หุ่นยนต์อัจฉริยะภาพเพื่อการเกษตรและเทคโนโลยีชีวภาพ

Farming and Biotechnology Intelligent Robot

ณัฐกานต์ ใสสะอาด
Nattakan Saisa-ard

ปิยะ เดชาวาศน์
Piya Dechawart

พรภวิทย์ สุทิน
Pornpawit Sutin

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2561

หุ่นยนต์อัจฉริยะภาพเพื่อการเกษตรและเทคโนโลยีชีวภาพ

Farming and Biotechnology Intelligent Robot

โดย

ณัฐกานต์ ใสสะอาด

ปิยะ เดชาวาศน์

พรภวิทย์ สุทิน

อาจารย์ที่ปรึกษา

รศ. จิรวัฒน์ ปานกลาง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2561

ปริญญานิพนธ์ **ปีการศึกษา 2561**

สาขาวิชา วิศวกรรมอิเล็กทรอนิกส์

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง หุ่นยนต์อัจฉริยะภาพเพื่อการเกษตรและเทคโนโลยีชีวภาพ

Farming and Biotechnology Intelligent Robot

จัดทำโดย นางสาวณัฐกานต์ ใสสะอาด รหัสนักศึกษา 58010362

นายปิยะ เดชาวาศน์ รหัสนักศึกษา 58010791

นายพรภวิทย์ สุทิน รหัสนักศึกษา 58010839

ปริญญานิพนธ์นี้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

ลงชื่อ.....

อาจารย์ที่ปรึกษา

(รศ. จิรวัฒน์ ปานกลาง)

วันที่ ๕ / พ.ย. / ๖๒

หัวข้อปริญญานิพนธ์	หุ่นยนต์อัจฉริยะภาพเพื่อการเกษตรและเทคโนโลยีชีวภาพ		
นักศึกษา	นางสาวณัฐกานต์ ใสสะอาด	รหัสนักศึกษา	58010362
	นายปิยะ เดชาวาศน์	รหัสนักศึกษา	58010791
	นายพรภวิทย์ สุทิน	รหัสนักศึกษา	58010839
ปริญญา	วิศวกรรมศาสตร์บัณฑิต		
ภาควิชา	วิศวกรรมอิเล็กทรอนิกส์		
ปีการศึกษา	2561		
อาจารย์ที่ปรึกษาปริญญานิพนธ์	รศ.จิรวัฒน์ ปานกลาง		

บทคัดย่อ

หุ่นยนต์อัจฉริยะภาพเพื่อการเกษตรและเทคโนโลยีชีวภาพเพื่อช่วยการเพาะเมล็ดเป็นต้นอ่อนที่พร้อมลงแปลงปลูกในอนาคตซึ่งประกอบด้วยส่วนฐานและส่วนที่เคลื่อนที่โดยส่วนฐานมีลักษณะเป็นรูปสี่เหลี่ยม และมีตัวขับเคลื่อนที่แบบเชิงเส้นเชื่อมต่อกับส่วนฐาน และส่วนที่เคลื่อนที่ ทำให้ส่วนที่เคลื่อนที่นี้สามารถเคลื่อนที่ได้สามมิติ,หยอดเมล็ดพืชในตำแหน่งที่ต้องการและทำการเพาะเลี้ยงจนเมล็ดกลายเป็นต้นอ่อน โครงงานนี้คณะผู้จัดทำได้ออกแบบหุ่นยนต์ ซึ่งมีการใช้ stepping motor ทั้งหมด 4 ตัว เป็นตัวขับเคลื่อนให้เคลื่อนที่ไปได้ 3 แกน สั่งงานผ่านการควบคุมบอร์ด arduino โดยจะเคลื่อนที่ไปจุดที่กำหนดและใช้เซอร์โวมอเตอร์ควบคุมจำนวนหยอดเมล็ดพืช

Project Title	Farming and Biotechnology Intelligent Robot	
Student	Miss Nattakan Saisa-ard	Student ID 58010362
	Mr. Piya Dechawat	Student ID 58010791
	Mr. Pornpawit Sutin	Student ID 58010839
Degree	Bachelor of Engineering	
Program	Electronics Engineering	
Year	2018	
Project Advisor	Assoc. Prof. Jirawath Parnklang	

ABSTRACT

Farming and Biotechnology intelligent Robot for that help plant seeds until seeds grow in to sprout. The system consists of two parts, which are the base and the moving parts. The base has square shape and linear drive is connected the base. The moving parts able to move 3D, can drop the seeds to collect position and cultivate until seeds grow into sprout. In this project, we are designed the robot by using 4 stepping motors and able to move X, Y, Z axis. The moving part are controlled by arduino controller and move to collect positions. The plant seeds is controlled by servo motor to drop in the ground.

กิตติกรรมประกาศ

การจัดทำโครงงานและรายงานเรื่อง “หุ่นยนต์เพื่อการเกษตรและเทคโนโลยีชีวภาพ” ของหลักสูตรวิศวกรรมอิเล็กทรอนิกส์ชิ้นนี้สามารถสำเร็จลุล่วงไปด้วยดี ต้องขอขอบคุณ รศ.จิรวัฒน์ ปานกลาง อาจารย์ที่ปรึกษาโครงงาน ที่คอยให้คำแนะนำตลอดการทำงานและให้ความช่วยเหลืออย่างเต็มที่ รวมถึงเอื้อเฟื้ออุปกรณ์และสถานที่ ในการทำโครงงานครั้งนี้เป็นอย่างดียิ่ง นอกจากนี้ต้องขอขอบคุณอาจารย์ทุกท่านที่มอบวิชาความรู้และให้คำแนะนำเรื่องต่างๆ

สุดท้ายนี้ ต้องขอขอบคุณ และขอบใจ ครอบครัวและเพื่อนๆของคณะผู้จัดทำ ที่คอยให้กำลังใจและถามไถ่ความเป็นไปของโครงงานอยู่เสมอ ทำให้คณะผู้จัดทำ มีกำลังใจที่จะจัดทำโครงงานและรายงานฉบับนี้ให้สำเร็จลุล่วงคณะผู้จัดทำหวังว่าโครงงานนี้จะเป็นประโยชน์สำหรับผู้สนใจ

ณัฐกานต์ ใสสะอาด
ปิยะ เดชาวาศน์
พรภวิทย์ สุทิน
คณะผู้จัดทำ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูปภาพ

รูปที่ หน้า

สารบัญรูปภาพ (ต่อ)

รูปที่ หน้า

สารบัญรูปภาพ (ต่อ)

รูปที่ หน้า

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญ

ในปัจจุบันเทคโนโลยีเข้ามามีบทบาทอย่างมากในชีวิตประจำวันเช่น ด้านอุตสาหกรรมก็ต้องมีเทคโนโลยีเข้ามาขับเคลื่อน ด้านเกษตรกรรมที่ต้องเปลี่ยนจากแบบดั่งเดิมที่ใช้แรงงานคนในการเพาะปลูกเป็นแบบสมัยใหม่ที่การนำเอาเทคโนโลยีเข้ามาใช้หรือ smart farming เข้ามาบริหารและจัดการการเพาะปลูกของเกษตรกร เพื่อให้สอดคล้องกับ Thailand 4.0 ที่ช่วยพัฒนาประเทศเปลี่ยนแปลงเศรษฐกิจแบบดังเดิม ไปสู่เศรษฐกิจที่ขับเคลื่อนด้วยนวัตกรรม ภาคการเกษตรเขามามีบทบาทสำคัญต่อระบบเศรษฐกิจและวิถีชีวิตของคนไทตั้งแต่อดีตจนถึงปัจจุบันจากการผลิตเพื่อยังชีพมาเป็นการผลิตเพื่อการค้าและการส่งออก สินค้าส่งออกโดยแต่ก่อนในการทำเกษตรกรรมในแต่ละครั้งต้องใช้แรงและเวลาในการทำเกษตรมากพอสมควร ดังนั้นเทคโนโลยีจึงเข้ามามีบทบาทเพื่อให้ลดต้นทุนในการผลิตพืชผลทางการเกษตร ในการจัดทำโครงงานครั้งนี้ เพื่อศึกษาการทำงานของโปรแกรม Arduino ในการทำงานกับเครื่อง Farming and Biotechnology Intelligent Robot เพื่อให้ตอบสนองต่อความต้องการการใช้งาน และเพื่อการทำเทคโนโลยีมาพัฒนาและประยุกต์ใช้ให้เกิดประโยชน์ต่อด้านการเกษตร และในด้านอื่น ๆ ต่อไป

1.2 วัตถุประสงค์

1 เพื่อศึกษาการเขียนโปรแกรม Arduino

2 สามารถนำความรู้ที่ได้ศึกษาในห้องเรียนนำมาประยุกต์ใช้เพื่อสร้างชิ้นงาน

3 เพื่อให้มีความรู้ความเข้าใจเกี่ยวกับการควบคุมสเตปมอเตอร์ด้วย Arduino

4 เพื่อศึกษาการทำงานของโปรแกรม Arduino ในการสร้างนวัตกรรม

5 เพื่อศึกษาหาความรู้ด้วยตนเอง

1.3 ขอบเขตของโครงงาน

1 โครงงานนี้จะมีการพัฒนาส่วนของ Hardware และ software ที่ใช้ในการควบคุมการทำงานการหมุนของมอเตอร์เท่านั้น

2 สามารถควบคุมมอเตอร์เคลื่อนไปยังตำแหน่งที่เราต้องการ

3 ควบคุมการหยอดเมล็ดพืชตามที่เราต้องการ

1.4 ผลที่คาดว่าจะได้รับ

1 เข้าใจวิธีการทำงานของโปรแกรม Arduino ในการสั่งงานเครื่อง Farming and Biotechnology intelligent Robot

2 สามารถนำนวัตกรรมที่ได้ไปพัฒนาและต่อยอดในการนำไปใช้ในด้านนวัตกรรมเกษตร

3 มีทักษะในกระบวนการทำงานและสร้างนวัตกรรมเพิ่มขึ้น

4 สามารถนำนวัตกรรมไปใช้งานได้จริงและสามารถช่วยลดเวลาในการทำการเกษตร

1.5 รายละเอียดและเนื้อหาของโครงงาน

1.5.1 แผนการดำเนินโครงการ

1 ศึกษาสภาพปัญหาในเรื่องที่สนใจ พร้อมค้นหาข้อมูล ทฤษฎีที่เกี่ยวข้อง

2 ศึกษาเครื่องมือที่ใช้ในการทำโครงการ

3 ออกแบบโครงสร้างของชิ้นงาน

4 สร้างชิ้นงานและทดสอบชิ้นงาน

5 เก็บข้อมูลการทดลองของชิ้นงาน

6 สรุปผลการทดลองของชิ้นงาน

7 จัดทำรายงานฉบับสมบูรณ์

1.5.2 ระยะเวลาดำเนินงาน

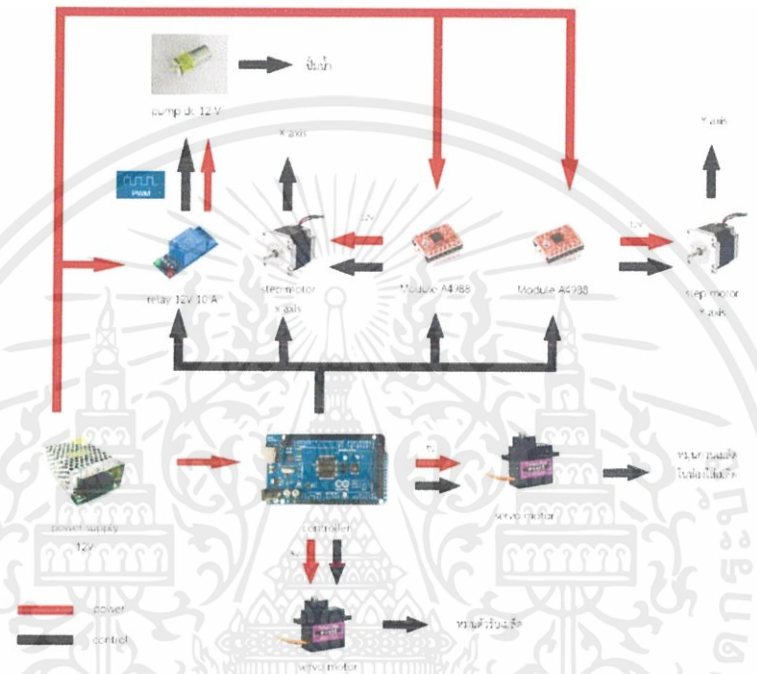
ตารางที่ 1.1 ระยะเวลาดำเนินงาน

แผนงานของการทำโครงงาน								
No.	ขอบเขตการทำ Project	Project I (๒๕๖๑)				Project II (๒๕๖๒)		
		ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.
๑	ศึกษาข้อมูลหน้าที่การทำงานของอุปกรณ์ต่างๆ	⟷						
๒	ซื้ออุปกรณ์ ที่จะนำไปใช้ในการทำ Project	⟷						
๓	ทำโครงสร้างใหม่ให้เหมาะสมกับการปลูกแบบไฮโดรโปนิกส์	⟷	⟷					
๔	สามารถบังคับให้ Stepping Moter ตามแนวแกน X , Y , Z	⟷	⟷	⟷				
๕	สามารถปล่อยเมล็ดพืชผัก ๓ เม็ด ไปตำแหน่งที่ปลูกได้	⟷	⟷	⟷	⟷			
๖	สามารถรดน้ำต้นไม้ได้					⟷	⟷	
๗	ทำโครงสร้างใหม่ให้เรียบร้อย					⟷	⟷	⟷

บทที่ 2

ทฤษฎีพื้นฐานและหลักการทำงาน

ความรู้พื้นฐานที่ได้ศึกษาค้นคว้าเกี่ยวกับโครงงานหุ่นยนต์อัจฉริยะภาพเพื่อการเกษตรและเทคโนโลยีชีวภาพแบ่งออกได้ดังนี้



รูปที่ 2.1 แสดงบล็อกไดอะแกรมของหุ่นยนต์อัจฉริยะภาพเพื่อการเกษตรและเทคโนโลยีชีวภาพ

2.1 Arduino คืออะไร

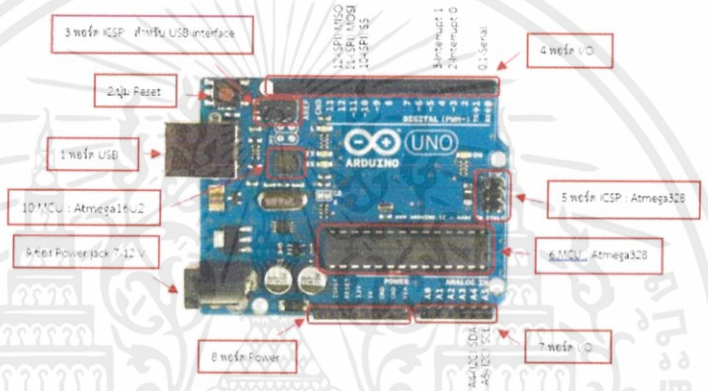
Arduino อ่านว่า (อา-ดู-อิ-โน่ หรือ อาดุยโน่) เป็นบอร์ดไมโครคอนโทรเลอร์ตระกูล AVR ที่มีการพัฒนาแบบ Open Source คือมีการเปิดเผยข้อมูล ทั้งด้าน Hardware และ Software ตัวบอร์ด Arduino ถูกออกแบบมาให้ใช้งานได้ง่าย ดังนั้นจึงเหมาะสำหรับผู้เริ่มต้นศึกษา ทั้งนี้ผู้ใช้งานยังสามารถดัดแปลง เพิ่มเติม พัฒนาต่อยอดทั้งตัวบอร์ด หรือโปรแกรมต่อได้อีกด้วย

ความง่ายของบอร์ด Arduino ในการต่ออุปกรณ์เสริมต่างๆ คือผู้ใช้งานสามารถต่อวงจรอิเล็กทรอนิคส์จากภายนอกแล้วเชื่อมต่อเข้ามาที่ขาI/Oของบอร์ดหรือเพื่อความสะดวกสามารถเลือกต่อกับบอร์ดเสริม (Arduino Shield) ประเภทต่างๆ เช่น Arduino XBee Shield, Arduino Music Shield, Arduino Relay Shield, Arduino Wireless Shield, Arduino GPRS Shield เป็นต้น มาเสียบกับบอร์ดบนบอร์ด Arduino แล้วเขียนโปรแกรมพัฒนาต่อได้เลย

2.1.1 จุดเด่นที่ทำให้บอร์ด Arduino เป็นที่นิยม

1. ง่ายต่อการพัฒนา มีรูปแบบคำสั่งพื้นฐาน ไม่ซับซ้อนเหมาะสำหรับผู้เริ่มต้น
2. มี Arduino Community กลุ่มคนที่ร่วมกันพัฒนาที่แข็งแรง
3. Open Hardware ทำให้ผู้ใช้สามารถนำบอร์ดไปต่อยอดใช้งานได้หลายด้าน
4. ราคาไม่แพง
5. Cross Platform สามารถพัฒนาโปรแกรมบน OS ใดก็ได้

2.1.2 ส่วนประกอบหลักของบอร์ด Arduino



รูปที่ 2.2 แสดงส่วนต่างๆของArduino 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/O Port:**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/O Port:** นอกจากจะเป็น Digital I/O แล้ว ยังเปลี่ยนเป็น ช่องรับสัญญาณอนาล็อก ตั้งแต่ขา A0-A5
8. **Power Port:** ไฟเลี้ยงของบอร์ดเมื่อต้องการจ่ายไฟให้กับวงจรภายนอก ประกอบด้วยขาไฟเลี้ยง +3.3 V, +5V, GND, V_{in}
9. **Power Jack:** รับไฟจาก Adapter โดยที่แรงดันอยู่ระหว่าง 7-12 V
10. **MCU** ของ Atmega16U2 เป็น MCU ที่ทำหน้าที่เป็น USB to Serial โดย Atmega328 จะติดต่อกับ Computer ผ่าน Atmega

2.1.3 Arduino Uno R3

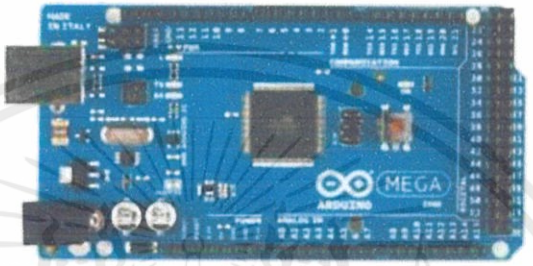
Arduino Uno R3 เป็นบอร์ด Arduino ที่ได้รับความนิยมมากที่สุด เนื่องจากราคาไม่แพง ส่วนใหญ่โปรเจคและ Library ต่างๆที่พัฒนาขึ้นมา Support จะอ้างอิงกับบอร์ดนี้เป็นหลัก เนื่องจากเป็นขนาดที่เหมาะสำหรับการเริ่มต้นเรียนรู้ Arduino และมี Shields ให้เลือกใช้งานได้มากกว่าบอร์ด Arduino รุ่นอื่นๆที่ออกแบบมาเฉพาะมากกว่า โดยบอร์ด Arduino Uno ได้มีการพัฒนาเรื่อยมาตั้งแต่ R2 R3 และรุ่นย่อยที่เปลี่ยนชิปไอซีเป็นแบบ SMD และข้อดีอีกอย่างคือ กรณีที่ MCU เสีย ผู้ใช้งานสามารถซื้อมาเปลี่ยนเองได้ง่าย

ข้อมูลจำเพาะ

ชิปไอซีไมโครคอนโทรเลอร์	ATmega328
ใช้แรงดันไฟฟ้า	5V
รองรับการจ่ายแรงดันไฟฟ้า (ที่แนะนำ)	7 – 12V
รองรับการจ่ายแรงดันไฟฟ้า (ที่จำกัด)	6 – 20V
พอร์ต Digital I/O	14 พอร์ต (มี 6 พอร์ต PWM output)
พอร์ต Analog Input	6 พอร์ต
กระแสไฟที่จ่ายได้ในแต่ละพอร์ต	40mA
กระแสไฟที่จ่ายได้ในพอร์ต 3.3V	50mA
พื้นที่โปรแกรมภายใน	32KB พื้นที่โปรแกรม, 500B ใช้โดย Bootloader
พื้นที่แรม	2KB
พื้นที่หน่วยความจำถาวร (EEPROM)	1KB
ความถี่คริสตัล	16MHz
ขนาด	68.6x53.4 mm
น้ำหนัก	25 กรัม

2.1.4 Arduino Mega 2560 R3

บอร์ด Arduino Mega 2560 R3เป็นบอร์ด Arduino ที่ออกแบบมาสำหรับงานที่ต้องใช้ I/O มากกว่า Arduino Uno R3 เช่น งานที่ต้องการรับสัญญาณจาก Sensor หรือควบคุมมอเตอร์ Servo หลายๆ ตัว ทำให้ Pin I/O ของบอร์ด Arduino Uno R3 ไม่สามารถรองรับได้ ทั้งนี้บอร์ด Mega 2560 R3 ยังมีความหน่วยความจำแบบ Flash มากกว่า Arduino Uno R3 ทำให้สามารถเขียนโค้ดโปรแกรมเข้าไปได้มากกว่า ในความเร็วของ MCU ที่เท่ากัน

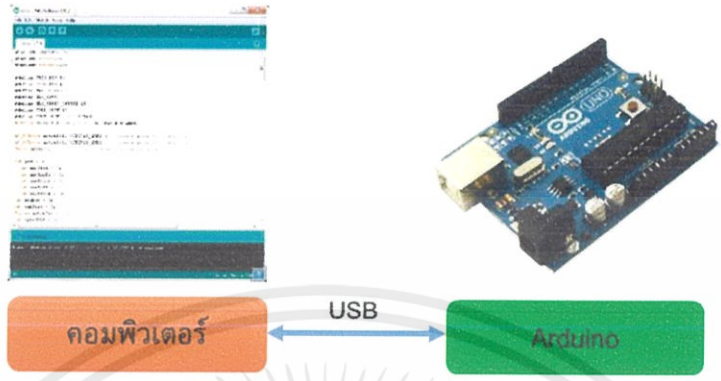


รูปที่ 2.3 แสดงบอร์ด Arduino Mega 2560 R3

ข้อมูลจำเพาะ

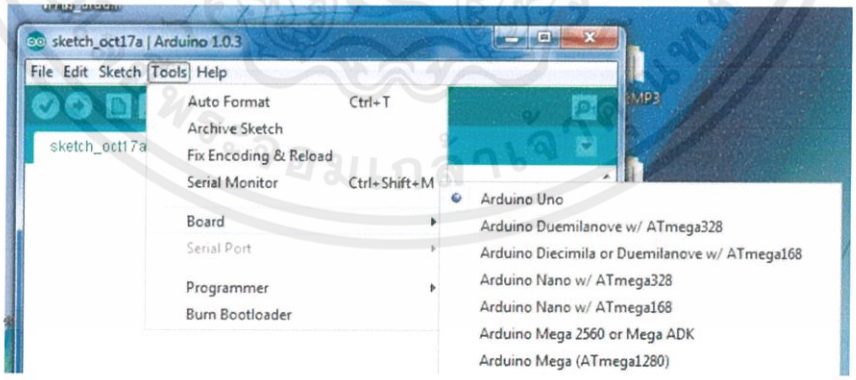
ชิปไอซีไมโครคอนโทรเลอร์	ATmega2560
ใช้แรงดันไฟฟ้า	5V
รองรับการจ่ายแรงดันไฟฟ้า (ที่แนะนำ)	7 – 12V
รองรับการจ่ายแรงดันไฟฟ้า (ที่จำกัด)	6 – 20V
พอร์ต Digital I/O	54 พอร์ต (มี 15 พอร์ต PWM output)
พอร์ต Analog Input	16 พอร์ต
กระแสไฟฟ้ารวมที่จ่ายได้ในทุกพอร์ต	40mA
กระแสไปที่จ่ายได้ในพอร์ต 3.3V	50mA
พื้นที่โปรแกรมภายใน	256KB แต่ 8KB ถูกใช้โดย Bootloader
พื้นที่แรม	8KB
พื้นที่หน่วยความจำถาวร (EEPROM)	4KB
ความถี่คริสตัล	16MHz

2.1.5 รูปแบบการเขียนโปรแกรมบน arduino

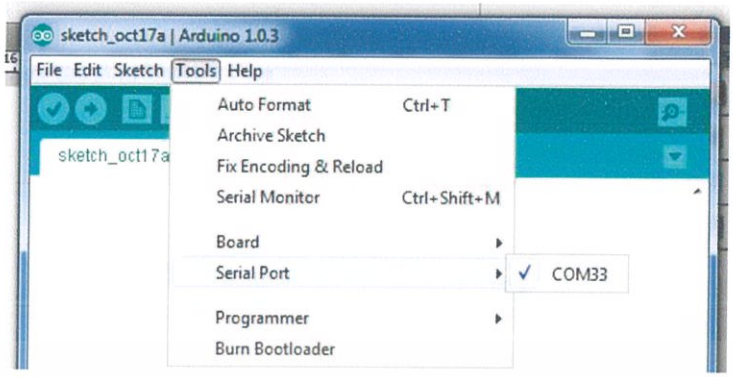


รูปที่ 2.4 แสดงการเชื่อมต่อบอร์ด arduino กับคอมพิวเตอร์

1. เขียนโปรแกรมบนคอมพิวเตอร์ผ่านทางโปรแกรม ArduinoIDE
2. หลังจากที่เขียนโค้ดโปรแกรมเรียบร้อยแล้วให้ผู้ใช้งานเลือกรุ่นบอร์ดArduinoที่ใช้และหมายเลข Com port
3. กดปุ่ม Verify เพื่อตรวจสอบความถูกต้องและ Compile โค้ดโปรแกรม จากนั้นกดปุ่ม Upload โค้ด โปรแกรมไปยังบอร์ด Arduinoผ่านทางสาย USB เมื่ออับโหลดเรียบร้อยแล้ว จะแสดงข้อความแถบข้างล่าง “Done uploading” และบอร์ดจะเริ่มทำงานตามที่เขียนโปรแกรมไว้ได้ทันที



รูปที่ 2.5 แสดงการเลือกรุ่นบอร์ด Arduino



รูปที่ 2.6 แสดงการเลือกหมายเลข comport ของบอร์ด

รูปที่5 กดปุ่ม Verify เพื่อตรวจสอบความถูกต้องและ Compile โค้ดโปรแกรม

รูปที่6 Upload โค้ดโปรแกรม

รูปที่ 2.7 โค๊ดโปรแกรม

2.1.6 โครงสร้างโปรแกรมสำหรับ Arduino

ในการเขียนโค้ด (sketch) สำหรับ Arduino สามารถแบ่งได้เป็น 4 ส่วนสำคัญคือ

1. การประกาศค่าคงที่และตัวแปรภายนอก
2. การสร้างฟังก์ชันขึ้นมาใช้งานใหม่ (เพื่อเรียกใช้งานใหม่)
3. การสร้างฟังก์ชัน setup()
4. การสร้างฟังก์ชัน loop()

ชนิดข้อมูลพื้นฐานสำหรับ Arduino C/C++

Byte ใช้สำหรับข้อมูลที่เป็นเลขจำนวนเต็มตั้ง 0 ถึง 255

Int ใช้สำหรับข้อมูลที่เป็นเลขจำนวนเต็มได้ตั้งแต่ -32768 ถึง +32767

Long ใช้สำหรับข้อมูลที่เป็นเลขจำนวนเต็มได้ตั้งแต่ –2,147,483,648 ถึง +2,147,489,647

Float ใช้สำหรับข้อมูลที่เป็นเลขทศนิยม เป็นค่าที่เป็นบวกหรือลบในช่วงที่กว้างกว่าชนิดข้อมูลแบบ byte และ int และเป็นตัวเลขทศนิยมได้ด้วยแต่มีความละเอียดเพียง 6-7 ตำแหน่งหลังจุดทศนิยมในเลขฐานสิบ

Boolean ใช้สำหรับข้อมูลที่เป็นค่าทางลอจิก true (จริง) หรือ false (เท็จ) เท่านั้น

คำสั่งพื้นฐานสำหรับ arduino ที่ควรรู้

pinMode( )	ใช้กำหนดทิศทางสัญญาณ(I/O direction) ของขาดิจิทัล
digitalread( )	ใช้อ่านค่าขาดิจิทัลที่ถูกกำหนดให้เป็นอินพุท
digitalWrite( )	ใช้เขียนค่า Low หรือ HIGH ให้ขาดิจิทัลที่ถูกกำหนดให้เป็นเอาท์พุท
analogWrite( )	ใช้สร้างสัญญาณ PWM เป็นเอาท์พุท
analogRead( )	ใช้อ่านค่าอนาล็อคอินพุท
analogReference( )	กำหนดระดับแรงดันอ้างอิงสำหรับการอ่านค่าจากขาอนาล็อคอินพุท
delay( )	รอให้ระยะเวลาผ่านไปตามระยะเวลากำหนด (มิลลิวินาที) ก่อนที่จะทำขั้นตอนต่อไป
delayMicroseconds ( )	รอให้ระยะเวลาผ่านไปตามระยะเวลาที่กำหนด (ไมโครวินาที)
randomSeed ( )	กำหนดค่าเริ่มต้นสำหรับการสร้างเลขแบบสุ่มเทียม
random ( )	ให้ค่าเป็นเลขสุ่มเทียม(Pseudo-Random Number)
millis ( )	บอกเวลาที่ผ่านไปในหน่วยเป็นมิลลิวินาทีนับตั้งแต่โปรแกรมเริ่มต้นทำงาน
min ( )	ให้ค่าน้อยที่สุดระหว่างตัวเลขสองค่าที่นำมาเปรียบเทียบ
max ( )	ให้ค่ามากที่สุดระหว่างตัวเลขสองค่าที่นำมาเปรียบเทียบ
abs ( )	ให้ค่าสัมบูรณ์ของตัวเลข
constrain ( )	ให้ค่าที่ไม่เกินช่วงที่กำหนด
map ( )	ให้ค่าที่ได้จากการย่อหรือขยายเชิงเส้นตามช่วงที่กำหนด
pow ( )	คำนวณเลขยกกำลัง (Power)

sqrt( )	คำนวณค่ารากที่สอง
sin( )	คำนวณค่า sin ในหน่วยเรเดียน (radian)
cos( )	คำนวณค่า cos ในหน่วยเรเดียน (radian)
tan( )	คำนวณค่า tan ในหน่วยเรเดียน (radian)

2.2 มอเตอร์ไฟฟ้า

มอเตอร์ไฟฟ้าเป็นอุปกรณ์ที่นิยมใช้กันอย่างแพร่หลายในงานอุตสาหกรรม เป็นอุปกรณ์ที่ใช้ควบคุมเครื่องจักรกล มอเตอร์มีหลายแบบหลายชนิดที่ใช้ให้เหมาะสมกับงาน ดังนั้นเราจึงต้องทราบถึงความหมายและ ชนิดของมอเตอร์ไฟฟ้า ตลอดถึงคุณสมบัติการใช้งานของมอเตอร์แต่ละชนิด เพื่อให้เกิดประสิทธิภาพสูงสุดในการใช้ งานของมอเตอร์นั้นๆ มอเตอร์มอเตอร์ไฟฟ้า (Motor) หมายถึงเครื่องกลไฟฟ้าชนิดหนึ่งที่เปลี่ยนแปลงพลังงานไฟฟ้ามาเป็น พลังงานกล มอเตอร์ไฟฟ้าที่ใช้พลังงานไฟฟ้าเปลี่ยนเป็นพลังงานกล มีทั้งพลังงานไฟฟ้ากระแสสลับและพลังงานไฟฟ้ากระแสตรง มอเตอร์ไฟฟ้าแบ่งออกตามชนิดของกระแสไฟฟ้า ได้ 2 ชนิดดังนี้

- มอเตอร์ไฟฟ้ากระแสสลับ(Alternating Current Motor)

  1. มอเตอร์ไฟฟ้ากระแสสลับชนิด 1 เฟส

     - สปลิทเฟสมอเตอร์(Split-Phase motor)
     - คาปาซิเตอร์มอเตอร์ (Capacitor motor)
     - รีพลัชชั่นมอเตอร์(Repulsion-type motor)
     - ยูนิเวอร์แซลมอเตอร์ (Universal motor)
     - เช็ดเดดโพลมอเตอร์ (Shaded-pole motor)

  2. มอเตอร์ไฟฟ้ากระแสสลับชนิด 2 เฟส

  3. มอเตอร์ไฟฟ้ากระแสสลับชนิด 3 เฟส

- มอเตอร์ไฟฟ้ากระแสตรง (Direct Current Motor) แบ่งออกเป็น 3 ชนิดได้แก่

  1. มอเตอร์แบบอนุกรมหรือเรียกว่าซีรีส์มอเตอร์ (Series Motor)

  2. มอเตอร์แบบขนานหรือเรียกว่าชันท์มอเตอร์ (Shunt Motor)

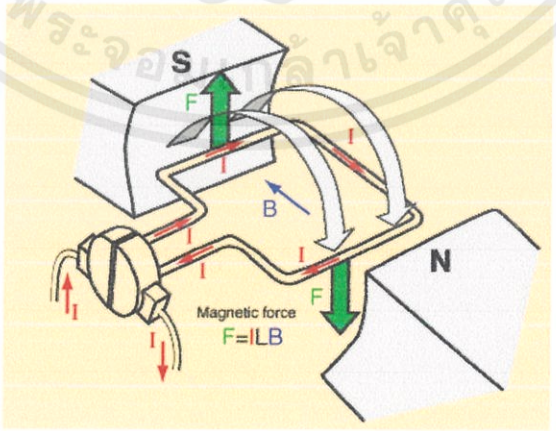
  3. มอเตอร์ไฟฟ้าแบบผสมหรือเรียกว่าคอมปาวด์มอเตอร์ (Compound Motor)

มอเตอร์ไฟฟ้ากระแสตรงเป็นต้นกำลังขับเคลื่อนที่สำคัญอย่างหนึ่งในโรงงานอุตสาหกรรม เพราะมีคุณสมบัติ ที่ดีเด่นในด้านการปรับความเร็วได้ ตั้งแต่ความเร็วต่ำสุดจนถึงสูงสุด นิยมใช้กันมากในโรงงานอุตสาหกรรม

2.2.1 หลักการของมอเตอร์กระแสไฟฟ้าตรง

หลักการของมอเตอร์ไฟฟ้ากระแสตรง (Motor Action) เมื่อมีแรงดันกระแสไฟฟ้าตรงเข้าไปในมอเตอร์ ส่วนหนึ่งจะเข้าแปรงถ่านผ่านคอมมิวเตเตอร์เข้าไปในขดลวดอาร์มาเจอร์สร้างสนามแม่เหล็กขึ้น และกระแสไฟฟ้า อีกส่วนหนึ่งจะไหลเข้าไปในขดลวดสนามแม่เหล็ก (Field coil) สร้างขั้วเหนือ-ใต้ขึ้น เกิดสนามแม่เหล็ก 2 สนาม ใน ขณะเดียวกัน ตามคุณสมบัติของเส้นแรงแม่เหล็กจะไม่ตัดกัน ทิศทางตรงข้ามจะหักล้างกันและทิศทางเดียวจะ เสริมแรงกัน ทำให้เกิดแรงบิดในตัวอาร์มาเจอร์ DC Motor ประกอบด้วย 2 ส่วนหลักๆ ได้แก่ ทุ่นอาร์มาเจอร์และสเตเตอร์ ทุ่นอาร์มาเจอร์ก็คือส่วนที่หมุน ส่วนสเตเตอร์คือส่วนที่เป็นขดลวดที่สร้างสนามแม่เหล็ก หลักการทำงานของ DC Motor คือการนำกระแสตรงมา ตัดผ่านสนามแม่เหล็ก ทำให้เกิดทอร์กขึ้นในทิศที่เหมาะสมและสร้างให้เกิดการหมุนของทุ่นอาร์มาเจอร์

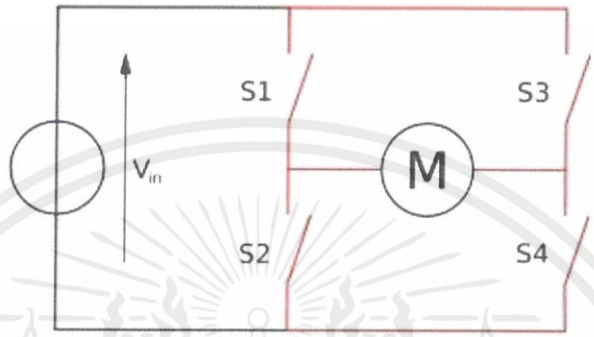
รูปที่ 2.8 ส่วนประกอบของมอเตอร์ไฟฟ้ากระแสตรง



รูปที่ 2.9 หลักการทำงานของมอเตอร์ไฟฟ้ากระแสตรง

2.2.2 การควบคุมทิศทาง และความเร็วรอบของมอเตอร์

การควบคุมทิศทาง และความเร็วรอบของมอเตอร์ ไม่ว่าจะเป็นการขับเคลื่อนสายพานหรือล้อของ หุ่นยนต์ไม่ให้หมุนแบบอิสระและสามารถควบคุมได้ ดังนั้นจึงมีวงจรที่ใช้ในการควบคุมมอเตอร์ขึ้นมาแบบที่นิยม ใช้กันเรียกว่าวงจร “H-Bridge”

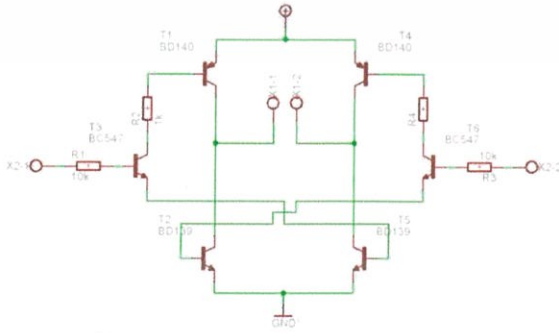


รูปที่ 2.10 การควบคุมมอเตอร์ H-Bridge

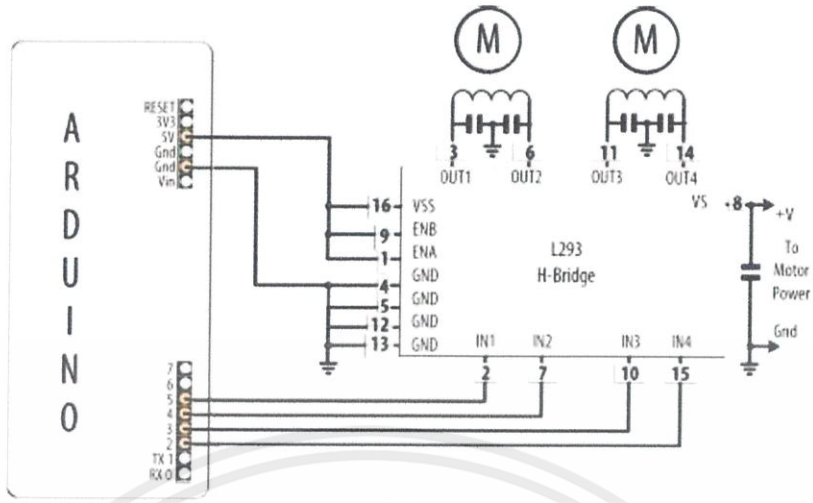
วงจรนี้ทำหน้าที่ได้ทั้งคุมทิศทางและความเร็วของมอเตอร์ โดยเริ่มจากการคุมทิศทางการหมุน โดยปกติ หากต้องการกลับทิศการหมุนของมอเตอร์กระแสตรง วิธีหนึ่งที่ทำได้คือกลับทิศแหล่งจ่ายวงจร H-Bridge ด้านบน

- หากต้องการให้หมุนตามเข็ม (Clockwise :CW) ให้ S1 และ S4 ปิดวงจร และให้ S2 และ S3 เปิดวงจร
- หากต้องการให้หมุนทวนเข็ม (Counter Clockwise :CCW) ให้ S2 และ S3 ปิดวงจร และให้ S1 และ S4 เปิดวงจร

สังเกตว่าสวิตช์จะทำงานเป็นคู่ คู่แรกทำงาน คู่สองต้องเปิดวงจร และในทางตรงข้ามก็คือคู่สองทำงาน คู่ แรกต้องเปิดวงจร ต่อมามีการทำให้การเปิดปิดเป็นแบบที่ง่ายกว่าเดิม โดยใช้อุปกรณ์สารกึงตัวนำเช่น MOSFET หรือ IGBT หรืออื่นๆ แล้วแต่ความเหมาะสม เช่นขนาดกระแสแรงดันที่ต้องการควบคุม



รูปที่ 2.11 Schematic ของวงจร H-Bridge

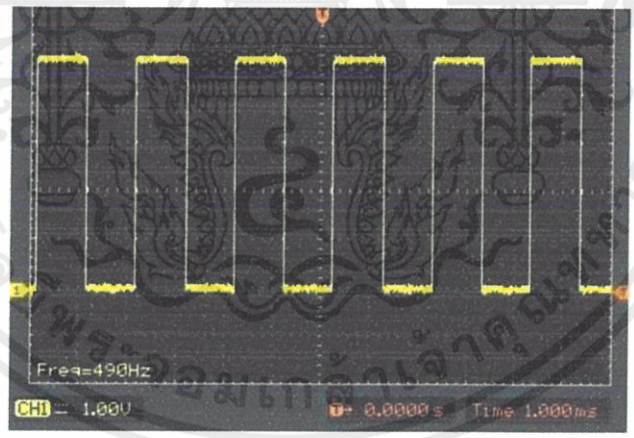


รูปที่ 2.12 การต่อบอร์ด Arduino ควบคุม H-Bridge

2.3 การควบคุมอุปกรณ์ต่างๆด้วย Arduino

2.3.1 การควบคุมดีซีมอเตอร์ด้วย Arduino

การควบคุมความเร็วรอบของ DC Motor ด้วยเทคนิค Pulse Width Modulation (PWM)



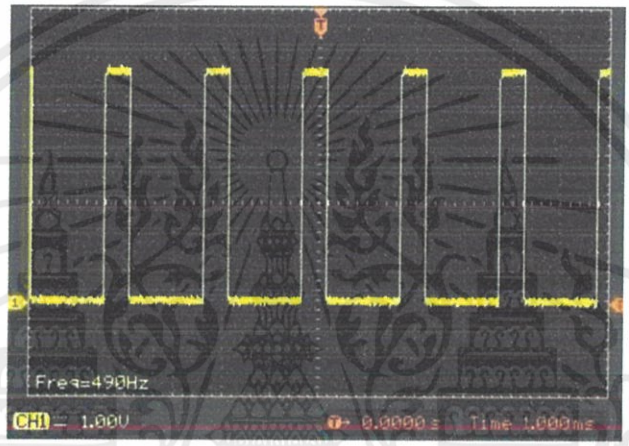
รูปที่ 2.13 ออสซิโลสโคปแสดงสัญญาณ PWM

PWM (Pulse Width Modulation) คือ การมอดูเลตสัญญาณให้มีความกว้างตามสัดส่วนที่กำหนดบน ความถี่ Carrier ที่ต้องการใช้งาน ซึ่งโดยปกติแล้วจะใช้ประโยชน์ในการควบคุมการเปิดปิดของวงจรอิเล็กทรอนิกส์ กำลัง เช่น วงจรbuck วงจรboost วงจรbuck-boost เป็นต้น นอกจากนี้ยังใช้สามารถใช้ประกอบกับวงจร H - Bridge เพื่อควบคุมความเร็วรอบของมอเตอร์ หรือว่า วงจรพื้นฐานเช่นต้องการจะหรี่หลอด LED ก็ได้อีกด้วย พารามิเตอร์ที่ใช้ระบุรูปร่างหน้าตาของ PWM

ที่สำคัญมี 2 ค่าด้วยกันคือ ความถี่ของคลื่นพาหะ (Carrier Frequency) และ อัตราส่วนหน้าที่ (Duty Ratio)

Carrier Frequency ใน Arduino มีความถี่ประมาณ 490 Hz ในกรณีที่ใช้ Library ปกติใน Arduino IDE และยังสามารถปรับให้มีความถี่สูงขึ้นเป็นค่าอื่นๆ เช่น 31.25 kHz ได้อีกด้วย

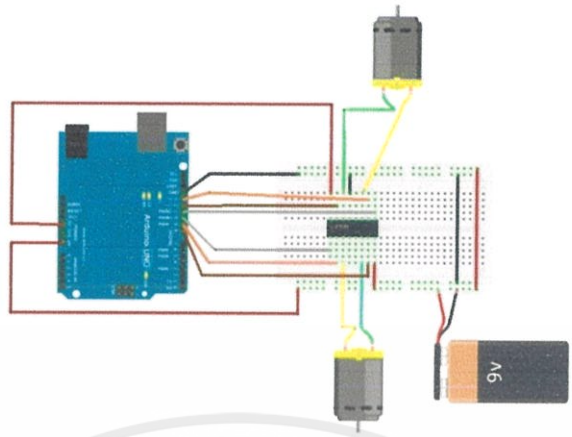
Duty Ratio คือสัดส่วนของเวลาที่จ่ายแรงดันต่อคาบของ Carrier Frequency ซึ่งกรณีที่เห็นในรูปที่ 17 คือเวลาที่จ่ายแรงดัน 1.02 มิลลิวินาทีต่อคาบ 2.04 มิลลิวินาที ซึ่งจะได้ Duty Ratio = 50%



รูปที่ 2.14 สัญญา PWM ความถี่ Carrier ที่ 490 Hz

พิจารณารูปที่ 2.16 กำหนดให้PWM Port 3 ในการกำเนิดสัญญาณนี้โดยใช้คำสั่ง analogWriteซึ่งหมายถึงการเขียนให้ออก Port 3 โดยมีสัดส่วน Duty Ratio คือ 128/256 หรือ 50% นั่นเอง การที่ใช้ คำสั่ง analogWrite แต่สัญญาณไปออกขา Digital(PWM) ก็เป็นเพราะ Arduino Board อย่าง UNO ไม่มี Analog Output channel มีแต่ PWM Output ที่ Pin ด้าน Digital

สัญญา PWM ถึงแม้จะมีความถี่ Carrier ที่ 490 Hz สามารถใช้ในการควบคุมมอเตอร์กระแสตรงโดยใช้ วงจร H-Bridge โดย PWM ใช้ได้กับ IC เบอร์ L298P และ L293 หรือ Motor Shield นอกจากนี้ถ้าจะต่อวงจร สร้าง H-Bridge กรณีที่มอเตอร์มีขนาดใหญ่กว่า 2 A โดยต่อเข้ากับขา INPUT ของ IC ที่ใช้และต่อกราวด์ของ Arduino กับแหล่งจ่ายแรงดันหลักของมอเตอร์



รูปที่ 2.15 วงจร H-Bridge โดย PWM ใช้กับ IC เบอร์ L293D

อีกหนึ่งรูปแบบที่มีการประยุกต์ใช้งานคือวงจร Switching Power Supply คือการปรับให้ Arduino กำเนิดสัญญา PWM แบบความถี่สูงๆ เช่น 31.25 kHz การใช้งาน Arduino ในส่วนนี้ต้องเข้าไปแก้ไข Register จำนวน 4 ตัวใน Atmel 328 ที่ใช้เป็นตัวไมโครคอนโทรลเลอร์ของ Arduino UNO ใช้สำหรับการทำ งานกับวงจร Power Electronics เพราะความถี่ 490 Hz ไม่เพียงพอสำหรับการทำ Switching Power Supply แบบต่างๆ ก็ เพราะว่าขนาดของ Capacitor กับ Inductor ในวงจรเพื่อใช้ Filter ความถี่จะต้องมีขนาดใหญ่มาก

รูปที่ 2.16 สัญญาณจากออสซิลโลสโคปมีความถี่ 31.25 KHz

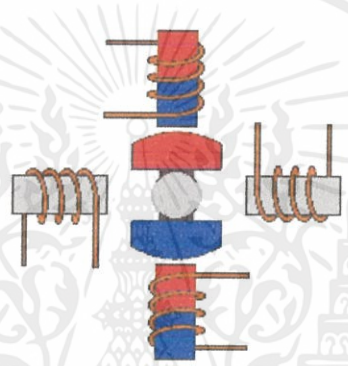
2.3.2 การควบคุมสเตปมอเตอร์ด้วย Arduino

ในหัวข้อที่ผ่านมาได้นำเสนอการควบคุม DC Motor สาเหตุที่ต้องศึกษา DC Motor ก่อนก็เพราะว่าการทำงานของมอเตอร์ชนิดอื่นๆ เช่น Stepper Motor จะมีพื้นฐานมาจาก DC Motor ในบางด้านแต่ Stepper Motor จะมีความซับซ้อนกว่าทั้งหลักการทำงานและการควบคุม Stepper

Motor หรือ Stepping Motor มีข้อดีที่สำคัญคือการควบคุมตำแหน่งของการหมุนได้อย่าง แม่นยำ โดยไม่ต้องใช้การควบคุมแบบป้อนกลับ (Feedback Control) ด้วยเหตุนี้ จึงเป็นที่นิยมใช้ในอุปกรณ์ที่ ต้องการควบคุมตำแหน่งและมุมอย่างแม่นยำ เช่น ปริ้นเตอร์ สแกนเนอร์เครื่องเล่นแผ่นดิสค์เป็นต้น

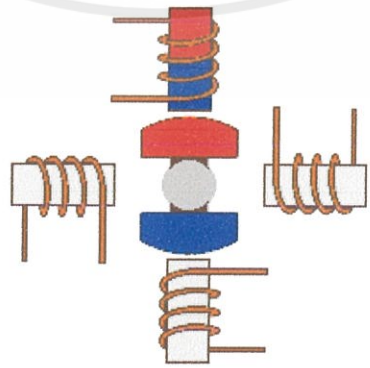
2.3.2.1 หลักการของการทำงานของ Stepper Motor

หลักการของการทำงานของ Stepper Motor คือการบังคับให้แม่เหล็กถาวรบนแกนโรเตอร์หมุนไปตาม ทิศการบังคับของขดลวดที่ติดตั้งบนสเตเตอร์จะซับซ้อนกว่า DC Motor ตรงที่การบังคับให้หมุนนั้นไม่ได้เป็นแค่การ ใส่แรงดันคงที่ไปที่ขั้วบวกลบเท่านั้น ต้องใส่แรงดันให้ถูกต้องจากจังหวะที่ควรจะเป็นดังรูปที่ 11 ถึงจะหมุนได้



รูปที่ 2.17 การจ่ายกระแสไฟฟ้าเข้าไปที่ขดลวดทีละขด

จะเห็นว่ามีขดลวดที่ควบคุมการหมุน โดยแต่ละขดห่างกัน 90 องศา การหมุนก็จะทำโดยการจ่ายกระแส เข้าไปที่ขดลวดทีละขด เพื่อทำให้เกิดสนามแม่เหล็ก ซึ่งจะไปดูดให้แม่เหล็กถาวะที่อยู่บนโรเตอร์เคลื่อนที่ โดยทิศของการหมุนก็จะขึ้นกับลำดับการจ่ายกระแสเข้าไปที่ขดลวด โดยการบังคับในลักษณะนี้เรียกว่า Single coil excitation หรือการกระตุ้นทีละขดลวด โดยจะมีการกระตุ้นหรือการจ่ายกระแสเข้าขดลวดอยู่ 4 จังหวะต่อการ หมุน 1 รอบ อีกรูปแบบหนึ่งจะซับซ้อนกว่าแต่จะให้ทอร์กมากกว่า คือการป้อนแบบ Full Step Drive หรือการป้อน แบบทีละ 2 ขดลวด



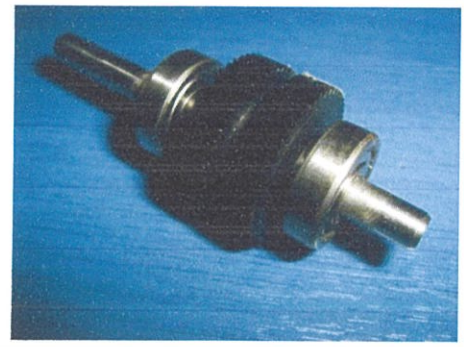
รูปที่ 2.18 การจ่ายกระแสไฟฟ้าเข้าไปที่ขดลวดทีละสองขด

ถ้าต้องการควบคุมให้มีความละเอียดมากขึ้นอีก สามารถทำได้โดยสามารถคุมให้มอเตอร์หมุนได้ละเอียดขึ้น จาก 90 องศา เหลือ 45 องศา โดยที่ไม่ต้องปรับเปลี่ยนตัวฮาร์ดแวร์ใดๆ เป็นเพียงการเปลี่ยนวิธีการจ่ายกระแสเข้าขดลวดเท่านั้น แต่ Stepper Motor ที่ใช้จริงมีการพัฒนาต่อ โดยเพิ่มจำนวนขดลวดและปรับให้แม่เหล็กถาวรมีซี่ (Teeth) ซึ่งทำหน้าที่เป็นจำนวนขั้วของแม่เหล็กมากขึ้น การเพิ่มจำนวนซี่โดยแต่ละซี่เป็นขั้วเหนือและขั้วใต้สลับกันนั้น สามารถทำได้โดยเอาเหล็กที่มีสภาพความนำแม่เหล็กมาขึ้นรูปเป็นจานที่มีซี่อยู่รอบๆ จากนั้นขั้นตอนการควบคุมก็มี หลักการเดียวกับ Stepper Motor ที่หมุนได้ทีละ 90 องศา

เนื่องจาก Stepper Motor ที่ใช้กันจริง ๆ มีซี่และขดลวดมาก จึงทำให้สามารถควบคุมการหมุนได้ละเอียด มาก ๆ โดยการควบคุม 1 Cycle จะทำให้มอเตอร์หมุนไป 0.9 - 5 องศา แล้วแต่เทคนิคที่ใช้ในการควบคุมการหมุน

รูปที่ 2.19 การเพิ่มจำนวนขดลวดและปรับให้แม่เหล็กถาวรมีซี่ (Teeth)

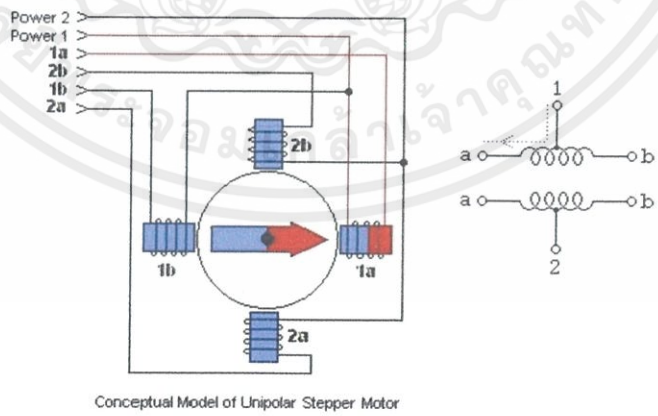
สังเกตเห็นว่ามีขดลวดอยู่ 6 ขดลวด การทำงานจะต้องควบคุมเป็นขั้น ให้ครบ 75 ขั้นจึงหมุนได้ครบ 1 รอบ หรือ 5 องศาต่อขั้นการเรียงขดลวด 6 ขดก็ไม่ได้วางห่างกัน 60 องศาแบบตรงๆ ถ้าทำแบบนั้นมอเตอร์จะไม่ หมุน แต่ต้องเรียงให้คู่แรกคือขดลวดบนสุดและล่างสุดวางห่างจากคู่ที่สอง และสามเป็นมุม 60 + 5 = 65 องศา เพื่อให้มันหมุนทีละ 5 องศา มุมต่างนี้มีผลต่อการควบคุมและการเขียนโปรแกรม



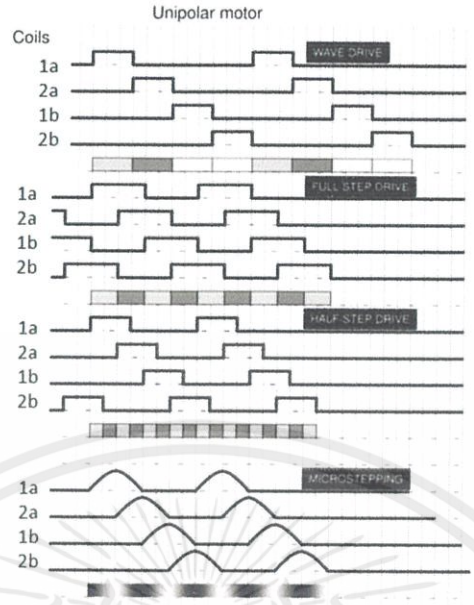
รูปที่ 2.20 ลักษณะของแกนโรเตอร์แบบแม่เหล็กถาวรมีซี่ (Teeth)

อุปกรณ์จริงอย่างรูปที่ 2.20 จะเห็นว่ามี 2 จาน แต่ละจานมี 50 ซี่ ทำให้รวมมีซี่ที่สร้างเป็นขั้วเหนือและใต้ รวม 100 ซี่ สำหรับ Stepper Motor ที่มีใช้กันอยู่จะมี 2 แบบ ขึ้นกับการต่อขดลวดภายในมอเตอร์ ได้แก่แบบ Unipolar และแบบ Bipolar แบบที่นิยมใช้คือแบบ Unipolar เพราะว่าการควบคุมง่ายกว่า เนื่องจากไม่ต้องกลับ ทิศของกระแสที่ป้อนเข้าไปที่ขดลวด แต่ถ้าเป็นแบบ Bipolar ก็ทำได้ยาก แต่สามารถประยุกต์ใช้งานได้มากกว่า ใน Stepper Motor จริงๆนั้นขดลวดจะทำงานเป็น 2 ชุด (ถึงแม้จะมีขดลวดมากกว่า 4 ก็ตาม) ถ้าตามรูปที่ 16 คือ a กับ b โดยมี 1 กับ 2 เป็นแหล่งจ่ายไปบวกลบ และ 1a 1b 2a 2b เป็นตัวกำหนดทิศทางการไหลของกระแส ผ่านขดลวดทั้งสอง

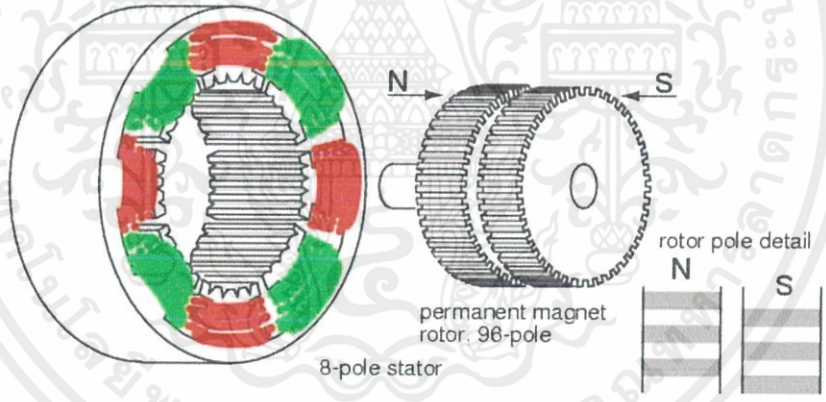
การทำให้ Stepper Motor หมุนแบบง่ายที่สุดก็คือ การจ่ายไฟเข้าไปทีละขดตามลำดับ 1a 2a 1b 2b เรียกวิธีการควบคุมแบบนี้ว่า "Wave Drive" การทำให้ Stepper Motor หมุนแบบ Full Step Drive ทำได้โดยให้ กระแสไหลผ่านขดลวดทีละ 2 ขด ทำให้ได้ทอร์กออกมามากกว่าแบบแรกประมาณ 2 เท่า ซึ่งเป็นวิธีที่ library ของ Arduino เลือกใช้



รูปที่ 2.21 Unipolar Stepper motor



รูปที่ 2.22 การป้อนกระแสไฟฟ้าให้กับ Stepper motor แบบต่าง ๆ



รูปที่ 2.23 การวางขั้วแม่เหล็กถาวรใน Stepper motor

ข้อมูลที่สำคัญอีกอย่างที่ควรทราบคือการทำงานของ Stepper Motor นั้นใช้กระแสที่สูงมากกว่าที่ Arduino สามารถขับได้ดังนั้นจึงต้องใช้วงจรขยายสัญญาณเรียกว่า "วงจร Darlington" เป็น IC สำเร็จรูปเรียกว่า ULN2003AN หรือรุ่นใกล้เคียง

2.3.3 การควบคุมเซอร์โวมอเตอร์ด้วย Arduino

2.3.3.1 ความรู้เบื้องต้นเกี่ยวกับเซอร์โวมอเตอร์

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

Feedback Control คือระบบควบคุมที่มีการนำค่าเอาต์พุตของระบบนำมาเปรียบเทียบกับค่าอินพุต เพื่อ ควบคุมและปรับแต่งให้ค่าเอาต์พุตของระบบให้มีค่าเท่ากับหรือใกล้เคียงกับค่าอินพุต

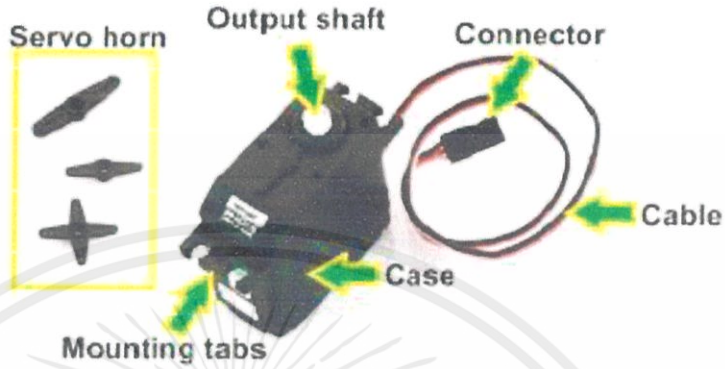
ส่วนประกอบภายนอก RC Servo Motor

รูปที่ 2.24 แสดงลักษณะทั่วไปของ servo motor

2.3.3.2 ส่วนประกอบของ servo motor

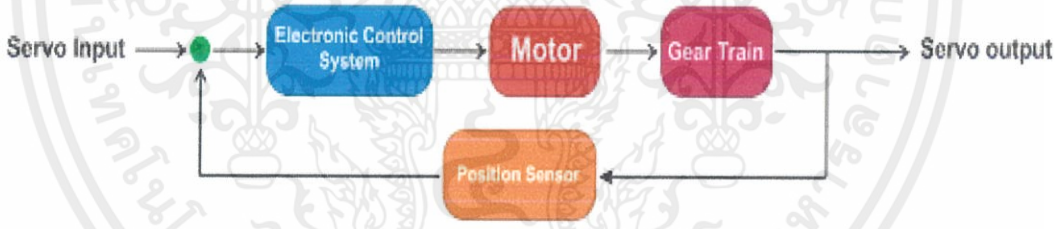
- Case ตัวถัง หรือ กรอบของตัว Servo Motor
- Mounting Tab ส่วนจับยึดตัว Servo กับชิ้นงาน
- Output Shaft เพลาส่งกำลัง
- Servo Horns ส่วนเชื่อมต่อกับ Output shaft เพื่อสร้างกลไกล
- Cable สายเชื่อมต่อเพื่อจ่ายไฟฟ้า และ ควบคุม Servo Motor จะประกอบด้วยสายไฟ 3 เส้น และ ใน RC Servo Motor จะมีสีของสายแตกต่างกันไปดังนี้
- สายสีแดง คือ ไฟเลี้ยง (4.8-6V)
- สายสีดำ หรือ น้ำตาล คือ กราวด์

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



รูปที่ 2.25 แสดงส่วนประกอบของ servo motor

2.3.3.3 บล็อกไดอะแกรมของ servo motor



รูปที่ 2.26 แสดงบล็อกไดอะแกรมการทำงานของ servo motor

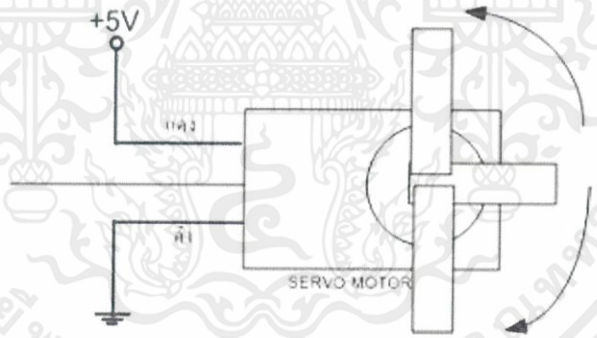
- Motor เป็นส่วนของตัวมอเตอร์
- Gear Train หรือ Gearbox เป็นชุดเกียร์ทดแรง
- Position Sensor เป็นเซ็นเซอร์ตรวจจับตำแหน่งเพื่อหาค่าองศาในการหมุน
- Electronic Control System เป็นส่วนที่ควบคุมและประมวลผล

2.3.3.4 หลักการทำงานของ RC servo motor

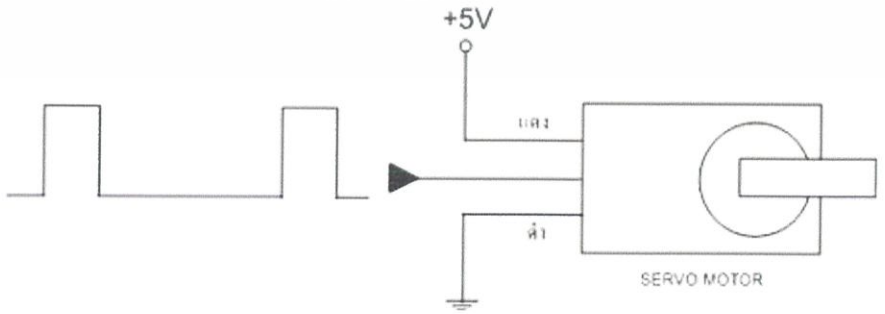
เมื่อจ่ายสัญญาณพัลส์เข้ามายัง RC Servo Motor ส่วนวงจรควบคุม (Electronic Control System) ภายใน Servo จะทำการอ่านและประมวลผลค่าความกว้างของสัญญาณพัลส์ที่ส่งเข้ามาเพื่อแปลค่าเป็นตำแหน่ง องศาที่ต้องการให้ Motor หมุนเคลื่อนที่ไปยังตำแหน่งนั้น แล้วส่งคำสั่งไปทำการควบคุมให้ Motor หมุนไปยัง ตำแหน่งที่ต้องการ โดยมี Position Sensor เป็นตัวเซ็นเซอร์คอยวัดค่ามุมที่ Motor กำลังหมุนเป็น Feedback กลับมาให้วงจรควบคุมเปรียบเทียบกับค่าอินพุตเพื่อควบคุมให้ได้ตำแหน่งที่ต้องการอย่างถูกต้องแม่นยำ สัญญาณ RC ในรูปแบบ PWM ตัว RC Servo Motor ออกแบบมาใช้สำหรับรับคำสั่งจาก Remote Control ที่ใช้ควบคุมของเล่นด้วย สัญญาณวิทยุต่างๆ เช่นเครื่องบินบังคับ รถบังบังคับ เรือบังคับ เป็นต้น ซึ่ง Remote จำพวกนี้ที่ภาครับจะแปลงความถี่วิทยุออกมาในรูปแบบสัญญาณ PWM (Pulse Width Modulation)

2.3.3.5 การควบคุมการทำงานของ servo motor

เซอร์โวมอเตอร์ แบบทั่วไปจะมีมุมในการหมุนจำกัดที่ 180 องศาเท่านั้น แต่จะมีเซอร์โวแบบที่มีการดัดแปลงวงจรภายใน ซึ่งสามารถหมุนได้ถึง 360 องศา

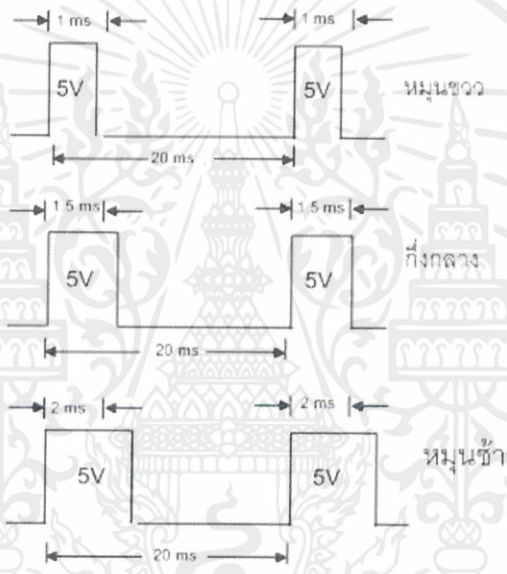


รูปที่ 2.27 แสดงมุมจำกัดในการหมุนของ servo motor



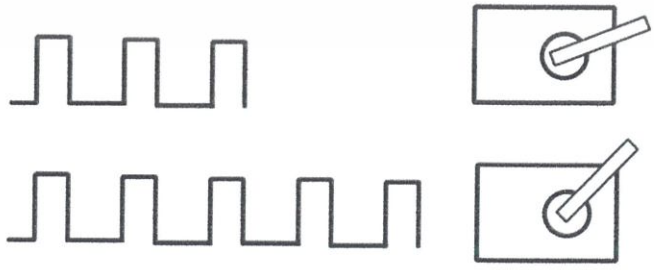
รูปที่ 2.28 แสดงการต่อใช้งาน servo motor

ในการควบคุมเซอร์โวมอเตอร์นั้น ทำได้โดยอาศัยความกว้างของพัลส์ที่ทำการป้อนให้เซอร์โวมอเตอร์ โดยสัญญาณพัลส์นี้จะเป็นสัญญาณ TTL โดยระดับสูงหรือ 1 จะมีแรงดัน 5 VDC ระดับต่ำหรือ 0 จะมีแรงดัน 0 VDC เซอร์โวมอเตอร์หมุนไปทางขวา หรือ ตามเข็มนาฬิกา จะต้องสร้างสัญญาณพัลส์ ขนาด 1 ms เมื่อเซอร์โวมอเตอร์หมุนไปทางขวาสุดแล้วมอเตอร์จะหยุดหมุนเอง ถ้าป้อนพัลส์เข้าไปอีกจะเซอร์โวมอเตอร์จะไม่ทำงาน เซอร์โวมอเตอร์หมุนไปทางซ้าย หรือ ทวนเข็มนาฬิกา จะต้องสร้างสัญญาณพัลส์ ขนาด 2 ms เมื่อเซอร์โวมอเตอร์หมุนไปทางซ้ายสุดแล้วมอเตอร์จะหยุดหมุนเอง ถ้าป้อนพัลส์เข้าไปอีกจะเซอร์โวมอเตอร์จะไม่ทำงาน เซอร์โวมอเตอร์ไปตำหแหน่งกึ่งกลาง หากต้องการให้เซอร์โวมอเตอร์หนุนไปทางซ้าย หรือ ทวนเข็มนาฬิกา จะต้องสร้างสัญญาณพัลส์ ขนาด 1.5 ms



รูปที่ 2.29 แสดงความกว้างของสัญญาณที่มีผลต่อการหมุนของมอเตอร์

เซอร์โวมอเตอร์แต่ละตัวจะมีการทำงานของความกว้างของพัลส์ไม่เท่ากัน ค่าดังกล่าวเป็นค่าประมาณเท่านั้น ดังนั้นถ้าต้องการค่าที่ถูกต้อง ต้องดูที่คู่มือด้วยทุกครั้ง และจำนวนลูกสัญญาณพัลส์จะมีผลต่อองศาในการหมุนด้วยเช่นเดียวกัน



รูปที่ 2.30 จำนวนสัญญาณพลัลส์ที่มีผลต่อองศา

2.4 ปั๊มน้ำ

ปั๊มน้ำ (water pump) คือ อุปกรณ์สำหรับส่งน้ำหรือถ่ายเทของเหลวจากที่หนึ่งไปยังอีกที่หนึ่ง หรือหมุนเวียนน้ำหรือของเหลวให้ผสมกันในบริเวณที่จำกัด การทำงานของปั๊มน้ำมีทั้งแบบที่ใช้มอเตอร์ไฟฟ้าและแบบที่ใช้เครื่องยนต์เพื่อทำหน้าที่หมุนส่งกำลังให้เครื่องสูบทำงานเพิ่มแรงดันและส่งน้ำไปตามท่โดยแบบที่ใช้ส่วนใหญ่เป็นแบบไฟฟ้า โดยสามารถแบ่งออกได้เป็น 2 ประเภทคือปั๊มน้ำดีซี(DC)กับปั๊มน้ำเอซี(AC)

2.4.1 ปั๊มน้ำดีซี

ปั๊มน้ำดีซีคือปั๊มน้ำที่ใช้ไฟฟ้ากระแสตรงในการขับเคลื่อนมอเตอร์ซึ่งปั๊มน้ำกระแสตรงที่นิยมใช้ได้แก่

1. ปั๊มน้ำดีซีแบบจุ่ม(DC-brushless Submersible Pump) คือการใช้งานจะต้องจุ่มตัวปั๊มลงไปในน้ำเพื่อจะดูดน้ำไปยังอีกที่หนึ่ง ซึ่งอัตราการไหลของน้ำ(ลิตรต่อนาที) ความสูงสูงสุดที่น้ำจะส่งไป(เมตร)และระยะทางน้ำที่ดูดไปถึงปลายทาง(เมตร)
2. ปั๊มน้ำดีซีบาดาล(DC Submersible pump) คือการใช้งานจะต้องจุ่มปั๊มลงไปในน้ำเหมือนกันกับปั๊มจุ่มชนิดแรก แต่มีข้อดีที่เพิ่มขึ้นมาคือสามารถต่อวงจรตัวปั๊มโดยตรงได้เลยโดยที่ไม่ทำให้ตัวปั๊มชนิดนี้เสียหายเพราะโครงสร้างที่ออกแบบมาสามารถรองรับการใช้งานที่แรงดันกระแสตรงที่มีการเปลี่ยนแปลงได้
3. ปั๊มแรงดัน(DC pressure pump) คือปั๊มน้ำที่จะเพิ่มแรงดันน้ำปลายทางให้แรงขึ้นมีหน่วยแรงดันเป็นบาร์(bars) นิยมใช้กับการเพิ่มความแรงของน้ำเช่นล้างรถ ระบบสปริงเกิลพ่นน้ำ หรือสามารถนำไปประยุกต์ใช้การสูบน้ำขึ้นที่สูงก็ได้เช่นกันเพราะแรงดันน้ำปลายทางที่สูบจากปั๊มเพิ่มขึ้น

รูปที่ 2.31 แสดงภาพปั๊มน้ำ

2.5 รีเลย์

รีเลย์(relay) คือ อุปกรณ์อิเล็กทรอนิกส์ที่ทำหน้าที่เป็นสวิตช์ตัด-ต่อวงจร โดยใช้แม่เหล็กไฟฟ้า และการที่จะให้รีเลย์ทำงานก็ต้องจ่ายไฟให้อุปกรณ์ เมื่อรีเลย์ได้รับการจ่ายไฟ จะทำให้หน้าสัมผัสติดกัน กลายเป็นวงจรปิด และตรงข้ามทันทีที่ไม่ได้จ่ายไฟให้รีเลย์ ก็จะกลายเป็นวงจรเปิด ไฟที่เราใช้ป้อนให้กับตัวรีเลย์ก็จะเป็นไฟที่มาจากเพาเวอร์ของอุปกรณ์ไฟฟ้า ดังนั้นทันทีที่เปิดเครื่องก็จะทำให้รีเลย์ทำงาน

2.5.1 ส่วนประกอบของรีเลย์

รีเลย์ประกอบด้วยส่วนสำคัญ 2 ส่วนหลัก

1. ส่วนของขดลวด (coil) เหนี่ยวนำกระแสต่ำ ทำหน้าที่สร้างสนามแม่เหล็กไฟฟ้าให้แกนโลหะไปกระทุ้งให้หน้าสัมผัสต่อกัน ทำงานโดยการรับแรงดันจากภายนอกต่อคร่อมที่ขดลวดเหนี่ยวนำนี้ เมื่อขดลวดได้รับแรงดัน (ค่าแรงดันที่รีเลย์ต้องการขึ้นกับชนิดและรุ่นตามที่ผู้ผลิตกำหนด) จะเกิดสนามแม่เหล็กไฟฟ้าทำให้แกนโลหะด้านในไปกระทุ้งให้แผ่นหน้าสัมผัสต่อกัน

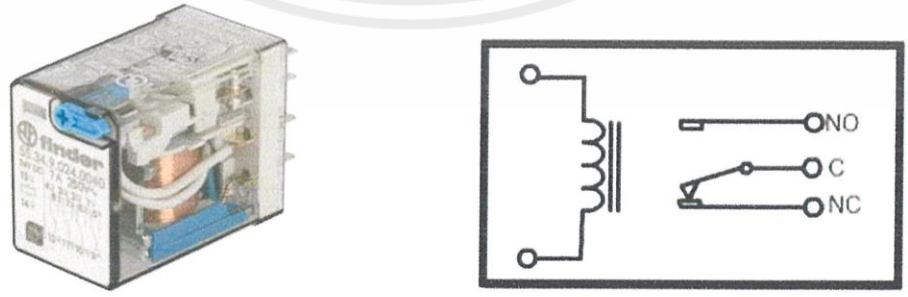
2. ส่วนของหน้าสัมผัส (contact) ทำหน้าที่เหมือนสวิตช์จ่ายกระแสไฟให้กับอุปกรณ์ที่เราต้องการ

2.5.2 จุดต่อใช้งานมาตรฐาน

จุดต่อ NC ย่อมาจาก normal close หมายความว่าปกติดปิด หรือ หากยังไม่จ่ายไฟให้ขดลวดเหนี่ยวนำหน้าสัมผัสจะติดกัน โดยทั่วไปเรามักต่อจุดนี้เข้ากับอุปกรณ์หรือเครื่องใช้ไฟฟ้าที่ต้องการให้ทำงานตลอดเวลา

จุดต่อ NO ย่อมาจาก normal open หมายความว่าปกติเปิด หรือหากยังไม่จ่ายไฟให้ขดลวดเหนี่ยวนำหน้าสัมผัสจะไม่ติดกัน โดยทั่วไปเรามักต่อจุดนี้เข้ากับอุปกรณ์หรือเครื่องใช้ไฟฟ้าที่ต้องการควบคุมการเปิดปิดเช่นโคมไฟสนามหนือหน้าบ้าน

จุดต่อ C ย่อมากจาก common คือจุดร่วมที่ต่อมาจากแหล่งจ่ายไฟ



รูปที่2.32 แสดงสัญลักษณ์ของรีเลย์

2.5.3 ประเภทของรีเลย์

แบ่งออกตามลักษณะการใช้งานได้เป็น 2 ประเภทคือ

1. รีเลย์กำลัง (power relay) หรือมักเรียกกันว่าคอนแทกเตอร์ (Contactor or Magnetic contactor)ใช้ในการควบคุมไฟฟ้ากำลัง มีขนาดใหญ่กว่ารีเลย์ธรรมดา

2. รีเลย์ควบคุม (control Relay) มีขนาดเล็กกำลังไฟฟ้าต่ำ ใช้ในวงจรควบคุมทั่วไปที่มีกำลังไฟฟ้าไม่มากนัก หรือเพื่อการควบคุมรีเลย์หรือคอนแทกเตอร์ขนาดใหญ่

2.5.4 ชนิดของรีเลย์

การแบ่งชนิดของรีเลย์สามารถแบ่งได้ 11 แบบ คือ ชนิดของรีเลย์แบ่งตามลักษณะของคอยล์ หรือ แบ่งตามลักษณะการใช้งาน (Application) ได้แก่รีเลย์ดังต่อไปนี้

1. รีเลย์กระแส (Current relay) คือ รีเลย์ที่ทำงานโดยใช้กระแสมีทั้งชนิดกระแสขาด (Under-current) และกระแสเกิน (Over current)

2. รีเลย์แรงดัน (Voltage relay) คือ รีเลย์ ที่ทำงานโดยใช้แรงดันมีทั้งชนิดแรงดันขาด (Under-voltage) และ แรงดันเกิน (Over voltage)

3. รีเลย์ช่วย (Auxiliary relay) คือ รีเลย์ที่เวลาใช้งานจะต้องประกอบเข้ากับรีเลย์ชนิดอื่น จึงจะทำงานได้

4. รีเลย์กำลัง (Power relay) คือ รีเลย์ที่รวมเอาคุณสมบัติของรีเลย์กระแส และรีเลย์แรงดันเข้าด้วยกัน

5. รีเลย์เวลา (Time relay) คือ รีเลย์ที่ทำงานโดยมีเวลาเข้ามาเกี่ยวข้องด้วย ซึ่งมีอยู่ด้วยกัน 4 แบบ คือ

- รีเลย์กระแสเกินชนิดเวลาผกผันกับกระแส (Inverse time over current relay) คือ รีเลย์ ที่มีเวลาทำงานเป็นส่วนกลับกับกระแส

- รีเลย์กระแสเกินชนิดทำงานทันที (Instantaneous over current relay) คือรีเลย์ที่ทำงานทันทีทันใดเมื่อมีกระแสไหลผ่านเกินกว่าที่กำหนดที่ตั้งไว้

- รีเลย์แบบดิฟฟินิตไทม์เล็ก (Definite time lag relay) คือ รีเลย์ ที่มีเวลาการทำงานไม่ขึ้นอยู่กับความมากน้อยของกระแสหรือค่าไฟฟ้าอื่นๆ ที่ทำให้เกิดงานขึ้น

- รีเลย์แบบอินเวอสดิฟฟินิตมินิมั่มไทม์เล็ก (Inverse definite time lag relay) คือ รีเลย์ ที่ทำงานโดยรวมเอาคุณสมบัติของเวลาผกผันกับกระแส (Inverse time) และ แบบดิฟฟินิตไทม์แล็ก (Definite time lag relay) เข้าด้วยกัน

6. รีเลย์กระแสต่าง (Differential relay) คือ รีเลย์ที่ทำงานโดยอาศัยผลต่างของกระแส

7. รีเลย์มีทิศ (Directional relay) คือรีเลย์ที่ทำงานเมื่อมีกระแสไหลผิดทิศทาง มีแบบรีเลย์กำลังมีทิศ (Directional power relay) และรีเลย์กระแสมีทิศ (Directional current relay)

8. รีเลย์ระยะทาง (Distance relay) คือ รีเลย์ระยะทางมีแบบต่างๆ ดังนี้

- รีแอกแตนซ์รีเลย์ (Reactance relay)
- อิมพีแดนซ์รีเลย์ (Impedance relay)
- โมห์รีเลย์ (Mho relay)
- โอห์มรีเลย์ (Ohm relay)
- โพลาไรซ์โมห์รีเลย์ (Polaized mho relay)
- ออฟเซทโมห์รีเลย์ (Off set mho relay)

9. รีเลย์อุณหภูมิ (Temperature relay) คือ รีเลย์ที่ทำงานตามอุณหภูมิที่ตั้งไว้

10. รีเลย์ความถี่ (Frequency relay) คือ รีเลย์ที่ทำงานเมื่อความถี่ของระบบต่ำกว่าหรือมากกว่าที่ตั้งไว้

11. บูคโฮลซ์รีเลย์ (Buchholz 's relay) คือรีเลย์ที่ทำงานด้วยก๊าซ ใช้กับหม้อแปลงที่แช่อยู่ในน้ำมันเมื่อเกิด ฟอลต์ ขึ้นภายในหม้อแปลง จะทำให้น้ำมันแตกตัวและเกิดก๊าซขึ้นภายในไปดันหน้าสัมผัส ให้รีเลย์ทำงาน

2.5.5 ประโยชน์ของรีเลย์

1. ทำให้ระบบส่งกำลังมีเสถียรภาพ (Stability) สูงโดยรีเลย์จะตัดวงจรเฉพาะส่วนที่เกิดผิดปกติออกเท่านั้น ซึ่งจะเป็นการลดความเสียหายให้แก่ระบบน้อยที่สุด
2. ลดค่าใช้จ่ายในการซ่อมแซมส่วนที่เกิดผิดปกติ
3. ลดความเสียหายไม่เกิดลุกลามไปยังอุปกรณ์อื่นๆ
4. ทำให้ระบบไฟฟ้าไม่ดับทั้งระบบเมื่อเกิดฟอลต์ขึ้นในระบบ

2.6 การหาความคลาดเคลื่อน

ความคลาดเคลื่อน (error) หรือ static error คือ ผลต่างระหว่างค่าที่วัดได้กับค่าที่แท้จริง โดยทั่วไปแสดงเป็นเปอร์เซ็นต์ (%) ถ้าค่าที่วัดได้ใกล้เคียงกับค่าจริงมากแสดงว่าการวัดนั้นมีความแม่นยำหรือความถูกต้อง (accuracy) สูง โดยการวัดทุกครั้งมักมีค่าความคลาดเคลื่อนเกิดขึ้นเสมอ การเข้าใจถึงสาเหตุจะช่วยลดความคลาดเคลื่อนให้น้อยลงได้ โดยความคลาดเคลื่อนที่เกิดขึ้นอาจเป็นสาเหตุทำให้เกิดความไม่แน่นอน (uncertainty)

การวัดความคลาดเคลื่อนแบ่งออกเป็น 3 ชนิดได้แก่

- ความคลาดเคลื่อนที่เกิดจากผู้วัด (gross error หรือ human error)
- ความคลาดเคลื่อนเชิงระบบ (systematic error)
- ความคลาดเคลื่อนแบบสุ่ม (random error)

2.6.1 การคำนวณค่าความคลาดเคลื่อนจากการวัด (calculation of measurement error)

ค่าความคลาดเคลื่อนสัมบูรณ์ (absolute error) คือ ค่าปริมาณความแตกต่างระหว่างค่าจริงกับค่าที่ได้จากการวัด สามารถหาได้จากสมการ

$$\text{Relative error} = \left| \frac{X_{mea} - X_t}{X_t} \right|$$

$$\text{\% Error} = \text{Relative error x100}$$

โดย X_t คือ ค่าจริง (True value)

X_{mea} คือ ค่าที่ได้จากการวัด (Measure value)

2.7 การหาค่าประสิทธิภาพ

$$E = \frac{\sum X}{N \times A} \times 100$$

$$= \frac{\text{ค่าเฉลี่ย}}{A} \times 100$$

เมื่อ A = จำนวนเมล็ดที่ต้องการ

บทที่3

วิธีดำเนินงาน

3.1 คุณสมบัติตามที่ต้องการ

1. สั่งการมอเตอร์ไปยังตำแหน่งที่กำหนดไว้
2. ควบคุมมอเตอร์ผ่านบอร์ด Arduino เคลื่อนที่ตามแนวแกน X Y
3. หยอดเมล็ดพืชและรดน้ำตามที่กำหนดไว้

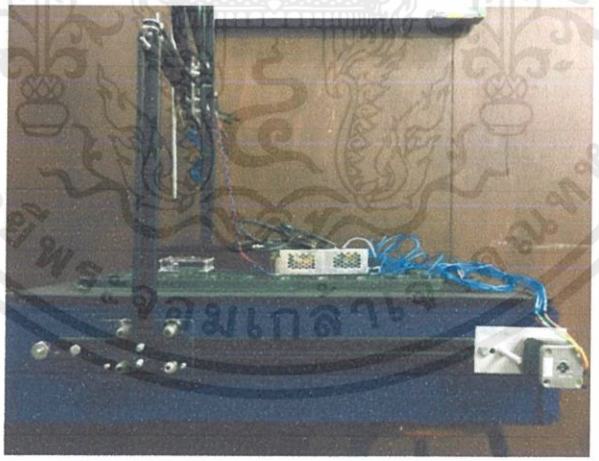
3.2 ส่วนประกอบ

3.2.1 ส่วนโครงสร้าง

โครงสร้างแบ่งออกได้ดังนี้

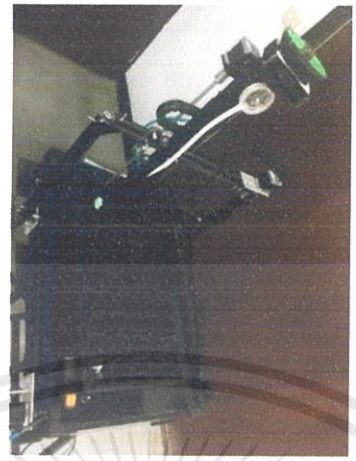
1. ส่วนฐานของเครื่องทำจากไม้มาประอบเป็นรูปสี่เหลี่ยมผืนผ้ามีขนาดความกว้าง 39 ซม. ยาว 34 ซม. สูง 13 ซม. สำหรับปลูกผัก

2. ส่วนของแกน x และ แกน y ใช้สเตปมอเตอร์ในการควบคุมการเคลื่อนที่โดยใช้อะลูมิเนียมโปรไฟล์ยึดกับส่วนฐานเพื่อรับน้ำหนักของมอเตอร์ดังรูปที่ 3.1



รูปที่ 3.1 โครงสร้างในแนวแกน X

3. ส่วนของแกน Z ใช้สเตปมอเตอร์ควบคุมในการเคลื่อนที่ดังในรูป 3.2

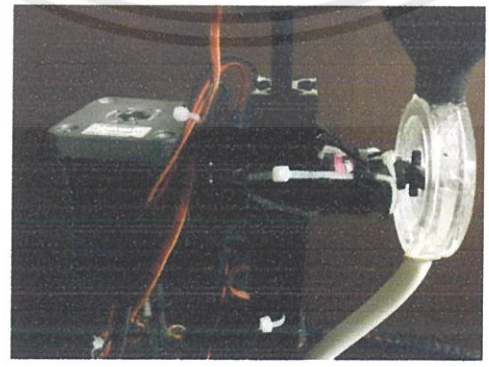


รูปที่ 3.2 โครงสร้างในแนวแกน Z

4. ออกแบบแผ่นยึดล้อจำนวน 2 แผ่นโดยมีขนาดกว้าง 6.5 ซม. ยาว 10.2 ซม.และขนาดกว้าง10.1 ซม ยาว 8.5 ซม ในโปรแกรม autocad

รูปที่ 3.3 แสดงแผ่นยึดล้อ

5. ส่วนของการหยอดเมล็ดใช้เซอร์โวมอเตอร์ในการควบคุมจำนวนการหยอดเมล็ดโดยกำหนดให้หยอดเมล็ดครั้งละ 3 เมล็ด



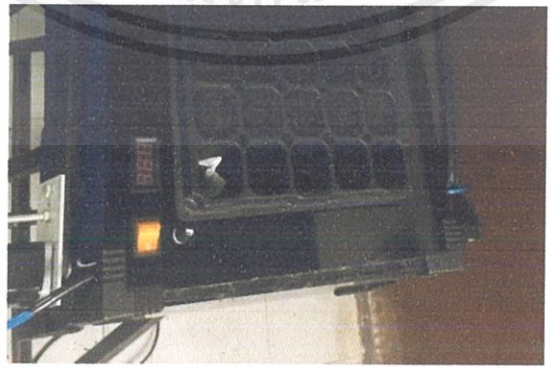
รูปที่ 3.4 แสดงส่วนของการหยอดเมล็ด

รูปที่ 3.5 แสดงส่วนของที่ใส่เมล็ดพืช

6. แสดงส่วนการให้น้ำโดยใช้ปั๊มน้ำดีซี 12 โวลต์ดังแสดงในรูป 3.7

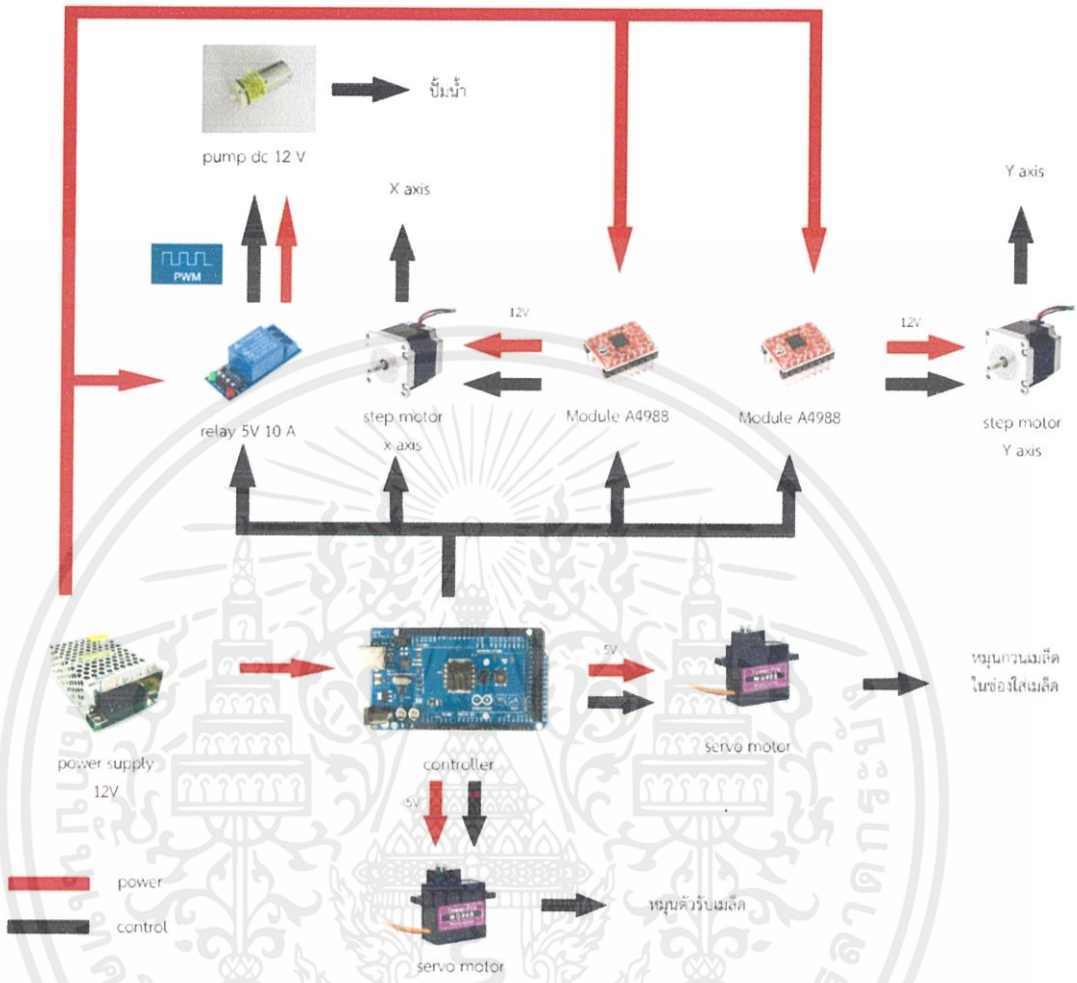
รูปที่ 3.6 แสดงส่วนของการให้น้ำ

7. ประกอบบอร์ดทั้งหมดลงกล่อง สร้างเป็นกล่องควบคุมอยู่ด้านในของแปลงผัก



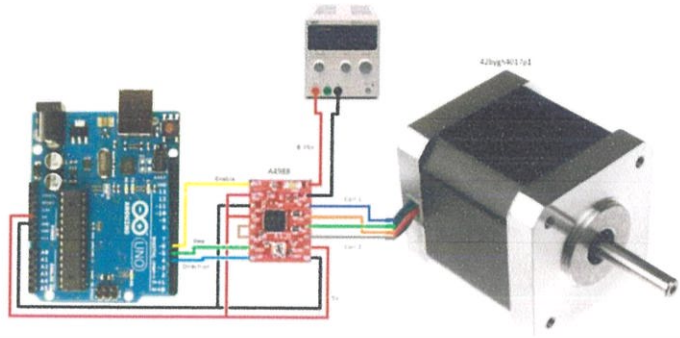
รูปที่ 3.7 แสดงส่วนต่างๆในการควบคุมมอเตอร์

3.2.2 ส่วนควบคุม

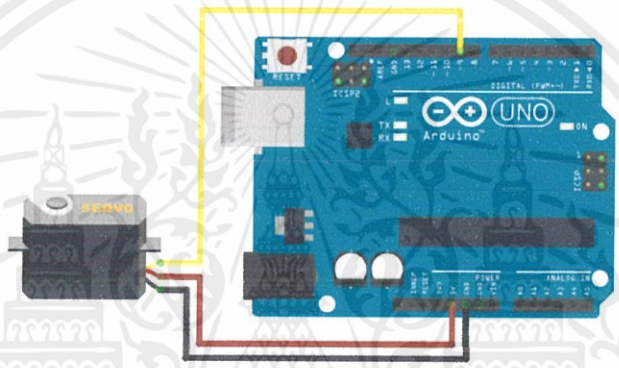


รูปที่ 3.8 แสดงบล็อกไดอะแกรม

การควบคุมการเคลื่อนที่ต่างๆใน หุ่นยนต์อัจฉริยะภาพเพื่อการเกษตรและเทคโนโลยีชีวภาพ ต้องใช้ สเตปมอเตอร์ 3 ตัวในการบังคับแกน x y z โดยการควบคุมสเตปมอเตอร์ต้องใช้แรงดันที่เหมาะสมเพื่อให้มอเตอร์หมุนไปตำแหน่งที่ต้องการโดยองศาในการหมุนขึ้นอยู่กับคุณสมบัติของมอเตอร์ตัวนั้น ๆว่าสามารถหมุนได้กี่องศาในแต่ละรอบแล้วจึงคำนวณหาระยะทางการเคลื่อนที่องศาการหมุนของมอเตอร์ มอเตอร์ทั้งสามจะถูกควบคุมด้วย microcontroller ควบคุมกับวงจรไดร์ฟกระแสมอเตอร์โดยสั่งผ่าน microcontroller ดังบล็อกไดอะแกรมดังรูป



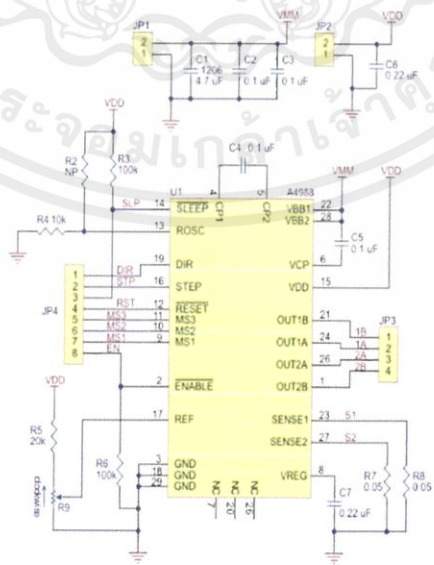
รูปที่ 3.9 แสดงส่วนควบคุมสเตปมอเตอร์



รูปที่ 3.10 แสดงส่วนควบคุมเซอร์โวมอเตอร์

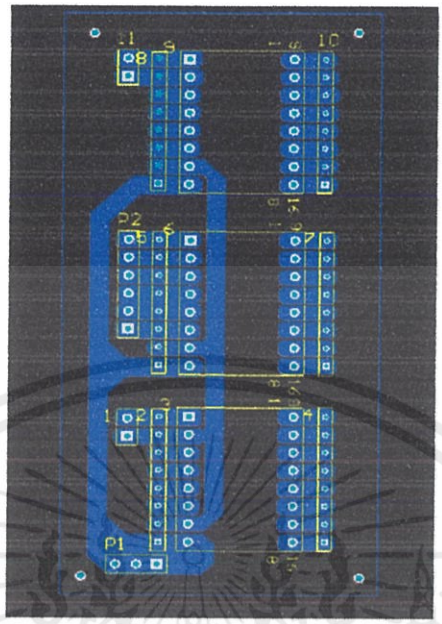
3.2.3 บอร์ดไดร์ฟมอเตอร์

1. วาด schematic ในโปรแกรม Altium



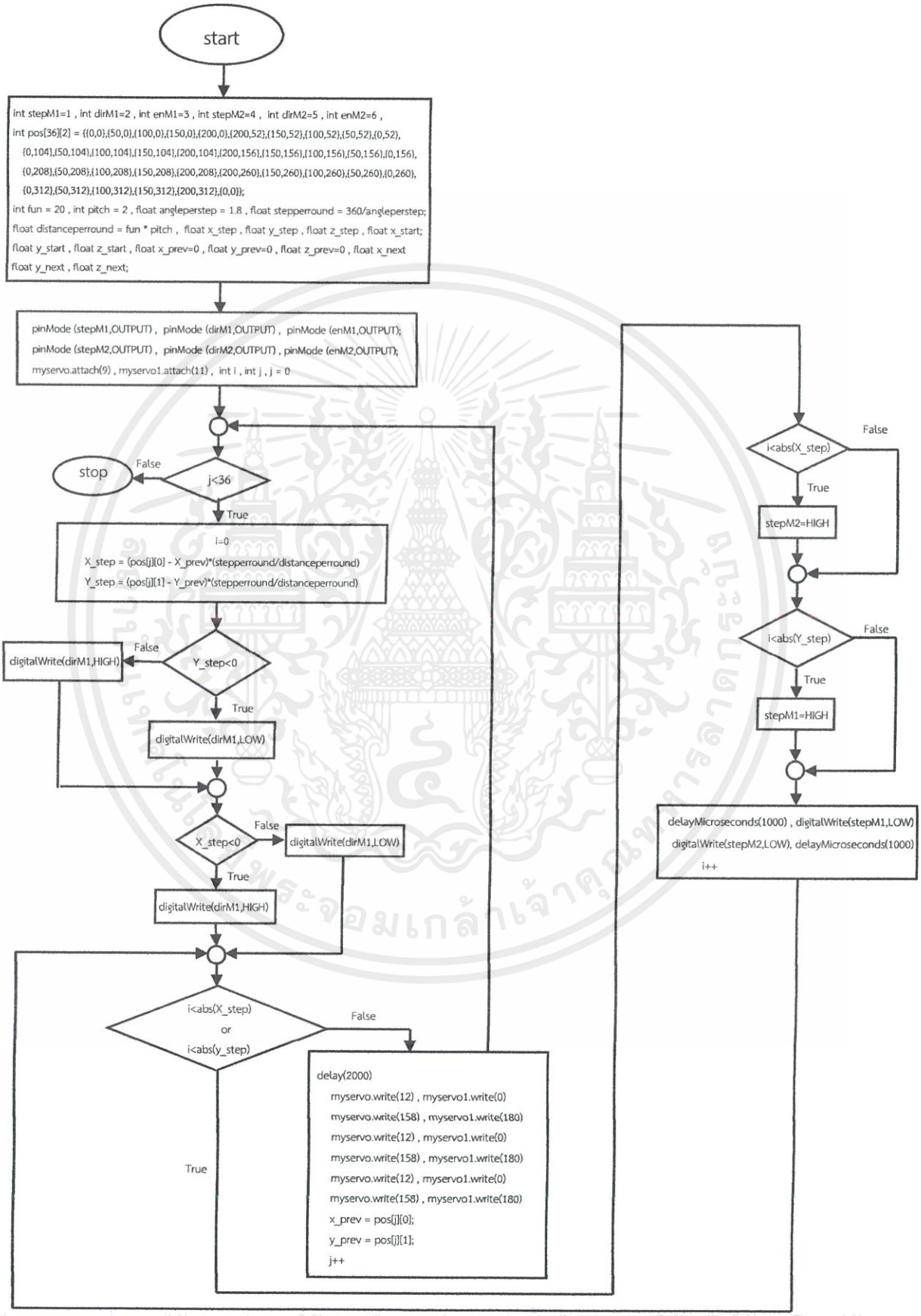
รูปที่ 3.11 แสดง schematic บอร์ดไดร์ฟมอเตอร์

2. ออกแบบ PCB บอร์ดไดร์ฟมอเตอร์



รูปที่ 3.12 แสดง PCB บอร์ดไดร์ฟมอเตอร์

3.2.4 ส่วนโปรแกรม



บทที่ 4
ผลการทดลอง

4.1 ผลการทดสอบการหยอดเมล็ด

ตารางที่ 4.1 ผลการทดสอบจำนวนเมล็ดที่หยอดลงไปในแต่ละหลุมโดยในแต่ละหลุมไม่เกิน 3 เมล็ด

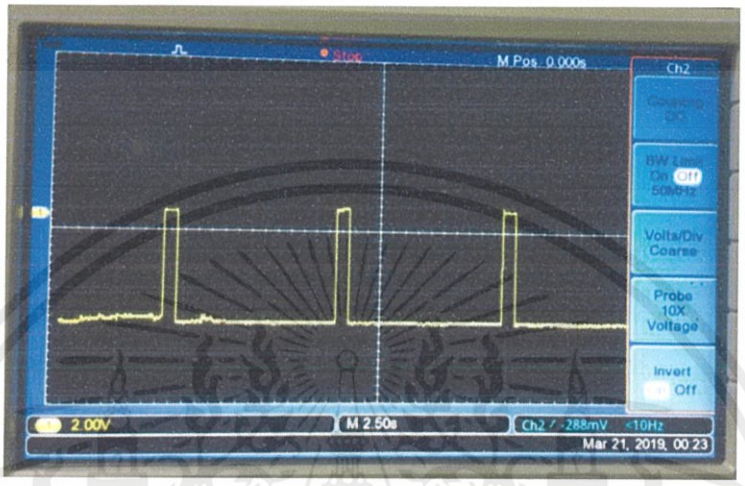
หลุมที่	ครั้งที่			เฉลี่ย	%ความคลาดเคลื่อน	%ประสิทธิภาพ
	จำนวนเมล็ดที่ลง(เมล็ด)					
	1	2	3			
1	3	3	3	3.00	0.00	100
2	3	3	3	3.00	0.00	100
3	3	3	2	2.67	11.00	89
4	2	3	3	2.67	11.00	89
5	3	3	3	3.00	0.00	100
6	3	3	3	3.00	0.00	100
7	3	3	3	3.00	0.00	100
8	3	3	3	3.00	0.00	100
9	3	2	3	2.67	11.00	89
10	3	3	2	2.67	11.00	89
11	3	3	3	3.00	0.00	100
12	3	3	2	2.67	11.00	89
13	3	2	3	2.67	11.00	89
14	3	3	3	3.00	0.00	100
15	3	3	3	3.00	0.00	100

ตารางที่ 4.1 (ต่อ)

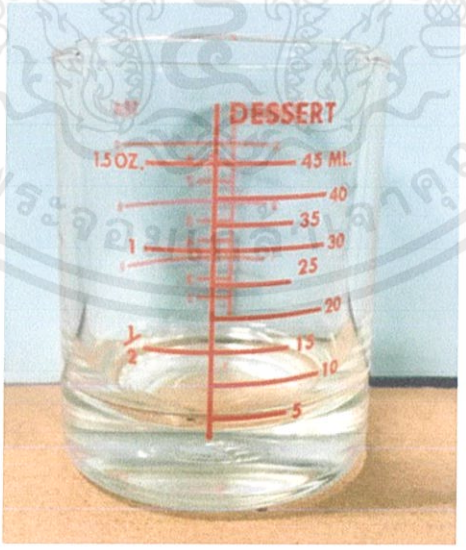
หลุมที่	ครั้งที่ จำนวนเมล็ดที่ลง			เฉลี่ย	%ความคลาดเคลื่อน	ประสิทธิภาพ
	1	2	3			
16	3	3	3	3.00	0.00	100
17	3	3	3	3.00	0.00	100
18	3	3	3	3.00	0.00	100
19	3	2	3	2.67	11.00	89
20	3	2	3	2.67	11.00	89
21	3	2	3	2.67	11.00	89
22	3	3	2	2.67	11.00	89
23	3	3	2	2.67	11.00	89
24	3	3	2	2.67	11.00	89
25	2	3	3	2.67	11.00	89
26	3	3	3	3.00	0.00	100
27	3	3	3	3.00	0.00	100
28	3	3	3	3.00	0.00	100
29	3	3	3	3.00	0.00	100
30	3	3	3	3.00	0.00	100
31	3	3	3	3.00	0.00	100
32	3	3	3	3.00	0.00	100
33	2	3	3	2.67	11.00	89
34	3	2	3	2.67	11.00	89
35	3	3	2	2.67	11.00	89

4.2 การอ่าน PWM Signal

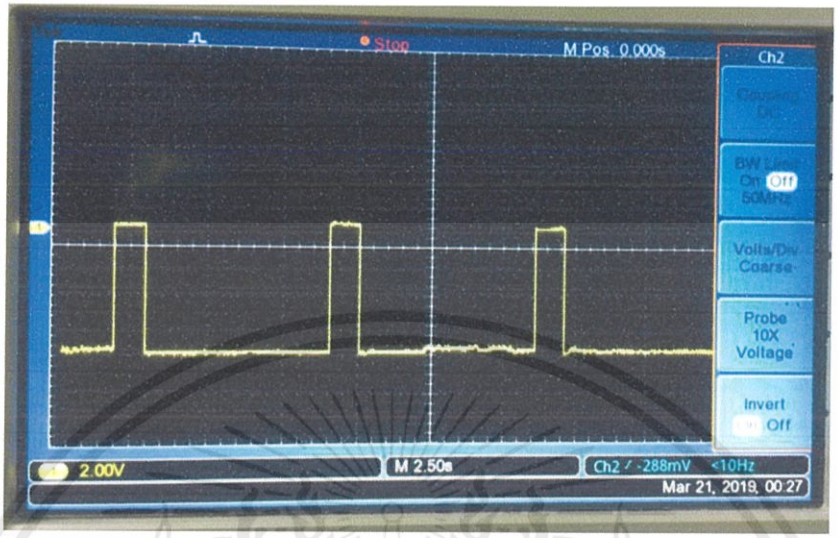
โปรแกรมบอร์ดคอนโทรล Arduino ให้ควบคุมปั๊มน้ำโดยวัดค่าอินพุตที่สามารถทำให้ปั๊มน้ำสามารถทำงานได้



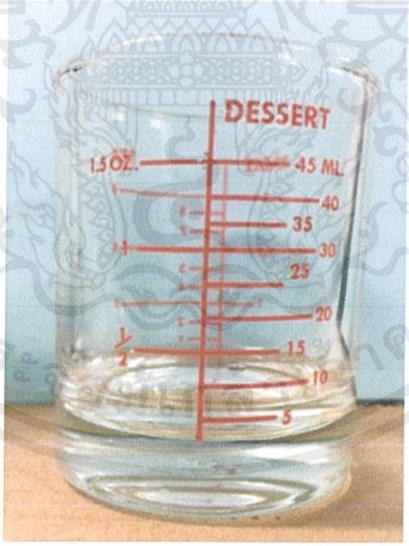
รูปที่ 4.1 แสดงสัญญาณสั่งให้ปั๊มน้ำทำงานที่เวลา 0.5 วินาที



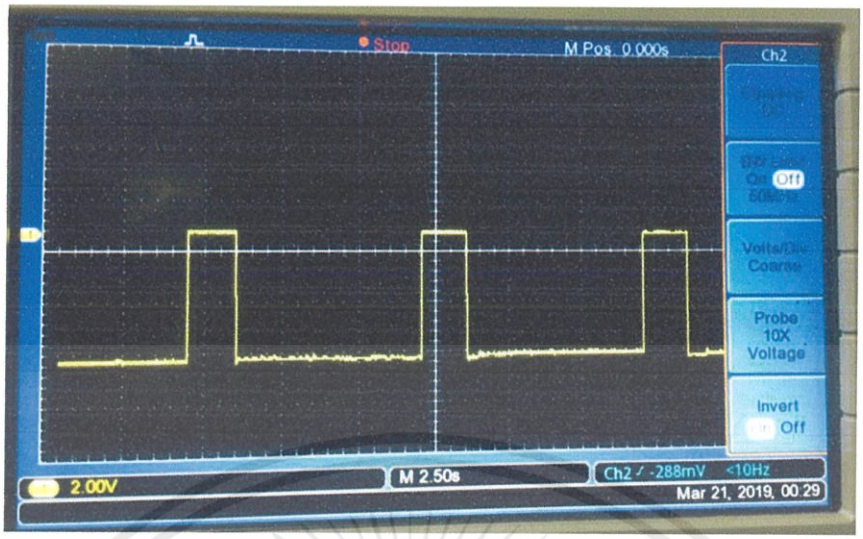
รูปที่ 4.2 แสดงปริมาณน้ำที่เวลา 0.5 วินาที



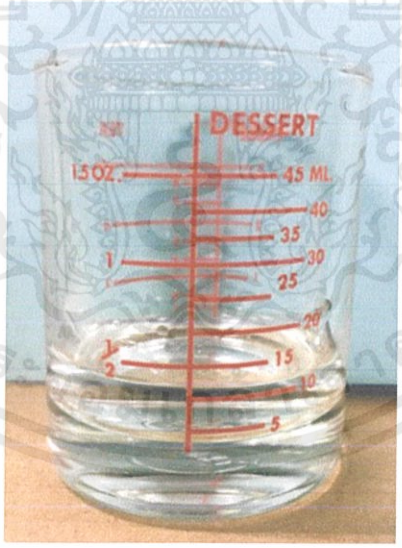
รูปที่ 4.3 แสดงสัญญาณสั่งให้ปั๊มน้ำทำงานที่เวลา 1 วินาที



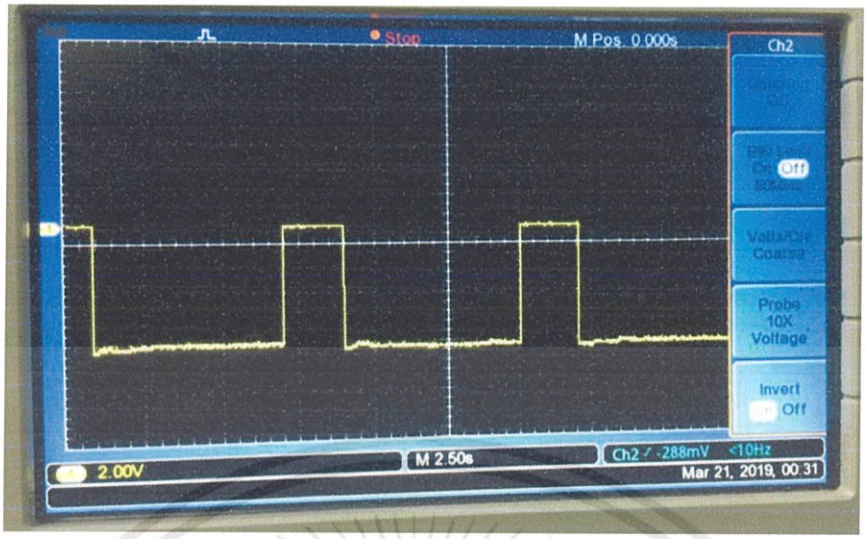
รูปที่ 4.4 แสดงปริมาณน้ำที่เวลา 1 วินาที



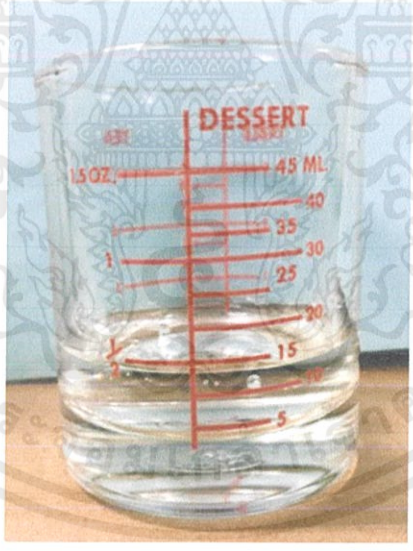
รูปที่ 4.5 แสดงสัญญาณสั่งให้ปั๊มน้ำทำงานที่เวลา 1.5 วินาที



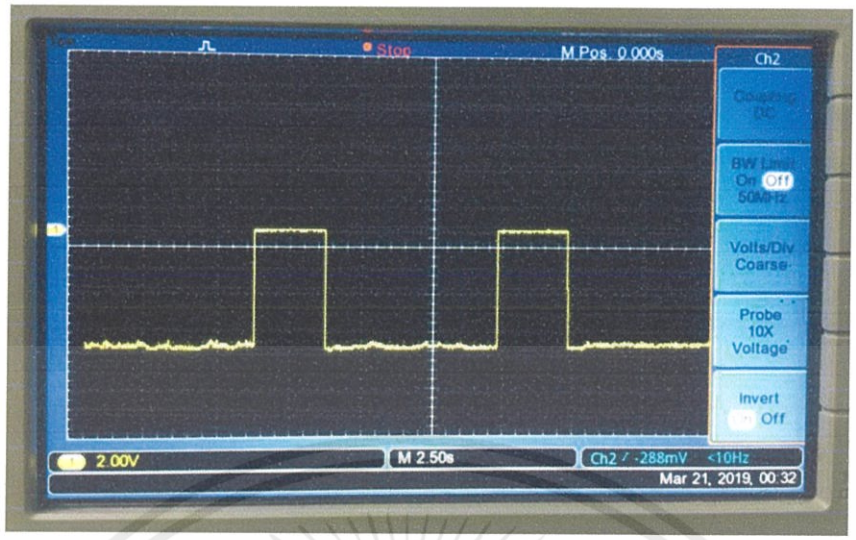
รูปที่ 4.6 แสดงปริมาณน้ำที่เวลา 1.5 วินาที



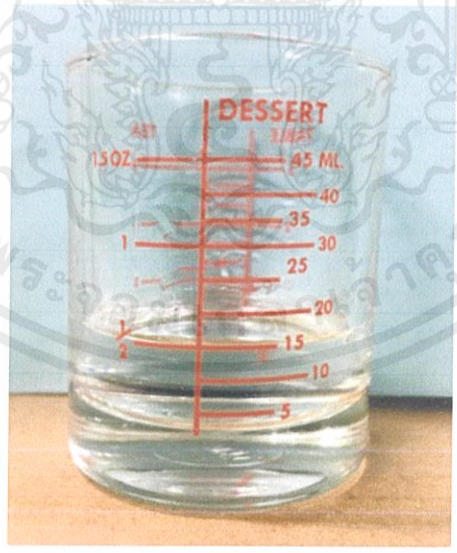
รูปที่ 4.7 แสดงสัญญาณสั่งให้ปั๊มน้ำทำงานที่เวลา 2.0 วินาที



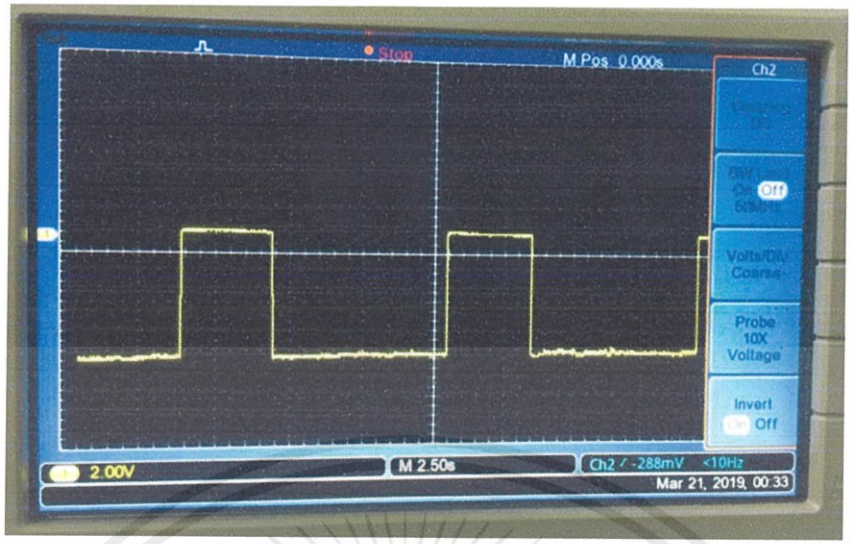
รูปที่ 4.8 แสดงปริมาณน้ำที่เวลา 2.0 วินาที



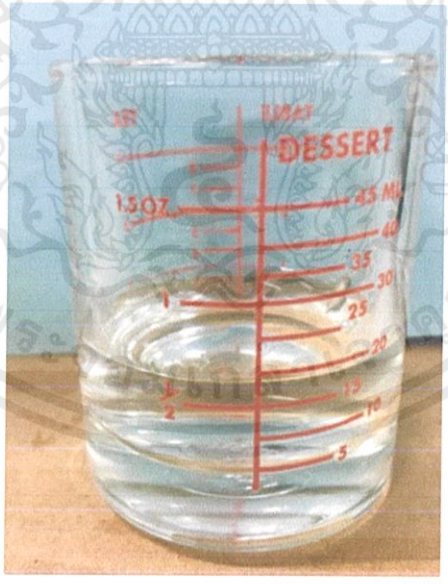
รูปที่ 4.9 แสดงสัญญาณสั่งให้ปั๊มน้ำทำงานที่เวลา 2.5 วินาที



รูปที่ 4.10 แสดงปริมาณน้ำที่เวลา 2.5 วินาที



รูปที่ 4.11 แสดงสัญญาณสั่งให้ปั๊มน้ำทำงานที่เวลา 3.0 วินาที



รูปที่ 4.12 แสดงปริมาณน้ำที่เวลา 3.0 วินาที

4.3 ตัวอย่างผลผลิตที่ได้

รูปที่ 4.13 แสดงการหยอดเมล็ดพืชในแต่ละหลุม

รูปที่ 4.14 แสดงการงอกของเมล็ดพืช

บทที่ 5
สรุปผลการทดลอง

5.1 สรุปผลการทดลอง

จากการทดลองเพื่อทดสอบการทำงานของหุ่นยนต์อัจฉริยะภาพเพื่อการเกษตรและเทคโนโลยีชีวภาพ พบว่าเครื่องสามารถทำงานได้ตามที่ต้องการคือสามารถเคลื่อนที่ไปตำแหน่งที่ต้องการได้หยอดเมล็ดพืชตามที่กำหนดไว้คือหลุมละ 3 เมล็ดพร้อมกับรดน้ำพืช โดยใช้เซอร์โวมอเตอร์เป็นตัวควบคุมจำนวนเมล็ดให้เหมาะสม ใช้สเตปมอเตอร์ในการเคลื่อนที่ไปยังตำแหน่งที่กำหนดไว้ และควบคุมปั๊มน้ำโดยใช้สัญญาณ PWM โดยที่เวลา 3 วินาทีเป็นเวลาที่มีปริมาณน้ำเหมาะสมตามที่เราต้องการ

5.2 ปัญหาและอุปสรรค

จากการทดลองพบปัญหาคือ

- การเลือกขนาดของเมล็ดพืชและท่อที่ไม่เหมาะสมกัน ทำให้เมล็ดพืชติดในท่อส่งผลให้เมล็ดไม่ไหลออกมา
- การต่อสายไฟที่ไม่ดี ส่งผลให้วงจรเกิดความเสียหาย

5.3 ข้อเสนอแนะ

- ควรออกแบบและจัดวางอุปกรณ์ให้เหมาะสม
- ควรเลือกอุปกรณ์ในการทำโครงงานให้เหมาะสม

เอกสารอ้างอิง

[1] “Arduino.” (ระบบออนไลน์).
แหล่งที่มา: https://www.thaieasyelec.com/article-wiki/basic-electronics 18 พฤศจิกายน 2561

[2] “ไมโคคอนโทรเลอร์” (ระบบออนไลน์)
แหล่งที่มา: http://www.sbt.ac.th/new/sites/default/files/TNP_Unit_1.pdf 18 พฤศจิกายน 2561

[3] “บอร์ด arduino mega R3”
แหล่งที่มา: http://paiboondev.blogspot.com/p/arduino.html 18 พฤศจิกายน 2561

[4] “โครงสร้างการโปรแกรมสำหรับ Arduino” (ระบบออนไลน์).
แหล่งที่มา: http://cpre.kmutnb.ac.th/esl/wp-content/uploads/2014/10/ESL_Arduino_Workshop_for_CprE-2014-10-12.pdf 18 พฤศจิกายน 2561

[5]”การควบคุมมอเตอร์และสเตปมอเตอร์ด้วยโปรแกรม Arduino” (ระบบออนไลน์).
แหล่งที่มา: http://www.sbt.ac.th/new/sites/default/files/TNP_Unit_8.pdf 19 พฤศจิกายน 2561

[6] “การควบคุมเซอร์โวมอเตอร์ด้วยโปรแกรม Arduino” (ระบบออนไลน์)
แหล่งที่มา : http://www.semi-shop.com/knowledge/knowledge_detail.php?sk_id=105
http://www.sbt.ac.th/new/sites/default/files/TNP_Unit_9.pdf 19 พฤศจิกายน 2561

[7]”บอร์ด Arduino mega2560 R3” (ระบบออนไลน์)
แหล่งที่มา : https://www.myarduino.net/article/4 2 มกราคม 2562

[8] “ปั๊มน้ำดีซี” (ระบบออนไลน์)
แหล่งที่มา : https://solarsmileknowledge.com 14 มกราคม 2562

[9]” รีเลย์” (ระบบออนไลน์)
แหล่งที่มา : https://mall.factomart.com/relay/ 14 มกราคม 2562

[10] “การหาค่าความคลาดเคลื่อน” (ระบบออนไลน์)
แหล่งที่มา : http://www.foodnetworksolution.com/wiki/word/7240/error 14 มกราคม 2562

ภาคผนวก

โปรแกรมควบคุม

```
#include <Servo.h>

Servo myservo;

Servo myservo1;

const int stepM1=1;

const int dirM1=2;

const int enM1=3;

const int stepM2=4;

const int dirM2=5;

const int enM2=6;

//int pos[10][2] = {{0,0},{50,0},{100,0},{150,0},{200,0},{200,50},{150,50},{100,50},{50,50},{0,50}};

int pos[36][2] = {{0,0},{50,0},{100,0},{150,0},{200,0},

                  {200,52},{150,52},{100,52},{50,52},{0,52},

                  {0,104},{50,104},{100,104},{150,104},{200,104},

                  {200,156},{150,156},{100,156},{50,156},{0,156},

                  {0,208},{50,208},{100,208},{150,208},{200,208},

                  {200,260},{150,260},{100,260},{50,260},{0,260},

                  {0,312},{50,312},{100,312},{150,312},{200,312},{0,0}};

int fun = 20;

int pitch = 2;

float angleperstep = 1.8;

float stepperround = 360/angleperstep;

float distanceperround = fun * pitch;

float x_step;

float y_step;
```

```
 float z_step;
 float x_start;
 float y_start;
 float z_start;
 float x_prev=0;
 float y_prev=0;
 float z_prev=0;
 float x_next;
 float y_next;
 float z_next;
 void setup() {
   pinMode (stepM1,OUTPUT);
   pinMode (dirM1,OUTPUT);
   pinMode (enM1,OUTPUT);
   pinMode (stepM2,OUTPUT);
   pinMode (dirM2,OUTPUT);
   pinMode (enM2,OUTPUT);
myservo.attach(9);
myservo1.attach(11);
 int i;
 int j;
 int k;
 i = 0;
 j = 0;
 k = 0;
```

```
 /*
 x_start = 250 * (stepperround/distanceperround);
 digitalWrite(dirM2,LOW);
 i = 0;
 while(i<x_start)
 {
   digitalWrite(stepM2,HIGH);
   delayMicroseconds(1000);
   digitalWrite(stepM2,LOW);
   delayMicroseconds(1000);
   i++;
 }
 y_start = 310 * (stepperround/distanceperround);
 digitalWrite(dirM1,LOW);
 i = 0;
 while(i<y_start)
 {
   digitalWrite(stepM1,HIGH);
   delayMicroseconds(1000);
   digitalWrite(stepM1,LOW);
   delayMicroseconds(1000);
   i++;
 }
```

```
  z_start = 300 * (stepperround/distanceperround);
   digitalWrite(dirM3,LOW);
   i = 0;
   while(i<z_start)
   {
     digitalWrite(stepM3,HIGH);
     delayMicroseconds(1000);
     digitalWrite(stepM3,LOW);
     delayMicroseconds(1000);
     i++;
   }
   */
while(j < 36)
  {
      i = 0;
      x_step = (pos[j][0] - x_prev) * (stepperround/distanceperround);
      y_step = (pos[j][1] - y_prev) * (stepperround/distanceperround);
      //x_step = pos[j][0] * (stepperround/distanceperround);
      //y_step = pos[j][1] * (stepperround/distanceperround);
      if (y_step < 0)
      {
        digitalWrite(dirM1,LOW);
      }
      else
```

```
{
  digitalWrite(dirM1,HIGH);
}
if (x_step < 0)
{
  digitalWrite(dirM2,HIGH);
}
else
{
  digitalWrite(dirM2,LOW);
}
while((i<abs(x_step)) || (i<abs(y_step)))
{
  if (i < abs(x_step))
  {
    digitalWrite(stepM2,HIGH);
  }
  if (i < abs(y_step))
  {
    digitalWrite(stepM1,HIGH);
  }
  delayMicroseconds(1000);
  digitalWrite(stepM1,LOW);
  digitalWrite(stepM2,LOW);
  delayMicroseconds(1000);
```

```
  i++;}
 delay (2000);
 myservo.write(12);//บน
 myservo1.write(0);
 delay (1000);
 myservo.write(158);//ล่าง
 myservo1.write(180);
 delay (1000);
 myservo.write(12);
 myservo1.write(0);
 delay(1000);
 myservo.write(158);
 myservo1.write(180);
 delay(1000);
 myservo.write(12);
 myservo1.write(0);
 delay(1000);
 myservo.write(158);
 myservo1.write(180);
 delay(8000);
 x_prev = pos[j][0];
 y_prev = pos[j][1];
 /*
 z_step = 300 * (stepperround/distanceperround);
 digitalWrite(dirM3,HIGH);
```

```
      i = 0;
  while(i<z_step)
      {
        digitalWrite(stepM3,HIGH);
        delayMicroseconds(1000);
        digitalWrite(stepM3,LOW);
        delayMicroseconds(1000);
        i++;
      }
       digitalWrite(dirM3,LOW);
      i = 0;
      while(i<z_step)
      {
        digitalWrite(stepM3,HIGH);
        delayMicroseconds(1000);
        digitalWrite(stepM3,LOW);
        delayMicroseconds(1000);
        i++;
      }
*/
/*
      digitalWrite(dirM1,LOW);
      digitalWrite(dirM2,LOW);
      i = 0;
      while((i<x_step) || (i<y_step))
```

```
  {
      if (i < x_step)
   {
     digitalWrite(stepM2,HIGH);
   }
   if (i < y_step)
   {
     digitalWrite(stepM1,HIGH);
   }
   delayMicroseconds(1000);
   digitalWrite(stepM1,LOW);
   digitalWrite(stepM2,LOW);
   delayMicroseconds(1000);
   i++;
  }
  */
/*
  z_step = 300 * (stepperround/distanceperround);
  digitalWrite(dirM3,HIGH);
  i = 0;
  while(i<z_step)
  {
   digitalWrite(stepM3,HIGH);
   delayMicroseconds(1000);
   digitalWrite(stepM3,LOW);
```

```
        delayMicroseconds(1000);
        i++;
      }
      digitalWrite(dirM3,LOW);
      i = 0;
      while(i<z_step)
      {
        digitalWrite(stepM3,HIGH);
        delayMicroseconds(1000);
        digitalWrite(stepM3,LOW);
        delayMicroseconds(1000);
        i++;
      }
*/
      j++;
  }
/*
   if (x_stepstep > y_stepstep)
   {
       if (i < x_stepstep)
       {
         digitalWrite(stepM2,HIGH);
       }
       else
       {
```

```
            break; //ถ้าไม่เป็นจิงจะหยุด
          }
          if (i < y_stepstep)
          {
            digitalWrite(stepM1,HIGH);
          }
      }
      else if (x_stepstep < y_stepstep)
      {
          if (i < y_stepstep)
          {
            digitalWrite(stepM1,HIGH);
          }
          else
          {
            break; //ถ้าไม่เป็นจิงจะหยุด
          }
          if (i < x_stepstep)
          {
            digitalWrite(stepM2,HIGH);
          }
      }
  */
  /*
    while(true) //ถ้าเงื่อนไขเป็นจิง (เป็นจิงตลอด)
```

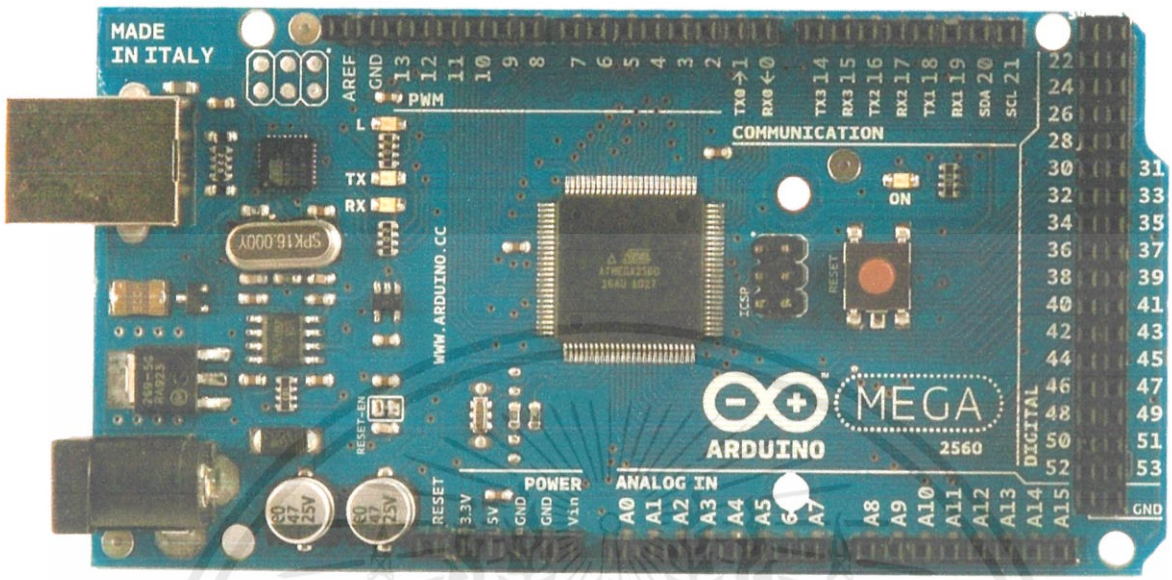
```
  {
    if (x < 1000)
    {
      digitalWrite(stepM1,HIGH);
    }
    else
    {
      break; //ถ้าไม่เป็นจิงจะหยุด
    }
    if (x <700)
    {
      digitalWrite(stepM2,HIGH);
    }
    delayMicroseconds(1000);
    digitalWrite(stepM1,LOW);
    digitalWrite(stepM2,LOW);
    delayMicroseconds(1000);
    x++;
  }
  digitalWrite(dirM1,LOW);
  digitalWrite(dirM2,LOW);
  x = 0;
  while(true)
  {
    if (x < 1000)
```

```
    {
      digitalWrite(stepM1,HIGH);
    }
    else
    {
      break;
    }
    if (x < 700)
    {
      digitalWrite(stepM2,HIGH);
    }
    delayMicroseconds(1000);
    digitalWrite(stepM1,LOW);
    digitalWrite(stepM2,LOW);
    delayMicroseconds(1000);
    x++;
  }
*/
}
void loop() {
  /*
  digitalWrite(dirM1,HIGH);
  //digitalWrite(dirM2,HIGH);
  for(int x=0;x<1000;x++){
    digitalWrite(stepM1,HIGH);
```

```
    delayMicroseconds(1200);
    digitalWrite(stepM1,LOW);
    delayMicroseconds(1200);
  }
  delay (2000);
  digitalWrite(dirM1,LOW);
  for(int x=0;x<1000;x++){
    digitalWrite(stepM1,HIGH);
    delayMicroseconds(1200);
    digitalWrite(stepM1,LOW);
    delayMicroseconds(1200);
  }
  delay(2000);
digitalWrite(dirM2,HIGH);
  for(int i=0;i<700;i++){
    digitalWrite(stepM2,HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepM2,LOW);
    delayMicroseconds(1000);
  }
  delay(2000);
  digitalWrite(dirM2,LOW);
  for(int i=0;i<700;i++){
    digitalWrite(stepM2,HIGH);
    delayMicroseconds(1000);
```

```
      digitalWrite(stepM2,LOW);
      delayMicroseconds(1000);
    }
    delay(2000);
  digitalWrite(dirM3,HIGH);
    for(int i=0;i<2000;i++){
      digitalWrite(stepM3,HIGH);
      delayMicroseconds(1000);
      digitalWrite(stepM3,LOW);
      delayMicroseconds(1000);
    }
    delay(2000);
    digitalWrite(dirM3,LOW);
    for(int i=0;i<700;i++){
      digitalWrite(stepM3,HIGH);
      delayMicroseconds(1000);
      digitalWrite(stepM3,LOW);
      delayMicroseconds(1000);
    }
     delay(2000);
  */
}
```

Arduino MEGA 2560



CE

Product Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Index

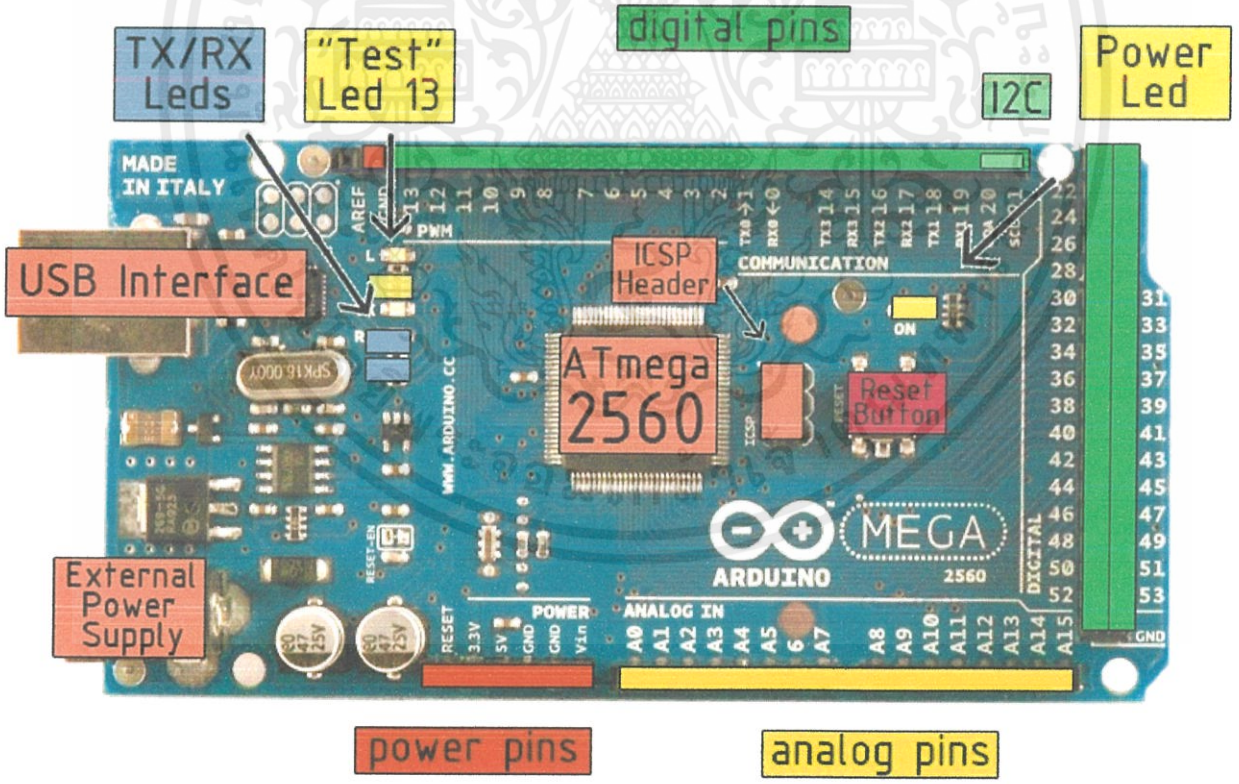
Technical Specification

EAGLE files: arduino-mega2560-reference-design.zip Schematic: arduino-mega2560-schematic.pdf

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

the board



Power

The Arduino Mega2560 can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega 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 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.
- **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.
- **I^2C: 20 (SDA) and 21 (SCL).** Support I^2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I^2C pins on the Duemilanove.

The Mega2560 has 16 analog inputs, 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 analogReference() function.

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.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Mega's digital pins.

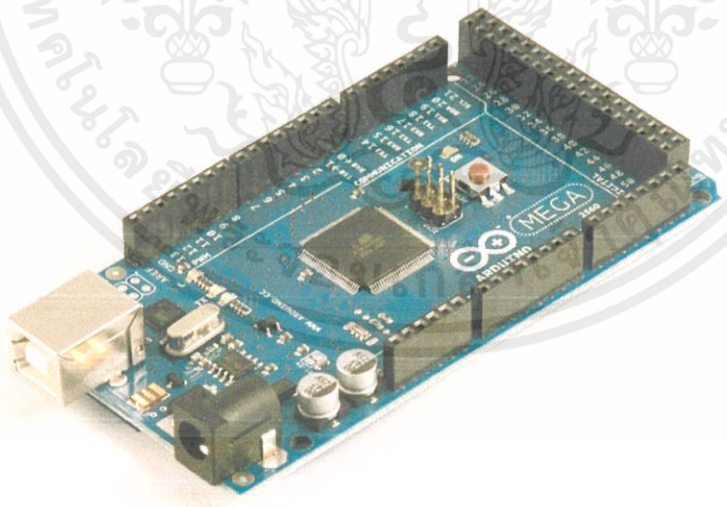
The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. To use the SPI communication, please see the ATmega2560 datasheet.

Programming

The Arduino Mega2560 can be programmed with the Arduino software (download). For details, see the reference and tutorials.

The Atmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.



Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Mega2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Mega2560 is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Mega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Mega contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Mega has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics and Shield Compatibility

The maximum length and width of the Mega PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega is designed to be compatible with most shields designed for the Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega and Duemilanove / Diecimila. **Please note that I^2C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).**

How to use Arduino

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).

Arduino is a cross-platoform program. You'll have to follow different instructions for your personal OS. Check on the Arduino site for the latest instructions. *http://arduino.cc/en/Guide/HomePage*

Linux Install

Windows Install

Mac Install

Once you have downloaded/unzipped the arduino IDE, you can Plug the Arduino to your PC via USB cable.

Blink led

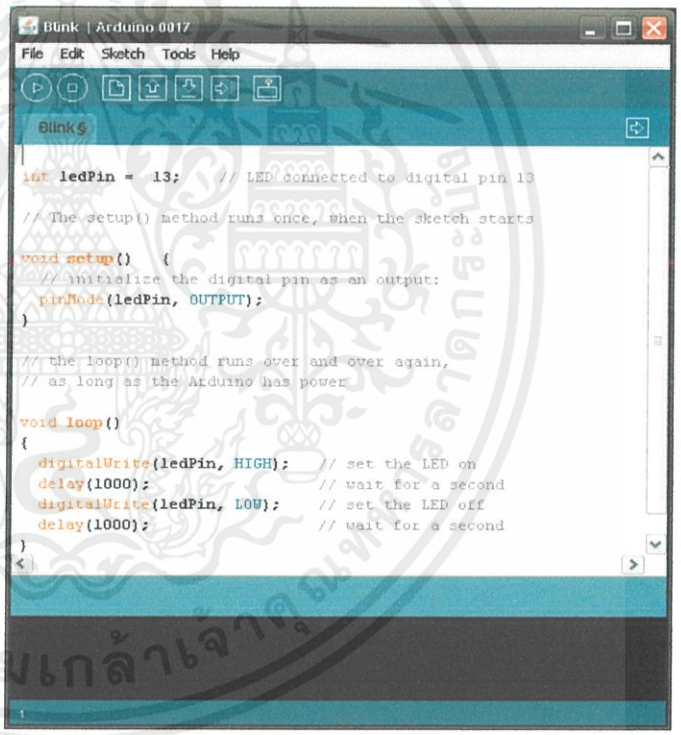
Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select

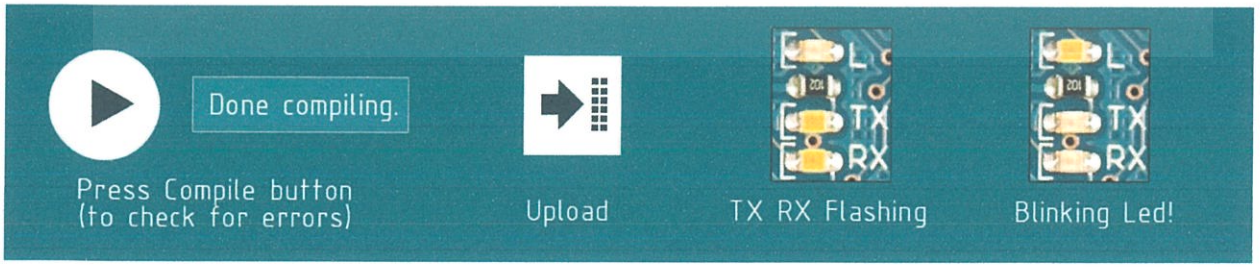
File>Sketchbook>
Arduino-0017>Examples>
Digital>Blink

Once you have your skecth you'll see something very close to the screenshot on the right.

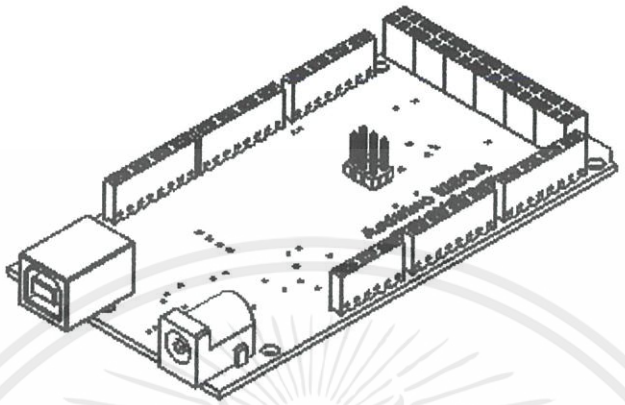
In **Tools>Board** select MEGA

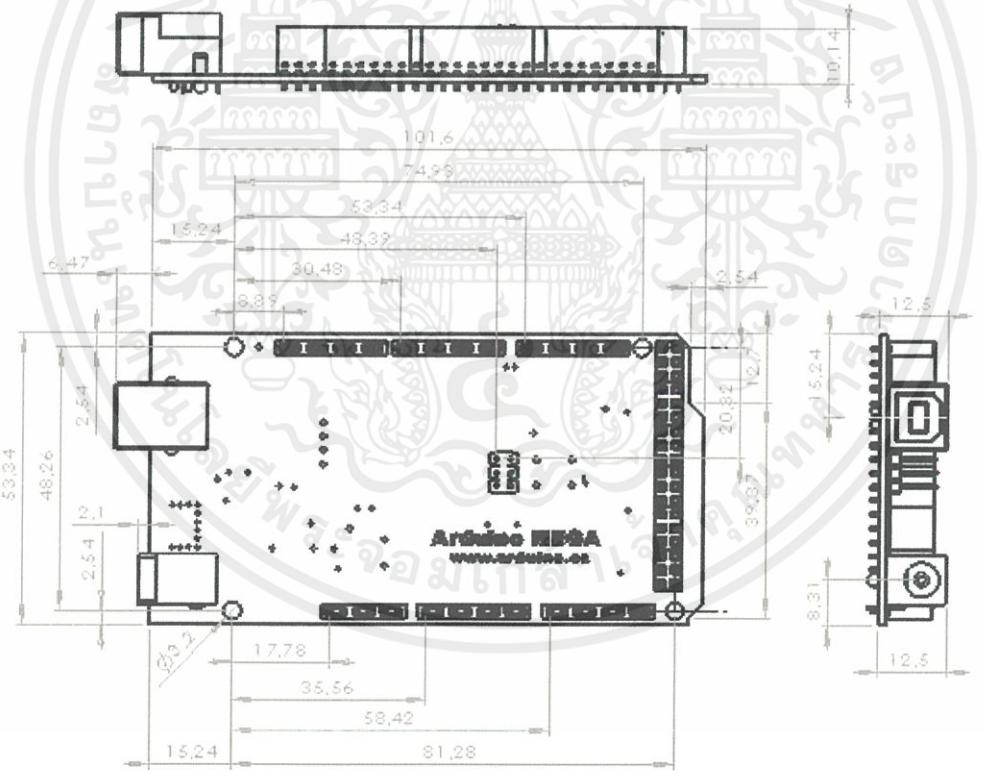
Now you have to go to
Tools>SerialPort
and select the right serial port, the one arduino is attached to.





Dimensioned Drawing





Terms & Conditions

1. Warranties

1.1 The producer warrants that its products will conform to the Specifications. This warranty lasts for one (1) years from the date of the sale. The producer shall not be liable for any defects that are caused by neglect, misuse or mistreatment by the Customer, including improper installation or testing, or for any products that have been altered or modified in any way by a Customer. Moreover, The producer shall not be liable for any defects that result from Customer's design, specifications or instructions for such products. Testing and other quality control techniques are used to the extent the producer deems necessary.

1.2 If any products fail to conform to the warranty set forth above, the producer's sole liability shall be to replace such products. The producer's liability shall be limited to products that are determined by the producer not to conform to such warranty. If the producer elects to replace such products, the producer shall have a reasonable time to replacements. Replaced products shall be warranted for a new full warranty period.

1.3 EXCEPT AS SET FORTH ABOVE, PRODUCTS ARE PROVIDED "AS IS" AND "WITH ALL FAULTS." THE PRODUCER DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE

1.4 Customer agrees that prior to using any systems that include the producer products, Customer will test such systems and the functionality of the products as used in such systems. The producer may provide technical, applications or design advice, quality characterization, reliability data or other services. Customer acknowledges and agrees that providing these services shall not expand or otherwise alter the producer's warranties, as set forth above, and no additional obligations or liabilities shall arise from the producer providing such services.

1.5 The Arduino™ products are not authorized for use in safety-critical applications where a failure of the product would reasonably be expected to cause severe personal injury or death. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Arduino™ products are neither designed nor intended for use in military or aerospace applications or environments and for automotive applications or environment. Customer acknowledges and agrees that any such use of Arduino™ products which is solely at the Customer's risk, and that Customer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

1.6 Customer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products and any use of Arduino™ products in Customer's applications, notwithstanding any applications-related information or support that may be provided by the producer.

2. Indemnification

The Customer acknowledges and agrees to defend, indemnify and hold harmless the producer from and against any and all third-party losses, damages, liabilities and expenses it incurs to the extent directly caused by: (i) an actual breach by a Customer of the representation and warranties made under this terms and conditions or (ii) the gross negligence or willful misconduct by the Customer.

3. Consequential Damages Waiver

In no event the producer shall be liable to the Customer or any third parties for any special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of the products provided hereunder, regardless of whether the producer has been advised of the possibility of such damages. This section will survive the termination of the warranty period.

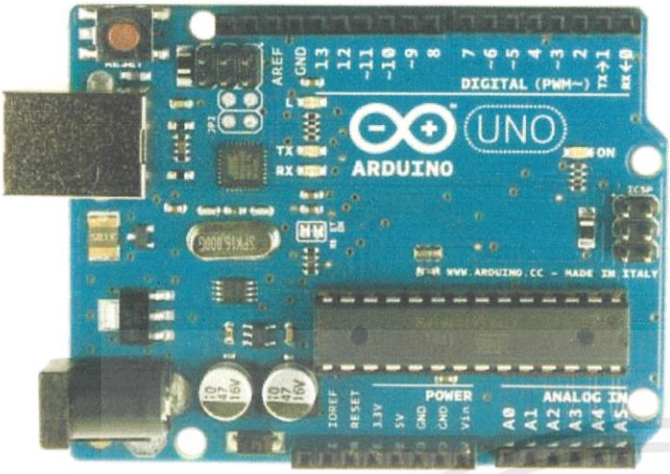
4. Changes to specifications

The producer may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." The producer reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.

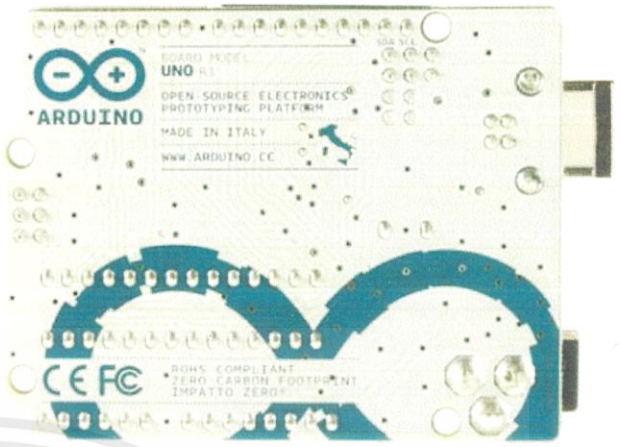
Enviromental Policies

The producer of Arduino™ has joined the Impatto Zero® policy of LifeGate.it. For each Arduino board produced is created / looked after half squared Km of Costa Rica's forest's.

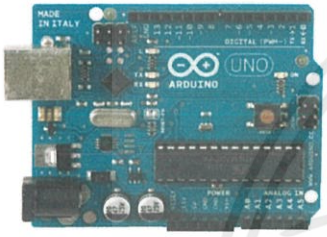
Arduino Uno



Arduino Uno R3 Front



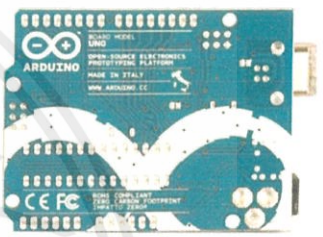
Arduino Uno R3 Back



Arduino Uno R2 Front

Arduino Uno SMD

Arduino Uno Front



Arduino Uno Back

Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.
The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.
Revision 2 of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode.
Revision 3 of the board has the following new features:

- 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V

Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Schematic & Reference Design

EAGLE files: arduino-uno-Rev3-reference-design.zip (NOTE: works with Eagle 6.0 and newer)
Schematic: arduino-uno-Rev3-schematic.pdf
Note: The Arduino reference design can use an Atmega8, 168, or 328, Current models use an ATmega328, but an Atmega8 is shown in the schematic for reference. The pin configuration is identical on all three processors.

Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.
External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.
The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.
The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.**This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

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 kOhms. 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. See the attachInterrupt() function for details.
- **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.

See also the mapping between Arduino pins and ATmega328 ports. The mapping for the Atmega8, 168, and 328 is identical.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).
A SoftwareSerial library allows for serial communication on any of the Uno's digital pins.
The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

Programming

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the reference and tutorials.
The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).
You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.
The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available . The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.
- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload. This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.
The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

A4988

DMOS Microstepping Driver with Translator and Overcurrent Protection

Features and Benefits

- Low $R_{DS(ON)}$ outputs
- Automatic current decay mode detection/selection
- Mixed and Slow current decay modes
- Synchronous rectification for low power dissipation
- Internal UVLO
- Crossover-current protection
- 3.3 and 5 V compatible logic supply
- Thermal shutdown circuitry
- Short-to-ground protection
- Shorted load protection
- Five selectable step modes: full, $^1/_2$, $^1/_4$, $^1/_8$, and $^1/_{16}$

Package:

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

Approximate size

Description

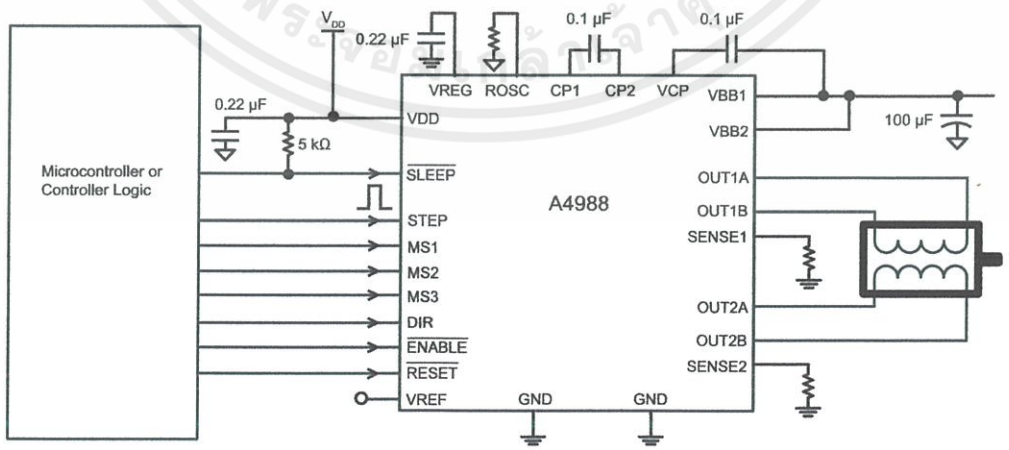
The A4988 is a complete microstepping motor driver with built-in translator for easy operation. It is designed to operate bipolar stepper motors in full-, half-, quarter-, eighth-, and sixteenth-step modes, with an output drive capacity of up to 35 V and ±2 A. The A4988 includes a fixed off-time current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

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

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

Continued on the next page...

Typical Application Diagram



Description (continued)

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

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

Selection Guide

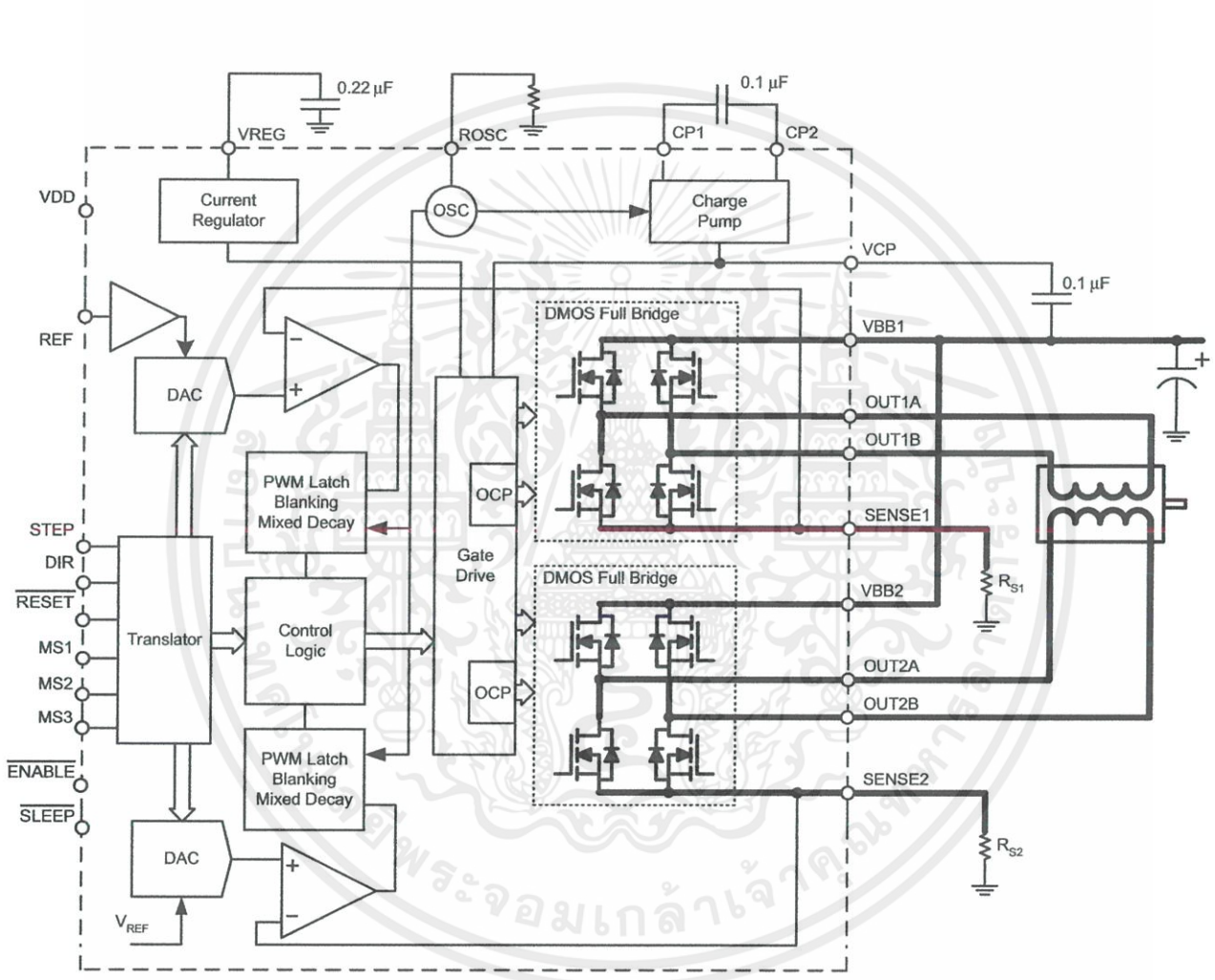
Part Number	Package	Packing
A4988SETTR-T	28-contact QFN with exposed thermal pad	1500 pieces per 7-in. reel

Absolute Maximum Ratings

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

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

Functional Block Diagram



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

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

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

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

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

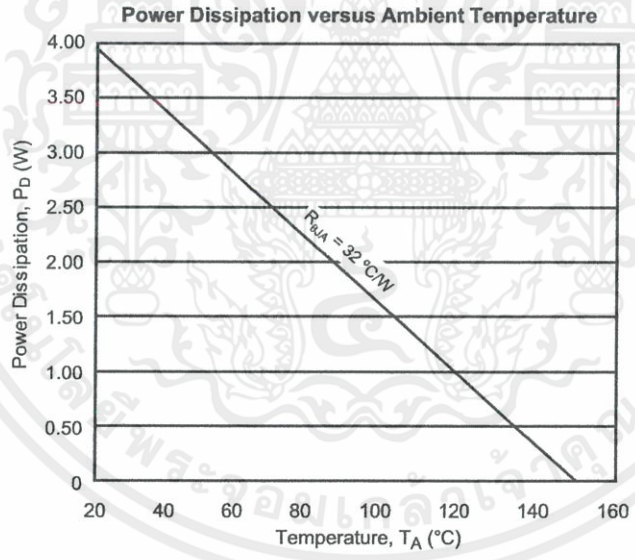
[3]V_{ERR} = [(V_{REF}/8) – V_{SENSE}] / (V_{REF}/8).

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

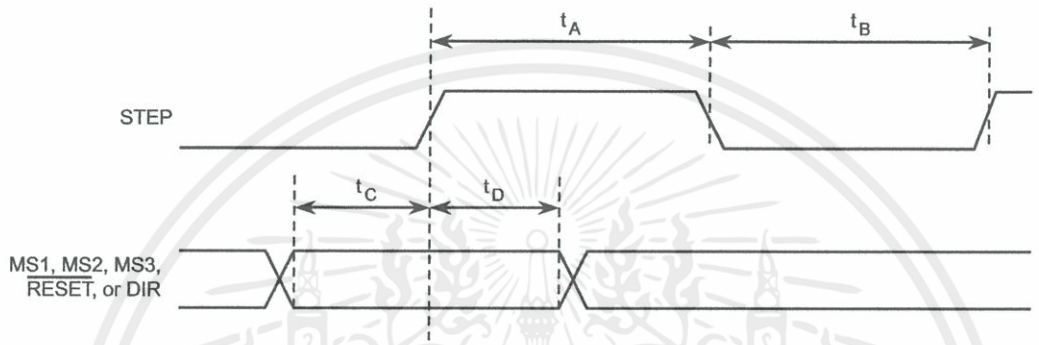
THERMAL CHARACTERISTICS

Characteristic	Symbol	Test Conditions*	Value	Units
Package Thermal Resistance	$R_{\theta JA}$	Four-layer PCB, based on JEDEC standard	32	°C/W

*Additional thermal information available on Allegro Web site.



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



Time Duration	Symbol	Typ.	Unit
STEP minimum, HIGH pulse width	t_A	1	µs
STEP minimum, LOW pulse width	t_B	1	µs
Setup time, input change to STEP	t_C	200	ns
Hold time, input change to STEP	t_D	200	ns

Figure 1. Logic Interface Timing Diagram

Table 1. Microstepping Resolution Truth Table

MS1	MS2	MS3	Microstep Resolution	Excitation Mode
L	L	L	Full Step	2 Phase
H	L	L	Half Step	1-2 Phase
L	H	L	Quarter Step	W1-2 Phase
H	H	L	Eighth Step	2W1-2 Phase
H	H	H	Sixteenth Step	4W1-2 Phase

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

Functional Description

Device Operation. The A4988 is a complete microstepping motor driver with a built-in translator for easy operation with minimal control lines. It is designed to operate bipolar stepper motors in full-, half-, quarter-, eighth, and sixteenth-step modes. The currents in each of the two output full-bridges and all of the N-channel DMOS FETs are regulated with fixed off-time PWM (pulse width modulated) control circuitry. At each step, the current for each full-bridge is set by the value of its external current-sense resistor (R_{S1} and R_{S2}), a reference voltage (V_{REF}), and the output voltage of its DAC (which in turn is controlled by the output of the translator).

At power-on or reset, the translator sets the DACs and the phase current polarity to the initial Home state (shown in figures 8 through 12), and the current regulator to Mixed Decay Mode for both phases. When a step command signal occurs on the STEP input, the translator automatically sequences the DACs to the next level and current polarity. (See table 2 for the current-level sequence.) The microstep resolution is set by the combined effect of the MSx inputs, as shown in table 1.

When stepping, if the new output levels of the DACs are lower than their previous output levels, then the decay mode for the active full-bridge is set to Mixed. If the new output levels of the DACs are higher than or equal to their previous levels, then the decay mode for the active full-bridge is set to Slow. This automatic current decay selection improves microstepping performance by reducing the distortion of the current waveform that results from the back EMF of the motor.

Microstep Select (MSx). The microstep resolution is set by the voltage on logic inputs MSx, as shown in table 1. The MS1 and MS3 pins have a 100 kΩ pull-down resistance, and the MS2 pin has a 50 kΩ pull-down resistance. When changing the step mode the change does not take effect until the next STEP rising edge.

If the step mode is changed without a translator reset, and absolute position must be maintained, it is important to change the step mode at a step position that is common to both step modes in order to avoid missing steps. When the device is powered down, or reset due to TSD or an over current event the translator is set to the home position which is by default common to all step modes.

Mixed Decay Operation. The bridge operates in Mixed decay mode, at power-on and reset, and during normal running according to the ROSC configuration and the step sequence, as shown in figures 8 through 12. During Mixed decay, when the trip point is reached, the A4988 initially goes into a fast decay mode for 31.25% of the off-time, t_{OFF}. After that, it switches to Slow decay mode for the remainder of t_{OFF}. A timing diagram for this feature appears on the next page.

Typically, mixed decay is only necessary when the current in the winding is going from a higher value to a lower value as determined by the state of the translator. For most loads automatically-selected mixed decay is convenient because it minimizes ripple when the current is rising and prevents missed steps when the current is falling. For some applications where microstepping at very low speeds is necessary, the lack of back EMF in the winding causes the current to increase in the load quickly, resulting in missed steps. This is shown in figure 2. By pulling the ROSC pin to ground, mixed decay is set to be active 100% of the time, for both rising and falling currents, and prevents missed steps as shown in figure 3. If this is not an issue, it is recommended that automatically-selected mixed decay be used, because it will produce reduced ripple currents. Refer to the Fixed Off-Time section for details.

Low Current Microstepping. Intended for applications where the minimum on-time prevents the output current from regulating to the programmed current level at low current steps. To prevent this, the device can be set to operate in Mixed decay mode on both rising and falling portions of the current waveform. This feature is implemented by shorting the ROSC pin to ground. In this state, the off-time is internally set to 30 µs.

Reset Input ($\overline{\text{RESET}}$). The $\overline{\text{RESET}}$ input sets the translator to a predefined Home state (shown in figures 8 through 12), and turns off all of the FET outputs. All STEP inputs are ignored until the $\overline{\text{RESET}}$ input is set to high.

Step Input (STEP). A low-to-high transition on the STEP input sequences the translator and advances the motor one increment. The translator controls the input to the DACs and the direc-

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

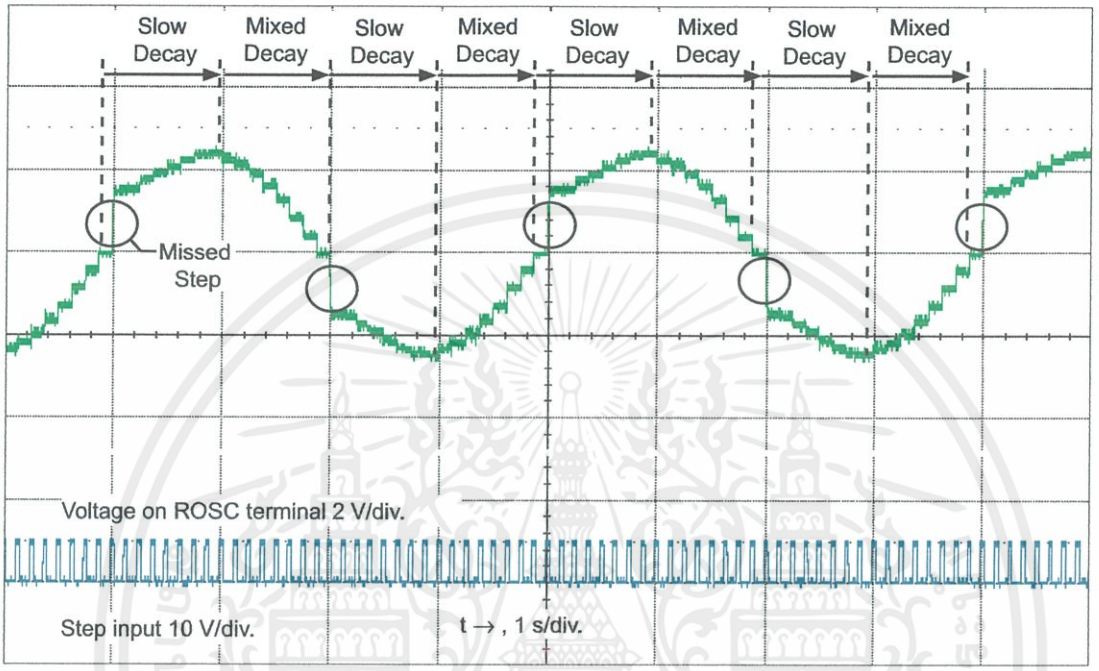


Figure 2. Missed steps in low-speed microstepping

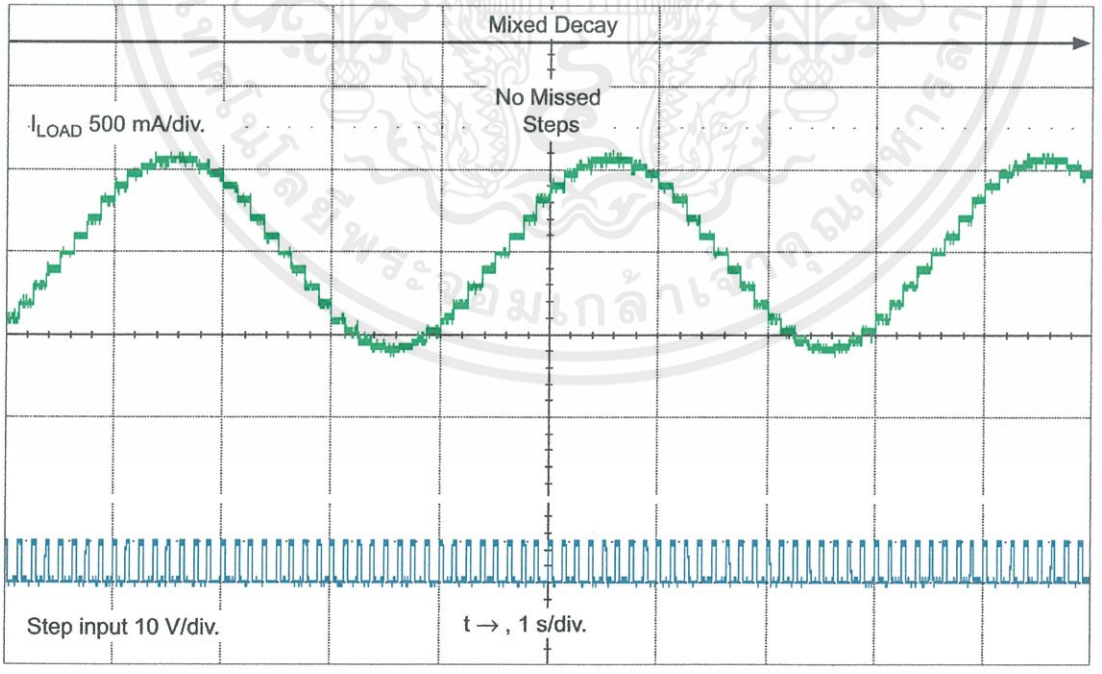


Figure 3. Continuous stepping using automatically-selected mixed stepping (ROSC pin grounded)

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

tion of current flow in each winding. The size of the increment is determined by the combined state of the MSx inputs.

Direction Input (DIR). This determines the direction of rotation of the motor. Changes to this input do not take effect until the next STEP rising edge.

Internal PWM Current Control. Each full-bridge is controlled by a fixed off-time PWM current control circuit that limits the load current to a desired value, I_{TRIP}. Initially, a diagonal pair of source and sink FET outputs are enabled and current flows through the motor winding and the current sense resistor, R_{Sx}. When the voltage across R_{Sx} equals the DAC output voltage, the current sense comparator resets the PWM latch. The latch then turns off the appropriate source driver and initiates a fixed off time decay mode

The maximum value of current limiting is set by the selection of R_{Sx} and the voltage at the VREF pin. The transconductance function is approximated by the maximum value of current limiting, $I_{TripMAX}$ (A), which is set by

$$I_{TripMAX} = V_{REF} / (8 \times R_S)$$

where R_S is the resistance of the sense resistor (Ω) and V_{REF} is the input voltage on the REF pin (V).

The DAC output reduces the V_{REF} output to the current sense comparator in precise steps, such that

$$I_{trip} = (\%I_{TripMAX} / 100) \times I_{TripMAX}$$

(See table 2 for $\%I_{TripMAX}$ at each step.)

It is critical that the maximum rating (0.5 V) on the SENSE1 and SENSE2 pins is not exceeded.

Fixed Off-Time. The internal PWM current control circuitry uses a one-shot circuit to control the duration of time that the DMOS FETs remain off. The off-time, t_{OFF}, is determined by the ROSC terminal. The ROSC terminal has three settings:

- ROSC tied to VDD — off-time internally set to 30 µs, decay mode is automatic Mixed decay except when in full step where decay mode is set to Slow decay
- ROSC tied directly to ground — off-time internally set to 30 µs, current decay is set to Mixed decay for both increasing and decreasing currents, except in full step where decay mode is set to Slow decay. (See Low Current Microstepping section.)
- ROSC through a resistor to ground — off-time is determined by the following formula, the decay mode is automatic Mixed decay for all step modes.

$$t_{OFF} \approx R_{OSC} / 825$$

Where t_{OFF} is in µs.

Blanking. This function blanks the output of the current sense comparators when the outputs are switched by the internal current control circuitry. The comparator outputs are blanked to prevent false overcurrent detection due to reverse recovery currents of the clamp diodes, and switching transients related to the capacitance of the load. The blank time, t_{BLANK} (µs), is approximately

$$t_{BLANK} \approx 1 \text{ µs}$$

Shorted-Load and Short-to-Ground Protection. If the motor leads are shorted together, or if one of the leads is shorted to ground, the driver will protect itself by sensing the overcurrent event and disabling the driver that is shorted, protecting the device from damage. In the case of a short-to-ground, the device will remain disabled (latched) until the $\overline{\text{SLEEP}}$ input goes high or VDD power is removed. A short-to-ground overcurrent event is shown in figure 4.

When the two outputs are shorted together, the current path is through the sense resistor. After the blanking time (≈1 µs) expires, the sense resistor voltage is exceeding its trip value, due to the overcurrent condition that exists. This causes the driver to go into a fixed off-time cycle. After the fixed off-time expires the driver turns on again and the process repeats. In this condition the driver is completely protected against overcurrent events, but the short is repetitive with a period equal to the fixed off-time of the driver. This condition is shown in figure 5.

During a shorted load event it is normal to observe both a positive and negative current spike as shown in figure 3, due to the direction change implemented by the Mixed decay feature. This is shown in figure 6. In both instances the overcurrent circuitry is protecting the driver and prevents damage to the device.

Charge Pump (CP1 and CP2). The charge pump is used to generate a gate supply greater than that of VBB for driving the source-side FET gates. A 0.1 µF ceramic capacitor, should be connected between CP1 and CP2. In addition, a 0.1 µF ceramic capacitor is required between VCP and VBB, to act as a reservoir for operating the high-side FET gates.

Capacitor values should be Class 2 dielectric ±15% maximum, or tolerance R, according to EIA (Electronic Industries Alliance) specifications.

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

V_{REG} (VREG). This internally-generated voltage is used to operate the sink-side FET outputs. The VREG pin must be decoupled with a 0.22 µF ceramic capacitor to ground. V_{REG} is internally monitored. In the case of a fault condition, the FET outputs of the A4988 are disabled.

Capacitor values should be Class 2 dielectric ±15% maximum, or tolerance R, according to EIA (Electronic Industries Alliance) specifications.

Enable Input (ENABLE). This input turns on or off all of the FET outputs. When set to a logic high, the outputs are disabled. When set to a logic low, the internal control enables the outputs as required. The translator inputs STEP, DIR, and MSx, as well as the internal sequencing logic, all remain active, independent of the ENABLE input state.

Shutdown. In the event of a fault, overtemperature (excess T_J) or an undervoltage (on VCP), the FET outputs of the A4988 are disabled until the fault condition is removed. At power-on, the UVLO (undervoltage lockout) circuit disables the FET outputs and resets the translator to the Home state.

Sleep Mode (SLEEP). To minimize power consumption when the motor is not in use, this input disables much of the internal circuitry including the output FETs, current regulator, and charge pump. A logic low on the SLEEP pin puts the A4988 into Sleep mode. A logic high allows normal operation, as well as start-up (at which time the A4988 drives the motor to the Home microstep position). When emerging from Sleep mode, in order to allow the charge pump to stabilize, provide a delay of 1 ms before issuing a Step command.

Mixed Decay Operation. The bridge operates in Mixed Decay mode, depending on the step sequence, as shown in figures 8 through 12. As the trip point is reached, the A4988 initially goes into a fast decay mode for 31.25% of the off-time, t_{OFF}. After that, it switches to Slow Decay mode for the remainder of t_{OFF}. A timing diagram for this feature appears in figure 7.

Synchronous Rectification. When a PWM-off cycle is triggered by an internal fixed-off time cycle, load current recirculates according to the decay mode selected by the control logic. This synchronous rectification feature turns on the appropriate FETs during current decay, and effectively shorts out the body diodes with the low FET $R_{DS(ON)}$. This reduces power dissipation significantly, and can eliminate the need for external Schottky diodes in many applications. Synchronous rectification turns off when the load current approaches zero (0 A), preventing reversal of the load current.

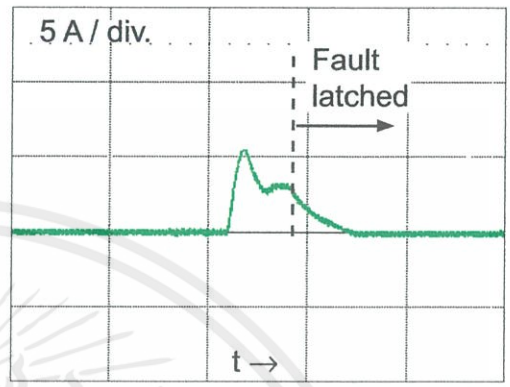


Figure 4. Short-to-ground event

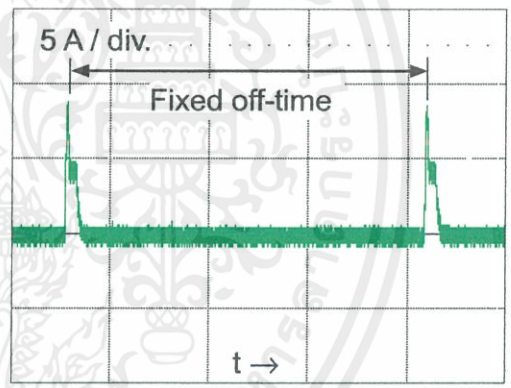


Figure 5. Shorted load (OUTxA → OUTxB) in Slow decay mode

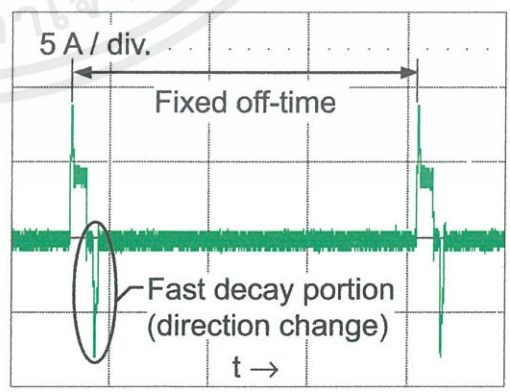
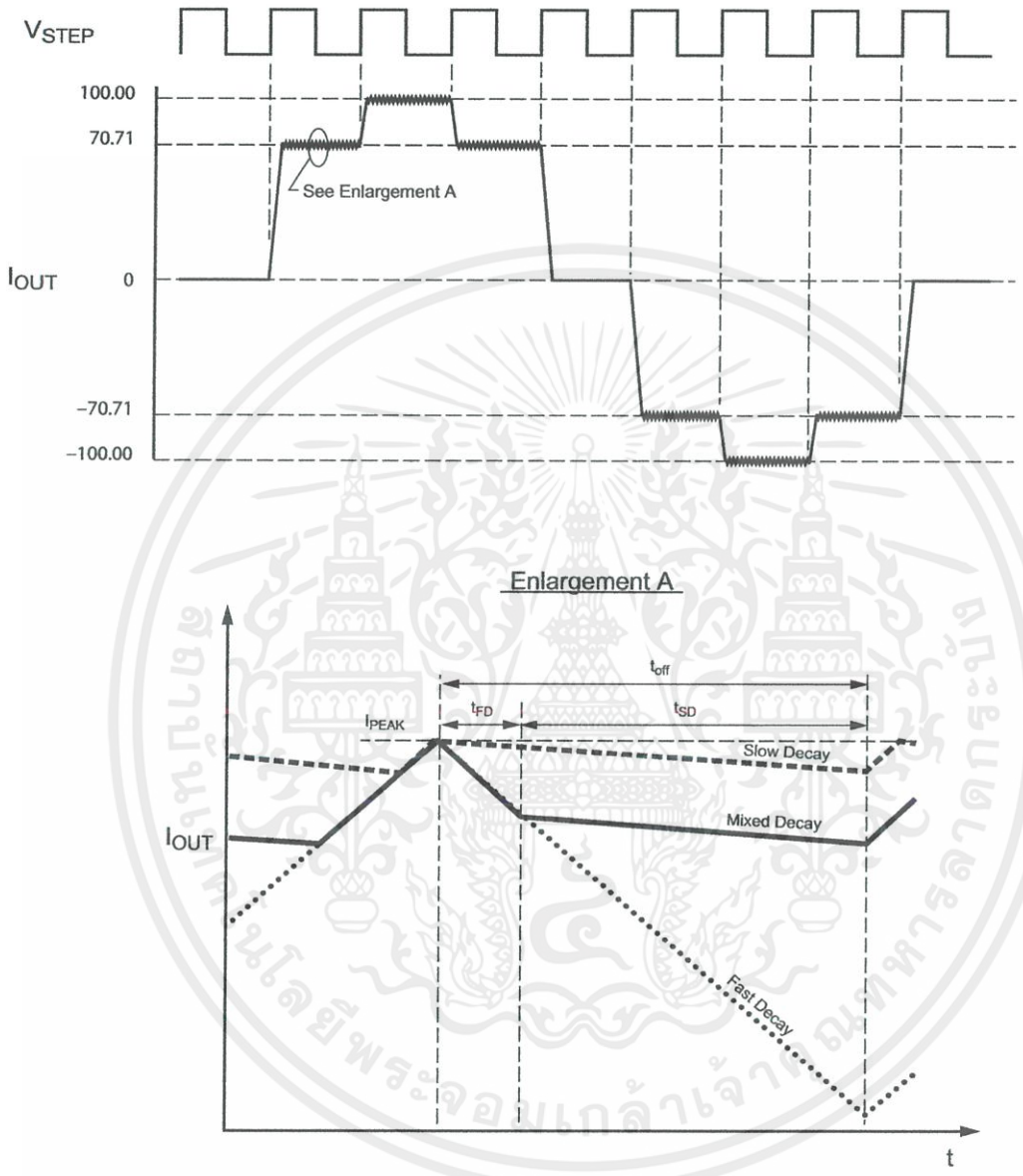


Figure 6. Shorted load (OUTxA → OUTxB) in Mixed decay mode

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



Symbol	Characteristic
t_{off}	Device fixed off-time
I_{PEAK}	Maximum output current
t_{SD}	Slow decay interval
t_{FD}	Fast decay interval
I_{OUT}	Device output current

Figure 7. Current Decay Modes Timing Chart

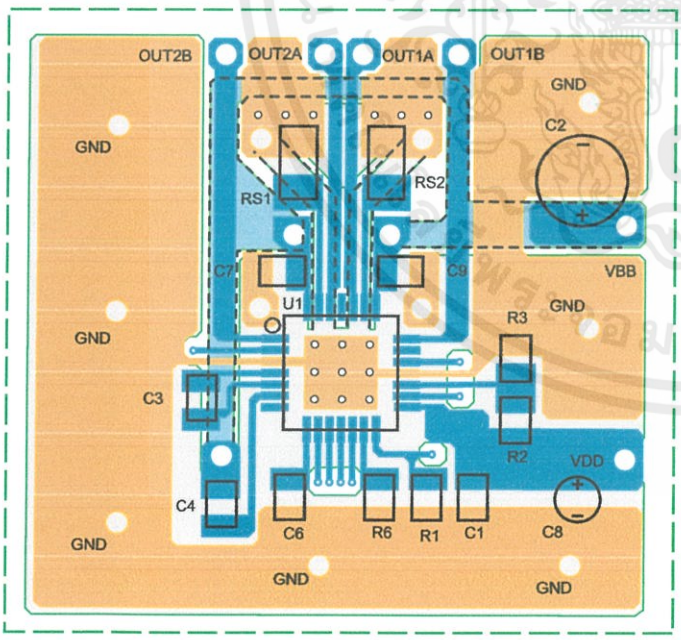
Allegro MicroSystems, Inc.
115 Northeast Cutoff
Worcester, Massachusetts 01615-0036 U.S.A.
1.508.853.5000; www.allegromicro.com

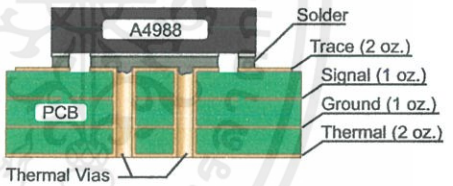
Application Layout

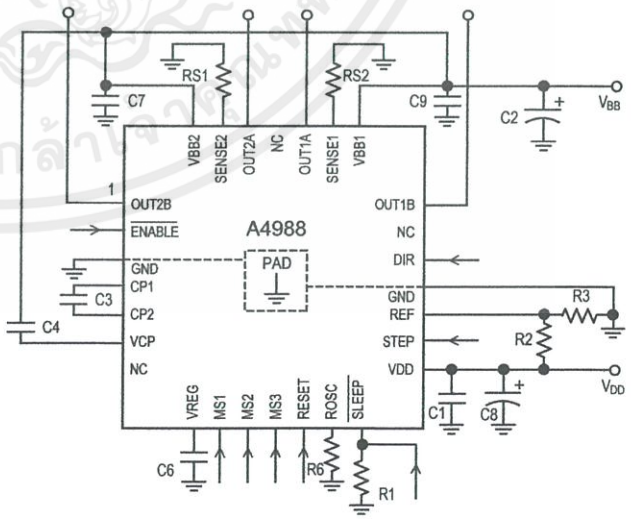
Layout. The printed circuit board should use a heavy ground-plane. For optimum electrical and thermal performance, the A4988 must be soldered directly onto the board. Pins 6, 7, 18, and 19 are internally fused, which provides a path for enhanced thermal dissipation. Theses pins should be soldered directly to an exposed surface on the PCB that connects to thermal vias are used to transfer heat to other layers of the PCB.

In order to minimize the effects of ground bounce and offset issues, it is important to have a low impedance single-point ground, known as a *star ground*, located very close to the device. By making the connection between the pad and the ground plane directly under the A4988, that area becomes an ideal location for a star ground point. A low impedance ground will prevent ground bounce during high current operation and ensure that the supply voltage remains stable at the input terminal.

The two input capacitors should be placed in parallel, and as close to the device supply pins as possible. The ceramic capacitor (CIN1) should be closer to the pins than the bulk capacitor (CIN2). This is necessary because the ceramic capacitor will be responsible for delivering the high frequency current components. The sense resistors, RSx, should have a very low impedance path to ground, because they must carry a large current while supporting very accurate voltage measurements by the current sense comparators. Long ground traces will cause additional voltage drops, adversely affecting the ability of the comparators to accurately measure the current in the windings. The SENSEx pins have very short traces to the RSx resistors and very thick, low impedance traces directly to the star ground underneath the device. If possible, there should be no other components on the sense circuits.

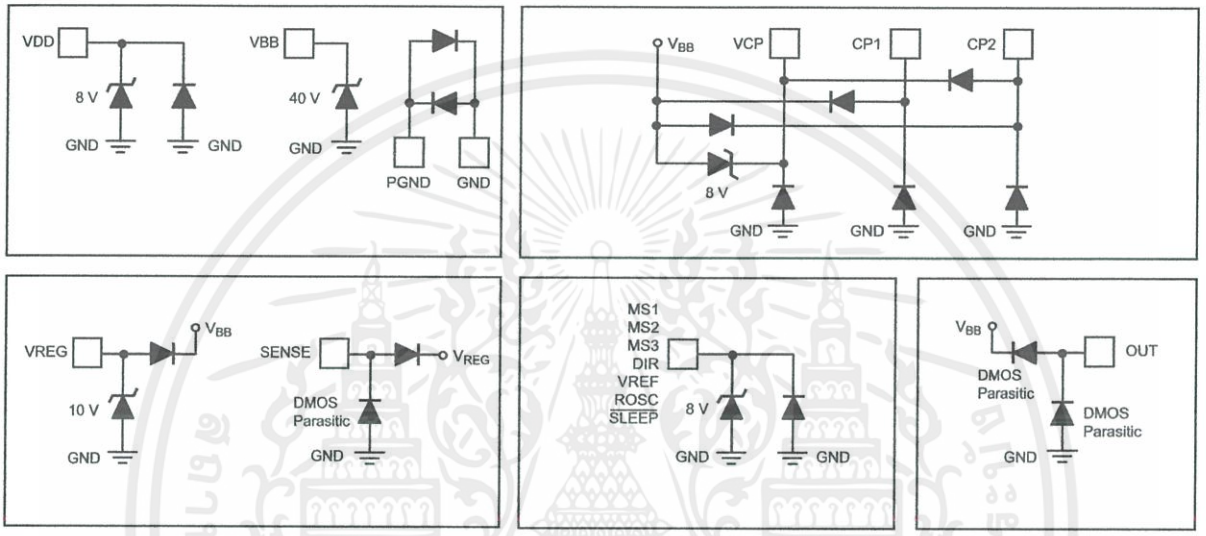






Allegro MicroSystems, Inc.
115 Northeast Cutoff
Worcester, Massachusetts 01615-0036 U.S.A.
1.508.853.5000; www.allegromicro.com

Pin Circuit Diagrams



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

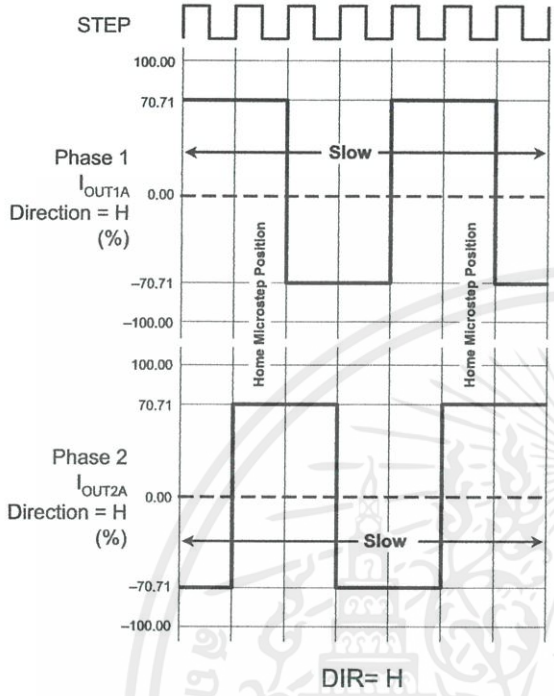


Figure 8. Decay Mode for Full-Step Increments

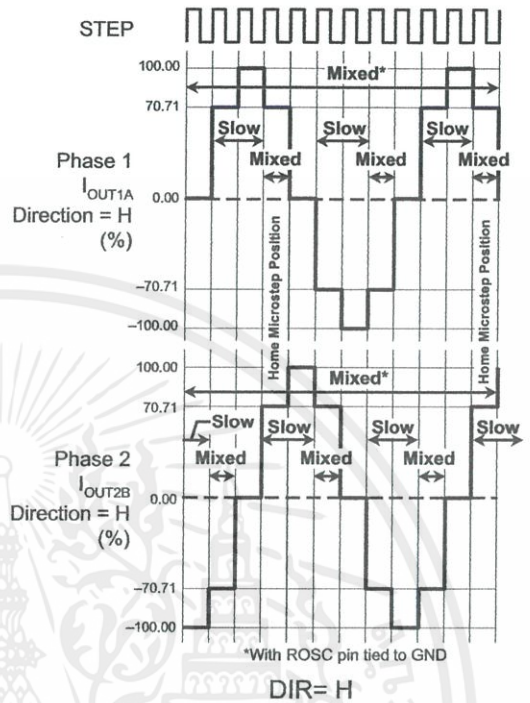


Figure 9. Decay Modes for Half-Step Increments

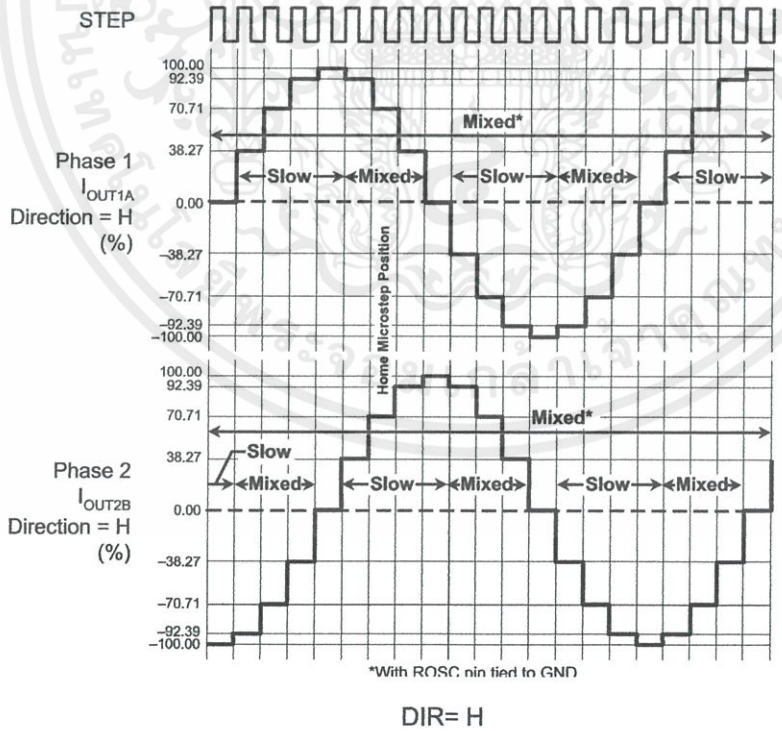


Figure 10. Decay Modes for Quarter-Step Increments

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

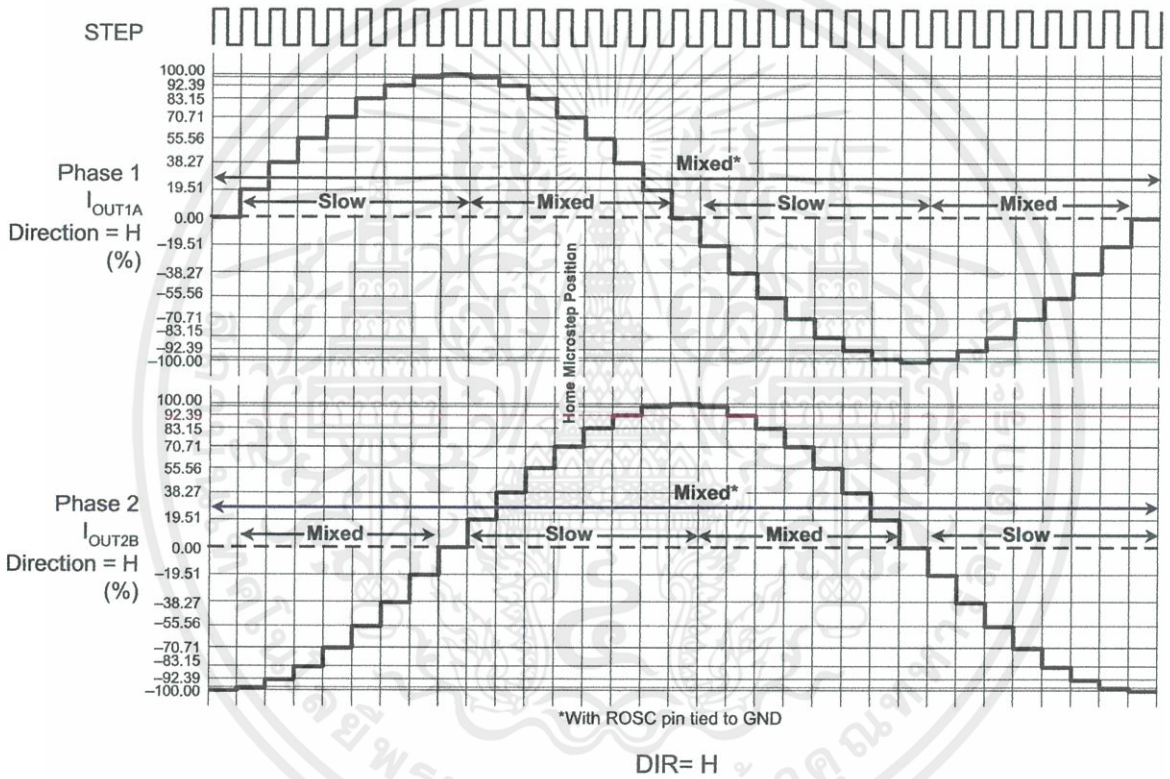


Figure 11. Decay Modes for Eighth-Step Increments

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

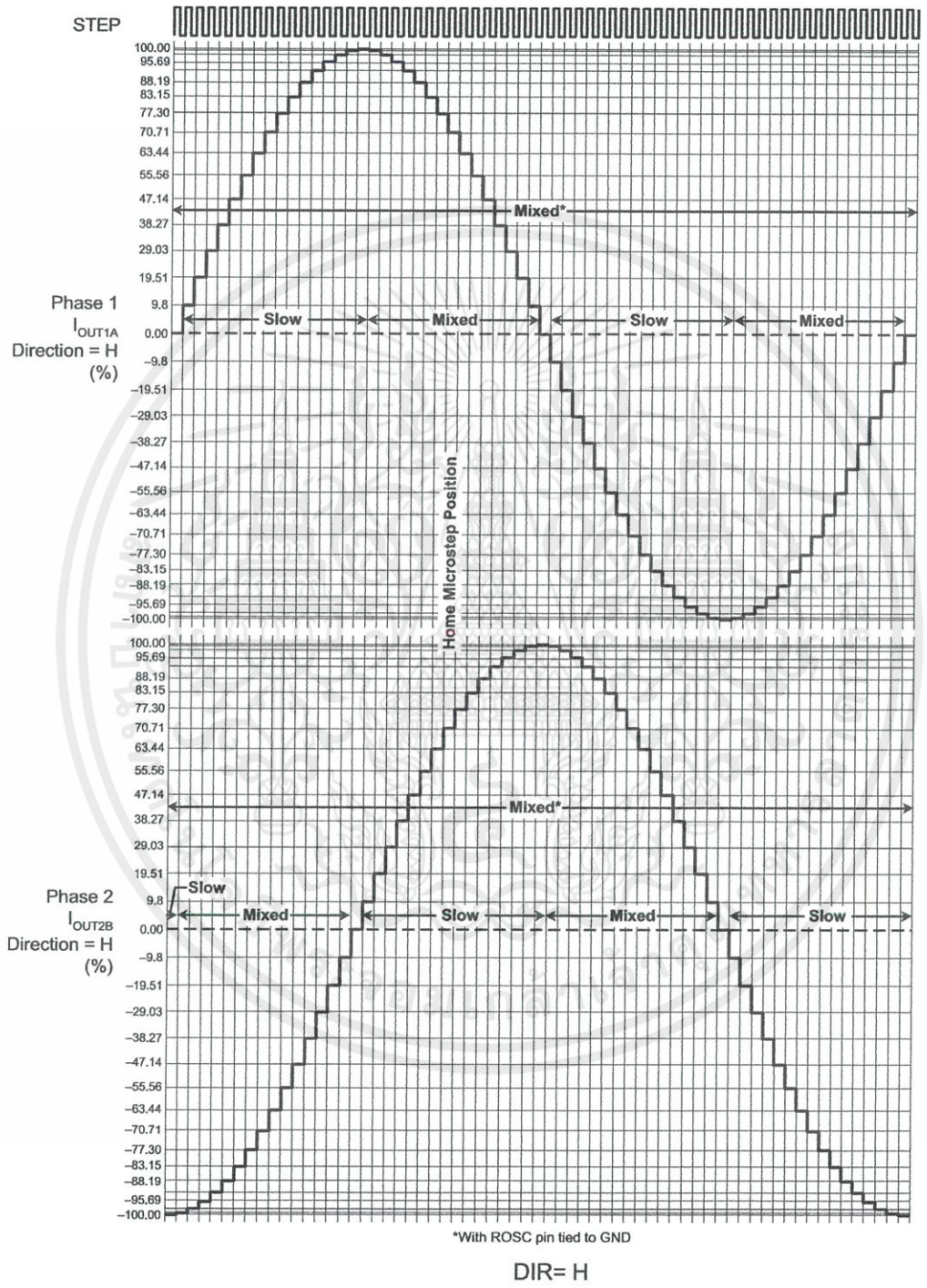


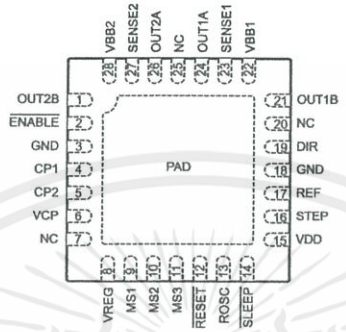
Figure 12. Decay Modes for Sixteenth-Step Increments

Table 2. Step Sequencing Settings

Home microstep position at Step Angle 45°; DIR = H

Full Step #	Half Step #	1/4 Step #	1/8 Step #	1/16 Step #	Phase 1 Current [% $I_{tripMax}$] (%)	Phase 2 Current [% $I_{tripMax}$] (%)	Step Angle (°)	Full Step #	Half Step #	1/4 Step #	1/8 Step #	1/16 Step #	Phase 1 Current [% $I_{tripMax}$] (%)	Phase 2 Current [% $I_{tripMax}$] (%)	Step Angle (°)
	1	1	2	1	100.00	0.00	0.0		5	9	17	33	–100.00	0.00	180.0
				2	99.52	9.80	5.6					34	–99.52	–9.80	185.6
			2	3	98.08	19.51	11.3				18	35	–98.08	–19.51	191.3
				4	95.69	29.03	16.9					36	–95.69	–29.03	196.9
		2	3	5	92.39	38.27	22.5			10	19	37	–92.39	–38.27	202.5
				6	88.19	47.14	28.1					38	–88.19	–47.14	208.1
			4	7	83.15	55.56	33.8				20	39	–83.15	–55.56	213.8
				8	77.30	63.44	39.4					40	–77.30	–63.44	219.4
1	2	3	5	9	70.71	70.71	45.0	3	6	11	21	41	–70.71	–70.71	225.0
				10	63.44	77.30	50.6					42	–63.44	–77.30	230.6
			6	11	55.56	83.15	56.3				22	43	–55.56	–83.15	236.3
				12	47.14	88.19	61.9					44	–47.14	–88.19	241.9
		4	7	13	38.27	92.39	67.5			12	23	45	–38.27	–92.39	247.5
				14	29.03	95.69	73.1					46	–29.03	–95.69	253.1
			8	15	19.51	98.08	78.8				24	47	–19.51	–98.08	258.8
				16	9.80	99.52	84.4					48	–9.80	–99.52	264.4
	3	5	9	17	0.00	100.00	90.0		7	13	25	49	0.00	–100.00	270.0
				18	–9.80	99.52	95.6					50	9.80	–99.52	275.6
			10	19	–19.51	98.08	101.3				26	51	19.51	–98.08	281.3
				20	–29.03	95.69	106.9					52	29.03	–95.69	286.9
		6	11	21	–38.27	92.39	112.5			14	27	53	38.27	–92.39	292.5
				22	–47.14	88.19	118.1					54	47.14	–88.19	298.1
			12	23	–55.56	83.15	123.8				28	55	55.56	–83.15	303.8
				24	–63.44	77.30	129.4					56	63.44	–77.30	309.4
2	4	7	13	25	–70.71	70.71	135.0	4	8	15	29	57	70.71	–70.71	315.0
				26	–77.30	63.44	140.6					58	77.30	–63.44	320.6
			14	27	–83.15	55.56	146.3				30	59	83.15	–55.56	326.3
				28	–88.19	47.14	151.9					60	88.19	–47.14	331.9
		8	15	29	–92.39	38.27	157.5			16	31	61	92.39	–38.27	337.5
				30	–95.69	29.03	163.1					62	95.69	–29.03	343.1
			16	31	–98.08	19.51	168.8				32	63	98.08	–19.51	348.8
				32	–99.52	9.80	174.4					64	99.52	–9.80	354.4

Pin-out Diagram



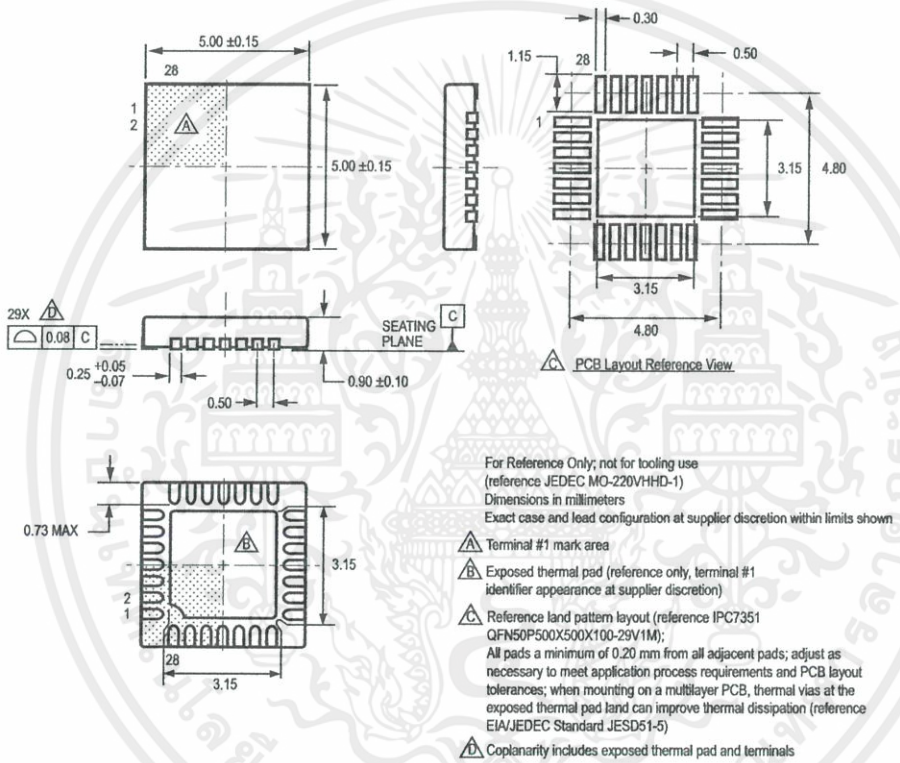
Terminal List Table

Name	Number	Description
CP1	4	Charge pump capacitor terminal
CP2	5	Charge pump capacitor terminal
VCP	6	Reservoir capacitor terminal
VREG	8	Regulator decoupling terminal
MS1	9	Logic input
MS2	10	Logic input
MS3	11	Logic input
$\overline{\text{RESET}}$	12	Logic input
ROSC	13	Timing set
$\overline{\text{SLEEP}}$	14	Logic input
VDD	15	Logic supply
STEP	16	Logic input
REF	17	G_m reference voltage input
GND	3, 18	Ground*
DIR	19	Logic input
OUT1B	21	DMOS Full Bridge 1 Output B
VBB1	22	Load supply
SENSE1	23	Sense resistor terminal for Bridge 1
OUT1A	24	DMOS Full Bridge 1 Output A
OUT2A	26	DMOS Full Bridge 2 Output A
SENSE2	27	Sense resistor terminal for Bridge 2
VBB2	28	Load supply
OUT2B	1	DMOS Full Bridge 2 Output B
$\overline{\text{ENABLE}}$	2	Logic input
NC	7, 20, 25	No connection
PAD	–	Exposed pad for enhanced thermal dissipation*

*The GND pins must be tied together externally by connecting to the PAD ground plane under the device.

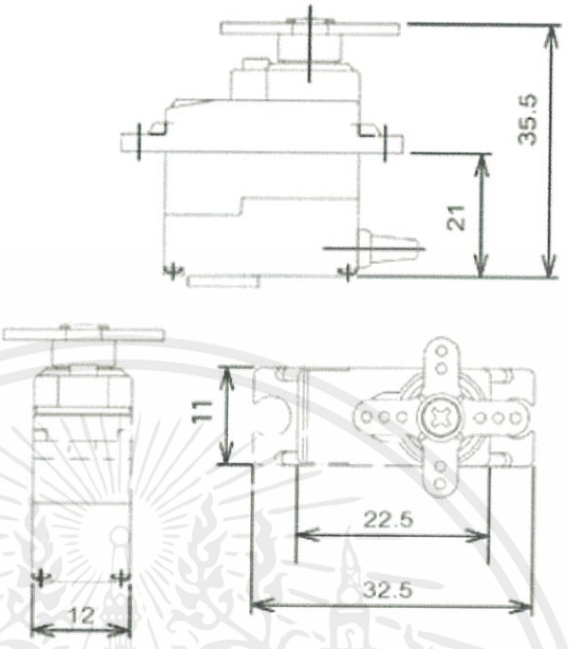
Allegro MicroSystems, Inc.
115 Northeast Cutoff
Worcester, Massachusetts 01615-0036 U.S.A.
1.508.853.5000; www.allegromicro.com

ET Package, 28-Pin QFN with Exposed Thermal Pad



For the latest version of this document, visit our website:
www.allegromicro.com

MG90S
Metal Gear Servo



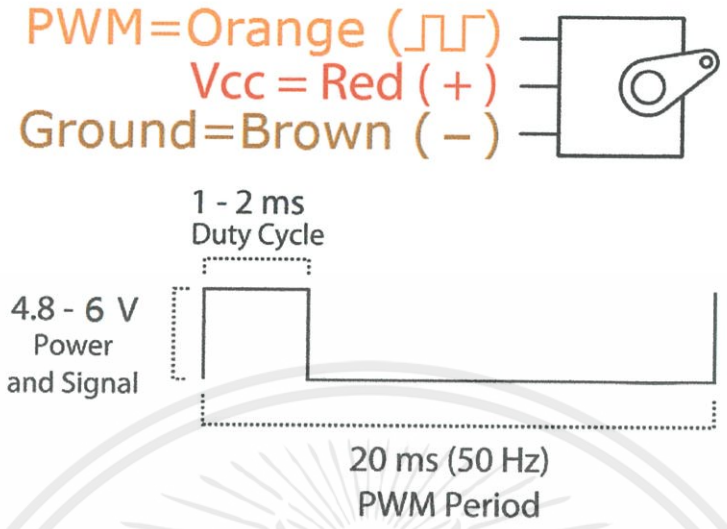
MG90S servo, Metal gear with one bearing

Tiny and lightweight with high output power, this tiny servo is perfect for RC Airplane, Helicopter, Quadcopter or Robot. This servo has *metal gears* for added strength and durability.

Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but *smaller*. You can use any servo code, hardware or library to control these servos. Good 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. It comes with a 3 horns (arms) and hardware.

Specifications

- Weight: 13.4 g
- Dimension: 22.5 x 12 x 35.5 mm approx.
- Stall torque: 1.8 kgf·cm (4.8V), 2.2 kgf·cm (6 V)
- Operating speed: 0.1 s/60 degree (4.8 V), 0.08 s/60 degree (6 V)
- Operating voltage: 4.8 V - 6.0 V
- Dead band width: 5 μs



Position "0" (1.5 ms pulse) is middle, "90" (~2 ms pulse) is all the way to the right, "-90" (~1 ms pulse) is all the way to the left.