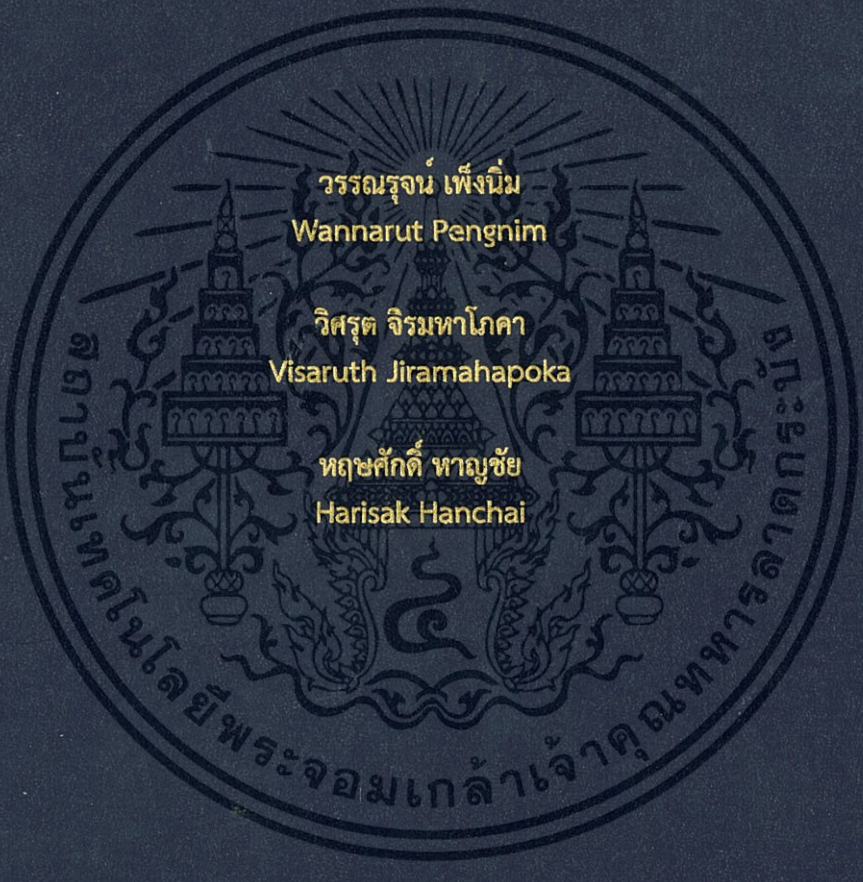
เครื่องจ่ายยาอัตโนมัติ
Automatic Pharmacy Dispensing Systems

วรรณรุจน์ เพ็งนิ่ม
Wannarut Pengnim

วิศรุต จิรมหาโภคา
Visaruth Jiramahapoka

หฤษศักดิ์ หาญชัย
Harisak Hanchai



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.2560

เครื่องจ่ายยาอัตโนมัติ

Automatic Pharmacy Dispensing Systems

โดย

วรรณรุจน์ เพ็งนิ่ม

วิศรุต จิรมหาโภคา

หฤษศักดิ์ หาญชัย

อาจารย์ที่ปรึกษา

อ.ชินภัทร นันทจิวากรชัย

ปริญญานิพนธ์เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2560

ปริญญานิพนธ์ ปีการศึกษา 2560

ภาควิชา วิศวกรรมอิเล็กทรอนิกส์

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องจ่ายยาอัตโนมัติ

Automatic Pharmacy Dispensing Systems

ผู้จัดทำ นายวรรณรุจน์ เพ็งนิ่ม รหัสประจำตัว 57011110

นายวิศรุต จิรมหาโภคา รหัสประจำตัว 57011186

นายหฤษศักดิ์ หาญชัย รหัสประจำตัว 57011445

ปริญญานิพนธ์นี้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

(อ.ชินภัทร นันทจิวากรชัย)

อาจารย์ที่ปรึกษา

หัวขอโครงงาน	เครื่องจ่ายยาอัตโนมัติ
นักศึกษา	นาย วรรณรุจน์ เพ็งนิ่ม รหัสนักศึกษา 57011110
	นาย วิศรุต จิรมหาโภคา รหัสนักศึกษา 57011186
	นาย หฤษศักดิ์ หาญชัย รหัสนักศึกษา 57011445
ปริญญา	วิศวกรรมศาสตรบัณฑิต
ภาควิชา	วิศวกรรมอิเล็กทรอนิกส
ปการศึกษา	2560
อาจารยที่ปรึกษาโครงงาน	อ.ชินภัทร นันทจิวากรชัย

บทคัดย่อ

ปริญญานิพนธ์นี้คือการศึกษาและออกแบบเครื่องจ่ายยาอัตโนมัติ เพื่อช่วยอำนวยความสะดวกแก่แพทย์และเภสัชกรในการให้บริการแก่ผู้ป่วย โดยเครื่องนี้ทำหน้าที่เหมือนคลังเก็บยาที่สามารถจ่ายยาออกมาได้โดยรับคำสั่งจากคอมพิวเตอร์ส่งผ่านข้อมูลไปยังบอร์ดไมโครคอนโทรลเลอร์ ทำหน้าที่ควบคุมการทำงานของอุปกรณ์ต่างๆ ซึ่งจะแบ่งเป็นหลายช่องแยกออกไปตามชนิดของยา เครื่องจ่ายยาอัตโนมัติเครื่องนี้ใช้ขดลวดโซลินอยเป็นตัวดีดกล่องยาออกมาจากรางข้ามแผงกั้น โดยมีเซ็นเซอร์ 2 ตัว ตรวจว่ากล่องยาหมดแล้วหรือยังและกล่องยาข้ามผ่านแผงกั้นออกไปแล้วหรือไม่ ในการควบคุมระบบนั้นเราใช้ ไมโครคอนโทรลเลอร์ ในการประมวลผล และใช้วิธียูอาร์ทในการสื่อสารระหว่างบอร์ดไมโครคอนโทรเลอร์เพื่อขยายช่องเอาท์พุทซึ่งสามารถขยายออกได้มากขึ้น

Project title	Automatic Pharmacy Dispensing Systems	
Students	Mr. Wannarut Pengnim	Student ID 57011110
	Mr. Visaruth Jiramahapoka	Student ID 57011186
	Mr. Harisak Hanchai	Student ID 57011445
Degree	Bachelor of Engineering	
Program	Electronics Engineering	
Academic Year	2017	
Project Advisor	Professor Chinnapat Nantajivagonchai	

ABSTRACT

This thesis is a development of the Automatic Pharmacy Dispensing Systems. This systems are storing medicine boxes and dispensing the boxes with solenoid coils. We use Arduino boards as microcontroller connect each other by UART protocol. In our systems we have infrared sensor to check on the boxes to make sure the solenoid push it away and another sensor to check on our systems when the boxes have run out.

กิตติกรรมประกาศ

การทำโครงงานครั้งนี้สำเร็จลุล่วงไปด้วยความอนุเคราะห์จากบุคลากรหลายท่านที่ไม่อาจกล่าวถึงได้ทั้งหมด ซึ่งท่านแรกขอขอบคุณ ผู้ช่วยศาสตราจารย์ อ.ชินภัทร นันทจิวากรชัย อาจารย์ที่ปรึกษา ที่ค่อยช่วยเหลือและให้คำแนะนำในการทำโครงงาน และ ท่านต่อไป ผศ.ประภากร สุวรรณะ ที่เป็นผู้ดูแลเกี่ยวกับโครงงานของนักศึกษาชั้นปีที่ 4 ขอขอบพระคุณพ่อและแม่ของคณะผู้จัดทำ ที่คอยเป็นกำลังใจหลักและเป็นผู้สนับสนุนเงินทุนหลักในการทำงาน รวมไปถึงเพื่อนๆผู้เป็นกำลังหลักคอยช่วยเหลือให้คำแนะนำรวมทั้งช่วยกันในการแก้ปัญหาที่เกิดขึ้นในระหว่างการทำโครงงานนี้ทั้งหมดทำให้ผลของโครงงานสำเร็จลุล่วงไปด้วยดี

สุดท้ายนี้คณะผู้จัดทำหวังว่าโครงงานนี้จะเป็นประโยชน์สำหรับผู้สนใจและผู้นำผลงานนี้ไปใช้ได้

วรรรุจณ์ เพ็งนิ่ม
วิศรุต จิรมหาโภคา
หฤษศักดิ์ หาญชัย

สารบัญ

หน้า

บทคัดย่อภาษาไทย.....I
บทคัดย่อภาษาอังกฤษ.....II
กิตติกรรมประกาศ.....III
สารบัญ.....IV
สารบัญรูป.....VI
สารบัญตาราง.....VII
บทที่ 1 บทนำ.....1
1.1 ความเป็นมาและความสำคัญ.....1
1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา.....1
1.3 สมมุติฐานของการศึกษา.....1
1.4 ขอบเขตการวิจัยและขั้นตอนการศึกษา.....1
1.5 ประโยชน์ที่คาดว่าจะได้รับจากโครงงาน.....2
บทที่ 2 ทฤษฎีและหลักการ.....3
2.1 หลักการทำงานของเครื่องจ่ายยาอัตโนมัติ.....3
2.2 โครงสร้างพื้นฐานของเครื่องจ่ายยาอัตโนมัติ.....3
2.2.1 ขดลวดโซลินอยด์.....3
2.2.2 เซ็นเซอร์แสง.....4
2.2.2.1 ตัวต้านทานชนิดไวต่อแสง (LDR).....4
2.2.2.2 โฟโต้ไอโอด (Photo Diode).....4
2.2.2.3 โฟโต้ทรานซิสเตอร์ (Photo Transistors).....5
2.2.2.4 อินฟราเรดเซนเซอร์ (IR Sensor).....5
2.2.2.5 เซนเซอร์ชนิดใช้แสง (Reflective Optical Sensor).....6
2.2.3 อาร์ดูโน่เมก้า (Arduino Mega).....6
2.2.4 บลัซเซอร์ (Buzzer).....7
2.2.5 ไดโอดเปล่งแสง (LED).....8
2.3 รูปแบบการสื่อสารระหว่างไมโครคอนโทรลเลอร์.....8
2.3.1 การสื่อสารแบบอนุกรม.....8
2.3.1.1 แบบซิงโคนัส (Synchronous).....8
2.3.1.2 แบบอะซิงโคนัส (Asynchronous).....8
2.3.2 การสื่อสารผ่านเอสพีไอ (SPI).....9
2.3.3 การสื่อสารผ่านยูอาร์ท (UART).....9

สารบัญ (ต่อ)

หน้า

2.4 ทรานซิสเตอร์ทำงานเป็นสวิตส์ 11

บทที่ 3 วิธีการดำเนินการ 12

3.1 การเลือกใช้ไมโครคอนโทรลเลอร์ 12

3.2 การเลือกใช้ขดลวดโซลินอยด์ 12

3.3 การเลือกใช้เซนเซอร์ 12

3.4 การเขียนโค้ดให้กับระบบ 12

3.5 ผังงาน (Flow chart) 13

3.6 วงจรขับลวดโซลินอยด์ 14

3.7 การเลือกใช้การสื่อสารระหว่างบอร์ดไมโครคอนโทรเลอร์ 15

3.8 การเลือกใช้พาวเวอร์ซัพพลาย 15

3.9 การออกแบบฮาร์ดแวร์ 16

บทที่ 4 ผลการทดลอง 18

4.1 การทดสอบขดลวดโซลินอยด์ 18

4.2 การทดสอบเซนเซอร์อินฟราเรด 20

4.3 การทดสอบการทดงานของระบบ 22

บทที่ 5 สรุปผลการทดลอง 23

5.1 สรุปผลการทดลอง 23

5.2 ปัญหา 23

5.3 ข้อเสนอแนะ 23

เอกสารอ้างอิง 24

ภาคผนวก 25

สารบัญรูป

รูปที่ หน้า

รูปที่ 1 การไหลของกระแสไฟฟ้าในขดลวดโซลินอยด์....3

รูปที่ 2 ตัวต้านทานชนิดไวต่อแสง....4

รูปที่ 3 โฟโต้ไอโอด....4

รูปที่ 4 โฟโต้ทรานซิสเตอร์....5

รูปที่ 5 อินฟราเรดเซนเซอร์....5

รูปที่ 6 เซนเซอร์ชนิดใช้แสง....6

รูปที่ 7 อาร์ดูโน่เมก้า (Arduino Mega)....7

รูปที่ 8 บัซเซอร์ (Buzzer)....7

รูปที่ 9 ไดโอดเปล่งแสง (LED)....8

รูปที่ 10 การเชื่อมต่อการสื่อสารแบบ SPI ระหว่างอุปกรณ์ Master – Slave หลายตัว....10

รูปที่ 11 การเชื่อมต่อแบบ UART....10

รูปที่ 12 ผังงาน (Flow chart)....13

รูปที่ 13 schematic ของวงจรขับลวดโซลินอยด์....14

รูปที่ 14 PCB ของวงจรขับลวดโซลินอยด์....15

รูปที่ 15 ภาพด้านหน้าของเครื่องจ่ายยา....16

รูปที่ 16 ภาพด้านในของเครื่องจ่ายยา....16

รูปที่ 17 ภาพด้านข้างของเครื่องจ่ายยา....17

รูปที่ 18 ภาพด้านหลังของเครื่องจ่ายยา....17

รูปที่ 19 ช่วงที่ขดลวดโซลินอยด์ทำงาน....18

รูปที่ 20 ช่วงที่ขดลวดโซลินอยด์ไม่ทำงาน....19

รูปที่ 21 การทำงานของเซนเซอร์ตอนที่ตรวจไม่เจอสิ่งของ....20

รูปที่ 22 การทำงานของเซนเซอร์ตอนที่ตรวจเจอสิ่งของ....21

สารบัญตาราง

ตารางที่ หน้า

ตารางที่ 1 การทดสอบขดลวดโซลินอยด์.....18

ตารางที่ 2 การทดสอบเซนเซอร์อินฟราเรด.....20

ตารางที่ 3 การทดสอบการทำงานของระบบ.....22

บทที่ 1
บทนำ

1.1ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันการใช้ระบบอัตโนมัติได้เข้ามามีบทบาทในการทำงานขององกรณ์ต่างๆมากขึ้น เพื่อช่วยอำนวยความสะดวก ประหยัดเวลา และยังสามารถลดต้นทุนได้อีกด้วย ทำให้เป็นที่นิยมในอุตสาหกรรมไปจนถึงการใช้งานในครัวเรือน จากประโยชน์ที่กล่าวมาข้างต้น เครื่องจ่ายยาอัตโนมัติเป็นหนึ่งในระบบอัตโนมัติที่น่าสนใจ แต่ยังไม่เป็นที่นิยมในประเทศไทย และหากต้องการที่จะใช้เครื่องจ่ายยาอัตโนมัตินี้เราต้อนำเข้าจากต่างประเทศ จึงเห็นว่าสร้างเครื่องจ่ายยาอัตโนมัตินี้ได้ จะสามารถเพิ่มประสิทธิภาพในการทำงานของโรงพยาบาลได้มากขึ้น

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

1.2.1.ศึกษาการทำงานของเครื่องจ่ายยาอัตโนมัติ
1.2.2.สร้างตัวอย่างเครื่องจ่ายยาอัตโนมัติ
1.2.3.ศึกษาการสื่อสารกันระหว่างไมโครคอนโทรเลอร์

1.3 สมมุติฐานของการศึกษา

1.3.1.สามารถเข้าใจวิธีการทำงานของเครื่องจ่ายยาอัตโนมัติได้
1.3.2.สามารถใช้บอร์ดArduinoในการควบคุมระบบได้
1.3.3.สามารถนำความรู้ในห้องเรียนมาใช้งานได้จริง
1.3.4.สามารถทำงานร่วมกับผู้อื่นได้

1.4 ขอบเขตการวิจัย

ทำการศึกษาเครื่องจ่ายยาอัตโนมัติที่มีคุณสมบัติดังนี้
1.4.1.สามารถดีดกล่องยาออกจากรางได้
1.4.2.สามารถบอกได้ว่ากล่องยาที่อยู่ในรางหมดแล้วหรือยัง
1.4.3.สามารถใช้คำสั่งจากคอมพิวเตอร์ผ่านบอร์ด Arduino ได้

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1.ได้รับความรู้เกี่ยวกับการทำงานของเครื่องจ่ายยาอัตโนมัติได้

1.5.2.ได้เรียนรู้ในการทำงานร่วมกับผู้อื่น

1.5.3.ได้เรียนรู้การใช้บอร์ด Arduino ในการควบคุมวงจรได้

1.5.4.ได้รับประสบการณ์ในการทำงาน เพื่อที่จะสามารถแก้ไขปัญหาได้อย่างมีประสิทธิภาพ

บทที่ 2

หลักการและทฤษฎี

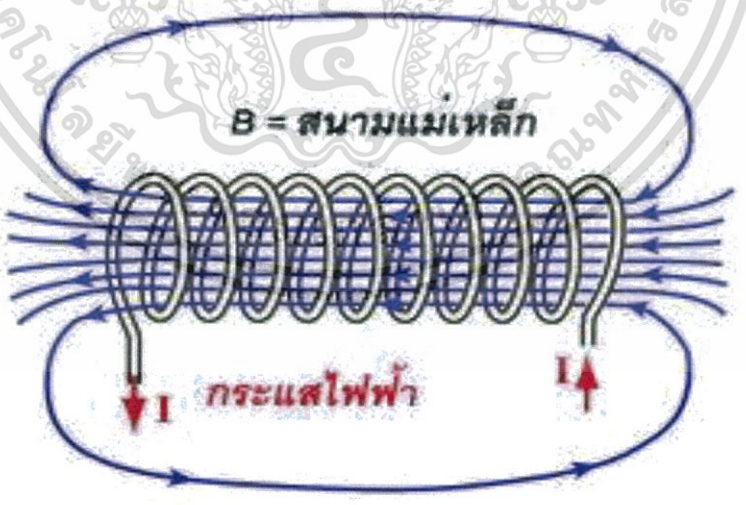
2.1 หลักการทำงานของเครื่องจ่ายยาอัตโนมัติ

เป็นเครื่องจักรที่สามารถจ่ายยาออกมาให้แก่ผู้บริโภค โดยคอมพิวเตอร์เป็นตัวออกคำสั่งผ่านบอร์ดวงจรอาร์ดูโน่ ซึ่งจะใช้ขดลวดโซลินอยด์เป็นตัวดีดกล่องยาผ่านแผงกั้น ออกมา และมีเซนเซอร์ในการบอกว่ากล่องยาหมดแล้วหรือไม่ และเราสามารถขยายจำนวนช่องได้โดยใช้วิธียูอาร์ทในการสื่อสารระหว่าง ไมโครคอนโทรลเลอร์หลายๆบอร์ดได้

2.2 โครงสร้างพื้นฐานของเครื่องจ่ายยาอัตโนมัติ

2.2.1 ขดลวดโซลินอยด์

เมื่อให้กระแสไฟฟ้าขนาดคงที่ไหลผ่านขดลวดโซลินอยด์ จะเกิดสนามแม่เหล็กรอบขดลวดโซลินอยด์ สนามแม่เหล็กบริเวณแกนกลางจะมีค่าเพิ่มขึ้นถ้าจำนวนรอบเพิ่มขึ้น ถ้าใส่แกนเหล็กอ่อนไว้ในขดลวดเมื่อมีกระแสไฟฟ้าผ่าน แกนเหล็กอ่อนจะกลายเป็นแม่เหล็ก แม่เหล็กที่เกิดขึ้นนี้ เรียกว่า แม่เหล็กไฟฟ้า ซึ่งค่าของสนามแม่เหล็กไฟฟ้า จะขึ้นอยู่กับค่าของ กระแสไฟฟ้าที่ไหลผ่าน และ จำนวนรอบของขดลวด ขดลวดโซลินอยด์ หมายถึง ขดลวดตัวนำที่มีฉนวนหุ้ม นำมาพันเป็นขดลวดวงกลมหลายรอบเรียงซ้อนกันเป็นทรงก ระบอกโดยมีรัศมีคงที่



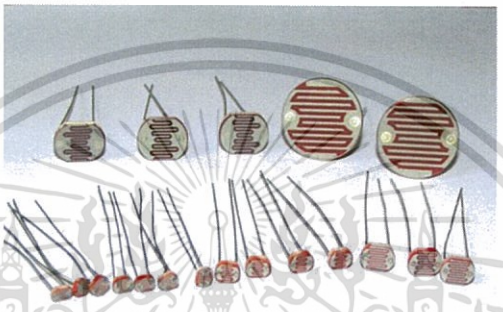
รูปที่ 1 การไหลของกระแสไฟฟ้าในขดลวดโซลินอยด์

2.2.2 เซนเซอร์แสง(Optical Sensor)

คืออุปกรณ์อิเล็กทรอนิกส์ที่เปลี่ยนแปลงค่าความต้านทาน หรือการนำไฟฟ้า ที่ไหลผ่านตัวมันได้เมื่อมีแสงมาตกกระทบมีหลายชนิดดังนี้

2.2.2.1 ตัวต้านทานชนิดไวต่อแสง (LDR)

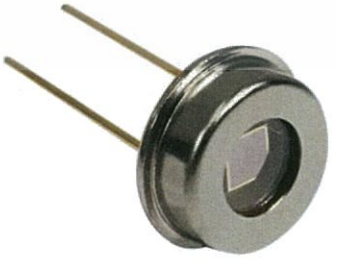
ตัวต้านทานแปรค่าตามแสง หรือ LDR (ย่อมาจาก Light Dependent Resistor) คืออุปกรณ์อิเล็กทรอนิกส์ที่ใช้ตรวจจับแสง โดยหามีแสงมาตกกระทบน้อยจะทำให้มีความต้านทานมากและหากมีแสงมาตกกระทบมากความต้านทานจะน้อยลง



รูปที่ 2 ตัวต้านทานชนิดไวต่อแสง

2.2.2.2 โฟโต้ไอโอด (Photo Diode)

โฟโต้ไอโอด (ภาษาอังกฤษ Photo Diode) จะถูกแบ่งออกเป็นอีก 2 ชนิด คือ 1.ตอบสนองต่อแสงที่สามารถมองเห็นได้ 2.ตอบสนองต่อแสงความถี่สูง
ย่านอินฟาเรดหลักการทำงานคือ เมื่อมีแสงมีตกกระทบมาก จะทำให้สามารถนำกระแสได้มาก หากมีแสงมาตกกระทบน้อยจนถึงจุดจุดหนึ่ง จะไม่นำกระแสเลย การนำโฟโต้ไดโอดไปใช้งานจะต้องต่อในลักษณะไบอัสกลับจึงจะสามารถใช้งานได้



รูปที่ 3 โฟโต้ไอโอด

2.2.2.3 โฟโต้ทรานซิสเตอร์ (Photo Transistors)

โฟโต้ทรานซิสเตอร์(ภาษาอังกฤษ Photo Transistors) แบ่งเป็น 2 ชนิดเช่นเดียวกับโฟโต้ไดโอด ลักษณะภายนอกคล้ายๆกับโฟโต้ไดโอด การใช้งานก็เช่นเดียวกันจำเป็นต้องดูDatasheetประกอบการใช้งาน

รูปที่ 4 โฟโต้ทรานซิสเตอร์

2.2.2.4 อินฟราเรดเซนเซอร์ (IR Sensor)

คืออุปกรณ์ที่นำโฟโต้ไดโอด หรือโฟโต้ทรานซิสเตอร์ มารวมเข้ากับวงจรควบคุมภายในเพื่อใช้สำหรับความถี่สูงโดยเฉพาะอินฟราเรดเซนเซอร์นั้นจะตอบสนองกับแสงอินฟาเรดเท่านั้นใช้งานร่วมกับไดโอดเปล่งแสง อินฟาเรดนิยมใช้ส่งข้อมูลที่อยู่ในระยะไกล เครื่องใช้ไฟฟ้าที่ใช้งานอินฟราเรดเซนเซอร์ก็จำพวก โทรทัศน์ เครื่องเล่น ดีวีดี หรือวิทยุในรถยนต์ กล่องรับ ดาวเทียม เป็นต้น

รูปที่ 5 อินฟราเรดเซนเซอร์

2.2.2.5 เซนเซอร์ชนิดใช้แสง (Reflective Optical Sensor)
คืออุปกรณ์ที่นำโฟโต้ทรานซิสเตอร์ หรือโฟโต้ไดโอด มารวมกับไดโอดเปล่งแสงอินฟาเรต เพื่อใช้ในการตรวจจับการสะท้อนแสง หรือระยะความใกล้ของวัตถุ หลักการคือเมื่อมีแสงไปตกกระทบกับวัตถุใดๆ วัตถุนั้นจะสะท้อนแสงกับมาที่โฟโต้ไดโอด หรือโฟโต้ทรานซิสเตอร์ ตัวอย่างที่นำไปใช้งานจริงก็เช่น หุ่นยนต์วิ่งตามเส้น

รูปที่ 6 เซนเซอร์ชนิดใช้แสง

2.2.3 อาร์ดูโน่เมก้า (Arduino Mega)

เป็นบอร์ดไมโครคอนโทรลเลอร์ตระกูล AVR ที่มีการพัฒนาแบบ Open Source คือมีการเปิดเผยข้อมูลทั้งด้าน Hardware และ Software ตัว บอร์ด Arduino ถูกออกแบบมาให้ใช้งานได้ง่าย ดังนั้นจึงเหมาะสำหรับผู้เริ่มต้นศึกษา ทั้งนี้ผู้ใช้งานยังสามารถดัดแปลงเพิ่มเติมพัฒนาต่อยอดทั้งตัวบอร์ดหรือโปรแกรมต่อได้อีกด้วย

ความง่ายของบอร์ด Arduino ในการต่ออุปกรณ์เสริมต่างๆ คือผู้ใช้งานสามารถต่อวงจรอิเล็กทรอนิคส์จากภายนอกแล้วเชื่อมต่อเข้ามาที่ขา I/O ของบอร์ด (ดูตัวอย่างรูปที่ 1) หรือเพื่อความสะดวกสามารถเลือกต่อกับบอร์ดเสริม (Arduino Shield) ประเภทต่างๆ (ดูตัวอย่างรูปที่ 2) เช่น Arduino XBee Shield, Arduino Music Shield, Arduino Relay Shield, Arduino Wireless Shield, Arduino GPRS Shield เป็นต้น มาเสียบกับบอร์ดบนบอร์ด Arduino แล้วเขียนโปรแกรมพัฒนาต่อได้เลย

Arduino mega 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

รูปที่ 7 อาร์ดูโน่เมก้า (Arduino Mega)

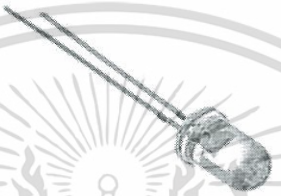
2.2.4 บัซเซอร์ (Buzzer)

เป็นอุปกรณ์ไฟฟ้าที่นำผลของแม่เหล็กไฟฟ้ามาดึงดูดให้แกนอามาเจอร์ (Armature) เคลื่อนที่มาเคาะกับกระดิ่ง (Bell) ทำให้เกิดเสียงดังได้ โครงสร้างภายในประกอบด้วยแท่งเหล็กรูปตัวยู (U-Shaped) พันขดลวดรอบๆ แท่งเหล็กนี้ต่ออนุกรมกับหน้าสัมผัสซึ่งเปิดปิดได้โดยการเคลื่อนที่ของก้านอามาเจอร์การใช้งานต้องต่อกระดิ่งไฟฟ้าอนุกรมกับสวิทช์กดปุ่ม (Push Button) และแหล่งจ่ายไฟฟ้า เช่น แบตเตอรี่ เมื่อกดสวิตช์กระแสไฟฟ้าจะผ่านหน้าสัมผัสและขดลวด ทำให้เกิดการดึงดูดอามาเจอร์ให้เคลื่อนที่มาเคาะกระดิ่งทำให้เกิดเสียงดัง ในขณะที่อามาเจอร์เคลื่อนที่ก็จะตัดวงจรไฟฟ้าออกไปด้วย ดังนั้นเมื่อก้านอามาเจอร์เคาะกระดิ่ง แล้วก็จะดีดไปตำแหน่งเดิมทันที และต่อวงจรไฟฟ้าอีกครั้ง เมื่อใดที่ปล่อยมือจากสวิตช์กระบวนการที่เกิดขึ้นก็จะหยุดลง

รูปที่ 8 บัซเซอร์ (Buzzer)

2.2.5 ไดโอดเปล่งแสง (LED)

หลอด LED หรือไดโอดเปล่งแสง โครงสร้างประกอบไปด้วยสารกึ่งตัวนำสองชนิด (สารกึ่งตัวนำชนิด N และสารกึ่งตัวนำชนิด P) ประกบเข้าด้วยกัน มีผิวข้างหนึ่งเรียบคล้ายกระจกเมื่อจ่ายไฟฟ้ากระแสตรงผ่านตัว LED โดยจ่ายไฟบวกให้ขาแอโนต (A) จ่ายไฟลบให้ขาแคโทด (K) ทำให้อิเล็กตรอนที่สารกึ่งตัวนำชนิด N มีพลังงานสูงขึ้น จนสามารถวิ่งข้ามรอยต่อจากสารชนิด N ไปรวมกับโฮลในสารชนิด P การที่อิเล็กตรอนเคลื่อนที่ผ่านรอยต่อ PN ทำให้เกิดกระแสไหล เป็นผลให้ระดับพลังงานของอิเล็กตรอนเปลี่ยนไปและคายพลังงานออกมาในรูปคลื่นแสง



รูปที่ 9 ไดโอดเปล่งแสง (LED)

2.3 รูปแบบการสื่อสารระหว่างไมโครคอนโทรเลอร์

2.3.1 การสื่อสารแบบอนุกรม

คือการใช้สาย 1 เส้นรับ-ส่งข้อมูลแบบต่อเนื่อง โดยอาศัยเทคนิคต่าง ๆ ในการสื่อสาร เช่นการใช้สัญญาณทริกเพื่อรับข้อมูลเข้า การใช้บิตเริ่มต้นกำหนดการรับข้อมูล โดยอาจจะอาศัยและไม่อาศัยเวลาในการทำงาน ทั้งนี้การสื่อสารแบบอนุกรมสามารถแบ่งได้เป็น 2 รูปแบบคือ

2.3.1.1 แบบซิงโคนัส (Synchronous) – เป็นการสื่อสารที่ใช้สายสัญญาณข้อมูลอย่างน้อย 1 เส้น และมีสายอีก 1 เส้นกำหนดจังหวะการรับข้อมูล ข้อดีของการสื่อสารแบบนี้คือการรับส่งข้อมูลมีความผิดพลาดน้อยหรือไม่มีความผิดพลาดเลย แต่ข้อเสียคือต้องใช้สายสัญญาณอย่างน้อย 2 เส้นในการสื่อสาร โดยโปรโตคอลที่ทำงานแบบซิงโคนัสได้แก่ I^2C I^2S และ SPI

2.3.1.2 แบบอะซิงโคนัส (Asynchronous) – เป็นการสื่อสารที่ใช้สายสัญญาณข้อมูลเพียงเส้นเดียวในการทำงาน โดยอาศัยสัญญาณจากบิตเริ่มต้น และบิตสิ้นสุดในการบอกจังหวะการรับส่งข้อมูล การสื่อสารแบบนี้จำเป็นต้องอาศัยเวลามาเป็นตัวกำหนดการรับสัญญาณเข้ามาซึ่งหากมีการตั้งค่าที่ผิดจะทำให้อ่านข้อมูลที่ส่งมาได้ผิดพลาด ข้อดีของการสื่อสารแบบนี้คือใช้สายสัญญาณเพียง 1 เส้นก็สามารถรับ-ส่งข้อมูลได้แล้ว แต่ข้อเสียคือมีความผิดพลาดในการสื่อสารได้ง่าย โดยโปรโตคอลที่ทำงานแบบอะซิงโคนัสคือ UART

2.3.2 การสื่อสารผ่านเอสพีไอ (SPI)

SPI หรือ Serial Peripheral Interface เป็นวิธีการสื่อสารอนุกรมแบบ Synchronous อีกรูปแบบหนึ่ง ซึ่งทำงานในรูปแบบที่ให้อุปกรณ์ตัวหนึ่งทำหน้าที่เป็น Master ในขณะที่อีกตัวหนึ่งทำหน้าที่เป็น Slave และสามารถส่งข้อมูลในโหมด Full-duplex นั่นหมายความว่า สัญญาณสามารถส่งหากันได้ระหว่าง Master และ Slave ได้อย่างต่อเนื่อง

SCLK (Serial Clock) ใช้ส่งสัญญาณนาฬิกาจากอุปกรณ์ Master ไปยังอุปกรณ์ Slave เพื่อกำหนดจังหวะการรับส่งข้อมูล

MOSI (Master Out Slave In) ใช้ส่งข้อมูลจากอุปกรณ์ Master ไปยังอุปกรณ์ Slave

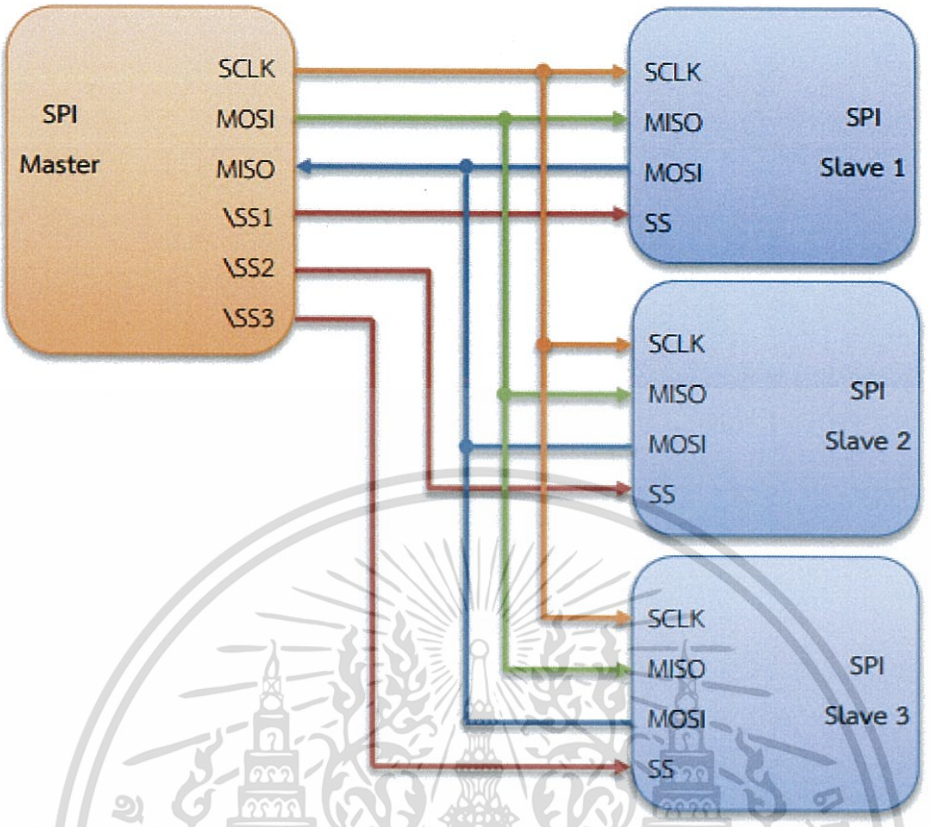
MISO (Master In Slave Out) ใช้รับข้อมูลจากอุปกรณ์ Slave

\SS (Slave Select) หรือ ขา CS (Chip Select) ใช้ส่งสัญญาณ Low ไปยังอุปกรณ์ Slave ที่ต้องการรับส่งข้อมูล

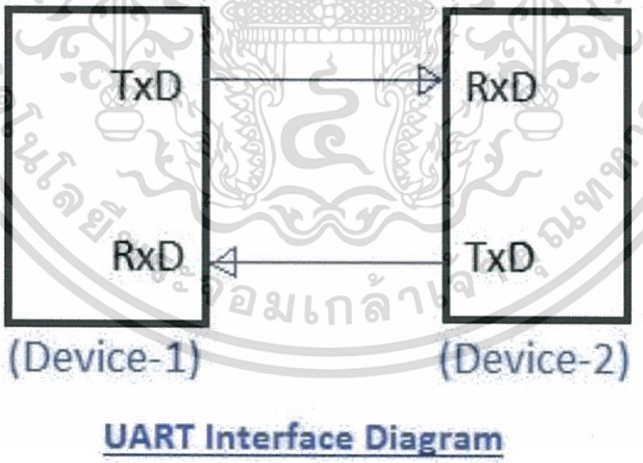
อุปกรณ์ Master ทำหน้าที่เป็นตัวควบคุมการสื่อสารทั้งหมด โดยควบคุมการสื่อสารตามสัญญาณนาฬิกาและสายสัญญาณ SS ตัวมาสเตอร์จะเป็นตัวที่ตัดสินใจเลือก รับ หรือ ส่งข้อมูลไปยังอุปกรณ์ Slave สัญญาณเส้น SS หรือ Slave select ในกรณี ที่มีตัว Slave มากกว่า 1 ตัว โดยการทำให้เส้น SS มีระดับสัญญาณเป็น Low เมื่อต้องการติดต่อกับ Slave ตัวใด จากรูป 1-2 หากต้องการติดต่อสื่อสารกับอุปกรณ์ Slave ตัวใด ก็เพียงทำให้สัญญาณ SS ของ Slave ตัวนั้น มีระดับสัญญาณเป็น Low

2.3.3การสื่อสารผ่านยูอาร์ท (UART)

UART ย่อมาจาก Universal Asynchronous Receiver Transmitter ใช้การสื่อสารแบบอนุกรม โดยเป็นแบบอะซิงโคนัสใช้สายสัญญาณเพียง 2 เส้น คือ Tx และ Rx ในการรับ – ส่งข้อมูล โดยเส้น Tx จะเป็นเส้นที่ใช้ส่งข้อมูล และเส้น Rx เป็นเส้นที่ใช้รับข้อมูล เมื่อนำไปต่อใช้งานจะต้องต่อไขว้กันระหว่างอุปกรณ์และไมโครคอนโทรลเลอร์ และต้องกำหนดความเร็วในการสื่อสารซึ่งสามารถเลือกได้ดังนี้ 300 600 1200 2400 4800 9600 14400 19200 28800 38400 57600 และ 115200 ทั้งนี้ความเร็วที่กำหนดมักขึ้นอยู่กับอุปกรณ์ที่จะเชื่อมต่อด้วยโดยมักจะกำหนดเป็น 9600 เมื่ออุปกรณ์กำหนดเป็น 9600 ที่ตัวไมโครคอนโทรลเลอร์ต้องกำหนดเป็น 9600 ด้วย จึงจะสามารถรับ-ส่งข้อมูลได้ถูกต้อง



รูปที่ 10 การเชื่อมต่อการสื่อสารแบบ SPI ระหว่างอุปกรณ์ Master – Slave หลายตัว



รูปที่ 11 การเชื่อมต่อแบบ UART

2.4 ทรานซิสเตอร์ทำงานเป็นสวิตช์

ทรานซิสเตอร์ถูกใช้กันทั่วไปให้ทำหน้าที่เป็นสวิตช์อิเล็กทรอนิกส์, ทั้งสำหรับการใช้งาน พลังงานสูง เช่น switched-mode power supplies และ สำหรับการใช้งานพลังงานต่ำ เช่น ลอจิกเกต

ในวงจรทรานซิสเตอร์แบบ emitter ลงกราวด์ เป็นวงจรสวิตช์ขับขดลวดโซลินอยด์ที่ในสถานะปกติจะ OFF ขดลวดจะไม่ทำงาน เมื่อแรงดันไฟฟ้าที่ base สูงขึ้น, กระแส emitter และ collector (I_{ce}) เพิ่มขึ้นแบบ exponential จนอิ่มตัว แรงดันที่ collector จะลดลงเข้าใกล้ emitter (หรือใกล้ศูนย์) กระแส I_{ce} จะไหลผ่านโหลดเต็มที่ ซึ่งในวงจรนี้คือขดลวด ทำให้ขดลวดดีด เราจึงเรียกสถานะของสวิตช์ในขณะนี้ว่า *ON*

การให้กระแสที่ base (I_{be}) อย่างเพียงพอเป็นปัญหาที่สำคัญในการใช้ทรานซิสเตอร์ให้ทำงานเป็นสวิตช์. ทรานซิสเตอร์ให้ gain เป็นกระแส จึงได้กระแสค่อนข้างมากที่ collector ที่จะถูกสลับ โดยกระแสที่มีขนาดเล็กใน base อัตราส่วนของกระแสเหล่านี้แตกต่างกันไป ขึ้นอยู่กับชนิด ของทรานซิสเตอร์และแม้กระทั่งทรานซิสเตอร์ประเภทเดียวกันก็แตกต่างกัน ขึ้นอยู่กับกระแสใน collector ในตัวอย่างวงจรสวิทช์ จะมีตัวต้านทานที่ต้องเลือก (สมมติว่าเป็น 1K)ให้มีขนาดที่ให้กระแสที่ base มีเพียงพอ เพื่อให้แน่ใจว่าทรานซิสเตอร์จะทำงานอิ่มตัว

ในวงจรสวิตช์ใดๆ ค่าของแรงดันไฟฟ้า อินพุต จะถูกจ่ายให้มีขนาดที่จะทำให้ได้ เอาต์พุต เป็น OFF หรือ ON โดยสมบูรณ์ ทรานซิสเตอร์จึงจะทำหน้าที่เป็นสวิตช์ที่ดีและการทำงานแบบนี้ เป็นเรื่องธรรมดาใน วงจรดิจิตอลที่ต้องการเพียง “OFF” และ “ON” เท่านั้น

บทที่ 3

วิธีการดำเนินการ

ในการออกแบบเครื่องจ่ายยาอัตโนมัตินั้น ได้ออกแบบ คือ สามารถรับข้อมูลจาก โปรแกรม ที่สร้างจาก Visual studio เพื่อจ่ายยาที่ต้องการได้

3.1 การเลือกใช้ไมโครคอนโทรลเลอร์

ในโครงงานนี้ได้เลือกใช้อุปกรณ์ในส่วนของไมโครคอนโทรลเลอร์เป็น ชนิด อาร์ดูโน่เมก้าและอาร์ดูโน่ยูโน ซึ่งไมโครคอนโทรลเลอร์ นี้ มีฟังก์ชั่นและ พอร์ทอินพุทและเอาท์พุทเพียงพอต่อการใช้งานในส่วนโครงงานนี้ และมีพอร์ท interrupt เพียงพอต่อการใช้งานในโครงงานนี้ โดยใช้อาร์ดูโน่ยูโนเป็นมาสเตอร์และใช้อาร์ดูโน่เมก้าเป็นตัวเสลฟ

3.2 การเลือกใช้ขดลวดโซลินอยด์

ในโครงงานนี้เลือกใช้เป็น ขดลวดโซลินอยด์เบอร์ JF-0530B ซึ่งใช้ไฟเลี้ยง 12 โวลล์ และกินกระแส 350 มิลลิแอมแปร์ มีแรงดีดประมาณ 5 นิวตัน ซึ่งเพียงพอในการใช้ดีดกล่องยาให้ข้ามแผงกั้น

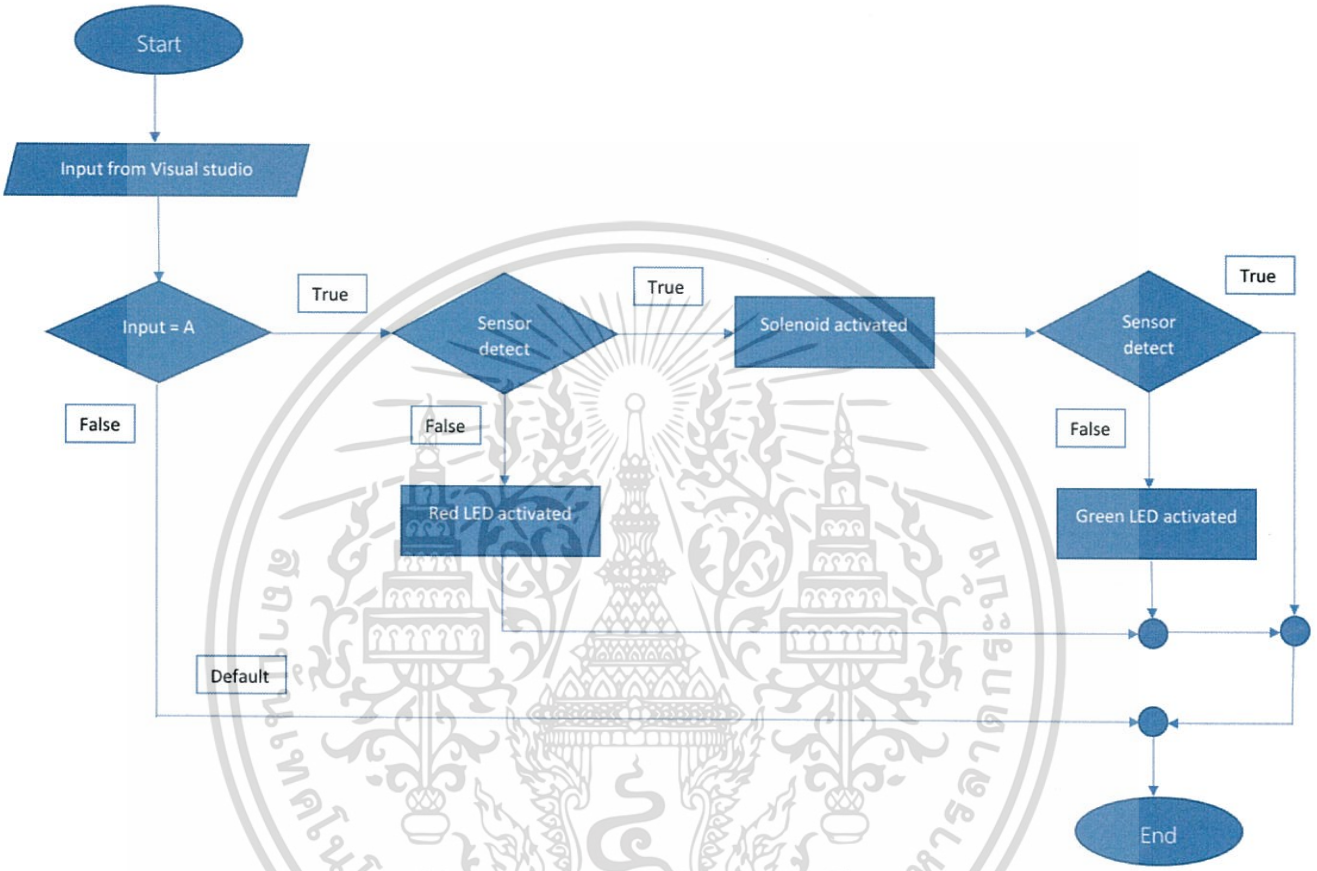
3.3 การเลือกใช้เซนเซอร์

ในโครงงานนี้เลือกใช้เป็น FC-51 ir sensor ซึ่งใช้ไฟเลี้ยง 3.3-5 โวลล์ มีระยะประมาณ 2-30 ซ.ม. ซึ่งเป็น sensor เพื่อตรวจจับตำแหน่งของกล่องยา และ ตรวจสอบจำนวนที่เหลืออยู่ของกล่องยา

3.4 การเขียนโค้ดให้กับระบบ

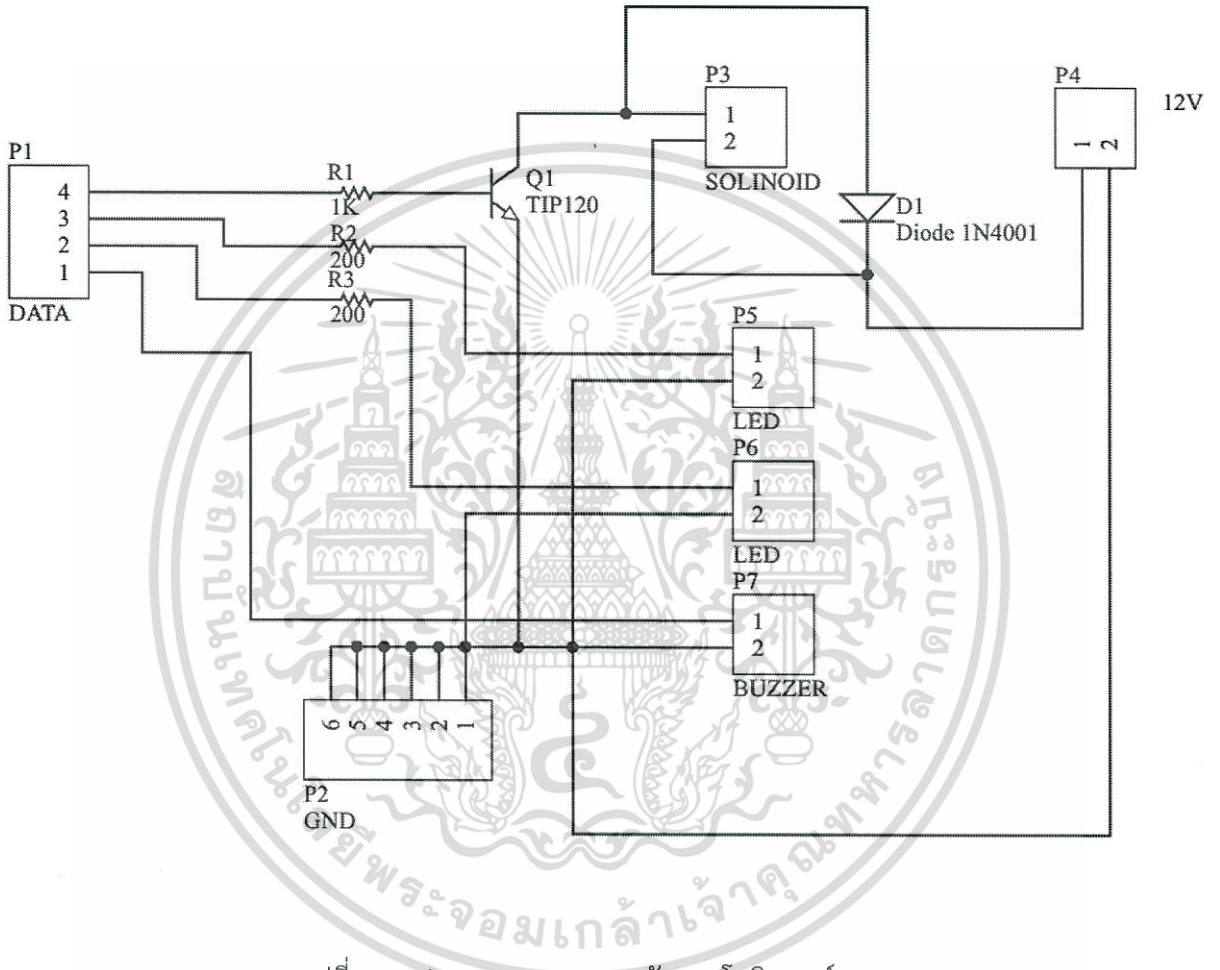
เขียนโค้ดในโปรแกรมวิชวลสตูดิโอ (Visual studio) โดยใช้ภาษาซีพลัสพลัส (C++) ในการเขียนโปรแกรมเพื่อที่จะสร้างอินเตอร์เฟสขึ้นสำหรับเลือกคำสั่ง และเขียนโค้ดการทำงานของระบบลงในอาร์ดูโน่บอร์ดโดยให้มาสเตอร์รับคำสั่งจากอินเตอร์เฟสแล้วส่งคำสั่งไปยังเสลฟ ซึ่งเสลฟจะดำเนินการตามที่เขียนโค้ดเอาไว้

3.5 ผังงาน (Flow chart)

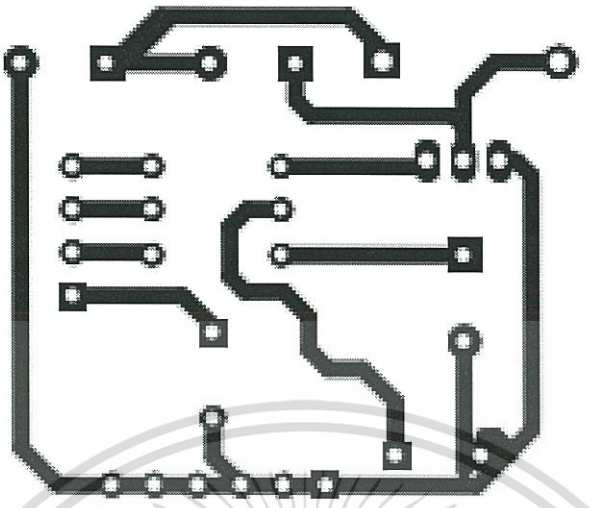


รูปที่ 12 ผังงาน (Flow chart)

3.6 วงจรขับขดลวดโซลินอยด์



รูปที่ 13 schematic ของวงจรขับลวดโซลินอยด์



รูปที่ 14 PCB ของวงจรขับลวดโซลินอยด์

3.7 การเลือกใช้การสื่อสารระหว่างบอร์ดไมโครคอนโทรเลอร์

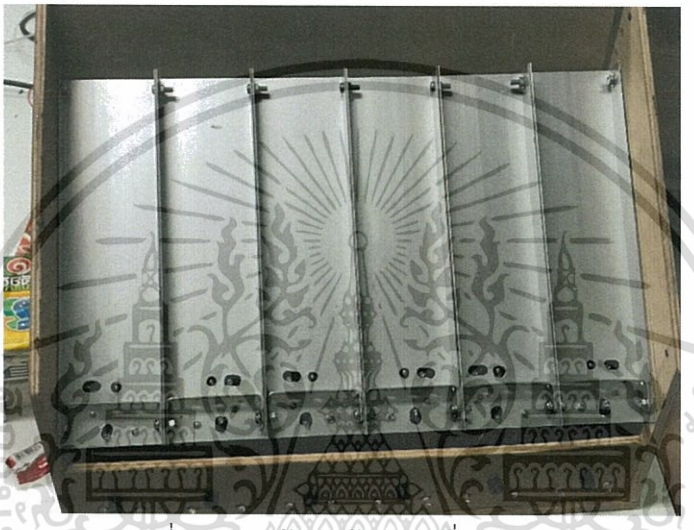
ในโครงงานนี้สามารถเลือกใช้การสื่อสารได้หลายแบบ ซึ่งในโครงงานนี้ได้เลือกใช้การสื่อสารแบบยูอาร์ท (UART) เพราะมีความซับซ้อนน้อยกว่าแบบซิงโคนัส การสื่อสารแบบอนุกรมโดยเป็นแบบอะซิงโคนัสใช้สายสัญญาณเพียง 2 เส้น คือ Tx และ Rx ในการรับและส่งข้อมูล ไม่ต้องใช้คล็อก (Clock) ในการส่งสัญญาณ และสามารถกำหนดความเร็วในการรับส่งข้อมูลได้ โดยในโครงใช้ความเร็วเท่ากับ 9600 และยังสามารถสื่อสารได้แบบฟูลดูเพล็ก (Full-duplex) ได้อีกด้วย

3.8 การเลือกใช้พาวเวอร์ซัพพลาย

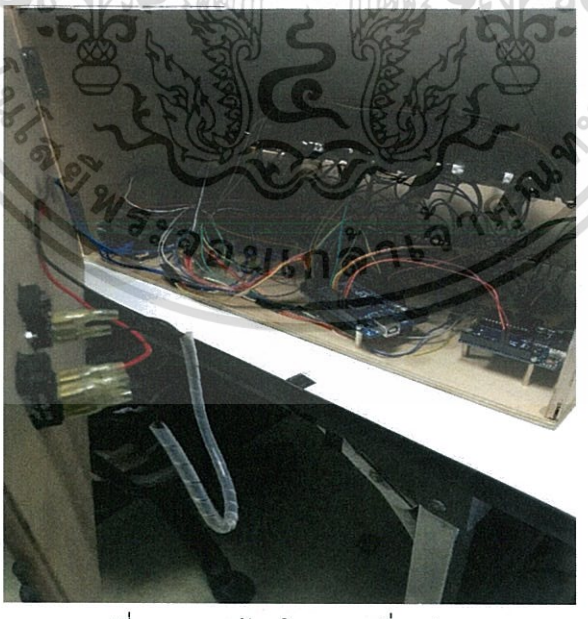
ในการเลือกใช้พาวเวอร์ซัพพลาย ต้องคำนวนจากกระแสและแรงดันที่อุปกรณ์แต่ละตัวใช้ ซึ่งอุปกรณ์แต่ละตัวใช้กระแสและแรงดันไม่เท่ากัน เซนเซอร์ใช้แรงดัน 5 โวลล์ ส่วนขดลวดโซลินอยด์ใช้แรงดัน 12 โวลล์ จึงจำเป็นต้องเลือกใช้พาวเวอร์ซัพพลาย 12 โวลล์ และใช้ไอซีควบคุมแรงดันเบอร์ 7805 เพื่อควบคุมแรงดันให้เหลือ 5 โวลล์ เพื่อใช้กับเซนเซอร์ ในส่วนของกระแสขดลวดโซลินอยด์ 1 ตัว ใช้กระแส 350 มิลลิแอมแปร์ มีทั้งหมด 6 ตัว จึงควรใช้พาวเวอร์ซัพพลาย 12 โวลล์ กระแส 3 แอมแปร์ขึ้นไป โดยเผื่อให้เซนเซอร์ 6 ตัว 1 แอมแปร์ โครงงานนี้ใช้พาวเวอร์ซัพพลายขนาด 12 โวลล์ 5 แอมแปร์

3.9 การออกแบบฮาร์ดแวร์

เลือกใช้รางอะลูมิเนียมเพราะมีน้ำหนักเบา แข็งแรง และมีผิวเรียบทำให้กล่องสามารถไหลได้สะดวก โดยใช้ความยาว 30 ซม. เนื่องจากไม่ต้องการให้ตัวต้นแบบมีขนาดใหญ่เกินไป และใช้ไม้อัดในการทำโครงสร้าง เพราะว่าง่ายต่อการตัดและมีราคาถูก โดยเราต้องการความชันเพื่อให้กล่องยาสามารถไหลได้ โดยเราเลือกใช้มุมระหว่างรางอะลูมิเนียมกับแนวระนาบเท่ากับ 37 องศา เพราะว่ามีความชันเพียงพอและง่ายต่อการคำนวน โดยสามารถคำนวนได้จากทฤษฎีบทพีทาโกรัสเกี่ยวกับความสัมพันธ์ของด้านทั้งสามของสามเหลี่ยมมุมฉาก



รูปที่ 15 ภาพด้านหน้าของเครื่องจ่ายยา



รูปที่ 16 ภาพด้านในของเครื่องจ่ายยา

รูปที่ 17 ภาพด้านข้างของเครื่องจ่ายยา

รูปที่ 18 ภาพด้านหลังของเครื่องจ่ายยา

บทที่ 4

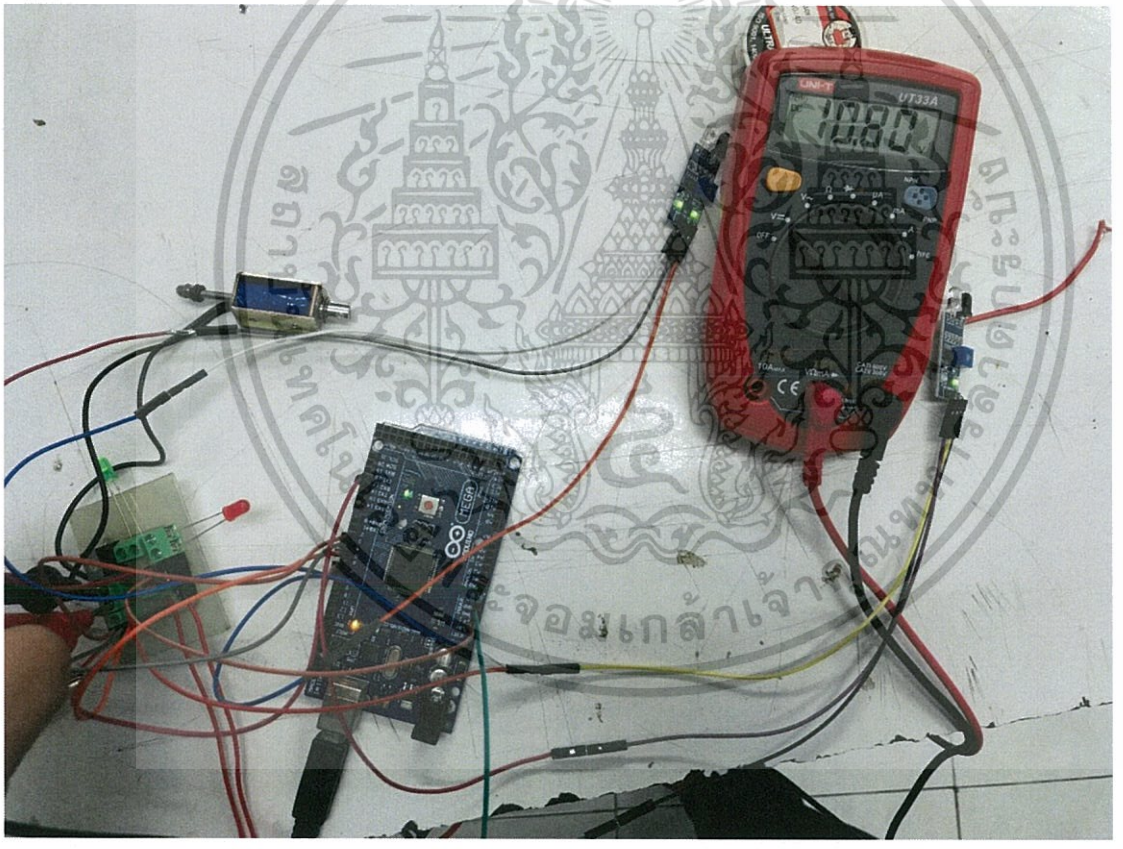
ผลการทดลอง

4.1 การทดสอบขดลวดโซลินอยด์

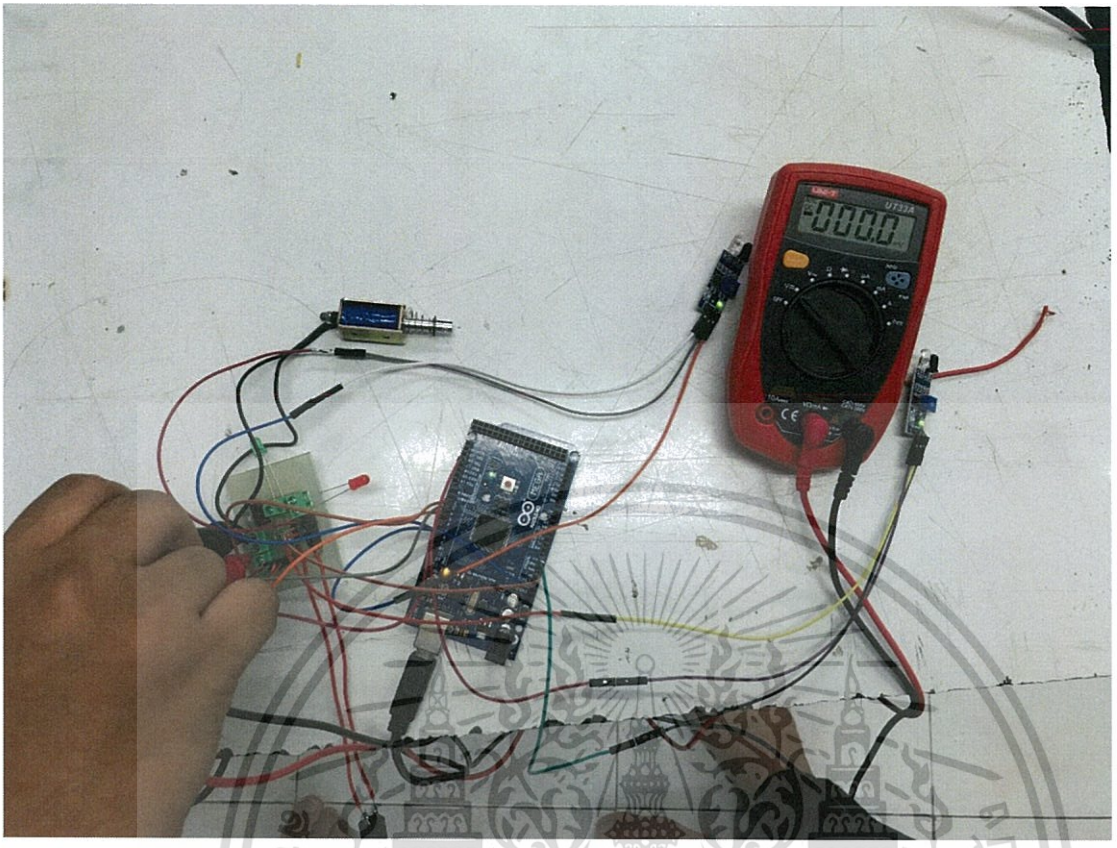
เมื่อทำการป้อนคำสั่งจากโปรแกรม วิชวลสตูโอ มาที่ไมโครคอนโทรลเลอร์และไมโครคอนโทรลเลอร์จะส่งแรงดันไปยังบอร์ดขับขดลวดโซลินอยด์ได้แรงดันที่ขดลวดโซลินอยด์ 10.8 โวลล์ ดังแสดงในรูปที่ 19 และเมื่อไม่ได้ทำการป้อนคำสั่งจากโปรแกรม แรงดันที่ขดลวดโซลินอยด์จะเท่ากับ 0 โวลล์ ดังแสดงในรูปที่ 20

	ACTIVATE	NO ACTIVATE
VOLTAGE	10.8 V	0 V

ตารางที่ 1



รูปที่ 19 ช่วงที่ขดลวดโซลินอยด์ทำงาน



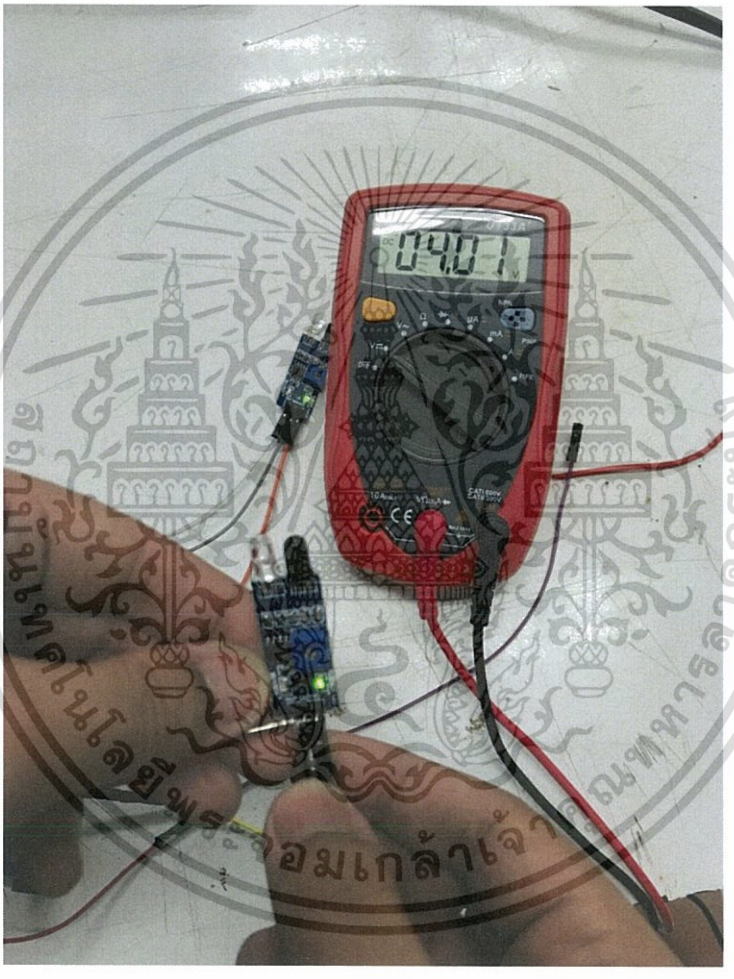
รูปที่ 20 ช่วงที่ขดลวดโซลินอยด์ไม่ทำงาน

4.2 การทดสอบเซนเซอร์อินฟราเรด

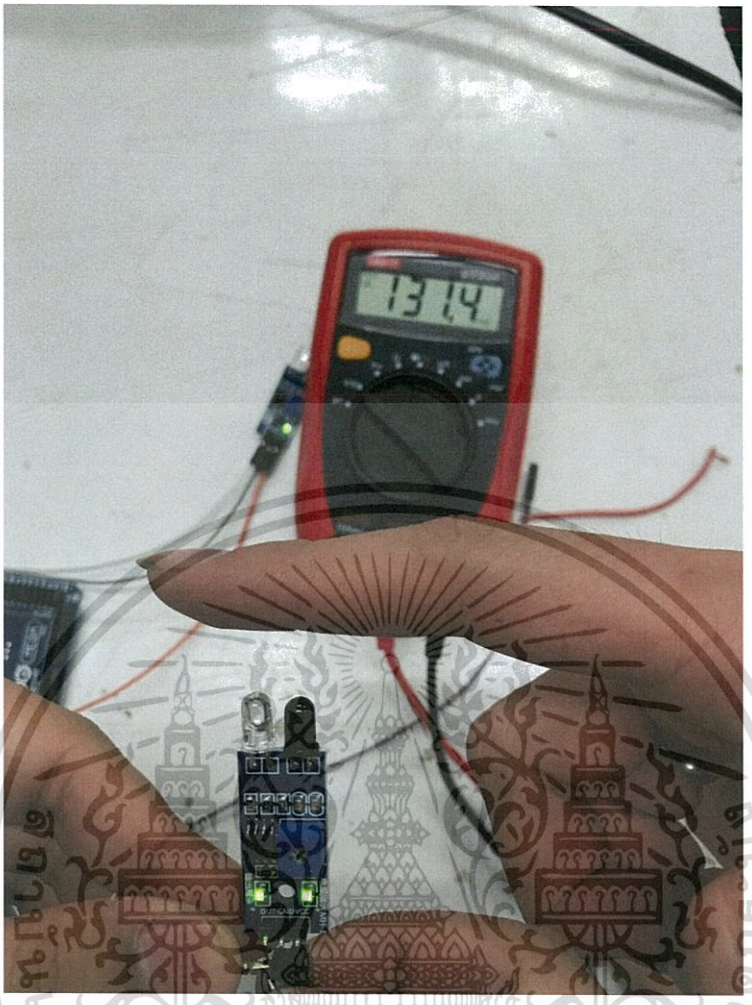
ทำการทดสอบการทำงานของเซนเซอร์เมื่อไม่ตรวจจับวัตถุเซนเซอร์จะให้แรงดันเอาท์พุทเท่ากับ 4.01 โวลล์ ดังแสดงในรูปที่ 21 และเมื่อเซนเซอร์ตรวจจับวัตถุได้จะให้แรงดันเอาท์พุทเท่ากับ 0.13 โวลล์ ดังแสดงในรูปที่ 22

	NO DETECT	DETECT
VOLTAGE	4.01 V	0.13 V

ตารางที่ 2



รูปที่ 21 การทำงานของเซนเซอร์ตอนที่ตรวจไม่เจอสิ่งของ



รูปที่ 22 การทำงานของเซนเซอร์ตอนที่ตรวจเจอสิ่งของ

4.3 การทดสอบการทำงานของระบบ

ในการทดสอบการทำงานของระบบจริง 10 ครั้ง พบว่ามีบางครั้งที่ขดลวดโซลินอยด์ดีดไม่ข้ามแผงกั้น และบางครั้งเซนเซอร์ตรวจจับผิดพลาด ดังแสดงในตาราง 3

	TESTING TIMES (NO.)	SUCCESS	FAILURE
SOLENOID	10	8	2
SENSOR 1	10	9	1
SENSOR 2	10	9	1

ตารางที่ 3

บทที่ 5

สรุปผลการทดลอง

5.1 สรุปผลการทดลอง

จากผลการทดลองเราจะเห็นได้ว่า ที่ขดลวดโซลินอยด์มีค่าความผิดพลาดมากกว่าที่เซนเซอร์อินฟราเรด ซึ่งเกิดจากปัจจัยหลายๆอย่างคือ การออกแบบโครงสร้างฮาร์ดแวร์ไม่ดีพอ ทำให้เกิดการติดขัดในการทำงานเช่น กล่องยาไม่ไหลไปที่จุดดีด และในส่วนของแรงดีดของขดลวดโซลินอยด์ที่เราใช้ในบางครั้งอาจจะไม่เพียงพอในการดีดกล่องยาที่มีน้ำหนักมากเกินไป ทั้งนี้เราสามารถแก้ไขได้โดยการเพิ่มความชันของรางเพื่อทำให้กล่องยาไหลได้ง่ายขึ้น และเรายังสามารถเพิ่มกำลังของขดลวดโซลินอยด์เพื่อที่จะสามารถดีดกล่องยาให้ออกไปได้ ในส่วนของเซนเซอร์อินฟราเรด ความผิดพลาดส่วนมากจะเกิดจากการที่เซนเซอร์มีความไวในการตรวจจับสูงทำให้การตรวจจับผิดพลาดไป แต่เราสามารถปรับได้ที่ตัวเซนเซอร์จึงทำให้เกิดความผิดพลาดน้อยลง

5.2 ปัญหา

อินฟราเรดเซนเซอร์ตรวจจับผิดพลาด เพราะในการติดตั้งตัวเซนเซอร์กับรางอะลูมิเนียมให้พอดี ทำได้ค่อนข้างยาก ทำให้เซนเซอร์ไปตรวจจับโดนรางอะลูมิเนียมแทนกล่องยา จึงทำให้มีขอผิดพลาดเกิดขึ้น ส่วนขดลวดโซลินอยด์ดีดกล่องยาไม่ข้ามแผงกั้นเนื่องจากขนาดแผงกั้นสูงเกินไปเล็กน้อย ทำให้ในบางครั้งแรงดีดไม่เพียงพอที่จะดีดกล่องยาให้ข้ามแผงกั้น และในการให้คำสั่งหลายๆคำสั่งพร้อมๆกันทำให้เกิดข้อผิดพลาดในการดีดของขดลวดโซลินอยด์ จึงทำการแก้ปัญหาโดยให้คำสั่งกับไมโครคอนโทรลเลอร์แค่ครั้งละ 1 คำสั่งเท่านั้น

5.3 ข้อเสนอแนะ

ในการออกแบบฮาร์ดแวร์ควรมีการวางแผนและการวัดขนาดที่แม่นยำมากกว่านี้ และควรเพิ่มการทำงานในส่วนของการติดฉลากยาเพราะในปัญหาที่เกิดขึ้นจริง เภสัชกรต้องมาทำการเขียนฉลากแต่ละกล่องเอง ทำให้ไม่สะดวกในการทำงาน

เอกสารอ้างอิง

[1] Sonthaya Nongnuch. 2018. "**Optical Sensor**" [Online]

Available: http://www.elec-za.com/เซ็นเซอร์แสง-optical-sensor/

[2] Aimagin. 2018. "**การใช้งานพอร์ตสื่อสาร UART**"[Online]

Available: http://aimagin.com/blog/การใช้งานพอร์ตสื่อสาร-uart/?lang=th

[3] Aimagin. 2018. "**การใช้งานพอร์ตสื่อสาร SPI**"[Online]

Available: http://aimagin.com/blog/spi/?lang=th

[4] D. K. a. R. Kalinsky. 2018. "**Introduction to Serial Peripheral Interface**," Embedded.com, UBM Electronics, 2002. [Online].

Available: http://www.embedded.com/electronics-blogs/beginner-s-corner/4023908/Introduction-to-Serial-Peripheral-Interface.

ภาคผนวก

โค้ดวิชวลสตูดิโอ

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.Sql;
using System.Data.SqlClient;
using System.IO.Ports;

namespace main
{
    public partial class Form1 : Form
    {
        string[] arr1 = new string[6];
        private SerialPort myport;
        SqlConnection con = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\me\Documents\dataaa.mdf;Integr
ated Security=True;Connect Timeout=30");

        public Form1()
        {

            InitializeComponent();
            init();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            disp_data();
        }
```

```
        private void button1_Click(object sender, EventArgs e)
        {
            con.Open();
            SqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "insert into [Table] values
('"+textBox1.Text+"','"+textBox2.Text+"','"+textBox3.Text+"','"+dateTimePicker1.Text+"','"+comboBo
x1.Text+"','"+domainUpDown1.Text+"')";
            cmd.ExecuteNonQuery();
            con.Close();
            disp_data();
            MessageBox.Show("Saved successfull");
            if (comboBox1.Text == "A" && domainUpDown1.Text == "1")
            {
                (arr1[0]) = "A";
            }
            if (comboBox1.Text == "B" && domainUpDown1.Text == "1")
            {
                (arr1[1]) = "B";
            }
            if (comboBox1.Text == "C" && domainUpDown1.Text == "1")
            {
                (arr1[2]) = "C";
            }
            if (comboBox1.Text == "D" && domainUpDown1.Text == "1")
            {
                (arr1[3]) = "D";
            }
            if (comboBox1.Text == "E" && domainUpDown1.Text == "1")
            {
                (arr1[4]) = "E";
            }
            if (comboBox1.Text == "F" && domainUpDown1.Text == "1")
            {
```

```
        (arr1[5]) = "F";
    }
}
public void disp_data()
{
    con.Open();
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select * from [Table]";
    cmd.ExecuteNonQuery();
    DataTable dt = new DataTable();
    SqlDataAdapter da = new SqlDataAdapter(cmd);
    da.Fill(dt);
    dataGridView1.DataSource = dt;
    con.Close();
}

private void button2_Click(object sender, EventArgs e)
{
    con.Open();
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "delete from [Table]";
    cmd.ExecuteNonQuery();
    con.Close();
    disp_data();
    MessageBox.Show("delete successfull");
}

private void button3_Click(object sender, EventArgs e)
{
    con.Open();
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
```

```
        cmd.CommandText = "update [Table] set name='"+textBox1.Text+"' where
name='"+textBox2.Text+"'";
        cmd.ExecuteNonQuery();
        con.Close();
        disp_data();
        MessageBox.Show("update successfull");
    }

    private void button4_Click(object sender, EventArgs e)
    {
        MessageBox.Show("OK!");

    }

    private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
    {

    }

    private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
    {

    }

    private void domainUpDown1_SelectedItemChanged(object sender, EventArgs e)
    {

    }
    private void init()
    {
        try
        {
            myport = new SerialPort();
            myport.BaudRate = 9600;
            myport.PortName = "COM4";
```

```
            myport.Open();
        }
        catch (Exception)
        {
            MessageBox.Show("Error");
        }
        button1.Enabled = true;

    }

    private void label1_Click(object sender, EventArgs e)
    {

    }

    private void label3_Click(object sender, EventArgs e)
    {

    }

    private void textBox2_TextChanged(object sender, EventArgs e)
    {

    }

    private void label4_Click(object sender, EventArgs e)
    {

    }

    private void button5_Click(object sender, EventArgs e)
    {
        myport.WriteLine(arr1[0]);
        myport.WriteLine(arr1[1]);
        myport.WriteLine(arr1[2]);
```

```
        myport.WriteLine(arr1[3]);
        myport.WriteLine(arr1[4]);
        myport.WriteLine(arr1[5]);

    }

    private void button6_Click(object sender, EventArgs e)
    {
        MessageBox.Show("Clear!");
        (arr1[0]) = "0";
        (arr1[1]) = "0";
        (arr1[2]) = "0";
        (arr1[3]) = "0";
        (arr1[4]) = "0";
        (arr1[5]) = "0";

    }

    private void button4_Click_1(object sender, EventArgs e)
    {

    }

    private void editsurname_Click(object sender, EventArgs e)
    {
        con.Open();
        SqlCommand cmd = con.CreateCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "update [Table] set surname='" + textBox2.Text + "' where
surname='" + textBox1.Text + "'";
        cmd.ExecuteNonQuery();
        con.Close();
        disp_data();
        MessageBox.Show("update successfull");
    }
```

```
        private void button4_Click_2(object sender, EventArgs e)
        {
            myport.WriteLine("G");
            MessageBox.Show("reset!");
        }
    }
}
```

โค้ดการทำงานของมาสเตอร์

```
#include<SoftwareSerial.h>
SoftwareSerial mySerial(10, 11);
char x = 0;
void setup(void)
{
  Serial.begin(9600);
  mySerial.begin(9600);
}
void loop ()
{
  while (Serial.available() == 0) {}
  x = Serial.read();
  mySerial.write(x);


 }
```

โค้ดการทำงานของเสลฟ

```
#include<SoftwareSerial.h>

SoftwareSerial mySerial(10 ,11);

int TRAN1 = 4;

int LED1 = 5;

int LED2 = 6;

int BUZ1 = 7;

int TRAN2 = 22;

int LED3 = 24;

int LED4 = 26;

int BUZ2 = 28;

int TRAN3 = 30;

int LED5 = 32;

int LED6 = 34;

int BUZ3 = 36;

int TRAN4 = 38;

int LED7 = 40;

int LED8 = 42;

int BUZ4 = 44;

int TRAN5 = 31;

int LED9 = 33;

int LED10 = 35;

int BUZ5 = 37;
```

```
int TRAN6 = 39;
int LED11 = 41;
int LED12 = 43;
int BUZ6 = 45;

void setup (void)
{
 pinMode(TRAN1,OUTPUT);
 pinMode(LED1,OUTPUT);
 pinMode(LED2,OUTPUT);
 pinMode(BUZ1,OUTPUT);
 pinMode(TRAN2,OUTPUT);
 pinMode(LED3,OUTPUT);
 pinMode(LED4,OUTPUT);
 pinMode(BUZ2,OUTPUT);
 pinMode(TRAN3,OUTPUT);
 pinMode(LED5,OUTPUT);
 pinMode(LED6,OUTPUT);
 pinMode(BUZ3,OUTPUT);
 pinMode(TRAN4,OUTPUT);
 pinMode(LED7,OUTPUT);
 pinMode(LED8,OUTPUT);
 pinMode(BUZ4,OUTPUT);
```

```
 pinMode(TRAN5,OUTPUT);
 pinMode(LED9,OUTPUT);
 pinMode(LED10,OUTPUT);
 pinMode(BUZ5,OUTPUT);
 pinMode(TRAN6,OUTPUT);
 pinMode(LED11,OUTPUT);
 pinMode(LED12,OUTPUT);
 pinMode(BUZ6,OUTPUT);
 Serial.begin(9600);
 digitalWrite(TRAN1,LOW);
 mySerial.begin(9600);
}

void loop(void)
{
if(mySerial.available() > 0)
{
  char data = mySerial.read();
  attachInterrupt(digitalPinToInterrupt(2),blink1,CHANGE);
  attachInterrupt(digitalPinToInterrupt(3),blink2,CHANGE);
  attachInterrupt(digitalPinToInterrupt(18),blink3,CHANGE);
  attachInterrupt(digitalPinToInterrupt(19),blink4,CHANGE);
  attachInterrupt(digitalPinToInterrupt(20),blink5,CHANGE);
```

```
  attachInterrupt(digitalPinToInterrupt(21),blink6,CHANGE);


  //mySerial.print(data);
  //mySerial.write(data);
  //Serial.print(data);
switch(data)
{
case 'A' : if(digitalRead(A0) == LOW)
{
digitalWrite(TRAN1,HIGH);
digitalWrite(LED1,LOW);
digitalWrite(LED2,HIGH);break;
}
else if(digitalRead(A0) == HIGH)
{
digitalWrite(BUZ1,HIGH);
delay(500);
digitalWrite(BUZ1,LOW);
digitalWrite(LED1,HIGH);break;
}
case 'B' : if(digitalRead(A1) == LOW)
{
digitalWrite(TRAN2,HIGH);
```

```
digitalWrite(LED3,LOW);
digitalWrite(LED4,HIGH);break;
}
else if(digitalRead(A1) == HIGH)
{
digitalWrite(BUZ2,HIGH);
delay(500);
digitalWrite(BUZ2,LOW);
digitalWrite(LED3,HIGH);break;
}
case 'C' : if(digitalRead(A2) == LOW)
{
digitalWrite(TRAN3,HIGH);
digitalWrite(LED5,LOW);
digitalWrite(LED6,HIGH);break;
}
else if(digitalRead(A2) == HIGH)
{
digitalWrite(BUZ3,HIGH);
delay(500);
digitalWrite(BUZ3,LOW);
digitalWrite(LED5,HIGH);break;
}
```

```
case 'D' : if(digitalRead(A3) == LOW)
{
digitalWrite(TRAN4,HIGH);
digitalWrite(LED7,LOW);
digitalWrite(LED8,HIGH);break;
}
else if(digitalRead(A3) == HIGH)
{
digitalWrite(BUZ4,HIGH);
delay(500);
digitalWrite(BUZ4,LOW);
digitalWrite(LED7,HIGH);break;
}
case 'E' : if(digitalRead(A4) == LOW)
{
digitalWrite(TRAN5,HIGH);
digitalWrite(LED9,LOW);
digitalWrite(LED10,HIGH);break;
}
else if(digitalRead(A4) == HIGH)
{
digitalWrite(BUZ5,HIGH);
delay(500);
```

```
digitalWrite(BUZ5,LOW);
digitalWrite(LED9,HIGH);break;
}
case 'F' : if(digitalRead(A5) == LOW)
{
digitalWrite(TRAN6,HIGH);
digitalWrite(LED11,LOW);
digitalWrite(LED12,HIGH);break;
}
else if(digitalRead(A5) == HIGH)
{
digitalWrite(BUZ6,HIGH);
delay(500);
digitalWrite(BUZ6,LOW);
digitalWrite(LED11,HIGH);break;
}
case 'G' :
digitalWrite(TRAN1,LOW);
digitalWrite(LED1,LOW);
digitalWrite(LED2,LOW);
digitalWrite(BUZ1,LOW);
digitalWrite(TRAN2,LOW);
digitalWrite(LED3,LOW);
```

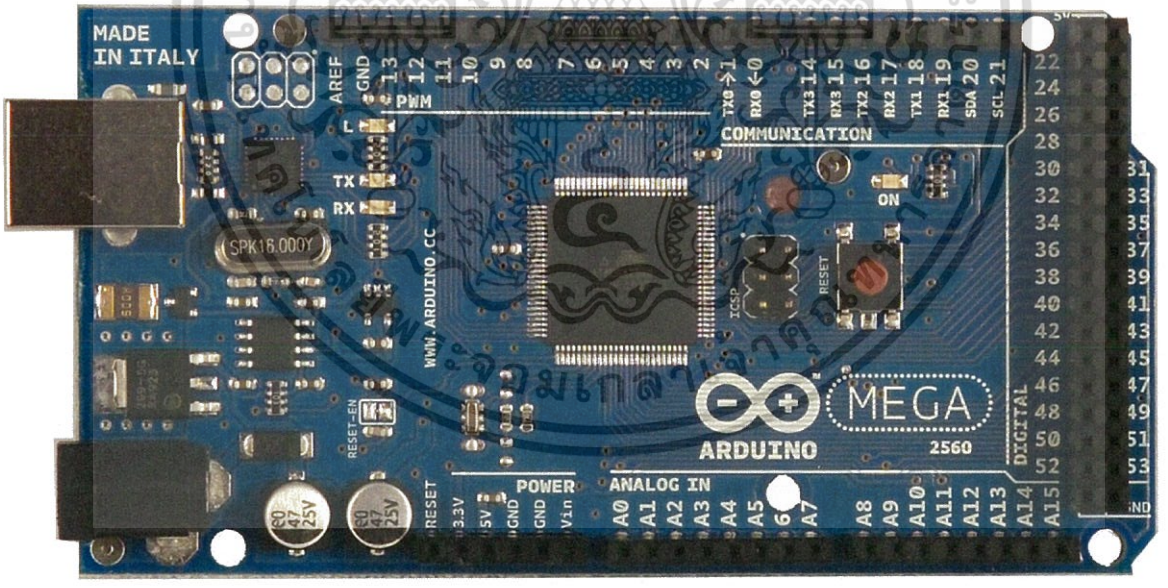
```
digitalWrite(LED4,LOW);
digitalWrite(BUZ2,LOW);
digitalWrite(TRAN3,LOW);
digitalWrite(LED5,LOW);
digitalWrite(LED6,LOW);
digitalWrite(BUZ3,LOW);
digitalWrite(TRAN4,LOW);
digitalWrite(LED7,LOW);
digitalWrite(LED8,LOW);
digitalWrite(BUZ4,LOW);
digitalWrite(TRAN5,LOW);
digitalWrite(LED9,LOW);
digitalWrite(LED10,LOW);
digitalWrite(BUZ5,LOW);
digitalWrite(TRAN6,LOW);
digitalWrite(LED11,LOW);
digitalWrite(LED12,LOW);
digitalWrite(BUZ6,LOW);
break;
}
}
}
void blink1()
```

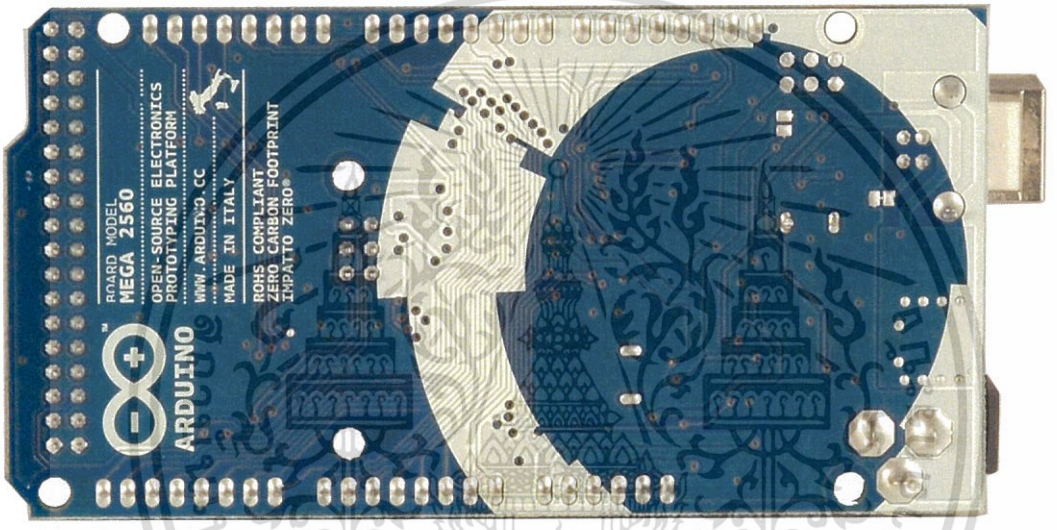
```
{
digitalWrite(TRAN1,LOW);
digitalWrite(LED2,LOW);
}
void blink2()
{
digitalWrite(TRAN2,LOW);
digitalWrite(LED4,LOW);
}
void blink3()
{
digitalWrite(TRAN3,LOW);
digitalWrite(LED6,LOW);
}
void blink4()
{
digitalWrite(TRAN4,LOW);
digitalWrite(LED8,LOW);
}
void blink5()
{
digitalWrite(TRAN5,LOW);
digitalWrite(LED10,LOW);
```

```
}
void blink6()
{
digitalWrite(TRAN6,LOW);
digitalWrite(LED12,LOW);
}
```

ARDUINO

Arduino Mega 2560 Datasheet





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.
- **I2C: 20 (SDA) and 21 (SCL).** Support I2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I2C 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.

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 I2C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).*

JF-0530B DC12V 300mA 5N/10mm Pull-Push-Type Solenoid Electromagnet

Specifications:
Product Name:DC Solenoid Electromagnet
Model:JF-0530B
Rated Voltage:DC 12V
Type:Pull Push
Rated Current:300mA
Force& Stroke:5N/10mm
Body Size:30x16x15mm
Plunger Bar Size:6x58mm
Mounting Hole Dia.:2.5mm
Cable Length:20cm
Material:Metal, Electronic Parts

Description:
1. Pull push type, linear motion, plunger return, DC solenoid electronmagnet.
2. DC Solenoid Electromagnet mainly used in vending machines, transport equipment, office facility household appliance, mechanical, etc.
3. Solenoids of this category work externally through pulling pushing in plunger.
4. When energized, doing work through pulling pushing in plunger joined object.

Note:Please don't power for a long time.Just about seconds.