เครื่องพิมพ์แบบสามมิติ : ส่วนชุดขับมอเตอร์

3D PRINTER : MOTOR DRIVER PART

ชนกานต์ หมื่นปา

ทินวัฒน์ วงศ์สถาพรพัฒน์

ธนวัฒน์ เงยวิจิตร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

เครื่องพิมพ์แบบสามมิติ : ส่วนชุดขับมอเตอร์

3D PRINTER : MOTOR DRIVER PART

ชนกานต์ หมื่นปา
ทินวัฒน์ วงศ์สถาพรพัฒน์
ธนวัฒน์ เงยวิจิตร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมระบบควบคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2556

3D PRINTER : MOTOR DRIVER PART

CHANAKARN MUENPA

TINNAWAT WONGSATARPORNPAT

TANAWAT NGOEIWIJIT

THIS THESIS IS SUBMITTED IN PATIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN CONTROL ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2013

ปริญญานิพนธ์ปีการศึกษา 2556
สาขาวิชาวิศวกรรมการวัดและควบคุม คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องพิมพ์แบบสามมิติ : ส่วนชุดขับมอเตอร์
3D PRINTER : MOTOR DRIVER PART

ผู้จัดทำ นายชนกานต์ หมื่นปา 53010288
นายทินวัฒน์ วงศ์สถาพรพัฒน์ 53010593
นายธนวัฒน์ เงยวิจิตร 53010652

.....อาจารย์ที่ปรึกษา
(ศาสตราจารย์ ดร.วันชัย ริ้วรุจา)

.....อาจารย์ที่ปรึกษา
(ผู้ช่วยศาสตราจารย์เทพจิตร์ เชยโภคา)

.....อาจารย์ที่ปรึกษา
(ผู้ช่วยศาสตราจารย์ ดร.วรรณดี เพชรมณีล้ำค่า)

เครื่องพิมพ์แบบสามมิติ : ส่วนชุดขับมอเตอร์

โดย
นายชนกานต์ หมื่นปา 53010288
นายทินวัฒน์ วงศ์สถาพรพัฒน์ 53010593
นายธนวัฒน์ เงยวิจิตร 53010652

อาจารย์ที่ปรึกษา
ศาสตราจารย์ ดร.วันชัย ริ้วรุจา
ผู้ช่วยศาสตราจารย์เทพจิตร์ เชยโภคา
ผู้ช่วยศาสตราจารย์ ดร.วรรณดี เพชรมณีล้ำค่า

ปีการศึกษา 2556

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอทฤษฎีและการออกแบบเครื่องพิมพ์สามมิติโดยโครงสร้างของเครื่องพิมพ์สามมิติประกอบด้วย ส่วนชุดโครงสร้าง ส่วนหัวฉีด ส่วนโปรแกรมและส่วนชุดขับมอเตอร์ จุดมุ่งหมายของโครงงานนี้คือสร้างเครื่องพิมพ์สามมิติ ที่สามารถพิมพ์พลาสติกออกมาเป็นตัวชิ้นงานได้สมบูรณ์ตามต้นแบบที่ป้อนเข้าไป โดยในปริญญานิพนธ์ฉบับนี้รับผิดชอบในส่วนของชุดขับมอเตอร์

ขั้นตอนดำเนินการเริ่มจากศึกษา ออกแบบ และประกอบชุดวงจรอิเล็กทรอนิกส์ที่จำเป็นทั้งหมดประกอบด้วย บอร์ดควบคุม และบอร์ดขับเคลื่อน เพื่อใช้ในการขับเคลื่อนสเต็ปมอเตอร์จำนวน 4 ตัว ซึ่งใช้ควบคุมการป้อนพลาสติกให้แก่หัวฉีด 1 ตัว และอีก 3 ตัวที่เหลือใช้ในการขับเคลื่อน 3 แกน แล้วจึงเขียนโปรแกรมคอมพิวเตอร์โดยใช้ภาษาซีพลัสพลัสเพื่อขับเคลื่อนมอเตอร์ให้ไปตามทิศทางที่ต้องการพิมพ์ตามคำสั่งจากต้นแบบแม่พิมพ์ที่ส่งมาจากส่วนโปรแกรม และขับมอเตอร์ให้ป้อนเส้นพลาสติกเข้าสู่หัวฉีดจากนั้นทำการทดลองการขับเคลื่อน 3 แกน และทดลองการขับเพื่อป้อนพลาสติกให้ได้อุณหภูมิที่เหมาะสมสำหรับการขึ้นรูป จากการทดลองพบว่าเครื่องพิมพ์สามารถขับเคลื่อน 3 แกน และสามารถพิมพ์ออกมาตามแบบได้ โดยมีค่าความผิดพลาดอยู่ในระดับที่ยอมรับได้

3D PRINTER : MOTOR DRIVER PART

By

Mr. Chanakarn Muenpa 53010288

Mr. Tinnawat Wongsatarpornpat 53010593

Mr. Tanawat Ngoeivijit 53011375

Advisors

Prof.Dr. Vanchai Riewruja

Asst.Prof.Thepjit Cheypoca

Asst.Prof.Dr. Wandee Petchmaneelumka

Academic Year 2013

ABSTRACT

This thesis presents theory and design of three-Dimensional Printer. The three-dimensional printer composes of hardware, software, injection and motor driver parts. The goal is to build the three-dimensional printer that can print plastic models from the designed digital model. Responsibility of this project is the motor driver part.

The procedures compose learning, design and assembling the necessary electronic circuits, including microcontroller board (MEGA 1280) and stepping motor driver board (A4988) in order to drive four stepping motors. One motor is used to feed filament into injection. Another three motors is used for movement the injection into x, y and z axes. Visual C++ language is used to program for driving the stepping motors in order to move the stepping motors to the commended direction and drive the stepping motor of injection. The movements of the three-axes stepping motors are experimented. Stepping motor for feed plastic into nozzle at appropriate temperature is tested. The experimental results show that the printer can move three axes and print the designed plastic model. The error is acceptable.

กิตติกรรมประกาศ

การจัดทำปริญญานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วงไปได้ด้วยดี เพราะได้รับความช่วยเหลือเป็นอย่างดีจาก ศาสตราจารย์ ดร.วันชัย ริ้วรุจา ผู้ช่วยศาสตราจารย์เทพจิตร์ เชยโภคา และผู้ช่วยศาสตราจารย์ ดร.วรรณดี เพชรมณีล้ำค่าที่ได้กรุณาให้คำปรึกษาและคำแนะนำที่ดีมาโดยตลอดตั้งแต่ต้น คอยติดตามความคืบหน้าของผลงาน รวมทั้งเอื้อเฟื้ออุปกรณ์ที่จำเป็นและความช่วยเหลืออื่นๆ ที่เป็นประโยชน์ต่อโครงงาน ผู้จัดทำรู้สึกซาบซึ้งและขอกราบขอบพระคุณอย่างสูง

ขอบคุณเพื่อนๆ ในกลุ่มงานทุกคนที่ให้ความร่วมมือในการทำงาน ช่วยเหลือซึ่งกันและกัน สนับสนุนอุปกรณ์ที่ขาดเหลือ กระตุ้นเตือน รวมทั้งคอยถามไถ่ความคืบหน้าของโครงงานอยู่เสมอ

สุดท้ายนี้ผู้จัดทำขอกราบขอบพระคุณบิดา มารดา และครอบครัวที่คอยเป็นกำลังใจที่ดีตลอดมา รวมถึงการสนับสนุนในเรื่องของงบประมาณที่ขาดเหลือ ตลอดจนเป็นแรงบันดาลใจที่ดีที่สุดที่ทำให้โครงงานนี้สำเร็จสมบูรณ์ลงได้

ผู้จัดทำ
นายชนกานต์ หมื่นปา
นายทินวัฒน์ วงศ์สถาพรพัฒน์
นายธนวัฒน์ เงยวิจิตร

สารบัญ

สารบัญ(ต่อ)

สารบัญรูป

สารบัญรูป(ต่อ)

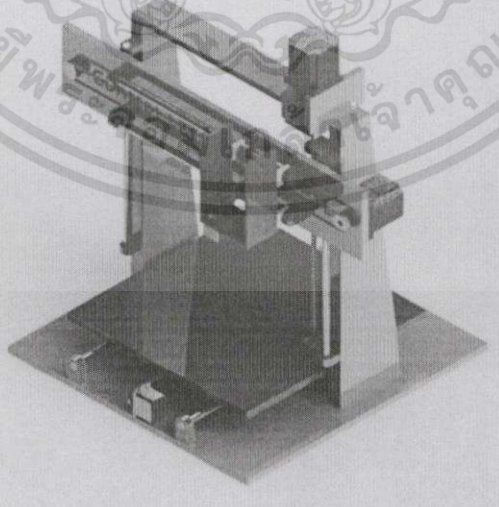
สารบัญตาราง

บทที่ 1
บทนำ

1.1 กล่าวนำ

เครื่องพิมพ์สามมิติ เป็นการพิมพ์ด้วยกระบวนการพิมพ์สามแกน ให้มีรูปร่างเหมือนจริงจากรูปแบบดิจิตอล ในปัจจุบันเครื่องพิมพ์สามมิติเข้ามามีบทบาททั้งในด้านการผลิต และการออกแบบทางสถาปัตยกรรม งานทางด้านวิศวกรรม รวมไปถึงในด้านการแพทย์ โดยที่การทำงานของเครื่องมี 4 ส่วนหลักด้วยกัน คือส่วนของหัวฉีดพลาสติก ส่วนของโปรแกรม ส่วนขับเคลื่อนมอเตอร์และส่วนโครงสร้างเครื่องการพิมพ์ชิ้นงานสามมิติเป็นกระบวนการพิมพ์ให้มีรูปร่างเหมือนของจริงจากรูปแบบดิจิตอล การพิมพ์ชิ้นงานสามมิติคือความสำเร็จในกระบวนการเติมแต่งวัสดุ ที่วางในแต่ละชั้นที่มีรูปร่างที่แตกต่างกันเครื่องพิมพ์แบบสามมิติเครื่องแรกถูกสร้างขึ้นในปี 1984 โดย 3D Systems Corp โดย Chuck Hull เป็นนักประดิษฐ์เครื่องพิมพ์แบบสามมิติที่ทันสมัยและเป็นผู้ริเริ่มเทคโนโลยีมาตรฐาน De Factor

หลักการทั่วไป การสร้างงานสามมิติจะใช้คอมพิวเตอร์ในการออกแบบ หรือซอฟต์แวร์ในการสร้างแบบจำลองการเคลื่อนไหวของเครื่องดิจิตอลตัดตามขวาง ความต่อเนื่องสำหรับการพิมพ์ขึ้นอยู่กับเครื่องที่ใช้ จะสร้างวัสดุเป็นชั้นๆ รวมกันวางบนแพลตฟอร์มที่เป็นชั้นจนเสร็จสมบูรณ์สุดท้ายจะได้เป็นรูปแบบสามมิติที่ได้รับการพิมพ์เป็นกระบวนการแบบ WYSIWYG ที่เป็นรูปแบบเสมือนและรูปแบบจำลองทางกายภาพเกือบจะเหมือนกัน การติดต่อระหว่างซอฟต์แวร์ CAD และเครื่องพิมพ์จะเป็นรูปแบบไฟล์ STL เป็นรูปแบบไฟล์ที่สร้างขึ้นเข้าเครื่องพิมพ์และไฟล์ VRML หรือ WRL จะใช้สำหรับเทคโนโลยีการพิมพ์ 3D ที่สามารถพิมพ์สีได้เต็มรูปแบบ



รูปที่ 1.1 เครื่องพิมพ์แบบสามมิติ

1.2 วัตถุประสงค์ในการทำปริญญานิพนธ์

1. ทำการศึกษาและทดลองการใช้งานอุปกรณ์ต่างๆ ในระบบ

2. ศึกษาการทำงานของ Microcontroller รุ่น ET-EASY-MEGA 1280 และ Driver A4988

3. ทำการศึกษาและสร้างระบบควบคุมตำแหน่งการเคลื่อนที่ของสเต็ปปิ้งมอเตอร์ (Stepping Motor) ซึ่งประกอบด้วยวงจรขับเคลื่อน และอุปกรณ์ Microcontroller อุปกรณ์อิเล็กทรอนิกส์ รวมถึงการควบคุมและประมวลผล โดยโครงงานชิ้นนี้ใช้โปรแกรมอาดูโน่ (Arduio) และภาษาซีพลัสพลัส (C++) เพื่อสามารถสั่งงานและทำการควบคุมการทำงานของสเต็ปปิ้งมอเตอร์ให้เป็นไปตามตำแหน่งที่ต้องการ

1.3 ขอบเขตของโครงงาน

1. ศึกษาและสร้างระบบควบคุมตำแหน่งการเคลื่อนที่สามแกนของหัวฉีดและการป้อนพลาสติก

2. ศึกษาและสร้างชุดคำสั่งควบคุมการขับมอเตอร์และควบคุมอุณหภูมิส่วนหัวฉีด

3. จัดทำวงจรชุดขับเคลื่อนมอเตอร์

4. ทดสอบระบบการใช้งาน ปรับปรุงและแก้ไข

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถนำความรู้ที่ได้รับจากการศึกษาทางทฤษฎีนำมาปฏิบัติได้จริง

2. ทราบถึงหลักการทำงานของชุดขับเคลื่อนมอเตอร์ Microcontroller และการเขียนโปรแกรมเพื่อควบคุมอุปกรณ์ดังกล่าว

3. สามารถนำไปพัฒนาต่อ เช่น การเพิ่มชุดหัวฉีดอีก 1 ชุดสำหรับการสร้างลายบนชิ้นงาน หรือการเปลี่ยนชนิดของพลาสติกที่ใช้ในการสร้างชิ้นงาน

1.5 รายละเอียดของปริญญานิพนธ์

เนื้อหาที่จะกล่าวในปริญญานิพนธ์ฉบับนี้ประกอบด้วย

บทที่ 1 บทนำกล่าวนำถึงวัตถุประสงค์ขอบเขตของโครงงาน ประโยชน์ที่คาดว่าจะได้รับ และรายละเอียดของปริญญานิพนธ์

บทที่ 2 ทฤษฎี หลักการ และความรู้ที่เกี่ยวข้องในการสร้างเครื่องพิมพ์สามมิติ

บทที่ 3 หลักการทำงาน นำเสนอการประกอบโครงสร้างของระบบควบคุมการขับมอเตอร์

บทที่ 4 การทดลอง เป็นส่วนการทดสอบการขับเคลื่อนหัวฉีดตามแนวแกนต่างๆ

บทที่ 5 บทวิจารณ์และสรุป สรุปผลการดำเนินงาน ปัญหา และแนวทางการปรับปรุงพัฒนาโครงงาน

บทที่ 2
ทฤษฎีและความรู้ที่เกี่ยวข้อง

การทำชิ้นงานพลาสติกหรือโลหะต้นแบบหรืองานเฉพาะที่มีจำนวนไม่มาก วิธีการและเครื่องมือที่นิยมใช้กันคือใช้เครื่อง CNC (Computer Numerical Control) กัดวัสดุทำเป็นแม่แบบขึ้นมา ซึ่งต้องใช้บุคลากรที่มีความชำนาญ รวมถึงใช้งบประมาณค่อนข้างสูง กว่าที่จะได้ต้นแบบมาใช้งาน เมื่อมีการพัฒนาเครื่องพิมพ์ 3 มิติขึ้นมา กระบวนการแบบเดิมๆ ก็เริ่มเปลี่ยนหากจะมองในเชิงเปรียบเทียบ เครื่องพิมพ์ 3 มิติก็คือเครื่อง CNC แต่มีการเพิ่มเติมและปรับเปลี่ยนอุปกรณ์ โดยมีการเปลี่ยนจากหัวกัดเป็นหัวฉีดพลาสติก (ในกรณีที่เป็นเครื่องพิมพ์ 3 มิติสำหรับงานพลาสติก) ด้านการทำงานจะต่างกัน โดยเครื่อง CNC ใช้หัวสว่านติดกับดอกสว่านแบบต่างๆ ในการแกะสลัก ตัด เจาะวัตถุดิบให้เป็นรูปร่างต่างๆ ตามต้องการ มีระบบมอเตอร์เคลื่อนที่ไปในทิศทาง 3 มิติ (แกน X, Y และ Z) ในขณะที่เครื่องพิมพ์ 3 มิติก็ใช้การเคลื่อนที่ของมอเตอร์เหมือนกับเครื่อง CNC แต่ที่ต่างกันคือ มีหัวฉีดที่จะฉีดหรือโรยพลาสติกเป็นชั้นๆ เรียงตัวขึ้นไป สร้างเป็นรูปทรงต่างๆ ในแต่ละชั้นมีความละเอียดในระดับมิลลิเมตร นอกจากนั้นยังมีการพัฒนาเทคนิคอื่นๆ ทำให้เครื่องพิมพ์ 3 มิติสร้างสรรค์งานได้จากหลากหลายวัสดุ และมีความละเอียดเพิ่มขึ้น ซึ่งก็ต้องแลกด้วยราคาที่แพงขึ้นด้วย

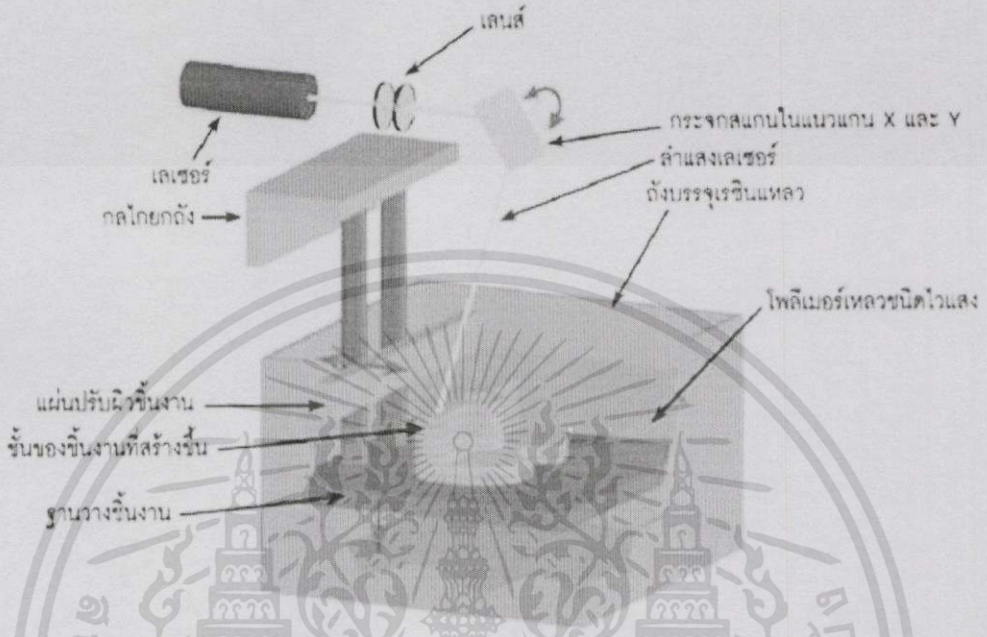
2.1 เทคนิคการสร้างชิ้นงานของเครื่องพิมพ์ 3 มิติ

ในปี ค.ศ. 2011 เริ่มมีการเปิดตัวจำหน่ายเครื่องพิมพ์ 3 มิติที่ใช้เทคนิคการสร้างต้นแบบอย่างรวดเร็ว (Rapid Prototype) ออกมา นับเป็นการสร้างปรากฏการณ์ใหม่แก่วงการสร้างโมเดลต้นแบบ จากนั้นมีการพัฒนาอย่างต่อเนื่อง มีเครื่องพิมพ์ 3 มิติออกมาจำหน่ายหลายรุ่น หลายราคา ส่วนเทคนิคการสร้างชิ้นงานของเครื่องพิมพ์ 3 มิติสรุปได้ 3 วิธีดังนี้

2.1.1 การใช้โพลีเมอร์ชนิดไวแสงในการขึ้นรูปชิ้นงาน (Photopolymerization)

เป็นเทคนิคที่ใช้แสงเพื่อขึ้นรูปจากวัสดุของเหลวให้กลายเป็นของแข็ง โดยใช้เทคนิคการขึ้นรูปชิ้นงานแบบ Stereo Lithography Apparatus (SLA) ที่คิดค้นโดย Chuck Hull จาก 3D Systems หลักการคือ ใช้วัสดุประเภทโพลีเมอร์เหลวที่สามารถแข็งตัวได้ เมื่อถูกกระตุ้นจากการฉายแสงอัลตร้าไวโอเล็ตหรือ UV (Ultra Violet) จาก UV เลเซอร์ โดยมีกระบะใส่เรซิ่นเหลวและมีระบบควบคุมให้ยกขึ้นและเคลื่อนที่ในแนวดิ่ง เพื่อสร้างชิ้นงานทีละชั้นจากล่างขึ้นบน แล้วสแกนยิงแสง UV เลเซอร์ในแนวราบ เพื่อเปลี่ยนให้โพลีเมอร์เหลวชนิดไวต่อแสงเป็นของแข็งตามตำแหน่งที่ต้องการ โดยชิ้นงานที่มีส่วนโค้งเว้า หรือมีรูปทรงที่แปลกๆ หรือมีความละเอียดซับซ้อนมาก อาจจะต้องสร้างส่วนที่ใช้ค้ำยันและรองรับเรียกว่า ซับพอร์ต (Support) ขึ้นมาพร้อมๆกับชิ้นงาน เมื่อขึ้นรูปชิ้นงานเสร็จก็ตัดส่วนซับพอร์ตนี้ออกไป ความแข็งแรงของวัสดุที่ขึ้นรูปด้วยเทคนิคนี้จะมีในระดับหนึ่งพอๆ กับพลาสติกทั่วไป ตัวชิ้นงานที่ได้มีความละเอียดเรียบในระดับหนึ่ง ตรงตามที่ได้ออกแบบไว้ในโปรแกรมวาดแบบ 3 มิติ

และยังสามารถสร้างชิ้นส่วนกลไกต่างๆ เพื่อทดสอบการทำงานของเครื่องต้นแบบได้ หรือใช้เป็นวัสดุชิ้นส่วนจริงในเครื่องมือต่างๆ ได้เลย



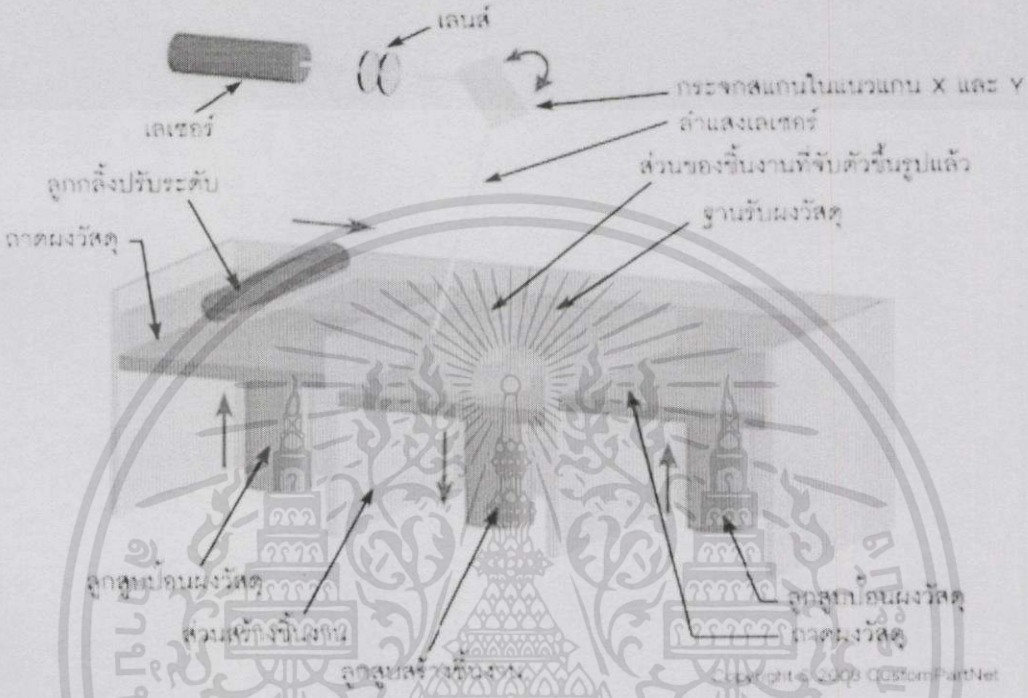
รูปที่ 2.1 กระบวนการขึ้นรูปด้วยเทคนิค Stereo Lithography Apparatus

จากเทคนิค SLA มีผู้พัฒนาต่อไปเป็นเทคนิค Micro Stereo Lithography เป็นเทคนิคที่เหมาะสำหรับการสร้างต้นแบบที่มีขนาดเล็กมาก ซึ่งเรียกว่า วิธีการขึ้นรูปวัสดุแบบ PolyJet เป็นการใช้หัวฉีดขนาดเล็กหลายๆ หัวฉีดเรซิ่นเหลวลงไปบนแท่นวางชิ้นงานพร้อมๆ กับการฉายแสง UV เพื่อทำให้เรซิ่นเหลวนี้แข็งตัวทันที ด้วยเทคนิคนี้ทำให้ไม่ต้องมีกระบะโพลีเมอร์เหลวขนาดใหญ่ ทำให้มีความสะดวกมากยิ่งขึ้น

2.1.2 การใช้เลเซอร์หรือกาวในการเชื่อมผงวัสดุให้เป็นชิ้นงาน (Granular Materials Binding)

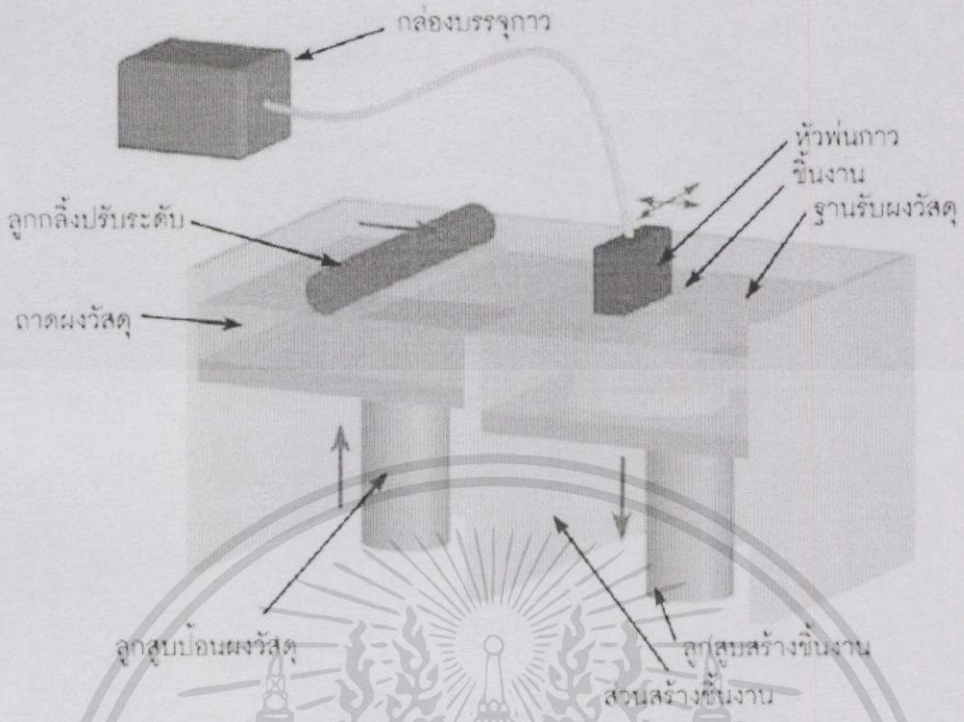
เป็นเทคนิคที่ใช้วัสดุที่เป็นผงเล็กๆ เมื่อได้รับความร้อนแล้วแข็งตัว จะเรียกเทคนิคนี้ว่า Selective Laser Sintering (SLS) โดยส่วนมากใช้วัสดุที่เป็นผงพลาสติก เนื่องจากมีจุดหลอมเหลวต่ำขึ้นรูปได้ง่าย ใช้ความร้อนจากเลเซอร์ที่ฉายไปยังตำแหน่งที่เป็นส่วนหน้าตัดขวางของชิ้นงาน ทำให้ผงพลาสติกเล็กๆ รอบบริเวณที่ถูกฉายแสงเลเซอร์นี้หลอมละลายและยึดเกาะกัน โดยยึดเกาะกับชั้นที่อยู่ก่อนหน้านี้ด้วย เพื่อให้เกิดเป็นโครงร่างรูปทรง 3 มิติขึ้นมาจากการเลื่อนแท่นใส่ผงพลาสติกลงเล็กน้อยในแนวดิ่ง จากนั้นจะกวาดผงพลาสติกมาคลุมทับส่วนที่ให้ความร้อนไปแล้ว ทำการฉายแสงเลเซอร์ใหม่ จะวนทำกระบวนการนี้ไปเรื่อยๆ จนครบทั้งรูปทรงที่ได้เขียนแบบไว้ จากนั้น ผงพลาสติก

ส่วนอื่นที่ไม่โดนความร้อนจากเลเซอร์จะถูกลมเป่าออกไปเหลือเพียงชิ้นงานที่หลอมละลายแล้วแข็งตัวเป็นรูปทรงที่สมบูรณ์เทคนิคนี้ใช้เวลามากในการขึ้นรูปชิ้นงาน ผิววัสดุจะหยาบและความแข็งแรงจะน้อยกว่าวัสดุที่ขึ้นรูปด้วยวิธี Stereo Lithography



รูปที่ 2.2 กระบวนการขึ้นรูปด้วยเทคนิค Selective Laser Sintering

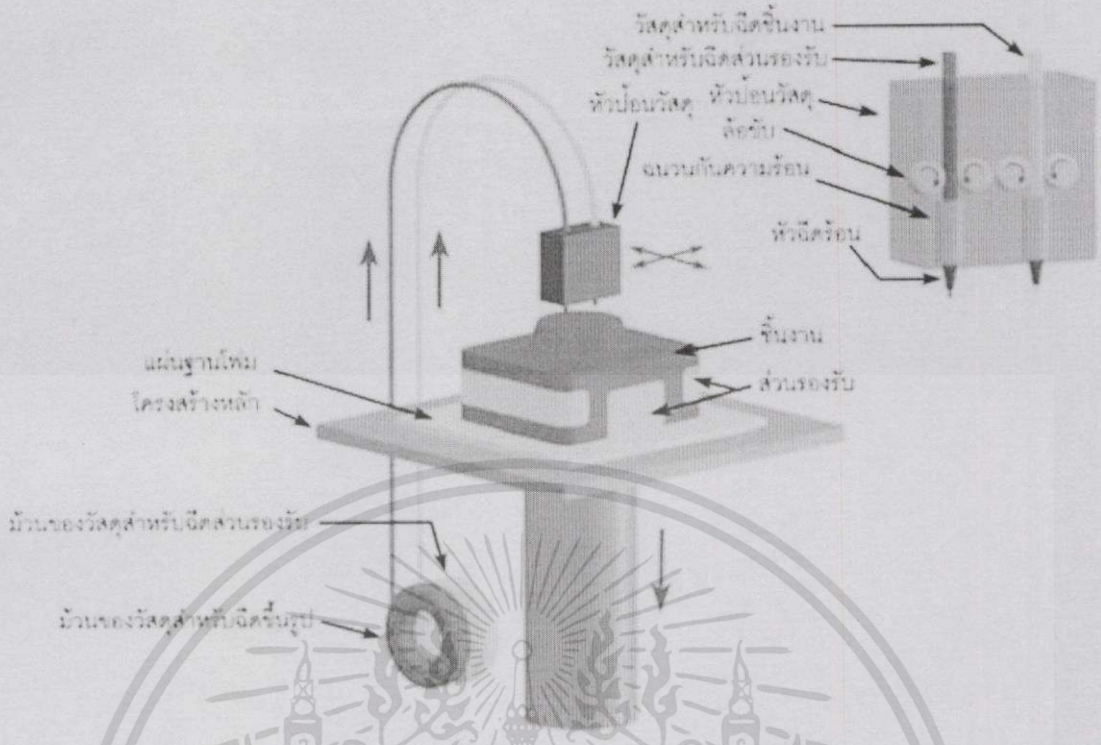
ยังมีอีกเทคนิคหนึ่งคือ การพ่นกาวลงไปยังผงพลาสติกให้ติดกันเป็นโครงสร้างตามที่ได้ออกแบบไว้ วิธีการแบบนี้เรียกว่า Three Dimensional Printing หรือ 3DP โดยหลักการทำงานของเทคนิคนี้คือ พ่นกาวด้วยหัวฉีดลงไปบนกระบะที่มีผงแป้งที่เกลี่ยหน้าให้เรียบไว้แล้ว ผงพลาสติกบริเวณที่โดนกาวก็จะติดแข็งตัว วิธีนี้สามารถเลือกให้ชิ้นงานมีสีต่างๆ แตกต่างกันได้ โดยผสมสีลงไปในกาวที่ใช้ ซึ่งก็เหมือนกับการพิมพ์หมึกลงไปบนกระดาษของเครื่องพิมพ์แบบอิงก์เจ็ตนั่นเอง ทำให้รูปทรง 3 มิติที่สร้างขึ้นมีความแข็งตัวและมีสีสันที่แตกต่างกันไปด้วย แต่วัสดุที่ขึ้นรูปด้วยวิธีนี้จะมีความเปราะบางมากกว่าวิธีการขึ้นรูปแบบชิ้นงานอื่น จึงเหมาะกับการทำชิ้นงานต้นแบบมากกว่านำไปใช้งานจริง



รูปที่ 2.3 กระบวนการขึ้นรูปชิ้นงานด้วยเทคนิค 3DP

2.1.3 การใช้พลาสติกร้อนเรียงตัวขึ้นเป็นชิ้นงาน (Extrusion Process)

เป็นวิธีที่นิยมและมีการพัฒนากันมากที่สุด เนื่องจากวัสดุที่ใช้ในการขึ้นรูปมีหลากหลาย ราคาไม่แพง มีการพัฒนาชุดควบคุมและซอฟต์แวร์ในแบบโอเพ่นซอร์ส (Open Source) เทคนิคนี้ใช้เส้นพลาสติกมาผ่านหัวที่ให้ความร้อนจนพลาสติกละลายเป็นของเหลวแล้วฉีดหรือโรยเรียงเป็นชั้นๆ เรียกเทคนิคการขึ้นรูปแบบนี้ว่า Fused Deposition Modeling หรือ FDM โดยมีการฉีดส่วนรองรับสำหรับรูปทรงที่โค้งงอ หรือมีความซับซ้อน เพื่อเพิ่มความแข็งแรงไม่ให้ล้มระหว่างการขึ้นรูปชิ้นงาน เมื่อขึ้นรูปเสร็จแล้วก็จะตัดออกได้ในภายหลัง ชิ้นงานที่ขึ้นรูปด้วยเทคนิคนี้จะมีผิวชิ้นงานที่ไม่เรียบ มีลักษณะเป็นชั้นๆ เนื่องจากขึ้นรูปด้วยการเรียงตัวเชื่อมติดกันของเส้นพลาสติกขนาดเล็กมากๆ ความแข็งแรงของชิ้นงานที่ขึ้นรูปด้วยวิธีนี้ขึ้นอยู่กับวัสดุของเส้นพลาสติกที่นำมาใช้ ซึ่งก็คือ ABS และ PLA โดย ABS มีความแข็งแรงมากกว่า PLA แต่ PLA ปลอดภัย ไม่ไวไฟ และไม่มีกลิ่นฉุนในขณะขึ้นรูปชิ้นงาน เนื่องจาก PLA ทำมาจากวัสดุธรรมชาติ ไม่เป็นพิษต่อสิ่งแวดล้อม



รูปที่ 2.4 กระบวนการขึ้นรูปด้วยเทคนิค FDM

2.2 โครงสร้างของเครื่องพิมพ์สามมิติ

เครื่องพิมพ์ 3 มิติที่นิยมผลิตและใช้งานมากที่สุดคือ เครื่องพิมพ์ที่ใช้เทคนิค FDM เนื่องจากกลไกของตัวเครื่องพิมพ์มีความซับซ้อนน้อย ใช้งบประมาณต่ำสุดเมื่อเทียบกับเครื่องพิมพ์ที่ใช้เทคนิคอื่น ในโครงงานนี้ทำการศึกษาและออกแบบส่วนประกอบและการทำงานของเครื่องพิมพ์แบบสามมิติแบบนี้เป็นหลัก เครื่องพิมพ์แบบสามมิติแบ่งโครงสร้างหลักออกเป็น 4 ส่วน ได้แก่ ส่วนโครงสร้าง ส่วนโปรแกรม ส่วนหัวฉีด และส่วนชุดขับมอเตอร์

2.2.1 ส่วนโครงสร้าง

โครงสร้างของเครื่องพิมพ์สามมิติที่ได้ทำการศึกษานี้ เลือกใช้แบบ Moving Bed เป็นเทคนิคที่เลื่อนฐานวางชิ้นงานในแนวแกน Y เท่านั้น ส่วนหัวฉีดจะติดกับลิเนียร์สไลด์หรือตัวเลื่อนความแม่นยำสูง เพื่อเลื่อนในแนวแกน X พร้อมกับฉีดเส้นพลาสติกลงบนฐานวางชิ้นงาน และเลื่อนขึ้นในแนวแกน Zทุกครั้งเมื่อขึ้นแบบชิ้นงานในแต่ละชั้นเสร็จแล้ว การเลื่อนในแนวแกน X และ Y จะใช้ลิเนียร์สไลด์ช่วยในการเคลื่อนที่ ทำให้การขึ้นรูปชิ้นงานมีความเร็วไม่มากนัก โครงสร้างของเครื่องที่ง่ายไม่ซับซ้อน

2.2.2 ส่วนหัวฉีด

หัวฉีดร้อน นอซเซิล (Nozzle) หรือบางครั้งเรียกว่า Hot End เป็นส่วนที่ให้ความร้อนกับเส้นพลาสติกที่ป้อนเข้ามาจากส่วนควบคุมการป้อนเส้นพลาสติกหรือ Extruder โดยอุณหภูมิที่หัวฉีดนี้จะอยู่ในช่วงที่ทำให้เส้นพลาสติกหลอม ซึ่งมีค่าประมาณ 190 ถึง 250 องศาเซลเซียส (ขึ้นอยู่กับชนิดของเส้นพลาสติก) มีตัวตรวจจับและวัดอุณหภูมิเป็นตัวตรวจสอบและส่งสัญญาณไปยังส่วนควบคุมอุณหภูมิ หัวฉีดร้อนนี้ทำมาจากโลหะ บรรจุในบล็อกอะลูมิเนียม และมีแผ่นระบายความร้อนติดอยู่ที่ตัวถังด้วย โดยรูของหัวฉีดมีขนาดอยู่ที่ 0.2 ถึง 0.8 มม. รูที่มีขนาดเล็กจะทำให้ฉีดพลาสติกออกมาได้ละเอียด ทำให้ได้ชิ้นงานที่เรียบเนียนยิ่งขึ้น

วัสดุที่ใช้ขึ้นรูปชิ้นงานในเครื่องพิมพ์ 3 มิติแบบ FDM วัตถุดิบที่สำคัญคือ เส้นพลาสติกหรือ Filament มีด้วยกันหลายวัสดุและหลายสี เช่น

2.2.2.1 เส้นพลาสติกแบบ ABS (Acrylonitrile Butadiene Styrene)

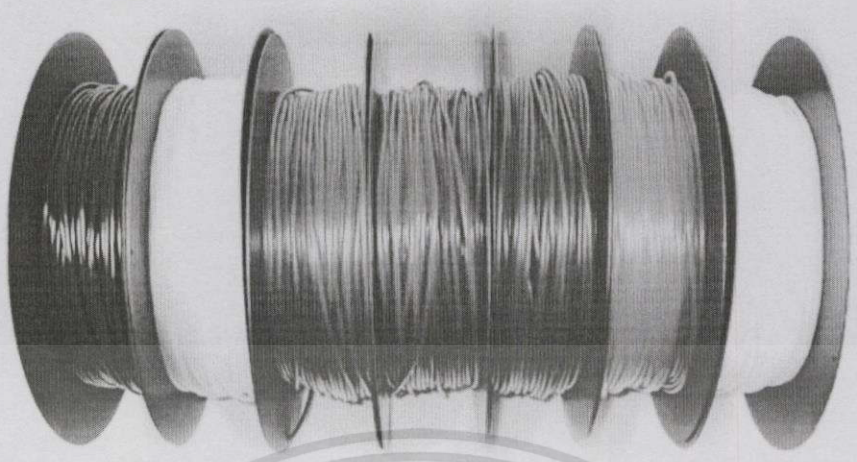
เป็นเส้นพลาสติกที่ได้รับความนิยมชนิดหนึ่งในงานพิมพ์ชิ้นงาน 3 มิติ โดยใช้อุณหภูมิที่หัวฉีดร้อนอยู่ที่ 215 ถึง 250 องศาเซลเซียส แต่มีข้อเสียคือ เมื่อหลอมแล้วจะเกิดไอระเหยออกมาที่เป็นอันตรายต่อคนและสัตว์เลี้ยง เส้นพลาสติก ABS นี้มีส่วนผสมของอะซิโทน (Acetone) ทำให้พื้นผิวของชิ้นงานที่ขึ้นรูปเสร็จแล้วมีความเรียบเนียน เงางามเมื่อใช้ ABS ในการพิมพ์ชิ้นงานจำเป็นที่จะต้องใช้ฐานวางชิ้นงานแบบร้อน

2.2.2.2 เส้นพลาสติกแบบ PLA (Polylactic Acid หรือ Polylactide)

เป็นเส้นพลาสติกที่มีส่วนผสมจากวัตถุดิบชีวภาพ เช่น ข้าวโพดหรือมันฝรั่ง ใช้อุณหภูมิที่หัวฉีดพลาสติกอยู่ที่ 160 ถึง 220 องศาเซลเซียส เมื่อเส้นพลาสติก PLA หลอมจะมีกลิ่นคล้ายๆ กับข้าวโพดคั่ว ซึ่งไม่เป็นอันตราย และไม่เป็นพิษต่อสิ่งแวดล้อม นอกจากนั้นเมื่อใช้ PLA ในการพิมพ์ชิ้นงานก็ไม่จำเป็นที่จะต้องใช้ฐานวางชิ้นงานแบบร้อน แต่ถ้าใช้ก็จะทำให้ฐานของชิ้นงานเรียบเนียนขึ้น โครงงานนี้เลือกใช้พลาสติกชนิดนี้และไม่ใช้ฐานวางชิ้นงานแบบร้อน

2.2.2.3 เส้นพลาสติกแบบ PVA (Polyvinyl Alcohol)

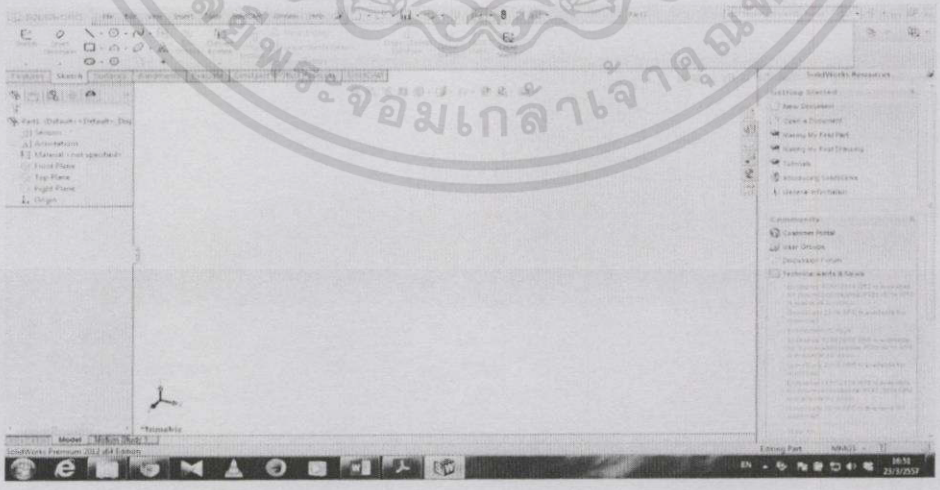
เป็นเส้นพลาสติกชนิดพิเศษที่มีการผสมผสานกันหลายสี ใช้อุณหภูมิที่หัวฉีดพลาสติกอยู่ที่ 190 องศาเซลเซียส วัสดุแบบนี้ละลายน้ำได้ ชิ้นงานที่ขึ้นรูปด้วยเส้นพลาสติกชนิดนี้อาจจะต้องระวังเรื่องความชื้น เพราะอาจส่งผลใช้ชิ้นงานสลายไปได้



รูปที่ 2.5 ตัวอย่างเส้นพลาสติก

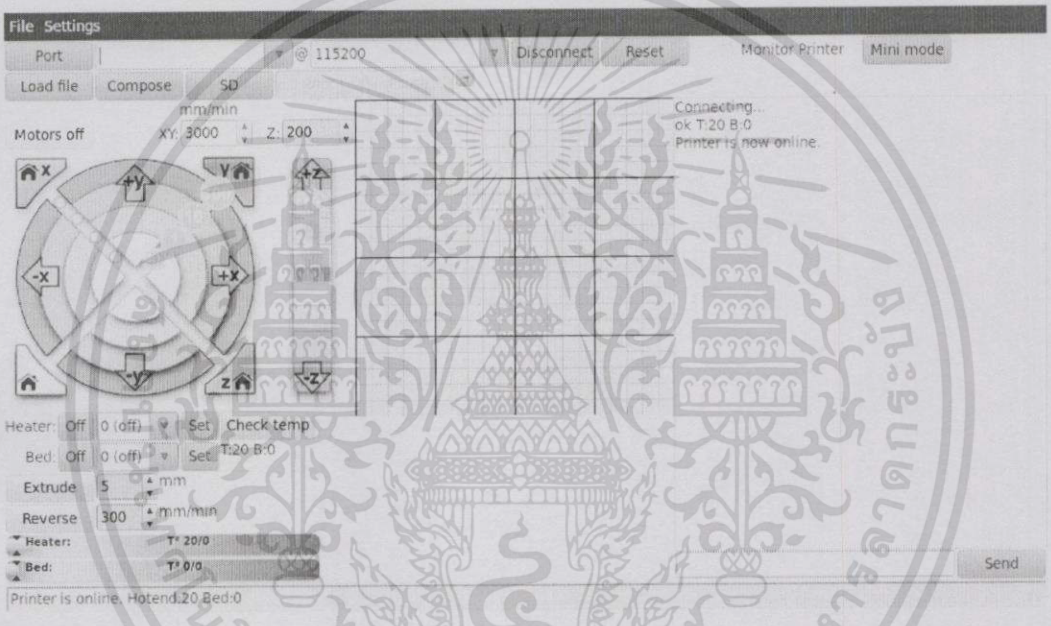
2.2.3 ส่วนโปรแกรม

มีให้เลือกหลายตัวเช่น Solidworks, Autocad, Autodesk หรือ Sketchup ซึ่งผู้ใช้งานจะต้องจัดซื้อเพื่อให้ได้ซอฟต์แวร์เหล่านี้มาใช้งานอย่างถูกต้องบางตัวอาจมีให้ทดลองใช้งานฟรีในระยะเวลาจำกัดแบบฟรีแวร์ (Freeware) ถ้าถามว่าผู้ใช้งานสามารถออกแบบสร้างชิ้นงานสามมิติ โดยไม่ใช้ซอฟต์แวร์เชิงพาณิชย์ได้หรือไม่คำตอบคือได้โดยใช้ซอฟต์แวร์ในกลุ่มฟรีแวร์หรือแบบโอเพนซอร์ส (Open Source) หากแต่ฟังก์ชันการใช้งานอาจมีไม่ครบถ้วน แต่ก็ใช้สร้างชิ้นงานสามมิติได้ในระดับหนึ่งมีผู้พัฒนาหลากหลายตัวที่สะดวกมากที่สุดตอนนี้คือซอฟต์แวร์วาดชิ้นงานสามมิติผ่านเครือข่ายอินเทอร์เน็ตเก็บข้อมูลไว้ในระบบ Cloud ทำให้เข้าถึงการใช้งานได้ทุกที่ที่มีคอมพิวเตอร์แท็บเล็ต หรือสมาร์ตโฟนและเครือข่ายอินเทอร์เน็ต ในที่นี้ขอยกตัวอย่างโปรแกรมคือ SolidWorks



รูปที่ 2.6 ตัวอย่างโปรแกรม SolidWorks

Pronterface เป็นอีกหนึ่งซอฟต์แวร์ที่พัฒนาให้มีความยืดหยุ่นในการทำงานสูง ปรับแต่ง ตั้งค่าได้อย่างอิสระ จึงเหมาะกับเครื่องพิมพ์ 3 มิติแบบโอเพนซอร์ส เช่น RepRap หรือรุ่นอื่นๆ เพียงตั้งค่าให้เหมาะสมกับเครื่องพิมพ์ที่ทำการพัฒนาขึ้นเท่านั้น ตัวซอฟต์แวร์มีความสามารถในการปรับเปลี่ยนตำแหน่ง แนวการวางชิ้นงาน ขนาดของชิ้นงานโดยแปลงไฟล์ .STL เป็น G-code ควบคุมตำแหน่งของระบบเคลื่อนที่ในระนาบต่างๆ ของเครื่องพิมพ์ ควบคุมการให้ความร้อนและอุณหภูมิของหัวฉีดร้อนและฐานวางชิ้นงานแบบร้อนผ่าน GUI ของซอฟต์แวร์ได้ และมีกราฟแสดงค่าอุณหภูมิตลอดการทำงาน เชื่อมต่อกับเครื่องพิมพ์ผ่านพอร์ต USB หรือนำไฟล์เอาต์พุตคัดลอกลงใน SD การ์ด เพื่อนำไปเสียบที่เครื่องพิมพ์ 3 มิติ แล้วพิมพ์ชิ้นงานต่อไป

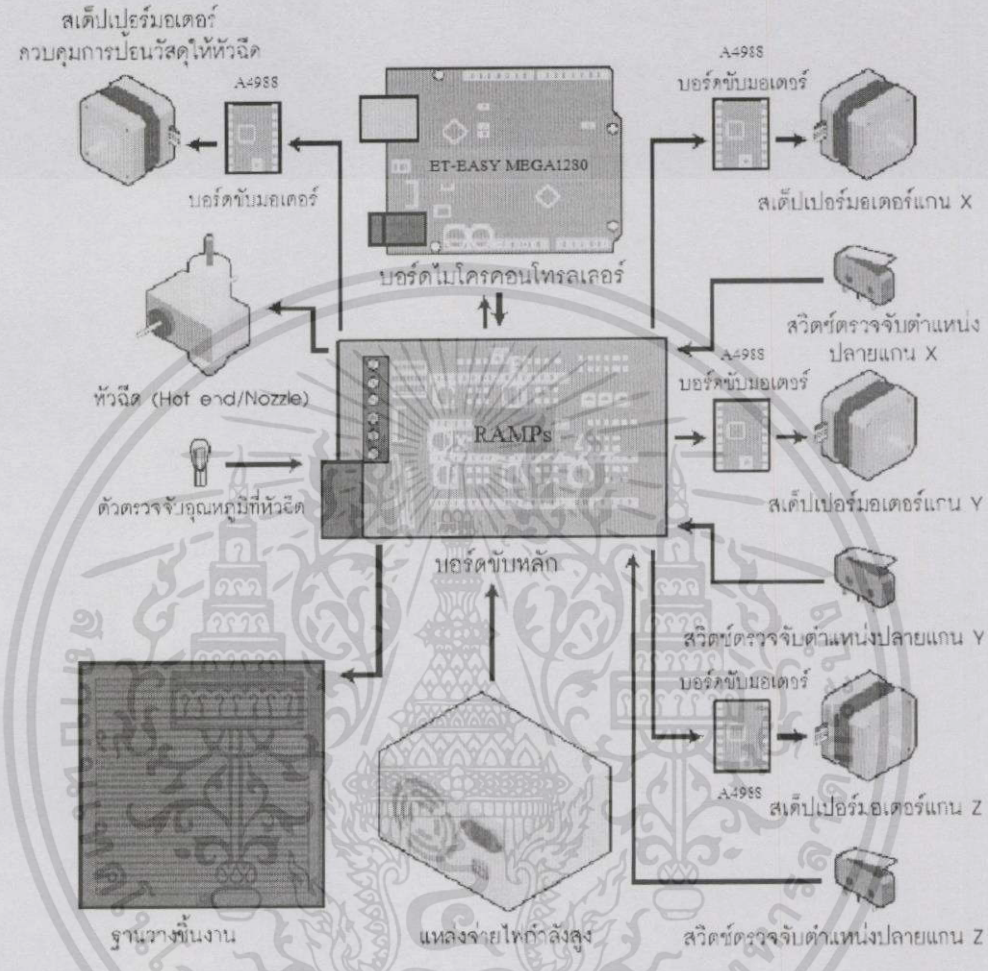


รูปที่ 2.7 ตัวอย่างโปรแกรม Pronterface

2.2.4 ส่วนชุดขับมอเตอร์

จากรูปที่ 2.8แสดงให้เห็นถึงระบบและวงจรควบคุมการทำงานทั้งหมดของเครื่องพิมพ์ 3 มิติแบบ FDM บอร์ดไมโครคอนโทรลเลอร์เป็นตัวประมวลผลและควบคุมหลัก โดยเชื่อมต่อกับส่วนอื่นผ่านวงจรขับหลักที่ได้พลังงานไฟฟ้าจากแหล่งจ่ายไฟตรงกระแสไฟฟ้าสูง เพื่อจ่ายให้กับสเต็ปปิ้งมอเตอร์ผ่านบอร์ดขับมอเตอร์ (Stepping Motor) เพื่อควบคุมให้สเต็ปปิ้งมอเตอร์หมุนตามสเต็ปที่ต้องการซึ่งสัมพันธ์กับตำแหน่งของฐานวางชิ้นงาน มีสเต็ปปิ้งมอเตอร์อีกตัวหนึ่งทำงานเพื่อรีดให้เส้นพลาสติกผ่านเข้าไปยังหัวฉีด (Hot End) ภายใต้การควบคุมให้มีความร้อนที่เหมาะสมกับชนิดของเส้นพลาสติก โดยมีการตรวจสอบอุณหภูมิที่หัวฉีดด้วย ชิ้นส่วนในระบบไมโครคอนโทรลเลอร์และวงจรควบคุมในส่วนชุดขับมอเตอร์ ได้แก่สเต็ปปิ้งมอเตอร์จำนวน 4 ตัวต่อเข้าบอร์ดขับมอเตอร์ A4988 และต่อเข้ากับบอร์ดขับหลัก Shield for Arduino หรือ RAMPs 1.4 โดยบอร์ดหลักนี้จะเชื่อมต่อกับ

แหล่งจ่ายไฟ End Stop ตัวตรวจจับอุณหภูมิที่หัวฉีดจ่ายไฟให้แก่หัวฉีดและประกอบเข้ากับบอร์ดไมโครคอนโทรลเลอร์ ET-EASY MEGA 1280 (DUINO MEGA)

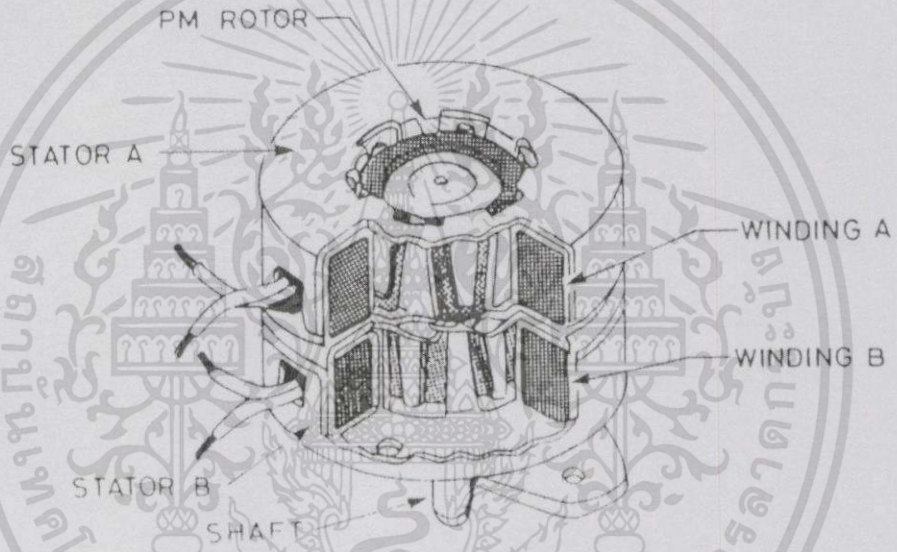


รูปที่ 2.8 ระบบไมโครคอนโทรลเลอร์และวงจรควบคุมการทำงาน

บทที่ 3
หลักการทำงาน

3.1. สเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์ เป็นมอเตอร์ที่ขับเคลื่อนด้วยพัลส์ ลักษณะการขับเคลื่อน จะหมุนรอบแกนได้ 360 องศา มีลักษณะไม่ต่อเนื่อง แต่มีลักษณะเป็นสเต็ป โดยแต่ละสเต็ปจะขับเคลื่อนได้ 1, 1.5, 1.8 หรือ 2 องศา แล้วแต่ละโครงสร้างของมอเตอร์ลักษณะที่นำมอเตอร์ไปใช้ จะเป็นงานที่ต้องการตำแหน่งแม่นยำ โดยในการทำเครื่องพิมพ์สามมิตินี้ จะใช้สเต็ปมอเตอร์แบบแม่เหล็กถาวร (Permanent Magnet Stepping Motor หรือ PM) ชนิด2 ขั้ว (Bipolar Stepping Motor)



รูปที่ 3.1 ลักษณะทั่วไปของสเต็ปปิ้งมอเตอร์

3.1.1 การแบ่งประเภทของสเต็ปปิ้งมอเตอร์

3.1.1.1 การแบ่งประเภทของสเต็ปปิ้งมอเตอร์ตามลักษณะโครงสร้าง

แบบแม่เหล็กถาวร (Permanent Magnet Stepping Motor หรือ PM)

Stepping Motor แบบ PM นี้จะใช้ส่วนของ Stator สำหรับพันขดลวดไว้หลายๆ โพล (Pole) และจะมีส่วน Rotor เป็นรูปทรงกระบอกฟันเลื่อย ซึ่งส่วนของ Rotor นี้จะทำด้วยแม่เหล็กถาวร โดยเมื่อป้อนไฟฟ้ากระแสตรงให้กับขดลวดที่พันอยู่บนส่วนของแกน Stator จะทำให้เกิดสนามแม่เหล็กไฟฟ้าขึ้น ในขดลวด Stator และสนามแม่เหล็กไฟฟ้าที่เกิดขึ้นนี้จะทำให้เกิดแรงดันผลักกับขั้วแม่เหล็กของส่วน Rotor จึงเป็นผลให้แกน Rotor ของมอเตอร์สามารถเคลื่อนที่ไปได้ โดย Stepping Motor แบบนี้จะมีแรงฉุดยึดให้ส่วนของ Rotor หยุดอยู่กับที่ได้ดี ถึงแม้ว่าจะยังไม่ได้

จ่ายไฟให้กับขดลวดของมอเตอร์เลยก็ตาม ลักษณะเด่นของ Stepping Motor แบบนี้คือ ให้แรงบิดสูง แต่มักมีความเร็วและระยะทางในการเคลื่อนที่ไม่ละเอียดมากนัก

แบบแปรผันค่ารีลัคแตนซ์ (Variable Reluctance Stepping Motor หรือ VR)

Stepping Motor แบบ VR นี้ส่วนของ Rotor จะสามารถเคลื่อนที่ได้อย่างอิสระโดย Rotor นี้ทำขึ้นจากสารแม่เหล็กกำลังอ่อน (Force Low Magnetic) มีลักษณะเป็นทรงกระบอกฟันเลื่อย ซึ่งจะมีความสัมพันธ์โดยตรงกับจำนวนโพล (Pole) ของขดลวดใน Stator เพื่อใช้สำหรับกำหนดระยะของมุมที่จะหมุนไปในแต่ละครั้งของการเคลื่อนที่ เมื่อเราป้อนกระแสไฟฟ้าให้กับขดลวดใน Stator จะทำให้เกิดแรงบิดเพื่อไปหมุน Rotor ให้เคลื่อนที่ไปในเส้นทางของอำนาจแม่เหล็กที่มีค่า Reluctance ต่ำที่สุด จึงทำให้มอเตอร์แบบนี้มีความเร็วสูงและตำแหน่งของการเคลื่อนที่มีความแม่นยำสูงและมีเสถียรภาพดี แต่มีแรงบิดน้อยกว่าแบบ PM

แบบผสม (Hybrid Stepping Motor หรือ H)

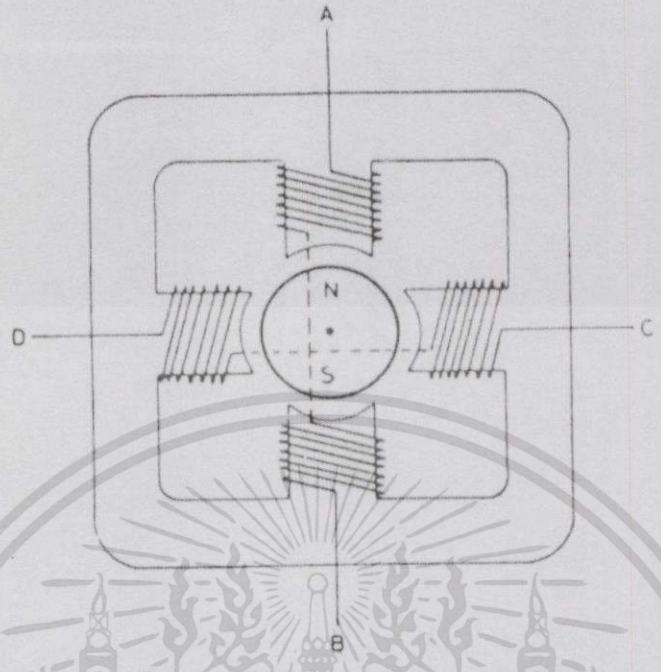
Stepping Motor แบบนี้เป็นการนำเอาข้อดีของมอเตอร์แบบ PM และ VR มารวมกัน โดยจะออกแบบให้ส่วนของ Stator มีลักษณะคล้ายกับมอเตอร์แบบ VR และส่วนของ Rotor มีหมวกหุ้มปลายทำด้วยสารแม่เหล็กกำลังสูง โดยในการออกแบบจะควบคุมขนาดของรูปร่างของหมวกแม่เหล็กเป็นอย่างดี เพื่อให้ได้มุมการหมุนของมอเตอร์ในแต่ละครั้งที่ละเอียดและแม่นยำที่สุด คุณสมบัติเด่นของมอเตอร์แบบนี้ คือ แรงบิดสูง ขนาดกะทัดรัด ระยะการหมุนมีความละเอียดแม่นยำมาก และมีแรงฉุดยึดให้แกน Rotor หยุดนิ่งอยู่กับที่ได้ดี ถึงแม้ว่าจะยังไม่ได้จ่ายกระแสไฟฟ้าให้กับขดลวดของมอเตอร์เลยก็ตาม แต่มอเตอร์แบบนี้มักมีราคาสูงกว่ามอเตอร์แบบอื่นๆ

3.1.1.2 การแบ่งประเภทของสเต็ปปิ้งมอเตอร์ตามโครงสร้างของขดลวด

การที่ Stepping Motor จะสามารถเคลื่อนที่ไปในทิศทางที่เราต้องการได้นั้น เราจะต้องทำการจ่ายกระแสไฟฟ้าให้กับขดลวดของ Motor เพื่อให้เกิดเป็นอำนาจของสนามแม่เหล็กขึ้นอย่างเป็นลำดับที่ถูกต้องและสัมพันธ์กัน ซึ่งเทคนิควิธีการที่ใช้ในการควบคุมการเคลื่อนที่ของ Stepping Motor นั้นสามารถทำได้หลายวิธี โดยในแต่ละวิธีการนั้นต่างก็มีข้อดี/ข้อเสียแตกต่างกัน โดยการควบคุมการเคลื่อนที่ของ Stepping Motor สามารถแบ่งตามโครงสร้างของขดลวดได้ 2 แบบ คือ

1. Stepping Motor แบบ 2 ขั้ว (Bipolar Stepping Motor)
2. Stepping Motor แบบหลายขั้ว (Unipolar Stepping Motor)

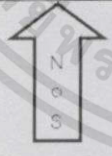
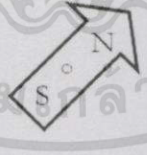
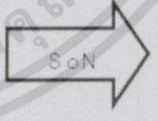
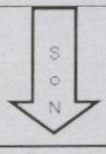
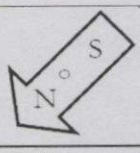
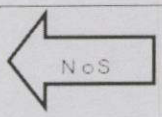
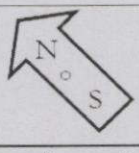
Stepping Motor แบบ 2 ขั้ว (Bipolar Stepping Motor)



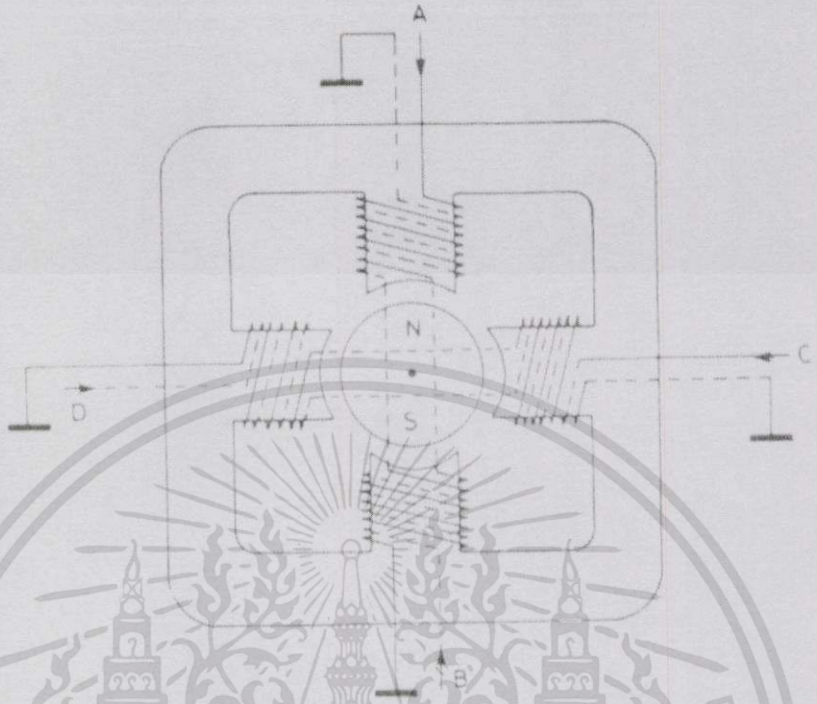
รูปที่ 3.2 โครงสร้างสเต็ปปิ้งมอเตอร์แบบ 2 ขั้ว

สำหรับ Stepping Motor แบบนี้จะประกอบไปด้วยขดลวดจำนวน 2 ชุด คือ ชุด A-B และ C-D ซึ่งเราสามารถสังเกตลักษณะเบื้องต้นของ Stepping Motor แบบนี้ได้โดยดูจากจำนวนสายของขดลวดที่ต่อออกมาภายนอก ซึ่งจะมีเพียง 4 เส้น คือ A, B, C, และ D อย่างละ 1 เส้นโดย A และ B จะเป็นขดลวดเดียวกัน ส่วน C และ D ก็จะเป็นขดลวดเดียวกัน

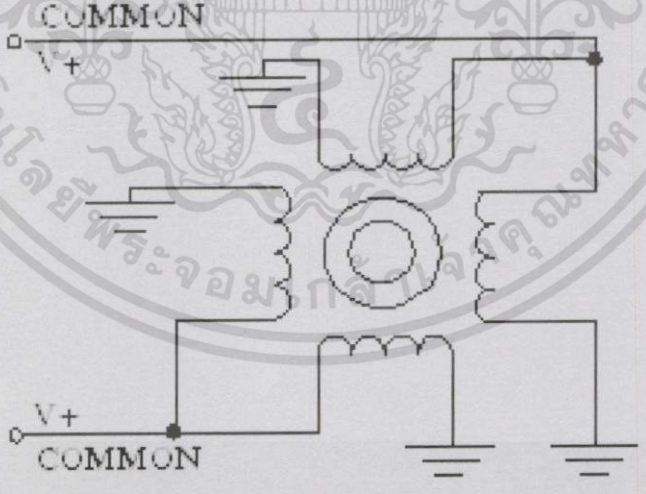
ตารางที่ 3.1 การจ่ายกระแสไฟฟ้าให้ขดลวดสเตเตอร์ของสเต็ปปิ้งมอเตอร์แบบ 2 ขั้ว

ตำแหน่งของ Rotor				
การควบคุมขดลวด A-B	กระแสไหล A➡B	กระแสไหล A➡B	ไม่มีกระแสไหล	กระแสไหล B➡A
การควบคุมขดลวด C-D	ไม่มีกระแสไหล	กระแสไหล C➡D	กระแสไหล C➡D	กระแสไหล ⇔
ตำแหน่งของ Rotor				
การควบคุมขดลวด A-B	กระแสไหล B➡A	กระแสไหล B➡A	ไม่มีกระแสไหล	กระแสไหล A➡B
การควบคุมขดลวด C-D	ไม่มีกระแสไหล	กระแสไหล D➡C	กระแสไหล D➡C	กระแสไหล D➡C

Stepping Motor (Unipolar Stepping Motor)



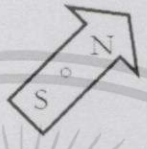
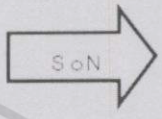
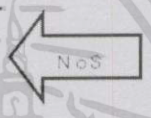
รูปที่ 3.3 โครงสร้างสเต็ปปิ้งมอเตอร์แบบหลายขั้ว



รูปที่ 3.4 สายคอมมอนของสเต็ปปิ้งมอเตอร์แบบหลายขั้ว

สำหรับมอเตอร์แบบนี้จะมีขดลวดประกอบอยู่ด้วยกัน 4 ขด คือ ขดลวด -A ขดลวด -B ขดลวด -C และขดลวด -D ลักษณะโดยทั่วไปของมอเตอร์แบบนี้คือจะมีสายต่อออกมาภายนอกหลายเส้น บางแบบอาจมีถึง 8 เส้น แต่บางแบบมีเพียง 5 เส้น โดยเป็นสายต่อของขดลวด 4 เส้นและสายคอมมอนของขดลวดทั้งหมดรวมกันอีก 1 เส้นโดยมอเตอร์ที่ใช้ในการทดลองนี้มี 6 เส้น คือมีสายคอมมอน 2 เส้น

ตารางที่ 3.2 การจ่ายกระแสไฟฟ้าให้ขดลวดสเตเตอร์ของสเต็ปปิ้งมอเตอร์แบบหลายขั้ว

ตำแหน่งของ Rotor				
ขดลวดที่กระแสไหลผ่าน	เฉพาะขดลวด A	ขดลวด A และ C	เฉพาะขดลวด C	ขดลวด B และ C
ตำแหน่งของ Rotor				
ขดลวดที่กระแสไหลผ่าน	เฉพาะขดลวด B	ขดลวด B และ D	เฉพาะขดลวด D	ขดลวด D และ A

3.1.2 การควบคุมสเต็ปปิ้งมอเตอร์

ในการควบคุมมอเตอร์เพื่อที่จะให้มอเตอร์หมุน มีวิธีการควบคุมกระแสไฟที่จ่ายให้กับขดลวดสเตเตอร์ (Stator) ในแต่ละเฟสของมอเตอร์อย่างเป็นลำดับที่แน่นอน โดยถ้าหากเราต้องการให้กระแสไหลในเฟสใดๆ ก็จะทำให้สถานะของเฟสนั้นๆ เป็นสถานะลอจิก "1" และในการกระตุ้นเฟสของมอเตอร์มีอยู่ด้วยกัน 2 แบบคือ

3.1.2.1 การกระตุ้นเฟสแบบฟูลสเต็ปมอเตอร์ (Full Step Motor)

สามารถแบ่งการกระตุ้นเฟสออกได้เป็นอีก 2 วิธีด้วยกันคือ

1. **การกระตุ้นเฟสแบบฟูลสเต็ป 1 เฟส** (Single-Phase Driver) หรือแบบเวฟจะเป็นการป้อนกระแสไฟให้กับขดลวดของมอเตอร์ทีละขด โดยจะป้อนกระแสเรียงตามลำดับกันไป ดังนั้นกระแสที่ไหลในขดลวดจะทำการไหลในทิศทางเดียวกันทุกขด ลักษณะเช่นนี้จึงทำให้แรงขับของมอเตอร์มีน้อย

ตารางที่ 3.3 การกระตุ้นเฟสแบบฟูลสเต็ป 1 เฟส

Step	#1	#2	#3	#4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

2. การกระตุ้นเฟสแบบฟูลสเต็ป 2 เฟส (Two-Phase Driver) เป็นการป้อนกระแสให้กับขดลวด 2 ขด ของมอเตอร์พร้อมๆ กันไป และจะกระตุ้นเรียงถัดกันไปเช่นเดียวกับแบบหนึ่งเฟส ดังนั้นการกระตุ้นแบบนี้จึงต้องใช้กำลังไฟมากขึ้น และจะทำให้มีแรงบิดของมอเตอร์มากกว่าการกระตุ้นแบบ 1 เฟส

ตารางที่ 3.4 การกระตุ้นเฟสแบบฟูลสเต็ป 2 เฟส

Step	#1	#2	#3	#4
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

3.1.2.2 การกระตุ้นเฟสแบบฮาล์ฟสเต็ป (Half Step Motor)

หรือ One-Two Phase Driver คือการกระตุ้นเฟสแบบฟูลสเต็ป 1 เฟส และ 2 เฟส เรียงลำดับกันไป แรงบิดที่ได้จากการกระตุ้นเฟสแบบนี้จะมีเพิ่มมากขึ้น เพราะช่วงของสเต็ปมีระยะสั้นลง ในการกระตุ้นแบบนี้เราจะต้องมีการกระตุ้นที่เฟสถึง 2 ครั้ง จึงจะได้ระยะของสเต็ปเท่ากับการกระตุ้นเพียงครั้งเดียวของแบบฟูลสเต็ป 2 แบบแรก ความละเอียดของการหมุนตำแหน่งองศาต่อสเต็ปก็เป็นสองเท่าของแบบแรก ความถูกต้องของตำแหน่งที่กำหนดจึงมีมากขึ้น

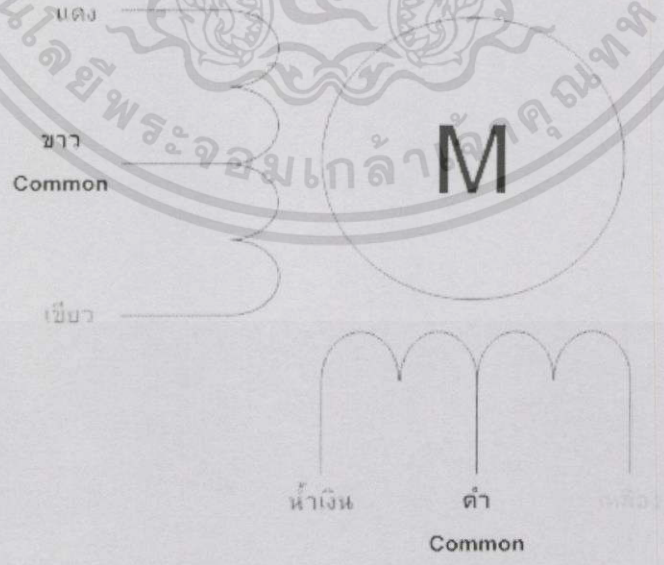
ตารางที่ 3.5 การกระตุ้นเฟสแบบฮาล์ฟสเต็ป

Step	#1	#2	#3	#4
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1

รูปที่ 3.5 สเต็ปปิ้งมอเตอร์ชนิด PM

3.1.3 การหาสายเฟสและสายคอมมอน

ก่อนอื่นเราจะต้องหาสายเฟสและสายคอมมอนของมอเตอร์ก่อน โดยทำการวัดค่าความต้านทานจับคู่แต่ละสาย ถ้าหากว่าเข้าคู่เฟสกับคอมมอนจะได้ค่าความต้านทานค่าน้อย หากวัดคู่ขาเฟสกับเฟสค่าความต้านทานจะสูงมาก ดำเนินการวัดโดยจับคู่ทีละคู่ไปเรื่อยๆ และเนื่องจากรุ่นนี้เป็นมอเตอร์แบบ 6 สายมีสายคอมม่อน 2 เส้น เพราะมีขดลวด 2 ชุด ถ้าหากจับคู่ไม่ถูกชุดขดลวด ค่าความต้านทานที่ได้ก็จะมีค่าสูงมากเช่นกัน



รูปที่ 3.6 สายเฟสและสายคอมมอนของสเต็ปปิ้งมอเตอร์

เมื่อหาสายเฟสกับสายคอมม่อนได้แล้ว เราจะนำสายเฟสมาหาลำดับเฟส โดยต่อสายคอมม่อนทั้ง 2 เส้นเข้ากับแหล่งจ่ายไฟ 5V และต่อสายเฟสเข้ากับกราวด์ โดยคั่นกลางด้วยสวิตช์ แล้วทดลองกดสวิตช์ไล่ไปทีละเฟส ถ้าหากลำดับถูกต้องมอเตอร์จะขยับเป็นสเต็ปๆ ไปในทิศทางเดียวกัน ถ้าหากพบมอเตอร์ขยับคนละทิศให้สลับลำดับการกดสวิตช์ไปเรื่อย จนพบลำดับที่ถูกต้อง

3.1.4 การนำ Step Motor ไปใช้หมุนแกน X, Y, Z

3.1.4.1 แกน X

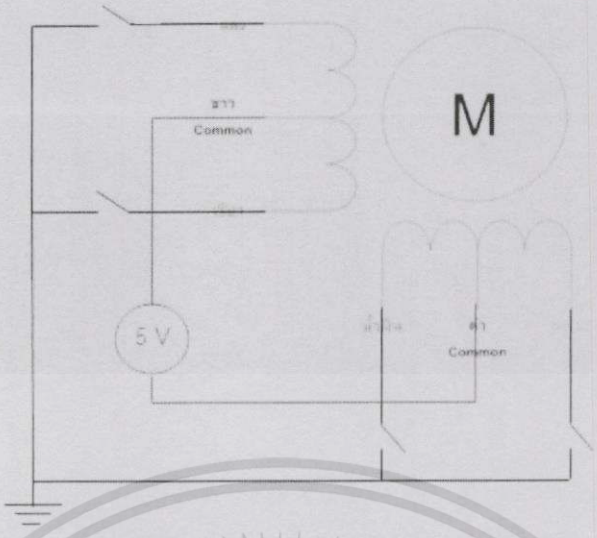
ส่วนของฐานที่ใช้รองรับพลาสติกที่ถูกฉีดออกมาจะถูกใช้ในการเคลื่อนที่ตามแกน X โดยหลักการในการนำมอเตอร์ไปเคลื่อนที่แกนคือ เมื่อมอเตอร์หมุนไปทิศทางใดทิศทางหนึ่ง สายพานที่คล้องอยู่กับเฟืองมอเตอร์จะเคลื่อนที่จากการหมุนของเฟืองมอเตอร์ โดยอีกด้านของสายพานจะเคลื่อนที่ผ่านรอกที่ติดอยู่กับตัวโครง เมื่อสายพานเคลื่อนที่ส่วนของสายพานที่ยึดติดอยู่กับฐานจะทำให้ฐานเคลื่อนที่ โดยจะมีแกนสไลด์และบุชชิ่งรองรับการเคลื่อนที่ และเมื่อมอเตอร์หมุนไปอีกทางจะทำให้ฐานถูกเลื่อนไปอีกด้าน ซึ่งการเคลื่อนที่เข้าออกของฐานก็คือการเคลื่อนที่ตามแนวแกน X นั่นเอง

3.1.4.2 แกน Y

ส่วนของตัวหัวฉีดจะทำหน้าที่เป็นทั้งแกน Y และ Z ซึ่งหลักการการนำมอเตอร์มาเคลื่อนที่ในแกน Y คือ เมื่อมอเตอร์หมุนไปทิศทางใดทางหนึ่งสายพานที่คล้องอยู่กับเฟืองมอเตอร์จะเคลื่อนที่จากการหมุนของเฟืองมอเตอร์โดยอีกด้านของสายพานจะเคลื่อนที่ผ่านรอกเมื่อสายพานเคลื่อนที่ ส่วนของสายพานที่ยึดติดอยู่กับหัวฉีดจะทำให้หัวฉีดเคลื่อนที่ โดยจะมีแกนสไลด์และบุชชิ่งรองรับการเคลื่อนที่ และเมื่อมอเตอร์หมุนไปอีกทางจะทำให้หัวฉีดถูกเลื่อนไปอีกด้าน ซึ่งการเคลื่อนที่ซ้ายขวาของหัวฉีดก็คือการเคลื่อนที่ตามแนวแกน Y นั่นเอง

3.1.4.3 แกน Z

ส่วนของการนำมอเตอร์มาใช้ในการเคลื่อนที่ของแกน Z จะมีหลักการคือ เมื่อมอเตอร์ที่ถูกติดไว้ด้านบนตัวโครงหมุนจะทำให้สกรูเกลียวหมุน และเนื่องจากสกรูเกลียวถูกติดไว้กับแผ่นป้ายคอนโทรล ซึ่งทำให้แผ่นป้ายขึ้นหรือลงตามการหมุนของสกรูเกลียว หัวฉีดที่ติดกับแผ่นป้ายก็จะขึ้นหรือลงตามแผ่นป้าย และเมื่อมอเตอร์หมุนไปอีกทิศทางหนึ่ง จะทำให้หัวฉีดขึ้นหรือลงสวนทางในตอนแรก ซึ่งการเคลื่อนที่ขึ้นหรือลงของหัวฉีดก็คือการเคลื่อนตามแนวแกน Z นั่นเอง



รูปที่ 3.7 การจ่ายกระแสไฟฟ้าให้สเต็ปปิ้งมอเตอร์

3.2 ชุดขับมอเตอร์

3.2.1 บอร์ด ET-EASY MEGA 1280 (DUINO MEGA)

บอร์ดนี้มี 54 Input/Output Digital ,16 Input Analog, 4 Hardware Serial Ports, 16 MHz Oscillator, การเชื่อมต่อ USB, รูต่อแจ็คไฟและปุ่มรีเซ็ต บอร์ดนี้ได้บรรจุทุกๆ อย่างที่จำเป็นต่อการสนับสนุนไมโครคอนโทรลเลอร์ เพียงแค่เชื่อมต่อกับคอมพิวเตอร์ด้วยสายเคเบิ้ลหรือใช้พลังงานไฟฟ้าจากอะแดปเตอร์ แบตเตอรี่ AC/DC เพื่อที่จะเริ่มต้นใช้งานบอร์ด ซึ่งบอร์ด Mega ได้ถูกออกแบบมาเพื่อให้เข้ากับ Shield for Arduino

คุณสมบัติ

- เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิต ประสิทธิภาพสูงแต่ใช้พลังงานต่ำ ในตระกูล AVR
- สถาปัตยกรรมแบบ RISC (Reduced Instruction Set Computer)
- มีชุดคำสั่ง 135 คำสั่ง และส่วนใหญ่คำสั่งเหล่านี้จะใช้เพียง 1 สัญญาณนาฬิกาในการประมวลผลคำสั่ง
- มีรีจิสเตอร์สำหรับใช้งานทั่วไปขนาด 8 บิต จำนวน 32 ตัว
- ทำงานได้สูงสุดที่ 16 ล้านคำสั่งต่อวินาที (MIPS) เมื่อใช้สัญญาณนาฬิกา 16 เมกะเฮิร์ซ (MHz)
- หน่วยความจำแฟลชสำหรับโปรแกรมขนาด 128 กิโลไบต์ เขียน/ลบได้ 10,000 ครั้ง
- หน่วยความจำแบบ EEPROM ขนาด 4 กิโลไบต์ เขียน/ลบได้ 100,000 ครั้ง
- หน่วยความจำแรมชนิดเอสแรม (SRAM) ขนาด 8 กิโลไบต์
- เก็บข้อมูลได้กว่า 20 ปีที่อุณหภูมิ 85°C และกว่า 100 ปีที่อุณหภูมิ 25°C
- มีระบบโปรแกรมตัวเองอยู่ในตัวชิพ
- สามารถทำการอ่านขณะเขียนได้จริง โดยสามารถล็อกการทำงานได้เพื่อความปลอดภัยของซอฟต์แวร์

• มีการเชื่อมประสานกับ JTAG (IEEE std. 1149.1 Compliant)

คุณสมบัติการเชื่อมต่อกับอุปกรณ์ภายนอก

• มีตัวตั้งเวลาและตัวนับขนาด 8 บิต จำนวน 2 ตัว ที่สามารถแยกโหมดการทำงานจากกันได้ 2 โหมด คือ Prescaler และ Compare

• มีตัวตั้งเวลาและตัวนับขนาด 16 บิต จำนวน 4 ตัว ที่แยกโหมดการทำงานได้ 3 โหมด คือ Prescaler, Compare และ Capture

• มีตัวนับแบบเวลาจริง (Real Time Counter) ที่แยกวงจรกำหนดความถี่ได้

• มี PWM จำนวน 12 ช่องสัญญาณที่สามารถกำหนดความละเอียดได้ 16 บิต

• มีตัวปรับผลการเปรียบเทียบของเอาต์พุต

• มีตัวแปลงสัญญาณอนาล็อกให้เป็นดิจิตอลขนาด 10 บิต จำนวน 16 ช่องสัญญาณ

• มีพอร์ตสื่อสารอนุกรมที่สามารถกำหนดอัตราการรับ/ส่งได้จานวน 4 พอร์ต

• เชื่อมประสานอนุกรมแบบ SPI ได้ทั้งการเป็นมาสเตอร์และสเลฟ (Master/Slave)

• มีการเชื่อมประสานแบบอนุกรมด้วยสายสัญญาณ 2 เส้นแบบส่งข้อมูลแบบเรียงไบต์ (Byte Oriented)

• มีตัวตั้งเวลาแบบวอตช์ด็อกที่สามารถกำหนดการทำงานได้โดยสามารถแยกสัญญาณนาฬิกาได้จากตัวชิพ

• มีตัวเปรียบเทียบสัญญาณแบบอนาล็อกอยู่ในตัว

• มีการรองรับการขัดจังหวะและการเวก-อัพ (Wake-up) เมื่อมีการเปลี่ยนแปลงเกิดขึ้นกับขาของชิพ

คุณสมบัติพิเศษ

• มีระบบเริ่มระบบเมื่อมีการรีเซ็ตและมีระบบตรวจจับการเกิดบราวน์เอาต์ (Brown-out) ที่สามารถกำหนดการทำงานได้

• มีตัวตรวจหาความเที่ยงตรงของออสซิเลเตอร์อยู่ในตัว (Internal Calibrated Oscillator)

• มีแหล่งการขัดจังหวะทั้งภายในและภายนอก (External and Internal Interrupt Sources)

• มีโหมดการทำงานสลีป 6 แบบ คือ : Idle, ADC Noise Reduction, Power-save, Power-down, Standby และ Extended Standby

• อินพุต/เอาต์พุต และตัวถัง

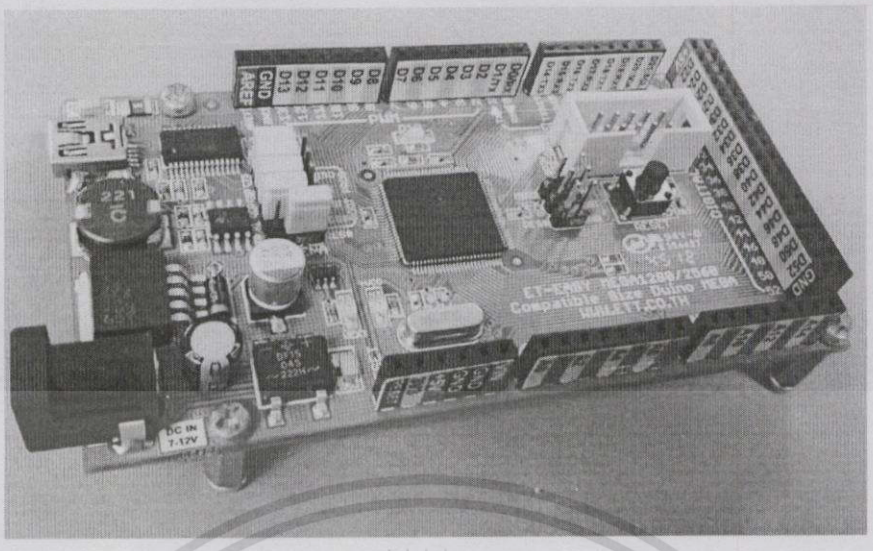
• มีขาของอินพุต/เอาต์พุต ที่สามารถกำหนดการทำงานได้ 86 ขา

• ตัวถังแบบ TQFP ชนิด 100 ขา

• ช่วงอุณหภูมิที่ชิพทำงานได้ -40°C ถึง 85°C

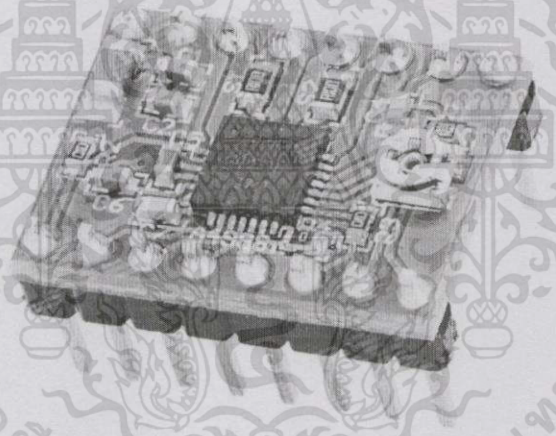
• โหมดการทำงาน: ที่ 1 MHz ต้องการแรงดัน 1.8V กระแส 500 μA

• โหมดเพาเวอร์ดาวน์ (Power-down) ต้องการกระแสเพียง 0.1 μA ที่แรงดัน 1.8V



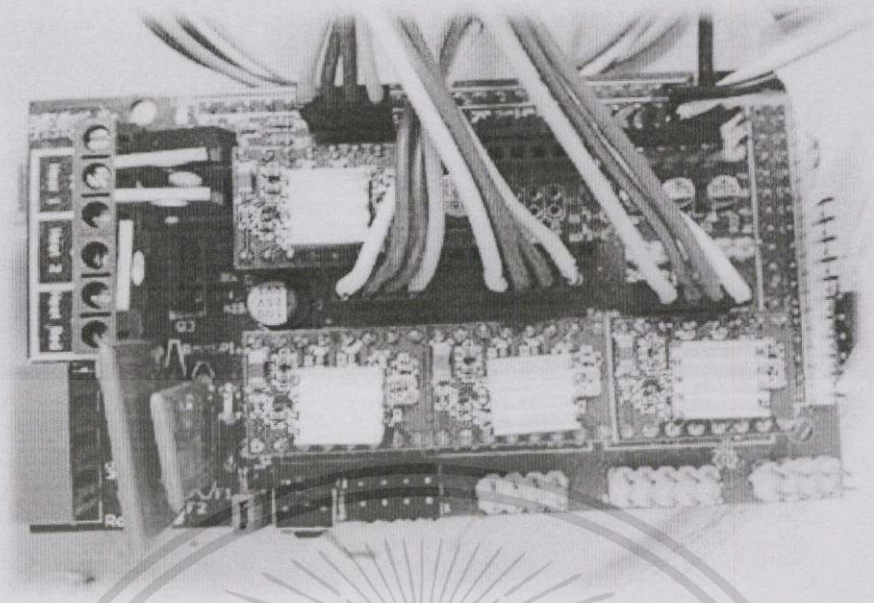
รูปที่ 3.8 บอร์ด ET-EASY MEGA 1280 (DUINO MEGA)

3.2.2 บอร์ดขับมอเตอร์ (Stepping Motor Driver Board A4988)



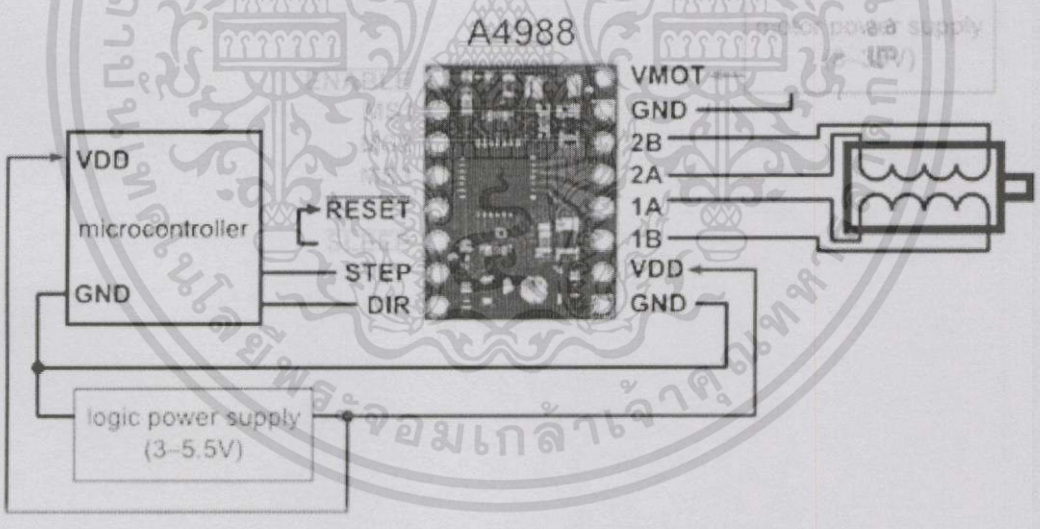
รูปที่ 3.9 Stepping Motor Driver Board (A498)

วงจรขับเคลื่อน Stepping Motor โดยรับคำสั่งการทำงานมาจาก Microcontroller Board ซึ่งในโครงงานนี้ใช้ Microcontroller Board เป็น Arduino MEGA 1280 โดยผ่าน Shield Board Ramp 1.4



รูปที่ 3.10 การใช้ Stepping Motor Driver Board (A498)

3.2.2.1 วิธีการต่อ Stepping Motor Driver Board A4988



รูปที่ 3.11 วงจรของ Stepping Motor Driver Board A4988

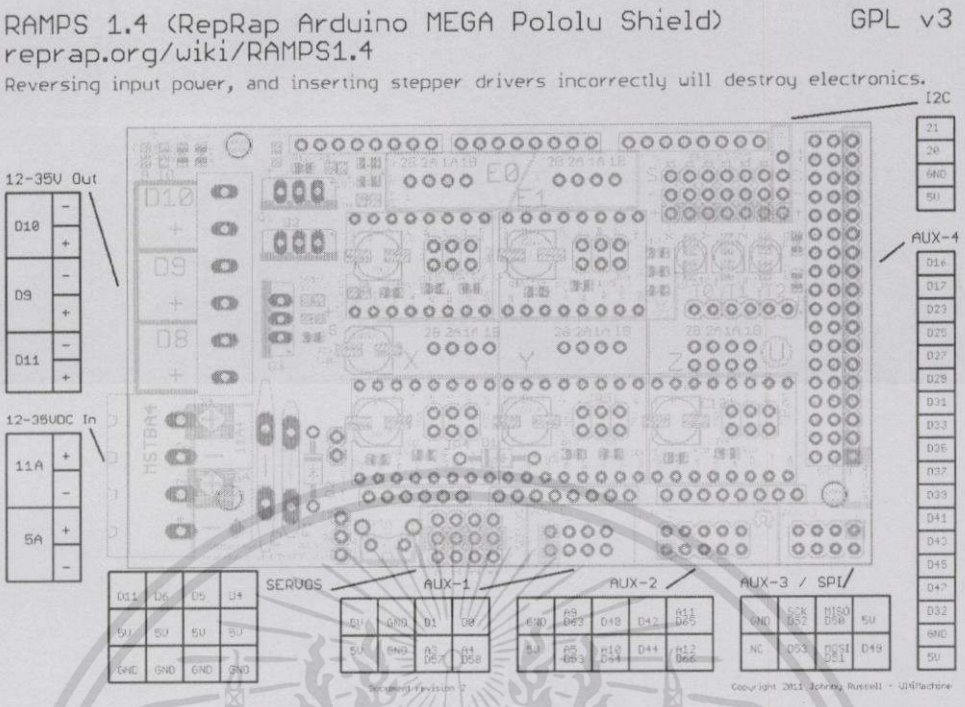
- การจ่ายไฟเลี้ยงให้ Driver จะใช้ไฟฟ้าที่จ่ายจาก Power Supply 12V 5A ให้กับ Arduino MEGA 1280 และ Stepping Motor ก็ได้รับไฟเลี้ยงจาก Arduino MEGA 1280 Board เช่นกัน
- การต่อ Stepping Motor หลังจากหาเฟสได้แล้ว จึงทำการต่อตามรูปที่ 3.11 ได้

3.2.3 Reprap Arduino Mega Pololu Shied หรือ RAMPS 1.4

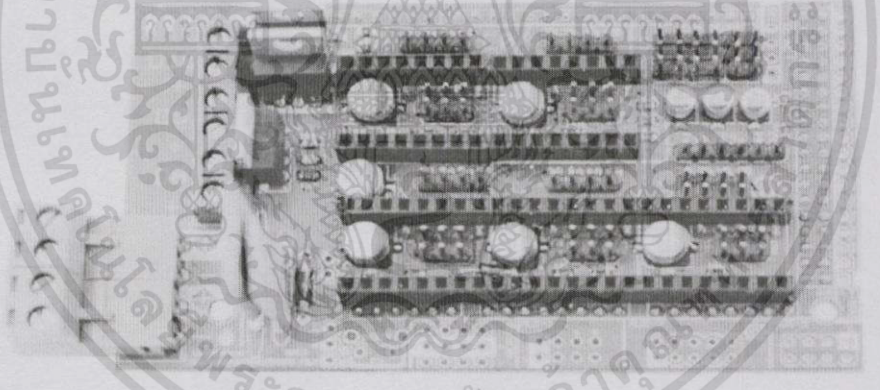
Reprap Arduino Mega Pololu Shied หรือ RAMPS ถูกออกแบบมาเพื่อให้เหมาะสมต่อการใช้งานกับอุปกรณ์อิเล็กทรอนิกส์ของ Reprap รวมในแพคเกจขนาดเล็ก 1 ชุด ในราคาต่ำ RAMPS เชื่อมต่อกับบอร์ด Arduino Mega ด้วยแพลตฟอร์มที่มีประสิทธิภาพ และมีช่องมากมายสำหรับเชื่อมต่อการพัฒนาการต่อไป การออกแบบโมดูลาร์ได้รวมการเชื่อมต่อการขับเคลื่อนมอเตอร์ และส่วนควบคุมหัวฉีดพลาสติกไว้ในบอร์ด RAMPS เพื่อง่ายต่อการใช้งาน

คุณสมบัติ

- ใช้สำหรับเครื่องพิกัด 3 แกน และส่วนหัวฉีด
- สามารถพัฒนาเพื่อใช้ในการควบคุมอุปกรณ์อื่นได้
- มีทรานซิสเตอร์ MOSFETs 3 ตัว สำหรับเครื่องความร้อน พัดลม และวงจรเทอร์มิสเตอร์ 3 วงจร
- ฟิวส์ขนาด 5A เพื่อความปลอดภัย และการป้องกันองค์ประกอบเพิ่มเติม
- การควบคุมการให้ความร้อนฐาน ใช้ฟิวส์ขนาด 11A
- สามารถต่อ Stepper Driver ได้ 5 บอร์ด
- มีขาอยู่ทางด้านบน ทำให้การใส่อุปกรณ์ง่ายต่อการเปลี่ยนแปลง หรือนำออกเพื่อการออกแบบในอนาคต
- มีขา I2C และ SPI ไว้สำหรับการพัฒนาในอนาคตต่อไป
- ทรานซิสเตอร์ MOSFETs ทั้งหมด เชื่อมต่อกับขา PMW เพื่อความเอนกประสงค์
- ลักษณะการเชื่อมต่อกลไกควบคุมใช้ในการต่อกับ End Stops, Motor, LEDs ขาต่อจะถูกชุบด้วยทอง ไฟ 3A มีขนาดเล็กและเป็นสากล
- รองรับ USB ชนิด B
- เพิ่ม SD Card ได้
- เพิ่มช่องต่อสำหรับมอเตอร์ในแกน Z อีก 1 ตัว



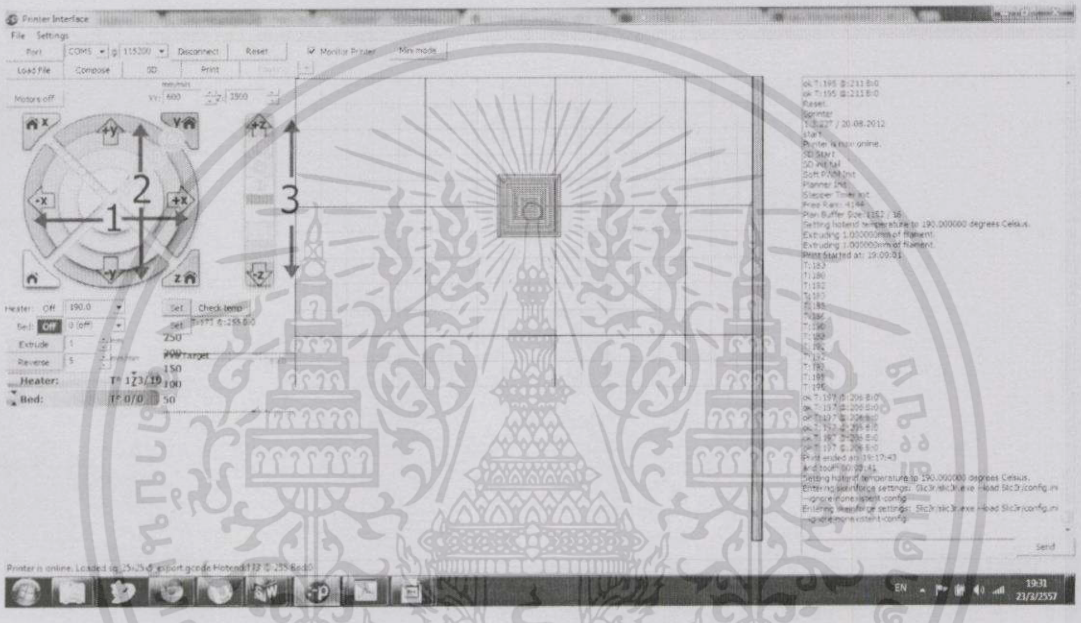
รูปที่ 3.12 วงจรของ Reprap Arduino Mega Pololu Shied



รูปที่ 3.13 Reprap Arduino Mega Pololu Shied หรือ RAMPS 1.4

3.3 การใช้งานโปรแกรมพรอนเตอร์เฟส

คือโปรแกรมมีความสามารถปรับตำแหน่งแนวการวางชิ้นงานและขนาดของชิ้นงาน เป็นโปรแกรมที่ใช้ในการควบคุมแกน 3 แกน ปรับแต่งและตั้งค่าได้อย่างอิสระ จึงเหมาะกับการใช้งานกับเครื่องปริ้นสามมิติ ทำงานโดยการแปลงไฟล์ .STL เป็น G-code ระบุตำแหน่งของระบบเคลื่อนที่ในระนาบต่างๆของเครื่องพิมพ์ควบคุมการให้ความร้อน, อุณหภูมิของหัวฉีดร้อนผ่านโปรแกรม และมีกราฟแสดงค่าอุณหภูมิตลอดการทำงานเชื่อมต่อโปรแกรมในคอมพิวเตอร์กับเครื่องพิมพ์ผ่านพอร์ต USB แล้วใช้โปรแกรมในการทดลองต่อไป



รูปที่ 3.14 โปรแกรม Pronterface

โดยสามารถอธิบายการควบคุมการเคลื่อนที่ของแกนต่างๆ โดยใช้แถบเมนูควบคุมแกนทั้งสามแกนแบบ Manual ตามรูปที่ 3.14 ดังนี้

- หมายเลข 1 เป็นการควบคุมการเคลื่อนที่ของแกน X โดยกดที่หน้าจอในแถบของแกน X ให้เคลื่อนที่ไปในระยะทางและทิศทางที่เราต้องการ ในโปรแกรมพรอนเตอร์เฟสจะนิยมใช้ความยาวของระยะทางในหน่วยมิลลิเมตร จากรูปหากต้องการให้แกน X เคลื่อนที่ไป 10 มิลลิเมตร ในทิศทางขวาจากตำแหน่งเดิมก็สามารถกดที่แถบตรงเลข 10 ในทิศทาง X+

- หมายเลข 2 เป็นการควบคุมการเคลื่อนที่ของแกน Y โดยมีการควบคุมการทำงานเหมือนกับแกน X แต่จะเคลื่อนที่ในทิศไปข้างหน้าและข้างหลัง (Y-, Y+)

- หมายเลข 3 เป็นการควบคุมการเคลื่อนที่ของแกน Z โดยแกน Z จะเคลื่อนที่ในทิศทางขึ้นลง (Z-, Z+) ในระยะทางที่เราต้องการ

- ใช้โปรแกรมพรอนเตอร์เฟสควบคุมความเร็วของสเต็ปปิ้งมอเตอร์ที่ทำหน้าที่ป้อนพลาสติกให้หัวฉีด และยังสามารถวัดค่าความร้อนที่ฮีทเตอร์แล้วแสดงผลออกมาเป็นกราฟได้

นอกจากนี้เราสามารถใช้โปรแกรมพรอนเตอร์เฟสกำหนดความเร็วในการเคลื่อนที่ของแกนต่างๆ ได้ ในการทดลองของเครื่องพิมพ์สามมิตินี้ เราใช้ความเร็วในการเคลื่อนที่ของทั้งสามแกนเป็น 20 มิลลิเมตรต่อวินาที และความเร็วในการขับสเต็ปปิ้งมอเตอร์ให้ป้อนพลาสติกให้กับหัวฉีดด้วยความเร็ว 1 มิลลิเมตรต่อวินาที

บทที่ 4
การทดลองและผลการทดลอง

ในบทนี้จะกล่าวถึงการทดลอง ผลของการทดลอง วงจรและภาคการขับมอเตอร์ ให้มีการเคลื่อนที่ไปตามตำแหน่งที่สั่งการ โดยมีรายละเอียดการสั่งการดังนี้

4.1 การทดลองวงจรและภาคขับมอเตอร์

ในส่วนนี้เป็นการศึกษาเฉพาะส่วนวงจรและภาคขับมอเตอร์ ซึ่งนับตั้งแต่การส่ง G-Code ไปยัง Microcontroller Board Arduino MEGA 1280 จากนั้นตัว Microcontroller ทำการประมวลผลตามที่ได้เขียนโปรแกรมไว้และส่งสัญญาณออกมาเป็น พัลส์ (Pulse) และผ่านอุปกรณ์อิเล็กทรอนิกส์ต่างๆ เพื่อใช้ขับมอเตอร์ไปตามตำแหน่งต่างๆ ซึ่งมีผลการทดลองดังนี้

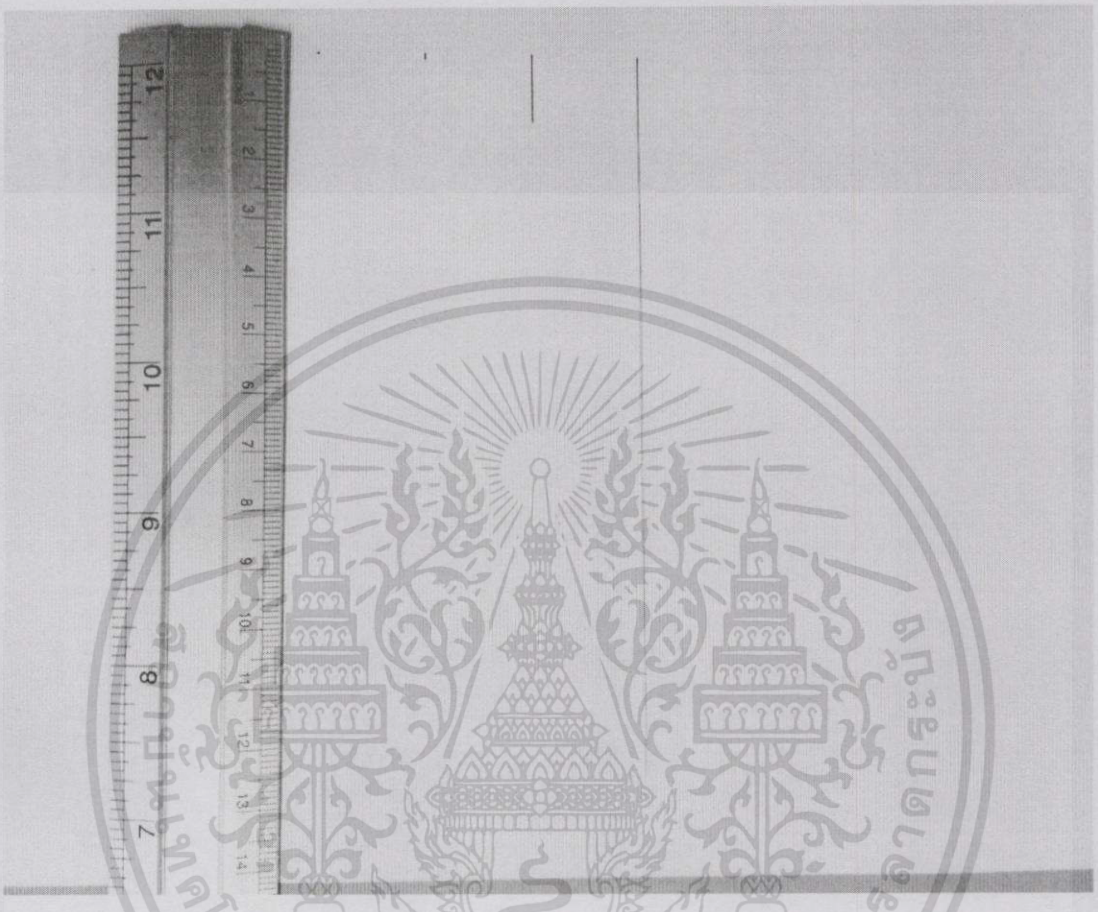
4.1.1 การเคลื่อนที่ตามแกน X

ในส่วนของการเคลื่อนที่ตามแกน X จะทำการสั่งการจากโปรแกรมพรอนเตอร์เฟสให้มอเตอร์หมุนไปตามค่าของโปรแกรมและบันทึกระยะการเคลื่อนที่จากปลายปากกา โดยใช้คำสั่งคือ 0.1 step, 1 step, 10 step และ 100 step ตามลำดับได้ผลการทดลองดังนี้

รูปที่ 4.1 ผลการเคลื่อนที่ตามแกน X

4.1.2 การเคลื่อนที่ตามแกน Y

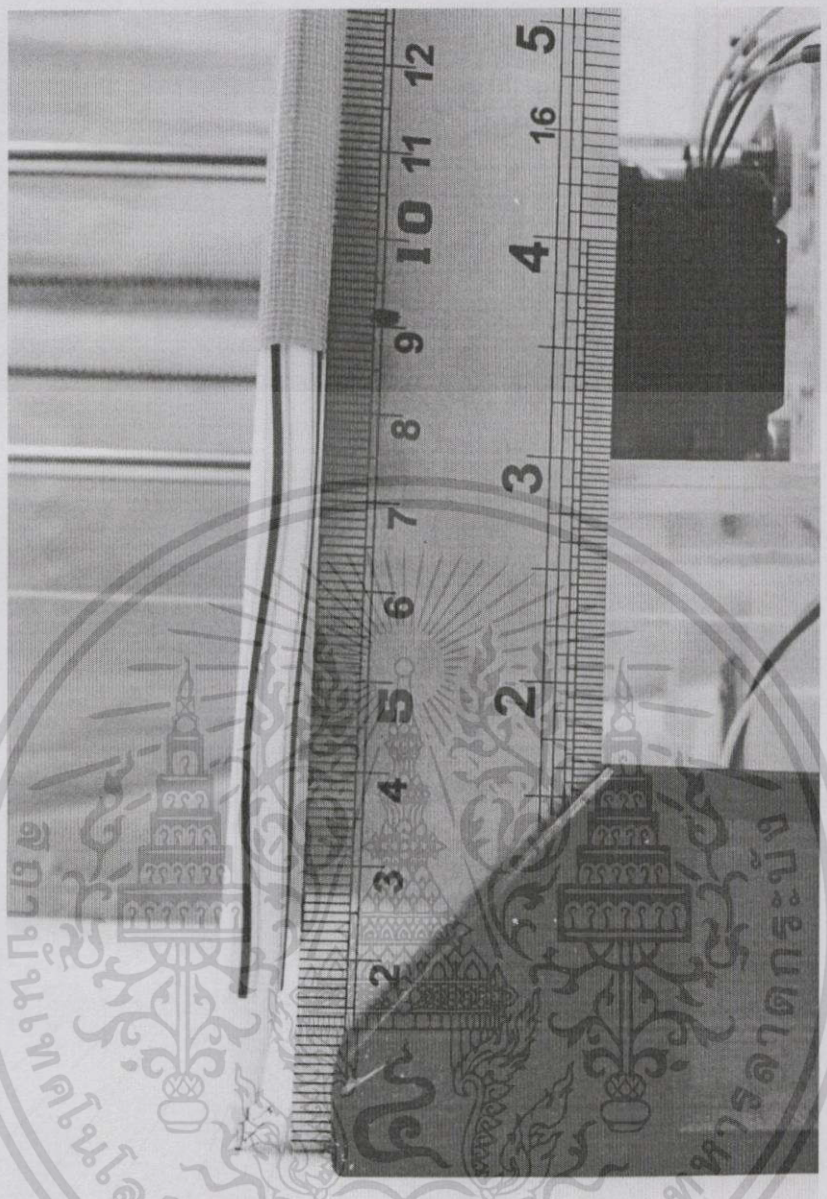
เช่นเดียวกับการทดลองตามแกน X การทดลองการเคลื่อนที่ตามแกน Y จะสั่งการให้มอเตอร์หมุนตามค่า 0.1 step, 1 step, 10 step และ 100 step ตามลำดับได้ผลการทดลองดังนี้



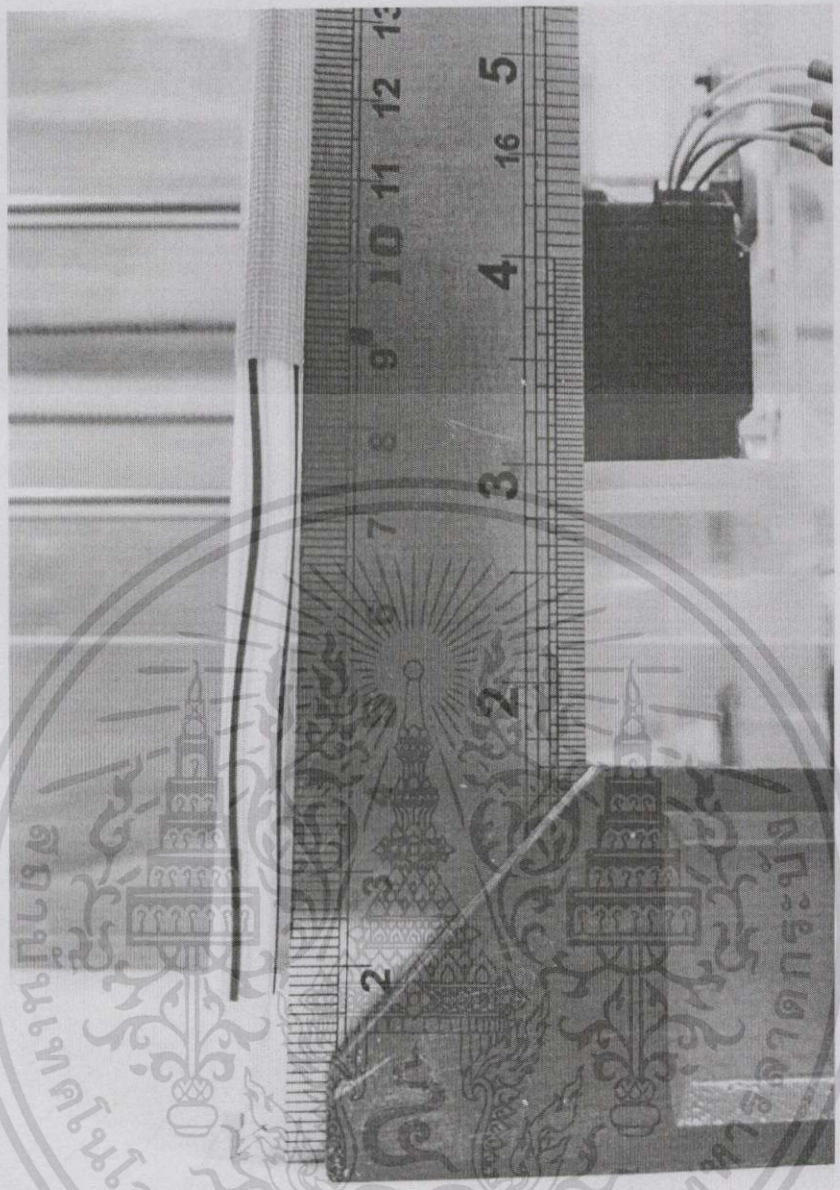
รูปที่ 4.2 ผลการเคลื่อนที่ตามแกน Y

4.1.3 การเคลื่อนที่ตามแกน Z

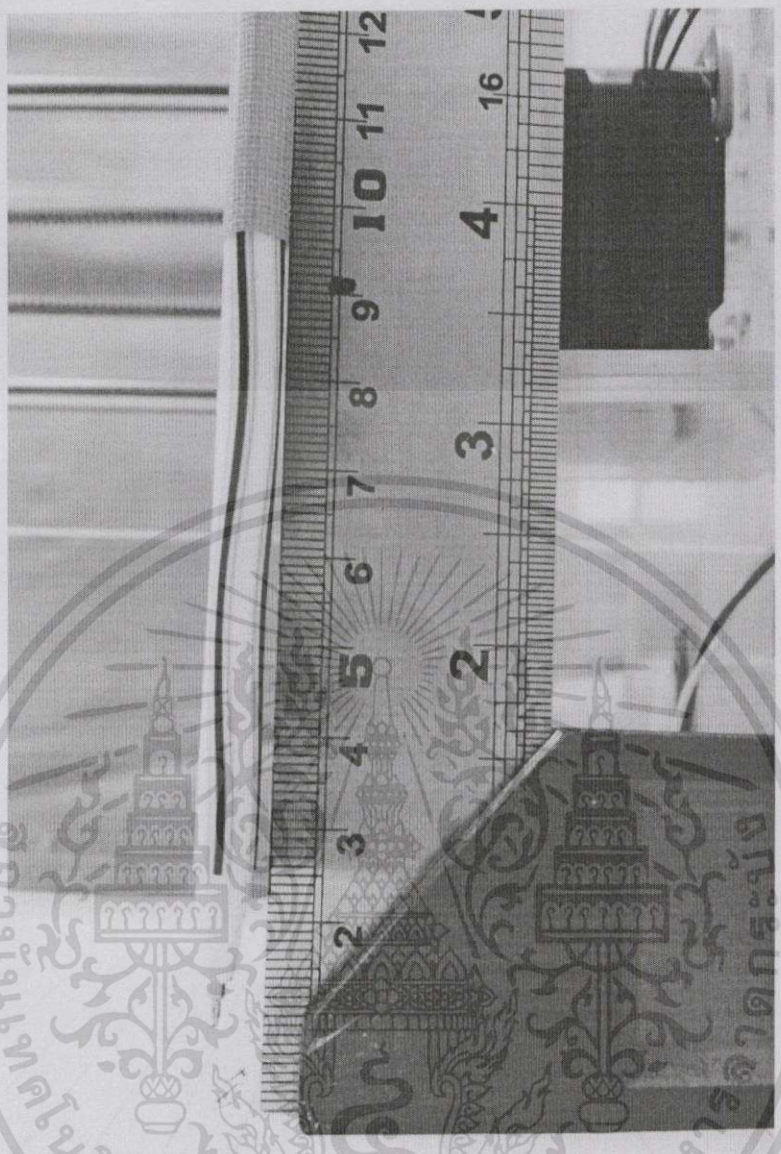
ในส่วนของแกน Z เราจะทำการปรับค่าของแกน Z เพื่อดูความสัมพันธ์ระหว่างค่าที่ปรับและระยะการขึ้นลงของแกน Z โดยในรูปที่ 4.3 เมื่อปลายปากกาอยู่ในจุดเริ่มต้น จากนั้นทำการปรับค่าไปที่ 0.1 step แล้วใช้ไม้บรรทัดวัดระยะการขึ้นลงของปลายปากกา ได้ผลการทดลองดังรูปที่ 4.4 จากนั้นปรับค่าไปที่ 1 step ได้ผลการทดลองดังรูปที่ 4.5 และปรับค่าไปที่ 10 step ซึ่งจะได้ผลการทดลองดังรูปที่ 4.6



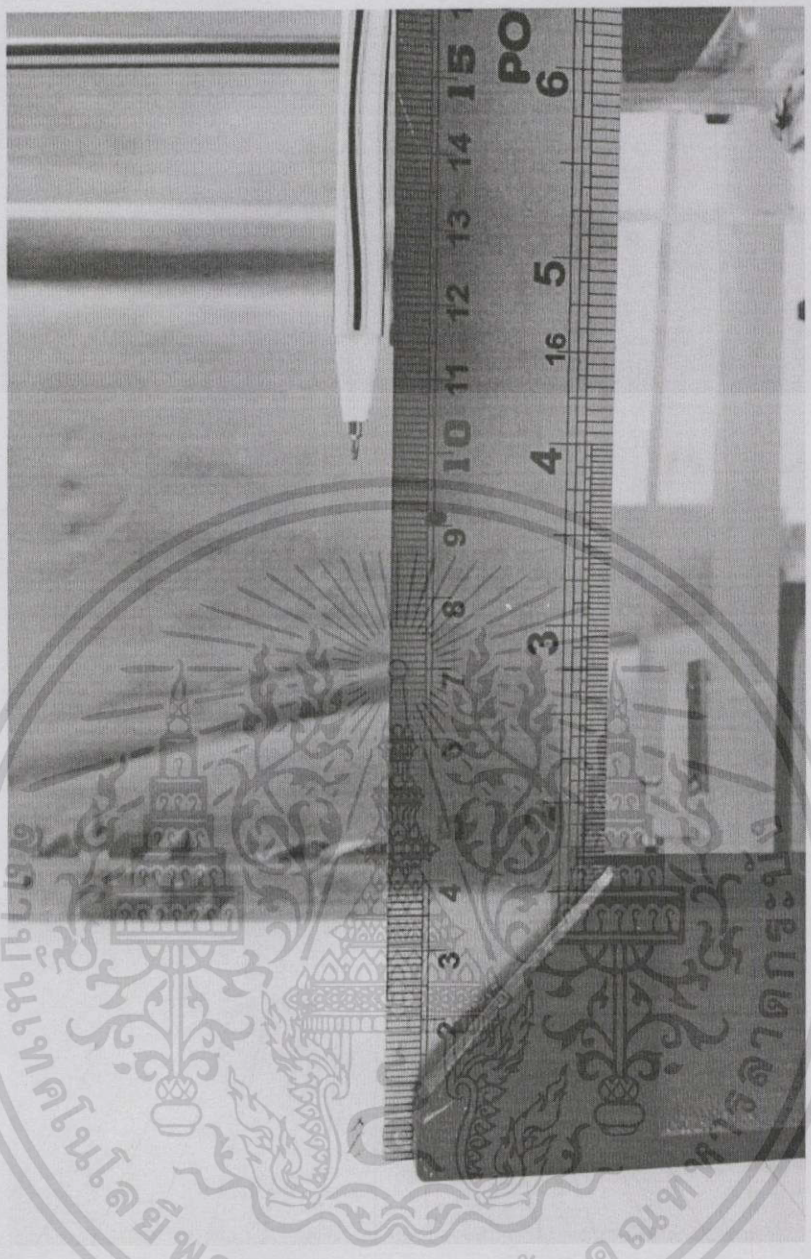
รูปที่ 4.3 ขณะปลายปากกาอยู่ที่จุดเริ่มต้น



รูปที่ 4.4 ปรับค่าไปที่ 0.1 cm/step



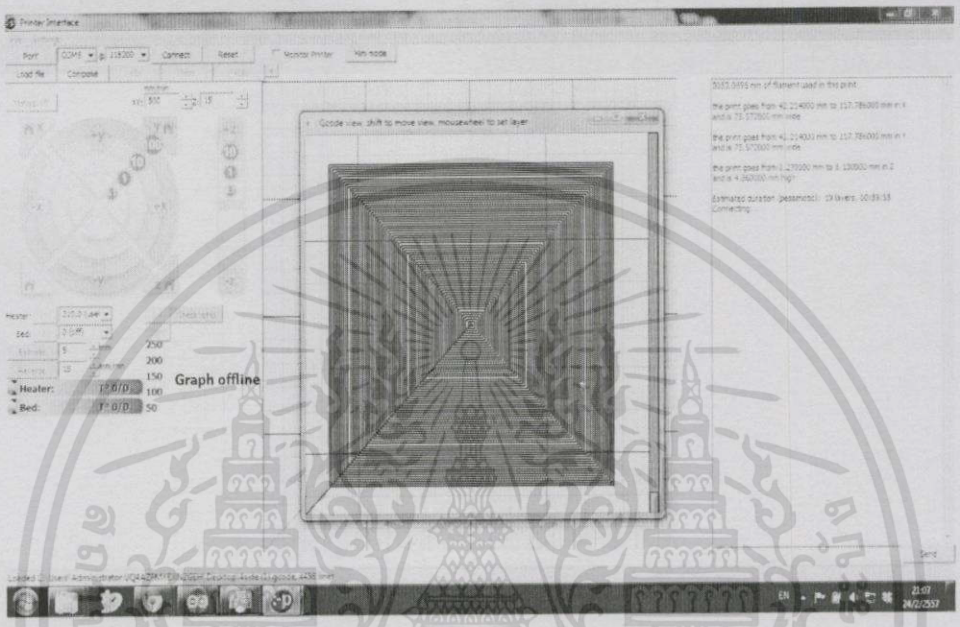
รูปที่ 4.5 ปรับค่าไปที่ 1 cm/step



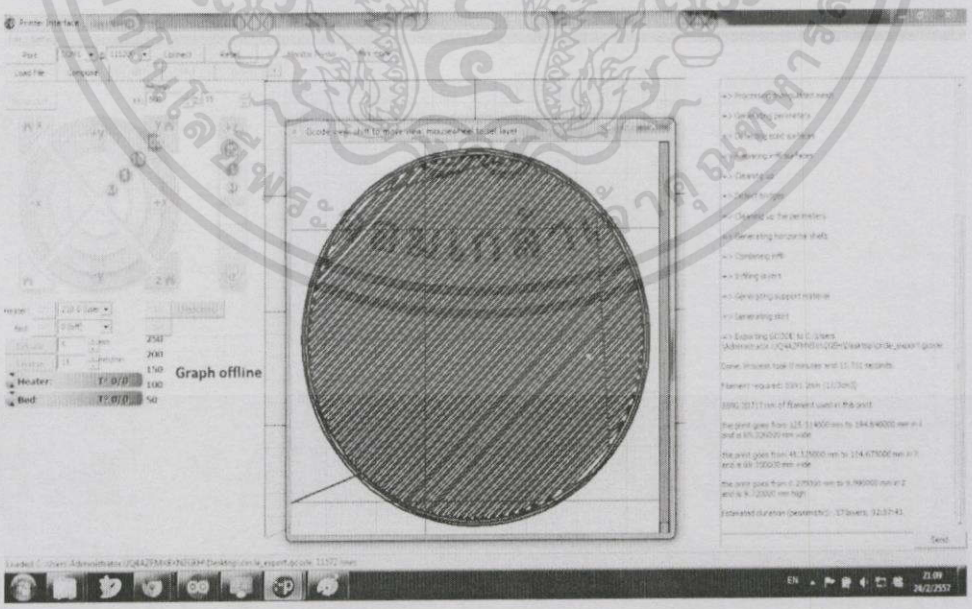
รูปที่ 4.6 ปรับค่าไปที่ 10 cm/step

4.1.4 การเคลื่อนที่ตามแกน XY

ในส่วนของแกน XY เราจะทำการพิมพ์ภาพออกมาเพื่อดูผลการเคลื่อนที่ของแกนมอเตอร์ตามแกน XY โดยใช้ปากกาในการพิมพ์ภาพ เริ่มจากการนำ G-Code ของรูปที่ต้องการพิมพ์ไว้มาใส่เข้าไปที่โปรแกรมพรอนเตอร์เฟส ซึ่งโปรแกรมจะแสดงตำแหน่ง G-Code ที่ป้อนเข้าไปยัง Board Arduino MEGA 1280 ดังรูปที่ 4.7 และรูปที่ 4.8

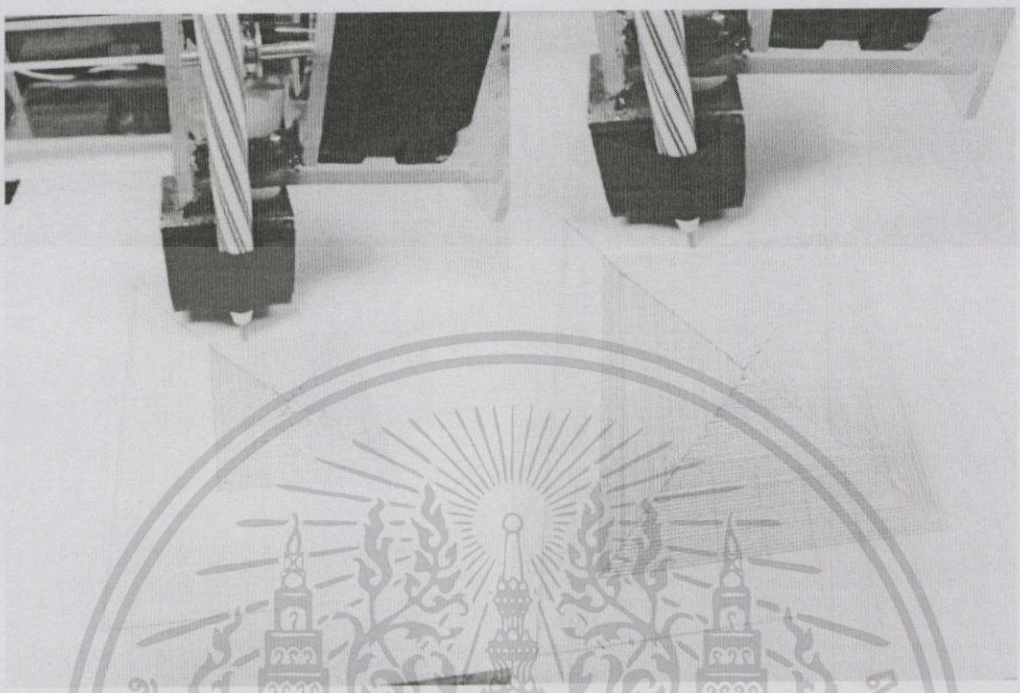


รูปที่ 4.7 ลักษณะวิธีการพิมพ์รูปสี่เหลี่ยม

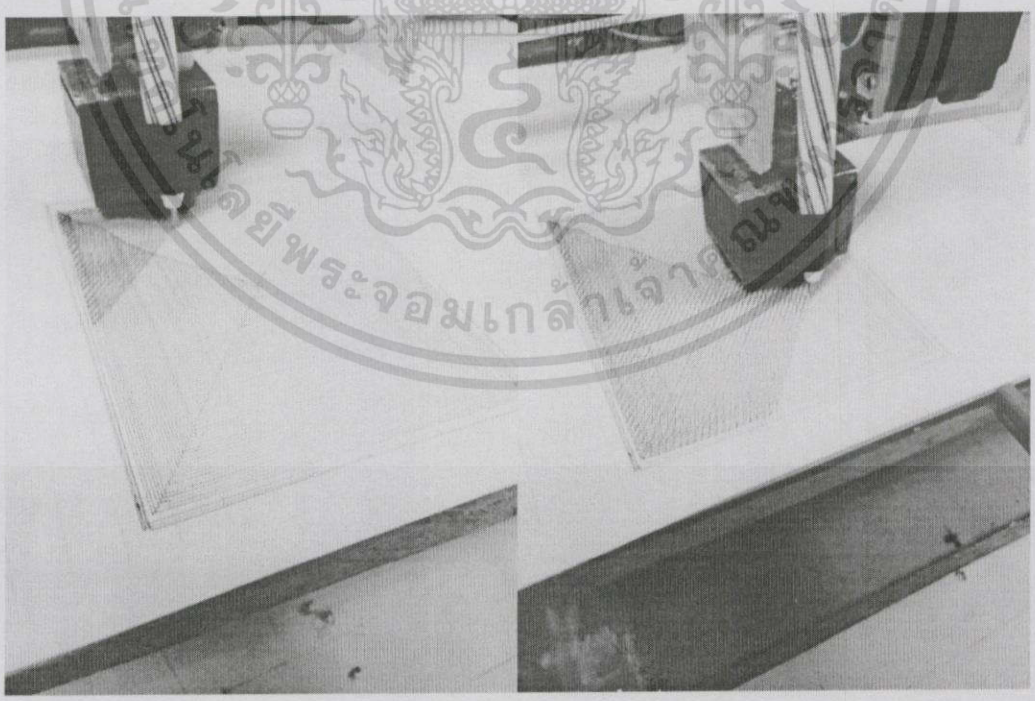


รูปที่ 4.8 ลักษณะวิธีการพิมพ์รูปวงกลม

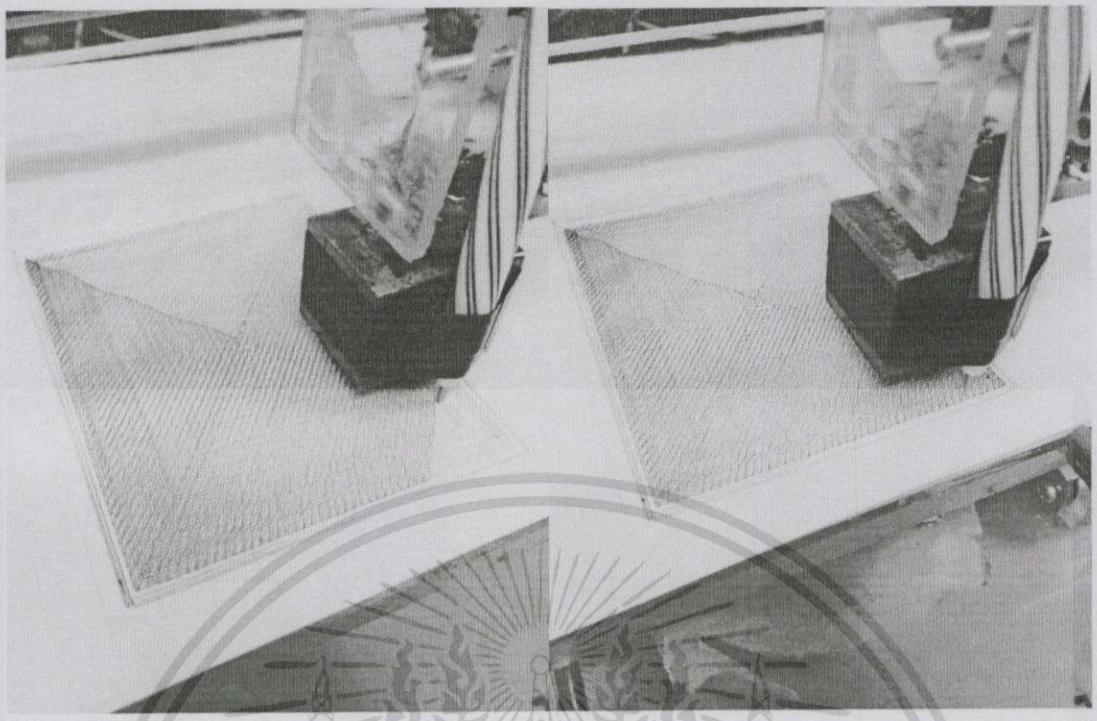
หลังจากนั้นทำการเชื่อมต่อเข้ากับ Board Arduino MEGA 1280 และสั่งให้เริ่มการเคลื่อนโดยใช้ปากกาแทนปลายหัวฉีดได้ผลการทดลองดังนี้



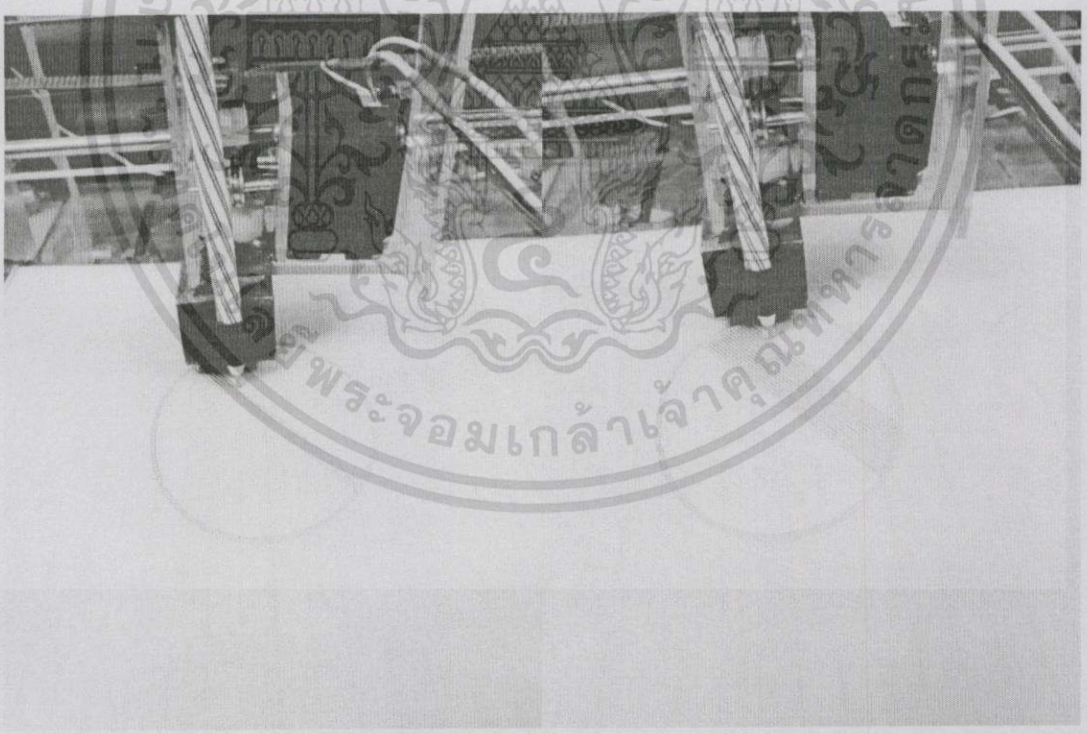
รูปที่ 4.9 ผลการเคลื่อนที่ของมอเตอร์โดยใช้คำสั่งเป็นรูปสี่เหลี่ยม (1)



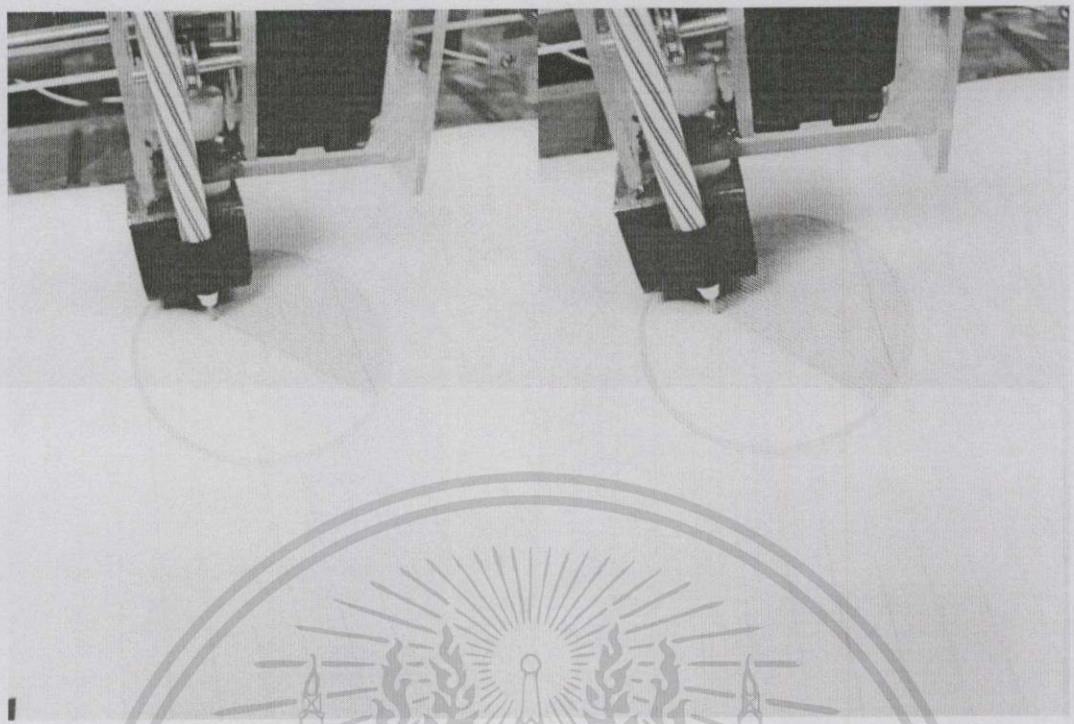
รูปที่ 4.10 ผลการเคลื่อนที่ของมอเตอร์โดยใช้คำสั่งเป็นรูปสี่เหลี่ยม (2)



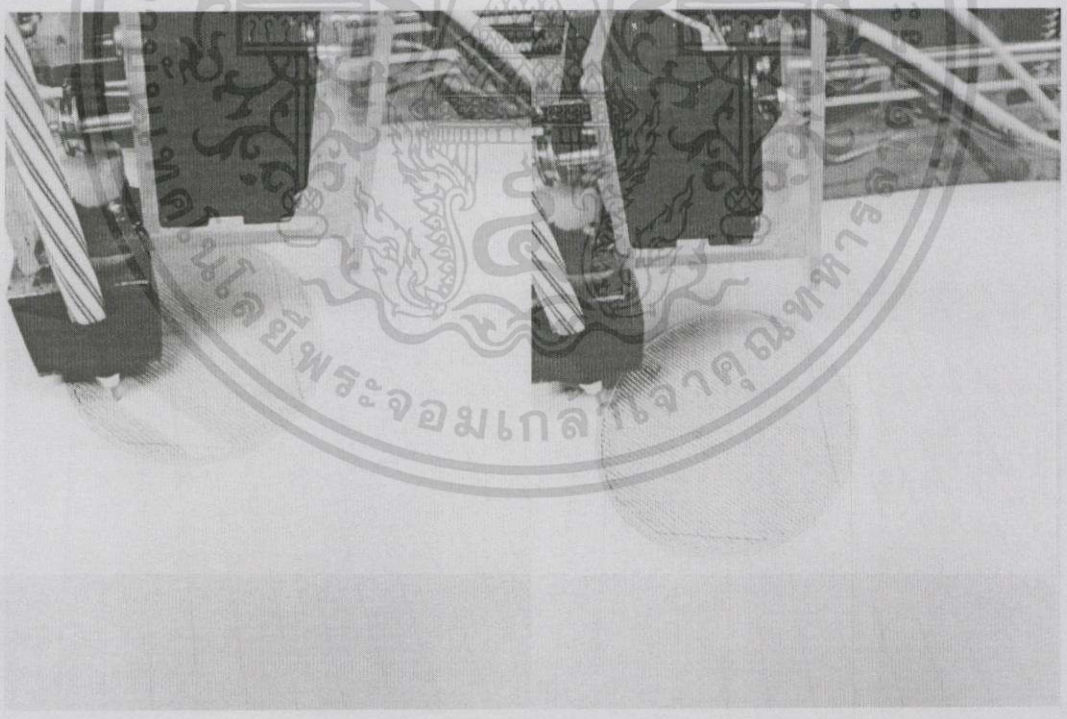
รูปที่ 4.11 ผลการเคลื่อนที่ของมอเตอร์โดยใช้คำสั่งเป็นรูปสี่เหลี่ยม (3)



รูปที่ 4.12 ผลการเคลื่อนที่ของมอเตอร์โดยใช้คำสั่งเป็นรูปวงกลม (1)

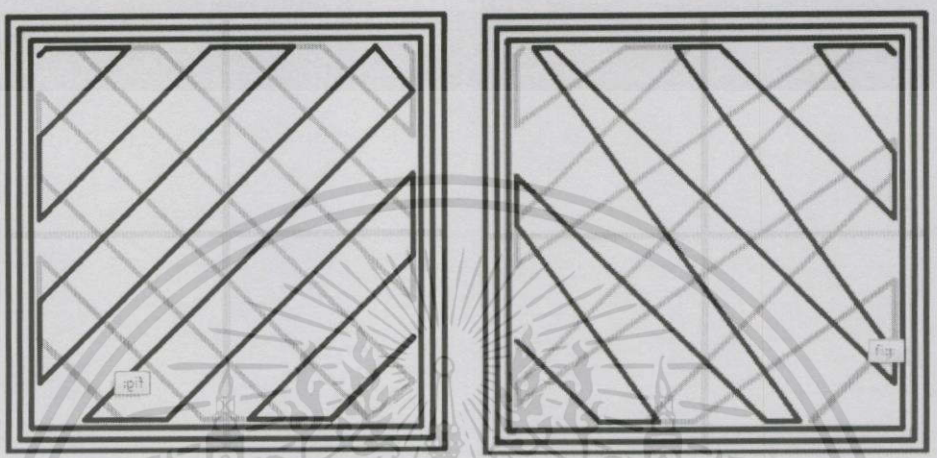


รูปที่ 4.13 ผลการเคลื่อนที่ของมอเตอร์โดยใช้คำสั่งเป็นรูปสี่เหลี่ยม (2)



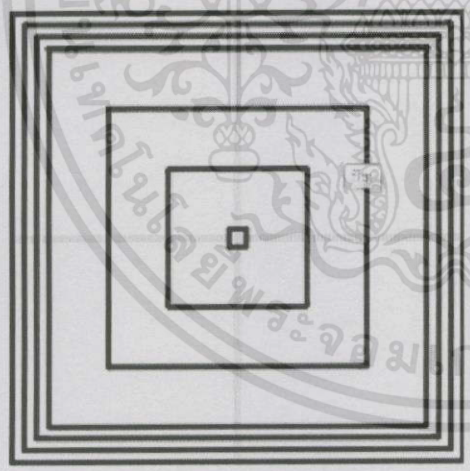
รูปที่ 4.14 ผลการเคลื่อนที่ของมอเตอร์โดยใช้คำสั่งเป็นรูปวงกลม (3)

จากผลการทดลองการเคลื่อนที่ของมอเตอร์ จะเห็นได้ว่าการเคลื่อนที่จะเริ่มจากขอบของชิ้นงานดังแสดงในรูปที่ 4.15 จากนั้นจะเคลื่อนที่เป็นเส้นตรงกลับไปกลับมาภายในขอบเส้นที่ได้วาดไว้ ซึ่งรูปแบบการเคลื่อนที่สามารถกำหนดได้จากโปรแกรมพรอนเตอร์เฟส โดยจะสามารถกำหนดได้ทั้งหมด 7 รูปแบบดังนี้โดยการทดลองนี้เราเลือกใช้รูปแบบการเคลื่อนที่แบบ Concentric

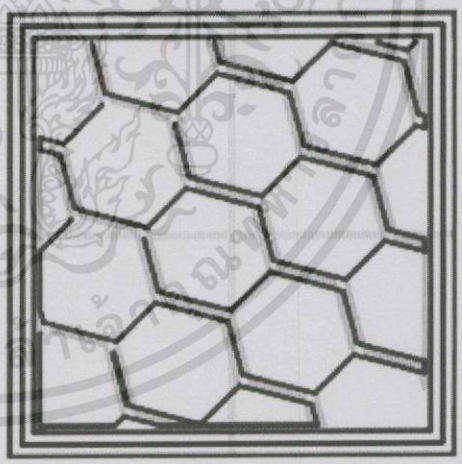


ก. Rectilinear

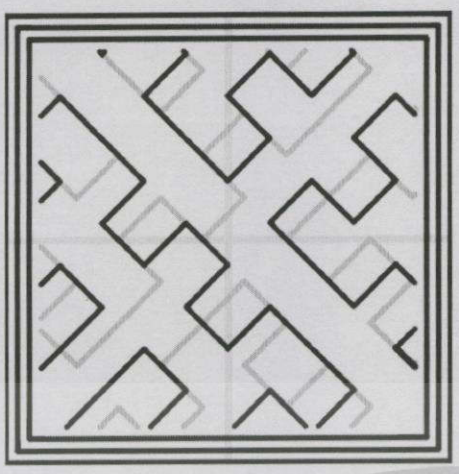
ข. Line



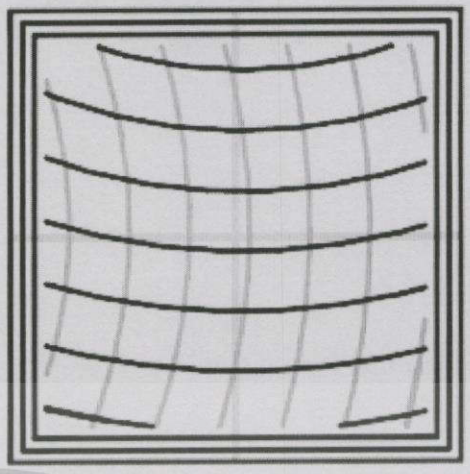
ข. Concentric



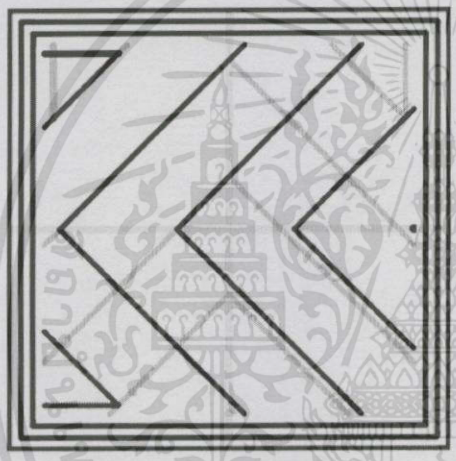
ง. Honeycomb



จ. Hilbert Curve



ฉ. Archimedean Chords



ช. Octgram Spiral

รูปที่ 4.15 แบบการเคลื่อนที่ของมอเตอร์ตามแกน XY จากโปรแกรม Pronterface

บทที่ 5
สรุปผลการดำเนินงาน

จากการทดลอง เมื่ออัพโหลดโค้ดโปรแกรมไปที่บอร์ดอาดูโน่เมกก้า 1280 (Arduino MEGA 1280) บอร์ดจะทำการประมวลผลโปรแกรมและส่งสัญญาณต่อไป จนกระทั่งสัญญาณเข้าไปที่สเต็ปมอเตอร์ จะเห็นได้ว่ามอเตอร์จะควบคุมการทำงานของแกน X, Y และ Z ซึ่งทำให้ปากกาสามารถวาดรูปออกมาได้ตามรูปที่เราได้เขียนขึ้น และปากกาสามารถขยับขึ้นลงในระยะที่เราได้ปรับไว้ ดังนั้นสามารถสรุปได้ว่า โปรแกรมที่นำมาใช้ในการประมวลผล รวมไปถึงอุปกรณ์อิเล็กทรอนิกส์ที่นำมาใช้ร่วมกับบอร์ดประมวลผล Arduino MEGA 1280 สามารถทำให้ปากกาวาดรูปได้ออกมาเหมือนกับที่ได้วาดขึ้นบนคอมพิวเตอร์ แต่ยังมีความคลาดเคลื่อนที่เกิดขึ้นคือ สั่งให้วาดภาพเป็นรูปวงกลมแต่ผลที่ออกมาเป็นรูปวงรี เนื่องจากพารามิเตอร์ของมอเตอร์แต่ละตัวไม่เท่ากัน วิธีแก้ไขคือเข้าไปปรับค่าพารามิเตอร์ในส่วนของโปรแกรม Arduino ให้สั่งสเต็ปปิ้งมอเตอร์หมุนในตำแหน่งที่เหมาะสม

5.1 ปัญหาที่พบและแนวทางแก้ไข

5.1.1 ปัญหาที่พบ

- ตำแหน่งการติดตั้งมอเตอร์และตัวสวิตช์ไม่สัมพันธ์กับตัวโครงสร้าง
- ทดลองกับหัวฉีดจริงไม่ได้เพราะหัวฉีดมีปัญหาด้านการฉีดพลาสติก
- ผลการทดลองที่ได้ไม่เป็นไปตามที่คาดเพราะใช้มอเตอร์ที่มีขนาดและจำนวนเฟืองที่ต่างกัน
- การใช้โปรแกรมพรอนเตอร์เฟสมีปัญหา
- มอเตอร์เคลื่อนที่ไม่สัมพันธ์กับตำแหน่งที่ต้องการ

5.1.2 แนวทางการแก้ปัญหา

- ตัดชิ้นส่วนแผ่นอะคริลิกชิ้นใหม่ให้เหมาะสม
- ใช้ปากกาแทนหัวฉีดเพื่อใช้ในการทดลองและเก็บผลการทดลอง
- ตั้งค่าความเร็วของมอเตอร์ต่อสเต็ปให้สัมพันธ์กัน
- ศึกษาโปรแกรมจากเว็บไซด์ต่างๆ

5.2 ข้อเสนอแนะและแนวทางในการค้นคว้าพัฒนา

สเต็ปปิ้งมอเตอร์ที่ใช้ในปัจจุบันนั้นมีอยู่อีกหลายแบบ จึงควรศึกษาเพิ่มเติมว่ามีตัวสเต็ปปิ้งมอเตอร์แบบใดบ้างที่เหมาะสมกับการเคลื่อนที่เป็นไปตามสเต็ปอย่างแม่นยำ และที่สำคัญความเร็วในการพิมพ์ชิ้นงานแต่ละชิ้นมีการใช้เวลาที่นาน ดังนั้นเพื่อให้ได้ชิ้นงานออกมารวดเร็ว ควรมีการพัฒนาในด้านนี้ด้วย

นอกจากนั้นยังมีโปรแกรมอีกจำนวนมากที่ใช้ในการเขียนภาพสามมิติ จึงควรมีการศึกษาหาโปรแกรมใหม่ๆ เพื่อให้ใช้งานได้กับเครื่องพิมพ์สามมิติ ให้เครื่องพิมพ์สามมิติสามารถผลิตชิ้นงานออกมาได้หลากหลาย

เอกสารอ้างอิง

[1] "Inventor 3D". [Online]. Available: http://www.inventor-3d.com/home/2014.

[2] "A4988 Stepper Motor Driver Carrier". [Online]. Available: http://www.pololu.com/product/11822001.

[3] "RepRap Prusa Mendel V2 Complete Build". [Online]. Available: http://filear.com/?p=267 2012.

ภาคผนวก

ภาคผนวก ก

โปรแกรมควบคุมสเต็ปปิ้งมอเตอร์

โปรแกรมซีพลัสพลัสซึ่งเขียนเพื่อใช้ในการควบคุมสเต็ปปิ้งมอเตอร์โค้ดโปรแกรมการใช้งานหลักสรุปได้ดังต่อไปนี้

Configuration.h

```
#ifndef CONFIGURATION_H
#define CONFIGURATION_H
#define MOTHERBOARD 33
#define THERMISTORHEATER 1
#define THERMISTORBED 1
#define _AXIS_STEP_PER_UNIT {80, 80, 3200/1.25,700}
#define ENDSTOPPULLUPS
const bool X_ENDSTOP_INVERT = false;
const bool Y_ENDSTOP_INVERT = false;
const bool Z_ENDSTOP_INVERT = false;
#define BAUDRATE 115200
#define X_ENABLE_ON 0
#define Y_ENABLE_ON 0
#define Z_ENABLE_ON 0
#define E_ENABLE_ON 0
const bool DISABLE_X = false;
const bool DISABLE_Y = false;
const bool DISABLE_Z = true;
const bool DISABLE_E = false;
const bool INVERT_X_DIR = false;
const bool INVERT_Y_DIR = false;
const bool INVERT_Z_DIR = true;
const bool INVERT_E_DIR = false;
#define X_HOME_DIR -1
#define Y_HOME_DIR -1
#define Z_HOME_DIR -1
const bool min_software_endstops = false;
const bool max_software_endstops = true;
const int X_MAX_LENGTH = 200;
const int Y_MAX_LENGTH = 200;
```

```
const int Z_MAX_LENGTH = 100;
const int NUM_AXIS = 4;
#define _MAX_FEEDRATE {400, 400, 2, 45}
#define _HOMING_FEEDRATE {1500,1500,120}
#define _AXIS_RELATIVE_MODES {false, false, false, false}
#define MAX_STEP_FREQUENCY 30000
define MAX_RETRACT_FEEDRATE 100
#ifdef RAPID_OSCILLATION_REDUCTION
const long min_time_before_dir_change = 30;
#endif
skeinforge 40+, for older versions raise them a lot.
#define _ACCELERATION
#define _RETRACT_ACCELERATION 2000
#define _MAX_XY_JERK 20.0
#define _MAX_Z_JERK 0.4
#define _MAX_E_JERK 5.0
#define _MAX_ACCELERATION_UNITS_PER_SQ_SECOND {5000,5000,50,5000
#define MINIMUM_PLANNER_SPEED 0.05
#define DEFAULT_MINIMUMFEEDRATE       0.0
#define DEFAULT_MINTRAVELFEEDRATE     0.0
#define _MIN_SEG_TIME 20000
const int dropsegments=5;
#define _DEF_CHAR_UUID "00000000-0000-0000-0000-000000000000"
#ifdef SDSUPPORT
  #define BLOCK_BUFFER_SIZE 16
  #define BLOCK_BUFFER_MASK 0x0f
#else
  #define BLOCK_BUFFER_SIZE 16
  #define BLOCK_BUFFER_MASK 0x0f
#endif
#define MM_PER_ARC_SEGMENT 1
#define N_ARC_CORRECTION 25
#define FAN_SOFT_PWM
#define MINIMUM_FAN_START_SPEED  0
#define MINIMUM_FAN_START_TIME  6000    //6sec
#ifdef AUTOTEMP
    #define AUTO_TEMP_MAX 240
    #define AUTO_TEMP_MIN 205
```

```
    #define AUTO_TEMP_FACTOR 0.025
    #define AUTOTEMP_OLDWEIGHT 0.98
#endif
#define HEATER_CURRENT 255
#define HEATER_CHECK_INTERVAL 500
#define BED_CHECK_INTERVAL 5000
#define DISABLE_CHECK_DURING_ACC
#ifndef DISABLE_CHECK_DURING_ACC
#endif
#define DISABLE_CHECK_DURING_TRAVEL 1000
#define MINTEMP 5
#define MAXTEMP 275
#define BED_USES_THERMISTOR
#define CONTROLLERFAN_SEC 60
#define EXTRUDERFAN_DEC 50
#ifdef DEBUG
#endif
#endif
```

Sprinter.h

```
#if defined(ARDUINO) && ARDUINO >= 100
  #include "Arduino.h"
#else
  #include <WProgram.h>
#endif
#include "fastio.h"
extern "C" void __cxa_pure_virtual();
#define  FORCE_INLINE __attribute__((always_inline)) inline
#if X_ENABLE_PIN > -1
#define  enable_x() WRITE(X_ENABLE_PIN, X_ENABLE_ON)
#define disable_x() WRITE(X_ENABLE_PIN,!X_ENABLE_ON)
#else
#define enable_x() ;
#define disable_x() ;
#endif
#if Y_ENABLE_PIN > -1
#define  enable_y() WRITE(Y_ENABLE_PIN, Y_ENABLE_ON)
#define disable_y() WRITE(Y_ENABLE_PIN,!Y_ENABLE_ON)
#else
#define enable_y() ;
#define disable_y() ;
#endif
#if Z_ENABLE_PIN > -1
#define  enable_z() WRITE(Z_ENABLE_PIN, Z_ENABLE_ON)
#define disable_z() WRITE(Z_ENABLE_PIN,!Z_ENABLE_ON)
#else
#define enable_z() ;
#define disable_z() ;
#endif
#if E_ENABLE_PIN > -1
#define  enable_e() WRITE(E_ENABLE_PIN, E_ENABLE_ON)
#define disable_e() WRITE(E_ENABLE_PIN,!E_ENABLE_ON)
#else
#define enable_e() ;
#define disable_e() ;
#endif
```

```
#define X_AXIS 0
#define Y_AXIS 1
#define Z_AXIS 2
#define E_AXIS 3
typedef struct {
  long steps_x, steps_y, steps_z, steps_e;
  unsigned long step_event_count;
  long accelerate_until;
  long decelerate_after;
  long acceleration_rate;
  unsigned char direction_bits;
*_DIRECTION_BIT in config.h)
  #ifdef ADVANCE
    long advance_rate;
    volatile long initial_advance;
    volatile long final_advance;
    float advance;
  #endif
  float nominal_speed;
  float entry_speed;
  float max_entry_speed;
  float millimeters;
  float acceleration;
  unsigned char recalculate_flag;
  unsigned char nominal_length_flag;
  long nominal_rate;
  long initial_rate;
  long final_rate;
  long acceleration_st;
  volatile char busy;
} block_t;
void FlushSerialRequestResend();
void ClearToSend();
void analogWrite_check(uint8_t check_pin, int val);
void showString (PGM_P s);
void manage_inactivity(byte debug);
void get_command();
void get_coordinates();
```

```
void prepare_move();
void prepare_arc_move(char isclockwise);
FORCE_INLINE void process_commands();
#ifdef USE_ARC_FUNCTION
  FORCE_INLINE void get_arc_coordinates();
#endif
void kill(byte debug);
void check_axes_activity();
void plan_init();
void st_init();
void tp_init();
void plan_buffer_line(float x, float y, float z, float e, float feed_rate);
void plan_set_position(float x, float y, float z, float e);
void st_wake_up();
void st_synchronize();
void st_set_position(const long &x, const long &y, const long &z, const long &e);
void check_buffer_while_arc();
#ifdef SDSUPPORT
void print_disk_info(void);
#endif
#if (MINIMUM_FAN_START_SPEED > 0)
void manage_fan_start_speed(void);
#endif
#ifdef DEBUG
void log_message(char*   message);
void log_bool(char* message, bool value);
void log_int(char* message, int value);
void log_long(char* message, long value);
void log_float(char* message, float value);
void log_uint(char* message, unsigned int value);
void log_ulong(char* message, unsigned long value);
#endif
```

Sprinter

```
char axis_codes[NUM_AXIS] = {'X', 'Y', 'Z', 'E'};
float axis_steps_per_unit[4] = _AXIS_STEP_PER_UNIT;
float max_feedrate[4] = _MAX_FEEDRATE;
float homing_feedrate[] = _HOMING_FEEDRATE;
bool axis_relative_modes[] = _AXIS_RELATIVE_MODES;
float move_acceleration = _ACCELERATION;
float retract_acceleration = _RETRACT_ACCELERATION;
float max_xy_jerk = _MAX_XY_JERK;
float max_z_jerk = _MAX_Z_JERK;
float max_e_jerk = _MAX_E_JERK;
unsigned long min_seg_time = _MIN_SEG_TIME;
#ifdef PIDTEMP
 unsigned int PID_Kp = PID_PGAIN, PID_Ki = PID_IGAIN, PID_Kd = PID_DGAIN;
#endif
long  max_acceleration_units_per_sq_second[4] =
_MAX_ACCELERATION_UNITS_PER_SQ_SECOND;
float mintravelfeedrate = DEFAULT_MINTRAVELFEEDRATE;
float minimumfeedrate = DEFAULT_MINIMUMFEEDRATE;
unsigned long axis_steps_per_sqr_second[NUM_AXIS];
unsigned long plateau_steps;
volatile int feedmultiply=100; //100->original / 200 -> Factor 2 / 50 -> Factor 0.5
int saved_feedmultiply;
volatile bool feedmultiplychanged=false;
volatile int extrudemultiply=100; //100->1 200->2
float destination[NUM_AXIS] = {0.0, 0.0, 0.0, 0.0};
float current_position[NUM_AXIS] = {0.0, 0.0, 0.0, 0.0};
float add_homing[3]={0,0,0};
static unsigned short virtual_steps_x = 0;
static unsigned short virtual_steps_y = 0;
static unsigned short virtual_steps_z = 0;
bool home_all_axis = true;
int feedrate = 1500, next_feedrate, saved_feedrate;
long gcode_N, gcode_LastN;
bool relative_mode = false;
bool is_homing = false;
#ifdef USE_ARC_FUNCTION
```

```
float offset[3] = {0.0, 0.0, 0.0};
#endif
#ifdef STEP_DELAY_RATIO
  long long_step_delay_ratio = STEP_DELAY_RATIO * 100;
#endif
#ifdef RAPID_OSCILLATION_REDUCTION
  float cumm_wait_time_in_dir[NUM_AXIS]={0.0,0.0,0.0,0.0};
  bool prev_move_direction[NUM_AXIS]={1,1,1,1};
  float osc_wait_remainder = 0.0;
#endif
#define MAX_CMD_SIZE 96
#define BUFSIZE 6
char cmdbuffer[BUFSIZE][MAX_CMD_SIZE];
bool fromsd[BUFSIZE];
unsigned char bufindr = 0;
unsigned char bufindw = 0;
unsigned char buflen = 0;
char serial_char;
int serial_count = 0;
boolean comment_mode = false;
char *strchr_pointer;
int hotendtC = 0, bedtempC = 0;
unsigned long previous_millis_cmd = 0;
unsigned long max_inactive_time = 0;
unsigned long stepper_inactive_time = 0;
unsigned char manage_monitor = 255;
void analogWrite_check(uint8_t check_pin, int val)
{
  #if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
    if((check_pin != 11) && (check_pin != 12) && (check_pin != 13))
    {
        analogWrite(check_pin, val);
    }
  #endif
void setup()
{
  Serial.begin(BAUDRATE);
  showString(PSTR("Sprinter\r\n"));
```

```
showString(PSTR(_VERSION_TEXT));
showString(PSTR("\r\n"));
showString(PSTR("start\r\n"));
for(int i = 0; i < BUFSIZE; i++)
{
   fromsd[i] = false;
}
#if X_DIR_PIN > -1
  SET_OUTPUT(X_DIR_PIN);
#endif
#if Y_DIR_PIN > -1
  SET_OUTPUT(Y_DIR_PIN);
#endif
#if Z_DIR_PIN > -1
  SET_OUTPUT(Z_DIR_PIN);
#endif
#if E_DIR_PIN > -1
  SET_OUTPUT(E_DIR_PIN);
#endif
#if (X_ENABLE_PIN > -1)
  SET_OUTPUT(X_ENABLE_PIN);
if(!X_ENABLE_ON) WRITE(X_ENABLE_PIN,HIGH);
#endif
#if (Y_ENABLE_PIN > -1)
  SET_OUTPUT(Y_ENABLE_PIN);
if(!Y_ENABLE_ON) WRITE(Y_ENABLE_PIN,HIGH);
#endif
#if (Z_ENABLE_PIN > -1)
  SET_OUTPUT(Z_ENABLE_PIN);
if(!Z_ENABLE_ON) WRITE(Z_ENABLE_PIN,HIGH);
#endif
#if (E_ENABLE_PIN > -1)
  SET_OUTPUT(E_ENABLE_PIN);
if(!E_ENABLE_ON) WRITE(E_ENABLE_PIN,HIGH);
#endif
#ifdef CONTROLLERFAN_PIN
  SET_OUTPUT(CONTROLLERFAN_PIN);
#endif
```

```
#ifdef EXTRUDERFAN_PIN
  SET_OUTPUT(EXTRUDERFAN_PIN);
#endif
#ifdef ENDSTOPPULLUPS
#if X_MIN_PIN > -1
  SET_INPUT(X_MIN_PIN);
  WRITE(X_MIN_PIN,HIGH);
#endif
#if X_MAX_PIN > -1
  SET_INPUT(X_MAX_PIN);
  WRITE(X_MAX_PIN,HIGH);
#endif
#if Y_MIN_PIN > -1
  SET_INPUT(Y_MIN_PIN);
  WRITE(Y_MIN_PIN,HIGH);
#endif
#if Y_MAX_PIN > -1
  SET_INPUT(Y_MAX_PIN);
  WRITE(Y_MAX_PIN,HIGH);
#endif
#if Z_MIN_PIN > -1
  SET_INPUT(Z_MIN_PIN);
  WRITE(Z_MIN_PIN,HIGH);
#endif
#if Z_MAX_PIN > -1
  SET_INPUT(Z_MAX_PIN);
  WRITE(Z_MAX_PIN,HIGH);
#endif
#else
#if X_MIN_PIN > -1
  SET_INPUT(X_MIN_PIN);
#endif
#if X_MAX_PIN > -1
  SET_INPUT(X_MAX_PIN);
#endif
#if Y_MIN_PIN > -1
  SET_INPUT(Y_MIN_PIN);
#endif
```

```
  #if Y_MAX_PIN > -1
    SET_INPUT(Y_MAX_PIN);
  #endif
  #if Z_MIN_PIN > -1
    SET_INPUT(Z_MIN_PIN);
  #endif
  #if Z_MAX_PIN > -1
    SET_INPUT(Z_MAX_PIN);
  #endif
  #endif
  #if (X_STEP_PIN > -1)
    SET_OUTPUT(X_STEP_PIN);
  #endif
  #if (Y_STEP_PIN > -1)
    SET_OUTPUT(Y_STEP_PIN);
  #endif
  #if (Z_STEP_PIN > -1)
    SET_OUTPUT(Z_STEP_PIN);
  #endif
  #if (E_STEP_PIN > -1)
    SET_OUTPUT(E_STEP_PIN);
  #endif
#ifdef HEATER_USES_MAX6675
  SET_OUTPUT(SCK_PIN);
  WRITE(SCK_PIN,0);
  SET_OUTPUT(MOSI_PIN);
  WRITE(MOSI_PIN,1);
  SET_INPUT(MISO_PIN);
  WRITE(MISO_PIN,1);
  SET_OUTPUT(MAX6675_SS);
  WRITE(MAX6675_SS,1);
#endif
  showString(PSTR("Planner Init\r\n"));
  plan_init();
  showString(PSTR("Stepper Timer init\r\n"));
  st_init();
  showString(PSTR("Free Ram: "));
  Serial.println(FreeRam1());
```

```
  showString(PSTR("Plan Buffer Size:"));
  Serial.print((int)sizeof(block_t)*BLOCK_BUFFER_SIZE);
  showString(PSTR(" / "));
  Serial.println(BLOCK_BUFFER_SIZE);
  for(int8_t i=0; i < NUM_AXIS; i++)
  {
    axis_steps_per_sqr_second[i] = max_acceleration_units_per_sq_second[i] *
axis_steps_per_unit[i];
  }
}
void loop()
{
  if(buflen < (BUFSIZE-1))
    get_command();
  if(buflen)
  {
#ifdef SDSUPPORT
    if(savetosd)
    {
      if(strstr(cmdbuffer[bufindr],"M29") == NULL)
      {
        write_command(cmdbuffer[bufindr]);
        showString(PSTR("ok\r\n"));
      }
      else
      {
        file.sync();
        file.close();
        savetosd = false;
        showString(PSTR("Done saving file.\r\n"));
      }
    }
    else
    {
      process_commands();
    }
#else
    process_commands();
```

```
#endif
    buflen = (buflen-1);
    bufindr++;
    if(bufindr == BUFSIZE) bufindr = 0;
  }
  manage_heater();
  manage_inactivity(1);
  #if (MINIMUM_FAN_START_SPEED > 0)
    manage_fan_start_speed();
  #endif
}
```

Pins.h

```
#if MOTHERBOARD == 33
#define MOTHERBOARD 3
#define RAMPS_V_1_3
#endif
#if MOTHERBOARD == 3
#define KNOWN_BOARD 1
//////////////////FIX THIS///////////////
#ifndef __AVR_ATmega1280__
 #ifndef __AVR_ATmega2560__
 #error Oops!  Make sure you have 'Arduino Mega' selected from the 'Tools -> Boards' menu.
 #endif
#endif
// uncomment one of the following lines for RAMPS v1.3 or v1.0, comment both for v1.2 or 1.1
// #define RAMPS_V_1_3
// #define RAMPS_V_1_0
#ifdef RAMPS_V_1_3
#define X_STEP_PIN         54
#define X_DIR_PIN          55
#define X_ENABLE_PIN       38
#define X_MIN_PIN          3
#define X_MAX_PIN          -1  //2 //Max endstops default to disabled "-1", set to commented
value to enable.
#define Y_STEP_PIN         60
#define Y_DIR_PIN          61
#define Y_ENABLE_PIN       56
#define Y_MIN_PIN          14
#define Y_MAX_PIN          -1  //15
#define Z_STEP_PIN         46
#define Z_DIR_PIN          48
#define Z_ENABLE_PIN       62
#define Z_MIN_PIN          18
#define Z_MAX_PIN          -1  //19
#define E_STEP_PIN         26
#define E_DIR_PIN          28
#define E_ENABLE_PIN       24
#define E_1_STEP_PIN       36
```

```
#define E_1_DIR_PIN        34
#define E_1_ENABLE_PIN     30
#define SDPOWER            -1
#define SDSS               53
#define LED_PIN            13
#define FAN_PIN            9
#define PS_ON_PIN          12
#define KILL_PIN           -1
#define ALARM_PIN          -1
#define HEATER_0_PIN       10
#define HEATER_1_PIN       8
#define TEMP_0_PIN         13   // ANALOG NUMBERING
#define TEMP_1_PIN         14   // ANALOG NUMBERING
#define TEMP_2_PIN         15   // ANALOG NUMBERING
#else // RAMPS_V_1_1 or RAMPS_V_1_2 as default
#define X_STEP_PIN         26
#define X_DIR_PIN          28
#define X_ENABLE_PIN       24
#define X_MIN_PIN          3
#define X_MAX_PIN          -1   //2
#define Y_STEP_PIN         38
#define Y_DIR_PIN          40
#define Y_ENABLE_PIN       36
#define Y_MIN_PIN          16
#define Y_MAX_PIN          -1   //17
#define Z_STEP_PIN         44
#define Z_DIR_PIN          46
#define Z_ENABLE_PIN       42
#define Z_MIN_PIN          18
#define Z_MAX_PIN          -1   //19
#define E_STEP_PIN         32
#define E_DIR_PIN          34
#define E_ENABLE_PIN       30
#define SDPOWER            48
#define SDSS               53
#define LED_PIN            13
#define PS_ON_PIN          -1
#define KILL_PIN           -1
```

```
#define ALARM_PIN          -1
#ifdef RAMPS_V_1_0 // RAMPS_V_1_0
  #define HEATER_0_PIN     12    // RAMPS 1.0
  #define HEATER_1_PIN     -1    // RAMPS 1.0
  #define FAN_PIN          11    // RAMPS 1.0
#else // RAMPS_V_1_1 or RAMPS_V_1_2
  #define HEATER_0_PIN     10    // RAMPS 1.1
  #define HEATER_1_PIN      8    // RAMPS 1.1
  #define FAN_PIN           9    // RAMPS 1.1
#endif
#define TEMP_0_PIN          2    // MUST USE ANALOG INPUT NUMBERING NOT DIGITAL OUTPUT
NUMBERING!!!!!!!!!
#define TEMP_1_PIN          1    // MUST USE ANALOG INPUT NUMBERING NOT DIGITAL OUTPUT
NUMBERING!!!!!!!!!
#endif
// SPI for Max6675 Thermocouple
#ifndef SDSUPPORT
// these pins are defined in the SD library if building with SD support
  #define SCK_PIN          52
  #define MISO_PIN         50
  #define MOSI_PIN         51
  #define MAX6675_SS       53
#else
  #define MAX6675_SS       49
#endif
#endif
```

ภาคผนวกข

เอกสารคู่มืออุปกรณ์อิเล็กทรอนิกส์

ข.1 เอกสารคู่มือการใช้งานPololu 8-35V 2A Single Bipolar Stepper Motor Driver Board A4988

Allegro

A4988

DMOS Microstepping Driver with Translator and Overcurrent Protection

Features and Benefits

- Low $R_{DS(on)}$ outputs
- Automatic current decay mode detection/selection
- Mixed and Slow current decay modes
- Synchronous rectification for low power dissipation
- Internal UVLO
- Crossover-current protection
- 3.3 and 5 V compatible logic supply
- Thermal shutdown circuitry
- Short-to-ground protection
- Shorted load protection
- Five selectable step modes: full, ½, ¼, ⅛, and 1/16

Package:

28-contact QFN
with exposed thermal pad
5 mm × 5 mm × 0.90 mm
(ET package)

Approximate size

Description

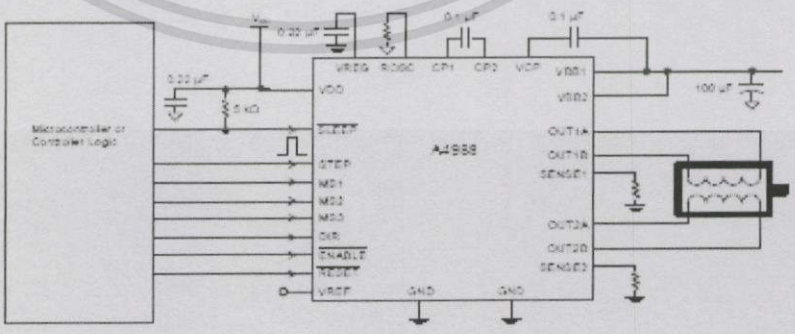
The A4988 is a complete microstepping motor driver with built-in translator for easy operation. It is designed to operate bipolar stepper motors in full-, half-, quarter-, eighth-, and sixteenth-step modes, with an output drive capacity of up to 35 V and ±2 A. The A4988 includes a fixed off-time current regulator which has the ability to operate in Slow or Mixed decay modes.

The translator is the key to the easy implementation of the A4988. Simply inputting one pulse on the STEP input drives the motor one microstep. There are no phase sequence tables, high frequency control lines, or complex interfaces to program. The A4988 interface is an ideal fit for applications where a complex microprocessor is unavailable or is overburdened.

During stepping operation, the chopping control in the A4988 automatically selects the current decay mode, Slow or Mixed. In Mixed decay mode, the device is set initially to a fast decay for a proportion of the fixed off-time, then to a slow decay for the remainder of the off-time. Mixed decay current control results in reduced audible motor noise, increased step accuracy, and reduced power dissipation.

* continued on the next page ...

Typical Application Diagram



Description (continued)

Internal synchronous rectification control circuitry is provided to improve power dissipation during PWM operation. Internal circuit protection includes: thermal shutdown with hysteresis, undervoltage lockout (UVLO), and crossover-current protection. Special power-on sequencing is not required.

The A4988 is supplied in a surface mount QFN package (ES), 5 mm × 5 mm, with a nominal overall package height of 0.90 mm and an exposed pad for enhanced thermal dissipation. It is lead (Pb) free (suffix –T), with 100% matte tin plated leadframes.

Selection Guide

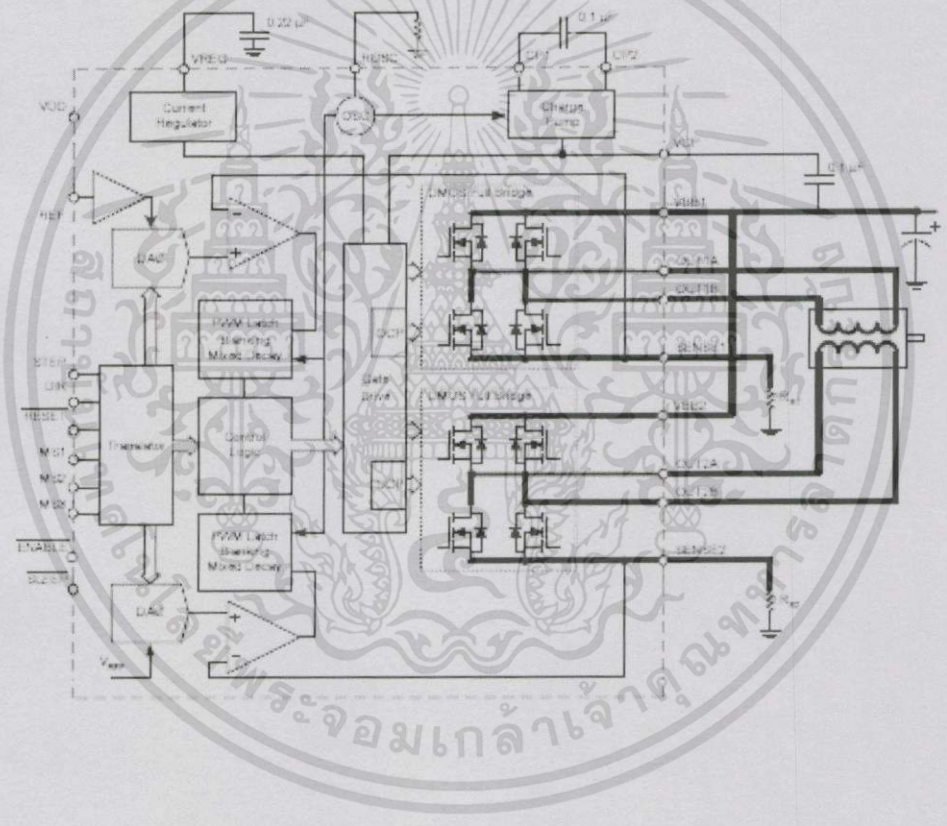
Part Number	Package	Packing
A4988SETTR-T	28-contact QFN with exposed thermal pad	1500 pieces per 7-in. reel

Absolute Maximum Ratings

Characteristic	Symbol	Notes	Rating	Units
Load Supply Voltage	V_{BB}		35	V
Output Current	I_{OUT}		±2	A
Logic Input Voltage	V_{IN}		−0.3 to 5.5	V
Logic Supply Voltage	V_{DD}		−0.3 to 5.5	V
VBBx to OUTx			35	V
Sense Voltage	V_{SENSE}		0.5	V
Reference Voltage	V_{REF}		5.5	V
Operating Ambient Temperature	T_A	Range S	−20 to 85	°C
Maximum Junction	T_J (max)		150	°C
Storage Temperature	T_{stg}		−55 to 150	°C

Allegro MicroSystems, Inc.
115 Northeast Cutoff
Worcester, Massachusetts 01615-0036 U.S.A.
1.508.853.5000; www.allegromicro.com

Functional Block Diagram



ELECTRICAL CHARACTERISTICS[1] at T_A = 25°C, V_{BB} = 35 V (unless otherwise noted)

Characteristics	Symbol	Test Conditions	Min.	Typ.[2]	Max.	Units
Output Drivers						
Load Supply Voltage Range	V_{BB}	Operating	8	–	35	V
Logic Supply Voltage Range	V_{DD}	Operating	3.0	–	5.5	V
Output On Resistance	$R_{DS(ON)}$	Source Driver, I_{OUT} = –1.5 A	–	320	430	mΩ
		Sink Driver, I_{OUT} = 1.5 A	–	320	430	mΩ
Body Diode Forward Voltage	V_F	Source Diode, I_F = –1.5 A	–	–	1.2	V
		Sink Diode, I_F = 1.5 A	–	–	1.2	V
Motor Supply Current	I_{BB}	f_{PWM} < 50 kHz	–	–	4	mA
		Operating, outputs disabled	–	–	2	mA
Logic Supply Current	I_{DD}	f_{PWM} < 50 kHz	–	–	8	mA
		Outputs off	–	–	5	mA
Control Logic						
Logic Input Voltage	$V_{IN(1)}$		V_{DD}×0.7	–	–	V
	$V_{IN(0)}$		–	–	V_{DD}×0.3	V
Logic Input Current	$I_{IN(1)}$	V_{IN} = V_{DD}×0.7	–20	<1.0	20	µA
	$I_{IN(0)}$	V_{IN} = V_{DD}×0.3	–20	<1.0	20	µA
Microstep Select	R_{MS1}	MS1 pin	–	100	–	kΩ
	R_{MS2}	MS2 pin	–	50	–	kΩ
	R_{MS3}	MS3 pin	–	100	–	kΩ
Logic Input Hysteresis	$V_{IN(HYS)}$	As a % of V_{DD}	5	11	19	%
Blank Time	t_{BLANK}		0.7	1	1.3	µs
Fixed Off-Time	t_{OFF}	OSC = VDD or GND	20	30	40	µs
		R_{OSC} = 25 kΩ	23	30	37	µs
Reference Input Voltage Range	V_{REF}		0	–	4	V
Reference Input Current	I_{REF}		–3	0	3	µA
Current Trip-Level Error[3]	e_I	V_{REF} = 2 V, %$I_{TripMAX}$ = 38.27%	–	–	±15	%
		V_{REF} = 2 V, %$I_{TripMAX}$ = 70.71%	–	–	±5	%
		V_{REF} = 2 V, %$I_{TripMAX}$ = 100.00%	–	–	±5	%
Crossover Dead Time	t_{DT}		100	475	800	ns
Protection						
Overcurrent Protection Threshold	I_{OCPST}		2.1	–	–	A
Thermal Shutdown Temperature	T_{TSD}		–	165	–	°C
Thermal Shutdown Hysteresis	T_{TSDHYS}		–	15	–	°C
VDD Undervoltage Lockout	$V_{UVLO(VDD)}$	V_{DD} rising	2.7	2.8	2.9	V
VDD Undervoltage Hysteresis	$V_{UVLO(VDDHYS)}$		–	90	–	mV

[1]For input and output current specifications, negative current is defined as coming out of (sourcing) the specified device pin.

[2]Typical data are for initial design estimations only, and assume optimum manufacturing and application conditions. Performance may vary for individual units, within the specified maximum and minimum limits.

[3]V_{ERR} = [(V_{REF}/8) – V_{SENSE}] / (V_{REF}/8).

ข.2 เอกสารคู่มือการใช้งานET-EASY MEGA1280 (ARDUINO MEGA)

Overview

The ATmega640/1280/1281/2560/2561 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega640/1280/1281/2560/2561 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 135 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-Chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 64K/128K/256K Bytes of In-System Self-Programmable Flash
  - 4K Bytes EEPROM
  - 8K Bytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/ 100 years at 25°C
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
  - Endurance: Up to 64K Bytes Optional External Memory Space
- JTAG (IEEE std. 1149.1 compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Six/Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits (ATmega1281/2561, ATmega640/1280/2560)
  - Output Compare Modulator
  - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
  - Two/Four Programmable Serial USART (ATmega1281/2561,ATmega640/1280/2560)
  - Master/Slave SPI Serial Interface
  - Byte Oriented 2-wire Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 54/86 Programmable I/O Lines (ATmega1281/2561, ATmega640/1280/2560)
  - 64-pad QFN/MLF, 64-lead TQFP (ATmega1281/2561)
  - 100-lead TQFP, 100-ball CBGA (ATmega640/1280/2560)
  - RoHS/Fully Green
- Temperature Range:
  - -40°C to 85°C Industrial
- Ultra-Low Power Consumption
  - Active Mode: 1 MHz, 1.8V: 500 µA
  - Power-down Mode: 0.1 µA at 1.8V
- Speed Grade:
  - ATmega640V/ATmega1280V/ATmega1281V:
    0 - 4 MHz @ 1.8 - 5.5V, 0 - 8 MHz @ 2.7 - 5.5V
  - ATmega2560V/ATmega2561V:
    0 - 2 MHz @ 1.8 - 5.5V, 0 - 8 MHz @ 2.7 - 5.5V
  - ATmega640/ATmega1280/ATmega1281:
    0 - 8 MHz @ 2.7 - 5.5V, 0 - 16 MHz @ 4.5 - 5.5V
  - ATmega2560/ATmega2561:
    0 - 16 MHz @ 4.5 - 5.5V

ATMEL

8-bit AVR® Microcontroller with 64K/128K/256K Bytes In-System Programmable Flash

ATmega640/V
ATmega1280/V
ATmega1281/V
ATmega2560/V
ATmega2561/V

Preliminary

resistors are activated. The Port H pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port H also serves the functions of various special features of the ATmega640/1280/2560 as listed on page 92.

3.3.11 Port J (PJ7..PJ0)

Port J is a 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port J output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port J pins that are externally pulled low will source current if the pull-up resistors are activated. The Port J pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port J also serves the functions of various special features of the ATmega640/1280/2560 as listed on page 95.

3.3.12 Port K (PK7..PK0)

Port K serves as analog inputs to the A/D converter.

Port K is a 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port K output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port K pins that are externally pulled low will source current if the pull-up resistors are activated. The Port K pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port K also serves the functions of various special features of the ATmega640/1280/2560 as listed on page 98.

3.3.13 Port L (PL7..PL0)

Port L is a 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port L output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port L pins that are externally pulled low will source current if the pull-up resistors are activated. The Port L pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port L also serves the functions of various special features of the ATmega640/1280/2560 as listed on page 99.

3.3.14 $\overline{\text{RESET}}$

Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in "System and Reset Characteristics" on page 375. Shorter pulses are not guaranteed to generate a reset.

3.3.15 XTAL1

Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

3.3.16 XTAL2

Output from the inverting Oscillator amplifier.

3.3.17 AVCC

AVCC is the supply voltage pin for Port F and the A/D Converter. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.

3.3.18 AREF

This is the analog reference pin for the A/D Converter.

4. Resources

A comprehensive set of development tools and application notes, and datasheets are available for download on http://www.atmel.com/avr.

5. About Code Examples

This documentation contains simple code examples that briefly show how to use various parts of the device. Be aware that not all C compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Please confirm with the C compiler documentation for more details.

These code examples assume that the part specific header file is included before compilation. For I/O registers located in extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" instructions must be replaced with instructions that allow access to extended I/O. Typically "LDS" and "STS" combined with "SBRS", "SBRC", "SBR", and "CBR".

6. Data Retention

Reliability Qualification results show that the projected data retention failure rate is much less than 1 PPM over 20 years at 85°C or 100 years at 25°C.

ภาคผนวกค

โปสเตอร์