การประยุกต์ใช้เทคโนโลยีกราฟิกสามมิติกับการออกกำลังกาย

APPLICATION3D FOR EXERCISE

นางสาวฐิมาภรณ์ สุวาส

THIMPORN SUWAS

นายณรงค์ฤทธิ์ ศรีสุขใส

NARONGRIT SRISUKSAI

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมสารสนเทศ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2553

APPLICATION3D FOR EXERCISE

MS. THIMAPORN SUWAS

MR. NARONGRIT SRISUKSAI

THIS THESIS IS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INFORMATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2010

หัวข้อปริญญานิพนธ์	การประยุกต์ใช้เทคโนโลยีกราฟิกสามมิติกับการออกกำลังกาย Application 3D for exercise		
รายชื่อนักศึกษา	นางสาวฐิมาภรณ์	สุวาส	รหัสนักศึกษา 50010412
	นายณรงค์ฤทธิ์	ศรีสุขใส	รหัสนักศึกษา 50010431
ปริญญา	วิศวกรรมศาสตรบัณฑิต		
สาขาวิชา	วิศวกรรมสารสนเทศ		
พ.ศ.	2553		
อาจารย์ที่ปรึกษาปริญญานิพนธ์	ผศ.บุณย์ชนะ ภู่ระหงษ์		

ปริญญานิพนธ์ฉบับนี้ ได้รับการอนุมัติให้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรวิศวกรรมศาสตรบัณฑิต คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

.....

(ผศ.บุณย์ชนะ ภู่ระหงษ์)
อาจารย์ผู้ควบคุมปริญญานิพนธ์

หัวข้อปริญญานิพนธ์	การประยุกต์ใช้เทคโนโลยีกราฟิกสามมิติกับการออกกำลังกาย Application 3D for exercise
รายชื่อนักศึกษา	นางสาวฐิมาภรณ์ สุวาส รหัสนักศึกษา 50010412 นายณรงค์ฤทธิ์ ศรีสุขใส รหัสนักศึกษา 50010431
ปริญญา	วิศวกรรมศาสตรบัณฑิต
สาขาวิชา	วิศวกรรมสารสนเทศ
พ.ศ.	2553
อาจารย์ที่ปรึกษาปริญญานิพนธ์	ผศ.บุณย์ชนะ ภู่ระหงษ์

บทคัดย่อ

ปริญญานิพนธ์นี้ ทำการออกแบบและสร้างโปรแกรมเกมกราฟิกสามมิติที่ตอบสนองตามการเคลื่อนไหวของศีรษะและการปั่นจักรยานของผู้เล่น โดยใช้เทคโนโลยีกราฟิกสามมิติด้วยโปรแกรม Blender3D กับเทคโนโลยีไมโครคอนโทรลเลอร์ตระกูล AVR ผ่านบอร์ดไมโครคอนโทรลเลอร์ Arduino มาประยุกต์ใช้เข้าด้วยกันเพื่อความเพลิดเพลินในการออกกำลังกายและทำให้การออกกำลังกายมีความสนุกสนานมากขึ้น โดยผู้เล่นสามารถเล่นเกมด้วยการปั่นจักรยานให้ภาพกราฟิกในเกมเคลื่อนที่ไปข้างหน้าควบคู่กับการบังคับทิศทาง ด้วยการเคลื่อนไหวศีรษะของผู้เล่น โดยความเร็วในการเคลื่อนที่ของเกมขึ้นอยู่กับความเร็วในการปั่นจักรยานของผู้เล่น

Thesis Title	Application 3D for exercise	
Student	Miss THIMAPORN SUWAS	Student ID. 50010412
	Mr. NARONGRIT SRISUKSRI	Student ID. 50010431
Degree	Bachelor of Engineering	
Program	Information Engineering	
Year	2009	
Thesis Advisor	Asst.Prof. Boonchana Purahong	

ABSTRACT

This project design and build Graphics 3D's game respond to head moving and cycling of player by applying Graphics 3D Technology with Blender and AVR Microcontroller using Arduino Board together. For the purpose of creating fun and enjoyable with exercise. When player cycling the Graphics game will move forward and player can control direction by moving head. The moving's velocity depend on cycling character of player

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้สำเร็จลุล่วงได้ด้วยดี เนื่องด้วยได้รับคำแนะนำและชี้แจงแนวทางการศึกษาข้อมูล รวมถึงการแก้ไขในส่วนบกพร่องต่างๆ เป็นอย่างดีตลอดระยะเวลาในการทำงานจาก ผศ.บุณย์ชนะ ภู่ระหงษ์ อาจารย์ที่ปรึกษาโครงการ โดยคณะผู้จัดทำต้องขอขอบพระคุณท่านเป็นอย่างสูง

ขอขอบพระคุณ ดร.พนารัตน์ เชิญถนอมวงศ์ รวมถึงคณาจารย์ภาควิชาวิศวกรรมสารสนเทศทุกท่าน ที่มีส่วนช่วยในการขัดเกลาให้ปริญญานิพนธ์นี้สำเร็จลุล่วงได้อย่างสมบูรณ์

ขอขอบคุณ พี่นัทและพี่ปุ้(นายวสันต์ จามกระโทก) ที่สละเวลาให้คำปรึกษา รวมถึงช่วยชี้แนะแนวทางในการแก้ไขปัญหาต่างๆที่เกิดขึ้นเป็นอย่างดีเสมอมา

กราบขอบพระคุณคุณพ่อ คุณแม่และทุกคนในครอบครัวที่ให้การสนับสนุน ดูแลห่วงใยคอยให้กำลังใจเสมอมาไม่ว่าจะเกิดปัญหาใดๆ

สุดท้ายขอขอบคุณทุกๆท่านที่มีส่วนร่วมในความสำเร็จของปริญญานิพนธ์นี้ที่ไม่สามารถกล่าวไว้ ณ ที่นี้ได้หมด รวมถึงเพื่อนๆภาควิศวกรรมสารสนเทศรุ่นที่ 10 ที่คอยช่วยเหลือกัน ไม่ทอดทิ้งกัน คุณประโยชน์อันใดที่เกิดจากปริญญานิพนธ์นี้เป็นผลมาจากความกรุณาของทุกท่านที่กล่าวมาข้างต้น คณะผู้จัดทำซาบซึ้งเป็นอย่างยิ่ง จึงใคร่ขอขอบพระคุณไว้ ณ ที่นี้ด้วย

นางสาวฐิมาภรณ์ สุวาส

นายณรงค์ฤทธิ์ ศรีสุขใส

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปริญญานิพนธ์

ในปัจจุบันโลกของเทคโนโลยีมีความก้าวหน้าและเติบโตเร็วไปในทุกๆ ด้าน ทั้งทางด้านการสื่อสาร คมนาคม เป็นต้น เทคโนโลยีที่ทันสมัยมากมายต่างเข้ามามีบทบาทในชีวิตของมนุษย์ ดังนั้นมนุษย์ในสังคมปัจจุบันจึงต้องปรับตัวพัฒนาให้เท่าทันเทคโนโลยีอยู่ตลอดเวลา โดยเฉพาะชีวิตในสังคมเมือง เนื่องด้วยสังคมเมืองมีการใช้ชีวิตที่บีบรัด มีข้อจำกัดทางด้านเวลาและสถานที่ทำให้การดำเนินชีวิตไม่สามารถมีรูปแบบอิสระเช่นยุคสมัยก่อน การดำเนินชีวิตต่างๆ ที่สามารถทำได้สะดวกกลับลำบากมากยิ่งขึ้น อาทิ การออกกำลังกาย คนในสังคมเมืองอันแออัดย่อมต้องมีสถานที่จำกัดในการออกกำลังกาย จึงทำให้ต้องไปหาสถานที่ออกกำลังกายนอกบ้านเพื่อเปิดโลกทัศน์และไม่ให้เกิดความอึดอัดและความจำเจ โดยสถานที่ดังกล่าวอาจจะต้องใช้เวลาในการหาสูง อีกทั้งยังหาได้ยากมีความน่าเบื่อ ทางด้านสถานที่ อากาศ และเวลา ดังนั้นการออกกำลังกายในสังคมเมืองจึงไม่สามารถทำได้โดยสะดวก การออกกำลังกายในที่ร่มจึงเป็นตัวเลือกอันดับต้นๆ ของคนในสังคมเมืองในยุคปัจจุบัน โดยคาดว่าในอนาคตการออกกำลังกายในรูปแบบนี้จะได้รับความนิยมสูงขึ้น แต่เนื่องจากพื้นที่ที่มีจำกัดการออกกำลังกายในรูปแบบดังกล่าวอาจจะสร้างความเบื่อหน่ายได้ง่าย

ดังนั้นการประยุกต์ใช้แอพพลิเคชั่นในรูปแบบกราฟิกสามมิติเพื่อนำเข้ามาผสมผสานกับการออกกำลังกาย อาจช่วยให้การออกกำลังกายมีสีสัน เป็นที่น่าสนใจมากขึ้น นอกจากจะสามารถเสริมสุขภาพกาย ยังสามารถเสริมสุขภาพจิตได้ และอาจจะทำให้บุคคลที่ไม่ชอบออกกำลังกายหันมาปรับเปลี่ยนทัศนคติในด้านการดูแลสุขภาพตนเองมากขึ้น

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1.2.1 เพื่อศึกษาการทำงานของเซนเซอร์วัดความเร่งเบอร์ ADXL345

1.2.2 เพื่อศึกษาการทำงานของไมโครคอนโทรลเลอร์ในตระกูล AVR โดยใช้Arduino

1.2.3 เพื่อศึกษาหลักการติดต่อข้อมูลระหว่างฮาร์ดแวร์กับซอฟแวร์ผ่านพอร์ตอนุกรม

1.2.4 เพื่อศึกษาหลักการรับ-ส่งข้อมูลระหว่างบอร์ด Arduino กับเซนเซอร์วัดความเร่งโดยใช้ I2C BUS

1.2.5 เพื่อศึกษาหลักการรับส่งค่าของไดนาโมจักรยาน

1.2.6 เพื่อศึกษาการสร้างกราฟิกในรูปแบบสามมิติ ให้เกิดความน่าสนใจและเป็นแรงจูงใจในการออกกำลังกายโดยใช้ Blender3D

1.2.7 เพื่อศึกษาโปรแกรมควบคุมกราฟิกสามมิติ ด้วยโปรแกรมภาษา Python

1.3 ขอบเขตของปริญญานิพนธ์

1.3.1 ออกแบบแอพพลิเคชั่นระบบกราฟิกสามมิติ คล้ายคลึงเกมให้ผู้เล่นสามารถควบคุมทิศทางได้

1.3.2 ผู้เล่น(หรือผู้ใช้งานเครื่องออกกำลังกาย)สามารถกำหนดทิศทางในเกมได้โดยการเอียงศีรษะไปทางทิศนั้นๆ

1.4 ประโยชน์ที่คาดว่าจะได้รับจากปริญญานิพนธ์

1.4.1 เรียนรู้และเข้าใจหลักการทั่วไปของเซนเซอร์วัดความเร่ง

1.4.2 เรียนรู้และเข้าใจการเขียนไมโครคอนโทลเลอร์ตลอดจนสามารถประยุกต์ใช้งานได้

1.4.3 สามารถเขียนโปรแกรมติดต่อระหว่างฮาร์ดแวร์กับซอฟแวร์ผ่านพอร์ตอนุกรมได้

1.4.4 มีความรู้ความเข้าใจโครงสร้างการส่งข้อมูลแบบ I2C BUS และสามารถเขียนโปรแกรมติดต่อระหว่างฮาร์ดแวร์ได้

1.4.5 มีความรู้ความเข้าใจหลักการทั่วไปของโปรแกรม Blender3D ตลอดจนสามารถสร้างในรูปแบบของเกมสามมิติได้

1.4.6 มีความรู้ความเข้าหลักการทั่วไปของโปรแกรมภาษา Python และสามารถเขียนควบคุมเกมสามมิติได้

1.5 อุปกรณ์ที่ต้องใช้

1.5.1 ฮาร์ดแวร์ (Hardware)

- Accelerometer ADXL345 จำนวน 1 ตัว
- ชุดโมดูลบอร์ด Arduino รุ่น Duemilanove Compatible จำนวน 1 ชุด
- ไอซีเบอร์ AT mega328 ของ AVR จำนวน 1 ตัว
- สาย Serial connector จำนวน 1 เส้น
- เครื่องคอมพิวเตอร์สำหรับพัฒนาและแสดงผล จำนวน 1 เครื่อง
- จักรยาน จำนวน 1 คัน
- ไดนาโมจักรยาน จำนวน 1 ตัว

1.5.2 ซอฟต์แวร์ (Software)

- โปรแกรม Blender3D สำหรับเขียนภาพกราฟิกสามมิติ
- โปรแกรม Python สำหรับควบคุมกราฟิกสามมิติ
- โปรแกรม Arduino สำหรับเขียนคำสั่งควบคุมการทำงานของชุดโมดูลบอร์ดไมโครคอนโทรลเลอร์ Arduino

1.6 เนื้อหาภายในปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้ประกอบไปด้วยเนื้อหาในบทต่างๆดังนี้

บทที่ 1 กล่าวถึงความเป็นมาและความสำคัญของปริญญานิพนธ์ วัตถุประสงค์ของปริญญานิพนธ์ ขอบเขตของปริญญานิพนธ์ ประโยชน์ที่คาดว่าจะได้รับจากปริญญานิพนธ์ อุปกรณ์ที่ต้องใช้ และขั้นตอนการดำเนินงาน

บทที่ 2 กล่าวถึงทฤษฎีพื้นฐานที่เกี่ยวข้องกับปริญญานิพนธ์ฉบับนี้ได้นำมารวบรวมไว้เพื่อเป็นพื้นฐานในการศึกษาและการทำการทดลองตามปริญญานิพนธ์

บทที่ 3 กล่าวถึงการออกแบบโครงสร้างด้านฮาร์ดแวร์และซอฟต์แวร์ เพื่อนำไปสร้างให้พร้อมใช้งานได้จริง

บทที่ 4 กล่าวถึงการทดลองและผลการทดลอง ซึ่งจะอธิบายถึงรายละเอียดการทำงานต่างๆ

บทที่ 5 กล่าวถึงการสรุปผลการดำเนินการ ซึ่งเป็นการนำผลการทดลองมาสรุปและชี้แจงถึงปัญหา อุปสรรคในระหว่างการดำเนินการ พร้อมชี้แนะแนวทางในการศึกษาและการพัฒนาต่อไป

1.7 ขั้นตอนการดำเนินงาน

ตารางที่ 1.1 แสดงระยะเวลาในการดำเนินงาน

ที่	รายละเอียด	พ.ศ. 2553							พ.ศ. 2554		
		มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.
1	**วางขอบเขตและคิดจุดประสงค์งาน**	←	→								
2	**กระบวนการศึกษาหาข้อมูล**		←	→							
3	ศึกษาหลักการทำงานของอุปกรณ์		←	→							
4	ศึกษาข้อมูลเกี่ยวกับโปรแกรมที่ใช้		←	→							
5	**กระบวนการออกแบบ**		←	→	→						
6	ออกแบบวงจรและตรวจสอบ		←	→							
7	ออกแบบโปรแกรมและตรวจสอบ			←	→						
8	**กระบวนการสร้างและพัฒนา**					←	→	→	→		
9	จัดเตรียมอุปกรณ์					←	→				
10	สร้างฮาร์ดแวร์และทดสอบ						←	→			
11	สร้างโปรแกรมเพื่อรวมระบบระหว่างโปรแกรมกราฟิกและฮาร์ดแวร์							←	→		
12	**ทดสอบใช้งานและปรับปรุงแก้ไข**							←	→	→	
13	**จัดทำเอกสารปริญญานิพนธ์**					←	→	→	→	→	

บทที่ 2
ทฤษฎีพื้นฐานที่ใช้

ปริญญานิพนธ์นี้มีวัตถุประสงค์คือ การประยุกต์ใช้เซ็นเซอร์สามแกนกับกราฟิกสามมิติ เพื่อพัฒนาการออกกำลังกายให้มีรูปแบบที่น่าสนใจและแตกต่างไปจากเดิม โดยอาศัยการทำงานจากสองส่วนหลักๆ คือ ส่วนของอุปกรณ์และส่วนของกราฟิกสามมิติ อุปกรณ์ที่ใช้เป็นชุดโมดูลบอร์ดไมโครคอนโทรลเลอร์ Arduino ที่มี MCU เป็น ATmega328 ควบคุมร่วมกับเซนเซอร์วัดความเร่งทิศทางแกน X, Y, Z ที่เรียกว่า Accelerometer ADXL345 ใช้ลักษณะการส่งข้อมูลในรูปแบบอนุกรม I2C Bus โดยการทำงานของชุดอุปกรณ์ดังกล่าวจะเกิดขึ้นเมื่อมีการส่งค่าในรูปแบบอนาล็อกจากไดนาโมจักรยาน ในส่วนของกราฟิกสามมิติมีการออกแบบกราฟิกโมเดลด้วยโปรแกรม Blender3D และควบคุมการทำงานระหว่างอุปกรณ์กับกราฟิกสามมิติด้วยโปรแกรมภาษา Python ดังนั้นเพื่อให้อุปกรณ์และโปรแกรมต่างๆสามารถติดต่อกันได้ตามขอบเขตที่วางไว้ จึงต้องศึกษาข้อมูลและทฤษฎีที่เป็นพื้นฐานสำคัญ โดยแบ่งออกเป็นกลุ่มต่างๆ ตามลักษณะการทำงานดังนี้

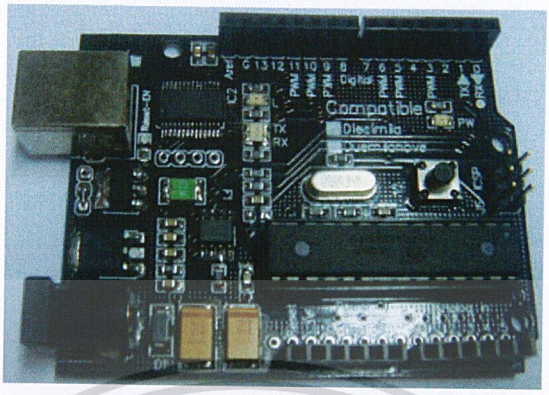
2.1 ทฤษฎีไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ (Microcontroller) เป็นอุปกรณ์อิเล็กทรอนิกส์แบบหนึ่งที่รวมเอาหน่วยประมวลผล หน่วยคำนวณทางคณิตศาสตร์และลอจิก วงจรรับสัญญาณอินพุต วงจรส่งสัญญาณเอาท์พุต รวมถึงหน่วยความจำวงจรกำเนิดสัญญาณนาฬิกาไว้ด้วยกัน ทำให้สามารถนำไปใช้งานแทนวงจรอิเล็กทรอนิกส์ที่ซับซ้อนได้เป็นอย่างดี โดยไมโครคอนโทรลเลอร์มาจากคำสองคำรวมกันคือ “ไมโคร” ซึ่งหมายถึงไมโครโปรเซสเซอร์ เป็นอุปกรณ์ประมวลผลข้อมูลขนาดเล็กภายในประกอบด้วย หน่วยประมวลผลกลางหรือ CPU ประกอบด้วยหน่วยคำนวณทางคณิตศาสตร์และลอจิก วงจรเชื่อมต่อหน่วยความจำ วงจรกำเนิดสัญญาณนาฬิกา อีกคำหนึ่งคือคำว่า “คอนโทรลเลอร์” หมายถึง อุปกรณ์ควบคุม ดังนั้นไมโครคอนโทรลเลอร์จึงเป็นอุปกรณ์ที่ใช้ในการควบคุม โดยที่สามารถเขียนโปรแกรมเพื่อกำหนดรูปแบบการควบคุมได้อย่างเป็นอิสระ

2.1.1 ชุดโมดูลบอร์ดไมโครคอนโทรลเลอร์ Arduino รุ่น Duemilanove Compatible

เป็นไมโครคอนโทลเลอร์ตระกูล AVR โดยใช้ไมโครคอนโทลเลอร์เบอร์ ATmega 328 เป็น MCU ประจำบอร์ด โดย MCU รุ่นนี้ มีขา pin ทั้งหมด 28 ขาและมีจุดเด่นคือเป็นไมโครคอนโทรลเลอร์ขนาดเล็ก แต่เพียบพร้อมไปด้วยทรัพยากรพื้นฐานต่างๆ อย่างครบถ้วน จึงมีความเหมาะสมเป็นอย่างยิ่งในการใช้งานทั่วไป สำหรับภายในมีทั้งระบบฮาร์ดแวร์ของ SPI,

UART , I2C, Watchdog, Timer/Counter,PWM เป็นต้นและสามารถใช้ในการพัฒนาโปรแกรมด้วย Arduino ได้ทันที



รูปที่ 2.1 ชุดโมดูลบอร์ดไมโครคอนโทรลเลอร์ Arduino

2.1.2 คุณสมบัติของบอร์ดไมโครคอนโทรลเลอร์

ชุดโมดูลบอร์ดไมโครคอนโทรลเลอร์ Arduino มี ATmega328 เป็น MCU ตระกูล AVR ประจำบอร์ด โดยเลือกใช้แหล่งกำเนิดสัญญาณนาฬิกาแบบ Crystal Oscillator ที่มีค่า 16 MHz เพื่อให้สามารถใช้งานพอร์ทสื่อสารอนุกรมได้อย่างลงตัว

- บอร์ดสามารถเปลี่ยนการติดตั้ง MCU เป็นแบบ 28 pin หรือเบอร์อื่นในอนุกรมเดียวกันได้โดยไม่ต้องมีการดัดแปลงใดๆ เช่น ATmega88 เป็นต้น
- มีหน่วยความจำแบบ Flash 32 Kbyte โดยแบ่งเป็น Boot loader 2 Kbyte
- มี EPROM 1Kbyte/SRAM 2Kbyte
- มีพอร์ตเป็น 14 digital input/output pins ซึ่งมี 6 pin สามารถสร้างเป็น PWM outputs
- มีพอร์ต 6 Analog input/output pins
- ไฟกระแสตรง (DC) ขา I/O pin มีค่า 40 mA
- ไฟกระแสตรง (DC) ขา 3.3V pin มีค่า 50 mA
- MCU ประจำบอร์ดที่ได้รับการติดตั้ง Boots loader สามารถอัพโหลดโค้ดให้บอร์ดผ่านทางพอร์ตสื่อสารแบบอนุกรมได้ทันที
- มีขั้วต่อ USB Interface สื่อสารข้อมูลแบบอนุกรมเข้า คอมพิวเตอร์ได้ทันที
- มี LED สำหรับแสดงสถานะไฟเลี้ยง
- มี LED แสดงสถานการณ์รับส่งข้อมูล
- ใช้ไฟเลี้ยง ประจำบอร์ด 7-12 Volt33

2.1.3 ทฤษฎีภาษา Arduino

"Arduino" เป็นภาษาอิตาลีซึ่งเป็นชื่อของโครงการพัฒนาไมโครคอนโทรลเลอร์ตระกูล AVR แบบ Open source ที่ได้รับการปรับปรุงมาจากโครงการ Open source ของAVR

Arduino มีจุดเด่นในเรื่องของความง่ายในการเรียนรู้และใช้งาน เนื่องจากมีการออกคำสั่งต่างๆขึ้นมาสนับสนุนการใช้งาน ด้วยรูปแบบที่ง่ายไม่ซับซ้อน ซึ่งแม้ว่า Arduino จะมีรูปแบบการใช้งานคล้ายๆกับไมโครคอนโทรลเลอร์อย่าง Basic Stap ของ Parallax แต่ก็มีจุดเด่นกว่ารายอื่น คือ

- ราคาไม่แพง เนื่องจากมี Source Code และวงจร แจกให้ฟรี สามารถต่อวงจรขึ้นมาใช้เองได้ รวมถึงมีการเปิดเผยวงจร Source code ทั้งหมดทำให้สามารถนำไปพัฒนาต่อยอดได้ดี ทั้งด้านฮาร์ดแวร์และซอฟแวร์
- โปรแกรมที่ใช้พัฒนาของ Arduino สามารถรองรับการทำงานทั้ง Window, Linux และ OSx
- รูปแบบคำสั่งง่ายต่อการใช้งาน แต่สามารถนำไปใช้งานจริงๆ กับส่วนที่มีความซับซ้อนมากๆได้ และยังสามารถสร้างคำสั่งรวมถึง Library ใหม่ๆ ขึ้นมาใช้งานเองได้ เมื่อมีความชำนาญมากขึ้น

2.1.4 เปรียบเทียบภาษาซีกับ Arduino

สำหรับการเขียนโปรแกรมของ Arduio นั้นใช้ภาษา C++ ซึ่งเป็นรูปแบบของภาษาซีประยุกต์รูปแบบหนึ่งที่มีโครงสร้างการทำงานของตัวภาษาโปรแกรมโดยรวมคล้ายกับ ภาษาซีมาตรฐานทั่วไป เพียงแต่ได้มีการปรับปรุงเพื่อลดความยุ่งยากในการใช้งานลดลง และให้ผู้ใช้สามารถใช้งานเขียนโปรแกรมได้ง่าย สะดวกมากกว่าการเขียนภาษาซีแบบมาตรฐาน แต่ในความเป็นจริงนั้นโปรแกรมดังกล่าวไม่ใช่ C-compiler โดยตรง เนื่องจาก Arduino จะมีลักษณะการทำงานเช่นเดียวกับ Text Editor ของภาษา C++ ตัวหนึ่งโดยจะทำงานร่วมกับ Utility บางส่วนที่ Arduino สร้างขึ้นมารองรับโดย Arduino จะใช้รูปแบบการทำงานของ Editor เป็นฉากหน้า ในการติดต่อสื่อสารกับผู้ใช้เท่านั้น ส่วนเบื้องหลังจริงๆแล้ว Arduino จะไปเรียกใช้ตัวแปลภาษาซีและ Utility อื่นที่ใช้เป็นเครื่องมือพัฒนาโปรแกรมของไมโครคอนโทรลเลอร์ตระกูล AVR อีกทีหนึ่ง Arduino จะเลือกใช้คอมไพลเลอร์ของ "GNU AVR-GCC Toolchain" ร่วมกับ Library function ของ "avr-libc" ส่วน Utility ที่ใช้ในการอัพโหลดโค้ดให้กับ AVR จะใช้ในส่วนของ "AVRDude"

2.1.5 โครงสร้างการเขียนโปรแกรมภาษาซีของ Arduino

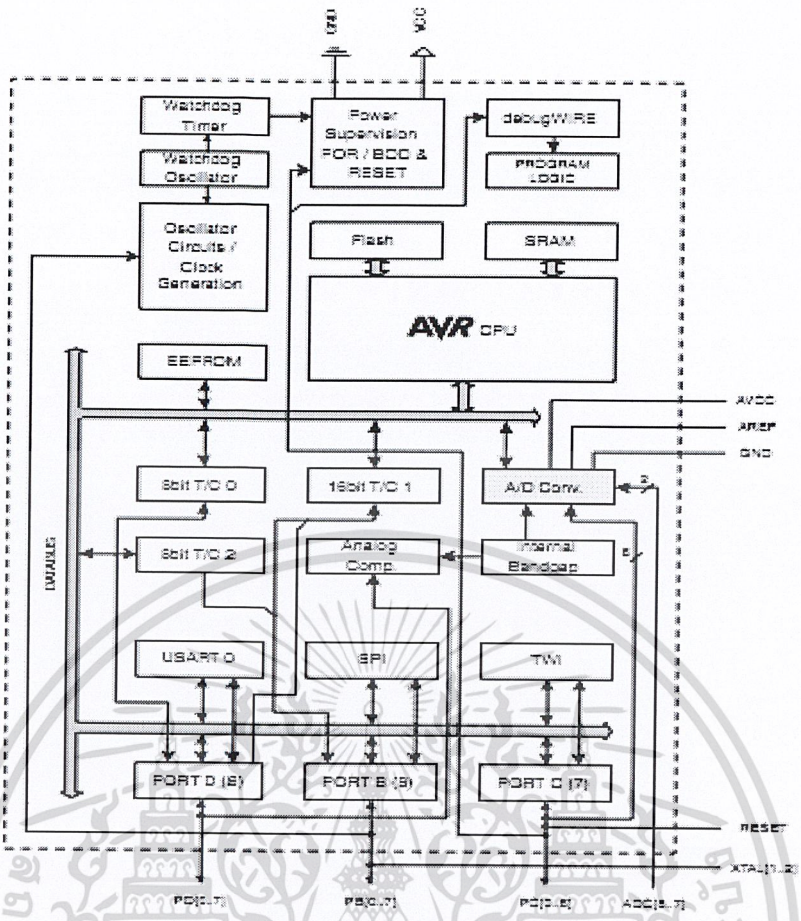
ภาษาซีของ Arduino จะจัดแบ่งรูปแบบโครงสร้างของการเขียนโปรแกรมเป็นส่วนย่อยๆ หลายๆส่วน โดยเรียกแต่ละส่วนว่า "ฟังก์ชั่น" และเมื่อนำฟังก์ชันมารวมเข้าด้วยกัน ก็จะเรียกว่า "โปรแกรม" โดยโครงสร้างการเขียนโปรแกรมของ Arduino ทุกๆโปรแกรมจะประกอบไปด้วยฟังก์ชั่นจำนวนเท่าใดก็ได้ แต่อย่างน้อยสุดต้องมี 2 ฟังก์ชั่นคือ setup() และ loop()

- **Setup()** : เป็นฟังก์ชั่นบังคับที่ต้องกำหนดให้มีทุกๆโปรแกรม ถึงแม้ว่าบางโปรแกรมจะไม่ต้องการใช้งานก็ยังจำเป็นต้องประกาศไว้เสมอ เพียงแต่ไม่ต้องเขียนคำสั่งใดๆไว้หลังวงเล็บปีกกา {} ที่ใช้เป็นตัวกำหนดขอบเขตของฟังก์ชั่น โดยฟังก์ชันนี้จะใช้สำหรับบรรจุคำสั่งที่ต้องการให้โปรแกรมทำงานเพียงรอบเดียว ตอนเริ่มต้นทำงานของโปรแกรมครั้งแรกเท่านั้น ซึ่งได้แก่คำสั่งเกี่ยวกับการ Setup ค่าการทำงานต่างๆ เช่น การกำหนดหน้าที่ของการใช้งานของ PinMode และค่า Baudrate สำหรับการใช้งานสื่อสารพอร์ตอนุกรม เป็นต้น
- **Loop()** : เป็นส่วนฟังก์ชั่นบังคับที่ต้องกำหนดให้มีในทุกโปรแกรม เช่นเดียวกันกับฟังก์ชั่น Setup() โดยฟังก์ชั่น Loop() นี้จะใช้ในการบรรจุคำสั่งที่ต้องการให้โปรแกรมทำงานเป็นวนรอบซ้ำๆ กันไปไม่รู้จบ ซึ่งเปรียบเทียบได้กับ ฟังก์ชั่น main () ใน ANSCI-C นั่นเอง

2.2 ทฤษฎีไมโครคอนโทรลเลอร์ตระกูล AVR รุ่น ATmega 328P

2.2.1 สถาปัตยกรรมชั้นสูงแบบ RISC(Reduce Instruction Set Computer)

- RISC คือตัวทำให้การประมวลผลมีความเร็ว 1 คำสั่ง/1 clock หรือ CPU สามารถประมวลคำสั่งได้ 1 MIPS/MHz
- ชุดคำสั่ง 131 คำสั่งต่อ 1 รอบนาฬิกา
- รีจิสเตอร์ขนาด 8 บิต 32 ตัว
- ความเร็วในการประมวลผลมากกว่า 20 ล้านคำสั่งต่อวินาที (MIPS) ที่ 20 MHz



รูปที่ 2.2 สถาปัตยกรรมชั้นสูงแบบ RISC [1]

โดยมีคุณสมบัติต่างๆดังต่อไปนี้

หน่วยความจำ

- แบบ 32 Kbyte สามารถเขียนลบโปรแกรมได้ 10,000 ครั้ง
- แบบ EPROM 1Kbyte สามารถเขียนและลบโปรแกรมได้ 100,000 ครั้ง
- แบบ SRAM 2 Kbyte

ไฟเลี้ยง

- ระหว่าง 1.8 ถึง 2.5 v

ความถี่สัญญาณนาฬิกา

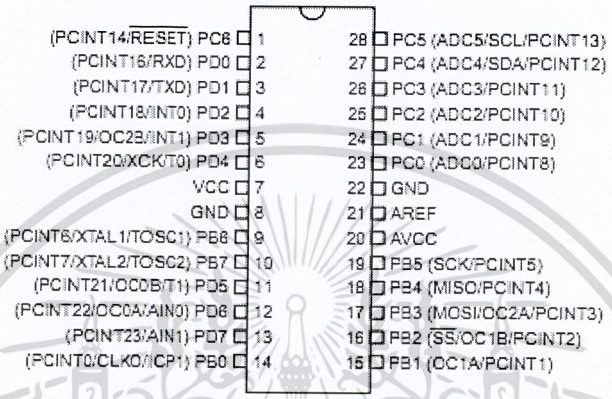
- ระหว่าง 0 ถึง 4 MHz
- มีการรองรับอุปกรณ์ต่อพ่วง
- อุปกรณ์สื่อสารข้อมูลแบบอนุกรมแบบ I2C และ USART

อื่นๆ

- มีระบบ Reset แบบอัตโนมัติเมื่อจ่ายกระแสไฟฟ้าเข้าไมโครคอนโทรลเลอร์
- มีฟังก์ชั่นตรวจสอบแรงดัน

- มีระบบการขัดจังหวะทั้งภายในและภายนอก
- มีระบบตรวจจับความผิดพลาดของ CPU
- มีโหมดอนุรักษ์พลังงาน 5 mode ได้แก่ Idle, ADC Noise Reduction, Power-Save, Power-Down, Standby

2.2.2 ส่วนประกอบต่างๆของไอซี ATmega 328P



รูปที่ 2.3 ไอซี ATmega 328P [1]

ตารางที่ 2.1 ตารางแสดงฟังก์ชั่นการทำงานของ ATmega 328P

ชื่อ	รายละเอียด	ขา
GND	ขากราวด์ต่อสายดิน	8,22
VCC	ไฟเลี้ยง 1.8 ถึง 5.5v	7
Port B(PB 7: 0) XTAL1/XTAL2/TOSC1/TOSC2	- เป็นพอร์ต 2 ทิศทาง ขนาด 8 บิต โดยสามารถกำหนดให้ขาของแต่ละพอร์ตสามารถตั้งค่าให้ Pull up Resistor ได้ (ภายในเป็นอิสระแยกจากกันเพื่อดึงแรงดันของลอจิก 1 ให้เท่ากับ 5 v) - สามารถใช้งานพิเศษตามความต้องการของ ATmega 328 โดยขึ้นอยู่กับการตั้งค่าสัญญาณนาฬิกาที่ขา PB6 ที่ใช้เป็นแรงดัน Oscillator และขาอินพุตของวงจรสัญญาณ Clock Oscillator	9,10,14-19
PC6/RESET	ขา Reset	1

ชื่อ	รายละเอียด	ขา
Port D (PD7:0)	- เป็นพอร์ตสองทิศทางขนาด 8 บิต โดยสามารถกำหนดให้ขาของแต่ละพอร์ตสามารถตั้งค่าให้ Pull up Resistor ได้ - สามารถใช้งานพิเศษตามความต้องการของ ATmega 328	1-6,11-13
AVCC	ใช้จ่ายไฟให้กับวงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอล มักจะต่อเข้ากับขา VCC	20
AREF	แรงดันอ้างอิงที่ใช้งานในส่วนของวงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอลมักต่อกับ VCC	21
ADC7:6 (TQFP and QFN/MLF Package Only)	ขากำลังงานใช้แปลงสัญญาณอนาล็อกเป็นดิจิตอล	23-28
Port C (PC5:0)	- เป็นพอร์ตสองทิศทางขนาด 8 บิต โดยสามารถกำหนดให้ขาของแต่ละพอร์ตสามารถตั้งค่าให้ Pull up Resistor ได้ - สามารถใช้งานพิเศษตามความต้องการของ ATmega 328	23-28

2.3 ทฤษฎีการสื่อสารแบบอนุกรม

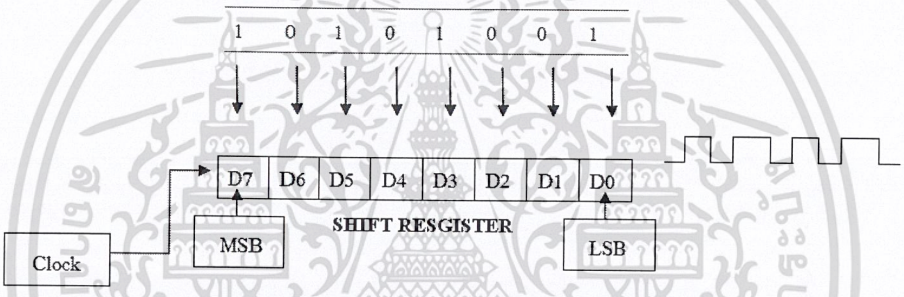
ทฤษฎีการสื่อสารแบบอนุกรม เป็นการรับส่งข้อมูลทีละบิตจนครบ 1 ไบต์ ถ้าต้องการส่งข้อมูล 1 ไบต์คือ D0 - D7 อาจจะส่ง D0 ออกไปก่อน แล้วตามด้วย D1 ไปเรื่อยๆจนถึง D7 โดยการส่งข้อมูลแบบอนุกรมมีความแตกต่างกับแบบขนานทั้งข้อดีและข้อเสียดังนี้ การส่งข้อมูลแบบขนานสามารถรับส่งข้อมูลได้อย่างรวดเร็ว เพราะการส่งข้อมูลสามารถทำได้ทีเดียวครบ 1 ไบต์ แต่ถ้าต้องการส่งระยะทางไกลๆจะต้องสิ้นเปลืองสายสัญญาณมาก สำหรับการส่งข้อมูลแบบอนุกรมจะแตกต่างคือ ถ้าส่งระยะทางไกลๆจะประหยัดสายสัญญาณ จากการใช้สายสัญญาณอย่างน้อยสองเส้นคือ สายสัญญาณกับสายกราวด์ แต่การรับส่งข้อมูลจะใช้เวลานานกว่าเนื่องจากเป็นการรับส่งค่าทีละบิต

2.3.1 การสื่อสารข้อมูลแบบ Asynchronous

เป็นการรับและส่งข้อมูลโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วยแต่จะใช้การกำหนดค่าอัตราเร็วในการรับส่งข้อมูลให้มีค่าเท่ากัน ซึ่งเรียกอัตรานี้ว่า Baud rate (Bit/second) รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบ Asynchronous ประกอบด้วย 4 ส่วน คือ

1. บิตเริ่มต้น (Start bit) มีขนาด 1 บิต
2. บิตข้อมูลแบบอนุกรม มีขนาด 8 บิต
3. บิตตรวจสอบพาริตี้ (parity bit) มีขนาด 1 บิตหรือไม่มีก็ได้
4. บิตหยุด (Stop bit) มีขนาด 1 บิต

เมื่อไมโครคอมพิวเตอร์ต้องการรับส่งข้อมูลแบบอนุกรม ตัวไมโครคอมพิวเตอร์จะส่งข้อมูลออกไปแบบขนานก่อน จากนั้นจะมีอุปกรณ์ที่มาต่อพอร์ต เพื่อแปลงข้อมูลแบบขนานให้เป็นอนุกรมทีหนึ่ง ตัวแปลงข้อมูลนี้อาจพิจารณาว่าเป็น Shift Register



รูปที่ 2.4 การส่งข้อมูลแบบอนุกรม [2]

สำหรับตัวรับข้อมูลแบบอนุกรมนั้นเมื่อตัวรับข้อมูลทำงานจะเป็นการรับเข้ามาใน shift Register แล้วส่งข้อมูลให้ไมโครคอมพิวเตอร์แบบขนานอีกที ระบบคอมพิวเตอร์ในปัจจุบันมีตัวแปลง Pararell to Serial และ Serial to Parallel อยู่ในชิพไอซีเรียกว่า Universal Asynchronous Receiver Transmitter (UART) การส่งข้อมูลแบบอนุกรมนั้นจะต้องมีการเพิ่มเติมข้อมูลบางอย่างเข้าไปเพื่อทำให้ข้อมูลมีความถูกต้องมากขึ้น

Start	D0	D1	D2	D3	D4	D5	D6	D7	**Parity**	**Stop**

รูปที่ 2.5 บิตต่างๆของการรับส่งข้อมูลแบบอนุกรม [2]

ถ้ามีการส่งข้อมูลแบบ 8 บิตจะต้องมีการส่งบิตแรกออกไปก่อน เรียกว่า บิตเริ่มต้น (Start Bit) ถ้ามีการส่งข้อมูลหลายๆไบต์ออกมา บิตนี้จะเป็นตัวบอกว่ามีข้อมูลใหม่มาแล้ว โดยทั่วไปบิตเริ่มต้นมักมีระดับลอจิกเป็น “0” ต่อจากนั้นจะเป็น ข้อมูลบิต D0 - D7 จากนั้นตามด้วยบิตตรวจสอบและบิตหยุด เพื่อบอกการสิ้นสุดของบิตข้อมูลซึ่งบิตหยุดอาจมากกว่า 1 บิตก็ได้

การสื่อสารแบบอนุกรมนี้ การกำหนดอัตราเร็วในการรับส่งข้อมูลจะบอกเป็นบิตต่อวินาที bps ที่เรียกว่าอัตราบอดเรต (BAUD RATE) โดยค่ามาตรฐานที่ใช้กรณีหลายค่า เช่น ค่า 9600 หรือค่า 115200 เป็นต้น

2.4 หลักการพื้นฐานของ USB

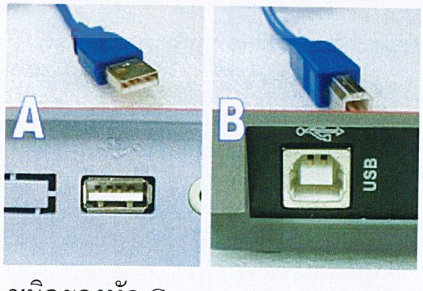
USB หรือ Universal Serial Bus ได้ถูกกำหนดขึ้นมาโดยบริษัทยักษ์ใหญ่ผู้นำทางด้านอุปกรณ์ไฟฟ้า อิเล็คทรอนิคส์และคอมพิวเตอร์ ซึ่งประกอบไปด้วย COMPAQ, IBM, DEC, Intel, Microsoft, NEC และ Northern Telecom ช่วยกันวางมาตรฐานให้อยู่ในรูปแบบเดียวกัน โดยในยุคเริ่มแรกนั้นมาตรฐานของ USB ที่ ออกสู่สาธารณะชนเป็นครั้งแรกเมื่อวันที่ 11 พฤศจิกายน ปี ค.ศ. 1994 คือ Revision 0.7 และได้ปรับปรุงแก้ไขเรื่อยมา จนกระทั่งเมื่อ วันที่ 15 มกราคม ค.ศ. 1996 ได้ออกมาเป็น Revision 1.0 (USB1.0) ได้เป็นผลสำเร็จและยังได้ปรับปรุง พัฒนาแก้ไขปัญหาต่างๆ เรื่อยมาจนเมื่อวันที่ 23 กันยายน ค.ศ. 1998 ได้เป็น Revision 1.1 (USB1.1) เมื่อความเร็วที่ได้ ยังไม่เพียงพอกับความต้องการ

ดังนั้นทางกลุ่มผู้พัฒนา หรือ USB-IF (USB Implementers Forum, Inc.) ได้ร่างมาตรฐาน USB รุ่นใหม่ และได้ข้อสรุป เป็นมาตรฐานที่แน่นอน คือ USB 2.0 ในเดือนเมษายน ปี ค.ศ. 2000 สำหรับความเร็วในการรับ-ส่งข้อมูลนั้น USB1.1 จะมีความเร็วอยู่ที่ 12Mbps ส่วน USB 2.0 นั้น รองรับระดับการรับส่งข้อมูลได้ถึง 3 ระดับ คือ

- ความเร็ว 1.5 Mbps (Low Speed) สำหรับการเชื่อมต่อกับอุปกรณ์ที่ไม่จำเป็นต้องส่งข้อมูลคราวละมากๆ
- ความเร็ว 12 Mbps (Full Speed) สำหรับการเชื่อมต่อกับ USB 1.1
- ความเร็ว 480 Mbps (Hi-Speed) สำหรับการเชื่อมต่อกับ USB 2.0 ด้วยกัน

2.4.1 ชนิดของหัว Connector

Connector มีอยู่ 2 แบบ คือ แบบ A และ แบบ B และ Socket ดังรูป



รูปที่ 2.6 ชนิดของหัว Connector แบบ A และแบบ B [3]

ลักษณะของการทำงานของหัวต่อทั้งสองแบบมีดังนี้

- แบบ A : เป็นการส่งข้อมูลจากอุปกรณ์ไปยังเครื่อง Computer เพื่อการประมวล เรียกว่า UpStream
- แบบ B : เป็นการส่งข้อมูลเข้าหาอุปกรณ์ เรียกว่า DownStream

2.4.2 ลักษณะการเชื่อมต่อ

การเชื่อมต่อใช้งานนั้นสามารถ เชื่อมต่อร่วมกันได้ทั้งที่เป็น USB1.1 และ USB 2.0 แต่จะได้ความเร็วที่ต่างกัน

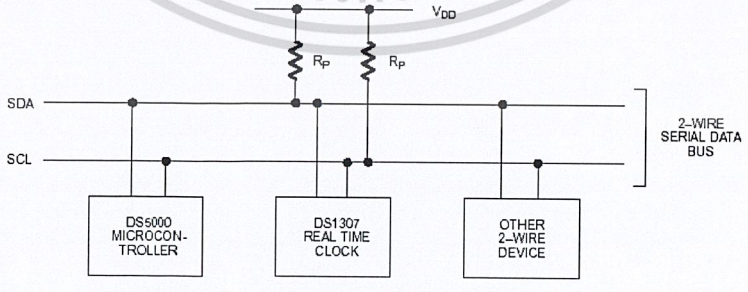
ถ้าหากต่ออุปกรณ์มาตรฐาน USB1.1 บนระบบบัสที่เป็น USB2.0 จะได้ความเร็ว = 12Mbps

ถ้าหากต่ออุปกรณ์มาตรฐาน USB2.0 บนระบบบัสที่เป็น USB1.1 จะได้ความเร็ว = 12Mbps

ถ้าหากต่ออุปกรณ์มาตรฐาน USB2.0 บนระบบบัสที่เป็น USB2.0 จะได้ความเร็ว = 480Mbps

2.5 ทฤษฎีการสื่อสารแบบ I2C Bus

Inter Integrate Circuit Bus (IIC Bus) เรียกสั้นๆว่า "I2C Bus" ซึ่งเป็นชื่อของการติดต่อสื่อสารแบบอนุกรมแบบหนึ่ง ถูกคิดค้นและพัฒนาโดย PHILIPS SEMICONDUCTOR เมื่อหลายปีก่อน แต่เพิ่งมาได้รับความแพร่หลายเมื่อไม่นานมานี้ ซึ่งปัจจุบันนี้ได้มีหลายบริษัทให้ความสนใจในการติดต่อสื่อสารแบบนี้เป็นอย่างมาก เนื่องจากรูปแบบการเชื่อมต่ออุปกรณ์ด้วยระบบบัสแบบนี้จะมีข้อดีคือใช้สัญญาณในการเชื่อมต่อเพียง 2 เส้น (SCL และ SDA) แต่สามารถเชื่อมต่ออุปกรณ์จำนวนหลายตัวในบัสเดียวได้ ซึ่งปัจจุบันถือว่าเป็นยุคสมัยของไมโครคอนโทรลเลอร์ขนาดเล็ก เนื่องจากระบบการทำงานของวงจรต่างๆมุ่งเน้นออกแบบให้มีขนาดเล็กกะทัดรัดและสามารถใช้งานได้หลายหลาย ดังนั้นพวก chip support ต่างๆจึงได้ออกแบบมาเพื่อตอบสนองการใช้งานแบบ I2C Bus มากยิ่งขึ้น



รูปที่ 2.7 ลักษณะการการเชื่อมต่ออุปกรณ์แบบ I2C Bus [4]

2.5.1 การเชื่อมต่อบัสแบบ I2C Bus

การเชื่อมต่อนี้จะใช้สัญญาณทั้งหมด 2 เส้นคือ SCL และ SDA โดยการติดต่อระหว่างอุปกรณ์จะเป็นแบบ 2 ทิศทาง โดยสัญญาณทั้งสองเส้นจะต้องต่อกับตัวต้านทาน Pull up ไว้เพื่อให้สถานะของบัสในขณะที่ไม่ถูกใช้งานเป็นลอจิก "1" ทั้งคู่โดยอุปกรณ์ต่างๆที่ถูกออกแบบมาเพื่อเชื่อมต่อกับบัสแบบนี้ จะต้องสร้างวงจรภาคเอาพุตท์ให้เป็นแบบ Open Drain หรือ Open collector เสมอเพื่อให้อุปกรณ์สามารถติดต่อกันได้มากกว่า 1 อุปกรณ์

2.5.2 หลักการของ I2C

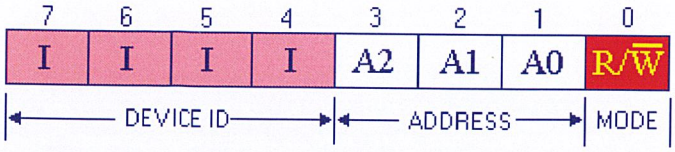
บัส I2C ประกอบด้วยสายสัญญาณ 2 เส้นดังที่ได้กล่าวมาแล้วคือ SDA และ SCL อุปกรณ์ที่ต่อพ่วงบนบัสสามารถมีได้มากมาย ดังนั้นจึงต้องมีการกำหนดรูปแบบของการติดต่อบนบัส หรือเรียกว่า โปรโตคอล (protocol) เพื่อให้ผู้ใช้งานทราบว่า ขณะนี้อุปกรณ์ใดติดต่อกันอยู่ และอุปกรณ์ตัวใดเป็นตัวรับหรือตัวส่ง ต่อไปนี้เป็นการอธิบายลักษณะ หน้าที่ และนิยามของอุปกรณ์ที่ต่ออยู่บนบัส I2C เพื่อเป็นข้อตกลงพื้นฐานก่อนที่จะอธิบายการทำงานของบัส I2C ต่อไป

อุปกรณ์ที่เป็นผู้สร้างข้อมูลหรือส่งข้อมูลเรียกว่า ตัวส่ง (transmitter) อุปกรณ์ที่เป็นผู้รับข้อมูลเรียกว่า ตัวรับ (receiver) ในอุปกรณ์บนบัส I2C สามารถเป็นได้ทั้งตัวรับและตัวส่งบางอุปกรณ์ทำหน้าที่เป็นตัวรับเพียงอย่างเดียวจะไม่มีอุปกรณ์ใดบนบัส I2C เรียกว่า มาสเตอร์ (master) อุปกรณ์ที่ถูกควบคุมหรืออุปกรณ์ที่ต่อพ่วงเข้าไปบนบัส I2C เรียกว่า สเลฟ (slave) โดยข้อกำหนด 2 ประการสำคัญของการติดต่อบนบัส I2C คือ

1.) การถ่ายทอดข้อมูลจะเกิดขึ้นได้เมื่อบัสว่างเท่านั้น

2.) ในระหว่างการถ่ายทอดข้อมูล เมื่อใดก็ตามที่สาย SCL มีสถานะเป็นลอจิกสูง สายข้อมูลต้องรักษาข้อมูลไว้ อย่าให้เกิดการเปลี่ยนแปลงขึ้นเด็ดขาด มิฉะนั้น สัญญาณที่เกิดขึ้นจะได้รับการแปลความหมายเป็นสัญญาควบคุมแทน

การรับส่งข้อมูลของ I2C บัส จะเริ่มต้นด้วยการที่ตัวแม่สร้างสภาวะเริ่มต้น (Start Condition) เพื่อขอใช้บัสจากนั้นจึงเริ่มการส่งรหัสควบคุม (Control Byte) เพื่อใช้ระบุตำแหน่งของแอดเดรสตัวลูกที่ต้องการใช้ติดต่อด้วยในระบบบัส โดยค่าตำแหน่งแอดแดรสนี้อุปกรณ์แต่ละตัวจะมีรหัสแอดเดรสที่แตกต่างกันออกไป ไม่มีการซ้ำกันในระบบบัสเดียวกัน โดยรหัส Control Byte จะมีขนาด 8 บิตซึ่ง 7 บิตแรก เริ่มจาก MSB จะเป็นค่าตำแหน่งแอดเดรสของตัวลูก ส่วนบิตที่ 8(LSB) จะเป็นบิตสุดท้ายของไบท์ใช้สำหรับระบุทิศทางของการรับส่ง (R/W) โดยถ้าบิต LSB มีค่าเป็น "0" จะหมายถึงตัวแม่เขียนข้อมูลไปหาตัวลูก โดยข้อมูลจะรับส่งกันครั้งละ 1 ไบต์ และปิดท้ายข้อมูลของแต่ละไบต์ด้วยบิตแสดงการตอบรับ (Acknowledge Bit) โดยลักษณะโครงสร้าง Control Byte ของอุปกรณ์ แบบ I2C มีดังนี้



รูปที่ 2.8 ลักษณะของ Control Byte ของ I2C Bus [4]

จากรูปที่ 2.8 จะเห็นได้ว่ารหัส Control Byte ของอุปกรณ์ I2C จะมีขนาด 8 บิต โดยที่บิตที่ 7 ถึงบิตที่ 4 จะเป็นรหัสประจำตัวของอุปกรณ์แต่ละตัวที่ถูกกำหนดไว้ตายตัวจากโรงงาน ซึ่งผู้ใช้งานต้องศึกษาข้อมูลจากคู่มือ Datasheet ของอุปกรณ์นั้นเองว่าอุปกรณ์ที่จะนำมาใช้งานมีรหัสประจำตัวเป็นเท่าใด ส่วนบิตที่ 3 ถึงบิตที่ 1 จะมีไว้สำหรับการเลือกต่ออุปกรณ์ที่อยู่ในบัส โดยค่าของทั้ง 3 บิตนี้ จะต้องเป็นค่าที่ ตรงกับการกำหนดสถานะทางลอจิก ให้กับขาสัญญาณจอง A2,A1,A0 ของอุปกรณ์ด้วย ตัวอย่างเช่น อุปกรณ์ที่มีรหัสประจำตัวเป็น “0111” อาจถูกออกแบบให้สามารถต่อร่วมกันภายในบัสเดียวกันจำนวน 8 ตัว โดยการกำหนดสภาวะของลอจิกให้กับขาสัญญาณ A2,A1,A0 ให้มีความแตกต่างกันจากวงจรที่ต่ออยู่

แต่อย่างไรก็ตาม อุปกรณ์ I2C บางตัว อาจถูกออกแบบให้ต่อได้ในระบบบัสเดียวกันเพียงตัวเดียวโดยไม่มีขาสัญญาณในการเลือกตำแหน่งของอุปกรณ์อยู่ด้วย ค่าของ Control Byte ในตำแหน่งบิตที่ 3 ถึงบิตที่ 1 ก็อาจถูกกำหนดไว้ตายตัวเป็น 000 เสมอก็ได้ ส่วนค่า Control Byte ของบิต 0 นั้นจะถูกกำหนดไว้เป็นทิศทางของข้อมูล โดยถ้าบิต 0 มีค่าเป็น 1หมายถึงตัวแม่ต้องการอ่านค่าข้อมูลจากอุปกรณ์ ถ้ามีค่าเป็น 0 หมายถึงตัวแม่ต้องการเขียนค่า

สำหรับการรับส่งข้อมูลนั้นไม่มีการกำหนดตายตัวว่าจะต้องส่งกันครั้งละกี่ไบต์ ขึ้นอยู่กับข้อตกลงระหว่างอุปกรณ์แต่ละชนิดจะกำหนดขึ้น โดยในการรับส่งแต่ละครั้ง ตัวแม่จะเป็นตัวควบคุมการรับส่งเองทั้งหมด ซึ่งในกรณีที่ตัวแม่ต้องการติดต่อกับอุปกรณ์หลายๆตัวนั้น หลังจากตัวแม่สร้างสภาวะเริ่มต้นเสร็จแล้ว และทำการติดต่อข้อมูลกับอุปกรณ์เรียบร้อยแล้ว ถ้าตัวแม่ต้องการติดต่อข้อมูลกับอุปกรณ์อื่นไม่จำเป็นต้องสร้างสภาวะสิ้นสุด แต่ตัวแม่สามารถสร้างสภาวะเริ่มต้นขึ้นมาใหม่พร้อมกับส่ง Control Byte ใหม่ไปติดต่อกับอุปกรณ์ที่ต้องการได้ทันที เมื่อทำการติดต่อรับส่งข้อมูลเรียบร้อยแล้วจึงสั่งหยุด Stop Condition เพื่อเป็นการเลิกใช้บัสที่มีอยู่และปล่อยให้เป็นบัสว่าง

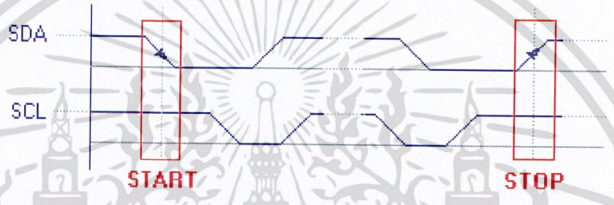
\

สภาวะที่เกิดขึ้นบนบัส I2C มีด้วยกัน 5 สภาวะ ดังนี้

1.) บัสว่าง (Bus not busy) สภาวะนี้เกิดขึ้นเมื่อสถานะลอจิกบนสาย SDA และ SCL เป็นลอจิกสูงทั้งคู่ นั้นหมายความว่า การถ่ายทอดข้อมูลสามารถเริ่มต้นขึ้นได้

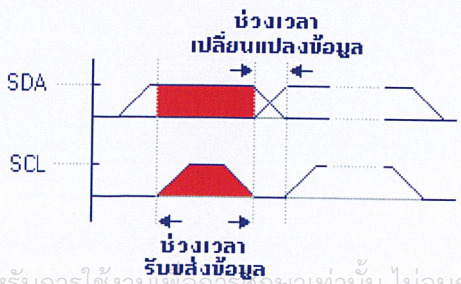
2.) เริ่มต้นการถ่ายทอดข้อมูล (start data transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากสูงไปต่ำ ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่า สภาวะเริ่มต้น (START)

3.) หยุดการถ่ายทอดข้อมูล (stop data transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากต่ำไปสูง ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่า สภาวะหยุด (STOP)



รูปที่ 2.9 เงื่อนไขของสภาวะเริ่มต้น และสิ้นสุดของ I2C Bus [4]

4.) ข้อมูลดำรงอยู่บนบัส (data valid) สภาวะนี้เกิดขึ้นถัดจากสภาวะเริ่มต้น โดยสถานะลอจิกที่เกิดขึ้นบนสาย SDA คือข้อมูลที่ทำการถ่ายทอด เมื่อสาย SCL เป็นลอจิกสูง สถานะที่สาย SDA ต้องคงที่ เพื่อให้อุปกรณ์รับรู้ข้อมูลในจังหวะนั้นว่าเป็น “0” หรือ “1” ข้อมูลอาจเกิดการเปลี่ยนแปลงได้ในขณะที่สาย SCL เป็นลอจิกต่ำ แต่เมื่อใดก็ตามที่ต้องการให้เกิดการถ่ายทอดข้อมูลอย่างสมบูรณ์ สถานะลอจิกที่ขา SDA ต้องคงที่ตลอดช่วงเวลาที่สาย SCL มีสถานะลอจิกสูง หากเกิดการเปลี่ยนแปลงสถานะลอจิกในขณะที่สาย SCL มีลอจิกสูงอยู่นั้น อุปกรณ์มาสเตอร์ที่ทำการควบคุมการถ่ายทอดข้อมูลจะแปลความหมายเป็นสภาวะหยุดหรือสภาวะเริ่มต้นก็ได้ ทำให้ข้อมูลที่ทำการถ่ายทอดเกิดความผิดพลาดขึ้น



รูปที่ 2.10 ช่วงเวลารับส่งบิตข้อมูลของ I2C Bus [4]

5.) รับข้อมูล (acknowledge) เกิดขึ้นหลังจากที่การถ่ายทอดข้อมูลจากตัวส่งมายังตัวรับเกิดขึ้นอย่างสมบูรณ์ โดยตัวส่งจะทำการส่งข้อมูลมา 1 บิตเรียกว่า บิตรับรู้ (acknowledge bit) มีสถานะเป็นลอจิกสูง หลังจากส่งข้อมูลมาครบถ้วน ส่วนอุปกรณ์มาสเตอร์จะทำการส่งสัญญาณรับรู้พิเศษ ซึ่งสัมพันธ์กับสัญญาณนาฬิกา เพื่อตอบสนองบิตรับรู้ที่ส่งมาจาก ตัวส่ง ทางด้านตัวรับจะส่งบิตรับรู้ที่มีสถานะลอจิกต่ำลงบนบัส อุปกรณ์สเลฟที่ถูกอ้างถึงในการติดต่อหรือกำลังติดต่ออยู่ในขณะนั้นก็จะกำเนิดบิตรับรู้เพื่อตอบสนอง ให้ทราบว่าได้รับข้อมูลในแต่ละไบต์เรียบร้อยแล้ว

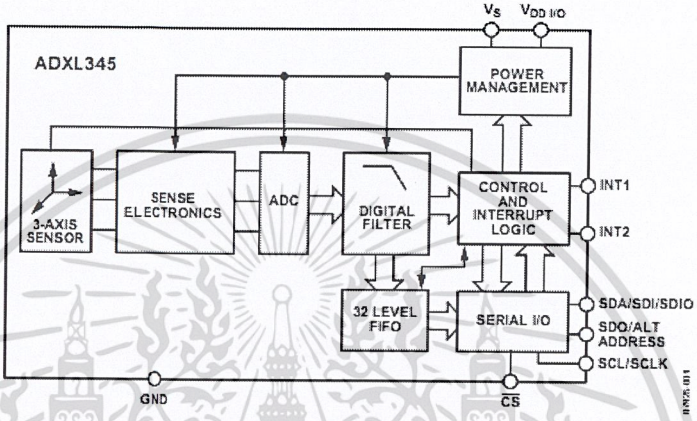
2.6 Accelerometer ADXL345

รูปที่ 2.11 เซนเซอร์วัดความเร่ง ADXL345 [5]

ADXL345 เป็นเซนเซอร์วัดความเร่งสามแกน มีขนาดเล็กบางและใช้กำลังน้อย เซนเซอร์นี้ให้ค่าความละเอียดสูงถึง 16g ซึ่งข้อมูลสามารถส่งได้ถึง 16 บิต สามารถส่งข้อมูลได้ทั้งการส่งข้อมูลแบบ SPI หรือ I2C อีกทั้ง ADXL345 สามารถนำมาประยุกต์ใช้กับ Application ในรูปแบบ Motion ได้ดีเพราะสามารถตอบสนองการเคลื่อนไหวได้อย่างละเอียดสูงถึง 3.9 mg/LSB นั่นหมายความว่าสามารถเปลี่ยนแปลงค่ามุมได้ครั้งละต่ำกว่า 1 องศา

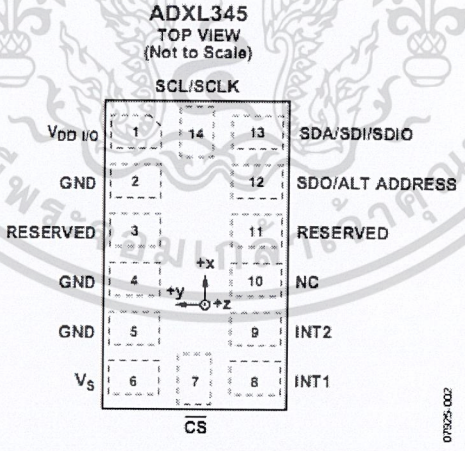
2.6.1 โครงสร้างภายในของ ADXL345

ภายในโครงสร้างของเซนเซอร์ ADXL345 มี ADC เพื่อแปลงจากสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล มีตัวกรองสัญญาณดิจิตอล รวมถึงการส่งข้อมูลสองแบบคือ แบบอินเทอร์รัพท์และแบบ 32 LEVEL FIFO อีกทั้งยังมีการส่งข้อมูลแบบอนุกรมได้ทั้งรูปแบบ SPI BUS และ I2C BUS นอกจากนี้ยังมีตัวควบคุมการเกิดการอินเทอร์รัพท์จากภายนอก และยังมีความสามารถด้านจัดการทรัพยากรที่ใช้ภายในอีกด้วย



รูปที่ 2.12 โครงสร้างการทำงานของ ADXL345 [5]

2.6.2 ฟังก์ชั่นการทำงานของขาต่างๆ ของ ADXL345



รูปที่ 2.13 โครงสร้างของ ADXL345 [5]

ตารางที่ 2.2 ตารางรายละเอียดฟังก์ชั่นขาต่างๆของ ADXL345 [5]

ชื่อ	รายละเอียด	ขา
VDD I/O	ไฟเลี้ยง -0.3 -3.9 volt	1
GND	ขา Ground ต่อสายดิน	2
RESERVED	ขาสำรองไฟ จะต่อเข้า Vs หรือเปิดว่างไว้	3
GND	ขา Ground ต่อสายดิน	4
GND	ขา Ground ต่อสายดิน	5
Vs	ไฟเลี้ยง -0.3 -3.9 volt	6
CS	Chip select	7
INT1	Interrupt 1 Output.	8
INT2	Interrupt 2 Output.	9
NC	ไม่มีการติดต่อ	10
RESERVED	ขาสำรองไฟ จะต่อเข้า Vs หรือเปิดว่างไว้	11
SDO/ALT ADDRESS	Serial Data Output (SPI 4-Wire)/Address สำหรับ I2C	12
SDA/SDI/SDIO	Serial Data(I2C)/Serial Data Input(SPI 4-Wire)/Serial Data Input และ Output (SPI)	13
SCL/SCLK	Serial Data Output (SPI 4-Wire)/Alternate I2C Address Select (I2C)	14

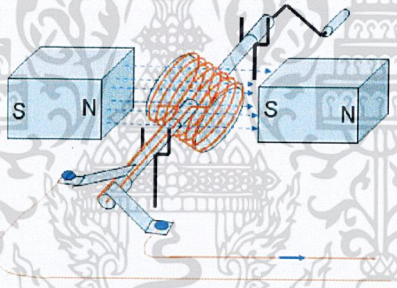
2.6.3 FEATURES

- Ultralow power มีค่าต่ำ 23 μA ในโหมดปกติและ 0.1 μA ในโหมด standby ที่ VS=2.5 V
- Power consumption ควบคุมอัตโนมัติด้วย bandwidth
- ผู้ใช้สามารถเลือกใช้ความละเอียดได้ถึง 16 บิต
- ความละเอียดหลักที่ใช้งาน 10 bit
- Supply voltage range มี 2.0 V ถึง 3.6 V
- I/O voltage range มี 1.7 V ถึง VS
- ส่งข้อมูลผ่าน serial port ใช้ได้ทั้ง SPI (3- และ 4-wire) และ I2C digital interfaces
- Wide temperature อยู่ในช่วง −40°C ถึง +85°C
- 10,000 *g* shock survival

- ขนาดเล็ก 3 mm × 5 mm × 1 mm LGA package
- Bandwidth selectable via serial command
- Flexible interrupt modes mappable to either interrupt pin
- Measurement ranges selectable via serial command
- Wide temperature อยู่ในช่วง −40°C ถึง +85°C

2.7 หลักการพื้นฐานของไดนาโม

เครื่องกำเนิดไฟฟ้าหรือไดนาโม (Generator or dynamo) สามารถเปลี่ยนพลังงานกลเป็นพลังงานไฟฟ้า โดยอาศัยการเหนี่ยวนำในขดลวดของเครื่องกำเนิดไฟฟ้าทำให้เกิดกระแสไฟฟ้าเกิดขึ้น ซึ่งสามารถจัดกระแสไฟได้สองแบบคือ กระแสตรง และกระแสสลับ พลังงานไฟฟ้าจากเครื่องกำเนิดไฟฟ้าเป็นไปตามกฎการอนุรักษ์พลังงาน โดยทั่วไปเครื่องกำเนิดไฟฟ้าจะประกอบด้วยขดลวดเหนี่ยวนำเคลื่อนที่สัมพัทธ์กับแท่งแม่เหล็กทำให้เกิดไฟฟ้า ตัวอย่างการใช้งาน เช่น กรณีไฟฟ้าพลังน้ำ จะมีน้ำตกลงมาหมุนใบพัดของเครื่องกำเนิดไฟฟ้า



รูปที่ 2.14 การทำงานพื้นฐานของไดนาโม [6]

โดยสามารถคำนวณแรงเคลื่อนไฟฟ้าจากเครื่องกำเนิดไฟฟ้าได้ดังนี้

สมมติ ขดลวดมี N รอบ และแต่ละรอบมีพื้นที่ A ให้ขดลวดหมุนด้วยอัตราเร็วเชิงมุม รอบแกนที่ตั้งฉากกับสนามแม่เหล็ก B แรงเคลื่อนไฟฟ้าคือ

$$E = E_o \sin(w.t) \qquad (2.1)$$

จากสมการที่ (2.1) เมื่อ E_o คือแรงเคลื่อนไฟฟ้าสูงสุด

โดย

$$E_o = NBAw \qquad (2.2)$$

2.7.1 หลักการเหนี่ยวนำให้เกิดกระแสไฟฟ้าของไดนาโม

ไดนาโมเป็นอุปกรณ์ที่ทำหน้าที่เปลี่ยนพลังงานกลให้เป็นพลังงานไฟฟ้า มีส่วนประกอบสำคัญ ได้แก่ ขดลวดที่พันอยู่รอบแกน เรียกว่า อาเมเจอร์ (armature) แม่เหล็ก 2 แท่งหันขั้วต่างกันเข้าหากัน เพื่อให้เกิดสนามแม่เหล็กโดยจะมีเส้นแรงแม่เหล็กพุ่งจากขั้วเหนือไปยังขั้วใต้ และบริเวณขั้วจะมีความเข้มของสนามแม่เหล็กมากกว่าบริเวณอื่นๆ

หลักการเหนี่ยวนำให้เกิดกระแสไฟฟ้าจากไดนาโมอาจทำได้ดังนี้ การหมุนขดลวดตัดสนามแม่เหล็กจะทำให้สนามแม่เหล็กเปลี่ยนแปลงจึงเกิดกระแสไฟฟ้าขึ้น ไมเคิล ฟาราเดย์ (Michael Faraday) นักวิทยาศาสตร์ชาวอังกฤษ (พ.ศ.2334-2410) เป็นผู้ค้นพบหลักการที่ว่า “กระแสไฟฟ้าเหนี่ยวนำเกิดจากการเปลี่ยนแปลงสนามแม่เหล็กที่ผ่านขดลวด” หากต้องการสร้างไดนาโมให้สามารถผลิตกระแสไฟฟ้าได้มากขึ้น สามารถทำได้โดยเพิ่มจำนวนรอบของขดลวดและหมุนขดลวดให้เร็วขึ้น

ไดนาโมสามารถแบ่งออกเป็น 2 ชนิดคือ

1. ไดนาโมไฟฟ้ากระแสสลับ AC Dynamo
   ประกอบด้วยแท่งแม่เหล็ก 2 แท่ง ขดลวดและแหวนลื่นโดย แหวนลื่น 2 วงสัมผัสกับแปรงตัวนำไฟฟ้าซึ่งจะรับกระแสไฟฟ้าจากขดลวดออกสู่วงจรภายนอก
2. ไดนาโมไฟฟ้ากระแสตรง DC Dynamo
   ประกอบด้วยแท่งแม่เหล็ก 2 แท่ง ขดลวดและแหวนแยกโดย แหวนแยกแต่ละอันสัมผัสกับแปรงตัวนำไฟฟ้าซึ่งจะรับกระแสไฟฟ้าจากขดลวดออกสู่วงจรภายนอก โดยไดนาโมกระแสตรง (Direct current dynamo) หมายถึง ไดนาโมที่ผลิตไฟกระแสตรง (DC) ส่วนประกอบเหมือนกับไดนาโมกระแสสลับทุกอย่าง ต่างกันที่วงแหวนเท่านั้น

รูปที่ 2.15 ไดนาโมจักรยาน [7]

2.8 วงจรไฟฟ้าอนุกรม

สมการแบ่งแรงดัน (Voltage Divider Equation) หมายถึง วงจรอนุกรมที่ใช้ตัวต้านทานอย่างน้อย 2 ตัวเป็นตัวแบ่งแรงดันไฟฟ้า

$$V_{Rn} = V_T \frac{R_n}{R_T} \tag{2.3}$$

โดย V_{Rn} คือ แรงดันตกคร่อมตัวต้านทานใดๆ

V_T คือ แรงดันของแหล่งจ่ายไฟฟ้า

R_n คือ ตัวต้านทานใดๆ ในวงจร

R_T คือ ค่าความต้านทานรวมของวงจร

2.9 โปรแกรมออกแบบกราฟิก Blender

โปรแกรม Blender เป็นโปรแกรมที่นำเอาความสามารถต่างๆในการสร้างสรรค์ไฟล์รูปแบบสามมิติมารวมกัน เช่น การขึ้นรูปโมเดล , เรนเดอร์ , แอนิเมชั่น ฯลฯ โดย Blender สามารถทำงานได้ในระบบปฏิบัติการที่หลากหลาย และมีขนาดไฟล์ติดตั้งที่ไม่มากนักเมื่อเทียบกับโปรแกรมกราฟิกสามมิติตัวอื่นๆ เนื่องจาก Blender มีรูปแบบซอฟต์แวร์แบบ Open Source เพราะตัวดั้งเดิมถูกสร้างขึ้นโดยบริษัท Not a Number หรือที่รู้จักกันในชื่อของ NaN ในปัจจุบันได้ถูกพัฒนาต่อเนื่องทำให้ Blender เกิดเป็นรูปแบบของ "ซอฟต์แวร์เสรี (Open Source)" ภายใต้ลิขสิทธิ์ GNU/GPL โดยเปิดให้ผู้ใช้สามารถดาวน์โหลดมาใช้งานงานได้ฟรี ซึ่งความสามารถของโปรแกรมถือได้ว่าเทียบเท่ากับโปรแกรมสร้างงานสามมิติทั่วไป รวมถึงมีการทำงานที่ไม่ซับซ้อนและมีเครื่องมือที่ใช้งานง่าย

2.9.1 จุดเด่นของโปรแกรม Blender

โปรแกรม Blender เป็นโปรแกรม Open Source ที่สามารถดาวน์โหลดมาใช้งานได้ฟรี และยังมีความสามารถทัดเทียมกับโปรแกรมสร้างงานสามมิติโปรแกรมอื่นๆโดยจุดเด่นที่น่าสนใจของโปรแกรม Blender มีดังนี้

- เป็นโปรแกรมที่ใช้ทรัพยากรระบบและพื้นที่ในการติดตั้งโปรแกรมน้อย (ไม่เกิน 30 ถึง 40 MB) เมื่อเทียบกับโปรแกรมสามมิติอื่นๆ
- มีความสามารถในการสร้างงานได้หลายรูปแบบ เช่น การสร้างการ์ตูนแอนิเมชั่น งานดีไซน์ งานสถาปัตยกรรมและตกแต่งภายใน การสร้างสเปเชียลเอ็ฟเฟ็กต์และการสร้างเกม เป็นต้น

- ทำงานได้หลายแพลตฟอร์ม รองรับหลายระบบปฏิบัติการ เช่น Windows , Mac , Linux และอื่นๆ
- เป็นโปรแกรมที่มีกลุ่มผู้ใช้งานมากกว่า 250,000 คนทั่วโลก รวมทั้งมีกลุ่มศิลปินที่นิยมใช้ Blender และเปิดเว็บไซต์ให้ความรู้พร้อมเว็บบอร์ดให้สอบถามปัญหาเกี่ยวกับการใช้งานได้ตลอดเวลา เช่น www.blender.org , www.blendernation.com , http://blenderartists.org , http://3d-synthesis.com เป็นต้น
- โปรแกรม Blender สามารถใช้เป็นโปรแกรมพื้นฐานสำหรับผู้ที่ต้องการศึกษาการสร้างงานสามมิติ รวมทั้งบริษัทขนาดเล็กและขนาดกลาง ก็สามารถนำโปรแกรมไปใช้สร้างชิ้นงานตามวัตถุประสงค์ของบริษัทได้เช่นกัน เนื่องจากไม่ต้องจ่ายงบประมาณในการซื้อโปรแกรม และง่ายแก่การเข้าใจ

2.9.2 ความต้องการระบบคอมพิวเตอร์โปรแกรม

โปรแกรม Blender ถูกพัฒนาให้ทำงานได้ดีมากขึ้น มีความยืดหยุ่นของหน้าต่างการทำงาน (GUI หรือ Graphic User Interface) สูง และมีขนาดเบา ดังนั้นความต้องการระบบสำหรับคอมพิวเตอร์จึงไม่สูงมากนัก ซึ่งนี่ถือเป็นข้อดีของ Blender

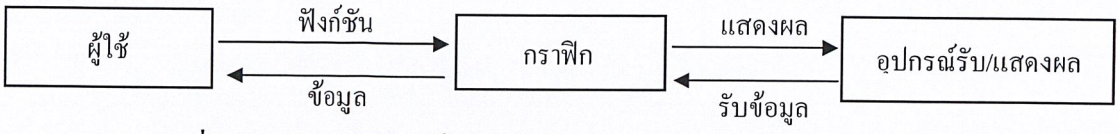
ตารางที่ 2.3 แสดงความต้องการพื้นฐานของระบบคอมพิวเตอร์ที่ต้องมีเมื่อใช้โปรแกรม Blender

ระบบคอมพิวเตอร์	รายละเอียด
- ระบบปฏิบัติการ	Windows 2000,XP,Vista Mac OS X 10.3+,Python 2.3 Suits PowerMac G5,Powerbook G4,iMac G5 Linux glibc 2.3.6,includes FFMPG Suits
- ความเร็วซีพียู	700 MHz หรือสูงกว่า
- RAM	512 MB
- พื้นที่สำหรับลงโปรแกรม	27 MB
- การ์ดแสดงผล	1024×768 16-bit color ,RAM 8 MB (แนะนำให้รองรับมาตรฐาน OpenGL)
- ระบบและอุปกรณ์อื่นๆ	เมาส์แบบ 3 ปุ่ม

2.9.3 ฟังก์ชันทางกราฟิก (Graphic Functions)

การทำงานของระบบกราฟิก ผู้ใช้ไม่จำเป็นต้องทราบว่าการทำงานต้องทำอย่างไร รู้เพียงแต่อินพุต (input) และเอาต์พุต (output) ก็เพียงพอ โดยในระบบกราฟิกจะมีฟังก์ชันที่ถูก

เรียกใช้ โดยมีการรับค่าของอินพุตจากอุปกรณ์ต่างๆ เช่น เมาส์หรือคีย์บอร์ด หรืออุปกรณ์อื่นๆ เช่น แมสเสจจากระบบปฏิบัติการ หรืออินเทอร์รัพท์ก็ได้ ผลลัพธ์ที่ได้จากการทำงานจะแสดงออกไปยังอุปกรณ์ในการแสดงผล เช่น จอภาพ CRT หรือจอ LCD ทั้งนี้เอาต์พุตอาจจะมองอินพุตเป็นฟังก์ชันที่ผู้ใช้เรียกใช้และเอาต์พุตคือ ผลของการทำงานที่แสดงไปยังจอภาพ ดังรูปด้านล่าง



รูปที่ 2.16 ระบบกราฟิกที่เปรียบเสมือนกล่องดำ (Black Box) [8]

2.10 โปรแกรมภาษา Python

Python คือชื่อภาษาที่ใช้ในการเขียนโปรแกรมภาษาหนึ่งที่มีอยู่ในตอนนี้ โดยความสามารถของภาษานี้สูงไม่แพ้ภาษาอื่นๆ Python นั้นเป็นภาษาชนิด Open Source ทำให้ทุกคนสามารถที่จะนำ Python มาพัฒนาโปรแกรมได้ฟรีๆ โดยไม่ต้องเสียค่าใช้จ่าย เนื่องจากความเป็น Open Source จึงทำให้มีคนเข้ามาช่วยกันพัฒนาให้ Python มีความสามารถสูงขึ้น และสามารถใช้งานได้ครอบคลุมกับทุกลักษณะงาน

Python ถูกสร้างขึ้นมาโดย Guido Van Rossumโค้ดของ Python ถูกสร้างขึ้นมาจากภาษาซี การประมวลผลจะทำในแบบอินเทอร์พรีเตอร์คือจะประมวลผลไปทีละบรรทัดและปฏิบัติตามคำสั่งที่ได้รับ Python เวอร์ชันแรกคือเวอร์ชัน 0.9.0 ออกมาเมื่อปี 2533 และเวอร์ชันปัจจุบันคือ 2.5.2 ออกเมื่อวันที่ 21 กุมภาพันธ์ 2551 และได้มีการพัฒนา Python ในรุ่นที่ 3 คือ Python 3000 หรือ Py3k โดยจะมีการปรุงปรุงใหม่เกือบหมด และตอนนี้อยู่ในระหว่างการทดลองอยู่

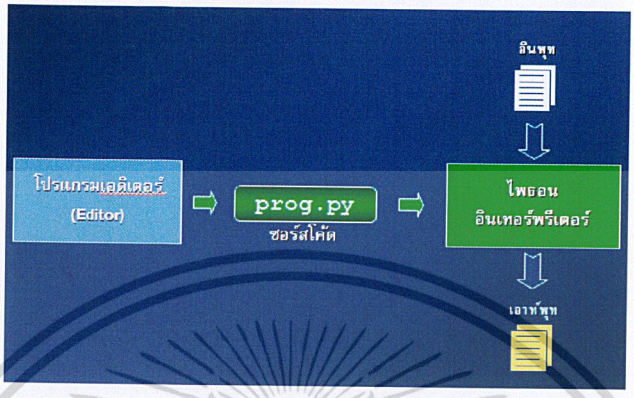
2.10.1 จุดเด่นของโปรแกรมภาษา Python

โปรแกรมภาษา Python เป็นโปรแกรมภาษาแบบ Open Source ที่สามารถดาวน์โหลดมาใช้งานได้ฟรี และยังมีความสามารถทัดเทียมกับโปรแกรมภาษาอื่นๆ โดยจุดเด่นที่น่าสนใจของโปรแกรมมีดังนี้

- ภาษา Python สนับสนุนแนวคิดแบบออปเจกต์โอเรียนเทด หรือ OOP (Object Oriented Programming)
- Python เป็นภาษาคอมพิวเตอร์ที่ไม่คิดมูลค่าการใช้งานและเป็นภาษาที่มีความยืดหยุ่นสูงมาก

- โค้ดที่เขียนด้วย Python สามารถนำไปรันบนระบบปฏิบัติการอื่นๆ ได้ เช่น Linux, Ms-windows (95, 98, NT, 2000, XP), Amiga, Be-OS, OS/2, VMS, QNX, และระบบอื่นๆอีกมากมาย
- Python สนับสนุนเทคโนโลยี COM ของ Ms-windows
- Python รวมมาตรฐานการอินเตอร์เฟส Tkinter ซึ่งสนับสนุนบนระบบ Windows, Ms-windows และ Macintosh การใช้คำสั่ง Tkinter API ช่วยให้โปรแกรมเมอร์ไม่ต้องแก้ไขโค้ดเมื่อนำไปรันบนระบบปฏิบัติการอื่นๆ
- Python เป็น Dynamic typing คือ สามารถเปลี่ยนชนิดข้อมูลได้ง่ายและสะดวก
- Python มี Buil-in Object Types คือ โครงสร้างของข้อมูลที่สามารถใช้ได้ใน Python ประกอบด้วย ลิสต์, ดิกชันนารี, สตริง ที่ง่ายต่อการใช้งานและมีประสิทธิภาพสูง
- Pythonมีเครื่องมือต่างๆ มากมาย เช่น การประมวลผลเท็กซ์ไฟล์ การเรียงข้อมูล การเชื่อมต่อสตริง การตรวจสอบเงื่อนไขของข้อความ การแทนคำ เป็นต้น
- Python มีโมดูลสำหรับจัดการ Regular Expresion
- Python มีโมดูลที่สร้างขึ้นจากนักพัฒนาสนับสนุนมากมาย ได้แก่ COM, Image, CORBA, ORBs, XML เป็นต้น
- Python จัดการหน่วยความจำอย่างอัตโนมัติ สามารถจัดการพื้นที่หน่วยความจำที่ไม่ต่อเนื่องให้ทำงานได้อย่างมีประสิทธิภาพ
- Python อนุญาตให้ฝังชุดคำสั่งของ Python เอาไว้ภายในโค้ดภาษา C/C++ ได้
- Python อนุญาตให้โปรแกรมเมอร์สร้าง Dynamic Link Library (DLL) เพื่อใช้ร่วมกับ Python
- Python มีโมดูลสนับสนุนเกี่ยวกับเน็ตเวิร์ก โปรเซส เธรด regular, expression, xml, GUI และอื่นๆ
- Pythonประกอบด้วยโมดูลสำหรับสร้าง Internet Script และติดต่อกับอินเทอร์เน็ตผ่าน Sockets, และทำหน้าที่เป็น CGI Script ตลอดจนใช้งานคำสั่ง FTP, Glopher, XML และอื่นๆอีกมาก
- Python สามารถประมวลผลทางด้านวิทยาศาสตร์ และวิศวกรรมศาสตร์ได้อย่างมีประสิทธิภาพ
- Python มีฟังก์ชันสนับสนุนฐานข้อมูล เช่น MySQL, Sybase, Oracle, Informix, ODBC และอื่นๆ
- Python มีไลบรารีสนับสนุนด้านการสร้างภาพกราฟิก เช่น ทำภาพเบลอ หรือภาพชัด หรือเขียนข้อความบนภาพ ตลอดจนบันทึกไฟล์ในรูปแบบต่างๆ ได้อย่างสะดวกและมีประสิทธิภาพ

- Python มีไลบรารีสนับสนุนด้านปัญญาประดิษฐ์
- Python มีไลบรารีสำหรับสร้างเอกสาร PDF โดยไม่ต้องติดตั้ง Acrobat Writer
- Python มีไลบรารีสำหรับสร้าง Shockwaves Flash (SWF) โดยไม่ต้องติดตั้ง Macromedia Flash



รูปที่ 2.17 ขั้นตอนการทำงานของโปรแกรม Python [9]

2.10.2 โปรแกรมภาษา Python ในแพลตฟอร์มต่างๆ

ผู้เขียนโปรแกรมภาษา Python สามารถเลือกใช้แพลตฟอร์มได้ดังนี้

CPython (ซีไพทอน) : เป็นแพลตฟอร์มภาษาไพทอนดั้งเดิม โปรแกรมอินเทอร์พรีเตอร์ถูกเขียนโดยภาษาซี ซึ่งคอมไพล์ใช้ได้บนหลายระบบปฏิบัติการ เช่น Windows,Linux,Unix การใช้งานสามารถทำได้โดยการติดตั้งโปรแกรมอินเทอร์พรีเตอร์และแพ็กเกจที่จำเป็นต่างๆ

Jython (ไจทอน) : เป็นแพลตฟอร์มภาษาไพทอนที่ถูกพัฒนาบนแพลตฟอร์มจาวา เพื่อเพิ่มอำนวยความสะดวกในการใช้ความสามารถภาษาสคริปต์ของไพทอนลงในซอฟต์แวร์จาวาอื่นๆ การใช้งานสามารถทำได้โดยการติดตั้งจาวาและเรียนไลบรารีของไจทอนซึ่งมาในรูปไบนารีเพื่อใช้งาน

Python.NET (ไพทอนดอตเน็ต) : เป็นการพัฒนาภาษาไพทอนให้สามารถทำงานบนดอตเน็ตเฟรมเวิร์กของไมโครซอฟท์ได้ โดยโปรแกรมที่ถูกเขียนจะถูกแปลงเป็น CLR ปัจจุบันมีโครงการที่นำภาษาไพทอนมาใช้บน .NET Framework ของไมโครซอฟท์แล้วคือโครงการ IronPython

2.10.3 ไลบรารีภาษา Python

การเขียนโปรแกรมในภาษาไพทอนโดยใช้ไลบรารีต่างๆ เป็นการลดภาระของโปรแกรมเมอร์ได้เป็นอย่างดี ทำให้โปรแกรมเมอร์ไม่ต้องเสียเวลากับการเขียนคำสั่งที่ซ้ำๆ เช่น การแสดงผลข้อมูลออกสู่หน้าจอ หรือการรับค่าต่างๆ

ไพทอนมีชุดไลบรารีมาตรฐานมาให้ตั้งแต่อินเตอร์พรีเตอร์ นอกจากนั้นยังมีผู้พัฒนาจากทั่วโลกดำเนินการพัฒนาไลบรารีซึ่งช่วยอำนวยความสะดวกในด้านต่างๆ โดยจะเผยแพร่ในรูปแบบของแพ็กเกจต่างๆ ซึ่งสามารถติดตั้งเพิ่มเติมได้ ดังนี้

- wxPython : อีกทางเลือกหนึ่งสำหรับเขียนส่วนติดต่อกับผู้ใช้กับกราฟิก ซึ่งสามารถใช้ได้หลายระบบปฏิบัติการ
- SciPy : รวมโครงสร้างข้อมูลและการคำนวณต่าง ๆ ที่จำเป็นต้องใช้ในการเขียนโปรแกรมคำนวณทางวิทยาศาสตร์
- py2exe : ใช้สำหรับแปลงโปรแกรมที่เขียนในภาษาไพทอนให้อยู่ในรูปแบบของระบบปฏิบัติการวินโดวส์
- PyWin32 : ใช้สำหรับติดต่อเรียกใช้บริการบนระบบปฏิบัติการวินโดวส์และคลาสใน Microsoft Foundation Classes: MFC
- MySQLdb : ใช้สำหรับติดต่อกับระบบฐานข้อมูล MySQL
- psycopg2 : ใช้สำหรับติดต่อกับระบบฐานข้อมูล โพสต์เกรสคิวเอล
- PyGTK : GTK+ สำหรับ Python ใช้สำหรับสร้างส่วนติดต่อกับผู้ใช้แบบกราฟิก ซึ่งสามารถใช้ได้หลายระบบปฏิบัติการ
- PyQt : Qt สำหรับ Python ใช้สำหรับสร้างส่วนติดต่อกับผู้ใช้แบบกราฟิก ซึ่งสามารถใช้ได้หลายระบบปฏิบัติการ

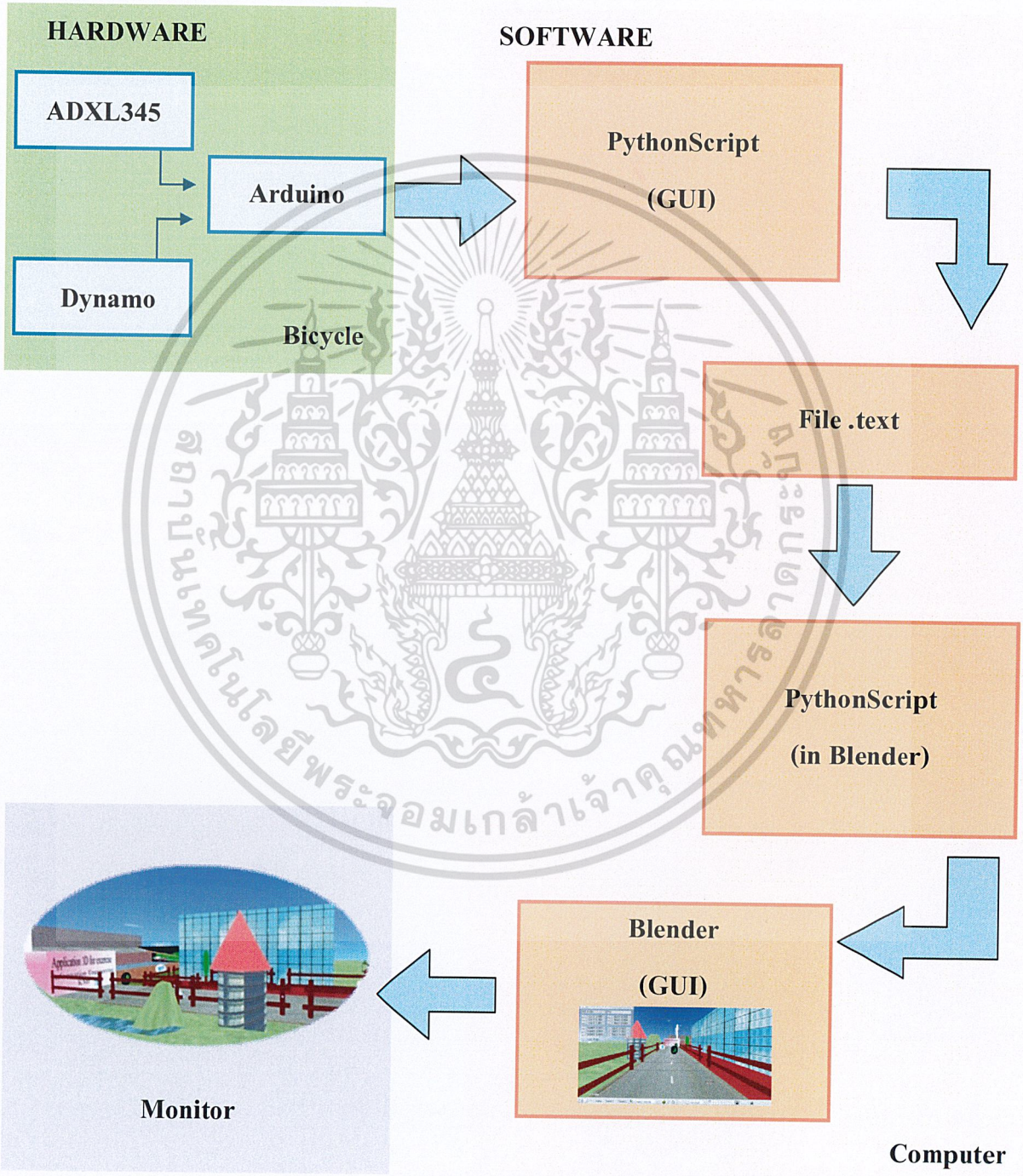
2.10.4 ตัวอย่างการนำไปใช้ของโปรแกรมภาษา Python

จะเห็นได้ว่าจุดเด่นของโปรแกรมภาษา Python มีอยู่มากและใช้งานได้อย่างรวดเร็ว โดยโปรแกรมภาษา Python ได้ถูกนำไปใช้ประโยชน์ในต่างๆมากมาย เช่น

- Red Hat ใช้ Python เป็นเครื่องมือสำหรับการติดตั้งระบบปฏิบัติการ Linux (Installer)
- Google สร้างขึ้นด้วย Python
- Infoseek ใช้ Python จัดการ web search products
- Yahoo! ใช้ Python ในการจัดการด้าน Internet services
- NASA ใช้ Python สำหรับ mission-control-system
- Lawrence Livermore Labs ใช้ Python สำหรับงาน numeric programming
- Industrial Light and Magic ใช้ Python สร้างภาพแอนิเมชัน

บทที่ 3
การออกแบบและการสร้าง

3.1 หลักการทำงานของระบบโดยรวม



รูปที่ 3.1 หลักการทำงานของระบบโดยรวม

จากรูปที่ 3.1 การทำงานของระบบโดยรวมจะแบ่งเป็น 2 ส่วนใหญ่ๆ คือ ส่วนของชุดอุปกรณ์และส่วนของโปรแกรมกราฟิกสามมิติ

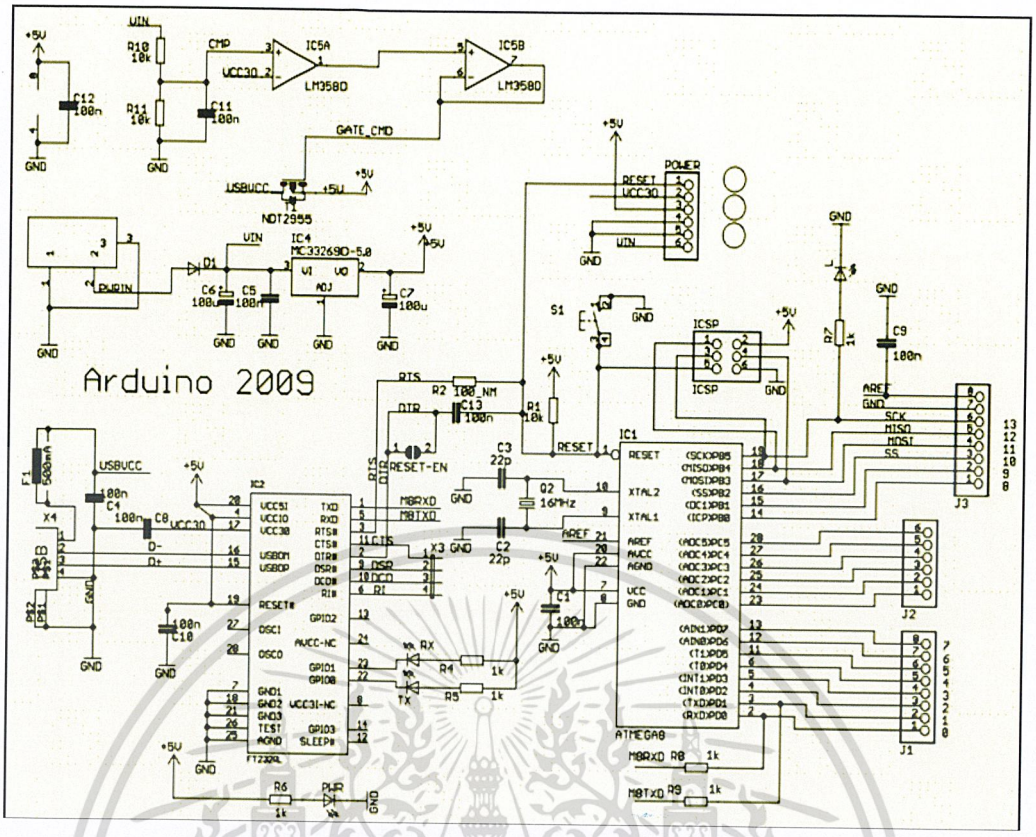
โดยการทำงานคือ ชุดโมดูลบอร์ดไมโครคอนโทรลเลอร์ Arduino จะทำหน้าที่ควบคุมการทำงานของอุปกรณ์ ADXL345(Accelerometer) เซนเซอร์วัดความเร่งแนว 3 แกน (แกน X, Y, Z) ที่ใช้สำหรับตรวจวัดการเคลื่อนไหวศีรษะของผู้ออกกำลังกายเพื่อให้สามารถบังคับซ้ายขวาตามความต้องการของผู้เล่นได้ และสำหรับการเคลื่อนที่ภายในเกมจะเคลื่อนที่ตามการปั่นจักรยานของผู้ออกกำลังกาย โดยมีไดนาโมจักรยานทำหน้าที่ส่งค่าอนาล็อกให้กับชุดโมดูลบอร์ดไมโครคอนโทรลเลอร์ Ardunio เมื่อชุดโมดูลบอร์ดดังกล่าวได้รับค่าข้อมูลจึงจะส่งค่าไป(โดยซอฟแวร์ที่ใช้เป็นภาษา PythonScript และค่าที่ได้รับมาจะมี 4 ค่านั่นคือ X, Y, Z และ Dynamo) เก็บไว้ในรูปแบบไฟล์ .text เพื่อให้สามารถนำค่าข้อมูลดังกล่าวไปประมวลผลในส่วนของโปรแกรมกราฟิกสามมิติที่ถูกออกแบบขึ้น โดยใช้โปรแกรม Blender3D และควบคุมการใช้งานต่างๆของเกมด้วยโปรแกรมภาษา PythonScript และส่งค่าให้กับ Monitor แสดงผลการทำงานของโปรแกรมกราฟิกสามมิติต่อไป

3.2 ส่วนของชุดอุปกรณ์

อุปกรณ์จะต้องอาศัยการติดต่อกันระหว่าง ชุดโมดูลบอร์ดไมโครคอนโทรลเลอร์ Arduino, ADXL345(Accelerometer) และไดนาโมจักรยาน โดยหลักการต่างๆมีดังต่อไปนี้

3.2.1 ชุดโมดูลบอร์ดไมโครคอนโทรลเลอร์ Arduino

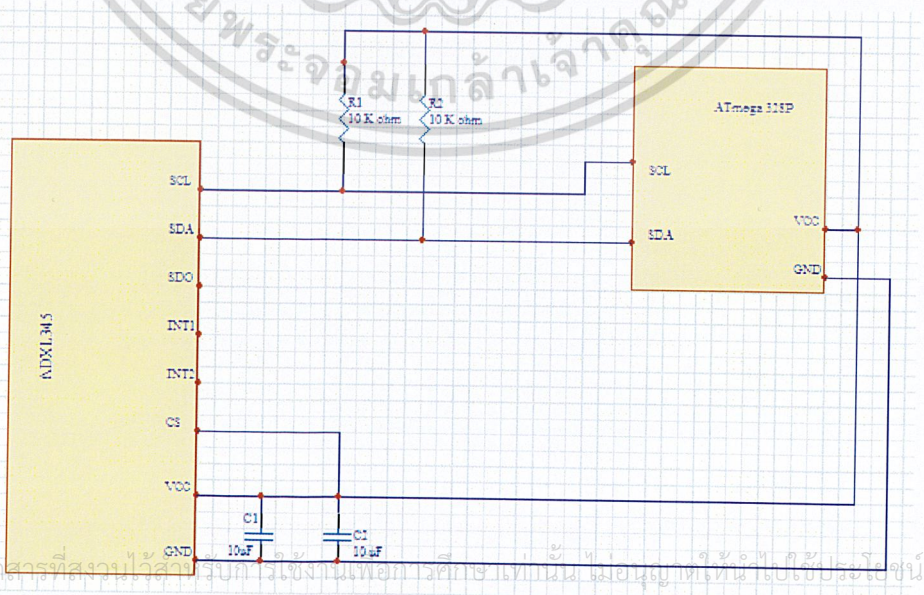
ชุดโมดูลบอร์ดไมโครคอนโทรลเลอร์ Arduino เป็นบอร์ดทดลองสำเร็จรูปโดยใช้ MCU ในตระกูล AVR เบอร์ ATmega 328 สามารถส่งข้อมูลได้รวดเร็วและมีช่องสำหรับต่อกับอุปกรณ์ภายนอกได้อย่างสะดวก อีกทั้งสามารถใช้ไฟเลี้ยงจากการต่อพอร์ตUSB ได้อีกด้วย



รูปที่ 3.2 วงจรภายในชุดโมดูลบอร์ดไมโครคอนโทรลเลอร์ Arduino Duemilanove [1]

3.2.2 ADXL345 (Accelerometer)

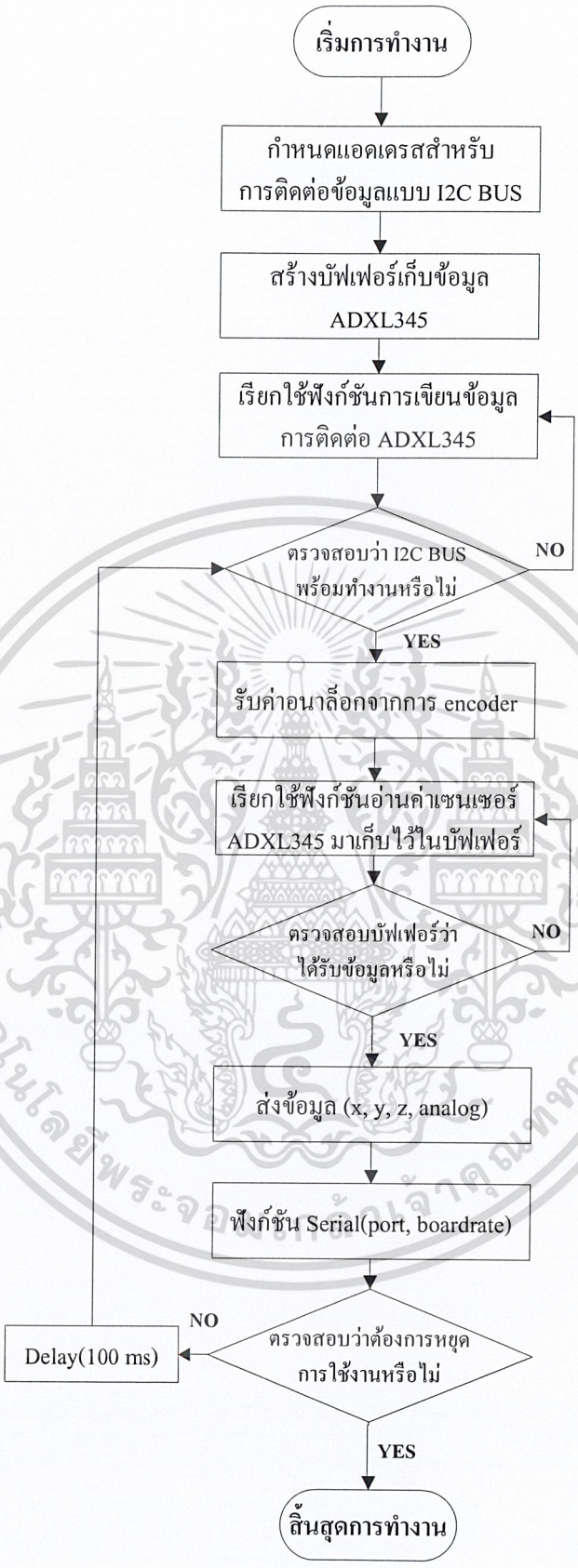
การติดต่อเซนเซอร์ ADXL345 กับบอร์ด Arduino Duemilanove ใช้หลักการส่งข้อมูลแบบ I2C Bus ดังนั้นการออกแบบวงจรของเซนเซอร์จึงใช้เพียงสาย SDA กับ SCL ในการส่งข้อมูลระหว่างบอร์ด Arduino Duemilanove โดยใช้ไฟเลี้ยง 3.3 v จากบอร์ดได้ด้วยเช่นกัน



รูปที่ 3.3 การออกแบบการเชื่อมต่อข้อมูลแบบ I2C Bus กับบอร์ดทดลอง Arduino

3.2.3 หลักการทำงานของ Arduino ในการส่งข้อมูล

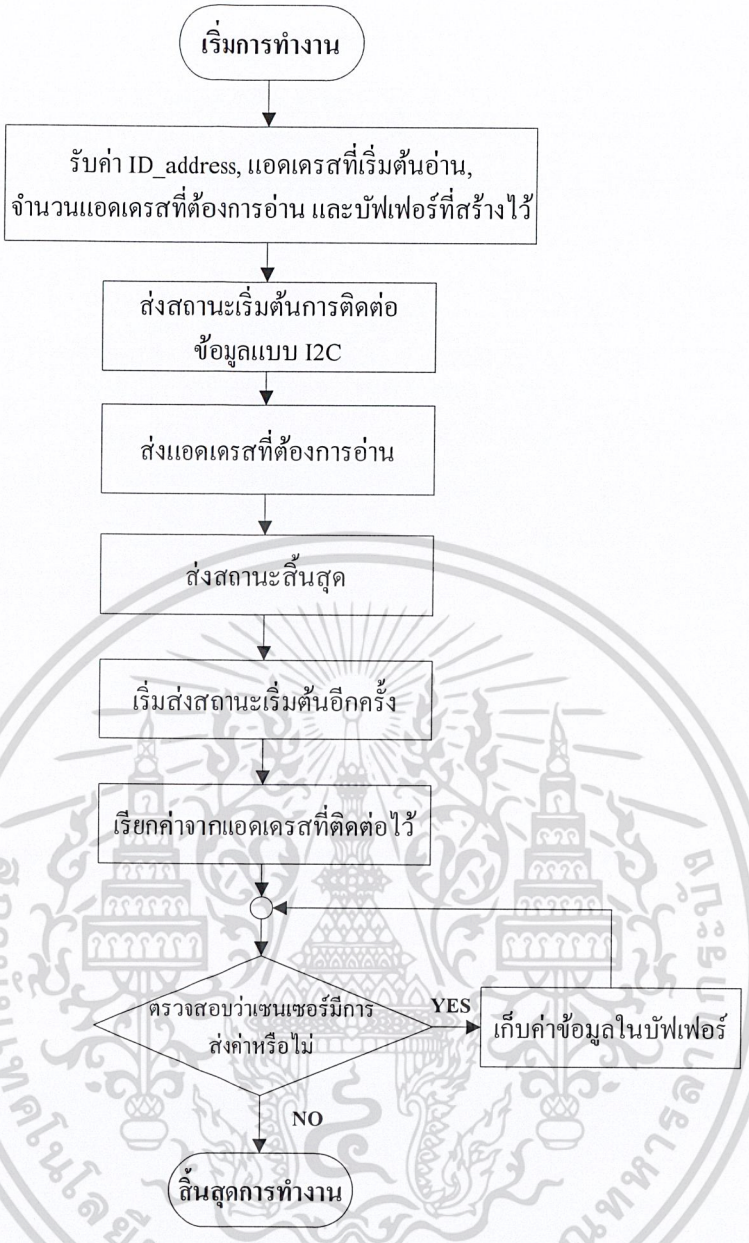
1. กำหนดแอดเดรสสำหรับการติดต่อข้อมูลของ Arduino โดยอาศัยการส่งแบบ I2C Bus ให้กำหนดพอร์ตอนาล็อกสำหรับรับค่าอินพุทจากไดนาโมจักรยาน
2. สร้างบัฟเฟอร์ขึ้นเพื่อสำหรับเก็บข้อมูลที่ได้รับจาก ADXL345(Accelerometer) โดยข้อมูลดังกล่าวจะถูกนำเก็บเพื่อนำไปใช้งาน
3. เรียกใช้ฟังก์ชันการเขียนข้อมูลเพื่อให้อุปกรณ์โมดูลบอร์ด Arduino ติดต่อกับ ADXL345
4. ตรวจสอบว่าการส่งข้อมูลในรูปแบบ I2C BUS พร้อมทำงานหรือไม่
   - ถ้าการส่งข้อมูลในรูปแบบ I2C BUS ไม่พร้อมทำงานจะทำการเรียกใช้ฟังก์ชันการเขียนข้อมูลติดต่อ ADXL345 (ในข้อ 3) อีกครั้ง
   - ถ้าการส่งข้อมูลในรูปแบบ I2C BUS พร้อมทำงานจะรับค่าอนาล็อกจากไดนาโมจักรยาน เพื่อส่งข้อมูลดังกล่าวต่อไป
5. ทำการรับข้อมูลชนิดอนาล็อกจากไดนาโมจักรยาน
6. เรียกใช้ฟังก์ชันการอ่านค่าข้อมูลจากเซนเซอร์ ADXL345 (ในหัวข้อ 3.2.4) และนำค่าข้อมูลดังกล่าวมาเก็บไว้ในบัฟเฟอร์
7. ตรวจสอบบัฟเฟอร์ว่าได้รับข้อมูลเรียบร้อยหรือไม่
   - ถ้าตรวจสอบบัฟเฟอร์พบว่ายังไม่มีการรับข้อมูล จะย้อนกลับไปเพื่อเรียกใช้ฟังก์ชันการอ่านค่าข้อมูลจากเซนเซอร์ ADXL345 มาเก็บในบัฟเฟอร์ (ในข้อ 6) อีกครั้ง
   - ถ้าตรวจสอบบัฟเฟอร์พบว่ามีการส่งข้อมูลเกิดขึ้น จะส่งข้อมูล (x, y, z, Analog) และฟังก์ชัน Serial (Port, Board rate) ออกไปใช้งาน
8. ตรวจสอบว่าต้องการหยุดการใช้งานหรือไม่
   - ถ้าตรวจสอบว่าไม่ต้องการหยุดการใช้งาน จะเพิ่มค่า delay เป็นขนาด 100 ms
   - ถ้าตรวจสอบว่าต้องการหยุดการใช้งาน จะสิ้นสุดการทำงาน



รูปที่ 3.4 Flow chat หลักการทำงานของ Arduino ในการส่งข้อมูล

3.2.4 หลักการทำงานของ Arduino ในการอ่านค่าข้อมูลจากเซนเซอร์ ADXL345

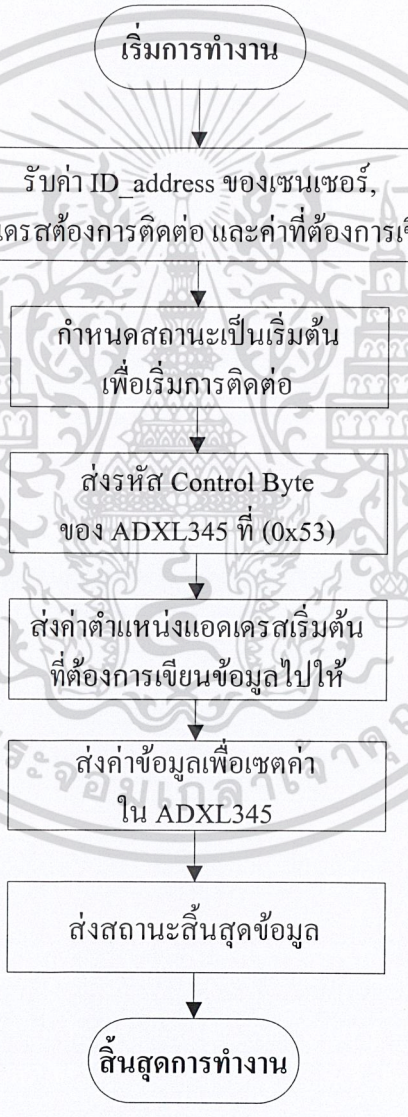
1. เริ่มต้นการทำงาน โดยรับค่า ID_Address, แอดเดรสที่เริ่มต้นอ่าน, จำนวนแอดเดรสที่ต้องการอ่าน และบัฟเฟอร์ที่สร้างไว้เพื่อเข้ามารองรับแอดเดรสที่อ่านได้
2. ส่งสถานะเริ่มต้นการติดต่อข้อมูลแบบ I2C
3. ส่งแอดเดรสที่ต้องการอ่าน
4. ส่งสถานะสิ้นสุด
5. เริ่มส่งสถานะเริ่มต้นใหม่อีกครั้ง
6. เรียกค่าจากแอดเดรสที่ติดต่อไว้
7. ตรวจสอบว่าเซนเซอร์ส่งค่าข้อมูลหรือไม่
    - ถ้าตรวจสอบเซนเซอร์พบว่ามีการส่งค่าข้อมูล จะทำการส่งค่าไปเก็บในบัฟเฟอร์ที่สร้างไว้ และกลับไปตรวจสอบเซนเซอร์อีกครั้ง
    - ถ้าตรวจสอบเซนเซอร์พบว่าไม่มีการส่งค่าข้อมูล จะสิ้นสุดการทำงาน



รูปที่ 3.5 Flow chat หลักการอ่านค่าข้อมูลของเซนเซอร์ ADXL345

3.2.5 หลักการทำงานของ Arduino ในการเขียนค่าข้อมูลไปยังเซนเซอร์ ADXL345

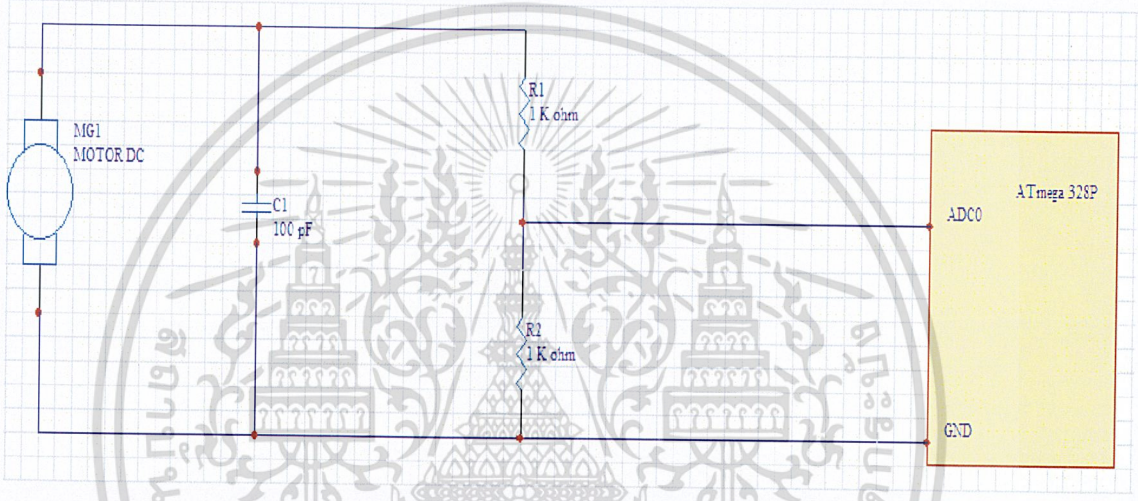
1. เริ่มต้นการทำงานโดยรับค่า ID_address ของเซนเซอร์, แอดเดรสที่ต้องการติดต่อ และค่าที่ต้องการเขียน
2. กำหนดสถานะเป็นเริ่มต้น เพื่อเริ่มการติดต่อสื่อสารระหว่าง Arduino และ ADXL345
3. ส่งรหัส Control Byte ของ ADXL345 ไปที่ (0x53)
4. ส่งค่าตำแหน่งแอดเดรสที่ต้องการเขียนข้อมูลติดต่อภายใน slave
5. ส่งค่าข้อมูลเพื่อปรับค่าใน ADXL345
6. ส่งสถานะสิ้นสุดการทำงาน



รูปที่ 3.6 Flow chat หลักการเขียนค่าข้อมูลของเซนเซอร์ ADXL345

3.2.6 หลักการทำงานของ Arduino ในการอ่านค่าข้อมูลที่ได้รับจากไดนาโมจักรยาน

ในส่วนนี้เป็นการออกแบบการติดต่อระหว่าง Arduino และไดนาโม หลักการคือ ใช้ตัวเก็บประจุแบบไม่มีขั้วต่อคร่อมกันระหว่างไดนาโมจักรยาน เพื่อป้องกันแรงดันที่อาจมีค่าสูง และใช้ตัวต้านทานสองตัวซึ่งมีค่าตัวละ 1K ต่อแบบอนุกรม เพื่อช่วยในการลดค่าแรงดันจากไดนาโมจักรยานให้เหลือเพียง 5 โวลต์ เหตุผลที่สร้างวงจรดังกล่าว เนื่องจากหากปั่นจักรยานจะเกิดการหมุนของไดนาโมจักรยานเกิดขึ้น เมื่อความเร็วในการปั่นสูงขึ้นจะเกิดแรงดันที่ทำให้ส่งเข้าไปในบอร์ดไมโครคอนโทรลเลอร์ Arduino สูงเกินกว่าบอร์ดไมโครคอนโทรลเลอร์จะรับได้ และส่งผลต่ออาจทำให้บอร์ดดังกล่าวเสียหาย

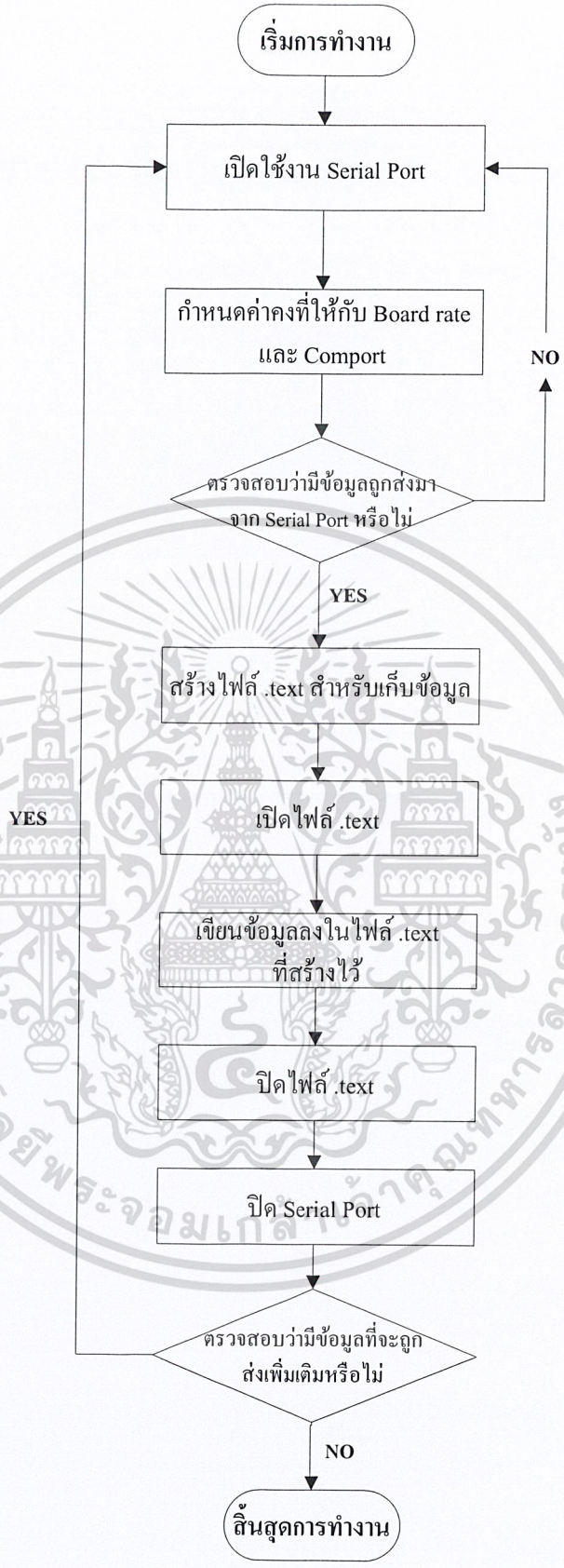


รูปที่ 3.7 วงจรไดนาโมจักรยานต่อเข้าสู่บอร์ด Arduino

3.3 ส่วนของโปรแกรมกราฟิกสามมิติ

3.3.1 ส่วนของการรับข้อมูลของโปรแกรมภาษา Python จากไมโครคอนโทรลเลอร์บอร์ด Arduino โดยส่งผ่านทาง Serial Port

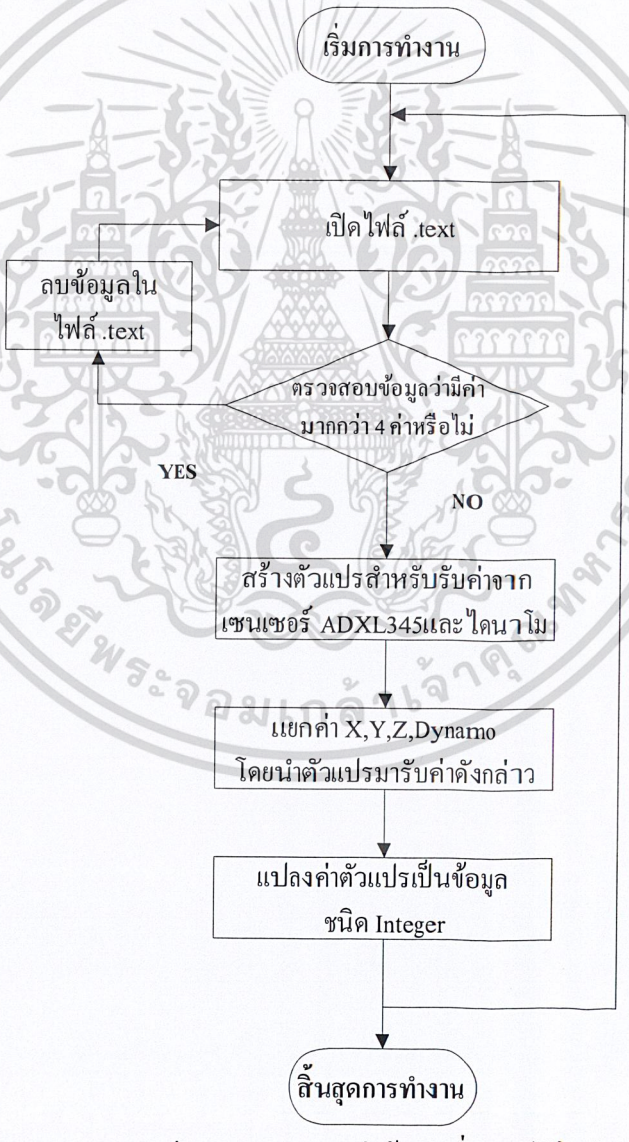
1. เริ่มต้นการทำงานโดยเปิดใช้งาน Serial Port
2. กำหนดค่าคงที่ให้กับ Board rate และ Comport สำหรับรับค่าจาก Serial Port
3. ตรวจสอบว่ามีข้อมูลถูกส่งมาจาก Serial Port หรือไม่
    - ถ้าตรวจสอบพบว่าส่งข้อมูลไม่ถูกส่งมา จะทำการส่งค่าว่างเปล่า(ค่า null)กลับไปเพื่อรอรับข้อมูลใหม่ที่จะเข้ามา (ในข้อ 1) อีกครั้ง
    - ถ้าตรวจสอบพบว่ามีการส่งข้อมูลเข้ามา จะนำค่าข้อมูลดังกล่าวเก็บไว้ในไฟล์ .text
4. สร้างไฟล์ .text เพื่อเก็บค่าข้อมูลที่ถูกส่งมาจาก Serial Port
5. เปิดไฟล์ .text
6. เขียนข้อมูลลงในไฟล์ .text ที่สร้างไว้
7. ปิดไฟล์ .text
8. ปิด Serial Port
9. ตรวจสอบว่าต้องการรับค่าจาก Serial Port เพิ่มเติมหรือไม่
    - ถ้าตรวจสอบพบว่าต้องการรับค่าข้อมูลเพิ่มเติมจะทำการวนลูป(ไปที่ข้อ 1) เพื่อทำการรับค่าข้อมูลใหม่
    - ถ้าตรวจสอบพบว่าไม่ต้องการรับค่าข้อมูลเพิ่มเติม จะสิ้นสุดการทำงาน



รูปที่ 3.8 Flow chat ส่วนของการรับข้อมูลโปรแกรมภาษา Python รับค่าข้อมูลจาก ไมโครคอนโทรลเลอร์บอร์ด Arduino โดยส่งผ่านทาง Serial Port

3.3.2 ส่วนของการแยกค่าข้อมูลที่ส่งมาไว้ในไฟล์ .text

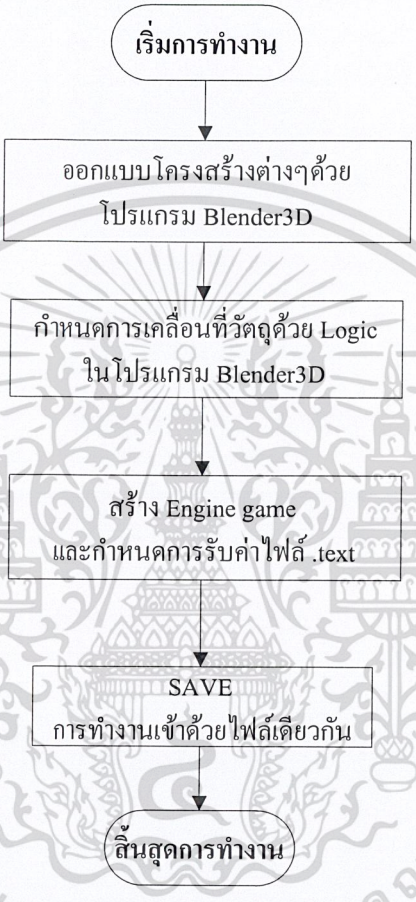
1. เปิดไฟล์ .text ที่ได้จัดเก็บค่าไว้
2. ตรวจสอบข้อมูลที่เข้ามาว่ามีค่ามากกว่า 4 ค่าต่อการส่งข้อมูล 1 ครั้งหรือไม่
   - ถ้าตรวจสอบพบว่าค่าที่ส่งมามีค่ามากกว่า 4 ค่า จะทำการลบข้อมูลในไฟล์ .text และรอค่าข้อมูลใหม่
   - ถ้าตรวจสอบพบว่าค่าที่ส่งมามีค่าเท่ากับ 4 ค่า จะทำการกำหนดตัวแปรมารับค่าดังกล่าวเป็นค่า dX, dY, dZ, Dynamo
3. สร้างตัวแปรสำหรับรับค่าข้อมูลจากเซนเซอร์และไดนาโมจักรยาน
4. แยกค่า X, Y, Z, Dynamo ออกจากกันและนำตัวแปรที่สร้างขึ้นมารับค่าข้อมูล
5. แปลงค่าข้อมูลดังกล่าวเป็น Integer



รูปที่ 3.9 Flow chat ส่วนของการแยกค่าข้อมูลที่ส่งมาไว้ในไฟล์ .text

3.3.3 ส่วนของการออกแบบรูปทรงสามมิติ

1. ออกแบบโครงสร้างต่างๆด้วยโปรแกรม Blender3D
2. กำหนดการเคลื่อนที่เบื้องต้นด้วย Logic ในโปรแกรม Blender3D
3. สร้าง Engine game และกำหนดให้มีการรับค่าจากไฟล์ .text ด้วยโปรแกรมภาษา Python
4. Save โปรแกรม Blender3D และโปรแกรมภาษา Python ลงในไฟล์เดียวกัน



รูปที่ 3.10 Flow chat ส่วนของการออกแบบรูปทรงสามมิติ

บทที่ 4
การทดลองและผลการทดลอง

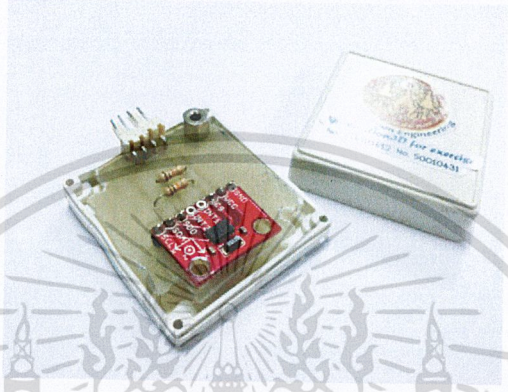
เนื้อหาในบทนี้จะกล่าวถึงรายละเอียดขั้นตอนการทดลองอ่านค่าข้อมูลจากเซนเซอร์วัดความเร่ง ADXL345 และไดนาโมจักรยาน โดยใช้บอร์ดไมโครคอนโทรลเลอร์ Arduino จากนั้นจึงส่งข้อมูลไปประมวลผลยัง โปรแกรมกราฟิกสามมิติ (Blender3D) และควบคุมการทำงานโดยโปรแกรมภาษา Python โดยมีรายละเอียดการทดลองดังนี้

4.1 การทำงานของอุปกรณ์โดยรวม

4.1.1 ส่วนของอุปกรณ์

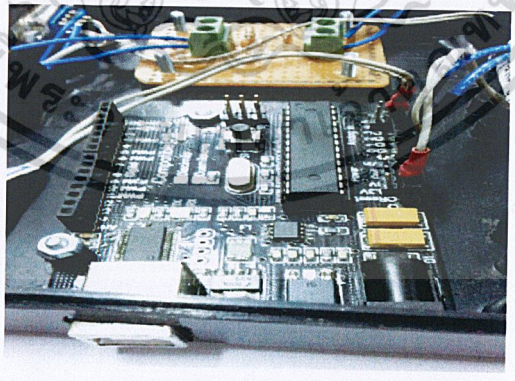
รูปที่ 4.1 ส่วนของอุปกรณ์

หมายเลข 1 คือเซนเซอร์วัดความเร่ง ADXL345 โดยจะมีการติดตั้งให้เชื่อมต่อกับบอร์ดไมโครคอนโทรลเลอร์ Arduino (หมายเลข 2) และส่งค่าที่ได้ในตามแกน X, Y, Z ด้วยหลักการการส่งข้อมูลแบบ I2C Bus ผ่านทาง ขา SCL (PC.5) และ SDA (PC.4) โดยเซนเซอร์วัดความเร่ง ADXL345 มีความสัมพันธ์กับโปรแกรมกราฟิกสามมิติในกรณีที่ผู้เล่นต้องการควบคุมซ้าย–ขวา



รูปที่ 4.2 ADXL345 พร้อมกล่องสำหรับติดตั้งบนศีรษะ

หมายเลข 2 คือบอร์ดไมโครคอนโทรลเลอร์ Arduino ที่มีการติดตั้งให้เชื่อมต่อกับเซนเซอร์วัดความเร่ง ADXL345(หมายเลข 1) และไดนาโมจักรยาน(หมายเลข 3) เพื่อรับค่าแกน X, Y, Z (จากเซนเซอร์วัดความเร่ง ADXL345) และรับค่าจากความดันในการปั่นจากไดนาโมจักรยาน(โดยแรงดันที่จ่ายเข้ามามีค่าระหว่าง 6-12 โวลต์)



รูปที่ 4.3 บอร์ดไมโครคอนโทรลเลอร์ Arduino

หมายเลข 3 คือ ไดนาโมจักรยาน โดยมีการติดตั้งให้เชื่อมต่อกับบอร์ดไมโครคอนโทรลเลอร์ Arduino (หมายเลข 2) มีสายเชื่อมต่อไปยังบอร์ดดังกล่าว 2 เส้นคือ GND และขา ADC0 ก่อนการนำค่าข้อมูลเข้าไปในบอร์ดไมโครคอนโทรลเลอร์ Arduino ต้องผ่านวงจรเพื่อป้องกันการกระชากของไฟและยังช่วยกรองแรงดัน ไดนาโมจักรยานจะนำมาใช้สำหรับการปั่นเพื่อให้เกิดแรงดันและนำไปใช้ในบอร์ดไมโครคอนโทรลเลอร์ Arduino เพื่อส่งค่าดังกล่าวให้ประมวลผลทางโปรแกรมกราฟิกสามมิติ การนำค่าดังกล่าวไปใช้มีความสัมพันธ์กับโปรแกรมกราฟิกสามมิติ ในกรณีที่ผู้เล่นต้องการเคลื่อนที่ ผู้เล่นสามารถปั่นจักรยานควบคุมการเคลื่อนที่เองได้

รูปที่ 4.4 ไดนาโมจักรยานและวงจรกรองแรงดัน

หมายเลข 4 คือคอมพิวเตอร์ ที่ใช้สำหรับรับค่าข้อมูลจากบอร์ดไมโครคอนโทรลเลอร์ Arduino และนำค่าข้อมูลดังกล่าวมาใช้ในการประมวลผลในโปรแกรมเกมสามมิติ สุดท้ายผลดังกล่าวจะแสดงออกมาทางมอนิเตอร์(หน้าจอแสดงผล)

รูปที่ 4.5 หน้าโปรแกรมเกมสามมิติที่แสดงออกมาทางหน้าจอแสดงผล

4.2 การทดสอบการทำงานของบอร์ดไมโครคอนโทรลเลอร์ Arduino Duemilanove

การทดสอบการทำงานของบอร์ดไมโครคอนโทรลเลอร์ Arduino Duemilanove ด้วยการนำโปรแกรมตัวอย่างมาคอมไพล์ และ อัพโหลดโปรแกรมไปยังโมดูล

4.2.1 การเปิดโปรแกรม Arduino

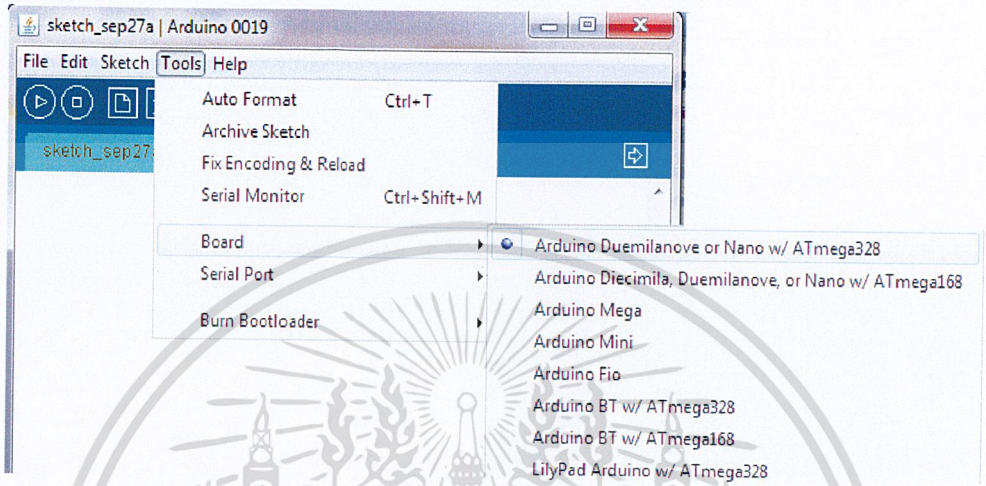
ทำการเปิดโปรแกรมโดยเข้าไปที่ โปรแกรม Arduino จะขึ้นมาดังรูปที่ 4.6

รูปที่ 4.6 แสดงหน้าต่างของโปรแกรม Arduino

4.2.2 กำหนดค่าเบื้องต้นสำหรับใช้งานกับ บอร์ดไมโครคอนโทรลเลอร์ Arduino

4.2.2.1 การเลือกบอร์ดไมโครคอนโทรลเลอร์

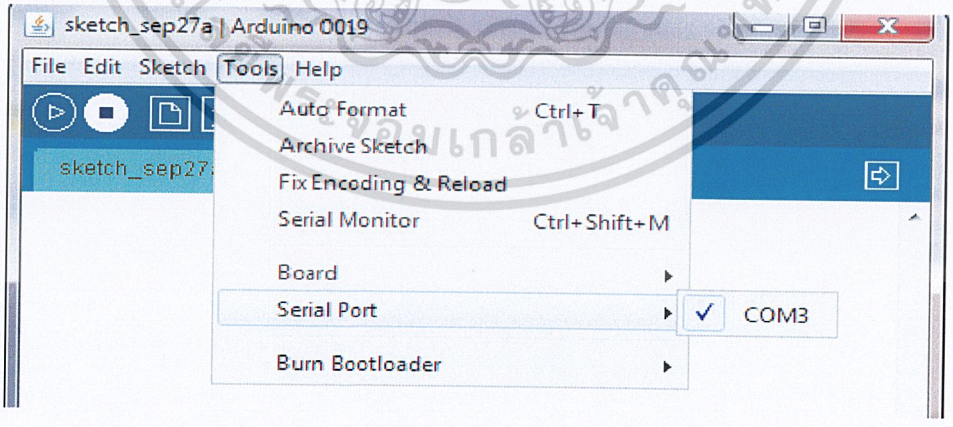
เลือกเมนู Tool>>Board>>Arduino Duemilanove or Nano w /ATmega328 ดังรูปที่ 4.7



รูปที่ 4.7 การเลือกไมโครคอนโทรลเลอร์ของ Arduino

4.2.2.2 การเลือกพอร์ตอนุกรมที่ใช้ในการติดต่อกับบอร์ด

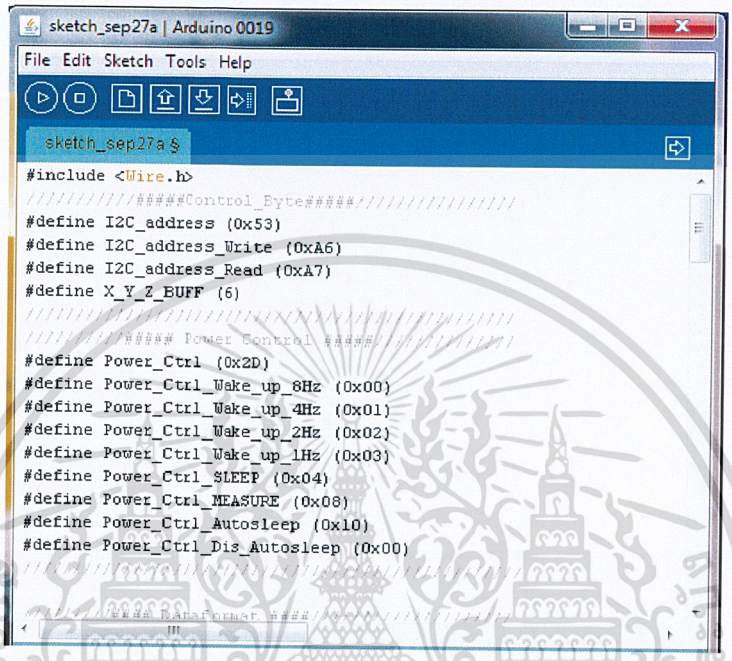
การอัพโหลดโปรแกรมจาก Arduino จะทำผ่านพอร์ตอนุกรม ซึ่งจะต้องกำหนดหมายเลขพอร์ตที่จะใช้ดังนี้ เลือกเมนู Tool>> Serial Port โปรแกรมจะแสดงพอร์ตอนุกรมที่มีในคอมพิวเตอร์ให้ผู้ใช้สามารถเลือกได้ ดังรูปที่ 4.8



รูปที่ 4.8 การเลือกพอร์ตอนุกรมที่ใช้ติดต่อไมโครคอนโทรลเลอร์ของ Arduino

4.2.3 การทดลองเขียนโปรแกรมรับค่าจากเซนเซอร์ ADXL345 และไดนาโมจักรยาน

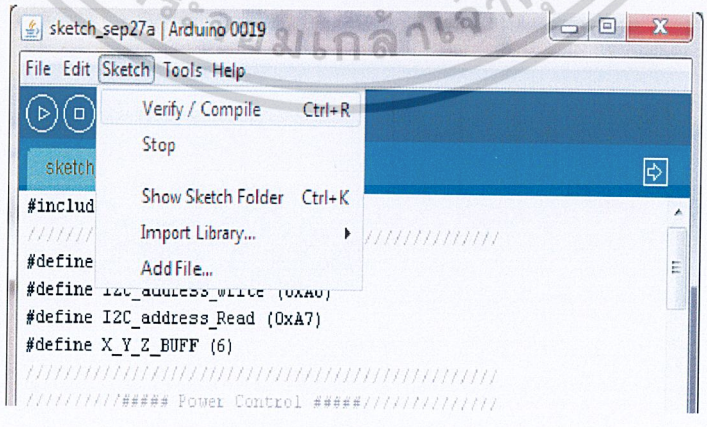
การเขียนโปรแกรมของ Arduino จะใช้รูปแบบของภาษา C และ C++ ซึ่งเป็นโปรแกรมที่เขียนง่ายไม่ซับซ้อนมากนัก โดยเมื่อเปิดโปรแกรม จะมีพื้นที่สำหรับเขียน code ดังรูปที่ 4.9



รูปที่ 4.9 แสดงหน้าต่างของโปรแกรม Arduino และโปรแกรมที่เขียน

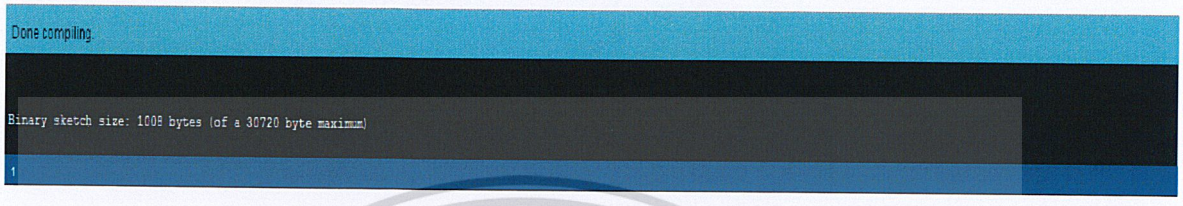
4.2.4 ทดลองการคอมไพล์โปรแกรม

เมื่อเปิดไฟล์โปรแกรมแล้วจากนั้นทำการคอมไพล์โปรแกรมเพื่อตรวจสอบความถูกต้องก่อนเขียนข้อมูลลงในบอร์ดไมโครคอนโทลเลอร์ Arduino ดังรูปที่ 4.10



รูปที่ 4.10 แสดงการเลือกเมนูคำสั่งเพื่อทำการคอมไพล์โปรแกรม

เมื่อทำการคอมไพล์โปรแกรมแล้วได้ผลการคอมไพล์ที่แถบแสดงสถานะด้านล่างของโปรแกรม ที่แถบสถานะจะปรากฏข้อความ Done Compiling และหน้าต่างแสดงผลข้อความว่า Binary sketch size: 1008 bytes (of a 30720 byte maximum) โดยแสดงว่าโปรแกรมที่ได้จากการคอมไพล์มีขนาด 1008 bytes จากทั้งหมด 30720 byte ที่สามารถใช้งานได้

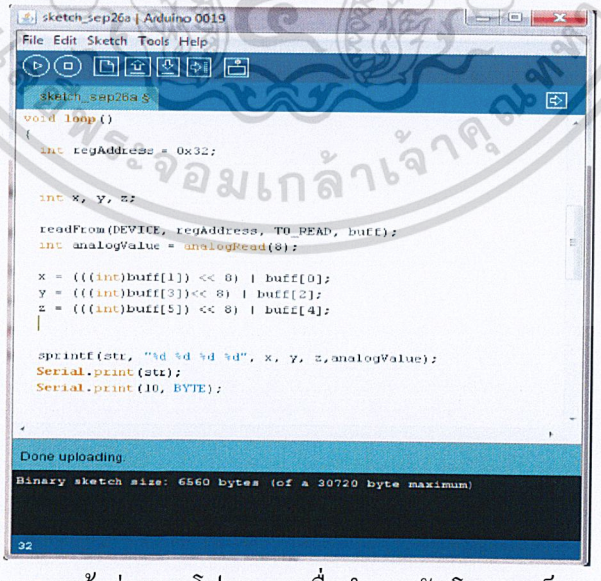


รูปที่ 4.11 แถบแสดงสถานการณ์คอมไพล์

4.2.5 การอัพโหลดข้อมูลไปยังบอร์ด ไมโครคอนโทรลเลอร์ Arduino Duemilanove

การอัพโหลดข้อมูล ไปยังบอร์ดไมโครคอนโทรลเลอร์ Arduino Duemilanove มีขั้นตอนดังนี้

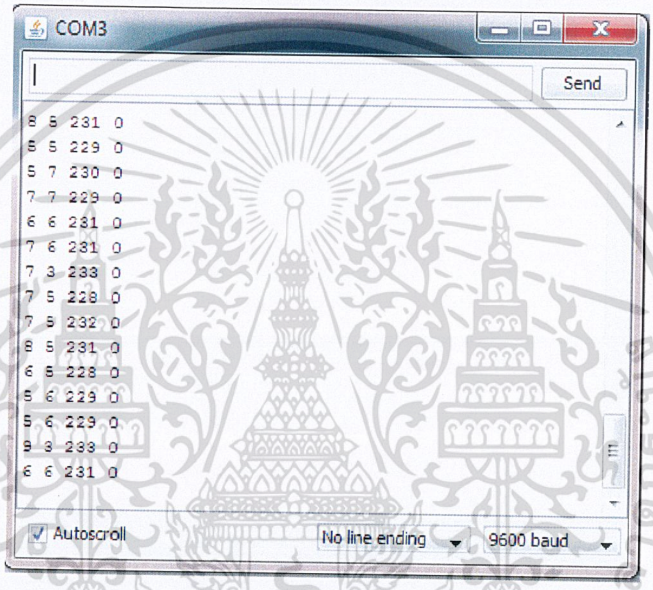
1. ต่อ USB พอร์ตเข้ากับบอร์ดไมโครคอนโทรลเลอร์ Arduino Duemilanove และคอมพิวเตอร์
2. กดสวิตซ์ Reset ผลคือจะมีไฟสีแดงกระพริบขึ้น แสดงว่าบอร์ดพร้อมทำการอัพโหลดข้อมูล
3. เลือกเมนู File >> Upload to I/O Board ที่โปรแกรม Arduino
4. อัพโหลดโปรแกรมที่ต้องการเขียน
5. เมื่ออัพโหลดเสร็จสมบูรณ์ แถบแสดงสถานะของโปรแกรมจะขึ้นข้อความ Done Uploading ดังรูปที่ 4.12



รูปที่ 4.12 หน้าต่างของโปรแกรมเมื่อทำการอัพโหลดเสร็จสมบูรณ์

4.2.6 การทดลองแสดงข้อมูลของ Serial Monitor ในโปรแกรม Arduino

โปรแกรม Arduino จะมี Serial Monitor สำหรับไว้แสดงค่าข้อมูลของพอร์ตที่ติดต่อเมื่อเปิดโปรแกรมแล้วสามารถเรียกใช้หน้าต่าง Serial Monitor โดยการ Ctrl +Shift + M โปรแกรมจะแสดงหน้าต่าง Serial Monitor ขึ้นมา จากนั้นจึงทำการเลือกพอร์ตและ Board Rate ที่ตั้งไว้ก็สามารถแสดงผลของโปรแกรมที่เขียนไว้ได้ตามรูปที่ 4.13 ซึ่งจะแสดงข้อมูลที่รับค่าข้อมูลจากเซนเซอร์ ADXL345 และไดนาโมจักรยานโดยจะเห็นได้ว่ามีการแสดงค่าครั้งละ 4 ค่า ลำดับค่าที่ 1-3 คือค่าในแกน X, Y, Z ตามลำดับ ลำดับค่าที่ 4 เป็นค่าข้อมูลของไดนาโมจักรยาน



รูปที่ 4.13 หน้าต่างของ Serial Monitor แสดงค่าข้อมูลของเซนเซอร์ ADXL345และไดนาโมจักรยาน

4.3 การแสดงกราฟข้อมูลจากเซนเซอร์วัดความเร่ง ADXL345

การทดลองในส่วนนี้เป็นการนำข้อมูลที่อ่านได้จากเซนเซอร์วัดความเร่ง ADXL345 มาแสดงในรูปแบบกราฟเพื่อแสดงความสัมพันธ์ของ แกน X,Y,Z เมื่อทำการวางตำแหน่งของเซนเซอร์ในระนาบต่างๆ ที่เวลาต่างกัน

4.3.1 แสดงค่าที่ได้จากเซนเซอร์วัดความเร่ง ADXL345 ในระนาบ Z+

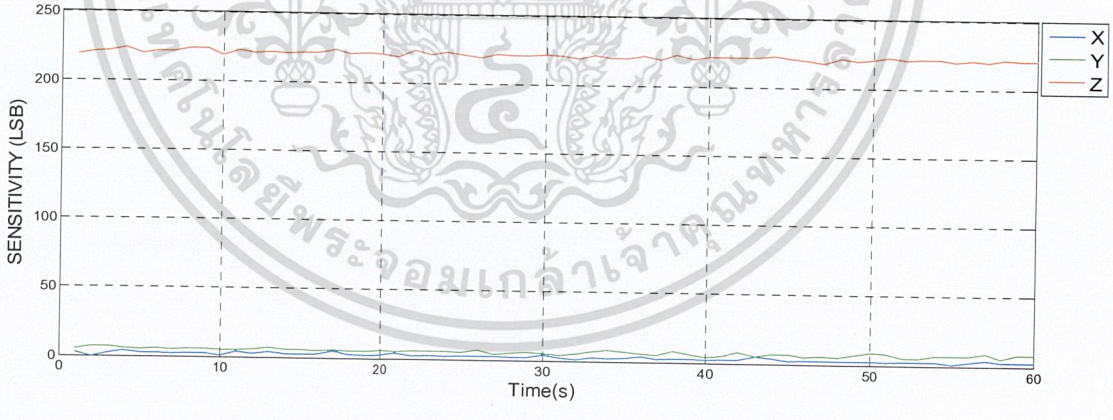
$X_{out} = 0g$

$Y_{out} = 0g$

$Z_{out} = +1g$

รูปที่ 4.14 การวางเซนเซอร์ระนาบ Z+

ผลการทดลองจะได้ค่าในแกน Z ที่มีค่าสูงมาก โดยค่าในแกน X และ Y มีค่าเข้าใกล้ศูนย์ ดังรูปที่ 4.15



รูปที่ 4.15 กราฟแสดงค่า Sensitivity ของแกน X, Y, Z เทียบกับเวลา ระนาบ Z+

4.3.2 แสดงค่าที่ได้จากเซนเซอร์วัดความเร่ง ADXL345 ในระนาบ Z-

$X_{out} = 0g$

$Y_{out} = 0g$

$Z_{out} = -1g$

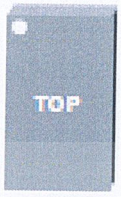
รูปที่ 4.16 การวางเซนเซอร์ระนาบ Z-

ผลการทดลองจะได้ค่าในแกน Z ที่มีค่าติดลบสูงมากโดยค่าในแกน X และ Y มีค่าเข้าใกล้ศูนย์ ดังรูปที่ 4.17

รูปที่ 4.17 กราฟแสดงค่า Sensitivity ของแกน X, Y, Z เทียบกับเวลา ระนาบ Z-

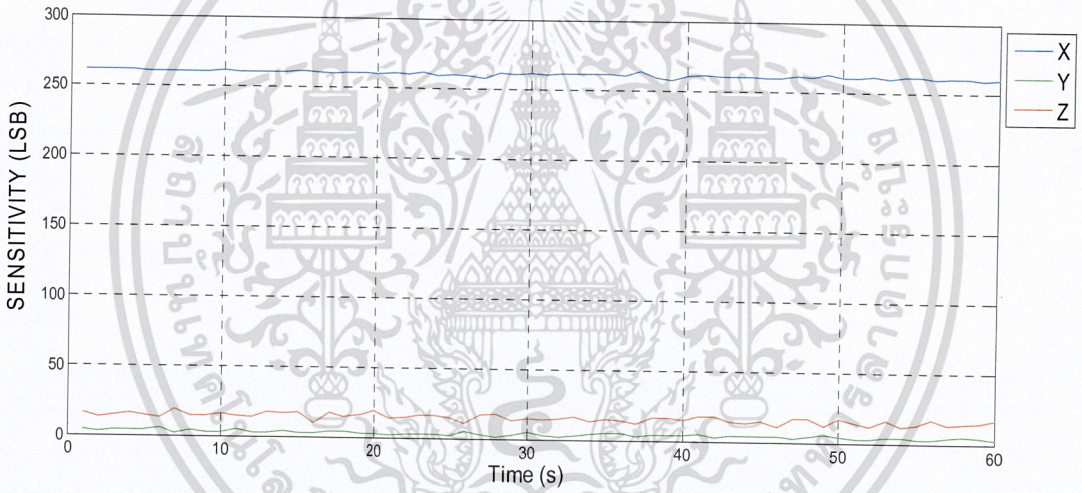
4.3.3 แสดงค่าที่ได้จากเซนเซอร์วัดความเร่ง ADXL345 ในระนาบ X+

$X_{out} = 1g, Y_{out} = 0g, Z_{out} = 0g$



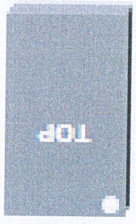
รูปที่ 4.18 การวางเซนเซอร์ระนาบ X+

ผลการทดลองจะได้ค่าในแกน X จะมีค่าสูงมากโดยค่าในแกน Y และ Z มีค่าเข้าใกล้ศูนย์ รูปที่ 4.19



รูปที่ 4.19 กราฟแสดงค่า Sensitivity ของแกน X, Y, Z เทียบกับเวลา ระนาบ X+

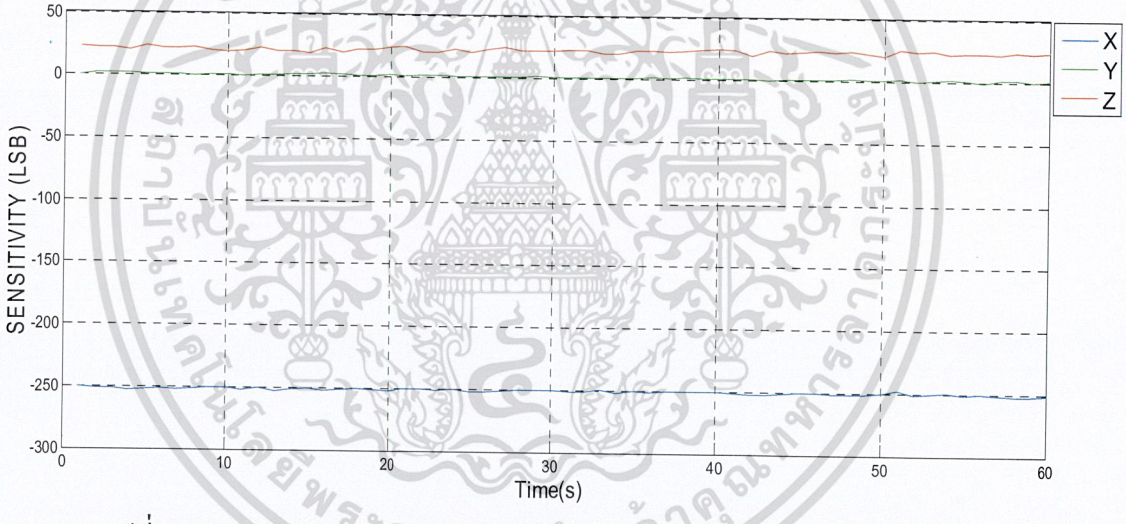
4.3.4 แสดงค่าที่ได้จาก เซนเซอร์วัดความเร่ง ADXL345 ในระนาบ X-



$X_{out} = -1g, Y_{out} = 0g, Z_{out} = 0g$

รูปที่ 4.20 การวางเซนเซอร์ระนาบ X-

ผลการทดลองจะได้ว่าค่าในแกน X จะติดลบสูงมากโดยค่าในแกน Y และ Z มีค่าเข้าใกล้ศูนย์ ดังรูปที่ 4.21



รูปที่ 4.21 กราฟแสดงค่า Sensitivity ของแกน X, Y, Z เทียบกับเวลา ระนาบ X-

4.3.5 แสดงค่าที่ได้จาก เซนเซอร์วัดความเร่ง ADXL345 ในระนาบ Y+

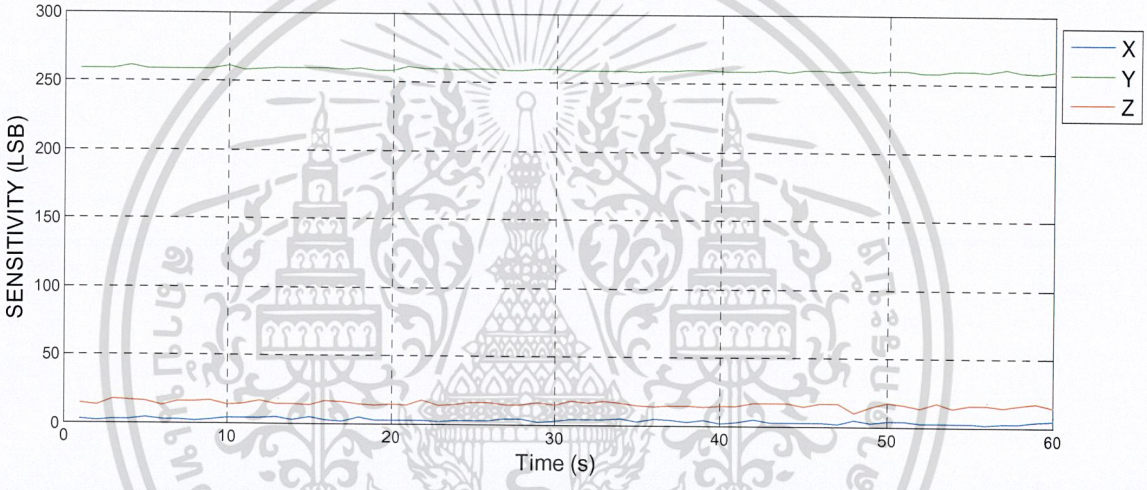
$X_{out} = 0g$

$Y_{out} = 1g$

$Z_{out} = 0g$

รูปที่ 4.22 การวางเซนเซอร์ระนาบ Y+

ผลการทดลองจะได้ค่าในแกน Y จะมีค่าสูงมากโดยค่าในแกน X และ Z มีค่าเข้าใกล้ศูนย์ รูปที่ 4.23



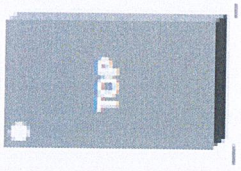
รูปที่ 4.23 กราฟแสดงค่า Sensitivity ของแกน X, Y, Z เทียบกับเวลา ระนาบ Y+

4.3.6 แสดงค่าที่ได้จาก เซนเซอร์วัดความเร่ง ADXL345 ในระนาบ Y-

$X_{out} = 0g$

$Y_{out} = -1g$

$Z_{out} = 0g$



รูปที่ 4.24 การวางเซนเซอร์ระนาบ Y-

ผลการทดลองจะได้ว่าค่าในแกน Y จะติดลบสูงมากโดยค่าในแกน X และ Z มีค่าเข้าใกล้ศูนย์ ดังรูปที่ 4.25

รูปที่ 4.25 กราฟแสดงค่า Sensitivity ของแกน X, Y, Z เทียบกับเวลา ระนาบ Y-

4.4 การทดลองใช้งานโปรแกรมภาษา Python สำหรับการรับค่าข้อมูลจากบอร์ดไมโครคอนโทรลเลอร์ Arduino เพื่อนำเข้าเก็บไว้ในรูปแบบไฟล์ .text

เนื่องจากโปรแกรม Blender3D เป็นโปรแกรมที่มีพื้นฐานอยู่ในระบบปฏิบัติการ Linux จึงต้องใช้ Python script ในการเขียนโปรแกรมรับค่าข้อมูลจากบอร์ดไมโครคอนโทรลเลอร์ เพื่อให้สามารถนำไปใช้งานใน โปรแกรม Blender3D ได้ ผลการทดลองคือเมื่อเปิดใช้งานโปรแกรมที่เขียนไว้จะมีค่าจากบอร์ดไมโครคอนโทรลเลอร์มาเก็บไว้ในไฟล์.text ที่ได้สร้างไว้ ทำให้สามารถนำไปใช้ประมวลผลในโปรแกรม Blender3D ได้

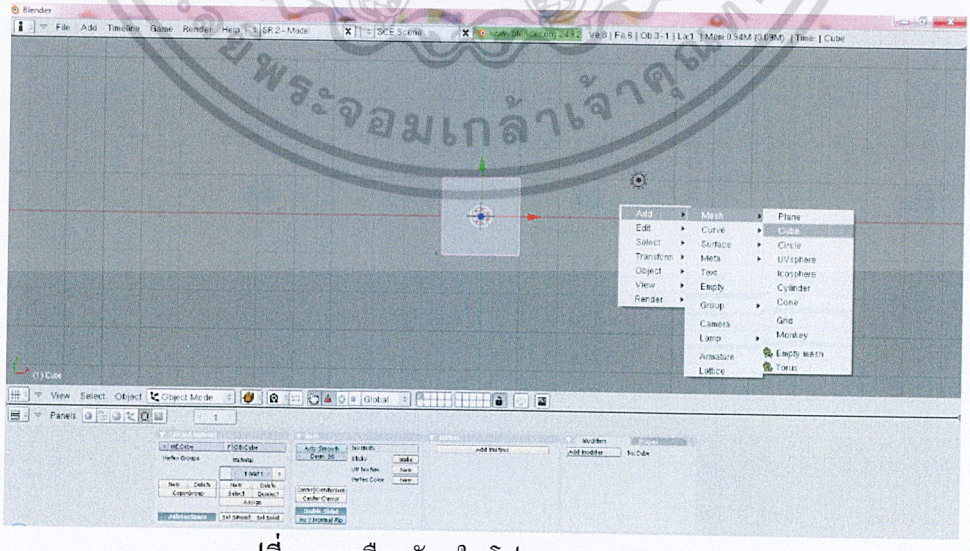
4.5 ทดลองการใช้งานโปรแกรม Blender3D

4.5.1 การทดลองใช้โปรแกรม Blender3D สำหรับออกแบบและสร้างกราฟิกสามมิติ

การออกแบบและสร้างกราฟิกสามมิติในโปรแกรม Blender3D สามารถออกแบบตามความต้องการหรือตามความคิดสร้างสรรค์ของผู้ออกแบบได้ แต่สิ่งสำคัญคือจะต้องรู้ทักษะเบื้องต้นสำหรับการออกแบบเสียก่อน เพราะโปรแกรมดังกล่าวมีการใช้งานรวมถึงไลบรารี่ที่เฉพาะตัว การทำงานต่างๆจึงสามารถนำมาใช้ได้ง่ายหากผู้ออกแบบมีทักษะดังกล่าว

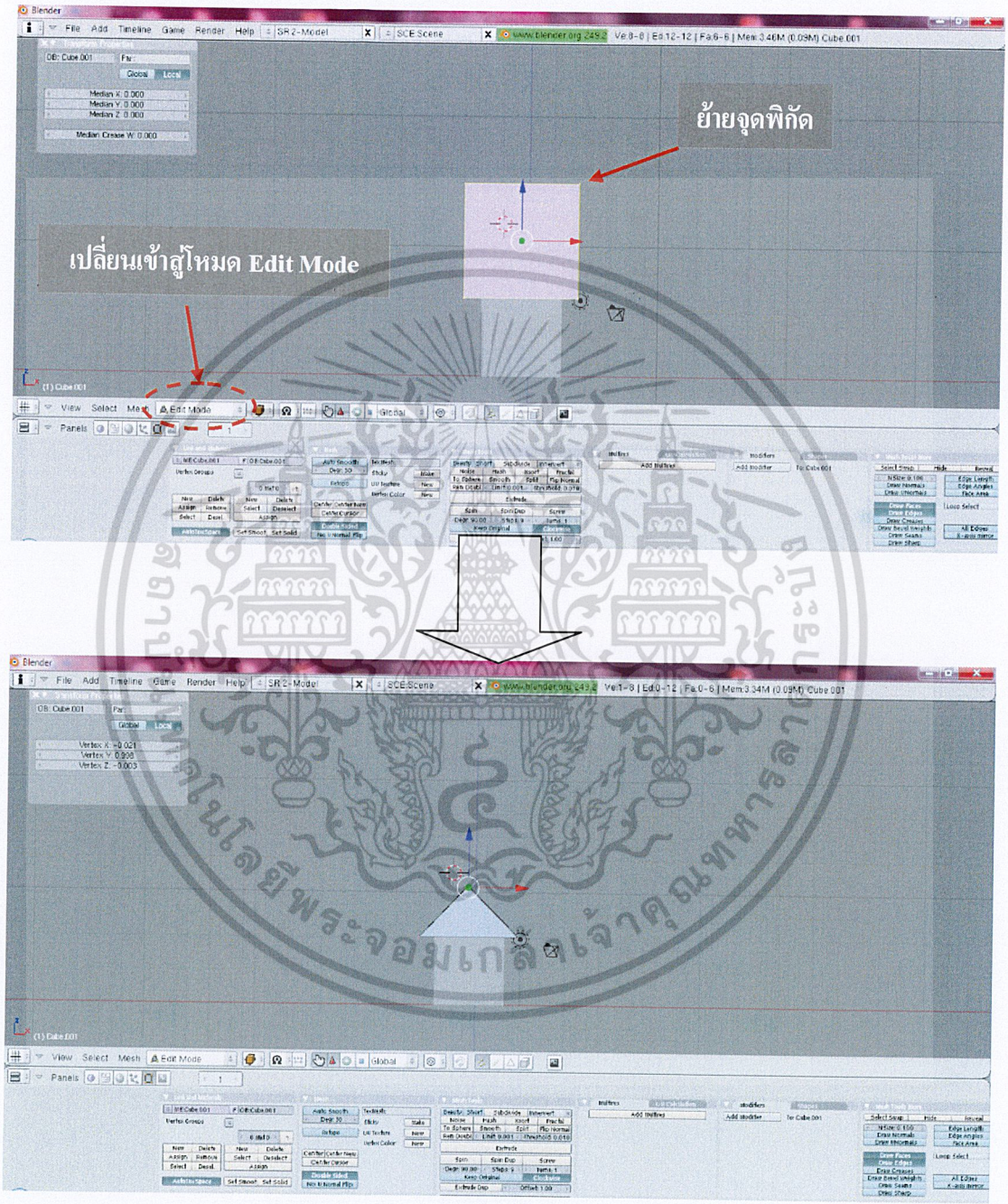
4.5.2 การออกแบบและสร้างกราฟิกสามมิติ (โมเดลบ้าน)

1. เปิดโปรแกรม Blender3D เพิ่มวัตถุตามรูปทรงที่ต้องการในที่นี่เลือกวัตถุชนิด Cube 2 ชิ้นและ Plane โดยเลือกที่ Spread Bar >> Add >> Mesh >> Cube และ Spread Bar >> Add >> Mesh >> Plane



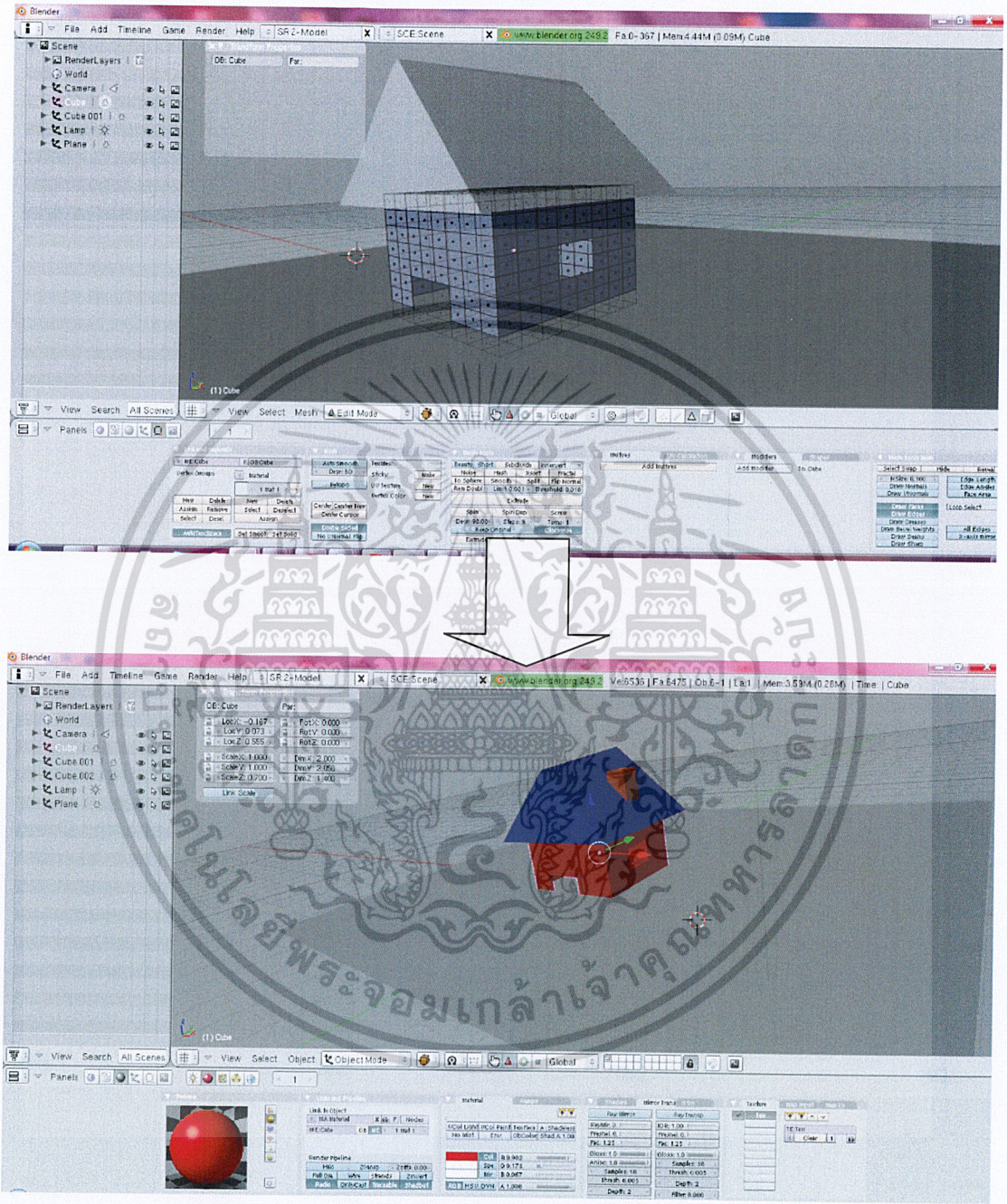
รูปที่ 4.26 เลือกวัตถุในโปรแกรม Blender3D

2. ทำหลังคาโมเดลรูปบ้านโดยปรับโหมดเป็นโหมด Edit Mode และคลิกเม้าส์ที่จุดพิกัดและย้ายจุดพิกัดดังกล่าวตามความต้องการ เช่นตัวอย่างดังรูปที่ 4.27



รูปที่ 4.27 ย้ายจุดพิกัดเพื่อสร้างโมเดลรูปบ้าน

3. ตกแต่งโมเดลรูปบ้านตามความต้องการของผู้ออกแบบ ดังเช่นตัวอย่างรูปที่ 4.28



รูปที่ 4.28 ตกแต่งโมเดลรูปบ้าน

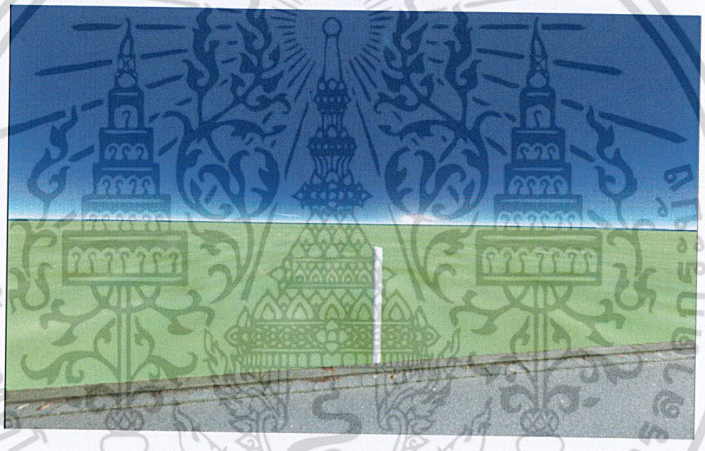
4.6 การทดลองนำค่าที่ส่งเข้ามาจากไมโครคอนโทรลเลอร์ Arduino ไปประมวลผลในโปรแกรมกราฟิกสามมิติ

ในส่วนนี้เป็นการนำค่าที่ได้จากบอร์ดในทิศทางแกนต่างๆ มาใช้งานในโปรแกรมเกมที่ได้สร้างไว้โดยแบ่งการรับค่าจาก 2 ส่วนคือ

4.6.1 การรับค่าจากเซนเซอร์วัดความเร่ง ADXL345

4.6.1.1 การเอียงศีรษะในระนาบ X-

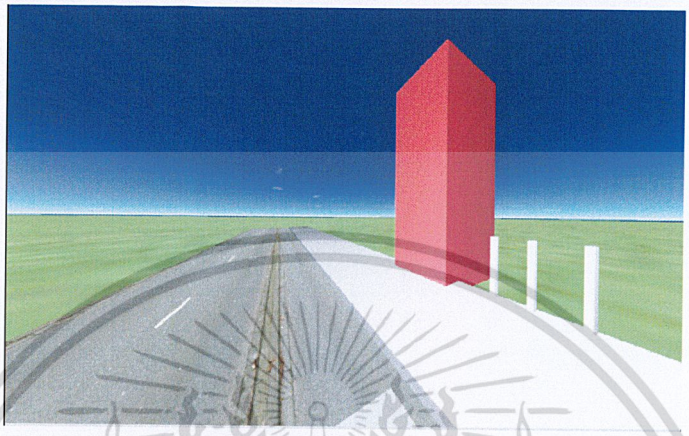
เมื่อเอียงศีรษะในระนาบ X- ค่าข้อมูลแกน X ของเซนเซอร์วัดความเร่งจะได้ค่าเป็นลบทำให้การหมุนของภาพในโปรแกรมกราฟิกสามมิติ เอียงในทิศทาง –X ดังรูปที่ 4.29



รูปที่ 4.29 ภาพที่ได้จากเกมกราฟิกสามมิติ เมื่อเอียงศีรษะในระนาบ X-

4.6.1.2 การเอียงศีรษะในระนาบ X+

เมื่อเอียงศีรษะในระนาบ X+ ค่าข้อมูลแกน X ของเซนเซอร์วัดความเร่งจะได้ค่าเป็นบวกทำให้มีการหมุนของภาพในโปรแกรมกราฟิกสามมิติ เอียงในทิศทาง +X ดังรูปที่ 4.30



รูปที่ 4.30 ภาพที่ได้จากเกมกราฟิกสามมิติ เมื่อเอียงศีรษะในระนาบ X+

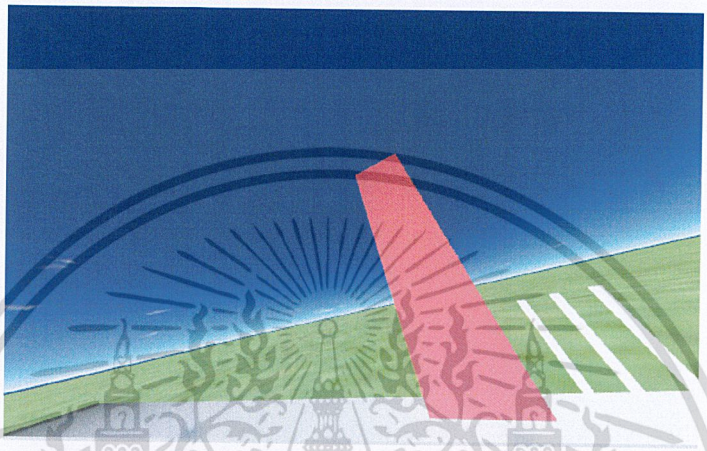
4.6.1.3 การเอียงศีรษะในระนาบ Y-

เมื่อเอียงศีรษะในระนาบ Y- ค่าข้อมูลแกน Y ของเซนเซอร์วัดความเร่งจะได้ค่าเป็นลบ แต่ได้ตั้งค่าให้ภาพกราฟิกสามมิติหมุนในระนาบ แกน Z ทำให้มีการหมุนของภาพในโปรแกรมกราฟิกสามมิติ เอียงในทิศทาง -Z ดังรูปที่ 4.31

รูปที่ 4.31 ภาพที่ได้จากเกมกราฟิกสามมิติ เมื่อเอียงศีรษะในระนาบ Y-

4.6.1.4 การเอียงศีรษะในระนาบ Y+

เมื่อเอียงศีรษะในระนาบ Y+ ค่าข้อมูลแกน Y ของเซนเซอร์วัดความเร่งจะได้ค่าเป็นบวก แต่ได้ตั้งค่าให้ภาพกราฟิกสามมิติหมุนในระนาบ แกน Z ทำให้มีการหมุนของภาพในโปรแกรมกราฟิกสามมิติ เอียงในทิศทาง +Z ดังรูปที่ 4.32



รูปที่ 4.32 แสดงภาพที่ได้จากเกมกราฟิกสามมิติ เมื่อเอียงศีรษะในระนาบ Y+

4.6.2 การรับค่าจากไดนาโมจักรยาน

เมื่อผู้เล่นทำการปั่นจักรยานทำให้ไดนาโมทำงานส่งเป็นค่าสัญญาณอนาล็อก มาประมวลผลในโปรแกรมเกมกราฟิกสามมิติที่สร้างไว้ ผลที่ได้คือเมื่อปั่นจักรยานทำให้ภาพในโปรแกรมเกมกราฟิกสามมิติมีการเคลื่อนที่ไปข้างหน้า และเมื่อปั่นจักรยานเร็วขึ้นจะส่งผลให้การเคลื่อนที่เร็วขึ้นด้วย

รูปที่ 4.33 ภาพที่ได้จากเกมกราฟิกสามมิติ เมื่อไดนาโมทำงาน

บทที่ 5
สรุปผลการดำเนินงาน

5.1 สรุปผล

ปริญญานิพนธ์นี้ได้ทำการออกแบบและสร้างโปรแกรมเกมกราฟิกสามมิติกับเครื่องออกกำลังกายด้วยจักรยาน เพื่อเพิ่มความเพลิดเพลินและแรงจูงใจให้กับการออกกำลังกายมากยิ่งขึ้น ทำให้ผู้ที่ไม่รักการออกกำลังกายเกิดความสนใจและมีวิสัยทัศน์ที่ดีกับการออกกำลังกายมากยิ่งขึ้น การดำเนินงานการออกแบบและสร้างดังกล่าวสามารถแบ่งได้เป็น 2 ส่วนหลักคือ

1. ส่วนการทำงานของอุปกรณ์บอร์ดไมโครคอนโทรลเลอร์ Arduino Duemilanove เกิดการรับค่าข้อมูลที่ถูกส่งค่าจากเซนเซอร์วัดความเร่ง ADXL345 และไดนาโมจักรยาน(ในรูปแบบอนาล็อก) โดยค่าดังกล่าวที่รับได้ จะถูกส่งผ่านทางพอร์ตอนุกรมเพื่อแสดงผลและนำไปใช้กับส่วนของกราฟิกสามมิติ
2. ส่วนการทำงานของโปรแกรมเกมกราฟิกสามมิติ เกิดจากการออกแบบและสร้างโครงสร้างรูปทรงสามมิติด้วยโปรแกรม Blender3D และควบคุมการทำงานต่างๆของระบบด้วยโปรแกรมภาษา Python โดยการแสดงผลเกิดขึ้นได้ เพราะโปรแกรมเกมกราฟิกสามมิติสามารถรับค่าได้จากผู้เล่นโดยตรง อันเนื่องมาจากการส่งค่าข้อมูลจากบอร์ดไมโครคอนโทรลเลอร์ Arduino Duemilanove และไดนาโมจักรยานผ่านพอร์ตอนุกรมมาประมวลผลกับรูปทรงสามมิติที่สร้างขึ้น

5.2 ปัญหาและข้อจำกัดของการดำเนินงาน

1. เนื่องจากงบประมาณในการดำเนินงานมีอย่างจำกัด อีกทั้งอุปกรณ์ที่ต้องใช้ในการดำเนินงานบางส่วนมีราคาค่อนข้างสูงจึงทำให้เกิดข้อจำกัดทางด้านงบประมาณ
2. โดยปกติการรับค่าจากเซนเซอร์วัดความเร่ง ADXL345 จะต้องเชื่อมต่อด้วยสายการส่งข้อมูลมายังบอร์ดไมโครคอนโทรลเลอร์ Arduino Duemilanove เมื่อนำไปติดบริเวณศีรษะจึงทำให้เกิดความยุ่งยากในการใช้งาน
3. อุปกรณ์เซนเซอร์วัดความเร่ง ADXL345 ไม่เป็นที่นิยมใช้ในประเทศไทย จึงมีความจำเป็นที่จะต้องหาข้อมูลจากต่างประเทศ ซึ่งเป็นอุปสรรคในเรื่องของภาษา

5.3 แนวทางการพัฒนาต่อ

การออกแบบและสร้างโปรแกรมเกมกราฟิกสามมิติกับการออกกำลังกายด้วยจักรยานสามารถใช้งานได้จริง แต่ยังมีส่วนของปัญหาและข้อจำกัดการดำเนินงานในบางส่วนที่เกิดขึ้น ดังนั้นหากต้องการให้สมบูรณ์แบบจนกระทั่งสามารถใช้ในการค้าพาณิชย์ได้ต่อไป จึงต้องดำเนินแนวทางการพัฒนาซึ่งมี 2 ส่วนดังนี้

1. ส่วนอุปกรณ์ จากปัญหาการรับค่าจากเซนเซอร์วัดความเร่ง ADXL345 ที่รับข้อมูลจากบอร์ดไมโครคอนโทรลเลอร์ Arduino Duemilanove ทำให้เกิดความยุ่งยากของสายติดต่อ ดังนั้นอาจจะพัฒนาได้โดยเปลี่ยนรูปแบบการติดต่อข้อมูลเป็นแบบไร้สายเพื่อให้เกิดความสะดวกในการใช้งานมากยิ่งขึ้น
2. ส่วนโปรแกรมกราฟิกสามมิติ จากการสร้างโปรแกรมเกมกราฟิกสามมิติดังกล่าว สามารถเพิ่มเติมทางด้านกราฟิกสามมิติให้มีความหลายหลายและความน่าสนใจกับผู้เล่นมาขึ้นกว่าเดิมตามความต้องการของผู้พัฒนาได้

บรรณานุกรม

[1] ภาคผนวก ข., Datasheet หรือ http://www.alldatasheet.net/

[2] เอกชัย มะการ, เรียนรู้เข้าใจใช้งานไมโครคอนโทรลเลอร์ตระกูล AVR ด้วย Arduino, กรุงเทพฯ : บริษัท อีทีที จำกัด, 2552

[3] http://www.justusers.net/knowledges/usb.htm

[4] http://www.thaimicrotron.com/CCS-628/Referrence/I2CBUS.htm

[5] ภาคผนวก ข., Datasheet หรือ
http://www.sparkfun.com/datasheets/Sensors/Accelerometer/ADXL345.pdf

[6] http://www.slideshare.net/ampair14901/ss-5093688

[7] http://lpsci.nfe.go.th/elearning/basic_science/basic64.php

[8] น.ท.ไพศาล โมลิสกุลมงคล, คอมพิวเตอร์กราฟิกส์ ใช้ OpenGL, กรุงเทพฯ : หจก.ไทยเจริญการพิมพ์, 2550

[9] cpe.ku.ac.th/~thanachat/documents/204111/01_Intro.ppt

[10] โอภาส ศิริครรชิตถาวร วรพจน์ กรแก้ววัฒนกุล ชัยวัฒน์ ลิ้มพรจิตรวิไล, เรียนรู้ควบคุมอย่างง่ายด้วยโปรแกรมภาษา C กับ Arduino และบอร์ดไมโครคอนโทรลเลอร์ POP-168, กรุงเทพฯ : บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด, 2552

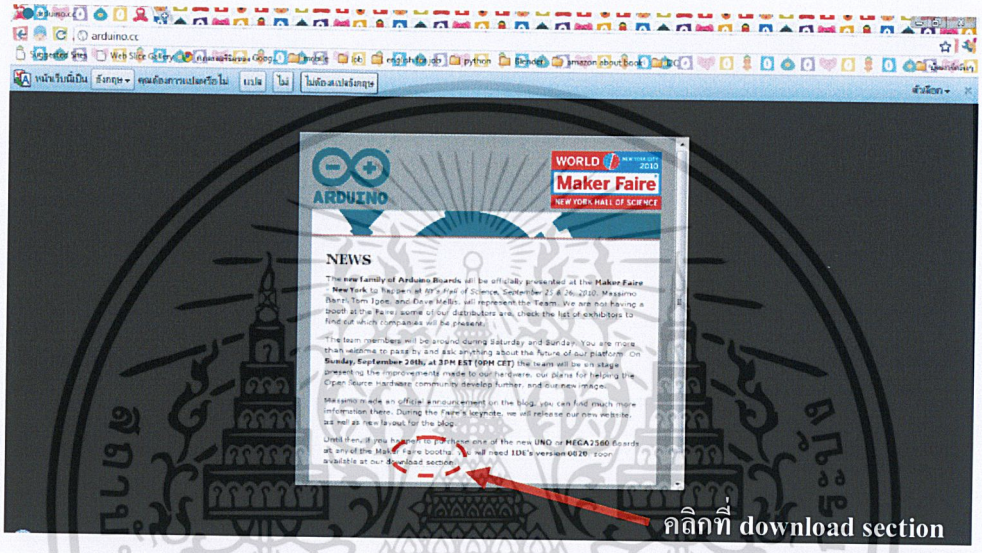
ภาคผนวก ก.

คู่มือการติดตั้งไดรเวอร์และโปรแกรม

1. การติดตั้งโปรแกรม Arduino

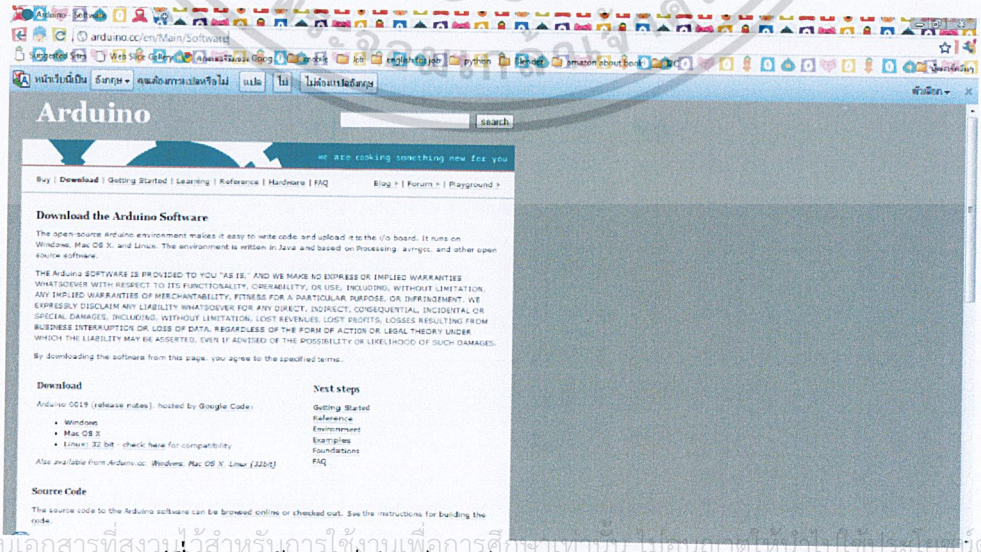
เนื่องจากโปรแกรมภาษา Arduino เป็นโปรแกรมชนิด Open Source จึงสามารถหาดาวน์โหลดติดตั้งได้ฟรี อีกทั้งยังมีการติดตั้งการใช้งานที่ง่าย สะดวก และรวดเร็วอีกด้วย โดยวิธีการติดตั้งต่างๆ มีดังต่อไปนี้

1. ดาวน์โหลดโปรแกรมภาษา Arduino จากเว็บไซต์ arduino.cc และคลิกที่ download section บริเวณด้านล่าง



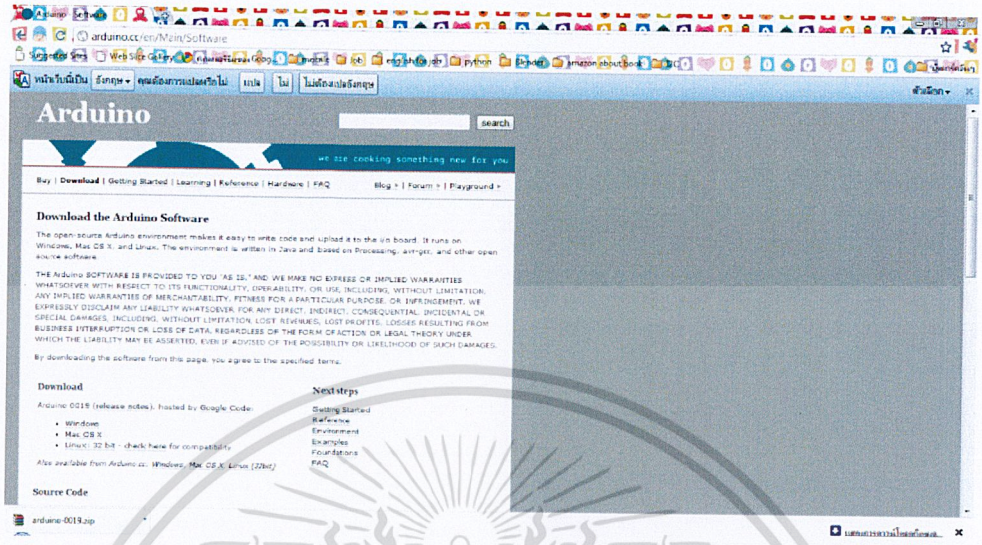
รูปที่ ก.1 หน้าแรกของเว็บไซต์เพื่อดาวน์โหลดโปรแกรมภาษา Arduino

2. คลิกเลือกตามแพลตฟอร์มของคอมพิวเตอร์ที่ต้องการเช่น Windows, Mac OS X หรือ Linux 32 bit



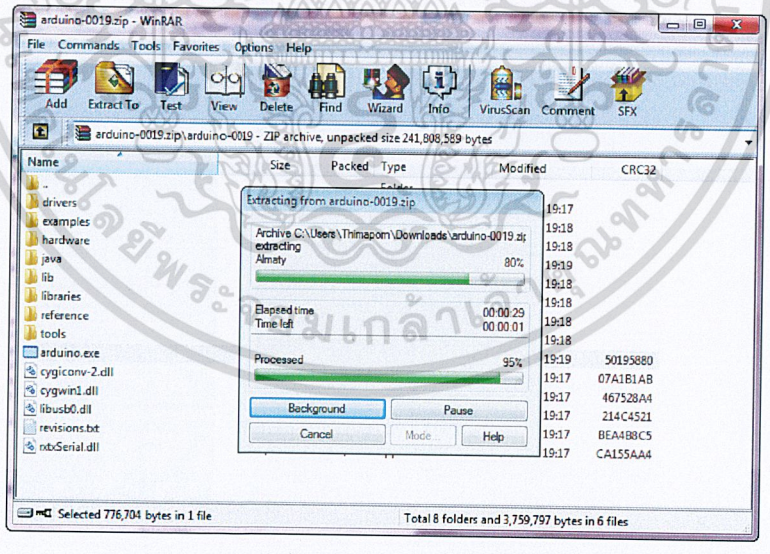
รูปที่ ก.2 หน้าต่างเว็บไซต์ดาวน์โหลดโปรแกรมภาษา Arduino

3. เมื่อดาวน์โหลดเสร็จเรียบร้อยแล้ว จะพบชื่อไฟล์เป็น arduino-0019.zip ให้ทำการดับเบิ้ลคลิกเพื่อทำการติดตั้ง



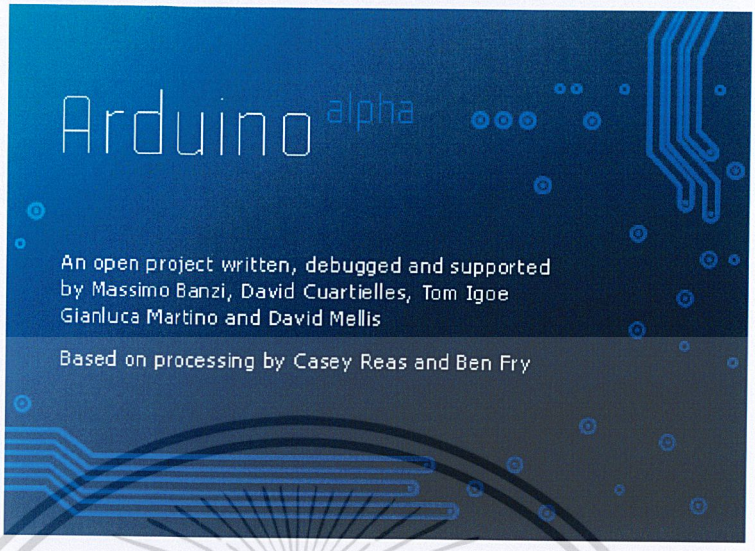
รูปที่ ก.3 ไฟล์ที่ได้รับหลังจากดาวน์โหลดจากเว็บไซต์

4. เมื่อดับเบิ้ลคลิกไปจะพบหน้าต่างดังกล่าว ให้ดับเบิ้ลคลิกเข้าไปที่ไฟล์ arduino.exe เพื่อทำการติดตั้งโปรแกรม



รูปที่ ก.4 เปิดไฟล์ arduino.exe เพื่อติดตั้งโปรแกรม

5. เมื่อติดตั้งสมบูรณ์จะมีหน้าต่างเข้าสู่โปรแกรมถูกเปิดขึ้นอัตโนมัติ



รูปที่ ก.5 หน้าแรกของโปรแกรม Arduino

6. หลังจากหน้าต่างแรกของโปรแกรมถูกเปิดขึ้น จะมีหน้าต่างสำหรับเขียนโปรแกรมเปิดขึ้นทันทีโดยไม่ต้องกดปุ่มใดๆ

รูปที่ ก.6 หน้าต่างของโปรแกรม Arduino

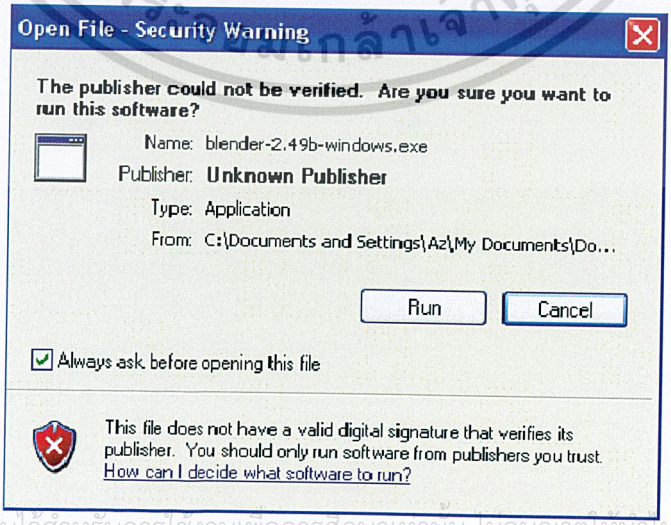
2. การติดตั้งโปรแกรม Blender3D

เนื่องจากโปรแกรม Blender3D เป็นภาษาที่ไว้สำหรับสร้างโมเดลและเขียนกราฟิกสามมิติ ซึ่งโปรแกรมดังกล่าวเป็นชนิด Open Source จึงสามารถหาดาวน์โหลดเพื่อติดตั้งใช้ได้ฟรี อีกทั้งยังมีการติดตั้งการใช้งานที่ง่าย สะดวก และใช้พื้นที่น้อยในการติดตั้ง จึงเหมาะกับงานที่จะนำไปพัฒนาต่อมากมาย สำหรับวิธีการติดตั้งต่างๆ มีดังต่อไปนี้

1. ดาวน์โหลดโปรแกรม Blender3D จากเว็บไซต์ www.blender.org/download/get-blender/ และคลิกเลือกแพลตฟอร์มที่เหมาะสม

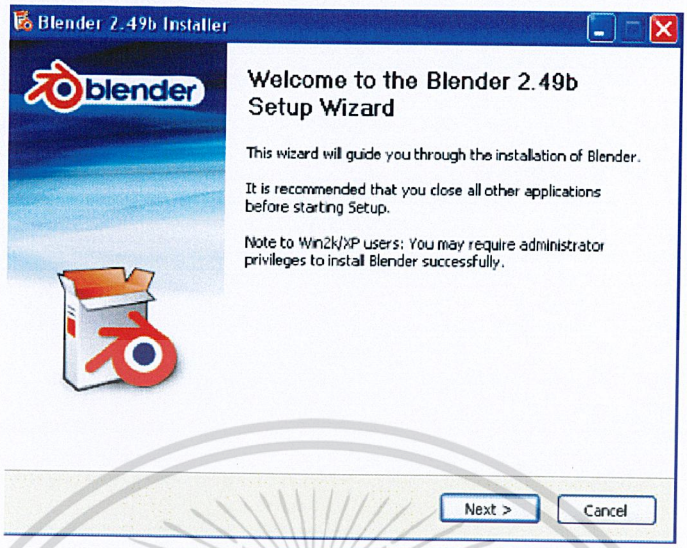
รูปที่ ก.7 หน้าแรกของเว็บไซต์เพื่อดาวน์โหลดโปรแกรม Blender

2. หน้าต่างเพื่อเริ่มการติดตั้งโปรแกรม Blender



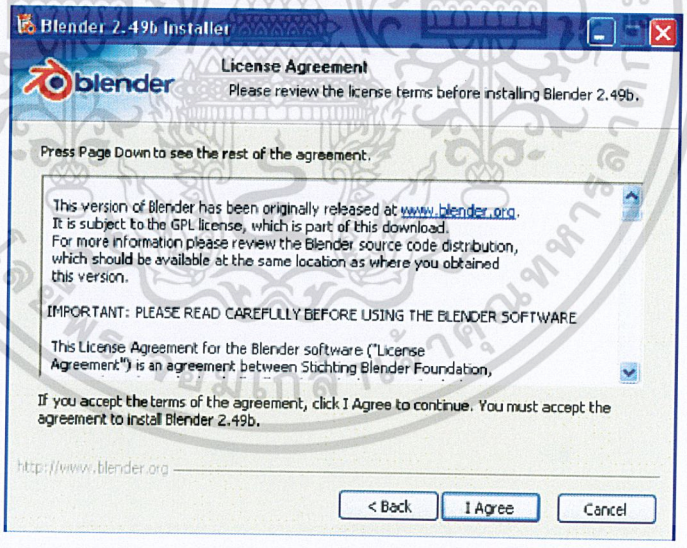
รูปที่ ก.8 หน้าต่างเพื่อเริ่มการติดตั้ง

3. คลิก Next เพื่อเข้าสู่การติดตั้งขั้นต่อไป หากไม่ต้องการติดตั้งให้กด Cancel เพื่อออก



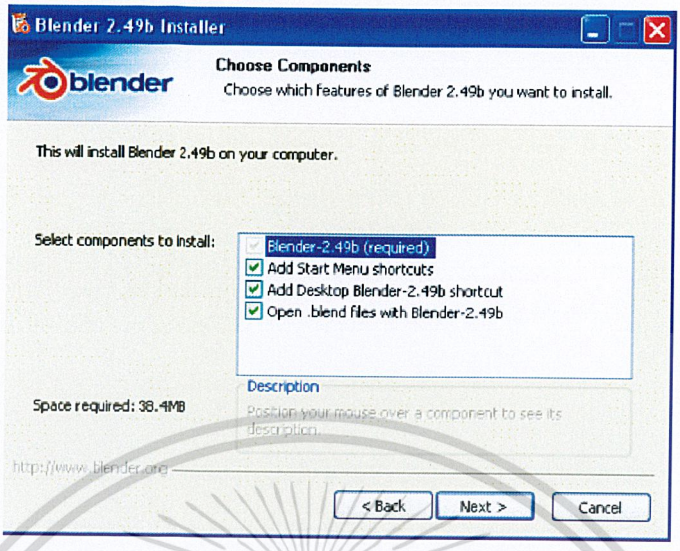
รูปที่ ก.9 เริ่มการติดตั้งโปรแกรม

4. รายละเอียดเงื่อนไขของการติดตั้งโปรแกรม Blender3D คลิก I Agree เพื่อเข้าสู่ขั้นตอนการติดตั้งต่อไป



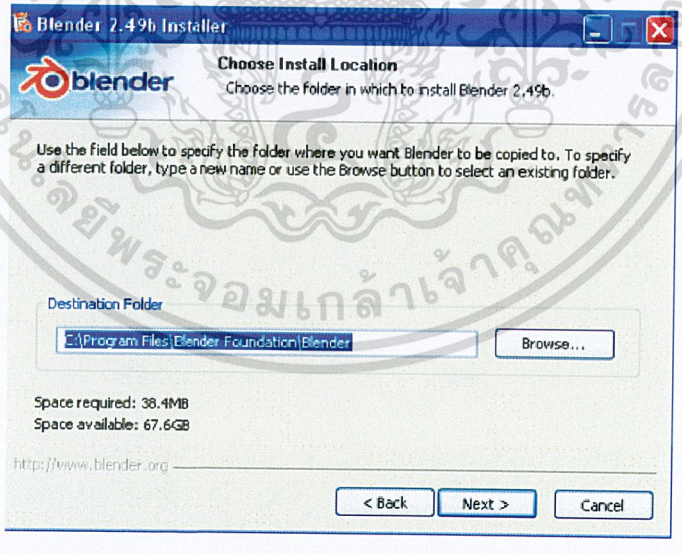
รูปที่ ก.10 รายละเอียดและเงื่อนไข

6. เลือก components(ส่วนประกอบ) สำหรับการติดตั้งโปรแกรม



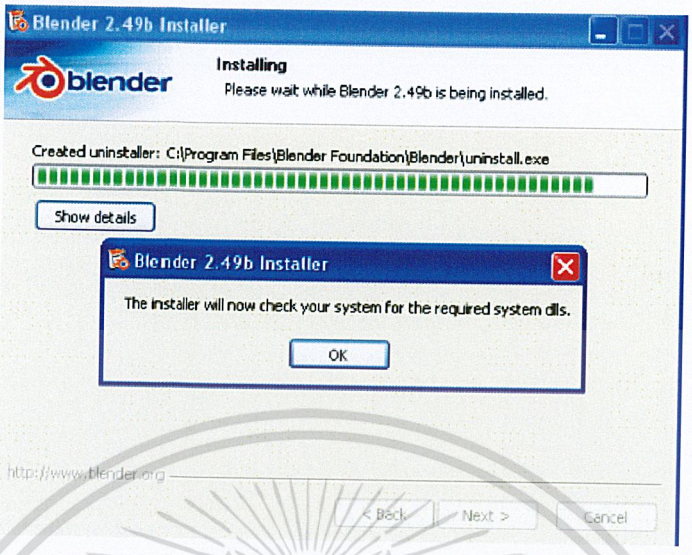
รูปที่ ก.11 หน้าต่างส่วนประกอบสำหรับการติดตั้ง

7. เลือกพื้นที่ปลายทางที่ต้องการติดตั้ง หากต้องการเปลี่ยนปลายทางในการติดตั้งให้กด Browse แล้วเลือกปลายทางที่ต้องการ เมื่อเลือกปลายทางเสร็จสิ้นแล้วให้คลิกที่ปุ่ม Next เพื่อเข้าสู่ขั้นตอนการติดตั้งต่อไป



รูปที่ ก.12 หน้าต่างแสดงพื้นที่ปลายทางที่ต้องการติดตั้ง

8. หน้าต่างเพื่อรอการติดตั้งของโปรแกรม Blender3D



รูปที่ ก.13 หน้าต่างกำลังดำเนินการติดตั้งโปรแกรมลงในเครื่องที่ใช้งาน

9. สิ้นสุดขั้นตอนการติดตั้งโปรแกรม Blender3D สำหรับขั้นตอนสุดท้ายนี้จะมีให้เลือกว่าต้องการสั่งให้มีการรันโปรแกรม Blender3D ทันทีหรือไม่ จากนั้นคลิกปุ่ม Finish เพื่อเสร็จสิ้นการติดตั้งโปรแกรม Blender3D อย่างสมบูรณ์

รูปที่ ก.14 หน้าจอขั้นตอนสิ้นสุดการติดตั้งโปรแกรม Blender3D

ภาคผนวก ข.

Datasheet

3-Axis, ±2 *g*/±4 *g*/±8 *g*/±16 *g* Digital Accelerometer

ADXL345

FEATURES

- Ultralow power: as low as 23 µA in measurement mode and 0.1 µA in standby mode at V_S = 2.5 V (typical)
- Power consumption scales automatically with bandwidth
- User-selectable resolution
  - Fixed 10-bit resolution
  - Full resolution, where resolution increases with *g* range, up to 13-bit resolution at ±16 *g* (maintaining 4 mg/LSB scale factor in all *g* ranges)
- Patent pending, embedded memory management system with FIFO technology minimizes host processor load
- Single tap/double tap detection
- Activity/inactivity monitoring
- Free-fall detection
- Supply voltage range: 2.0 V to 3.6 V
- I/O voltage range: 1.7 V to V_S
- SPI (3- and 4-wire) and I²C digital interfaces
- Flexible interrupt modes mappable to either interrupt pin
- Measurement ranges selectable via serial command
- Bandwidth selectable via serial command
- Wide temperature range (−40°C to +85°C)
- 10,000 *g* shock survival
- Pb free/RoHS compliant
- Small and thin: 3 mm × 5 mm × 1 mm LGA package

APPLICATIONS

- Handsets
- Medical instrumentation
- Gaming and pointing devices
- Industrial instrumentation
- Personal navigation devices
- Hard disk drive (HDD) protection

GENERAL DESCRIPTION

The ADXL345 is a small, thin, ultralow power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16 *g*. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3- or 4-wire) or I²C digital interface.

The ADXL345 is well suited for mobile device applications. It measures the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (3.9 mg/LSB) enables measurement of inclination changes less than 1.0°.

Several special sensing functions are provided. Activity and inactivity sensing detect the presence or lack of motion by comparing the acceleration on any axis with user-set thresholds. Tap sensing detects single and double taps in any direction. Free-fall sensing detects if the device is falling. These functions can be mapped individually to either of two interrupt output pins. An integrated, patent pending memory management system with a 32-level first in, first out (FIFO) buffer can be used to store data to minimize host processor activity and lower overall system power consumption.

Low power modes enable intelligent motion-based power management with threshold sensing and active acceleration measurement at extremely low power dissipation.

The ADXL345 is supplied in a small, thin, 3 mm × 5 mm × 1 mm, 14-lead, plastic package.

FUNCTIONAL BLOCK DIAGRAM

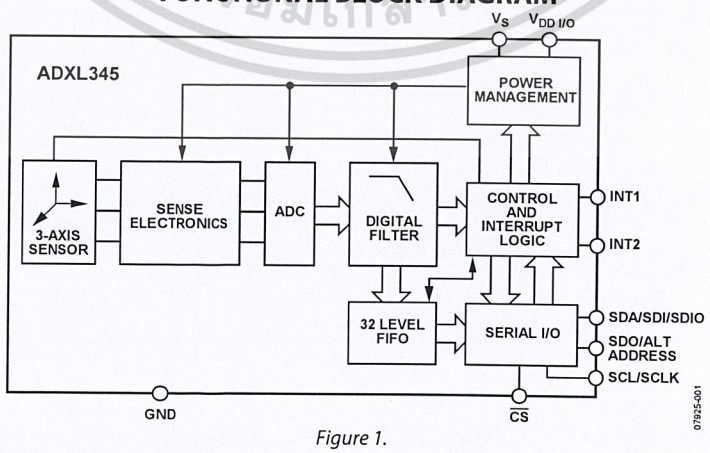


Figure 1.

Rev. A

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700
www.analog.com
Fax: 781.461.3113

TABLE OF CONTENTS

REVISION HISTORY

4/10—Rev. 0 to Rev. A

Changes to Features Section and General Description Section 1
Changes to Specifications Section 3
Changes to Table 2 and Table 3 5
Added Package Information Section, Figure 2, and Table 4; Renumbered Sequentially 5
Changes to Pin 12 Description, Table 5 6
Added Typical Performance Characteristics Section 7
Changes to Theory of Operation Section and Power Sequencing Section 12
Changes to Powers Savings Section, Table 7, Table 8, Auto Sleep Mode Section, and Standby Mode Section 13
Changes to SPI Section 14
Changes to Figure 36 to Figure 38 15
Changes to Table 9 and Table 10 16
Changes to I²C Section and Table 11 17
Changes to Table 12 18
Changes to Interrupts Section, Activity Section, Inactivity Section, and FREE_FALL Section 19
Added Table 13 19
Changes to FIFO Section 20
Changes to Self-Test Section and Table 15 to Table 18 21
Added Figures 42 and Table 14 21
Changes to Table 19 22
Changes to Register 0x1D—THRESH_TAP (Read/Write) Section, Register 0x1E, Register 0x1F, Register 0x20—OFSX, OFSY, OSXZ (Read/Write) Section, Register 0x21—DUR (Read/Write) Section, Register 0x22—Latent (Read/Write) Section, and Register 0x23—Window (Read/Write) Section 23
Changes to ACT_X Enable Bits and INACT_X Enable Bit Section, Register 0x28—THRESH_FF (Read/Write) Section, Register 0x29—TIME_FF (Read/Write) Section, Asleep Bit Section, and AUTO_SLEEP Bit Section 24
Changes to Sleep Bit Section 25
Changes to Power Supply Decoupling Section, Mechanical Considerations for Mounting Section, and Tap Detection Section 27
Changes to Threshold Section 28
Changes to Sleep Mode vs. Low Power Mode Section 29
Added Offset Calibration Section 29
Changes to Using Self-Test Section 30
Added Data Formatting of Upper Data Rates Section, Figure 48, and Figure 49 31
Added Noise Performance Section, Figure 50 to Figure 52, and Operation at Voltages Other Than 2.5 V Section 32
Added Offset Performance at Lowest Data Rates Section and Figure 53 to Figure 55 33

6/09—Revision 0: Initial Version

SPECIFICATIONS

T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V, acceleration = 0 *g*, C_S = 10 µF tantalum, $C_{I/O}$ = 0.1 µF, output data rate (ODR) = 800 Hz, unless otherwise noted. All minimum and maximum specifications are guaranteed. Typical specifications are not guaranteed.

Table 1.

Parameter	Test Conditions	Min	Typ[1]	Max	Unit
SENSOR INPUT	Each axis				
Measurement Range	User selectable		±2, ±4, ±8, ±16		*g*
Nonlinearity	Percentage of full scale		±0.5		%
Inter-Axis Alignment Error			±0.1		Degrees
Cross-Axis Sensitivity[2]			±1		%
OUTPUT RESOLUTION	Each axis				
All *g* Ranges	10-bit resolution		10		Bits
±2 *g* Range	Full resolution		10		Bits
±4 *g* Range	Full resolution		11		Bits
±8 *g* Range	Full resolution		12		Bits
±16 *g* Range	Full resolution		13		Bits
SENSITIVITY	Each axis				
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	All *g*-ranges, full resolution	230	256	282	LSB/*g*
	±2 *g*, 10-bit resolution	230	256	282	LSB/*g*
	±4 *g*, 10-bit resolution	115	128	141	LSB/*g*
	±8 *g*, 10-bit resolution	57	64	71	LSB/*g*
	±16 *g*, 10-bit resolution	29	32	35	LSB/*g*
Sensitivity Deviation from Ideal	All *g*-ranges		±1.0		%
Scale Factor at X_{OUT}, Y_{OUT}, Z_{OUT}	All *g*-ranges, full resolution	3.5	3.9	4.3	mg/LSB
	±2 *g*, 10-bit resolution	3.5	3.9	4.3	mg/LSB
	±4 *g*, 10-bit resolution	7.1	7.8	8.7	mg/LSB
	±8 *g*, 10-bit resolution	14.1	15.6	17.5	mg/LSB
	±16 *g*, 10-bit resolution	28.6	31.2	34.5	mg/LSB
Sensitivity Change Due to Temperature			±0.01		%/°C
0 *g* OFFSET	Each axis				
0 *g* Output for X_{OUT}, Y_{OUT}		−150	0	+150	mg
0 *g* Output for Z_{OUT}		−250	0	+250	mg
0 *g* Output Deviation from Ideal, X_{OUT}, Y_{OUT}			±35		mg
0 *g* Output Deviation from Ideal, Z_{OUT}			±40		mg
0 *g* Offset vs. Temperature for X-, Y-Axes			±0.4		mg/°C
0 *g* Offset vs. Temperature for Z-Axis			±0.8		mg/°C
NOISE					
X-, Y-Axes	ODR = 100 Hz for ±2 *g*, 10-bit resolution or all *g*-ranges, full resolution		0.75		LSB rms
Z-Axis	ODR = 100 Hz for ±2 *g*, 10-bit resolution or all *g*-ranges, full resolution		1.1		LSB rms
OUTPUT DATA RATE AND BANDWIDTH	User selectable				
Output Data Rate (ODR)[3, 4, 5]		0.1		3200	Hz
SELF-TEST[6]					
Output Change in X-Axis		0.20		2.10	*g*
Output Change in Y-Axis		−2.10		−0.20	*g*
Output Change in Z-Axis		0.30		3.40	*g*
POWER SUPPLY					
Operating Voltage Range (V_S)		2.0	2.5	3.6	V
Interface Voltage Range ($V_{DD\ I/O}$)		1.7	1.8	V_S	V
Supply Current	ODR ≥ 100 Hz		140		µA
	ODR < 10 Hz		30		µA
Standby Mode Leakage Current			0.1		µA
Turn-On and Wake-Up Time[7]	ODR = 3200 Hz		1.4		ms

Parameter	Test Conditions	Min	Typ[1]	Max	Unit
TEMPERATURE					
Operating Temperature Range		−40		+85	°C
WEIGHT					
Device Weight			30		mg

[1] The typical specifications shown are for at least 68% of the population of parts and are based on the worst case of mean ±1 σ, except for 0 *g* output and sensitivity, which represents the target value. For 0 *g* offset and sensitivity, the deviation from the ideal describes the worst case of mean ±1 σ.
[2] Cross-axis sensitivity is defined as coupling between any two axes.
[3] Bandwidth is the −3 dB frequency and is half the output data rate, bandwidth = ODR/2.
[4] The output format for the 3200 Hz and 1600 Hz ODRs is different than the output format for the remaining ODRs. This difference is described in the Data Formatting of Upper Data Rates section.
[5] Output data rates below 6.25 Hz exhibit additional offset shift with increased temperature, depending on selected output data rate. Refer to the Offset Performance at Lowest Data Rates section for details.
[6] Self-test change is defined as the output (*g*) when the SELF_TEST bit = 1 (in the DATA_FORMAT register, Address 0x31) minus the output (*g*) when the SELF_TEST bit = 0. Due to device filtering, the output reaches its final value after 4 × τ when enabling or disabling self-test, where τ = 1/(data rate). The part must be in normal power operation (LOW_POWER bit = 0 in the BW_RATE register, Address 0x2C) for self-test to operate correctly.
[7] Turn-on and wake-up times are determined by the user-defined bandwidth. At a 100 Hz data rate, the turn-on and wake-up times are each approximately 11.1 ms. For other data rates, the turn-on and wake-up times are each approximately τ + 1.1 in milliseconds, where τ = 1/(data rate).

ABSOLUTE MAXIMUM RATINGS

Table 2.

Parameter	Rating
Acceleration	
Any Axis, Unpowered	10,000 *g*
Any Axis, Powered	10,000 *g*
V_S	−0.3 V to +3.9 V
$V_{DD\ I/O}$	−0.3 V to +3.9 V
Digital Pins	−0.3 V to $V_{DD\ I/O}$ + 0.3 V or 3.9 V, whichever is less
All Other Pins	−0.3 V to +3.9 V
Output Short-Circuit Duration (Any Pin to Ground)	Indefinite
Temperature Range	
Powered	−40°C to +105°C
Storage	−40°C to +105°C

Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

THERMAL RESISTANCE

Table 3. Package Characteristics

Package Type	θ_{JA}	θ_{JC}	Device Weight
14-Terminal LGA	150°C/W	85°C/W	30 mg

PACKAGE INFORMATION

The information in Figure 2 and Table 4 provide details about the package branding for the ADXL345. For a complete listing of product availability, see the Ordering Guide section.

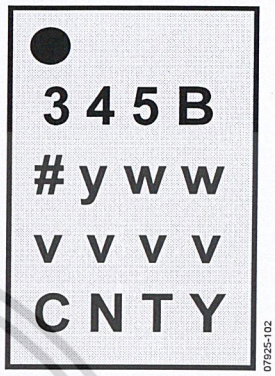


Figure 2. Product Information on Package (Top View)

Table 4. Package Branding Information

Branding Key	Field Description
345B	Part identifier for ADXL345
#	RoHS-compliant designation
yww	Date code
vvvv	Factory lot code
CNTY	Country of origin

ESD CAUTION

ESD (electrostatic discharge) sensitive device. Charged devices and circuit boards can discharge without detection. Although this product features patented or proprietary protection circuitry, damage may occur on devices subjected to high energy ESD. Therefore, proper ESD precautions should be taken to avoid performance degradation or loss of functionality.

PIN CONFIGURATION AND FUNCTION DESCRIPTIONS

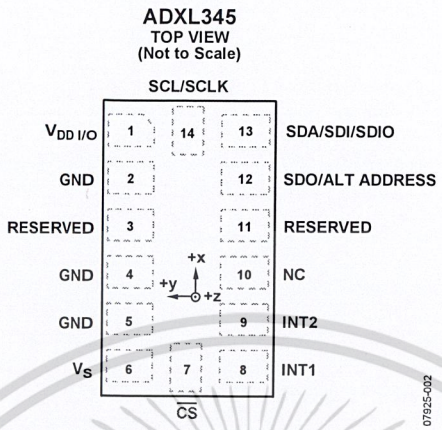


Figure 3. Pin Configuration (Top View)

Table 5. Pin Function Descriptions

Pin No.	Mnemonic	Description
1	$V_{DD\ I/O}$	Digital Interface Supply Voltage.
2	GND	This pin must be connected to ground.
3	RESERVED	Reserved. This pin must be connected to V_S or left open.
4	GND	This pin must be connected to ground.
5	GND	This pin must be connected to ground.
6	V_S	Supply Voltage.
7	$\overline{CS}$	Chip Select.
8	INT1	Interrupt 1 Output.
9	INT2	Interrupt 2 Output.
10	NC	Not Internally Connected.
11	RESERVED	Reserved. This pin must be connected to ground or left open.
12	SDO/ALT ADDRESS	Serial Data Output (SPI 4-Wire)/Alternate I²C Address Select (I²C).
13	SDA/SDI/SDIO	Serial Data (I²C)/Serial Data Input (SPI 4-Wire)/Serial Data Input and Output (SPI 3-Wire).
14	SCL/SCLK	Serial Communications Clock. SCL is the clock for I²C, and SCLK is the clock for SPI.

TYPICAL PERFORMANCE CHARACTERISTICS

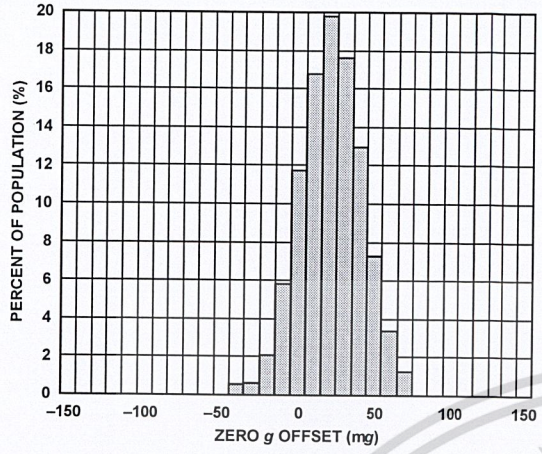


Figure 4. X-Axis Zero g Offset at 25°C, V_S = 2.5 V

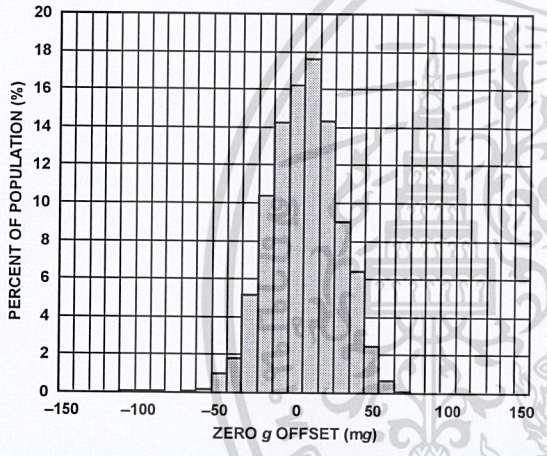


Figure 5. Y-Axis Zero g Offset at 25°C, V_S = 2.5 V

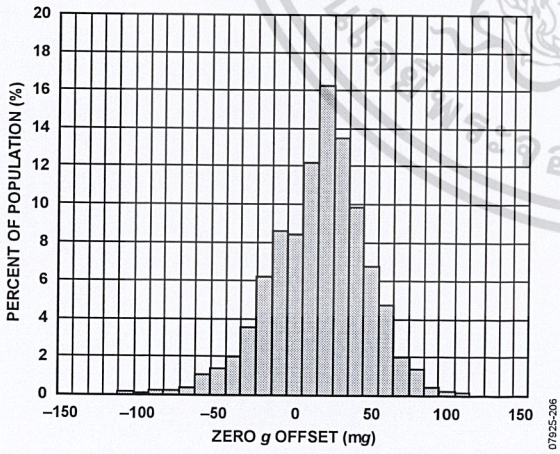


Figure 6. Z-Axis Zero g Offset at 25°C, V_S = 2.5 V

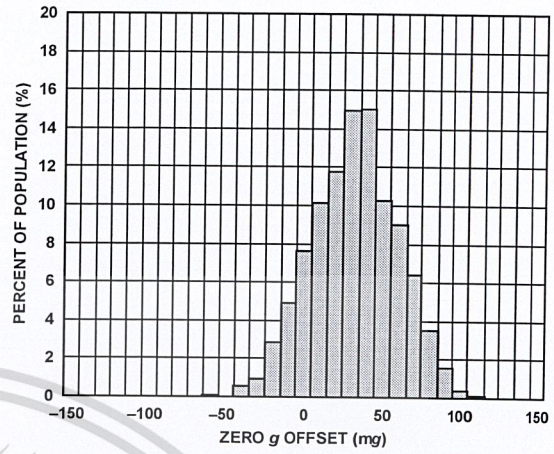


Figure 7. X-Axis Zero g Offset at 25°C, V_S = 3.3 V

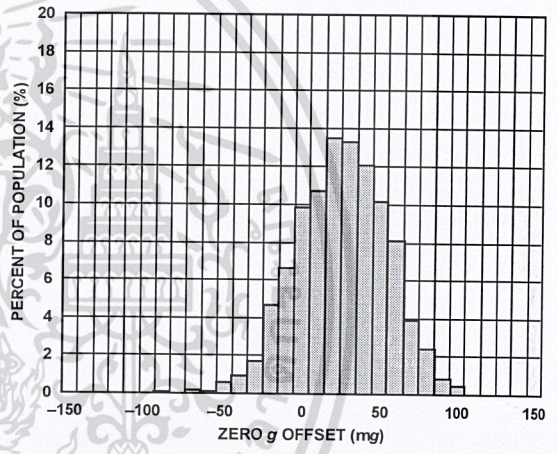


Figure 8. Y-Axis Zero g Offset at 25°C, V_S = 3.3 V

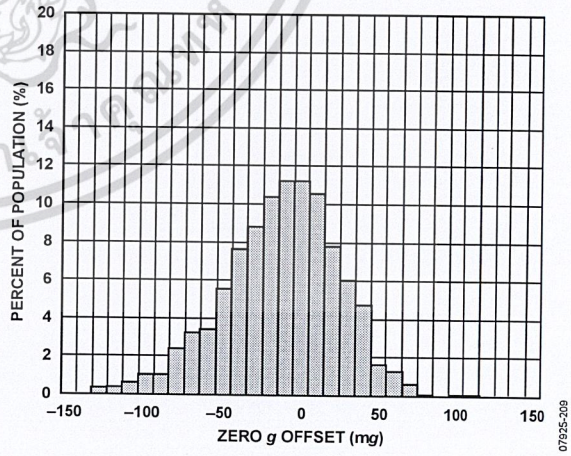


Figure 9. Z-Axis Zero g Offset at 25°C, V_S = 3.3 V

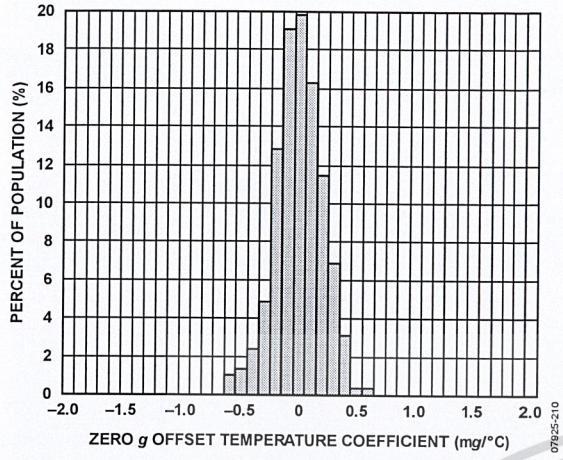


Figure 10. X-Axis Zero g Offset Temperature Coefficient, V_S = 2.5 V

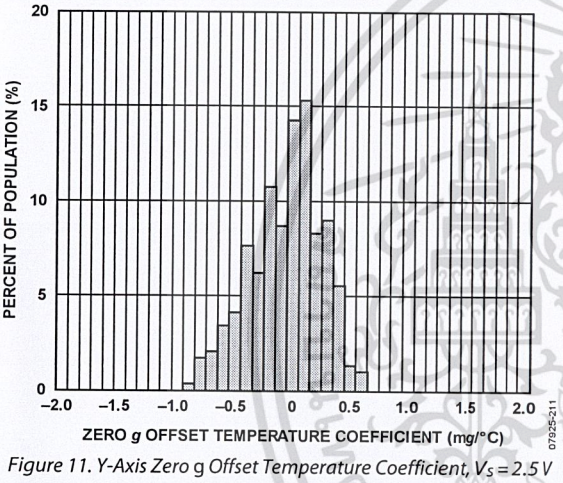


Figure 11. Y-Axis Zero g Offset Temperature Coefficient, V_S = 2.5 V

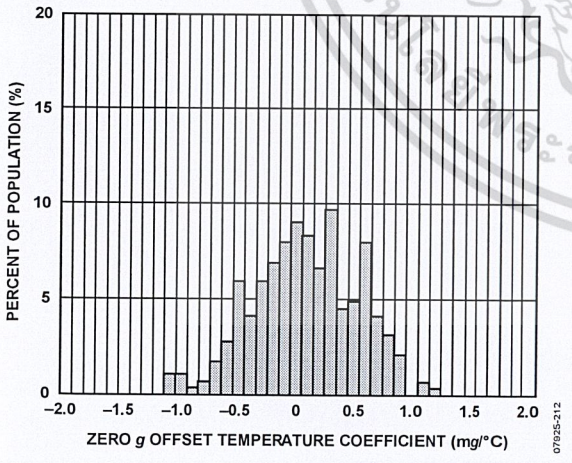


Figure 12. Z-Axis Zero g Offset Temperature Coefficient, V_S = 2.5 V

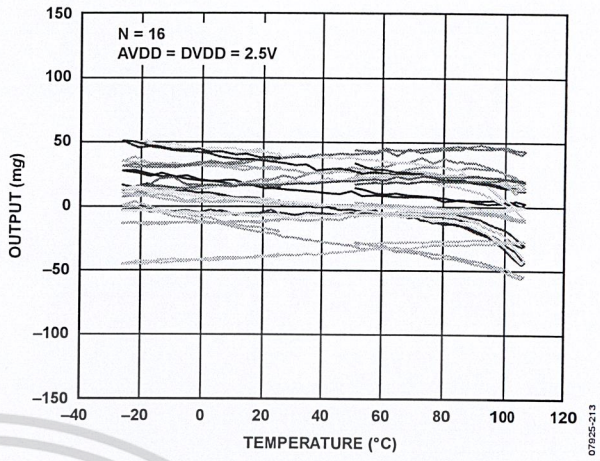


Figure 13. X-Axis Zero g Offset vs. Temperature—Eight Parts Soldered to PCB, V_S = 2.5 V

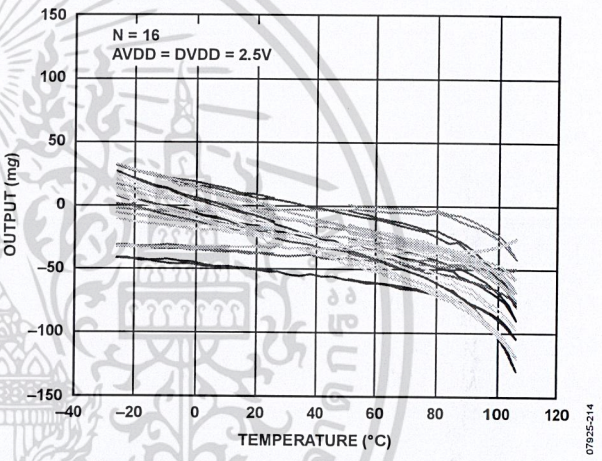


Figure 14. Y-Axis Zero g Offset vs. Temperature—Eight Parts Soldered to PCB, V_S = 2.5 V

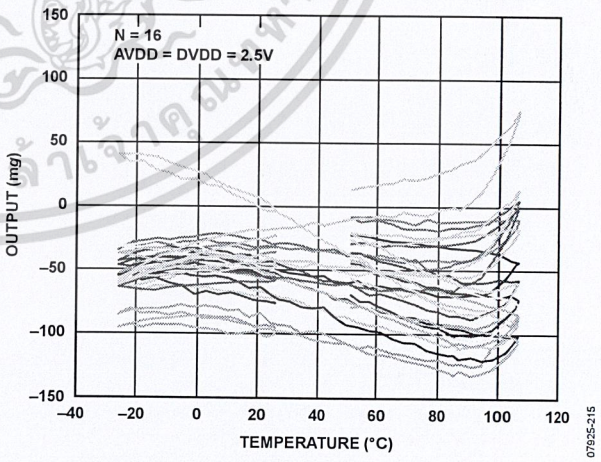


Figure 15. Z-Axis Zero g Offset vs. Temperature—Eight Parts Soldered to PCB, V_S = 2.5 V

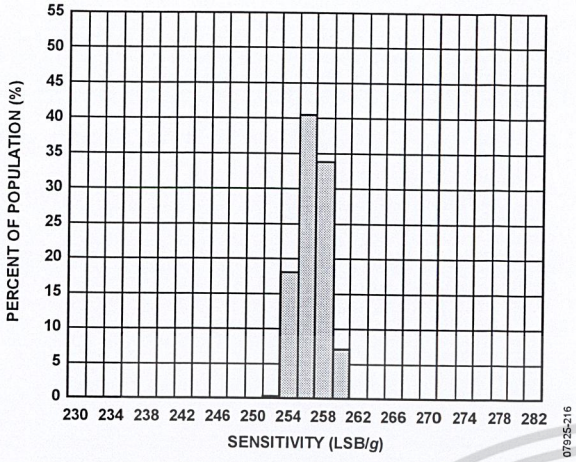


Figure 16. X-Axis Sensitivity at 25°C, V_S = 2.5 V, Full Resolution

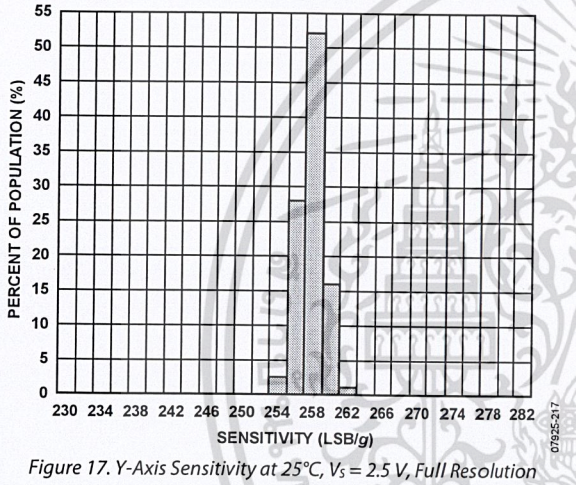


Figure 17. Y-Axis Sensitivity at 25°C, V_S = 2.5 V, Full Resolution

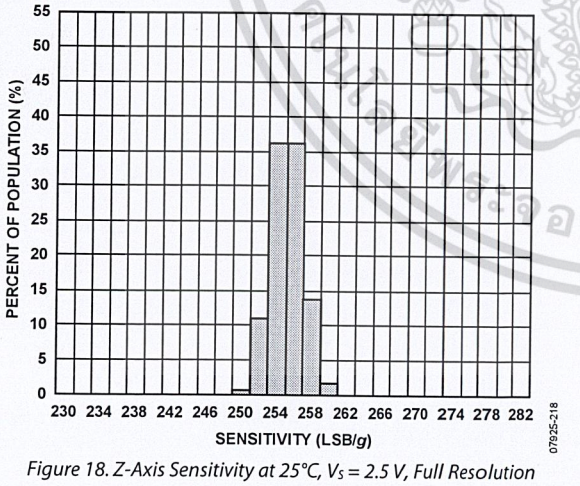


Figure 18. Z-Axis Sensitivity at 25°C, V_S = 2.5 V, Full Resolution

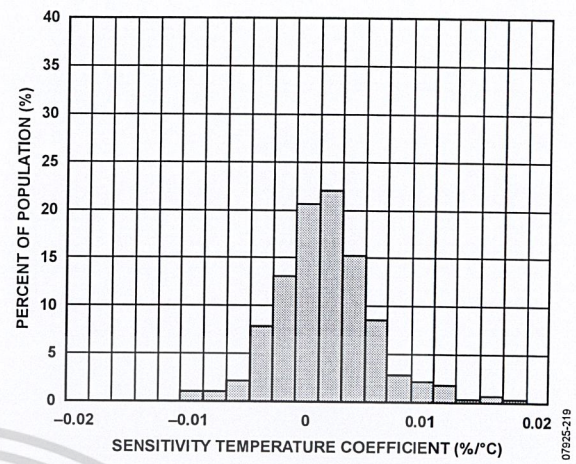


Figure 19. X-Axis Sensitivity Temperature Coefficient, V_S = 2.5 V

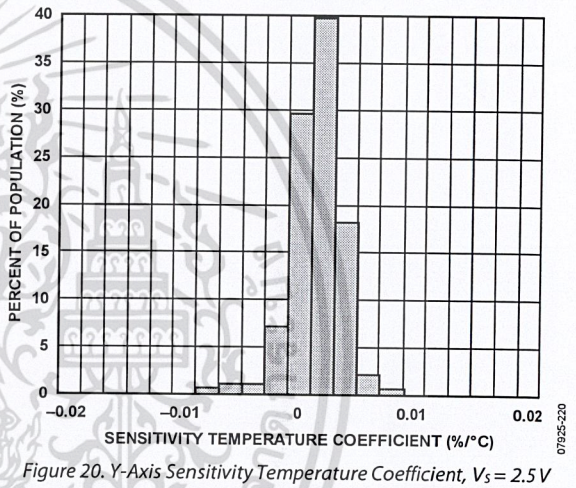


Figure 20. Y-Axis Sensitivity Temperature Coefficient, V_S = 2.5 V

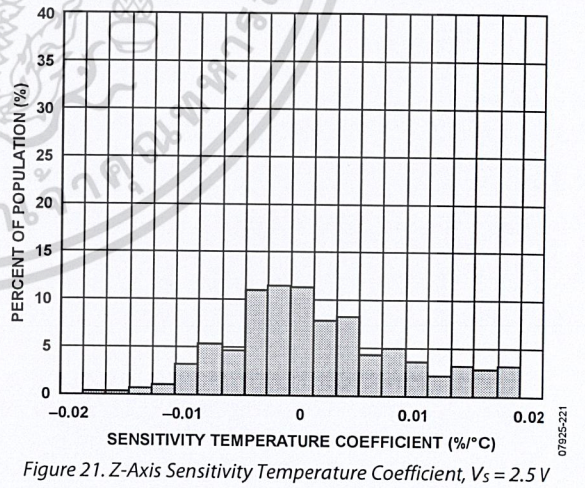


Figure 21. Z-Axis Sensitivity Temperature Coefficient, V_S = 2.5 V

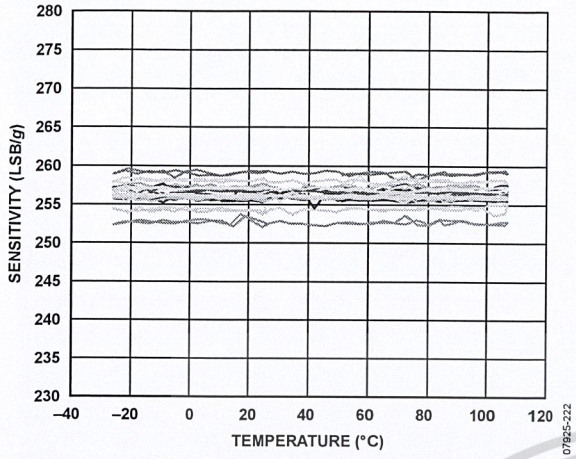


Figure 22. X-Axis Sensitivity vs. Temperature—
Eight Parts Soldered to PCB, V_S = 2.5 V, Full Resolution

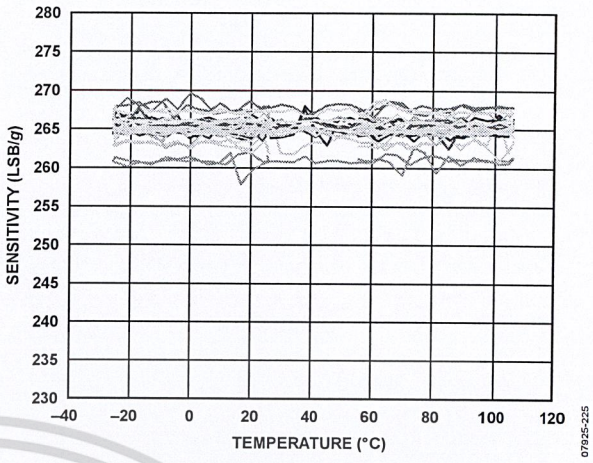


Figure 25. X-Axis Sensitivity vs. Temperature—
Eight Parts Soldered to PCB, V_S = 3.3 V, Full Resolution

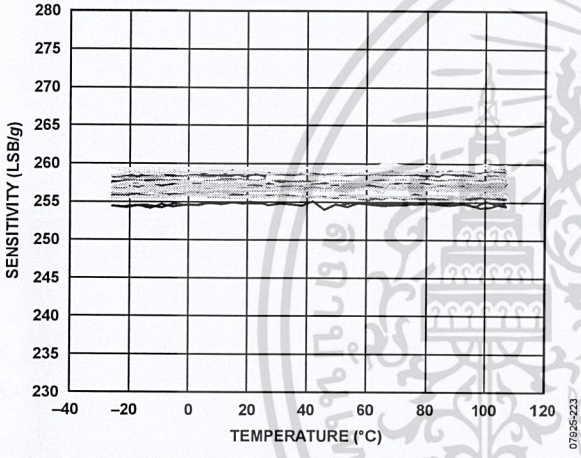


Figure 23. Y-Axis Sensitivity vs. Temperature—
Eight Parts Soldered to PCB, V_S = 2.5 V, Full Resolution

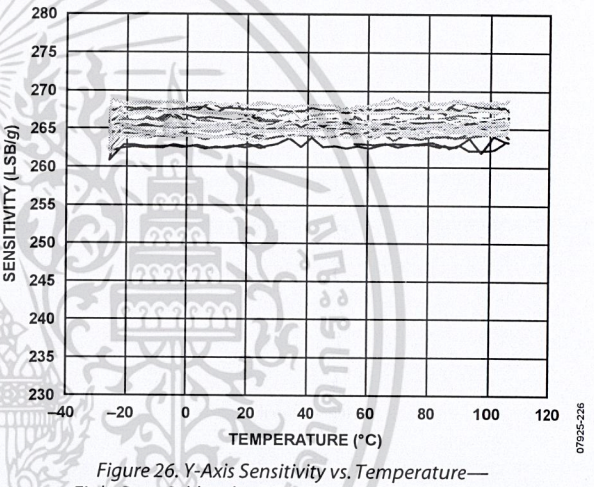


Figure 26. Y-Axis Sensitivity vs. Temperature—
Eight Parts Soldered to PCB, V_S = 3.3 V, Full Resolution

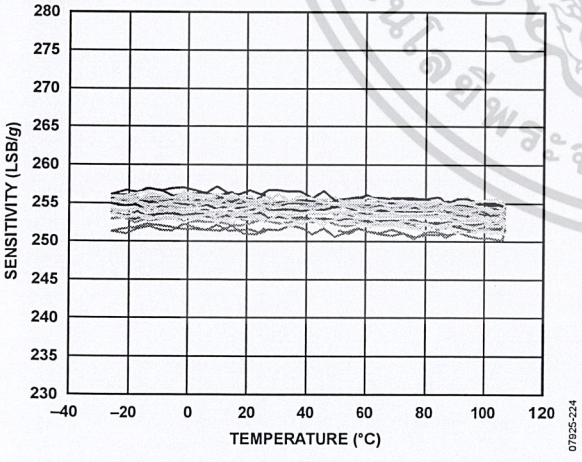


Figure 24. Z-Axis Sensitivity vs. Temperature—
Eight Parts Soldered to PCB, V_S = 2.5 V, Full Resolution

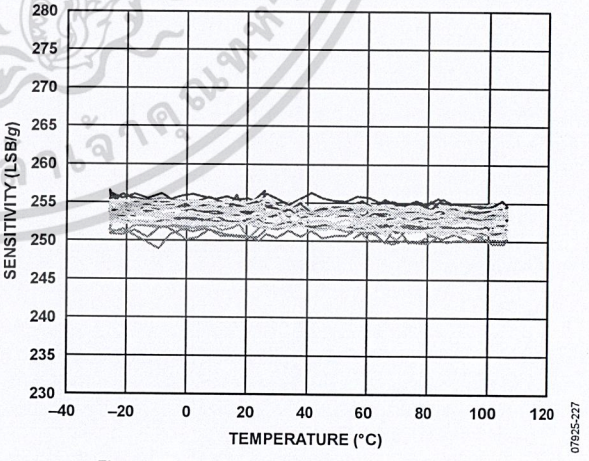


Figure 27. Z-Axis Sensitivity vs. Temperature—
Eight Parts Soldered to PCB, V_S = 3.3 V, Full Resolution

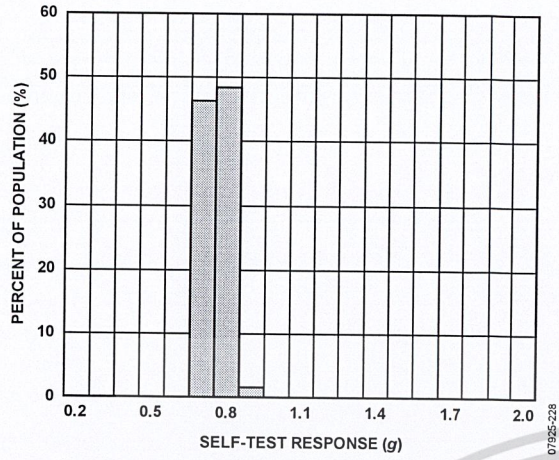


Figure 28. X-Axis Self-Test Response at 25°C, V_S = 2.5 V

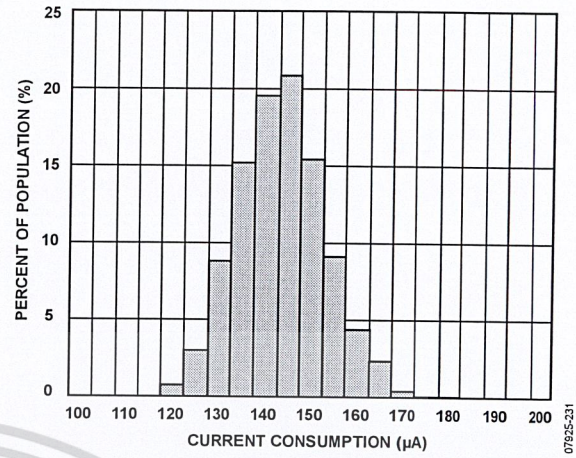


Figure 31. Current Consumption at 25°C, 100 Hz Output Data Rate, V_S = 2.5 V

Figure 29. Y-Axis Self-Test Response at 25°C, V_S = 2.5 V

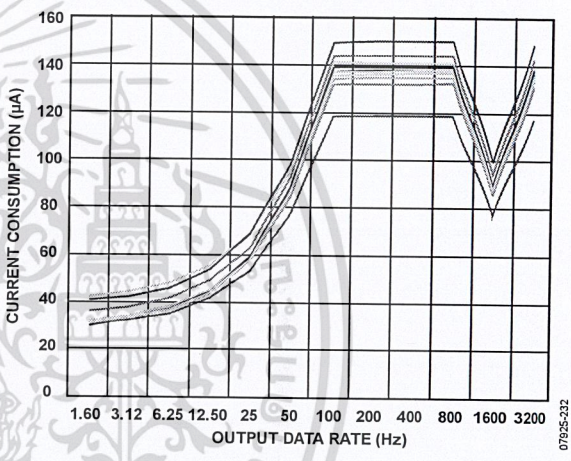


Figure 32. Current Consumption vs. Output Data Rate at 25°C—10 Parts, V_S = 2.5 V

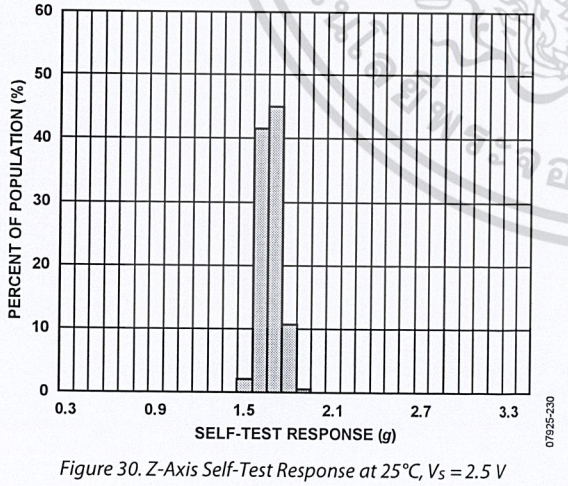


Figure 30. Z-Axis Self-Test Response at 25°C, V_S = 2.5 V

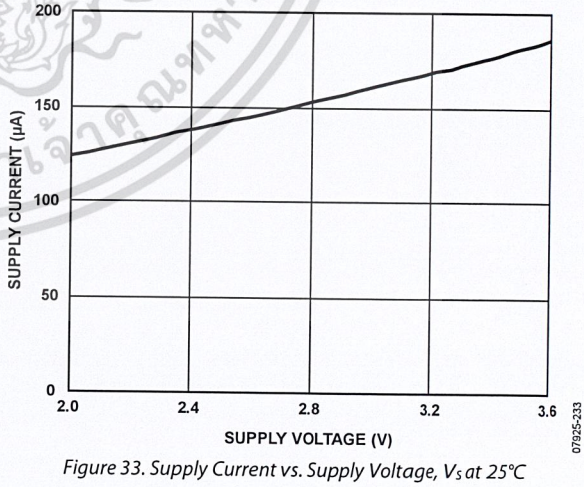


Figure 33. Supply Current vs. Supply Voltage, V_S at 25°C

THEORY OF OPERATION

The ADXL345 is a complete 3-axis acceleration measurement system with a selectable measurement range of ±2 *g*, ±4 *g*, ±8 *g*, or ±16 *g*. It measures both dynamic acceleration resulting from motion or shock and static acceleration, such as gravity, that allows the device to be used as a tilt sensor.

The sensor is a polysilicon surface-micromachined structure built on top of a silicon wafer. Polysilicon springs suspend the structure over the surface of the wafer and provide a resistance against forces due to applied acceleration.

Deflection of the structure is measured using differential capacitors that consist of independent fixed plates and plates attached to the moving mass. Acceleration deflects the proof mass and unbalances the differential capacitor, resulting in a sensor output whose amplitude is proportional to acceleration. Phase-sensitive demodulation is used to determine the magnitude and polarity of the acceleration.

POWER SEQUENCING

Power can be applied to V_S or $V_{DD\ I/O}$ in any sequence without damaging the ADXL345. All possible power-on modes are summarized in Table 6. The interface voltage level is set with the interface supply voltage, $V_{DD\ I/O}$, which must be present to ensure that the ADXL345 does not create a conflict on the communication bus. For single-supply operation, $V_{DD\ I/O}$ can be the same as the main supply, V_S. In a dual-supply application, however, $V_{DD\ I/O}$ can differ from V_S to accommodate the desired interface voltage, as long as V_S is greater than or equal to $V_{DD\ I/O}$.

After V_S is applied, the device enters standby mode, where power consumption is minimized and the device waits for $V_{DD\ I/O}$ to be applied and for the command to enter measurement mode to be received. (This command can be initiated by setting the measure bit (Bit D3) in the POWER_CTL register (Address 0x2D).) In addition, while the device is in standby mode, any register can be written to or read from to configure the part. It is recommended to configure the device in standby mode and then to enable measurement mode. Clearing the measure bit returns the device to the standby mode.

Table 6. Power Sequencing

Condition	V_S	$V_{DD\ I/O}$	Description
Power Off	Off	Off	The device is completely off, but there is a potential for a communication bus conflict.
Bus Disabled	On	Off	The device is on in standby mode, but communication is unavailable and creates a conflict on the communication bus. The duration of this state should be minimized during power-up to prevent a conflict.
Bus Enabled	Off	On	No functions are available, but the device does not create a conflict on the communication bus.
Standby or Measurement	On	On	At power-up, the device is in standby mode, awaiting a command to enter measurement mode, and all sensor functions are off. After the device is instructed to enter measurement mode, all sensor functions are available.

POWER SAVINGS

Power Modes

The ADXL345 automatically modulates its power consumption in proportion to its output data rate, as outlined in Table 7. If additional power savings is desired, a lower power mode is available. In this mode, the internal sampling rate is reduced, allowing for power savings in the 12.5 Hz to 400 Hz data rate range at the expense of slightly greater noise. To enter low power mode, set the LOW_POWER bit (Bit 4) in the BW_RATE register (Address 0x2C). The current consumption in low power mode is shown in Table 8 for cases where there is an advantage to using low power mode. Use of low power mode for a data rate not shown in Table 8 does not provide any advantage over the same data rate in normal power mode. Therefore, it is recommended that only data rates shown in Table 8 are used in low power mode. The current consumption values shown in Table 7 and Table 8 are for a V_S of 2.5 V.

Table 7. Typical Current Consumption vs. Data Rate (T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V)

Output Data Rate (Hz)	Bandwidth (Hz)	Rate Code	I_{DD} (μA)
3200	1600	1111	140
1600	800	1110	90
800	400	1101	140
400	200	1100	140
200	100	1011	140
100	50	1010	140
50	25	1001	90
25	12.5	1000	60
12.5	6.25	0111	50
6.25	3.13	0110	45
3.13	1.56	0101	40
1.56	0.78	0100	34
0.78	0.39	0011	23
0.39	0.20	0010	23
0.20	0.10	0001	23
0.10	0.05	0000	23

Table 8. Typical Current Consumption vs. Data Rate, Low Power Mode (T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V)

Output Data Rate (Hz)	Bandwidth (Hz)	Rate Code	I_{DD} (μA)
400	200	1100	90
200	100	1011	60
100	50	1010	50
50	25	1001	45
25	12.5	1000	40
12.5	6.25	0111	34

Auto Sleep Mode

Additional power can be saved if the ADXL345 automatically switches to sleep mode during periods of inactivity. To enable this feature, set the THRESH_INACT register (Address 0x25) and the TIME_INACT register (Address 0x26) each to a value that signifies inactivity (the appropriate value depends on the application), and then set the AUTO_SLEEP bit (Bit D4) and the link bit (Bit D5) in the POWER_CTL register (Address 0x2D). Current consumption at the sub-12.5 Hz data rates that are used in this mode is typically 23 μA for a V_S of 2.5 V.

Standby Mode

For even lower power operation, standby mode can be used. In standby mode, current consumption is reduced to 0.1 μA (typical). In this mode, no measurements are made. Standby mode is entered by clearing the measure bit (Bit D3) in the POWER_CTL register (Address 0x2D). Placing the device into standby mode preserves the contents of FIFO.

SERIAL COMMUNICATIONS

I²C and SPI digital communications are available. In both cases, the ADXL345 operates as a slave. I²C mode is enabled if the $\overline{CS}$ pin is tied high to $V_{DD\ I/O}$. The $\overline{CS}$ pin should always be tied high to $V_{DD\ I/O}$ or be driven by an external controller because there is no default mode if the $\overline{CS}$ pin is left unconnected. Therefore, not taking these precautions may result in an inability to communicate with the part. In SPI mode, the $\overline{CS}$ pin is controlled by the bus master. In both SPI and I²C modes of operation, data transmitted from the ADXL345 to the master device should be ignored during writes to the ADXL345.

SPI

For SPI, either 3- or 4-wire configuration is possible, as shown in the connection diagrams in Figure 34 and Figure 35. Clearing the SPI bit (Bit D6) in the DATA_FORMAT register (Address 0x31) selects 4-wire mode, whereas setting the SPI bit selects 3-wire mode. The maximum SPI clock speed is 5 MHz with 100 pF maximum loading, and the timing scheme follows clock polarity (CPOL) = 1 and clock phase (CPHA) = 1. If power is applied to the ADXL345 before the clock polarity and phase of the host processor are configured, the $\overline{CS}$ pin should be brought high before changing the clock polarity and phase. When using 3-wire SPI, it is recommended that the SDO pin be either pulled up to $V_{DD\ I/O}$ or pulled down to GND via a 10 kΩ resistor.

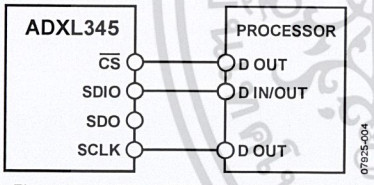


Figure 34. 3-Wire SPI Connection Diagram

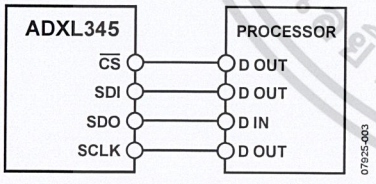


Figure 35. 4-Wire SPI Connection Diagram

$\overline{CS}$ is the serial port enable line and is controlled by the SPI master. This line must go low at the start of a transmission and high at the end of a transmission, as shown in Figure 36. SCLK is the serial port clock and is supplied by the SPI master. SCLK should idle high during a period of no transmission. SDI and SDO are the serial data input and output, respectively. Data is updated on the falling edge of SCLK and should be sampled on the rising edge of SCLK.

To read or write multiple bytes in a single transmission, the multiple-byte bit, located after the R/$\overline{W}$ bit in the first byte transfer (MB in Figure 36 to Figure 38), must be set. After the register addressing and the first byte of data, each subsequent set of clock pulses (eight clock pulses) causes the ADXL345 to point to the next register for a read or write. This shifting continues until the clock pulses cease and $\overline{CS}$ is deasserted. To perform reads or writes on different, nonsequential registers, $\overline{CS}$ must be deasserted between transmissions and the new register must be addressed separately.

The timing diagram for 3-wire SPI reads or writes is shown in Figure 38. The 4-wire equivalents for SPI writes and reads are shown in Figure 36 and Figure 37, respectively. For correct operation of the part, the logic thresholds and timing parameters in Table 9 and Table 10 must be met at all times.

Use of the 3200 Hz and 1600 Hz output data rates is only recommended with SPI communication rates greater than or equal to 2 MHz. The 800 Hz output data rate is recommended only for communication speeds greater than or equal to 400 kHz, and the remaining data rates scale proportionally. For example, the minimum recommended communication speed for a 200 Hz output data rate is 100 kHz. Operation at an output data rate above the recommended maximum may result in undesirable effects on the acceleration data, including missing samples or additional noise.

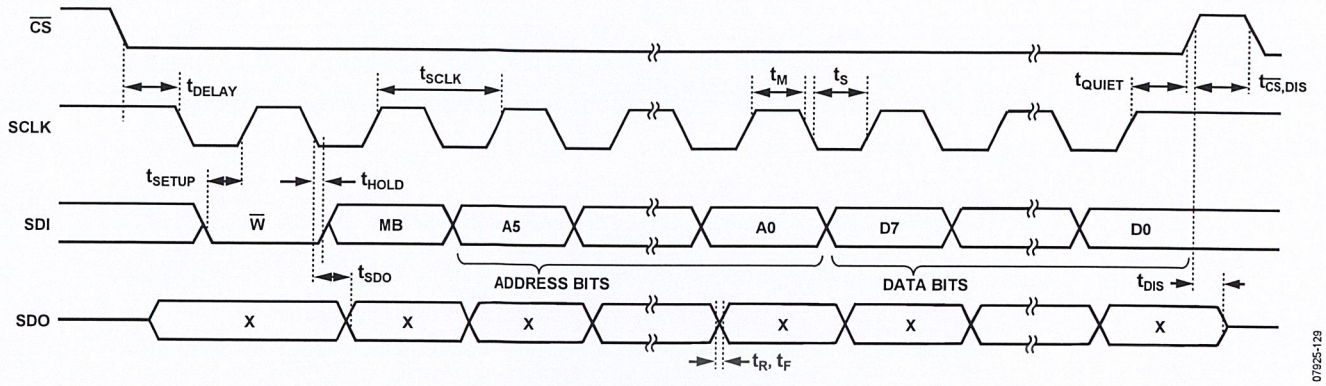


Figure 36. SPI 4-Wire Write

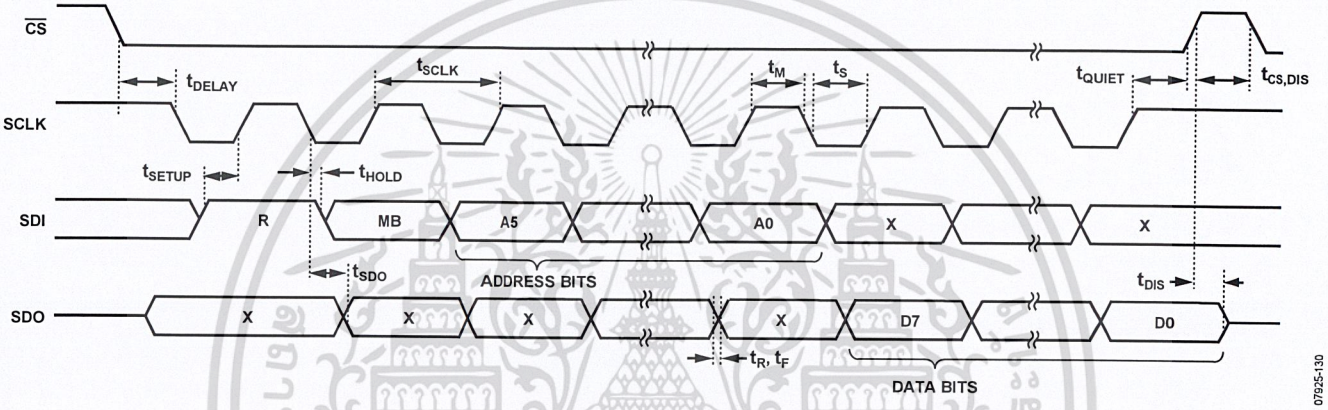
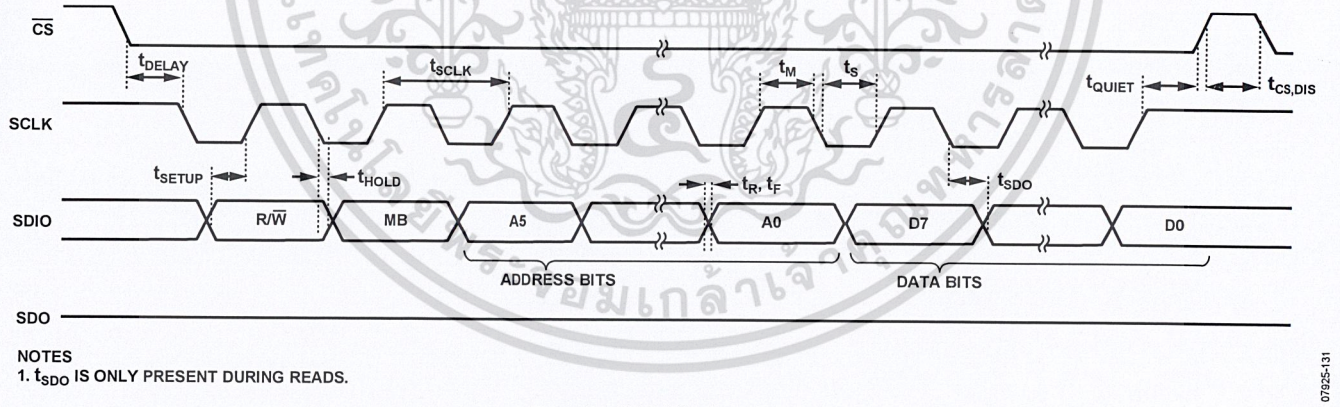


Figure 37. SPI 4-Wire Read



NOTES
1. t_{SDO} IS ONLY PRESENT DURING READS.

Figure 38. SPI 3-Wire Read/Write

Table 9. SPI Digital Input/Output

Parameter	Test Conditions	Limit[1] Min	Limit[1] Max	Unit
Digital Input				
Low Level Input Voltage (V_{IL})			$0.3 \times V_{DD\ I/O}$	V
High Level Input Voltage (V_{IH})		$0.7 \times V_{DD\ I/O}$		V
Low Level Input Current (I_{IL})	$V_{IN} = V_{DD\ I/O}$		0.1	μA
High Level Input Current (I_{IH})	$V_{IN} = 0$ V	−0.1		μA
Digital Output				
Low Level Output Voltage (V_{OL})	$I_{OL} = 10$ mA		$0.2 \times V_{DD\ I/O}$	V
High Level Output Voltage (V_{OH})	$I_{OH} = -4$ mA	$0.8 \times V_{DD\ I/O}$		V
Low Level Output Current (I_{OL})	$V_{OL} = V_{OL,\ max}$	10		mA
High Level Output Current (I_{OH})	$V_{OH} = V_{OH,\ min}$		−4	mA
Pin Capacitance	$f_{IN} = 1$ MHz, $V_{IN} = 2.5$ V		8	pF

[1] Limits based on characterization results, not production tested.

Table 10. SPI Timing ($T_A = 25°C$, $V_S = 2.5$ V, $V_{DD\ I/O} = 1.8$ V)[1]

Parameter	Limit[2, 3] Min	Limit[2, 3] Max	Unit	Description
f_{SCLK}		5	MHz	SPI clock frequency
t_{SCLK}	200		ns	1/(SPI clock frequency) mark-space ratio for the SCLK input is 40/60 to 60/40
t_{DELAY}	5		ns	$\overline{CS}$ falling edge to SCLK falling edge
t_{QUIET}	5		ns	SCLK rising edge to $\overline{CS}$ rising edge
t_{DIS}		10	ns	$\overline{CS}$ rising edge to SDO disabled
$t_{CS,DIS}$	150		ns	$\overline{CS}$ deassertion between SPI communications
t_S	$0.3 \times t_{SCLK}$		ns	SCLK low pulse width (space)
t_M	$0.3 \times t_{SCLK}$		ns	SCLK high pulse width (mark)
t_{SETUP}	5		ns	SDI valid before SCLK rising edge
t_{HOLD}	5		ns	SDI valid after SCLK rising edge
t_{SDO}		40	ns	SCLK falling edge to SDO/SDIO output transition
t_R[4]		20	ns	SDO/SDIO output high to output low transition
t_F[4]		20	ns	SDO/SDIO output low to output high transition

[1] The $\overline{CS}$, SCLK, SDI, and SDO pins are not internally pulled up or down; they must be driven for proper operation.
[2] Limits based on characterization results, characterized with $f_{SCLK} = 5$ MHz and bus load capacitance of 100 pF; not production tested.
[3] The timing values are measured corresponding to the input thresholds (V_{IL} and V_{IH}) given in Table 9.
[4] Output rise and fall times measured with capacitive load of 150 pF.

I²C

With $\overline{CS}$ tied high to $V_{DD\ I/O}$, the ADXL345 is in I²C mode, requiring a simple 2-wire connection, as shown in Figure 39. The ADXL345 conforms to the *UM10204 I²C-Bus Specification and User Manual*, Rev. 03—19 June 2007, available from NXP Semiconductor. It supports standard (100 kHz) and fast (400 kHz) data transfer modes if the bus parameters given in Table 11 and Table 12 are met. Single- or multiple-byte reads/writes are supported, as shown in Figure 40. With the ALT ADDRESS pin high, the 7-bit I²C address for the device is 0x1D, followed by the R/$\overline{W}$ bit. This translates to 0x3A for a write and 0x3B for a read. An alternate I²C address of 0x53 (followed by the R/$\overline{W}$ bit) can be chosen by grounding the ALT ADDRESS pin (Pin 12). This translates to 0xA6 for a write and 0xA7 for a read.

There are no internal pull-up or pull-down resistors for any unused pins; therefore, there is no known state or default state for the $\overline{CS}$ or ALT ADDRESS pin if left floating or unconnected. It is required that the $\overline{CS}$ pin be connected to $V_{DD\ I/O}$ and that the ALT ADDRESS pin be connected to either $V_{DD\ I/O}$ or GND when using I²C.

Due to communication speed limitations, the maximum output data rate when using 400 kHz I²C is 800 Hz and scales linearly with a change in the I²C communication speed. For example, using I²C at 100 kHz would limit the maximum ODR to 200 Hz. Operation at an output data rate above the recommended maximum may result in undesirable effect on the acceleration data, including missing samples or additional noise.

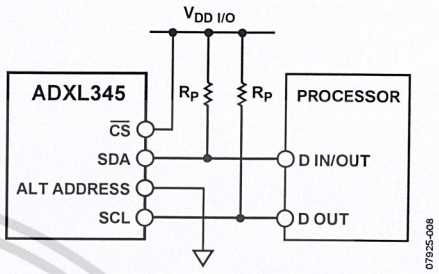


Figure 39. I²C Connection Diagram (Address 0x53)

If other devices are connected to the same I²C bus, the nominal operating voltage level of these other devices cannot exceed $V_{DD\ I/O}$ by more than 0.3 V. External pull-up resistors, R_P, are necessary for proper I²C operation. Refer to the *UM10204 I²C-Bus Specification and User Manual*, Rev. 03—19 June 2007, when selecting pull-up resistor values to ensure proper operation.

Table 11. I²C Digital Input/Output

Parameter	Test Conditions	Limit[1] Min	Limit[1] Max	Unit
Digital Input				
Low Level Input Voltage (V_{IL})			$0.3 \times V_{DD\ I/O}$	V
High Level Input Voltage (V_{IH})		$0.7 \times V_{DD\ I/O}$		V
Low Level Input Current (I_{IL})	$V_{IN} = V_{DD\ I/O}$		0.1	μA
High Level Input Current (I_{IH})	$V_{IN} = 0$ V	−0.1		μA
Digital Output				
Low Level Output Voltage (V_{OL})	$V_{DD\ I/O} < 2$ V, $I_{OL} = 3$ mA		$0.2 \times V_{DD\ I/O}$	V
	$V_{DD\ I/O} \geq 2$ V, $I_{OL} = 3$ mA		400	mV
Low Level Output Current (I_{OL})	$V_{OL} = V_{OL,\ max}$	3		mA
Pin Capacitance	$f_{IN} = 1$ MHz, $V_{IN} = 2.5$ V		8	pF

[1] Limits based on characterization results; not production tested.

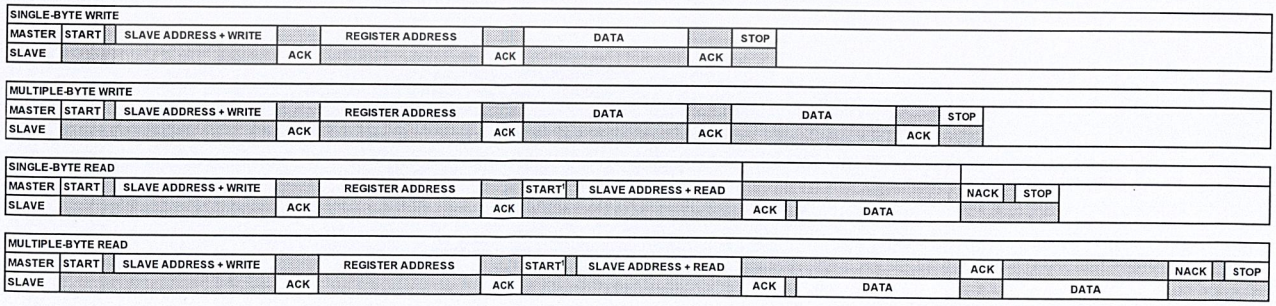


Figure 40. I²C Device Addressing

Table 12. I²C Timing (T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V)

Parameter	Limit[1, 2] Min	Max	Unit	Description
f_{SCL}		400	kHz	SCL clock frequency
t_1	2.5		µs	SCL cycle time
t_2	0.6		µs	t_{HIGH}, SCL high time
t_3	1.3		µs	t_{LOW}, SCL low time
t_4	0.6		µs	$t_{HD, STA}$, start/repeated start condition hold time
t_5	100		ns	$t_{SU, DAT}$, data setup time
t_6[3, 4, 5, 6]	0	0.9	µs	$t_{HD, DAT}$, data hold time
t_7	0.6		µs	$t_{SU, STA}$, setup time for repeated start
t_8	0.6		µs	$t_{SU, STO}$, stop condition setup time
t_9	1.3		µs	t_{BUF}, bus-free time between a stop condition and a start condition
t_{10}		300	ns	t_R, rise time of both SCL and SDA when receiving
	0		ns	t_R, rise time of both SCL and SDA when receiving or transmitting
t_{11}		250	ns	t_F, fall time of SDA when receiving
		300	ns	t_F, fall time of both SCL and SDA when transmitting
	20 + 0.1 C_b[7]		ns	t_F, fall time of both SCL and SDA when transmitting or receiving
C_b		400	pF	Capacitive load for each bus line

[1] Limits based on characterization results, with f_{SCL} = 400 kHz and a 3 mA sink current; not production tested.
[2] All values referred to the V_{IH} and the V_{IL} levels given in Table 11.
[3] t_6 is the data hold time that is measured from the falling edge of SCL. It applies to data in transmission and acknowledge.
[4] A transmitting device must internally provide an output hold time of at least 300 ns for the SDA signal (with respect to $V_{IH(min)}$ of the SCL signal) to bridge the undefined region of the falling edge of SCL.
[5] The maximum t_6 value must be met only if the device does not stretch the low period (t_3) of the SCL signal.
[6] The maximum value for t_6 is a function of the clock low time (t_3), the clock rise time (t_{10}), and the minimum data setup time ($t_{5(min)}$). This value is calculated as $t_{6(max)} = t_3 - t_{10} - t_{5(min)}$.
[7] C_b is the total capacitance of one bus line in picofarads.

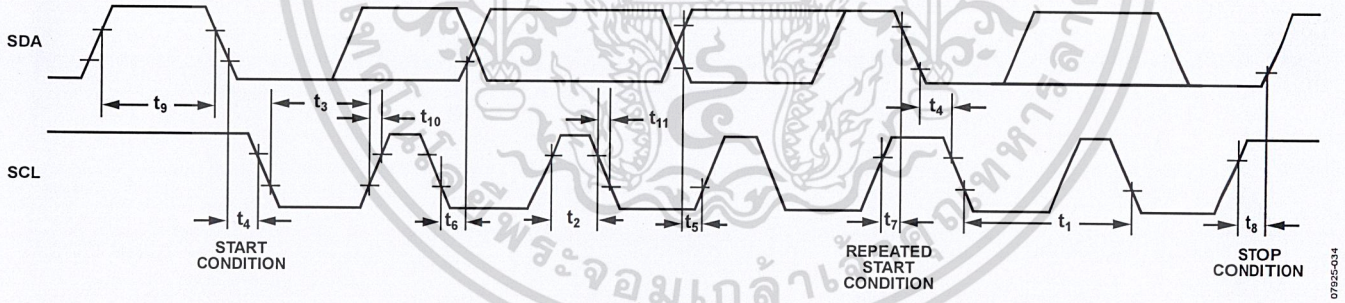


Figure 41. I²C Timing Diagram

INTERRUPTS

The ADXL345 provides two output pins for driving interrupts: INT1 and INT2. Both interrupt pins are push-pull, low impedance pins with output specifications shown in Table 13. The default configuration of the interrupt pins is active high. This can be changed to active low by setting the INT_INVERT bit in the DATA_FORMAT (Address 0x31) register. All functions can be used simultaneously, with the only limiting feature being that some functions may need to share interrupt pins.

Interrupts are enabled by setting the appropriate bit in the INT_ENABLE register (Address 0x2E) and are mapped to either the INT1 pin or the INT2 pin based on the contents of the INT_MAP register (Address 0x2F). When initially configuring the interrupt pins, it is recommended that the functions and interrupt mapping be done before enabling the interrupts. When changing the configuration of an interrupt, it is recommended that the interrupt be disabled first, by clearing the bit corresponding to that function in the INT_ENABLE register, and then the function be reconfigured before enabling the interrupt again. Configuration of the functions while the interrupts are disabled helps to prevent the accidental generation of an interrupt before desired.

The interrupt functions are latched and cleared by either reading the data registers (Address 0x32 to Address 0x37) until the interrupt condition is no longer valid for the data-related interrupts or by reading the INT_SOURCE register (Address 0x30) for the remaining interrupts. This section describes the interrupts that can be set in the INT_ENABLE register and monitored in the INT_SOURCE register.

DATA_READY

The DATA_READY bit is set when new data is available and is cleared when no new data is available.

SINGLE_TAP

The SINGLE_TAP bit is set when a single acceleration event that is greater than the value in the THRESH_TAP register (Address 0x1D) occurs for less time than is specified in the DUR register (Address 0x21).

DOUBLE_TAP

The DOUBLE_TAP bit is set when two acceleration events that are greater than the value in the THRESH_TAP register (Address 0x1D) occur for less time than is specified in the DUR register (Address 0x21), with the second tap starting after the time specified by the latent register (Address 0x22) but within the time specified in the window register (Address 0x23). See the Tap Detection section for more details.

Activity

The activity bit is set when acceleration greater than the value stored in the THRESH_ACT register (Address 0x24) is experienced on any participating axis, set by the ACT_INACT_CTL register (Address 0x27).

Inactivity

The inactivity bit is set when acceleration of less than the value stored in the THRESH_INACT register (Address 0x25) is experienced for more time than is specified in the TIME_INACT register (Address 0x26) on all participating axes, as set by the ACT_INACT_CTL register (Address 0x27). The maximum value for TIME_INACT is 255 sec.

FREE_FALL

The FREE_FALL bit is set when acceleration of less than the value stored in the THRESH_FF register (Address 0x28) is experienced for more time than is specified in the TIME_FF register (Address 0x29) on all axes (logical AND). The FREE_FALL interrupt differs from the inactivity interrupt as follows: all axes always participate and are logically AND'ed, the timer period is much smaller (1.28 sec maximum), and the mode of operation is always dc-coupled.

Watermark

The watermark bit is set when the number of samples in FIFO equals the value stored in the samples bits (Register FIFO_CTL, Address 0x38). The watermark bit is cleared automatically when FIFO is read, and the content returns to a value below the value stored in the samples bits.

Table 13. Interrupt Pin Digital Output

Parameter	Test Conditions	Limit[1] Min	Limit[1] Max	Unit
Digital Output				
Low Level Output Voltage (V_{OL})	I_{OL} = 300 µA		0.2 × $V_{DD\ I/O}$	V
High Level Output Voltage (V_{OH})	I_{OH} = −150 µA	0.8 × $V_{DD\ I/O}$		V
Low Level Output Current (I_{OL})	V_{OL} = $V_{OL, max}$	300		µA
High Level Output Current (I_{OH})	V_{OH} = $V_{OH, min}$		−150	µA
Pin Capacitance	f_{IN} = 1 MHz, V_{IN} = 2.5 V		8	pF
Rise/Fall Time				
Rise Time (t_R)[2]	C_{LOAD} = 150 pF		210	ns
Fall Time (t_F)[3]	C_{LOAD} = 150 pF		150	ns

[1] Limits based on characterization results, not production tested.
[2] Rise time is measured as the transition time from $V_{OL, max}$ to $V_{OH, min}$ of the interrupt pin.
[3] Fall time is measured as the transition time from $V_{OH, min}$ to $V_{OL, max}$ of the interrupt pin.

Overrun

The overrun bit is set when new data replaces unread data. The precise operation of the overrun function depends on the FIFO mode. In bypass mode, the overrun bit is set when new data replaces unread data in the DATAX, DATAY, and DATAZ registers (Address 0x32 to Address 0x37). In all other modes, the overrun bit is set when FIFO is filled. The overrun bit is automatically cleared when the contents of FIFO are read.

FIFO

The ADXL345 contains patent pending technology for an embedded memory management system with 32-level FIFO that can be used to minimize host processor burden. This buffer has four modes: bypass, FIFO, stream, and trigger (see FIFO Modes). Each mode is selected by the settings of the FIFO_MODE bits (Bits[D7:D6]) in the FIFO_CTL register (Address 0x38).

Bypass Mode

In bypass mode, FIFO is not operational and, therefore, remains empty.

FIFO Mode

In FIFO mode, data from measurements of the x-, y-, and z-axes are stored in FIFO. When the number of samples in FIFO equals the level specified in the samples bits of the FIFO_CTL register (Address 0x38), the watermark interrupt is set. FIFO continues accumulating samples until it is full (32 samples from measurements of the x-, y-, and z-axes) and then stops collecting data. After FIFO stops collecting data, the device continues to operate; therefore, features such as tap detection can be used after FIFO is full. The watermark interrupt continues to occur until the number of samples in FIFO is less than the value stored in the samples bits of the FIFO_CTL register.

Stream Mode

In stream mode, data from measurements of the x-, y-, and z-axes are stored in FIFO. When the number of samples in FIFO equals the level specified in the samples bits of the FIFO_CTL register (Address 0x38), the watermark interrupt is set. FIFO continues accumulating samples and holds the latest 32 samples from measurements of the x-, y-, and z-axes, discarding older data as new data arrives. The watermark interrupt continues occurring until the number of samples in FIFO is less than the value stored in the samples bits of the FIFO_CTL register.

Trigger Mode

In trigger mode, FIFO accumulates samples, holding the latest 32 samples from measurements of the x-, y-, and z-axes. After a trigger event occurs and an interrupt is sent to the INT1 or INT2 pin (determined by the trigger bit in the FIFO_CTL register), FIFO keeps the last n samples (where n is the value specified by the samples bits in the FIFO_CTL register) and then operates in FIFO mode, collecting new samples only when FIFO is not full. A delay of at least 5 µs should be present between the trigger event occurring and the start of reading data from the FIFO to allow the FIFO to discard and retain the necessary samples. Additional trigger events cannot be recognized until the trigger mode is reset. To reset the trigger mode, set the device to bypass mode and then set the device back to trigger mode. Note that the FIFO data should be read first because placing the device into bypass mode clears FIFO.

Retrieving Data from FIFO

The FIFO data is read through the DATAX, DATAY, and DATAZ registers (Address 0x32 to Address 0x37). When the FIFO is in FIFO, stream, or trigger mode, reads to the DATAX, DATAY, and DATAZ registers read data stored in the FIFO. Each time data is read from the FIFO, the oldest x-, y-, and z-axes data are placed into the DATAX, DATAY and DATAZ registers.

If a single-byte read operation is performed, the remaining bytes of data for the current FIFO sample are lost. Therefore, all axes of interest should be read in a burst (or multiple-byte) read operation. To ensure that the FIFO has completely popped (that is, that new data has completely moved into the DATAX, DATAY, and DATAZ registers), there must be at least 5 µs between the end of reading the data registers and the start of a new read of the FIFO or a read of the FIFO_STATUS register (Address 0x39). The end of reading a data register is signified by the transition from Register 0x37 to Register 0x38 or by the $\overline{CS}$ pin going high.

For SPI operation at 1.6 MHz or less, the register addressing portion of the transmission is a sufficient delay to ensure that the FIFO has completely popped. For SPI operation greater than 1.6 MHz, it is necessary to deassert the $\overline{CS}$ pin to ensure a total delay of 5 µs; otherwise, the delay is not sufficient. The total delay necessary for 5 MHz operation is at most 3.4 µs. This is not a concern when using I^2C mode because the communication rate is low enough to ensure a sufficient delay between FIFO reads.

SELF-TEST

The ADXL345 incorporates a self-test feature that effectively tests its mechanical and electronic systems simultaneously. When the self-test function is enabled (via the SELF_TEST bit in the DATA_FORMAT register, Address 0x31), an electrostatic force is exerted on the mechanical sensor. This electrostatic force moves the mechanical sensing element in the same manner as acceleration, and it is additive to the acceleration experienced by the device. This added electrostatic force results in an output change in the x-, y-, and z-axes. Because the electrostatic force is proportional to V_S^2, the output change varies with V_S. This effect is shown in Figure 42. The scale factors shown in Table 14 can be used to adjust the expected self-test output limits for different supply voltages, V_S. The self-test feature of the ADXL345 also exhibits a bimodal behavior. However, the limits shown in Table 1 and Table 15 to Table 18 are valid for both potential self-test values due to bimodality. Use of the self-test feature at data rates less than 100 Hz or at 1600 Hz may yield values outside these limits. Therefore, the part must be in normal power operation (LOW_POWER bit = 0 in BW_RATE register, Address 0x2C) and be placed into a data rate of 100 Hz through 800 Hz or 3200 Hz for the self-test function to operate correctly.

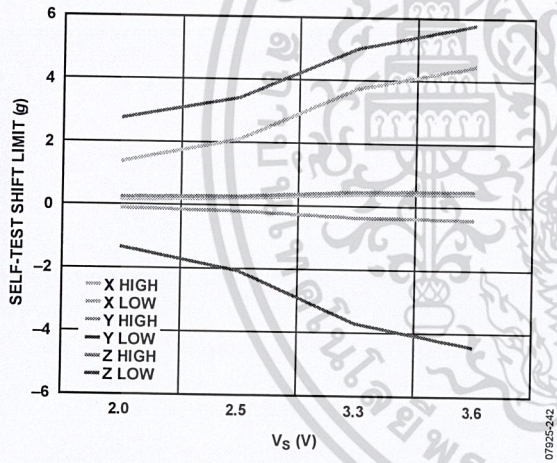


Figure 42. Self-Test Output Change Limits vs. Supply Voltage

Table 14. Self-Test Output Scale Factors for Different Supply Voltages, V_S

Supply Voltage, V_S (V)	X-Axis, Y-Axis	Z-Axis
2.00	0.64	0.8
2.50	1.00	1.00
3.30	1.77	1.47
3.60	2.11	1.69

Table 15. Self-Test Output in LSB for ±2 *g*, 10-Bit or Full Resolution (T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V)

Axis	Min	Max	Unit
X	50	540	LSB
Y	–540	–540	LSB
Z	75	875	LSB

Table 16. Self-Test Output in LSB for ±4 *g*, 10-Bit Resolution (T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V)

Axis	Min	Max	Unit
X	25	270	LSB
Y	–270	–25	LSB
Z	38	438	LSB

Table 17. Self-Test Output in LSB for ±8 *g*, 10-Bit Resolution (T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V)

Axis	Min	Max	Unit
X	12	135	LSB
Y	–135	–12	LSB
Z	19	219	LSB

Table 18. Self-Test Output in LSB for ±16 *g*, 10-Bit Resolution (T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V)

Axis	Min	Max	Unit
X	6	67	LSB
Y	–67	–6	LSB
Z	10	110	LSB

REGISTER MAP

Table 19.

Address					
Hex	Dec	Name	Type	Reset Value	Description
0x00	0	DEVID	R	11100101	Device ID
0x01 to 0x1C	1 to 28	Reserved			Reserved; do not access
0x1D	29	THRESH_TAP	R/$\overline{W}$	00000000	Tap threshold
0x1E	30	OFSX	R/$\overline{W}$	00000000	X-axis offset
0x1F	31	OFSY	R/$\overline{W}$	00000000	Y-axis offset
0x20	32	OFSZ	R/$\overline{W}$	00000000	Z-axis offset
0x21	33	DUR	R/$\overline{W}$	00000000	Tap duration
0x22	34	Latent	R/$\overline{W}$	00000000	Tap latency
0x23	35	Window	R/$\overline{W}$	00000000	Tap window
0x24	36	THRESH_ACT	R/$\overline{W}$	00000000	Activity threshold
0x25	37	THRESH_INACT	R/$\overline{W}$	00000000	Inactivity threshold
0x26	38	TIME_INACT	R/$\overline{W}$	00000000	Inactivity time
0x27	39	ACT_INACT_CTL	R/$\overline{W}$	00000000	Axis enable control for activity and inactivity detection
0x28	40	THRESH_FF	R/$\overline{W}$	00000000	Free-fall threshold
0x29	41	TIME_FF	R/$\overline{W}$	00000000	Free-fall time
0x2A	42	TAP_AXES	R/$\overline{W}$	00000000	Axis control for single tap/double tap
0x2B	43	ACT_TAP_STATUS	R	00000000	Source of single tap/double tap
0x2C	44	BW_RATE	R/$\overline{W}$	00001010	Data rate and power mode control
0x2D	45	POWER_CTL	R/$\overline{W}$	00000000	Power-saving features control
0x2E	46	INT_ENABLE	R/$\overline{W}$	00000000	Interrupt enable control
0x2F	47	INT_MAP	R/$\overline{W}$	00000000	Interrupt mapping control
0x30	48	INT_SOURCE	R	00000010	Source of interrupts
0x31	49	DATA_FORMAT	R/$\overline{W}$	00000000	Data format control
0x32	50	DATAX0	R	00000000	X-Axis Data 0
0x33	51	DATAX1	R	00000000	X-Axis Data 1
0x34	52	DATAY0	R	00000000	Y-Axis Data 0
0x35	53	DATAY1	R	00000000	Y-Axis Data 1
0x36	54	DATAZ0	R	00000000	Z-Axis Data 0
0x37	55	DATAZ1	R	00000000	Z-Axis Data 1
0x38	56	FIFO_CTL	R/$\overline{W}$	00000000	FIFO control
0x39	57	FIFO_STATUS	R	00000000	FIFO status

REGISTER DEFINITIONS

Register 0x00—DEVID (Read Only)

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	0	0	1	0	1

The DEVID register holds a fixed device ID code of 0xE5 (345 octal).

Register 0x1D—THRESH_TAP (Read/Write)

The THRESH_TAP register is eight bits and holds the threshold value for tap interrupts. The data format is unsigned, therefore, the magnitude of the tap event is compared with the value in THRESH_TAP for normal tap detection. The scale factor is 62.5 mg/LSB (that is, 0xFF = 16 *g*). A value of 0 may result in undesirable behavior if single tap/double tap interrupts are enabled.

Register 0x1E, Register 0x1F, Register 0x20—OFSX, OFSY, OFSZ (Read/Write)

The OFSX, OFSY, and OFSZ registers are each eight bits and offer user-set offset adjustments in twos complement format with a scale factor of 15.6 mg/LSB (that is, 0x7F = 2 *g*). The value stored in the offset registers is automatically added to the acceleration data, and the resulting value is stored in the output data registers. For additional information regarding offset calibration and the use of the offset registers, refer to the Offset Calibration section.

Register 0x21—DUR (Read/Write)

The DUR register is eight bits and contains an unsigned time value representing the maximum time that an event must be above the THRESH_TAP threshold to qualify as a tap event. The scale factor is 625 µs/LSB. A value of 0 disables the single tap/double tap functions.

Register 0x22—Latent (Read/Write)

The latent register is eight bits and contains an unsigned time value representing the wait time from the detection of a tap event to the start of the time window (defined by the window register) during which a possible second tap event can be detected. The scale factor is 1.25 ms/LSB. A value of 0 disables the double tap function.

Register 0x23—Window (Read/Write)

The window register is eight bits and contains an unsigned time value representing the amount of time after the expiration of the latency time (determined by the latent register) during which a second valid tap can begin. The scale factor is 1.25 ms/LSB. A value of 0 disables the double tap function.

Register 0x24—THRESH_ACT (Read/Write)

The THRESH_ACT register is eight bits and holds the threshold value for detecting activity. The data format is unsigned, so the magnitude of the activity event is compared with the value in the THRESH_ACT register. The scale factor is 62.5 mg/LSB. A value of 0 may result in undesirable behavior if the activity interrupt is enabled.

Register 0x25—THRESH_INACT (Read/Write)

The THRESH_INACT register is eight bits and holds the threshold value for detecting inactivity. The data format is unsigned, so the magnitude of the inactivity event is compared with the value in the THRESH_INACT register. The scale factor is 62.5 mg/LSB. A value of 0 may result in undesirable behavior if the inactivity interrupt is enabled.

Register 0x26—TIME_INACT (Read/Write)

The TIME_INACT register is eight bits and contains an unsigned time value representing the amount of time that acceleration must be less than the value in the THRESH_INACT register for inactivity to be declared. The scale factor is 1 sec/LSB. Unlike the other interrupt functions, which use unfiltered data (see the Threshold section), the inactivity function uses filtered output data. At least one output sample must be generated for the inactivity interrupt to be triggered. This results in the function appearing unresponsive if the TIME_INACT register is set to a value less than the time constant of the output data rate. A value of 0 results in an interrupt when the output data is less than the value in the THRESH_INACT register.

Register 0x27—ACT_INACT_CTL (Read/Write)

D7	D6	D5	D4
ACT ac/dc	ACT_X enable	ACT_Y enable	ACT_Z enable
D3	**D2**	**D1**	**D0**
INACT ac/dc	INACT_X enable	INACT_Y enable	INACT_Z enable

ACT AC/DC and INACT AC/DC Bits

A setting of 0 selects dc-coupled operation, and a setting of 1 enables ac-coupled operation. In dc-coupled operation, the current acceleration magnitude is compared directly with THRESH_ACT and THRESH_INACT to determine whether activity or inactivity is detected.

In ac-coupled operation for activity detection, the acceleration value at the start of activity detection is taken as a reference value. New samples of acceleration are then compared to this reference value, and if the magnitude of the difference exceeds the THRESH_ACT value, the device triggers an activity interrupt.

Similarly, in ac-coupled operation for inactivity detection, a reference value is used for comparison and is updated whenever the device exceeds the inactivity threshold. After the reference value is selected, the device compares the magnitude of the difference between the reference value and the current acceleration with THRESH_INACT. If the difference is less than the value in THRESH_INACT for the time in TIME_INACT, the device is considered inactive and the inactivity interrupt is triggered.

ACT_x Enable Bits and INACT_x Enable Bits

A setting of 1 enables x-, y-, or z-axis participation in detecting activity or inactivity. A setting of 0 excludes the selected axis from participation. If all axes are excluded, the function is disabled. For activity detection, all participating axes are logically OR'ed, causing the activity function to trigger when any of the participating axes exceeds the threshold. For inactivity detection, all participating axes are logically AND'ed, causing the inactivity function to trigger only if all participating axes are below the threshold for the specified time.

Register 0x28—THRESH_FF (Read/Write)

The THRESH_FF register is eight bits and holds the threshold value, in unsigned format, for free-fall detection. The acceleration on all axes is compared with the value in THRESH_FF to determine if a free-fall event occurred. The scale factor is 62.5 *m*g/LSB. Note that a value of 0 *m*g may result in undesirable behavior if the free-fall interrupt is enabled. Values between 300 *m*g and 600 *m*g (0x05 to 0x09) are recommended.

Register 0x29—TIME_FF (Read/Write)

The TIME_FF register is eight bits and stores an unsigned time value representing the minimum time that the value of all axes must be less than THRESH_FF to generate a free-fall interrupt. The scale factor is 5 ms/LSB. A value of 0 may result in undesirable behavior if the free-fall interrupt is enabled. Values between 100 ms and 350 ms (0x14 to 0x46) are recommended.

Register 0x2A—TAP_AXES (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	Suppress	TAP_X enable	TAP_Y enable	TAP_Z enable

Suppress Bit

Setting the suppress bit suppresses double tap detection if acceleration greater than the value in THRESH_TAP is present between taps. See the Tap Detection section for more details.

TAP_x Enable Bits

A setting of 1 in the TAP_X enable, TAP_Y enable, or TAP_Z enable bit enables x-, y-, or z-axis participation in tap detection. A setting of 0 excludes the selected axis from participation in tap detection.

Register 0x2B—ACT_TAP_STATUS (Read Only)

D7	D6	D5	D4	D3	D2	D1	D0
0	ACT_X source	ACT_Y source	ACT_Z source	Asleep	TAP_X source	TAP_Y source	TAP_Z source

ACT_x Source and TAP_x Source Bits

These bits indicate the first axis involved in a tap or activity event. A setting of 1 corresponds to involvement in the event, and a setting of 0 corresponds to no involvement. When new data is available, these bits are not cleared but are overwritten by the new data. The ACT_TAP_STATUS register should be read before clearing the interrupt. Disabling an axis from participation clears the corresponding source bit when the next activity or single tap/double tap event occurs.

Asleep Bit

A setting of 1 in the asleep bit indicates that the part is asleep, and a setting of 0 indicates that the part is not asleep. This bit toggles only if the device is configured for auto sleep. See the AUTO_SLEEP Bit section for more information on autosleep mode.

Register 0x2C—BW_RATE (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	LOW_POWER	Rate			

LOW_POWER Bit

A setting of 0 in the LOW_POWER bit selects normal operation, and a setting of 1 selects reduced power operation, which has somewhat higher noise (see the Power Modes section for details).

Rate Bits

These bits select the device bandwidth and output data rate (see Table 7 and Table 8 for details). The default value is 0x0A, which translates to a 100 Hz output data rate. An output data rate should be selected that is appropriate for the communication protocol and frequency selected. Selecting too high of an output data rate with a low communication speed results in samples being discarded.

Register 0x2D—POWER_CTL (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
0	0	Link	AUTO_SLEEP	Measure	Sleep	Wakeup	

Link Bit

A setting of 1 in the link bit with both the activity and inactivity functions enabled delays the start of the activity function until inactivity is detected. After activity is detected, inactivity detection begins, preventing the detection of activity. This bit serially links the activity and inactivity functions. When this bit is set to 0, the inactivity and activity functions are concurrent. Additional information can be found in the Link Mode section.

When clearing the link bit, it is recommended that the part be placed into standby mode and then set back to measurement mode with a subsequent write. This is done to ensure that the device is properly biased if sleep mode is manually disabled; otherwise, the first few samples of data after the link bit is cleared may have additional noise, especially if the device was asleep when the bit was cleared.

AUTO_SLEEP Bit

If the link bit is set, a setting of 1 in the AUTO_SLEEP bit enables the auto-sleep functionality. In this mode, the ADXL345 automatically switches to sleep mode if the inactivity function is enabled and inactivity is detected (that is, when acceleration is below the THRESH_INACT value for at least the time indicated by TIME_INACT). If activity is also enabled, the ADXL345 automatically wakes up from sleep after detecting activity and returns to operation at the output data rate set in the BW_RATE register. A setting of 0 in the AUTO_SLEEP bit disables automatic switching to sleep mode. See the description of the Sleep Bit in this section for more information on sleep mode.

If the link bit is not set, the AUTO_SLEEP feature is disabled and setting the AUTO_SLEEP bit does not have an impact on device operation. Refer to the Link Bit section or the Link Mode section for more information on utilization of the link feature.

When clearing the AUTO_SLEEP bit, it is recommended that the part be placed into standby mode and then set back to measurement mode with a subsequent write. This is done to ensure that the device is properly biased if sleep mode is manually disabled; otherwise, the first few samples of data after the AUTO_SLEEP bit is cleared may have additional noise, especially if the device was asleep when the bit was cleared.

Measure Bit

A setting of 0 in the measure bit places the part into standby mode, and a setting of 1 places the part into measurement mode. The ADXL345 powers up in standby mode with minimum power consumption.

Sleep Bit

A setting of 0 in the sleep bit puts the part into the normal mode of operation, and a setting of 1 places the part into sleep mode. Sleep mode suppresses DATA_READY, stops transmission of data to FIFO, and switches the sampling rate to one specified by the wakeup bits. In sleep mode, only the activity function can be used. When the DATA_READY interrupt is suppressed, the output data registers (Register 0x32 to Register 0x37) are still updated at the sampling rate set by the wakeup bits (D1:D0).

When clearing the sleep bit, it is recommended that the part be placed into standby mode and then set back to measurement mode with a subsequent write. This is done to ensure that the device is properly biased if sleep mode is manually disabled; otherwise, the first few samples of data after the sleep bit is cleared may have additional noise, especially if the device was asleep when the bit was cleared.

Wakeup Bits

These bits control the frequency of readings in sleep mode as described in Table 20.

Table 20. Frequency of Readings in Sleep Mode

Setting		
D1	**D0**	**Frequency (Hz)**
0	0	8
0	1	4
1	0	2
1	1	1

Register 0x2E—INT_ENABLE (Read/Write)

D7 DATA_READY	**D6** SINGLE_TAP	**D5** DOUBLE_TAP	**D4** Activity
D3 Inactivity	**D2** FREE_FALL	**D1** Watermark	**D0** Overrun

Setting bits in this register to a value of 1 enables their respective functions to generate interrupts, whereas a value of 0 prevents the functions from generating interrupts. The DATA_READY, watermark, and overrun bits enable only the interrupt output; the functions are always enabled. It is recommended that interrupts be configured before enabling their outputs.

Register 0x2F—INT_MAP (R/$\overline{W}$)

D7 DATA_READY	**D6** SINGLE_TAP	**D5** DOUBLE_TAP	**D4** Activity
D3 Inactivity	**D2** FREE_FALL	**D1** Watermark	**D0** Overrun

Any bits set to 0 in this register send their respective interrupts to the INT1 pin, whereas bits set to 1 send their respective interrupts to the INT2 pin. All selected interrupts for a given pin are ORed.

Register 0x30—INT_SOURCE (Read Only)

D7 DATA_READY	**D6** SINGLE_TAP	**D5** DOUBLE_TAP	**D4** Activity
D3 Inactivity	**D2** FREE_FALL	**D1** Watermark	**D0** Overrun

Bits set to 1 in this register indicate that their respective functions have triggered an event, whereas a value of 0 indicates that the corresponding event has not occurred. The DATA_READY, watermark, and overrun bits are always set if the corresponding events occur, regardless of the INT_ENABLE register settings, and are cleared by reading data from the DATAX, DATAY, and DATAZ registers. The DATA_READY and watermark bits may require multiple reads, as indicated in the FIFO mode descriptions in the FIFO section. Other bits, and the corresponding interrupts, are cleared by reading the INT_SOURCE register.

Register 0x31—DATA_FORMAT (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
SELF_TEST	SPI	INT_INVERT	0	FULL_RES	Justify	Range	

The DATA_FORMAT register controls the presentation of data to Register 0x32 through Register 0x37. All data, except that for the ±16 *g* range, must be clipped to avoid rollover.

SELF_TEST Bit

A setting of 1 in the SELF_TEST bit applies a self-test force to the sensor, causing a shift in the output data. A value of 0 disables the self-test force.

SPI Bit

A value of 1 in the SPI bit sets the device to 3-wire SPI mode, and a value of 0 sets the device to 4-wire SPI mode.

INT_INVERT Bit

A value of 0 in the INT_INVERT bit sets the interrupts to active high, and a value of 1 sets the interrupts to active low.

FULL_RES Bit

When this bit is set to a value of 1, the device is in full resolution mode, where the output resolution increases with the *g* range set by the range bits to maintain a 4 mg/LSB scale factor. When the FULL_RES bit is set to 0, the device is in 10-bit mode, and the range bits determine the maximum *g* range and scale factor.

Justify Bit

A setting of 1 in the justify bit selects left-justified (MSB) mode, and a setting of 0 selects right-justified mode with sign extension.

Range Bits

These bits set the *g* range as described in Table 21.

Table 21. *g* Range Setting

Setting		
D1	D0	*g* Range
0	0	±2 *g*
0	1	±4 *g*
1	0	±8 *g*
1	1	±16 *g*

Register 0x32 to Register 0x37—DATAX0, DATAX1, DATAY0, DATAY1, DATAZ0, DATAZ1 (Read Only)

These six bytes (Register 0x32 to Register 0x37) are eight bits each and hold the output data for each axis. Register 0x32 and Register 0x33 hold the output data for the x-axis, Register 0x34 and Register 0x35 hold the output data for the y-axis, and Register 0x36 and Register 0x37 hold the output data for the z-axis. The output data is twos complement, with DATAx0 as the least significant byte and DATAx1 as the most significant byte, where x represent X, Y, or Z. The DATA_FORMAT register (Address 0x31) controls the format of the data. It is recommended that a multiple-byte read of all registers be performed to prevent a change in data between reads of sequential registers.

Register 0x38—FIFO_CTL (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
FIFO_MODE		Trigger	Samples				

FIFO_MODE Bits

These bits set the FIFO mode, as described in Table 22.

Table 22. FIFO Modes

Setting			
D7	D6	Mode	Function
0	0	Bypass	FIFO is bypassed.
0	1	FIFO	FIFO collects up to 32 values and then stops collecting data, collecting new data only when FIFO is not full.
1	0	Stream	FIFO holds the last 32 data values. When FIFO is full, the oldest data is overwritten with newer data.
1	1	Trigger	When triggered by the trigger bit, FIFO holds the last data samples before the trigger event and then continues to collect data until full. New data is collected only when FIFO is not full.

Trigger Bit

A value of 0 in the trigger bit links the trigger event of trigger mode to INT1, and a value of 1 links the trigger event to INT2.

Samples Bits

The function of these bits depends on the FIFO mode selected (see Table 23). Entering a value of 0 in the samples bits immediately sets the watermark status bit in the INT_SOURCE register, regardless of which FIFO mode is selected. Undesirable operation may occur if a value of 0 is used for the samples bits when trigger mode is used.

Table 23. Samples Bits Functions

FIFO Mode	Samples Bits Function
Bypass	None.
FIFO	Specifies how many FIFO entries are needed to trigger a watermark interrupt.
Stream	Specifies how many FIFO entries are needed to trigger a watermark interrupt.
Trigger	Specifies how many FIFO samples are retained in the FIFO buffer before a trigger event.

0x39—FIFO_STATUS (Read Only)

D7	D6	D5	D4	D3	D2	D1	D0
FIFO_TRIG	0	Entries					

FIFO_TRIG Bit

A 1 in the FIFO_TRIG bit corresponds to a trigger event occurring, and a 0 means that a FIFO trigger event has not occurred.

Entries Bits

These bits report how many data values are stored in FIFO. Access to collect the data from FIFO is provided through the DATAX, DATAY, and DATAZ registers. FIFO reads must be done in burst or multiple-byte mode because each FIFO level is cleared after any read (single- or multiple-byte) of FIFO. FIFO stores a maximum of 32 entries, which equates to a maximum of 33 entries available at any given time because an additional entry is available at the output filter of the device.

APPLICATIONS INFORMATION

POWER SUPPLY DECOUPLING

A 1 µF tantalum capacitor (C_S) at V_S and a 0.1 µF ceramic capacitor ($C_{I/O}$) at $V_{DD\ I/O}$ placed close to the ADXL345 supply pins is recommended to adequately decouple the accelerometer from noise on the power supply. If additional decoupling is necessary, a resistor or ferrite bead, no larger than 100 Ω, in series with V_S may be helpful. Additionally, increasing the bypass capacitance on V_S to a 10 µF tantalum capacitor in parallel with a 0.1 µF ceramic capacitor may also improve noise.

Care should be taken to ensure that the connection from the ADXL345 ground to the power supply ground has low impedance because noise transmitted through ground has an effect similar to noise transmitted through V_S. It is recommended that V_S and $V_{DD\ I/O}$ be separate supplies to minimize digital clocking noise on the V_S supply. If this is not possible, additional filtering of the supplies, as previously mentioned, may be necessary.

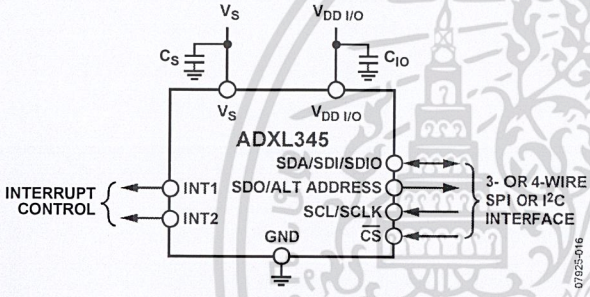


Figure 43. Application Diagram

MECHANICAL CONSIDERATIONS FOR MOUNTING

The ADXL345 should be mounted on the PCB in a location close to a hard mounting point of the PCB to the case. Mounting the ADXL345 at an unsupported PCB location, as shown in Figure 44, may result in large, apparent measurement errors due to undampened PCB vibration. Locating the accelerometer near a hard mounting point ensures that any PCB vibration at the accelerometer is above the accelerometer's mechanical sensor resonant frequency and, therefore, effectively invisible to the accelerometer. Multiple mounting points, close to the sensor, and/or a thicker PCB also help to reduce the effect of system resonance on the performance of the sensor.

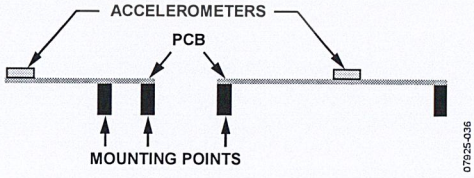


Figure 44. Incorrectly Placed Accelerometers

TAP DETECTION

The tap interrupt function is capable of detecting either single or double taps. The following parameters are shown in Figure 45 for a valid single and valid double tap event:

- The tap detection threshold is defined by the THRESH_TAP register (Address 0x1D).
- The maximum tap duration time is defined by the DUR register (Address 0x21).
- The tap latency time is defined by the latent register (Address 0x22) and is the waiting period from the end of the first tap until the start of the time window, when a second tap can be detected, which is determined by the value in the window register (Address 0x23).
- The interval after the latency time (set by the latent register) is defined by the window register. Although a second tap must begin after the latency time has expired, it need not finish before the end of the time defined by the window register.

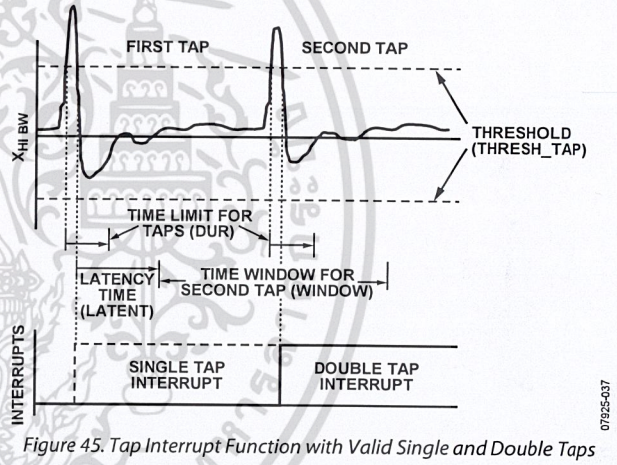


Figure 45. Tap Interrupt Function with Valid Single and Double Taps

If only the single tap function is in use, the single tap interrupt is triggered when the acceleration goes below the threshold, as long as DUR has not been exceeded. If both single and double tap functions are in use, the single tap interrupt is triggered when the double tap event has been either validated or invalidated.

Several events can occur to invalidate the second tap of a double tap event. First, if the suppress bit in the TAP_AXES register (Address 0x2A) is set, any acceleration spike above the threshold during the latency time (set by the latent register) invalidates the double tap detection, as shown in Figure 46.

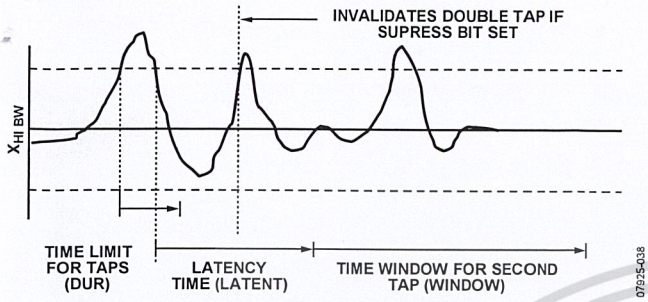


Figure 46. Double Tap Event Invalid Due to High g Event When the Suppress Bit Is Set

A double tap event can also be invalidated if acceleration above the threshold is detected at the start of the time window for the second tap (set by the window register). This results in an invalid double tap at the start of this window, as shown in Figure 47. Additionally, a double tap event can be invalidated if an acceleration exceeds the time limit for taps (set by the DUR register), resulting in an invalid double tap at the end of the DUR time limit for the second tap event, also shown in Figure 47.

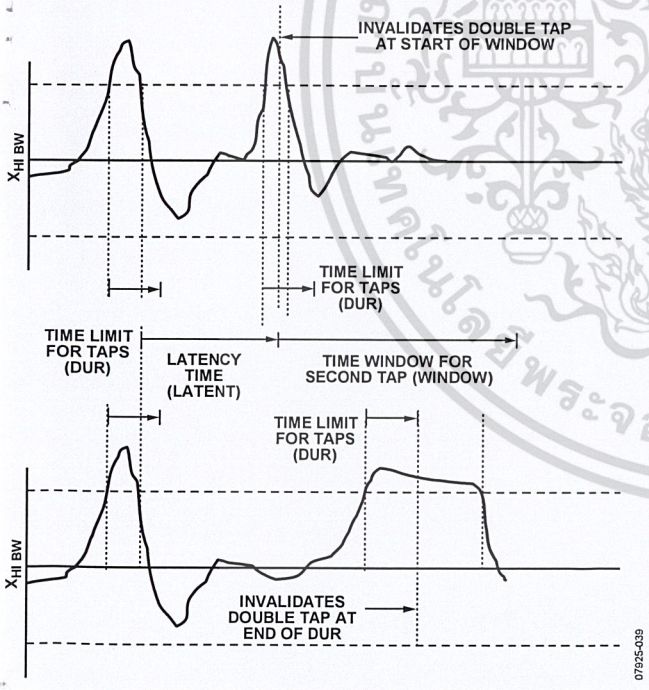


Figure 47. Tap Interrupt Function with Invalid Double Taps

Single taps, double taps, or both can be detected by setting the respective bits in the INT_ENABLE register (Address 0x2E). Control over participation of each of the three axes in single tap/double tap detection is exerted by setting the appropriate bits in the TAP_AXES register (Address 0x2A). For the double tap function to operate, both the latent and window registers must be set to a nonzero value.

Every mechanical system has somewhat different single tap/double tap responses based on the mechanical characteristics of the system. Therefore, some experimentation with values for the DUR, latent, window, and THRESH_TAP registers is required. In general, a good starting point is to set the DUR register to a value greater than 0x10 (10 ms), the latent register to a value greater than 0x10 (20 ms), the window register to a value greater than 0x40 (80 ms), and the THRESH_TAP register to a value greater than 0x30 (3 *g*). Setting a very low value in the latent, window, or THRESH_TAP register may result in an unpredictable response due to the accelerometer picking up echoes of the tap inputs.

After a tap interrupt has been received, the first axis to exceed the THRESH_TAP level is reported in the ACT_TAP_STATUS register (Address 0x2B). This register is never cleared but is overwritten with new data.

THRESHOLD

The lower output data rates are achieved by decimating a common sampling frequency inside the device. The activity, free-fall, and single tap/double tap detection functions without improved tap enabled are performed using undecimated data. Because the bandwidth of the output data varies with the data rate and is lower than the bandwidth of the undecimated data, the high frequency and high *g* data that is used to determine activity, free-fall, and single tap/double tap events may not be present if the output of the accelerometer is examined. This may result in functions triggering when acceleration data does not appear to meet the conditions set by the user for the corresponding function.

LINK MODE

The function of the link bit is to reduce the number of activity interrupts that the processor must service by setting the device to look for activity only after inactivity. For proper operation of this feature, the processor must still respond to the activity and inactivity interrupts by reading the INT_SOURCE register (Address 0x30) and, therefore, clearing the interrupts. If an activity interrupt is not cleared, the part cannot go into autosleep mode. The asleep bit in the ACT_TAP_STATUS register (Address 0x2B) indicates if the part is asleep.

SLEEP MODE VS. LOW POWER MODE

In applications where a low data rate and low power consumption is desired (at the expense of noise performance), it is recommended that low power mode be used. The use of low power mode preserves the functionality of the DATA_READY interrupt and the FIFO for postprocessing of the acceleration data. Sleep mode, while offering a low data rate and power consumption, is not intended for data acquisition.

However, when sleep mode is used in conjunction with the AUTO_SLEEP mode and the link mode, the part can automatically switch to a low power, low sampling rate mode when inactivity is detected. To prevent the generation of redundant inactivity interrupts, the inactivity interrupt is automatically disabled and activity is enabled. When the ADXL345 is in sleep mode, the host processor can also be placed into sleep mode or low power mode to save significant system power. When activity is detected, the accelerometer automatically switches back to the original data rate of the application and provides an activity interrupt that can be used to wake up the host processor. Similar to when inactivity occurs, detection of activity events is disabled and inactivity is enabled.

OFFSET CALIBRATION

Accelerometers are mechanical structures containing elements that are free to move. These moving parts can be very sensitive to mechanical stresses, much more so than solid-state electronics. The 0 *g* bias or offset is an important accelerometer metric because it defines the baseline for measuring acceleration. Additional stresses can be applied during assembly of a system containing an accelerometer. These stresses can come from, but are not limited to, component soldering, board stress during mounting, and application of any compounds on or over the component. If calibration is deemed necessary, it is recommended that calibration be performed after system assembly to compensate for these effects.

A simple method of calibration is to measure the offset while assuming that the sensitivity of the ADXL345 is as specified in Table 1. The offset can then be automatically accounted for by using the built-in offset registers. This results in the data acquired from the DATA registers already compensating for any offset.

In a no-turn or single-point calibration scheme, the part is oriented such that one axis, typically the z-axis, is in the 1 *g* field of gravity and the remaining axes, typically the x- and y-axis, are in a 0 *g* field. The output is then measured by taking the average of a series of samples. The number of samples averaged is a choice of the system designer, but a recommended starting point is 0.1 sec worth of data for data rates of 100 Hz or greater. This corresponds to 10 samples at the 100 Hz data rate. For data rates less than 100 Hz, it is recommended that at least 10 samples be averaged together. These values are stored as X_{0g}, Y_{0g}, and Z_{+1g} for the 0 *g* measurements on the x- and y-axis and the 1 *g* measurement on the z-axis, respectively.

The values measured for X_{0g} and Y_{0g} correspond to the x- and y-axis offset, and compensation is done by subtracting those values from the output of the accelerometer to obtain the actual acceleration:

$$X_{ACTUAL} = X_{MEAS} - X_{0g}$$

$$Y_{ACTUAL} = Y_{MEAS} - Y_{0g}$$

Because the z-axis measurement was done in a +1 *g* field, a no-turn or single-point calibration scheme assumes an ideal sensitivity, S_Z for the z-axis. This is subtracted from Z_{+1g} to attain the z-axis offset, which is then subtracted from future measured values to obtain the actual value:

$$Z_{0g} = Z_{+1g} - S_Z$$

$$Z_{ACTUAL} = Z_{MEAS} - Z_{0g}$$

The ADXL345 can automatically compensate the output for offset by using the offset registers (Register 0x1E, Register 0x1F, and Register 0x20). These registers contain an 8-bit, twos complement value that is automatically added to all measured acceleration values, and the result is then placed into the DATA registers. Because the value placed in an offset register is additive, a negative value is placed into the register to eliminate a positive offset and vice versa for a negative offset. The register has a scale factor of 15.6 mg/LSB and is independent of the selected *g*-range.

As an example, assume that the ADXL345 is placed into full-resolution mode with a sensitivity of typically 256 LSB/*g*. The part is oriented such that the z-axis is in the field of gravity and x-, y-, and z-axis outputs are measured as +10 LSB, −13 LSB, and +9 LSB, respectively. Using the previous equations, X_{0g} is +10 LSB, Y_{0g} is −13 LSB, and Z_{0g} is +9 LSB. Each LSB of output in full-resolution is 3.9 mg or one-quarter of an LSB of the offset register. Because the offset register is additive, the 0 *g* values are negated and rounded to the nearest LSB of the offset register:

$$X_{OFFSET} = -Round(10/4) = -3 \text{ LSB}$$

$$Y_{OFFSET} = -Round(-13/4) = 3 \text{ LSB}$$

$$Z_{OFFSET} = -Round(9/4) = -2 \text{ LSB}$$

These values are programmed into the OFSX, OFSY, and OFXZ registers, respectively, as 0xFD, 0x03 and 0xFE. As with all registers in the ADXL345, the offset registers do not retain the value written into them when power is removed from the part. Power-cycling the ADXL345 returns the offset registers to their default value of 0x00.

Because the no-turn or single-point calibration method assumes an ideal sensitivity in the z-axis, any error in the sensitivity results in offset error. For instance, if the actual sensitivity was 250 LSB/*g* in the previous example, the offset would be 15 LSB, not 9 LSB. To help minimize this error, an additional measurement point can be used with the z-axis in a 0 *g* field and the 0 *g* measurement can be used in the Z_{ACTUAL} equation.

USING SELF-TEST

The self-test change is defined as the difference between the acceleration output of an axis with self-test enabled and the acceleration output of the same axis with self-test disabled (see Endnote 4 of Table 1). This definition assumes that the sensor does not move between these two measurements, because if the sensor moves, a non–self-test related shift corrupts the test.

Proper configuration of the ADXL345 is also necessary for an accurate self-test measurement. The part should be set with a data rate greater than or equal to 100 Hz. This is done by ensuring that a value greater than or equal to 0x0A is written into the rate bits (Bit D3 through Bit D0) in the BW_RATE register (Address 0x2C). The part also must be placed into normal power operation by ensuring the LOW_POWER bit in the BW_RATE register is cleared (LOW_POWER bit = 0) for accurate self-test measurements. It is recommended that the part be set to full-resolution, 16 *g* mode to ensure that there is sufficient dynamic range for the entire self-test shift. This is done by setting Bit D3 of the DATA_FORMAT register (Address 0x31) and writing a value of 0x03 to the range bits (Bit D1 and Bit D0) of the DATA_FORMAT register (Address 0x31). This results in a high dynamic range for measurement and a 3.9 mg/LSB scale factor.

After the part is configured for accurate self-test measurement, several samples of x-, y-, and z-axis acceleration data should be retrieved from the sensor and averaged together. The number of samples averaged is a choice of the system designer, but a recommended starting point is 0.1 sec worth of data for data rates of 100 Hz or greater. This corresponds to 10 samples at the 100 Hz data rate. For data rates less than 100 Hz, it is recommended that at least 10 samples be averaged together. The averaged values should be stored and labeled appropriately as the self-test disabled data, that is, X_{ST_OFF}, Y_{ST_OFF}, and Z_{ST_OFF}.

Next, self-test should be enabled by setting Bit D7 (SELF_TEST) of the DATA_FORMAT register (Address 0x31). The output needs some time (about four samples) to settle after enabling self-test. After allowing the output to settle, several samples of the x-, y-, and z-axis acceleration data should be taken again and averaged. It is recommended that the same number of samples be taken for this average as was previously taken. These averaged values should again be stored and labeled appropriately as the value with self-test enabled, that is, X_{ST_ON}, Y_{ST_ON}, and Z_{ST_ON}. Self-test can then be disabled by clearing Bit D7 (SELF_TEST) of the DATA_FORMAT register (Address 0x31).

With the stored values for self-test enabled and disabled, the self-test change is as follows:

$$X_{ST} = X_{ST\_ON} - X_{ST\_OFF}$$

$$Y_{ST} = Y_{ST\_ON} - Y_{ST\_OFF}$$

$$Z_{ST} = Z_{ST\_ON} - Z_{ST\_OFF}$$

Because the measured output for each axis is expressed in LSBs, X_{ST}, Y_{ST}, and Z_{ST} are also expressed in LSBs. These values can be converted to *g*'s of acceleration by multiplying each value by the 3.9 mg/LSB scale factor, if configured for full-resolution mode. Additionally, Table 15 through Table 18 correspond to the self-test range converted to LSBs and can be compared with the measured self-test change when operating at a V_S of 2.5 V. For other voltages, the minimum and maximum self-test output values should be adjusted based on (multiplied by) the scale factors shown in Table 14. If the part was placed into ±2 *g*, 10-bit or full-resolution mode, the values listed in Table 15 should be used. Although the fixed 10-bit mode or a range other than 16 *g* can be used, a different set of values, as indicated in Table 16 through Table 18, would need to be used. Using a range below 8 *g* may result in insufficient dynamic range and should be considered when selecting the range of operation for measuring self-test.

If the self-test change is within the valid range, the test is considered successful. Generally, a part is considered to pass if the minimum magnitude of change is achieved. However, a part that changes by more than the maximum magnitude is not necessarily a failure.

DATA FORMATTING OF UPPER DATA RATES

Formatting of output data at the 3200 Hz and 1600 Hz output data rates changes depending on the mode of operation (full-resolution or fixed 10-bit) and the selected output range.

When using the 3200 Hz or 1600 Hz output data rates in full-resolution or ±2 g, 10-bit operation, the LSB of the output data-word is always 0. When data is right justified, this corresponds to Bit D0 of the DATAx0 register, as shown in Figure 48. When data is left justified and the part is operating in ±2 g, 10-bit mode, the LSB of the output data-word is Bit D6 of the DATAx0 register. In full-resolution operation when data is left justified, the location of the LSB changes according to the selected output range.

For a range of ±2 g, the LSB is Bit D6 of the DATAx0 register; for ±4 g, Bit D5 of the DATAx0 register; for ±8 g, Bit D4 of the DATAx0 register; and for ±16 g, Bit D3 of the DATAx0 register. This is shown in Figure 49.

The use of 3200 Hz and 1600 Hz output data rates for fixed 10-bit operation in the ±4 g, ±8 g, and ±16 g output ranges provides an LSB that is valid and that changes according to the applied acceleration. Therefore, in these modes of operation, Bit D0 is not always 0 when output data is right justified and Bit D6 is not always 0 when output data is left justified. Operation at any data rate of 800 Hz or lower also provides a valid LSB in all ranges and modes that changes according to the applied acceleration.

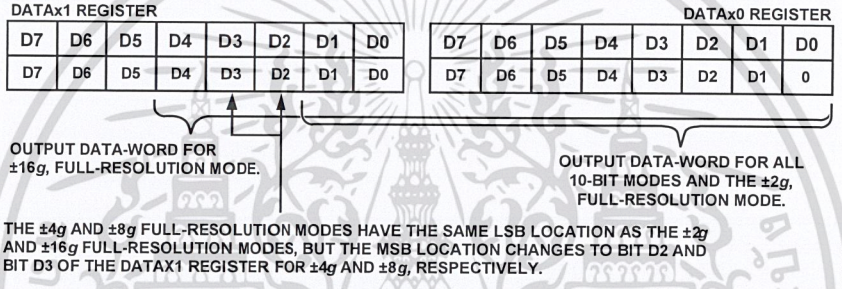


Figure 48. Data Formatting of Full-Resolution and ±2 g, 10-Bit Modes of Operation When Output Data Is Right Justified

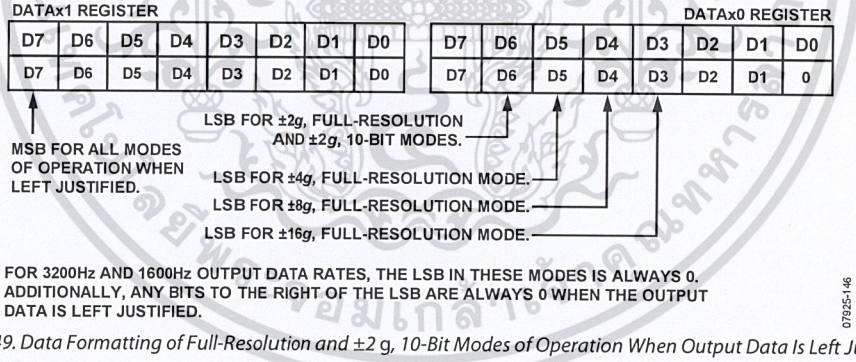


Figure 49. Data Formatting of Full-Resolution and ±2 g, 10-Bit Modes of Operation When Output Data Is Left Justified

NOISE PERFORMANCE

The specification of noise shown in Table 1 corresponds to the typical noise performance of the ADXL345 in normal power operation with an output data rate of 100 Hz (LOW_POWER bit (D4) = 0, rate bits (D3:D0) = 0xA in the BW_RATE register, Address 0x2C). For normal power operation at data rates below 100 Hz, the noise of the ADXL345 is equivalent to the noise at 100 Hz ODR in LSBs. For data rates greater than 100 Hz, the noise increases roughly by a factor of √2 per doubling of the data rate. For example, at 400 Hz ODR, the noise on the x- and y-axes is typically less than 1.5 LSB rms, and the noise on the z-axis is typically less than 2.2 LSB rms.

For low power operation (LOW_POWER bit (D4) = 1 in the BW_RATE register, Address 0x2C), the noise of the ADXL345 is constant for all valid data rates shown in Table 8. This value is typically less than 1.8 LSB rms for the x- and y-axes and typically less than 2.6LSB rms for the z-axis.

The trend of noise performance for both normal power and low power modes of operation of the ADXL345 is shown in Figure 50.

Figure 51 shows the typical Allan deviation for the ADXL345. The 1/f corner of the device, as shown in this figure, is very low, allowing absolute resolution of approximately 100 µg (assuming that there is sufficient integration time). Figure 51 also shows that the noise density is 290 µg/√Hz for the x-axis and y-axis and 430 µg/√Hz for the z-axis.

Figure 52 shows the typical noise performance trend of the ADXL345 over supply voltage. The performance is normalized to the tested and specified supply voltage, V_S = 2.5 V. In general, noise decreases as supply voltage is increased. It should be noted, as shown in Figure 50, that the noise on the z-axis is typically higher than on the x-axis and y-axis; therefore, while they change roughly the same in percentage over supply voltage, the magnitude of change on the z-axis is greater than the magnitude of change on the x-axis and y-axis.

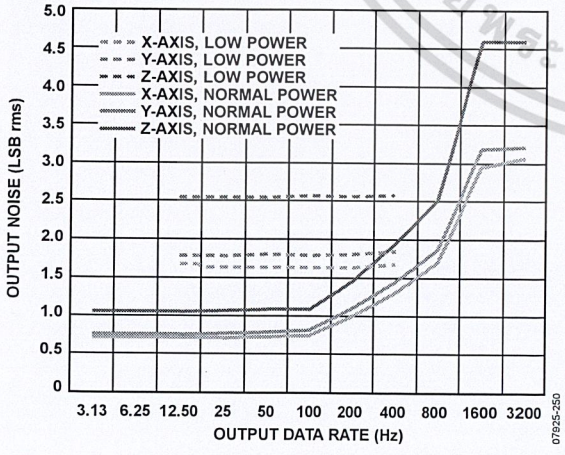


Figure 50. Noise vs. Output Data Rate for Normal and Low Power Modes, Full-Resolution (256 LSB/g)

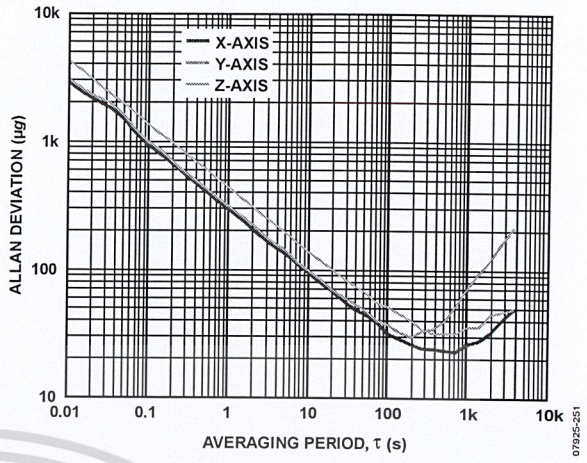


Figure 51. Root Allan Deviation

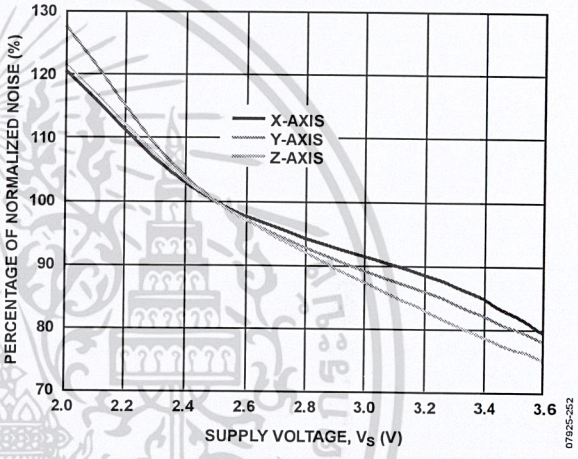


Figure 52. Normalized Noise vs. Supply Voltage, V_S

OPERATION AT VOLTAGES OTHER THAN 2.5 V

The ADXL345 is tested and specified at a supply voltage of V_S = 2.5 V; however, it can be powered with V_S as high as 3.6 V or as low as 2.0 V. Some performance parameters change as the supply voltage changes: offset, sensitivity, noise, self-test, and supply current.

Due to slight changes in the electrostatic forces as supply voltage is varied, the offset and sensitivity change slightly. When operating at a supply voltage of V_S = 3.3 V, the x- and y-axis offset is typically 25 mg higher than at V_S = 2.5 V operation. The z-axis is typically 20 mg lower when operating at a supply voltage of 3.3 V than when operating at V_S = 2.5 V. Sensitivity on the x- and y-axes typically shifts from a nominal 256 LSB/*g* (full-resolution or ±2 *g*, 10-bit operation) at V_S = 2.5 V operation to 265 LSB/*g* when operating with a supply voltage of 3.3 V. The z-axis sensitivity is unaffected by a change in supply voltage and is the same at V_S = 3.3 V operation as it is at V_S = 2.5 V operation. Simple linear interpolation can be used to determine typical shifts in offset and sensitivity at other supply voltages.

Changes in noise performance, self-test response, and supply current are discussed elsewhere throughout the data sheet. For noise performance, the Noise Performance section should be reviewed. The Using Self-Test section discusses both the operation of self-test over voltage, a square relationship with supply voltage, as well as the conversion of the self-test response in *g*'s to LSBs. Finally, Figure 33 shows the impact of supply voltage on typical current consumption at a 100 Hz output data rate, with all other output data rates following the same trend.

OFFSET PERFORMANCE AT LOWEST DATA RATES

The ADXL345 offers a large number of output data rates and bandwidths, designed for a large range of applications. However, at the lowest data rates, described as those data rates below 6.25 Hz, the offset performance over temperature can vary significantly from the remaining data rates. Figure 53, Figure 54, and Figure 55 show the typical offset performance of the ADXL345 over temperature for the data rates of 6.25 Hz and lower. All plots are normalized to the offset at 100 Hz output data rate; therefore, a nonzero value corresponds to additional offset shift due to temperature for that data rate.

When using the lowest data rates, it is recommended that the operating temperature range of the device be limited to provide minimal offset shift across the operating temperature range. Due to variability between parts, it is also recommended that calibration over temperature be performed if any data rates below 6.25 Hz are in use.

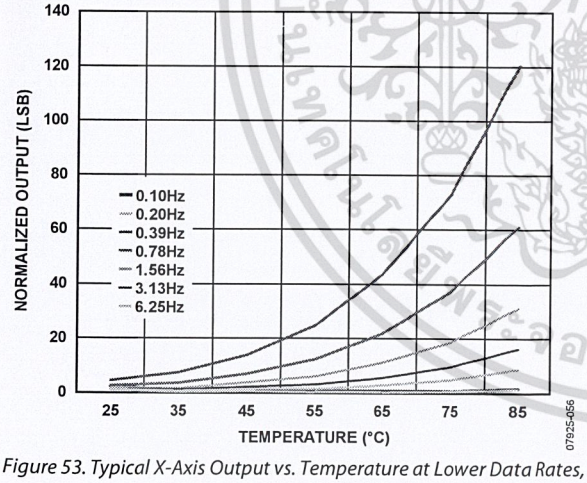


Figure 53. Typical X-Axis Output vs. Temperature at Lower Data Rates, Normalized to 100 Hz Output Data Rate, V_S = 2.5 V

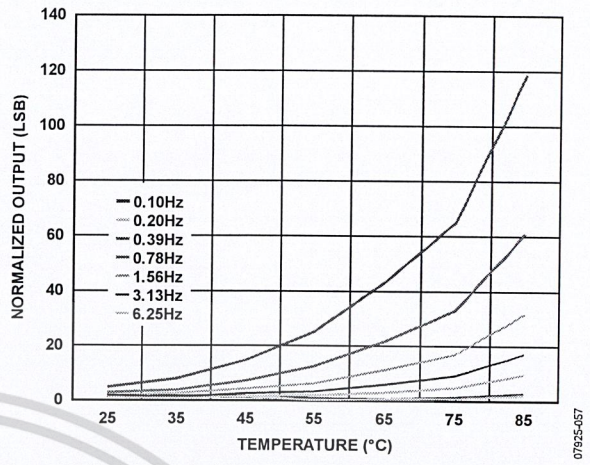


Figure 54. Typical Y-Axis Output vs. Temperature at Lower Data Rates, Normalized to 100 Hz Output Data Rate, V_S = 2.5 V

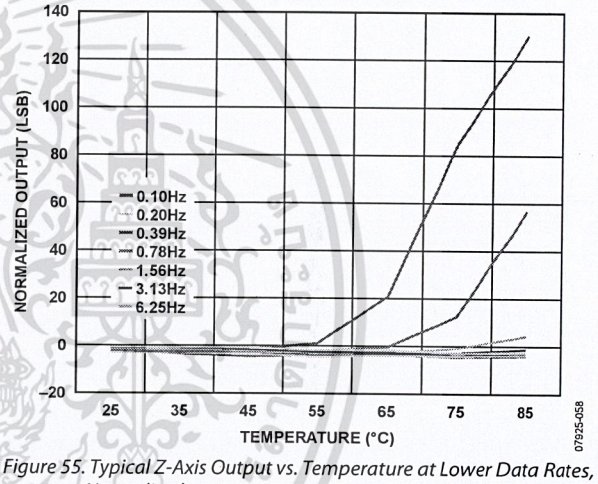


Figure 55. Typical Z-Axis Output vs. Temperature at Lower Data Rates, Normalized to 100 Hz Output Data Rate, V_S = 2.5 V

AXES OF ACCELERATION SENSITIVITY

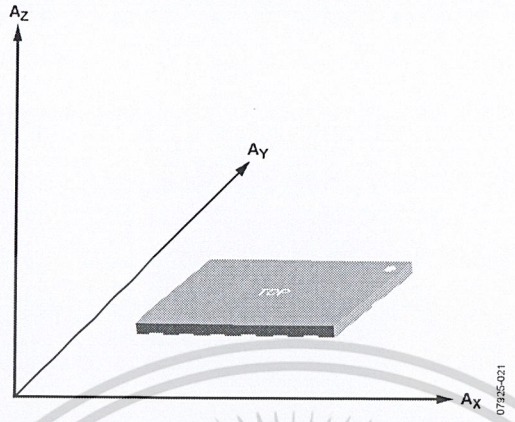


Figure 56. Axes of Acceleration Sensitivity (Corresponding Output Voltage Increases When Accelerated Along the Sensitive Axis)

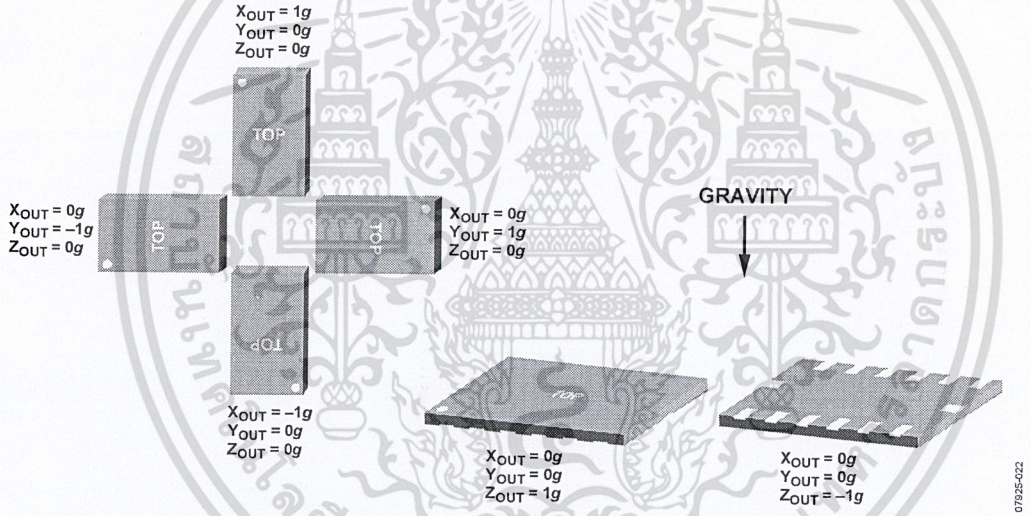


Figure 57. Output Response vs. Orientation to Gravity

LAYOUT AND DESIGN RECOMMENDATIONS

Figure 58 shows the recommended printed wiring board land pattern. Figure 59 and Table 24 provide details about the recommended soldering profile.

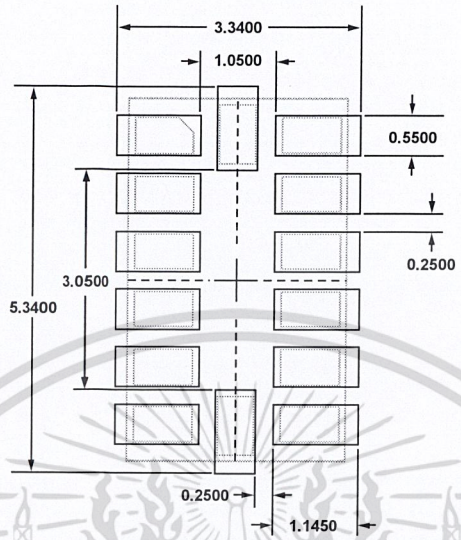


Figure 58. Recommended Printed Wiring Board Land Pattern (Dimensions shown in millimeters)

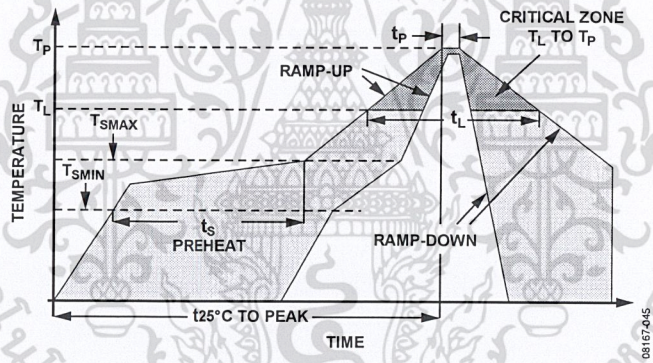


Figure 59. Recommended Soldering Profile

Table 24. Recommended Soldering Profile[1, 2]

Profile Feature	Condition	
	Sn63/Pb37	Pb-Free
Average Ramp Rate from Liquid Temperature (T_L) to Peak Temperature (T_P)	3°C/sec maximum	3°C/sec maximum
Preheat		
Minimum Temperature (T_{SMIN})	100°C	150°C
Maximum Temperature (T_{SMAX})	150°C	200°C
Time from T_{SMIN} to T_{SMAX} (t_S)	60 sec to 120 sec	60 sec to 180 sec
T_{SMAX} to T_L Ramp-Up Rate	3°C/sec maximum	3°C/sec maximum
Liquid Temperature (T_L)	183°C	217°C
Time Maintained Above T_L (t_L)	60 sec to 150 sec	60 sec to 150 sec
Peak Temperature (T_P)	240 + 0/−5°C	260 + 0/−5°C
Time of Actual T_P − 5°C (t_P)	10 sec to 30 sec	20 sec to 40 sec
Ramp-Down Rate	6°C/sec maximum	6°C/sec maximum
Time 25°C to Peak Temperature	6 minutes maximum	8 minutes maximum

[1] Based on JEDEC Standard J-STD-020D.1.
[2] For best results, the soldering profile should be in accordance with the recommendations of the manufacturer of the solder paste used.

OUTLINE DIMENSIONS

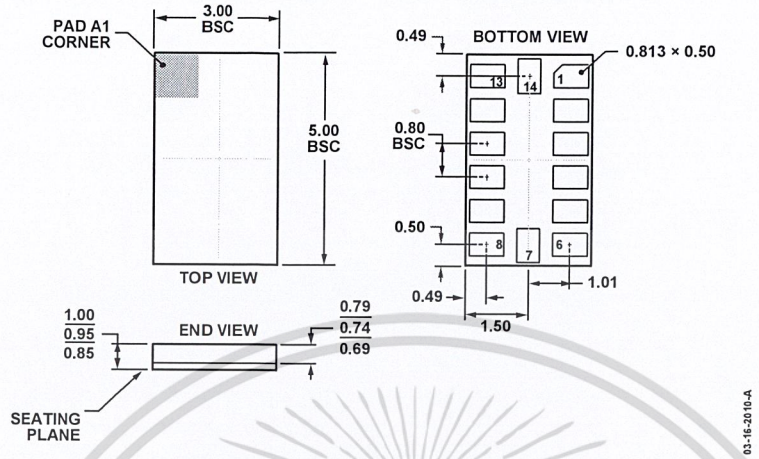


Figure 60. 14-Terminal Land Grid Array [LGA]
(CC-14-1)
Solder Terminations Finish Is Au over Ni
Dimensions shown in millimeters

ORDERING GUIDE

Model[1]	Measurement Range (*g*)	Specified Voltage (V)	Temperature Range	Package Description	Package Option
ADXL345BCCZ	±2, ±4, ±8, ±16	2.5	−40°C to +85°C	14-Terminal Land Grid Array [LGA]	CC-14-1
ADXL345BCCZ-RL	±2, ±4, ±8, ±16	2.5	−40°C to +85°C	14-Terminal Land Grid Array [LGA]	CC-14-1
ADXL345BCCZ-RL7	±2, ±4, ±8, ±16	2.5	−40°C to +85°C	14-Terminal Land Grid Array [LGA]	CC-14-1
EVAL-ADXL345Z				Evaluation Board	
EVAL-ADXL345Z-M				Analog Devices Inertial Sensor Evaluation System, Includes ADXL345 Satellite	
EVAL-ADXL345Z-S				ADXL345 Satellite, Standalone	

[1] Z = RoHS Compliant Part.

D07925-0-4/10(A)