เครื่องชั่งน้ำหนักแสดงค่าดัชนีมวลกาย

THE BODY MASS INDEX WEIGHING DISPLAY MACHINE

นายปฏิภาณ สุขดิษฐ์ รหัส 53362846

นายสวนัท อุดมรักษ์ รหัส 53363126

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2556

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อโครงงาน	เครื่องชั่งน้ำหนักแสดงค่าดัชนีมวลกาย	
ผู้ดำเนินโครงงาน	นายปฏิภาณ สุขดิษฐ์	รหัส 53362846
	นายสวนัท อุดมรักษ์	รหัส 53363126
ที่ปรึกษาโครงงาน	ดร.มุฑิตา สงฆ์จันทร์	
สาขาวิชา	วิศวกรรมไฟฟ้า	
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์	
ปีการศึกษา	2556	

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

.....ที่ปรึกษาโครงงาน
(ดร.มุฑิตา สงฆ์จันทร์)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ศุภวรรณ พลพิทักษ์ชัย)

.....กรรมการ
(ดร.ปิยดนัย ภาชนะพรรณ์)

ชื่อหัวข้อโครงงาน	เครื่องชั่งน้ำหนักแสดงค่าดัชนีมวลกาย
ผู้ดำเนินโครงงาน	นายปฏิภาณ สุขดิษฐ์ รหัส 53362846
	นายสวนัท อุดมรักษ์ รหัส 53363126
ที่ปรึกษาโครงงาน	ดร.มุฑิตา สงฆ์จันทร์
สาขาวิชา	วิศวกรรมไฟฟ้า
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2556

บทคัดย่อ

โครงงานนี้เป็นการสร้างเครื่องชั่งน้ำหนักแสดงค่าดัชนีมวลกายและสามารถเก็บข้อมูลได้ โดยใช้ไมโครคอนโทรลเลอร์รับข้อมูลน้ำหนักจากโหลดเซลล์ และข้อมูลส่วนสูงจากการป้อนค่าผ่านแป้นกดแล้วนำมาประมวลผลคำนวณเป็นค่าดัชนีมวลกาย และสามารถเก็บค่าน้ำหนัก และค่าดัชนีมวลกายหลังการชั่งน้ำหนักเพื่อใช้ในการเปรียบเทียบความเปลี่ยนแปลงของร่างกายได้ ซึ่งจากการทดลองเครื่องชั่งน้ำหนักสามารถชั่งน้ำหนักได้โดยมีค่าความผิดพลาดไม่เกิน 1 เปอร์เซ็นต์ และสามารถบันทึกข้อมูล น้ำหนัก ค่าดัชนีมวลกาย ลำดับค่า และวันที่ทำการบันทึกได้อย่างถูกต้อง

Project title	The Body Mass Index Weighing Display Machine		
Name	Mr. Patipan	Sookdit	ID. 53362846
	Mr. Sawanat	Udomrak	ID. 53363126
Project advistor	Miss Mutita	Songjun, Ph.D.	
Major	Electrical Engineering		
Department	Electrical and Computer Engineering		
Academic year	2013		

Abstract

This project is build the body mass index weighing display machine by using microcontroller receive weight's data from load cell and receive height's data from keypad then processing to the body mass index value. A machine can save weight's data and the body mass index data for compare about a user body. From the result, a machine have error from weighing process is less than 1 percent and it's can save a weight's data, body mass index data , order of data and date of save correctly.

กิตติกรรมประกาศ

โครงงานนี้สำเร็จลุล่วงไปได้ด้วยความกรุณาเป็นอย่างยิ่งจาก ดร.มุฑิตา สงฆ์จันทร์ ซึ่งเป็นที่ปรึกษาโครงงานและให้ความกรุณาในการเอาใจใส่ในรายละเอียดในการสร้างชิ้นงานและตรวจทานปริญญานิพนธ์รวมถึงการให้คำแนะนำและให้คำปรึกษาในการแก้ไขปัญหาตลอดการดำเนินงาน คณะผู้ดำเนินงานโครงการขอกราบขอบพระคุณเป็นอย่างสูงและขอระลึกถึงความกรุณาของท่านไว้ตลอดไป

ขอขอบพระคุณคณาจารย์ทุกท่านที่ประสิทธ์ประสาทวิชาความรู้ให้กับผู้ดำเนินงานโครงงาน ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.ศุภวรรณ พลพิทักษ์ชัย และดร.ปิยดนัย ภาชนะพรรณ์ ที่ให้เกียรติเป็นกรรมการโครงงาน นอกจากนี้ยังขอกราบขอบพระคุณภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ที่ให้ความสะดวกในการยืมอุปกรณ์เครื่องมือวัดและเครื่องมือในการทำงานต่างๆมาใช้งาน ขอขอบคุณเพื่อนทุกๆคนและบุคคลท่านอื่นๆที่มีส่วนร่วมในการให้คำแนะนำและการช่วยเหลือจนทำให้โครงงานนี้สำเร็จลุล่วงไปได้ด้วยดี และขอขอบคุณทุกๆท่านที่ไม่ได้กล่าวถึงซึ่งเป็นกำลังใจทำให้ได้รับความสำเร็จเช่นทุกวันนี้ ณ ที่นี้ด้วย

นายปฏิภาณ สุขดิษฐ์
นายสวนัท อุดมรักษ์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป(ต่อ)

รูปที่ หน้า

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

เครื่องชั่งน้ำหนักดิจิตอลสำหรับชั่งวัตถุมีการใช้งานที่แพร่หลายในด้านการค้าเช่น ตามห้างสรรพสินค้าทั่วไป เป็นต้น เครื่องชั่งน้ำหนักดิจิตอลตามท้องตลาดมีหลากหลายยี่ห้อและรุ่นให้เลือกใช้ตามความต้องการ ที่แตกต่างกันในแต่ละรุ่นคือ ค่าน้ำหนักที่รับได้สูงสุดและความละเอียดในการชั่ง ซึ่งจาก 2 ข้อนี้ทำให้เครื่องชั่งน้ำหนักดิจิตอลมีราคาที่แตกต่างกันไป ตั้งแต่หลักหลายพันจนถึงหลักหมื่น ซึ่งจากราคาที่สูงนี้ ทางผู้จัดทำจึงต้องการสร้างเครื่องชั่งน้ำหนักดิจิตอลสำหรับชั่งวัตถุที่สามารถรับน้ำหนักสูงสุดได้มากและมีความละเอียดที่สูงในราคาที่ถูกลง และปัจจุบันผู้คนรักและใส่ใจในสุขภาพของตนเองมากขึ้น จึงได้มีการพัฒนาให้สามารถชั่งน้ำหนักคนและประมวลผลค่าดัชนีมวลกายได้ และเพื่อความสะดวกในการเปรียบเทียบค่าน้ำหนักของผู้ใช้งานจึงทำให้สามารถเก็บค่าข้อมูลหลังชั่งน้ำหนักได้

1.2 วัตถุประสงค์ของโครงงาน

สร้างเครื่องชั่งน้ำหนักและประมวลผลค่าดัชนีมวลกาย และเก็บบันทึกค่าทั้งหมดได้

1.3 ขอบเขตของโครงงาน

1. เครื่องชั่งน้ำหนักดิจิตอลสามารถทำงานได้ในสถานที่ที่มีแหล่งจ่ายไฟเท่านั้น
2. ในการชั่งน้ำหนักสามารถรับน้ำหนักวัตถุหรือบุคคลได้สูงสุด 150 กิโลกรัม
3. การเก็บบันทึกข้อมูลการใช้งานสามารถเก็บข้อมูลได้ 50 ข้อมูล
4. เครื่องชั่งน้ำหนักดิจิตอลสามารถแสดงค่าน้ำหนัก และค่าดัชนีมวลกายได้

1.4 ขั้นตอนและแผนการดำเนินงานของโครงงาน

ขั้นตอนและแผนการดำเนินงานสามารถแสดงได้ดังตารางที่ 1.1

ตารางที่ 1.1 แผนการดำเนินงานของโครงงาน

ลำดับ	การดำเนินการ	ปี 2557							
		ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค.	มิ.ย.	ก.ค.	ส.ค.
1	ศึกษาวิธีการชั่งน้ำหนักโดยใช้กระบวนการต่างๆ	←	—	→					
2	เขียนโปรแกรมในการชั่งน้ำหนักและบันทึกค่า			←	—	→			
3	ตรวจสอบการทำงานของโปรแกรมการชั่งน้ำหนักและบันทึกค่า				←	—	→		
4	ติดตั้งอุปกรณ์ของเครื่องชั่งน้ำหนักและตรวจสอบการทำงานอีกครั้ง					←	—	→	
5	สรุปผลการทดลองและจัดทำรูปเล่ม							←	— →

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ลดค่าใช้จ่ายในการจัดซื้อจากตามท้องตลาดซึ่งมีราคาที่สูง

2. สร้างความสะดวกในการจดจำข้อมูลเมื่อผู้ใช้งานต้องการเปรียบเทียบข้อมูลการชั่งน้ำหนักและค่าดัชนีมวลกายในแต่ละครั้ง

3. เป็นต้นแบบในการพัฒนาเพื่อเพิ่มประสิทธิภาพในการใช้งาน

1.6 งบประมาณของโครงงาน

1. ค่าอุปกรณ์ไฟฟ้าและอิเล็กทรอนิกส์	1,000	บาท
2. ค่าเอกสาร	1,000	บาท
รวมเป็นเงินทั้งสิ้น (สองพันบาทถ้วน)	2,000	บาท

(หมายเหตุ: ถัวเฉลี่ยทุกรายการ)

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 ไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ (Microcontroller) คือ ชิพประมวลผลชนิดหนึ่งซึ่งทำหน้าที่ประมวลผลตามโปรแกรมหรือชุดคำสั่ง โครงสร้างภายในจะเป็นวงจรรวมขนาดใหญ่ประกอบไปด้วยหน่วยคำนวณที่คณิตศาสตร์และลอจิก บัสข้อมูล บัสควบคุม บัสที่อยู่ พอร์ตขนาน พอร์ตอนุกรม รีจิสเตอร์ หน่วยความจำ วงจรนับ วงจรจับเวลาและวงจรอื่นๆ รวมอยู่ภายในชิพหรือไอซีไมโครคอนโทรลเลอร์ ซึ่งถูกออกแบบเพื่อใช้งานควบคุมสามารถติดต่อกับอุปกรณ์อินพุตและเอาต์พุตได้สะดวกใช้งานได้ง่ายสามารถทำงานได้โดยใช้ชิพเดียว มีคำสั่งสนับสนุนในการเขียนโปรแกรมควบคุมและสามารถเข้าถึงข้อมูลแบบบิตได้ ตระกูลไมโครคอนโทรลเลอร์ที่ใช้ในโครงงานนี้เป็นตระกูล PIC

2.1.1ไมโครคอนโทรลเลอร์ตระกูล PIC

ไมโครคอนโทรลเลอร์ตระกูล PIC เป็นไมโครคอนโทรลเลอร์ที่มีการออกแบบที่รวมทุกอย่างไว้ในชิพตัวเดียวโดยไม่ต้องต่ออุปกรณ์ใดๆ เพิ่มเติม ได้แก่ หน่วยประมวลผลกลาง (CPU) ,หน่วยความจำโปรแกรม(Program memory),หน่วยความจำข้อมูล(Data memory) ,Timer , หน่วยเก็บความจำแบบหน่วยความจำข้อมูลอีอีพรอม(EEPROM) ,ระบบควบคุมป้อนกลับ (Feedback Control System),จุดควบคุมวิกฤต (Critical Control Point) ,การแปลงสัญญาณอนาลอกเป็นดิจิตอล (Analog to Digital Convertor) เป็นต้น ผลที่ตามมาก็คือแผ่นวงจรจะมีขนาดเล็ก และอุปกรณ์ที่ใช้จะไม่มาก บางงานอาจจะใช้แค่ PIC เพียงตัวเดียวโดยไม่ต้องใช้ชิพอื่นมาเพิ่มเติมเ นี่คือคุณสมบัติพิเศษของ PIC แต่เนื่องจากการที่รวมทุกอย่างไว้ในชิพเดียว ทำให้หน่วยความจำโปรแกรม และหน่วยความจำข้อมูลไม่สามารถขยายโดยใช้กับหน่วยความจำภายนอกได้ในทางทฤษฎี PIC จึงเหมาะสำหรับงานเล็กๆ ไม่ใช่งานใหญ่ๆ ที่ต้องใช้การคำนวณหน่วยความจำเยอะๆ ความเร็วในการทำงานของ PIC ปัจจุบันสัญญาณนาฬิกาสูงสุดของ PIC มีค่าเท่ากับ 20 เมกกะเฮิรตซ์ ดังนั้นหนึ่งคำสั่งของ PIC ใช้เวลาเพียง 0.25 ไมโครวินาที

2.1.1.1 โครงสร้างภายในของไมโครคอนโทรลเลอร์ตระกูล PIC

โครงสร้างภายในของไมโครคอนโทรลเลอร์ตระกูล PIC นั้นประกอบด้วยหน่วยประมวลผลกลางหน่วยประมวลผลกลางจะประกอบไปด้วยวงจรต่างๆหลากหลาย ที่จำเป็นสำหรับประมวลผล และการคำนวณ เช่น วงจรถอดรหัสคำสั่ง ,วงจรควบคุมสัญญาณนาฬิกา,วงจรควบคุมการทำงาน,วงจรตั้งเวลา และรวมทั้งหน่วยคำนวณทางลอจิกและคณิตศาสตร์เป็นต้น หน่วยความจำ(Memory Unit) การเขียนโปรแกรมด้วยภาษาซีให้กับไมโครคอนโทรลเลอร์ต้อง

คำนึงถึงขนาดของหน่วยความจำของไมโครคอนโทรลเลอร์ด้วย เพราะหากเขียนให้โปรแกรมที่มีขนาดความจุมากกว่าขนาดของหน่วยความจำโปรแกรมไมโครคอนโทรลเลอร์ ก็ไม่สามารถบรรจุโปรแกรมลงได้ครบสมบูรณ์ ทั้งนี้ต้องระมัดระวังในการใช้หน่วยความจำแบบอื่นในไมโครคอนโทรลเลอร์ด้วยในไมโครคอนโทรลเลอร์ สามารถแบ่งหน่วยความจำได้ 3 แบบ คือ หน่วยความจำแบบแฟลช (Flash Program Memory) โดยปกติไมโครคอนโทรลเลอร์หลายๆยี่ห้อถูกออกแบบให้มีคุณสมบัติในการเขียนโปรแกรมและการลบโปรแกรมได้มากกว่า 100,000 ครั้งและการทำงานมีความเร็วสูงเหมาะใช้ในการพัฒนางานที่มีขนาดใหญ่ หน่วยความจำนี้มีขนาด 1 กิโลไบต์ ถึง 32 กิโลไบต์หน่วยความจำข้อมูลหน่วยความจำนี้เป็นหน่วยความจำชั่วคราว ใช้เก็บข้อมูลขณะประมวลผล เมื่อหยุดจ่ายไฟเลี้ยง ข้อมูลก็หายไป มีความจุตั้งแต่ 64 ถึง 1536 ไบต์ หน่วยความจำข้อมูลอีอีพรอม(EEPROM Data Memory) เป็นหน่วยความจำที่สามารถเขียนและลบโปรแกรมได้ด้วยกระแสไฟฟ้าในหน่วยความจำรอม มีความจุขนาดถึง256 ไบต์ พอร์ตอินพุตและเอาต์พุต(I/O Port) ไมโครคอนโทรลเลอร์ทุกตัวต้องมีขาต่อใช้งาน และสามารถเป็นได้ทั้งขาสัญญาณออกและขาสัญญาณเข้า จึงจะสามารถติดต่อกับอุปกรณ์ภายนอก หรือนำไปใช้งานอื่นได้ เช่น จอแสดงผล, ปุ่มกด, เซ็นเซอร์ เป็นต้น ในการใช้งานจะแบ่งพอร์ตออกเป็นพอร์ตละกี่บิตก็แล้วแต่ ส่วนใหญ่ตามมาตรฐานพอร์ตหนึ่งจะมี 8 บิต(ขาต่อใช้งาน 8 ขา) ส่วนจะมีกี่พอร์ตก็ตามแต่เบอร์ใช้งานที่ถูกออกแบบมา เช่น ไมโครคอนโทรลเลอร์เบอร์ PIC16F877 ก็จะมีพอร์ตใช้งานอยู่ 5 พอร์ต คือพอร์ต A มี 6 ขา, พอร์ต B มี 8 ขา,พอร์ต C มี 8 ขา,พอร์ต D มี 8 ขา,และพอร์ต E มี 3 ขา

2.1.1.2 การเขียนโปรแกรมไมโครคอนโทรลเลอร์

ภาษาที่ใช้สำหรับการเขียนโปรแกรมบนไมโครคอนโทรลเลอร์แบ่งได้เช่นเดียวกับภาษาในการเขียนโปรแกรมบนคอมพิวเตอร์คือ ภาษาเครื่อง ภาษาแอสเซมบลี และภาษาซี

ภาษาเครื่องเป็นภาษาระดับต่ำสุดประกอบด้วยรหัสฐาน 2 คือ 0 กับ 1 เท่านั้น ซึ่งเป็นภาษาที่มนุษย์จะทำความเข้าใจได้ยาก เพราะต้องอาศัยการจดจำรหัสคำสั่งต่างๆ รวมถึงต้องเข้าใจโครงสร้างภายในของไมโครคอนโทรลเลอร์ด้วย จึงได้มีการคิดค้นสิ่งที่เรียกว่าคอมไพล์เลอร์ (Compiler) ขึ้นมาเพื่อทำให้มนุษย์สามารถเขียนโปรแกรมด้วยภาษาระดับสูงที่มนุษย์เข้าใจได้ โดยคอมไพล์เลอร์ทำหน้าที่เปลี่ยนภาษาระดับสูงนั้นให้กลายเป็นภาษาเครื่องนั่นเอง

ภาษาแอสเซมบลี เป็นภาษาที่ใช้รหัสคำสั่งที่เป็นตัวอักษรภาษาอังกฤษมาแทนคำสั่งเลขฐาน 2 ในแบบของภาษาเครื่องทำให้ภาษาแอสเซมบลีกลายเป็นภาษาที่มนุษย์ทำความเข้าใจได้ง่ายขึ้น นอกจากนั้นแล้วยังเป็นภาษาที่ยังทำให้โปรแกรมมีการทำงานได้อย่างรวดเร็วเพราะมีการสั่งงานไปที่ฮาร์ดแวร์โดยตรง ด้วยเหตุนี้ทำให้ผู้พัฒนาโปรแกรมจำเป็นต้องเข้าใจ โครงสร้าง

ภายในไมโครคอนโทรลเลอร์อย่างละเอียดด้วย ซึ่งกลายเป็นข้อด้อยของภาษาแอสเซมบลีไป คอมไพล์เลอร์ที่ทำหน้าที่แปลงภาษาแอสเซมบลีให้เป็นภาษาเครื่องเรียกว่า แอสเซมเบลอร์

ภาษาซีเป็นภาษาระดับสูงที่มีความใกล้เคียงกับภาษามนุษย์ ทำให้ทำความเข้าใจได้ง่าย นอกจากนั้นแล้วการเขียนโปรแกรมภาษาซีจึงไม่ต้องจำเป็นต้องเข้าใจโครงสร้างภายในของไมโครคอนโทรลเลอร์อย่างละเอียด เพียงแต่เข้าใจการเขียนโปรแกรมแบบโครงสร้างก็เพียงพอแล้ว ภาษาซีสามารถใช้ในการเข้าถึงโครงสร้างภายในของไมโครคอนโทรลเลอร์ได้โดยตรง ให้ได้โปรแกรมที่เขียนทำงานได้รวดเร็ว ดังนั้นภาษาซี จึงเป็นที่นิยมแพร่หลายในการเขียนโปรแกรมไมโครคอนโทรลเลอร์ คอมไพล์เลอร์ที่ใช้ในการแปลงภาษาซีเป็นภาษาเครื่องมีอยู่มากมาย เช่น คอมไพล์เลอร์ PIC C Complier เป็นต้น

2.1.1.3 รูปแบบการทำงานของขาไมโครคอนโทรลเลอร์หมายเลข **PIC16F877**

ไมโครคอนโทรลเลอร์ที่ใช้งานในปัจจุบันมีอยู่ด้วยกันหลากหลายรุ่น และแต่ละรุ่นมีโครงสร้างภายในที่ต่างกัน ให้ผู้ใช้งานได้เลือกใช้ได้อย่างเหมาะสม โดยไมโครคอนโทรลเลอร์หมายเลข PIC16F877 มีคุณสมบัติดังนี้

1. มีคำสั่งภายในแอสเซมบลี 35 คำสั่ง
2. ใช้ความถี่ออสซิลเลเตอร์ได้สูงสุด 20 เมกะเฮิรตซ์
3. หน่วยความจำโปรแกรมหน่วยความจำแบบแฟลชขนาด 8 K word (14 bit -word)
4. มีหน่วยความจำแบบ RAM ขนาด 368 ไบต์
5. มีหน่วยเก็บความจำแบบหน่วยความจำข้อมูลอีอีพรอม ขนาด 256 ไบต์
6. มีการตอบสนองอินเตอร์รัพท์ทั้งหมด 14 แหล่ง
7. สามารถเลือกระดับการป้องกันข้อมูลได้(Code Protection)
8. มีโหมดประหยัดพลังงาน(Save mode)
9. สามารถเลือกแหล่งสัญญาณนาฬิกาได้หลายโหมด XT RC และออสซิลเลเตอร์พลังงานต่ำ
10. มีฟังก์ชั่นการรักษาเสถียรภาพการทำงานได้แก่ POR , PWRT, OST, BOR และ WDT
11. มีโปรแกรมตัวชิพแบบ ICSP (In-Circuit Serial Programming)
12. สามารถทำงานที่ไฟเลี้ยงวงจรตั้งแต่ 2.0 โวลต์ ถึง 5.5 โวลต์
13. ขาพอร์ตอินพุต/เอาต์พุตแต่ละขาสามารถรับและปล่อยกระแสได้สูงสุด 25 มิลลิแอมแปร์

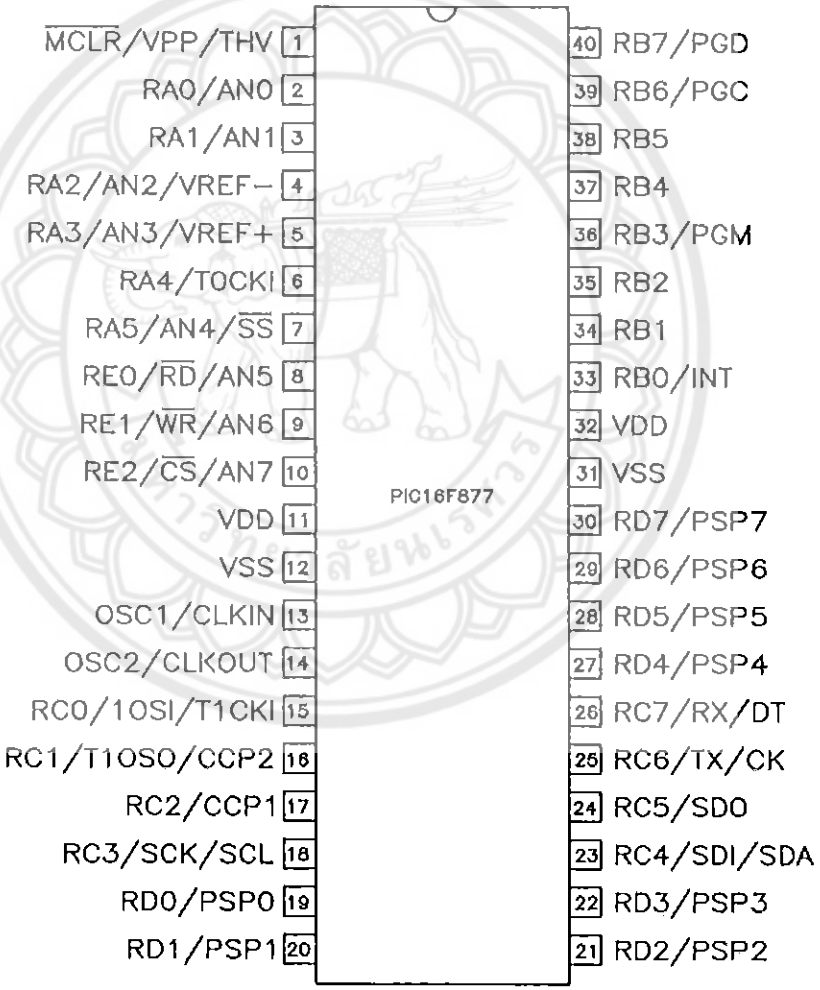
14. มีโมดูล Timer / Counter ใช้งานทั้งหมด 3 ตัว Timer 0 , Timer 1, และ Timer 2

15. มีโมดูล CCP (Compare / Capture / PWM) จำนวน 2 ชุด

16. มีโมดูลการแปลงสัญญาณอนาลอกเป็นดิจิตอลความละเอียดขนาด 8 บิต และ 10 บิตจำนวน 8 ช่องภายในตัวชิพ

17. มีโมดูลสื่อสารอนุกรมแบบ USART (Universal Synchronous Asynchronous Receiver / Transmitter)

สำหรับไมโครคอนโทรลเลอร์ในตระกูล PIC หมายเลข 16F877A แสดงการจัดขาของไมโครคอนโทรลเลอร์ในแบบ 40 ขา ดังรูปที่ 2.1 และมีรายละเอียดการทำงานดังตารางที่ 2.1



รูปที่ 2.1 รูปแบบการทำงานของไมโครคอนโทรลเลอร์หมายเลข PIC16F877

ตารางที่ 2.1 รายละเอียดการทำงานแต่ละขาของไมโครคอนโทรลเลอร์หมายเลข PIC16F877

ขา	หน้าที่การทำงาน
OSC1/CLKIN	อินพุตของออสซิลเลเตอร์คริสตัล/ขาเข้าของแหล่งกำเนิดสัญญาณ
	นาฬิกาภายนอก
OSC2/LKOUT	เอาต์พุตของออสซิลเลเตอร์คริสตัล
MCLR/V_{PP}	อินพุตของสวิตช์รีเซ็ต
RA0/AN0 RA1/AN1 RA2/AN2/V_{REF-} RA3/AN3/V_{REF+} RA4/T0CKI RA5/SS/AN4	เป็นพอร์ตที่ทำหน้าที่หลักเป็นอินพุตและเอาต์พุตให้กับอุปกรณ์ภายนอก (RA0-5) และสามารถเป็นอนาลอกอินพุต(Analog input) (AN0-4) เป็นแรงดันอ้างอิงอนาลอกเชิงลบและเชิงบวก(Negative and Positive analog reference voltage)(V_{REF-},V_{REF+}) เป็นอินพุตของสัญญาณนาฬิกาที่ไปยัง อุปกรณ์นับเวลา 0 (T0CKI) เป็นตัวเลือกรองสำหรับการซิงโครนัสพอร์ตอนุกรม Slave select for the synchronous serial port (SS)
RB0/INT RB1 RB2 RB3/GM RB4 RB5 RB6/PGC RB7/PGD	เป็นพอร์ตที่ทำหน้าที่หลักเป็นอินพุตและเอาต์พุตให้กับอุปกรณ์ภายนอก (RB0-7) ซึ่งพอร์ตนี้สามารถทำการโปรแกรมซอฟแวร์สำหรับอินพุตภายในได้ และสามารถเป็นขาอินเทอรัพท์ภายนอกได้ (RB0) เป็นอินพุตการเขียนโปรแกรมแรงดันต่ำได้ (RB3)
RC0/T1OSO/T0CKI RC1/T1OSI/CCP2 RC2/CCP1 RC3/SCK/SCL RC4/SDI/SDA RC5/SDO RC6/TX/CK	เป็นพอร์ตที่ทำหน้าที่หลักเป็นอินพุตและเอาต์พุตให้กับอุปกรณ์ภายนอก (RC0-6) และสามารถเป็นเอาต์พุตตัวนับเวลาออสซิลเลเตอร์ที่ 1 หรืออินพุตสัญญาณนาฬิกาของตัวนับเวลาที่ 1 ได้ (RC1) เป็นอินพุตตัวนับเวลาออสซิลเลเตอร์ที่ 1 ได้ (RC2) เป็นอินพุตหรือเอาต์พุตของการซิงโครนัสอนุกรมสัญญาณนาฬิกา (RC3) เป็น SPI Data In (SPI mode) หรือ data I/O (I^2C mode) (RC4) เป็น SPI Data Out (SPI mode) (RC5)
V_{SS}	เชื่อมต่อกับกราวด์
V_{DD}	เชื่อมต่อกับไฟเลี้ยง 5 โวลต์

2.2 โหลดเซลล์

โหลดเซลล์ (Load Cell) คืออุปกรณ์ที่ใช้วัดค่าแรงหรือน้ำหนักที่กระทำกับโหลดเซลล์โดยที่โหลดเซลล์จะเปลี่ยนแรงหรือน้ำหนักให้เป็นสัญญาณทางไฟฟ้า ซึ่งโหลดเซลล์สามารถจำแนกได้ 5 กลุ่มได้แก่ โหลดเซลล์แบบสเตรนเกจ (Strain gauge) โหลดเซลล์แบบไฮดรอลิก (Hydraulic) โหลดเซลล์แบบไพโซรีซิสทีฟ (Piezoresistive) โหลดเซลล์แบบนิวแมติก(Pneumatic) และโหลดเซลล์แบบแมกเนโตสเตร็กทีฟ (Magnetostrictive) ซึ่งโครงงานนี้เป็นโหลดเซลล์แบบสเตรนเกจ

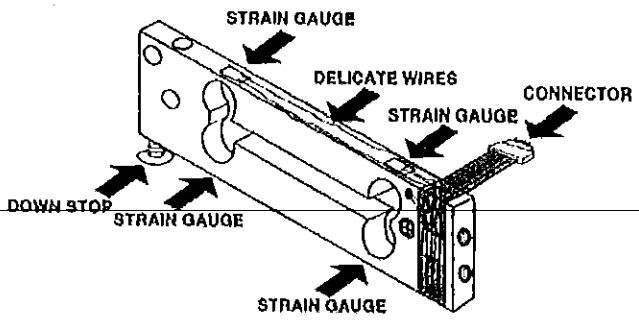
2.2.1 โหลดเซลล์แบบสเตรนเกจ

การใช้งานโหลดเซลล์แบบสเตรนเกจในยุคแรกเริ่มนั้นยุ่งยากพอสมควร ซึ่งจะใช้ทฤษฎีการเปลี่ยนแปลงความต้านทานทางไฟฟ้าของโลหะจะเกิดขึ้นเมื่อโลหะนั้นถูกกระทำภายใต้โหลดเป็นทฤษฎีที่นิยมใช้ในการวัดค่าความเครียด (Strain) ซึ่งได้รับการพัฒนาโดยใช้เส้นโลหะขนาดเล็กนำมาขดรวมกันเป็นกลุ่มหรือเรียกว่าเมทาลิกสเตรนเกจ

ต่อมาเมื่ออุปกรณ์อิเล็กทรอนิกส์ที่มีประสิทธิภาพมากขึ้นทำให้โหลดเซลล์แบบสเตรนเกจ กลายเป็นอุปกรณ์ที่ใช้งานได้ง่ายขึ้นและมีราคาที่ถูกลง จนกระทั่งนำเอาวัสดุสารกึ่งตัวนำมาใช้แทนเส้นลวดโลหะในการผลิต เพื่อเพิ่มประสิทธิภาพและให้เหมาะกับงานหลายประเภทมากยิ่งขึ้นมีชื่อเรียกว่า เซมิคอนดักเตอร์สเตรนเกจ (Semiconductor Strain Gauge) ไม่เพียงจะสามารถนำมาใช้วัดค่าความเครียดสำหรับหาค่าความเค้นแต่สามารถนำมาใช้กับทรานสดิวเซอร์สำหรับวัดค่าต่างๆได้หลายชนิด เช่น โหลดเซลล์สำหรับวัดแรงทรานสดิวเซอร์สำหรับวัดแรงบิดนี่คือหลักการของโครงงานนี้

2.2.1.1 หลักการของโหลดเซลล์แบบสเตรนเกจ

หลักการของโหลดเซลล์แบบสเตรนเกจ คือ เมื่อมีน้ำหนักมากระทำความเครียดจะเปลี่ยนเป็นความต้านทานไฟฟ้าในสัดส่วนโดยตรงกับแรงที่มากระทำ โดยใช้เกจวัดความเครียด 4 ตัวในการวัดเพื่อให้ได้ความไวสูงสุดและมีการชดเชยผลของอุณหภูมิขณะทำการวัดด้วยเกจทั้ง 4 จะเชื่อมต่อเข้าด้วยกันเพื่อช่วยในการปรับตั้งค่าชดเชยวงจร โดยทั่วไปเกจ 2 ตัวจะอยู่ในสภาพถูกดึงและอีก 2 ตัวอยู่ในสภาพถูกกด ตัวต้านทานทั้ง 4 จะเชื่อมต่อเข้าด้วยกันเพื่อใช้แปลงแรงที่กระทำกับตัวของมันไม่ว่าจะเป็นแรงกดหรือแรงดึงส่งสัญญาณออกมาเป็นแรงดันไฟฟ้าโดยที่แรงดันไฟฟ้าที่ได้จะมีหน่วยเป็น มิลลิโวลต์/โวลต์สำหรับโหลดเซลล์ที่ใช้ในโครงงานนี้จะมีอัตราการสัญญาณเอาต์พุตต่อน้ำหนักที่กระทำ 2มิลลิโวลต์/โวลต์ที่การใช้งานแบบเต็มกำลัง (Full Load) สัญญาณที่ได้จะเท่ากับ 10มิลลิโวลต์ โครงสร้างของโหลดเซลล์แบบทรานสดิวเซอร์ มีลักษณะดังรูปที่ 2.2 และคุณสมบัติของโหลดเซลล์แบบทรานสดิวเซอร์มีคุณสมบัติดังตารางที่ 2.2



รูปที่ 2.2 โครงสร้างของโหลดเซลล์แบบทรานสดิวเซอร์

ตารางที่ 2.2 คุณสมบัติของโหลดเซลล์

รายละเอียด	คุณสมบัติ
ความจุน้ำหนัก	150 กิโลกรัม
ทิศทางของแรง	แรงบีบและแรงดึงเครียด (Compression and Tension)
ใช้แรงดันไฟ	10 VDC ถึง 12 VDC
จ่ายไฟออก	1.8 มิลลิโวลต์/โวลต์
ประเภทโหลดเซลล์	แบบจุดเดี่ยว (Single Point)
ตัวเรือน	อลูมิเนียมอัลลอย(Aluminum Alloy)
อุณหภูมิขณะใช้งาน	-30 ถึง 70 องศาเซลเซียส

2.3 วงจรแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล

สัญญาณที่ใช้ในอุปกรณ์อิเลคทรอนิคส์ มี 2 ชนิด คือ สัญญาณอนาลอก และสัญญาณดิจิตอล สัญญาณอนาลอก จะใช้ในอุปกรณ์ทั่วๆไปและใช้ในการควบคุมแบบเก่า

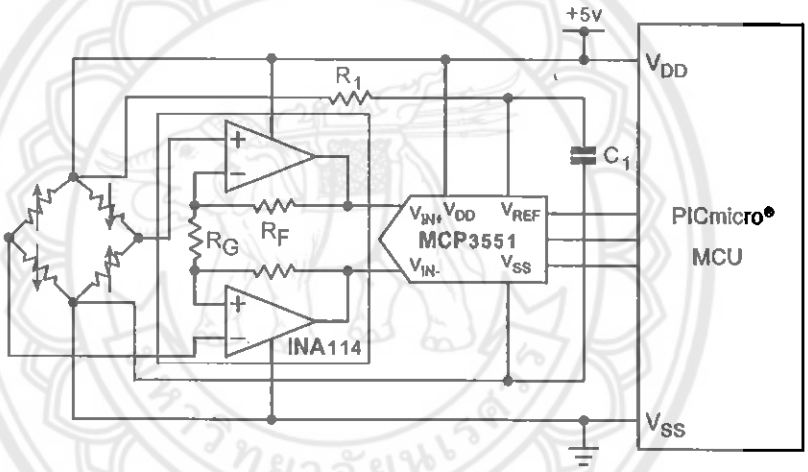
ในปัจจุบันมีไมโครโปรเซสเซอร์ และไมโครคอนโทรลเลอร์ เข้ามาช่วยในการควบคุมอุปกรณ์ต่างๆ มากมาย ซึ่งทำให้การควบคุมนั้นทำได้ง่าย และรวดเร็วยิ่งขึ้น แต่ในการควบคุมนั้นเราจำเป็นต้องใช้ สัญญาณดิจิตอลในการติดต่อกับไมโครโปรเซสเซอร์ หรือไมโครคอนโทรลเลอร์ แต่ในความเป็นจริงนั้น เราใช้สัญญาณอนาลอกในการควบคุม ดังนั้นเราจึงจำเป็นต้องมีการเปลี่ยนสัญญาณอนาลอกเป็นสัญญาณดิจิตอล แล้วจึงนำสัญญาณนั้นเข้ามาสู่ไมโครโปรเซสเซอร์ หรือไมโครคอนโทรลเลอร์เพื่อใช้ควบคุมระบบต่อไป

แม้ว่าสัญญาณอนาลอกนั้นมีความแน่นอน และแม่นยำสูง แต่สัญญาณอนาลอกนั้นควบคุมได้ยาก เนื่องจากในสภาพแวดล้อม มีสัญญาณรบกวนอยู่มากและการที่จะทำให้ การควบคุมแบบ

อนาลอก มีความสามารถควบคุม เท่ากับการควบคุมแบบดิจิตอลนั้นทำได้ยากเนื่องจากวงจรควบคุมแบบอนาลอกจะต้องมีความซับซ้อนสูง

อย่างไรก็ตาม สัญญาณดิจิตอลก็ไม่สามารถทดแทนความละเอียดของสัญญาณอนาลอกได้อย่างสมบูรณ์ แต่ทำให้การควบคุมนั้นทำให้ง่าย และสะดวกยิ่งขึ้น

จากที่กล่าวไปข้างต้นว่าเอาต์พุตของโหลดเซลล์เป็นแรงดัน ดังนั้นไมโครคอนโทลเลอร์จึงจำเป็นต้องแปลงค่าแรงดันอนาลอกเป็นดิจิตอล เสียก่อน โดยปกติแล้วภายในไมโครคอนโทลเลอร์นั้นสามารถแปลงอนาลอกเป็นดิจิตอลได้แต่โครงงานนี้ต้องการความละเอียดที่สูง ซึ่งจะต้องใช้อนาลอกเป็นดิจิตอลที่มีความละเอียด 16 บิตขึ้นไป แต่ภายในไมโครคอนโทรลเลอร์สามารถปรับได้สูงสุดเพียง 10 บิตเท่านั้น ด้วยเหตุผลนี้เองจึงต้องใช้ไอซีแปลงอนาลอกเป็นดิจิตอล ภายนอกโดยในโครงงานเลือกใช้ไอซี MCP3551 มีความละเอียดสูงถึง 22 บิตซึ่งมีการต่อวงจรในโครงงานตามรูปที่ 2.3



รูปที่ 2.3 วงจรรับแรงดันจากโหลดเซลล์มาขยายและเปลี่ยนเป็นสัญญาณดิจิตอลเข้าสู่ไมโครคอนโทรลเลอร์

จากรูปที่ 2.3 เป็นการอ่านค่าจากโหลดเซลล์ที่ผ่านการขยายแล้ว เพื่อให้แรงดันสูงเพียงพอและลดค่าความผิดพลาดจากการอ่านแรงดันของอนาลอกเป็นดิจิตอลไอซีเบอร์ MCP3551 มีความเร็วการแปลงค่าต่ำแต่มีความละเอียดสูง เนื่องจากมีการวัดแบบ Delta-sigma ADC ที่เป็นการแปลงสัญญาณที่มีความละเอียดสูง เป็นการแปลงที่ถือว่าเป็นอุดมคติ และทำงานได้หลายย่านความถี่ ตั้งแต่สัญญาณไฟฟ้ากระแสตรงไปจนถึงหลักเมกะเฮิรตซ์ การทำงานของDelta-sigma ADC สัญญาณอินพุตจะถูกเก็บตัวอย่างสัญญาณ(Oversample)โดยตัวกล้ำสัญญาณ(Modulator)หลังจากนั้นจะนำสัญญาณมากรองเพื่อให้ได้ค่าที่ถูกต้องมากขึ้นโดยวงจรกรองดิจิตอล(Digital Filter)จึงทำให้ได้ค่าอนาลอกเป็นดิจิตอลที่มีความละเอียดที่มีอัตราการสุ่ม (Sampling) ต่ำ เพราะว่า Delta-sigma

ทำการเก็บตัวอย่างสัญญาณสัญญาณอินพุตจึงทำให้ได้สัญญาณเรียบขึ้นเหมาะกับงานที่เกี่ยวข้องกับเครื่องมือวัด โดยปกติแล้ว Delta-sigma จะรับสัญญาณความแตกต่างระหว่าง 2 อินพุต แทนที่จะเป็นการวัดโวลต์เทียบกราวด์ การวัดสัญญาณความแตกต่างก็จะเห็นได้ตามรูปที่ 2.3

2.4 จอแสดงผลแอลซีดี

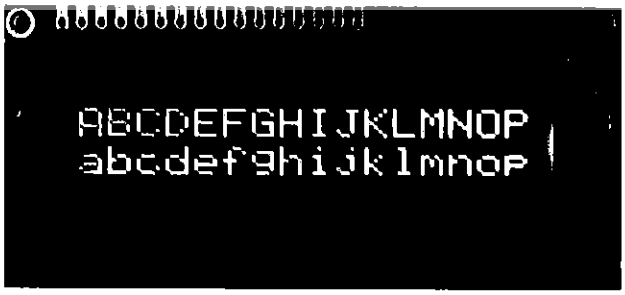
เทคโนโลยีมอนิเตอร์ แอลซีดี ย่อมาจากหน้าจอแสดงผลผลึกเหลว (Liquid Crystal Display) ซึ่งเป็นจอแสดงผลแบบดิจิตอล โดยภาพที่ปรากกฏขึ้นเกิดจากแสงที่ถูกปล่อยออกมาจากหลอดไฟด้านหลังจอภาพ (Black Light) ผ่านชั้นกรองแสง (Polarized filter) แล้ววิ่งไปยังคริสตัลเหลวที่เรียงตัวด้วยกัน 3 เซลล์ คือ แสงสีแดง แสงสีเขียว และแสงสีน้ำเงิน กลายเป็นพิกเซล (Pixel) ที่สว่างสดใสเกิดขึ้น

เทคโนโลยีที่พัฒนามาใช้กับ แอลซีดี นั้นแบ่งออกเป็น 2 ประเภทคือ

1. โพสซิธีปเมทริกซ์ (Super-Twisted Nematic) เป็นเทคโนโลยีแบบเก่าที่ให้ความคมชัดและความสว่างน้อยกว่าใช้ในจอโทรศัพท์มือถือทั่วไป

2. แอกซิธีปเมทริกซ์ (Thin Film Transistor) สามารถแสดงภาพได้คมชัดและสว่างกว่าแบบแรกใช้ในจอมอนิเตอร์หรือโน้ตบุ๊ค

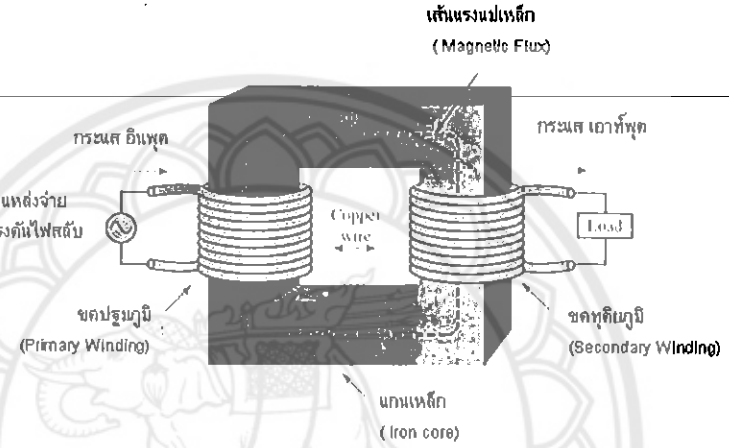
หลักการพื้นฐานคือการบังคับให้หยดของผลึกเหลว (Liquid crystal) ซึ่งมีแผ่นแก้วกักเอาไว้ให้ไปปิดรูช่องแสง ซึ่งแสงถูกฉายมาด้านหลังของหน้าจอ ก่อให้เกิดการแสดงผลเป็นตัวอักษร หรือตัวเลขในรูปแบบต่างๆ ได้ตามต้องการ จุดเด่นของหน้าจอแอลซีดีขาวดำ หรือเรียกอีกอย่างว่าหน้าจอแบบโมโนโกลด์ คือการทำงานที่ไม่อาศัยปืนยิงอิเล็กตรอน จึงช่วยให้ด้านลึกของจอภาพมีขนาดสั้นกว่ามอนิเตอร์แบบซีดีที (CDT) ถึง 3เท่าและด้วยรูปร่างที่แนบราบทางด้านหน้าและด้านหลัง ขนาดเล็กกะทัดรัดและน้ำหนักเบาและประหยัดพลังงานไฟฟ้า สำหรับการแสดงผลโครงงานนี้ได้ใช้จอแสดงผลแอลซีดีแบบ 16 ตัวอักษร 2 บรรทัด ซึ่งแสดงตัวอย่างดังรูปที่ 2.4



รูปที่ 2.4 ลักษณะของจอแอลซีดี 16 ตัวอักษร 2 บรรทัด

2.5 หม้อแปลงไฟฟ้า

การทำงานของหม้อแปลงไฟฟ้านั้น อาศัยหลักการความสัมพันธ์ระหว่างกระแสไฟฟ้ากับเส้นแรงแม่เหล็กในการสร้างแรงเคลื่อนเหนี่ยวนำให้กับตัวนำ คือ เมื่อกระแสไหลผ่านขดลวดตัวนำจะทำให้เกิดเส้นแรงแม่เหล็กรอบๆตัวนำนั้น และถ้ากระแสที่ป้อนมีขนาดและทิศทางที่เปลี่ยนแปลงไปมา จะทำให้สนามแม่เหล็กที่เกิดขึ้นมีการเปลี่ยนแปลงตามไปด้วย ถ้าสนามแม่เหล็กที่มีการเปลี่ยนแปลงดังกล่าวตัดผ่านตัวนำ จะเกิดแรงเคลื่อนเหนี่ยวนำขึ้นที่ตัวนำนั้นโดยขนาดของแรงเคลื่อนเหนี่ยวนำจะสัมพันธ์กับความเข้มของสนามแม่เหล็ก และความเร็วในการตัดผ่านตัวนำของสนามแม่เหล็ก

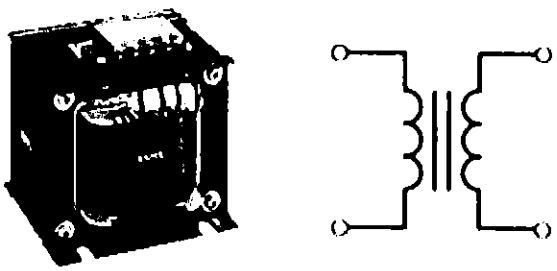


รูปที่ 2.5 โครงสร้างของหม้อแปลงไฟฟ้า

2.5.1 ชนิดของหม้อแปลงไฟฟ้า

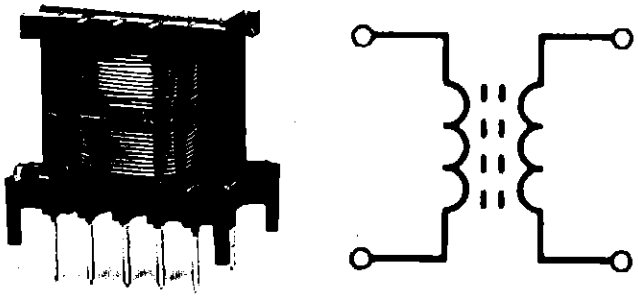
เราสามารถแบ่งชนิดของหม้อแปลงไฟฟ้าได้ตามแกนทั้ง 3 แบบ คือ

1.หม้อแปลงชนิดแกนเหล็ก (Iron core transformer) หม้อแปลงชนิดนี้จะใช้แผ่นเหล็กอ่อนหลายๆ แผ่นซึ่งส่วนใหญ่จะใช้รูปทรงตัว E และตัว I ประกอบกันเป็นแกนซึ่งส่วนใหญ่จะใช้ในงานทั่วไปที่มีความถี่ไม่สูงนัก เช่น หม้อแปลงในงานส่งกำลังไฟฟ้า หรือหม้อแปลงแปลงแรงดันไฟฟ้าตามบ้านเป็นแรงดันต่ำๆ ตามที่ต้องการ หม้อแปลงชนิดนี้มีประสิทธิภาพสูงที่สุด



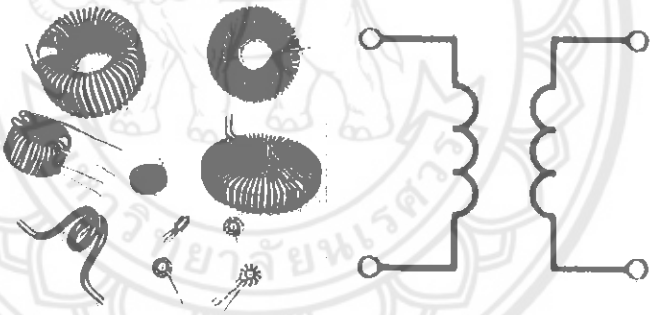
รูปที่ 2.6 ลักษณะของหม้อแปลงชนิดแกนเหล็ก

2. หม้อแปลงชนิดแกนเฟอร์ไรท์ (Ferrite core transformer) หม้อแปลงชนิดนี้ส่วนใหญ่จะใช้ในงานที่มีความถี่สูง เช่น เครื่องรับ เครื่องส่ง สัญญาณวิทยุ หรือในวงจรสวิตชิ่ง เพราะไม่สามารถใช้หม้อแปลงชนิดแกนเหล็กได้



รูปที่ 2.7 ลักษณะของหม้อแปลชนิดแกนเฟอร์ไรท์

3. หม้อแปลงชนิดแกนอากาศ (Air core transformer) หม้อแปลงชนิดนี้จะใช้ในงานความถี่สูงมากๆ เนื่องจากในงานที่ใช้ความถี่สูงมากๆหากใช้หม้อแปลงชนิดอื่นจะเกิดการสูญเสียอย่างมาก

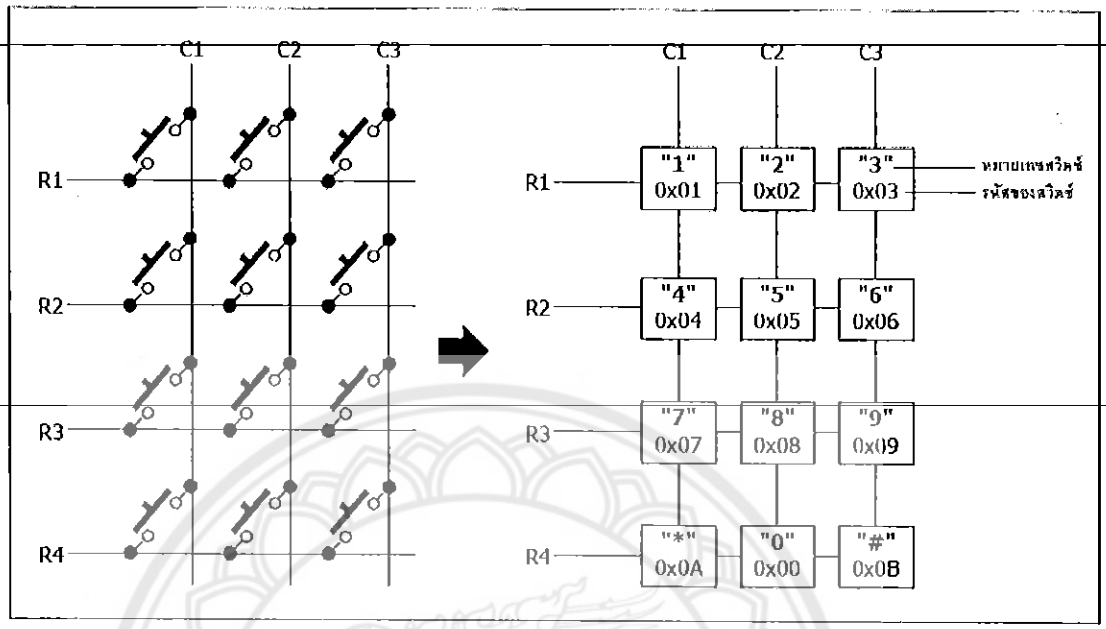


รูปที่ 2.8 ลักษณะของหม้อแปลงชนิดแกนอากาศ

2.6สวิตช์เมตริกซ์

สวิตช์เมตริกซ์(Matrix switch) หรือคีย์แพด(Keypad) ถูกนำมาใช้เป็นอุปกรณ์ในการป้อนข้อมูลให้กับงานทางด้านไมโครคอนโทรลเลอร์ นอกเหนือจากสวิตช์แบบ กดติดปล่อยดับแบบธรรมดา (Push Button Switch) โดยเฉพาะกับงานที่ต้องมีการป้อนข้อมูลทั้งตัวอักษรและตัวเลขเป็นจำนวนมาก จะพบว่าสวิตช์เมตริกซ์ถูกเลือกมาใช้งานเสมอ ที่เห็นได้ในชีวิตประจำวัน เช่น แป้นกดตัวเลขของระบบโทรศัพท์ เตาอบไมโครเวฟ เป็นต้น

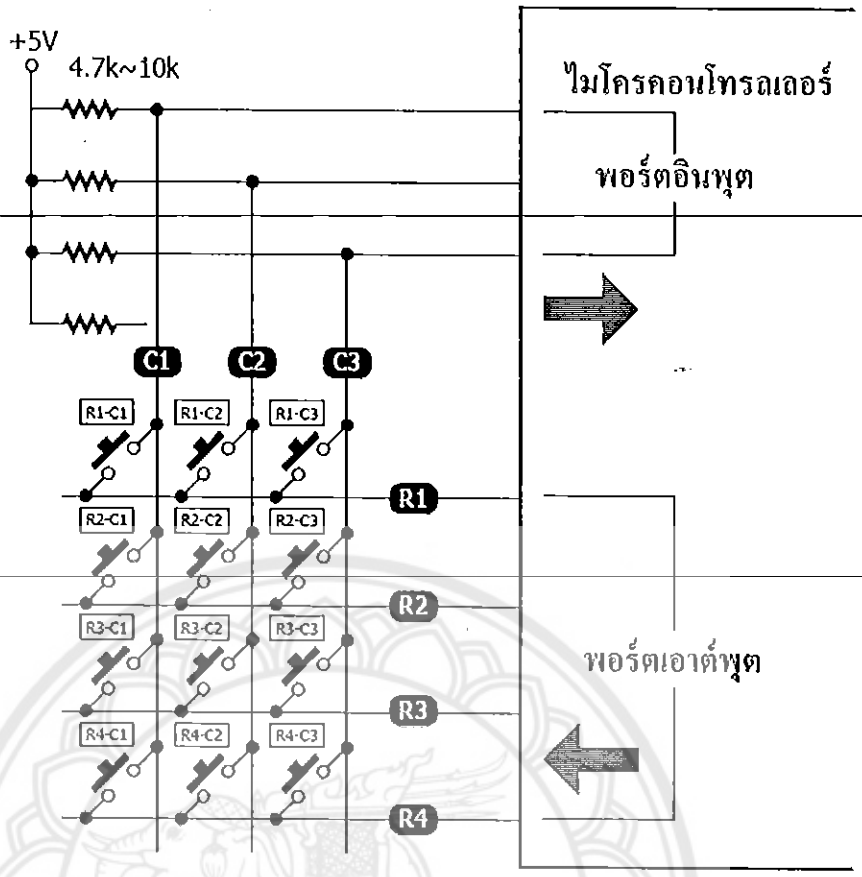
การต่อใช้งานสวิตช์แบบเมตริกซ์ เป็นการนำสวิตช์ธรรมดามาต่อกันในแบบเมตริกซ์ คือ ขาด้านหนึ่งจะต่อในแนวหลัก(column) และขาอีกด้านหนึ่งจะต่ออยู่ในแนวแถว(row) แสดงดังรูปที่ 2.9



รูปที่ 2.9 วงจรสวิตช์เมตริกซ์ ขนาด 4x3

หลักในการอ่านค่าจากคีย์แพดนี้คือ จะต้องกำหนดรหัสประจำตำแหน่งของสวิตช์แต่ละตัวไว้ไม่ให้ซ้ำกัน ดังนั้นเมื่อสวิตช์ตัวใดถูกกดจะได้ค่ารหัสของสวิตช์ตัวดังกล่าวออกมา ซึ่งจะกำหนดค่าคงที่ให้กับสวิตช์แต่ละตัวไว้ดังนี้

สวิตช์ตำแหน่ง R1 (row 1 หรือแถวที่ 1), C1 (column 1 หรือหลักที่ 1) มีค่า 0x01
สวิตช์ตำแหน่ง R1 (row 1 หรือแถวที่ 1), C2 (column 2 หรือหลักที่ 2) มีค่า 0x02
สวิตช์ตำแหน่ง R1 (row 1 หรือแถวที่ 1), C3 (column 2 หรือหลักที่ 3) มีค่า 0x03
สวิตช์ตำแหน่ง R2 (row 2 หรือแถวที่ 2), C1 (column 1 หรือหลักที่ 1) มีค่า 0x04
สวิตช์ตำแหน่ง R2 (row 2 หรือแถวที่ 2), C2 (column 2 หรือหลักที่ 2) มีค่า 0x05
สวิตช์ตำแหน่ง R2 (row 2 หรือแถวที่ 2), C3 (column 2 หรือหลักที่ 3) มีค่า 0x06
สวิตช์ตำแหน่ง R3 (row 3 หรือแถวที่ 3), C1 (column 1 หรือหลักที่ 1) มีค่า 0x07
สวิตช์ตำแหน่ง R3 (row 3 หรือแถวที่ 3), C2 (column 2 หรือหลักที่ 2) มีค่า 0x08
สวิตช์ตำแหน่ง R3 (row 3 หรือแถวที่ 3), C3 (column 2 หรือหลักที่ 3) มีค่า 0x09
สวิตช์ตำแหน่ง R4 (row 4 หรือแถวที่ 4), C1 (column 1 หรือหลักที่ 1) มีค่า 0x10
สวิตช์ตำแหน่ง R4 (row 4 หรือแถวที่ 4), C2 (column 2 หรือหลักที่ 2) มีค่า 0x00
สวิตช์ตำแหน่ง R4 (row 4 หรือแถวที่ 4), C3 (column 2 หรือหลักที่ 3) มีค่า 0x11



รูปที่ 2.10 การต่อวงจรสวิตช์เมตริกซ์ เข้ากับไมโครคอนโทรลเลอร์

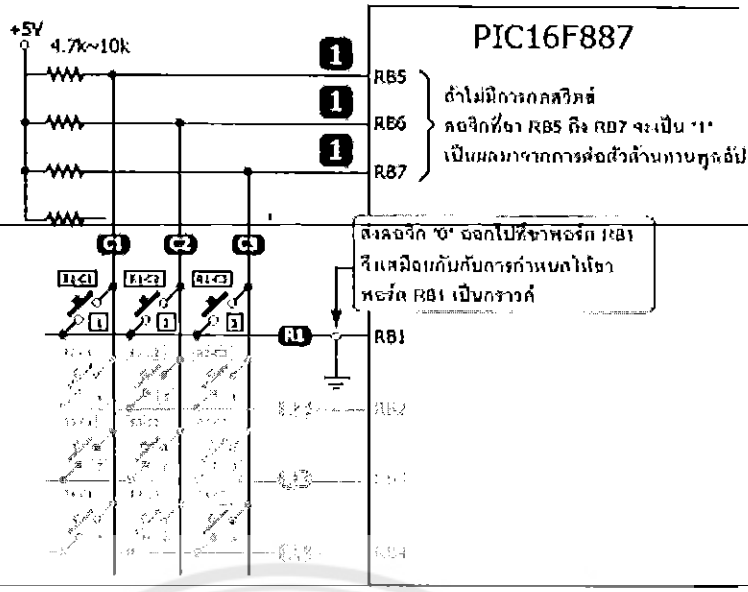
วิธีการอ่านค่าจากสวิตช์สรุปได้ดังนี้

(1) กำหนดให้ ขาพอร์ต RB5 ถึง RB7 เป็นพอร์ตอินพุตดิจิตอล (ดูรูปที่ 2.11ประกอบ)

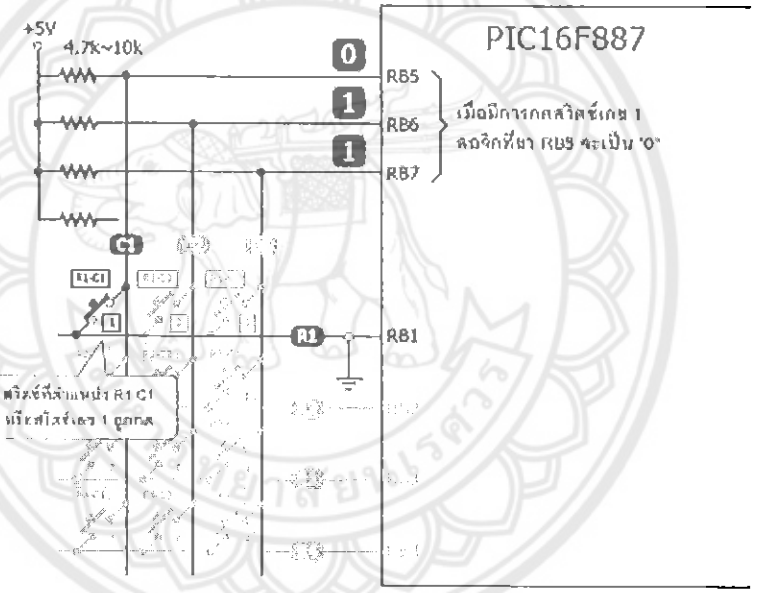
(2) กำหนดให้ ขาพอร์ต RB1 ถึง RB4 เป็นพอร์ตเอาต์พุตดิจิตอล (ดูรูปที่ 2.11 ประกอบ)

(3) ส่งลอจิก "1"ออกไปยังขาพอร์ต RB1 ถึง RB4

(4) จากนั้นเริ่มต้นการตรวจสอบแถวที่ 1 ด้วยการทำให้ ขา RB1 เป็น "0" แล้วอ่านค่าจากขาพอร์ต RB5 ถึงRB7 ว่าขาใดเป็นลอจิก "0" หรือไม่ หากขาใดเป็นลอจิก "0" นั่นคือเกิดการกดสวิตช์ที่ขานั้น หากขา RB5 เป็น "0" นั่นคือเกิดการกดสวิตช์ที่ตำแหน่ง R1, C1 ซึ่งก็คือ สวิตช์ เลข 1 โปรแกรมจะคืนค่าเป็น 0x01 กลับมา การวนรออ่านค่านี้ จะใช้เวลา 350 มิลลิวินาที หากไม่มีการกดสวิตช์เลยจะเปลี่ยนการตรวจสอบไปยังแถวที่ 2



(ก) สถานะของขาพอร์ตเมื่อเริ่มต้นการตรวจสอบการกดสวิตช์เมตริกซ์แถวที่ 1



(ข) สถานะของขาพอร์ตเมื่อมีการกดสวิตช์เมตริกซ์ตำแหน่ง C1-R1

รูปที่ 2.11 การตรวจสอบสถานะการทำงานของสวิตช์เมตริกซ์

(5) การตรวจสอบแถวที่2 จะเกิดขึ้นเมื่อทำให้ขา RB1 กลับมาเป็น “1” แล้วทำให้ขา RB2 เป็น “0” จากนั้นรออ่านค่าจากขาพอร์ต RB5 ถึง RB7 เช่นเดิม หากในคราวนี้ ที่ ขา RB5 เป็น “0” นั่นหมายถึงเกิดการกดสวิตช์ที่ตำแหน่ง R2, C1 ซึ่งก็ คือสวิตช์ เลข 4 โปรแกรมจะคืนค่าเป็น 0x04 กลับมา หากไม่มีการกดสวิตช์ เลย จะเปลี่ยนการตรวจสอบไปยังแถวที่ 3 และ 4 แล้ววนกลับมายังแถวที่ 1 อีกครั้ง

(6) เมื่อการตรวจสอบมาถึงแถวที่ 4 จะพบว่า ต้องมีการตรวจสอบสวิตช์3 ตัวคือ *, 0 และ # ดังนั้นค่าที่ ได้ จากการตรวจสอบการกดสวิตช์ จะเกิดได้ 3 ค่าคือ 10 หรือ 0x0A, 0 หรือ 0x00 และ 11 หรือ 0x0B

(7) เมื่อได้ค่าของสวิตช์แล้ว จึงนำค่าดังกล่าวไปใช้งานต่อได้ โดยเราจะนำเอาค่าตัวเลขที่ได้แสดงทางโมดูลแอลซีดีและนำไปประมวลผลเป็นค่าส่วนสูง

2.7 การอ่าน-เขียนข้อมูลEEPROM

ใน PIC16F877 จะมีหน่วยความจำข้อมูลแบบอีอีพรอมขนาด 256 ไบต์ สามารถอ่านและเขียนได้ในขณะทำงานตามปกติภายใต้ย่านไฟเลี้ยงปกติซึ่งใน CCS ได้เตรียมชุดคำสั่งอ่านเขียนข้อมูลกับหน่วยความจำไว้ในรูปของฟังก์ชั่นดั้งนี้

ฟังก์ชั่น read_eeprom()

เป็นฟังก์ชั่นสำหรับอ่านหน่วยความจำข้อมูลอีอีพรอม มีรูปแบบการใช้งานดังนี้

value = read_eeprom(address);

โดยที่

valueคือ ข้อมูล 8 บิต แบบ int

address คือ ค่าแอดเดรส 8 บิต แบบ intมีค่าตั้งแต่ 0 - 255

ฟังก์ชั่น write_eeprom()

เป็นฟังก์ชั่นสำหรับเขียนหน่ายความจำข้อมูลอีอีพรอม มีรูปแบบการใช้งานดังนี้

write_eeprom(address , value)

โดยที่

value คือ ข้อมูล 8 บิต แบบ intที่ต้องการเขียนลงในหน่วยควมจำ

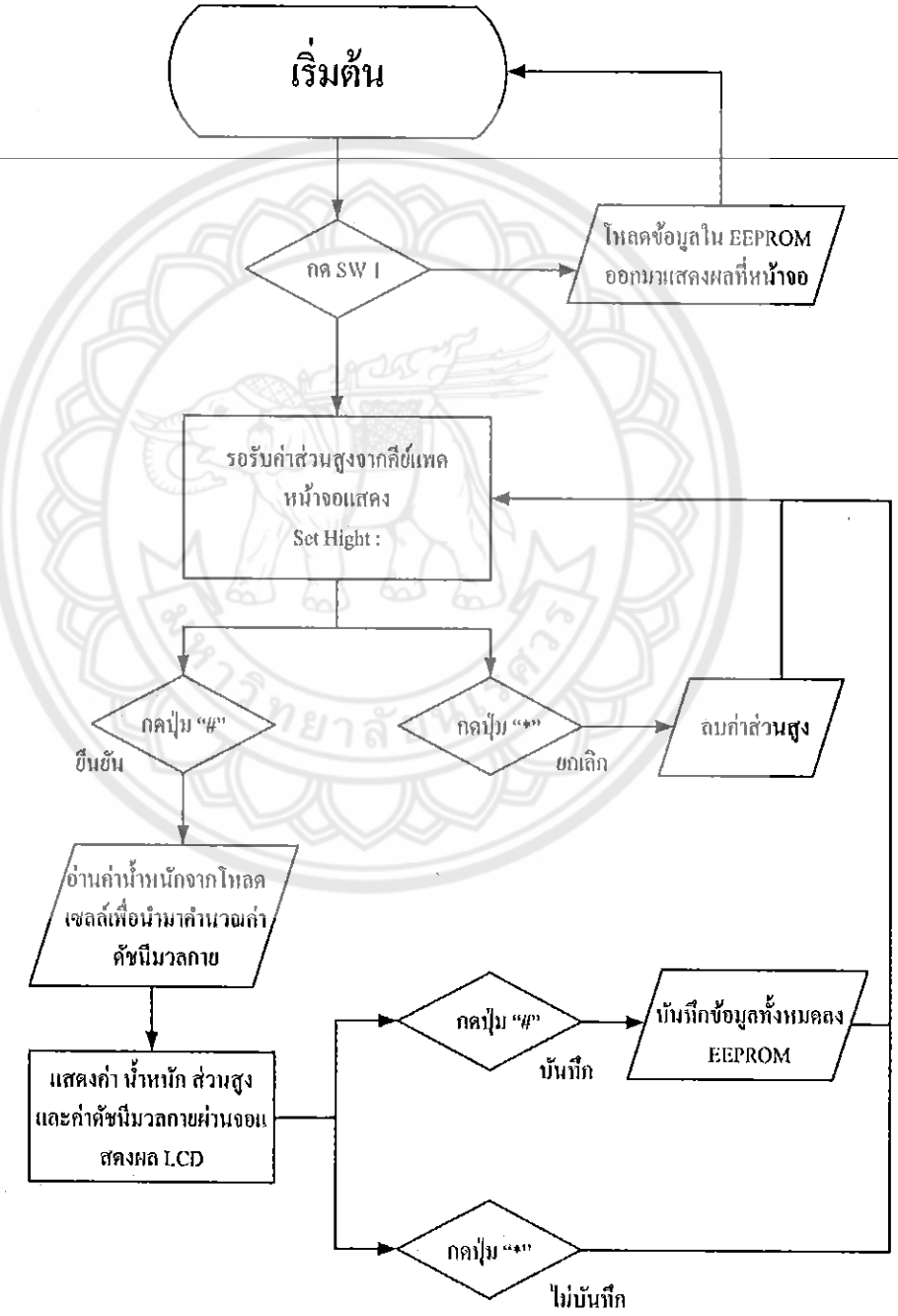
address คือ ค่าแอดเดรส 8 บิต แบบ intมีค่าตั้งแต่ 0 – 255

ในโครงงานเครื่องชั่งน้ำหนักแสดงค่าดัชนีมวลกายนี้จะเก็บข้อมูลทั้งหมด 2 รูปแบบ คือ น้ำหนักและค่าดัชนีมวลกายถึงแม้ว่าจะเก็บ 2 อย่าง แต่เนื่องด้วยหน่วยความจำของ EEPROM เป็นแบบ int 8 บิต แต่ค่าของน้ำหนักและค่าดัชนีมวลกายที่เราจะเก็บนั้นเป็นรูปแบบ float หรือจุดทศนิยม ดังนั้นจึงต้องแปลงค่าทั้งน้ำหนักและค่าดัชนีมวลกายให้เป็น int เสียก่อน โดยการแปลงข้อมูลจาก float 1 ตัวจะต้องใช้ int จำนวน 2 ไบต์ ดังนั้นในการเก็บข้อมูล 1 ครั้งจะมี น้ำหนัก 2 ไบต์ และ ค่าดัชนีมวลกายจำนวน 2 ไบต์ รวมเก็บข้อมูลการวัดหนึ่งครั้งจะใช้ EEPROM ทั้งหมด 4 ไบต์ เนื่องด้วย EEPROM ทั้งหมดมี 255 ไบต์ จึงสามารถเก็บข้อมูลได้สูงสุด ครั้ง (255/4=63) แต่เนื่องด้วยจะต้องแบ่ง EEPROM ไว้เก็บจำนวนครั้งที่เก็บไว้อีก 4 ไบต์ ดังนั้นจึงสามารถเก็บข้อมูลการวัดได้สูงสุด 62 ครั้งตามทฤษฎี

บทที่ 3

วิธีการดำเนินโครงงาน

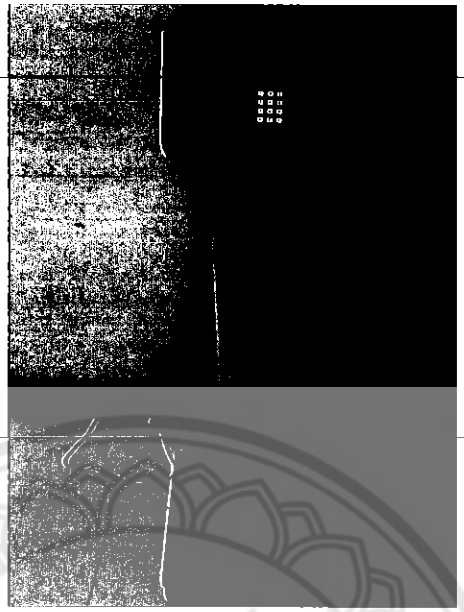
ในกรณีศึกษานี้ เป็นเรื่องการชั่งน้ำหนักและคำนวณค่าดัชนีมวลกาย หัวใจหลักของโครงงานนี้คือ ไมโครคอนโทรลเลอร์ตระกูล PIC เบอร์ PIC16F877 มีหน้าที่คือ คำนวณน้ำหนักจากการอ่านค่าแรงดันจาก MCP3551 รับค่าการกดสวิตช์ (SW1-SW3) และสวิตช์เมตริกซ์เพื่อแสดงผลผ่านทางแอลซีดี



รูปที่ 3.1 แผนภาพแสดงภาพรวมของระบบ

3.1 การออกแบบโครงสร้างของเครื่องชั่งน้ำหนัก

การออกแบบโครงสร้างของเครื่องชั่งน้ำหนักประกอบด้วยโครงสร้างหลักดังรูปที่ 3.2



รูปที่ 3.2 เครื่องชั่งน้ำหนัก

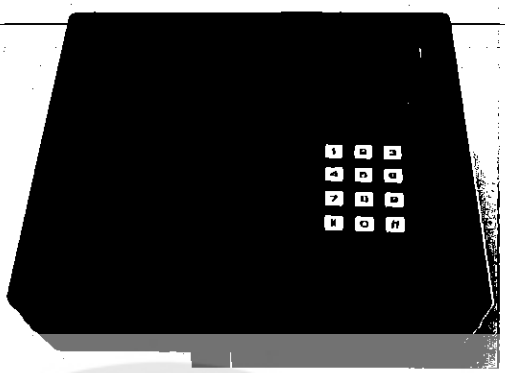
3.1.1 โครงสร้างหลักของเครื่องชั่งน้ำหนัก

โครงสร้างหลักของเครื่องชั่งน้ำหนักเป็นโครงสร้างรูปสี่เหลี่ยม ที่ทำจากเหล็กเหลี่ยม ทำให้โครงสร้างมีความแข็งแรง และมีน้ำหนักไม่มาก โดยมีลักษณะการออกแบบโครงสร้างดังรูปที่ 3.3

รูปที่ 3.3 โครงสร้างส่วนลำตัวของเครื่องชั่งน้ำหนัก

3.1.2 กล่องใส่วงจร

โครงสร้างของกล่องใส่วงจร นั้นทำจากพลาสติกบางแต่ค่อนข้างแข็ง โดยมีลักษณะเป็นกล่องทรงสี่เหลี่ยม โดยกล่องใส่วงจรจะวางไว้ตำแหน่งบนสุดของเครื่องชั่งน้ำหนักดังรูปที่ 3.4



รูปที่ 3.4 กล่องใส่วงจร

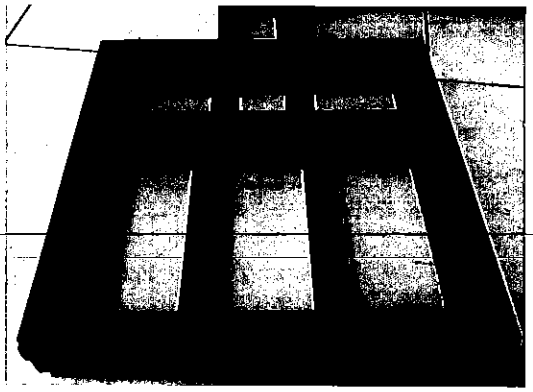
3.1.3 โหลดเซลล์

ส่วนของการรับค่าน้ำหนักจะใช้โหลดเซลล์ในการรับค่าของวัตถุดังรูปที่ 3.5

รูปที่ 3.5 โหลดเซลล์

3.1.4 ฐานรองเครื่องชั่งน้ำหนัก

ส่วนของฐานรองจะใช้เหล็กเหลี่ยมมาเชื่อมกันเป็นรูปสี่เหลี่ยมจัตุรัส เพื่อให้มีความมั่นคงแข็งแรง ดังรูปที่ 3.6



รูปที่ 3.6 ฐานรองเครื่องชั่งน้ำหนัก

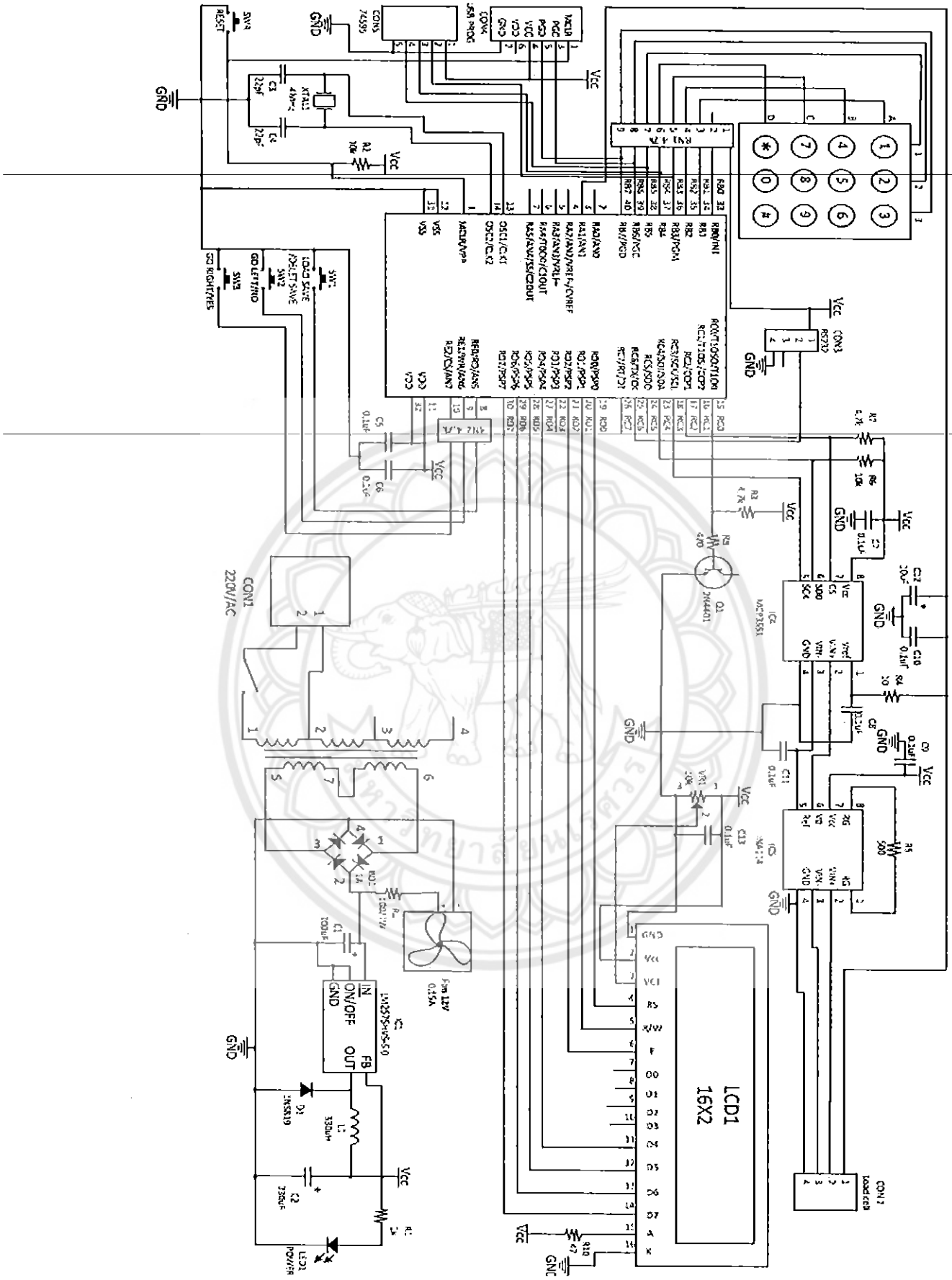
3.1.5 ฐานรองเหยียบ

ฐานเหยียบใช้เหล็กเหลี่ยมเชื่อมกันเหมือนกับฐาน แต่จะใช้แผ่นอะคริลิควางด้านบน เพื่อลดการใช้เหล็ก จึงทำให้น้ำหนักไม่สูงมาก ดังรูปที่ 3.7

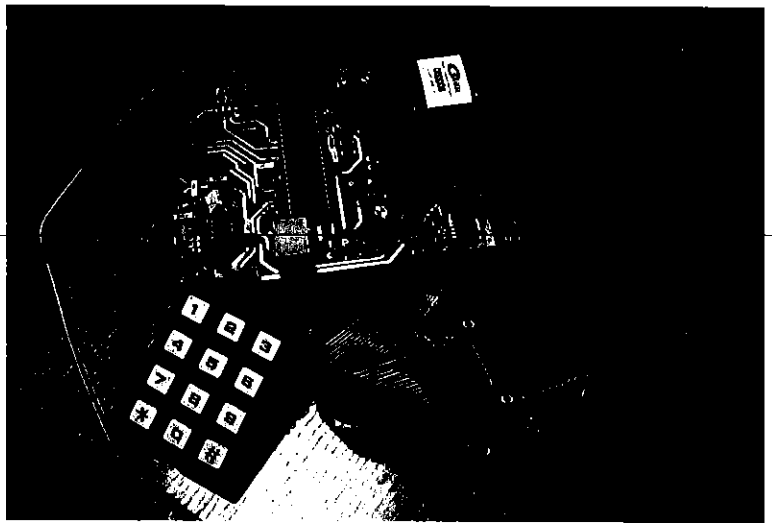
รูปที่ 3.7 ฐานรองเหยียบ

3.1.6 วงจรการทำงาน

รูปวงจรในส่วนของภาคควบคุมพร้อมทั้งแสดงผลด้วยจอแอลซีดีขนาด 16 ตัวอักษร 2 บรรทัด หัวใจหลักคือ ไมโครคอนโทรลเลอร์ตระกูล PIC เบอร์ PIC16F877 มีหน้าที่คำนวณน้ำหนักจากการอ่านค่าแรงดันจาก MCP3551 แสดงผลผ่านทางจอแอลซีดี ดังรูปที่ 3.8



รูปที่ 3.8 วงจรการทำงานของเครื่องชั่งน้ำหนัก



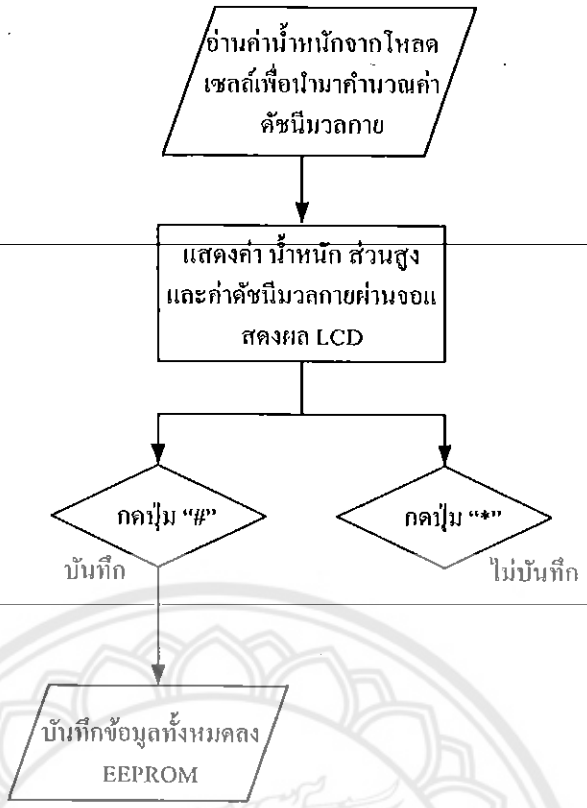
รูปที่ 3.9 ภาพวงจร(ประกอบเสร็จ)

3.2 ขั้นตอนการทำงานของเครื่องชั่งน้ำหนัก

การทำงานของเครื่องชั่งน้ำหนักแสดงค่าดัชนีมวลกายและเก็บข้อมูลได้แบ่งออกเป็น 2 ส่วน คือ ส่วนการรับค่า และส่วนการคำนวณ

3.2.1 ส่วนของการรับค่า

จากแผนภาพในรูปที่ 3.10 มีขั้นตอนการทำงานดังนี้ คือ เมื่อเปิดเครื่องเริ่มการทำงานจะปรากฏหน้าจอแสดงข้อความให้ใส่ค่าส่วนสูง หลังจากนั้นให้ทำการขึ้นชั่งน้ำหนักโดยการยืนนิ่งๆ บนฐานเครื่องชั่งน้ำหนักประมาณ 3 วินาที หลังจากนั้นโหลดเซลล์จะอ่านค่าน้ำหนักและส่งไปยังส่วนการคำนวณต่อไป



รูปที่ 3.10 แผนภาพส่วนของการรับค่า

3.2.2 ส่วนของการคำนวณ

จากแผนภาพในรูปที่ 3.11 จะแสดงการคำนวณ ซึ่งมีกระบวนการย่อยทั้งหมด 2 กระบวนการย่อย คือ

1. กระบวนการคำนวณน้ำหนัก

-ไอซี INA114 รับค่าจากโหลดเซลล์ แล้วส่งค่าไปที่ไอซี MCP3551 และไอซี PIC16F877

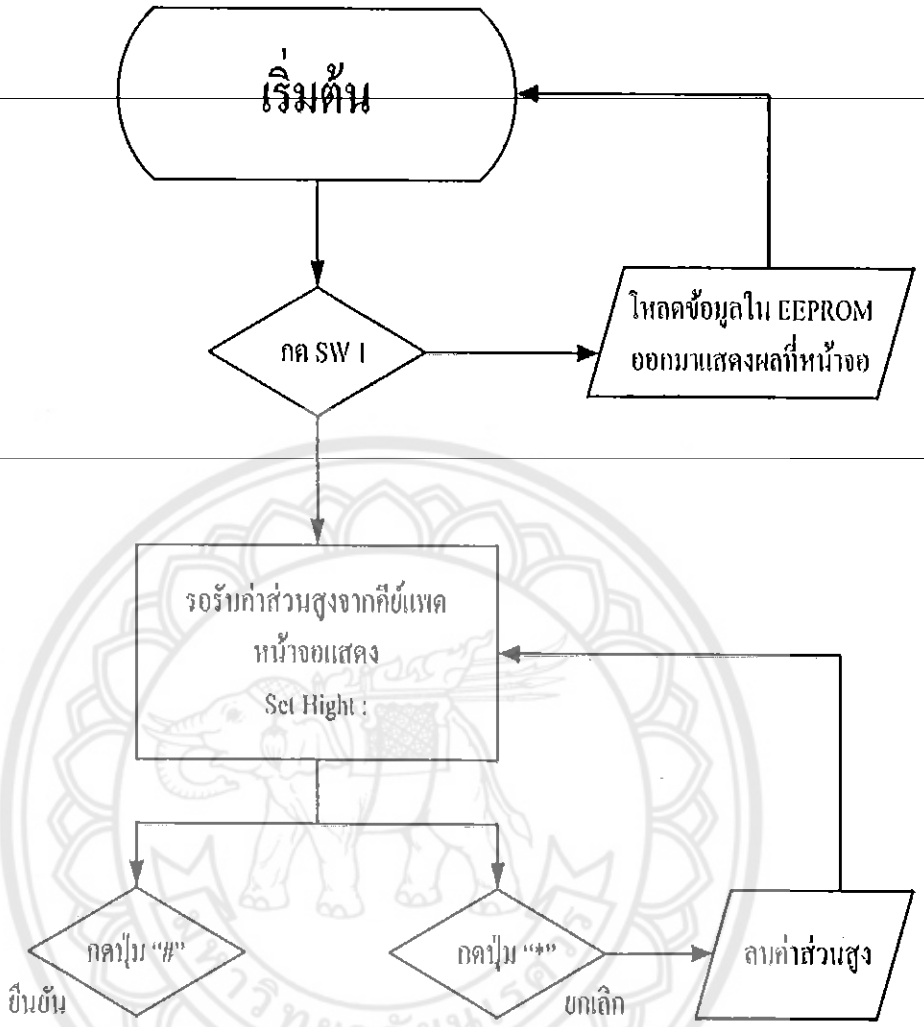
-ไอซีPIC16F877 จะคำนวณน้ำหนักโดยแรงดันที่ผ่านการแปลงด้วยความละเอียด 22 บิต จะถูกนำมาเข้ากระบวนการคำนวณด้วยอัตรา 118 ต่อ 10 กรัม (ถ้าอ่านค่าได้ 11,800 กรัม แสดงว่าน้ำหนักของเท่ากับ 1 กิโลกรัม)

2.กระบวนการคำนวณค่าดัชนีมวลกาย

-นำค่าน้ำหนักที่ได้จากการคำนวณและส่วนสูงที่ใส่ในกระบวนการรับค่า มาคำนวณโดยใช้สมการในการคำนวณดังนี้

$$\text{ค่าดัชนีมวลกาย} = \frac{\text{น้ำหนักตัว (กิโลกรัม)}}{\text{ส่วนสูง} \times \text{ส่วนสูง(เมตร)}}$$

หลังจากผู้ใช้ได้รับค่าดัชนีมวลกายมาแล้ว จะสามารถรู้ได้ว่าตนเองมีรูปร่างของร่างกายเป็นอย่างไร โดยดูจากคำอธิบายที่ติดอยู่ที่ตัวเครื่องชั่งน้ำหนัก



รูปที่ 3.11 แผนภาพส่วนของการคำนวณ

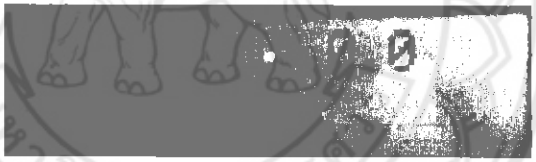
บทที่ 4

ผลการทดลอง

หลังจากสร้างเครื่องชั่งน้ำหนักแสดงค่าดัชนีมวลกายและเก็บข้อมูลได้ ผู้ดำเนินโครงงานได้ทำการทดสอบการทำงานของเครื่องชั่งน้ำหนักแสดงค่าดัชนีมวลกายและเก็บข้อมูลได้ โดยแบ่งการทดลองออกเป็น 4 การทดลอง ดังนี้

1. การทดลองความถูกต้องของการชั่งน้ำหนัก
2. การทดลองความถูกต้องของการคำนวณค่าดัชนีมวลกาย
3. การทดลองประสิทธิภาพของเครื่อง
4. การทดลองประสิทธิภาพและความถูกต้องของการบันทึกค่า

รูปที่ 4.1 หน้าจอหลักของเครื่องชั่งน้ำหนัก



รูปที่ 4.2 ภาพหน้าจอระหว่างรอการชั่งน้ำหนัก

หมายเหตุ: H คือ ส่วนสูง, W คือ น้ำหนัก, BMI คือค่าดัชนีมวลกาย

4.1การทดลองความถูกต้องของการชั่งน้ำหนัก

การทดลองนี้ ผู้ดำเนินโครงงานจะทำการทดลองโดยการนำแผ่นน้ำหนัก ขนาด 10-60 กิโลกรัม มาชั่งเพื่อเปรียบเทียบความถูกต้องของน้ำหนักที่ได้ โดยจะชั่งอย่างละ 4 ครั้ง เพื่อคำนวณหาค่าเปอร์เซ็นความผิดพลาดของการชั่งน้ำหนัก

ตารางที่ 4.1การทดลองการชั่งน้ำหนักจากแผ่นน้ำหนัก 10 กิโลกรัม

ชั่งครั้งที่	น้ำหนักที่ใช้ (กิโลกรัม)	น้ำหนักที่ชั่งได้ (กิโลกรัม)
1	10	10.0
2	10	10.0
3	10	10.0
4	10	10.0
ค่าเฉลี่ย	10	10.0
	ค่าความผิดพลาด(%)	0.00

ตารางที่ 4.2 การทดลองการชั่งน้ำหนักจากแผ่นน้ำหนัก 20 กิโลกรัม

ชั่งครั้งที่	น้ำหนักที่ใช้ (กิโลกรัม)	น้ำหนักที่ชั่งได้ (กิโลกรัม)
1	20	20.0
2	20	20.0
3	20	20.0
4	20	20.0
ค่าเฉลี่ย	20	20.0
	ค่าความผิดพลาด(%)	0.00

ตารางที่ 4.3 ตัวอย่างการชั่งน้ำหนักจากแผ่นน้ำหนัก 30 กิโลกรัม

ชั่งครั้งที่	น้ำหนักที่ใช้ (กิโลกรัม)	น้ำหนักที่ชั่งได้ (กิโลกรัม)
1	30	30.0
2	30	30.0
3	30	30.0
4	30	30.0
ค่าเฉลี่ย	30	30.0
	ค่าความผิดพลาด(%)	0.00

ตารางที่ 4.4 การทดลองการชั่งน้ำหนักจากแผ่นน้ำหนัก 40 กิโลกรัม

ชั่งครั้งที่	น้ำหนักที่ใช้ (กิโลกรัม)	น้ำหนักที่ชั่งได้ (กิโลกรัม)
1	40	40.1
2	40	40.1
3	40	40.1
4	40	40.1
ค่าเฉลี่ย	40	40.1
	ค่าความผิดพลาด(%)	0.25

ตารางที่ 4.5 การทดลองการชั่งน้ำหนักจากแผ่นน้ำหนัก 50 กิโลกรัม

ชั่งครั้งที่	น้ำหนักที่ใช้ (กิโลกรัม)	น้ำหนักที่ชั่งได้ (กิโลกรัม)
1	50	50.1
2	50	50.1
3	50	50.1
4	50	50.1
ค่าเฉลี่ย	50	50.1
	ค่าความผิดพลาด(%)	0.20

ตารางที่ 4.6 การทดลองการชั่งน้ำหนักจากแผ่นน้ำหนัก 60 กิโลกรัม

ชั่งครั้งที่	น้ำหนักที่ใช้ (กิโลกรัม)	น้ำหนักที่ชั่งได้ (กิโลกรัม)
1	60	60.2
2	60	60.2
3	60	60.2
4	60	60.2
ค่าเฉลี่ย	60	60.2
	ค่าความผิดพลาด(%)	0.33

ตารางที่ 4.7 การทดลองการชั่งน้ำหนักจากแผ่นน้ำหนัก 70 กิโลกรัม

ชั่งครั้งที่	น้ำหนักที่ใช้ (กิโลกรัม)	น้ำหนักที่ชั่งได้ (กิโลกรัม)
1	70	70.1
2	70	70.1
3	70	70.1
4	70	70.1
ค่าเฉลี่ย	70	70.1
	ค่าความผิดพลาด(%)	0.14

ตารางที่ 4.8 การทดลองการชั่งน้ำหนักจากแผ่นน้ำหนัก 80 กิโลกรัม

ชั่งครั้งที่	น้ำหนักที่ใช้ (กิโลกรัม)	น้ำหนักที่ชั่งได้ (กิโลกรัม)
1	80	80.1
2	80	80.1
3	80	80.1
4	80	80.1
ค่าเฉลี่ย	80	80.1
	ค่าความผิดพลาด(%)	0.12

ตารางที่ 4.9 การทดลองการชั่งน้ำหนักจากแผ่นน้ำหนัก 90 กิโลกรัม

ชั่งครั้งที่	น้ำหนักที่ใช้ (กิโลกรัม)	น้ำหนักที่ชั่งได้ (กิโลกรัม)
1	90	90.1
2	90	90.1
3	90	90.1
4	90	90.1
ค่าเฉลี่ย	90	90.1
	ค่าความผิดพลาด(%)	0.11

ตารางที่ 4.10 การทดลองการชั่งน้ำหนักจากแผ่นน้ำหนัก 100 กิโลกรัม

ชั่งครั้งที่	น้ำหนักที่ใช้ (กิโลกรัม)	น้ำหนักที่ชั่งได้ (กิโลกรัม)
1	100	100.1
2	100	100.1
3	100	100.1
4	100	100.1
ค่าเฉลี่ย	100	100.1
	ค่าความผิดพลาด(%)	0. 10

ตารางที่ 4.11 การทดลองการชั่งน้ำหนักจากแผ่นน้ำหนัก 110 กิโลกรัม

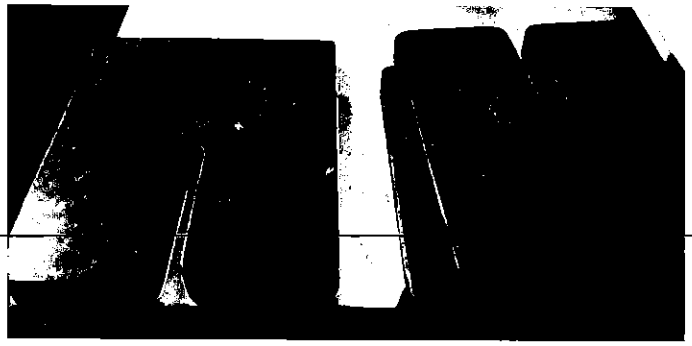
ชั่งครั้งที่	น้ำหนักที่ใช้ (กิโลกรัม)	น้ำหนักที่ชั่งได้ (กิโลกรัม)
1	110	110.1
2	110	110.1
3	110	110.1
4	110	110.1
ค่าเฉลี่ย	110	110.1
	ค่าความผิดพลาด(%)	0.09

ตารางที่ 4.12 การทดลองการชั่งน้ำหนักจากแผ่นน้ำหนัก 120 กิโลกรัม

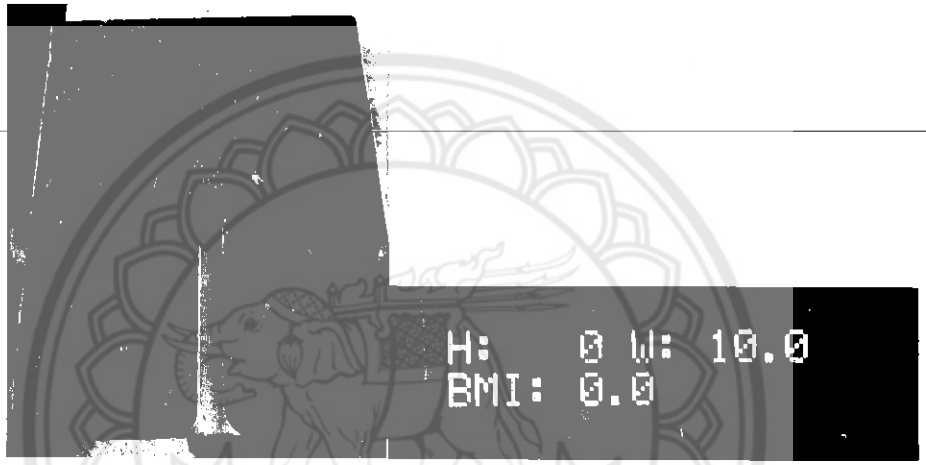
ชั่งครั้งที่	น้ำหนักที่ใช้ (กิโลกรัม)	น้ำหนักที่ชั่งได้ (กิโลกรัม)
1	120	120.0
2	120	120.0
3	120	120.0
4	120	120.0
ค่าเฉลี่ย	120	120.0
	ค่าความผิดพลาด(%)	0.00

หมายเหตุ

- ความผิดพลาดอาจเกิดขึ้นจากแผ่นชั่งน้ำหนักได้เช่นกัน



รูปที่ 4.3 แผ่นน้ำหนักที่นำมาทดลอง 10 กิโลกรัม

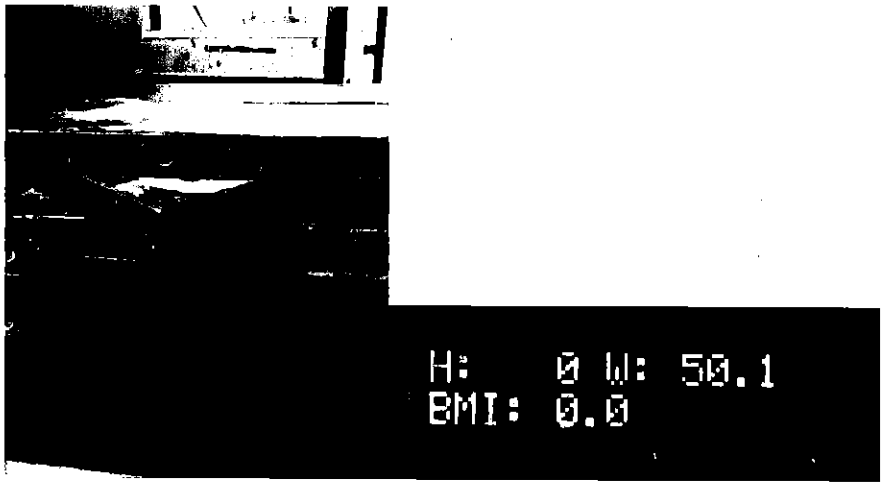


รูปที่ 4.4 ทดสอบความถูกต้องของน้ำหนักที่นำมาชั่ง 10 กิโลกรัม

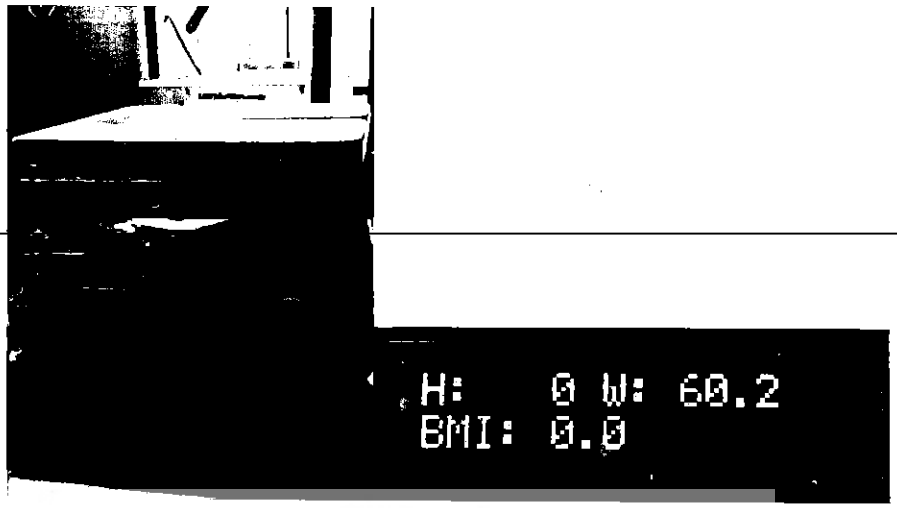
รูปที่ 4.5 ทดสอบความถูกต้องของน้ำหนักที่นำมาชั่ง 20 กิโลกรัม

รูปที่ 4.6 ทดสอบความถูกต้องของน้ำหนักที่นำมาชั่ง 30 กิโลกรัม

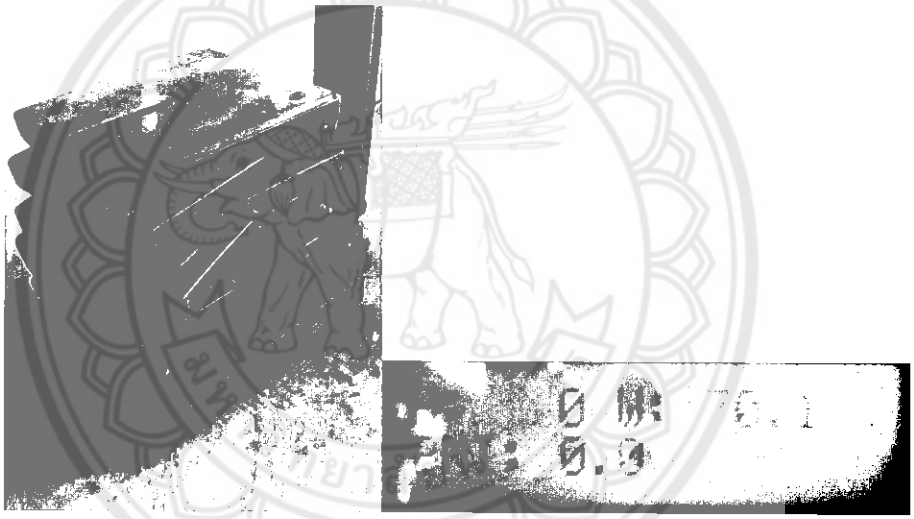
รูปที่ 4.7 ทดสอบความถูกต้องของน้ำหนักที่นำมาชั่ง 40 กิโลกรัม



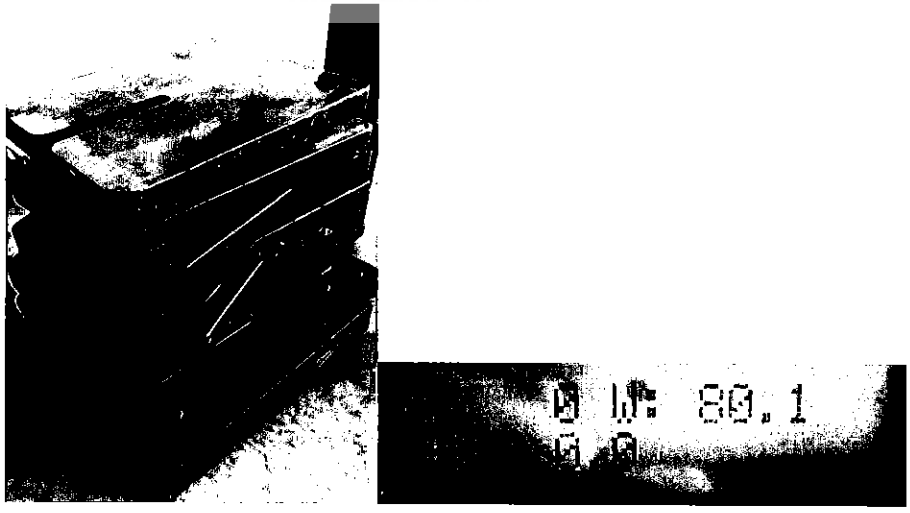
รูปที่ 4.8 ทดสอบความถูกต้องของน้ำหนักที่นำมาชั่ง 50 กิโลกรัม



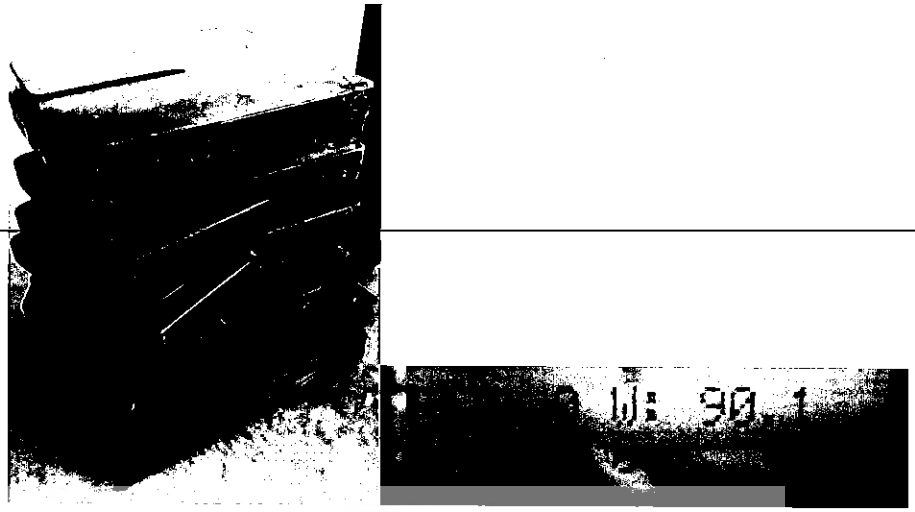
รูปที่ 4.9 ทดสอบความถูกต้องของน้ำหนักที่นำมาชั่ง 60 กิโลกรัม



รูปที่ 4.10 ทดสอบความถูกต้องของน้ำหนักที่นำมาชั่ง 70 กิโลกรัม

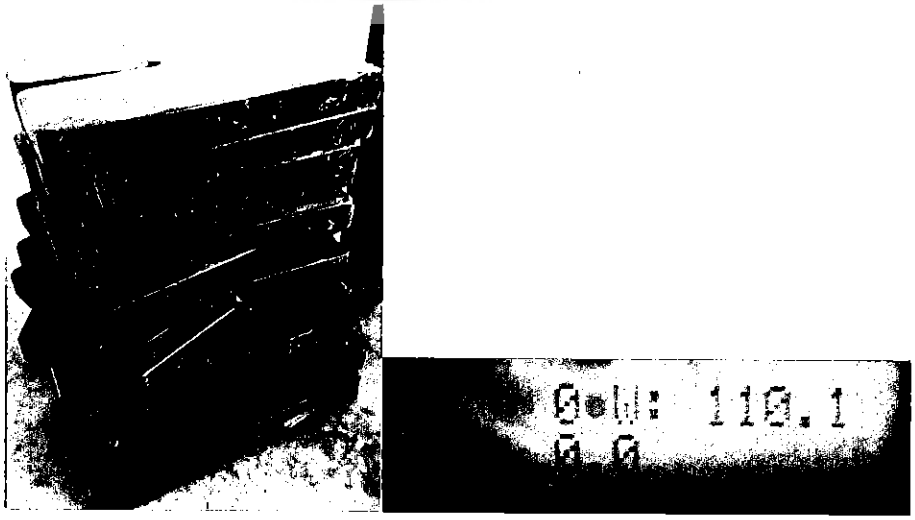


รูปที่ 4.11 ทดสอบความถูกต้องของน้ำหนักที่นำมาชั่ง 80 กิโลกรัม

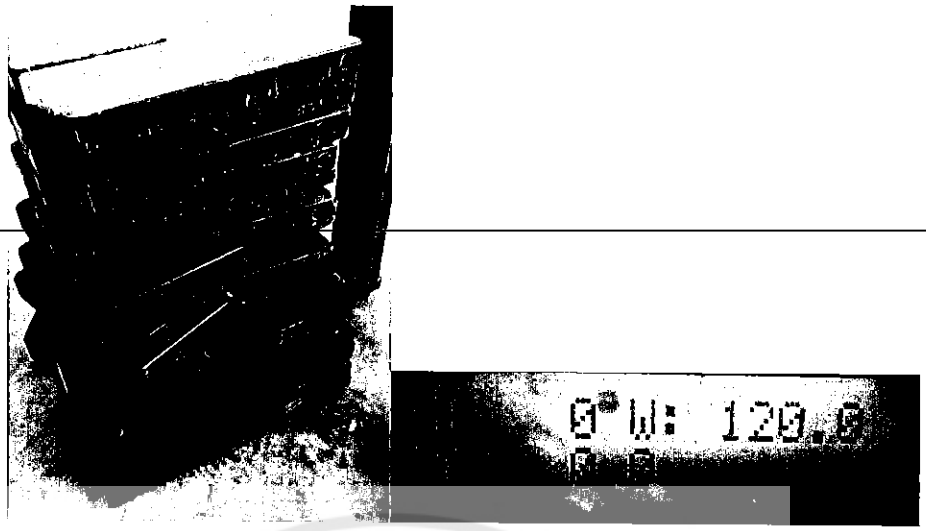


รูปที่ 4.12 ทดสอบความถูกต้องของน้ำหนักที่นำมาชั่ง 90 กิโลกรัม

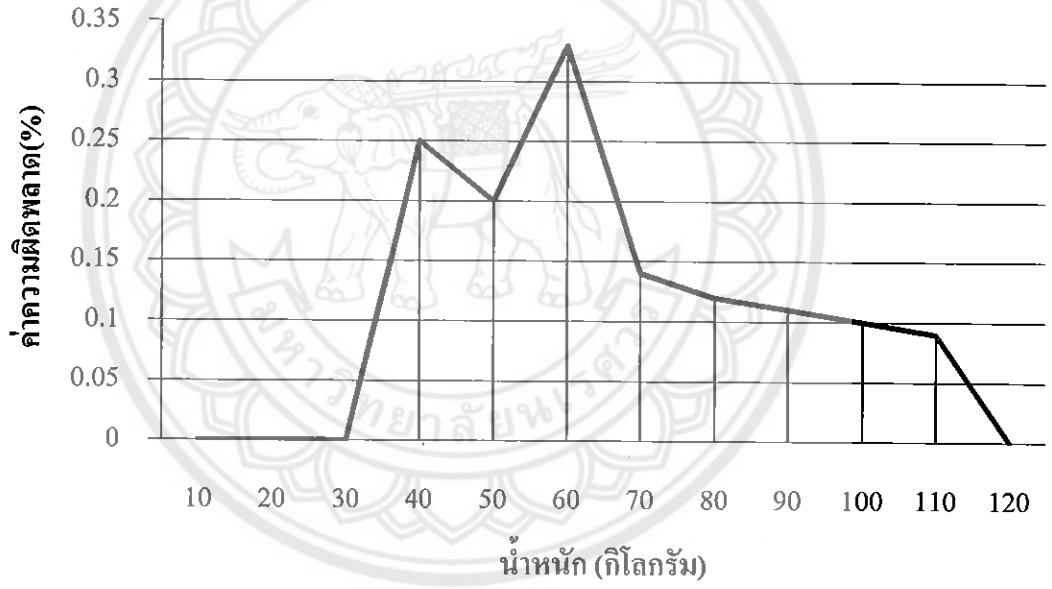
รูปที่ 4.13 ทดสอบความถูกต้องของน้ำหนักที่นำมาชั่ง 100 กิโลกรัม



รูปที่ 4.14 ทดสอบความถูกต้องของน้ำหนักที่นำมาชั่ง 110 กิโลกรัม



รูปที่ 4.15 ทดสอบความถูกต้องของน้ำหนักที่นำมาชั่ง 120 กิโลกรัม



รูปที่ 4.16 กราฟแสดงค่าความผิดพลาดของความถูกต้องของการชั่งน้ำหนัก

สรุปผลการทดลอง

จากการทดลองพบว่ามีการเกิดค่าความผิดพลาดซึ่งอาจเกิดขึ้นจากแผ่นชั่งน้ำหนักหรือตัวเครื่องเองซึ่งค่าความผิดพลาดมีค่าน้อยกว่า 0.5 %

4.2 การทดลองความถูกต้องของการคำนวณค่าดัชนีมวลกาย

การทดลองนี้ ผู้ดำเนินโครงงานจะทำการทดลองโดยการหาอาสาสมัครมาทำการทดลองใช้เครื่องชั่งน้ำหนัก แล้วนำมาเปรียบเทียบกับค่าดัชนีมวลกายที่ได้จากการคำนวณจากสมการด้านล่างนี้

$$\text{ค่าดัชนีมวลกาย} = \frac{\text{น้ำหนักตัว (กิโลกรัม)}}{\text{ส่วนสูง} \times \text{ส่วนสูง(เมตร)}}$$

ตารางที่ 4.13 การเปรียบเทียบข้อมูลจากเครื่องชั่งน้ำหนักและการคำนวณ คนที่1

ครั้งที่	ส่วนสูง (เซนติเมตร)	น้ำหนัก (กิโลกรัม)	ค่าดัชนีมวลกาย (ที่ได้จากเครื่อง)	ข้อมูลจากการคำนวณ ดัชนีมวลกาย
1	172	83.8	28.3	28.32
2	172	83.8	28.3	28.32
3	172	83.8	28.3	28.32
4	172	83.8	28.3	28.32
5	172	83.7	28.3	28.29
ค่าเฉลี่ย		83.78	28.3	28.31
			ค่าความผิดพลาด (%)	0.03

ตารางที่ 4.14 การเปรียบเทียบข้อมูลจากเครื่องชั่งน้ำหนักและการคำนวณ คนที่2

ครั้งที่	ส่วนสูง (เซนติเมตร)	น้ำหนัก (กิโลกรัม)	ค่าดัชนีมวลกาย (ที่ได้จากเครื่อง)	ข้อมูลจากการคำนวณ ดัชนีมวลกาย
1	165	87.3	32.0	32.06
2	165	87.3	32.0	32.06
3	165	87.3	32.0	32.06
4	165	87.3	32.0	32.06
5	165	87.2	32.0	32.03
ค่าเฉลี่ย		87.28	32.0	32.05
			ค่าความผิดพลาด (%)	0.16

ตารางที่ 4.15 การเปรียบเทียบข้อมูลจากเครื่องชั่งน้ำหนักและการคำนวณ คนที่3

ครั้งที่	ส่วนสูง (เซนติเมตร)	น้ำหนัก (กิโลกรัม)	ค่าดัชนีมวลกาย (ที่ได้จากเครื่อง)	ข้อมูลจากการคำนวณ ดัชนีมวลกาย
1	170	89.8	31.0	31.07
2	170	89.4	30.9	30.93
3	170	89.4	30.9	30.93
4	170	89.6	31.0	31.00
5	170	89.7	31.0	31.04
ค่าเฉลี่ย		89.58	30.96	30.99
			ค่าความผิดพลาด (%)	0.09

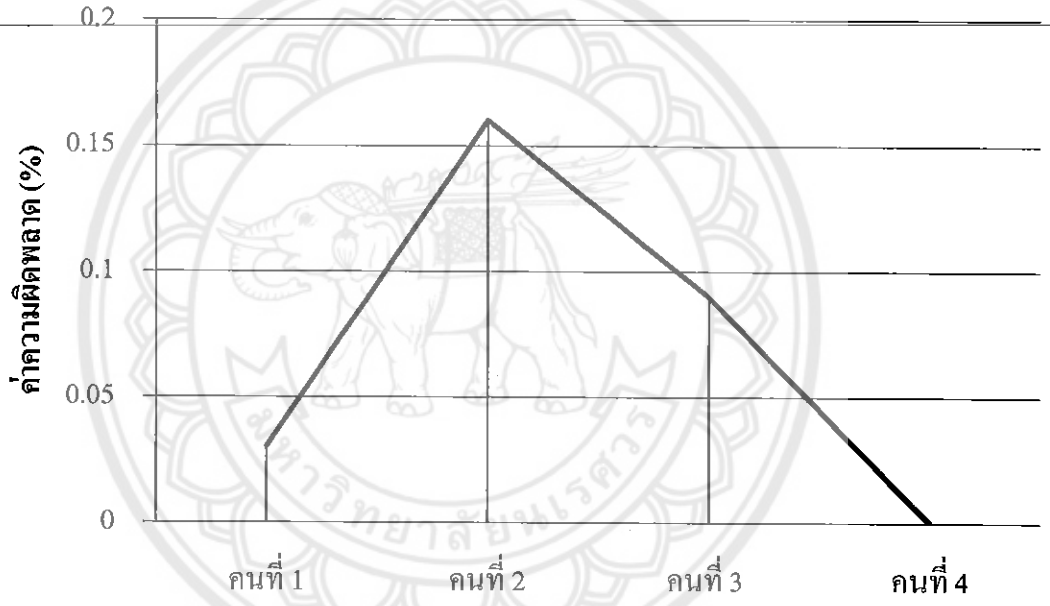
ตารางที่ 4.16 การเปรียบเทียบข้อมูลจากเครื่องชั่งน้ำหนักและการคำนวณ คนที่4

ครั้งที่	ส่วนสูง (เซนติเมตร)	น้ำหนัก (กิโลกรัม)	ค่าดัชนีมวลกาย (ที่ได้จากเครื่อง)	ข้อมูลจากการคำนวณ ดัชนีมวลกาย
1	171	54.4	18.6	18.60
2	171	54.4	18.6	18.60
3	171	54.4	18.6	18.60
4	171	54.4	18.6	18.60
5	171	54.4	18.6	18.60
ค่าเฉลี่ย		54.4	18.6	18.60
			ค่าความผิดพลาด (%)	0.00

หมายเหตุ

- ค่าที่ได้มาจากกลุ่มตัวอย่าง 4 คนชั่งน้ำหนักคนละ 5 ครั้ง

รูปที่ 4.17 อาสาสมัครทำการทดลองใช้เครื่องชั่งน้ำหนัก



รูปที่ 4.18 กราฟแสดงค่าความผิดพลาดของความถูกต้องของการคำนวณค่าดัชนีมวลกาย

สรุปผลการทดลอง

จากผลการทดลองพบว่าค่าความผิดพลาดที่เกิดขึ้นเกิดจากขั้นตอนการขึ้นชั่งน้ำหนักของผิดบุคหรืออาจเกิดจากความผิดพลาดของการอ่านค่าน้ำหนักของตัวเครื่อง แต่การคำนวณค่าดัชนีมวลกายสามารถคำนวณได้ถูกต้องตามทฤษฎี

4.3 การทดลองประสิทธิภาพของเครื่อง

การทดลองนี้ ผู้ดำเนินโครงงานจะทำการทดลองโดยการเปิดเครื่องทิ้งไว้และชั่งน้ำหนักตลอดเวลาที่เปิดเครื่อง เพื่อดูถึงประสิทธิภาพของเครื่องชั่งน้ำหนัก และตรวจสอบการบริโภคพลังงานไฟฟ้าของเครื่องชั่งน้ำหนัก ดังตารางที่ 4.3

ตารางที่ 4.17 การชั่งน้ำหนักตามช่วงเวลาการเปิดเครื่องทิ้งไว้

ระยะเวลา (นาที)	ส่วนสูง (เซนติเมตร)	น้ำหนัก (กิโลกรัม)	ค่าดัชนีมวลกาย
0	172	84.1	28.4
10	172	84.3	28.4
20	172	84.6	28.5
30	172	84.5	28.5
40	172	84.5	28.5
50	172	84.5	28.5
60	172	84.6	28.5
	ค่าเฉลี่ย	84.44	28.47

หมายเหตุ

- การชั่งแต่ละครั้งน้ำหนักแตกต่างกันเกิดจากการทิ้งน้ำหนักของผู้ชั่งและความผิดพลาดของเครื่อง

สรุปผลการทดลอง

จากการทดลองเปิดเครื่องไว้ตลอดแล้วขึ้นชั่งน้ำหนักทุกๆ 10 นาที ปรากฏว่าเครื่องชั่งน้ำหนักยังแสดงค่าไม่คาดเคลื่อนและใช้งานได้ตามปกติ

การบริโภคพลังงานไฟฟ้า

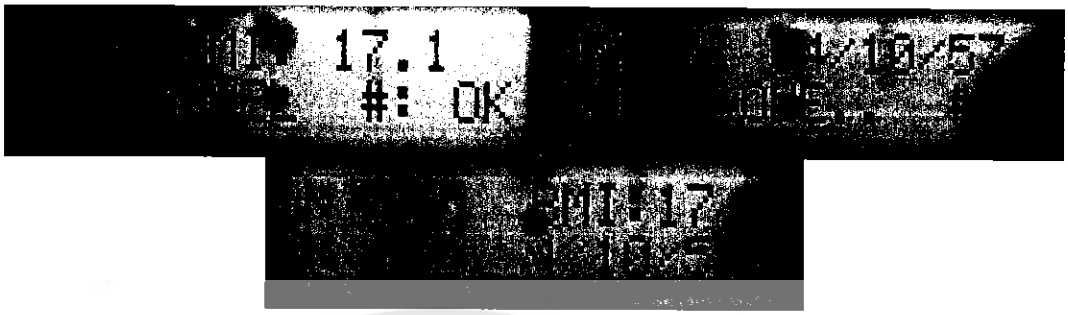
การบริโภคพลังงานไฟฟ้าของเครื่องชั่งน้ำหนักนี้ อยู่ที่ประมาณ 10 วัตต์ต่อชั่วโมง เทียบเท่ากับหลอดไฟ 1 หลอด

4.4 การทดลองประสิทธิภาพและความถูกต้องของการบันทึกค่า

การทดลองนี้ ผู้ดำเนินโครงงานจะทำการทดลองโดยการชั่งน้ำหนัก เพื่อบันทึกค่าน้ำหนัก และค่าดัชนีมวลกาย ไว้ในเครื่องชั่งน้ำหนักนี้ แล้วเรียกดูค่าที่บันทึกมาภายหลัง ดังที่แสดงในรูปด้านล่างนี้

รูปที่ 4.19 หน้าจอหลักของเครื่องชั่งน้ำหนัก

รูปที่ 4.20 หน้าจอรอการป้อนค่าวัน/เดือน/ปี ก่อนการบันทึก



รูปที่ 4.21 การแสดงข้อมูลการชั่งน้ำหนักเปรียบเทียบกับข้อมูลที่บันทึก ค่าที่ 1
ข้อมูลการชั่งน้ำหนัก(รูปบน),ข้อมูลที่บันทึก(รูปล่าง)

รูปที่ 4.22 การแสดงข้อมูลการชั่งน้ำหนักเปรียบเทียบกับข้อมูลที่บันทึก ค่าที่ 2
ข้อมูลการชั่งน้ำหนัก(รูปบน),ข้อมูลที่บันทึก(รูปล่าง)

รูปที่ 4.23 การแสดงข้อมูลการชั่งน้ำหนักเปรียบเทียบกับข้อมูลที่บันทึก ค่าที่ 30
ข้อมูลการชั่งน้ำหนัก(รูปบน),ข้อมูลที่บันทึก(รูปล่าง)

หมายเหตุ: # คือ ลำดับข้อมูล, W คือ น้ำหนัก, BMI คือค่าดัชนีมวลกาย

สรุปผลการทดลอง

จากการทดลองแสดงให้เห็นว่าเครื่องชั่งน้ำหนักเครื่องนี้ บันทึกข้อมูลได้ 30 ข้อมูล หากพื้นที่ความจำมีความจุข้อมูลเต็ม 30 ข้อมูล แล้วผู้ชั่งกดบันทึกข้อมูลที่ 31 ข้อมูลที่ 31 จะบันทึกลงไปในหน่วยความจำเป็นข้อมูลที่ 30 และข้อมูลตั้งแต่ข้อมูลที่ 2 ถึง 30 เดิมจะถูกเลื่อนขึ้นหนึ่งลำดับ และข้อมูลที่ 1 จะหายไป

บทที่ 5

สรุปผลการทดลองและข้อเสนอแนะ

จากการดำเนินโครงงานสามารถสรุปผล ชี้แจงปัญหาที่เกิดขึ้นในระหว่างการดำเนินงาน รวมทั้งแนวทางการแก้ปัญหา พร้อมให้ข้อเสนอแนะในการนำโครงงานไปพัฒนาต่อไป

5.1 สรุปผลการดำเนินโครงงาน

โครงงานเครื่องชั่งน้ำหนักแสดงค่าดัชนีมวลกายและเก็บข้อมูลได้ เป็นการออกแบบและสร้างต้นแบบระบบชั่งน้ำหนักแสดงค่าดัชนีมวลกายและเก็บข้อมูลได้เพื่อทำการศึกษาหลักการของการชั่งน้ำหนักและการคำนวณค่าดัชนีมวลกาย

โดยการดำเนินการได้ทำการเขียนโปรแกรม 3 ส่วนคือ ส่วนโปรแกรมการคำนวณน้ำหนักจากโหลดเซลล์ ส่วนโปรแกรมแสดงผลจอแอลซีดี และส่วนโปรแกรมการรับค่าจากสวิตช์เมตริกซ์ โดยส่วนโปรแกรมการรับค่าจากสวิตช์เมตริกซ์ จะส่งค่าไปยังส่วนโปรแกรมการคำนวณน้ำหนักจากโหลดเซลล์ เพื่อประมวลผลการคำนวณ และแสดงค่าออกทางส่วนโปรแกรมแสดงผลจอแอล-ซีดี

5.2 ปัญหาและแนวทางการแก้ไข

1. ระหว่างการชั่งน้ำหนัก ผู้ชั่งจะต้องยืนเพื่อรอการประมวลค่าน้ำหนักโดยใช้เวลา 3 - 15 วินาที

แนวทางการแก้ไขปัญหา ควรเปลี่ยนไปใช้โหลดเซลล์ชนิดอื่นหรือใช้วิธีอื่นในการชั่งน้ำหนัก

2. ไมโครคอนโทรลเลอร์ เบอร์ PIC16F877มีหน่วยความจำน้อย ทำให้ไม่สามารถเก็บข้อมูลได้มากพอ

แนวทางการแก้ไขปัญหา ควรเปลี่ยนไปใช้ ไมโครคอนโทรลเลอร์เบอร์อื่น ที่มีหน่วยความจำที่เยอะขึ้นอาทิเช่น เบอร์ PIC18F452 เป็นต้น

5.3 แนวทางการพัฒนาโครงงาน

1. เครื่องชั่งน้ำหนักแสดงค่าดัชนีมวลกายและเก็บข้อมูลได้ ควรมีการทำให้สามารถหยอดเหรียญได้ เพื่อนำไปใช้ประโยชน์ในเชิงพาณิชย์ได้

2. เครื่องชั่งน้ำหนักแสดงค่าดัชนีมวลกายและเก็บข้อมูลได้ ควรมีที่วัดส่วนสูงไว้ในตัว เพื่อสะดวกสำหรับผู้ที่ไม่ทราบส่วนสูงของตัวเอง

3. เครื่องชั่งน้ำหนักแสดงค่าดัชนีมวลกายและเก็บข้อมูลได้ ควรทำให้เก็บข้อมูลได้มากขึ้น

ทั้งนี้แนวทางในการพัฒนาเพื่อเพิ่มศักยภาพการทำงานของเครื่องชั่งน้ำหนักแสดงค่าดัชนีมวลกายและเก็บข้อมูลได้ เพื่อเป็นต้นแบบในการพัฒนาระบบควรคำนึงถึงประโยชน์ที่ได้รับและความคุ้มค่าในการลงทุน แต่เนื่องจากโครงงานนี้สามารถนำไปเป็นตัวอย่างทั้งในเรื่องของการศึกษาและการใช้งานจริงได้จึงควรมีการศึกษาและพัฒนาสร้างต่อเพื่อให้เกิดประโยชน์ต่อไป

เอกสารอ้างอิง

[1]ปรัชญา ศิริภูรี(2550) "วิชาการวิเคราะห์และออกแบบระบบ"

[2]ศมิทธิ์ เอมสมบัติ"(PICKit2 Lite) เครื่องโปรแกรม MCU-PIC ผ่านทาง USB พอร์ต โดยใช้ connector แบบ ICD2 ตามมาตรฐาน Microchip"

[3] ศิวะพงษ์ ธิโสภา (2556) "ไมโครคอนโทรลเลอร์ PIC"

[4] ศิวะพงษ์ ธิโสภา (2555) "การใช้งาน ccs compiler"

[5]อภิรักษ์ นามแถ่ง (2555) "วารสาร Semiconductor Electronics Plus+ ฉบับที่ 367" กรุงเทพฯ

ภาคผนวก ก

โปรแกรมการคำนวณน้ำหนักจากโหลดเซลล์

```
#include <16F877.h>
//#fuses XT, NOWDT, NOLVP
#FUSES NOWDT              //No Watch Dog Timer
#FUSES XT                 //Crystal osc <= 4mhz for PCM/PCH , 3mhz to 10 mhz for PCD
#FUSES NOPUT              //No Power Up Timer
#FUSES NOPROTECT          //Code not protected from reading
#FUSES BROWNOUT           //Reset when brownout detected
#FUSES NOLVP              //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOCPD              //No EE protection
#FUSES NOWRT              //Program memory not write protected
#FUSES NODEBUG            //No Debug mode for ICD

#device *=16
//#use delay(clock=4000000)
#use delay(crystal=4000000)

#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

#rom
0x2100={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}

//#use fast_io(c)
//#use fast_io(d)

#define SPI_MODE_0_0 (SPI_L_TO_H | SPI_XMIT_L_TO_H)
#define SPI_MODE_0_1 (SPI_L_TO_H)
#define SPI_MODE_1_0 (SPI_H_TO_L)
```

```
#define SPI_MODE_1_1 (SPI_H_TO_L | SPI_XMIT_L_TO_H)
#define loop_get  10//15//10//25
#define power_chip   PIN_A1
//#define Compensate   0
#define up   PIN_E1
#define setup   PIN_E0
#define down   PIN_E2
#define sub_kg   2.935//2.8//118

Unsigned int32 value=0,value_save=0,start_kg=0;
int16 kilo;
int16 num1,num2;
int8 i=0,p=12,value_hight,temp_hight[3],r;
//int8 sec,min,hour,date,month,year;
int8 bmi_check=0,bmi_int=0,count_loop=0;
//Signed int8 comp=0;
float bmi,kilo_f;
//float comp2=0,bmi,temp_bmi,kilo_f,bmi_old;
int1 flag_subval=0;
char k;
int8 sector_ee=0,addr_count_loop=0x00,addr_full_eeprom=0xFF;
int1 flag_full_eeprom=0;
//int32 sector_ee=0,addr_count_loop=0x00;

#include "flex_lcd.c"
#include "KBD.C"
#include "math.h"
union combine
{
  int32 word;
  int8 b[4];
};
```

```
#byte PORTC=0x7
#bit SDI=PORTC.4
//will need a bit defintion here for CS
//#byte PORTD=0x8
#bit CS=PORTC.2

int32 read3551(void)
{
  union combine temp;

  //drop CS
  CS=0;
  delay_cycles(1);
  CS=1;
  delay_cycles(1);
  //At this point conversion has been triggered.
  CS=0;
  delay_us(1);
  //wait for conversion to complete
  while (SDI==1)
  {
    CS=1;
    delay_cycles(1);
    CS=0;
  }
  //Now read data.
  temp.b[2]=spi_read(0);
  temp.b[1]=spi_read(0);
  temp.b[0]=spi_read(0);
  temp.b[3]=0;
  CS=1;
```

```
  return temp.word;
}
void reset_kg()
{
  int8 j;
  lcd_gotoxy(1,1);
  lcd_putc("   CALIBRATE    ");
  lcd_gotoxy(1,2);
  lcd_putc(" PLEASE WAIT... ");

  for (j=0; j<loop_get; j++)
  {
   output_high(power_chip);
   value = read3551();
   value_save = value_save+value;
  }
  value_save = 0;
  value = 0;
  for (j = 0; j<loop_get; j++)
  {
   output_high(power_chip);
   value = read3551();
   value_save = value_save+value;
  }
  start_kg = (value_save/loop_get);
//  start_kg = (value_save/loop_get)-Compensate;
//  lcd_gotoxy(1,2);
//  printf(LCD_PUTC, "start_kg = %lu  ",start_kg);
  value_save = 0;
  i=0;

///  delay_ms(100);
```

```
}

void read_kg()
{
  value = read3551();
  value_save = value_save+value;

//  printf("value = %lu\n\r",value);////////////////////////////////////////////////////
  i++;
  if (i>=loop_get)
  {
    value_save = value_save/loop_get;
    value_save = value_save - start_kg;
    flag_subval = 0;

    if ((value_save > 4000000000)&&(value_save < 4294967199))
    {
      flag_subval = 1;
    }
    if (value_save > 4294967200)
    {
      value_save = 0;
    }
    kilo = value_save/sub_kg;
//    kilo = value_save/comp2;
//    kilo = value_save/118.0;///comp2

//    printf("kilo = %lu\n\r",kilo);////////////////////////////////////////////////////

////////////error sub value///////////////
//    if (kilo>4000)
//    {
```

```
//        kilo = 0;
//        flag_subval = 1;
//    }
////////////////////////////////////////
//    display_lcd();
      kilo_f = kilo;
      kilo_f = kilo_f/100;
//    lcd_gotoxy(1,1);
//    lcd_putc(" Weight Digital ");
//    lcd_gotoxy(1,2);
//    printf(LCD_PUTC, "kilo= %2.1f ",kilo_f);

      i = 0;
      value_save = 0;
   }
//kilo_f = 87.5;
//kilo = 87.5*100;
}
void write_data_eeprom(int8 day, int8 mount, int8 year)
{
  int16 show_lcd;
  int16 bmi_ten,bmi_point;
  show_lcd = kilo;
  num1 = show_lcd/100;
  num2 = (show_lcd-(num1*100))/10;
  bmi_ten = bmi;
  bmi_point = (bmi*10)-(bmi_ten*10);
////////////////////////////////////////////////////////////////
  if(count_loop>30)/////////limit 255 time for alarm
  {
    flag_full_eeprom = 1;
    write_eeprom (addr_full_eeprom, flag_full_eeprom);
```

```
    count_loop = 0;
    sector_ee = 1;
  }
  write_eeprom (addr_count_loop, count_loop);////
//////////////////////////////////////////////////////////////////////

  value_hight = count_loop+1;/////////seqence

  write_eeprom (sector_ee,value_hight);// number for get;
  sector_ee++;
  delay_us(5);

  write_eeprom (sector_ee,num1);///kilo ten////   writeByte_24512x8(sector_ee,month);
  sector_ee++;
  delay_us(5);

  write_eeprom (sector_ee,num2);//kilo point//   writeByte_24512x8(sector_ee,year);
  sector_ee++;
  delay_us(5);

  write_eeprom (sector_ee,bmi_ten);//   writeByte_24512x8(sector_ee,hour);
  sector_ee++;
  delay_us(5);

  write_eeprom (sector_ee,bmi_point);//   writeByte_24512x8(sector_ee,hour);
  sector_ee++;
  delay_us(5);
//////////////////Add day mouny year////////////////
  write_eeprom (sector_ee,day);
  sector_ee++;
  delay_us(5);
```

```
  write_eeprom (sector_ee,mount);
  sector_ee++;
  delay_us(5);

  write_eeprom (sector_ee,year);
  sector_ee++;
  delay_us(5);
////////////////////////////////////////////////////////
  count_loop++;
  lcd_gotoxy(1,2);
  printf(LCD_PUTC, "SAVE DONE..  #%2u",count_loop);
//  if(count_loop>30)/////////limit 255 time for alarm
//  {
//    flag_full_eeprom = 1;
//    write_eeprom (addr_full_eeprom, flag_full_eeprom);
//    count_loop = 0;
//    sector_ee = 1;
//  }
  write_eeprom (addr_count_loop, count_loop);////



}

void read_data_eeprom()
{
int8 m3;
int16 ee_show=0x01;
int8 data[8],count_ee=0;

  ee_show=0x01;
  lcd_gotoxy(1,1);
  lcd_putc("\f READ DATA BMI  ");
```

```
  delay_ms(500);
////////////////////////////////////////////////////////
  if (flag_full_eeprom == 1)
  {
    count_loop = 30;
  }
  else count_loop = read_eeprom (addr_count_loop);
////////////////////////////////////////////////////////
  if (count_loop==0)
  {
    lcd_gotoxy(1,1);
    lcd_putc("\f DON'T HAVE BMI ");
    goto end_read;
  }
  count_ee = 0;

//        for (m3=0;m3<5;m3++)
  for (m3=0;m3<8;m3++)///add day mount year
  {
    data[m3]=read_eeprom(ee_show);/////////date
    delay_ms(1);
    ee_show++;
  }

//  lcd_gotoxy(1,1);
//  printf(LCD_PUTC, "#: %2u W: %2u.%1u    ",data[0],data[1],data[2]);
//  lcd_gotoxy(1,2);
//  printf(LCD_PUTC,"BMI:%2u.%1u %2u/%2u/%2u"data[3],data[4],data[5],data[6],data[7]);

  lcd_gotoxy(1,1);
  printf(LCD_PUTC, "W:%2u.%1u  BMI:%2u.%1u ",data[1],data[2],data[3],data[4]);
  lcd_gotoxy(1,2);
```

```
  printf(LCD_PUTC,"#:%2u   %2u/%2u/%2u"data[0],data[5],data[6],data[7]);

  delay_ms(2000);

read_ee:  if (!input(up))
   {
    delay_ms(150);
    if (!input(up))
    {
      count_ee++;
      if(count_ee<count_loop)
      {
       ee_show = (count_ee*8)+1;
//        ee_show = (count_ee*5)+1;
       for (m3=0;m3<8;m3++)
       {
        data[m3]=read_eeprom(ee_show);////////date
        delay_ms(1);
        ee_show++;
       }

       lcd_gotoxy(1,1);
       printf(LCD_PUTC, "W:%2u.%1u  BMI:%2u.%1u ",data[1],data[2],data[3],data[4]);
       lcd_gotoxy(1,2);
       printf(LCD_PUTC,"#:%2u   %2u/%2u/%2u"data[0],data[5],data[6],data[7]);

//        lcd_gotoxy(1,1);
//        printf(LCD_PUTC, "#: %2u W: %2u.%1u    ",data[0],data[1],data[2]);
//        lcd_gotoxy(1,2);
//        printf(LCD_PUTC,"BMI:%2u.%1u %2u/%2u/%2u
"data[3],data[4],data[5],data[6],data[7]);
//        printf(LCD_PUTC,"BMI: %2u.%1u         "data[3],data[4]);
```

```
        }
        else
        {
          count_ee--;
//              lcd_gotoxy(1,1);
//              printf(LCD_PUTC, "\f BMI NOT FOUND! ");
        }
      }
    }
    if (!input(down))
    {
     delay_ms(150);
     if (!input(down))
     {
       count_ee--;
       if(count_ee==255)
       {
         count_ee = 0;
       }
       ee_show = (count_ee*8)+1;
       for (m3=0;m3<8;m3++)
       {
         data[m3]=read_eeprom(ee_show);/////////date
         delay_ms(1);
         ee_show++;
       }

       lcd_gotoxy(1,1);
       printf(LCD_PUTC, "W:%2u.%1u  BMI:%2u.%1u ",data[1],data[2],data[3],data[4]);
       lcd_gotoxy(1,2);
       printf(LCD_PUTC,"#:%2u   %2u/%2u/%2u"data[0],data[5],data[6],data[7]);
```

```
//      lcd_gotoxy(1,1);
//      printf(LCD_PUTC, "#: %2u W: %2u.%1u    ",data[0],data[1],data[2]);
//      lcd_gotoxy(1,2);
//      printf(LCD_PUTC,"BMI:%2u.%1u %2u/%2u/%2u
"data[3],data[4],data[5],data[6],data[7]);
//      printf(LCD_PUTC,"BMI: %2u.%1u        "data[3],data[4]);
//          }
    }
   }
   if (!input(setup))
   {
    delay_ms(100);
    if (!input(setup))
    {
      lcd_gotoxy(1,1);
      lcd_putc("\fDELETE DATA BMI");
      lcd_gotoxy(1,2);
      lcd_putc("SW2:NO   SW3:YES");
//          delay_ms(100);

loop_sub_ee:  if (!input(down))
      {
        delay_ms(150);
        if (!input(down))
        {
//              lcd_gotoxy(1,2);
//              lcd_putc("DON'T CLEAR DATA");
//              delay_ms(1000);
          goto end_read;
        }
      }
      if (!input(up))
```

```
        {
          delay_ms(150);
          if (!input(up))
          {
//                lcd_gotoxy(1,2);
//                lcd_putc("CLEAR DATA.....");
//                delay_ms(1000);
            write_eeprom (addr_count_loop, 0);////
            sector_ee = 1;
            count_loop = 0;
            goto end_read;
          }
        }
        goto loop_sub_ee;
      }
    }
    goto read_ee;
end_read: delay_ms(5);
}

void main()
{
int8 p_temp=0;
//int8 m3;
//int16 ee_show=0x01;
int8 data[8];//,count_ee=0;
int8 day, mount,d_lcd;
int8 year;
float temp_bmi,bmi_old;

  set_tris_c(0b10010000);   // C3 = CLK out, C4 = SDI, C6=XMIT, C7=RCV
//  set_tris_d(0b00000000);   // D0 = CS
```

```
  set_tris_a(0x00);

  CS=1;   //Start with CS high
//  setup_spi(SPI_MASTER | SPI_MODE_0_0 | SPI_CLK_DIV_64 );
  setup_spi(SPI_MASTER | SPI_MODE_1_1 | SPI_CLK_DIV_64 );
  setup_adc_ports(NO_ANALOGS);
  setup_adc(ADC_OFF);
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
  setup_timer_1(T1_DISABLED);
  setup_timer_2(T2_DISABLED,0,1);

  port_b_pullups(true);

  output_high(power_chip);

  lcd_init();
  lcd_gotoxy(1,1);
  lcd_putc("  DIGITAL BMI  ");
  lcd_gotoxy(1,2);
  lcd_putc("PLEASE WAIT.....");
  delay_ms(1000);

  kbd_init();
//  reset_kg();
//  comp2 = sub_kg;

  if (p==12)
  {
    lcd_gotoxy(1,1);
    printf(LCD_PUTC, "SET HEIGHT:    ");
    lcd_gotoxy(1,2);
    printf(LCD_PUTC,"*: CANCEL  #: OK");
```

```
  }

  count_loop = read_eeprom (addr_count_loop);
//  sector_ee = (count_loop*5)+1;
  sector_ee = (count_loop*8)+1;

  flag_full_eeprom = read_eeprom(addr_full_eeprom);

 while(TRUE)
 {
//  goto save;
////////////////////////////////////////////////////////
  k=kbd_getc();

  if(!input(setup))
  {
    delay_ms(100);
    if(!input(setup))
    {
      read_data_eeprom();
    }

    goto end_save;
  }

      lcd_gotoxy(p,1);
      printf(LCD_PUTC, "%c",k);
      temp_hight[p_temp] = k-48;
      p_temp++;
    }
    if(k=='*')
    {
```

```
        lcd_gotoxy(1,1);
        printf(LCD_PUTC, "SET HEIGHT:     ");
        p=12;
        p_temp=0;
      }
      if(k=='#')
      {
        reset_kg();
        bmi=0;
        bmi_old = bmi;
        kilo_f=0;
        temp_bmi=0;
    r_save:  read_kg();
        value_hight = ((temp_hight[0]*100)+(temp_hight[1]*10)+(temp_hight[2]));
        temp_bmi = value_hight;
        temp_bmi /= 100;
        temp_bmi = pwr(temp_bmi,2);

        bmi = kilo_f / temp_bmi;

        lcd_gotoxy(1,1);
        printf(LCD_PUTC, "H: %3u W: %2.1f    ",value_hight,kilo_f);
        lcd_gotoxy(1,2);
        printf(LCD_PUTC,"BMI: %2.1f      ",bmi);
        p=30;
        delay_ms(150);
        read_kg();

        bmi_check++;

        if ((bmi_old!=bmi)||(bmi<1))
        {
```

```
            bmi_check=0;
            bmi_old = bmi;
            bmi_int = bmi;
            goto r_save;
          }
          if (bmi_check<5)
          {
            goto r_save;
          }
          d_lcd = 8;
          p_temp = 0;
          lcd_gotoxy(1,1);
          printf(LCD_PUTC,"YOUR BMI: %2.1f  ",bmi);
          lcd_gotoxy(1,2);
          printf(LCD_PUTC,"*: CANCEL  #: OK");
save:     k=kbd_getc();
          if(k=='*')
          {
            lcd_gotoxy(1,1);
            printf(LCD_PUTC, "NON SAVE.....");
            lcd_gotoxy(1,2);
            printf(LCD_PUTC,"                ");
            delay_ms(1500);
            goto end_save;
          }
          if(k=='#')
          {
            if (count_loop < 50)
            {
              lcd_gotoxy(1,1);
              lcd_putc("D/M/Y: --/--/-- ");
get_day:      k=kbd_getc();
```

```
            if(k=='*')
            {
              lcd_gotoxy(1,1);
              lcd_putc("D/M/Y: --/--/-- ");
              d_lcd = 8;
              p_temp = 0;
            }
            else if(k=='#')
            {
              goto end_set_day;
            }
            else if (k!=0)
            {
              lcd_gotoxy(d_lcd,1);
              printf(LCD_PUTC, "%c",k);
              data[p_temp] = k-48;
              d_lcd++;
              p_temp++;
              if ((d_lcd==10)||(d_lcd==13))
              {
                d_lcd++;
              }
            }
            goto get_day;
end_set_day:  day = (data[0]*10)+data[1];
            mount = (data[2]*10)+data[3];
            year = (data[4]*10)+data[5];
//            if (count_loop)
//            {
//              count_loop
//            }
//            lcd_gotoxy(1,2);
```

```
//            printf(LCD_PUTC, "Save Done..  #%2u",count_loop+1);
              write_data_eeprom(day,mount,year);
            }
            else
            {
              lcd_gotoxy(1,1);
              printf(LCD_PUTC, "  MEMORY FULL!  ");
              lcd_gotoxy(1,1);
              printf(LCD_PUTC, "DON'T SAVE..    ");
            }
            delay_ms(1500);
            goto end_save;
          }
        goto save;
end_save: p_temp=0;
        bmi=0;
        bmi_old = bmi;
        value_save=0;
        kilo = 0;
        kilo_f=0;
        temp_bmi=0;
        p=12;
        lcd_gotoxy(1,1);
        printf(LCD_PUTC, "SET HEIGHT:     ");
        lcd_gotoxy(1,2);
        printf(LCD_PUTC,"*: CANCEL  #: OK");
      }
    }
   }
}
```

ภาคผนวก ข

โปรแกรมแสดงผลจอแอลซีดี

```
// flex_lcd.c

// These pins are for the Microchip PicDem2-Plus board,
// which is what I used to test the driver.  Change these
// pins to fit your own board.
/**
#define LCD_DB4   PIN_B4
#define LCD_DB5   PIN_B5
#define LCD_DB6   PIN_B6
#define LCD_DB7   PIN_B7

#define LCD_E     PIN_B2
#define LCD_RS    PIN_B0
#define LCD_RW    PIN_B1
*/
#define LCD_DB4   PIN_D4
#define LCD_DB5   PIN_D5
#define LCD_DB6   PIN_D6
#define LCD_DB7   PIN_D7

#define LCD_E     PIN_D2
#define LCD_RS    PIN_D0
#define LCD_RW    PIN_D1

// If you only want a 6-pin interface to your LCD, then
// connect the R/W pin on the LCD to ground, and comment
// out the following line.

#define USE_LCD_RW  1

//=============================================
```

```
#define lcd_type 2        // 0=5x7, 1=5x10, 2=2 lines
#define lcd_line_two 0x40 // LCD RAM address for the 2nd line


int8const LCD_INIT_STRING[4] =
{
 0x20 | (lcd_type<< 2), // Func set: 4-bit, 2 lines, 5x8 dots
 0xc,                  // Display on
 1,                    // Clear display
 6                     // Increment cursor
};


//-----------------------------------
voidlcd_send_nibble(int8 nibble)
{
// Note:  !!converts an integer expression
// to a boolean (1 or 0).
output_bit(LCD_DB4, !!(nibble& 1));
output_bit(LCD_DB5, !!(nibble& 2));
output_bit(LCD_DB6, !!(nibble& 4));
output_bit(LCD_DB7, !!(nibble& 8));

delay_cycles(1);
output_high(LCD_E);
delay_us(2);
output_low(LCD_E);
}

//-----------------------------------
// This sub-routine is only called by lcd_read_byte().
// It's not a stand-alone routine.  For example, the
```

```
// R/W signal is set high by lcd_read_byte() before
// this routine is called.

#ifdef USE_LCD_RW
-----------------------------------------------------------------------
int8 lcd_read_nibble(void)
{
int8retval;
// Create bit variables so that we can easily set
// individual bits in the retval variable.
#bit retval_0 = retval.0
-----------------------------------------------------------------------
#bit retval_1 = retval.1
#bit retval_2 = retval.2
#bit retval_3 = retval.3

retval = 0;

output_high(LCD_E);
delay_cycles(1);

retval_0 = input(LCD_DB4);
retval_1 = input(LCD_DB5);
retval_2 = input(LCD_DB6);
retval_3 = input(LCD_DB7);

output_low(LCD_E);

return(retval);
}
#endif

//-------------------------------------
// Read a byte from the LCD and return it.
```

```
#ifdef USE_LCD_RW
int8 lcd_read_byte(void)
{
int8 low;
int8 high;

output_high(LCD_RW);
delay_cycles(1);

high = lcd_read_nibble();

low = lcd_read_nibble();

return( (high<<4) | low);
}
#endif

//-----------------------------------
// Send a byte to the LCD.
voidlcd_send_byte(int8 address, int8 n)
{
output_low(LCD_RS);

#ifdef USE_LCD_RW
while(bit_test(lcd_read_byte(),7)) ;
#else
delay_us(60);
#endif

if(address)
output_high(LCD_RS);
```

```
else
output_low(LCD_RS);

delay_cycles(1);

#ifdef USE_LCD_RW
output_low(LCD_RW);
delay_cycles(1);
#endif

output_low(LCD_E);

lcd_send_nibble(n >> 4);
lcd_send_nibble(n & 0xf);
}

//-----------------------------
voidlcd_init(void)
{
int8 i;

output_low(LCD_RS);

#ifdef USE_LCD_RW
output_low(LCD_RW);
#endif

output_low(LCD_E);

delay_ms(15);

for(i=0 ;i < 3; i++)
```

```
  {
lcd_send_nibble(0x03);
delay_ms(5);
  }
```

```
lcd_send_nibble(0x02);

for(i=0; i <sizeof(LCD_INIT_STRING); i++)
  {
lcd_send_byte(0, LCD_INIT_STRING[i]);
```

```
   // If the R/W signal is not used, then
   // the busy bit can't be polled.  One of
   // the init commands takes longer than
   // the hard-coded delay of 60 us, so in
   // that case, lets just do a 5 ms delay
   // after all four of them.
   #ifndef USE_LCD_RW
delay_ms(5);
   #endif
  }

}

//-----------------------------

voidlcd_gotoxy(int8 x, int8 y)
{
int8 address;

if(y != 1)
address = lcd_line_two;
```

```
else
address=0;

address += x-1;
lcd_send_byte(0, 0x80 | address);
}

//-----------------------------
voidlcd_putc(char c)
{
switch(c)
  {
case '\f':
lcd_send_byte(0,1);
delay_ms(2);
break;

case '\n':
lcd_gotoxy(1,2);
break;

case '\b':
lcd_send_byte(0,0x10);
break;

default:
lcd_send_byte(1,c);
break;
  }
}

//-----------------------------
```

```
#ifdef USE_LCD_RW
charlcd_getc(int8 x, int8 y)
{
char value;

lcd_gotoxy(x,y);

// Wait until busy flag is low.
while(bit_test(lcd_read_byte(),7));

output_high(LCD_RS);
value = lcd_read_byte();
output_low(lcd_RS);

return(value);
}
#endif
```

ภาคผนวก ค

โปรแกรมการรับค่าจากสวิตช์เมตริกซ์

```
#define use_portb_kbd TRUE
#if defined(__PCH__)
#if defined use_portb_kbd
  #byte kbd = 0xF81              // This puts the entire structure
#else
  #byte kbd = 0xF83              // This puts the entire structure
#endif
#else
#if defined use_portb_kbd
  #byte kbd = 6                  // on to port B (at address 6)
#else
  #byte kbd = 8                  // on to port D (at address 8)
#endif
#endif

#if defined use_portb_kbd
  #define set_tris_kbd(x) set_tris_b(x)
#else
  #define set_tris_kbd(x) set_tris_d(x)
#endif

//Keypad connection:  (for example column 0 is B2)
        //Bx:

#ifdefblue_keypad ////////////////////////////////// For the blue keypad
#define COL0 (1 << 2)
#define COL1 (1 << 3)
#define COL2 (1 << 6)

#define ROW0 (1 << 4)
#define ROW1 (1 << 7)
#define ROW2 (1 << 1)
```

```
#define ROW3 (1 << 5)

#else ////////////////////////////////////////////// For the black keypad
#define COL0 (1 << 5)
#define COL1 (1 << 6)
#define COL2 (1 << 7)

#define ROW0 (1 << 1)
#define ROW1 (1 << 2)
#define ROW2 (1 << 3)
#define ROW3 (1 << 4)

#endif

#define ALL_ROWS (ROW0|ROW1|ROW2|ROW3)
#define ALL_PINS (ALL_ROWS|COL0|COL1|COL2)

// Keypad layout:
charconst KEYS[4][3] = {{'1','2','3'},
                        {'4','5','6'},
                        {'7','8','9'},
                        {'*','0','#'}};

#define KBD_DEBOUNCE_FACTOR 33    // Set this number to apx n/333 where
                                  // n is the number of times you expect
                                  // to call kbd_getc each second


voidkbd_init() {
}

charkbd_getc( ) {
```

```
static BYTE kbd_call_count;
static int1 kbd_down;
static char last_key;
static BYTE col;
```

```
  BYTE kchar;
  BYTE row;

kchar='\0';
if(++kbd_call_count>KBD_DEBOUNCE_FACTOR) {
```

```
switch (col) {
case 0   : set_tris_kbd(ALL_PINS&~COL0);
kbd=~COL0&ALL_PINS;
break;
case 1   : set_tris_kbd(ALL_PINS&~COL1);
kbd=~COL1&ALL_PINS;
break;
case 2   : set_tris_kbd(ALL_PINS&~COL2);
kbd=~COL2&ALL_PINS;
break;
    }

if(kbd_down) {
if((kbd& (ALL_ROWS))==(ALL_ROWS)) {
kbd_down=FALSE;
kchar=last_key;
last_key='\0';
      }
    } else {
if((kbd& (ALL_ROWS))!=(ALL_ROWS)) {
if((kbd& ROW0)==0)
row=0;
```

```
else if((kbd& ROW1)==0)
row=1;
else if((kbd& ROW2)==0)
row=2;
else if((kbd& ROW3)==0)
row=3;
last_key =KEYS[row][col];
kbd_down = TRUE;
//////beep////////////////////////////////////////
//            output_high(bk);
//            bk_lcd_off=0;
//            output_high(buzzer);
//            delay_ms(200);
//            output_low(buzzer);
////////////////////////////////////////////////
        } else {
          ++col;
if(col==3)
col=0;
        }
      }
kbd_call_count=0;
  }
set_tris_kbd(ALL_PINS);
return(kchar);
}
```

ประวัติผู้ดำเนินโครงงาน

ชื่อ นายภูวเดช สุขดิษฐ์
ภูมิลำเนา 258 หมู่ 3 ต.แม่พูล อ.ลับแล
จ.อุตรดิตถ์ 53130

ประวัติการศึกษา

-จบมัธยมศึกษาจากโรงเรียนอุตรดิตถ์

-ปัจจุบันกำลังศึกษาระดับปริญญาตรีชั้นปีที่ 4
สาขาวิศวกรรมไฟฟ้าคณะวิศวกรรมศาสตร์
มหาวิทยาลัยนเรศวร

E-mail: patipans53@gmail.com

ชื่อ นายสวนัท อุดมรักษ์
ภูมิลำเนา 33/8 หมู่ 3 ต.เจ้าปลุก อ.มหาราช
จ.พระนครศรีอยุธยา13150

ประวัติการศึกษา

-จบมัธยมศึกษาจากโรงเรียนอ่างทองปัทมโรจน์วิทยาคม

-ปัจจุบันกำลังศึกษาระดับปริญญาตรีชั้นปีที่ 4
สาขาวิศวกรรมไฟฟ้าคณะวิศวกรรมศาสตร์
มหาวิทยาลัยนเรศวร

E-mail: sawanat_kiku@hotmail.com