เครื่องร่างภาพขนาดเล็กควบคุมด้วยพิกัดตัวเลข
Mini Sketching CNC

อิทธิกร จุฑามณี
Itthikorn Juthamanee
อุกฤษฎ์ ตันฑเทอดธรรม
Aukit Tanthathoedtham

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2559

เครื่องร่างภาพขนาดเล็กควบคุมด้วยพิกัดตัวเลข

Mini Sketching CNC

อิทธิกร จุฑามณี
Itthikorn Juthamanee
อุกฤษฎ์ ตันฑเทอดธรรม
Aukit Tanthathoedtham

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2559

เครื่องร่างภาพขนาดเล็กควบคุมด้วยพิกัดตัวเลข

Mini Sketching CNC

โดย

อิทธิกร จุฑามณี

อุกฤษฎ์ ตันฑเทอดธรรม

อาจารย์ที่ปรึกษา

ผศ.ดร.แสงระวี บัวแก้ว

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2559

ปริญญานิพนธ์ ปีการศึกษา 2559

ภาควิชา วิศวกรรมอิเล็กทรอนิกส์

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องร่างภาพขนาดเล็กควบคุมด้วยพิกัดตัวเลข

Mini Sketching CNC

ผู้จัดทำ นาย อิทธิกร จุฑามณี รหัสประจำตัว 56011465

นาย อุกฤษฎ์ ตันฑเทอดธรรม รหัสประจำตัว 56011470

รายงานนี้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

(ผศ. ดร. แสงระวี บัวแก้ว)

อาจารย์ที่ปรึกษา

หัวข้อโครงงาน	เครื่องร่างภาพขนาดเล็กควบคุมด้วยพิกัดตัวเลข			
นักศึกษา	นาย อิทธิกร	จุฑามณี	รหัสประจำตัว	56011465
	นาย อุกฤษฎ์	ตันฑเทอดธรรม	รหัสประจำตัว	56011470
ปริญญา	วิศวกรรมศาสตร์บัณฑิต			
ภาควิชา	วิศวกรรมอิเล็กทรอนิกส์			
ปีการศึกษา	2559			
อาจารย์ที่ปรึกษาโครงการ	ผศ.ดร. แสงระวี บัวแก้ว			

บทคัดย่อ

โครงงานนี้นำเสนอการออกแบบและสร้างเครื่องร่างภาพขนาดเล็กที่ควบคุมได้ด้วยพิกัดตัวเลข เครื่องร่างภาพนี้มีการทำงานภายใต้ขอบเขตของชิ้นงานไม่เกิน 10x15 เซนติเมตร การทำงานของเครื่องร่างภาพอาศัยหลักการป้อนพิกัดเชิงตัวเลข หรือ จีโค้ด ให้กับไมโครคอนโทรลเลอร์ ซึ่งการแปลงภาพ หรือตัวอักษร ให้อยู่ในรูปพิกัดตัวเลข ทำได้ โดย อาศัยโปรแกรม Inkscape จากนั้นพิกัดตัวเลขจะส่งไปยังไมโครคอนโทรลเลอร์โดย อาศัยโปรแกรม Grbl Controller ซึ่งจะทำให้ เครื่องร่างภาพทำการวาดภาพหรือ วาดตัวอักษรตามที่จีโค้ดเป็นตัวกำหนด ในส่วนของเครื่องร่างภาพ ประกอบด้วย 3 ส่วน ที่สำคัญคือ ส่วนแนวแกน X ส่วนแนวแกน Y ส่วนแนวแกน Z โดยในส่วนของ แนวแกน X และ แนวแกน Y จะใช้สเต็ปเปอร์มอเตอร์ในการบังคับ จำนวน 2 ชุด และในส่วนแนวแกน Z จะใช้ เซอร์โวมอเตอร์ในการควบคุม เครื่องร่างภาพที่สร้างขึ้นมีประโยชน์สามารถนำไป วาดรูป เขียนตัวอักษร หรือใช้วาดลาย PCB ได้

Project Title	Mini Sketching CNC		
Student Name	Mr. Itthikorn	Juthamanee	56011465
	Mr. Aukit	Tanthathoedtham	56011470
Degree	Bachelor of Engineering		
Major	Electronics Engineering		
Academic Year	2559		
Advisor	Asst. Prof. Dr. Sangrawee Buakaew		

Abstract

This project represents mini sketching computer numerical control (CNC) machine. The machine could work properly with 10x15 centimeters sheets. Working processes require G-codes which could be generated from converting picture files by Inkscape program as microcontroller input and to connect to the microcontroller board, Grbl program is needed. The G-code data contain directions and commands which specify the machine's drawing. The machine has 3 directions working separately, X direction and Y direction are controlled by 2 stepper motors and Z direction is controlled by servo motor. Mini sketching CNC could be used in many occasions such as drawing pictures, drawing letters and drawing PCB prints.

กิตติกรรมประกาศ

โครงงานชิ้นนี้เสร็จสมบูรณ์ได้ เนื่องจากได้รับการสนับสนุนและให้คำปรึกษา ความรู้ คำแนะนำจากอาจารย์ที่ปรึกษา ผศ.ดร.แสงระวี บัวแก้ว ที่ให้การสนับสนุนด้านอุปกรณ์ และสถานที่ในการทำงาน รวมถึงการดูแลอย่างดีในด้านต่างๆ นอกจากนี้ยังได้รับการสนับสนุนจากอาจารย์ทุกท่านในภาควิชาอิเล็กทรอนิกส์ และรุ่นพี่ที่ให้ความช่วยเหลือทั้งปริญญาตรีและปริญญาโท รวมถึงชุมนุม Automotive ที่ให้การสนับสนุนในการทำชิ้นงาน และขอบคุณคุณพ่อ คุณแม่ ที่ให้การสนับสนุนเงินทุนในการทำโครงงานชิ้นนี้ จึงใคร่ขอขอบพระคุณผู้มีอุปการคุณทุกท่านมา ณ ที่นี้

อิทธิกร จุฑามณี

อุกฤษฎ์ ตันฑเทอดธรรม

สารบัญ

หน้า

สารบัญตาราง

สารบัญรูป

รูปที่ หน้า

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา (STATEMENT AND SIGNIFICANCE OF THE PROBLEMS)

ในปัจจุบันนั้นเครื่องวาดภาพที่ควบคุมด้วยพิกัดตัวเลขนั้นได้มีการประยุกต์ออกไปในหลายๆ รูปแบบไม่ว่าจะใช้ในการตัด หรือ กลึงเหล็กภายในโรงงานต่างๆ หรือแบบสำเร็จรูปก็มีขายมากมายแต่ด้วยเนื่องจากราคาที่สูง เพื่อที่จะแก้ไขปัญหาในส่วนนี้ จึงทำการศึกษาและทำการสร้างเครื่องวาดภาพดังกล่าวขึ้นมาเองในขนาดเล็กเพื่อที่จะทำความเข้าใจถึงการทำงานต่างๆของตัวอุปกรณ์ให้มากยิ่งขึ้น

1.2 จุดมุ่งหมายและวัตถุประสงค์ของการศึกษา (GOAL AND OBJECTIVE)

1.2.1 สามารถนำความรู้ที่ได้จากการค้นคว้านำมาสร้างชิ้นงานและสามารถนำไปใช้ได้จริง

1.2.2 ศึกษาความรู้เพิ่มเติมในเรื่องต่างๆที่เกี่ยวกับชิ้นงานของเรา เช่น การควบคุมมอเตอร์ และบอร์ดไมโครคอนโทลเลอร์

1.2.3 สามารถสร้างและพัฒนาโปรแกรมบนคอมพิวเตอร์ให้มีประสิทธิภาพ และเหมาะสมกับงานที่ต้องการได้

1.2.4 สามารถนำความรู้ที่ได้จากการศึกษาและสร้างชิ้นงานมาประยุกต์ใช้กับงานอื่นๆ หรือชิ้นงานที่ต้องการส่วนประกอบที่คล้ายคลึงกันได้

1.3 สมมุติฐานของการศึกษา (HYPOTHESIS TO BE TESTED)

หาจุดบกพร่องที่เกิดขึ้นและสามารถแก้ไขจุดบกพร่องเหล่านั้นได้

1.4 ขอบเขตการศึกษา (SCOPE OR LIMITATION OF THE STUDY)

1.4.1 สามารถสร้างเครื่องวาดภาพขนาดเล็กที่ควบคุมด้วยพิกัดตัวเลขได้

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 สามารถนำโครงงานชิ้นนี้ไปใช้งานจริงได้ในราคาที่ไม่แพงเกินไป

1.5.2 ได้รับความรู้เรื่องทฤษฎีการทำงานของมอเตอร์และการควบคุม

1.5.3 สามารถเขียนและพัฒนาโปรแกรมบนคอมพิวเตอร์ได้

บทที่ 2
เครื่องวาดภาพขนาดเล็กควบคุมด้วยพิกัดตัวเลข

2.1 Cnc machine (Computerized Numerical control)

2.1.1 ความเป็นมาของเครื่องซีเอ็นซี

ในอดีตการทำชิ้นงานที่มีความละเอียดต้องอาศัยความชำนาญของช่างผู้ควบคุม ในการตัดเฉือนชิ้นงานโดยใช้เครื่องจักรทั่วไป โดยการทำงานลักษณะนี้ต้องอาศัยการควบคุมเฝ้าดูตำแหน่งอยู่ตลอดเวลาสามารถทำชิ้นงานได้เฉพาะอย่างเท่านั้น ต่อมาเริ่มมีการพัฒนาเครื่องจักรที่ใช้กระแสไฟฟ้าในการควบคุมมอเตอร์มาใช้งาน

เริ่มแรกเครื่องซีเอ็นซีถูกเรียกว่าเครื่องจักรที่ใช้ควบคุมด้วยระบบตัวเลข (Numerical control) ถูกคิดขึ้นที่สถาบัน M.I.T. ในปีค.ศ. 1948 โดยมีจุดประสงค์ในขณะนั้นเพื่อมาผลิตชิ้นส่วนเครื่องบิน และต่อมาในภายหลังเมื่อคอมพิวเตอร์ได้เข้ามามีบทบาทมากขึ้น จึงได้นำคอมพิวเตอร์มาใช้ร่วมกับระบบเอ็นซีจึงกลายมาเป็นซีเอ็นซี (Computerized Numerical control)

2.1.1.1 ความแตกต่างระหว่างระบบเอ็นซี และ ระบบซีเอ็นซี

1. การแสดงรูปแบบจำลองการทำงานตามโปรแกรมที่ป้อนในระบบจอภาพ
2. การแก้ไขและลบโปรแกรมสามารถทำได้ที่เครื่องจักรโดยตรง
3. การเก็บความจำภายนอก
4. การคำนวณต่างๆโดยใช้โปรแกรม

2.1.1.2 ข้อดีและข้อเสียของเครื่องจักรกลเอ็นซีและซีเอ็นซี

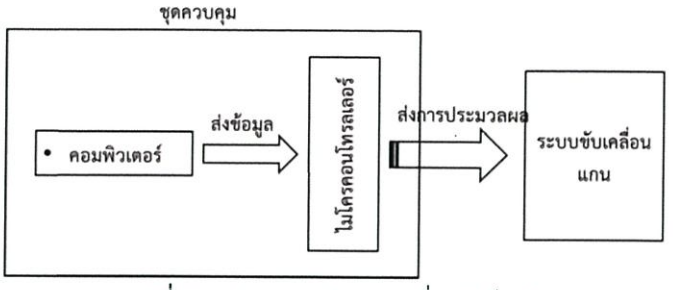
ข้อดี

1. มีความยืดหยุ่นการทำงานสูงในด้านการแก้ไขหรือเปลี่ยนแปลงเฉพาะโปรแกรม
2. ความเที่ยงตรงในการทำงานของเครื่องจักร
3. เวลาในการผลิตสั้น
4. ได้ชิ้นงานที่มีความซับซ้อนมาก

ข้อเสีย

1. ราคาสูง
2. การบำรุงรักษามีความซับซ้อนมาก
3. ต้องใช้การเขียนโปรแกรมที่มีความเฉพาะสูง
4. ชิ้นส่วนอะไหล่ที่ใช้ในการซ่อมบำรุงหายาก

2.1.2 หลักการทำงานของเครื่อง

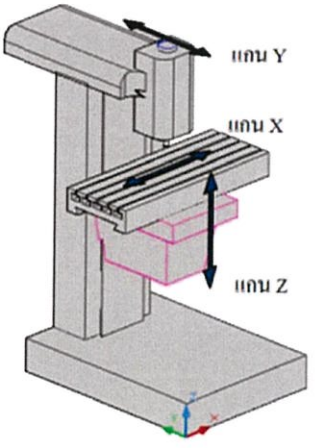


รูปที่ 2.1 ส่วนการทำงานเครื่องซีเอ็นซี

เครื่องซีเอ็นซีสามารถแบ่งการทำงานได้เป็น 2 ส่วนใหญ่

2.1.2.1 ชุดควบคุม ในเครื่องจักรซีเอ็นซีจะมีชุดควบคุมที่ใช้อ่านและเขียนโปรแกรมผ่านคอมพิวเตอร์ก่อนจะส่งข้อมูลให้ไมโครคอนโทรลเลอร์ประมวลผล เมื่อไมโครคอนโทรลเลอร์ได้ประมวลผลแล้วก็จะส่งการประมวลผลไปยังระบบขับเคลื่อนแกน

2.1.2.2 ระบบขับเคลื่อนแกน ประกอบไปด้วยวงจรไดร์ฟสเต็ปเปอร์มอเตอร์ เนื่องจากสัญญาณพัลส์ที่ออกมาจากไมโครคอนโทรลเลอร์ไม่เพียงพอในการทำให้มอเตอร์ทำงานได้ จึงต้องมีการใช้วงจรไดร์ฟมาช่วยในการทำให้สเต็ปเปอร์มอเตอร์ทำงาน โดยเนื้อหาเกี่ยวกับมอเตอร์ที่ใช้งานจะกล่าวถึงในหัวข้อถัดไป เครื่องซีเอ็นซีประกอบไปด้วยแกน 3 แกน คือ x y z ดังในรูปที่ 2 โดยจะมีมอเตอร์ประจำอยู่ทั้ง 3 แกนเพื่อเป็นตัวขับเคลื่อนเพลาแกนให้เคลื่อนที่ไปตามแนวแกน โดยแกน x และ y จะใช้สเต็ปเปอร์เนื่องจากต้องการแรงบิดที่สูงเพราะตัวเครื่องเองมีน้ำหนักที่ค่อนข้างมาก ส่วนแกน z ใช้เซอร์โวมอเตอร์ในการบังคับ

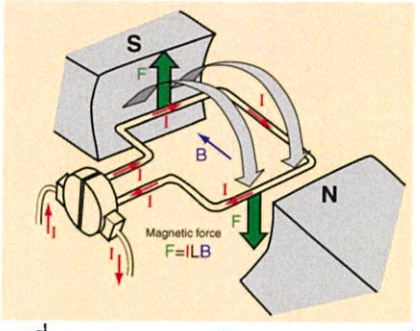


รูปที่ 2.2 แกน XYZ บนเครื่องซีเอ็นซี

2.2 Electric motor

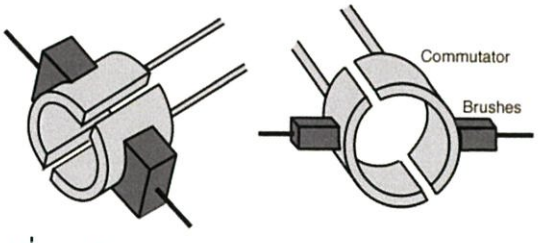
2.2.1 หลักการทำงานของมอเตอร์

มอเตอร์ไฟฟ้ากระแสตรงประกอบด้วย แม่เหล็กถาวร 2 ขั้ววางอยู่ระหว่างขดลวดตัวนำ ขดลวดตำนำจะได้รับแรงดันไฟตรงป้อนให้ในการทำงาน ทำให้เกิดอำนาจแม่เหล็ก 2 ชุด มีขั้วแม่เหล็กเหมือนกันวางใกล้กัน เกิดแรงผลักดันทำให้ขดลวดตัวนำหมุนเคลื่อนที่ได้ การทำงานเบื้องต้นของมอเตอร์ไฟฟ้ากระแสตรง แสดงดังรูปที่ 2.3



รูปที่ 2.3 แสดงการทำงานของมอเตอร์

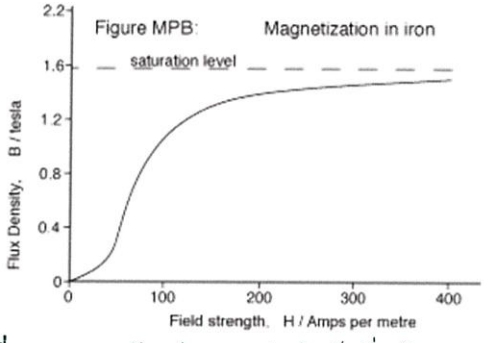
จากรูปที่ 2.3 เป็นการทำงานเบื้องต้นของมอเตอร์ไฟฟ้ากระแสตรง มีแรงดันไฟตรงจ่ายผ่านแปรงถ่านไปคอมมิวเตเตอร์ ผ่านไปให้ขดลวดตัวนำที่อาร์เมเจอร์ ทำให้ขดลวดอาร์เมเจอร์เกิดสนามแม่เหล็กไฟฟ้าขึ้นมา ทางด้านขวามือเป็นขั้วเหนือ (N) และด้านซ้ายเป็นขั้วใต้ (S) เหมือนกับขั้วแม่เหล็กถาวรที่วางอยู่ใกล้ๆ เกิดอำนาจแม่เหล็กผลักดันกัน อาร์เมเจอร์หมุนไปในทิศทางตามเข็มนาฬิกา พร้อมกับคอมมิวเตเตอร์ (Commutator) หมุนตามไปด้วย แปรงถ่าน (Brushes) สัมผัสกับส่วนของคอมมิวเตเตอร์เปลี่ยนไปในอีกปลายหนึ่งของขดลวด แต่มีผลทำให้เกิดขั้วแม่เหล็กที่อาร์เมเจอร์เหมือนกับขั้วแม่เหล็กถาวรที่อยู่ใกล้ๆอีกครั้ง ทำให้อาร์เมเจอร์ยังคงถูกผลักให้หมุนไปในทิศทางตามเข็มนาฬิกาตลอดเวลา เกิดการหมุนของอาร์เมเจอร์คือมอเตอร์ไฟฟ้าทำงาน



รูปที่ 2.4 ลักษณะของ Commutator และ Brushes

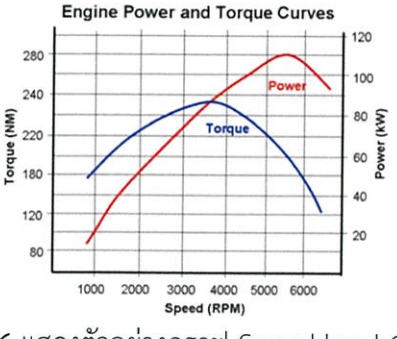
2.2.2 คุณสมบัติทั่วไป

คุณสมบัติทางเทคนิคเป็นคุณสมบัติประจำเครื่องกลไฟฟ้าแต่ละประเภทเช่นเดียวกันที่ให้รายละเอียดซึ่งเจาะลึกเข้าไปในเชิงวิชาการสามารถทดสอบและวัดด้วยเครื่องวัดได้ด้วยวิธีทดลองในห้องปฏิบัติการทดลอง ส่วนใหญ่จะแสดงด้วยกราฟเพื่อแสดงให้เห็นความสัมพันธ์ระหว่างค่าหนึ่งกับอีกค่าหนึ่ง เช่น สมรรถในการกำเนิดแรงเคลื่อนไฟฟ้าของเครื่องกำเนิดไฟฟ้าแสดงด้วย “กราฟแม่เหล็กอิ่มตัว (Saturation หรือ Magnetization curve) ”

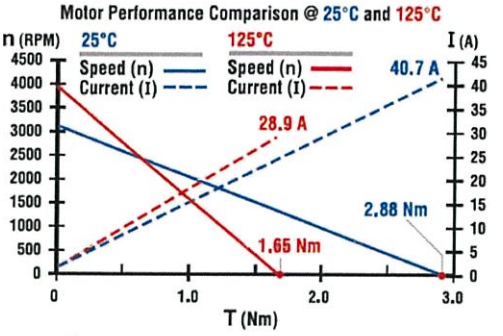


รูปที่ 2.5 แสดงตัวอย่างกราฟแม่เหล็กอิ่มตัวของ iron

สมรรถนะในการจ่ายโหลดของเครื่องกำเนิดไฟฟ้าแสดงด้วย External Characteristic ส่วนคุณสมบัติทางเทคนิคของมอเตอร์จะแสดงด้วย Performance Curve ซึ่งได้แก่ สมรรถนะในการหมุนขับโหลด (Speed load Curves หรือ Speed/load Characteristic) แสดงให้เห็นความสัมพันธ์ระหว่างความเร็วรอบกับกระแสมอเตอร์ (n = ความเร็วรอบให้อยู่บนแกน Y และ I_a = กระแสอาร์เมเจอร์ให้อยู่บนแกน X) หรืออาจให้แสดงความสัมพันธ์ระหว่างความเร็วรอบ (n เป็นแกน Y) กับทอร์ค หรือกำลังที่หมุนขับงาน (T= ทอร์ค, P=กำลังวัตต์หรือกิโลวัตต์ ให้อยู่บนแกน x) จุดประสงค์เพื่อต้องการแสดงให้เห็นถึงความเปลี่ยนแปลงของความเร็วรอบของมอเตอร์ที่หมุนขับโหลดว่าจะมีการเปลี่ยนแปลงไปอย่างไรเมื่อโหลดเปลี่ยนแปลงไป



รูปที่ 2.6 แสดงตัวอย่างกราฟ Speed load Curves



รูปที่ 2.7 ตัวอย่าง Performance Curve

2.2.3 กฎมือซ้ายสำหรับมอเตอร์ (Fleming's left hand rule)

เนื่องจากมีความสัมพันธ์อย่างแน่นอนเกิดขึ้นระหว่างทิศทางของสนามแม่เหล็ก ทิศทางของกระแสไฟฟ้าในตัวนำและทิศทางที่ตัวนำเคลื่อนที่ซึ่งมีความสัมพันธ์ของปริมาณเหล่านี้ให้ตั้งเป็นกฎมอเตอร์ขึ้น ซึ่งกฎนี้ได้นำไปใช้แบบเดียวกันกับกฎมือขวาของเครื่องกำเนิดไฟฟ้าเป็นแต่เพียงใช้มือซ้ายแทนเท่านั้น กฎนี้ ได้แสดงให้เห็นดังรูปที่ 2.8 และได้กล่าวไว้ดังนี้คือ นิ้วกลางและนิ้วหัวแม่มือ นิ้วชี้และนิ้วกลาง ให้ตั้งฉากซึ่งกันและกัน โดยใช้นิ้วชี้ ชี้ไปตามทิศทางของสนามแม่เหล็ก (Magnetic flux = B) นิ้วกลางชี้ไปตามทิศทางการไหลของกระแสไฟฟ้า (Current = I) แล้วนิ้วหัวแม่มือจะบอกทิศทางของการเคลื่อนที่ของตัวนำ (Force = F) แรงที่เกิดขึ้นในตัวนำการกระทำของแรงที่เกิดขึ้นเป็นตัวนำที่มีกระแสไฟฟ้าไหลผ่านในขณะที่มันวางอยู่ในสนามแม่เหล็กจะเป็นปฏิภาคโดยตรงกับความหนาแน่นของเส้นแรงแม่เหล็ก ความยาวของตัวนำและค่ากระแสไฟฟ้าที่ไหลผ่านตัวนำแรงที่เกิดขึ้นบนตัวนำสามารถหาได้จากสมการ

$$F = BIL$$

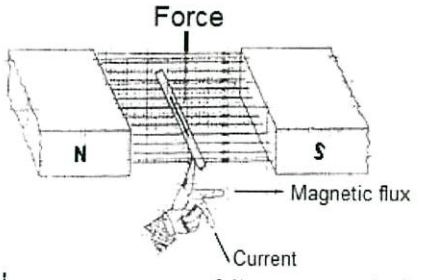
เมื่อ

F = แรงที่เกิดขึ้นบนตัวนำหนึ่งตัว หน่วย นิวตัน (N)

B = ความหนาแน่นสนามแม่เหล็ก หน่วย (Wb/m^2)

I = กระแสที่ไหลในตัวนำ หน่วย แอมแปร์ (A)

L = ความยาวของตัวนำ หน่วย เมตร (m)



รูปที่ 2.8 แสดงภาพการใช้งานของกฎมือซ้าย

2.2.4 ประเภทของมอเตอร์

2.2.4.1 มอเตอร์ไฟฟ้ากระแสตรง

หลักการของมอเตอร์ไฟฟ้ากระแสตรง (Motor Action) เมื่อเป็นแรงดันกระแสไฟฟ้าตรงเข้าไปในมอเตอร์ ส่วนหนึ่งจากแปรงถ่านผ่านคอมมิวเตเตอร์เข้าไปในขดลวดอาร์มาเจอร์สร้างสนามแม่เหล็กขึ้น และกระแสไฟฟ้าอีกส่วนหนึ่งจะไหลเข้าไปในขดลวดสนามแม่เหล็ก (Field coil) สร้างขั้วเหนือ-ใต้ขึ้น จะเกิดสนามแม่เหล็ก 2 สนาม ในขณะเดียวกัน ตามคุณสมบัติของเส้นแรง แม่เหล็ก จะไม่ตัดกันทิศทางตรงข้ามจะหักล้างกัน และทิศทางเดียวจะเสริมแรงกัน ทำให้เกิดแรงบิดในตัวอาร์มาเจอร์ ซึ่งวางแกนเพลาและแกนเพลานี้ สวมอยู่กับตลับลูกปืนของมอเตอร์ ทำให้อาร์มาเจอร์นี้หมุนได้ ขณะที่ตัวอาร์มาเจอร์ทำหน้าที่หมุนได้นี้เรียกว่า โรเตอร์ (Rotor) ซึ่งหมายความว่าตัวหมุน การที่อำนาจเส้นแรงแม่เหล็กทั้งสองมีปฏิกิริยาต่อกัน ทำให้ขดลวดอาร์มาเจอร์หรือโรเตอร์ หมุนไปนั้นเป็นไปตามกฎซ้ายของเฟลมมิ่ง (Fleming'left hand rule)

มอเตอร์ไฟฟ้ากระแสตรงแบ่งออกเป็น 3 ชนิดได้แก่

- มอเตอร์แบบอนุกรมหรือเรียกว่าซีรีส์มอเตอร์ (Series Motor)
- มอเตอร์แบบอนุขนานหรือเรียกว่าชันท์มอเตอร์ (Shunt Motor)
- มอเตอร์ไฟฟ้าแบบผสมหรือเรียกว่าคอมเปาวด์มอเตอร์ (Compound Motor)

ส่วนประกอบของมอเตอร์ไฟฟ้ากระแสตรง

เฟรมหรือโยค (Frame Or Yoke) เป็นโครงภายนอกทำหน้าที่เป็นทางเดินของเส้นแรงแม่เหล็กจากขั้ว

ขั้วแม่เหล็ก (Pole) ประกอบด้วย 2 ส่วนคือแกนขั้วแม่เหล็กและขดลวด

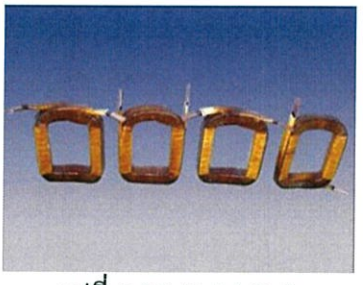
รูปที่ 2.9 ภาพขดลวดพันอยู่รอบขั้วแม่เหล็ก (1)

รูปที่ 2.10 ภาพขดลวดพันอยู่รอบขั้วแม่เหล็ก (2)

แกนขั้ว (Pole Core) ทำด้วยแผ่นเหล็กบางๆ กั้นด้วยฉนวนประกอบกันเป็นแท่งยึดติดกับเฟรม ส่วนปลายที่ทำเป็นรูปโค้งนั้นเพื่อโค้งรับรูปกลมของตัวโรเตอร์เรียกว่าขั้วแม่เหล็ก (Pole Shoes) มีวัตถุประสงค์ให้ขั้วแม่เหล็กและโรเตอร์ใกล้ชิดกันมากที่สุดเพื่อให้เกิดช่องอากาศน้อยที่สุด เพื่อให้เกิดช่องอากาศน้อยที่สุดจะมีผลให้เส้นแรงแม่เหล็กจากขั้วแม่เหล็กจากขั้วแม่เหล็กผ่านไปยังโรเตอร์มากที่สุดแล้วทำให้เกิดแรงบิดหรือกำลังบิดของโรเตอร์มากเป็นการทำให้มอเตอร์มีกำลังหมุน (Torque)

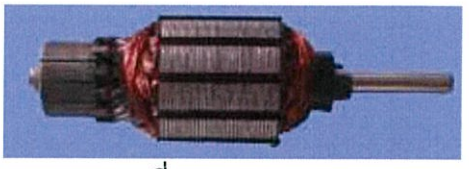
รูปที่ 2.11 แกนขั้ว

ขดลวดสนามแม่เหล็ก (Field Coil) จะพันอยู่รอบๆแกนขั้วแม่เหล็กขดลวดนี้ทำหน้าที่รับกระแสจากภายนอกเพื่อสร้างเส้นแรงแม่เหล็กให้เกิดขึ้น และเส้นแรงแม่เหล็กนี้จะเกิดการหักล้างและเสริมกันกับสนามแม่เหล็กของอาเมเจอร์ทำให้เกิดแรงบิดขึ้น



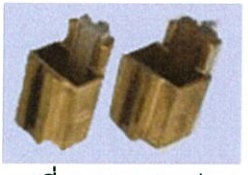
รูปที่ 2.12 Field Coil

แกนเพลา (Shaft) เป็นตัวสำหรับยึดคอมมิวเตเตอร์ และยึดแกนเหล็กอาร์มาเจอร์ (Armature Core) ประกอบเป็นตัวโรเตอร์แกนเพลานี้จะวางอยู่บนแบริ่ง เพื่อบังคับให้หมุนอยู่ในแนวนิ่งไม่มีการสั่นสะเทือนได้



รูปที่ 2.13 แกนเพลา

แปรงถ่าน ทำด้วยคาร์บอนมีรูปร่างเป็นแท่งสี่เหลี่ยมผืนผ้าในซองแปรงมีสปริงกดอยู่ด้านบน เพื่อให้ถ่านนี้สัมผัสกับซี่คอมมิวเตเตอร์ตลอดเวลาเพื่อรับกระแส และส่งกระแสไฟฟ้าระหว่างขดลวดอาร์มาเจอร์ กับวงจรไฟฟ้าจากภายนอก คือถ้าเป็นมอเตอร์กระแสไฟฟ้าตรงจะทำหน้าที่รับกระแสจากภายนอกเข้าไปยังคอมมิวเตเตอร์ให้ลวดอาร์มาเจอร์เกิดแรงบิดทำให้มอเตอร์หมุนได้



รูปที่ 2.14 แปรงถ่าน

2.2.4.2 มอเตอร์ไฟฟ้ากระแสสลับ

มอเตอร์ไฟฟ้ากระแสสลับ หมายถึง มอเตอร์ที่ใช้กับระบบไฟฟ้ากระแสสลับ เป็นเครื่องกลไฟฟ้าที่ทำหน้าที่เปลี่ยนพลังงานไฟฟ้าให้เป็นพลังงานกล ส่วนที่ทำหน้าที่เปลี่ยนพลังงานไฟฟ้าคือขดลวดในสเตเตอร์และส่วนที่ทำหน้าที่ให้พลังงานกล คือ ตัวหมุนหรือโรเตอร์ ซึ่งเมื่อขดลวดในสเตเตอร์ได้รับพลังงานไฟฟ้าก็จะสร้างสนามแม่เหล็กขึ้นมาในตัวที่อยู่กับที่หรือ สเตเตอร์ ซึ่งสนามแม่เหล็กที่เกิดขึ้นนี้จะมีการเคลื่อนที่หรือหมุนไปรอบ ๆ สเตเตอร์ เนื่องจากการต่างเฟสของกระแสไฟฟ้าในขดลวดและการเปลี่ยนแปลงของกระแสไฟฟ้า ในขณะที่สนามแม่เหล็กเคลื่อนที่ไปสนามแม่เหล็กจากขั้วเหนือก็จะพุ่งเข้าหาขั้วใต้ ซึ่งจะไปตัดกับตัวนำที่เป็นวงจรปิดหรือขดลวดกรงกระรอกของตัวหมุนหรือโรเตอร์ ทำให้เกิดการเหนี่ยวนำของกระแสไฟฟ้าขึ้นในขดลวดของโรเตอร์ ซึ่งสนามแม่เหล็กของโรเตอร์นี้จะเคลื่อนที่ตามทิศทางการเคลื่อนที่จองสนามแม่เหล็กที่สเตเตอร์ ก็จะทำให้โรเตอร์ของมอเตอร์เกิดจะพลังงานกลสามารถนำไปขับภาระที่ต้องการหมุนได้

ชนิดของมอเตอร์ไฟฟ้ากระแสสลับ มอเตอร์ไฟฟ้ากระแสสลับแบ่งออกเป็น 2 ชนิดใหญ่ๆ คือ มอเตอร์อะซิงโครนัส และ มอเตอร์ซิงโครนัส ซึ่งที่กล่าวในบทนี้จะเป็นมอเตอร์อะซิงโครนัส ที่เรียกว่ามอเตอร์ชนิดเหนี่ยวนำ ซึ่งจะมีขนาดตั้งแต่เล็กๆไปจนถึงขนาดหลายร้อยแรงม้า มอเตอร์ชนิดเหนี่ยวนำมีทั้งที่เป็นมอเตอร์ชนิด 1 เฟสและชนิดที่เป็นมอเตอร์ 3 เฟส มอเตอร์ชนิดเหนี่ยวนำนั้นส่วนมากแล้วจะหมุนด้วยความเร็วคงที่แต่ก็มีบางชนิดที่สามารถเปลี่ยนแปลงความเร็วได้ เช่น มอเตอร์สลิปริงหรือมอเตอร์ชนิดขดลวดพัน ซึ่งจะเป็นมอเตอร์ชนิด 3 เฟส มอเตอร์ไฟฟ้ากระแสสลับชนิดเหนี่ยวนำเป็นเครื่องกลไฟฟ้าที่เปลี่ยนพลังงานไฟฟ้าให้เป็นพลังงานกล ในการเปลี่ยนพลังงานไฟฟ้าให้เป็นพลังงานกลนี้ โรเตอร์

ไม่ได้รับพลังงานไฟฟ้าโดยตรงแต่จะได้จากการเหนี่ยวนำ ดังนั้นจึงเรียกว่ามอเตอร์ชนิดเหนี่ยวนำซึ่งแบ่งออกเป็น 2 ชนิด คือ

1. มอเตอร์ชนิดกรงกระรอก ซึ่งมีทั้งที่เป็นมอเตอร์ 1 เฟสและชนิดที่เป็น 3 เฟส

2. มอเตอร์ชนิดขดลวดพันหรือชนิดวาวนด์หรือมอเตอร์สลิปริง ซึ่งจะเป็นมอเตอร์ชนิด 3 เฟส

โดยทั่วไป มอเตอร์ทุกประเภทจะมีส่วนประกอบหลัก หรือส่วนประกอบเบื้องต้นคล้ายกันคือสเตเตอร์หรือตัวที่อยู่กับที่และโรเตอร์หรือตัวหมุน แต่จะแตกต่างกันในเรื่องของรายละเอียดของส่วนประกอบปลีกย่อยอื่นๆ

ส่วนประกอบของมอเตอร์ไฟฟ้ากระแสสลับ

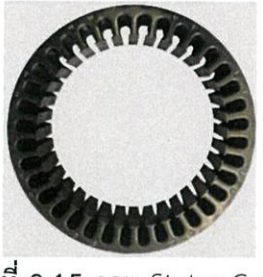
สเตเตอร์หรือตัวอยู่กับที่ (Stator) จะเป็นส่วนที่อยู่กับที่ซึ่งจะประกอบด้วยโครงของมอเตอร์ แกนเหล็กสเตเตอร์ และขดลวด

โครงมอเตอร์ (Frame or Yoke) จะทำด้วยเหล็กหล่อทรงกระบอกกลวง ฐานส่วนล่างจะเป็นขาตั้ง มีกล่องสำหรับต่อสายไฟอยู่ด้านบนหรือด้านข้าง โครงจะทำหน้าที่ยึดแกนเหล็กสเตเตอร์ให้แน่นอยู่กับที่ผิวด้านนอกของโครงมอเตอร์ จะออกแบบให้มีลักษณะเป็นครีบ เพื่อช่วยในการระบายความร้อน

ในกรณีที่เป็นมอเตอร์ขนาดเล็กๆ โครงจะทำด้วยเหล็กหล่อ แต่ถ้าเป็นมอเตอร์ขนาดใหญ่ โครงจะทำด้วยเหล็กหล่อเหนียว ซึ่งจะทำให้มอเตอร์มีขนาดเล็กกะทัดรัดมากขึ้น แต่ถ้าใช้เหล็กหล่อก็จะให้มีขนาดใหญ่ น้ำหนักมาก

นอกจากนี้แล้วโครงของมอเตอร์ยังอาจทำด้วยเหล็กหล่อเหนียวม้วนเป็นแผ่นม้วนรูปทรงกระบอก แล้วเชื่อมติดกันให้มีความแข็งแรง เช่น มอเตอร์สปลิตเฟส เป็นต้น

แกนเหล็กสเตเตอร์ (Stator Core) ทำด้วยแผ่นเหล็กบาง ๆ มีลักษณะกลม เจาะตรงกลางและเซาะร่องภายในโดยรอบ แผ่นเหล็กชนิดนี้เรียกว่า ลามิเนท ซึ่งจะถูกเคลือบด้วย ซิลิกอน เหล็กแต่ละแผ่นจะมีความหนาประมาณ 0.025 นิ้ว หลังจากนั้นจึงนำไปอัดเข้าด้วยกันจนมีความหนาที่เหมาะสม เรียกว่าแกนเหล็กสเตเตอร์



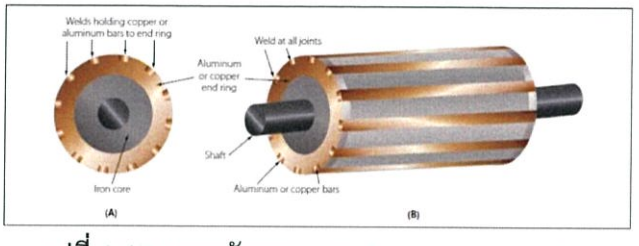
รูปที่ 2.15 ภาพ Stator Core

ขดลวด (Stator Winding) จะมีลักษณะป็นเส้นลวดทองแดงเคลือบฉนวนที่เรียกว่า อีนาเมล (Enamel) พันอยู่ในร่องของแกนเหล็กสเตเตอร์ตามรูปแบบต่าง ๆ ของการพันมอเตอร์

รูปที่ 2.16 Stator Winding

โรเตอร์หรือตัวหมุน (Rotor) มอเตอร์ชนิดเหนี่ยวนำจะมีโรเตอร์ 2 ชนิด คือ โรเตอร์แบบกรงกระรอกและโรเตอร์แบบขดลวดพันหรือแบบวาวนด์ ซึ่งจะมีส่วนประกอบดังนี้คือ แกนเหล็ก โรเตอร์ขดลวด ใบพัด และเพลา

โรเตอร์แบบกรงกระรอก (Squirrel cage rotor) จะประกอบด้วยแผ่นเหล็กบาง ๆ ที่เรียกว่าแผ่นเหล็กลามิเนท ซึ่งจะเป็นแผ่นเหล็กชนิดเดียวกันกับสเตเตอร์ มีลักษณะเป็นแผ่นกลมๆ เซาะร่องผิวภายนอกเป็นร่องโดยรอบ ตรงกลางจะเจาะรูสำหรับสวมเพลา และจะเจาะรูรอบๆ รูตรงกลางที่สวมเพลาทั้งนี้เพื่อช่วยให้ในการระบายความร้อน และยังทำให้โรเตอร์มีน้ำหนักเบาลง เมื่อนำแผ่นเหล็กไปสวมเข้ากับแกนเพลาแล้วจะได้เป็นแกนเหล็กโรเตอร์ หลังจากนั้นก็จะใช้แท่งตัวทองแดงหรือแท่งอะลูมิเนียมหล่ออัดเข้าไปในร่องของแกนเหล็กสเตเตอร์เข้าไปวางทั้งสองด้านด้วย วงแหวนตัวนำทั้งนี้เพื่อให้ขดลวดครบวงจรไฟฟ้าหรืออาจนำแกนเหล็กสเตเตอร์เข้าไปในแบบพิมพ์แล้วฉีดอะลูมิเนียมเหลวเข้าไปในร่อง ก็จะได้อะลูมิเนียมอัดแน่นอยู่ในร่องจนเต็มและจะได้ขดลวดตัวนำแบบกรงกระรอกฝังอยู่ในแกนเหล็ก ดังแสดงในรูปที่ 2.17

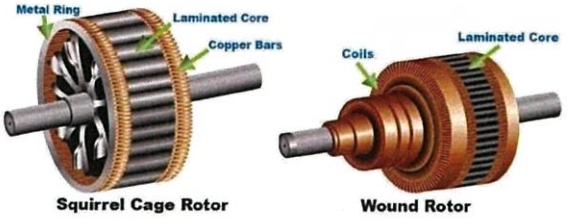


รูปที่ 2.17 แสดงลักษณะของ Squirrel cage rotor

ขดลวดในโรเตอร์นั้นจะเป็นลักษณะของตัวนำที่เป็นแท่งซึ่งอาจใช้ทองแดง หรืออะลูมิเนียมประกอบเข้าด้วยกันเป็นลักษณะคล้ายกรงนกหรือกรงกระรอก

โรเตอร์แบบขดลวดพันหรือแบบวาวนด์ (Wound Rotor) โรเตอร์ชนิดนี้จะมีส่วนประกอบคล้าย ๆ กับโรเตอร์แบบกรงกระรอก คือ มีแกนเหล็กที่เป็นแผ่นลามิเนทอัดเข้าด้วยกันแล้วสวมเข้าที่เพลา แต่จะแตกต่างกันตรงที่ขดลวด จะเป็นเส้นลวดชนิดที่หุ้มด้วยน้ำยาฉนวนอีนาเมลพันลงไปในร่องสล็อตของโร

เตอร์จำนวน 3 ชุด ซึ่งจะมีลักษณะเหมือนกับที่พันบนสเตเตอร์ของมอเตอร์ 3 เฟสแล้วต่อวงจรขดลวดเป็นแบบสตาร์ โดยนำปลายทั้ง 3 ที่เหลือต่อเข้ากับวงแหวนตัวนำ ทั้งนี้เพื่อให้สามารถต่อวงจรของขดลวดของโรเตอร์เข้ากับตัวต้านทานที่ปรับค่าได้ที่อยู่ภายนอกตัวมอเตอร์ เพื่อการปรับค่าความต้านทานของโรเตอร์ ซึ่งจะสามารถควบคุมความเร็วของโรเตอร์ได้



รูปที่ 2.18 เปรียบเทียบระหว่าง Squirrel cage rotor และ Wound rotor

ฝาครอบ (End Plate) ส่วนมากจะทำด้วยเหล็กหล่อ เจาะรูตรงกลางและคว้านเป็นรูกลมใหญ่เพื่อัดแบริ่งหรือตลับลูกปืนรองรับแกนเพลาของโรเตอร์

ฝาครอบใบพัด (Fan End Plate) จะมีลักษณะเป็นแผ่นเหล็กเหนียวขึ้นรูปให้มีขนาดสวมฝาครอบได้พอดี มีรูเจาะเพื่อระบายอากาศ และยึดติดกับฝาครอบด้านที่มีใบพัด ส่วนใหญ่จะมีในมอเตอร์ 3 เฟสและมอเตอร์ 1 เฟสขนาดใหญ่

ใบพัด (Fan) จะทำด้วยเหล็กหล่อ มีลักษณะเท่ากันทุกครีบเท่ากันทุกครีบ จะสวมยึดอยู่บนเพลาด้านตรงข้ามกันกับเพลางาน ใบพัดนี้จะช่วยในการระบายอากาศและความร้อนได้มากทีเดียวใบพัดนี้ส่วนใหญ่จะมีในมอเตอร์ 3 เฟสและมอเตอร์ 1 เฟสขนาดย่อยถึงขนาดใหญ่ เช่นเดียวกับฝาครอบใบพัด

สลักเกลียว (Bolt) จะทำด้วยเหล็กเหนียวจะมีลักษณะเป็นเกลียวตลอด ถ้าเป็นมอเตอร์ 3 เฟส จะประกอบด้วยสลักเกลียว 8 ตัว ทำหน้าที่ยึดฝาครอบให้ติดกับโครง ถ้าเป็นมอเตอร์ 1 เฟสขนาดเล็ก เช่น มอเตอร์สปลิตเฟสจะเป็นสลักเกลียวยาวตลอดความยาวของตัวมอเตอร์ ทำเกลียวเฉพาะด้านปลายและมีน็อตขันยึดไว้ ดังนั้นจึงมีเพียง 4 ตัว

มอเตอร์ไฟฟ้ากระแสสลับ จะมี 2ประเภทใหญ่ ๆ คือ

- มอเตอร์อะซิงโครนัส (Asynchronous motor)

- มอเตอร์ซิงโครนัส (Synchronous motor) ซึ่งจะเป็นมอเตอร์ประเภทที่อาศัยหลักการเหนี่ยวนำ

โรเตอร์ของมอเตอร์เหนี่ยวนำ จะมี 2 ประเภท คือ

- โรเตอร์แบบกรงกระรอก (Squirrel cage rotor) ซึ่งจะมีลักษณะเป็นตัวนำหลายตัวประกอบเข้าด้วยกันเป็นรูปทรงกระบอก แล้วลัดวงจรหัวท้ายด้วยวงแหวนตัวนำ

- โรเตอร์แบบขดลวดพันหรือแบบวาวนด์ (Wound rotor) จะมีลักษณะเป็นขดลวดพันในร่องสล็อตของโรเตอร์จำนวน 3 ชุด ขดลวดด้านปลายต่อเข้าด้วยกันเป็นแบบสตาร์ ด้านต้นต่อเข้ากับวงแหวนลื่นหรือสลิปริง (Slip ring)

ส่วนประกอบทั่วไปของมอเตอร์ไฟฟ้ากระแสสลับ จะประกอบด้วย

สเตเตอร์หรือตัวอยู่กับที่ (Stator) มีลักษณะเป็นแผ่นลามิเนทประกอบเข้าด้วยกันเป็นแกนเหล็ก มีร่องเอาไว้สำหรับพันขดลวดเพื่อทำหน้าที่เป็นตัวกำเนิดสนามแม่เหล็กและเป็นวงจรแม่เหล็ก

โรเตอร์หรือตัวหมุน (Rotor) มีลักษณะเป็นแกนเหล็กทรงกระบอกจะหมุนอยู่ในช่องสเตเตอร์ซึ่งจะทำหน้าที่กำเนิดกำลังกลเพื่อส่งไปขับโหลด

ฝาครอบทั้ง 2 ด้าน (End Plate) จะมีหน้าที่ยึดโรเตอร์ให้หมุนอยู่ในช่องของ สเตเตอร์อย่างสมดุล

2.3 Stepping Motor

Step Motor เป็นมอเตอร์จะหมุนเป็นสเต็ป ทีละสเต็ปเล็กๆเท่ากับองศาที่กำหนดไว้ตามมอเตอร์แต่ละตัวโดยจะขยับตามสัญญาณพัลซ์ที่ส่งเข้ามาทีละสเต็ป ในสิ่งที่อธิบายข้างต้น เรียกว่า Step angle โดยมีหน่วยเป็นองศา ยิ่ง step angle มีขนาดที่เล็กก็จะทำให้การหมุนของมอเตอร์มีความละเอียดมากขึ้นตามไปด้วย โดยทางวิศวกรรรมจะสนใจในจำนวนสเต็ปต่อ 1 รอบการหมุน (steps per revolution) แทนด้วยตัว S และ Step angle แทนด้วย θ_s โดยมีความสัมพันธ์ ดังนี้

$$\theta_s = 360/s$$

และ S มีความสัมพันธ์กับจำนวนฟันของ rotor (N_r) และจำนวน phase แทนด้วย m

สำหรับ VR motor

$$S = mN_r$$

สำหรับ PM motor และ Hybrid motor

$$S = 2mN_r$$

และยังมีความสัมพันธ์เพิ่มเติมเกี่ยวกับความเร็วในการหมุนของมอเตอร์อีกด้วย ซึ่งในสเต็ปมอเตอร์ความเร็วในการหมุนจะหมายถึง สเต็ปต่อวินาที หรือ stepping rate โดยจะสามารถเทียบกับมอเตอร์อื่นๆที่ใช้หน่วยรอบต่อนาทีโดยสูตรดังนี้

$$n = 60f/S$$

โดย n = ความเร็วในการหมุน (r.p.m)

f = stepping rate (Hz)

S = steps per revolution

สามารถสรุปข้อดีและข้อเสียของ Stepping motor ได้ดังนี้

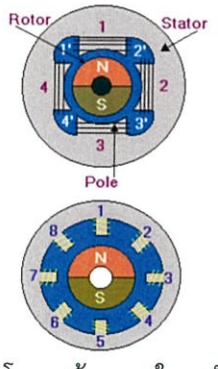
ข้อดี

- มุมที่จะหมุนมีค่าตามสัดส่วนของจำนวนของพัลส์อินพุตที่ใช้ขับมอเตอร์
- ความเร็วในการหมุน (rotation speed) มีค่าตามสัดส่วนและสัมพันธ์กับความถี่ของสัญญาณพัลส์อินพุตที่ใช้ขับมอเตอร์
- ใช้ในการควบคุมตำแหน่งแบบระบบเปิดที่มีความแม่นยำสูง โดยไม่มีต้องใช้สัญญาณป้อนกลับของการกำหนดตำแหน่ง
- ไม่มีความผิดพลาดสะสมของการกำหนดตำแหน่ง
- เหมาะกับงานที่ต้องการกลไกเคลื่อนที่ความเร็วต่ำ แรงบิดสูง โดยไม่ต้องใช้ระบบเฟืองทดรอบเพิ่มเติม
- สามารถกำเนิดและรักษาแรงบิดได้ในทันทีที่มอเตอร์ถูกกระตุ้นให้ทำงาน
- สามารถรักษาสภาวะการหมุนของแกนได้โดยไม่ทำให้มอเตอร์เสียหาย
- ไม่มีแปรงถ่าน ทำให้มีอายุการใช้งานที่ยาวนาน
- มีลูกปืนความเที่ยงตรงสูง เพื่อช่วยการหมุนของแกนมีความแม่นยำ

ข้อเสีย

- การกำทอนหรือการเกิดเรโซแนนซ์ทำให้ไม่สามารถควบคุมการทำงานของสเต็ปเปอร์มอเตอร์ได้
- การทำให้มอเตอร์สามารถหมุนแกนด้วยความเร็วสูงทำได้ยาก
- หากเกิดแรงบิดสูงเกินกว่าที่รับได้หรือเกิดโอเวอร์ทอร์กมอเตอร์จะสูญเสียการรับรู้ตำแหน่งของแกนหมุน จะต้องกลับไปเริ่มต้นการอินิเชียลใหม่
- ให้แรงบิดที่น้อยกว่ามอเตอร์ไฟตรงและมอเตอร์ไฟสลับที่ขนาดของตัวมอเตอร์เท่ากัน

โครงสร้างของขั้วแม่เหล็กบนสเตเตอร์ทำมาจากแผ่นเหล็กวงแหวนที่มีซี่ยื่นออกมาประกอบกันเป็นชั้นๆ โดยที่แต่ละซี่นั้นจะมีคอยล์ (ขดลวด) พันสวมอยู่ เมื่อมีการป้อนกระแสผ่านคอยล์ทำให้เกิดสนามแม่ เหล็กไฟฟ้า (Electromagnetic) ดูดังรูปด้านล่างนี้ จะแสดงถึงองค์ประกอบดังกล่าว



รูปที่ 2.19 โครงสร้างภายในสเต็ปมอเตอร์

ในที่นี้ซึ่งถ้าเพิ่มจำนวนของขั้วแม่เหล็กมากขึ้นจะเพิ่มจำนวนของสเต็ปต่อวงจรรอบมากขึ้นตามด้วย

ลักษณะการนำไปใช้งาน สเต็ปมอเตอร์ ใช้งานลักษณะ Open Loop System แปลเป็นภาษาไทย ระบบเปิด คือ สเต็ปมอเตอร์สามาทำงานได้โดยไม่ต้องมีการ ป้อนค่าพารามิเตอร์กลับมา (Feed back) แต่ทุกวิธีที่ต้องการกำหนดตำแหน่งที่แน่นอนนั้นล่ะ จะต้องการป้อนกลับไปยังระบบและตัวบอก ตำแหน่งว่าถูกต้องหรือผิดผลาดให้รับทราบ

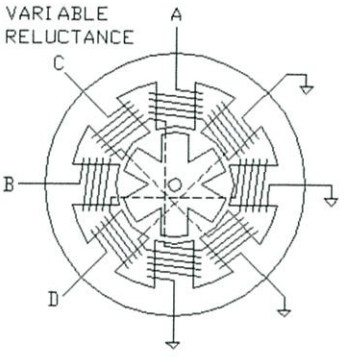
ดังเช่นวิธีที่ใช้กับสเต็ปมอเตอร์ คือเรานำลิมิตสวิตช์ ติดตามตำแหน่งที่จะตรวจจับ เมื่อสเต็ปมอเตอร์ เริ่มหมุนแล้วหมุนไปจนถึงตำแหน่งของสวิตช์ตรวจจับสัญญาณ สวิตช์ทำงานก็จะป้อนกลับไปสู่ระบบ ซึ่งก็จะทำให้รู้การทำงานของสเต็ปมอเตอร์ตลอด ตัววงจรไมโครคอนโทรลเลอร์เองจะมีจุดอ้างอิงไว้ให้เริ่มต้นการทำงานและอ้างอิงตำแหน่งได้ถูกต้อง

โดยแนวทางสเต็ปมอเตอร์เป็นอุปกรณ์จำพวกเชิงกลทางไฟฟ้า โดยมีกรุ๊ปของไบนารี่โวลท์เตทเป็นอินพุตและการเคลื่อนที่ แบบเชิงมุมเป็นเอาท์พุต หรือว่าหมุนทีละสเต็ปซึ่งอยู่ระหว่าง 0.1 - 30 องศา อยู่ที่โครงสร้างของสเต็ปมอเตอร์ โดยตามสัญญาณ พัลซ์ที่จ่ายให้กับขดสเตเตอร์ทำให้เกิดแรงผลักแก่โรเตอร์หมุนไป สเต็ปมอเตอร์มีขดลวดหลายชุดในที่นี้เราเรียกว่า Phase ดังนั้นสัญญาณที่ต่อเนื่องเป็น Sequence ลักษณะของ Binary ซึ่งจะต้องไปผ่านวงจร Driver ก็จะ ทำให้โรเตอร์หมุนไปอย่างต่อเนื่องที่กล่าวมาสามารถดูได้จากรูปด้านบน

ในปัจจุบัน stepping motor สามารถแบ่งได้หลักๆ 3 ประเภทดังนี้

VR motor (variable-reluctance motor)

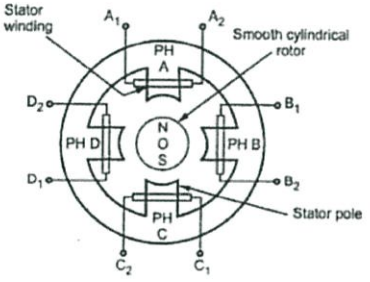
สเต็ปมอเตอร์แบบ VR จะมีการหมุนโรเตอร์ได้อย่างอิสระ แม้จะไม่ได้จ่ายไฟให้โรเตอร์ที่ทำจากสารเฟอร์โรแมกเนติก กำลังอ่อน มีลักษณะเป็นฟันเลื่อย รูปทรงกระบอกโดยจะมีความสัมพันธ์ โดยตรงกับจำนวนโพลในสเตเตอร์ แรงบิดที่เกิดขึ้นจะไปหมุนโรเตอร์ ไปในเส้นทางของอำนาจแม่เหล็กที่มีค่ารีลักแตนท์ต่ำที่สุด ตำแหน่งที่จะเกิดแน่นอนและมีเสถียรภาพแต่จะเกิดขึ้นได้หลายๆ จุดดังนั้นเมื่อป้อนไฟเข้าขดลวดต่างๆ ในมอเตอร์แตกต่างขดกันไป ก็ทำให้มอเตอร์ หมุนไปตำแหน่งต่างๆ กันโรเตอร์ของ VR จะมีความเฉื่อยของโรเตอร์น้อยจึงมีความเร็วรอบสูงกว่ามอเตอร์แบบ PM



รูปที่ 2.20 แสดงโครงสร้างภายในแบบภาพตัดมุมบนของ VR motor

PM motor (permanent-magnet motor)

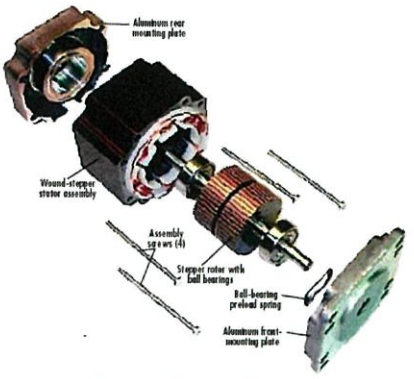
สเต็ปมอเตอร์แบบ PM จะมีสเตเตอร์ที่พันขดลวดไว้หลายๆ โพล โดยมีโรเตอร์เป็นรูปทรงกระบอกฟันเลื่อย และโรเตอร์นี้จะทำด้วยแม่เหล็กถาวร เพื่อป้อนไฟกระแสตรง ให้กับขดสเตเตอร์ จะทำให้เกิดแรงแม่เหล็กไฟฟ้าผลักต่อโรเตอร์ ทำให้มอเตอร์หมุนมอเตอร์แบบ PM จะเกิดแรงฉุดยึดให้โรเตอร์หยุดอยู่กับที่ แม้จะไม่ได้ป้อนไฟเข้าขดลวด



รูปที่ 2.21 แสดงโครงสร้างภายในแบบภาพตัดมุมบนของ PM motor

Hybrid motor

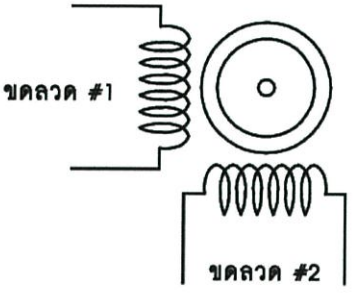
สเต็ปมอเตอร์แบบ Hybrid จะเป็นการนำ VR กับ PM motor มาผสมกัน โดยจะมีสเตเตอร์คล้ายกับที่ใช้ใน VR motor และนำโรเตอร์ที่มีความเป็นแม่เหล็กแบบ PM motor มาใส่และโดยเพิ่มจากที่โรเตอร์เป็นทรงกระบอก เป็นเพิ่มซี่ฟันบนตัวโรเตอร์รอบนอกอีกด้วย ข้อดีก็คือ ให้แรงบิดสูงและมีขนาดกระทัดรัด และให้แรงฉุดยึดโรเตอร์นิ่งกับที่ตอนไม่จ่ายไฟ



รูปที่ 2.22 แสดงโครงสร้างภายในของ Hybrid motor

และยังสามารถแบ่งสเต็ปมอเตอร์ได้ตามการพันขดสเตเตอร์ได้อีกด้วย การพันมีด้วยกัน 2 วิธี คือ แบบ Bipolar กับ แบบ Unipolar

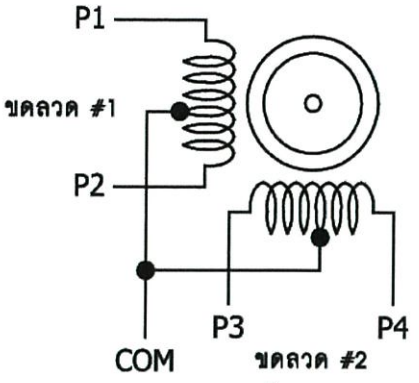
แบบ Bipolar



รูปที่ 2.23 แสดงการพันขวดลวดบนขั้วสเตเตอร์แบบ Bipolar

จะมีการพันขดลวดหนึ่งขด ในแต่ขั้วแม่เหล็กของสเตเตอร์ โดยขั้วแม่เหล็กที่เกิดขึ้น ที่สเตเตอร์จะถูกกำหนดโดยทิศทางของการไหลของกระแสไฟฟ้า ซึ่งสามารถทำให้เกิดขั้วแม่เหล็กในทิศทางตรงกันข้ามได้เพียง การกลับทิศทางของการไหลในกระแสไฟฟ้า โดยมาจากการควบคุมของวงจรสวิทชิ่งให้กลับขั้วไฟฟ้า

แบบ Unipolar



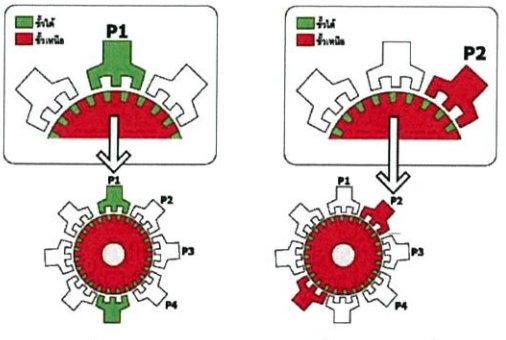
รูปที่ 2.24 แสดงการพันขวดลวดบนขั้วสเตเตอร์แบบ Unipolar

แบบนี้มีการพันขดลวด 2 ขด บนแต่ละขั้วแม่เหล็กของสเตเตอร์ ทำให้แต่ละขดลวดเกิดขั้วแม่เหล็กในทิศทางตรงกันข้าม ในการกลับทิศทางขั้วแม่เหล็กทำได้โดยใช้วงจรสวิทชิ่งในการสลับขั้ว

ความแตกต่างระหว่างการพัน 2 แบบนี้คือ แบบยูนิโพล่าร์จะทำให้เกิดแรงบิดน้อยกว่าแบบไบโพล่าร์ และ สายไฟที่ต่อมาจากตัวสเต็ปมอเตอร์ซึ่งแบบไบโพลาร์จะมี 4 สาย ส่วนเป็นแบบยูนิโพล่าร์จะมี 5 สายหรือ 6 สาย

การทำงานของสเต็ปเปอร์มอเตอร์แบบไฮบริด

สเต็ปเปอร์มอเตอร์แบบไฮบริดมีแม่เหล็กถาวรที่มีลักษณะเป็นทรงกระบอกที่มีขั้วเหนือและใต้สลับกันตามแนวของทรงกระบอกทำหน้าที่เป็นส่วนของโรเตอร์ ขดลวดทั้ง 4 เฟสที่พันรอบแกนเหล็กทำหน้าที่เป็นสเตเตอร์ที่มีขั้วแม่เหล็กเปลี่ยนแปลงตามสัญญาณกระตุ้นที่ส่งเข้าไปในตัวมอเตอร์ ทำให้เกิดแรงดูดและแรงผลักกับโรเตอร์ทำให้แกนของมอเตอร์เกิดการหมุนและล็อกตำแหน่งได้ตามที่ผู้ใช้งานต้องการ

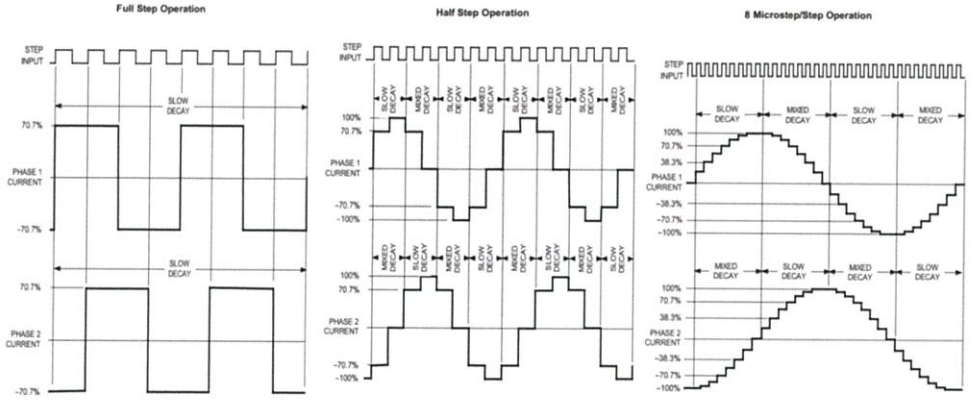


รูปที่ 2.25 ภาพจำลองการหมุนของสเต็ปมอเตอร์แบบยูนิโพล่าร์

ในรูปที่ 25 แสดงภาพจำลองของการหมุนของสเต็ปเปอร์มอเตอร์ที่มีความละเอียด 15 องศาต่อสเต็ป นั่นคือ มีจำนวนสเต็ปของการหมุนครบรอบเท่ากับ 24 สเต็ปได้ โดยใช้การขับแบบฟูลสเต็ป 1 เฟส

เมื่อป้อนสัญญาณพัลส์กระตุ้นเข้าที่เฟส P1 ทำให้เกิดขั้วแม่เหล็กใต้ขึ้น จึงเกิดแรงผลักขั้วแม่เหล็กใต้ของเพื่อให้พบกับขั้วแม่เหล็กเหนือของโรเตอร์ ในจังหวะนั้นเองแกนหมุนของมอเตอร์จะเกิดการเคลื่อนที่เปลี่ยนตำแหน่งไป 1 สเต็ป เมื่อขั้วแม่เหล็กใต้ของสเตเตอร์พบกับขั้วแม่เหล็กเหนือของโรเตอร์จะเกิดแรงแม่เหล็กดูดกัน ทำให้แกนหมุนหยุดนิ่ง ถ้าสังเกตต่อไปที่ขั้วของสเตอร์ของขดลวดในเฟสที่เหลือจะพบว่ามันเหลื่อมกัน ทำให้แรงแม่เหล็กเกิดการหักล้างกัน

จากนั้นป้อนสัญญาณเข้าที่เฟส P2 ทำให้เกิดขั้วแม่เหล็กเหนือขึ้นที่สเตเตอร์นั้น ทำให้เกิดแรงผลักอีก 1 จังหวะ ส่งผลให้แกนหมุนของมอเตอร์เคลื่อนที่ต่อเนื่องไปอีก 1 สเต็ป เมื่อขั้วแม่เหล็กเหนือของสเตเตอร์พบกับขั้วแม่เหล็กใต้ของโรเตอร์จะเกิดแรงแม่เหล็กดูดกัน ทำให้แกนหมุนหยุดนิ่ง และจะเป็นเช่นนี้ไปตลอดหากมีการป้อนสัญญาณกระตุ้นไล่ตามลำดับมายัง P3 และ P4 แล้ววนกลับไปที่ P1 อีก แกนหมุนของมอเตอร์ก็จะเกิดการเคลื่อนที่เปลี่ยนมุมไปอย่างต่อเนื่องจนครบ 1 เมื่อเคลื่อนที่ครบ 24 สเต็ปหากสเต็ปเปอร์มอเตอร์มีความละเอียดของการหมุนมากขึ้น เช่น 7.5 องศาต่อสเต็ป จำนวนสเต็ปที่ต้องการใน 1 รอบจะเพิ่มเป็น 48 สเต็ป และสูงถึง 200 สเต็ปหากมอเตอร์มีความละเอียด 1.8 องศาต่อสเต็ป



รูปที่ 2.26 รูปสัญญาณการไดรฟ์มอเตอร์ในแบบต่างๆ

การสั่งงานควบคุมการหมุนของสเต็ปมอเตอร์

การควบคุมและสั่งงานให้สเต็ปมอเตอร์ทำงาน ไปทีละสเต็ปสามารถทำได้โดยการจ่ายกำลังไฟไปยังขดลวด ในแต่ละขอบนสเตเตอร์ โดยการป้อนจะทำในลักษณะเป็นลำดับหรือเรียกว่า ซีเควนเวียลในลูปที่ถูกต้อง ซึ่งจะแบบ ได้เป็น 3 รูปแแบ คือ แบบเวฟ(wave) แบบ 2 เฟส(2 phase) และแบบครึ่งสเต็ป (half step) ทั้ง 3 แบบนี้ก็จะมีข้อดีและข้อเสียต่างกันออกไป

แบบเวฟหรือฟูลสเต็ป 1 เฟส

จะเป็นการกระตุ้นแบบที่ง่ายที่สุด ซึ่งจะทำการกระตุ้นขดลวดทีละขดในเวลาหนึ่งๆเรียงกันไป ตัวอย่างเช่น ขดที่ 1 , 2 , 3 , 4 ,1, 2 , 3 , 4 เป็นลำดับอย่างนี้ หรือ ขด 1 , 4 , 3 , 2 , 1 , 4 , 3 , 2 เป็นลำดับกันไป ทั้งนี้ขึ้นอยู่กับทิศทางที่เราต้องให้มอเตอร์หมุนไป วงจรที่นำมากระตุ้นนั้นจะมีราคาค่อนข้างจะถูกกว่าและง่ายกว่า ดังในรูปของวงจรการจ่ายไฟ ที่อยู่ด้านบนนั้น

เราสามารถเขียนขั้นตอนการทำงานเป็นตารางออกมาได้ดังนี้

Step No.	Phase 1	Phase 2	Phase 3	Phase 4
1	ON			
2		ON		
3			ON	
4				ON
5	ON			
6		ON		

ตารางที่ 2.1 การจ่ายสัญญาณแบบเวฟ

แบบ 2 เฟส (2 Phase)

แบบนี้ก็จะคล้ายกับการกระตุ้นในแบบเวฟแต่จะต่างกันตรงที่ แบบ 2 เฟส จะกระตุ้นทีละ 2 ขดที่อยู่ใกล้กันใน เวลาเดียวกัน และจะเรียงลำดับกันไป ดังเช่นแบบเดียวกับแบบเวฟ จะยกตัวอย่างการกระตุ้นขดลวดในลักษณะ ชีเควนให้เดูดังนี้ 12 ,23 ,34 ,41 ,12 ,23 ,34 ,41 เรียงลำดับกันไปเรื่อยๆ หรือจะเป็น 14, 43, 32, 21, 14, 43, 32, 21 เรียงกันไปเรื่อยๆเช่นกัน ถ้าจะมากล่าวถึงข้อดีข้อเสียของแบบ 2 เฟส แล้วมีดังนี้

ข้อดี การที่เราจะเพิ่มจำนวนขดลวดที่ถูกกระตุ้นจะทำให้แรงบิดได้มากกว่า แบบเวฟ ซึ่งโรเตอร์จะหมุนด้วยแรง ดึงแบบเต็มๆแรงจาก ทั้ง 2 ขดลวดที่กระตุ้นพร้อมกัน

ข้อเสีย แบบ 2 เฟส จะกระตุ้นขดลวดนั้นต้องใช้กำลังไฟมากขึ้นเป็น 2 เท่าของแบบเวฟ เราสามารถเขียนลำดับการกระตุ้นของขดลวดแบบ 2 เฟส ได้ดังในตารางต่อไปนี้

Step No.	Phase 1	Phase 2	Phase 3	Phase 4
1	ON	ON		
2		ON	ON	
3			ON	ON
4	ON			ON
5	ON	ON		
6		ON	ON	

ตารางที่ 2.2 การจ่ายสัญญาณแบบ 2 เฟส

แบบครึ่งสเต็ป

แบบนี้แบบรูปแบบผสมผสานของการกระตุ้นระหว่าง แบบเวฟ กับ แบบ 2 เฟส เพื่อให้จำนวนรอบของสเต็ปให้ มากขึ้นเป็น 2 เท่า ซึ่งในระบบนี้จะทำการกระตุ้นขดลวดเรียงกันไปเรื่อยๆเป็นลำดับ ดังจะยกตัวอย่างต่อไปนี้ 1, 12, 2, 23, 3, 34, 4, 41, 1, 12, 2, 23, 3, 34, 4, 41, 1 เป็นลำดับอยู่อย่างนี้เรื่อยไป ถ้าจะกลับทิศทางการหมุนก็จะได้เป็นดังนี้ 1, 41, 4, 43, 3, 32, 2, 21, 1, 41, 4, 43, 3, 32, 2, 21, 1 เป็นลำดับ

ข้อดี การกระตุ้นแบบนี้จะให้แรงบิดที่เพิ่มมากขึ้น เนื่องจากช่วงสเต็ปที่มีระยะสั้นลงอีกประการหนึ่งแต่ละ สเต็ปเกิดแรงดึงจากขดลวด 2 ขดที่ถูกกระตุ้นพร้อมกันเป็นผลให้ค่าตำแหน่งความถูกต้องมากขึ้นไปด้วย

ข้อเสีย ต้องจ่ายกำลังไฟเป็น 2 เท่าของแบบเวฟหรือจ่ายกำลังไฟเท่ากับแบบ 2เฟส ดังนั้นสามารถนำลำดับการทำงานของ แบบครึ่งเฟส ในรูปของตารางได้ดังนี้

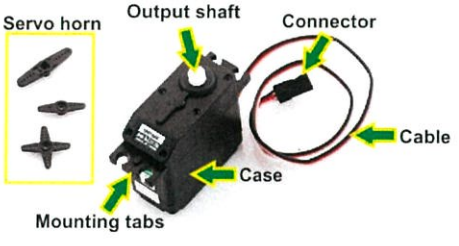
Step No.	Phase 1	Phase 2	Phase 3	Phase 4
1	ON			
2	ON	ON		
3		ON		
4		ON	ON	
5			ON	
6			ON	ON
7				ON
8	ON			ON
9	ON			
10	ON	ON		

ตารางที่ 2.3 การจ่ายสัญญาณแบบครึ่งสเต็ป

2.4 Servo Motor

Servo เป็นคำศัพท์ที่ใช้กันทั่วไปในระบบควบคุมอัตโนมัติ มาจากภาษาละตินคำว่า Sevus หมายถึง “ทาส” (Slave) ในเชิงความหมายของ Servo Motor ก็คือ Motor ที่เราสามารถสั่งงานหรือตั้งค่า แล้วตัว Motor จะหมุนไปยังตำแหน่งองศาที่เราสั่งได้เองอย่างถูกต้อง โดยใช้การควบคุมแบบป้อนกลับ (Feedback Control) ในบทความนี้จะกล่าวถึง RC Servo Motor ซึ่งนิยมนำมาใช้ในเครื่องเล่นที่บังคับด้วยคลื่นวิทยุ (RC = Radio - Controlled) เช่น เรือบังคับวิทยุ รถบังคับวิทยุ เฮลิคอปเตอร์บังคับวิทยุ เป็นต้น

Feedback Control คือ ระบบควบคุมที่มีการวัดค่าเอาต์พุตของระบบนำมาเปรียบเทียบกับค่าอินพุตเพื่อควบคุมและปรับแต่งให้ค่าเอาต์พุตของระบบให้มีค่า เท่ากับ หรือ ใกล้เคียงกับค่าอินพุต ส่วนประกอบภายนอก RC Servo Motor



รูปที่ 2.27 ส่วนประกอบ Servo motor

- Case ตัวถัง หรือ กรอบของตัว Servo Motor
- Mounting Tab ส่วนจับยึดตัว Servo กับชิ้นงาน
- Output Shaft เพลาส่งกำลัง
- Servo Horns ส่วนเชื่อมต่อกับ Output shaft เพื่อสร้างกลไกล
- Cable สายเชื่อมต่อเพื่อ จ่ายไฟฟ้า และ ควบคุม Servo Motor จะประกอบด้วยสายไฟ 3 เส้น และ ใน RC Servo Motor จะมีสีของสายแตกต่างกันไปดังนี้

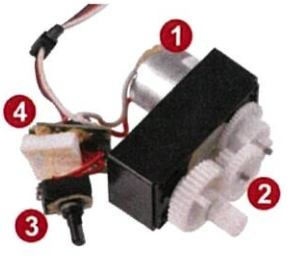
  สายสีแดง คือ ไฟเลี้ยง (4.8-6V)

  สายสีดำ หรือ น้ำตาล คือ กราวด์

  สายสีเหลือง (ส้ม ขาว หรือฟ้า) คือ สายส่งสัญญาณพัลซ์ควบคุม (3-5V)

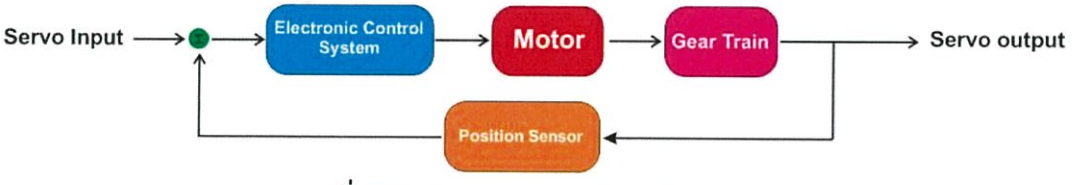
- Connector จุดเชื่อมต่อสายไฟ

ส่วนประกอบภายใน RC Servo Motor



รูปที่ 2.28 ลักษณะภายในของ Servo Motor

- Motor เป็นส่วนของตัวมอเตอร์
- Gear Train หรือ Gearbox เป็นชุดเกียร์ทดแรง
- Position Sensor เป็นเซ็นเซอร์ตรวจจับตำแหน่งเพื่อหาค่าองศาในการหมุน
- Electronic Control System เป็นส่วนที่ควบคุมและประมวลผล

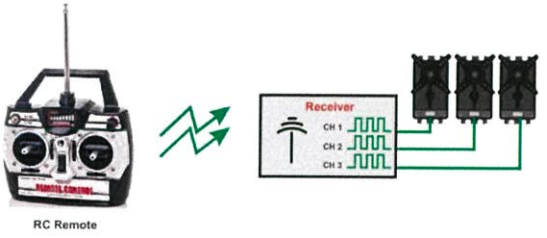


รูปที่ 2.29 Servo Motor Block Diagram

หลักการทำงานขอRC Servo Motor

เมื่อจ่ายสัญญาณพัลซ์เข้ามายัง RC Servo Motor ส่วนวงจรควบคุม (Electronic Control System) ภายใน Servo จะทำการอ่านและประมวลผลค่าความกว้างของสัญญาณพัลซ์ที่ส่งเข้ามาเพื่อแปลค่าเป็นตำแหน่งองศาที่ต้องการให้ Motor หมุนเคลื่อนที่ไปยังตำแหน่งนั้น แล้วส่งคำสั่งไปทำการควบคุมให้ Motor หมุนไปยังตำแหน่งที่ต้องการ โดยมี Position Sensor เป็นตัวเซ็นเซอร์คอยวัดค่ามุมที่ Motor กำลังหมุน เป็น Feedback กลับมาให้วงจรควบคุมเปรียบเทียบกับค่าอินพุตเพื่อควบคุมให้ได้ตำแหน่งที่ต้องการอย่างถูกต้องแม่นยำ

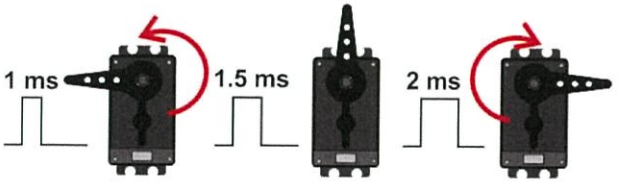
สัญญาณ RC ในรูปแบบ PWM



รูปที่ 2.30 ภาพแสดงการส่งสัญญาณแบบจำลอง

ภาครับจะแปลงความถี่วิทยุออกมาในรูปแบบสัญญาณ PWM (Pulse Width Modulation) มุมหรือองศาจะขึ้นอยู่กับความกว้างของสัญญาณพัลซ์ ซึ่งโดยส่วนมากความกว้างของพัลซ์ที่ใช้ใน RC Servo Motor จะอยู่ในช่วง 1-2 ms หรือ 0.5-2.5 ms

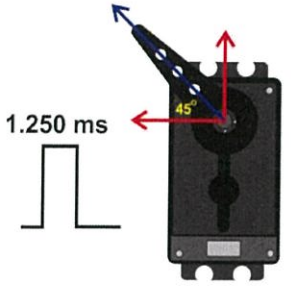
ยกตัวอย่างเช่นหากกำหนดความกว้างของสัญญาณพัลซ์ไว้ที่ 1 ms ตัว Servo Motor จะหมุนไปทางด้ายซ้ายจนสุด ในทางกลับกันหากกำหนดความกว้างของสัญญาณพัลซ์ไว้ที่ 2 ms ตัว Servo Motor จะหมุนไปยังตำแหน่งขวาสุด แต่หากกำหนดความกว้างของสัญญาณพัลซ์ไว้ที่ 1.5 ms ตัว Servo Motor ก็จะหมุนมาอยู่ที่ตำแหน่งตรงกลางพอดี



รูปที่ 2.31 การหมุนของ Servo motor ตามความกว้างของสัญญาณ

ดังนั้นสามารถกำหนดองศาการหมุนของ RC Servo Motor ได้โดยการเทียบค่า เช่น RC Servo Motor สามารถหมุนได้ 180 องศา โดยที่ 0 องศาใช้ความกว้างพัลซ์เท่ากับ 1000 us ที่ 180 องศาความกว้างพัลซ์เท่ากับ 2000 us เพราะฉะนั้นค่าที่เปลี่ยนไป 1 องศาจะใช้ความกว้างพัลซ์ต่างกัน (2000-1000)/180 เท่ากับ 5.55 us

จากการหาค่าความกว้างพัลซ์ที่มุม 1 องศาข้างต้น หากต้องกำหนดให้ RC Servo Motor หมุนไปที่มุม 45 องศาจะหาค่าพัลซ์ที่ต้องการได้จาก 5.55 x 45 เท่ากับ 249.75 us แต่ที่มุม 0 องศาเราเริ่มที่ความกว้างพัลซ์ 1ms หรือ 1000 us เพราะฉะนั้นความกว้างพัลซ์ที่ใช้กำหนดให้ RC Servo Motor หมุนไปที่ 45 องศา คือ 1000 + 249.75 เท่ากับประมาณ 1250 us



รูปที่ 2.32 การหมุน 45 องศาของ Servo motor

2.5 Arduino Board

Arduino เป็นบอร์ดไมโครคอนโทรเลอร์ตระกูล AVR ที่มีการพัฒนาแบบ Open Source คือมีการเปิดเผยข้อมูลทั้งด้าน Hardware และ Software ตัว บอร์ด Arduino ถูกออกแบบมาให้ใช้งานได้ง่าย ดังนั้นจึงเหมาะสำหรับผู้เริ่มต้นศึกษา ทั้งนี้ผู้ใช้งานยังสามารถดัดแปลง เพิ่มเติม พัฒนาต่อยอดทั้งตัวบอร์ดหรือโปรแกรมต่อได้อีกด้วย

ความง่ายของบอร์ด Arduino ในการต่ออุปกรณ์เสริมต่างๆ คือผู้ใช้งานสามารถต่อวงจรอิเล็กทรอนิคส์จากภายนอกแล้วเชื่อมต่อเข้ามาที่ขา I/O ของบอร์ดหรือเพื่อความสะดวกสามารถเลือกต่อกับบอร์ดเสริม (Arduino Shield) ประเภทต่างๆ เช่น Arduino XBee Shield, Arduino Music Shield, Arduino Relay Shield, Arduino Wireless Shield, Arduino GPRS Shield เป็นต้น มาเสียบกับบอร์ดบนบอร์ด Arduino แล้วเขียนโปรแกรมพัฒนาต่อได้เลย

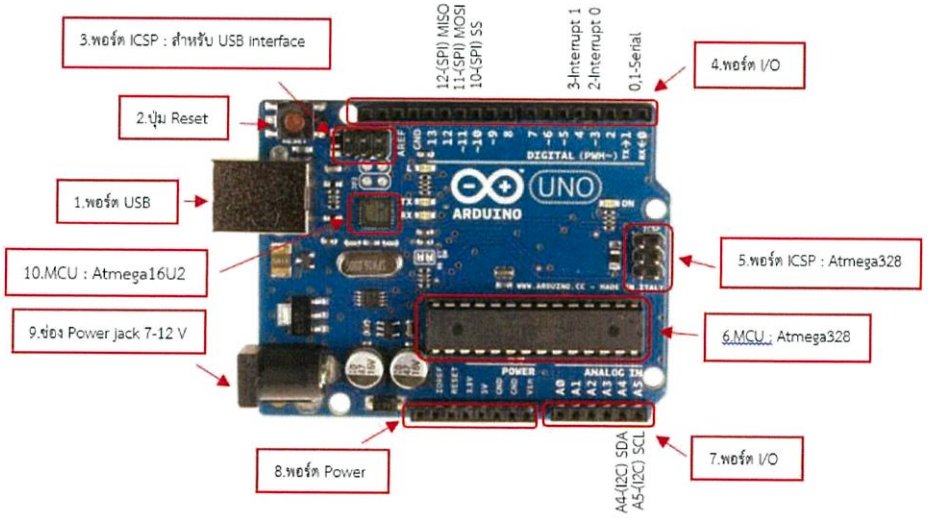
Auduino นั้นได้ใช้ชิป AVR เป็นหลักใน Auduino แทบรุ่น สาเหตุมาจากไมโครคอนโทรเลอร์ของตระกูล AVR นั้นมีความทันสมัย ในชิปในบางตัวสามารถเชื่อมต่อผ่าน USB ได้โดยตรง สามารถใช้กับคอมพิวเตอร์สมัยใหม่ได้เป็นอย่างดี และในไมโครคอนโทรเลอร์ตะกูล AVR ยังมีส่วนของโปรแกรมพิเศษที่เรียกว่า Bootloader อยู่ในระดับล่างกว่าส่วนโปรแกรมปกติ ซึ่งจะเป็นส่วนโปรแกรมที่จะถูกเรียกขึ้นมาก่อนการเรียกโปรแกรมปกติ ทำให้สามารถเขียนสั่งให้ทำงานใดๆก็ได้ ก่อนการเรียกโปรแกรมปกติ ทำให้ Arduino นั้นอาศัยส่วนโปรแกรมแกรมพิเศษนี้ในการทำให้ชิปสามารถโปรแกรมผ่านพอร์ตอนุกรมชนิด UART ได้ จึงทำให้การเขียนโปรแกรมลงไปในชิปใช้เพียง USB to UART ก็เพียงพอแล้ว แต่การโปรแกรมด้วยการใช้โปรโตคอล UART

ในโปรเจคนี้ใช้ Arduino Uno R3 เป็นหลัก โดย Uno R3 เป็นบอร์ด Arduino ที่ได้รับความนิยมมากที่สุด เนื่องจากราคาไม่แพง ส่วนใหญ่โปรเจคและ Library ต่างๆ ที่พัฒนาขึ้นมา Support จะอ้างอิงกับบอร์ดนี้เป็นหลัก และข้อดีอีกอย่างคือ กรณีที่ MCU เสีย ผู้ใช้งานสามารถซื้อมาเปลี่ยนเองได้ง่าย สามารถเขียนโปรแกรมบนซอฟท์แวร์ Arduino IDE และโปรแกรมผ่านพอร์ต USB

Features ต่างๆของ Arduino Uno R3

- ATmega328 microcontroller
- Input voltage - 7-12V
- 14 Digital I/O Pins (6 PWM outputs)
- 6 Analog Inputs
- 32KB Flash Memory (0.5KB for boot loader)
- 2KB SRAM
- 1KB EEPROM
- 16Mhz Clock Speed

Layout & Pin out Arduino Board (Model: Arduino UNO R3)



รูปที่ 2.33 แสดงภาพของ Arduino UNO R3

1. USBPort: ใช้สำหรับต่อกับ Computer เพื่ออับโหลดโปรแกรมเข้า MCU และจ่ายไฟให้กับบอร์ด
2. Reset Button: เป็นปุ่ม Reset ใช้กดเมื่อต้องการให้ MCU เริ่มการทำงานใหม่
3. ICSP Port ของ Atmega16U2 เป็นพอร์ตที่ใช้โปรแกรม Visual Com port บน Atmega16U2
4. I/OPort:Digital I/O ตั้งแต่ขา D0 ถึง D13 นอกจากนี้ บาง Pin จะทำหน้าที่อื่นๆ เพิ่มเติมด้วย เช่น Pin0,1 เป็นขา Tx,Rx Serial, Pin3,5,6,9,10 และ 11 เป็นขา PWM

5. ICSP Port: Atmega328 เป็นพอร์ตที่ใช้โปรแกรม Bootloader

6. MCU: Atmega328 เป็น MCU ที่ใช้บนบอร์ด Arduino

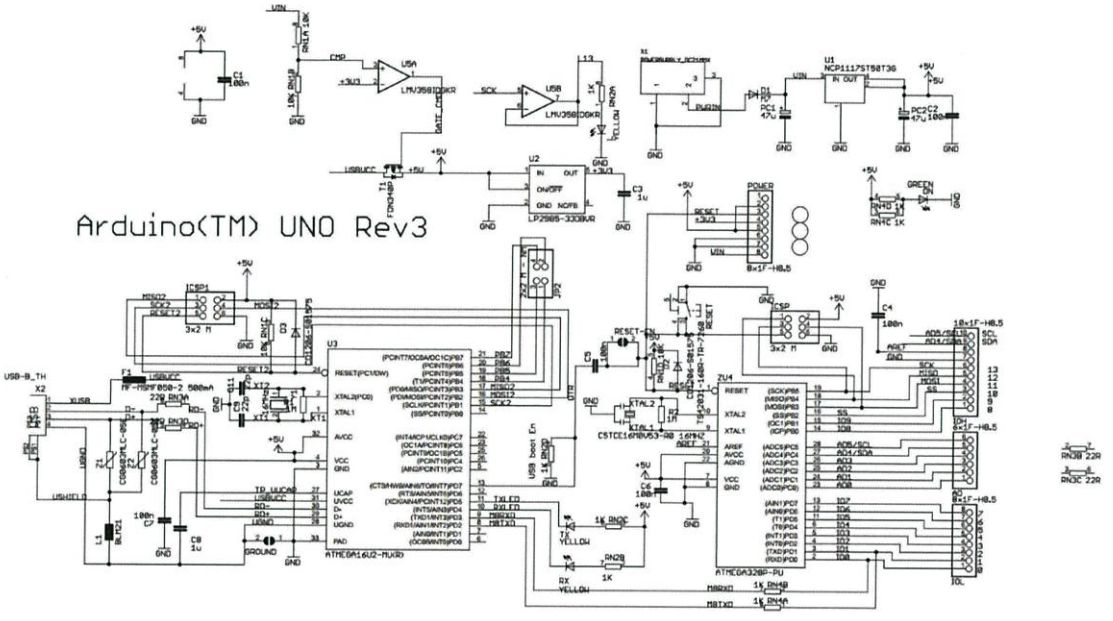
7. I/OPort: นอกจากจะเป็น Digital I/O แล้ว ยังเปลี่ยนเป็น ช่องรับสัญญาณอนาล็อก ตั้งแต่ขา A0-A5

8. Power Port: ไฟเลี้ยงของบอร์ดเมื่อต้องการจ่ายไฟให้กับวงจรภายนอก ประกอบด้วยขาไฟเลี้ยง +3.3 V, +5V, GND, Vin

9. Power Jack: รับไฟจาก Adapter โดยที่แรงดันอยู่ระหว่าง 7-12 V

10. MCU ของ Atmega16U2 เป็น MCU ที่ทำหน้าที่เป็น USB to Serial โดย Atmega328 จะติดต่อกับ Computer ผ่าน Atmega16U2

Arduino standalone board



รูปที่ 2.34 ภาพ Schematic ของ Arduino R3

2.6 การเขียน Code บน Arduino

โครงสร้างโปรแกรมภาษา C บน Arduino จะมีลักษณะแบบเดียวกับ C ทั่วๆไป

2.6.1 ปรีโปรเซสเซอร์ไดเร็กทีฟ (Preprocessor directives)

โดยปกติแล้วเกือบทุกโปรแกรมต้องมี โดยส่วนนี้จะเป็นส่วนที่คอมไพลเลอร์จะมีการประมวลผลและทำตามคำสั่งก่อนที่จะมีการคอมไพล์โปรแกรม ซึ่งจะเริ่มต้นด้วยเครื่องหมายไดเร็คทีฟ (directive) หรือเครื่องหมายสี่เหลี่ยม # แล้วจึงตามด้วยชื่อคำสั่งที่ต้องการเรียกใช้ หรือกำหนด โดยปกติแล้วส่วนนี้จะอยู่ในส่วนบนสุด หรือส่วนหัวของโปรแกรม และต้องอยู่นอกฟังก์ชั่นหลักใดๆก็ตาม

#include เป็นคำสั่งที่ใช้อ้างอิงไฟล์ภายนอก เพื่อเรียกใช้ฟังก์ชั่น หรือตัวแปรที่มีการสร้างหรือกำหนดไว้ในไฟล์นั้น รูปแบบการใช้งานคือ

#include <ชื่อไฟล์.h>

ตัวอย่างเช่น

#include <Servo.h>

#include <Stepper.h>

จากตัวอย่าง จะเห็นว่าได้มีการอ้างอิงไฟล์ Stepper.h และไฟล์ Servo.h ซึ่งเป็นไลบารี่พื้นฐานที่มีอยู่ใน Arduino ทำให้เราสามารถใช้ฟังก์ชั่นเกี่ยวกับเวลาที่ไลบารี่ Stepper มีการสร้างไว้ให้ใช้งานได้

การอ้างอิงไฟล์จากภายใน หรือการอ้างอิงไฟล์ไลบารี่ที่มีอยู่แล้วใน Arduino หรือเป็นไลบารี่ที่เพิ่มเข้าไปเอง จะใช้เครื่องหมาย <> ในการคร่อมชื่อไฟล์ไว้ เพื่อให้โปรแกรมคอมไพลเลอร์เข้าใจว่าควรไปหาไฟล์เหล่านี้จากในโฟลเดอร์ไลบารี่ แต่หากต้องการอ้างอิงไฟล์ที่อยู่ในโฟลเดอร์โปรเจค จะต้องใช้เครื่องหมาย "" คร่อมแทน ซึ่งคอมไพล์เลอร์จะวิ่งไปหาไฟล์นี้โดยอ้างอิงจากไฟล์โปรแกรมที่คอมไพล์เลอร์อยู่

เช่น

#include "myFunction.h"

จากตัวอย่างด้านบน คอมไพล์เลอร์จะไปหาไฟล์ myFunction.h ภายในโฟลเดอร์โปรเจคทันที หากไม่พบก็จะแจ้งเป็นข้อผิดพลาดออกมา

#define เป็นคำสั่งที่ใช้ในการแทนข้อความที่กำหนดไว้ ด้วยข้อความที่กำหนดไว้ ซึ่งการใช้คำสั่งนี้ ข้อดี คือ ไม่มีการอ้างอิงกับตัวโปรแกรมเลย

รูปแบบ

#define NAME VALUE

ตัวอย่างเช่น

```
#define LEDPIN 13
```

จากตัวอย่าง ไม่ว่าคำว่า LEDPIN จะอยู่ส่วนใดของโค้ดโปรแกรมก็ตาม คอมไพล์เลอร์จะแทนคำว่า LEDPIN ด้วยเลข 13 แทน ซึ่งข้อดีคือไม่ต้องสร้างเป็นตัวแปรขึ้นมาเพื่อเปลืองพื้นที่แรม และยังช่วยให้โปรแกรมทำงานเร็วขึ้นอีกด้วยเพราะซีพียูไม่ต้องไปขอข้อมูลมาจากแรมหลายๆทอด

2.6.2 ส่วนของการกำหนดค่า (Global declarations)

ส่วนนี้จะเป็นส่วนที่ใช้ในการกำหนดชนิดตัวแปรแบบนอกฟังก์ชั่น หรือประกาศฟังก์ชั่น เพื่อให้ฟังก์ชั่นที่ประกาศสามารถกำหนด หรือเรียกใช้ได้จากทุกส่วนของโปรแกรม

เช่น

```
int pin = 13;
void blink(void) ;
```

2.6.3. ฟังก์ชั่น setup() และฟังก์ชั่น loop()

ฟังก์ชั่น setup() และฟังก์ชั่น loop() เป็นคำสั่งที่ถูกบังคับให้ต้องมีในทุกโปรแกรม โดยฟังก์ชั่น setup() จะเป็นฟังก์ชั่นแรกที่ถูกเรียกใช้ นิยมใช้กำหนดค่า หรือเริ่มต้นใช้งานไลบารี่ต่างๆ เช่น ในฟังก์ชั่น setup() จะมีคำสั่ง pinMode() เพื่อกำหนดให้ขาใดๆก็ตามเป็นดิจิตอลอินพุต หรือเอาต์พุต ส่วนฟังก์ชั่น loop() จะเป็นฟังก์ชั่นที่ทำงานหลังจากฟังก์ชั่น setup() ได้ทำงานเสร็จสิ้นไปแล้ว และมีการวนรอบแบบไม่รู้จบ เมื่อฟังก์ชั่น loop() งานครบตามคำสั่งแล้ว ฟังก์ชั่น loop() ก็จะถูกเรียกขึ้นมาใช้อีก ตัวอย่าง

```
int pin = 13;
void setup() {
  pinMode(pin, OUTPUT);
}
void loop() {
  digitalWrite(pin, HIGH);
  delay(1000);
  digitalWrite(pin, LOW);
  delay(1000);
}
```

จากตัวอย่าง จะเห็นว่ามีการประกาศตัวแปรแบบนอกฟังก์ชั่น ทำให้สามารถกำหนด หรือเรียกใช้จากในฟังก์ชั่นใดๆก็ตามได้ ในฟังก์ชั่น setup() ได้มีการกำหนดให้ขาที่ 13 เป็นดิจิตอลเอาต์พุต และในฟังก์ชั่น loop() มีการกำหนดให้พอร์ต 13 มีลอจิกเป็น 1 และใช้ฟังก์ชั่น delay() ในการหน่วงเวลา 1

วินาที แล้วจึงกำหนดให้พอร์ต 13 มีสถานะลอจิกเป็น 0 แล้วจึงหน่วงเวลา 1 วินาที จบฟังก์ชั่น loop() และจะเริ่มทำฟังก์ชั่น loop() ใหม่ ผลที่ได้คือไฟกระพริบบนบอร์ด Arduino Uno ในพอร์ตที่ 13 ทำงานแบบไม่รู้จบ

ในทุกๆการทำงานของฟังก์ชั่น จะต้องเริ่มด้วยการกำหนดค่าที่ส่งกลับ ตามด้วยชื่อฟังก์ชั่น แล้วตามด้วยเครื่องหมายปีกกาเปิด { และจบด้วยเครื่องหมายปีกกาปิด } ภายในฟังก์ชั่น หากจะเรียกฟังก์ชั่นใช้งานย่อยใดๆ จะต้องมีเครื่องหมายเซมิโคล่อน ; ต่อท้ายเสมอ

การกำหนดชนิดค่าที่ส่งกลับเป็น void หมายถึงไม่มีการส่งค่ากลับ แต่สามารถใช้คำสั่ง return; ตรงๆได้ เพื่อให้จบการทำงานของฟังก์ชั่นก่อนจะไปถึงบรรทัดสุดท้ายของฟังก์ชั่น

2.6.4 การสร้างฟังก์ชั่น และการใช้งานฟังก์ชั่น (Users-defined function)

ในการสร้างฟังก์ชั่นขึ้นมา คำสั่งต่างๆที่อยู่ภายในฟังก์ชั่น ต้องอยู่ภายใต้เครื่องหมายปีกกาเปิด { และปีกกาปิด } เท่านั้น ภายใต้เครื่องหมาย {} เราสามารถนำฟังก์ชั่นหรือคำสั่งใดๆก็ได้มาใส่ไว้ แต่จะต้องคั่นแต่ละคำสั่งด้วยเครื่องหมายเซมิโคล่อน ; โดยจะนำคำสั่งทั้งหมดไว้บรรทัดเดียวกันเลย หรือแยกบรรทัดกันก็ได้เพื่อความสวยงามของโค้ด (ไม่มีผลกับขนาดของโปรแกรมหลังคอมไพล์)

ตัวอย่าง

```
void Mode(int pin) {
  pinMode(pin, OUTPUT);
}
void setup() {
  Mode(13);
}
```

2.6.5 ส่วนอธิบายโปรแกรม (Progarm comments)

ส่วนอธิบายโปรแกรม หรือการคอมเม้นโปรแกรมเป็นส่วนที่สำคัญอย่างมากที่จะช่วยให้ผู้ที่ไม่ได้เขียนโปรแกรม หรือเป็นผู้เขียนโปรแกรมเข้าใจโปรแกรมได้ง่ายขึ้นโดยอ่านจากคอมเม้น แทนการทำความเข้าใจโปรแกรมโดยอ่านแต่ละฟังก์ชั่น ส่วนอธิบายโปรแกรม หรือส่วนคอมเม้นนี้ จะไม่มีผลใดๆกับขนาดของโปรแกรมหลังคอมไพล์ เนื่องจากส่วนนี้จะถูกตัดทิ้งทั้งหมดเนื่องจากไม่ได้ถูกนำไปใช้งาน มีผลเพียงแค่ว่าไฟล์โค้ดโปรแกรมจะใหญ่ขึ้นมา หากมีการคอมเม้นโค้ดเยอะๆ แต่ขนาดก็จะเพิ่มขึ้นตามตัวอักษร ดังนั้นการคอมเม้นโค้ดจึงไม่คิดพื้นที่มากนัก แต่ผู้เขียนแนะนำให้คอมเม้นโค้ดให้สั้น และกระชับ เพื่อให้เกิดความรวดเร็วในการทำความเข้าใจ และไม่ยาวจนต้องเลื่อนสกอร์บาร์ไปทางขวาเพื่ออ่านคอมเม้นเพิ่มเติมอีก

การคอมเม้นโค้ดมีอยู่ 2 รูปแบบ คือเปิดด้วย /* และปิดด้วย */ เป็นการคอมเม้นโค้ดแบบข้ามบรรทัด คือตราบใดที่ยังไม่มี */ ตรงส่วนนั้นจะเป็นคอมเม้นทั้งหมด เช่น

```
/*
Hello world
10/4/2560
*/
void setup() { .... }
```

และแบบที่ 2 เป็นการคอมเม้นบรรทัดเดียว คือเปิดด้วยเครื่องหมาย // และปิดด้วยการขึ้นบรรทัดใหม่ เช่น

```
void setup() {
  pinMode(13, OUTPUT); // Set pin 13 to output
}
```

2.7 การสื่อสารข้อมูลแบบอนุกรม (Serial Communication)

การสื่อสารแบบอนุกรม (Serial Communication) เป็นการรับส่งออกมาทีละ Bit จึงมีความล่าช้ากว่าการสื่อสารแบบขนาน อย่างก็ไรตาม ตัวกลางสำหรับการสื่อสารอาจจะใช้สายส่งเพียงคู่เดียว จึงมีค่าใช้จ่ายน้อยกว่าและทำให้อุปกรณ์มีขนาดเล็ก นอกจากนี้ การสื่อสารแบบอนุกรมสามารถสื่อสารแบบเครือข่ายได้ จึงเป็นที่นิยมมากกว่า

2.7.1 การสื่อสารแบบอนุกรม

การสื่อสารข้อมูลแบบอนุกรม เป็นการรับส่งข้อมูลทีละบิต แทนที่จะทำการรับส่งข้อมูลพร้อมกันทุกบิตในเวลาเดียวกัน ข้อดีของการสื่อสารแบบนี้คือ ใช้จำนวนสายในการสื่อสารน้อย สามารถรับส่งได้ในระยะ ทางที่ไกล ๆ แต่ก็มีข้อเสียในด้านเวลา เพราะต้องใช้เวลาในการสื่อสารมาก เมื่อเทียบกับการสื่อสารแบบขนาน อีกทั้งโอกาสเกิดการผิดพลาดของข้อมูลก็สูงกว่าแบบขนาน

รูปแบบของการสื่อสาร

1. แบบซิมเพล็กซ์ (Simplex) เป็นการสื่อสารทางเดียว
2. แบบฮาล์ฟดูเพล็กซ์ (Half-duplex) เป็นการสื่อสารได้ทั้งสองทาง แต่จะต้องผลัดกันรับ-ส่ง
3. แบบฟูลดูเพล็กซ์ (Full-duplex) เป็นการสื่อสารได้ทั้งสองทางและทางได้ในเวลาเดียวกัน

รูปแบบการสื่อสารข้อมูลแบบอนุกรม

การสื่อสารแบบอนุกรมโดยทั่วไป มี 2 ลักษณะ

- การสื่อสารแบบซิงโครนัส (Synchronous)
- การสื่อสารแบบอะซิงโครนัส (Asynchronous)

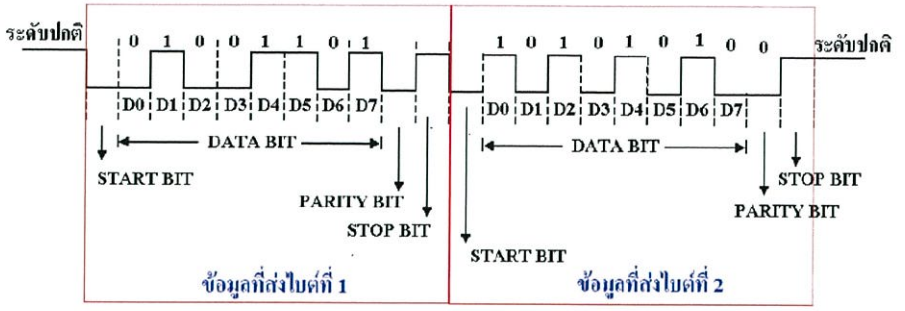
Asynchronous คือสามารถสื่อสารแบบ Full duplex กล่าวคือสามารถรับและส่งข้อมูลระหว่าง Receiver และ Transmitter ได้ในเวลาเดียวกัน นอกจากนี้ ไม่ต้องใช้สายสัญญาณ Clock เพื่อกำหนดจังหวะการรับส่งข้อมูล แต่มีการกำหนดรูปแบบ Format หรือ Protocol การรับส่งข้อมูลขึ้นมาแทน และอาศัยการกำหนดความเร็วของการรับส่งข้อมูลให้เท่ากัน

ตัวอย่างโปรโตคอลอะซิงโครนัส Serial Communication - Example Protocols

- Morse code
- RS-232 - Recommended Standard 232
- RS422, RS-423, RS-485
- I2C - Inter-Integrated Circuit
- SPI - Serial Peripheral Interface
- USB - Universal Serial Bus
- Firewire
- Ethernet
- Serial ATA - Serial Advanced TEchnology Attachment
- Serial Attach SCSI - Serial Attached Small Computer System
- Interface
- SONET - Synchronous Optical Network
- PCI Express - Peripheral Component Interconnect Express

การเชื่อมต่อแบบ Universal Asynchronous Receiver Transmitter

UART ย่อมาจากคำว่า Universal Asynchronous Receiver Transmitter เป็นการเชื่อมต่อและสื่อสารข้อมูลอนุกรมกับอุปกรณ์ต่างๆ เช่น คอมพิวเตอร์, RFID, GPS, GSM Module, Wifi Module เป็นต้น



รูปที่ 2.35 รูปแบบการสื่อสารแบบ Universal Asynchronous Receiver Transmitter

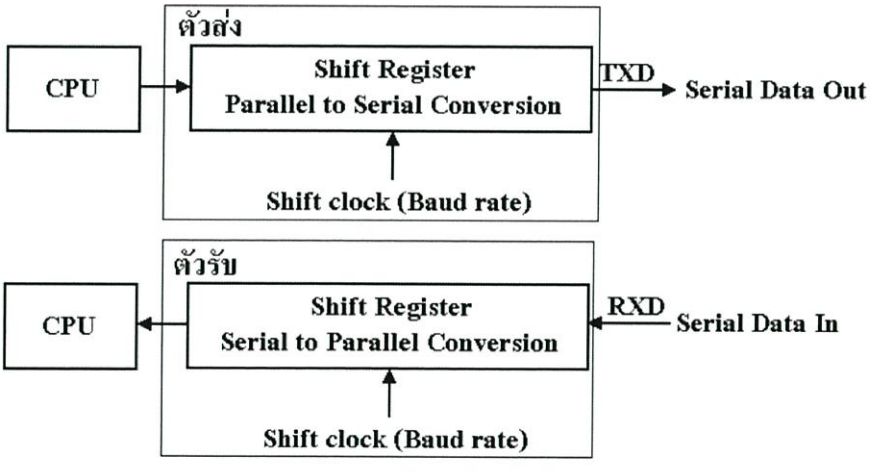
จากรูป 2.35 แสดงรูปแบบการสื่อสารของ UART โดยผู้ใช้ต้องกำหนดคุณสมบัติเหล่านี้ให้เหมือนกันทั้งฝั่ง Receiver และ Transmitter ซึ่งประกอบด้วย

- Start Bit บอกจุดเริ่มต้นข้อมูล มีขนาด 1 บิต เป็นสถานะ Low
- Data Bit ค่าข้อมูลมีได้ 5 ถึง 8 บิต
- Parity Bit บิตสำหรับใช้ตรวจสอบความผิดพลาดของข้อมูล มีได้ 0 ถึง 1 บิต
- Stop Bit บิตใช้บอกจุดสิ้นสุดข้อมูล มีได้ 1 1.5 และ 2 บิต

ความเร็วในการสื่อสาร

ความเร็วในการสื่อสาร หมายถึงจำนวนบิตที่ใช้รับส่งข้อมูลต่อวินาที โดยปกติจะมีค่าเท่ากับ 110 150 300 1200 2400 4800 9600 และ 19200 บิตต่อวินาที อัตราความเร็วนี้บางครั้งก็เรียกว่าอัตราบอด (Baud rate) ทั้งตัวส่งและตัวรับต้องกำหนดให้มีความเร็วในการสื่อสารเท่ากัน

หลักการรับ-ส่งข้อมูลแบบอนุกรม



รูปที่ 2.36 การรับส่งข้อมูลแบบอนุกรม

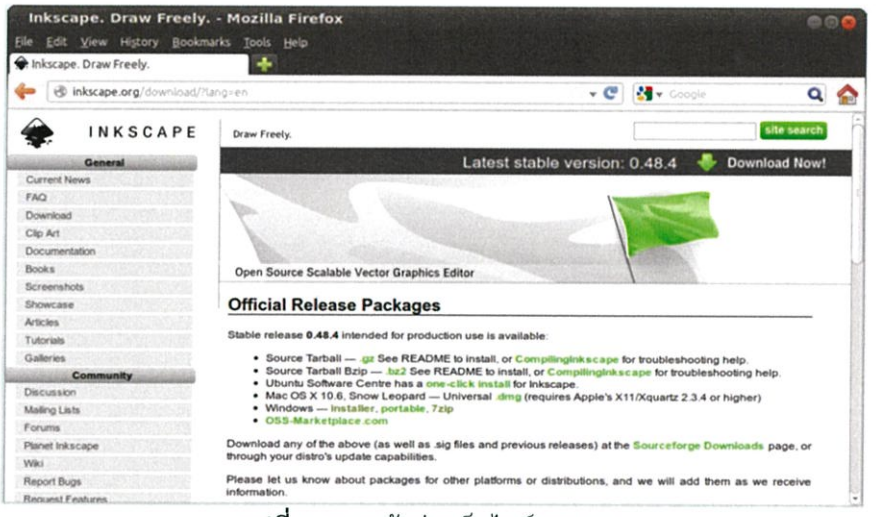
โดย การใช้งานของไมโครคอนโทรลเลอร์ AVR Atmega168 สามารถสื่อสารข้อมูลแบบอนุกรมได้โดยใช้โมดูล USART (Universal Synchronous and Asynchronous serial Receiver and Transmitter) ชื่อพอร์ทที่ใช้งาน คือ Receive (Rx) และ Transmit (Tx)

2.8 โปรแกรมควบคุมการทำงานไมโครคอนโทรลเลอร์

2.8.1 โปรแกรม Inkscape

Inkscape เป็นโปรแกรมวาดภาพแบบเว็กเตอร์ ในตระกูลเดียวกับ Adobe Illustrator หรือ Corel Draw ที่พบได้ทั่วไป แต่ Inkscape เป็นซอฟต์แวร์โอเพ่นซอร์สที่สามารถนำมาใช้งานได้ฟรี

Inkscape สามารถใช้ทำงานได้ตั้งแต่งานทั่วๆไป เช่นการทำผัง ไปจนถึงงานวาดเส้นที่ซับซ้อนระดับมืออาชีพที่ต้องการความประณีตสูง และ ถึงแม้ Inkscape จะเป็นโปรแกรมฟรี แต่ศักยภาพการทำงานนั้นไม่แพ้ซอฟต์แวร์เชิงพาณิชย์ราคาแพงๆทั้งหลาย ทั้งยังได้รับการยอมรับจากผู้ใช้งานทั่วโลกอีกเช่นกัน



รูปที่ 2.37 หน้าต่างเว็บไซต์ของ Inkscape

ความสามารถที่หลากหลายของ Inkscape นั้นในโปรเจคนี้ได้ใช้ในส่วนของการทำ G code จากไฟล์ภาพ .JPG ซึ่งเป็นไฟล์รูปภาพที่สามารถหาได้ทั่วไป

ภาพเวคเตอร์และบิตแมพ

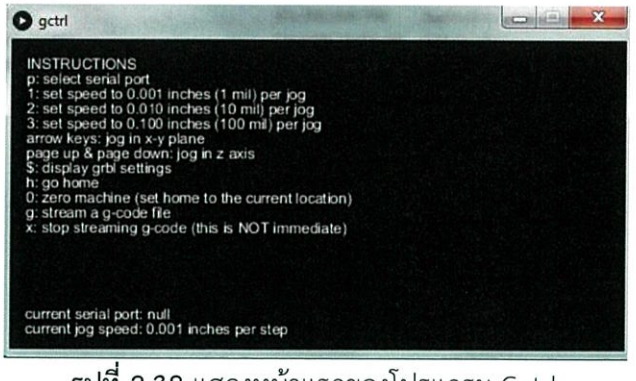
ภาพดิจิตอลหรือภาพที่สร้างด้วยโปรแกรมบนคอมพิวเตอร์ แบ่งเป็น 2 กลุ่มใหญ่ๆ ก็คือ ภาพบิตแมพ (Bitmap) และภาพเว็คเตอร์ (Vector)

ภาพบิตแมพ (Bitmap หรือ Raster) เป็นภาพที่ประกอบด้วยจุดเล็กๆ เรียงตัวกัน เรามักเรียกจุดเหล่านี้ว่า พิกเซล (Pixels) ไฟล์ภาพบิตแมพ เช่น ไฟล์นามสกุล .JPG, .BMP, .GIF หรือ .PNG เป็นต้น คุณภาพของภาพ ถูกกำหนดโดยจำนวน Resolution หรือ ความละเอียดของจำนวนพิกเซลต่อนิ้ว มีหน่วยเป็น PPI เช่น 72 PPI, 96 PPI (สำหรับใช้งานบน เว็บไซต์, จอคอมพิวเตอร์, ทีวี) หรือ 300 PPI, 600 PPI (สำหรับงานสิ่งพิมพ์)

ภาพเว็คเตอร์ (Vector) เป็นภาพที่ถูกสร้างด้วยการคำนวณทางคณิตศาสตร์ ข้อดีก็คือ สามารถขยายภาพมากเท่าไรก็ได้ โดยที่ภาพไม่แตก ไฟล์ภาพแบบเว็คเตอร์ เช่นไฟล์นามสกุล .SVG, .EPS, .PS, .AI, หรือ .CDR เป็นต้น การนำภาพแบบเว็คเตอร์ไปใช้งานมักจะต้องแปลงไปเป็นภาพบิตแมพก่อน เพราะโปรแกรมดูภาพ, โปรแกรมเปิดเว็บไซต์ต่างๆมักจะไม่รองรับภาพเว็คเตอร์

2.8.2 โปรแกรม Gctrl

Grbl เป็น หนึ่งใน โปรแกรมประเภท CAM (Computer Aided Manufacturing) ซึ่งแปลเป็นภาษาไทยว่า คอมพิวเตอร์ช่วยในการผลิต เป็นการนำคอมพิวเตอร์มาช่วยในการสร้างรหัสจี (G-code) เพื่อควบคุมเครื่องจักรซีเอ็นซีในการกัดขึ้นรูปชิ้นส่วน โดยใช้ข้อมูลทางรูปร่างจาก CAD จากความก้าวหน้าของเทคโนโลยี ITCAM สามารถใช้ข้อมูลจาก CAD ในการกำหนดว่าจะใช้เครื่องจักรใดในการผลิต วัสดุชิ้นงานมีขนาดเท่าใด วางตาแหน่งอ้างอิงอย่างไร ใช้เครื่องมืออะไรในการตัดเฉือน จะใช้วิธีตัดเฉือนแบบไหนกี่ขั้นตอน รวมไปถึงการจำลองขั้นตอนการทางานเพื่อดูเส้นทางการตัด เฉือนของเครื่องมือตัดเฉือน และตรวจสอบความผิดพลาดในการผลิต โดยโปรแกรม Grbl นั้นเป็นโปรแกรม Open source ที่ใช้อย่างแพร่หลายในการใช้ร่วมกับเครื่อง CNC มีรูปแบบ หน้าตาของโปรแกรมหลากหลาย ขึ้นอยู่กับผู้พัฒนา โดย โปรแกรมที่นำมาใช้นี้มีชื่อว่า Gctrl



รูปที่ 2.38 แสดงหน้าแรกของโปรแกรม Gctrl

2.8.3 คำสั่งที่ใช้สั่งการไมโครคอนโทรลเลอร์ (G code)

สำหรับการป้อนคำสั่งให้กับไมโครคอนโทรลเลอร์ในเครื่องจักร CNC นั้นจะถูกเรียกว่าเป็นการป้อนผ่านโปรแกรม NC โดย จะมีลักษณะเหมือนโปรแกรมคอมพิวเตอร์ทั่วไป ประกอบด้วยหลายบรรทัด ในแต่ละบรรทัดจะประกอบไปด้วยคำสั่งต่างๆ สำหรับ NC โปรแกรมมีศัพท์เรียกเฉพาะเมื่อเทียบกับโปรแกรมภาษาคอมพิวเตอร์ทั่วไป ดังนี้

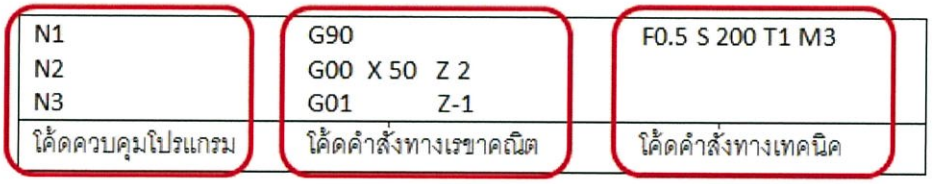
	ภาษาคอมพิวเตอร์ทั่วไป	NC โปรแกรม
1	บรรทัด (Line)	บล็อก (Block)
2	คำสั่ง	เวิร์ด (Word)

ตารางที่ 2.4 เปรียบเทียบภาษาคอมและ NC โปรแกรม

โดย ในแต่ละบล็อกประกอบด้วยหลายเวิร์ด ในแต่ละเวิร์ดประกอบด้วยหนึ่งตัวอักษรภาษาอังกฤษ หรือ เรียกว่า “โค้ด (Code)”ซึ่งเป็นคำสั่งให้เครื่องจักรกล CNC ทำงานในลักษณะที่ต้องการ แล้วตามด้วยตัวเลขสำหรับประกอบการสั่งการหรือการทำงานนั้นๆ

โค้ดต่างๆที่ใช้ใน NC โปรแกรม สามารถแยกได้เป็น 3 ประเภท คือ

1. โค้ดคำสั่งการควบคุมโปรแกรม (Program Control Instructions)
2. โค้ดคำสั่งทางเรขาคณิต (Geometric Instructions)
3. โค้ดคำสั่งทางเทคนิค (Technical Instructions)



รูปที่ 2.39 โค้ดประเภทต่างๆ ในโปรแกรม NC

โดย คำสั่งการควบคุมโปรแกรมเกี่ยวข้องกับการกำหนดลำดับขั้นตอนการควบคุมการทำงานของเครื่องจักร CNC ส่วนคำสั่งทางเรขาคณิตเกี่ยวข้องกับการเคลื่อนที่ของเครื่องจักรเพื่อให้ได้รูปทรงเรขาคณิตของชิ้นงานที่ต้องการ (ได้แก่ โค้ดจี G code) และคำสั่งทางเทคนิคเป็นการควบคุมทางเทคนิคของการทำงานของเครื่องจักร เช่น ความเร็วฟีด ความเร็วสปินเดิล การเปิด-ปิดสปินเดิล และการเปลี่ยนทูล เป็นต้น

กลุ่มโค้ดหลักที่ใช้ในโปรแกรมเอ็นซี คือ โค้ดจี (G code) และ โค้ดเอ็ม (M code)

จีโค้ด (G code)

จีโค้ด เป็นคำสั่งที่ทำให้ระบบควบคุมหรือคอนโทรลเลอร์สั่งการให้เครื่องจักรกล CNC ทำการแมชชีนให้เป็นรูปทรงเรขาคณิตตามความต้องการ โดยในการแมชชีนใดๆ คอนโทรลเลอร์ต้องทราบทิศทางและตำแหน่งของการเคลื่อนที่ของทูล ลักษณะการเคลื่อนที่เป็นเส้นตรงหรือเส้นโค้งวงกลม หน่วยความยาวที่ใช้ และบอกตำแหน่งการเคลื่อนที่แบบ absolute หรือ increment เป็นต้น

จีโค้ด มีมาตรฐานอุตสาหกรรม เช่นมาตรฐาน ISO6983/BS 3635 มาตรฐาน ANSI/EIA RS-274D, (ประเทศสหรัฐอเมริกา), BS3635(ประเทศอังกฤษ) เป็นต้น โดยทุกมาตรฐานดังกล่าวมี จีโค้ดพื้นฐานที่เหมือนกัน NC โปรแกรมที่ใช้มาตรฐานเหล่านี้นิยมเรียกว่า G code program

G code พื้นฐาน	
โค้ด	คำสั่ง
G00	การเคลื่อนที่แนวเส้นตรงจากจุดหนึ่งไปยังอีกจุดหนึ่งด้วยความเร็วฟีดสูงสุดโดนไม่โดนชิ้นงาน หรือ Rapid run
G01	การเคลื่อนที่แนวเส้นตรงลึกเข้าไปในชิ้นงานด้วยความเร็วฟีดที่กำหนด
G02	การเคลื่อนที่แนวเส้นโค้งวงกลมลึกเข้าไปในชิ้นงานในทิศตามเข็มนาฬิกา
G03	การเคลื่อนที่แนวเส้นโค้งวงกลมลึกเข้าในเนื้อของชิ้นงานในทิศทวนเข็มนาฬิกา
G04	หยุดการเคลื่อนที่ในระยะเวลาที่กำหนด หรือ Dwell
G17	กำหนดใช้ระนาบ XY
G18	กำหนดใช้ระนาบ XZ
G19	กำหนดใช้ระนาบ YZ
G20/G70	กำหนดหน่วยความยาวเป็น นิ้ว Inch
G21/G71	กำหนดหน่วยความยาวเป็น มิลลิเมตร mm
G80	ยกเลิก Cycle ต่างๆ
G81-G83	ไซเคิลการเจาะรู Drill cycle
G84	ไซเคิลการทำเกลียว
G85-G88	ไซเคิลการคว้านรู Boring cycle
G90	กำหนดโปรแกรมให้เป็นแบบ Absolute
G91	กำหนดโปรแกรมให้เป็นแบบ Increment
G94	ให้ค่าฟีดเป็น มม/นาที (mm/min) หรือ นิ้ว/นาที (inch/min)
G95	ให้ค่าฟีดเป็น มม/รอบ (mm/rev) หรือ นิ้ว/รอบ (inch/rev)

G96	ให้ความเร็วผิว Surface speed คงที่เป็น (m/min)
G97	ให้สปินเดิลหมุนด้วยความเร็วรอบคงที่เป็น รอบ/นาที (rpm)
G98-G99	ไม่ได้ใช้ใน ISO6983 และ RS-274D

ตารางที่ 2.5 G code พื้นฐาน

นอกเหนือจาก G code แล้วยังประกอบด้วยโค้ดอื่นๆอีก 7 ประเภท ที่สามารถนำมาใช้งานร่วมกันกับ G code ได้ คือ

1. เลขที่บล็อก (เลขที่บรรทัด) : N
2. ตำแหน่งหรือระยะทางความยาว : X,Y,Z
3. ตำแหน่งจุดศูนย์กลางวงกลม : I,J,K
4. ความเร็วสปินเดิลและความเร็วตัด : S,V
5. ความเร็วฟีด : F
6. เลขที่ทูล : T
7. อื่นๆ : B,D,O

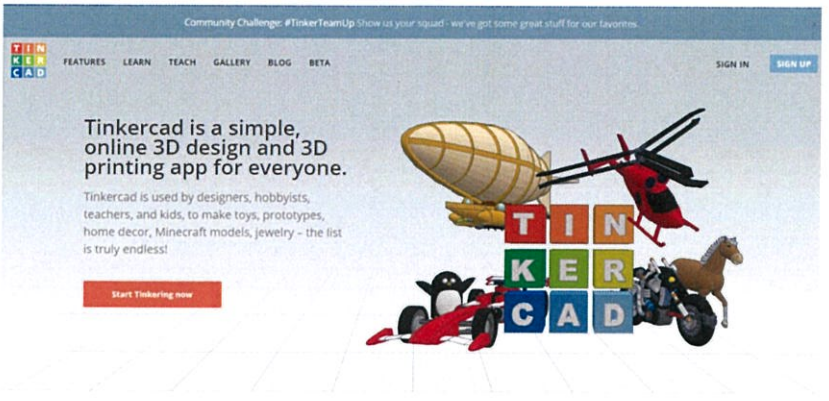
2.9 การสร้างชิ้นงานจาก 3D Printer

เนื่องจากการออกแบบส่วนของ Pen holder นั้นไม่สามารถทำด้วยแผ่นไม้ หรือแผ่นอคริลิคได้ จึงต้องนำการทำชิ้นงานจากเครื่อง 3D printer มาช่วยในการสร้างชิ้นงานที่มีรูปแบบเฉพาะ โดยการใช้งานเครื่อง 3D printer นั้นต้องการไฟล์โมเดล 3 มิติจากโปรแกรมออกแบบชิ้นงาน 3มิติ เช่น Autocad Solidwrok เป็นต้น

โปรแกรมสำหรับออกแบบโมเดล 3 มิติ มีมากมายหลายโปรแกรม มีทั้งแบบ open source และ non-open source ขึ้นอยู่กับว่าจะเอาไปใช้ออกแบบชิ้นงานประเภทไหน โปรแกรมออกแบบที่นิยมใช้มากตัวหนึ่งคือ Sketch Up ของ Google ที่ผู้ใช้ส่วนใหญ่ใช้ออกแบบบ้านหรือตึก ถ้าออกแบบแนววิศวกรรมจะใช้โปรแกรม Solidwork หรือ Autocad Inventor แต่ถ้าออกแบบแนวตัวละคร การ์ตูนหรือปั้นรูปเหมือนจะใช้โปรแกรม zbrush ซึ่งโปรแกรมส่วนใหญ่จะต้องเสียเงินในการซื้อ และต้องใช้เวลาเรียนรู้ในการใช้โปรแกรมในระดับหนึ่ง

โปรแกรมที่ชื่อว่า Tinkercad นั้นเป็นโปรแกรมออกแบบชิ้นงาน 3มิติ ที่ทำงานบน Web Browser ซึ่งผู้ใช้จำเป็นต้องต่อ Internet เพื่อใช้งาน ข้อดีของโปรแกรม Tinkercad นั้นคือเป็นแบบ open source แถมการใช้งานไม่ซับซ้อน สามารถออกแบบแล้วบันทึกเป็นไฟล์สำหรับใช้กับ เครื่องพิมพ์ 3D Printer ได้ ซึ่งโปรแกรม Tinkercade นี้อยู่ในเครือของ Autodesk ซึ่งเป็นเครือของโปรแกรมที่ใช้

สำหรับออกแบบในสาขางานต่างๆ เช่น Autocad จุดประสงค์ของโปรแกรม Tinkercad นั้นถูกออกแบบมาเพื่อการใช้งานที่ไม่ซับซ้อน เหมาะสำหรับผู้ที่เริ่มศึกษาการออกแบบโมเดล 3 มิติ



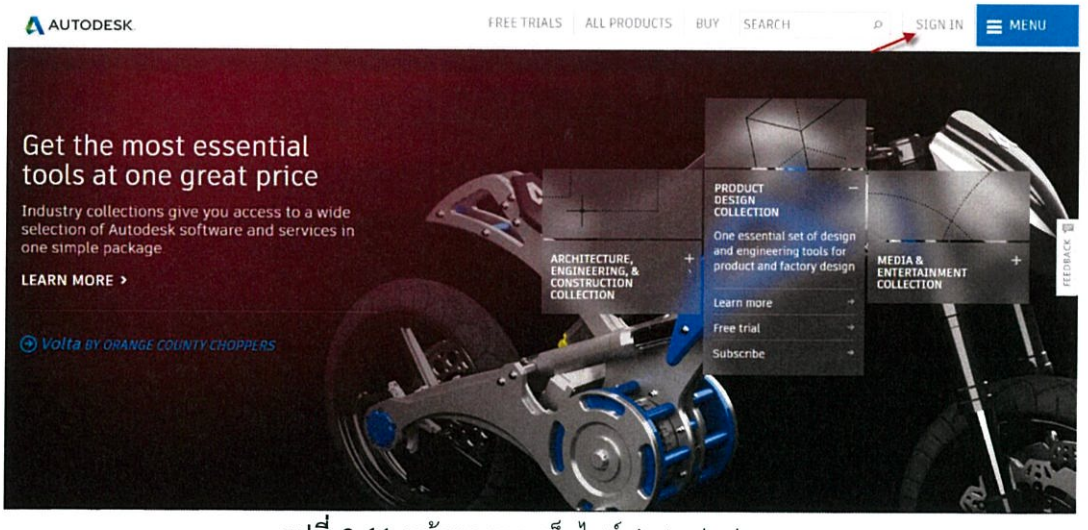
รูปที่ 2.40 หน้าแรกของเว็บไซต์ Tinkercad

2.9.1 การเริ่มต้นใช้งาน Tinkercad

อย่างที่ได้กล่าวไปในข้างต้นนั้น การใช้งาน Tinkercad ต้องทำการเชื่อมต่อกับ Internet และเข้าใช้งานผ่าน www.tinkercad.com การจะใช้งานได้นั้นต้องสมัครสมาชิกของ Autodesk ก่อน จากนั้นจึงจะสามารถใช้งาน Tinkercad ได้ โดยการเป็นสมาชิกของ Autodesk นั้นจะสามารถใช้งานโปรแกรมอื่นๆ ในเครือของ Autodesk ได้

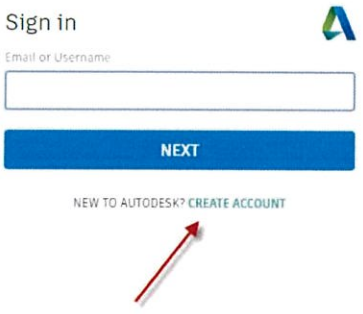
การสมัครสมาชิก Autodesk

1. เข้าไปที่ www.autodesk.com และ คลิ้กที่ Sign in



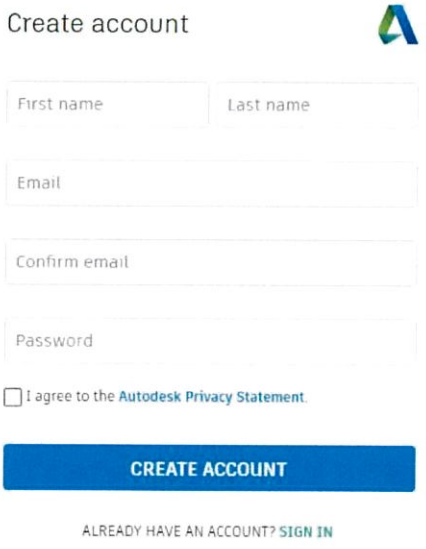
รูปที่ 2.41 หน้าแรกของเว็บไซต์ Autodesk

2. เลือก Create account



รูปที่ 2.42 หน้าแรกการ sign in

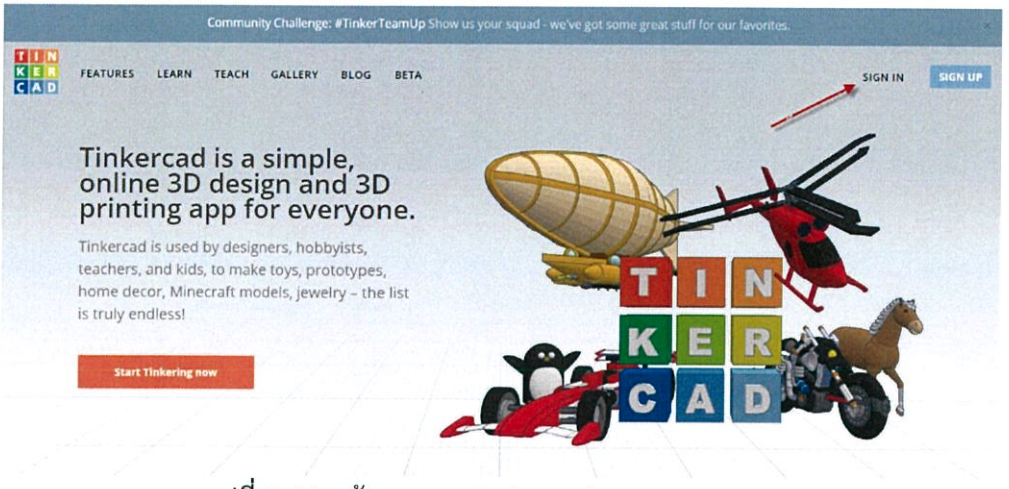
3. กรอกข้อมูล เสร็จสิ้นการสมัครสมาชิก Autodesk



รูปที่ 2.43 กรอกข้อมูลที่จำเป็นสำหรับการสมัคร

เมื่อเสร็จสิ้นการสมัครสมาชิกแล้ว การใช้งานโปรแกรมต่างๆของ Autodesk นั้นสามารถทำได้ทันที มีเพียงบางโปรแกรมที่ต้องทำซื้อก่อนจึงจะสามารถใช้งานได้ เช่น Autocad

4. เข้าไปที่ www.tinkercad.com จากนั้นทำการ sign in



รูปที่ 2.44 หน้าแรกของ Tinkercad และ การ Sign in

หลังจากทำการ Sign in แล้วจะสามารถเริ่มต้นใช้งานได้ทันที โดยการใช้ Tinkercad เพื่อออกแบบชิ้นงาน เมื่อออกแบบเสร็จเรียบร้อยแล้วสามารถเซฟให้เป็นไฟล์ .stl เพื่อใช้กับเครื่อง 3D printer ได้

บทที่ 3
การออกแบบและสร้างชิ้นงาน

คุณสมบัติของเครื่องร่างภาพ

- สั่งการส่วนเขียน ให้ร่างออกมาเป็นภาพตามพิกัดกัดภาพที่ถูกแปลงเข้าไป
- การสั่งการส่วนเขียนใช้ไมโครคอนโทรลเลอร์ควบคุมมอเตอร์ 3 ตัว เคลื่อนที่ตามแกน X, Y และ Z
- พิกัดภาพจะถูกแปลงผ่านทางซอฟต์แวร์คอมพิวเตอร์

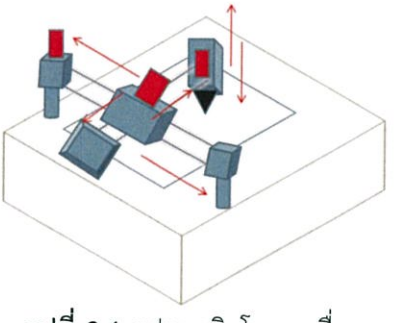
3.1 ส่วนประกอบของเครื่องร่างภาพขนาดเล็ก

เครื่องร่างภาพขนาดเล็กมีส่วนประกอบหลักอยู่ 3 ส่วน คือ

1. ส่วนโครงเครื่อง
2. ส่วนควบคุม
3. ส่วนแปลงรูปภาพเป็นพิกัดตัวเลข

3.1.1 ส่วนโครงเครื่อง

แนวคิดด้านการออกแบบโครงเครื่อง คือ ความต้องการจำกัดขนาดของเครื่องให้มีขนาดไม่ใหญ่มาก เนื่องจากแผ่นรองวาดที่ต้องการมีขนาดเพียงแค่ 15*21 ซม. หรือ เล็กกว่า ดังนั้นตัวเครื่องจึงถูกออกแบบให้มีลักษณะคล้ายเครื่องหมาย + ดังรูปที่ 3.1

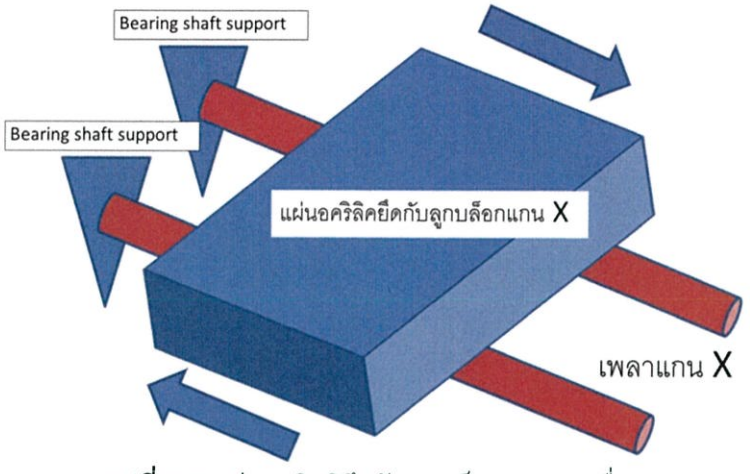


รูปที่ 3.1 รูปแนวคิดโครงเครื่อง

โดย โครงเครื่องแบ่งได้เป็น 4 ส่วน

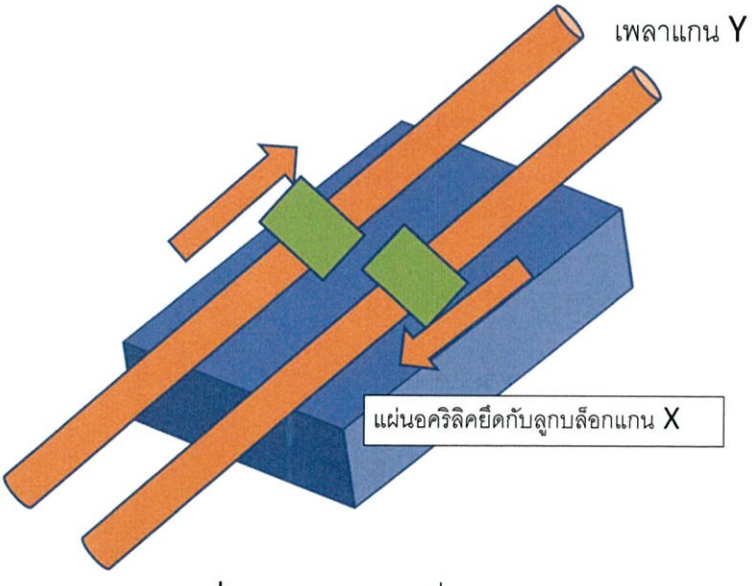
1. ส่วนฐานของเครื่อง ทำจากไม้ประกอบเป็นสี่เหลี่ยมผืนผ้า ขนาดประมาณ 30*35 ซม. เพื่อใช้วางแผ่นรองเขียนผิวเรียบขนาดไม่เกิน 15* 21 ซม. สำหรับวาดรูป

2. ส่วนของแกน X ใช้ Bearing shaft support ในการยึดรางสไลด์เพลากลมขนาดเส้นผ่านศูนย์กลาง 12 มม. ความยาวประมาน 30 ซม.จำนวน 2 เส้น ให้ติดกับฐาน และใช้แผ่นอคริลิคยึดกับลูกบล็อกเป็นส่วนที่เลื่อนตามแนวรางสไลด์ด้วย Stepper motor ดังรูปที่ 3.2



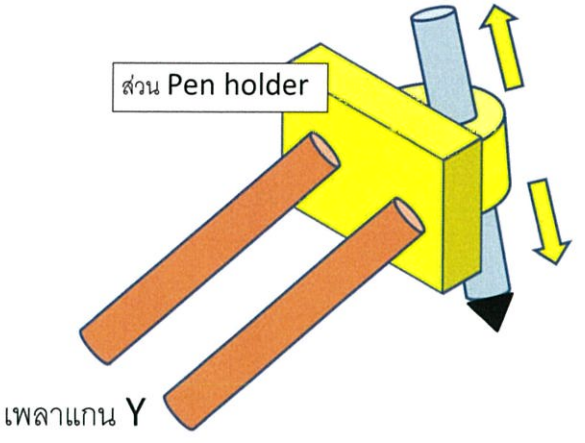
รูปที่ 3.2 แผ่นอคริลิคยึดกับลูกบล็อกแกน X เลื่อนตามแนวแกน

3. ส่วนของแกน Y ใช้แผ่นอคริลิคยึดกับลูกบล็อกแกน X เป็นฐานของแกน Y และทำให้รางสไลด์เพลากลมขนาดเส้นผ่านศูนย์กลาง 12 มม. ความยาวประมาน 40 ซม. จำนวน 2 เส้น เลื่อนตามแนวแกนด้วย Stepper motor แทน ดังรูปที่ 3.3



รูปที่ 3.3 เพลาแกน Y เลื่อนตามแนวแกน

4. ส่วนของแกน Z ใช้แผ่นอคริลิคทำเป็น Pen holder ใช้สำหรับจับปากกาเพื่อใช้เขียนรูป ยึดติดกับเพลาแกน Y และทำให้เลื่อนขึ้นลงตามแนวด้วย Servo motor ดังรูปที่ 3.4



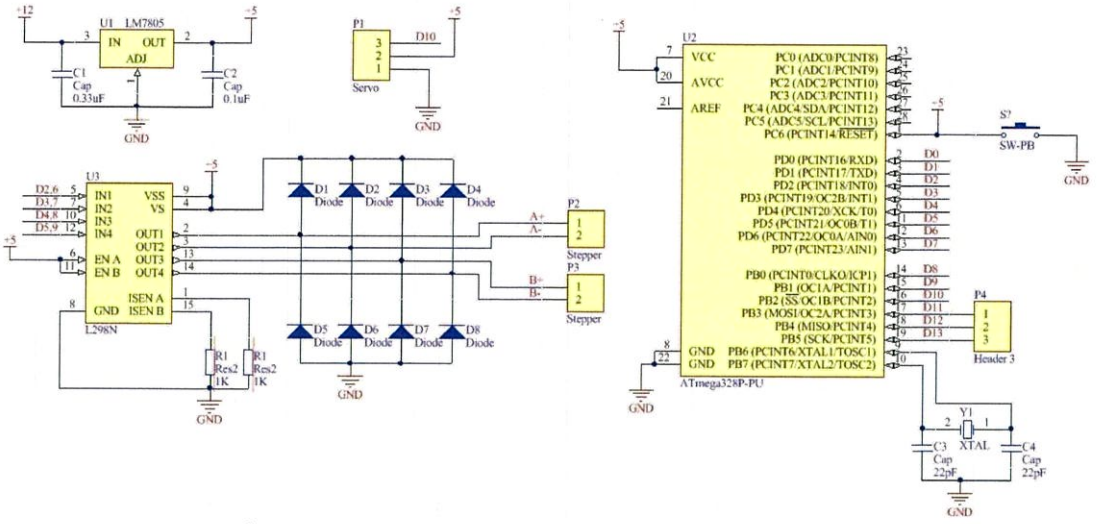
รูปที่ 3.4 ส่วน Pen holder เลื่อนตามแนวแกน

3.1.2 ส่วนควบคุม

การควบส่วนต่างๆของเครื่องร่างภาพ ต้องใช้มอเตอร์จำนวน 3 ตัว แบ่งเป็น Stepper motor จำนวน 2 ตัว ในการบังคับแกน X,Y และ Servo motor จำนวน 1 ตัว ในการบังคับ Pen holder แกน Z

การควบคุม Stepper motor ต้องใช้การใส่แรงดันให้ถูกจังหวะที่ควรจะเป็น Stepper motor จึงจะหมุนไปตามตำแหน่งที่ต้องการ โดยองศาการหมุนของ Stepper motor ขึ้นอยู่กับคุณสมบัติของ Stepper motor ตัวนั้นๆ ว่าสามารถหมุนได้กี่องศาต่อการจ่ายแรงดันแต่ละครั้ง แล้วจึงค่อยคำนวณหาระยะทางการเคลื่อนองศาการหมุนของมอเตอร์

ส่วนของ Servo motor จะทำงานตามสัญญาณพัลส์ที่มีคาบความกว้างอยู่ระหว่าง 1 มิลลิวินาที ถึง 2 มิลลิวินาที การส่งสัญญาณพัลส์ดังกล่าวมีผลให้มอเตอร์หมุน โดยทิศทางการหมุนขึ้นอยู่กับพัลส์บวก การคำนวณส่วนของระยะการเคลื่อนที่ของ Servo motor ต้องการเพียงแค่ให้ยกปากกาขึ้นเหนือกระดาษ แต่จะมากหรือน้อยขึ้นอยู่กับความสัมพันธ์ของการเคลื่อนที่แกน X และ Y หากยกนานไป แกนอื่นจะทำงานช้าลงตามไปด้วย

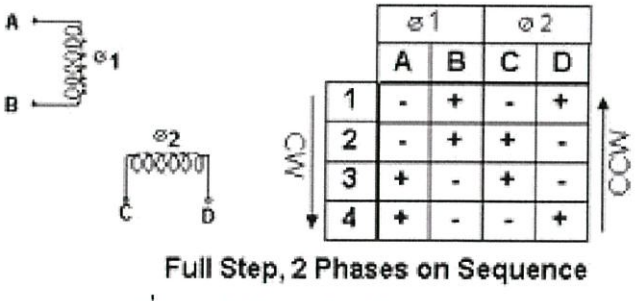


รูปที่ 3.5 วงจรการไดร์ฟ Stepper motor และ Servo Motor

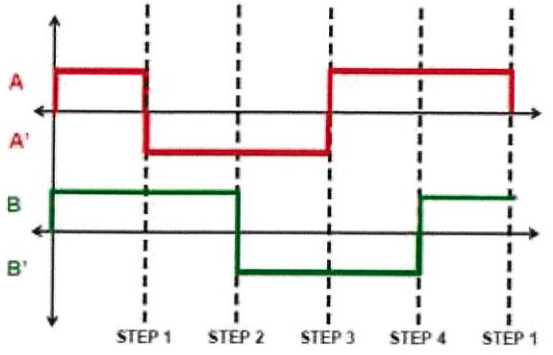
มอเตอร์ทั้ง 3 ที่ใช้ จะถูกควบคุมด้วย Microcontroller ควบคู่กับวงจรไดร์ฟกระแสมอเตอร์ จำนวน 2 ตัว โดยใช้กับคู่ Stepper motor ส่วนของ Servo motor สามารถสั่งผ่าน Microcontroller โดยตรงได้เลย

โปรแกรมการควบคุม Stepper motor เบื้องต้น

โปรแกรมการควบคุม Stepper motor ให้หมุนไปยังตำแหน่งต่างๆโดยสามารถที่จะกำหนดจำนวน Step และความเร็วที่จะหมุนได้ภายในตัวโปรแกรม โดย Stepper motor ที่ทำการนำมาไดร์ฟนั้น ในการหมุน 1 รอบจะมีทั้งหมด 200 Step โดยหมุนได้ Step ละ 1.8° โดยการไดรฟ์นั้นจะใช้การป้อนสัญญาณแบบ Full Step ไปยังตัวมอเตอร์ โดยการไดร์ฟนั้นจะเป็นไปตาม รูปที่ 3.6 โดยหากต้องการให้หมุนตามเข็มนาฬิกาก็ให้ป้อนสัญญาณตามลำดับจากลำดับ 1 ลงไปถึงลำดับ 4 แต่ถ้าหากต้องการให้หมุนทวนเข็มนาฬิกา ให้ป้อนสัญญาณ โดยนำลำดับที่ 4 ป้อนเข้าไปก่อนตามลำดับจนถึงลำดับที่ 1



รูปที่ 3.6 ลำดับการป้อนสัญญาณ



รูปที่ 3.7 รูปภาพสัญญาณในการป้อนให้มอเตอร์หมุนตามเข็มนาฬิกา

```
const int ENA = 7;
const int IN1 = 6;
const int IN2 = 5;
const int ENB = 8;
const int IN4 = 9;
const int IN3 = 10;
const int ledPin = 13;
```

กำหนดชื่อของแต่ละพอร์ตบน Board

```
void setup()
{
  pinMode(ENA,OUTPUT);
  pinMode(IN1,OUTPUT);
  pinMode(IN2,OUTPUT);
  pinMode(ENB,OUTPUT);
  pinMode(IN3,OUTPUT);
  pinMode(IN4,OUTPUT);
  pinMode(ledPin,OUTPUT);
  digitalWrite(ledPin, LOW);
```

ตั้งกำหนดค่าและกำหนด output ของและพอร์ตที่ตั้งชื่อไว้แล้ว

```
  //reverse(step,delay);
  reverse(400,10);
  //forward(step,delay);
  forward(400,10);
```

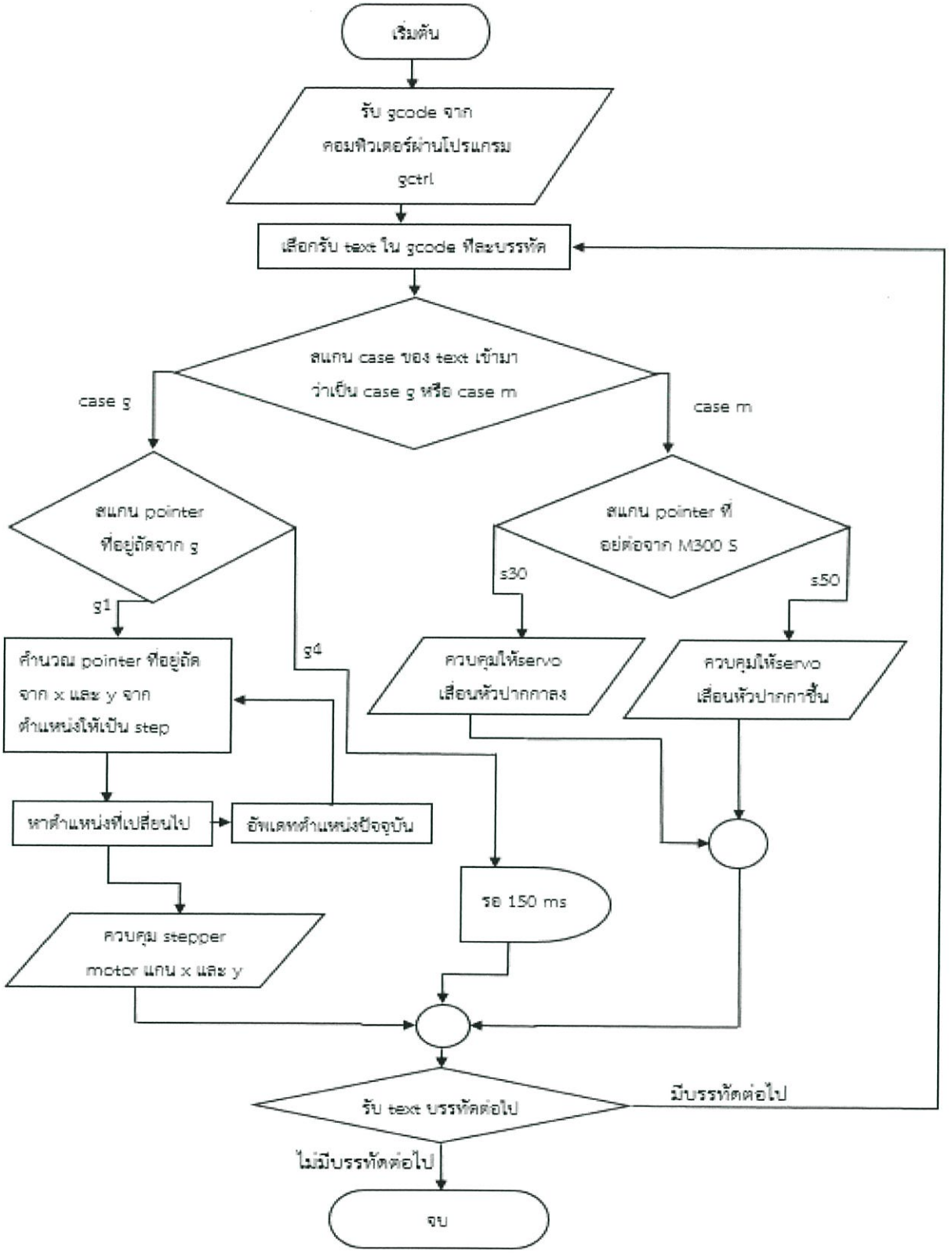
กำหนดความความเร็วและจำนวนสเต็ปการหมุน ยิ่ง Delay มีค่าน้อยยิ่งหมุนเร็ว

```
}

void loop()
{
}
void reverse(int i, int j) {
  while (1)  {
    digitalWrite(IN1, 0);
    digitalWrite(IN2, 1);
    digitalWrite(IN3, 0);
    digitalWrite(IN4, 1);
    delay(j);
    i--;
    if (i < 1) break;

    digitalWrite(IN1, 0);
    digitalWrite(IN2, 1);
    digitalWrite(IN3, 1);
    digitalWrite(IN4, 0);
    delay(j);
    i--;
    if (i < 1) break;

    digitalWrite(IN1, 1);
    digitalWrite(IN2, 0);
    digitalWrite(IN3, 1);
    digitalWrite(IN4, 0);
    delay(j);
    i--;
    if (i < 1) break;
```

```
    digitalWrite(IN1, 1);
    digitalWrite(IN2, 0);
    digitalWrite(IN3, 0);
    digitalWrite(IN4, 1);
    delay(j);
    i--;
    if (i < 1) break;
  }
}
```

จาก Code ด้านบน เป็นการกำหนดให้บอร์ดส่งสัญญาณออกเป็น Sequence แบบ Full step เพื่อให้เกิดการหมุนของมอเตอร์

Flowchart ของโปรแกรมใน Arduino UNO

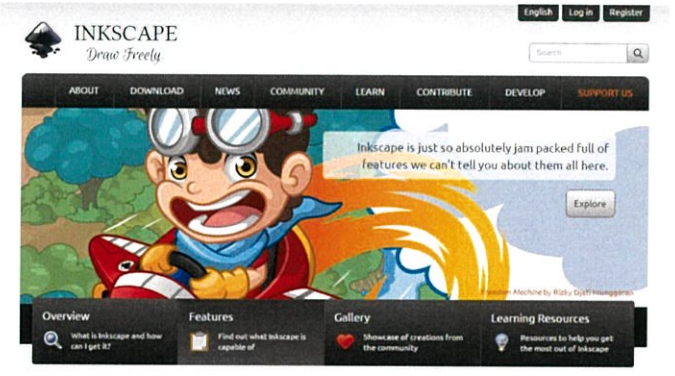


3.1.3 ส่วนแปลงรูปภาพเป็นพิกัดตัวเลข

การที่จะทำให้ส่วนควบคุม บังคับ Pen holder จับปากกาเขียนรูปที่ต้องการ ต้องอาศัยพิกัดตัวเลขในการกำหนดว่าจะให้มอเตอร์แต่ละตัวนั้น จะหมุนกี่องศา ฉะนั้นการใช้ซอฟต์แวร์คอมพิวเตอร์ Inkscape ที่ใช้แปลงรูปภาพ ให้กลายเป็น G-code จึงถูกนำมาใช้ควบคู่กับการสั่งการของ Microcontroller โดยการที่จะส่งข้อมูล G-code ได้นั้นต้องใช้โปรแกรมเสริมที่ชื่อว่า Grbl Controller ควบคู่กับโปรแกรม ArduinoIDE ถึงจะครบถ้วนกระบวนการทำงาน

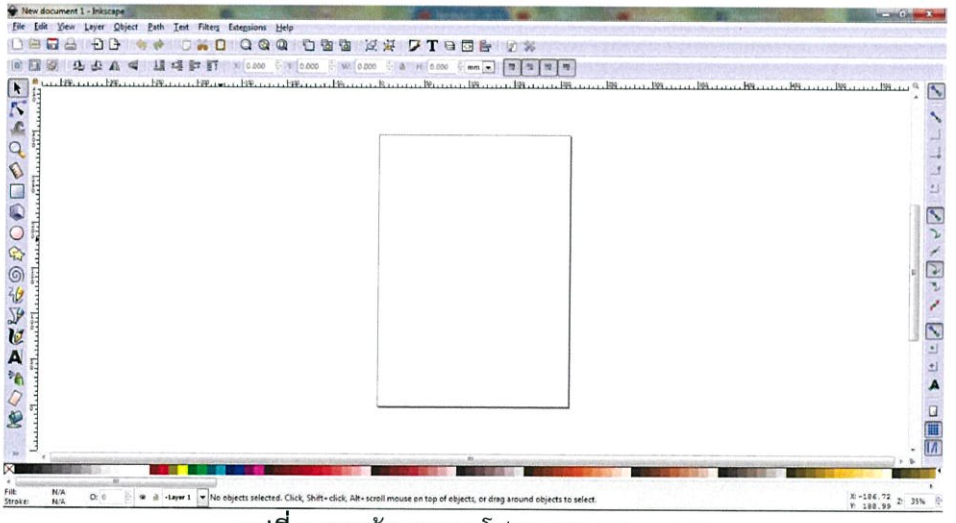
การใช้งานโปรแกรม Inkscapeเพื่อสร้าง G-code

โปรแกรม Inkscape เป็นโปรแกรม open source เพื่อที่ผู้ใช้สามารถทำสำเนา แจกจ่าย และแก้ไขปรับปรุงได้ ดาวน์โหลดได้จาก www.inkscape.org/en/ และทำการติดตั้งลงบนคอมพิวเตอร์ได้ทั้งระบบปฏิบัติการแบบ Windows, Linux และ Mac OS



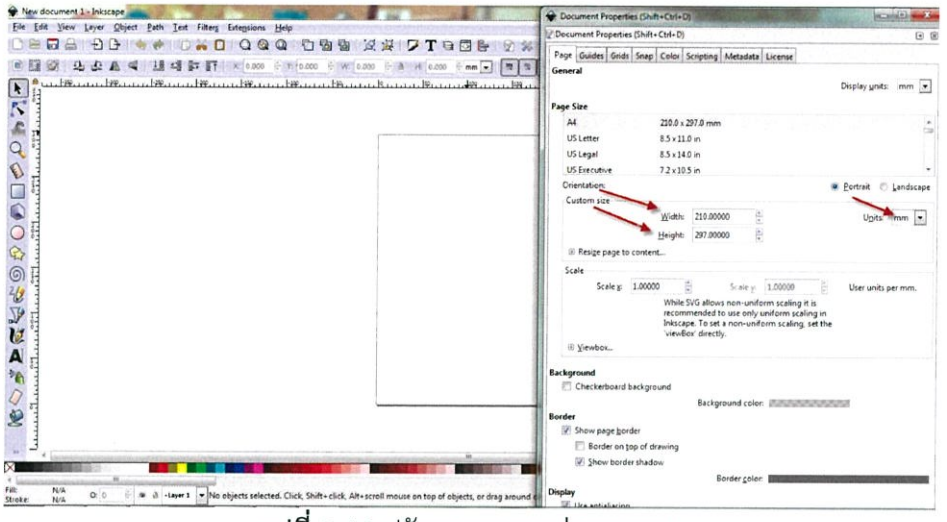
รูปที่ 3.8 หน้าเว็บไซต์ Inkscape

- เมื่อดาวน์โหลดและติดตั้งและเข้าโปรแกรมไปจะเจอหน้าแรก



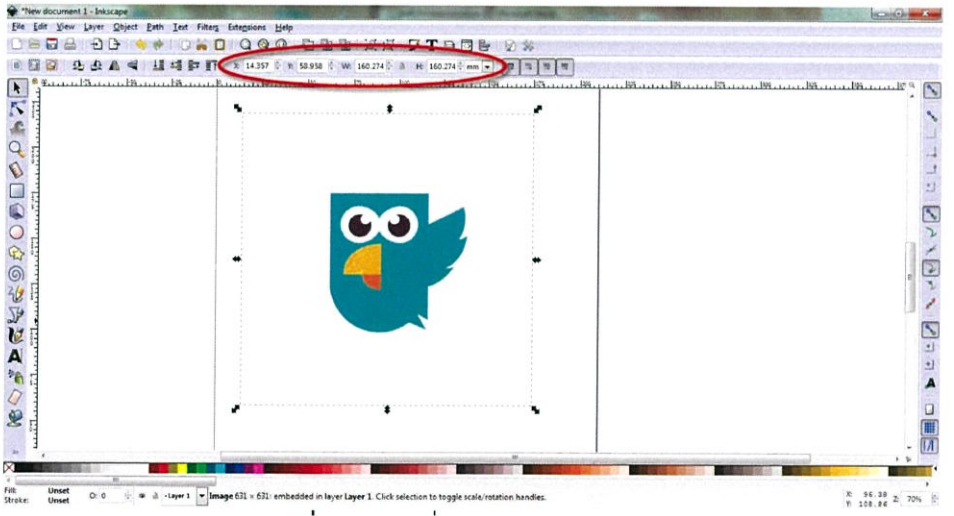
รูปที่ 3.9 หน้าแรกของโปรแกรม Inkscape

- ทำการปรับขนาดหน้ากระดาษ โดยไปที่ file -> Document Properties แล้วปรับความกว้าความยาวได้ตามต้องการ



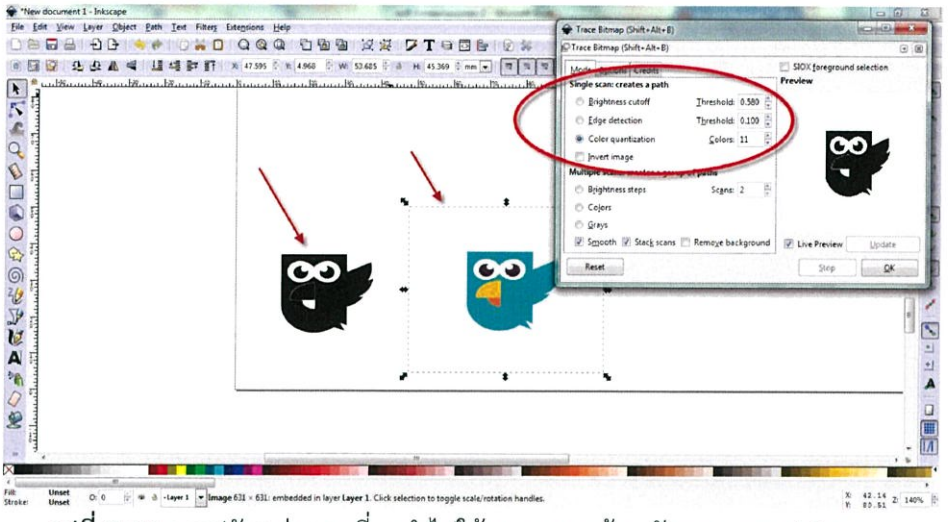
รูปที่ 3.10 ปรับขนาดของแผ่นกระดาษ

- ทำการ Import รูปที่ต้องการโดยไปที่ file -> Import -> รูปที่ต้องการ โดยสามารถปรับขนาดและหน่วยที่ใช้ได้ตามวงกลมสีแดง



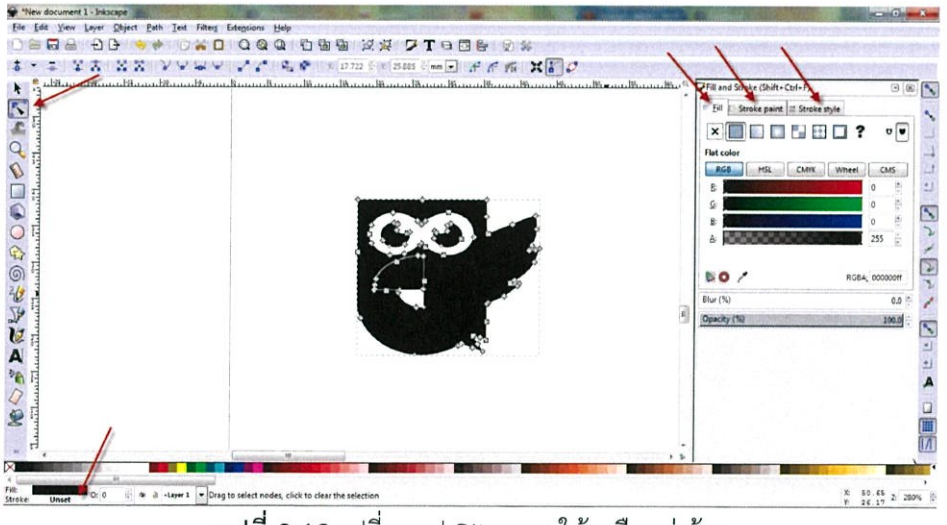
รูปที่ 3.11 รูปที่แสดงจากการ import

- เลือก Path -> Trace to bit map เลือก live preview จากนั้นปรับค่าตามที่วงกลมได้ตามต้องการ เมื่อกด OK จะได้ภาพ bit map ซ้อนกับภาพต้นฉบับ เราสามารถลบภาพต้นฉบับได้เลย



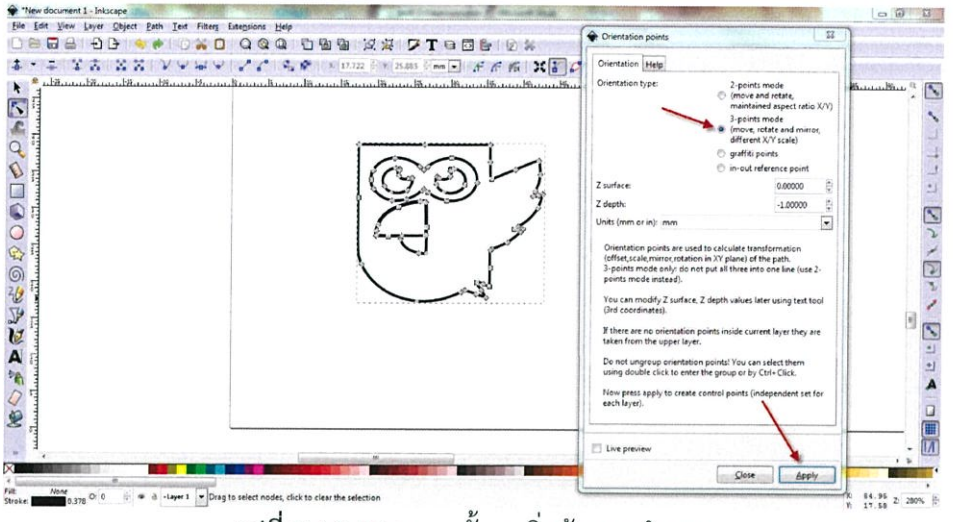
รูปที่ 3.12 การปรับแต่งภาพที่จะนำไปใช้ และ ภาพต้นฉบับและ ภาพ bit map

- กดที่ Edit path by notes แล้วตามด้วยคลิ้กที่ Unset บริเวณออพชั่น fill&stroke จากนั้นจะมีหน้าต่างขึ้นมาบริเวณด้านขวา เลือก no plaint ที่ Fill เลือก Flat color ที่ Stroke paint และปรับขนาดของเส้นที่ Stroke style



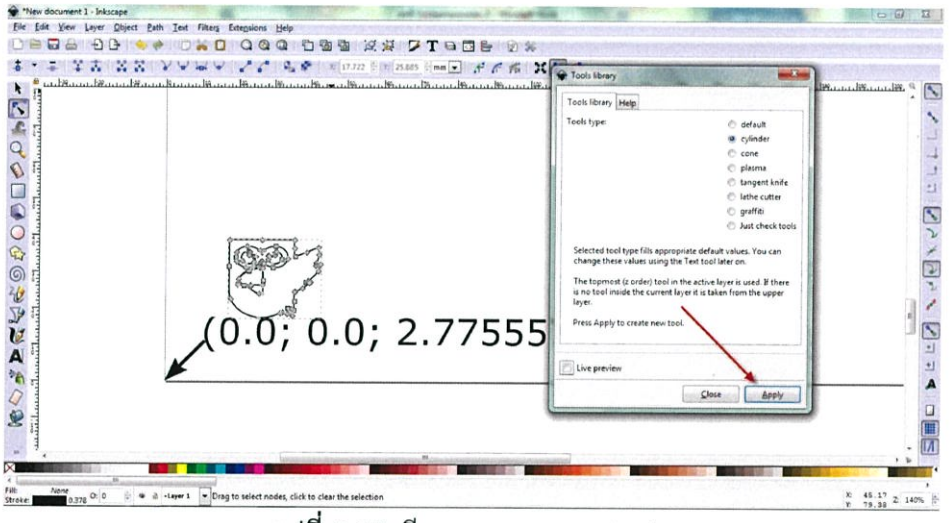
รูปที่ 3.13 เปลี่ยนรูป Bit map ให้เหลือแต่เส้น

- กำหนดจุดเริ่มต้นของการทำงาน โดยเลือกที่ Extensions -> G-code tool -> Orientation points แล้วเลือกตามลูกศร

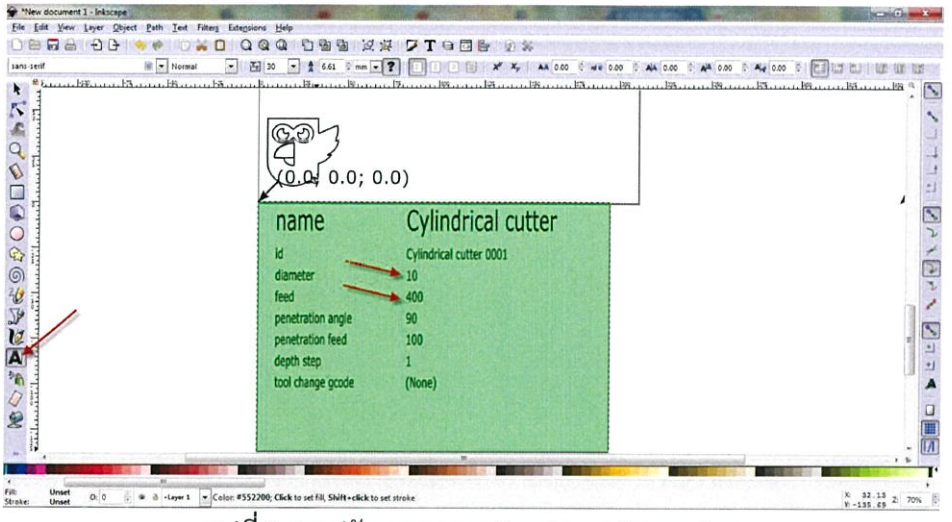


รูปที่ 3.14 แสดงการตั้งจุดเริ่มต้นการทำงาน

- กำหนดขนาดของ Tools โดยไปที่ Extensions -> G-code tool -> tools libraries หลังจากนั้นเลือกที่ Text editor ปรับขนาด Feed และ Diameter ตามต้องการ

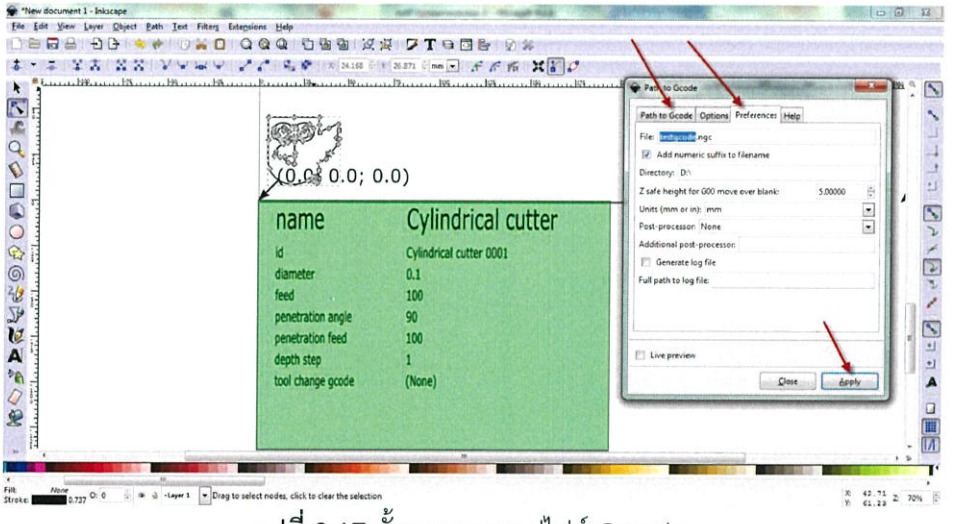


รูปที่ 3.15 เรียกเมนูขนาดของ tools



รูปที่ 3.16 ปรับขนาดของ Feed และ Diameter

- เซฟไฟล์เป็น G-code พร้อมนำไปใช้งาน โดยเลือกที่ Extensions -> G-code tools -> Path to G-code -> Preferences ตั้งชื่อไฟล์ และ โฟลเดอร์ที่จะเก็บไฟล์จากนั้น เลือก Path to G-code แล้วจึงกด Apply จะได้ไฟล์ G-code พร้อมใช้งาน



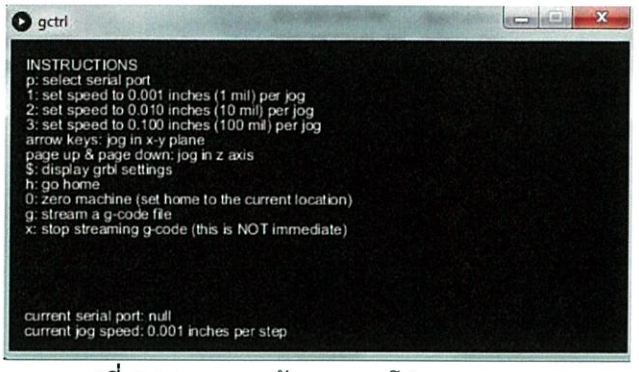
รูปที่ 3.17 ขั้นตอนการเซฟไฟล์ G-code

หลังจากที่ได้ G-code มาแล้ว การจะให้ Microcontroller ทำงานตาม G-code ต้องมีซอฟต์แวร์ Grbl Controller ในการเชื่อมต่อกับ Microcontroller อีกที ในซอฟต์แวร์ตัวนี้สามารถใส่พิกัด G-code ที่ได้มาแล้วสั่งให้เครื่องร่างภาพ วาดรูปออกมาตามพิกัดได้เลย

การส่งข้อมูล G-code ให้ Microcontroller ด้วยโปรแกรม Gctrl

โปรแกรม Grbl controller คือโปรแกรมที่ใช้ส่งค่า G-code ทีละค่าให้กับไมโครคอนโทรลเลอร์ โดยไมโครคอนโทรลเลอร์สามารถใส่โค้ดเพื่อรับค่าคำสั่ง G-code และนำไปใช้ประมวลผลได้

- หน้าแรกของโปรแกรม



รูปที่ 3.18 แสดงหน้าแรกของโปรแกรม Gctrl

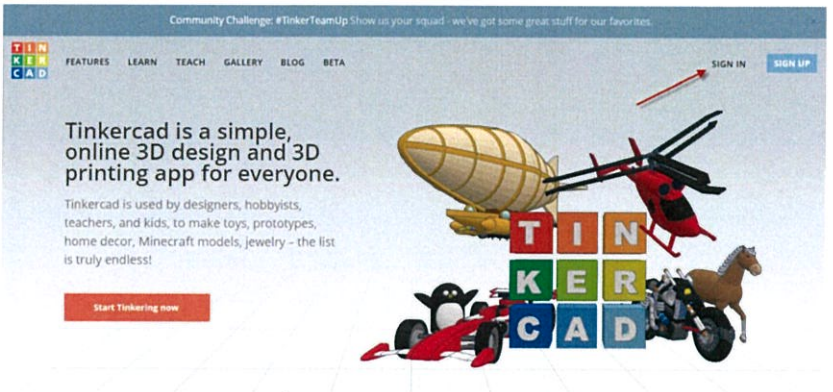
- เมื่อทำการต่อสาย Usb ของไมโครคอนโทรลเลอร์เข้ากับคอมพิวเตอร์แล้ว สามารถกดปุ่ม G เพื่อเลือกไฟล์ G-code ได้เลย และเราสามารถสั่งหยุดการทำงานได้โดยการกดปุ่ม X

การออกแบบชิ้นงาน 3 มิติ

ในส่วนของการออกแบบชิ้นงาน 3 มิติจากเครื่อง 3D printer นั้น ได้ใช้โปรแกรมที่ชื่อว่า Tinkercad ซึ่งเป็นโปรแกรมในเครือเดียวกันกับ Autodesk โดยโปรแกรม Tinkercad นั้นสามารถสร้างชิ้นงาน 3 มิติ ที่มีความละเอียดไม่สูงมากได้เป็นอย่างดี เนื่องจากในการใช้งานต้องเชื่อมต่อ Internet และใช้งานผ่านเว็บไซต์ โดยการเริ่มต้นใช้งานของโปรแกรม Tinkercad นั้น ผู้ใช้ต้องเป็นสมาชิกของ Autodesk เสียก่อนจึงจะสามารถเข้าใช้งานได้

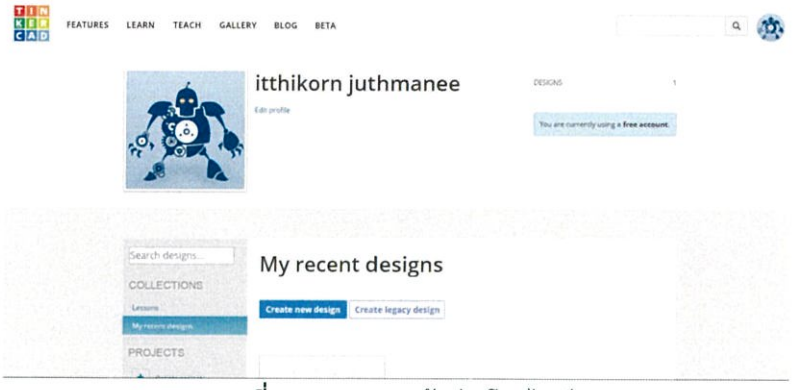
การใช้งานโปรแกรม Tinkercad นั้นมีความซับซ้อนไม่สูงมาก เหมาะสำหรับโปรเจคเครื่องร่างภาพขนาดเล็ก ในการทำตัวจับปากกาเพื่อใช้ในการเขียน และ วาดของเครื่อง

1. เข้าไปที่ www.tinkercad.com และทำการ sign in



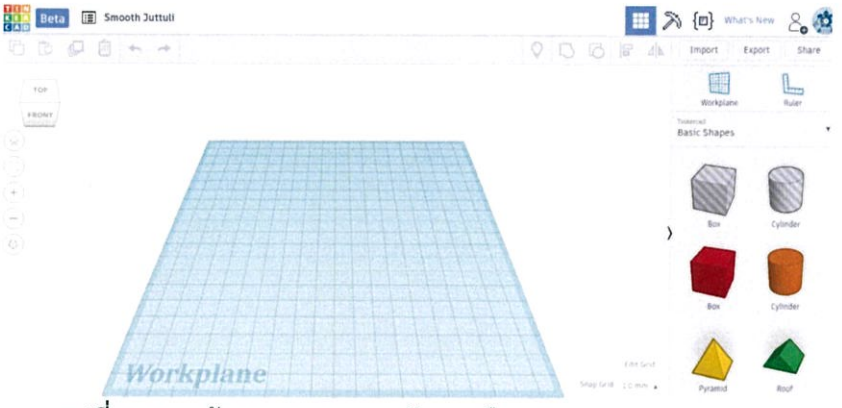
รูปที่ 3.19 หน้าแรกของเว็บไซต์

2. เมื่อทำการ Sign in เป็นที่เรียบร้อยแล้วจะพบกับหน้า Profile ของเรา โดยหน้านี้เราสามารถสร้าง Project ใหม่ของเรา หรือ จะสร้างเพียงแค่ชิ้นงานเดียวครั้งเดียวก็ได้ โดยชิ้นงานนี้จะทำการสร้างเพียง 2 ชิ้นส่วน ให้เลือกที่ Create new design



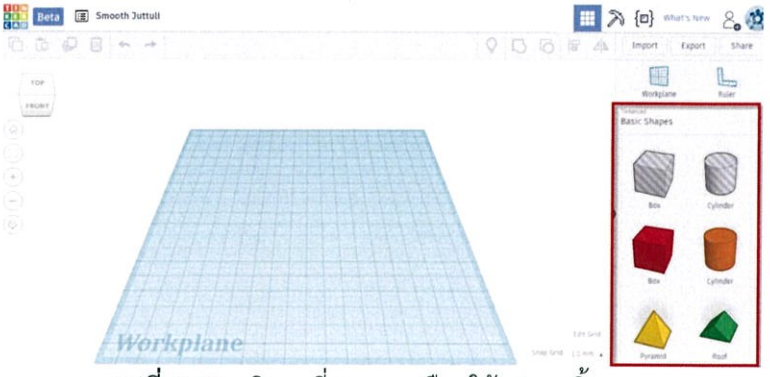
รูปที่ 3.20 แสดงหน้าต่างโปรไฟล์

3. หน้า Workplane



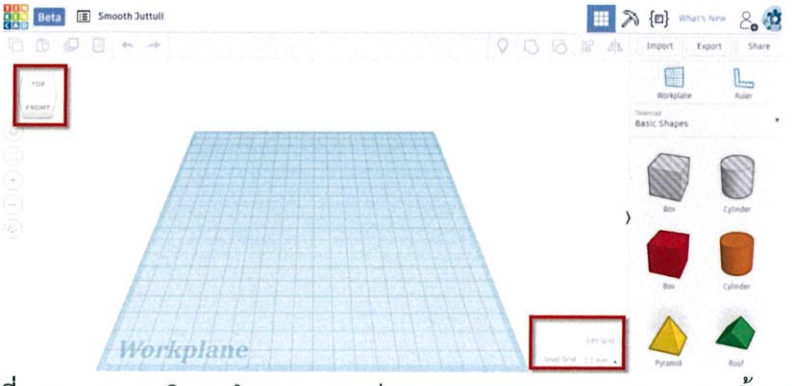
รูปที่ 3.21 หน้า Workplane หลังจากเลือก Create new design

4. ทำการเลือกรูปทรงของชิ้นงาน, รูปแบบตัวหนังสือหรือสัญลักษณ์ และ รูปแบบ Hole



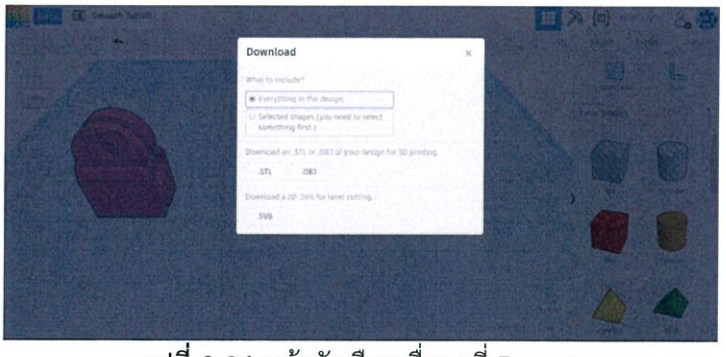
รูปที่ 3.22 บริเวณที่สามารถเลือกใช้รูปทรงชิ้นงาน

5. ปรับแต่งขนาดของช่องตาราง และ สามารถเลือกมุมมองในการมองชิ้นงานได้จาก 2 ส่วนนี้



รูปที่ 3.23 แสดงบริเวณกำหนดขนาดช่องตารางและมุมมองของการมองชิ้นงาน

6. ทำการสร้างชิ้นงาน 3 มิติ ตามที่ได้ร่างแบบไว้
7. เมื่อสร้างชิ้นงาน 3 มิติเรียบร้อยแล้วสามารถ Export ออกมาเป็นไฟล์ได้ .Stl และ .obj เพื่อนำไปใช้กับเครื่อง 3D printer จากรูปให้ทำการเลือกข้อแรก Everything in the design



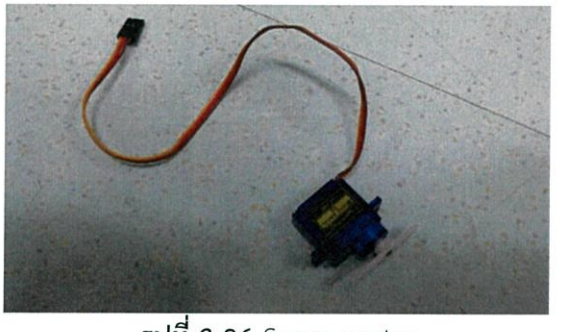
รูปที่ 3.24 หน้าตัวเลือกเมื่อกดที่ Export

3.2 โครงเครื่องร่างภาพ

ส่วนประกอบของโครง

- stepper motor
- servo motor
- เพลาเหล็ก
- สายพาน
- แผ่น อะคริลิก
- slide block
- พูเล่
- shaft support
- วงจรDrive L298N
- บอร์ด Arduino UNO

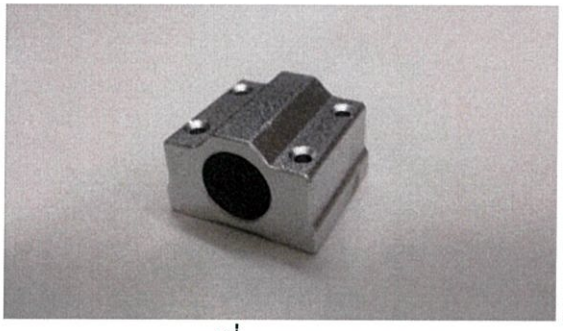
รูปที่ 3.25 Stepper motor



รูปที่ 3.26 Servo motor

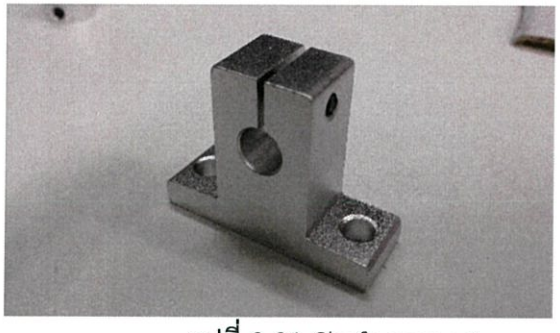
รูปที่ 3.27 แกนเหล็ก

รูปที่ 3.28 สายพาน

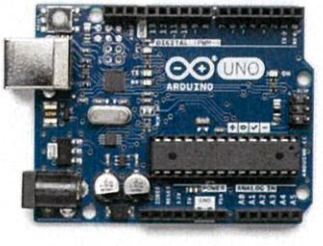


รูปที่ 3.29 Slide block

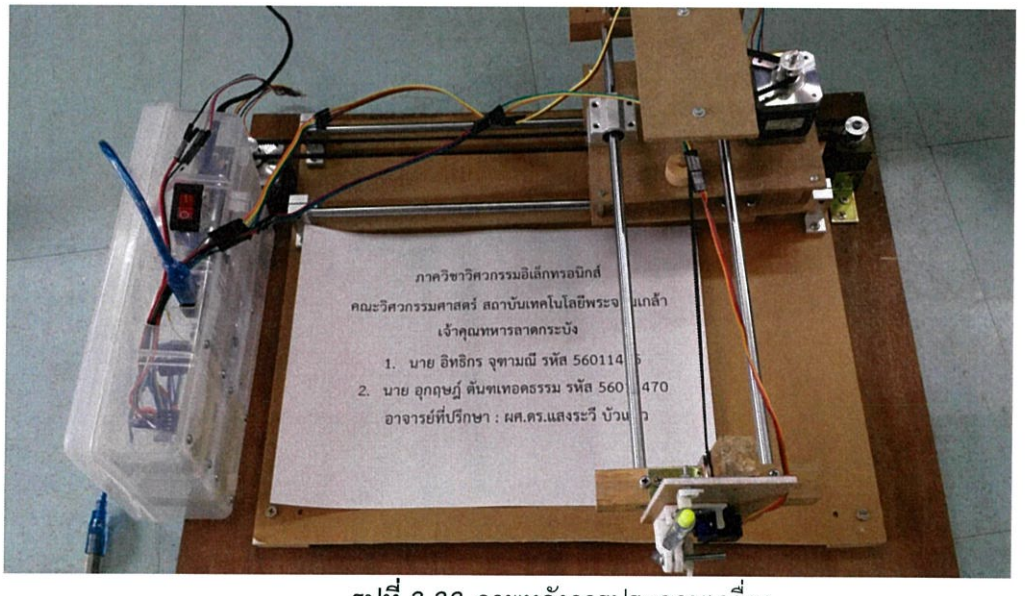
รูปที่ 3.30 พูเล่



รูปที่ 3.31 Shaft support



รูปที่ 3.32 บอร์ด Arduino



รูปที่ 3.33 ภาพหลังการประกอบเครื่อง

บทที่ 4

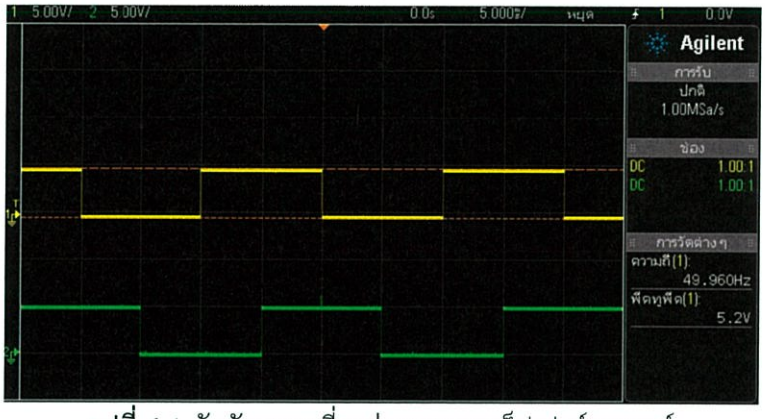
ผลการทดลอง

อุปกรณ์ที่ใช้ในการทดลอง

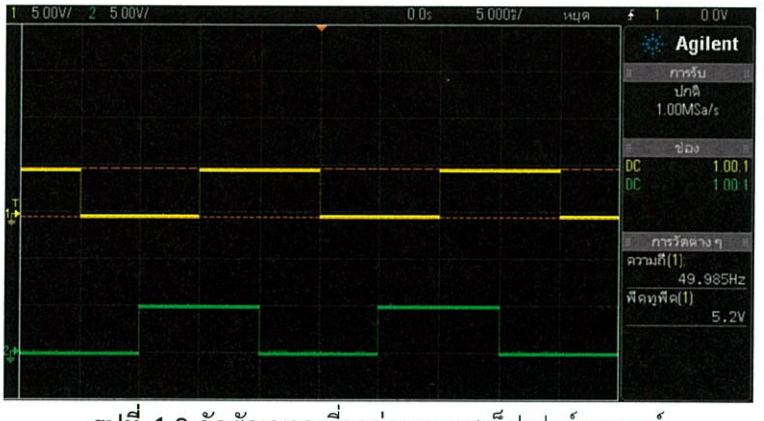
- วงจรไดร์ฟมอเตอร์
- บอร์ด Arduino
- Stepper Motor

วิธีการทดลอง

ทำการโปรแกรมบอร์ดคอนโทรล Arduino ให้ควบคุมให้มอเตอร์เกิดการหมุนโดยวัดค่าอินพุทที่สามารถทำให้มอเตอร์สามารถหมุนได้



รูปที่ 4.1 วัดสัญญาณที่ขาคู่บวกของสเต็ปเปอร์มอเตอร์



รูปที่ 4.2 วัดสัญญาณที่ขาคู่ลบของสเต็ปเปอร์มอเตอร์

จากการทดลองถ้าเราป้อนสัญญาณเข้าในลักษณะข้างต้นอย่างต่อเนื่องแบบ full step จะทำให้สามารถหมุนมอเตอร์ไปได้อย่างต่อเนื่อง

การทดลองการทำงานของเครื่องร่างภาพ

1. การทดลองวัดระยะการเคลื่อนที่ของเครื่องร่างภาพตามแนวแกน X โดยการป้อน Step หลายๆ ค่าให้กับมอเตอร์ โดยมอเตอร์มีองศาการหมุนอยู่ที่ 1.8°ต่อ 1 step และพูเล่สายพานมีรอบวงเท่ากับ 4 เซนติเมตร โดยใช้คำสั่งทดสอบการทำงานดังนี้

```
#include <Stepper.h> ➔➔ เลือกใช้คำสั่ง Stepper.h จากในไลบรารี่
const int stepsPerRevolution = 200; ➔➔ ตั้งค่าตามเสป็คของมอเตอร์ โดย Stepper จะอยู่ที่ 200
Stepper myStepperX(stepsPerRevolution, 8,9,10,11); ➔➔ ขาที่ใช้เชื่อมต่อกับไมโครคอนโทรลเลอร์
void setup() {

myStepperX.setSpeed(100); ➔➔ เลือกความเร็วการทำงาน

myStepperX.step(160); ➔➔ เลือกจำนวน Step ที่ต้องการ
 delay(100);
}
void loop() {
}
```

ทดลองคำสั่งเคลื่อนที่ตามแนวแกน X			
จำนวน step	ระยะการเคลื่อนที่ตามที่คำนวน (cm)	ระยะการเคลื่อนที่จริง (cm)	ค่าผิดพลาด (%)
40	0.8	0.8	0.00
80	1.6	1.65	3.12
120	2.4	2.45	2.08
160	3.2	3.2	0.00
200	4	4	0.00
240	4.8	4.8	0.00
280	5.6	5.5	1.79
320	6.4	6.4	0.00
360	7.2	7.25	0.69
400	8	8.1	1.25
ความผิดพลาดเฉลี่ย			0.89

ตารางที่ 4.1 แสดงระยะการเคลื่อนจริงกับตามที่คำนวณ (แกน X)

จากการทดสอบการทำงานของเครื่องร่างภาพเคลื่อนที่ตามแนวแกน X นั้นได้ผลความผิดพลาดเฉลี่ยอยู่ที่ 0.89% เนื่องจากแกน X อยู่ติดกับส่วนฐานของเครื่องทำให้ความแข็งแรงมีมาก และไม่เคลื่อนที่จากที่คำนวณไว้มาก

2. การทดลองวัดระยะการเคลื่อนที่ของเครื่องร่างภาพตามแนวแกน Y โดยการป้อน Step หลายๆ ค่าให้กับมอเตอร์ โดยมอเตอร์มีองศาการหมุนอยู่ที่ 1.8°ต่อ 1 step และพูเล่สายพานมีรอบวงเท่ากับ 4 เซนติเมตร โดยใช้คำสั่งทดสอบการทำงานดังนี้

   #include <Stepper.h> ➔➔ เลือกใช้คำสั่ง Stepper.h จากในไลบรารี่

   const int stepsPerRevolution = 200; ➔➔ ตั้งค่าตามเสป็คของมอเตอร์ โดย Stepper จะอยู่ที่ 200

   Stepper myStepperX(stepsPerRevolution, 2,3,4,5); ➔➔ ขาที่ใช้เชื่อมต่อกับไมโครคอนโทรลเลอร์

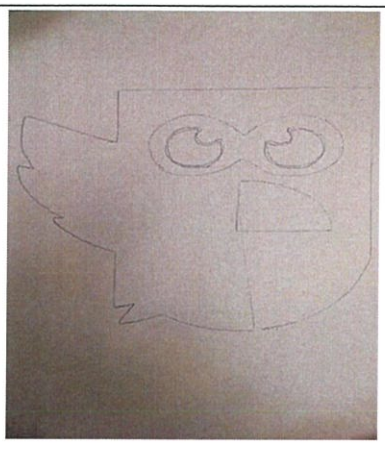
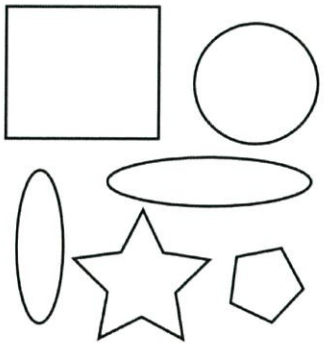
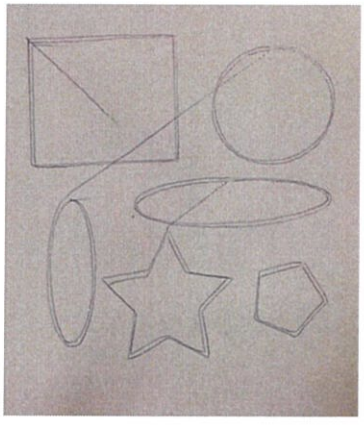
```
void setup() {


myStepperX.setSpeed(100); ➔➔ เลือกความเร็วการทำงาน


myStepperX.step(160); ➔➔ เลือกจำนวน Step ที่ต้องการ
 delay(100);
}
void loop() {
}
```

ทดลองคำสั่งเคลื่อนที่ตามแนวแกน Y			
จำนวน step	ระยะการเคลื่อนที่ตามที่คำนวน (cm)	ระยะการเคลื่อนที่จริง (cm)	ค่าผิดพลาด (%)
40	0.8	0.7	12.50
80	1.6	1.4	12.50
120	2.4	2.25	6.25
160	3.2	2.85	10.94
200	4	3.9	2.50
240	4.8	4.3	10.42
280	5.6	4.8	14.29
320	6.4	6.2	3.13
360	7.2	6.9	4.17
400	8	8.15	1.88
ความผิดพลาดเฉลี่ย			7.86

ตารางที่ 4.2 แสดงระยะการเคลื่อนจริงกับตามที่คำนวณ (แกน Y)

จากการทดสอบการทำงานของเครื่องร่างภาพเคลื่อนที่ตามแนวแกน Y นั้นได้ผลความผิดพลาดเฉลี่ยอยู่ที่ 7.86% เนื่องจากแกน Y อยู่ด้านบนแกน X และมีความแข็งแรงของฐานที่น้อยกว่า เมื่อเครื่องทำงานจึงเกิดการสั่นสะเทือน ผลที่ได้จึงมีความผิดพลาดจากการคำนวณที่ค่อนข้างมาก

3. การทดลองใช้เครื่องในการวาดภาพและเขียนตัวหนังสือ พร้อมทั้งนำมาเปรียบเทียบกับต้นฉบับ

เปรียบเทียบรูปที่ได้กับรูปต้นแบบ	
	
	

ตารางที่ 4.3 เปรียบเทียบรูปต้นฉบับกับรูปจริง

เปรียบเทียบตัวหนังสือที่ได้กับรูปตัวหนังสือต้นแบบ	
ABC abc 123 กขค	

ตารางที่ 4.4 เปรียบเทียบตัวหนังสือที่ได้กับตัวหนังสือต้นฉบับ

บทที่ 5
สรุปผลการทดลอง อุปสรรค ปัญหา และข้อเสนอแนะ

5.1 วิเคราะห์และสรุปผลการทดลอง

จากการทดลองนั้นหลังจากที่ได้ส่ง G code ไปที่ Arduino UNO เพื่อสั่งการให้ตัวเครื่องร่างภาพนั้นวาดภาพออกมา เบื้องต้นเครื่องร่างภาพมีความผิดพลาดในแนววาดแกน Y สูงในลักษณะที่มีการขับเคลื่อนที่ของสายพานผิดแปลกไป เนื่องจากสายพานมีความหย่อนจึงทำให้เกิดความผิดพลาดส่วนนี้ขึ้น หลังจากการแก้ไขส่วนนี้แล้ว เครื่องร่างภาพสามารถที่จะวาดภาพออกมาได้ถูกต้อง และแม่นยำโดยมีสัดส่วนของภาพวาดตรงตามรูปที่นำมาทำการแปลงไฟล์ให้เป็น G code และทำการส่ง แต่มีข้อผิดพลาดส่วนอื่นคือตัวฐานของเครื่องร่างภาพมีความไม่สม่ำเสมอกันในช่วงเวลาที่ตัวเครื่องทำการยกปากกานั้นทำให้มีรอยขีดข่วนที่ไม่ต้องการเกิดขึ้น ในระหว่างที่ตัวปากกาทำการเปลี่ยนตำแหน่งเพื่อที่วาดภาพต่อๆไป

5.2 อุปสรรคและปัญหา

ปัญหาในส่วนของการสร้างตัวโครงของตัวเครื่องร่างภาพ หากทำการออกแบบตัวโครงเครื่องได้ไม่ดีพอจะทำให้ขั้นตอนการตัดเพื่อสร้างส่วนชิ้นงานและการประกอบนั้นมีความยุ่งยากมากขึ้นและทำให้สิ้นเปลืองชิ้นส่วนที่เสียต่างๆ รวมทั้งการประกอบโครงร่างหลังจากที่ทำการเตรียมชิ้นงานต่างๆเรียบร้อยแล้วมีปัญหาในการติดตั้งสายพานที่สืบต่อมาจากการออกแบบ ทำให้ติดตั้งสายพานได้ยาก สายพานจึงเกิดการหย่อนไม่ตึง จะส่งผลให้การวาดภาพออกมานั้นเกิดความไม่ถูกต้องและผิดเพี้ยนไป เนื่องจาก เกิดความผิดพลาดในการเคลื่อนที่ของตัวหัวปากกาที่ติดตั้งกับสายพาน และยังมีความร้อนที่มากในตัวของวงจรที่ใช้ในการไดร์ฟกระแสเพื่อควบคุมตัวสเต็ปเปอร์มอเตอร์

5.3 ข้อเสนอแนะ

ควรที่จะออกแบบโครงร่างของเครื่องร่างภาพให้เรียบร้อยและมีความเรียบง่ายเพื่อที่จะทำให้ตัวชิ้นงานนั้นง่ายต่อการถอดและประกอบ เพื่อที่จะทำการซ่อมแซมและพัฒนาต่างๆภายหลัง และตัวโครงต่างๆควรเลือกวัสดุที่ง่ายต่อการจัดการต่างๆเช่นตัด เจาะ และมีความแข็งแรงและคงทน ไม้ไม่ควรใช้ไม้ที่อาจเกิดการบิดเบี้ยวหรือหดต่างๆในภายหลังการเตรียมชิ้นงาน ตัวโครงเครื่องร่างภาพนั้นยังสามารถที่จะพัฒนาตัวเครื่องให้กลายเป็นเครื่องเจาะหรือเครื่องกลึงต่างๆได้อีกด้วย เพียงแต่ต้องมีงบประมาณที่มากพอและวัสดุต่างๆที่ต้องการความแข็งแรงมากขึ้นเพื่อที่จะรองรับน้ำหนักและการสั่นสะเทือนต่างๆในงานเหล่านั้น เพื่อที่จะทำให้งานที่ออกมานั้นมีประสิทธิภาพเพียงพอ

เอกสารอ้างอิง

[1] อาจารย์ มงคล ศิริธนานุกูลวงศ์ , http://tisade.blogspot.com/

[2] Brian w. Evans. Arduino programming notebook

[3] ผศ. สัจจาทิพย์ ทัศนียพันธุ์. Conventional & Computer-controlled machines. Kasetsart University.

[4] Vasan Kuldiloksawet. Inkscape Encyclopedia

[5] Seven Vinton. Arduino lessons

[6] http://praponangkeaw-motor.blogspot.com/2015/02/r-b-f1-b-f2-b-f1-f2-2-a1-a2-d1-d2-l-l.html

[7] http://www.rmutphysics.com/charud/scibook/electric4/bottee9.htm

[8] Takashi Kenjo and Akira Sugawara (1994) : "Stepping Motor and Their Microprocessor Cntrols Second Edition", Clarendon Press, Oxford 1994

[9] http://www.thaiio.com/Hardware/stepmoter.htm

[10] http://202.129.59.73/tn/motor10-52/motor1.htm

[11] http://praponangkeaw-motor.blogspot.com/2015/02/r-b-f1-b-f2-b-f1-f2-2-a1-a2-d1-d2-l-l.html

[12] http://www.rmutphysics.com/charud/scibook/electric4/bottee9.htm

[13] Facebook page : CNC DIY by GRBL or @laserDIY9934

ภาคผนวก

Code บน Arduino เพื่อรับ G code

```
#include <Servo.h>
#include <Stepper.h>
#define LINE_BUFFER_LENGTH 512
const int penZUp = 80;
const int penZDown = 20;
const int penServoPin = 6;
const int stepsPerRevolution = 200;
Servo penServo;
Stepper myStepperY(stepsPerRevolution, 2,3,4,5);
Stepper myStepperX(stepsPerRevolution, 8,9,10,11);
struct point {
  float x;
  float y;
  float z;
};
struct point actuatorPos;
float StepInc = 1;
int StepDelay = 0;
int LineDelay = 50;
int penDelay = 50;
float StepsPerMillimeterX = 5;
float StepsPerMillimeterY = 5;
float Xmin = 0;
float Xmax = 300;
float Ymin = 0;
float Ymax = 300;
float Zmin = 0;
float Zmax = 5;
```

```
float Xpos = Xmin;
float Ypos = Ymin;
float Zpos = Zmax;

void setup() {
  Serial.begin( 9600 );
  penServo.attach(penServoPin);
penServo.write(penZUp);
  delay(200);
  myStepperX.setSpeed(50);
  myStepperY.setSpeed(50);
}
void loop()
{
  delay(200);
  char line[ LINE_BUFFER_LENGTH ];
  char c;
  int lineIndex;
  bool lineIsComment, lineSemiColon;

  lineIndex = 0;
  lineSemiColon = false;
  lineIsComment = false;

  while (1) {
       while ( Serial.available()>0 ) {
     c = Serial.read();
     if (( c == '\n') || (c == '\r') ) {
       if ( lineIndex > 0 ) {
         line[ lineIndex ] = '\0';
```

```
    if (verbose) {
      Serial.print( "Received : ");
      Serial.println( line );
    }
    processIncomingLine( line, lineIndex );
    lineIndex = 0;
  }
  else {
  }
  lineIsComment = false;
  lineSemiColon = false;
  Serial.println("ok");
}
else {
  if ( (lineIsComment) || (lineSemiColon) ) {
    if ( c == ')' )  lineIsComment = false
  }
  else {
    if ( c <= ' ' ) {
    }
    else if ( c == '/' ) {
    }
    else if ( c == '(' ) {
      lineIsComment = true;
    }
    else if ( c == ';' ) {
      lineSemiColon = true;
    }
    else if ( lineIndex >= LINE_BUFFER_LENGTH-1 ) {
      Serial.println( "ERROR - lineBuffer overflow" );
```

```
            lineIsComment = false;
            lineSemiColon = false;
          }
          else if ( c >= 'a' && c <= 'z' ) {
            line[ lineIndex++ ] = c-'a'+'A';
          }
          else {
            line[ lineIndex++ ] = c;
          }
        }
      }
    }
  }
}
void processIncomingLine( char* line, int charNB ) {
  int currentIndex = 0;
  char buffer[ 64 ];
  struct point newPos;

  newPos.x = 0.0;
  newPos.y = 0.0;

  while( currentIndex < charNB ) {
    switch ( line[ currentIndex++ ] ) {
    case 'U':
      penUp();
      break;
    case 'D':
      penDown();
      break;
```

```
 case 'G':
   buffer[0] = line[ currentIndex++ ];
   buffer[1] = '\0';

   switch ( atoi( buffer ) ){
   case 0:
   case 1:
     char* indexX = strchr( line+currentIndex, 'X' );
     char* indexY = strchr( line+currentIndex, 'Y' );
     if ( indexY <= 0 ) {
       newPos.x = atof( indexX + 1);
       newPos.y = actuatorPos.y;
     }
     else if ( indexX <= 0 ) {
       newPos.y = atof( indexY + 1);
       newPos.x = actuatorPos.x;
     }
     else {
       newPos.y = atof( indexY + 1);
       indexY = '\0';
       newPos.x = atof( indexX + 1);
     }
     drawLine(newPos.x, newPos.y );
     actuatorPos.x = newPos.x;
     actuatorPos.y = newPos.y;
     break;
   }
   break;
 case 'M':
   buffer[0] = line[ currentIndex++ ];
```

```
      buffer[1] = line[ currentIndex++ ];
      buffer[2] = line[ currentIndex++ ];
      buffer[3] = '\0';
      switch ( atoi( buffer ) ){
      case 300:
       {
         char* indexS = strchr( line+currentIndex, 'S' );
         float Spos = atof( indexS + 1);
         if (Spos == 30) {
           penDown();
         }
         if (Spos == 50) {
           penUp();
         }
         break;
       }
      case 114:
        Serial.print( "Absolute position : X = " );
        Serial.print( actuatorPos.x );
        Serial.print( "  -  Y = " );
        Serial.println( actuatorPos.y );
        break;
      default:
        Serial.print( "Command not recognized : M");
        Serial.println( buffer );
      }
    }
  }
}
void drawLine(float x1, float y1) {
```

```
if (verbose)
{
  Serial.print("fx1, fy1: ");
  Serial.print(x1);
  Serial.print(",");
  Serial.print(y1);
  Serial.println("");
}
if (x1 >= Xmax) {
  x1 = Xmax;
}
if (x1 <= Xmin) {
  x1 = Xmin;
}
if (y1 >= Ymax) {
  y1 = Ymax;
}
if (y1 <= Ymin) {
  y1 = Ymin;
}

if (verbose)
{
  Serial.print("Xpos, Ypos: ");
  Serial.print(Xpos);
  Serial.print(",");
  Serial.print(Ypos);
  Serial.println("");
}
```

```
if (verbose)
{
  Serial.print("x1, y1: ");
  Serial.print(x1);
  Serial.print(",");
  Serial.print(y1);
  Serial.println("");
}
x1 = (int)(x1*StepsPerMillimeterX);
y1 = (int)(y1*StepsPerMillimeterY);
float x0 = Xpos;
float y0 = Ypos;
long dx = abs(x1-x0);
long dy = abs(y1-y0);
int sx = x0<x1 ? StepInc : -StepInc;
int sy = y0<y1 ? StepInc : -StepInc;
long i;
long over = 0;

if (dx > dy) {
  for (i=0; i<dx; ++i) {
    myStepperX.step(sx);
    over+=dy;
    if (over>=dx) {
      over-=dx;
      myStepperY.step(sy);
    }
    delay(StepDelay);
  }
}
```

```
else {
  for (i=0; i<dy; ++i) {
    myStepperY.step(sy);
    over+=dx;
    if (over>=dy) {
      over-=dy;
      myStepperX.step(sx);
    }
    delay(StepDelay);
  }
}

if (verbose)
{
  Serial.print("dx, dy:");
  Serial.print(dx);
  Serial.print(",");
  Serial.print(dy);
  Serial.println("");
}

if (verbose)
{
  Serial.print("Going to (");
  Serial.print(x0);
  Serial.print(",");
  Serial.print(y0);
  Serial.println(")");
}
```

```
  delay(LineDelay);
  Xpos = x1;
  Ypos = y1;
}

void penUp() {
  penServo.write(penZUp);
  delay(LineDelay);
  Zpos=Zmax;
  if (verbose) {
    Serial.println("Pen up!");
  }
}
void penDown() {
  penServo.write(penZDown);
  delay(LineDelay);
  Zpos=Zmin;
  if (verbose) {
    Serial.println("Pen down.");
  }
}
```