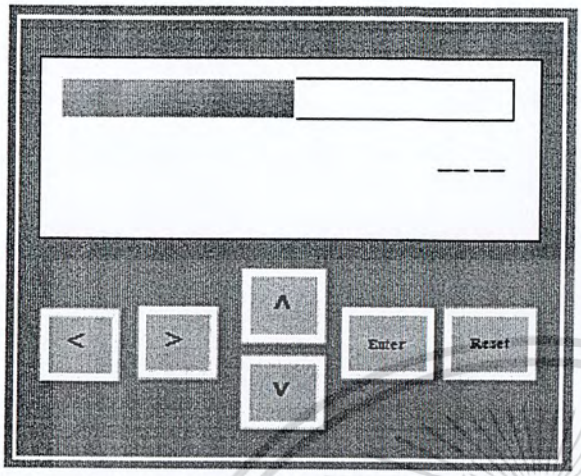


หน้าจอที่ 4.2 [ซึ่งจะเข้าสู่หน้าจอนี้โดยผ่านการเลือกจากหน้าจอที่ 2 ก่อน] ในกรณีนี้ต้องเลือกที่โหมดไท้ซีเล็ก (Time Select) แล้วเลือกโหมดนาที (Minute) จากนั้นแล้วทำการป้อนค่าเป้าหมายที่ต้องการ โดยการกดปุ่มซ้าย (<) หรือปุ่มขวา (>) เพื่อเลือกค่าเวลาที่ต้องการซึ่งมีย่านจะการใช้งานคือ 5 10 15 30 นาที ซึ่งเป็นค่าที่มีการกำหนดไว้แล้ว เมื่อเลือกค่าที่ต้องการแล้วทำการกดปุ่มเอ็นสเตอร์ (Enter)



หน้าจอที่ 4.3 [ซึ่งจะเข้าสู่หน้าจอนี้โดยผ่านการเลือกจากหน้าจอที่ 2 ก่อน] ในกรณีนี้ต้องเลือกที่โหมดไทม์ซีเร็กต์ (Time Select) แล้วเลือกโหมดชั่วโมง(Hour) จากนั้นแล้วทำการป้อนค่าเป้าหมายที่ต้องการ โดยการกดปุ่มซ้าย (<) หรือปุ่มขวา (>) เพื่อเลือกค่าเวลาที่ต้องการซึ่งมีย่านจะการใช้งานคือ 1 2 3 ชั่วโมง ซึ่งเป็นค่าที่มีการกำหนดไว้แล้ว เมื่อเลือกค่าที่ต้องการแล้วทำการกดปุ่มเอ็นเตอร์ (Enter)

การแสดงผลของการนับเวลา



หน้าจอที่ 5 [ซึ่งจะเข้าสู่หน้าจอนี้โดยผ่านการกดปุ่มเอ็นเตอร์ (Enter) จากหน้าจอที่ 3 หรือหน้าจอที่ 4 ก่อน] ในกรณีนี้เมื่อตั้งค่าเป้าหมายหรือเวลาที่ต้องการแล้วจึงกดปุ่มเอ็นเตอร์ (Enter) เพื่อรัน (Run) โปรแกรมซึ่งจะเป็นการนับเวลาตามที่ตั้งค่าเอาไว้ การแสดงผลจะแบ่งออกเป็น 3 ส่วนด้วยกันคือ

- แสดงผลการนับเวลาเป็นเปอร์เซ็นต์ (%)
- แสดงผลการนับเวลาในรูปแบบของบาร์กราฟ ซึ่งเกิดจากการแปลงการนับเวลาในรูปแบบเปอร์เซ็นต์
- แสดงเป็นตัวเลข คือ เป็นเวลาที่นับได้

2. โค้ดของตัวตั้งเวลาแบบโปรแกรมค่าได้ (Code of Programmable Timer)

```
#include <avr/interrupt.h>
#include <avr/io.h>
#include "LCDHC595.h"
LCDHC595 lcd = LCDHC595(4, 3, 2);//DATA, STROB, CLOCK
#define _SPLASH_SCREEN_DELAY 3000
#define INIT_TIMER_COUNT 6
#define RESET_TIMER2 TCNT2 = INIT_TIMER_COUNT

// Digital
#define LED0   5
#define LED1   6 // ปกติดับ ถ้าถึงเวลาที่ตั้งแล้ว จะติดสว่าง

// LCD Line Control
#define _LCD_LINE0_ 0x80
#define _LCD_LINE1_ 0xc0

// Analog
#define KEY_LEFT   0 // S2 Left1
#define KEY_RIGHT  1 // S3 Left2
#define KEY_UP     2 // S4 Top
#define KEY_DOWN   3 // S5 bottom
#define KEY_ENTER  5 // S6 Right

// ตารางเก็บค่าเวลาที่ต้องการในแต่ละเมนู
int timeSelectSecMenuTable[] = {60,180,300,600};
int timeSelectMinMenuTable[] = {300, 600, 900, 1800}; // จำนวนวินาทีใน 5,10,15,30 นาที
int timeSelectHourMenuTable[] = {3600,7200,10800};   // จำนวนวินาทีใน 1, 2 และ 3 ชม.
int hh=0,mm=0,ss=0;
```

```
// ตัวแปรที่ใช้ใน interrupt
int int_counter = 0;
volatile int m_second = 0;
volatile int m_oldSecond = 0;

// ส่วนการตั้งค่า Timer
void setupTimer()
{
 TCCR2A |= (1<<CS22);
 TCCR2A &= ~((1<<CS21) | (1<<CS20));
 // Use normal mode
 TCCR2A &= ~((1<<WGM21) | (1<<WGM20));
 // Use internal clock - external clock not used in Arduino
 ASSR |= (0<<AS2);
 //Timer2 Overflow Interrupt Enable
 TIMSK2 |= (1<<TOIE2) | (0<<OCIE2A);
 RESET_TIMER2;
 sei();
}

// ส่วน Interrupt timer
// Arduino รันที่ความเร็ว 16 Mhz, ดังนั้นเราต้องการ 1000 ms ต่อ 1 วินาที
// สูตรคำนวน 1/ ((16000000 / 64) / 256) = 1 / 1000
ISR(TIMER2_OVF_vect) {
 RESET_TIMER2;
 int_counter += 1;
 if (int_counter == 1000) {
  m_second += 1;
  int_counter = 0;
 }
};
/*****************************************************************************/
```

```
// ตรวจสอบการกดปุ่ม
byte checkKeyPress(byte key){
  if (analogRead(key) < 10) {
    delay(20); // debounce
    if(analogRead(key) < 10) { // still press
      return 1;
    }
  }
  return 0;
}

// debounce การปล่อยมือจากปุ่ม
void debounceUnpressed(byte key){
  while(analogRead(key) < 10);
  delay(20);
}

// ชุดคำสั่งช่วยให้แสดงผล LCD ในตำแหน่งที่ต้องการ
void lcdPrint(char txt[],byte row, byte col){
  if (row == 0){
    lcd.SetCursor(_LCD_LINE0_ + col);
  }else{
    lcd.SetCursor(_LCD_LINE1_ + col);
  }
  lcd.Print(txt);
}
/*************************************************************************/
```

```
// แสดงหน้าจอต้อนรับ และค้างไว้ 5 วินาที
void firstPage(){
 lcdPrint("Arduino",0,5);
 lcdPrint("For Timer",1,4);
 delay(_SPLASH_SCREEN_DELAY);
}
/*****************************************************************************/

// ชุดคำสั่งลบเครื่องหมายลูกศร ของเมนูการเลือกรูปแบบการตั้งเวลา
void clearPointer_timeMenu(){
 // ลบลูกศร
 lcdPrint(" ",0,0);
 lcdPrint(" ",1,0);
}

// หน้าจอแสดงเมนู Time Setting , Time Select
byte timeMenu(){
 lcd.ClearScreen();

// แสดงหัวข้อ
 lcdPrint("Time Setting",0,2);
 lcdPrint("Time Select",1,2);

// แสดงลูกศร
 lcdPrint(">",0,0);
 byte currentMenu = 0; // 0 = อันแรก = Time Setting, 1 = อันที่สอง = Time Select
 byte enter = false;
 while(!enter){
  if( checkKeyPress(KEY_UP) || checkKeyPress(KEY_DOWN) ){
   if (currentMenu == 0){
    currentMenu = 1;
   }else{
```

```
        currentMenu = 0;
      }

  // แสดง pointer ณ ตำแหน่ง menu ปัจจุบัน
      clearPointer_timeMenu();
      lcdPrint(">",currentMenu,0);
      debounceUnpressed(KEY_UP);
      debounceUnpressed(KEY_DOWN);
    }
    //
    if ( checkKeyPress(KEY_ENTER) ){
      enter = true;
      debounceUnpressed(KEY_ENTER);
    }
  }
  return currentMenu;
}
/*************************************************************************/

// ชุดคำสั่งลบเครื่องหมายลูกศร ณ เมนูการเลือกรูปแบบการตั้งเวลา
void clearPointer_timeTypeMenu(){
 // ลบลูกศร
 clearPointer_timeMenu();
 lcdPrint(" ",0,7);
}

// หน้าจอแสดงเมนูเพื่อเลือกรูปแบบเวลา
byte timeTypeMenu(){
 lcd.ClearScreen();
```

```
// แสดงหัวข้อ
 lcdPrint("Sec",0,2);
 lcdPrint("Minute",0,9);
 lcdPrint("Hour",1,2);

// แสดงลูกศร
 lcdPrint(">",0,0);
 byte keypressed = 0; // ใช้ตรวจว่ามีการกดปุ่มจริงหรือไม่
 byte currentMenu = 0; // 0 = อันแรก = Sec, 1 = อันที่สอง = Minutes, 2 = อันที่สาม = hour
 byte enter = false;
 while(!enter){
  if( checkKeyPress(KEY_LEFT) ){
   if (currentMenu == 0){
    currentMenu = 2;
   }else{
    currentMenu--;
   }
   keypressed = true;
   debounceUnpressed(KEY_LEFT);
  }

  if ( checkKeyPress(KEY_RIGHT) ){
   if (currentMenu == 2){
    currentMenu = 0;
   }else{
    currentMenu++;
   }
   keypressed = true;
   debounceUnpressed(KEY_RIGHT);
  }
  if (keypressed){
```

```
// ลบและแสดง pointer ณ ตำแหน่ง menu ปัจจุบัน
      clearPointer_timeTypeMenu();
      switch(currentMenu){
        case 0: lcdPrint(">",0,0); break;
        case 1: lcdPrint(">",0,7); break;
        case 2: lcdPrint(">",1,0); break;
      }
      keypressed = false;
    }

// กด ENTER เพื่อเลือก
    if ( checkKeyPress(KEY_ENTER) ){
      enter = true;
      debounceUnpressed(KEY_ENTER);
    }
  }

  return currentMenu;
}
/*****************************************************************************/

// ชุดคำสั่งลบเครื่องหมายลูกศร ณ เมนูการเลือกรูปแบบการตั้งเวลา
void clearPointer_timeSelectSecMenu(){

// ลบลูกศร
  lcdPrint(" ",0,4);
  lcdPrint(" ",1,4);
  lcdPrint(" ",0,10);
  lcdPrint(" ",1,10);
}
```

```
// หน้าจอแสดงเมนูเพื่อเลือกเวลาในแบบวินาที
int timeSelectSecMenu(){
 lcd.ClearScreen();

// แสดงหัวข้อ (pointer 4,10)
   //     0123456789ABCDE
 lcdPrint("Sec > 60    180",0,0);
 lcdPrint("      300   600",1,0);

// แสดงลูกศรแรก
 //lcdPrint(">",0,4);
 byte keypressed = 0; // ใช้ตรวจว่ามีการกดปุ่มจริงหรือไม่
 byte currentMenu = 0; // 0 = อันแรก = 60, 1 = อันที่สอง = 180, 2 = อันที่สาม = 300, 3 = อันที่สี่
= 600
 byte enter = false;
 while(!enter){
  if( checkKeyPress(KEY_LEFT) ){
   if (currentMenu == 0){
    currentMenu = 3;
   }else{
    currentMenu--;
   }
   keypressed = true;
   debounceUnpressed(KEY_LEFT);
  }
  if ( checkKeyPress(KEY_RIGHT) ){
   if (currentMenu == 3){
    currentMenu = 0;
   }else{
    currentMenu++;
   }
   keypressed = true;
```

```
    debounceUnpressed(KEY_RIGHT);
   }

 // ลบและแสดง pointer ณ ตำแหน่ง menu ปัจจุบัน update เมื่อกดปุ่มเท่านั้น
   if (keypressed){
    clearPointer_timeSelectSecMenu();
    switch(currentMenu){
     case 0: lcdPrint(">",0,4); break;
     case 1: lcdPrint(">",0,10); break;
     case 2: lcdPrint(">",1,4); break;
     case 3: lcdPrint(">",1,10); break;
    }
    keypressed = false;
   }

 // กด ENTER เพื่อเลือก
   if ( checkKeyPress(KEY_ENTER) ){
    enter = true;
    debounceUnpressed(KEY_ENTER);
   }
  }

  return timeSelectSecMenuTable[currentMenu];
}
/*****************************************************************************/

// ชุดคำสั่งลบเครื่องหมายลูกศร ณ เมนูการเลือกรูปแบบการตั้งเวลา นาที
void clearPointer_timeSelectMinMenu(){

// ลบลูกศร
 lcdPrint(" ",0,4);
 lcdPrint(" ",1,4);
```

```
  lcdPrint(" ",0,10);
  lcdPrint(" ",1,10);
}

// หน้าจอแสดงเมนูเพื่อเลือกเวลาในแบบนาที
int timeSelectMinMenu(){
 lcd.ClearScreen();

// แสดงหัวข้อ (pointer 4,10)
   //     0123456789ABCDE
 lcdPrint("Min > 5     10",0,0);
 lcdPrint("      15    30",1,0);

// แสดงลูกศรแรก
 //lcdPrint(">",0,4);
 byte keypressed = 0; // ใช้ตรวจว่ามีการกดปุ่มจริงหรือไม่
 byte currentMenu = 0; // 0 = อันแรก = 5, 1 = อันที่สอง = 10, 2 = อันที่สาม = 15, 3 = อันที่สี่ = 30
 byte enter = false;
 while(!enter){
  if( checkKeyPress(KEY_LEFT) ){
   if (currentMenu == 0){
    currentMenu = 3;
   }else{
    currentMenu--;
   }
   keypressed = true;
   debounceUnpressed(KEY_LEFT);
  }
  if ( checkKeyPress(KEY_RIGHT) ){
   if (currentMenu == 3){
    currentMenu = 0;
   }else{
```

```
        currentMenu++;
      }
      keypressed = true;
      debounceUnpressed(KEY_RIGHT);
    }

  // ลบและแสดง pointer ณ ตำแหน่ง menu ปัจจุบัน update เมื่อกดปุ่มเท่านั้น
    if (keypressed){
     clearPointer_timeSelectMinMenu();
     switch(currentMenu){
       case 0: lcdPrint(">",0,4); break;
       case 1: lcdPrint(">",0,10); break;
       case 2: lcdPrint(">",1,4); break;
       case 3: lcdPrint(">",1,10); break;
     }
     keypressed = false;
    }

  // กด ENTER เพื่อเลือก
    if ( checkKeyPress(KEY_ENTER) ){
     enter = true;
     debounceUnpressed(KEY_ENTER);
    }
  }

  return timeSelectMinMenuTable[currentMenu];
}
/*****************************************************************************/
// ชุดคำสั่งลบเครื่องหมายลูกศร ณ เมนูการเลือกรูปแบบการตั้งเวลา ชั่วโมง
void clearPointer_timeSelectHourMenu(){
// ลบลูกศร
 lcdPrint(" ",0,5);
```

```
  lcdPrint(" ",0,10);
  lcdPrint(" ",1,5);

}
// หน้าจอแสดงเมนูเพื่อเลือกเวลาในแบบชั่วโมง
int timeSelectHourMenu(){
 lcd.ClearScreen();
 // แสดงหัวข้อ (pointer 5,10)
   //     0123456789ABCDE
 lcdPrint("Hour > 1    2",0,0);
 lcdPrint("       3",1,0);

// แสดงลูกศรแรก
 //lcdPrint(">",0,4);
 byte keypressed = 0; // ใช้ตรวจว่ามีการกดปุ่มจริงหรือไม่
 byte currentMenu = 0; // 0 = อันแรก = 5, 1 = อันที่สอง = 10, 2 = อันที่สาม = 15, 3 = อันที่สี่ = 30
 byte enter = false;
 while(!enter){
  if( checkKeyPress(KEY_LEFT) ){
    if (currentMenu == 0){
      currentMenu = 2;
    }else{
      currentMenu--;
    }
    keypressed = true;
    debounceUnpressed(KEY_LEFT);
  }
  if ( checkKeyPress(KEY_RIGHT) ){
    if (currentMenu == 2){
      currentMenu = 0;
    }else{
      currentMenu++;
```

```
      }
      keypressed = true;
      debounceUnpressed(KEY_RIGHT);
    }

  // ลบและแสดง pointer ณ ตำแหน่ง menu ปัจจุบัน update เมื่อกดปุ่มเท่านั้น
    if (keypressed){
      clearPointer_timeSelectHourMenu();
      switch(currentMenu){
        case 0: lcdPrint(">",0,5); break;
        case 1: lcdPrint(">",0,10); break;
        case 2: lcdPrint(">",1,5); break;
      }
      keypressed = false;
    }

// กด ENTER เลือก
    if ( checkKeyPress(KEY_ENTER) ){
      enter = true;
      debounceUnpressed(KEY_ENTER);
    }
  }

  return timeSelectHourMenuTable[currentMenu];
}
/*****************************************************************************/
```

```
// เมนูตั้งเวลาแบบกำหนดเอง คืนค่าเป็นจำนวนวินาที
int timeSettingSecMenu(){
 lcd.ClearScreen();
 lcd.Command(lcd_cursor_on);
 lcd.Command(lcd_cursor_blink);

// แสดงหัวข้อ
  //     0123456789ABCDE
 lcdPrint("Sec",0,0);
 lcdPrint("___",1,0);
 lcd.SetCursor(0 + _LCD_LINE1_); // เพื่อ reset cursor ให้อยู่อันแรก
 char number[4]= {'0','0','0','\0'}; // เก็บค่าที่ user ต้องการ
 byte currentCursor = 0; // 0 - 2
 byte keypress = false; // มีการกดปุ่มหรือไม่
 byte enter = false;
 while(!enter){
  if( checkKeyPress(KEY_LEFT) ){
   if (currentCursor == 0){
    currentCursor = 2;
   }else{
    currentCursor--;
   }
   keypress = true;
   debounceUnpressed(KEY_LEFT);
  }
  if ( checkKeyPress(KEY_RIGHT) ){
   if (currentCursor == 2){
    currentCursor = 0;
   }else{
    currentCursor++;
   }
   keypress = true;
```

```
    debounceUnpressed(KEY_RIGHT);
   }
   if ( checkKeyPress(KEY_UP) ){
    if (number[currentCursor] == '9') {
     number[currentCursor] = '0';
    }else{
     number[currentCursor]++;
    }
    keypress = true;
    debounceUnpressed(KEY_UP);
   }
   if ( checkKeyPress(KEY_DOWN) ){
    if (number[currentCursor] == '0') {
     number[currentCursor] = '9';
    }else{
     number[currentCursor]--;
    }
    keypress = true;
    debounceUnpressed(KEY_DOWN);
   }

// ถ้ามีการกดปุ่มให้ update
   if (keypress){
// แสดงผล
    lcd.SetCursor(0 + _LCD_LINE1_ ); // เพื่อ reset cursor ให้อยู่อันแรก
    lcd.Print(number);   // แสดงค่า
// แสดงตำแหน่ง cursor
    lcd.SetCursor(currentCursor + _LCD_LINE1_ ); // เพื่อ reset cursor ให้อยู่อันแรก
    keypress = false;
   }

//ต้องใช้ key enter เพื่อระบุค่าที่ต้องการ
```

```
    if ( checkKeyPress(KEY_ENTER) ){
      enter = true;
      debounceUnpressed(KEY_ENTER);
    }
   }
   lcd.Command(lcd_cursor_off);
   return atoi(number);
}
/*****************************************************************************/
// เมนูตั้งเวลาแบบกำหนดเอง คืนค่าเป็นจำนวนวินาที
int timeSettingMinMenu(){
 lcd.ClearScreen();
 lcd.Command(lcd_cursor_on);
 lcd.Command(lcd_cursor_blink);

// แสดงหัวข้อ
  //     0123456789ABCDE
 lcdPrint("Minute",0,0);
 lcdPrint("__",1,0);
 lcd.SetCursor(0 + _LCD_LINE1_ ); // เพื่อ reset cursor ให้อยู่อันแรก
 char number[3]= {'0','0','\0'}; // เก็บค่าที่ user ต้องการ
 byte currentCursor = 0; // 0 - 1
 byte keypress = false; // มีการกดปุ่มหรือไม่
 byte enter = false;
 while(!enter){
  if( checkKeyPress(KEY_LEFT) || checkKeyPress(KEY_RIGHT) ){
    if (currentCursor == 0){
      currentCursor = 1;
    }else if ( currentCursor == 1 ){
      currentCursor = 0;
    }
    keypress = true;
```

```
    debounceUnpressed(KEY_LEFT);
    debounceUnpressed(KEY_RIGHT);
   }

  if ( checkKeyPress(KEY_UP) ){
   if (number[currentCursor] == '9') {
    number[currentCursor] = '0';
   }else{
    number[currentCursor]++;
   }
   keypress = true;
   debounceUnpressed(KEY_UP);
  }
  if ( checkKeyPress(KEY_DOWN) ){
   if (number[currentCursor] == '0') {
    number[currentCursor] = '9';
   }else{
    number[currentCursor]--;
   }
   keypress = true;
   debounceUnpressed(KEY_DOWN);
  }

// ถ้ามีการกดปุ่มให้ update
  if (keypress){
// แสดงผล
   lcd.SetCursor(0 + _LCD_LINE1_ ); // เพื่อ reset cursor ให้อยู่อันแรก
   lcd.Print(number);   // แสดงค่า
// แสดงตำแหน่ง cursor
   lcd.SetCursor(currentCursor + _LCD_LINE1_ ); // เพื่อ reset cursor ให้อยู่อันแรก
   keypress = false;
  }
```

```
 //ต้องใช้ key enter  เพื่อระบุค่าที่ต้องการ
   if ( checkKeyPress(KEY_ENTER) ){
     enter = true;
     debounceUnpressed(KEY_ENTER);
   }
  }
  lcd.Command(lcd_cursor_off);
  return atoi(number)*60;
}
/*************************************************************************
********/
// เมนูตั้งเวลาแบบกำหนดเอง คืนค่าเป็นจำนวนวินาที
int timeSettingHourMenu(){
 lcd.ClearScreen();
 lcd.Command(lcd_cursor_on);
 lcd.Command(lcd_cursor_blink);

// แสดงหัวข้อ
  //     0123456789ABCDE
 lcdPrint("Hour",0,0);
 lcdPrint("_",1,0);
 lcd.SetCursor(0 + _LCD_LINE1_ ); // เพื่อ reset cursor ให้อยู่อันแรก
 char number[2]= {'0','\0'}; // เก็บค่าที่ user ต้องการ
 byte currentCursor = 0; // 0 - 1
 byte keypress = false; // มีการกดปุ่มหรือไม่
 byte enter = false;

 while(!enter)
  if ( checkKeyPress(KEY_UP) ){
   if (number[currentCursor] == '9') {
    number[currentCursor] = '0';
```

```
    }else{
      number[currentCursor]++;
    }
    keypress = true;
    debounceUnpressed(KEY_UP);
  }
  if ( checkKeyPress(KEY_DOWN) ){
    if (number[currentCursor] == '0') {
      number[currentCursor] = '9';
    }else{
      number[currentCursor]--;
    }
    keypress = true;
    debounceUnpressed(KEY_DOWN);
  }

// ถ้ามีการกดปุ่มให้ update
  if (keypress){
// แสดงผล
    lcd.SetCursor(0 + _LCD_LINE1_ ); // เพื่อ reset cursor ให้อยู่อันแรก
    lcd.Print(number);    // แสดงค่า
// แสดงตำแหน่ง cursor
    lcd.SetCursor(currentCursor + _LCD_LINE1_ ); // เพื่อ reset cursor ให้อยู่อันแรก
    keypress = false;
  }

//ต้องใช้ key enter เพื่อระบุค่าที่ต้องการ
  if ( checkKeyPress(KEY_ENTER) ){
    enter = true;
    debounceUnpressed(KEY_ENTER);
  }
}
```

```
  lcd.Command(lcd_cursor_off);
  return atoi(number)*3600;
}
/*****************************************************************************/
// คำนวนเวลาคงเหลือและแสดงผล
void timeProcessAndDisplay(int delaySecond){
  byte complete = false;
  unsigned long countDelay = 0;
  int percent = 0;
  byte barCounter = 0;
  lcd.ClearScreen();
// แสดงผล %
  lcd.SetCursor(_LCD_LINE1_ + 0);
  lcd.Print("0 %");
   m_oldSecond = m_second;
  while(!complete) {
   while(m_oldSecond == m_second); // แทนการ delay(960)
   m_oldSecond = m_second;
   countDelay++;
// คำนวน %
   percent = (int)(countDelay * 100 / delaySecond);

// แสดงผล Bar ใช้วิธีเทียบค่า % กับตารางที่กำหนด เพื่อให้แสดงผลได้ตรง
   lcd.SetCursor(_LCD_LINE0_ + 0);
   for(int x=0;x<(percent/6.66);x++)
   {
    Serial.print("*");
    lcd.Print(205,BYTE)
   }
```

```
// แสดงผล %
  lcd.SetCursor(_LCD_LINE1_ + 0);
  lcd.Print( percent );
  Serial.print( percent );
  lcd.Print(" %");
  Serial.print(" %");
  Serial.println();

// ตรวจสอบว่าครบ 60 วินาที,นาที,ชั่วโมงแล้วหรือยัง
  if(++ss==60){
   ss=0;
   if(++mm==60){
    mm=0;
    if(++hh==100)
     hh=0;
   }
  }

// แสดงผล
  lcd.SetCursor(_LCD_LINE1_ + 7);
  Serial.write(0x0D);
  lcd.Print(hh/10);
  Serial.print(hh/10);
  lcd.Print(hh%10);
  Serial.print(hh%10);
  lcd.Print(":");
  Serial.print(":");
  lcd.Print(mm/10);
  Serial.print(mm/10);
  lcd.Print(mm%10);
  Serial.print(mm%10);
  lcd.Print(":");
```

```
    Serial.print(":");
    lcd.Print(ss/10);
    Serial.print(ss/10);
    lcd.Print(ss%10);
    Serial.print(ss%10);

// กระพริบ
    if(ss%2==1)
    digitalWrite(LED1, HIGH);
    else
    digitalWrite(LED1, LOW);

// ถ้าครบเวลาที่ตั้งแล้วให้ดับไฟ
    if (countDelay >= delaySecond){
     complete = true;
     hh=mm=ss=0;
    digitalWrite(LED0, LOW);
    digitalWrite(LED1, LOW);
   }

 }

// ครบเวลาแล้วแสดงไฟติด
 digitalWrite(LED1, LOW);
 // lcd.SetCursor(_LCD_LINE0_ + 0);
 //lcd.Print( "****************" );
 byte exit = false;
 while(!exit){
  if ( checkKeyPress(KEY_ENTER) ){
    exit = true;
    debounceUnpressed(KEY_ENTER);
  }
```

```
  }
 }
 /*****************************************************************************/
 void setup() {
  setupTimer();
  Serial.begin(19200);
  pinMode(LED0, OUTPUT);
  pinMode(LED1, OUTPUT);
  lcd.Initial();
 }
 void loop()
 {
  while(1){
   digitalWrite(LED0, HIGH); // สั่งดับ
   digitalWrite(LED1, HIGH);

// แสดงหน้าจอแรก (หน้าจอต้อนรับ)
   lcd.ClearScreen();
   firstPage();

// เข้าสู่หน้าจอเมนู Time setting , Time Select
   byte currentMenu = timeMenu();  // เลือกว่าจะเป็นแบบ Time Setting หรือ Time Select
   byte timeType = timeTypeMenu(); // เลือกแบบเวลาว่าเป็น Sec,Min หรือ Hour
   int totalSeconds = 0; // เวลาที่จะตั้ง(เป็นวินาที)
   if (currentMenu == 0){

// ถ้าเลือกแบบ Time Setting
    switch(timeType){
     case 0:
          totalSeconds = timeSettingSecMenu();
          break;
     case 1:
```

```
            totalSeconds = timeSettingMinMenu();
            break;
        case 2:
            totalSeconds = timeSettingHourMenu();
            break;
      }
    } else if (currentMenu == 1){
// ถ้าเลือกแบบ Time Select
     switch(timeType){
       case 0:
            totalSeconds = timeSelectSecMenu();
            break;
       case 1:
            totalSeconds = timeSelectMinMenu();
            break;
       case 2:
            totalSeconds = timeSelectHourMenu();
            break;
     }
   }

// คำนวนเวลาคงเหลือและแสดงผล
   timeProcessAndDisplay(totalSeconds);
 }
}
```

ภาคผนวก ข

1. โครงสร้างบอร์ด Arduino
2. โครงสร้างบอร์ด Arduino Serial LCD Keypad Shield
3. ATmega 168 Datasheet
4. FTDI 232 Datasheet
5. HC595 Shift Register Datasheet
6. โปรแกรมอาดูโน่ (Arduino 0021)
7. Library of Arduino Serial LCD Keypad Shield