การศึกษาเครื่องควบคุมแบบพีไอดีโดยใช้ STM32

A Study of PID Controller by STM32

ภควัต สิทธิสร
ภาณุพงศ์ เรืองเกษม
ศรายุทธ แซ่หลี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมการวัดคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2556

การศึกษาเครื่องควบคุมแบบพีไอดีโดยใช้ STM32
A Study of PID Controller by STM32

ภควัต สิทธิสร

ภาณุพงศ์ เรืองเกษม

ศรายุทธ แซ่หลี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมการวัดคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.2556

A Study of PID Controller by STM32

Pakkawat Sittisorn
Panupong Ruangkasam
Sarayoot Saelee

THIS THESIS IS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING
FACULTY OF ENGINEERING
KING MONKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2013

ภาควิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

.....

หัวข้อปริญญานิพนธ์	การศึกษาเครื่องควบคุมแบบพีไอดีโดยใช้ STM32		
	A Study of PID Controller by STM32		
นักศึกษาผู้จัดทำ	นายภควัต สิทธิสร	**รหัสนักศึกษา**	53011194
	นายภาณุพงศ์ เรืองเกษม	**รหัสนักศึกษา**	53011236
	นายศรายุทธ แซ่หลี	**รหัสนักศึกษา**	53011536
ปริญญา	วิศวกรรมศาสตร์บัณฑิต		
สาขาวิชา	วิศวกรรมการวัดคุม		
ปีการศึกษา	2556		

อาจารย์ผู้ควบคุมปริญญานิพนธ์	ลายมือชื่อ
รองศาสตราจารย์วิริยะ กองรัตน์	

ปริญญานิพนธ์เรื่อง	การศึกษาเครื่องควบคุมแบบพีไอดีโดยใช้ STM32	
โดย	นายภควัต สิทธิสร	รหัสนักศึกษา 53011194
	นายภาณุพงศ์ เรืองเกษม	รหัสนักศึกษา 53011236
	นายศรายุทธ แซ่หลี	รหัสนักศึกษา 53011536
ปริญญา	วิศวกรรมศาสตร์บัณฑิต	
สาขาวิชา	วิศวกรรมการวัดคุม	
ปีการศึกษา	2556	
อาจารย์ที่ปรึกษา	รองศาสตราจารย์วิริยะ กองรัตน์	

บทคัดย่อ

โครงงานนี้นำเสนอการศึกษาเครื่องควบคุมแบบพีไอดีโดยใช้ STM32 F0 series 32-bit ARM® Cortex™ microcontroller เป็นตัวรับค่าอินพุทจาก Key pad และค่า Feedback จาก process จากนั้นคอนโทรลเลอร์ภายในจะทำการประมวลผลโดยใช้วิธีการคำนวณค่า PID เพื่อส่งสัญญาณไปควบคุม Process ให้เข้าหาค่า SV พร้อมทำการแสดงผลข้อมูลของต่างๆของกระบวนการออกทางจอ LCD

Thesis Title	A Study of PID Controller By STM32
Authors	Mr. Pakkawat Sittisorn
	Mr. Panupong Ruangkasam
	Mr. Sarayoot Saelee
Degree	Bachelor of Engineering
Program	Instrumentation Engineering
Year	2013
Thesis Advisor	Assoc.Prof.Viriya Kongrat

Abstract

This project presents a study of PID controller by STM32 F0 series 32-bit ARM® Cortex™. We use the microcontroller for receiving input from a key pad and feedback value from our process. The internal controller will evaluate with PID method for sending the control signal to the process and showing all of the process's data on LCD monitor.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดีเพราะได้รับความอนุเคราะห์จาก รองศาสตราจารย์ วิริยะ กองรัตน์ ที่ได้ให้คำปรึกษาแนะนำรวมถึงอุปกรณ์แก่ผู้วิจัยตลอดมา ผู้วิจัยรู้สึกซาบซึ้งและขอกราบขอบพระคุณเป็นอย่างสูง คณาจารย์ภาควิชาวิศวกรรมการวัดคุมทุกท่านที่ได้มอบคำแนะนำหลายอย่างๆที่เป็นประโยชน์ตลอดจนถึงกำลังใจในการทำวิจัยตลอดมา เพื่อนๆ พี่ๆ ทุกท่านที่คอยช่วยให้กำลังใจในการทำวิจัยและความช่วยเหลือในด้านต่างๆ ทำให้ผู้จัดทำสามารถก้าวผ่านปัญหาต่างๆในระหว่างที่ทำงานวิจัยชิ้นนี้ได้สำเร็จ

ขอกราบขอบพระคุณ บิดา มารดา อันเป็นที่รักยิ่ง ที่สนับสนุนและเป็นแรงบันดาลใจในการทำงานวิจัยชิ้นนี้ คุณค่า ประโยชน์และคำชมเชยอันได้จากปริญญานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

คณะผู้จัดทำ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

รูป หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปริญญานิพนธ์

ในขบวนการผลิตโดยเฉพาะการผลิตแบบต่อเนื่อง และการผลิตที่มีขนาดใหญ่ เช่น กระบวนการกลั่นน้ำมัน กระบวนการผลิตไฟฟ้า กระบวนการถลุงเหล็ก นั้นล้วนแล้วแต่เป็นขบวนการที่มีความซับซ้อนค่อนข้างมาก การควบคุมกระบวนการผลิตเพื่อให้ได้ผลิตผลที่ดีมีคุณภาพตามต้องการ (desired productivity) รวมทั้งต้องประหยัดด้วย เหล่านี้นับเป็นสิ่งสำคัญ อันเป็นที่ต้องการสำหรับเจ้าของกิจการ การนำเทคโนโลยีการวัดคุมและระบบควบคุม(instrumentation and control system) เข้ามาใช้งานจึงเป็นสิ่งสำคัญที่สุด เพื่อให้บรรลุตามวัตถุประสงค์ของกระบวนการการผลิตที่ดีตามต้องการ ในปัจจุบัน เทคโนโลยีระบบควบคุม ได้รับการพัฒนาอย่างรวดเร็ว อุปกรณ์แบบใหม่ ๆ ได้ถูกนำมาใช้ในงาน เพื่อเพิ่มประสิทธิภาพและประสิทธิผลในการผลิต เครื่องควบคุม PID โดยใช้ STM 32 เหมาะสมและสะดวกต่อการใช้งาน โดยสามารถควบคุมได้จากคอมพิวเตอร์ในซอฟต์แวร์ที่เรียกว่า Keil พร้อมทั้งศึกษา Microcontroller ที่เรียกว่า STM 32 ซึ่งเป็นอุปกรณ์ใช้สำหรับควบคุมการทำงานของอุปกรณ์ที่ต่อพ่วง ประมวลผลค่าที่ได้จากอินพุตและ แสดงผลลัพธ์ที่ได้ไปยัง Display

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1. ศึกษาการใช้งานตัวควบคุมแบบ PID
2. ศึกษาวิธีการปรับจูน PID แบบต่างๆ
3. ศึกษาการใช้งานโปรแกรม Keil เพื่อควบคุม STM32 ให้ทำงานตามต้องการ
4. ศึกษาการเขียนโปรแกรมด้วยภาษาซี
5. ศึกษาการแปลงสัญญาณ Analog ให้เป็น Digital และ Digital ให้เป็น Analog

1.3 ขอบเขตของปริญญานิพนธ์

1. ศึกษาการควบคุมกระบวนการแบบ PID Control และทำความเข้าใจตัวควบคุม PID
2. การปรับจูน PID แบบต่างๆ การใช้โปรแกรม Keil เพื่อควบคุม STM32 ให้ทำงานได้ตามต้องการ รวมไปถึงการทำงานของ Transmitter ในการใช้สัญญาณไฟฟ้าเพื่อไปควบคุมอุปกรณ์ตามที่ต้องการ

1.4 ขั้นตอนการศึกษา

การออกแบบระบบต้นแบบ

1. ศึกษาภาพรวมของโปรเจคพร้อมวางแผนการทำงาน
2. ศึกษาคุณลักษณะและวิธีการใช้งานของอุปกรณ์แต่ละชิ้น
3. ศึกษาวิธีการเชื่อมต่ออุปกรณ์แต่ละตัวให้สามารถทำงานร่วมกันได้ผ่าน I2C Bus

การศึกษาการเขียนโปรแกรม

1. ศึกษาการเขียนโปรแกรมด้วยภาษาซี
2. ทดสอบเขียนโปรแกรมด้วย Keil พร้อมกับทดลองโดยการเขียนโปรแกรมทำให้ไฟกระพริบ
3. ทดสอบเขียนโปรแกรมเพื่อให้ LCD และ 7-Segment แสดงผล
4. ทดสอบเขียนโปรแกรมให้คำนวณค่า PID

1.5 ประโยชน์ที่คาดว่าจะได้รับ

เนื่องจากทางคณะผู้ดำเนินโครงการกำลังศึกษาในหลักสูตรวิศวกรรมการวัดคุม ดังนั้นจึงอยากศึกษาเกี่ยวกับหลักสูตรที่ได้ศึกษามา ซึ่งผู้ดำเนินโครงการสนใจในการศึกษาเครื่องควบคุม PID โดยใช้ STM 32 เพื่อมีความรู้ความเข้าใจเรื่องเครื่องควบคุมแบบ PID และ การเขียนโปรแกรมของ Micro Controller ที่เรียกว่า STM 32 มาควบคุมการทำงานของอุปกรณ์เอาท์พุตและทำการประมวลผลค่าอินพุต จากนั้นแสดงผลไปยัง Display โดยใช้หลักการควบคุมแบบ PID

บทที่ 2
ทฤษฎีและหลักการที่เกี่ยวข้อง

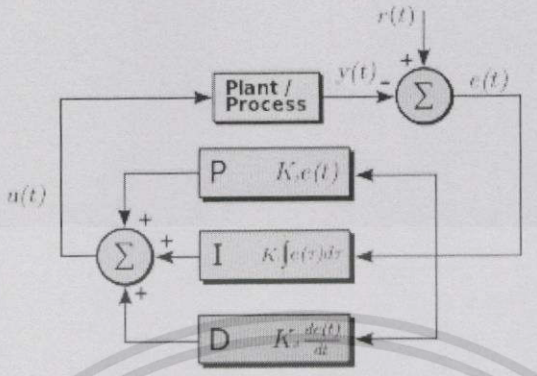
2.1 ตัวควบคุม PID (PID Controller)

ระบบควบคุมแบบสัดส่วน-ปริพันธ์-อนุพันธ์ (PID Controller) เป็นกลไกการควบคุมแบบป้อนกลับที่ใช้งานอย่างแพร่หลายในอุตสาหกรรม โดยตัวควบคุม PID จะคำนวณค่าความแตกต่างของของความผิดพลาดระหว่างค่าตัวแปรของกระบวนการ (Process variable) และค่าเป้าหมาย (Setpoint) ที่เราออกแบบไว้แล้วพยายามทำให้ค่าความผิดพลาดนั้นเหลือน้อยที่สุดโดยการปรับแต่งค่าสัญญาณขาออก (Output) ของกระบวนการ

ระบบควบคุมแบบ PID นี้ประกอบไปด้วยค่าตัวแปรคงที่ 3 ตัว คือ P (Proportional), I (Integral), D (Derivative) โดย P จะขึ้นอยู่กับค่าความผิดพลาดในปัจจุบัน ส่วน I จะมาจากผลรวมของค่าความผิดพลาดที่ผ่านมา และ D มาจากการทำนายค่าความผิดพลาดในอนาคต โดยค่าตัวแปรทั้งหมดนี้จะอิงจากอัตราความเปลี่ยนแปลงในปัจจุบัน ผลรวมของทั้ง 3 ตัวแปรนี้จะถูกใช้เพื่อนำมาปรับแต่งกระบวนการผ่านทางอุปกรณ์ควบคุมต่างๆ เช่น ตำแหน่งของวาล์วควบคุม

ระบบควบคุมแบบ PID ได้ถูกพิจารณาว่าเป็นระบบควบคุมที่ดีที่สุด โดยการปรับตั้งค่าตัวแปรทั้งสามในระบบควบคุม ตัวควบคุมก็สามารถที่จะออกแบบการควบคุมที่เหมาะสมกับระบบของเราได้ การตอบสนองของระบบก็จะอยู่ในลักษณะของการตอบสนองของตัวควบคุมต่อค่าความผิดพลาด เช่น การเกิดโอเวอร์ชูต (Overshoot) หรือการแกว่ง (Oscillation) ในระหว่างเข้าหาค่าเป้าหมาย

ในการนำไปประยุกต์ใช้งานบางครั้งอาจใช้เพียงหนึ่งหรือสองตัวแปรขึ้นอยู่กับความเหมาะสมของกระบวนการ โดยระบบควบคุมแบบ PID จะถูกเรียกว่าระบบควบคุมแบบ PI, PD, P, หรือ I ขึ้นอยู่กับว่าใช้งานแบบใด ในการใช้งานทั่วไปเราจะใช้การควบคุมแบบ PI เนื่องจากค่าอนุพันธ์เกิดการแปรปรวนกับสัญญาณรบกวนจากการวัดได้ง่าย ส่วนในกรณีที่ไม่ใส่ค่าปริพันธ์จะช่วยป้องกันระบบจากการพุ่งเกินค่าเป้าหมายได้



รูปที่ 2.1 แสดงบล็อกไดอะแกรมของระบบควบคุม PID แบบป้อนกลับ

2.1.1 พื้นฐานของลูปควบคุม (Control loop basics)

ตัวอย่างที่คุ้นเคยสำหรับลูปควบคุมที่เราจะยกตัวอย่างก็คือการปรับแต่งท่อน้ำร้อนและน้ำเย็นเพื่อผสมน้ำในภาชนะเพื่อให้มีอุณหภูมิตามที่เราออกแบบ เมื่อมีคนสัมผัสน้ำในภาชนะที่บรรจุเพื่อวัดอุณหภูมิของน้ำก็จะใช้ความรู้สึกตัดสินว่าอุณหภูมิเหมาะสมหรือยังถ้ายังก็จะทำให้เกิดการป้อนกลับ (feedback) กลับมา ทำให้ต้องปรับวาล์วน้ำร้อนหรือน้ำเย็นจนกว่าอุณหภูมิของน้ำจะเป็นไปตามที่เราต้องการ อุณหภูมิของน้ำที่สัมผัสจะเป็นค่า Process variable หรือ Process value (PV) อุณหภูมิที่เราต้องการจะเป็นค่า Setpoint (SP) สัญญาณขาเข้า (ตำแหน่งของท่อน้ำ) และสัญญาณขาออกของ PID จะถูกเรียกว่า Manipulated variable (MV) หรือ Control Variable (CV) ความแตกต่างระหว่างอุณหภูมิที่วัดได้กับค่าเป้าหมายก็คือค่า Error (e) แล้วประเมินว่าน้ำในภาชนะร้อนหรือเย็นมากน้อยแค่ไหน หลังจากวัดค่าอุณหภูมิ (PV) พร้อมทั้งคำนวณค่าความผิดพลาดเรียบร้อย ตัวควบคุมก็จะตัดสินใจว่าจะเปิดก๊อกเพิ่มหรือลดแค่ไหน (MV) เนื่องจากเราสามารถปรับแต่งอย่างไรไหนก็ได้ตั้งแต่น้ำเย็นจนถึงน้ำร้อน ส่วนนี้จะเป็นตัวอย่างในส่วนของการควบคุม Proportional ถ้าในกรณีที่น้ำในภาชนะทำความร้อนได้ช้า ตัวควบคุมก็จะพยายามเพิ่มความเร็วของกระบวนการด้วยการเปิดก๊อกน้ำร้อนให้กว้างขึ้นชั่วขณะหนึ่ง ในส่วนนี้จะเป็นตัวอย่างของ Derivative ถ้าอุณหภูมิของภาชนะต่ำเกินไปทั้งๆที่มีอัตราการไหลของน้ำอุ่นที่ดี ตัวควบคุมก็จะเปิดก๊อกน้ำร้อนมากขึ้นไปเรื่อยๆ ในส่วนนี้จะเป็นตัวอย่างของการควบคุม Integral

2.1.2 ทฤษฎีการควบคุม PID

การควบคุม PID เป็นชื่อที่ถูกเรียกจากการรวมเทอมของตัวแปรทั้งสาม โดยตัวแปรทั้งสามนี้จะกลายเป็นค่า Manipulated variable (MV) เทอมของ Proportional, Integral, และ Derivative จะถูกคำนวณเป็นค่า Output ของตัวควบคุม PID กำหนดให้ u(t) เป็นค่า Output ของตัวควบคุม ดังนั้นสมการของ PID จะได้

$$u(t) = MV(t) = K_p e(t) + K_i \int_0^t e(T)dT + K_d \frac{d}{dt} e(t) \quad (2.1)$$

โดย

K_p: อัตราการขยายสัดส่วน, ตัวแปรปรับค่าได้

K_i: อัตราขยายปริพันธ์, ตัวแปรปรับค่าได้

K_d: อัตราขยายอนุพันธ์, ตัวแปรปรับค่าได้

e : ค่าความผิดพลาด $= SP - PV$

t : เวลา หรือ เวลาชั่วขณะ(ปัจจุบัน)

T : ตัวแปรปริพันธ์

2.1.3 สัดส่วน (Proportional term)

เทอมของสัดส่วน (บางครั้งเรียก อัตราขยาย) จะเปลี่ยนแปลงเป็นสัดส่วนของค่าความผิดพลาด การตอบสนองของสัดส่วนสามารถทำได้โดยการคูณค่าความผิดพลาดด้วยค่าคงที่ K_p, หรือที่เรียกว่าอัตราขยายสัดส่วน

เทอมของสัดส่วนจะเป็นไปตามสมการ:

$$P_{out} = K_p e(t) \tag{2.2}$$

เมื่อ

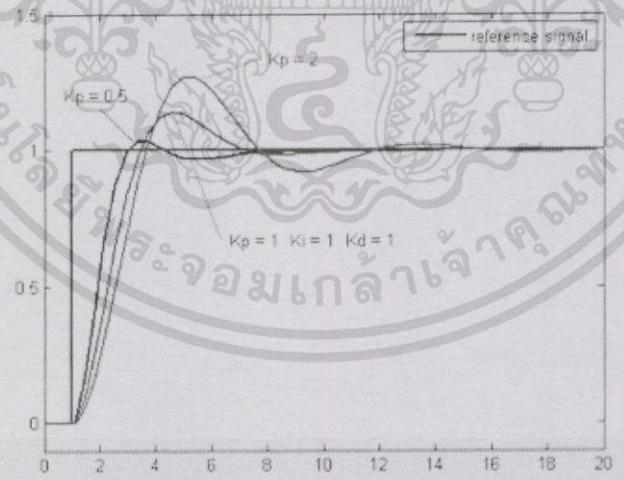
P_{out} : สัญญาณขาออกของเทอมสัดส่วน

K_p : อัตราขยายสัดส่วน, ตัวแปรปรับค่าได้

e : ค่าความผิดพลาด $= SP - PV$

t : เวลา

ผลอัตราขยายสัดส่วนที่สูงค่าความผิดพลาดก็จะเปลี่ยนแปลงมากเช่นกัน แต่ถ้าสูงเกินไประบบจะไม่เสถียร ได้ ในทางตรงกันข้าม ผลอัตราขยายสัดส่วนที่ต่ำ ระบบควบคุมจะมีผลตอบสนองต่อกระบวนการน้อยตามไปด้วย



รูปที่ 2.2 แสดงกราฟ PV ต่อเวลา, K_p กำหนดเป็น 3 ค่า(K_iและ K_dคงที่)

2.1.4 ปริพันธ์ (Integral term)

ผลจากเทอมปริพันธ์ (บางครั้งเรียก *reset*) เป็นสัดส่วนของขนาดความผิดพลาดและระยะเวลาของความผิดพลาด ผลรวมของความผิดพลาดในทุกช่วงเวลา (ปริพันธ์ของความผิดพลาด) จะให้ออฟเซตสะสมที่ควรจะเป็นในก่อนหน้า ความผิดพลาดสะสมจะถูกคูณโดยอัตราขยายปริพันธ์ ขนาดของผลของเทอมปริพันธ์จะกำหนดโดยอัตราขยายปริพันธ์, K_i

เทอมปริพันธ์จะเป็นไปตามสมการ:

$$I_{out} = K_i \int_0^t e(\tau)d\tau \tag{2.3}$$

เมื่อ

I_{out}: สัญญาณขาออกของเทอมปริพันธ์

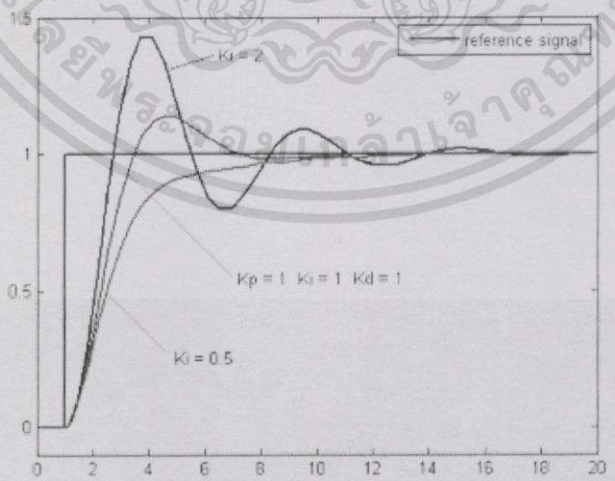
K_i: อัตราขยายปริพันธ์, ตัวแปรปรับค่าได้

e: ความผิดพลาด $= SP - PV$

t : เวลา

τ: ตัวแปรปริพันธ์หุ่น

เทอมปริพันธ์ (เมื่อรวมกับเทอมสัดส่วน) จะเร่งกระบวนการให้เข้าสู่จุดที่ต้องการและขจัดความผิดพลาดที่เหลืออยู่ที่เกิดจากการใช้เพียงเทอมสัดส่วน แต่อย่างไรก็ตาม เทอมปริพันธ์เป็นการตอบสนองต่อความผิดพลาดสะสมในอดีต จึงสามารถทำให้เกิดโอเวอร์ชูตได้ (ข้ามจุดที่ต้องการและเกิดการหันเหไปทางทิศทางอื่น)



รูปที่ 2.3 แสดงกราฟ PV ต่อเวลา, K_i กำหนดเป็นสามค่า (K_p และ K_dคงที่)

2.1.5 อนุพันธ์ (Derivative term)

อัตราการเปลี่ยนแปลงของความผิดพลาดจากกระบวนการนั้นคำนวณหาจากความชันของความผิดพลาดทุกๆเวลา (นั่นคือ เป็นอนุพันธ์อันดับหนึ่งสัมพันธ์กับเวลา) และคูณด้วยอัตราขยายอนุพันธ์ K_d ขนาดของผลของเทอมอนุพันธ์ (บางครั้งเรียก *อัตรา*) ขึ้นกับ อัตราขยายอนุพันธ์ K_d

เทอมอนุพันธ์เป็นไปตามสมการ:

$$D_{out} = K_d \frac{d}{dt} e(t) \tag{2.4}$$

เมื่อ

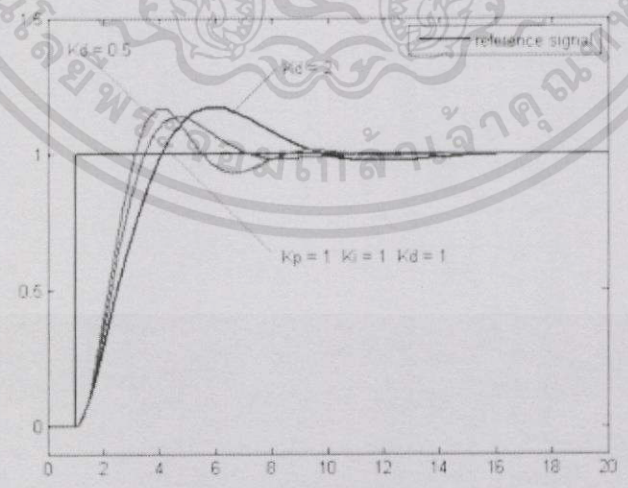
D_{out}: สัญญาณขาออกของเทอมอนุพันธ์

K_d: อัตราขยายอนุพันธ์, ตัวแปรปรับค่าได้

e : ความผิดพลาด $= SP - PV$

t : เวลา

เทอมอนุพันธ์จะชะลออัตราการเปลี่ยนแปลงของสัญญาณขาออกของระบบควบคุมและด้วยผลนี้จะช่วยให้ระบบควบคุมเข้าสู่จุดที่ต้องการ ดังนั้นเทอมอนุพันธ์จะใช้ในการลดขนาดของโอเวอร์ชูตที่เกิดจาเทอมปริพันธ์และทำให้เสถียรภาพของการรวมกันของระบบควบคุมดีขึ้น แต่อย่างไรก็ตามอนุพันธ์ของสัญญาณรบกวนที่ถูกขยายในระบบควบคุมจะไวมากต่อการรบกวนในเทอมของความผิดพลาดและสามารถทำให้กระบวนการไม่เสถียรได้ถ้าสัญญาณรบกวนและอัตราขยายอนุพันธ์มีขนาดใหญ่เพียงพอ



รูปที่ 2.4 แสดงกราฟ PV ต่อเวลา, สำหรับ K_d 3 ค่า (K_p และ K_i คงที่)

2.1.6 การปรับจูน

2.1.6.1 การปรับจูนด้วยมือ

ถ้าระบบยังคงทำงาน ขั้นแรกให้ตั้งค่า K_i และ K_d เป็นศูนย์ เพิ่มค่า K_p จนกระทั่งสัญญาณขาออกเกิดการแกว่ง (oscillate) แล้วตั้งค่า K_p ให้เหลือครึ่งหนึ่งของค่าที่ทำให้เกิดการแกว่งสำหรับการตอบสนองชนิด "quarter amplitude decay" แล้วเพิ่ม K_i จนกระทั่งออฟเซตถูกต้องในเวลาที่พอเพียงของกระบวนการ แต่ถ้า K_i มากไปจะทำให้ไม่เสถียร สุดท้ายถ้าต้องการ ให้เพิ่มค่า K_d จนกระทั่งลูปอยู่ในระดับที่ยอมรับได้ แต่ถ้า K_d มากเกินไปจะเป็นเหตุให้การตอบสนองและโอเวอร์ชูตเกินยอมรับได้ ปกติการปรับจูน PID ถ้าเกิดโอเวอร์ชูตเล็กน้อยจะช่วยให้เข้าสู่จุดที่ต้องการเร็วขึ้น แต่ในบางระบบไม่สามารถยอมให้เกิดโอเวอร์ชูตได้ และถ้าค่า K_p น้อยเกินไปก็จะทำให้เกิดการแกว่ง

ตารางที่ 2.1 ความสัมพันธ์ของค่า K_p, K_i, K_d

ตัวแปร	ช่วงเวลาขาขึ้น (Rise time)	โอเวอร์ชูต (Overshoot)	เวลาสู่สมดุล (Settling time)	ความผิดพลาดสถานะคงตัว (Steady-state-error)	เสถียรภาพ
K_p	ลด	เพิ่ม	เปลี่ยนแปลงเล็กน้อย	ลด	ลด
K_i	ลด	เพิ่ม	เพิ่ม	ลดลงอย่างมีนัยสำคัญ	ลด
K_d	ลดลงเล็กน้อย	ลดลงเล็กน้อย	ลดลงเล็กน้อย	ตามทฤษฎีไม่มีผล	ดีขึ้นถ้า K_d น้อยๆ

2.1.6.2 วิธีการ Ziegler–Nichols

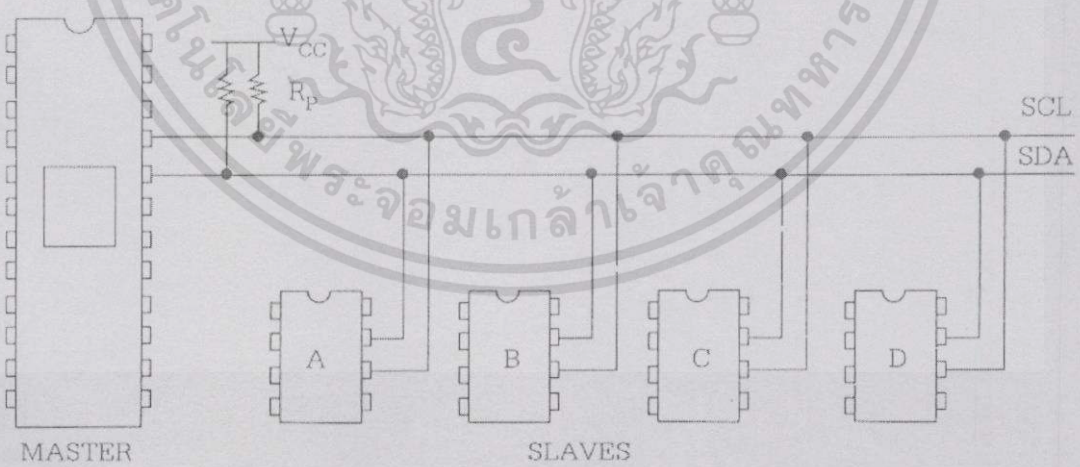
วิธีการนี้นำเสนอโดย John G. Ziegler และ Nathaniel B. Nichols ในคริสต์ทศวรรษที่ 1940 ขั้นแรกให้ตั้งค่า K_i และ K_d เป็นศูนย์ เพิ่มอัตราขยาย *P* สูงที่สุด, K_u, จนกระทั่งเริ่มเกิดการแกว่ง นำค่า K_u และค่าช่วงการแกว่ง P_u มาหาค่าตัวแปรที่เหลือดังตาราง:

ตารางที่ 2.2 วิธีคำนวณ Ziegler–Nichols

Control type	K_p	K_i	K_d
P	$0.5K_u$	-	-
PI	$0.45K_u$	$1.2K_p/P_u$	-
PLD	$0.60K_u$	$2K_p/P_u$	$K_pP_u/8$

2.2 I2C Bus

I2C Bus ย่อมาจาก Inter Integrate Circuit Bus (IIC) นิยมเรียกสั้นๆว่า **I2C BUS** (ไอ-แสควี-บัส) เป็นการสื่อสารอนุกรม แบบซิงโครนัส (Synchronous) เพื่อใช้ ติดต่อสื่อสาร ระหว่างไมโครคอนโทรลเลอร์ (MCU) กับอุปกรณ์ภายนอก ซึ่งถูกพัฒนาขึ้นโดยบริษัท Philips Semiconductors โดยใช้สายสัญญาณเพียง 2 เส้นเท่านั้น คือ serial data (SDA) และสาย serial clock (SCL) ซึ่งสามารถเชื่อมต่ออุปกรณ์ จำนวนหลายๆ ตัว เข้าด้วยกันได้ ทำให้ MCU ใช้พอร์ตเพียง 2 พอร์ตเท่านั้น
การสื่อสารอนุกรม แบบซิงโครนัส (Synchronous) คือ



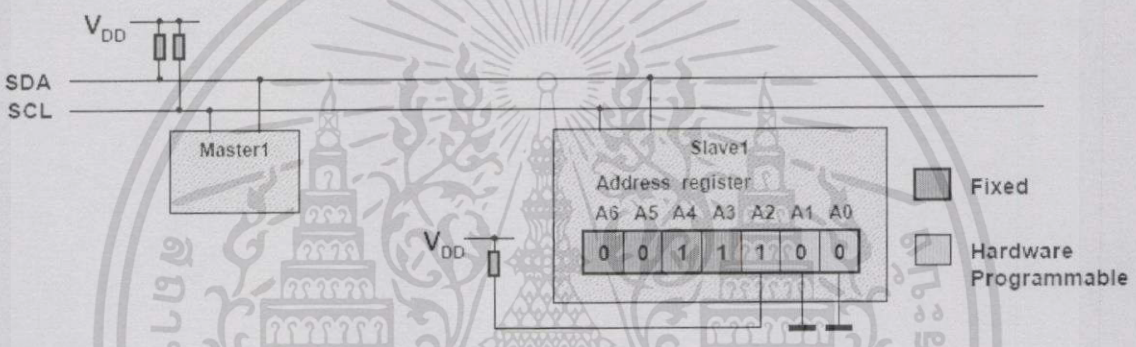
รูปที่ 2.5 แสดงการสื่อสารอนุกรม แบบซิงโครนัส (Synchronous)

2.2.1 Addressing

การสื่อสารแบบ I2C ประกอบด้วยรหัสประจำตัวของอุปกรณ์(Device Addressหรือ Slave Address) ประกอบด้วยบิต 1-7 และบิต 0 เป็นบิตควบคุมการเขียนอ่าน

1. รหัสประจำตัวของอุปกรณ์ ประกอบด้วยรหัสประจำตัวจากผู้ผลิต Product ID 4 บิต (บิต 4-7) ที่เปลี่ยนแปลงแก้ไขไม่ได้ และ Address 3 บิต (บิต 1-3) ซึ่งผู้ใช้ สามารถ กำหนด เองได้ รวมแล้วเป็นรหัส 7 บิต ใช้ระบุตัวอุปกรณ์ ที่ต่ออยู่บนบัส จะมีค่าซ้ำกันไม่ได้

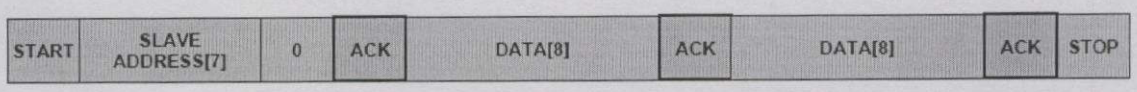
2. บิตควบคุมการเขียนอ่าน (Mode) บิต 0 เมื่อ MCU ต้องการเขียนข้อมูลไปยังอุปกรณ์ก็กำหนดให้บิตนี้เป็น 0 และเมื่อต้องการ อ่านข้อมูล จากอุปกรณ์ ก็กำหนดให้บิตนี้เป็น 1



รูปที่ 2.6 แสดง Addressing

2.2.2 Communication

1. การติดต่อสื่อสารเริ่มขึ้นเมื่อมีสภาวะ Start condition
2. Start bit จะตามด้วย Slave Address
3. Slave Address จะตามด้วย Read หรือ Write Bit
4. ทำการส่ง หรือ อ่าน Data
5. อุปกรณ์ที่รับข้อมูลต้องทำการส่ง Acknowledge bit
6. การติดต่อสื่อสารหยุดเมื่อมีสภาวะ Stop condition



รูปที่ 2.7 แสดงตัวอย่างการส่งข้อมูล (บิตควบคุมการเขียนอ่าน=0)

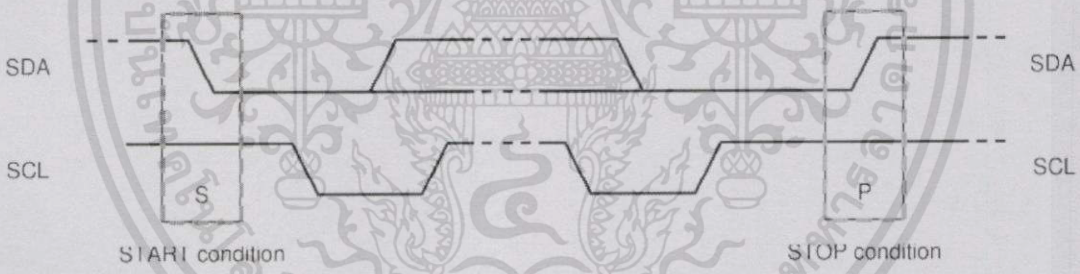
START	SLAVE ADDRESS[7]	1	ACK	DATA[8]	ACK	DATA[8]	ACK	STOP

รูปที่ 2.8 แสดงตัวอย่างการอ่านข้อมูล (บิตควบคุมการเขียนอ่าน=1)

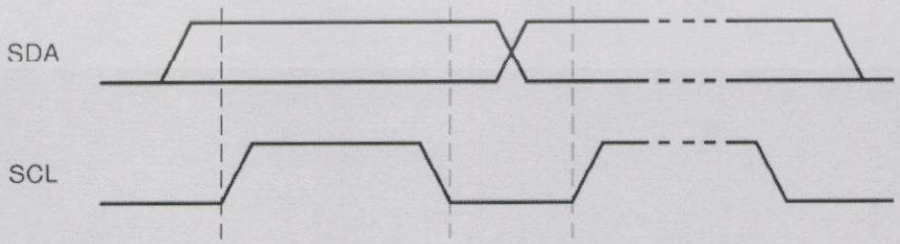
ตารางที่ 2.3 แสดงสภาวะระบบสื่อสาร

สภาวะบนระบบสื่อสาร	SDA	SCL
1. ไม่มีการสื่อสาร	High	High
2. เริ่มส่งข้อมูล(Start condition)	↓	High
3. หยุดส่งข้อมูล(Stop condition)	↑	High
4. การรับส่งข้อมูล 1 บิตใช้ clock 1 ลูกขณะรับส่งข้อมูล ข้อมูลบน SDA ต้องคงที่และ SDL เป็น High ข้อมูลบน SDA เปลี่ยนแปลงได้เมื่อ SCL เป็นLow	No chang chang	High Low
5. ส่ง Acknowledge bit เมื่อรับข้อมูลครบ 1 byte แล้ว	Low	High

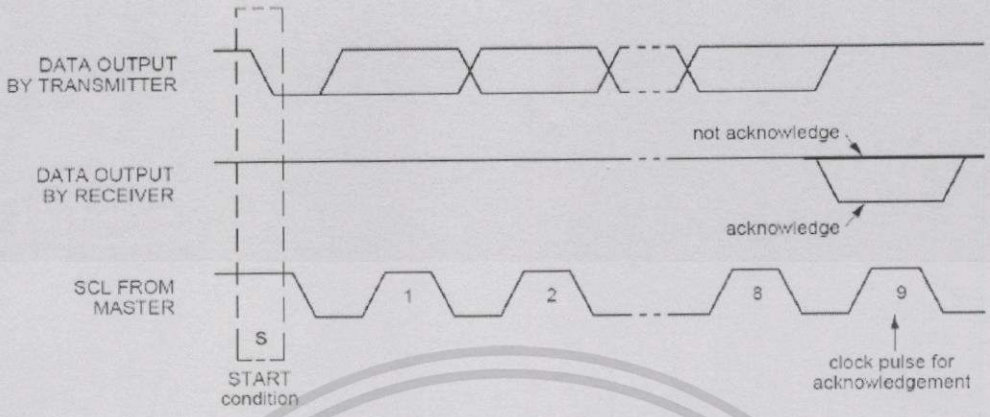
2.2.2.1 START & STOP conditions



รูปที่ 2.9 แสดงสภาพเริ่มและหยุดส่งข้อมูล



รูปที่ 2.10 แสดงสภาพระหว่างส่งข้อมูล



รูปที่ 2.11 แสดง Acknowledge / NOT-Acknowledge

2.2.3 ข้อกำหนดของ I2C

อุปกรณ์รับสัญญาณต้องส่งสัญญาณขอบขาลงไปที่สายสัญญาณ SDA ขณะที่สายสัญญาณ SCL High และต้องยังคงสถานะ Low เอาไว้ในขณะที่ สายสัญญาณ SCL ยังอยู่ที่ High

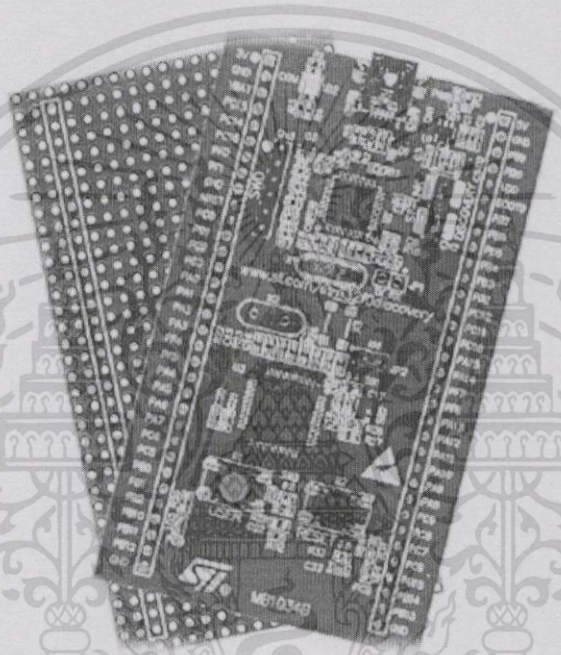
2.3 สรุป

ในบทนี้จะเป็นการอธิบายวิธีการควบคุมแบบ PID ตั้งแต่ความหมายของค่าพารามิเตอร์รวมถึงวิธีการปรับจูน และการติดต่อสื่อสารแบบ I2C โดยทั้งสองทฤษฎีนี้จะถูกนำมาใช้ในการทำงานร่วมกับไมโครคอนโทรลเลอร์เพื่อควบคุมกระบวนการรวมถึงการต่อต่อสื่อสารระหว่างอุปกรณ์แต่ละตัวกับตัวไมโครคอนโทรลเลอร์ด้วย โดยฮาร์ดแวร์และซอฟแวร์ที่ใช้ในการทำงานจะขออธิบายเพิ่มต่อในบทที่ 3

บทที่ 3
การประยุกต์ใช้ฮาร์ดแวร์และซอฟแวร์

3.1 ฮาร์ดแวร์

3.1.1 STM32F0DISCOVERY



รูปที่ 3.1 แสดง STM32F0DISCOVERY

❖ **ขั้นตอนการใช้งานเริ่มต้น**

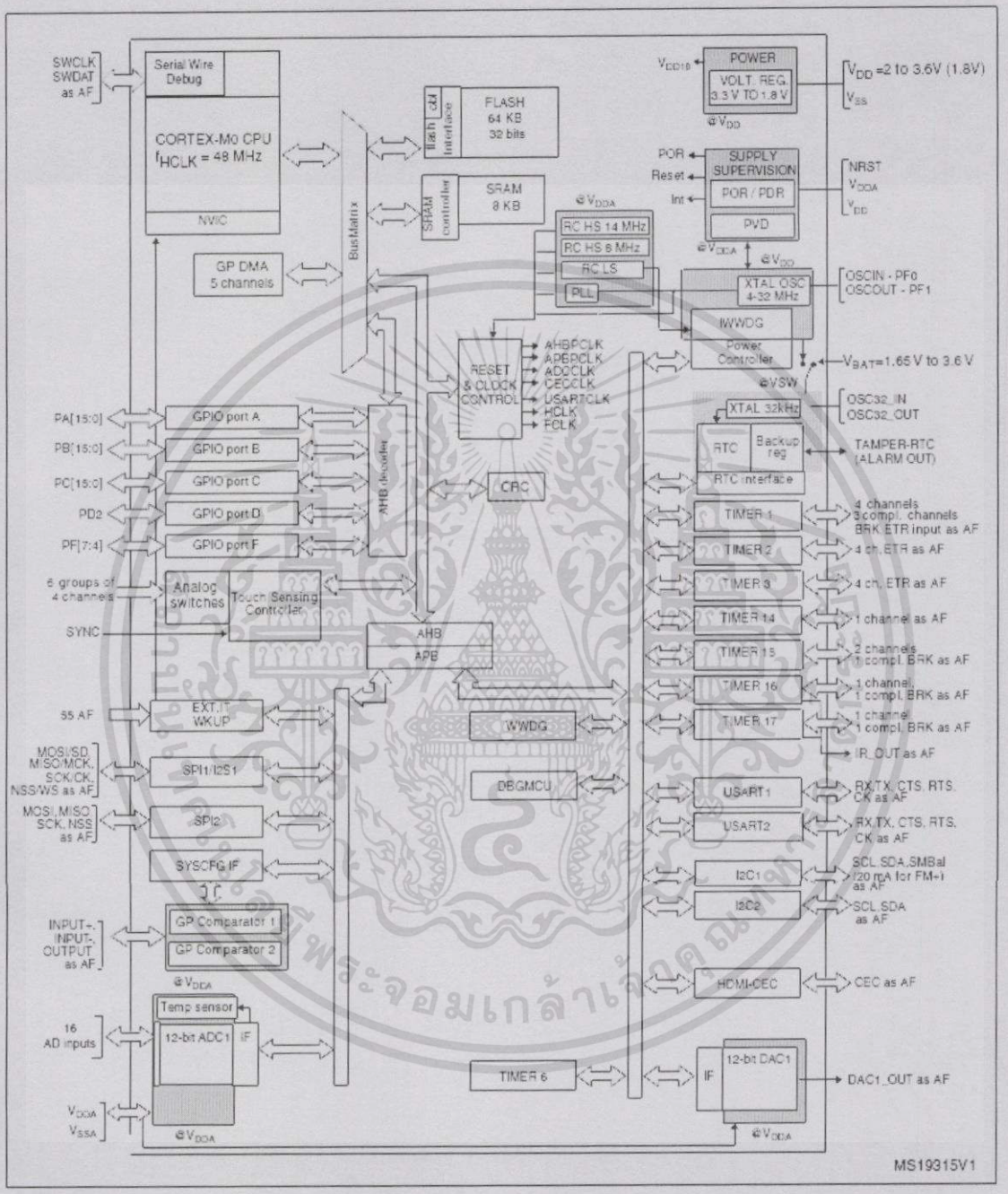
ลำดับขั้นตอนด้านล่างนี้เป็นการปรับแต่งไมโครคอนโทรเลอร์ STM32 เพื่อใช้งาน

- ตรวจสอบว่า JP2 และ CN2 ทำงาน
- เชื่อมต่อไมโครคอนโทรเลอร์กับเครื่อง PC ผ่านทางสาย USB เพื่อป้อนไฟให้กับบอร์ด โดยที่ LD1(PWR) และ LD2(COM) จะขึ้นสีแดงส่วน LD3จะติดสีเขียวแบบกระพริบ
- กดปุ่ม User button (ด้านล่างซ้ายของบอร์ด)
- สังเกตลักษณะของตัว LD3 ว่ามีการกระพริบแบบไหนจากการกดปุ่ม User button B1

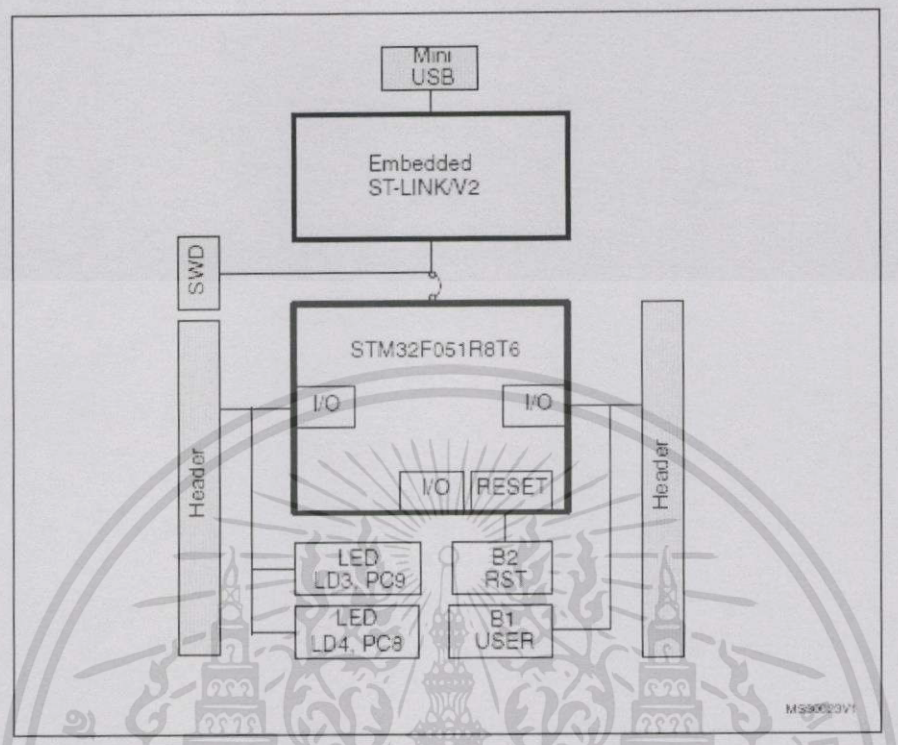
- การกดปุ่ม User button B1 แต่ละครั้งจะถูกยืนยันด้วยไฟสีฟ้าของ LD4

❖ คุณสมบัติ

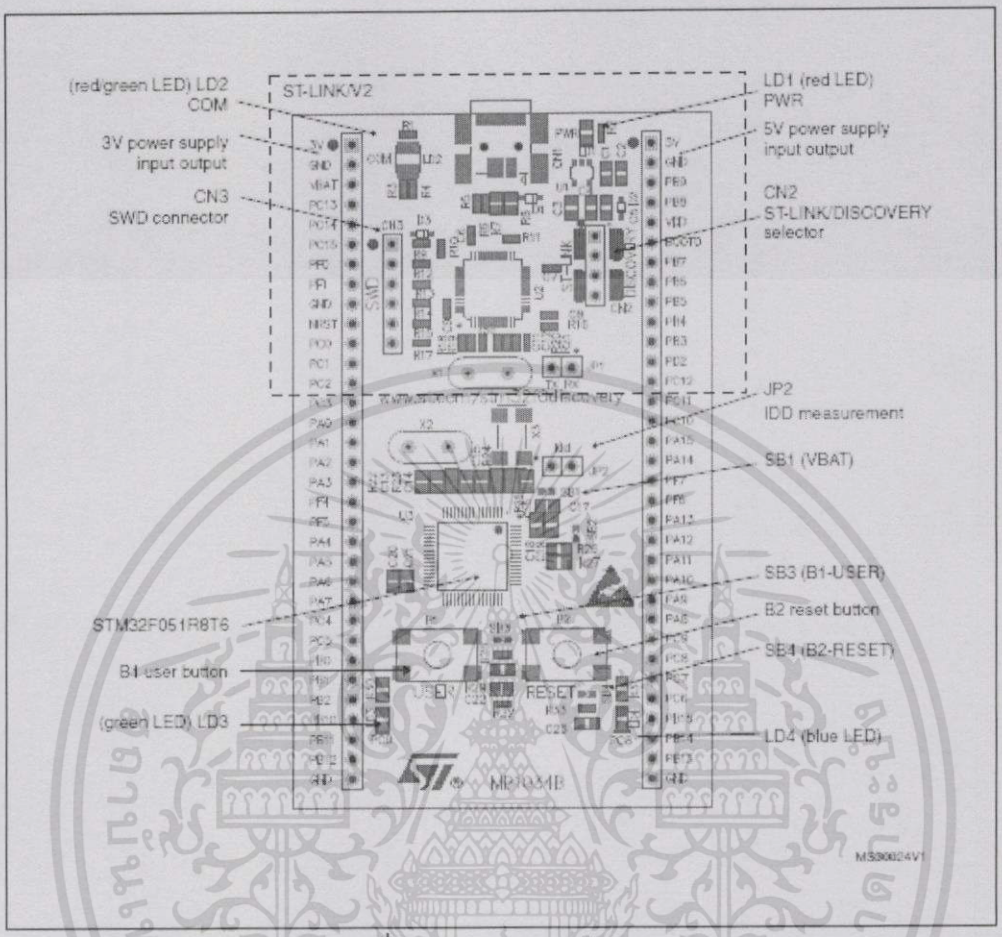
- คอร์และเงื่อนไขการทำงาน
  - –ARM® Cortex™-M0 0.9 DMIPS/MHz up to 48 MHz
- หน่วยความจำแฟลชขนาด 64 KB , แรม 8 KB
- เชื่อมต่อด้วย ST-LINK/V2 เพื่อโปรแกรมและดีบัค
- ใช้แหล่งจ่ายผ่าน USB bus หรือจาก แหล่งจ่ายภายนอกขนาด 5 V
- ใช้แหล่งจ่ายภายนอกขนาด 3 V และ 5 V
- LEDs 4 แบบ
  - –LD1 (แดง) สำหรับใช้งานแหล่งจ่าย 3.3 V
  - –LD2 (แดง/เขียว) สำหรับการเชื่อมต่อผ่าน USB
  - –LD3 (เขียว) สำหรับ PC9 output
  - –LD4 (ฟ้า) สำหรับ PC8 output
- ปุ่มกด 2 แบบ (user และ reset):
- การเชื่อมต่อที่มีประสิทธิภาพสูง
  - –6 Mbit/s USART
  - –18 Mbit/s SPI with 4- to 16-bit data frame
  - –1 Mbit/s I²C fast-mode plus
  - –HDMI CEC
- ส่งเสริมการควบคุม
  - –1x 16-bit 3-phase PWM motor control timer
  - –5x 16-bit PWM timers
  - –1x 16-bit basic timer
  - –1x 32-bit PWM timer
  - –12 MHz I/O toggling



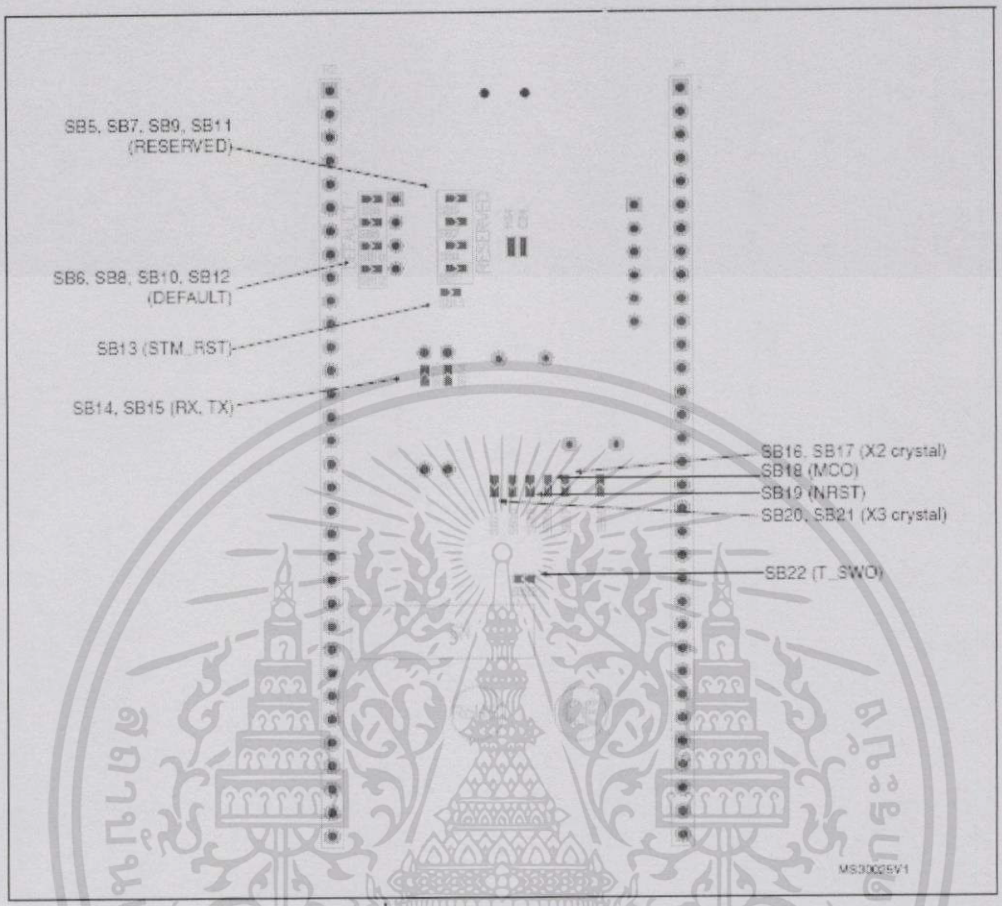
รูปที่ 3.2 แสดงภาพรวมอุปกรณ์



รูปที่ 3.3 แสดงฮาร์ดแวร์และแผนผัง



รูปที่ 3.4 แสดงมุมมองด้านบน



รูปที่ 3.5 แสดงมุมมองด้านล่าง

ตารางที่ 3.1 คำอธิบายหน้าที่ของแต่ละขา

ชื่อ	ตัวย่อ	คำอธิบาย
Pin name	Unless otherwise specified in brackets below the pin name, the pin function during and after reset is the same as the actual pin name	
Pin type	S	Supply pin
	I	Input only pin
	I/O	Input / output pin
I/O structure	FT	5 V tolerant I/O
	FTf	5 V tolerant I/O, FM+ capable
	TTa	3.3 V tolerant I/O directly connected to ADC
	TC	Standard 3.3V I/O
	B	Dedicated BOOT0 pin
	RST	Bidirectional reset pin with embedded weak pull-up resistor

ตารางที่ 3.2 แสดงฟังก์ชันของแต่ละขา

ชื่อขา (ฟังก์ชันหลังจากรีเซ็ท)	ประเภทของขา	โครงสร้างของ I/O	ฟังก์ชันของขา	
			ฟังก์ชันเลือก	ฟังก์ชันเพิ่มเติม
VBAT	S		Backup power supply	
PC13	I/O	TC		RTC_TAMP1, RTC_TS, RTC_OUT, WKUP2
PC14-OSC32_IN (PC14)	I/O	TC		OSC32_IN
PC15-OSC32_OUT (PC15)	I/O	TC		OSC32_OUT
PF0-OSC_IN (PF0)	I/O	FT		OSC_IN
PF1-OSC_OUT (PF1)	I/O	FT		OSC_OUT
NRST	I/O	RST	Device reset input / internal reset output (active low)	
PC0	I/O	TTa	EVENTOUT	ADC_IN10
PC1	I/O	TTa	EVENTOUT	ADC_IN11
PC2	I/O	TTa	EVENTOUT	ADC_IN12
PC3	I/O	TTa	EVENTOUT	ADC_IN13
VSSA	S		Analog ground	
VDDA	S		Analog power supply	
PA0	I/O	TTa	USART2_CTS, TIM2_CH1_ETR,	ADC_IN0, COMP1_INM6,

			COMP1_OUT, TSC_G1_IO1	RTC_TAMP2, WKUP1
PA1	I/O	TTa	USART2_RTS, TIM2_CH2, TSC_G1_IO2, EVENTOUT	ADC_IN1, COMP1_INP
PA2	I/O	TTa	USART2_TX, TIM2_CH3, TIM15_CH1, COMP2_OUT, TSC_G1_IO3	ADC_IN2, COMP2_INM6
PA3	I/O	TTa	USART2_RX, TIM2_CH4, TIM15_CH2, TSC_G1_IO4	ADC_IN3, COMP2_INP
PF4	I/O	FT	EVENTOUT	
PF5	I/O	FT	EVENTOUT	
PA4	I/O	TTa	SPI1_NSS/I2S1_WS, USART2_CK, TIM14_CH1, TSC_G2_IO1	ADC_IN4, COMP1_INM4, COMP2_INM4, DAC1_OUT
PA5	I/O	TTa	SPI1_SCK/I2S1_CK, CEC, TIM2_CH_ETR, TSC_G2_IO2	ADC_IN5, COMP1_INM5, COMP2_INM5
PA6	I/O	TTa	SPI1_MISO/I2S1_MCK, TIM3_CH1, TIM1_BKIN, TIM16_CH1, COMP1_OUT,	ADC_IN6

			TSC_G2_IO3, EVENTOUT	
PA7	I/O	TTa	SPI1_MOSI/I2S1_SD, TIM3_CH2, TIM14_CH1, TIM1_CH1N, TIM17_CH1, COMP2_OUT, TSC_G2_IO4, EVENTOUT	ADC_IN7
PC4	I/O	TTa	EVENTOUT	ADC_IN14
PC5	I/O	TTa	TSC_G3_IO1	ADC_IN15
PB0	I/O	TTa	TIM3_CH3, TIM1_CH2N, TSC_G3_IO2, EVENTOUT	ADC_IN8
PB1	I/O	TTa	TIM3_CH4, TIM14_CH1, TIM1_CH3N, TSC_G3_IO3	ADC_IN9
PB2	I/O	FT	TSC_G3_IO4	
PB10	I/O	FT	I2C2_SCL, CEC, TIM2_CH3, TSC_SYNC	
PB11	I/O	FT	I2C2_SDA, TIM2_CH4, TSC_G6_IO1, EVENTOUT	
VSS	S		Digital ground	
VDD	S		Digital power supply	
PB12	I/O	FT	SPI2_NSS, TIM1_BKIN,	

			TSC_G6_IO2, EVENTOUT	
PB13	I/O	FT	SPI2_SCK, TIM1_CH1N, TSC_G6_IO3	
PB14	I/O	FT	SPI2_MISO, TIM1_CH2N, TIM15_CH1, TSC_G6_IO4	
PB15	I/O	FT	SPI2_MOSI, TIM1_CH3N, TIM15_CH1N, TIM15_CH2	RTC_REFIN
PC6	I/O	FT	TIM3_CH1	
PC7	I/O	FT	TIM3_CH2	
PC8	I/O	FT	TIM3_CH3	
PC9	I/O	FT	TIM3_CH4	
PA8	I/O	FT	USART1_CK, TIM1_CH1, EVENTOUT, MCO	
PA9	I/O	FT	USART1_TX, TIM1_CH2, TIM15_BKIN, TSC_G4_IO1	
PA10	I/O	FT	USART1_RX, TIM1_CH3, TIM17_BKIN, TSC_G4_IO2	
PA11	I/O	FT	USART1_CTS, TIM1_CH4,	

			COMP1_OUT, TSC_G4_IO3, EVENTOUT	
PA12	I/O	FT	USART1_RTS, TIM1_ETR, COMP2_OUT, TSC_G4_IO4, EVENTOUT	
PA13(SWDAT)	I/O	FT	IR_OUT, SWDAT	
PF6	I/O	FT	2C2_SCL	
PF7	I/O	FT	2C2_SDA	
PA14(SWCLK)	I/O	FT	USART2_TX, SWCLK	
PA15	I/O	FT	SPI1_NSS/I2S1_WS, USART2_RX, TIM2_CH_ETR, EVENTOUT	
PC10	I/O	FT		
PC11	I/O	FT		
PC12	I/O	FT		
PD2	I/O	FT	TIM3_ETR	
PB3	I/O	FT	SPI1_SCK/I2S1_CK, TIM2_CH2, TSC_G5_IO1, EVENTOUT	
PB4	I/O	FT	SPI1_MISO/I2S1_MCK, TIM3_CH1, TSC_G5_IO2, EVENTOUT	
PB5	I/O	FT	SPI1_MOSI/I2S1_SD, I2C1_SMBA,	

			TIM16_BKIN, TIM3_CH2	
PB6	I/O	FTf	I2C1_SCL, USART1_TX, TIM16_CH1N, TSC_G5_IO3	
PB7	I/O	FTf	I2C1_SDA, USART1_RX, TIM17_CH1N, TSC_G5_IO4	
BOOT0	I	B	Boot memory selection	
PB8	I/O	FTf	I2C1_SCL, CEC, TIM16_CH1, TSC_SYNC	
PB9	I/O	FTf	I2C1_SDA, IR_OUT, TIM17_CH1, EVENTOUT	
VSS	S		Digital ground	
VDD	S		Digital power supply	

3.1.1.1 ARM® CortexTM-M0 core

หน่วยประมวลผล ARM Cortex ™-M0 เป็นหน่วยประมวลผมรุ่นล่าสุดของ ARM ซึ่งได้รับการพัฒนาให้เป็นอุปกรณ์ที่มีต้นทุนต่ำแต่สามารถตอบสนองการทำงานด้วย MCU ได้เช่นเดิม โดยมีการลดจำนวนขาแต่มีการใช้พลังงานลดลง ขณะที่ประสิทธิภาพของการประมวลผลยังคงดีเยี่ยมพร้อมทั้งระบบขั้นสูงที่ตอบสนองต่อสิ่งรบกวนต่างๆ

3.1.1.2 General-purpose inputs/outputs (GPIOs)

เราสามารถกำหนดขาแต่ละขาของ GPIO ให้เป็นเอาท์พุทหรืออินพุทได้ผ่านทางซอฟต์แวร์หรือจะสลับกับกับอุปกรณ์ภายนอกอื่นๆก็ได้ โดยส่วนมากขา GPIO จะถูกแบ่งเป็นฟังก์ชั่นดิจิตอลหรืออนาล็อกสลับกัน การกำหนดค่า I/O สามารถถูกล็อกได้ถ้าจำเป็นด้วยขั้นตอนพิเศษเพื่อหลีกเลี่ยงการเขียนข้อมูลผิดพลาดให้รีจีสเตอร์ I/O

3.1.1.3 ตัวแปลงค่าอนาล็อกไปเป็นดิจิตอล (ADC)

ตัวแปลงสัญญาณอนาล็อกเป็นดิจิตอลขนาด 12บิต ได้เพิ่มแชแนลภายในเป็น 16 ช่อง และภายนอก 3 ช่อง (เซ็นเซอร์อุณหภูมิ, แรงดันไฟฟ้าอ้างอิง, เครื่องมือวัดแรงดัน VBAT) และทำการแปลงได้ทั้งในโหมด Single-shot และ Scan modes โดยใน Scan modes จะทำการแปลงอัตโนมัติกับกลุ่มที่สัญญาอนาล็อกที่เลือกไว้
โดย ADC จะถูกสั่งการผ่านทางตัวควบคุม DMA

- ตัวแปลงค่าดิจิตอลไปเป็นอนาล็อก (DAC)

แชแนล DAC สามารถใช้ในการแปลงสัญญาณดิจิตอลให้เป็นสัญญาณอนาล็อก
สัญญาณดิจิตอลนี้สนับสนุนคุณสมบัติดังต่อไปนี้

- การจัดเรียงข้อมูลซ้ายหรือขวาในโหมด 12 บิต
- ความสามารถในการปรับปรุงข้อมูลให้ตรงกัน
- ความสามารถในการ DMA
- ตัวกระตุ้นจากภายนอกในการแปลงสัญญาณ

3.1.2 MCP3424

❖ คุณสมบัติ

- มีอินพุต 4 ช่อง
- I2C Interface
- มีโหมดมาตรฐาน, รวดเร็วและความเร็วสูง
- กำหนดแอดเดรสสำหรับเชื่อมต่อกับอุปกรณ์ภายนอก
- ใช้แหล่งจ่ายไฟ 2.7V to 5.5V
- ช่วงของอุณหภูมิในการใช้งาน -40°C to +125°C

- Voltage Reference ของบอร์ด (VREF): ความเที่ยงตรง: 2.048V ± 0.05%
- โปรแกรมเพื่อกำหนดค่า Gain ของบอร์ด (PGA):
  - ค่า Gains 1, 2, 4 หรือ 8

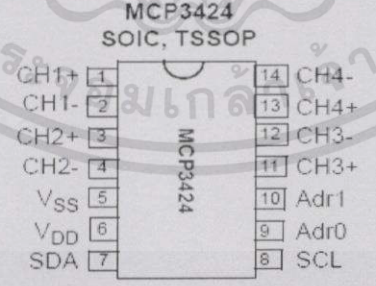
❖ **ภาพรวมอุปกรณ์**

MCP3424 เป็นตัวแปลงสัญญาณ analog เป็น digital ที่มีสัญญาณรบกวนน้อยและมีความเที่ยงตรงสูง อุปกรณ์ชิ้นนี้สามารถแปลงสัญญาณพร้อมให้ output ขนาด 18 บิต ในอัตรา 3.75,15,60 หรือ 240 ต่อวินาที ขึ้นอยู่กับการตั้งค่าผ่านทาง I2C อุปกรณ์สามารถสอบเทียบค่า offset และค่า gain ได้อัตโนมัติ อุปกรณ์ชิ้นนี้มามารถส่งข้อมูลได้อย่างถูกต้องในสภาวะที่มีรูปแบบความผันผวนของอุณหภูมิและแหล่งจ่ายไฟฟ้า

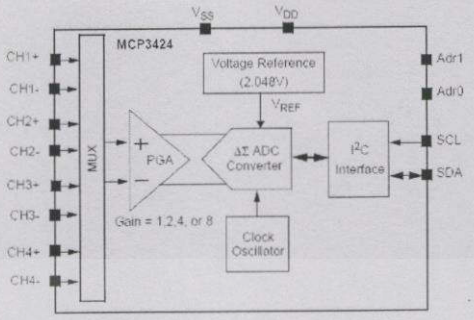
MCP3424 ,มีการแปลงสัญญาณสองแบบ

1. การแปลงสัญญาณแบบส่ง output ครั้งเดียว อุปกรณ์จะดำเนินการแปลงสัญญาณเพียงครั้งเดียว และจะส่งกระแสต่ำสแตนด์บายไว้อัตโนมัติจนกว่าจะมีคำสั่งแปลงสัญญาณครั้งต่อไป วิธีนี้จะลดการใช้กระแสไฟฟ้าลงอย่างมากในช่วงที่ไม่ได้มีการใช้งาน
2. การแปลงสัญญาณแบบส่ง output อย่างต่อเนื่อง จะแปลงสัญญาณอย่างต่อเนื่องโดยไม่หยุดตามความเร็วที่ได้ตั้งเอาไว้ การแปลงสัญญาณจะเลือกสัญญาณที่เข้ามาล่าสุดแปลงออกไปทาง output

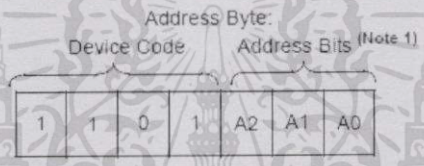
อุปกรณ์นี้จะดำเนินการเมื่อมีแหล่งจ่ายไฟฟ้า 2.7 V ถึง 5.5 V และมีสาย 2 เส้นในการต่อกับพอร์ต I2C ตำแหน่ง I2C address bit สำหรับ MCP3424 เลือกโดยใช่การเชื่อมต่อภายนอก (เลือกจาก pin Adr0 and Adr1



รูปที่ 3.6 แสดง Package Types



รูปที่ 3.7 แสดง Functional Block Diagram



รูปที่ 3.8 แสดง Address Byte

ตารางที่ 3.3 แสดงการทำงานของแต่ละขา

MCP3424	Sym	Description
1	CH1+	Positive Differential Analog Input Pin of Channel 1
2	CH1-	Negative Differential Analog Input Pin of Channel 1
3	CH2+	Positive Differential Analog Input Pin of Channel 2
4	CH2-	Negative Differential Analog Input Pin of Channel 2
5	Vss	Ground Pin
6	VDD	Positive Supply Voltage Pin
7	SDA	Bidirectional Serial Data Pin of the I2C Interface
8	SCL	Serial Clock Pin of the I2C Interface
9	Adr0	I2 C Address Selection Pin
10	Adr1	I2C Address Selection Pin

11	CH3+	Positive Differential Analog Input Pin of Channel 3
12	CH3-	Negative Differential Analog Input Pin of Channel 3
13	CH4+	Positive Differential Analog Input Pin of Channel 4
14	CH4-	Negative Differential Analog Input Pin of Channel 4

ตารางที่ 3.4 แสดงตำแหน่งบิต และ ขาเลือกตำแหน่ง สำหรับ MCP3424

I2C DeviceAddress Bits			Logic Status of Address Selection Pins	
A2	A1	A0	Adr0 Pin	Adr1 Pin
0	0	0	0 (Addr_Low)	0 (Addr_Low)
0	0	1	0 (Addr_Low)	Float
0	1	0	0 (Addr_Low)	1 (Addr_High)
1	0	0	1 (Addr_High)	0 (Addr_Low)
1	0	1	1 (Addr_High)	Float
1	1	0	1 (Addr_High)	1 (Addr_High)
0	1	1	Float	0 (Addr_Low)
1	1	1	Float	1 (Addr_High)
0	0	0	Float	Float

$$V_{IN} = (CHn+) - (CHn-)$$

$$V_{INCOM} = \frac{(CHn+) + (CHn-)}{2}$$

Where:

n = nth input channel (n=1, 2, 3, or 4)

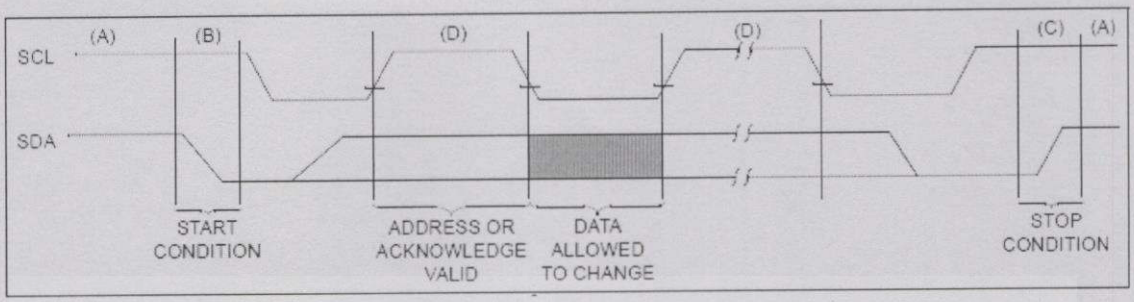
รูปที่ 3.9 แสดงย่านแรงดันอินพุต

$$\text{Number of Output Code} = (\text{Maximum Code} - 1) \times PGA \times \frac{(CHn+ - CHn-)}{2.048V}$$

Where:

See Table 4-3 for Maximum Code

รูปที่ 3.10 แสดงสมการ



รูปที่ 3.11 แสดง Data Transfer Sequence on I2C Serial Bus

3.1.3 MCP 4728

❖ **คุณสมบัติ**

- Voltage output DAC ขนาด 12-บิต
- เลือกใช้ Voltage reference ได้ทั้งภายในและภายนอก
- ย่านการใช้งานของ Voltage output : ใช้ V_{REF} ภายใน (2.048V) :
- 0.000V ถึง 2.048V กำหนดค่า Gain = 1
- 0.000V ถึง 4.096V กำหนดค่า Gain = 2

ถ้าใช้ V_{REF} ภายนอก (VDD) : 0.000V ถึง VDD

- เลือกโหมด Normal หรือ Power-Down
- ใช้พลังงานน้อย
- ทำงานโดยใช้แหล่งจ่ายไฟขนาด : 2.7V to 5.5V

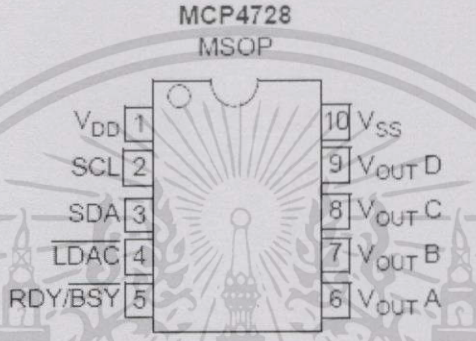
อินเตอร์เฟสของ I^2C :

- บิตแอดเดรส: ผู้ใช้สามารถโปรแกรมเข้าสู่ EEPROM
- โหมดมาตรฐาน (100 kbps), เร็ว (400 kbps) และ ความเร็วสูง (HS) (3.4 Mbps)
- ย่านอุณหภูมิที่ใช้งาน: -40°C ถึง +125°C

❖ **คุณสมบัติ**

อุปกรณ์ MCP4728 เป็นตัวแปลงสัญญาณ digital เป็น analog (DAC) ขนาด 12 บิต

ด้วยหน่วยความจำ (EEPROM) ค่า input code DAC, การกำหนดค่าบิตของอุปกรณ์, และบิตตำแหน่งของ I^2C จะถูกโปรแกรมลงที่หน่วยความจำ (EEPROM) โดยการใช้คำสั่งอินเตอร์เฟสของ I^2C คุณสมบัติของหน่วยความจำคือจะช่วยให้อุปกรณ์เก็บค่า input code ไว้ได้ในระหว่างที่ปิดไฟ, เรียกใช้ค่าเอาท์พุตได้ทันทีหลังจากเปิดไฟพร้อมกับบันทึกค่าที่เคยตั้งไว้ คุณลักษณะนี้มีประโยชน์อย่างมากในตอนที่อุปกรณ์ DAC นี้ถูกใช้เป็นตัวสนับสนุนอุปกรณ์อื่นในด้านเนตเวิร์ก



รูปที่ 3.12 แสดง Package Type

ตารางที่ 3.5 แสดงการทำงานของแต่ละขา

Pin No.	Name	Function
1	VDD	Supply Voltage
2	SCL	I2C Serial Clock Input
3	SDA	I2C Serial Data Input and Output
4	LDAC	This pin is used for two purposes: (a) Synchronization Input. It is used to transfer the contents of the DAC input registers to the output registers (VOUT). (b) Select the device for reading and writing I2C address bits.
5	RDY/BSY	This pin is a status indicator of EEPROM programming activity. An external pull-up resistor (about 100 kΩ) is

		needed from RDY/BSY pin to VDD line.
6	VOUT A	Buffered analog voltage output of channel A. The output amplifier has rail-to-rail operation.
7	VOUT B	Buffered analog voltage output of channel B. The output amplifier has rail-to-rail operation.
8	VOUT C	Buffered analog voltage output of channel C. The output amplifier has rail-to-rail operation.
9	VOUT D	Buffered analog voltage output of channel D. The output amplifier has rail-to-rail operation.
10	VSS	Ground reference.

$$V_{OUT} = \frac{(V_{REF} \times D_n)}{4096} \times G_x \leq V_{DD}$$

Where:

V_{REF} = 2.048V for internal reference selection

D_n = DAC input code

G_x = Gain Setting

รูปที่ 3.13 แสดง VOUT FOR VREF = INTERNAL REFERENCE

$$V_{OUT} = \frac{(V_{DD} \times D_n)}{4096}$$

Where:

D_n = DAC input code

รูปที่ 3.14 แสดง VOUT FOR VREF = VDD

3.1.4 LCD 4x20

LCD รายละเอียดเกี่ยวกับโมดูล LCDในโมดูล LCD จะมีส่วนประกอบหลักๆ 3 ส่วนดังนี้

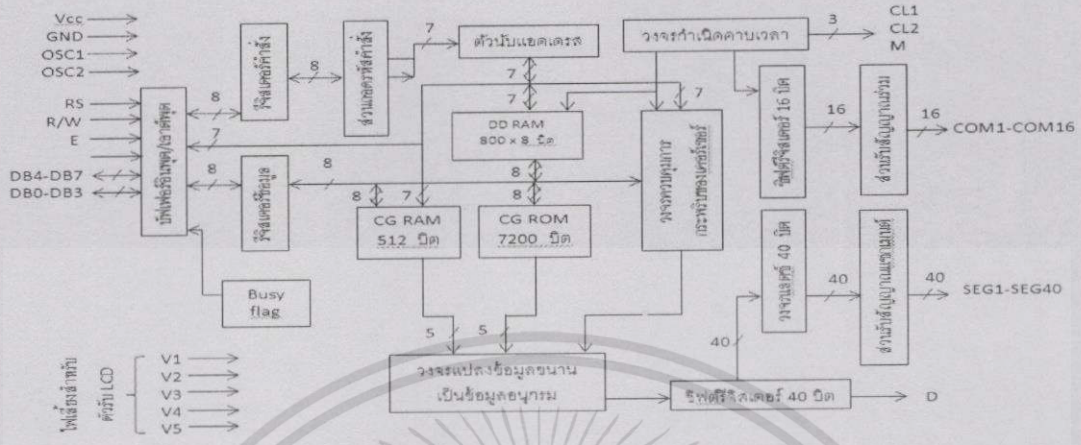
- **ตัวแสดงผล(display)**ภายในเป็นผลึกเหลวที่สามารถแสดงผลให้เห็นโดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมในการมองข้อมูลที่แสดงผลบนจอ LCD

- **ตัวควบคุม(controller)**เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาควบคุมการทำงานของโมดูล LCD เช่น ลบจอภาพ แสดงตัวอักษร หรือเลื่อนเคอร์เซอร์ เป็นต้น ตัวควบคุมนี้ใช้ชิปควบคุมโดยเฉพาะ ชิปที่นิยมใช้คือเบอร์ HD44780 และ HD61830 โดย HD44780 ใช้ควบคุม LCD แบบอักขระ ส่วน HD61830 ใช้ควบคุม LCD กราฟฟิก
- **ตัวขับ(driver)**เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผล แสดงข้อมูลตามที่กำหนดชิปที่ใช้ทำหน้าที่เป็นตัวขับนี้ได้แก่ เบอร์ HD44100H , MSM5259 เป็นต้น

3.1.4.1 โครงสร้างภายในของตัวควบคุมโมดูล LCD

ในการใช้งานโมดูล LCD จำเป็นต้องทำความเข้าใจเกี่ยวกับโครงสร้าง และคำสั่งที่ใช้ในการควบคุมให้ดีเสียก่อน ในนี้จะขอยกตัวอย่าง LCD แบบอักขระ เพราะสามารถเข้าใจได้ง่าย ให้รูปที่ P15-1 เป็นบล็อกไดอะแกรมภายในของชิปควบคุม LCD เบอร์ HD44780 ซึ่งใช้ในโมดูล LCD แบบอักขระ ประกอบด้วย

- **บัฟเฟอร์อินพุตเอาต์พุต** เป็นส่วนที่ใช้ติดต่อรับส่งข้อมูลกับตัวอุปกรณ์ภายนอกเพื่อถ่ายทอดข้อมูลเข้าออกภายในตัวควบคุม
- **รีจิสเตอร์คำสั่ง(Instruction Registor:IR)** เป็นรีจิสเตอร์ใช้รับข้อมูลคำสั่งจากอุปกรณ์ภายนอกเพื่อนำไปควบคุมการแสดงผล
- **รีจิสเตอร์ข้อมูล (Data Registor : DR)** เป็นรีจิสเตอร์ใช้รับข้อมูลจากอุปกรณ์ภายนอกเพื่อส่งต่อไปยังหน่วยความจำที่ทำหน้าที่เก็บข้อมูลแสดงผลหรือนำข้อมูลไปสร้างตัวอักษรเพิ่มเติมในแรมเก็บตัวอักษร
- **แรมเก็บข้อมูลแสดงผล(display Data RAM : DDRAM)** เป็นหน่วยความจำแรมทำหน้าที่เก็บข้อมูลที่มาจากรีจิสเตอร์ DR ตัวควบคุมจะนำข้อมูลใน DDRAM นี้ไปเปิดตาราง (Look up-lable) ของตัวอักษรที่เก็บไว้ในหน่วยความจำรอมและแรมเก็บตัวอักษรเพื่อนำไปแสดงที่ตัวแสดงผล



รูปที่ 3.15 แสดงไดอะแกรมการทำงานของโมดูล LCD แบบอักขระ

- **รอมเก็บตัวอักษร (Character Generator ROM : CDROM)**

เป็นหน่วยความจำรอมที่ใช้เก็บข้อมูลตัวอักษรหรือสัญลักษณ์ที่สามารถอ่านออกไปแสดงที่ตัวแสดงผลได้ มีขนาด 7200 บิต โดยจะถูกอ่านได้ด้วยค่าของข้อมูลใน DDRAM

- **แรมเก็บตัวอักขระ (Character Generator RAM : CGRAM)**

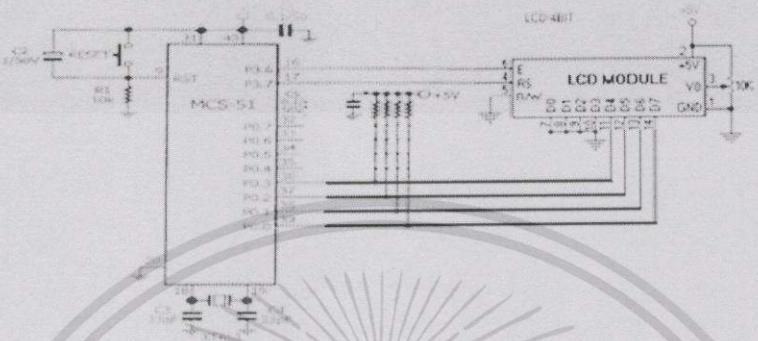
เป็นหน่วยความจำแรมความจุ 512 บิตใช้เก็บอักษรที่มีการสร้างเพิ่มเติมขึ้นใหม่ ในกรณีที่ตัวอักษรใน CGROM ไม่เพียงพอ การเขียนและอ่านค่าไปใช้นั้นทำได้เช่นเดียวกับ CGROM คือ เขียนข้อมูลลงใน DDRAM แล้วตัวข้อมูลจะมาอ่านค่าจาก CGRAM เอง

แฟลก BUSY เป็นส่วนที่ทำหน้าที่แจ้งสถานะการทำงานของตัวควบคุมให้อุปกรณ์ภายนอกทราบว่าตัวควบคุมพร้อมที่จะรับข้อมูลหรือคำสั่งหรือไม่ดังนั้นก่อนการส่งข้อมูลหรือคำสั่งมายังตัวควบคุมต้องตรวจสอบสถานะของ แฟลก BUSY นี้เสียก่อน

3.1.4.2 โมดูล LCD ขนาด 20 ตัวอักษร 4 บรรทัด (LCD 20x4)

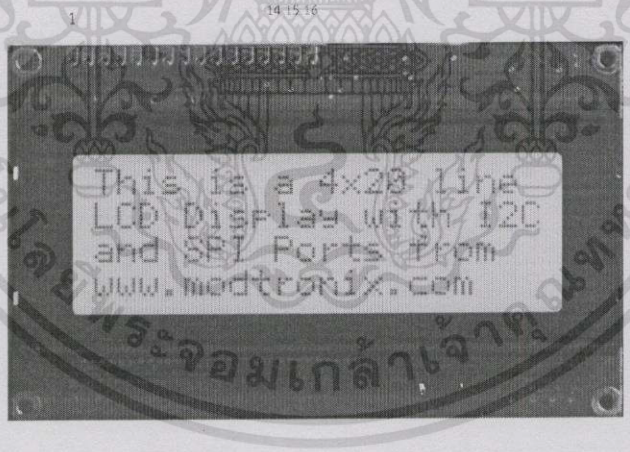
สำหรับโมดูล LCD ที่ยกมาใช้ในการศึกษา เป็นขนาด 20 ตัวอักษร 4 บรรทัด เนื่องจากราคาถูก หาง่าย และเป็นโมดูล LCD ที่มีโครงสร้างเป็นมาตรฐาน มีผู้ผลิตหลายราย และมีการ

ระบุเบอร์แตกต่างกันออกไปตามผู้ผลิต อาทิ LM020L ของฮิตาชิ,DMC-16117A ของคอปเทร็ก (Optrex) เป็นต้น แต่อย่างไรก็ตามคอนโทรลเลอร์ที่ใช้คือเบอร์เดียวกันนั่นคือเบอร์ HD44750 ของฮิตาชิ



รูปที่ 3.16 แสดงวงจรสำหรับทดลองการแสดงข้อความบนโมดูล LCD 20 ตัวอักษร 4 บรรทัด

โมดูล LCD ขนาด20 x4 มีขาต่อใช้งานทั้งสิ้น 14 ขา มีการจัดขา มีการจัดขาดังในรูปที่ P15-2 สำหรับรายละเอียดการทำงานของแต่ละขามีดังนี้



รูปที่ 3.17 แสดง LCD 4x20

ตารางที่ 3.6 รูปร่างและการจัดขาโมดูล LCD แบบอักขระ

RS	R/W	E	การทำงาน
0	0		เขียนคำสั่ง
0	1		อ่านสถานะของโมดูล LCD
1	0		เขียนข้อมูล
1	1		อ่านข้อมูล

ตารางที่ 3.7 แสดงการหน้าที่การทำงานของแต่ละขา

ขา	สัญลักษณ์	คำอธิบาย	ระดับ	หน้าที่	
1	VSS	Ground	-	0V	Ground
2	VDD	Power Supply	-	+5V	ต่อกับแรงดันไฟเลี้ยง +5V
3	V0	LCD Control	-	-	ต่อกับแรงดันเพื่อปรับความเข้มของการแสดงผล
4	RS	Register Select	H/L	RS=0 หมายถึงต้องการติดต่อกับรีจิสเตอร์คำสั่ง (Instruction Register)	
5	R/W	Read/Write	H/L	RS=1 หมายถึงต้องการติดต่อกับรีจิสเตอร์ข้อมูล (Data Register)	
6	E	Enable	H,H->L	Enable Signal	
7-14	DB0-DB7	Data Bus	H/L	Data Bus Line	
15	A	Back Light A	-	Back Light +5V (สำหรับรุ่นที่มี Back Light)	
16	K	Back Light K	-	Back Light 0V (สำหรับรุ่นที่มี Back Light)	

Vss (ขา1) : ต่อกราวด์

Vdd (ขา2): ต่อไฟเลี้ยง +5V

Vo (ขา3) : เป็นขาอินพุตรับแรงดันเพื่อนปรับความเข้มของการแสดงผล

RS(ขา4): เป็นขาอินพุตใช้ในการแยกชนิดของข้อมูลที่ทำการประมวลผลในขณะนั้น ว่าเป็นคำสั่งสำหรับรีจิสเตอร์ IR หรือเป็นข้อมูลสำหรับรีจิสเตอร์ DR โดยถ้าขานี้เป็น “00” ข้อมูลที่ส่งมาจะเป็นคำสั่ง แต่ถ้าขานี้เป็น “1” ข้อมูลที่ส่งมาจะเป็นข้อมูลที่จะแสดงผล

R/W(ขา5): เป็นขาที่ใช้เลือกการอ่านหรือเขียนข้อมูลกับโมดูล LCD ถ้าเป็น “0” เป็นการกำหนดให้เขียนข้อมูล แต่ถ้าเป็น “1” จะเป็นการอ่านข้อมูล

E (ขา6) : เป็นขาสำหรับรับสัญญาณพัลส์เอ็นเอเบิลโมดูล LCD ให้ทำงาน

D0-D7 (ขา 7-14): เป็นขาที่ใช้ปันทางผ่านของข้อมูลระหว่าง LCD กับอุปกรณ์ภายนอกขนาด 8 บิต อนึ่งขา RS,R/W และ E จะใช้ร่วมกัน

3.1.5 Keypad

แป้นกดนี้เป็นปุ่มกด 16 ปุ่มให้องค์ประกอบของอินเตอร์เฟซที่มีประโยชน์ต่อมนุษย์สำหรับโปรเจคไมโครคอนโทรลเลอร์ มีวิธีที่ง่ายในการติดปุ่มกดในความหลากหลายของการใช้งาน

❖ **คุณสมบัติ**

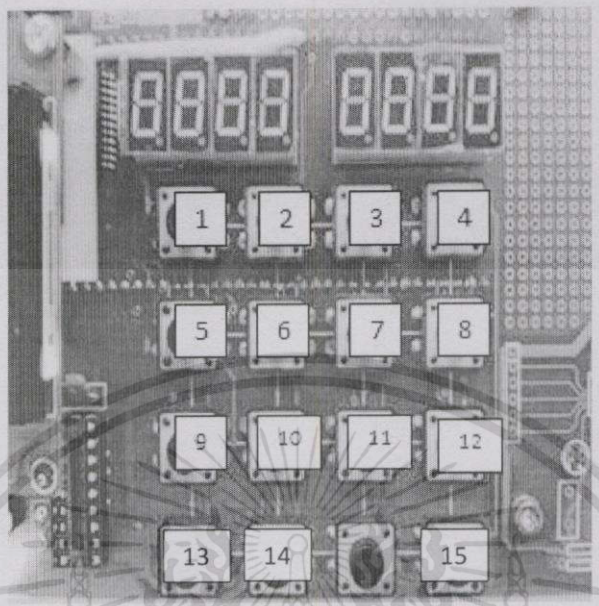
- อัตราส่วนราคาดี / ประสิทธิภาพ
- อินเตอร์เฟซที่ใช้งานง่ายกับไมโครคอนโทรลเลอร์อื่นๆ

❖ **คุณสมบัติที่สำคัญ**

- Ratingสูงสุด: 24 VDC ที่ 30 มิลลิแอมป์
- อินเตอร์เฟซ: 8 ขาเข้าถึงเมทริกซ์ 4x4
- อุณหภูมิในการทำงาน: 32 to 122 °F (0 to 50°C)

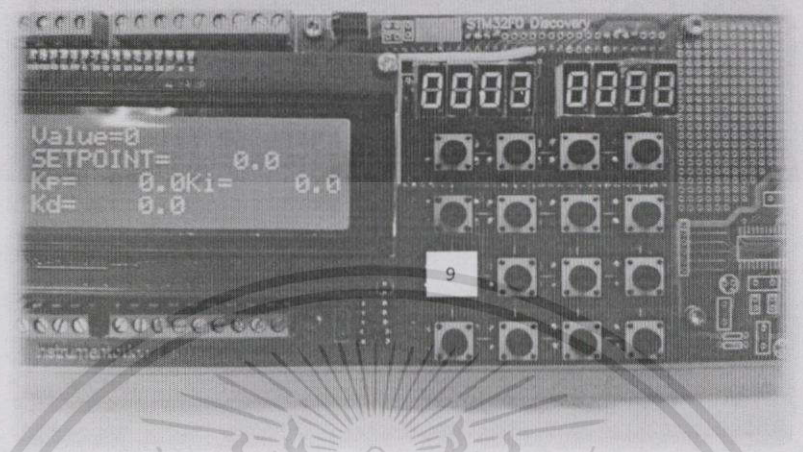
❖ **ไอเดียการประยุกต์ใช้**

- ระบบรักษาความปลอดภัย
- เลือกเมนู
- การป้อนข้อมูลสำหรับระบบ

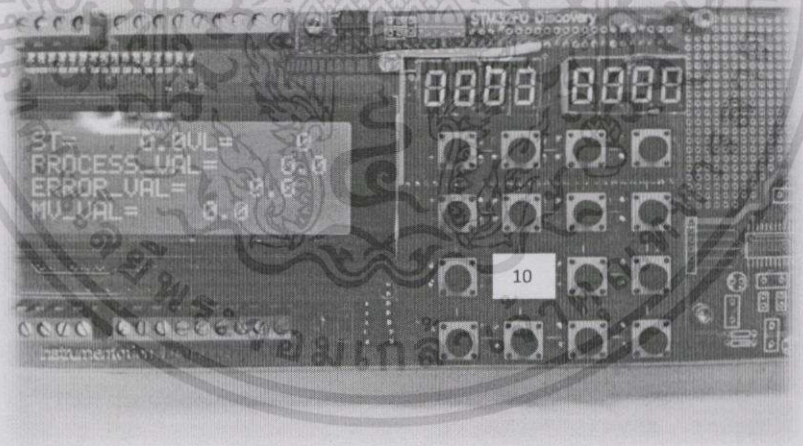


รูปที่ 3.18 แสดง keypad

ปุ่ม 1 = เพิ่มค่า +0.1
ปุ่ม 2 = เพิ่มค่า +1
ปุ่ม 3 = เพิ่มค่า +10
ปุ่ม 4 = เพิ่มค่า +100
ปุ่ม 5 = เลือกค่า Kp
ปุ่ม 6 = เลือกค่า Ki
ปุ่ม 7 = เลือกค่า Kd
ปุ่ม 8 = เลือกค่าที่ตั้งไว้
ปุ่ม 9 = ไปหน้า 1
ปุ่ม 10 = ไปหน้า 2
ปุ่ม 11 = รีเซ็ตค่าที่เรากดให้เป็นศูนย์
ปุ่ม 12 = เลือกปุ่ม ตกลง
ปุ่ม 13 = เลือก Mode Auto
ปุ่ม 14 = เลือก Manual
ปุ่ม 15 = เพิ่มค่า +1000

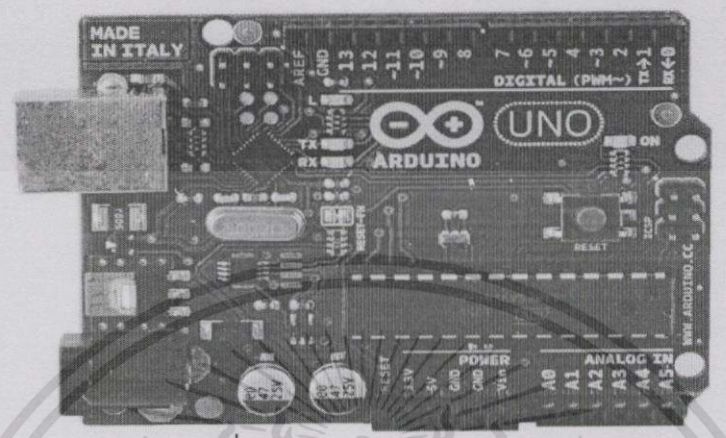


รูปที่ 3.19 แสดงหน้า 1 – ถ้ากดปุ่ม 9, จอภาพจะแสดงผลมาหน้านี้



รูปที่ 3.20 แสดงหน้า 2 – ถ้ากดปุ่ม 10, จอภาพจะแสดงผลมาหน้านี้

3.1.6 Microcontroller Arduino Uno



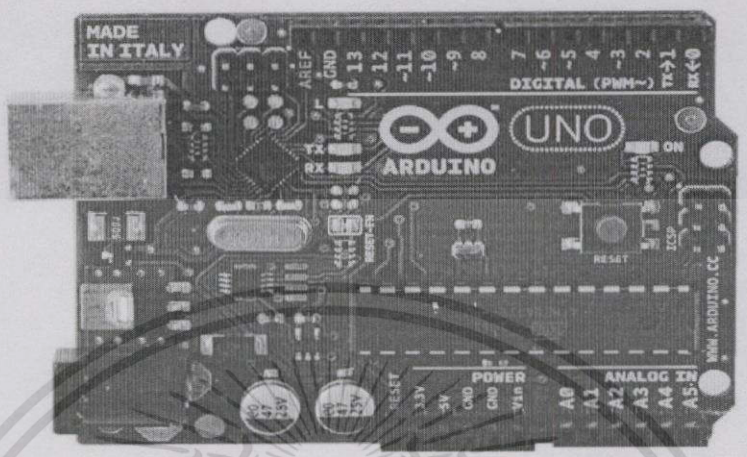
รูปที่ 3.21 แสดง Arduino Uno

Arduino Uno ที่ใช้เป็นบอร์ดไมโครคอนโทรลเลอร์รุ่น ATmega328 มี 14 ขาอินพุต / เอาท์พุตดิจิตอล (ขาที่ 6 สามารถใช้เป็นสัญญาณ PWM), 6 อินพุตอะนาล็อก, 16 MHz CRYSTAL OSCILLATOR, เชื่อมต่อด้วย USB, มีช่องเสียบไฟเลี้ยง, หัว ICSP และปุ่มรีเซ็ต ซึ่งมีทุกอย่างที่จำเป็นเพื่อการควบคุม ไฟเลี้ยงบอร์ดจะใช้จากการเชื่อมต่อกับคอมพิวเตอร์ด้วยสายเคเบิล USB หรือใช้ AC-to-DC adapter หรือแบตเตอรี่ Uno แตกต่างจากบอร์ดก่อนหน้านี้ตรงที่จะไม่ได้ใช้ FTDI USE-to-serial driver chip แต่จะมีคุณสมบัติ ATmega8U2 และมีโปรแกรมแปลง USB-to-serial converter Revision 2 ของ Uno board มี resistor pulling และสาย 8U2 HWB ต่อกับ ground

❖ **คุณสมบัติทั่วไปของ Arduino Uno**

- แรงดันไฟฟ้า 5V
- แรงดัน (ที่เหมาะใช้งาน) 7 - 12V
- แรงดันอินพุต (จำกัด) 6 – 20V
- ขาดิจิตอล I/O 14 (ขาที่ 6 ให้สัญญาณ PWM)
- ขาอนาล็อกอินพุต 6 ขา
- กระแสตรงในขา I/O 40 มิลลิแอมป์
- กระแสตรงสำหรับขา 3.3V 50 mA
- หน่วยความจำแฟลช 32 กิโลไบต์ (ATmega328) ที่ 0.5 กิโลไบต์โดยใช้ bootloader

สถาปัตยกรรมภายนอกของ Arduino Uno



รูปที่ 3.22 แสดงสถาปัตยกรรมของ Arduino Uno

ATmega 328P

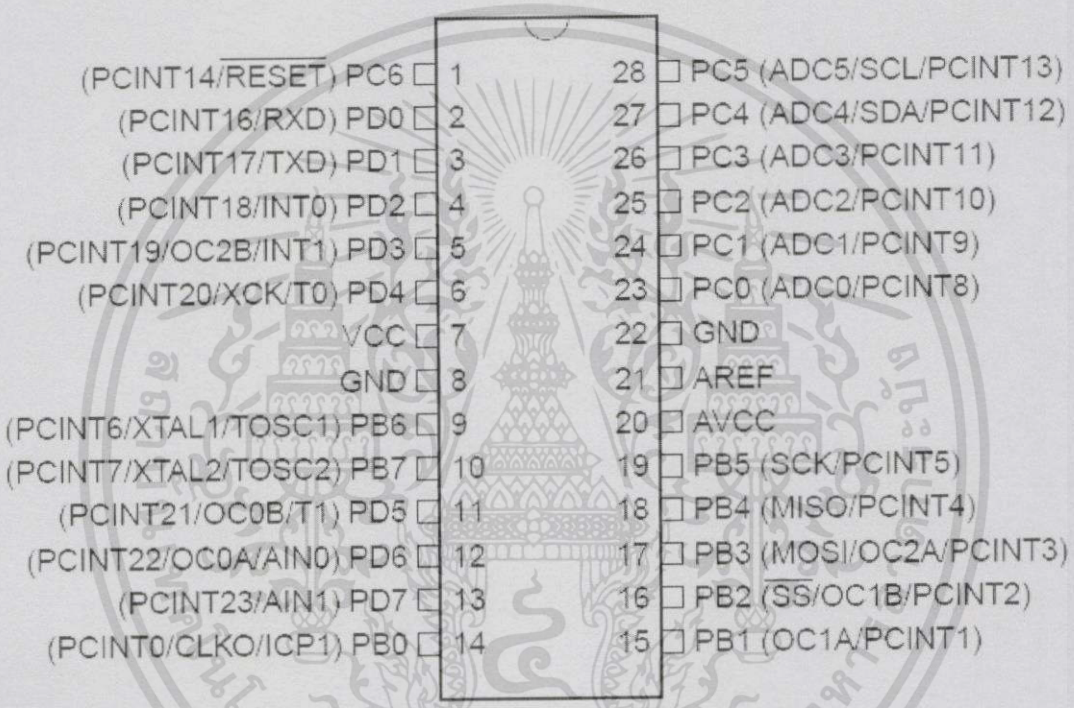
เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิตที่ใช้เป็นตัวประมวลผลหลักของบอร์ด Arduino Uno มีสมรรถนะในการทำงานค่อนข้างสูงอีกทั้งยังกินพลังงานต่ำ

คุณสมบัติเด่นของ ATmega 328P

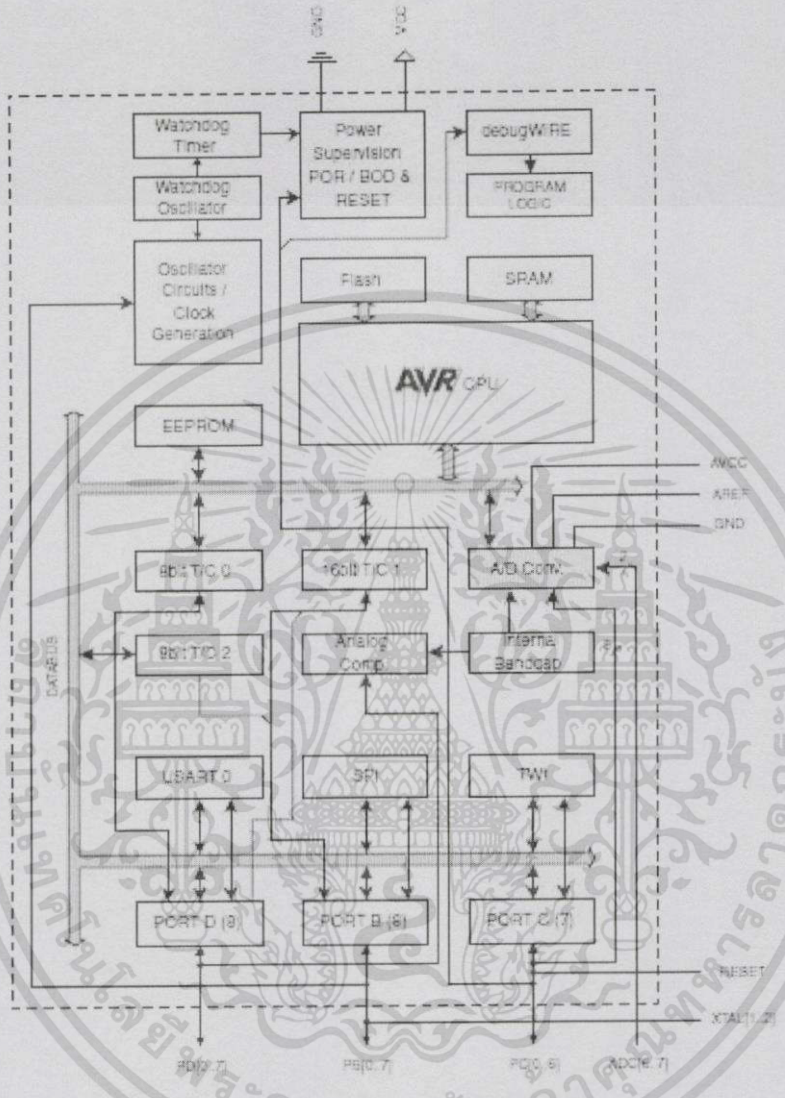
- ❖ หน่วยความจำ
  - หน่วยความจำแบบเฟลชสำหรับเก็บโปรแกรมขนาด 32 กิโลไบต์
  - หน่วยความจำแบบ EEPROM ขนาด 1 กิโลไบต์
  - หน่วยความจำ SRAM ภายในขนาด 2 กิโลไบต์
- ❖ แรงดันใช้งาน
  - 1.8-5.5 โวลต์
- ❖ อุณหภูมิทำงาน
  - -40 ถึง 85 องศาเซลเซียส
- ❖ อัตราการใช้พลังงาน
  - โหมดทำงาน 0.2 มิลลิแอมป์
  - โหมดปิดการทำงาน 0.1 ไมโครแอมป์
  - โหมดประหยัดพลังงาน 0.75 ไมโครแอมป์

❖ คุณสมบัติอื่นๆ

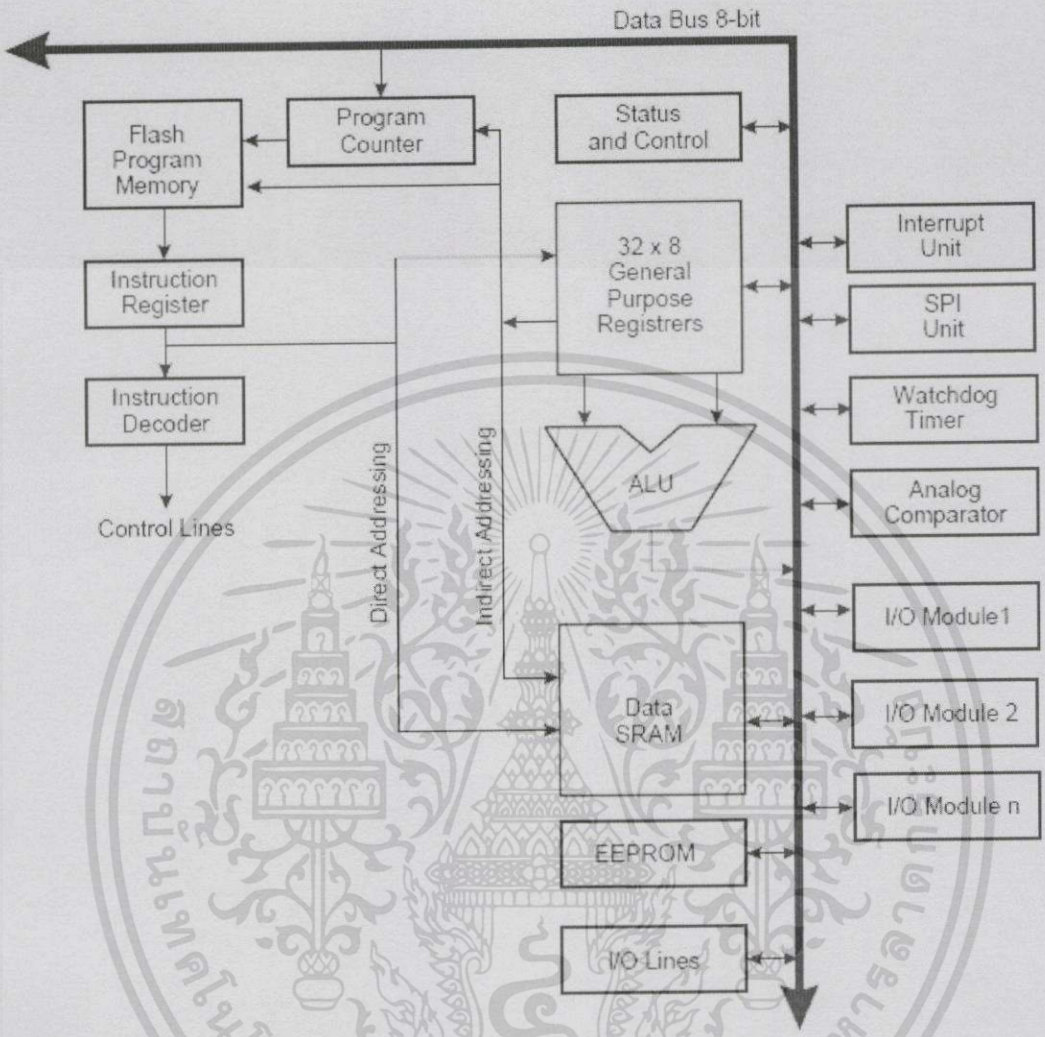
- ตัวนับ/ตัวจับเวลา 8 บิต สองตัว 16 บิตหนึ่งตัว
- ตัวจับเวลาจริงใช้ออสซิเลเตอร์แยก
- 6 ช่องสัญญาณ PWM
- โปรแกรมข้อมูลผ่าน USART



รูปที่ 3.23 แสดงขาต่างๆของ ATmega 328P



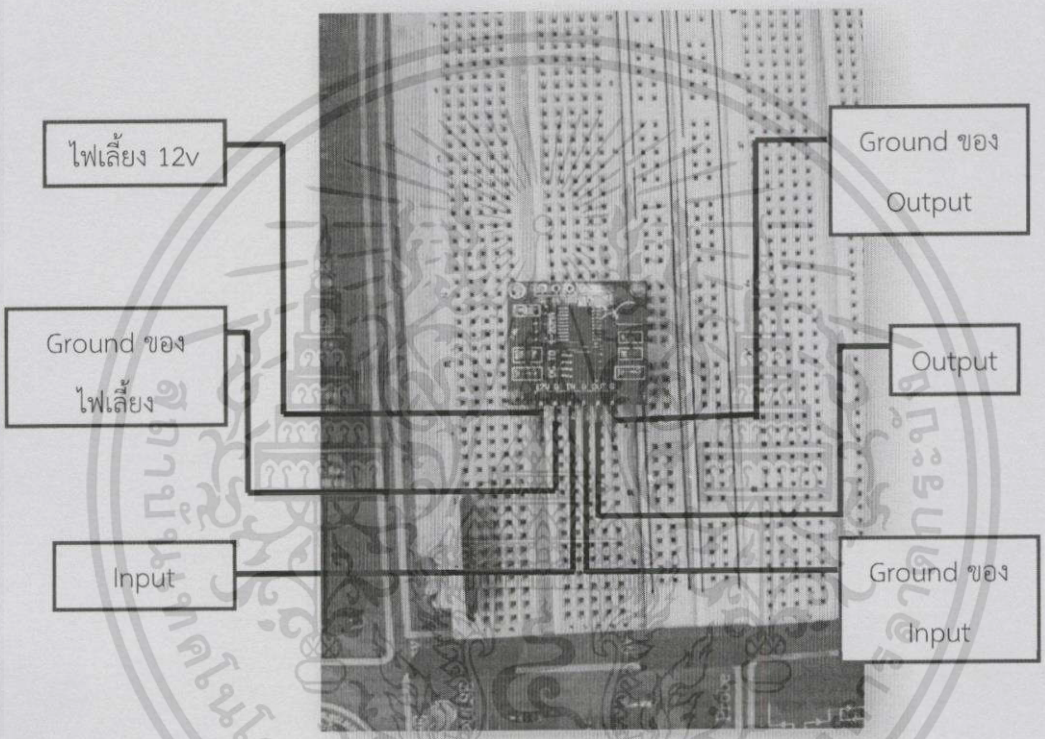
รูปที่ 3.24 แสดงสถาปัตยกรรมภายในของ ATmega 328P



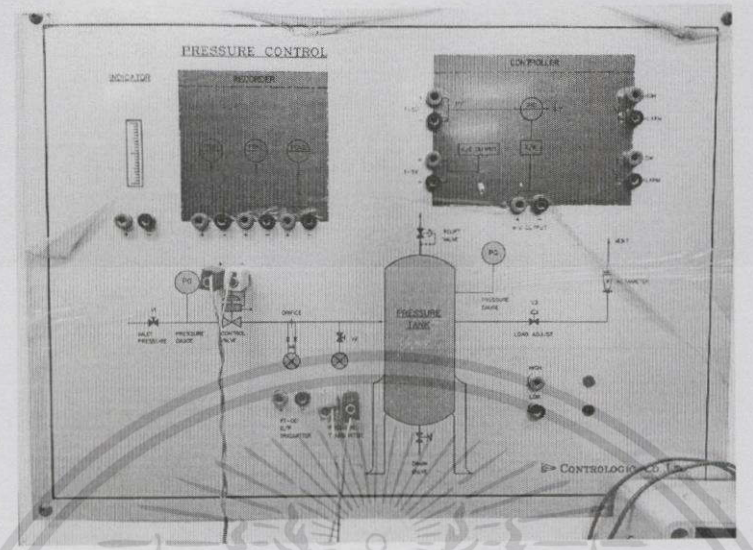
รูปที่ 3.25 แสดงบล็อกไดอะแกรมการทำงานของ ATmega 328P

3.1.7 Voltage to Current(V/I) Transmitter signal Module 0-5Vto4-20 mA linear conversion

รับสัญญาณ 0-5v จากไมโครคอนโทรเลอร์ เพื่อนำมาแปลงเป็น 4-20mA ก่อนจะส่งไปควบคุม Control Valve



รูปที่ 3.26 แสดงหน้าที่ของแต่ละขา



รูปที่ 3.27 ส่วนเชื่อมต่ออุปกรณ์ในกระบวนการกับอุปกรณ์ภายนอก

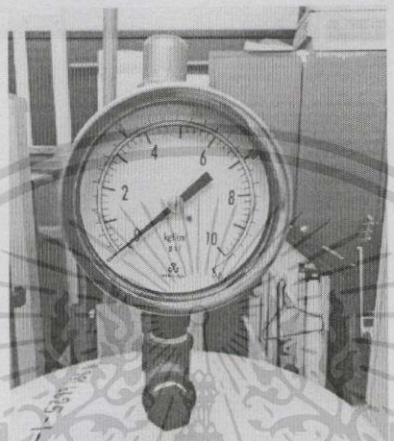
3.1.8 Load Adjust และ Rotameter

เป็นส่วนที่ใช้ปรับความดันที่ระบายออกจาก Pressure Tank โดยอัตราการไหลสามารถดูได้จาก Rotameter ที่ต่ออยู่กับ Load Adjust

รูปที่ 3.28 แสดง Load Adjust ที่ต่ออยู่กับ Rotameter

3.1.9 Pressure Gauge

อุปกรณ์วัดระดับความดัน สามารถอ่านค่าความดันได้จากหน้าปัดแสดงผลที่มีเข็มชี้บอกระดับความดัน



รูปที่ 3.29 แสดง Pressure Gauge

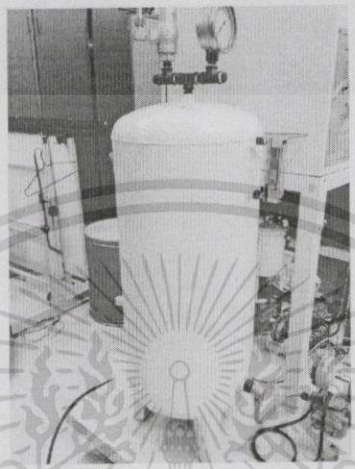
3.1.10 Relief Valve

วาล์วใช้ในการควบคุมหรือจำกัดแรงดันในระบบ เมื่อในระบบมีระดับความดันเกินที่ตั้งไว้จะมีการระบายความดันออก

รูปที่ 3.30 แสดง Relief Valve

3.1.11 Pressure Tank

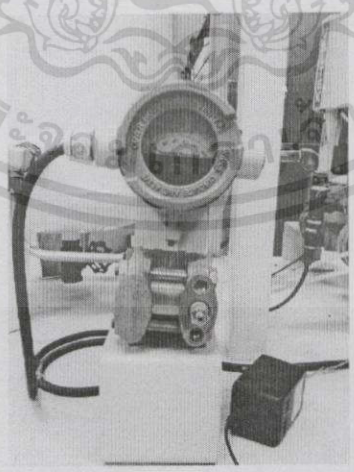
ถังสำหรับเก็บความดันในระบบ



รูปที่ 3.31 แสดง Pressure Tank

3.1.12 D/P transmitter

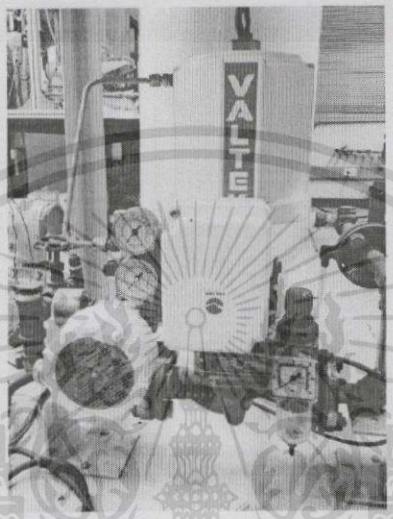
เป็นอุปกรณ์วัดความดัน โดยใช้หลักการความดันแตกต่างของระบบ กับ ความดันบรรยากาศส่งสัญญาณเอาท์พุตออกมาเป็น 1-5V



รูปที่ 3.32 แสดง D/P transmitter

3.1.13 Control valve

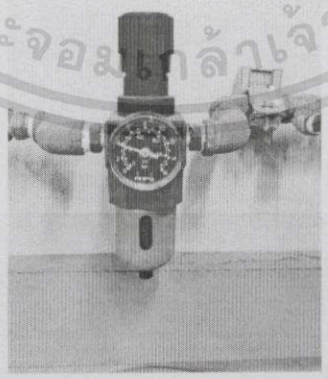
เป็นวาล์วสำหรับควบคุมการไหลเข้าของความดันลมใน Pressure Tank รับสัญญาณควบคุมเป็น 4-20mA



รูปที่ 3.33 แสดง Control valve

3.1.14 Regulator

ส่วนสหรับปรับแรงดันลมก่อนเข้าระบบ ไม่ให้แรงดันลมสูงเกินไป และ อยู่ในระดับที่อุปกรณ์สามารถทำงานโดยไม่เกิดการเสียหาย



รูปที่ 3.34 แสดง Regulator

3.2 ซอฟแวร์

3.2.1 โปรแกรมควบคุมไมโครคอนโทรลเลอร์

3.2.1.1 Arduino IDE

Arduino IDE เป็นโปรแกรมที่ใช้ในการออกแบบและเขียนโปรแกรมเพื่อควบคุมการทำงานของไมโครคอนโทรลเลอร์ตระกูล Arduino โดยตัวโปรแกรมเองนั้นพัฒนาขึ้นมาโดยใช้ภาษาจาวา แต่การเขียนโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์นั้นใช้ภาษาซีที่ถูกดัดแปลงให้สามารถเข้าใจได้ง่ายและใช้งานได้ง่ายขึ้นด้วย library พื้นฐานที่จำเป็นหลายอย่าง นอกจากใช้ในการเขียนโปรแกรมแล้วยังสามารถใช้ในการคอมไพล์เพื่อตรวจสอบโปรแกรมว่ามีข้อผิดพลาดหรือไม่และทำการอัพโหลดโปรแกรมไปยังบอร์ดได้ทันทีอีกด้วย โดยตัวโปรแกรมสามารถดาวน์โหลดได้ฟรีจากเว็บไซต์ของผู้ผลิต

รูปที่ 3.35 แสดงโปรแกรม Arduino IDE

การใช้งาน Arduino IDE

- เริ่มต้นด้วยการเปิดโปรมแกรม Arduino IDE ขึ้นมาหรือหากเปิดไว้อยู่แล้วต้องการสร้างใหม่ไปที่ File เลือก New หากต้องการสร้างไฟล์ใหม่หรือเลือก Open ในการเปิดไฟล์ที่มีอยู่แล้วและสามารถเลือกเปิดไฟล์ตัวอย่างได้จาก Example
- ต่อไปจะเป็นการเขียนโปรแกรม โดยโปรแกรมในการควบคุม MCU Arduino ที่เขียนบน Arduino IDE นั้นจะแบ่งเป็น 2 ฟังก์ชั่นหลักคือ

- **ฟังก์ชั่น setup**

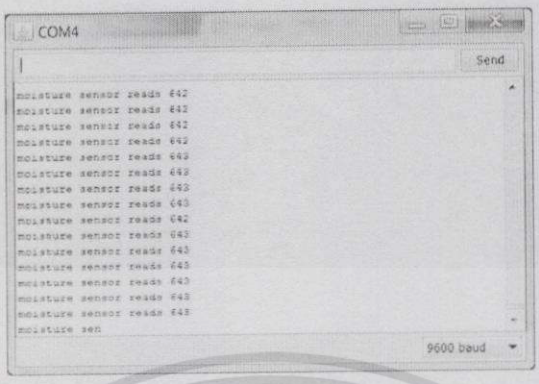
  เป็นฟังก์ชั่นที่ใช้ในการเตรียมค่าต่างๆก่อนที่จะทำงานเช่น การกำหนดค่าตัวแปรค่าคงที่ต่างๆ

- **ฟังก์ชั่น loop**

  คำสั่งที่อยู่ภายในฟังก์ชั่นนี้จะมีการทำงานวนซ้ำไปเรื่อยๆตั้งแต่คำสั่งแรกไปจนคำสั่งสุดท้ายแล้วกลับมาเริ่มที่คำสั่งแรกใหม่ ส่วนที่คือส่วนที่จะเป็นตัวกำหนดการทำงานของไมโครคอนโทรลเลอร์ตลอดที่ยังมีการจ่ายพลังงานให้กับมัน

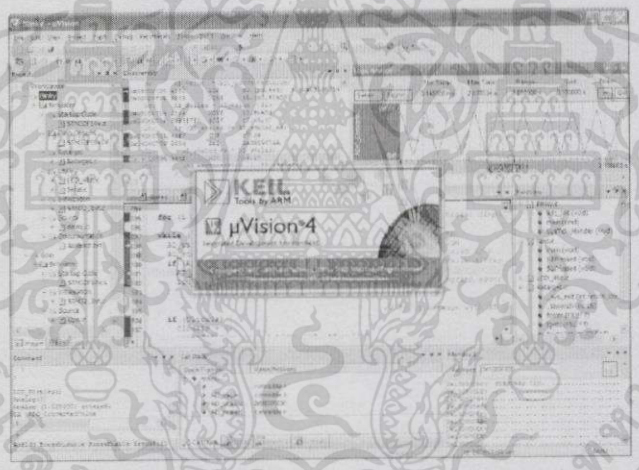
รูปที่ 3.36 แสดงตัวอย่างโปรแกรมของ Arduino IDE

- หลังจากที่ได้เขียนโปรแกรมเรียบร้อยขั้นต่อไปเป็นการตรวจข้อผิดพลาดของการเขียนโปรแกรมโดยการใช้งานปุ่ม verify หากโปรแกรมที่เขียนนั้นมีข้อผิดพลาดจะมีข้อความแสดงให้ทราบ
- หากโปรแกรมไม่มีข้อผิดพลาดใดๆขั้นตอนต่อไปคือการอัพโหลดโปรแกรมลงไปยังบอร์ดไมโครคอนโทรลเลอร์แต่ก่อนหน้านั้นควรมีการตั้งค่าการเชื่อมต่อก่อนด้วยการไปที่ Tools จากนั้นเลือก Board ทำการเลือกบอร์ดไมโครคอนโทรลเลอร์ให้ถูกต้องตามที่ใช้งานจากนั้นไปที่ port เลือกพอร์ตที่ไมโครคอนโทรลเลอร์เชื่อมต่ออยู่ หลังจากทำการตั้งค่าการเชื่อมต่อแล้วกดปุ่ม upload เพื่อทำการอัพโหลดโปรแกรมไปยังบอร์ดไมโครคอนโทรลเลอร์รอจนกระทั่งขึ้นข้อความ “Done uploading”
- การใช้งาน serial monitor เมื่อต้องการดูค่าต่างๆจากไมโครคอนโทรลเลอร์นั้นสามารถทำได้โดยใช้งานคำสั่ง Serial แต่ต้องทำการกำหนดค่า Baud rate ให้ตรงกันกับที่กำหนดไว้ในตัวโปรแกรมด้วย



รูปที่ 3.37 แสดงตัวอย่างการใช้งาน serial monitor

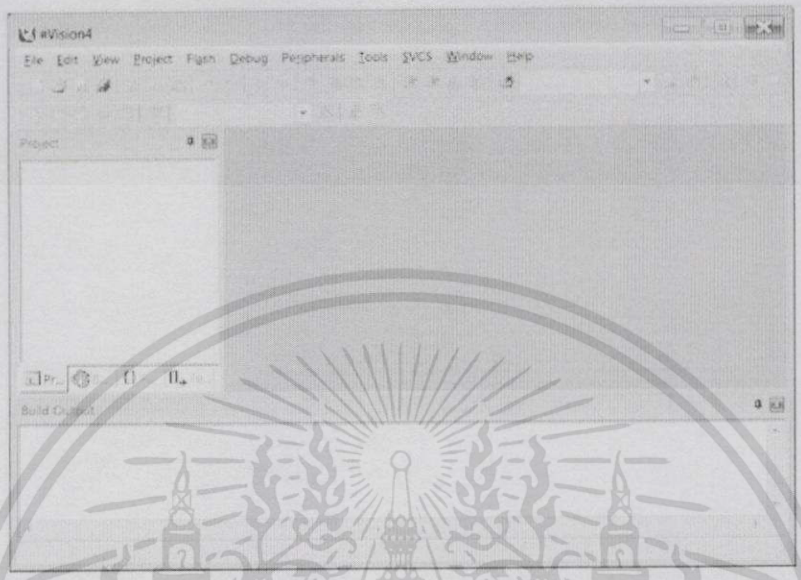
3.2.1.2 Keil MDK-ARM (Microcontroller Development Kit)



รูปที่ 3.38 แสดง Keil MDK-ARM

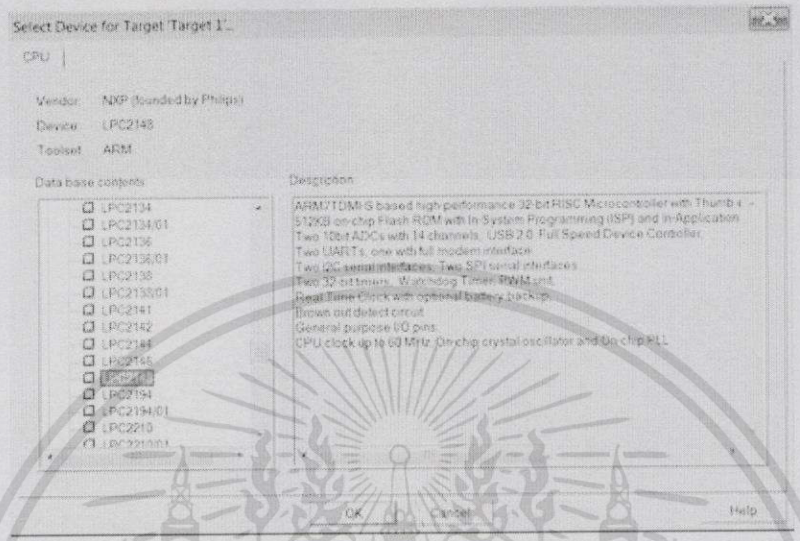
Keil MDK-ARM (Microcontroller Development Kit) เป็นโปรแกรมที่ประกอบด้วย ARM compilation tools สำหรับคอมไพล์โค้ด C/C++ และ Keil uVision ซึ่งทำหน้าที่เป็น IDE (Integrated Development Environment) นอกจากนั้นยังได้รวมไลบรารี่และระบบปฏิบัติการเวลาจริง (RTX Real-Time Operating System) ไว้เป็นตัวเลือกสำหรับการพัฒนาโปรแกรมระบบสมองกลฝังตัวที่มีไมโครคอนโทรลเลอร์เป็นหน่วยประมวลผล ซอฟต์แวร์นี้ใช้ได้กับไมโครคอนโทรลเลอร์หลายตระกูล เช่น ARM7TDMI, Cortex-M series เป็นต้น

เริ่มต้นด้วยการสร้างโปรเจคใหม่ (Create New Project) ในหน้าต่างหลักของ uVision4



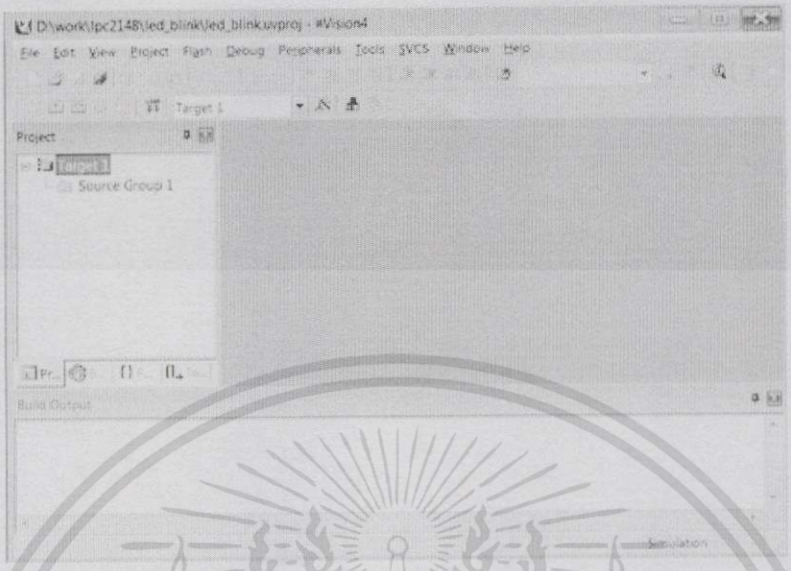
รูปที่ 3.39 แสดงหน้าต่างหลักของ uVision IDE

รูปที่ 3.40 แสดงกลับสู่หน้าต่างหลัก แล้วทำขั้นตอน “Build Target”
ถ้าไม่มี error จะได้ไฟล์ .hex ที่สามารถนำไปโปรแกรมลงชิปได้



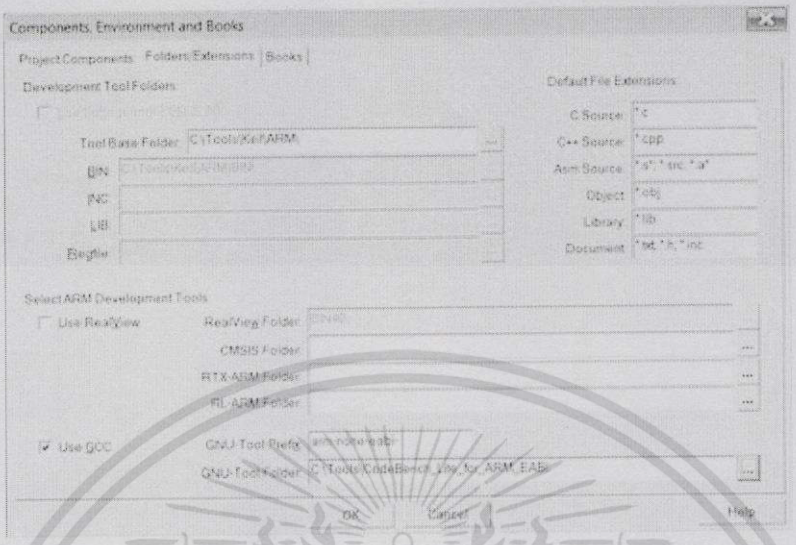
รูปที่ 3.41 แสดงเลือกชิปเป้าหมาย (เลือก NXP LPC2148)

รูปที่ 3.42 แสดงเมื่อถึงขั้นตอนที่ถามว่า จะสร้างไฟล์ Startup.s ลงในโปรเจคหรือไม่ ให้เลือกไม่ใส่ (No)

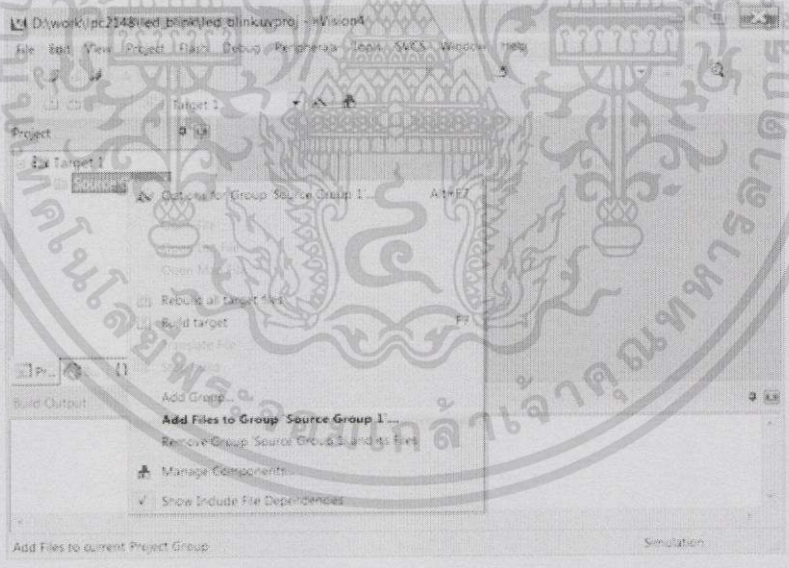


รูปที่ 3.43 แสดงกลับมาสู่หน้าต่างหลักเมื่อได้สร้างโปรเจคใหม่แล้ว

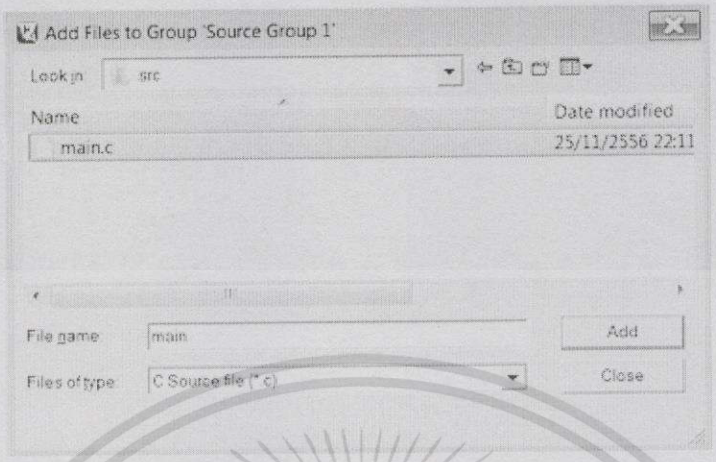
รูปที่ 3.44 แสดงเลือกเมนู Project – Manage – Components, Environment, Books ...
เพื่อเลือกคอมไพล์เลอร์ที่ต้องการใช้งาน



รูปที่ 3.45 แสดงเลือกใช้ “Use GCC” ซึ่งเป็นคอมไพล์เลอร์ของ Sourcery Code bench โดยจะต้องระบุ Folder / Directory (GNU-Tools Folder) ที่ได้ติดตั้งโปรแกรมดังกล่าว



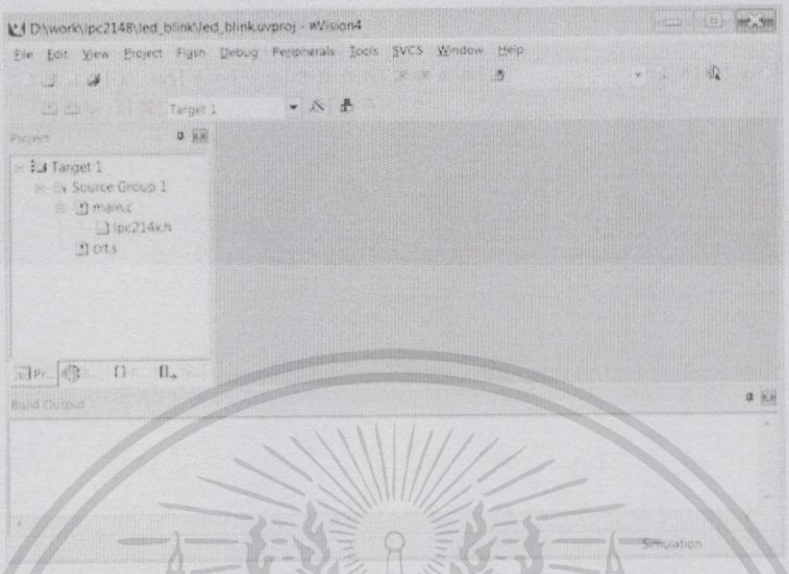
รูปที่ 3.46 แสดงขั้นตอนนี้เป็นการใส่ไฟล์ Source Code ลงในโปรเจค



รูปที่ 3.47 แสดงเลือกไฟล์ main.c (โค้ดภาษา C) จาก Source Code ตัวอย่าง

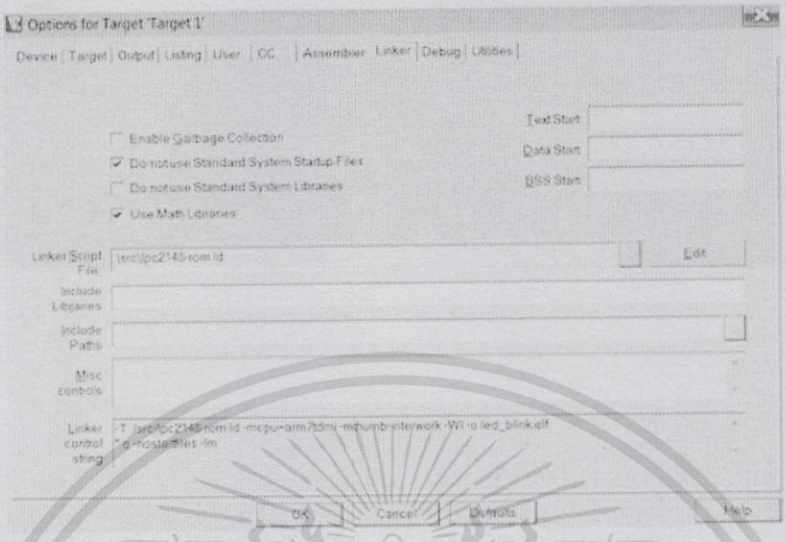
เพื่อใส่เป็นส่วนหนึ่งของโปรเจค

รูปที่ 3.48 แสดงเลือกไฟล์ crt.s (โค้ด Startup ภาษา Assembly) จาก Source Code ตัวอย่าง



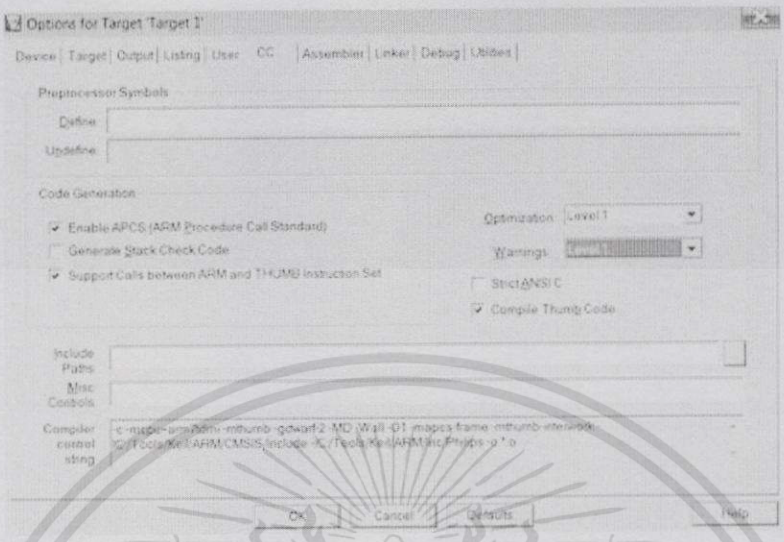
รูปที่ 3.49 แสดงกลับสู่หน้าต่างหลักเมื่อได้เพิ่มไฟล์จาก Source Code ตัวอย่างลงในโปรเจคแล้ว

รูปที่ 3.50 แสดงเลือกจากเมนู "Option for Target ..." และไปที่หน้า Tab "Output" แล้วเลือก "Create .HEX file"

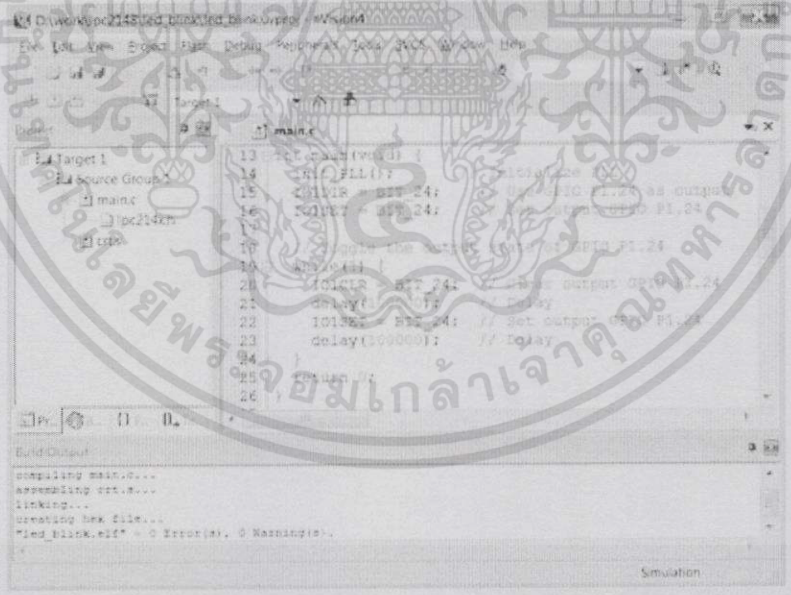


รูปที่ 3.51 แสดงไปที่หน้า Tab "Linker" เพื่อเลือกไฟล์ Linker Script

รูปที่ 3.52 แสดงเลือกไฟล์ Linker Script "lpc2148-rom.ld" จาก Source Code ตัวอย่าง



รูปที่ 3.53 แสดงไปที่หน้า Tab "CC" สามารถกำหนดตัวเลือกสำหรับ Compiler เช่น Optimization Level



รูปที่ 3.54 แสดงการกลับสู่หน้าต่างหลัก แล้วทำขั้นตอน "Build Target" ถ้าไม่มี error จะได้ไฟล์ .hex ที่สามารถนำไปโปรแกรมลงชิปได้

❖ **แถบเครื่องมือและไอคอนเครื่องมือ**

μVision IDE ประกอบด้วยแถบเครื่องมือที่มีปุ่มสำหรับเรียกใช้คำสั่งต่างๆมากมาย

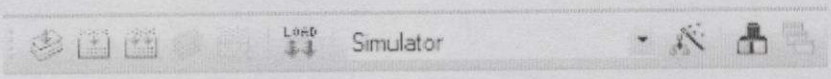
- แถบเครื่องมือไฟล์จะประกอบด้วยปุ่มคำสั่งที่ใช้เพื่อแก้ไข source files หรือกำหนดค่าโปรแกรมต่างๆของ μVision
- แถบเครื่องมือการสร้างจะประกอบด้วยปุ่มสำหรับชุดคำสั่งที่ใช้สำหรับสร้างโปรเจค
- แถบเครื่องมือดีบักมีปุ่มสำหรับชุดคำสั่งที่ใช้สำหรับการดีบัก

- **แถบเครื่องมือไฟล์**

	New File – opens an empty text window
	Open File – dialog to open an existing file
	Save File – saves the contents of the current file
	Save All – saves changes in all open files
	Cut – deletes the selected text and copies it to the clipboard
	Copy – copies the currently selected text to the clipboard
	Paste – inserts text from the clipboard to the current cursor position
	Undo changes – removes prior changes in an edit window
	Redo changes – restores the last change that was undone

	Navigate Backwards – moves cursor to its former backward position
	Navigate Forwards – moves cursor to its former forward position
	Bookmark – sets or removes a bookmark at cursor position
	Previous Bookmark – moves the cursor to the bookmark previous to the current cursor position
	Next Bookmark – moves cursor to the bookmark ahead of the current cursor position
	Clear All Bookmarks – removes bookmarks in the current document
	Indent – moves the lines of the highlighted text one tab stop to the right
	Unindent – moves all highlighted text lines one tab stop to the left
	Set Comment – converts the selected code/text to comment lines
	Remove Comment – converts the selected text lines back to code lines
	Find in Files – searches for text in files; results shown in an extra window
	Find – searches for specified text in current document
	Incremental Find – finds expression as you type
	Debug Session – starts/stops debugging
	Breakpoint – sets or removes a breakpoint at cursor position
	Disable Breakpoint – disables the breakpoint at cursor position
	Disable All Breakpoints – disables all breakpoints in all documents
	Kill All Breakpoints – removes all breakpoints from all documents
	Project Window – dropdown to enable/disable project related windows
	Configure – dialog to configure your editor, shortcuts, keywords, ...

- **แถบเครื่องมือการสร้าง**



	Translate/Compile – compiles or assembles the file in the current edit window
	Build – builds and links those files of the project that have changed or whose dependencies have changed
	Rebuild – re-compiles, re-assembles, and re-links all files of the project
	Batch Build – re-builds the application based on batch instructions. This feature is active in a Multi-Project environment only.
	Stop Build – halts the build process
LOAD	Download – downloads your application to the target system flash
	Target – drop-down box to select your target system (in the Toolbar example above: Simulator)
	Target Options – dialog to define tool and target settings. Set device, target, compiler, linker, assembler, and debug options here. You can also configure your flash device from here.
	File Extensions, Environments, and Books – dialog to configure targets, groups, default folders, file extensions, and additional books
	Manage Multi-Project Workspace – dialog to add or remove individual projects or programs to or from your multi-project container

- **แถบเครื่องมือดีบัก**

RST	Reset – Resets the microcontroller CPU or simulator while debugging

	Run – continues target program execution to next breakpoint
	Stop – halts target program execution
	Step One Line – steps to the next instruction or into procedure calls
	Step Over – steps over a single instruction and over procedure calls
	Step Out – steps out of the current procedure
	Run to Line – runs the program until the current cursor line
	Show Current Statement – Shows next statement to be executed
	Command Window – displays/hides the Command Window
	Disassembly Window – displays/hides the Disassembly Window
	Symbol Window – displays/hides Symbols, Variables, Ports, ...
	Register Window – displays/hides Registers
	Call Stack Window – displays/hides the Call Stack tree
	Watch Window – drop-down to display/hide Locals and Watch Windows
	Memory Window – drop-down to display/hide Memory Windows
	Serial Window – drop-down to display/hide UART-peripheral windows and the Debug printf() View
	Logic Analyzer – displays variable values graphically; Also used as a drop-down to display/hide the Performance Analyzer and Code Coverage Window.
	Performance Analyzer – displays, in graphical form, the time consumed by modules and functions as well as the number of function calls
	Code Coverage – dialog to view code execution statistics in a different way than with the Performance Analyzer
	System Viewer – view the values of your Peripheral Registers
	Instruction Trace – displays/hides the Instruction Trace Window
	Toolbox – shows/hides the Toolbox dialog. Depending on your target system, various options are available.
	Debug Restore Views – drop-down to select the preferred window layout while debugging

3.3 สรุป

ในบทนี้จะเป็นการอธิบายถึงฮาร์ดแวร์และซอฟแวร์ต่างๆที่ใช้ในการทำงานอย่างละเอียดตั้งแต่คุณสมบัติพื้นฐาน, การเชื่อมต่อ, และการตั้งค่าของอุปกรณ์แต่ละตัว การศึกษาฮาร์ดแวร์และซอฟแวร์แต่ละตัวชนิดอย่างละเอียดจะทำให้เราสามารถมองเห็นภาพรวมการทำงานและออกแบบระบบที่มีประสิทธิภาพที่ดีได้

บทที่ 4
วิธีการดำเนินงาน

4.1 กล่าวนำ

ในส่วนของการดำเนินงานทางคณะผู้จัดทำได้แบ่งออกเป็นสามส่วนหลักๆ คือ ส่วนการออกแบบโปรแกรมสื่อสารแบบ I2C ถัดมาคือการออกแบบโปรแกรมส่วนควบคุม แบบ PID และสุดท้ายส่วนของการออกแบบโปรแกรมแสดงผลเป็นกราฟ

4.2 การออกแบบโปรแกรมสื่อสารแบบ I2C

อุปกรณ์ที่ใช้การสื่อสารแบบ I2C ในโครงการนี้มีสองอุปกรณ์คือ DAC MCP4728 และ ADC MCP3424 โดยการออกแบบโปรแกรมนั้นแบ่งออกเป็นสองแบบ คือ การอ่านข้อมูล และการส่งข้อมูล

4.2.1 DAC MCP4728

การออกแบบโปรแกรมจะเป็นลักษณะการเขียนข้อมูลลงไปที่ IC การสื่อสารกับ IC จะต้องมีการส่งข้อมูล และรับข้อมูลดังนี้

- ส่ง Address byte โดย bit สุดท้ายส่ง write bit ไป
- ส่งWRITE COMMAND TYPES และ ส่ง Configuration byte
- ส่งข้อมูลไปที่ IC

ตารางที่ 4.1 การกำหนดค่า Configuration byte

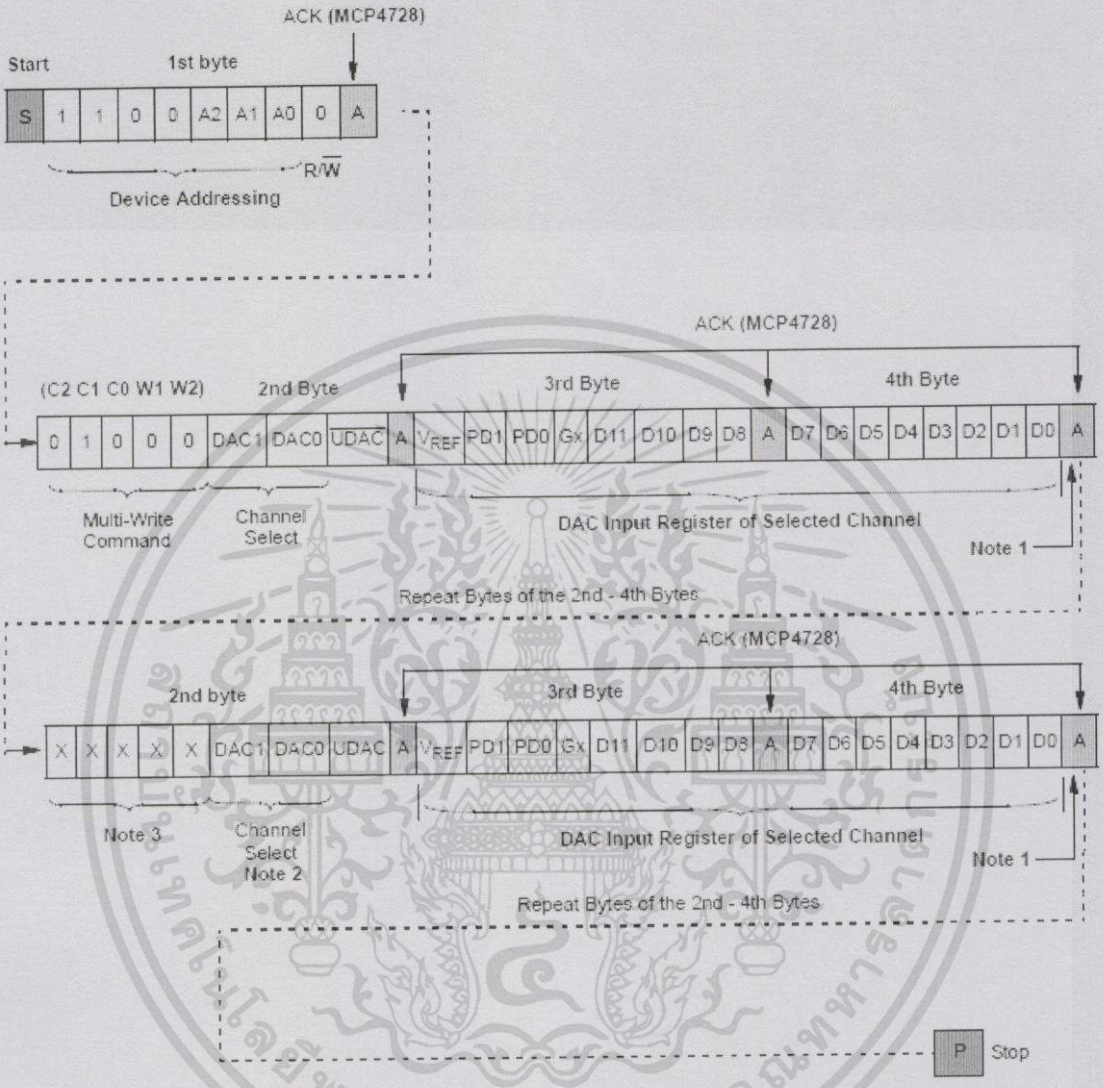
Bit Name	Functions
RDY/BSY	This is a status indicator (flag) of EEPROM programming activity: 1 = EEPROM is not in programming mode 0 = EEPROM is in programming mode **Note:** RDY/BSY status can also be monitored at the RDY/BSY pin.
(A2, A1, A0)	Device I2C address bits. See **Section 5.3 "MCP4728 Device Addressing"** for more details.
VREF	Voltage Reference Selection bit: 0 = VDD

	1 = Internal voltage reference (2.048V) **Note:** Internal voltage reference circuit is turned off if all channels select external reference(VREF = VDD).
DAC1, DAC0	DAC Channel Selection bits: **00** = Channel A **01** = Channel B **10** = Channel C **11** = Channel D
PD1, PD0	Power-Down selection bits: **00** = Normal Mode **01** = VOUT is loaded with 1 kΩ resistor to ground. Most of the channel circuits are powered off. **10** = VOUT is loaded with 100 kΩ resistor to ground. Most of the channel circuits are poweredoff. **11** = VOUT is loaded with 500 kΩ resistor to ground. Most of the channel circuits are poweredoff. **Note:** See Table 4-7 and Figure 4-1 for more details.
GX	Gain selection bit: **0** = x1 (gain of 1) **1** = x2 (gain of 2) **Note:** Applicable only when internal VREF is selected. If VREF = VDD, the device uses a gain of 1 regardless of the gain selection bit setting.
UDAC	DAC latch bit. Upload the selected DAC input register to its output register (VOUT): **0** = Upload. Output (VOUT) is updated. **1** = Do not upload. **Note:** UDAC bit affects the selected channel only.

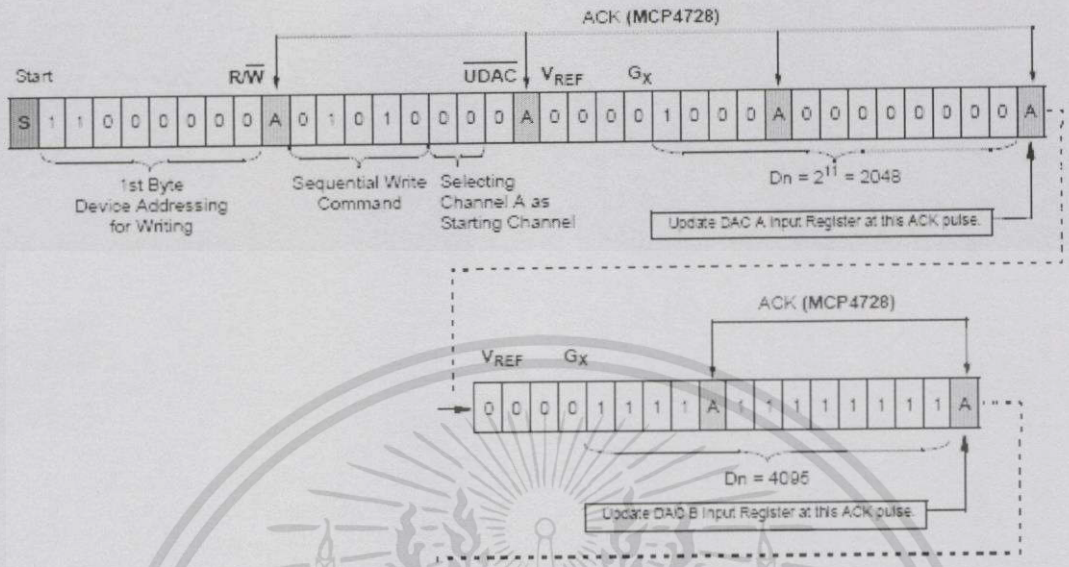
ตารางที่ 4-2 การส่งค่า WRITE COMMAND TYPES

Command Field			Write Function		Command Name	Function
C2	C1	C0	W1	W0		
Fast Mode Write						
0	0	X	Not Used		Fast Write for DAC Input Registers	This command writes to the DAC input registers sequentially with limited configuration bits. The data is sent sequentially from channels A to D. The input register is written at the acknowledge clock pulse of the channel's last input data byte. EEPROM is not affected. **(Note 1)**
Write DAC Input Register and EEPROM						
0	1	0	0	0	Multi-Write for DAC Input Registers	This command writes to multiple DAC input registers, one DAC input register at a time. The writing channel register is defined by the DAC selection bits (DAC1, DAC0). EEPROM is not affected. **(Note 2)**
			1	0	Sequential Write for DAC Input Registers and EEPROM	This command writes to both the DAC input registers and EEPROM sequentially. The sequential writing is carried out from a starting channel to channel D. The starting channel is defined by the DAC selection bits (DAC1 and DAC0). The input register is written at the acknowledge clock pulse of the last input data byte of each register. However, the EEPROM data is written altogether at the

						same time sequentially at the end of the last byte. **(Note 2),(Note 3)**
			1	1	Single Write for DAC Input Register and EEPROM	This command writes to a single selected DAC input register and its EEPROM. Both the input register and EEPROM are written at the acknowledge clock pulse of the last input data byte. The writing channel is defined by the DAC selection bits (DAC1 and DAC0). **(Note 2),(Note 3)**
Write I2C Address Bits (A2, A1, A0)						
0	1	1	Not Used		Write I2C Address Bits	This command writes new I2C address bits (A2, A1, A0) to the DAC input register and EEPROM.
Write VREF, Gain, and Power-Down Select Bits						
1	0	0	Not Used		Write Reference (VREF) selection bits to Input Registers	This command writes Reference (VREF) selection bits of each channel.
1	1	0	Not Used		Write Gain selection bits to Input Registers	This command writes Gain selection bits of each channel.
1	0	1	Not Used		Write Power-Down bits to Input Registers	This command writes Power-Down bits of each channel.



รูปที่ 4.1 แสดง Multi-Write Command: Write Multiple DAC Input Registers.



รูปที่ 4.2 แสดง Sequential Write Command

4.2.2 ADC MCP3424

การออกแบบโปรแกรมจะเป็นลักษณะการอ่านข้อมูลลงจาก IC การสื่อสารกับ IC จะต้องมีการส่งข้อมูลและรับข้อมูลดังนี้

- ส่ง Address byte โดย bit สุดท้ายส่ง write bit ไป
- ส่ง Configuration byte
- ส่ง Address byte โดย bit สุดท้ายส่ง read bit ไป
- รับข้อมูลจาก IC มาเก็บไว้ที่ตัวแปร

การกำหนดค่า Configuration byte สามารถกำหนดได้ดังนี้

bit 7 **RDY:** Ready Bit

This bit is the data ready flag. In read mode, this bit indicates if the output register has been updated with a latest conversion result. In One-Shot Conversion mode, writing this bit to "1" initiates a new conversion.

Reading RDY bit with the read command:

1 = Output register has not been updated

0 = Output register has been updated with the latest conversion result

Writing RDY bit with the write command:

Continuous Conversion mode: No effect

One-Shot Conversion mode:

1 = Initiate a new conversion

0 = No effect

bit 6-5 **C1-C0:** Channel Selection Bits

00 = Select Channel 1 **(ค่าเริ่มต้น)**

01 = Select Channel 2

10 = Select Channel 3

11 = Select Channel 4 (

bit 4 **O/C:** Conversion Mode Bit

1 = Continuous Conversion Mode **(ค่าเริ่มต้น)**. The device performs data conversions continuously

0 = One-Shot Conversion Mode. The device performs a single conversion and enters a low power

standby mode until it receives another write or read command

bit 3-2 **S1-S0:** Sample Rate Selection Bit

00 = 240 SPS (12 bits) **(ค่าเริ่มต้น)**

01 = 60 SPS (14 bits)

10 = 15 SPS (16 bits)

11 = 3.75 SPS (18 bits)

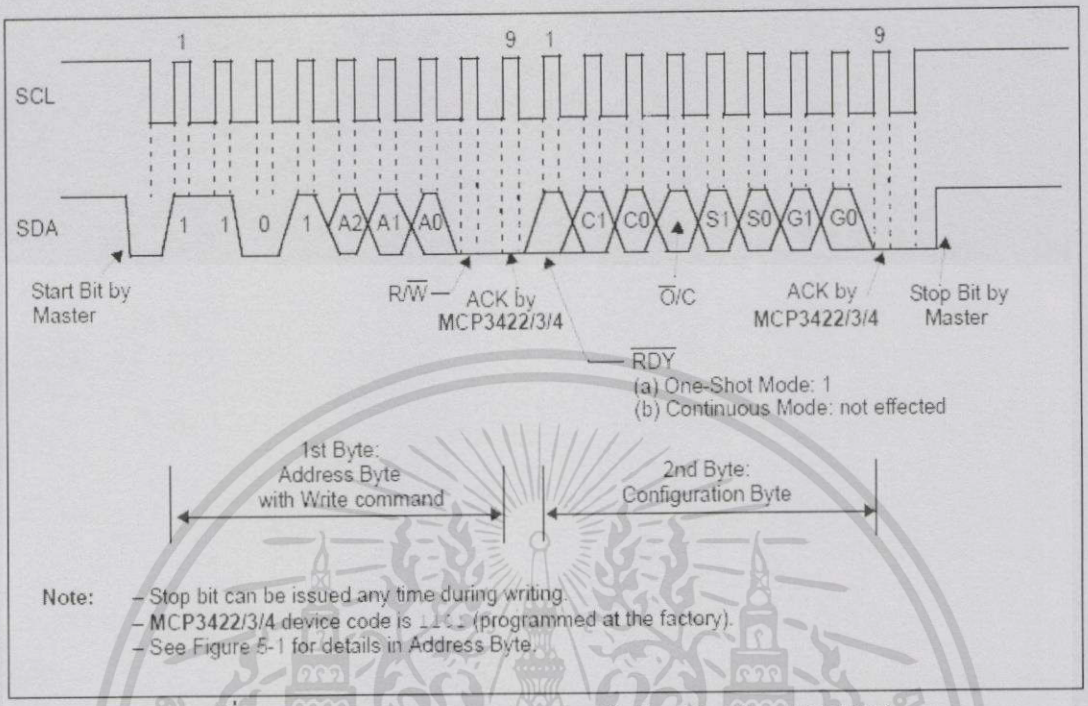
bit 1-0 **G1-G0:** PGA Gain Selection Bits

00 = x1 **(ค่าเริ่มต้น)**

01 = x2

10 = x4

11 = x8



รูปที่ 4.3 แสดง Timing Diagram For Writing To The MCP3424.

4.3 การออกแบบโปรแกรมส่วนควบคุม แบบ PID

ภาพรวม : การควบคุมแบบ PID เป็นวิธีที่ใช้งานอย่างแพร่หลายเพื่อให้ได้และรักษาค่าเป้าหมายของกระบวนการ สมการของการควบคุมแบบ PID แบบสมการมาตรฐานที่

$$Drive = \left(k_p \times error\right) + \left(k_i \times \sum error\right) + \left(k_d \times \frac{dP}{dT}\right) \quad (4.1)$$

เมื่อค่าความผิดพลาดคือค่าความแตกต่างระหว่างค่าตัวแปรของกระบวนการในปัจจุบัน(ความดัน) และค่าเป้าหมายที่เราต้องการโดยปกติค่าค่าความผิดพลาดมักจะเขียนอยู่ในรูป$error = (Value - Setpoint)$ ส่วน $\sum error$ คือ ผลรวมของค่าความผิดพลาดก่อนหน้า $\frac{dP}{dT}$ จะเป็นช่วงเวลาที่เปลี่ยนแปลงไปของค่าตัวแปรกระบวนการในขณะที่ถูกควบคุม ค่าอัตราขยาย k_p, อัตราขยายปริพันธ์ k_i, และอัตราขยายอนุพันธ์ k_d เป็นค่าสัมประสิทธิ์ที่จะปรับแต่งสมการ PID เพื่อทำให้เกิดระบบที่เหมาะสมในขณะที่ถูกควบคุม $Drive$ คือค่าสัญญาณในการควบคุม(กระแส) ที่จะนำไปใช้กับตัวควบคุม(Control Valve) เพื่อให้ได้และรักษาค่าเป้าหมาย

วิธีการปรับแต่ง : ไม่ใช่ว่าทุกส่วนในสมการ *PID* นั้นจะสำคัญต่อการใช้งานทั้งหมด เราต้องศึกษาและตัดสินใจว่าส่วนไหนที่เกี่ยวข้องหรืออัตราขยายตัวไหนควรจะใช้ในสมการของเรา แม้ว่าจะมีวิธีในการปรับแต่งค่า *PID* มากมาย แต่ค่าอัตราขยายก็มักจะเข้าหาในลักษณะเฉพาะสำหรับระบบเล็กๆ เราสามารถปรับแต่งจากการสังเกตการตอบสนองในปัจจุบันของกระบวนการต่อกลุ่มของการเปลี่ยนแปลงในลักษณะเฉพาะนี้จนกว่าระบบจะมีการควบคุมที่ดีได้

โปรแกรมของการควบคุมแบบ *PID*:

```
PID:
Error = Setpoint - Actual
Integral = Integral + (Error*dt)
Derivative = (Error - Previous_error)/dt
Drive = (Error*kP) + (Integral*kI) + (Derivative*kD)
Previous_error = Error
wait(dt)
GOTO PID
```

โค๊ดด้านบนนี้ไม่ได้ถูกเขียนมาสำหรับภาษาคอมพิวเตอร์หรือไมโครคอนโทรเลอร์โดยเฉพาะแต่ออกแบบมาเพื่อแสดงให้เห็นพื้นฐานการวิเคราะห์เพื่อใช้งานการควบคุมแบบ *PID* และโค๊ดนี้ยังได้เอารายละเอียดบางส่วนที่มักจะใช้ออกไปด้วย

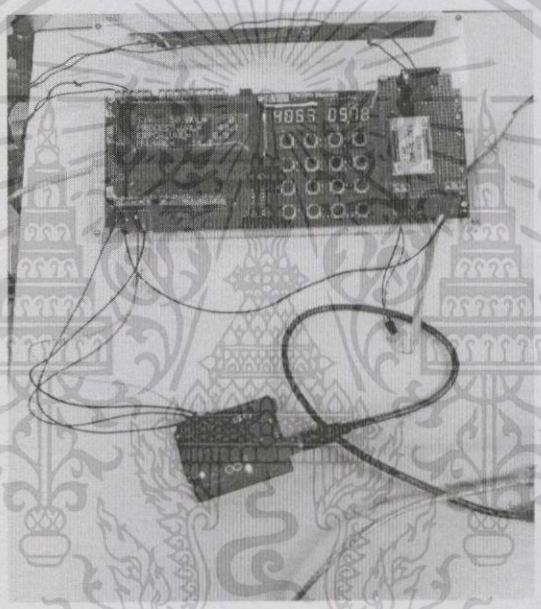
โปรแกรมด้านล่างนี้ถูกเขียนรูปของภาษาซี ซึ่งเราสามารถใช้เป็นแนวทางในการพัฒนาการควบคุม *PID* ของเราได้

```
Actual = analogRead(Position);
Error = SetPt - Actual;
P = Error*kP;                          // calc proportional term
I = Integral*kI;                        // integral term
D = (Last-Actual)*kD;                  // derivative term
Drive = P + I + D;                     // Total drive = P+I+D
Drive = Drive*ScaleFactor;             // scale Drive to be in the range 0-
255
Last = Actual;                         // save current value for next time
```

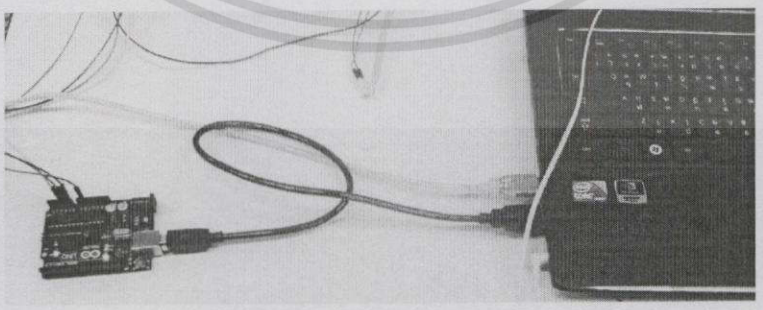
4.4 การออกแบบโปรแกรมแสดงผลเป็นกราฟ

ในด้านของการออกแบบโปรแกรมแสดงผลให้เป็นกราฟ เราต้องการแสดงค่าออกมาที่โปรแกรม LabView โดยการเชื่อมต่อผ่าน arduino มีขั้นตอนดังนี้

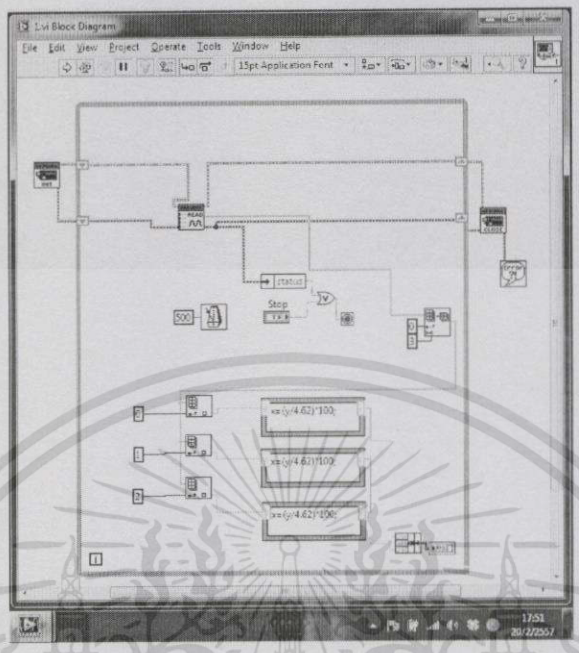
- ส่งค่าที่ต้องการแสดงเป็นกราฟจาก STM32 ไปที่ arduino
- เชื่อมต่อ arduino เข้า computer โดยผ่านสาย USB
- ออกแบบโปรแกรมแสดงกราฟ ใน arduino



รูปที่ 4.4 แสดงการเชื่อมต่อ STM32 กับ arduino



รูปที่ 4.5 แสดงการเชื่อมต่อ arduino กับ computer



รูปที่ 4.6 แสดงการเขียนโปรแกรม LabView

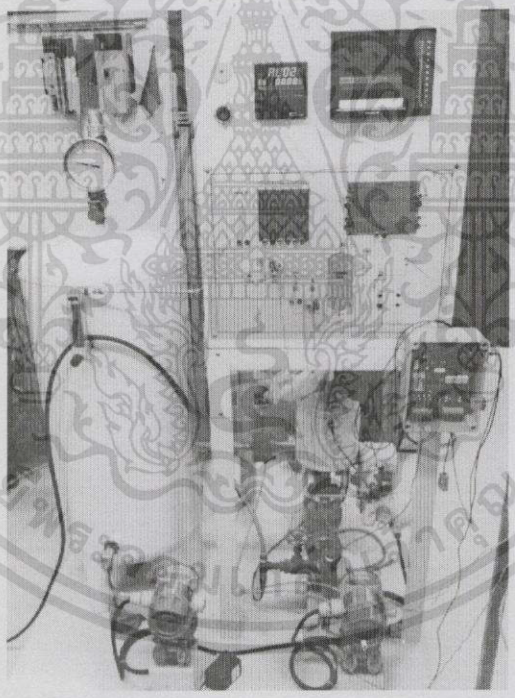
4.5 สรุป

จากนี้หาในบทนี้บรรยาถึงวิธีการดำเนินงานในการออกแบบโปรแกรมทุกส่วนของโครงงานนี้ แบ่งเป็น 3 ส่วน ดังนี้

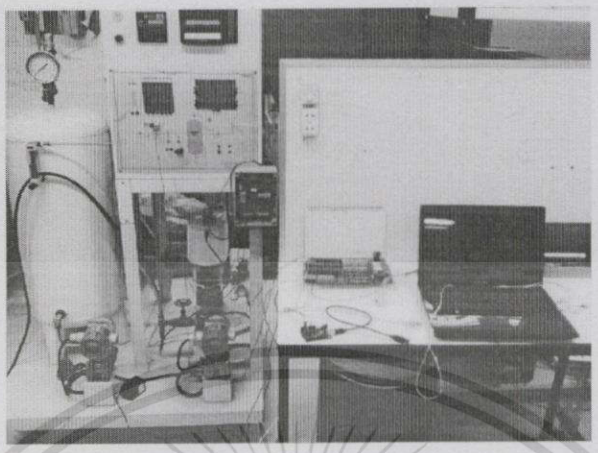
- ส่วนอุปกรณ์แปลงสัญญาณ A to D และ D to A เป็นส่วนของการออกแบบโปรแกรมการสื่อสาร I2C
- ส่วนควบคุม PID
- ส่วนแสดงผลที่เป็นกราฟ ออกแบบโดยใช้โปรแกรม LabView

บทที่ 5
การทดลองและผลการทดลอง

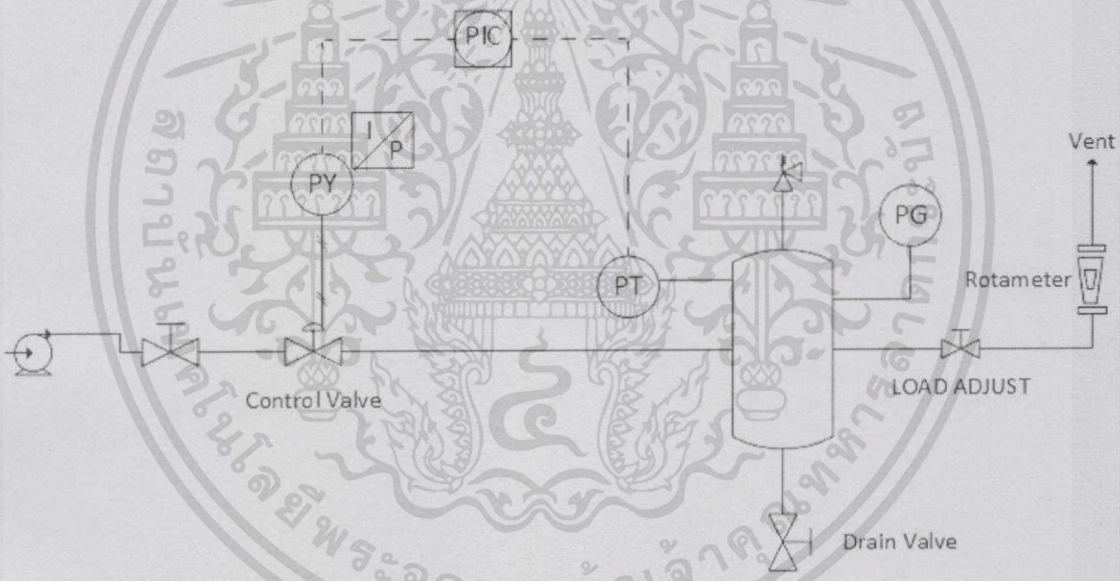
ในขั้นตอนการทดลองเครื่องควบคุม PID โดย STM 32 ทางคณะผู้จัดทำเลือกทดสอบเครื่องควบคุมกับระบบควบคุมความดัน ที่มี Pressure Tank, Control Valve และ D/P transmitter เป็นอุปกรณ์ การทดลองจะมีการใส่ค่า k_p, k_i และ k_d เข้าไปที่ไมโครคอนโทรเลอร์ STM 32 การแสดงผลนั้นจะแยกเป็นสองส่วนคือ ส่วนแสดงผม ค่า Setpoint, k_p, k_i, k_d, MV และ PV จะแสดงค่าอยู่ LCD 4x20 และอีกส่วนคือส่วนแสดงผลที่เป็นกราฟผลตอบสนอง จะแสดงผลที่จอคอมพิวเตอร์โดยใช้โปรแกรม LabView โดยใช้ไมโครคอนโทรเลอร์ Arduino เชื่อมต่อกับ ไมโครคอนโทรเลอร์ STM 32



รูปที่ 5.1 แสดงกระบวนการควบคุมความดัน

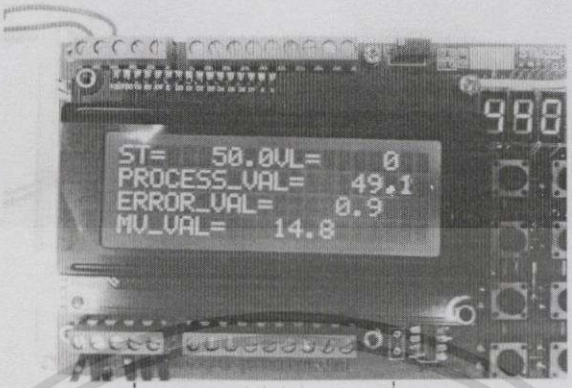


รูปที่ 5.2 แสดงการต่ออุปกรณ์เพื่อทดสอบเครื่องควบคุม PID



รูปที่ 5.3 แสดงการออกแบบกระบวนการควบคุมความดัน

จากรูปที่ 5-3 การทำงานของระบบในกระบวนการจะเริ่มเมื่อปั๊มลม ปั๊มลมเข้ามาที่ Control Valve และเมื่อมีคำสั่งเปิด Control Valve จะมีความดันลมเข้าไปที่ Pressure Tank โดยไมโครคอนโทรเลอร์ STM 32 จะเป็นตัวสั่งควบคุมระดับการ เปิด/ปิด Control Valve มีการกำหนดค่า Set Point ให้กับไมโครคอนโทรเลอร์ STM 32 และ ไมโครคอนโทรเลอร์ STM 32 จะรับค่า PV มาจาก D/P transmitter ไมโครคอนโทรเลอร์ STM 32 ทำการประมวลผลตามโปรแกรมออกแบบไว้ สัญญาณที่ออกไปที่ควบคุม Control Valve คือ 4-20 mA สัญญาณที่รับมาจาก D/P transmitter คือ 1-5 V



รูปที่ 5.4 แสดงการแสดงผลที่จอ LCD

รูปที่ 5.5 แสดงการแสดงผลที่จอคอมพิวเตอร์ ผ่านโปรแกรม LabView

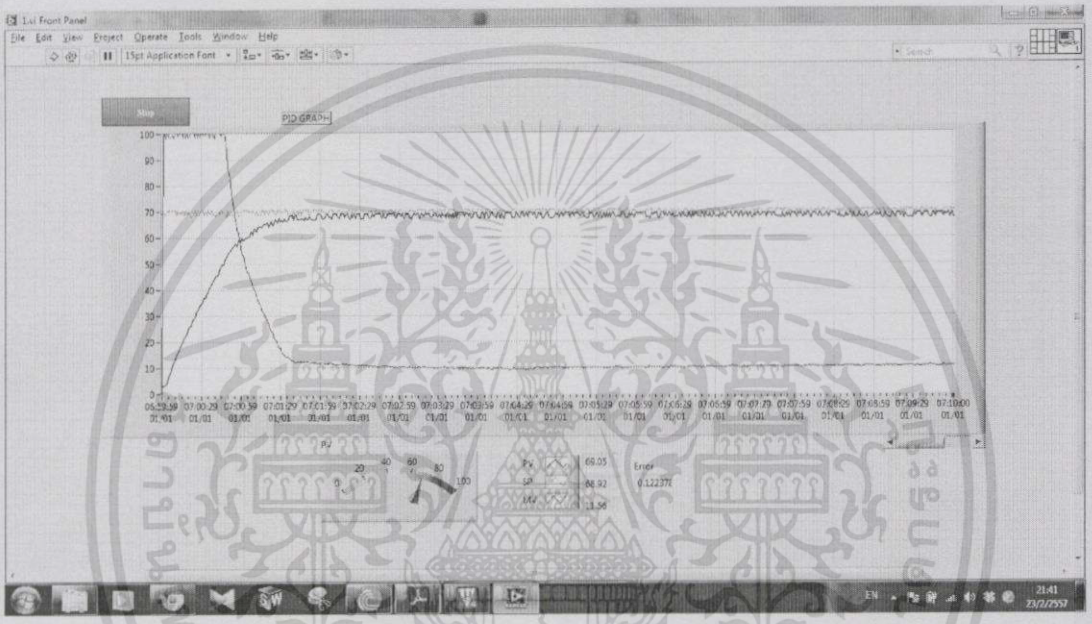
5.1 การทดลองควบคุมความดันใน Tank

การควบคุมความดันใน Tank โดยใช้วิธีวงรอบปิด

คือ การที่เรากำหนดค่าเป้าหมาย (SV) ให้กับไมโครคอนโทรเลอร์ จากนั้นไมโครคอนโทรลเลอร์จะประมวลตามโปรแกรมที่ผู้จัดทำได้เขียนไว้โดยรับค่า k_p, k_i, k_d จาก KeyPad จากผู้ควบคุมอีกทีหนึ่ง แล้วส่งสัญญาณ (4-20mA) ไปควบคุมวาล์วเพื่อเปิดวาล์วให้ความดันลมเข้าไปใน Tank แล้ววัดความดันใน Tank โดย D/P transmitter แล้วส่งค่า (1-5V) กลับมาที่ไมโครคอนโทรเลอร์เพื่อประมวลผล

5.2 กำหนดค่าพารามิเตอร์ และได้ผลการทดลองดังนี้

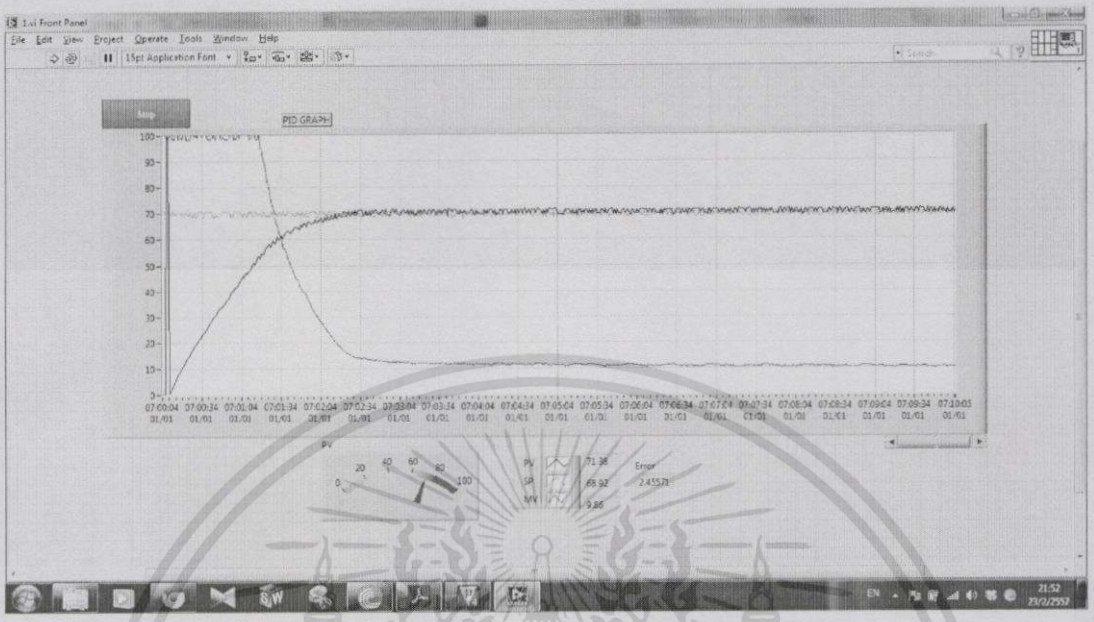
- ความดันลม 5 bar
- ปรับ Pressure vent อยู่ที่ 3000 l/h
- กำหนดค่า Setpoint=70



รูปที่ 5.6 กราฟแสดงผลตอบสนองของกระบวนการควบคุม Presssure Tankจากค่า

$$k_p=5000,\ k_i=0,\ k_d=0$$

การทดลองที่ตั้งค่าเป้าหมายไว้ที่ 70 เปอร์เซ็นและใช้อัตราขยายสัดส่วนที่ 5000 พบว่าค่าตัวแปรกระบวนการมีค่าเข้าใกล้เป้าหมายมากขึ้นเรื่อยๆ แต่ไม่สารมารถเข้าไปที่ค่าเป้าหมายได้ โดยมีค่า offset ประมาณ 1.2

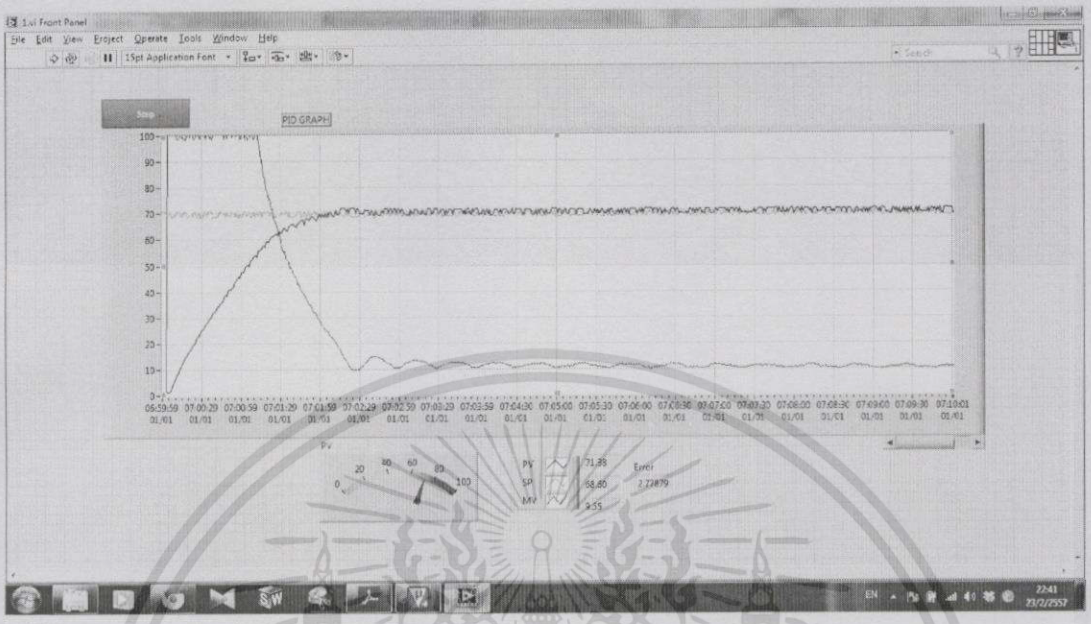


รูปที่ 5.7 กราฟแสดงผลตอบสนองของกระบวนการควบคุม Presssure Tankจากค่า

k_p=5000, k_i=265, k_d=0

การทดลองที่ตั้งค่าเป้าหมายไว้ที่ 70 เปอร์เซ็นและใช้อัตราขยายที่ 5000 และอัตราขยายปริพันธ์ที่265 พบว่า ค่าตัวแปรกระบวนการมีค่าเข้าใกล้เป้าหมายมากขึ้นเรื่อยๆ และเข้าไปที่ค่าเป้าหมายได้ โดยมีโดยมีค่าอยู่ช่วง 0.3% ของค่าที่สถาวะคงที่และมีค่าอยู่ที่ช่วงนี้ตลอด

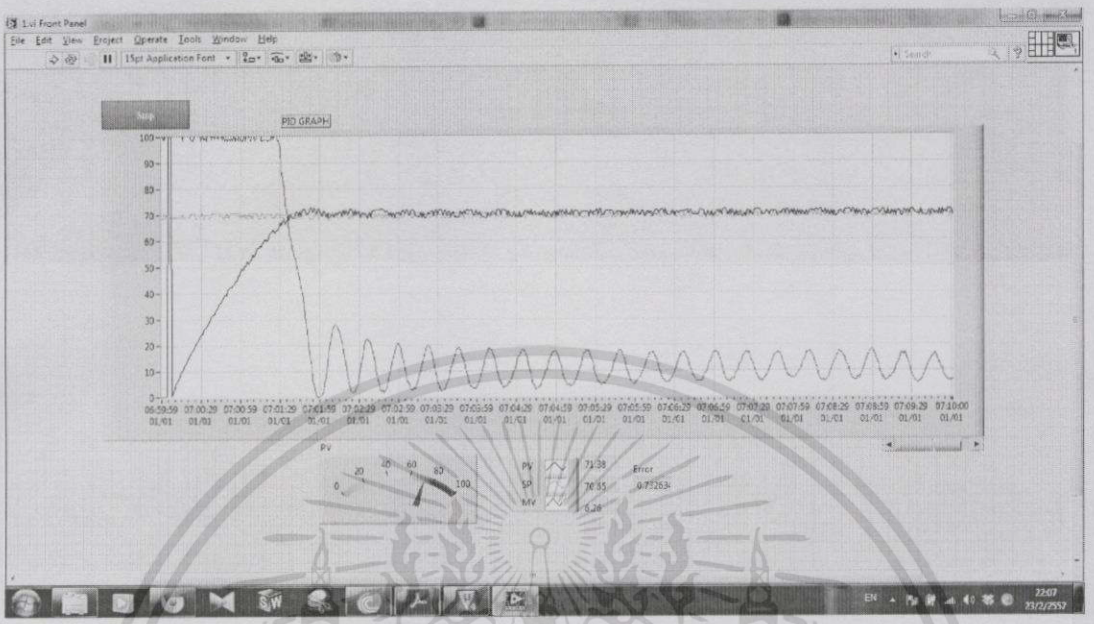
Delay time	Rise time	overshoot	Settling time
25s	45s	0%	80s



รูปที่ 5.8 กราฟแสดงผลตอบสนองของกระบวนการควบคุม Presssure Tankจากค่า

$$k_p=5000,\ k_i=400,\ k_d=0$$

การทดลองที่ตั้งค่าเป้าหมายไว้ที่ 70 เปอร์เซ็นและใช้อัตราขยายที่ 5000 และปรับอัตราขายของปริพีนธ์ให้มากกว่าปกติโดยปรับอัตราขยายปริพันธ์ที่ 400 พบว่าค่าตัวแปรกระบวนการมีค่าเข้าใกล้เป้าหมายมากขึ้นเรื่อยๆ จะเห็นว่าค่าตัวแปรกระบวนการมีการแกว่งอยู่เล็กน้อย สังเกตจากการแกว่งของแต่ค่า output ของตัวควบคุม(MV)



รูปที่ 5.9 กราฟแสดงผลตอบสนองของกระบวนการควบคุม Presssure Tankจากค่า

$$k_p=10000,\ k_i=1000,\ k_d=0$$

การทดลองที่ตั้งค่าเป้าหมายไว้ที่ 70 เปอร์เซ็น โดยปรับอัตราขยายสัดส่วนและอัตราขายของปริพันธ์ให้มากกว่าปกติ ใช้อัตราขยายที่ 1000 และปรับอัตราขยายปริพันธ์ที่ 1000 พบว่าค่าตัวแปรกระบวนการมีเห็นว่าค่าตัวแปรกระบวนการมีการแกว่งอยู่ตลอดเวลา สังเกตจากการแกว่งของแต่ค่า output ของตัวควบคุม(MV) ที่แกว่งไปเรื่อยๆ ไม่หยุด

5.3 สรุป

เนื้อหาในบทนี้ได้อธิบายถึงการทดลองและผลการทดลองและการหาค่าพารามิเตอร์ในการควบคุมกระบวนการ โดยการปรับค่าพารามิเตอร์แบบ Manual ซึ่งการใช้วิธีนี้เป็นวิธีที่ต้องใช้เวลา, ความรู้และประสบการณ์ และเมื่อปรับค่าที่เหมาะสมได้ จะทำให้ได้ผลตอบสนองที่น่าพอใจ

บทที่ 6
สรุปผลการวิจัยและข้อเสนอแนะ

6.1 สรุปผลการทำงานวิจัย

จากการทดลองเพื่อทดสอบ Pressure Process โดยใช้เครื่องควบคุม PID เป็นตัวควบคุมพบว่าเครื่องควบคุมสามารถควบคุมระบบได้เป็นที่น่าพอใจสามารถทำให้กระบวนการเข้าสู่ค่าเป้าหมายได้โดยใช้เวลาไม่มาก ในส่วนการควบคุมเราจะใช้ไมโครคอนโทรลเลอร์ STM32 เป็นตัวควบคุมและแสดงผลออกมาทั้งทางจอ LCD และในรูปแบบกราฟโดยผ่านทางไมโครคอนโทรลเลอร์ Arduino เป็นตัว interface กับโปรแกรม Labview เพื่อแสดงกราฟ

6.1.2 ปัญหาและอุปสรรค

1. การสั่งงานจากไมโครคอนโทรลเลอร์ไปยังวาล์วควบคุมในบางครั้งเกิดความล่าช้า ทำให้เกิดการคำนวณที่ผิดพลาด
2. กราฟแสดงผลตอบสนองค่าที่เปลี่ยนแปลงอย่างรวดเร็วของกระบวนการได้ไม่ดีเพียงพอ ทำให้รูปทรงของกราฟไม่เป็นไปตามที่ต้องการ
3. การส่งสัญญาณระหว่างไมโครคอนโทรลเลอร์ STM32 กับไมโครคอนโทรลเลอร์ Arduino จำเป็นจะต้องแปลงค่าข้อมูลในลักษณะของกราฟเส้นตรงก่อนที่จะทำการส่งสัญญาณและหลังจากรับสัญญาณ ทำให้เกิดค่าความผิดพลาดขึ้น

6.2 ข้อเสนอแนะ

การออกแบบเครื่องควบคุม PID โดยใช้ไมโครคอนโทรลเลอร์จำเป็นที่จะต้องศึกษาข้อมูลของไมโครคอนโทรลเลอร์, โปรแกรมที่ใช้ทำงานร่วมกับไมโครคอนโทรลเลอร์, และส่วนแสดงผลต่างๆอย่างเข้าใจ เพื่อที่จะได้ไม่ต้องนำไมโครคอนโทรลเลอร์หรือโปรแกรมอื่นๆมาเชื่อมต่อเพิ่มซึ่งจะทำให้เกิดค่าความผิดพลาดในระบบมากขึ้น

บรรณานุกรม

[1]รศ.อาจินต์ น่วมสำราญ. 2556. **Virtual Instrumentation using LabVIEW การวัดคุมเสมือนด้วยโปรแกรม LabVIEW.** พิมพ์ครั้งที่ 1. กรุงเทพฯ : ภาควิชาวิศวกรรมการวัดคุม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

[2]อธิภู สันติภูมิโพธ, อภินันท์ จันทิมา, อรรถกร เกิดกิจ. 2554. **การออกแบบตัวควบคุมพัซซีลอจิกสำหรับกระบวนการระดับน้ำ.** วิทยานิพนธ์ปริญญาวิศวกรรมศาสตร์บัณฑิต. กรุงเทพฯ : ภาควิชาวิศวกรรมการวัดคุม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

[3]ปภณพรรธน์ ศิรกิตติวรพงษ์, วัสพล กุลชัยกุล, วัชรี สอดศรี. 2555. **การควบคุมกระบวนการด้วยตัวควบคุมแบบ PID โดยใช้ LabVIEW.** วิทยานิพนธ์ปริญญาวิศวกรรมศาสตร์บัณฑิต. กรุงเทพฯ : ภาควิชาวิศวกรรมการวัดคุม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

[4]http://home.npru.ac.th/piya/webscilab/file/Scilab-LabVIEW-Gateway.pdf

[5]http://en.wikipedia.org/wiki/PID_controller

[6]http://www.ee.kmutnb.ac.th/eerobot/esl/learning/index.php?article=lpc2148-keil-sourcery&start=20

[7]https://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en541737#documentation

[8]http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en536354

[9]http://www.maelabs.ucsd.edu/mae156alib/control/PID-Control-Ardunio.pdf