การพัฒนาการขับเคลื่อนและนำทางของหุ่นยนต์เพื่อช่วยเหลือผู้พิการทางสายตาในการเลือกซื้อสินค้า

DEVELOPMENT OF THE NAVIGATION SYSTEM FOR A PROTOTYPED SHOPPING ROBOT

กฤษณะ ไทยศรี

ธิปไตย มณีบูรณ์

นิสิต ภูระหงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

การพัฒนาการขับเคลื่อนและนำทางของหุ่นยนต์เพื่อช่วยเหลือผู้พิการทางสายตาในการเลือกซื้อสินค้า

DEVELOPMENT OF THE NAVIGATION SYSTEM FOR A PROTOTYPED SHOPPING ROBOT

กฤษณะ ไทยศรี

ธิปไตย มณีบูรณ์

นิสิต ภู่ระหงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

DEVELOPMENT OF THE NAVIGATION SYSTEM FOR A PROTOTYPED SHOPPING ROBOT

KRISANA THAISRI

TIPATAI MANEEBOON

NISIT PHURAHONG

THIS THESIS IS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF ENGINEERING IN CONTROL ENGINEERING
FACULTY OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2013

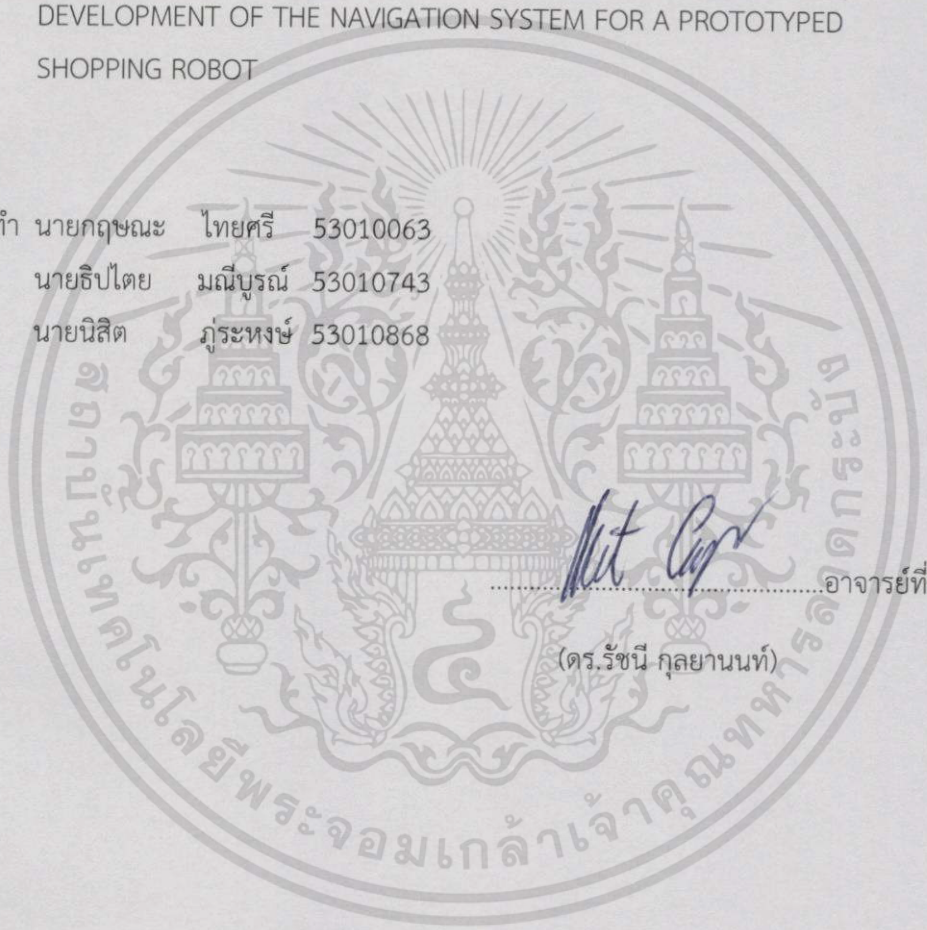
ปริญญานิพนธ์ปีการศึกษา 2556

สาขาวิชาวิศวกรรมการวัดและควบคุม คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาการขับเคลื่อนและนำทางของหุ่นยนต์เพื่อช่วยเหลือผู้พิการทางสายตาในการเลือกซื้อสินค้า

DEVELOPMENT OF THE NAVIGATION SYSTEM FOR A PROTOTYPED SHOPPING ROBOT

ผู้จัดทำ นายกฤษณะ ไทยศรี 53010063
นายธิปไตย มณีบูรณ์ 53010743
นายนิสิต ภู่ระหงษ์ 53010868



.....อาจารย์ที่ปรึกษา

(ดร.รัชนี กุลยานนท์)

การพัฒนาการขับเคลื่อนและนำทางของหุ่นยนต์เพื่อช่วยเหลือผู้พิการทางสายตาในการเลือกซื้อสินค้า

โดย
กฤษณะ ไทยศรี 53010063
ธิปไตย มณีบูรณ์ 53010743
นิสิต ภู่ระหงษ์ 53010868

อาจารย์ที่ปรึกษา
ดร.รัชนี กุลยานนท์

ปีการศึกษา 2556

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ นำเสนอการพัฒนาการขับเคลื่อนและนำทางของหุ่นยนต์เพื่อช่วยเหลือผู้พิการทางสายตาในการเลือกซื้อสินค้า โดยใช้หลักการของ A* Algorithm ในการเลือกเส้นทางที่เหมาะสมให้กับตัวรถหุ่นยนต์นำทาง เพื่อให้ตัวรถหุ่นยนต์นำทางสามารถเดินทางจากจุดเริ่มต้นไปยังจุดเป้าหมายได้ โดยที่สามารถหลบหลีกสิ่งกีดขวางที่ปรากฏอยู่ในแผนที่ได้ ในการทดสอบทฤษฎี A* Algorithm เราได้ทำการสร้างตัวรถหุ่นยนต์นำทางขึ้น โดยใช้ DC Motors ในการขับเคลื่อนตัวรถหุ่นยนต์นำทาง ติดตั้ง Encoder ที่ล้อขับเคลื่อนเพื่อวัดความเร็วของตัวรถหุ่นยนต์นำทาง ใช้ Compass Module เพื่อวัดทิศทางของตัวรถหุ่นยนต์นำทาง และมี RFID เพื่อช่วยในการตรวจสอบการเคลื่อนที่ของตัวรถหุ่นยนต์นำทางว่าเคลื่อนที่มายังทิศทางที่ถูกต้องหรือไม่ ข้อมูลทั้งหมดที่วัดได้จะถูกส่งต่อมาที่ตัวไมโครคอนโทรลเลอร์ เพื่อควบคุมความเร็วและทิศทางของตัวรถหุ่นยนต์นำทาง

DEVELOPMENT OF THE NAVIGATION SYSTEM FOR A PROTOTYPED SHOPPING ROBOT

By

Krisana Thaisri 53010063

Tipatai Maneeboon 53010743

Nisit Phurahong 53010868

Advisor

Dr.Rutchanee Gullayanon

Academic Year 2013

ABSTRACT

The objective of this study is to develop a navigation system for a prototyped shopping robot in order to help disability shoppers in supermarkets. We use A* Algorithm theory to determine an optimal path which can maneuver the shopping robot to its destination while avoiding obstacles shown in the map. To test the A* Algorithm, we create a robot base with dc motors for the driving mechanism, encoders at each wheel for velocity measurements, and a compass module for robot direction measurement. RFID check points are added at pre-determined locations on the map. They are used to check if the shopping robot has accomplished a certain portion of the pre-determined path. All measurement data are sent to the microcontroller to control both the speed and the direction of the shopping robot.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ประสบความสำเร็จได้ด้วยดี คณะผู้จัดทำขอกราบขอบพระคุณอาจารย์ที่ปรึกษา ดร.รัชนี กุลยานนท์ เป็นอย่างสูงที่คอยแนะนำวิธีแก้ไขปัญหาต่างๆ ในโครงงานทั้งทางทฤษฎีและทางปฏิบัติแก่คณะผู้จัดทำมาโดยตลอด ทำให้ผู้จัดทำเข้าใจที่มาของปัญหา และสามารถแก้ไขปัญหาต่างๆ ที่เกิดขึ้นได้อย่างถูกวิธี รวมทั้งเอื้อเฟื้ออุปกรณ์ที่จำเป็น และความช่วยเหลืออื่นๆ ที่เป็นประโยชน์แก่โครงงาน

ขอขอบพระคุณสาขาวิชาวิศวกรรมการวัดและควบคุม ที่ได้เอื้อเฟื้ออุปกรณ์ และสถานที่ในการทำวิจัย จนผู้จัดทำสามารถทำปริญญานิพนธ์นี้เสร็จสิ้น

ขอขอบคุณเพื่อนและรุ่นพี่ทุกคนที่ให้กำลังใจ สนับสนุนอุปกรณ์ที่ขาดเหลือ กระตุ้นเตือน รวมทั้งคอยถามไถ่ความคืบหน้าของโครงงานตลอดจนคอยให้คำปรึกษาและให้กำลังใจอยู่เสมอ

สุดท้ายนี้ผู้จัดทำขอกราบขอบพระคุณบิดา มารดา และครอบครัว ที่คอยเป็นกำลังใจที่ดีตลอดมา รวมถึงการสนับสนุนในเรื่องของงบประมาณที่ขาดเหลือ ตลอดจนเป็นแรงบันดาลใจที่ทำให้โครงงานนี้สำเร็จสมบูรณ์ได้

คณะผู้จัดทำ
นายกฤษณะ ไทยศรี
นายธิปไตย มณีบูรณ์
นายนิสิต ภู่ระหงษ์

สารบัญ

	หน้า
บทคัดย่อ	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญรูป	VI
สารบัญตาราง	VIII
บทที่ 1 บทนำ	1
1.1 กล่าวนำ	1
1.2 วัตถุประสงค์ของปริญญานิพนธ์	2
1.3 การศึกษาและจัดทำปริญญานิพนธ์	2
บทที่ 2 ทฤษฎีและความรู้พื้นฐาน	3
2.1 ทฤษฎี A* Algorithm	4
2.2 การเคลื่อนที่ในแบบต่างๆ	8
2.2.1 การเคลื่อนที่ในแนวเส้นตรง	8
2.2.2 การเคลื่อนที่เชิงมุม	9
2.3 รัศมีในการเลี้ยว	10
บทที่ 3 การออกแบบและการทดลอง	12
3.1 การออกแบบโปรแกรมจำลองการเคลื่อนที่	12
3.1.1 แผนที่	12
3.1.2 การพัฒนา A* Algorithm	14
3.1.3 ลำดับการทำงานของโปรแกรม	16
3.2 หุ่นยนต์นำทาง	18
3.3 แผนภาพการทำงานของหุ่นยนต์นำทาง	18
3.4 อุปกรณ์	19
3.4.1 Arduino UNO R3	20
3.4.2 DC Motors	21
3.4.3 วงจรลดแรงดัน	23
3.4.5 Encoder	23
3.4.6 HMC5883L 3-Axis Digital Compass Module	24
3.4.7 RFID Module RC-522	25
3.5 การต่อวงจรรวม	26

สารบัญ(ต่อ)

	หน้า
3.6 การเขียนโปรแกรม	26
3.6.1 การเขียนโปรแกรมแบบจำลองการเคลื่อนที่ของรถหุ่นยนต์นำทาง	26
3.6.2 การเขียนโปรแกรมใช้ในการควบคุมตัวรถหุ่นยนต์นำทาง	27
3.6.3 การเขียนโปรแกรมใช้สำหรับ Compass Module	27
3.6.4 การเขียนโปรแกรมใช้สำหรับ RFID	27
บทที่ 4 การทดลองและผลการทดลอง	28
4.1 ผลการทดลองจากโปรแกรม Matlab	28
4.1.1 การสร้างเส้นทางการเดิน	28
4.1.2 กำหนดจุดเริ่มต้น และจุดเป้าหมาย	29
4.1.3 รันโปรแกรมในส่วนของ A* Algorithm	30
4.1.4 รันโปรแกรมในส่วนที่นำข้อมูลจาก A* Algorithm มาพัฒนาต่อ	31
4.1.5 วิเคราะห์และสรุปผล	31
4.2 การทดลองความแม่นยำ HMC5883L Compass Module	32
4.3 การทดลองการเคลื่อนที่ของหุ่นยนต์นำทาง	33
บทที่ 5 สรุปและวิจารณ์ปัญหา	38
5.1 สรุปผลการดำเนินงาน	38
5.2 ปัญหาและอุปสรรค	38
5.3 แนวทางในการแก้ไขปัญหา	38
5.4 แนวทางการพัฒนาต่อ	39
เอกสารอ้างอิง	40
ภาคผนวก	41
ภาคผนวก ก	42
ภาคผนวก ข	50

สารบัญรูป

รูปที่	หน้า
2.1 แสดงจุดเริ่มต้นและจุดเป้าหมาย	5
2.2 แสดงค่า F, G และ H ที่ได้จากการคำนวณ	5
2.3 แสดงการเลือกช่องกริดที่มีค่า F น้อยที่สุด	6
2.4 แสดงการตรวจสอบ และเลือกค่า G ของเส้นทางใหม่ที่มีค่าต่ำกว่า	6
2.5 แสดงเส้นทางเดินไปยังจุดเป้าหมายในลักษณะแนวนอน และแนวตั้งเท่านั้น	7
2.6 แสดงการย้อนกลับจากจุดเป้าหมายจนถึงจุดเริ่มต้น	7
2.7 แสดงตำแหน่งสิ่งกีดขวาง และมุมเลี้ยว	10
3.1 แสดงการกำหนดค่าแผนที่ด้วยโปรแกรม Microsoft Excel	12
3.2 ตัวอย่างหน้าต่างแผนที่จำลองด้วยโปรแกรม Matlab	13
3.3 แสดงผลการควบคุมเส้นทางการเดินด้วยพื้นที่สีเทา	14
3.4 แสดงมุมการเลี้ยวแบบตั้งฉาก	14
3.5 แสดงมุมการเลี้ยวที่มีความโค้งมากขึ้น	15
3.6 แสดงมุมเลี้ยวที่แปรผันตามสิ่งกีดขวาง	15
3.7 แสดงตัวอย่างการกำหนดจุดเริ่มต้น และจุดเป้าหมาย	16
3.8 แสดงตัวอย่างเส้นทางที่ได้จาก A* Algorithm	17
3.9 โครงสร้างหุ่นยนต์นำทาง	18
3.10 แผนภาพการทำงานของหุ่นยนต์นำทาง	19
3.11 Arduino UNO R3	20
3.12 Pin Arduino UNO R3	20
3.13 แผนภาพของไมโครคอนโทรลเลอร์	21
3.14 IC เบอร์ L293D	22
3.15 วงจรควบคุมการขับมอเตอร์	22
3.16 วงจรลดแรงดัน 7. 2 V เป็น 5.0 V	23
3.17 Encoder LM393	24
3.18 HMC5883L 3-Axis Digital Compass Module	24
3.19 RFID RC522, S50 IC Card, S50 IC Key	25
3.20 MFRC522	25
3.21 การต่อวงจรรวมเข้ากับอุปกรณ์ต่างๆ	26

สารบัญรูป(ต่อ)

รูปที่	หน้า
4.1 แสดงผลการควบคุมเส้นทางการเดินด้วยพื้นที่สีเทา	28
4.2 แสดงพื้นที่ที่ทำการสร้างสิ่งกีดขวางเสมือนล้อมรอบสิ่งกีดขวางจริง	29
4.3 แสดงตำแหน่งเริ่มต้น และเป้าหมาย	29
4.4 แสดงการจำลองจุดเป้าหมายใหม่	30
4.5 แสดงเส้นทางจากจุดเริ่มต้นไปยังจุดเป้าหมาย	30
4.6 แสดงผลจากแนวคิดการพัฒนาเส้นทางเดิน	31
4.7 แสดงผลมุมจากการวาง HMC5883L หยุดนิ่งที่มุม 0 องศา	32
4.8 แสดงผลมุมจากการวาง HMC5883L หยุดนิ่งที่มุม 45 องศา	32
4.9 แสดงผลมุมที่ทำการหมุน HMC5883L ที่ 0 ถึง 90 องศา	33

สารบัญตาราง

ตารางที่	หน้า
2.1 เปรียบเทียบ Navigation System ของหุ่นเดินตามเส้นและหุ่นที่ใช้วิธี A* Algorithm	3
3.1 รายละเอียดของตัวไมโครคอนโทรลเลอร์	20
3.2 คุณสมบัติ Encoder	23
3.3 คุณสมบัติ HMC5883L 3-Axis Digital Compass Module	24
3.4 การเชื่อมต่อ Arduino UNO R3 เข้ากับ Compass Module	25
4.1 ความผิดพลาดการเดินตรง	34
4.2 ความผิดพลาดการเดินเลี้ยวซ้าย	34
4.3 ความผิดพลาดการเดินเลี้ยวขวา	35
4.4 ความผิดพลาดการเดินตรงแล้วเลี้ยวซ้าย	35
4.5 ความผิดพลาดการเดินตรงแล้วเลี้ยวขวา	36
4.6 ความผิดพลาดรวมในการเคลื่อนที่ของหุ่นยนต์นำทาง	36

บทที่ 1
บทนำ

1.1 บทนำ

ในปัจจุบันมีสิ่งที่สร้างขึ้นเพื่ออำนวยความสะดวกให้กับมนุษย์เกิดขึ้นมากมายเพื่อตอบสนองความต้องการในชีวิตประจำวันให้ง่าย และรวดเร็วมากขึ้น พฤติกรรมที่ทำเป็นประจำอย่างหนึ่ง คือ การเลือกซื้อสินค้าในการดำรงชีวิตทั้งจากร้านสะดวกซื้อ หรือห้างสรรพสินค้า ในห้างสรรพสินค้าจะเห็นว่ามีคนทั่วไปเดินเลือกสินค้ามากมายเป็นเรื่องปกติ แต่สำหรับผู้พิการหรือแม้กระทั่งคนที่ไม่ค่อยมีเวลามากนักอาจเป็นเรื่องที่ลำบาก และไม่สะดวกที่จะทำได้ตามปกติ จึงเหมาะที่จะมีสิ่งที่อำนวยความสะดวกสำหรับกลุ่มคนเหล่านี้ขึ้น จุดประสงค์ในการทำ PickitBot คือ หุ่นยนต์ที่เป็นตัวแทนในการทำสิ่งเหล่านี้แทนมนุษย์ได้ ทั้งการไปยังที่ตั้งของสินค้า (Navigation) และการหยิบสินค้าจากชั้นวาง (Manipulator) ตามความต้องการของผู้ใช้งาน

ความคิดริเริ่มที่จะเริ่มทำตัวหุ่นยนต์เพื่อช่วยเหลือผู้ที่มาใช้บริการตามห้างสรรพสินค้าซุปเปอร์มาเก็ตนั้น มีจุดเริ่มต้นมาจากการที่ว่าในปัจจุบันนั้นผู้บริโภคนั้นได้หันมาซื้อของเครื่องใช้ วัสดุอุปกรณ์รวมถึงเครื่องอุปโภคบริโภคต่างๆ ในซุปเปอร์มาร์เก็ตกันเป็นจำนวนมาก เนื่องจากในปัจจุบันเป็นสังคมที่ต้องการความสะดวกสบายและเร่งรีบ การที่มาซื้อสินที่ซุปเปอร์มาเก็ตนั้นจึงตอบโจทย์ผู้บริโภค เนื่องจากมีเครื่องใช้วัสดุอุปกรณ์รวมถึงอาหารนั้นครบในทุกๆ ด้าน ผู้บริโภคที่มาใช้บริการที่ซุปเปอร์มาเก็ตนั้น ประกอบไปด้วยลูกค้าที่หลากหลายปัญหาที่เราพบเห็นก็คือ การซื้อของในซุปเปอร์มาเก็ตอาจจะบริการได้ไม่ครอบคลุมถึงลูกค้าในทุกๆ ส่วนอย่างกรณีของผู้สูงอายุที่เดินทางมาเพื่อซื้อของ ก็อาจจะทำได้ไม่สะดวกนั้นผู้พิการทางด้านส่วนต่างๆ ของร่างกายซึ่งไม่อาจจะเดินไปยังจุดที่วางสินค้าที่ต้องการ หรือไม่สามารถหยิบจับสินค้าที่ต้องการได้ รวมถึงครอบครัวที่ต้องการใช้เวลาไปทำสิ่งอื่น แทนที่จะมาเดินเลือกซื้อสินค้าที่ต้องการ ซึ่งการที่ประดิษฐ์หุ่นยนต์เพื่อช่วยเหลือผู้ที่มาซื้อสินค้าในซุปเปอร์มาเก็ตขึ้นมานั้นจะช่วยตอบโจทย์ในการแก้ไขปัญหาต่างๆ ที่ได้พบเนื่องจากหุ่นยนต์เพื่อช่วยเหลือผู้ที่มาซื้อสินค้า ที่ประดิษฐ์ขึ้นมานั้นจะสามารถระบุตำแหน่งของสินค้ารวมถึงจำนวนสินค้าที่ต้องการ โดยที่ตัวหุ่นมีตะกร้าติดอยู่พร้อมกับตัวหุ่นจะสามารถเคลื่อนที่ไปยังเป้าหมายสินค้าที่ต้องการโดยใช้เวลาและพลังงานน้อยที่สุด และจะมีอุปกรณ์แขนกลเพื่อช่วยเหลือในการหยิบจับสินค้าที่ต้องการ รวมถึงปริมาณสินค้าที่ถูกต้องอีกด้วยโดยที่ตัวหุ่นยนต์ไม่จำเป็นที่จะต้องมีคนคอยควบคุม ซึ่งจะช่วยให้ผู้ที่พิการในส่วนต่างๆ รวมถึงผู้สูงวัย หรือบุคคลที่ไม่มีเวลาในการเลือกซื้อสินค้า ซึ่งจะถือว่าเป็นประโยชน์อย่างยิ่งในการช่วยเหลือสังคม ให้มีการดำรงชีวิตประจำวันได้ดียิ่งขึ้นถึงเป็นการยกระดับการเลือกซื้อสินค้าในห้างซุปเปอร์มาเก็ตขึ้นมาอีกระดับหนึ่ง

1.2 วัตถุประสงค์ในการทำโครงงาน

1. สามารถสร้างแบบจำลองโดยการเขียนโปรแกรมของการเคลื่อนที่จากจุดเริ่มต้นไปยังจุดเป้าหมายโดยหลบหลีกสิ่งกีดขวางที่ปรากฏอยู่บนแผนที่ได้
2. ตัวรถนำทางสามารถเคลื่อนที่จากจุดเริ่มต้นไปยังจุดเป้าหมายปลายทาง

1.3 การศึกษาและจัดทำปริญญานิพนธ์

ในการทำโครงงานส่วน Navigation นั้นเป็นส่วนของการนำทางที่ต้องการให้มีความผิดพลาดน้อยที่สุดในระหว่างการทำงานจึงต้องศึกษาหาข้อมูลเกี่ยวกับ Algorithm ที่เหมาะสมกับเป้าหมายที่ต้องการ เมื่อได้ Algorithm แล้วจึงหาวิธีสร้างโปรแกรม และจำลองการทำงานเพื่อหา Function ที่สามารถนำมาใช้งานได้จริง และสามารถสร้างหุ่นยนต์นำทางโดยเขียนโปรแกรมจากจุดเริ่มต้นไปยังจุดเป้าหมายได้ โดยที่ผู้ศึกษาสามารถศึกษาได้จากปริญญานิพนธ์ ซึ่งประกอบไปด้วยดังนี้

บทที่ 1 บทนำ กล่าวถึงความเป็นมาในการสร้างโครงงานชิ้นนี้

บทที่ 2 ความรู้ทางทฤษฎีและความรู้พื้นฐานของ A* Algorithm และการเคลื่อนที่แบบต่างๆ

บทที่ 3 การออกแบบและการทดลองของแผนที่ การพัฒนา A* Algorithm ลำดับการทำงานของโปรแกรม แผนภาพการทำงานของหุ่นยนต์นำทาง รวมทั้งอุปกรณ์ต่างๆ

บทที่ 4 การทดลองและผลการทดลองในโปรแกรม Matlab การทดลอง HMC5883L 3-Axis Digital Compass Module และการทดลองการเคลื่อนที่ของหุ่นยนต์นำทาง

บทที่ 5 สรุปการดำเนินงาน ปัญหาและอุปสรรค แนวทางในการแก้ไข

บทที่ 2

ทฤษฎีและความรู้พื้นฐาน

ในปัจจุบันมีแนวคิดการสร้างต้นแบบหุ่นยนต์อยู่หลายวิธี เช่น หุ่นยนต์เดินตามเส้นซึ่งเป็นที่รู้จักอย่างแพร่หลาย คือ การนำทางหุ่นยนต์ด้วยการตรวจจับเส้นที่มีการวางไว้ โดยใช้การทำงานของเซนเซอร์ เพื่อตรวจจับการสะท้อนของรังสีอินฟาเรดทำให้หุ่นยนต์สามารถเดินไปด้วยการตรวจจับเส้นไปเรื่อยๆ จนกว่าจะถึงจุดเป้าหมาย วิธีต่อมาคือการนำทางหุ่นยนต์โดยใช้ A* Algorithm เป็น Algorithm ที่หาเส้นทางการเดินระหว่างตำแหน่งสองตำแหน่งบนแผนที่ โดย A* Algorithm จะหาเส้นทางที่สั้นที่สุดเพื่อให้ไปถึงตำแหน่งที่ต้องการเพียงแค่มีแผนที่ และกำหนดจุดเป้าหมายให้กับหุ่นยนต์

ในวิธีการนำทางหุ่นยนต์เดินตามเส้น เมื่อนำไปประยุกต์ใช้งานกับสถานที่จริงจำเป็นจะต้องปรับสภาพแวดล้อมโดยติดตั้งเส้น เพื่อให้เข้ากับหลักการทำงาน เช่น หุ่น MK Robot มีการเคลื่อนที่แบบเดินตามเส้นโดยใช้ตัวตรวจจับอัลตราโซนิก เพื่อใช้ในการตรวจจับเส้นซึ่งสถานที่นั้นต้องมีเส้นสีขาว หรือสติกเกอร์เมื่อมีการป้อนข้อมูล หุ่นยนต์จะทำการค้นหาเส้นทางการเคลื่อนที่จากแผนผังไปยังจุดหมายที่ได้รับ ที่เราได้ทำการป้อนไว้เป็นฐานข้อมูลล่วงหน้าอยู่แล้ว

ส่วนวิธีการนำทางหุ่นยนต์โดยใช้ A* Algorithm เมื่อนำไปประยุกต์ใช้กับสถานที่จริงอาจไม่ต้องเปลี่ยนแปลงสภาพแวดล้อมมากนัก สำหรับหุ่นยนต์ Pickitbot นั้นเลือกใช้การนำทางไปยังเป้าหมายด้วยวิธีใช้ A* Algorithm จะเป็นการเคลื่อนที่อย่างอิสระโดยจะเลือกเส้นทางที่เหมาะสมเองมีการรับจุดเริ่มต้นกับจุดเป้าหมายแล้วจะทำการค้นหาทางที่ใกล้ที่สุดในแผนที่ ที่กำหนดไว้

วิธีการนำทางของหุ่นยนต์เดินตามเส้น และวิธีการนำทางหุ่นยนต์โดยใช้ A* Algorithm สามารถอ้างอิงจากตาราง 2.1

ตารางที่ 2.1 Navigation System ของหุ่นเดินตามเส้น และหุ่นที่ใช้วิธี A* Algorithm

หัวข้อ	หุ่นยนต์เดินตามเส้น	หุ่นยนต์ที่ใช้ A* Algorithm
การเดินทางไปถึงจุดมุ่งหมาย	✓	✓
เลือกเส้นทางเดินอย่างอิสระ		✓
ฐานข้อมูลแผนผังของสถานที่	✓	✓
ง่ายต่อการปรับปรุงสถานที่เมื่อนำไปใช้จริง		✓

ในบทนี้นำเสนอในส่วนของ Navigation เมื่อทำการศึกษาการเดินตามเส้นกับการใช้วิธีของ A* Algorithm แล้ว จะเห็นได้ว่าวิธี A* Algorithm สามารถทำได้ดีกว่าเพียงอาจจะมีความซับซ้อนในเรื่องของ Software ก็ตาม แต่ก็ยังเป็นข้อดีอีกประการหนึ่ง เพื่อที่จะลดปัญญาต่างๆ ได้ ที่หุ่นยนต์เดินตามเส้นนั้นทำไม่ได้

โครงการพัฒนาหุ่นต้นแบบเพื่อช่วยผู้บริโภคซื้อสินค้าในร้านค้าซุปเปอร์มาร์เก็ต แบ่งออกเป็น 2 กลุ่มหุ่นยนต์ในส่วนของการเคลื่อนที่ (Navigation System Group) และหุ่นยนต์ในส่วนของแขนกล Manipulation Group โครงงานนี้จะศึกษาเกี่ยวกับในส่วนของการเคลื่อนที่ (Navigation System) เพื่อที่จะหาเส้นทางการเดินจากจุดเริ่มต้นไปยังจุดสิ้นสุด และสามารถหลบหลีกสิ่งกีดขวางในแผนที่ได้ โดยใช้วิธี A* Algorithm

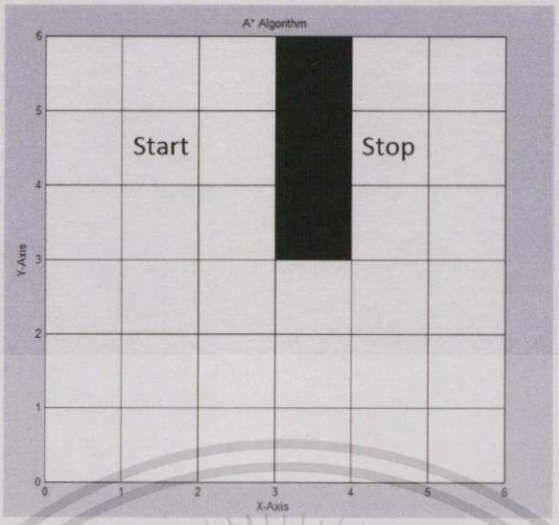
2.1 ทฤษฎี A* Algorithm

A* Algorithm เป็นวิธีการที่ใช้ในการหาเส้นทาง และการหากราฟซึ่งเป็นกระบวนการในการหาเส้นทางระหว่างจุด (เรียกจุดดังกล่าวว่า โหนด (Node)) ขั้นตอนวิธีนี้มีประสิทธิภาพ และความแม่นยำสูงจึงมีการนำไปใช้งานอย่างแพร่หลาย โดยใช้สมการในการเคลื่อนที่จากจุดเริ่มต้นไปยังจุดหมายปลายทาง

$$F = G + H$$

เมื่อ G = ค่าพลังงานที่ใช้ในการเคลื่อนที่จากจุดเริ่มต้นไปยังจุดที่สนใจ
H = ค่าพลังงานที่ใช้ในการเคลื่อนที่จากจุดที่สนใจไปยังจุดเป้าหมาย

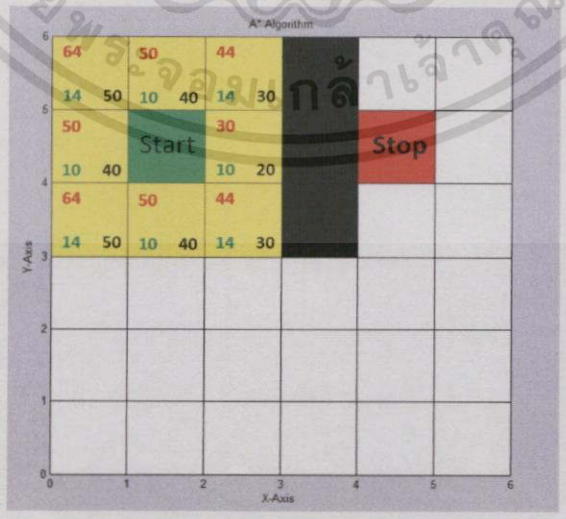
ขั้นตอนแรกเริ่มต้นกระบวนการโดยการกำหนดจุดเริ่มต้น (Start) และจุดเป้าหมาย (Stop) โดยรวมถึงสิ่งกีดขวางที่ขวางทางอยู่ด้วย



รูปที่ 2.1 แสดงจุดเริ่มต้นและจุดเป้าหมาย

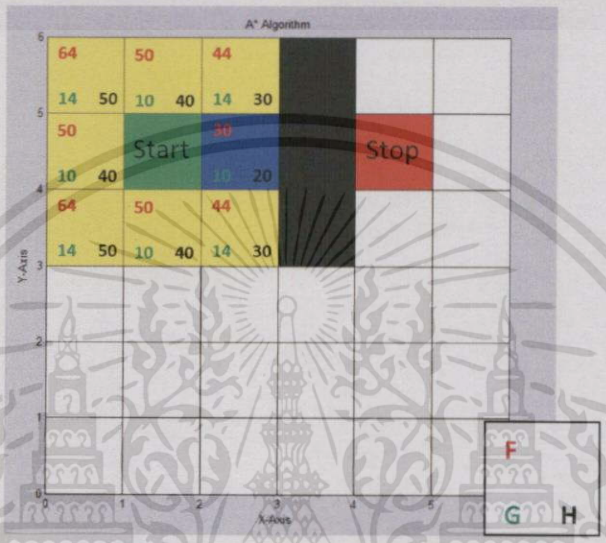
ในส่วนนี้เราจะให้ค่า G ในส่วนของช่องแนวตั้ง และแนวนอนมีค่าเท่ากับ 10 และมีค่าเท่ากับ 14 ในช่องทแยงมุม ในส่วนของค่า H เราจะคิดค่าจากจุดที่สนใจไปยังจุดเป้าหมายโดยไม่มีการเคลื่อนที่ในแนวทแยงมุม โดยรวมถึงสิ่งกีดขวางที่ขวางทางอยู่ด้วย โดยนำค่า G และ H ของแต่ละช่องนำมาเป็นผลรวม หรือค่า F ให้ในแต่ละช่อง

กระบวนการโดยการนำจุดเริ่มต้นมา (Start) ทำการเพิ่มใน Open List เพื่อทำการพิจารณาขั้นตอนต่อมาทำการหาช่องที่จุดเริ่มต้นของเราสามารถทำการเคลื่อนที่ไปหาได้ หรือก็คือช่องของกริดที่อยู่ติดกับช่องของจุดเริ่มต้น โดยที่เราจะไม่สนใจในกรณีที่ช่องกริดนั้นเป็นสิ่งกีดขวาง ทำการเพิ่มช่องกริดที่อยู่รอบด้านเพิ่มไปใน Open List จากนั้นทำการตั้งค่า Start ให้เป็น Parent Square ซึ่งตำแหน่ง Parent Square จะเป็นตำแหน่งที่สำคัญที่ช่วยในการบอกเส้นทางจากจุดเริ่มต้นไปยังจุดเป้าหมายของเราต่อไป หลังจากนั้นทำการเปลี่ยนตำแหน่งของ Start จาก Open List ไปยัง Closed List ดังรูปที่ 2.2



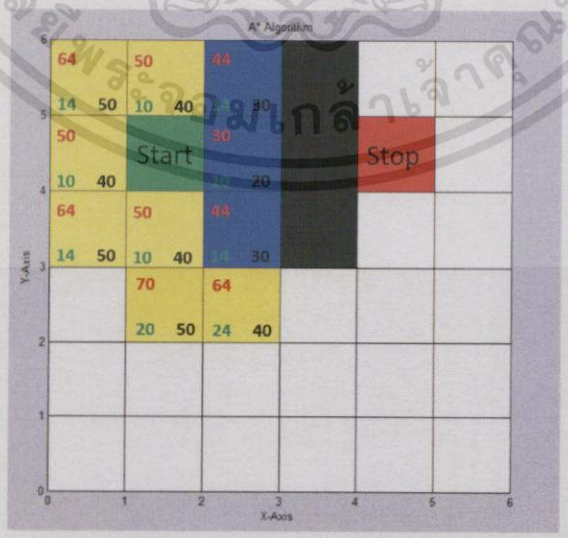
รูปที่ 2.2 แสดงค่า F, G และ H ที่ได้จากการคำนวณ

ขั้นตอนต่อมาจะทำการเลือกช่องที่อยู่ติดกับ Start ซึ่งเราได้ทำการจัดเก็บไว้ใน Open List มาทำการหาช่องกริดที่มีค่า F น้อยที่สุด เมื่อเราทำการเลือกช่องกริดที่มีค่า F น้อยที่สุดได้แล้วนั้น ทำการเปลี่ยนตำแหน่งจาก Open list ไปเก็บไว้ใน Close List จากนั้นทำการตรวจสอบตำแหน่งกริดที่มีตำแหน่งรอบกับจุดที่เราได้เลือก ถ้าหากตำแหน่งช่องกริดที่มีตำแหน่งรอบกับจุดที่เราได้เลือกยังไม่มีอยู่ใน Open List ทำการเพิ่มตำแหน่งของจุดเหล่านั้นเพิ่มไปยัง Open List โดยที่ไม่ต้องสนใจกริดที่เป็นตำแหน่งสิ่งกีดขวาง จากนั้นทำการตั้งค่าจุดที่เลือกให้เป็น Parent Square ดังรูปที่ 2.3



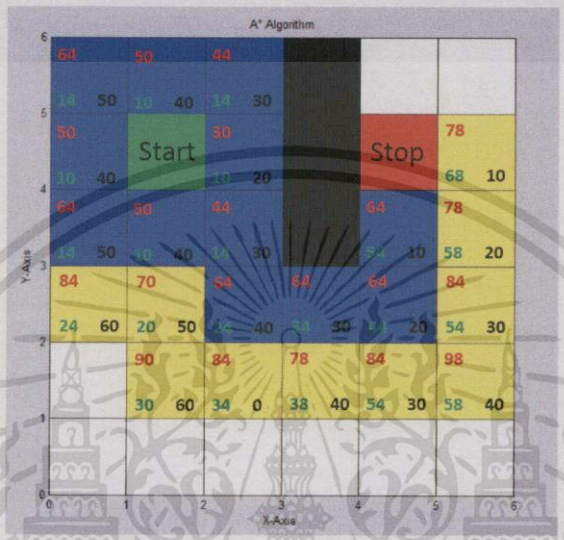
รูปที่ 2.3 แสดงการเลือกช่องกริดที่มีค่า F น้อยที่สุด

ถ้าหากกริดที่มีตำแหน่งรอบกับจุดที่เราได้เลือกมีอยู่ใน Open List อยู่แล้ว ทำการตรวจสอบเส้นทางการเดินที่ดีที่สุด โดยทำการตรวจสอบจากค่า G หากค่า G ของเส้นทางใหม่มีค่าต่ำกว่า ทำการเปลี่ยน Parent Square จากจุดเก่ามาเป็นจุดใหม่ ดังรูปที่ 2.4

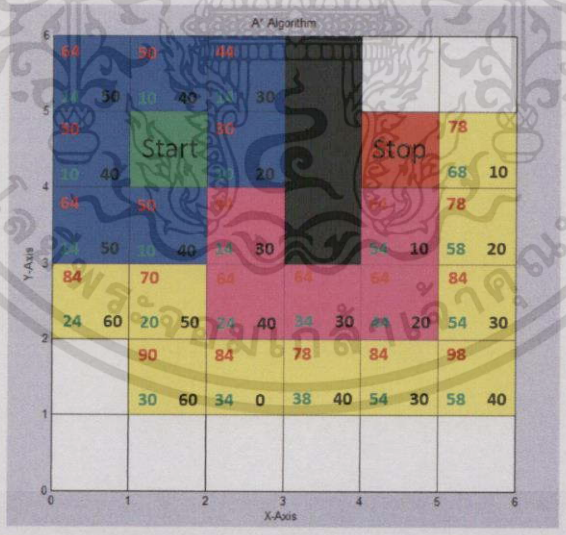


รูปที่ 2.4 แสดงการตรวจสอบและเลือกค่า G ของเส้นทางใหม่ที่มีค่าต่ำกว่า

ในกรณีที่เราทำการตรวจสอบกริดรอบข้างแล้วเราพบว่าเป็นสิ่งกีดขวาง เราจะไม่ทำการเดินเป็นมุมทะแยง แต่จะใช้การเดินในลักษณะแนวตั้งและแนวนอนเท่านั้น แล้วจะหยุดการทำงานก็ต่อเมื่อถึงเป้าหมายแล้วทำการเพิ่มจุดเป้าหมายเข้าไปใน Closed List ทำการเก็บข้อมูลเส้นทางการเดิน ดังรูปที่ 2.5 จากนั้นทำการเดินย้อนกลับจากช่องเป้าหมายย้อนกลับไปแต่ละช่องซึ่งเป็น Parent Square จนกระทั่งย้อนกลับไปยังจุดเริ่มต้น ดังรูปที่ 2.6



รูปที่ 2.5 แสดงเส้นทางเดินไปยังจุดเป้าหมายในลักษณะแนวนอน และแนวตั้งเท่านั้น



รูปที่ 2.6 แสดงการเดินย้อนกลับจากจุดเป้าหมายจนถึงจุดเริ่มต้น

2.2 การเคลื่อนที่ในแบบต่างๆ

ในการเขียนโปรแกรมจำลองการเคลื่อนที่ของตัวรถหุ่นยนต์นำทางในโปรแกรม Matlab รวมถึงการเคลื่อนที่ของตัวรถหุ่นยนต์นำทาง เราจำเป็นที่จะต้องมีการศึกษาถึงลักษณะการเคลื่อนที่ในแบบต่างๆ เพื่อที่จะนำไปใช้ได้อย่างถูกต้อง โดยในที่นี้ เราจะศึกษาถึงการเคลื่อนที่ในแนวเส้นตรงและการเคลื่อนที่เชิงมุม

2.2.1 การเคลื่อนที่ในแนวเส้นตรง

ระยะทาง (Distance) คือ ความยาวตามเส้นทางที่ตัวหุ่นยนต์เคลื่อนที่เป็นปริมาณสเกลลาร์ คือมีแต่ขนาดอย่างเดียวมีหน่วยเป็นเมตร โดยทั่วไปเราใช้สัญลักษณ์ s

อัตราเร็ว (Speed) คือ ระยะทางที่วัตถุเคลื่อนที่ได้ในหนึ่งหน่วยเวลา หรืออัตราการเปลี่ยนระยะทาง หน่วยในระบบเอสไอ มีหน่วยเป็นเมตร/วินาที

$$v = \frac{\Delta s}{\Delta t}$$

ความเร่ง (Acceleration) คือ ความเร็วที่เปลี่ยนแปลงไปในหนึ่งหน่วยเวลา หรืออัตราการเปลี่ยนแปลงความเร็ว มีหน่วยเป็นเมตร/วินาที^{2}

$$a = \frac{\Delta v}{\Delta t}$$

สมการการเคลื่อนที่ในแนวเส้นตรง

$$v = v_0 + at$$

$$\Delta x = \left(\frac{v + v_0}{2}\right)t$$

$$\Delta x = v_0 t + \frac{1}{2}at^2$$

$$v^2 = v_0^2 + 2a\Delta x$$

เมื่อ v_0 = ความเร็วเริ่มต้น $\left(\frac{m}{s}\right)$

v = ความเร็วตอนปลาย $\left(\frac{m}{s}\right)$

x = ระยะทาง (m)

a = ความเร่ง $\left(\frac{m}{s^2}\right)$

2.2.2 การเคลื่อนที่เชิงมุม

การกระจัดเชิงมุม (Angular Displacement) คือมุมที่กวาดไปในระนาบของการเคลื่อนที่มีหน่วยเป็นเรเดียน

$$\theta = \frac{s}{r}$$

ความเร็วเชิงมุม (Angular Speed) คือการกระจัดเชิงมุมที่เปลี่ยนไปในหนึ่งหน่วยเวลา หน่วยในระบบเอสไอ มีหน่วยเป็นเรเดียน/วินาที

$$\omega = \frac{\Delta\theta}{\Delta t}$$

ความเร่งเชิงมุม (Angular Acceleration) คือความเร็วเชิงมุมที่เปลี่ยนไปในหนึ่งหน่วยเวลา มีหน่วยเป็นเรเดียน/วินาที^{2}

$$\alpha = \frac{\Delta\omega}{\Delta t}$$

สมการการเคลื่อนที่เชิงมุม

$$\omega = \omega_0 + \alpha t$$

$$\Delta\theta = \left(\frac{\omega + \omega_0}{2}\right)t$$

$$\Delta\theta = \omega_0 t + \frac{1}{2}\alpha t^2$$

$$\omega^2 = \omega_0^2 + 2\alpha\Delta\theta$$

เมื่อ ω_0 = ความเร็วเริ่มต้น $\left(\frac{\text{rad}}{\text{s}}\right)$

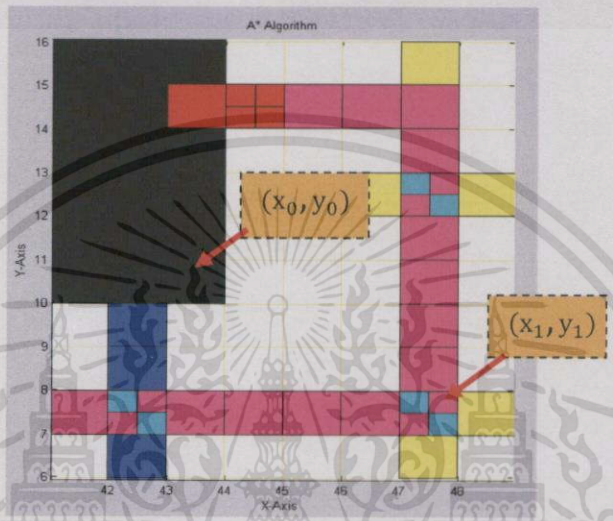
α = ความเร็วตอนปลาย $\left(\frac{\text{rad}}{\text{s}}\right)$

θ = ระยะทาง (radian)

α = ความเร่ง $\left(\frac{\text{rad}}{\text{s}^2}\right)$

2.3 รัศมีในการเลี้ยว

เมื่อทำการหาเส้นทางในการเดินจากจุดเริ่มต้นถึงจุดเป้าหมายแล้ว ทำการคำนวณหาขนาดมุมเลี้ยวที่เหมาะสม เพื่อป้องกันการที่หุ่นยนต์เข้าใกล้สิ่งกีดขวางมากเกินไปในขณะที่ทำการเลี้ยว โดยจากรูปที่ 2.7 แสดงตำแหน่งของสิ่งกีดขวางบริเวณมุมเลี้ยว (x_0, y_0) และตำแหน่งของเส้นทางเดินที่เป็นมุมเลี้ยว (x_1, y_1)



รูปที่ 2.7 แสดงตำแหน่งสิ่งกีดขวางและมุมเลี้ยว

เมื่อทราบตำแหน่งของสิ่งกีดขวางและมุมเลี้ยวแล้ว สามารถหาระยะห่างของทั้ง 2 ตำแหน่งได้โดยใช้สมการ

$$L = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}$$

เมื่อ L = ระยะห่างระหว่างสิ่งกีดขวางและมุมเลี้ยว

$x_0 - x_1$ = ผลต่างของตำแหน่งในแนวแกน x

$y_0 - y_1$ = ผลต่างของตำแหน่งในแนวแกน y

จากนั้นนำระยะห่างระหว่างสิ่งกีดขวางและมุมเลี้ยว มาคำนวณกับความกว้างของหุ่นยนต์เพื่อให้ได้ขนาดของมุมโดยประมาณ เพื่อให้หุ่นยนต์สามารถเลี้ยวได้ห่างจากสิ่งกีดขวางมากพอที่จะไม่เกิดโอกาสเสี่ยงในการชนกับสิ่งกีดขวาง โดยใช้สมการ

$$R = L - D$$

เมื่อ R = ขนาดของมุมเลี้ยว

L = ระยะห่างระหว่างสิ่งกีดขวางและมุมเลี้ยว

D = ความกว้างของตัวหุ่นยนต์

นำขนาดมุมเลี้ยวที่คำนวณได้และตำแหน่งของมุมที่ทำการเลี้ยว มาหาตำแหน่งที่ทำการเริ่มเลี้ยว และตำแหน่งหยุดเลี้ยวที่ทำการแปรผันตามสิ่งกีดขวางเพื่อลดโอกาสที่หุ่นยนต์เข้าใกล้สิ่งกีดขวางมากเกินไป และนำข้อมูลการเลี้ยวที่ได้รวมกับเส้นทางการเดินทางตรงได้เส้นทางการเดินจากจุดเริ่มต้นถึงจุดเป้าหมาย

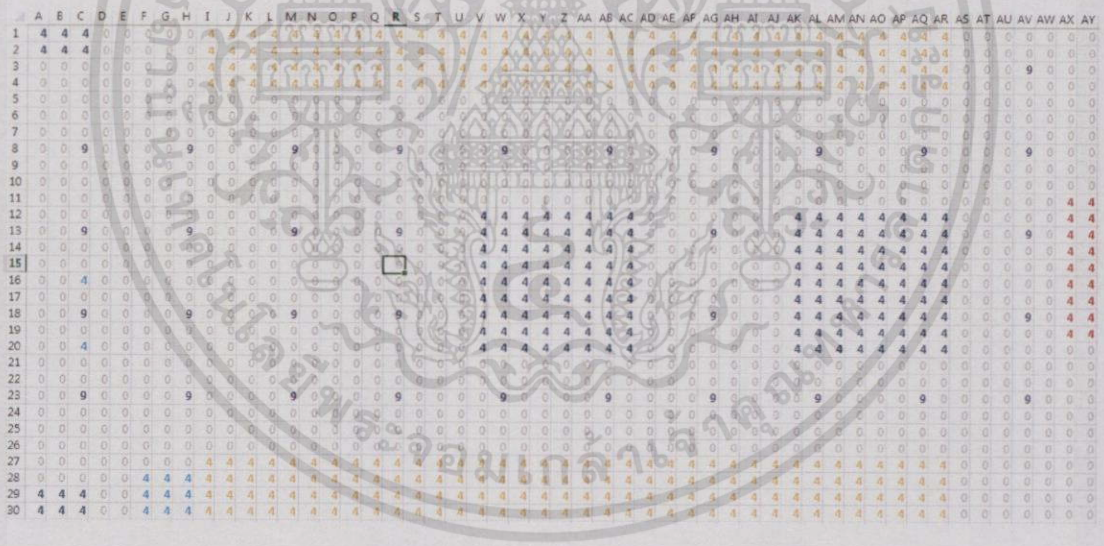
บทที่ 3
การออกแบบและการทดลอง

3.1 การออกแบบโปรแกรมจำลองการเคลื่อนที่

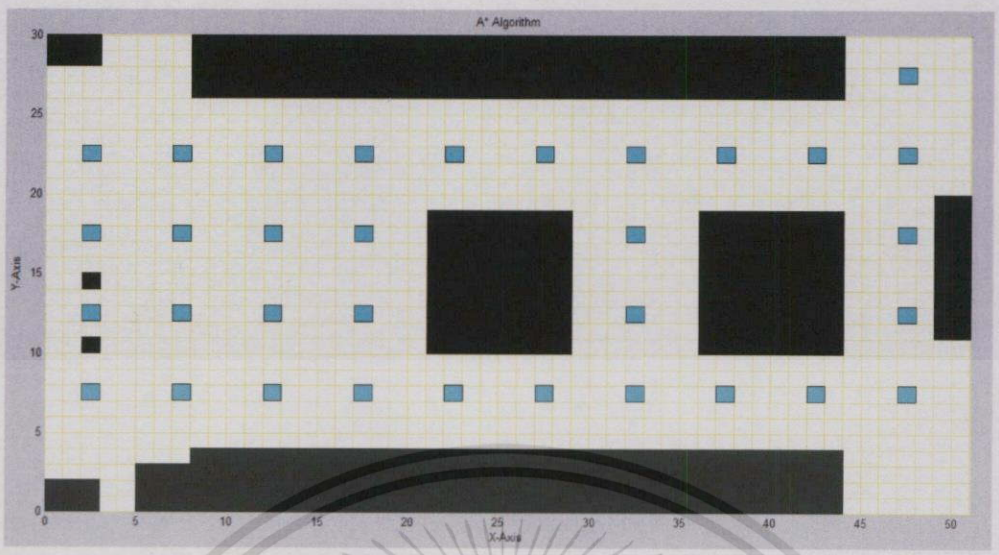
ในการออกแบบโปรแกรมจำลองการเคลื่อนที่ของหุ่นยนต์นำทาง ประกอบด้วยการสร้างแผนที่จำลองด้วยโปรแกรม Microsoft Excel การพัฒนาเส้นทางที่ได้จาก A* Algorithm และแสดงลำดับการทำงานด้วยโปรแกรม Matlab

3.1.1 แผนที่

การจำลองแผนที่เพื่อทำการหาเส้นทางการเดินโดยการกำหนดค่าด้วยโปรแกรม Microsoft Excel ดังรูปที่ 3.1 และนำเข้าสู่โปรแกรม Matlab ซึ่งเก็บข้อมูลในรูปแบบเมตริก และแสดงหน้าต่างแผนที่จำลองด้วยโปรแกรม Matlab ดังรูปที่ 3.2



รูปที่ 3.1 แสดงการกำหนดค่าแผนที่ด้วยโปรแกรม Microsoft Excel



รูปที่ 3.2 ตัวอย่างหน้าต่างแผนที่จำลองด้วยโปรแกรม Matlab

แผนที่ประกอบด้วย

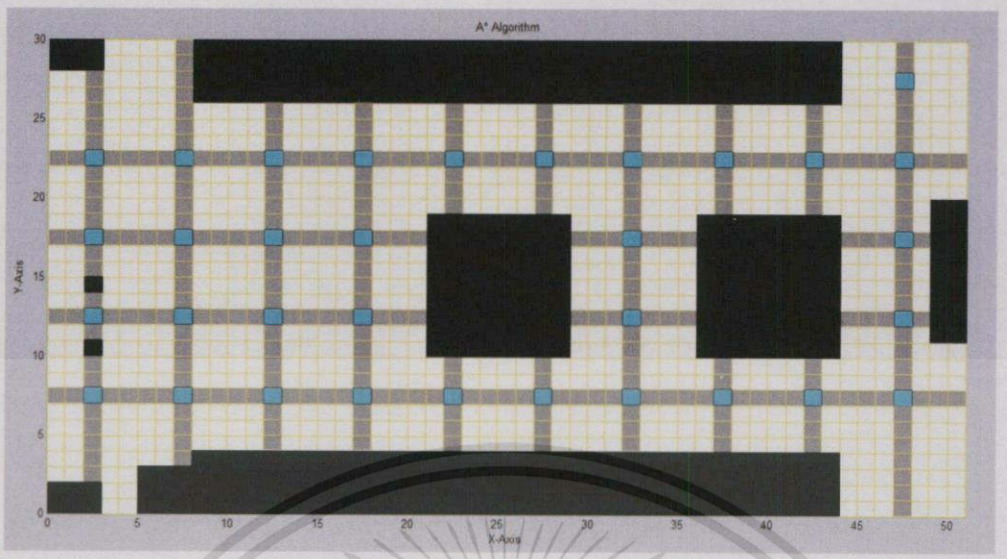
1. พื้นที่สีขาว คือ พื้นที่ที่สามารถเป็นเส้นทางการเดินไปยังจุดหมายได้ โดยกำหนดค่าเป็น 0

2. พื้นที่สีดำ คือ การจำลองสิ่งกีดขวางที่มีอยู่ในสถานที่จริง โดยกำหนดค่าเป็น 4

3. พื้นที่สีฟ้า คือ จุดตรวจสอบ (Check Point) โดยกำหนดค่าเป็น 9 และมีระยะห่างจากจุดตรวจสอบหนึ่งไปอีกจุดตรวจสอบเท่ากัน

4. ความกว้างและความยาวของกริดมีขนาดเท่ากับ 20 cm

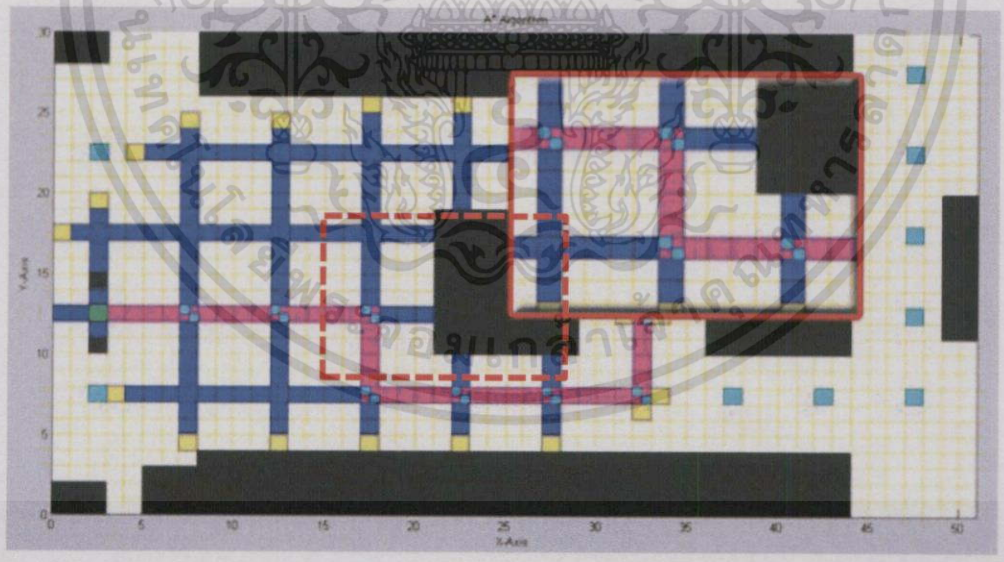
จากนั้นทำการจำลองสิ่งกีดขวางเสมือนขึ้นเพื่อควบคุมเส้นทางการเดินให้ผ่านจุดตรวจสอบเสมอ โดยการเปลี่ยนค่าพื้นที่สีขาวที่ไม่ได้อยู่ในแนวเดียวกับจุดตรวจสอบ (ทั้งแนวตั้งและแนวนอน) จากเดิมกำหนดค่าเป็น 0 เปลี่ยนเป็นกำหนดค่าเป็น 4 ดังรูปที่ 3.3 เพื่อนำไปใช้สำหรับการหาเส้นทางการเดินจาก A* Algorithm ต่อไป



รูปที่ 3.3 แสดงผลการควบคุมเส้นทางการเดินด้วยพื้นที่สีเทา

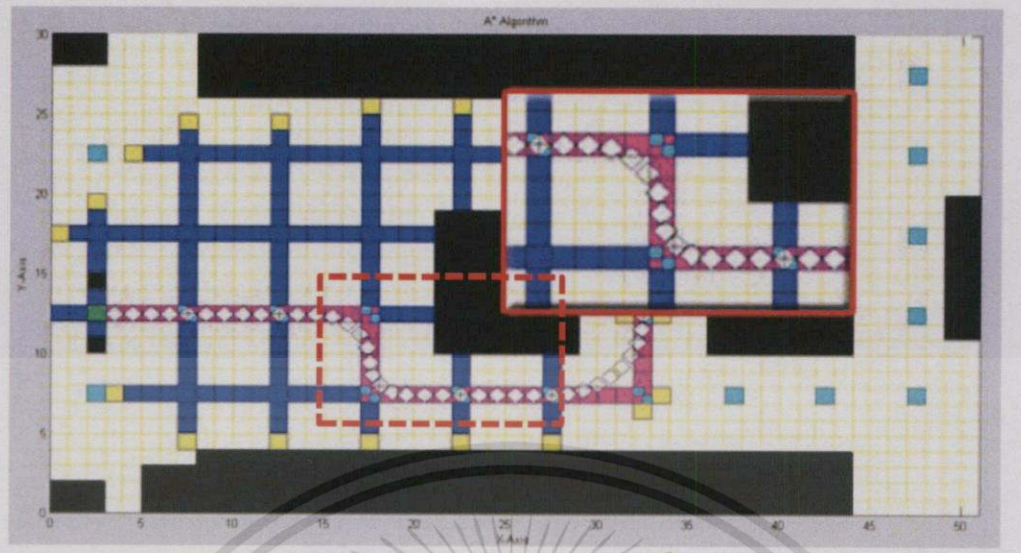
3.1.2 การพัฒนา A* Algorithm

เมื่อได้เส้นทางการเดินจาก A* Algorithm แล้วจะพบว่ามีเส้นทางการเดินแบบแนวตรง และมุมในการเลี้ยวแบบตั้งฉาก โดยจะผ่านจุดตรวจที่กำหนดไว้ในแผนที่ แสดงดังรูปที่ 3.4



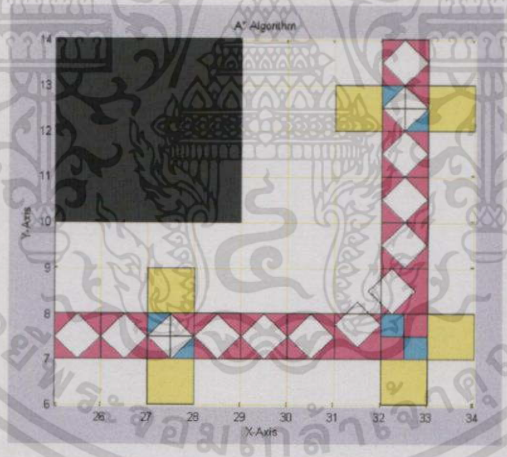
รูปที่ 3.4 แสดงมุมการเลี้ยวแบบตั้งฉาก

จึงนำมาปรับหาเส้นทางการเดินในกรณีที่มีเส้นทางมุมการเลี้ยวแบบตั้งฉาก ให้เปลี่ยนเป็นเส้นทางที่มีมุมในการเลี้ยวที่มีความโค้งมากขึ้น ด้วยการเพิ่มโปรแกรมกระบวนการหาพิกัดเพื่อปรับแต่งผลจากกระบวนการคิดเดิม ดังรูปที่ 3.5



รูปที่ 3.5 แสดงมุมการเลี้ยวที่มีความโค้งมากขึ้น

คำนวณขนาดรัศมีการเลี้ยวให้แปรผันตามระยะห่างระหว่างมุมเลี้ยวและสิ่งกีดขวาง เพื่อแก้ปัญหาหุ่นยนต์เข้าใกล้สิ่งกีดขวางมากเกินไป ดังรูปที่ 3.6 ซึ่งกรณีนี้ถ้าหุ่นยนต์เข้าใกล้สิ่งกีดขวางอาจทำให้เกิดความเสียหายต่อหุ่นยนต์ได้



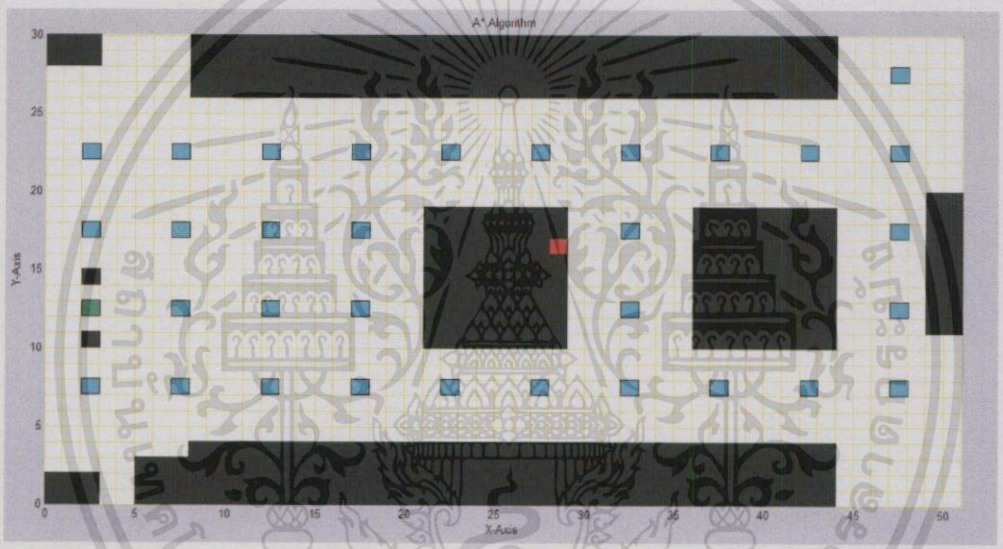
รูปที่ 3.6 แสดงมุมเลี้ยวที่แปรผันตามสิ่งกีดขวาง

3.1.3 ลำดับการทำงานของโปรแกรม

1. สร้างแผนที่จำลองด้วยโปรแกรม Microsoft Excel

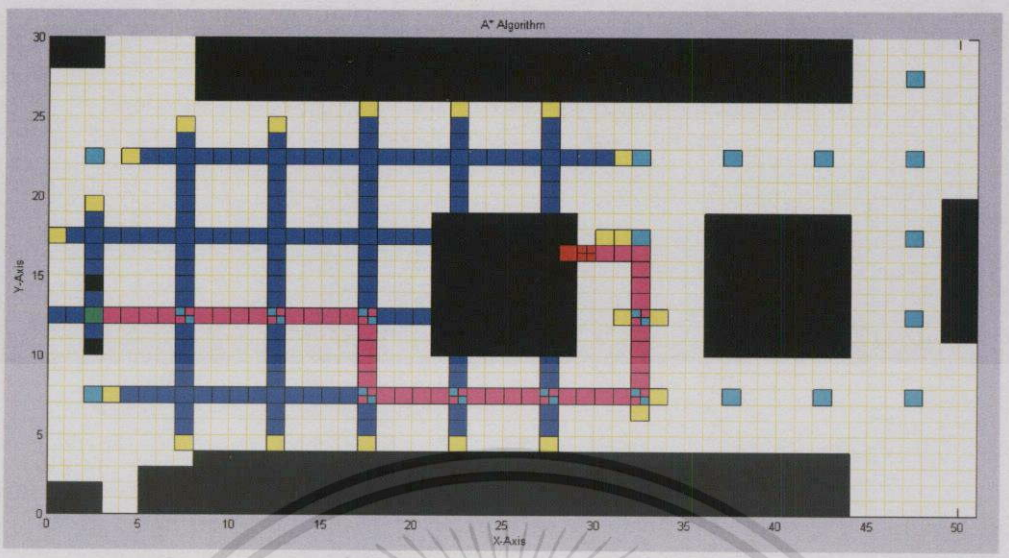
2. นำเข้าแผนที่จากโปรแกรม Microsoft Excel เข้าสู่โปรแกรม Matlab

3. กำหนดจุดเริ่มต้น และจุดเป้าหมายในขอบเขตของแผนที่ที่สร้างไว้ โดยมีข้อจำกัดคือ จุดเริ่มต้นมีการจำกัดทิศทางการเดินได้เฉพาะทิศทางด้านหน้าของหุ่นยนต์ และจุดเป้าหมายจะต้องอยู่บนแนวที่ควบคุมเส้นทางการเดินไว้ หรือกำหนดให้อยู่ในตำแหน่งเดียวกับสิ่งกีดขวางที่มีพื้นที่ติดกับแนวเส้นทางการเดิน ดังรูปที่ 3.7



รูปที่ 3.7 แสดงตัวอย่างการกำหนดจุดเริ่มต้น และจุดเป้าหมาย

4. ทำการหาเส้นทางการเดินจากจุดเริ่มต้นไปยังจุดเป้าหมายด้วย A* Algorithm และเก็บข้อมูลเส้นทางการเดิน และจุดตรวจสอบที่เป็นเส้นทางในการเดินเพื่อใช้ในขั้นตอนต่อไป ดังรูปที่ 3.8



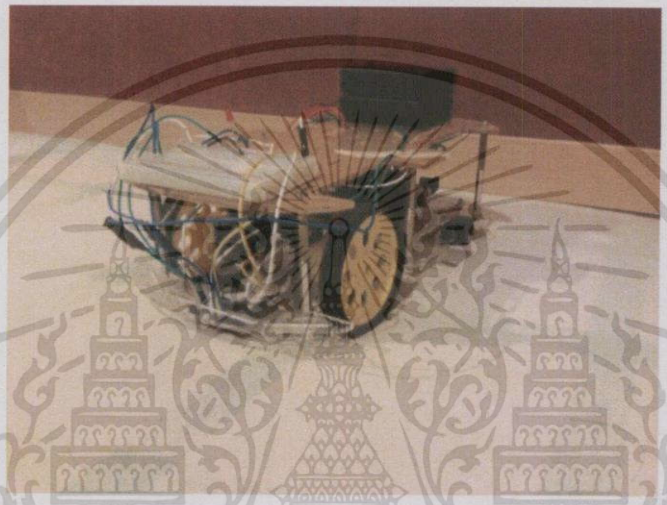
รูปที่ 3.8 แสดงตัวอย่างเส้นทางที่ได้จาก A* Algorithm

5. นำค่าจุดตรวจสอบที่เป็นเส้นทางการเดินจากจุดเริ่มต้นไปยังจุดเป้าหมายมาพัฒนาต่อ โดยการปรับมุมการเลี้ยวแบบตั้งฉาก ให้เป็นมุมการเลี้ยวที่มีความโค้งมากขึ้น

6. ทำการรวมเส้นทางในแนวตรง และการเลี้ยวที่ได้จากขั้นตอนที่ 5 ให้เป็นเส้นทางใหม่ที่ต่างไปจากข้อมูลที่ได้จาก A* Algorithm

3.2 หุ่นยนต์นำทาง

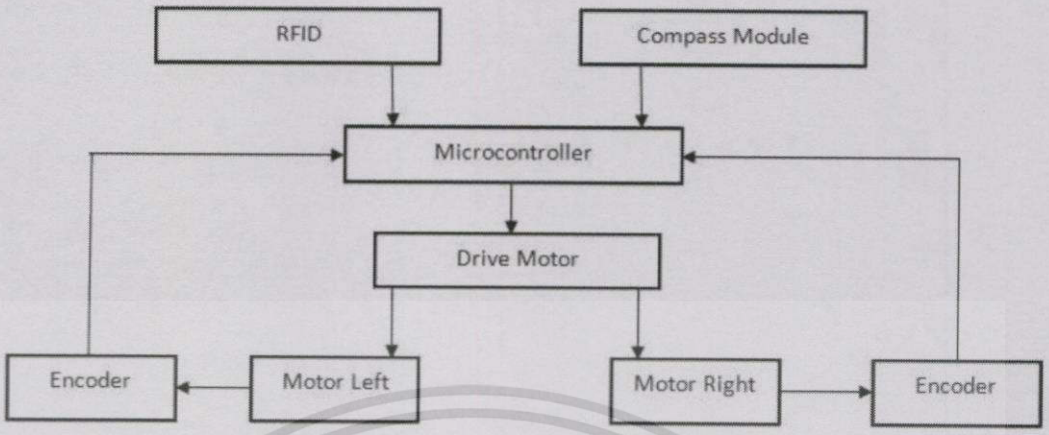
ตัวโครงสร้างของตัวรถหุ่นยนต์นำทางนั้นเป็นแบบสำเร็จรูปสร้างขึ้นจากอะคลีลิก [6] โดยดัดแปลงล้อที่ใช้ในการเคลื่อนที่ทั้งหมดสี่ล้อ สองล้อด้านหน้าของตัวรถติดตั้งอยู่ร่วมกับ DC Motors เพื่อใช้ในการขับเคลื่อนตัวรถหุ่นยนต์นำทาง สองล้อด้านหลังของตัวรถใช้ล้อที่สามารถเคลื่อนที่ได้อย่างอิสระ เพื่อที่จะทำให้มุมเลี้ยวของตัวรถสามารถทำมุมเลี้ยวได้ตามที่ต้องการ ดังแสดงในรูปที่ 3.9



รูปที่ 3.9 โครงสร้างหุ่นยนต์นำทาง

3.3 แผนภาพการทำงานของหุ่นยนต์นำทาง

ในส่วนของตัวรถหุ่นยนต์นำทาง ตัวรถนั้นสร้างขึ้นจากอะคลีลิก จะมีวงจรขับมอเตอร์เพื่อให้รถสามารถเคลื่อนที่ได้ตามที่ต้องการ โดยจะทำงานร่วมกับอุปกรณ์เซนเซอร์ต่างๆ โดยเซนเซอร์ที่ใช้ได้แก่ตัว Encoder ใช้ในการวัดความเร็วของรถที่กำลังเคลื่อนที่ Compass Module ใช้ในการตรวจวัดมุมองศาของตัวรถ เพื่อที่จะบังคับให้ตัวรถเคลื่อนที่ไปยังทิศทางที่ต้องการได้อย่างถูกต้อง และในขณะที่รถกำลังเคลื่อนที่อยู่ในเส้นทางจะมี RFID เมื่อตัวรถหุ่นยนต์นำทางเคลื่อนที่ผ่าน RFID จะมีการสั่งให้หลอดไฟติดเกิดขึ้น เพื่อคอยตรวจสอบว่ารถได้เดินทางมายังเส้นทางที่ถูกต้อง โดยเซนเซอร์ทั้งหมดจะทำงานและส่งค่ามาประมวลผลที่ตัวไมโครคอนโทรลเลอร์ Arduino UNO R3 เพื่อให้ตัวรถหุ่นยนต์นำทางสามารถเคลื่อนที่จากจุดเริ่มต้นไปยังจุดเป้าหมายปลายทางได้อย่างถูกต้อง แสดงดังรูปที่ 3.10



รูปที่ 3.10 แผนภาพการทำงานของหุ่นยนต์นำทาง

3.4 อุปกรณ์

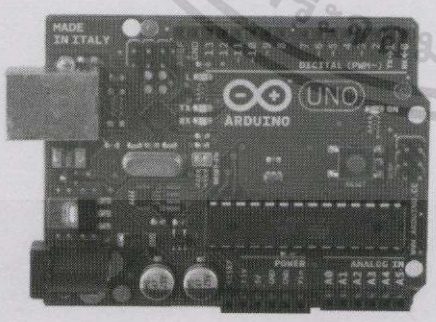
- ตัวเก็บประจุ 1.0 µF
- ตัวต้านทาน 1.0 kΩ
- มอเตอร์กระแสตรงขนาดเล็ก 6.0 V
- Arduino UNO R3
- IC เบอร์ L293D
- Encoder บอร์ด Op-Amp LM-393
- HMC5883L 3-Axis Digital Compass Module
- Regulator เบอร์ LM-7805
- Module RFID RC-522, S50 IC Card, S50 IC Key
- ถ่าน AA 1.2 V 1,900 mA

3.4.1 Arduino UNO R3

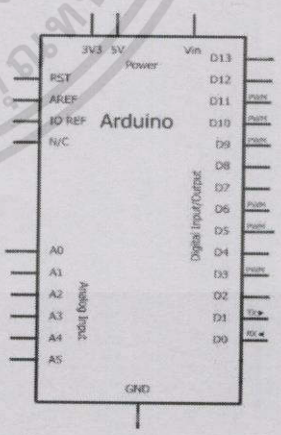
ในการประมวลผลของหุ่นยนต์ช่วยเหลือผู้พิการในห้างสรรพสินค้า เราได้ใช้ตัวไมโครคอนโทรลเลอร์ Arduino UNO R3 ในการประมวลผล โดยจะรับค่าอัตราความเร็วที่ต้องการแล้วนำไปบังคับมอเตอร์เพื่อให้ได้ความเร็วตามที่ต้องการ และรับค่าจาก RFID เพื่อที่จะดูตำแหน่งของหุ่นนำทาง

ตารางที่ 3.1 ตารางรายละเอียดของตัวไมโครคอนโทรลเลอร์

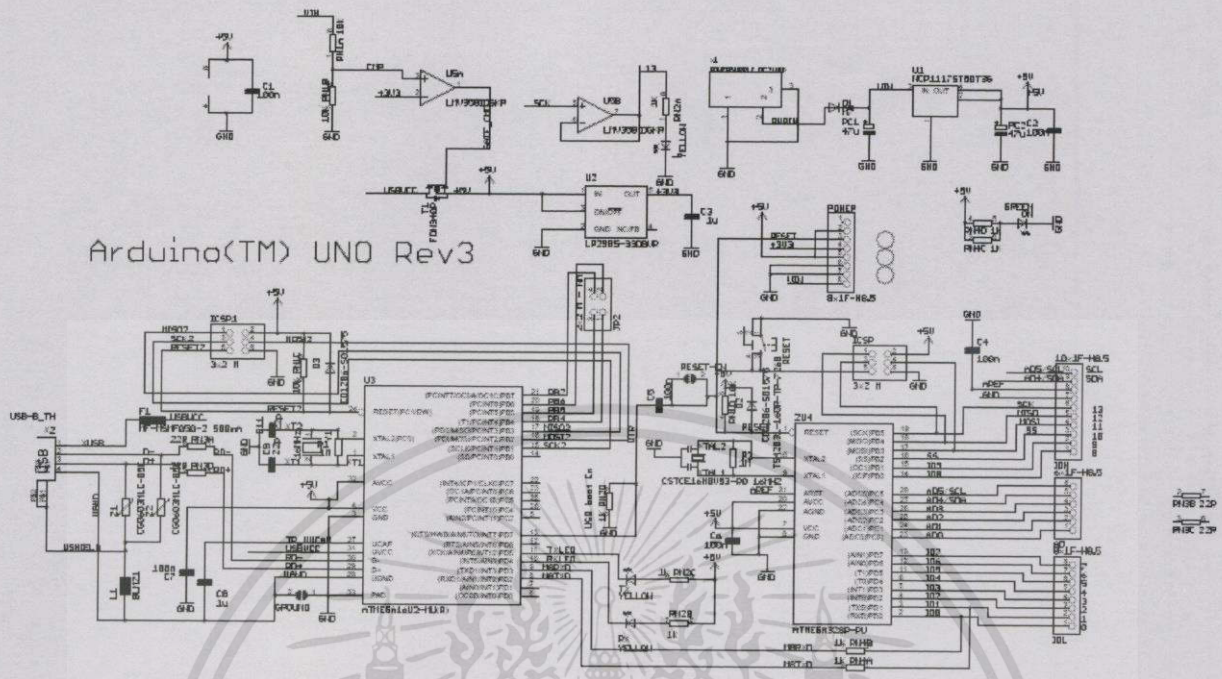
Microcontroller	ATmega328
Operating Voltage	5 V
Input Voltage (recommended)	7-12 V
Input Voltage (limits)	6-20 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by boot loader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz



รูปที่ 3.11 Arduino UNO R3



รูปที่ 3.12 Pin Arduino UNO R3



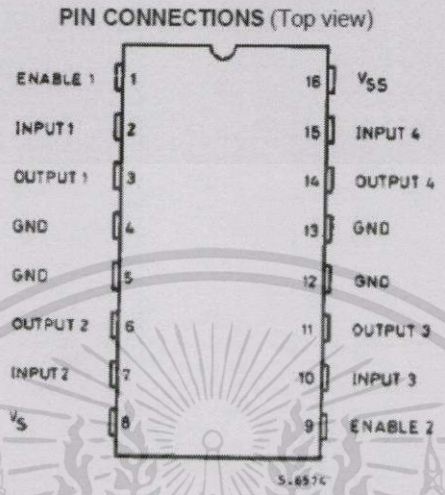
รูปที่ 3.13 แผนภาพของไมโครคอนโทรลเลอร์

3.4.2 DC Motors

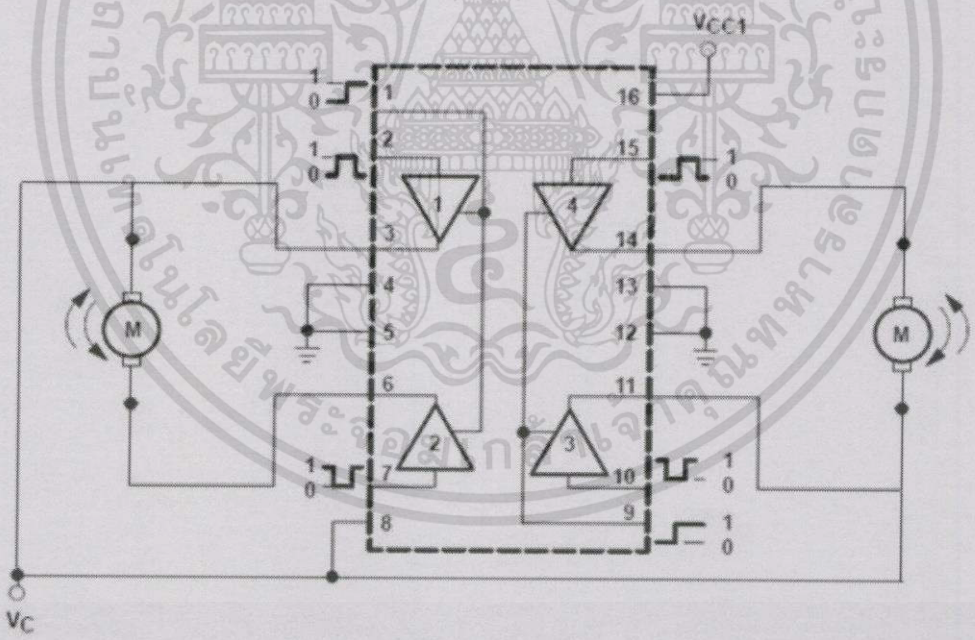
การทำงานมอเตอร์ไฟฟ้ากระแสตรงประกอบด้วย แม่เหล็กถาวร 2 ขั้ว วางอยู่ระหว่างขดลวดตัวนำขดลวดตัวนำจะได้รับแรงดันไฟฟ้ากระแสตรงป้อนให้ในการทำงาน ถ้าแม่เหล็ก 2 ชุด มีขั้วแม่เหล็กเหมือนกัน วางใกล้กันเกิดแรงผลักกัน ทำให้ขดลวดตัวนำหมุนเคลื่อนที่ได้ เมื่อมีกระแสไหลผ่านเข้าไปในมอเตอร์กระแสตรงจะแบ่งออกเป็น 2 ทาง คือ ส่วนที่หนึ่งจะผ่านเข้าไปที่ขดลวดสนามแม่เหล็ก (Field Coil) ทำให้เกิดสนามแม่เหล็กขึ้น และอีกส่วนหนึ่งจะผ่านแปลงถ่าน ซึ่งขดลวดอาร์เมเจอร์ทำให้เกิดสนามแม่เหล็ก ทั้งสองสนามจะเกิดขึ้นขณะเดียวกันตามคุณสมบัติของเส้นแรงแม่เหล็ก และจะไม่มีการติดกัน แต่จะหักล้างและเสริมกัน ซึ่งทำให้เกิดแรงบิตในอาร์เมเจอร์ ทำให้อาร์เมเจอร์หมุนซึ่งในการหมุนนั้นจะเป็นไปตามกฎมือซ้ายของเฟลมมิ่ง (Fleming's Left Hand Rule)

การควบคุมทิศทางการหมุนของ DC Motors วงจรควบคุมนี้จะประกอบไปด้วยลอจิก สาเหตุที่เลือกเพราะสามารถควบคุมการขับมอเตอร์แบบ Step Motor ได้ โดยจะใช้เป็น IC เบอร์ L293D แสดงดังรูปที่ 3.14 เพื่อเป็น Driver ในการขับมอเตอร์ 2 ตัว [7]

เฟืองยังเป็นอุปกรณ์ที่ใช้ในการส่งถ่ายกำลังระหว่างเพลากับเพลา โดยอาศัยฟันและเฟืองทั้งสองขบกัน นอกจากนี้เฟืองยังสามารถใช้ในการทดสอบเพื่อเพิ่ม และลดความเร็วของเฟืองตัวที่ใช้ขับได้



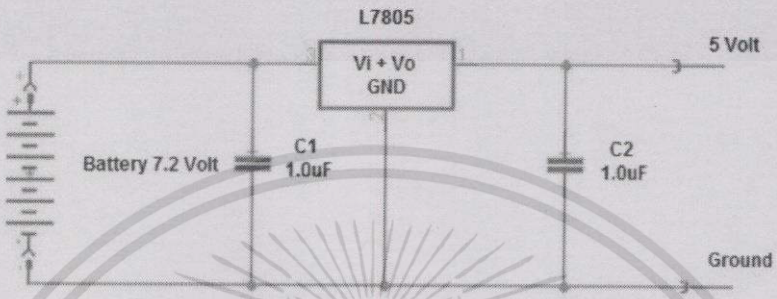
รูปที่ 3.14 IC เบอร์ L293D



รูปที่ 3.15 วงจรควบคุมการขับมอเตอร์

3.4.3 วงจรลดแรงดัน

มอเตอร์ที่ใช้ในโครงงานใช้ไฟแรงดันสูงไม่ได้เราจึงลดแหล่งจ่ายแรงดันให้เหมาะสม [1] ดังรูปที่ 3.16



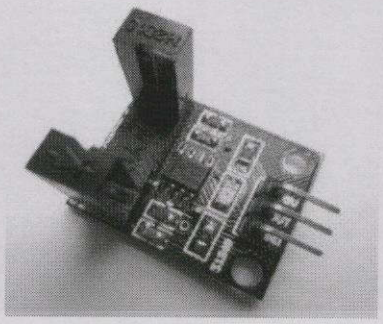
รูปที่ 3.16 วงจรลดแรงดัน 7.2 V เป็น 5.0 V

3.4.5 Encoder

Encoder นั้นจะใช้เพื่อตรวจจับความเร็วของมอเตอร์เพื่อที่จะนำค่าที่ได้มาเข้าสู่ระบบควบคุม เพื่อที่จะทำให้ตัวมอเตอร์มีความเร็วตามที่ต้องการ โดยหลักการทำงานของ Encoder จะทำโดยสร้างสัญญาณพัลส์ที่แปรผันตรงกับการหมุนของตัวมอเตอร์ โดยที่ตัวเซนเซอร์จะทำหน้าที่เป็นอุปกรณ์ตรวจจับส่งลำแสงออกไป โดยที่ตัวของเพลามอเตอร์จะมีจานหมุนที่มีช่องเล็กๆ ติดอยู่ เมื่อลำแสงเคลื่อนที่ผ่านช่องของจานหมุนก็จะทำให้เกิดพัลส์เกิดขึ้น ซึ่งพัลส์ที่ได้จะแสดงถึงความเร็วเท่ากับเพลาของตัวมอเตอร์

ตารางที่ 3.2 คุณสมบัติ Encoder

Name	Encoder Sensor Module LM393
Use the chip	LM-393
Power supply	5 V
Communication modes	Used in Motor Speed Detection, On the Radio-Infrared Head
Measuring	Output valid Signal is Low, Motor Speed
Size	32 mm X Wide 11 mm X High 20 mm
slot width	10 mm

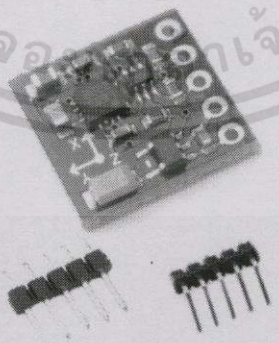


รูปที่ 3.17 Encoder LM393

3.4.6 HMC5883L 3-Axis Digital Compass Module

ตารางที่ 3.3 คุณสมบัติ HMC5883L 3-Axis Digital Compass Module

Name	HMC5883L Module (Three-Axis Magnetic Field Module)
Use the chip	HMC5883L
Power supply	3V ~ 5V
Communication modes	Standard IIC Communication Protocol (3.3V ~ 5V TTL)
Measuring range	± 1.3 - 8 Gauss
Size	14 mm x 13 mm



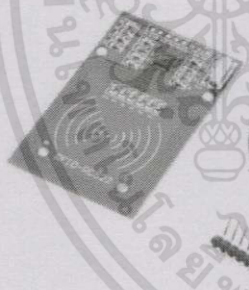
รูปที่ 3.18 HMC5883L 3-Axis Digital Compass Module

ตารางที่ 3.4 การเชื่อมต่อ Arduino UNO R3 เข้ากับ Compass Module

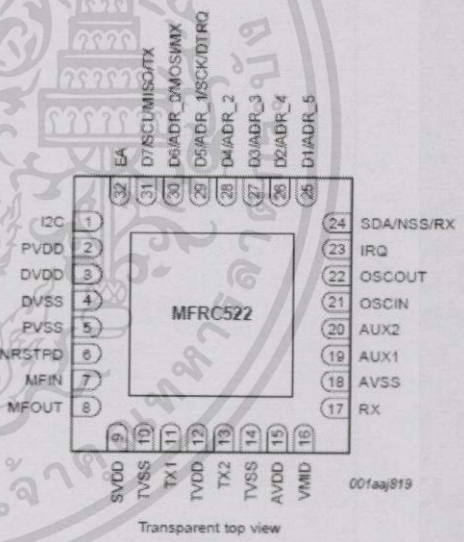
Arduino UNO R3	HMC5883L
3.3 V	VCC
Analog 4	SDA
Analog 5	SCL
GND	GND

3.4.7 RFID Module RC-522

ในการเคลื่อนที่ของตัวรถจากจุดเริ่มต้นไปยังจุดหมายปลายทางนั้น จะมีการติดตั้งจุดตรวจสอบในระหว่างเส้นทางการเดินทางเพื่อคอยตรวจสอบว่า ตัวรถเดินทางมายังเส้นทางที่ถูกต้อง โดยใช้ RFID Module RC-522 ในการเป็นจุดตรวจสอบ โดยใช้ไฟเลี้ยงขนาด 3.3 V เพื่อใช้ในการทำงาน โดยที่ตัว RFID Module RC-522 จะติดตั้งอยู่ที่ตัวรถ เมื่อตัวรถวิ่งผ่าน Tag จะสั่งการให้หลอดไฟที่แสดงสถานะเปิดขึ้น



รูปที่ 3.19 RFID RC522,S50 IC Card, S50 IC Key

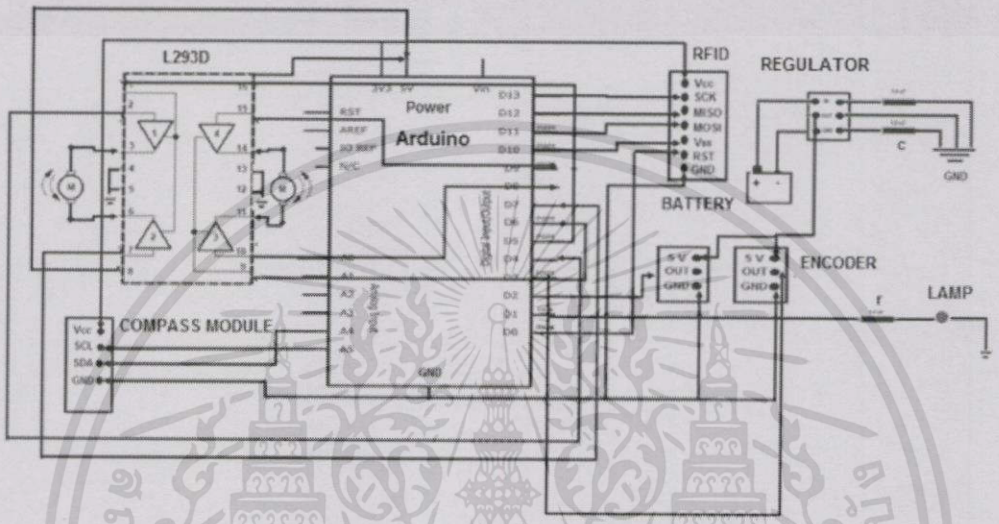


รูปที่ 3.20 MFRC522

ระยะของการอ่านค่าจะอยู่ที่ 0 - 60 mm ซึ่งจะขึ้นอยู่กับแรงดัน ถ้ามีค่าต่ำจะมีผลต่อระยะของการอ่านค่า แต่ตัวเซนเซอร์กับตัว S50 IC Card มีระยะความสูงห่างได้สูงสุดได้ 25 mm รัศมีวงกลมที่อ่านได้ 10 mm

3.5 การต่อวงจรรวม

ในการต่อวงจรอุปกรณ์ต่างๆ เข้ากับตัว Arduino เราจำเป็นที่ต้องมีความรู้ ความเข้าใจกับคุณสมบัติของอุปกรณ์นั้นๆ ในวงจรรวม จะใช้ Arduino ขา Digital ทั้งหมด 13 ขา สำหรับเชื่อมต่อเข้ากับ Encoder, RFID และ IC L293D เป็นอุปกรณ์ Drive Motor และใช้ขา Analog 2 ขา สำหรับเชื่อมต่อเข้ากับ HMC5883L Compass Module เพื่อนำไปใช้ในการควบคุมทิศทาง ดังรูปที่ 3.21



รูปที่ 3.21 การต่อวงจรรวมเข้ากับอุปกรณ์ต่างๆ

3.6 การเขียนโปรแกรม

3.6.1 การเขียนโปรแกรมแบบจำลองการเคลื่อนที่ของรถหุ่นยนต์นำทาง

เป็นการเขียนโปรแกรมในส่วนของแบบจำลองการเคลื่อนที่จากจุดเริ่มต้นไปยังจุดเป้าหมายของตัวรถหุ่นยนต์นำทาง โดยที่สามารถเลือกเส้นทางที่สั้นที่สุดในการเดินทางจากจุดเริ่มต้นยังจุดเป้าหมายปลายทางได้ ตัวรถหุ่นยนต์นำทางสามารถหลบหลีกสิ่งกีดขวางหรืออุปสรรคที่ปรากฏอยู่บนแผนที่ได้ โดยใช้โปรแกรมในการหาเส้นทางดังแสดงในภาคผนวก ก [4] และยังสามารถหารัศมีวงเลี้ยวที่เหมาะสมเมื่อมีสิ่งกีดขวางบริเวณนั้นได้ ทำให้ไม่มีความเสี่ยงของตัวรถเมื่อเวลาเลี้ยวแล้วจะไปชนสิ่งกีดขวางหรืออุปสรรค

3.6.2 การเขียนโปรแกรมใช้ในการควบคุมตัวรถหุ่นยนต์นำทาง

เป็นการเขียนโปรแกรมเพื่อใช้ในการควบคุมความเร็วและระยะทางของตัวรถหุ่นยนต์นำทางตามที่ผู้ใช้ต้องการ โดยความเร็วและระยะทางของตัวรถที่ได้ มาจากการอ่านค่าที่ได้จาก Encoder ซึ่งติดอยู่กับเพลาของมอเตอร์ทั้งสองล้อซึ่งอยู่ด้านหน้าของตัวรถ ซึ่งพัลส์ที่อ่านได้ของ Encoder แปรผันโดยตรงกับความเร็วและระยะทางเมื่อตัวรถหุ่นยนต์นำทางเคลื่อนที่ ดังแสดงในภาคผนวก ข

3.6.3 การเขียนโปรแกรมใช้สำหรับ Compass Module

ในการเคลื่อนที่ของตัวรถหุ่นยนต์นำทาง นอกจากการควบคุมความเร็วและระยะทางของตัวรถแล้ว เราจำเป็นที่จะต้องมีการควบคุมทิศทางของตัวรถหุ่นยนต์นำทางให้ได้ตามที่ต้องการอีกด้วย เพื่อที่จะให้ตัวรถหุ่นยนต์นำทางสามารถเคลื่อนที่ไปได้ยังทิศทางที่ต้องการได้อย่างถูกต้อง โดยเราจะใช้ตัว HMC5883L Compass Module ในการวัดค่ามุมของตัวรถเพื่อที่จะนำไปปรับให้ได้ค่ามุมของรถตามที่ต้องการ ดังแสดงในภาคผนวก ข [2]

3.6.4 การเขียนโปรแกรมใช้สำหรับ RFID

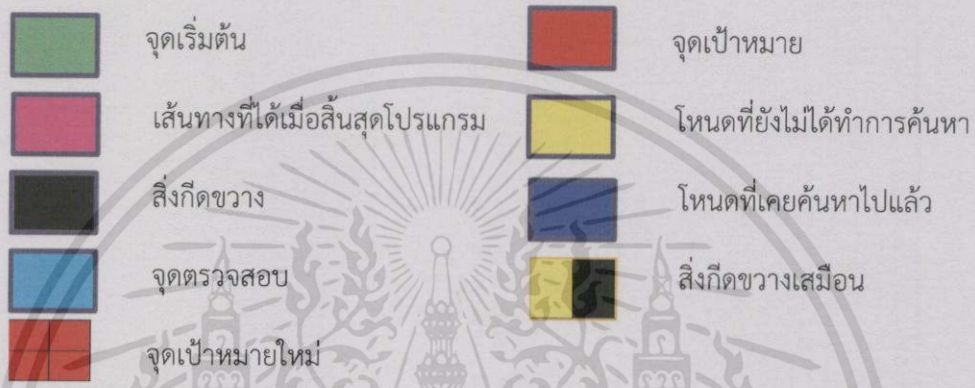
ในขณะที่รถหุ่นยนต์นำทางเคลื่อนที่จากกจุดเริ่มต้นไปยังจุดเป้าหมายปลายทาง ระหว่างทางการเคลื่อนที่ของตัวรถหุ่นยนต์นำทางจะมีการติดตั้ง RFID Key ไว้ระหว่างทางการเคลื่อนที่ของตัวรถ เพื่อคอยตรวจสอบว่าตัวรถหุ่นยนต์นำทางสามารถเดินทางมายังเส้นทางที่ต้องการได้หรือไม่ โดยจะมีการติดตั้งตัว RFID ไว้ที่ด้านใต้ของตัวรถหุ่นยนต์นำทาง เมื่อตัวรถหุ่นยนต์นำทางเคลื่อนที่ผ่าน RFID Key ซึ่งติดอยู่ตามพื้นเส้นทางในการเคลื่อนที่ ตัว RFID ก็จะสามารถอ่านค่า RFID Key ได้ หลอดไฟแสดงสถานะก็จะติดขึ้นเป็นการแสดงว่าตัวรถหุ่นยนต์นำทางสามารถเคลื่อนที่มายังเส้นทางตามที่ต้องการได้ [5]

บทที่ 4

การทดลองและผลการทดลอง

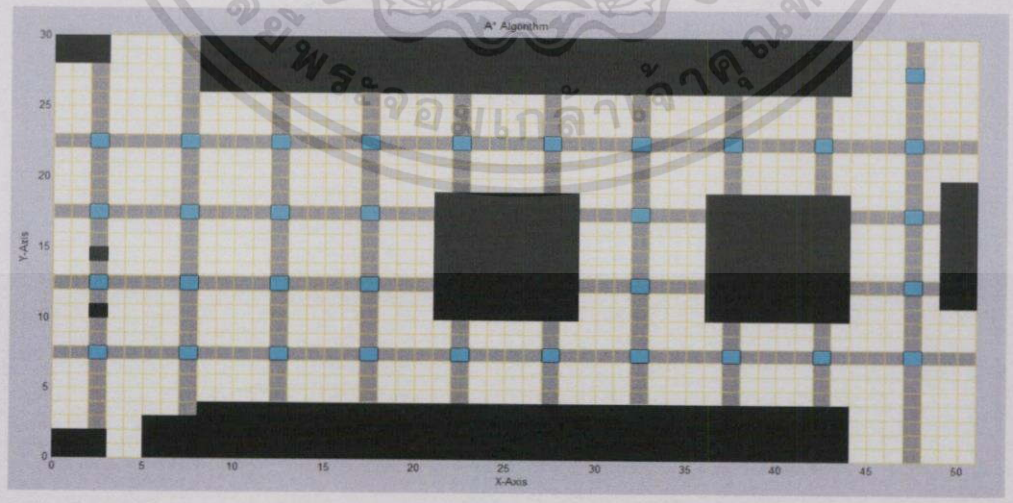
4.1 ผลการทดลองในโปรแกรม Matlab

สัญลักษณ์ในการแสดงผลการจำลองการเคลื่อนที่ของหุ่นยนต์



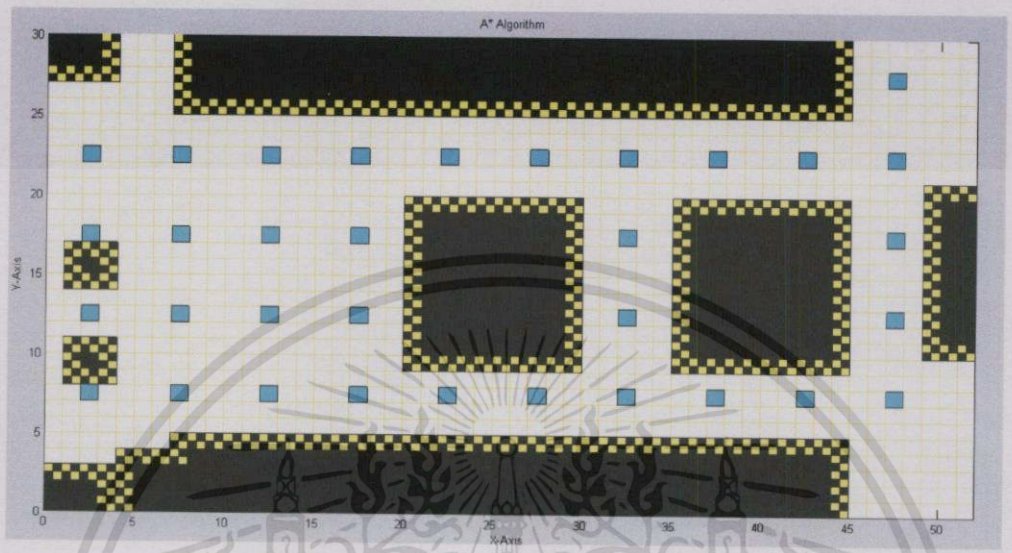
4.1.1 การสร้างเส้นทางการเดิน

ทำการจำลองสิ่งกีดขวางเสมือนลงในแผนที่เพื่อควบคุมเส้นทางการเดินของหุ่นยนต์ให้ผ่านจุดตรวจสอบเสมอ โดยการเปลี่ยนค่าพื้นที่สีขาวที่ไม่ได้อยู่ในแนวเดียวกับจุดตรวจสอบ (ทั้งแนวตั้งและแนวนอน) ให้เป็นพื้นที่ที่หุ่นยนต์ไม่สามารถเดินได้ ซึ่งจะได้เส้นทางที่หุ่นยนต์สามารถเดินได้โดยจำลองเป็นพื้นที่สีเทาดังรูปที่ 4.1 แล้วนำไปใช้สำหรับการหาเส้นทางการเดินจาก A* Algorithm



รูปที่ 4.1 แสดงผลการควบคุมเส้นทางการเดินด้วยพื้นที่สีเทา

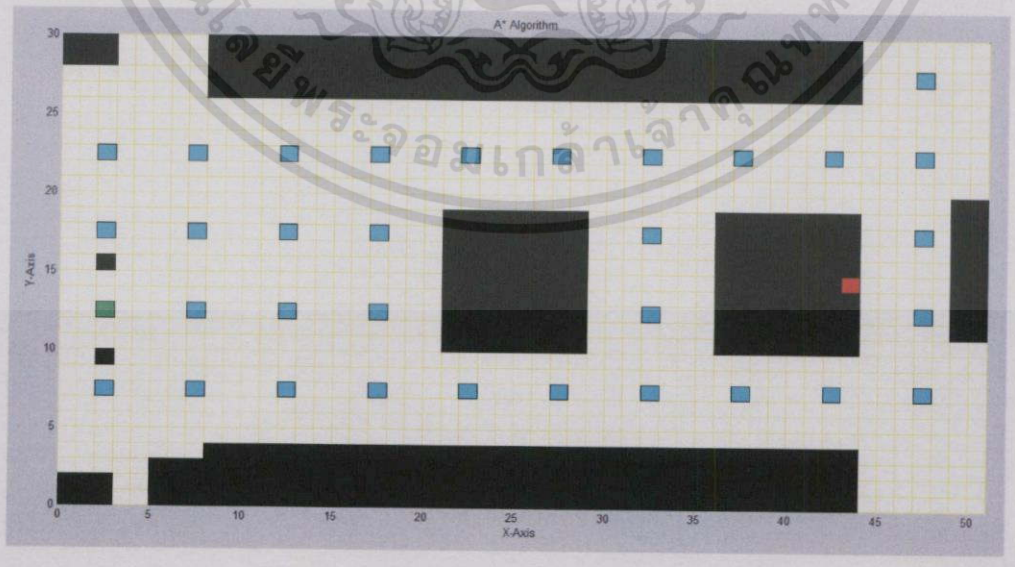
การจำลองสิ่งกีดขวางเสมือนรอบสิ่งกีดขวางที่มีอยู่จริงในแผนที่ เพื่อป้องกันโอกาสที่หุ่นยนต์จะเดินเข้าใกล้สิ่งกีดขวางมากเกินไป ในกรณีที่หุ่นยนต์มีขนาดใหญ่กว่าช่องกริดนั้นอาจทำให้เกิดความเสียหายกับหุ่นได้ ซึ่งแสดงได้ดังรูป 4.2



รูปที่ 4.2 แสดงพื้นที่ที่ทำการสร้างสิ่งกีดขวางเสมือนล้อมรอบสิ่งกีดขวางจริง

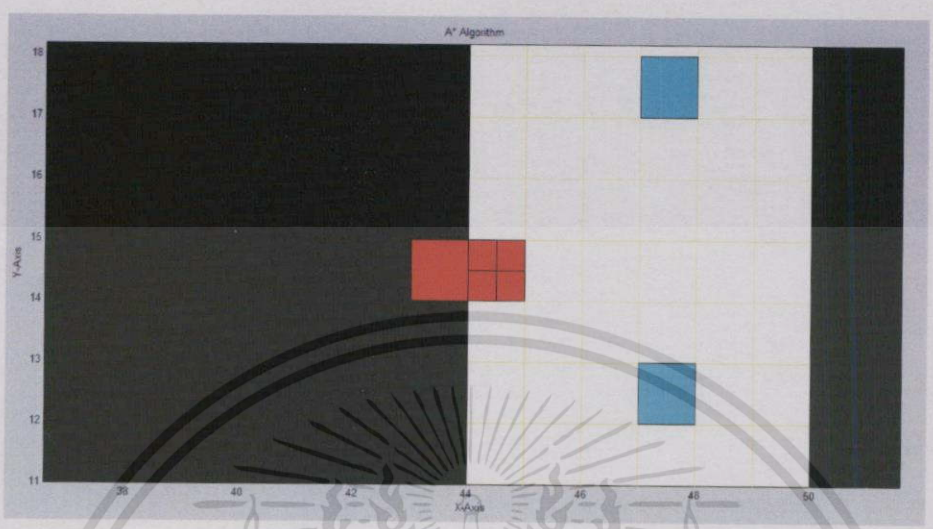
4.1.2 กำหนดจุดเริ่มต้น และจุดเป้าหมาย

จุดเริ่มต้นมีการจำกัดทิศทางการเดินได้เฉพาะทิศทางด้านหน้าของหุ่นยนต์ และจุดเป้าหมายจะต้องอยู่บนแนวที่ควบคุมเส้นทางการเดินไว้ หรือกำหนดให้อยู่ในตำแหน่งเดียวกับสิ่งกีดขวางที่มีพื้นที่ติดกับแนวเส้นทางการเดิน ดังรูปที่ 4.3



รูปที่ 4.3 แสดงตำแหน่งเริ่มต้น และเป้าหมาย

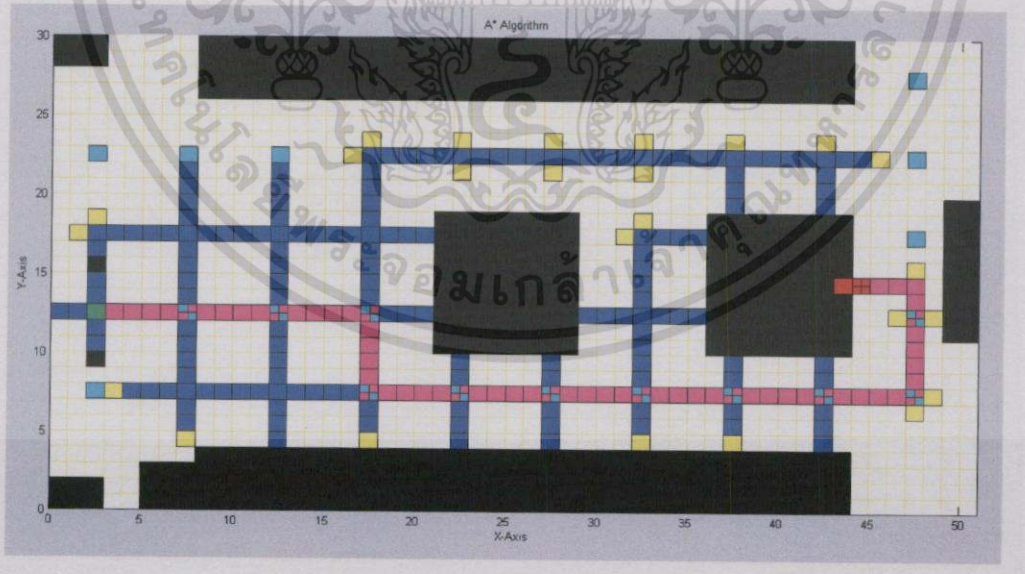
ในกรณีที่เป้าหมายอยู่ในตำแหน่งเดียวกับสิ่งกีดขวาง ทำการสร้างจุดเป้าหมายขึ้นมาใหม่บนพื้นที่ที่สามารถเดินได้ แล้วให้เป็นจุดเป้าหมายใหม่ก่อนทำการหาเส้นทางเดิน ดังแสดงในรูปที่ 4.4



รูปที่ 4.4 แสดงการจำลองจุดเป้าหมายใหม่

4.1.3 รันโปรแกรมในส่วนของ A* Algorithm

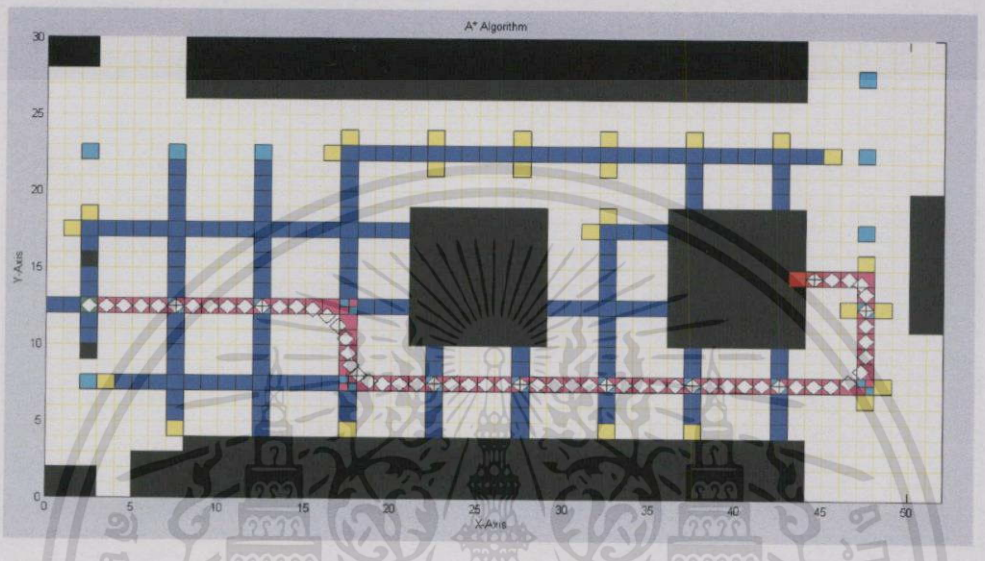
ทำการหาเส้นทางการเดินจากจุดเริ่มต้นไปยังจุดเป้าหมายด้วย A* Algorithm เก็บข้อมูลตำแหน่งเส้นทางการเดิน และจุดตรวจสอบที่เป็นเส้นทางในการเดิน แล้วแสดงผลดังรูปที่ 4.5



รูปที่ 4.5 แสดงเส้นทางจากจุดเริ่มต้นไปยังจุดเป้าหมาย

4.1.4 รันโปรแกรมในส่วนที่นำข้อมูลจาก A* Algorithm มาพัฒนาต่อ

จากการนำ A* Algorithm มาพัฒนาในส่วนของมุมในการเลี้ยว ให้มีความโค้งมากขึ้น ทำการรวมเส้นทางในแนวตรง และมุมเลี้ยวที่ได้ ให้เป็นเส้นทางการเดินใหม่ที่ต่างไปจากข้อมูลที่ได้จาก A* Algorithm แล้วแสดงผลดังรูปที่ 4.6



รูปที่ 4.6 แสดงผลจากแนวคิดการพัฒนาเส้นทางเดิน

ซึ่งในส่วนนี้จะมีการคำนวณข้อมูลต่างๆ เก็บเป็นชุดข้อมูลในรูปแบบตัวแปรในโปรแกรม Matlab คือ ค่าความเร็ว เวลา ความเร่ง รัศมีในการเลี้ยวในแต่ละส่วนโค้ง และเก็บค่าตำแหน่งของกริดต่างๆ ที่เป็นเส้นทางเดินเป็นชุดข้อมูลในแนวแกนนอน และแนวแกนตั้ง รวมถึงมีการแสดงผลเป็นภาพเคลื่อนไหวจำลองตำแหน่ง และเวลาที่ใช้ในการเดินจริงในแต่ละกริด

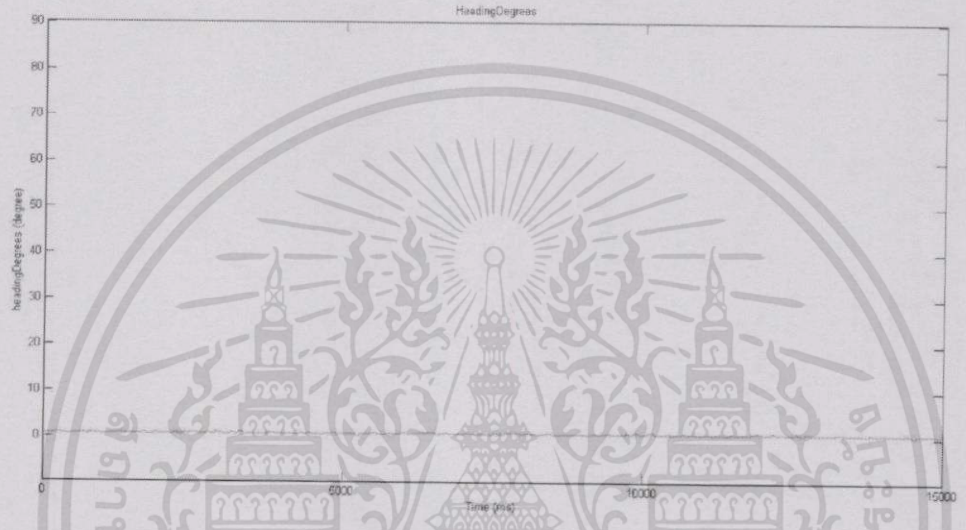
4.1.5 วิเคราะห์และสรุปผล

การจำลองการเดินทางจากจุดเริ่มต้นไปยังจุดเป้าหมาย ด้วยการเขียนโปรแกรม Matlab สามารถจำลองการเคลื่อนที่ผ่านสิ่งกีดขวาง ได้แม่นยำยิ่งขึ้น โดยมีการสร้างสิ่งกีดขวางเสมือนขึ้น และการเลี้ยวที่มีความโค้งมากขึ้น ที่รวมเส้นทางในแนวตรง และมุมเลี้ยวที่ได้ รวมทั้งการเก็บข้อมูลต่างๆ เป็นไปได้อย่างถูกต้อง

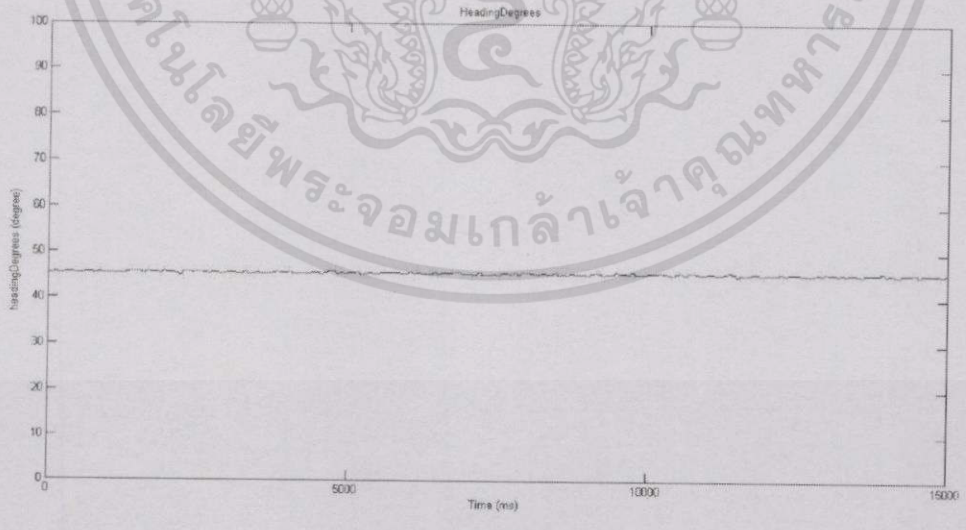
4.2 การทดลองความแม่นยำ HMC5883L 3-Axis Digital Compass Module

ทำการต่อ HMC5883L เข้ากับ Arduino UNO R3 เพื่อทำการอ่านค่ามุมที่แสดงในหน้าต่างโปรแกรม Arduino แล้วนำมาสร้างกราฟแสดงผลด้วยโปรแกรม Matlab

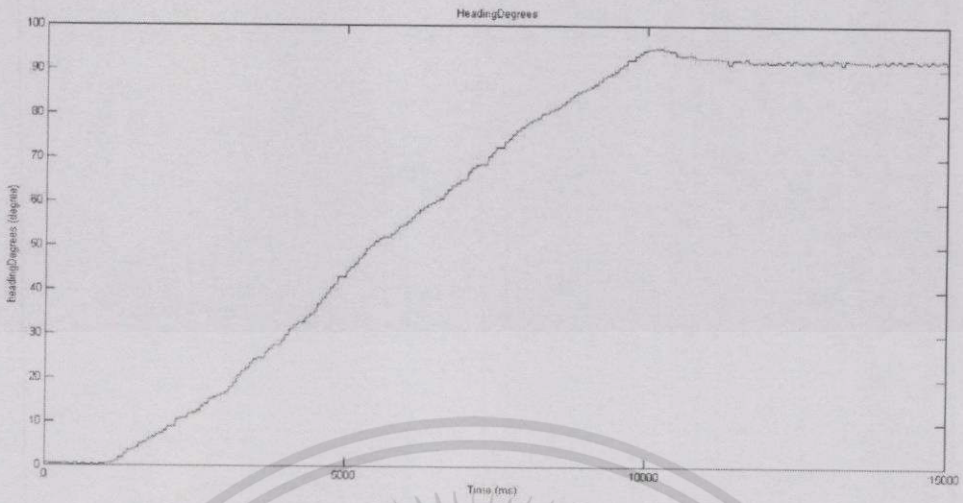
โดยทำการวางหยุดนิ่งที่มุม 0 องศา 45 องศา และทำการหมุน 0 ถึง 90 องศา แล้วบันทึกค่านำมาสร้างกราฟแสดงผล ดังรูปที่ 4.7 รูปที่ 4.8 และรูปที่ 4.9



รูปที่ 4.7 แสดงผลมุมจากการวาง HMC5883L หยุดนิ่งที่มุม 0 องศา



รูปที่ 4.8 แสดงผลมุมจากการวาง HMC5883L หยุดนิ่งที่มุม 45 องศา



รูปที่ 4.9 แสดงผลมุมที่ทำการหมุน HMC5883L ที่ 0 ถึง 90 องศา

วิเคราะห์ผลการทดลอง

จากการทดลองตัวเซนเซอร์ HMC5883L 3-Axis Digital Compass Module ทำการเชื่อมต่อกับ Arduino UNO R3 เพื่อหาความแม่นยำของมุมที่วัดได้จากองศาต่างๆ พบว่ามีค่าความคลาดเคลื่อนจากตัวมุมองศาที่ได้ตั้งค่าไว้ในแต่ละองศาไม่เกิน 0.5 องศา จึงพบว่าตัว HMC5883L 3-Axis Digital Compass Module มีความแม่นยำเพียงพอที่นำมาใช้ควบคุมทิศทางในการเคลื่อนที่ของตัวรถหุ่นยนต์นำทาง

4.3 การทดลองการเคลื่อนที่ของหุ่นยนต์นำทาง

ในการเคลื่อนที่จากจุดเริ่มต้นไปยังจุดเป้าหมายซึ่งมี S50 IC Card ติดอยู่กับพื้นทุกๆ 50 cm โดยวัดค่าความผิดพลาดได้ ทางตรงดังตารางที่ 4.1 ทางเลี้ยวซ้ายดังตารางที่ 4.2 ทางเลี้ยวขวาดังตารางที่ 4.3 ทางตรงแล้วเลี้ยวซ้ายดังตารางที่ 4.4 และทางตรงแล้วเลี้ยวขวาดังตารางที่ 4.5 โดยที่ตัวเซนเซอร์กับตัว S50 IC Card ผิดพลาดได้ ±1 cm ในแนวระนาบ แล้วทำการวัดค่าความผิดพลาดระหว่างจุดกึ่งกลาง S50 IC Card กับจุดกึ่งกลางรถว่าห่างกันเท่าไหร่ในแนวราบ โดยที่ตัวหุ่นยนต์นำทางมีขนาดความกว้าง 13 cm บันทึกค่า

การทดลองเมื่อปล่อยรถเป็นจำนวน 10 ครั้งตามเส้นทางการเดินต่างๆ

✓ เมื่อ RFID อ่านค่าได้

- เมื่อมีค่าความผิดพลาดเมื่อวัดจากตัวอ่าน RFID ไปในทางซ้าย

+ เมื่อมีค่าความผิดพลาดเมื่อวัดจากตัวอ่าน RFID ไปในทางขวา

ตารางที่ 4.1 ความผิดพลาดการเดินตรง

จำนวนครั้งที่	ค่าความผิดพลาด (cm)
1	✓
2	✓
3	✓
4	-1
5	✓
6	✓
7	✓
8	✓
9	-1
10	✓

ตารางที่ 4.2 ความผิดพลาดการเดินเลี้ยวซ้าย

จำนวนครั้งที่	ค่าความผิดพลาด (cm)
1	+1
2	✓
3	✓
4	✓
5	✓
6	-1
7	✓
8	✓
9	+2
10	✓

ตารางที่ 4.3 ความผิดพลาดการเดินเลี้ยวขวา

จำนวนครั้งที่	ค่าความผิดพลาด (cm)
1	✓
2	+1
3	✓
4	✓
5	✓
6	✓
7	+2
8	✓
9	✓
10	✓

ตารางที่ 4.4 ความผิดพลาดการเดินตรงแล้วเลี้ยวซ้าย

จำนวนครั้งที่	ค่าความผิดพลาด (cm)
1	✓
2	+1
3	✓
4	-2
5	-1
6	✓
7	✓
8	✓
9	✓
10	✓

ตารางที่ 4.5 ความผิดพลาดการเดินตรงแล้วเลี้ยวขวา

จำนวนครั้งที่	ค่าความผิดพลาด (cm)
1	✓
2	✓
3	✓
4	+1
5	✓
6	-1
7	✓
8	✓
9	✓
10	-2

สรุปผลการทดลอง

จากการทดลองการเคลื่อนที่ของหุ่นยนต์นำทางในข้างต้น โดยหุ่นยนต์นำทางมีความกว้างของตัวรถ 13 cm และทำการวัดค่าความผิดพลาด (e) ในแนวระนาบระหว่าง RFID ที่ติดอยู่ด้านใต้ตัวรถกับตัวเซนเซอร์ S50 IC Card ซึ่งติดอยู่ในแนวเส้นทางการเดิน โดยที่ตัว RFID จะทำการตรวจจับเซนเซอร์ S50 IC Card ได้ก็ต่อเมื่อ $|e| \leq 1$ cm และในกรณีที่ RFID ไม่สามารถตรวจจับ S50 IC Card ได้จะพิจารณาออกเป็น 2 กรณี กรณีแรกเมื่อ $1 < |e| \leq 6.5$ cm เซนเซอร์ S50 IC Card จะอยู่ภายใต้ตัวรถ และกรณีที่สองเมื่อ $|e| > 6.5$ cm เซนเซอร์ S50 IC Card จะไม่อยู่ภายใต้ตัวรถ โดยสามารถหาความผิดพลาดรวมของแต่ละเส้นทางเดินได้ ดังตารางที่ 4.6

ตารางที่ 4.6 ความผิดพลาดรวมในการเคลื่อนที่ของหุ่นยนต์นำทาง

เส้นทางการเดิน	$\|e\| \leq 1$ cm	$1 < \|e\| \leq 6.5$ cm	$\|e\| > 6.5$ cm
เดินตรง	8	2	0
เลี้ยวซ้าย	7	3	0
เลี้ยวขวา	8	2	0
เดินตรงแล้วเลี้ยวซ้าย	7	3	0
เดินตรงแล้วเลี้ยวขวา	7	3	0

การจำลองการเดินทางจากจุดเริ่มต้นไปยังจุดเป้าหมาย ด้วยการเขียนโปรแกรม Matlab สามารถจำลองการเคลื่อนที่ไปได้อย่างถูกต้อง ในส่วนของตัวรถนำทางสามารถจำลองการเดินทางจากจุดเริ่มต้นไปยังจุดเป้าหมายปลายทางได้ จากการทดลองพบว่ายังคงมีค่าความผิดพลาดเกิดขึ้นในบางครั้ง

ในส่วนของตัวรถนำทางสามารถจำลองการเดินทางจากจุดเริ่มต้นไปยังจุดเป้าหมายได้ โดยค่าความผิดพลาด $1 < |e| \leq 6.5$ cm ในทางตรงมีความผิดพลาด 20% เลี้ยวซ้ายมีความผิดพลาด 30% เลี้ยวขวามีความผิดพลาด 20% เดินตรงแล้วเลี้ยวซ้ายมีความผิดพลาด 30% เดินตรงแล้วเลี้ยวซ้ายมีความผิดพลาด 30% คิดเป็นความผิดพลาดรวม 26%

บทที่ 5

สรุปและวิจารณ์ปัญหา

5.1 สรุปวิธีการดำเนินงาน

ในระยะแรกของการทำโครงงานจะเป็นการศึกษาเกี่ยวกับทฤษฎี A* Algorithm เพื่อที่จะนำมาใช้ในการหาเส้นทางการเดินจากจุดเริ่มต้นไปยังเป้าหมายได้อย่างถูกต้อง ขั้นตอนถัดมาในส่วนของตัวรถหุ่นยนต์นำทางนำมาใช้ประกอบร่วมกับเซนเซอร์ Encoder, Compass Module และ RFID โดยเซนเซอร์ทั้งหมดจะทำงานและส่งค่ามาประมวลผลที่ตัวไมโครคอนโทรลเลอร์ Arduino UNO R3 เพื่อให้ตัวรถหุ่นยนต์นำทางสามารถเคลื่อนที่จากจุดเริ่มต้นไปยังจุดเป้าหมายปลายทางได้อย่างถูกต้อง

5.2 ปัญหาและอุปสรรค

1. ค่าที่อ่านได้จาก Encoder มีค่าความผิดพลาดเกิดขึ้น ทำให้ค่าที่จะนำมาใช้เพื่อควบคุมมอเตอร์เกิดความผิดพลาด

2. ค่าที่อ่านได้จากตัว Compass Module ยังมีค่าความผิดพลาดเกิดขึ้น ทำให้เกิดความผิดพลาดขึ้นบางขณะในการควบคุมทิศทางของรถหุ่นยนต์นำทาง

3. DC Motors ยังไม่สามารถทำความเร็วได้ละเอียดตามที่ต้องการ

5.3 แนวทางในการแก้ไขปัญหา

1. เลือกใช้ Encoder รวมถึงจานหมุนที่ติดกับเพลาให้มีความละเอียดมากยิ่งขึ้น เพื่อที่จะได้นำไปควบคุม DC Motors ได้อย่างแม่นยำมากยิ่งขึ้น

2. เลือกใช้ Compass Module ที่มีค่าความละเอียดมากยิ่งขึ้น จะช่วยเพิ่มความลื่นไหลในการเคลื่อนที่ของตัวรถหุ่นยนต์นำทางได้ดียิ่งขึ้น

3. เลือกใช้ DC Motors ที่มีประสิทธิภาพดียิ่งขึ้น เพื่อให้สามารถควบคุมความเร็วของรถได้อย่างละเอียดมากยิ่งขึ้น

5.4 แนวทางการพัฒนาต่อ

เพิ่มในส่วนการเชื่อมต่อ RFID เข้ากับ Matlab เพื่อทำการยืนยันสถานที่ ณ ปัจจุบันของตัวรถนำทางได้

เอกสารอ้างอิง

[1] วงจรลดแรงดัน

http://learn.adafruit.com/category/learn-arduino [11 มีนาคม 2557]

[2] HMC5883L 3-Axis Digital Compass Module

http://bildr.org/2012/02/hmc5883l_arduino/ [12 มีนาคม 2557]

[3] A* Pathfinding for Beginners

http://www.policyalmanac.org/games/aStarTutorial.htm [12 มีนาคม 2557]

[4] Robot Navigation Algorithm

https://github.com/jmiseikis/RobotNavigation [13 มีนาคม 2557]

[5] RFID Module RC-522

http://www.grantgibson.co.uk/2012/04/how-to-get-started-with-the-mifare-mf522-an-and-arduino/ [13 มีนาคม 2557]

[6] Smart Car Chassis Robot V7

http://www.ebay.com/itm/4WD-V7-Smart-Car-Chassis-Robot-V7%20/1211515713-15?pt=LH_DefaultDomain_0&hash=item1c35324573 [13 มีนาคม 2557]

[7] L293D Drive Motor

http://users.ece.utexas.edu/~valvano/Datasheets/L293d.pdf [13 มีนาคม 2557]

ภาคผนวก

ภาคผนวก ก

Code A* Algorithm

```
% clear variables
clear
clc
% Input Start & Stop
figure
startX=3; startY=13;
targetX=44; targetY=15;
% Create variables
newOpenListItemID = 0; parentXval = 0; parentYval = 0;
m = 0; u = 0; v = 0; temp = 0; corner = 0;
numberOfOpenListItems = 0; addedGCost = 0; tempGCost = 0; path = 0;
% Variables used as constants
walkable = 1;
onOpenList = 7;
curve = 2;
onClosedList = 8; unwalkable = 4; checkPoint = 5;
checkPointSpace = 5; checkPointcornerII = 2; checkPointcornerIII = 3;
sameUnwalkable = 22; neighborUnwalk = sameUnwalkable;
found = 1; nonexistent = 7;
% Import map form Excel
whichList2 = xlsread ('E:\2556 4_2\project\291013\Ecc.xlsx','ROOM');
% Rotate data of map
whichList = rot90(whichList2,-1);
whichListOrigi = whichList;
% Find mapsize
mapWidth = size(whichList, 1);
mapHeight = size(whichList, 2);
% Display map
createMap_2(mapWidth,mapHeight)
% Show map obstacle
[ whichList ] = showObstacle(
mapWidth,mapHeight,unwalkable,whichList,D,neighborUnwalk,checkPoint,checkPointcornerII,checkPointcornerIII );
% Show Start & Stop
test_str( startX,startY )
test_stp( targetX,targetY )
% Record checkpoint
[ k,checkpointX,checkpointY ] = recordCheckpoint(
mapWidth,mapHeight,checkPoint,checkPointcornerII,checkPointcornerIII,whichList );
% Find same unwaikable
[ whichList ] = findSameUnwalkable(
mapWidth,mapHeight,k,whichList,neighborUnwalk,unwalkable,checkPoint,checkPointcornerII,checkPointcornerIII,sa
meUnwalkable,checkpointX,checkpointY );
% Find new targe out of obstacle
 [ targetX_1,targetY_1,whichList,targetX,targetY ] = findTargetOutOfObstacle(
whichList,targetX,targetY,unwalkable,D,checkPoint,checkPointcornerII,checkPointcornerIII,sameUnwalkable,neighbor
Unwalk );
```

```
 parentX = zeros(mapWidth,mapHeight);        % 2D array stores x coord of parent for every cell
parentY = zeros(mapWidth,mapHeight);        % 2D array stores y coord of parent for every cell
Gcost = zeros(mapWidth,mapHeight);          % 2D array stores G cost of each cell
Hcost = zeros(1,mapWidth*mapHeight);        % 1D array stores H cost of open cell list
pathlength = 0;                             % Will store pathlength
% Check if start and finish coordinates are the same
if ( (startX == targetX) && (startY == targetY) )
    disp('Start and finish positions are the same');
    return;
end
% Add starting position to the open list
numberOfOpenListItems = 1;
openList(1) = 1;
openX(1) = startX;
openY(1) = startY;
% If checkpoint same start
if ( (whichList(startX,startY) ==  checkPoint) || (whichList(startX,startY) ==  checkPointcornerII)||
(whichList(startX,startY) ==  checkPointcornerIII) )
    checkPointStrX = startX;
    checkPointStrY = startY;
else
    checkPointStrX = 0;
    checkPointStrY = 0;
end
% If checkpoint same stop
if ( (whichList(targetX,targetY) ==  checkPoint) || (whichList(targetX,targetY) ==  checkPointcornerII)||
(whichList(targetX,targetY) ==  checkPointcornerIII) )
    checkPointStpX = targetX;
    checkPointStpY = targetY;
else
    checkPointStpX = 0;
    checkPointStpY = 0;
end
% Counter used for printing
i = 1;
% Create an inifinite loop (until goal is reached or defined unreachable
while (1)

   % Check if there are any members in the open list
   if (numberOfOpenListItems ~= 0)
        % Get the values of the first item in the list into current vals
        parentXval = openX(openList(1));
        parentYval = openY(openList(1));
        % set the coordinate to closed list
        whichList(parentXval, parentYval) = onClosedList;
     % Print out current position being analysed
     i = i + 1;
     % Draw current position in blue
     if ( ~((parentXval == startX) && (parentYval == startY) || (parentXval == targetX) && (parentYval == targetY)) )
        % Show current (Blue)
        test_parent( parentXval,parentYval,targetX,targetY )
```

```
    end

    % ## List sorting ##
    % Put last element into the slot of first one
    openList(1) = openList(numberOfOpenListItems);

    % Remove the first element from the open list
    numberOfOpenListItems = numberOfOpenListItems - 1;

    % Make a copy of open list excluding first element
    tempOpenList = openList(1:numberOfOpenListItems);

    % Clear openList
    clear openList;

    % Assign tempOpenList to openList, clear temp one
    openList = tempOpenList;
    clear tempOpenList;
    v = 1;
    % Put the item into it's appropriate slot by looking at the Fcost
    while (1)
      u = v;
      % If both children exist
      if (2*u+1 <= numberOfOpenListItems)
        % Check if F cost of the parent is greater than each child
        % Then select the lowest of the two children
        if (Fcost(openList(u)) >= Fcost(openList(2*u)))
          v = 2*u;
        end
        if (Fcost(openList(v)) >= Fcost(openList(2*u+1)))
          v = 2*u+1;
        end
        % If only 1 child exists
      elseif (2*u <= numberOfOpenListItems)
        % Check if F cost of the parent is greater than child
        if (Fcost(openList(u)) >= Fcost(openList(2*u)))
          v = 2*u;
        end
      end

      % If parent's F is larger than one of its children, swap them
      if (u ~= v)
        temp = openList(u);
        openList(u) = openList(v);
        openList(v) = temp;
        % Otherwise - break out of the loop
      else
        break;
      end
    end
```

```
% Check the neighbouring squares
for a=(parentXval-1):(parentXval+1)

    for b=(parentYval-1):(parentYval+1)

        % Check if it is within bounds of a map
        if ( (a > 0) && (b > 0) && (a <= mapWidth) && (b <= mapHeight) && ( (a==(parentXval)) ||
(b==(parentYval))  ) )

            if (whichList(a,b) ~= onClosedList)

             % Check if not on closed list
             if (whichList(a,b) ~= onClosedList)

                % Check if not an obstacle
              if ( (whichList(a,b) ~= unwalkable) && (whichList(a,b) ~= neighborUnwalk) && (whichList(a,b) ~=
sameUnwalkable) )
                corner = walkable;

                % Corner detection to prevent cutting
                if (a == parentXval-1)
                  if (b == parentYval-1)
                    if ( (whichList(parentXval-1,parentYval) == unwalkable ) || ...
                         (whichList(parentXval,parentYval-1) == unwalkable) || ...
                         (whichList(parentXval-1,parentYval) == sameUnwalkable ) || ...
                         (whichList(parentXval,parentYval-1) == sameUnwalkable) )
                      corner = unwalkable;
                    end
                  elseif (b == parentYval+1)
                    if ( (whichList(parentXval,parentYval+1) == unwalkable) || ...
                         (whichList(parentXval-1,parentYval) == unwalkable) || ...
                         (whichList(parentXval,parentYval+1) == sameUnwalkable) || ...
                         (whichList(parentXval-1,parentYval) == sameUnwalkable) )
                      corner = unwalkable;
                    end
                  end
                elseif (a == parentXval+1)
                  if (b == parentYval-1)
                    if ( (whichList(parentXval,parentYval-1) == unwalkable) || ...
                         (whichList(parentXval+1,parentYval) == unwalkable) || ...
                         (whichList(parentXval,parentYval-1) == sameUnwalkable) || ...
                         (whichList(parentXval+1,parentYval) == sameUnwalkable) )
                      corner = unwalkable;
                    end
                  elseif (b == parentYval+1)
                    if ( (whichList(parentXval+1,parentYval) == unwalkable) || ...
                         (whichList(parentXval,parentYval+1) == unwalkable) || ...
                         (whichList(parentXval+1,parentYval) == sameUnwalkable) || ...
                         (whichList(parentXval,parentYval+1) == sameUnwalkable) )
                      corner = unwalkable;
                    end
```

```
    end
end

% If corner is walkable
if (corner == walkable)
    % If not on the open list, add it
    if (whichList(a,b) ~= onOpenList)
        % Create a new open list item
        newOpenListItemID = newOpenListItemID + 1;
        m = numberOfOpenListItems + 1;
        openList(m) = newOpenListItemID;
        openX(newOpenListItemID) = a;
        openY(newOpenListItemID) = b;


        if (whichList (a,b) == checkPoint ) || (whichList (a,b) == checkPointcornerII ) || (whichList
(a,b) == checkPointcornerIII )
            neighbor_checkPointX(Z) = a;
            neighbor_checkPointY(Z) = b;
            Z = Z+1;
        end

        if (whichList (a,b) ~= checkPoint ) && (whichList (a,b) ~= checkPointcornerII ) &&
(whichList (a,b) ~= checkPointcornerIII )
            % Draw current position in yellow
            if ( ~((a == startX) && (b == startY) || (a == targetX) && (b == targetY)) )

                % Show neighbor (yellow)
                test_neighbor2(a,b,startX,startY)
                pause (0.01)
            end
        end
        % Calculate its G cost
        if ( (abs(a - parentXval) == 1) && (abs(b - parentYval) == 1) )
            addedGCost = 14;
        else
            addedGCost = 10;
        end
        % Update Gcost map
        Gcost(a,b) = Gcost(parentXval,parentYval) + addedGCost;

        % Get H and F costs and parent
        Hcost(openList(m)) = 10*(abs(a - targetX) + abs(b - targetY));
        Fcost(openList(m)) = Gcost(a,b) + Hcost(openList(m));
        parentX(a,b) = parentXval;
        parentY(a,b) = parentYval;
        % Listing
        while (m ~= 1)
            % Check if child's F cost < parent's F cost. If so, swap them
            if (Fcost(openList(m)) <= Fcost(openList(round(m/2))))
                temp = openList(round(m/2));
```

```
                openList(round(m/2)) = openList(m);
                openList(m) = temp;
                m = round(m/2);
              else
                break;
              end
            end
            % Increment openlist items
            numberOfOpenListItems = numberOfOpenListItems + 1;
            % Change current node into open list
            whichList(a,b) = onOpenList;
          end
        else
            fprintf('%d. Corner found, walking around it \n', i);
          i = i + 1;
        end

      end
    else % if whichList(a,b) = onOpenList
      % Calculate its G cost
      if ( (abs(a - parentXval) == 1) && (abs(b - parentYval) == 1) )
        addedGCost = 14;
      else
        addedGCost = 10;
      end
      tempGcost = Gcost(parentXval,parentYval) + addedGCost;

      % If this path is shorter, change Gcost, Fcost and the parent cell
      if (tempGcost < Gcost(a,b)),
        parentX(a,b) = parentXval;
        parentY(a,b) = parentYval;
        Gcost(a,b) = tempGcost;

        % Changing G cost also changes F cost, sothe open list has to be updated and reordered
        % Look for item
        for x=1:numberOfOpenListItems,
          % Identify the item
          if ( (openX(openList(x)) == a) && (openY(openList(x)) == b) )
            % Change F cost
            Fcost(openList(x)) = Gcost(a,b) + Hcost(openList(x));
            % Reorder the list if needed
            m = x;
            while (m ~= 1)
              % If child < parent, swap them
              if (Fcost(openList(m)) < Fcost(openList(round(m/2))))
                temp = openList(round(m/2));
                openList(round(m/2)) = openList(m);
                openList(m) = temp;
                m = round(m/2);
              else
                break;
```

```
                    end
                end
                % Exit the loop when found
                break;
            end
        end
    end
end
end
end
end
% If open list is empty
else
    path = nonexistent;
    break;
end
% If target is added to open list, path has been found
if (whichList(targetX,targetY) == onOpenList)
    path = found;
    break;
end
end
% If path was found
if (path == found)
    pathLength = 0;
    % Backtrack the path using parents
    pathX = targetX;
    pathY = targetY;
    % Print out backtracking
    fprintf('%d. Backtracing to find the shortest route \n', i);
    i = i + 1;
    % Loop until starting position is reached
    while(1)
        % Lookup parent of current cell
        tempx = parentX(pathX,pathY);
        pathY = parentY(pathX,pathY);
        pathX = tempx;
        % Draw return path in magenta
        if ( ~((pathX == startX) && (pathY == startY) || (pathX == targetX) && (pathY == targetY)) )
            test_path( pathX,pathY)
            pause(0.05);
            %nav_drawCurrPos(figHandle, pathX, pathY, 'm', objMap);
        end
        % Get the length
        pathLength = pathLength + 1;

        % If starting position reached, break out of the loop
        if ( (pathX == startX) && (pathY == startY) )
            break;
        end
```

```
    end

    % Print out failure
    fprintf('%d. Shortest route is shown in magenta. Total length: %d steps \n', i, pathLength);

end
```

ภาคผนวก ข

Code Arduino

```
// Reference the I2C Library
#include <Wire.h>
// Reference the HMC5883L Compass Library
#include <HMC5883L.h>
#define encoder0PinA  2
#define encoder1PinA  3
#include <SPI.h>
#define   uchar     unsigned char
#define   uint      unsigned int
//Maximum length of the array
#define MAX_LEN 16
HMC5883L compass;
double disAll[] = {50,69,0}, type[] = {1,2,1};
int error = 0,disError = 0,target;
volatile long encoder0Pos = 0, long encoder1Pos = 0;
long j=0;
int IN1 = 4, EN1 = 5, IN2 = 7, IN4 = 8, EN2 = 6, IN5 = 9, degreeSetpoint, PV;
long time2 = 0;
int timeCurve = 0;
void setup()
{
 Serial.begin(9600);
 //*****moter + encoder*****//
 pinMode(IN1,OUTPUT);pinMode(IN2,OUTPUT);pinMode(IN4,OUTPUT);pinMode(IN5,OUTPUT);
 pinMode(EN1,OUTPUT);pinMode(EN2,OUTPUT);pinMode(encoder0PinA, INPUT); pinMode(encoder1PinA, INPUT);
 attachInterrupt(0, doEncoderMotor0, CHANGE);
 attachInterrupt(1, doEncoderMotor1, CHANGE);

 //*****HMC5883L compass*******//
 Serial.println("Starting the I2C interface.");
 Wire.begin(); // Start the I2C interface.
 Serial.println("Constructing new HMC5883L");
 compass = HMC5883L(); // Construct a new HMC5883 compass.
 Serial.println("Setting scale to +/- 1.3 Ga");
 error = compass.SetScale(1.3); // Set the scale of the compass.
 if(error != 0) // If there is an error, print it out.
 Serial.println(compass.GetErrorText(error));
 Serial.println("Setting measurement mode to continous.");
 error = compass.SetMeasurementMode(Measurement_Continuous); // Set the measurement mode to
Continuous
 if(error != 0) // If there is an error, print it out.
 Serial.println(compass.GetErrorText(error));
}
// Our main program loop.
void loop()
{
```

```
target = disAll[j];
disError = target-encoder0Pos;
float  disL = encoder0Pos, disR = encoder1Pos;
int casei = type[j];
// Retrive the raw values from the compass (not scaled).
MagnetometerRaw raw = compass.ReadRawAxis();
// Retrived the scaled values from the compass (scaled to the configured scale).
MagnetometerScaled scaled = compass.ReadScaledAxis();
// Values are accessed like so:
int MilliGauss_OnThe_XAxis = scaled.XAxis;// (or YAxis, or ZAxis)
// Calculate heading when the magnetometer is level, then correct for signs of axis.
float heading = atan2(scaled.YAxis, scaled.XAxis);
float declinationAngle = 0.0457;
heading += declinationAngle;
// Correct for when signs are reversed.
if(heading < 0)
  heading += 2*PI;
// Check for wrap due to addition of declination.
if(heading > 2*PI)
  heading -= 2*PI;
// Convert radians to degrees for readability.
float headingDegrees = heading * 180/M_PI;
// Output the data via the serial port.
Output(raw, scaled, heading, headingDegrees);
long time = millis();
// Record degree's robot
if (time <= 2000)
{
   degreeSetpoint = headingDegrees;
   Serial.print ( "degreeSetpoint = " );
   Serial.print ( degreeSetpoint );
}
else
{
 // Forward
 if (casei == 1 )
 {
       PV = headingDegrees;
       analogWrite(EN1,(170)); digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);
       analogWrite(EN2,(150)); digitalWrite(IN4, HIGH); digitalWrite(IN5, LOW);
       if (PV < degreeSetpoint-1)
       {
         analogWrite(EN1,(170*1.1)); digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);
         analogWrite(EN2,(150)); digitalWrite(IN4, HIGH); digitalWrite(IN5, LOW);
       }
       else if (PV > degreeSetpoint+1)
       {
         analogWrite(EN1,(170));
         digitalWrite(IN1, HIGH);
         digitalWrite(IN2, LOW);
         analogWrite(EN2,150*1.1);
```

```
        digitalWrite(IN4, HIGH);
        digitalWrite(IN5, LOW);
       }
       else
       {
        analogWrite(EN1,(170));
        digitalWrite(IN1, HIGH);
        digitalWrite(IN2, LOW);
        analogWrite(EN2,150);
        digitalWrite(IN4, HIGH);
        digitalWrite(IN5, LOW);
       }
}
// Turn Right
else if(casei == 2 )
{
 if ((millis()-time2) >= 3500/9 ) {
    time2 = millis();
    timeCurve = timeCurve + 1 ;
    Serial.print ( "  timeCurve = " );
    Serial.print ( timeCurve );
    degreeSetpoint = degreeSetpoint+10 ;
    if (degreeSetpoint > 360)
    {
     degreeSetpoint = degreeSetpoint-360;
    }
    else if (degreeSetpoint < 0)
    {
     degreeSetpoint = degreeSetpoint+ 360;
    }
}
      PV = headingDegrees;
      analogWrite(EN1,(200));
      digitalWrite(IN1, HIGH);
      digitalWrite(IN2, LOW);
      analogWrite(EN2,(100));
      digitalWrite(IN4, HIGH);
      digitalWrite(IN5, LOW);
    if (PV < degreeSetpoint-1)
    {
     analogWrite(EN1,(200*1.1));
     digitalWrite(IN1, HIGH);
     digitalWrite(IN2, LOW);
     analogWrite(EN2,(100));
     digitalWrite(IN4, HIGH);
     digitalWrite(IN5, LOW);
    }
    else if (PV > degreeSetpoint+1)
    {
     analogWrite(EN1,(200));
     digitalWrite(IN1, HIGH);
```

```
        digitalWrite(IN2, LOW);
        analogWrite(EN2,100*1.1);
        digitalWrite(IN4, HIGH);
        digitalWrite(IN5, LOW);
      }
       else
      {
        analogWrite(EN1,(20));
        digitalWrite(IN1, HIGH);
        digitalWrite(IN2, LOW);
        analogWrite(EN2,100);
        digitalWrite(IN4, HIGH);
        digitalWrite(IN5, LOW);
      }
 }
 // Turn Left
 else
{
  if ((millis()-time2) >= 3800/9 ) {
     time2 = millis();
     timeCurve = timeCurve + 1 ;
     Serial.print ( "  timeCurve = " );
     Serial.print ( timeCurve );
     degreeSetpoint = degreeSetpoint-10 ;
     if (degreeSetpoint > 360)
     {
      degreeSetpoint = degreeSetpoint-360;
     }
     else if (degreeSetpoint < 0)
     {
      degreeSetpoint = degreeSetpoint+ 360;
     }
 }
       PV = headingDegrees;
       analogWrite(EN1,(120));
       digitalWrite(IN1, HIGH);
       digitalWrite(IN2, LOW);
       analogWrite(EN2,(140));
       digitalWrite(IN4, HIGH);
       digitalWrite(IN5, LOW);
     if (PV < degreeSetpoint-1)
     {
       analogWrite(EN1,(120*1.1));
       digitalWrite(IN1, HIGH);
       digitalWrite(IN2, LOW);
       analogWrite(EN2,(140));
       digitalWrite(IN4, HIGH);
       digitalWrite(IN5, LOW);
       Serial.print ( " C1 " );
     }
     else if (PV > degreeSetpoint+1)
```

```
        {
          analogWrite (EN1,(110));
          digitalWrite (IN1, HIGH);
          digitalWrite (IN2, LOW);
          analogWrite(EN2,140*1.1);
          digitalWrite(IN4, HIGH);
          digitalWrite(IN5, LOW);
          Serial.print ( " C2 " );
        }
         else
        {
          analogWrite(EN1,(120));
          digitalWrite(IN1, HIGH);
          digitalWrite(IN2, LOW);
          analogWrite(EN2,140);
          digitalWrite(IN4, HIGH);
          digitalWrite(IN5, LOW);
          Serial.print ( " C3 " );
        }
    }
  }
  if (disError <= 0)
  {
      j++;
      encoder0Pos = 0, encoder1Pos = 0;
      target = disAll[j];
      // Stop
       if (target == 0)
      {
        digitalWrite(EN1,0);
        digitalWrite(IN1, LOW);
        digitalWrite(IN2, LOW);
        digitalWrite(EN2,0);
        digitalWrite(IN4, LOW);
        digitalWrite(IN5, LOW);
        Serial.print ( "  stop  " );
      }
      //digitalWrite(EN1,0);
      //digitalWrite(EN2,0);
  }
  if (timeCurve > 9)
  {
        timeCurve = 0;
        digitalWrite(EN1,0);
        digitalWrite(IN1, LOW);
        digitalWrite(IN2, LOW);
        Serial.print ( "Time Stop" );
        digitalWrite(EN2,0);
        digitalWrite(IN4, LOW);
        digitalWrite(IN5, LOW);
        Serial.print ( "  stop  " );
```

```
  }
}
// Output the data down the serial port.
void Output(MagnetometerRaw raw, MagnetometerScaled scaled, float heading, float headingDegrees)
{
   Serial.print ( "degreeSetpoint = " );
   Serial.print ( degreeSetpoint );Serial.print ( " , " );
   Serial.print(PV);
   Serial.println(" Degrees   \t");
}
void doEncoderMotor0(){
  if (digitalRead(encoder0PinA) == HIGH) {
     encoder0Pos = encoder0Pos + 1;
  }
}
void doEncoderMotor1(){
   if (digitalRead(encoder1PinA) == HIGH) {
     encoder1Pos = encoder1Pos + 1;
   }
}
```