เครื่องดันเข็มฉีดยา

SYRINGE PUMP

โดย

นายชาญฤทธิ์ ยศสันติกุล

นายวาที รณรงค์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

เครื่องดันเข็มฉีดยา

SYRINGE PUMP

โดย

นาย ชาญฤทธิ์ ยศสันติกุล

นาย วาที รณรงค์

อาจารย์ที่ปรึกษา

อ.ดร. กิติพล ชิตสกุล

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ ปีการศึกษา 2541

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องดันเข็มฉีดยา

SYRINGE PUMP

ผู้จัดทำ

1. นายชาญฤทธิ์ ยศสันติกุล เลขประจำตัว 39013195
2. นายวาที รณรงค์ เลขประจำตัว 39013206

..... อาจารย์ที่ปรึกษา

(ดร.กิติพล ชิตสกุล)

วันที่ 19 มีนาคม พ.ศ. 2542

เครื่องดันเข็มฉีดยา

นายชาญฤทธิ์ ยศสันติกุล
นายวาที รณรงค์
อ.ดร. กิติพล ชิตสกุล (อาจารย์ที่ปรึกษา)
ปีการศึกษา 2541

บทคัดย่อ

เครื่องดันเข็มฉีดยาเป็นเครื่องมือที่นำมาใช้งานมากมายทั้งในทางการแพทย์ ทางวิทยาศาสตร์ และในงานอุตสาหกรรมที่ต้องการควบคุมการไหลหรือปริมาณของของเหลวได้อย่างต่อเนื่อง ปริญญานิพนธ์ฉบับนี้ ได้กล่าวถึงการออกแบบสร้างต้นแบบเครื่องดันเข็มฉีดยา ควบคุมด้วยไมโครคอนโทรลเลอร์ที่สามารถควบคุมปริมาณ ความนิ่มนวลของการฉีด ที่กำหนดให้อัตราการไหลต่ำและคงที่ นอกจากนี้ได้ออกแบบให้ การใช้งานของเครื่องต้นแบบนี้ใช้งานได้ง่าย เครื่องต้นแบบนี้ ก็มีข้อจำกัดคือ สามารถใช้เฉพาะกับหลอดฉีดยาขนาด 50 ลูกบาศก์เซ็นติเมตรได้เพียงขนาดเดียวเท่านั้น และในเบื้องต้นสามารถควบคุมอัตราไหลได้ตั้งแต่ 1 ลูกบาศก์เซ็นติเมตรต่อนาที ถึง 3.8 ลูกบาศก์เซ็นติเมตรต่อวินาที ความแม่นยำของปริมาณการไหลเฉลี่ย 2%

SYRINGE PUMP

Chanrit yossontikul

Watee ronnarong

Dr. Kitiphol chitsakul (Advisor)

Academic year 1998

Abstract

Syringe pump are widely used in medicine, scientific laboratories and industries where need to continuously control the flow or quantity of a fluid. A design and construction of a prototype of microcontroller based syringe pump is described in this thesis. The objective of this work is to obtain a syringe pump which provides smooth delivery and consistency of flow at low flow rate. It is designed to be a user friendly device. However the prototype has been limited to used with only one size of syringes (50c.c.). The flow rate can be set between 1c.c./min.-3.8c.c./sec. with average accurately of 2%

สารบัญ

สารบัญ (ต่อ)

สารบัญรูป

สารบัญตาราง

บทที่ 1
บทนำ

ในปัจจุบันเครื่องมือเครื่องใช้ต่าง ๆ ที่ผลิตออกมานั้น ได้รับการพัฒนาให้มีสมรรถนะสูงขึ้น การใช้งานก็ง่ายกว่าในอดีต และจากการที่ได้นำเทคโนโลยีสมัยใหม่มาปรับปรุงฟังก์ชันเหล่านี้นั่นเอง ตัวอย่างที่เห็นอย่างได้ชัดเจนได้แก่ การนำเอาไมโครคอนโทรลเลอร์ มาใช้ควบคุมการทำงานของเครื่องมือต่างๆ เนื่องจากไมโครคอนโทรลเลอร์มีการประมวลผลการทำงาน และการควบคุมระบบได้อย่างมี ประสิทธิภาพ สามารถติดต่อสื่อสารกับคอมพิวเตอร์ส่วนบุคคล(PC)ในการวิเคราะร์ผล การเก็บข้อมูลได้ดี แม่นยำ และยังสามารถเขียนซอฟต์แวร์สนับสนุนการทำงานได้เป็นอย่างดี สามารถตรวจสอบข้อผิดพลาดของการทำงานได้ง่ายตลอดจนสามารถขยายระบบได้ในอนาคต

โครงงานนี้เป็นการออกแบบ สร้างต้นแบบเครื่องดันเข็มฉีดยา ควบคุมด้วยไมโครคอนโทรลเลอร์ที่สามารถควบคุมปริมาณ ความนิ่มนวลของการฉีด ที่กำหนดให้อัตราการไหลต่ำและคงที่ นอกจากนี้ยังได้ออกแบบให้ การใช้งานของเครื่องต้นแบบนี้ใช้งานได้ง่าย และสามารถนำไปใช้ในทางการแพทย์ ซึ่งต้องการทั้งปริมาณ และอัตราการไหล (flow-Rate) ที่คงที่สม่ำเสมอ เช่น ใช้ในการให้เด็กอ่อนเพราะในกรณีนี้นั้นต้องการปริมาณอาหารที่สม่ำเสมอเพื่อป้องกันการสำลักของเด็กได้ และเครื่องดันเข็มฉีดยานี้ยังมีความสำคัญมากเกี่ยวกับการฉีดปริมาณของยาให้กับคนไข้ นั่นคือพิษของยาบางชนิดนั้น จะต้องให้ด้วยปริมาณที่คงที่ และสม่ำเสมอ เพื่อที่จะได้ไม่เกิดอันตรายกับคนไข้นั่นเอง นอกจากนั้นยังสามารถนำไปใช้ในงานอุตสาหกรรม เช่นการผสมสารเคมี ผสมสี หรือผสมของเหลว เช่นเครื่องหยอดกาว และอื่นๆได้ เพื่อเพิ่มประสิทธิภาพและคุณภาพของงานให้ดีขึ้นเป็นผลให้มีกำลังการผลิตมากขึ้นเนื่องจากมีของเสียน้อยลง

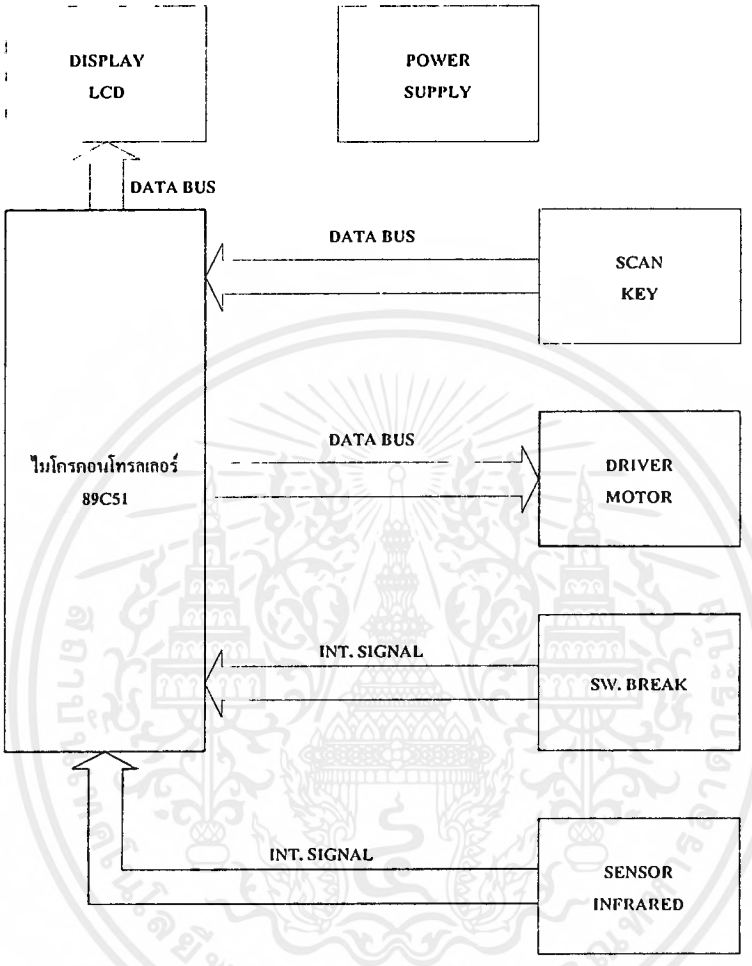
1.1 คุณสมบัติและความสามารถของเครื่องต้นแบบ

เมื่อพิจารณาจากความต้องการต่าง ๆ แล้วจึงกำหนดในเบื้องต้นว่าเครื่องต้นแบบนี้ควรจะต้องมีฟังก์ชันและลักษณะดังนี้คือ

- ควบคุมอัตราการไหลได้สม่ำเสมอคงที่
- ควบคุมปริมาณของของเหลวในหลอดได้
- ควบคุมทั้งปริมาณ และเวลาที่ต้องการให้มีความสัมพันธ์กันได้ เช่น ต้องการปริมาตร ต่อเวลาเท่าไรนั่นเอง
- สามารถสั่งการและแสดงผลที่เครื่องต้นแบบได้เลย

1.2 หลักการและโครงสร้างเบื้องต้นของระบบ

จากฟังก์ชันและลักษณะของเครื่องต้นแบบที่กำหนดไว้ข้างต้น โครงสร้างของเครื่องต้นแบบควรจะเป็นไปตาม บล็อกไดอะแกรม ดังรูปที่ 1.1



รูปที่ 1.1 รูปแสดงบล็อกไดอะแกรมของระบบรวม

จากบล็อกไดอะแกรม จะเห็นว่าระบบรวมทั้งหมด จะประกอบด้วย หน่วยประมวลผล คือ ไมโครคอนโทรลเลอร์ ซึ่งจะมีหน้าที่ในการสั่งการและแสดงผลต่างๆ ซึ่งจะส่งข้อมูลเข้าออกดาต้าบัสไปยังบล็อกต่างๆ คือ รับข้อมูลการกดคีย์จากบล็อกสแกนคีย์ (SCAN KEY) ส่งข้อมูลพัลส์ไปยังบล็อกขับมอเตอร์ (DRIVE MOTOR) และทำการรับสัญญานอิเตอร์รัปต์ (INT. SIGNAL) จากบล็อกสวิทช์เบรก (SW. BREAK) และบล็อกเซนเซอร์อินฟาเรด (SENSOR INFRARED) สุดท้ายไมโครคอนโทรลเลอร์จะส่ง ข้อมูลมาควบคุมบล็อก DRIVE MOTOR เพื่อทำการขับมอเตอร์ต่อไป

1.3 โครงสร้างของปริญญานิพนธ์

ปริญญานิพนธ์เล่มนี้เป็นผลจากการศึกษาและทดลองตลอดสองภาคการศึกษา เพื่อออกแบบสร้างเครื่องดันเข็มฉีดยาควบคุมด้วยไมโครคอนโทรลเลอร์ ซึ่งเนื้อหาของปริญญานิพนธ์เล่มนี้จะประกอบด้วยส่วนต่าง ๆ แยกเป็นบท ๆ ไปดังนี้

บทที่ 1 บทนำ กล่าวถึงความเป็นมาและแนวคิดของโครงงาน

บทที่ 2 ทฤษฎีและหลักการทำงานของไมโครคอนโทรลเลอร์สเต็ปเปอร์และLED อินฟราเรด

บทที่ 3 องค์ประกอบโดยรวมของระบบและการทำงาน

บทที่ 4 การสร้างและการออกแบบโปรแกรมควบคุม

บทที่ 5 การทดลอง และ ผลการทดลอง

บทที่ 6 บทสรุปและวิจารณ์

บทที่ 2

ทฤษฎีและหลักการทำงานของไมโครคอนโทรลเลอร์, สเต็ปเปอร์และLED อินฟราเรด

ชิ้นส่วนสำคัญของโครงงานนี้คือสเต็ปเปอร์มอเตอร์ที่ใช้ในการขับเคลื่อนทางกล ดังนั้นในบทนี้จะกล่าวโดยสังเขปถึงชิ้นส่วนนี้ทั้งในส่วนโครงสร้างและการขับเคลื่อนและควบคุมที่ได้ศึกษามา นอกจากนี้จะกล่าวถึงอุปกรณ์ทางแสงที่นำมาใช้ในการตรวจจับตำแหน่ง รวมทั้งหลักการทำงานของไมโครคอนโทรลเลอร์

2.1 สเต็ปเปอร์มอเตอร์และการขับเคลื่อน

สเต็ปเปอร์มอเตอร์มีความแตกต่างจากมอเตอร์ทั่ว ๆ ไปคือเมื่อได้รับกำลังไฟฟ้าจะหมุนเพียงเล็กน้อยตามเส้นรอบวงและหยุด แต่สเต็ปเปอร์มอเตอร์จะหมุนด้วยตัวเลข(ฐานสอง)ได้อย่างละเอียดโดยการใช้คอมพิวเตอร์เป็นตัวกำหนดและจัดเก็บตัวเลขเหล่านั้นไว้

สเต็ปเปอร์มอเตอร์สามารถใช้งานในระบบเปิด (Open loop system) นั่นก็คือทำงานได้โดยไม่ต้องการการป้อนกลับ (Feedback) แต่อย่าไรก็ดีหากต้องการกำหนดตำแหน่งในรอบการหมุนได้อย่างถูกต้องจำเป็นที่ต้องใช้การป้อนกลับไปยังระบบขับและควบคุมให้รับรู้ ปัญหาก็คืออะไรจะใช้เป็นตัวบอกได้ว่าแกนหมุนมาอยู่ในตำแหน่งที่ถูกต้องแล้วหรือเกิดการผิดพลาดขึ้น (error)

วิธีหนึ่งใช้กันโดยทั่วไปกับสเต็ปเปอร์มอเตอร์ก็คือ การใช้สวิตช์ติดตั้งไว้ที่ตำแหน่งที่ต้องการตรวจจับ (limit switch) เมื่อสเต็ปเปอร์มอเตอร์เริ่มหมุนและหมุนจนกระทั่งถึงตำแหน่งของสวิตช์ตรวจจับ สัญญาณก็จะถูกป้อนกลับเข้าสู่ระบบและทราบการทำงานของสเต็ปเปอร์มอเตอร์ได้ตลอดเวลา ซึ่งโดยปกติในวงจรคอนโทรลเลอร์จะมีการกำหนดจุดอ้างอิงตำแหน่งได้อย่างถูกต้อง

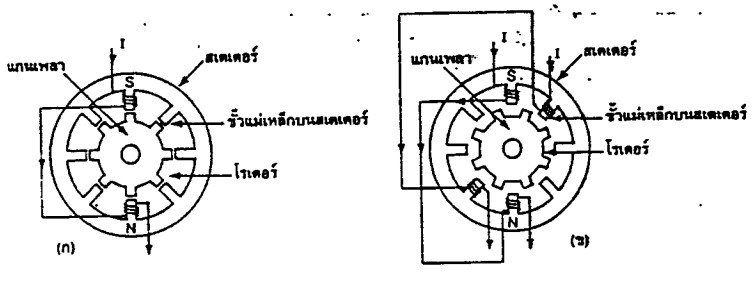
เช่นเดียวกับมอเตอร์ทั่วไป การหมุนของโรเตอร์ (rotor) เกิดจากแรงกระทำของสนามแม่เหล็ก ไฟฟ้าที่เกิดขึ้นระหว่างโรเตอร์และสเตเตอร์ (stator) ซึ่งขึ้นอยู่กับการจัดวางขั้วแม่เหล็ก (pole) การหมุนทำได้ทั้งแบบต่อเนื่องและกลับทิศทางไปมา โดยกระบวนการทางไฟสลับหรือการจัดวางแปรงถ่าน และการจัดแยกคอมมิวเตเตอร์ และทำการสวิตช์ซิ่งกำลังไฟฟ้าให้เกิดแรงดึงดูดของแม่เหล็ก (magnetic attraction) ที่ขั้วแม่เหล็กสร้างและหยุดสลับกัน ผลก็คือเกิด สนามแม่เหล็กหมุนขึ้นบน สเตเตอร์โดยการจ่ายกำลังไฟฟ้าทีละคู่ของขั้วแม่เหล็กในทิศทางตรงกันข้ามไปตลอดเวลา และเมื่อต้องการหยุดหมุนสามารถทำได้โดยหยุดการเกิดขั้วแม่เหล็กที่จุดหนึ่งโดยหยุดการสวิตช์ซิ่งในลำดับต่อไปเสีย การหมุนกลับทิศทางก็ทำได้เช่นเดียวกับที่กล่าวมาแล้ว เพียงแต่ทำการสวิตช์ซิ่งกำลังไฟฟ้าให้เกิดสนามแม่เหล็กหมุนในทิศทางกลับกัน หรือกลับลำดับการสวิตช์ของมัน

โครงสร้างของขั้วแม่เหล็กบนสเตเตอร์ ประกอบขึ้นจากแผ่นเหล็กวงแหวน ที่มีซี่ยื่นออกมา แต่ละซี่ที่ยื่นออกมานั้น จะมีคอยล์พันสวมอยู่ ดังนั้นเมื่อป้อนกระแสไฟฟ้าผ่านคอยล์ทำให้เกิดสนามแม่เหล็กไฟฟ้าขึ้น ด้านตรงข้ามของแต่ละขั้วแม่เหล็ก จะได้รับกระแสไฟฟ้าในขณะเดียวกัน แต่ว่าจะไหลวนในทิศทางตรงกันข้าม ทำให้เกิดสนามแม่เหล็กไฟฟ้าในทิศตรงข้ามขึ้น ดังแสดงในรูปที่ 2.1 (ก) ดังนั้นถ้าเพิ่มจำนวนของขั้วแม่เหล็กมากขึ้นจะเพิ่มจำนวนของสเต็ปต่อวงรอบมากขึ้นตามไปด้วย

อย่างไรก็ตามผู้ใช้งานสามารถเพิ่มจำนวนของสเต็ปได้อีกวิธีหนึ่งโดยไม่ต้องปรับเปลี่ยนโครงสร้างภายใน โดยทำการจ่ายกำลังไฟฟ้าไปยังขั้วแม่เหล็ก 2 ขั้วที่อยู่ใกล้กันในเวลาเดียวกัน ซึ่งจะทำให้ โรเตอร์หยุดหมุนอยู่ระหว่างกลางของ 2 ขั้วแม่เหล็กนั้นหรือเคลื่อนที่ไปครึ่งสเต็ปเท่านั้น และวิธีการนี้ยังช่วยให้เกิดแรงบิด (torque) มากขึ้นด้วย ดังแสดงในรูที่ 2.1 (ข)

สเต็ปเปอร์มอเตอร์โดยทั่วไปมีจำนวนของขั้วแม่เหล็กหรือจำนวนสเต็ปต่อรอบเป็นจำนวนของขั้วแม่เหล็กหรือจำนวนสเต็ปต่อรอบเป็นจำนวนมาก ปกติอยู่ที่ประมาณ 100-400 สเต็ปต่อรอบ การมีจำนวนสเต็ปมาก ๆ นี้ไม่ได้เพิ่มที่จำนวนขั้วแม่เหล็กไฟฟ้าที่สเตเตอร์ แต่ทำได้โดยเพิ่มจำนวนซี่ขั้ว แม่เหล็กที่โรเตอร์จำนวนสเต็ปต่อรอบทั้งหมด โดยจะได้จากการคูณจำนวนขั้วแม่เหล็กบนสเตเตอร์ และจำนวน ซี่ที่โรเตอร์ ดังเช่น ถ้ามีขั้วแม่เหล็ก 3 ขั้วบนสเตเตอร์ และ 8 ซี่ขั้วแม่เหล็กบน โรเตอร์สเต็เปอร์มอเตอร์ ตัวนี้จะทำงานที่ 24 สเต็ปต่อรอบ หรือหมุนเป็น 15 องศาต่อสเต็ป เป็นต้น

การใช้วงจรดิจิตอลคอนโทรลเลอร์กำหนดการจ่ายกำลังไฟฟ้า เข้าสู่ขดลวดบนสเตเตอร์แบบ ซีเควนเชียล (sequential) ทำให้สามารถควบคุมการเคลื่อนที่ทุกสเต็ปได้ เช่นเดียวกับการควบคุมใน วงจรดีซีเซอร์โว (DC servo) แต่การควบคุมด้วยดิจิตอลไม่จำเป็นต้องมีการป้อนกลับการเคลื่อนที่ทุกสเต็ปได้จากการคำนวณจำนวนรอบหรือมุมในการหมุนที่ต้องการ แล้วจึงส่งข้อมูลที่ได้ไปควบคุมการหมุนของมอเตอร์ พิกัดในการทำงานอาทิความเร็ว มุมในการเคลื่อนที่ ตำแหน่งของเพลาถูกกำหนดจากข้อมูลที่ส่งมาควบคุม



รูปที่ 2.1 (ก) แสดงสเต็ปเปอร์มอเตอร์ที่มีการต่อวงจรขดลวดภายใน เพื่อกระตุ้นให้เกิดขั้วแม่เหล็กขึ้น 1 ขั้ว ในทิศทางตรงกันขาม ส่วนขดลวดอื่นๆ จะไม่ถูกกระตุ้นเลย (ข) แสดงการต่อ

วงจร ขดลวดแบบกระตุ้นให้เกิดขั้วแม่เหล็กพร้อมกัน 2 ขั้วที่อยู่ใกล้กัน ทำให้โรเตอร์เคลื่อนที่มาหยุดอยู่ระหว่างขั้วแม่เหล็กทั้งสอง

2.1.1 ชนิดของสเต็ปเปอร์มอเตอร์

สเต็ปเปอร์มอเตอร์แบ่งตามพื้นฐานได้เป็น 3 ชนิดคือ วาริเอเบิลรีลักแดนซ์ (variable reluctance: VR) เพอร์มาเนนต์ แมกเน็ต (permanent magnet: PM) และแบบไฮบริด (hybrid)

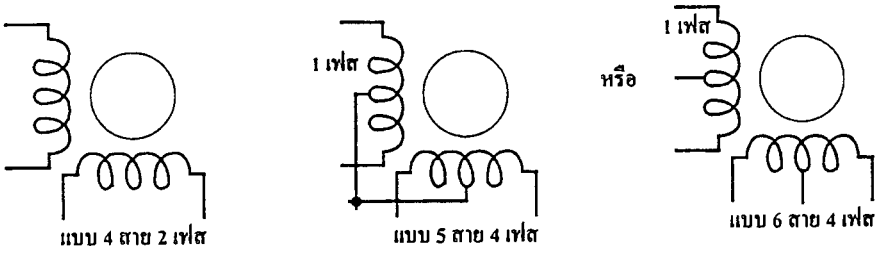
ชนิดวาริเอเบิลรีลักแดนซ์มีโครงสร้างของโรเตอร์แบบมัลติทูธ (multi-tooth) ทำจากเหล็กอ่อน เราจะทราบได้ว่าเป็นมอเตอร์ชนิดนี้โดยการทดสอบได้ง่าย ๆ คือใช้นิ้วหมุนเพลาของมอเตอร์และสังเกตมอเตอร์ชนิดนี้ที่โรเตอร์จะไม่เกิดปรากฏการณ์ทางแม่เหล็ก (magnetism) มันจึงหมุนได้ตลอดโดยไม่ติดขัดแตกต่างจากชนิด PM และชนิดไฮบริดซึ่งมีสนามแม่เหล็กที่โรเตอร์เมื่อหมุนจะรู้สึกขัด ๆ เหมือนเป็นฟันเฟือง สเต็ปเปอร์มอเตอร์ชนิดนี้มีจุดด้อยในเรื่องของความถูกต้องของตำแหน่งและทำงานได้ไม่ดีนักเมื่อมีสเต็ปในการหมุนสูง

ชนิดเพอร์มาเนนต์แมกเน็ต มีโครงสร้างของโรเตอร์แบบเรียบไม่มีซี่ขั้วแม่เหล็ก และบนโรเตอร์จะเป็นแบบแม่เหล็กถาวรการควบคุมทำได้โดยป้อนกระแสกระตุ้นที่ขดลวดบนสเตเตอร์ เช่น ถ้าเป็น สเตเตอร์แบบ 4 เฟสจะมีขั้วแม่เหล็กอยู่ 4 ขั้วซึ่งมีคอยล์พันอยู่แยกจากกัน ขั้วแม่เหล็กบนสเตเตอร์เมื่อป้อนกระแสไฟฟ้าเข้าสู่ขดลวด และโรเตอร์จะอยู่คงที่ที่ขั้วแม่เหล็กบนสเตเตอร์นั้นถึงแม้ว่าจะไม่ป้อนกระแสไฟฟ้าอีกต่อไป ทำให้เกิดเป็นแรงยึดหน่วงขึ้น สเต็ปเปอร์มอเตอร์ชนิดนี้มีข้อดีในเรื่องของความถูกต้องของตำแหน่ง และความเร็วมากขึ้นเมื่อเปรียบเทียบกับชนิดอื่น

ชนิดไฮบริดเป็นชนิดที่นิยมใช้งานกันมากที่สุด โดยเฉพาะนำมาใช้กับงานอย่างมากในอุปกรณ์ที่ใช้ในเครื่องคอมพิวเตอร์ชนิดไฮบริดมีโครงสร้างภายในซึ่งได้จากการรวบเอาโครงสร้างของสเตเตอร์ชนิดวาริเอเบิลรีลักแตนซ์ และโครงสร้างของโรเตอร์จากชนิดเพอร์มาเนนต์แมคเน็ตมาประกอบเข้าด้วยกันจึงทำให้เป็นมอเตอร์ชนิดที่มีแรงยึดหน่วงสูง มีแรงบิดดึงและผลักได้ดีซึ่งมีความคงที่และทำงานได้ดี ถึงแม้ว่าจะมีสเต็ปต่อรอบในการหมุนสูง

การพันขดลวด หรือ คอยล์ บนสเต็ปเปอร์มอเตอร์ มีอยู่ 2 วิธีคือ แบบไบโพลาร์ (bipolar) และแบบยูนิโพลาร์ (unipolar) ดังแสดงในรูปที่ 2.2

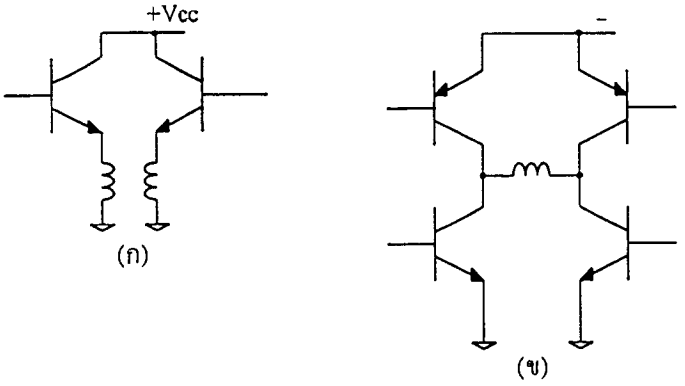
สเต็ปเปอร์มอเตอร์แบบไบโพลาร์มีการพันขดลวด 1 ขดบนแต่ละขั้วแม่เหล็กของสเตเตอร์ขั้ว แม่เหล็กที่เกิดขึ้นบนสเตเตอร์ถูกกำหนดโดยทิศทางของกระแสไฟฟ้า และสามารถทำให้เกิดขั้วแม่เหล็กในทิศทางตรงข้ามได้โดยการกลับทิศการไหลของกระแสไฟฟ้า ซึ่งการกำหนดทิศการไหลและการกลับทิศทางของกระแสไฟฟ้าทำได้โดยการใช้วงจรสวิตชิ่งกลับขั้วไฟฟ้า



รูปที่ 2.2 แสดงการพันขดลวด บนสเตเตอร์ของสเต็ปเปอร์มอเตอร์ ด้านซ้ายเป็นแบบ ไบโพลาร์ ซึ่งมีทั้งแบบ 5 สาย และ 6 สาย 4 เฟส

สำหรับยูนิโพลาร์จะมีการพันขดลวด 2 ขด บนแต่ละขั้วแม่เหล็กของสเตเตอร์ ซึ่งแต่ละขดจะทำให้เกิดขั้วแม่เหล็กในทิศทางตรงข้ามกัน การกลับขั้วแม่เหล็กเปลี่ยนไปมาทำได้โดยการสวิตชิ่งกระแสไฟฟ้าจากขดลวดขดหนึ่งไปยังอีกขดหนึ่งแทนเท่านั้นโดยปกติขดลวดทั้งสองจะมีการเชื่อมต่อกันหรือมีจุดร่วม เพื่อลดจำนวนของสายไฟที่ต่อจากมอเตอร์ วงจรจ่ายกำลังไฟฟ้าของมอเตอร์แบบ ยูนิโพลาร์ทำได้ง่ายกว่าชนิดไบโพลาร์ เพราะมันต้องการเพียงสวิตช์ธรรมดาในการเปิดและปิดกำลังไฟฟ้าให้กับขดลวดบนสเตเตอร์ ในทิศทางที่ต้องการให้หมุนได้ทันที รูปที่ 2.3 แสดงวงจรจ่ายกำลังไฟฟ้าซึ่งใช้ทรานซิสเตอร์ทำหน้าที่ เป็นตัวตัวสวิตชิ่งให้กับสเต็ปเปอร์มอเตอร์ที่มีการพันขดลวดทั้ง 2 แบบ จะเห็นได้ว่าในแบบของยูนิโพลาร์เป็นวงจรที่ง่ายและไม่มีความซับซ้อนเลย

อย่างไรก็ตามการพันขดลวดแบบยูนิโพลาร์ ก็มีจุดด้อยตรงที่การพันแบบนี้ จำทำให้เกิดแรงบิดน้อยกว่าแบบไบโพลาร์ เพราะจะมีเพียงครึ่งหนึ่งขดลวดที่ถูกกระตุ้น ให้ทำงานเท่านั้นในระยะเวลาหนึ่งการพิจารณาว่าสเต็ปเปอร์มอเตอร์ตัวใด มีการพันขดลวดแบบใดสังเกตได้ง่าย โดยถ้าเป็นแบบไบโพลาร์จะมีสายไฟต่อออก จากมอเตอร์เพียง 4 สาย และถ้าเป็นแบบ ยูนิโพลาร์จะมี 5 หรือ 6 สาย หรือทราบได้โดยการอ่านป้าย (name plate) ที่คิดอยู่กับมอเตอร์ก็ได้



รูปที่ 2.3 แสดงวงจรจ่ายกำลังไฟฟ้าให้กับสเต็ปเปอร์มอเตอร์ทั้ง 2 แบบ
(ก) สำหรับชนิดยูนิโพลาร์ ซึ่งใช้ TR. สวิตช์เพียงตัวเดียวต่อหนึ่งคอยล์
(ข) สำหรับชนิดไบโพลาร์ ซึ่งต้องใช้ TR. สวิตช์ 4 ตัวต่อหนึ่งคอยล์

2.1.2 การกระตุ้นและควบคุมการหมุนของสเต็ปเปอร์มอเตอร์

การกระตุ้นแบบควบคุมการหมุนของมอเตอร์ให้เคลื่อนที่ไปแต่ละสเต็ปทำได้โดยจ่ายกำลังไฟฟ้าไปยังขดลวดแต่ละขดบนสเตเตอร์ ซึ่งต้องป้อนเป็นแบบซีเควนเชียลในรูปแบบที่ถูกต้องด้วย แบ่งออกได้เป็น 3 รูปแบบคือ แบบเวฟ (wave) แบบ 2 เฟส (two phase) และแบบครึ่งสเต็ป(half step) ทั้ง 3 แบบต่างก็มีข้อดีและข้อเสียแตกต่างกันออกไป

แบบเวฟเป็นการกระตุ้นรูปแบบที่ง่ายที่สุดโดยทำการกระตุ้นขดลวดทีละขดในเวลาหนึ่งและเรียงถัดกันไป ดังเช่น ขดที่ 1, 2, 3, 4, 1 หรือ 1, 4, 3, 2, 1 ขึ้นอยู่กับทิศทางที่ต้องการให้หมุน ดังนั้นจึงมี ขดลวดเพียงขดเดียวในเวลาหนึ่งที่ถูกกระตุ้นเท่านั้น วงจรกระตุ้นแบบเวฟจึงมีราคาถูกและง่าย ขั้นตอนการทำงานต่าง ๆ แสดงดังในตารางที่ 2.1

ตารางที่ 2.1 แสดงขั้นตอนการกระตุ้นขดลวดแต่ละเฟสแบบเวฟ

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่3	เฟสที่ 4
1	ทำงาน	-	-	-
2	-	ทำงาน	-	-
3	-	-	ทำงาน	-
4	-	-	-	ทำงาน

แบบ 2 เฟสเป็นการกระตุ้นอีกรูปแบบหนึ่งซึ่งคล้ายกับแบบเวฟ แต่การกระตุ้นแบบนี้จะทำการกระตุ้นโดยจ่ายกำลังไฟฟ้าไปที่ขดลวด 2 ขด ที่อยู่ใกล้กันในเวลาเดียวกัน และเรียงถัดกันไป

เช่นเดียวกับแบบเวฟคือ ขดลวดที่ถูกกระต้น 12 , 23 ,34 ,41 ,12 หรือ 14, 43, 32, 21,14 ขึ้นอยู่กับทิศทางการหมุนการเพิ่มจำนวนของขดลวดที่ถูกกระตุ้นนี้ทำให้เพิ่มแรงบิดได้มากกว่าแบบเวฟ โรเตอร์จะเคลื่อนที่ด้วยแรงดึงอย่างเต็มแรงจาก 2 ขดลวดที่ถูกกระตุ้นพร้อมกัน และต่อไปด้วยแรงดึงจากอีก 2 ขดลวดถัดไป สำหรับ ข้อเสียก็คือการกระตุ้นแบบนี้ต้องใช้แหล่งจ่ายกำลังไฟฟ้ามากขึ้น ขั้นตอนการทำงานต่าง ๆ แสดงใน ตารางที่ 2.2

ตารางที่ 2.2 แสดงขั้นตอนการกระตุ้นขดลวดแต่ละเฟสแบบ 2 เฟส

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่3	เฟสที่ 4
1	ทำงาน	ทำงาน	-	-
2	-	ทำงาน	ทำงาน	-
3	-	-	ทำงาน	ทำงาน
4	ทำงาน	-	-	ทำงาน

แบบครึ่งสเต็ปเป็นรูปแบบที่เกิดจากการผสมผสานระหว่างการกระตุ้นแบบ เวฟและ แบบ 2 เฟสเพื่อเพิ่มจำนวนของสเต็ปต่อรอบอีกเท่าตัวหนึ่งในระบบนี้จะทำการกระตุ้นขดลวดเรียงกันไปเป็นลำดับดังนี้ ขดลวดที่ถูกระตุ้น 1, 12, 2, 23 3, 34 ,4 ,41, 1 หรือในการหมุนอีกทิศทางหนึ่งจะได้เป็น 1, 14, 4 ,43 ,3, 32, 2, 21, 1 แรงบิดที่ได้จากการกระตุ้นแบบนี้จะเพิ่มขึ้นอีกเพราะช่วงสเต็ปมีระยะสั้นลงและต่อละ สเต็ปเกิดแรงดึงจากขดลวด 2 ขดที่ถูกระตุ้นพร้อมกัน ความถูกต้องของตำแหน่งมีเพิ่มมากขึ้น แต่ต้องพึงระวังไว้อีกประการหนึ่ง ว่าเมื่อกระตุ้นให้ทำงานในรูปแบบนี้ จะต้องทำการหมุนถึง 2 สเต็ปจึงจะได้ เท่ากับ 1 สเต็ปเต็มเหมือนกับในการควบคุม 2 แบบแรก สำหรับแหล่งจ่ายกำลังไฟฟ้าต้องใช้เทียบเท่ากับแบบ 2 เฟสจึงจะเพียงพอ ขั้นตอนการทำงาน ต่าง ๆ แสดงดังในตารางที่ 2.3

ตารางที่ 2.3 แสดงขั้นตอนการกระตุ้นขดลวดแต่ละเฟสแบบครึ่งสเต็ป

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่3	เฟสที่ 4
1	ทำงาน	-	-	-
2	ทำงาน	ทำงาน	-	-
3	-	ทำงาน	-	-
4	-	ทำงาน	ทำงาน	-
5	-	-	ทำงาน	-
6	-	-	ทำงาน	ทำงาน
7	-	-	-	ทำงาน
8	ทำงาน	-	-	ทำงาน

2.2 LED กำลังสูง ที่ให้แสงย่านใกล้อินฟราเรด

รอยต่อ P-N ของ LED เป็นแหล่งกำเนิดโฟตอนที่ยอดเยี่ยมมาก สมัยแรกๆ ได้มีการค้นพบว่ารอยต่อแกลเลียมอาร์เซไนด์ (GaAs) ซึ่งให้แสงย่านใกล้อินฟราเรด ทำให้เกิดโฟตอน 88 ตัว ต่ออิเล็กตรอน 100ตัว เป็นประสิทธิภาพควอนตัมที่น่าทึ่งไม่น้อยทีเดียว

แต่ปัญหาก็มีจนได้ เนื่องจากโฟตอนที่ปล่อยออกมาจากรอยต่อ ส่วนหนึ่งโดนสกัดกั้นโดยกรอบของชิ้นสารเอาท์พุตจริงๆ จึงน้อยกว่า 88 เปอร์เซ็นต์

อีกส่วนหนึ่งในการแผ่รังสีจากรอยต่อ ก็ถูกดูดกลืนไปโดยตัวไดโอดเองและยังมีการแผ่รังสีบางส่วนที่กระทบผิวไดโอดด้วยมุมที่มากกว่ามุมวิกฤติ (Critical angle) จึงโดนสะท้อนไปยังไดโอดทำให้เกิดการสูญเสียด้วย

ปัญหาเรื่องมุมวิกฤติ เป็นปัญหาสำคัญประการหนึ่ง ซึ่งอธิบายได้ดังนี้

อากาศมีดัชนีหักเหเท่ากับ 1 (หรือ 1.0003 เมื่อเทียบกับสุญญากาศ) ดัชนีหักเหของเพชรคือ 2.42 ในขณะที่ GaAs มีดัชนีการหักเหของแสงเท่ากับ 3.5 ซึ่งเป็นหนึ่งในสสารเพียงไม่กี่ชนิดที่มีดัชนีของ การหักเหมากกว่าเพชร ความแตกต่างของดัชนีการหักเหนี้ทำให้เกิดการแผ่รังสีของแสงที่เกิดขึ้นภายในชั้นสาร GaAs เมื่อกระทบผิวที่ติดต่อกับอากาศภายนอกมีมุมวิกฤติ 16 องศา หากมุมตกกระทบที่รอยต่อที่ผิวมากกว่า 16 องศา แสงนั้นจะถูกสะท้อนกลับมายังชั้นสาร ปรากฏการเช่นนี้เรียกว่า ปรากฏการสะท้อนกลับภายใน (Total internal reflection)

วิธีลดความไม่สมดุลย์ระหว่างดัชนีการหักเหของแสงมีอยู่หลายวิธี

วิธีหนึ่งเป็นของเท็กซัสอินสตรูเมนต์คือ ทำผิวหน้าของชิ้นไดโอดเป็นรูปโดม ทำให้แสงจากภายในมายังผิวต่อระหว่างอากาศและ GaAs ด้วยมุมที่ไม่เกิน 16 องศา วิธีนี้สามารถเพิ่มประสิทธิภาพให้สูงกว่าชิ้นไดโอดแบบแบนราบประมาณ 10 เท่า

การผลิตไดโอดแบบโดมนี้มีราคาค่อนข้างแพง แล้วยังต้องการสาร GaAs มากกว่าแบบแบนราบด้วย จึงมีการใช้อีพ็อกซี่เคลือบไดโอด เพื่อทำให้ดัชนีการหักเหของแสงสมดุลย์ดัชนีการหักเหของสารเคลือบนี้มีค่าตั้งแต่ 1.4 ถึง 1.8 แม้ว่าจะไม่เท่ากับดัชนีหักเหของ GaAs (3.5) ก็ตามแต่ก็มีค่าใกล้เคียงมากกว่าอากาศ

ส่วน LED ที่ทำจากแกลเลียมฟอสไฟด์ (GaP) ที่ให้แสงสีเขียวและมีมุมวิกฎติ 17.7 องศา การเคลือบไดโอดด้วยอิพ็อกซี่ที่มีดัชนีหักเห 1.66 ทำให้เพิ่มมุมวิกฎติเป็น 30.3 องศา และทำให้การเปล่งแสงจากไดโอดเพิ่ม ขึ้นอีก 2.5 เท่า

หันมาพิจารณาถึงแสงที่ปล่อยออกมาจากขอบของสารที่ใช้ในการสร้าง LED ในสมัยแรกๆ นั้นแสงส่วนนี้จะสูญเสียไปถ้าชิ้นสารถูกวางไว้ในกรอบโลหะ ปัญจุบัน LED ส่วนใหญ่จึงถูกวางไว้ในตัวสะท้อน (reflectors) ตัวเล็กๆ ตัวสะท้อนนี้จะสะท้อนแสงที่เปล่งจากขอบของชิ้นสารออกมาสู่ภายนอก

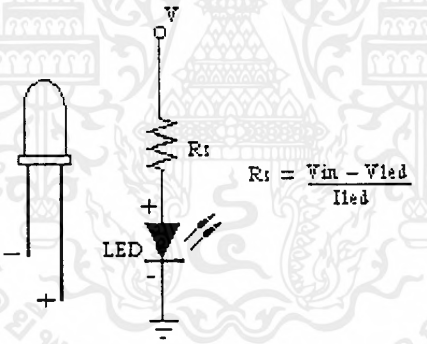
2.2.1 คุณสมบัติของ LED อินฟราเรด

แรงดันตกคร่อมที่รอยต่อ P-N ของไดโอด ต้องมีค่ามากกว่าแรงดันเทรชโฮลด์จึงสามารถทำให้ ไดโอดนำกระแสได้ สำหรับซิลิคอนไดโอดแรงดันทำงานมีค่าประมาณ 0.6 โวลต์ ส่วน LED ที่ให้แสงในย่านมองเห็นได้ถ้าทำจากสาร GaP ซึ่งให้แสงสีเขียว จะมีค่าแรงดันทำงานประมาณ 2.1 ถึง 2.8 โวลต์ ถ้าเป็น LED ที่ทำจาก AlGaAs ให้แสงสีแดง มีแรงดันทำงาน 1.75 ถึง 2.5 โวลต์ส่วน LED ที่ให้แสงใกล้ย่านอินฟราเรดทำจากสาร GaAs มีแรงดันทำงาน 1.5 โวลต์ โดยให้แสงมีความยาวคลื่น 940 นาโนเมตร และ ถ้าทำจาก AlGaAs จะได้แสงความยาวคลื่น 880 นาโนเมตร ที่แรงดัน 1.75 โวลต์

พลังงานที่ได้จากการเปล่งแสงของ LED หาได้จากกระแสไบแอสตรงของไดโอด และต้องระมัดระวังไม่ให้กระแสส่วนนี้มีค่าสูงจนเกิดความร้อนอันจะทำอันตรายต่อชิ้นไดโอด

สิ่งจำเป็นในการใช้งาน LED อย่างต่อเนื่องคือ การต่อตัวต้านทานอนุกรมที่เหมาะสม เพื่อจำกัดกระแส ดังรูปที่ 2.5 แสดงการจำกัดกระแสเบื้องต้นของ LED โดยการคำนวณตามสูตร

หากใช้กระแสเกินพิกัดจะทำให้อายุการใช้งานสั้นลง LED กำลังสูงต้องยึดติดกับแผ่นระบายความร้อนเสมอเมื่อใช้กับกระแสเกินระดับปกติ



รูปที่ 2.4 การจำกัดกระแสของ LED เบื้องต้น

2.3 คุณสมบัติทั่วไปของไมโครคอนโทรลเลอร์ MCS-51

คุณสมบัติทั่วไปที่สำคัญของไมโครคอนโทรลเลอร์ตระกูล MCS-51

- เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิต
- มีวงจรออสซิลเลเตอร์และวงจรผลิตสัญญาณนาฬิกาภายในไอซี
- มีขาสัญญาณอินพุตเอาร์พุตจำนวน 32 บิต
- สามารถเชื่อมต่อหน่วยความจำข้อมูลภายนอก (external data memory) โดยอ้างตำแหน่งแอดเดรสได้ถึง 64 K

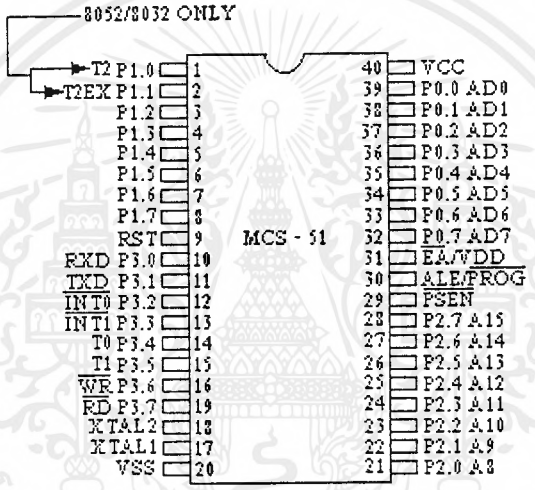
- สามารถเชื่อมต่อหน่วยความจำโปรแกรมภายนอก (external program memory) โดยอ้างตำแหน่งแอดเดรสได้ถึง 64 K
- มีหน่วยความจำโปรแกรมภายในตัว (on-chip program memory) ขนาด 4 K โดยเฉพาะเบอร์ 8052 จะมีหน่วยความจำในส่วนนี้ถึง 8 K สำหรับเบอร์ 8031 และ M8032 จะไม่มีหน่วยความจำในส่วนนี้
- มีหน่วยความจำข้อมูลภายในตัว (on chip data memory) ขนาด 128 ไบต์ โดยเฉพาะเบอร์ 8032 และ 8052 จะมีหน่วยความจำในส่วนนี้ถึง 256 ไบต์
- หน่วยความจำข้อมูลภายในบางส่วนสามารถเข้าถึงข้อมูลระดับบิตได้ด้วย ทำให้การควบคุมหรือการตรวจสอบสถานะบิตทำได้ง่าย ส่งผลให้การเขียนโปรแกรมทำได้ง่ายมากขึ้น
- มีไทเมอร์ / เคาน์เตอร์ (timer / counter) ขนาด 16 บิต จำนวน 2 ตัว โดยเฉพาะเบอร์8032 หรือ 8052 จะมีไทเมอร์ / เคานเตอร์จำนวน 3 ตัว
- การอินเตอร์รัปต์สามารถทำได้จาก 5 แหล่งกำเนิดโดยเฉพาะเบอร์ 8032 และ 8052 จะทำการอินเตอร์รัปต์ได้จาก 6 แหล่งกำเนิด โดยการอินเตอร์รัปต์ยังสามารถจัดระดับความสำคัญได้เป็น 2 ระดับ
- มีพอร์ตสื่อสารอนุกรมภายในตัวเอง ซึ่งทำงานเป็นแบบฟูลดูเพล็กซ์ (full duplex)
- มีคำสั่งในการคำนวณทางคณิตศาสตร์และทางตรรกศาสตร์
- คำสั่งโดยส่วนใหญ่ใช้เวลาการทำงานเพียง 1 ไมโครวินาที เมื่อใช้คริสตอลความถี่ 12 เมกะเฮิรตซ์
- ต้องการแหล่งจ่ายไฟ 5 โวลต์ เพียงชุดเดียว

2.3.1 โครงสร้างภายนอกของ MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ทุกเบอร์จะมีตำแหน่งขาพื้นฐานที่เหมือนกัน ดังแสดงในรูปที่ 2.5 สำหรับหน้าที่การใช้งานของแต่ละขามีดังนี้

- ขา Vcc เป็นขาป้อนแรงดันไฟเลี้ยง + 5 โวลท์
- ขา Vss เป็นขากราวด์
- ขาพอร์ต 0 (Port 0) มี 8 ขาได้แก่ขา P0.0 – P0.7 เป็นขาพอร์ตอินพุตเอาร์พุตแบบ 2 ทิศทาง สำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุตเอาร์พุตพอร์ตต้องทำการเขียนค่า 1 ไปยังแต่ละบิตของพอร์ต เพื่อกำหนดให้ขาพอร์ตเหล่านั้นอยู่ในสถานะปล่อยลอย ซึ่งในสถานะนี้เองที่สามารถนำมาใช้เป็นพอร์ตอินพุตอิมพีแดนซ์สูงได้ นอกจากพอร์ตนี้จะใช้งานเป็นพอร์ตอินพุตเอาร์พุตแล้วมันยังถูกใช้งานติดต่อกับหน่วยความจำภายนอกด้วย โดยทำหน้าที่ในการกำหนดตำแหน่งแอดเดรสไบต์ต่ำ (A0 – A7) ซึ่งจะใช้งานเป็นแบบมัลติเพล็กกับการรับส่งข้อมูลขนาด 8 บิต (D0 – D7)

- ขาพอร์ต 1 (Port 1) มี 8 ขา ได้แก่ขา P1.0 – P1.7 เป็นขาพอร์ตอินพุตเอาร์พุตแบบ 2 ทิศทาง สำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุตพอร์ตต้องทำการเขียนค่า 1 ไปยังแต่ละบิตของพอร์ต เพื่อกำหนดให้เป็นพอร์ตอินพุต นอกจากนี้สำหรับเบอร์ 8032 และ 8052 ขาพอร์ต P1.0 และ P1.1 จะถูกนำมาใช้งานเป็นขา T2 และ T2EX ตามลำดับด้วย
- ขาพอร์ต 2 (Port 2) มี 8 ขา ได้แก่ขา P2.0 – P2.7 เป็นขาพอร์ตอินพุตเอาร์พุตแบบ 2 ทิศทางสำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุตพอร์ตต้องทำการเขียนค่า 1 ไปยังแต่ละบิตของพอร์ตเพื่อกำหนดให้เป็นพอร์ตอินพุต นอกจากพอร์ตนี้จะใช้งานเป็นพอร์ตอินพุต-เอาร์พุตแล้วมันยังถูกใช้งานในการติดต่อกับหน่วยความจำภายนอกด้วย โดยทำหน้าที่ในการกำหนดตำแหน่งแอดเดรสไบต์สูง (A0 – A15)



รูปที่ 2.5 แสดงการจัดตำแหน่งขาต่าง ๆของไมโครคอนโทรลเลอร์ตระกูล MCS-51

- ขาพอร์ต 3 (Port 3) มี 8 ขา ได้แก่ขา P3.0 – P3.7 เป็นขาพอร์ตอินพุตเอาร์พุตแบบ 2 ทิศทางสำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุตพอร์ตต้องทำการเขียนค่า 1 ไปยังแต่ละบิตของพอร์ต เพื่อกำหนดให้เป็นพอร์ตอินพุตนอกจากพอร์ตนี้จะใช้งานเป็นพอร์ตอินพุตเอาร์พุตแล้วมันยังถูกใช้งานในหน้าที่พิเศษต่าง ๆ ดังแสดงในตารางที่ 2.4

ขาพอร์ต	หนาที่พิเศษ
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	INT0 (external interrupt 0)

P3.3	INT1 (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	WR (external data memory write strobe)
P3.7	RD (external data memory read strobe)

ตารางที่ 2.4 แสดงหน่าที่พิเศษของแต่ละขาของพอร์ต P3

- ขารีเซต (RST) ใช้สำหรับรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยการรีเซตต้องคงสถานะเป็น 1 อย่างน้อยนาน 2 แมชีนไซเกิล ในขณะที่ออสซิลเลเตอร์ยังทำงานอยู่
- ขา ALE/PROG เป็นชาสัญญาณเพื่อทำหน้าที่ควบคุมการแลตช์ (latch) ค่าตำแหน่งแอดเดรสไบต์ต่ำ (Address Latch Enable) เมื่อต้องการติดต่อกับหน่วยความจำภายนอก นอกจากนี้ขานี้ยังทำหน้าที่เป็นอินพุตรับพัลส์ในการโปรแกรม (program pulse input) ในส่วนของหน่วยความจำ EPROM สำหรับไมโครคอนโทรลเลอร์ในตระกูล MCS-51 ที่มีหน่วยความจำโปรแกรมภายในเป็น EPROM
- ขา PSEN (Program Store Enable) ทำหน้าที่เป็นสัญญาณสโตรบเพื่ออ่านคำสั่งจากหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ประมวลผลคำสั่งจากหน่วยความจำภายนอก ขานี้จะส่งสัญญาณสโตรบจำนวน 2 ครั้งในแต่ละแมชีนไซเกิล แต่ในขณะที่ติดต่อกับหน่วยความจำข้อมูลภายนอกจะไม่มีการส่งสัญญาณสโตรบแต่อย่างใด
- ขา EA/VPP (External Access enable/VPP) เป็นขาสำหรับการเลือกใช้หน่วยความจำโปรแกรมจากภายในหรือภายนอก โดยถ้ามีสถานะเป็น 0 จะหมายถึงให้ไมโครคอนโทรลเลอร์รับคำสั่งจากหน่วยความจำภายนอกที่ตำแหน่งแอดเดรส 0 – 0FFFH (0-1FFFH ถ้าเป็นเบอร์ 8052) อย่างไรก็ตามถ้าบิตป้องกัน (security bit) ในหน่วยความจำ EPROM ถูกโปรแกรมไว้ ไมโครคอนโทรลเลอร์จะไม่รับคำสั่งจากหน่วยความจำภายนอกเลย นอกจากนี้ขานี้ยังทำหน้าที่รับแรงดันไฟสำหรับการโปรแกรม (Vpp) ขนาด 21 โวลท์ เพื่อใช้ในระหว่างการโปรแกรม EPROM
- ขา XTAL1 และขา XTAL2 เป็นขาอินพุตของวงจรอินเวอร์ติ้งออสซิลเลเตอร์แอมปลิไฟเออร์ (inverting oscillator amplifier) สำหรับใช้ต่อร่วมกับคริสตอลภายนอก

2.3.2 การจัดหน่วยความจำ

ในไมโครคอนโทรลเลอร์ตระกูล MCS-51 แบ่งชนิดหรือหน้าที่ของหน่วยความจำออกเป็น 2ส่วนคือหนว่ยความจำโปรแกรม (program memory) และหน่วยความจำข้อมูล (data memory) หน่วยความจำโปรแกรม จะใช้สำหรับเก็บโปรแกรมควบคุมการทำงาน ของไมโครคอนโทรลเลอร์ ซึ่งบางเบอร์จะมีหน่วยความจำในส่วนนี้อยู่ภายในตัว โดยอาจจะมีขนาดไม่เท่ากัน หรือเป็นหน่วยความจำ ต่างชนิดกัน เช่น บางเบอร์เป็น ROM และบางเบอร์อาจเป็น EPROM และบางเบอร์อาจไม่มีหน่วยความจำในส่วนนี้เลย โปรแกรมการทำงานจะถูกเก็บไว้ยังหน่วยความจำโปรแกรมภายนอกทั้งหมด

สำหรับหน่วยความจำข้อมูล จะใช้สำหรับเก็บข้อมูลหรือค่าตัวแปรต่างๆ จากการทำงานของโปรแกรม ซึ่งใน MCS-51 ทุกเบอร์จะมีหน่วยความจำในส่วนนี้อยู่จำนวนหนึ่ง แต่อาจมีขนาดมากน้อยต่างกันไปในแต่ละเบอร์

2.3.3 หน่วยความจำโปรแกรม

หน่วยความจำโปรแกรมสามารถแบ่งออกได้เป็น 2 ส่วนคือ หน่วยความจำโปรแกรมภายในและหน่วยความจำโปรแกรมภายนอก หน่วยความจำโปรแกรมภายในจะถูกเลือกใช้งานถ้าขาสัญญาณ EA มีค่าเป็น 1 โดยจะถูกใช้งานในช่วงแอดเดรส 0 – 0FFFH (หรือช่วงแอดเดรส 0 –1FFFH ในเบอร์ 8052) นอกเหนือจากช่วงแอดเดรสนี้จะใช้หน่วยความจำโปรแกรมภายนอกทั้งหมด ในกรณีตรงกันข้ามถ้าขาสัญญาณ EA มีค่าเป็น 0 ในช่วงแอดเดรส 0 – 0FFFH (หรือช่วงแอดเดรส0 – 1FFFH ในเบอร์ 8052) จะถูกใช้จากหน่วยความจำภายนอก หรือกล่าวได้ว่าถ้าขาสัญญาณ EA มีค่าเป็น 0 จะเป็นการเลือกใช้ หน่วยความจำโปรแกรมภายนอกทั้งหมดตลอดช่วงแอดเดรส

2.3.4 หน่วยความจำข้อมูล

หน่วยความจำข้อมูลสามารถแบ่งออกได้เป็น 2 ส่วนคือ หน่วยความจำข้อมูลภายในและหน่วยความจำข้อมูลภายนอก สำหรับหน่วยความจำข้อมูลภายในยังแบ่งออกได้เป็น 2 ส่วนย่อยคือ ส่วนที่ใช้เก็บข้อมูลทั่วไปและส่วนที่ใช้เป็นรีจิสเตอร์หน้าที่พิเศษหรือ SFR (Special Function Register) โดยส่วนที่ใช้เก็บข้อมูลทั่วไปจะถูกใช้สำหรับเก็บข้อมูลหรือค่าตัวแปรต่าง ๆ จากการทำงานของโปรแกรม ส่วนรีจิสเตอร์หน้าที่พิเศษจะถูกใช้งานเป็นรีจีสเตอร์ควบคุมการทำงานของไมโครคอนโทรเลอร์

ไมโครคอนโทรลเลอร์ตระกูล MCS - 51 ทุกเบอร์จะมีหน่วยความจำข้อมูลภายในขนาด 128 ไบต์เป็นอย่างน้อย และบางเบอร์อาจมีถึงขนาด 256 ไบต์

2.3.5 รีจิสเตอร์หน้าที่พิเศษ (SFR)

รีจิสเตอร์หน้าที่พิเศษมีบทบาทอย่างมากในการควบคุมการทำงานของไมโครคอนโทรลเลอร์และทำให้การเขียนโปรแกรมสามารถทำได้สะดวกมากขึ้น รีจิสเตอร์หน้าที่พิเศษทำหน้าที่สำคัญคือ ควบคุมการทำงานในส่วนต่าง ๆ ภายในไมโครคอนโทรลเลอร์และทำหน้าที่แสดงสถานะการทำงาน ซึ่งในรีจิสเตอร์หน้าที่พิเศษบางตัวยังสามารถเข้าถึงได้ในระดับบิต (bit addressable) ด้วย

2.3.6 รีจิสเตอร์ใช้งานทั่วไป

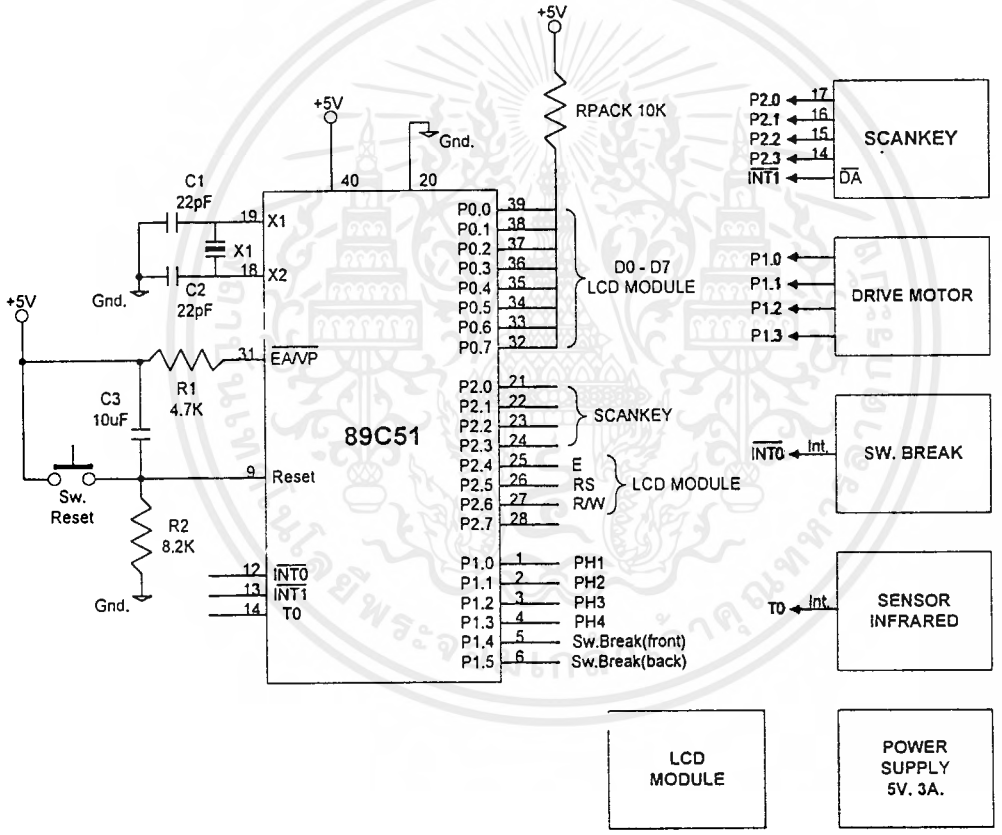
รีจิสเตอร์ใช้งานทั่วไปมีไว้สำหรับให้ผู้เขียนโปรแกรมสามารถนำข้อมูลไปพักไว้ชั่วคราวหรือใช้งานทั่วไปได้ตามต้องการ ซึ่งรีจิสเตอร์ใช้งานทั่วไปนี้มีอยู่ด้วยกัน 8 ตัวคือรีจิสเตอร์ R0 – R7 โดยที่ รีจิสเตอร์ทั้ง 8 ตัวถูกจัดให้อยู่รวมกันและมีให้เลือกใช้ถึง 4 แบงค์ (bank) นั่นคือมีรีจิสเตอร์ใช้งานทั่วไปถึง 32 ตัวให้ใช้งาน เพียงแต่การเลือกใช้รีจิสเตอร์ R0 – R7 ในแบงค์ใดแบงค์หนึ่งจะถูกกำหนดจากบิต RS0,RS1 ในรีจิสเตอร์หน้าที่พิเศษ PSW ดังนั้นการเลือกใช้จึงเลือกได้เพียงแบงค์เดียวในขณะใดขณะหนึ่ง อย่างไรก็ตามค่าข้อมูลที่เก็บไว้ในรีจิสเตอร์แบงค์ใดก็ตามที่มีชื่อเดียวกันแต่อยู่คนละแบงค์จะไม่มีผลซึ่งกันและกันเลย ทำให้ผู้เขียนโปรแกรมใช้งานรีจิสเตอร์ทั่วไปนี้ได้ทั้ง 32 ตัวอย่างเต็มที่และไม่ยุ่งยากในการเขียนโปรแกรม

บทที่ 3

องค์ประกอบโดยรวมของระบบและการทำงาน

3.1 โครงสร้างและองค์ประกอบของระบบ

จากหลักการเบื้องต้นและคุณสมบัติต่าง ๆ ที่ได้กำหนดขึ้นมา เขียนเป็นบล็อกไดอะแกรมที่แบ่งเป็นส่วนต่าง ๆ ตามรูปที่ 3.1 และเพื่อให้ง่ายต่อการทำความเข้าใจจะอธิบายการทำงานแยกส่วนกันไป



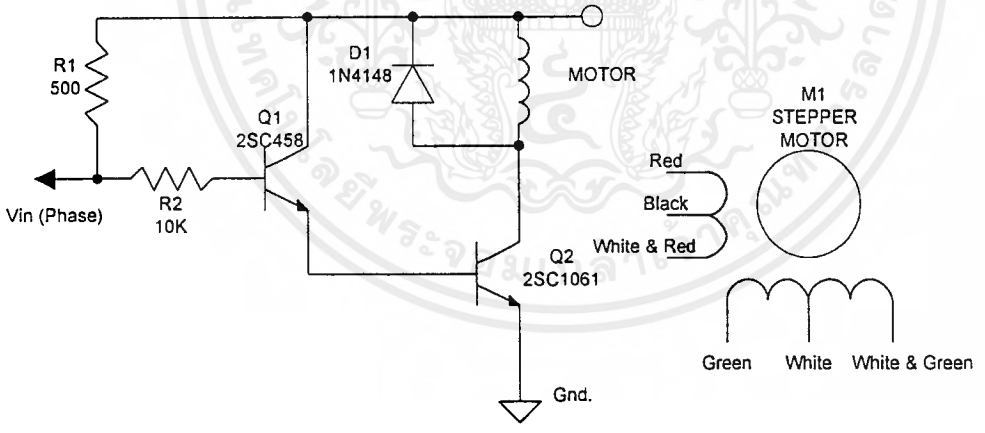
รูปที่ 3.1 บล็อกไดอะแกรมของระบบ

3.2 ภาคประมวลผล

จากบล็อกไดอะแกรมจะเห็นว่าเราจะใช้ IC ประมวลผลเบอร์ 89C51 ซึ่งมีหลักการทำงานที่ง่าย และเขียนโปรแกรมควบคุมที่ง่ายต่อการเข้าใจ โดยตามรูปที่ 3.1 มีหลักการทำงานคือ 89C51 จะรอรับ ข้อมูลจากภาคสแกนคีย์ โดยภาค สแกนคีย์ จะส่งสัญญาณ อินเตอร์รัปต์ ออกมาเมื่อมีการกดคีย์ และก็จะส่งข้อมูลคีย์ ที่กดออกมาส่งต่อไปยัง PORT 2.0 – 2.3 และเมื่อ 89C51 ได้รับสัญญาณอินเตอร์รัปต์ ก็จะกระโดดไปยังโปรแกรมของอินเตอร์รัปต์ของภาคสแกนคีย์ เพื่อดูว่าเป็นข้อมูลของคีย์อะไร แล้วทำตาม คำสั่งนั้น

ส่วนตรวจรับสัญญาณอินเตอร์รัปต์ ของภาคสวิทช์เบรคนั้น ที่ตัว 89C51 จะใช้ขาสัญญาณอินเตอร์รัปต์คือ INT0 เพื่อคอยเช็คว่าการดันหลอดฉีดหมดหลอดหรือยัง และส่วนของภาคเซ็นเซอร์อินฟราเรดนั้น เราจะใช้ขาสัญญาณ T0 เพื่อคอยเช็คว่าตอนนี้มีปริมาณของเหลวเหลืออยู่ในหลอดเท่าไหร่และยังภาคแสดงผลนั้น ไมโครคอนโทรลเลอร์จะส่งข้อมูลออกไปแสดงผล และในส่วนสุดท้ายคือภาคขับมอเตอร์ ที่ตัว 89C51 นั้นจะคอยส่งพัลส์ที่เป็นลำดับเพื่อขับมอเตอร์ให้หมุนตามต้องการต่อไป

3.3 ภาคขับเคลื่อนมอเตอร์ (DRIVE MOTOR)



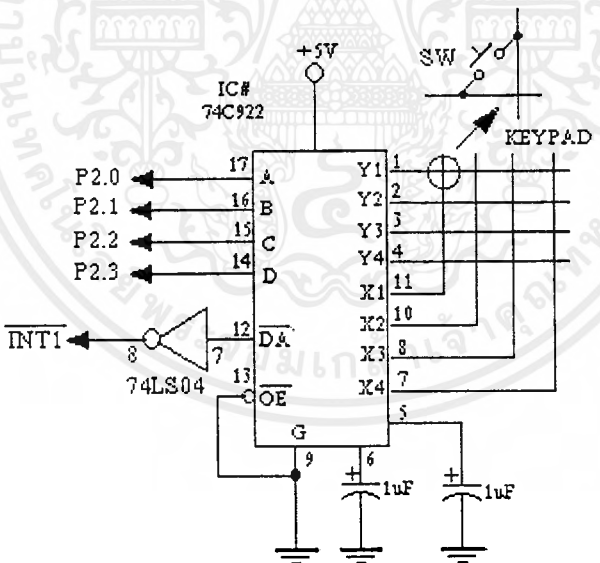
รูปที่ 3.2 วงจรขับเคลื่อนมอเตอร์

ในโครงงานนี้เราเลือกใช้ สเต็ปเปอร์มอเตอร์ที่มีการพันขดลวดแบบยูนิโนลาร์ แบบ 6 สาย 4 เฟส (เป็นของเก่าและหาซื้อได้ในราคาไม่แพง) ซึ่งจากชิ้นงานจะต้องใช้วงจรขับเคลื่อน 4 ชุด (ต่อมอเตอร์ 1 ตัวนั่นเอง) แต่ละชุดจะใช้ขับมอเตอร์แต่ละเฟสจะแยกกันไป การทำงานของแต่ละชุดก็คือ เมื่อมีพัลส์มาที่อินพุตของแต่ละชุด สัญญาณพัลส์จะไปกระตุ้น Q1 ให้ทำงาน โดย Q1 และ Q2 ต่อกันแบบดาร์ลิงตัน เมื่อ Q1 ทำงาน Q2 ก็จะทำงานด้วย เมื่อ Q2 ทำงานก็จะมีกระแสไหลผ่านขด

ลวดมอเตอร์ ทำให้เกิดการขับเคลื่อนของมอเตอร์ ไดโอด 1N4148 ทำหน้าที่ป้องกันแรงดันย้อนกลับจากการยุบตัวของเส้นแรงแม่เหล็กของขดลวดมอเตอร์ และ R_B ทำหน้าที่จำกัดกระแสเกินที่ไปขับ Q1 และ R 500 โอห์มจะทำหน้าที่จำกัดกระแสที่ไหลเข้าตัว 89C51 เมื่อ เป็นสถานะลอจิกต่ำ (LOGIC "0")

มอเตอร์ที่ใช้ตามที่ระบุในแผ่นกำกับ ต้องการกำลังงานจากแหล่งจ่ายกำลังงาน 4V, 1.6A ต่อเฟส ดังนั้นจะมีค่า RL = 2.5 Ωต่อเฟส และที่มอเตอร์ และมีสายไฟทั้งหมด 6 เส้น คือ สายสีแดง , สีขาวแดง , สีเขียว ,สีขาว และสีขาวเขียว มอเตอร์จะหมุนเมื่อป้อนไฟ 4V ที่สายไฟสีดำกับสีขาวตลอด ส่วนที่เหลือจะต้องให้ศักดาเป็นศูนย์ ตามลำดับ ดังนี้ ขาวแดง , ขาวเขียว, แดงและเขียว นั่นคือการหมุน ตามเข็มนาฬิกา แต่ถ้าต้องการให้มอเตอร์หมุนทวนเข็มนาฬิกา ก็สามารถทำได้โดยสลับลำดับการให้ศักดาเป็นศูนย์ของสายสัญญาณ การควบคุมการหมุนของมอเตอร์ให้หยุดหรือหมุน ทั้งตามเข็ม และทวนเข็มได้ในลักษณะ ดังกล่าวอย่างต่อเนื่องโดยใช้ขบวนพัลส์เรียงลำดับมานั่นเอง ซึ่งในโครงงานนี้จะใช้ 89C51 กำเนิดสัญญาณขับเคลื่อนในลักษณะดังกล่าว

3.4 ภาคสแกนคีย์ (SCAN KEY)

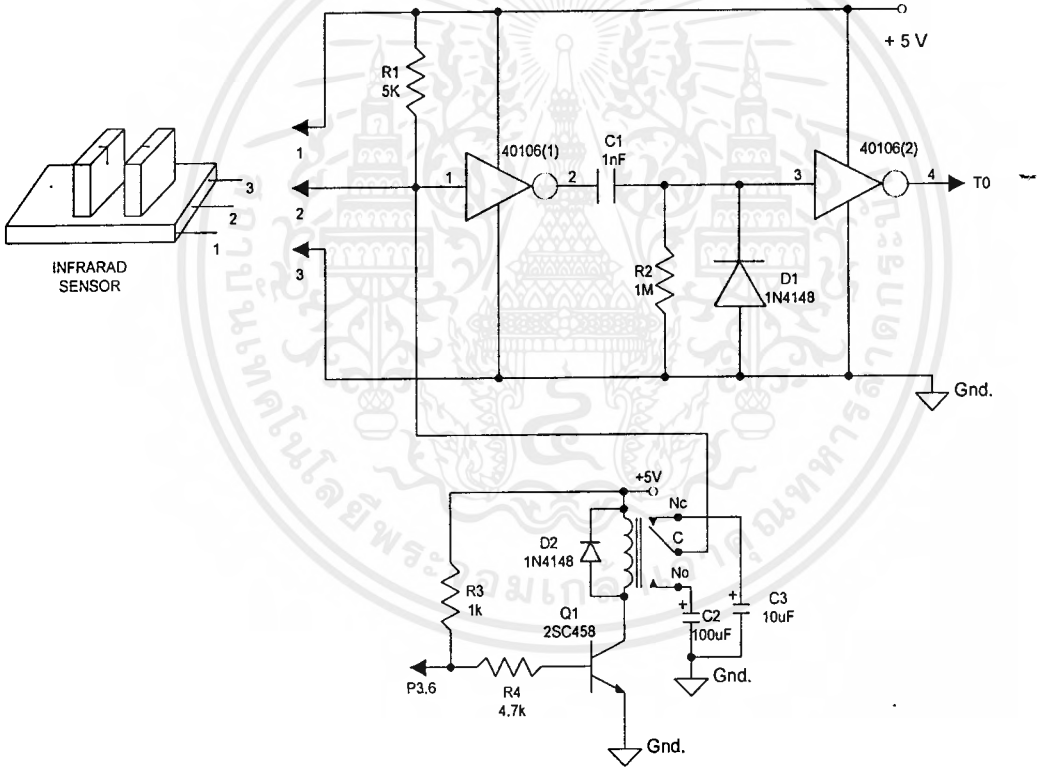


รูปที่ 3.3 วงจรภาคสแกนคีย์

จากรูปวงจรจะเห็นว่าใช้ IC ตัวเดียว คือ IC เบอร์ 74C922 ซึ่งเป็น IC ที่ผลิตขึ้นมาทำงานนี้โดยเฉพาะ คือการสแกนคีย์ โดยสแกนคีย์ ได้ 16 คีย์ โดย IC มีขาต่อขั้วของแถวและหลักต่าง ๆ ตามรูป และที่สำคัญจะมีขา OE (OUTPUT ENABLE) และขา DA (DATA INTERRUPT) ซึ่งขา DA นั้นจะเป็นขา เอาท์พุท ให้สัญญาณออกมาเป็น HIGH (ACTIVE HIGH) เมื่อมีการกดคีย์ ทุก

ครั้ง เราจึงต้องเอา NOT GATE มาต่อขึ้นไว้เพื่อให้เป็นแอคทีฟทางด้านต่ำ (ACTIVE LOW) แทน เพราะขาอินเตอร์รัปต์ของ 89C51 เป็น ACTIVE LOW ดังนั้นเมื่อ เรากดคีย์ 89C51 ก็จะรู้ทันทีว่า คีย์ถูกกดแล้ว รับข้อมูลได้แล้ว ส่วนขา OE นั้นจะ ACTIVE LOW เราจึงต่อลงกราวด์ ไว้เพื่อให้ IC อีนาเบิล (enable) เอาสัญญาณข้อมูลออกเอาท์พุทตลอดเวลา เพื่อไม่ให้ 89C51 ต้องเสียเวลามาสั่งให้ IC ตัวนี้อีนาเบิล เมื่อมีการกดคีย์ ทางด้านขาเอาท์พุท สัญญาณข้อมูลจะออกเป็น BCD 4 บิต ส่งให้ 89C51 ต่อไปและ C ที่ขา 6 ค่า 1μF ทำหน้าที่กำหนดเวลาให้กับ IC 74C922 ตัวนี้เพื่อแก้การกระเด้ง (BOUNCING) ของหน้าสัมผัสสวิทช์ เมื่อถูกกดเรียกว่า DEBOUNCE ตัวเก็บประจุ 1μF ที่ขา 5 ทำหน้าที่กำหนดเวลาของพัลส์ที่ขา DA

3.5 ภาคอินฟราเรดเซ็นเซอร์ (INFRARED SENSOR)

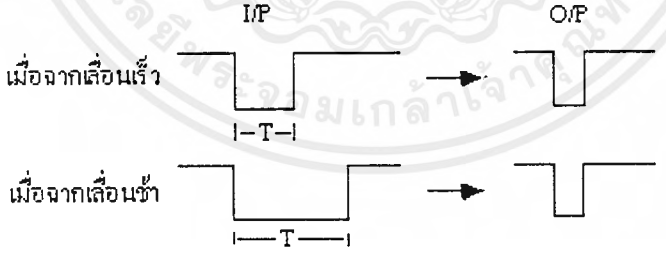


รูปที่ 3.4 วงจรภาค อินฟราเรดเซ็นเซอร์

จากรูปที่ 3.4 จะมีทั้งตัวส่งและตัวรับ ซึ่งจะเป็นโมดูล LED อินฟราเรด เมื่อเอาฉากที่ทำขึ้นมาโดยเฉพาะ โดยจะมีลักษณะเป็นซี่ๆ คือทึบแสงและโปร่งแสง โดยถ้าส่วนรับแสงของโมดูลนั้นโดนบังด้วยส่วนทึบแสง ที่ตัวโมดูลก็จะส่งสัญญาณระดับลอจิก “0” ออกมา และเมื่อไม่โดนบังแสงก็จะให้ระดับลอจิก “1” ออกมา โดยระดับลอจิกนี้จะส่งเข้าไอซีนอตเกท (not gate) และมี R,C ต่อ

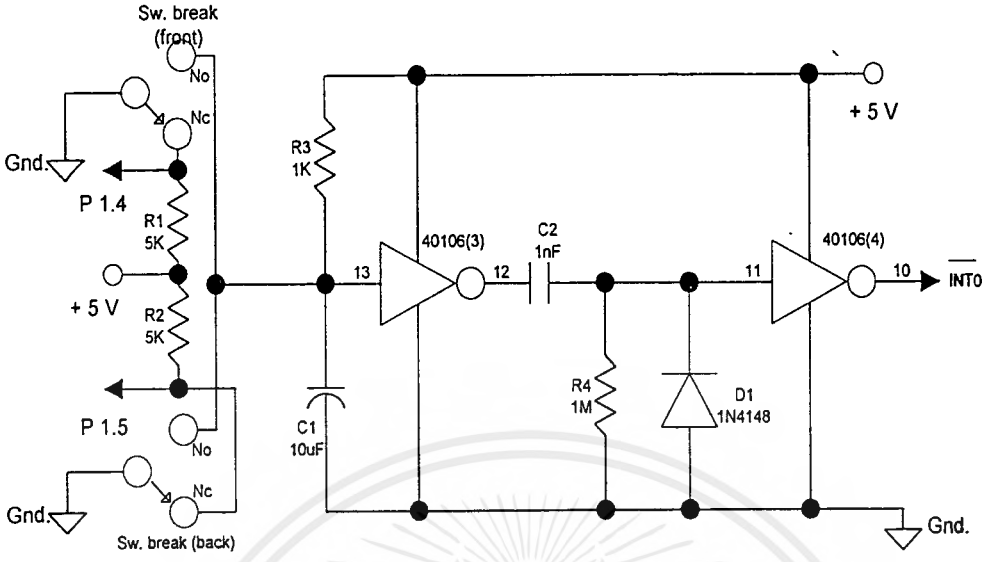
ร่วมกันที่ขาอินพุตของไอซีด้วย ซึ่งจะทำงานร่วมกันเป็นตัวแก้สัญญาณการกระเด้ง (bounce) ที่เกิดจากการสั่นของฉากที่ตัดแสงตัว โมดูลอินฟราเรด และค่า R นั่นคือค่า 5 KΩ ส่วนค่า C นั้นจะมีถึง 2 ตัว คือ 10 μF และ 100 μF ซึ่ง C ค่า 10 μF จะแก้การสั่นเมื่อหลอดฉีดยาเคลื่อนที่ไม่ช้ามากนัก ส่วน C ค่า 100 μF จะแก้การสั่นเมื่อหลอดฉีดยาเคลื่อนที่ช้ามากๆ ซึ่งจะได้สัญญาณ bounce ที่ไม่เหมือนกัน และเมื่อได้สัญญาณพัลส์เกิดเป็นพัลส์ 1 ลูก ป้อนเข้าวงจรฮาฟโมโนสเตเบิล (Half Monostable) ซึ่งเป็นวงจรที่จะทริกขอบขาขึ้นของพัลส์ แล้วให้พัลส์ ออกเอาท์พุท 1 ลูกเท่านั้น ตราบเท่าที่ยังไม่มีการทริกด้วยขอบขาขึ้น และเมื่อมีพัลส์ที่เป็นเฉพาะขอบขาขึ้นเท่านั้นเข้ามาอีก 1 ลูก ก็จะให้พัลส์ออกไปอีก 1 ลูก นั่นเอง ซึ่งวงจรจะประกอบได้ด้วย C1 1μF ต่อกับ VR 1MΩ ทำหน้าที่เป็น RC Differential ซึ่งมีเวลาเก็บและคายประจุตามค่าของ R,C ส่วน Diode 1N4148 ทำหน้าที่ป้องกันสัญญาณ I/P ที่จะเป็นช่วงลบเข้า IC40106(2) ที่ขา 3 และ IC40106(2) ขา 3,4 ทำหน้าที่ตรวจระดับสัญญาณ เพื่อสร้างเป็น pulse ที่มีคาบเวลาแน่นอน ออกไป

ที่ต้องใช้วงจรทริกขอบขาขึ้นของพัลส์นั้นเนื่องจากว่า เครื่องต้นแบบนั้นได้ออกแบบให้มีฉากกั้น เพื่อตัดแสงที่มีความถี่ช่วงตัดแสงคงที่ แต่จะเลื่อนมาตัดแสงด้วยเวลาไม่คงที่ กล่าวคือเครื่องต้นแบบนี้จะดันเข็มฉีดยาช้าหรือเร็ว เราสามารถตั้งได้ ดังนั้นจะมีพัลส์ ที่ป้อนเข้าวงจรทริกขอบพัลส์ (trig edge) นั้นจะมีช่วงเวลาไม่เท่ากัน ดังรูปที่ 3.5 ซึ่งเป็นตัวอย่างของรูปคลื่นอินพุต และเอาต์พุตที่ได้ซึ่งจะเห็นว่ารูปคลื่น เอาต์พุตจะไม่เปลี่ยนแปลงตามช่วงเวลาของพัลส์อินพุตเลย ดังนั้นจึงทำให้พัลส์ที่ป้อนเข้า 89C51 เพื่อเช็คดูว่า เข็มฉีดยาเลื่อนไปได้เท่าไรแล้ว เพื่อความถูกต้องแม่นยำนั่นเอง



รูปที่ 3.5 แสดงรูปคลื่น I/P and O/P pulse

3.6 ภาค สวิทช์หยุดหลอด (SW. BREAK)



รูปที่ 3.6 รูปแสดงวงจรสวิทช์เบรค

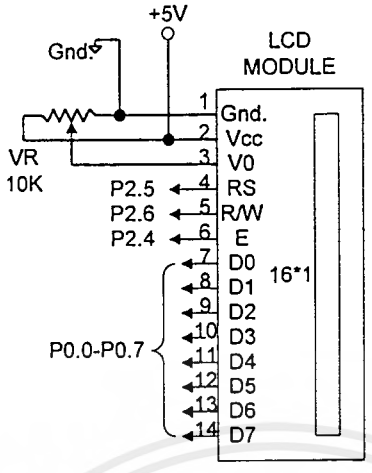
วงจรสวิทช์หยุดหลอดหรือสวิทช์เบรคในรูปที่ 3.6 ในเครื่องต้นแบบจริงจะใช้ 1 ชุด แต่จะใช้ สวิทช์ 2 ตัว ทั้งด้านหน้าและด้านหลัง ซึ่งจะมาต่อขนานกัน คือถ้าเข็มฉีดยาดันเข็มฉีดยาไปชนสวิทช์ข้างใดข้างหนึ่งแล้ว วงจรนี้ก็จะส่ง พัลส์ ไป อินเตอร์รัปต์ 89C51 เพื่อให้ทำการหยุดดันเข็มฉีดยาทันที

หลักการทำงานก็จะมีวงจร debounce ซึ่งอาศัย R และ C ต่อกันเป็น RC อินติเกรเตอร์และต่อร่วมกับนอตเกท (NOT GATE) เพื่อผลิตพัลส์ที่แน่นอนจากการกดสวิทช์ และเมื่อได้พัลส์จากการกดสวิทช์ ซึ่งจะเป็นขอบขาลง และทำการกลับสัญญาณให้เป็นขอบขาขึ้น ด้วย IC40106(3) ขา 13,12 แล้ว ก็จะป้อนเข้าวงจรฮาฟโมโนสเตเบิล(Half Monostable) หรือวงจร Edge Trigging เพื่อให้ได้พัลส์เพียงลูกเดียวไม่ว่าจะกดสวิทช์ ค้างไว้นานเท่าไหร่ก็ตาม

เนื่องจากสวิทช์เบรค จะต้องถูกกดค้าง ดังนั้นสัญญาณพัลส์ จะมีแค่ขอบของพัลส์ และระดับ ลอจิกที่คงที่ ดังนั้นจะต้องใช้วงจรทริกแค่ขอบแล้วได้สัญญาณพัลส์ 1 ลูก เอาไปอินเตอร์รัปต์ 89C51 เท่านั้น การอินเตอร์รัปต์เราจะใช้ขา INT0

ส่วน R1 และ R2 ค่า 5 KΩ นั้นจะทำหน้าที่เป็นจุดอ้างอิงให้ 89C51 ทำการตรวจเช็คว่าตอนนี้ ตัวดันหลอดฉีดยาอยู่ส่วนใดของหลอด โดยถ้าตัวดันหลอดอยู่สุดทางด้านท้าย จะทำให้ขา P1.5 เป็น “1” และถ้าตัวดันหลอดอยู่สุดทางด้านหน้า ก็จะทำให้ขา P1.4 เป็น “1” (ถ้าขา P1.5 เป็น “1” ขา P1.4 จะเป็น “0” แทน) แต่ถ้าตัวดันหลอดไม่อยู่สุดทั้งทางด้านหน้าและทางด้านหลัง ก็จะทำให้ทั้งขา P1.4 และขา P1.5 เป็น “0” ทั้งคู่

3.7 ภาคแสดงผล (DISPLAY LCD)



รูปที่ 3.7 รูปแสดงวงจรภาคแสดงผลด้วย LCD

เราจะใช้การแสดงผลการทำงานต่างๆบน LCD ซึ่ง LCD ที่ใช้จะเป็นแบบ dot matrix LCD module ขนาด 1 บรรทัด 16 ตัวอักษร ซึ่งการต่อ LCD จริงๆที่จะใช้กับเครื่องต้นแบบจะต่อผ่านพอร์ต P0.0-0.7 ของตัว 89C51 ส่วนการแสดงผลนั้นจะต้องใช้การเขียนโปรแกรมควบคุมทุกอย่าง และ VR 10 kΩ ทำหน้าที่เป็นตัวปรับความสว่างของหน้าจอ LCD

บทที่ 4

การสร้างและการออกแบบโปรแกรมควบคุม

เครื่องต้นแบบได้ถูกสร้างจากชิ้นงานหลายชิ้น แล้วค่อยนำมาประกอบเข้าด้วยกัน จึงจะแบ่งการอธิบายรายละเอียดเป็นส่วนๆ ดังนี้

4.1 ชิ้นส่วนภายนอก

จะประกอบไปด้วย

1. ตัวกล่อง เลือกใช้กล่องอเนกประสงค์หาซื้อได้ง่ายและราคาถูกกว่าพับประกอบเอง เป็นโลหะแข็งแรงพอและมีขนาดเหมาะสมกับปริมาตรโดยรวมของชิ้นส่วน ทั้งนี้เพื่อให้รองรับชิ้นส่วนทางกลและชิ้นส่วนอื่น ๆ ที่มีน้ำหนักมาก ทำให้ชิ้นส่วนต่าง ๆ สามารถยึดตรึงกับตัวกล่องอย่างแข็งแรง อย่างไรก็ดีทำให้น้ำหนักโดยรวมเพิ่มขึ้น แต่ยังคงทำให้สามารถเคลื่อนย้ายได้สะดวก

2. ตัวจับยึดฐานเข็มฉีดยาภายนอก ซึ่งได้ออกแบบไว้ให้จับได้เพียงขนาดเดียว คือ ขนาด 50 c.c. โดยเริ่มจากวัดขนาดหลอดของจริงที่หาซื้อมาได้ แล้วนำขนาดมาเปรียบเทียบกับตัวกล่อง เพื่อให้ขนาดที่เหมาะสมกับทั้งหลอด และตัวกล่องอีกที เมื่อได้ขนาดแล้วทำการออกแบบรูปทรงให้พอเหมาะกับขนาดที่วัดได้ เมื่อได้ทั้งรูปทรง หรือรูปแบบ และขนาดที่เหมาะสมแล้ว ก็นำแบบไปจ้างร้านอลูมิเนียมทำการตัด, พับ, ขึ้นรูป ตามแบบ เมื่อได้เป็นชิ้นงานแล้วก็นำมาตะไบให้พอเหมาะอีกทีแล้วนำเจาะรูยึดเข้ากับตัวกล่อง ส่วนตัวล็อคหลอดด้านบนนั้นจะใช้บานพับ และตัวจับยึดเฉพาะ

4.2 ชิ้นส่วนภายใน

จะประกอบไปด้วย

1. ในส่วนของแหล่งจ่ายไฟเนื่องจากมอเตอร์กินกระแสค่อนข้างสูงจึงไม่ได้สร้างแหล่งจ่ายไฟเอง แต่ใช้ชุดสำเร็จรูปแบบเชิงเส้นที่มีขนาดเล็กทำให้ลงกล่องอุปกรณ์ได้ง่าย

2. ชุดควบคุมที่ 1 ซึ่งเป็นชุดควบคุมหลักซึ่งจะมีสายต่าง ๆ เดินต่อไปยังไมโครสวิทช์ (MICRO SW) ทั้งหมด และสายแพที่จะต่อไปยังคีย์แพ็ด (KEY-PAD), เซ็นเซอร์ (SENSOR) และสายแพต่อไปยังชุดควบคุม มอเตอร์

3. ชุดควบคุมที่ 2 ซึ่งเป็นชุดควบคุมมอเตอร์นั่นเอง จะประกอบไปด้วยทรานซิสเตอร์ ที่เป็นชุด ขับกระแสสูงๆ จึงต้องยึดกับแผ่นระบายความร้อน ในเครื่องต้นแบบจะยึดติดกับฐานของเกลียวหนอนแทน

4. ชุดเฟือง ซึ่งจะประกอบไปด้วยมอเตอร์กับเฟือง ซึ่งใช้ทดรอบและทดแรงบิดของมอเตอร์ และทดความเร็วของการหมุนด้วย เพื่อป้องกันการกระตุกในการขับเป็น STEP อย่างช้า ๆ

เนื่องจากชุดเฟืองนี้ใช้เป็นของเก่ามาที่ยึดติดกันมาแล้วไม่ได้ออกแบบเอง จึงทำการคำนวณทางทฤษฎีในส่วนนี้ทำได้ลำบาก กระประกอบเพียงแต่นำมาจับยึดกับอลูมิเนียมเพื่อยึดตัวมอเตอร์กับเฟืองเข้ากับตัวกล่องอีกที

5. ชุดเกลียวหนอน ซึ่งจะประกอบไปด้วย ตัวยึดที่ดันเข็มฉีดยากับตัวเกลียวหนอน ซึ่งตัวเกลียวหนอนจะเป็นของเก่าที่หาซื้อมาได้อีกเช่นกัน โดยเราพยายามเลือกตัวที่เหมาะสมและคิดว่าใช้ได้ดีที่สุดมา โดยฐานเราจะได้ใช้อลูมิเนียมยึดโดยมีแบริ่งยึดเกลียวหนอนทั้ง 2 ข้าง ให้หมุนได้อิสระและต่อกับเฟืองที่มาจากมอเตอร์ เพื่อทำการจับให้หมุนเพื่อมอเตอร์หมุนเกลียวหนอนจะหมุนและตัวยึดที่ดันเข็มฉีดยาก็จะเลื่อนเข้าออกตามการหมุนทวนเข็มนาฬิกาของมอเตอร์และที่ฐานอลูมิเนียมก็จะต้องทำการติดตั้ ไมโครสวิทช์ ไว้ด้วย และยังมีเซ็นเซอร์อินฟราเรดที่ต้องติดตั้งเพื่อให้ตรงกับการเคลื่อนของเข็มด้วย

4.3 หลักการทำงานของโปรแกรมควบคุม

เพื่อให้สอดคล้องกับวัตถุประสงค์ของการใช้งาน และความสามารถในการรองรับของระบบทางกลซึ่งชิ้นส่วนใหญ่เป็นชุดสำเร็จรูปไม่ได้ออกแบบขึ้นมาใหม่ จึงออกแบบให้การทำงานของเครื่องต้นแบบนี้จะมีการทำงานอยู่ด้วยกัน 2 โหมด คือ

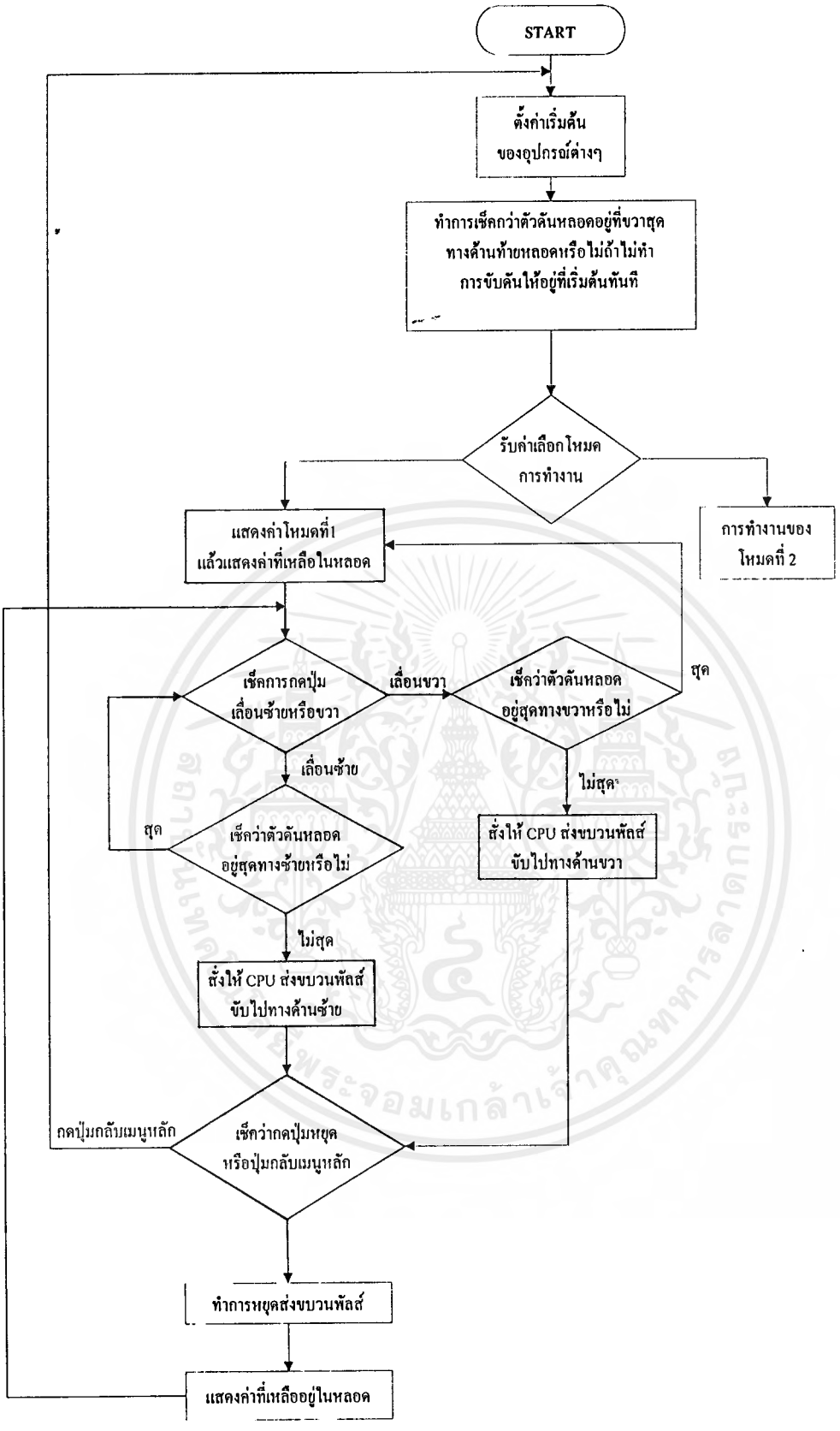
1. โหมด กำหนดเอง (manual mode)
2. โหมด กึ่งอัตโนมัติ (semi automatic mode)

4.3.1 โหมด กำหนดเอง

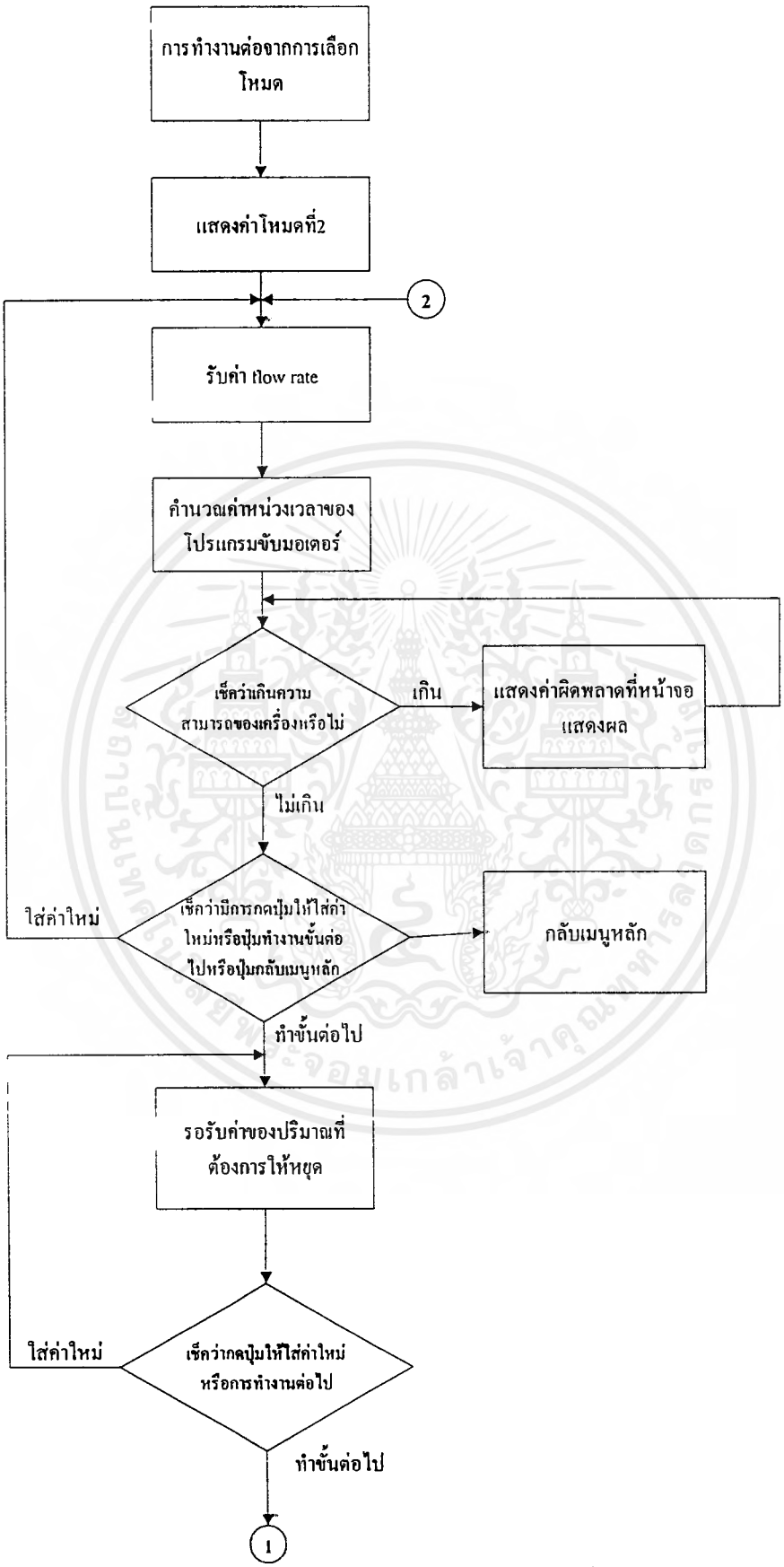
การทำงานในโหมดที่ 1 นั้น จะเป็นการควบคุมให้มีการดันชิ้นส่วนเคลื่อนที่ของหลอดฉีดยาไปข้างหน้าและไปข้างหลัง ด้วยความเร็วคงที่ หรือด้วยอัตราการไหล (flow rate) ที่คงที่ และถ้าหยุดการสั่งงานข้างต้นแล้ว (ด้วยการยกเลิกการกดคีย์) ตัวดันหลอดก็จะหยุดดันทันที โดยที่หน้าจอจะแสดงค่าของปริมาณของของเหลวที่คงเหลืออยู่ในหลอด ซึ่งจากการทำงานนี้นั้นสามารถนำมาเขียนโปรแกรม โดยมีแผนผัง (flow chart) ดังรูปที่ 4.1

4.3.2 โหมด กึ่งอัตโนมัติ

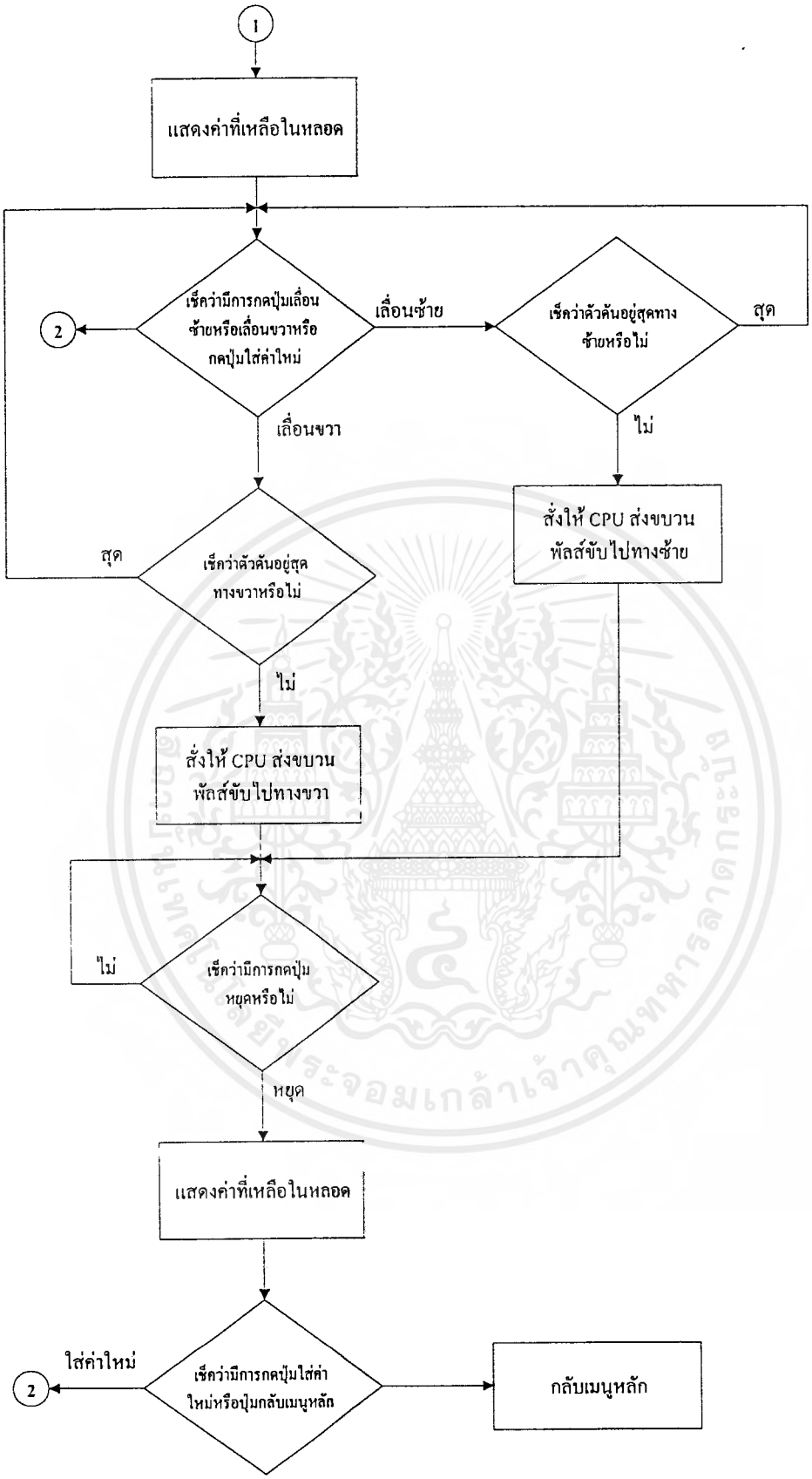
เป็นการทำงานโหมดที่ 2 โดยการทำงานในโหมดนี้นั้น สามารถที่จะตั้งค่าอัตราการไหล (flow rate) และค่าปริมาณที่ต้องการให้หยุดได้ เมื่อป้อนค่าที่ต้องการเรียบร้อยแล้ว และสั่งให้ดันหลอดแล้ว เครื่องก็จะคอยเช็คว่าครบค่าปริมาณที่ต้องการให้หยุด หรือมีการปุ่มให้หยุดหรือไม่ ถ้าครบค่าปริมาณหรือกดปุ่มให้หยุด เครื่องก็จะหยุดดันหลอดทันที แล้วก็จะแสดงค่าของปริมาณที่คงเหลือในหลอด และหลังจากนั้นก็จะทำการเช็คว่ามีการกดปุ่มให้ทำการดันหลอดต่อ หรือกดปุ่มให้มีการป้อนค่าใหม่หรือไม่ แล้วก็จะทำงานตามนั้น เมื่อได้ลำดับการทำงานนี้แล้ว นำมาเขียนโปรแกรมโดยมีผังงาน (flow chart) ดังรูปที่ 4.2



รูปที่ 4.1 แสดงผังงานในส่วนเริ่มต้นและส่วนของโหมดที่ 1



รูปที่ 4.2 แสดง ผังงานของโปรแกรมในโหมดที่ 2

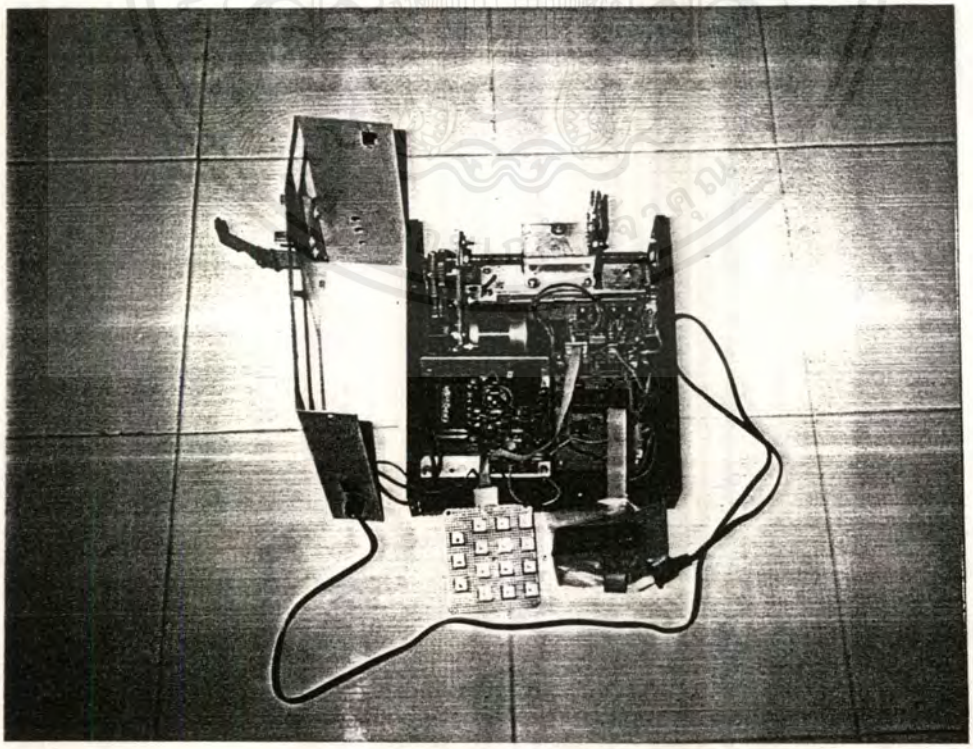


รูปที่ 4.2 แสดง ผังงานของโปรแกรมในโหมดที่ 2 (ต่อ)

4.4 รูปร่างของตัวเครื่องจริง

รูปที่ 4.3 และ 4.4 แสดงขนาดและรูปร่างภายในและภายนอกของเครื่องต้นแบบ

รูปที่ 4.3 รูปแสดงภายนอกของเครื่องต้นแบบจริง



รูปที่ 4.4 รูปแสดงภายในของเครื่องต้นแบบจริง

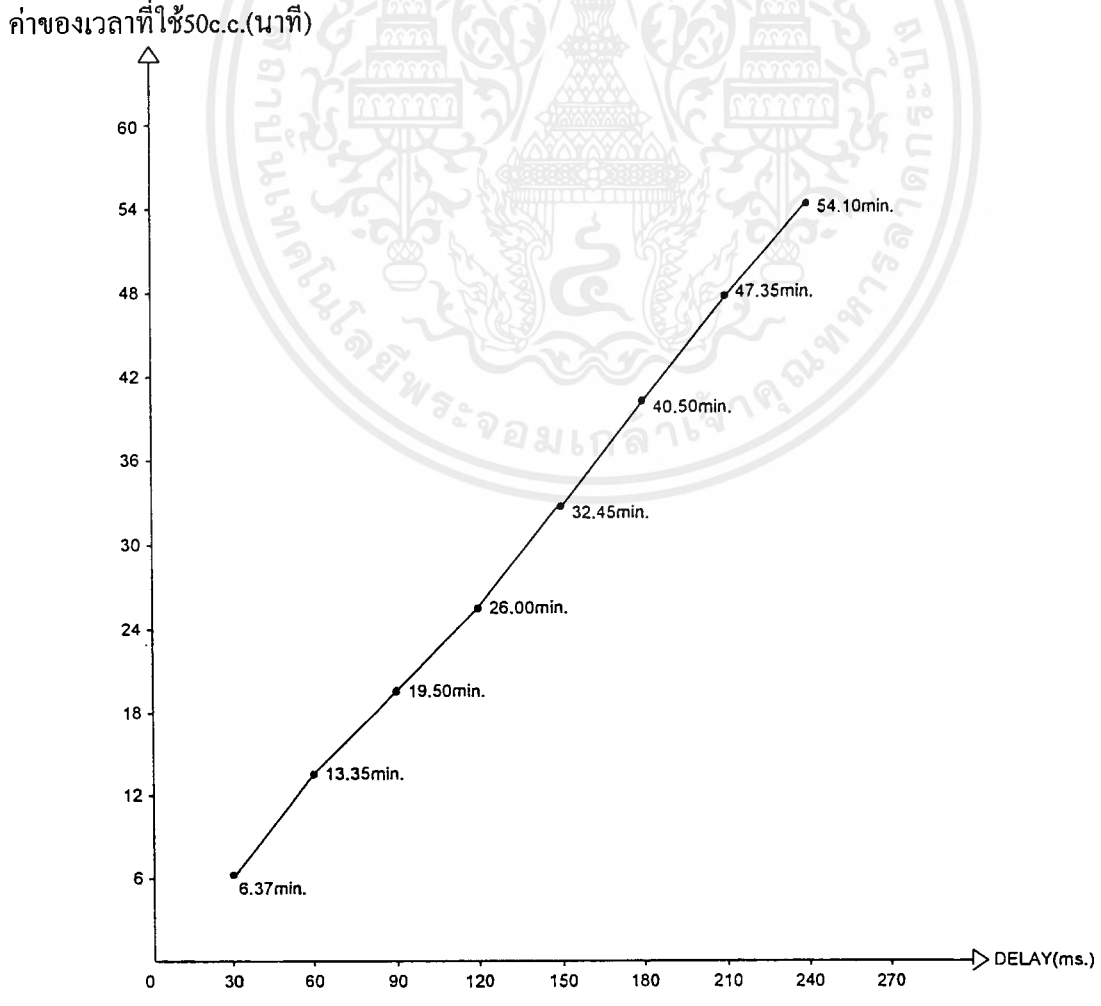
บทที่ 5

การทดลองและผลการทดลอง

ในบทนี้จะแสดงรายละเอียดการทดลองและผลที่ได้ในแต่ละชิ้นส่วนซึ่งจำเป็นต่อการนำมาใช้ในการปรับปรุงการทำงานรวมถึงการออกแบบในบางจุดของต้นแบบใหม่ นอกจากนี้การทดสอบการทำงานต่าง ๆ ของทั้งระบบทำให้ทราบข้อมูลทางเทคนิคของต้นแบบ

5.1 การทดสอบคุณสมบัติของมอเตอร์ เพื่อนำค่ามาเขียนโปรแกรมควบคุมความเร็วของการดันหลอดให้ได้ความเร็วตามต้องการ

ทดสอบโดยการจับเวลาของการดันที่ความเร็วต่างๆของมอเตอร์แล้วนำค่าเวลาที่ได้จากการจับเวลามาพล็อตกราฟ โดยให้แกนตั้งคือ แกนของเวลา และแกนนอนคือ แกนของความเร็วจะได้ผลตามรูปกราฟ 5.1



รูปที่ 5.1 กราฟแสดงคุณสมบัติของมอเตอร์

จากกราฟจะเห็นว่า ความเร็วของมอเตอร์จะมีลักษณะเป็นเชิงเส้น ซึ่งเราสามารถนำค่าความเป็นเชิงเส้นของมอเตอร์นี้มาเขียนโปรแกรมเพื่อควบคุมความเร็วในการดันหลอดฉีดยาได้ (ซึ่งในการเขียนโปรแกรมนั้นจะอาศัยความเป็นเชิงเส้นของมอเตอร์มาเป็นตัวแปรในการคำนวณหาค่าหน่วงเวลาในโปรแกรมเพื่อขับมอเตอร์ที่เวลาและความเร็วต่างๆกัน)

5.2 การทดสอบความถูกต้องของอัตราการไหลที่ตั้ง

เนื่องจากไม่มีเครื่องมือวัดโดยตรงได้ใช้การวัดทางอ้อมโดยวัดเวลาด้วยนาฬิกาที่อัตราการไหลต่าง ๆ แล้วนำค่าที่ได้มาเปรียบเทียบกับค่าเวลาที่ใช้จริง ซึ่งเป็นค่าเวลาที่ป้อนให้กับตัวเครื่อง เช่น เราป้อนค่าให้กับตัวเครื่อง 50 c.c./13 sec. แสดงว่าเราต้องการให้เครื่องดันของเหลวในหลอดฉีดยาด้วยปริมาณ 50 c.c. ใช้เวลาในการดัน 13 sec. ซึ่งเป็นเวลาจริงที่เราต้องการ แล้วก็จับเวลาที่ได้จากการดันหลอดฉีดยา ซึ่งเป็นค่าที่ได้จากการวัดแล้วนำมาเปรียบเทียบกัน แล้วคิดเป็นเปอร์เซ็นต์ผิดพลาดตามสูตร

$$\%\text{ ผิดพลาด} = \left(\left| \text{ค่าเวลาที่ใช้จริง} - \text{ค่าเวลาที่ได้จากการวัด} \right| / \text{ค่าเวลาที่ใช้จริง} \right) * 100$$

ซึ่งจะได้ผลการทดลองตามตารางที่ 5.1

ค่าที่ป้อน	ค่าจากการคำนวณ (50c.c.)	ค่าจากการวัด (50c.c.)	% คลาดเคลื่อน
50 c.c./13 sec.	13 sec.	12.7 sec.	2.3%
4 c.c./5 sec.	1.02 min.	59.6 sec.	4.6%
4 c.c./10 sec.	2.05 min.	2.11 min.	4.8%
4 c.c./15 sec.	3.07 min.	2.57 min.	5.6%
1 c.c./5 sec.	4.10 min.	3.58 min.	4.8%
4 c.c./25 sec.	5.12 min.	4.54 min.	5.7%
4 c.c./30 sec.	6.15 min.	5.57 min.	4.8%
4 c.c./35 sec.	7.17 min.	6.56 min.	4.8%
1 c.c./10 sec.	8.20 min.	7.53 min.	5.4%

4 c.c./45 sec.	9.22 min.	8.52 min.	5.3%
4 c.c./50 sec.	10.25 min.	9.53 min.	5.1%
4 c.c./55 sec.	11.27 min.	10.50 min.	5.4%
4 c.c./60 sec.	12.30 min.	11.55 min.	4.67%
2 c.c./50 sec.	2.05 min.	2.13 min.	6.4%
1 c.c./37 sec.	3.05 min.	3.17 min.	6.5%
1 c.c./50 sec.	4.10 min.	4.29 min.	7.6%
1 c.c./60 sec.	5.00 min.	5.12 min.	7.0%

ตารางที่ 5.1 แสดงแสดงค่าผิดพลาดของเวลาที่ใช้ในการดันหลอดปริมาตร 50 cc ที่อัตราไหลต่าง ๆ

จากผลการทดลองจะเห็นว่าค่าเวลาจริงที่เราต้องการและค่าเวลาที่ได้จากการจับเวลาจะมีค่าใกล้เคียงกันผิดพลาดเฉลี่ย 5.3 %

5.3 วัดปริมาณที่ได้จากการดันหลอดฉีดยา

ทดสอบโดยให้เครื่องดันของเหลวออกมาตามปริมาณที่ต้องการแล้วนำของเหลวนั้นไปวัดปริมาณแล้วนำค่าปริมาณนั้นมาเปรียบเทียบกับค่าที่ป้อนให้กับเครื่อง โดยใช้เครื่องตวงคือ กระบอกแก้วซึ่งมีความผิดพลาด $\pm$ 0.5 c.c.ซึ่งปริมาณที่ใช้ทดสอบจะมีค่าตามตาราง และทุกๆ ปริมาณที่วัดจะวัดจากปริมาณของเหลว 50 c.c.ซึ่งก็คือวัดจากของเหลวเต็มหลอด แล้วนำค่าที่ได้มาคิดเปอร์เซ็นต์ผิดพลาด ซึ่งใช้สูตรเดียวกับข้อ 5.2 โดยเปลี่ยนจากเวลาเป็นปริมาณ ซึ่งจะได้ผลตามตาราง 5.2 ค่าที่ได้จากตารางเป็นค่าที่ได้จากการทดสอบซึ่งใช้ความเร็วเดียวกันทุกปริมาณ ซึ่งใช้ในการทดสอบการทำงานของภาคเซ็นต์เซอร์ ว่าชุดเซ็นต์เซอร์ที่สร้างขึ้นมากับสเกลที่หลอดฉีดยา มีสเกลตรงกันหรือไม่ซึ่งจากการทดลองจะเห็นได้ว่าค่าที่ได้จะใกล้เคียงกัน หรือเท่ากันในบางค่าซึ่งแสดงให้เห็นว่า ตัวเซ็นต์เซอร์สามารถทำงานได้ ซึ่งค่าผิดพลาดสามารถยอมรับได้

ปริมาณที่ต้องการจริง	ปริมาณที่ได้จากการวัด	% คลาดเคลื่อน
5 c.c.	4.9 c.c.	2%
10 c.c.	10 c.c.	-
15 c.c.	14.9 c.c.	2%
20 c.c.	20 c.c.	-
25 c.c.	24.5 c.c.	2%
30 c.c.	30 c.c.	-
35 c.c.	35 c.c.	-
40 c.c	40 c.c.	-
45 c.c.	45 c.c.	-
50 c.c.	49 c.c.	2%

ตารางที่ 5.2 ตารางแสดงค่าผิดพลาดของปริมาณ

5.4 ทดสอบความคงที่ของอัตราการไหลของหลอดฉีดยาเมื่อมีโหลด

เป็นการทดสอบเพื่อสังเกตุความคงที่ของอัตราการไหลที่ตั้งไว้เมื่อต้องดันของเหลวเข้าสู่ภาชนะที่มีความดันสูงกว่าบรรยากาศ เป็นการทดสอบอย่างคร่าว ๆ ใช้น้ำหนักของ ๆ เหลวแทนความดันภายในภาชนะ โดยใช้ของเหลวใส่ในภาชนะทรงกระบอกแล้วต่อสายมายังหลอดฉีดยาโดยในหลอดฉีดยาจะมีของเหลวอยู่เต็มหลอดคือ 50 c.c.แล้วใส่ของเหลวลงไปในภาชนะทรงกระบอกประมาณ 900 c.c.แล้วยกภาชนะทรงกระบอกให้สูงขึ้นจากตัวเครื่องตามตาราง แล้วทดลองให้หลอดฉีดยาดันของเหลวออกจากหลอดพร้อมกับจับเวลาไปด้วยแล้วนำเวลาในแต่ละความสูงมาเปรียบเทียบกันดูว่าเวลาคงที่หรือไม่ จะได้ผลตามตาราง

ภาชนะทรงกระบอก สูงจากตัวเครื่อง	เวลาที่ใช้ (50 c.c./13 sec.)	เวลาที่ใช้ (50 c.c./37 sec.)	เวลาที่ใช้ (50 c.c./1.25 min.)
No load	11.97 sec.	35.43 sec.	59.06 sec.
20 cm.	11.93 sec.	35.56 sec.	59.06 sec.
40 cm.	12.07 sec.	35.56 sec.	58.82 sec.
60 cm.	12.15 sec	35.56 sec.	58.95 sec.
80 cm.	12.31 sec	35.66 sec.	59.09 sec.
100 cm.	12.31 sec.	35.78 ec.	59.00 sec.

ตารางที่ 5.3 ตารางแสดงเวลาที่ใช้ในการดันของเหลวที่โหลดต่างๆ

จากตารางจะเห็นว่าที่ระดับความสูงต่างๆจะได้ค่าเวลาตามตารางซึ่งจะเห็นว่าค่าที่ได้จะมีค่าใกล้เคียงกัน แตกต่างกันเพียงเล็กน้อย แสดงให้เห็นว่าอัตราการไหลของของเหลวจะเข้าใกล้ค่าคงที่มากไม่ว่าโหลดจะเปลี่ยลแปลงก็ตาม และที่ความเร็วต่ำๆอัตราการไหลก็จะคงที่มากขึ้นเพราะที่ความเร็วต่ำแรงจะมากกว่าที่ความเร็วสูง ซึ่งค่าที่ได้จากตารางค่าผิดพลาดเกิดจากการจับเวลาด้วย

บทที่ 6
บทสรุปและวิจารณ์

ในปริญญานิพนธ์ฉบับนี้ได้กล่าวถึงความเป็นมาของโครงงาน วัตถุประสงค์และรายละเอียดการออกแบบสร้างชุดดันเข็มฉีดยาที่ได้กระทำตลอดภาคการศึกษานี้ ตลอดจนทฤษฎีเบื้องต้นที่เกี่ยวกับการออกแบบโครงงานชิ้นนี้

เนื่องจากในงานหลายๆงาน มีความจำเป็นที่จะต้องการควบคุมปริมาณที่แม่นยำ และอัตราการไหลของของเหลวให้คงที่และสม่ำเสมอแน่นอน จึงได้มีแนวความคิดในการออกแบบระบบโดยการนำไมโครคอนโทรลเลอร์มาควบคุมการทำงานทั้งหมด และออกแบบโครงสร้างภายในและภายนอกให้สามารถใช้งานจริงได้ โดยในการทำงานนี้นั้น จะยากที่สุดตรงเรื่องของเฟืองกับมอเตอร์ที่จะต้องหาให้ได้เหมาะสม และเป็นไปได้ที่สุดที่ซื้อมาแล้วจะใช้งานได้จริงและดีที่สุด เพราะถ้าจ้างทำเฟืองนั้นก็จะแพงมาก จึงต้องเลือกหาของเก่าแทน ส่วนการใช้งานของเครื่องต้นแบบนั้นถูกออกแบบให้ง่ายต่อการใช้มากที่สุด

จากการทดสอบสามารถขับเคลื่อนเข็มฉีดยาขนาด 50 ลบ.ซม. เดินหน้า,ถอยหลังและแบบที่สามารถควบคุมอัตราการไหลได้ในลักษณะลูปเปิด จากการทดสอบ พบว่าที่มีโหลด กับไม่มีโหลดนั้น เวลาในการดันหลอดจะใกล้เคียงกัน แสดงให้เห็นว่าเครื่องต้นแบบมีความสามารถพอที่จะขับให้มีความคงที่ ทั้งปริมาณและความเร็ว ค่าความเร็วสูงสุดที่วัดได้ประมาณ 50 ลบ.ซม. ต่อ 13 วินาที (230.8 ลบ.ซม./นาที) และค่าความเร็วต่ำสุดจะประมาณ 50 ลบ.ซม. ต่อ 50 นาที (1 ลบ.ซม./นาที) โดยถ้าตั้งค่าความเร็วนอกเหนือจากนี้ เครื่องต้นแบบจะไม่สามารถทำงานได้

และการทดสอบการทำงานในส่วนควบคุมอัตราการไหลนั้น ความสามารถของเครื่องต้นแบบนี้ สามารถทำงานได้ดีกล่าวคือ มีความผิดพลาดไม่มากนัก ประมาณ 2 % โดยค่าผิดพลาดนี้พิจารณาจากการวัดปริมาณของของเหลวที่ถูกฉีดออกมาจากหลอดฉีดยาเทียบกับถ้วยตวงมาตราฐานว่าผิดพลาดไปเท่าใด และในส่วนของข้อจำกัดนั้นคือยังไม่สามารถควบคุมผ่านคอมพิวเตอร์ส่วนบุคคลได้ (PC) คือถ้าเครื่องต้นแบบสามารถควบคุมการทำงานผ่านเครื่อง PC ได้ ก็สามารถทำให้การใช้งานนั้นคล่องตัวมากขึ้น และสามารถเก็บบันทึกค่าต่างๆได้

จากการทดลองและการทำงานของโครงงานชิ้นนี้ ทำให้เราได้มีประสบการณ์ในการออกแบบ สร้างวงจร และได้นำวิชาที่เรียนในภาคทฤษฎีมาปฏิบัติจริง รวมทั้งเรียนรู้และฝึกฝนการแก้ปัญหาในด้านการศึกษาค้นคว้าด้วยตนเอง และบางอย่างที่เรียนแล้วแต่ไม่ค่อยได้ใช้งาน ไม่ได้ทดลองทำจริงมากนัก เมื่อลองมาทำจริงแล้วจึงคิด ๆ ขัด ๆ บางเรื่อง และได้ทำการค้นคว้าทดลองจนออกมาเป็นวงจรในโครงงานนี้

ในโครงงานนี้แม้ยังมีข้อจำกัด และค่าความผิดพลาดอยู่ แต่เชื่อว่าผลการศึกษาทดลองนี้ น่าจะเป็นแนวทางในการพัฒนาให้เครื่องต้นแบบนี้ มีข้อผิดพลาดและข้อจำกัดน้อยลง และมีความสามารถมากขึ้น ซึ่งในปัจจุบันนี้ในท้องตลาดที่มีขายเครื่องในลักษณะนี้นั้น ยังเป็นเครื่องที่ผลิตในต่างประเทศอยู่และมีราคาแพงมาก ซึ่งถ้าเราออกแบบและผลิตใช้เองได้นั้นก็จะเป็นประโยชน์มากในอนาคตต่อไป

ภาคผนวก ก

โปรแกรมควบคุมภาษาแอสเซมบลี

```
;***********************************
;* SOFTWARE FOR SYRINGE PUMP       *
;*        Program control for 89C51 *
;                 (a & te 3R/2)     *
;***********************************
BUF_SWBREAK     EQU     20H
BUF_SCANKEY     EQU     21H
BUF_COUNTER     EQU     22H
DATA_COUNTER    EQU     23H
LOOP1           EQU     24H
LOOP2           EQU     25H
BUF_ASCII       EQU     26H
BUF_DATABREAK   EQU     27H
LOOP3           EQU     28H
LOOP4           EQU     29H
BUF_AUTO1       EQU     2AH
BUF_AUTO2       EQU     2BH
BUF_AUTO3       EQU     2CH
BUF_AUTO4       EQU     2DH
BUF_STOP        EQU     2EH
BUF_AUTO2H      EQU     2FH
BUF_AUTO2L      EQU     30H
BUF_DIVIED1     EQU     31H
BUF_DIVIED2     EQU     32H
DATA_DELAY      EQU     33H
LOOP5           EQU     34H
LOOP6           EQU     35H
DATA_COUNTER1   EQU     36H
BUF_STOP1       EQU     37H
BUF_SW          EQU     38H
DATA_STOPAT     EQU     39H
BUF_STOPAT      EQU     3AH
```

```
BUF_STOPAT1     EQU     3BH
BUF_STOPAT2     EQU     3CH
BUF_ERROR1      EQU     3DH
BUF_ERROR2      EQU     3EH
BUF_ERROR3      EQU     3FH
BUF_ERROR4      EQU     40H

                ORG     0000H
                LJMP    START

                ORG     0003H
                LJMP    SW_BREAK        ;INT0 FOR SW_BREAK

                ORG     0013H
                LJMP    SCAN_KEY        ;INT1 FOR SCAN_KEY

                ORG     000BH
                LJMP    COUNTER         ;INT. TIMER0 FOR COUNTER

                ORG     0030H
START:          MOV     SP,#50H
                MOV     TMOD,#00000110B ;Set T0 mode2 counter
                SETB    EA              ;Set enable global int.
                SETB    IT0             ;Set falling edge type for INT0
                SETB    IT1             ;Set falling edge type for INT1
                SETB    EX0             ;Enable External INT0
                SETB    EX1             ;Enable External INT1
                SETB    ET0             ;Enable interrupt timer0 for counter
                MOV     IP,#00000111B
                MOV     TL0,#0FFH
                MOV     TH0,#0FFH
                SETB    TR0
```

```
                MOV   P1,#0
                CLR   P2.5
                CLR   P2.6
                CLR   P2.4
                MOV   BUF_ASCII,#38H      ;SET LCD
                LCALL WRCOMMAND
                MOV   BUF_ASCII,#0FH      ;SET LCD
                LCALL WRCOMMAND
                MOV   BUF_ASCII,#6        ;SET LCD
                LCALL WRCOMMAND

                MOV   DPTR,#TAB1
                MOV   R1,#16
                LCALL LINE
                LCALL DELAY1S

INITIAL:        MOV   DPTR,#TAB9
                MOV   R1,#16
                LCALL LINE
                MOV   BUF_SWBREAK,#0      ;First time,set up to 50 c.c.
                CLR   ET0                 ;Disable display
                SETB  P1.5
                MOV   A,P1
                JNB   ACC.5,INI           ;If not 50 c.c. to set up reverse
MAIN_INI:       SETB  ET0                 ;Back to enable display again

MAIN2:          MOV   DPTR,#TAB3
                MOV   R1,#13
                LCALL LINE

MAIN:           MOV   P1,#0
                MOV   BUF_SCANKEY,#0
```

```
        MOV   BUF_DATABREAK,#0
        MOV   BUF_SWBREAK,#0
        MOV   BUF_AUTO1,#0
        MOV   BUF_AUTO2,#0
        MOV   BUF_AUTO3,#0
        MOV   BUF_AUTO4,#0
        MOV   BUF_DIVIED1,#0
        MOV   BUF_DIVIED2,#0
        MOV   BUF_ASCII,#0C5H
        LCALL  WRCOMMAND
        MOV   BUF_ASCII,#20H      ;show " " when back this loop again
        LCALL  WRITE
        MOV   BUF_ASCII,#0C5H
        LCALL  WRCOMMAND
MAIN1:  MOV   A,BUF_SCANKEY
        JZ    MAIN1
        CJNE  A,#1,$+6            ;If press number 1 to ONE
        LJMP  ONE
        CJNE  A,#2,$+6            ;If press number 2 to TWO
        LJMP  TWO
        SJMP  MAIN1               ;If press other buttoms,return to main1

INI:    LJMP  INI_REV

ONE:    MOV   BUF_ASCII,#31H      ;SHOW "1"
        LCALL  WRITE
ONE3:   MOV   A,BUF_SCANKEY
ONE1:   CJNE  A,#13,ONE2
        LJMP  MAIN
ONE2:   CJNE  A,#14,ONE3
        MOV   DPTR,#TAB4
        MOV   R1,#16
```

```
                LCALL  LINE
                LCALL  DELAY1S
                MOV    DPTR,#TAB6
                MOV    R1,#16
                LCALL  LINE
                MOV    BUF_ASCII,#0C1H
                LCALL  WRCOMMAND
                MOV    BUF_ASCII,#35H        ;SHOW "5"
                LCALL  WRITE
                MOV    BUF_ASCII,#30H        ;SHOW "0"
                LCALL  WRITE
MAIN_ONE:       MOV    BUF_SCANKEY,#0        ;Initial value
                MOV    BUF_DATABREAK,#0
                MOV    BUF_SWBREAK,#0
                MOV    DATA_COUNTER,#50
                MOV    IP,#00000010B         ;TIMER0 HIGH IP.
                CLR    P3.6
                CLR    IT1
MAIN_ONE1:      MOV    P1,#0
                MOV    A,BUF_SCANKEY
                CJNE   A,#13,MAIN_ONE1
                SETB   IT1
                MOV    IP,#00000111B
                LJMP   INITIAL

MUNUAL:         PUSH   PSW
                MOV    A,BUF_SWBREAK
                JNZ    CHECK
DRIVE:          MOV    A,BUF_SCANKEY
                CJNE   A,#10,$+15            ;If press number 10 to forward
                SETB   P1.4                  ;Protect when press not true
                MOV    A,P1
```

```
        JB    ACC.4,$+6
        LJMP  FORWARD
        MOV   A,BUF_SCANKEY
        CJNE  A,#11,$+13          ;If press number 11 to reverse
        SETB  P1.5                ;Protect when press not true
        MOV   A,P1
        JB    ACC.5,$+6
        LJMP  REVERSE
        POP   PSW
        LJMP  RELEASE             ;If press other buttoms,return main

CHECK:  MOV   A,BUF_DATABREAK
        CJNE  A,BUF_SCANKEY,DRIVE
        POP   PSW
        LJMP  RELEASE

TWO:    MOV   P1,#0
        MOV   BUF_ASCII,#32H      ;SHOW "2"
        LCALL WRITE
TWO3:   MOV   A,BUF_SCANKEY
TWO1:   CJNE  A,#13,TWO2
        LJMP  MAIN
TWO2:   CJNE  A,#14,TWO3
        MOV   DPTR,#TAB5
        MOV   R1,#16
        LCALL LINE
        LCALL DELAY1S
        MOV   BUF_SWBREAK,#0
TWO5:   MOV   DPTR,#TAB7
        MOV   R1,#12
        LCALL LINE
        MOV   P1,#0
```

```
                MOV   BUF_ASCII,#85H
                LCALL  WRCOMMAND
                MOV   BUF_ASCII,#20H      ;show " " when back this loop again
                LCALL  WRITE
                MOV   BUF_ASCII,#85H
                LCALL  WRCOMMAND
                MOV   DATA_DELAY,#0
                MOV   R4,#2
TWOE:           MOV   BUF_AUTO1,BUF_AUTO2
TWO6:           MOV   BUF_SCANKEY,#10
TWO4:           MOV   A,BUF_SCANKEY
                CJNE  A,#10,$+5
                SJMP  TWO4
                CJNE  A,#11,$+5
                SJMP  TWO4
                CJNE  A,#12,$+6
                LJMP  INITIAL
                CJNE  A,#14,$+5
                SJMP  TWO4
                CJNE  A,#13,$+5
                SJMP  TWO5
                MOV   BUF_AUTO2,A
                ADD   A,#30H
                MOV   BUF_ASCII,A
                LCALL  WRITE
                LCALL  DELAY1mS
                DJNZ  R4,TWOE
                MOV   A,BUF_AUTO1
                MOV   B,#10
                MUL   AB
                ADD   A,BUF_AUTO2
                CJNE  A,#51,$+5
```

```
                SJMP   TWO5
                JNC    TWO5
                MOV    BUF_AUTO1,A

TWO11:          MOV    BUF_ASCII,#0C4H
                LCALL  WRCOMMAND
                MOV    BUF_ASCII,#20H
                LCALL  WRITE
                MOV    BUF_ASCII,#0C5H
                LCALL  WRCOMMAND
                MOV    BUF_ASCII,#20H
                LCALL  WRITE
                MOV    BUF_ASCII,#0C4H
                LCALL  WRCOMMAND
                MOV    R4,#2
TWO8:           MOV    BUF_SCANKEY,#10
TWO9:           MOV    A,BUF_SCANKEY
                CJNE   A,#10,$+5
                SJMP   TWO9
                CJNE   A,#11,$+5
                SJMP   TWO9
                CJNE   A,#12,$+5
                SJMP   TWO9
                CJNE   A,#14,$+5
                SJMP   TWO9
                CJNE   A,#13,$+5
                SJMP   TWO11
                MOV    BUF_AUTO2,A
                ADD    A,#30H
                MOV    BUF_ASCII,A
                LCALL  WRITE
                LCALL  DELAY1mS
```

```
        DJNZ   R4,TWO10
        MOV    A,B
        MOV    B,#10
        MUL    AB
        ADD    A,BUF_AUTO2
        CJNE   A,#61,$+5
        SJMP   TWO11
        JNC    TWO11
        MOV    BUF_AUTO2,A

        MOV    BUF_ASCII,#0C7H
        LCALL  WRCOMMAND
        MOV    BUF_ASCII,#73H      ;"sec"
        LCALL  WRITE
        MOV    BUF_AUTO3,#73H
M:      MOV    A,BUF_SCANKEY
        CJNE   A,#12,$+5
        SJMP   UNIT0
        CJNE   A,#13,$+6
        LJMP   TWO5
        CJNE   A,#14,M
        SJMP   ENTER

TWO10:  MOV    A,BUF_AUTO2
        MOV    B,A
        LJMP   TWO8

UNIT0:  MOV    A,BUF_AUTO3
        CJNE   A,#73H,UNIT1
        MOV    BUF_ASCII,#0C7H
        LCALL  WRCOMMAND
        MOV    BUF_ASCII,#6DH      ;"min"
```

```
                LCALL  WRITE
                MOV    BUF_SCANKEY,#0
                MOV    BUF_AUTO3,#6DH
                SJMP   M

UNIT1:          MOV    BUF_ASCII,#0C7H
                LCALL  WRCOMMAND
                MOV    BUF_ASCII,#73H       ;"sec"
                LCALL  WRITE
                MOV    BUF_SCANKEY,#0
                MOV    BUF_AUTO3,#73H
                SJMP   M

ENTER:          MOV    A,BUF_AUTO2
                MOV    B,#4                 ;CONSTANT=4
                MUL    AB
                MOV    BUF_AUTO2L,A
                MOV    A,BUF_AUTO3
                CJNE   A,#6DH,ENTER1
                MOV    A,BUF_AUTO2L
                MOV    B,#60
                MUL    AB
                MOV    BUF_AUTO2H,B
                MOV    BUF_AUTO2L,A
                LCALL  CHK_ERROR
                MOV    A,BUF_AUTO2L
                MOV    B,BUF_AUTO1
                DIV    AB
                MOV    BUF_DIVIED1,A
                MOV    BUF_DIVIED2,B
DIVI:           MOV    A,BUF_AUTO2H
                JZ     NON3                 ;JMP IF A=0
```

```
                DEC   BUF_AUTO2H
                MOV   A,#0FFH
                MOV   B,BUF_AUTO1
                DIV   AB
                ADD   A,BUF_DIVIED1
                MOV   BUF_DIVIED1,A
                MOV   A,B
                ADD   A,#1
                ADD   A,BUF_DIVIED2
                MOV   B,BUF_AUTO1
                DIV   AB
                ADD   A,BUF_DIVIED1
                MOV   BUF_DIVIED1,A
                MOV   BUF_DIVIED2,B
                MOV   BUF_AUTO4,BUF_DIVIED1
                SJMP  DIVI

ENTER1:         MOV   A,BUF_AUTO2L
                MOV   B,BUF_AUTO1
                DIV   AB
                MOV   BUF_AUTO4,A
                SJMP  NON

NON3:           MOV   A,BUF_AUTO1
                CJNE  A,BUF_ERROR4,CHK_ERROR1
NON:            MOV   A,BUF_AUTO4
                JZ    NON1              ;IF A=0 JUMP
                CJNE  A,#241,NON2
NON1:           MOV   DPTR,#TAB8
                MOV   R1,#16
                LCALL LINE
                LCALL DELAY1S
```

```
                LJMP   TWO5

CHK_ERROR1:     JNC    NON
                SJMP   NON1

NON2:           JNC    NON1            ;IF C=0 JUMP
                MOV    P1,#0
                MOV    DPTR,#TAB10
                MOV    R1,#16
                LCALL  LINE
STOP_AT:        MOV    BUF_ASCII,#0C1H
                LCALL  WRCOMMAND
                MOV    BUF_ASCII,#20H
                LCALL  WRITE
                MOV    BUF_ASCII,#0C2H
                LCALL  WRCOMMAND
                MOV    BUF_ASCII,#20H
                LCALL  WRITE
                MOV    BUF_ASCII,#0C1H
                LCALL  WRCOMMAND
                MOV    DATA_STOPAT,#0
                MOV    BUF_STOPAT2,#0
                MOV    R4,#2
STOP_AT2:       MOV    BUF_STOPAT1,BUF_STOPAT2
                MOV    BUF_SCANKEY,#10
STOP_AT1:       MOV    A,BUF_SCANKEY
                CJNE   A,#10,$+5
                SJMP   STOP_AT1
                CJNE   A,#11,$+5
                SJMP   STOP_AT1
                CJNE   A,#12,$+6
                LJMP   INITIAL
```

```
                CJNE   A,#14,$+5
                SJMP   STOP_AT1
                CJNE   A,#13,$+5
                SJMP   STOP_AT
                MOV    BUF_STOPAT2,A
                ADD    A,#30H
                MOV    BUF_ASCII,A
                LCALL  WRITE
                DJNZ   R4,STOP_AT2
                MOV    A,BUF_STOPAT1
                MOV    B,#10
                MUL    AB
                ADD    A,BUF_STOPAT2
                MOV    DATA_STOPAT,A
STOP_AT3:       MOV    A,BUF_SCANKEY
                CJNE   A,#14,$+6
                LJMP   $+15
                CJNE   A,#12,$+6
                LJMP   INITIAL
                CJNE   A,#13,STOP_AT3
                LJMP   STOP_AT
                MOV    DPTR,#TAB6
                MOV    R1,#16
                LCALL  LINE
                MOV    BUF_STOP,BUF_SWBREAK
                MOV    BUF_ASCII,#0C1H
                LCALL  WRCOMMAND
                MOV    BUF_ASCII,#35H        ;SHOW "5"
                LCALL  WRITE
                MOV    BUF_ASCII,#30H        ;SHOW "0"
                LCALL  WRITE
                MOV    DATA_COUNTER,#50
```

```
                MOV   A,BUF_STOP
                CJNE  A,#1,CHON1
OUT_LCD1:       MOV   BUF_ASCII,#0C1H
                LCALL WRCOMMAND
                MOV   A,DATA_COUNTER1
                CJNE  A,#10,NOCHANGE1
                SJMP  CHANGE1
NOCHANGE1:      JC    ASCZ1
CHANGE1:        MOV   B,#0AH
                DIV   AB
                ADD   A,#30H
                MOV   BUF_ASCII,A
                LCALL WRITE
                MOV   A,B
                ADD   A,#30H
                MOV   BUF_ASCII,A
                LCALL WRITE
CHON:           MOV   DATA_COUNTER,DATA_COUNTER1
CHON1:          MOV   BUF_DATABREAK,#0
                MOV   BUF_SCANKEY,#0
                MOV   BUF_STOP1,#0
WAITGO:         MOV   DATA_DELAY,BUF_AUTO4
                MOV   A,DATA_DELAY
                CJNE  A,#10,$+6
                LJMP  RY100U
                JC    RY10U
RY100U:         SETB  P3.6
RELAY:          MOV   P1,#0
                MOV   BUF_STOP,BUF_SWBREAK
                MOV   BUF_SCANKEY,#0
                MOV   BUF_COUNTER,DATA_STOPAT
WAITGO1:        MOV   A,BUF_SCANKEY
```

```
                CJNE   A,#10,$+15
                SETB   P1.4            ;Protect when press not true
                MOV    A,P1
                JB     ACC.4,$+6
                LJMP   FORWARDA
                MOV    A,BUF_SCANKEY
                CJNE   A,#11,$+13
                SETB   P1.5            ;Protect when press not true
                MOV    A,P1
                JB     ACC.5,$+6
                LJMP   REVERSEA
                CJNE   A,#13,WAITGO1
                LJMP   TWO5

ASCZ1:          ADD    A,#30H
                MOV    BUF_ASCII,A
                LCALL  WRITE
                MOV    BUF_ASCII,#20H      ;SHOW " "
                LCALL  WRITE
                SJMP   CHON

RY10U:          CLR    P3.6
                SJMP   RELAY

SCAN_KEY:       PUSH   ACC
                PUSH   PSW
                MOV    P2,#0FH             ;SET P2.0-P2.3 IS I/P DATA
                MOV    A,P2
                ANL    A,#00001111B
                MOV    BUF_SCANKEY,A
                JNB    IT1,MUNUAL1         ;IF IT0=0 JUMP
RELEASE:        POP    PSW
```

```
                POP   ACC
                RETI

MUNUAL1:        LJMP  MUNUAL

FORWARD:        MOV   R3,#8
                MOV   DPTR,#TABF    ;loop for drive motor forward
STEPF:          MOV   A,#00H
                MOVC  A,@A+DPTR
                JZ    FORWARD
                MOV   P1,A
                ACALL DELAY1ms      ;delay of step
                INC   DPTR
                DJNZ  R3,STEPF
                MOV   BUF_DATABREAK,#10
                MOV   BUF_SWBREAK,#0
                POP   PSW
                SJMP  RELEASE

REVERSE:        MOV   R3,#8
                MOV   DPTR,#TABR    ;loop for drive motor reverse
STEPR:          MOV   A,#00H
                MOVC  A,@A+DPTR
                JZ    REVERSE
                MOV   P1,A
                ACALL DELAY1ms      ;delay of step
                INC   DPTR
                DJNZ  R3,STEPR
                MOV   BUF_DATABREAK,#11
                MOV   BUF_SWBREAK,#0
                POP   PSW
                SJMP  RELEASE
```

```
COUNTER:    PUSH   PSW
            PUSH   ACC
            MOV    A,BUF_DATABREAK
            CJNE   A,#10,INC_DATA
            DEC    DATA_COUNTER
            DEC    BUF_COUNTER
            MOV    A,BUF_COUNTER
            CJNE   A,#0,$+9
            MOV    BUF_SWBREAK,#01H
            MOV    BUF_STOP1,#01H
OUT_LCD:    MOV    BUF_ASCII,#0C1H
            LCALL  WRCOMMAND
            MOV    A,DATA_COUNTER
            CJNE   A,#10,NOCHANGE
            SJMP   CHANGE
NOCHANGE:   JC     ASCZ
CHANGE:     MOV    B,#0AH
            DIV    AB
            ADD    A,#30H
            MOV    BUF_ASCII,A
            LCALL  WRITE
            MOV    A,B
            ADD    A,#30H
            MOV    BUF_ASCII,A
            LCALL  WRITE
            POP    ACC
            POP    PSW
            RETI

INC_DATA:   INC    DATA_COUNTER
            SJMP   OUT_LCD
```

```
ASCZ:       ADD   A,#30H
            MOV   BUF_ASCII,A
            LCALL WRITE
            MOV   BUF_ASCII,#20H      ;SHOW " "
            LCALL WRITE
            POP   ACC
            POP   PSW
            RETI

LINE:       MOV   BUF_ASCII,#1        ;CLEAR DISPLAY
            LCALL WRCOMMAND
            MOV   R0,#8
LINE1:      MOV   A,#0
            MOVC  A,@A+DPTR
            MOV   BUF_ASCII,A
            LCALL WRITE
            INC   DPTR
            DEC   R1
            DJNZ  R0,LINE1
            CJNE  R1,#0,LINE2
OUTLINE:    RET
LINE2:      MOV   BUF_ASCII,#0C0H     ;NEW LINE
            LCALL WRCOMMAND
STR1:       MOV   A,#0
            MOVC  A,@A+DPTR
            MOV   BUF_ASCII,A
            LCALL WRITE
            INC   DPTR
            DJNZ  R1,STR1
            SJMP  OUTLINE
```

```
FORWARDA:       MOV   A,BUF_STOP
                JNZ   AUTO_CHECK1
FOR:            MOV   BUF_SWBREAK,#0
FORWARD1:       MOV   DPTR,#TABF     ;loop for drive motor forward
STEPF1:         MOV   A,#00H
                MOVC  A,@A+DPTR
                JZ    FORWARD1
                MOV   P1,A
                ACALL DELAYA         ;delay of step
                INC   DPTR
                MOV   BUF_DATABREAK,#10
                MOV   A,BUF_SWBREAK
                JNZ   TWOAUTO        ;If A=1 JUMP
                SJMP  STEPF1

REVERSEA:       MOV   A,BUF_STOP
                JNZ   AUTO_CHECK2
REV:            MOV   BUF_SWBREAK,#0
REVERSE1:       MOV   DPTR,#TABR     ;loop for drive motor reverse
STEPR1:         MOV   A,#00H
                MOVC  A,@A+DPTR
                JZ    REVERSE1
                MOV   P1,A
                ACALL DELAYA         ;delay of step
                INC   DPTR
                MOV   BUF_DATABREAK,#11
                MOV   A,BUF_SWBREAK
                JNZ   TWOAUTO        ;If A=1 JUMP
                SJMP  STEPR1

TWOAUTO:        MOV   DATA_COUNTER1,DATA_COUNTER
                LCALL DELAY1S
```

```
                LJMP   WAITGO

AUTO_CHECK1:    MOV    A,BUF_STOP1
                CJNE   A,#1,$+8
                MOV    BUF_STOP1,#0
                SJMP   FOR
                MOV    A,BUF_DATABREAK
                CJNE   A,BUF_SCANKEY,FOR
                LJMP   WAITGO

AUTO_CHECK2:    MOV    A,BUF_STOP1
                CJNE   A,#1,$+8
                MOV    BUF_STOP1,#0
                SJMP   REV
                MOV    A,BUF_DATABREAK
                CJNE   A,BUF_SCANKEY,REV
                LJMP   WAITGO

WRITE:          MOV    P0,#0FFH
                CLR    P2.5
                SETB   P2.6
                SETB   P2.4
BUSY:           JB     P0.7,BUSY
                CLR    P2.4
WR1:            SETB   P2.5
                CLR    P2.6
                MOV    P0,BUF_ASCII
                SETB   P2.4
                CLR    P2.4
                RET

WRCOMMAND:      MOV    P0,#0FFH
```

```
                CLR   P2.5
                SETB  P2.6
                SETB  P2.4
BUSY1:          JB    P0.7,BUSY1
                CLR   P2.4
WRCOMM1:        CLR   P2.5
                CLR   P2.6
                MOV   P0,BUF_ASCII
                SETB  P2.4
                CLR   P2.4
                RET

DELAY1S:        PUSH  PSW
                MOV   R7,#10
AGAIN:          MOV   LOOP1,#179
DLY1:           MOV   LOOP2,#0
                DJNZ  LOOP2,$
                DJNZ  LOOP1,DLY1
                DJNZ  R7,AGAIN
                POP   PSW
                RET

DELAY1ms:       PUSH  PSW       ;DELAY=1*1ms=1ms
                MOV   R6,#1
AGAIN1:         MOV   LOOP3,#2
DLY2:           MOV   LOOP4,#0
                DJNZ  LOOP4,$
                DJNZ  LOOP3,DLY2
                DJNZ  R6,AGAIN1
                POP   PSW
                RET
```

```
DELAYA:     PUSH   PSW        ;DELAY=1*1ms=1ms
            MOV    R5,DATA_DELAY
AGAIN2:     MOV    LOOP5,#2
DLY3:       MOV    LOOP6,#0
            DJNZ   LOOP6,$
            DJNZ   LOOP5,DLY3
            DJNZ   R5,AGAIN2
            MOV    A,BUF_SCANKEY
            CJNE   A,#13,$+9
            MOV    BUF_SWBREAK,#1
            MOV    BUF_STOP1,#1
            POP    PSW
            RET

SW_BREAK:   MOV    BUF_SWBREAK,#01H  ;If swbreak is pressed,buf=01H
            RETI

INI_REV:    MOV    DPTR,#TABR     ;loop for drive motor reverse
INI_STEPR:  MOV    A,#00H
            MOVC   A,@A+DPTR
            JZ     INI_REV
            MOV    P1,A
            ACALL  DELAY1ms
            INC    DPTR
            MOV    A,BUF_SWBREAK
            JNZ    INI1           ;If A=1 JUMP
            SJMP   INI_STEPR

INI1:       LJMP   MAIN_INI

CHK_ERROR:  PUSH   PSW
            MOV    BUF_ERROR3,B      ;B=BUF_AUTO2H (RESULT OF X60)
```

```
                MOV   B,#240
                DIV   AB              ;A=BUF_AUTO2L (RESULT OF X60)
                MOV   BUF_ERROR1,A
                MOV   BUF_ERROR2,B
ERROR_DIVI:     MOV   A,BUF_ERROR3
                JZ    ERROR_NON       ;JMP IF A=0
                DEC   BUF_ERROR3
                MOV   A,#0FFH
                MOV   B,#240
                DIV   AB
                ADD   A,BUF_ERROR1
                MOV   BUF_ERROR1,A
                MOV   A,B
                ADD   A,#1
                ADD   A,BUF_ERROR2
                MOV   B,#240
                DIV   AB
                ADD   A,BUF_ERROR1
                MOV   BUF_ERROR1,A
                MOV   BUF_ERROR2,B
                MOV   BUF_ERROR4,BUF_ERROR1
                SJMP  ERROR_DIVI
ERROR_NON:      POP   PSW
                RET

TAB1:           DB    " SYRINGE PUMP "
TAB3:           DB    " SELECT MODE="
TAB4:           DB    "   MANUAL    "
TAB5:           DB    " SEMI AUTOMATIC "
TAB6:           DB    "REMAIN =   c.c."
TAB7:           DB    "RATE=  c.c./"
TAB8:           DB    "  DATA ERROR  "
```

```
TAB9:    DB   " PLEASE WAIT! "
TAB10:   DB   "STOP AT =  c.c."
TABF:    DB   08H,0CH,04H,06H,02H   ;value of data o/p(half step)
         DB   03H,01H,09H,00H
TABR:    DB   01H,03H,02H,06H
         DB   04H,0CH,08H,09H,00H

         END
```

ภาคผนวก ข

การโปรแกรมการใช้งาน

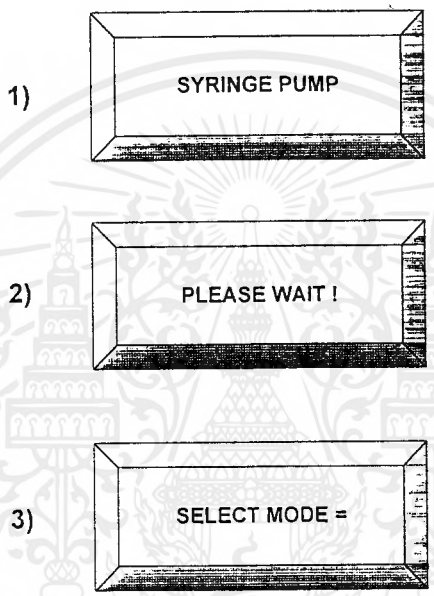
การโปรแกรมการใช้งาน

จากคุณสมบัติของเครื่องต้นแบบ เราสามารถแบ่งการทำงานได้ 2 mode คือ

1. MODE MUNUAL
2. MODE SEMI AUTOMATIC

1. การทำงานเริ่มต้น

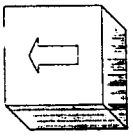
เมื่อทำการเปิดสวิทช์เข้าเครื่องให้เครื่องทำงานแล้วนั้น ที่จอจะแสดงผลตามลำดับดังรูปคือ

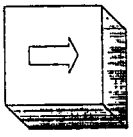


ภาคผนวกรูปที่ 1 รูปแสดงหน้าจอในขณะเริ่มต้นการทำงานของเครื่อง

2. การทำงานของ mode 1 คือ mode munual

mode munual คือ โหมดการทำงานที่สามารถควบคุมการขับหลอดฉีดยา ไปข้างหน้าหรือไปข้างหลัง ด้วยอัตราการไหลและความเร็วคงที่ ด้วยค่าความเร็วสูงสุด โดยเมื่อเลือกโหมด 1 จากเมนูหลักแล้ว ที่จอ LCD ก็จะแสดงค่าปริมาณของของเหลวที่เหลืออยู่ในหลอดขณะนั้นได้ การควบคุมทำได้โดยการกดปุ่มลูกศรไปข้างหน้า หรือไปข้างหลัง และเมื่อปล่อยมือจากปุ่มแล้ว ตัวดันหลอดก็จะหยุดเองทันที และปุ่มที่สำคัญอีกปุ่มหนึ่งคือปุ่ม ST. คือปุ่มกลับเมนูหลัก


→ ทำการดันเข็มไปข้างหน้า


→ ทำการดันเข็มไปข้างหลัง

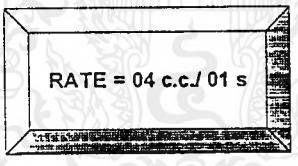
→ กลับเมนูหลัก

ภาคผนวกรูปที่ 2 แสดงปุ่มใช้งานใน mode munual

3. การทำงานของ mode 2 คือ mode semi automatic

mode semi automatic คือ mode การทำงานที่สามารถควบคุมค่าอัตราไหล(flow rate) และค่าปริมาณที่ต้องการให้หยุดได้ โดยมีลำดับขั้นตอนการใส่ค่าดังนี้

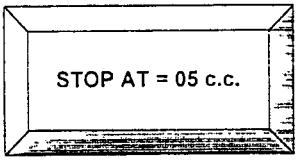
- เลือก mode 2
- ใส่ค่า flow rate ที่ต้องการ เช่น ต้องการ 4 C.C / 1 Sec. ที่หน้าจอ LCDจะ แสดงดังนี้



ภาคผนวกรูปที่ 3 แสดงหน้าจอ LCD เมื่อป้อนค่าอัตราการไหล

เมื่อใส่ค่าที่ต้องการแล้วก็ทำการกดคีย์ enter หรือถ้าต้องการกดค่าใหม่ทำได้โดยกดปุ่ม ST. และที่ขั้นตอนนี้สามารถกลับเมนูหลักได้โดยการกดปุ่ม S/M

- เมื่อต้องการใส่ค่าเป็นนาที (min) ทำได้โดยการกดปุ่ม S/M เพื่อเปรี่ยนจากS ไป M และ M เป็น S โดยS หมายถึง วินาที (Sec.) และ M หมายถึง นาที (Min.)
- ลำดับต่อไปก็เป็นการใส่ค่าปริมาณที่ต้องการให้หยุดดันหลอดฉีดยาโดยจะเป็นการใส่ค่าตัวเลขเท่านั้น ถ้าต้องการเปลี่ยนค่าใหม่ให้กดปุ่ม ST. (ถ้ากดปุ่ม S/M จะกลับเมนูหลักทันที) และเมื่อ

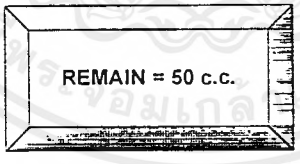


ภาคผนวกรูปที่ 4 แสดงการป้อนค่าปริมาณที่ต้องการให้หยุด

- ป้อนค่าที่ต้องการแล้วก็ทำการกดปุ่ม enter เพื่อทำตามลำดับต่อไป (ถ้าป้อนค่า "00"หรือเกิน 50 C.C.ก็จะดันไปจนสุดหลอด)
- เมื่อใส่ค่าปริมาณแล้ว ถ้าค่าอัตราไหลเกิดผิดพลาด คือ ค่านี้เกินความสามารถของเครื่องที่จะทำได้ ที่หน้าจอ LCD จะปรากฏคำว่า " DATA ERROR " แล้วให้ดำเนินการป้อนค่าใหม่

ภาคผนวกรูปที่ 5 แสดงการผิดพลาดของค่าอัตราการไหล

- เมื่อผ่านการป้อนค่าปริมาณ และค่าอัตราการไหลไม่ผิดพลาดแล้วที่หน้าจอก็จะแสดงค่าปริมาณที่เหลืออยู่ในหลอดโดยแสดงที่จอ LCD เช่น

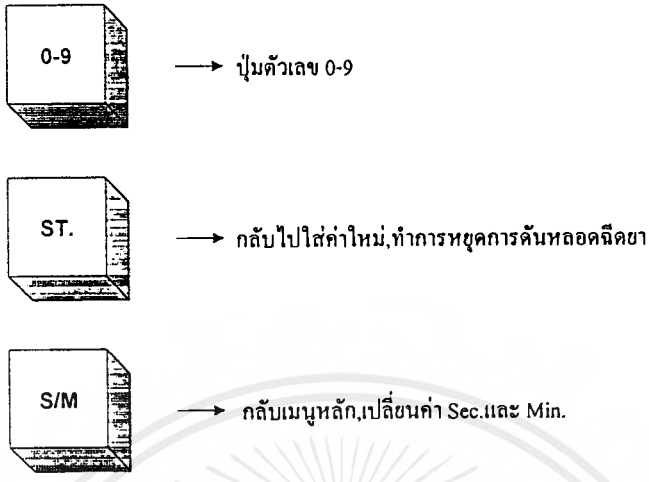


ภาคผนวกรูปที่ 6 แสดงค่าปริมาณของของเหลวที่เหลืออยู่ในหลอด

การควบคุมการดันหลอดทำได้โดยการกดปุ่มลูกศร เพื่อเลื่อนไปข้างหน้าหรือข้างหลัง และถ้าต้อลการหยุดการดัน ทำได้โดยการกดปุ่ม ST. และเมื่อต้องการดันต่อก็ทำการกดปุ่มลูกศรเลื่อนไปข้างหน้าหรือข้างหลังเหมือนเดิม

- เมื่อต้องการป้อนค่าใหม่ ทำได้โดยการกดปุ่ม ST. (การทำงานส่วนนี้ จะกระทำหลังจากทำข้อที่ผ่านมา) ที่จอ LCD ก็จะให้ป้อนค่าใหม่อีกครั้งตามขั้นตอนที่ผ่านมา โดยที่ค่าปริมาณ

ของของเหลวที่เหลือในหลอดจะยังคงค่าเดิมอยู่ (จนกว่าจะกลับเมนูหลัก จึงจะกลับค่าเป็น 50 C.C.)



ภาคผนวกรูปที่ 7 รูปแสดงปุ่มใช้งานที่ mode semi automatic

กิตติกรรมประกาศ

การทำงานครั้งนี้จะไม่สำเร็จได้เลยถ้า ไม่ได้ผู้เชี่ยวชาญทางด้านนี้ ที่คอยให้คำแนะนำและวิธีแก้ไขปัญหาต่าง ๆ ตลอดจนสอนวิธีการทดลองที่สำคัญให้ ทั้งทางทฤษฎี และทางปฏิบัติ ท่านเหล่านั้นคือ อาจารย์ ดร. กิติพล ชิตสกุล รวมทั้งเพื่อน ๆ และน้องๆ ที่คอยช่วยแก้ปัญหาที่เกิดขึ้นและแนะนำวิธีหลาย ๆ แบบในการออกแบบวงจร และการเขียนโปรแกรม อนึ่งความดีต่าง ๆ ที่เกิดจากปริญญานิพนธ์ฉบับนี้ ผู้เขียนขอมอบให้ท่านอาจารย์ที่ได้ประสิทธิ์ ประสาทวิชาต่าง ๆ ให้กับผู้เขียน

ชาญฤทธิ์ ยศสันติกุล

(นายชาญฤทธิ์ ยศสันติกุล)

วาที รณรงค์

(นายวาที รณรงค์)

ผู้จัดทำ

หนังสืออ้างอิง

1. กฤษดา วิศวธีรานนท์, "เรียน/เล่น/ใช้ ไอซีดิจิตอล", ซีเอ็ดยูเคชั่น กรุงเทพ 2532
2. มนัส สังวรศิลป์ และ สมเกียรติ ศุภเดช, "ทฤษฎีและการออกแบบวงจรพัลส์" , อิเลคทรอนิคส์ เวิลด์ พิมพ์ครั้งที่ 2 กรุงเทพ 2526
3. สมยศ จุณณะปิยะ, "การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ตระกูล MCS-51", คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง,พิมพ์ครั้งที่ 2 กรุงเทพ 2541
4. ธีรวัฒน์ ประกอบผล, "การประยุกต์ใช้งานไมโครคอนโทรลเลอร์", ประชาชน กรุงเทพ 2540
5. สุนทร วิทูสุรพจน์, "การใช้งานไมโครคอนโทรลเลอร์ตระกูล 8051", ซีเอ็ดยูเคชั่น กรุงเทพ 2541
6. สุนทร วิทูสุรพจน์, "การโปรแกรมภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ตระกูล 8051", ซีเอ็ดยูเคชั่น กรุงเทพ 2541
7. Jame Bignell and Robert Donovan, "DIGITAL ELETRONICS", Delmar Publishers Inc., 1994.