เครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย

ACTIVITY BODY MOVEMENT RECORDER

โดย

นายฐิติพงศ์ ฟุ่นเต่ย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2557

เครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย

ACTIVITY BODY MOVEMENT RECORDER

โดย

นายฐิติพงศ์ ฟุ่นเต่ย 54010361

อาจารย์ที่ปรึกษา

รศ.ดร. ปราโมทย์ วาดเขียน

รศ.ดร. จีรสุดา โกษียาภรณ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2557

ปริญญานิพนธ์ปีการศึกษา 2557

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย

ACTIVITY BODY MOVEMENT RECORDER

ผู้จัดทำ

นายฐิติพงศ์ ฟุ่นเต่ย 54010361

..... อาจารย์ที่ปรึกษา

(รศ.ดร. ปราโมทย์ วาดเขียน)

..... อาจารย์ที่ปรึกษาร่วม

(รศ.ดร. จีรสุดา โกษียาภรณ์)

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้สามารถสำเร็จลุล่วงไปได้ด้วยดี เนื่องจากได้รับความอนุเคราะห์อย่างดียิ่งจากท่านอาจารย์ที่ปรึกษา รศ.ดร. ปราโมทย์ วาดเขียน และ รศ.ดร. จีรสุดา โกษียาภรณ์ ที่ให้คำปรึกษา คำแนะนำ ตลอดระยะเวลาในการทำปริญญานิพนธ์

ขอขอบพระคุณคณาจารย์ ภาควิชาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกท่าน ที่ได้อบรมสั่งสอนและประสิทธิ์ประสาทวิชาความรู้ให้แก่ผู้จัดทำ

ขอบอบพระคุณผู้ที่มีส่วนเกี่ยวข้องทุกท่าน และบิดา มารดา ของผู้จัดทำที่คอยช่วยเหลือสนับสนุน และให้กำลังใจ ตลอดระยะเวลาในการทำปริญญานิพนธ์

นายฐิติพงศ์ ฟุ่นเต่ย
ผู้จัดทำ

เครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย

ACTIVITY BODY MOVEMENT RECORDER

โดย นายฐิติพงศ์ ฟุ่นเต่ย 54010361

อาจารย์ที่ปรึกษา รศ.ดร. ปราโมทย์ วาดเขียน

อาจารย์ที่ปรึกษาร่วม รศ.ดร. จีรสุดา โกษียาภรณ์

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการออกแบบและสร้างเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย ซึ่งประกอบไปด้วย การยืน การนั่ง การนอน และการเดิน พร้อมด้วยการออกแบบระบบคำนวณอัตราการใช้พลังงานของร่างกายจากพฤติกรรมดังกล่าวในชีวิตประจำวัน โดยใช้โมดูลเซ็นเซอร์ LSM9DS0 สำหรับอ่านค่าความเร่งเชิงเส้น 3 แกนเพื่อตรวจจับการเคลื่อนไหวของร่างกาย โดยข้อมูลกิจกรรมการเคลื่อนไหวของร่างกายในแต่ละวันจะถูกนำมาประมวลผลเพื่อคำนวณหาพลังงานที่ใช้ไปของร่างกายในแต่ละวัน เมื่อได้ค่าพลังงานดังกล่าวจะทำให้สามารถคำนวณพลังงานที่จะได้รับจากอาหารที่จะบริโภคในแต่ละวันเพื่อควบคุมไม่ให้บริโภคเกินปริมาณของพลังงานที่ร่างกายต้องการใช้

ABSTRACT

This project is to design and construct a recorder for motion activity of the body, composed of standing, sitting, sleeping and walking. In addition, a system for calculating the energy of the body from daily life activity is designed. The sensor module LSM9DS0 is employed for reading the 3-axis linear accelerometer to detect movement of the body. The recorded data of those activities will be later processed to calculate the daily power usage of the body. The obtained value of power usage of the body leads to calculate the energy content of the food. This recorder thus helps people to control their food consumption behaviors and to gain suitable energy from food in each day.

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญรูป

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

สารบัญตาราง

สารบัญตาราง(ต่อ)

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันนี้พฤติกรรมการบริโภคของมนุษย์เปลี่ยนแปลงไปจากเดิมอย่างมาก พลังงานที่ได้รับจากการบริโภคอาหารเข้าไปมากกว่าพลังงานที่ร่างกายต้องใช้ เนื่องจากอาหารส่วนมากในปัจจุบันเป็นอาหารจำพวก แป้ง ไขมัน และน้ำตาลมาก

ดังนั้นผู้จัดทำปริญญานิพนธ์จึงเกิดแนวคิดในการออกแบบและสร้างเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน พร้อมด้วยการออกแบบระบบคำนวณอัตราการใช้พลังงานของร่างกายจากพฤติกรรมการใช้ชีวิตประจำวัน ซึ่งเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย จะถูกใช้เพื่อตรวจจับการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน โดยเครื่องบันทึกกิจกรรมการเคลื่อนไหว จะใช้โมดูลเซ็นเซอร์ LSM9DS0 สำหรับอ่านค่าความเร่งเชิงเส้น 3 แกน จากนั้นข้อมูลจากโมดูลเซ็นเซอร์ LSM9DS0 และค่าของเวลาที่ได้จากฐานเวลา จะส่งไปยังไมโครคอนโทรลเลอร์เพื่อประมวลผลหาลักษณะการเคลื่อนไหวของร่างกาย ข้อมูลพฤติกรรมการเคลื่อนไหวของร่างกายในแต่ละวันจะถูกนำมาประมวลผล เพื่อคำนวณหาอัตราการใช้พลังงานของร่างกายต่อไป

1.2 วัตถุประสงค์

1) เพื่อสร้างและออกแบบเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกายซึ่งประกอบไปด้วย การยืน การนั่ง การนอน และการเดิน พร้อมด้วยการเก็บบันทึกข้อมูล

2) คำนวณอัตราการใช้พลังงานของร่างกายจากกิจกรรมที่เกิดขึ้น

1.3 ขอบเขตของปริญญานิพนธ์

1) บันทึกกิจกรรม การยืน การนั่ง การนอน และการเดิน

2) การบันทึกเก็บข้อมูลได้ 1 วัน

3) เขียนโปรแกรมคำนวณการใช้พลังงานจากกิจกรรมที่บันทึกในแต่ละวันในคอมพิวเตอร์

1.4 ผลลัพธ์สุดท้ายที่จะได้รับ

1) ได้เครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกายซึ่งประกอบไปด้วย การยืน การนั่ง การนอน และการเดิน

2) สามารถนำข้อมูลกิจกรรมการเคลื่อนไหวของร่างกาย มาคำนวณหาอัตราการใช้พลังของร่างกาย ในแต่ละวันได้

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 ลักษณะการเคลื่อนไหวเบื้องต้นของร่างกาย

การเคลื่อนไหวเบื้องต้นของร่างกายโดยทั่วไปมี 2 ลักษณะ คือการเคลื่อนไหวแบบไม่เคลื่อนที่และการเคลื่อนไหวแบบเคลื่อนที่

1. การเคลื่อนไหวแบบไม่เคลื่อนที่ เป็นการใช้ส่วนต่าง ๆ ของร่างกายเคลื่อนไหวโดยที่ร่างกายอยู่กับที่ เช่น การยืน การนั่ง การนอน การยกไหล่ขึ้นลง การกระพริบตา เป็นต้น

2. การเคลื่อนไหวแบบเคลื่อนที่ หมายถึงการเคลื่อนไหวที่ต้องเคลื่อนออกจากจุดยืนเดิมการเคลื่อนไหวแบบนี้ขึ้นอยู่กับการใช้เท้าเป็นสำคัญ โดยการก้าวเท้าออกไปในลักษณะต่างๆ กัน เช่น การเดิน การวิ่ง การกระโดดไปข้างหน้าและข้างหลัง เป็นต้น

สำหรับในปริญญานิพนธ์นี้ จะมีลักษณะการเคลื่อนไหวของร่างกายอยู่ 4 แบบ คือ การยืน การนั่ง การนอน และการเดิน

2.2 ไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ คือ อุปกรณ์ที่สามารถสร้างระบบควบคุมได้ โดยอุปกรณ์นี้มีขนาดเล็ก และเป็นอุปกรณ์ประเภทสารกึ่งตัวนำที่มีการรวมเอาฟังก์ชันการทำงานต่างๆ ไว้ในตัว ซึ่งมีลักษณะคล้ายกับคอมพิวเตอร์ ซึ่งในที่นี้หมายถึงอุปกรณ์ภายในที่ประกอบด้วย หน่วยประมวลผลกลาง และพอร์ตในการเชื่อมต่อแบบต่างๆ

2.2.1 ส่วนประกอบทั่วไปของไมโครคอนโทรลเลอร์

1) หน่วยประมวลผลกลาง (Control Processing Unit)

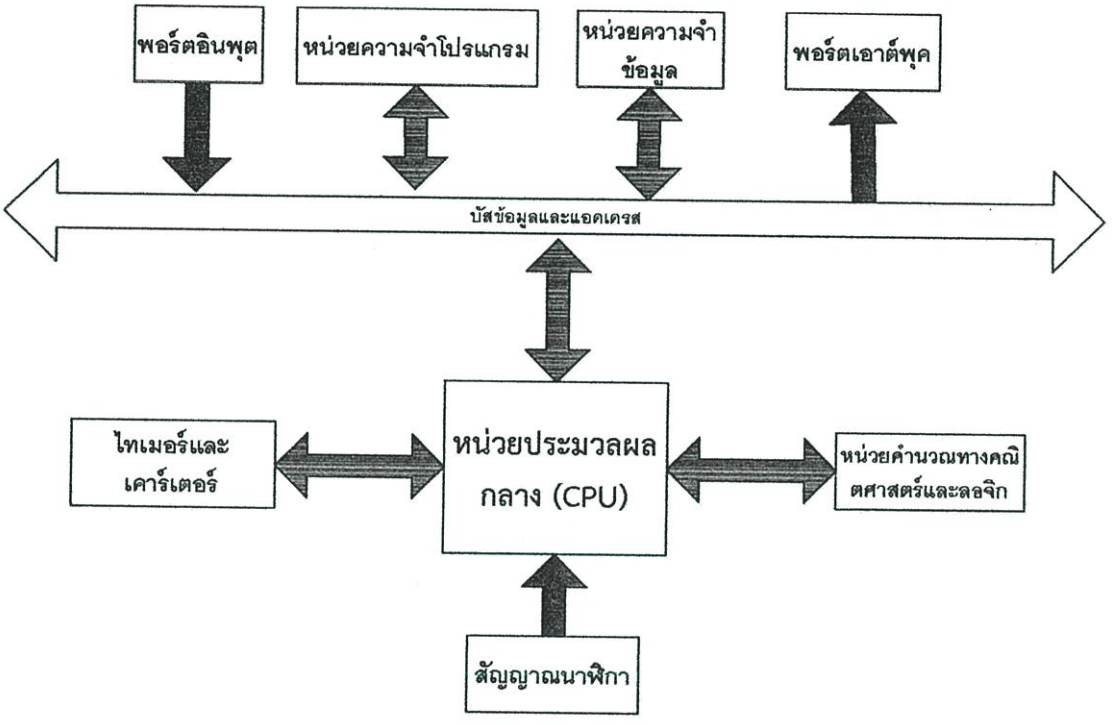
2) หน่วยความจำ ซึ่งประกอบด้วย RAM (Random Access Memory) และ EEPROM /EPROM/PROM/ROM (Erasable Programmable Read Only Memory)

3) หน่วยรับ และแสดงผลข้อมูล (Input/Output) ซึ่งมีพอร์ตขยายแบบขนาน (Parallel) และอนุกรม (Serial)

4) ตัวนับเวลา (Timer)

5) หน่วยควบคุมการอินเตอร์รัป (Interrupt Controller)

ส่วนประกอบเหล่านี้เป็นเพียงส่วนประกอบพื้นฐานของไมโครคอนโทรลเลอร์ สามารถแสดงดังรูปที่ 2.1 ไมโครคอนโทรลเลอร์นั้นยังมีส่วนประกอบอย่างอื่นอีกเพื่อเพิ่มเติมความสามารถขึ้นอยู่กับไมโครคอนโทรลเลอร์แต่ละเบอร์ด้วย เช่น A/D (Analog to Digital), PWM (Pulse Width Modulator) เป็นต้น

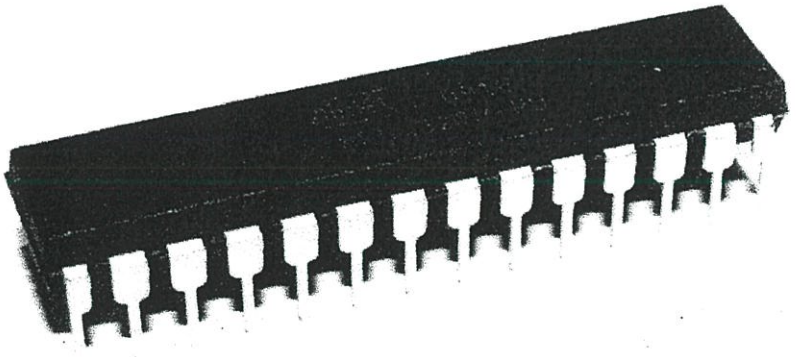


รูปที่ 2.1 โครงสร้างของไมโครคอนโทรลเลอร์

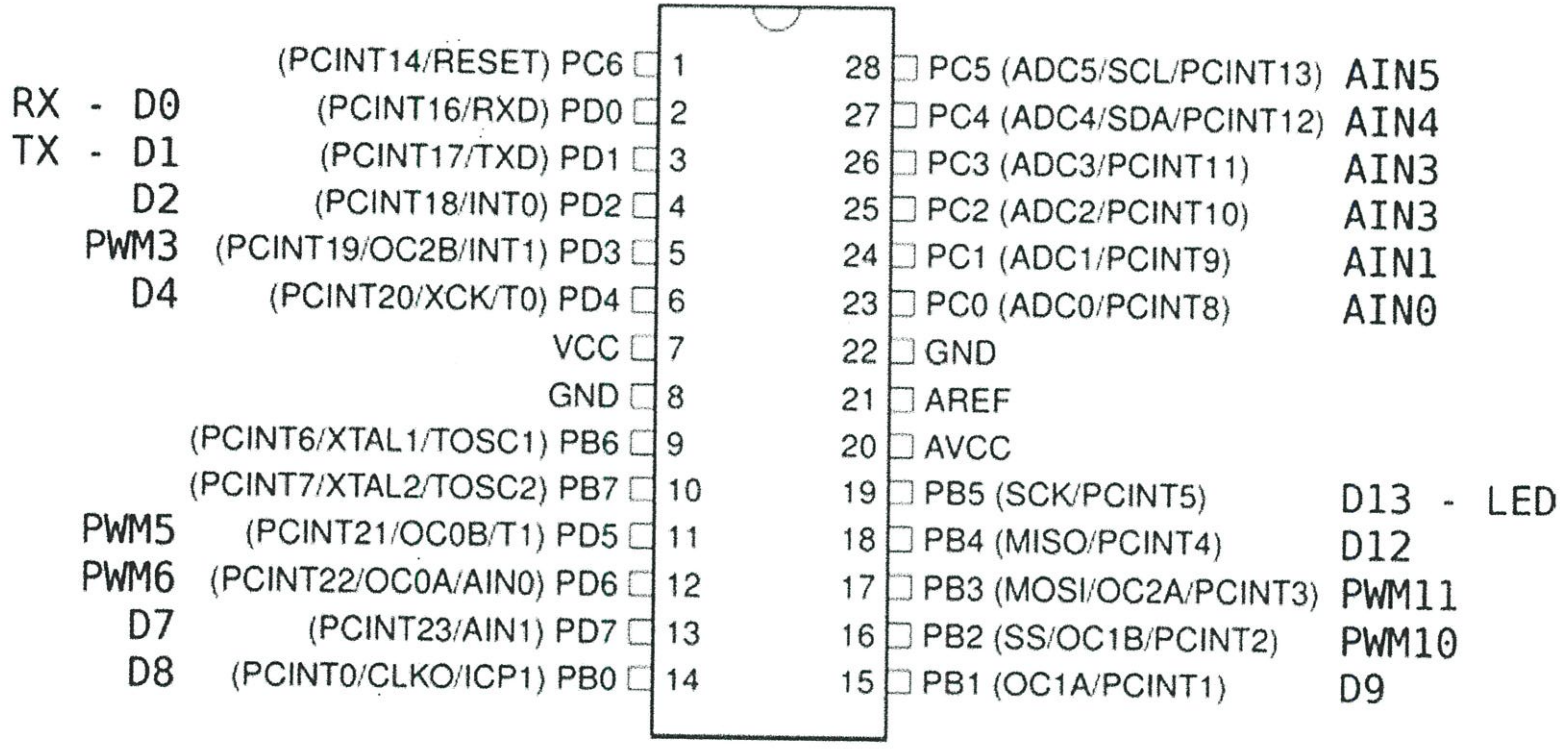
2.2.2 ไมโครคอนโทรลเลอร์ตระกูล เอวีอาร์ (AVR Microcontroller)

ผลิตโดยบริษัท ATMEL มีสถาปัตยกรรมแบบ RISC คือหนึ่งคำสั่งสามารถทำงานโดยใช้สัญญาณนาฬิกาเพียง 1 ลูก และประกอบด้วยหลากหลายขนาด สำหรับในปริญญานิพนธ์นี้จะใช้ไมโครคอนโทรลเลอร์เบอร์เอทีเมก้า 328 (ATMega168) แสดงดังรูปที่ 2.2 โดยส่วนสีดำนั้นประกอบจากพลาสติก (Plastic Dual in-line package: PDIP) ซึ่งเป็นสถาปัตยกรรมแบบ 8 บิต (8 bits Microcontroller) มีจำนวนขาทั้งหมด 28 ขาโดยมีการจัดเรียงตำแหน่งของขาต่างๆดังรูปที่ 2.3 มีรายละเอียดขาต่างๆ ตามตารางที่ 2.1 และไมโครคอนโทรลเลอร์เบอร์เอทีเมก้า 328 นั้นจะมีคุณสมบัติที่สำคัญต่างๆดังนี้

- หน่วยความจำโปรแกรมแบบแฟลช (Flash Memory) ขนาด 16 กิโลไบต์
- หน่วยความจำข้อมูลแบบเอสแรม (SRAM) ขนาด 1 กิโลไบต์
- หน่วยความจำข้อมูลแบบอีอีพรอม (EEPROM) ขนาด 512 ไบต์
- สนับสนุนการรับส่งข้อมูลแบบไอสแควร์ซี (I^2C:Inter Integrated Bus)
- อินพุตพอร์ต (Input Port) และเอาต์พุตพอร์ต (Output Port) จำนวน 20 พอร์ต
- ตัวแปลงสัญญาณอนาล็อกเป็นดิจิตอล (Analog to Digital Converter) ขนาด 10 บิต จำนวน 8 ช่อง
- ความถี่ใช้งานสูงสุด 20 เมกกะเฮิรตซ์
- ทำงานตั้งแต่แรงดัน 1.8-5.5 โวลต์
- มีวงจรสื่อสารอนุกรม
- มีพอร์ตเอาต์พุต จำนวน 23 พอร์ต



รูปที่ 2.2 ไมโครคอนโทรลเลอร์ตระกูลเอวีอาร์ เบอร์เอทีเมก้า 328 [1]



รูปที่ 2.3 การจัดวางขาของไมโครคอนโทรลเลอร์ เบอร์เอทีเมก้า 328 [2]

ตารางที่ 2.1 รายละเอียดขาต่างๆของไมโครคอนโทรลเลอร์เบอร์เอทีเมก้า 328

ชื่อ	ตำแหน่งขา
VCC : ขาแรงดันไฟตรง	ขา 7
GND : ขากราวด์	ขา 8
พอร์ตบี : ขาพอร์ตอินพุต-เอาต์พุตดิจิตอล • PB0 – PB5 • PB6 • PB7	 ขา 14 - 19 ขา 9 ขา 10
พอร์ตซี : ขาพอร์ตอินพุต-เอาต์พุตอะนาลอก • PC0 – PC5 • PC6	 ขา 23 - 28 ขา 1
พอร์ตดี : ขาพอร์ตอินพุต-เอาต์พุตดิจิตอล • PD0 – PD4 • PD5 – PD7	 ขา 2-6 ขา 11 - 13
AVCC : ขาแรงดันสำหรับพอร์ตเอและโมดูลแปลงสัญญาณอะนาลอกเป็นดิจิตอล	ขา 20
AREF : ขาแรงดันอะนาลอกอ้างอิงสำหรับโมดูลแปลงสัญญาณอะนาลอกเป็นดิจิตอล	ขา 21
AGND : ขากราวด์สำหรับพอร์ตเอ และโมดูลแปลงสัญญาณอะนาลอกเป็นดิจิตอล	ขา 22

โดยที่เอวีอาร์จะสามารถใช้เอวีอาร์สตูดิโอ(AVRStudio) หรือ อาร์ดูอิโน่ (Arduino) ในการพัฒนาตัวไมโครคอนโทรลเลอร์ได้ โดยในปริญญานิพนธ์นี้จะเลือกใช้ซอฟต์แวร์ของอาร์ดูอิโนในการพัฒนาตัวโปรแกรมของไมโครคอนโทรลเลอร์ได้ โดยในปริญญานิพนธ์นี้จะเลือกใช้ซอฟต์แวร์ที่ใช้งานง่าย และมีหมวดหมู่ของโปรแกรม (Program Library) เป็นจำนวนมาก เพราะว่าอาร์ดูอิโน่เป็นโครงการพัฒนาไมโครคอนโทรลเลอร์ของเอวีอาร์ซึ่งเป็นซอฟต์แวร์แบบที่สามารถให้ผู้ใช้พัฒนาได้ (Open Source) ที่ประกอบไปด้วยส่วนของฮาร์ดแวร์ที่ใช้ไมโครคอนโทรลเลอร์เอทีเมก้า 328 (Development Boad) และส่วนของการพัฒนาโปรแกรมที่ใช้สำหรับติดต่อกับไมโครคอนโทรลเลอร์โดยจะทำการติดต่อทางอาร์เอส 232 (RS 232)

2.3 การสื่อสารข้อมูล (Data communication)

การสื่อสารข้อมูลคือการส่งข้อมูลที่ทำการเข้ารหัสแล้วระหว่างอุปกรณ์ 2 ชนิดโดยมีองค์ประกอบหลักสำคัญอยู่ 3 ส่วน คือ อุปกรณ์ฝั่งส่ง อุปกรณ์ฝั่งรับ และตัวกลาง ซึ่งชนิดของตัวกลางที่นิยมใช้ในปัจจุบัน คือ ตัวกลางที่มีลักษณะเป็นสายเชื่อมต่อระหว่างผู้ส่งกับผู้รับ ซึ่งแบ่งออกเป็น 2 กลุ่มใหญ่ๆ คือ ตัวกลางประเภทสายส่งสัญญาณไฟฟ้า กับ ตัวกลางประเภทไร้สาย โดยสามารถแบ่งรูปแบบในการสื่อสารข้อมูลได้เป็น 3 แบบคือ การสื่อสารแบบซิมเพล็กซ์ (Simplex) หรือการสื่อสารแบบทางเดียว (One-way communication) การสื่อสารแบบฮาล์ฟดูเพล็กซ์ (Half Duplex) หรือการสื่อสารแบบทางใดทางหนึ่ง และการสื่อสารแบบสองทาง (Full Duplex)

- การสื่อสารแบบซิมเพล็กซ์ (Simplex) หรือการสื่อสารแบบทางเดียว (One-way communication) ประกอบด้วยช่องสัญญาณเพียงช่องเดียว และปลายทางด้านหนึ่งเป็นผู้รับ ตัวอย่างเช่น การกระจายเสียงของสถานีวิทยุต่าง ๆ การแพร่ภาพทางโทรทัศน์ การส่งน้ำตามท่อ หรือการจราจรระบบทางเดียว เป็นต้น สามารถแสดงดังรูปที่ 2.4

ผู้ส่ง >>>>> ช่องสัญญาณ >>>>> ผู้รับ

รูปที่ 2.4 รูปแบบการสื่อสารแบบทางเดียว

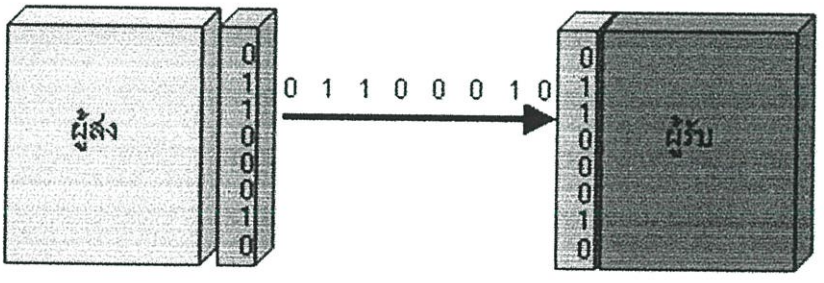
- การสื่อสารแบบฮาล์ฟดูเพล็กซ์ (Half Duplex) หรือการสื่อสารแบบทางใดทางหนึ่ง การส่งข้อมูลผ่านช่องสัญญาณเดียวนั้นจะสามารถส่งไปได้ทั้งสองทาง แต่ต้องสลับกัน จะส่งในเวลาเดียวกันไม่ได้ ตัวอย่างเช่น วิทยุสื่อสารในรถตำรวจ นั่นคือเมื่อผู้รับได้รับข้อมูลแล้ว ผู้รับจะใช้ระยะเวลาหนึ่งในการตีความ และทราบว่าข้อมูลจากผู้ส่งหมดแล้ว และพร้อมที่จะตอบกลับไปซึ่งช่วงเวลานี้เรียกว่า Reaction time และเมื่อผู้รับต้องการส่งข้อมูลตอบกลับไปจะมีการกดสวิตช์ ซึ่ง

ต้องใช้เวลาในการเปลี่ยนสภาวะจากผู้รับเป็นผู้ส่ง ช่วงเวลาที่กดสวิตช์นี้เรียกว่า Line turnaround time รวมกันเรียกว่า System turnaround time

- การสื่อสารสองทาง (Full duplex) เป็นการติดต่อสองทาง คือเป็นผู้รับข้อมูล และผู้ส่งข้อมูลในเวลาเดียวกันได้ ตัวอย่างการใช้งาน เช่น การติดต่อระหว่างเทอร์มินัลกับคอมพิวเตอร์แม่ บางชนิดที่ไม่ต้องใช้เวลารอสามารถโต้ตอบได้ทันทีหรือการพูดคุยทางโทรศัพท์ เป็นต้น

2.3.1 การสื่อสารข้อมูลแบบอนุกรม (Serial data transmission)

การสื่อสารข้อมูลแบบอนุกรม (Serial data transmission) เป็นการส่งข้อมูล ครั้งละ 1 บิต ไปบนสัญญาณจนครบจำนวนข้อมูลที่มีอยู่ สามารถนำไปใช้กับสื่อนำข้อมูลที่มีเพียง 1 ช่องสัญญาณได้ สื่อนำข้อมูลที่มี 1 ช่องสัญญาณนี้จะมีราคาถูกกว่าสื่อนำข้อมูลที่มีหลายช่องสัญญาณ และเนื่องจากการสื่อสารแบบอนุกรมมีการส่งข้อมูลได้ครั้งละ 1 บิตเท่านั้น การส่งข้อมูลประเภทนี้จึงช้ากว่าการส่งข้อมูลครั้งละหลายบิต มีการสื่อสารดังรูปที่ 2.5

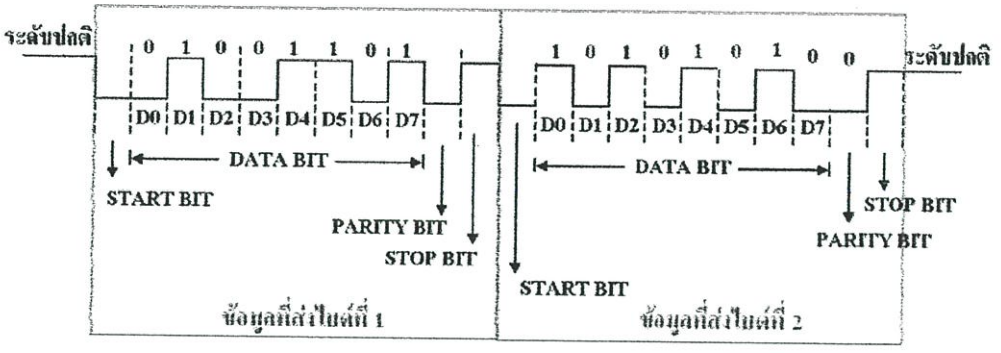


รูปที่ 2.5 โครงสร้างการสื่อสารข้อมูลแบบอนุกรม [3]

2.3.2 ประเภทของการสื่อสาร

ประเภทของการสื่อสารตามลักษณะสัญญาณในการสื่อสารได้ 2 แบบ คือ

1) การสื่อสารแบบซิงโครนัส (Synchronous) สำหรับการสื่อสารแบบซิงโครนัสนี้จะใช้สัญญาณนาฬิกาควบคุมการรับส่งสัญญาณ เช่น สายคีย์บอร์ดคอมพิวเตอร์ โดยจะมีสายสัญญาณเส้นหนึ่งเป็นสายสัญญาณนาฬิกา ส่วนอีก เส้นหนึ่งเป็นสายของข้อมูล (และมักจะมีสายกราวด์ด้วย) สำหรับการสื่อสารแบบซิงโครนัสนี้เหมาะสำหรับการทำงานในระยะใกล้ข้อมูลที่จะส่งมีไม่มากนัก เพราะถ้าระยะทางไกลขึ้นจะทำให้สัญญาณนาฬิกามีปัญหา อีกทั้งต้องมีสายหลายเส้นทำให้สิ้นเปลือง รูปแบบการส่งข้อมูลแบบซิงโครนัสจะเป็นการส่งครั้งละ 1 ไบต์ ดังรูปที่ 2.6



รูปที่ 2.6 การส่งข้อมูลแบบซิงโครนัส [4]

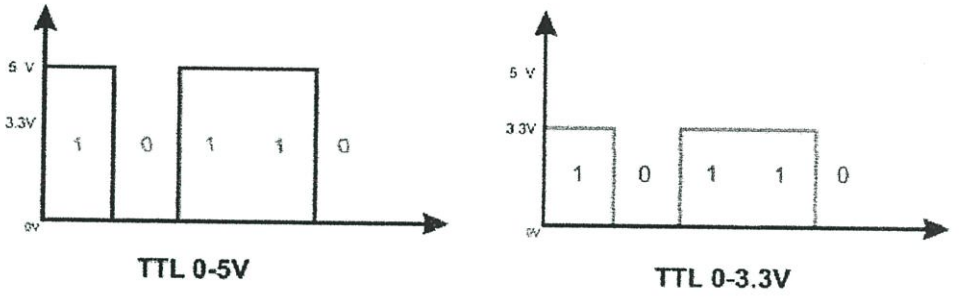
2) การสื่อสารแบบอะซิงโครนัส (Asynchronous) สำหรับการสื่อสารแบบอะซิงโครนัสนั้นจะใช้สายสัญญาณเพียงตัวเดียวแต่จะใช้รูปแบบการส่งข้อมูล หรือ Bit Pattern เป็นตัวกำหนดว่าส่วนไหนเป็นตัวเริ่มต้นข้อมูล ส่วนไหนเป็นตัวข้อมูล ส่วนไหนจะเป็นตัวตรวจสอบความถูกต้องของข้อมูล และส่วนไหนเป็นส่วนปิดท้ายของข้อมูล โดยต้องกำหนดให้สัญญาณนาฬิกาเท่ากันทั้งภาครับและภาคส่ง ซึ่งจะมีอุปกรณ์พิเศษที่เรียกว่า UART หรือ Universal Asynchronous Receiver/Transmitter คอยควบคุมการรับ และการส่งข้อมูล ในการสื่อสารแบบอนุกรมข้อมูลที่ใช้ในการสื่อสารจะอยู่ในลักษณะของกลุ่มบิตข้อมูล (Bit Stream) จำเป็นจะต้องสนใจเรื่องอัตราเร็วในการรับส่งข้อมูลโดยจะใช้การกำหนดอัตราของบอร์ด (Baud Rate) ในการกำหนดซึ่งจะมีค่ามาตรฐานได้แก่ 110 150 300 1200 2400 4800 9600 19200 เป็นต้น

2.3.3 มาตรฐานอาร์เอส 232

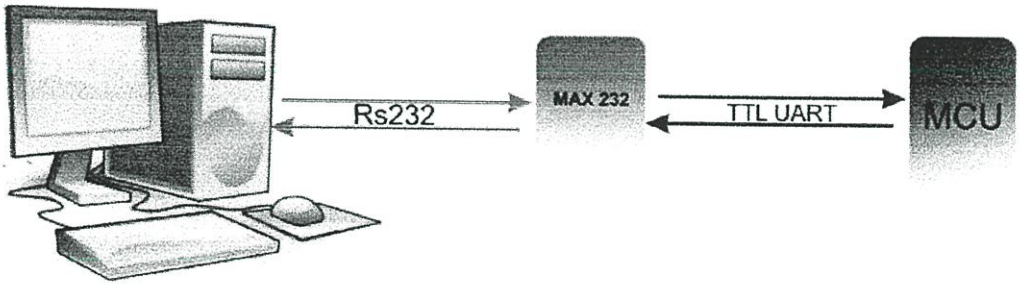
เป็นมาตรฐานการเชื่อมต่อข้อมูลแบบอนุกรมที่นิยมใช้มากที่สุด กำหนดโดย EIA (Electronics Industry Association) หรือสมาคมผู้ประกอบการอุตสาหกรรมอิเล็กทรอนิกส์ของอเมริกา ตั้งแต่ปี 1969 เริ่มต้นจากความต้องการที่จะกำหนดมาตรฐานการเชื่อมต่อระหว่างคอมพิวเตอร์กับโมเด็มในสมัยนั้น จุดประสงค์ของมาตรฐานนี้คือเพื่อบรรยายคุณลักษณะของการเชื่อมต่อ อุปกรณ์รับส่งข้อมูลปลายทาง (Data Terminal Equipment: DTE) กับ อุปกรณ์สื่อสารข้อมูล (Data Communication Equipment : DCE) ซึ่งจะขึ้นอยู่กับผู้ผลิต โดยที่ RS 232 นี้ ย่อมาจาก Recommend Standard 232 ซึ่งจะมีระดับแรงดันให้อยู่ในช่วง –15 ถึง 15 โวลต์ โดย ลอจิก “0” จะมีแรงดันอยู่ในช่วง 3 ถึง 15 โวลต์ และลอจิก “1” จะมีแรงดันในช่วง -15 ถึง -3 โวลต์ ซึ่งอาร์เอส 232 นั้นมีสายสัญญาณ 3 เส้นก็คือสายส่งข้อมูล (Tx) สายรับข้อมูล (Rx) และกราวด์ (GND)

2.3.4 ทีทีแอล (Transistor-Transistor Logic)

TTL เป็นระดับแรงดันที่ถูกกำหนดขึ้นในยุคแรกๆเพื่อใช้ระหว่าง ทรานซิสเตอร์ กับทรานซิสเตอร์ ภายในวงจรรวม (IC) ดังนั้น TTL จะใช้ระดับแรงดัน อยู่ที่ 0 ถึง 5 โวลต์ แต่ในปัจจุบันมีอุปกรณ์หลายเบอร์ที่ทำงานในช่วง 0 ถึง 3.3 โวลต์ (เรียกแรงดันระดับนี้ว่า LVTTL) ซึ่งผู้ใช้ควรตรวจสอบจาก Datasheet ของอุปกรณ์ที่ใช้เสียก่อนว่าเป็นระดับแรงดันแบบใด เพราะหากใช้ผิดประเภทจะทำให้อุปกรณ์เสียหาย แสดงระดับแรงดันดังรูปที่ 2.7 และแสดงการเปลี่ยนระดับแรงดัน TTL เป็น RS232 ในฝั่งส่ง และ เปลี่ยน RS232 เป็น TTL ในฝั่งรับ สามารถแสดงดังรูปที่ 2.8



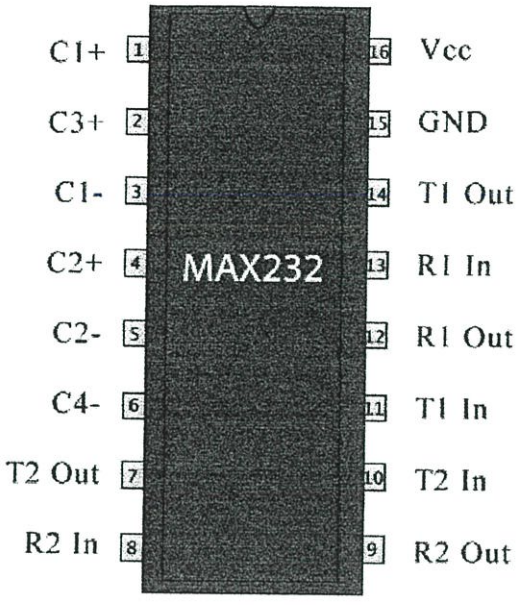
รูปที่ 2.7 กราฟแรงดันทีทีแอลโดยที่ 0 โวลต์แทนลอจิก “0” และ 5 โวลต์หรือ 3.3 โวลต์แทนลอจิก “1” [5]



รูปที่ 2.8 แสดงการเปลี่ยนระดับแรงดันระหว่าง TTL กับ RS232 [5]

2.3.5 การแปลงระดับแรงดัน

การที่จะทำการสื่อสารระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์นั้นจำเป็นจะต้องทำการแปลงระดับแรงดัน เนื่องจากข้อมูลที่ออกมาจากพอร์ตอนุกรมของคอมพิวเตอร์มีระดับแรงดัน ตามมาตรฐานอาร์เอส 232 แต่ข้อมูลที่สามารถสื่อสารกับไมโครคอนโทรลเลอร์ได้นั้น ต้องมีระดับแรงดันแบบทีทีแอล จะเห็นว่ามีระดับแรงดันที่ต่างกันจึงจำเป็นจะต้องทำการแปลงระดับแรงดัน โดยที่การแปลงระดับแรงดันที่ใช้ในปริญญานิพนธ์นี้จะอาศัยไอซีเบอร์แม็กซ์ 232 (IC MAX 232) ในการแปลงระดับแรงดันซึ่งแม็กซ์ 232 นั้นมีการจัดเรียงขาแสดงดังรูปที่ 2.9



รูปที่ 2.9 การจัดเรียงขาของไอซีแม็กซ์ 232 [6]

จากรูปที่ 2.9 จะเห็นได้ว่าขาของไอซีแม็กซ์ 232 ที่เกี่ยวข้องกับการสื่อสารข้อมูลจะได้แก่ขาที่ 7, 8, 9, 10, 11, 12,13, และ 14 โดยที่รายละเอียดต่างๆ ของขาดังกล่าวแสดงได้ตามตารางที่ 2.2

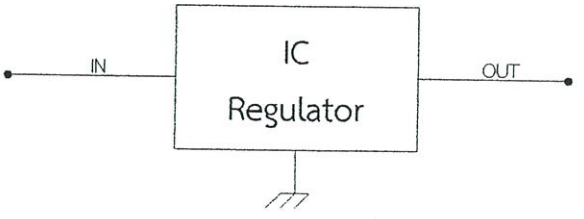
ตารางที่ 2.2 รายละเอียดของขาต่างๆที่เกี่ยวข้องกับการสื่อสารข้อมูล

หมายเลขของขา	ชื่อขา	รายละเอียดของขา	ระดับแรงดัน(โวลต์)
7	T2Out	ส่งออกสัญญาณอาร์เอส 232	-15 ถึง 15
8	R2In	รับเข้าสัญญาณอาร์เอส 232	-15 ถึง 15
9	R2Out	ส่งออกสัญญาณทีทีแอล	0 ถึง 3.3 หรือ 0 ถึง 5
10	T2In	รับเข้าสัญญาณทีทีแอล	0 ถึง 3.3 หรือ 0 ถึง 5
11	T1In	รับเข้าสัญญาณทีทีแอล	0 ถึง 3.3 หรือ 0 ถึง 5
12	R1Out	ส่งออกสัญญาณทีทีแอล	0 ถึง 3.3 หรือ 0 ถึง 5
13	R1In	รับเข้าสัญญาณอาร์เอส 232	-15 ถึง 15
14	T1Out	ส่งออกสัญญาณอาร์เอส 232	-15 ถึง 15

จากตารางที่ 2.2 จะเห็นว่าไอซีแม็กซ์ 232 สามารถแปลงระดับแรงดันได้ 2 ชันแนล และการที่จะแปลงสัญญาณนั้นจำเป็นจะต้องเลือกการใช้ขาของไอซีแม็กซ์ 232 ให้ตรงกัน คือ ถ้าเลือกขา 8 ในการรับเข้าสัญญาณอาร์เอส 232 สัญญาณทีทีแอลที่จะออกที่ขา 9 เป็นต้น

2.3.6 วงจรแปลงแรงดันกระแสตรง

ไอซีเร็กกูเลเตอร์ภายในประกอบด้วยวงจรเร็กกูเลเตอร์แบบอนุกรม มีขาต่อใช้งาน 3 ขา ประกอบด้วยขาอินพุต เอาต์พุต และกราวด์ ซึ่งจะจ่ายแรงดันค่าใดค่าหนึ่งโดยเฉพาะ ในปริญญานิพนธ์นี้ จะใช้ไอซีเบอร์ 7833 ใช้แปลงแรงดันกระแสตรงจาก 5 โวลต์ เป็น 3.3 โวลต์ แสดงดังรูปที่ 2.10



รูปที่ 2.10 โครงสร้างของไอซีเร็กกูเรเตอร์

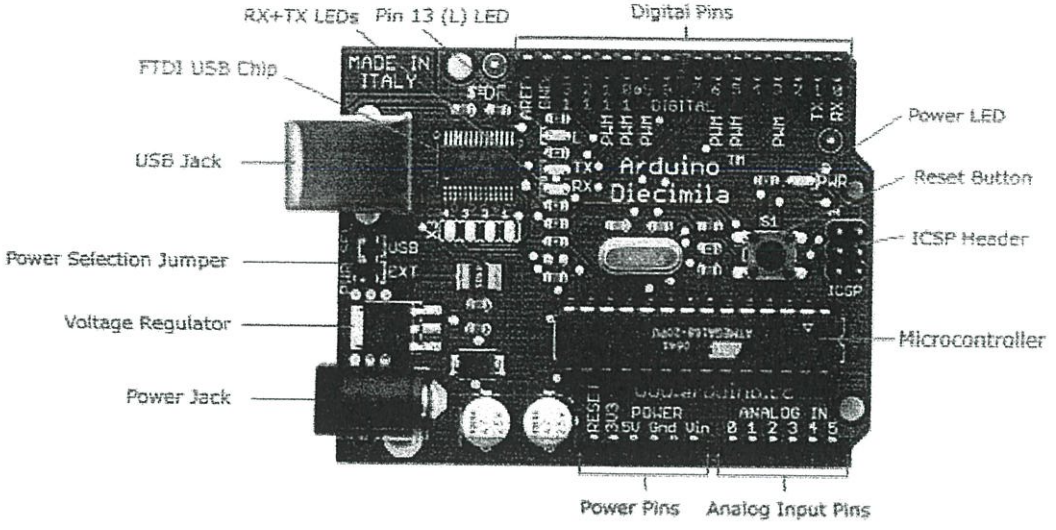
จุดเด่นของไอซีเร็กกูเลเตอร์ค่าคงที่คือ สามารถต่อวงจรได้ง่ายไม่ต้องต่ออุปกรณ์ภายนอกเพิ่มเติมมากนัก จากวงจร จะมีตัวเก็บประจุค่าหนึ่งต่อไว้ด้านอินพุต เพื่อป้องกันการเกิดออสซิเลตที่ความถี่สูง ซึ่งจะทำให้วงจรขาดเสถียรภาพ เอาต์พุตที่ออกจากไอซีเร็กกูเลเตอร์ จะได้แรงดันเอาต์พุตที่เรียบพอสมควรอยู่แล้ว แต่อาจจะใส่ตัวเก็บประจุเพื่อช่วยปรับปรุงแรงดันให้เรียบขึ้น

2.4 โปรแกรมเขียนคำสั่งสำหรับไมโครคอนโทรลเลอร์ตระกูลเอวีอาร์

การเขียนโปรแกรมให้ไมโครคอนโทรลเลอร์ตระกูลเอวีอาร์สามารถทำงานได้ตามที่ต้องการ สามารถใช้เอวีอาร์ สตูดีโอ (AVR Studio) ซึ่งเป็นซอฟต์แวร์สำหรับสร้างโปรแกรมเขียนคำสั่งสำหรับไมโครคอนโทรลเลอร์ที่ใช้ภาษาแอสแซมบลีในการเขียนโปรแกรม (AVR assembler) โดยที่มีวินเอวีอาร์ (WinAVR) เป็นตัวแปลโปรแกรมภาษาซี/ซีพลัสพลัส (C/C++ Compiler) หรือสามารถใช้อาร์ดูอิโน่ (Arduino) ซึ่งเป็นแพลตฟอร์มมาตรฐานเปิด (Open source Platform) โดยการเขียนโปรแกรมให้กับไมโครคอนโทรลเลอร์ตระกูลเออวีอาร์ในปริญญานิพนธ์นี้ จะใช้อาร์ดูอิโน่ เพราะง่ายต่อการเขียนโปรแกรม ถึงแม้จะเปลี่ยนรุ่นของไมโครคอนโทรลเลอร์เป็นรุ่นใดก็ไม่ต้องแก้ไขโปรแกรมใดๆ อีก

2.4.1 โปรแกรมอาร์ดูอิโน่

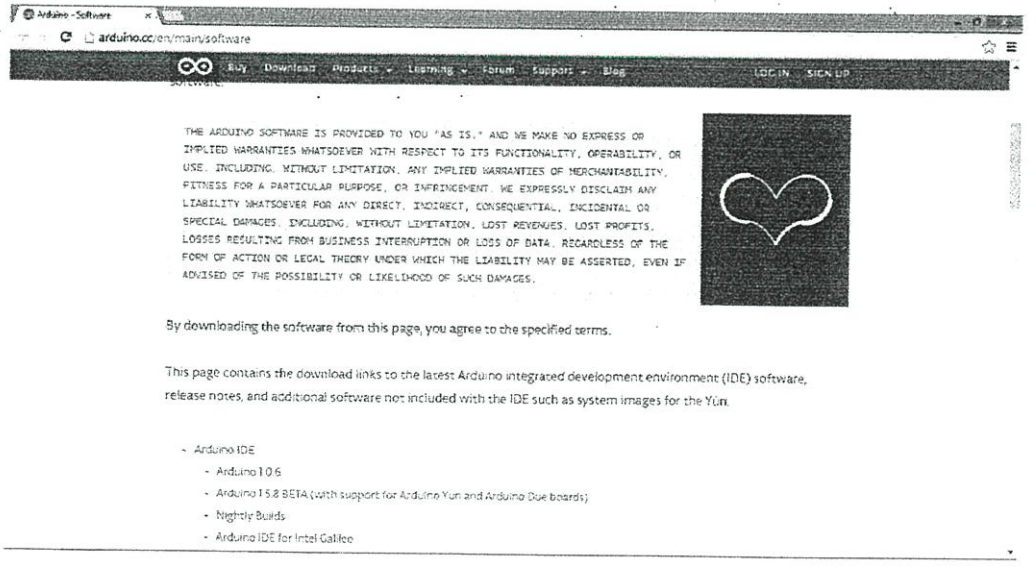
โปรแกรมอาร์ดูอิโน่มีจุดเด่นในเรื่องของความง่ายในการเรียนรู้และใช้งาน เนื่องจากมีการออกแบบคำสั่งต่างๆ ขึ้นมาสนับสนุนการใช้งานด้วยรูปแบบที่ไม่ซับซ้อน แต่สามารถนำไปใช้ในงานที่มีความซับซ้อนได้ สามารถสร้างคำสั่งและไลบรารี่ (Library) ใหม่ๆ ขึ้นมาเองได้ เมื่อผู้ใช้งานมีความชำนาญมากขึ้นแล้ว รองรับการทำงานทั้ง วินโดว์ (Windows), ลีนุกซ์ (Linux) และแมคอินทอช โอเอสเท็น (Macintosh OS X) และราคาไม่แพง เนื่องจากมีการเปิดเผยวงจรและคำสั่ง อีกทั้งยังมีบอร์ดสำเร็จรูปออกวางจำหน่าย ดังรูปที่ 2.11 ทำให้ผู้ใช้งานสามารถต่อวงจรขึ้นมาใช้งานได้เอง หรือนำไปพัฒนาต่อยอดได้ตามความต้องการ



รูปที่ 2.11 ตัวอย่างบอร์ดอาร์ดูอิโน่สำเร็จรูป [7]

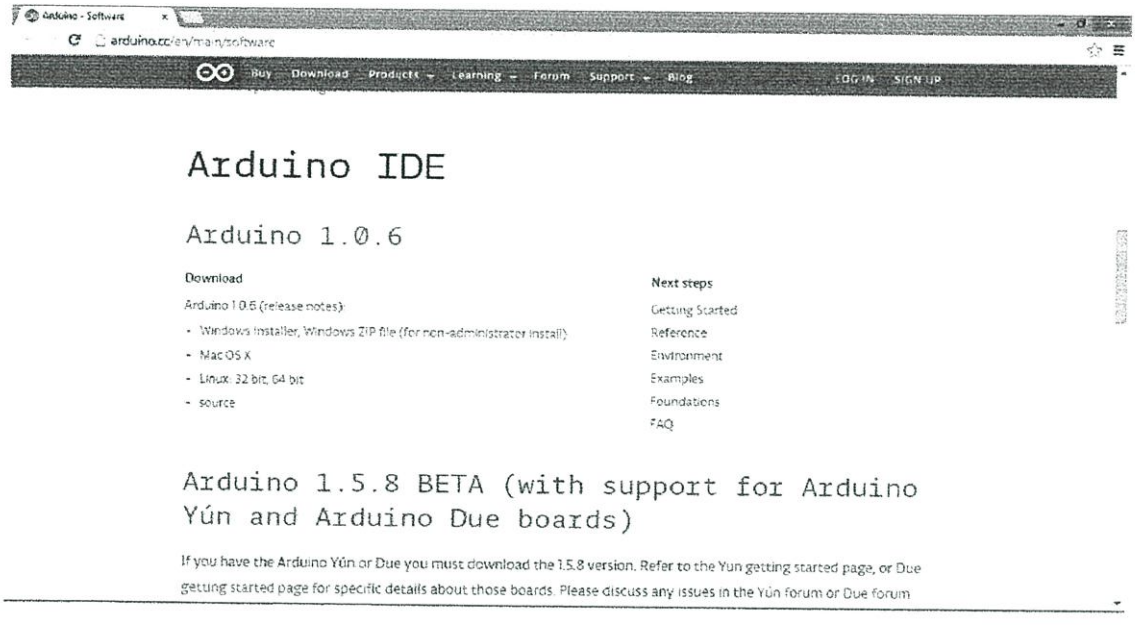
2.4.1.1 การติดตั้งโปรแกรมอาร์ดูอิโน่ ไมโครคอนโทรลเลอร์เป็นอุปกรณ์ที่ไม่มีโปรแกรมพื้นฐานของคอมพิวเตอร์ จึงจำเป็นต้องมีการติดตั้งโปรแกรมก่อน โดยมีขั้นตอนดังนี้

1) เข้าไปที่เว็บไซต์ของอาร์ดูอิโน่ http://arduino.cc/en/Main/Software โดยมีหน้าเว็บไซต์เป็นดังรูปที่ 2.12



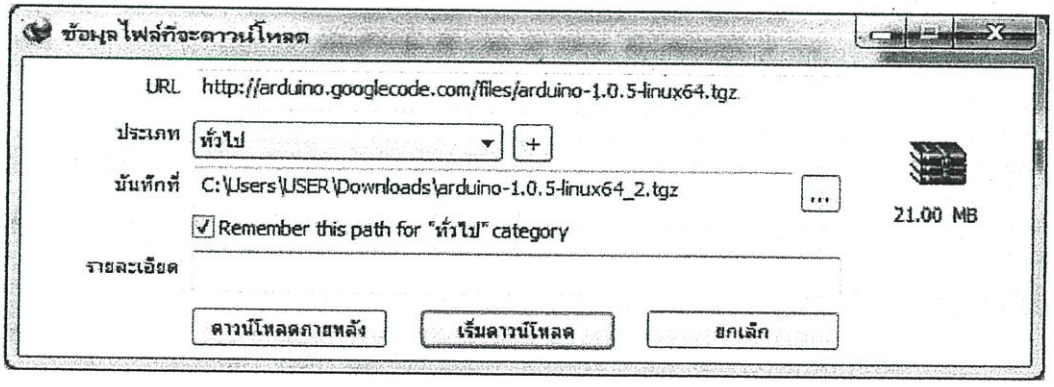
รูปที่ 2.12 หน้าเว็บไซต์อาร์ดูอิโน่

2) ไปที่หัวข้อ Download แล้วกดเลือกระบบปฏิบัติการของคอมพิวเตอร์ที่ต้องการติดตั้งโปรแกรมดังรูปที่ 2.13



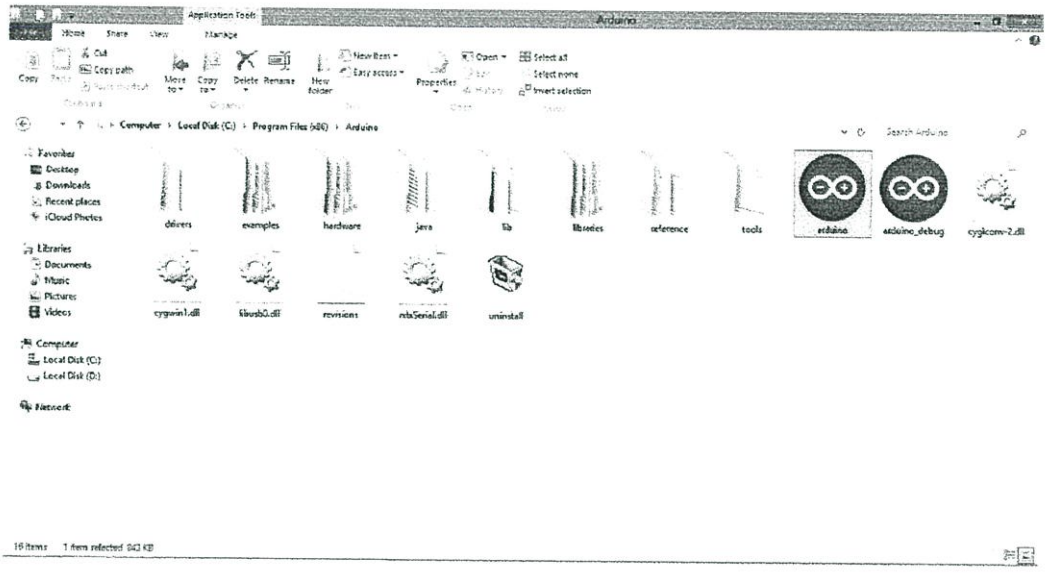
รูปที่ 2.13 ระบบปฏิบัติการของคอมพิวเตอร์ต่างๆ ที่สามารถติดตั้งโปรแกรมได้

เมื่อกดเลือกระบบปฏิบัติการของคอมพิวเตอร์ที่ต้องการจะติดตั้งโปรแกรมแล้วจะมีหน้าต่างขึ้นมา ซึ่งเป็นหน้าต่างดาวน์โหลดโปรแกรม ดังรูปที่ 2.14



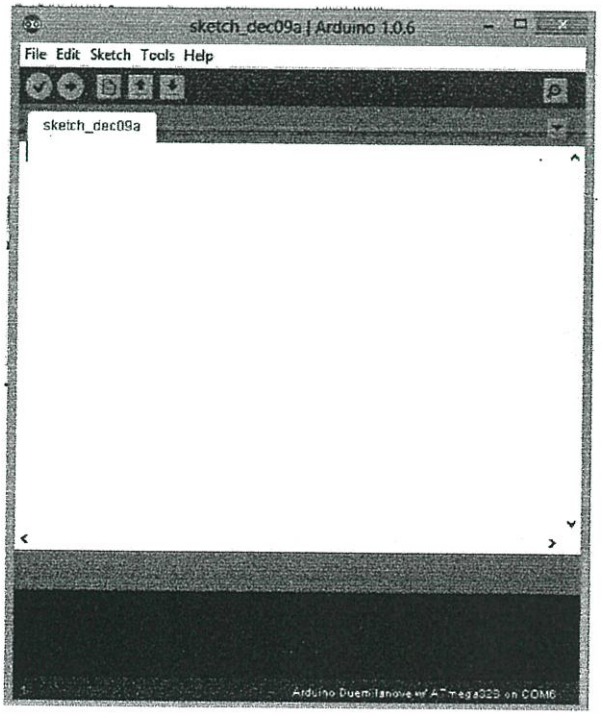
รูปที่ 2.14 หน้าต่างดาวน์โหลดโปรแกรม

3) เมื่อดาวน์โหลดเสร็จแล้วให้ทำการแตกไฟล์จะได้ดังรูปที่ 2.15



รูปที่ 2.15 ไฟล์ที่ได้หลังทำการแตกไฟล์เสร็จแล้ว

จากรูปที่ 2.15 ให้ทำการคลิกที่ไฟล์ที่มีกรอบล้อมรอบไว้ ซึ่งเป็นตัวโปรแกรมอาร์ดูอิโน่ จะได้ ดังรูปที่ 2.16

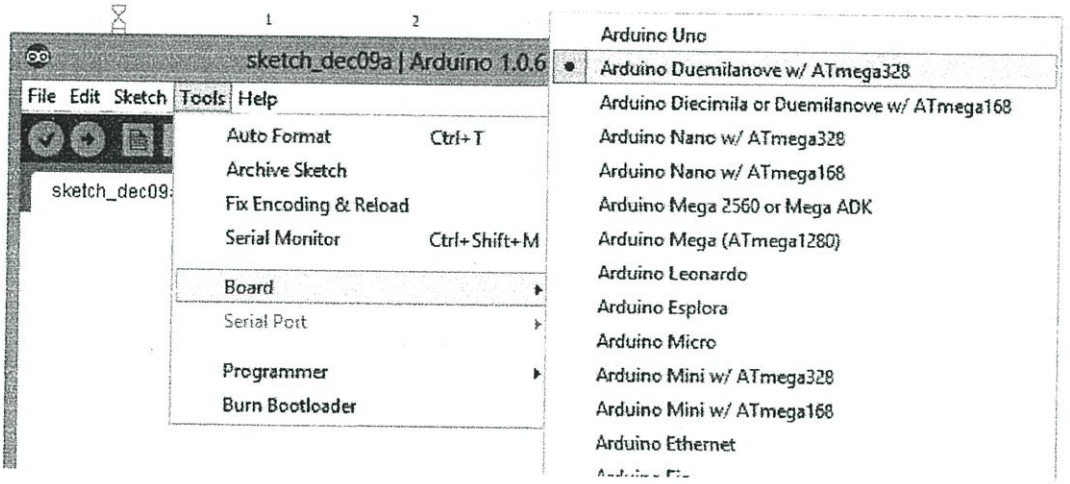


รูปที่ 2.16 หน้าต่างของโปรแกรมอาร์ดูอิโน่

2.4.1.2 การใช้งานโปรแกรมอาร์ดูอิโน่

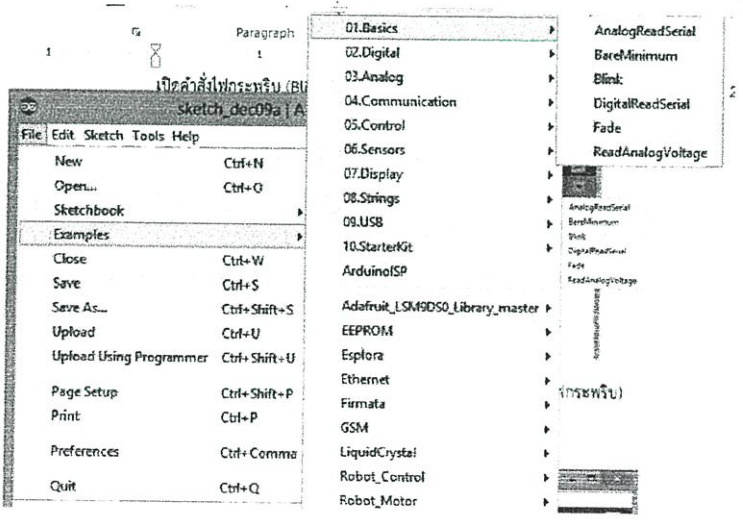
เมื่อทำการติดตั้งโปรแกรมอาร์ดูอิโน่เสร็จแล้ว ก็จะเข้าสู่การใช้งานโปรแกรม โดยมีขั้นตอน ดังนี้

1) เลือกบอร์ดไมโครคอนโทรลเลอร์ที่ใช้โดยมีขั้นตอน คือ Tools > Board >บอร์ดไมโครคอนโทรลเลอร์ที่ใช้ดังรูปที่ 2.17



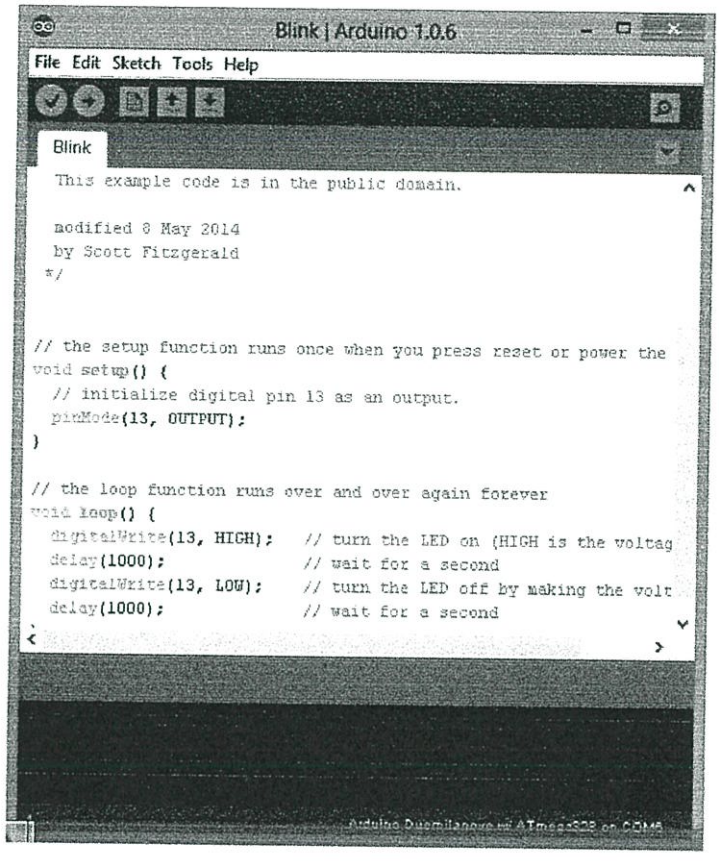
รูปที่ 2.17 ขั้นตอนการเลือกบอร์ดไมโครคอนโทรลเลอร์ที่ใช้

2) เขียนคำสั่ง ซึ่งในที่นี้เป็นการเปิดใช้คำสั่งพื้นฐาน โดยมีขั้นตอน คือ File > Example > เลือกหมวดหมู่ต่อไปหรือกดปุ่ม Open เลือกหมู่ต่อไป แต่ในกรณีนี้เป็นการเปิดคำสั่งไฟกระพริบ (Blink) จึงมีขั้นตอน คือ File > Example > 01.Basics > Blink ดังรูปที่ 2.18



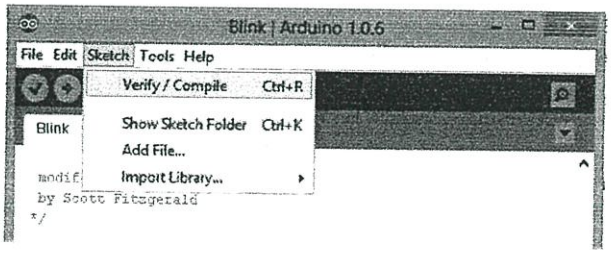
รูปที่ 2.18 ขั้นตอนการเปิดคำสั่งพื้นฐาน (ไฟกระพริบ)

เมื่อกดเลือกคำสั่งไฟกระพริบแล้ว จะได้เป็นดังรูปที่ 2.19



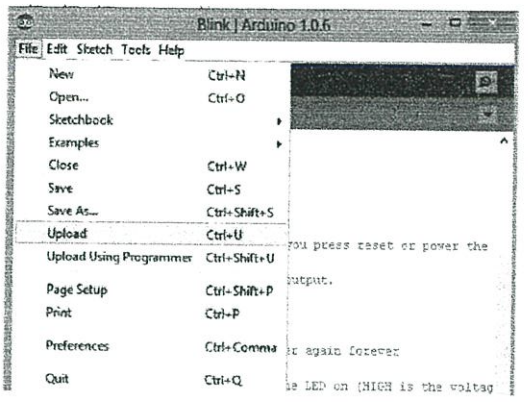
รูปที่ 2.19 หน้าต่างของโปรแกรมอาร์ดูอิโน่ เมื่อเปิดคำสั่งไฟกระพริบ

3) ตรวจสอบว่าคำสั่งนี้มีความผิดพลาดหรือไม่ มีขั้นตอน คือ Sketch > Verify > Compile > ดังรูปที่ 2.20 หรือกดปุ่ม Verify  ซึ่งควรจะทำทุกครั้งเมื่อเขียนคำสั่งเสร็จ ก่อนจะทำการป้อนคำสั่งเข้าไมโครคอนโทรลเลอร์ ดังรูปที่ 2.20



รูปที่ 2.20 ขั้นตอนการตรวจสอบความผิดพลาดของคำสั่ง

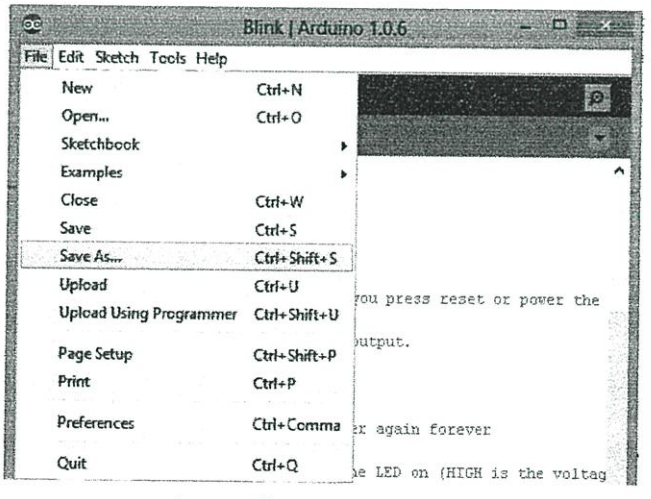
4) ป้อนคำสั่งเข้าสู่ไมโครคอนโทรลเลอร์มีขั้นตอน คือ File > Upload ดังรูปที่ 2.21 หรือกดปุ่ม Upload  ดังรูปที่ 2.21



รูปที่ 2.21 ขั้นตอนการป้อนคำสั่งเข้าสู่ไมโครคอนโทรลเลอร์

และรอจนกระทั่งโปรแกรมทำการป้อนคำสั่งเข้าสู่ไมโครคอนโทรลเลอร์เสร็จ

5) บันทึกคำสั่ง มีขั้นตอน คือ File > Save หรือ Save As ดังรูปที่ 2.22 หรือกดปุ่ม Save



รูปที่ 2.22 ขั้นตอนการบันทึกคำสั่ง

และรอจนกระทั่งโปรแกรมทำการบันทึกคำสั่งเสร็จ ดังรูปที่ 2.23 ซึ่งไฟล์ที่ถูกบันทึกไว้จะมีสกุลไฟล์เป็น .ino

รูปที่ 2.23 เมื่อบันทึกคำสั่งเสร็จ

6) ปิดโปรแกรมโดยกดปุ่ม ที่มุมขวาบนของโปรแกรม

2.4.1.3 ส่วนประกอบของคำสั่งในโปรแกรมอาร์ดูอิโน่

ในการเขียนโปรแกรมอาร์ดูอิโน่นั้นมีหลักการคล้ายคลึงกับการเขียนโปรแกรมภาษาซีพลัสพลัส แต่มีความยุ่งยากน้อยกว่าโดยที่ส่วนประกอบของคำสั่งมีทั้งหมด 3 ส่วน คือ เฮดเดอร์ (header), เซ็ตอัพ (set up()) และลูป (loop()) ดังรูปที่ 2.24

```
Include<... .h>

                ประกาศตัวแปร

Set up()

        {

        คำสั่งที่ถูกเรียกขึ้นมาใช้รอบเดียวตอนเริ่มโปรแกรม

        }

Loop()
        {
                คำสั่งที่ถูกเรียกขึ้นมาใช้ซ้ำๆกัน
                ตามลำดับและเงื่อนไขที่กำหนดให้
        }
```

รูปที่ 2.24 ส่วนประกอบของโปรแกรมอาร์ดูอิโน่

ส่วนของเฮดเดอร์เป็นส่วนที่มีหรือไม่มีก็ได้ เป็นส่วนที่ทำหน้าที่บอกตัวแปลภาษาของอาร์ดูอิโนให้รับรู้ว่าในการแปลคำสั่งของโปรแกรมนี้ มีไฟล์ภายนอกใดบ้างที่จำเป็นต้องใช้ร่วมในการแปลคำสั่ง ถือว่าเป็นคำสั่งพิเศษ ไม่ใช่คำสั่งสำหรับสั่งงานในโปรแกรม เพราะฉะนั้นจึงไม่ต้องมีเครื่องหมายเซมิโคลอน (;) และเป็นส่วนที่ใช้ในการประกาศตัวแปรต่าง ๆ

ส่วนของเซ็ตอัพเป็นส่วนของฟังก์ชันบังคับที่ต้องกำหนดให้มีทุกๆโปรแกรม ถึงแม้ว่าในบางโปรแกรมจะไม่ได้ใช้งาน ก็จำเป็นต้องประกาศตัวแปรไว้โดยฟังก์ชันนี้ใช้สำหรับบรรจุคำสั่งที่ถูกเรียกขึ้นมาใช้รอบเดียวตอนเริ่มโปรแกรม

ส่วนของลูปเป็นส่วนของฟังก์ชันบังคับที่ต้องกำหนดให้มีในทุกๆ โปรแกรม เหมือนกับส่วนของเซ็ตอัพ โดยฟังก์ชันนี้ใช้สำหรับบรรจุคำสั่งที่ถูกเรียกขึ้นมาใช้ซ้ำๆกัน ตามลำดับและเงื่อนไขที่กำหนดให้

2.5 การเชื่อมต่ออุปกรณ์แบบ I²C BUS (Inter Integrate Circuit Bus)

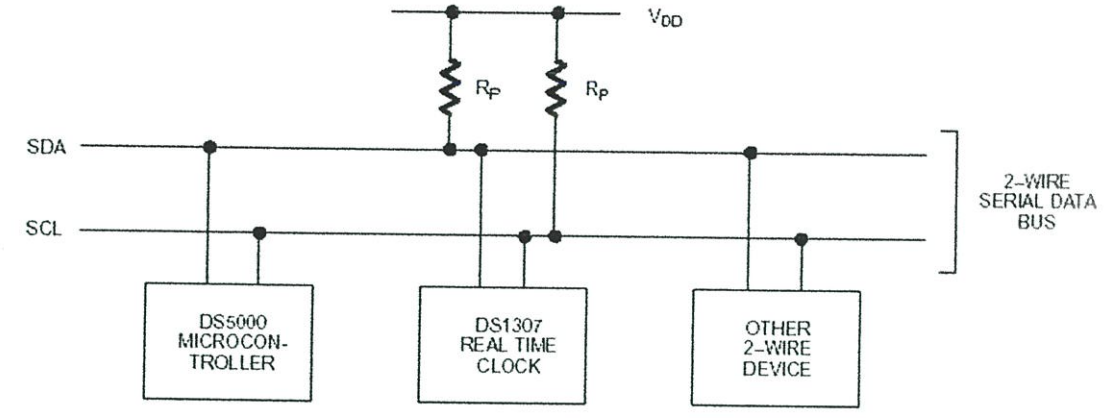
Inter Integrate Circuit Bus (IIC) นิยมเรียกสั้นๆว่า I²C BUS (ไอ-สแควร์-ซี-บัส) เป็นการสื่อสารอนุกรม แบบซิงโครนัส (Synchronous) ใช้ในการติดต่อสื่อสาร ระหว่างไมโครคอนโทรลเลอร์กับอุปกรณ์ภายนอก ซึ่งถูกพัฒนาขึ้นโดยบริษัท Philips Semiconductors โดยใช้สายสัญญาณเพียง 2 เส้นเท่านั้น คือ สายข้อมูลอนุกรม Serial Data (SDA) และสายสัญญาณนาฬิกา Serial Clock (SCL) ซึ่งสามารถ เชื่อมต่ออุปกรณ์ จำนวนหลายๆ ตัว เข้าด้วยกันได้ ทำให้ตัวไมโครคอนโทรลเลอร์สามารถเชื่อมต่อกับอุปกรณ์ภายนอกได้โดยใช้พอร์ตเพียง 2 พอร์ตเท่านั้น

สายข้อมูลอนุกรม และ สายสัญญาณนาฬิกา เป็นสายสัญญาณ 2 ทิศทาง (bi-directional line) ต้องมีการต่อตัวต้านทานพูลอัพกับแรงดัน +5V ไว้ตลอดเวลา เพื่อให้สายมีสถานะลอจิกสูงในขณะที่ไม่มีการติดต่อใช้งาน ทั้งยังช่วยป้องกันสัญญาณรบกวนที่อาจมีเข้ามาในสายสัญญาณทั้งสอง

2.5.1 การเชื่อมต่ออุปกรณ์แบบ I²C BUS

I²C BUS ใช้สายสัญญาณ 2 เส้น คือ สายข้อมูลอนุกรม Serial Data (SDA) และสายสัญญาณนาฬิกา Serial Clock (SCL) สำหรับต่อกับอุปกรณ์แบบ 2 ทิศทาง โดยที่ขาสัญญาณของทั้ง 2 จะต้องต่อกับตัวต้านทานแบบพลูอัพ 2-10 กิโลโอห์ม เพื่อให้สายมีสถานะลอจิกสูงในขณะที่ไม่มีการติดต่อใช้งาน ทั้งยังช่วยป้องกันสัญญาณรบกวนที่อาจมีเข้ามาในสายสัญญาณทั้งสอง เนื่องจาก

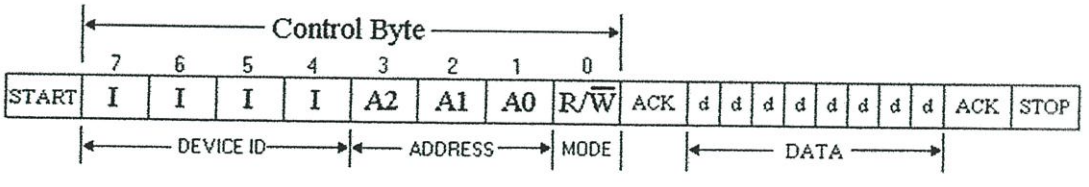
เอาต์พุตมีลักษณะเป็น แบบ Open Drain หรือเป็นแบบ Open Collector เพื่อให้เอาต์พุตเชื่อมต่อกันได้หลายตัว แสดงดังรูปที่ 2.25



รูปที่ 2.25 ลักษณะการเชื่อมต่ออุปกรณ์แบบ I^2C BUS [8]

2.5.2 การเขียน-อ่านข้อมูลกับอุปกรณ์แบบ I^2C BUS

การเขียน - อ่านข้อมูลกับอุปกรณ์แบบ I^2C BUS มีหลักการคือ ไมโครคอนโทรลเลอร์จะเริ่มต้นการส่งข้อมูลด้วยการ ส่งสถานะเริ่มต้น (START Conditions) เพื่อแสดงการขอใช้บัสตามด้วยรหัสควบคุม (Control Byte) ซึ่งประกอบ ด้วยรหัสประจำตัวอุปกรณ์ Device ID, Device Address และ Mode ในการเขียนหรืออ่านข้อมูล เมื่ออุปกรณ์รับทราบว่าไมโครคอนโทรลเลอร์ต้องการจะติดต่อด้วยก็ต้องส่งสถานะรับรู้ (Acknowledge) หรือแจ้งให้ไมโครคอนโทรลเลอร์รับรู้ว่าข้อมูลที่ได้ส่งมามีความถูกต้อง และเมื่อสิ้นสุดการส่งข้อมูล ไมโครคอนโทรลเลอร์จะต้องส่งสถานะสิ้นสุด (STOP Conditions) เพื่อบอกกับอุปกรณ์ว่า สิ้นสุดการใช้บัส โดยจะมีเฟรมของข้อมูล ดังรูปที่ 2.26



รูปที่ 2.26 เฟรมของข้อมูล I^2C BUS [8]

2.5.2.1 สถานะบัสว่าง

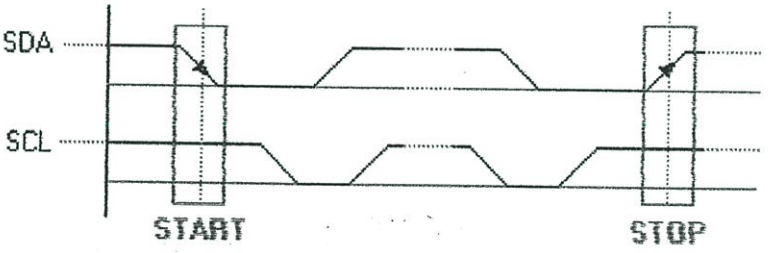
สถานะบัสว่าง (Bus not busy) สภาวะนี้เกิดขึ้นเมื่อ สถานะลอจิกบนสาย SDA และ SCL มีลอจิกสูงทั้งคู่ นั่นหมายความว่า การถ่ายทอดข้อมูลสามารถเริ่มต้นขึ้นได้

2.5.2.2 การกำหนดสถานะเริ่มต้นและสถานะสิ้นสุดของ BUS

ลักษณะการกำหนดสถานะเริ่มต้นและสถานะสิ้นสุดของ I^2C BUS จะกำหนดได้ดังนี้

- เมื่อต้องการส่งข้อมูล ไมโครคอนโทรลเลอร์จะต้องส่งสถานะเริ่มต้น (START Conditions) คือให้ SDA เปลี่ยนลอจิกจาก “1” มาเป็น “0” ในขณะที่ SCL มีค่าเป็น “1”
- เมื่อสิ้นสุดการการใช้บัส ไมโครคอนโทรลเลอร์จะต้องส่งสถานะสิ้นสุด (STOP Conditions) คือให้ SDA เปลี่ยนลอจิกจาก “0” มาเป็น “1” ในขณะที่ SCL มีค่าเป็น “1”

การกำหนดสถานะเริ่มต้นและสถานะสิ้นสุดของ I^2C BUS (START and STOP Conditions) แสดงดังรูปที่ 2.27



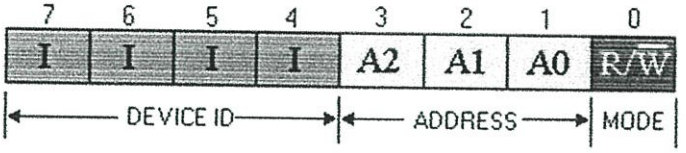
รูปที่ 2.27 สถานะเริ่มต้นและสถานะสิ้นสุดของ BUS [8]

2.5.2.3 รหัสควบคุมของ I^2C BUS (Control Byte)

รหัสควบคุมของ I^2C BUS (Control Byte) ประกอบด้วย

1) รหัสประจำตัวของอุปกรณ์ (Device ID) ประกอบด้วยบิต 1-7 และบิต 0 เป็นบิตควบคุมการเขียนอ่าน โดยรหัสประจำตัวของอุปกรณ์ (Device ID) จะประกอบไปด้วย รหัสประจำตัวของอุปกรณ์ ประกอบด้วยรหัสประจำตัวจากผู้ผลิต Product ID 4 บิต (บิตที่ 4-7) ที่เปลี่ยนแปลงแก้ไขไม่ได้ และ Device Address 3 บิต (บิตที่ 1-3) ซึ่งผู้ใช้ สามารถกำหนดเองได้รวมแล้วเป็นรหัส 7 บิต ใช้ระบุตัวอุปกรณ์ ที่ต่ออยู่บนบัส จะมีค่าซ้ำกันไม่ได้

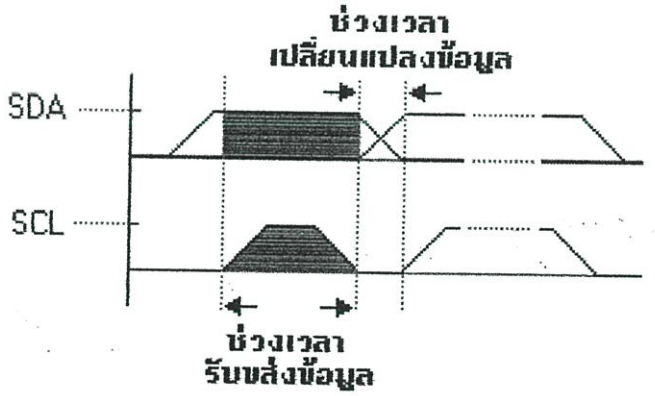
2) บิตควบคุมการเขียนอ่าน (Mode) บิตที่ 0 เมื่อไมโครคอนโทรลเลอร์ต้องการเขียนข้อมูลไปยังอุปกรณ์ก็กำหนดให้บิตนี้เป็น 0 และเมื่อต้องการ อ่านข้อมูล จากอุปกรณ์ ก็กำหนดให้บิตนี้เป็น 1 โดยจะมีเฟรมของข้อมูลแสดงดังรูปที่ 2.28



รูปที่ 2.28 เฟรมของรหัสควบคุม [8]

2.5.2.4 ช่วงเวลารับส่งบิตข้อมูลของ I^2C BUS

การรับ-ส่งข้อมูลจะกระทำผ่านขา SDA โดยจะมีช่วงเวลาการรับ - ส่งข้อมูลดังนี้คือ ช่วงเวลาในการรับ-ส่งข้อมูล จะกระทำในสภาวะที่ขา SCL มีลอจิกเป็น “1” และช่วงเวลาการเปลี่ยนแปลงข้อมูล จะกระทำในสภาวะที่ขา SCL มีลอจิกเป็น “0” แสดงได้ดังรูปที่ 2.29



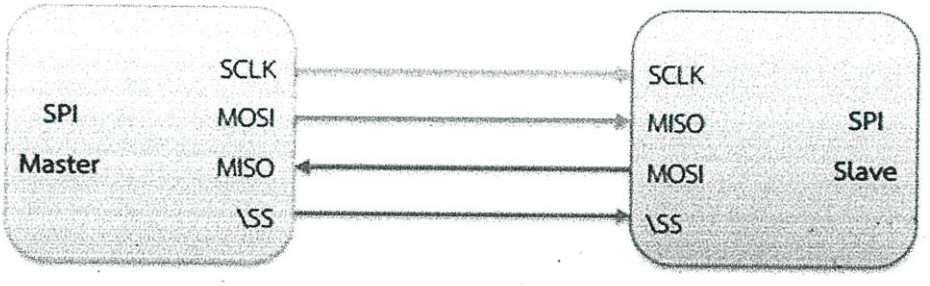
รูปที่ 2.29 ช่วงเวลาในการรับส่งข้อมูล [8]

2.6 การเชื่อมต่ออุปกรณ์แบบ SPI (Serial Peripheral Interface)

SPI หรือ Serial Peripheral Interface เป็นวิธีการสื่อสารอนุกรมแบบ Synchronous อีกรูปแบบหนึ่ง ซึ่งทำงานในรูปแบบที่ให้อุปกรณ์ตัวหนึ่งทำหน้าที่เป็น Master ในขณะที่อีกตัวหนึ่งทำหน้าที่เป็น Slave และสามารถส่งข้อมูลในโหมด Full-duplex นั่นหมายความว่า สัญญาณสามารถส่งหากันได้ระหว่าง Master และ Slave ได้อย่างต่อเนื่อง รูปแบบข้อมูลการสื่อสารหรือ Protocol ของแบบ SPI นี้ ไม่ได้กำหนดมาตรฐานกำหนดตายตัว ว่าข้อมูลที่ส่งหากันต้องอยู่ในรูปแบบหรือ Format แบบไหน เป็นการให้อิสระในการออกแบบ Protocol การสื่อสารกันเอง

2.6.1 การเชื่อมต่ออุปกรณ์แบบ SPI

SPI ต้องการสายสัญญาณ 4 เส้น บางครั้งเราเรียกว่า บัสอนุกรม "four wire" เส้นสัญญาณทั้ง 4 เส้น แสดงดังรูปที่ 2.30 และรายละเอียดของสายสัญญาณแสดงดังตารางที่ 2.3



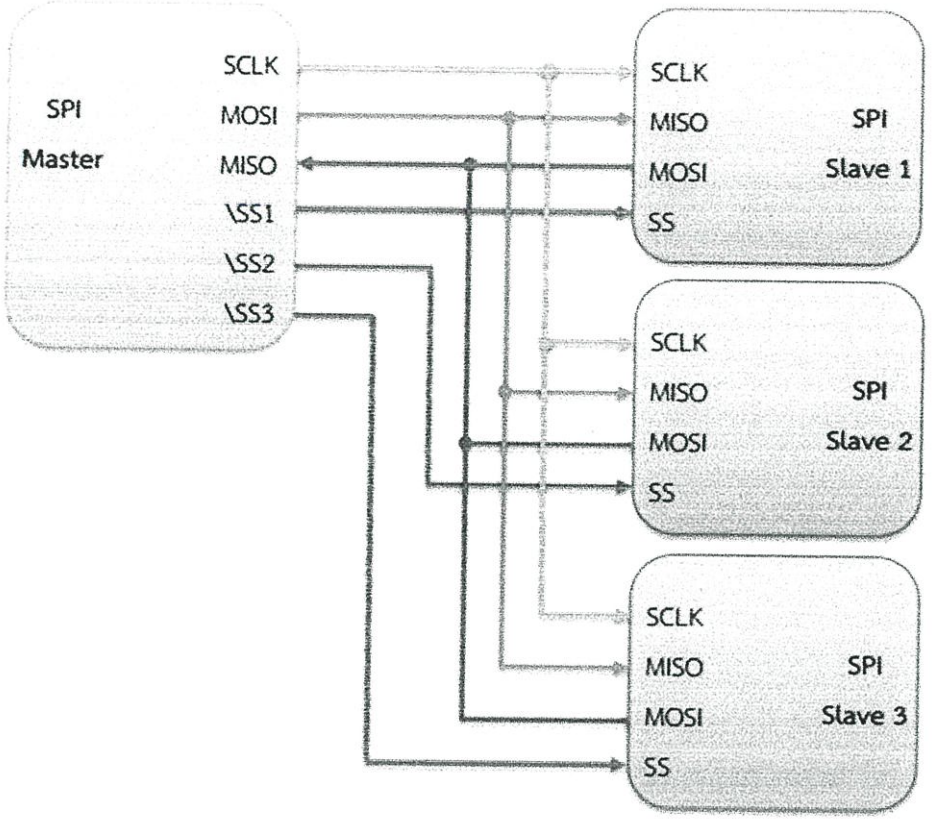
รูปที่ 2.30 การเชื่อมต่อการสื่อสารแบบ SPI ระหว่างอุปกรณ์ Master – Slave โดยมีสายสัญญาณ สี่เส้น หรือ Four Wire [9]

ตารางที่ 2.3 สายสัญญาณต่างๆที่ใช้ในการส่ง แบบ SPI

เส้น	ชื่อ	การทำงาน
SCLK	Serial Clock	ใช้ส่งสัญญาณนาฬิกาจากอุปกรณ์ Master ไปยังอุปกรณ์ Slave เพื่อกำหนดจังหวะการรับส่งข้อมูล
MOSI	Master Out Slave In	ใช้ส่งข้อมูลจากอุปกรณ์ Master ไปยังอุปกรณ์ Slave
MISO	Master In Slave Out	ใช้รับข้อมูลจากอุปกรณ์ Slave
SS	Slave Select	ใช้ส่งสัญญาณ Low ไปยังอุปกรณ์ Slave ที่ต้องการรับส่งข้อมูล หรือเป็นสัญญาณที่ Master ใช้เป็นตัวเลือกว่าจะติดต่อกับ Slave ตัวใด

อุปกรณ์ Master ทำหน้าที่เป็นตัวควบคุมการสื่อสารทั้งหมด โดยควบคุมการสื่อสารตามสัญญาณนาฬิกา ตัว Master จะเป็นตัวที่ตัดสินใจเลือก รับ หรือ ส่งข้อมูลภายในการสื่อสาร ส่วนตัวที่เป็น Slave จะทำหน้าที่รับคำสั่งต่างๆจาก Master

สัญญาณเส้น SS หรือ Slave select ในกรณี ที่มีตัว Slave มากกว่า 1 ตัว โดยการทำให้เส้น SS มีระดับสัญญาณเป็น Low เมื่อต้องการติดต่อกับ Slave ตัวใด ก็เพียงทำให้สัญญาณ SS ของ Slave ตัวนั้น มีระดับสัญญาณเป็น Low ดังรูปที่ 2.31



รูปที่ 2.31 การเชื่อมต่อการสื่อสารแบบ SPI ระหว่างอุปกรณ์ Master – Slave หลายตัว [9]

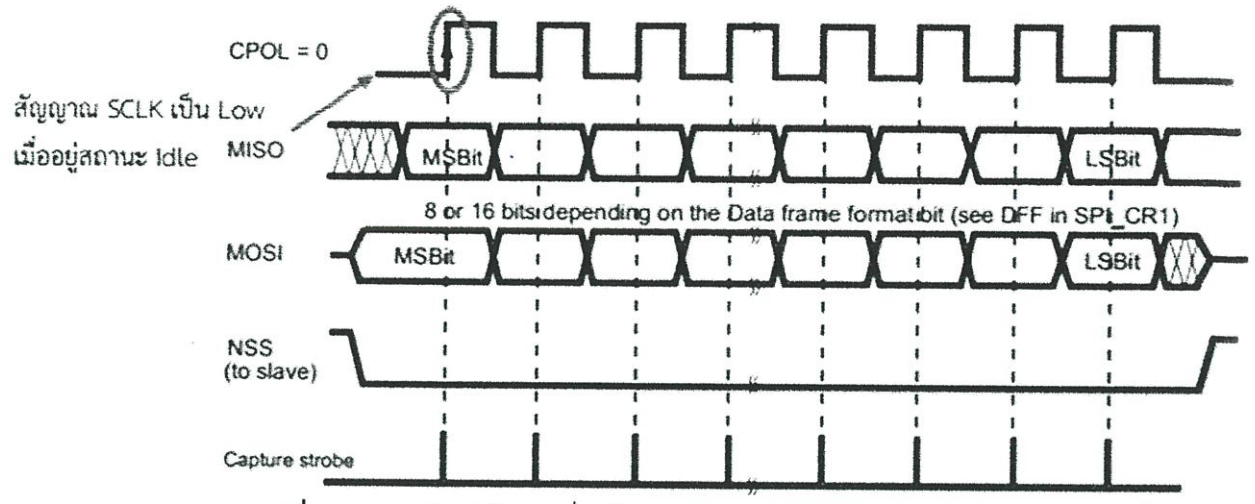
การสื่อสารแบบ SPI ไม่มีรูปแบบที่แน่นอน ดังนั้นต้องกำหนดรูปแบบการสื่อสารของอุปกรณ์ Master ให้ตรงกับอุปกรณ์ Slave โดยมีรายละเอียดที่สำคัญ ดังนี้

1. กำหนดทิศทางการสื่อสาร Full Duplex (รับส่งข้อมูลได้พร้อมกัน) Half Duplex_Tx ส่งข้อมูลอย่างเดียว Half Duplex_Rx รับข้อมูลอย่างเดียว

2. รูปแบบการรับส่งข้อมูล 8 บิต หรือ 16 บิต โดยสามารถเลือกส่งบิตนัยยะสำคัญสูงสุด (MSB) หรือ บิตนัยยะสำคัญต่ำสุด (LSB) ก่อน

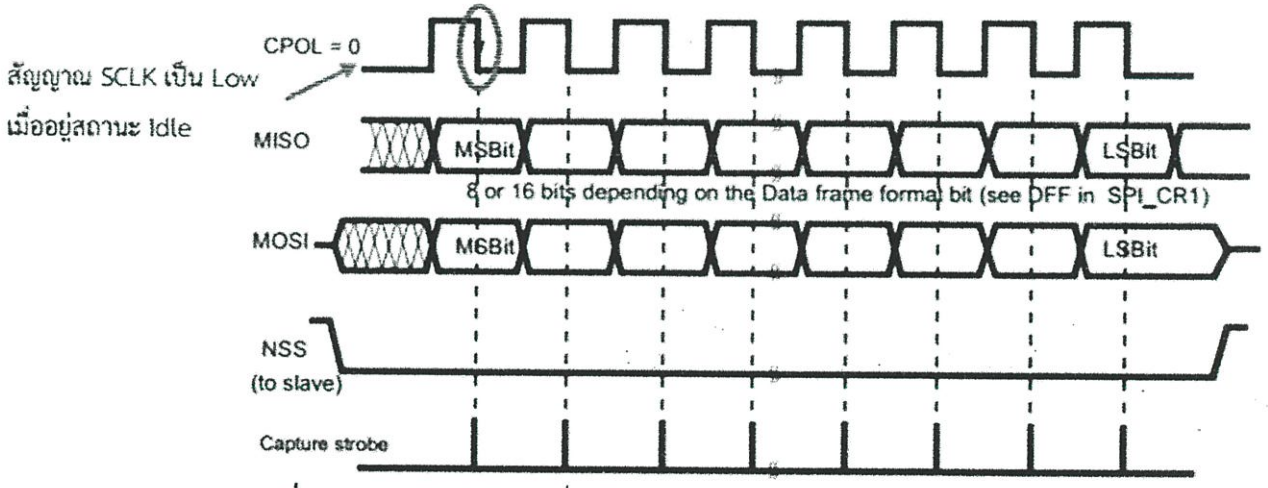
3. ลักษณะของขอบสัญญาณนาฬิกา เพื่อกำหนดจังหวะการรับ-ส่งข้อมูล สามารถเลือกได้สี่แบบ ดังนี้

Clock polarity Low และ Clock phase 1st edge (CPOL_Low/CPHA_1Edge) ในสถานะเริ่มต้น สัญญาณนาฬิกาจะอยู่ในสถานะ Low ระบบจะเริ่มอ่านข้อมูลบิตแรก เมื่อพบขอบแรกของสัญญาณซึ่งเป็นลักษณะขอบขาขึ้น ดังรูปที่ 2.32



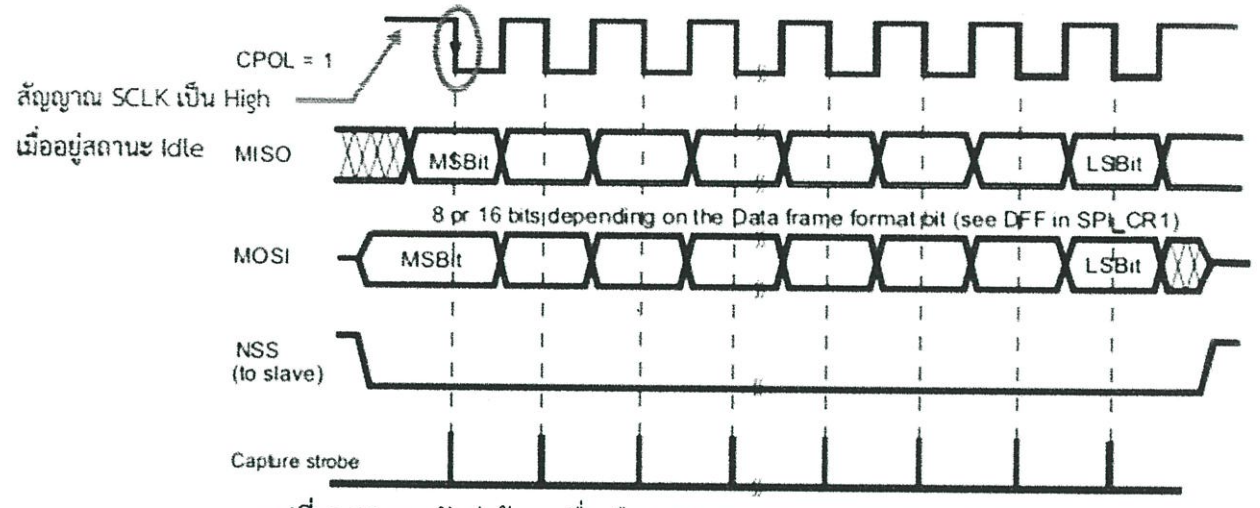
รูปที่ 2.32 การรับส่งข้อมูลเมื่อเลือก CPOL = 0 และ CPHA = 0 [9]

Clock polarity Low และ Clock phase 2st edge (CPOL_Low/CPHA_2Edge) ระบบจะเริ่มอ่านข้อมูลบิตแรก เมื่อพบขอบที่สองของสัญญาณซึ่งเป็นลักษณะขอบขาลง ดังรูปที่ 2.33



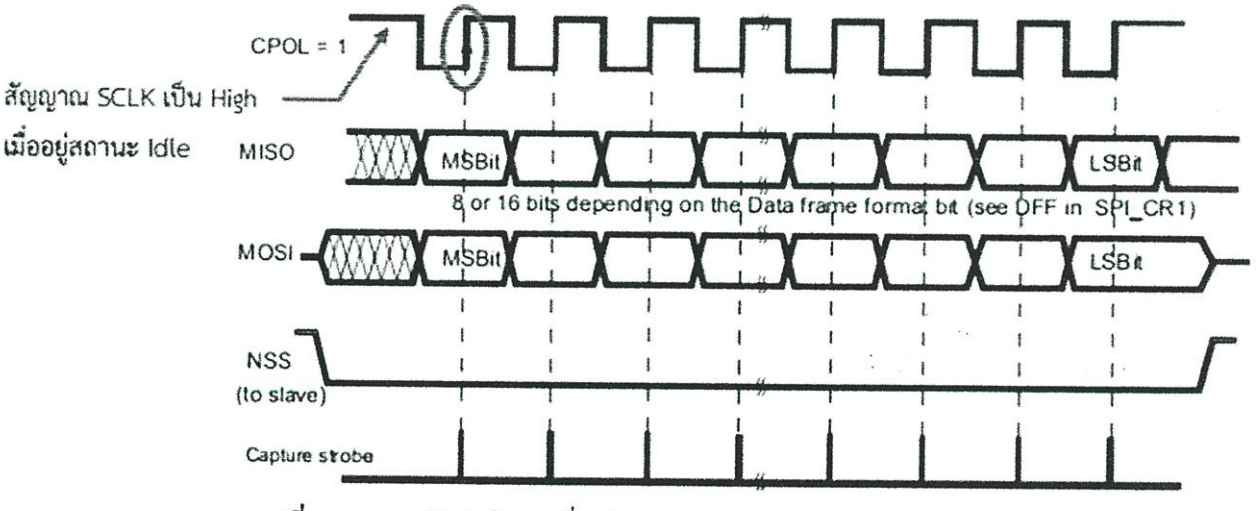
รูปที่ 2.33 การรับส่งข้อมูลเมื่อเลือก CPOL = 0 และ CPHA = 1 [9]

Clock polarity High และ Clock phase 1st edge (CPOL_High/CPHA_1Edge) ในสถานะเริ่มต้น สัญญาณนาฬิกาจะอยู่ในสถานะ High ระบบจะเริ่มอ่านข้อมูลบิตแรก เมื่อพบขอบแรกของสัญญาณซึ่งเป็นลักษณะขอบขาลง ดังรูปที่ 2.34



รูปที่ 2.34 การรับส่งข้อมูลเมื่อเลือก CPOL = 1 และ CPHA = 0 [9]

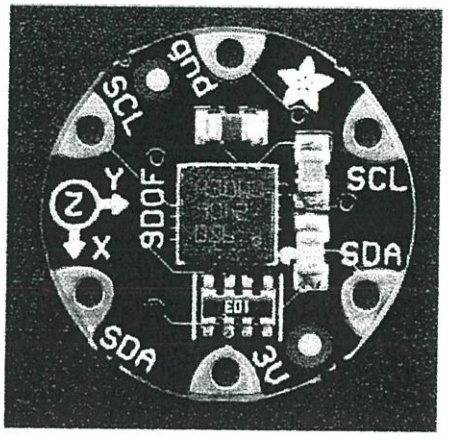
Clock polarity High และ Clock phase 2st edge (CPOL_High/CPHA_2Edge) ระบบจะเริ่มอ่านข้อมูลบิตแรก เมื่อพบขอบที่สองของสัญญาณซึ่งเป็นลักษณะขอบขาขึ้น ดังรูปที่ 2.35



รูปที่ 2.35 การรับส่งข้อมูลเมื่อเลือก CPOL = 1 และ CPHA = 1 [9]

2.7 โมดูลเซ็นเซอร์ LSM9DS0

โมดูลเซ็นเซอร์ LSM9DS0 แสดงดังรูปที่ 2.36 เป็นโมดูลที่สามารถวัดความเร่งเชิงเส้น 3 แกน (linear acceleration) วัดค่าสนามแม่เหล็ก 3 แกน และวัดค่าความเร็วเชิงมุม 3 แกน แกนอ้างอิงคือแกน X, แกน Y และแกน Z และใช้การรับส่งข้อมูลในรูปแบบของ I^2C BUS หรือ SPI



รูปที่ 2.36 โมดูลเซ็นเซอร์ LSM9DS0 [10]

2.7.1 คุณสมบัติของโมดูลเซ็นเซอร์ LSM9DS0

- อ่านค่าความเร่งเชิงเส้น ความเร็วเชิงมุม สนามแม่เหล็กได้ใน 3 แกนอ้างอิง
- สามารถทำงานได้ในระดับไฟเลี้ยงต่ำที่ 2.4-3.6 โวลต์
- ส่งข้อมูลได้ 16 บิต

2.7.2 ขาต่างๆ ของโมดูล

การจัดเรียงขาต่าง ๆ ชิปภายในโมดูลเซ็นเซอร์ LSM9DS0 จะเป็นไปตามรูปที่ 2.36 และตารางที่ 2.4

ตารางที่ 2.4 การจัดเรียงขาของโมดูลเซ็นเซอร์ LSM9DS0

ขา	ชื่อ	ความหมาย
1	VDD	ไฟเลี้ยง เป็นขาสำหรับต่อไฟเลี้ยง (2.4 โวลต์ ถึง 3.6 โวลต์)
2	GND	กราวด์ เป็นขาไว้สำหรับต่อลงกราวด์ของวงจร
3	SCL	ขาสำหรับต่อ สายสัญญาณนาฬิกา Serial Clock (SCL)
4	SDA	ขาสำหรับต่อ สายข้อมูลอนุกรม Serial Data (SDA)

2.7.3 ค่า I^2C Address ของโมดูลเซ็นเซอร์ LSM9DS0

ค่า I^2C Address ของโมดูลเซ็นเซอร์ LSM9DS0 จะมีค่าดังตารางที่ 2.5 และ 2.6

ตารางที่ 2.5 ค่า I^2C Address ของค่าความเร่งเชิงเส้น และ ค่าสนามแม่เหล็ก

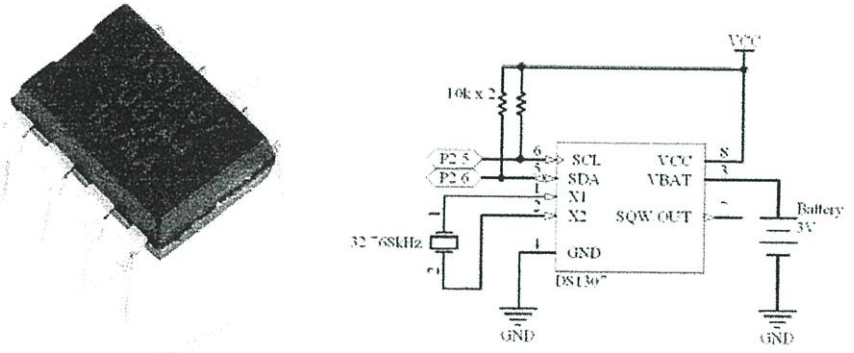
Command	Address
Read	00111101 (3D)
Write	00111100 (3C)
Read	00111011 (3B)
Write	00111010 (3A)

ตารางที่ 2.6 ค่า I^2C Address ของค่าความเร็วเชิงมุม

Command	Address
Read	11010101 (D5h)
Write	11010100 (D4h)
Read	11010111 (D7h)
Write	11010110 (D6h)

2.8 วงจรเรียลไทม์คล็อก (Real Time Clock)

ในการสร้างวงจรเรียลไทม์คล็อกนั้น ในปริญญานิพนธ์นี้จะใช้ไอซี DS1307 ซึ่ง DS1307 เป็นไอซีฐานเวลาของดัลลัสเซมิคอนดักเตอร์ (Dallas Semiconductor) มีบัสรับส่งข้อมูลแบบ I²C ซึ่งเป็นแบบ 2 เส้น สามารถสื่อสารได้ 2 ทิศทาง (bi-direction bus) ฐานเวลาของ DS1307 นั้นสามารถเก็บข้อมูล วินาที , นาที , ชั่วโมง , วัน , วันที่ , เดือน และปีได้ ระบบเวลาสามารถทำงานโหมดรูปแบบ 24 ชั่วโมง หรือ 12 ชั่วโมง AM/PM ก็ได้ ภายในมีระบบตรวจจับแหล่งจ่ายไฟ โดยถ้าแหล่งจ่ายไฟหลักถูกตัดไป DS1307 สามารถสวิตช์ไปใช้ไฟจากแบตเตอรี่ และทำงานต่อไป โดยที่ยังสามารถรักษาข้อมูลไว้ได้ โครงสร้างมีขาทั้งหมด 8 ขาดังแสดงในรูปที่ 2.37 และมีรายละเอียดการทำงานของขาต่าง ๆ ดังตารางที่ 2.7



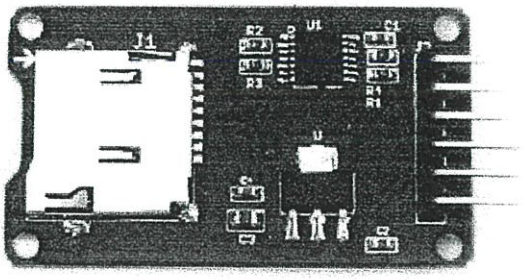
รูปที่ 2.37 ไอซี DS1307 และวงจร Real Time Clock [11]

ตารางที่ 2.7 การจัดเรียงขาของไอซี DS1307

ขา	ชื่อ	ความหมาย
1	X1	ใช้ต่อกับคริสตอลความถี่มาตรฐาน 32.768 kHz เพื่อสร้างฐานเวลาจริงให้กับ IC
2	X2	ใช้ต่อกับคริสตอลความถี่มาตรฐาน 32.768 kHz เพื่อสร้างฐานเวลาจริงให้กับ IC
3	VBAT	ใช้ต่อกับแบตเตอรี่ 3V เพื่อรักษาการทำงาน ในกรณีที่ไม่มีไฟเลี้ยงจ่าย
4	GND	ใช้ต่อกราวด์
5	SDA	ขาสำหรับต่อ สายข้อมูลอนุกรม Serial Data (SDA
6	SCL	ขาสำหรับต่อ สายสัญญาณนาฬิกา Serial Clock (SCL)
7	SQW/Out	ขาเอาต์พุตสัญญาณ Square Wave สามารถเลือกความถี่ได้
8	VCC	ใช้ต่อไฟเลี้ยง +5V

ระบบบัสข้อมูลแบบ I^2C (Inter-IC Communication) ได้ถูกพัฒนาขึ้นโดยบริษัท ฟิลิปส์ (Phillips) การรับส่งข้อมูลใช้สายสัญญาณเพียงแค่ 2 เส้น คือ สายสัญญาณข้อมูล SDA (Serial Data line) และสายสัญญาณนาฬิกา SCL (Serial Clock line) มีการทำงานเป็นแบบMaster , Slave โดยอุปกรณ์ที่ทำหน้าที่เป็น Master (วงจรไมโครคอนโทรลเลอร์) จะควบคุมการรับส่งข้อมูล และควบคุมสัญญาณนาฬิกาบน SCL ส่วนอุปกรณ์ Slave (DS1307) นั้นจะทำงานภายใต้การควบคุมของอุปกรณ์ Master

2.9 Micro SD Card Module



รูปที่ 2.38 Micro SD Card Module [12]

Micro SD Card Module เป็นอุปกรณ์ที่ใช้สำหรับบันทึกข้อมูล ใช้การรับส่งข้อมูลในรูปแบบของ Serial Peripheral Protocol (SPI) โดยทั่วไป การสื่อสารแบบ SPI นั้นสามารถใช้สื่อสารระหว่างบอร์ดไมโครคอนโทรลเลอร์กับอุปกรณ์ต่อพ่วงอื่นๆได้พร้อมกันหลายๆชิ้น เช่น ติดต่อ ADC, SD Card Module, Sensors ต่างๆได้ และรายละเอียดต่างๆ ของขาดังกล่าวแสดงได้ตามตาราง ที่ 2.8

2.9.1 คุณสมบัติของ Micro SD Card Module

- รองรับการ์ด Micro SD, การ์ด Micro SDHC (การ์ดความเร็วสูง)
- ใช้ไฟเลี้ยงในการทำงานที่ระดับ 4.5 - 5 โวลต์
- แผงวงจรควบคุมแรงดันไฟฟ้า 3.3 โวลต์ มีให้ในตัวบอร์ด

ตารางที่ 2.8 การจัดเรียงขาของ Micro SD Card Module

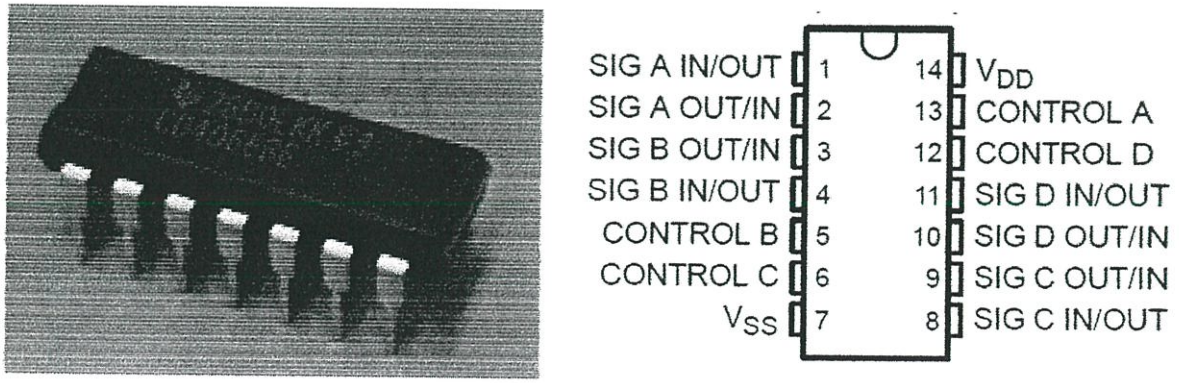
ชื่อ	ความหมาย
VDD	ไฟเลี้ยง เป็นขาสำหรับต่อไฟเลี้ยง (4.5 โวลต์ ถึง 5 โวลต์)
GND	กราวด์ เป็นขาไว้สำหรับต่อลงกราวด์ของวงจร
SCK	ใช้ส่งสัญญาณนาฬิกาจากอุปกรณ์ Master ไปยังอุปกรณ์ Slave เพื่อกำหนดจังหวะการรับส่งข้อมูล
MOSI	ใช้ส่งข้อมูลจากอุปกรณ์ Master ไปยังอุปกรณ์ Slave
MISO	ใช้รับข้อมูลจากอุปกรณ์ Slave
CS	ใช้ส่งสัญญาณ Low ไปยังอุปกรณ์ Slave ที่ต้องการรับส่งข้อมูล หรือเป็นสัญญาณที่ Master ใช้เป็นตัวเลือกว่าจะติดต่อกับ Slave ตัวใด

2.10 ไอซี CD4066B

ไอซี CD4066B เป็นไอซีที่ใช้สำหรับสร้างวงจรสวิทช์ โดยสามารถทำการส่งผ่านหรือมัลติเพล็กซ์ของสัญญาณอะนาลอกหรือสัญญาณดิจิตอล โดยโครงสร้างมีขาทั้งหมด 14 ขาดังแสดงในรูปที่ 2.39 และรายละเอียดต่างๆ ของขาดังกล่าวแสดงได้ตามตารางที่ 2.9

2.10.1 คุณสมบัติของไอซี CD4066B

- รองรับได้ทั้งสัญญาณอะนาลอกและสัญญาณดิจิตอล
- ใช้ไฟเลี้ยงในการทำงานที่ระดับ 3 - 18 โวลต์
- ไอซี 1 ตัว ประกอบด้วยสวิทช์ 4 ตัว
- ควบคุมการรับส่งสัญญาณได้ทั้งหมด 4 ช่องทางในไอซี 1 ตัว



รูปที่ 2.39 ไอซี CD4066B [13]

ตารางที่ 2.9 การจัดเรียงขาของ ไอซี CD4066B

ขา	ชื่อ	ความหมาย
1	SIG A IN/OUT	ขา A สำหรับรับสัญญาณขาเข้าหรือส่งออกสัญญาณ
2	SIG A OUT/IN	ขา A สำหรับส่งออกสัญญาณหรือรับสัญญาณขาเข้า
3	SIG B IN/OUT	ขา B สำหรับรับสัญญาณขาเข้าหรือส่งออกสัญญาณ
4	SIG B OUT/IN	ขา B สำหรับส่งออกสัญญาณหรือรับสัญญาณขาเข้า
5	CONTROL B	ใช้ควบคุมการรับส่งสัญญาณขาเข้าและสัญญาณขาออก ของขา B
6	CONTROL C	ใช้ควบคุมการรับส่งสัญญาณขาเข้าและสัญญาณขาออก ของขา C
7	VSS	ใช้ต่อกราวด์ของวงจร
8	SIG C IN/OUT	ขา C สำหรับรับสัญญาณขาเข้าหรือส่งออกสัญญาณ
9	SIG C OUT/IN	ขา C สำหรับส่งออกสัญญาณหรือรับสัญญาณขาเข้า
10	SIG D OUT/IN	ขา D สำหรับส่งออกสัญญาณหรือรับสัญญาณขาเข้า
11	SIG D IN/OUT	ขา D สำหรับรับสัญญาณขาเข้าหรือส่งออกสัญญาณ
12	CONTROL D	ใช้ควบคุมการรับส่งสัญญาณขาเข้าและสัญญาณขาออก ของขา D
13	CONTROL A	ใช้ควบคุมการรับส่งสัญญาณขาเข้าและสัญญาณขาออก ของขา A
14	VDD	ไฟเลี้ยง เป็นขาสำหรับต่อไฟเลี้ยง (3 โวลต์ ถึง 18 โวลต์)

2.11 การคำนวณอัตราการใช้พลังงานของร่างกาย

2.11.1 การคำนวณอัตราการใช้พลังงานในภาวะร่างกายปกติ ใน 1 วัน (สูตร BMR)

ในการคำนวณหาค่า BMR สามารถทำได้จากสูตร ดังนี้ [11]

ผู้หญิง BMR = 655.1 + (9.56 x W) + (1.85 x H) – (4.7 x A)

ผู้ชาย BMR = 66.5 + (13.7 x W) + (5.0 x H) – (6.8 x A)

เมื่อ W = น้ำหนักตัว เป็นกิโลกรัม

H = ความสูง เป็นเซนติเมตร

A = อายุ เป็นปี

BMR ที่คำนวณได้มีหน่วยเป็นกิโลแคลอรีต่อวันหรือต่อ 24 ชั่วโมง เรียกว่า basal energy expenditure (BEE)

ตารางที่ 2.10 ค่า BMR ของคนในแต่ละช่วงอายุ [11]

อายุ (ปี)	ผู้ชาย				ผู้หญิง			
	ต่อชั่วโมง		ต่อวัน		ต่อชั่วโมง		ต่อวัน	
	กิโลแคลอรี	กิโลจูล	กิโลแคลอรี	กิโลจูล	กิโลแคลอรี	กิโลจูล	กิโลแคลอรี	กิโลจูล
20	70	293	1680	7030	60	251	1440	6010
30	67	280	1608	6820	59	247	1418	5930
40	66	276	1584	6580	59	245	1414	5880
50	65	272	1560	6520	58	242	1392	5800
60	64	267	1536	6420	56	234	1344	5620
70	62	259	1488	6210	54	226	1296	5420
80	60	251	1440	6010	52	218	1248	5230

2.11.2 ค่าพลังงานที่ใช้สำหรับทำกิจกรรม (ที่ใช้ในการทำปริญญานิพนธ์)

ตารางที่ 2.11 จำนวนพลังงานที่ใช้ในการทำกิจกรรมต่างๆ (ไม่รวมพลังงาน BMR) [11]

ชนิดของกิจกรรม	จำนวนพลังงานที่ต้องการต่อชั่วโมง		จำนวนพลังงานที่ต้องการต่อวินาที	
	กิโลจูล	กิโลแคลอรี	กิโลจูล	กิโลแคลอรี
ยืน	586	140	0.1628	0.0389
นั่ง	63	15	0.0175	0.0042
นอน	314	75	0.0872	0.0208
เดิน	1255	300	0.3486	0.0833

2.11.3 โปรแกรมที่ใช้คำนวณค่าพลังงานที่ใช้ทำกิจกรรม (ยืน นั่ง นอน และเดิน)

ในการทำปริญญานิพนธ์นี้ จะใช้โปรแกรม แมทแลป (**Matlab**) ในการเขียนโปรแกรมคำนวณค่าพลังงานที่ใช้ทำกิจกรรม (ยืน นั่ง นอน และเดิน)

2.11.3.1 โปรแกรม แมทแลป (**Matlab**)

โปรแกรม แมทแลป (**Matlab**) เป็นซอฟต์แวร์ในการคำนวณและการเขียนโปรแกรม โปรแกรมหนึ่ง ที่มีความสามารถครอบคลุมตั้งแต่ การพัฒนาอัลกอริธึม การสร้างแบบจำลองทางคณิตศาสตร์ และการจำลองของระบบ การสร้างระบบควบคุม การสร้างเมตริกซ์ ผลิตโดยบริษัทแมตเวิรกส์

โปรแกรม แมทแลป สามารถทำงานได้ทั้งในลักษณะของการติดต่อโดยตรง คือการเขียนคำสั่งเข้าไปทีละคำสั่ง เพื่อให้แมตแลปประมวลผลไปเรื่อยๆ หรือสามารถที่จะรวบรวม ชุดคำสั่งเรานั้นเป็นโปรแกรมก็ได้ ข้อสำคัญอย่างหนึ่งของแมตแลปก็คือข้อมูลทุกตัวจะถูกเก็บใน ลักษณะของแถวลำดับ คือในแต่ละตัวแปรจะได้รับการแบ่งเป็นส่วนย่อยเล็กๆขึ้น ซึ่งการใช้ตัวแปรเป็นแถวลำดับการเขียนโปรแกรมในภาษาขั้นต่ำทั่วไป ซึ่งทำให้สามารถที่จะแก้ปัญหาของตัวแปรที่อยู่ในลักษณะของเมทริกซ์และเวกเตอร์ได้โดยง่าย ซึ่งทำให้ลดเวลาการทำงานลงได้อย่างมากเมื่อเทียบกับการเขียนโปรแกรมโดยภาษาซีหรือภาษาฟอร์แทรน

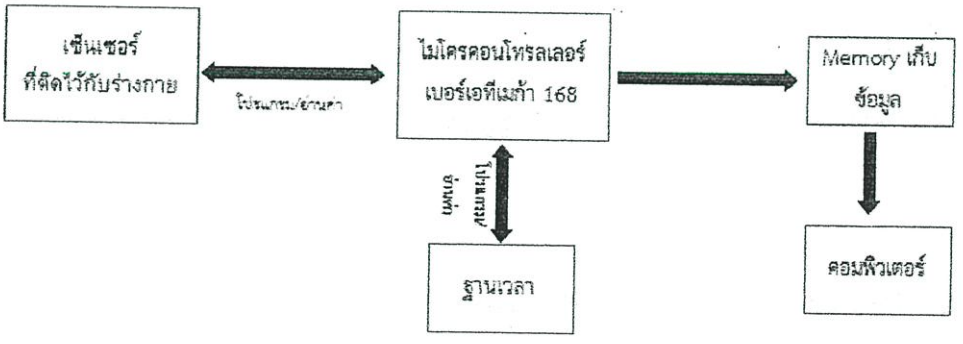
บทที่ 3

การออกแบบและการจัดทำปริญญานิพนธ์

3.1 การออกแบบ

3.1.1 การออกแบบระบบโดยรวม

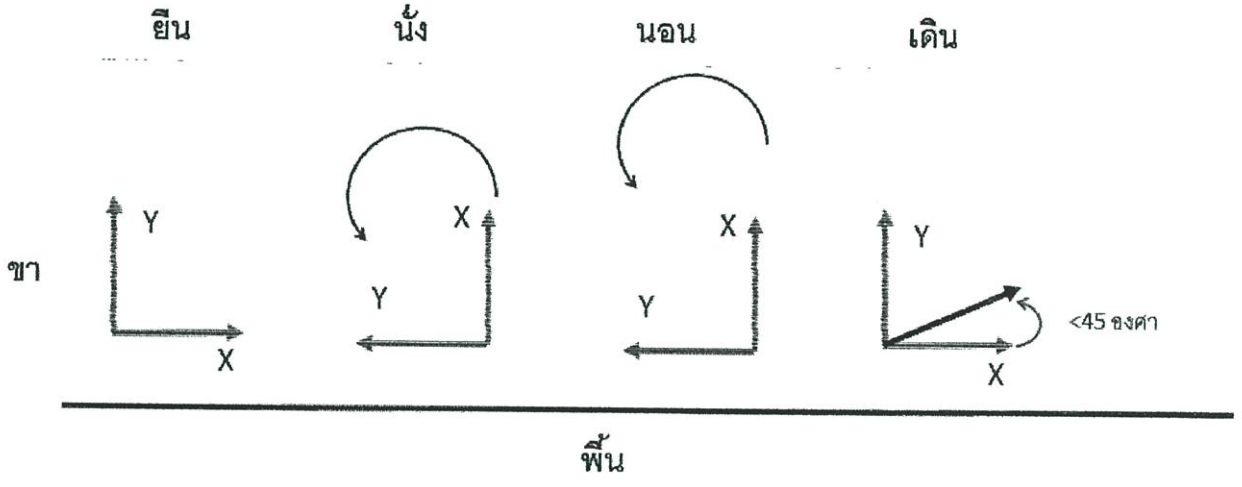
การออกแบบระบบสำหรับเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน เมื่อเกิดการเคลื่อนไหวของร่างกาย ตัวเซ็นเซอร์จะทำการส่งค่าความเร่งเชิงเส้น 3 แกน มาที่ไมโครคอนโทรลเลอร์เบอร์เอทีเมก้า 168 ที่ใช้ร่วมกับโปรแกรมอาร์ดูอิโนในการเขียนคำสั่งให้ไมโครคอนโทรลเลอร์ทำงาน เพื่อประมวลผลว่าในขณะนั้นร่างกายอยู่ในลักษณะ ยืน นั่ง นอน หรือเดิน พร้อมกับการบันทึกค่าเวลาที่ ณ เวลาที่เคลื่อนไหวของร่างกาย จากนั้นอุปกรณ์ไมโครคอนโทรลเลอร์จะส่งทั้งค่าความเร่งเชิงเส้น 3 แกน และค่าเวลามาเก็บไว้ในเอสดีการ์ดโมดูล โดยข้อมูลกิจกรรมการเคลื่อนไหวของร่างกายในแต่ละวันจะถูกนำมาประมวลผลเพื่อคำนวณหาอัตราการใช้พลังงานของคนในคอมพิวเตอร์โดยการออกแบบโครงสร้างโดยรวมของเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกายจะแสดงดังบล็อกไดอะแกรมรูปที่ 3.1



รูปที่ 3.1 บล็อกไดอะแกรมของเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย

3.1.2 การออกแบบโมดูลเซ็นเซอร์ LSM9DS0 กับลักษณะการเคลื่อนไหวของร่างกาย

ในการออกแบบโมดูลเซ็นเซอร์ LSM9DS0 กับลักษณะการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน นั้นโดยจะใช้โมดูลเซ็นเซอร์ LSM9DS0 ติดที่บริเวณต้นขาขวา เพื่อบ่งบอกถึงการเคลื่อนไหวของร่างกาย โดยจะมีลักษณะการออกแบบ ดังรูปที่ 3.2



รูปที่ 3.2 การออกแบบลักษณะของโมดูลเซ็นเซอร์ LSM9DS0 ในการเคลื่อนไหวของร่างกาย (การยืน การนั่ง การนอน และการเดิน)

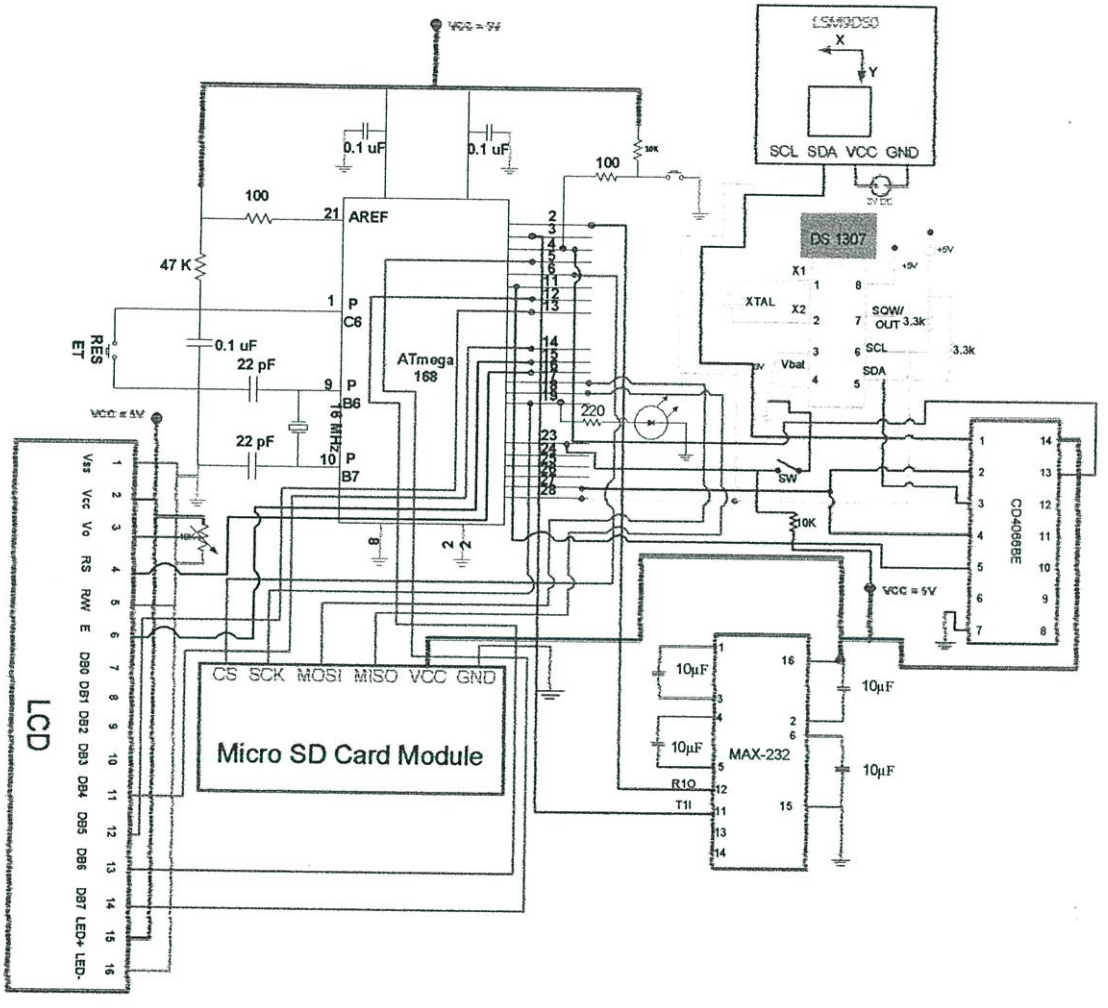
จากรูปที่ 3.2 ในลักษณะการยืนนั้น โมดูลเซ็นเซอร์ LSM9DS0 นั้นจะอยู่ในทิศทางที่แกน X จะขนานกับพื้น และแกน Y จะตั้งฉากกับพื้น

การนั่ง โมดูลเซ็นเซอร์ LSM9DS0 นั้นจะอยู่ในทิศทางที่แกน X จะตั้งฉากกับพื้น และแกน Y จะขนานกับพื้น

การนอน โมดูลเซ็นเซอร์ LSM9DS0 นั้นจะอยู่ในทิศทางที่แกน X จะตั้งฉากกับพื้น และแกน Y จะขนานกับพื้น เช่นเดียวกับการนั่ง

การเดิน โมดูลเซ็นเซอร์ LSM9DS0 นั้นจะอยู่ในทิศทางที่แกน X จะมีการเปลี่ยนค่ามุมในระหว่างการก้าวเท้าไม่เกิน 45 องศา และแกน Y จะตั้งฉากกับพื้น

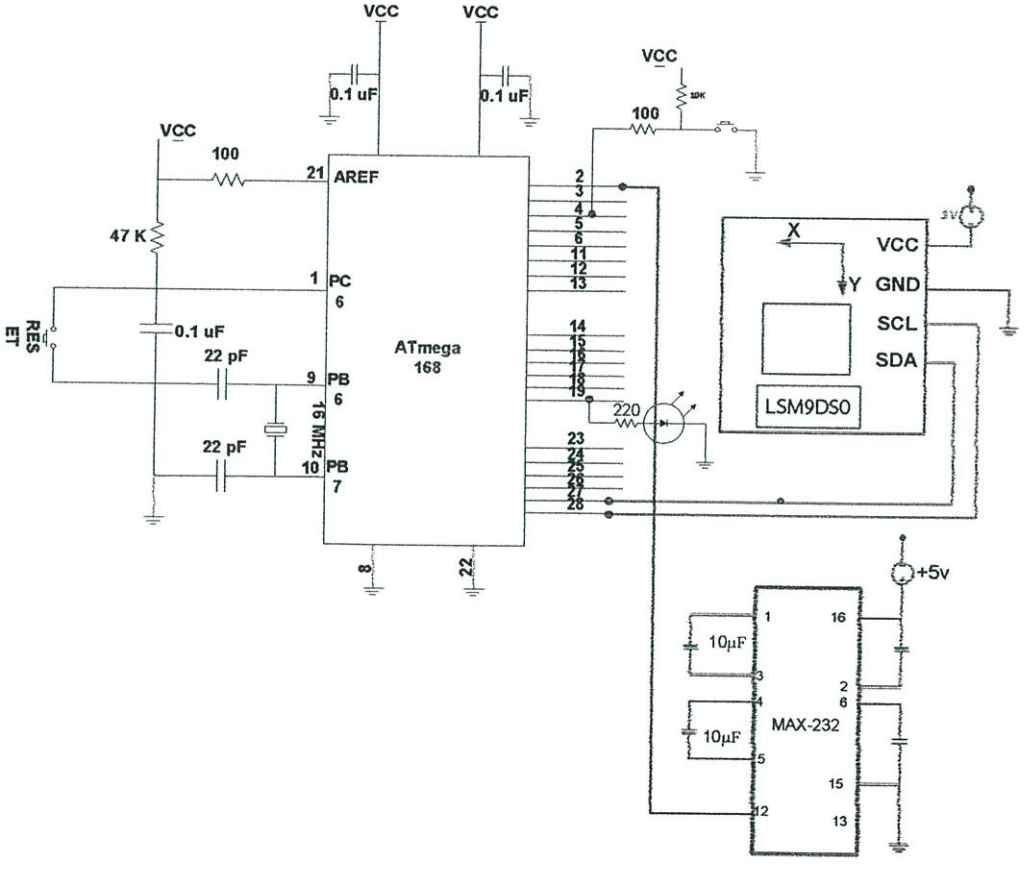
โดยเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย มีโครงสร้าง และการต่อวงจรโดยรวมทั้งหมดของเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย จะแสดงได้ดังรูปที่ 3.3



รูปที่ 3.3 การต่อวงจรรวมทั้งหมดของเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย

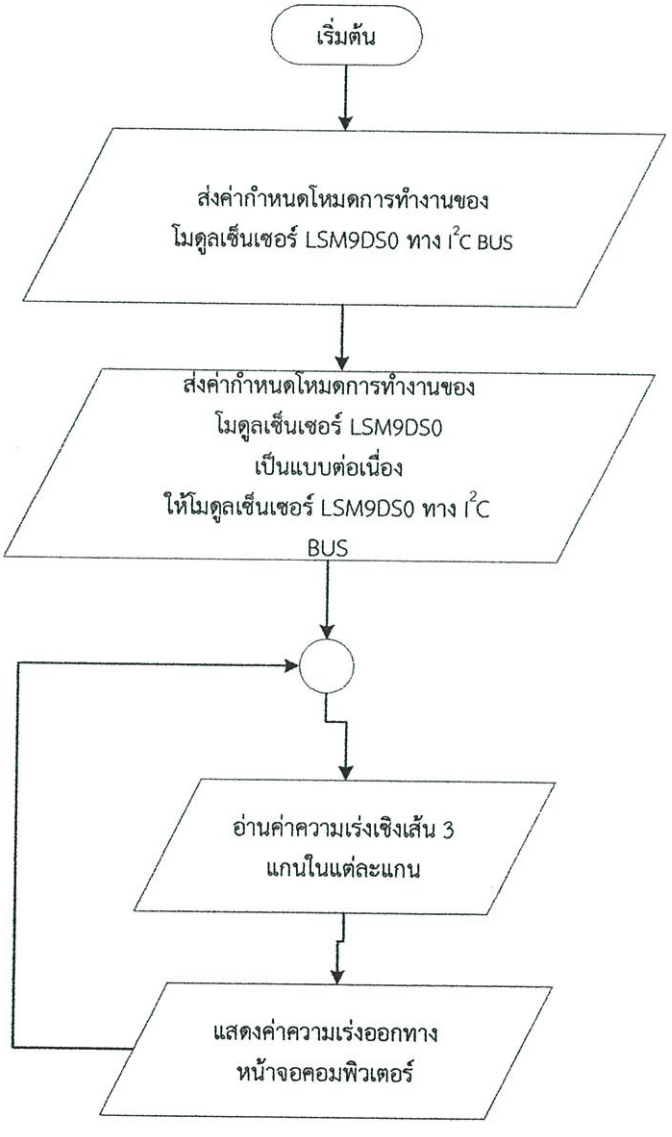
3.1.3 การออกแบบวงจรของโมดูลเซ็นเซอร์ LSM9DS0 ต่อเข้ากับวงจรไมโครคอนโทรลเลอร์

ในการเชื่อมต่อเพื่อการตั้งค่าและอ่านค่าโมดูลเซ็นเซอร์ LSM9DS0 จะใช้รูปแบบการติดต่อสื่อสารสำหรับส่งและรับข้อมูล แบบ I^2C BUS (Inter Integrate Circuit Bus) โดยมีรูปแบบการเชื่อมต่อของขาไอซีไมโครคอนโทรลเลอร์เบอร์เอทีเมก้า 168 ขา PC4 (ขาที่27) สำหรับรับ-ส่งข้อมูล (SDA) และขา PC5 (ขาที่ 28) สำหรับส่งสัญญาณนาฬิกา (SCL) ออกจากไมโครคอนโทรลเลอร์ให้กับโมดูลเซ็นเซอร์ LSM9DS0 แสดงการเชื่อมต่อของวงจรดังรูปที่ 3.4



รูปที่ 3.4 วงจรการเชื่อมต่อไมโครคอนโทรลเลอร์กับโมดูลเซ็นเซอร์ LSM9DS0

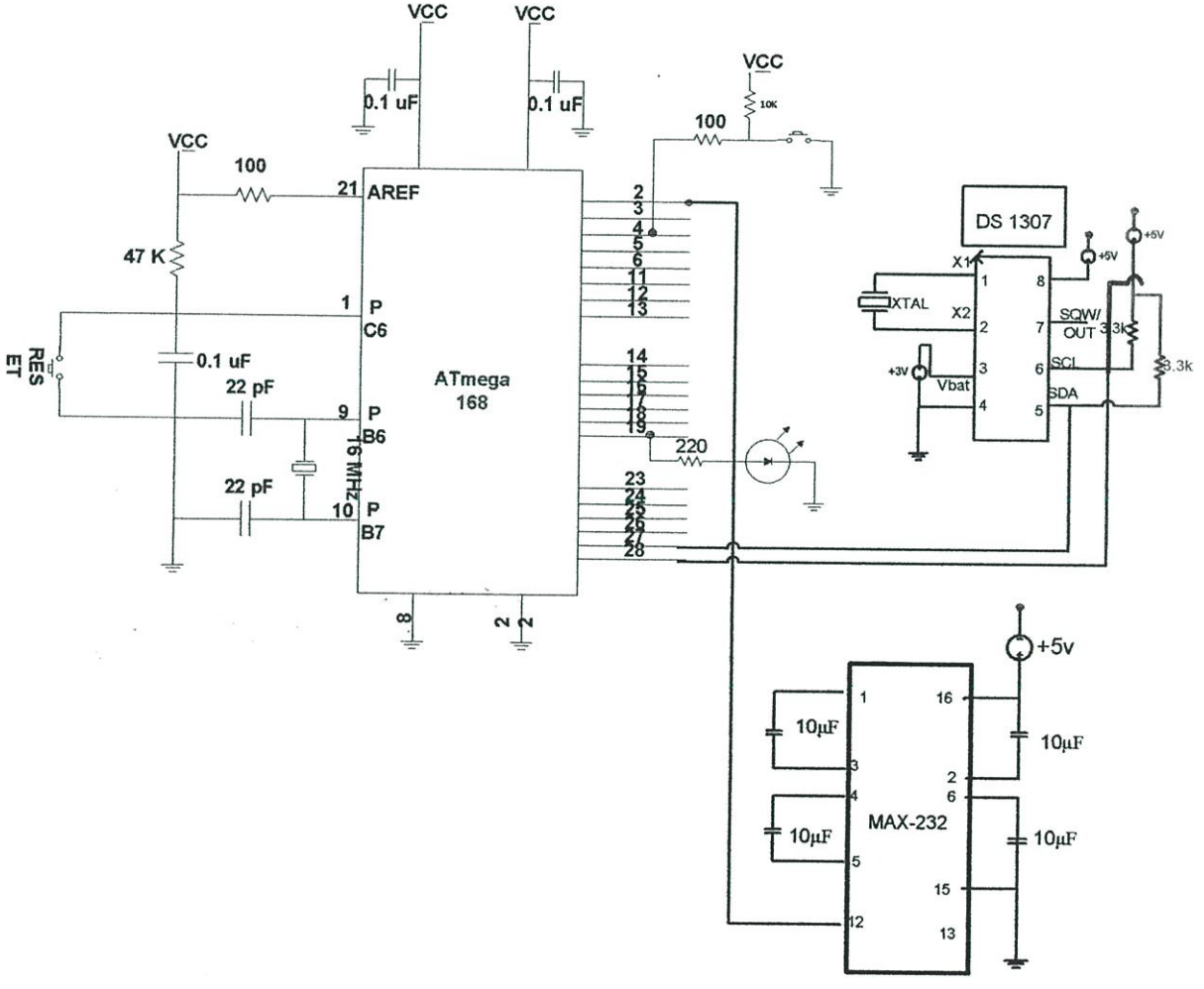
โดยมีโฟลว์ชาร์ตการทำงานของโปรแกรมของโมดูลเซ็นเซอร์ LSM9DS0 ต่อกับวงจรไมโครคอนโทรลเลอร์ แสดงดังรูปที่ 3.5



รูปที่ 3.5 โฟลว์ชาร์ตการทำงานของโปรแกรมของโมดูลเซ็นเซอร์ LSM9DS0 ต่อกับวงจรไมโครคอนโทรลเลอร์

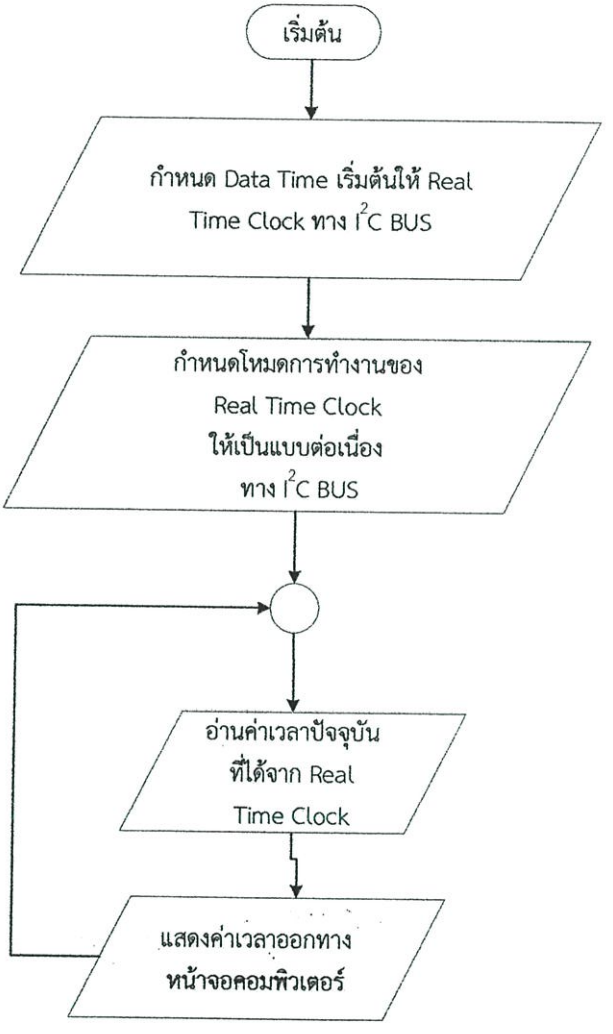
3.1.4 การออกแบบวงจรเรียลไทม์คล็อก (Real Time Clock) ต่อเข้ากับวงจรไมโครคอนโทรลเลอร์

รูปที่ 3.6 เป็นการต่อวงจรเรียลไทม์คล็อกเข้ากับวงจรไมโครคอนโทรลเลอร์โดยทำการต่อขา SCL,SDA ของเรียลไทม์คล็อกกับขาที่ 28,27 ของไมโครคอนโทรลเลอร์ตามลำดับ โดยวงจรเรียลไทม์คล็อกนี้จะทำหน้าที่เป็นฐานเวลาให้กับวงจรไมโครคอนโทรเลอร์ เพื่อใช้เป็นเวลาอ้างอิงให้กับเซ็นเซอร์ในขณะที่กำลังตรวจจับการเคลื่อนไหวของร่างกาย ณ เวลานั้น



รูปที่ 3.6 วงจรการเชื่อมต่อไมโครคอนโทรลเลอร์กับวงจรเรียลไทม์คล็อก

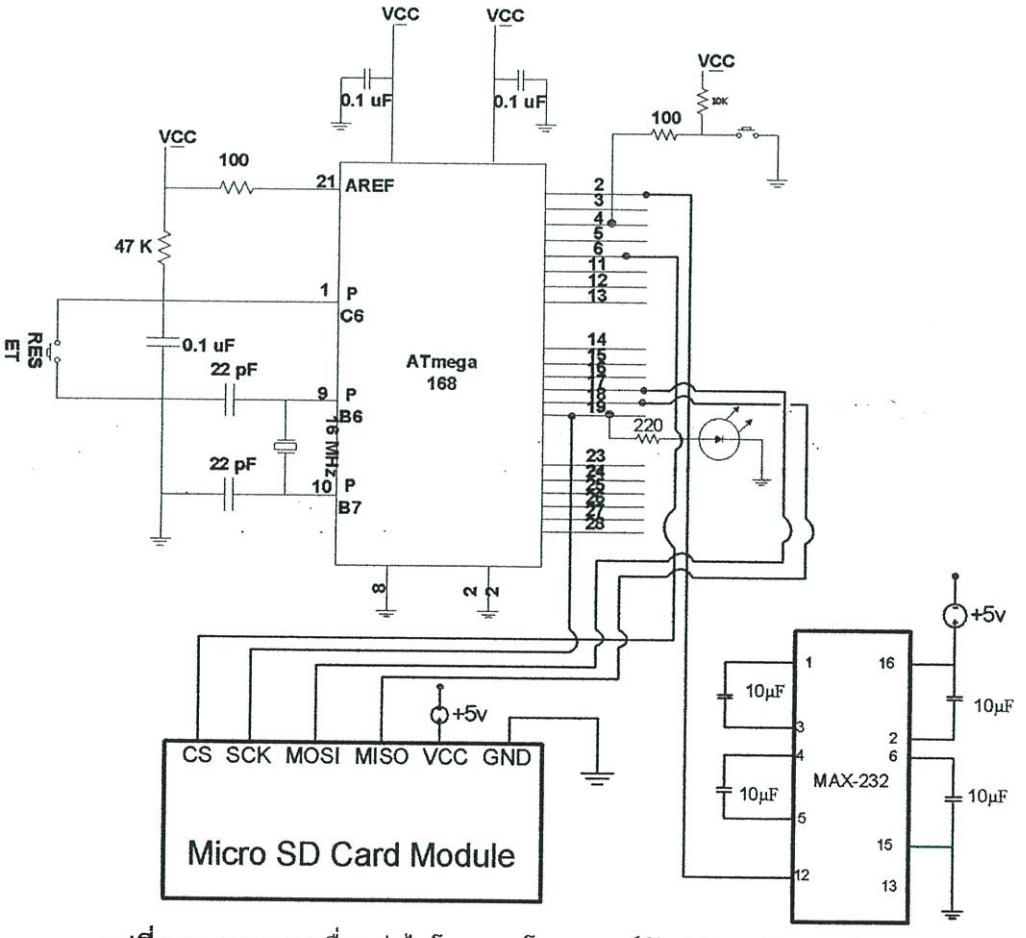
โดยมีโฟลว์ชาร์ตการทำงานของโปรแกรมของวงจรเรียลไทม์คล็อกต่อกับวงจรไมโครคอนโทรลเลอร์ แสดงดังรูปที่ 3.7



รูปที่ 3.7 โฟลว์ชาร์ตการทำงานของโปรแกรมของวงจรเรียลไทม์คล็อกต่อกับวงจรไมโครคอนโทรลเลอร์

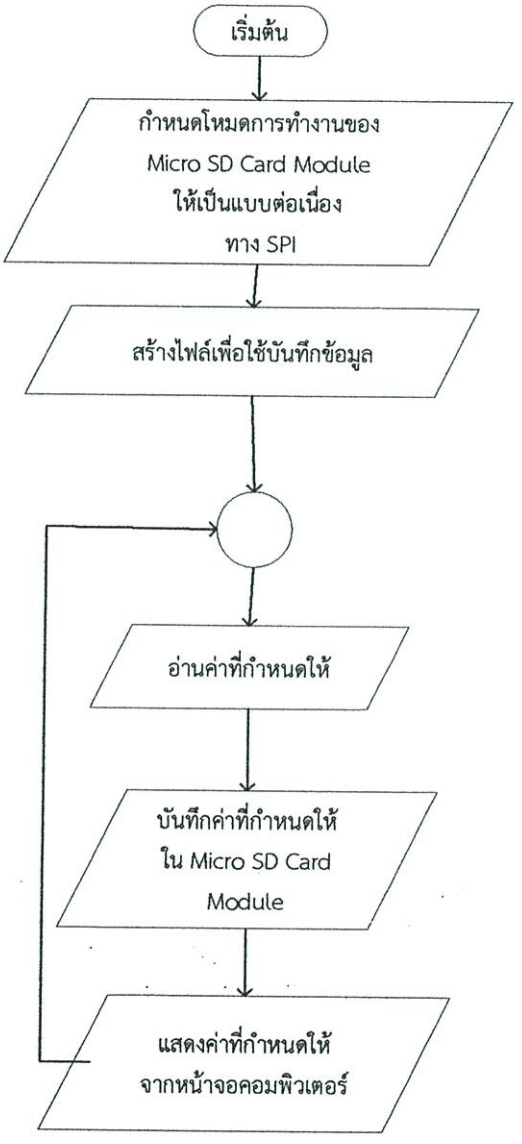
3.1.5 การออกแบบวงจร Micro SD Card Module ต่อเข้ากับวงจรไมโครคอนโทรลเลอร์

ในการเชื่อมต่อ Micro SD Card Module กับไมโครคอนโทรลเลอร์ จะใช้รูปแบบการติดต่อสื่อสารแบบ Serial Peripheral Protocol (SPI) โดย SPI จะส่งและรับข้อมูลทีละบิต (Bit Serial) และใช้สัญญาณนาฬิกา เป็นตัวกำหนดจังหวะการทำงาน ในการทำงานของอุปกรณ์ในระบบบัส แบ่งเป็น SPI Master และ SPI Slave มีรูปแบบการเชื่อมต่อของขาไอซีไมโครคอนโทรลเลอร์เบอร์เอทีเมก้า 168 โดยที่ CS ต่อขา PD4 (ขาที่6) SCK ต่อขา PB5 (ขาที่19) MOSI ต่อขา PB3 (ขาที่17) MISO ต่อขา PB4 (ขาที่18) โดยวงจร SD Card นี้จะทำหน้าที่เป็นอุปกรณ์เก็บข้อมูลของระบบ ดังรูปที่ 3.8



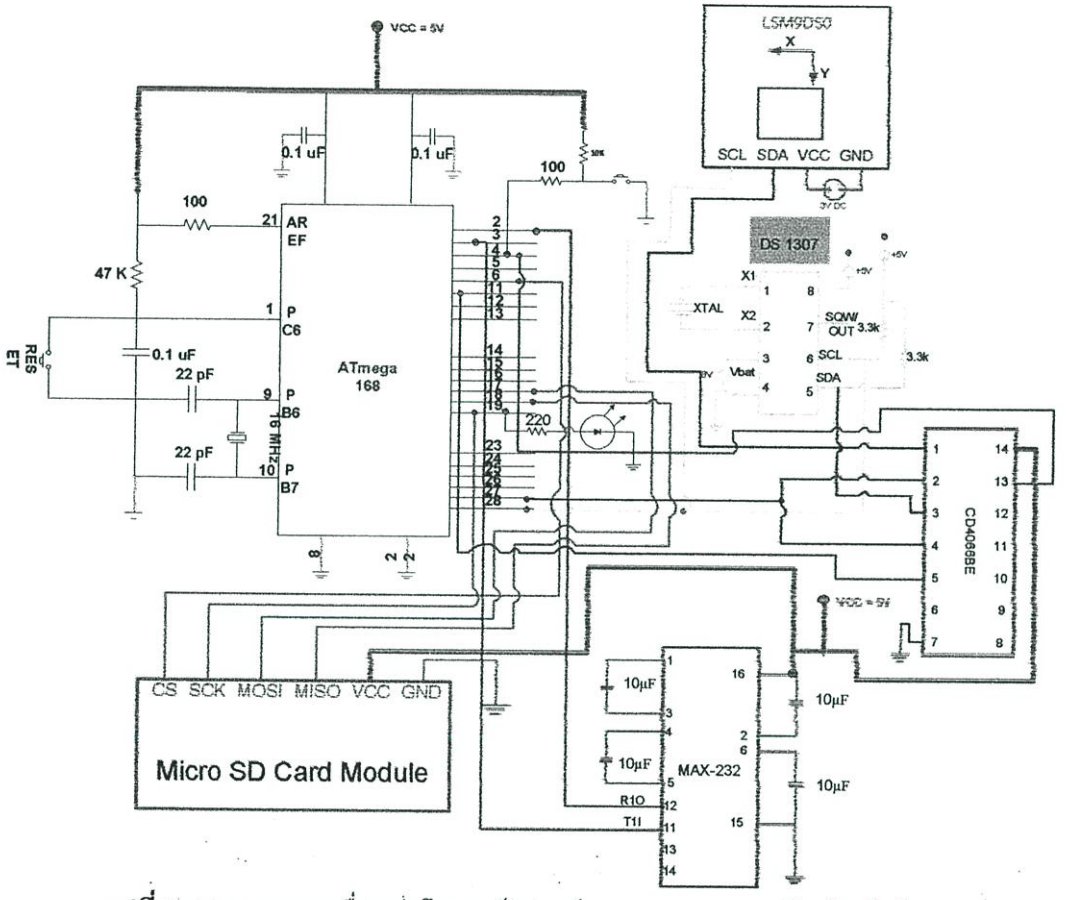
รูปที่ 3.8 วงจรการเชื่อมต่อไมโครคอนโทรลเลอร์กับ Micro SD Card Module

โดยมีโฟลว์ชาร์ตการทำงานของโปรแกรมของวงจรMicro SD Card Module ต่อกับวงจรไมโครคอนโทรลเลอร์ แสดงดังรูปที่ 3.9



รูปที่ 3.9 โฟลว์ชาร์ตการทำงานของโปรแกรมของวงจร Micro SD Card Module ต่อกับวงจรไมโครคอนโทรลเลอร์

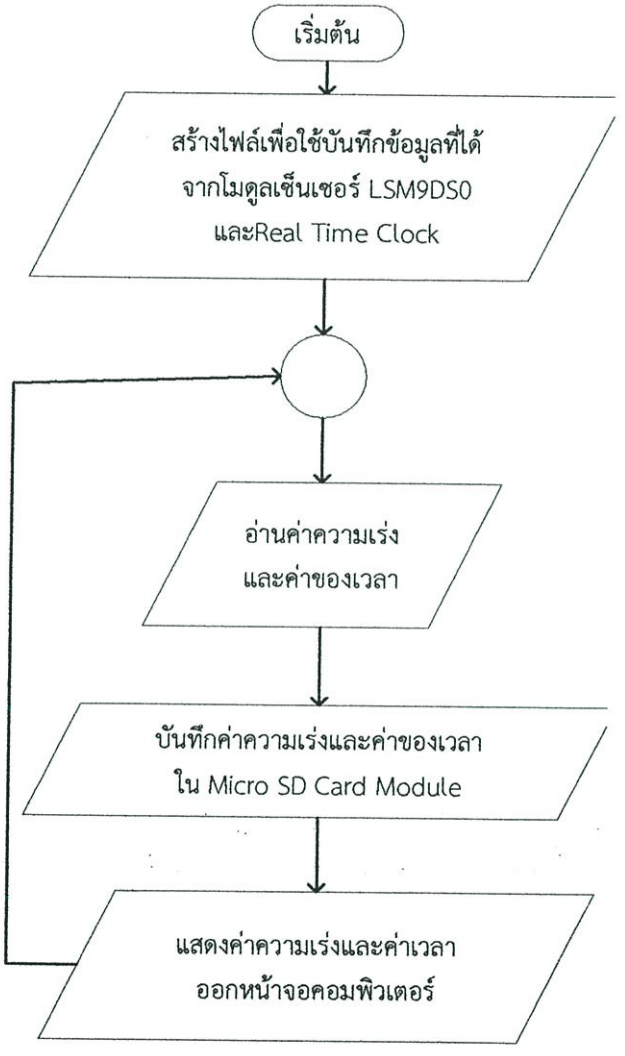
3.1.6 การออกแบบวงจรโมดูลเซ็นเซอร์ LSM9DS0 วงจรเรียลไทม์คล็อก (Real Time Clock) และวงจร Micro SD Card Module ต่อเข้ากับวงจรไมโครคอนโทรลเลอร์



รูปที่ 3.10 วงจรการเชื่อมต่อโมดูลเซ็นเซอร์ LSM9DS0 วงจรเรียลไทม์คล็อก และวงจร Micro SD Card Module ต่อเข้ากับไมโครคอนโทรลเลอร์

รูปที่ 3.10 สามารถต่อวงจรโมดูลเซ็นเซอร์ LSM9DS0 วงจรเรียลไทม์คล็อกและ Micro SD Card Module เข้ากับวงจรไมโครคอนโทรลเลอร์ได้โดยต่อขา SCL ของวงจรโมดูลเซ็นเซอร์ LSM9DS0 และวงจรเรียลไทม์คล็อก เข้ากับขา 28 ของวงจรไมโครคอนโทรลเลอร์ และต่อขา SDA ของโมดูลเซ็นเซอร์ LSM9DS0 และวงจรเรียลไทม์คล็อกเข้ากับขา 1,8 ของไอซี CD4066B ตามลำดับ โดยมีขา 13(Control A) และขา 6(Control C) เป็นขาควบคุมการรับส่งข้อมูล ผ่านทางขา 4(PD2) และ 11(PD5) ของวงจรไมโครคอนโทรลเลอร์ตามลำดับ แล้วต่อขา CS, SCK, MOSI,

MIOS ของ Micro SD Card Module เข้ากับขา 6, 19, 17, 18 ของไมโครคอนโทรลเลอร์โดยมีโฟลว์ชาร์ตการทำงานของโปรแกรมของวงจรโมดูลเซ็นเซอร์ LSM9DS0 วงจรเรียลไทม์คล็อก และวงจร Micro SD Card Module ต่อกับวงจรไมโครคอนโทรลเลอร์ แสดงดังรูปที่ 3.11



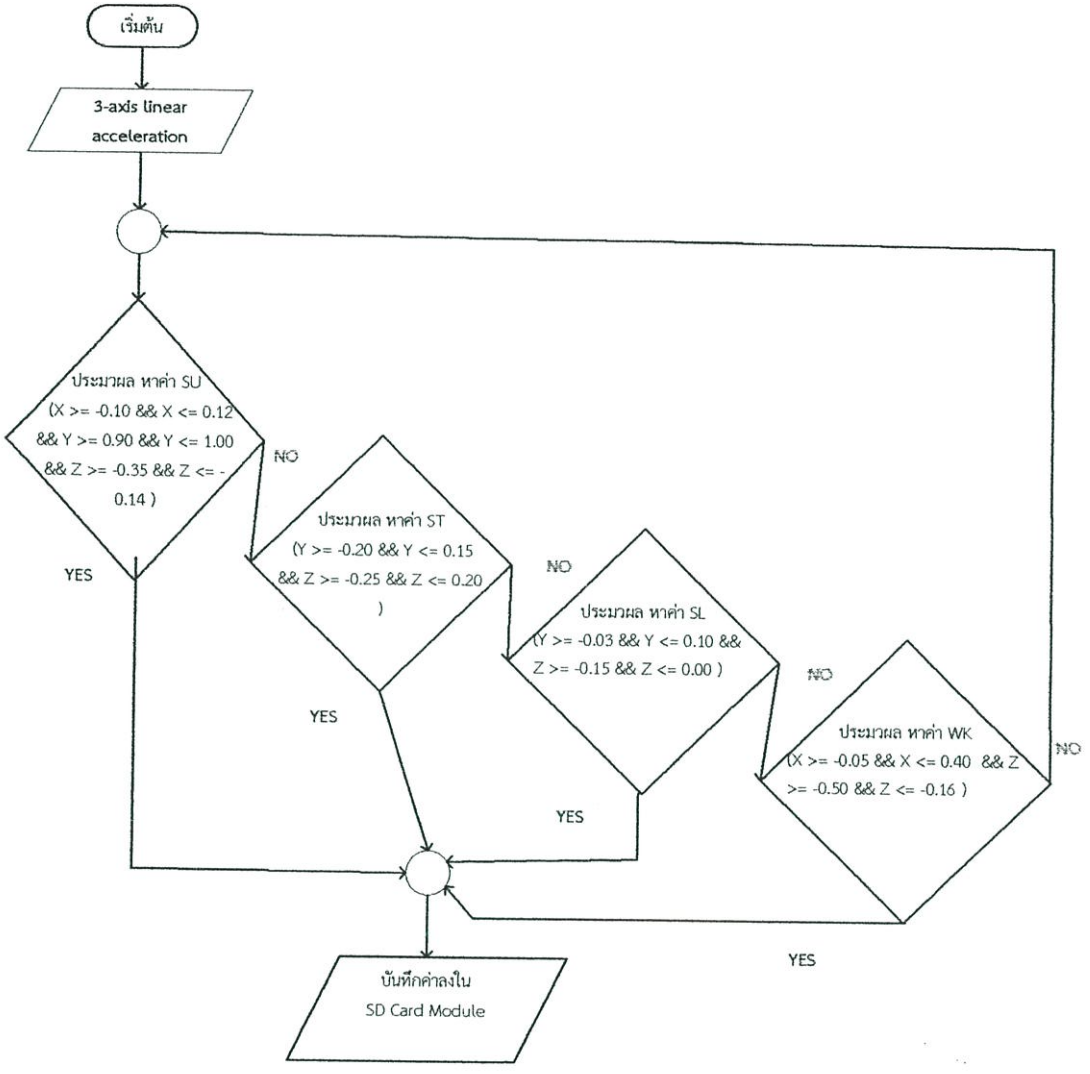
รูปที่ 3.11 โฟลว์ชาร์ตการทำงานของโปรแกรมของวงจรโมดูลเซ็นเซอร์ LSM9DS0 วงจรเรียลไทม์คล็อก **และวงจร** Micro SD Card Module ต่อกับวงจรไมโครคอนโทรลเลอร์

3.1.7 การออกแบบฟังก์ชั่นการประมวลผลหาลักษณะการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน

การประมวลผลหาลักษณะการเคลื่อนไหวของร่างกายโดยใช้โปรแกรม อาร์ดูอิโน่ (Arduino) โดยลักษณะการเคลื่อนไหวของร่างกายมี 4 แบบคือ ยืน (standing: SU) นั่ง (Sitting: ST) นอน (Sleeping: SL) และเดิน (walking: WK) ข้อมูลแต่ละลักษณะการเคลื่อนไหวของร่างกายที่ได้จากเซนเซอร์นั้นจะมีช่วงความเร่งในแกน X แกน Y และแกน Z ที่แตกต่างกันดังตารางที่ 3.1 และมีโฟลว์ชาร์ตแสดงการทำงานของฟังก์ชั่นการประมวลผลหาลักษณะการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน แสดงดังรูปที่ 3.12

ตารางที่ 3.1 สรุปค่าความเร่งเชิงเส้น 3 แกน ในลักษณะการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน ของผู้ทดลองทั้ง 5 คน

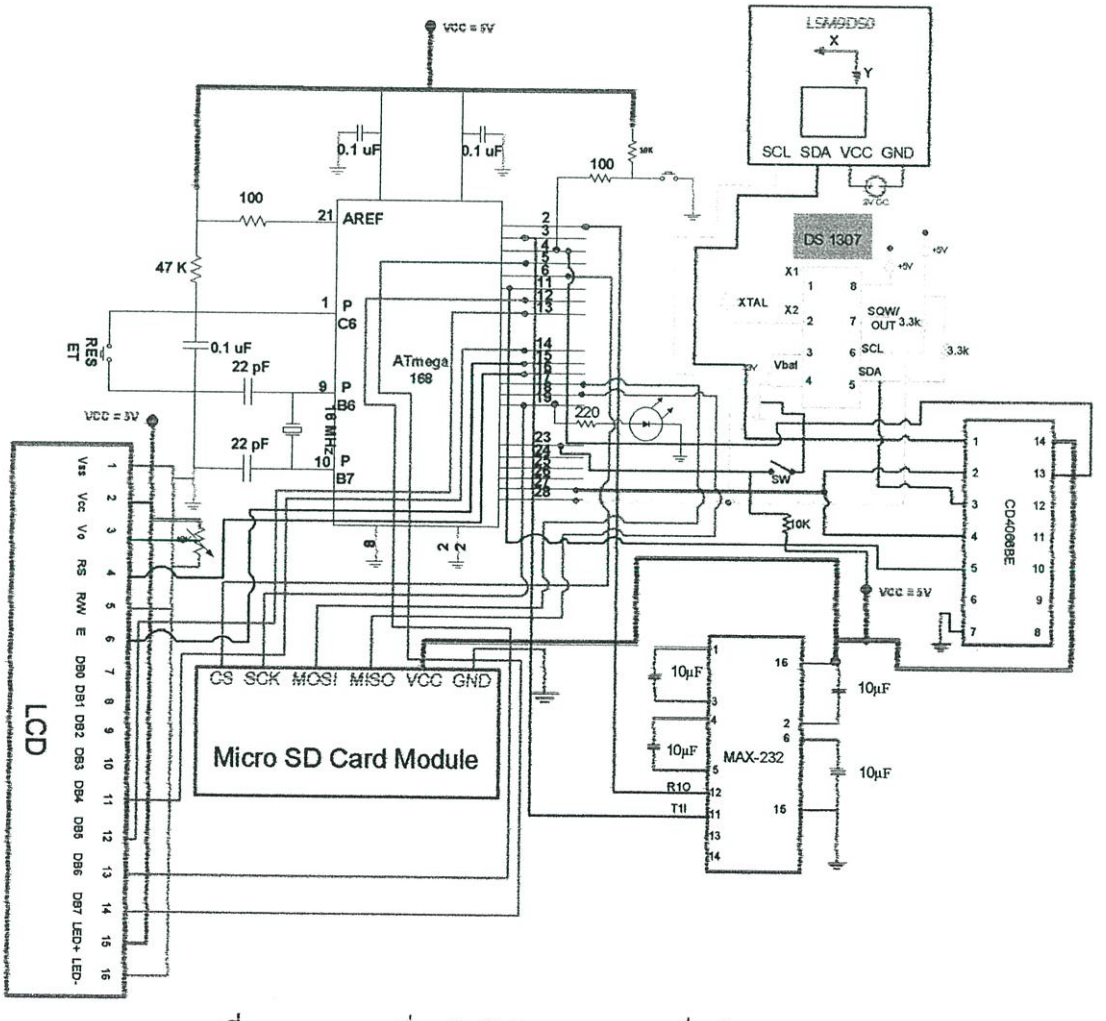
ค่าความเร่งในแต่ละแนวแกน	ลักษณะการเคลื่อนไหวของร่างกาย			
	ยืน (SU)	นั่ง (ST)	นอน (SL)	เดิน (WK)
แกน X (m/s^2)	-0.15 ⟷ 0.05	0.72 ⟷ 0.95	0.80 ⟷ 0.90	-0.30 ⟷ 0.40
แกน Y (m/s^2)	0.90 ⟷ 1.05	0.27 ⟷ 0.45	-0.15 ⟷ 0.05	0.80 ⟷ 1.10
แกน Z (m/s^2)	-0.35 ⟷ -0.15	-0.20 ⟷ 0.10	-0.20 ⟷ 0.15	-0.50 ⟷ 0.00



รูปที่ 3.12 โฟลว์ชาร์ตแสดงการทำงานของฟังก์ชันในการประมวลผลหาลักษณะการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน

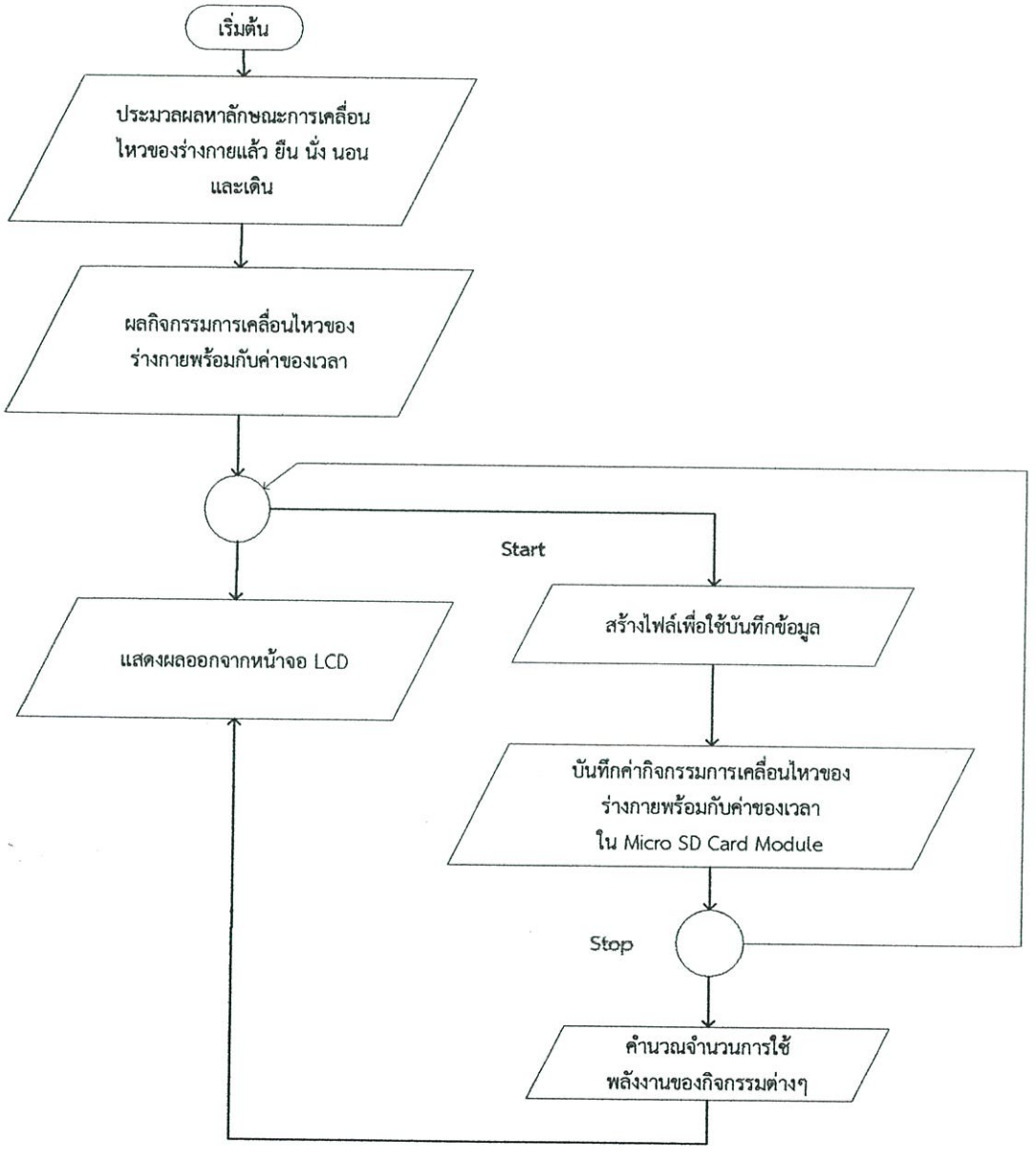
3.1.8 การออกแบบวงจรเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย

ในการเชื่อมต่อวงจรเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย ประกอบด้วย วงจรไมโครคอนโทรลเลอร์ วงจรโมดูลเซ็นเซอร์ LSM9DS0 วงจรเรียลไทม์คล็อก วงจร Micro SD Card Module และวงจรจอ LCD ในการทำงานของวงจรเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย จะแสดงผลกิจกรรมการเคลื่อนไหวของร่างกาย ณ เวลาปัจจุบัน บนหน้าจอ LCD ที่ติดกับตัวเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย และเมื่อกดปุ่ม Start เครื่องจะทำการบันทึกผลของกิจกรรมการเคลื่อนไหวของร่างกายกับค่าของเวลา ลงใน SD Card และเมื่อกดปุ่ม Stop เครื่องจะทำการคำนวณอัตราการใช้พลังงานของร่างกายจากกิจกรรมที่ทำไปตั้งแต่ที่เริ่มทำการบันทึก บนหน้าจอ LCD ดังรูปที่ 3.13



รูปที่ 3.13 วงจรเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย

โดยมีโฟลว์ชาร์ตการทำงานของโปรแกรมของวงจรเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย แสดงดังรูปที่ 3.14



รูปที่ 3.14 โฟลว์ชาร์ตการทำงานของโปรแกรมของวงจรเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย

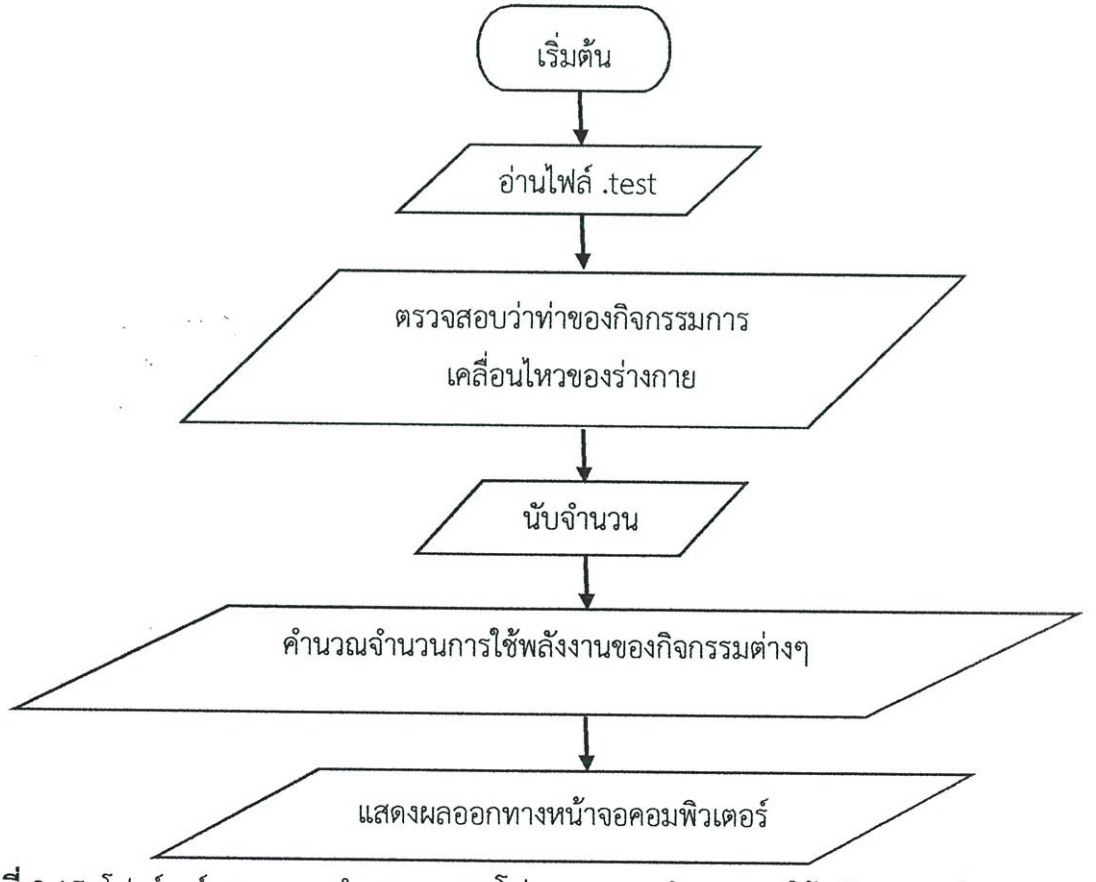
3.1.9 การออกแบบโปรแกรมการคำนวณการใช้พลังงานของร่างกาย

ค่าพลังงานที่ใช้สำหรับทำกิจกรรม (ที่ใช้ในการทำปริญญานิพนธ์)

ตารางที่ 3.2 จำนวนพลังงานที่ใช้ในการทำกิจกรรมต่างๆ

ชนิดของกิจกรรม	จำนวนพลังงานที่ต้องการต่อชั่วโมง		จำนวนพลังงานที่ต้องการต่อวินาที	
	กิโลจูล	กิโลแคลอรี	กิโลจูล	กิโลแคลอรี
ยืน	586	140	0.1628	0.0389
นั่ง	63	15	0.0175	0.0042
นอน	314	75	0.0872	0.0208
เดิน	1255	300	0.3486	0.0833

โดยมีโฟลว์ชาร์ตแสดงการทำงานของโปรแกรมการคำนวณการใช้พลังงานของร่างกาย แสดงดังรูปที่ 3.15



รูปที่ 3.15 โฟลว์ชาร์ตแสดงการทำงานของของโปรแกรมการคำนวณการใช้พลังงานของร่างกาย

3.9.1.1 คำนวณอัตราการใช้พลังงานในภาวะร่างกายปกติของผู้ทำการทดลอง ใน 1 วัน ดังสมการที่ (3-1) [11]

$$BMR = 66.5 + (13.7 \times W) + (5.0 \times H) - (6.8 \times A) \quad (3\text{-}1)$$

เมื่อ W = น้ำหนักตัว 77 กิโลกรัม

H = ความสูง 175 เซนติเมตร

A = อายุ 23 ปี

จากสมการ (3-1) น้ำหนัก ส่วนสูงและอายุมีผลต่อการเผาผลาญพลังงาน เมื่อหาค่า BMR (Basal Metabolic Rate) ทำให้รู้ได้ว่ามีการการเผาผลาญพลังงานโดยไม่ทำกิจกรรมอะไรเลย และหากมีกิจกรรมออกกำลังกายจะมีการเผาผลาญพลังงานโดยคำนวณได้ดังนี้

"การเผาผลาญพลังงานโดยปกติ = BMR x ตัวแปร" (3-2)

โดยตัวแปรจะขึ้นอยู่กับการออกกำลังของร่างกาย ดังนี้

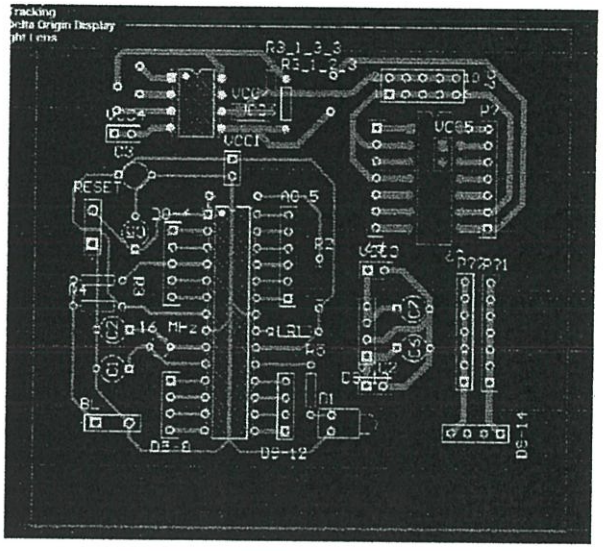
- นั่งทำงานอยู่กับที่ และไม่ได้ออกกำลังกายเลย = BMR x 1.2
- ออกกำลังกายหรือเล่นกีฬาเล็กน้อย ประมาณอาทิตย์ละ 1-3 วัน = BMR x 1.375
- ออกกำลังกายหรือเล่นกีฬาปานกลาง ประมาณอาทิตย์ละ 3-5 วัน = BMR x 1.55
- ออกกำลังกายหรือเล่นกีฬาอย่างหนัก ประมาณอาทิตย์ละ 6-7 วัน = BMR x 1.725
- ออกกำลังกายหรือเล่นกีฬาอย่างหนักทุกวันเช้าเย็น = BMR x 1.9

ดังนั้นจะได้ค่าอัตราการใช้พลังงานในภาวะร่างกายปกติของผู้ทำการทดลอง ใน 1 วัน ดังสมการที่ (3-3)

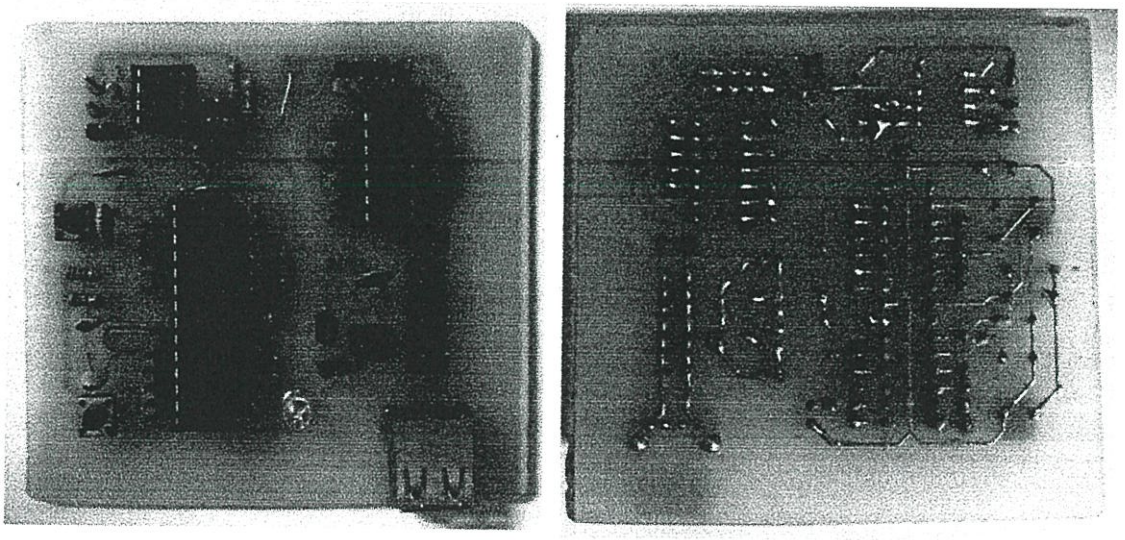
$$BMR = [66.5 + (13.7 \times 77) + (5.0 \times 175) - (6.8 \times 23)] \times 1.2 = 2207 \text{ กิโลแคลอรี่} \quad (3\text{-}3)$$

3.2 เครื่องมือที่ใช้ในการทดลอง

3.2.1 วงจรรวมของระบบทั้งหมด ประกอบด้วยวงจรไมโครคอนโทรลเลอร์, วงจรเรียลไทม์คล็อก, วงจรแปลงแรงดันกระแสตรง และวงจรสวิทช์



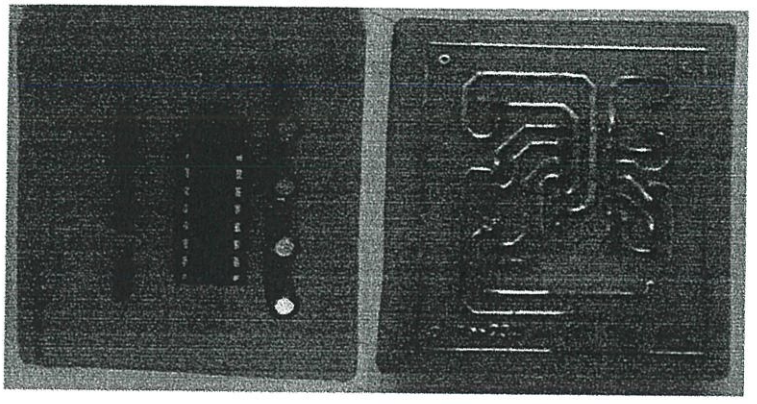
รูปที่ 3.16 PCB ของวงจรรวมของระบบทั้งหมด



ก) ด้านหน้าของวงจร ข) ด้านหลังของวงจร

รูปที่ 3.17 วงจรรวมของระบบทั้งหมด

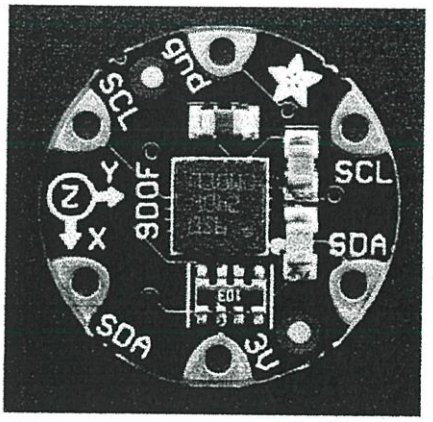
3.2.2 วงจรแปลงระดับสัญญาณโดยใช้แม็กซ์ 232



ก) ด้านหน้าของวงจร ข) ด้านหลังของวงจร

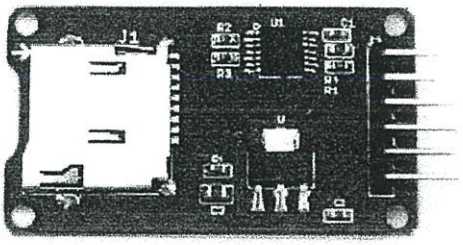
รูปที่ 3.18 วงจรแปลงระดับสัญญาณโดยใช้แม็กซ์ 232

3.2.3 โมดูลเซ็นเซอร์ LSM9DS0



รูปที่ 3.19 โมดูลเซ็นเซอร์ LSM9DS0

3.2.4 Micro SD Card Module

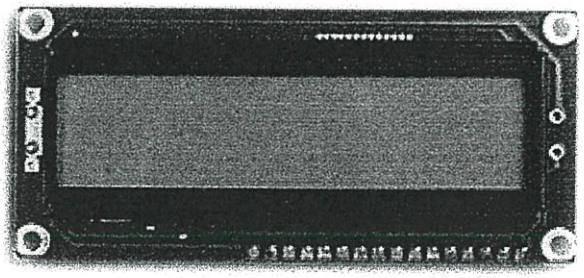


รูปที่ 3.20 Micro SD Card Module

3.2.5 Micro SDHC

รูปที่ 3.21 Micro SDHC

3.2.6 จอ LCD



รูปที่ 3.22 จอ LCD

3.3 การจัดเก็บผลการทดลอง

ในส่วนการจัดเก็บผลการทดลองนั้นจะแบ่งเป็น 5 ส่วนหลักๆซึ่งจะประกอบไปด้วย

1) การเก็บผลค่าความเร่งเชิงเส้น 3 แกน จากโมดูลเซ็นเซอร์ LSM9DS0

การอ่านค่าจากโมดูลเซ็นเซอร์ LSM9DS0

- การกำหนดโหมดในการทำงานของโมดูลเซ็นเซอร์ LSM9DS0
- การอ่านค่าความเร่งเชิงเส้น 3 แกน ในแต่ละแกนอ้างอิงของโมดูลเซ็นเซอร์ LSM9DS0

2) การอ่านค่าของเวลาจากวงจรเรียลไทม์คล็อก (Real Time Clock)

3) การเก็บผลรวมของค่าความเร่งเชิงเส้น 3 แกน กับค่าของเวลา ลงใน Micro SD Card Module

4) การเก็บผลกิจกรรมการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน ลงใน Micro SD Card Module

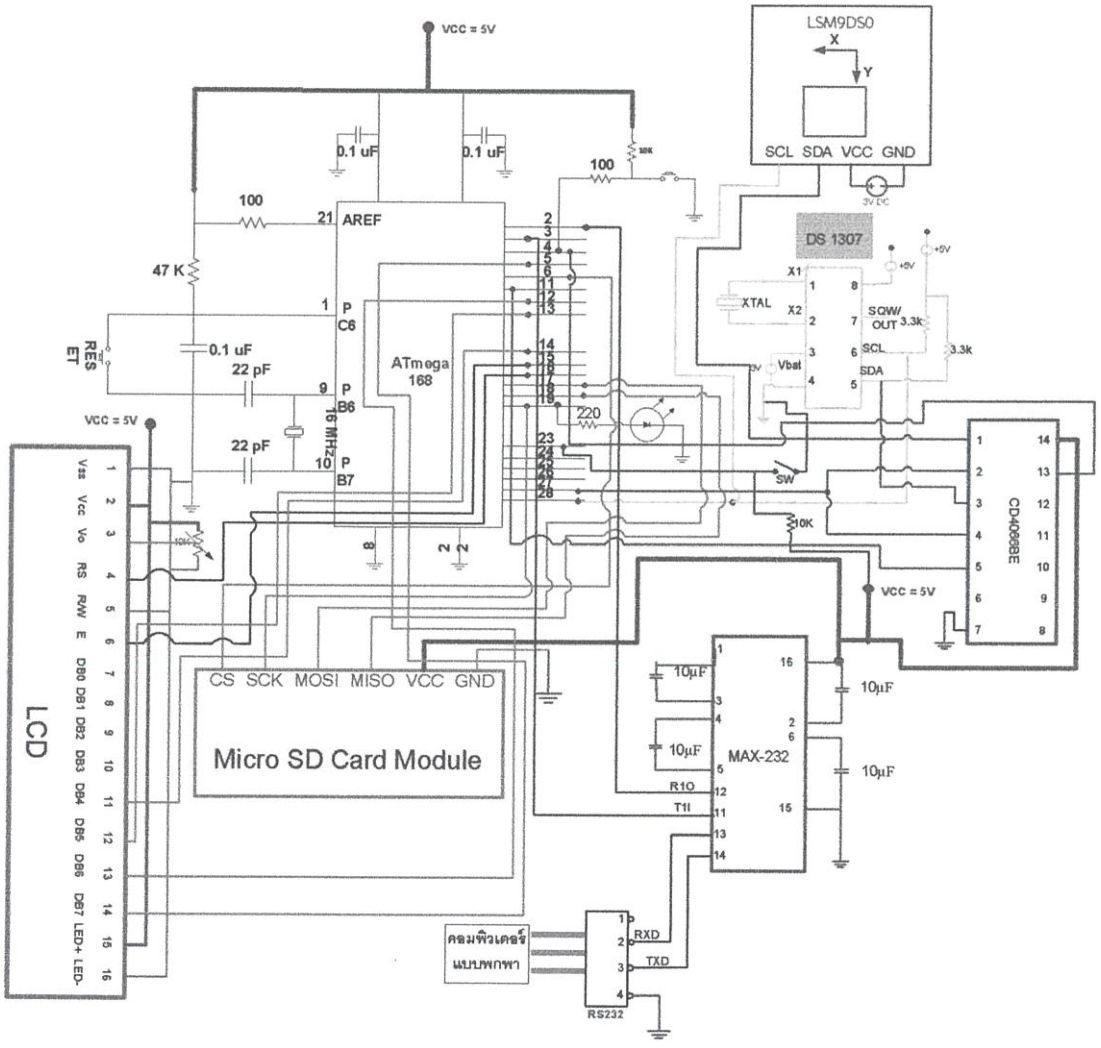
5) คำนวณการใช้พลังงานจากกิจกรรมการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน ใน 1 วัน

บทที่ 4

ผลการทดลอง

4.1 วงจรรวมของการทดลอง

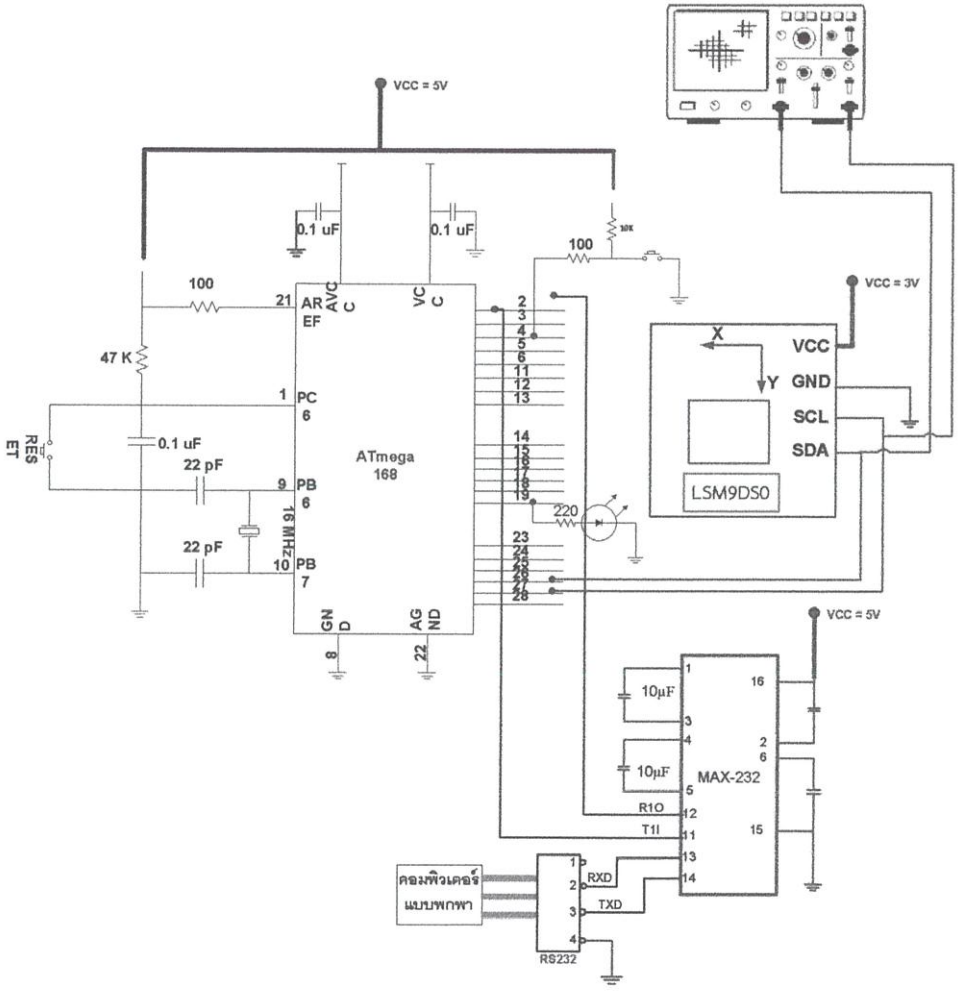
เครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย มีโครงสร้าง และการต่อวงจร โดยรวมทั้งหมดของเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย จะแสดงได้ดังรูปที่ 4.1



รูปที่ 4.1 การต่อวงจรรวมทั้งหมดของเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย

4.2 การอ่านค่าจากโมดูลเซ็นเซอร์ LSM9DS0

การอ่านค่าจากโมดูลเซ็นเซอร์ LSM9DS0 จะอาศัยการรับ-ส่งข้อมูล โดยใช้ I^2C BUS (Inter Integrate Circuit Bus) ในการติดต่อกับโมดูลเซ็นเซอร์ LSM9DS0 เพื่ออ่านค่า ทำได้โดยการใช้ตัวไมโครคอนโทรลเลอร์เบอร์เอทีเมก้า 168 เขียนโปรแกรมให้กับตัวไมโครคอนโทรลเลอร์ โดยใช้โปรแกรมอาร์ดูอิโน่ โปรแกรมชุดคำสั่งให้กับตัวไมโครคอนโทรลเลอร์ การทดลองมีการต่ออุปกรณ์โดยรวมในการทดลองในหัวข้อนี้ แสดงดังรูปที่ 4.2



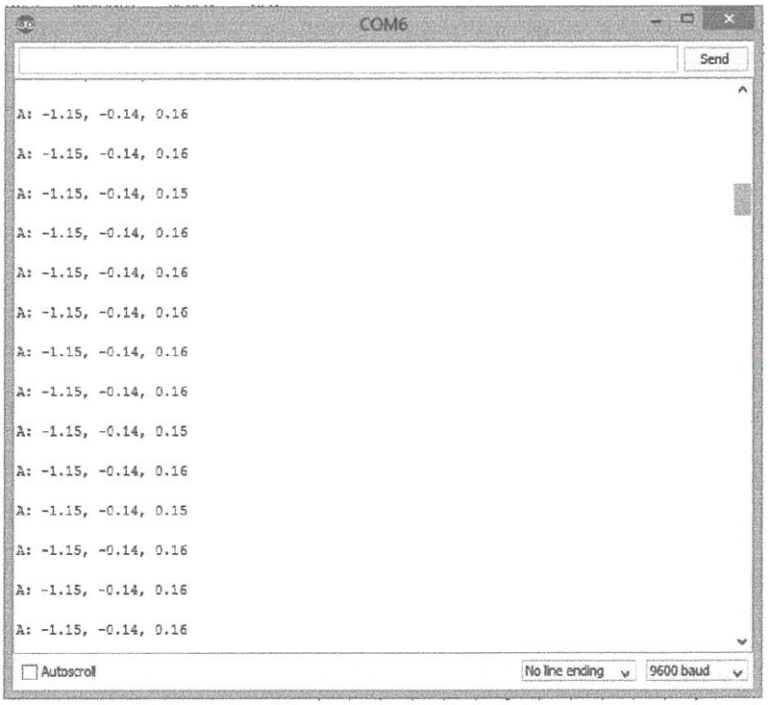
รูปที่ 4.2 การต่อวงจรเพื่อวัดผลในการอ่านค่าจากโมดูลเซ็นเซอร์ LSM9DS0

ในการเขียนโปรแกรมจะใช้ Libraly Wire.h ในการเขียนโปรแกรม โดยจะมีการกำหนดให้ ขา SCL ต่อเข้ากับขา analog 5 (ขา 28) ของตัวไมโครคอนโทรลเลอร์ และขา SDA ต่อเข้ากับขา analog 4 (ขา 27) ของตัวไมโครคอนโทรลเลอร์ โดยโฟลว์ชาร์ตการทำงานของโปรแกรมแสดงในข้อหัวที่ 3.1.3 เมื่อทำการวัดสัญญาณต่างๆ จะได้ผลการดำเนินการทั้งหมด ดังนี้

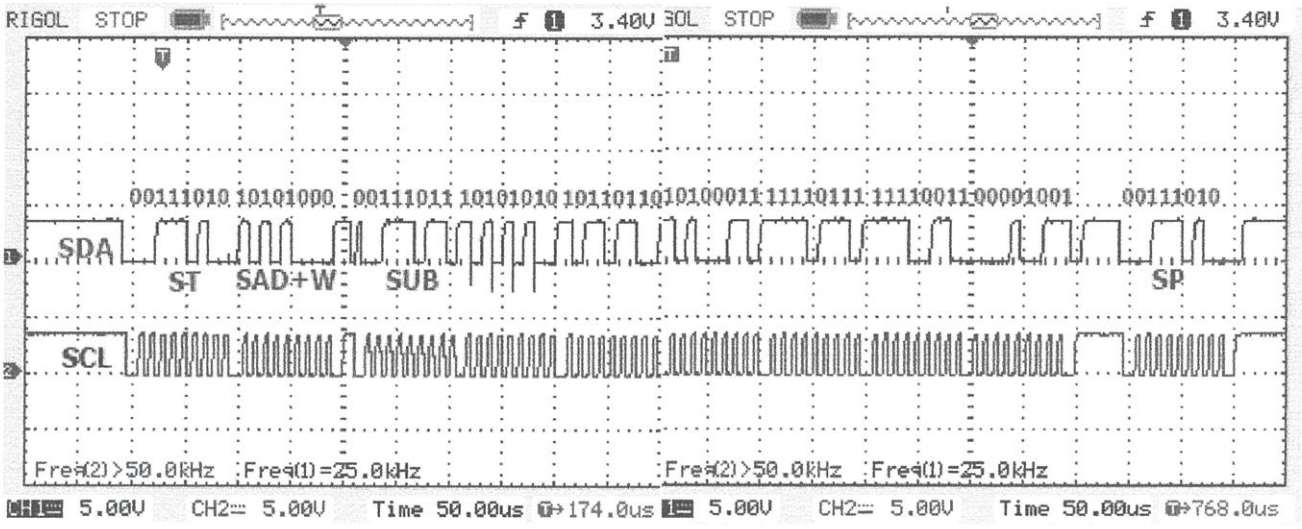
4.2.1 อ่านค่าสัญญาณของโมดูลเซ็นเซอร์ LSM9DS0 โดยอ่านค่าความเร่งเชิงเส้น 3 แกน

ในการกำหนดค่าโหมดการทำงานให้กับโมดูลเซ็นเซอร์ LSM9DS0 จะทำการกำหนดให้โมดูลเซ็นเซอร์ LSM9DS0 ทำงานใน Continuous Mode โดยเมื่อทำการวัดสัญญาณจะได้ผลการทดลองดังนี้

เขียนข้อมูลในส่วนของ Control Byte ให้กับโมดูลเซ็นเซอร์ LSM9DS0 โดยให้อ่านค่าความเร่งเชิงเส้น 3 แกน และส่งข้อมูลไปทาง I^2C BUS โดยจะมีผลค่าความเร่งเชิงเส้น 3 แกน แสดงดังรูปที่ 4.3 และลักษณะของสัญญาณ ดังรูปที่ 4.4



รูปที่ 4.3 หน้าจอคอมพิวเตอร์แสดงผลค่าความเร่งเชิงเส้น 3 แกน (linear acceleration) โดยค่าความเร่งแกน X = -1.15 Y = -0.14 และ Z = -0.16 m/s^2 ตามลำดับ



รูปที่ 4.4 สัญญาณพัลส์ที่วัดได้จากขา SDA กับ SCL
ในส่วนของ START (ST), SAD + W, Sub-Address (SUB) , ข้อมูล(DATA) , STOP (ST)

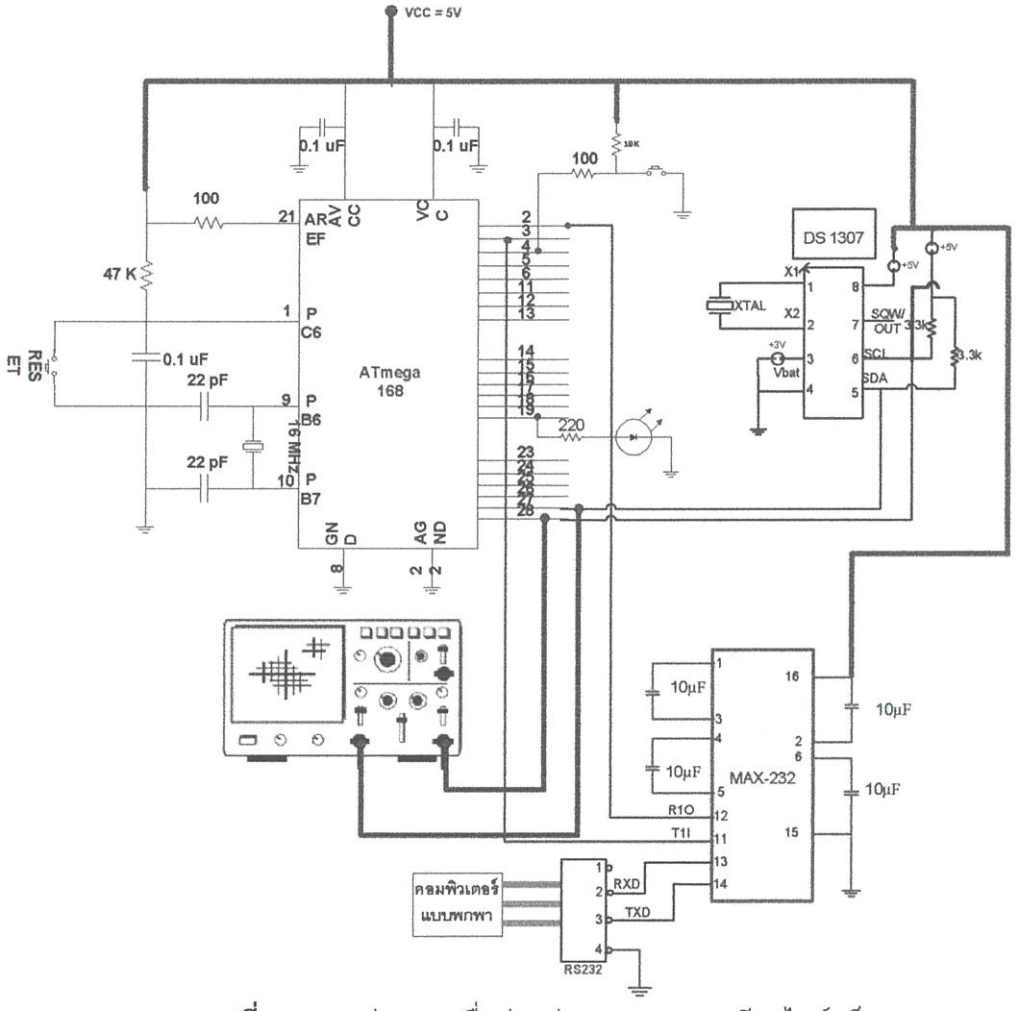
จากรูปที่ 4.4 แสดงสัญญาณพัลส์ของบิตข้อมูลที่วัดได้จากขา SCL และ SDA ของโมดูลเซ็นเซอร์ LSM9DS0 จะเห็นว่าจากสัญญาณพัลส์ที่วัดได้จากขา SDA คือสัญญาณพัลส์แสดงค่าความเร่งเชิงเส้น 3 แกน ส่วนสัญญาณพัลส์จากขา SCL จะแสดงสัญญาณนาฬิกาของแต่ละไบต์ มีทั้งหมด 9 ไบต์ซึ่งในรูปที่ 4.4 ที่ขา SDA จะสามารถอ่านค่าออกมาได้เป็นความเร่งเชิงเส้น 3 แกน คือ แกน X = -1.15 Y = -0.14 และ Z = -0.16 m/s^2 ตามลำดับ

4.3 การอ่านค่าของเวลาจากวงจรเรียลไทม์คล็อก (Real Time Clock)

การอ่านค่าเวลาจากวงจรเรียลไทม์คล็อก โดยที่ไอซี DS1307 มีการรับส่งข้อมูลแบบ I^2C ซึ่งเป็นแบบ 2 wire สามารถสื่อสารได้ 2 ทิศทาง (bi-direction bus) ฐานเวลาของ DS1307 นั้นสามารถเก็บข้อมูล วินาที, นาที, ชั่วโมง, วัน, วันที่, เดือน และปี ได้ ระบบเวลาสามารถทำงานโหมดรูปแบบ 24 ชั่วโมง หรือ 12 ชั่วโมง AM/PM ก็ได้ ภายมีระบบตรวจจับแหล่งจ่ายไฟ โดยถ้าแหล่งจ่ายไฟหลักถูกตัดไป DS1307 สามารถสวิตซ์ไปใช้ไฟจากแบตเตอรี่ และทำงานต่อไป

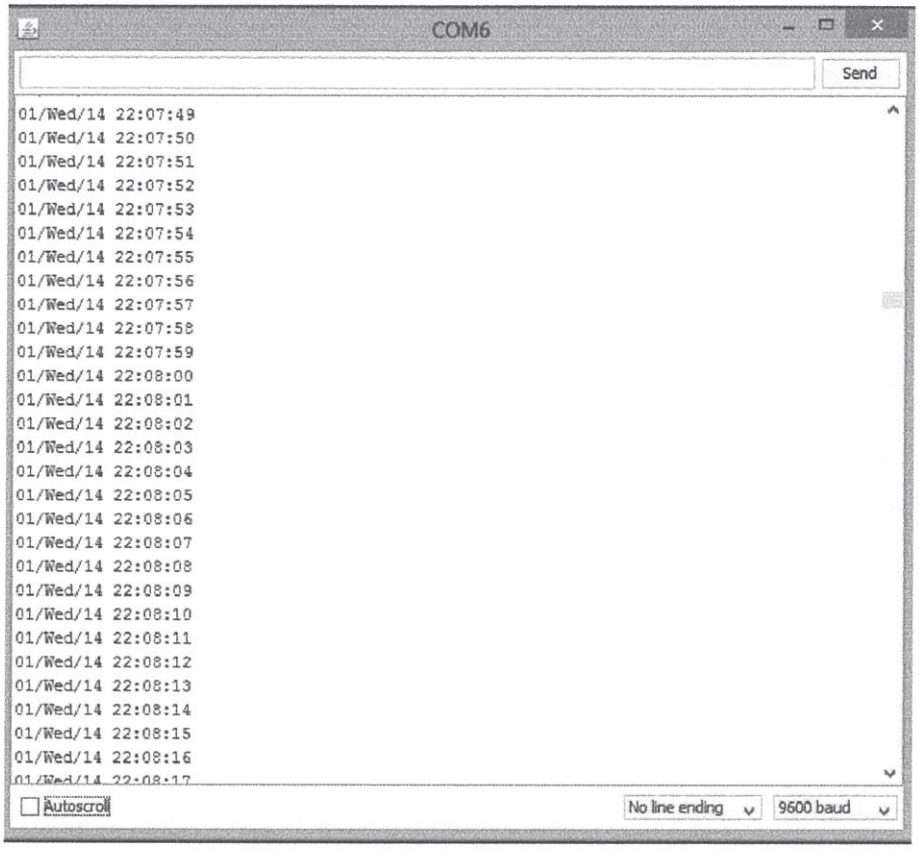
ในการติดต่อกับวงจรเรียลไทม์คล็อกเพื่ออ่านค่า ทำได้โดยการใช้ตัวไมโครคอนโทรลเลอร์เบอร์เอทีเมก้า 168 เขียนโปรแกรมให้กับตัวไมโครคอนโทรลเลอร์ โดยใช้

โปรแกรมอาร์ดูอิโน่ โปรแกรมชุดคำสั่งให้กับตัวไมโครคอนโทรลเลอร์ โดยจะโฟลว์ชาร์ตการทำงานของโปรแกรมในข้อหัวที่ 3.1.4 การทดลองมีการต่ออุปกรณ์โดยรวมในการทดลองในหัวข้อนี้ แสดงดังรูปที่ 4.5

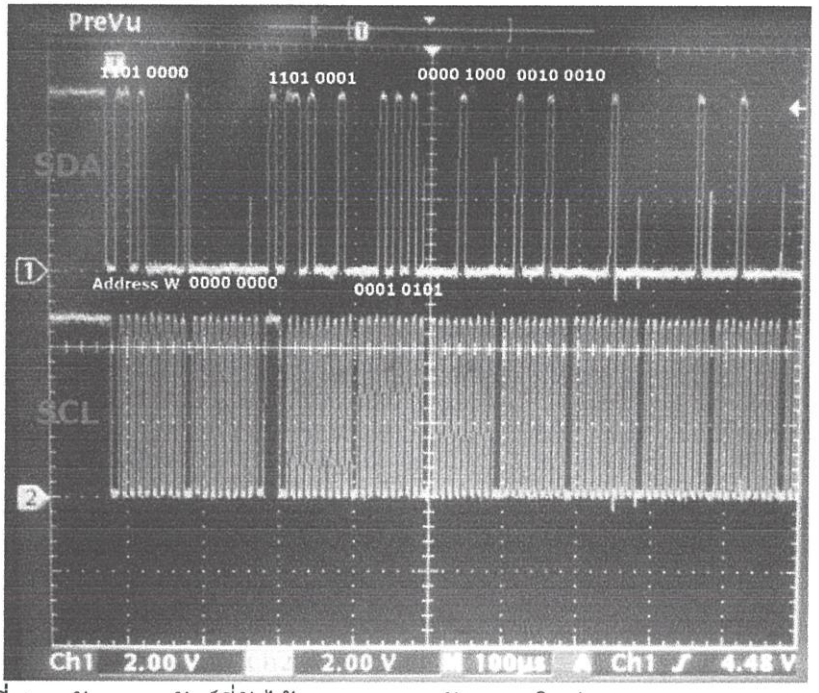


รูปที่ 4.5 การต่อวงจรเพื่ออ่านค่าเวลาจากวงจรเรียลไทม์คล็อก

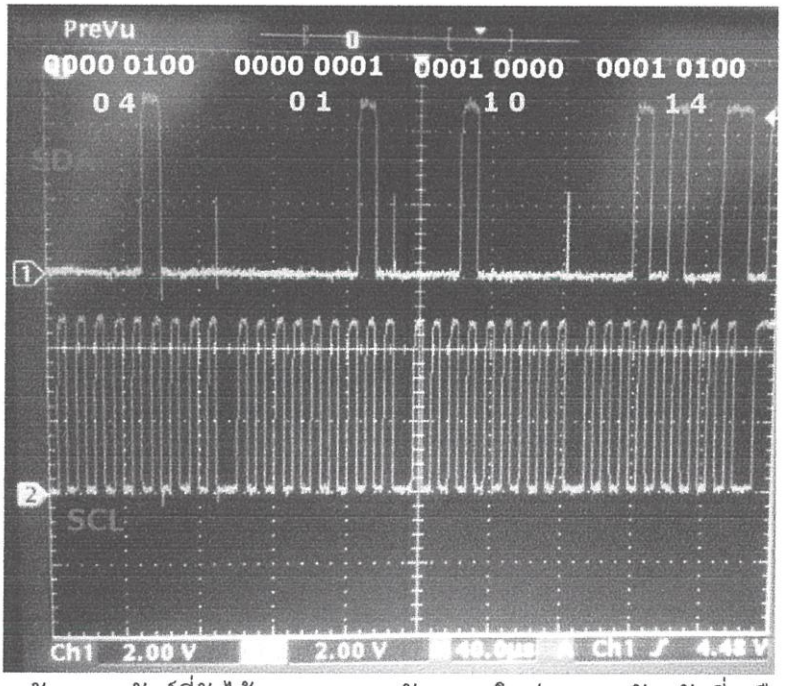
จากรูปที่ 4.5 สามารถต่อวงจรเรียลไทม์คล็อกเข้ากับวงจรไมโครคอนโทรลเลอร์ได้โดยต่อขา 4 (SDA), 5(SCL) ของวงจรเรียลไทม์คล็อกเข้ากับขา 27,28 ของวงจรไมโครคอนโทรลเลอร์ เมื่อทำการวัดสัญญาณต่างๆ จะได้ผลการดำเนินการทั้งหมด โดยจะมีผลค่าเวลาและวันที่ แสดงดังรูปที่ 4.6 มีลักษณะสัญญาณ ดังรูปที่ 4.7 และรูปที่ 4.8



รูปที่ 4.6 หน้าจอคอมพิวเตอร์แสดงผลค่าเวลาและวันที่ คือ วันพุธที่ 1 เดือนตุลาคม 2014 เวลา 22.08.15 น.



รูปที่ 4.7 สัญญาณพัลส์ที่วัดได้จากขา SDA กับ SCL ในส่วนของ Address Write, Register Address, วินาที, นาที, ชั่วโมง (บอกค่าเวลา)

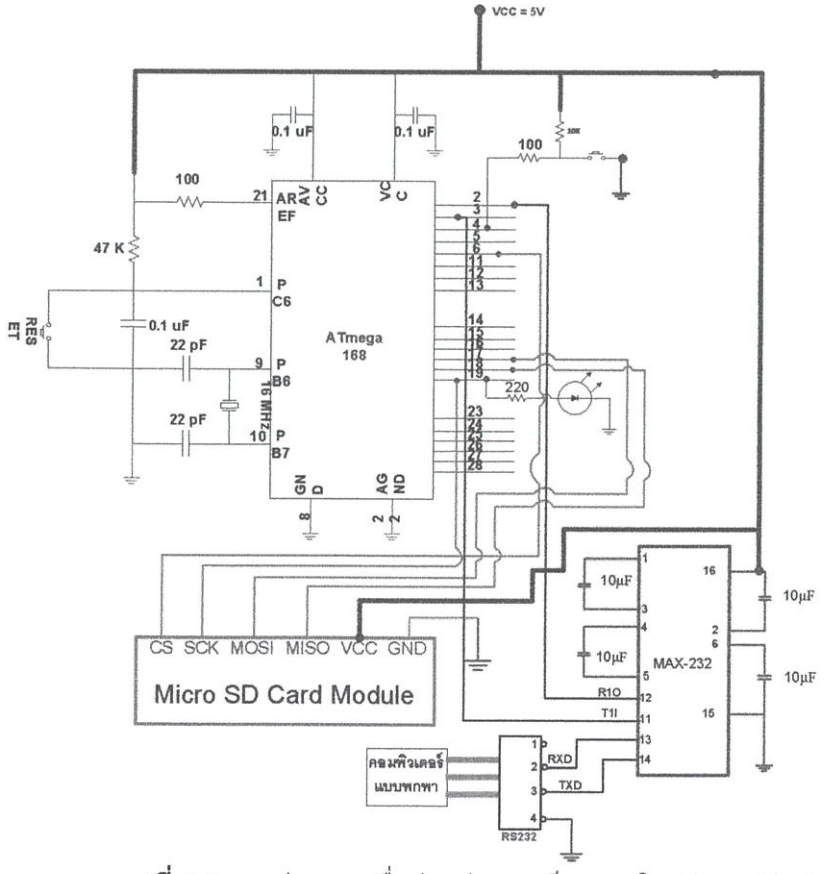


รูปที่ 4.8 สัญญาณพัลส์ที่วัดได้จากขา SDA กับ SCL ในส่วนของ วัน, วันที่, เดือน, ปี (บอกค่าวันที่)

จากรูปที่ 4.7 และรูปที่ 4.8 แสดงสัญญาณพัลส์ของบิตข้อมูลที่วัดได้จากขา SCL และ SDA ของวงจรเรียลไทม์คล็อกจะเห็นว่าจากสัญญาณพัลส์ที่วัดได้จากขา SDA คือสัญญาณพัลส์แสดงค่าเวลาและวันที่ ส่วนสัญญาณพัลส์จากขา SCL จะแสดงสัญญาณนาฬิกาของแต่ละไบต์ มีทั้งหมด 9 ไบต์ซึ่งในรูปที่ 4.7 ที่ขา SDA จะสามารถอ่านค่าออกมาได้เป็นเวลา 22.08.15 น. ส่วนรูปที่ 4.8 จะสามารถอ่านค่าออกมาได้เป็น วันพุธ(04) ที่ 1 เดือนตุลาคม 2014

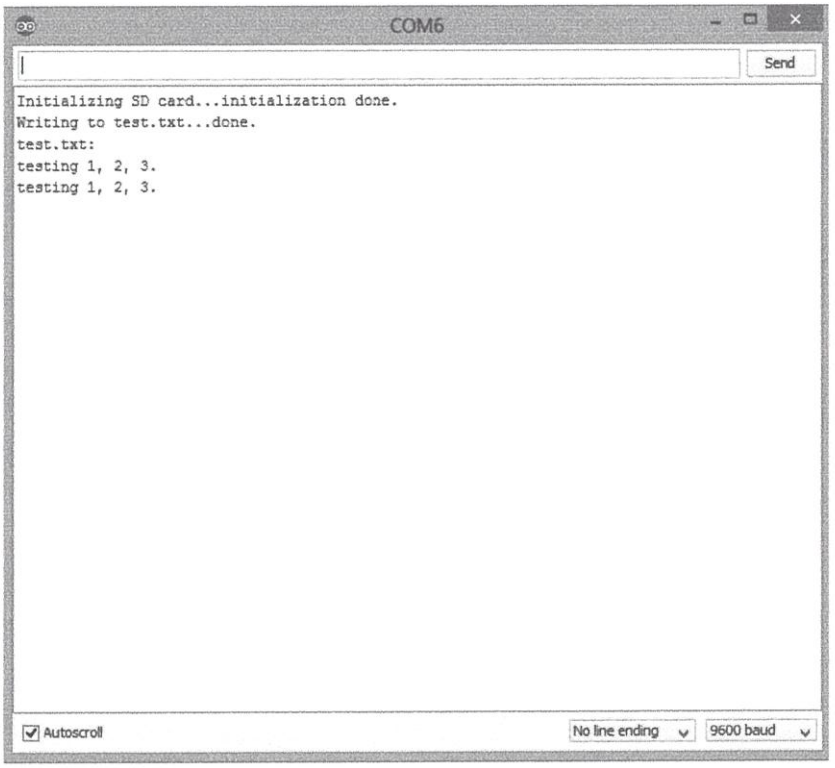
4.4 การอ่านค่าและเก็บผลลงใน Micro SD Card Module

การอ่านค่าและเก็บผลลงใน Micro SD Card Module จะใช้รูปแบบการติดต่อสื่อสารแบบ Serial Peripheral Protocol (SPI) โดย SPI จะส่งและรับข้อมูลทีละบิต (Bit Serial) และใช้สัญญาณนาฬิกา เป็นตัวกำหนดจังหวะการทำงาน ในการทำงานของอุปกรณ์ในระบบบัส แบ่งเป็น SPI Master และ SPI Slave การทดลองมีการต่ออุปกรณ์โดยรวมในการทดลองในหัวข้อนี้ แสดงดังรูปที่ 4.9



รูปที่ 4.9 การต่อวงจรเพื่ออ่านค่าและเก็บผลลงใน Micro SD Card Module

จากรูปที่ 4.9 เมื่อต่อ Micro SD Card Module เข้ากับวงจรไมโครคอนโทรลเลอร์และใช้ไมโครคอนโทรลเลอร์ โดยจะใช้โฟลว์ชาร์ตการทำงานของโปรแกรม ในข้อหัวที่ 3.1.5 ทำการกำหนดให้ Micro SD Card Module บันทึกค่า testing 1, 2, 3 จะได้ผลการดำเนินงาน ดังรูปที่ 4.10

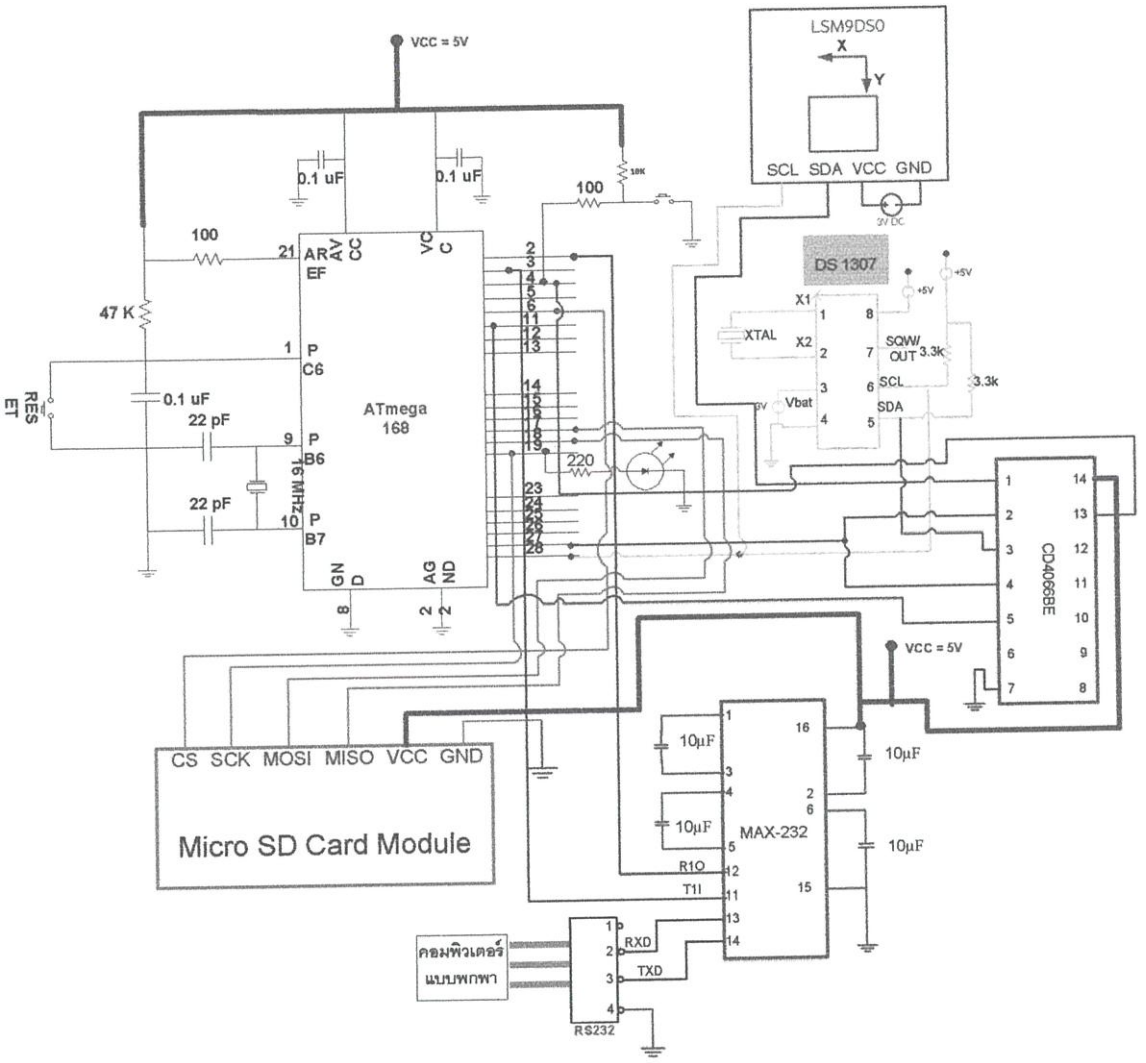


รูปที่ 4.10 หน้าจอคอมพิวเตอร์แสดงผลค่าที่บันทึกลงใน Micro SD Card Module โดยค่าที่ได้คือ testing 1, 2, 3

4.5 การเก็บผลรวมของค่าความเร่งเชิงเส้น 3 แกน กับค่าของเวลา ลงใน Micro SD Card Module

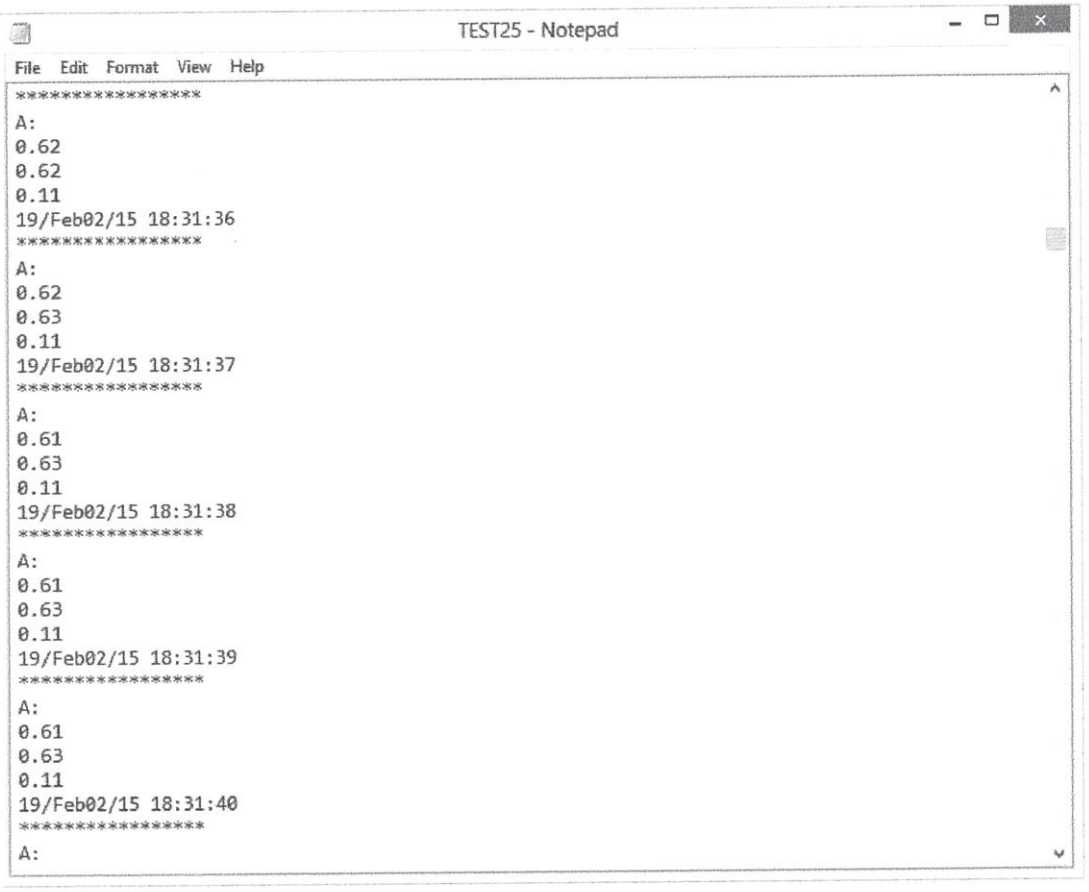
การเก็บผลค่าความเร่งเชิงเส้น 3 แกน จากโมดูลเซ็นเซอร์ LSM9DS0 และค่าเวลาจากวงจรเรียลไทม์คล็อก เพื่อบันทึกข้อมูลลงใน Micro SD Card Module ที่ใช้เก็บข้อมูลให้กับระบบ โดยที่โมดูลเซ็นเซอร์ LSM9DS0 และวงจรเรียลไทม์คล็อก จะอาศัยการรับ-ส่งข้อมูล โดยใช้ I^2C BUS (Inter Integrate Circuit Bus) และ Micro SD Card Module จะอาศัยการรับ-ส่งข้อมูลโดยใช้ SPI ในการติดต่อกับไมโครคอนโทรลเลอร์

การทดลองมีการต่ออุปกรณ์โดยรวมในการทดลองในหัวข้อนี้ แสดงดังรูปที่ 4.11



รูปที่ 4.11 การต่อวงจรโดยรวมของระบบ

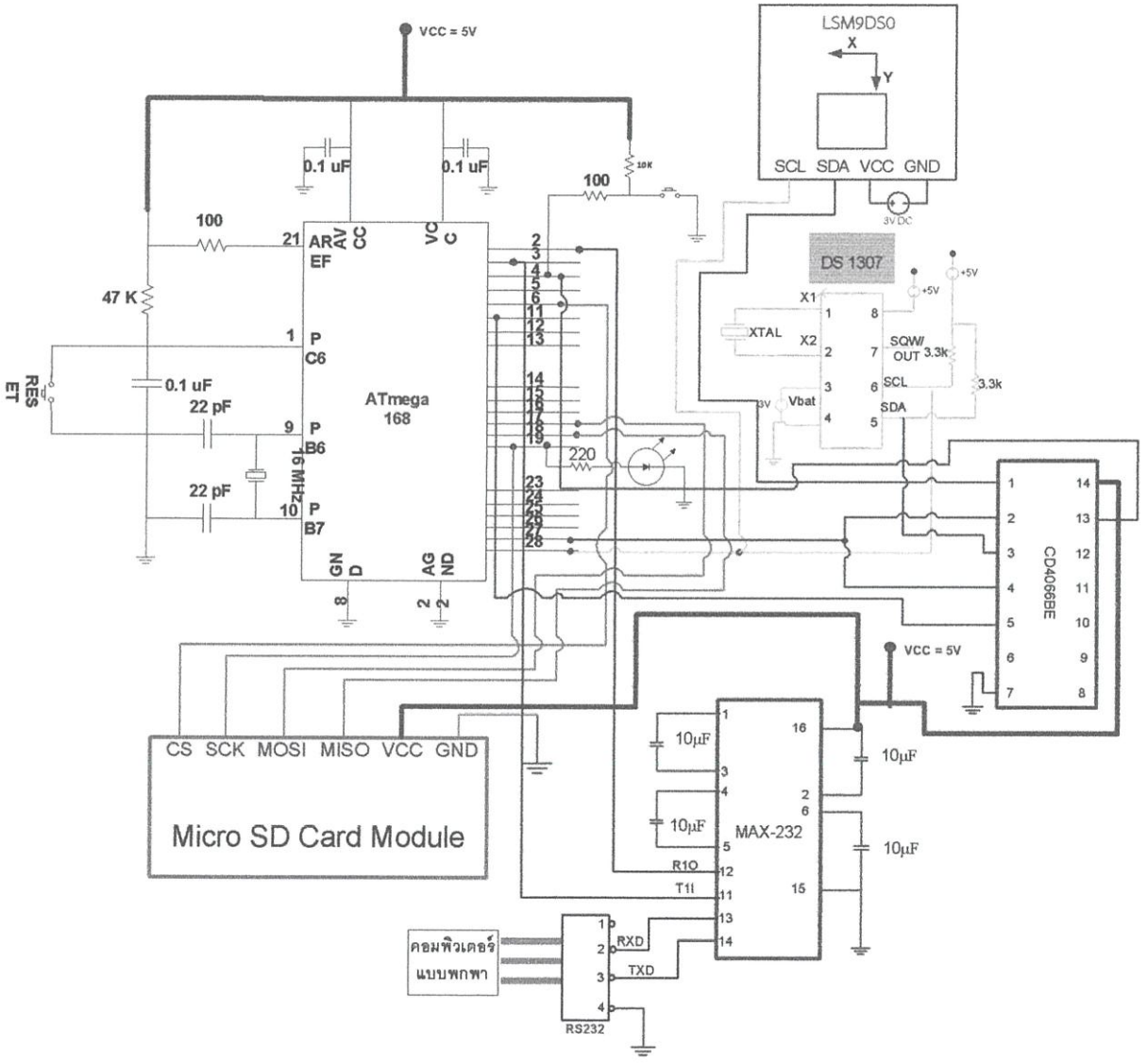
จากรูปที่ 4.11 สามารถต่อวงจรโมดูลเซ็นเซอร์ LSM9DS0 วงจรเรียลไทม์คล็อก และวงจร Micro SD Card Module ร่วมกับวงจรไมโครคอนโทรลเลอร์ โดยจะใช้โฟลว์ชาร์ตการทำงานของโปรแกรมในข้อหัวที่ 3.1.6 วงจรนี้จะใช้ในการเก็บข้อมูลของค่าความเร่งเชิงเส้น 3 แกน ที่ได้จากโมดูลเซ็นเซอร์ LSM9DS0 และค่าเวลาจากวงจรเรียลไทม์คล็อก ณ เวลานั้นๆ โดยสร้างไฟล์ .test ในการบันทึกข้อมูลลงใน SD Card Module จะได้ผลการดำเนินการทั้งหมด ดังรูปที่ 4.12



รูปที่ 4.12 ค่าความเร่งเชิงเส้น 3 แกน (แกน X = 0.61 m/s^2 แกน Y = 0.63 m/s^2 แกน Z = 0.11 m/s^2) และ แสดงค่าวันที่และเวลา ในไฟล์ .test ที่ได้กำหนดไว้

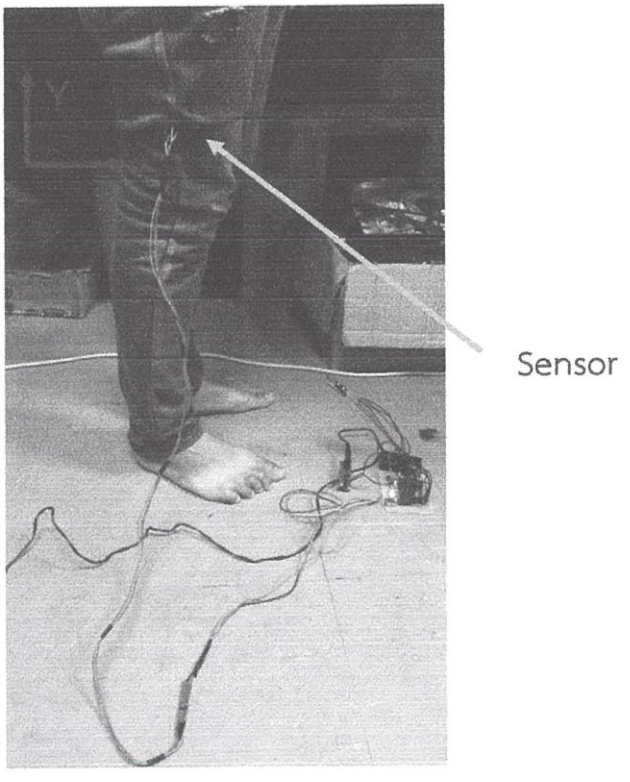
4.6 การเก็บผลค่าความเร่งเชิงเส้น 3 แกน ในลักษณะการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน ลงใน Micro SD Card Module ของผู้ทำการทดลอง 5 คน

การทดลองมีการต่ออุปกรณ์โดยรวมในการทดลองในหัวข้อนี้ แสดงดังรูปที่ 4.13

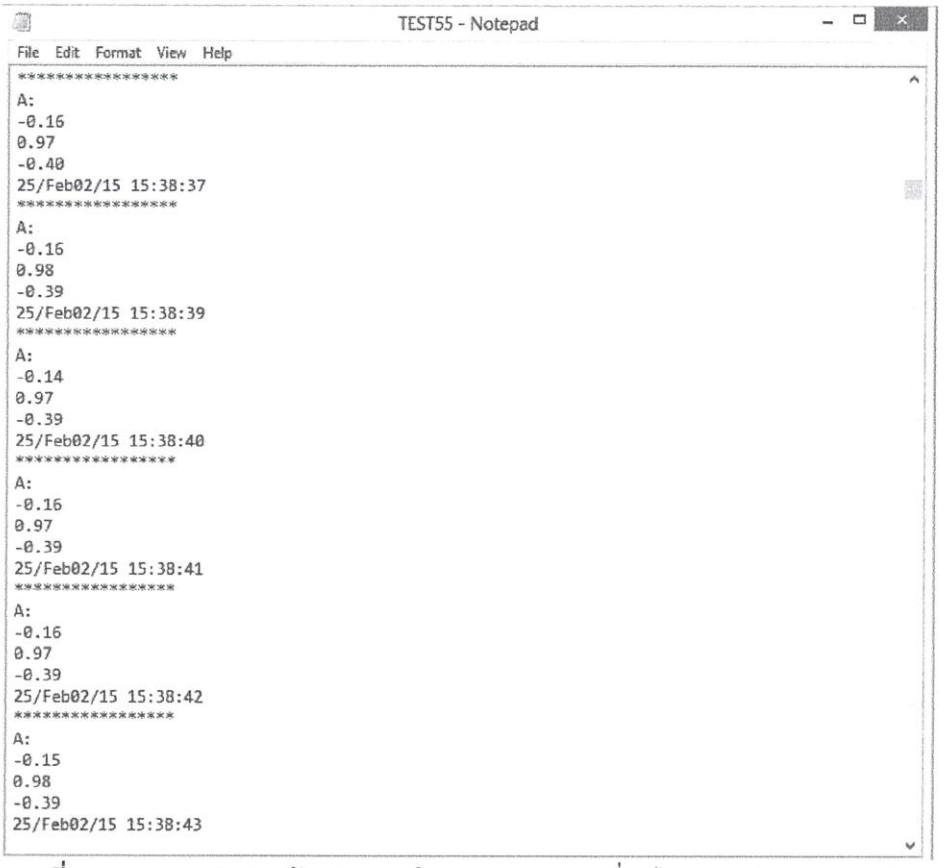


รูปที่ 4.13 การต่อวงจรรวมของเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย

จากรูปที่ 4.13 สามารถต่อวงจรรวมของเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย โดยจะใช้โฟลว์ชาร์ตการทำงานของโปรแกรมในข้อหัวที่ 3.1.6 โดยสร้างไฟล์ .test ในการบันทึกข้อมูลลงใน SD Card Module และติดโมดูลเซ็นเซอร์ LSM9DS0 ตรงที่บริเวณต้นขาขวาดังรูปที่ 4.14 จะได้ผลค่าความเร่งเชิงเส้น 3 แกน ในลักษณะการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน ของโมดูลเซ็นเซอร์ LSM9DS0 และค่าของเวลา ตามลำดับ ดังรูปที่ 4.15

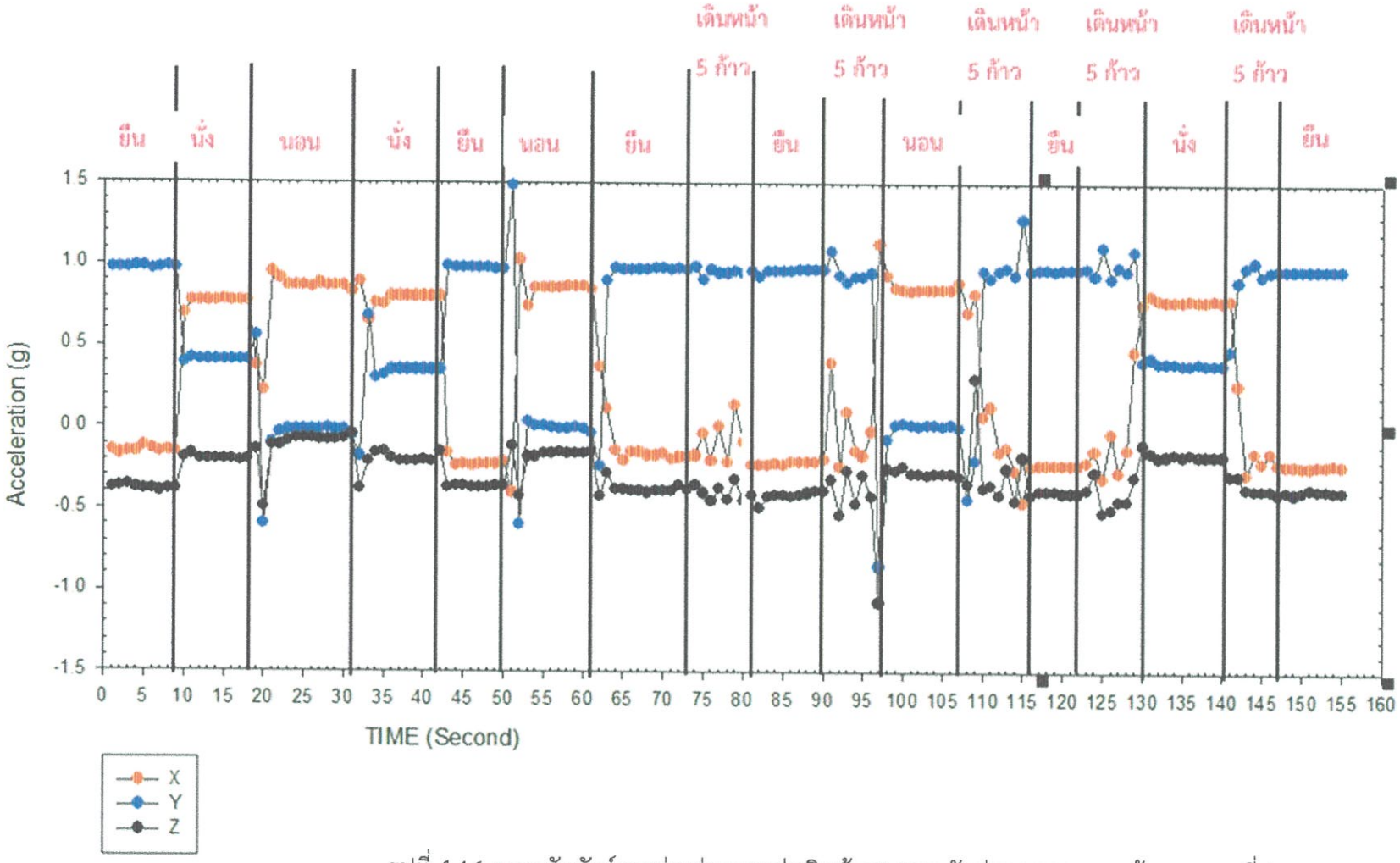


รูปที่ 4.14 การทดลองเก็บผลค่าความเร่งเชิงเส้น 3 แกน ในลักษณะการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน ของโมดูลเซ็นเซอร์ LSM9DS0 และค่าของเวลา ลงใน Micro SD Card Module

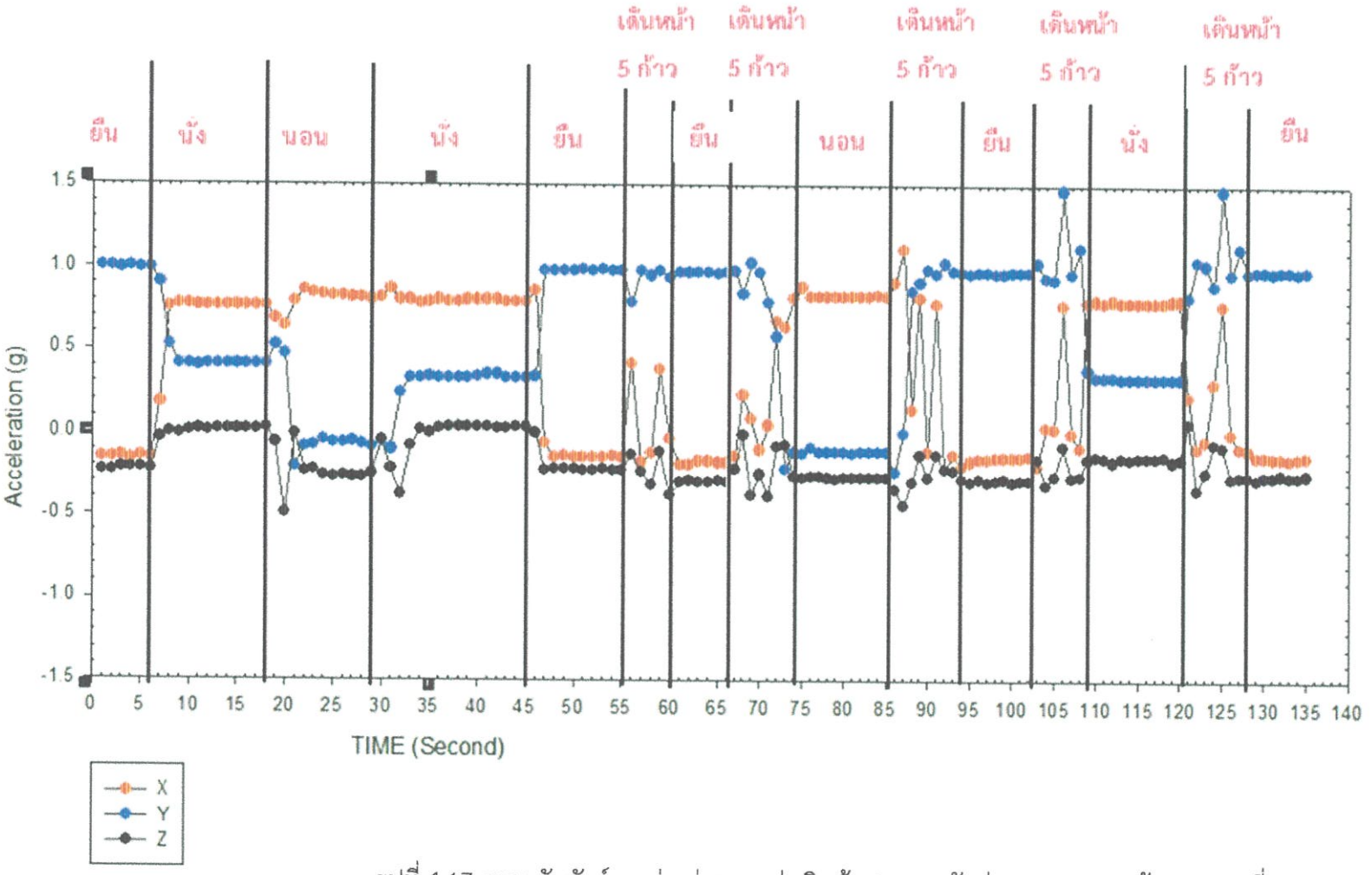


รูปที่ 4.15 ความเร่งเชิงเส้น 3 แกน ในลักษณะการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน การเดิน และค่าของเวลา ในไฟล์ .test ที่ได้กำหนดไว้

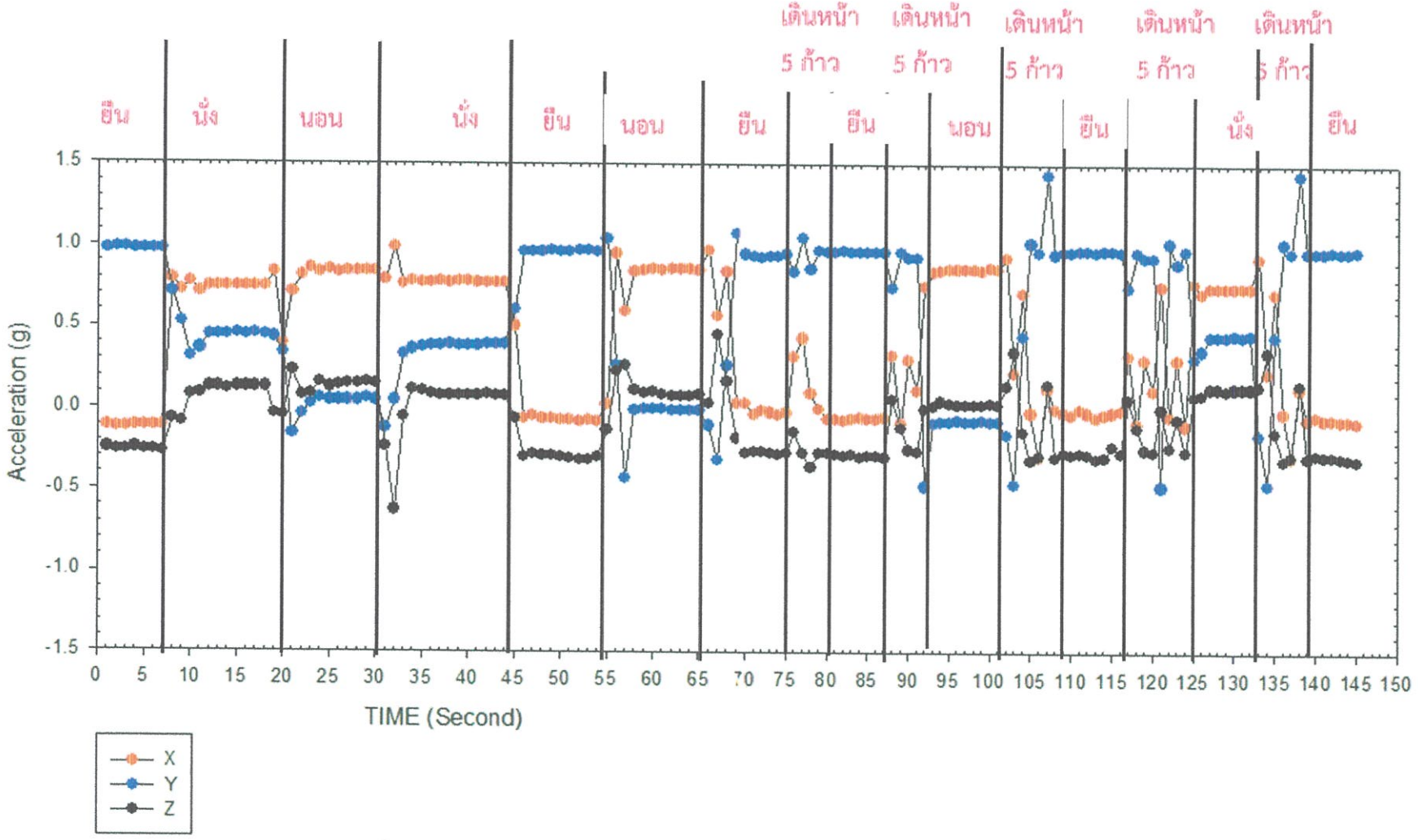
เมื่อนำค่าความเร่งเชิงเส้น 3 แกน ในลักษณะการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน ในไฟล์ .test มาพล็อตกราฟเทียบกับค่าของเวลา ของผู้ทำการทดลองทั้ง 5 คน โดยแต่ละค่าความเร่งเชิงเส้นมีระยะห่างกันทุกๆ 1 วินาที แสดงดังรูปที่ 4.16, 4.17, 4.18, 4.19 และ 4.20 ตามลำดับ



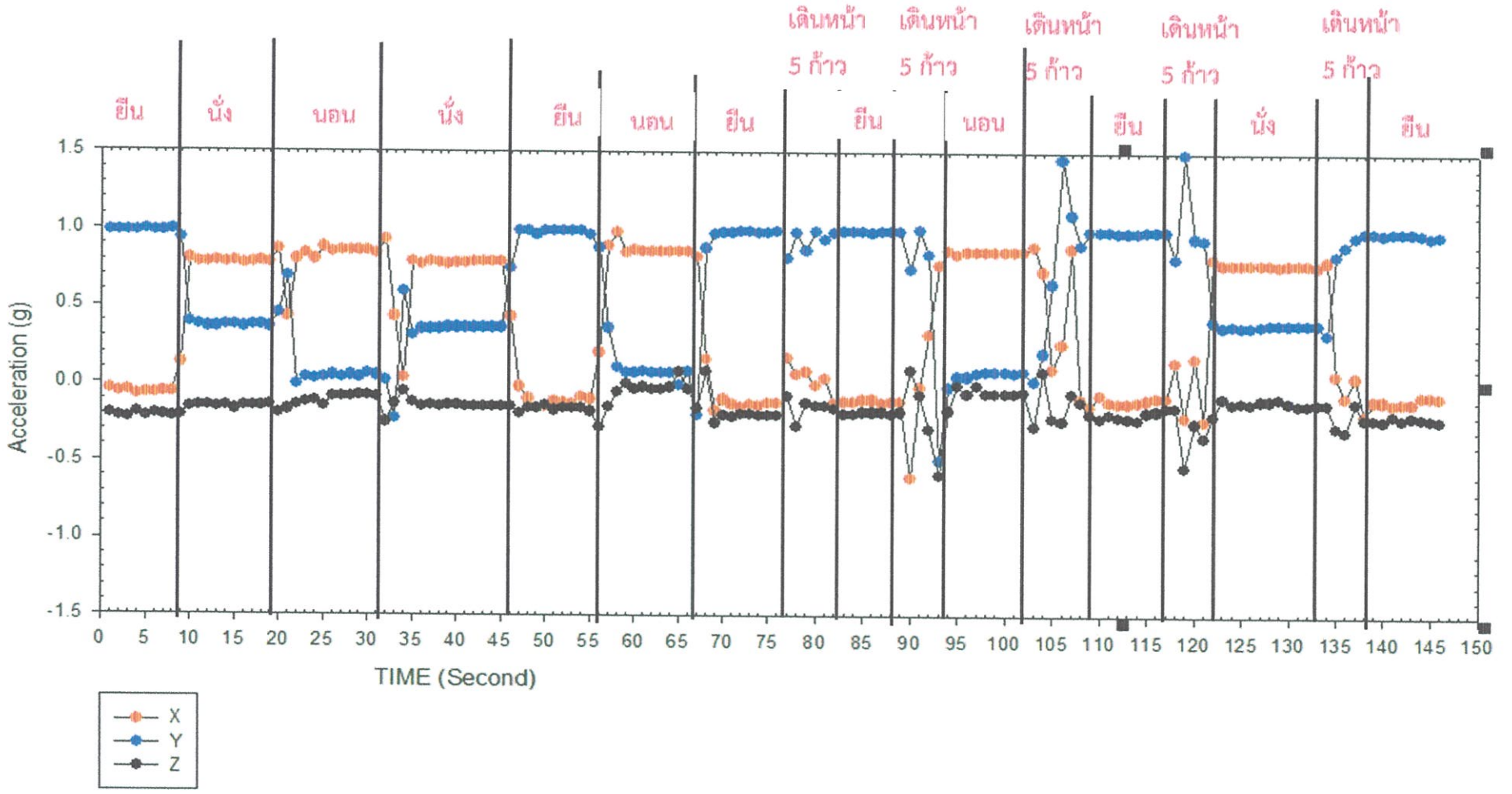
รูปที่ 4.16 ความสัมพันธ์ระหว่างค่าความเร่งเชิงเส้น 3 แกน กับค่าของเวลา ของผู้ทดลองคนที่ 1



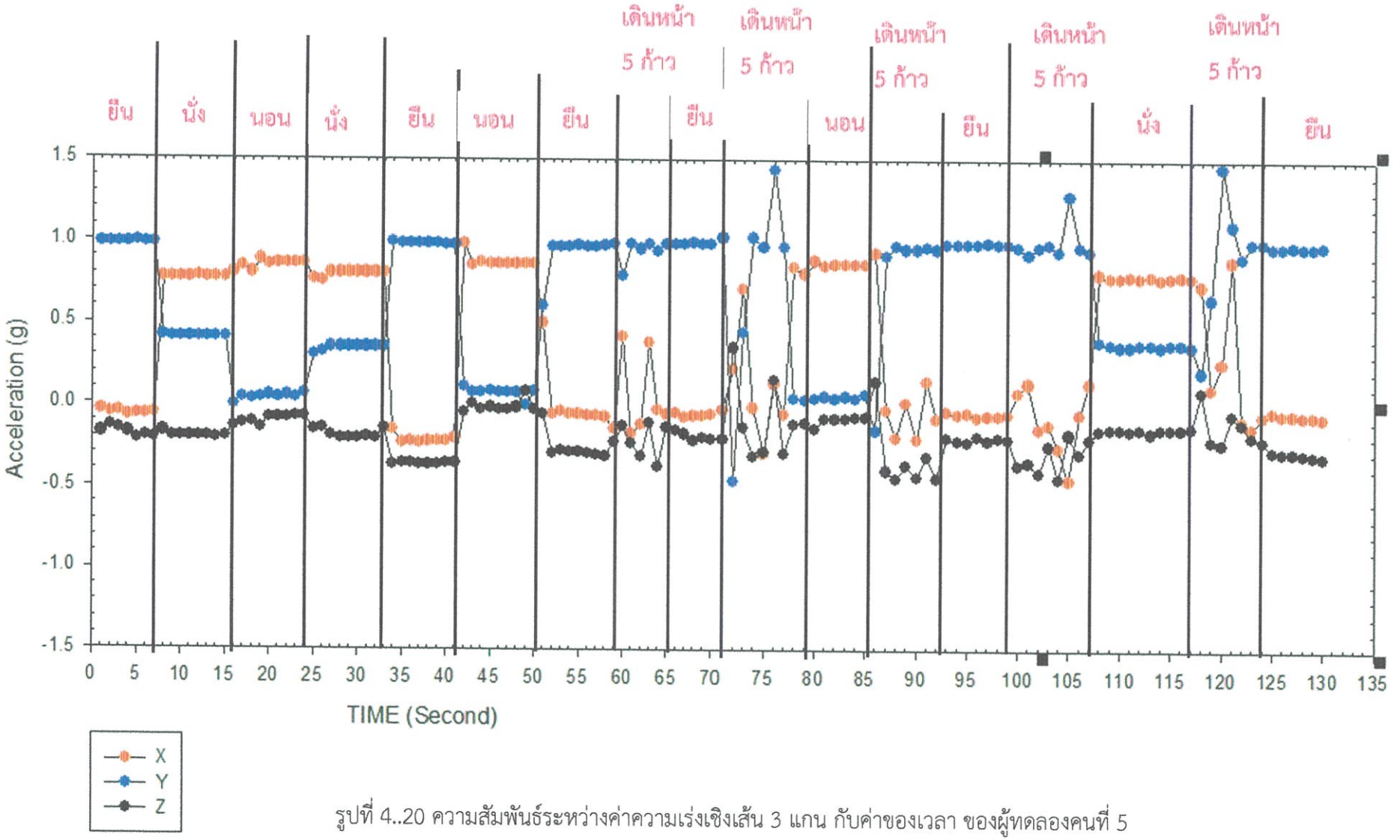
รูปที่ 4.17 ความสัมพันธ์ระหว่างค่าความเร่งเชิงเส้น 3 แกน กับค่าของเวลา ของผู้ทดลองคนที่ 2



รูปที่ 4.18 ความสัมพันธ์ระหว่างค่าความเร่งเชิงเส้น 3 แกน กับค่าของเวลา ของผู้ทดลองคนที่ 3



รูปที่ 4.19 ความสัมพันธ์ระหว่างค่าความเร่งเชิงเส้น 3 แกน กับค่าของเวลา ของผู้ทดลองคนที่ 4



รูปที่ 4..20 ความสัมพันธ์ระหว่างค่าความเร่งเชิงเส้น 3 แกน กับค่าของเวลา ของผู้ทดลองคนที่ 5

จากรูปที่ 4.16, 4.17, 4.18, 4.19 และ 4.20 เมื่อพิจารณาค่าความเร่งเชิงเส้น 3 แกน ในลักษณะการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน จะได้ค่าความเร่งเชิงเส้น 3 แกน ของผู้ทดลองทั้ง 5 คน ดังในตารางที่ 4.1

ตารางที่ 4.1 ค่าความเร่งเชิงเส้น 3 แกน ในลักษณะการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน ของผู้ทดลองทั้ง 5 คน

จำนวนผู้ทำการทดลอง	ค่าความเร่งในแต่ละแนวแกน	ลักษณะการเคลื่อนไหวของร่างกาย			
		ยืน (Stand)	นั่ง (Sit)	นอน (Sleep)	เดิน (Walk)
คนที่ 1	แกน X (g)	-0.15 ⟷ 0.05	0.72 ⟷ 0.95	0.80 ⟷ 0.90	-0.30 ⟷ 0.40
	แกน Y (g)	0.90 ⟷ 1.05	0.27 ⟷ 0.45	-0.15 ⟷ 0.05	0.80 ⟷ 1.10
	แกน Z (g)	-0.35 ⟷ -0.15	-0.20 ⟷ 0.10	-0.20 ⟷ 0.15	-0.50 ⟷ 0.00
คนที่ 2	แกน X (g)	-0.20 ⟷ -0.05	0.70 ⟷ 0.85	0.78 ⟷ 0.90	-0.25 ⟷ 1.00
	แกน Y (g)	0.90 ⟷ 1.10	0.30 ⟷ 0.45	-0.15 ⟷ 0.00	0.75 ⟷ 1.15
	แกน Z (g)	-0.35 ⟷ -0.18	-0.20 ⟷ 0.10	-0.35 ⟷ -0.15	-0.50 ⟷ 0.05
คนที่ 3	แกน X (g)	-0.20 ⟷ -0.05	0.70 ⟷ 0.85	0.80 ⟷ 0.90	-0.15 ⟷ 1.15
	แกน Y (g)	0.90 ⟷ 1.10	0.30 ⟷ 0.50	-0.15 ⟷ 0.10	0.80 ⟷ 1.20
	แกน Z (g)	-0.40 ⟷ -0.20	-0.20 ⟷ 0.15	-0.15 ⟷ -0.20	-0.45 ⟷ 0.10
คนที่ 4	แกน X (g)	-0.20 ⟷ -0.05	0.70 ⟷ 0.85	0.78 ⟷ 0.90	-0.25 ⟷ 1.05
	แกน Y (g)	0.90 ⟷ 1.10	0.30 ⟷ 0.45	-0.15 ⟷ 0.15	0.75 ⟷ 1.15
	แกน Z (g)	-0.35 ⟷ -0.18	-0.25 ⟷ 0.10	-0.15 ⟷ -0.15	-0.45 ⟷ 0.05
คนที่ 5	แกน X (g)	-0.20 ⟷ 0.05	0.72 ⟷ 0.95	0.80 ⟷ 0.90	-0.30 ⟷ 0.40
	แกน Y (g)	0.90 ⟷ 1.05	0.25 ⟷ 0.45	-0.20 ⟷ 0.15	0.80 ⟷ 1.15
	แกน Z (g)	-0.35 ⟷ 0.10	-0.20 ⟷ 0.10	-0.20 ⟷ 0.15	-0.50 ⟷ 0.10

ตารางที่ 4.2 สรุปค่าความเร่งเชิงเส้น 3 แกน ในลักษณะการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน ของผู้ทดลองทั้ง 5 คน

ค่าความเร่งในแต่ละแนวแกน	ลักษณะการเคลื่อนไหวของร่างกาย			
	ยืน (Stand)	นั่ง (Sit)	นอน (Sleep)	เดิน (Walk)
แกน X (m/s^2)	-0.15 ⟷ 0.05	0.72 ⟷ 0.95	0.80 ⟷ 0.90	-0.30 ⟷ 0.40
แกน Y (m/s^2)	0.90 ⟷ 1.05	0.27 ⟷ 0.45	-0.15 ⟷ 0.05	0.80 ⟷ 1.10
แกน Z (m/s^2)	-0.35 ⟷ -0.15	-0.20 ⟷ 0.10	-0.20 ⟷ 0.15	-0.50 ⟷ 0.00

จากตารางที่ 4.2 นำค่าความเร่งเชิงเส้น 3 แกนมาพิจารณาสร้างฟังก์ชัน ในโปรแกรม อาร์ดูอิโน่ และการทำงานของฟังก์ชั่นการประมวลผลหาลักษณะการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน ในหัวข้อที่ 3.1.7

4.6.1 ทำการทดสอบฟังก์ชั่นการประมวลผลหาลักษณะการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน ของผู้ทำการทดลอง

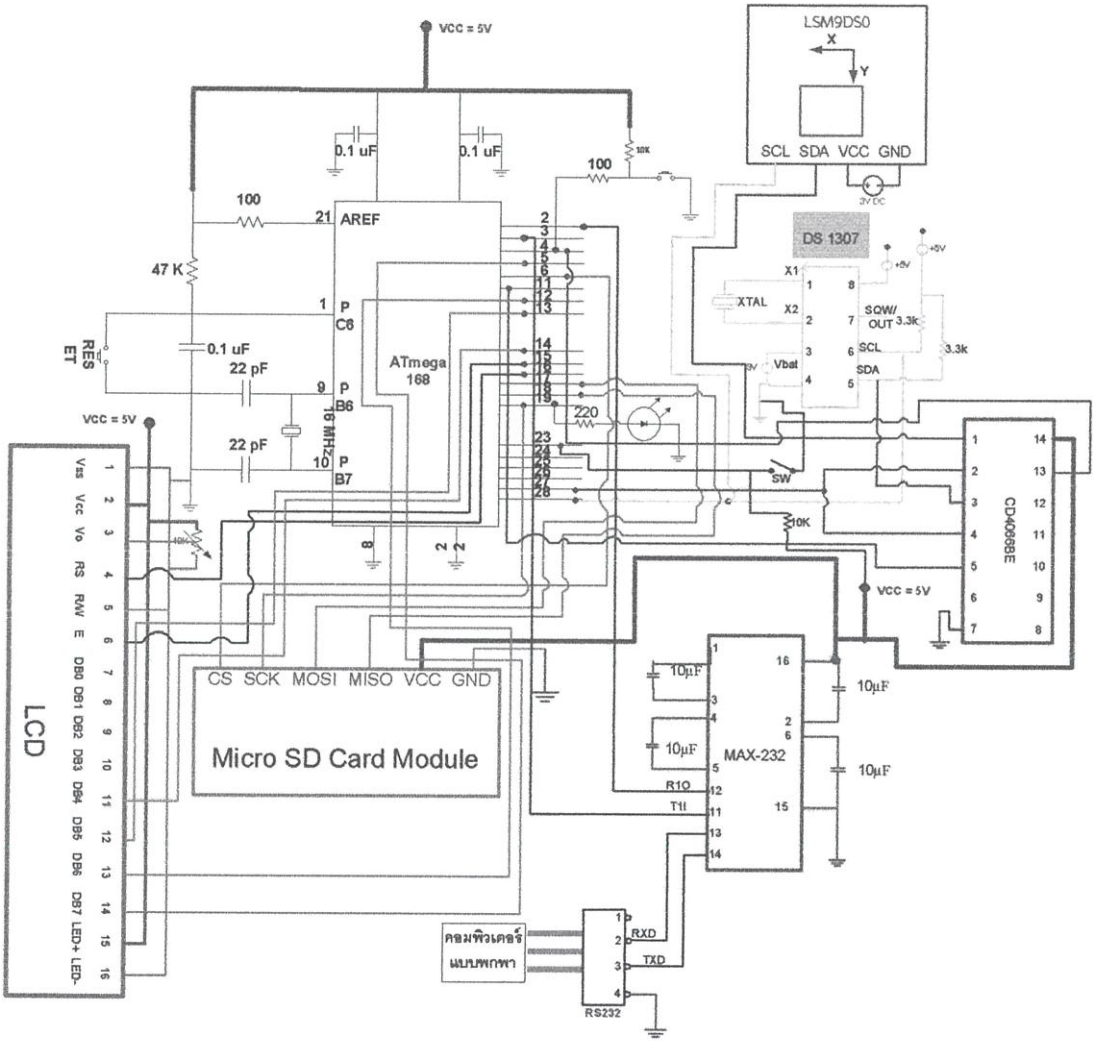
โดยทำการทดสอบการทำงานฟังก์ชั่นการประมวลผลหาลักษณะการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน กับผู้ทำการทดลอง 5 คน ในเวลา 10.20.00 น. ถึง 11.20.00 น. ของวันที่ 21 เมษายน พ.ศ. 2558 โดยทำการทดลองคนละ 20 นาที และสามารถสรุปการทำงานของฟังก์ชันการประมวลผลหาลักษณะการเคลื่อนไหวของร่างกายของผู้ทำการทดลอง 5 ทั้งคนดังตารางที่ 4.3

ตารางที่ 4.3 การทดสอบฟังก์ชั่นการประมวลผลหาลักษณะการเคลื่อนไหวของร่างกาย

ลักษณะการเคลื่อนไหวของร่างกาย	จำนวนเวลาที่ใช้ในการทดสอบ 5 นาที (300 วินาที)	จำนวนเวลาที่วัดได้จริง (วินาที)	เปอร์เซ็นต์ความผิดพลาด
ยืน (Stand)	300	254	15.33%
นั่ง (Sit)	300	262	12.67%
นอน (Sleep)	300	263	12.33%
เดิน (Walk)	300	239	20.33%

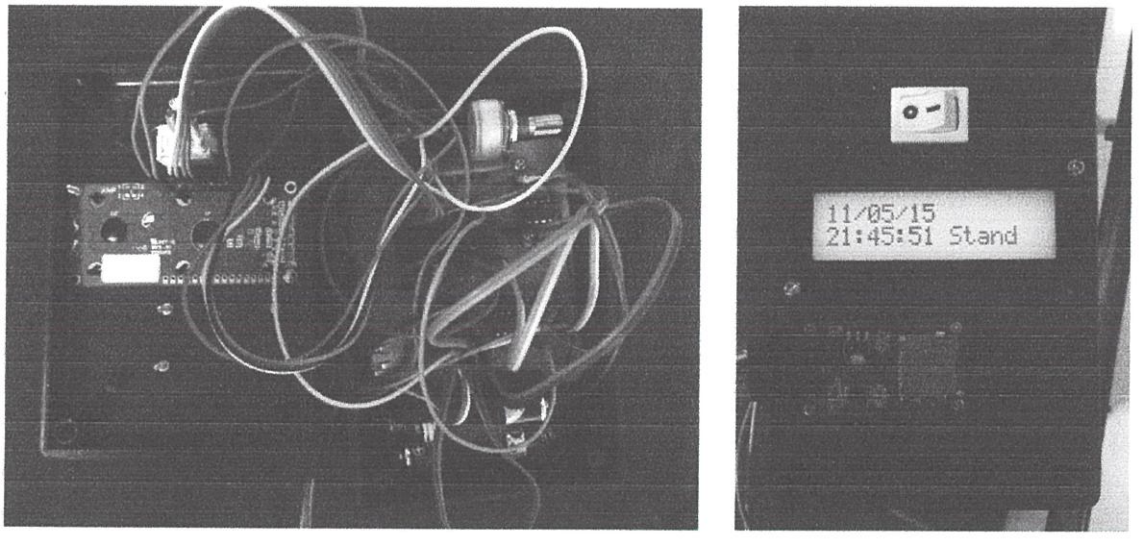
4.7 การทดสอบการทำงานของเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน

การทดลองมีการต่ออุปกรณ์โดยรวมในการทดลองในหัวข้อนี้ แสดงดังรูปที่ 4.21



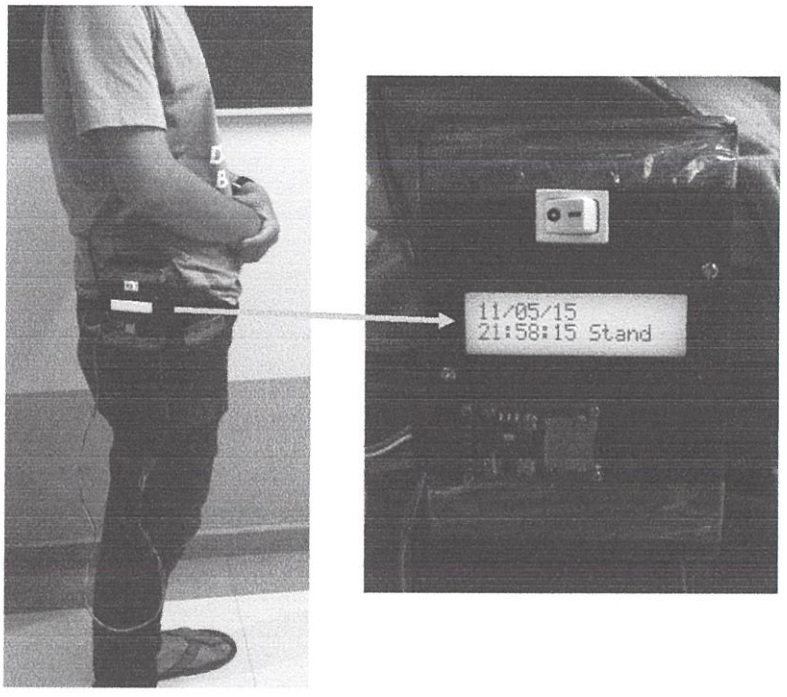
รูปที่ 4.21 วงจรเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย

จากรูปที่ 4.21 สามารถต่อวงจรเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย โดยจะใช้โฟลว์ชาร์ตการทำงานของโปรแกรมในข้อหัวที่ 3.1.8 โดยจะแสดงผลกิจกรรมการเคลื่อนไหวของร่างกายและค่าเวลาปัจจุบัน บนหน้าจอ LCD ที่ติดกับตัวเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกายแสดงดังรูปที่ 4.22 และเมื่อกดปุ่ม Start เครื่องจะทำการบันทึกผลของกิจกรรมการเคลื่อนไหวของร่างกายกับค่าของเวลา ลงใน SD Card และเมื่อกดปุ่ม Stop เครื่องจะทำการคำนวณอัตราการใช้พลังงานของร่างกายจากกิจกรรมที่ทำไปตั้งแต่ที่เริ่มทำการบันทึกบนหน้าจอ LCD โดยติดโมดูลเซ็นเซอร์ LSM9DS0 ตรงที่บริเวณต้นขาขวา จะได้ผลค่ากิจกรรมการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน การเดิน พร้อมกับค่าของเวลา และค่าพลังงานของร่างกายจากกิจกรรมที่ใช้ไป ดังรูปที่ 4.23, 4.24, 4.25, 4.26, 4.27 และ 4.28 ตามลำดับ

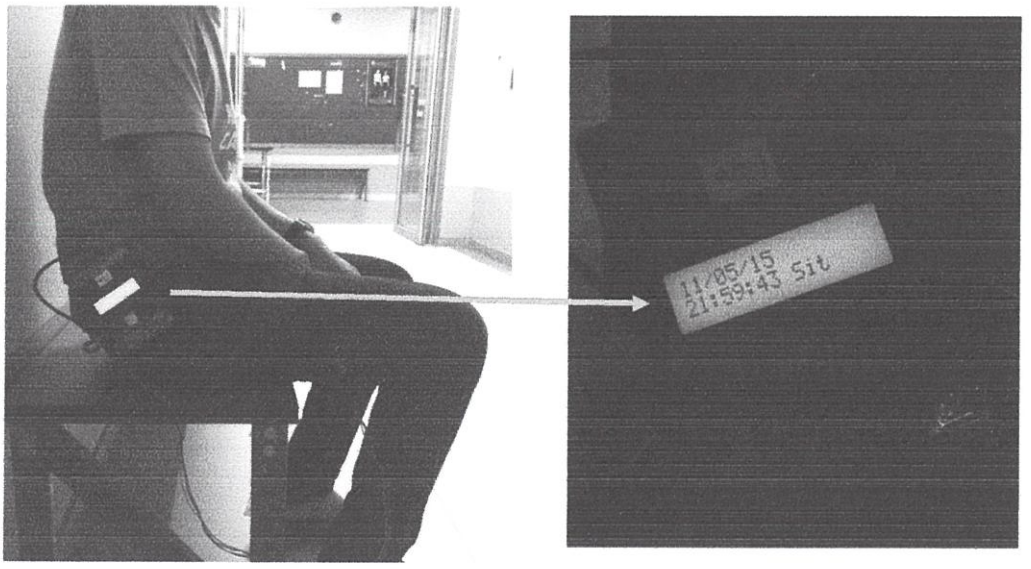


ก) ภายในกล่องอุปกรณ์ ข) ภายนอกกล่องอุปกรณ์

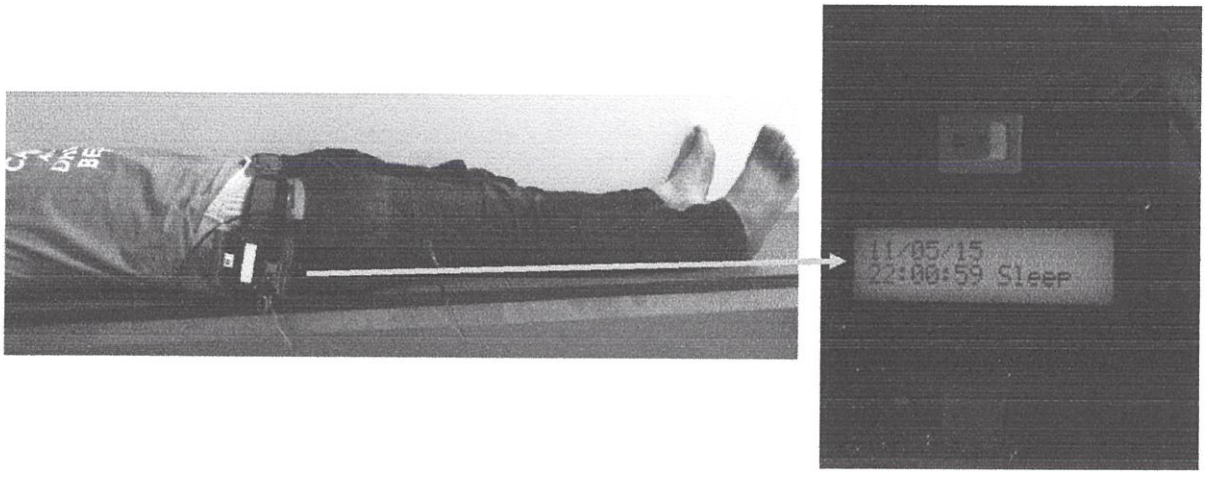
รูปที่ 4.22 กล่องอุปกรณ์เครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย



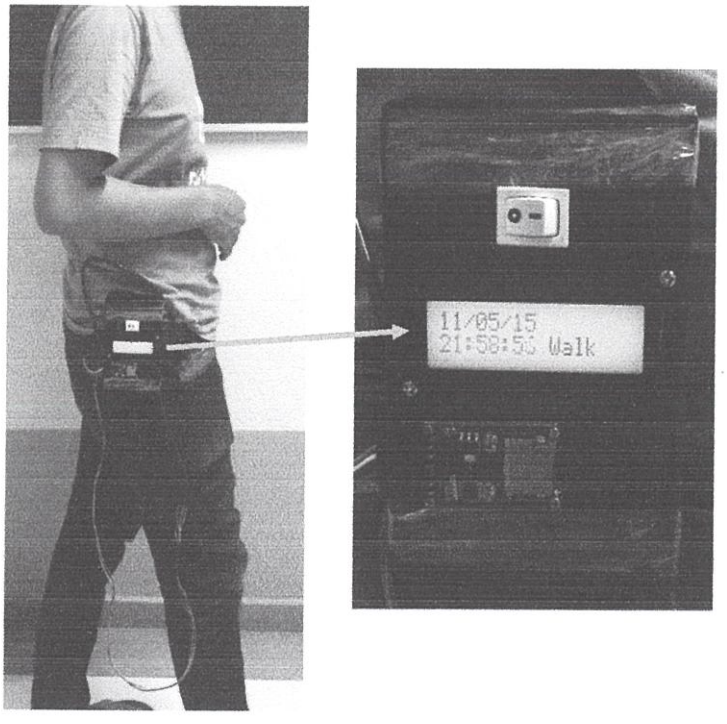
รูปที่ 4.23 การเคลื่อนไหวของร่างกายในลักษณะยืน และแสดงผลกิจกรรมการเคลื่อนไหวของร่างกายและค่าเวลาบนหน้าจอ LCD



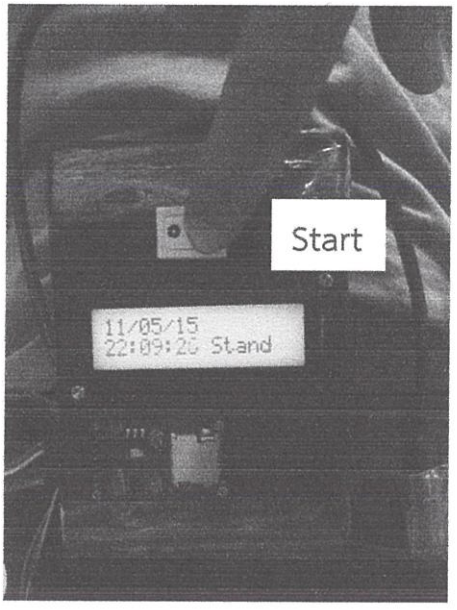
รูปที่ 4.24 การเคลื่อนไหวของร่างกายในลักษณะนั่ง และแสดงผลกิจกรรมการเคลื่อนไหวของร่างกายและค่าเวลาบนหน้าจอ LCD



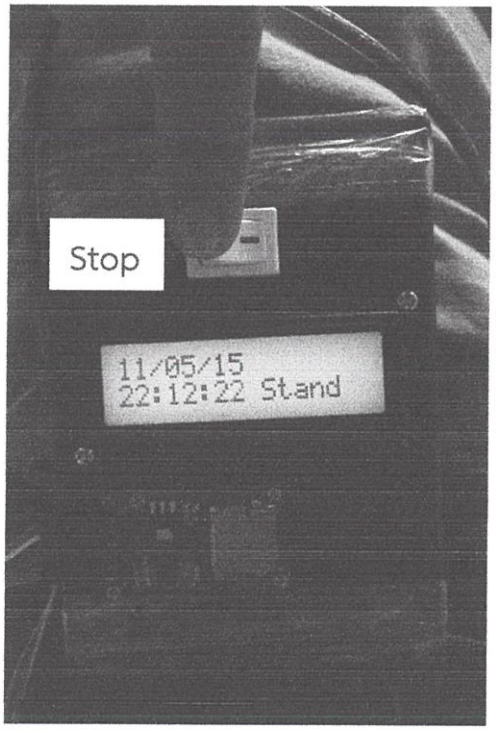
รูปที่ 4.25 การเคลื่อนไหวของร่างกายในลักษณะนอน และแสดงผลกิจกรรมการเคลื่อนไหวของร่างกายและค่าเวลาบนหน้าจอ LCD



รูปที่ 4.26 การเคลื่อนไหวของร่างกายในลักษณะเดิน และแสดงผลกิจกรรมการเคลื่อนไหวของร่างกายและค่าเวลาบนหน้าจอ LCD



รูปที่ 4.27 กดปุ่ม Strat เพื่อเริ่มบันทึกผลกิจกรรมการเคลื่อนไหวของร่างกายและค่าเวลาลงใน Micro SD Card Module

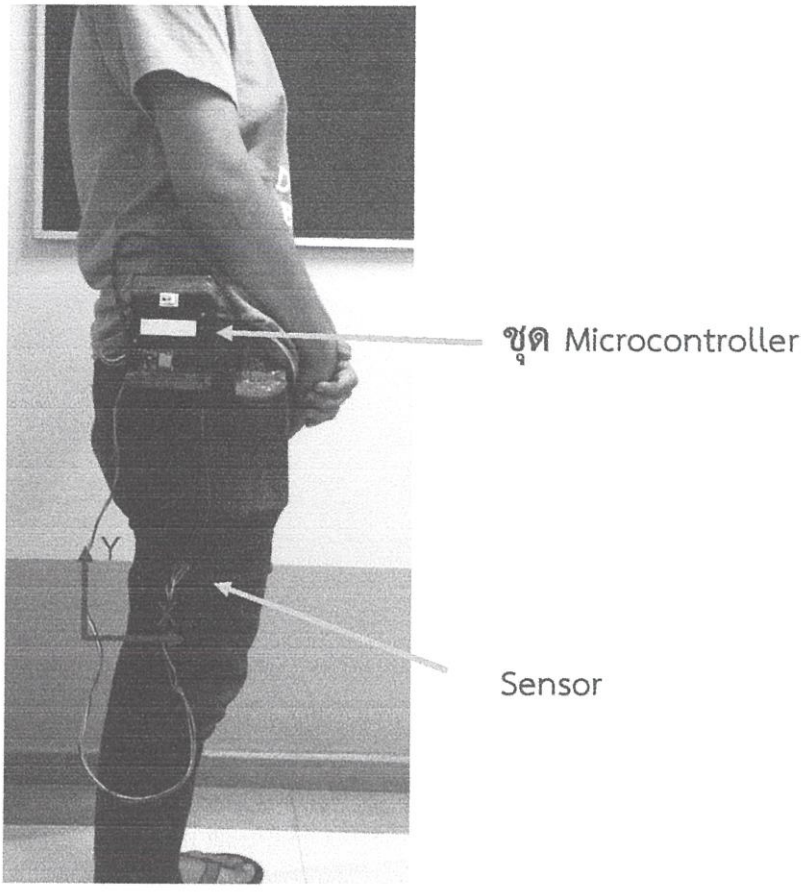


รูปที่ 4.28 กดปุ่ม Stop เพื่อสิ้นสุดการบันทึกผลกิจกรรมการเคลื่อนไหวของร่างกายและค่าเวลาลงใน Micro SD Card Module

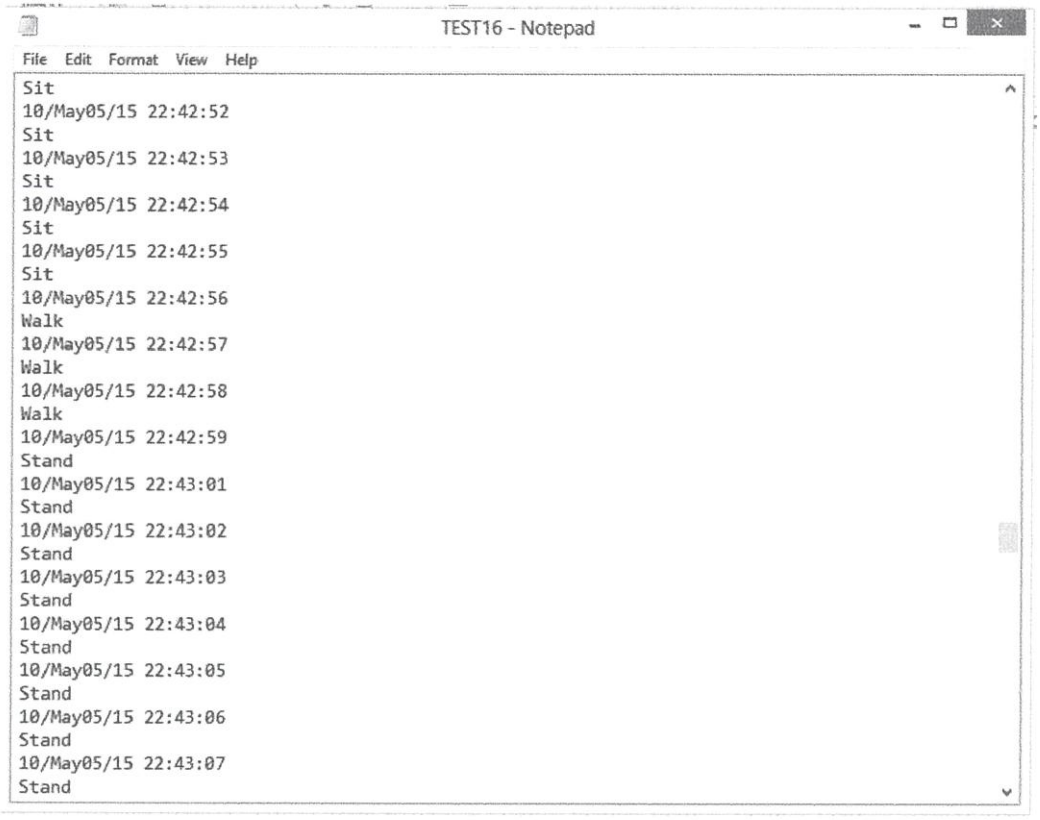
รูปที่ 4.29 ค่าพลังงานที่ใช้สำหรับทำกิจกรรมการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน (เริ่มบันทึก – สิ้นสุกการบันทึก) เท่ากับ 2.75 กิโลแคลอรี่

4.8 การเก็บผลค่ากิจกรรมการเคลื่อนไหวของร่างกาย ยืน นั่ง นอน เดิน และค่าของเวลา ลงใน Micro SD Card Module เป็นเวลา 1 วัน

ทำการต่อวงจรตามรูปที่ 4.13 และใช้โปรแกรมในข้อหัวที่ 3.1.8 โดยสร้างไฟล์ .test ในการบันทึกข้อมูลลงใน SD Card Module และติดโมดูลเซ็นเซอร์ LSM9DS0 ตรงที่บริเวณต้นขาขวาดังรูปที่ 4.29 ทำการเก็บผลค่ากิจกรรมการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน การเดินเป็นเวลา 22 ชั่วโมง จะได้ผลค่ากิจกรรมการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน การเดิน และค่าของเวลา จากวงจรเรียลไทม์คล็อก ตามลำดับ ดังรูปที่ 4.30



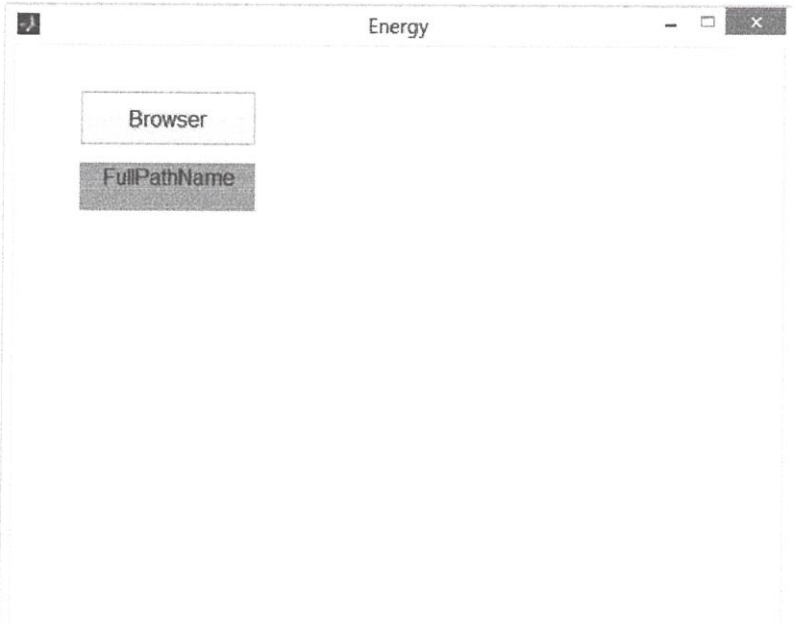
รูปที่ 4.30 การเก็บผลค่ากิจกรรมการเคลื่อนไหวของร่างกาย ยืน นั่ง นอน และเดินของโมดูลเซ็นเซอร์ LSM9DS0 และค่าของเวลา ลงใน Micro SD Card Module เป็นเวลา 1 วัน



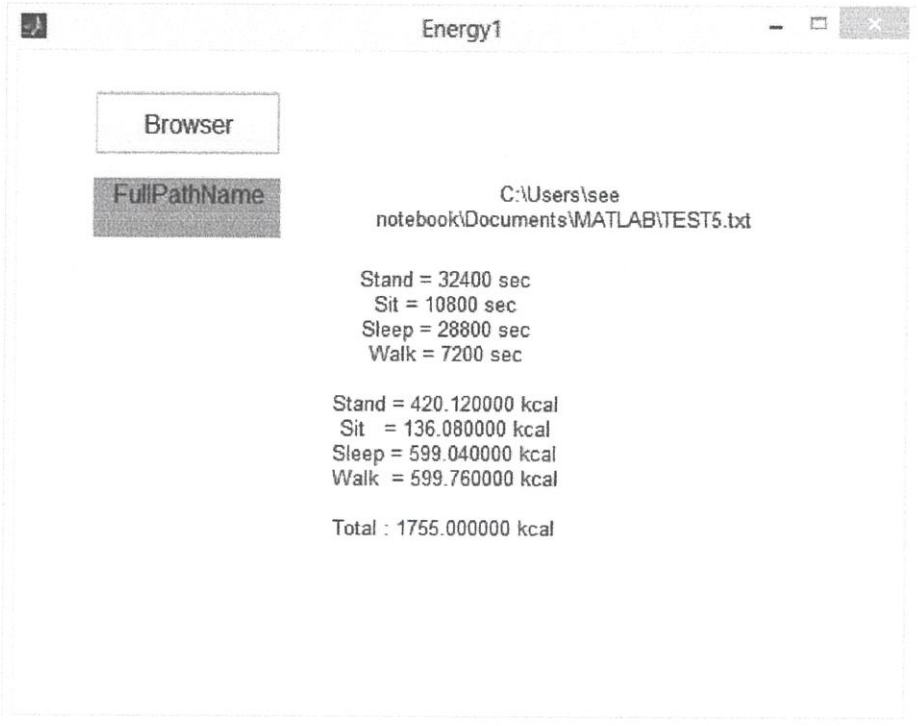
รูปที่ 4.31 ค่ากิจกรรมการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน การเดิน และค่าของเวลาเป็นเวลา 1 วัน ในไฟล์ .test ที่ได้บันทึกไว้ (โดยที่ Stand คือ ยืน Sit คือ นั่ง Sleep คือ นอน และ Walk คือ เดิน)

4.9 การคำนวณการใช้พลังงานจากกิจกรรมการเคลื่อนไหวของร่างกาย ยืน นั่ง นอน และเดิน ใน 1 วัน

นำไฟล์ .text ของค่ากิจกรรมการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน การเดิน และค่าของเวลา ที่ทำการเก็บผลเป็นเวลา 1 วัน ในหัวข้อที่ 4.8 มาคำนวณการใช้พลังงานจากกิจกรรมการเคลื่อนไหวของร่างกาย โดยค่าพลังงานที่ใช้สำหรับทำกิจกรรมการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน การเดิน จะแสดงดังตารางที่ 3.3 โดยใช้โปรแกรมในข้อหัวที่ 3.1.9 ในการคำนวณ หน้าต่างโปรแกรมที่ใช้สำหรับการเลือกเปิดไฟล์ .txt แสดงดังรูปที่ 4.32 และค่าพลังงานที่ใช้สำหรับทำกิจกรรมที่ได้รับการคำนวณแล้วแสดงดังรูปที่ 4.33



รูปที่ 4.32 หน้าต่างโปรแกรมสำหรับเลือกเปิดไฟล์ .text ที่จะใช้ในการคำนวณการใช้พลังงานจากกิจกรรมการเคลื่อนไหวของร่างกาย



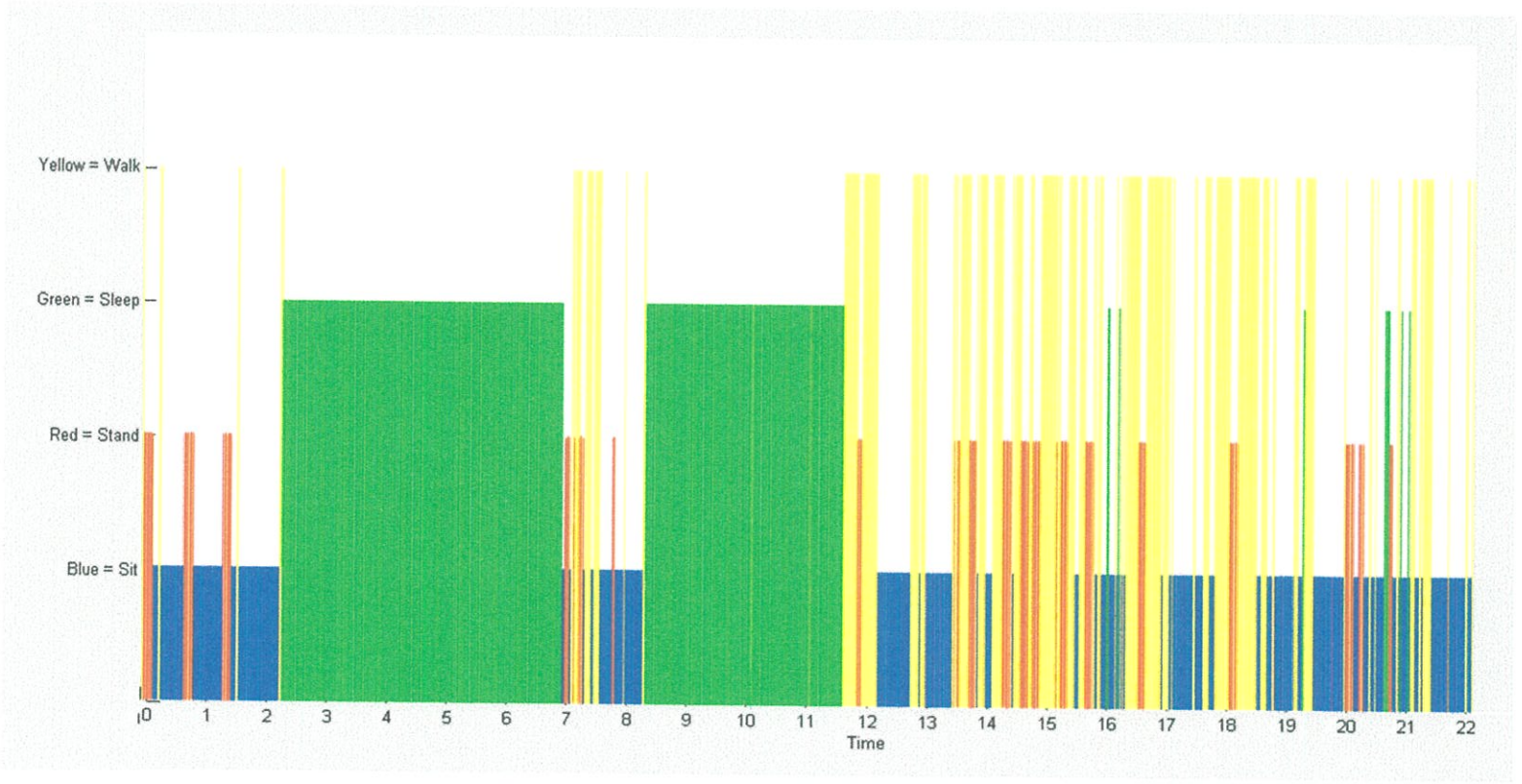
รูปที่ 4.33 ค่าพลังงานที่ใช้สำหรับทำกิจกรรมการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน เป็นเวลา 1 วัน มีค่าเท่ากับ 1755 กิโลแคลอรี่

ดังนั้นผลการทดลองเมื่อคำนวณการใช้พลังงานจากกิจกรรมการเคลื่อนไหวของร่างกาย ยืน นั่ง นอน และเดิน ใน 1 วัน สามารถสรุปได้ดังตารางที่ 4.4 ดังนี้

ตารางที่ 4.4 ค่าพลังงานที่ใช้สำหรับทำกิจกรรมการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน เป็นเวลา 1 วัน

ค่าที่คำนวณ (กิโลแคลอรี่)	ค่าที่เก็บผลได้จริง (กิโลแคลอรี่)	เปอร์เซ็นต์ความผิดพลาด
2207	1755	20.48%

เมื่อนำผลลักษณะการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน กับค่าของเวลาใน 1 วัน มาพล็อตกราฟ จะได้กราฟดังรูปที่ 4.34



รูปที่ 4.34 ลักษณะการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน กับค่าของเวลาใน 1 วัน

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

ปริญญานิพนธ์นี้นำเสนอการออกแบบและสร้างเครื่องบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน และการเดิน พร้อมด้วยระบบคำนวณอัตราการใช้พลังงานของร่างกายจากพฤติกรรมการใช้ชีวิตประจำวัน สามารถสรุปได้ดังนี้

1. ระบบสามารถรับค่าความเร่งเชิงเส้น 3 แกน จากโมดูลเซ็นเซอร์ LSM9DS0 ผ่านทางโปรแกรมอาร์ดูอิโน่
2. ระบบสามารถรับค่าของเวลา จากฐานเวลาโดยแสดงผลผ่านทางโปรแกรมอาร์ดูอิโน่
3. ระบบสามารถประมวลผลหาลักษณะการเคลื่อนไหวของร่างกายการยืน การนั่ง การนอน และการเดิน ผ่านทางโปรแกรมอาร์ดูอิโน่
4. ระบบสามารถแสดงผลค่ากิจกรรมการเคลื่อนไหวของร่างกายบนหน้าจอ LCD และสามารถแสดงผลค่าพลังงานที่ใช้ไปจากกิจกรรมการเคลื่อนไหวของร่างกายบนหน้าจอ LCD ผ่านทางโปรแกรมอาร์ดูอิโน่
5. ระบบสามารถบันทึกกิจกรรมการเคลื่อนไหวของร่างกาย การยืน การนั่ง การนอน การเดิน และค่าของเวลา เป็นเวลา 1 วัน และส่งข้อมูลไปจัดเก็บใน SD Card Module ผ่านทางโปรแกรมอาร์ดูอิโน่
6. ระบบสามารถคำนวณอัตราการใช้พลังงานของร่างกายจากกิจกรรมการเคลื่อนไหวของร่างกาย ใน 1 วัน ผ่านทางโปรแกรมแมทแลป
7. ระบบมีอัตราความผิดพลาดในการประมวลผลลักษณะการเคลื่อนไหวของร่างกาย คือ 20.48%
8. ระบบมี delay ในการประมวลผลลักษณะการเคลื่อนไหวของร่างกาย คือ 1.02 วินาที

5.2 ปัญหาและข้อเสนอแนะ

1) ในส่วนของในส่วนของฐานเวลา วงจรเรียลไทม์คล็อก (Real Time Clock) จะต้องป้อนไฟเลี้ยง 3 โวลต์ ให้วงจรตลอดเวลา เพื่อรักษาสภาพการทำงานให้วงจร

2) ในส่วนของโครงสร้างการเชื่อมต่ออุปกรณ์ เนื่องจากต้องใช้สายนำสัญญาณต่างๆ ในระยะทางที่ค่อนข้างยาวในการติดตั้งจริง จึงอาจเกิดการสูญเสียของสัญญาณภายในสายได้ จึงควรใช้สายสัญญาณให้สั้นที่สุด

3) ในส่วนของผลการทดลองที่ได้พบว่าเสถียรภาพของข้อมูลที่ได้ยังไม่ดีพอในการตัดสินลักษณะการเคลื่อนไหวของร่างกาย ควรปรับปรุงฟังก์ชั่นที่ใช้ในการตัดสินลักษณะการเคลื่อนไหวของร่างกายให้มีความแม่นยำมากที่สุด

บรรณานุกรม

[1] atmega328 datasheet. [ออนไลน์].เข้าถึงได้จาก:
http://atmega32-avr.com/atmega328-datasheet/

[2] ATMEGA328 Datasheet. [ออนไลน์].เข้าถึงได้จาก:
http://html.alldatasheet.com/html-pdf/392243/ATMEL/ATMEGA328/152/1/ATMEGA328.html

[3] หลักการสื่อสารข้อมูลเบื้องต้นและเครือข่ายคอมพิวเตอร์. [ออนไลน์].เข้าถึงได้จาก:
http://www.thaigoodview.com/node/131676?page=0,5

[4] การส่งผ่านข้อมูลดิจิตอล. [ออนไลน์].เข้าถึงได้จาก:
komvech.wpm.ac.th/file/computer-project/chapter5.pptx

[5] UART/TTL/RS232/MAX232/MAX3232. [ออนไลน์].เข้าถึงได้จาก:
http://www.thaieasyelec.com/article-wiki/basic-electronics/uart-ttl-rs232-max232-max3232.html

[6] MAX232. [ออนไลน์].เข้าถึงได้จาก:
http://www.engineersgarage.com/electronic-components/max232-datasheet

[7] ARDUINO. [ออนไลน์].เข้าถึงได้จาก: http://www.arduino.cc/

[8] การเชื่อมต่ออุปกรณ์แบบ I2C. [ออนไลน์].เข้าถึงได้จาก:
http://www.thaimicrotron.com/CCS-628/Referrence/I2CBUS.htm

[9] การใช้งานพอร์ตสื่อสาร SPI. [ออนไลน์].เข้าถึงได้จาก:
http://aimagin.com/blog/spi/?lang=th.

[10] Overview. [ออนไลน์].เข้าถึงได้จาก:
https://learn.adafruit.com/adafruit-lsm9ds0-accelerometer-gyro-magnetometer-9-dof-breakouts/overview

[11] การใช้งาน RTC (Real Time Clock) ด้วย DS1307. [ออนไลน์].เข้าถึงได้จาก:
http://www.mind-tek.net/ds1307.php.

[12] โมดูล Micro SD Card Micro SD Card Module MicroSD Card Adapter (Catalex). [ออนไลน์].เข้าถึงได้จาก: http://www.arduinoall.com/product/557/%E0%B9%82%E0%B8%A1%E0%B8%94%E0%B8%B9%E0%B8%A5-micro-sd-card-micro-sd-card-module-microsd-card-adapter-catalex

[13] CD4066B Datasheet. [ออนไลน์].เข้าถึงได้จาก: http://html.alldatasheet.com/html-pdf/26881/TI/CD4066B/21/1/CD4066B.html.http://aimagin.com/blog/spi/?lang=th.

[14] DS1307 Datasheet. [ออนไลน์].เข้าถึงได้จาก: http://html.alldatasheet.com/html-pdf/58481/DALLAS/DS1307/180/1/DS1307.html

[15] LiquidCrystal Library. [ออนไลน์].เข้าถึงได้จาก: http://arduino.cc/en/Tutorial/LiquidCrystal.

[16] LSM9DS0 Datasheet. [ออนไลน์].เข้าถึงได้จาก: http://www.adafruit.com/datasheets/LSM9DS0.pdf.

[17] การเชื่อมต่ออุปกรณ์แบบ I^2C. [ออนไลน์].เข้าถึงได้จาก: http://www.thaimicrotron.com/CCS-628/Referrence/I2CBUS.htm.

[18] Wiring & Test. [ออนไลน์].เข้าถึงได้จาก: https://learn.adafruit.com/adafruit-lsm9ds0-accelerometer-gyro-magnetometer-9-dof-breakouts/wiring-and-test.

[19] ดร. นิธิยา รัตนาปนนท์ และ ดร. วิบูลย์ รัตนาปนนท์ .หลักโภชนศาสตร์.พิมพ์ครั้งที่ 2. กรุงเทพฯ : โอเดียนสโตร์, 2556.

[20] ดร. ปิ่นมณี ขวัญเมือง.หลักโภชนาการ.พิมพ์ครั้งที่ 1. กรุงเทพฯ : สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2547.

[21] กันตพิชญ์ แก้วสุวรรณ และ ภมร ยงวัฒนานันท์. “ระบบจำแนกและติดตามอิริยาบถของผู้สูงอายุแบบไร้สาย”. ปริญญานิพนธ์ปริญญาวิศวกรรมศาสตรบัณฑิต,สาขาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2555

ภาคผนวก

Source Code

โปรแกรมรวม

```
#include <SPI.h> // Included for SFE_LSM9DS0 library
#include <Wire.h>
#include <SFE_LSM9DS0.h>
#include <LiquidCrystal.h>
#include <SD.h>
LiquidCrystal lcd(10, 9, 8, 7, 6, 3);
float A=0;
float B=0;
float C=0;
int D=0;
int E=0;
int F=0;
int G=0;
int H=0;
int SWpin = A0;   // select the input pin for the potentiometer      // select the pin
for the LED
int SWValue = 0;
String s1;
File myFile;
#define DS1307_ADDRESS 0x68


#define LSM9DS0_XM  0x1D // Would be 0x1E if SDO_XM is LOW
#define LSM9DS0_G   0x6B // Would be 0x6A if SDO_G is LOW
// Create an instance of the LSM9DS0 library called `dof` the
// parameters for this constructor are:
// [SPI or I2C Mode declaration],[gyro I2C address],[xm I2C add.]
LSM9DS0 dof(MODE_I2C, LSM9DS0_G, LSM9DS0_XM);
```

```
#define PRINT_CALCULATED
//#define PRINT_RAW

#define PRINT_SPEED 500 // 500 ms between prints
const int Pin1 = 2;
const int Pin2 = 5;
const int Pin3 = A0;
byte zero = 0x00; //workaround for issue #527
int tk;


const char *shown_m[12] =
{
  "Jan", "Feb", "Mar", "Apr", "May", "Jun",
  "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" };

const char *shown_d[7] =
{
  "Sun", "Mon", "Tue", "Wed", "Thu", "Fri",
  "Sat" };

void setup()
{

  pinMode(Pin1, OUTPUT);
  pinMode(Pin2, OUTPUT);
  digitalWrite(Pin1, HIGH);
  digitalWrite(Pin2,LOW );
```

```
  Serial.begin(9600); // Start serial at 115200 bps
  // Use the begin() function to initialize the LSM9DS0 library.
  // You can either call it with no parameters (the easy way):
  lcd.begin(16, 2);
  uint16_t status = dof.begin();
  // Or call it with declarations for sensor scales and data rates:
  //uint16_t status = dof.begin(dof.G_SCALE_2000DPS,
  //                            dof.A_SCALE_6G, dof.M_SCALE_2GS);

  // begin() returns a 16-bit value which includes both the gyro
  // and accelerometers WHO_AM_I response. You can check this to
  // make sure communication was successful.
    Serial.print("Initializing SD card...");
  // On the Ethernet Shield, CS is pin 4. It's set as an output by default.
  // Note that even if it's not used as the CS pin, the hardware SS pin
  // (10 on most Arduino boards, 53 on the Mega) must be left as an output
  // or the SD library functions will not work.
   pinMode(10, OUTPUT);

  if (!SD.begin(4)) {
    Serial.println("initialization failed!");
    return;
  }
  Serial.println("initialization done.");
  Serial.println();

  setDateTime();
}

void loop()
{
```

```
SWValue= analogRead(SWpin);
digitalWrite(Pin1, HIGH);
digitalWrite(Pin2, LOW);
printAccel(); // Print "A: ax, ay, az"
Serial.println();
delay(10);

digitalWrite(Pin2, HIGH);
digitalWrite(Pin1, LOW);
printDate();
Serial.println();
delay(1000);
//delay(PRINT_SPEED);

Serial.println (SWValue);
if (SWValue < 500) { // sw = HIGH
 if (s1 == "Stand"){
   E=E++;
 }
 else if (s1 == "Sit"){
   F=F++;
 }
 else if (s1 == "Sleep"){
   G=G++;
 }
 else if (s1 == "Walk"){
   H=H++;
 }
 Serial.println();
 Serial.println(E);
 Serial.println(F);
```

```
    Serial.println(G);
    Serial.println(H);
  }
  else if(SWValue >= 500) { // sw = LOW
    int temp = E + F + G + H;
    if (temp > 0) {
      double cal = calculateCal();
      showCalOnLCD(cal);
      delay(15000);
      E = 0;
      F = 0;
      G = 0;
      H = 0;
    }
  }
}

double calculateCal() {
  return E*0.0389 + F*0.0042 + G*0.0208 + H*0.0833;
}

void showCalOnLCD(double cal) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Enegy");
  lcd.setCursor(0, 1);
  lcd.print(cal);
  lcd.print(" kcal");
}

void printAccel()
```

```
{
  // To read from the accelerometer, you must first call the
  // readAccel() function. When this exits, it'll update the
  // ax, ay, and az variables with the most current data.
  dof.readAccel();

  // Now we can use the ax, ay, and az variables as we please.
  // Either print them as raw ADC values, or calculated in g's.

  lcd.clear();
  lcd.setCursor(9, 1);
  lcd.print(s1);
#ifdef PRINT_CALCULATED
  A=dof.calcAccel(dof.ax);
  B= dof.calcAccel(dof.ay);
  C=dof.calcAccel(dof.az);// g's. Give the function the value that you want to convert.
  D=SEE(A,B,C);

#endif

  if(SWValue < 500) {
    myFile = SD.open("test.txt", FILE_WRITE);
    // if the file opened okay, write to it:
    if (myFile) {
      myFile.println(s1);
      myFile.close();
    }
  }
}
```

```
void printDate(){
  Wire.beginTransmission(DS1307_ADDRESS);
  byte zero = 0x00;
  Wire.write(zero);
  Wire.endTransmission();
  Wire.requestFrom(DS1307_ADDRESS, 7);


  byte second = (Wire.read());
  int second_h = (second & B11110000) >> 4;
  int second_l = (second & B00001111);

  byte minute= (Wire.read());
  int minute_h = (minute & B11110000) >> 4;
  int minute_l = (minute & B00001111);

  byte hour = (Wire.read()); //24 hour time
  int hour_h = (hour & B11110000)  >> 4;
  int hour_l = (hour & B00001111);

  byte weekDay = (Wire.read()); //0-6 -> sunday - Saturday
  int weekDay_h = (weekDay & B11110000) >> 4;
  int weekDay_l = (weekDay & B00001111);

  byte monthDay = (Wire.read());
  int monthDay_h = (monthDay & B11110000) >> 4;
  int monthDay_l = (monthDay & B00001111);

  byte month = (Wire.read());
  int month_h = (month & B11110000) >> 4;
```

```
int month_l = (month & B00001111);

byte year = (Wire.read());
int year_h = (year & B11110000) >> 4;
int year_l = (year & B00001111);


Serial.print(monthDay_h);
Serial.print(monthDay_l);
Serial.print("/");
//Serial.print(shown_m[month-1]);
Serial.print(month_h);
Serial.print(month_l);
Serial.print("/");
Serial.print(year_h);
Serial.print(year_l);
Serial.print(" ");
Serial.print(hour_h);
Serial.print(hour_l);
Serial.print(":");
Serial.print(minute_h);
Serial.print(minute_l);
Serial.print(":");
Serial.print(second_h);
Serial.println(second_l);
//  lcd.clear();
lcd.setCursor(0, 0);
//lcd.print(shown_d[weekDay-1]);
//lcd.print(" ");
lcd.print(monthDay_h);
lcd.print(monthDay_l);
```

```
lcd.print("/");
//lcd.print(shown_m[month-1]);
lcd.print(month_h);
lcd.print(month_l);
lcd.print("/");
lcd.print(year_h);
lcd.print(year_l);
lcd.setCursor(0, 1);
lcd.print(hour_h);
lcd.print(hour_l);
lcd.print(":");
lcd.print(minute_h);
lcd.print(minute_l);
lcd.print(":");
lcd.print(second_h);
lcd.print(second_l);

if(SWValue < 500) {
 myFile = SD.open("test.txt", FILE_WRITE);
 // if the file opened okay, write to it:
 if (myFile) {
  myFile.print(monthDay_h);
  myFile.print(monthDay_l);
  myFile.print("/");
  myFile.print(shown_m[month-1]);
  myFile.print(month_h);
  myFile.print(month_l);
  myFile.print("/");
  myFile.print(year_h);
  myFile.print(year_l);
  myFile.print(" ");
```

```
      myFile.print(hour_h);
      myFile.print(hour_l);
      myFile.print(":");
      myFile.print(minute_h);
      myFile.print(minute_l);
      myFile.print(":");
      myFile.print(second_h);
      myFile.println(second_l); // close the file:
      myFile.close();

      //Serial.println("done.");
    }
  }

}

void setDateTime(){

 byte second =      59; //0-59
 byte minute =      40; //0-59
 byte hour =        21; //0-23
 byte weekDay =      2; //1-7
 byte monthDay =    11; //1-31
 byte month =        5; //1-12
 byte year  =       15; //0-99

 Wire.beginTransmission(DS1307_ADDRESS);
 Wire.write(zero); //stop Oscillator
 Wire.write(decToBcd(second));
 Wire.write(decToBcd(minute));
 Wire.write(decToBcd(hour));
```

```
  Wire.write(decToBcd(weekDay));
  Wire.write(decToBcd(monthDay));
  Wire.write(decToBcd(month));
  Wire.write(decToBcd(year));
  Wire.write(zero); //start
  Wire.endTransmission();

}

byte decToBcd(byte val){
  return ( (val/10*16) + (val%10) );
}


int SEE(float X,float Y,float Z)
{
  if ( Y >= 0.90 && Y <= 1.10 && Z >= -0.35 && Z <= -0.10 )
  {
    Serial.println("Stand");
    s1="Stand";
  }
  else if (X > 0.75 && X <= 0.95 && Y > -0.05 && Y <= 0.40 && Z > -0.40 && Z <= 0.15 )
  {
    Serial.println("Sit");
    s1="Sit";
  }
  else if (Y >= -0.20 && Y <= 0.30 && Z >= -0.40 && Z <= 0.90 )
  {
    Serial.println("Sleep");
    s1="Sleep";
  }
```

```
  else
  {
    Serial.println("Walk");
    s1="Walk";
  }
}
```

โปรแกรมอ่านค่าความเร่งเชิงเส้น 3 แกนจากโมดูลเซ็นเซอร์ LSM9DS0

```
#include <SPI.h> // Included for SFE_LSM9DS0 library
#include <Wire.h>
#include <SFE_LSM9DS0.h>
#define LSM9DS0_XM  0x1D // Would be 0x1E if SDO_XM is LOW
#define LSM9DS0_G   0x6B // Would be 0x6A if SDO_G is LOW
// Create an instance of the LSM9DS0 library called `dof` the
// parameters for this constructor are:
// [SPI or I2C Mode declaration],[gyro I2C address],[xm I2C add.]
LSM9DS0 dof(MODE_I2C, LSM9DS0_G, LSM9DS0_XM);


#define PRINT_CALCULATED
//#define PRINT_RAW

#define PRINT_SPEED 500 // 500 ms between prints

void setup()
{
 Serial.begin(9600); // Start serial at 115200 bps
 // Use the begin() function to initialize the LSM9DS0 library.
 // You can either call it with no parameters (the easy way):
 uint16_t status = dof.begin();
```

```
  // Or call it with declarations for sensor scales and data rates:
  //uint16_t status = dof.begin(dof.G_SCALE_2000DPS,
  //                            dof.A_SCALE_6G, dof.M_SCALE_2GS);

  // begin() returns a 16-bit value which includes both the gyro
  // and accelerometers WHO_AM_I response. You can check this to
  // make sure communication was successful.
  Serial.print("LSM9DS0 WHO_AM_I's returned: 0x");
  Serial.println(status, HEX);
  Serial.println("Should be 0x49D4");
  Serial.println();
}

void loop()
{

  printAccel(); // Print "A: ax, ay, az"

  Serial.println();

  delay(PRINT_SPEED);
}

void printAccel()
{
  // To read from the accelerometer, you must first call the
  // readAccel() function. When this exits, it'll update the
  // ax, ay, and az variables with the most current data.
  dof.readAccel();

  // Now we can use the ax, ay, and az variables as we please.
```

```
  // Either print them as raw ADC values, or calculated in g's.
  Serial.print("A: ");
#ifdef PRINT_CALCULATED
  // If you want to print calculated values, you can use the
  // calcAccel helper function to convert a raw ADC value to
  // g's. Give the function the value that you want to convert.
  Serial.print(dof.calcAccel(dof.ax), 2);
  Serial.print(", ");
  Serial.print(dof.calcAccel(dof.ay), 2);
  Serial.print(", ");
  Serial.println(dof.calcAccel(dof.az), 2);
#elif defined PRINT_RAW
  Serial.print(dof.ax);
  Serial.print(", ");
  Serial.print(dof.ay);
  Serial.print(", ");
  Serial.println(dof.az);
#endif

}
```

โปรแกรมอ่านค่าเวลาและวันที่จากวงจรเรียลไทม์คล็อก (Real Time Clock)

```
#include "Wire.h"
#define DS1307_ADDRESS 0x68
byte zero = 0x00; //workaround for issue #527
```

```
 int tk;


 const char *shown_m[12] =
{ "Jan", "Feb", "Mar", "Apr", "May", "Jun",
  "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" };

 const char *shown_d[7] =
{ "Sun", "Mon", "Tue", "Wed", "Thu", "Fri",
  "Sat" };

 void setup(){
  Wire.begin();
  Serial.begin(9600);

  setDateTime();
  //}
}
 void loop(){
  printDate();
  delay(1000);
  }

byte bcdToDec(byte val)  {
// Convert binary coded decimal to normal decimal numbers
  return ( (val/16*10) + (val%16) );
}

void printDate(){
  Wire.beginTransmission(DS1307_ADDRESS);
  byte zero = 0x00;
```

```
Wire.write(zero);
Wire.endTransmission();
Wire.requestFrom(DS1307_ADDRESS, 7);


byte second = (Wire.read());
int second_h = (second & B11110000) >> 4;
int second_l = (second & B00001111);

byte minute= (Wire.read());
int minute_h = (minute & B11110000) >> 4;
int minute_l = (minute & B00001111);

byte hour = (Wire.read()); //24 hour time
int hour_h = (hour & B11110000)  >> 4;
int hour_l = (hour & B00001111);

byte weekDay = (Wire.read()); //0-6 -> sunday - Saturday
int weekDay_h = (weekDay & B11110000) >> 4;
int weekDay_l = (weekDay & B00001111);

byte monthDay = (Wire.read());
int monthDay_h = (monthDay & B11110000) >> 4;
int monthDay_l = (monthDay & B00001111);

byte month = (Wire.read());
int month_h = (month & B11110000) >> 4;
int month_l = (month & B00001111);

byte year = (Wire.read());
int year_h = (year & B11110000) >> 4;
```

```
  int year_l = (year & B00001111);

  Serial.print(monthDay_h);
  Serial.print(monthDay_l);
  Serial.print("/");
  Serial.print(shown_m[month-1]);
  //Serial.print(month_h);
  //Serial.print(month_l);
  Serial.print("/");
  Serial.print(year_h);
  Serial.print(year_l);
  Serial.print(" ");
  Serial.print(hour_h);
  Serial.print(hour_l);
  Serial.print(":");
  Serial.print(minute_h);
  Serial.print(minute_l);
  Serial.print(":");
  Serial.print(second_h);
  Serial.println(second_l);

}

void setDateTime(){

  byte second =      59; //0-59
  byte minute =      07; //0-59
  byte hour =        22; //0-23
  byte weekDay =      4; //1-7
  byte monthDay =     1; //1-31
  byte month =       10; //1-12
```

```
  byte year  =       14; //0-99

  Wire.beginTransmission(DS1307_ADDRESS);
  Wire.write(zero); //stop Oscillator
  Wire.write(decToBcd(second));
  Wire.write(decToBcd(minute));
  Wire.write(decToBcd(hour));
  Wire.write(decToBcd(weekDay));
  Wire.write(decToBcd(monthDay));
  Wire.write(decToBcd(month));
  Wire.write(decToBcd(year));
  Wire.write(zero); //start
  Wire.endTransmission();

}

byte decToBcd(byte val){
  return ( (val/10*16) + (val%10) );
}
```

โปรแกรมอ่านและบันทึกข้อมูลลงใน Micro SD Card Module

```
#include <SD.h>

File myFile;
```

```
void setup()
{
 // Open serial communications and wait for port to open:
  Serial.begin(9600);
   while (!Serial) {
    ;
  }


  Serial.print("Initializing SD card...");

   pinMode(10, OUTPUT);

  if (!SD.begin(4)) {
    Serial.println("initialization failed!");
    return;
  }
  Serial.println("initialization done.");


  myFile = SD.open("test.txt", FILE_WRITE);

  // if the file opened okay, write to it:
  if (myFile) {
    Serial.print("Writing to test.txt...");
    myFile.println("testing 1, 2, 3.");
          // close the file:
    myFile.close();
    Serial.println("done.");
  } else {
```

```
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }

  // re-open the file for reading:
  myFile = SD.open("test.txt");
  if (myFile) {
    Serial.println("test.txt:");

    // read from the file until there's nothing else in it:
    while (myFile.available()) {
          Serial.write(myFile.read());
    }
    // close the file:
    myFile.close();
  } else {
        // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }
}

void loop(){
}
```

<u>โปรแกรมคำนวณการใช้พลังงานจากกิจกรรมการเคลื่อนไหวของร่างกาย ยืน นั่ง นอน และเดิน ในแมทแลป</u>

```
function varargout = Energy(varargin)
% ENERGY MATLAB code for Energy.fig
%      ENERGY, by itself, creates a new ENERGY or raises the existing
%      singleton*.
%
```

```
%      H = ENERGY returns the handle to a new ENERGY or the handle to
%      the existing singleton*.
%
%      ENERGY('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ENERGY.M with the given input arguments.
%
%      ENERGY('Property','Value',...) creates a new ENERGY or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Energy_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Energy_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Energy

% Last Modified by GUIDE v2.5 13-May-2015 02:28:00

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Energy_OpeningFcn, ...
                   'gui_OutputFcn',  @Energy_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
```

```
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Energy is made visible.
function Energy_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Energy (see VARARGIN)

% Choose default command line output for Energy
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Energy wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Energy_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
```

```
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[filename pathname] = uigetfile({'*.txt'},'File Selector');
fullpathname = strcat(pathname,filename);
d = textread(fullpathname, '%s');
set(handles.text2,'string', fullpathname)
%%
% Sit Stand Sleep Walk
Sit = 0;
Stand = 0;
Sleep = 0;
Walk = 0;
for i=1:length(d)
   switch d{i}
      case 'Sit'
         Sit = Sit + 1;
         d{i} = 1;
      case 'Stand'
         Stand = Stand + 1;
         d{i} = 2;
```

```
        case 'Sleep'
            Sleep = Sleep + 1;
            d{i} = 3;
        case 'Walk'
            Walk = Walk + 1;
            d{i} = 4;
    end
end
calStand = Stand*0.0389;
calSit = Sit*0.0042;
calSleep = Sleep*0.0208;
calWalk = Walk*0.0833;

totalCalories = calStand + calSit + calSleep + calWalk;
%text = 'ST = %d sec\nSU = %d sec\nSL = %d sec\nWK = %d sec\n\n', Sit, Stand,
Sleep, Walk;
v = sprintf('Stand = %d sec\nSit = %d sec\nSleep = %d sec\nWalk = %d sec', Stand,
Sit, Sleep, Walk);
v = strcat(v,sprintf('\n\nStand = %f kcal\nSit   = %f kcal\nSleep = %f kcal\nWalk  = %f
kcal\n\nTotal : %f kcal\n\n', calStand, calSit, calSleep, calWalk, totalCalories));
set(handles.text3,'string', v)
%%

act = [];
time = [];
j = 1;
for i=1:3:length(d)
    if (length(d{i}) == 1)
        act = [act d{i}];
        time{j} = d{i+2};
        j = j + 1;
```

```
    end
end

temp = act;
temp(temp~=1) = NaN;
act1 = temp;
temp = act;
temp(temp~=2) = NaN;
act2 = temp;
temp = act;
temp(temp~=3) = NaN;
act3 = temp;
temp = act;
temp(temp~=4) = NaN;
act4 = temp;

clear d, temp;

for i=1:length(time)
    % split % convert % sum
    tempCell = textscan(time{i},'%d', 'Delimiter', ':');
    time{i} = tempCell{1}(1)*3600 + tempCell{1}(2)*60 + tempCell{1}(3);
end

time = cell2mat(time);
wholeDay = zeros(1,24*3600);
wholeDay_act1 = wholeDay;
wholeDay_act2 = wholeDay;
wholeDay_act3 = wholeDay;
wholeDay_act4 = wholeDay;
```

```
wholeDay_act1(time) = act1;
wholeDay_act2(time) = act2;
wholeDay_act3(time) = act3;
wholeDay_act4(time) = act4;

figure, hold on, bar((1:86400)/1000,wholeDay_act1,'b','EdgeColor','b'),
bar((1:86400)/1000,wholeDay_act2,'r', 'EdgeColor','r'),
bar((1:86400)/1000,wholeDay_act3,'g', 'EdgeColor','g'),
bar((1:86400)/1000,wholeDay_act4,'y', 'EdgeColor','y'), axis([0 length(wholeDay)/1000 0
5]);
set(gca,'XTick', (1:600:86400)/1000);
set(gca,'XTickLabel', {  '00:00','00:10','00:20','00:30','00:40','00:50' ...
                        ,'01:00','01:10','01:20','01:30','01:40','01:50' ...
                        ,'02:00','02:10','02:20','02:30','02:40','02:50' ...
                        ,'03:00','03:10','03:20','03:30','03:40','03:50' ...
                        ,'04:00','04:10','04:20','04:30','04:40','04:50' ...
                        ,'05:00','05:10','05:20','05:30','05:40','05:50' ...
                        ,'06:00','06:10','06:20','06:30','06:40','06:50' ...
                        ,'07:00','07:10','07:20','07:30','07:40','07:50' ...
                        ,'08:00','08:10','08:20','08:30','08:40','08:50' ...
                        ,'09:00','09:10','09:20','09:30','09:40','09:50' ...
                        ,'10:00','10:10','10:20','10:30','10:40','10:50' ...
                        ,'11:00','11:10','11:20','11:30','11:40','11:50' ...
                        ,'12:00','12:10','12:20','12:30','12:40','12:50' ...
                        ,'13:00','13:10','13:20','13:30','13:40','13:50' ...
                        ,'14:00','14:10','14:20','14:30','14:40','14:50' ...
                        ,'15:00','15:10','15:20','15:30','15:40','15:50' ...
                        ,'16:00','16:10','16:20','16:30','16:40','16:50' ...
                        ,'17:00','17:10','17:20','17:30','17:40','17:50' ...
                        ,'18:00','18:10','18:20','18:30','18:40','18:50' ...
                        ,'19:00','19:10','19:20','19:30','19:40','19:50' ...
```

```
                ,'20:00','20:10','20:20','20:30','20:40','20:50' ...
                ,'21:00','21:10','21:20','21:30','21:40','21:50' ...
                ,'22:00','22:10','22:20','22:30','22:40','22:50' ...
                ,'23:00','23:10','23:20','23:30','23:40','23:50' });
set(gca,'YTick', 1:1:4)
set(gca,'YTickLabel',{'Blue = Sit', 'Red = Stand', 'Green = Sleep', 'Yellow = Walk'})
xlabel('Time');
```