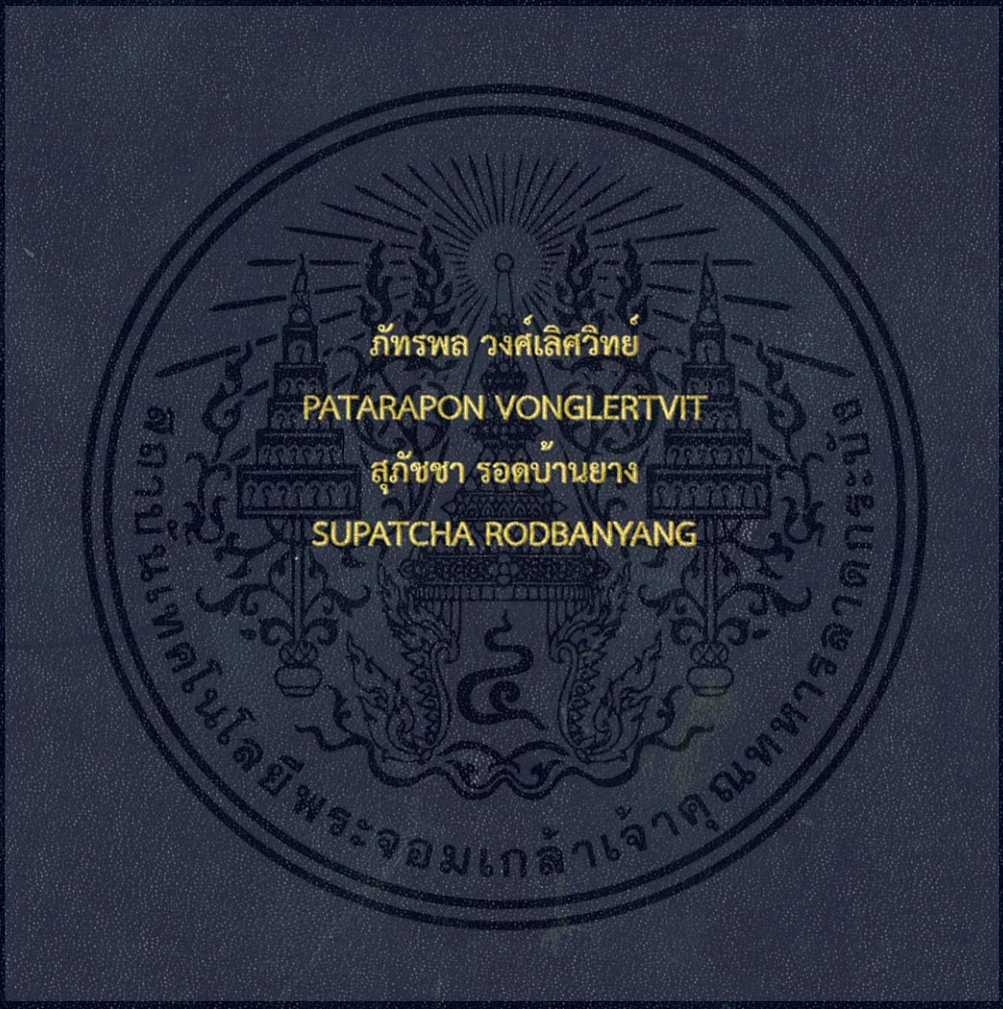
การพัฒนาเครื่องฝึกการหลบหมัด
Development of Slip Punches Training Machine (SPTM)



ภัทรพล วงศ์เลิศวิทย์

PATARAPON VONGLERTVIT

สุภัชชา รอดบ้านยาง

SUPATCHA RODBANYANG

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2557

การพัฒนาเครื่องฝึกการหลบหมัด
Development of Slip Punches Training Machine (SPTM)

โดย

ภัทรพล วงศ์เลิศวิทย์

สุภัชชา รอดบ้านยาง

อาจารย์ที่ปรึกษา

ผศ.ดร. กิติพล ชิตสกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2557

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2557

สาขาวิชา อิเล็กทรอนิกส์

คณะ วิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาเครื่องฝึกการหลบหมัด

Development of Slip Punches Training Machine (SPTM)

ผู้จัดทำ

นายภัทรพล วงศ์เลิศวิทย์ รหัสนักศึกษา 54010975

นางสาวสุภัชชา รอดบ้านยาง รหัสนักศึกษา 54011417

ปริญญานิพนธ์นี้ผ่านการตรวจโดยอาจารย์ที่ปรึกษาแล้ว

(ผศ. ดร. กิติพล ชิตสกุล)

อาจารย์ที่ปรึกษา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การพัฒนาเครื่องฝึกการหลบหมัด

นายภัทรพล วงศ์เลิศวิทย์ รหัส 54010975
นางสาวสุภัชชา รอดบ้านยาง รหัส 54011417
ผศ.ดร.กิติพล ชิตสกุล อาจารย์ที่ปรึกษา
ปีการศึกษา 2557

บทคัดย่อ

รายงานฉบับนี้นำเสนอผลการพัฒนาเครื่องทดสอบสมรรถภาพของนักกีฬาซึ่งวัดเวลาตอบสนองระหว่างตากับมือเป็นการทดสอบความเร็วในการโยกหลบ ระบบยังคงใช้ LED วงแหวนเป็นสิ่งกระตุ้น ผู้ใช้จะต้องปิดแสงด้วยการโยกศีรษะหลบในทิศทางตรงข้ามกับทิศทางแสง โดยใช้เซ็นเซอร์ตรวจจับการเคลื่อนไหวในสามทิศทางได้แก่โยกไปด้านหลัง ด้านซ้ายและด้านขวา ข้อมูลจากเซ็นเซอร์จะถูกส่งไปยังบอร์ดVX-propeller ด้วยโมดูลไร้สาย nRF24L01 รูปแบบในการทดสอบจะแบ่งเป็น 2 โหมด คือ โหมดโจมตี (Attack Mode) และโหมดป้องกัน (Defense Mode) ทั้งสองโหมดสามารถเลือกโหมดย่อยได้อีก 3 โหมดคือ โหมดสุ่มตำแหน่ง (Random Mode) โหมดรูปแบบเฉพาะ (Pattern Mode) และโหมดกำหนดเอง (manual mode) ค่าเวลาตอบสนองแสดงบนจอภาพ VGA แล้วบันทึกผลการทดสอบที่ได้ลงใน SD CARD เพื่อเก็บไว้เป็นฐานข้อมูลและนำไปใช้วิเคราะห์ผลต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Development of Slip Punches Training Machine (SPTM)

Patarapon Vonglertvit 54010975
Supatcha Rodbanyang 54011417
Asst.Prof. Dr.KitipholChitsakul (Advisor)
Academic Year 2014

Abstract

SPTM has been developed for reaction time assessment which is used for the sportive performance evaluation in modern sport science. By using light as stimulus, athletes have to response by slip punching. By using an accelerometer as sensor to detect the movement in response, the data acquired are sent via the nRF24L01, a wireless module, to the VX- propeller. Two Modes of test including attack and defense mode are provided. Also 3 activating modes random, pattern and manual mode have been included. The testing results are reported on VGA screen.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

โครงงานชิ้นนี้สำเร็จได้ด้วยความช่วยเหลือจากอาจารย์ ผศ. ดร. กิติพล ชิตสกุล ซึ่งเป็นที่ปรึกษาในการทำโครงงาน และนายชัชชัย คางเป็นผู้ให้คำแนะนำและให้คำปรึกษาในการทาโครงงาน และช่วยหาวิธีแก้ปัญหาและอุปสรรคที่เกิดขึ้น

ขอบคุณอาจารย์ทุกท่านที่ช่วยสอนความรู้ให้ และขอบคุณรุ่นพี่รุ่นน้องและเพื่อนๆที่ได้ให้ความช่วยเหลือด้วยความหวังดีมาตลอด

สุดท้ายนี้ ขอขอบคุณคนในครอบครัวที่คอยเป็นกำลังใจที่ดีตลอดมา และเป็นแรงบันดาลใจที่ทำให้โครงงานชิ้นนี้สำเร็จลุล่วงได้ด้วยดี

ภัทรพล วงศ์เลิศวิทย์

(นายภัทรพล วงศ์เลิศวิทย์)

สุภัชชา รอดบ้านยาง

(นางสาวสุภัชชา รอดบ้านยาง)

ผู้จัดทำ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เรื่อง **หน้า**

บทคัดย่อ.....I
Abstract.....II
กิตติกรรมประกาศ.....III
สารบัญ.....IV-VI
สารบัญรูป.....VII-VIII
สารบัญตาราง.....IX
บทที่ 1 บทนำ.....1
1.1 ความเป็นมาของโครงงาน.....1
1.2 วัตถุประสงค์.....1
1.3 แนวคิดของระบบ.....1
1.4 โครงสร้างของรายงาน.....2
บทที่ 2 ทฤษฎีที่เกี่ยวข้อง.....3
2.1 การทดสอบไหวพริบของ Development of Slip Punches Training Machine.....3
2.2 ทฤษฎีทางด้าน Hardware.....3
2.2.1 PIC.....3
2.2.2 ตัวแปลงสัญญาณ RS-232 Level Converter.....4
2.2.3 Silicon Controlled Rectifier (SCR).....5
2.2.4 รีเลย์ (Relay).....5
2.3 ทฤษฎีทางด้าน Software.....6
2.3.1 ภาษาที่ใช้สำหรับไมโครคอนโทรลเลอร์ควบคุม VX-Propeller.....6
2.3.2 ภาษาที่ใช้สำหรับไมโครคอนโทรลเลอร์ PIC.....6
2.3.3 ไทเมอร์ และ เคาน์เตอร์.....7
2.3.4 อินเตอร์รัปต์.....8
2.4 ทฤษฎีพื้นฐานการสื่อสารแบบอนุกรม.....9
2.4.1 UART (Universal Asynchronous Receiver Transmitter).....9
2.4.2 มาตรฐาน RS-232.....9
2.4.3 มาตรฐาน RS-422 หรือ 422-A.....10
2.4.4 มาตรฐาน RS-485.....10
2.4.5 องค์ประกอบของการรับส่งข้อมูลแบบอนุกรม.....10
2.5 VX-Propeller.....11
2.5.1 คุณสมบัติเด่นของโพรเพลเลอร์.....11
2.5.2 คุณสมบัติทางเทคนิคของโพรเพลเลอร์.....12
2.5.3 หลักการทำงานของโพรเพลเลอร์.....13

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

เรื่อง **หน้า**

2.6 ระบบการตรวจสอบตนเอง (Self-testing).....18
2.7 Arduino Nano.....18
2.7.1 คุณสมบัติทั่วไป.....18
2.8 nRF24L01.....20
2.8.1 คุณสมบัติทั่วไป.....20
2.9 Accelerometer และ ADXL335.....21
2.9.1 คุณสมบัติทั่วไปของ ADXL335.....24
บทที่ 3 การออกแบบ.....25
3.1 การออกแบบโหมดทดสอบนักกีฬา.....25
3.1.1 โหมดโจมตี (Attack Mode).....27
3.1.2 โหมดป้องกัน (Defense Mode).....27
3.2 โหมดสอบย่อย.....28
3.2.1 โหมดการทดสอบแบบสุ่มตำแหน่ง (RANDOM MODE).....28
3.2.2 โหมดการทดสอบรูปแบบเฉพาะ (PATTERN MODE).....29
3.2.3 โหมดการทดสอบแบบกำหนดเอง (MANUAL MODE).....30
3.3 การออกแบบด้าน Hardware.....31
3.3.1 วงจรรับส่งสัญญาณไร้สาย.....31
3.3.2 วงจร PIC 16F688.....32
3.3.3 วงจรโมดูลปุ่มกด.....33
3.3.4 วงจร MAX 232.....33
3.3.5 วงจร PIC 16F688.....34
3.3.6 วงจร Comparator.....34
3.4 การออกแบบด้าน Software.....35
3.4.1 หลักการทำงานของโครงงานด้าน Software.....35
3.4.2 การคำนวณการนับโอเวอร์โฟลว์โดยใช้ TIMER 0 ของไมโครคอนโทรลเลอร์ PIC.....40
3.4.3 การชดเชยค่าเวลา.....41
3.4.4 หลักการทำงานของระบบการตรวจสอบตนเอง.....42
3.4.5 การคำนวณค่า G.....46
บทที่ 4 การทดลองและผลการทดลอง.....47
4.1 การส่งสัญญาณแบบไร้สาย.....47
4.2 วงจร Comparator.....48
4.3 การบันทึกผลใน SD Card.....48
4.4 ความแม่นยำในการจับเวลา.....49

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

เรื่อง	หน้า
บทที่ 5 บทสรุป	51
บรรณานุกรม	52
ภาคผนวก	53

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

รูป **หน้า**

รูปที่ 1 แนวคิดของระบบ....1
รูปที่ 2.1 รูปแสดงถึงความจำเป็นในการฝึกซ้อมการตอบสนอง....3
รูปที่ 2.2 ตัวอย่างของ PIC แบบ EPROM....4
รูปที่ 2.3 IC MAX 232....4
รูปที่ 2.4 สัญลักษณ์และโครงสร้างของ SCR....5
รูปที่ 2.5 ลักษณะและสัญลักษณ์ของรีเลย์....6
รูปที่ 2.6 แสดงตำแหน่งขาและรูปร่างของไมโครคอนโทรลเลอร์โพรเพลเลอร์แบบตัวถัง DIP 40 ขา....12
รูปที่ 2.7 บล็อกไดอะแกรมแสดงการทำงานภายในของโปรเซสเซอร์ทั้ง 8 ตัวของโพรเพลเลอร์....14
รูปที่ 2.8 บอร์ด VX-Propeller และรายละเอียดตำแหน่งอุปกรณ์ต่างๆ ที่ติดตั้งบนบอร์ด....15
รูปที่ 2.9 แสดงวงจรสมบูรณ์ของบอร์ด VX-Propeller (version 1.5)....17
รูปที่ 2.10 รูปบอร์ด Arduino Nano....18
รูปที่ 2.11 รูปแสดงตำแหน่งขาของบอร์ด Arduino Nano....19
รูปที่ 2.12 แสดงวงจรสมบูรณ์ของ โมดูล nRF24L01....20
รูปที่ 2.13 Block diagram ของ โมดูล nRF24L01....21
รูปที่ 2.14 ลักษณะการตรวจวัดความเร่ง....22
รูปที่ 2.15 โครงสร้างพื้นฐานของมิเตอร์วัดอัตราเร่งแบบไซซมิกแมส....22
รูปที่ 2.16 โครงสร้างพื้นฐานของมิเตอร์วัดอัตราเร่งแบบเพียโซอิเล็กทริก....23
รูปที่ 2.17 Block diagram ของ ADXL335....24
รูปที่ 3.1 โครงสร้างการทำงานของระบบ....25
รูปที่ 3.2 การติดไฟของปุ่มกด....26
รูปที่ 3.3 การส่งค่า Reaction time กลับมายังภาค MASTER และการแสดงผลทางจอภาพ VGA....26
รูปที่ 3.4 การทำงานของโหมดป้องกันขณะเปิดไฟโมดูลปุ่มกด....27
รูปที่ 3.5 การทำงานของโหมดป้องกันเมื่อมีการตอบสนอง....28
รูปที่ 3.6 การทำงานของระบบในโหมดการทดสอบแบบสุ่มตำแหน่ง....28
รูปที่ 3.7 การเปิดไฟล์รูปแบบเพื่อนำไปใช้ในโหมดการทดสอบรูปแบบเฉพาะ....29
รูปที่ 3.8 การทำงานของระบบในโหมดการทดสอบรูปแบบเฉพาะ....29
รูปที่ 3.9 แสดงการทดสอบในโหมดการทดสอบแบบกำหนดเอง....30
รูปที่ 3.10 การทำงานของระบบในโหมดการทดสอบแบบกำหนดเอง....30
รูปที่ 3.11 โครงสร้างของวงจรรับส่งสัญญาณไร้สาย....31
รูปที่ 3.12 วงจรภาคส่งสัญญาณไร้สาย....31
รูปที่ 3.13 วงจรภาครับสัญญาณไร้สาย....32
รูปที่ 3.14 วงจรรวมของPIC 16F688 เพื่อควบคุม Module....32
รูปที่ 3.15 วงจรของโมดูลปุ่มกด....33

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป(ต่อ)

รูป **หน้า**

รูปที่ 3.16 วงจรของ IC MAX232....33
รูปที่ 3.17 วงจรรวมของPIC 16F688 เพื่อควบคุม Module....34
รูปที่ 3.18 วงจร Comparator....34
รูปที่ 3.19 หลักการทำงานของระบบ....35
รูปที่ 3.20 หลักการทำงานของระบบ (ต่อ)....36
รูปที่ 3.21 หลักการทำงานของระบบในโหมดป้องกัน....37
รูปที่ 3.22 หลักการทำงานของระบบ (ต่อ)....37
รูปที่ 3.23 โฟลว์ชาร์ตการทำงานของ Vx-Propeller....38
รูปที่ 3.24 โฟลว์ชาร์ตการทำงานของโปรแกรมย่อย....39
รูปที่ 3.25 โฟลว์ชาร์ตการทำงานของไมโครคอนโทรลเลอร์ของโมดูลปุ่มกด (Slave 16F688)....40
รูปที่ 3.26 a) – b) แสดงความผิดพลาดในการนับโอเวอร์โฟลว์ของไมโครคอนโทรลเลอร์....41
รูปที่ 3.26 c) – e) แสดงความผิดพลาดทั้งหมดที่เกิดขึ้นในการจับเวลาของไมโครคอนโทรลเลอร์....41
รูปที่ 3.27 หลักการทำงานของระบบตรวจสอบตนเอง....43
รูปที่ 3.28 หลักการทำงานของระบบตรวจสอบตนเอง (ต่อ)....44
รูปที่ 3.29 หลักการทำงานของระบบตรวจสอบตนเอง (ต่อ)....45
รูปที่ 3.30 หลักการทำงานของระบบตรวจสอบตนเอง (ต่อ)....45
รูปที่ 3.31 ค่า G ในแนวแกน x y และ z ของ ADXL335....46
รูปที่ 4.1 วงจรตัวส่งสัญญาณไร้สายส่วนภาคส่ง....47
รูปที่ 4.2 วงจรตัวส่งสัญญาณไร้สายส่วนภาครับ....47
รูปที่ 4.3 serial monitor ของภาคส่งจากโปรแกรม Arduino....47
รูปที่ 4.4 serial monitor ของภาครับจากโปรแกรม Arduino....47
รูปที่ 4.5 สัญญาณอินพุทเทียบกับสัญญาณเอ้าท์พุทของวงจร Comparator....48
รูปที่ 4.6 Folder ภายใน SD Card....48
รูปที่ 4.7 ไฟล์ SPTM_xxxx.CSV ผลการทดสอบภายใน Folder RECORD....49
รูปที่ 4.8 การบันทึกผลการทดสอบลงในไฟล์ SPTM_xxxx.CSV....49

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

ตาราง **หน้า**

ตารางที่ 2.1 แสดงรายละเอียดหน้าที่การทำงานของขาต่างๆของโพรเพลเลอร์.....13
ตารางที่ 2.2 แสดงรายละเอียดหน้าที่การทำงานของขาต่างๆของ Arduino Nano.....19
ตารางที่ 4.1 ความคลาดเคลื่อนโหมดโจมตี.....50
ตารางที่ 4.2 ความคลาดเคลื่อนโหมดป้องกัน.....50

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 ความเป็นมาของโครงงาน

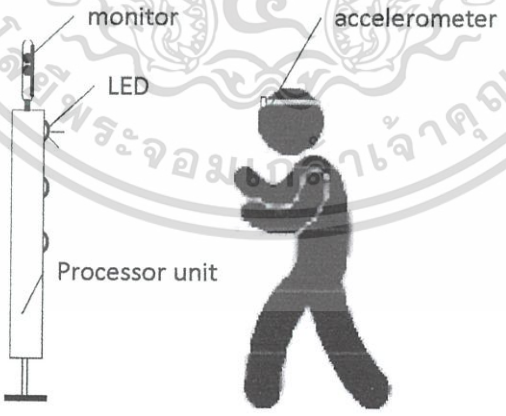
โครงงาน Development of Slip Punches Training Machine (SPTM) หรือการพัฒนาเครื่องฝึกการหลบหมัด เป็นการพัฒนาอุปกรณ์ฝึกซ้อมและทดสอบสมรรถนะของนักกีฬาในรูปแบบปฏิกิริยาตอบสนองระหว่างมือกับตาและความรวดเร็วในการเคลื่อนไหวร่างกาย ในกีฬาบางประเภทโดยเฉพาะการต่อสู้เช่นมวยสากลนั้น เอาชนะคู่ต่อสู้จำเป็นต้องมีปฏิกิริยาในการหลบหลีกและตอบโต้โต้ตอบจากคู่ต่อสู้ ซึ่งการฝึกซ้อมแบบเดิมจำเป็นต้องมีคู่ซ้อม แนวคิดนี้เป็นการพัฒนาอุปกรณ์ขึ้นเพื่อช่วยในการฝึกซ้อมกีฬาโดยไม่ต้องอาศัยคู่ซ้อมและยังสามารถใช้ในการประเมินสมรรถนะของนักกีฬาได้อีกด้วย

1.2 วัตถุประสงค์

พัฒนาเครื่องวัดและประเมินความเร็วในการหลบหมัด เพื่อใช้ในการทดสอบความคล่องตัวของนักมวยหรือนักกีฬาที่ต้องการทดสอบและการตอบสนองระหว่างสายตากับร่างกายของผู้ทดสอบ

1.3 แนวคิดของระบบ

ใช้แสงจาก LED เป็นสิ่งเร้า (แทนการส่งหมัดหรืออาวุธจากคู่ต่อสู่) วัดความรวดเร็วของนักกีฬาในการหลบศีรษะไปทิศตรงข้ามกับตำแหน่งของแสงไฟ โดยจะถูกตรวจสอบทิศทางจาก Accelerometerซึ่งติดตั้งบนหมวกป้องกันศีรษะ(head guard) วัดเวลาการตอบสนองด้วยไมโครคอนโทลเลอร์และแสดงผลผ่านทางจอภาพ VGA แสดงในรูปที่ 1



รูปที่ 1 แนวคิดของระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.4 โครงสร้างของรายงาน

ผลที่ได้จากการค้นคว้าทฤษฎี ที่เกี่ยวข้อง การสร้างและทดสอบได้นำมารายงานในรายงานฉบับนี้ซึ่งมีเนื้อหาแบ่งออกเป็น 5 บท แต่ละบทจะมีเนื้อหา ดังนี้

บทที่ 1 กล่าวถึง ที่มา และจุดประสงค์ของโครงงานนี้

บทที่ 2 กล่าวถึง ทฤษฎีต่างๆ ที่เกี่ยวข้องกับโครงงานนี้

บทที่ 3 กล่าวถึง แนวทางในการออกแบบ

บทที่ 4 กล่าวถึง ผลการทดสอบการทำงาน

บทที่ 5 กล่าวถึง บทสรุปของโครงงาน

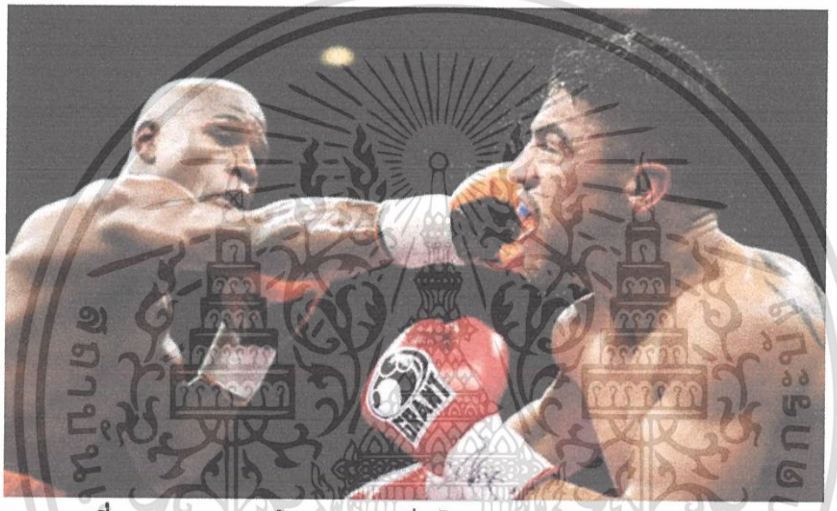
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่2

ทฤษฎีที่เกี่ยวข้อง

2.1 การทดสอบไหวพริบของ Development of Slip Punches Training Machine

การทดสอบไหวพริบของ SPTM นั้นเป็นแบบการฝึกการออกกำลังกายแบบจำเพาะต่อความไวของสายตาและการสั่งการของสมองโดยผู้เข้ารับการทดสอบจะใช้ความไวในการใช้หัวเคลื่อนที่ใน 3 ทิศทางต่อเนื่องกันไป เช่น ไฟติดด้านซ้ายให้หลบขวา ไฟติดด้านขวาให้หลบซ้าย โดยการทดสอบนั้นอยู่ในบริเวณที่จำกัดในระยะใกล้ซึ่งมีความจำเป็นสำหรับกีฬาหลายประเภทโดยเฉพาะนักกีฬามวย พร้อมกับการจับเวลา และใช้เวลาในรอบที่น้อยที่สุดเป็นผลการทดสอบ



รูปที่ 2.1 รูปแสดงถึงความจำเป็นในการฝึกซ้อมการตอบสนอง

2.2 ทฤษฎีทางด้าน Hardware ของระบบวัดเวลาตอบสนอง

2.2.1 PIC (Programmable Integrated Circuit)

PIC คือ microcontroller ตระกูลหนึ่ง ซึ่งคอนเซ็ปท์ของ microcontroller ตระกูลนี้ก็คือ พยายามรวมเอาทุกอย่างเอาไว้ในตัวของมันไม่ว่าจะเป็น PROGRAM MEMROY, RAM, EEPROM, SERIAL, I2C, PWM, A/D ฯลฯ โดยไม่จำเป็นต้องต่ออุปกรณ์เสริมจากภายนอก ในตัวของ PIC จะมีฟังก์ชันที่ใช้ในการประมวลผล รวมทั้งหน่วยความจำ

MCU (microcontroller unit) ในตระกูล PIC ถ้าแบ่งออกตามชนิดของ PROGRAM MEMORY แบ่งได้เป็น 3 แบบคือ

1. OTP (one time programmable)
2. EPROM (erasable programmable ROM)
3. EEPROM / Flash (electronically erasable programmable ROM)

1. OTP เป็น chip ที่มีราคาถูกที่สุด ในสามประเภท สาเหตุก็มาจากว่า chip แบบ OTP จะสามารถทำการโปรแกรมได้แค่ครั้งเดียวเท่านั้น หลักจาก chip ได้ถูกโปรแกรมไปแล้วจะไม่สามารถโปรแกรมเข้าไปใหม่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ได้อีก ดังนั้น chip ประเภทนี้จะนิยมใช้หลังจากได้พัฒนาโปรแกรมจนกระทั่งจุดบกพร่องต่างๆ ในโปรแกรม ไม่มีอีกแล้ว เพราะจะมีต้นทุนต่ำเมื่อเทียบตัว memory ประเภทอื่น จะมีตัวอักษร C แสดงบนตัว chip เช่น 16C84, 16C74

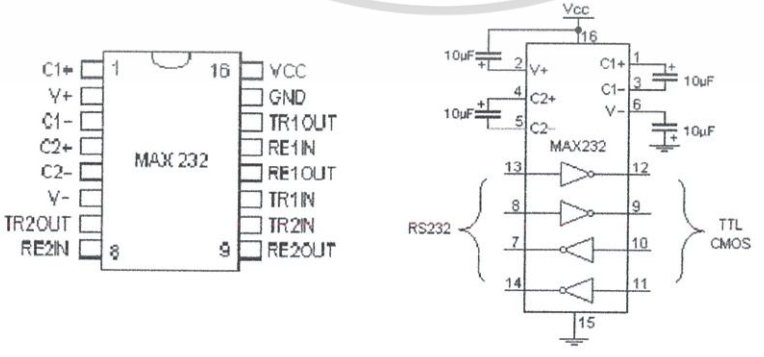
2. EPROM เป็น chip ที่มี program memory ที่เมื่อเขียนโปรแกรมเข้าไปแล้วสามารถโปรแกรมใหม่ด้วยการลบโปรแกรมเดิมโดยให้แสง UV (Ultra Violet) ส่องผ่านเข้าไปยัง chip ประมาณ 5-10 นาที ดังนั้นที่ด้านบนของ chip จะมีกรอบกระจกเพื่อให้แสง UV สามารถส่องผ่านเข้าไปในตัว chip ได้ แต่ก็มีจำนวนครั้งในการลบโปรแกรมเช่นกัน เมื่อลบโปรแกรมด้วยแสง UV มากๆ เข้าก็จะเกิดอาการด้าน คือโปรแกรมไม่เข้านั้นเอง จะมีตัวอักษร JW หรือว่าดูเอาว่ามีกรอบกระจกอยู่บน chip หรือไม่

รูปที่ 2.2 ตัวอย่างของ PIC แบบ EPROM

3. EEPROM / Flash เป็น chip ที่ออกมาไม่กี่ปีนี้เองส่วนของ program memory สามารถอ่านหรือเขียนด้วยสัญญาณทางไฟฟ้า ใช้เวลาในการลบข้อมูลไม่กี่วินาที และสามารถลบ และเขียนใหม่ได้หลายพันครั้ง ทำให้เป็นที่นิยมที่สุดใน 3 ประเภท มีตัวอักษร F เป็นตัวบอก เช่น 16F84, 16F877

2.2.2 ตัวแปลงสัญญาณ RS-232 Level Converter

MAX 232, ICL 232 เป็นไอซีที่แปลงระดับสัญญาณของ RS-232 มาเป็นระดับ TTL และใช้แปลงระดับสัญญาณ TTL ไปเป็นระดับสัญญาณ RS-232 เป็นไอซี 16 ขา ใช้ไฟ 5 V_{dc} ภายในมีวงจรแปลง RS-232 เป็น TTL สองชุด และวงจรแปลง TTL เป็น RS-232 อีกสองชุด รองรับมาตรฐาน RS-232 ตามข้อกำหนด EIA/TIA ภายในวงจร MAX232 มีวงจรทวีแรงดันจาก 5V_{dc} เป็น +10 V and -10 V และวงจรกลับขั้วแรงดัน

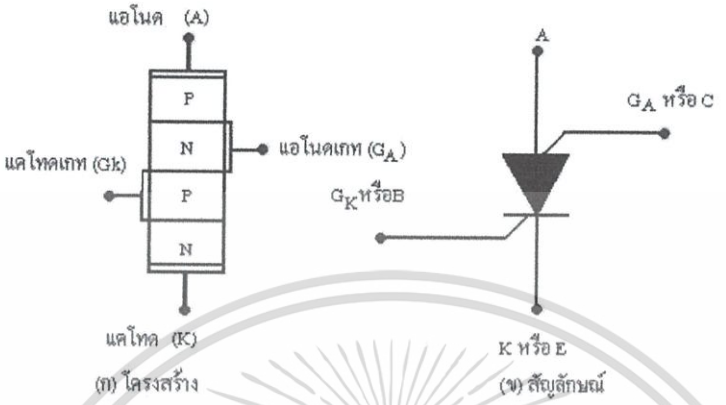


รูปที่ 2.3 IC MAX 232

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.3 Silicon Controlled Rectifier (SCR)

เป็นอุปกรณ์สารกึ่งตัวนำ 3 ขา คือ Gate, Anode, Cathode จะทำหน้าที่เป็นสวิทช์อิเล็คทรอนิกส์ กระแสไฟฟ้าเพียงเล็กน้อยที่ไหลเข้าเกทจะทำให้กระแสจำนวนมากไหลผ่านทั้งสองข้างโดยมีสถานะปิดหรือเปิดเท่านั้นไม่ขยายสัญญาณเหมือนทรานซิสเตอร์



รูปที่ 2.4 สัญลักษณ์และโครงสร้างของ SCR

2.2.4 รีเลย์ (Relay)

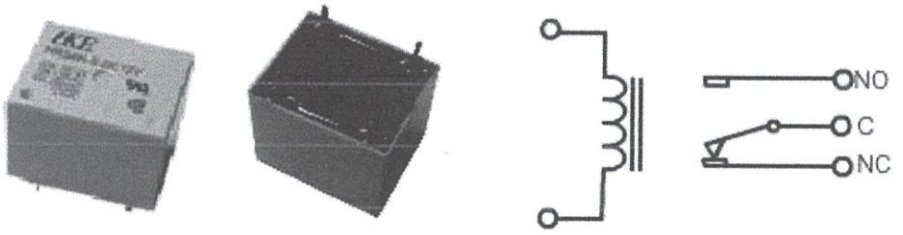
รีเลย์ คือ อุปกรณ์อิเล็กทรอนิกส์ที่ ทำหน้าที่ ตัด-ต่อวงจร คล้ายกับสวิตช์ โดยใช้หลักการหน้าสัมผัส และการที่จะให้มันทำงานก็ต้องจ่ายไฟให้มันตามที่กำหนด เพราะเมื่อจ่ายไฟให้กับตัวรีเลย์ มันจะทำให้หน้าสัมผัสติดกัน กลายเป็นวงจรปิด และตรงข้ามทันทีที่ไม่ได้จ่ายไฟให้มัน มันก็จะกลายเป็นวงจรเปิด ไฟที่เราใช้ป้อนให้กับตัวรีเลย์ก็จะเป็นไฟที่มาจาก เพาเวอร์ฯ ของเครื่องเรา ดังนั้นทันทีที่เปิดเครื่อง ก็จะทำให้รีเลย์ทำงาน

ประเภทของรีเลย์

รีเลย์ เป็นอุปกรณ์ทำหน้าที่เป็นสวิตช์มีหลักการทำงานคล้ายกับขดลวดแม่เหล็กไฟฟ้าหรือโซลินอยด์ (solenoid) รีเลย์ใช้ในการควบคุมวงจรไฟฟ้าได้อย่างหลากหลาย รีเลย์เป็นสวิตช์ควบคุมที่ทำงานด้วยไฟฟ้า แบ่งออกตามลักษณะการใช้งานได้เป็น 2 ประเภทคือ

1. รีเลย์กำลัง (Power relay) หรือมักเรียกกันว่าคอนแทกเตอร์ (Contactor or Magneticcontactor) ใช้ในการควบคุมไฟฟ้ากำลัง มีขนาดใหญ่กว่ารีเลย์ธรรมดา
2. รีเลย์ควบคุม (Control Relay) มีขนาดเล็กกำลังไฟฟ้าต่ำ ใช้ในวงจรควบคุมทั่วไปที่มีกำลังไฟฟ้าไม่มากนัก หรือเพื่อการควบคุมรีเลย์หรือคอนแทกเตอร์ขนาดใหญ่ รีเลย์ควบคุม บางทีเรียกกันง่าย ๆ ว่า "รีเลย์"

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.5 ลักษณะและสัญลักษณ์ของรีเลย์

2.3 ทฤษฎีทางด้าน Software

2.3.1 ภาษาที่ใช้สำหรับไมโครคอนโทรลเลอร์ควบคุม VX-Propeller

การพัฒนาเครื่องฝึกการหลบหมัดได้มีการปรับปรุงระบบใหม่ โดยมีการเปลี่ยนจากการใช้คอมพิวเตอร์มาเป็นไมโครคอนโทรลเลอร์ VX-Propeller ในการรับ-ส่งข้อมูล และแสดงผล แทน ดังนั้นจึงต้องมีการเขียนโปรแกรมบนไมโครคอนโทรลเลอร์ VX-Propellerขึ้นมาเพื่อใช้ในการติดต่อกับไมโครคอนโทรลเลอร์บนปุ่มกดผ่านทางพอร์ตอนุกรม โดย VX-Propeller มีภาษาเป็นของตัวเองชื่อว่า สปิน (Spin) และ ภาษาแอสเซมบลี โดยในโครงงานนี้ได้ใช้มีการใช้งานเฉพาะภาษาสปินเท่านั้น ลักษณะการเขียนโปรแกรมสปินเป็นแบบออบเจ็กต์ พัฒนาขึ้นโดย Parallax ผู้ผลิตชิป propeller

2.3.2 ภาษาที่ใช้สำหรับไมโครคอนโทรลเลอร์ PIC

ภาษาที่ใช้สำหรับการเขียนโปรแกรมบนไมโครคอนโทรลเลอร์ PIC ที่อยู่บนปุ่มกด แบ่งได้เป็น ภาษาระดับสูง และภาษาระดับต่ำ

- ภาษาระดับสูงเช่น C, Basic ข้อดี คือเขียนง่าย แก้ไขเปลี่ยนแปลง หรือเพิ่มเติมได้ง่าย ส่วน ข้อเสียก็คือการทำงานจะช้า ขนาดโปรแกรมที่เขียนมีขนาดใหญ่
- ภาษาระดับต่ำ ซึ่งก็คือ ภาษา Assembly ข้อดีคือ ตัว compiler แจกฟรี ขนาดโปรแกรมหลังจาก compiled แล้วมีขนาดเล็ก โปรแกรมมีความเร็ว แต่ข้อเสียก็คือ เขียนยาก เพราะลักษณะภาษาไม่ค่อยสื่อความหมาย แก้ไขเปลี่ยนแปลงยาก

โดยในการทำโครงงานครั้งนี้ได้เลือกใช้ภาษาซีในการเขียนโปรแกรมเพื่อควบคุมการทำงานของไมโครคอนโทรลเลอร์ เพราะเป็นภาษาที่เขียนง่ายและนิยมใช้กันอย่างแพร่หลาย

ภาษาซี

ภาษาซีได้รับการพัฒนาขึ้นโดย Dennis Ritchie ขณะที่ทำงานอยู่ที่ Bell Laboratories โดยพัฒนาขึ้นจากหลักการพื้นฐานของภาษาบี (B) และ บีซีพีแอล (BCPL) ในช่วงปี ค.ศ. 1971 ถึง 1973 แต่ได้เพิ่มชนิดข้อมูลและความสามารถอื่นๆ ให้มากขึ้น และนำภาษาซีไปใช้พัฒนาระบบปฏิบัติการยูนิกซ์ (UNIX) บนเครื่องคอมพิวเตอร์ DEC PDP-11 ภาษาซีเป็นที่นิยมใช้อย่างแพรหลายในช่วงต้นทศวรรษที่ 1980 จนกระทั่งมีความพยายามกำหนดมาตรฐานของภาษาเพื่อให้สามารถใช้ภาษาซีได้บนเครื่องคอมพิวเตอร์ใดๆ ในปี ค.ศ. 1983 โดย ANSI (The American National Standards Institute) ได้ตั้งคณะกรรมการ X3J11 เพื่อร่างมาตรฐานดังกล่าว และได้รับการตรวจสอบและยอมรับโดย ANSI และ ISO (The International Standards Organization) โดยมีการตีพิมพ์มาตรฐานของภาษาซีในปี ค.ศ. 1990 จากความมีประสิทธิภาพและสามารถทำงานบนเครื่องคอมพิวเตอร์ใดๆ ของภาษาซี จึงได้มีการนำภาษาซีไปใช้ในการพัฒนา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ระบบปฏิบัติการต่างๆ และใช้เป็นต้นแบบของภาษาอื่นๆ ที่สำคัญในปัจจุบัน เช่น ซีพลัสพลัส (C++) จาวา (Java) เป็นต้น

2.3.3 ไทเมอร์ และ เคาน์เตอร์

ไมโครคอนโทรลเลอร์ PIC เบอร์ 16F688 มีไทเมอร์ (Timer) ให้เลือกใช้งานได้ 2 ตัว คือ ไทเมอร์0 (Timer0) และ ไทเมอร์1 (Timer1) โดยเป็นได้ทั้งตัวจับเวลาหรือที่เรียกว่า ไทเมอร์ (Timer) เพื่อนับจำนวนพัลส์สัญญาณนาฬิกาภายในตัวไมโครคอนโทรลเลอร์ และตัวนับหรือที่เรียกว่า เคาน์เตอร์ (Counter) เพื่อนับสัญญาณจากภายนอก

โมดูลไทเมอร์0 (Timer0 Module) ไทเมอร์0 เป็นได้ทั้งตัวจับเวลา (Timer) และตัวนับสัญญาณ (Counter) โดยมีคุณสมบัติดังนี้

1. เป็นไทเมอร์/เคาน์เตอร์ขนาด 8 บิต
2. มีปรีสเกลเลอร์ขนาด 8 บิต (ตัวหารความถี่สัญญาณนาฬิกา) ดังนี้ 1, 2, 4, 8, 16, 32, 64, 128, 256
3. เลือกสัญญาณนาฬิกาได้ทั้งภายใน(ตัวจับเวลา) และสัญญาณนาฬิกาภายนอก(ตัวนับสัญญาณ)
4. อินเตอร์รัปต์เมื่อเกิดโอเวอร์โฟลว์จากการนับค่า 255 (FFh) ไปเป็น 0 (00h) หรือนับได้ 256 ค่า

โมดูลไทเมอร์1 (Timer1 Module) ไทเมอร์1 เป็นได้ทั้งตัวจับเวลา (Timer) และตัวนับสัญญาณ (Counter) โดยมีคุณสมบัติดังนี้

1. เป็นไทเมอร์/เคาน์เตอร์ขนาด 16 บิต โดยแบ่งเป็นรีจิสเตอร์ขนาด 8 บิต 2 ตัว คือ TMR1H และ TMR1L
2. มีปรีสเกลเลอร์ขนาด 8 บิต (ตัวหารความถี่สัญญาณนาฬิกา) ดังนี้ 1, 2, 4, 8
3. เกิดโอเวอร์โฟลว์จากการนับค่า 65535 ไปเป็น 0 หรือนับได้ 65536 ค่า
4. เป็นไทเมอร์สำหรับทำงานร่วมกับโมดูล CCP (Capture/Compare/PWM Module) ในโหมด Capture (ตรวจจับสัญญาณ) และโหมด Compare (เปรียบเทียบข้อมูล)

โมดูลไทเมอร์2 (Timer2 Module) ไทเมอร์2 เป็นได้ทั้งตัวจับเวลา (Timer) และตัวนับสัญญาณ (Counter) โดยมีคุณสมบัติดังนี้

1. เป็นไทเมอร์ 8 บิต
2. มีปรีสเกลเลอร์ (ตัวหารความถี่สัญญาณนาฬิกา) ดังนี้ 1, 4, 16 และ โพสต์สเกลเลอร์ดังนี้ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
3. อินเตอร์รัปต์เมื่อเกิดโอเวอร์โฟลว์จากการนับค่า 255 (FFh) ไปเป็น 0 (00h) หรือนับได้ 256 ค่า
4. เป็นฐานเวลาในการสร้าง PWM (Pulse-Width Modulation)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3.4 อินเตอร์รัปต์

การอินเตอร์รัปต์ (Interrupt) คือ การขัดจังหวะการทำงานของ CPU หรือโปรแกรมที่กำลังทำงานอยู่เพื่อมาทำงานในส่วนของโปรแกรมบริการอินเตอร์รัปต์ที่ผู้พัฒนาโปรแกรมได้กำหนดไว้ล่วงหน้าแล้ว ประโยชน์ของการใช้อินเตอร์รัปต์นั้นจะช่วยให้ประหยัดเวลาในการทำงานของโปรแกรมหลักที่ไม่ต้องไปคอยตรวจสอบเงื่อนไขใดเงื่อนไขหนึ่งตลอดเวลาการทำงานของโปรแกรม โดยมอบหน้าที่การตรวจสอบนี้ให้กับการบริการอินเตอร์รัปต์แทนไมโครคอนโทรลเลอร์ PIC จะมีบริการการตอบสนองการใช้งานอินเตอร์รัปต์ที่ค่อนข้างมาก และหลากหลายประเภท เช่น

- อินเตอร์รัปต์จากไทเมอร์/เคาน์เตอร์โอเวอร์โฟลว์
- อินเตอร์รัปต์จากการเปลี่ยนแปลงสัญญาณอะนาลอกเป็นดิจิตอล
- อินเตอร์รัปต์จากโมดูล CCP1 และ CCP2
- อินเตอร์รัปต์เนื่องจากการเปลี่ยนแปลงสัญญาณของพอร์ต B
- อินเตอร์รัปต์เนื่องจากการรับส่งข้อมูลอนุกรม RS-232 (USART)

การอินเตอร์รัปต์ที่กล่าวมานี้เป็นเพียงบางส่วนของการบริการอินเตอร์รัปต์ภายใน PIC เท่านั้น เนื่องจากการอินเตอร์รัปต์จะขึ้นอยู่กับเบอร์ PIC ที่เลือกใช้งาน

ตัวอย่างอินเตอร์รัปต์ของ PIC 16F688

INT_EXT	คือ	การอินเตอร์รัปต์จากสัญญาณภายนอก
INT_AD	คือ	การอินเตอร์รัปต์จากการแปลงสัญญาณอะนาลอกเป็นดิจิตอล
INT_RDA	คือ	การอินเตอร์รัปต์เมื่อมีการรับข้อมูลจาก RS232
INT_TIMER1	คือ	การอินเตอร์รัปต์จากการเกิด Overflow ของ TIMER1
INT_TIMER0	คือ	การอินเตอร์รัปต์จากการเกิด Overflow ของ TIMER0
INT_EEPROM	คือ	การอินเตอร์รัปต์จากการโปรแกรมเสร็จสมบูรณ์
INT_COMP	คือ	การอินเตอร์รัปต์จากการเปรียบเทียบสัญญาณ
INT_RA	คือ	การอินเตอร์รัปต์เมื่อพอร์ต A มีการเปลี่ยนแปลงลอจิก
INT_COMP2	คือ	การอินเตอร์รัปต์จากการเปรียบเทียบสัญญาณ
INT_OSC_FAIL	คือ	การอินเตอร์รัปต์เมื่อเกิดการผิดพลาดของ Oscillator
INT_TBE	คือ	การอินเตอร์รัปต์เมื่อบัฟเฟอร์ส่งข้อมูลของพอร์ต RS-232 ว่าง
INT_RAx	คือ	การอินเตอร์รัปต์เมื่อพอร์ต Ax มีการเปลี่ยนแปลงลอจิก

2.4 ทฤษฎีพื้นฐานการสื่อสารแบบอนุกรม

ในการสื่อสารแบบอนุกรมในเครื่องคอมพิวเตอร์นั้นจะมีความเร็วในการสื่อสารช้ากว่าแบบขนาน เป็นเพราะว่าการเคลื่อนย้ายข้อมูลแบบอนุกรมนั้นเป็นการส่งข้อมูลทีละ 1 บิต แต่พอร์ตขนานนั้นสามารถส่งข้อมูลได้ครั้งละหลายๆ บิตพร้อมกัน ดังนั้นจึงทำให้การสื่อสารข้อมูลแบบอนุกรมมีความเร็วต่ำกว่าแบบขนาน แต่ว่าข้อดีของการส่งข้อมูลแบบอนุกรมคือ สามารถส่งข้อมูลได้ระยะทางไกลกว่าแบบขนาน และอีกทั้งสายสัญญาณก็มีน้อยกว่าการส่งข้อมูลแบบขนานอีกด้วย การสื่อสารแบบอนุกรมสามารถแบ่งออกได้เป็น 3 รูปแบบ ดังนี้

1. Simplex สามารถส่งข้อมูลได้อย่างเดียว เป็นการสื่อสารแบบทางเดียว
2. Half–Duplex สามารถส่งข้อมูลไปยังปลายทางและสามารถรับข้อมูลจากปลายทางได้ แต่ไม่สามารถทำการส่งและรับข้อมูลในเวลาเดียวกันได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. Full–Duplex สามารถรับและส่งข้อมูลได้ในเวลาเดียวกัน

การสื่อสารแบบอนุกรมสามารถแบ่งประเภทของการสื่อสาร ตามลักษณะสัญญาณในการสื่อสารได้ 2 แบบ คือ

1. การสื่อสารแบบซิงโครนัส (Synchronous) สำหรับการสื่อสารแบบซิงโครนัสนี้จะใช้สัญญาณนาฬิกาควบคุมการรับส่งสัญญาณ เช่น สายคีย์บอร์ดคอมพิวเตอร์ โดยจะมีสายสัญญาณเส้นหนึ่งเป็นสายสัญญาณนาฬิกา ส่วนอีกเส้นหนึ่งเป็นสายของข้อมูล (และมักจะมีสายกราวน์ด้วย) สำหรับการสื่อสารแบบซิงโครนัสนี้ เหมาะสำหรับการทำงานในระยะใกล้ ข้อมูลที่จะส่งมีไม่มากนัก เพราะถ้าระยะทางไกลขึ้นจะทำให้สัญญาณนาฬิกามีปัญหา อีกทั้งต้องมีสายหลายเส้นทำให้สิ้นเปลืองมาก
2. การสื่อสารแบบอะซิงโครนัส (Asynchronous) สำหรับการสื่อสารแบบอะซิงโครนัสนั้น จะใช้สายสัญญาณเพียงตัวเดียว แต่จะใช้รูปแบบการส่งข้อมูล หรือ Bit Pattern เป็นตัวกำหนดว่าส่วนไหนเป็นตัวเริ่มต้นข้อมูล ส่วนไหนเป็นตัวข้อมูล ส่วนไหนจะเป็นตัวตรวจสอบความถูกต้องของข้อมูล และส่วนไหนเป็นส่วนปิดท้ายของข้อมูล โดยต้องกำหนดให้สัญญาณนาฬิกาเท่ากันทั้งภาครับและภาคส่ง ซึ่งจะมีอุปกรณ์พิเศษที่เรียกว่า UART หรือ Universal Asynchronous Reciever/Trasmitter คอยควบคุมการรับและการส่งข้อมูล

2.4.1 UART (Universal Asynchronous Receiver Transmitter)

Universal Asynchronous Receiver Transmitter ซึ่งหมายถึงอุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัสนั่นเอง หน้าที่หลักของ UART คือ ทำหน้าที่แปลงข้อมูลที่อยู่ในรูปแบบขนานจากคอมพิวเตอร์ให้อยู่ในรูปแบบอนุกรมแบบอะซิงโครนัสแล้วส่งออกไป และทำหน้าที่แปลงสัญญาณอนุกรมแบบอะซิงโครนัสที่ป้อนเข้ามายัง UART ให้เป็นแบบขนานก่อนที่จะส่งเข้าคอมพิวเตอร์ ซึ่งนอกจาก UART จะส่งข้อมูลไปยังคอมพิวเตอร์แล้ว ยังทำการแจ้งข้อมูลอื่นๆ ให้คอมพิวเตอร์ทราบด้วย เช่น อัตราความเร็วในการรับส่งข้อมูล (บอดเรต), รูปแบบการส่งข้อมูล, ความผิดพลาดที่เกิดขึ้นระหว่างการถ่ายทอดข้อมูล (ผิดพลาดจากพาริตี้, เฟรมข้อมูล, โอเวอร์รัน) เป็นต้น ภายใน UART จะมีส่วนของวงจรสร้างอัตราการถ่ายทอดข้อมูลแบบโปรแกรมได้ (Programmable Buadrate Generator) โดยการกำหนดค่าตัวหารให้กับสัญญาณนาฬิกาของ UART โดยตัวหารนี้มีขนาด 16 บิตดังนั้นจึงกำหนดตัวหารให้อยู่ในช่วง 10 – 65, 535 UART สามารถรับส่งข้อมูลได้ทั้งแบบฮาล์ฟดูเพล็กซ์ (Half Duplex) และฟูลดูเพล็ก (Full Doplex)

2.4.2 มาตรฐาน RS-232

เป็นมาตรฐานการเชื่อมต่อข้อมูลแบบอนุกรมที่มีคนนิยมใช้มากที่สุด กำหนดโดย EIA (Electronics Industry Association) หรือสมาคมผู้ประกอบการอุตสาหกรรมอิเล็กทรอนิกส์ของอเมริกา ตั้งแต่ปี 1969 ใช้กับการสื่อสารแบบจุดต่อจุด โดยใช้สายเชื่อมต่อ DB แบบ 25 และ 9 เข็ม มีการทำงานแบบสองทางพร้อมกัน (Full-duplex) โดยอาจใช้สายสัญญาณอื่นร่วมด้วยเพื่อทำแฮนด์เชค (Hand-shake) หรือไม่ก็ได้ มาตรฐาน RS-232 จำกัดความยาวสายไว้ที่ 50 ฟุต (หรือประมาณ 15 เมตร) สำหรับการส่งสัญญาณที่ความเร็ว 19,200 บิทต่อวินาที โดยที่ความยาวสายจะต้องสั้นลงถ้าต้องการสื่อสารที่ความเร็วสูงขึ้น ปัญหาหลักของ RS-232 คือไม่ทนต่อ Noise เนื่องจากข้อมูลในสาย TX และ RX ต้องเปรียบเทียบระดับสัญญาณกับ GND เมื่อ GND ถูกรบกวนทำให้ GND เปลี่ยนไปจากเดิม ซึ่งจะมีผลต่อสัญญาณแน่นอน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4.3 มาตรฐาน RS-422 หรือ 422-A

ถูกกำหนดขึ้นโดยสมาคมผู้ประกอบการอุตสาหกรรมอิเล็กทรอนิกส์หรือ EIA เช่นเดียวกันกับมาตรฐาน RS-232 โดยมีจุดมุ่งหมายที่จะแก้ปัญหาเรื่องความยาวของสายสื่อสารโดยใช้การส่งสัญญาณแบบผลต่าง (Differential) แทนที่จะใช้การส่งสัญญาณแบบอ้างอิงกับจุดกราวนด์ เช่นเดียวกันกับ RS-232 การส่งสัญญาณแบบ Differential นี้ช่วยลดปัญหาสัญญาณรบกวนจาก 2 ปัจจัยด้วยกัน ได้แก่ ปัญหาแรงดันกราวนด์ 2 ฝั่งสายไม่เท่ากัน อันเกิดจากกระแสไฟฟ้าที่ไหลในสายกราวนด์ที่ยาวมากๆ ก่อให้เกิดความต่างศักดิ์ และปัญหาสัญญาณรบกวนที่เกิดจากแม่เหล็กไฟฟ้าเหนี่ยวนำในสาย โดยหากสายไฟที่ใช้ถูกตีเกลียวและวางไว้ใกล้กัน เมื่อมีแรงดันเหนี่ยวนำจะปรากฏแรงดันรบกวนบนสายทั้งสองเท่าๆ กันเป็นผลให้ ตัวรับที่อ่านความต่างศักดิ์ระหว่างสายอ่านข้อมูลได้เช่นเดิม ทั้งสองปัจจัยนี้เองเป็นสาเหตุที่ทำให้ความต้านทานต่อสัญญาณรบกวนของการสื่อ สารแบบ RS-232 ด้อยกว่า RS-422) ตามมาตรฐาน RS-422 นี้จะใช้สายสัญญาณทั้งหมด 4 เส้น (2 เส้นสำหรับการส่งสัญญาณ และอีก 2 เส้นสำหรับรับสัญญาณ) และสามารถใช้ความยาวสายสัญญาณได้ถึง 4,000 ฟุต (หรือ 1.2 กม.) ที่ความเร็ว 100,000 บิทต่อวินาที และการสื่อสารเป็นแบบ 2 ทางพร้อมกัน (Full Duplex)

2.4.4 มาตรฐาน RS-485

กำหนดโดยสมาคมผู้ประกอบการอุตสาหกรรมอิเล็กทรอนิกส์หรือ EIA เป็นมาตรฐานการเชื่อมต่อสัญญาณแบบอนุกรม (Serial Communication) มีลักษณะการเชื่อมต่อเป็นแบบหลายจุด (Multi-point) หรือ Multi-drop สายสัญญาณที่ใช้มีทั้งแบบที่เป็น 2 สายและแบบที่เป็น 4 สาย การต่อแบบหลายจุดนี้ทำให้สามารถมองสายสัญญาณเป็นบัสนำสัญญาณได้ (Signal Bus) จำนวนคอมพิวเตอร์หรืออุปกรณ์ที่สามารถอยู่บน RS-485 บัสหนึ่งถูกกำหนดไว้ที่ 32 ตัว ในกรณีที่ต้องการเพิ่มจะต้องมีตัวทวนสัญญาณ (Signal Repeater) หรือใช้ตัวส่ง-รับสัญญาณที่มีอิมพิแดนซ์ (ความต้านทานเสมือน) สูงขึ้น ซึ่งเราอาจเพิ่มจำนวนจุดเชื่อมต่อขึ้นได้ถึง 128 จุด ความยาวของสายสัญญาณตามมาตรฐาน RS-485 นี้สามารถยาวได้ถึง 1.2 กม. เช่นเดียวกับมาตรฐาน RS-422 แต่การสื่อสารจะเป็นแบบสองทางไม่พร้อมกัน (Half Duplex) มีเพียงคอมพิวเตอร์หรืออุปกรณ์ตัวเดียวเท่านั้นที่สามารถส่งสัญญาณออกได้ ณ เวลาหนึ่งๆ ส่วนที่เหลือจะเป็นผู้รับสัญญาณ นอกจากนี้ RS-485 ยังทนต่อสัญญาณรบกวนได้ดีกว่า RS-232 เพราะไม่ได้ใช้การอ้างอิงสัญญาณกับ GND แต่ใช้ความแตกต่างระหว่างสาย 2 สาย (A และ B) เป็นตัวบอกว่า Logic '1' หรือ Logic '0' วิธีนี้จะป้องกัน GND loop ที่เกิดขึ้น

2.4.5 องค์ประกอบของการรับส่งข้อมูลแบบอนุกรม

1. Start Bit (ขนาด 1 บิต) จะใส่ที่จุดเริ่มต้นเสมอเพื่อเตือนอุปกรณ์ฝ่ายรับว่าข้อมูลกำลังจะมาถึง
2. Data Character (ขนาด 7 บิต หรือ 8 บิต) การส่งบิต ข้อมูลจะส่งเป็นกลุ่มๆโดยทั่วไปจะส่งเป็น 7 บิต หรือ 8 บิต ซึ่งเพียงพอสำหรับการส่ง Ascii Word
3. Parity Bit (ขนาด 1 บิต) ใช้สำหรับตรวจสอบความถูกต้องของข้อมูลที่ส่งเราจะใส่บิตพาริตี้เข้าไป บิตพาริตี้มีหลายแบบดังนี้

    - พาริตี้คู่ (Even Parity) ค่าของบิตพาริตี้นี้เมื่อรวมกันทุกๆ บิตของข้อมูลแล้วจะต้องมีจำนวนบิตที่เป็นเลข 1 เป็นเลขคู่ ตัวอย่างเช่น ข้อมูล 1000111 มีเลข 1 ทั้งหมด 3 ตัว ดังนั้นบิตพาริตี้จะเป็น 0
    - พาริตี้คี่ (Odd Parity) ค่าของบิตพาริตี้นี้เมื่อรวมกันทุกๆบิตของข้อมูลแล้ว จะต้องมีจำนวน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บิตที่ เป็นเลข 1 เป็นเลขคี่ ตัวอย่างเช่น ข้อมูล 1000101 มีเลข 1 ทั้งหมด 3 ตัว ดังนั้นบิตพาริตี้จะเป็น 1

- ไม่มีพาริตี้ (None) ถ้าตั้งค่าบิตพาริตี้เป็น None ทั้งภาครับและภาคส่งจะไม่มีการตรวจสอบบิตพาริตี้

4. Stop Bit (ขนาด 1 บิต หรือ 2 บิต) เป็นบิตที่ส่งมาปิดท้ายข้อมูล

2.5 VX-Propeller

ในหัวข้อนี้จะขอกล่าวถึงไมโครคอนโทรลเลอร์ Propeller หรือ โพรเพลเลอร์ ซึ่งเป็นชื่อของไมโครคอนโทรลเลอร์ 32 บิต อนุกรม ที่มีสถาปัตยกรรมพิเศษคือ มีซีพียูภายใน 8 ตัวหรือ 8 ค็อก (cog) ที่สามารถทำงานแยกจากกันอย่างอิสระหรือร่วมกันทำงานก็ได้

2.5.1 คุณสมบัติเด่นของโพรเพลเลอร์

- ประกอบด้วย 8 ซีพียู หรือเรียกว่า 8 ค็อก ที่สามารถทำงานได้พร้อมกันอย่างเป็นอิสระ โดยมีการควบคุมการใช้ทรัพยากรร่วมกันผ่านทางตัวเชื่อมโยงกลางหรือ central hub

- มีความเร็วในการทำงานสูง และด้วยการทำงานที่เป็นอิสระของแต่ละค็อก ทำให้สามารถรองรับการตอบสนองต่อเหตุการณ์ต่างๆ ที่เกิดขึ้นในระบบได้เร็วเพียงพอ จึงไม่ต้องใช้กระบวนการอินเตอร์รัปต์ช่วย ทำให้การเขียนโปรแกรมเพื่อรองรับการทำงานในแต่ละเหตุการณ์ลดความซับซ้อนลงได้อย่างมาก

- มีการใช้สัญญาณนาฬิกาของระบบร่วมกัน ทำให้สามารถอ้างอิงค่าเวลาหลักเดียวกันได้ทำให้การทำงานของแต่ละค็อกมีจังหวะที่สอดคล้องกัน

- แต่ละค็อกจะประกอบด้วยตัวประมวลผลหรือโปรเซสเซอร์ที่มีการทำงานอย่างเป็นอิสระ มีแรม 2 กิโลไบต์ ที่เมื่อกำหนดให้ทำงานเป็นรีจิสเตอร์ 32 บิต จะได้ทั้งสิ้นถึง 512 ตัว มีโมดูลตัวนับความสามารถสูงที่ทำงานร่วมกับเฟสล็อกลูป ทำให้แต่ละค็อกทำงานได้เร็วถึง 80 MHz มีวงจรกำเนิดสัญญาณภาพและส่วนควบคุมพอร์ตอินพุตเอาท์พุตที่เป็นอิสระ

- สัญญาณนาฬิกาของระบบมาได้จาก 3 แหล่งคือ วงจรออสซิลเลเตอร์ RC ภายในเลือกได้ระหว่าง 12 หรือ 20 MHz จากแหล่งกำเนิดสัญญาณนาฬิกาภายนอก และจากการทวีคูณความถี่ของคริสตอลด้วยวงจรเฟสล็อกลูป โดยปกติแล้วจะเลือกใช้คริสตอล 5 MHz แล้วเลือก PLLx16 ทำให้ได้สัญญาณนาฬิกาของระบบที่มีความถี่ 80 MHz ในขณะที่ส่วนเชื่อมโยงกลางจะทำงานด้วยความถี่ที่ลดลงครึ่งหนึ่งของสัญญาณนาฬิกาหลัก

- มีขาพอร์ตอินพุตเอาต์พุตรวม 32 ขา โดยกำหนดให้ใช้ 2 ขาสำหรับต่อกับหน่วยความจำ อีอีพรอมสำหรับเก็บโปรแกรมของผู้ใช้งาน และอีก 2 ขา สำหรับการดาวน์โหลดโปรแกรม สามารถขับกระแสซิงก์และซอร์สสูงสุด 40 mA ต่อขา

- โพรเพลเลอร์ใช้หน่วยความจำอีอีพรอมภายนอกในการเก็บโปรแกรมของผู้ใช้งาน ทำให้อายุการใช้งานของตัวชิปจึงไม่ขึ้นกับจำนวนรอบของการลบและโปรแกรมใหม่ของหน่วยความจำโปรแกรม

- การดาวน์โหลดโปรแกรมทำได้ง่ายมากเพียงต่อเข้ากับวงจรเชื่อมต่อพอร์ตอนุกรม อาทิ วงจรของไอซี MAX232 หรือต่อผ่านชิปแปลงสัญญาณพอร์ต USB เป็นพอร์ตอนุกรมอย่าง FT232RL ไม่ต้องการเครื่องโปรแกรมใดๆเพิ่มเติม

- ด้วยความเร็วในการทำงานที่สูง และมีส่วนกำเนิดสัญญาณภาพที่มากถึง 8 ชุด ทำให้เหมาะสมอย่างมากในการนำโพรเพลเลอร์ไปใช้ในการกำเนิดสัญญาณภาพไม่ว่าจะแสดงผลด้วยจอโทรทัศน์ด้วยสัญญาณวีดีโอหรือแสดงผลด้วยจอ VGA ด้วยสัญญาณแม่สีแสง นั่นคือพื้นฐานหลักในการสร้างเครื่องเล่นวีดีโอเกม และการสร้างระบบนำเสนอ (presentation) ภาพกราฟฟิกด้วยไมโครคอนโทรลเลอร์เพียงตัวเดียว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- สามารถเชื่อมต่อกับคีย์บอร์ดและเมาส์ได้ และเมื่อรวมกับความสามารถการสร้างสัญญาณภาพได้ จึงสามารถนำโพรเพลเลอร์ไปสร้างเครื่องคอมพิวเตอร์ขนาดเล็กแบบใช้จอโทรทัศน์เป็นตัวแสดงผลได้

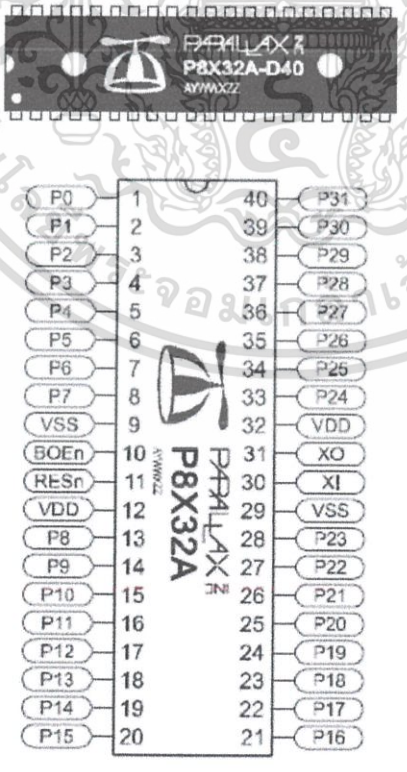
- ใช้ไฟเลี้ยงในย่าน 2.7 ถึง 3.6 V กระแสไฟฟ้าสูงสุดเมื่อขับโหลดเต็มที่คือ 300 mA

- มีตัวถังให้เลือกใช้ 3 แบบคือ DIP 40 ขา , LQFP 44 ขา และ QFN 44 ขา

2.5.2 คุณสมบัติทางเทคนิคของโพรเพลเลอร์

- เป็นไมโครคอนโทรลเลอร์ที่ภายในประกอบไปด้วยโปรเซสเซอร์ขนาด 32 บิตถึง 8 ชุด
- ทำงานที่แรงดัน 3.3 โวลต์ (2.7 V ถึง 3.6 V)
- ขาพอร์ตสามารถจ่ายกระแสซิงก์และซอร์สได้ 40 mA ต่อขา และ 100 mA ต่อพอร์ต (8 ขา)
- มีวงจรกำเนิดสัญญาณนาฬิกาภายใน 12 MHz หรือ 20 kHz เลือกกำหนดค่าได้
- ทำงานด้วยสัญญาณนาฬิกาจากภายนอกความถี่ตั้งแต่ดีซีถึง 80 MHz
- สามารถใช้คริสตอล 4 MHz ถึง 8 MHz ร่วมกับตัวคูณสัญญาณนาฬิกาความถี่สูงสุด 80 MHz
- สามารถเชื่อมต่อคริสตอลภายนอกได้ โดยไม่จำเป็นต้องเชื่อมต่อตัวเก็บประจุ
- หน่วยความจำของทั้งระบบ แบ่งเป็นหน่วยความจำอีอีพรอม 32 กิโลไบต์ (KB) และหน่วยความจำแรม 32 KB
- ในแต่ละโปรเซสเซอร์มีหน่วยความจำแรม ตัวละ 2 KB
- การจัดการหน่วยความจำเป็นแบบ 32 บิต
- จำนวนพอร์ตอินพุตเอาต์พุต 32 ขา

รูปร่างและตำแหน่งขาของโพรเพลเลอร์แสดงในรูปที่ 2.5 สำหรับตารางที่ 2.1 แสดงรายละเอียดหน้าที่การทำงานของขาต่างๆ ของโพรเพลเลอร์ เบอร์ P8X32A



รูปที่ 2.6 แสดงตำแหน่งขาและรูปร่างของไมโครคอนโทรลเลอร์โพรเพลเลอร์แบบตัวถัง DIP 40 ขา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตำแหน่งขา	หน้าที่
P0 – P31	ขาพอร์ตอินพุตเอาต์พุตเอนกประสงค์ พอร์ต A สามารถจ่ายกระแส ซอร์ส / ซิงก์ ได้ 40 mA ที่แรงดัน 3.3 VDC ระดับลอจิกมีจุดตัดที่ ½ VDD หรือ 1.6 VDC ที่แรงดันไฟเลี้ยง 3.3 V ขาพอร์ต บางขายังมีหน้าที่พิเศษ เมื่อจ่ายไฟครั้งแรกหรือรีเซต (แต่สามารถสั่งให้ขาอินพุตเอาต์พุตได้ผ่านซอฟต์แวร์) ได้แก่ - P28 เป็นขา SCL ของ I2C สำหรับการเชื่อมต่ออีอีพรอมภายนอก - P29 เป็นขา SDA ของ I2C สำหรับการเชื่อมต่ออีอีพรอมภายนอก - P30 เป็นขา Tx ส่งข้อมูล สำหรับการสื่อสารอนุกรมกับคอมพิวเตอร์และดาวน์โหลดโปรแกรม - P31 เป็นขา Rx รับข้อมูล สำหรับการสื่อสารอนุกรมกับคอมพิวเตอร์และดาวน์โหลดโปรแกรม
VDD	ขาไฟบวก 3.3 V (2.7 – 3.6 VDC)
VSS	ขากราวด์
BOE	ขาเอ็นเอเบิล Brown out (รีเซตเมื่อแรงดันต่ำกว่าที่กำหนด) ทำงานที่ลอจิก “0” ถ้าขานี้เป็น “0” ขารีเซตจะทำหน้าที่เป็นขาเอาต์พุต เพื่อแสดงสถานะ แต่ยังสามารถส่งลอจิก “0” ให้ขารีเซตเพื่อรีเซตไมโครคอนโทรลเลอร์ได้ ถ้าให้ขานี้เป็น “1” ขานี้จะทำหน้าที่เป็นขาอินพุตแบบชมิตทริกเกอร์
RES	ขารีเซต ทำงานที่ลอจิก “0” เมื่อขานี้เป็น “0” Propeller จะถูกรีเซต Cog ทั้งหมดจะถูกดิสเอเบิ้ล ขาพอร์ตอินพุตเอาต์พุตจะอยู่ในสถานะลอย Propeller จะรีเซตระยะเวลา 50 mS เมื่อขารีเซตเปลี่ยนสถานะจาก “0” เป็น “1”
XI	ขาอินพุตของคริสตอล ใช้ต่อกับแหล่งกำเนิดสัญญาณนาฬิกาจากภายนอก (ขา XO ไม่ใช้งาน) หรือต่อกับขาด้านหนึ่งของคริสตอลหรือเรโซเนเตอร์ (ขาอีกข้างต่อกับ XO) โดยไม่จำเป็นต้องต่อตัวต้านทานหรือตัวเก็บประจุภายนอก
XO	ขาเอาต์พุตของคริสตอล ออกแบบมาเป็นขาป้อนกลับของคริสตอล การต่อคริสตอลเข้ากับขา XI และ XO จะต้องสัมพันธ์กับค่าที่กำหนดให้กับรีจิสเตอร์ CLK ด้วย

ตารางที่ 2.1 แสดงรายละเอียดหน้าที่การทำงานของขาต่างๆของโพรเพลเลอร์

2.5.3 หลักการทำงานของโพรเพลเลอร์

รูปที่ 2.6 แสดงให้เห็นบล็อกไดอะแกรมการทำงานภายในของโพรเพลเลอร์ ซึ่งประกอบด้วยโปรเซสเซอร์ที่ทำงานแยกกันอิสระถึง 8 ชุด โดยจะเรียกโปรเซสเซอร์เหล่านี้ว่า Cog หมายเลข 0 ถึง 7

1. ค็อก (Cogs)

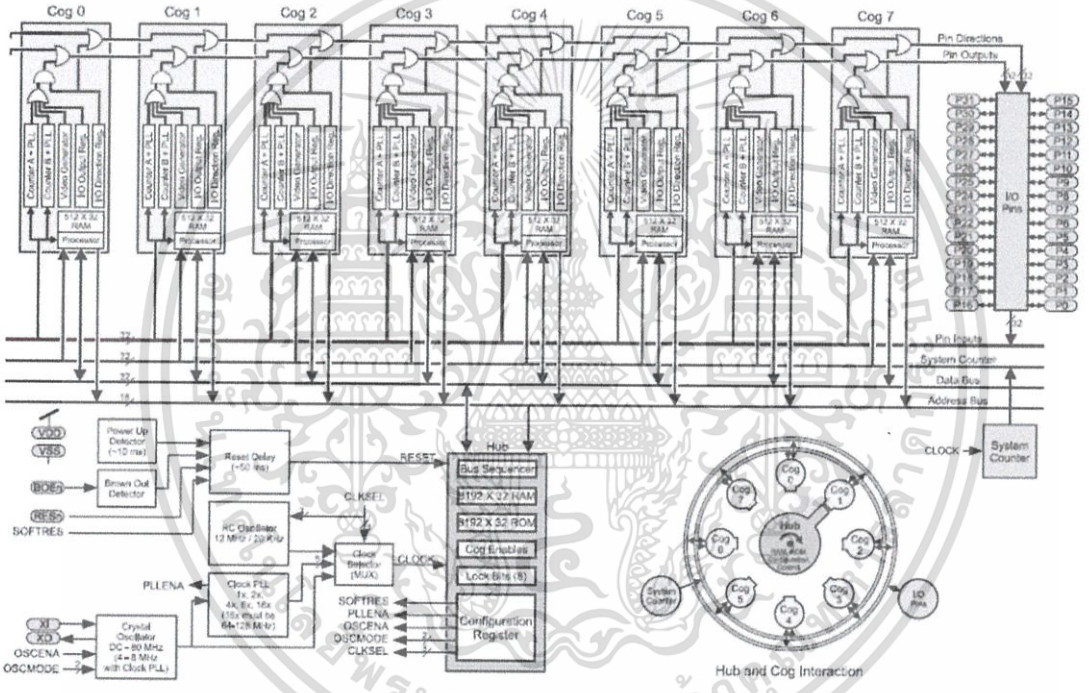
ในแต่เฟสล็อกลูป 2 ตัว โมดูลสร้างสัญญาณวีดีโอ รีจิสเตอร์พอร์ตอินพุตเอาต์พุต รีจิสเตอร์กำหนดทิศทางของพอร์ตอินพุตเอาต์พุต และรีจิสเตอร์ตัวอื่นๆ ซึ่งไม่ได้แสดงให้เห็นในบล็อกไดอะแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ค็อกทั้ง 8 ตัวทำงานด้วยวงจรกำเนิดสัญญาณนาฬิกาหลัก ซึ่งค็อกแต่ละตัวจะอ้างอิงการทำงานกันได้ด้วยสัญญาณนาฬิกาและจะเริ่มต้นทำงานพร้อมกันและใช้ทรัพยากรด้วยกัน ค็อกแต่ละตัวสามารถสั่งให้ทำงานหรือหยุดทำงานได้ในขั้นตอนการทำงานของโปรแกรม และสามารถควบคุมให้แต่ละค็อกทำงานไปพร้อมๆกันได้ โดยจะทำงานเป็นอิสระหรือ เชื่อมโยงถึงกันได้ผ่านหน่วยความจำแรมหลัก (Main RAM) ซึ่งแยกไปต่างหาก

หน่วยความจำภายในค็อกแต่ละตัว เรียกว่า ค็อกแรม (Cog RAM) โดยค็อกแรมจะแบ่งหน่วยความจำ เป็นรีจิสเตอร์ขนาด 32 บิต จำนวน 512 ตัวสามารถใช้งานได้อย่างอิสระ ยกเว้นรีจิสเตอร์ 16 ตำแหน่งสุดท้าย ซึ่งสงวนไว้สำหรับรีจิสเตอร์ฟังก์ชั่นพิเศษ เช่น รีจิสเตอร์เคาน์เตอร์ รีจิสเตอร์พอร์ตอินพุตเอาต์พุต เป็นต้น

ละค็อกจะประกอบไปด้วยหน่วยความจำแรม 2 KB โดยกำหนดเป็นหน่วยความจำแบบ 32 บิต จำนวน 512 ตัว นอกจากนี้ภายในโปรเซสเซอร์แต่ละตัวยังมีโมดูลเคาน์เตอร์แบบพิเศษพร้อม



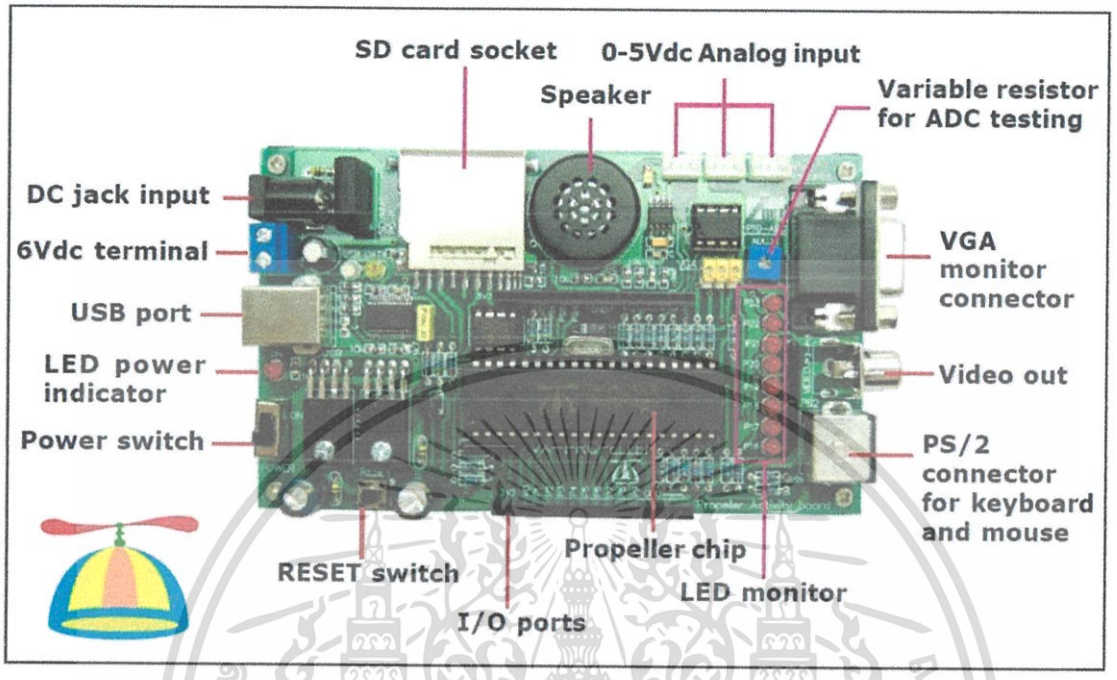
รูปที่ 2.7 บล็อกไดอะแกรมแสดงการทำงานภายในของโปรเซสเซอร์ทั้ง 8 ตัวของโพรเพลเลอร์

2. ฮับ (Hub) : ส่วนเชื่อมโยงหลัก

ฮับทำหน้าที่จัดระเบียบการทำงานของระบบทั้งหมด โดยจะยอมให้ค็อกทีละตัวเท่านั้นที่จะติดต่อกับทรัพยากรหลักของระบบ โดยจะหมุนเวียนติดต่อกับค็อกตั้งแต่หมายเลข 0 ถึง 7 แล้วกลับไปที่หมายเลข 0 ใหม่เป็นลักษณะวนรอบ ส่วนของฮับและระบบบัสของมันทำงานด้วยความเร็วครึ่งหนึ่งของสัญญาณนาฬิกาของทั้งระบบ ทำให้ค็อก 1 ตัว จะถูกติดต่อทุกๆ 16 ไซเคิลของสัญญาณนาฬิกา และใช้เวลา 7 ไซเคิลเพื่อเอ็กซิคิวซ์คำสั่ง ดังนั้น ฮับจะติดต่อกับค็อกตัวใดตัวหนึ่งได้ อาจใช้คาบเวลาเพียง 7 ไซเคิล หรือนานถึง 22 ไซเคิล เนื่องจากจะต้องรอให้ฮับวนมาจนครบรอบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. VX-Propeller บอร์ดไมโครคอนโทรลเลอร์โพรเพลเลอร์



รูปที่ 2.8 บอร์ด VX-Propeller และรายละเอียดตำแหน่งอุปกรณ์ต่างๆ ที่ติดตั้งบนบอร์ด

1.) คุณสมบัติทางเทคนิค

- ใช้ชิปโพรเพลเลอร์เบอร์ P8X32-D40 พร้อมคริสตอล 5 MHz สามารถรันด้วยความถี่สัญญาณนาฬิกาสูงสุด 80 MHz ด้วยวงจรเฟสล็อกลูป (PLLx16) ภายในตัวชิป มีความเร็วในการประมวลผล 160 MIPS จาก 8 ค็อก แต่ละค็อกมีความเร็ว 20 MIPS
- หน่วยความจำอีอีพรอม 32 กิโลไบต์ จากไอซีอีอีพรอมภายนอกเบอร์ 24LC256
- หน่วยความจำแรมภายใน 32 กิโลไบต์
- มีจุดต่อพอร์ตสำหรับติดต่ออุปกรณ์ภายนอก 14 ขา
- ดาวน์โหลดข้อมูลและโปรแกรมผ่านทางพอร์ต USB
- ต้องการแรงดันไฟตรงในย่าน +6 ถึง +12 V บนบอร์ดมีวงจรควบคุมไฟเลี้ยงคงที่ที่ +5V และ +3.3V
- LED 8 ช่อง ทำงานด้วยลอจิก “1” (ต่อร่วมกับวงจรเชื่อมต่อจอภาพ VGA)
- มีวงจรเชื่อมต่อจอภาพ VGA (ต่อร่วมกับ LED)
- มีวงจรเชื่อมต่อจอภาพด้วยสัญญาณวีดีโอ รองรับทั้งระบบ PAL และ NTSC
- มีลำโพง 8 โอห์ม 100 mW สำหรับขับเสียง
- มีวงจรเชื่อมต่อผ่านพอร์ต PS2 สามารถเชื่อมต่อกับคีย์บอร์ดและเมาส์ได้
- มีวงจรเชื่อมต่อซ็อกเก็ตของ SD CARD

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- เชื่อมต่อไอซีแปลงสัญญาณอนาลอกเป็นดิจิตอล 10 บิต 4 ช่อง แบ่งเป็นจุดต่ออินพุตสัญญาณอนาลอก 3 ช่อง และมีตัวต้านทานปรับค่าได้บนบอร์ดเพื่อทดสอบสัญญาณอนาลอกอีก 1 ช่อง

2.) การทำงานของวงจร

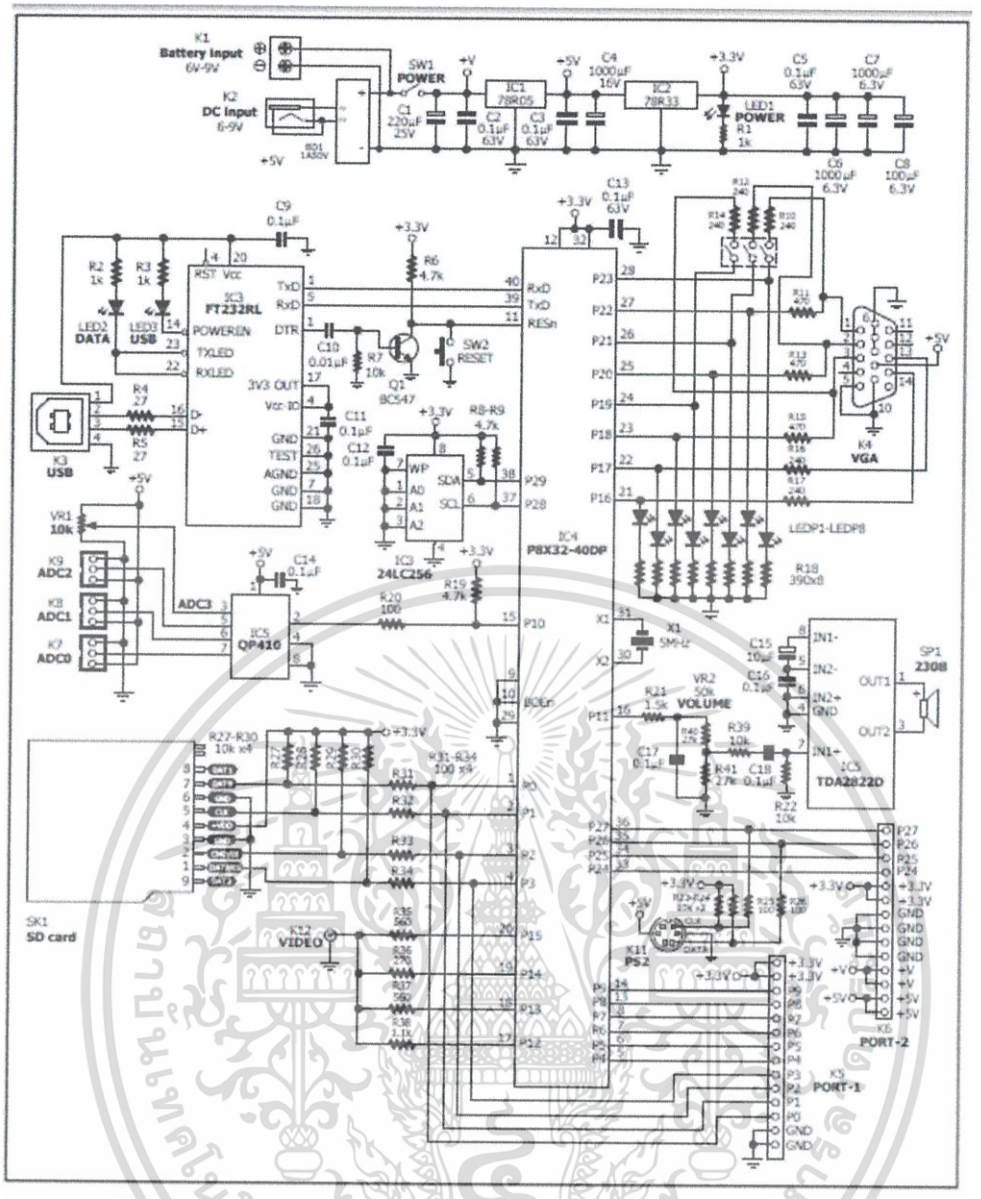
บอร์ด VX-Propeller เป็นบอร์ดที่ดึงเอาความสามารถของไมโครคอนโทรลเลอร์ Propeller มาใช้อย่างเต็มที่ โดยแสดงรายละเอียดการเชื่อมต่อวงจรทั้งหมดดังรูปที่ 2.8สามารถอธิบายการทำงานในส่วนต่างๆได้ดังนี้

ภาคจ่ายไฟจะรับแรงดันอินพุตจากแหล่งจ่ายไฟตรงสองแหล่งด้วยกันคือ จากจุดต่อแจ็กอะแดปเตอร์และจุดต่อเทอร์มินอลบล็อก สำหรับจุดต่อแจ็กอะแดปเตอร์จะมีไดโอดบริดจ์ต่อไว้เพื่อป้องกันการกลับขั้วของวงจร ก่อนส่งให้กับไอซี 78R05 เพื่อเรกูเลตแรงดันให้เหลือ 5 โวลต์ใช้เลี้ยงวงจร ในส่วนแรงดัน 5 โวลต์เช่นจุดต่อ VGA และจุดต่อคีย์บอร์ด จากนั้นส่งเข้าไปยังเรกูเลเตอร์เบอร์ 78R33 เพื่อเรกูเลตให้เหลือ 3.3 โวลต์ เป็นไฟเลี้ยงให้กับไมโครคอนโทรลเลอร์และอุปกรณ์ต่อพ่วงอื่นๆ

ภาคสื่อสารข้อมูลกับคอมพิวเตอร์และดาวน์โหลดโปรแกรม เชื่อมต่อกับคอมพิวเตอร์ผ่านพอร์ต USB โดยมีไอซี FT232RL ทำหน้าที่แปลงรูปแบบการสื่อสารข้อมูลจาก USB ให้เป็นการสื่อสารแบบอนุกรม ซึ่ง Propeller ต้องการสายสัญญาณในการเชื่อมต่อ 3 สาย ประกอบด้วย Tx , Rx และ DTR โดยขาสัญญาณ DTR จะต่อเข้ากับขา Reset ของ Propeller เพื่อรีเซตตัวมันและเข้าสู่โหมดโปรแกรม แต่เนื่องจากจะต้องป้อนสัญญาณรีเซตด้วยลอจิก “0” จึงจำเป็นต้องต่อทรานซิสเตอร์ Q1 ไว้เพื่อกลับสถานะลอจิกและจะรีเซตเมื่อขา DTR มีลอจิก “1” ไอซี FT232RL เมื่อติดต่อกับคอมพิวเตอร์เป็นที่เรียบร้อยจะแสดงสถานะที่ LED3 โดย LED3 (สีน้ำเงิน) จะติดสว่าง และเมื่อมีการส่งข้อมูล LED4 (สีเหลือง) ก็จะติดสว่าง

ส่วนหัวใจหลักของวงจรเป็นชิปโพรเพลเลอร์ทำงานด้วยคริสตอลภายนอก 5 MHz ซึ่งสามารถใช้วงจรคูณสัญญาณภายในให้ได้ความถี่สูงถึง 80 MHz ตัวมันจะมีขาอินพุตเอาต์พุตทั้งหมด 32 ขา ถูกใช้งานเพื่อสื่อสารข้อมูลกับคอมพิวเตอร์ 2 ขา คือขา Rx และ Tx ใช้เชื่อมต่อกับหน่วยความจำอีอีพรอมเพื่อเก็บข้อมูลโปรแกรมอีก 2 ขา นอกนั้นใช้งานได้เอนกประสงค์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.9 แสดงวงจรสมบูรณ์ของบอร์ด VX-Propeller (version 1.5)

บอร์ด VX-Propeller ได้จัดการเชื่อมต่อขาพอร์ตกับอุปกรณ์ภายนอกไว้ดังนี้

P0 – P3	ต่อกับซ็อกเก็ต SD CARD
P10	ต่อกับไอซีแปลงสัญญาณอนาลอกเป็นดิจิตอลเบอร์ QP410
P11	ต่อกับลำโพง
P12-P15	ต่อกับวงจรจัดสัญญาณวีดีโอ
P16-P23	ต่อกับวงจรจัดสัญญาณเพื่อเชื่อมต่อกับมอนิเตอร์ VGA และต่อพ่วงเข้ากับ LED จำนวน 8 ดวง โดยมีสวิตซ์จั๊มเปอร์เลือกการทำงาน
P26-P27	ต่อกับจุดต่อ PS2 สำหรับต่อคีย์บอร์ดภายนอก

สำหรับขาที่เหลือต่อไปยังคอนเน็กเตอร์ PORT-1 และ PORT-2 ซึ่งออกแบบตำแหน่งให้สามารถสร้างบอร์ดเสริมในภายหลังได้โดยง่าย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6 ระบบการตรวจสอบตนเอง (Self-testing)

ระบบการตรวจสอบตนเอง คือ โปรแกรมที่ถูกฝังไว้ภายใน สำหรับการตรวจสอบความพร้อมโดยรวมของระบบ รวมไปถึงอุปกรณ์ที่สำคัญต่างๆ หากซีพียูกับอุปกรณ์ต่างๆในระบบนั้นไม่มีความสัมพันธ์กันหรือมีการทำงานผิดพลาด หรือเชื่อมถึงกันไม่ถูกต้อง ระบบจะแจ้งเตือนให้ผู้ใช้งานทราบทันที เพื่อให้มีการแก้ไขอย่างถูกต้อง โดยการแจ้งเตือนนั้นมักจะอยู่ในรูปแบบของเสียงหรือข้อความต่างๆ เพื่อให้สามารถวิเคราะห์ถึงปัญหาที่เกิดขึ้นได้ โดยทั่วไปแล้ว การตรวจสอบตนเองจะเกิดขึ้นอย่างรวดเร็ว ยกเว้นในกรณีที่เกิดข้อผิดพลาดขึ้น

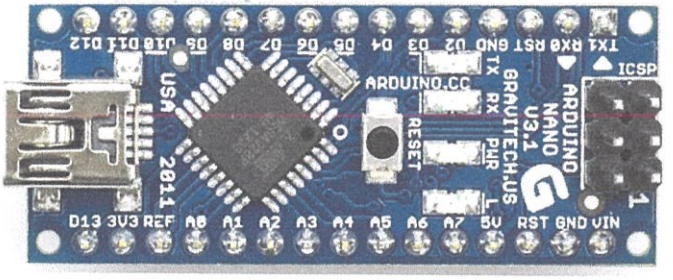
ในการพัฒนาเครื่องฝึกการหลบหมัดนั้น อุปกรณ์ที่ทำการตรวจสอบคือ ปุ่มกด ทุกๆปุ่มในระบบ นั่นเอง หากซีพียูกับปุ่มกดในระบบไม่มีความสัมพันธ์กันหรือปุ่มกดปุ่มใดมีการทำงานผิดพลาด ระบบจะแจ้งเตือนให้ผู้ใช้งานทราบผ่านทางจอภาพ VGA เพื่อให้มีการแก้ไขอย่างถูกต้อง แต่หากผู้ใช้งานต้องการใช้งานระบบในทันที ซีพียูของระบบจะตัดการเชื่อมต่อกับปุ่มกดที่มีการทำงานที่ผิดพลาดออกไป เพื่อให้ระบบนั้นสามารถทำงานต่อไปได้ตามปกติ

2.7 Arduino Nano

ในหัวข้อนี้จะขอกล่าวถึงบอร์ดไมโครคอนโทรลเลอร์สำเร็จรูป Arduino ซึ่งรวมเอาตัวไมโครคอนโทรลเลอร์และอุปกรณ์อื่นๆที่จำเป็น มาในบอร์ดเดียว เหมาะแก่สามารถนำไปพัฒนาต่อได้ง่าย

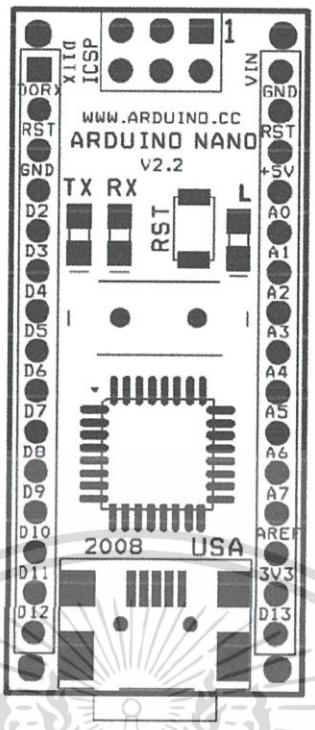
2.7.1 คุณสมบัติทั่วไป

- ไมโครคอนโทรลเลอร์	ATmega328
- แหล่งจ่ายไฟ	5V
- แรงดันอินพุต(แนะนำ)	7-12V
- แรงดันอินพุต(จำกัด)	6-20V
- ขาดิจิตอล I/O	14 ขา (6 รองรับเอาต์พุตแบบ PWM
- ขาอนาล็อกอินพุต	8 ขา
- กระแส DC ต่อขา I/O	40 mA
- Flash Memory	32 KB (ATmega328) of 2 KB used by bootloader
- SRAM	2 KB (ATmega328)
- EEPROM	1 KB (ATmega328)
- Clock Speed	16 MHz
- ขนาด	0.73”x1.70”



รูปที่ 2.10 รูปบอร์ด Arduino Nano

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.11 รูปแสดงตำแหน่งขาของบอร์ด Arduino Nano

ตำแหน่งขา	ชื่อ	ประเภท	รายละเอียด
1-2, 5-16	D0-D13	I/O	Digital input/output port 0 to 13
3, 28	RESET	Input	Reset (active low)
4, 29	GND	PWR	Supply Ground
17	3V3	Output	+3.3V output (from FTDI)
18	AREF	Input	ADC reference
19-26	A7-A0	Input	Analog input channel 0 to 7
27	+5V	Output or Input	+5V output (from on-board regulator) or +5V (input from external power supply)
30	VIN	PWR	Supply voltage

ตารางที่ 2.2 แสดงรายละเอียดหน้าที่การทำงานของขาต่างๆของ Arduino Nano

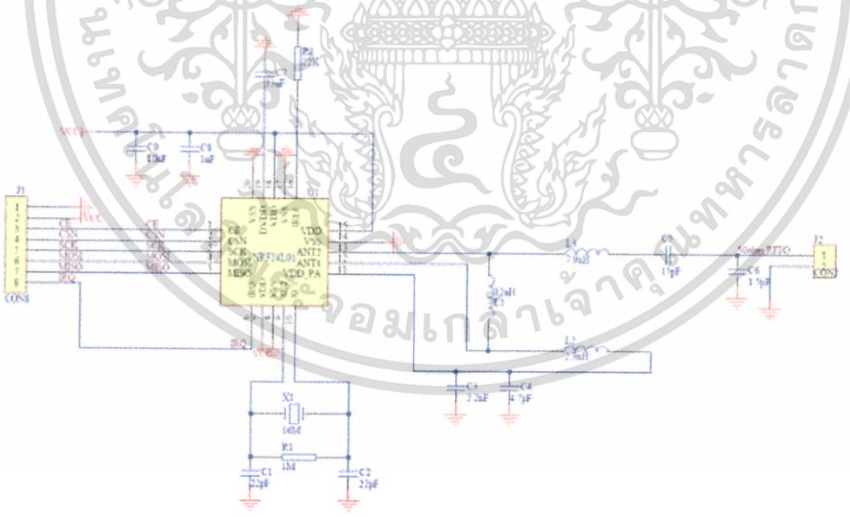
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.8 nRF24L01

โมดูล nRF24L01 เป็นโมดูลสื่อสารไร้สาย ที่สามารถเขียนโปรแกรมให้เป็นได้ ทั้งตัวรับและตัวส่ง สามารถใช้กับ Arduino ได้หลาย ๆ ตัวพร้อมกัน มีความเร็ว 2.4G จึงสื่อสารได้รวดเร็วและไม่ต้องการเสาอากาศที่ยาว มีขนาดเล็กสะดวกในการต่อใช้งาน สามารถประยุกต์ใช้งานได้หลายอย่างเช่น ใช้เป็นอุปกรณ์ส่งข้อมูลของเซนเซอร์อัตโนมัติสำหรับควบคุม อุณหภูมิ ความชื้น การแจ้งเตือนต่าง ๆ ควบคุมและติดตามหุ่นยนต์ Robot Control and Monitoring ได้ในระยะ 15-500 เมตร โมดูลนี้ใช้ชิป NRF24L01+m ทำงานด้วยความเร็วสูง High-speed SPI interface ใช้พลังงานต่ำ รองรับการทำงานร่วมกับ Arduino และมีเสาอากาศมาให้ในตัว

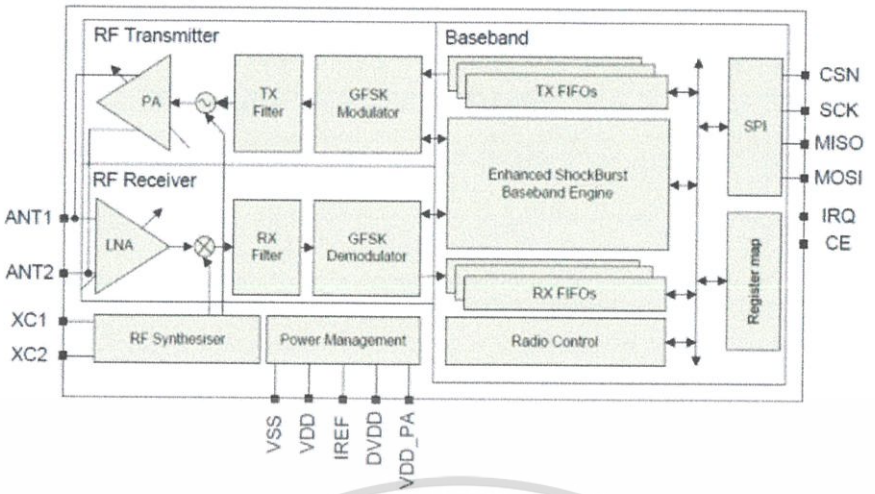
2.8.1 คุณสมบัติทั่วไป

- รองรับสัญญาณ 2.4 GHz ISM band (free, unlicensed band)
- อัตราการรับส่งข้อมูล 250 kbps, 1 Mbps and 2 Mbps
- กินกระแสต่ำ 11.3 mA Tx with 1 mW output power, 26 μA ในโหมด standby-I, 900 nA ในโหมด power down
- แรงดันไฟเลี้ยง 1.9 – 3.6V (ทนได้ถึง 5V)
- Automatic acknowledge sending with automatic retries
- RX and TX FIFO's with ACK user data possibility
- Simple 8 pin (7 pin without IRQ) SPI interface: VCC, GND, CE, CSN, SCK, MISO, MOSI and optional IR



รูปที่ 2.12 แสดงวงจรสมบูรณ์ของ โมดูล nRF24L01

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.13 Block diagram ของ โมดูล nRF24L01

การเชื่อมต่อ Pin กับ SPI ของโมดูล nRF24L01 ดังนี้

1. **GND** : Ground.
2. **VCC** : 3.3V.
3. **CE** : Chip (RX/TX) Enable
4. **CSN** : Chip Select Not
5. **SCK** : SPI Shift Clock, up to 10 MHz.
6. **MOSI :** Master-Out-Slave-In, used to shift data from the microcontroller to the device.
7. **MISO** : Master-In-Slave-Out, used to shift data from the device to the microcontroller.
8. **IRQ** : Optional Interrupt Request pin. Signals RX/TX status like packet sent or received.

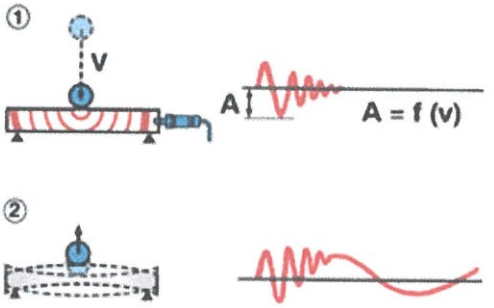
2.9 Accelerometer และ ADXL335

Accelerometer คือ เครื่องตัววัดความเร่งของการเคลื่อนที่ของวัตถุ โครงสร้างของ accelerometer จะประกอบด้วยสปริงและลูกตุ้มน้ำหนัก เมื่อมีการเคลื่อนที่ด้วยความเร่งลูกตุ้มน้ำหนักจะถูกกดไปอีกฝั่งตรงข้ามกับการเคลื่อนที่ สปริงก็ทำหน้าที่ดึงกลับเข้าที่อีกครั้งเมื่อหยุดการเคลื่อนที่ การเคลื่อนที่ด้วยความเร็วคงที่คือความเร่งเท่ากับศูนย์ ค่าที่วัดได้ก็จะไม่เปลี่ยนแปลง ตัวเซ็นเซอร์ภายในที่จะใช้ในการตรวจวัดความเร่งของลูกตุ้มที่อยู่ในระบบนั้นมีหลายชนิด เช่น เพียโซอิเล็กทริค สเตรนเกจ เป็นต้น
โดยที่สามารถแบ่งลักษณะการตรวจวัดได้ 2 ลักษณะ

1. การตรวจวัดการช็อก (Shock) และการสั่นสะเทือน (Vibration)

2. การตรวจวัดอัตราเร่งของวัตถุ เพื่อนำข้อมูลไปใช้ในการระบุตำแหน่ง ความเร็ว และระยะทางที่ได้จากการเคลื่อนที่

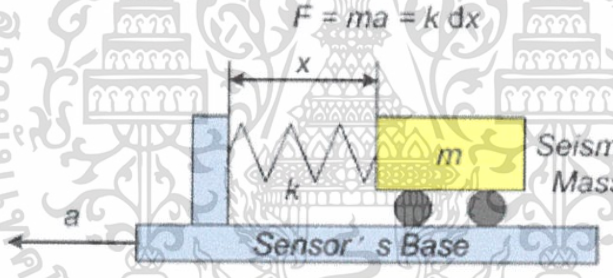
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.14 ลักษณะการตรวจวัดความเร่ง

มิเตอร์วัดความเร่งนี้โดยหลักๆแล้วจะแบ่งเป็น 2 ชนิด

1. มิเตอร์วัดอัตราเร่งแบบไซซมิกแมส (seismic mass accelerometer) มิเตอร์ชนิดนี้อาศัยหลักการตรวจวัดระยะขจัดเชิงเส้นแล้วนำไปคำนวณหาอัตราเร่งที่เกิดขึ้น โดยวัตถุชิ้นหนึ่งจะมีความเร่งได้ ก็จะต้องมีแรงมากระทำ ยิ่งมีแรงมากระทำมาก ก็จะยิ่งมีความเร่งมาก ในขณะเดียวกันแรงต้านการเคลื่อนที่ก็จะมากด้วย นอกจากนี้เมื่อมีแรงมาทำให้วัตถุเกิดการเคลื่อนที่ ก็จะมีระยะขจัด ซึ่งก็จะแปรผันตรงกับแรงที่มากระทำที่วัตถุ ยิ่งแรงมากระยะขจัดยิ่งมาก จากความสัมพันธ์ดังกล่าวได้นำไปใช้เป็นหลักการพื้นฐานของมิเตอร์วัดอัตราเร่งแบบไซซมิกแมสในการตรวจวัดอัตราเร่งของวัตถุในเทอมของระยะขจัดที่เกิดขึ้น



รูปที่ 2.15 โครงสร้างพื้นฐานของมิเตอร์วัดอัตราเร่งแบบไซซมิกแมส

โครงสร้างนี้มีมวล m ที่เรียกว่ามวลตรวจการสั่นไหว (seismic mass) ยึดติดอยู่กับสปริงที่มีค่า spring constant เท่ากับ k และมวลนี้สามารถเคลื่อนที่ในแนวระดับได้ เมื่อตัวเซนเซอร์ตัวนี้ถูกทำให้มีอัตราเร่งเกิดขึ้นจะส่งผลให้มวล m เคลื่อนที่ ซึ่งระยะที่เคลื่อนที่ออกไปจะเป็นระยะขจัดเท่ากับ x และมีทิศทางตรงกันข้ามกับการเคลื่อนที่ของตัวมิเตอร์

ดังนั้นอัตราเร่ง a ของวัตถุสามารถคำนวณหาค่าได้จากความสัมพันธ์ต่อไปนี้

$a = xk/m$ โดยที่

a คือ อัตราเร่งของวัตถุ หน่วย เมตร/วินาที

x คือ ระยะขจัดของมวล m หน่วย เมตร

k คือ ค่าคงที่ของสปริง หน่วย นิวตัน/เมตร

m คือ น้ำหนักของมวล m หน่วย กิโลกรัม

จากสมการดังกล่าวจะแสดงให้เห็นว่า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

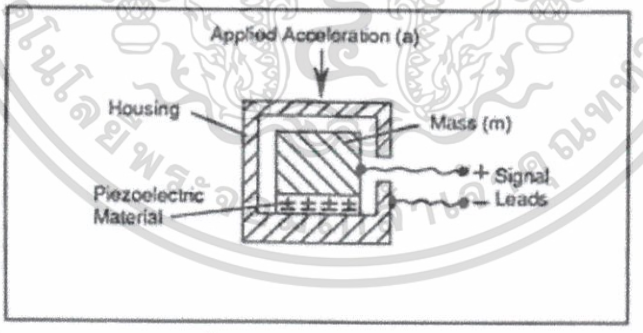
- เมื่ออัตราเร่งของวัตถุมีค่าเพิ่มขึ้น ทำให้ระยะขจัดของมวล m มีค่าเพิ่มขึ้นตามไปด้วย
- เมื่ออัตราเร่งของวัตถุม่ค่าลดลง ทำให้มวล m เคลื่อนที่ไปดันสปริง
- เมื่ออัตราเร่งของวัตถุหยุดลง ก็จะทำให้มวล m เคลื่อนที่กลับมาอยู่ตำแหน่งเดิม (ตำแหน่งอ้างอิง)

แต่ในทางปฏิบัติเราสามารถวัดระยะขจัดของมวล m ได้โดยอาศัยมิเตอร์อีกชนิดหนึ่ง คือมิเตอร์วัดระยะขจัดเชิงเส้น (LVDT, potentiometer)

ส่วนการวิเคราะห์หาค่าอัตราเร่งที่เกิดขึ้นเราสามารถคำนวณหาได้โดยใช้คอมพิวเตอร์ มิเตอร์วัดอัตราเร่งแบบไซซมิกแมสนี้จะนิยมใช้ในการตรวจวัดลักษณะการช็อกและลักษณะการสั่นสะเทือนที่มีความถี่ต่ำมากๆ เช่น ในเครื่องมือตรวจวัดแผ่นดินไหว หรือในเครื่องมือตรวจวัดการปะทุใต้ดินของภูเขาไฟ ฯลฯ

2. มิเตอร์วัดอัตราเร่งแบบเพียโซอิเล็กทริก (piezoelectric accelerometer)

คุณสมบัติพื้นฐานทางไฟฟ้าของผลึกเพียโซอิเล็กทริก (piezoelectric crystal) ถูกค้นพบโดย Pierre และ Jacques Curie ในราวปี ค.ศ.1880 ซึ่งเจ้า piezoelectric crystal นี้มันมีคุณสมบัติพิเศษคือ เมื่อมันถูกแรงทางกลมากระทำ มันจะสร้างประจุไฟฟ้าขึ้นมา โดยเป็นสัดส่วนกับแรงกระทำนั้นซึ่งจากคุณสมบัติพิเศษนี้ได้ถูกดัดแปลงนำไปใช้สร้างอุปกรณ์ต่างๆมากมาย เช่น ใช้เป็นแบตเตอรี่จ่ายพลังงานไฟฟ้าให้กับนาฬิกาข้อมือดิจิตอลที่เราใช้ทั่วไป และยังใช้สร้างมิเตอร์วัดอัตราเร่งแบบเพียโซอิเล็กทริกอีกด้วย โครงสร้างของมอเตอร์วัดอัตราเร่งแบบเพียโซอิเล็กทริกจะประกอบด้วย seismic mass ยึดติดกับ piezoelectric crystal และบรรจุอยู่ในตัวถังป้องกัน โดย piezoelectric crystal ที่นิยมนำมาใช้งานได้แก่ ผลึกควอตซ์ และผลึกโซเดียมโปตัสเซียมตาเตรต (sodium potassium tartrate) เพราะมีความทนทานต่อแรงกระทำ และราคาไม่แพงมากนัก



รูปที่ 2.16 โครงสร้างพื้นฐานของมิเตอร์วัดอัตราเร่งแบบเพียโซอิเล็กทริก

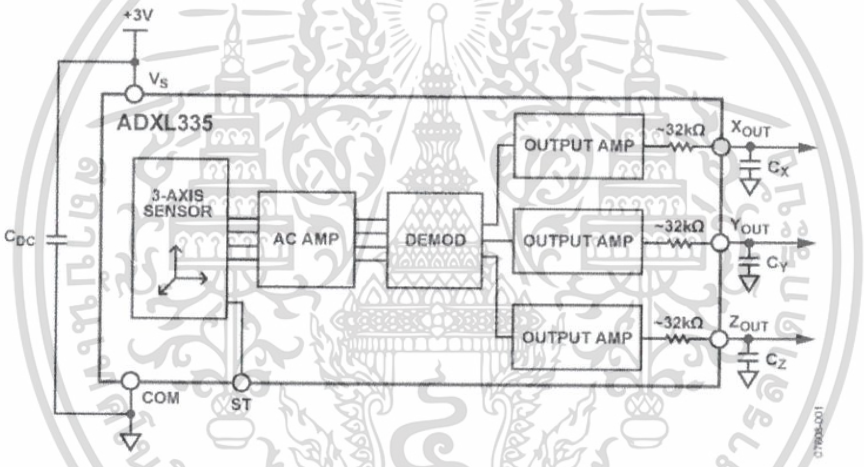
เมื่อ seismic mass (m) ถูกทำให้เกิดอัตราเร่งขึ้น (ถูกกด) มันจะส่งผ่านแรงกดไปกระทำกับ piezoelectric crystal ที่ถูกยึดติดอยู่ด้วยกัน ด้วยคุณสมบัติพิเศษของมันจะทำให้ประจุไฟฟ้าถูกสร้างขึ้นและถูกสายนำสัญญาณออกไปยังเอาต์พุตของวงจร โดยที่ด้านเอาต์พุตจะต้องมีวงจรขยายประจุไฟฟ้า (charge amplifier) เพื่อขยายค่าประจุไฟฟ้าที่ได้ให้เป็นแรงดันเอาต์พุตตามสัดส่วนของอัตราเร่งที่เกิด จะได้สามารถแสดงผลได้ด้วยโวลต์มิเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

มิเตอร์วัดอัตราเร่งแบบเพียโซอิเล็กทริกตอบสนองต่อทางด้านความถี่สูงได้ดี แต่ในทางกลับกันก็จะมีผลตอบสนองทางด้านความถี่ต่ำที่ไม่ดีนัก มีขนาดค่อนข้างเล็ก น้ำหนักเบา และสามารถใช้งานที่มีอัตราเร่งได้สูงถึง 250,000 m.s-2

2.9.1 คุณสมบัติทั่วไปของ ADXL335

- มีเซนเซอร์สามแกน
- แพ็กเกจมีขนาดเพียง 4 mm × 4 mm × 1.45 mm LFCSP
- กินพลังงานต่ำเพียง 350 μA (typical)
- Single-supply operation 1.8 V to 3.6 V
- 10,000 *g* shock survival
- เสถียรเมื่อใช้งานที่อุณหภูมิต่างๆ
- BW adjustment with a single capacitor per axis
- RoHS/WEEE lead-free compliant



รูปที่ 2.17 Block diagram ของ ADXL335

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

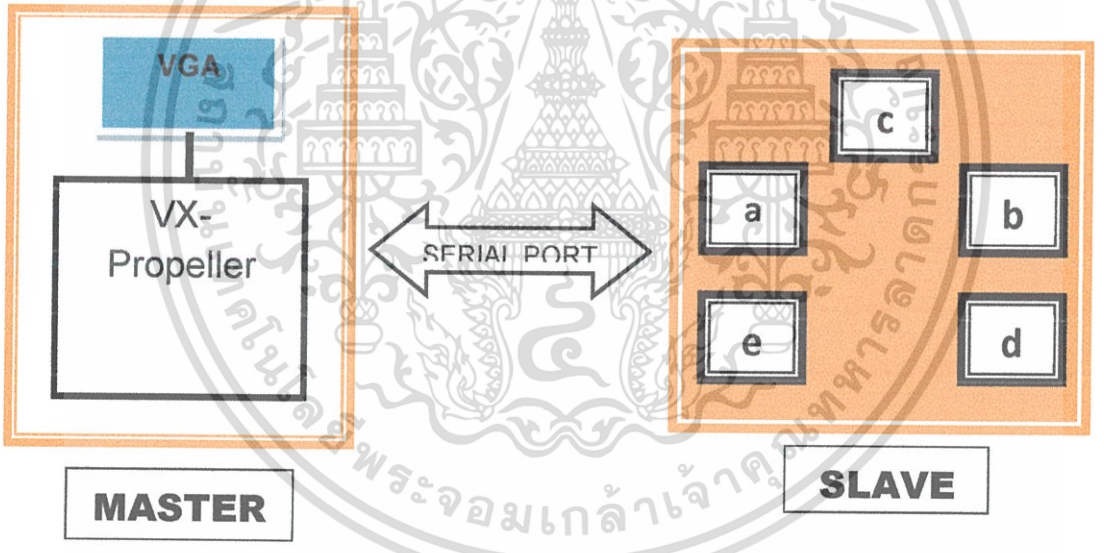
บทที่ 3

การออกแบบ

3.1 การออกแบบโหมดทดสอบนักกีฬา

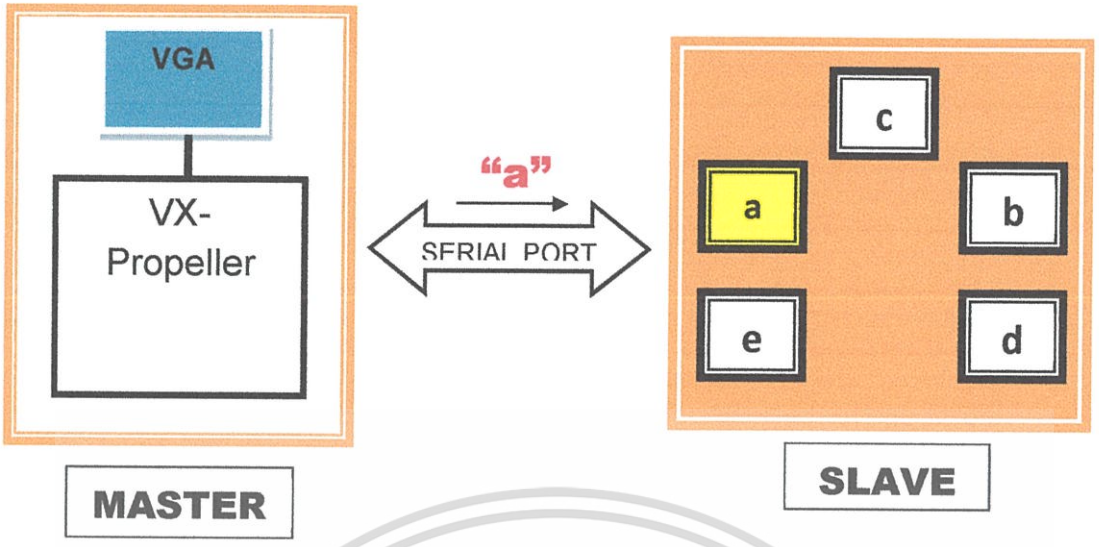
การพัฒนาเครื่องฝึกการหลบหมัดนั้นถูกแบ่งการทำงานออกเป็น 2 ภาค ได้แก่ ภาคควบคุม (MASTER) และ ภาคถูกควบคุม (SLAVE) การทำงานทั้ง 2 ภาคนั้นจะถูกเชื่อมต่อเข้าถึงกันผ่านทางพอร์ตอนุกรม RS-232 ดังแสดงได้ดังรูปที่ 3.1 โดยภาค MASTER จะประกอบไปด้วยไมโครคอนโทรลเลอร์ VX-Propeller และจอภาพ VGA ใช้สำหรับการแสดงผล ส่วนภาค SLAVE นั้นก็คือปุ่มกดแต่ละปุ่ม

เมื่อเริ่มต้นการทดสอบ VX-Propeller จะส่งรหัสตัวอักษรผ่านทางพอร์ตอนุกรม RS-232 ไปยังปุ่มกดทุกปุ่ม หากรหัสอักษรที่ส่งไปนั้นตรงกับรหัสประจำตัวของปุ่มกดปุ่มใด ปุ่มกดนั้นจะมีไฟติดขึ้นมา ดังแสดงได้ดังรูปที่ 3.2

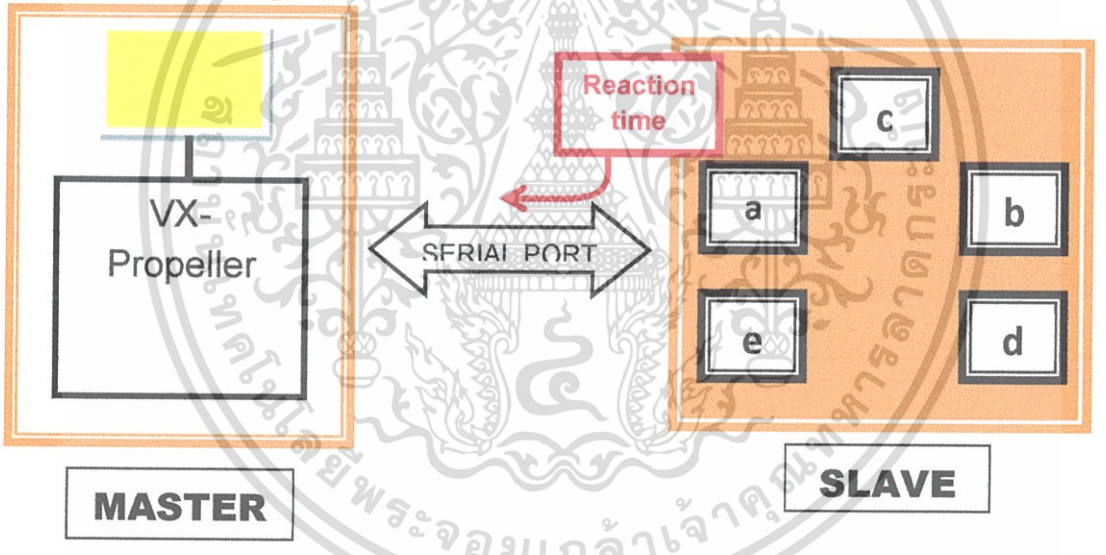


รูปที่ 3.1 โครงสร้างการทำงานของระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.2 การติดไฟของปุ่มกด



รูปที่ 3.3 การส่งค่า Reaction time กลับมายังภาค MASTER และการแสดงผลทางจอภาพ VGA

ไมโครคอนโทรลเลอร์บนปุ่มกดจะทำการจับเวลา ตั้งแต่ตอนที่ไฟบนปุ่มกดสว่างขึ้น จนกระทั่งไฟบนปุ่มกดดับลง เมื่อเสร็จสิ้นการจับเวลาแล้ว ปุ่มกดจะส่งค่าเวลาที่จับได้กลับมายังภาค MASTER ผ่านทางพอร์ตอนุกรม ดังแสดงได้ดังรูปที่ 3.3 โดย VX –Propeller จะทำการรับค่าเวลา และส่งค่าเวลานี้ออกไปยังจอภาพ VGA เพื่อทำการแสดงผล เพื่อให้ผู้ทดสอบได้รับทราบว่าการทดสอบในรอบแรกนั้นได้เสร็จสิ้น หลังจากนั้น VX-Propeller จะทำการส่งรหัสตัวอักษรตัวต่อไป ออกไปยังปุ่มกด เพื่อทำการทดสอบในรอบต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปแบบในการทดสอบนักกีฬานั้นถูกแบ่งออกเป็น 2 โหมด ได้แก่

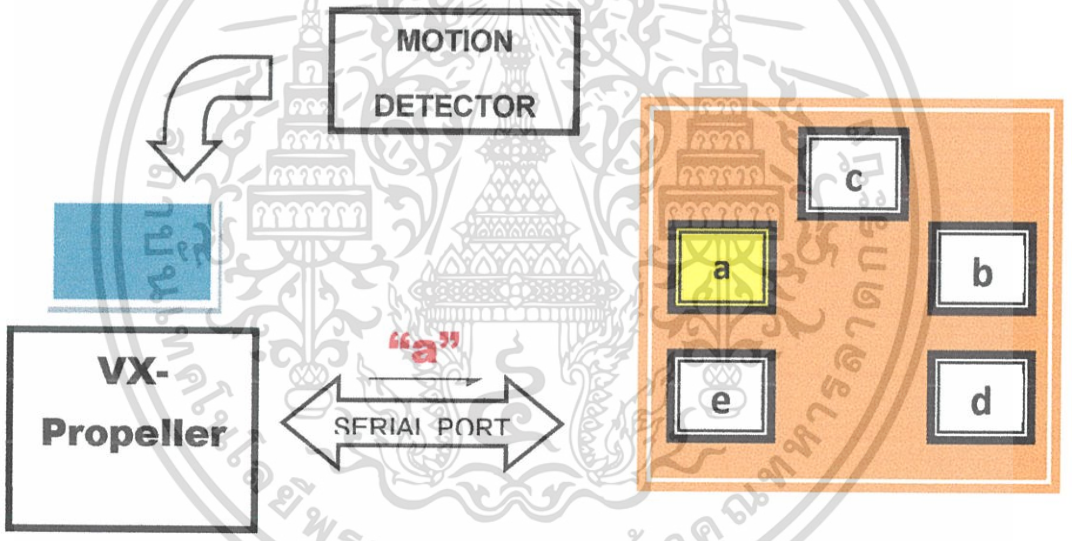
1. โหมดโจมตี (Attack Mode)
2. โหมดป้องกัน (Defense Mode)

3.1.1 โหมดโจมตี (Attack Mode)

โหมดโจมตีคือโหมดที่ใช้ทดสอบนักกีฬาในขณะโจมตีคู่ต่อสู้ โดยใช้หลักการของการพัฒนาเครื่องฝึกการหลบหมัดคือ เมื่อเริ่มทดสอบ VX Propeller จะส่งรหัสตัวอักษรมาทำให้ไฟบนปุ่มสว่างขึ้น ผู้ทดสอบต้องทำการกดปุ่มเพื่อทำให้ไฟดับทันที เปรียบเสมือนการโจมตีขณะที่คู่ต่อสู้เปิดโอกาสในการโจมตีจุดต่างๆบนร่างกายนั่นเอง

3.1.2 โหมดป้องกัน (Defense Mode)

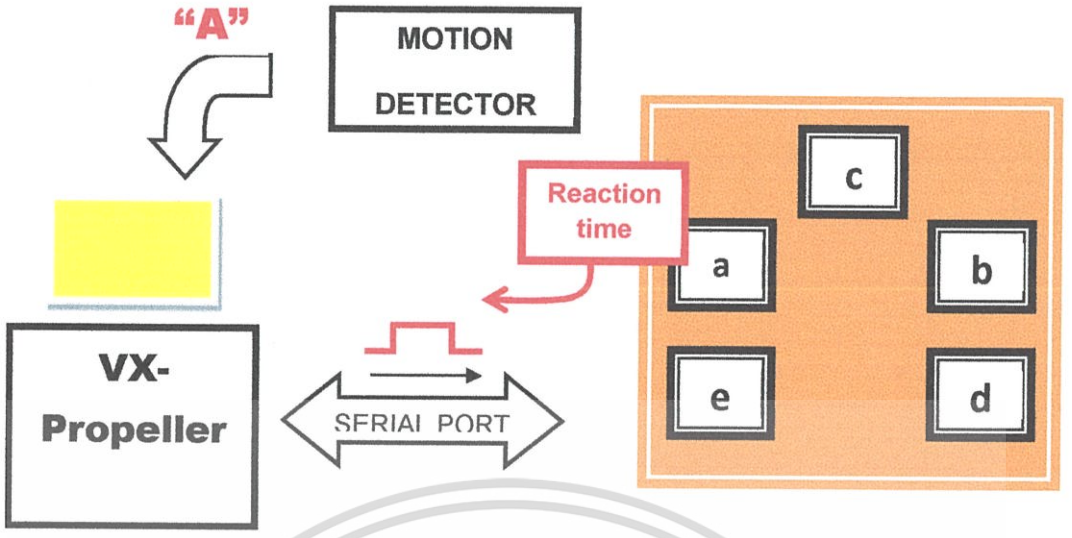
โหมดป้องกันคือโหมดที่ใช้ทดสอบนักกีฬาในขณะตั้งรับ หลักการทดสอบมีดังนี้ เมื่อเริ่มทดสอบ VX Propeller จะส่งรหัสตัวอักษรมาทำให้ไฟบนปุ่มสว่างขึ้นดังรูป 3.4 ผู้ทดสอบต้องทำการโยกศีรษะในทิศทางตรงกันข้ามเพื่อปิดไฟทันที เปรียบเสมือนกับการโยกหลบการโจมตีจากคู่ต่อสู้ของนักกีฬา



รูปที่ 3.4 การทำงานของโหมดป้องกันขณะเปิดไฟโมดูลปุ่มกด

เมื่อผู้ทดสอบโยกศีรษะวงจรตรวจจับการเคลื่อนไหวจะส่งตัวอักษรที่แสดงทิศทางของการโยกหลบไปที่ Vx-Propeller ถ้าการโยกหลบถูกต้อง Vx-Propeller จะส่งสัญญาณ pulse ไปปิดไฟแล้วรอรับค่าเวลาการตอบสนองที่ได้ไปแสดงผล ดังแสดงในรูป 3.5

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.5 การทำงานของโหมดป้องกันเมื่อมีการตอบสนอง

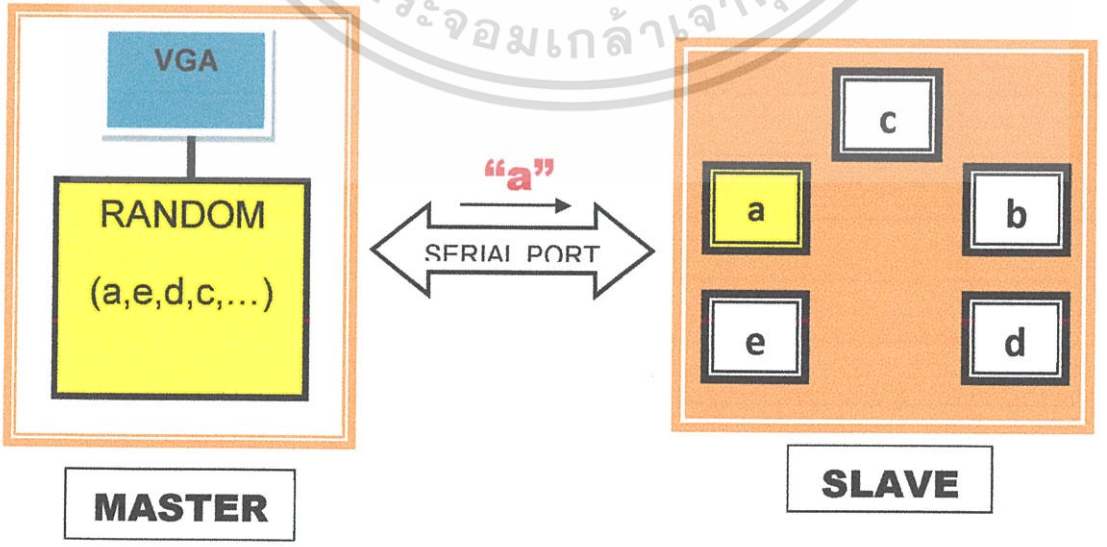
3.2 โหมดทดสอบย่อย

รูปแบบในการทดสอบนักกีฬาทั้งสองแบบยังแบ่งออกเป็น 3 โหมดย่อย ได้แก่

1. โหมดการทดสอบแบบสุ่มตำแหน่ง (RANDOM MODE)
2. โหมดการทดสอบรูปแบบเฉพาะ (PATTERN MODE)
3. โหมดการทดสอบแบบกำหนดเอง (MANUAL MODE)

3.2.1 โหมดการทดสอบแบบสุ่มตำแหน่ง (RANDOM MODE)

โหมดการทดสอบแบบสุ่มตำแหน่ง คือ โหมดการทดสอบนักกีฬาโดยการสุ่มเลือกปุ่มกดที่ต้องการให้ไฟติด ไม่มีรูปแบบของลำดับการติดไฟที่แน่นอน โดยมีหลักการทำงานดังนี้ เมื่อเริ่มทำการทดสอบ VX-Propeller จะทำการสุ่มรหัสตัวอักษรที่จะส่งออกไปยังปุ่มกด เพื่อให้ปุ่มกดนั้นไฟติดขึ้น ดังแสดงได้ดังรูปที่ 3.6

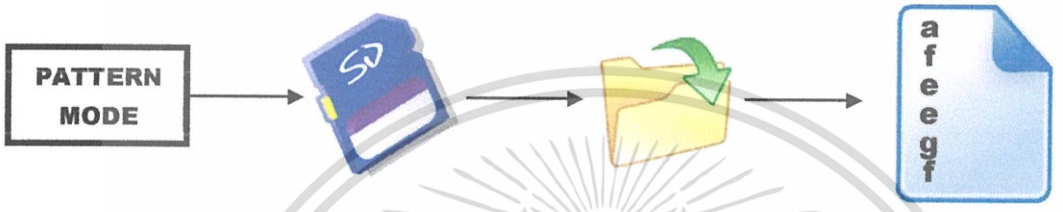


รูปที่ 3.6 การทำงานของระบบในโหมดการทดสอบแบบสุ่มตำแหน่ง

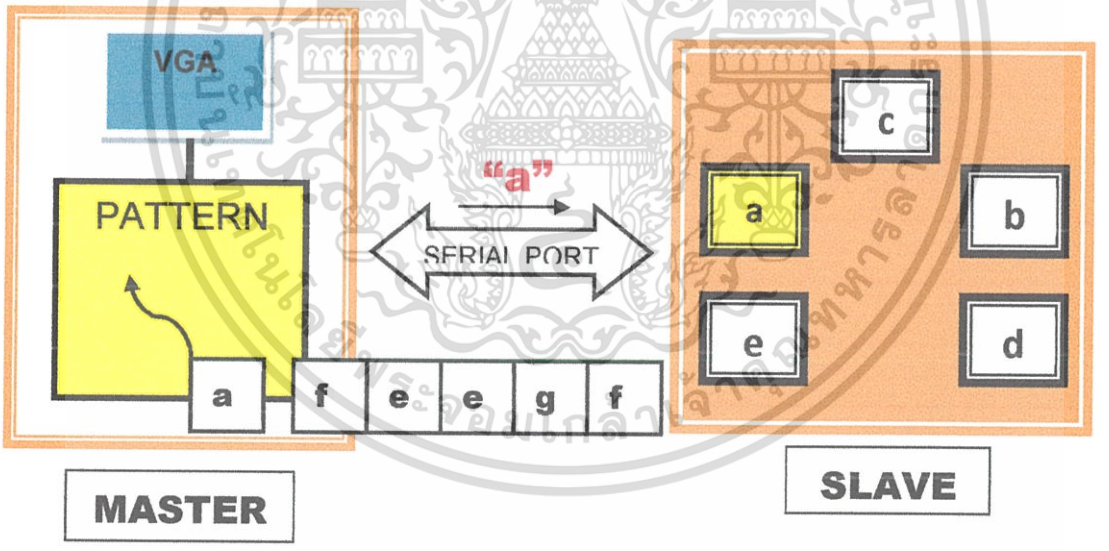
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.2 โหมดการทดสอบรูปแบบเฉพาะ (PATTERN MODE)

โหมดการทดสอบรูปแบบเฉพาะ คือ โหมดทดสอบนักกีฬาที่สามารถกำหนดลำดับการติดไฟของปุ่มกด ล่วงหน้าได้ โดยการสร้างไฟล์รูปแบบซึ่งบันทึกลำดับการติดไฟไว้ ทำการบันทึกไฟล์รูปแบบดังกล่าวลงใน SD CARD เพื่อนำไปใช้งานกับระบบ โดยมีหลักการทำงานดังนี้ เมื่อเริ่มทำการทดสอบ VX-Propeller จะทำการเปิดไฟล์รูปแบบที่ผู้ทดสอบเลือกจาก SD CARD หลังจากนั้นจะส่งรหัสอักษรออกไปทีละตัวอักษรตามลำดับที่บันทึกไว้ในไฟล์รูปแบบ ดังแสดงได้ดังรูปที่ 3.7 และ 3.8



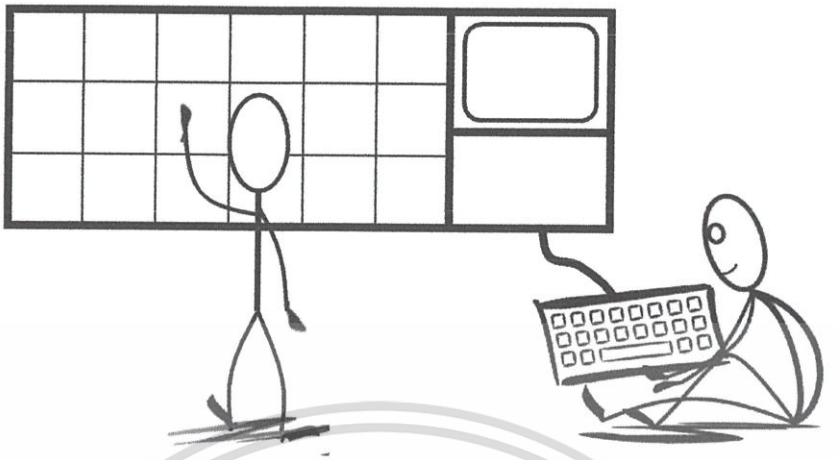
รูปที่ 3.7 การเปิดไฟล์รูปแบบเพื่อนำไปใช้ในโหมดการทดสอบรูปแบบเฉพาะ



รูปที่ 3.8 การทำงานของระบบในโหมดการทดสอบรูปแบบเฉพาะ

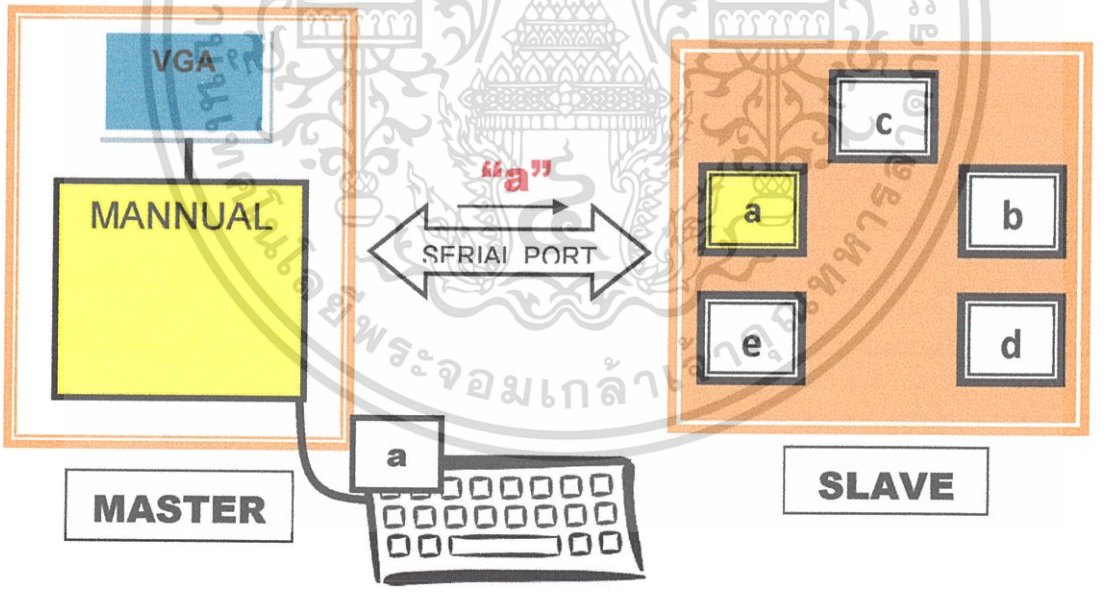
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.3 โหมดการทดสอบแบบกำหนดเอง (MANUAL MODE)



รูปที่ 3.9 แสดงการทดสอบในโหมดการทดสอบแบบกำหนดเอง

โหมดการทดสอบแบบกำหนดเอง คือ โหมดทดสอบนักกีฬาที่สามารถเลือกปุ่มกดที่ต้องการให้ไฟติดได้ โดยการสั่งการผ่านทางคีย์บอร์ด โหมดทดสอบนี้จะต้องมีผู้ร่วมทดสอบหนึ่งคน เพื่อทำหน้าที่ในการควบคุมการสั่งการทางคีย์บอร์ด ดังแสดงได้ดังรูปที่ 3.9 และ 3.10



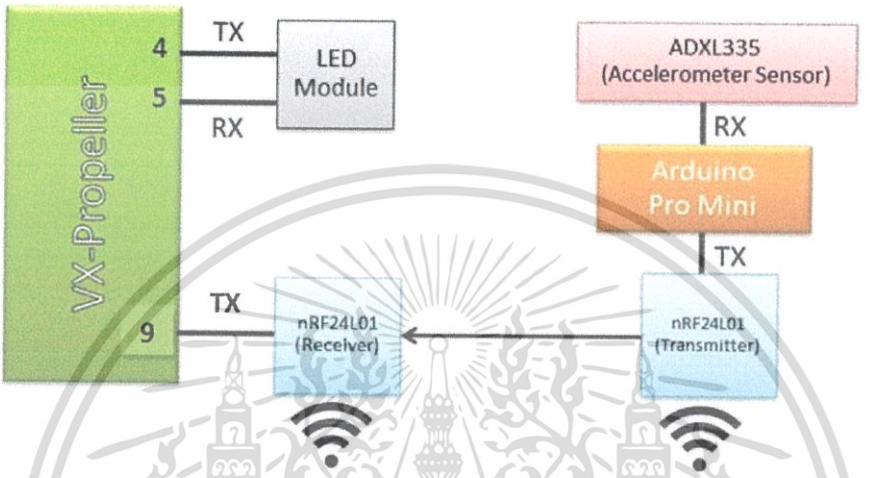
รูปที่ 3.10 การทำงานของระบบในโหมดการทดสอบแบบกำหนดเอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

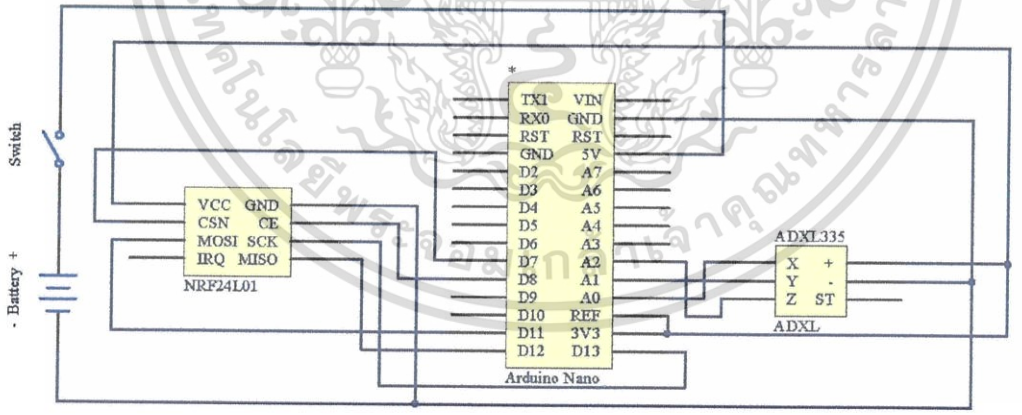
3.3 การออกแบบด้าน Hardware

3.3.1 วงจรรับส่งสัญญาณไร้สาย

วงจรส่งสัญญาณไร้สายใช้โมดูล nRF24L01 เป็นตัวรับและตัวส่งสัญญาณ หลักการทำงานคือ ภาคส่งจะรับค่าความเร่งจากแกน x y z มาจาก ADXL335 แล้วส่งตัวอักษรแสดงถึงการเคลื่อนไหวนั้นไปยังภาครับ ภาคส่งจะทำงานด้วยคำสั่งจาก Arduino Nano ภาครับจะทำงานด้วยคำสั่งจาก Arduino pro mini ตัวอักษรที่ได้จะถูกส่งต่อไปยังขา P9 ของ VX-propeller

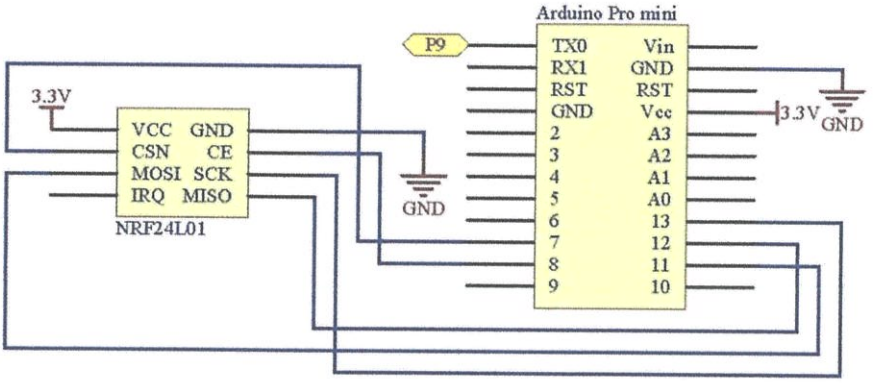


รูปที่ 3.11 โครงสร้างของวงจรรับส่งสัญญาณไร้สาย



รูปที่ 3.12 วงจรภาคส่งสัญญาณไร้สาย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



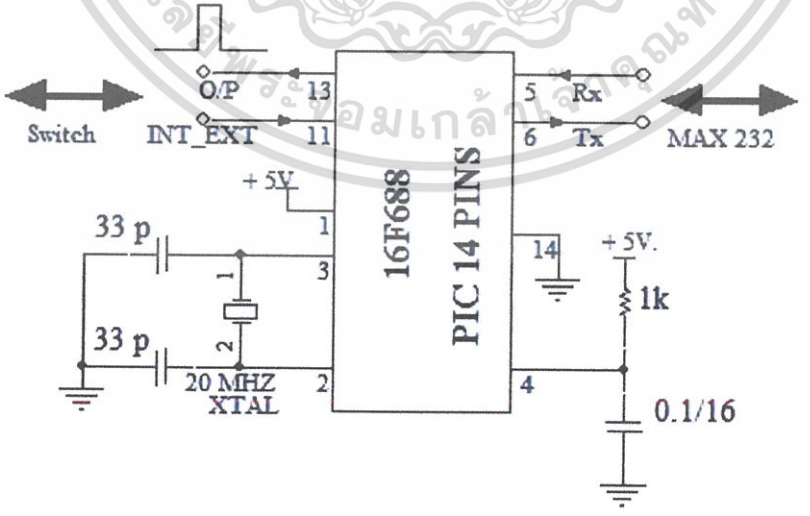
รูปที่ 3.13 วงจรภาครับสัญญาณไร้สาย

3.3.2 วงจร PIC 16F688

เลือกใช้ PIC เบอร์ 16F688 ประกอบเข้ากับโมดูลปุ่มกด เพื่อทำหน้าที่กระตุ้นให้ SCR ทำงาน และรับพัลส์จากการกดสวิทช์ตอบสนอง พร้อมกับนับพัลส์ที่สร้างขึ้น 16F688 มีคุณสมบัติที่สำคัญดังนี้

- มี 14 ขา 12 I/O pins (รูปร่างค่อนข้างเล็ก)
- สามารถใช้ Oscillator จากภายนอกได้
- มีไทเมอร์0 และ ไทเมอร์1
- มี Power-on Reset (POR)
- รองรับการใช้งานพอร์ตอนุกรม RS-232
- รองรับการใช้งานอินเตอรรัปต์จากภายนอก (External Interrupt)
- สามารถลบและเขียนข้อมูลใหม่ได้มากกว่า 100,000 ครั้ง

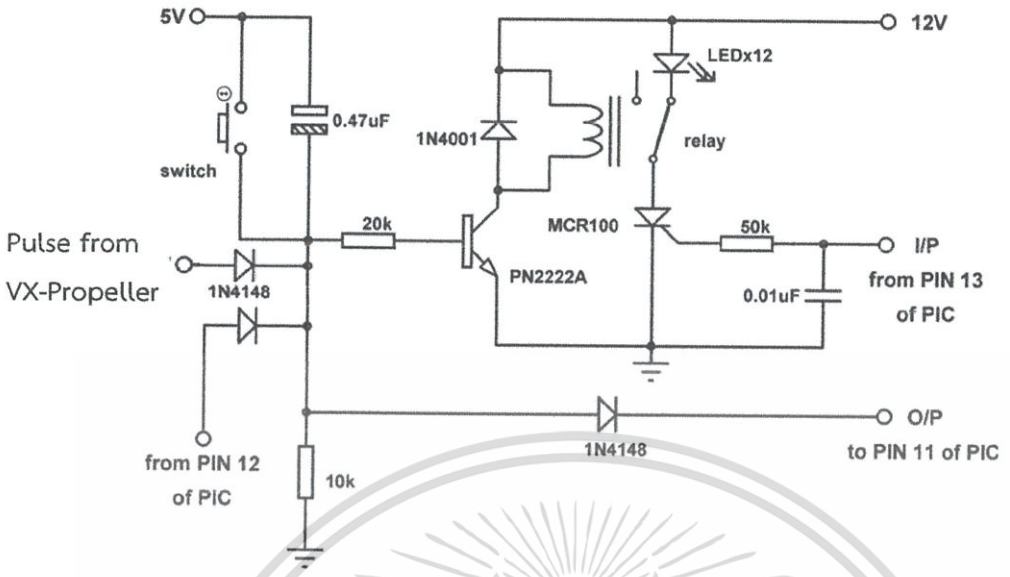
จะได้รูปวงจรดังนี้



รูปที่ 3.14 วงจรรวมของPIC 16F688 เพื่อควบคุม Module

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3.3 วงจรโมดูลปุ่มกด



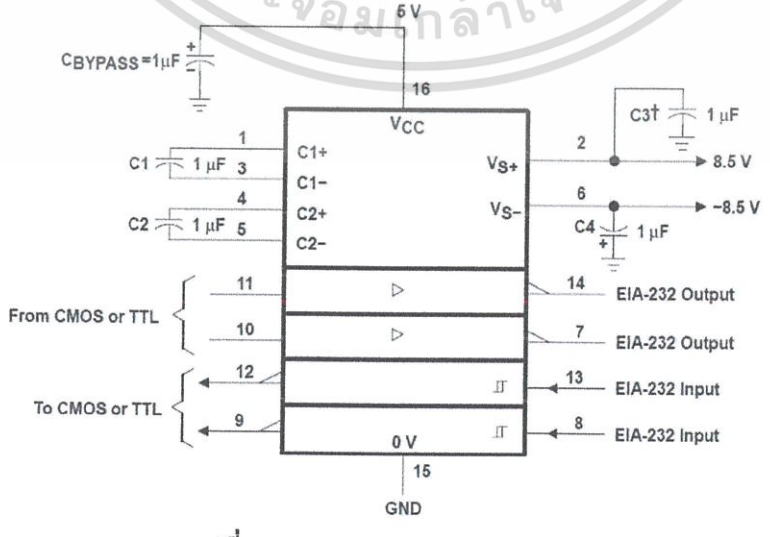
รูปที่ 3.15 วงจรของโมดูลปุ่มกด

หลักการทำงานของวงจรคือ

1. เมื่อมีพัลส์เข้ามาทางขาเกตของ SCR จะกระตุ้นให้ SCR ทำงาน (ON) ไฟจะติดทันที
2. ถ้ายังไม่มีการกดสวิตช์หรือ pulse จากขา 12 SCR จะไม่ OFF ไฟก็ยังคงติดต่อไป
3. เมื่อมีการกดสวิตช์หรือมี pulse จากขา 12 ทรานซิสเตอร์จะทำงานหรือมี pulse จาก Vx-Propeller กระแสไหลผ่าน Relay เมื่อ Relay ทำงาน จะทำให้วงจรที่ขาแอโนดของ SCR เปิด ทำให้ SCR หยุดการทำงาน (OFF) ไฟก็จะดับลงทันที

3.3.4 วงจร MAX 232

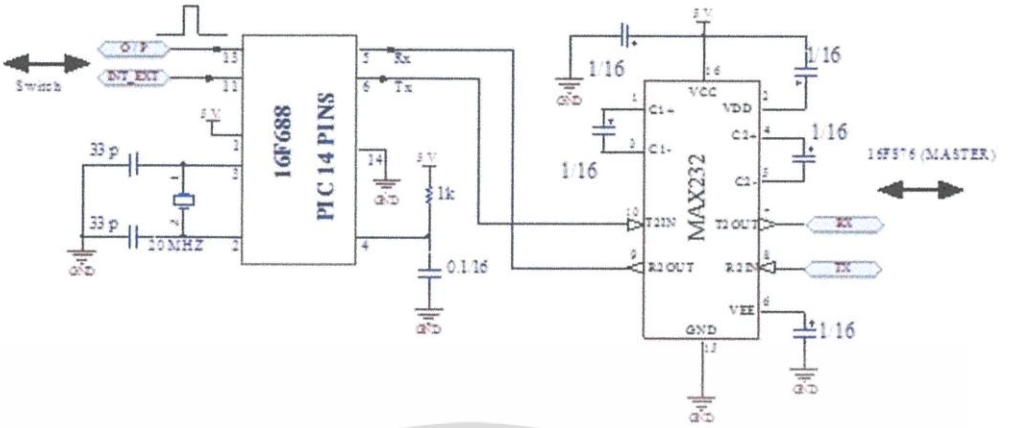
MAX232 เป็นไอซีที่แปลงระดับสัญญาณของ RS-232 มาเป็นระดับ TTL และใช้แปลงระดับสัญญาณ TTL ไปเป็นระดับสัญญาณ RS-232 มีการต่อวงจร ดังรูป



รูปที่ 3.16 วงจรของ IC MAX232

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

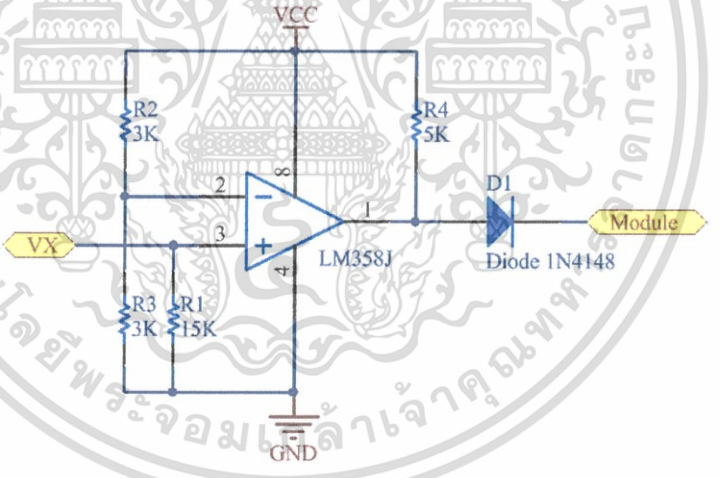
3.3.5 วงจร PIC 16F688



รูปที่ 3.17 วงจรรวมของPIC 16F688 เพื่อควบคุม Module

3.3.6 วงจร Comparator

การปิดไฟโมดูลปุ่มกดในโหมดป้องกัน Vx-Propeller จะส่งสัญญาณ pulse ขนาด 3.3 V ไปที่วงจร Comparator ซึ่งทำหน้าที่เปรียบเทียบแรงดันของ pulse กับแรงดันอ้างอิงขนาด 3 V แรงดัน output ขนาด 5 V จะถูกส่งไปที่วงจรโมดูลปุ่มกดเพื่อปิดไฟ



รูปที่ 3.18 วงจร Comparator

$$V_{ref} = \left(\frac{R_3}{R_2+R_3}\right) V_{cc} \qquad \text{---------- (1)}$$

$$= \frac{3\text{K}}{6\text{K}}(6\text{V})$$

$$= 3\text{V}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4 การออกแบบด้าน Software

3.4.1 หลักการทำงานของโครงงานด้าน Software

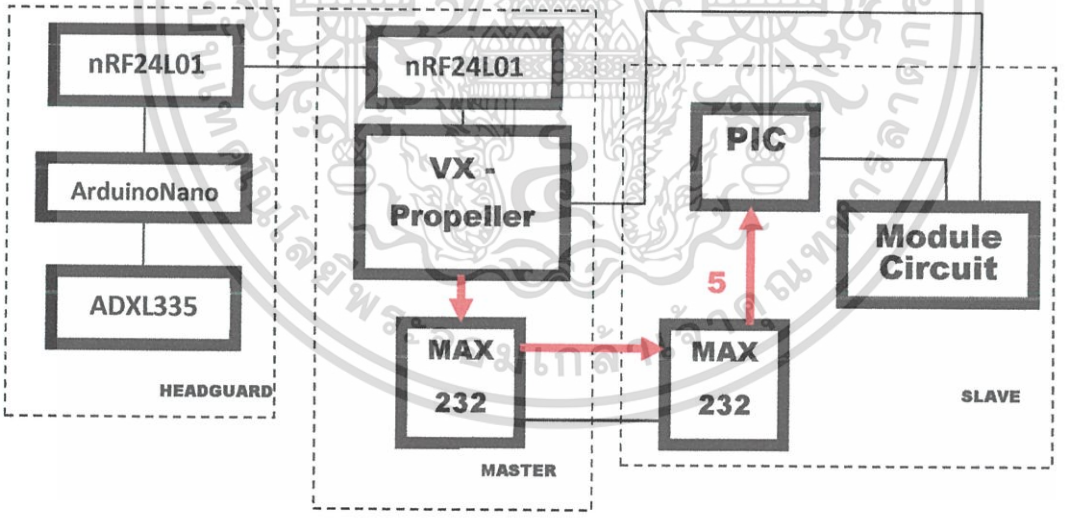
อธิบายหลักการทำงานอย่างละเอียดได้ดังนี้

- เริ่มแรกระบบจะทำการตรวจสอบก่อนว่า SD CARD นั้นได้เชื่อมต่ออยู่หรือไม่ หากพบว่าไม่ได้เชื่อมต่อ ระบบจะแสดงข้อความแจ้ง และหยุดการทำงานของระบบ เมื่อเชื่อมต่อ SD CARD แล้ว ผู้ใช้งานจะต้องทำการรีเซตระบบใหม่

- หากพบว่า SD CARD ได้ถูกเชื่อมต่อแล้ว ระบบจะเปิดหน้าเมนูหลักสำหรับการเลือกโหมดการทดสอบขึ้นมา โดยผู้ใช้งานสามารถเลือกโหมดการทดสอบได้ผ่านทางคีย์บอร์ด

- เมื่อผู้ใช้งานได้เลือกโหมดการทดสอบแล้ว ระบบจะให้ผู้ใช้งานได้เซ็ตค่าต่างๆ ก่อนการทดสอบ เช่น จำนวนรอบในการทดสอบ เป็นต้น

- เมื่อผู้ใช้งานยืนยันคำสั่งแล้ว ก็จะเริ่มการทดสอบโดย VX-Propeller จะส่งสัญญาณออกมาผ่านทางพอร์ตอนุกรม ผ่าน IC MAX232 และส่งต่อไปยัง PIC 16F688 ดังแสดงได้รูปที่ 3.17 โดยสัญญาณนี้จะเข้าที่ขา 5 ของ PIC คือ ขา Rx ซึ่งใช้ในการรับข้อมูลจากพอร์ตอนุกรมของ VX-Propeller

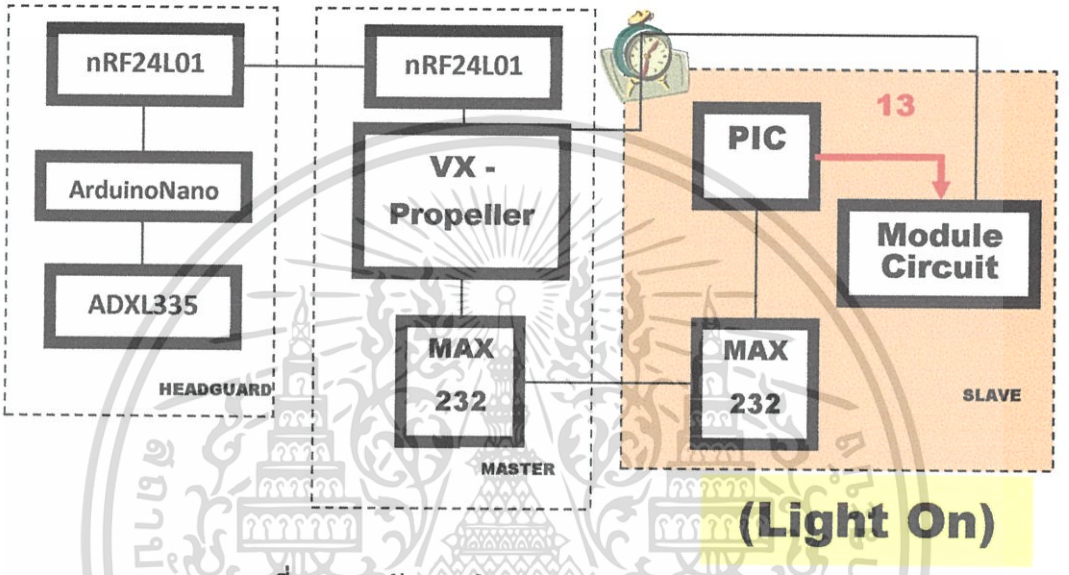


รูปที่ 3.19 หลักการทำงานของระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- เมื่อสัญญาณถูกส่งเข้ามาที่ขา 5 ของ PIC จะเป็นการกระตุ้นให้เกิดการทำงานของอินเตอร์รัปต์จากพอร์ตอนุกรม (INT_RDA) ซึ่งเมื่อเกิดการอินเตอร์รัปต์จากพอร์ตอนุกรม จะทำให้ PIC เริ่มการทำงานของ TIMER0 (เริ่มจับเวลา)

- เมื่อ TIMER 0 เริ่มทำงาน TIMER 0 จะนับจำนวนของโอเวอร์โฟลว์จาก TIMER0 ไปเรื่อยๆ ในขณะเดียวกัน PIC ก็จะส่งสัญญาณพัลส์ออกทางขา 13 (ขา RA0) ส่งไปยังวงจรของโมดูลปุ่มกด ซึ่งจะทำให้ไฟที่ปุ่มกดติด

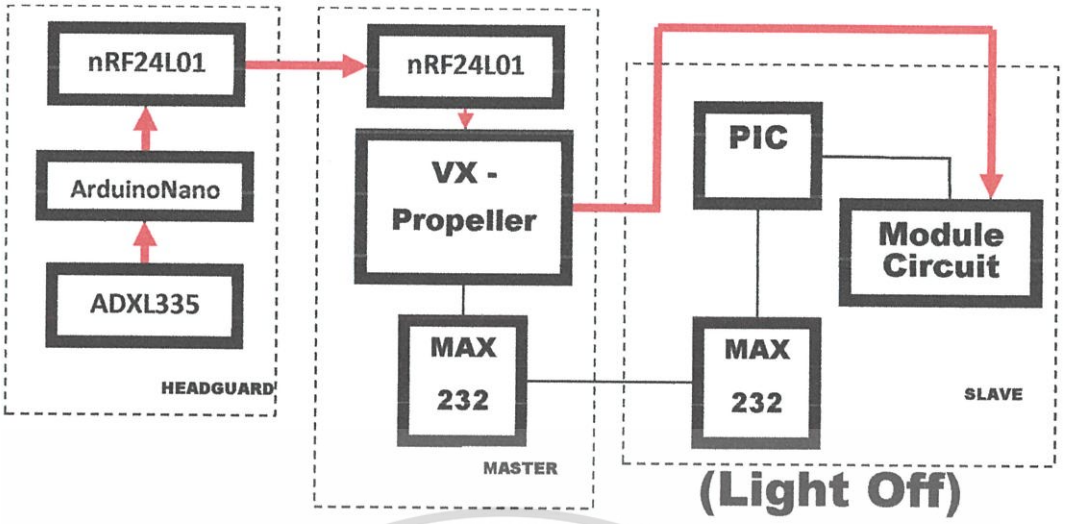


รูปที่ 3.20 หลักการทำงานของระบบ (ต่อ)

- เมื่อมีการกดปุ่มสวิตช์จะทำให้ไฟดับลง พร้อมกับเกิดการเปลี่ยนแปลงสัญญาณจาก High เป็น Low และส่งสัญญาณนี้ไปเข้าที่ขา 11 ของ PIC คือ ขา INT ซึ่งใช้ในการตรวจจับการเกิดอินเตอร์รัปต์จากภายนอก (INT_EXT)

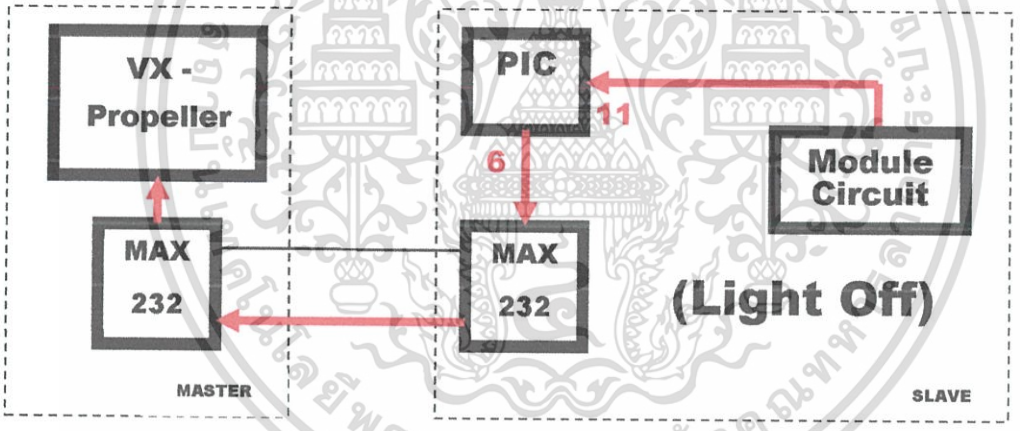
- ในกรณีของโหมดป้องกัน Vx-Propeller จะรอรับตัวอักษรที่แสดงทิศทางการโยกแล้วตรวจสอบว่าถูกต้องหรือไม่ ถ้าตัวอักษรที่ได้รับถูกต้อง Vx-Propeller จะส่งสัญญาณ pulse ไปปิดไฟเช่นเดียวกับการกดปุ่มสวิทช์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.21 หลักการทำงานของระบบในโหมดป้องกัน

- เมื่อเกิดการอินเตอร์รัปต์จากภายนอก จะทำให้ PIC หยุดการทำงานของ TIMER0 (หยุดจับเวลา) และส่งค่าโอเวอร์โฟลว์ที่แปลงค่าแล้วออกไปทางขา 6 คือ ขา Tx ที่ใช้ในการส่งข้อมูลออกไปยังพอร์ตอนุกรมของ VX-Propeller

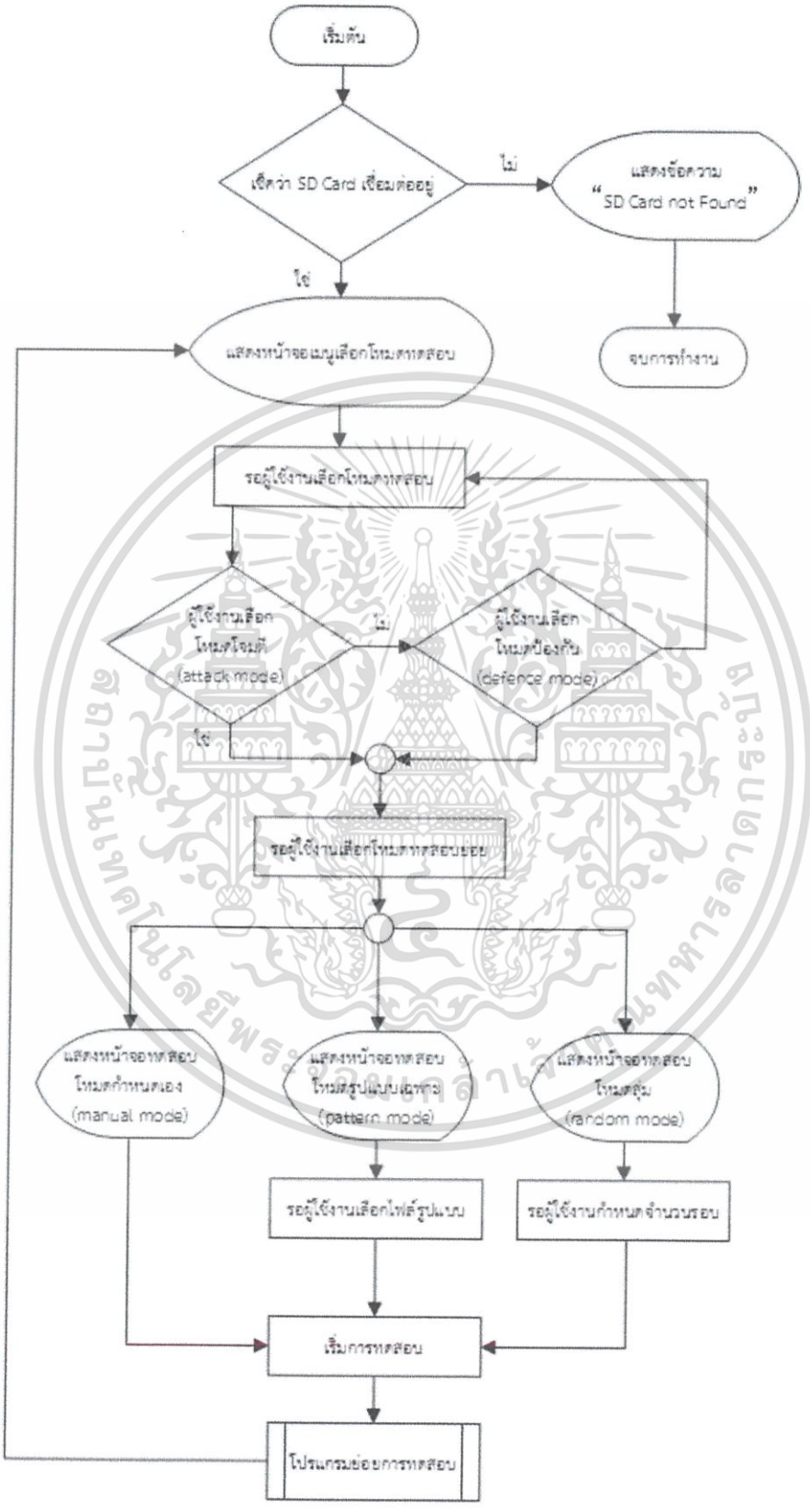


รูปที่ 3.22 หลักการทำงานของระบบ (ต่อ)

- VX-Propeller จะทำการแปลงค่าที่ได้มาให้เป็นค่าเวลา Reaction time พร้อมบันทึกค่าที่ได้ลงบน SD CARD นั่นคือเสร็จสิ้นการทดสอบ 1 รอบ
- หลังจากนั้น VX-Propeller จะส่งสัญญาณออกไปยังปุ่มกดผ่านทางพอร์ตอนุกรมอีกครั้ง จนกว่าจะครบจำนวนรอบที่กำหนดไว้ เมื่อครบจำนวนรอบที่กำหนดไว้แล้ว การทดสอบจะสิ้นสุดลง VX-Propeller จะทำการแสดงค่า Reaction Time ทั้งหมดออกทางจอภาพ VGA
- ในระหว่างการทดสอบ ผู้ใช้งานสามารถยกเลิกการทดสอบกลางคันได้ โดยหากผู้ใช้งานไม่ทำการกดปุ่มภายในเวลา 5 วินาที ระบบจะถามผู้ใช้งานว่าต้องการทำการทดสอบต่อหรือไม่ หากต้องการ ให้กดปุ่มพร้อมทำการทดสอบรอบต่อไป แต่ถ้าไม่ต้องการ ระบบจะหยุดการทดสอบ พร้อมทั้งแสดงค่า Reaction Time ออกทางจอภาพ VGA

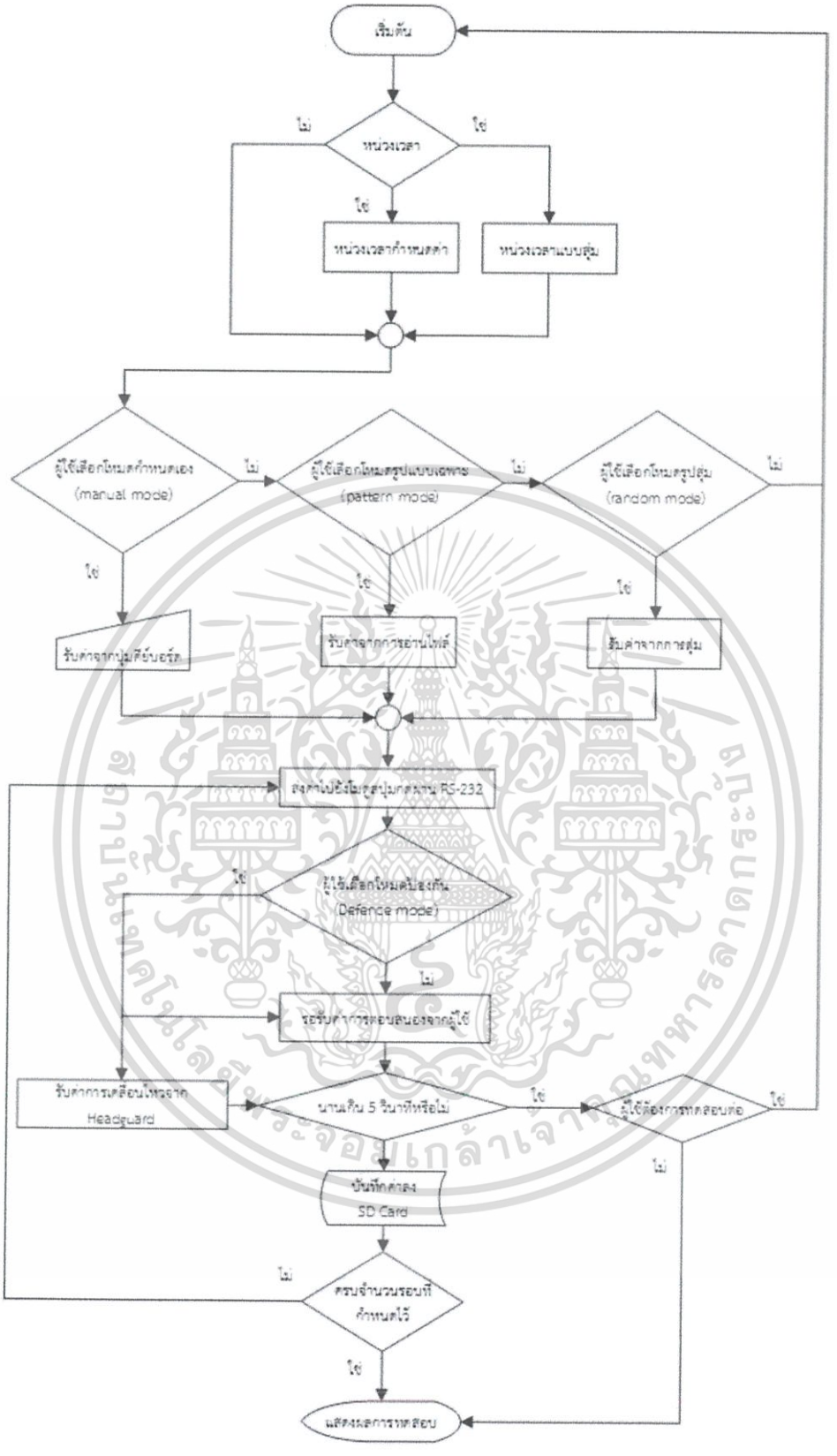
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากหลักการการทำงานข้างต้น สามารถนำมาเขียนเป็นโฟลว์ชาร์ตของการทำงานได้ดังรูปที่ 3.23, 3.24 และ 3.25



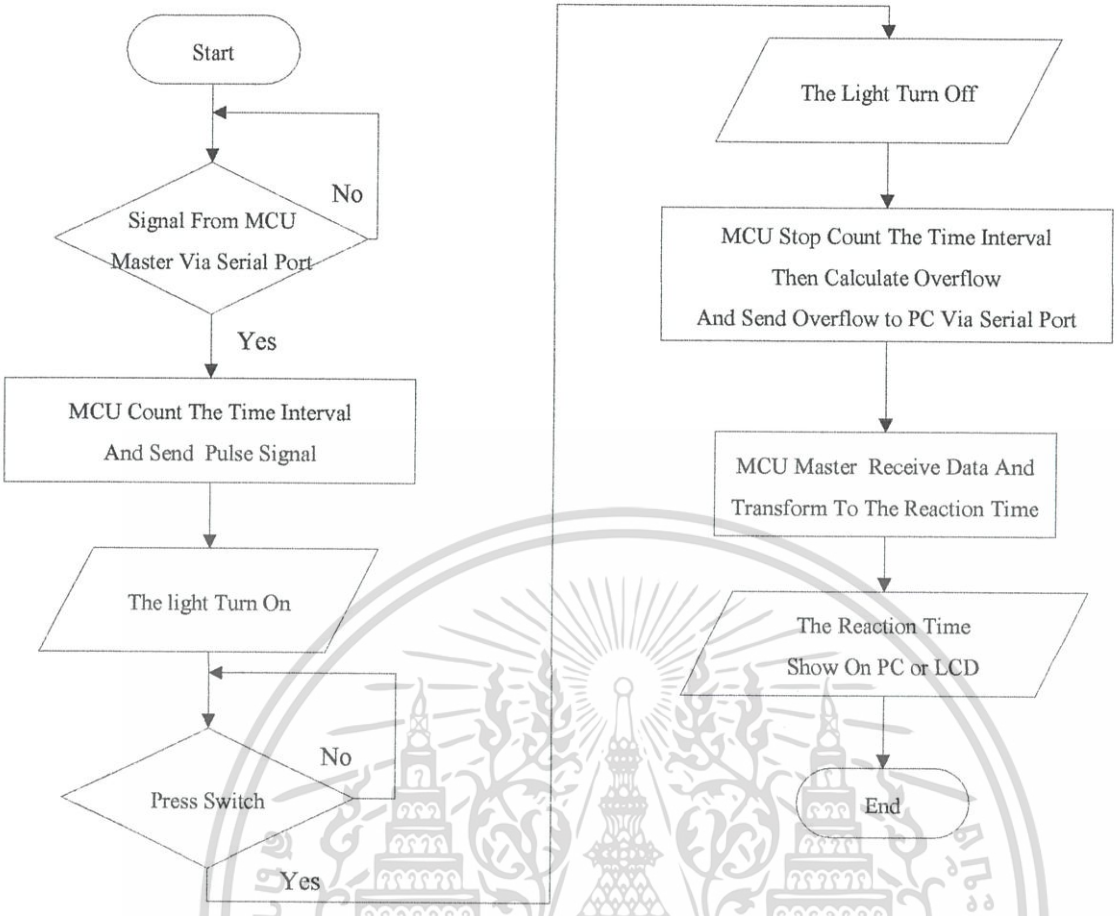
รูปที่ 3.23 โฟลว์ชาร์ตการทำงานของ Vx-Propeller

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.24 โฟลว์ชาร์ตการทำงานของโปรแกรมย่อย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.25 โฟลว์ชาร์ตการทำงานของไมโครคอนโทรลเลอร์ของโมดูลปุ่มกด (Slave 16F688)

3.4.2 การคำนวณการนับโอเวอร์โฟลว์โดยใช้ TIMER 0 ของไมโครคอนโทรลเลอร์ PIC

- TIMER0 นับได้สูงสุด 256 ครั้ง (0-255)
- ใช้ปรีสเกลเลอร์ = 1
- ใช้ External Oscillator = 20 MHz

จะได้ คาบเวลาในการนับ 1 ครั้ง (T) = $\frac{1}{\left(\frac{f}{4\times 1}\right)}$

$$T = \frac{1}{\left(\frac{20M}{4\times 1}\right)}$$

$$= 0.2\ \mu s$$

เพราะฉะนั้นการเกิดโอเวอร์โฟลว์ 1 ครั้ง จะใช้เวลา = 256 × 0.2 µs

= 51.2 µs

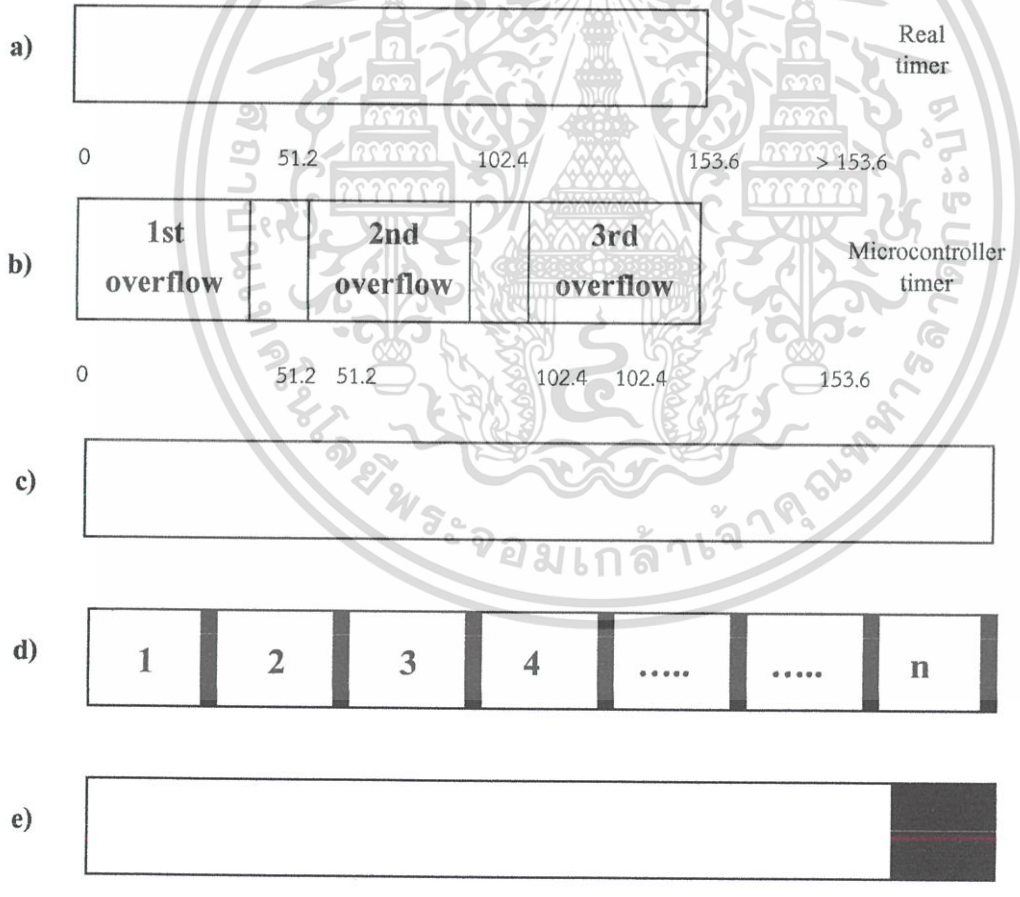
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะได้	เวลา	51.2 µs	เกิดโอเวอร์โฟลว์ 1	ครั้ง
เพราะฉะนั้น	เวลา	1 ms	เกิดโอเวอร์โฟลว์ 19.5	ครั้ง

ดังนั้นเมื่อไมโครคอนโทรลเลอร์นับค่าโอเวอร์โฟลว์ได้ทุกๆ 19 ครั้ง ก็จะเพิ่มค่าตัวแปรทีละหนึ่งไปเรื่อยๆ แล้วส่งค่าเวลาที่ได้ออกมาให้กับ VX-Propeller แล้วแสดงผลทางจอภาพ VGA เช่น นับค่าโอเวอร์โฟลว์ได้ 95,000 ครั้ง นำมาหารด้วย 19 จะได้ 95000/19 = 5000

3.4.3 การชดเชยค่าเวลา

ในการนับค่าโอเวอร์โฟลว์ได้ครบในแต่ละครั้ง ไมโครคอนโทรลเลอร์จะมีการเสียเวลาส่วนหนึ่งไปกับการเช็คเงื่อนไข การเพิ่มค่าตัวแปร การเซ็ตค่าพารามิเตอร์ต่างๆ ทำให้การจับเวลานั้นไม่เป็นไปอย่างต่อเนื่อง กล่าวคือ จะมีการเสียเวลาส่วนหนึ่งไปก่อนที่จะเริ่มนับค่าโอเวอร์โฟลว์ใหม่ ทำให้ค่าเวลาที่อ่านได้นั้น มีค่าน้อยกว่า ค่าเวลาที่จับได้จริง ซึ่งสามารถแสดงการเปรียบเทียบได้ดังรูปที่ 3.25 a) และ 3.25 b) เราจึงทำการปรับปรุงแก้ไขโดยการบวกเวลาชดเชยเพิ่มเข้าไปให้กับค่าเวลาที่อ่านได้ ก่อนที่จะส่งค่าเวลาออกไปยัง VX-Propeller ซึ่งจะต้องทราบก่อนว่าค่าเวลาที่สูญเสียไปทั้งหมดนั้นมีค่าเท่าใด จึงจะทราบได้ว่าค่าเวลาชดเชยที่จะนำมาบวกเพิ่มเข้าไปนั้นมีค่าเท่าใด



รูปที่ 3.25 a) – b) แสดงความผิดพลาดในการนับโอเวอร์โฟลว์ของไมโครคอนโทรลเลอร์
รูปที่ 3.25 c) – e) แสดงความผิดพลาดทั้งหมดที่เกิดขึ้นในการจับเวลาของไมโครคอนโทรลเลอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ส่วนในรูปที่ 3.25 c) และ 3.25 d) นั้นเป็นการเปรียบเทียบกันระหว่างค่าเวลาที่จับได้จริงกับค่าเวลาที่ไมโครคอนโทรลเลอร์อ่านได้ตามลำดับ ซึ่งจะเห็นได้ว่าพื้นที่สีขาวในรูปที่ 3.25 d) ทั้งหมดรวมกันนั้นมีค่าน้อยกว่าพื้นที่สีขาว ในรูปที่ 3.25 c)

ขั้นตอนการหาค่าเวลาสูญเสีย

1. เปิดโค้ดโปรแกรมแอสแซมบลีในส่วนของอินเตอร์รัปต์ไทม์เมอร์ 0
2. นับจำนวน instruction cycles ที่ใช้ในการทำคำสั่งแต่ละคำสั่ง (ในที่นี้คือคำสั่งในบรรทัด a) , b) , c) และ d))

```
#INT_TIMER0                                  //Function Interrupt TIMER0
void Timer0_ISR()
{
        if(int_count++>19)                                    ---------- a)
        {
                ++i;                                          ---------- b)
                int_count =0;                                 ---------- c)
         }
        if(i>5000)                                            ---------- d)
        {
                fprintf(PC,"xxxx");                           ---------- e)
                disable_interrupts(INT_TIMER0);               ---------- f)
                n = 1;                                        ---------- g)
        }
}
```

อธิบายการทำงานของโปรแกรมได้ดังนี้ (เฉพาะส่วนของอินเตอร์รัปต์ไทม์เมอร์ 0)

- int_count คือตัวแปรที่ใช้ในการนับโอเวอร์โฟลว์ โดยเริ่มต้นกำหนดให้ int_count = 0 (เมื่อ int_count มีค่าเกิน 19 หรือนับโอเวอร์โฟลว์ได้ 19 ครั้ง จะได้เวลา 1 ms)
- i คือตัวแปรที่ใช้ในการนับเวลา (ms) กล่าวคือตัวแปร i จะเพิ่มขึ้น 1 ทุกๆ เวลา 1 ms โดยเริ่มต้นจะกำหนดให้ i = 0
- ใช้ External Oscillator = 20 MHz ดังนั้นจะได้ 1 instruction cycle = = 200 ns
- คำสั่งในบรรทัด a) ใช้เวลาในการทำคำสั่ง 6 instruction cycles = 6 x 200 = 1200 ns
- คำสั่งในบรรทัด b) ใช้เวลาในการทำคำสั่ง 3 instruction cycles = 3 x 200 = 600 ns
- คำสั่งในบรรทัด c) ใช้เวลาในการทำคำสั่ง 1 instruction cycle = 1 x 200 = 200 ns
- คำสั่งในบรรทัด d) ใช้เวลาในการทำคำสั่ง 14 instruction cycles = 14 x 200 = 2800 ns
- เมื่อไมโครคอนโทรลเลอร์นับโอเวอร์โฟลว์ครบในแต่ละครั้ง ไมโครคอนโทรลเลอร์จะถูกอินเตอร์รัปต์โดยไทม์เมอร์ 0 โปรแกรมจะข้ามกระโดดเข้ามาทำงานในส่วนของ #INT_TIMER0

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- เมื่อ int_count มีค่าไม่เกิน 19 (ไมโครคอนโทรลเลอร์ยังนับโอเวอร์โฟลว์ไม่ครบ 19 ครั้งหรือยังไม่ครบ 1 ms) โปรแกรมจะทำคำสั่งเฉพาะบรรทัด a) และ บรรทัด d) เท่านั้น ซึ่งจะสูญเสียเวลาไปกับการทำคำสั่งทั้งหมด = 1200 + 2800 = 4000 ns

- เมื่อ int_count มีค่าเกิน 19 (ไมโครคอนโทรลเลอร์นับโอเวอร์โฟลว์ครบ 19 ครั้งหรือครบเวลา 1 ms) โปรแกรมจะทำคำสั่งตั้งแต่บรรทัด a) , b) , c) จนถึงบรรทัด d) ซึ่งจะสูญเสียเวลาไปกับการทำคำสั่งทั้งหมด = 1200 + 600 + 200 + 2800 = 4800 ns

- ดังนั้นในเวลา 1 ms ไมโครคอนโทรลเลอร์จะสูญเสียเวลาไปกับการทำคำสั่งต่างๆทั้งหมด
= (4000 x 18) + 4800 = 76800 ns = 0.0768 ms

- กล่าวคือ หากค่าเวลาที่จับได้จริงคือ 1 ms แล้ว ค่าเวลาที่อ่านได้ จะมีค่าเท่ากับ = 1 – 0.0768 = 0.9232 ms ซึ่งจะเห็นได้ว่าค่าเวลาที่อ่านได้นั้นมีค่าน้อยกว่าค่าเวลาที่จับได้จริง

- ในทางกลับกัน หากค่าเวลาที่อ่านได้นั้นเป็น 0.9232 ms ค่าเวลาที่จับได้จริงจะมีค่าเท่ากับ = 0.9232 + 0.0768 = 1 ms นั่นเอง

- สามารถเขียนเป็นสูตรคำนวณได้ดังนี้

$$W = (0.0768\ i) + i$$
$$= 1.0768\ i$$

โดยที่ i คือ ค่าเวลาที่อ่านได้ก่อนทำการปรับปรุง (ms)

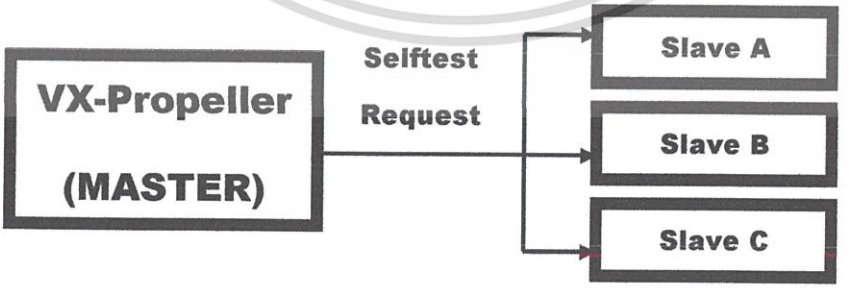
W คือ ค่าเวลาที่อ่านได้หลังจากทำการปรับปรุงแล้ว ซึ่งมีค่าใกล้เคียง หรือมีค่าเท่ากับค่าเวลาที่จับได้จริง (ms)

ดังนั้นเมื่อการจับเวลาสิ้นสุดลง ค่า i ที่ได้ จะต้องถูกนำมาคำนวณอีกครั้งหนึ่ง โดยการใช้สูตรคำนวณข้างต้น เพื่อให้ได้ค่า W ซึ่งเป็นค่าเวลาใหม่ที่มีความแม่นยำมากยิ่งขึ้น ก่อนที่จะส่งค่าเวลาใหม่นี้ออกไปยังไมโครคอนโทรลเลอร์ VX-Propeller พร้อมแสดงค่าเวลาออกทางจอภาพ VGA

3.4.4 หลักการทำงานของระบบการตรวจสอบตนเอง

สามารถอธิบายหลักการทำงานได้ดังนี้

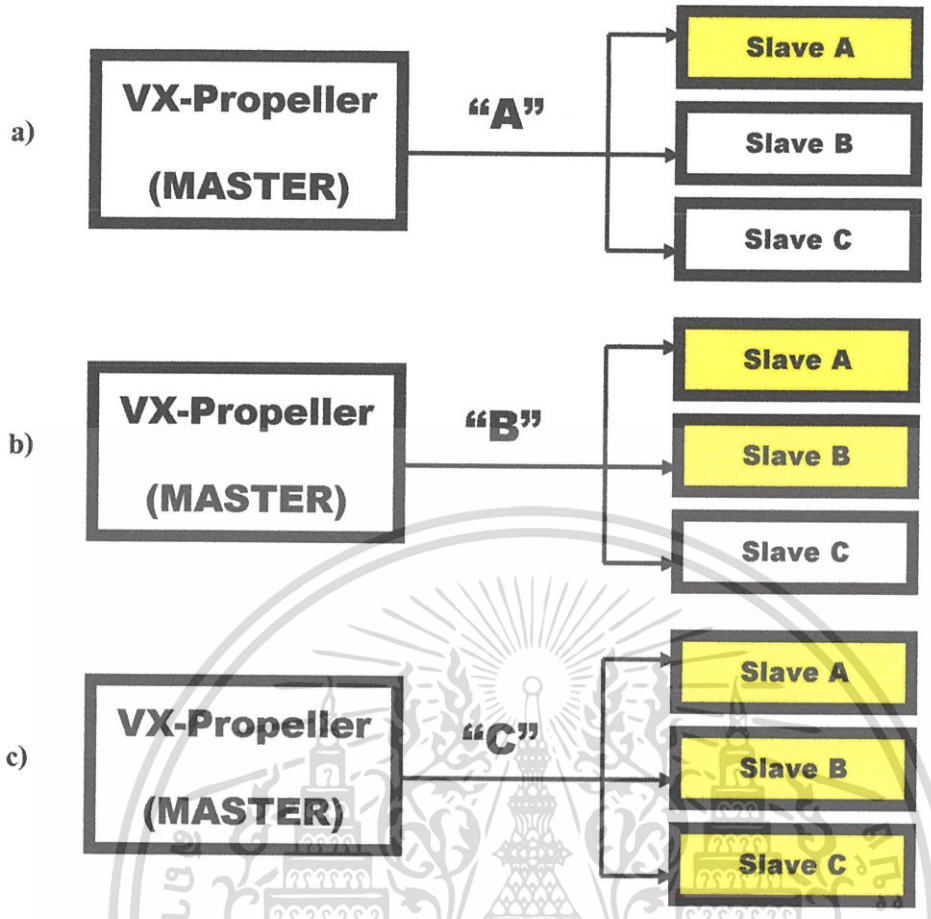
- เริ่มแรก VX-Propeller จะส่งสัญญาณร้องขอไปยังปุ่มกดทุกปุ่ม เพื่อให้ปุ่มกดทุกปุ่มนั้นเตรียมความพร้อมสำหรับการตรวจสอบตนเอง



รูปที่ 3.26 หลักการทำงานของระบบตรวจสอบตนเอง

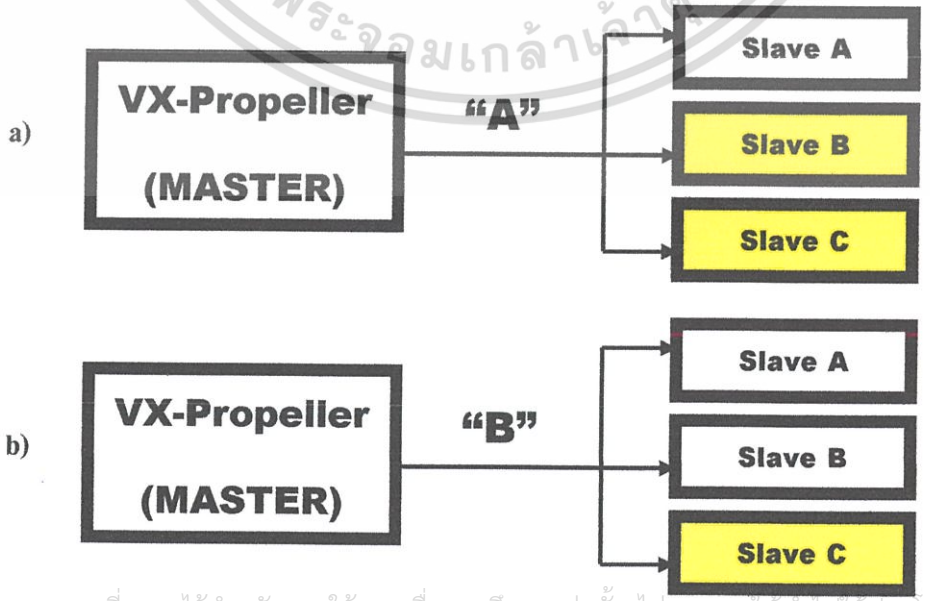
- หลังจากนั้น VX-Propeller จะส่งสัญญาณไปยังปุ่มกด เพื่อให้ไฟติดทีละปุ่ม โดยเริ่มจากปุ่มกด Slave A , Slave B , Slave C ตามลำดับ ดังแสดงได้ดังรูปที่ 3.27

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

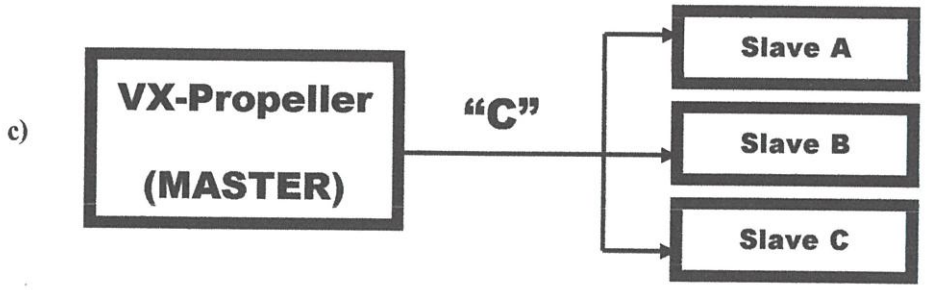


รูปที่ 3.27 หลักการทำงานของระบบตรวจสอบตนเอง (ต่อ)

- หลังจากนั้น VX-Propeller จะส่งสัญญาณไปยังปุ่มกดอีกครั้ง เพื่อให้ไฟดับทีละปุ่ม โดยเริ่มจากปุ่มกด Slave A , Slave B , Slave C ตามลำดับ ดังแสดงได้ดังรูปที่ 3.28 เมื่อไฟบนปุ่มกดทุกปุ่มได้ดับลง ระบบจะแจ้งให้ผู้ใช้งานทราบว่าการตรวจสอบตนเองนั้นได้สิ้นสุดลงแล้ว

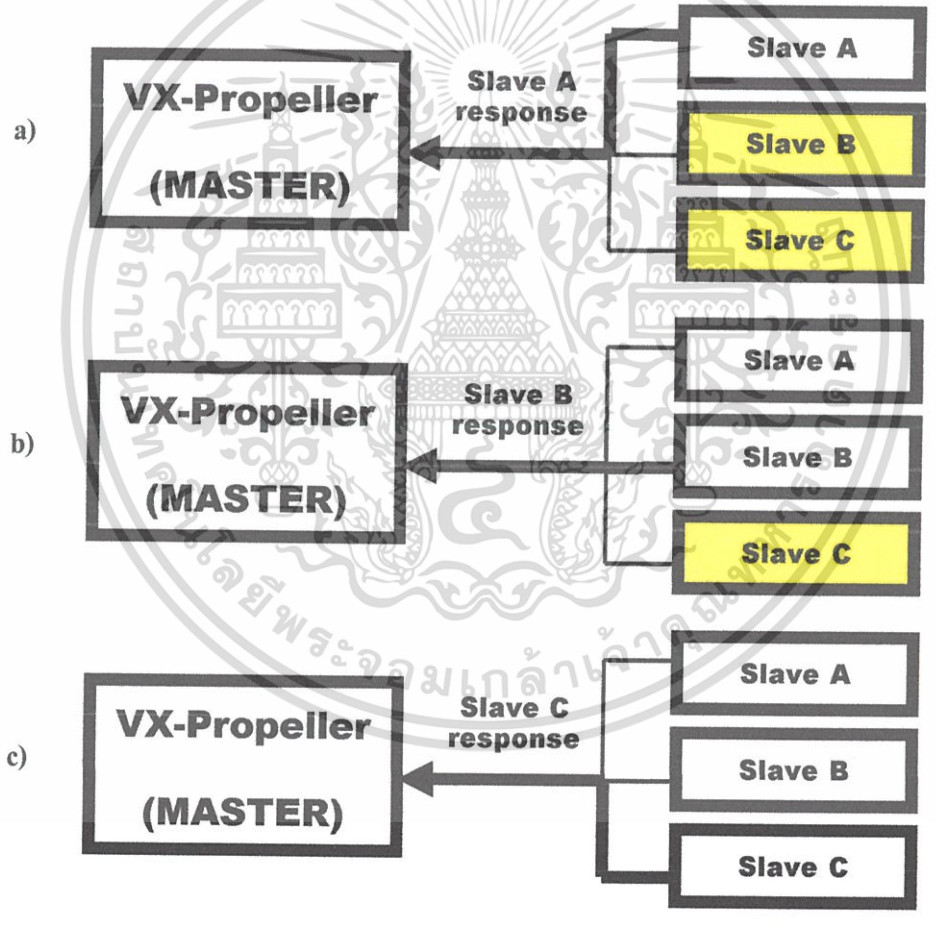


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.28 หลักการทำงานของระบบตรวจสอบตนเอง (ต่อ)

- เมื่อปุ่มกดปุ่มใดไฟดับแล้ว ปุ่มกดจะส่งสัญญาณตอบกลับไปยัง VX-Propeller เพื่อให้ทราบว่าปุ่มกดนั้นยังสามารถทำงานได้ตามปกติ ดังแสดงได้ดังรูปที่ 3.29



รูปที่ 3.29 หลักการทำงานของระบบตรวจสอบตนเอง (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4.5 การคำนวณค่า G

โหมดป้องกันตรวจจับการโยกศีรษะออกมาในรูปของค่าความเร่งจาก ADXL335 ซึ่งต้องทำการแปลงค่าความเร่งที่อ่านได้ให้อยู่ในรูปค่า G เสียก่อนด้วยโปรแกรมภายใน Arduino Nano ค่า G ที่ตำแหน่งต่างๆ จะเป็นดังรูปที่ 3.30 จากข้อมูลใน Datasheet ของ ADXL335 เมื่อใช้งานที่แรงดัน 3.3 V แรงดันที่อ่านได้เมื่อความเร่งเป็น 0 คือ 1.65 V แรงดันจะมีการเปลี่ยนแปลง 330 mV/G และสัญญาณที่ได้จาก Analog to Digital Convertor จะมีค่าอยู่ระหว่าง 0 ADC ถึง 1023 ADC

จะได้

$$\frac{330\ \text{mV}}{G} = \frac{330\ \text{mV} \times (1023\ \text{ADC units})}{G \times 3.3\ V} \quad \text{---------- (2)}$$

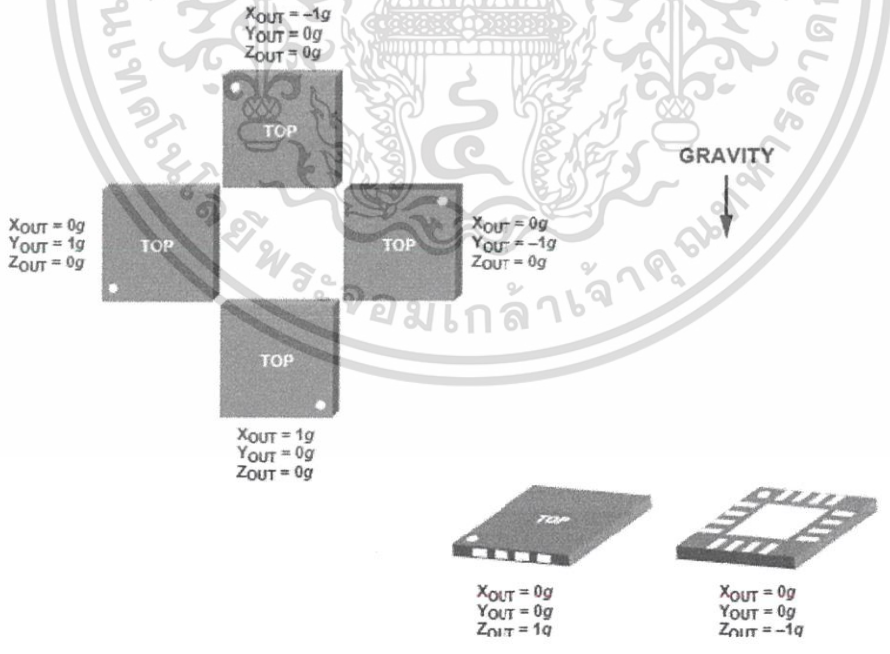
$$= 102.3\ (\text{ADC units})/G$$

ดังนั้นจึงสามารถใช้ความสัมพันธ์นี้แปลงค่าความเร่งที่อ่านได้จาก ADXL335 ในหน่วย ADC เป็นค่าความเร่งในหน่วย G จากสูตร

$$\text{ความเร่ง (G)} = \frac{\text{ความเร่ง (ADC)} - \text{Zero G}}{Scale} \quad \text{---------- (3)}$$

โดยที่ Zero G คือ ค่าที่อ่านได้เมื่อไม่มีความเร่ง มีค่าเท่ากับ 512 ADC

Scale คือ ปริมาณการเปลี่ยนของหน่วย ADC ต่อ 1G มีค่าเท่ากับ 102.3



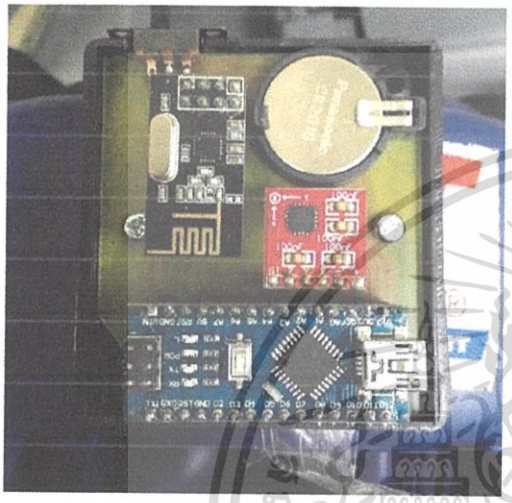
รูปที่ 3.30 ค่า G ในแนวแกน x y และ z ของ ADXL335

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
การทดลอง และผลการทดลอง

เครื่องทดสอบการหลบหมัดได้มีการปรับปรุงและพัฒนาในหลายๆส่วน เพื่อให้มีประสิทธิภาพในการทดสอบมากขึ้น ผลการทดสอบของส่วนที่แก้ไขและเพิ่มเติมมีดังนี้

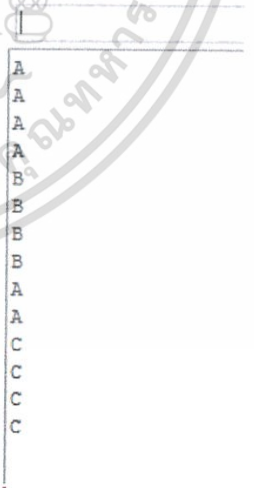
4.1 การส่งสัญญาณแบบไร้สาย



รูปที่ 4.1 วงจรตัวส่งสัญญาณไร้สายส่วนภาคส่ง

รูปที่ 4.2 วงจรตัวส่งสัญญาณไร้สายส่วนภาครับ

รูปที่ 4.3 serial monitor ของภาคส่งจากโปรแกรม Arduino

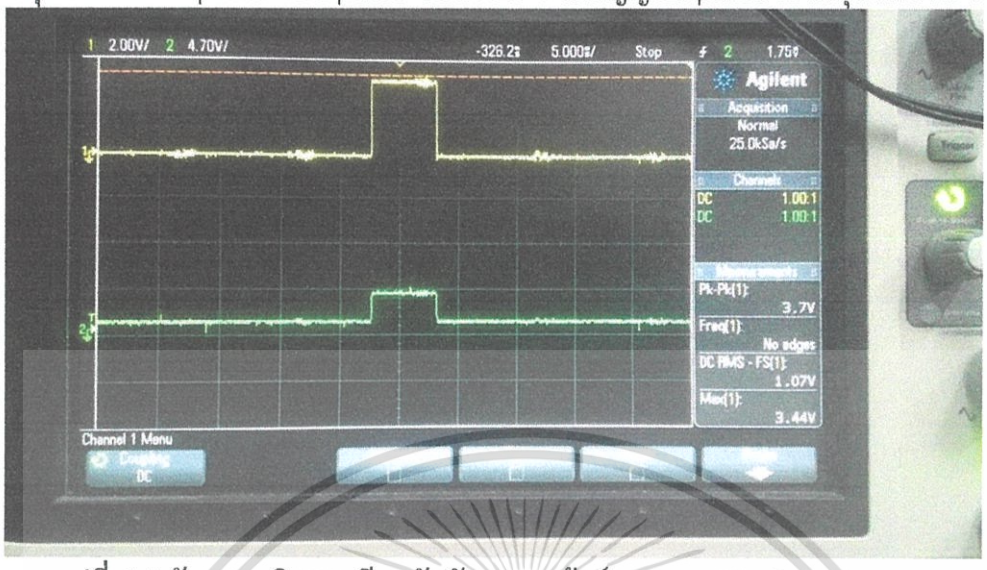


รูปที่ 4.4 serial monitor ของภาครับจากโปรแกรม Arduino

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2 วงจร Comparator

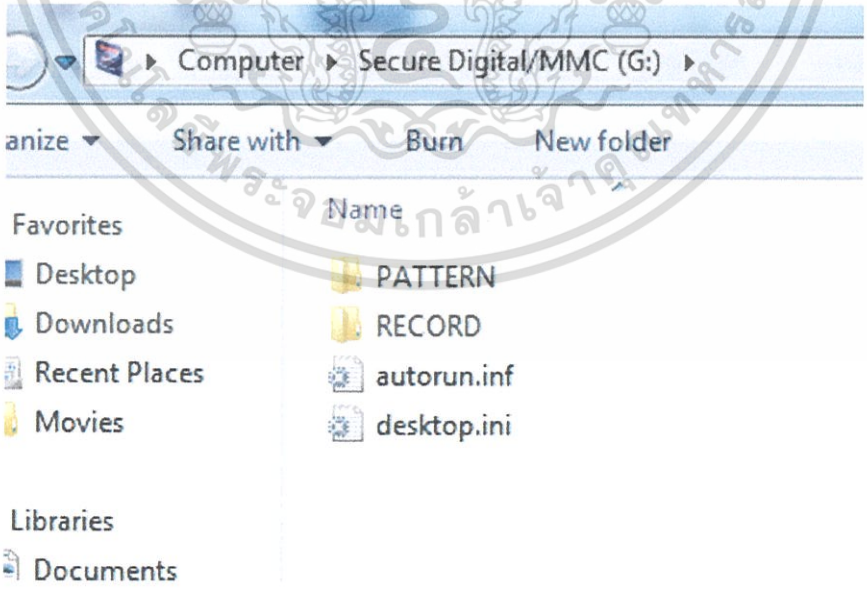
อินพุทจาก Vx-Propeller เป็น pulse ขนาด 3.3 V ได้สัญญาณ pulse เอ้าท์พุทขนาด 5 V



รูปที่ 4.5 สัญญาณอินพุทเทียบกับสัญญาณเอ้าท์พุทของวงจร Comparator

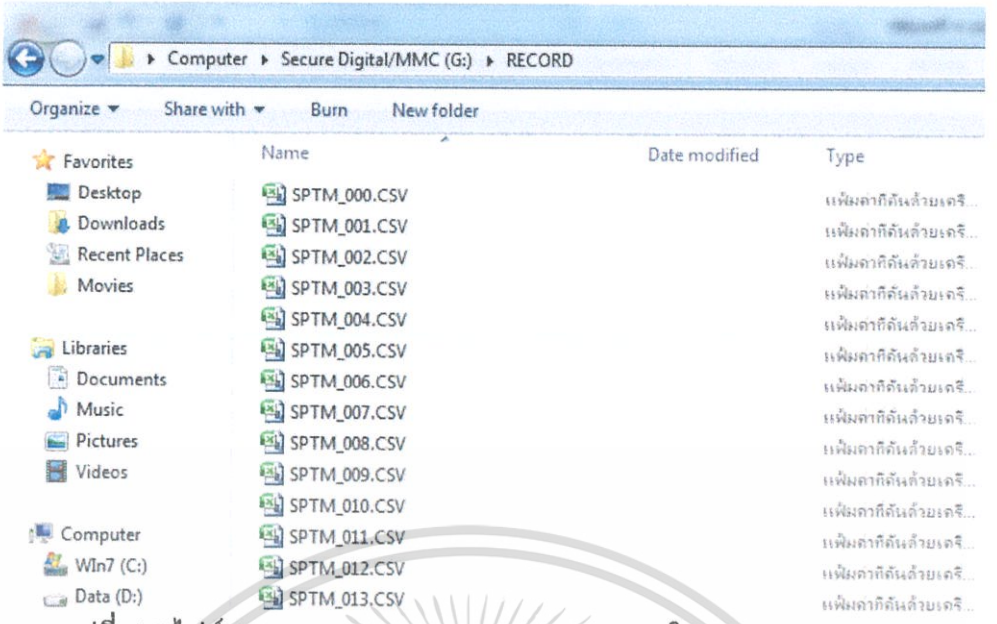
4.3 การบันทึกผลใน SD Card

การบันทึกผลการทดสอบได้เก็บไว้ในโฟลเดอร์ที่ชื่อว่า RECORD โดยรูปแบบไฟล์ที่ใช้ในการบันทึกข้อมูลเป็นสกุลไฟล์ .csv ซึ่งเป็นสกุลไฟล์ที่ใช้งานกับ Microsoft Excel ทำให้สามารถนำไปประยุกต์ใช้ในงานด้านอื่นๆต่อไปได้ โดยได้กำหนดให้ไฟล์ดังกล่าวมีชื่อไฟล์เป็น SPTM_xxxx.CSV (xxxx คือ ตัวเลขตั้งแต่ 0001 ถึง 9999) รูปแบบการบันทึกจะปรับปรุงจากโครงงานเดิมเพื่อให้สะดวกในการนำไปวิเคราะห์มากขึ้น การบันทึกข้อมูลนั้นจะบันทึกทั้งข้อมูลที่เป็นค่าเวลา Reaction time รวมไปถึงข้อมูลของผู้ทำการทดสอบด้วย



รูปที่ 4.6 Folder ภายใน SD Card

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.7 ไฟล์ SPTM_xxxx.CSV ผลการทดสอบภายใน Folder RECORD

L7

	A	B	C	D	E	F	G	H
1	Name	Gender	Age	ATK/DEF	Mode	Round	Button	Time
2	Bewty	Female	22	Attack	Manual		1 A	1053
3	Bewty	Female	22	Attack	Manual		2 A	
4	Bewty	Female	22	Attack	Manual		3 A	1343
5	Bewty	Female	22	Attack	Manual		4 D	1336
6	Bewty	Female	22	Attack	Manual		5 A	483
7								

รูปที่ 4.8 การบันทึกผลการทดสอบลงในไฟล์ SPTM_xxxx.CSV

4.4 ความแม่นยำในการจับเวลา

เนื่องจากมีการเพิ่มโหมดป้องกันเข้าไปในระบบอาจทำให้มีผลต่อการจับเวลาของโครงงานเดิม จึงมีการทดสอบเพื่อวัดความแม่นยำในการจับเวลาของโหมดป้องกันเทียบกับโหมดโจมตี การทดสอบเพื่อวัดความแม่นยำในการจับเวลามีขั้นตอนดังนี้

1. เลือกขาใดขาหนึ่งของ PIC16F688 ที่ยังไม่ได้ถูกใช้งาน เพื่อใช้สำหรับเป็นขาในการทดสอบ

2. เมื่อเริ่มทำการจับเวลา ให้เซ็ตลอจิกเป็นลอจิก 1 (5V) ที่ขาทดสอบ ของ PIC (ทำได้โดยวิธีการเขียนโปรแกรม)

3. เมื่อหยุดจับเวลา ให้เซ็ตลอจิกเป็นลอจิก 0 (0V) ที่ขาทดสอบ ของ PIC (ทำได้โดยวิธีการเขียนโปรแกรม)

4. ทำการทดสอบระบบโดยใช้งานโหมดการทดสอบ Manual Modeและ Defense Mode

5. วัดสัญญาณที่ขาทดสอบ ของ PIC โดยการใช้ออสซิลโลสโคป จะได้สัญญาณ Pulse ออกมาหนึ่งลูก ค่าเวลาที่จับได้จริงจะมีค่าเท่ากับค่าความกว้างของสัญญาณ Pulse นั้น

จากการทดสอบพบว่าค่าความคลาดเคลื่อนที่ได้จากทั้งสองโหมดมีค่าใกล้เคียงกัน ดังแสดงในตารางที่ 4.1 และ 4.2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ค่าเวลาที่อ่านได้จากความกว้างของสัญญาณ Pulse จาก Oscilloscope (ms)	ค่าเวลาที่อ่านได้จาก จอภาพ VGA (ms)	เปอร์เซ็นต์ความคลาดเคลื่อน (%)
2,879	2883	0.12
532.9	533	0.01
643.6	643	0.09
601.8	603	0.19
1,536	1537	0.05
743.8	745	0.16
2,005.50	2008	0.12
1,259.10	1260	0.07
2,533	2536	0.11
1,061.00	1062	0.09

ตารางที่ 4.1 ความคลาดเคลื่อนโหมดโจมตี

ค่าเวลาที่อ่านได้จากความกว้างของสัญญาณ Pulse จาก Oscilloscope (ms)	ค่าเวลาที่อ่านได้จาก จอภาพ VGA (ms)	เปอร์เซ็นต์ความคลาดเคลื่อน (%)
363.4	363	0.11
2,222.10	2,222	0.00
1,095.60	1096	0.03
1,177.10	1179	0.16
1,312.20	1314	0.13
789.90	790	0.01
609.90	610	0.01
2,057.90	2060	0.10
1,181.30	1183	0.14
715.20	716	0.11

ตารางที่ 4.2 ความคลาดเคลื่อนโหมดป้องกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า นำไปใช้

บทที่ 5
บทสรุป

ในปริญญานิพนธ์นี้ได้กล่าวถึงการพัฒนาเครื่องฝึกการหลบหมัดซึ่งช่วยในการทดสอบสมรรถภาพแก่นักกีฬาหรือผู้ที่ต้องการพัฒนาการตอบสนองของร่างกาย ได้มีการปรับปรุงแก้ไขจากต้นแบบการพัฒนาระบบวัดเวลาการตอบสนองระหว่างตากับมือ โดยได้มีการพัฒนาระบบเพิ่มเติมในส่วนของการป้องกัน (Defence Mode) โดยการใช้เซนเซอร์วัดความเร่ง ADXL335 (Accelerometer) ร่วมกับบอร์ด Arduino และโมดูล Wireless nRF24L01 ประกอบกันเป็นวงจรติดกับหมวกป้องกัน(Headguard) มาใช้ในการทดสอบกับระบบเพื่อให้สะดวกต่อการใช้และเพิ่มความคล่องตัวให้ผู้ทดสอบ พร้อมเมนูการเลือกใช้งานที่เปลี่ยนแปลงใหม่จากเดิม วิธีการบันทึกข้อมูลลง SD CARD ในรูปแบบของไฟล์ .csv ซึ่งเป็นสกุลไฟล์ที่ใช้งานกับ Microsoft Excel ทำให้สามารถนำข้อมูลเหล่านี้ไปประยุกต์ใช้งานในด้านอื่นๆต่อไปได้ เช่น การเปรียบเทียบสมรรถภาพของนักกีฬา

โครงงานนี้สามารถพัฒนาต่อไปได้อีกในอนาคต โดยเพิ่มเติมการวัดความแรงของหมัดด้วยเซนเซอร์ซึ่งมีข้อดีให้สามารถพัฒนาขีดความสามารถของโครงงานนี้ให้สูงขึ้นไปอีก ทั้งนี้เพื่อจะทำให้โครงงานมีมาตรฐานในการทดสอบและฝึกปฏิกิริยาการตอบสนองของนักกีฬาที่ดียิ่งขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

[1] กัญญาณัฐ ภควัตมงคลขวัญชนก ดาวังปาวีรุตม์ แพทย์สุวรรณและ วุฒิศักดิ์กำมะหยี่ ระบบวัดเวลาการตอบสนองระหว่างตากับมือโดยไมโครคอมพิวเตอร์ ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง กรุงเทพฯ 2553

[2] ชัชชัย คาง การพัฒนาระบบวัดเวลาการตอบสนองระหว่างตากับมือปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง กรุงเทพฯ2554

[3] กฤษฎา ใจเย็น / ชัยวัฒน์ลิ้มพรจิตรวิไล , “ก้าวแรกกับ Propeller มัลติคอร์ไมโครคอนโทรล-เลอร์” INEX กรุงเทพฯ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก
โค้ดโปรแกรม

1. โค้ดโปรแกรมในส่วนของ PIC

```
#include <16F688.h>                                          //Device
#fuses HS ,NOWDT,NOPROTECT                                   //Option
#use delay(clock=20000000)                                   //set clock
#use rs232(baud=9600, xmit=PIN_C3, rcv=PIN_C2 ,stream=PC)    //set baud rate,PIN
#PRIORITY EXT,TIMER0                                         //ser priority interrupt
char A;                                                      //set parameter
char B;
int16 i=0;
int j=0;
int k=0;
int n=0;
int int_count = 0;
int interrupt = 0;
float f = 0.0768;
int16 w;
void recievech()                          //Function recieve character
{
  label:
  A = fgetc(PC);                          //wait until a PIN is detected
  delay_ms(50);
  if (A=='c')                             //Detected character 'c'(Address)
  {
    if (j == 1)                           //turn on light for self-testing
    {
      output_high(PIN_A0);
      delay_ms(100);
      output_low (PIN_A0);
      lab:
      A = fgetc(PC);
      delay_ms(50);
      if (A=='c')
      {
        xx:
        output_high (PIN_A1);             //turn off light for selt-testing
        delay_ms(100);
```

เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        output_low (PIN_A1);
        goto label;
      }
      else
      {
        goto lab;
      }
    }
    else if (k == 1)
    {
      //m = 1;
      goto xx;
    }
    else
    {
    i=0;
    k=1;
    output_high(PIN_A0);                    //Trig
    enable_interrupts(INT_TIMER0);          //Turn on Interrupt TIMER0
    setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);    //setup timer
    set_timer0(0);                          //Start timer0
    output_high(PIN_C1);
    delay_ms(1);
    output_low(PIN_A0);
    goto label;
    }
}
else if (A == '+')
{
    A = fgetc(PC);
    delay_ms (50);
    if (A == '+')              //receive selftest request signal
    {
      j = 1;
    }

    goto label;
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
   else
   {
      goto label;
   }
}

void main ()                                    //main function
{
   set_tris_a(0xFC);                            //Set PortA
   enable_interrupts(GLOBAL);                   //All Interrupts on
   enable_interrupts(INT_EXT);
   ext_int_edge(L_TO_H);                        //external interrupt Low to High
   output_low (PIN_A1);
   output_low (PIN_C1);

   while(TRUE)
   {
      recievech();                              //go to order recievech()
      i=0;

   }
}

#INT_TIMER0                                     //Function Interrupt TIMER0
void Timer0_ISR()
{
   if(int_count++>19)
   {
      ++i;
      int_count =0;
   }
   if(i>4643)
   {
   fprintf(PC,"xxxx");
   disable_interrupts(INT_TIMER0);
   n = 1;
   }
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#INT_EXT
void IntExt_isr(void)
{
   output_low (PIN_C1);
   disable_interrupts(INT_TIMER0);              //Turn off Interrupt TIMER0
if ((A=='c') && (i!=0) && (j ==0) && (n == 0))
{
    w = (f * i) + i ;                            //adjust reaction time before printing
    fprintf (PC,"%lu\r\n",w);                    //print reaction time
    //fprintf(PC,"%lu\r\n",i);
}
else if (j ==1)
{
    fprintf (PC,"%c\r\n",'c');                   //print selftest response signal
}
else if (n ==1)
{
    fprintf (PC,"&");
}
      i=0;                                       //reset parameter
      j=0;
      k=0;
      n=0;

}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. โค้ดโปรแกรมในส่วนของภาคส่ง

```
#include <SPI.h>
#include <nRF24L01p.h>

nRF24L01p transmitter(7,8);//CSN,CE
//const int xpin = A0;                  // x-axis of the accelerometer
const int ypin = A1;                  // y-axis
const int zpin = A2;                  // z-axis (only on 3-axis models)

void setup(){
  delay(150);
  Serial.begin(9600);
  SPI.begin();
  SPI.setBitOrder(MSBFIRST);
  transmitter.channel(90); // ตั้งช่องความถี่ให้ตรงกัน
  transmitter.TXaddress("ALL"); // ตั้งชื่อตำแหน่งให้ตรงกัน ชื่อตั้งได้สูงสุด 5 ตัวอักษร
  transmitter.init();
  analogReference(EXTERNAL);
  //pinMode(xpin, INPUT);
  pinMode(ypin, INPUT);
  pinMode(zpin, INPUT);
}

String message;
void loop(){
 //int x = analogRead(xpin);
 int y = analogRead(ypin);
 int z = analogRead(zpin);
 float zero_G = 512.0;
 float scale = 106;
 //float xx = ((float)x - zero_G)/scale;
 float yy = ((float)y - zero_G)/scale;
 float zz = ((float)z - zero_G)/scale;
   if (yy >= 0.4)
   {
     Serial.println("A");
     transmitter.txPL("A");  // ค่าที่ต้องการส่ง
     transmitter.send(FAST); // สั่งให้ส่งออกไป
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    }
    if (yy <= -0.4)
    {
      Serial.println("B");
      transmitter.txPL("B");  // ค่าที่ต้องการส่ง
      transmitter.send(FAST); // สั่งให้ส่งออกไป
    }
    if (zz >= -0.80)
    {
      Serial.println("C");
      transmitter.txPL("C");  // ค่าที่ต้องการส่ง
      transmitter.send(FAST); // สั่งให้ส่งออกไป
    }
  delay(250);
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. โค้ดโปรแกรมในส่วนของภาครับ

```
#include <SPI.h>
#include <nRF24L01p.h>

nRF24L01p receiver(7,8);//CSN,CE

void setup(){
  delay(150);
  Serial.begin(115200);
  SPI.begin();
  SPI.setBitOrder(MSBFIRST);
  receiver.channel(90);  // ตั้งช่องความถี่ให้ตรงกัน
  receiver.RXaddress("ALL");  // ตั้งชื่อตำแหน่งให้ตรงกัน ชื่อตั้งได้สูงสุด 5 ตัวอักษร
  receiver.init();
}

String message;

void loop(){
  if(receiver.available()){
    receiver.read(); // สั่งให้เริ่มอ่าน
    receiver.rxPL(message); // สั่งใหอ่านเก็บไว้ที่ตัวแปร
    Serial.println(message);
    message="";
  }
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้