แขนกลควบคุมด้วยไมโครคอนโทรลเลอร์

Robotic arm controlled by microcontroller

ปภาวี เจริญชัยปรารถนา
Paphawee Charoenchaiprathana

พุฒิกร ปิยกิจเลิศอนันต์
Puthikorn Piyakitlertanan

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2561

แขนกลควบคุมด้วยไมโครคอนโทรลเลอร์

Robotic arm controlled by microcontroller

โดย

ปภาวี เจริญชัยปรารถนา

พุฒิกร ปิยกิจเลิศอนันต์

อาจารย์ที่ปรึกษา

ผศ.ดร.แสงระวี บัวแก้ว

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2561

ปริญญานิพนธ์ปีการศึกษา 2561

ภาควิชา วิศวกรรมอิเล็กทรอนิกส์

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง แขกลควบคุมด้วยไมโครคอนโทรลเลอร์

Robotic arm controlled by microcontroller

ผู้จัดทำ นางสาวปภาวี เจริญชัยปรารถนา รหัสประจำตัว 58010728

นายพุฒิกร ปิยกิจเลิศอนันต์ รหัสประจำตัว 58010918

ปริญญานิพนธ์นี้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

.....

ผศ.ดร.แสงระวี บัวแก้ว

อาจารย์ที่ปรึกษา

หัวข้อปริญญานิพนธ์ แขนกลควบคุมด้วยไมโครคอนโทรลเลอร์

นักศึกษา นางสาว ปภาวี เจริญชัยปรารถนา รหัสนักศึกษา 58010728

นาย พุฒิกร ปิยกิจเลิศอนันต์ รหัสนักศึกษา 58010918

ปริญญา วิศวกรรมศาสตร์บัณฑิต

ภาควิชา วิศวกรรมอิเล็กทรอนิกส์

ปีการศึกษา 2561

อาจารย์ที่ปรึกษา ผศ.ดร.แสงระวี บัวแก้ว

บทคัดย่อ

โครงงานนี้นำเสนอ การออกแบบและการสร้างแขนกลที่ถูกควบคุมด้วยไมโครคอนโทรลเลอร์ โดยมีวัตถุประสงค์เพื่อใช้ในการทำงานร่วมกับชุดควบคุมที่อาศัยการเคลื่อนไหวของนิ้วมือมนุษย์ทั้งห้านิ้ว โครงสร้างแขนกลนี้ประกอบด้วยสองคือส่วนของมือกลและส่วนแขน ที่มีการทำงานร่วมกันซึ่งทั้งสองส่วนนี้ควบคุมการทำงานโดยไมโครคอนโทรลเลอร์และเซอร์โวมอเตอร์ สำหรับส่วนของเซอโวมอเตอร์จะถูกควบคุมด้วยโปรแกรม วิชวลเบสิกจะส่งสัญญาณข้อมูลไปในไมโครโปรเซสเซอร์ “อาดูโน่”เพื่อให้ประมวลผลและควบคุมการทำงานของนิ้วมือทั้งห้าซึ่งจะสามารถช่วยสื่อสารออกมาเป็นรูปตัวอักษรในภาษาใบ้

Thesis Title	Robotic arm controlled by microcontroller	
Students	Miss Paphawee Charoenchaiprathana	Student ID 57010728
	Mr. Puthikorn Piyakitlertanan	Student ID 58010918
Degree	Bachelor of Engineering	
Program	Electronics Engineering	
Academic Year	2018	
Thesis Advisor	Assistant Professor Dr.Seangrawee Buakeaw	

ABSTRACT

This project provides details and informations in the designing and construction of microprocessor-controlling artificial arm with five servo motors. The structure of the artificial arm constructs from 2 parts which are the hand and the arm designed to work collaboratively and both of their functionality will depend on the microprocessing unit and the servo motor. Speaking of which, the functionality of the servo motor will be resulted from the Visual basic and also the microprocessing unit in order to send the control signal via microprocessor " arduino uno r3" to the artificial arm unit and the microprocessing unit will control the motion of every single limbs using the servo motor to helps to communicate with humans in sign languages , eventually

กิตติกรรมประกาศ

โครงงาน “แขนกลควบคุมโดยไมโครคอนโทรลเลอร์”สำเร็จลุล่วงไปได้ด้วยดีจากความช่วยเหลือและให้คำปรึกษาจาก ผศ.ดร.แสงระวี บัวแก้ว อาจารย์ที่ปรึกษาที่คอยช่วยเหลือในการทดลองและให้ความรู้รวมถึงการแก้ไขปัญหาต่างๆระหว่างดำเนินการผลิตชิ้นงานออกมาและคำแนะนำจากอาจารย์ภาควิชาอิเล็คทรอนิกส์ คณะผู้จัดทำโครงงานขอขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณพ่อแม่ของคณะผู้จัดทำที่คอยเป็นกำลังใจหลักและเป็นผู้สนับสนุนเงินทุน หลักในการทำโครงงาน รวมไปถึงเพื่อนๆผู้เป็นกำลังหลักคอยช่วยเหลือ ให้คำแนะนำ รวมทั้ง ช่วยกันในการแก้ไขปัญหาที่เกิดขึ้นในระหว่างการทำโครงงานนี้ทั้งหมดทำให้ผลของโครงงานสำเร็จลุลวงไปด้วยดี

สุดท้ายนี้ คณะผู้จัดททำหวังว่าโครงงานนี้จะเป็นประโยชน์สำหรับผู้สนใจและผู้นำผลงานนี้ ไปใช้งานได้

ปภาวี เจริญชัยปรารถนา 58010728

พุฒิกร ปิยกิจเลิศอนันต์ 58010918

สารบัญ

หน้า

สารบัญรูปภาพ

บทที่ 2 แขนกลอุตสาหกรรม

บทที่ 1

บทนำ

1.1 หลักการและเหตุผล

สำหรับกระบวนการผลิตในทางอุตสาหกรรม หุ่นยนต์แขนกลถือว่าเป็นส่วนสำคัญและได้เข้ามามีบทบาทเป็นอย่างมากในการผลิตหลายๆขั้นตอน โดยเฉพาะอย่างยิ่งในส่วนของการหยิบจับและเคลื่อนย้ายวัตถุจากตำแหน่งหนึ่งไปยังอีกตำแหน่งหนึ่ง ซึ่งพบมากในกระบวนการผลิตแทบจะทุกประเภทดังนั้นการควบคุมให้หุ่นยนต์แขนกลเคลื่อนที่และหยิบจับวัตถุได้อย่างถูกต้องและแม่นยำหรืออย่างน้อยที่สุดคือให้เกิดความผิดพลาดน้อยที่สุดเท่าที่ เป็นไปได้จึงเป็นสิ่งสำคัญที่ควรจะถูกพิจารณา ในที่นี้จะกล่าวถึงแขนกลเพื่อช่วยเหลือผู้พิการทางการได้ยินหรือผู้ที่เป็นใบ้ โดยแขนกลที่ได้จัดทำนี้จะช่วยเป็นสื่อกลางทางการสื่อสาร โดยเรื่องราวของงานสามมิตินั้นมีมากกว่านั้นอีกเพียบ เพราะตอนนี้งานสามมิติได้ถูกนำมาพัฒนาและนำมาใช้ในทางการแพทย์อย่างกว้างขวาง การพิมพ์งานสามมิติคือการนำข้อมูลต่างๆ มาสร้างเป็นภาพบนหน้าจอคอมพิวเตอร์ จัดแต่งรูปทรงตามความต้องการจากนั้นก็เพิ่มความลึกหรือมิติของงานชิ้นนั้นเข้าไป ก่อนจะใช้เครื่องพิมพ์แบบพิเศษสร้างงานออกมาเป็นรูปทรงที่จับต้องได้ ส่วนวิธีการสร้างอวัยวะสามมิติของมนุษย์นั้นก็ไม่ได้ซับซ้อนอย่างที่คิด เพราะเพียงแค่สร้างแบบจำลองอวัยวะในคอมพิวเตอร์ขึ้นมา ดูจุดเชื่อมต่อของกระดูกและกล้ามเนื้อของอวัยวะนั้นๆ ก่อนพิมพ์ชิ้นส่วนออกมาทีละชิ้นๆ แล้วนำมาต่อกันเหมือนเวลาที่เราเล่นต่อหุ่นยนต์ หรือเลโก้นั่นเอง แต่สิ่งที่ยากสุด ก็คือการทำอย่างไรให้ตัวต่อนั้นสามารถนำมาใช้งานได้จริง

เพื่อเป็นการศึกษาทำความเข้าใจเกี่ยวกับการสร้างแขนกลผู้จัดทำโครงงานจึงมีแนวคิดที่สร้างแขนเทียมที่มีรูปร่างเหมือนมือและแขนของมนุษย์จริงๆ ประกอบด้วยเซอโวมอเตอร์ 5 ตัว เพื่อศึกษาการควบคุมมอเตอร์ในลักษณะการหมุนทิศทางต่างๆและออกแบบวงจรควบคุมรวมถึงการเขียนโปรแกรมควบคุมบอร์ดไมโครคอนโทรลเลอร์ได้ รวมทั้งศึกษาการเขียนโปรแกรม Visual C เป็นคำสั่งเพื่อควบคุมการทำงานของแขนกลและยังสามารถออกแบบเป็นภาษาของคนใบ้โดยรูปร่างของมือจะสามารถแปลเป็นภาษาอังกฤษตามตัวอักษร A-Z นอกจากนี้การเขียนโปรแกรมเพื่อควบคุมอัตโนมัติแขนกลจำลองนี้ยังสามารถใช้เป็นสื่อการเรียนการสอนเกี่ยวกับระบบควบคุมโครงสร้างและการเขียนโปรแกรมสำหรับควบคุมแขนกลได้ ซึ่งสามารถนำการสร้างแขนกลนี้ไปใช้ประโยชน์ต่อในอนาคตได้

1.2 วัตถุประสงค์

1. สามารถออกแบบและสร้างหุ่นยนต์แขนกลขนาดเล็กโดยใช้แผ่นฟิลาเม้นเป็นโครงสร้างและใช้เซอร์โวมอเตอร์ในการควบคุมการทำงาน

2.เพื่อศึกษาการทำงานบอร์ด Arduino

3.เพื่อศึกษาและเขียนโปรแกรมที่ใช้ในการควบคุมแขนกลได้

4.สามารถนำไปใช้งานได้จริง

1.3 ขอบข่ายของงาน

1.หุ่นยนต์แขนกลขนาดเล็กน้ำหนักไม่เกิน 5kg มี 7 ข้อต่อ ควบคุมด้วยเซอโวมอเตอร์หุ่นยนต์แขนกลสามารถหยิบจับวัสดุที่มีน้ำหนักเบาและสามารถใช้งานได้เสมือนมือมนุษย์จริงๆใช้การควบคุมแบบ auto การควบคุมข้อต่อด้วยการควบคุมข้อต่อด้วยเซอโวมอเตอร์ให้สามารถดึงเส้นเอ็นให้นิ้วมือสามารถกำหรือแบมือได้โดยใช้หลักการเขียนโปรแกรมของ Arduino

1.4 แผนการดำเนินงาน

1.ศึกษาหลักการทำงานของระบบ

2.หาแหล่งอ้างอิงเพื่อออกแบบ 3d printed ในโปรแกรม Solid work

3.ศึกษาวิธีการเขียน Codeในการควบคุมแขนกล

4.วางแผนอุปกรณ์ ที่ต้องการซื้อ

5.ออกแบบ แขนกล ในโปนแกรมSolid work

6.สั่งพิมพ์งานสามมิติ

7.แกะsupportหรือเส้นใยฟิลาเม้นที่เกินออกมาจากชิ้นงานที่พิมพ์ออกมา

8.นำชิ้นส่วนแบบพิมพ์สามมิติที่ได้มาประกอบเข้าด้วยกัน

9.ทำการร้อยเส้นเอ็นขึ้นไปตามข้อต่อต่างๆของนิ้วมือ

10.ยึดเส้นเอ็นเข้ากับ Servo motor และประกอบลงในตัวแขนกล

11.ติดตั้งเชื่อมต่อกับ บอร์ด Arduino uno R3

12.ศึกษาและสร้างบอร์ด Arduino stane alone

13.เขียนCode เพื่อให้เซอร์โวมอเตอร์สามารถดึงเส้นเอ็นให้งอลงหรือดึงขึ้นได้

14.ทำหน้าต่าง User Interface โดยใช้โปรแกรม Visual Basic เพื่อเป็นตัวสั่งการทำงานของแขนกล

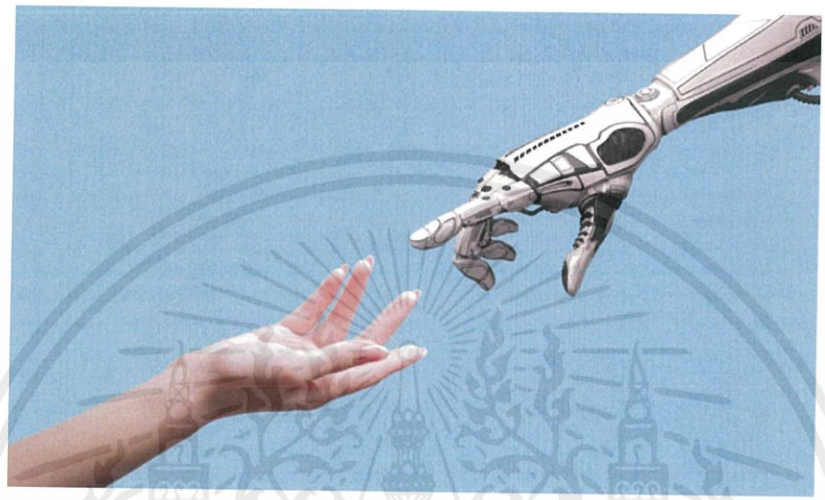
15.ทดลองการทำงาน

16.แก้ไขปัญหา

บทที่ 2

หลักการและทฤษฎี

2.1 แขนกลอุตสาหกรรม



รูปที่ 2.1 แขนกลอุตาสหกรรม

ความคิดของการสร้างสรรค์สิ่งประดิษฐ์ใหม่ๆ จะกล่าวถึงแขนกลที่มีลักษณะคล้ายคลึงกับมือของมนุษย์ แขนกลเป็นหุ่นยนต์ชนิดหนึ่งที่นำมาใช้งานในวงการอุตสาหกรรมการผลิต ได้ถูกนำมาใช้แทนแรงงานมนุษย์ในงานที่ต้องทำอย่างต่อเนื่องตลอด 24 ชั่วโมง ปกติมนุษย์ก็สามารถทำงานได้ทุกอย่างแต่ข้อจำกัดของมนุษย์นั้นไม่สามารถทำงานได้อย่างต่อเนื่องยาวนานจะเกิดความเหน็ดเหนื่อยเมื่อยล้าจึงต้องมีการพักผ่อน

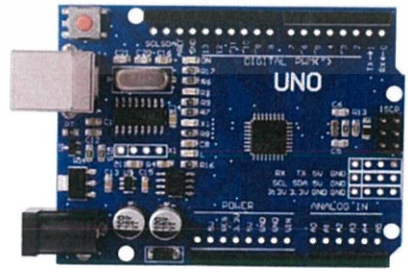
การพัฒนามือของหุ่นยนต์เพื่อใช้งานด้านการแพทย์นั้นผู้พัฒนาจะต้องพยายามออกแบบแขนกลให้มีลักษณะคล้ายกับสรีระมือของมนุษย์ให้มากที่สุดเพื่อที่จะสื่อสารเป็นภาษามือได้แบบตามที่ต้องการ

2.2 ไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ คือ อะไรจริงๆแล้วคำว่า ไมโครคอนโทรลเลอร์ คือ ชิพประมวลผลชนิดหนึ่ง เป็นสมองของหุ่นยนต์ ไมโครคอนโทรลเลอร์ช่วยให้นักออกแบบในการติดต่อเซ็นเซอร์และอุปกรณ์อิเล็กทรอนิกส์ควบคุมพิเศษร่วมกัน (พร้อมกับสิ่งที่จำเป็นอื่นใดสำหรับโครงการ) และมีตรรกะโดยรวมของหุ่นยนต์ ไมโครคอนโทรลเลอร์ที่มี Core Processor หน่วยความจำและอินพุต / เอาต์พุต Programmable อุปกรณ์ต่อพ่วงหน่วยความจำโปรแกรมในรูปแบบของ Ferroelectric RAM หรือแฟลชหรือ OTP รอมก็มักจะรวมอยู่ในชิป เช่นเดียวกับจำนวนเงินขนาดเล็กโดยทั่วไปของแรม ไมโครคอนโทรลเลอร์ได้รับการออกแบบสำหรับการใช้งานที่ฝัง

ตัวในทางตรงกันข้ามกับไมโครโปรเซสเซอร์ที่ใช้ในเครื่องคอมพิวเตอร์ส่วนบุคคลหรือการใช้งานอเนกประสงค์อื่น ๆ ซึ่งประกอบด้วยชิปที่ไม่ต่อเนื่องต่างๆ ไมโครคอนโทรลเลอร์ที่ใช้ในผลิตภัณฑ์ควบคุมโดยอัตโนมัติและอุปกรณ์เช่น ระบบควบคุมเครื่องยนต์รถยนต์, อุปกรณ์ทางการแพทย์ implantable, การควบคุมระยะไกล, เครื่องใช้สำนักงาน, เครื่องใช้ไฟฟ้า, เครื่องมือไฟฟ้า, ของเล่นและระบบฝังตัวอื่น ๆ โดยการลดขนาดและค่าใช้จ่ายเมื่อเทียบกับการออกแบบที่ใช้ไมโครโปรเซสเซอร์ที่แยกต่างหาก, หน่วยความจำและอินพุต / อุปกรณ์ส่งออกที่ควบคุมขนาดเล็กทำให้ประหยัดในการควบคุมอุปกรณ์ดิจิทัลมากยิ่งขึ้นและกระบวนการ ไมโครคอนโทรลเลอร์สัญญาณผสมอยู่ร่วมกันบูรณาการแบบอะนาล็อกชิ้นส่วนจำเป็นในการควบคุมระบบอิเล็กทรอนิกส์ไม่ใช่ดิจิตอล ไมโครคอนโทรลเลอร์บางคนอาจใช้คำสี่บิตและทำงานที่ความถี่ต่ำเป็น 4 เฮิร์ทซ์สำหรับการใช้พลังงานต่ำ (มิลลิวัตต์หลักเดียวหรือไมโคร) พวกเขามักจะมีความสามารถในการรักษาฟังก์ชันการทำงานขณะที่รอให้เหตุการณ์เช่นการกดปุ่มหรือการขัดจังหวะอื่น ๆ การใช้พลังงานในขณะนอนหลับ (นาฬิกา CPU และอุปกรณ์ต่อพ่วงส่วนใหญ่ปิด) อาจเป็นเพียง nanowatts ทำให้คนอีกจำนวนมากเหมาะสำหรับการใช้งานที่ยาวนานของแบตเตอรี่ยาวนาน ไมโครคอนโทรลเลอร์อื่น ๆ อาจจะทำหน้าที่บทบาทของประสิทธิภาพการทำงานที่มีความสำคัญที่พวกเขาอาจจะต้องดำเนินการมากขึ้นเช่นประมวลผลสัญญาณดิจิตอล (DSP) มีความเร็วสัญญาณนาฬิกาที่สูงขึ้นและการใช้พลังงาน ไมโครคอนโทรลเลอร์จะถือว่าเป็นระบบที่ตนเองมีกับหน่วยประมวลผลหน่วยความจำและอุปกรณ์ต่อพ่วงและสามารถนำมาใช้เป็นระบบฝังตัว. [13] ส่วนใหญ่ของไมโครคอนโทรลเลอร์ในการใช้งานในวันนี้จะฝังตัวอยู่ในเครื่องจักรอื่น ๆ เช่นรถยนต์, โทรศัพท์, เครื่องใช้ไฟฟ้าและอุปกรณ์ต่อพ่วงสำหรับระบบคอมพิวเตอร์ ในขณะที่บางระบบฝังตัวมีความซับซ้อนมากหลายมีความต้องการน้อยที่สุดสำหรับหน่วยความจำและระยะเวลาของหลักสูตรด้วยไม่มีระบบปฏิบัติการและความซับซ้อนซอฟต์แวร์ต่ำ ทั่วไปเข้าและส่งออกอุปกรณ์รวมถึงสวิทช์, รีเลย์, solenoids ไฟ LED แสดงของเหลวใสขนาดเล็กหรือที่กำหนดเอง, อุปกรณ์คลื่นความถี่วิทยุและเซ็นเซอร์สำหรับข้อมูลเช่นอุณหภูมิความชื้นระดับแสง ฯลฯ ระบบฝังตัวมักจะมีแป้นพิมพ์หน้าจอ, ดิสก์ เครื่องพิมพ์หรืออุปกรณ์ที่เป็นที่รู้จักของ I / O อื่น ๆ ของคอมพิวเตอร์ส่วนบุคคลและอุปกรณ์ที่อาจจะขาดการปฏิสัมพันธ์ของมนุษย์ใด ๆ

โครงสร้างของบอร์ด Arduino uno R3



รูปที่ 2.2 Arduino Uno R3

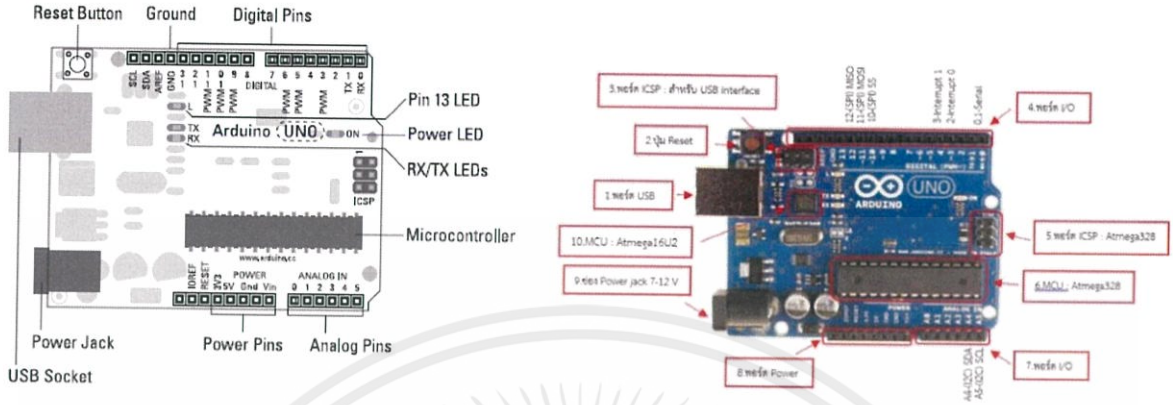
Arduino UNO นั้นก็มีออกมาหลายรุ่นด้วยกัน โดยมีพัฒนาการมาเรื่อยๆตั้งแต่ Arduino UNO รุ่นดั้งเดิม Arduino UNO R2 และพัฒนาจนมาเป็นArduino UNOรุ่นปัจจุบัน นั่นก็คือ Arduino UNO R3 ซึ่ง “R3” นี้แสดงถึงรุ่นที่ได้ทำการแก้ไขปรับปรุงเป็นครั้งที่3 (Third Revision) นั่นเอง

นอกจากนั้นแล้ว Arduino UNO R3 เอง ก็แบ่งย่อยออกเป็น 2 แบบ ตามชนิดของชิป Microcontroller ที่ใช้ ได้แก่ Arduino UNO R3 แบบธรรมดา จะใช้ชิปMCU แบบ DIP (Dual Inline Package) ซึ่งเป็นชิปที่เสียบเข้ากับ Socket อีกทีสามารถถอดเปลี่ยนได้ และ Arduino UNO R3 SMD จะใช้ชิปMCU แบบ SMD (Surface Mount Device) ซึ่งเป็นชิปที่ถูกบัดกรีติดลงบนบอร์ดเลย บอร์ดลักษณะนี้จะไม่สามารถถอดเปลี่ยนชิปได้ ซึ่งบอร์ดแบบSMDจะมีต้นทุนที่ถูกกว่าจึงทำให้ราคาถูกกว่าบอร์ดแบบDIP แต่ทั้งสองบอร์ดก็มีฟังก์ชัน สเปคและการใช้งานที่เหมือนกันทุกประการ

Arduino UNO R3 Specification

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P)of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

ส่วนประกอบหลักของบอร์ด Arduino UNO R3



รูปที่2.3 ส่วนประกอบหลักของบอร์ด Arduino Uno R3

Microcontroller

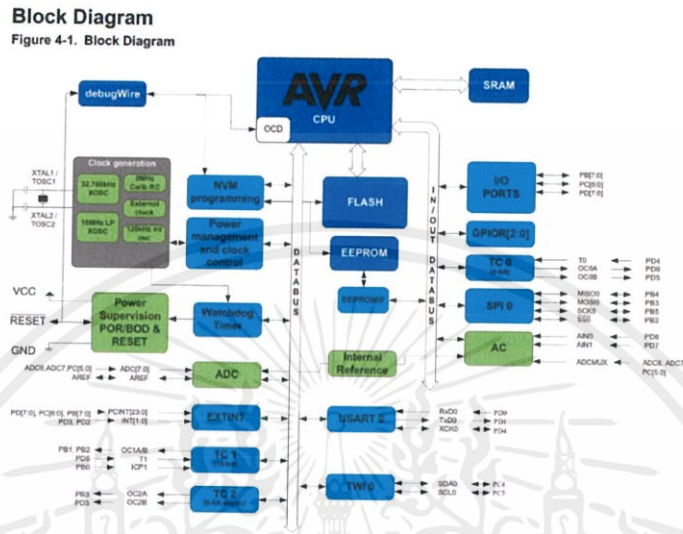
Arduino UNO R3 จะใช้ชิป Microcontroller เป็นชิป ผลิตโดยบริษัท Atmel ซึ่งส่วนนี้จะเป็นเหมือนสมองของบอร์ด ที่ใช้สำหรับการประมวลผลและควบคุม I/O

รูปที่ 2.4 Atmega 328P-PU

คุณสมบัติ

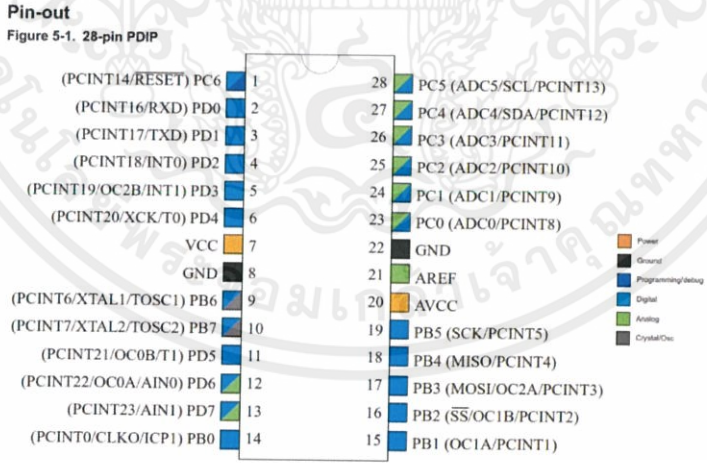
- ขนาด 8 บิท รันได้maxที่ 20 MHz แต่ที่ลงบอร์ดส่วนใหญ่ใช้ 16 MHz ความเร็วในการทำงานคำสั่งใกล้เคียงกับความถี่ที่ใส่
- 2 Timer/Counter 8บิท และ1ตัวที่16บิท ในโหมดต่างๆ
- มีชุดคำสั่ง ภาษาเครื่อง 131 คำสั่ง มีรีจิสเตอร์ทั่วไปที่ใช้งานคำสั่งทางคณิตศาสตร์ได้ 32ตัว*8บิท และ2 รีจิสเตอร์อิสระที่ทำงาน1ครั้ง/1cycle

- 23 ขา I/O ที่โปรแกรมได้ มีหลายขา6-8ขา(ขึ้นกับโมเดลMCU) ที่เป็น ADC (Analog to digital conversion) ดิจิตอลขนาด10บิท(10bit 15KSPS) แปลงอนาล็อกเป็นดิจิตอล10บิทได้ เข้าใจว่าแปลงโวลท์ชนิดอนาล็อก0-5โวลท์ที่อินพุทเข้าที่ขาI/O



รูปที่ 2.5 Blog diagram Atmaga 328P-PU

pin ชนิด 28ขา หรือขาแต่ละขา สั่งหรือรับอะไรได้บ้าง ของ ATmega328p ตัดมาจาก pdf ของผู้ผลิต



รูปที่ 2.6 แสดงช่องpin out ของ ATmega328P-PU

Header Sockets

เป็น Socket ที่เชื่อมต่อมาจากขาของ Microcontroller ซึ่งเรียงอยู่ตรงขอบๆของบอร์ด Arduino ไว้สำหรับต่อสายไฟเพื่อรับค่า Input หรือส่งออก Output โดยจะมีLabel กำกับหมายเลขPinอยู่เพื่อให้สะดวกต่อการใช้งาน โดย Header Socket จะแบ่งออกเป็น 3 กลุ่มหลัก คือ Digital Pin, Analog in Pin และ Power Pin

Digital Pin

เป็นPinสำหรับรับและส่งสัญญาณที่เป็น Digital โดยมี 2 สถานะ คือ On (0V) หรือ Off (5V)

Analog in Pin

เป็น Pin สำหรับรับค่าสัญญาณที่เป็น Analog

Power Pin

เป็น Pin สำหรับจ่ายไฟให้กับอุปกรณ์ต่างๆ ซึ่งจะมีทั้ง 5V และ 3.3V และ Vin โดยVinจะให้ค่าความต่างศักย์เท่ากับไฟที่ต่อมาจาก external power jack

USB Socket

ใช้ในการเชื่อมต่อสาย USB เพื่ออัพโหลดโปรแกรมลงชิป และจ่ายไฟให้กับบอร์ด

External power jack

เป็นช่องสำหรับ Power Adapter เพื่อต่อไฟจากภายนอก

LED

L pin 13(LED on board) เป็น ledที่อยู่บนบอร์ดเพื่อเช็คสถานะการทำงาน

ON: Power LED เป็นไฟสถานะแสดงไฟเลี้ยงของบอร์ด

RX,TX เป็นไฟแสดงว่าขณะนี้บอร์ดกำลังรับหรือส่งข้อมูลอยู่

Reset Bottom

เป็นปุ่มสำหรับ Reset โปรแกรมบนบอร์ดให้หยุดการทำงานเดิมและเริ่มต้นทำงานใหม่ตั้งแต่แรก

Programming

Arduino Uno สามารถรองรับการโปรแกรมด้วย Arduino Software โดยสามารถใช้ได้ทั้งในระบบปฏิบัติการ Windows , Mac OS X และ Linux

2.3. อุปกรณ์ที่ใช้ในการทำบอร์ด arduino(stand alone)

อุปกรณ์	จำนวน
Arduino UNO R3 พร้อม MCU บนตัว	1
MCU ATmega328P	1
Crystal กำเนิดความถี่ 16.00 MHz	1
ตัวเก็บประจุ 22 pF	2
ตัวต้านทาน 10 K โอห์ม	1
Protoboard	1
สายไฟจัมเปอร์	ตามความเหมาะสม

ขั้นตอนที่ 1 : Burn Bootloader

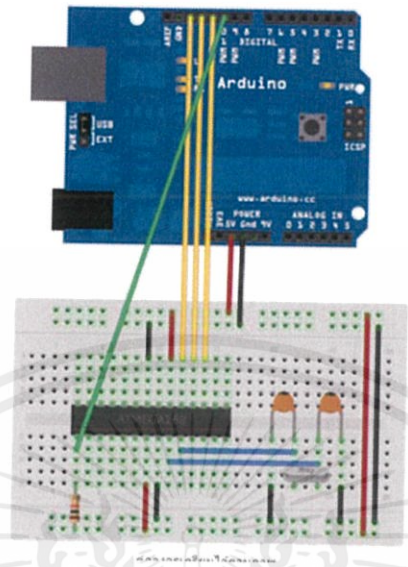
เป็นขั้นตอนในการเปลี่ยนไมโครคอนโทรลเลอร์ทั่วๆไป ให้สามารถใช้งานกับ Code ของ Arduino ได้ โดย Arduino UNO R3 นั้น ใช้ Microcontroller รุ่น ATmega328P ซึ่งเป็นไมโครคอนโทรลเลอร์ของ Atmel แต่อยู่ๆถ้านำมาใช้เลย ก็จะไม่สามารถใช้กับ Arduino ได้ จึงต้องมีการ Burn Bootloader ก่อนนั่นเอง โดย Microcontroller ที่ติดมากับ UNO R3 ตั้งแต่แรก จะมีการ burn ไว้อยู่แล้ว แต่ไมโครคอนโทรลเลอร์ที่แยกซื้อมาต่างหาก อาจจะยังไม่ได้ burn และถึงแม้ว่าจะ burn ไว้แล้ว ก็สามารถทำซ้ำอีกรอบได้เช่นกัน

การBurn Boothloader คือสำหรับไมโครคอนโทรลเลอร์ทั่วๆไป การที่จะโปรแกรมหรืออัปโหลด Code ลงบนตัวมันได้ จะต้องใช้เครื่องโปรแกรม หรืออาจเรียกว่าเครื่องเบิร์น แต่สำหรับ MCU ของ Arduino หรือ MCU บางรุ่น จะมีชุดโปรแกรมเล็กๆ หรือที่เรียกว่า Firmware ซึ่งเป็นโปรแกรมที่จะทำหน้าที่ upload code โดยไม่จำเป็นต้องมีเครื่องเบิร์น และเราเรียก Firmware นั้นว่า Bootloader นั่นเอง ผู้ที่ใช้ Arduino จึงไม่จำเป็นต้องมีเครื่องเบิร์น แต่เราก็จะต้องเบิร์น Bootloader ลงบนตัว MCU ก่อน

ขั้นตอนการ Burn Bootloader

1. ต่อวงจร

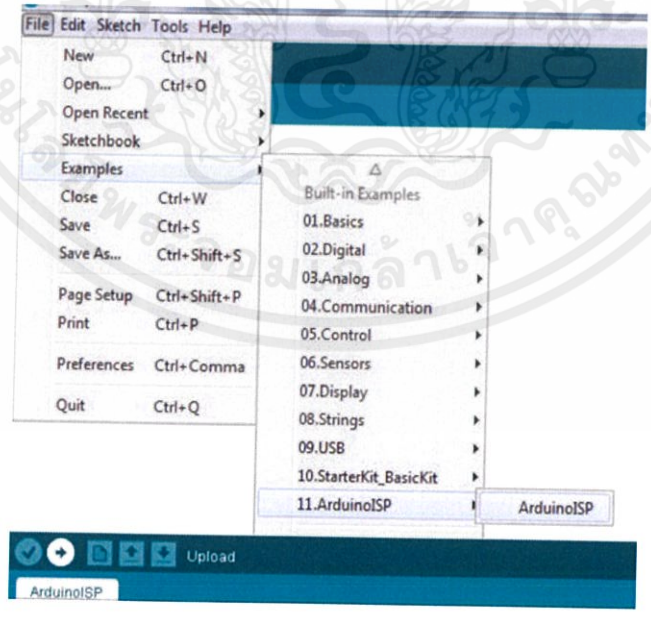
2. Upload code และ Burn ได้เลย



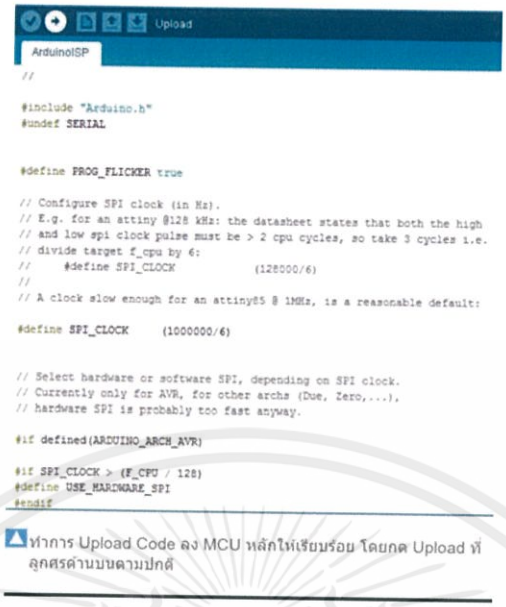
รูปที่ 2.7 การเชื่อมต่อArduinoบนบอร์ด Stand alone

โดยเราจะเรียก MCU บนArduino ว่า MCU หลัก และเรียก MCU บน Protoboard ว่า MCU เป้าหมาย (MCU ที่กำลังจะโดนเบิร์น)

ทำการ Upload Code Arduino ISP ลงบน MCU หลักก่อน ซึ่งเป็น Code ที่จะใช้ในการ Burn bootloader ให้ MCU เป้าหมาย โดยไปที่ Examples > ArduinoISP

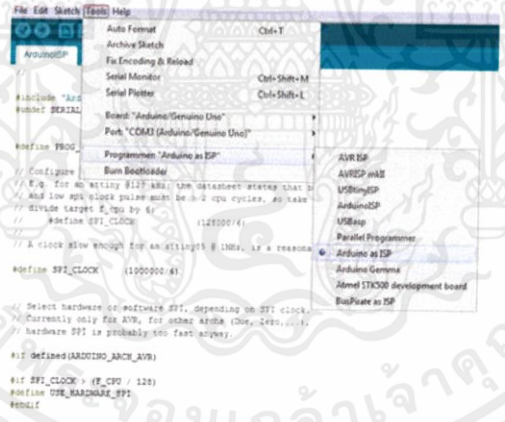


รูปที่ 2.8 การBurn Booth loader



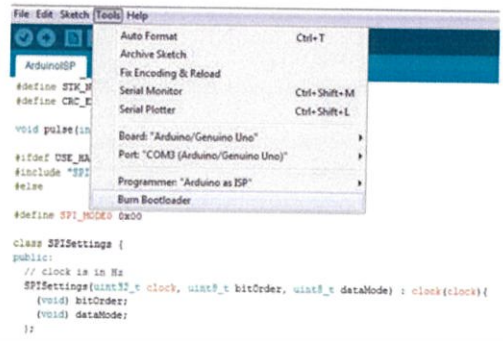
รูปที่ 2.9 การBurn Booth loader

ทำการ Upload Code ลง MCU หลักให้เรียบร้อย โดยกด Upload ที่ลูกศรด้านบนตามปกติ



รูปที่ 2.10 การBurn Booth loader

จากนั้น เปลี่ยนประเภทการ Upload ให้เป็น Arduino as ISP (จากเดิมจะเป็น Parallel Programmer) เพื่อเปลี่ยนจากบอร์ด Arduino ให้กลายเป็นเครื่องเบิร์น bootloader ชั่วคราว



รูปที่ 2.11 การBurn Booth loader

จากนั้น กด Burn Bootloader และรอจนกระทั่งการเบิร์นเสร็จสิ้น MCU เป้าหมายก็จะมี Bootloader ฝังอยู่ในตัวเอง เพื่อรอการ Upload Code ใหม่ๆในขั้นตอนต่อไป

2.3.2 การนำ MCU แบบ Standaloneไปใช้

หลังจากที่ได้ MCU แบบ Standalone มาแล้ว การที่เราจะนำไปใช้งานได้ ก็จะต้องมีการต่อวงจรเล็กๆเพิ่ม อีก 2 วงจร ได้แก่

1.วงจรกำเนิดความถี่

2.วงจรไฟเลี้ยง

1.วงจรกำเนิดความถี่

การใช้งานไมโครคอนโทรลเลอร์ทั่วๆไป จะต้องมีวงจรกำเนิดความถี่ให้กับมัน เพราะความถี่นี้ จะเป็นตัวกำหนดจังหวะการทำงานและการนับเวลาต่างๆของ MCU

วงจรกำเนิดความถี่ ไม่มีอะไรซับซ้อน มีอุปกรณ์ดังนี้

อุปกรณ์	จำนวน
Crystal กำเนิดความถี่ 16.00 MHz	1 ตัว
ตัวเก็บประจุ 22 pF	2 ตัว

รูปที่ 2.12 อุปกรณ์กำหนดความถี่

โดยการนำ Crystal ต่อเข้าที่ขา 9 และ 10 ของ MCU ATmega328P (ต่อด้านไหนก็ได้ เพราะ Crystal ไม่มีขั้ว) และนำตัวเก็บประจุจัมป์ระหว่างขาทั้ง 2 ของ Crystal กับ Ground

2.วงจรไฟเลี้ยง

เป็นอีกวงจรที่สำคัญมาก เพราะก่อนอื่นต้องรู้ว่า MCU ATmega328P รับไฟเลี้ยงได้ไม่เกิน 5.5 V แต่ในบางครั้ง เราอาจต้องการใช้กับไฟเลี้ยง หลายๆแบบที่หามาได้ เช่น 5 V , 9 V หรือ 12 V เป็นต้น ดังนั้น เราจึงจำเป็นต้องมี IC Voltage Regulator เพื่อจำกัดแรงดันให้ไม่เกิน 5 V เพื่อประสิทธิภาพสูงสุดของ MCU และป้องกันความเสียหายจากไฟแรงดันเกิน

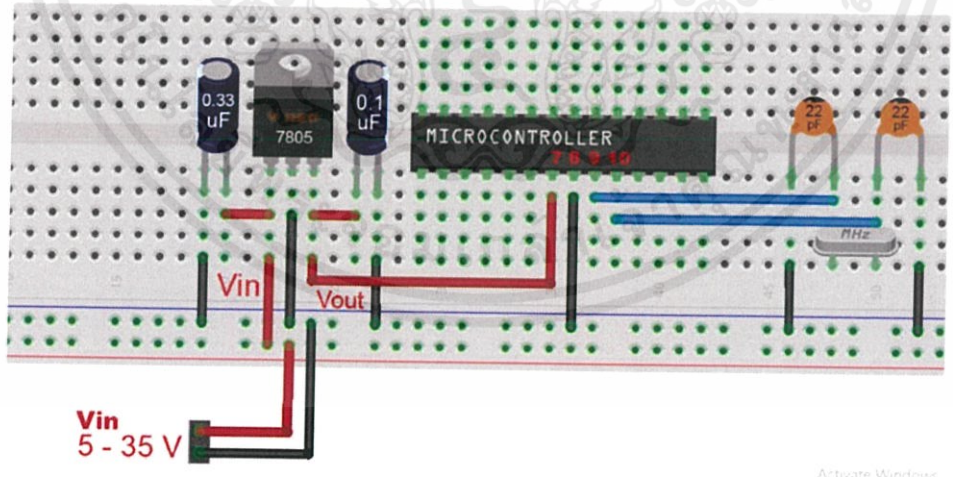
โดยมีอุปกรณ์ดังนี้

อุปกรณ์	จำนวน
IC 7805 Voltage Regulator	1 ตัว
ตัวเก็บประจุ 0.1 uF	1 ตัว
ตัวเก็บประจุ 0.33 uF	1 ตัว

รูปที่ 2.13 อุปกรณ์ของวงจรไฟเลี้ยง

* ตัวเก็บประจุจะเพิ่มความเสถียรและความเรียบของสัญญาณไฟฟ้าได้มาก

IC 7805 Voltage Regulator ทำหน้าที่แปลงแรงดันขาเข้า จาก 5 - 35 V ให้กลายเป็นแรงดันขาออก 4.8 - 5.2 V นั่นหมายความว่า เราสามารถใช้แหล่งจ่ายได้หลากหลายตั้งแต่ 5 - 35 V



วงจรกำเนิดความถี่และไฟเลี้ยงของ MCU Arduino Standalone

รูปที่ 2.14 ทดลองการทำงานบอร์ด Stand alone บน โฟโต้บอร์ด

2.4 Servo motor

รูปที่ 2.15 Servo motor

เซอร์โวมอเตอร์ (Servo Motor) เป็นมอเตอร์ที่มีการควบคุมการเคลื่อนที่ของมัน (State) ไม่ว่าจะเป็นระยะ ความเร็ว มุมการหมุน โดยใช้การควบคุมแบบป้อนกลับ (Feedback control) เป็นอุปกรณ์ที่สามารถควบคุมเครื่องจักรกล หรือระบบการทำงานนั้นๆ ให้เป็นไปตามความต้องการ เช่น ควบคุมความเร็ว (Speed), ควบคุมแรงบิด (Torque), ควบคุมแรงตำแหน่ง (Position), ระยะทางในการเคลื่อนที่(หมุน) (Position Control) ของตัวมอเตอร์ได้ ซึ่งมอเตอร์ทั่วไปไม่สามารถควบคุมในลักษณะงานเบื้องต้นได้ โดยให้ผลลัพธ์ตามความต้องการที่มีความแม่นยำสูง

ประเภทของเซอร์โวมอเตอร์

โดยทั่วจะมีทั้งดีซีและเอซีเซอร์โว ในเครื่องจักรรุ่นเก่าๆเราจะพบว่า DC Servo Motor มีการใช้เครื่องจักรกลอุตสาหกรรมมากกว่า AC Servo Motor เนื่องจากช่วงที่ผ่านมาการควบคุมกระแสกระแสสูงๆนั้นจะต้องใช้ SCRs แต่ปัจจุบันทรานซิสเตอร์ได้พัฒนาขีดความสามารถให้ตัดต่อกระแสสูงและใช้งานที่ความถี่ได้สูงๆขึ้น จึงทำให้ระบบควบคุมทางเอซีและระบบเซอร์โวได้ถูกนำมาใช้งานมากขึ้น ซึ่งสามารถแยกประเภทของเซอร์โวได้ดังนี้

1. มอเตอร์ชนิดที่มีแปรงถ่าน

เซอร์โวมอเตอร์ชนิดนี้ที่สเตเตอร์จะเป็นแม่เหล็กถาวร ส่วนโรเตอร์ยังใช้แปรงถ่านและคอมมิวเตอร์เรียงกระแสเข้าสู่ขดลวดอาร์เมเจอร์ เหมือนกับดีซีมอเตอร์ทั่วไป

2. เซอร์โวมอเตอร์ชนิดที่ไม่มีแปรงถ่าน

เซอร์โวมอเตอร์ในกลุ่มนี้ประกอบด้วยดีซีเซอร์โว (DC Brushless Servo โรเตอร์ทำด้วยแม่เหล็กถาวร) เอซีเซอร์โว (AC Servo) ซึ่งมีทั้งแบบซิงโครนัสเซอร์โว อะซิงโครนัสเซอร์โว (การนำอินดัคชั่นมอเตอร์มาใช้ทำเป็นระบบขับเคลื่อนเซอร์โวมอเตอร์) และ สเตปปิ้งเซอร์โวมอเตอร์

โครงสร้างของเซอร์โวมอเตอร์

ข้อจำกัดอย่างหนึ่งของระบบควบคุมเซอร์โว ก็คือการใช้งานจะต้องเป็นแบบ Closed loopเท่านั้น การใช้งานระบบควบคุมเซอร์โวไม่สามารถเลือกควบคุมเป็นแบบ Open loop ได้เหมือนกันระบบขับเคลื่อนเอซี (AC Drives) การตอบสนองของระบบเซอร์โว เช่น อัตราเร่ง แรงบิด และตำแหน่งที่ควบคุม จะไม่เป็นไปตามวัตถุประสงค์หากไม่มีสัญญาณป้อนกลับไปยังชุดขับเคลื่อนเซอร์โว

การควบคุมการทำงานในระบบนี้อุปกรณ์ป้อนกลับหรือเอ็นโค๊ดเตอร์ (Encoder) จะมีบทบาทความสำคัญอย่างยิ่งเสมือนกับเป็นของคู่กันชนิดที่เรียกว่าขาดซึ่งกันและกันไม่ได้ ในทางปฏิบัติจึงทำเซอร์โวมอเตอร์และเอ็นโค๊ดเตอร์ ถูกออกแบบและผลิตสร้างขึ้นมาคู่กันในลักษณะเป็นแพ็คเกจ (Package) ซึ่งมี Encoder ติดอยู่ที่ส่วนท้ายของมอเตอร์

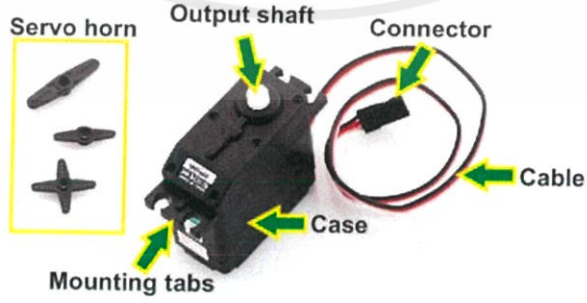
ข้อดีของ Servo motor ได้แก่

- สามารถให้ค่าทอร์กที่สูง
- สามารถเคลื่อนที่ความเร็วสูง
- ใช้งานกับการควบคุมความเร็วได้ดี
- มีหลากหลายขนาดให้เลือก (มากกว่า Stepper motor)
- เงียบ ไม่เหมือน Stepper motor
- ควบคุมได้ดีกว่า Stepper และไม่มีการสะดุด

ข้อเสียของ Servo Motor ได้แก่

- แพงว่า Stepper motor
- ไม่สามารถทำงานโดยการควบคุมแบบเปิด
- ต้องมีการจูนค่าในการควบคุม (สำหรับ Servo ที่มีราคาสูงกว่าแบบที่ใช้กับมือสมัครเล่น)
- ในกรณีที่ใช้ DC motor ต้องมีการบำรุงรักษา เนื่องจากแปรงถ่านอาจสึก

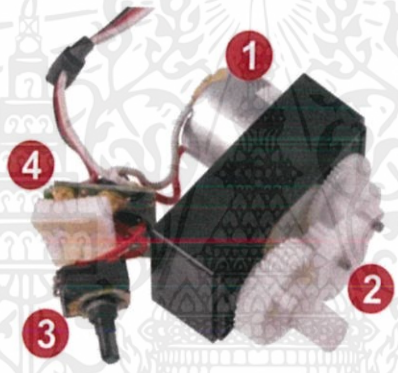
ส่วนประกอบของServo motor



รูปที่ 2.16 ส่วนประกอบของ 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
    - o สายสีแดง คือ ไฟเลี้ยง (4.8-6V)
    - o สายสีดำ หรือ น้ำตาล คือ กราวด์
    - o สายสีเหลือง (ส้ม ขาว หรือฟ้า) คือ สายส่งสัญญาณพัลซ์ควบคุม (3-5V)
- Connector จุดเชื่อมต่อสายไฟ

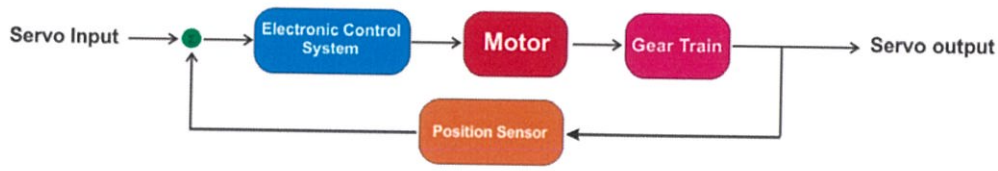
ส่วนประกอบภายใน RC Servo Motor



รูปที่ 2.17 ส่วนประกอบภายในServo motor

1. Motor เป็นส่วนของตัวมอเตอร์
2. Gear Train หรือ Gearbox เป็นชุดเกียร์ทดแรง
3. Position Sensor เป็นเซ็นเซอร์ตรวจจับตำแหน่งเพื่อหาค่าองศาในการหมุน
4. Electronic Control System เป็นส่วนที่ควบคุมและประมวลผล

Servo Motor Block Diagram



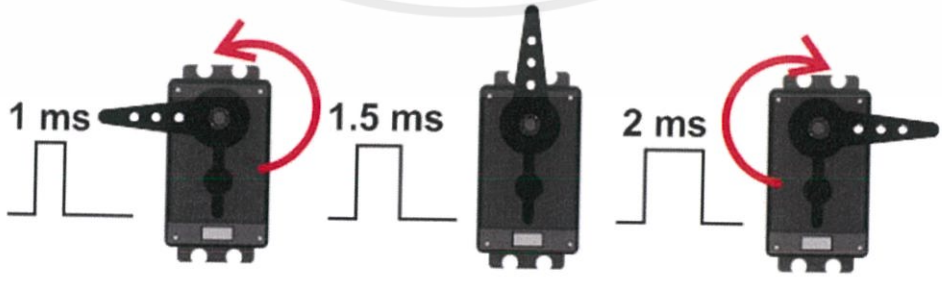
รูปที่ 2.18 Servo Motor Block Diagram

หลักการทำงานของ RC 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

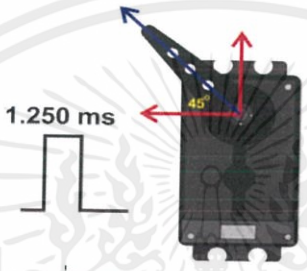
ตัว RC Servo Motor ออกแบบมาใช้สำหรับรับคำสั่งจาก Remote Control ที่ใช้ควบคุมของเล่นด้วยสัญญาณวิทยุต่างๆ เช่น เครื่องบินบังคับ รถบังบังคับ เรือบังคับ เป็นต้น ซึ่ง Remote จำพวกนี้ที่ภาครับจะแปลงความถี่วิทยุออกมาในรูปแบบสัญญาณ PWM (Pulse Width Modulation)มุมหรือองศาจะขึ้นอยู่กับความกว้างของสัญญาณพัลซ์ ซึ่งโดยส่วนมากความกว้างของพัลซ์ที่ใช้ใน 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 2.19 การหมุนของ Servo motor

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 2.20 การหมุนของ Servo motor

วิธีควบคุม RC Servo Motor ด้วย Arduino

Arduino มีไลบรารี่สำหรับสั่งงาน RC Servo Motor มาให้ใช้งานอยู่แล้วเป็นฟังก์ชั่นสำเร็จรูปและใช้งานได้ง่ายในหน้าเว็บไซต์ http://arduino.cc/en/reference/servo ได้ให้ข้อมูลไว้ว่า Servo Library ของ Arduino สามารถสั่งงาน RC Servo Motor ได้ทั้งแบบหมุนไป-กลับได้ 0-180 องศา (ที่กล่าวถึงตามตัวอย่างข้างต้น) และแบบต่อเนื่องที่หมุนครบรอบได้เรียกว่าเป็น Continuous Rotation Servo (ซึ่งในช่วงท้ายบทความจะกล่าวถึงเพิ่มเติม) โดยสามารถรองรับการเชื่อมต่อ RC Servo Motor ได้ถึง 12 ตัวกับบอร์ด Arduino UNO และรองรับสูงสุดถึง 48 ตัวหากใช้บอร์ด Arduino Mega

ฟังก์ชั่นภายใน Servo Library

- attach()
- write()
- writeMicroseconds()
- read()
- attached()
- detach()

attach()

Description

คือฟังก์ชั่นที่ใช้ในการกำหนดขาสัญญาณที่ Servo Motor ต่อกับ Arduino และกำหนดความกว้างของพัลซ์ที่ 0 องศาและ 180 องศา

Syntax

Servo.attach(pin)

Servo.attach(pin,min,max)

Parameters

Pin: คือ ขาสัญญาณของ Arduino ที่ใช้เชื่อมต่อกับ Servo Motor

Min: คือ ความกว้างของพัลซ์ที่ 0 องศาของ Servo ตัวที่ใช้ในหน่วยไมโครวินาที (us) โดยปกติแล้วหากไม่มีการตั้งค่าโปรแกรมจะกำหนดค่าไว้ที่ 544 us

Max: คือ ความกว้างของพัลซ์ที่ 180 องศาของ Servo ตัวที่ใช้ในหน่วยไมโครวินาที (us) โดยปกติแล้วหากไม่มีการตั้งค่าโปรแกรมจะกำหนดค่าไว้ที่ 2400 us

Write()

Description

คือฟังก์ชั่นที่ใช้ควบคุมตำแหน่งที่ต้องการให้ Servo Motor หมุนไปยังองศาที่กำหนดสามารถกำหนดเป็นค่าองศาได้เลย คือ 0-180 องศา แต่ใน Servo Motor ที่เป็น Full Rotation คำสั่ง write จะเป็นการกำหนดความเร็วในการหมุน โดย

ค่าเท่ากับ 90 คือคำสั่งให้ Servo Motor หยุดหมุน

ค่าเท่ากับ 0 คือการหมุนด้วยความเร็วสูงสุดในทิศทางหนึ่ง

ค่าเท่ากับ 180 คือการหมุนด้วยความเร็วสูงสุดในทิศทางตรงกันข้าม

Syntax

servo.write(angle)

Parameters

Angle: คือมุมที่ต้องการให้ RC Servo Motor แบบ 0-180 องศาหมุนไป แต่หากเป็น RC Servo Motor แบบ Full Rotation ค่า Angle คือ การกำหนดความเร็วและทิศทางในการหมุน

writeMicroseconds()

Description

คือฟังก์ชั่นที่ใช้ควบคุมตำแหน่งที่ให้ Servo Motor หมุนไปยังตำแหน่งองศาที่กำหนดโดยกำหนดเป็นค่าความกว้างของพัลซ์ในหน่วย us ซึ่งปกติแล้ว RC Servo Motor จะใช้ความกว้างของพัลซ์อยู่ที่ 1000-2000 us ตามที่ได้กล่าวไปข้างต้นแล้ว แต่ RC Servo Motor บางรุ่นหรือบางยี่ห้อไม่ได้ใช้ ช่วงความกว้างของพัลซ์ตามที่ได้กล่าวเอาไว้นี้ อาจจะใช้ช่วง 700-2300 แทนก็สามารถใช้ฟังก์ชั่น writeMicroseconds นี้เพื่อกำหนดความกว้างพัลซ์ได้เอง

//การใช้ฟังก์ชั่น writeMicroseconds สามารถกำหนดค่าได้อิสระ ตรงนี้ "ต้องระวังในการใช้งาน" หากสั่งงาน RC Servo Motor (แบบ 0 - 180 องศา) จนหมุนไปเกินจุดสิ้นสุดคือเกินทั้งฝั่ง 0 หรือ 180 องศา จะทำให้เกิดเสียงคร่างดังจากการหมุนไปต่อไม่ได้และมอเตอร์จะกินกระแสสูงขึ้นด้วยในเวลาเดียวกันนั้น ซึ่งอาจทำให้ RC Servo Motor เกิดความเสียหายได้

Syntax

servo.writeMicroseconds(uS)

Parameters

uS: คือค่าความกว้างของพัลซ์ที่ต้องการกำหนดในหน่วยไมโครวินาที (โดยตัวแปร int)

read()

Description

คือฟังก์ชั่นอ่านค่าองศาที่สั่งเข้าไปด้วยฟังก์ชั่น write() เพื่อให้รู้ว่าตำแหน่งองศาสุดท้ายที่เราสั่งเข้าไปนั้นมีค่าเท่าไหร่ซึ่งค่าที่อ่านออกมานั้นจะมีค่าอยู่ในช่วง 0 - 180

Syntax servo.read()

Parameters

ไม่มี: จะ Return ค่า 0-180

attached()

Description

คือฟังก์ชั่นตรวจสอบว่า Servo ที่เราต้องการใช้กำลังต่ออยู่กับขสัญญาณของ Arduino หรือไม่

Syntax

servo.attached()

Parameters

ไม่มี: จะ Return ค่า True ออกมา หาก Servo Motor เชื่อมต่ออยู่กับ Arduino แต่ถ้าหาก Return ออกมาเป็นค่าอื่นถือว่าไม่เชื่อมต่อ

detach()

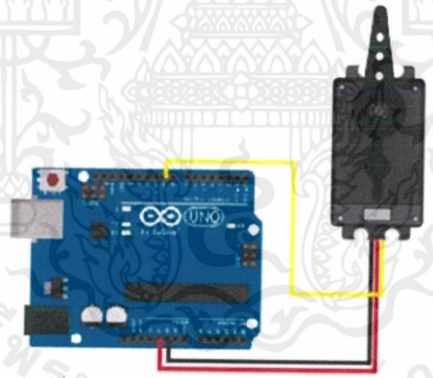
Description

คือฟังก์ชั่นคืนสถานะของขาที่เรากำหนดให้เป็นขาควบคุม Servo Motor ด้วยคำสั่ง attached() ให้กลับคือสู่การใช่งานปกติ

Syntax

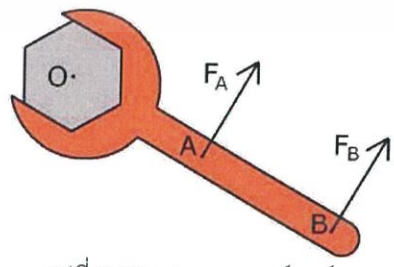
servo.detach()

ตัวอย่างการเชื่อมต่อ RC Servo Motor เข้ากับบอร์ด Arduino



รูปที่ 2.21 การต่อเซอร์โวเข้าบอร์ดArduino

2.5 การคำนวณหาค่า Toque

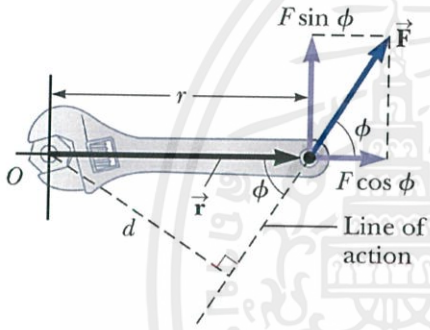


รูปที่ 2.22 แสดงเวคเตอร์ทอร์ค

ทอร์กหรือแรงบิดคืออะไร

ทอร์กคือแรงที่พยายามจะหมุนมวลคุณสามารถสร้างทอร์กได้ด้วยตนเองโดยการใช้ประแจขันน็อตแรงที่คุณกระทำกับด้ามจับคือทอร์กที่พยายามหมุนน็อตให้แน่นนั่นเองหน่วยอังกฤษของทอร์กคือ ปอนด์ - นิ้ว หรือ ปอนด์ - ฟุต หน่วย SI คือ นิวตัน - เมตร หน่วยของทอร์กเกิดจากแรงคูณด้วยระยะทางถ้าคุณจะหาทอร์กคุณจะต้องคูณแรงที่กระทำกับระยะทางที่วัดห่างจากจุดหมุนในแนวตั้งฉาก ในกรณีของการขันน็อตถ้าประแจมีด้ามยาว 1 ฟุตและคุณออกแรงขนาด 200 ปอนด์ ในแนวตั้งฉากกับด้ามทอร์กที่คุณได้คือ 200 ปอนด์-ฟุตแต่ถ้าคุณใช้ประแจที่มีด้ามยาว 2 ฟุตคุณใช้แรงเพียง 100 ปอนด์เพื่อสร้างทอร์กขนาดเดียวกันเครื่องยนต์จะต้องสร้างทอร์กเพื่อหมุนเพลาข้อเหวี่ยง ขนาดทอร์ก เท่ากับ ขนาดของแรงคูณด้วยระยะห่างจากแกนหมุนซึ่งตั้งฉากกับแนวแรง

จากรูป เรานิยามขนาดของทอร์ก ที่สอดคล้องกับแรง $\boldsymbol{F}$ รอบแกนที่ผ่านจุด O ด้วยสมการดังนี้



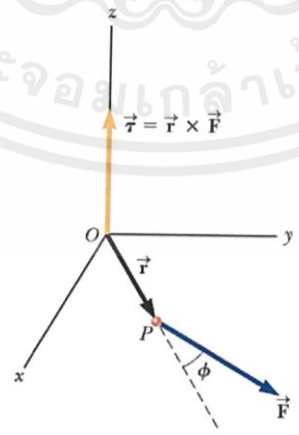
รูปที่2.23 การคำนวณหาแรงบิดทอร์ค

$$\tau = RF\sin\theta$$

เมื่อ $\boldsymbol{r}$ คือ ระยะระหว่างแกนหมุนกับจุดที่แรงกระทำ

เวคเตอร์ของทอร์ค

เวกเตอร์ทอร์ก $\boldsymbol{\tau}$มีความสัมพันธ์กับเวกเตอร์ $\boldsymbol{r}$ และ $\boldsymbol{F}$ โดยสามารถหาความสัมพันธ์โดยใช้ตัวดำเนินการทางคณิตศาสตร์ที่เรียกว่า ตัวคูณเวกเตอร์

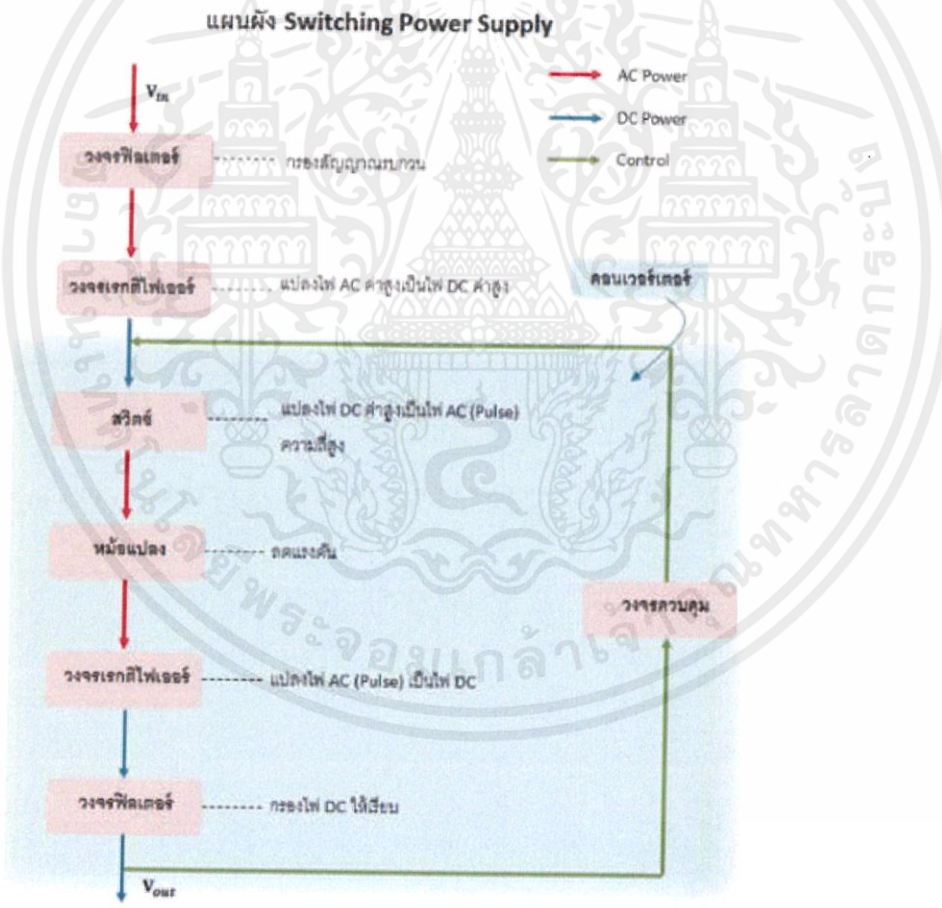


รูปที่ 2.24 เวคเตอร์ของทอร์ค

2.6 วงจรจ่ายไฟ

หลักการทำงาน Switching Power Supplyในปัจจุบัน ได้มีการใช้เทคโนโลยีแหล่งจ่ายกำลังสวิตชิ่งกันอย่างแพร่หลาย ซึ่ง Switching Power Supply นั้นถูกสร้างขึ้นมาเพื่อใช้ในงานอิเล็กทรอนิกส์ เป็นแหล่งจ่ายไฟให้กับอุปกรณ์ต่างๆ และสามารถเปลี่ยนแรงดันไฟจากไฟสลับโวลต์สูงให้เป็นแรงดันไฟตรงโวลต์ต่ำได้ ซึ่งองค์ประกอบพื้นฐานนั้นโดยทั่วไปจะคล้ายกันและสิ่งที่สำคัญที่สุดขององค์ประกอบนี้คือ คอนเวอร์เตอร์
Switching Power Supply จะประกอบด้วย 3 ส่วนใหญ่ๆ คือ

- วงจรฟิลเตอร์และเรกติไฟเออร์ ทำหน้าที่แปลงแรงดันไฟสลับเป็นไฟตรง
- คอนเวอร์เตอร์ ทำหน้าที่แปลงไฟตรงเป็นไฟสลับความถี่สูง และแปลงกลับเป็นไฟตรงโวลต์ต่ำ
- วงจรควบคุม ทำหน้าที่ควบคุมการทำงานของคอนเวอร์เตอร์ เพื่อให้ได้แรงดันเอาต์พุตตามต้องการ



รูปที่ 2.25 องค์ประกอบภาคจ่ายไฟ

และที่เราเลือกใช้วงจรจ่ายไฟ แบบ step down converter (สวิตชิ่งพาวเวอร์ซัพพลาย)

รูป 2.26 สวิตชิ่งพาวเวอร์ซัพพลาย

รายละเอียด

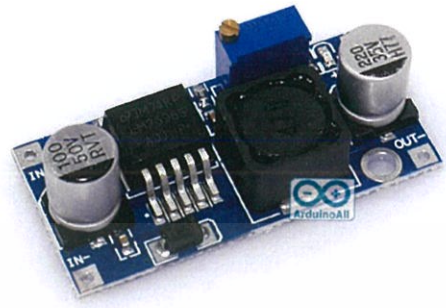
- ระบบตัดไฟอัตโนมัติ เมื่อมีการช๊อตวงจร
- 1x Switching power supply AC 100-240V to DC 12V 5A 60W module
- แหล่งจ่ายไฟแบบสวิตชิ่ง
- แรงดันอินพุต :100-240VAC
- แรงดันเอาต์พุต: 12Vdc
- กระแสเอาต์พุต: 20A
- กำลังเอาต์พุต : 240W

-ขนาด 110*78*36mm

หรือทางเลือกวงจรจ่ายไฟที่เราสามารถเลือกใช้ได้คือถ่าน Panasonic Neo 9V

รูปที่ 2.27 ถ่านไฟฉาย 6F22NT/1SL 9V

2.6.1 LM2596 Converter Buck Step Down Regulator Power Module



รูปที่ 2.28 lm2596 Module

DC Step Down LM2596 / LM2596S Module (3A)

อุปกรณ์ตัวนี้ทำหน้าที่แปลงระดับแรงดันไฟฟ้าตรงลง โดยสามารถปรับค่าแรงดัน output ได้โดย Potentiometer ที่มีอยู่บนบอร์ด สามารถจ่ายกระแสได้ถึง 3 A และใช้หลักการแปลงโดยวงจร Buck Converter ความถี่ Swithching 150 kHz ทำให้ทำงานเงียบ และแรงดันเรียบ

Features:

Input voltage: 4V-35V

Output voltage: 1.23V-30V

Input current: 3A(maximum)

DC-DC Buck Converter Step Down Module LM2596 Power Supply

Specifications:

Conversion efficiency: 92%(highest)

Switching frequency: 150KHz

Output ripple: 30mA9maxmum)

2.7 User Interface

สิ่งที่มีไว้ให้ผู้ใช้ใช้ในการกระทำกับระบบหรือสิ่งของต่างๆ ซึ่งอาจจะเป็น คอมพิวเตอร์ เครื่องจักร เคลื่องกล อุปกรณ์ไฟฟ้าต่างๆ หรือระบบที่มีความซับซ้อนอื่นๆ เพื่อให้สิ่งนั้นๆทำงานตามต้องการของผู้ใช้

ส่วนต่อประสานกับผู้ใช้สามารถจัดได้เป็น 2 ประเภทใหญ่ได้แก่

- ส่วนที่นำข้อมูลเข้า หรือส่วนสั่งงาน เรียกว่า อินพุต (input)
- ส่วนที่ใช้แสดงผลลัพธ์ หรือส่วนที่ไว้รอคำสั่งจากผู้ใช้ เรียกว่า เอาต์พุต (output)

นักออกแบบ UI ในปัจจุบันมีโอกาสเกือบจะไร้ขีดจำกัด ในการทำงานบนเว็บไซต์ โมบายแอพ เทคโนโลยีอุปการณ์สำหรับสวมใส่ (wearable technology) และอุปกรณ์สมาร์ทโฮมต่างๆ ที่กล่าวมานั้นเป็นเพียงส่วนน้อย ตราบใดที่คอมพิวเตอร์ยังคงเป็นส่วนหนึ่งของชีวิตประจำวันจะมีความจำเป็นที่จะต้องทำให้อินเทอร์เฟซที่ช่วยให้ผู้ใช้ ทุกวัย ทุกเบื้องหลังและประสบการณ์ทางเทคนิค สามารถใช้งานได้อย่างมีประสิทธิภาพ

2.7.1 Visual Basic

Visual Basic เป็นภาษาโปรแกรมภาษาหนึ่ง พัฒนาโดยบริษัทไมโครซอฟท์ บรรจุอยู่ภายใต้ผลิตภัณฑ์ชุดหนึ่งชื่อว่า Microsoft Visual Studioเรานิยมเรียกย่อๆ ว่า VB ภายในชุดของเจ้า Microsoft Visual Studio นี้ ประกอบไปด้วยภาษาต่างๆ หลายภาษา แยกแยะกันไปตามความสามารถและความถนัดได้แก่

1.Visual Basic

2.Visual C++

3.Visual FoxPro

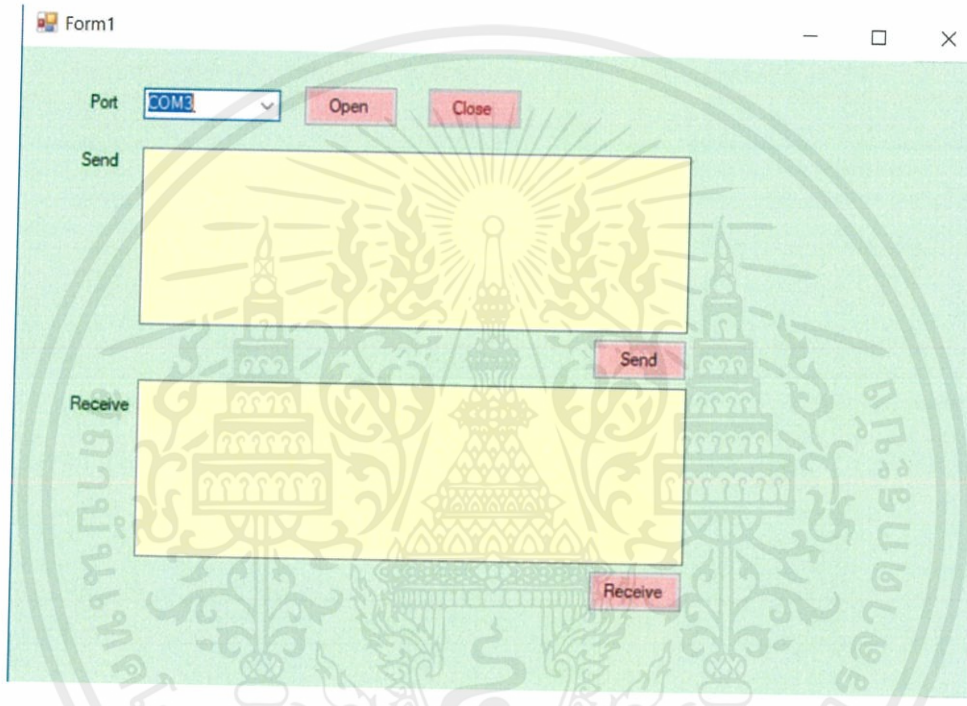
4.Visual InterDev

5.Visual J++

6.Visual SourceSafe)

วัตถุประสงค์ ก็คือ ต้องการให้เป็นภาษาที่ใช้สำหรับพัฒนาโปรแกรมหรือแอปพลิเคชั่น ที่ทำงานภายใต้ระบบปฏิบัติการ Windows และ Windows NTการใช้งาน Visual Basic นั้นค่อนข้างง่าย โปรแกรมเมอร์สามารถวาดและวางองค์ประกอบต่างๆ บนหน้าจอเพื่อติดต่อกับผู้ใช้ (User Interface)ได้ตามต้องการ

เมื่อวาดหน้าจอได้เสร็จก็เขียนโปรแกรมซึ่งเป็นลักษณะแบบมีโครงสร้างทางภาษาคล้ายคลึงกับภาษาอังกฤษ เพื่อเป็นการเชื่อมโยงความสัมพันธ์ขององค์ประกอบแต่ละตัวบนหน้าจอเข้าด้วยกัน ให้ทำงานอย่างสัมพันธ์กัน ตามที่โปรแกรมเมอร์ต้องการ ตามหลักการของ Object-Oriented นั่นเอง



รูปที่2.29 หน้าต่าง UI ที่ได้ทำขึ้นเพื่อติดต่อกับตัวแขนกล

2.7.2 ภาษามือ

ภาษามือ (อังกฤษ: sign language) เป็นการ เป็นอวัจนภาษาอย่างหนึ่ง ที่ประกอบด้วย การสื่อสารด้วยมือ, การสื่อสารด้วยร่างกาย และการใช้ริมฝีปากในการสื่อความหมายแทนการใช้เสียงพูด การสื่อสารจะใช้ลักษณะของมือที่ทำเป็นสัญลักษณ์ การเคลื่อนไหวมือ แขนและร่างกาย และการแสดงความรู้สึกทางใบหน้าเพื่อช่วยในการสื่อสารความคิดของผู้สื่อ ภาษาสัญลักษณ์ส่วนใหญ่มักใช้ในกลุ่มผู้พิการทางหู ซึ่งรวมทั้งผู้พิการทางหูเอง ผู้ตีความหมาย (interpreter) ผู้ร่วมงาน เพื่อน และครอบครัวของผู้พิการทางหูซึ่งอาจจะพอได้ยินบ้างหรือไม่ได้ยินเลย

รูปที่ 2.30ความสัมพันธ์ลักษณะนิ้วมือกับตัวอักษรภษาอังกฤษ

บทที่3

การออกแบบและการสร้างแขนกล

ในบทนี้จะกล่าวถึงการออกแบบแขนกลแบบ Articulated

3.1 การออกแบบโครงสร้างแขนกล

ในการศึกษา และเก็บข้อมูลได้เลือกออกแบบแขนกลเป็นแบบ Articulated ทุกแกนการเคลื่นที่เป็นแบบหมุน รูปแบบการเคลื่อนที่คล้ายมือและแขนของมนุษย์ ซึ่งประกอบด้วยข้อต่อต่างๆตามช่วงนิ้ว มือฝ่ามือ รวมถึงข้อมือ ทำให้การทำงานเสมือนมือของมนุษย์ที่สามารถหยิบของหรือชิ้นงานขึ้นมาได้ การออกแบบแขนกลประกอบด้วย 2 ขั้นตอนใหญ่

-ขั้นตอนแรกคือการจำลองแบบโดยการวาดรูปใช่กระดาษแข็งพร้อมดูถึงโอกาศที่จะสามารถจะทำงานได้ ซึ่งแบบต่างๆที่เราได้ทำมาจำลองนั้นอาจจะต้องลองผิดลองถูกหลายครั้งจนกว่าจะเจอแบบที่คิดว่ามีความเป้นไปได้สูงสุด

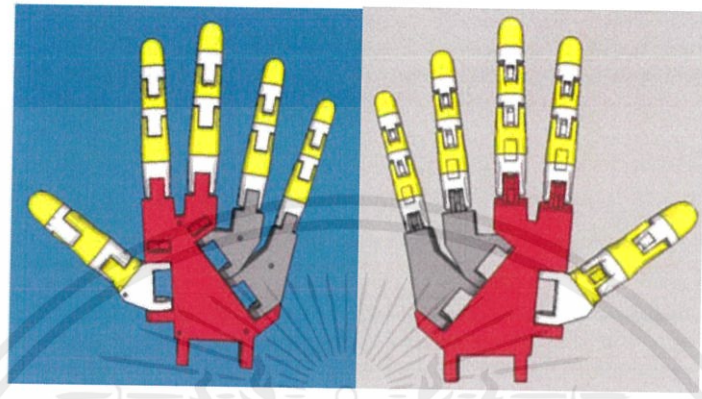
-ขั้นตอนที่ 2 คือการสร้างโครง 3 มิติ ของแขนกลโดยใช้โปรแกรม solid work ในการสร้างแบบจำลองเพื่อให้ได้ขนาดหรือรูปแบบตามที่เราต้องการ

3.2 ขั้นตอนในการออกแบบแขนกล

หลังจากได้ศึกษาการทำแขนกลอย่างคร่าวๆ จากบทความและเว็ปไซต์จึงได้เริ่มลงมือทำโครงงานชิ้นนี้ขึ้นมาโดยเริ่มศึกษาการทำแขนกลจากบทความต่างๆทางของทางWebsite inmoov ที่เคยได้มีคนทำมาแล้วซึ่งเริ่มจากการคำนวณทอร์คตามข้อต่อต่างๆบนข้อต่อของนิ้วแต่ละนิ้วรวมถึงข้อต่อบริเวณข้อมือ เพื่อที่จะสามารถหามอเตอร์ตามแรงบิดที่ส่งตามน้ำหนักที่แบกรับได้ในแต่ละข้อหลังจากออกแบบทอร์คแล้วจึงออกแบบจำลอง(prototype) เพื่อดูลักษณะการหมุนของแต่ละข้อต่อที่จะเป็นไปได้ซึ่งบางข้อต่อน้ำหนักที่ต้องรับทำงานหนักเกินไปก็อาจจะทำให้เฟืองรูดหรือมีแรงบิดไม่มากพอที่จะสามารถดึงเส้นเอ็นที่ยึดติดกับข้อต่อบริเวณนิ้วมือให้งอลงมาได้ ซึ่งเราก็ต้องเปลี่ยนมอเตอร์ที่มี Toruqe สูงขึ้นเพื่อที่จะหมุนได้ตามที่เราต้องการ เมื่อได้ออกแบบสำหรับควบคุมมอเตอร์ให้มอเตอร์ไม่ได้รับความเสีย และทำงานได้อย่างมีประสิทธิภาพ และสุดท้ายเมื่อโครงงานเสร็จสมบูรณ์จึงทำการเขียนโปรแกรมVisual basic studio เป็นส่วนกลางที่ใช้ติดต่อสื่อสารระหว่างโปรแกรมและผู้ใช้งานโดยส่งผ่านข้อมูลไปให้ควบคุมการทดลองมอเตอร์ผ่านบอร์ด Arduino

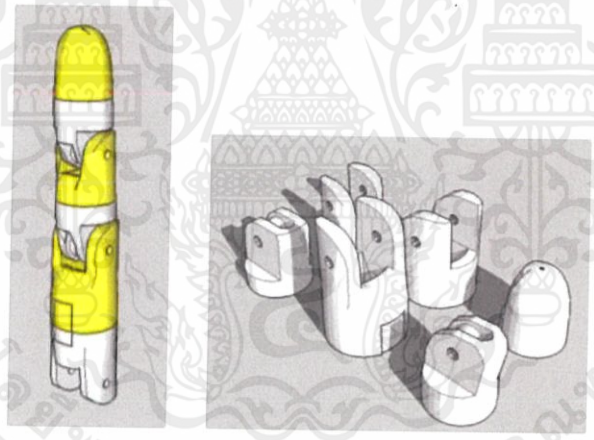
3.2.1 การออกแบบโครงสร้าง

การออกแบบบริเวณมือจะแบ่งออกเป็น 3 ส่วนหลักคือ ส่วนที่เป็นนิ้วมือ(สีขาวและเหลือง) ส่วนที่เป็นบริเวณฝ่ามือ (สีแดง) และส่วนที่เป็นข้อต่อฝ่ามือบริเวณนิ้วนางและนิ้วก้อย(สีเทา)



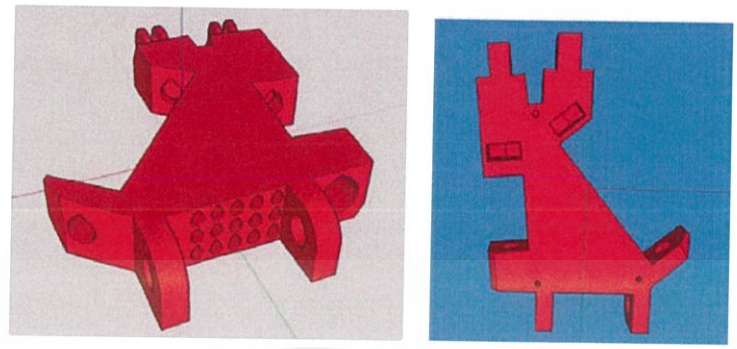
รูปที่ 3.1 แบบจำลองรูปมือ

3.2.1.1 ส่วนที่เป็นนิ้วมือ(สีเหลืองและสีขาว) ทุกนิ้วจะมี6ชิ้นส่วนและข้อต่อนิ้ว 3 ข้อต่อในแต่ละนิ้ว



รูปที่ 3.2 แบบจำลองข้อต่อนิ้วมือ

3.2.1.2 ส่วนที่เป็นฝ่ามือ(สีแดง) ทำหน้าที่เป็นตัวยึดระหว่างข้อต่อนิ้วหัวแม่มือ นิ้วชี้ และนิ้วกลาง



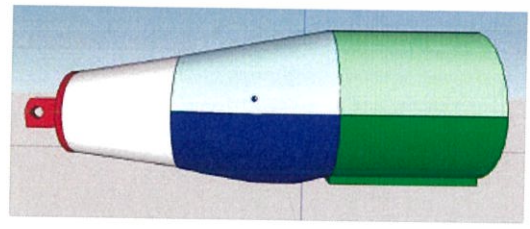
รูปที่ 3.3 แบบจำลองฝ่ามือ

3.2.1.3 ส่วนที่เป็นข้อต่อบริเวณฝ่ามือ (สีเทา) ทำหน้าที่เป็นข้อต่อเพื่อใช้ยึดระหว่างนิ้วนางและนิ้วก้อยและยังช่วยเวลาหยิบจับสิ่งของที่มีขนาดเล็กเพราะตัวข้อต่อก็สามารถสามารถหมุนพับลงมาได้ด้วย

รูปที่ 3.4 แบบจำลองข้อต่อบริเวณนิ้วนางและนิ้วก้อย

3.2.2 ส่วนที่เป็นปริเวณปลายแขน

ปลายแขนจะทำหน้าที่เป็นตัวบรรจุแบตเตอรี่ เซอโวโมเตอร์ ไมโครคอนโทรลเลอร์ และกลไกอื่น ๆ เพื่อให้มีการเคลื่อนไหวของนิ้วมือ แขนถูกแบ่งออกเป็น 4 ส่วนเพื่อบรรจุอุปกรณ์ที่แตกต่างกัน ซึ่งแสดงในรูปที่ 1.9 สีแดง-เป็นส่วนที่ยึดกับบริเวณมือเส้นเอ็นจะถูกร้อยลงมาผ่านบริเวณนี้ สีขาว-บรรจุเซอร์โวมอเตอร์เพื่อทำให้บริเวณข้อมือสามารถขยับได้ สีน้ำเงิน-ถูกบรรจุด้วยเซอร์โวมอเตอร์จำนวน 5 ตัวเพื่อเพื่อใช้ดึงเอ็นบริเวณนิ้วมือทั้ง 5ให้ขยับ และสีเขียว-เป็นที่บรรจุอาดูโน่หรือตัวไมโครคอนโทรลเลอร์และแบตเตอรี่ต่างๆ



รูปที่ 3.5 แบบจำลองท่อนแขน

3.3 ขั้นตอนการประกอบแขนกล

ขั้นตอนที่ 1

หลังจากพิมพ์สามมิติ ต้องทำการแกะซัพพอร์ตของเส้นใย 3D Print ให้เรียบร้อย

รูปที่ 3.6 แบบพิมพ์ที่ยังติดSupportออกมา

ขั้นตอนที่ 2

เลือกทำส่วนที่เป็นบริเวณของฝ่ามือโดยทำการจัดสรีระของนิ้วมือตามที่ได้ออกแบบไว้ จากนั้นทำการตะไบข้อต่อต่างๆให้พอดีกันกับแต่ละล็อค

รูปที่ 3.7 นำแบบพิมพ์สามมิติตะไบเข้าด้วยกัน

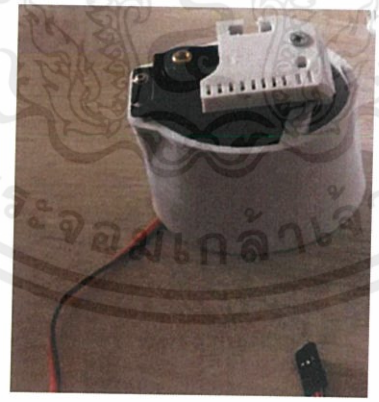
ขั้นตอนที่ 3

ทำการร้อยเอ็น 2 เส้นโดยเส้นแรกจะร้อยขึ้นไปตามบริเวณด้านหน้าของฝ่ามือส่วนเส้นที่ 2 ร้อยจากบริเวณด้านหลังของฝ่ามือขึ้นไปตามข้อต่อนิ้วมือทุกข้อจนถึงข้อต่อบนสุดของนิ้วมือแล้วทำการมัดปมให้แน่นหนาและทำการยึดน็อตตามข้อต่อต่างๆเพื่อทำให้สามารถหมุนและขยับได้แต่ในที่นี้เราเลือกใช้เส้นฟิลาเม้นขนาด 3 mmเพื่อสะดวกต่อการใช้งานและง่ายต่อการแก้ไข

รูปที่3.8 นำเอ็นมาร้อยลงแบบพิมพ์สามมิติ

ขั้นตอนที่ 4

ขั้นตอนนี้จะทำการประกอบบริเวณข้อมือโดยทำการยึดเซอโวมอเตอร์เบอร์ MG 996r ลงไป



รูปที่ 3.9 ส่วนที่เป็นข้อมือของแขน

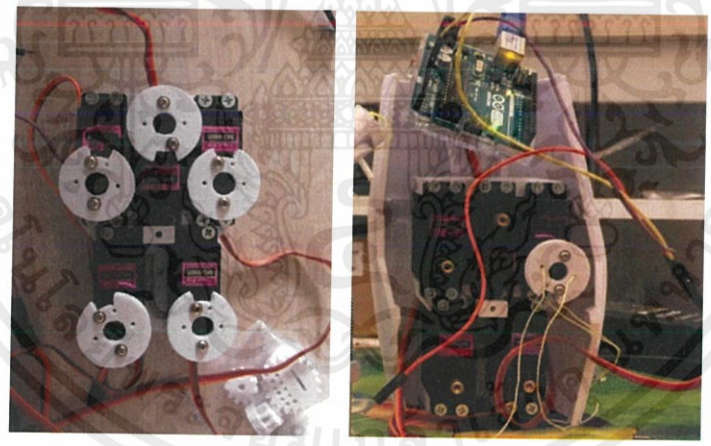
ขั้นตอนที่ 5

เป็นขั้นตอนของบริเวณประกอบตรงส่วนแขน โดยทำการใช้กาวในการยึดส่วนประกอบทั้ง 2 ส่วนเข้าไว้ด้วยกัน

รูปที่ 3.10 ส่วนท่อนแขน

ขั้นตอนที่ 6

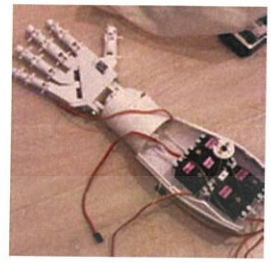
ทำการบรรจุเซอโวมอเตอร์มอเตอร์ 5 ตัว(MG 996r)ลงบนฐานยึดเซอร์โวมอเตอร์ซึ่งได้มาจากการพิมพ์ 3 มิติ ออกมาตามรูปที่ 3.11และหลักจากนั้นทำการยึดฐานเซอร์โวเข้ากับบริเวณท่อนแขนแล้วนำเอ็นที่ร้อยจากนิ้วมือทั้ง 5 นิ้วยึดกับเซอร์โวแต่ละตัวแบบพิมพ์ 3 มิติ ตามรูปที่ 3.11



รูปที่ 3.11 บรรจุ Servo Bedลงในท่อนแขน

ขั้นตอนที่ 7

ประกอบส่วนที่เป็นฝ่ามือเข้ากับส่วนที่เป็นท่อนแขน



รูปที่ 3.12 ท่อนแขนที่เชื่อมต่อกับฝ่ามือ

ขั้นตอนที่ 8

ทำการปิดตัวแขนกลทั้งสองด้านเพื่อนำใช้งาน

รูปที่ 3.13 แขนหลังจากประกอบเสร็จทั้งหมด

3.4 การควบคุมโดยใช้บอร์ดไมโครคอนโทรลเลอร์

3.4.1 ขั้นตอนการทำทำบอร์ด Arduino แบบ stand alone

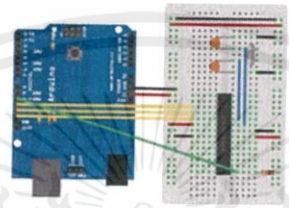
เป็นขั้นตอนในการเปลี่ยนไมโครคอนโทรลเลอร์ทั่วๆไป ให้สามารถใช้งานกับ Code ของ Arduino ได้ โดย Arduino UNO R3 นั้น ใช้ Microcontroller รุ่น ATmega328P ซึ่งเป็นไมโครคอนโทรลเลอร์ของ Atmel แต่อยู่ๆถ้านำมาใช้เลย ก็จะไม่สามารถใช้กับ Arduino ได้ จึงต้องมีการ Burn Bootloader ก่อนนั่นเอง โดย Microcontroller ที่ติดมากับ UNO R3 ตั้งแต่แรก จะมีการ burn ไว้อยู่แล้ว แต่ไมโครคอนโทรลเลอร์ที่แยกซื้อมาต่างหาก อาจจะยังไม่ได้ burn และถึงแม้ว่าจะ burn ไว้แล้ว ก็สามารถทำซ้ำอีกรอบได้เช่นกัน

ขั้นตอนการ Burn Bootloader

1. ต่อวงจร

2. Upload code และ Burn Boothloader ได้เลย

3.นำATmega328 P-PU หลังจากBurn Boothloader ไปใส่ในแผ่นทองแดงที่ได้ออกแบบไว้



รูปที่3.14 เชื่อมอาดูโน่และลองเชื่อมบนโฟโต้บอร์ด

3.4.2 อุปกรณ์ที่ใช้ในบอร์ดArduino(stand alone)

1. Crystal กำเนิดความถี่ 16.00 MHz 1 ตัว

2.ตัวเก็บประจุ 22 pF 2 ตัว

3. IC 7805 Voltage Regulator 1 ตัว

4.ตัวเก็บประจุ 0.1 uF 1 ตัว

5.ตัวเก็บประจุ 0.33 uF 1 ตัว

6.Switch 1 ตัว

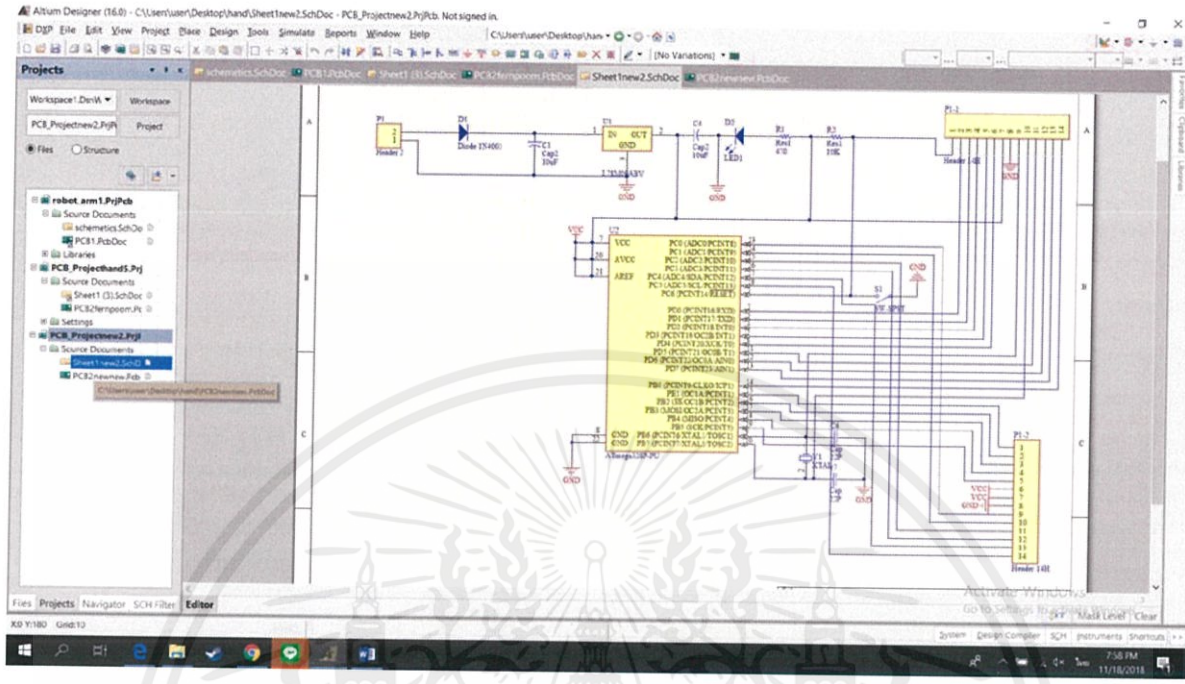
7.Resistor 470 โอห์ม 1 ตัว

8.Resistor 10K โอห์ม 1 ตัว

9.ATmega 328 P-Pu 1 ตัว

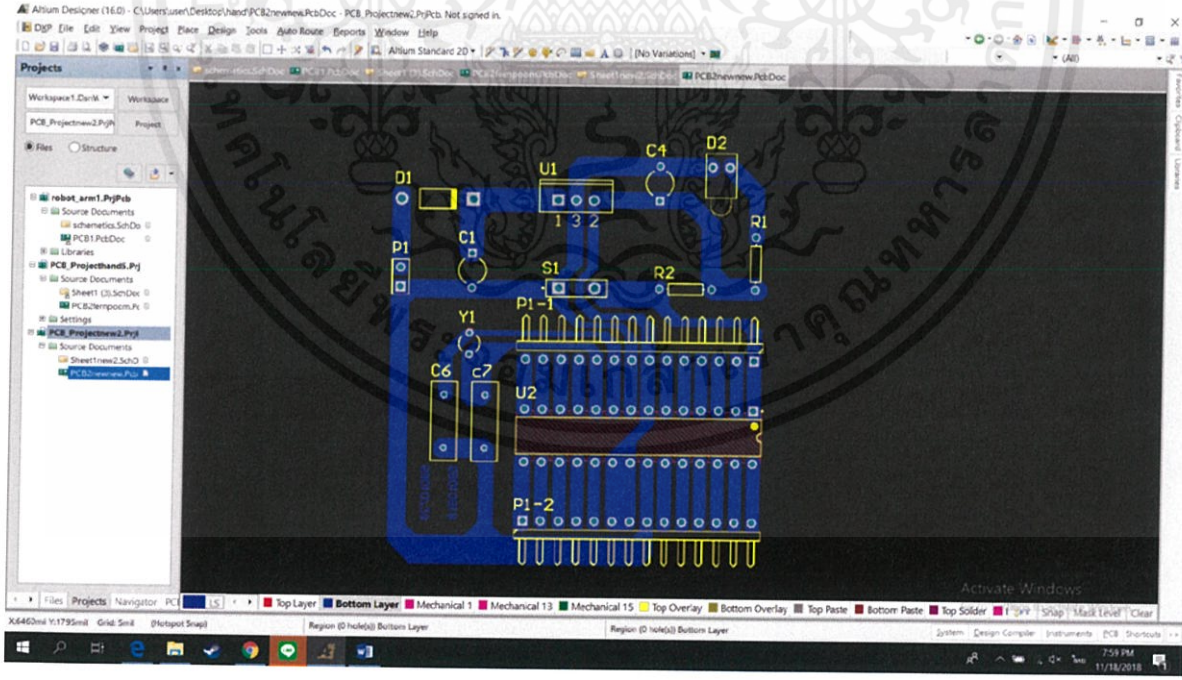
10.pin header

3.4.2 ขั้นตอนการออกแบบ Schmetics ในโปรแกรม Altium



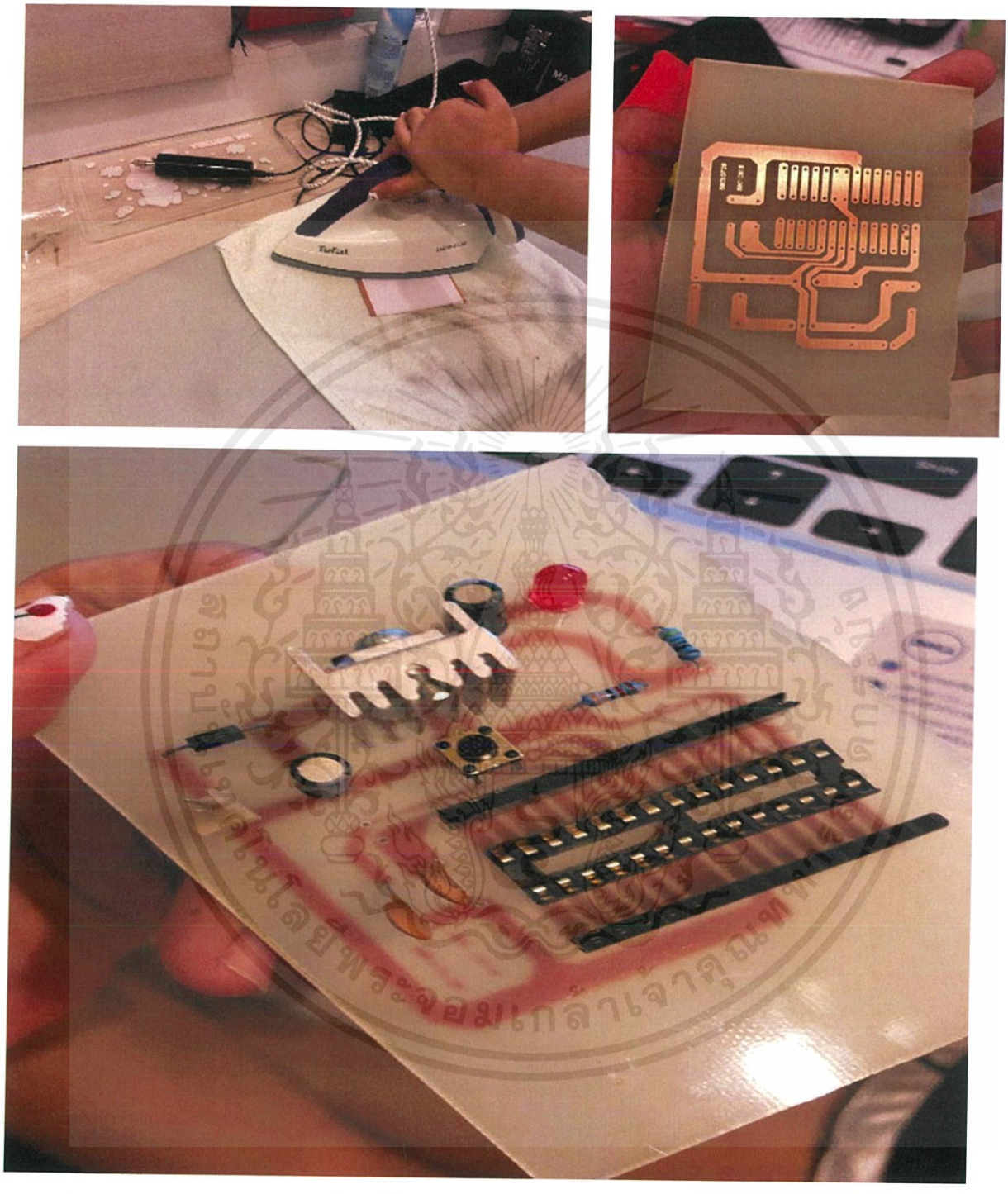
รูปที่3.15 ออกแบบ Schmetics

3.4.3 ขั้นตอนการออกแบบลายวงจรบนแผ่นทองแดง



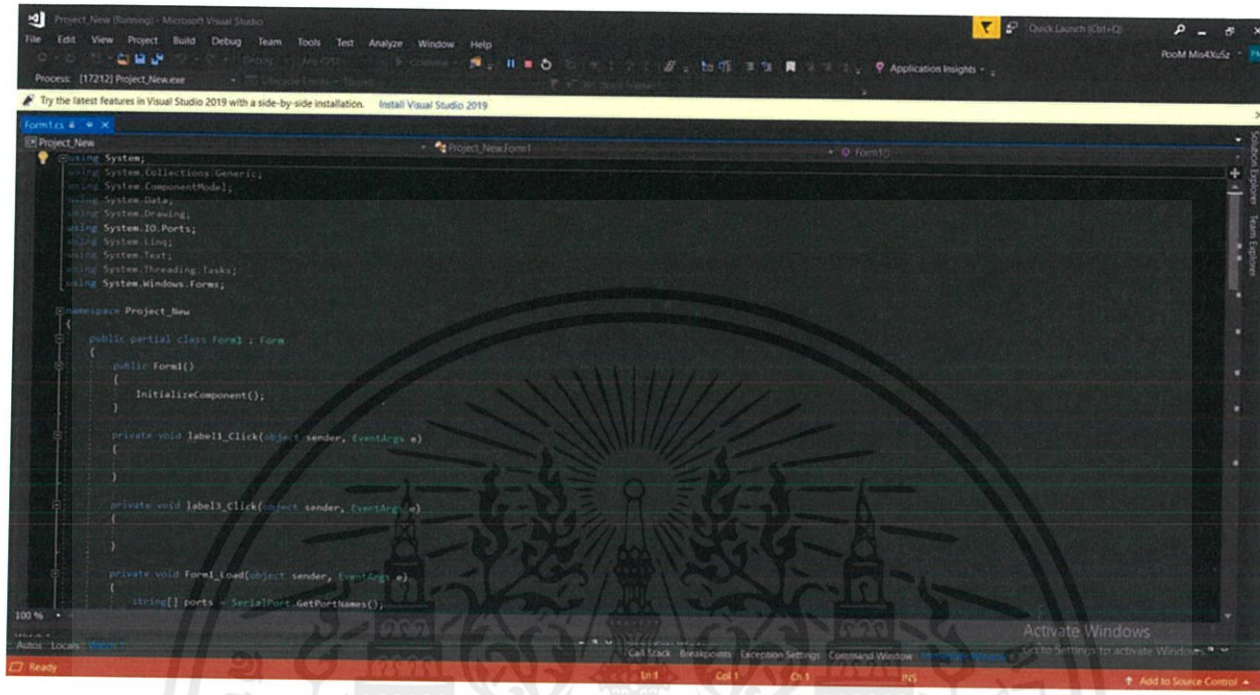
รูปที่3.16 ออกแบบ PCB

3.4.4 ขั้นตอนการทำบอร์ด Arduino

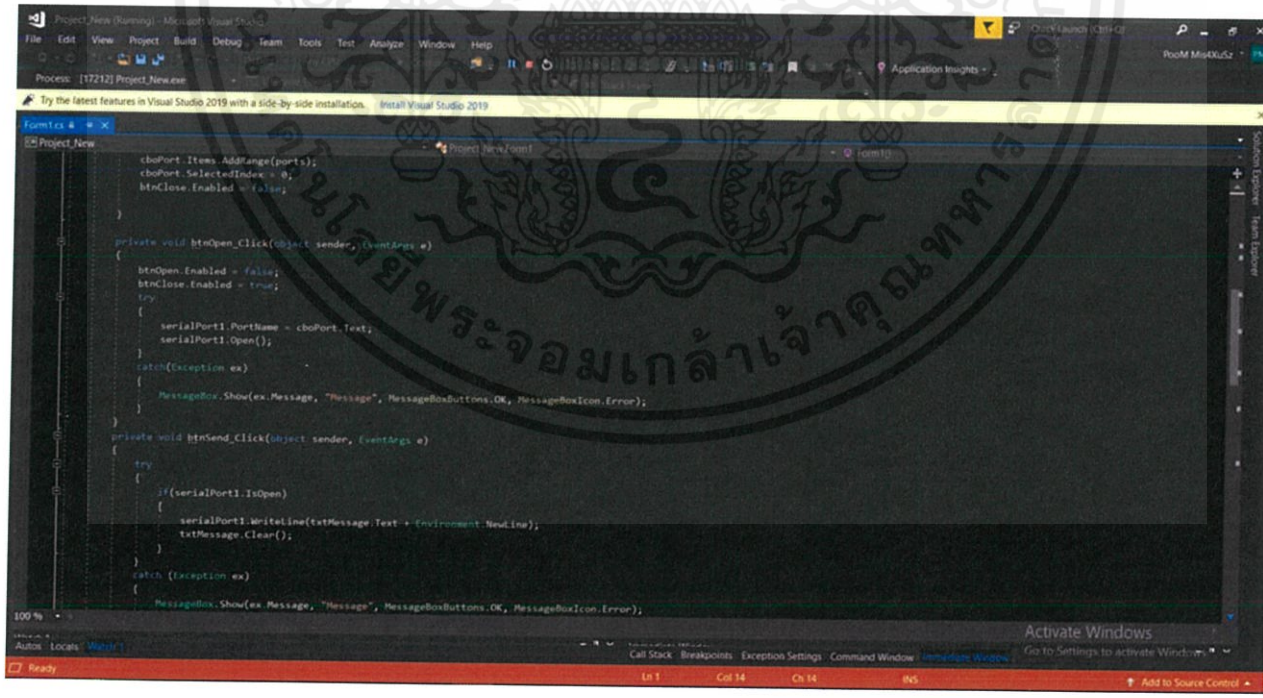


รูปที่3.17 ขั้นตอนการทำบอร์ด Arduino Stand alone

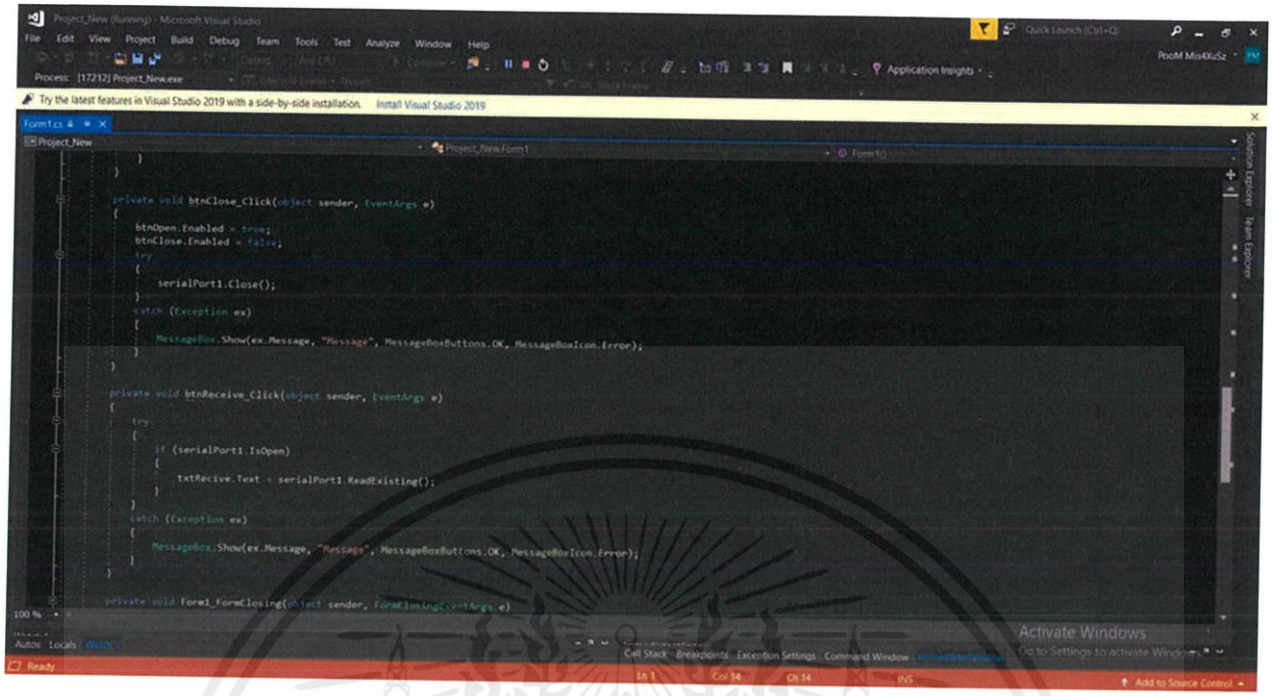
3.5 หน้าต่างโค้ดโปรแกรมและตัว UI ที่ใช้ติดต่อกับไมโครคอนโทรเลอร์เพื่อสั่งการตัวแขนกล



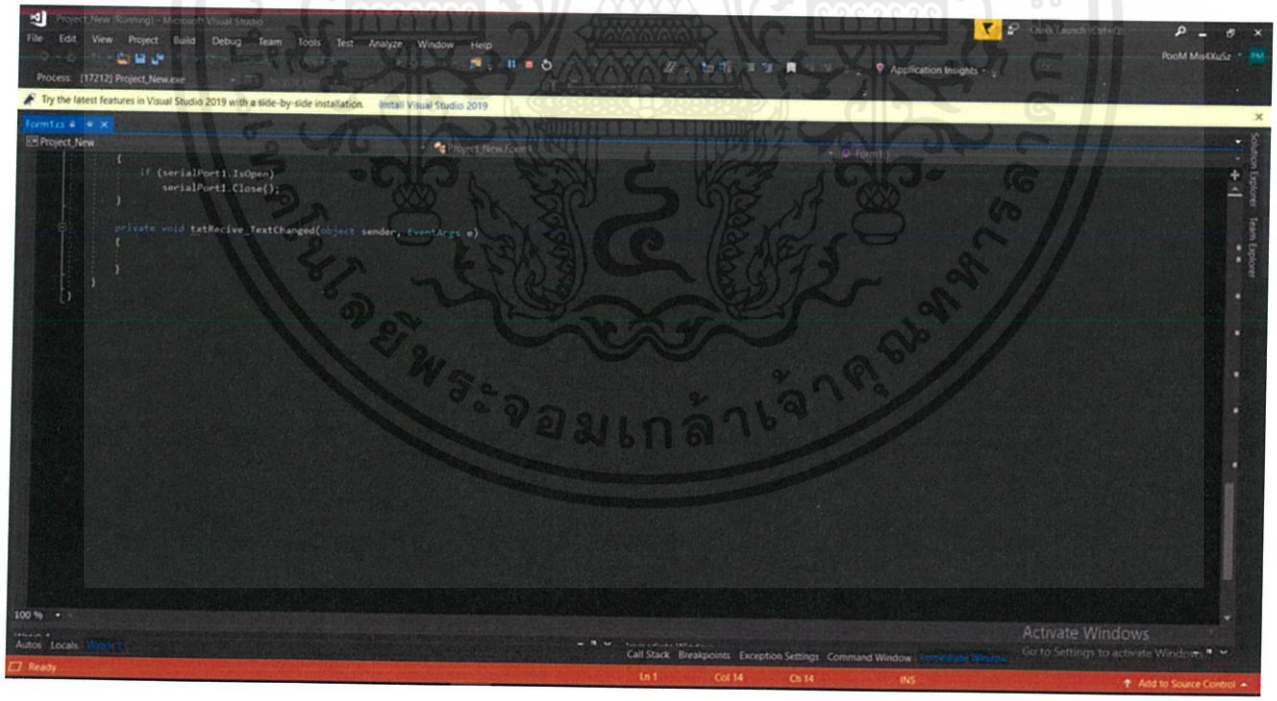
รูปที่3.18 หน้าต่างโค้ดโปรแกรมใน Visual Basic



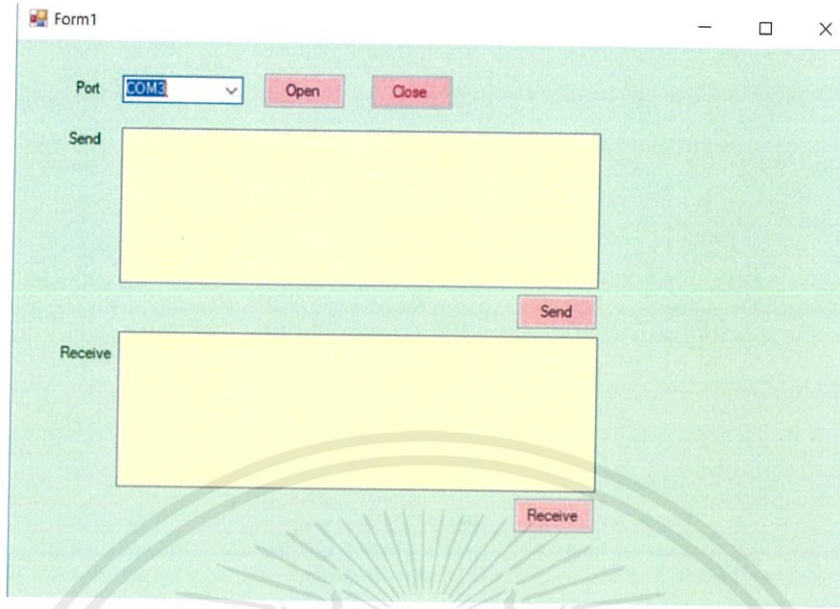
รูปที่3.19 หน้าต่างโค้ดโปรแกรมใน Visual Basic



รูปที่3.20 หน้าต่างโค้ดโปรแกรมใน Visual Basic

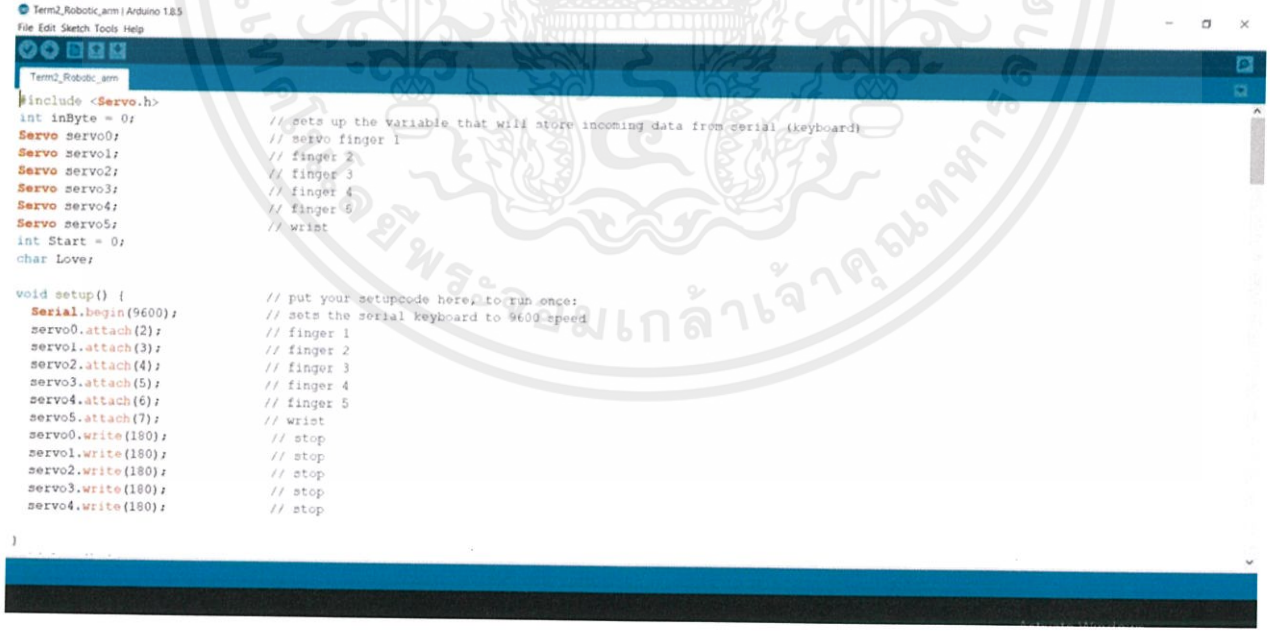


รูปที่3.21 หน้าต่างโค้ดโปรแกรมใน Visual Basic



รูปที่3.22 หน้าต่าง UI ที่ได้ทำขึ้นเพื่อติดต่อกับตัวแขนกล

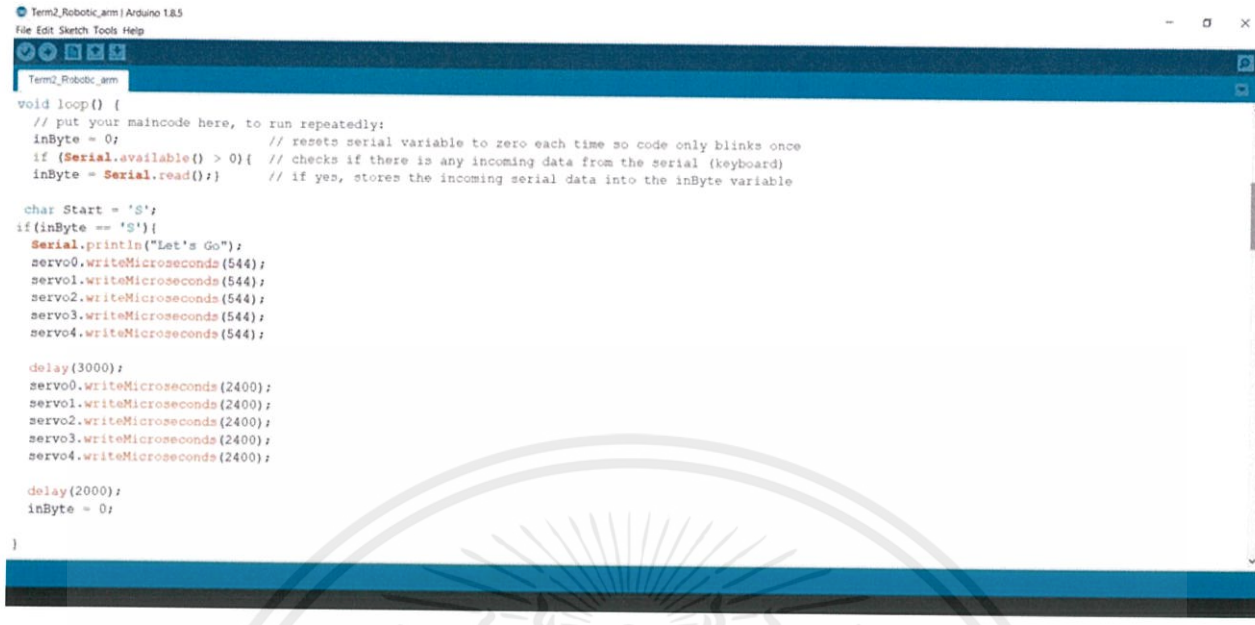
3.6 โค้ดที่ใช้ทำการควบคุมการเคลื่อนไหวของนิ้วมือ



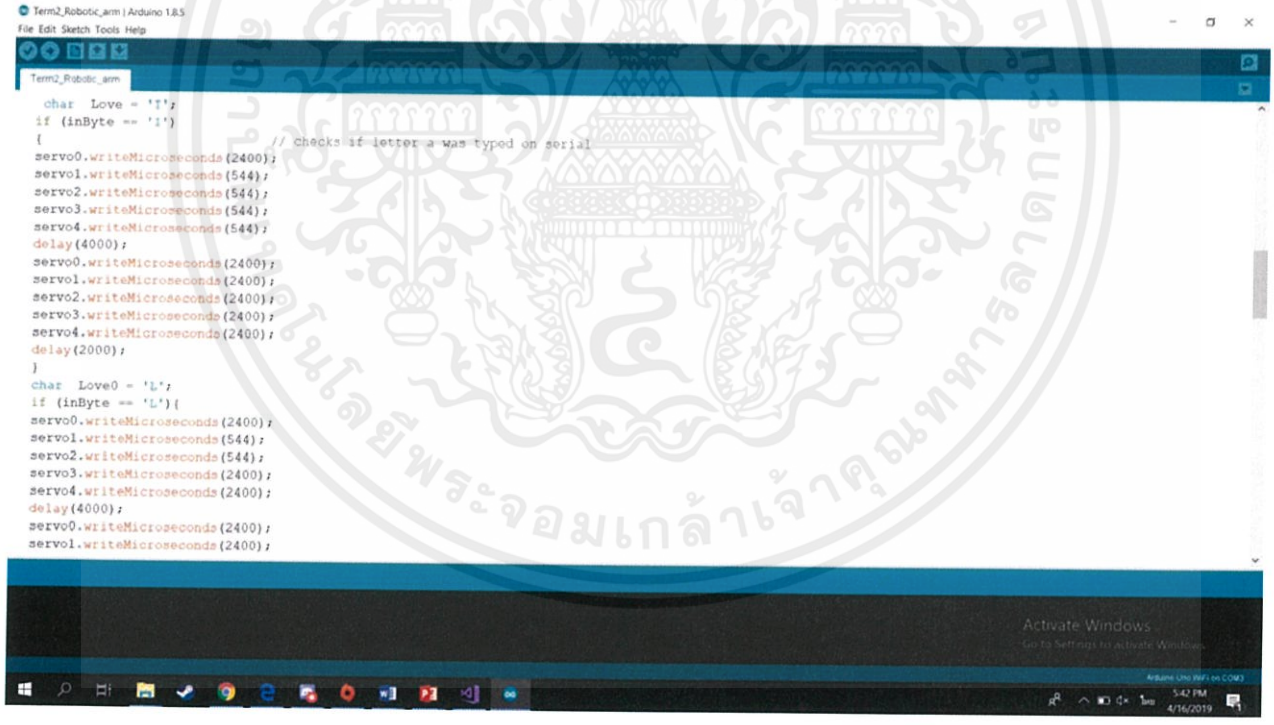
```
Term2_Robotic_arm | Arduino 1.8.5
File Edit Sketch Tools Help
Term2_Robotic_arm
#include <Servo.h>
int inByte = 0;                 // sets up the variable that will store incoming data from serial (keyboard)
Servo servo0;                   // servo finger 1
Servo servo1;                   // finger 2
Servo servo2;                   // finger 3
Servo servo3;                   // finger 4
Servo servo4;                   // finger 5
Servo servo5;                   // wrist
int Start = 0;
char Love;

void setup() {                  // put your setupcode here, to run once:
  Serial.begin(9600);           // sets the serial keyboard to 9600 speed
  servo0.attach(2);             // finger 1
  servo1.attach(3);             // finger 2
  servo2.attach(4);             // finger 3
  servo3.attach(5);             // finger 4
  servo4.attach(6);             // finger 5
  servo5.attach(7);             // wrist
  servo0.write(180);             // stop
  servo1.write(180);             // stop
  servo2.write(180);             // stop
  servo3.write(180);             // stop
  servo4.write(180);             // stop
}
```

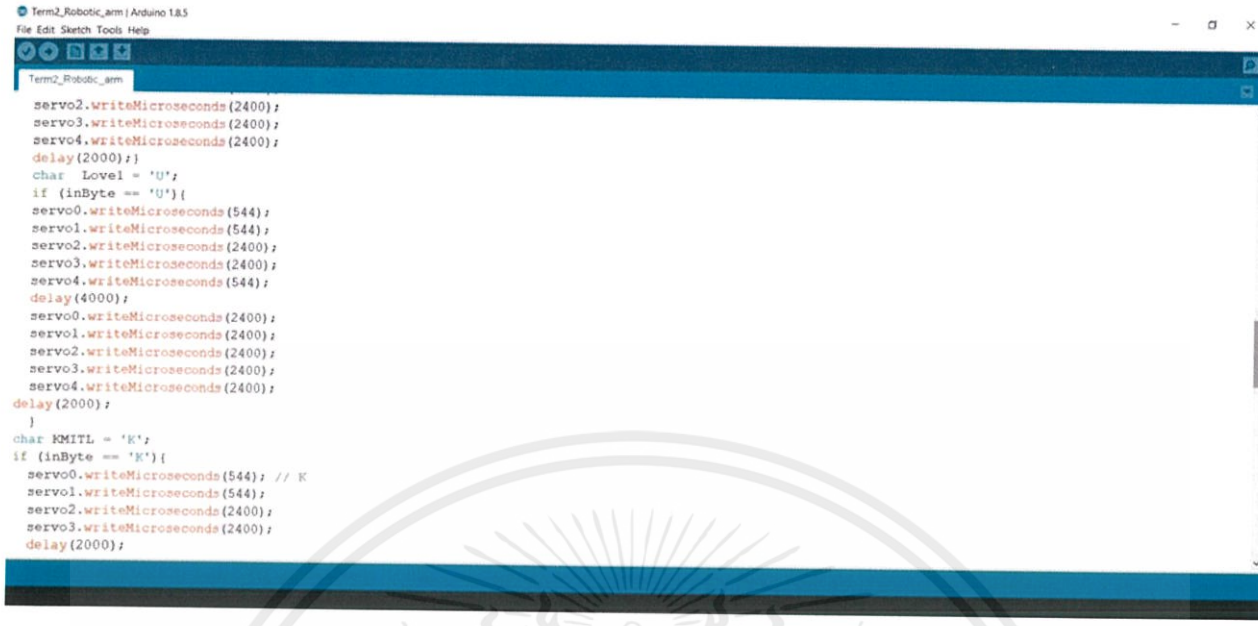
รูปที่3.23 โค้ดควบคุมการทำงาน Servo motor



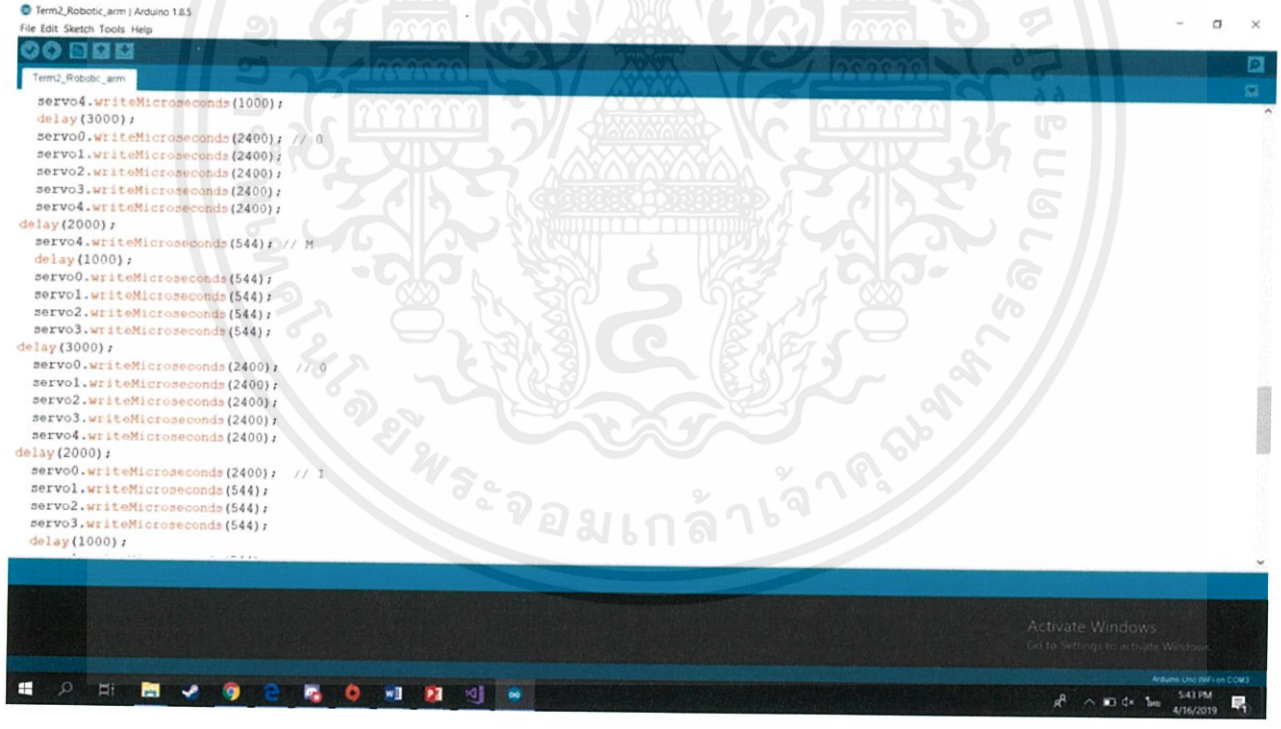
รูปที่ 3.24 โค้ดควบคุมการทำงาน Servo motor



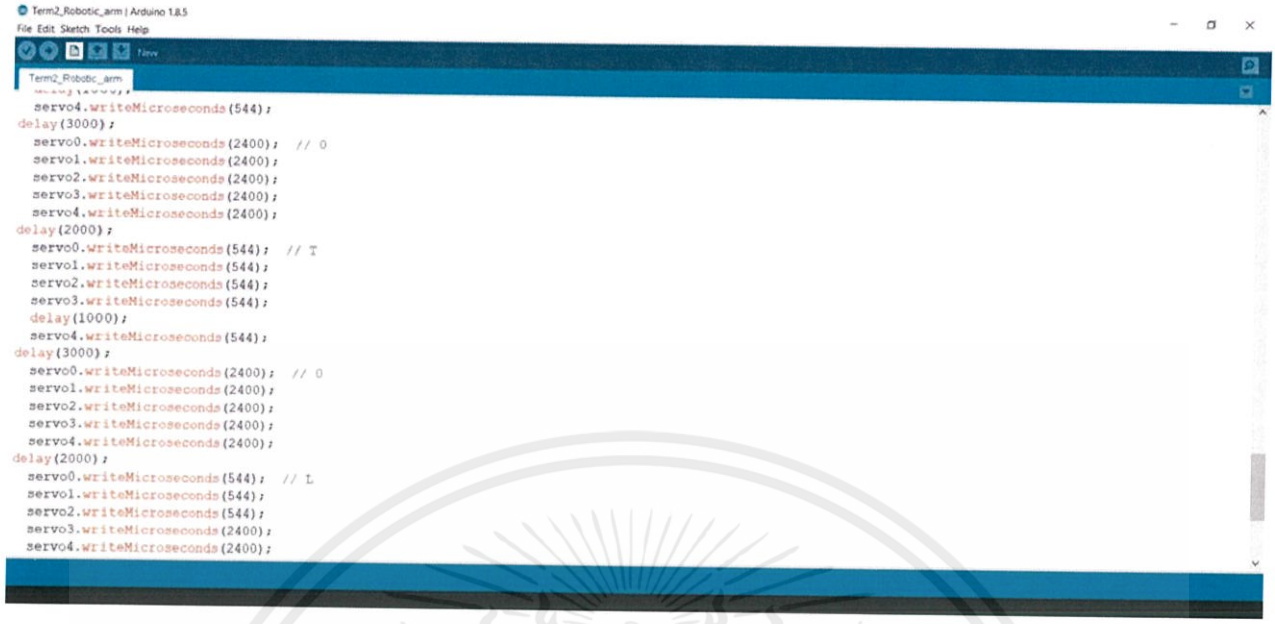
รูปที่ 3.25 โค้ดควบคุมการทำงาน Servo motor



รูปที่ 3.26 โค้ดควบคุมการทำงาน Servo motor



รูปที่ 3.27 โค้ดควบคุมการทำงาน Servo motor



รูปที่ 3.28 โค้ดควบคุมการทำงาน Servo motor

รูปที่ 3.29 โค้ดควบคุมการทำงาน Servo motor

บทที่ 4

การทดลองและผลการทดลอง

4.1 การทดลองโปรแกรมควบคุมเซอโวมอเตอร์

4.1.1 การทำงานของเซอร์โวมอเตอร์ การควบคุมการทำงานของเซอโวมอเตอร์ทำได้ดดยการป้อนสัญณาญพัลส์ให้กับมอเตอร์ซึ่งตำแหน่งและทิศทางการหมุนของมอเตอร์นี้ จะขึ้นอยู่กับความกว้างพัลส์

4.1.2 การทดลอง

1.สร้างสัญญาณ PWM ผ่านไมโครคอนโทรลเลอร์ ATmega 2560

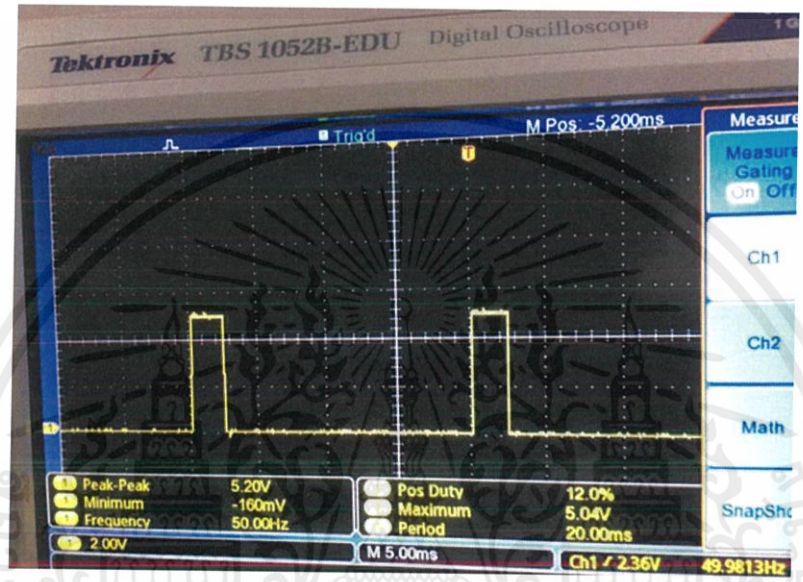
2.สังเกตผลจาก Oscilloscope ของเซอร์โวมอเตอร์แต่ละตัว

4.1.3 ผลการทดลอง

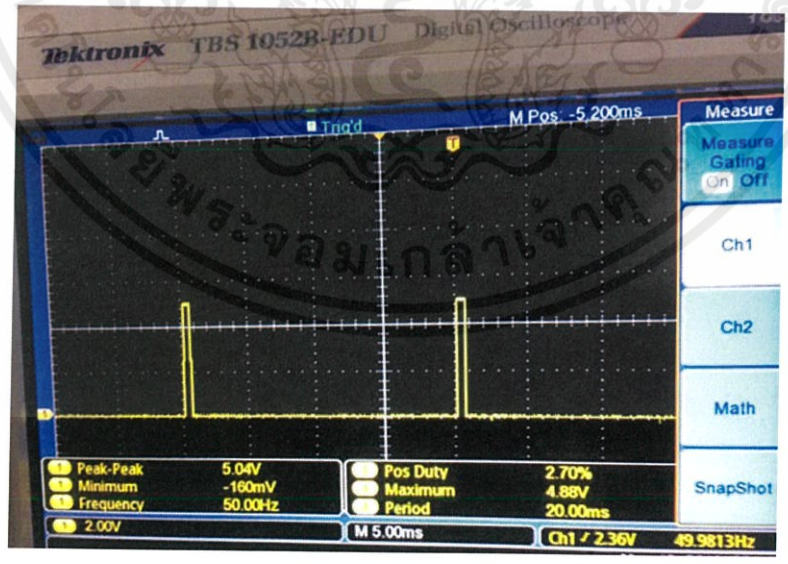
เมื่อเขียนโปรแกรมสร้างสัญญาณ PWM แล้วได้ผลดังนี้

รูปที่4.1 Servo motor MG996

Servo 1 MG996r (บริเวณหัวแม่มือ) ต้องคำนึงถึงค่า Toque เพราะเส้นเอ็นทุกนิ้วต้องใช้แรงอย่างมากที่จะทำให้เส้นเอ็นขยับ ในการควบคุมการทำงานในส่วนนี้จะใช้คำสั่งสัญญาณความกว้างพัลส์ซึ่งหาได้จากสมการ (Div*Time/Div) จะได้พัลส์ขนาด 2.5 ms ให้เซอร์โวหมุนไปที่ 180 องศา เพื่อที่จะให้กำมือและความกว้างพัลส์ขนาด 0.95 ms ให้เซอร์โวหมุนไปที่ 0 องศา เพื่อที่จะดึงนิ้วมือกลับขึ้นสู่สภาพเดิม

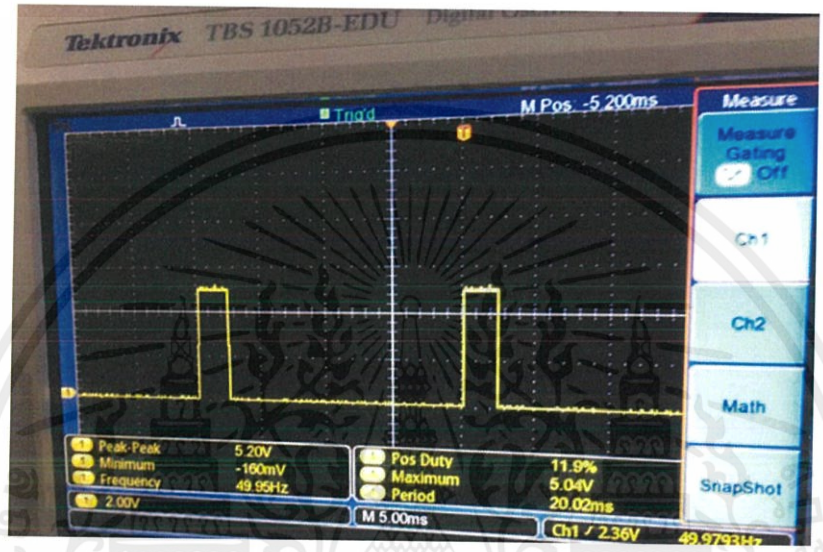


รูปที่ 4.2 สัญญาณพัลส์ที่เซอร์โว 180 องศา

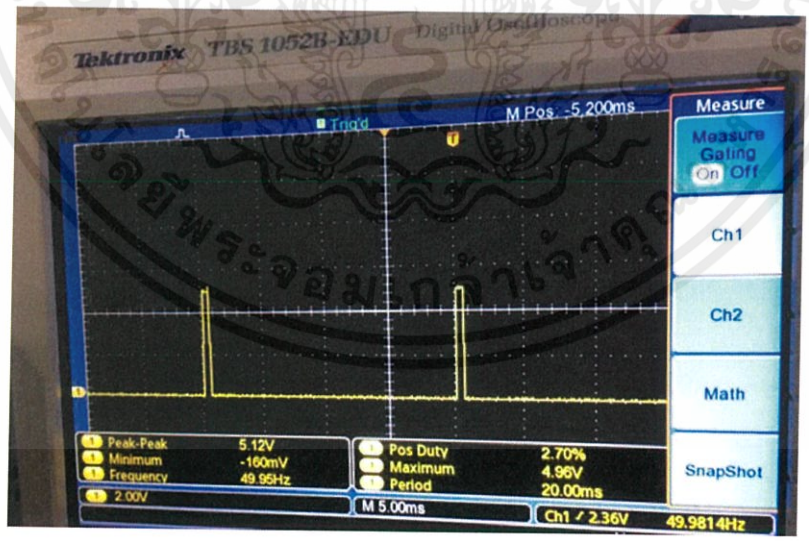


รูปที่ 4.3 สัญญาณพัลส์ที่เซอร์โว 0 องศา

Servo 2 MG996r (บริเวณนิ้วชี้) ต้องคำนึงถึงค่า Toque เพราะเส้นเอ็นทุกนิ้วต้องใช้แรงอย่างมากที่จะทำให้เส้นเอ็นขยับ ในการควบคุมการทำงานในส่วนนี้จะใช้คำสั่งสัญญาณความกว้างพัลส์ซึ่งหาได้จากสมการ (Div*Time/Div) จะได้พัลส์ขนาด 2.5 ms ให้เซอร์โวหมุนไปที่ 180 องศา เพื่อที่จะให้กำมือและความกว้างพัลส์ขนาด 0.80 ms ให้เซอร์โวหมุนไปที่ 0 องศา เพื่อที่จะดึงนิ้วมือกลับขึ้นสู่สภาพเดิม

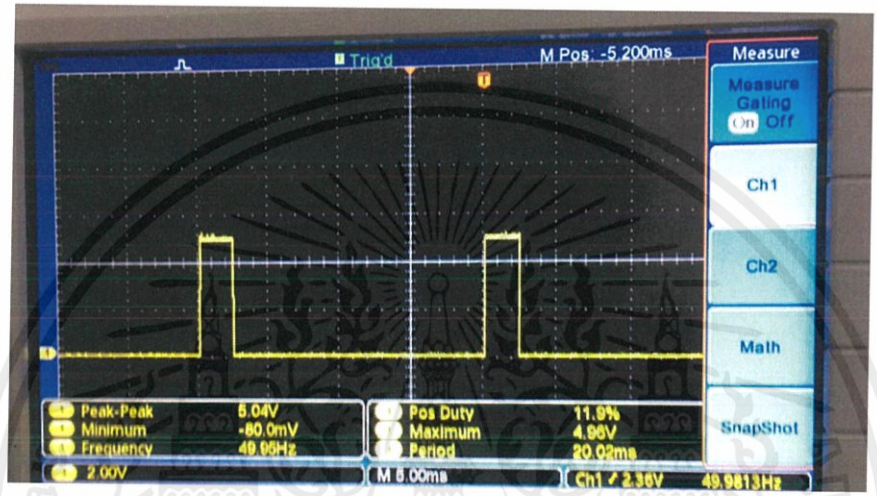


รูปที่ 4.4 สัญญาณพัลส์ที่เซอร์โว 180 องศา

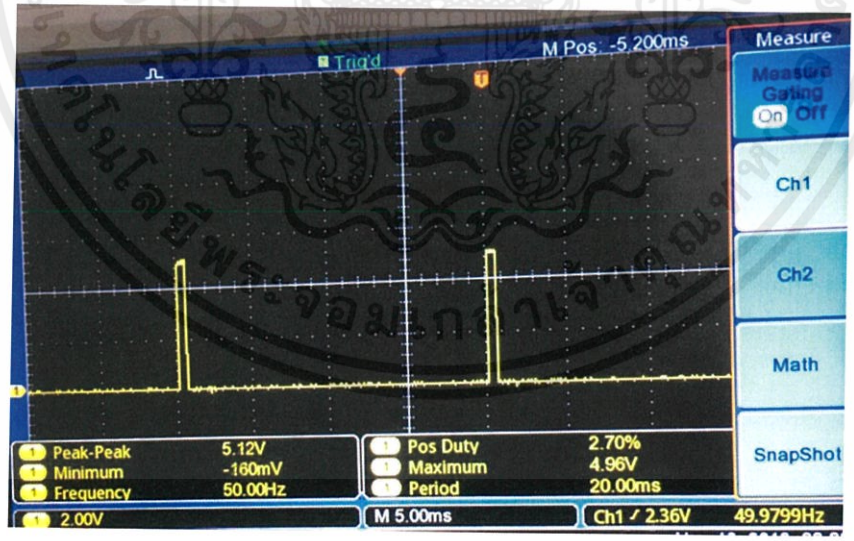


รูปที่ 4.5 สัญญาณพัลส์ที่เซอร์โว 0 องศา

Servo 3 MG996r (บริเวณนิ้วกลาง) ต้องคำนึงถึงค่า Toque เพราะเส้นเอ็นทุกนิ้วต้องใช้แรงอย่างมากที่จะทำให้เส้นเอ็นขยับ ในการควบคุมการทำงานในส่วนนี้จะใช้คำสั่งสัญญาณความกว้างพัลส์ซึ่งหาได้จากสมการ (Div*Time/Div) จะได้พัลส์ขนาด 2.25 ms ให้เซอร์โวหมุนไปที่ 180 องศา เพื่อที่จะให้กำมือและความกว้างพัลส์ขนาด 0.90 ms ให้เซอร์โวหมุนไปที่ 0 องศา เพื่อที่จะดึงนิ้วมือกลับขึ้นสู่สภาพเดิม

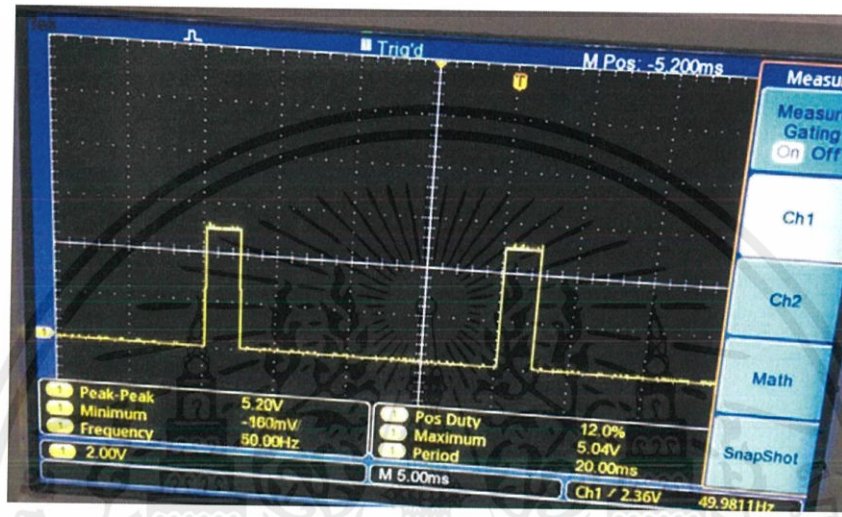


รูปที่ 4.6 สัญญาณพัลส์ที่เซอร์โว 180 องศา

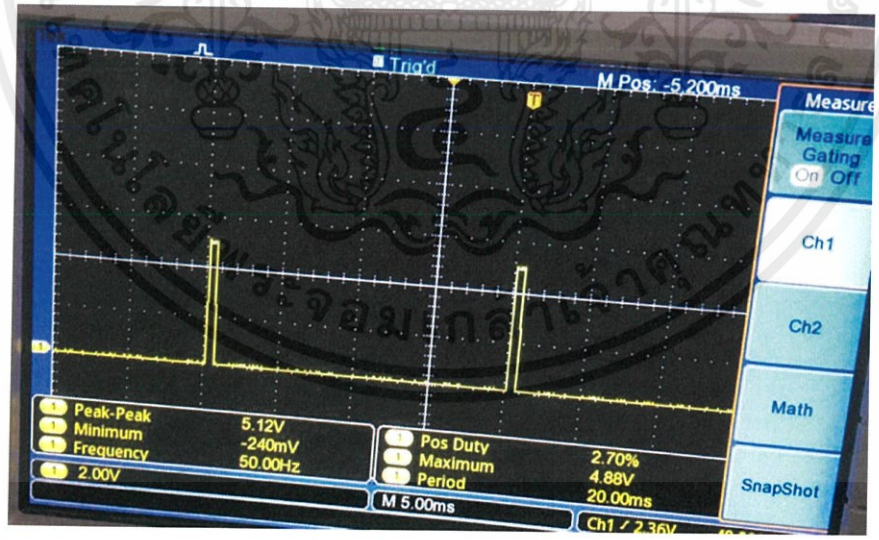


รูปที่ 4.7 สัญญาณพัลส์ที่เซอร์โว 0 องศา

Servo 5 MG996r (บริเวณนิ้วนาง) ต้องคำนึงถึงค่า Toque เพราะเส้นเอ็นทุกนิ้วต้องใช้แรงอย่างมากที่จะทำให้เส้นเอ็นขยับ ในการควบคุมการทำงานในส่วนนี้จะใช้คำสั่งสัญญาณความกว้างพัลส์ซึ่งหาได้จากสมการ (Div*Time/Div) จะได้พัลส์ขนาด 2.5 ms ให้เซอร์โวหมุนไปที่ 180 องศา เพื่อที่จะให้กำมือและความกว้างพัลส์ขนาด 1.00 ms ให้เซอร์โวหมุนไปที่ 0 องศา เพื่อที่จะดึงนิ้วมือกลับขึ้นสู่สภาพเดิม

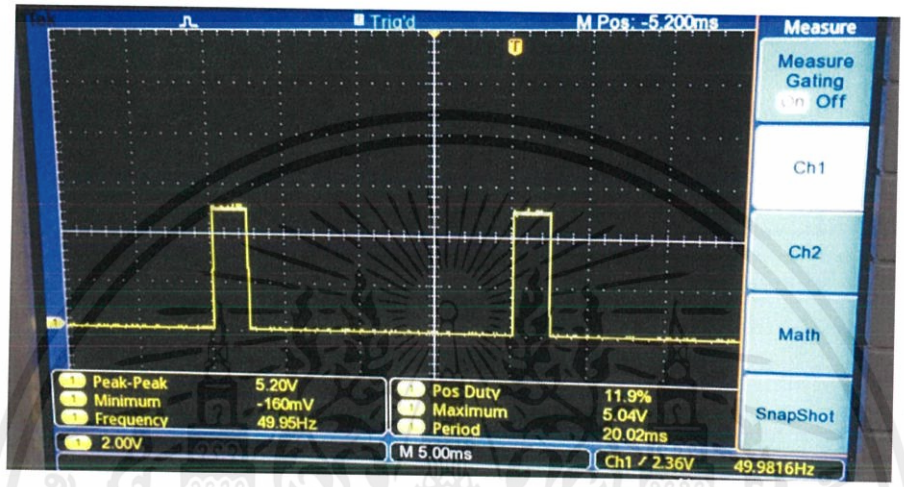


รูปที่ 4.8 สัญญาณพัลส์ที่เซอร์โว 180 องศา

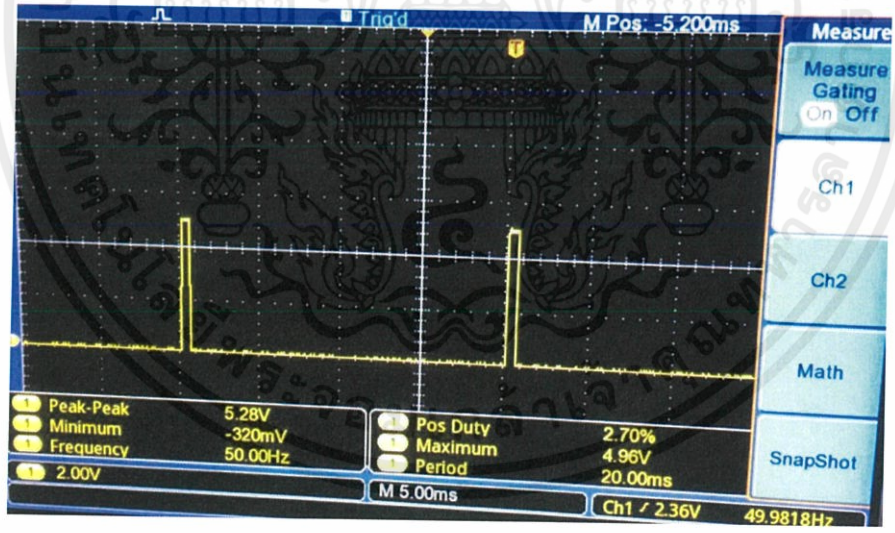


รูปที่ 4.9 สัญญาณพัลส์ที่เซอร์โว 0 องศา

Servo 5 MG996r (บริเวณนิ้วก้อย) ต้องคำนึงถึงค่า Toque เพราะเส้นเอ็นทุกนิ้วต้องใช้แรงอย่างมากที่จะทำให้เส้นเอ็นขยับ ในการควบคุมการทำงานในส่วนนี้จะใช้คำสั่งสัญญาณความกว้างพัลส์ซึ่งหาได้จากสมการ (Div*Time/Div) จะได้พัลส์ขนาด 2.47 ms ให้เซอร์โวหมุนไปที่ 180 องศา เพื่อที่จะให้กำมือและความกว้างพัลส์ขนาด 0.97 ms ให้เซอร์โวหมุนไปที่ 0 องศา เพื่อที่จะดึงนิ้วมือกลับขึ้นสู่สภาพเดิม

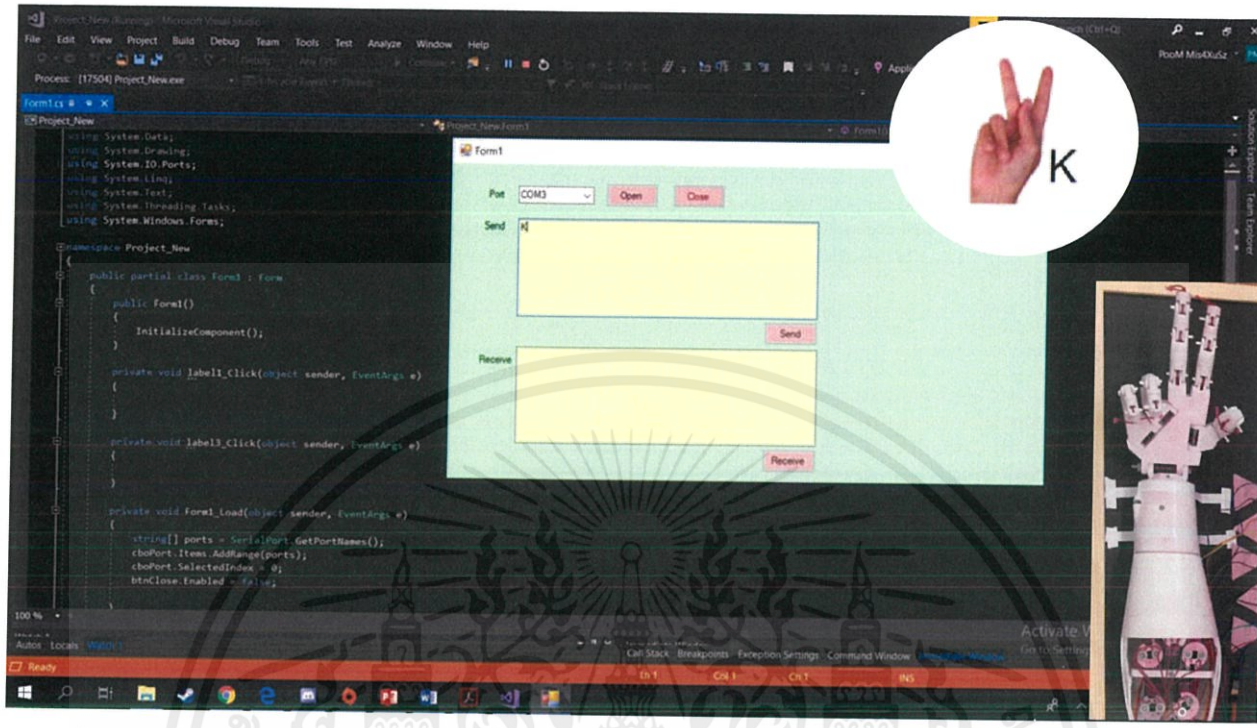


รูปที่ 4.10 สัญญาณพัลส์ที่เซอร์โว 180 องศา

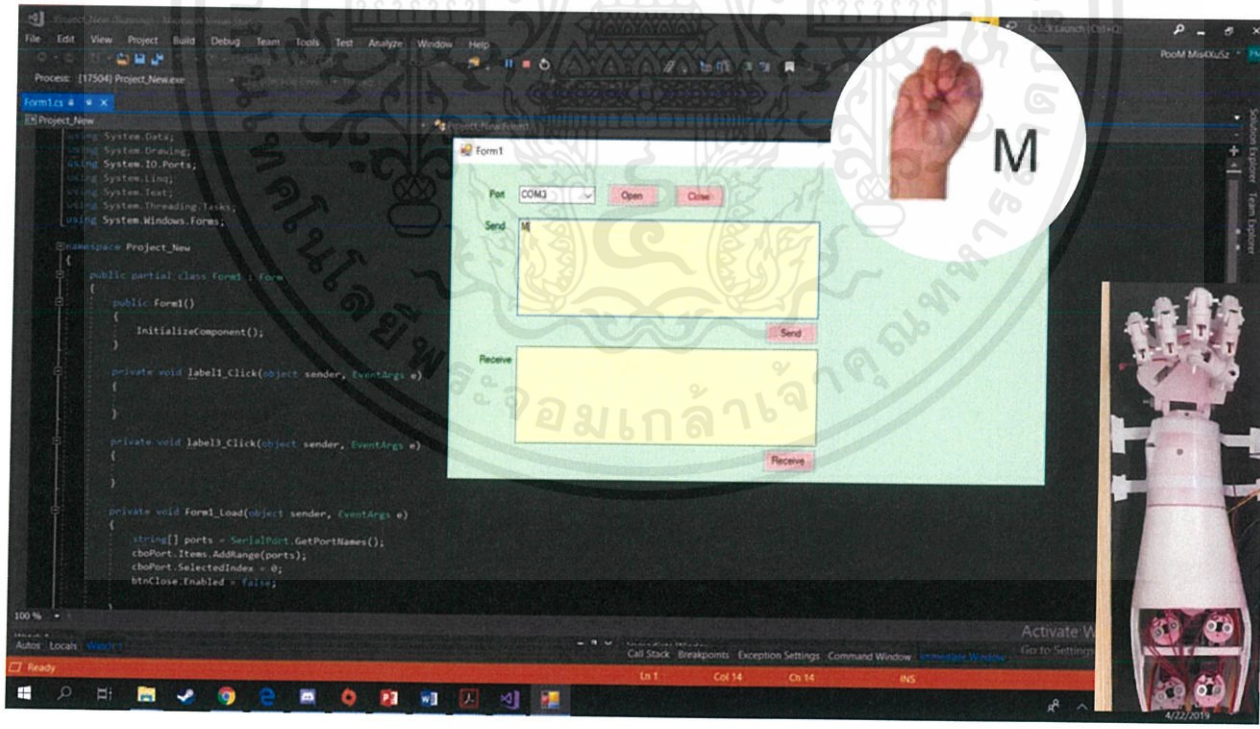


รูปที่ 4.11 สัญญาณพัลส์ที่เซอร์โว 0 องศา

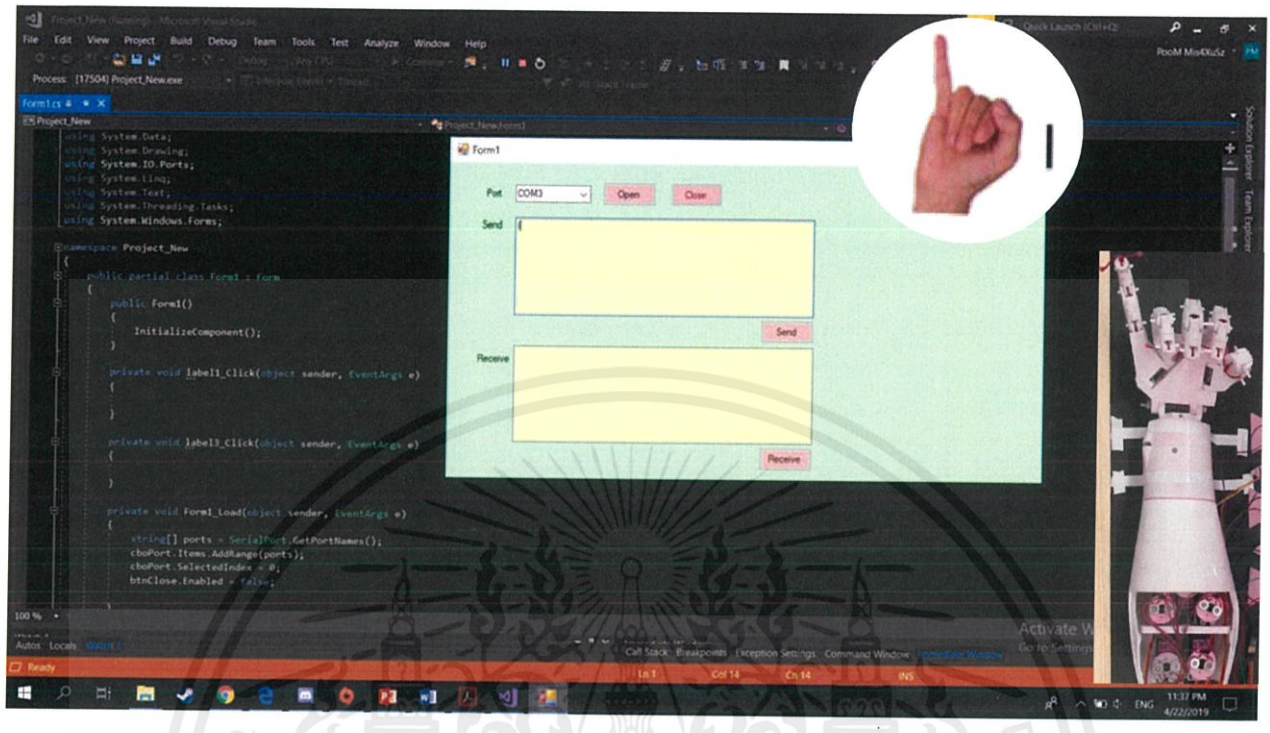
4.2 การสั่งงานแขนกลผ่านทางหน้าต่างใน User Interface



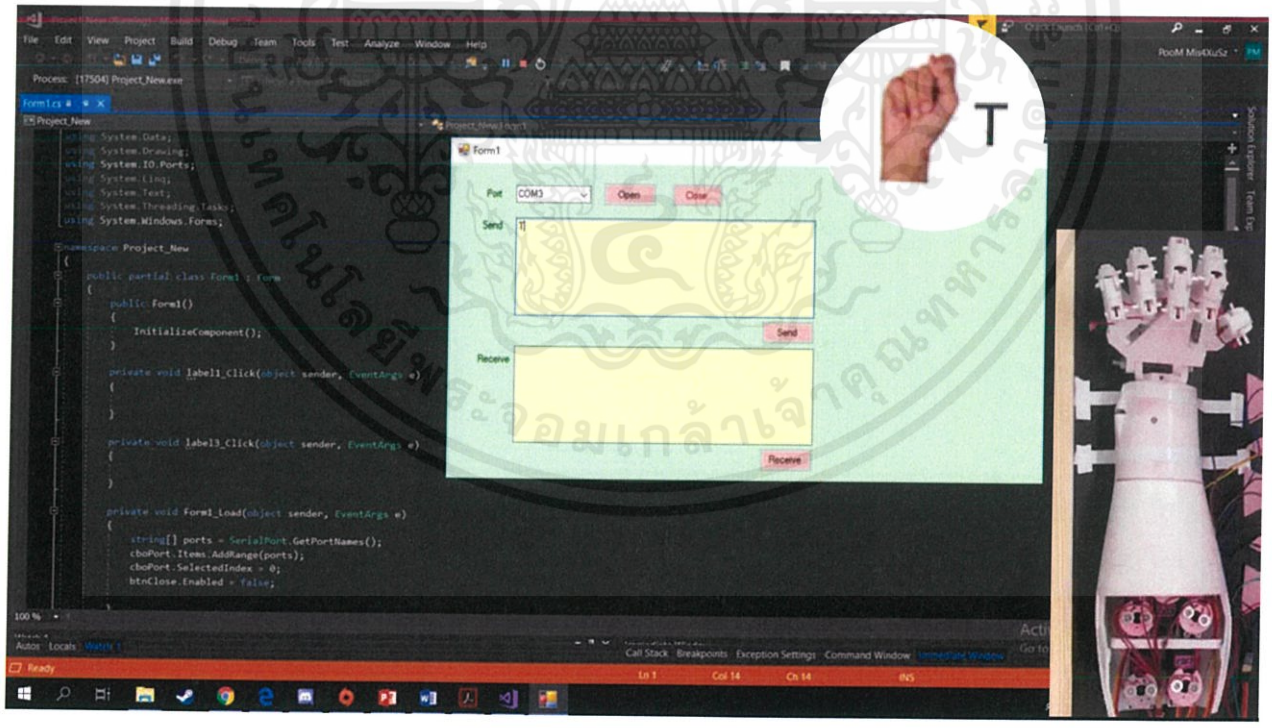
รูปที่4.12 ความสัมพันธ์ระหว่างโปรแกรมกับแขนกลรูปตัว K



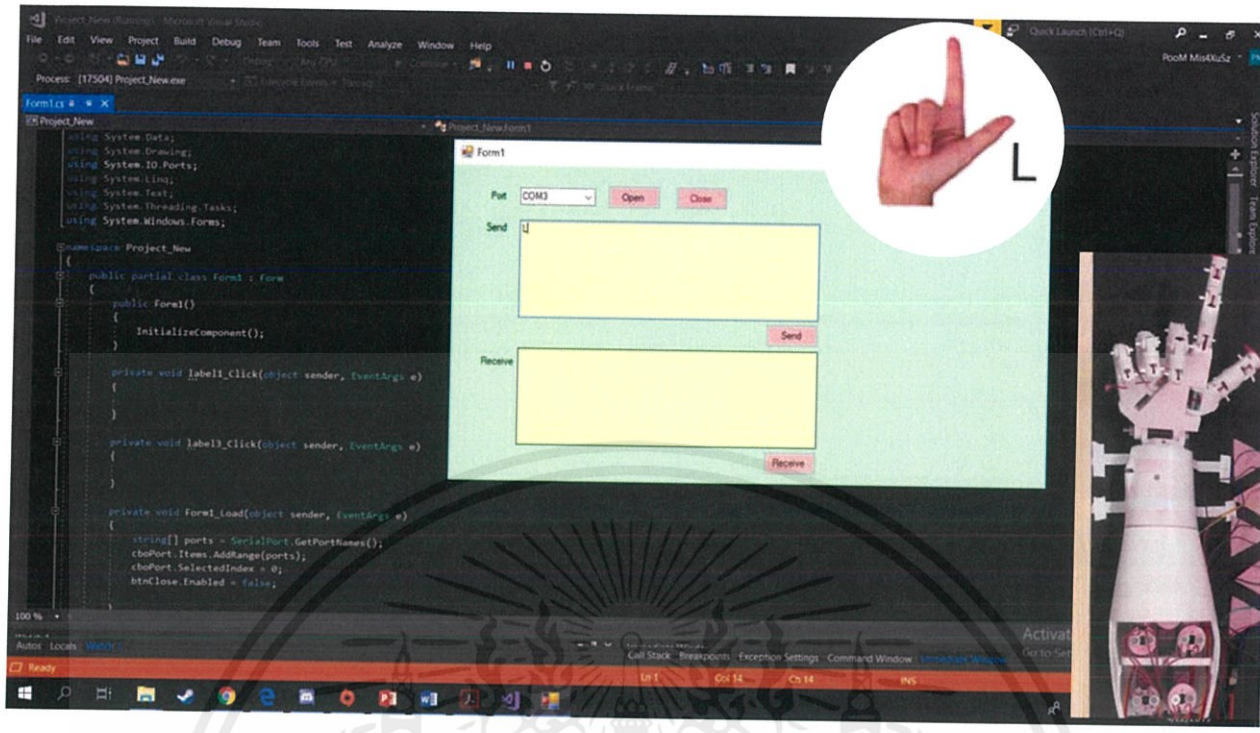
รูปที่4.13 ความสัมพันธ์ระหว่างโปรแกรมกับแขนกลรูปตัว M



รูปที่4.14 ความสัมพันธ์ระหว่างโปรแกรมกับแขนกลรูปตัว I



รูปที่4.15 ความสัมพันธ์ระหว่างโปรแกรมกับแขนกลรูปตัว T



รูปที่4.15 ความสัมพันธ์ระหว่างโปรแกรมกับแขนกลรูปตัว L

บทที่ 5

สรุปและวิจารณ์

การจัดทำโครงงานแขนกลที่ควบคุมด้วยไมโครคอนโทรลเลอร์นี้ สามารถสรุปและวิจารณ์ได้ดังนี้

5.1สรุปการทำงานของโครงงาน

แขนกลที่มีความยาวไม่เกิน 55 cm น้ำหนัก 1 kg การควบคุมเซอโวนั้นสามารถทำตามที่กำหนดไว้ตรงกับขอบเขตวัตถุประสงค์ตามที่ต้องการ โดยแขนกลสามารถควบคุมการงอขึ้น-ลงของนิ้วมือได้ตามที่เราต้องการ สามารถเคลื่อนที่ในแนวขึ้นลง และแนวแกนหมุนตามชุดคำสั่ง โดยสั่งจากการเขียนโปรแกรมที่คอมพิวเตอร์จะส่งคำสั่งไปให้ไมโครคอนโทรเลอร์แล้วไมโครคอนโทรลเลอร์ก็จะสั่งให้เซอร์โวทำงานตามคำสั่ง

5.2ปัญหาที่พบและแนวทางการแก้ไข

จากการศึกษาและรวบรวมข้อมูลในการทำแขนกลในช่วงแรกมีปัญหาที่เกิดขึ้น คือการใช้วงจรอิเล็คทรอนิกส์สำเร็จรูปและเซอโวที่ยังไม่คุ้นเคยกับการใช้งานของวงจรกับเซอโวชนิดนี้มาก่อน จึงเกิดปัญหาในการเขียนโปรแกรมป้อนข้อมูลลงไปทำให้แขนกลไม่ทำงานได้ตามต้องการ ต่อมาจึงได้ศึกษาข้อมูลและทดลองการเขียนโปรแกรมแบบต่างๆเพื่อที่จะหาคำสั่งที่สามารถควบคุมเซอโวได้ดีที่สุดและเกิดความผิดพลาดน้อยที่สุด นอกจากนั้นโครงสร้างยังทำมาจากพลาสติกซึ่งมีความแข็งแรงค่อนข้างต่ำทำให้เกิดความเสียหายได้ง่าย นอกจากนี้อาจทำการเปลี่ยนเซอร์โวให้มีคุณภาพที่สูงขึ้นและคุณสมบัติที่เหมาะกับงานที่ต้องนำมาใช้สร้างแขนกลชนิดเดิม และสิ่งที่เป็นปัญหามากที่สุดสำหรับการทำงานในครั้งนี้คือการยึดเว้นเอ็นไว้กับจานหมุนเซอโวเมื่อเซอโวหมุนไปทางด้านและกำลังจะหมุนกลับเส้นเอ็นที่ยึดอยู่บริเวณจานเซอโวได้หลุดลงไปอยู่ใต้แกนจนไม่สามารถทำให้เซอโวดึงเส้นเอ็นต่อไปได้ส่วนวิธีการแก้ไขปัญหาคือต้องพยายามยึดเอ็ดให้แน่นโดยเมื่อพันที่จานรอบเซอโวแล้วปลายของเส้นเอ็นจะนำขึ้นไปยึดอยู่กับน็อตบนจานเซอโวให้โอกาศที่เอ็นจะหลุดนั้นมีน้อยลง

5.3 ข้อเสนอแนะ

จากโครงสร้างแขนกลที่ใช้พลาสติกเป็นส่วนประกอบของโครงสร้าง ซึ่งเป็นวัสดุไม่คงทนดังนั้นในการสร้างส่วนประกอบโครงสร้างแขนกลนั้นควรเป็นวัสดุที่มีความแข็งแรงทนทานต่อแรงรับที่กระทำเช่น อลูมิเนียม สเตนเลส เส้นเอ็นควรมีขนาดหนาขึ้นเพื่อรองรับแรงที่เซอร์โวกระทำทำให้มีประสิทธิภาพในการทำรูปมือให้เป็นแบบตามที่เราต้องการ และเราควรศึกษาว่าใช้เซอร์โวชนิดใดขนาดใดจึงจะเหมาะสมกับกับงบประมาณที่เรามีและสามารถทำให้แขนกลทำงานได้ดีเหมาะสมตามที่เราต้องการ

และข้อเสนอแนะแบ่งเป็นส่วนต่างๆดังนี้

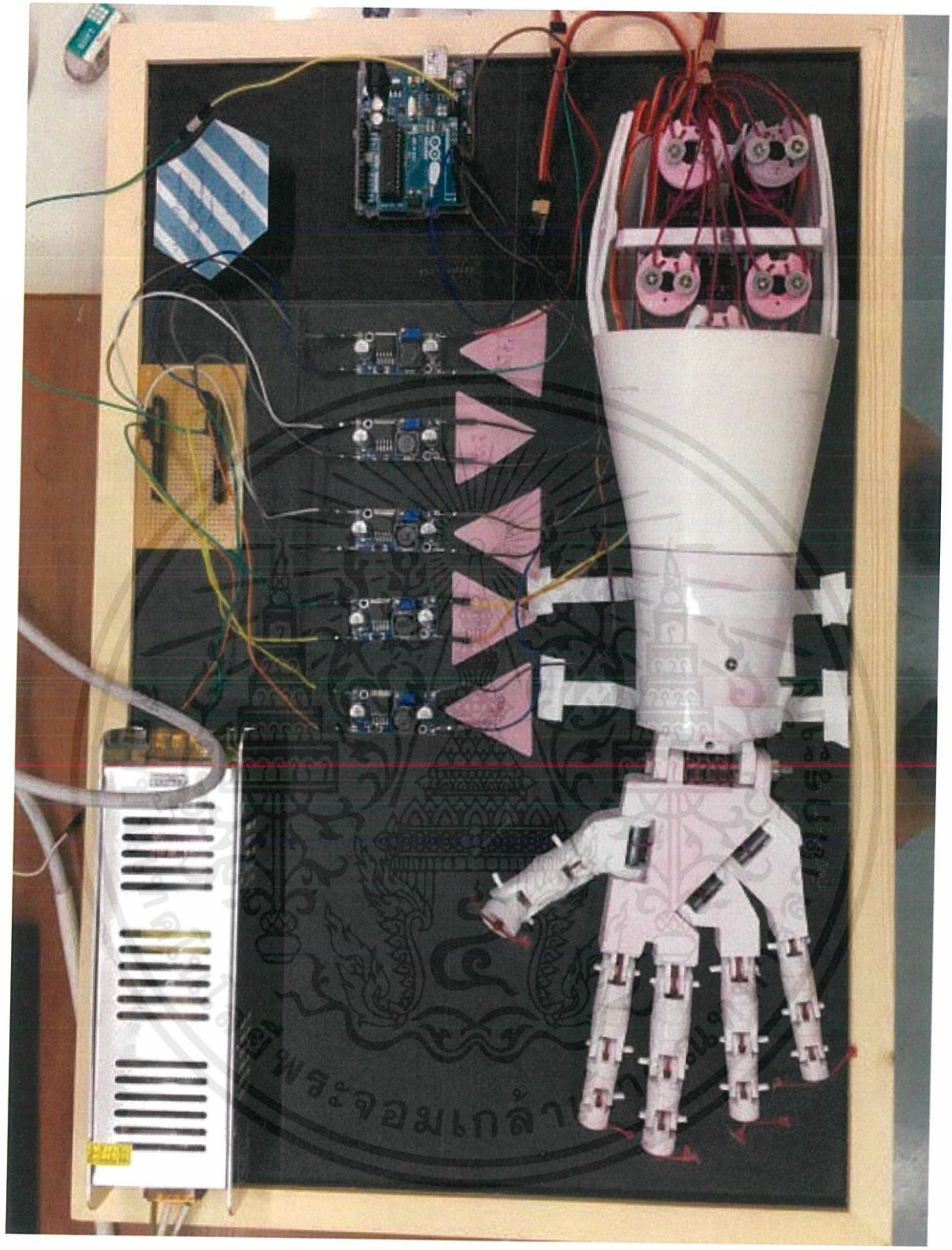
1.ควรมีการนำไปประยกต์ใช้กับการทำงานโดยอัตโนมัติ หรืออาจจะมีเซนเซอร์ และกล้องเข้ามามีบทบาทด้วย

2.ใช้วัสดุที่มีความแข็งแรงทนทานในการทำโครงสร้างของชิ้นงาน

3.ควรศึกษาและคำนวณการเลือกใช้เซอร์โวมอเตอร์ในแต่ละแกน

4.สามารถนำไปประยุกต์ใช้ในงานด้านอุตสาหกรรมที่มีการเขียนโปรแกรมลงไปเผื่อสามารถใช้รทภาษามือได้อย่าต่อเนื่อง

ภาคผนวก



Source code

```
#include <Servo.h>

int inByte = 0;          // sets up the variable that will store incoming data
from serial (keyboard)

Servo servo0;            // servo finger 1

Servo servo1;            // finger 2

Servo servo2;            // finger 3

Servo servo3;            // finger 4

Servo servo4;            // finger 5

Servo servo5;            // wrist

int Start = 0;

char Love;


void setup() {           // put your setupcode here, to run once:

  Serial.begin(9600);    // sets the serial keyboard to 9600 speed

  servo0.attach(2);      // finger 1

  servo1.attach(3);      // finger 2

  servo2.attach(4);      // finger 3

  servo3.attach(5);      // finger 4
```

```
  servo4.attach(6);          // finger 5
  servo5.attach(7);          // wrist
  servo0.write(180);          // stop
  servo1.write(180);          // stop
  servo2.write(180);          // stop
  servo3.write(180);          // stop
  servo4.write(180);          // stop

}
void loop() {
  // put your maincode here, to run repeatedly:
  inByte = 0;                 // resets serial variable to zero each time so code only
blinks once
  if (Serial.available() > 0){  // checks if there is any incoming data from the
serial (keyboard)
  inByte = Serial.read();}      // if yes, stores the incoming serial data into the
inByte variable

 char Start = 'S';
if(inByte == 'S'){
```

```
  Serial.println("Let's Go");
  servo0.writeMicroseconds(544);
  servo1.writeMicroseconds(544);
  servo2.writeMicroseconds(544);
  servo3.writeMicroseconds(544);
  servo4.writeMicroseconds(544);

  delay(3000);
  servo0.writeMicroseconds(2400);
  servo1.writeMicroseconds(2400);
  servo2.writeMicroseconds(2400);
  servo3.writeMicroseconds(2400);
  servo4.writeMicroseconds(2400);

  delay(2000);
  inByte = 0;

}
```

```
 char  Love = 'I';
if (inByte == 'I')
{                          // checks if letter a was typed on serial
servo0.writeMicroseconds(2400);
servo1.writeMicroseconds(544);
servo2.writeMicroseconds(544);
servo3.writeMicroseconds(544);
servo4.writeMicroseconds(544);
delay(4000);
servo0.writeMicroseconds(2400);
servo1.writeMicroseconds(2400);
servo2.writeMicroseconds(2400);
servo3.writeMicroseconds(2400);
servo4.writeMicroseconds(2400);
delay(2000);
}
char  Love0 = 'L';
if (inByte == 'L'){
servo0.writeMicroseconds(2400);
```

```
servo1.writeMicroseconds(544);
servo2.writeMicroseconds(544);
servo3.writeMicroseconds(2400);
servo4.writeMicroseconds(2400);
delay(4000);
servo0.writeMicroseconds(2400);
servo1.writeMicroseconds(2400);
servo2.writeMicroseconds(2400);
servo3.writeMicroseconds(2400);
servo4.writeMicroseconds(2400);
delay(2000);}
char  Love1 = 'U';
if (inByte == 'U'){
servo0.writeMicroseconds(544);
servo1.writeMicroseconds(544);
servo2.writeMicroseconds(2400);
servo3.writeMicroseconds(2400);
servo4.writeMicroseconds(544);
delay(4000);
```

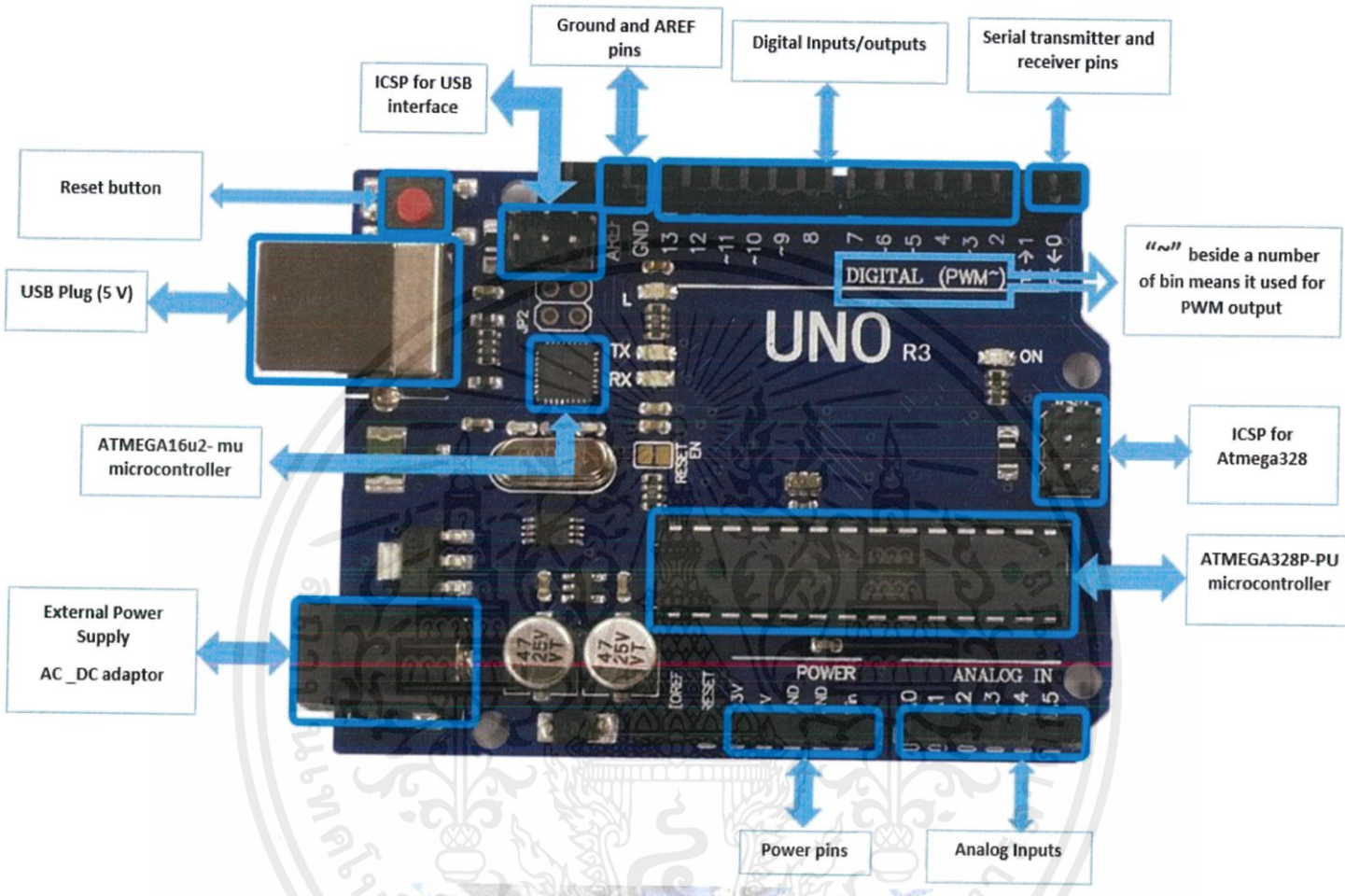
```
  servo0.writeMicroseconds(2400);
  servo1.writeMicroseconds(2400);
  servo2.writeMicroseconds(2400);
  servo3.writeMicroseconds(2400);
  servo4.writeMicroseconds(2400);
delay(2000);
 }
char KMITL = 'K';
if (inByte == 'K'){
  servo0.writeMicroseconds(544); // K
  servo1.writeMicroseconds(544);
  servo2.writeMicroseconds(2400);
  servo3.writeMicroseconds(2400);
  delay(2000);
  servo4.writeMicroseconds(1000);
  delay(3000);
  servo0.writeMicroseconds(2400); // 0
  servo1.writeMicroseconds(2400);
  servo2.writeMicroseconds(2400);
```

```
  servo3.writeMicroseconds(2400);
  servo4.writeMicroseconds(2400);
delay(2000);
  servo4.writeMicroseconds(544); // M
  delay(1000);
  servo0.writeMicroseconds(544);
  servo1.writeMicroseconds(544);
  servo2.writeMicroseconds(544);
  servo3.writeMicroseconds(544);
delay(3000);
  servo0.writeMicroseconds(2400);  // 0
  servo1.writeMicroseconds(2400);
  servo2.writeMicroseconds(2400);
  servo3.writeMicroseconds(2400);
  servo4.writeMicroseconds(2400);
delay(2000);
  servo0.writeMicroseconds(2400);  // I
  servo1.writeMicroseconds(544);
  servo2.writeMicroseconds(544);
```

```
  servo3.writeMicroseconds(544);
  delay(1000);
  servo4.writeMicroseconds(544);
delay(3000);
  servo0.writeMicroseconds(2400);  // 0
  servo1.writeMicroseconds(2400);
  servo2.writeMicroseconds(2400);
  servo3.writeMicroseconds(2400);
  servo4.writeMicroseconds(2400);
delay(2000);
  servo0.writeMicroseconds(544);  // T
  servo1.writeMicroseconds(544);
  servo2.writeMicroseconds(544);
  servo3.writeMicroseconds(544);
  delay(1000);
  servo4.writeMicroseconds(544);
delay(3000);
  servo0.writeMicroseconds(2400);  // 0
  servo1.writeMicroseconds(2400);
```

```
    servo2.writeMicroseconds(2400);
    servo3.writeMicroseconds(2400);
    servo4.writeMicroseconds(2400);
  delay(2000);
    servo0.writeMicroseconds(544);  // L
    servo1.writeMicroseconds(544);
    servo2.writeMicroseconds(544);
    servo3.writeMicroseconds(2400);
    servo4.writeMicroseconds(2400);
 delay(3000);
    servo0.writeMicroseconds(2400);  // 0
    servo1.writeMicroseconds(2400);
    servo2.writeMicroseconds(2400);
    servo3.writeMicroseconds(2400);
    servo4.writeMicroseconds(2400);
 delay(2000);
 }
 }
```

Arduino Uno R3



INTRODUCTION

Arduino is used for building different types of electronic circuits easily using of both a physical programmable circuit board usually microcontroller and piece of code running on computer with USB connection between the computer and Arduino.

Programming language used in Arduino is just a simplified version of C++ that can easily replace thousands of wires with words.

ARDUINO UNO-R3 PHYSICAL COMPONENTS

ATMEGA328P-PU microcontroller

The most important element in Arduino Uno R3 is ATMEGA328P-PU is an 8-bit Microcontroller with flash memory reach to 32k bytes. It's features as follow:

- High Performance, Low Power AVR
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Up to 20 MIPS Throughput at 20 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32K Bytes of In-System Self-Programmable Flash program memory
  - 256/512/512/1K Bytes EEPROM
  - 512/1K/1K/2K Bytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C
  - Optional Boot Code Section with Independent Lock Bits
  - In-System Programming by On-chip Boot Program
  - True Read-While-Write Operation
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
  - Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
  - Temperature Measurement
  - Programmable Serial USART

- Master/Slave SPI Serial Interface
- Byte-oriented 2-wire Serial Interface (Philips I2 C compatible)
- Programmable Watchdog Timer with Separate On-chip Oscillator
- On-chip Analog Comparator
- Interrupt and Wake-up on Pin Change

• Special Microcontroller Features

- Power-on Reset and Programmable Brown-out Detection
- Internal Calibrated Oscillator
- External and Internal Interrupt Sources
- Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby

• I/O and Packages

- 23 Programmable I/O Lines
- 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF

• Operating Voltage:

- 1.8 - 5.5V

• Temperature Range:

- -40°C to 85°C

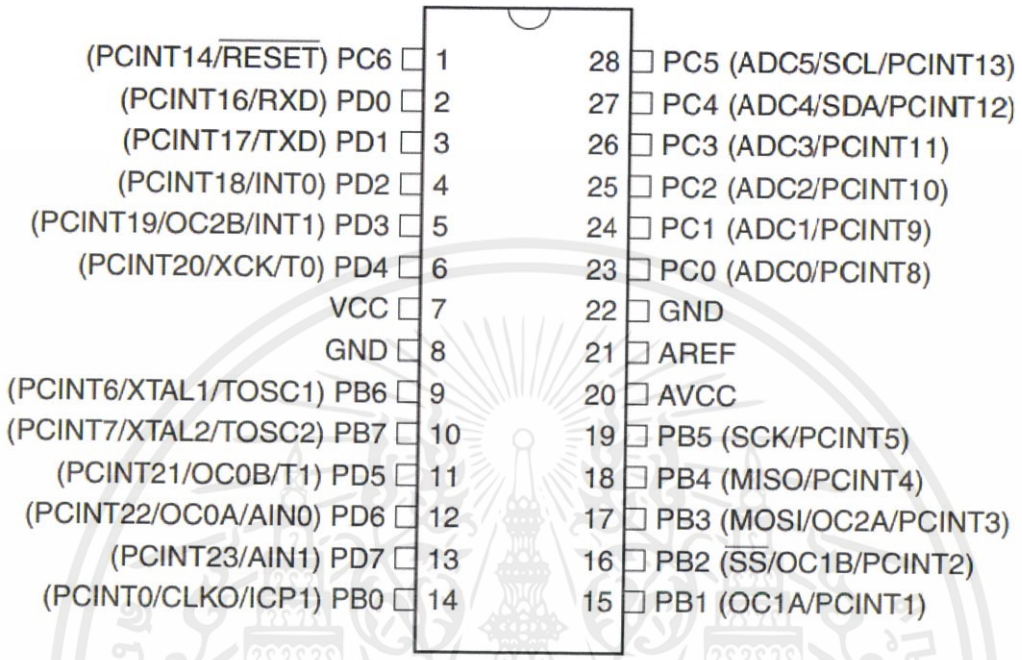
• Speed Grade:

- 0 - 4 MHz@1.8 - 5.5V, 0 - 10 MHz@2.7 - 5.5.V, 0 - 20 MHz @ 4.5 - 5.5V

• Power Consumption at 1 MHz, 1.8V, 25°C

- Active Mode: 0.2 mA
- Power-down Mode: 0.1 µA
- Power-save Mode: 0.75 µA (Including 32 kHz RTC)

- Pin configuration



ATMEGA16u2- mu microcontroller

Is a 8-bit microcontroller used as USB driver in Arduino uno R3 it's features as follow:

- High Performance, Low Power AVR

• Advanced RISC Architecture

- 125 Powerful Instructions – Most Single Clock Cycle Execution
- 32 x 8 General Purpose Working Registers
- Fully Static Operation
- Up to 16 MIPS Throughput at 16 MHz

• Non-volatile Program and Data Memories

- 8K/16K/32K Bytes of In-System Self-Programmable Flash
- 512/512/1024 EEPROM
- 512/512/1024 Internal SRAM
- Write/Erase Cycles: 10,000 Flash/ 100,000 EEPROM
- Data retention: 20 years at 85˚C/ 100 years at 25˚C

- Optional Boot Code Section with Independent Lock Bits
- In-System Programming by on-chip Boot Program hardware-activated after reset
- Programming Lock for Software Security

• USB 2.0 Full-speed Device Module with Interrupt on Transfer Completion

- Complies fully with Universal Serial Bus Specification REV 2.0
- 48 MHz PLL for Full-speed Bus Operation: data transfer rates at 12 Mbit/s
- Fully independent 176 bytes USB DPRAM for endpoint memory allocation
- Endpoint 0 for Control Transfers: from 8 up to 64-bytes
- 4 Programmable Endpoints:
  - IN or Out Directions
  - Bulk, Interrupt and Isochronous Transfers
  - Programmable maximum packet size from 8 to 64 bytes
  - Programmable single or double buffer
- Suspend/Resume Interrupts
- Microcontroller reset on USB Bus Reset without detach
- USB Bus Disconnection on Microcontroller Request

• Peripheral Features

- One 8-bit Timer/Counters with Separate Prescaler and Compare Mode (two 8-bit PWM channels)
- One 16-bit Timer/Counter with Separate Prescaler, Compare and Capture Mode(three 8-bit PWM channels)
- USART with SPI master only mode and hardware flow control (RTS/CTS)
- Master/Slave SPI Serial Interface
- Programmable Watchdog Timer with Separate On-chip Oscillator
- On-chip Analog Comparator
- Interrupt and Wake-up on Pin Change

• On Chip Debug Interface (debug WIRE)

• Special Microcontroller Features

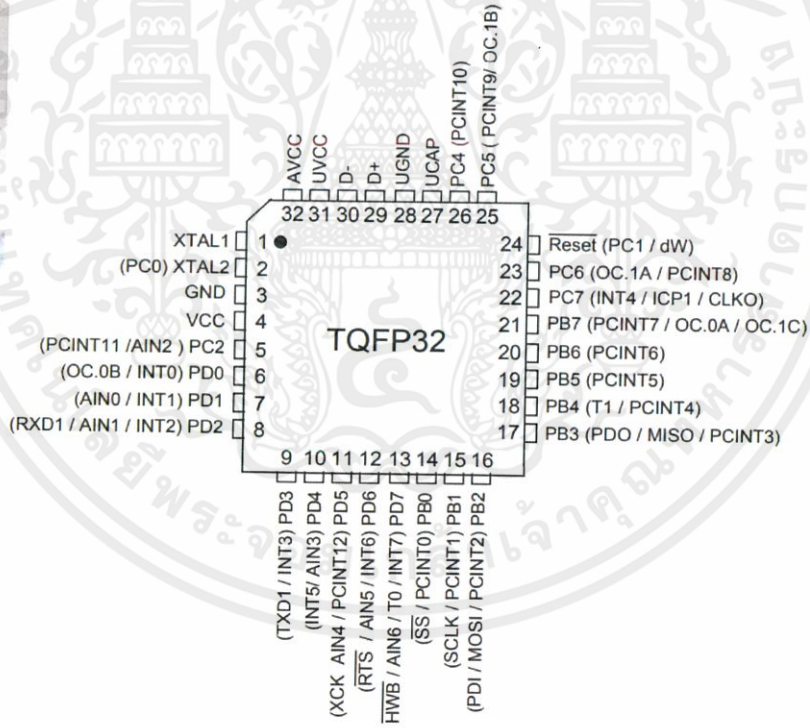
- Power-On Reset and Programmable Brown-out Detection
- Internal Calibrated Oscillator
- External and Internal Interrupt Sources
- Five Sleep Modes: Idle, Power-save, Power-down, Standby, and Extended Standby

• I/O and Packages

- 22 Programmable I/O Lines
- QFN32 (5x5mm) / TQFP32 packages

- Operating Voltages
  - 2.7 - 5.5V
- Operating temperature
  - Industrial (-40°C to +85°C)
- Maximum Frequency
  - 8 MHz at 2.7V - Industrial range
  - 16 MHz at 4.5V - Industrial range

- Pin configuration



OTHER ARDUINO UNO R3 PARTS

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode (), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 k Ohms. In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
- LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

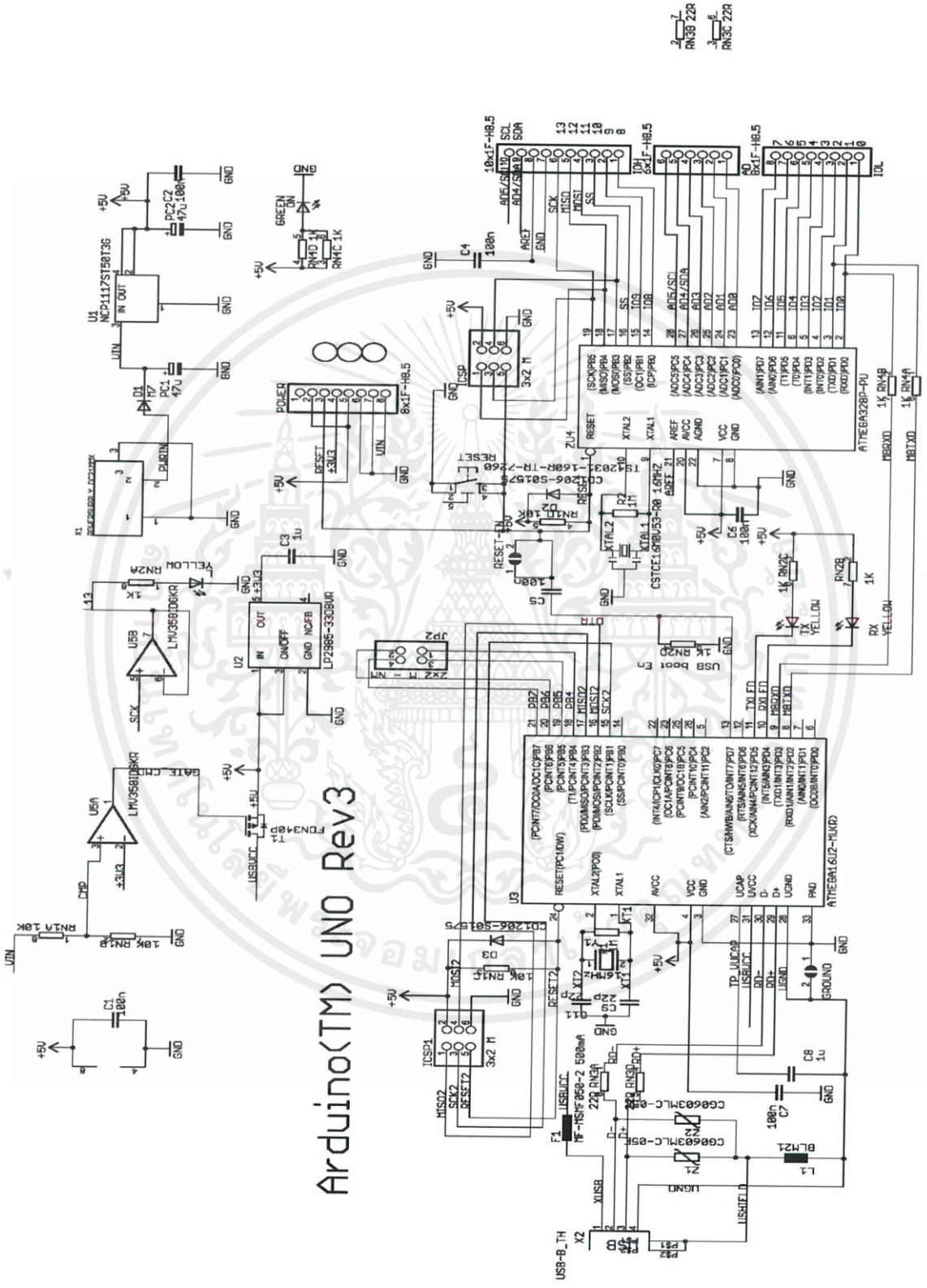
The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e.1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

- TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

There are a couple of other pins on the board:

- AREF: Reference voltage for the analog inputs. Used with analogReference().
- Reset: Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

ARDUINO UNO R3 SCHEMATIC DIAGRAM

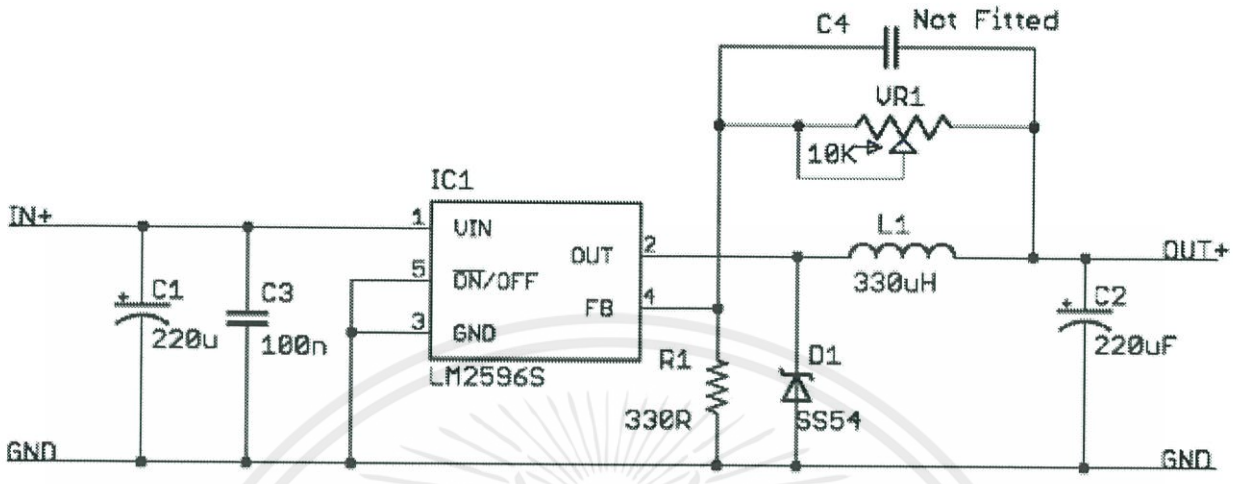


LM2596 DC-DC Adjustable PSU Module

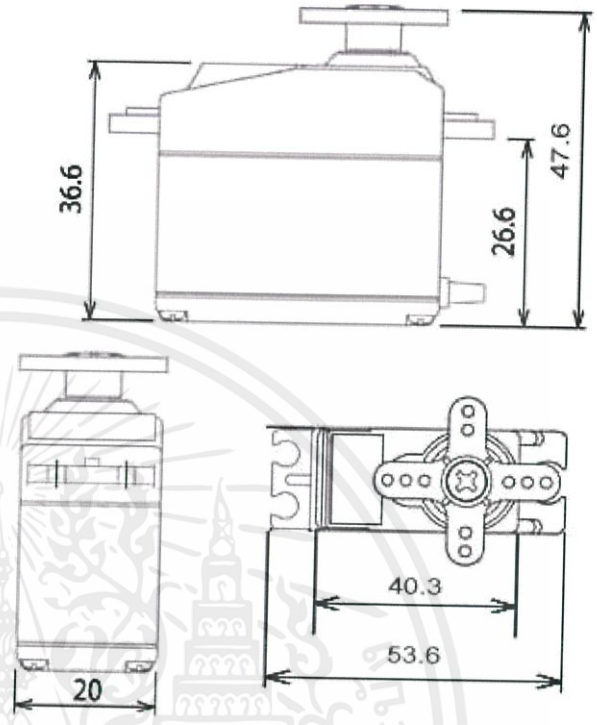
LM2596 DC to DC step down regulator, adjustable +1.23 to 35vdc output, 2A. Ideal for battery operated projects requiring a regulated powersupply.

Specifications

Regulator Type:	Step Down (Non Isolated input to Output)
Input Voltage:	+4 to 40vdc
Output Voltage:	+1.23 to 35vdc
Output Current:	2A rated, (3A maximum with heatsink)
Efficiency:	Up to 92% (when output voltage is set high)
Switching Frequency:	150kHz
Dropout Voltage:	2vdc minimum
Protection:	Short circuit current limiting
Load Regulation:	+/- 0.5%
Voltage Regulation:	+/- 2.5%
Temperature:	-40 to +85 deg C (output power less than 10Watts)
Board Size:	43.6mm L x 21mm W x 14mm H
Data Sheet:	National LM2596



MG996R High Torque
Metal Gear Dual Ball Bearing Servo



This High-Torque MG996R Digital Servo features metal gearing resulting in extra high 10kg stalling torque in a tiny package. The MG996R is essentially an upgraded version of the famous MG995 servo, and features upgraded shock-proofing and a redesigned PCB and IC control system that make it much more accurate than its predecessor. The gearing and motor have also been upgraded to improve dead bandwith and centering. The unit comes complete with 30cm wire and 3 pin 'S' type female header connector that fits most receivers, including Futaba, JR, GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum and Hitec.

This high-torque standard servo can rotate approximately 120 degrees (60 in each direction). You can use any servo code, hardware or library to control these servos, so it's great for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. The MG996R Metal Gear Servo also comes with a selection of arms and hardware to get you set up nice and fast!

Specifications

- Weight: 55 g
- Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 9.4 kgf·cm (4.8 V), 11 kgf·cm (6 V)
- Operating speed: 0.17 s/60° (4.8 V), 0.14 s/60° (6 V)

- Operating voltage: 4.8 V a 7.2 V
- Running Current 500 mA – 900 mA (6V)
- Stall Current 2.5 A (6V)
- Dead band width: 5 μs
- Stable and shock proof double ball bearing design
- Temperature range: 0 °C – 55 °C