การพัฒนาเครื่องคัดแยกสีมะนาว

ธนากร สุนทรวัฒน์

สนับสนุนงบประมาณโดย
มหาวิทยาลัยเทคโนโลยีราชมงคลรัตนโกสินทร์
ประจำปีงบประมาณ พ.ศ.2556

Developing Lemon Color Sorter

By

Tanakorn Suntornwat

Granted by

Rajamangala University of Technology Rattanakosin

Fiscal year 2013

กิตติกรรมประกาศ

ผู้จัดทำงานวิจัยชิ้นนี้ขอขอบคุณ บิดา-มารดา บูรพคณาจารย์ ทั้งหมดทุกท่านที่ให้การเลี้ยงดู อบรมสั่งสอนให้มีความรู้ที่สามารถทำให้งานวิจัยชิ้นนี้สำเร็จสุล่วงได้ด้วยดี ขอขอบคุณ อาจารย์ทรงสิทธิ์ สอนรอด ที่ช่วยตรวจสอบบทคัดย่อภาคภาษาอังกฤษ และขอบคุณ มหาวิทยาลัยเทคโนโลยีราชมงคลรัตนโกสินทร์ที่ให้การสนับสนุนทุนในการทำวิจัยในครั้งนี้

ประโยชน์อันใดที่เกิดจากงานวิจัยชิ้นนี้ ขอมอบเป็นกุศลให้กับท่านทั้งหลายที่ได้กล่าวถึงก่อนหน้านี้ ซึ่งเป็นผลมาจากความกรุณาของท่านดังกล่าวข้างต้น ผู้จัดทำขอขอบคุณมา ณ โอกาสนี้

ธนากร สุนทรวัฒน์

สิงหาคม 2556

บทคัดย่อ

รหัสโครงการ : Inno 02/2/2556
ชื่อโครงการ : การพัฒนาเครื่องคัดแยกสีมะนาว
ชื่อนักวิจัย : นายธนากร สุนทรวัฒน์

งานวิจัยชิ้นนี้เป็นการศึกษาและพัฒนาสร้างเครื่องคัดแยกสีมะนาว เพื่อลดภาระของเกษตรกรในการคัดแยกมะนาวก่อนส่งออกจำหน่าย เนื่องจากราคาขายผลมะนาวสีเขียวและสีเหลืองนั้นราคาต่างกัน เกษตรกรจึงจำเป็นต้องคัดแยกก่อนส่งออกขาย

หลักการทำงานของเครื่องเริ่มจากการดึงลูกมะนาวเข้าระบบคัดแยกโดยให้มะนาวตกลงไปในช่องที่สร้างขึ้นขนาดที่พอเหมาะกับลูกมะนาว แล้วส่งต่อให้เซนเซอร์สี ทำการตรวจจับสี เมื่อได้ค่าสีจากเซนเซอร์แล้วระบบจะนำค่าที่ได้ไปเปรียบเทียบกับค่าที่ได้บันทึกไว้ก่อนหน้านี้ เพื่อตัดสินใจในการคัดแยกว่ามะนาวลูกนี้เป็นสีเขียว หรือสีเหลือง ซึ่งผลจากการตัดสินใจนี้จะสั่งให้เซอร์โวมอเตอร์ หมุนแกนเพื่อเปลี่ยนทิศทางการกลิ้งออกของลูกมะนาวไปในช่องทางของสีที่ได้ทำการคัดแยกแล้ว ทั้งหมดนี้ถูกควบคุมการทำงานโดยไมโครคอนโทรลเลอร์ ตัวเครื่องมีความสามารถในการตั้งค่าสีของลูกมะนาวที่จะทำการคัดแยกได้ สามารถตั้งค่าการนับของมะนาวแต่ละสีเพื่อหยุดการคัดแยกได้ และสามารถนับจำนวนมะนาวที่ได้ทำการคัดแยกแล้วในแต่ละสีได้

ผลที่ได้จากการทดลองใช้งาน พบว่า เครื่องคัดแยกสีมะนาวสามารถทำงานได้ถูกต้องตามที่ออกแบบไว้โดยมีความผิดพลาดในการคัดแยก เฉลี่ย 3 % และมีความเร็วในการคัดแยกเฉลี่ย 30 ลูกต่อนาที

คำสำคัญ : มะนาว,เซ็นเซอร์สี,เซอร์โวมอเตอร์,ไมโครคอนโทรลเลอร์

E-mail Address : tanakorn.s@rmutr.ac.th
ระยะเวลาโครงการ : ตุลาคม 2555 – กันยายน 2556

Abstract

Code of Project : Inno 02/2/2556
Project Name : Developing Lemon Color Sorter
Researcher Name : Mr.Tanakorn Suntornwat

This research is to study and develop lemon color sorting machine designed to help reduce the burden on farmers in sorting the lemons before export. As the price of green lemons and yellow lemons is different, farmers need to sort them before export.

Principle of machine starts with loading the lemon into the sorting system, letting them drop into the holes created with the right sizes. The lemons, then, will be forwarded to color sensor system for color detection. When getting the right color value, the color value will be systematically compared with the recorded ones to sort the green and yellow lemons. The result of decision in the system will control servo-motor to turn the spindle to veer the lemons to the exit of desired color. All the mentioned processes are controlled by Microcontroller. The machine can set the color value of lemons in the sorting process, can set the counting system of the lemon in order to stop the machine during the sorting process, and can count the numbers of lemons which have already been in the color sorting process based on their sorted colors.

From the result of the experiment, it is found that the lemon color sorting machine can work effectively as it is designed. The average sorting error is 3% and the average sorting speed is 30 lemons per minute.

Key Words : Lemon , Color sensor , Servo-motor , Microcontroller

---

E-mail Address : tanakorn.s@rmutr.ac.th
Period of Project : October 2012 – September 2013

สารบัญ

รายการตาราง

รายการภาพประกอบ

รายการภาพประกอบ (ต่อ)

รายการสัญลักษณ์

Ω	=	โอห์ม
%	=	เปอร์เซ็นต์

ประมวลศัพท์และคำย่อ

ROM	=	Read Only Memory
RAM	=	Random Access Memory
MH_Z	=	Mega Hertz
CPU	=	Central Processing Unit
ALU	=	Arithmetic and Logic Unit
I/O	=	Input / Output
KB	=	Kilo Byte
XOR	=	Exclusive OR
LCD	=	Liquid Crystal Display
EEPROM	=	Electrical Electrical Program Read Only Memory
PCB	=	Printed Circuit Board
AC	=	Alternating Current
DC	=	Direct Current
NO	=	Normal Open
NC	=	Normal Close
LDR	=	Light Dependent Resistor
LED	=	Light Emitting Diode

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

มะนาวเป็นพืชเศรษฐกิจที่สำคัญในประเทศไทยที่เกษตรกรในหลายพื้นที่นิยมปลูกและยึดเป็นอาชีพหลัก เพื่อจำหน่ายทั้งในประเทศและส่งออกจำหน่ายในต่างประเทศ ประเทศไทยถือได้ว่ามีพื้นที่ที่เหมาะสมในการทำการเกษตร มะนาวสามารถเจริญเติบโตเป็นอย่างดีในประเทศไทยซึ่งในปัจจุบันความเจริญก้าวหน้าของโลกยุคปัจจุบันได้มีการนำเอาเทคโนโลยีมาใช้ในชีวิตประจำวันในงานหลาย ๆ ด้าน เช่น ภายในอาคารบ้านเรือน งานในโรงงานอุตสาหกรรม หน่วยงานราชการ สำนักงาน และตามบริษัทต่าง ๆ หรือแม้แต่ด้านงานเกษตร ดังนั้นจึงได้มีการคิดค้นและพัฒนาดังนั้นจึงได้มีแนวคิดที่จะนำเอาเทคโนโลยีที่มีอยู่อย่างใกล้ตัวมาใช้ให้เกิดประโยชน์อย่างมีประสิทธิภาพและให้มีการทำงานที่เร็วที่สุด

มนุษย์สามารถสร้างสรรค์สิ่งใหม่เพื่อนำมาช่วยเหลือหรือแทนที่สำหรับการทำงานของกิจกรรมต่างๆ แต่กระบวนการที่จะได้มาซึ่งเทคโนโลยีที่ทันสมัยก็ต้องอาศัยการค้นคว้า ทดลอง และการวิจัยเพื่อที่จะทำการประดิษฐ์คิดค้นสิ่งใหม่ๆ ให้เกิดประโยชน์อยู่เสมอ และในปัจจุบันสิ่งที่ได้รับความสนใจก็คือ เครื่องคัดแยกและนับจำนวนมะนาว ซึ่งมะนาวเป็นพืชเศรษฐกิจและมีความสำคัญที่มีการบริโภคในแต่ละวันเป็นจำนวนมาก จึงมีเกษตรกรในหลายพื้นที่ยึดการปลูกมะนาวเป็นอาชีพหลัก หากต้องขายผลผลิตให้ได้ราคาดีเกษตรกรจำต้องคัดเลือกผลผลิตที่มีคุณภาพที่ดีจะขายได้ในราคาที่สูงขึ้นและเป็นการนำเอาเทคโนโลยีมาประยุกต์การใช้งานในด้านการเกษตรซึ่งจะช่วยลดต้นทุนและแรงงานได้เป็นจำนวนมาก

จากปัญหาดังกล่าว จึงมีแนวคิดที่จะสร้างเครื่องคัดแยกและนับจำนวนมะนาวที่สามารถตรวจสอบสี นับจำนวนมะนาว และแสดงผล ซึ่งระบบทั้งหมดสามารถทำงานอัตโนมัติ เพื่ออำนวยความสะดวกสบายในการใช้งานมากที่สุด

1.2 วัตถุประสงค์

1.2.1 เพื่อประยุกต์ใช้เทคโนโลยีให้เหมาะสมกับสังคมเกษตรกรรมของประเทศไทย
1.2.2 เพื่อพัฒนา เครื่องมือเพื่อช่วยเกษตรกร โดยการออกแบบและสร้างเครื่องคัดแยกมะนาว
1.2.3 เพื่อทดสอบประสิทธิภาพการทำงานของเครื่องคัดแยกมะนาว
1.2.4 เพื่อเป็นอุปกรณ์ต้นแบบในการนำไปใช้งานจริงในอนาคต

1.3 กรอบแนวคิดงานวิจัย

1.3.1 ศึกษาลักษณะผลมะนาวที่จะทำการคัดแยกสี
1.3.2 ออกแบบและสร้างกลไกการทำงานในการดึงผลมะนาวเข้าระบบคัดแยก และกลไกการคัดแยก เพื่อแยกสีมะนาว สีเหลือง และ สีเขียว
1.3.3 ออกแบบและสร้างการติดตั้งเซนเซอร์และวงจรควบคุมทางไฟฟ้า
1.3.4 เขียนโปรแกรมควบคุม และปรับแต่งการทำงานให้สมบูรณ์

1.4 นิยามคำศัพท์

1.4.1 มะนาว : เป็นไม้ผลชนิดหนึ่ง ผลมีรสเปรี้ยวจัด จัดอยู่ในสกุลส้ม ผลสีเขียว เมื่อสุกจัดจะเป็นสีเหลือง เปลือกบาง ภายในมีเนื้อแบ่งกลีบๆ ชุ่มน้ำมาก นิยมใช้เป็นเครื่องปรุงรส นอกจากนี้ยังมีคุณค่าทางโภชนาการและทางการแพทย์

1.4.2 ไมโครคอนโทรลเลอร์ : คืออุปกรณ์ควบคุมที่มีขนาดเล็ก มีโครงสร้างภายในเหมือนคอมพิวเตอร์ สามารถทำงานตามโปรแกรมที่ผู้ใช้เขียนสั่งงานได้

1.4.3 เซอร์โวมอเตอร์ : คือ DC มอเตอร์ประเภทหนึ่งที่ควบคุมการหมุนและตำแหน่งที่หยุดหมุนโดยอาศัยคาบเวลาของสัญญาณพัลซ์ที่ป้อนเข้าไปควบคุม

1.4.4 เซ็นเซอร์สี : เป็นอุปกรณ์อิเล็กทรอนิกส์ที่ทำหน้าที่รับค่าสีที่ตรวจจับได้เป็นสัญญาณทางไฟฟ้าเพื่อนำไปใช้ในการควบคุมต่อไป

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ไมโครคอนโทรลเลอร์

2.1.1 ความหมายของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ คือ อุปกรณ์ประเภทสารกึ่งตัวนำที่รวมเอาสิ่งต่อไปนี้ไว้ในตัวเอง เช่น หน่วยประมวลผล (CPU) หน่วยความจำชั่วคราว (RAM) หน่วยความจำถาวร (ROM) พอร์ตอินพุต เอาท์พุต (I/O PORT)

ไมโครคอนโทรลเลอร์ที่มีใช้งานในปัจจุบันมีอยู่หลากหลายตระกูลมาก แต่ที่นิยมใช้กันอย่างแพร่หลาย ในปัจจุบันมีอยู่สองตระกูลด้วยกัน คือ MCS51 และ PIC ที่มีให้เหลือกหลายเบอร์ ส่วนตระกูลอื่นๆ นอกเหนือจากนี้ ก็มีความสามารถ ที่ไม่แพ้กัน เพียงแต่ว่าอาจจะใหม่ จึงไม่เป็นที่รู้จักเท่ากับตระกูลที่มีมาก่อนหน้า

ในการเลือกใช้เราก็ต้องดูว่า เราจำเป็นต้องใช้โมดูลไหนบ้างที่มีอยู่ในคอนโทรลเลอร์ INPUT OUTPUT ที่ใช้มีกี่ขา ROM RAM เพียงพอหรือไม่ และ เหตุผลอื่นๆ เพื่อที่จะได้เลือกใช้คอนโทรลเลอร์ ได้อย่างมีประสิทธิภาพ ในงบประมาณที่พอเหมาะ

2.1.2 ไมโครคอนโทรลเลอร์ตระกูล PIC-16F877

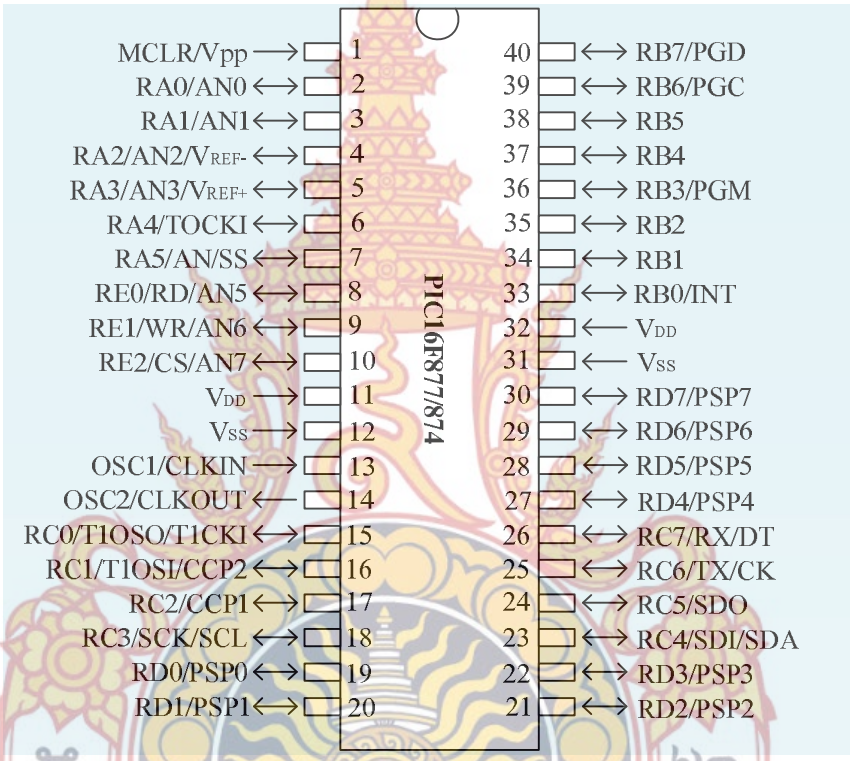
ไมโครคอนโทรลเลอร์ตระกูล PIC-16F877 จะมีสถาปัตยกรรมแบบฮาร์วาร์ด (Harvard Architecture) กล่าวคือ มีการแยกหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลออกจากกัน โดยมีบัสสำหรับติดต่อแยกกันด้วย ดังแสดงในภาพที่ 1 จะเห็นว่าซีพียูภายในไมโครคอนโทรลเลอร์จะติดต่อกับหน่วยความจำโปรแกรมด้วยบัสแอดเดรส 13 บิต และบัสข้อมูลหน่วยความจำโปรแกรม 14 บิต ในขณะที่บัสสำหรับติดต่อกับหน่วยความจำข้อมูลและรีจิสเตอร์ภายในเป็นแบบ 8 บิตทั้งบัสแอดเดรสและบัสข้อมูล

ภาพที่ 1 ไดอะแกรมรูปแบบสถาปัตยกรรมของไมโครคอนโทรลเลอร์

นอกจากการจัดสถาปัตยกรรมแบบนี้แล้ว การกระทำคำสั่งการทำงานของไมโครคอนโทรลเลอร์ตระกูล PIC-16F877 ยังคงใช้กระบวนการที่เรียกว่า ไปป์ไล์ (Pipeline) ทำให้สามารถเฟตช์คำสั่งถัดไป ในขณะที่กำลังกระทำคำสั่งให้เกิดผลตามคำสั่งนั้นๆ กำหนดหรือ

กระบวนการเอ็กซิคิวต์คำสั่งในปัจจุบัน ส่งผลให้ความเร็วในการทำงานของไมโครคอนโทรลเลอร์เพิ่มมากขึ้น นั่นจึงเป็นที่มาของความสามารถในการกระทำคำสั่ง 1 คำสั่งภายในสัญญาณนาฬิกา 1 ลูก (กระบวนการเฟตช์ (Fetch) เป็นกระบวนการเรียกคำสั่งออกจากหน่วยความจำโปรแกรมแล้วทำการแปลคำสั่งนั้นให้เป็นเลขฐานสิบหกเพื่อให้ซีพียูเข้าใจ ส่วนกระบวนการเอ็กซิคิวต์ (Execute) เป็นการกระทำคำสั่งให้เกิดผลลัพธ์ตามที่คำสั่งนั้นๆ กำหนด)

พื้นฐานการทำงานของไมโครคอนโทรลเลอร์ คือ ระบบดิจิตอลโดยค่าเอาท์พุตที่ได้จากไมโครคอนโทรลเลอร์จะเป็น 0 กับ 1แต่ก็สามารถนำมาประยุกต์เชื่อมต่อกับอุปกรณ์ภายนอกต่างๆ มากมาย โดยการเลือกพอร์ตใช้งานจากขาต่างๆ ดังแสดงในภาพที่ 1



ภาพที่ 2 แสดงขาของไมโครคอนโทรลเลอร์ตระกูล PIC-16F877

จากภาพขาของ PIC-16F877 กับ 16F874 จะสามรถดูได้จาก data sheet แต่ละขาจะมีหน้าที่แตกต่างกันไปซึ่งแยกออกเป็น พอร์ต A, พอร์ต B, พอร์ต C, พอร์ต D, พอร์ต E โดยพื้นฐานแล้วพอร์ตแต่ละพอร์ตสามารถทำงานเป็นอินพุตและเอาท์พุตเป็นดิจิตอล ยกเว้น พอร์ต A และ พอร์ต E ที่สามารถทำงานเป็นตัวรับสัญญาณอนาลอกแปลงเป็นค่าดิจิตอลเพื่อนำมาวัดปริมาณทางฟิสิกส์ต่างๆ ที่เห็นได้อย่างชัดเจน คือ นำมาวัดความต่างศักย์

2.1.3 คุณสมบัติของไมโครคอนโทรลเลอร์ ตระกูล PIC-16F877

มีคำสั่งให้ใช้งาน 35 คำสั่ง คำสั่งหนึ่งๆใช้เวลาทำงาน 1 ถึง 2 Cycle ทำงานได้สูงสุดที่ความถี่สัญญาณนาฬิกา 20 MHz ทำงานแบบ Pipe-line (มี 2 ท่อ) ทำให้ ณ เวลาหนึ่งทำงาน 2 อย่างพร้อมกัน หน่วยความจำโปรแกรมเป็นแบบ Flash มีขนาด 8K Word (1 word=14 บิต) มี

ขนาดหน่วยความจำ (RAM) 368 ไบต์ มี EEPROM ขนาด 256 ไบต์ ตอบสนองกับอินเตอร์รัพได้ทั้งหมด 14 แหล่ง มี Stack ให้ใช้ได้สูงสุด 8 ระดับ มีระบบ Power On Reset, Power Up Timer, Oscillator Start-up timer มีระบบ Code Protection สัญญาณนาฬิกามีหลายโหมดให้เลือกใช้งาน คือ อาจจะใช้ XTAL หรือวงจร RC ก็ได้ สามารถโปรแกรมด้วยไฟ +5VDC ได้ ใช้การโปรแกรมแบบ In-Circuit Serial Programming ทำงานที่ไฟเลี้ยง 2VDC ถึง 5.5VDC Current Sink และ Current Source อยู่ที่ 25mA มี Timer/Counter 3 ตัว มีโมดูล Capture/Compare/PWM อีก 2 ชุด มี A-TO-D Converter แบบ 10 บิต จำนวน 8 ช่องนำเข้าในตัวเอง มีระบบ USART สำหรับต่อกับ การสื่อสารแบบ RS232 หรือดีกว่า มีระบบตรวจระดับไปเลี้ยง (Brown-out reset) มี I/O พอร์ตทั้งหมด 5 พอร์ต แต่ละพอร์ตมีจำนวนบิตไม่เท่ากันรวมแล้ว จะมี I/O

จำนวน 33 บิตโดยแบ่งออกเป็น

PORTA = RA 5 + RA0 จำนวน 6 บิต

PORTB = RB 7 + RB0 จำนวน 8 บิต

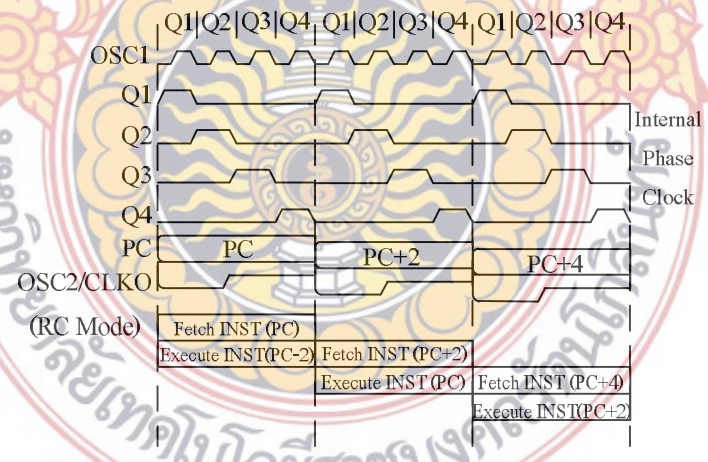
PORTC = RC 7 + RC0 จำนวน 8 บิต

PORTD = RD 7 + RD0 จำนวน 8 บิต

PORTE = RE 2 + RE0 จำนวน 3 บิต

2.1.4 สัญญาณนาฬิกา

ไมโครคอนโทรลเลอร์จะทำงานได้ต้องมีสัญญาณนาฬิกาให้กับตัวซึ่งในหนึ่งไซเคิล (Clock Bus) ของซีพียูจะประกอบไปด้วยสัญญาณนาฬิกาภายนอกจำนวน 4 ไซเคิล คือ Q1, Q2, Q3 และ Q4 ดังแสดงในภาพที่ 3 ดังนั้นความถี่ที่ซีพียูประมวลผลต่อหนึ่งคำสั่งจะเท่ากับความถี่ของสัญญาณนาฬิกาภายนอกหารด้วย 4 หรือหากจะพิจารณาความเร็วของไมโครคอนโทรลเลอร์ตระกูล PIC-16F877 สามารถประมวลผลต่อหนึ่งคำสั่งเท่ากับ 1/4 เท่าของความถี่ออสซิลเลเตอร์ภายนอก



ภาพที่ 3 สัญญาณนาฬิกา

2.1.5 โหมดสัญญาณนาฬิกา

PIC16F877 สามารถเลือกโหมดสัญญาณนาฬิกาเพื่อกำหนดสัญญาณการทำงานได้มากถึง 4 โหมด โดยการกำหนดที่บิด FOSC1 ในรีจิสเตอร์ Configuration Word ในการทำงานจะต้องเลือกโหมดหนึ่ง ดังรายละเอียด

โหมด LP (Low Power Crystal) ใช้กับคริสตอลหรือเซรามิกเรโซเนเตอร์พลังงานต่ำความถี่ 32KHz - 200KHz

โหมด LP (Crystal/Resonator) ใช้กับคริสตอลหรือเซรามิกเรโซเนเตอร์พลังงานต่ำความถี่ 200KHz - 4MHz

โหมด HS (High Speed Crystal/Resonator) ใช้กับคริสตอลหรือเซรามิกเรโซเนเตอร์พลังงานต่ำความถี่ 4MHz - 20MHz

โหมด RC สามารถกำหนดค่าความถี่ได้จากค่าความต้านทานและตัวเก็บประจุที่ต่อภายนอกเข้ากับขา OSC1/CLKIN

2.1.6 ประเภทของออสซิลเลเตอร์ ดังแสดงในตารางที่ 1

LP (Low Power Crystal) คริสตอลพลังงานต่ำ

XT (Crystal/Resonator) คริสตอล หรือ เรโซเนเตอร์

HS (High Speed Crystal/Resonator) คริสตอล หรือ เรโซเนเตอร์ความเร็วสูง

RC (External Resistor/Capacitor) วงจร RC ภายนอก

H4 (HS + PLL: High Speed Crystal/Resonator with PLL enabled) คูณ 4 PLL คือจะทำการคูณสัญญาณนาฬิกาที่เข้ามา ด้วย 4 เช่น OSC ความถี่ 10 MHz เมื่อผ่านกระบวน การนี้จะทำให้ได้ความถี่เท่ากับ 40 MHz

2.1.7 ออสซิลเลเตอร์แบบคริสตอล

ออสซิลเลเตอร์แบบคริสตอลที่ใช้ในไมโครคอนโทรลเลอร์ตระกูล PIC-16F877 นี้จะเลือกใช้แบบ XT จะต้องใช้วงจรรีโซเนเตอร์ แบบเซรามิกหรือคริสตอลต่อเข้าที่ขา OSC1 และ OSC2 เพื่อทำให้เกิดสัญญาณนาฬิกา การเลือกใช้ตัวเก็บประจุ สำหรับวงจรรีโซเนเตอร์แบบเซรามิก จะคำนึงถึงความถี่ต่างๆที่ใช้ ดังแสดงในตารางที่ 1

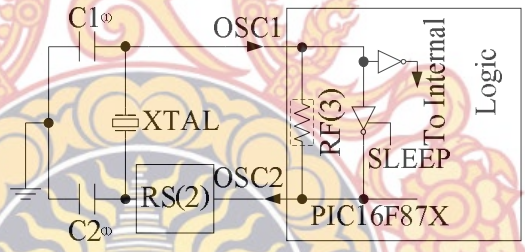
ตารางที่ 1 ค่าตัวเก็บประจุที่เหมาะสมกับความถี่ที่ใช้

Osc Type	Crystal Freq.	Cap. Range C1	Cap. Range C2
LP	32kHz	33pF	33pF
	200kHz	15pF	15pF
XT	200kHz	47-68pF	47-68pF
	1MHz	15pF	15pF

	4MHz	15pF	15pF
HS	4MHz	15pF	15pF
	8MHz	15-33pF	15-33pF

ชิพแบบ XT เป็นออสซิลเลเตอร์คริสตอลแบบมาตรฐาน ซึ่งอาจต้องการคริสตอลแบบสตริปคัต AT (AT Strip-cut) เพื่อหลีกเลี่ยงการโอเวอร์ไดรฟ์ (Overdrive)

วิธีการเชื่อมต่ออุปกรณ์ RC เข้ากับไมโครคอนโทรลเลอร์ตระกูล PIC-16F877 สำหรับค่า Rext ที่น้อยกว่า 2.2 กิโลโอห์ม ทำให้สัญญาณออสซิลเลเตอร์ที่ได้อาจจะไม่คงที่หรือหยุดนิ่งสำหรับค่า Rext ที่มีค่าสูงมากๆ (เช่น 1 เมกะโอห์ม) ออสซิลเตอร์จะมีความไวต่อสัญญาณรบกวนความชื้นและสภาวะแวดล้อมภายนอก ดังนั้นควรจะใช้ค่า Rext ให้มีค่าอยู่ในช่วง 5 กิโลโอห์ม ถึง 100 กิโลโอห์ม ถึงแม้ว่าออสซิลเลเตอร์จะทำงานได้โดยไม่ต้องต่อตัวเก็บประจุภายนอก (Cext=0 pF) แต่ควรใส่ค่าตัวเก็บประจุที่มากกว่า 20 pF เพื่อลดสัญญาณรบกวนและให้สัญญาณมีความคงที่ ถ้าไม่มีตัวเก็บประจุหรือตัวเก็บประจุภายนอกมีค่าน้อยเกินไป จะทำให้ความถี่ออสซิลเลเตอร์มีการเปลี่ยนแปลงอย่างกระทันหัน เนื่องจากการเปลี่ยนแปลงที่ตัวเก็บประจุภายนอก เช่น ที่แผ่นวงจรพิมพ์บริเวณตัวเก็บประจุหรือตัวนำของตัวเก็บประจุ



ภาพที่ 4 วิธีการเชื่อมต่ออุปกรณ์ RC เข้ากับ ไมโครคอนโทรลเลอร์ตระกูล PIC-16F877

2.2 เซนเซอร์

2.2.1 หลักการทำงานของอินฟาเรด

หลัก การของวงจรอินฟาเรด ตัวส่งแบบใช้กับโมดูลรับสำเร็จรูป(3ขา) จะส่งด้วยความถี่ 40KHzโดยประมาณ ประโยชน์เพื่อเป็นความถี่หลักในการตรวจรับว่าเป็นสัญญาณจริงๆไม่ใช่สัญญาณรบกวน ตัวรับแบบโมดูล(3ขา) โมดูลจะรับสัญญาณที่กระพริบด้วยความถี่ประมาณ40KHz ถ้าตรงก็จะให้เอาท์พุทที่ขาเอาท์พุทเป็น”0”หลักการของมันก็มีแค่ ส่งแสงอินฟราเรดไปยังวัตถุที่ต้องการตรวจจับ ถ้าพบวัตถุนั้นก็จะสะท้อนแสงกลับมายังตัวรับ สีที่สะท้อนได้ดีที่สุดก็คือสีขาว ถ้าเป็นสีดำจะดูดกลืนได้มากกว่า

2.2.2 วงจรเปรียบเทียบแรงดันโดยใช้ออปแอมป์

ต้องการให้แรงดันจากเซ็นเซอร์มากกว่า ค่าที่กำหนด (แรงดันอ้างอิง) ออกมาเป็น 0 ก็ให้ต่อแรงดันอ้างอิงเข้า + ,ค่าจากเซ็นเซอร์เข้า - (วงจรแบบอินเวิร์ส)

ต้องการให้แรงดันจากเซ็นเซอร์มากกว่า ค่าที่กำหนด (แรงดันอ้างอิง) ออกมาเป็น 1 ก็ให้ต่อแรงดันอ้างอิงเข้า - ,ค่าจากเซ็นเซอร์เข้า + (วงจรแบบไม่อินเวิร์ส)

2.2.2.1วงจรแบบอินเวิร์ส

ถ้าสัญญาณเซ็นเซอร์ เข้ามาขา 2 น้อยกว่าขา 3 ที่ปรับไว้ เช่น 2.8V (แรงดันอ้างอิง) จะได้ค่าออกมาเป็น 1

ถ้าสัญญาณเซ็นเซอร์ เข้ามาขา 2 มากกว่าขา 3 ที่ปรับไว้ เช่น 2.8V (แรงดันอ้างอิง) จะได้ค่าออกมาเป็น 0

ซึ่งหากนำไปใช้กับเซ็นเซอร์แรงดันสูงสุดที่ออกมาประมาณ 2.8V ที่ระยะ<10cm. แรงดันต่ำสุดที่ออกมาประมาณ 0.4V ที่ระยะ 80cm. ก็หมายถึงมันจะให้ค่าออกมาเป็น 1 เมื่อระยะมากกว่า10Cm

2.2.2.2 วงจรแบบไม่อินเวิร์ส

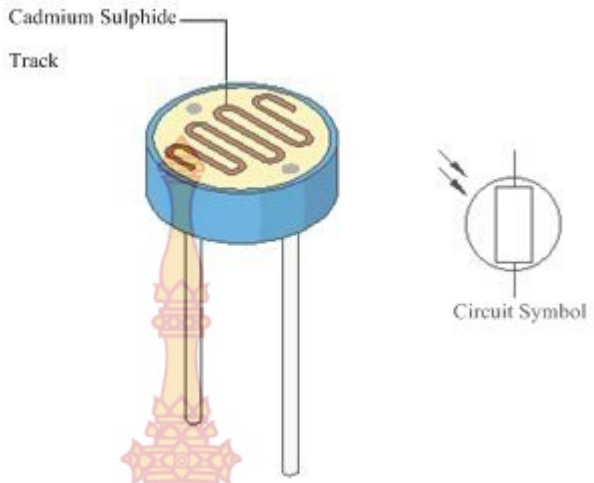
ถ้าสัญญาณเซ็นเซอร์เข้ามาขา 5 น้อยกว่าขา 4 ที่ปรับไว้ เช่น 2.8V (แรงดันอ้างอิง) จะได้ค่าออกมาเป็น 0

ถ้าสัญญาณเซ็นเซอร์ เข้ามาขา 5 มากกว่าขา 4 ที่ปรับไว้ เช่น 2.8V (แรงดันอ้างอิง) จะได้ค่าออกมาเป็น 1

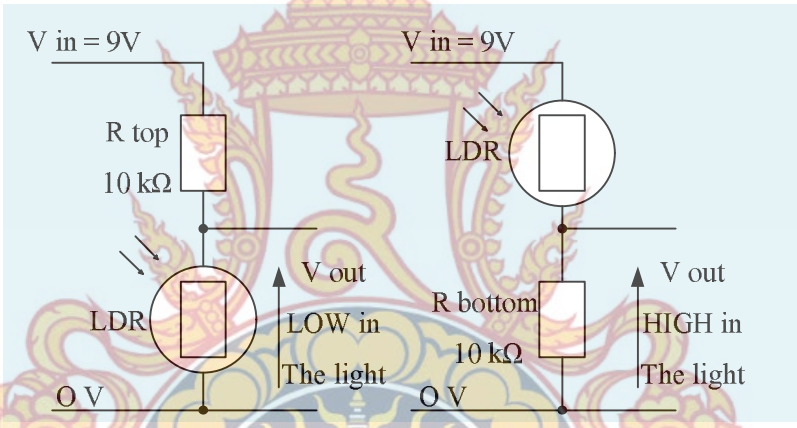
แรงดันสูงสุดที่ออกมาประมาณ 2.8V ที่ระยะ<10cm. แรงดันต่ำสุดที่ออกมาประมาณ 0.4V ที่ระยะ 80cm. ก็หมายถึง มันจะให้ค่าออกมาเป็น 1 เมื่อ ระยะน้อยกว่า 10 Cm

2.3 LDR

เป็น SENSOR ที่เปลี่ยนพลังงานแสงเป็นพลังงานไฟฟ้าโดยค่าความต้านทานขึ้นอยู่กับแสงที่ตกกระทบบนตัว LDR ถ้ามีแสงมากระทบจะให้ค่าความต้านทานต่ำ (ค่าแรงดันต่ำ) แต่ถ้าไม่มีแสงมาตกกระทบจะให้ค่าความต้านทานสูง (ค่าแรงดันสูง)



ภาพที่ 5 ลักษณะและสัญลักษณ์ LDR



ภาพที่ 6 วงจรต่อสลับระหว่าง LDR กับตัวต้านทาน

2.4 จอแอลซีดี

เทคโนโลยีมอนิเตอร์ LCD ย่อมาจาก Liquid Crystal Display ซึ่งเป็นจอแสดงผลแบบ (Digital) โดยภาพที่ปรากฏขึ้นเกิดจากแสงที่ถูกปล่อยออกมาจากหลอดไฟด้านหลังของจอภาพ (Black Light) ผ่านชั้นกรองแสง (Polarized filter) แล้ววิ่งไปยัง คริสตัลเหลวที่เรียงตัวด้วยกัน 3 เซลล์คือ แสงสีแดง แสงสีเขียวและแสงสีน้ำเงิน กลายเป็นพิกเซล (Pixel) ที่สว่างสดใสเกิดขึ้นจอภาพแบบ LCD (Liquid Crystal Display) มีการทำงานที่ซับซ้อนกว่าแบบ CRT เนื่องจากโครงสร้างภายในจะเป็นอุปกรณ์อิเล็กทรอนิกส์ทั้งหมด ในจอภาพ LCD จะมีผลึกคริสตัลเหลว (ตามชื่อ) ซึ่งเมื่อเราจ่ายแระแสไฟฟ้าเข้าไปยังผลึกคริสตัลเหลวนี้ก็จะเกิดการบิดตัวของผลึก แต่การบิดตัวนี้จะมากน้อยก็ขึ้นอยู่กับประมาณของกระแสไฟฟ้าที่จ่ายเข้าไปนั่นเอง

2.4.1 เทคโนโลยีมอนิเตอร์แบบ LCD มีจุดเด่นหลายประการคือ

ขนาดเล็กกะทัดรัดและน้ำหนักเบาด้วยการทำงานที่ไม่ต้องอาศัยปืนยิงอิเล็กตรอน จึงช่วยให้ด้านลึกของจอภาพมีขนาดสั้นกว่ามอนิเตอร์แบบ CDT ถึง 3 เท่าและด้วยรูปร่างที่แบน ราบทางด้านหน้าและด้านหลัง ในบางรุ่นจึงมีอุปกรณ์เสริมพิเศษสำหรับติดฝาผนังช่วยให้ประหยัดพื้นที่มากยิ่งขึ้น

พื้นที่การแสดงผลเต็มพื้นที่จากเทคโนโลยีพื้นฐานในการออกแบบ ทำให้จอมอนิเตอร์แบบ LCD สามารถแสดงผลได้เต็มพื้นที่เมื่อเปรียบเทียบกับแบบ CDT ขนาด 17 นิ้วเท่ากัน พื้นที่แสดงผลที่กว้างที่สุดจะอยู่ที่ 15 นิ้วกว่าๆ เท่านั้น ให้ภาพที่คมชัด มีรายละเอียดสูง และมีสัดส่วนที่ถูกต้องเนื่องจากมอนิเตอร์มีความแบนราบจริง ช่วยถนอมสายตาและมีอัตราการแผ่รังสีที่เป็นอันตรายต่อสุขภาพต่ำมาก ประหยัดพลังงานไฟฟ้าด้วยการใช้พลังงานไฟฟ้าที่ต่ำกว่าจอ CDT ถึง 60 เปอร์เซ็นต์

ความสามารถในการรองรับอินพุต (Input) ได้หลายๆแบบพร้อมกันเนื่องด้วยมอนิเตอร์แบบ LCD สามารถรับสัญญาณจากแหล่งสัญญาณดิจิตอลอื่นๆได้ เช่น โทรทัศน์หรือเครื่องเล่นดีวีดี และบางรุ่นสามารถทำภาพซ้อนจากหลายแหล่งข้อมูลได้ จึงทำให้จอมอนิเตอร์แบบ LCD เป็นได้ทั้งเครื่องรับโทรทัศน์และจอมอนิเตอร์ในเวลาเดียวกัน โดยไม่จำเป็นต้องซื้อมอนิเตอร์หลายๆตัวมาใช้งาน

2.4.2 ประเภทของจอภาพแบบ LCD

จอภาพแบบ LCD ยังสามารถแบ่งออกได้เป็น 2 ประเภทคือ แบบ Passive Matrix หรือ DSTN และแบบ Active Matrix หรือ TFT จอภาพ LCD ประเภท Passive Matrix มีใช้มานานแล้ว ซึ่งจอภาพแบบนี้จะนิยมใช้เป็นหน้าปัดนาฬิกา รวมถึงจอแสดงผลของ โทรศัพท์เคลื่อนที่รุ่นเก่า ๆ ด้วยส่วนแบบ Active Matrix จะมีคุณภาพในการแสดงผลที่ดีกว่าและก็มีราคาแพงกว่า แบบ Passive Matrix จึงนิยมใช้ในการแสดงผลที่ต้องการคุณภาพสูง เช่น ทำจอภาพสำหรับเครื่องคอมพิวเตอร์ หรือจอภาพโฆษณาขนาดใหญ่ที่ติดตั้งตามอาคารสูงๆ เป็นต้น

ประเภท Passive Matrix หรือมีชื่อเรียกอีกอย่างหนึ่งว่า DSTN จอภาพ LCD แบบนี้จะมีการนำทรานซิสเตอร์เข้ามาช่วยในการทำงานบางส่วน เพื่อให้การแสดงผลทำได้ดีกว่าจอภาพ LCD ในรุ่นแรกๆ แต่อย่างไรก็ตามจอภาพประเภท DSTN นี้ก็ยังให้คุณภาพของการแสดงผลไม่ดีนัก เพราะการควบคุมการทำงานในแต่ละจุดของจอภาพยังทำได้ช้า ดังนั้น เมื่อเราใช้จอภาพแบบนี้เล่นเกมหรือดูภาพยนตร์ จอภาพจะไม่สามารถแสดงผลได้ทันกับความเร็วของภาพ ในปัจจุบันจึงไม่เป็นที่นิยมใช้กันแล้ว

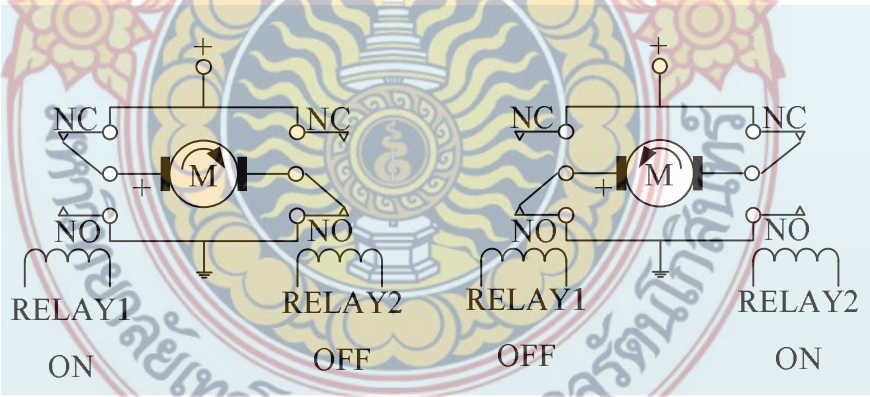
ประเภท Active Matrix หรือที่มีชื่อเรียกอีกอย่างหนึ่งว่าแบบ TFT เพราะจอภาพ LCD แบบนี้จะนำทรานซิสเตอร์แบบ Thin-Film มาใช้ควบคุมการแสดงผลในแต่ละจุดบนจอภาพ ซึ่งจะมีประสิทธิภาพในการควบคุมดีกว่าแบบ Passive Matrix มาก แต่ข้อเสียก็คือ การออกแบบที่ซับซ้อนมากขึ้นเพราะถ้าเราจะออกแบบ จอภาพ LCD แบบ Active Matrix ให้แสดงผลที่ความละเอียด 1,024x768x3 เพราะแต่ละจุดประกอบด้วยตัวควบคุมแม่สีแสง 3 สีก็คือ ต้องใช้ทรานซิสเตอร์ถึง 3 ตัวใน 1 จุด อีกทั้งการออกแบบก็ต้องใช้ความระมัดระวังเป็นอย่างมาก เพราะถ้ามีทรานซิสเตอร์เสียเพียงตัวเดียวจะส่งผลให้จุดบนจอภาพแสดงสีเพี้ยนหรือไม่ก็เป็นจุดดำมืดไปทันที

2.5 มอเตอร์กระแสตรง (DC MOTOR)

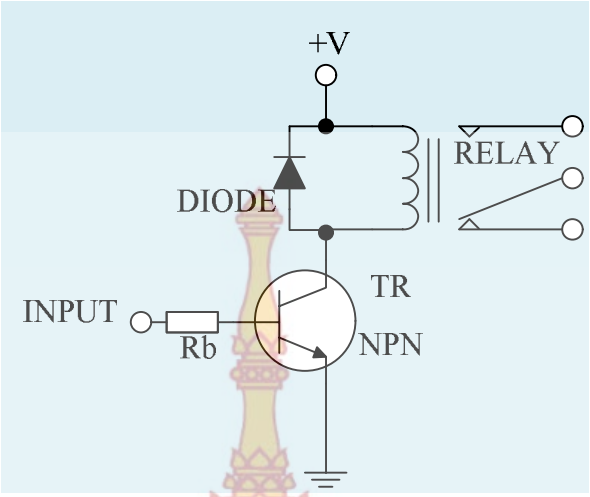
มอเตอร์กระแสตรงจะมีหลักการทำงานโดยวิธีการผ่านกระแสให้กับขดลวดในสนาม แม่เหล็ก ซึ่งจะทำให้เกิดแรงแม่เหล็ก โดยส่วนของแรงนี้จะขึ้นอยู่กับกระแสและกำลังของสนามแม่เหล็ก ทางเดินของฟลั๊กซ์แม่เหล็ก และสนามแม่เหล็กจะเกิดจากแท่งแม่เหล็กเฟอร์ไรต์ 2 ชิ้นที่ขึ้นรูปเป็นแบบโค้งยึดติดกับตัวถังได้พอดี เพื่อที่จะให้เส้นแรงแม่เหล็กวิ่งเข้าสู่ใจกลางของมอเตอร์ได้ ดังนั้นความเข้มของแม่เหล็กจะขึ้นอยู่กับขนาดความหนาของแม่เหล็กด้วย ซึ่งส่งผลให้ฟลั๊กซ์แม่เหล็กวิ่งไปบนตัวถังโลหะ กระแสไฟฟ้าในขดลวดที่พันกับทุ่นโรเตอร์ก็จะทำให้เกิดสนามแม่เหล็กไฟฟ้า และต้านกับสนามแม่เหล็กถาวร จึงเกิดเป็นแรงบิดเพื่อที่จะหมุนทุ่นโรเตอร์ ให้ไปในทิศทางเดียวกันกับทิศทางของสนามแม่เหล็กที่มีแรงมากกว่า กระแสก็จะไหลผ่านไปยังทุ่นโรเตอร์ โดยผ่านแปรงถ่าน ซึ่งจะสัมผัสกับแหวนตัวนำในทุ่นโรเตอร์ และแหวนคอมมิวเตเตอร์ ซึ่งจะถูกแบ่งออกเป็น 3 เซกเมนต์เพื่อที่จะทำหน้าที่นำกระแสเข้าขดลวดนั้นเอง

2.5.1 การขับและกลับทิศทางของมอเตอร์กระแสตรง (DC MOTOR)

ในการใช้ไอซีไมโครคอนโทรเลอร์เป็นตัวควบคุมการหมุน และทิศทางของมอเตอร์กระแสตรงนั้น เราจะต้องมีส่วนของวงจร ที่เรียกว่าวงจรขับมอเตอร์ (Driver) ในส่วนของวงจรกลับทิศทางของมอเตอร์นั้น สามารถที่จะใช้รีเลย์ต่อวงจร สวิตซ์เพื่อกลับทิศทางของขั้วไฟกระแสตรง หรืออาจใช้อุปกรณ์สารกึ่งตัวนำที่เป็นวงจรขับกำลังเช่น ทรานซิสเตอร์ มอสเฟต แล้วแต่วิธีที่เราจะเลือกใช้งานการใช้รีเลย์ควบคุมการ เปลี่ยนทิศทางการหมุนของมอเตอร์ โดยการควบคุมการปิด - เปิดที่รีเลย์ 2 ตัว ซึ่งจะทำหน้าที่กลับทิศทางของขั้วไฟที่ป้อนให้กับมอเตอร์ โดยการสลับการทำงานของรีเลย์ เช่นให้รีเลย์ตัวที่ 1 ทำงาน (ON) และรีเลย์ตัวที่ 2 หยุดทำงาน (OFF) จะทำให้มอเตอร์หมุนไปทางซ้าย และในทำนองเดียวกันถ้าหากรีเลย์ตัวที่ 1 หยุดทำงาน (OFF) และรีเลย์ตัวที่ 2 ทำงาน (ON) ก็จะทำให้มอเตอร์หมุนไปทางขวา

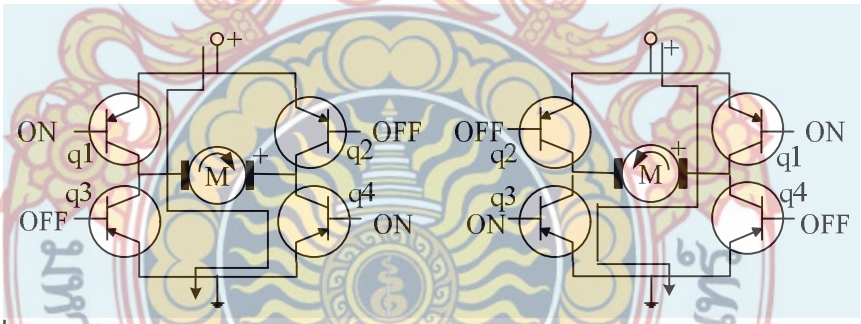


ภาพที่ 7 แสดงการกลับทิศทางของมอเตอร์กระแสตรงโดยใช้รีเลย์



ภาพที่ 8 แสดงการใช้ทรานซิสเตอร์เพื่อขับรีเลย์ให้ทำงาน

จากรูปเป็นวงจรขับรีเลย์โดยใช้ทรานซิสเตอร์ทำหน้าที่ขยายกระแส ด้วยเหตุผลเพราะไม่สามารถจะใช้ขา เอาต์พุตของไมโครคอนโทรลเลอร์ป้อนกระแสไฟที่ขดลวดของรีเลย์โดยตรงได้ เนื่องจากว่ากระแสที่จ่ายออกมาจากขา เอาต์พุตของไมโครคอนโทรลเลอร์มีค่าน้อยเกินไป ดังนั้นเราจึงต้องมีส่วนของวงจรทรานซิสเตอร์เพื่อที่จะทำการขยายกระแสให้ เพียงพอในการป้อนให้กับขดลวดของรีเลย์ ส่วนไดโอดนำมาต่อไว้สำหรับป้องกันแรงดันย้อนกลับที่เกิดจากการเหนี่ยวนำของสนามแม่เหล็กในขณะเกิดการยุบตัว ซึ่งอาจจะทำให้ทรานซิสเตอร์เสียหายได้



ภาพที่ 9 แสดงการใช้ทรานซิสเตอร์เป็นวงจรขับและกำหนดทิศทางของมอเตอร์กระแสตรง

จากรูปเป็นวงจรลิเนียร์บริดจ์แอมป์ ซึ่งจะประกอบไปด้วยทรานซิสเตอร์กำลัง 4 ตัวที่ทำหน้าที่ขับ และควบคุมทิศทางการหมุนของมอเตอร์ ถ้าหากกำหนดให้ทรานซิสเตอร์ Q1 และ Q4 อยู่ในสภาวะทำงาน (Active) กระแสไฟฟ้าจะไหลผ่านทรานซิสเตอร์จากซ้ายไปขวา โดยผ่านมอเตอร์กระแสตรงทำให้มอเตอร์หมุนไปทางขวา ในทำนองเดียวกันถ้าหากเราทำให้ทรานซิสเตอร์ Q2 และ Q3 อยู่ในสภาวะทำงาน (Active) กระแสไฟฟ้าก็จะไหลจากทางขวาไปทางซ้ายซึ่งจะส่งผลให้มอเตอร์กลับทิศการหมุนจากทางขวาไปทางซ้าย

2.5.2 การควบคุมความเร็วของมอเตอร์กระแสตรง

การควบคุมความเร็วของมอเตอร์กระแสตรงมีหลายวิธีด้วยกัน ซึ่งอาจจะใช้วิธีการควบคุมแบบพื้นฐานทั่วไปเช่นการควบคุมด้วยวิธีการใช้ตัว ต้านทานปรับค่าโดยต่ออนุกรมกับมอเตอร์ หรือใช้

วิธีการการควบคุมโดยการเปลี่ยนค่าของระดับแรงดันที่ป้อนให้กับ มอเตอร์ แต่การควบคุมในวิธีดังกล่าวถึงแม้ว่าจะควบคุมความเร็วมอเตอร์ให้คงที่ได้ แต่ที่ความเร็วต่ำจะส่งผลให้แรงบิดต่ำไปด้วย ดังนั้นเราจึงเลือกใช้วิธีการควบคุมโดยการจ่ายกระแสไฟให้กับมอเตอร์เป็น ช่วงๆ โดยอาศัยกระแสไฟที่ป้อนให้กับมอเตอร์ให้เป็นค่าเฉลี่ยที่เกิดขึ้นในแต่ละ ช่วง ซึ่งเราเรียกว่าวิธีการของการมอดูเลชั่นทางความกว้างของพัลส์ PWM (Pulse Width Modulation)

2.5.3 วิธีการมอดูเลชั่นทางความกว้างของพัลส์ (PWM)

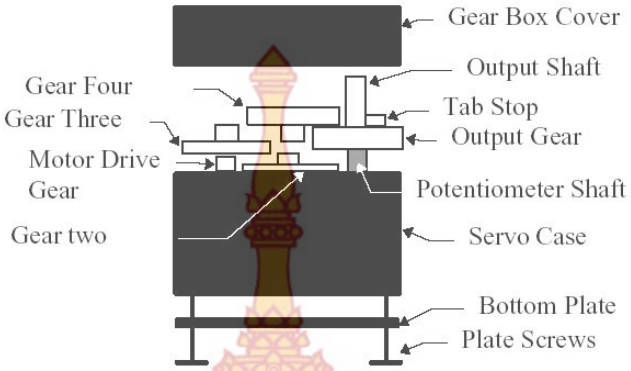
การมอดูเลชั่นทางความกว้างของพัลส์ PWM (Pulse Width Modulation) จะเป็นการปรับเปลี่ยนที่สัดส่วน และความกว้างของสัญญาณพัลส์ โดยความถี่ของสัญญาณพัลส์จะไม่มีการเปลี่ยนแปลง หรือเป็นการเปลี่ยนแปลงที่ค่าของดิวตี้ไซเกิล (duty cycle) นั้นเอง ซึ่งค่าของดิวตี้ไซเคิล คือช่วงความกว้างของพัลส์ที่มีสถานะลอจิกสูง โดยคิดสัดส่วนเป็นเปอร์เซตน์จากความกว้างของพัลส์ทั้งหมด ยกตัวอย่างเช่น ถ้าหากค่าดิวตี้ไซเคิลมีค่าเท่ากับเท่ากับ 50% ก็หมายถึงใน 1 รูปสัญญาณพัลส์จะมีช่วงของสัญญาณที่เป็นสถานะลอจิกสูงอยู่ครึ่งหนึ่ง และสถานะลอจิกต่ำอยู่อีกครึ่งหนึ่ง ดังรูป 6.27 และในทำนองเดียวกันถ้าหากค่าดิวตี้ไซเคิลมีค่ามาก หมายความว่าความกว้างของพัลส์ที่เป็นสถานะลอจิกสูงจะมีความกว้างมากขึ้น หากค่าดิวตี้ไซเคิลมีค่าเท่ากับ 100% ก็หมายความว่าจะไม่มีสถานะลอจิกต่ำเลย ซึ่งค่าดิวตี้ไซเคิลสามารถ จะหาได้จากค่าความสัมพันธ์ดังนี้

ค่าดิวตี้ไซเคิล = (ช่วงของสัญญาณพัลส์/คาบเวลาทั้งหมดของสัญญาณ) X 100%

2.6 เซอร์โวมอเตอร์

Servo motor คือ มอเตอร์ไฟฟ้ากระแสตรง (DC motor) ที่ถูกประกอบรวมกับ ชุดเกียร์ และ ส่วนควบคุม ต่างๆ ไว้ในโมดูลเดียวกัน หรือ ภายในกล่องพาสติกเดียวกัน โดยมอเตอร์ชนิดนี้จะมีสายต่อใช้งานเพียง 3 เส้นเท่านั้น คือ VCC,GNDและ สายสัญญาณควบคุม(Control Line) ซึ่งสามารถควบคุมให้มอเตอร์หมุนซ้าย หรือ ขวาได้จากสายสัญญาณเพียงเส้นเดียวโดยสัญญาณที่ใช้ควบคุมนี้จะเป็นสัญญาณ พัลส์วิดมอด (PWM) แบบ TTL Level ระดับแรงดันที่จ่ายให้มอเตอร์นี้จะอยู่ในช่วงประมาณ 4 ถึง 6 โวลท์ ขึ้นอยู่กับคุณสมบัติของมอเตอร์แต่ละตัว ข้อดีของมอเตอร์ชนิดนี้ก็คือ จะมีขนาดเล็กน้ำหนักเบา,ให้แรงบิตสูง ,กินพลังงานน้อย และ สามารถควบคุม ด้วยแรงดันลอจิกที่เป็น TTL ได้โดยตรงไม่จำเป็นต้องต่อวงจรขับ(Driver) อื่นๆ เพราะ มอเตอร์ชนิดนี้จะมีวงจรควบคุมบรรจุไว้ภายในอยู่แล้ว ซึ่งมอเตอร์ชนิดนี้สามารถควบคุมให้หมุนไปในตำแหน่ง หรือ ทิศทางองศาที่ต้องการได้ โดยอาศัยสัญญาณความกว้างพัลส์ ที่ป้อนให้มอเตอร์ แต่เซอร์โวมอเตอร์นี้จะหมุนได้แค่เพียงในช่วงประมาณ 180° หรือ ครึ่งรอบเท่านั้น หรือ บางรุ่นอาจหมุนได้ถึง 210° แต่จะไม่สามารถหมุนเป็นวงรอบได้เนื่องจากโครงสร้างภายในจะประกอบด้วย ตัวต้านทานชนิดปรับค่าได้ (VR) ที่ทำหน้าที่ตรวจสอบตำแหน่งการหมุนของมอเตอร์ และ ตัวต้านทานนี้จะถูกยึดติดกับแกนหมุนของมอเตอร์ ซึ่งจากการที่ตัวต้านทานปรับค่านี้ไม่สามารถหมุนเป็นวงรอบได้ ดังนั้น เซอร์โวมอเตอร์จึงถูกออกแบบให้หมุนได้เพียงแค่ประมาณ 180 องศา หรือ ครึ่งรอบเท่านั้น เพื่อป้องกันความเสียหายที่จะเกิดกับตัวต้านทานปรับค่าได้ แต่ถ้าหากเราต้องการให้มอเตอร์หมุนเป็นวงรอบ (360°) นั้นก็สามารถ

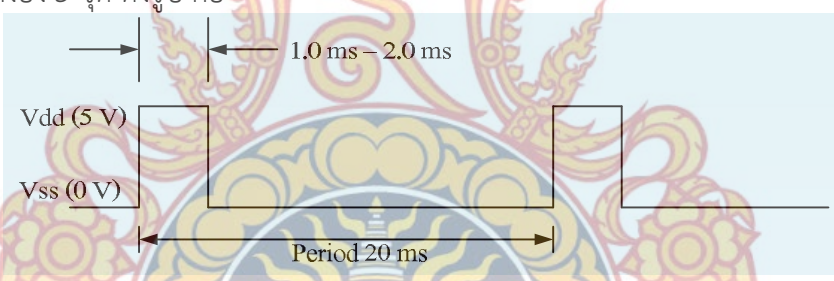
ทำได้ โดยจะต้องทำการปรับแต่ง (Modify) ดัดแปลงชิ้นส่วนบางอย่างของมอเตอร์ ซึ่งวิธีการต่างๆ จะได้กล่าวไว้ในภายหลัง



ภาพที่ 10 ส่วนประกอบต่างๆของ SERVOMOTOR

2.6.1 หลักการทำงานของ Servo motor

การควบคุมการทำงานของ เซอร์โวมอเตอร์ ทำได้โดย การป้อนสัญญาณความกว้างพัลส์ ให้กับมอเตอร์ซึ่งตำแหน่งและทิศทางการหมุนของมอเตอร์นี้จะขึ้นอยู่กับขนาดของความกว้างของพัลส์นั้นๆโดยทั่วไปแล้วความกว้างของสัญญาณพัลส์
จะมีจุดให้อ้างอิง 3 จุด ดังรูป คือ



ภาพที่ 11 PWMการทำงานของ SERVO

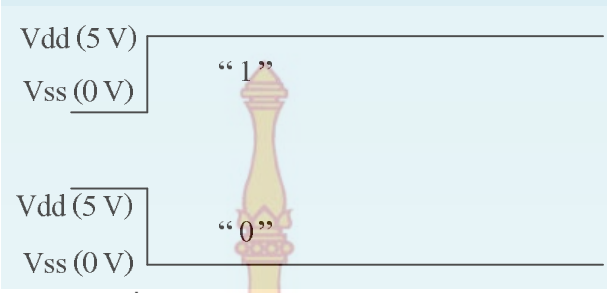
ถ้าสัญญาณความกว้างพัลส์ขนาด 1.5 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม 0 องศา หรือ จุดกึ่งกลางของมอเตอร์
ถ้าสัญญาณความกว้างพัลส์ขนาด 1 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม - 90 องศา หรือในทิศทางทวนเข็มนาฬิกา

ถ้าสัญญาณความกว้างพัลส์ขนาด 2 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม + 90 องศา หรือในทิศทางตามเข็มนาฬิกา

การควบคุมให้มอเตอร์หมุนทางด้านซ้ายจะต้องป้อนสัญญาณพัลส์ที่มีขนาดความกว้างพัลส์ 1 ms หรือ ให้น้อยกว่า 1.5 msโดยจะต้องป้อนสัญญาณพัลส์นี้ทุกๆ 20 ms (หรือในช่วงประมาณ 20ms – 30ms)

การควบคุมให้มอเตอร์หมุนทางด้านขวาจะต้องป้อนสัญญาณพัลส์ที่มีขนาดความกว้างพัลส์ 2 ms หรือ ไม่ต่ำกว่า 1.5 ms และจะต้องป้อนสัญญาณพัลส์นี้ทุกๆ 20 ms (หรือในช่วง

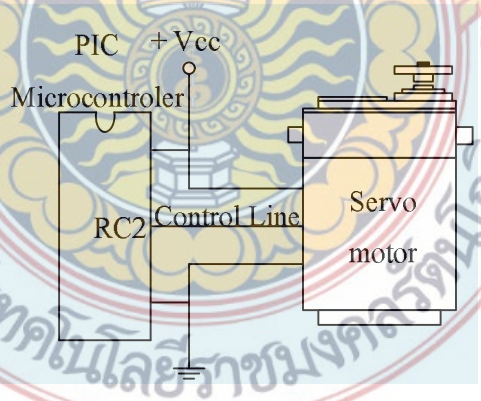
ประมาณ 20ms – 30ms) -การควบคุมให้มอเตอร์หยุดหมุน ทำได้โดยการส่งลอจิก “0” หรือ “1” ให้กับมอเตอร์ หรือ ก็คือการไม่จ่ายสัญญาณพัลส์ให้กับมอเตอร์นั่นเอง



ภาพที่ 12 การควบคุมให้มอเตอร์หยุดหมุน

2.6.2 ตัวอย่างการควบคุม Servo motor ด้วยไมโครคอนโทรลเลอร์ตระกูล PIC

ตัวอย่างโปรแกรมการควบคุม Servo motor ด้วยไมโครคอนโทรลเลอร์ตระกูล PIC เบอร์ 16F877 และ 18F458 โดยใช้บอร์ด CP-PIC V3.0 หรือ CP-PIC V4.0การควบคุมการทำงานของ Servo motor จะใช้หลักการสร้างสัญญาณพัลส์ขนาดความกว้างต่างๆ ส่งไปควบคุมการทำงานของมอเตอร์ ซึ่งในภาษาเบสิกนั้นจะใช้คำสั่ง PULSOUT Pin,Period เพื่อสร้างสัญญาณพัลส์ โดยการทำงานของคำสั่งนี้ค่า Period จะเปลี่ยนแปลงไปตามค่าของสัญญาณนาฬิกาที่จ่ายให้กับ CPU ทำให้คำสั่ง DEFINE OSC ไม่มีผลต่อการทำงานของคำสั่งนี้ เช่น ถ้า CPU ใช้ความถี่ 4 MHz จะทำให้หนึ่งหน่วยของค่า Period = 10 us ดังนั้นหากใช้คำสั่ง PULSOUT Pin,100ก็จะได้ค่าเวลาเท่ากับ 100 x 10us = 1000us หรือ 1 ms แต่ในตัวอย่างโปรแกรมนี้จะใช้งาน CPU ที่ความถี่ 10 MHz ซึ่งค่าเวลาต่อหน่วยของ Period จะเท่ากับ 4 us ดังนั้นถ้าหากต้องการเวลา 1ms ค่าของ Period จะเท่ากับ 250 คือ 4us x 250 = 1000 usและ คำสั่งที่ใช้ก็จะเป็น PULSOUT Pin,250 เป็นต้น โดยสามารถทดสอบด้วยการเปลี่ยนค่าเวลาเป็นค่าต่างๆ ดังโปรแกรม



ภาพที่ 13 การต่อใช้ PIC กับSERVOMOTOR

ตารางที่ 2 ตัวอย่างโปรแกรมภาษาเบสิก

ตัวอย่างโปรแกรมภาษาเบสิก
'/**
'/* Program : Control DC servo motor
'/* Filename : ServoMotor.bas
'/* CPU Control : PIC 16F877 or 18F458
'/* OSC : 10 MHz [HS mode]
'/* Assembler : PicBasicPro 2.41
'/**
INCLUDE "modedefs.bas" ' Include serial modes
TRISC = %00000000 ' PORTC is output
LOW PORTC.2
Loop: PULSOUT PORTC.1,250
PAUSE 20
goto Loop

2.7 สวิตชิ่งเพาเวอร์ซัพพลาย

สวิตชิ่งเพาเวอร์ซัพพลาย (Switching Power Supply) เป็นแหล่งจ่ายไฟตรงคงค่าแรงดันแบบหนึ่ง และสามารถเปลี่ยนแรงดันไฟจากไปสลับโวลต์สูง ให้เป็นแรงดันไฟตรงค่าต่ำ เพื่อใช้ในงานอิเลคทรอนิกส์ได้เช่นเดียวกันแหล่งจ่ายไฟเชิงเส้น (Linear Power Supply) ถึงแม้เพาเวอร์ซัพพลายทั้งสองแบบจะต้องมีการใช้หม้อแปลงในการลดทอนแรงดันสูงให้เป็นแรงดันต่ำเช่นเดียวกัน แต่สวิตชิ่งเพาเวอร์ซัพพลายจะต้องการใช้หม้อแปลงที่มีขนาดเล็ก และน้ำหนักน้อย เมื่อเทียบกับแหล่งจ่ายไฟเชิงเส้น อีกทั้งสวิตชิ่งเพาเวอร์ซัพพลายยังมีประสิทธิภาพสูงกว่าอีกด้วย

ในปัจจุบันสวิตชิ่งเพาเวอร์ซัพพลายได้เข้ามามีบทบาทกับชีวิตเราอย่างมาก เครื่องใช้อิเลคทรอนิกส์ขนาดเล็กซึ่งต้องการแหล่งจ่ายไฟที่มีกำลังสูงแต่มีขนาดเล็ก เช่น เครื่องคอมพิวเตอร์ เครื่องโทรสาร และ โทรทัศน์ จำเป็นจะต้องใช้สวิตชิ่งเพาเวอร์ซัพพลาย แนวโน้มการนำสวิตชิ่งเพาเวอร์ซัพพลายมาใช้ในเครื่องใช้อิเลคทรอนิกส์ทุกประเภทจึงเป็นไปได้สูง การศึกษาหลักการทำงานและการออกแบบสวิตชิ่งเพาเวอร์ซัพพลายจึงเป็นสิ่งจำเป็นที่ไม่อาจหลีกเลี่ยงได้สำหรับผู้ที่เกี่ยวข้องกับงานอิเล็คทรอนิกส์ทุกประเภท

หลักการทำงานเบื้องต้นของสวิตชิ่งเพาเวอร์ซัพพลาย โดยเน้นในส่วนของคอนเวอร์เตอร์ และวงจรควบคุม ซึ่งเป็นหัวใจในการทำงานของสวิตชิ่งเพาเวอร์ซัพพลาย พร้อมทั้งยกตัวอย่างและอธิบายการทำงานของวงจรสวิตชิ่งเพาเวอร์ซัพพลายที่สมบูรณ์ และใช้งานได้จริง

2.7.1 สวิตชิ่งเพาเวอร์ซัพพลายกับแหล่งจ่ายไฟเชิงเส้น

ข้อเปรียบเทียบของสวิตชิ่งเพาเวอร์ซัพพลายเมื่อเปรียบเทียบกับแหล่งจ่ายไฟ เชิงเส้น คือ ประสิทธิภาพที่สูง ขนาดเล็ก และน้ำหนักเบากว่าแหล่งจ่ายไฟเชิงเส้น เนื่องจากแหล่งจ่ายไฟเชิงเส้นใช้

หม้อแปลงความถี่ต่ำจึงมีขนาดใหญ่ และน้ำหนักมาก ขณะใช้งานจะมีแรงดันและกระแสผ่านตัวหม้อแปลงตลอดเวลา กำลังงานสูญเสียที่เกิดจากหม้อแปลงจึงมีค่าสูง การคงค่าแรงดันแหล่งจ่ายไฟเชิงเส้นส่วนมากจะใช้เพาเวอร์ทรานซิสเตอร์ต่ออนุ กรมที่เอาต์พุตเพื่อจ่ายกระแสและคงเค่าแรงดัน กำลังงานสูญเลียในรูปความร้อนจะมีค่าสูงและต้องใช้แผ่นระบายความร้อนขนาด ใหญ่ซึ่งกินเนื้อที่ เมื่อเพาเวอร์ซัพพลายต้องจ่ายกำลังงานสูงๆ จะทำให้มีขนาดใหญ่และมีน้ำหนักมาก ปกติแหล่งจ่ายไฟเชิงเส้นจะมีประสิทธิภาพประมาณ 30% หรืออาจทำได้สูงถึง 50% ในบางกรณี ซึ่งนับได้ว่าค่อนข้างต่ำเมื่อเปรียบเทียบกับสวิตชิ่งเพาเวอร์ซัพพลายซึ่งมี ประสิทธิภาพในช่วง 65%-80%

สวิตชิ่งเพาเวอร์ซัพพลายมีช่วงเวลาโคลสต์อัพประมาณ $20x10^{-3}$ ถึง $50x10^{-3}$ วินาที ในขณะที่แหล่งจ่ายไฟเชิงเส้นจะทำได้เพียงประมาณ $2x10^{-3}$ วินาที ซึ่งมีผลต่อการจัดหาแหล่งจ่ายไฟสำรองเพื่อป้องกันการหยุดทำงานของอุปกรณ์ที่ใช้กับเพาเวอร์ซัพพลายเมื่อเกิดการหยุดจ่ายแรงดันไฟสลับ รวมทั้งสวิตชิ่งเพาเวอร์ซัพพลายสามารถทำงานได้ในช่วงแรงดันอินพุตค่อนข้างกว้างจึงยังคงสามารถทำงานได้เมือเกิดกรณีแรงดันไฟอีกด้วย อย่างไรก็ตาม สวิตชิ่งเพาเวอร์ซัพพลายจะมีเสถียรภาพในการทำงานที่ต่ำกว่า และก่อให้เกิดสัญญาณรบกวนได้สูงเมื่อเปรียบเทียบกับแหล่งจ่ายไฟเชิงเส้น รวมทั้งสวิตชิ่งเพาเวอร์ซัพพลายยังมีความซับซ้อนของวงจรมากกว่าและมีราคาสูง ที่กำลังงานต่ำๆ แหล่งจ่ายไฟเชิงเส้นจะประหยัดกว่าและให้ผลดีเท่าเทียมกัน ดังนั้นสวิตชิ่งเพาเวอร์ซัพพลายจึงมักนิยมใช้กันในงานที่ต้องการกำลังงานตั้งแต่ 20 วัตต์ขึ้นไปเท่านั้น

2.7.2 หลักการทำงานของสวิตชิ่งเพาเวอร์ซัพพลาย

สวิตชิ่งเพาเวอร์ซัพพลายโดยทั่วไปมีองค์ประกอบพื้นฐานที่คล้ายคลึงกัน และไม่ซับซ้อนมากนัก ดังแสดงในรูปที่ 1 หัวใจสำคัญของสวิตชิ่งเพาเวอร์ซัพพลายจะอยู่ที่คอนเวอร์เตอร์ เนื่องจากทำหน้าที่ทั้งลดทอนแรงดันและคงค่าแรงดันเอาต์พุตด้วย องค์ประกอบต่างๆ ทำงานตามลำดับ

แรงดันไฟสลับค่าสูงจะผ่านเข้ามาทางวงจร RFI ฟิลเตอร์ เพื่อกรองสัญญาณรบกวนและแปลงเป็นไฟตรงค่าสูงด้วยวงจรเรกติไฟเออร์ เพาเวอร์ทรานซิสเตอร์จะทำงานเป็นเพาเวอร์คอนเวอร์เตอร์โดยการตัดต่อแรงดัน เป็นช่วงๆ ที่ความถี่ประมาณ 20-200 KHz จากนั้นจะผ่านไปยังหม้อแปลงสวิตชิ่งเพื่อลดแรงดันลง เอาต์พุตของหม้อแปลงจะต่อกับวงจรเรียงกระแส และกรองแรงดันให้เรียบ การคงค่าแรงดันจะทำได้โดยการป้อนกลับคาแรงดันที่เอาต์พุตกลับมายังวงจรควบ คุม เพื่อควบคุมให้เพาเวอร์ทรานซิสเตอร์นำกระแสมากขึ้นหรือน้อยลงตามการเปลี่ยน แปลงของแรงดันที่เอาต์พุต ซึ่งจะมีผลทำให้แรงดันเอาต์พุตคงที่ได้

2.7.3 คอนเวอร์เตอร์

คอนเวอร์เตอร์นับว่าเป็นส่วนสำคัญที่สุดในสวิตชิ่งเพาเวอร์ซัพพลาย มีหน้าที่ลดทอนแรงดันไฟตรงค่าสูงลงมาเป็นแรงดันไฟตรงค่าต่ำ และสามารถคงค่าแรงดันได้ คอนเวอร์เตอร์มีหลายแบบขึ้นอยู่กับลักษณะการจัดวงจรภายใน โดยคอนเวอร์เตอร์แต่ละแบบจะมีข้อดีข้อเสียที่แตกต่างกันออกไป การจะเลือกใช้คอนเวอร์เตอร์แบบใดสำหรับสวิตชิ่งเพาเวอร์ซัพพลายนั้นมีข้อควรพิจารณาจากลักษณะพื้นฐานของคอนเวอร์เตอร์แต่ละแบบดังนี้คือ

ลักษณะการแยกกันทางไฟฟ้าระหว่างอินพุตกับเอาต์พุตของคอนเวอร์เตอร์
ค่าแรงดันอินพุตที่จะนำมาใช้กับคอนเวอร์เตอร์
ค่ากระแสสูงสุดที่ไหลผ่านเพาเวอร์ทรานซิสเตอร์ในคอนเวอร์เตอร์ขณะทำงาน
ค่าแรงดันสูงสุดที่ตกคร่อมเพาเวอร์ทรานซิสเคอร์ในคอนเวอร์เตอร์ขณะทำงาน
การรักษาระดับแรงดันในกรณีที่คอนเวอร์เตอร์มีเอาต์พุตหลายค่าแรงดัน
การกำเนิดสัญญาณรบกวน RFI/EMI ของคอนเวอร์เตอร์

จากข้อพิจารณาดังกล่าว จะทำให้ผู้ออกแบบทราบขีดจำกัดของคอนเวอร์เตอร์และตัดสินใจเลือกใช้คอนเวอร์เตอร์แบบใดได้ ปัจจุบันได้มีการพัฒนาคอนเวอร์เตอร์ในรูปแบบต่างๆ ขึ้นมามากมาย ในที่นี้จะกล่าวถึงเฉพาะคอนเวอร์เตอร์ที่นิยมใช้เป็นในอุตสาหกรรมของสวิตชิ่งเพาเวอร์ซัพพลาย คือ

ฟลายแบคคอนเวอร์เตอร์ (Flyback converter)
ฟอร์เวิร์ดคอนเวอร์เตอร์ (Forward converter)
พุช-พูลคอนเวอร์เตอร์ (Push-Pull converter)
ฮาล์ฟบริดจ์คอนเวอร์เตอร์ (Half-Bridge converter)
ฟูลบริดจ์คอนเวอร์เตอร์ (Full-Bridge converter)

คอนเวอร์เตอร์ทั้ง 5 แบบนี้ มีลักษณะการทำงานที่ไม่แตกต่างกันจนเกินไปนัก และค่อนข้างง่ายต่อการทำความเข้าใจและศึกษา คอนเวอร์เตอร์เหล่านี้ยังสามารถแบ่งย่อยได้อีกหลายประเภทโดยการเพิ่มเทคนิคบางประการให้กับคอนเวอร์เตอร์ ในที่นี้จะกล่าวถึงแต่เพียงการทำงานพื้นฐานเท่านั้น

บทที่ 3
วิธีการทดลอง/ระเบียบวิธีวิจัย

ขั้นตอนการดำเนินงาน ออกแบบเครื่องคัดแยกสีและนับจำนวนมะนาว สามารถแบ่งขั้นตอนการดำเนินงานออกเป็นขั้นตอนดังต่อไปนี้

3.1 รวบรวมข้อมูลและวิเคราะห์ข้อมูล
3.2 ออกแบบและสร้างตัวเครื่องส่วนกลไก
3.3 ออกแบบและสร้างส่วนวงจรควบคุมทั้งหมด
3.4 ออกแบบโฟลชาร์ตการทำงานของซอร์ฟแวร์
3.5 เขียนและพัฒนาโปรแกรมควบคุมตามโฟลชาร์ต

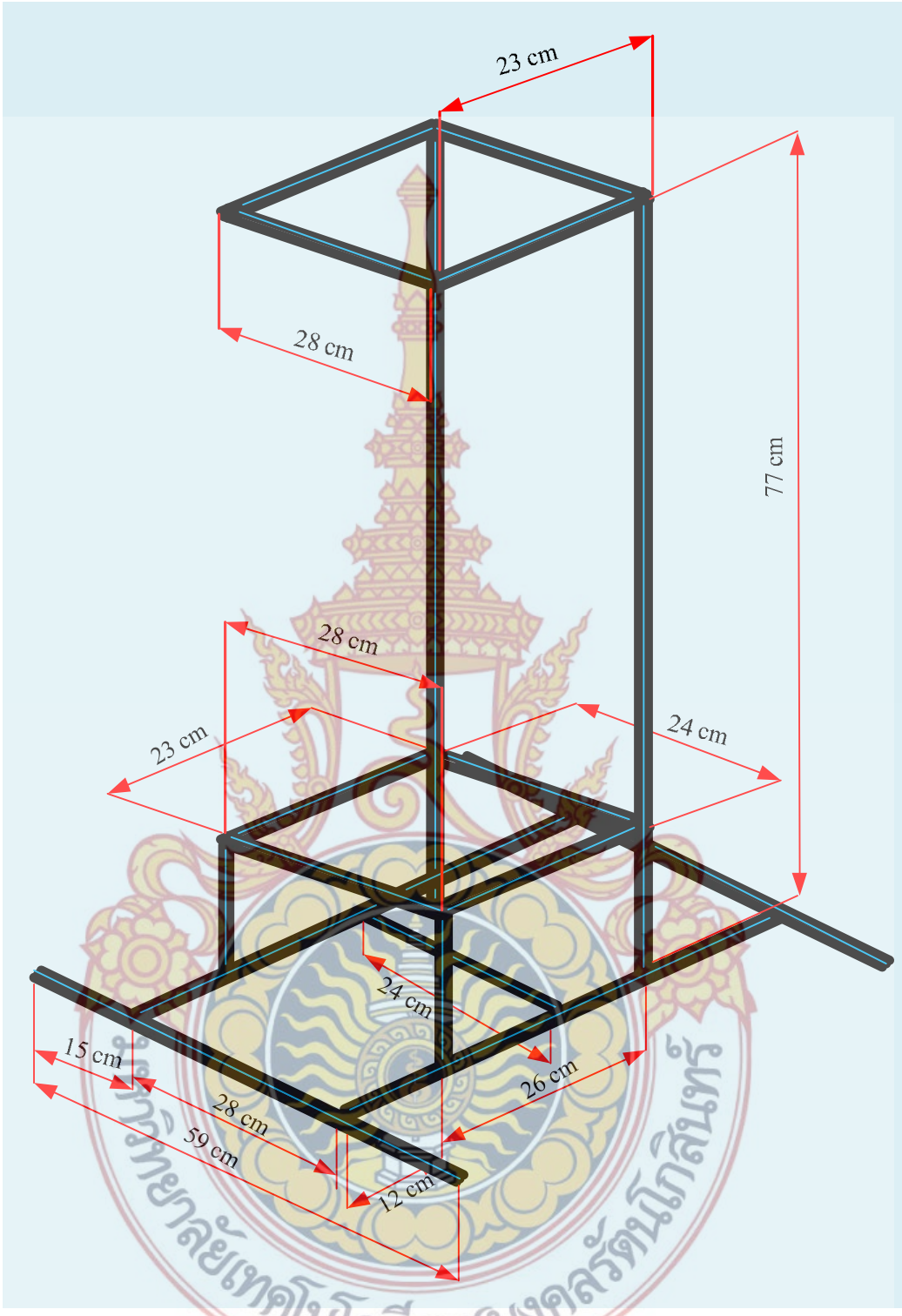
3.1 รวบรวมข้อมูลและวิเคราะห์ข้อมูล

การวิเคราะห์และรวบรวมข้อมูล เครื่องคัดแยกสีและนับจำนวนลูกมะนาว ซึ่งในรายงานฉบับนี้ได้รวบรวมเนื้อหาเกี่ยวกับการออกแบบวิธีการสร้างเครื่องคัดแยกสีและนับจำนวนมะนาว โดยการใช้ไมโครคอนโทรลเลอร์ควบคุมการทำงานส่วนของเอาท์พุตซึ่งประกอบด้วยมอเตอร์และจอแสดงผลแอลซีดีและส่วนการทำงานของอินพุตซึ่งประกอบด้วยเซนเซอร์และสวิตซ์ที่จะสั่งการให้ไมโครคอนโทรลเลอร์ทำงานตามข้อมูลการเขียนโปรแกรมภาษาซี โดยจะสั่งการทำงานของระบบด้วยไมโครคอนโทรลเลอร์ตระกูล PIC เบอร์ 16F877A เพื่อสั่งการควบคุม

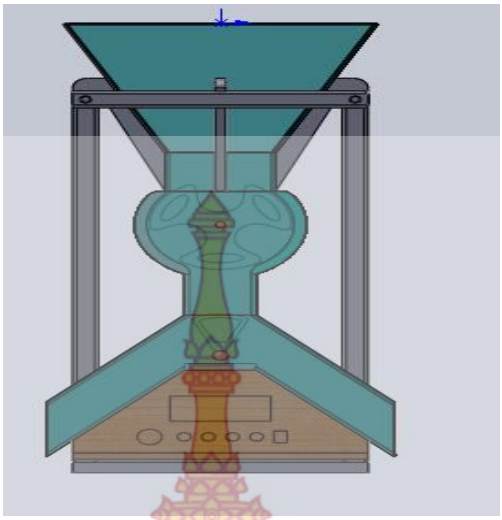
3.2 ออกแบบและสร้างตัวเครื่องส่วนกลไก

3.2.1 การออกแบบโครงสร้าง

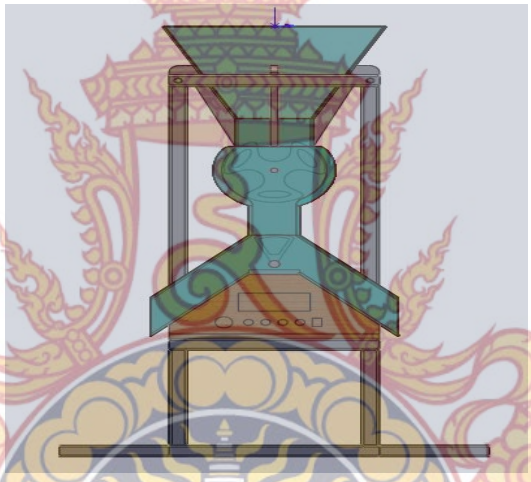
- การออกแบบชุดขาตั้งรองรับตัวเครื่องเครื่องคัดแยกสีและนับจำนวนมะนาว
- การออกแบบส่วนของตัวเครื่องคัดแยกสีและนับจำนวนมะนาว
- การออกแบบส่วนของกล่องใส่ชุดวงจรควบคุม



ภาพที่ 14 ขาตั้งตัวเครื่อง

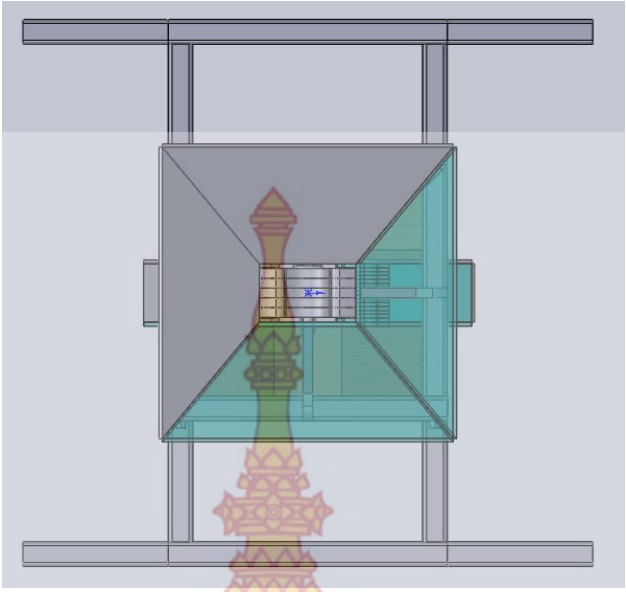


ภาพที่ 15 ส่วนของตัวเครื่องและกล่องใส่ชุดวงจรควบคุม

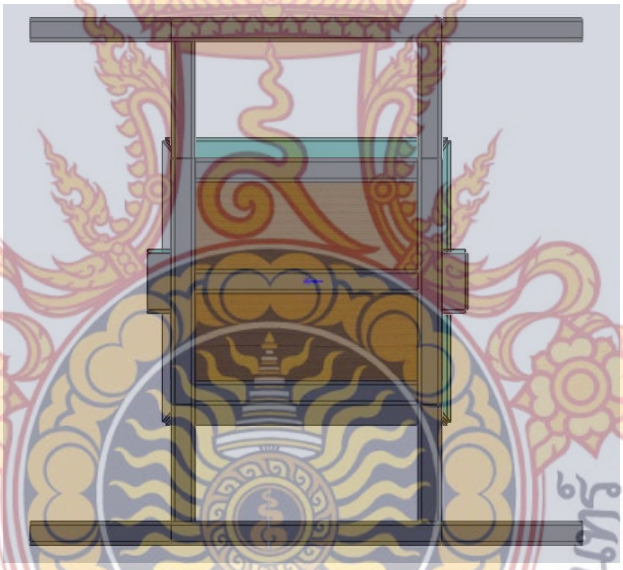


ภาพที่ 16 ด้านหน้า

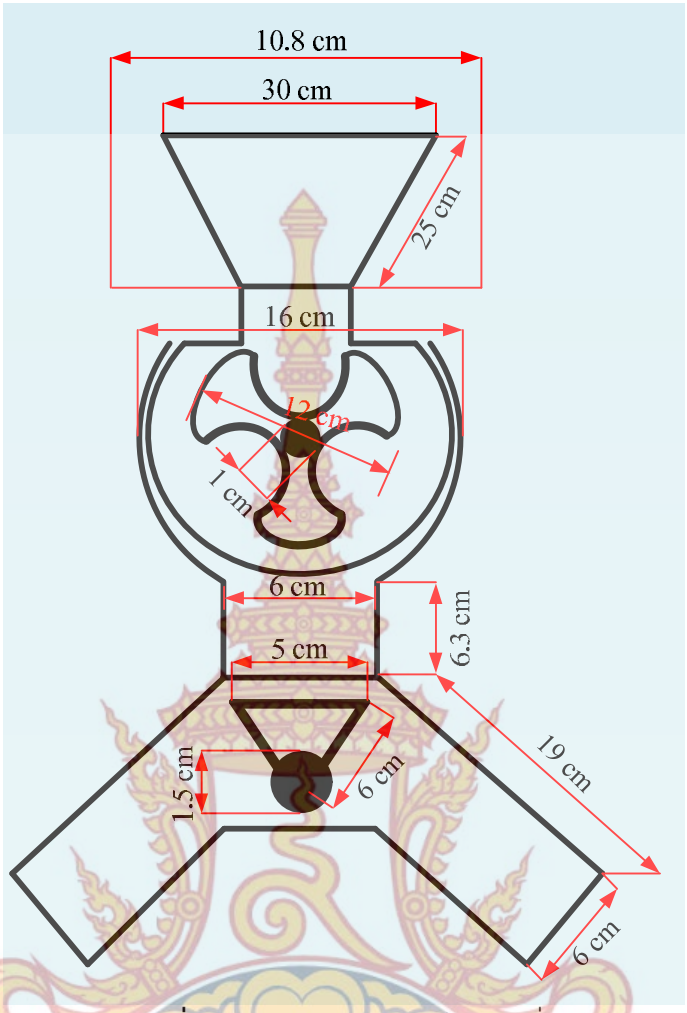
ภาพที่ 17 ด้านข้าง



ภาพที่ 18 ด้านบน



ภาพที่ 19 ด้านล่าง



ภาพที่ 20 ส่วนของขนาดภายในตัวเครื่อง

3.2.2 มอเตอร์ไฟฟ้ากระแสตรง

มอเตอร์ไฟฟ้ากระแสตรงมีคุณสมบัติคือ มีขนาดแรงดันไฟฟ้า 12 V ความเร็วรอบ 200 รอบต่อนาที นำมาต่อกับแกนหมุนในการควบคุมการตกของลูกมะนาว

ภาพที่ 21 มอเตอร์ไฟฟ้ากระแสตรง

3.2.3 เซอร์โวมอเตอร์

เซอร์โวมอเตอร์ ที่ถูกประกอบรวมกับชุดเกียร์และส่วนควบคุมต่างๆไว้ในโมดูลเดียวกัน หรือ ภายในกล่องพาสติกเดียวกัน โดยมอเตอร์ชนิดนี้จะมีสายต่อใช้งานเพียง 3 เส้นเท่านั้น คือ VCC,GNDและ สายสัญญาณควบคุม(Control Line) ซึ่งสามารถควบคุมให้มอเตอร์หมุนซ้าย หรือ ขวาได้จากสายสัญญาณเพียงเส้นเดียวโดยสัญญาณที่ใช้ควบคุมนี้จะเป็นสัญญาณ พัลส์วิดมอดูเลท (PWM)

ภาพที่ 22 เซอร์โวมอเตอร์

3.3 ออกแบบและสร้างส่วนวงจรควบคุม

3.3.1 การกำหนดอินพุตและเอาต์พุต

การกำหนดอินพุตและเอาท์พุตนั้นจะทำให้สะดวกต่อการออกแบบ ดังแสดงในตารางที่ 3 มีดังนี้

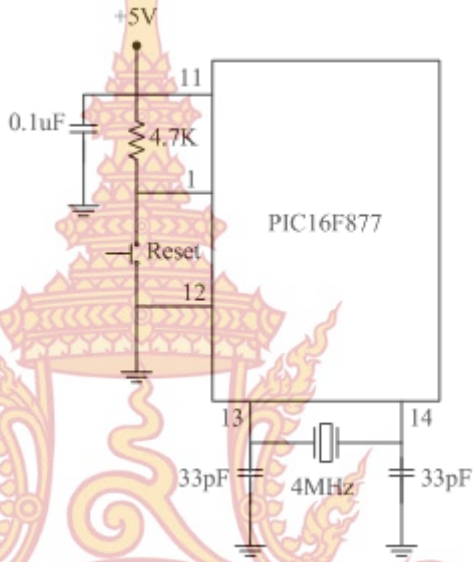
ตารางที่ 3 การแบ่งแยกอินพุตและเอาท์พุต

Input			Output		
การทำงาน	ชนิด I/P	จำนวน bit	การทำงาน	ชนิด O/P	จำนวน bit
สวิตซ์ Start/Stop	Digital	1	มอเตอร์กระแสตรง	Digital	1
สวิตซ์ Ok	Digital	1	เซอร์โวมอเตอร์	Digital	1
สวิตซ์ Cancel	Digital	1	แสดงผล	Digital	1
สวิตซ์ Up	Digital	1			
สวิตซ์ Down	Digital	1			
รับข้อมูลจาก sensor	Analog	4			
รับข้อมูลอินฟาเรท	Analog	1			
รวมจำนวน Input		10	รวมจำนวน Output		3

3.3.2 เลือกไมโครคอนโทรลเลอร์

การออกแบบระบบควบคุมการคัดแยกสีและนับจำนวนมะนาว ในงานวิจัยนี้จะเลือกใช้ไมโครคอนโทรลเลอร์ตระกูล PIC เบอร์ 16F877A เพื่อใช้ในการประมวลผลรับส่งข้อที่ได้จากการกดสวิตซ์และการตรวจวัดของเซนเซอร์ซึ่งติดต่อสื่อสารแบบ I2C

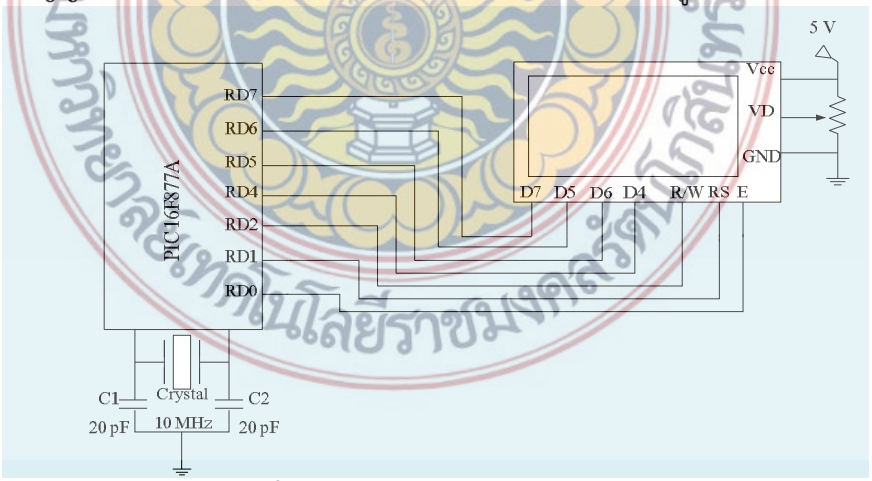
การออกแบบวงจรในงานวิจัยนี้ใช้คริสตัลที่มีความถี่ 4 MHz แต่จริงๆแล้วไมโครคอนโทรลเลอร์ตระกูล PIC-16F877 สามารถใช้ค่าสูงสุดเท่ากับ 20 MHz ซึ่งออสซิลเลเตอร์แบบต่างๆ จะสามารถกำหนดในโปรแกรมเพื่อให้ไมโครคอนโทรลเลอร์สามารถทำงานได้



ภาพที่ 23 วงจรพื้นฐานไมโครคอนโทรลเลอร์ตระกูล PIC เบอร์ 16F877

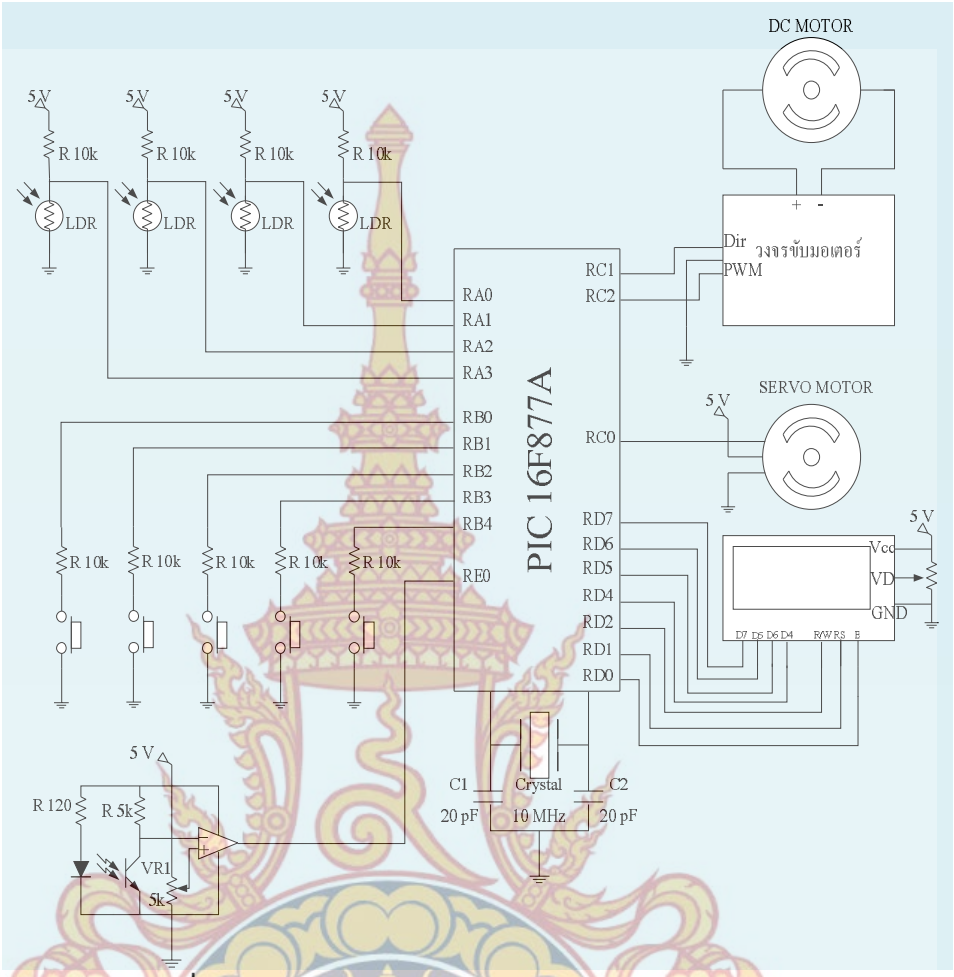
3.3.3 การแสดงสถานะจำนวน

ปริญญานิพนธ์นี้ออกแบบให้มีการแสดงค่าของกำหนดจำนวนลูกมะนาว



ภาพที่ 24 การแสดงผลจำนวนลูกมะนาว

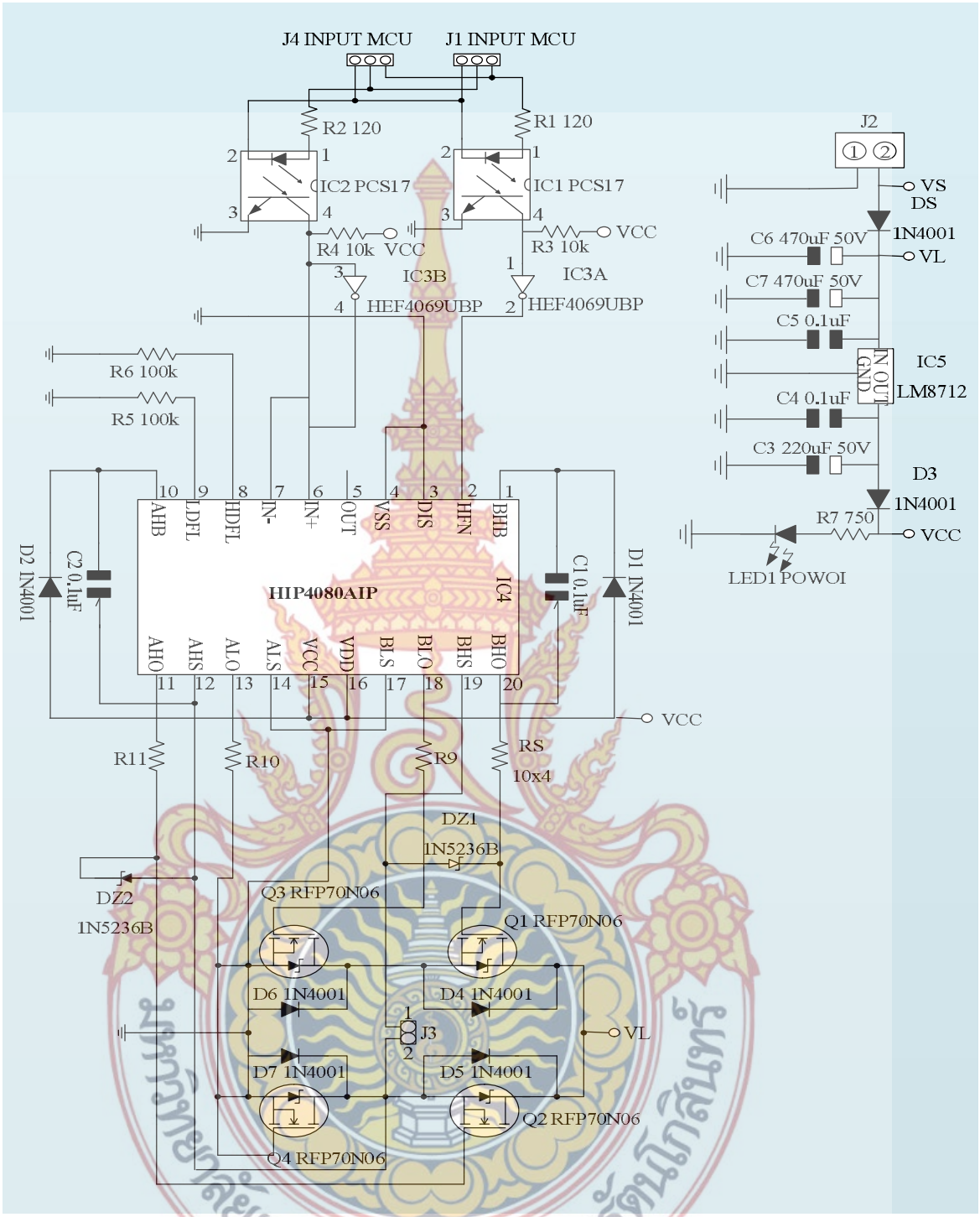
3.3.4 การต่อใช้งานวงจรร่วมกับ PIC 16F877



ภาพที่ 25 แสดงการต่อใช้งานวงจรร่วมกับ PIC16F877

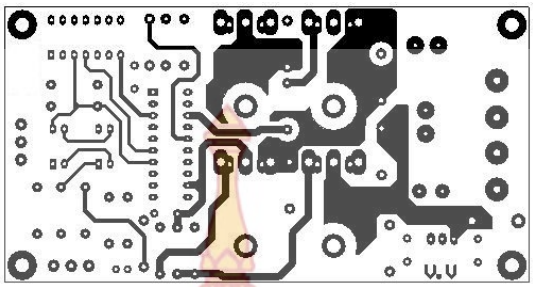
3.3.5 วงจรขับมอเตอร์

การสร้างวงจรขับมอเตอร์ เนื่องจากสัญญาณที่ได้จากไมโครคอนโทรลเลอร์มีค่าน้อยจึงต้องใช้วงจรขับแรงดันช่วยเพิ่มค่าแรงดันและกระแสให้เพียงพอสำหรับการขับ มอเตอร์ไฟฟ้ากระแสตรง ซึ่งวงจรขับแรงดันนี้สามารถให้กระแสได้สูงถึง 500 mA และแรงดัน 12 V

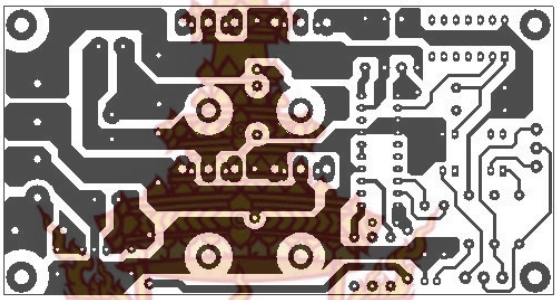


ภาพที่ 26 ลักษณะการต่อวงจรขับมอเตอร์

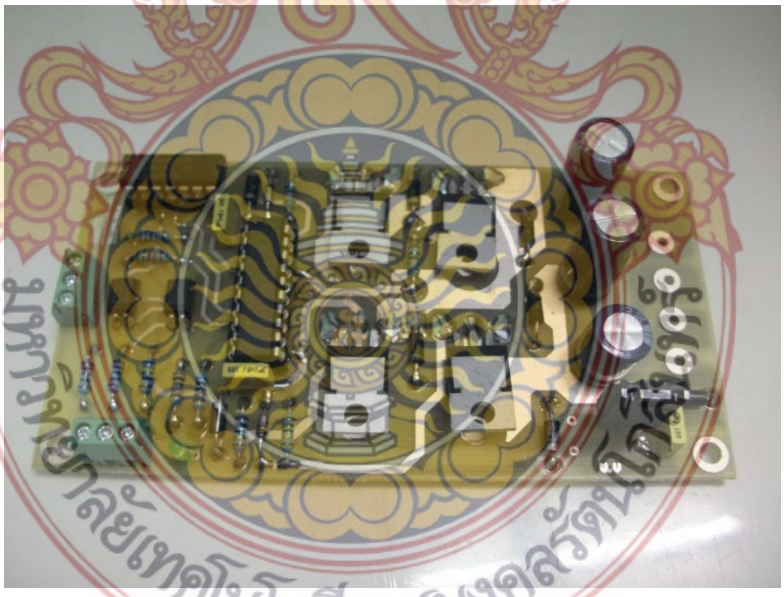
3.3.6 ออกแบบลายวงจรพิมพ์และสร้างแผ่นวงจรพิมพ์



ภาพที่ 27 PCBด้านบน



ภาพที่ 28 PCBด้านล่าง

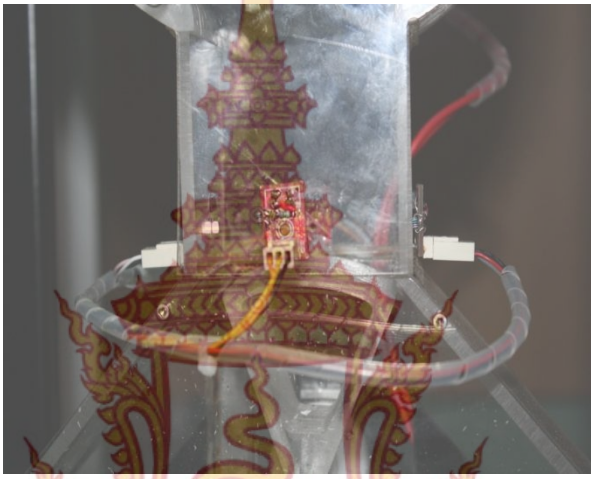


ภาพที่ 29 บอร์ดขับมอเตอร์

บทที่ 4
ผลการทดลอง

4.1 การทดลองการทำงานของอุปกรณ์แสดงผล

การแสดงผลของเครื่องคัดแยกสีนั้น จะใช้เซนเซอร์ในการตรวจจับสีและไมโครคอนโทรเลอร์ในการประมวลผลจากนั้นก็จะสั่งงานไปยังมอเตอร์เพื่อทำการคัดแยกสีและแสดงผลในจอ LCD ดังแสดงในภาพที่ 30



ภาพที่ 30 แสดงการติดตั้งเซนเซอร์ในการจับสี

ภาพที่ 31 การแสดงผลของ LCD

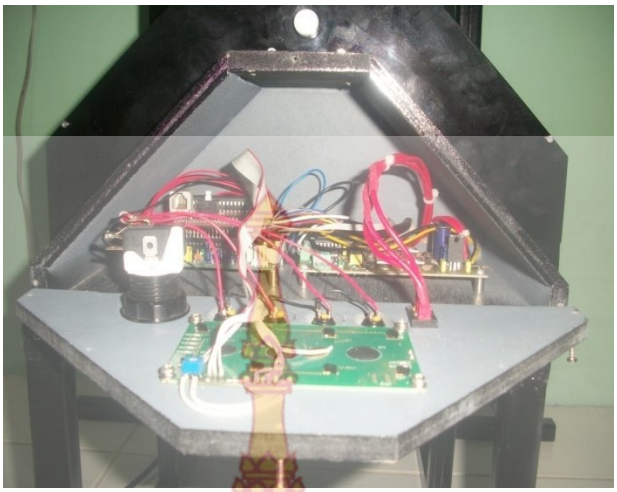
4.2 การทดลองโปรแกรมและการสั่งงานด้วยสวิตช์

ผลการทดลองการคัดแยกสีและนับจำนวนมะนาว ด้วยการเลือกสีและจำนวนโดยการกดสวิตช์เพื่อกำหนดค่าจำนวนมะนาวที่ต้องการ โดยใช้สวิตช์ชนิดกดติดปล่อยดับ สามารถควบคุมการทำงานได้ เช่น ถ้าต้องการมะนาวสีเขียว 10 ลูก ก็กดปุ่ม Ok และ Cancel พร้อมกันค้างไว้ประมาณสามวินาที จะเข้าสู่โปรแกรมการเลือกสีและจำนวน จากนั้นก็กดปุ่ม up หรือ down เพื่อเลือกสีและจำนวน และกดปุ่ม Ok ก็เข้าสู่โปรแกรม และกดปุ่ม Start/Stop เพื่อเริ่มและหยุดโปรแกรมมีการแสดงค่าของสีและจำนวนมะนาวบนหน้าจอ LCD แสดงในภาพที่ 32

ภาพที่ 32 การทดลองโปรแกรมและการสั่งงานด้วยสวิตช์

4.3 ผลการทดลองของภาคขับมอเตอร์

การทดลองของภาคขับมอเตอร์ เมื่อนำชุดควบคุมควบคุมมอเตอร์ต่อเข้ากับชุดควบคุมคอนโทรลเลอร์และมอเตอร์ แล้วทำการกำหนดค่าสีและจำนวนมะนาวสั่งงานด้วยสวิตช์ ผลที่ได้คือภาคขับมอเตอร์จะทำงานหลังจากกดสวิตช์ส่งค่าแล้วจะแสดงค่าที่ส่งทางจอ LCD โดยสามารถควบคุมการทำงานของมอเตอร์ได้ ซึ่งในการทดลองนี้ได้มะนาวสีเขียวและสีเหลืองมาทดลอง คือเมื่อมีมะนาวสีเขียวและสีเหลืองตกลงมาก็จะสามารถคัดสี และนับผลมะนาวได้ ดังแสดงในภาพที่ 35



ภาพที่ 33 แสดงการทำงานภาคขับมอเตอร์

ภาพที่ 34 การทดลองการหมุนของมอเตอร์

ภาพที่ 35 แสดงการทำงานเมื่อคัดสีมะนาว

4.4 ผลการทดลองความถูกต้องในการคัดแยกมะนาว

ทำการทดลองโดยใช้ตัวอย่างมะนาว สีเขียว และสีเหลือง อย่างละ 20 ลูก ทำการทดลองให้เครื่องทำการแยกสีจำนวน 10 ครั้ง เพื่อดูความถูกต้องในการคัดแยก และดูความเร็วในการคัดแยก ผลที่ได้ เป็นดังตารางที่ 4-1

ตารางที่ 4 ผลความถูกต้องในการคัดแยกสีมะนาว

ครั้งที่ทดลอง	มะนาวสีเขียว (ลูก)	มะนาวสีเหลือง (ลูก)	จำนวนลูกที่ผิดพลาด (ลูก)	ความผิดพลาด (%)	ความเร็ว (ลูก/นาที)
1	20	20	0	0	32
2	19	21	1	5	31
3	18	22	2	10	29
4	21	19	1	5	30
5	20	20	0	0	31
6	20	20	0	0	29
7	19	21	1	5	28
8	20	20	0	0	29
9	19	21	1	5	30
10	20	20	0	0	31
เฉลี่ย				3	30

บทที่ 5
สรุปผล อภิปรายผลและข้อเสนอแนะ

5.1 สรุปผลการทดลอง

จากผลการออกแบบ สร้าง และทดลองใช้งานเครื่องคัดแยกสีมะนาว โดยใช้หลักการทางกลในการป้อนผลมะนาวเข้าระบบครั้งละ 1 ลูก และทำการตรวจจับสีโดยใช้เซนเซอร์สี แล้วสั่งการให้เซอร์โวมอเตอร์หมุนไปในตำแหน่งที่กำหนด เพื่อเปลี่ยนทิศทางการไหลออกของผลมะนาวตามการตรวจจับของเซนเซอร์สี พบว่าเครื่องสามารถทำงานได้ตามการออกแบบ โดยสั่งการควบคุมรูปแบบการทำงานต่างๆ ได้จากสวิตช์หน้าเครื่อง ตามโปรแกรมที่ได้เขียนไว้ สามารถควบคุมเซอร์โวมอเตอร์ให้หมุนไปในทิศทางที่ต้องการเพื่อคัดแยกสีของผลมะนาวตามการตรวจจับสีของเซนเซอร์ และแสดงผลที่หน้าจอแสดงผลเพื่อให้ผู้ใช้สามารถควบคุมระบบได้ตามวัตถุประสงค์ที่กำหนด ทั้งนี้ได้ออกแบบโปรแกรมการทำงานให้สามารถตั้งค่าสีมะนาวที่จะทำการคัดแยก และตั้งค่าจำนวนมะนาวที่จะให้นับในแต่ละสีได้ จากการทดลองใช้งานโดยให้เครื่องทำการคัดแยกผลมะนาวสีเขียวและสีเหลืองอย่างละ 20 ลูก ทำการทดลอง 10 ครั้งพบว่า เครื่องคัดแยกสีมะนาวมีความผิดพลาดในการคัดแยกสีเฉลี่ย 3% และมีความเร็วในการคัดแยก เฉลี่ย 30 ลูกต่อนาที

5.2 อภิปรายผล

จากการออกแบบและสร้างเครื่องคัดแยกสีมะนาว ทำการทดลองใช้งานและบันทึกผล ทำให้ทราบว่าเครื่องสามารถทำงานได้ตามการออกแบบ โดยผลที่ได้จากการทดลองใช้งานสามารถกล่าวถึงข้อดี-ข้อเสียของเครื่องคัดแยกสีและนับจำนวนมะนาว ดังนี้

5.2.1 ข้อดี

ช่วยลดความความผิดพลาดในการนับมะนาวและคัดแยกสีมะนาว
สะดวกและง่ายต่อการควบคุมของผู้ใช้
ลดต้นทุนในการใช้แรงงานคน

5.2.2 ข้อเสีย

มีความผิดพลาดในการแยกสีในบางครั้ง
ความเร็วในการคัดแยกน้อยเกินไป

5.3 ข้อเสนอแนะ

ปัญหาการคัดแยกสีมะนาวมีการผิดพลาดเนื่องจากการตกของลูกมะนาวที่ไม่เป็นทรงกลมทำให้อาจมีการเอียงได้ในทิศทางใดทิศทางหนึ่งซึ่งมีผลทำให้เซนเซอร์มีการจับสีผิดพลาด มีผลต่อเครื่องให้ทำงานผิดพลาดได้ วิธีการแก้ไขปัญหานี้จะต้องเพิ่มเซนเซอร์เพิ่มเพื่อที่จะให้มีการจับสีได้มากขึ้นและลดความผิดพลาดที่เกิดจากการเอียงของลูกมะนาวได้ ในส่วนความเร็วในการคัดแยก สามารถแก้ไขได้โดยเพิ่มความเร็วในการส่งลูกมะนาวเข้าระบบ และเพิ่มความเร็วในการหมุนของเซอร์โวมอเตอร์ เพื่อเพิ่มความเร็วในการคัดแยก

บรรณานุกรม

1.ณัฏฐพล วงศ์สุนทรชัย และชัยวัฒน์ ลิ้มพรจิตรวิไล. **เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์PIC16F877**. ปรับปรุงครั้งที่ 3. กรุงเทพฯ : อินโนเวตีฟ เอ็กเพอริเมนต์, 2536

2.ชูชัย ธนสารตั้งเจริญ. การสื่อสารข้อมูล. กรุงเทพฯ : ห้างหุ้นส่วนจำกัด ฟิสิกส์เซ็นเตอร์, 2536

3.ประจิน พลังสันติกุล. **เรียนรู้และใช้งาน CCS คอมไพเลอร์เขียนโปรแกรมภาษา C ควบคุมไมโครคอนโทรลเลอร์ PIC**. อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด, 2521

4.มงคล ทองสงคราม. **อิเล็กทรอนิกส์เบื้องต้น**. กรุงเทพฯ : วีเจ ปริ๊นติ้ง, 2546.

ภาคผนวก

ภาคผนวก ก
โปรแกรมควบคุมการทำงาน

```
#include <16F877A.h>                          ; เรียกใช้งานเบอร์ PIC16F877
#device adc=10                                ; กำหนดโมดูล ADC เป็นขนาด 10 บิต
#FUSES NOWDT                                  ; ไม่ใช้งาน Watch dog timer
#FUSES HS                                     ; กำหนดโหมดเป็น High speed clock
#FUSES NOPUT                                  ; ไม่ใช้งาน Power Up Timer
#FUSES PROTECT                                ; ป้องกันการอ่านโค้ดโปรแกรม
#FUSES NODEBUG                                ; ไม่ใช้โหมดการตรวจสอบการทำงานของ
                                              ของโปรแกรม
#FUSES NOBROWNOUT                             ; ไม่ใช้งาน brownout reset
#FUSES NOLVP                                  ; ไม่ใช้งาน low voltage prgming,
                                              B3(PIC16) or B5(PIC18)    used for I/O
#FUSES NOCPD                                  ; ไม่ใช้งาน EE protection
#FUSES NOWRT                                  ; ไม่ป้องกันการเขียนหน่วยความจำ
#use delay(clock=10MHz)                       ; ใช้ clock 10 MHz
#use rs232(baud=9600,xmit=pin_b6,rcv=pin_b7,stream=COM1)      ; เปิดใช้งานโมดูล UART
                                              กรณีต้องการตรวจสอบการทำงานของ
                                              โปรแกรม ปกติจะไม่ได้ใช้งาน
#use fast_io(B)                               ; กำหนดให้พอร์ต B ทำงานในโหมด Fast
/*********************************************************************************/
/* INCLUDE LIBRARY
*/
/*********************************************************************************/
#include <lcd20x4.c>                          ; เรียกใช้งานไลบรารี่ LCD
/*********************************************************************************/
/* STATUS CONTROL
*/
/*********************************************************************************/
#define HI          1                         ; กำหนดคำว่า Hi แทนตัวเลข 1
#define LO          0                         ; กำหนดคำว่า LO แทนตัวเลข 0

/*********************************************************************************/
```

```
/* DEFINE BUTTONS
*/
/*****************************************************************************/
#define BT_ST()      input(PIN_B0)                ; กำหนดให้พอร์ต B0 เป็นปุ่ม เริ่มการ
                                                  ทำงาน
#define BT_OK()      input(PIN_B1)                ; กำหนดให้พอร์ต B1 เป็นปุ่ม ตกลง
#define BT_CL()      input(PIN_B2)                ; กำหนดให้พอร์ต B2 เป็นปุ่ม ยกเลิก
#define BT_UP()      input(PIN_B3)                ; กำหนดให้พอร์ต B3 เป็นปุ่ม เพิ่ม
#define BT_DW()      input(PIN_B4)                ; กำหนดให้พอร์ต B4 เป็นปุ่ม ลด

/*****************************************************************************/
/* SERVO MOTOR
*/
/*****************************************************************************/
#define Servo_Pin(x)  output_bit(pin_c0,x)        ; กำหนดให้พอร์ต C0 เป็นขาควบคุมเซอร์
                                                  โว
/*****************************************************************************/
/* SERVO POSITION
*/
/*****************************************************************************/
#define Servo_Left   312                          ; กำหนดคำว่า Servo_Left แทนค่า 312
                                                  หมายถึงตำแหน่ง เซอร์โวหันไปทางซ้าย
#define Servo_Center 490                          ; กำหนดคำว่า Servo_Center แทนค่า 490
                                                  หมายถึงตำแหน่ง  เซอร์โวอยู่ตรงกลาง
#define Servo_Right  625                          ; กำหนดคำว่า Servo_Right แทนค่า 625
                                                  หมายถึงตำแหน่งเซอร์โวหันไปทางขวา
#define T_20MS       6250
unsigned int16 Servo_Position = Servo_Center;     ; กำหนดตัวแปร สำหรับเก็บตำแหน่งของ
                                                  เซอร์โว

/*****************************************************************************/
/* DEFINE SENSOR
*/
/*****************************************************************************/
#define Sensor()            input(PIN_E0)         ; กำหนดพอร์ต E0 เป็นอินฟาเรท
```

```
#define Select_Sensor1()     set_adc_channel(0)     ; ขา ADC สำหรับอ่านค่าสีช่องที่ 1
#define Select_Sensor2()     set_adc_channel(1)     ; ขา ADC สำหรับอ่านค่าสีช่องที่ 2
#define Select_Sensor3()     set_adc_channel(2)     ; ขา ADC สำหรับอ่านค่าสีช่องที่ 3
#define Select_Sensor4()     set_adc_channel(3)     ; ขา ADC สำหรับอ่านค่าสีช่องที่ 4
#define Read_Sensor()        read_adc()             ; ต้องชื่อ Read_Sensor แทนการคำสั่ง
                                                    read_adc() เป็นคำสั่งอ่านอนาลอกเป็น
                                                    ดิจิตอล
/*****************************************************************************/
/* CONTROL MOTOR                                                             */
/*****************************************************************************/
#define Motor_On()           set_pwm1_duty(255);    ; กำหนดคำว่า Motor_On แทนคำสั่ง
                                                    set_pwm_duty (ให้มอเตอร์ ทำงาน)
#define Motor_Stop()         set_pwm1_duty(0);      ; กำหนดคำว่า Motor_Off แทนคำสั่ง
                                                    set_pwm_duty (ให้มอเตอร์ หยุดทำงาน)
/*****************************************************************************/
/* EEPROM                                                                    */
/*****************************************************************************/
#define Read_Status()        read_eeprom(0)         ; กำหนดคำว่า Read_Status แทนการอ่าน
                                                    หน่วยความจำตำแหน่ง 0
#define Record_Status()      write_eeprom(0,0xAA)   ; กำหนดคำว่า Record_Status แทนการ
                                                    เขียนข้อมูล 0xAA ไปที่หน่วยความจำ
                                                    ตำแหน่ง 0
/*****************************************************************************/
/* LEMON MODE
*/
/*****************************************************************************/
#define MODE_NOCNT_LEMON     0                      ; แทน 0 ด้วยโหมดไม่นับลูกมะนาว
#define MODE_CNT_GLEMON      1                      ; แทน 1 ด้วยโหมดนับลูกมะนาวสีเขียว
#define MODE_CNT_YLEMON      2                      ; แทน 2 ด้วยโหมดนับลูกมะนาวสีเหลือง
#define Read_Mode()          read_eeprom(13)        ; อ่านโหมดการทำงานที่หน่วยความจำ
#define Record_Mode(x)       write_eeprom(13,x)     ; บันทึกโหมดการทำงานที่หน่วยความจำ
int mode;
unsigned long Ref_Color;                            ; ตัวแปรเก็บค่าสีสำหรับอ้างอิงสี
unsigned long Num_GLemon;                           ; ตัวแปรเก็บจำนวนมะนาวสีเขียว ของ
                                                    โหมดนับมะนาวสีเขียว
```

```
unsigned long Num_YLemon;                    ; ตัวแปรเก็บจำนวนมะนาวสีเหลือง ของ
                                               โหมดนับมะนาวสีเหลือง
unsigned long GLemon = 0;                    ; ตัวแปรเก็บจำนวนมะนาวสีเขียว
unsigned long GLemon_Old=-1;                 ; ตัวแปรเก็บจำนวนมะนาวสีเขียวเก่า
unsigned long YLemon = 0;                    ; ตัวแปรเก็บจำนวนมะนาวสีเหลือง
unsigned long YLemon_Old=-1;                 ; ตัวแปรเก็บจำนวนมะนาวสีเหลืองเก่า
#define STOP    0                            ; แทนค่า 0 ด้วยสถานะ หยุดทำงาน
#define START  1                             ; แทนค่า 1 ด้วยสถานะ ทำงาน
int1  Status     = STOP;                     ; ตัวแปรเก็บสถานะ
int1  Status_Old = START;                    ; ตัวแปรเก็บสถานะเก่า
#define T3SEC         150                    ; แทนค่า 150 แทนเวลา 3 วินาที
#define T15SEC        750                    ; แทนค่า 750 แทนเวลา 15 วินาที
int16 count_time;
int16 count_time1;
/*****************************************************************************/
/* Function Prototypes
            */
/*****************************************************************************/
void Show_Menu(void);                        ; ฟังก์ชั่นสำหรับโชว์เมนูบนจอ LCD
void Run_Machine(void);                      ; ฟังก์ชั่นสำหรับแยกมะนาว
void Setting_Machine(void);                  ; ฟังก์ชั่นสำหรับตั้งค่าของมะนาว
void Setup(void);                            ; ฟังก์ชั่นสำหรับเลือกโหมด
void Calibrate(void);                        ; ฟังก์ชั่นสำหรับปรับค่าสี
void Record_REF_Color(unsigned long color);  ; ฟังก์ชั่นสำหรับบันทึกค่าสีอ้างอิง
unsigned long Read_REF_Color(void);          ; ฟังก์ชั่นสำหรับอ่านค่าสีอ้างอิงจาก
                                               หน่วยความจำ
void Record_Num_GLemon(unsigned long num);   ; ฟังก์ชั่นสำหรับบันทึกจำนวนของมะนาว
                                               สีเขียว
unsigned long Read_Num_GLemon(void);         ; ฟังก์ชั่นสำหรับอ่านจำนวนมะนาวสีเขียว
void Record_Num_YLemon(unsigned long num);   ; ฟังก์ชั่นสำหรับบันทึกจำนวนของมะนาว
                                               สีเหลือง
unsigned long Read_Num_YLemon(void);         ; ฟังก์ชั่นสำหรับอ่านจำนวนมะนาวสี
                                               เหลือง
unsigned long read_color(void);              ; ฟังก์ชั่นสำหรับอ่านสี

/*****************************************************************************/
```

```
/* Function : main()
            */
/*****************************************************************************/
void main (void)                                    ; ฟังก์ชั่นหลัก
{
PORT_B_PULLUPS(true);                               ; กำหนดให้พอร์ต B ต่อ Resister
set_tris_b(0x1F);                                   ; กำหนดให้พอร์ต B เป็นอินพุต

// Enable Timer0
setup_timer_0(RTCC_INTERNAL|RTCC_DIV_256);          ; เปิดใช้งานไทเมอร์ 0 จับเวลาขณะเครื่อง
                                                    ทำงาน
set_timer0(61);
// Enable timer1
setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);             ; เปิดใช้งานไทเมอร์ 1 สำหรับสร้างพัลส์
                                                    ควบคุมเซอร์โซ
set_timer1(0);
// Enable CCP module
setup_ccp1(CCP_PWM);                                ; เปิดใช้งานโมดูล PWM เพื่อใช้ควบคุม
                                                    ความเร็วมอเตอร์
setup_timer_2(T2_DIV_BY_16,255,1);                  ; เปิดใช้งานไทเมอร์ 2 สำหรับใช้งาน
                                                    ร่วมกับโมดูล PWM
set_timer2(0);
Motor_Stop();                                       ; สั่งให้มอเตอร์หยุดทำงาน
// Enable ADC module
setup_adc(ADC_CLOCK_INTERNAL|ADC_CLOCK_DIV_8);  ; เปิดใช้งานโมดูลสัญญาณ
                                                    อนาลอกเป็นดิจิตอล
setup_adc_ports(AN0_AN1_AN2_AN3_AN4);               ; เปิดใช้งานโมดูล ADC ช่อง 0,1,2,3,4
// Enable interrupts
enable_interrupts(INT_TIMER0);                      ; เปิดใช้งานอินเทอรัปต์ไทเมอร์ 0
enable_interrupts(INT_TIMER1);                      ; เปิดใช้งานอินเทอรัปต์ไทเมอร์ 1
enable_interrupts(GLOBAL);                          ; เปิดใช้งานอินเทอรัปต์ทั้งหมด
// Initail LCD module
lcd_init();                                         ; กำหนดการทำงาน LCD
if(Read_Status() != 0xAA){                          ; อ่านสถานะของหน่วยความจำ
Record_Status();                                    ;ทำการบันทึกค่าของหน่วยความจำเพื่อ
```

```
                                                    ไม่ให้เข้าในเงื่อนไขนี้อีก
Record_REF_Color(600);                              ; กำหนดค่าเริ่มต้นของสีอ้างอิงเป็น 600
                                                    และบันทึกลงหน่วยความจำ
Record_Num_GLemon(0);                               ; กำหนดจำนวนของมะนาวสีเขียวเป็น 0
                                                    ในหน่วยความจำ
Record_Num_YLemon(0);                               ; กำหนดจำนวนของมะนาวสีเหลืองเป็น 0
                                                    ในหน่วยความจำ
Record_Mode(MODE_NOCNT_LEMON);                      ; กำหนดโหมดเริ่มต้นให้เป็นโหมดไม่นับ
                                                    ลูกมะนาว

}
Ref_Color = Read_REF_Color();                       ; อ่านค่าสีอ้างอิงจากหน่วยความจำ
Num_GLemon = Read_Num_GLemon();                     ; อ่านจำนวนมะนาวสีเขียวจาก
                                                    หน่วยความจำ
Num_YLemon = Read_Num_YLemon();                     ; อ่านจำนวนมะนาวสีเหลืองจาก
                                                    หน่วยความจำ
mode = Read_Mode();                                 ; อ่านโหมดจากหน่วยความจำ
Servo_Position = Servo_Right;                       ; สั่งให้เซอร์โวหันไปทางขาวเพื่อทิ้งลูก
                                                    มะนาวที่อาจจะค้างในเครื่อง
delay_ms(2000);                                     ; หน่วงเวลา 2 วินาที
Servo_Position = Servo_Center;                      ; สั่งให้เซอร์โวไปยังตำแหน่งตรงกลาง
Show_Menu();                                        ; โชว์เมนู
                                                    ; โปรแกรมจะทำงานในลูปนี้ตลอด
while(true)
{
Run_Machine();                                      ; เรียกฟังก์ชั่นแยกลูกมะนาว
Setting_Machine();                                  ; เรียกฟังก์ชั่นตั้งค่าเครื่องแยกมะนาว
}
}
/*****************************************************************************/
/* Function : Timer0_Isr()
            */
/* Interrupt every : 20 ms
            */
/*****************************************************************************/
#INT_TIMER0                                         ; ฟังก์ชั่นอินเทอร์รัปต์ไทเมอร์ 0 เกิดอิน
```

```
                                                    เทอร์รัปต์ทุกๆ 10 ms
void Timer0_Isr(void)
{
set_timer0(61);
count_time++;                                       ; บวกเวลาไปอีก 1
count_time1++;                                      ; บวกเวลาไปอีก 1
}
/*****************************************************************************/
/* Function : Timer1_Isr()
            */
/*****************************************************************************/
#INT_TIMER1                                         ; ฟังก์ชั่นอินเทอร์รัปต์ไทเมอร์ 1
void Timer1_Isr(void)
{
static int1 Servo_Stat = 0;
switch(Servo_Stat)                                  ; เช็คสถานะ ของพัลส์
{
case 0:  set_timer1(65536-Servo_Position);          ; สำหรับสร้างพัลส์ช่วง High
Servo_Pin(HI);
Servo_Stat = 1;
break;
case 1:  set_timer1(65536-(T_20MS-Servo_Position)); ; สำหรับสร้างพัลส์ช่วง Low
Servo_Pin(LO);
Servo_Stat = 0;
break;
}
}
/*****************************************************************************/
/* Function : Show_Menu()
*/
/*****************************************************************************/
void Show_Menu(void)                                ;โชว์เมนูบนจอ LCD
{
lcd_gotoxy(1,1);printf(lcd_putc," Lemon         ");
lcd_gotoxy(1,2);printf(lcd_putc,"Green :            ");
```

```
lcd_gotoxy(1,3);printf(lcd_putc,"Yellow:              ");
lcd_gotoxy(1,4);printf(lcd_putc,"Status:              ");
}
/*****************************************************************************/
/* Function : Run_Machine()
*/
/*****************************************************************************/
void Run_Machine(void)
{
static int1   BT_Stat;                          ; สร้างตัวแปรสำหรับเก็บสถานะของ
                                                สวิตซ์
static int step = 0;                            ; สร้างตัวแปรสเต็ปการทำงานของการแยก
                                                มะนาว
unsigned long color_avg;                        ; ตัวแปรเก็บค่าสีที่ได้จากการเฉลี่ยแล้ว
if(GLemon != GLemon_Old){                       ; ตรวจสอบว่ามะนาวสีเขียวเปลี่ยนแปลง
                                                หรือไม่
GLemon_Old = GLemon;                            ; กำหนดจำนวนของมะนาวเก่าเป็นจำนวน
                                                มะนาวใหม่
lcd_gotoxy(8,2);printf(lcd_putc," %lu       ",GLemon); ; แสดงจำนวนมะนาวสีเขียว
}
if(YLemon != YLemon_Old){                       ; ตรวจสอบว่ามะนาวสีเหลืองเปลี่ยนแปลง
                                                หรือไม่
YLemon_Old = YLemon;                            ; กำหนดจำนวนของมะนาวเก่าเป็นจำนวน
                                                มะนาวใหม่
lcd_gotoxy(8,3);printf(lcd_putc," %lu       ",YLemon); ; แสดงจำนวนมะนาวสีเหลือง
}
if(Status != Status_Old){                       ; อ่านสถานะควบคุมว่าเปลี่ยนแปลงหรือไม่
Status_Old = Status;                            ; เปลี่ยนสถานะเก่าเป็นสถานะใหม่
lcd_gotoxy(8,4);
switch(Status)                                  ; ตรวจสอบสถานะควบคุม
{
                                                ; สถานะเริ่มทำงาน
case START: printf(lcd_putc,"   START   "); break;   ; แสดงสถานะเริ่มทำงานบนจอ LCD
                                                สถานะหยุดทำงาน
case  STOP: printf(lcd_putc,"   STOP    "); break;   ; แสดงสถานะหยุดทำงานบนจอ LCD
```

```
}
}
if((!BT_ST())&&(BT_Stat==0)){          ; ตรวจสอบว่ามีการกดสวิตซ์ START
                                       หรือไม่
delay_ms(10);                          ; หน่วงเวลาเพื่อตรวจสอบอีกครั้ง
if(!BT_ST()){                          ; ตรวจสอบว่ามีการกดสวิตซ์ START
                                       หรือไม่
BT_Stat = 1;                           ; กำหนดสถานะของ BT_Stat เป็น 1
Status = ~ Status;                     ; กลับสถานะ
switch(Status){                        ; ตรวจสอบสถานะ
case  STOP: Step = 0;                  ; สถานะหยุดทำงาน
Motor_Stop();                          ; สั่งให้มอเตอร์หยุดทำงาน
Servo_Position = Servo_Center;         ; สั่งให้เซอร์โวอยู่ตำแหน่งกลาง
break;
case START: Step = 1;                  ; สถานะทำงาน
break;
}
}
}
if(BT_ST()){                           ; ถ้าไม่มีการกดสวิตซ์ START
delay_ms(10);                          ; หน่วงเวลา
if(BT_ST()){                           ; ถ้าไม่มีการกดสวิตซ์ START
BT_Stat = 0;                           ; เคลียร์ BT_Stat เป็น 0
}
}
if(step == 1){                         ; ถ้าสเต็ปการทำงานเป็น 1
Motor_On();                            ; สั่งให้มอเตอร์ทำงาน
Step = 2;
Count_time1 = 0;
}
if(step == 2){                         ; ถ้าสเต็ปการทำงานเป็น 2
if(sensor()==0){
step = 1;                              ; กำหนดให้สเต็ปเป็น 1
Motor_Stop();                          ; สั่งให้มอเตอร์หยุดทำงาน
color_avg = read_color();              ; อ่านค่าของสีมะนาว
```

```
if(color_avg < Ref_Color){                    ; ถ้ามะนาวเป็นสีเหลือง
Servo_Position = Servo_Left;                  ; สั่งให้เซอร์โวไปทางซ้าย
YLemon++;                                     ; เพิ่มจำนวนมะนาวสีเหลืองอีกหนึ่ง
if(mode == MODE_CNT_YLEMON){                  ; ถ้าเป็นโหมดนับจำนวนลูกมะนาว
if(YLemon == Num_YLemon){                     ; ถ้าจำนวนมะนาวสีเหลืองเท่ากับจำนวน
                                              มะนาวที่ตั้งไว้
YLemon = 0;                                   ; เคลียร์จำนวนมะนาวสีเหลือง
Status = STOP;                                ; สถานะหยุดทำงาน
Motor_Stop();                                 ; สั่งให้มอเตอร์หยุดทำงาน
step = 0;                                     ; สั่งให้สเต็ปเป็น 0
}
}
}
else{                  // Green lemom
Servo_Position = Servo_Right;                 ; สั่งให้มอเตอร์ไปทางขวา
GLemon++;                                     ; เพิ่มมะนาวสีเขียว
if(mode == MODE_CNT_GLEMON){                  ; ถ้าเป็นโหมดนับจำนวนลูกมะนาว
if(GLemon == Num_GLemon){                     ; ถ้าจำนวนมะนาวสีเขียวเท่ากับจำนวน
                                              มะนาวที่ตั้งไว้
GLemon = 0;                                   ; เคลียร์จำนวนมะนาวสีเขียว
Status = STOP;                                ; สถานะหยุดทำงาน
Motor_Stop();                                 ; สั่งให้มอเตอร์หยุดทำงาน
step = 0;                                     ; สั่งให้สเต็ปเป็น 0
}
}
}
Servo_Position = Servo_Right;                 ; สั่งเซอร์โวไปทางขวา
delay_ms(500); หน่วงเวลา
Servo_Position = Servo_Center;                ; สั่งเซอร์โวไปตำแหน่งตรงกลาง
delay_ms(500);                                ; หน่วงเวลา
}
if(count_time1 >= T15SEC){                    ; ถ้าเวลาครบ 3 วินาที
Status = STOP;                                ; ให้สถานะเป็นหยุดการทำงาน
Motor_Stop();                                 ; สั่งให้มอเตอร์หยุดทำงาน
```

```
step = 0;                                   ; เปลี่ยนสเต็ปการทำงานเป็น 0
}
}
}

/*****************************************************************************/
/* Function : Setting_Machine()
*/
/*****************************************************************************/
void Setting_Machine(void)
{
if((!BT_OK())&&(!BT_CL())){                 ; ถ้ามีการกดปุ่ม ตกลง และปุ่ม ยกเลิก
if(count_time >= T3SEC){                    ; ถ้าเลาครบ 3 วินาที

Motor_Stop();                               ; สั่งไหม้มอเตอร์หยุดการทำงาน
                                            ; แสดงโหมดตั้งค่าต่างๆ
lcd_gotoxy(1,1);printf(lcd_putc,"    Select Mode    ");
lcd_gotoxy(1,2);printf(lcd_putc,"Press + : Setup    ");
lcd_gotoxy(1,3);printf(lcd_putc,"Press - : Calibrate ");
lcd_gotoxy(1,4);printf(lcd_putc,"Press C : Exit     ");


                                            ; หากมีการกดปุ่มให้รอจนกว่าไม่กดปุ่ม
while((!BT_OK())||(!BT_CL())||(!BT_UP())||(!BT_DW())); delay_us(10);
                                            ; หากมีการกดปุ่ม ยกเลิก ปุ่มเพิ่ม ปุ่มลด
while(( BT_CL())&&( BT_UP())&&( BT_DW())); delay_us(10);
if(!BT_UP()){                               ; ถ้ากดปุ่ม เพิ่ม
Setup();                                    ; เรียกโหมด Setup()
}
if(!BT_DW()){                               ; ถ้ากดปุ่มลด
Calibrate();                                ; เรียกโหมดตั้งค่าสี
}
Status = STOP;                              ; กำหนดสถานะเป็นหยุดทำงาน
Status_Old =~ Status;                       ; กำหนดสถานะเก่าเป็นสถานะตรงกันข้าม
                                            กับสถานะใหม่
GLemon_Old = -1;                            ; กำหนดจำนวนมะนาวสีเขียวเก่ามีจำนวน
                                            -1
```

```
YLemon_Old = -1;                                ; กำหนดจำนวนมะนาวสีเหลืองเก่ามี
                                                จำนวน -1
Show_Menu();                                    ; เรียกโชว์เมนู
}
}
else
{
count_time = 0;                                 ; เคลียร์เวลา
}
}
/*************************************************************************/
/* Function : Setup()
            */
/*************************************************************************/
void Setup(void)                                ; ฟังก์ชั่น Setup()
{
int1 Stat = 1;                                  ; ตัวแปรเก็บสถานะว่าจะออกจากโหมด
                                                setup หรือไม่
int index = 0;
int mode_buf;
unsigned long lemon_buf;
unsigned long lemon_old;
unsigned long Time_BT_UP = 0;
unsigned long Time_BT_DW = 0;

lemon_old = -1;
mode_buf  = mode;                               ; เอาโหมดปัจจุบัน มาเก็บให้โหมดสำลอง
                                                ; แสดงโหมด Setup บนจอ LCD
lcd_gotoxy(1,1);printf(lcd_putc,"    Select Lemon    ");
lcd_gotoxy(1,2);printf(lcd_putc," Mode  :            ");
lcd_gotoxy(1,3);printf(lcd_putc," Number:   -        ");
lcd_gotoxy(1,4);printf(lcd_putc,"                    ");
while(Stat){
switch(index){
case 0:  lcd_gotoxy(1,2);printf(lcd_putc,">");
```

```
                                                    ; แสดงโหมดนับลูกมะนาว
switch(mode_buf){
case MODE_NOCNT_LEMON:lcd_gotoxy(9,2);printf(lcd_putc,"  OFF  ");break;
case  MODE_CNT_GLEMON:lcd_gotoxy(9,2);printf(lcd_putc," Green ");break;
case  MODE_CNT_YLEMON:lcd_gotoxy(9,2);printf(lcd_putc," Yellow");break;
}
while((!BT_OK())||(!BT_CL())||(!BT_UP())||(!BT_DW())); delay_us(10);
while(( BT_OK())&&( BT_CL())&&( BT_UP())&&( BT_DW())); delay_us(10);
if(!BT_OK()){                                       ; ถ้ามีการกดปุ่ม ตกลง
switch(mode_buf){
                                                    ; โหมดไม่นับลูกมะนาว
case MODE_NOCNT_LEMON:  Stat  = 0;
break;
                                                    ; โหมดนับลูกมะนาวสีเขียว
case  MODE_CNT_GLEMON:  lemon_buf = Num_GLemon; index = 1;
lcd_gotoxy(1,2);printf(lcd_putc," ");
lcd_gotoxy(1,3);printf(lcd_putc,">");
while(!BT_OK()); delay_us(10);
break;
                                                    ; โหมดนับลูกมะนาวสีเหลือง
case  MODE_CNT_YLEMON:  lemon_buf = Num_YLemon; index = 1;
lcd_gotoxy(1,2);printf(lcd_putc," ");
lcd_gotoxy(1,3);printf(lcd_putc,">");
while(!BT_OK()); delay_us(10);
break;
}
}
if(!BT_CL()){                                       ; ถ้ากดปุ่มยกเลิก
Stat = 0;                                           ; ออกจากโหมด
}
if(!BT_UP()){                                       ; ถ้ามีการกดปุ่มตกลง
if(mode_buf<=1){                                    ; ถ้าโหมด น้อยกว่าหรือเท่ากับ 1
mode_buf++;                                         ; ให้mode_buf++
}
```

```
else{
mode_buf=0;                                          ; ให้mode_buf=0
}
}
if(!BT_DW()){                                        ; ถ้ากดปุ่มยกเลิก
if(mode_buf>0){                                      ; ถ้าโหมด มากกว่า 0
mode_buf--;                                          ; ให้ mode_buf--
}
else{
mode_buf=2;                                          ; ให้ mode_buf=2
}
}
break;
case 1:  if(lemon_buf != lemon_old){                 ; มะนาวเก่าไม่เท่ากับมะนาวใหม่
lemon_old = lemon_buf;                               ; โหลดจำนวนมะนาวไปยังมะนาวเก่า
lcd_gotoxy(10,3);printf(lcd_putc,"%lu      ",lemon_buf);
}
if(!BT_OK()){                                        ; ถ้ากดปุ่มตกลง
mode = mode_buf;                                     ; เอาค่าโหมดใหม่ไปยังตัวแปรโหมด
Record_Mode(mode);                                   ; บันทึกโหมด
                                                     ;เช็คโหมด
switch(mode){
                                                     ; โหมดนับมะนาวสีเขียว
case MODE_CNT_GLEMON:   Num_GLemon = lemon_buf;   ; โหลดจำนวนมะนาวสีเขียวใหม่
                                                     ไปยังตัวแปร Num_GLemon
Record_Num_GLemon(Num_GLemon);                       ; บันทึกจำนวนมะนาวลงหน่วยความจำ
break;
                                                     ; โหมดนับมะนาวสีเหลือง
case MODE_CNT_YLEMON:   Num_YLemon = lemon_buf;   ; โหลดจำนวนมะนาวสีเหลือง
                                                     ใหม่ไปยังตัวแปร Num_YLemon
Record_Num_YLemon(Num_YLemon);                       ; บันทึกจำนวนมะนาวลงหน่วยความจำ
break;
}
Stat = 0;                                            ; ออกจากโหมดการทำงาน
}
```

```
if(!BT_CL()){                          ; ถ้ากดปุ่ม ยกเลิก
Stat = 0;                              ; ออกจากโหมดการทำงาน
}

if(!BT_UP()){                          ; ถ้ากดปุ่ม ตกลง
if(Time_BT_UP == 30000){               ; ถ้าเวลาที่กด มากกว่า 30000 ลูป
if(!BT_UP()){                          ; ถ้ากดปุ่ม ตกลง
if(lemon_buf < 1000){                  ; ถ้า lemon_buf น้อยกว่า 1000
lemon_buf++;                           ; ให้ lemon_buf++
delay_ms(50);
}
else{
lemon_buf=0;                           ; ให้ lemon_buf = 0
}
}
}
else{
Time_BT_UP++;
}
}
if(BT_UP()){
if(Time_BT_UP>1000){                   ; ถ้าเวลาที่กด มากกว่า 1000 ลูป
if(lemon_buf < 1000){                  ; ถ้า lemon_buf น้อยกว่า 1000
lemon_buf++;                           ; ให้ lemon_buf++
}
else{
lemon_buf=0;                           ; ให้ lemon_buf=0
}
}
Time_BT_UP = 0;
}
if(!BT_DW()){                          ; ถ้ามีการกดปุ่มยกเลิก
if(Time_BT_DW == 30000){               ; ถ้าเวลาที่กด มากกว่า 30000 ลูป
if(!BT_DW()){                          ; ถ้ามีการกดปุ่มยกเลิก
if(lemon_buf > 0){                     ; ถ้า lemon_buf มากกว่า 0
```

```
lemon_buf--;                                    ; ให้ lemon_buf--;
delay_ms(50);
}
else{
lemon_buf=1000;                                 ; ให้ lemon_buf=1000
}
}
}
else{
Time_BT_DW++;
}
}

if(BT_DW()){                                    ; ถ้าไม่มีการกดปุ่มยกเลิก
if(Time_BT_DW>1000){                            ; ถ้าเวลาที่กด มากกว่า 1000 ลูป
if(lemon_buf>0){                                ; ถ้า lemon_buf มากกว่า 0
lemon_buf--;                                    ; ให้ lemon_buf--;
}
else{
lemon_buf=1000;                                 ; ให้ lemon_buf=1000
}
}
Time_BT_DW = 0;
}
break;
}
}
}
/*****************************************************************************/
/* Function : Calibrate
            */
/*****************************************************************************/
void Calibrate(void)
{
int sel_color = 0;                              ; ตัวแปรสำหรับเก็บชนิดของสีมะนาวที่
```

```
                                                    ต้องการตั้งค่าสี
int1 Stat = 1;                                      ; ตัวแปร Stat สำหรับเช็คว่ามีการออกจาก
                                                    โหมดตั้งค่าสีหรือไม่
unsigned long value;                                ; ตัวแปรเก็บค่าของสี
                                                    ; แสดงข้อความว่าให้ใส่ลูกมะนาวลงในถัง
lcd_gotoxy(1,1);printf(lcd_putc,"                   ");
lcd_gotoxy(1,2);printf(lcd_putc," Please put a lemon ");
lcd_gotoxy(1,3);printf(lcd_putc,"    into tray!     ");
lcd_gotoxy(1,4);printf(lcd_putc,"                   ");
Motor_On();                                         ; สั่งให้มอเตอร์หมุนเพื่อให้มะนาวหล่นลง
                                                    มาที่ช่องอ่านสีมะนาว
// Read sensor
while(Sensor()==1);
Motor_Stop();                                       ; สั่งให้มอเตอร์หยุดทำงาน
                                                    ; แสดงข้อความโหมดตั้งค่าสีมะนาว
lcd_gotoxy(1,1);printf(lcd_putc," Please select color");
lcd_gotoxy(1,2);printf(lcd_putc,"      of lemon.    ");
lcd_gotoxy(1,3);printf(lcd_putc," color :     -     ");
lcd_gotoxy(1,4);printf(lcd_putc,"Up:Green   Dw:Yellow");
while(Stat){                                        ; เช็คว่ามีการออกจากโหมดนี้หรือไม่
if(!BT_UP()){                                       ; ถ้ากดปุ่มตกลง
delay_us(100);                                      ; หน่วงเวลา
if(!BT_UP()){                                       ; ถ้ากดปุ่มตกลง
lcd_gotoxy(10,3);printf(lcd_putc,"Green ");         ; แสดงข้อความบอกว่าเลือกมะนาวสีเขียว
                                                    ในการปรับค่า
sel_color = 1;                                      ; ให้ sel_color เป็นการเลือกมะนาวสีเขียว
while(!BT_UP());delay_us(100);                      ; ถ้ามีการกดปุ่มเพิ่ม
}
}
if(!BT_DW()){                                       ; ถ้ามีการกดปุ่มยกเลิก
delay_us(100);                                      ; หน่วงเวลา
if(!BT_DW()){                                       ; ถ้ามีการกดปุ่ม ยกเลิก
lcd_gotoxy(10,3);printf(lcd_putc,"Yellow");         ; แสดงข้อความบอกว่าเลือกมะนาวสี
                                                    เหลืองในการปรับค่า
```

```
sel_color = 2;                                  ; ให้ sel_colorเป็นการเลือกมะนาวสี
เหลือง
while(!BT_DW());delay_us(100);                  ; รอจนกว่าจะปล่อยปุ่มลด
}
}
if(!BT_OK()){                                   ; ถ้ามีการกดปุ่มตกลง
delay_us(100);                                  ; หน่วงเวลา
if(!BT_OK()){                                   ; ถ้ามีการกดปุ่ม ตกลง
if(sel_color>0){
// Read color
value = read_color();                           ; อ่านค่าของสีมะนาว
switch(sel_color){                              ; ตรวจสอบสีของมะนาวที่ใช้ในการปรับ
                                                ค่า
case 1: Ref_Color = value-20; break;            ; ถ้ามะนาวสีเขียว ให้ค่าอ้างอิงสีเท่ากับ
                                                Ref_Color = value-20
case 2: Ref_Color = value+20; break;            ; ถ้ามะนาวสีเหลือง ให้ค่าอ้างอิงสีเท่ากับ
                                                 Ref_Color = value-20
}
Record_REF_Color(Ref_Color);                    ; บันทึกสีลงหน่วยความจำ
stat = 0;                                       ; กำหนดให้ตัวแปร Stat = 0
}
while(!BT_OK());delay_us(100);                  ; รอจนกว่าจะปล่อยปุ่ม ตกลง
}
}
if(!BT_CL()){                                   ; ถ้ามีการกดปุ่มยกเลิก
delay_us(100);                                  ; หน่วงเวลา
if(!BT_CL()){                                   ; ถ้ามีการกดปุ่มยกเลิก
stat = 0;                                       ; กำหนดให้ตัวแปร Stat = 0
while(!BT_OK());delay_us(100);                  ; รอจนกว่าจะปล่อยปุ่ม ตกลง
}
}
}
Servo_Position = Servo_Right;                   ; สั่งเซอร์โวไปทางขวา
delay_ms(500);                                  ; หน่วงเวลา
Servo_Position = Servo_Center;                  ; สั่งเซอร์โวไปตรงกลาง
```

```
}
/*****************************************************************************/
/* Function : Record_REF_Color(unsigned long color)
*/
/*****************************************************************************/
void Record_REF_Color(unsigned long color)
{
write_eeprom(1,color/1000);                    ; เก็บค่าสีหลัก 1000
write_eeprom(2,(color%1000)/100);              ; เก็บค่าสีหลัก 100
write_eeprom(3,((color%1000)%100)/10);         ; เก็บค่าสีหลัก 10
write_eeprom(4,((color%1000)%100)%10);         ; เก็บค่าสีหลัก 1
}
/*****************************************************************************/
/* Function : unsigned long Read_REF_Color()
*/
/*****************************************************************************/
unsigned long Read_REF_Color(void)
{
unsigned long color;

color  = (unsigned long)read_eeprom(1)*1000;   ; อ่านค่าสีหลัก 1000
color += (unsigned long)read_eeprom(2)*100;    ; อ่านค่าสีหลัก 100
color += (unsigned long)read_eeprom(3)*10;     ; อ่านค่าสีหลัก 10
color += (unsigned long)read_eeprom(4);        ; อ่านค่าสีหลัก 1
return(color);
}

/*****************************************************************************/
/* Function : Record_Num_GLemon(unsigned long num)
            */
/*****************************************************************************/
void Record_Num_GLemon(unsigned long num)
{
write_eeprom(5,num/1000);                      ; เก็บมะนาวสีเขียวหลัก 1000
write_eeprom(6,(num%1000)/100);                ; เก็บมะนาวสีเขียวหลัก 100
```

```
write_eeprom(7,((num%1000)%100)/10);              ; เก็บมะนาวสีเขียวหลัก 10
write_eeprom(8,((num%1000)%100)%10);              ; เก็บมะนาวสีเขียวหลัก 1
}
/*****************************************************************************/
/* Function : unsigned long Read_Num_GLemon()
*/
/*****************************************************************************/
unsigned long Read_Num_GLemon(void)
{
unsigned long num;
num  = (unsigned long)read_eeprom(5)*1000;        ; อ่านมะนาวสีขียวหลัก 1000
num += (unsigned long)read_eeprom(6)*100;         ; อ่านมะนาวสีเขียวหลัก 100
num += (unsigned long)read_eeprom(7)*10;          ; อ่านมะนาวสีเขียวหลัก 10
num += (unsigned long)read_eeprom(8);             ; อ่านมะนาวสีเขียวหลัก 1
return(num);
}


/*****************************************************************************/
/* Function : Record_Num_YLemon(unsigned long num)
            */
/*****************************************************************************/
void Record_Num_YLemon(unsigned long num)
{
write_eeprom(9,num/1000);                         ; เก็บมะนาวสีเหลืองหลัก 1000
write_eeprom(10,(num%1000)/100);                  ; เก็บมะนาวสีเหลืองหลัก 100
write_eeprom(11,((num%1000)%100)/10);             ; เก็บมะนาวสีเหลืองหลัก 10
write_eeprom(12,((num%1000)%100)%10);             ; เก็บมะนาวสีเหลืองหลัก 1
}
/*****************************************************************************/
/* Function : unsigned long Read_Num_YLemon()
*/
/*****************************************************************************/
unsigned long Read_Num_YLemon(void)
{
```

```
unsigned long num;
num  = (unsigned long)read_eeprom(9)*1000;        ; อ่านมะนาวสีเหลืองหลัก 1000
num += (unsigned long)read_eeprom(10)*100;        ; อ่านมะนาวสีเหลืองหลัก 100
num += (unsigned long)read_eeprom(11)*10;         ; อ่านมะนาวสีเหลืองหลัก 10
num += (unsigned long)read_eeprom(12);            ; อ่านมะนาวสีเหลืองหลัก 1
return(num);
}
/*****************************************************************************/
/* Function : unsigned long read_color()
*/
/*****************************************************************************/
unsigned long read_color(void)
{
unsigned long avg_color;
  // Select sensor 1
  Select_Sensor1();                               ; เลือกเซนเซอร์ตัวที่ 1
  delay_us(15);                                   ; หน่วงเวลา
  avg_color = Read_Sensor();                      ; อ่านค่าสี
  // Select sensor 2
  Select_Sensor2();                               ; เลือกเซนเซอร์ตัวที่ 2
  delay_us(15);                                   ; หน่วงเวลา
  avg_color += Read_Sensor();                     ; อ่านค่าสี
  // Select sensor 3
  Select_Sensor3();                               ; เลือกเซนเซอร์ตัวที่ 3
  delay_us(15);                                   ; หน่วงเวลา
  avg_color += Read_Sensor();                     ; อ่านค่าสี
  // Select sensor 4
  Select_Sensor4();                               ; เลือกเซนเซอร์ตัวที่ 4
  delay_us(15);                                   ; หน่วงเวลา
  avg_color += Read_Sensor();                     ; อ่านค่าสี
  // Average color
  avg_color = avg_color/4;                        ; หาค่าเฉลี่ยสี
return(avg_color);                                ; ส่งค่าเฉลี่ยสี
}
```

ภาคผนวก ข
คู่มือการใช้งาน

คู่มือการใช้งาน

เครื่องนี้เป็นอุปกรณ์ที่ช่วยเพิ่มประสิทธิภาพในการคัดแยกสีและนับจำนวนมะนาวโดยจะทำการคัดสีโดยเปรียบเทียบค่าของสีมะนาวและนำไปแสดงผลในจอแอลซีดี การใช้งานสามารถนำไปคัดแยกมะนาวที่กำลังจะสุกหรือสุกแล้วเพื่อคัดออกจากมะนาวสีเขียวเพื่อที่จะเก็บมะนาวได้นานขึ้น เป็นต้น

ขั้นตอนการใช้งาน

1) เสียบปลั๊กและเปิดสวิตซ์
2) ตั้งโปรแกรมโดยกดปุ่ม Ok และ Cancel พร้อมกัน 3 วินาที เพื่อเข้าสู่การตั้งโปรแกรม
3) เลือกฟังก์ชั่นการตั้งค่าสีและจำนวนมะนาว โดยกดปุ่ม Up และ Down
4) เมื่อได้ฟังก์ชั่นที่ต้องการแล้วกดปุ่ม Ok
5) กดปุ่ม Start เพื่อเริ่มทำงาน และปุ่ม Stop เพื่อหยุดการทำงาน

ประวัติผู้วิจัย

1.ชื่อ สกุล นายธนากร สุนทรวัฒน์

2.ตำแหน่งปัจจุบัน อาจารย์

3.หน่วยงานและสถานที่อยู่ที่ติดต่อได้

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีราชมงคลรัตนโกสินทร์ ศาลายา
96 หมู่ 3 ต.ศาลายา อ.พุทธมณฑล จ.นครปฐม (73170)
โทร:02-8894595 ต่อ xxxx
E-Mail : tanakorn.s@rmutr.ac.th , tsuntornwat@hotmail.com

4.ประวัติการศึกษา

วุฒิการศึกษา	สถานที่ศึกษา	สาขาวิชา	ปี พ.ศ.
ปริญญาตรี	ศูนย์กลางสถาบันเทคโนโลยีราชมงคล คลองหก (มทร.ธัญบุรี)	วิศวกรรมไฟฟ้า-อิเล็กทรอนิกส์	2539

5.สาขาวิชาการที่มีความชำนาญพิเศษ

-

4.ประสบการณ์ที่เกี่ยวข้องกับการบริหารงานวิจัย

1.หัวหน้าโครงการวิจัย : เครื่องมือวัดระยะความละเอียดสูง
งานวิจัยที่ทำเสร็จแล้ว : เครื่องมือวัดระยะความละเอียดสูง ปี 2550
การเผยแพร่ : นำไปทดลองใช้งานจริงในการวัดการทรุดตัวของดิน
แหล่งทุน : สถาบันวิจัยและพัฒนา มหาวิทยาลัยเทคโนโลยีราชมงคลรัตนโกสินทร์

2.หัวหน้าโครงการวิจัย : อุปกรณ์ประหยัดการใช้พลังงานไฟฟ้าสำหรับเครื่องปรับอากาศขนาดเล็ก
งานวิจัยที่ทำเสร็จแล้ว : อุปกรณ์ประหยัดการใช้พลังงานไฟฟ้าสำหรับเครื่องปรับอากาศขนาดเล็ก ปี 2554
การเผยแพร่ : นำไปทดลองใช้งานจริง โดยติดตั้งใช้งานในบ้านพักอาศัยของผู้วิจัย และเพื่อนอาจารย์ในมหาวิทยาลัย จำนวน 3 หลัง
แหล่งทุน : สถาบันวิจัยและพัฒนา มหาวิทยาลัยเทคโนโลยีราชมงคลรัตนโกสินทร์

3.ผู้ร่วมวิจัย : เครื่องตัดโลหะด้วยพลาสมาควบคุมด้วยแขนกลอัตโนมัติ
งานวิจัยที่ทำเสร็จแล้ว : เครื่องตัดโลหะด้วยพลาสมาควบคุมด้วยแขนกลอัตโนมัติ ปี 2553
การเผยแพร่ : นำไปทดลองใช้งานจริง โดยใช้ตัดโลหะเพื่อทำชิ้นงานตกแต่งภายในมหาวิทยาลัย
แหล่งทุน : สถาบันวิจัยและพัฒนา มหาวิทยาลัยเทคโนโลยีราชมงคลรัตนโกสินทร์

4.หัวหน้าโครงการวิจัย : แผนที่พูดได้สำหรับผู้พิการทางสายตา
งานวิจัยที่ทำเสร็จแล้ว : แผนที่พูดได้สำหรับผู้พิการทางสายตา ปี 2555
การเผยแพร่ : กำลังอยู่ในระหว่างดำเนินการวิจัย
แหล่งทุน : สถาบันวิจัยและพัฒนา มหาวิทยาลัยเทคโนโลยีราชมงคลรัตนโกสินทร์

5.หัวหน้าโครงการวิจัย : การพัฒนาเครื่องคัดแยกสีมะนาว
งานวิจัยที่ทำเสร็จแล้ว : เครื่องคัดแยกสีมะนาว ปี 2556
การเผยแพร่ : แสดงผลงานในงานราชมงคลวิชาการ ครั้งที่ 5 ปี 2556
แหล่งทุน : สถาบันวิจัยและพัฒนา มหาวิทยาลัยเทคโนโลยีราชมงคลรัตนโกสินทร์