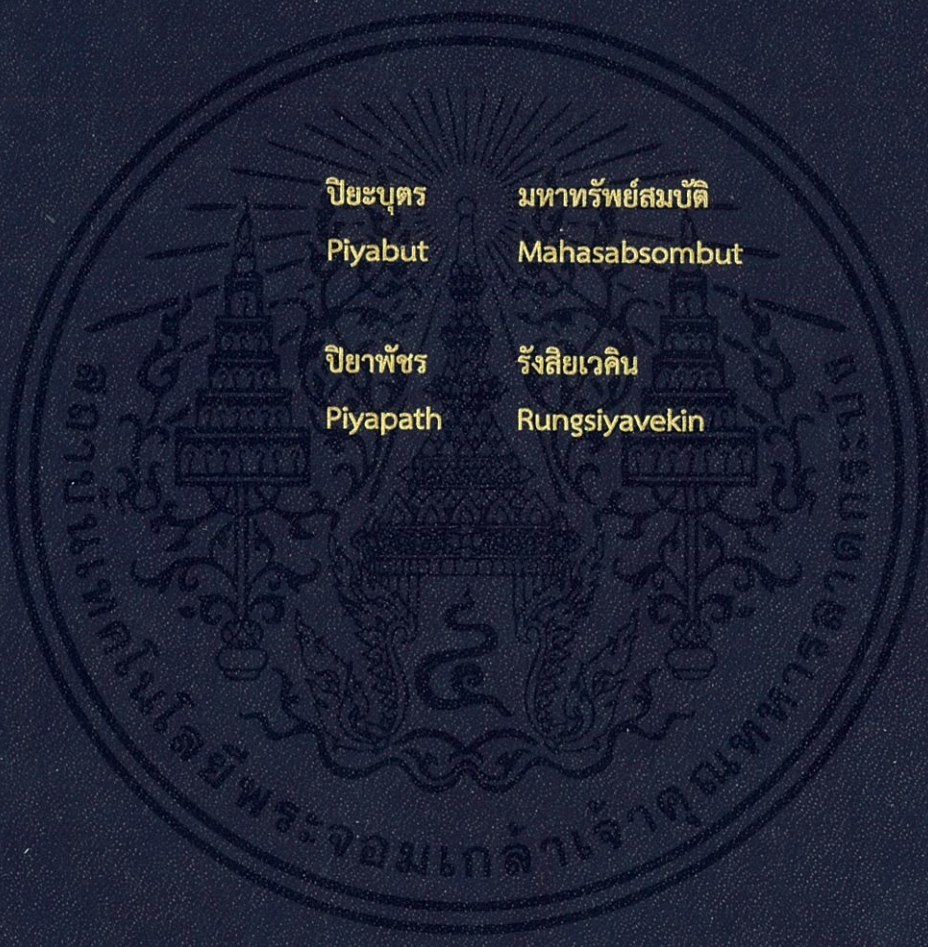
หุ่นยนต์สำรวจ

SURVEY ROBOT

ปิยะบุตร มหาทรัพย์สมบัติ
Piyabut Mahasabsombut

ปิยาพัชร รังสิยเวคิน
Piyapath Rungsiyavekin



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2561

หุ่นยนต์สำรวจ

SURVEY ROBOT

ปิยะบุตร มหาทรัพย์สมบัติ
Piyabut Mahasabsombut

ปิยาพัชร รังสิยเวคิน
Piyapath Rungsiyavekin

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2561

หุ่นยนต์สำรวจ

SURVEY ROBOT

โดย

ปิยะบุตร มหาทรัพย์สมบัติ

ปิยาพัชร รังสิยเวคิน

อาจารยที่ปรึกษา

ผศ.ดร.แสงระวี บัวแก้ว

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2561

ปริญญานิพนธ์ปีการศึกษา 2561

ภาควิชา วิศวกรรมอิเล็กทรอนิกส์

คณะ วิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาลาดกระบัง

เรื่อง หุ่นยนต์สำรวจ

SURVEY ROBOT

ผู้จัดทำ นาย ปิยะบุตร มหาทรัพย์สมบัติ รหัสประจำตัว 58010793

นางสาว ปิยาพัชร รังสิยเวคิน รหัสประจำตัว 58010796

ปริญญานิพนธ์นี้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

(ผศ.ดร.แสงระวี บัวแก้ว)
อาจารย์ที่ปรึกษา

หัวข้อปริญญานิพนธ์ หุ่นยนต์สำรวจ

นักศึกษา นาย ปิยะบุตร มหาทรัพย์สมบัติ รหัสประจำตัว 58010793

นางสาว ปิยาพัชร รังสิยเวคิน รหัสประจำตัว 58010796

ปริญญา วิศวกรรมศาสตรบัณฑิต

ภาควิชา วิศวกรรมอิเล็กทรอนิกส์

ปีการศึกษา 2561

อาจารย์ที่ปรึกษาปริญญานิพนธ์ ผศ.ดร.แสงระวี บัวแก้ว

บทคัดย่อ

โครงงานนี้นำเสนอการออกแบบและการสร้างหุ่นยนต์ที่สามารถสำรวจได้ทุกสภาพพื้นผิว บริเวณที่ซึ่งมีความเสี่ยงต่อการสำรวจของมนุษย์หรือที่ซึ่งมนุษย์เข้าถึงได้ยาก โดยหุ่นยนต์สำรวจที่ประดิษฐ์ขึ้นนั้นอาศัยการทำงานของไมโครคอนโทรลเลอร์ในการประมวลผลสัญญาณ สามารถควบคุมการทำงานได้ในระยะไกลผ่านการควบคุมของวิทยุบังคับไร้สาย โครงสร้างของหุ่นยนต์สำรวจที่สร้างขึ้นนี้ ประกอบด้วยสามส่วนสำคัญ คือ การขับเคลื่อนหุ่นยนต์ซึ่งประกอบด้วย มอเตอร์ไฟฟ้ากระแสตรง และบอร์ดไดร์มอเตอร์ ส่วนที่สองประกอบด้วยรีโมทไร้สายและตัวรับสัญญาณเพื่อควบคุมการทำงานของหุ่นยนต์ และส่วนสุดท้ายเป็นภาคจ่ายไฟ เมื่อตัวรับสัญญาณรับข้อมูลแล้ว ข้อมูลนั้นจะส่งไปยังไมโครคอนโทรลเลอร์เพื่อประมวลผลและสั่งการให้ทำงานภายใต้อัลกอริทึมที่ออกแบบไว้

Thesis Title	Survey Robot		
Student	Mr. Piyabut Mahasabsombut	Student ID	58010793
	Miss Piyapath Rungsiyavekin	Student ID	58010796
Degree	Bachelor of Engineering		
Program	Electronics Engineering		
Year	2018		
Thesis Advisor	Asst.Prof.Dr. Sangrawee Buakaew		

ABSTRACT

This project presents the design and construction of robots that can be surveyed for any surface condition. A risk area is difficult for human to explore. The present robot is based on the operation of the microcontroller using signal processing for controlling through wireless remote control. The structure of this survey robot consists of three parts: The first part is the robot driven, which consists of DC Motor and the drive motor. The second part consists of a wireless remote and a receiver to control the robot and the last part is the power supply. When the receiver receives the information, the data will be sent to the microcontroller for processing and working under the designed algorithm

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

รูปที่ หน้า

กิตติกรรมประกาศ

โครงงานเรื่องหุ่นยนต์สำรวจ สำเร็จลุล่วงได้ด้วยความกรุณาของอาจารย์ที่ปรึกษาโครงงานผศ.ดร.แสงระวี บัวแก้ว และอาจารย์ในภาควิชาวิศวกรรมอิเล็กทรอนิกส์ ที่ให้คำปรึกษาแนะนำในการศึกษาค้นคว้า แนะนำขั้นตอนและวิธีจัดทำโครงงานจนสำเร็จลุล่วงด้วยดี คณะผู้จัดทำจึงขอกราบขอบพระคุณเป็นอย่างสูงไว้ ณ โอกาสนี้

ขอกราบขอบพระคุณ บิดา มารดาของคณะผู้จัดทำที่คอยเป็นกำลังใจและเป็นผู้สนับสนุนเงินทุนหลักในการทำโครงงาน รวมไปถึงเพื่อนๆผู้เป็นกำลังใจหลักคอยช่วยเหลือให้คำแนะนำ รวมทั้งช่วยกันในการแก้ปัญหาที่เกิดขึ้นในระหว่างการทำโครงงานนี้ทั้งหมดทำให้ผลของโครงงานสำเร็จ ลุล่วงไปได้ด้วยดี สุดท้ายนี้ คณะผู้จัดทำหวังว่าโครงงานนี้จะเป็นประโยชน์สำหรับผู้สนใจและผู้นำผลงานนี้ไปใช้งานได้

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

ในปัจจุบันเทคโนโลยีได้ก้าวหน้าโดยการพัฒนาจากการทำงานของมนุษย์ไปเป็นการประดิษฐ์หุ่นยนต์เพื่อช่วยให้มนุษย์สะดวกสบายมากขึ้น โดยอาศัยการทำงานร่วมกันระหว่างมนุษย์และหุ่นยนต์ ทั้งสองจะทำงานและติดต่อสื่อสารร่วมกันโดยตรง มนุษย์มีหน้าที่ควบคุม และหุ่นยนต์ทำหน้าที่ภาคปฏิบัติด้วยพละกำลังกายภาพ ซึ่งในปัจจุบันนั้นได้เกิดเหตุการณ์ที่มนุษย์ไม่สามารถเข้าไปปฏิบัติในส่วนงานนั้นหรืออาจจะทำให้เกิดอันตรายต่อผู้ปฏิบัติได้ เช่น ภัยธรรมชาติ สถานที่ที่มนุษย์ยากที่จะเข้าไป ฯลฯ ดังนั้นหุ่นยนต์สำรวจจึงมีความสำคัญที่เข้ามามีบทบาทในการช่วยลดความเสี่ยงที่จะเกิดความเสียหายได้

1.2วัตถุประสงค์

1.2.1. เพื่อสร้างหุ่นยนต์ไปปฏิบัติงานในบริเวณที่มีความเสี่ยงและยากต่อการเข้าถึงของมนุษย์

1.2.2. เพื่อสร้างหุ่นยนต์ที่มีประโยชน์และราคาไม่แพงเท่ากับที่ตามท้องตลาดขาย

1.2.3. เพื่อที่จะศึกษาการประดิษฐ์และการทำงานของหุ่นยนต์ต้นแบบ และมีความรู้ไปต่อยอดพัฒนาหุ่นยนต์ที่มีประสิทธิภาพและสมบูรณ์ในภายภาคหน้า

1.3ขอบเขตของโครงงาน

1.3.1. ส่วนของฮาร์ดแวร์

1. หุ่นยนต์สามารถเคลื่อนที่โดยการเดินหน้า, ถอยหลัง, เลี้ยวซ้าย, เลี้ยวขวา
2. หุ่นยนต์มีแขนที่สามารถหยิบจับสิ่งของและเปิดประตูได้
3. หุ่นยนต์สามารถเดินได้ทุกสภาพพื้นผิว
4. ใช้ Microcontroller ในควบคุมการทำงานของหุ่นยนต์
5 .สามารถควบคุมหุ่นยนต์ผ่านทางระบบเครือข่ายไร้สาย

1.3.2. ส่วนของซอฟแวร์

1.มีการแสดงภาพจากกล้องบนตัวหุ่นยนต์

.2 สามารถควบคุมการทำงานของหุ่นยนต์ผ่านทางวิทยุบังคับไร้สาย

1.4ผลที่คาดว่าจะได้รับ

1.4.1. หุ่นยนต์สามารถควบคุมแบบไร้สาย

1.4.2. หุ่นยนต์สามารถเคลื่อนที่ไปในทิศทางต่าง ๆได้อย่างอิสระ

1.4.3. หุ่นยนต์สามารถเคลื่อนที่ได้บนพื้นผิวเรียบและขรุขระ

1.4.4. หุ่นยนต์สามารถแสดงภาพจากกล้องบนตัวหุ่นยนต์ได้

บทที่ 2
หลักการและทฤษฎี

2.1 ความหมายเบื้องต้น

หุ่นยนต์ คือเครื่องยนต์ชนิดหนึ่งที่มีลักษณะโครงสร้างและรูปร่างแตกต่างกันไปตามวัตถุประสงค์ หุ่นยนต์ในแต่ละประเภทจะมีหน้าที่การทำงานในด้านต่าง ๆ ตามการควบคุมโดยตรงของมนุษย์ หรืออาจมีการตั้งค่าให้หุ่นยนต์สามารถตัดสินใจได้เองในระดับใดระดับหนึ่ง การควบคุมระบบต่าง ๆ ในการสั่งงานระหว่างหุ่นยนต์และมนุษย์ สามารถทำได้โดยทางอ้อมและอัตโนมัติ โดยทั่วไปหุ่นยนต์ถูกสร้างขึ้นเพื่อสำหรับงานที่มีความยากลำบากหรืออันตรายเช่น งานสำรวจในพื้นที่บริเวณแคบ งานสำรวจในบริเวณที่เกิดภัยพิบัติ หรืองานสำรวจบนผิวของดวงจันทร์หรือดาวเคราะห์ที่ต่างๆ ปัจจุบันเทคโนโลยีของหุ่นยนต์เจริญก้าวหน้าอย่างรวดเร็ว หุ่นยนต์เริ่มเข้ามามีบทบาทกับชีวิตของมนุษย์ในหลากหลายด้าน เช่น ด้านอุตสาหกรรมการผลิต ด้านการแพทย์ ด้านงานสำรวจทั้งในโลกเราและงานสำรวจในอวกาศ หรือด้านการบันเทิง เช่นหุ่นยนต์ที่ถูกสร้างขึ้นเพื่อเป็นเครื่องเล่นของมนุษย์ ปัจจุบันนี้ได้มีการพัฒนาให้หุ่นยนต์นั้นมีลักษณะที่คล้ายมนุษย์มากขึ้นเพื่อผลทางจิตวิทยาในการอาศัยอยู่ร่วมกันกับมนุษย์ในชีวิตประจำวัน

2.2 ทฤษฎีที่เกี่ยวข้อง

2.2.1. มอเตอร์เกียร์ (Motor Gear)

เป็นเครื่องกลที่ทำหน้าที่เปลี่ยนพลังงานไฟฟ้าให้เป็นพลังงานกลในรูปแบบของการหมุนเคลื่อนที่ โดยการเหนี่ยวนำแม่เหล็กไฟฟ้า ด้วยส่วนที่หมุนได้พันด้วยขดลวด มอเตอร์เกียร์มีโครงสร้างที่สำคัญ คือ ส่วนแม่เหล็กถาวร และส่วนของขดลวดตัวนำซึ่งมีโครงสร้างคล้ายกับเครื่องกำเนิดไฟฟ้า มอเตอร์เกียร์ทำงานโดย การหมุนเคลื่อนที่ของขดลวดตัวนำและทิศทางการเคลื่อนที่การทำงานของมอเตอร์ไฟฟ้าเรียกได้ว่าเป็นอุปกรณ์ที่นิยมใช้กันมากที่สุด และยังได้แพร่หลายไปยังโรงงานอุตสาหกรรมต่างๆ เป็นอุปกรณ์ที่ได้ใช้ในการควบคุมงานและเครื่องจักรต่างๆ ในโรงงานอุตสาหกรรม เครื่องจักรกลอุตสาหกรรม เครื่องลำเลียง เป็นต้น

รูปที่ 2.1 มอเตอร์เกียร์ (Motor Gear)

2.2.2. มอเตอร์แกนชัก (Linear Actuator)

เป็นมอเตอร์ที่เคลื่อนที่เชิงเส้น ขับเคลื่อนด้วยไฟฟ้า สามารถนำมาใช้งานทดแทนระบบไฮดรอลิค ระบบนิวเมติกได้

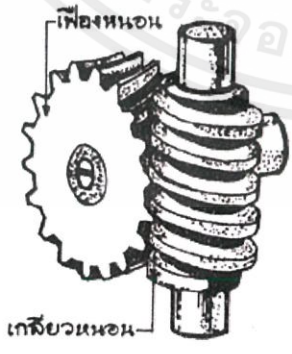
2.2.3. เฟือง

เฟือง คือ อุปกรณ์ที่ใช้ในการส่งถ่ายกำลังระหว่างเพลากับเพลา โดยอาศัยฟัน และเฟืองทั้งสองขบกัน นอกจากนี้เฟืองยังสามารถใช้ในการทดสอบเพื่อเพิ่มและลดความเร็วของเฟืองตัวที่ใช้ขับได้

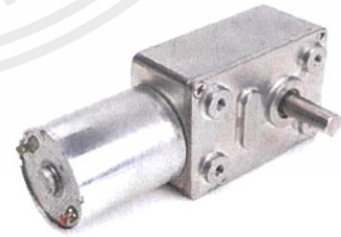
รูปที่ 2.2 มอเตอร์แกนชัก

2.2.3.1. เฟืองตัวหนอน (Worm Gear)

มีลักษณะเหมือนเกลียวและอาจมีเกลียวเดียวหรือหลายเกลียวได้ แต่ที่นิยมใช้กันมาก เป็นแบบหลายเกลียว เฟืองหนอนปกติต้องใช้เป็นคู่ คือ เกลียวหนอน (ตัวเล็ก) และ เฟืองหนอน (ตัวใหญ่) เฟืองชนิดนี้ให้อัตราทดสูงมาก แกนของเฟืองมักตั้งฉากกันและไม่ตัดกัน เฟืองชนิดนี้มักใช้ในการลดความเร็วรอบที่สูงมากๆ ให้เหลือความเร็วรอบต่ำๆ ซึ่งนำมาประยุกต์ใช้ในชุดหัวแบ่งที่ใช้กัดเฟือง หรือใช้ในแม่แรงยกของ เฟืองตัวหนอนประกอบด้วยสองชิ้นส่วนคือล้อตัวหนอน (Worm Wheel) และตัวเฟือง (Worm Gear) เฟืองตัวหนอนเป็นเฟืองที่มีการทำงานแบบ Self-locking และทำงานเงียบ แต่ก็ต้องยอมรับการสูญเสียพลังงานและแรงกระทำบนตัวหนอนที่สูงขึ้น



รูปที่ 2.3 ลักษณะของเฟืองตัวหนอน

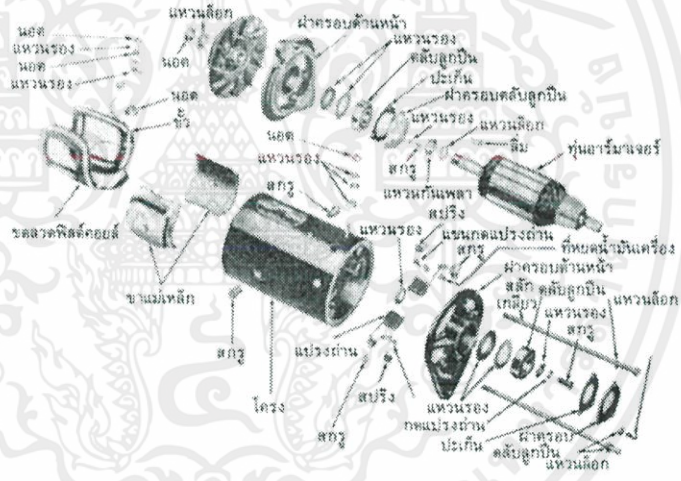


รูปที่ 2.4 เฟืองตัวหนอน 2 แกน

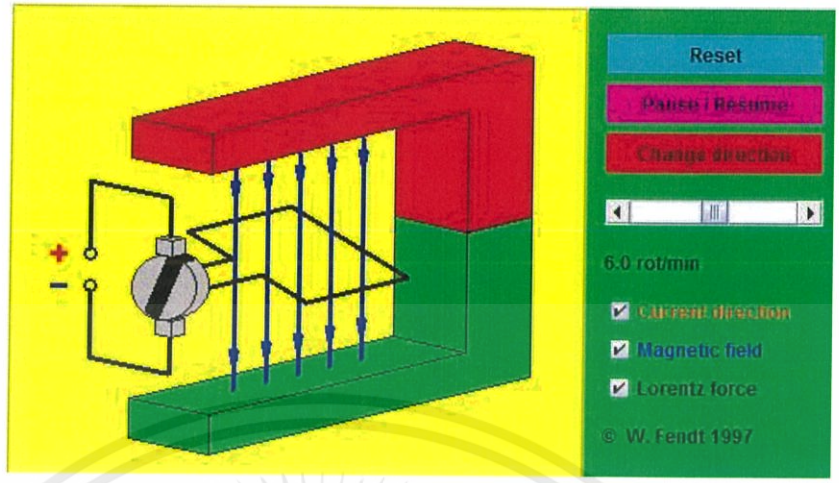
2.2.4. หลักการของมอเตอร์กระแสไฟฟ้าตรง

หลักการของมอเตอร์ไฟฟ้ากระแสตรง (Motor Action) เมื่อมีแรงดันกระแสไฟฟ้าตรงเข้าไปในมอเตอร์ ส่วนหนึ่งจะเข้าแปรงถ่านผ่านคอมมิวเตเตอร์เข้าไปในขดลวดอาร์มาเจอร์ สร้างสนามแม่เหล็กขึ้นและกระแสไฟฟ้า อีกส่วนหนึ่งจะไหลเข้าไปในขดลวดสนามแม่เหล็ก (Fieldcoil) สร้างขั้วเหนือ-ใต้ขึ้น เกิดสนามแม่เหล็ก 2 สนาม ในขณะเดียวกัน ตามคุณสมบัติของเส้นแรงแม่เหล็กจะไม่ตัดกัน ทิศทางตรงข้ามจะหักล้างกันและทิศทางเดียวจะเสริมแรงกัน ทาให้เกิดแรงบิดในตัวอาร์มาเจอร์

DC Motor ประกอบด้วย 2 ส่วนหลักๆ ได้แก่ ทุ่นอาร์มาเจอร์และสเตเตอร์ ทุ่นอาร์มาเจอร์ก็คือส่วนที่หมุน ส่วนสเตเตอร์คือส่วนที่เป็นขดลวดที่สร้างสนามแม่เหล็ก หลักการทางานของ DC Motor คือการนำกระแสตรงมาตัดผ่านสนามแม่เหล็ก ทาให้เกิดทอร์กขึ้นในทิศที่เหมาะสม และสร้างให้เกิดการหมุนของทุ่นอาร์มาเจอร์



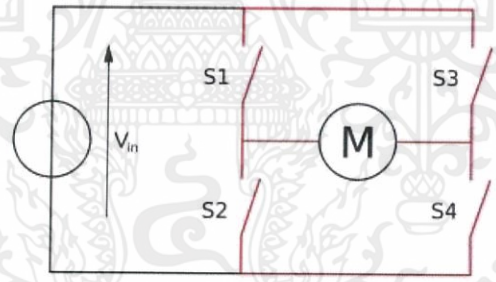
รูปที่ 2.5 ส่วนประกอบของมอเตอร์ไฟฟ้ากระแสตรง



รูปที่ 2.6 หลักการทำงานของมอเตอร์ไฟฟ้ากระแสตรง

2.2.4.1. การควบคุมทิศทาง และความเร็วรอบของมอเตอร์

การควบคุมทิศทาง และความเร็วรอบของมอเตอร์ ไม่ว่าจะเป็นการขับเคลื่อนสายพาน หรือล้อของหุ่นยนต์ไม่ให้หมุนแบบอิสระและสามารถควบคุมได้ ดังนั้นจึงมีวงจรที่ใช้ในการควบคุมมอเตอร์ขึ้นมาแบบที่นิยมใช้กันเรียกว่าวงจร “H-Bridge”



รูปที่ 2.7 การควบคุมมอเตอร์ H-Bridge

วงจรนี้ทำหน้าที่ได้ทั้งคุมทิศทางและความเร็วของมอเตอร์ โดยเริ่มจากการคุมทิศทางการหมุน โดยปกติหากต้องการกลับทิศการหมุนของมอเตอร์กระแสตรง วิธีหนึ่งที่ทำได้คือกลับทิศแหล่งจ่ายวงจร H-Bridge ด้านบน

หากต้องการให้หมุนตามเข็ม (Clockwise :CW) ให้ S1 และ S4 ปิดวงจร และให้ S2 และ S3 เปิดวงจรหากต้องการให้หมุนทวนเข็ม (Conter Clockwise :CCW) ให้ S2 และ S3 ปิดวงจร และให้ S1 และ S4 เปิดวงจร

สังเกตว่าสวิตช์จะทางานเป็นคู่ คู่แรกทางาน คู่สองต้องเปิดวงจร และในทางตรงข้ามก็คือคู่สองทางาน คู่แรกต้องเปิดวงจร ต่อมามีการทาให้การเปิดปิดเป็นแบบที่ง่ายกว่าเดิม โดยใช้อุปกรณ์สารกึ่งตัวนำเช่น MOSFET หรือ IGBT หรืออื่นๆ แล้วแต่ความเหมาะสม เช่นขนาดกระแสแรงดันที่ต้องการควบคุม

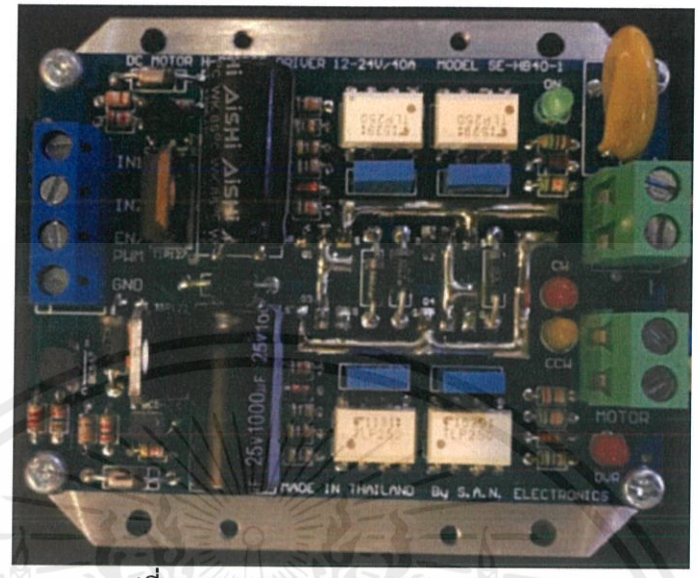
2.2.4.1.1 H-BRIDGE DRIVER DC MOTOR 80A

รูปที่ 2.8 H-BRIDGE DRIVER DC MOTOR 80A

รายละเอียดทางเทคนิค

1. Output : Single motor driver
- Motor DC Supply 12-48 V 80A (Max. กระแสพัลส(
- Full-Complementary Power MOSFET Driver
- With ultra-fast reverse recovery protection diodes
2. Input :
- Full Opto-isolated input interface signals
- 5V 8 mA TTL – Level
3. Drive Mode : independently with :
- ON – OFF Control
- Direction Control
- Speed Control (PWM Drives)
4. PWM Frequency : 400 Hz - 1000 Hz

2.2.4.1.2 H-BRIDGE DRIVER DC MOTOR 40A



รูปที่ 2.9 H-BRIDGE DRIVER DC MOTOR 40A

รายละเอียดทางเทคนิค

1. Output : Single motor driver

- Motor DC Supply 12-24 V 40A (Max. กระแสพัลส์(

- Complementary Power MOSFET H-Bridge Driver With ultra-fast reverse recovery inverse parallel protection diodes

2. Input :

- Full Opto-isolated ground

-loop input interface signals

- Control Voltage input : 3-5V / 8 mA(min.)

3. Drive Mode : independently with :

- Start-stop Control

- Direction Control

- Speed Control (PWM Drives)

4. PWM Duty cycle Range: 0-100%

5. PWM Frequency: 400 Hz

- 25000 Hz

6. Board built

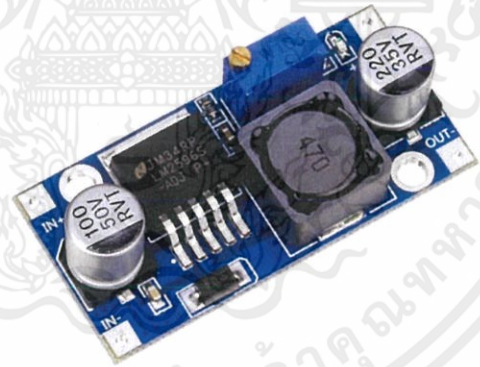
-in Transient voltage protection up to 100 V circuit

EN/PWM	IN2	IN1	การทำงานของมอเตอร์
5V/PWM	GND	5V	หมุนเดินหน้า
5V/PWM	5V	GND	หมุนกลับทาง
5V/PWM	5V	5V	Fast Stop หรือ Brake (ไม่แนะนำให้ใช้)
5V/PWM	GND	GND	Fast Stop หรือ Brake (แนะนำให้ใช้)

ตาราง2.1 ควบคุมการทำงาน

2.2.5 DC to DC Converter

วงจรลดแรงดันแบบ Step-Down หรือเรียกอีกแบบว่า Buck Converter (บัคคอนเวอร์เตอร์) ใช้ลดแรงดันจากแรงดันสูงให้ต่ำลง ใช้หลักการสวิตชิ่ง-ตัวเหนี่ยวนำ (L) จึงทำให้มีความร้อนและความสูญเสียกำลังไฟน้อย ไม่เหมือนกับการลดแรงดันโดยใช้ IC ตระกูล 78xx / 317 ทั่วไปที่ใช้หลักการลดทอนทำให้เกิดความร้อนสูง วงจรบัคคอนเวอร์เตอร์เมื่อลดแรงดันลงแล้วจะได้กระแส Output เพิ่มขึ้น



รูปที่ 2.10 Step-Down Adjustable DC-DC Switching Buck Converter

2.2.5.1 หลักการ DC-DC Converter

วงจร DC-DC เป็นแหล่งจ่ายไฟ DC เป็นค่าแรงดันไฟฟ้าที่แตกต่างกันของวงจร DC-DC เป็นสาขาของการเปลี่ยนเทคโนโลยีจ่ายไฟสลับเทคโนโลยีจ่ายไฟรวมถึง AC-DC, DC-DC สาขาสองสาขา วงจร DC-DC โดยฟังก์ชั่นแบ่งออกเป็น:

Boost converter: วงจรที่แปลงแรงดันไฟฟ้าต่ำเป็นแรงดันไฟฟ้าสูง

บั๊กแปลง: แปลงแรงดันไฟฟ้าสูงเป็นวงจรแรงดันต่ำ

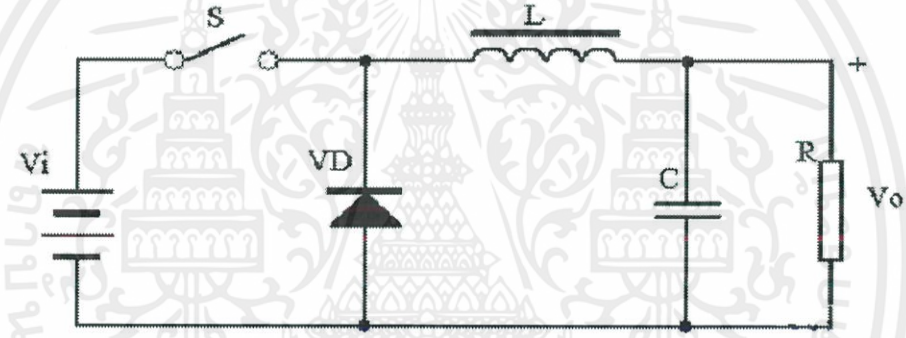
อุปกรณ์ย้อนกลับ: การเปลี่ยนแปลงขั้วแรงดันไฟฟ้าในวงจร, แหล่งจ่ายไฟเชิงบวกอุปทานเชิงลบ, แหล่งจ่ายไฟเชิงลบอุปทานบวกสองประเภท

DC-DC converter วงจรพื้นฐานที่มีตัวแปลงเพิ่ม, ตัวแปลงบั๊ก, ตัวแปลง step-up และ down converter สาม

ดังแสดงในรูปที่ 2.11 เมื่อปิดสวิตช์แรงดันไฟฟ้าที่ใช้กับปลายทั้งสองของตัวเหนี่ยวนำคือ (Vi-Vo) เมื่อแรงดึงดูดจากแรงดันไฟฟ้า (Vi-Vo) แรงแม่เหล็กเพิ่มขึ้น: Vi-Vo) * ตัน

เมื่อสวิทช์ปิดอยู่เนื่องจากกระแสไฟออกอย่างต่อเนื่องไดโอด VD จะกลายเป็นตัวนำความเหนี่ยวนำของแม่เหล็กลดการเหนี่ยวนำของฟลักซ์แม่เหล็ก: (โวฟ) * โทฟฟ์

(Vi-Vo) * Ton = (Vo) * Toff เพราะวัฏจักรหน้าที่ D <1, ดังนั้น="" vi=""> Vo, เมื่อสวิทช์ปิดและปิดสถานะให้สมดุล,

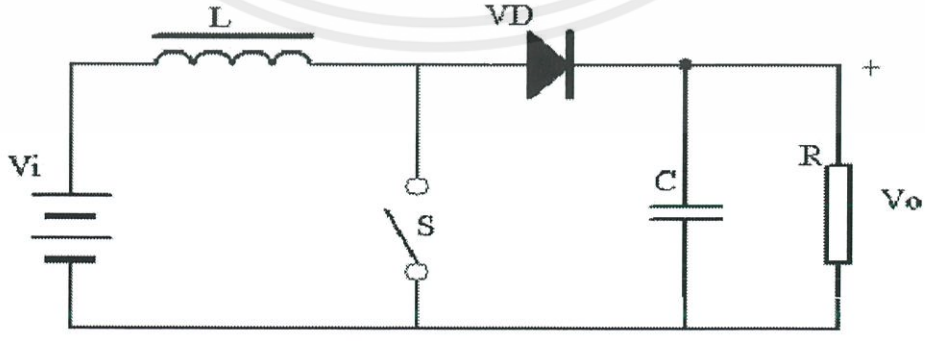


รูปที่ 2.11 บัฟเฟอร์แปลงวงจร

แผนภาพ Boost แปลงแผนภาพแสดงในรูปที่ 2.12 เมื่อปิดสวิตช์แรงดันไฟฟ้าเข้ากับเหนี่ยวนำเมื่อเหนี่ยวนำโดยแรงดันไฟฟ้า (Vi) กระตุ้นตัวเหนี่ยวนำเพิ่ม magnet fluux: (Vi) * Ton

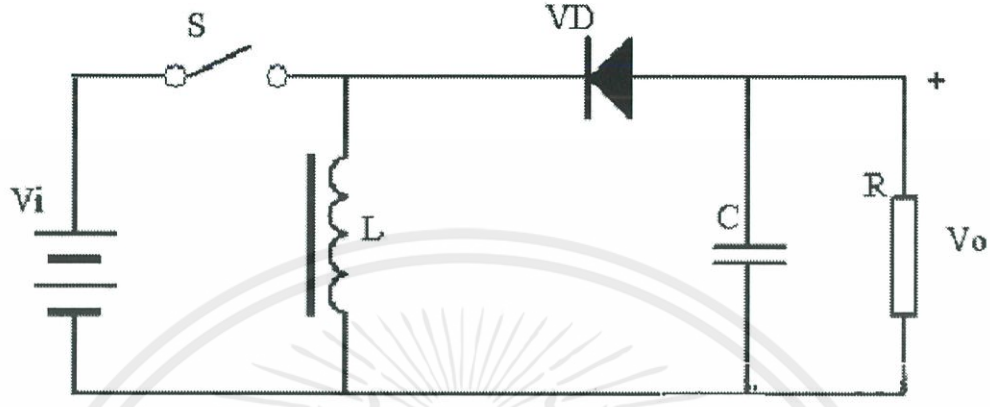
เมื่อสวิทช์ดับลงเนื่องจากกระแสไฟออกอย่างต่อเนื่องไดโอด VD จะกลายเป็นตัวนำตัวเหนี่ยวนำแม่เหล็กลดการเหนี่ยวนำของฟลักซ์แม่เหล็ก: (โวฟ) * โทฟฟ์

(Vi) * Ton = (Vo-Vi) * Toff เมื่อสวิตช์ปิดและสวิตช์ถูกตัดการเชื่อมต่อ Vi คือ D <1 ดังนั้น="">



รูปที่ 2.12 วงจรแปลง Boost

ตัวแปลงขึ้นและลง, ขั้วของขั้วกลับของหลักการที่แสดงในรูปที่ 3 เมื่อสวิทช์ปิดตัวเหนี่ยวนำของเหนี่ยวนำฟลักซ์แม่เหล็กลดลงโดย: (Vo) * Toff (Vi) * Ton = (Vo) * Toff ขึ้นอยู่กับ Ton จะแตกต่างจากค่า Toff อาจเป็น Vi Vo



รูปที่ 2.13 แผนผังการแปลง Step-up และ Down

2.2.5.2 รายละเอียดทางเทคนิค

Input Voltage - 3 to 40 V

Output Voltage Range - 1.23 to 37 V

Max Output Current - 3 A

Adjustment - 25-Turn Trimpot

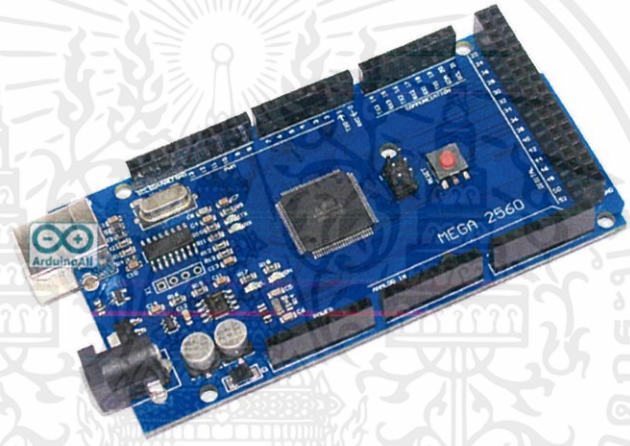
Efficiency - up to ~93%

Switching Frequency - 150 kHz

Built-In Protection - thermal shutdown and current limit

2.2.6. Arduino

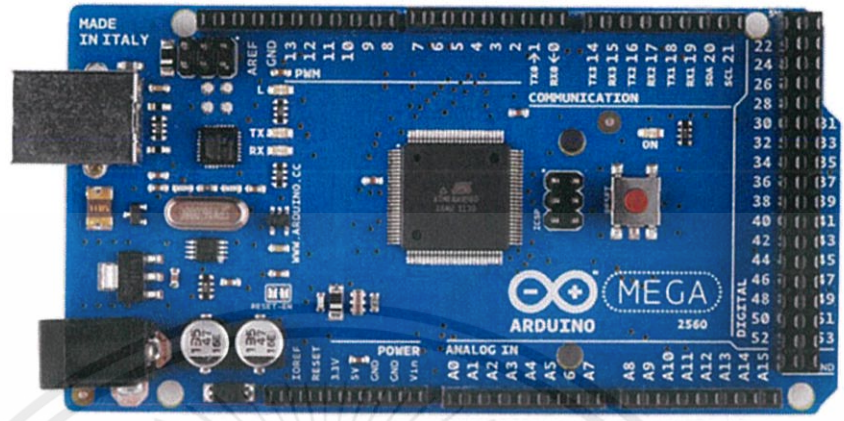
Arduino คือ บอร์ดไมโครคอนโทรเลอร์ตระกูล AVR มีการพัฒนาแบบ Open Source คือมีการเปิดเผยข้อมูลทั้งด้าน Hardware และ Software โดยด้าน Hardware คือ บอร์ดอิเล็กทรอนิกส์ขนาดเล็กสามารถเคลื่อนย้ายพกพาได้สะสวดโดยมีไมโครคอนโทรลเลอร์เป็นอุปกรณ์หลักที่สำคัญและมีอุปกรณ์อิเล็กทรอนิกส์ต่างๆมาประกอบรวมกัน ต่อมาในด้าน Software ตัวบอร์ด Arduino เป็นบอร์ดที่ใช้ไมโครคอนโทรลเลอร์ดังนั้นจึงมีลักษณะภาษาแบบเดียวกับ ภาษา C/C++ซึ่งเป็นภาษาสำหรับเขียนโปรแกรมที่แพร่หลาย ดังนั้นตัวบอร์ด Arduino จึงสามารถใช้งานได้ง่าย จึงเหมาะสำหรับผู้เริ่มต้นศึกษา ทั้งนี้ผู้ใช้งานยังสามารถดัดแปลง เพิ่มเติม พัฒนาต่อยอดทั้งตัวบอร์ดหรือโปรแกรมต่อได้อีกด้วย



รูปที่ 2.14 Arduino MEGA 2560 R3

ความสะดวกและไม่ซับซ้อนของบอร์ด Arduino ทำให้สามารถต่ออุปกรณ์เสริมต่างๆได้อย่างสะดวกนั่นคือผู้ใช้งานสามารถต่อวงจรอิเล็กทรอนิคส์ต่างๆจากภายนอกแล้วเชื่อมต่อเข้ามาที่ขา I/O ของบอร์ดหรือสามารถเลือกต่อกับบอร์ดเสริม (Arduino Shield) ประเภทต่างๆ เช่น Arduino XBee Shield, Arduino Music Shield, Arduino Relay Shield, Arduino Wireless Shield, Arduino GPRS Shield เป็นต้นมาเสียบกับบอร์ดบนบอร์ด Arduino แล้วเขียนโปรแกรมพัฒนาต่อได้

2.2.6.1 โครงสร้างของบอร์ด Arduino MEGA 2560 R3



รูปที่ 2.15 โครงสร้างของบอร์ด Arduino MEGA 2560 R3

Arduino Mega 2560 คือบอร์ดไมโครคอนโทรลเลอร์ที่พัฒนาจาก ATmega2560 มี 54 digital input/output โดยมี 14 ขา สามารถใช้เป็น output แบบ PWM ได้ มี analog inputs 16 ขา มี UARTs(hardware serial ports) 4 ขา ทำงานที่ความถี่ 16 MHz สามารถเชื่อมต่อกับคอมพิวเตอร์ด้วยสายเคเบิล USB หรือใช้ adaptor AC-to-DC เพื่อเริ่มต้นใช้งาน และมีปุ่ม reset สามารถต่อเข้ากับ shields ที่ออกแบบเพื่อใช้งานกับ Arduino Duemilanove หรือ Diecimila.

Arduino Mega 2560 R3 เป็นบอร์ด 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 ทำให้สามารถเขียนโค้ดโปรแกรมเข้าไปได้มากกว่าในความเร็วของ MCU ที่เท่ากัน

Pin ทั่วไป

- VIN เป็น input voltage ของบอร์ด Arduino โดยใช้แหล่งจ่ายจากภายนอก
- 5V เป็น output pin ที่ควบคุม 5 V จากบอร์ด
- 3V3 เป็น 3.3 volt supply ที่สร้างขึ้นจาก regulator บนบอร์ด และให้กระแสได้สูงสุด 50 mA
- GND เป็น ground pin
- IOREF เป็น pin ที่ให้ voltage reference กับไมโครคอนโทรลเลอร์ เพื่อเลือกค่าแรงดันให้กับ shield ที่มาเชื่อมต่อกับบอร์ด

หน่วยความจำ

ATmega2560 มีหน่วยความจำ 256 KB (8 KB ใช้สำหรับ bootloader) นอกจากนี้ยังมีอีก 8 KB สำหรับ SRAM และ 4 KB สำหรับ EEPROM

Input and Output

ในแต่ละ digital pins ทั้ง 54 pins บนบอร์ด Arduino Uno สามารถเป็นได้ทั้ง input และ output โดยจะทำงานที่แรงดัน 5 V และให้กระแสสูงสุด 40 mA

ฟังก์ชันอื่นๆ

• External Interrupts: 2 (interrupt 0) , 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), 21 (interrupt 2). pins เหล่านี้สามารถที่จะกำหนดค่าที่เรียกinterrupt ในค่าต่ำๆ , ขอบขาขึ้นและลง หรือเปลี่ยนแปลงค่า

• PWM: 2 ถึง 13 และ 44 ถึง 46 ให้ output PWM output 8-bits

• SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS) ใช้สำหรับรองรับการสื่อสารแบบ SPIโดยที่ไม่เกี่ยวข้องกันกับ ICSP header ซึ่งจะมีลักษณะคล้ายกับ Uno, Duemilanove และDiecimila

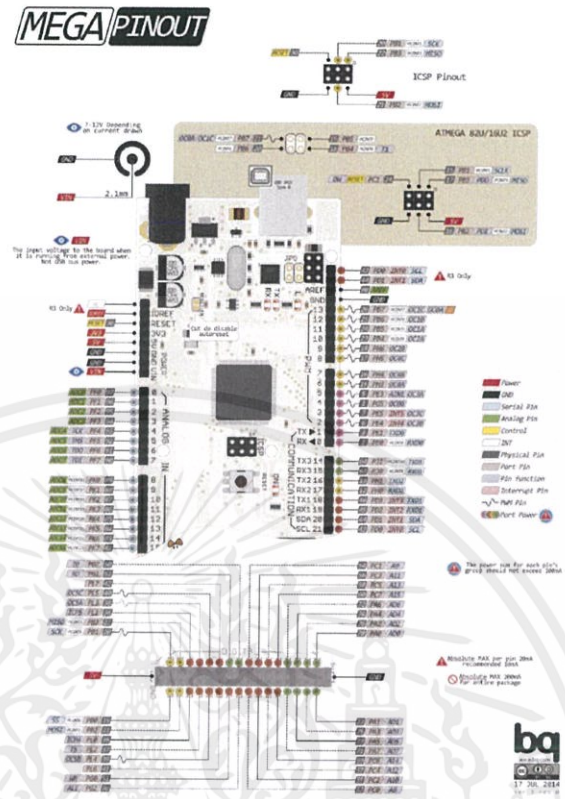
• LED 13 : เป็น build-in LED ที่เชื่อมต่อกับ digital pin 13 เมื่อ pin มีค่าเป็น HIGH LEDจะติด , แต่เมื่อ pin เป็น LOW LED จะดับ

• TWI : 20 (SDA) and 21 (SCL). รองรับการเชื่อมต่อแบบ TWI(I2C)

• บอร์ด Mega2560 มี 16 analog inputs แต่ละ pins ให้ความละเอียด 10 bits

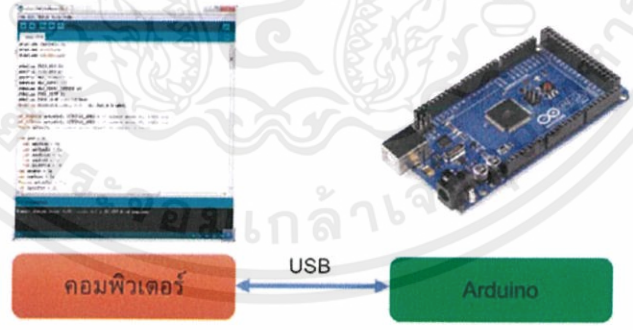
• AREF. แรงดันอ้างอิง สำหรับ analog input

• Reset ใช้ในการ reset ไมโครคอนโทรลเลอร์ โดยทั่วไปจะใช้โดยการเพิ่มปุ่ม reset ไว้บน shield เพื่อป้องกันปุ่มที่อยู่บนบอร์ด



รูปที่ 2.16 Pinout ของ Arduino MEGA 2560 R3

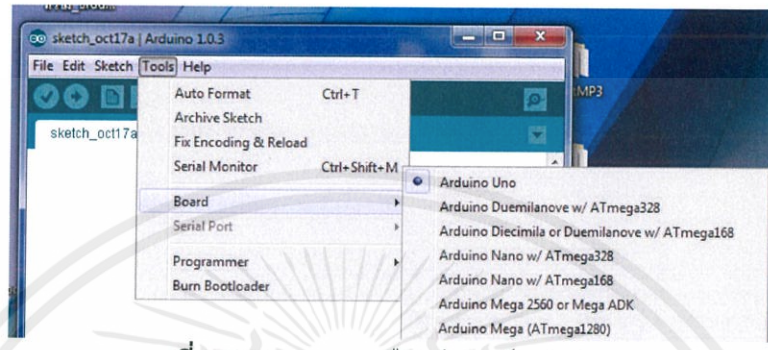
2.2.6.2 รูปแบบการเขียนโปรแกรมบน arduino



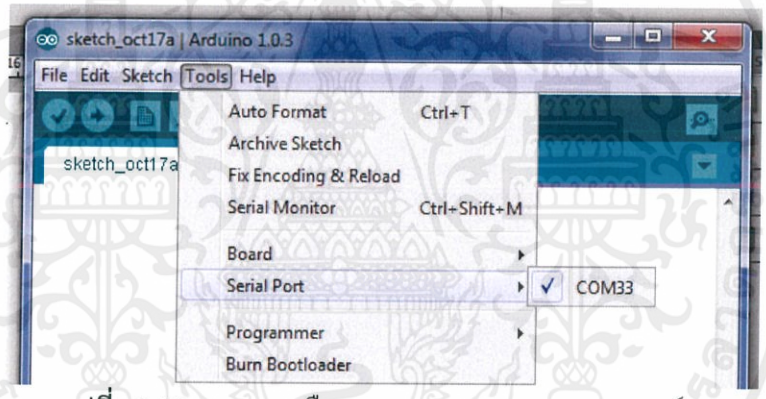
รูปที่ 2.17 แสดงการเชื่อมต่อบอร์ด arduino กับคอมพิวเตอร์

1. เขียนโปรแกรมบนคอมพิวเตอร์ผ่านทางโปรแกรม ArduinoIDE
2. หลังจากที่เขียนโค้ดโปรแกรมเรียบร้อยแล้วให้ผู้ใช้งานเลือกรุ่นบอร์ดArduinoที่ใช้และหมายเลข Com port

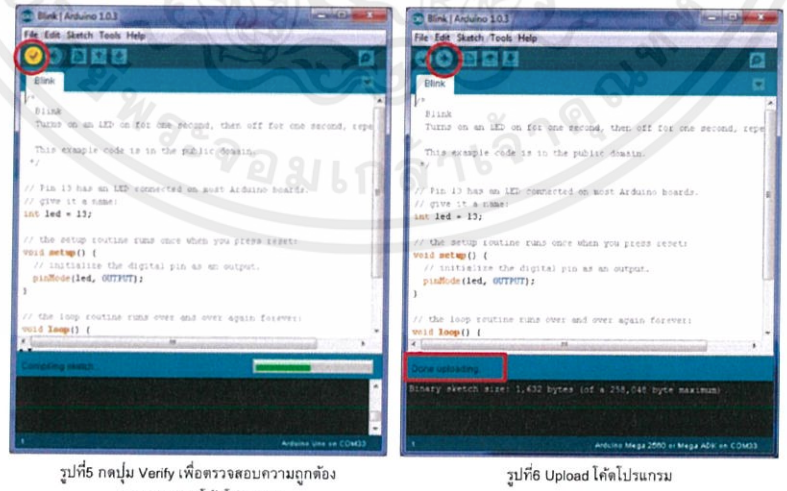
3. กดปุ่ม Verify เพื่อตรวจสอบความถูกต้องและ Compile โค้ดโปรแกรม จากนั้นกดปุ่ม Upload โค้ด โปรแกรมไปยังบอร์ด Arduinoผ่านทางสาย USB เมื่ออับโหลดเรียบร้อยแล้ว จะแสดงข้อความแถบข้างล่าง “Done uploading” และบอร์ดจะเริ่มทำงานตามที่เขียนโปรแกรมไว้ได้ทันที



รูปที่ 2.18 แสดงการเลือกรุ่นบอร์ด Arduino



รูปที่ 2.19แสดงการเลือกหมายเลข comport ของบอร์ด



รูปที่5 กดปุ่ม Verify เพื่อตรวจสอบความถูกต้องและ Compile โค้ดโปรแกรม

รูปที่6 Upload โค้ดโปรแกรม

รูปที่ 2.20โค๊ดโปรแกรม

2.2.6.3 โครงสร้างโปรแกรมสำหรับ Arduino

ในการเขียนโค้ด (sketch) สำหรับ Arduino สามารถแบ่งได้เป็น ส่วนสำคัญคือ 4

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

ชนิดข้อมูลพื้นฐานสำหรับ Arduino C/C++

Byte ใช้สำหรับข้อมูลที่เป็นเลขจำนวนเต็มตั้ง 255 ถึง 0

Int ใช้สำหรับข้อมูลที่เป็นเลขจำนวนเต็มได้ตั้งแต่ 32768-ถึง 32767+

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

Float ใช้สำหรับข้อมูลที่เป็นเลขทศนิยม เป็นค่าที่เป็นบวกหรือลบในช่วงที่กว้างกว่าชนิดข้อมูลแบบ byte และ int และเป็นตัวเลขทศนิยมได้ด้วยแต่มีความละเอียดเพียง ตำแหน่งหลังจุด 7-6 ทศนิยมในเลขฐานสิบ

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 ก่อนที่จะ (มิลลิวินาที) ทำ ขั้นตอนต่อไป

delayMicroseconds ( ) รอให้ระยะเวลาผ่านไปตามระยะเวลาที่กำหนด (ไมโครวินาที)

randomSeed ( ) กำหนดค่าเริ่มต้นสำหรับการสร้างเลขแบบสุ่มเทียม

random ( ) ให้ค่าเป็นเลขสุ่มเทียม)Pseudo-Random Number(

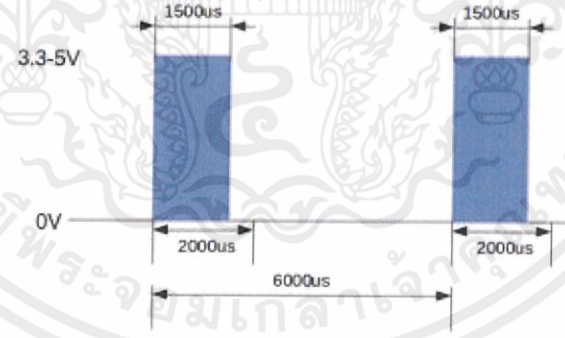
millis ( ) บอกเวลาที่ผ่านไปในหน่วยเป็นมิลลิวินาทีนับตั้งแต่โปรแกรมเริ่มต้นทำงาน

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(

2.2.7. การอ่านสัญญาณ PWM จากเครื่องรับสัญญาณ RC พร้อมด้วย Arduino

RC receivers รับสัญญาณเอาต์พุตสัญญาณพัลส์ความกว้าง (PWM) ในแต่ละช่องสัญญาณพัลส์เหล่านี้โดยทั่วไปอยู่ระหว่างหนึ่งถึงสองมิลลิวินาที ความยาวของชีพจร 1500 microseconds จะขับเคลื่อนเซอร์โวมาตรฐานไปครึ่งทาง 1000 วินาทีมีการเดินทางเต็มรูปแบบในทิศทางเดียวและ 2000 วินาทีก็เดินทางไปในทิศทางอื่นได้เต็มที่ มีระยะเวลา 20 มิลลิวินาทีระหว่างแต่ละชีพจร

สัญญาณ PWM มิลลิวินาทีเช่นนี้จะขับเซอร์โวแบบมาตรฐานไปยังจุดศูนย์กลาง 1.5



รูปที่2.21 สัญญาณ PWM จากเครื่องรับสัญญาณ RC

การอ่านสัญญาณ PWM Arduino มาพร้อมกับฟังก์ชันง่ายๆที่เรียกว่า In ()

การอ่านจากแหล่ง PWM สามารถทำได้ดังนี้

```
1 #define PWM_SOURCE 34
2 ...
3
4 pwmin = pulseIn(PWM_SOURCE, HIGH, 20000);
```

รูปที่2.22 Code การอ่านสัญญาณ PWM

จะอ่าน PWM จากช่องทางเดียวที่เชื่อมต่อกับดิจิตอลขา 34 ช่องเพิ่มเติมสามารถเพิ่มได้อย่างง่ายดายในลักษณะเดียวกัน

ข้อเสียคือการส่งไปยัง pulseIn () แต่ละครั้งอาจใช้เวลาประมาณ 20 มิลลิวินาที เนื่องจาก PulseIn ()รอ PIN จากดิจิตอล LOW ถึง HIGH และกลับไปที่ LOW อีกครั้ง

2.2.7.2 การอ่านสัญญาณ PWM โดยใช้ *hardware interrupts*

สามารถอ่านสัญญาณ PWM โดยใช้ hardware interrupts ซึ่งเป็นสัญญาณที่สร้างโดยฮาร์ดแวร์ที่ขัดจังหวะตัวประมวลผลอย่างแท้จริง ด้วย Arduino การขัดจังหวะฮาร์ดแวร์สามารถเกิดขึ้นได้โดยค่าการเปลี่ยนPin ซึ่งจะต่ำลงไปสูงขึ้นหรือลดลง โปรเซสเซอร์ตอบสนองต่อการขัดจังหวะโดยระงับการทำงานปัจจุบันและจัดการขัดจังหวะด้วยฟังก์ชันตัวจัดการการขัดจังหวะ (หรือที่เรียกว่า ISR-interrupt service routine) หลังจากที่ตัวจัดการการขัดจังหวะกลับมาแล้วตัวประมวลผลจะกลับมาทำงานก่อนหน้านี้

ในการอ่านค่า PWM input เราต้องทราบว่าเมื่อพินมีค่าสูงหรือต่ำมากดังนั้นจึงสนใจใน CHANGE interrupts เท่านั้น เมื่อขา PWM ไปที่ HIGH เครื่องจับเวลาจะเริ่มทำงาน เมื่อขาไปต่ำเราสามารถวัดเวลาชีพจรโดยการตรวจสอบระยะเวลาที่ผ่านไป

Arduino มีฟังก์ชัน attachInterrupt () ซึ่งช่วยให้สามารถจัดหาตัวจัดการการขัดจังหวะสำหรับเหตุการณ์และหมายเลขพินได้ ฟังก์ชัน micros () ช่วยให้สามารถวัดเวลาเป็นมิลลิวินาทีระหว่างขาไปสูงและกลับสู่ระดับต่ำ นี่คือตัวอย่างร่าง เพียงแค่พิมพ์พัลส์ PWM pulse เป็นหน่วยเป็นมิลลิวินาทีในพอร์ตอนุกรม

2.2.8.เครื่องส่งสัญญาณและเครื่องรับวิทยุ

2.2.8.1 เครื่องส่งสัญญาณวิทยุ (RADIO TRANSMITTERS)

เครื่องส่งสัญญาณวิทยุประกอบด้วยองค์ประกอบหลายอย่างที่ทำงานร่วมกันเพื่อสร้างคลื่นวิทยุที่มีข้อมูลที่เป็นประโยชน์เช่นเสียงวิดีโอหรือข้อมูลดิจิทัล

Power supply : ให้พลังงานไฟฟ้าที่จำเป็นสำหรับการใช้งานเครื่องส่งสัญญาณ

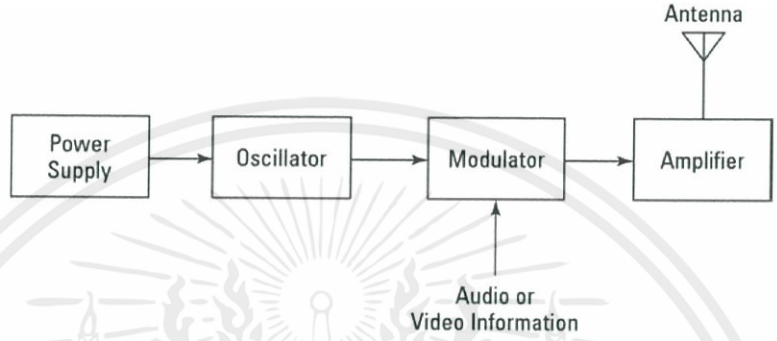
Oscillator : สร้างกระแสสลับที่ความถี่ที่เครื่องส่งสัญญาณจะส่ง oscillator มักจะสร้างคลื่นไซน์ซึ่งเรียกว่า wave carrier

Modulator : เพิ่มข้อมูลที่เป็นประโยชน์ไปยัง wave carrier มีสองวิธีหลักในการเพิ่มข้อมูลนี้ ครั้งแรกที่เรียกว่าการปรับความกว้างหรือ AM จะทำให้การเพิ่มขึ้นหรือลดลงเล็กน้อยต่อความเข้มของ

คลื่นผู้ให้บริการ ประการที่สองเรียกว่าการปรับความถี่หรือ FM ทำให้เพิ่มหรือลดลงเล็กน้อยในความถี่ของคลื่นผู้ให้บริการ

Amplifier : ขยายคลื่นผู้ให้บริการแบบปรับได้เพื่อเพิ่มพลังของมัน เครื่องขยายเสียงที่ทรงพลังมากขึ้นการกระจายเสียงมีประสิทธิภาพยิ่งขึ้น

Antenna : แปลงสัญญาณที่ขยายสัญญาณไปยังคลื่นวิทยุ



รูปที่2.23 ขั้นตอนการส่งสัญญาณวิทยุ

2.2.8.2.เครื่องรับวิทยุ (Receivers)

เครื่องรับวิทยุเป็นสิ่งที่ตรงกันข้ามกับเครื่องส่งสัญญาณวิทยุ ใช้เสาอากาศเพื่อจับคลื่นวิทยุประมวลผลคลื่นเหล่านั้นเพื่อแยกเฉพาะคลื่นที่สั่นสะเทือนที่ความถี่ที่ต้องการแยกสัญญาณเสียงที่เพิ่มเข้าไปในคลื่นเหล่านั้นขยายสัญญาณเสียงและเล่นกับลำโพง

Antenna: จับคลื่นวิทยุ โดยปกติเสาอากาศเป็นเพียงความยาวของสายไฟ เมื่อสายไฟสัมผัสกับคลื่นวิทยุคลื่นจะทำให้กระแสไฟสลับในเสาอากาศมีขนาดเล็กมาก

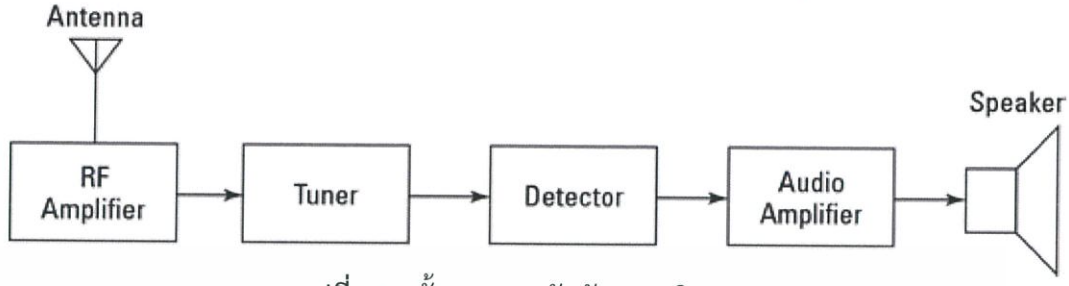
RF amplifier: เครื่องขยายเสียงที่มีความละเอียดอ่อนที่ขยายสัญญาณคลื่นวิทยุที่อ่อนแอ)RF) จากเสาอากาศเพื่อให้สัญญาณสามารถประมวลผลได้โดยTuner

Tuner: วงจรที่สามารถดึงสัญญาณของความถี่เฉพาะจากการผสมสัญญาณความถี่ต่างๆ เสาอากาศของตัวเองจับคลื่นวิทยุของคลื่นความถี่ทั้งหมดและส่งไปยังเครื่องขยายสัญญาณ RF ซึ่งจะช่วยขยายทุกคลื่น

Detector: รับผิดชอบในการแยกข้อมูลเสียงออกจากคลื่นผู้ให้บริการ สำหรับสัญญาณ AM นี้สามารถทำได้ด้วยไดโอดที่ปรับแก้สัญญาณกระแสสลับเท่านั้น สิ่งที่เหลืออยู่หลังจากไดโอดมีทางของมันด้วยสัญญาณกระแสสลับเป็นสัญญาณกระแสตรงที่สามารถป้อนเข้าวงจรเครื่องขยายเสียงได้ สำหรับสัญญาณ FM วงจรเครื่องตรวจจับจะซับซ้อนกว่าเล็กน้อย

Audio amplifier: งานของชิ้นส่วนนี้คือการขยายสัญญาณที่อ่อนแอซึ่งมาจากเครื่องตรวจจับ

เพื่อให้สามารถได้ยินได้ ซึ่งสามารถทำได้โดยใช้วงจรขยายทรานซิสเตอร์แบบง่ายๆ



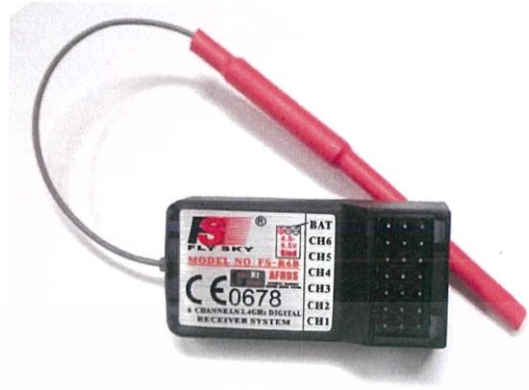
รูปที่2.24 ขั้นตอนการรับสัญญาณวิทยุ

2.2.9. Remote control และ Receiver

รูปที่2.25 Remote control

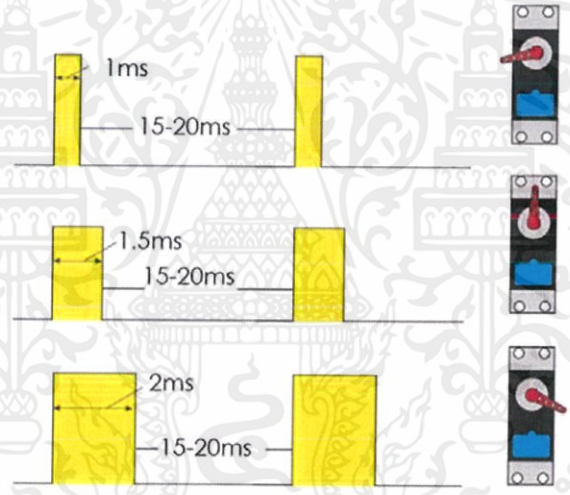
ปัจจุบันรีโมตคอนโทรลเปลี่ยนจากคลื่นวิทยุ FM มาเป็นคลื่นความถี่สูง 2.4 GHz ทำให้สามารถนำรีโมตคอนโทรล และรีซีฟเวอร์หลายๆ ตัวมาใช้ในที่เดียวกันได้ โดยคลื่นไม่ชนกัน ที่เห็นในรูปเป็นรีโมตคอนโทรลแบบ 4 ช่องสัญญาณซึ่งมีราคาขายไม่สูงมากนัก โดยตัวมันจะมาพร้อมรีซีฟเวอร์ที่มีรหัสหมายเลขตรงกัน

รูปที่2.26 Remote control และ Receiver



รูปที่2.27 Receiver

ภาครีซีฟเวอร์ทำหน้าที่รับสัญญาณจากรีโมตคอนโทรล เอาต์พุตจะเป็นขั้วต่อเซอร์โวมอเตอร์เพื่อไปควบคุมเซอร์โวมอเตอร์หรือสปีดคอนโทรล ดังนั้นถ้าเราจับสัญญาณควบคุมเซอร์โวที่มาจากรีซีฟเวอร์ได้ก็นำรีโมตไปใช้งานได้



รูปที่2.28 รูปแบบพัลส์ที่รีซีฟเวอร์ได้รับ

พัลส์สำหรับขับเซอร์โวมอเตอร์มีการเปลี่ยนแปลงความกว้างของพัลส์ช่วงบวกอยู่ในช่วง 1-2 มิลลิวินาที เพื่อให้เซอร์โวมอเตอร์เคลื่อนที่ไปในระยะต่างๆ ดังรูป เราต้องจับค่าความกว้างพัลส์ช่วงบวกแล้วนำไปใช้งาน

สำหรับ Arduino มีคำสั่งชื่อ PulseIn สำหรับอ่านค่าความกว้างพัลส์ โดยให้ผลลัพธ์หน่วยเป็นไมโครวินาที

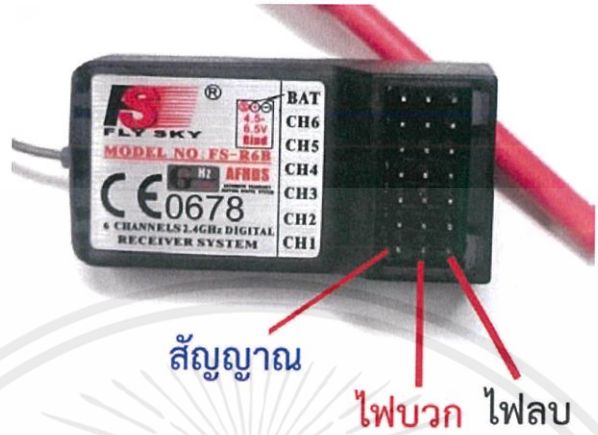
2.2.9.1 รูปแบบการใช้งานคำสั่ง

pulseIn(pin, value ,timeout)

โดย pin คือขาดิจิตอลอินพุตที่ต้องการอ่านค่า

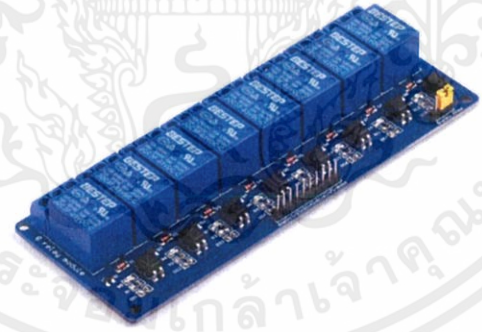
value คือลอจิกที่จะอ่านค่าความกว้างพัลส์ออกมา ในที่นี้จะอ่านที่ลอจิก “1”

timeout คือช่วงเวลาที่เรากำหนดว่าถ้าไม่เกิดพัลส์ขึ้นมาในช่วงเวลากี่ไมโครวินาทีให้ไปทำงานคำสั่งถัดไป



รูปที่2.29 แสดงตำแหน่งสัญญาณไฟบวกและลบ
สายสีแดง เป็น ไฟบวก (+)
สายสีดำ เป็น ไฟลบ (-)
สายสีขาว เป็น สัญญาณ)S)

2.2.10. Relay Module



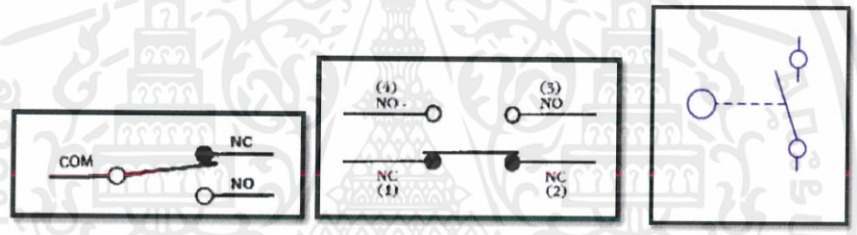
รูปที่2.30. 8 channel relay module

ชุดรีเลย์ที่ใช้ต่อกับเอาต์พุตของ PLC หรืออุปกรณ์อื่นๆ ที่เป็นทั้ง PNP หรือ NPN เพื่อนำ Contact Relay ไปใช้งาน ใช้พื้นที่ในการติดตั้งน้อย ซ่อมแซมได้ง่าย เคลื่อนย้ายได้สะดวก รูปทรงสวยงาม สามารถติดตั้ง บนราง DIN RIAL ในตู้ไฟฟ้าได้ มี LED โชว์สภาวะการทำงานของ Relay ใช้กับสายขนาด 2.5mm×2.5mm สามารถทนอุณหภูมิการติดตั้งได้ 0 ถึง 50 องศา

2.2.11. สวิตช์จำกัดระยะ) Limit Switch)

รูปที่2.32 สวิตช์จำกัดระยะ)Limit switch)

เป็นสวิตช์ที่จำกัดระยะทาง การทำงานอาศัยแรงกดภายนอกมากระทำ เช่น วางของทับที่ปุ่มกด หรือลูกเบี้ยวมาชนที่ปุ่มกด และเป็นผลทำให้หน้าสัมผัสที่ต่ออยู่กับก้านชน เปิดปิด ตามจังหวะของการชน-



รูปที่2.33 สัญลักษณ์สวิตช์จำกัดระยะ)Limit switch)

ดังนั้น จึงมีการนำไปใช้ประโยชน์ได้อย่างมากมาย เช่น ลิฟท์โดยสาร, ลิฟท์ขนของ, ประตูที่ทำงานด้วยไฟฟ้า, ระบบสายพานลำเลียง เป็นต้น และ ลิมิตสวิตช์)Limit switch) นั้นสามารถมีคอนแทคได้หลายอันมีคอนแทคปกติปิดและปกติเปิดมีโครงสร้างคล้ายสวิตช์ปุ่มกด

ข้อดีของสวิตช์จำกัดระยะ)Limit switch)

- ติดตั้งง่าย สะดวกต่อการใช้งาน
- ไม่ต้องมีไฟเลี้ยงวงจรในการทำงาน
- การทำงานเชื่อถือได้ มีความแม่นยำในการทำงาน
- ราคาต่ำกว่าอุปกรณ์ตรวจจับชนิดอื่น

บทที่ 3
วิธีการดำเนินงาน

คุณสมบัติของหุ่นยนต์สำรวจ

- สามารถสำรวจได้ทุกสภาพพื้นผิว บริเวณที่ซึ่งมีความเสี่ยงต่อการสำรวจของมนุษย์หรือที่ซึ่งมนุษย์เข้าถึงได้ยาก
- สามารถควบคุมการทำงานได้ในระยะไกลผ่านการควบคุมของวิทยุบังคับไร้สาย

3.1 แผนการดำเนินงาน

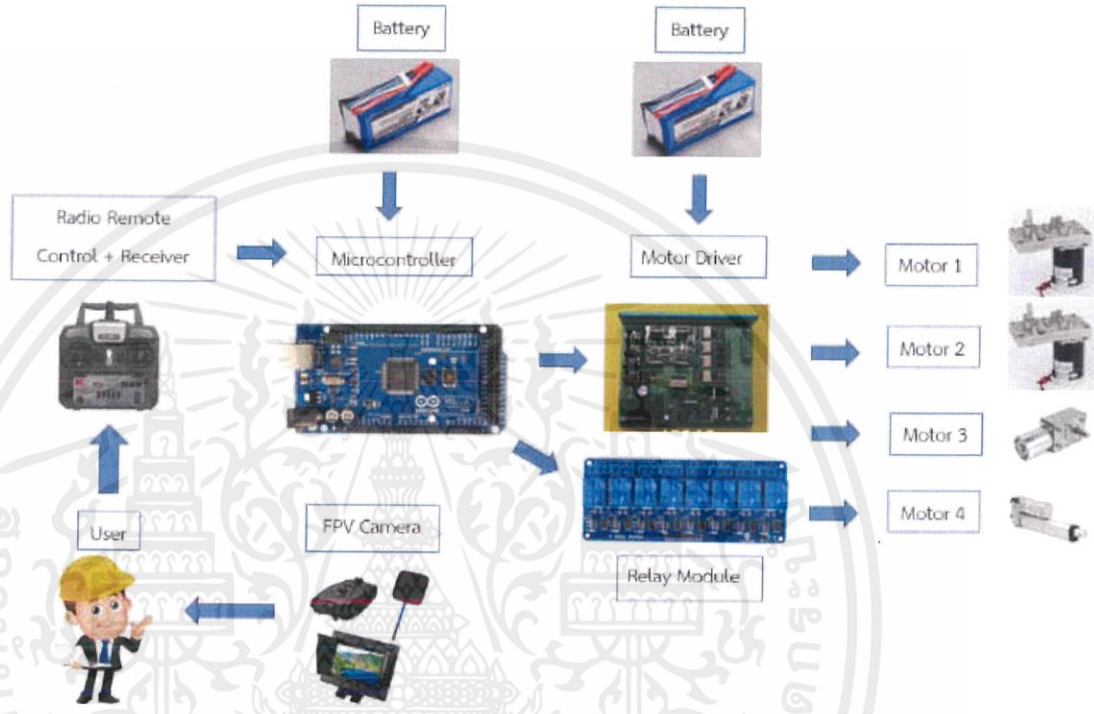
การสร้างหุ่นยนต์สำรวจได้มีการวางแผนการออกแบบและดำเนินการสร้างและระยะเวลาในการดำเนินงาน ซึ่งได้สรุปไว้ในตารางแสดงผลการดำเนินงานของโครงงานนี้

ระยะเวลา / ดำเนินงาน / ขั้นตอนการดำเนินงาน	เดือน			
	มกราคม	กุมภาพันธ์	มีนาคม	เมษายน
ศึกษาโครงการที่จะพัฒนาต่อ	↔			
เข้าพบอาจารย์เพื่อขอคำแนะนำ	←			→
ค้นคว้าข้อมูลเพิ่มเติมเกี่ยวกับโครงงาน	←	→		
จัดทำรูปเล่มรายงาน				↔
ทำการออกแบบการส่วนของหุ่นยนต์ที่พัฒนาต่อ		↔	→	
สร้างส่วนของหุ่นยนต์ที่พัฒนาต่อ			←	→
เตรียมนำเสนอหัวข้อโครงงาน				↔

ตาราง 3.1 แผนการดำเนินงานของโครงงาน

3.2 การออกแบบตัวหุ่นยนต์

ส่วนของการออกแบบและสร้างการหุ่นยนต์สำรวจผ่านรีโมทบังคับวิทยุ มีการออกแบบการทำงานโดยแบ่งออกเป็น2ส่วน คือการออกแบบด้าน Hardwareโดยมีหลักการทำงานของระบบ ดังรูป



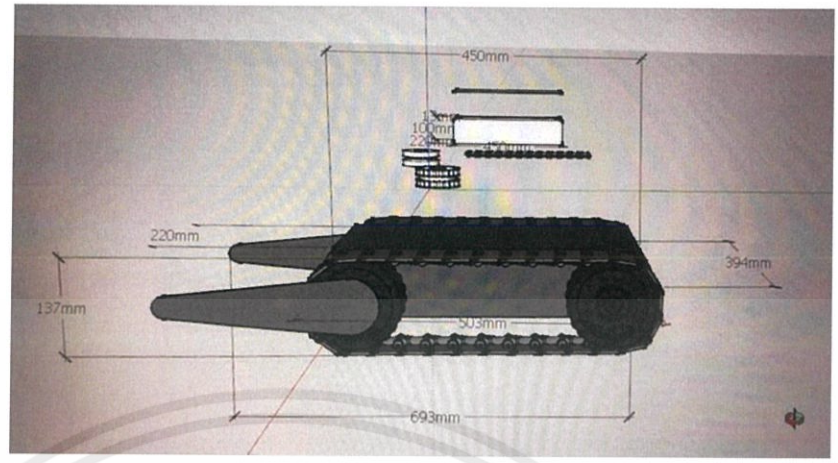
รูปที่ 3.1 หลักการทำงานของหุ่นยนต์สำรวจ

โดยเมื่อผู้ใช้สั่งการไปยังรีโมต ภายในตัวส่งของรีโมตจะส่งไปยังตัวรับของรีโมตเพื่อส่งสัญญาณต่อไปให้ยังไมโครคอนโทรลเลอร์ และเมื่อไมโครคอนโทรลเลอร์ได้รับสัญญาณก็จะส่งต่อไปยังบอร์ดไดร์ฟเพื่อให้บอร์ดไดร์ฟทำงานควบคุมมอเตอร์ต่างๆ และเมื่อกล้องได้รับสัญญาณภาพจะส่งไปยังจอแสดงภาพ

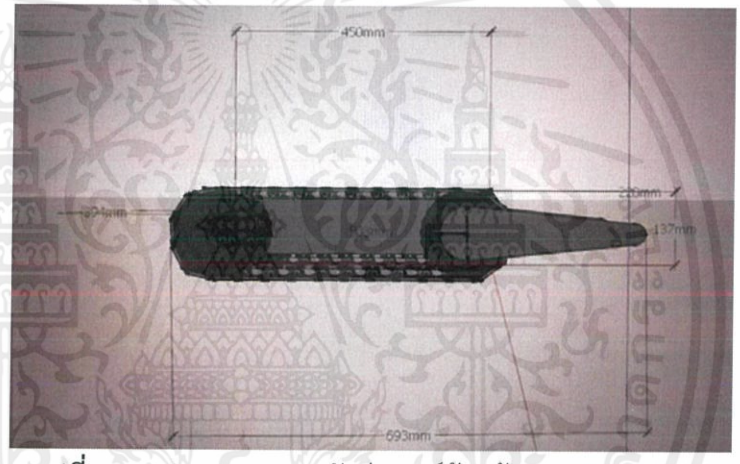
3.2.1. การออกแบบด้านฮาร์ดแวร์

การออกแบบตัวหุ่นยนต์โดยใช้โปรแกรม Sketch up

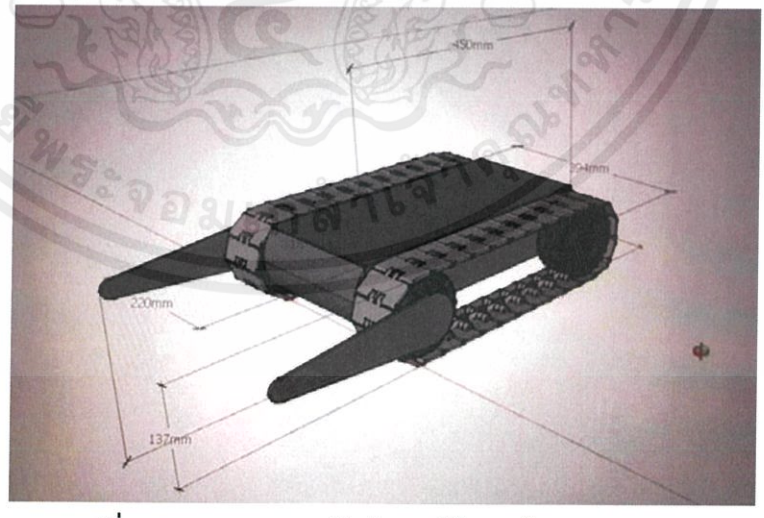
โดยออกแบบให้ตัวหุ่นยนต์มีขนาดกว้าง 394 mm ยาว 450 mm



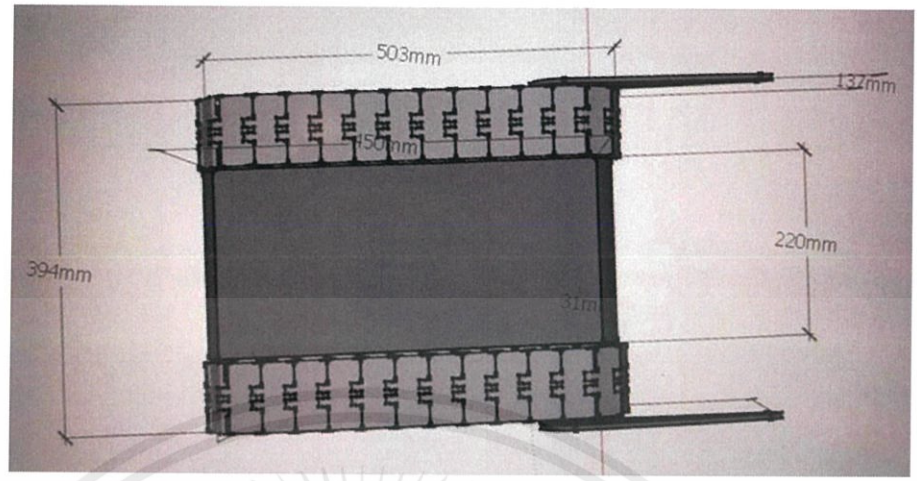
รูปที่ 3.2 รูปการออกแบบตัวหุ่นยนต์ด้านข้าง



รูปที่ 3.3 รูปการออกแบบตัวหุ่นยนต์ด้านข้าง



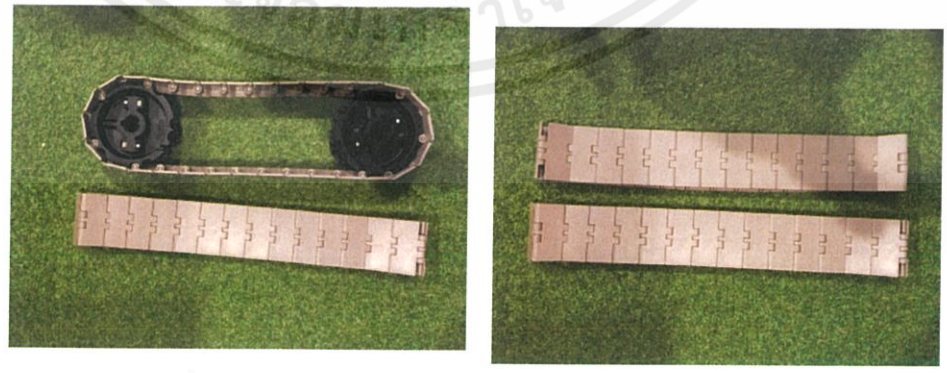
รูปที่ 3.4 การออกแบบตัวหุ่นยนต์ด้านหน้า



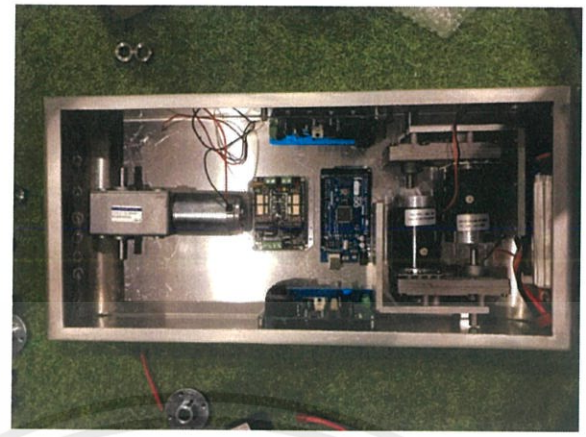
รูปที่ 3.5 การออกแบบตัวหุ่นยนต์ด้านบน

3.3. ขั้นตอนการทำโครงงาน

รูปที่ 3.6 นำแผ่นยางมาติดกับ Table top chain เพื่อเพิ่มแรงเสียดทาน

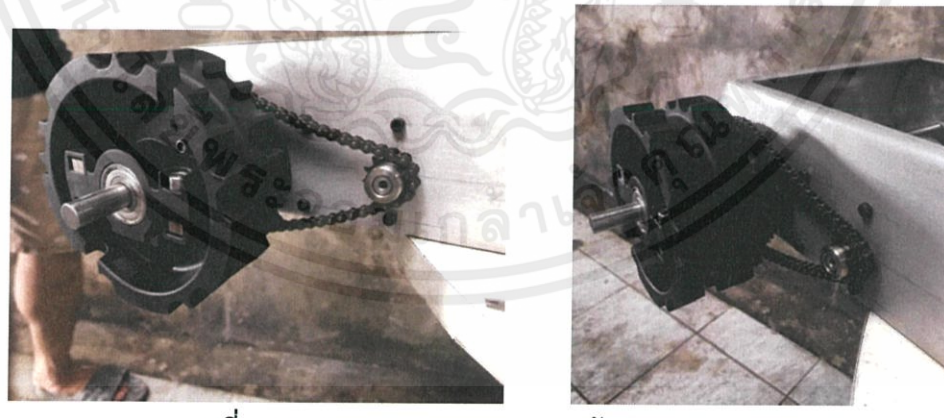


รูปที่ 3.7 นำTable top chain มาประกอบเข้ากับ Sprocket



รูปที่ 3.8 ลองวางอุปกรณ์ลงบนกล่อง

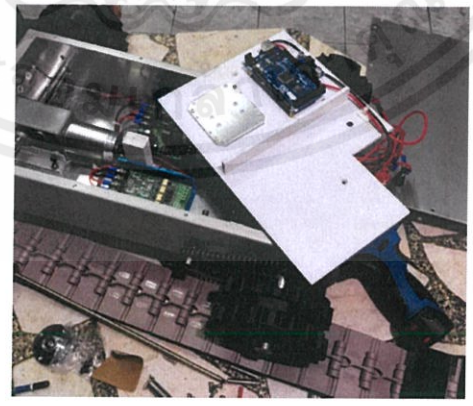
รูปที่ 3.9 เจาะรูใส่แกน Sprocket หน้า-หลัง



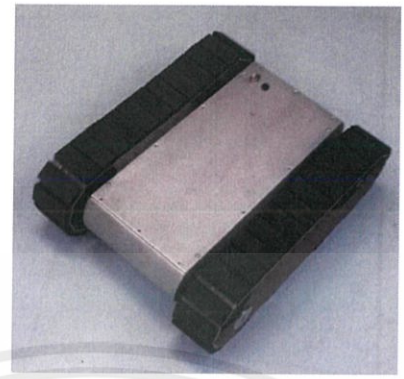
รูปที่ 3.10 ประกอบ Motor Gear เข้ากับ Sprocket

รูปที่ 3.11 ติดตั้งบอร์ด Drive

รูปที่ 3.12 ติดตั้งบอร์ด Controller



รูปที่ 3.13 เดินสายไฟ



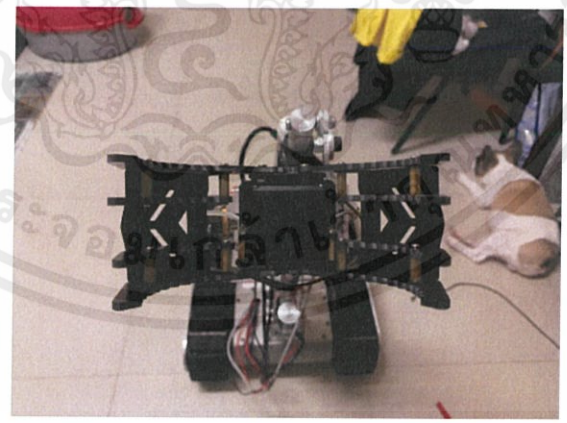
รูปที่ 3.14 ประกอบเสร็จ ปิดฝา

รูปที่ 3.15 ทดลองรองรับโหลด
(หมายเหตุ: ผู้ทดลองหนัก 70 กิโลกรัม)

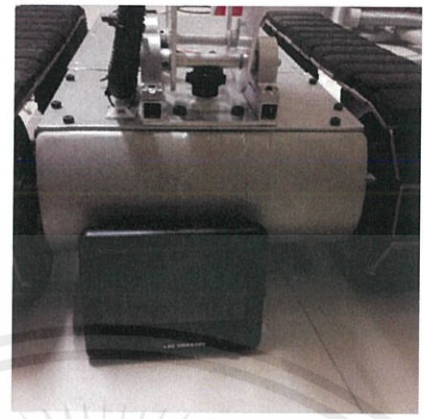
รูปที่ 3.16 ออกแบบส่วนของแขน

รูปที่ 3.17 ส่วนประกอบของแขน

รูปที่ 3.18 เมื่อประกอบแขนเสร็จ



รูปที่ 3.19 มือของหุ่นยนต์



รูปที่ 3.20 ติดกล้องและจอแสดงภาพ

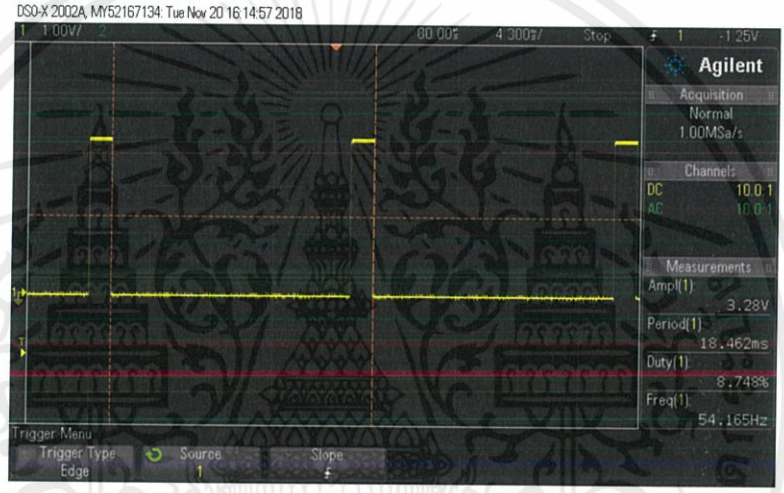
บทที่ 4

ผลการทดลอง

4.1 การอ่าน PWM Signals จาก Receiver ด้วย Arduino

โปรแกรมบอร์ดคอนโทรล Arduino ให้ควบคุมให้มอเตอร์เกิดการหมุนโดยวัดค่าอินพุทที่สามารถทำให้มอเตอร์สามารถหมุนได้

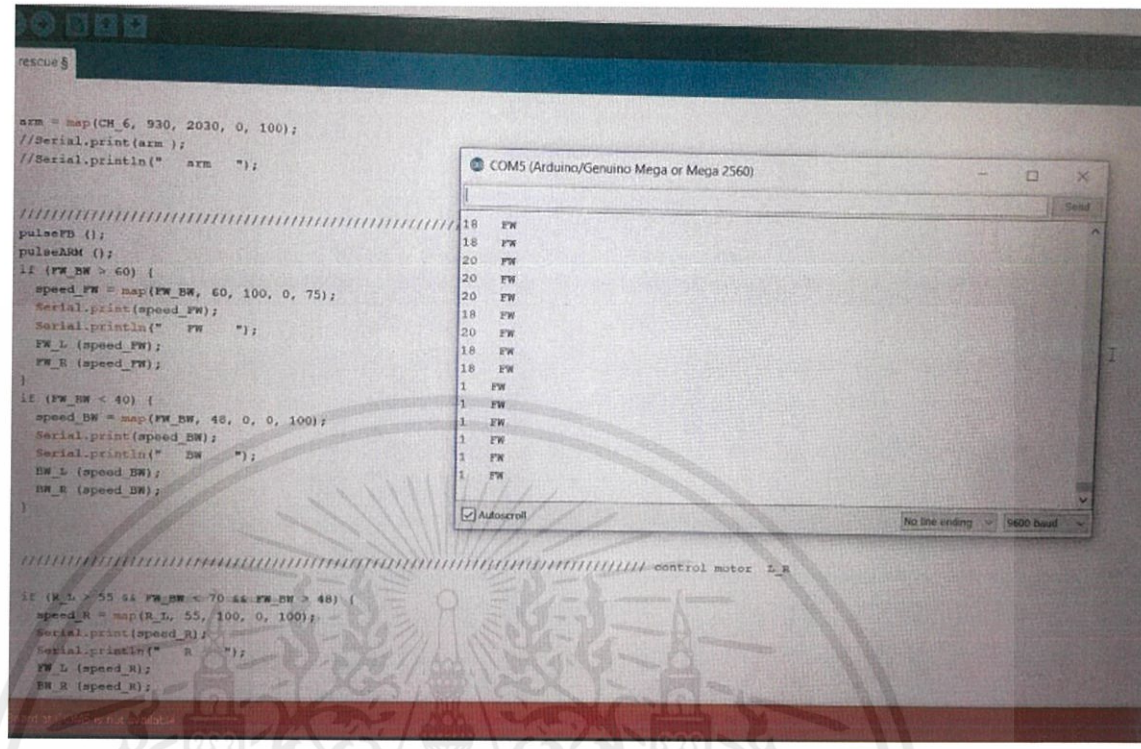
จากรูปที่ 4.1) คาบ = 18.462 ms และ Duty Cycle = 8.748% ซึ่งหมายความว่า Pulse width = 18.4628*0.08748 = 1.615 ms



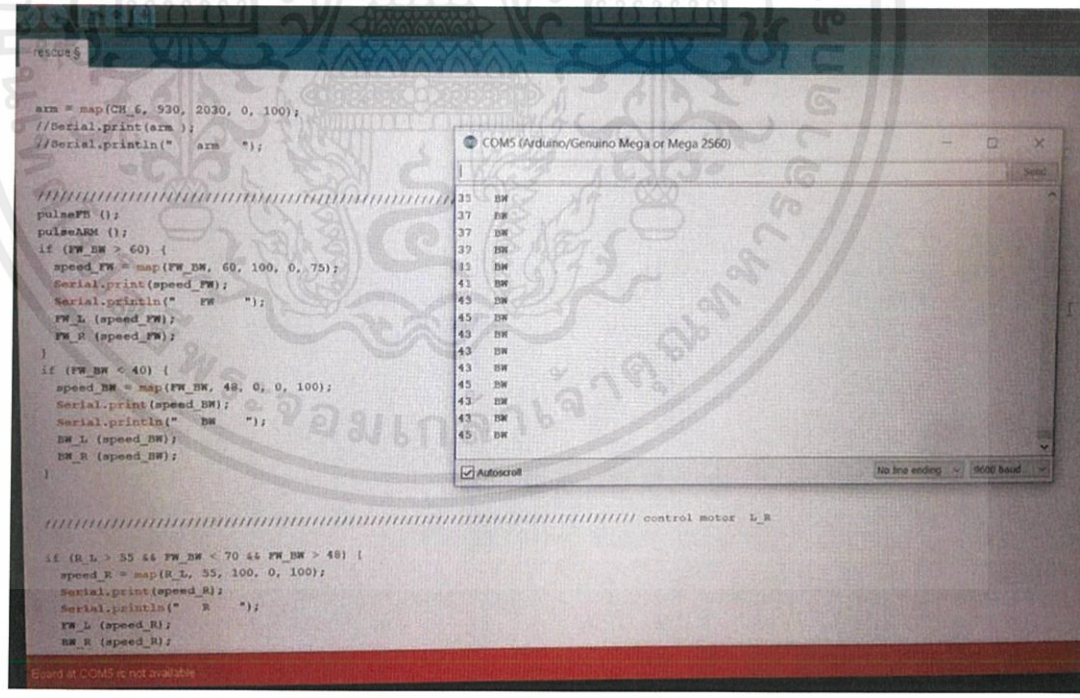
รูปที่ 4.1 สัญญาณตัว Receive ที่ Port A_0 (Channel1) สั่งให้หุ่นยนต์หันซ้าย-ขวา

จากรูปที่ 4.2) คาบ = 18.457 ms และ Duty Cycle = 7.683% ซึ่งหมายความว่า Pulse width = 18.457 *0.07683 = 1.418 ms

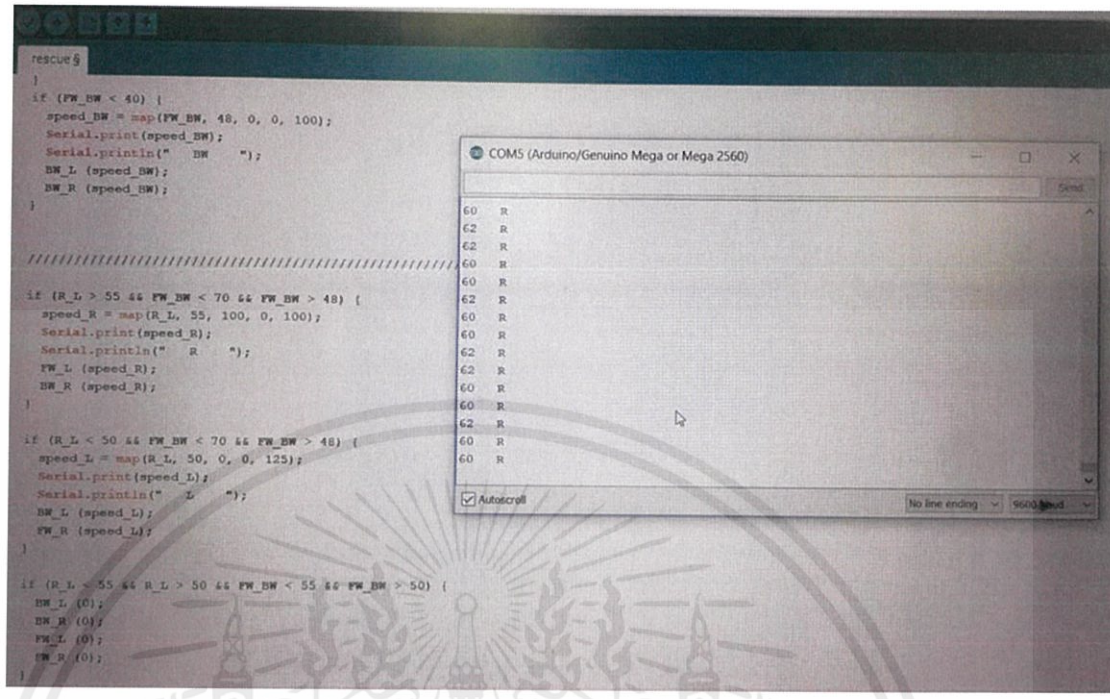
รูปที่ 4.2 สัญญาณตัว Receive ที่ Port A_1 (Channel2) สั่งให้หุ่นยนต์เดินหน้าและถอยหลัง



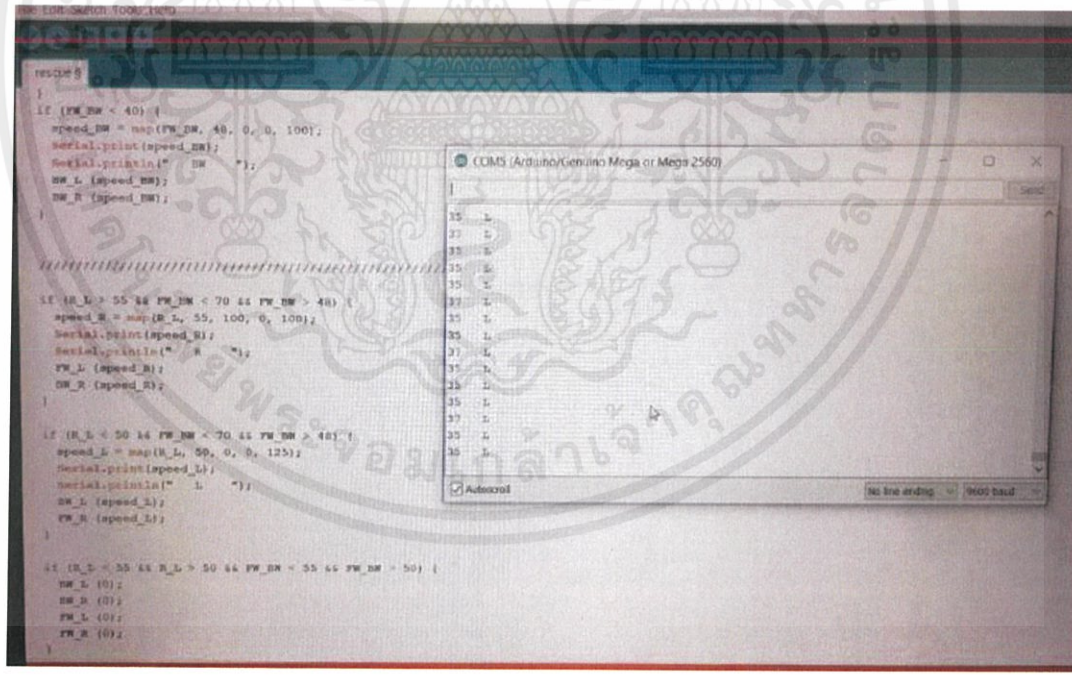
รูปที่ 4.3 เมื่อสั่งให้หุ่นยนต์เดินหน้า



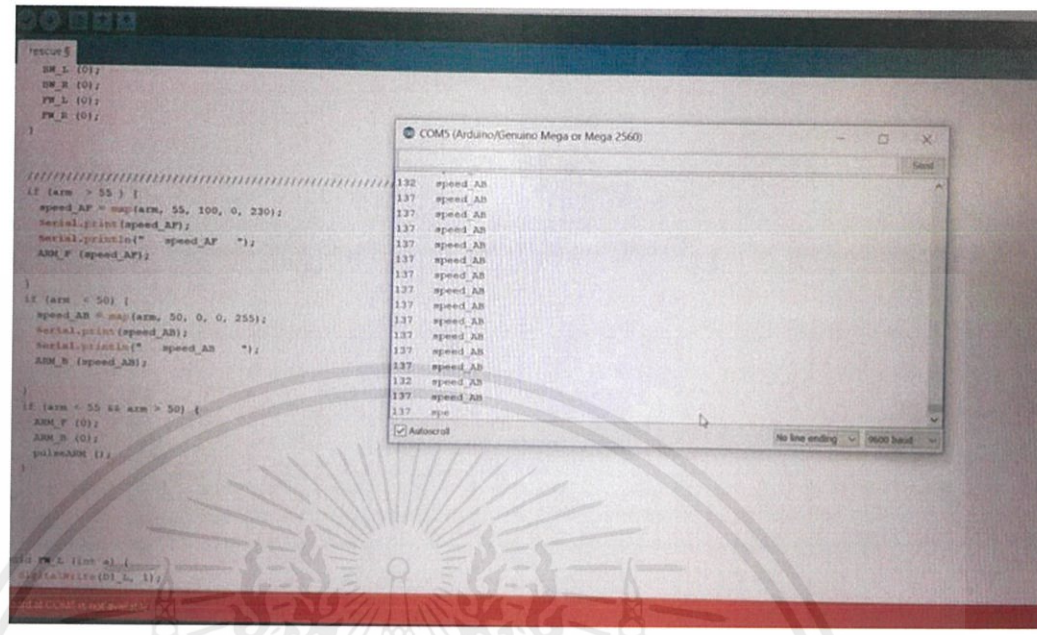
รูปที่ 4.4 เมื่อสั่งให้หุ่นยนต์ถอยหลัง



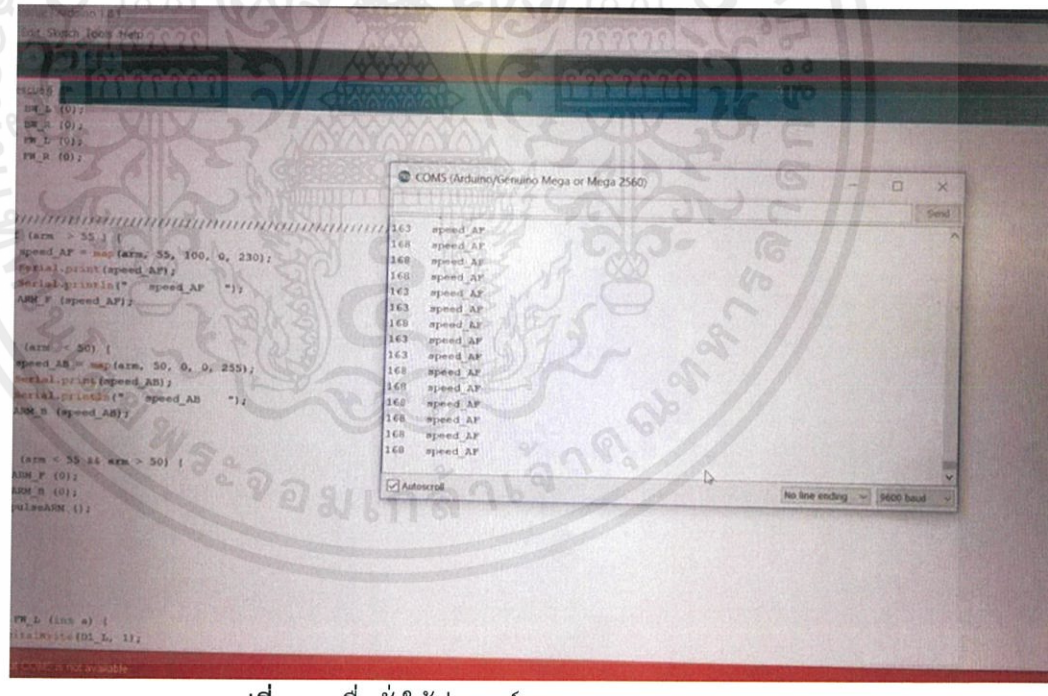
รูปที่ 4.5 เมื่อสั่งให้หุ่นยนต์เลี้ยวขวา



รูปที่ 4.6 เมื่อสั่งให้หุ่นยนต์เลี้ยวซ้าย



รูปที่ 4.7 เมื่อสั่งให้หุ่นยนต์แขนยกขึ้น



รูปที่ 4.8 เมื่อสั่งให้หุ่นยนต์แขนยกลง

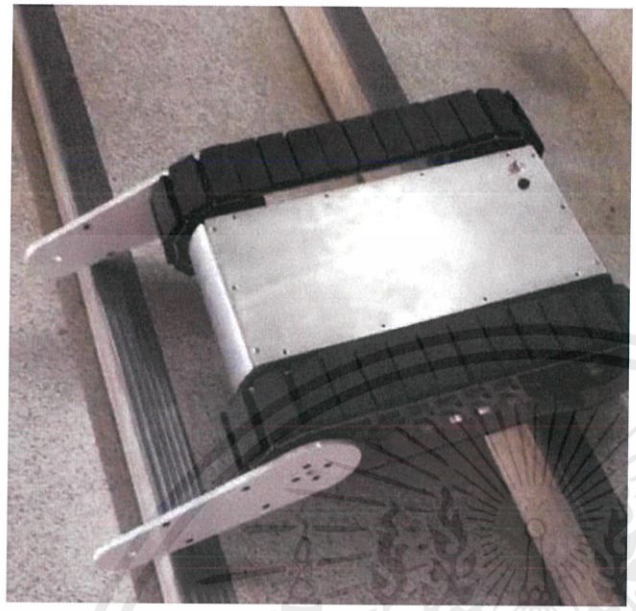
4.2 ผลการกินกระแส

จุดวัดการกินกระแส	กระแส (A)
ชุดขับเคลื่อนของหุ่นยนต์ : เดินหน้า/ถอยหลัง	1.8
ชุดขับเคลื่อนของหุ่นยนต์ : เลี้ยวซ้าย/เลี้ยวขวา	1.4
ชุดแขนของหุ่นยนต์ : มอเตอร์ตัวที่ 1	0.26
ชุดแขนของหุ่นยนต์ : มอเตอร์ตัวที่ 2	0.3
ชุดแขนของหุ่นยนต์ : มอเตอร์ตัวที่ 3	0.27

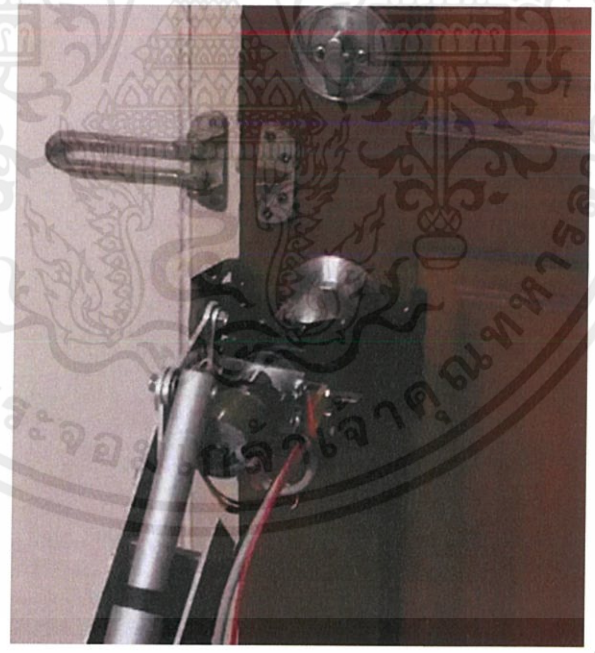
ตารางที่ 4.1 ผลการกินกระแส

4.3 ผลการทดลองตามเป้าหมาย

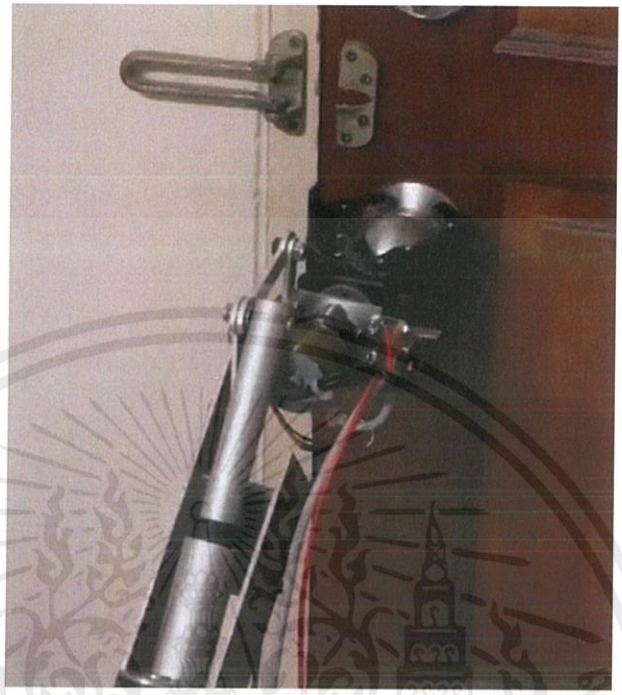
รูปที่ 4.9 สามารถเดินบนพื้นราบได้



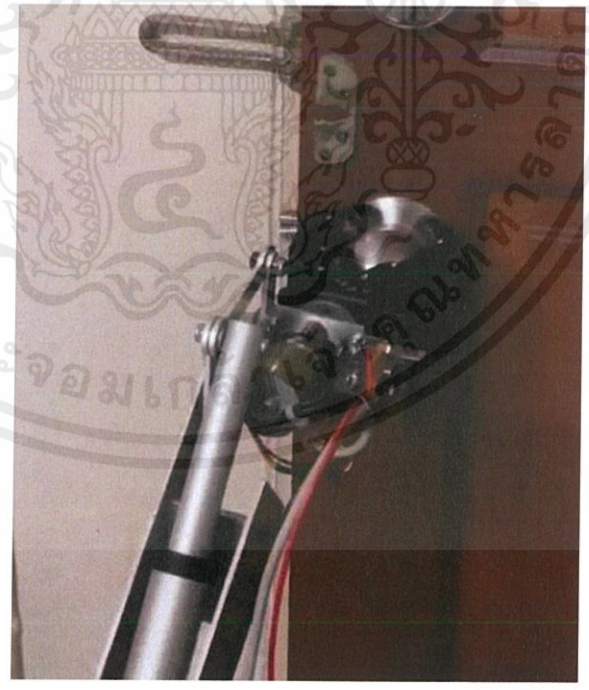
รูปที่ 4.10 สามารถเดินขึ้น-ลงบันไดได้



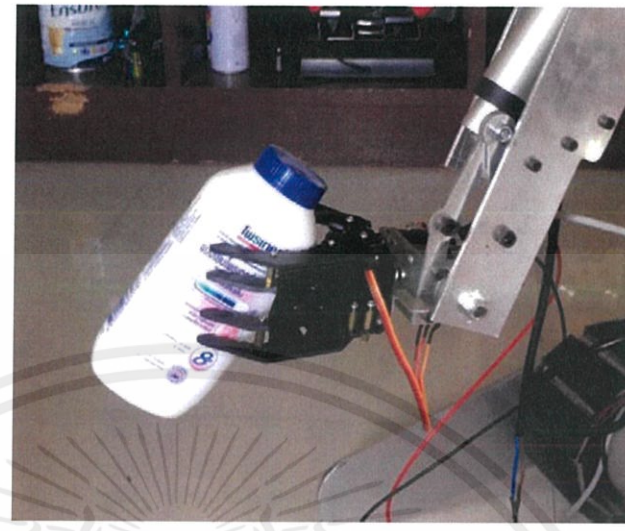
รูปที่ 4.11 สามารถบังคับแขนได้



รูปที่ 4.12 แขนสามารถจับลูกบิดได้



รูปที่ 4.13 แขนสามารถดึงลูกบิด เพื่อเปิดประตูได้



รูปที่ 4.14 แขนสามารถหยิบจับวัตถุได้

รูปที่ 4.15 มีจอแสดงภาพ

บทที่ 5

สรุปผลการทดลอง

5.1 สรุปผลการทดลอง

จากการทดลองเพื่อทดสอบการทำงานของหุ่นยนต์สำรวจ พบว่าเครื่องสามารถทำงานได้ตามที่ต้องการคือหุ่นยนต์สามารถเคลื่อนที่โดยการเดินหน้า, ถอยหลัง, เลี้ยวซ้าย, เลี้ยวขวา และหุ่นยนต์สามารถเดินได้ทุกสภาพพื้นผิว เช่น ทางลาดชัน,ขึ้นลงบันไดและแขนของหุ่นยนต์สามารถหยิบวัตถุ , เปิดประตูได้

5.2 ปัญหาและอุปสรรค

จากการทดลองพบปัญหาคือ

- งบประมาณไม่เพียงพอ ไม่สามารถทำงานได้อย่างเต็มที่
- อุปกรณ์เสียหายระหว่างประกอบชิ้นส่วนและไม่สามารถไปซื้อได้ทำให้สูญเสียเวลา
- ได้ของที่สั่งจากโรงกลึงช้า
- ระยะเวลาในการทำงานกระชั้นชิด

5.3 ข้อเสนอแนะ

- ควรสั่งซื้อของให้รวดเร็วกว่านี้
- ควรระมัดระวังเพื่อป้องกันอุปกรณ์เสียหาย
- ควรซื้ออุปกรณ์มาเผื่อการเสียหาย

เอกสารอ้างอิง

[1] "Reading PWM Signals From An RC Receiver With Arduino" .(ระบบออนไลน์) แหล่งที่มา: http://www.camelsoftware.com/2015/12/25/reading-pwm-signals-from-an-rc-receiver-with-arduino 20 พฤศจิกายน 2561

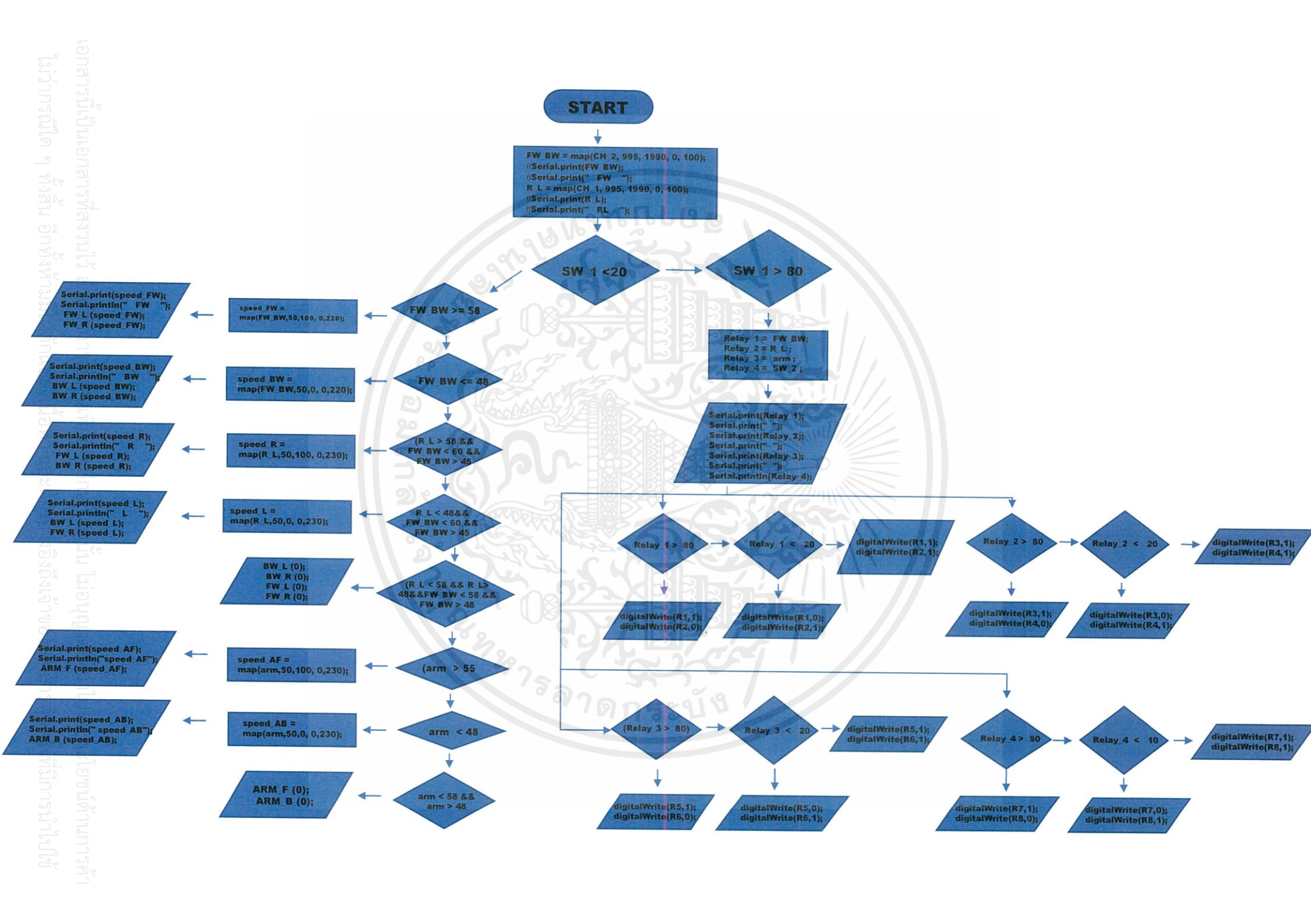
[2] "บอร์ด arduino mega R3" แหล่งที่มา: http://mbeddedweekly.blogspot.com/2014/08/arduino-mega2560.html 20 พฤศจิกายน 2561

[3] "รีโมตรถบังคับกับ POPBOT-XT" .(ระบบออนไลน์) แหล่งที่มา: http://doc.inex.co.th/rc-remotecontrol-with-popbot-xt 20 พฤศจิกายน 2561

[4] "H-BRIDGE DRIVER DC MOTOR 80A" .(ระบบออนไลน์) แหล่งที่มา: https://www.nattakit.com/16770963/h-bridge-driver-dc-motor-80a 20 พฤศจิกายน 2561

[5] "**การควบคุม RC Servo Motor ด้วย Arduino**" (ระบบออนไลน์) แหล่งที่มา : https://www.thaieasyelec.com/article-wiki/review-product-article/%E0%B8%9A%E0%B8%97%E0%B8%84%E0%B8%A7%E0%B8%B2%E0%B8%A1%E0%B8%95%E0%B8%B1%E0%B8%A7%E0%B8%AD%E0%B8%A2%E0%B9%88%E0%B8%B2%E0%B8%87%E0%B8%81%E0%B8%B2%E0%B8%A3%E0%B8%84%E0%B8%A7%E0%B8%9A%E0%B8%84%E0%B8%B8%E0%B8%A1-rc-servo-motor-%E0%B8%94%E0%B9%89%E0%B8%A7%E0%B8%A2-arduino.html 20 พฤศจิกายน 2561

ภาคผนวก


START
FW_BW = map(CH_2, 995, 1990, 0, 100);
//Serial.print(FW_BW);
//Serial.print(" FW ");
R_L = map(CH_1, 995, 1990, 0, 100);
//Serial.print(R_L);
//Serial.print(" RL ");
SW_1 <20
SW_1 > 80
FW_BW >= 58
speed_FW = map(FW_BW,50,100, 0,220);
Serial.print(speed_FW);
Serial.println(" FW ");
FW_L (speed_FW);
FW_R (speed_FW);
FW_BW <= 48
speed_BW = map(FW_BW,50,0, 0,220);
Serial.print(speed_BW);
Serial.println(" BW ");
BW_L (speed_BW);
BW_R (speed_BW);
(R_L > 58 && FW_BW < 60 && FW_BW > 45
speed_R = map(R_L,50,100, 0,230);
Serial.print(speed_R);
Serial.println(" R ");
FW_L (speed_R);
BW_R (speed_R);
R_L < 48&& FW_BW < 60 && FW_BW > 45
speed_L = map(R_L,50,0, 0,230);
Serial.print(speed_L);
Serial.println(" L ");
BW_L (speed_L);
FW_R (speed_L);
(R_L < 58 && R_L> 48&&FW_BW < 58 && FW_BW > 48
BW_L (0);
BW_R (0);
FW_L (0);
FW_R (0);
(arm > 55
speed_AF = map(arm,50,100, 0,230);
Serial.print(speed_AF);
Serial.println("speed_AF");
ARM_F (speed_AF);
arm < 48
speed_AB = map(arm,50,0, 0,230);
Serial.print(speed_AB);
Serial.println(" speed_AB");
ARM_B (speed_AB);
arm < 58 && arm > 48
ARM_F (0);
ARM_B (0);
Relay_1 = FW_BW;
Relay_2 = R_L ;
Relay_3 = arm ;
Relay_4 = SW_2 ;
Serial.print(Relay_1);
Serial.print(" ");
Serial.print(Relay_2);
Serial.print(" ");
Serial.print(Relay_3);
Serial.print(" ");
Serial.println(Relay_4);
Relay_1 > 80
Relay_1 < 20
digitalWrite(R1,1);
digitalWrite(R2,1);
digitalWrite(R1,1);
digitalWrite(R2,0);
digitalWrite(R1,0);
digitalWrite(R2,1);
Relay_2 > 80
Relay_2 < 20
digitalWrite(R3,1);
digitalWrite(R4,1);
digitalWrite(R3,1);
digitalWrite(R4,0);
digitalWrite(R3,0);
digitalWrite(R4,1);
(Relay_3 > 80)
Relay_3 < 20
digitalWrite(R5,1);
digitalWrite(R6,1);
digitalWrite(R5,1);
digitalWrite(R6,0);
digitalWrite(R5,0);
digitalWrite(R6,1);
Relay_4 > 90
Relay_4 < 10
digitalWrite(R7,1);
digitalWrite(R8,1);
digitalWrite(R7,1);
digitalWrite(R8,0);
digitalWrite(R7,0);
digitalWrite(R8,1);

โปรแกรมควบคุม

```
#define CH1 A0        // REMOTE
#define CH2 A1
#define CH3 A2
#define CH4 A3
#define CH5 A4
#define CH6 A5

#define D2_L 3        // Motor L
#define D1_L 4
#define PWM_L 5

#define D1_R 7        // Motor R
#define D2_R 8
#define PWM_R 6

#define D2_A 9        // Motor ARM
#define D1_A 10
#define PWM_A 11

#define R1 22         //  relay ch1
#define R2 24      //  relay ch2
#define R3 26      //  relay ch3
#define R4 28      //  relay ch4
#define R5 30      //  relay ch5
```

```
#define R6 32    //  relay ch6
#define R7 34    //  relay ch7
#define R8 36    //  relay ch8

unsigned int SW_1;
unsigned int SW_2;

int CH_1 = 0;
int CH_2 = 0;
int CH_3 = 0;
int CH_4 = 0;
int CH_5 = 0;
int CH_6 = 0;

unsigned int FW_BW = 0;
unsigned int R_L = 0;
unsigned int arm = 0;
unsigned int Relay_1 ;
unsigned int Relay_2 ;
unsigned int Relay_3 ;
unsigned int Relay_4 ;

unsigned int speed_FW;
unsigned int speed_BW;
unsigned int speed_L;
```

```
unsigned int speed_R;

unsigned int speed_AF;
unsigned int speed_AB;

void setup() {
 pinMode(CH1,INPUT);
 pinMode(CH2,INPUT);
 pinMode(CH3,INPUT);
 pinMode(CH4,INPUT);
 pinMode(CH5,INPUT);
 pinMode(CH6,INPUT);

 pinMode(D1_L,OUTPUT);
 pinMode(D2_L,OUTPUT);
 pinMode(PWM_L,OUTPUT);

 pinMode(D1_R,OUTPUT);
 pinMode(D2_R,OUTPUT);
 pinMode(PWM_R,OUTPUT);

 pinMode(D1_A,OUTPUT);
 pinMode(D2_A,OUTPUT);
 pinMode(PWM_A,OUTPUT);
```

```
  pinMode(R1,OUTPUT);
  pinMode(R2,OUTPUT);
  pinMode(R3,OUTPUT);
  pinMode(R4,OUTPUT);
  pinMode(R5,OUTPUT);
  pinMode(R6,OUTPUT);
  pinMode(R7,OUTPUT);
  pinMode(R8,OUTPUT);
  Serial.begin(115200);
}


void loop() {
CH_1 =  pulseIn(CH1, HIGH);  // relay 1
//Serial.print(CH_1);
//Serial.print(" 1    ");
CH_2 =  pulseIn(CH2, HIGH);  // ARM
//Serial.print(CH_2);
//Serial.print(" 2    ");

CH_3 =  pulseIn(CH3, HIGH);
//Serial.print(CH_3);
//Serial.print(" 3     ");

CH_4 =  pulseIn(CH4, HIGH);   //  FW_BW
```

```
//Serial.print(CH_4);
//Serial.print("  4    ");

CH_5 =  pulseIn(CH5, HIGH);
//Serial.print(CH_5);
//Serial.print(" 5   ");

CH_6 =  pulseIn(CH6, HIGH);  // R_L
//Serial.print(CH_6);
//Serial.println(" 6    ");


FW_BW = map(CH_2, 995, 1990, 0, 100);
//Serial.print(FW_BW);
//Serial.print("   FW    ");

R_L = map(CH_1, 995, 1990, 0, 100);
//Serial.print(R_L);
//Serial.print("   RL    ");

arm = map(CH_4, 1001, 1990, 0, 100);
//Serial.print(arm );
//Serial.print("   arm   ");

SW_1 = map(CH_5, 995, 1990, 0, 100);
```

```
//Serial.print(SW_1);
//Serial.print("   SW_1   ");

SW_2 = map(CH_3, 995, 1990, 0, 100);
//Serial.print(SW_2);
//Serial.println("   SW_2   ");
//////////////////////////////////////////////////////////////////////////////////////// loop 1

//////////////////////////////////////////////////////////////////////////////////////// control
motor  FW_BW
if(SW_1 < 20)
{
if(FW_BW >= 58){
speed_FW = map(FW_BW,50,100, 0,220);
Serial.print(speed_FW);
Serial.println("   FW    ");
 FW_L (speed_FW);
 FW_R (speed_FW);
}
if(FW_BW <= 48){
speed_BW = map(FW_BW,50,0, 0,220);
Serial.print(speed_BW);
Serial.println("   BW    ");
 BW_L (speed_BW);
 BW_R (speed_BW);
```

```
}

////////////////////////////////////////////////////////////////////////////////// control
motor  L_R

if(R_L > 58 && FW_BW < 60 && FW_BW > 45){
speed_R = map(R_L,50,100, 0,230);
Serial.print(speed_R);
Serial.println("    R    ");
 FW_L (speed_R);
 BW_R (speed_R);
}

if(R_L < 48&& FW_BW < 60 && FW_BW > 45){
speed_L = map(R_L,50,0, 0,230);
Serial.print(speed_L);
Serial.println("    L    ");
 BW_L (speed_L);
 FW_R (speed_L);
}

if(R_L < 58 && R_L> 48&&FW_BW < 58 && FW_BW > 48){
 BW_L (0);
 BW_R (0);
 FW_L (0);
```

```
 FW_R (0);
}


////////////////////////////////////////////////////////////////////// ARM

if(arm  > 55 ){
speed_AF = map(arm,50,100, 0,230);
Serial.print(speed_AF);
Serial.println("   speed_AF   ");
 ARM_F (speed_AF);

}
if(arm  < 48){
speed_AB = map(arm,50,0, 0,230);
Serial.print(speed_AB);
Serial.println("   speed_AB    ");
ARM_B (speed_AB);

}
if(arm < 58 && arm > 48){
 ARM_F (0);
 ARM_B (0);
}
}
```

```
////////////////////////////////////////////////////////////////////// loop 2

else if(SW_1 > 80)
{
Relay_1 =  FW_BW;
Relay_2 = R_L ;
Relay_3 =  arm ;
Relay_4 =  SW_2 ;
Serial.print(Relay_1);
Serial.print("  ");
Serial.print(Relay_2);
Serial.print("  ");
Serial.print(Relay_3);
Serial.print("  ");
Serial.println(Relay_4);
////////////////////////////////////////////////////////////////////// relay 1
if (Relay_1 >  80){
digitalWrite(R1,1);
digitalWrite(R2,0);
}
else if(Relay_1  <   20){
digitalWrite(R1,0);
```

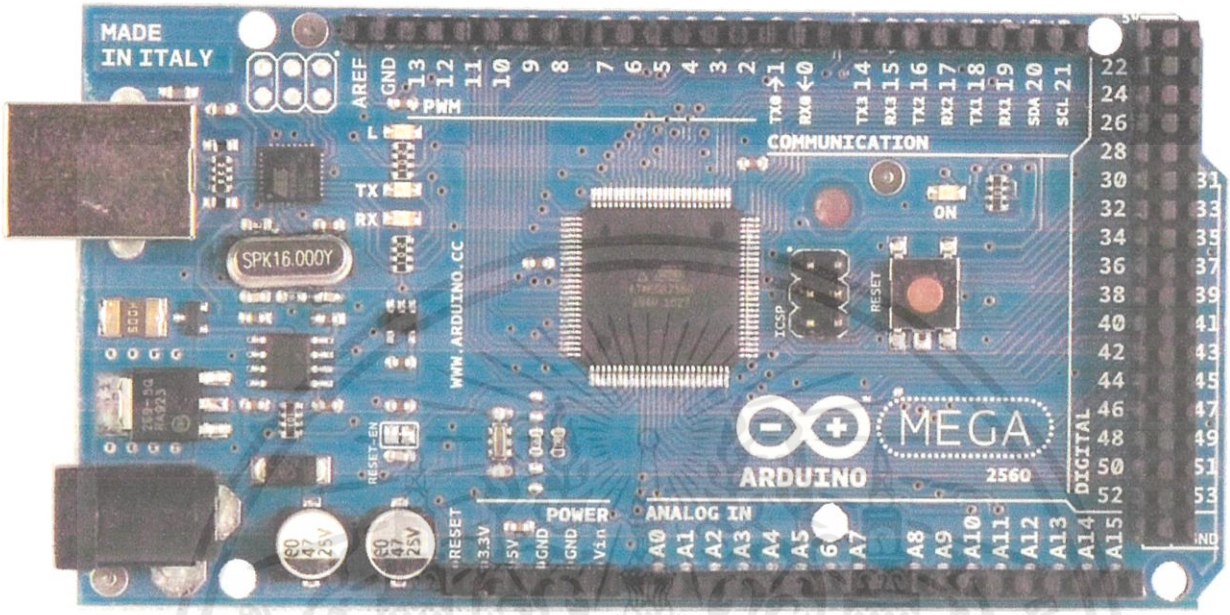
```
digitalWrite(R2,1);
}
else{
digitalWrite(R1,1);
digitalWrite(R2,1);
}

//////////////////////////////////////////////////////////////////// relay 2

if (Relay_2 >  80){
digitalWrite(R3,1);
digitalWrite(R4,0);
}
else if(Relay_2  <   20){
digitalWrite(R3,0);
digitalWrite(R4,1);
}
else{
digitalWrite(R3,1);
digitalWrite(R4,1);
}
//////////////////////////////////////////////////////////////////// relay 3

if (Relay_3 >  80){
digitalWrite(R5,1);
```

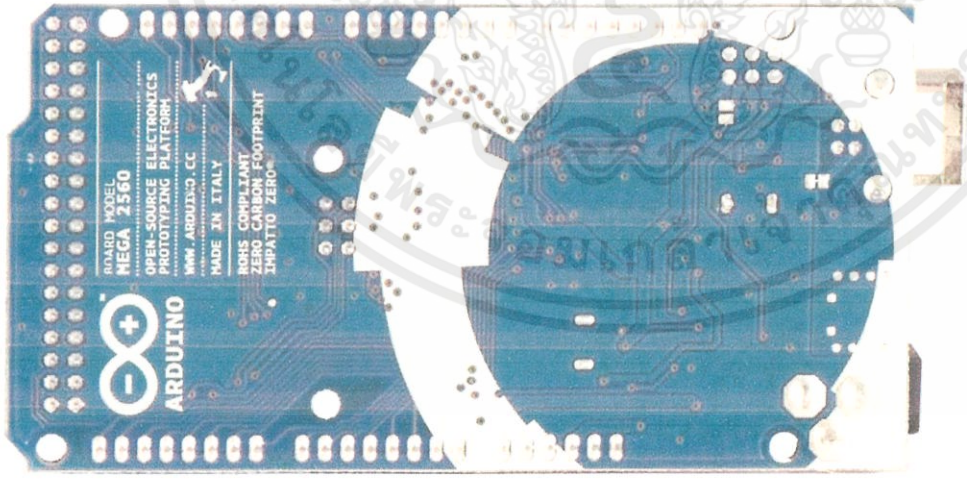
```
digitalWrite(R6,0);
}
else if(Relay_3  <   20){
digitalWrite(R5,0);
digitalWrite(R6,1);
}
else{
digitalWrite(R5,1);
digitalWrite(R6,1);
}
////////////////////////////////////////////////////////////////////// relay 4

if (Relay_4 >  90){
digitalWrite(R7,1);
digitalWrite(R8,0);
}
else if(Relay_4  <   10){
digitalWrite(R7,0);
digitalWrite(R8,1);
}
else{
digitalWrite(R7,1);
digitalWrite(R8,1);
}
}
```

```
}

void FW_L (int a){
digitalWrite(D1_L,1);
digitalWrite(D2_L,0);
analogWrite(PWM_L,a);

}

void BW_L (int b){
digitalWrite(D1_L,0);
digitalWrite(D2_L,1);
analogWrite(PWM_L,b);
}

void FW_R (int c){
digitalWrite(D1_R,1);
digitalWrite(D2_R,0);
analogWrite(PWM_R,c);
}

void BW_R (int d){
digitalWrite(D1_R,0);
digitalWrite(D2_R,1);
```

```
analogWrite(PWM_R,d);
}

void ARM_F (int e){
digitalWrite(D1_A,0);
digitalWrite(D2_A,1);
analogWrite(PWM_A,e);
}

void ARM_B (int f){
digitalWrite(D1_A,1);
digitalWrite(D2_A,0);
analogWrite(PWM_A,f);
}
```

Arduino Mega 2560





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.

Schematic & Reference Design

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

Power

The Arduino Mega 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 using the **SPI library**. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Uno, 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 or Diecimila.

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 Mega2560'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. For SPI communication, use the **SPI library**.

Programming

The Arduino Mega 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.

The ATmega8U2 firmware source code is available **in the Arduino repository**. The ATmega8U2 is loaded with a DFU bootloader, which can be activated by connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. 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 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 Mega2560 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

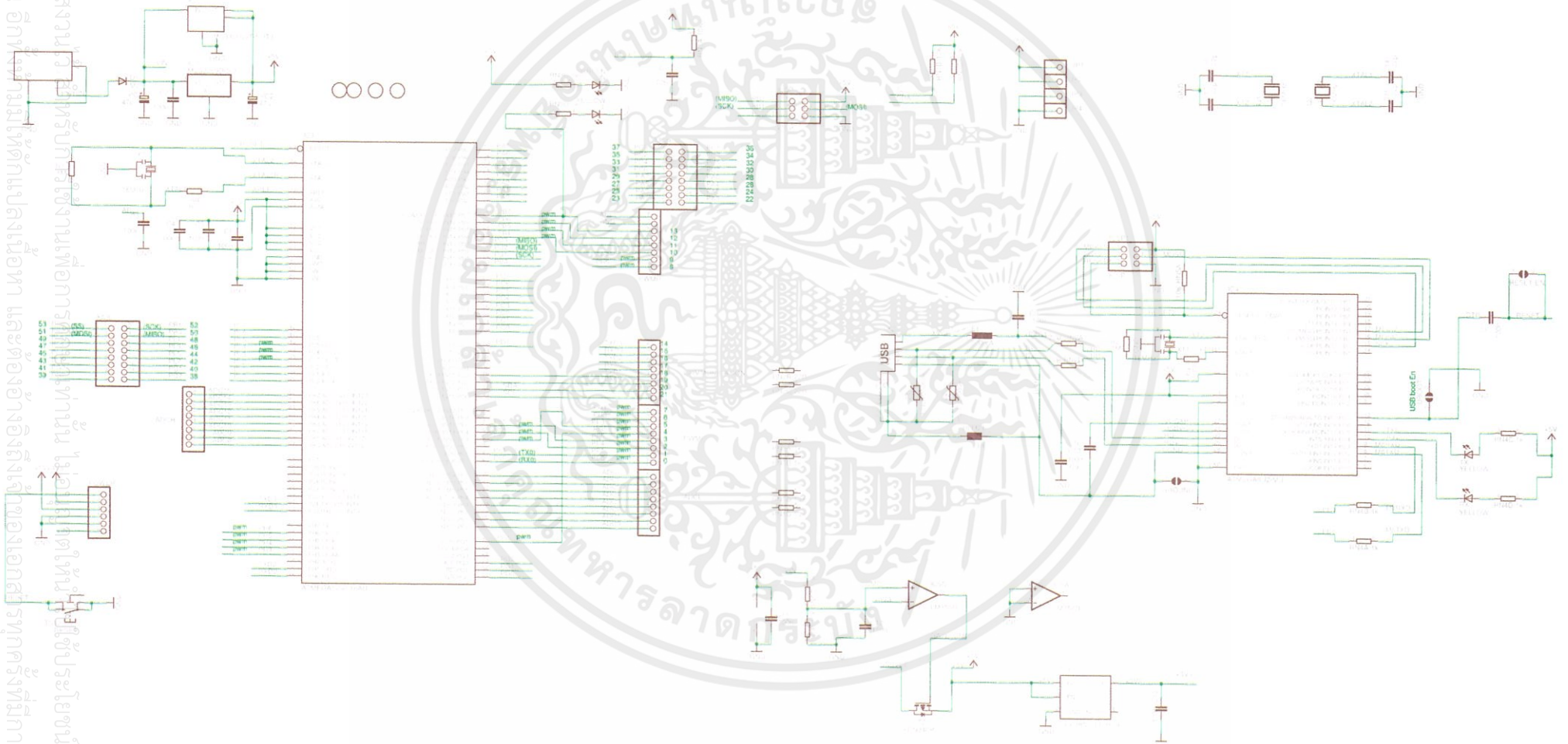
The Arduino Mega2560 has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics and Shield Compatibility

The maximum length and width of the Mega2560 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 Mega2560 is designed to be compatible with most shields designed for the Uno, 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 Mega2560 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).*

Arduino™ Mega 2560 Reference Design

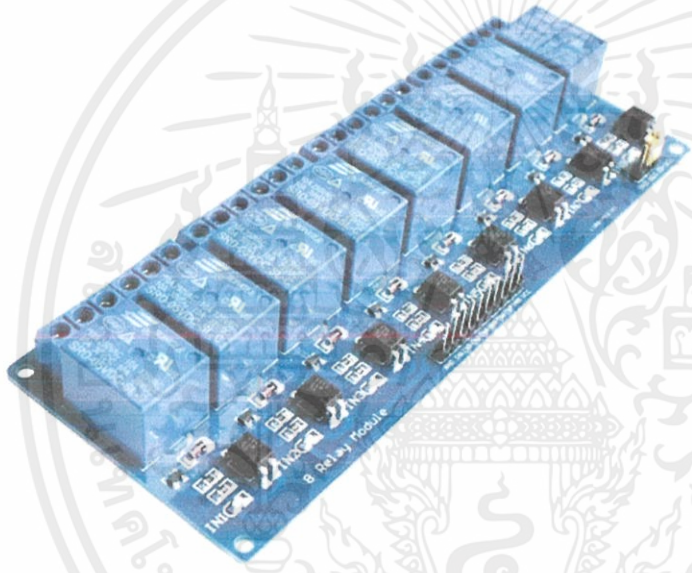


Handson Technology

User Guide

8 Channel 5V Optical Isolated Relay Module

This is a LOW Level 5V 8-channel relay interface board, and each channel needs a 15-20mA driver current. It can be used to control various appliances and equipment with large current. It is equipped with high-current relays that work under AC250V 10A or DC30V 10A. It has a standard interface that can be controlled directly by microcontroller. This module is optically isolated from high voltage side for safety requirement and also prevent ground loop when interface to microcontroller.



SKU: MDU1064

Brief Data:

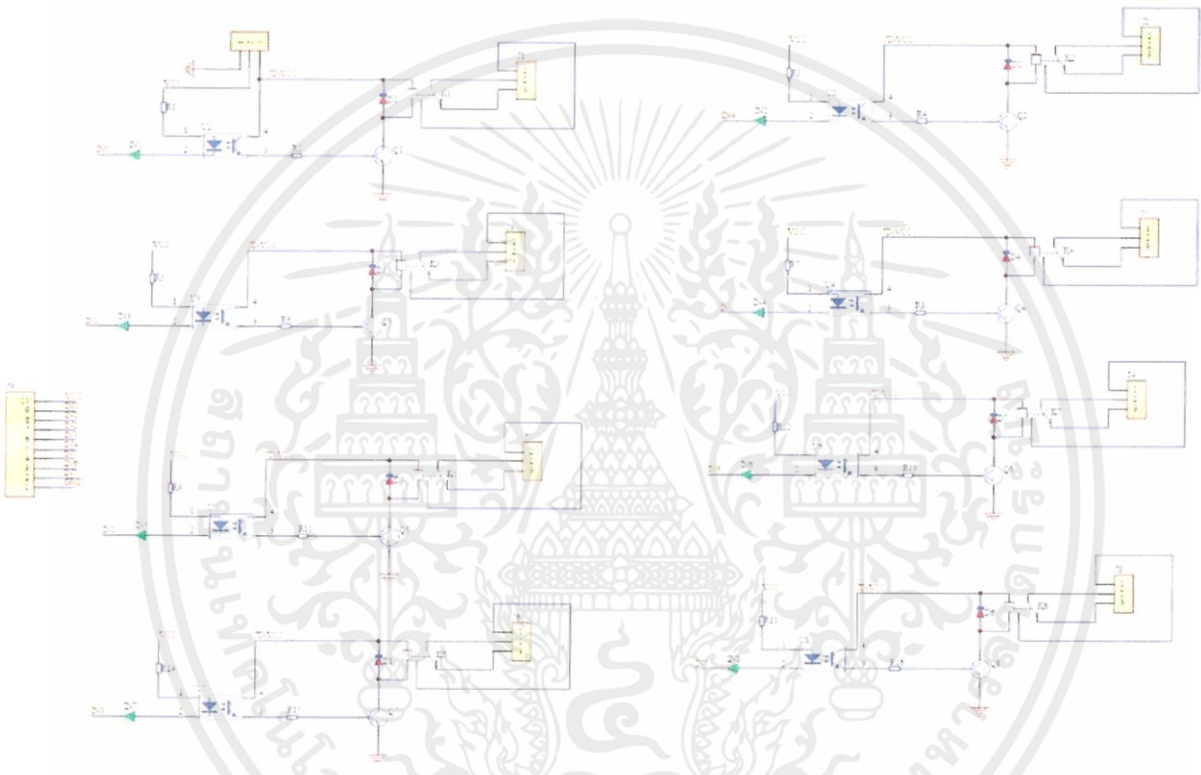
- Relay Maximum output: DC 30V/10A, AC 250V/10A.
- 8 Channel Relay Module with Opto-coupler. LOW Level Trigger expansion board, which is compatible with Arduino control board.
- Standard interface that can be controlled directly by microcontroller (8051, AVR, *PIC, DSP, ARM, ARM, MSP430, TTL logic).
- Relay of high quality low noise relays SPDT. A common terminal, a normally open, one normally closed terminal.
- Opto-Coupler isolation, for high voltage safety and prevent ground loop with microcontroller.
- Module Board: 138 x 56 mm.

Schematic:

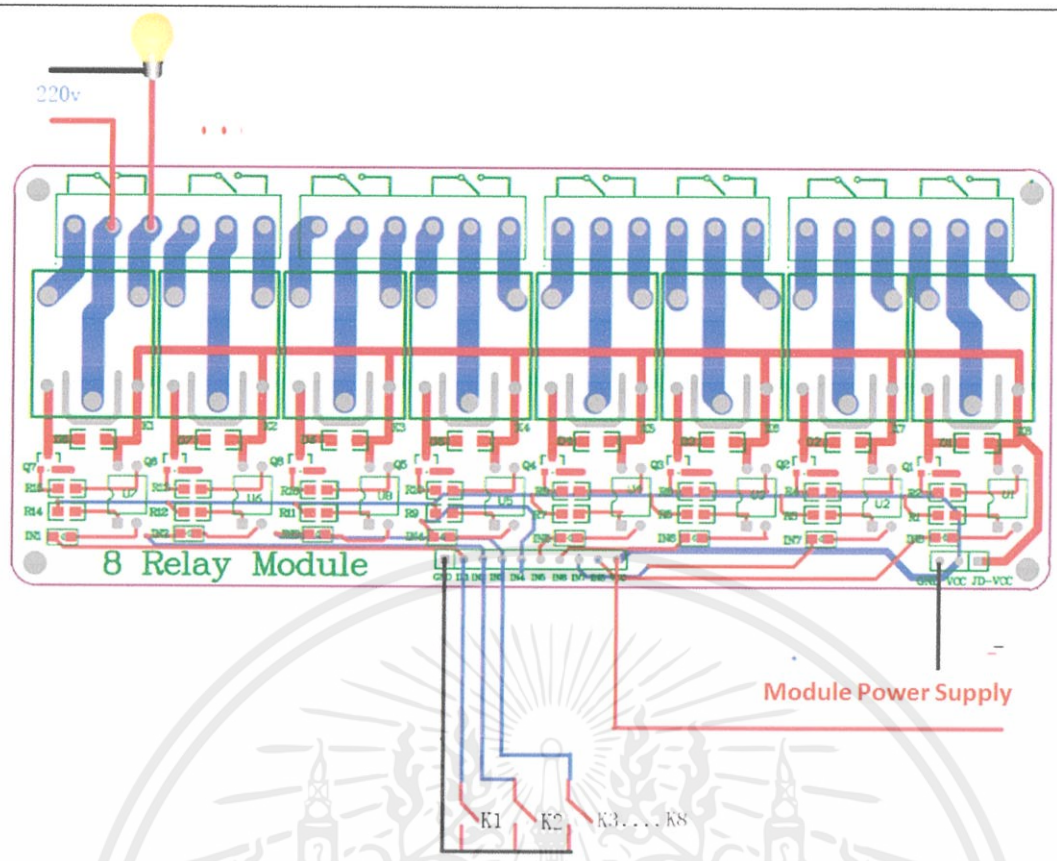
VCC and RY-VCC are also the power supply of the relay module. When you need to drive a large power load, you can take the jumper cap off and connect an extra power to RY-VCC to supply the relay; connect VCC to 5V of the MCU board to supply input signals.

NOTES: If you want complete optical isolation, connect "Vcc" to Arduino +5 volts but do NOT connect Arduino Ground. Remove the Vcc to JD-Vcc jumper. Connect a separate +5 supply to "JD-Vcc" and board Gnd. This will supply power to the transistor drivers and relay coils.

If relay isolation is enough for your application, connect Arduino +5 and Gnd, and leave Vcc to JD-Vcc jumper in place.



8 Channel Relay Module Schematic



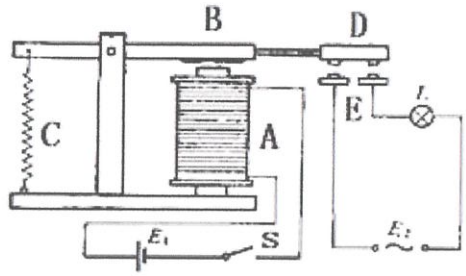
It is sometimes possible to use this relay boards with 3.3V signals, if the JD-VCC (Relay Power) is provided from a +5V supply and the VCC to JD-VCC jumper is removed. That 5V relay supply could be totally isolated from the 3.3V device, or have a common ground if opto-isolation is not needed. If used with isolated 3.3V signals, VCC (To the input of the opto-isolator, next to the IN pins) should be connected to the 3.3V device's +3.3V supply.

NOTE: Some Raspberry-Pi users have found that some relays are reliable and others do not actuate sometimes. It may be necessary to change the value of R1 from 1000 ohms to something like 220 ohms, or supply +5V to the VCC connection.

NOTE: The digital inputs from Arduino are Active LOW: The relay actuates and LED lights when the input pin is LOW, and turns off on HIGH.

Operating Principle:

See the picture below: A is an electromagnet, B armature, C spring, D moving contact, and E fixed contacts. There are two fixed contacts, a normally closed one and a normally open one. When the coil is not energized, the normally open contact is the one that is off, while the normally closed one is the other that is on.



Supply voltage to the coil and some currents will pass through the coil thus generating the electromagnetic effect. So the armature overcomes the tension of the spring and is attracted to the core, thus closing the moving contact of the armature and the normally open (NO) contact or you may say releasing the former and the normally closed (NC) contact. After the coil is de-energized, the electromagnetic force disappears and the armature moves back to the original position, releasing the moving contact and normally closed contact. The closing and releasing of the contacts results in power on and off of the circuit.

Input:

VCC : Connected to positive supply voltage (supply power according to relay voltage)

GND : Connected to supply ground.

IN1: Signal triggering terminal 1 of relay module

IN2: Signal triggering terminal 2 of relay module

IN3: Signal triggering terminal 3 of relay module

IN4: Signal triggering terminal 4 of relay module

IN5: Signal triggering terminal 5 of relay module

IN6: Signal triggering terminal 6 of relay module

IN7: Signal triggering terminal 7 of relay module

IN8: Signal triggering terminal 8 of relay module

Output:

Each module of the relay has one NC (normally close), one NO (normally open) and one COM (Common) terminal. So there are 8 NC, 8 NO and 8 COM of the channel relay in total. NC stands for the normal close port contact and the state without power. NO stands for the normal open port contact and the state with power. COM means the common port. You can choose NC port or NO port according to whether power or not.

Testing Setup:

When a low level is supplied to signal terminal of the 8-channel relay, the LED at the output terminal will light up. Otherwise, it will turn off. If a periodic high and low level is supplied to the signal terminal, you can see the LED will cycle between on and off.

Arduino Application Examples for 4-Channel Relay Board:

Step 1:

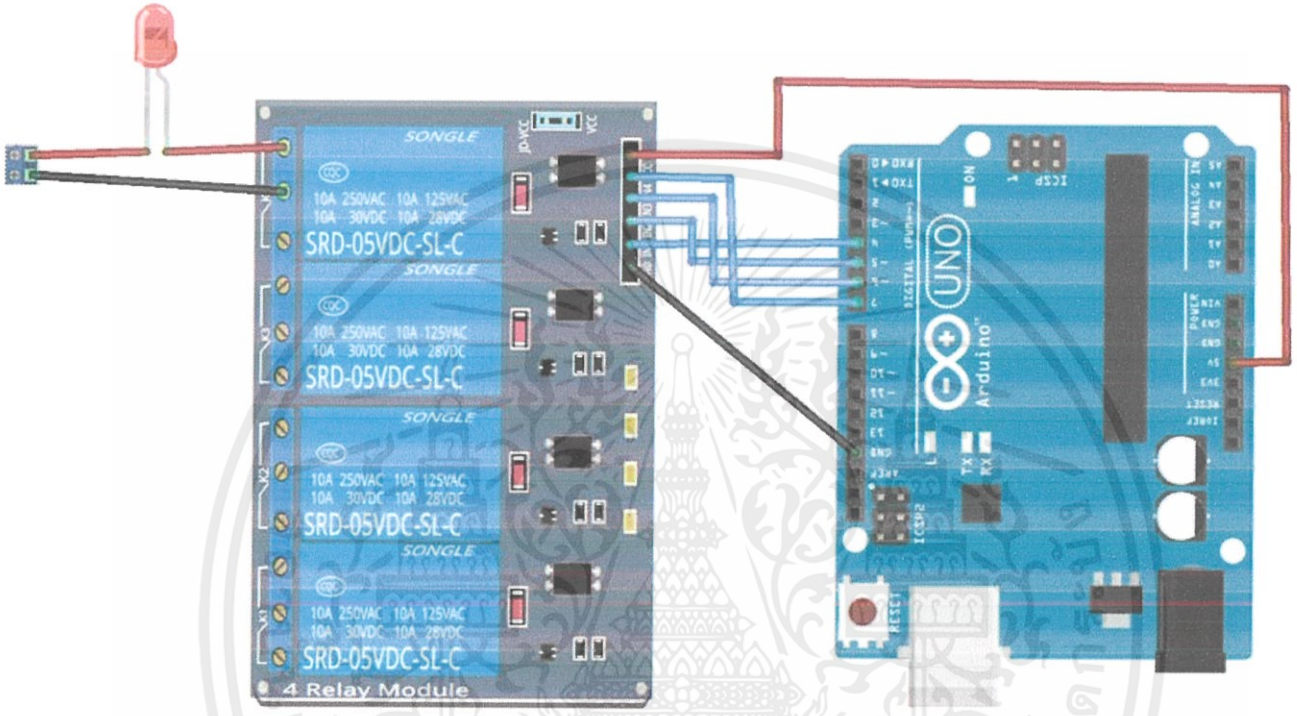
Connect the signal terminal IN1、IN2, IN3 & IN4 of 4-channel relay to digital pin 4, 5, 6, 7 of the Arduino Uno or ATMega2560 board, and connect an LED at the output terminal.

IN1> 4; IN2> 5; IN3>6; IN4>7

Step 2:

Upload the sketch "4 Channel Relay Demo " to the Arduino Uno or ATMega2560 board. Then you can see the LED cycle between on and off.

The actual figure is shown below:

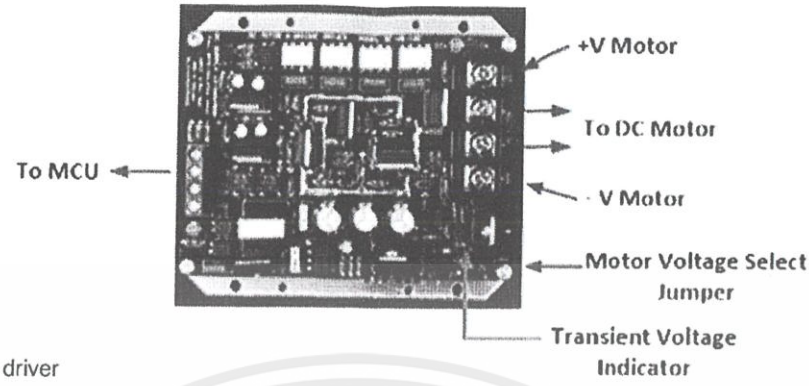


Arduino Sketch: 4 Channel Relay Demo

```
/*****************************************************
   Name: 4 channel_relay
   Description: control the 4 channel relay module to ON or OFF
   Website: www.handsontec.com
   Email: techsupport@handsontec.com
*****************************************************/

//the relays connect to

int RelayControl1 = 4;     // Digital Arduino Pin used to control the motor
int RelayControl2 = 5;
int RelayControl3 = 6;
int RelayControl4 = 7;


void setup()
{
  Serial.begin(9600);
  pinMode(RelayControl1, OUTPUT);
  pinMode(RelayControl2, OUTPUT);
  pinMode(RelayControl3, OUTPUT);
  pinMode(RelayControl4, OUTPUT);
```

```
}


void loop()
{

  digitalWrite(RelayControl1,HIGH);// NO1 and COM1 Connected (LED on)
  delay(1000);
  digitalWrite(RelayControl1,LOW);// NO1 and COM1 disconnected (LED off)
  delay(1000);
  digitalWrite(RelayControl2,HIGH);
  delay(1000);
  digitalWrite(RelayControl2,LOW);
  delay(1000);
  digitalWrite(RelayControl3,HIGH);
  delay(1000);
  digitalWrite(RelayControl3,LOW);
  delay(1000);
  digitalWrite(RelayControl4,HIGH);
  delay(1000);
  digitalWrite(RelayControl4,LOW);
  delay(1000);
}
```

DC Motor H-Bridge Driver 12-36V/80A Model SE-HB-100



Technical Specifications

1. Output : Single motor driver
   - Motor DC Supply 12-36 V 80A (Max.)
   - All N-Channel Power MOSFET H-Bridge Driver
     With ultra-fast reverse recovery inverse parallel protection diodes
2. Input :
   - Full Opto-isolated ground-loop input interface signals
   - Control Voltage input : 3-5V / 8 mA(min.)
3. Drive Mode : independently with :
   - Start-stop Control
   - Direction Control
   - Speed Control (PWM Drives)
4. PWM Duty cycle Range: 0-100%
5. PWM Frequency: 400 Hz - 25000 Hz
6. Board built-in Transient voltage protection up to 100 V circuit.

Input pins

EN/PWM	IN2	IN1	Motor Operation
0V/PWM	X	X	Free Run Stop
5V/PWM	GND	5V	Forward Drive
5V/PWM	5V	GND	Backward Drive
5V/PWM	5V	5V	Fast Stop or Brake (Not Recommend)
5V/PWM	GND	GND	Fast Stop or Brake (Recommend)

Remarks

1. The power cord from the battery and the motor cable should be no smaller than 2 sq.mm. or 18 -16AWG and tighten twist to reduce noise signal and XL as much as possible.

2. To adjust the motor speed, PWM signal from a microcontroller or PLC or other PWM source which voltage at 3.3V - 5V must be connected to the EN / PWM pin only !!. If you want to control the motor speed with linear DC 0-5V, the Motor Controller Interfaces board must be connected (please contract service center).

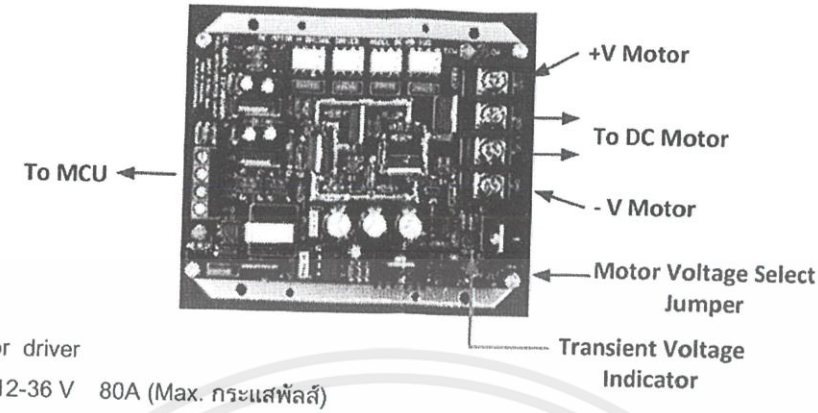
3. When an OVR bulb is blinking, show that the transient voltage 82V exceeded and the protection circuit is active. Users do not worry.

4. The black heat sinks on board is high heat during working condition. That It's normal condition, users do not worry.

This board can be connected to a microcontroller board which it has an output logic level 3-5V directly without modifying the circuit. If used with an Open Collector PLC, connect R 2K 1W to the terminals IN1 and IN2 and R 10K 1 / 4W to the EN terminal on the + 24V power supply of the PLC. The logic must be changed from logic 1 to 0 and from 0 to 1 in the control program, because the signal is Negative Logic.

If there are any problems, please contact us :
Mail: sombonie.san@gmail.com.
Line Id / Skipe Id : somboon2494
Mobile(Local) 081-2535810

บอร์ดขับมอเตอร์ดีซี แบบ H-Bridge 80A Model SE-HB-100



Technical Specifications

1. Output : Single motor driver
   - Motor DC Supply 12-36 V 80A (Max. กระแสพัลส์)
   - All N-Channel Power MOSFET H-Bridge Driver
     With ultra-fast reverse recovery inverse parallel protection diodes
2. Input :
   - Full Opto-isolated ground-loop input interface signals
   - Control Voltage input : 3-5V / 8 mA(min.)
3. Drive Mode : independently with :
   - Start-stop Control
   - Direction Control
   - Speed Control (PWM Drives)
4. PWM Duty cycle Range: 0-100%
5. PWM Frequency: 400 Hz - 25000 Hz
6. Board built-in Transient voltage protection up to 100 V circuit.

ตาราง ควบคุมการทำงาน

EN/PWM	IN2	IN1	การทำงานของมอเตอร์
0V/PWM	X	X	Free Run Stop (หยุดเมื่อหมดแรงเฉื่อย)
5V/PWM	GND	5V	หมุนเดินหน้า
5V/PWM	5V	GND	หมุนกลับทาง
5V/PWM	5V	5V	Fast Stop หรือ Brake (ไม่แนะนำให้ใช้)
5V/PWM	GND	GND	Fast Stop หรือ Brake (แนะนำให้ใช้)

หมายเหตุ

1. สายไฟที่ต่อจากแบตเตอรี่และสายต่อเข้ามอเตอร์ควรใช้ขนาดไม่ต่ำกว่า 2 ตร.มม. หรือ เบอร์ 18-16 และต้องตีเกลียวแน่น เพื่อลดค่า XL ให้มากที่สุด

2. หากต้องการปรับความเร็วมอเตอร์ จะต้องต่อสัญญาณ PWM จากไมโครคอนโทรลเลอร์ หรือ PLC หรือแหล่งกำเนิดสัญญาณ PWM อื่นๆ ที่สามารถปรับค่า Duty Cycle ได้ ที่มีขนาดแรงเคลื่อน 3.3V – 5V เข้าที่ขา EN/*PWM* ***(ห้ามนำแรงเคลื่อนไฟดีซี 0-5V มาป้อนเข้าขา EN/PWMโดยเด็ดขาด เพราะจะทำให้ Power MOSFET ทำงานแบบลิเนียร์ มีผลทำให้ MOSFET ร้อนใหม้และระเบิดได้)*** ถ้าหากต้องการจะควบคุมความเร็วมอเตอร์ด้วยไฟดีซี 0-5V จะต้องนำบอร์ด Motor Controller อินเตอเฟสมาต่อเสริม(ติดต่อฝ่ายบริการ)

3. ขณะใช้งาน หากมีหลอด OVR ติดแดงขึ้นเป็นบางครั้ง แสดงว่ามี Transient Voltage เกิน 82V เกิดขึ้น และวงจรป้องกันกำลังทำหน้าที่อยู่ ผู้ใช้ไม่ต้องกังวล

4. Heat sink สีดำทั้ง 2 ตัวที่อยู่บนบอร์ด ขณะใช้งานจะมีความร้อนสูง เป็นสภาพทำงานปกติ ผู้ใช้ไม่ต้องกังวล

บอร์ดนี้สามารถเชื่อมต่อการควบคุมกับบอร์ดไมโครคอนโทรลเลอร์ ที่มีระดับไฟลอจิกเอ้าท์พุท 3-5V ได้โดยตรงโดยไม่ต้องดัดแปลงวงจร หากใช้เชื่อมต่อกับ PLC แบบ Open Collector จะต้องต่อ R 2K 1W ที่ขั้ว IN1 และ IN2 และ R 10K 1/4W ที่ขั้วEN ไว้กับไฟ +24VของPLCด้วย และต้องเขียนโปรแกรมแบบกลับลอจิก จากลอจิก 1 เป็น 0 และจาก 0 เป็น 1 ด้วย เนื่องจากสัญญาณที่ได้เป็น Negative Logic

หากมีปัญหาเกี่ยวกับการใช้งาน โปรดติดต่อฝ่ายบริการ 081-2535810 สามารถแอดไลน์จากเบอร์นี้ได้