หุ่นยนต์เก็บของอัตโนมัติ

Automatic object collector robot

กนกพรรณ รัตนสิริเวช

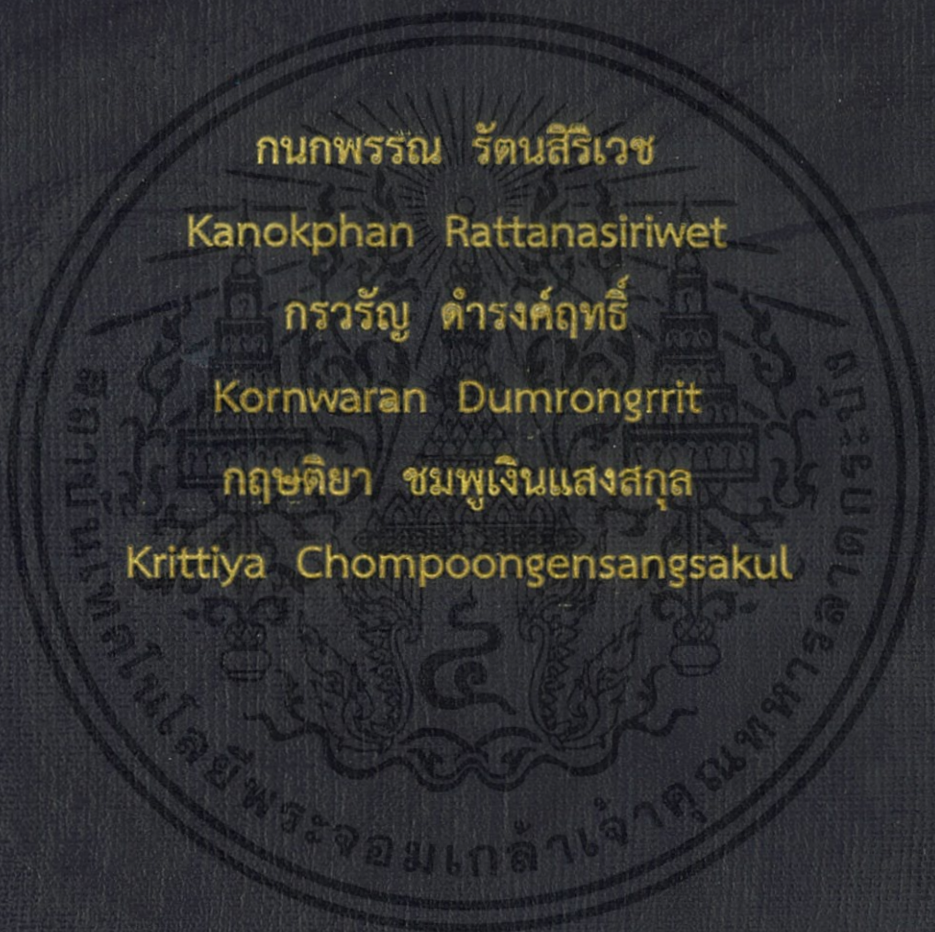
Kanokphan Rattanasiriwet

กรวรัญ ดำรงค์ฤทธิ์

Kornwaran Dumrongrrit

กฤษติยา ชมพูเงินแสงสกุล

Krittiya Chompoongensangsakul



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2558

หุ่นยนต์เก็บของอัตโนมัติ

Automatic object collector robot

กนกพรรณ รัตนสิริเวช
Kanokphan Rattanasiriwet
กรวรัญ ดำรงค์ฤทธิ์
Kornwaran Dumrongrit
กฤษติยา ชมพูเงินแสงสกุล
Krittiya Chompoongensangsakul

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2558

หุ่นยนต์เก็บของอัตโนมัติ

Automatic object collector robot

โดย

กนกพรรณ รัตนสิริเวช

กรวรัญ ดำรงค์ฤทธิ์

กฤษติยา ชมพูเงินแสงสกุล

อาจารย์ที่ปรึกษา

ผศ.ดร. ภัทรพงษ์ ผาสุขกิจ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2558

ปริญญานิพนธ์ปีการศึกษา 2558

สาขาวิชา วิศวกรรมอิเล็กทรอนิกส์

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง หุ่นยนต์เก็บของอัตโนมัติ

Automatic object collector robot

ผู้จัดทำ นางสาวกนกพรรณ รัตนสิริเวช **รหัสนักศึกษา** 55010008

นางสาวกรวรัญ ดำรงค์ฤทธิ์ **รหัสนักศึกษา** 55010021

นางสาวกฤษติยา ชมพูเงินแสงสกุล **รหัสนักศึกษา** 55010047

ปริญญานิพนธ์นี้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

ลงชื่อ.....

ผศ.ดร.ภัทรพงษ์ ผาสุขกิจ

อาจารย์ที่ปรึกษา

หัวข้อปริญญานิพนธ์	หุ่นยนต์เก็บของอัตโนมัติ
นักศึกษา	นางสาวกนกพรรณ รัตนสิริเวช รหัสนักศึกษา 55010008
	นางสาวกรวรัญ ดำรงค์ฤทธิ์ รหัสนักศึกษา 55010021
	นางสาวกฤษติยา ชมพูเงินแสงสกุล รหัสนักศึกษา 55010047
ปริญญา	วิศวกรรมศาสตรบัณฑิต
สาขาวิชา	วิศวกรรมอิเล็กทรอนิกส์
ปีการศึกษา	2558
อาจารย์ที่ปรึกษาปริญญานิพนธ์	ผศ.ดร.ภัทรพงษ์ ผาสุขกิจ

บทคัดย่อ

โครงงานนี้มีวัตถุประสงค์เพื่อศึกษาและสร้างหุ่นยนต์เก็บของ เนื่องจากในยุคปัจจุบันหุ่นยนต์ได้เข้ามามีบทบาทในชีวิตคนเรามากขึ้น หุ่นยนต์ได้ถูกนำไปพัฒนาใช้ในอุตสาหกรรม การเกษตรและการขนส่งสินค้าต่างๆ ทางคณะผู้จัดทำจึงเล็งเห็นถึงประโยชน์ของระบบหุ่นยนต์ เพื่อเพิ่มความสามารถของหุ่นยนต์ให้มีการตอบสนองอย่างรวดเร็ว โดยหุ่นยนต์นี้สามารถเคลื่อนที่ไปตามเส้นโดยอัตโนมัติ และเก็บวัตถุแต่ละสี และจะวางอยู่ในกล่องตามสีที่กำหนดไว้ ในโครงงานนี้ประกอบไปด้วยโครงสร้างของหุ่นยนต์และการควบคุมการเคลื่อนที่ของหุ่นยนต์ โดยใช้ไมโครคอนโทรลเลอร์ Arduino Mega 2560 R3 ในการควบคุมการเคลื่อนที่และแขนกลของหุ่นยนต์ ใช้เซนเซอร์ในการตรวจจับเส้นและเซนเซอร์มาตรวจวิเคราะห์จำแนกสี เมื่อพบวัตถุเป้าหมายระบบจะประมวลผลแล้วทำการควบคุมแขนกลของหุ่นยนต์ในการหยิบวัตถุของหุ่นยนต์ ระบบนี้หุ่นยนต์มีการตอบสนองที่รวดเร็วสำหรับใช้ในโรงงานอุตสาหรรม หรือใช้ในพื้นที่อันตรายที่คนไม่สามารถทำงานได้

Thesis Title	Automatic object collector robot	
Student	Miss. Kanokphan Rattanasiriwet	Student ID. 55010008
	Miss. Kornwaran Dumrongrit	Student ID. 55010021
	Miss. Krittiya Chompoongensangsakul	Student ID. 55010047
Degree	Bachelor of Engineering	
Program	Electronics Engineering	
Year	2015	
Thesis Advisor	Asst. Prof. Dr. Pattarapong Phasukkit	

ABSTRACT

This project is purpose to study and built robot collect object. In the present day, robot deal with people's daily life more. Robot have been developed in industrial Machinery and transport. The advantages of robot system is the main factor to make our project to increase the ability of the robot to respond quickly. The robot that can detect and follow the line drawn on the floor and collect the object, the objects consist of three boxes with different colors that need to be classified into three different places. In this project are consist of the structural design of the robot and how to control the robot. We use Arduino Mega 2560 R3 as control robot and robot arm that use Tracking Sensor and Color Recognition Sensor. When an object is found, the system will be process and control robot arm to pick up object. The robot responds very suitable for using in the factory or in areas that cannot work by human-being.

กิตติกรรมประกาศ

ในการจัดทำโครงงานหุ่นยนต์วิ่งเก็บของอัตโนมัติแบบเลือกเป้าหมาย สามารถเสร็จสมบูรณ์ได้เนื่องด้วยความกรุณาของบุคคลหลายท่านที่คอยช่วยเหลือและคอยให้คำปรึกษา รวมทั้งข้อเสนอแนะที่เป็นประโยชน์ต่อโครงงาน ทางคณะผู้จัดทำใคร่ขอแสดงความขอบพระคุณผู้ที่มีส่วนเกี่ยวข้องทุกท่าน

ผศ.ดร. ภัทรพงษ์ ผาสุขกิจ อาจารย์ที่ปรึกษาโครงงานผู้ที่เปิดโอกาสให้คณะผู้จัดทำได้เรียนรู้การทำงานในโครงงานนี้ และคอยให้คำปรึกษา อำนวยความสะดวกในเรื่องสถานที่และเครื่องมือเครื่องใช้ และคำแนะนำที่เป็นประโยชน์อย่างยิ่ง

เพื่อนๆ พี่ๆ ทุกท่านที่คอยให้ความช่วยเหลือที่ดีทุกด้าน ตลอดจนกำลังใจที่มอบให้แก่คณะผู้จัดทำตลอดมา

สุดท้ายผู้จัดทำขอกราบขอบพระคุณบิดาและมารดา ซึ่งเป็นผู้ให้โอกาสทางการศึกษาและคอยสนับสนุน รวมทั้งกำลังใจที่คอยมอบให้ตลอดมาอย่างหาที่เปรียบมิได้

กนกพรรณ รัตนสิริเวช
กรวรัญ ดำรงค์ฤทธิ์
กฤษติยา ชมพูเงินแสงสกุล

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป(ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญ

ในปัจจุบันหุ่นยนต์เคลื่อนที่แบบอัตโนมัติเข้ามามีบทบาทในชีวิตประจำวันของมนุษย์เป็นอย่างมากทั้งในด้านการขนส่งสินค้า ด้านการช่วยเหลือคนพิการ ด้านการสำรวจและกู้ภัย ทั้งงานในภาคอุตสาหกรรมด้านต่างๆ เช่น หุ่นยนต์ขนส่งและจัดเก็บสินค้าภายในโรงงาน หุ่นยนต์เคลื่อนย้ายคนพิการหรือผู้เจ็บป่วย หุ่นยนต์แม่บ้านทำความสะอาด หุ่นยนต์สำรวจและกู้ภัย เป็นต้น โครงงานนี้จึงเล็งเห็นถึงประโยชน์ของหุ่นยนต์เคลื่อนที่ ว่าสามารถนำมาประยุกต์ใช้ในการประมวลผลภาพและ Robot arm โดยโครงงานนี้สามารถแบ่งการทำงานออกเป็น 2 ส่วนหลัก คือ ส่วนแรกเป็นการตรวจจับเส้นควบคุมหุ่นยนต์ให้วิ่งตามเส้น ส่วนที่ 2 คือการให้หุ่นยนต์หยิบจับวัตถุเป้าหมาย

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

โครงงานนี้มีวัตถุประสงค์เพื่อสร้างหุ่นยนต์ที่สามารถเคลื่อนที่และหยิบสิ่งของได้อัตโนมัติโดยไม่ต้องมีผู้บังคับ และสามารถนำสิ่งของเหล่านั้นไปวางไว้ในจุดที่ต้องการได้ แทนการใช้รถยกของต่างๆ ที่จำเป็นต้องมีคนบังคับ

1.3 สมมุติฐานของการศึกษา

เพื่อให้หุ่นยนต์มีความเสถียรภาพ สามารถทำงานได้อย่างราบรื่น จึงใช้ Servo Driver และยังใช้เซ็นเซอร์ขาวดำจำนวน 4 ตัว เพื่อจับเส้น และเซ็นเซอร์สีจำนวน 1 ตัว เพื่อจับสีของกล่อง นอกจากนี้ยังมีการใช้บอร์ด Arduino MEGA เพื่อใช้ในการควบคุมหุ่นยนต์

1.4 ขอบเขตการศึกษา

1. ศึกษาการใช้โปรแกรม AutoCad ในการออกแบบหุ่นยนต์
2. ศึกษาการใช้โปรแกรม Solidwork ในการออกแบบหุ่นยนต์
3. ศึกษาอุปกรณ์ที่ต้องใช้ในการสร้างหุ่นยนต์
4. ศึกษาวงจร Full H Bridge Motor Controller
5. ศึกษาการใช้งาน Arduino เบื้องต้น
6. ศึกษา Robot Arm

1.5 ประโยชน์ที่คาดว่าจะได้รับ

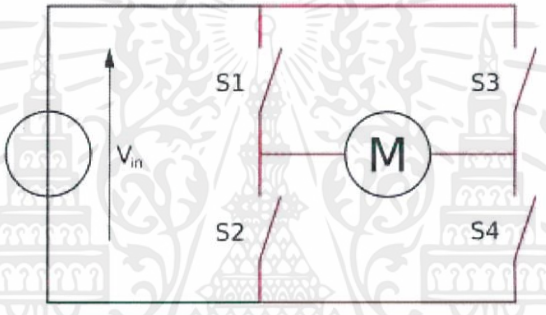
1. เข้าใจหลักการทำงานของวงจรควบคุมมอเตอร์ Full H Bridge Motor Controller
2. ได้เรียนรู้การออกแบบตัวโครงสร้างและชิ้นส่วนของหุ่นยนต์
3. ได้เรียนรู้ในส่วนโปรแกรมที่ใช้ในการควบคุมหุ่นยนต์
4. สร้างหุ่นยนต์ที่สามารถหยิบจับวัตถุและนำไปวางไว้ในจุดที่ต้องการโดยที่ไม่ต้องมีคนบังคับ

บทที่ 2
ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 มอเตอร์ คอลโทรลเลอร์

เราต้องการควบคุมทิศทาง และ ความเร็วรอบของมอเตอร์ ไม่ว่าจะเป็นการขับเคลื่อนสายพาน หรือ ล้อของหุ่นยนต์ เพราะเราคงไม่ต้องการให้หมุนแบบอิสระควบคุมไม่ได้ ดังนั้นก็เลยมีคนคิดวงจรที่ใช้ในการควบคุมมอเตอร์ขึ้นมา แบบที่นิยมใช้กันเรียกว่าวงจร "H-Bridge"

2.1.1 ควบคุมทิศทางการหมุน



รูปที่ 2.1 วงจรควบคุมทิศทางและความเร็วรอบของมอเตอร์

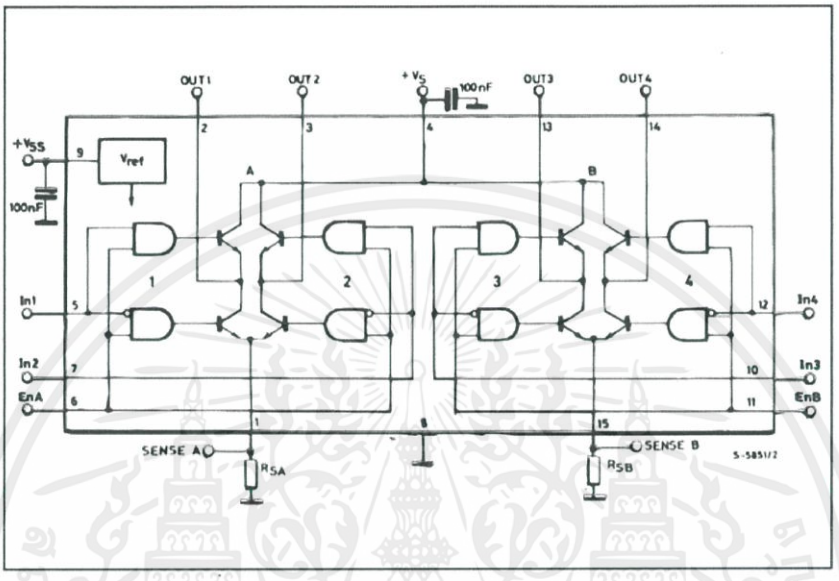
โดยปกติหากต้องการกลับทิศการหมุนของมอเตอร์กระแสตรง วิธีหนึ่งที่ทำได้คือ กลับทิศแหล่งจ่าย ทีนี้ลองดูที่รูปวงจร H-Bridge ในรูปที่ 2.1

- หากต้องการให้หมุนตามเข็ม (Clockwise :CW) ก็ให้ S1 และ S4 ปิดวงจร และให้ S2 และ S3 เปิดวงจร
- หากต้องการให้หมุนทวนเข็ม (Counter Clockwise :CCW) ก็ให้ S2และ S3 ปิดวงจร และให้ S1 และ S4 เปิดวงจร

จะเห็นว่าสวิตช์จะทำงานเป็นคู่ S1 คู่กับ S4 และ S2 คู่กับ S3 คู่แรกทำงาน คู่สองต้องเปิดวงจร และในทางตรงข้ามก็คือคู่สองทำงาน คู่แรกต้องเปิดวงจร ทีนี้จะทำอย่างไรให้การเปิดปิดเป็นแบบที่ง่ายกว่านี้ คำตอบก็คือ ใช้อุปกรณ์สารกึ่งตัวนำเช่น MOSFET หรือ IGBT หรือ อื่นๆ แล้วแต่ความเหมาะสม เช่น ขนาดกระแส แรงดันที่ต้องการควบคุม IC ที่มีวงจร H-Bridge เลือก 2 แบบหลักๆ คือ L293D และ L298P

2.2 วงจร L298-DUAL FULL-BRIDGE DRIVER

BLOCK DIAGRAM



รูปที่ 2.2 วงจร L298-DUAL FULL-BRIDGE DRIVER

2.2.1 หลักการทำงานของวงจร

วงจรในรูปที่ 2.2 นั้นเป็นวงของ L298-DUAL FULL-BRIDGE DRIVER ซึ่งเป็นวงจรที่สามารถควบคุมการการหมุนของมอเตอร์ได้ถึงสองตัว โดยการทำงานหลักๆนั้น คือ เมื่อจ่ายกระแสที่ขา Vs ซึ่งจะจ่ายกระแส 12 V เพื่อจ่ายให้กับมอเตอร์นั้นหมุนได้ โดยขา In1, In2 และ EnA เป็นขาที่ควบคุมทิศทางการหมุนของมอเตอร์ โดยถ้าเราจ่ายกระแส In1 และ EnA จะทำให้มอเตอร์หมุนในทิศทางที่ 1 และถ้าเราจ่ายกระแสที่ขา In2 และ EnA จะทำให้มอเตอร์หมุนไปอีกทิศทางหนึ่ง

2.3 บอร์ด Arduino Mega 2560 R3

Arduino Mega 2560

รูปที่ 2.3 บอร์ด Arduino Mega 2560 R3

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 ทำให้สามารถเขียนโค้ดโปรแกรมเข้าไปได้มากกว่า ในความเร็วของ MCU ที่เท่ากัน

2.3.1 ข้อจำกัดและส่วนประกอบของบอร์ด Arduino Mega 2560 R3

Arduino Mega 2560 เป็นบอร์ดไมโครคอนโทรลเลอร์ที่ทำงานบนพื้นฐานของ ATmega2560 ซึ่งประกอบด้วย

- 54 digital input/output pins (15 pin สามารถใช้เป็น PWM output ได้)
- 16 analog inputs
- 4 UARTs
- 16 MHz crystal oscillator (ใช้สำหรับกรองความถี่ให้กับบอร์ดไมโครคอนโทรลเลอร์)
- USB connection
- ช่องเสียบแหล่งจ่าย
- ICSP header: In-Circuit Serial Programming (ส่วนที่เป็น AVR ขนาดเล็กสำหรับการโปรแกรม Arduino ซึ่งประกอบด้วย MOSI, MISO, SCK, RESET, VCC, GND)

โดยบอร์ด Arduino Mega นี้ มีทุกสิ่งที่ไมโครคอนโทรลเลอร์จำเป็นต้องใช้อย่างการต่อไฟเลี้ยงสามารถทำได้ทั้งการเชื่อมต่อเข้ากับ USB cable หรือ จ่ายไฟด้วย AC-DC adapter หรือ การใช้แบตเตอรี่ซึ่ง Mega เป็นบอร์ดที่เข้ากันได้กับ shield ที่ออกแบบมาเพื่อ Arduino

Duemilanove หรือ Diecimila โดย Mega 2560 นี้มีความแตกต่างจากบอร์ดก่อนหน้าตรงที่ไม่ใช้ FTDI USB-to-serial driver chip แต่จะมี ATmega16U2 เข้ามาเป็นโปรแกรมแปลง USB-to-serial

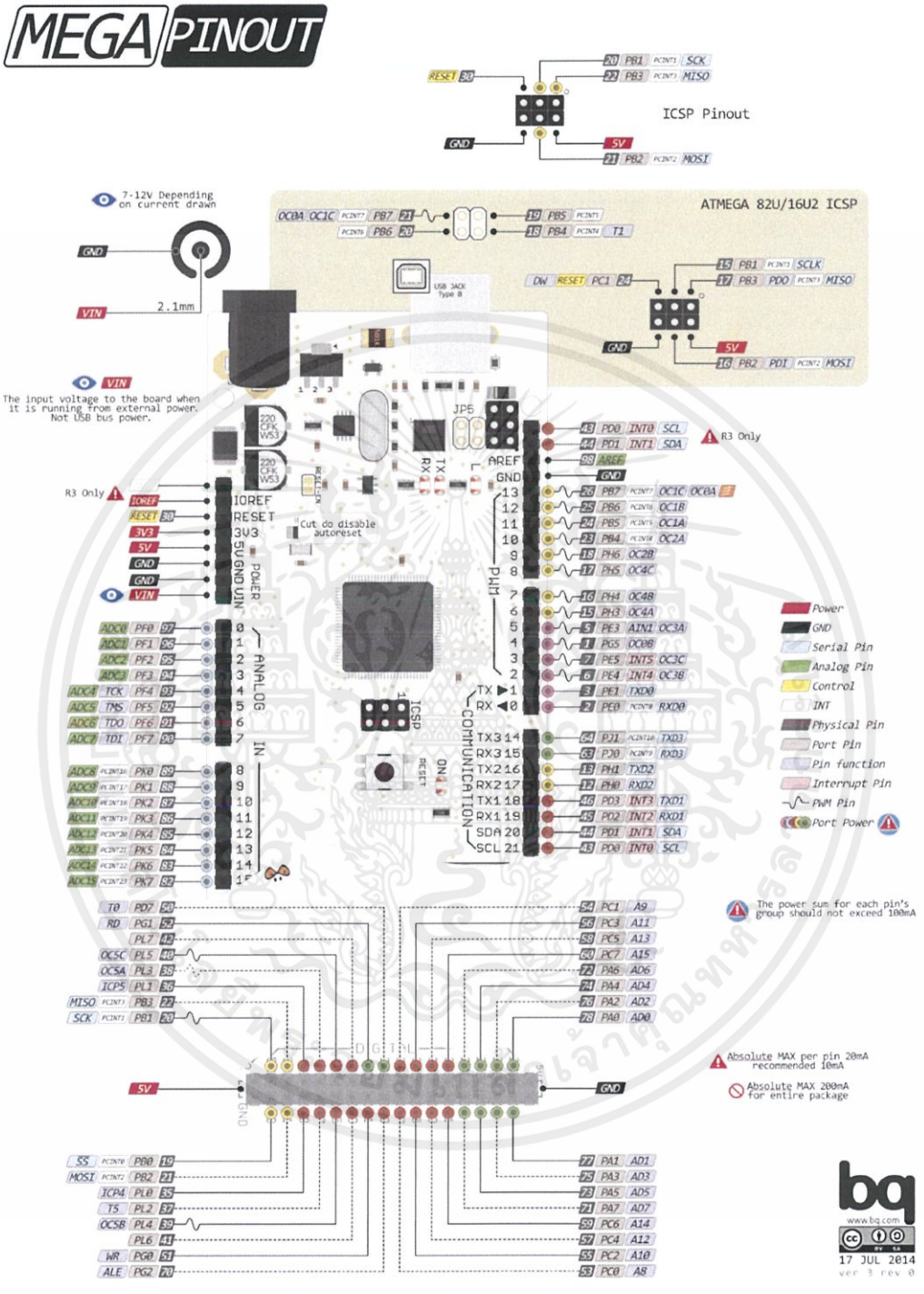
- Arduino Mega2560 Revision 2 มี ATmega8U2 ทำให้อัพเดท firmware ผ่าน USB protocal ที่เรียกว่า DFU (Device Firmware Update) ได้ง่ายขึ้น

- Arduino Mega Revision 3 มี featureใหม่ๆเพิ่มขึ้นมาดังนี้

- 1.0 pinout: เพิ่ม SDA และ SCL (อยู่ใกล้กับ AREF pin) และอีกสอง pins ใหม่ คือ IOREFเป็น pin ที่ใช้ในการเชื่อมต่อกับ shields เพื่อแปลงเป็นแรงดันที่ได้จากบอร์ด ส่วนอีก 1 pin ที่เหลือมีไว้สำหรับใช้ร่วมกับ AVR ในอนาคต

- วงจร Reset ที่ดีขึ้น

- ใช้ ATmega 16U2 แทน 8U2



รูปที่ 2.4 Pin ของบอร์ด Arduino Mega 2560 R3

2.3.2 Power

Arduino Mega สามารถเชื่อมรับพลังงานโดยการเชื่อมต่อ micro USB connector หรือจากpower supply จากภายนอกได้ โดยแหล่งพลังงานจะถูกเลือกโดยอัตโนมัติ แหล่งจ่ายจากภายนอกสามารถมาได้จาก AC-to-DC adapter หรือจากแบตเตอรี่ โดยต่อเข้ากับ 2.1 mm center-positive plug ไปยังช่องเสียบแหล่งจ่าย และการต่อเข้ากับแบตเตอรี่สามารถทำได้โดยการต่อเข้ากับ GND และ Vin pin header ของ power connecter

บอร์ดสามารถทำงานได้ในช่วงแรงดัน 6 ถึง 20 volts ถ้า แหล่งจ่ายมีค่าต่ำกว่า 7 V อาจส่งผลให้ 5 V pinมีแรงดันที่ต่ำกว่า 5V และ บอร์ดอาจจะไม่เสถียร แต่ถ้าหากแรงดันมีค่าสูงกว่า 12 V อาจส่งผลให้บอร์ดOverheat และอาจทำให้บอร์ดเสียหายได้ ดังนั้นช่วงแรงดันที่เหมาะสมกับบอร์ดคือ 7 V ถึง 12 V

- 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 เพื่อเลือกค่าแรงดันให้กับshield ที่มาเชื่อมต่อกับบอร์ด

2.3.3 Memory

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 และoutput โดยจะทำงานที่แรงดัน 5 V และให้กระแสสูงสุด 40 mA

ฟังก์ชันอื่นๆเพิ่มเติม

Serial: 0 (Rx) และ 1(Tx); Serial 1: 19(Rx) และ 18 (Tx); Serial 2: 17 (Rx) และ16(Tx); Serial 3:15 (Rx) และ 14 (Tx) ใช้สำหรับรับ (Rx) และส่ง (Tx) TTL serial dataโดย pin 0 และ 1 จะถูกเชื่อมต่อไปยัง corresponding pins ของ ATmega16U2 USB-to-TTL serial chip

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 และ Diecimila

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

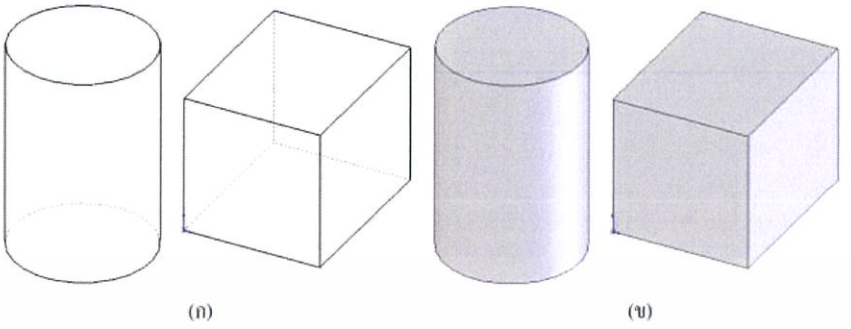
Reset ใช้ในการ reset ไมโครคอนโทรลเลอร์ โดยทั่วไปจะใช้โดยการเพิ่มปุ่ม reset ไว้บน shield เพื่อป้องกันปุ่มที่อยู่บนบอร์ด

2.3.4 Communication

Arduino Uno สามารถสื่อสารกับคอมพิวเตอร์, Arduino ตัวอื่นๆ หรือ microcontroller ได้ โดยที่ไมโครคอนโทรลเลอร์บนบอร์ด คือ ATmega32U4 จะให้การสื่อสารแบบอนุกรม UART TTL (5 V)ซึ่งมีอยู่ใน pins 0 (Rx) และ 1 (Tx) นอกจากนี้ 32U4 สามารถใช้การสื่อสารแบบอนุกรมผ่าน USB และจะปรากฏเป็น COM port เสมือนไปยัง Software แต่อย่างไรก็ตามต้องใช้ ไฟล์ .inf บนระบบปฏิบัติการ Windows แต่ OSX และ Linux สามารถ recognize ได้โดยอัตโนมัติ Arduino Mega สามารถรองรับการโปรแกรมด้วย Arduino Software โดยสามารถใช้ได้ทั้งในระบบปฏิบัตการ Windows , Mac OS X และ Linux

2.4 โปรแกรม Solidwork

โปรแกรม SolidWorks เป็นโปรแกรมช่วยเขียนแบบที่เน้นการเขียนแบบใน 3 มิติหรือ แบบ Parametric Solid โมเดล คำว่า Solid โมเดล หมายถึงแบบที่มีทรงตัน ซึ่งหมายถึงแบบใน 3 มิติที่มีเนื้อใน ซึ่งแตกต่างจากแบบ 3 มิติชนิด Wireframe ซึ่งสามารถใช้โปรแกรม CAD 2 มิติ บวกกับจิตนาการของคนเขียนแบบก็สามารถเขียนออกมาได้ Wireframe 3-D โมเดล จึงเป็นแบบ 3 มิติที่ประกอบด้วยลายเส้นหรือ line มาต่อๆ กัน ส่วนคำว่า Parametric Solid โมเดล หมายถึงแบบ 3 มิติที่ถูกสร้างขึ้นด้วยความสัมพันธ์ทางคณิตศาสตร์ ซึ่งเป็นการคำนวณภายในตัวโปรแกรม การเขียนแบบ 3 มิติลักษณะนี้จะมีความสะดวกกับ ผู้เขียนแบบมากกว่ารูป 2.5 แสดงแบบ 3 มิติชนิด Wireframe โมเดล และ Solid โมเดล



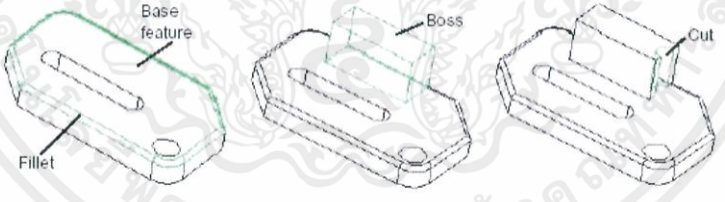
รูปที่ 2.5 แบบจำลอง 3 มิติ (ก) แบบ Wireframe และ (ข) แบบ Solid โมเดล

2.4.1 ขั้นตอนการสร้างชิ้นงาน 3 มิติและแบบสั่งงานในโปรแกรม SolidWorks

กระบวนการสร้างชิ้นงานใน SolidWorks ประกอบด้วย 3 ขั้นตอนหลัก ได้แก่

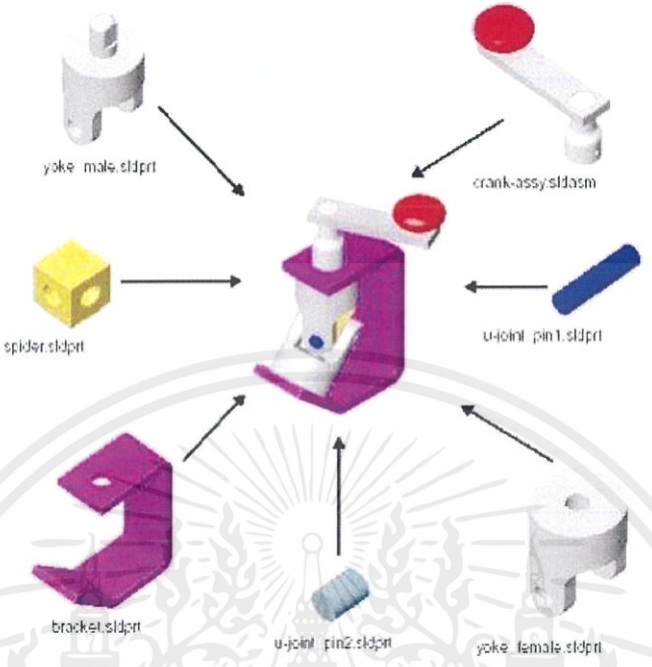
1) การสร้าง Sketch หรือลายเส้นใน 2 มิติบนระนาบ (plane, flat surface) ใดๆ ลายเส้นที่สร้างขึ้นส่วนใหญ่จะเป็นรูปปิด

2) ใช้คำสั่งใดๆ ใน Features ทำให้ Sketch กลายเป็นชิ้นงาน 3 มิติในการสร้างชิ้นงาน 3มิติที่ไม่ซับซ้อน Feature จะถูกสร้างซ้อนทับต่อๆ กันไป โดย feature แรกสุดโดยทั่วไปจะใช้ feature ที่มีขนาดใหญ่ เป็นหลักเรียกว่า Base ส่วน feature ที่สร้างต่อๆ มาจะเรียกว่า Boss ให้ดูรูปที่ 2.6



รูปที่ 2.6 ชิ้นงานหรือชิ้นส่วนแบบ Part ประกอบด้วยหลายๆ Features

3) นำ Parts หลายๆ ชิ้นมาประกอบกันเป็น Assembly ขอเรียกเป็นภาษาไทยว่า “ชิ้นงานประกอบ” ในกรณีที่ชิ้นงานประกอบไม่ซับซ้อน ก็อาจเรียกว่า Subassembly เราสามารถนำ partและ subassembly หลายๆ ชิ้น มาประกอบเป็นชิ้นงานประกอบขั้นสุดท้ายเรียกว่า Full Assembly (ดูรูปที่2.7)



รูปที่ 2.7 ชิ้นงานประกอบแบบ Assembly ประกอบด้วย Parts และ Sub-assembly

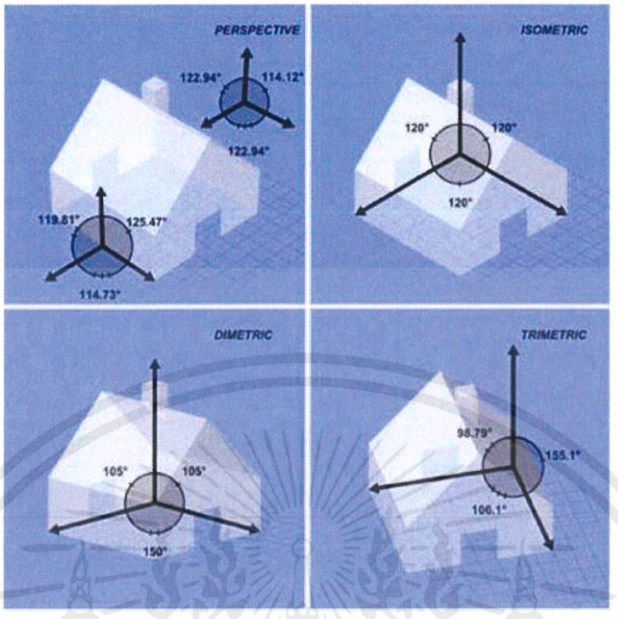
2.4.2 หลักการเขียนแบบพื้นฐานที่ควรรู้

แบบทางวิศวกรรม (Engineering Drawing) เป็นแบบจำลองโมเดล ที่สร้างขึ้นบนระนาบ e.g. กระดาษเขียนแบบ, จอภาพ เพื่อเป็นตัวแทนของชิ้นส่วนทางกล Mechanical Part แบ่งออกเป็น 2 ประเภท ใหญ่ๆ คือ แบบ 3 มิติและ แบบ 2 มิติ ในบางครั้งจะใช้คำว่า “มุมมอง ” หรือ View เพื่อขยายความว่า แบบในแต่ละประเภท เกิดจากการมองวัตถุในลักษณะใดเพื่อให้ได้แบบที่มีความยาวใน 2 หรือ 3 มิติ สำหรับแบบ 2 มิติ จะเกิดจากการมองตั้งฉากกับด้านใดด้านหนึ่งของวัตถุเพื่อให้เห็นขนาดที่แท้จริง (true length) แบบ 2 มิติ เรานิยมเรียกว่า ภาพฉาย (Projection)

2.4.2.1 มุมมอง 3 มิติ (3-D View)

ที่นิยมใช้ในการเขียนแบบเครื่องจักรกล คือ แบบ Axonometric หมายถึงการมองวัตถุให้เยื้องออกจากแกนหลักทั้ง 3 แกน (ไม่มองขนานกับแกนใดๆ เลย) ผลที่ได้ก็คือ ภาพวัตถุที่มีมิติใน ทั้ง 3 แกน ซึ่งแบ่งออกได้เป็น 3 ประเภท คือ

1. Isometric View คือ ภาพที่มีสามแกนหลักทำมุมเท่ากันทั้งหมด ทุกมุมมีขนาด 120 องศา
2. Dimetric View คือ มีมุมระหว่างแกนหลักในภาพ เท่ากัน 2 มุม
3. Trimetric View คือ ทั้งสามมุมระหว่างแกนหลักไม่เท่ากันเลย

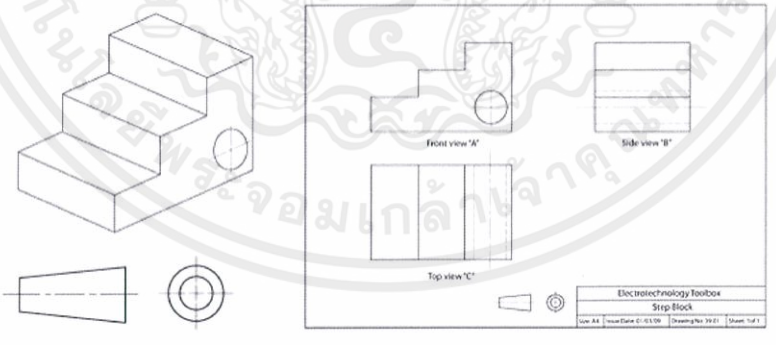


รูปที่ 2.8 มุมมอง 3 มิติ ที่ใช้ในการเขียนแบบ

2.4.2.2 **ภาพฉาย 2 มิติ (2-D Projection)**

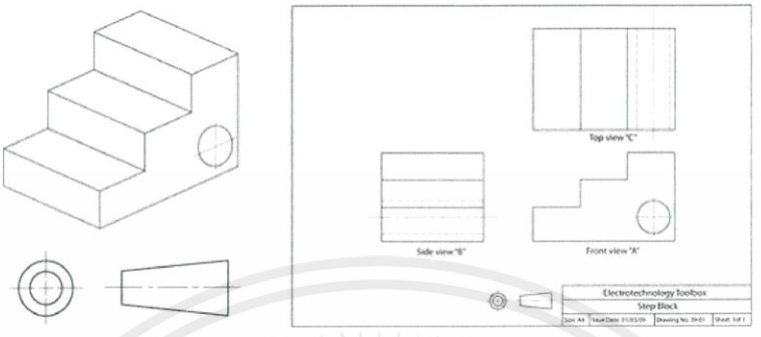
เนื่องจากเป็นการมองตั้งฉากกับด้านใดด้านหนึ่งของวัตถุจึงนิยม เรียกว่า ภาพฉายแบบ orthographic (Orthographic Projection) แบ่งออกเป็น 2 ประเภท คือ

1. แบบมุมที่หนึ่ง (First Angle Projection)



รูปที่ 2.9 การเขียนภาพฉาย orthographic แบบมุมที่หนึ่ง และสัญลักษณ์

2. แบบมุมที่สาม (Third Angle Projection)



รูปที่ 2.10 การเขียนภาพฉาย orthographic แบบมุมที่สาม และสัญลักษณ์

2.4.3 หน่วยในการวัดความยาวของการเขียนแบบทางกล

แบ่งออกเป็น 2 มาตรฐาน คือ

1. หน่วยเมตทริค (Metric Unit) โดยความยาวจะนิยมใช้เป็น มิลลิเมตรตัวย่อ คือ มม. (mm.) ข้อสังเกต ปกติความยาวในทางวิศวกรรมจะใช้เป็น เมตร (ม. หรือ m) การที่ใช้เป็นมิลลิเมตร เพราะในงานสร้างชิ้นงานทางกล ซึ่งมีขนาดค่อนข้างเล็กเราต้องการความความละเอียดในการวัด เพื่อให้ได้ขนาดที่ถูกต้องแม่นยำ หน่วย metric นี้อยู่ในกลุ่มเดียวกับ หน่วย S.I. ซึ่งเป็นที่นิยมใช้แพร่หลายมากที่สุด

2. หน่วยแบบอังกฤษ (English Unit) หรือ หน่วยแบบ Old English ความยาวจะเป็น นิ้ว (inch) จะใช้สัญลักษณ์ double quote (") ตามหลังตัวเลขความยาวเป็นนิ้ว เช่น 1" คือยาว 1 นิ้ว และ 3½ คือ ยาวสามนิ้วครึ่ง ประเทศที่ยังใช้ความยาวเป็นนิ้ว ที่สำคัญ คือ สหรัฐอเมริกาความยาวในหน่วยนิ้วยังสามารถแบ่งย่อยให้เล็กลงได้อีก หน่วยย่อยของนิ้วในภาษาไทยคือ "หุน" กำหนดให้ 1นิ้ว = 8 หุนฉะนั้น ½ นิ้ว จะเท่ากับ 4 หุน จะพบการใช้งานหน่วยนิ้วและหุน ในการบอกความยาวของน๊อตและสกรูและขนาดของดอกสว่าน เป็นต้น

2.5 เซนเซอร์แสง (Photoelectric sensor or Photo sensor)

เซนเซอร์แสง คือ เซนเซอร์ชนิดหนึ่งที่ใช้ลำแสงในการตรวจจับวัตถุ สามารถตรวจจับวัตถุได้ทุกชนิด มีระยะตรวจจับวัตถุไกล เวลาตอบสนองรวดเร็ว ใช้กับงานที่ต้องการความเร็วในกาตรวจจับสูง และตรวจจับวัตถุได้โดยไม่ต้องสัมผัส ตอบสนองการทำงานตามการเปลี่ยนแปลงความเข้มของแสงที่ได้รับ

2.5.1 คุณลักษณะโดยทั่วไป

- สามารถตรวจจับวัตถุแบบไม่ต้องสัมผัส
- สามารถตรวจจับวัตถุมากกว่า 10 เมตร
- สามารถตรวจจับวัตถุได้ทุกชนิด
- สามารถตรวจจับ สี, ขนาด, ความลึก, ตำแหน่ง, พื้นที่, และ อื่นๆ
- แสดงการตอบสนองโดยการกระพริบของ LED
- ความละเอียดสูง

2.5.2 ส่วนประกอบ

- Emitter (ตัวส่งสัญญาณ) : ประกอบด้วย ตัวกำเนิดแสง,หลอด LED และตัวสร้างสัญญาณมอดูเลสที่อัตราเร็วสูง ส่งเป็นแสงไปยังตัวรับสัญญาณ
- Receiver (ตัวรับสัญญาณ) : ประกอบด้วย ตัวรับแสงเพื่อแปลงสัญญาณ และส่วนของสวิตช์ทำหน้าที่เป็น Output
- Range (ช่วงสัญญาณ) : ตัวกำหนดระยะการทำงานของเซ็นเซอร์ หรือระยะการส่งสัญญาณ
- Opposed mode คือ ระยะจากตัวส่งถึงตัวรับสัญญาณ
- Retroreflective mode คือ ระยะจากเซ็นเซอร์ถึงแผ่นสะท้อน
- Proximity mode คือ ระยะจากเซ็นเซอร์ถึงวัตถุที่ต้องการตรวจจับ

Contrast

Contrast คือ อัตราส่วนของปริมาณแสงที่ตกกระทบบนตัวรับสัญญาณในขณะที่มีแสงมากเทียบกับปริมาณแสงตกกระทบขณะไม่มีแสง ค่า Contrast ยิ่งมีค่ามาก ยิ่งเพิ่มความน่าเชื่อถือในระบบการตรวจจับ

รูปที่ 2.11 ค่า contrast ต่างๆ

Beam Pattern

Beam pattern คือ กราฟ 2 มิติ ที่มีลักษณะเป็นรูปของแสงที่ตัวส่งสัญญาณปล่อยออกมา ใช้เพื่อพิจารณาการเลือกเซ็นเซอร์ในกรณีที่วัตถุที่ต้องการตรวจจับอยู่ใกล้กัน

Excess gain

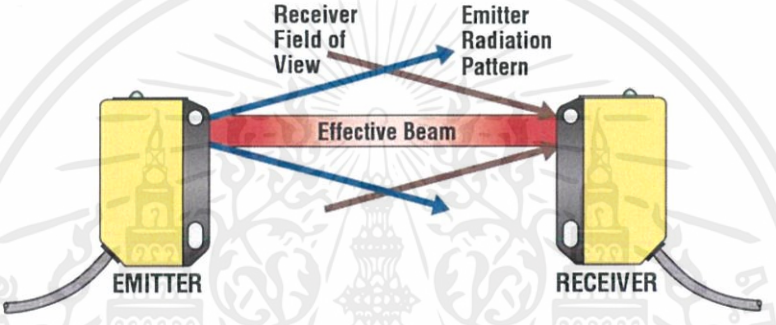
Excess gain คือ ตัวคูณเพื่อให้แสงที่ตกกระทบบนตัวรับสัญญาณมีค่าเพิ่มขึ้น มีค่ามากกว่าปริมาณแสงที่ต้องการใช้งาน

2.5.3 ลักษณะการทำงาน

Effective beam : แสงที่ใช้ในการตรวจจับ

Radiation pattern : พื้นที่ทั้งหมดของการส่งพลังงานออกมาเพื่อตรวจจับ

Field of view : พื้นที่ของการตอบสนองการทำงาน



รูปที่ 2.12 การทำงานของ Photo sensor

2.5.4 Types of Sensors (ชนิดของเซ็นเซอร์)

1. Self-contained sensors: ในเซ็นเซอร์หนึ่งตัว ประกอบด้วยสองส่วนคือ ส่วนของแสงตรวจจับ และ ส่วนของวงจรอิเล็กทรอนิกส์ เซ็นเซอร์ชนิดนี้จะดำเนินการด้วยตัวเอง ซึ่งมีทั้ง การ modulation, การ demodulation, การ amplification และ Output ที่เป็นสวิตช์

รูปที่ 2.13 Self-contained sensors

2. Remote systems: ระบบตรวจจับระยะไกล แบ่งส่วนแปลงสัญญาณและส่วนแสงตรวจจับวัตถุออกจากกัน ส่วนแสงตรวจจับประกอบด้วยตัวกำเนิดแสงเพียงอย่างเดียว ซึ่งมีขนาดเล็กมาก ส่วนตัวแปลงสัญญาณประกอบด้วย Input ที่เป็นไฟฟ้ากำลัง, amplification และ Output ที่เป็นสวิตช์ ระบบนี้ช่วยให้วงจรอิเล็กทรอนิกส์อยู่ห่างจากจุดที่ต้องการตรวจจับวัตถุ

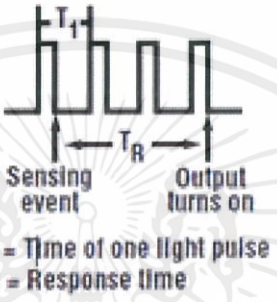
รูปที่ 2.14 Remote systems

3. Fiber optic systems: ระบบตรวจจับแบบใยแก้วนำแสง สามารถใช้ได้กับระบบตรวจจับระยะไกล หรือ ระบบตรวจจับแบบ Self-contained ใยแก้วนำแสงเป็นอุปกรณ์ที่ไม่มีวงจรอิเล็กทรอนิกส์ และส่วนที่เคลื่อนที่ ช่วยให้ท่อส่งมีความปลอดภัย และเป็นมิตรกับสิ่งแวดล้อม

รูปที่ 2.15 Fiber optic systems

2.5.5 เวลาในการตอบสนอง (Response time)

เวลาในการตอบสนอง (Response time) เป็นเวลาที่มากที่สุดในการตอบสนองของเซนเซอร์เมื่อรับสัญญาณอินพุตเข้ามาในขณะที่มีการตรวจจับวัตถุแล้วให้สัญญาณเอาท์พุตออกไปยังโหลดต่อไปเวลาในการตอบสนอง (Response time) เป็นเวลาระหว่างขอบของระยะที่ตรวจจับ (sensing event) แล้วมีการเปลี่ยนสถานะเอาท์พุต



รูปที่ 2.16 เวลาในการตอบสนอง

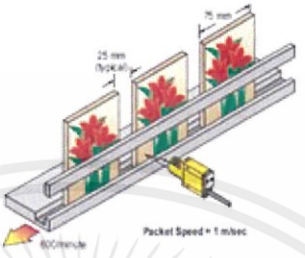
เวลาตอบสนองช่วยในการตัดสินใจกำหนดระยะเวลาที่วัตถุจะเคลื่อนที่ ซึ่งต้องอยู่ในช่วงที่เซนเซอร์สามารถตรวจจับได้ โดยเฉพาะในกรณีที่ต้องนำไปประยุกต์ตรวจจับ

- เหตุการณ์ที่มีความเร็วสูง
- วัตถุขนาดเล็กเคลื่อนที่ด้วยความเร็วสูง
- ช่องแคบๆระหว่างวัตถุ
- ลดระยะเวลาการตรวจจับ

2.5.6 การคำนวณเวลาตอบสนอง

สามารถกำหนดการตอบสนองความถี่ของเซนเซอร์ที่ต้องการได้เมื่อคุณรู้ขนาดความเร็วระยะห่างของวัตถุที่ตรวจจับ

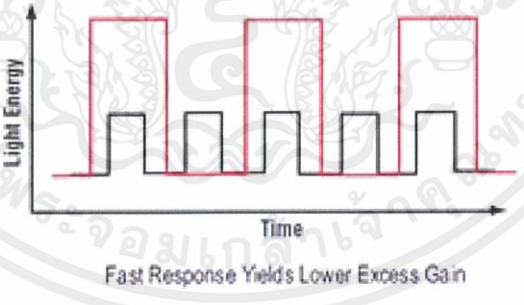
$$\text{เวลาตอบสนอง} = \frac{\text{ช่องว่างระหว่างวัตถุ}}{\text{ความเร็วของวัตถุ}}$$



รูปที่ 2.17 การคำนวณเวลาตอบสนอง

2.5.7 การมอดูเลชัน

ความเร็วของการตอบสนองของเซนเซอร์แสงถูกกำหนดโดยความถี่มอดดูเลชัน โดยเลือกได้ทางเดียวเท่านั้นคือระหว่างเวลาตอบสนองและระยะตรวจจับ (excess gain) ดังนั้นเซนเซอร์ตรวจจับที่มีความเร็วสูงจะมีระยะตรวจจับที่สั้นถ้า LED มีพัลส์น้อย สามารถเพิ่มพัลส์ด้วยการเพิ่มกระแสให้สูงขึ้นด้วยเหตุนี้จึงเป็นการเพิ่มพลังงานให้สูงขึ้น



รูปที่ 2.18 ความถี่มอดดูเลชัน

2.5.8 ความสามารถซ้ำค่าเดิม

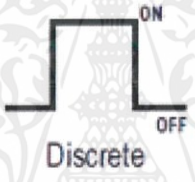
การเจาะจงซ้ำค่าเดิมถูกประยุกต์ใช้ในกรณีที่ผู้ใช้ต้องการรู้ตำแหน่งที่แม่นยำในการเคลื่อนที่ของวัตถุที่ต้องการตรวจจับ โดยที่สัญญาณเอาท์พุตของเซนเซอร์ถูกใช้เพื่อการเปิดปิดหลังจากที่มีการมอดูเลตสัญญาณพัลส์เข้ามา เวลาในการตอบสนองก่อนการเปิดให้เซนเซอร์มอดูเลสเท่ากับเวลาที่เซนเซอร์ใช้ในการนับจำนวนพัลส์ และเอาท์พุตของเซนเซอร์จะเปลี่ยนสภาพทันทีที่เซนเซอร์นับพัลส์

ที่ความถี่ที่ถูกต้อง ตั้งแต่ที่มีการตรวจจับที่ปรากฏในเวลาใดๆ ระหว่างรอบการมอดูเลส เวลาจริง ระหว่างการตรวจจับและการเปลี่ยนสถานะเอาท์พุตเซนเซอร์ที่สามารถแตกต่างกันได้ในการมอดูเลสแต่ละครั้ง การเปลี่ยนแปลงนี้เป็นความสามารถในการทำซ้ำของเซนเซอร์

2.5.9 เอาท์พุต

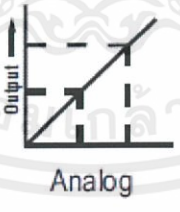
วงจรเอาท์พุตเป็นส่วนของเซนเซอร์ที่ต่อกับโหลดภายนอกที่ถูกส่งออกมาจากเซนเซอร์ การรู้ค่าแรงดันและค่ากระแสที่โหลดต้องการถือเป็นสิ่งสำคัญในการเลือกเซนเซอร์ เอาท์พุตเซนเซอร์ที่เป็นแบบต่อเนื่องจะต่ออยู่กับวงจรหรืออุปกรณ์ต่อไฟกระแสตรงที่แรงดันและกระแสระดับต่ำ เอาท์พุตเซนเซอร์ที่เป็นแบบไม่ต่อเนื่องจะต่อเข้ากับโหลดกระแสตรงหรือกระแสสลับก็ได้

เอาท์พุตของเซนเซอร์ ที่เป็นสัญญาณที่เป็นแบบไม่ต่อเนื่องจะมีเอาพุตเพียง 2 สถานะเท่านั้น คือ เปิดและปิด



รูปที่ 2.19 เอาท์พุตเซนเซอร์ที่เป็นแบบสัญญาณไม่ต่อเนื่อง

เอาท์พุตเซนเซอร์ที่เป็นแบบสัญญาณต่อเนื่อง เป็นตัวแปรหนึ่งที่มีการเปลี่ยนแปลงขนาดแรงดันหรือกระแสและเป็นสัดส่วนกัน โดยจะใช้สำหรับการวัดหรือการตอบสนองอย่างช้าๆ

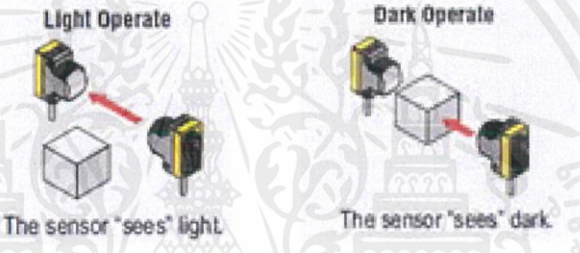


รูปที่ 2.20 เอาท์พุตเซนเซอร์ที่เป็นแบบสัญญาณต่อเนื่อง

2.5.10 การทำงานในสภาวะที่มีแสงมากและในสภาวะที่ไม่มีแสง (Light Operate or Dark Operate)

เซนเซอร์จะมีการกระตุ้นเมื่อมีการประยุกต์ที่ต้องการให้เซนเซอร์ทำงาน ด้วยสัญญาณแบบไม่ต่อเนื่องของเซนเซอร์ตรวจจับด้วยแสง อินพุตและเอาท์พุตเพียง 1 สถานะจาก 2 สถานะ คือ สภาวะที่มีแสง และสภาวะที่ไม่มีแสง

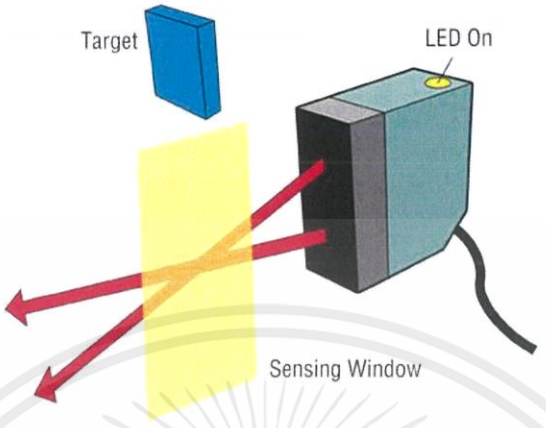
- สภาวะการทำงานที่มีแสง Light Operate (LO) สภาวะนี้เอาท์พุตของเซนเซอร์แสงจะให้พลังงาน เมื่อเซนเซอร์ตรวจจับสัญญาณที่มีการมอดูเลสของตัวมันเองได้
- สภาวะการทำงานที่มีแสง Dark operate (DO) เป็นองค์ประกอบเสริมของ LO ที่เซนเซอร์ให้พลังงาน เมื่อเซนเซอร์ไม่สามารถตรวจจับสัญญาณแสงที่มอดูเลตได้



รูปที่ 2.21 การทำงานในสภาวะที่มีแสงมากและในสภาวะที่ไม่มีแสง

2.5.11 Application

การประยุกต์ใช้งาน สามารถประยุกต์ใช้งานได้หลากหลาย ทั้งติดในไลน์การผลิตเพื่อตรวจสอบการบรรจุสินค้า การตรวจเช็คสินค้า หรือแม้กระทั่งใช้เป็นเซ็นเซอร์ตรวจการผ่านของรถ และปัจจุบันมีเซนเซอร์แสง หลายรุ่นหลายแบรนด์ ให้เลือกใช้ รวมทั้งมีทั้งแบบกันน้ำ กันฝุ่น ซึ่งแน่นอนว่าราคาย่อมแตกต่างกัน ดังนั้นก่อนตัดสินใจซื้อควรคำนึงถึงสภาพแวดล้อมและการนำไปใช้งานกันให้ดีก่อน



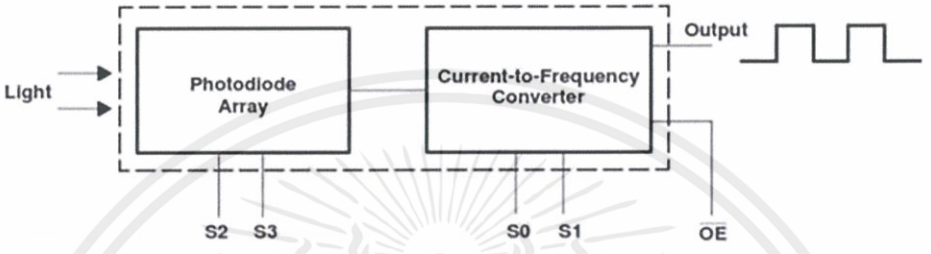
รูปที่ 2.22 การประยุกต์ใช้งาน

2.6 RGB Color Sensor (Color Recognition Sensor Detector)

รูปที่ 2.23 RGB Color Sensor

เป็นตัวอ่านค่าสี RGB หรือนำค่า RGB ไปแสดงบนหน้าจอคอมพิวเตอร์ โมดูลนี้จะช่วยเพิ่มความสามารถนี้ให้กับ Arduino โมดูล TCS230 Color Recognition Sensor ใช้ไฟเลี้ยง 3.3 - 5 โวลต์ ใช้สายสัญญาณ 3 เส้น มีสายสำหรับควบคุมไฟ LED อีก 1 เส้น สามารถสั่งเปิดไฟตอนกำลังอ่านค่าสี และสั่งให้ปิดเมื่ออ่านค่าสีเสร็จแล้วได้ ใช้เซ็นเซอร์ TAOS TCS230 เป็นตัวแยกความถี่ของแสง โดยใช้ photodiodes การแปลงแสงเป็นความถี่ อ่านได้จากค่า photodiodes ที่อยู่ในชิฟของ โมดูล TCS230 มีจำนวนทั้งหมด 8x8 ชิ้น สามารถอ่านสี แดง เขียว ฟ้า แสงสว่างจากแหล่งกำเนิดแสงโดยตรง ซึ่งค่าที่อ่านมาได้จะเป็นค่าความถี่ของแสง แดง เขียว ฟ้า ยังไม่ใช่ค่า RGB ที่

ใช้กันในคอม การควบคุมเซนเซอร์โมดูล TCS230 นี้ ทำได้โดยควบคุมจากขา s2 และ s3 และขา OUT จะให้ออกมาเป็นสัญญาณเวฟสี่เหลี่ยม (50% duty cycle) เป็นความถี่ที่อ่านได้จากเซนเซอร์แสง phototional โดยตรง สามารถขยายความเข้มของค่าที่อ่านได้ โดยควบคุมอัตราขยายที่ขา s0 และ s1



รูปที่ 2.24 การควบคุมเซนเซอร์

2.6.1 คุณสมบัติทั่วไป

High-Resolution Conversion of Light Intensity to Frequency

Programmable Color and Full-Scale Output Frequency

Communicates Directly With a Microcontroller

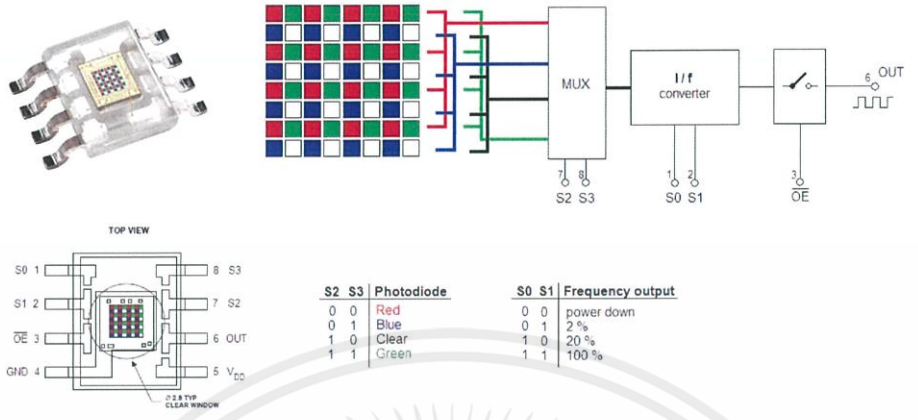
Single-Supply Operation (2.7 V to 5.5 V)

Power Down Feature

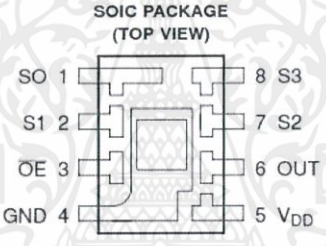
Nonlinearity Error Typically 0.2% at 50 kHz

Stable 200 ppm/°C Temperature Coefficient

Low-Profile Surface-Mount Package



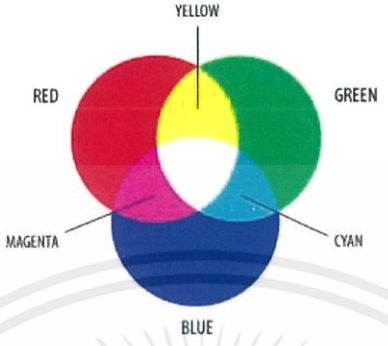
รูปที่ 2.25 คุณสมบัติของ TCS230



TERMINAL NAME	TERMINAL NO.	I/O	DESCRIPTION
GND	4		Power supply ground. All voltages are referenced to GND.
$\overline{OE}$	3	I	Enable for f_o (active low).
OUT	6	O	Output frequency (f_o).
S0, S1	1, 2	I	Output frequency scaling selection inputs.
S2, S3	7, 8	I	Photodiode type selection inputs.
V_{DD}	5		Supply voltage

รูปที่ 2.26 การจัดขาของ TCS230 และฟังก์ชันการทำงาน

2.6.2 หลักการทำงานของ TCS230



รูปที่ 2.27 แม่สี RGB

TCS230 มีเซนเซอร์ photodiodes ขนาด 8x8 ตัว โดย มีเซนเซอร์สีน้ำเงิน 16 ตัว สีเขียว 16 ตัว สีแดง 16 ตัวและ 16 ตัวสำหรับแสงสีขาว โดยจะให้สัญญาณเอาต์พุตออกมาทางขา s2 และ s3 เช่น ถ้า s2 และ s3 ให้สัญญาณ 0 หรือ L ออกมา แปลว่า อ่านได้ค่าแสงสีแดง แล้ว ถ้า s2 และ s3 ให้ค่าออกมาเป็น L และ H แสดงว่าเป็นสีน้ำเงิน ตามตารางในรูปนี้

S2	S3	PHOTODIODE TYPE
L	L	Red
L	H	Blue
H	L	Clear (no filter)
H	H	Green

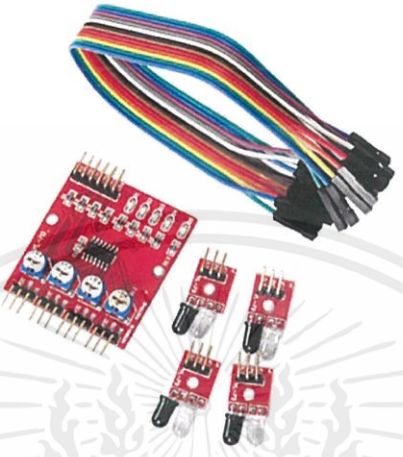
รูปที่ 2.28 ตารางสัญญาณเอาท์พุตขา s2 และ s3

สามารถขยายสัญญาณนี้ได้โดยควบคุมที่ขา s0 และ s1 โดย มี 4 ระดับ คือ ถ้า s0 กับ s1 เป็น L L หรือ 0 0 แปลว่าปิดการทำงาน ถ้า s0 กับ s1 เป็น H L แปลว่า ขยายสัญญาณที่ 20 %

S0	S1	OUTPUT FREQUENCY SCALING (f_o)
L	L	Power down
L	H	2%
H	L	20%
H	H	100%

รูปที่ 2.29 ตารางสัญญาณเอาท์พุตขา s0 และ s1

2.7 Tracking Sensor 4 Channel



รูปที่ 2.30 Tracking Sensor 4 Channel

เซนเซอร์สำหรับการตรวจจับเส้นสีขาวและดำตรวจจับเส้น ขาว-ดำ 4 จุดแยกอิสระ เพื่อตรวจจับเส้นหรือหลีกเลี่ยงสิ่งกีดขวาง ใช้ไฟ 3.3-5V ให้สัญญาณเอาต์พุตเป็นแบบ ดิจิตอล เมื่อมีวัตถุสีดำมาบังเซนเซอร์จะให้ค่า 0 เมื่อมีสีขาวจะให้ค่า 1 สามารถนำค่านี้ไปใช้กับ Arduino ได้ สามารถปรับระยะการเซนเซอร์ได้ที่ 0.1 - 60cm ใช้หลักการของการตรวจจับ Infrared ที่สะท้อนมาจากเส้น

2.7.1 คุณสมบัติโดยทั่วไป

- Operating Voltage: DC 3.3V-5V
- Working Current: try to choose more than 1A power supply
- Working temperature: - 10 ° C - + 50 ° C
- Installation aperture: M3 screw
- Detection distance: 1 mm to 60 CM adjustable, the closer the more stable performance, a white reflective distance.
- Size: the control board 42mm × 38mm × 12 mm (length × width × height)
- Small plate forward 25 mm × 12mm × 12 mm (length × width × height)
- Output interface: 6 wire interface (1234 to four signal outputs, + for the positive power supply, - for the negative supply that is ground)

- Output signal: TTL level (can be directly connected to the microcontroller I/0 number, reflected back to the sensor sensing infrared light , red light is on, the output low; when no infrared light , the light is not bright, high output power level.)

2.8 Servo motor

Servo motor คือ เป็นมอเตอร์ที่มีการควบคุมการเคลื่อนที่ของมัน (State) ไม่ว่าจะเป็นระยะ ความเร็ว มุมการหมุน โดยใช้การควบคุมแบบป้อนกลับ (Feedback control)

Feedback Control คือ ระบบควบคุมที่มีการวัดค่าเอาต์พุตของระบบนำมาเปรียบเทียบกับค่าอินพุตเพื่อควบคุมและปรับแต่งให้ค่าเอาต์พุตของระบบให้มีค่า เท่ากับ หรือ ใกล้เคียงกับค่าอินพุต

Stepper motor จะไม่มีการป้อนกลับ แต่ควบคุมการเคลื่อนที่แบบระบบเปิด โดยการส่งสัญญาณไปที่ตัว Stator ให้เคลื่อนที่ไปเป็น step ที่ตายตัว เช่น ทีละ 1.5 องศา เป็นต้น (ความละเอียดของการเคลื่อนที่ขึ้นกับคุณสมบัติของ Stepper motor และเทคนิคการควบคุม) ในขณะที่ Servo motor ต้องการสัญญาณป้อนกลับเพื่อใช้ในการประเมินตำแหน่ง หรือ ความเร็ว หรือ State อื่นๆ เพื่อไปประมวลการเคลื่อนไหวที่เหมาะสม เทียบกับตำแหน่งที่ผู้ใช้ระบุ โดยอาจจะใช้การควบคุมแบบปิดได้หลายๆ แบบ แล้วแต่ผู้ผลิต แต่แบบที่นิยมมากที่สุดก็คือ Proportional Integral Derivative (PID Control)

Servo motor ที่มีขายในปัจจุบันมีหลายแบบมาก เช่น แบบที่ประกอบด้วยมอเตอร์กระแสตรง (DC motor) แบบที่ใช้มอเตอร์กระแสสลับ (AC motor) แบบมอเตอร์ไร้แปรงถ่าน และมีหลากหลายวิธีการควบคุม

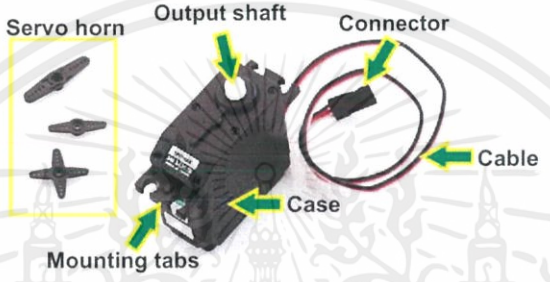
2.8.1 ข้อดีของ Servo motor

- สามารถให้ค่าทอร์กที่สูง
- สามารถเคลื่อนที่ความเร็วสูง
- ใช้งานกับการควบคุมความเร็วได้ดี
- มีหลากหลายขนาดให้เลือก (มากกว่า Stepper motor)
- เงียบ ต่างจาก Stepper motor
- ควบคุมได้ดีกว่า Stepper และไม่มีการสะดุดจังหวะเหมือนที่ Stepper motor เป็นในกรณีที่เจอกับทอร์กต้านสูงๆ

2.8.2 ข้อเสียของ Servo motor

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

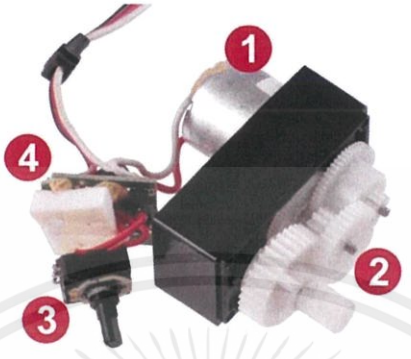
2.8.3 ส่วนประกอบภายนอก RC Servo Motor



รูปที่ 2.31 ส่วนประกอบภายนอก RC Servo Motor

- Case ตัวถัง หรือ กรอบของตัว Servo Motor
- Mounting Tab ส่วนจับยึดตัว Servo กับชิ้นงาน
- Output Shaft เพลาส่งกำลัง
- Servo Horns ส่วนเชื่อมต่อกับ Output shaft เพื่อสร้างกลไกล
- Cable สายเชื่อมต่อเพื่อ จ่ายไฟฟ้า และ ควบคุม Servo Motor จะประกอบด้วยสายไฟ 3 เส้น และ ใน RC Servo Motor จะมีสีของสายแตกต่างกันไปดังนี้
    - สายสีแดง คือ ไฟเลี้ยง (4.8-6V)
    - สายสีดำ หรือ น้ำตาล คือ กราวด์
    - สายสีเหลือง (ส้ม ขาว หรือฟ้า) คือ สายส่งสัญญาณพัลซ์ควบคุม (3-5V)
- Connector จุดเชื่อมต่อสายไฟ

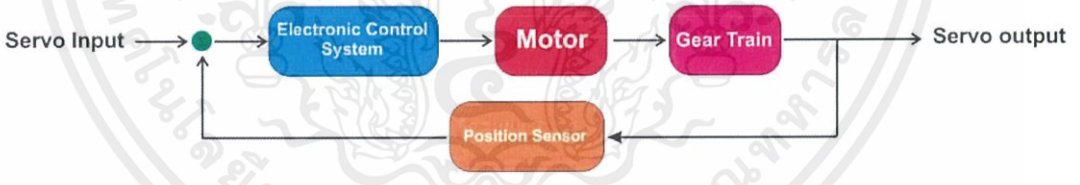
2.8.4 ส่วนประกอบภายใน RC Servo Motor



รูปที่ 2.32 ส่วนประกอบภายใน RC Servo Motor

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

2.8.5 Servo Motor Block Diagram



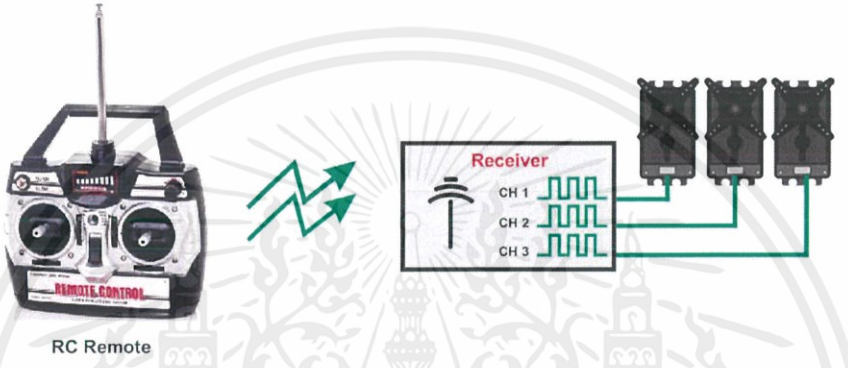
รูปที่ 2.33 Block Diagram

2.8.6 หลักการทำงานของ RC Servo Motor

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

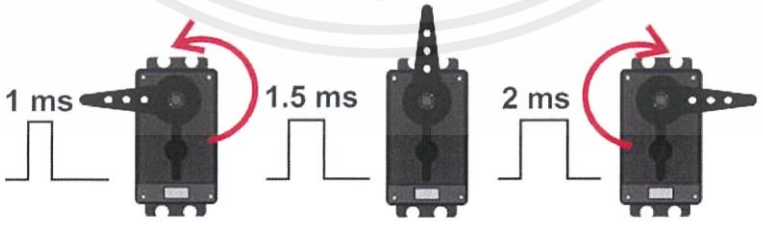
2.8.7 สัญญาณ RC ในรูปแบบ PWM

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



รูปที่ 2.34 สัญญาณ RC ในรูปแบบ PWM

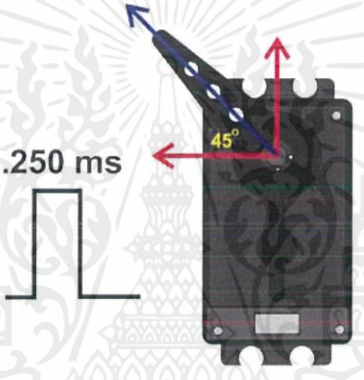
มุมหรือองศาจะขึ้นอยู่กับความกว้างของสัญญาณพัลซ์ ซึ่งโดยส่วนมากความกว้างของพัลซ์ที่ใช้ใน RC Servo Motor จะอยู่ในช่วง 1-2 ms หรือ 0.5-2.5 ms ยกตัวอย่างเช่นหากกำหนดความกว้างของสัญญาณพัลซ์ไว้ที่ 1 ms ตัว Servo Motor จะหมุนไปทางด้ายซ้ายจนสุด ในทางกลับกันหากกำหนดความกว้างของสัญญาณพัลซ์ไว้ที่ 2 ms ตัว Servo Motor จะหมุนไปยังตำแหน่งขวาสุด แต่หากกำหนดความกว้างของสัญญาณพัลซ์ไว้ที่ 1.5 ms ตัว Servo Motor ก็จะหมุนมาอยู่ที่ตำแหน่งตรงกลางพอดี



รูปที่ 2.35 การตอบสนองของเซอร์โว เมื่อจ่ายพัลซ์ขนาดต่างๆ

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

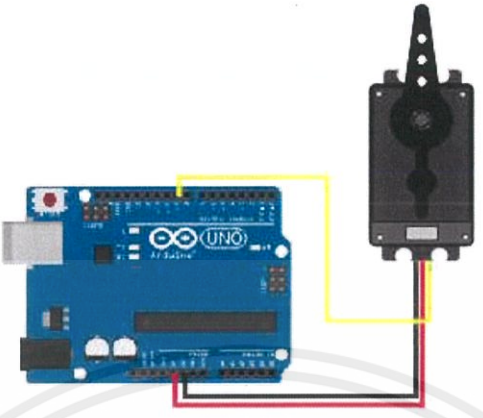
จากการหาค่าความกว้างพัลซ์ที่มุม 1 องศาข้างต้น หากต้องกำหนดให้ RC Servo Motor หมุนไปที่มุม 45 องศาจะหาค่าพัลซ์ที่ต้องการได้จาก 5.55 x 45 เท่ากับ 249.75 us แต่ที่มุม 0 องศาเราเริ่มที่ความกว้างพัลซ์ 1ms หรือ 1000 us เพราะฉะนั้นความกว้างพัลซ์ที่ใช้กำหนดให้ RC Servo Motor หมุนไปที่ 45 องศา คือ 1000 + 249.75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



รูปที่ 2.36 พัลซ์ขนาด 1.250 ms เซอร์โวหมุนไปที่ 45 องศา

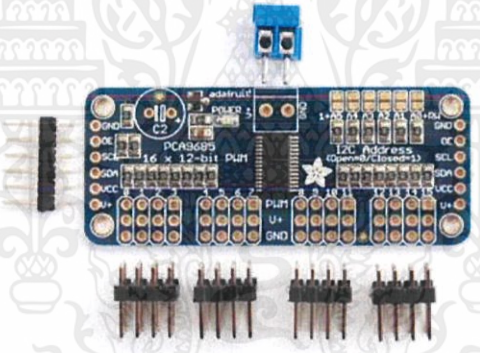
2.8.8 วิธีควบคุม RC Servo Motor ด้วย Arduino

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



รูปที่ 2.37 การเชื่อมต่อ RC Servo Motor เข้ากับบอร์ด Arduino

2.9 16-Channel 12-bit PWM Servo shield I2C interface PCA9685



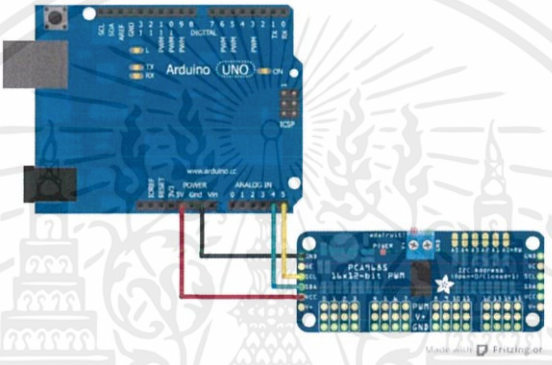
รูปที่ 2.38 Servo shield

โมดูลขับ Servo 16 ช่อง 12 bit แบบ PWM Servo ติดต่อแบบอินเตอร์เฟส I2C ใช้ IC Driver เบอร์ PCA9685 ในการควบคุม Servo 16 ตัว สามารถต่อโมดูลนี้หลายตัวกันโดยกำหนด address ของ I2C ให้ต่างกัน ก็จะสามารถควบคุม servo ได้มากขึ้น เช่น ถ้าต่อโมดูลเดียวควบคุมได้ 16 ตัว แต่ถ้าต่อ 2 โมดูล ก็สามารถควบคุมได้ 32 ตัว โดยใช้สายไฟเพียง 2 เส้น

2.9.1 คุณลักษณะ (Speccification)

R3 and later Arduino wiring (Uno, Mega & Leonardo)

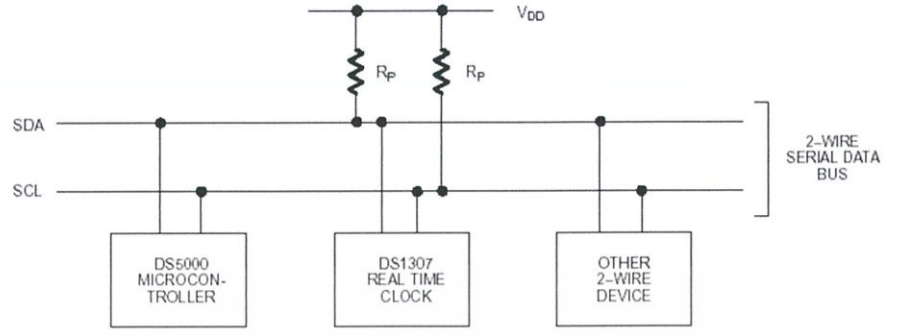
- +5v -> Vcc (สำหรับโมดูลขับ Servo 16 ช่อง 12 bit แบบ PWM Servo)
- +5v -> Vin (สำหรับเซอร์โว)
- GND -> GND
- SDA -> SDA
- SCL -> SCL



รูปที่ 2.39 การเชื่อมต่อ Servo Shield เข้ากับบอร์ด Arduino

2.9.2 การเชื่อมต่ออุปกรณ์แบบ I2C (I^2C)

I^2C = I2C Bus ย่อมาจาก Inter Integrate Circuit Bus (IIC) นิยมเรียกสั้นๆว่า BUS (ไอ-แสควร์-ซี-บัส) เป็นการสื่อสารอนุกรม แบบซิงโครนัส (Synchronous) เพื่อใช้ ติดต่อสื่อสาร ระหว่าง ไมโครคอนโทรลเลอร์ (MCU) กับอุปกรณ์ภายนอก ซึ่งถูกพัฒนาขึ้นโดยบริษัท Philips Semiconductors โดยใช้สายสัญญาณเพียง 2 เส้นเท่านั้น คือ serial data (SDA) และสาย serial clock (SCL) ซึ่งสามารถ เชื่อมต่ออุปกรณ์ จำนวนหลายๆ ตัว เข้าด้วยกันได้ ทำให้ MCU ใช้พอร์ตเพียง 2 พอร์ตเท่านั้น I^2C BUS ใช้สายสัญญาณ 2 เส้น คือ SCL ,SDA สำหรับติดกับอุปกรณ์แบบ 2 ทิศทาง โดยที่ขาสัญญาณทั้ง 2 จะต้องต่อกับตัวต้านทานแบบ pull up 2-10K เนื่องจากเอาต์พุตมีลักษณะเป็น แบบ Open Darin หรือเป็นแบบ Open Collector เพื่อให้เอาต์พุตเชื่อมต่อกันได้หลายตัว



รูปที่ 2.40 ลักษณะการการเชื่อมต่ออุปกรณ์แบบ I²C BUS

รูปที่ 2.41 รูปแบบการเขียน/อ่านข้อมูลแบบ I²C BUS

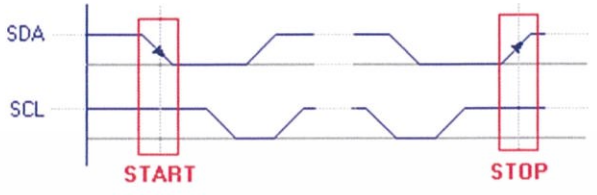
2.9.3 การเขียน-อ่านข้อมูลกับอุปกรณ์แบบ I²C BUS

การรับ-ส่งข้อมูลแบบ I²C BUSMCU จะเริ่มต้นการส่งข้อมูลด้วยการส่งสถานะเริ่มต้น (START Conditions) เพื่อแสดงการขอใช้บัส แล้วตามด้วย รหัสควบคุม (Control Byte) ซึ่งประกอบด้วยรหัส ประจำตัวอุปกรณ์ Device ID ,Device Address และ Mode ในการเขียนหรืออ่านข้อมูล เมื่ออุปกรณ์รับทราบว่า MCU ต้องการจะติดต่อด้วยก็ต้องส่งสถานะรับรู้ (Acknowledge) หรือแจ้งให้ MCU รับรู้ว่าข้อมูลที่ได้ส่งมามีความถูกต้องและเมื่อสิ้นสุดการส่งข้อมูล MCU จะต้องส่ง สถานะสิ้นสุด (STOP Conditions) เพื่อบอกกับอุปกรณ์ว่า สิ้นสุดการใช้บัส

2.9.4 สถานะบัสว่าง

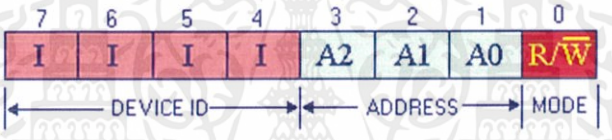
คือ เมื่อบัสไม่ได้ถูกใช้งาน ทั้ง SCL และ SDA จะเป็น 1 ทั้งคู่

2.9.5 การกำหนดสถานะเริ่มต้นและสถานะสิ้นสุดของ BUS



รูปที่ 2.42 I^2C BUS START and STOP Conditions

2.9.6 รหัสควบคุมของ I^2C BUS (Control Byte)



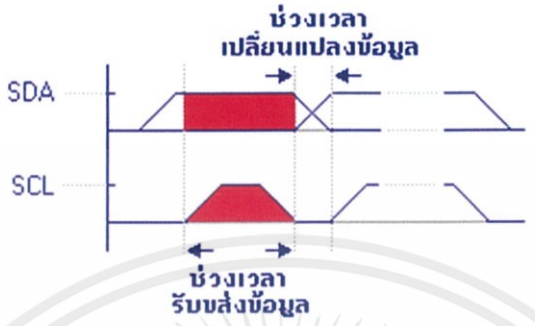
รูปที่ 2.43 I^2C BUS (Control Byte)

- รหัสควบคุมของ I^2C BUSประกอบด้วยรหัสประจำตัวของอุปกรณ์ (Device ID) ประกอบด้วยบิต 1-7 และบิต 0 เป็นบิตควบคุมการเขียนอ่าน

- รหัสประจำตัวของอุปกรณ์ ประกอบด้วยรหัสประจำตัวจากผู้ผลิต Product ID 4 บิต (บิต 4-7) ที่เปลี่ยนแปลงแก้ไขไม่ได้ และ Device Address 3 บิต (บิต 1-3) ซึ่งผู้ใช้ สามารถ กำหนด เองได้ รวมแล้วเป็นรหัส 7 บิท ใช้ระบุตัวอุปกรณ์ ที่ต่ออยู่บนบัส จะมีค่าซ้ำกันไม่ได้

- บิตควบคุมการเขียนอ่าน (Mode) บิต 0 เมื่อ MCU ต้องการเขียนข้อมูลไปยังอุปกรณ์ก็กำหนดให้บิตนี้เป็น 0 และเมื่อต้องการ อ่านข้อมูล จากอุปกรณ์ ก็กำหนดให้บิตนี้เป็น 1

2.9.7 ช่วงเวลารับส่งบิตข้อมูลของ I^2C BUS



รูปที่ 2.44 การรับส่งบิตข้อมูลของ I^2C BUS

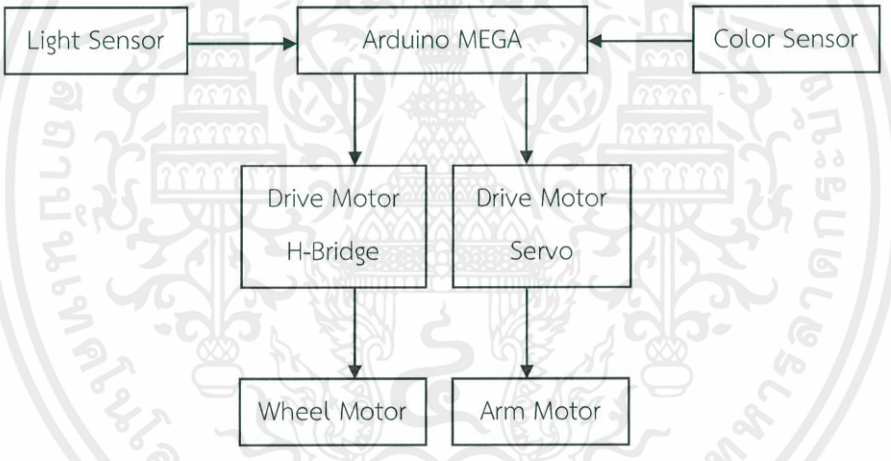
- สภาวะการรับ-ส่งข้อมูล จะกระทำในขณะที่ขา SCL เป็น 1
- สภาวะการเปลี่ยนแปลงข้อมูล จะกระทำในขณะที่ขา SCL เป็น 0

บทที่ 3
หลักการทำงาน

3.1 หลักการทำงานของระบบ หุ่นยนต์เก็บของอัตโนมัติ

หุ่นยนต์เก็บของอัตโนมัติ สามารถแบ่งเป็นส่วนประกอบในการทำงานออกเป็น 2 ส่วน คือ ส่วนแรกเป็นการตรวจจับเส้นควบคุมหุ่นยนต์ให้วิ่งตามเส้น ส่วนที่ 2 คือการให้หุ่นยนต์หยิบจับวัตถุเป้าหมาย

ส่วนวิธีการสำหรับโครงงานนี้แบ่งออกเป็น 2 ส่วน ส่วนแรกคือการสร้างหุ่นยนต์ให้สามารถวิ่งไปตามเส้นแบบอัตโนมัติโดยไม่ออกนอกเส้น และอีกส่วนคือการให้หุ่นยนต์ตรวจจับวัตถุแล้วนำไปวางไว้ในกล่องสีที่มีสีเหมือนกับวัตถุแบบอัตโนมัติ

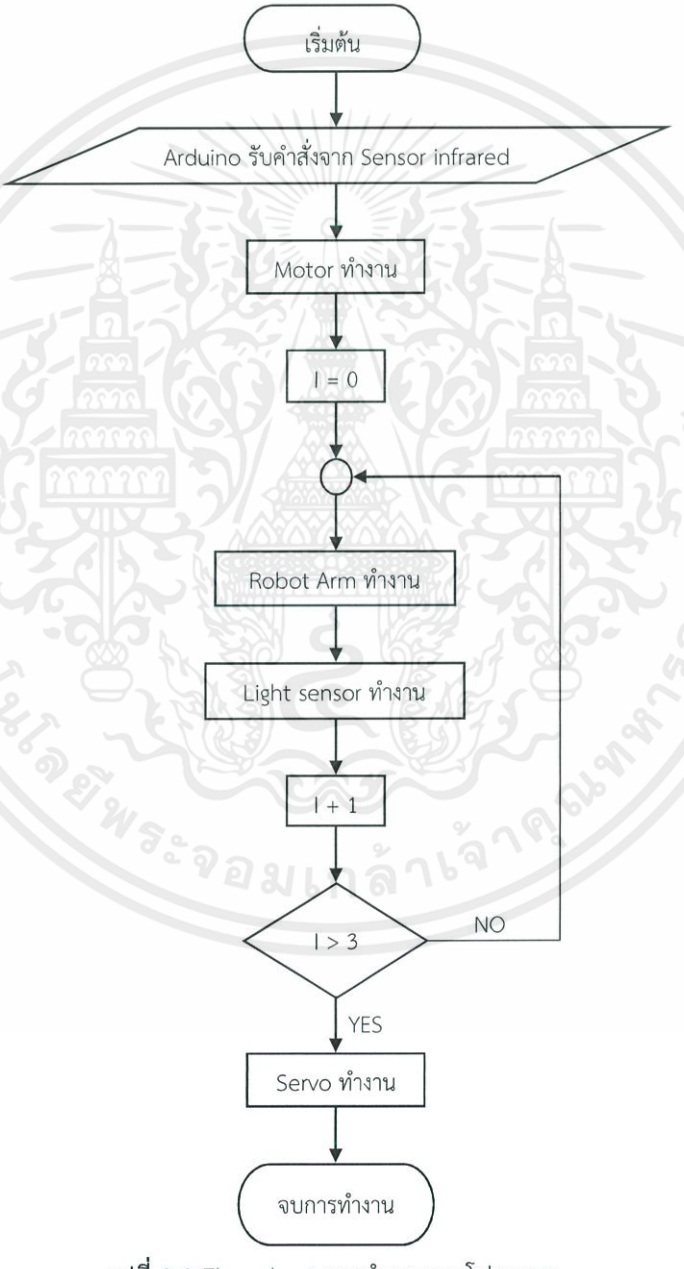


รูปที่ 3.1 Block Diagram ของระบบ

โดยโครงงานนี้ได้ใช้อุปกรณ์ Arduino Mega ในการควบคุมการทำงานของหุ่นยนต์ และประยุกต์ใช้งานร่วมกับกล้อง ในการตรวจจับสีของวัตถุเป้าหมาย แล้วให้แขนของหุ่นยนต์หยิบวัตถุไปวางไว้ในกล่องที่มีสีต่างๆตามสีของวัตถุ

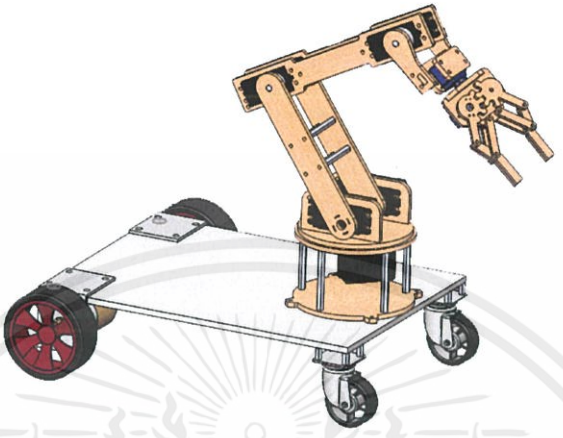
3.2 การทำงานของโปรแกรม

เมื่อเริ่มการทำงาน arduino รับค่ามาจากเซ็นเซอร์อินฟาเรต และสั่งให้มอเตอร์ทำงาน ตัวรถจะวิ่งไปตามเส้น เมื่อถึงจุดที่กำหนด โรบอทอาร์มทำงานโดยหยิบวัตถุ จากนั้นเซ็นเซอร์แสงจะส่งค่าไปให้ Arduino ประมวลผลสีของวัตถุ โรบอทอาร์มจะนำวัตถุไปวางไว้ในจุดที่กำหนดตามสีของวัตถุ ทำซ้ำจนครบทุกสี จากนั้นมอเตอร์จะทำงาน ให้รถเคลื่อนที่ไปหยุดที่จุดที่กำหนด



รูปที่ 3.2 Flowchart การทำงานของโปรแกรม

3.3 ออกแบบโครงสร้างในโปรแกรม Solidwork



รูปที่ 3.3 โครงสร้างที่ออกแบบจากโปรแกรม Solidwork

จากรูปที่ 3.3 เป็นการออกแบบโครงสร้างรวมของหุ่นยนต์ทั้งหมดนั้นเราได้เริ่มจากการนำข้อมูลทั้งหมดที่ได้จากการออกแบบใน AutoCad รวมถึง Dimension ของโครงสร้างและอุปกรณ์ต่างๆ นำมาออกแบบทีละชิ้นส่วน จากนั้นจึงนำแต่ละชิ้นส่วนนั้นมาประกอบเข้าด้วยกัน

บทที่ 4
การทดลองและผลการทดลอง

4.1 ขั้นตอนการทดลอง

ตอนที่ 1 การทดสอบทิศทางการขับมอเตอร์ โดยวัดไฟเข้า – ออกมอเตอร์

เขียนโปรแกรมให้กับบอร์ด Arduino เมื่อจ่ายไฟให้กับช่องทางต่างๆ จะทำให้รถวิ่งไปในทิศทางต่างกัน

ตอนที่ 2 การควบคุมให้หุ่นยนต์เคลื่อนที่ไปตามเส้นตรงและเส้นโค้ง

เขียนโปรแกรมเพื่อควบคุมหุ่นยนต์เคลื่อนที่ไปตามแนวเส้นนั้น ลักษณะของเส้นที่เป็นแนวตรงและเส้นโค้ง โดยใช้เงื่อนไขง่ายๆในการตรวจสอบและควบคุม

ตอนที่ 3 การกำหนดลำดับการหยิบวัตถุ และตำแหน่งการวางวัตถุ

เขียนโปรแกรมเพื่อกำหนดลำดับในการให้โรบอทอาร์มหยิบวัตถุสีต่างกัน จากตำแหน่งที่ต่างกัน และนำไปวางไว้ในตำแหน่งที่กำหนดไว้ตามสีของวัตถุ

ตอนที่ 4 ความแม่นยำในการวางวัตถุตามจุดที่กำหนด

ทดสอบความแม่นยำในการหยิบวัตถุต่างสีกันไปวางไว้ในตำแหน่งที่มีการกำหนดสีไว้

4.2 ผลการทดลอง

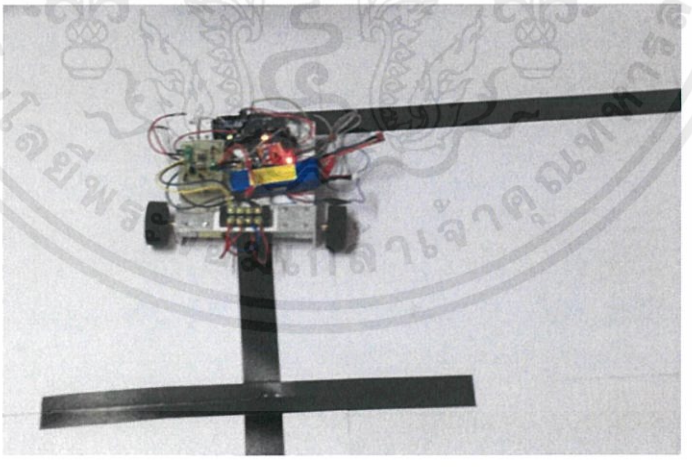
ตอนที่ 1 การทดสอบทิศทางการขับมอเตอร์ โดยวัดไฟเข้า – ออกมอเตอร์

ตารางที่ 4.1 ผลการทดสอบทิศทางการขับมอเตอร์ โดยวัดไฟเข้า – ออกมอเตอร์

มอเตอร์	V_{In}		V_{Out}	ทิศทางการเคลื่อนที่
	ช่องที่ 1	ช่องที่ 2		
มอเตอร์ ซ้าย	0	0	0	ไม่เคลื่อนที่
	1.3	0	1.3	ข้างหน้า
	0	1.3	-1.3	ข้างหลัง
มอเตอร์ ขวา	0	0	0	ไม่เคลื่อนที่
	1.3	0	1.3	ข้างหน้า
	0	1.3	-1.3	ข้างหลัง

ผลการทดลอง เมื่อจ่ายไฟให้มอเตอร์ผ่านช่องที่หนึ่งพบว่าล้อทั้งสองข้างจะเคลื่อนที่ไปข้างหน้า แต่เมื่อจ่ายไฟให้มอเตอร์ผ่านช่องที่สองล้อทั้งสองข้างจะเคลื่อนที่ไปข้างหลัง

ตอนที่ 2 การควบคุมให้หุ่นยนต์เคลื่อนที่ไปตามทางตรงและทางเลี้ยว



รูปที่ 4.1 หุ่นยนต์เคลื่อนที่ไปตามทางตรงและทางเลี้ยว

ผลการทดลอง หุ่นยนต์จะเคลื่อนที่ไปตามเส้นโดยจับเส้น เมื่อเจอทางแยกไปขวาหรือซ้ายก็จะเลี้ยวตามทางแยก

ตอนที่ 3 กำหนดลำดับการหยิบวัตถุ และตำแหน่งการวางวัตถุ

ตารางที่ 4.2 การกำหนดลำดับการหยิบวัตถุและตำแหน่งการวางวัตถุ

	สีของวัตถุ		
	สีแดง	สีเขียว	สีน้ำเงิน
ลำดับการหยิบวัตถุ	1	2	3
	2	3	1
	3	1	2

ผลการทดลอง ลำดับการหยิบวัตถุไม่มีผลต่อการอ่านค่าสีของตัวเซนเซอร์ เซนเซอร์ยังสามารถแยกสีและวางได้อย่างถูกต้อง

ตอนที่ 4 ความแม่นยำในการวางวัตถุตามจุดที่กำหนด

ตารางที่ 4.3 ความแม่นยำในการวางวัตถุตามจุดที่กำหนด

ครั้งที่	แดง	น้ำเงิน	เขียว
1	1	1	1
2	1	1	0
3	1	1	1
4	1	1	1
5	0	1	0
6	1	1	1
ทั้งหมด	5	6	4
%	83.33	100	66.67

ผลการทดลอง จากตารางจะพบว่าสีน้ำเงินจะอ่านค่าสีได้ถูกต้องแม่นยำเสมอ รองลงมาคือสีแดงที่อ่านค่าสีผิดไปหนึ่งครั้งจากหกครั้ง ส่วนสีเขียวอ่านค่าสีผิดสองครั้งจากหกครั้ง เซนเซอร์สามารถอ่านค่าสีเขียวได้ต่ำสุดและอ่านค่าสีน้ำเงินได้ดีที่สุด

บทที่ 5
สรุปผลการทดลองและข้อเสนอแนะ

5.1 สรุปผลการทดลอง

ตอนที่ 1 การทดสอบทิศทางการขับมอเตอร์ โดยวัดไฟเข้า – ออกมอเตอร์

เมื่อจ่ายไฟให้มอเตอร์ซ้ายและมอเตอร์ขวาด้วยช่องที่ 1 เหมือนกัน จะทำให้รถวิ่งตรงไปด้านหน้า และเมื่อจ่ายไฟให้มอเตอร์ซ้ายและมอเตอร์ขวาด้วยช่องที่ 2 เหมือนกันจะทำให้รถวิ่งไปด้านหลัง หากต้องการให้เลี้ยวซ้าย จ่ายไฟให้มอเตอร์ขวาข้างเดียว หากต้องการให้เลี้ยวขวา จ่ายไฟให้มอเตอร์ซ้ายข้างเดียว

ตอนที่ 2 การควบคุมให้หุ่นยนต์เคลื่อนที่ไปตามทางตรงและทางโค้ง

เป็นการทดสอบโดยการนำตัวรถหุ่นยนต์ที่เขียนโปรแกรมลงบน Arduino เสร็จแล้ว ไปวางไว้บนสนามที่มีการลากเส้นสีดำไว้เพื่อสังเกตผลการทำงาน พบว่าตัวรถสามารถวิ่งไปตามเส้นได้โดยไม่หลุดออกนอกเส้น และเมื่อถึงทางโค้ง ตัวรถสามารถเลี้ยวตามเส้นได้อย่างถูกต้อง และไม่ออกนอกเส้นทาง

ตอนที่ 4 การกำหนดลำดับการหยิบวัตถุ และตำแหน่งการวางวัตถุ

เมื่อโรบอทอาร์มหยิบวัตถุสีแดงจะวางวัตถุสีแดงที่ตำแหน่ง a ถ้าหยิบวัตถุสีเขียวจะวางที่ตำแหน่ง b และถ้าหากหยิบวัตถุสีน้ำเงินจะวางที่ตำแหน่ง c เสมอ ไม่ว่าจะหยิบวัตถุสีใดก่อนก็ตาม จึงสรุปได้ว่า หุ่นยนต์เก็บของอัตโนมัติสามารถวิ่งตามเส้นและเคลื่อนย้ายวัตถุไปวางที่ตำแหน่งที่ต้องการตามสีได้

ตอนที่ 5 ความแม่นยำในการวางวัตถุตามจุดที่กำหนด

จากการทดลองวางวัตถุไว้ที่ตำแหน่งต่างๆ และให้โรบอทอาร์มหยิบวัตถุต่างสีไปวางไว้ในตำแหน่งที่กำหนดไว้ตามสี และบันทึกผลพบว่า โรบอทอาร์มยังมีการนำวัตถุไปวางไว้ในตำแหน่งที่ไม่ตรงกับสีของวัตถุอยู่บ้าง เนื่องจากเซ็นเซอร์มีการประมวลผลผิด

5.2 ปัญหาที่เกิดขึ้นและแนวทางการแก้ไข

จากการพัฒนาชิ้นงานทั้งหมด ถึงแม้ว่าหุ่นยนต์เก็บของอัตโนมัติจะสามารถทำตามเป้าหมายที่ตั้งไว้ได้ แต่ยังคงประสบกับปัญหาที่ยังไม่สามารถแก้ไขได้ ดังนี้

ตัวเซ็นเซอร์มีความผิดพลาดสูงเมื่อวางวัตถุไว้ในระยะไกลกว่า 2 เซนติเมตร จึงจำเป็นต้องเขียนโปรแกรมสั่งใส่โรบอทอาร์มหยิบวัตถุมาไว้ในระยะที่เซ็นเซอร์มีความแม่นยำสูงก่อนจะประมวลผลค่าสีและนำไปวางไว้ในตำแหน่งต่างๆ

แบตเตอรี่ เมื่อใช้งานสักระยะ และมีการชาร์ตไฟ พบว่าเมื่อนำไปจ่ายไฟให้กับตัวหุ่นยนต์ จะเกิดการหยุดทำงาน อาจเนื่องมาจากแบตเตอรี่จ่ายไฟไม่เสถียร ทำให้มีอาการไฟตก บอร์ด Arduino จึงเกิดการหยุดชะงัก ส่งผลให้โปรแกรมไม่ทำงาน และตัวหุ่นยนต์หยุดการทำงานไป

ผู้ทำโครงงานยังขาดความรู้ความเข้าใจด้านเครื่องกลและด้านการเขียนโปรแกรมอย่างลึกซึ้ง จึงทำให้ไม่สามารถแก้ไขปัญหาทั้งหมดที่เกิดขึ้นได้

โรบอทอาร์มที่ทำมาจากแผ่นอะคริลิคมีความเปราะบาง การประกอบโรบอทอาร์ม หรือการเขียนโปรแกรมสั่งในเซอร์โวมอเตอร์ทำงานจึงจำเป็นต้องมีความระมัดระวังสูง

เนื่องจากโรบอทอาร์มมีน้ำหนักค่อนข้างมาก จึงทำให้เซอร์โวมอเตอร์ต้องรับภาระมาก ในบางครั้งแขนของโรบอทอาร์มไม่สามารถยกขึ้นได้ วิธีแก้คือเขียนโปรแกรมให้โรบอทอาร์มตั้งอยู่ในองศาที่รับภาระน้อยที่สุด และจ่ายกระแสไฟให้เพียงพอ

ตัวรถไม่สามารถเลี้ยวได้ทัน ทำให้รถวิ่งหลุดออกนอกเส้น อาจเนื่องมาจากเซ็นเซอร์ประมวลผลไม่เร็วพอ หรือแสงสว่างที่น้อยจนเซ็นเซอร์ไม่สามารถจับสีของเส้นได้ วิธีแก้คือเขียนโปรแกรมให้รถวิ่งช้าลง ทำระยะวิ่งให้ยาวขึ้นถ้าตัวรถมีความยาวมาก และวางรถไว้ในที่ๆมีแสงสว่างอย่างเพียงพอ ก็จะสามารถแก้ไขปัญหาดังกล่าวได้

เอกสารอ้างอิง

[1] Randy H. Shih. (2010) : "*AutoCAD 2011 Tutorial Book*",Schroff Development Corporation.

[2] LESLIE FELDMAN. (2011) : "*AutoCAD® 2012 Preview Guide*",Autodesk Inc.

[3] Dassault Systems Solidworks Corporation. (2010) : "*Student's Guide to Learning Solidworks Software*", Dassault Systems S.A. company, PMS0119-ENG, Massachusetts.

[4] Mountain A. (2014) : "*Arduino and Motor Control : Part 1*", Arduitronics, 2014, PP 1-2.

[5] John-David Warren, Josh Adams and Harald Molle. (2011) : "*Arduino Robotics*", Springer Science+Business Media, New York.

[6] Owen Bishop. (2007) : "*Robot Builder's Cookbook*", Elsevier Ltd, Massachusetts

ภาคผนวก

ภาคผนวก ก.

โปรแกรมของหุ่นยนต์เก็บของอัตโนมัติ

```
#include <Wire.h>
#include
    <Adafruit_PWMServoDriver.h>

// called this way, it uses the
    default address 0x40
Adafruit_PWMServoDriver pwm =
    Adafruit_PWMServoDriver(0x40);

// you can also call it with a
    different address you want
//Adafruit_PWMServoDriver pwm =
    Adafruit_PWMServoDriver(0x41);

// Depending on your servo make,
    the pulse width min and max
    may vary, you
// want these to be as small/large
    as possible without hitting the
    hard stop
// for max range. You'll have to
    tweak them as necessary to
    match the servos you
// have!
#define SERVOMIN  150 // this is
    the 'minimum' pulse length
    count (out of 4096)
#define SERVOMAX  600 // this is
    the 'maximum' pulse length
    count (out of 4096)

// our servo # counter
int a1 = 2;
int a2 = 3;
int b1 = 4;
int b2 = 5;
uint8_t servonum = 0;
uint8_t servonum1 = 1;
const int s0 = 43; // sensor pins
const int s1 = 42;
const int s2 = 46;
const int s3 = 47;
const int nLED = 44; // illuminating
    LED
const int out = 45; // TCS230
    output
// variables to store color values
int red = 0;
int green = 0;
int blue = 0;
int ref = 511;
int p0;
```

```
int p1;
int p2;
int p3;
int p4;
int p5;
int p6;
int p7;
int p8;
int p9;
int p10;
int pp0;
int pp1;
int pp2;
int pp3;
int pp4;
int pp5;
int pp6;
int pp7;
int pp8;
int pp9;
int pp10;
int ppp0;
int ppp1;
int ppp2;
int ppp3;
int ppp4;
int ppp5;
int ppp6;
int ppp7;
int ppp8;
int ppp9;
int ppp10;
int ppp11;
int d0;
int d1;
int d2;
int d3;
int d4;
int d5;
int d6;
int d7;
int dd0;
int dd1;
int dd2;
int dd3;
int dd4;
int dd5;
int dd6;
int dd7;
int dd8;
```

```
int dd9;
int ddd0;
int ddd1;
int ddd2;
int ddd3;
int ddd4;
int ddd5;
int ddd6;
int ddd7;
int ddd8;
int ddd9;
int st0;
int st1;
int st2;
int st3;
int stt0;
int stt1;
int stt2;
int stt3;
int sttt0;
int sttt1;
int sttt2;
int sttt3;
int sst0;
int sst1;
int sst2;
int sst3;
int t1;
int t2;
int m1;
int m2;

void setup() {
  pinMode(s0, OUTPUT);
  pinMode(s1, OUTPUT);
  pinMode(s2, OUTPUT);
  pinMode(s3, OUTPUT);
  pinMode(nLED, OUTPUT);
  pinMode(out, INPUT);
  digitalWrite(s0, HIGH);
  digitalWrite(s1, HIGH);
  digitalWrite(nLED, LOW);
  Serial.begin(9600);
  Serial.println("16 channel Servo
      test!");
  pwm.begin();
  pwm.setPWMFreq(60);  // Analog
      servos run at ~60 Hz updates
}
```

```
// you can use this function if you'd
    like to set the pulse length in
    seconds
// e.g. setServoPulse(0, 0.001) is a
    ~1 millisecond pulse width. its
    not precise!
void setServoPulse(uint8_t n,
    double pulse) {
  double pulselength;
  pulselength = 1000000;   //
    1,000,000 us per second
  pulselength /= 60;   // 60 Hz
  Serial.print(pulselength);
    Serial.println(" us per period");
  pulselength /= 4096;  // 12 bits of
    resolution
  Serial.print(pulselength);
    Serial.println(" us per bit");
  pulse *= 1000;
  pulse /= pulselength;
  Serial.println(pulse);
  pwm.setPWM(n, 0, pulse);
}

void loop() {
  int right1 = analogRead(A0);
  int right2 = analogRead(A1);
  int left1 = analogRead(A2);
  int left2 = analogRead(A3);

if((left1<511)&&(left2<511)&&(right1
    <511)&&(right2<511))
{
  font();
  m1=0;
}
if((left1>511)&&(left2<511)&&(right1
    <511)&&(right2>511))
{
  t1++;
  stopp();
  if(t1==1){
   t2++;
   if(t2==1){
    if(sst0<30){
   for(sst0=0;sst0<30;sst0++){
    if(sst0==2){
     moveDegree4(4,70);
     moveDegree5(5,70);
     delay(1000);
    }
   }
```

```
}
if(sst1<30){
  for(sst1=0;st1<30;st1++){
    if(sst1==2){
      moveDegree2(2,150);
      delay(1000);
    }
  }
}
if(sst2<30){
  for(sst2=0;sst2<30;sst2++){
    if(sst2==2){
      moveDegree6(6,112);
      delay(1000);
    }
  }
}
if(p6<30){
  for(p6=0;p6<30;p6++){
    if(p6==2){
      moveDegree6(6,112);
      delay(1000);
    }
  }
}
if(p7<30){
  for(p7=0;p7<30;p7++){
    if(p7==2){
      moveDegree2(2,130);
      delay(1000);
    }
  }
}
if(p8<30){
  for(p8=0;p8<30;p8++){
    if(p8==2){
      moveDegree3(3,60);
      delay(1000);
    }
  }
}
if(p1<30){
  for(p1=0;p1<30;p1++){
    if(p1==2){
      moveDegree0(0,10);
      delay(1000);
    }
  }
}
if(p5<30){
```

```
  for(p5=0;p5<30;p5++){
    if(p5==2){
      moveDegree4(4,100);
      moveDegree5(5,100);
      delay(1000);
    }
  }
}
if(p4<30){
  for(p4=0;p4<30;p4++){
    if(p4==2){
      moveDegree3(3,60);
      delay(1000);
    }
  }
}
if(p3<30){
  for(p3=0;p3<30;p3++){
    if(p3==2){
      moveDegree2(2,130);
      delay(1000);
    }
  }
}
if(p0<30){
  for(p0=0;p0<30;p0++){
    if(p0==2){
      moveDegree0(0,56);
      delay(1000);
    }
  }
}
if(p9<30){
  for(p9=0;p9<30;p9++){
    if(p9==2){
      moveDegree3(3,58);
      delay(1000);
      color();
Serial.print("R");
Serial.print(red, DEC);
Serial.print(" G");
Serial.print(green, DEC);
Serial.print(" B");
Serial.print(blue, DEC);
Serial.println();
    }
  }
}
}
if(t2==2){
```

```
  if(pp6<30){
  for(pp6=0;pp6<30;pp6++){
    if(pp6==2){
      moveDegree6(6,112);
      delay(1000);
    }
  }
}
if(pp7<30){
  for(pp7=0;pp7<30;pp7++){
    if(pp7==2){
      moveDegree2(2,130);
      delay(1000);
    }
  }
}
if(pp8<30){
  for(pp8=0;pp8<30;pp8++){
    if(pp8==2){
      moveDegree3(3,80);
      delay(1000);
    }
  }
}
if(pp1<30){
  for(pp1=0;pp1<30;pp1++){
    if(pp1==2){
      moveDegree0(0,10);
      delay(1000);
    }
  }
}
if(pp5<30){
  for(pp5=0;pp5<30;pp5++){
    if(pp5==2){
      moveDegree4(4,107);
      moveDegree5(5,107);
      delay(1000);
    }
  }
}
if(pp4<30){
  for(pp4=0;pp4<30;pp4++){
    if(pp4==2){
      moveDegree3(3,80);
      delay(1000);
    }
  }
}
if(pp3<30){
```

```
  for(pp3=0;pp3<30;pp3++){
    if(pp3==2){
      moveDegree2(2,130);
      delay(1000);
    }
  }
}
if(pp0<30){
  for(pp0=0;pp0<30;pp0++){
    if(pp0==2){
      moveDegree0(0,56);
      delay(1000);
    }
  }
}
if(pp10<30){
  for(pp10=0;pp10<30;pp10++){
    if(pp10==2){
      moveDegree4(4,105);
      moveDegree5(5,105);
      delay(1000);
    }
  }
}
if(pp9<30){
  for(pp9=0;pp9<30;pp9++){
    if(pp9==2){
      moveDegree3(3,55);
      delay(1000);
      color();
Serial.print("R");
Serial.print(red, DEC);
Serial.print(" G");
Serial.print(green, DEC);
Serial.print(" B");
Serial.print(blue, DEC);
Serial.println();
    }
  }
}
}
if(t2==3){
  if(ppp6<30){
  for(ppp6=0;ppp6<30;ppp6++){
    if(ppp6==2){
      moveDegree6(6,112);
      delay(1000);
    }
  }
}
```

```
if(ppp7<30){
  for(ppp7=0;ppp7<30;ppp7++){
    if(ppp7==2){
      moveDegree2(2,130);
      delay(1000);
    }
  }
}
if(ppp8<30){
  for(ppp8=0;ppp8<30;ppp8++){
    if(ppp8==2){
      moveDegree3(3,100);
      delay(1000);
    }
  }
}
if(ppp1<30){
  for(ppp1=0;ppp1<30;ppp1++){
    if(ppp1==2){
      moveDegree0(0,10);
      delay(1000);
    }
  }
}
if(ppp5<80){
  for(ppp5=0;ppp5<80;ppp5++){
    if(ppp5==2){
      moveDegree4(4,115);
      moveDegree5(5,115);
      delay(2000);
    }
  }
}
if(ppp4<30){
  for(ppp4=0;ppp4<30;ppp4++){
    if(ppp4==2){
      moveDegree3(3,100);
      delay(1000);
    }
  }
}
if(ppp3<30){
  for(ppp3=0;ppp3<30;ppp3++){
    if(ppp3==2){
      moveDegree2(2,130);
      delay(1000);
    }
  }
}
if(ppp0<30){
```

```
  for(ppp0=0;ppp0<30;ppp0++){
    if(ppp0==2){
      moveDegree0(0,56);
      delay(1000);
    }
  }
}
if(ppp10<30){
    for(ppp10=0;ppp10<30;ppp10+
    +){
    if(ppp10==2){
      moveDegree4(4,105);
      moveDegree5(5,105);
      delay(1000);
    }
  }
}
if(ppp11<30){
    for(ppp11=0;ppp11<30;ppp11+
    +){
    if(ppp11==2){
      moveDegree2(2,150);
      delay(1000);
    }
  }
}
if(ppp9<30){
  for(ppp9=0;ppp9<30;ppp9++){
    if(ppp9==2){
      moveDegree3(3,75);
      delay(1000);
      color();
Serial.print("R");
Serial.print(red, DEC);
Serial.print(" G");
Serial.print(green, DEC);
Serial.print(" B");
Serial.print(blue, DEC);
Serial.println();
    }
  }
}
if(t2==4){
  back();
  delay(5000);
}
}
Serial.println(t1);
```

```
Serial.print("t2");
Serial.println(t2);

if (red < blue && red < green)
{
  t1=0;
    if(d1<30){
  for(d1=0;d1<30;d1++){
    if(d1==2){
      moveDegree4(4,70);
      moveDegree5(5,70);
      delay(1000);
    }
  }
}
if(d0<30){
  for(d0=0;d0<30;d0++){
    if(d0==2){
      moveDegree3(3,90);
      delay(1000);
    }
  }
}
if(d2<30){
  for(d2=0;d2<30;d2++){
    if(d2==2){
      moveDegree2(2,150);
      delay(1000);
    }
  }
}
if(d3<30){
  for(d3=0;d3<30;d3++){
    if(d3==2){
      moveDegree6(6,30);
      delay(1000);
    }
  }
}
if(d4<30){
  for(d4=0;d4<30;d4++){
    if(d4==2){
      moveDegree4(4,105);
      moveDegree5(5,105);
      delay(1000);
    }
  }
}
if(d6<30){
  for(d6=0;d6<30;d6++){
```

```
    if(d6==2){
      moveDegree0(0,10);
      delay(1000);
    }
  }
}
if(st0<30){
  for(st0=0;st0<30;st0++){
    if(st0==2){
      moveDegree4(4,70);
      moveDegree5(5,70);
      delay(1000);
    }
  }
}
if(st1<30){
  for(st1=0;st1<30;st1++){
    if(st1==2){
      moveDegree2(2,150);
      delay(1000);
    }
  }
}
if(st2<30){
  for(st2=0;st2<30;st2++){
    if(st2==2){
      moveDegree6(6,112);
      delay(1000);
    }
  }
 }
}
else if (green < red && green <
    blue)
{
 t1=0;
 if(dd1<30){
 for(dd1=0;dd1<30;dd1++){
  if(dd1==2){
    moveDegree4(4,70);
    moveDegree5(5,70);
    delay(1000);
  }
 }
}
if(dd0<30){
 for(dd0=0;dd0<30;dd0++){
  if(dd0==2){
    moveDegree3(3,90);
    delay(1000);
```

```
    }
  }
}
if(dd2<30){
  for(dd2=0;dd2<30;dd2++){
    if(dd2==2){
      moveDegree2(2,150);
      delay(1000);
    }
  }
}
if(dd3<30){
  for(dd3=0;dd3<30;dd3++){
    if(dd3==2){
      moveDegree6(6,170);
      delay(1000);
    }
  }
}
if(dd4<30){
  for(dd4=0;dd4<30;dd4++){
    if(dd4==2){
      moveDegree4(4,105);
      moveDegree5(5,105);
      delay(1000);
    }
  }
}
if(dd6<30){
  for(dd6=0;dd6<30;dd6++){
    if(dd6==2){
      moveDegree0(0,10);
      delay(1000);
    }
  }
}
if(stt0<30){
  for(stt0=0;stt0<30;stt0++){
    if(stt0==2){
      moveDegree4(4,70);
      moveDegree5(5,70);
      delay(1000);
    }
  }
}
if(stt1<30){
  for(stt1=0;stt1<30;stt1++){
    if(stt1==2){
      moveDegree2(2,150);
      delay(1000);
```

```
    }
  }
}
if(stt2<30){
  for(stt2=0;stt2<30;stt2++){
    if(stt2==2){
      moveDegree6(6,112);
      delay(1000);
    }
  }
}
}
else if (blue < red && blue < green)
{
  t1=0;
  if(ddd1<30){
  for(ddd1=0;ddd1<30;ddd1++){
    if(ddd1==2){
      moveDegree4(4,70);
      moveDegree5(5,70);
      delay(1000);
    }
  }
}
if(ddd0<30){
  for(ddd0=0;ddd0<30;ddd0++){
    if(ddd0==2){
      moveDegree3(3,90);
      delay(1000);
    }
  }
}
if(ddd2<30){
  for(ddd2=0;ddd2<30;ddd2++){
    if(ddd2==2){
      moveDegree2(2,150);
      delay(1000);
    }
  }
}
if(ddd3<30){
  for(ddd3=0;ddd3<30;ddd3++){
    if(ddd3==2){
      moveDegree6(6,60);
      delay(1000);
    }
  }
}
if(ddd4<30){
  for(ddd4=0;ddd4<30;ddd4++){
```

```
    if(ddd4==2){
      moveDegree4(4,105);
      moveDegree5(5,105);
      delay(1000);
    }
  }
}
if(ddd6<30){
  for(ddd6=0;ddd6<30;ddd6++){
    if(ddd6==2){
      moveDegree0(0,10);
      delay(1000);
    }
  }
}
if(sttt0<30){
  for(sttt0=0;sttt0<30;sttt0++){
    if(sttt0==2){
      moveDegree4(4,70);
      moveDegree5(5,70);
      delay(1000);
    }
  }
}
if(sttt1<30){
  for(sttt1=0;sttt1<30;sttt1++){
    if(sttt1==2){
      moveDegree2(2,150);
      delay(1000);
    }
  }
}
if(sttt2<30){
  for(sttt2=0;sttt2<30;sttt2++){
    if(sttt2==2){
      moveDegree6(6,112);
      delay(1000);
    }
  }
}
}
if((left1>511)&&(left2>511)&&(right1
    >511)&&(right2>511))
{
m1++;
if(m1==1)
{
  m2++;
  if(m2==1)
```

```
    {
     right();
      delay(1000);
    }
    if(m2==2)
    {
      stopp();
    }
  }
}
if((left1>511)&&(left2>511)&&(right1
      <511)&&(right2<511))
{
 stopp();
 left();
}
      if((left1>511)&&(left2<511)&&(ri
      ght1<511)&&(right2<511))
{
 stopp();
 left();
}
      if((left1<511)&&(left2>511)&&(ri
      ght1<511)&&(right2<511))
{
 stopp();
 left();
}
      if(((left1<511)&&(left2<511)&&(ri
      ght1>511)&&(right2>511))
{
 stopp();
 right();
}
      if(((left1<511)&&(left2<511)&&(ri
      ght1<511)&&(right2>511))
{
 stopp();
 right();
}
      if(((left1<511)&&(left2<511)&&(ri
      ght1>511)&&(right2<511))
{
 stopp();
 right();
}
}
```

```
void moveDegree0(int
    servonum0,int degree){
  uint16_t pulselen = map(degree,
    0, 180, SERVOMIN, SERVOMAX);
  pwm.setPWM(servonum0, 0,
    pulselen);
}
void moveDegree1(int
    servonum1,int degree){
  uint16_t pulselen = map(degree,
    0, 180, SERVOMIN, SERVOMAX);
  pwm.setPWM(servonum1, 0,
    pulselen);
}
void moveDegree2(int
    servonum2,int degree){
  uint16_t pulselen = map(degree,
    0, 180, SERVOMIN, SERVOMAX);
  pwm.setPWM(servonum2, 0,
    pulselen);
}
void moveDegree3(int
    servonum3,int degree){
  uint16_t pulselen = map(degree,
    0, 180, SERVOMIN, SERVOMAX);
  pwm.setPWM(servonum3, 0,
    pulselen);
}
void moveDegree4(int
    servonum4,int degree){
  uint16_t pulselen = map(degree,
    0, 180, SERVOMIN, SERVOMAX);
  pwm.setPWM(servonum4, 0,
    pulselen);
}
void moveDegree5(int
    servonum5,int degree){
  uint16_t pulselen = map(degree,
    0, 180, SERVOMIN, SERVOMAX);
  pwm.setPWM(servonum5, 0,
    pulselen);
}
void moveDegree6(int
    servonum6,int degree){
  uint16_t pulselen = map(degree,
    0, 180, SERVOMIN, SERVOMAX);
  pwm.setPWM(servonum6, 0,
    pulselen);
}
void color() {
  digitalWrite(nLED,1);
  digitalWrite(s2, LOW);
  digitalWrite(s3, LOW);
```

```
  // count OUT, pRed, RED
  red = pulseIn(out, digitalRead(out)
      == HIGH ? LOW : HIGH);
  digitalWrite(s3, HIGH);
  //count OUT, pBLUE, BLUE
  blue = pulseIn(out,
      digitalRead(out) == HIGH ? LOW
      : HIGH);
  digitalWrite(s2, HIGH);
  // count OUT, pGreen, GREEN
  green = pulseIn(out,
      digitalRead(out) == HIGH ? LOW
      : HIGH);
  digitalWrite(nLED,0);
}
void font(){
  analogWrite(a1,50);
  analogWrite(a2,0);
  analogWrite(b1,45);
  analogWrite(b2,0);
}
void back(){
  analogWrite(a1,0);
  analogWrite(a2,60);
  analogWrite(b1,0);
  analogWrite(b2,60);
}
void left(){
  analogWrite(a1,0);
  analogWrite(a2,0);
  analogWrite(b1,45);
  analogWrite(b2,0);
}
void right(){
  analogWrite(a1,50);
  analogWrite(a2,0);
  analogWrite(b1,0);
  analogWrite(b2,0);
}
void stopp(){
  analogWrite(a1,0);
  analogWrite(a2,0);
  analogWrite(b1,0);
  analogWrite(b2,0);
}
```