การออกแบบและพัฒนาหุ่นยนต์หลายแพลตฟอร์ม : ระบบฮาร์ดแวร์ การควบคุมและแสดงตำแหน่ง

พีรเดช เปรมใจ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิศวกรรมคอมพิวเตอร์และโทรคมนาคม
วิทยาลัยนวัตกรรมด้านเทคโนโลยีและวิศวกรรมศาสตร์
มหาวิทยาลัยธุรกิจบัณฑิตย์
พ.ศ. 2560

A System Design and Implementation for Multi-platform Robots : Hardware, Control, and Positioning

Peradech Permjai

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Engineering

Department of Computer and Telecommunication Engineering

College of Innovative Technology and Engineering,

Dhurakij Pundit University

2017

ใบรับรองวิทยานิพนธ์

วิทยาลัยนวัตกรรมด้านเทคโนโลยีและวิศวกรรมศาสตร์

มหาวิทยาลัยธุรกิจบัณฑิตย์

ปริญญา วิศวกรรมศาสตรมหาบัณฑิต

หัวข้อวิทยานิพนธ์ การออกแบบและพัฒนาหุ่นยนต์หลายแพลตฟอร์ม : ระบบฮาร์ดแวร์การควบคุมและแสดงตำแหน่ง

เสนอโดย นายพีรเดช เปรมใจ

สาขาวิชา วิศวกรรมคอมพิวเตอร์และโทรคมนาคม

อาจารย์ที่ปรึกษาวิทยานิพนธ์ ผู้ช่วยศาสตราจารย์ ดร.ณรงค์เดช กีรติพรานนท์

ได้พิจารณาเห็นชอบโดยคณะกรรมการสอบวิทยานิพนธ์แล้ว

.....ประธานกรรมการ

(รองศาสตราจารย์ ดร.ลัญฉกร วุฒิสิทธิกุลกิจ)

.....กรรมการและอาจารย์ที่ปรึกษาวิทยานิพนธ์

(ผู้ช่วยศาสตราจารย์ ดร.ณรงค์เดช กีรติพรานนท์)

..... กรรมการ

(อาจารย์ ดร.ชัยพร เขมะภาตะพันธ์)

..... กรรมการ

(อาจารย์ ดร.ธนัญ จารุวิทยโกวิท)

วิทยาลัยนวัตกรรมด้านเทคโนโลยีและวิศวกรรมศาสตร์รับรองแล้ว

..... คณบดีวิทยาลัยนวัตกรรมด้านเทคโนโลยีและวิศวกรรมศาสตร์

(ผู้ช่วยศาสตราจารย์ ดร.ณรงค์เดช กีรติพรานนท์)

วันที่ 22 เดือน ก.ค. พ.ศ. 2560

หัวข้อวิทยานิพนธ์	การออกแบบและพัฒนาหุ่นยนต์หลายแพลตฟอร์ม : ระบบฮาร์ดแวร์การควบคุมและแสดงตำแหน่ง
ชื่อผู้เขียน	พีรเดช เปรมใจ
อาจารย์ที่ปรึกษา	ผศ. ดร.ณรงค์เดช กีรติพรานนท์
สาขาวิชา	วิศวกรรมคอมพิวเตอร์และโทรคมนาคม
ปีการศึกษา	2559

บทคัดย่อ

การเคลื่อนที่ของหุ่นยนต์แบบล้อมีหลายรูปแบบเช่นการเคลื่อนที่ แบบ Differential Drive เป็นการใช้ความเร็วที่แตกต่างกันของล้อ 2 ล้อ และการเคลื่อนแบบ Mecanum drive ใช้ความเร็วที่แตกต่างกันของล้อ 4 ล้อ ซึ่งการเปลี่ยนรูปแบบการเคลื่อนที่มีจำนวนมอเตอร์ที่แตกต่างกัน จะต้องมีการเปลี่ยนโปรแกรมในการสั่งงานการเคลื่อนที่ ซึ่งมีความซับซ้อนและใช้เวลานาน จึงควรมีวงจรที่สามารถควบคุมรองรับการสั่งงานมอเตอร์ได้หลายตัวและ โปรแกรมส่วนกลาง เป็นตัวกลางในการจัดการคำสั่งที่สั่งงานส่วนของการเคลื่อนที่ การอ่านค่า เซนเซอร์ ทำให้สามารถสั่งงานด้วยคำสั่งเดียวกันไปยังหุ่นยนต์แต่ละรูปแบบ และได้ผลการทำงานที่เหมือนกัน ระบบที่ออกแบบสามารถแสดงสถานะและตำแหน่งของหุ่นยนต์ โดยข้อมูลดังกล่าวถูกเก็บไว้ในฐานข้อมูล ผู้ดูแลระบบสามารถดูข้อมูลจากเครื่องคอมพิวเตอร์ได้ทุกที่ เมื่อเข้าถึงระบบสารสนเทศจะแสดงสถานะของหุ่นยนต์และตำแหน่งปัจจุบันบนแผนที่โรงงานในแบบเรียลไทม์ผ่านเว็บเบราเซอร์

วงจรและระบบควบคุมที่ได้จัดทำขึ้นสามารถลดระยะเวลาในการเปลี่ยนแพลตฟอร์มจากรูปแบบหุ่นยนต์ Mecanum Drive เป็นหุ่นยนต์ Omni Directional Drive ได้ถึง 50 เปอร์เซ็นต์ โดยอาศัยการทำงานร่วมกันของวงจรที่สามารถลดความซับซ้อนในการเชื่อมต่อวงจรอิเล็กโทรนิค

Thesis Title	A System Design and Implementation for Multi-platform Robots : Hardware, Control, and Positioning
Author	Peradech Permjai
Thesis Advisor	Asst. Prof. Narongdech Keeratipranon, Ph.D
Department	Computer and Telecommunication Engineering
Academic Year	2016

ABSTRACT

Mobile robot movement has many forms such as two wheel drive with different motor speeds, Mecanum drive with three motors. To switch between platforms, developer must rewrite motion control part which is complicated and time-consuming task. This problem can be alleviated if there is a middleware to handle this control layer. A set of sensor enable the command to be executed using the same command for each robot and obtain the same result. The proposed system can be designed to display the robot's status and position. Such information can be collected in a database, and the administrator can view from a PC anywhere. When accessed, the system shows the robot's status and the current position on the factory map in real-time via a web browser.

This system reduces the time to change the robot's motion pattern by approximate 50%. This is possible because of the ability of circuit hardware and control systems.

กิตติกรรมประกาศ

การจัดทำวิทยานิพนธ์ฉบับนี้ สำเร็จได้ด้วยความช่วยเหลือจากบุคคลหลายฝ่ายเป็นอย่างดี ผู้วิจัยขอขอบพระคุณ ผศ. ดร.ณรงค์เดช กีรติพรานนท์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่คอยให้ข้อคิดเห็นและคำแนะนำที่เป็นประโยชน์ต่องานวิจัย และขอบคุณ อาจารย์ ดร.ประศาสน์ จันทราทิพย์ ที่ให้ความคิดเห็นในการปรับปรุงวิทยานิพนธ์ ขอบคุณ อาจารย์ รศ.ดร.ลัญฉกร วุฒิสิทธิกุลกิจ ดร.ชัยพร เขมะภาตะพันธ์ และ อาจารย์ ดร.ธนัญ จารุวิทยโกวิท ซึ่งได้ให้คำชี้แนะแก้ไขข้อบกพร่องของวิทยานิพนธ์ฉบับนี้จนสำเร็จสมบูรณ์ด้วยดี

ส่วนหนึ่งของความสำเร็จในครั้งนี้ก็มาจากเพื่อนๆ ที่คอยแนะนำ และเป็นกำลังใจ ชมรมโรบอทลูกเจ้าแม่คลองประปาสนับสนุนอุปกรณ์สถานที่ รวมถึงคุณแม่และคุณพ่อที่ให้การสนับสนุนและเป็นกำลังใจให้ตลอดระยะเวลาที่จัดทำวิทยานิพนธ์ฉบับนี้

ท้ายสุดนี้ประโยชน์และความดีของวิทยานิพนธ์ฉบับนี้ ขอมอบแด่ผู้มีพระคุณทุกๆ ท่าน ที่ได้ให้ความช่วยเหลือและเสริมสร้างกำลังกาย กำลังใจให้ จนการจัดทำวิทยานิพนธ์ฉบับนี้สำเร็จลงได้ด้วยดี

พีรเดช เปรมใจ

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญภาพ

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันหุ่นยนต์ถูกแบ่งออกเป็น 2 ประเภทใหญ่ๆ หุ่นยนต์ที่ตั้งอยู่กับที่ (Fixed Robot) กับหุ่นยนต์ที่สามารถเคลื่อนที่ได้ (Mobile Robot) ซึ่งปัจจุบันหุ่นยนต์ที่เคลื่อนที่ได้ เริ่มมีบทบาทมากขึ้นทั้งในด้านอุตสาหกรรม และหุ่นยนต์ที่ใช้ในครัวเรือน หุ่นยนต์ที่ใช้งานในปัจจุบันจึงมีหลายรูปแบบที่ไม่เหมือนกัน เช่น หุ่นยนต์ดูดฝุ่นถูกออกแบบมาใช้ในครัวเรือนจะต้องมีความปลอดภัย การใช้งานง่ายไม่ซับซ้อนสามารถเข้าใจการทำงานได้ในทันที เมื่อเปรียบเทียบกับหุ่นยนต์ขนส่งสินค้าในโรงงาน(AGV)ที่จะต้องมีระบบที่คอยควบคุมจัดการการรับส่งอุปกรณ์ ระบบระบุตำแหน่งที่แม่นยำ และประสิทธิภาพของหุ่นยนต์ที่สามารถทำงานต่อเนื่องเป็นระยะเวลานาน หุ่นยนต์ทั้งสองแบบมีส่วนต่างๆ เช่น เซนเซอร์, Motor, อุปกรณ์ ที่แตกต่างกัน

การเคลื่อนที่ของ Mobile Robot มีหลายรูปแบบ เช่น การเคลื่อนที่ แบบ 2 ล้อโดยใช้ความเร็วแตกต่างกัน (Different Drive) การเคลื่อนที่ 4 ล้อ Mecanum drive ซึ่งการเปลี่ยนแพลตฟอร์มที่มีจำนวนมอเตอร์ที่แตกต่างกัน จะต้องมีการเปลี่ยนโปรแกรมในการสั่งงานการเคลื่อนที่มีความซับซ้อนและใช้เวลานาน จึงควรมีโปรแกรมส่วนกลาง (Middleware) เป็นตัวกลางการจัดการคำสั่งที่สั่งงานส่วนของการเคลื่อนที่ การอ่านค่า เซนเซอร์ ทำให้สามารถสั่งงานด้วยคำสั่งเดียวกันไปยังหุ่นยนต์แต่ละรูปแบบ และได้ผลการทำงานที่เหมือนกัน

ในการศึกษาวิจัยฉบับนี้ได้ทำการศึกษา การออกแบบระบบฮาร์ดแวร์ การควบคุมจึงต้องมีระบบ high level ในการจำลองการสั่งงาน เพื่อทดสอบการทำงานของ Middleware โดยมีการระบุตำแหน่งจากการสั่งงานการเคลื่อนที่ที่แตกต่างกันได้อย่างถูกต้อง

1.2 วัตถุประสงค์ของการวิจัย

1. ออกแบบและสร้างวงจรที่ใช้ในการขับเคลื่อน และโครงสร้างหุ่นยนต์ ที่สามารถประยุกต์ใช้ได้หลายรูปแบบ
2. สร้างระบบจัดการคำสั่งในการเคลื่อนที่ Middleware เพื่อสั่งงาน Low Level
3. สร้างระบบที่คอยตรวจสอบ สถานะของการทำงาน และแสดงตำแหน่งของหุ่นยนต์

1.3 ขอบเขตของการวิจัย

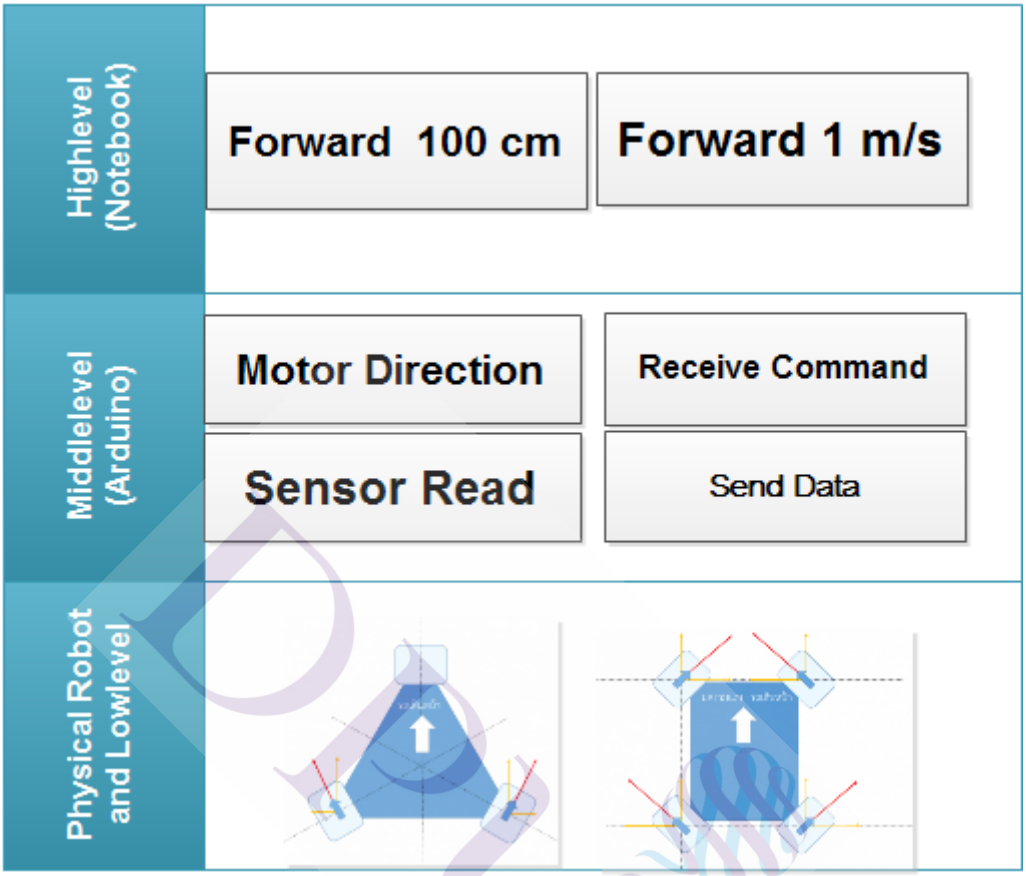
งานวิจัยนี้เป็นการจำลองสถานการณ์และวิเคราะห์ประสิทธิภาพการทำงานของหุ่นยนต์โดยมีสภาวะแวดล้อมจำลองที่กำหนดไว้ในสมมติฐานการวิจัย

1. สร้างวงจรขับมอเตอร์ 12-24 โวลต์ ขับกระแสได้มากกว่า 10 แอมป์
2. สร้างวงจรที่สามารถรองรับการสั่งงานมอเตอร์ได้ 1-4 ตัวพร้อมกัน
3. พัฒนาหุ่นยนต์เคลื่อนที่แบบ Omni Directional Drive และ Mecanum drive
4. พัฒนา Libraries สำหรับสั่งงาน Low Level
5. พัฒนาระบบจัดการคำสั่งในการเคลื่อนที่ของหุ่นยนต์เคลื่อนที่แบบ Omni Directional Drive และ Mecanum drive
6. พัฒนาระบบสารสนเทศที่แสดงตำแหน่ง หรือ สถานะของหุ่นยนต์

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. นำหุ่นยนต์ที่ได้พัฒนาขึ้นมาประยุกต์การใช้งานกับ อุปกรณ์ หรือ เซนเซอร์ เช่น ระบบ Navigation
2. ช่วยลดระยะเวลาในการพัฒนาเปลี่ยนแปลงรูปแบบการเคลื่อนที่ของ Mobile Robot
3. สามารถนำระบบการตรวจสอบสถานะหรือตำแหน่งมาช่วยปรับปรุงวิเคราะห์เพื่อเพิ่มประสิทธิภาพของหุ่นยนต์

1.5 ภาพรวมระบบ



ภาพที่ 1.1 ภาพรวมการออกแบบของหุ่นยนต์

งานวิจัยออกแบบเป็น 3 ส่วน ส่วนที่ 1 Physical Robot and Low level คือโครงสร้างและวงจรขับมอเตอร์ ส่วนที่ 2 Middle level ทำหน้าที่รับคำสั่งจาก High level เพื่อแปลงคำสั่งเป็นรูปแบบการเคลื่อนที่ ตามรูปแบบของหุ่นยนต์ การอ่านค่า เซนเซอร์ ส่วนที่ 3 High level ทำหน้าที่สั่งงาน Middle level โดยคำสั่งที่สั่งจะเป็นการสั่งเดินหน้าหรือถอยหลัง

1.6 ส่วนงานย่อย

ในการออกแบบ หุ่นยนต์ จะมีการแบ่งส่วนการแบ่งงานเป็น 3 ส่วน หลักๆ ดังต่อไปนี้

1. ส่วนงานด้านโครงสร้าง
2. ส่วนงานด้านอิเล็กทรอนิกส์และโปรแกรม Low level
3. ส่านงานด้านโปรแกรมการสั่งงาน และแสดงผลข้อมูล

1.7 อุปกรณ์ที่ใช้

1. PIPO X7s Dual Boot Mini PC
2. Arduino2560
3. Namiki Motor Omni and Mecanum wheel
4. Hokuyo UTM-30LX Scanning Laser Rangefinder

1.8 โปรแกรมที่ใช้พัฒนา

1. Microsoft Visual Studio
2. Altium Designer Release
3. AVR Studio

1.9 ภาษาคอมพิวเตอร์ที่ใช้พัฒนา

1. ภาษา C/C++ (หุ่นยนต์ส่วน Low Level)
2. ภาษา C# (ส่วน Hi Level)
3. PHP (ส่วนระบบสารสนเทศ)
4. Java (ส่วนระบบสารสนเทศ)
5. Node JS (ส่วนระบบสารสนเทศ)

1.10 แผนงาน

1. ศึกษา ค้นคว้า และรวบรวมงานวิจัยที่เกี่ยวข้องกับ Mobile Robot ทำการศึกษาทฤษฎีและรูปแบบที่เหมาะสม

2. ออกแบบหุ่นยนต์ Mobile Robot ออกแบบพัฒนา วงจร โครงสร้าง การควบคุม การติดต่อสื่อสาร ระบบที่ใช้ในการระบุตำแหน่ง เซนเซอร์ต่างๆ

3. พัฒนาระบบ Platform ระบุตำแหน่ง และส่วนควบคุมการเคลื่อนที่ พัฒนาโปรแกรมที่ใช้ระบุตำแหน่งบนตัว Mobile Robot และส่วนควบคุมการเคลื่อนที่โดยเป็นตัวที่จะคอยสั่งงานหุ่นยนต์โดยรวบรวมข้อมูลต่างๆ เช่น ข้อมูลตำแหน่งปัจจุบัน สิ่งกีดขวาง การเรียกใช้งาน และการบันทึกข้อมูลการใช้งานของ Mobile Robot

4. ออกแบบและพัฒนาระบบสารสนเทศที่ใช้ในการตรวจสอบข้อมูล ออกแบบระบบ ที่ผู้ดูแลระบบสามารถตรวจสอบข้อมูลการทำงานของหุ่นยนต์ ผ่านทางเว็บบราวเซอร์ ระบบจะทำการติดต่อฐานข้อมูลเพื่อแสดงสถานะการทำงานของหุ่นยนต์ในขณะนั้น

5. สร้างสถานการณ์จำลองให้กับระบบ หลังจากทำการออกแบบและรวบรวมข้อมูลที่ได้จากการใช้งาน นำข้อมูลเหล่านั้นมาทำการจำลองระบบเพื่อแสดงในเห็นถึงประสิทธิภาพและการทำงานของระบบ

6. เปรียบเทียบ วิเคราะห์ผลที่ได้ และสรุป เมื่อทำการจำลองระบบที่ทำการนำเสนอเสร็จแล้ว และทำการวิเคราะห์เปรียบเทียบถึงประสิทธิภาพของระบบที่ทำการนำเสนอ เพื่อที่จะสรุปผลการจำลองระบบว่าประสิทธิภาพของระบบนั้นเป็นอย่างไร

7. รวบรวมข้อมูลที่ได้ทั้งหมดจัดทำวิทยานิพนธ์ ทำการรวบรวมข้อมูลของระบบที่ทำการนำเสนอ ที่ได้ทำมาตั้งแต่ต้นเพื่อจัดทำเป็นวิทยานิพนธ์

1.11 การทดสอบ

การทดสอบการทำงานของหุ่นยนต์ จะทำการทดสอบเป็น 3 ระยะ โดยจะแบ่งตามลักษณะการทำงาน

ระยะแรกทดสอบความสามารถของวงจร ทดสอบการตอบสนองต่อคำสั่งและความถูกต้องในการทำงานของวงจรขับเคลื่อน

ระยะที่สองทดสอบระบบการสั่งงานการเคลื่อนที่โดยจำลองสภาพแวดล้อมที่เหมาะสมและทดสอบความถูกต้อง ความคลาดเคลื่อน ในขณะที่หยุดนิ่งและเคลื่อนไหว

ระยะที่สามทดสอบการรับส่งและแสดงผลข้อมูล โดยหาความเร็วที่เหมาะสมในการรับส่งข้อมูลที่ไม่กระทบกับการทำงานของหุ่นยนต์

บทที่ 2
แนวคิด ทฤษฎี และงานวิจัยที่เกี่ยวข้อง

หุ่นยนต์ถูกนำมาใช้งานแพร่หลาย โดยเฉพาะด้านอุตสาหกรรม ที่มีการพัฒนามาอย่างต่อเนื่องกว่า 50 ปี และมีแนวโน้มเพิ่มสูงขึ้นโดยหุ่นยนต์ทั่วโลกมีจำนวนทั้งสิ้น 8.6 ล้านตัว ในปี 2008 ทั้งนี้บริษัทขนาดใหญ่จำนวนมากได้ลงทุนด้านเทคโนโลยีหุ่นยนต์เพื่อปรับปรุงกระบวนการผลิตของตน ตัวอย่างเช่นบริษัท Foxconn[1] ซึ่งเป็นบริษัทผลิตอุปกรณ์อิเล็กทรอนิกส์ขนาดใหญ่ที่สุดของประเทศไต้หวัน วางแผนว่าในปี 2014 จะใช้หุ่นยนต์ช่วยในกระบวนการผลิตของตนจำนวน 1 ล้านตัว จากปัจจุบันที่มีการใช้งานหุ่นยนต์เพียง 10,000 ตัว และจากผลสำรวจของ International Federation of Robotics ของเยอรมนี ชี้ว่า ประเทศจีนกำลังจะเป็นยักษ์ใหญ่ด้านการใช้หุ่นยนต์ในการผลิต เห็นได้จากในปี 2014 จีนมีการเติบโตของการใช้หุ่นยนต์ในการผลิตถึงร้อยละ 54 ในขณะที่ทั่วโลกมีหุ่นยนต์เพื่อการผลิตกว่า 2.25 แสนเครื่อง แต่เติบโตเฉลี่ยที่ร้อยละ 27 ในทุก 10,000 โรงงานของจีน มีหุ่นยนต์เพื่อการผลิตราว 30 แห่ง นอกจากนี้คาดการณ์ว่าเยอรมนีซึ่งใช้หุ่นยนต์เพื่อการผลิตมากกว่าจีน 10-11 เท่า จะต้องยกตำแหน่งผู้นำของสายการผลิตโดยใช้หุ่นยนต์ให้จีนภายในปี 2017

2.1 การใช้งาน Mobile Robot

ปัจจุบันหุ่นยนต์ถูกแบ่งออกเป็น 2 ประเภทใหญ่ๆ ตามการเคลื่อนที่ได้แก่ หุ่นยนต์ที่ตั้งอยู่กับที่ Fixed robot หุ่นยนต์ประเภทนี้มีลักษณะโครงสร้างที่ใหญ่โต และมีน้ำหนักมาก ใช้พลังงานจากภายนอก มีการกำหนดขอบเขตพื้นที่ในการเคลื่อนไหว ตัวอย่างการใช้งานของหุ่นยนต์ประเภทนี้ เช่น แขนกลของหุ่นยนต์ที่ใช้ในด้านการแพทย์ แขนกลที่ใช้ในงานด้านอุตสาหกรรมด้านการผลิตหุ่นยนต์ กับหุ่นยนต์ที่สามารถเคลื่อนที่ได้ Mobile robot เป็นหุ่นยนต์ที่สามารถเคลื่อนที่ได้อย่างอิสระ มีการเคลื่อนที่ไปมาในสถานที่ต่างๆ จึงถูกออกแบบลักษณะโครงสร้างให้มีขนาดเล็ก น้ำหนักเบา และมีแหล่งพลังงานในตัวเอง

---

[1] Innovation brings rewards for Siasun, China Daily [ออนไลน], สืบค้นเมื่อ 7 เมษายน 2558, จาก http://africa.chinadaily.com.cn/weekly/2015-04/17/content_20454360.htm

ภาพที่ 2.1 Robot Arm ที่ใช้ในโรงงานอุตสาหกรรม

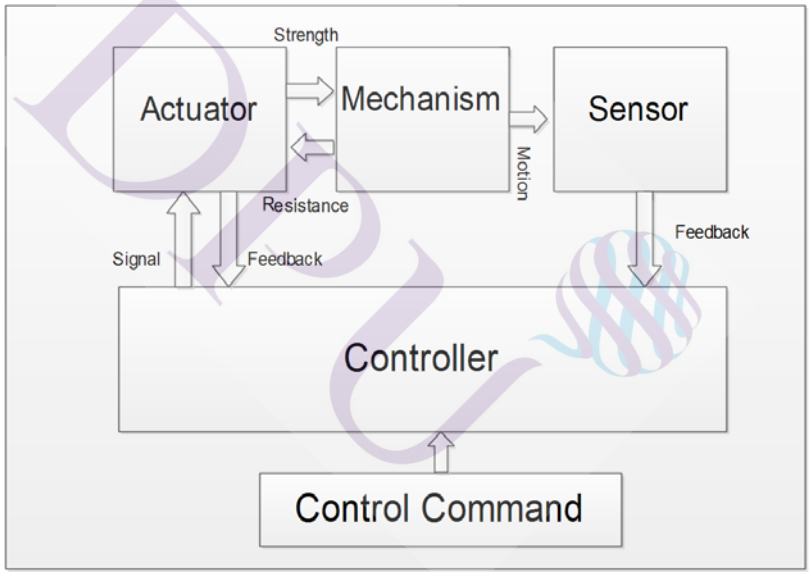
ปัจจุบันการใช้งาน Mobile Robot มีแนวโน้มมากขึ้นเมื่อเทียบกับ Fixed robot ลักษณะการใช้งาน[2] Mobile robot ในด้านการวิจัย การสำรวจพื้นที่ ที่มนุษย์ไม่สามารถปฏิบัติงานได้ เช่น ใต้ทะเลที่มีความลึกมาก

ภาพที่ 2.2 Rescue Robot AGV

[2] Mobile robot, wikipedia [ออนไลน], สืบค้นเมื่อ 1 มกราคม 2560, จาก https://en.wikipedia.org/wiki/Mobile_robot

ด้านอุตสาหกรรม หุ่นยนต์ (AGV) ใช้ในการขนส่งชิ้นงานระหว่างสายการผลิต ด้านความมั่นคง หุ่นยนต์ Rescue Robot หุ่นยนต์ทำลายวัตถุระเบิด ด้านความบันเทิง ใช้ในการประชาสัมพันธ์ หุ่นยนต์ ดินสอ ที่สามารถโต้ตอบกับผู้คนได้ ด้านการใช้งานในครัวเรือน หุ่นยนต์ดูดฝุ่น หรือ หุ่นยนต์ทำความสะอาดสระว่ายน้ำของ iRobot การใช้งานหุ่นยนต์ในด้านต่างๆ มีเพิ่มมากขึ้นเรื่อยๆ จากอดีตหุ่นยนต์ส่วนใหญ่ถูกใช้ในงานอุตสาหกรรม แต่ในปัจจุบันหุ่นยนต์ที่ถูกผลิตขึ้นมาเพื่ออำนวยความสะดวกหรือเพื่อความบันเทิง จากยอดขายของบริษัท iRobot ที่มุ่งพัฒนาหุ่นยนต์ Mobile Robot ที่ใช้ในครัวเรือนได้ขายหุ่นยนต์ที่เลียนแบบการทำความสะอาดของมนุษย์ไปได้มากกว่า 7.5 ล้านครัวเรือน

2.2 การควบคุมการเคลื่อนที่



ภาพที่ 2.3 โครงสร้างระบบควบคุมการเคลื่อนที่

ที่ผ่านมาเทคโนโลยีที่ใช้ในการควบคุมการเคลื่อนที่ได้พัฒนาจากระบบอนาล็อกเป็นระบบควบคุมแบบดิจิทัล ซึ่งจะมีชนิดของการเคลื่อนที่อยู่สองแบบคือ การเคลื่อนที่แบบจุดต่อจุด (Point-To-Point Motion) คือการเคลื่อนที่ ไปถึงเป้าหมายโดยสนใจเฉพาะตำแหน่งที่ต้องการ เช่น การเคลื่อนที่ของ เซอร์โวมอเตอร์ การเคลื่อนที่ของกระบอกนิวแมติกส์ แบบที่สองคือ การเคลื่อนที่แบบต่อเนื่อง (Continuous Motion) เป็นการเคลื่อนที่ที่บังคับเส้นทางที่ผ่านตลอดจนถึงเป้าหมายที่ต้องการเคลื่อนไปถึง เช่น การเคลื่อนที่ของหัวกัดในงานกัด (Milling) การเคลื่อนที่ของหุ่นยนต์เป็น

เส้นโค้ง ภายใต้ชนิดของการเคลื่อนที่ทั้งสองแบบจะประกอบด้วย 4 ส่วนหลักได้แก่ กลไก (Mechanism) ตัวตรวจรู้ (Sensor) ตัวขับเคลื่อน (Actuator) และตัวควบคุม (Controller)

2.2.1 กลไก (Mechanism) ที่ทำให้เคลื่อนที่

กลไก หมายถึง ส่วนของอุปกรณ์ที่ทำหน้าที่ส่งผ่านการเคลื่อนที่ ทำให้มีการเปลี่ยนตำแหน่งจากต้นทางไปยังปลายทางของการเคลื่อนที่ กลไกมีประโยชน์ต่อระบบควบคุมการเคลื่อนที่ในหลายบทบาท เช่น การเปลี่ยนรูปแบบการเคลื่อนที่ (Motion Conversion) การเปลี่ยนแกนหมุน (Axis-Of-Rotation Changing) การปรับความได้เปรียบเชิงกล (Mechanical Advantage Adjustment) ตัวอย่างของกลไกที่พบบ่อยคือ cam (เครื่องดัดสปริง) Crank-Slider (เครื่องฉีดพลาสติก) 4-Bar Linkage (หุ่นยนต์แบบ Close Kinematic Chain) เป็นต้น

2.2.2 ตัวตรวจรู้ (Sensor)

บทบาทของตัวตรวจรู้ในระบบควบคุมการเคลื่อนที่ ได้แก่ การตรวจรู้การเคลื่อนที่ (Motion Sensing) ภายในระบบ และตรวจรู้สภาพแวดล้อม (Environment Sensing) ภายนอก ระบบเพื่อให้ทราบสถานภาพที่จำเป็นต่อการควบคุมระบบ เช่น การใช้กล้องจับภาพวัตถุที่ต้องการหยิบกับการใช้ตัวนับรอบ (Encoder) วัดตำแหน่งของกลไก ทำให้สามารถทราบตำแหน่งเป้าหมายที่ต้องการให้กลไกเคลื่อนที่ไปถึง และการเคลื่อนที่ปัจจุบันของกลไก เป็นต้น

2.2.3 ตัวขับเคลื่อน (Actuator) motor กระบอกลม

ตัวขับเคลื่อนทำหน้าที่ป้อนแรงที่ใช้ เพื่อให้กลไกสามารถเคลื่อนที่ได้ โดยลักษณะแรงที่ป้อนให้จะแบ่งออกเป็นแรงขับ (Force) และแรงบิด (Torque) ซึ่งจะนำไปสู่การเคลื่อนที่ในลักษณะเส้นตรงและลักษณะหมุนการลำดับ ตัวขับเคลื่อนที่ใช้ในระบบควบคุมการเคลื่อนที่โดยทั่วไปจะมี 3 ชนิด ได้แก่ ชนิดขับเคลื่อนด้วยพลังงานนิวเมติก เหมาะกับงานเบาและต้องการอิสระในการติดตั้ง ชนิดขับเคลื่อนด้วยพลังงานไฮดรอลิก เหมาะกับงานหนักและต้องการความนิ่มนวล ชนิดขับเคลื่อนด้วยพลังงานไฟฟ้า เหมาะกับการที่ต้องการความละเอียดและต้องการความยืดหยุ่นสูง

a. ชนิดนิวเมติก b. ชนิดไฮดรอลิค c. ชนิดไฟฟ้า

ภาพที่ 2.4 ระบบการเคลื่อนที่

2.2.4 ตัวควบคุม (Controller) หน่วยประมวลผลต่างๆ

บทบาทพื้นฐานของตัวควบคุมต่อการเคลื่อนที่ของระบบมี 3 ด้าน คือ การเชื่อมต่อกับผู้ใช้ (Man Machine Interface) การควบคุมการเคลื่อนที่ (Motion Control) และการควบคุมตามเหตุการณ์ (Discrete-Event Control)

การเชื่อมต่อกับผู้ใช้ ครอบคลุมไปถึงการรับคำสั่งการเคลื่อนที่ การรายงานสภาพของระบบในรูปแบบที่ผู้ใช้ออกแบบไว้ การควบคุมการเคลื่อนที่ คือการควบคุมให้กลไกเคลื่อนที่ตามคำสั่งการเคลื่อนที่ภายใต้อิทธิพลต่างๆ เช่น การเปลี่ยนแปลงภาระงาน การเปลี่ยนแปลงอุณหภูมิ ส่วนการควบคุมตามเหตุการณ์ คือ การควบคุมให้ระบบตอบสนองการควบคุมแบบไม่ต่อเนื่องต่างได้ เช่น การเกิดภาวะฉุกเฉิน

2.3 เมคคานิค ล้อ รูปแบบการเคลื่อนที่ของ mobile robot[3]

รูปแบบการเคลื่อนที่ของหุ่นยนต์ Mobile robot แบ่งตามลักษณะการเคลื่อนที่ได้ 2 รูปแบบดังนี้

1. Holonomics locomotion คือรูปแบบการเคลื่อนที่ของหุ่นยนต์ที่มีองศาอิสระที่ควบคุมได้ เท่ากับองศาอิสระทั้งหมดของหุ่นยนต์ (Controllable degree of freedom = Total degree of freedom) หุ่นยนต์ที่เป็น holonomics จึงสามารถเคลื่อนที่ไปในทิศทางใดๆ ก็ได้

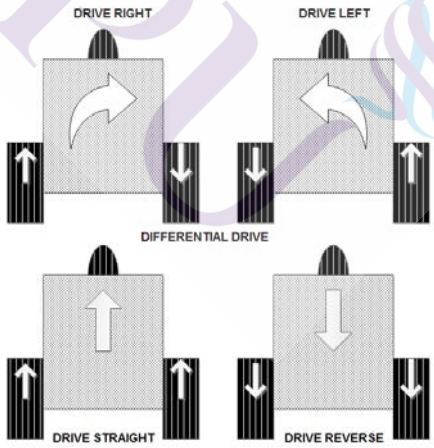
[3] Wheel Control Theory, Robotplatform [ออนไลน], สืบค้นเมื่อ 26 เมษายน 2559, จาก http://www.robotplatform.com/knowledge/Classification_of_Robots/wheel_control_theory.html

2. non-holonomics locomotion คือรูปแบบการเคลื่อนที่ของหุ่นยนต์ที่มีองศาอิสระที่ควบคุมได้น้อยกว่าองศาอิสระทั้งหมดของหุ่นยนต์ (Controllable degree of freedom < Total degree of freedom) หุ่นยนต์ที่เป็น non-holonomic จึงไม่สามารถเคลื่อนที่ไปในทิศทางใดๆ ก็ได้ในทันที ต้องมีการเลี้ยวไปเลี้ยวมาก่อน เช่น การเคลื่อนที่ของรถยนต์

ปัจจุบัน mobile robot ทีใช้ล้อเคลื่อนที่มีการพัฒนาหลายรูปแบบ การควบคุมล้อจะขึ้นอยู่กับรูปแบบของหุ่นยนต์ ซึ่งรูปแบบการเคลื่อนที่แบบต่างๆ ของ Mobile robot สามารถอธิบายได้ภายใต้ลักษณะของล้อที่ใช้ดังนี้

2.3.1 Different Drive (แบบ 2 ล้อ)

เป็นรูปแบบการเคลื่อนที่ที่ง่ายที่สุด มีล้อ 2 ล้อ คือล้อซ้ายและล้อขวา ถ้าต้องการเคลื่อนที่ไปตรงๆ ล้อซ้ายและล้อขวาจะหมุนด้วยความเร็วที่เท่ากัน ถ้าต้องการเลี้ยวซ้าย ล้อซ้ายจะหมุนด้วยความเร็วที่น้อยกว่าความเร็วของล้อขวา จะเรียกรูปแบบการเคลื่อนที่แบบนี้ว่า Different Drive ซึ่งจะเห็นว่ารูปแบบการเคลื่อนที่แบบ Different Drive นั้น ไม่สามารถเคลื่อนที่ไปในทิศใดๆ ก็ได้ตามต้องการในทันที ต้องทำการเลี้ยวหมุนตัวก่อน การเคลื่อนที่แบบ Different Drive จึงจัดเป็น non-holonomic

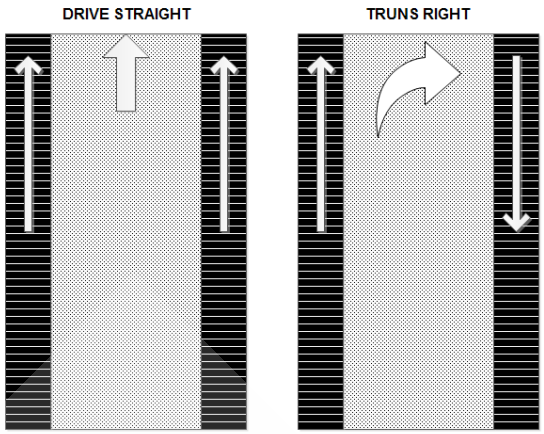


ภาพที่ 2.5 การเคลื่อนที่แบบ Different Drive

2.3.2 Skid steering (มากกว่า 2 ล้อขึ้นไป)

Skid steering มีหลักการคล้ายๆ กับ Different Drive แต่จะแตกต่างกันที่ Skid steering จะใช้กับหุ่นยนต์ที่มากกว่า 2 ล้อ เช่น 4, 6, 8,... ล้อขึ้นไป ในหุ่นยนต์ที่มีล้อมากๆ เวลาเลี้ยวหรือหมุนตัวจะเกิดการลื่นไถล (slip) บนล้อเทียบกับพื้น การ slip นี้ทำให้ Skid steering ใช้พลังงานใน

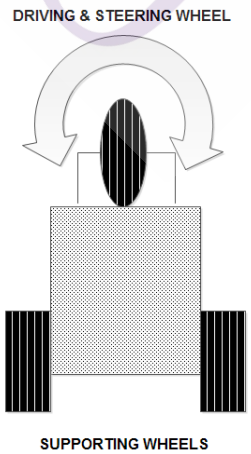
การเลี้ยวมากกว่า Different Drive แต่ด้วยจำนวนล้อที่มากกว่า และมีพื้นที่สัมผัสมากกว่า ทำให้มีแรงขับเคลื่อนมากกว่า



ภาพที่ 2.6 การเคลื่อนที่แบบ Skid steering

2.3.3 Tricycle Drive

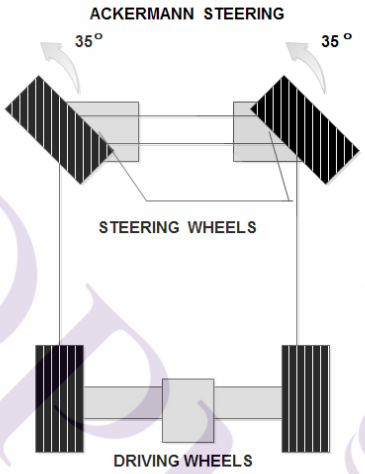
ลักษณะการเคลื่อนที่ของ Tricycle Drive จะเป็นแบบ 3 ล้อ โดยที่ล้อหน้าทำหน้าที่ขับเคลื่อนพร้อมทั้งเลี้ยวด้วย ซึ่งเป็นแบบ สามเหลี่ยม และมีล้อตาม 2 ล้อ



ภาพที่ 2.7 การเคลื่อนที่แบบ Tricycle Drive

2.3.4 Ackermann steering

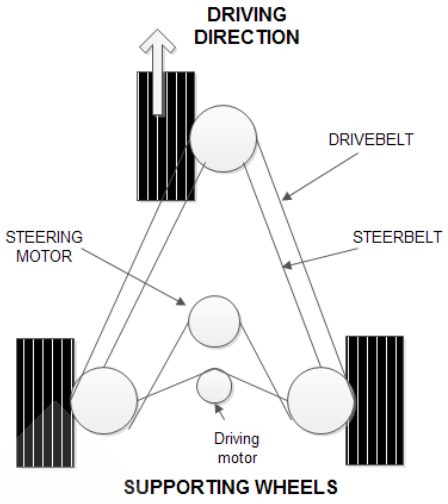
Ackermann Steering มีการเคลื่อนที่แบบรถยนต์ ประกอบด้วยล้อขับเคลื่อนและล้อเลี้ยว (บางครั้งล้อขับเคลื่อนและล้อเลี้ยวเป็นล้อเดียวกัน ในกรณีนี้จะต้องมีล้อหมุนอิสระคอยประคองด้วย เหมือนรถขับเคลื่อนล้อหน้า) เวลาเลี้ยวจะทำการหมุนล้อเลี้ยวให้ตั้งฉากกับเส้นที่ลากไปสู่จุดศูนย์กลางการเลี้ยว เพื่อทำให้ล้อทุกล้อเคลื่อนที่ไปตามเส้นโค้งของการเคลื่อนที่โดยไม่มีการ slip การที่ต้องหมุนล้อไปมาเพื่อเลี้ยว ทำให้ไม่สามารถเปลี่ยนทิศการเคลื่อนที่ไปทิศใดๆ ตามต้องการได้ทันที ทำให้ Ackermann steering เป็น non-holonomic



ภาพที่ 2.8 การเคลื่อนที่แบบ Ackermann steering

2.3.5 Synchronous Drive

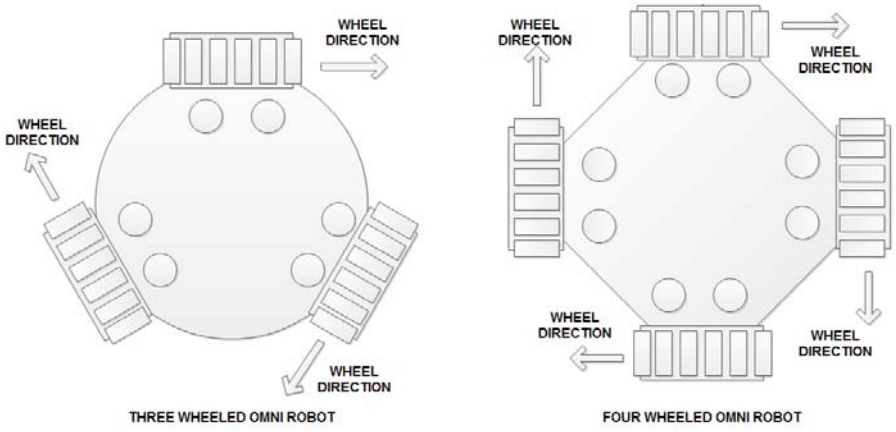
หลักการเคลื่อนที่ของ Synchronous Drive มีสายพานหรือ โซ่ หมุนทิศทางของล้อไปทิศทางเดียวกันเพื่อบังคับทิศทางการเคลื่อนที่ และล้อทุกล้อเคลื่อนที่โดยมอเตอร์อีกชุดหรือมอเตอร์ที่อยู่ติดกับล้อ



ภาพที่ 2.9 การเคลื่อนที่แบบ Synchronous Drive

2.3.6 Omni Directional Drive

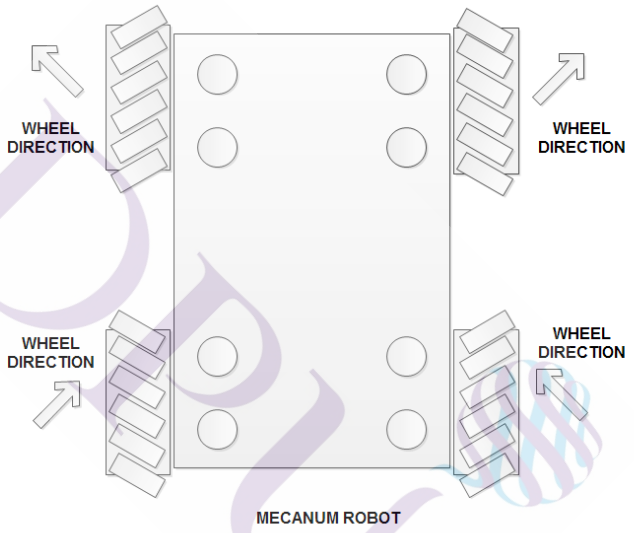
ล้อของ Omni drive จะมีลักษณะพิเศษคือ มี roller อยู่รอบๆ ล้อ ในทิศทางการหมุนที่ตั้งฉากกับการหมุนของล้อ การเคลื่อนที่แบบ Omni drive ต้องมีล้ออย่างน้อย 3 ล้อ วางทำมุมกัน เนื่องจากในแต่ละล้อมี roller อยู่ จึงเกิดการเคลื่อนที่ได้ 2 แกนพร้อมๆ กันในแต่ละล้อ ได้แก่ การเคลื่อนที่ตามแนวการหมุนของล้อ และการเคลื่อนที่ตามแนวการหมุนของ roller เมื่อต้องการเคลื่อนที่ไปยังทิศใดๆ ก็สามารถแตกความเร็วการเคลื่อนที่เข้าแนวการเคลื่อนที่ของล้อและของ roller ได้เสมอ ทำให้ Omni drive สามารถเคลื่อนที่ไปยังทิศใดๆ ก็ได้ เป็น holonomic การเคลื่อนที่ของ Omni drive จะต้องมีสมการที่เรียกว่า Jacobian/Inverse-jacobian matrix เพื่อแปลงความเร็วของหุ่นยนต์ ไปสู่ความเร็วของล้อแต่ละล้อ



ภาพที่ 2.10 การเคลื่อนที่แบบ Omni Directional Drive

2.3.7 Mecanum drive

ลักษณะการเคลื่อนที่ของ Mecanum drive ต่างกับ Omni drive ตรงที่ roller บนล้อไม่วางตั้งฉากกับล้อ แต่จะวางทำมุมกัน ส่วนมากจะทำมุม 45 องศา และล้อสามารถวางตัวขนานกันได้ตามปกติ ไม่ต้องวางทำมุมกัน การเคลื่อนที่จะใช้หลักการเดียวกับ Omni drive จึงต้องมี Jacobian/Inverse-jacobian matrix ในการคำนวณความเร็วล้อเช่นกัน แต่สมการจะง่ายกว่า Mecanum drive เป็นการเคลื่อนที่แบบ holonomic การเคลื่อนที่แบบ Mecanum drive จะให้แรงขับเคลื่อนเยอะในแนวการเคลื่อนที่ด้านหน้าและแนวด้านหลัง แต่การเคลื่อนที่ในแนวซ้าย และขวา จะต้องใช้พลังงานมากกว่า การวางตัวของล้อจะเหมาะสำหรับหุ่นยนต์ทรงสี่เหลี่ยม



ภาพที่ 2.11 การเคลื่อนที่แบบ Mecanum drive

2.4 Design ภาพรวม

หุ่นยนต์ที่ Mobile Robot ที่ถูกผลิตออกมาในปัจจุบันมีการออกแบบ ในส่วนสำคัญอยู่ 3 ส่วน ระบบการเคลื่อนที่ (Mechanism) ระบบการรับรู้ (Sensor) และ ระบบการสั่งงานวางแผนและตัดสินใจ (Application) การออกแบบหุ่นยนต์จึงขึ้นอยู่กับการนำไปใช้งาน ปัจจุบันมีการนำ Mobile Robot มาใช้ในอุตสาหกรรมหุ่นยนต์ KIVA[4] ของบริษัท Amazon ที่ใช้จัดการคลังสินค้า ช่วยเพิ่ม

[4] From “Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses,” by Peter R. Wurman, Raffaello D’Andrea, and Mick Mountz. (2007, July 13–17). AAAI 2007, pp. 1752-1759.

ประสิทธิภาพในศูนย์กระจายสินค้า หุ่นยนต์ TETRA-DS IV ที่ผลิตเพื่อใช้ในการพัฒนา Mobile Robot (Mobile Robot Platform)

ภาพที่ 2.12 หุ่นยนต์ KIVA ของ Amazon (ซ้าย) หุ่นยนต์ TETRA-DS IV(ขวา)

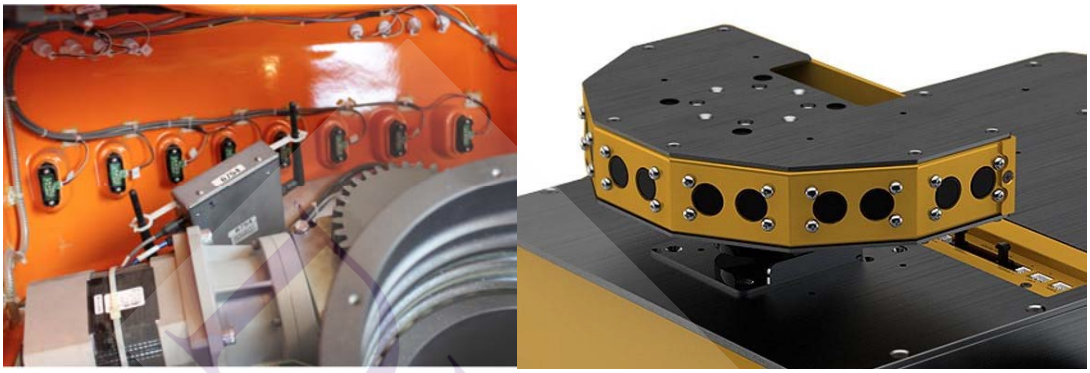
กลไกการเคลื่อนที่ของหุ่นยนต์[5] Kiva และ TETRA-DS IV ใช้การเคลื่อนที่แบบ differential drive โดยหุ่นยนต์ Kiva ในแต่ละล้อขับเคลื่อนด้วยมอเตอร์กระแสตรงไร้แปรงถ่าน (brushless DC motor) Pittman กำลัง 1 กิโลวัตต์ ต่อผ่านชุดเฟือง Brother อัตราทด 25:1 ไปยังล้อที่ทำขึ้นมาเฉพาะ ที่ด้านหน้าและหลังมีล้อเป็น (caster) คอยประคองอยู่ โดย caster ด้านหนึ่งมีระบบกันสะเทือนอยู่เพื่อให้ล้อทั้งหมดแตะพื้นส่วนในหุ่น TETRA-DS IV ใช้มอเตอร์กระแสสลับไร้แปรงถ่าน (brushless AC motor) กำลัง 100 วัตต์ อัตราทด 15:1

ภาพที่ 2.13 ระบบการเคลื่อนที่ของหุ่นยนต์ KIVA

---

[5] Meet the drone that already delivers your packages, Kiva robot teardown, Robohub [ออนไลน], สืบค้นเมื่อ 1 กุมภาพันธ์ 2559, จาก http://robohub.org/meet-the-drone-that-already-delivers-your-packages-kiva-robot-teardown/

กลไกป้องกันการวิ่งไปชนสิ่งกีดขวางของหุ่นยนต์ Kiva และ TETRA-DS IV มีอยู่ 2 ชนิดโดยในหุ่นยนต์ Kiva จะใช้ Infrared distance sensor เพื่อตรวจสอบสิ่งกีดขวางระยะ 8-15 เซนติเมตร และมีกันชน ออกแบบโดยใช้ท่ออัดลมติดรอบๆ เมื่อมีการชนแรงดันในท่อมะมีการเปลี่ยนแปลง ทำให้สามารถตรวจจับการชนได้ ส่วนในหุ่นยนต์ TETRA-DS IV ใช้ Ultrasonic distance sensor ในการตรวจสอบสิ่งกีดขวางระยะ 2-400 เซนติเมตร และใช้ Limit switch ต่อกับกันชนเพื่อตรวจสอบการชนระยะใกล้



ภาพที่ 2.14 IR distance sensor และ Ultrasonic sensor

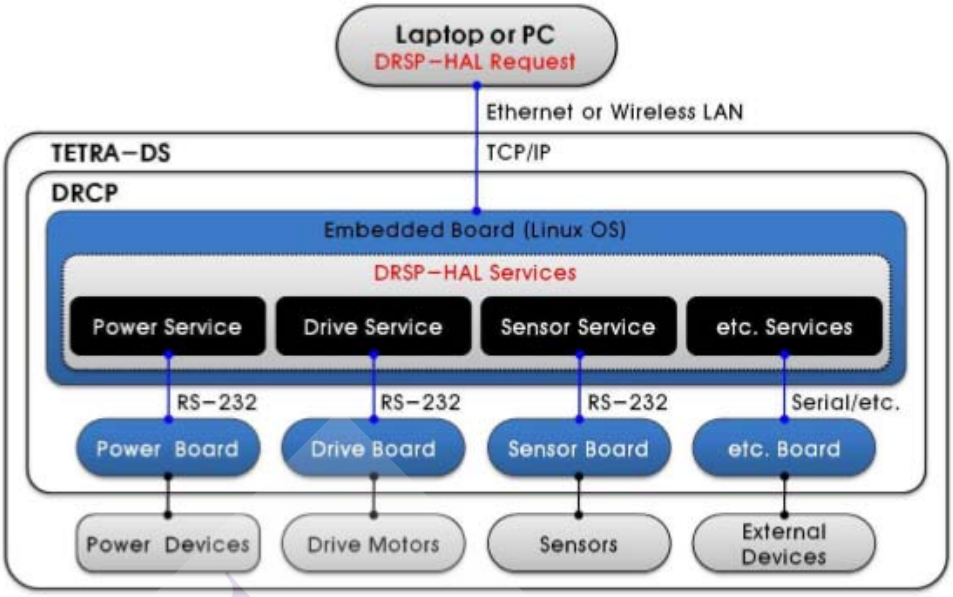
ภาพที่ 2.15 อุปกรณ์ตรวจจับการชนระยะใกล้

ระบบ ควบคุมหุ่นยนต์ คือ วงจร (onboard controller) ที่ตัดสินใจจากข้อมูลที่ได้รับจากเซนเซอร์ หรือคำสั่งจากผู้ใช้สั่งงานไปยังมอเตอร์ เพื่อควบคุมการเคลื่อนที่ ระบบความคุมหุ่นยนต์จึงต้องมีระบบย่อยๆ การเชื่อมต่อ I/O ระบบควบคุมจะต้องสื่อสารกับ เซนเซอร์ และอุปกรณ์

ขับเคลื่อน ตัวอย่าง เซนเซอร์ ที่สำคัญเช่น LIDAR และ GPS จะติดต่อสื่อสารโดยการเชื่อมต่อแบบอนุกรม (Serial) อุปกรณ์ควบคุมการขับเคลื่อน ต้องการ Digital พอร์ต หรือ CAN BUS ในการสั่งงาน

ภาพที่ 2.16 วงจรควบคุมมอเตอร์ และชุดสื่อสารไร้สายของหุ่นยนต์ Kiva

วงจรหลักของ Kiva มีวงจรควบคุมมอเตอร์ไร้แปรงถ่านถูกพัฒนาขึ้นจาก FPGA (Field-Programmable Gate Array) รับสัญญาณป้อนกลับจาก encoder และขับมอเตอร์ผ่าน FET (Field-Effect Transistor) ที่ต่อเป็น full-bridge ตัว FET ถูกติดบนโครงหุ่นเพื่อระบายความร้อน วงจรรองทำหน้าที่เชื่อมต่อกับอุปกรณ์ที่เหลือทั้งหมดเข้ากับวงจรหลัก ตั้งแต่ ชุดสื่อสารไร้สาย กล้อง ปุ่มหยุดฉุกเฉิน เซนเซอร์กันชน ชุดบริหารจัดการพลังงาน วงจรนี้ใช้หน่วยประมวลผลเป็นไมโครคอนโทรลเลอร์ 32 บิต Freescale MPC5123 ความถี่ 400 MHz มี PowerPC เป็นระบบปฏิบัติการ



ภาพที่ 2.17 แสดงตารางการสื่อสารระหว่างอุปกรณ์ของหุ่นยนต์ TETRA-DS IV

2.5 Robotics middleware

เป็นระบบซอฟต์แวร์ที่ถูกออกแบบมาเพื่อจัดการกับความซับซ้อนและความหลายหลายของฮาร์ดแวร์ ลดความซับซ้อนในการสื่อสารในระดับ Low Level ปัจจุบัน มี Robotics middleware อยู่หลายตัวซึ่งหุ่นยนต์จำนวนมากไม่ได้ใช้ middleware ที่มีอยู่ เนื่องจากการทำงานร่วมกันของอุปกรณ์ นักพัฒนาจึงต้องพัฒนา middleware ขึ้นมาด้วยตนเอง Robotics middleware ที่นิยมมากในปัจจุบันคือ ROS (Robot Operating System)[6] ด้วยเหตุผลที่เปิดใช้งานได้ฟรี มีภาษาให้เลือกพัฒนาที่หลากหลาย มีการอัพเดทอยู่ตลอดเวลา

[6] Robot_Operating_System, Wikipedia [ออนไลน], สืบค้นเมื่อ 26 มิถุนายน 2560, จาก .https://en.wikipedia.org/wiki/Robot_Operating_System

ภาพที่ 2.18 โลโก้ของ ROS

จากภาพที่ เป็นภาพโลโก้ของ ROS ที่ได้มีการอัพเดทจากอดีตถึงปัจจุบัน ROS คือ ระบบปฏิบัติการหุ่นยนต์ หรือ Framework ของการพัฒนาหุ่นยนต์ที่ Willow Garage (บริษัทที่วิจัยหุ่นยนต์ ใน USA) คิดค้นขึ้นมา เพื่อใช้กับหุ่นยนต์ PR2[7]

ภาพที่ 2.19 หุ่นยนต์ PR2

[7] PR2, wiki.ros [ออนไลน], สืบค้นเมื่อ 1 กรกฎาคม 2560, จาก http://wiki.ros.org/Robots/PR2

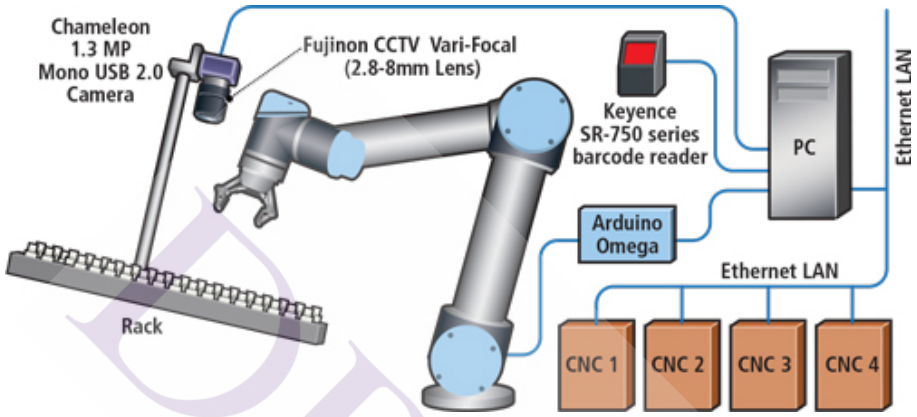
ข้อได้เปรียบของการใช้ระบบปฏิบัติการหุ่นยนต์ (ROS) คือ โครงสร้างจะแบ่งออกเป็น Module โดยจะมีหน้าที่ของมันเองในแต่ละ Module ROS ใช้การติดต่อสื่อสารแต่ละ Module ผ่านตัวกลางของระบบ เช่น ถ้ามีการเปลี่ยนแปลงการใช้งานหุ่นยนต์จากแพลตฟอร์ม 2 ล้อไปเป็น 4 ล้อ ก็สามารถทำได้ง่ายเพราะ ROS สื่อสารข้อมูลผ่านตัวกลาง อีกตัวอย่างของการใช้ระบบปฏิบัติการ ROS ก็คือ โครงการ Million Objects Challenge คือการสอนหุ่นยนต์ Baxter[8] ของบริษัท Rethink Robotics ซึ่งเป็นหุ่นยนต์แบบตั้งโต๊ะที่มีสองแขนและมีมือเป็นปากคีบ สามารถสอนให้ทำงานแบบซ้ำๆ แทนมนุษย์ โดยนักวิจัยจะสอนให้หุ่นยนต์เรียนรู้การหยิบจับวัตถุต่างๆ เมื่อเรียนรู้แล้วก็แชร์โปรแกรมดังกล่าวบนระบบคลาวด์ (Cloud Computing) และให้หุ่นยนต์ตัวอื่นๆ สามารถหยิบจับวัตถุเดียวกันได้โดยไม่ต้องโปรแกรมใหม่ หัวใจสำคัญที่ทำให้เกิดการพัฒนาในลักษณะนี้ได้ คือ หุ่นยนต์ทุกตัวสื่อสารกันได้เข้าใจเพราะหุ่นยนต์ทุกตัวใช้ภาษาสากลของหุ่นยนต์นั่นคือ ROS (Robot Operating System)

ภาพที่ 2.20 ภาพหุ่นยนต์ที่ใช้ในโครงการ Million Objects Challenge

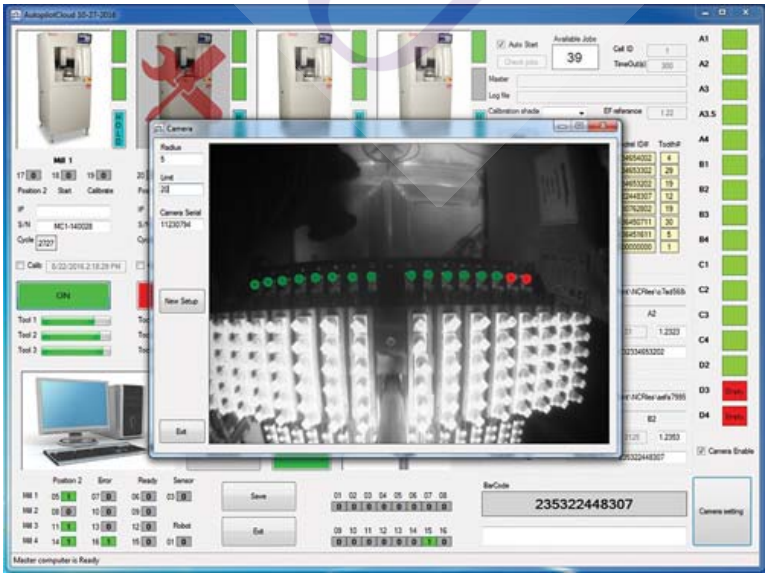
[8] A robot learns: Baxter's video takes first place, News from Brown [ออนไลน], สืบค้นเมื่อ 1 กรกฎาคม 2560, จาก https://news.brown.edu/articles/2015/12/baxter

2.6 การ Monitor หุ่นยนต์

การ Monitor หุ่นยนต์นั้นปัจจุบันจะใช้กับหุ่นยนต์ที่ทำงานตลอดเวลา เช่นในโรงงานอุตสาหกรรม เพื่อไว้คอยตรวจสอบความผิดพลาดหรือเหตุการณ์ที่เกิดขึ้น จากภาพที่ 2.22 ในหุ่นยนต์แบบ Fix robot การ Monitor มีไว้เพื่อตรวจสอบความถูกต้องของการทำงาน สามารถแจ้งให้ผู้ดูแลให้ทราบถึงสถาน ของวัสดุที่จะนำมาเข้าเครื่อง CNC ว่าหมดหรือไม่ โปรแกรมจะทำงานผ่านการเชื่อมต่อ Ethernet LAN ส่งขอมูลไปยังห้องควบคุม



ภาพที่ 2.21 การเชื่อมต่อ หุ่นยนต์ fix Robot



ภาพที่ 2.22 ตัวอย่างหน้าจอโปรแกรม

นอกจากการตรวจสอบการทำงานโปรแกรมที่ถูกสร้างขึ้นยังสามารถสั่งงานตัวหุ่นยนต์ได้ในระดับหนึ่ง ส่วนใน mobile robot เช่นเดียวกับ fix robot แต่การทำงาน จะผ่านสัญญาณ WIFI หรือระบบไร้สายอื่นๆ เพราะหุ่นยนต์จะเคลื่อนที่ โดยจะมีการส่งข้อมูลของ เซนเซอร์ กล้อง หรืออุปกรณ์อื่นๆ

ภาพที่ 2.23 ตัวอย่างหน้าจอโปรแกรมควบคุม

การตรวจสอบการทำงานแบบ web monitor เหมาะกับการแสดงผล ทางสถิติการแสดงจำนวนชิ้นงาน ค่าสถาน ใน mobile robot ไม่นิยมการตรวจสอบการทำงานผ่านระบบ web monitor เพราะจะมีการส่งข้อมูลจำนวนมาก เช่น กล้อง ข้อมูลจะต้อง Real Time ตลอดเวลาทำให้ไม่เหมาะกับการตรวจสอบ ผ่านระบบ web monitor

2.7 การระบุตำแหน่งของหุ่นยนต์

2.7.1 Laser range finder

การที่หุ่นยนต์จะรู้สภาพแวดล้อมต่างๆ ได้นั้น จะต้องอาศัยอุปกรณ์ที่มีความสำคัญต่อการรับรู้ของหุ่นยนต์ ที่เรียกว่า เซนเซอร์ ซึ่งสามารถแบ่งออกตามลักษณะการทำงานได้ 2 ประเภท คือประเภท Active และประเภท Passive เซนเซอร์ประเภท Active จะส่งสัญญาณออกไปแล้วรอการสะท้อนกลับมาของสัญญาณ เช่น เลเซอร์วัดระยะทาง (Laser range finder) ส่วนเซนเซอร์ประเภท Passive จะรับรู้ข้อมูลต่างๆ ได้โดยไม่ต้องส่งสัญญาณออกไปกระตุ้น ตัวอย่างเซนเซอร์ประเภทนี้ได้แก่ กล้องวิดีโอ (Computer vision)

ภาพที่ 2.24 Laser range finder

การระบุตำแหน่งของหุ่นยนต์ (Localization) เป็นกระบวนการที่หุ่นยนต์สามารถบอกตำแหน่งตัวเองได้ว่าอยู่ที่จุดใดในแผนที่ โดยแผนที่นั้นอาจเป็นแผนที่ ที่กำหนดไว้ล่วงหน้าแล้ว หรือเป็นแผนที่ ที่สร้างไปพร้อมๆ กับการระบุตำแหน่ง การระบุตำแหน่งด้วยวิธี Monte Carlo คือ การใช้การแซมเปิ้ลสุ่ม (Random Sample) ในการหาคำตอบสำหรับสมการที่ซับซ้อนด้วยคอมพิวเตอร์ เช่น การคำนวณระบบที่มีความไม่แน่นอนสูง หรือคำนวณได้ยากเช่น กระบวนการเชิงตัวเลขที่ต้องอาศัยสมการทางอินทีกรัล แคลคูลัส Monte Carlo หรือ MC มีชื่อเรียกอีกอย่างว่า Simulation-based ตัวอย่างการหาคำตอบด้วยวิธี Monte Carlo ยกตัวอย่างการหาพื้นที่วงกลม ด้วยรัศมี r สมมติว่าไม่รู้สูตรการคำนวณพื้นที่วงกลม รู้แต่ว่าพื้นที่สี่เหลี่ยมจตุรัสที่ครอบวงกลมพอดี คำนวณ ความน่าจะเป็น (Probability) พื้นที่ของวงกลมได้จาก อัตราส่วนของแซมเปิ้ลที่นับได้ในวงกลมกับแซมเปิ้ลที่กระจายในสี่เหลี่ยมทั้งหมด

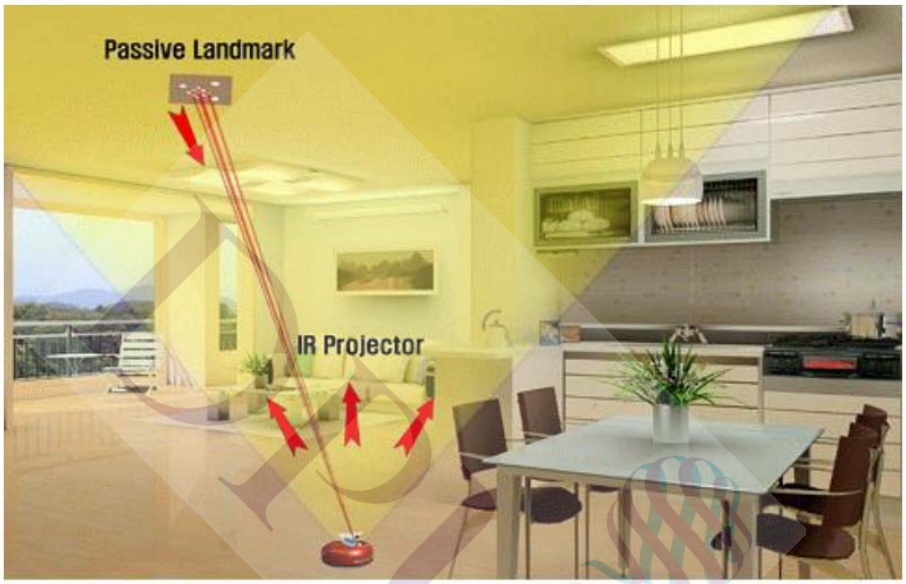
2.7.2 Odometry

การระบุตำแหน่งของหุ่นยนต์ ทำได้โดยวิธีการที่เรียกว่า Odometry คือ การระบุว่าหุ่นยนต์เคลื่อนที่ไปได้ในระยะทางเท่าไร ตำแหน่งและทิศทางต่างๆของหุ่นยนต์เมื่อเทียบกับจุดเริ่มต้น ลักษณะการทำงานของวิธีการ Odometry จะนับจำนวนรอบการหมุนที่ล้อของหุ่นยนต์โดยใช้ Encoder จากนั้นนำข้อมูลที่ได้มาแปลงกลับเป็นค่าระยะทาง ตำแหน่ง และทิศทางการเคลื่อนที่ การระบุตำแหน่งของหุ่นยนต์ด้วยวิธี Odometry นั้น จะมีความถูกต้อง แม่นยำค่อนข้างต่ำ เกิดจากปัญหาทางกายภาพของตัวเซ็นเซอร์ ปัญหาการลื่นไถล (Slip) ของหุ่นยนต์ รวมถึงสัญญาณรบกวนต่างๆ ทำให้ข้อมูลที่ได้มาไม่มีความสมบูรณ์ อีกทั้งในแต่ละช่วงเวลาที่เคลื่อนที่ไป ระบบ

จะสะสมความผิดพลาดเพิ่มขึ้นเรื่อยๆ ซึ่งทำให้หุ่นยนต์มีความผิดพลาดได้ ดังนั้นจำต้องมีการประยุกต์ใช้ เทคนิคหรือวิธีการอื่นๆ ควบคู่ไปด้วยเพื่อให้หุ่นยนต์มีความถูกต้องและแม่นยำมากขึ้น

2.7.3 Landmarks

การระบุตำแหน่งของหุ่นยนต์ โดยใช้กระบวนการ Image processing ตัวอย่างอุปกรณ์ Stargazer จะทำการติด Landmark ไว้บนเพดาน Stargazer จะมี IR Projector ฉายแสงอินฟราเรด ไปสะท้อนกับจุดบน Landmark แล้วจึงรับภาพที่สะท้อนมาประมวลผล



ภาพที่ 2.25 Laser range finder

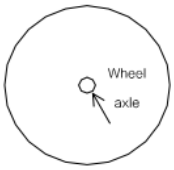
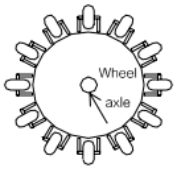
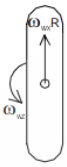
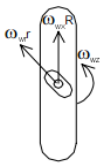
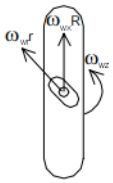
ความสูงที่ระหว่าง Stargazer กับแผ่น Landmark จะขึ้นอยู่กับความแม่นยำในการติดตั้งและจำนวน Landmark เห็นในขณะนั้น โดยแผ่น Landmark ประเภท 4*4 จะสามารถติดได้มากสุดจำนวน 4,095 แผ่น ความผิดพลาดที่วัดได้ในการอ่านค่าของ Landmark จากจุดๆเดียว มีอัตราความคลาดเคลื่อน 2.432 เซนติเมตร และความถี่ในการอ่านค่าเฉลี่ยอยู่ที่ 18.5 ครั้งต่อวินาที

2.8 การเคลื่อนที่ของหุ่นยนต์อุตสาหกรรม

ในปัจจุบันมีการนำ platform การเคลื่อนที่ของ Mecanum Drive มาใช้ในอุตสาหกรรมอย่างแพร่หลาย เนื่องจากข้อได้เปรียบทางด้านการเคลื่อนที่ มีความคล่องตัวมากกว่ารูปแบบอื่นๆ ตัวอย่างเช่น แพลตฟอร์มโหลดน้ำหนักแบบเคลื่อนที่ ระบบการขนส่งสำหรับโลจิสติกส์ KUKA "mobility" ซึ่งนำ platform การเคลื่อนที่แบบ Mecanum Drive มาใช้ เพราะมีอิสระในการเคลื่อนที่, มีความคล่องตัวในการหมุนเลี้ยวหรือปรับเปลี่ยนแนวการเคลื่อนที่ต่างๆ และรับโหลดได้สูงกว่าล้อปกติทั่วไป

ภาพที่ 2.26 แพลตฟอร์มโหลดน้ำหนักแบบเคลื่อนที่ KUKA "mobility"

ตารางที่ 2.1 การเปรียบเทียบล้อปกติ ล้อ Omni และล้อ Mecanum

ประเภทล้อ / หัวข้อ	Conventional wheel ล้อปกติ	Omni-direction wheel ล้อโอมนิ	Mecanum wheel ล้อแมคคานัม
Side view			
ความสัมพันธ์ทางคิเนเมติกของล้อแบบต่างๆ			
DOFs	2 DOFs	3 DOFs	3 DOFs
แนวการเคลื่อนที่	แนวการเคลื่อนที่แรก คือ ทิศทาง ตามแนวการวางล้อ แนวที่ สอง เกิดจากการหมุนที่จุดสัมผัสระหว่างล้อกับพื้น	แนวการเคลื่อนที่แรก คือ ทิศทางตามแนวการวางล้อ แนวที่สองคือการหมุนของลูกกลิ้งที่อยู่รอบๆ ล้อซึ่งทำมุม 90 องศากับล้อหลัก และแนวที่สาม เกิดจากการหมุนที่จุดสัมผัสระหว่างลูกกลิ้งและพื้น	แนวการเคลื่อนที่แรก คือ ทิศทางตามแนวการวางล้อ แนวที่สองคือการหมุนของลูกกลิ้งที่อยู่รอบๆ ล้อทำมุม 45 องศากับล้อหลัก และแนวที่สาม เกิดจากการหมุนที่จุดสัมผัสระหว่างลูกกลิ้งและพื้น

บทที่ 3
การออกแบบและพัฒนา

ในบทนี้จะกล่าวถึงการออกแบบระบบของโครงงานรวมทั้งอธิบายถึง แนวทางการวิจัยและพัฒนา เครื่องมือที่ใช้ในงานวิจัย แผนการดำเนินงาน ขั้นตอนและวิธีการดำเนินงาน

3.1 แนวทางการวิจัยและพัฒนา

งานวิจัยนี้มุ่งเน้นออกแบบหุ่นยนต์ Mobile Robot ที่สามารถนำมาประยุกต์ใช้งานได้หลากหลาย โดยออกแบบวงจรให้สามารถรองรับรูปแบบการทำงานของหุ่นยนต์ Mobile Robot ประเภทเคลื่อนที่ด้วยล้อ ให้สามารถปรับเปลี่ยนและแก้ไขรูปแบบของหุ่นยนต์โดยจะแสดงให้เห็นถึงการประยุกต์ใช้ตั้งแต่รูปแบบการเคลื่อนที่แบบ 3 ล้อ และ 4 ล้อ การเปลี่ยนรูปแบบการเคลื่อนที่ที่ใช้ในงานการแข่งขัน การใช้ในด้านการศึกษาและการทดสอบอุปกรณ์ที่ติดตั้งบนหุ่นยนต์ โดยมีแนวทางการพัฒนาดังนี้ หุ่นยนต์ที่ออกแบบจะแบ่งออกเป็น 5 ส่วนหลักๆ โดยมี 3 ส่วนทำงานอยู่บนตัวหุ่นยนต์และอีก 2 ส่วน ทำงานภายนอกหุ่นยนต์ โดยแบ่งได้ดังนี้

1. Motor Drive Control ทำหน้าที่รับคำสั่งจาก Main Controller มาควบคุมความเร็วให้ได้ตามเป้าหมาย
2. Middle Level Controller ทำหน้าที่ประมวลผล เซนเซอร์, ควบคุมการเคลื่อนที่, และแสดงผล ในระดับ Low Level
3. Robot Processing Unit ทำหน้าที่ระบุตำแหน่ง จาก Localization เซนเซอร์ และส่งข้อมูลการทำงานและจัดเก็บลงฐานข้อมูล
4. Server จัดเก็บข้อมูลของหุ่นยนต์
5. Web browser (GUI) แสดงข้อมูลการทำงานและตำแหน่งของหุ่นยนต์

3.1.1 ศึกษารวบรวมข้อมูล

1) ศึกษารูปแบบของ Mobile Robot ประเภทต่างๆ
2) ศึกษาหาความเหมาะสมในการออกแบบวงจร
3) ศึกษารูปแบบวิธีการใช้งานอุปกรณ์ตรวจสอบตำแหน่ง
4) ศึกษาหาวิธีการในการจัดเก็บตำแหน่งของหุ่นยนต์

3.1.2 การออกแบบระบบงาน

1) ออกแบบวงจรให้สามารถรองรับการทำงานของหุ่นยนต์ ในรูปแบบต่างๆ เช่น การรองรับจำนวนมอเตอร์ จำนวน เซนเซอร์ อุปกรณ์การเชื่อมต่อ แบตเตอรี่

2) ออกแบบโครงสร้างของหุ่นยนต์ที่ใช้ในการทดสอบ ที่สามารถใช้ความสามารถของวงจรที่ออกแบบ และสามารทดสอบการติดตั้ง เซนเซอร์ และอุปกรณ์ต่างๆ

3) ออกแบบโครงสร้างโปรแกรม คำสั่ง Libraries การติดต่อสื่อสารระหว่างอุปกรณ์ต่างๆ

3.1.3 พัฒนาระบบงาน

ทำการพัฒนาระบบให้สามารถทำงานได้ตามวัตถุประสงค์ ทำการทดสอบประสิทธิภาพการทำงานของระบบ รวมทั้งตรวจสอบความถูกต้องในการทำงานของระบบ

3.1.4 ทดสอบการใช้งาน

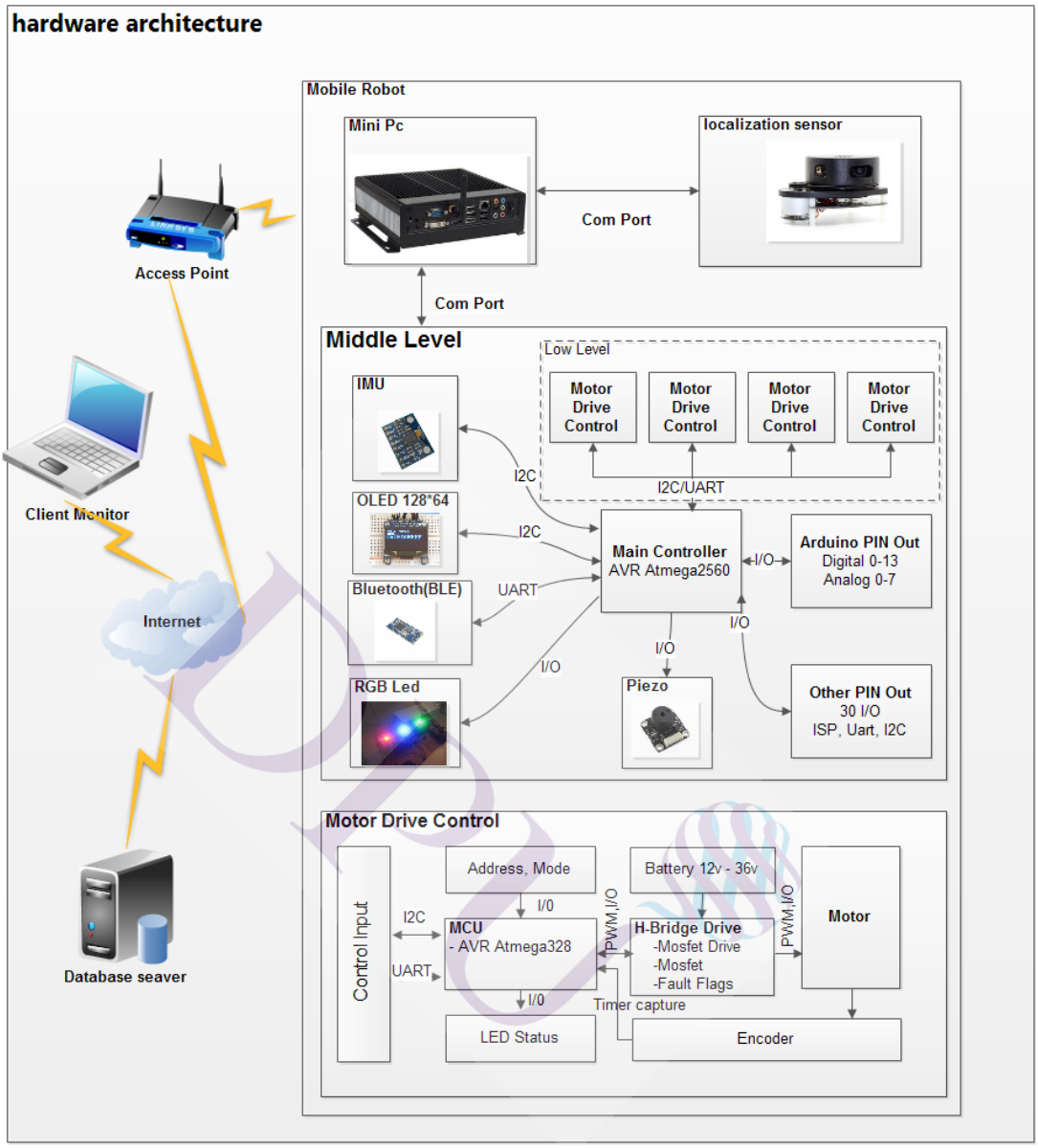
ทดสอบในแต่ละส่วนย่อย และทดสอบการทำงานโดยรวมของระบบโดยสร้างรูปแบบการทดสอบให้ระบบสามารถแสดงประสิทธิภาพของหุ่นยนต์ได้

3.1.5 สรุปผลการพัฒนา

นำข้อมูลที่ได้จากการทดสอบมาสรุปผล เพื่อวิเคราะห์การทำงานและประเมินความสามารถประสิทธิภาพของระบบ

3.2 ขั้นตอนและวิธีการดำเนินงาน

ระบบที่ออกแบบ ถูกแบ่งตามการทำงานออกเป็น 5 ส่วน ได้มีการออกแบบระบบโดยรวมให้ครอบคลุมการใช้งานได้หลากหลาย

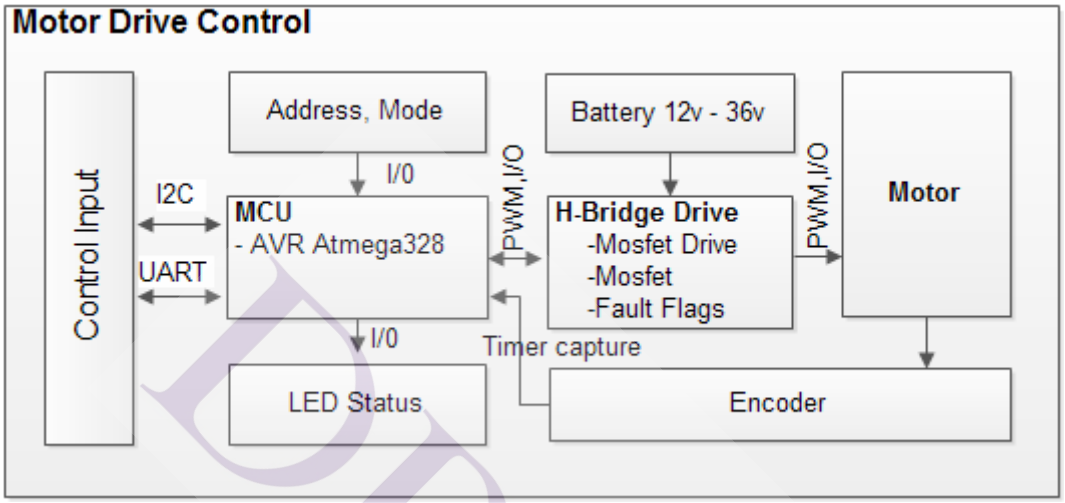


ภาพที่ 3.1 การออกแบบหุ่นยนต์ Mobile Robot

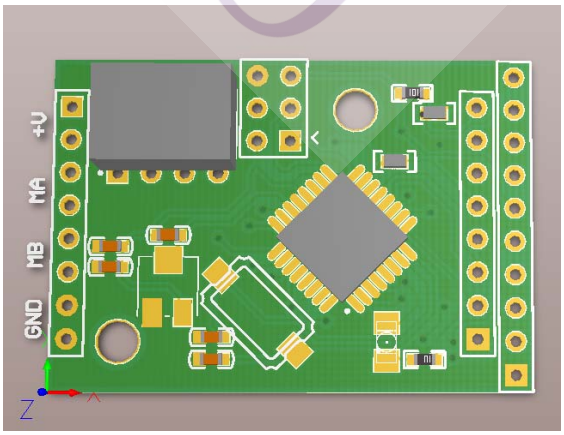
จากการศึกษาการทำงานของหุ่นยนต์ รูปแบบของหุ่นยนต์ Mobile Robot และศึกษาหาความเหมาะสมโดยรวมของหุ่นยนต์ โดยหุ่นยนต์ Mobile Robot จะประกอบด้วย 3 ส่วน ซึ่งได้แก่ ระบบการเคลื่อนที่ ระบบการรับรู้ และระบบการสั่งงานวางแผนและตัดสินใจ ในงานวิจัยนี้มุ่งเน้นการออกแบบวงจรที่ใช้กับ Mobile Robot ที่สามารถประยุกต์การใช้งานได้หลากหลาย

3.3 วงจร Feedback Controller

การรองรับ เซนเซอร์ วงจรในการขับเคลื่อนที่สามารถเพิ่มลดจำนวนมอเตอร์ได้ ซึ่งส่วนขับเคลื่อนถือว่าเป็นส่วนสำคัญที่สุดเราจึงให้ความสำคัญในการออกแบบ จากรูปที่ 3.2 เป็นการออกแบบ Motor Drive Control โดยจะแบ่งออกเป็น 2 ส่วนคือ ส่วน วงจร Feedback Controller กับวงจรขับมอเตอร์ H-Bridge



ภาพที่ 3.2 การออกแบบ Motor drive control



ภาพที่ 3.3 วงจรหน่วยประมวลผลบน Motor drive control

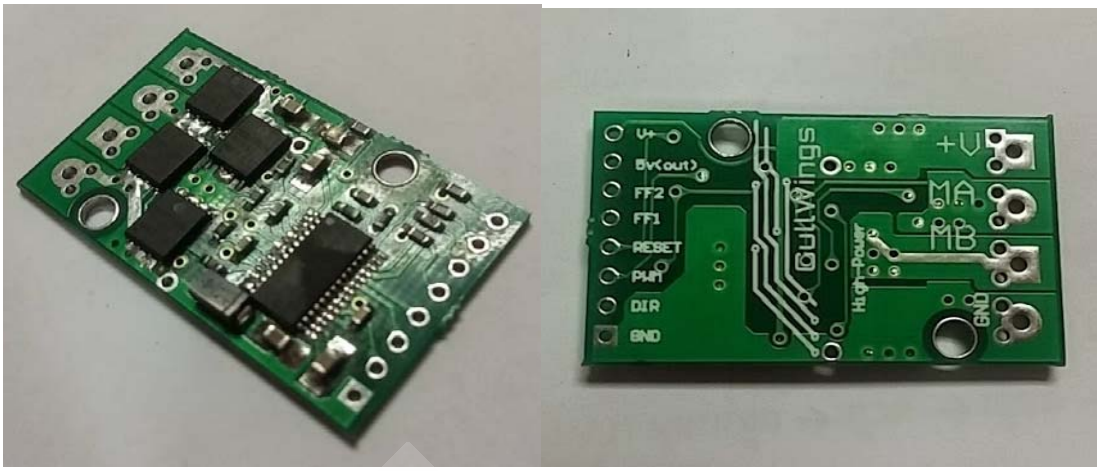
เป็นวงจรหน่วยประมวลผลที่ติดตั้งอยู่บน Motor drive control เพื่อรับคำสั่งจาก Main controller มาสั่งงาน วงจร H-Bridge เพื่อควบคุมความเร็วมอเตอร์ ภายในวงจะประกอบ MCU Atmega328 ซึ่งรับคำสั่งผ่านการเชื่อมต่อ UART หรือ I2C คำสั่งที่สั่งงานจะอยู่ในรูปแบบ Protocol ภายในตัว MCU จะมีการควบคุมความเร็วของมอเตอร์ด้วยระบบควบคุม PID ด้วยระบบป้อนกลับ (feedback control) โดยใช้ค่าที่ได้จาก Encoder ของมอเตอร์มาคำนวณความเร็วและระยะทางของมอเตอร์

ตารางที่ 3.1 รูปแบบ Protocol ที่ใช้สั่งงาน Motor drive control

Byte	**name**	**Value**	**รายละเอียด**
1	Start1	0xFF	Start Packet 1
2	Start2	0xFF	Start Packet 2
3	length	0x07	ความยาวของข้อมูล
4	Id	0x01 – 0x14,0xFE	ID ของ Motor drive control โดยที่ 0xFE จะเป็นการสั่งงานพร้อมกันทั้งหมด
5	Read/Write	0x01,0x02	อ่านหรือเขียน
6	Mode	0x01-0x04	โหมดคำสั่ง
7	Value	0x00-0xFF	ค่าตัวแปร
8	checksum	0x00-0xFF	ค่าความถูกต้องของ Packet คำนวณจาก ((length+id+ Read/Write + mode+ Value)&0xFF)^0xFF

3.4 วงจรขับมอเตอร์ H-Bridge

วงจรขับมอเตอร์ H-Bridge ประกอบด้วย Mosfet เบอร์ IRF7862 จำนวน 4 ตัวและ IC Drive Mosfet เบอร์ A3941 ตัววงจรสามารถรองรับการทำงานที่ 5.5-30 Volt รองรับกระแสไฟต่อเนื่องที่ 20 Amp และความถี่สูงสุด 40kHz



ภาพที่ 3.4 วงจรขับมอเตอร์ H-Bridge

ตารางที่ 3.2 ความหมายของ I/O บนวงจรขับมอเตอร์ H-Bridge

PIN	สถานเริ่มต้น	รายละเอียด
V+		ไฟเลี้ยงมอเตอร์ขั้วบวก 5.5 – 30 Volt
5V (out)		ไฟออก 5 Volt 50ma
GND		ไฟเลี้ยงมอเตอร์ขั้วลบ
OUTA		Output A ไปยัง Motor
OUTB		Output B ไปยัง Motor
PWMH	LOW	ขาสัญญาณ PWM
PWML	HIGH	ขา Control Option
DIR	LOW	ขากำหนดทิศทางการหมุน
RESET	HIGH	Reset IC Drive Mosfet ในกรณี Fault
FF1	LOW	Fault flag 1
FF2	LOW	Fault flag 2

ตารางที่ 3.3 แสดงการสั่งงานของวงจรขับมอเตอร์ H-Bridge

PWMH	PWML	DIR	OUTA	OUTB	Operation
H	H	L	GND	V+	Forward
H	H	H	V+	GND	Backward
L	H	X	GND	GND	Brake Low
H	L	X	V+	V+	Brake High
L	L	X	Z	Z	Coast

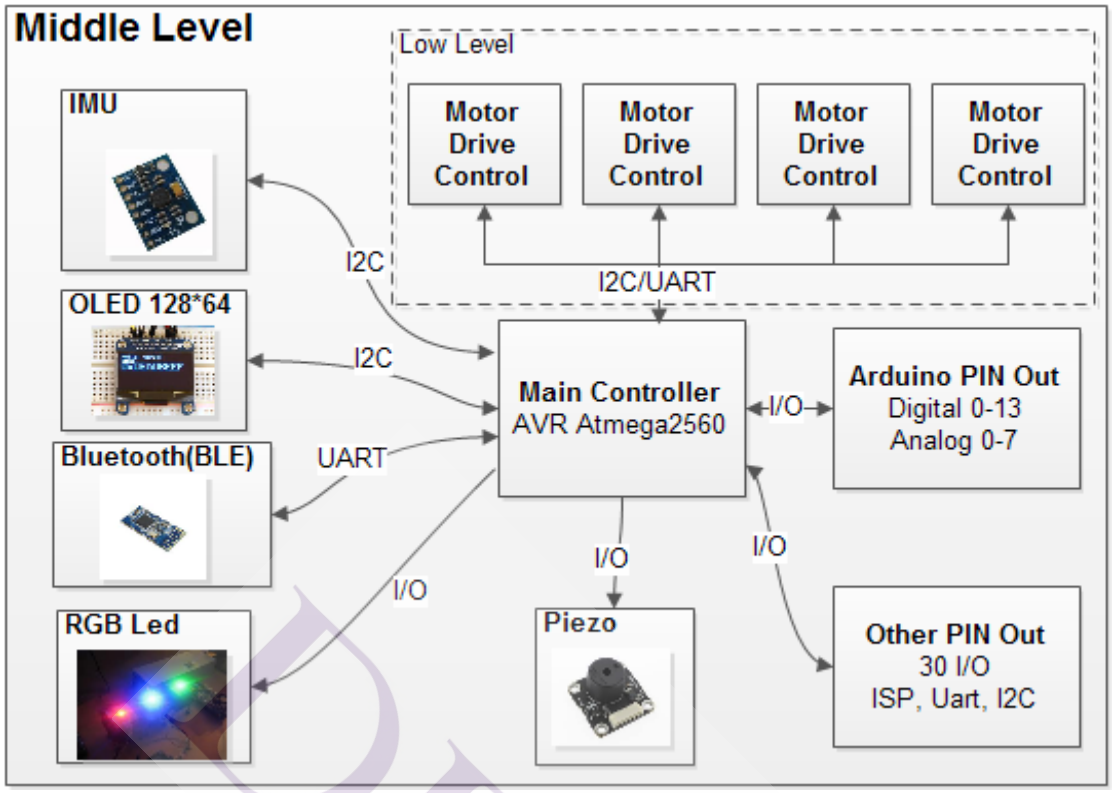
X = ไม่สนใจว่าจะเป็น L หรือ H; Z = high impedance (ไม่มีการเชื่อมต่อ)

ตารางที่ 3.4 ความหมาย Error ของวงจรขับมอเตอร์ H-Bridge

Flag State		รายละเอียด	Disable Outputs	Latched Until Reset
FF1	FF2			
L	L	ปกติ	No	No
L	H	ลัดวงจร	Yes	Yes
H	L	อุณหภูมิสูง	No	No
H	H	ไฟขาเข้าต่ำ	Yes	No

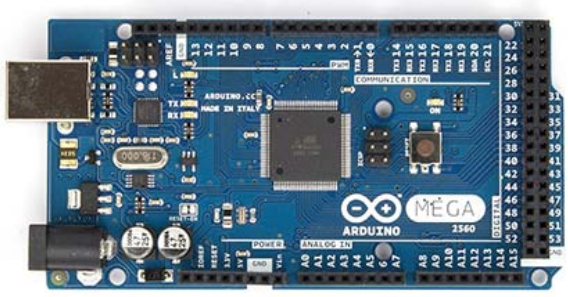
3.5 ออกแบบพัฒนาส่วน Middle Level

ส่วน Middle Level ที่ใช้ในการรับคำสั่งและสั่งงานการเคลื่อนที่โดยรวม การเชื่อมต่ออุปกรณ์หลายอย่างเข้าด้วยกัน ส่วนการสั่งงานหุ่นยนต์จะสั่งงานผ่าน COM Port ในส่วนของ Middle Level สามารถที่จะเขียนโปรแกรมบน Platform Arduino ได้ โดยไม่จำเป็นจะต้องต่อกับ Mini PC เพื่อรับคำสั่งการเคลื่อนที่ สามารถเรียกใช้โปรแกรมควบคุมการเคลื่อนที่จากโปรแกรมที่เขียนได้โดยตรง

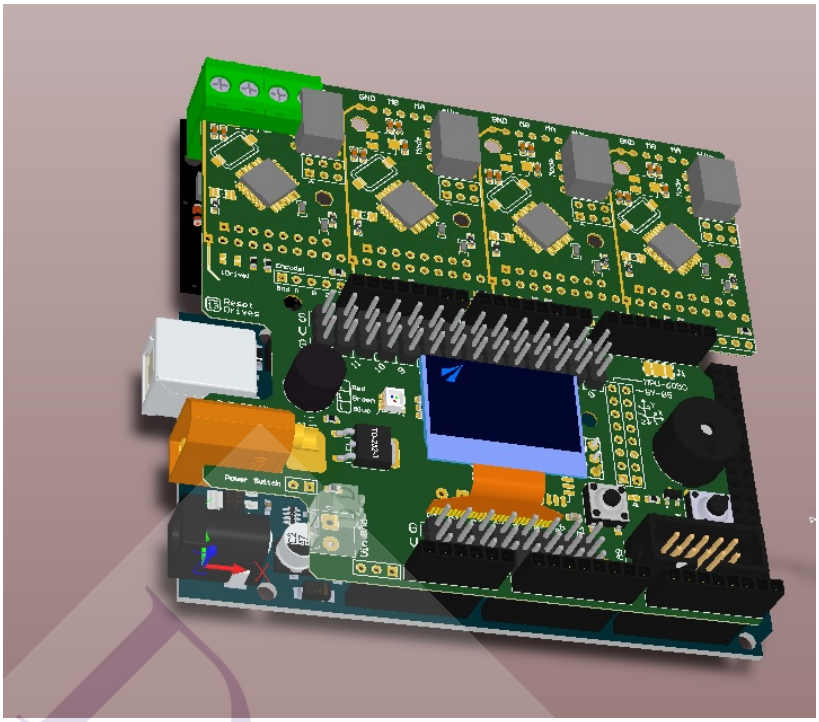


ภาพที่ 3.5 ภาพรวมการออกแบบ Middle Level

ส่วนของหน่วยประมวลผลในส่วน Middle Level จะใช้ IC Atmega2560 เพราะมีการเชื่อมต่ออุปกรณ์ที่หลากหลาย และมี I/O ให้ใช้มากกว่า 50 pin การออกแบบในส่วนของ Main controller เพื่อลดความยุ่งยากในการออกแบบจะใช้บอร์ด Arduino 2560 มาเป็น Main controller แล้วทำการออกแบบวงจรในลักษณะ Shield ใช้บอร์ด Arduino 2560 มาประกอบด้วยกันทำให้สามารถเขียนโปรแกรมโดยใช้ Arduino IDE ซึ่งเป็นการเขียนโปรแกรม Microcontroller ที่เข้าใจง่าย ความซับซ้อนน้อย



ภาพที่ 3.6 บอร์ด ArduinoMega2560



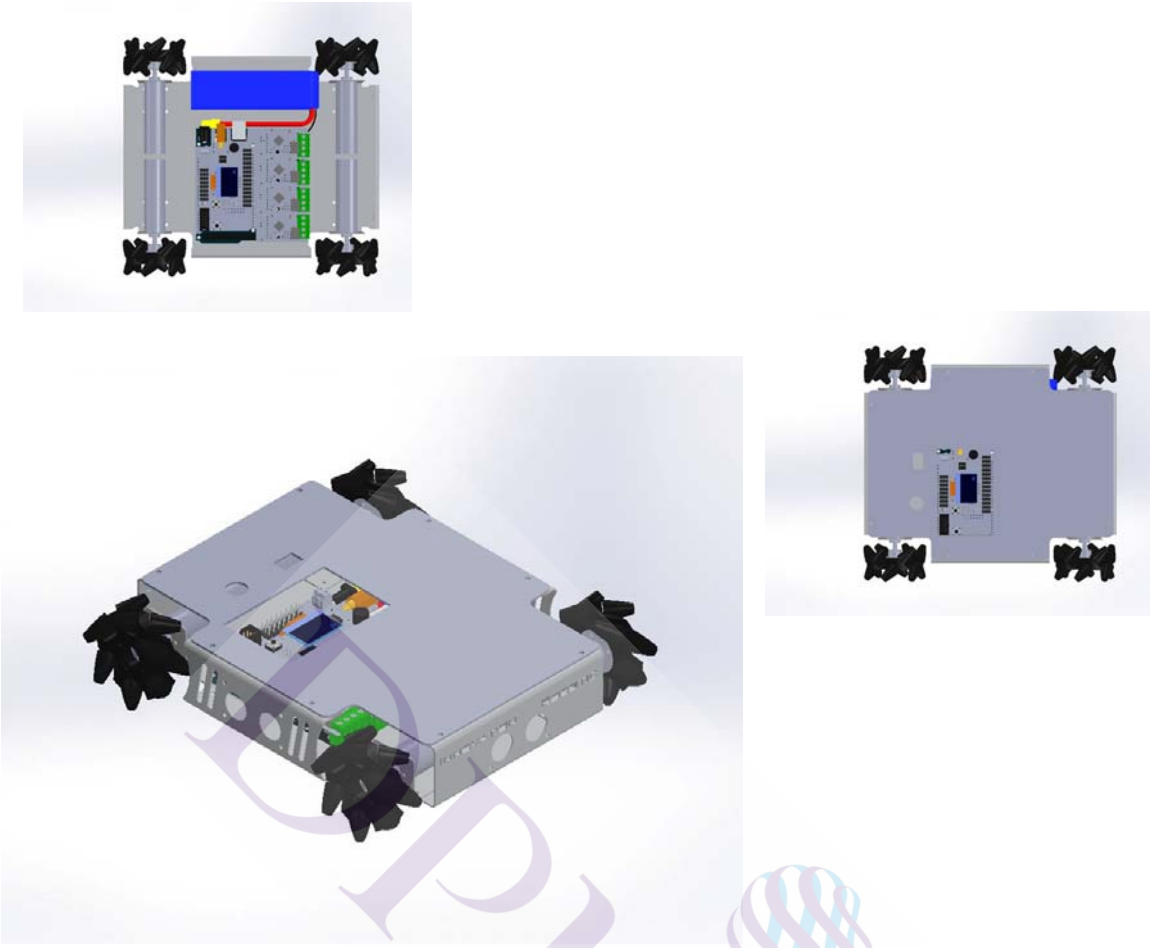
ภาพที่ 3.7 บอร์ด Low level

ในส่วนของบอร์ด Middle Level ที่ได้ออกแบบขึ้นประกอบไปด้วย Motor drive control จำนวน 4 ตัว, IMU, OLED, Bluetooth หรือ Wi-Fi โมดูล, RGB Led, Piezo ซึ่งมีหน้าที่ดังต่อไปนี้

1. Motor drive control หน่วยประมวลผลย่อยในการควบคุม วงจร H-Bridge
2. IMU เซนเซอร์ เป็น เซนเซอร์ ตรวจรู้ ความเร็วเชิงมุม และความเร่ง
3. OLED จอแสดงผล ขนาด 128*64 pixel
4. Bluetooth ,Wi-Fi โมดูลอุปกรณ์ติดต่อสื่อสาร
5. RGB Led ไฟแสดงสถานะ
6. Piezo ลำโพง

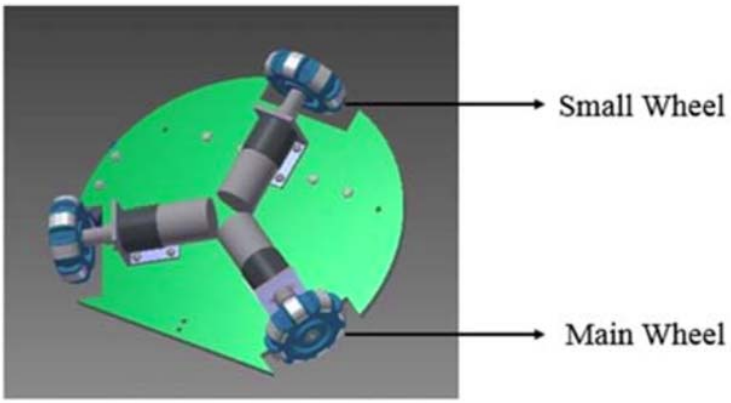
3.6 ออกแบบโครงสร้าง มอเตอร์ และ ล้อ

3.6.1 โครงสร้าง



ภาพที่ 3.8 รูปโครงสร้างของหุ่นยนต์ Mecanum ที่ออกแบบขึ้น

หุ่นยนต์ที่ออกแบบเพื่อใช้ทดสอบวงจรตัวที่ 1 จะใช้การเคลื่อนที่ แบบ Mecanum Directional เป็นหุ่นยนต์ที่มามารถเคลื่อนที่ได้ทุกทิศทาง โดยไม่ต้องหมุนตัวโดยโครงสร้างทำจากอลูมิเนียม ความหนา 2 มิลลิเมตร ตัดโดย laser cut และขึ้นรูปโดยการพับ ลักษณะของหุ่นยนต์ที่ออกแบบมี 4 ล้อ เพื่อให้สามารถทดสอบการทำงานของวงจรได้เต็มประสิทธิภาพ ขนาดหุ่นยนต์ 25 *22 เซนติเมตร สูง 10 เซนติเมตร



ภาพที่ 3.9 รูปโครงสร้างของหุ่น Omni ที่ออกแบบขึ้น

หุ่นยนต์ที่ออกแบบตัวที่ 2 จะใช้การเคลื่อนที่แบบ Omni Directional แบบ 3 ล้อ เป็นอีกรูปแบบที่สามารถเคลื่อนที่ได้ทุกทิศทางโดยไม่จำเป็นต้องหมุนตัว ล้อออมนิ (Omni Directional Wheel) จะถูกยึดกับตัวมอเตอร์วางในทิศ 120 องศาระหว่างกัน ล้อออมนิแต่ละล้อจะมี Main wheel ซึ่งหมุนรอบแกนขับปกติ และ Small wheel ทำมุม 90 องศา กับ Main wheel 25 *25 เซนติเมตร สูง 30 เซนติเมตร

3.6.2 มอเตอร์

ภาพที่ 3.10 มอเตอร์ที่ใช้ในระบบขับเคลื่อน

Namiki motor

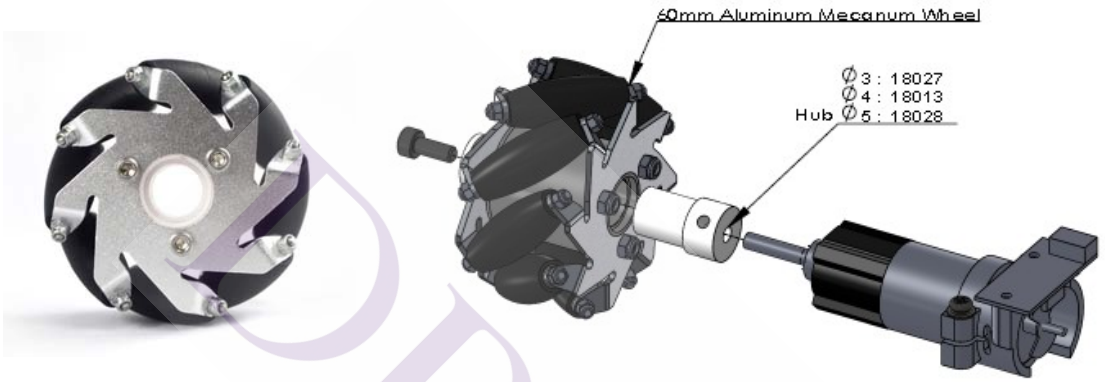
Namiki DC Motor 22CL-3501PG

12VDC 120RPM 1.8A

Shaft Diameter : 4mm Body Diameter : 22mm

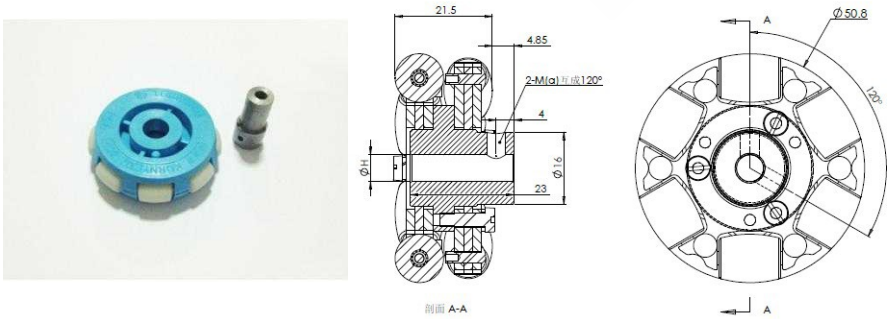
เป็นมอเตอร์มือสอง มีอัตตราทด 80:1 ทำให้มีกำลังสูง มี Encoder แบบ AB ซึ่งมาพร้อมกับตัวมอเตอร์ สามารถรู้ทิศทางและระยะในการหมุนของมอเตอร์

3.6.3 ล้อ



ภาพที่ 3.11 ล้อที่ใช้ในหุ่นยนต์ Mecanum

ล้อ Mecanum ขนาด 60 มิลลิเมตร ของบริษัท Nexus Robot ที่ใช้กับ ชุดโครงสร้างหุ่นยนต์ Mecanum

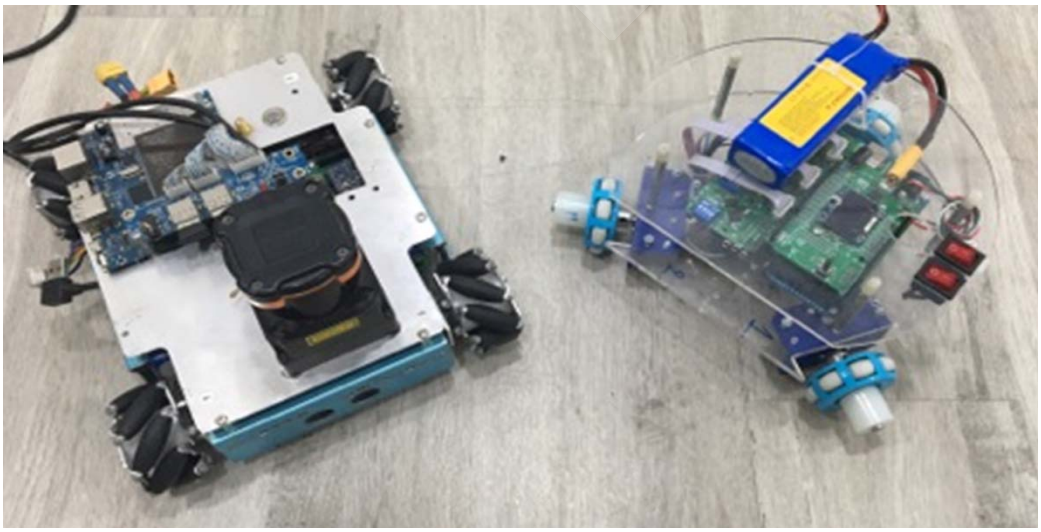


ภาพที่ 3.12 ล้อที่ใช้ในหุ่นยนต์ Omni

3.7 ระบบควบคุมการสั่งการระดับกลาง (Middle ware)

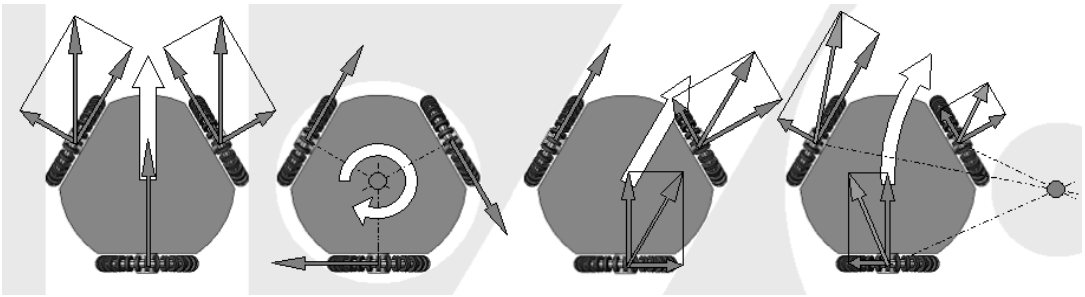
การออกแบบระบบควบคุมการสั่งการระดับกลาง การสร้างโครงสร้างของการเขียนโปรแกรม โดยมีการสร้างรูปแบบมาตรฐานในการควบคุมหุ่นยนต์ ให้สามารรองรับการสั่งงานด้วยรูปแบบเดียวกันไม่ว่าจะนำไปใช้กับหุ่นยนต์รูปแบบไหน สิ่งจำเป็นพื้นฐานในการเคลื่อนที่ของหุ่นยนต์ Mobile Robot คือ จำนวนล้อและรูปแบบการเคลื่อนที่ของหุ่นยนต์ประเภทต่างๆ เช่น หุ่นยนต์ 2 ล้อ จะใช้มอเตอร์ จำนวน 2 ตัว การเคลื่อนที่ไปข้างหน้าของหุ่นยนต์มอเตอร์ทั้งสองตัวจะต้องหมุนไปข้างหน้าด้วยความเร็วที่เท่ากัน ส่วนหุ่นยนต์ Omni ที่ใช้ มอเตอร์ จำนวน 3 ตัว การที่จะเคลื่อนที่ไปข้างหน้านั้นมอเตอร์ทั้ง 3 ตัวจะหมุนไปด้านหน้าเพียง 2 ตัวเท่านั้น เพื่อที่จะสามารถรองรับรูปแบบการใช้งานหุ่นยนต์ที่หลากหลาย โดยเราจะทำให้การสั่งงาน Motor drive control หรือมอเตอร์แต่ละตัวอยู่ในรูปของ object ที่มีความสามารถในกำหนดความเร็ว ตำแหน่ง ระยะทาง ทำให้ง่ายต่อการจัดการ จากตัวอย่างการสั่งงานให้หุ่นยนต์เคลื่อนที่ไปข้างหน้า ส่วนของระบบควบคุมการสั่งการระดับกลาง จะทำการจัดการรูปแบบการเคลื่อนที่ตามประเภทหุ่นยนต์ที่ระบุไว้ และสั่งงานไปยัง Motor drive control ที่กำหนดไว้ ทำให้การที่จะนำไปประยุกต์ใช้ในรูปแบบต่างๆ ได้ง่ายขึ้น

ในงานวิจัยนี้เพื่อทดสอบการออกแบบระบบโดยจะใช้หุ่นยนต์ Mobile robot 2 รูปแบบการเคลื่อนที่คือ หุ่นที่ใช้ล้อ Mecanum Drive ใช้มอเตอร์จำนวน 4 ตัว กับหุ่นยนต์ Omni Directional Drive แบบใช้มอเตอร์ 3 ตัว หุ่นยนต์ทั้ง 2 รูปแบบ จะรับคำสั่งจากโปรแกรมส่วนของ High Level โดยส่งทิศทางและความเร็วมาให้กับระบบควบคุมการสั่งการระดับกลาง ระบบควบคุมการสั่งการระดับกลาง จะทำการคำนวณ Jacobian matrix ของหุ่นยนต์แต่ละประเภท เพื่อหาความเร็วมอเตอร์แต่ละล้อและทำการสั่งงานส่วนของ Motor Drive Control ต่อไป



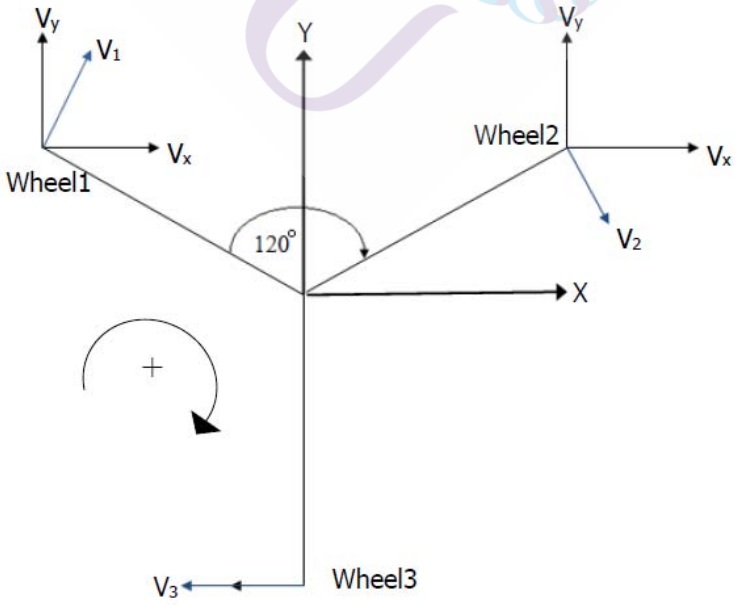
ภาพที่ 3.13 หุ่นยนต์ Mecanum Drive และ Omni Directional ที่พัฒนาขึ้น

3.7.1 การเคลื่อนที่แบบ Omni Directional สามล้อ



ภาพที่ 3.14 ลักษณะการเคลื่อนที่ในทิศทางต่างๆ ของ Omni Wheel

ในการควบคุมและสั่งการให้เคลื่อนที่จะมีหลักการต่างจากการเคลื่อนที่แบบสองล้อทั่วไป ซึ่งในการที่หุ่นยนต์จะเคลื่อนที่ไปในทิศทางใดๆ จะต้องอาศัยความสัมพันธ์ของตำแหน่งและทิศทางการวางตำแหน่งล้อ ทำให้ได้ทิศทางเวกเตอร์ของแต่ละล้อ เมื่อเรารู้ว่าหุ่นยนต์จะวิ่งไปทางใดด้วยความเร็วเท่าไร จะสามารถกำหนดได้ว่า เราต้องสั่งให้แต่ละล้อวิ่งทิศทางใดด้วยความเร็วเท่าไร ซึ่งวิธีแบบนี้เรียกว่า Inverse Kinematic ดังแสดงในภาพที่ 3.14



ภาพที่ 3.15 Inverse Kinematic Free Body Diagram

รูปแบบของ Omni Directional สามล้อซึ่งตำแหน่งของแต่ละล้อจะห่างกัน 120 องศา โดยที่ Main Wheel และ Small Wheel ตั้งฉากกัน 90 องศา ทำให้ได้ตำแหน่งเวกเตอร์ของแต่ละล้อ คือ V_1, V_2 และ V_3 ตั้งฉากกับแกน Main Wheel แกน X แทนความเร็วในแนวแกน X และแกน Y แทนความเร็วในแนวแกน Y กำหนดให้หุ่นยนต์หมุนตามเข็มมีทิศเป็น บวก (+) จุดศูนย์กลางหุ่นยนต์เป็นระยะ R_0

จากความสัมพันธ์ดังกล่าวทำให้ได้สมการ Inverse Kinematic ดังแสดงในสมการที่ 3.1-3.3

$$V_1 = V_y \sin 60 + V_x \cos 60 + \omega_{R0} R_0 \quad (3.1)$$

$$V_2 = -V_y \sin 60 + V_x \cos 60 + \omega_{R0} R_0 \quad (3.2)$$

$$V_3 = -V_x + \omega_{R0} R_0 \quad (3.3)$$

โดยที่

V_1 คือ ความเร็วเชิงเส้นของล้อ 1

V_2 คือ ความเร็วเชิงเส้นของล้อ 2

V_3 คือ ความเร็วเชิงเส้นของล้อ 3

V_x คือ ความเร็วเชิงเส้นในแนวแกน X

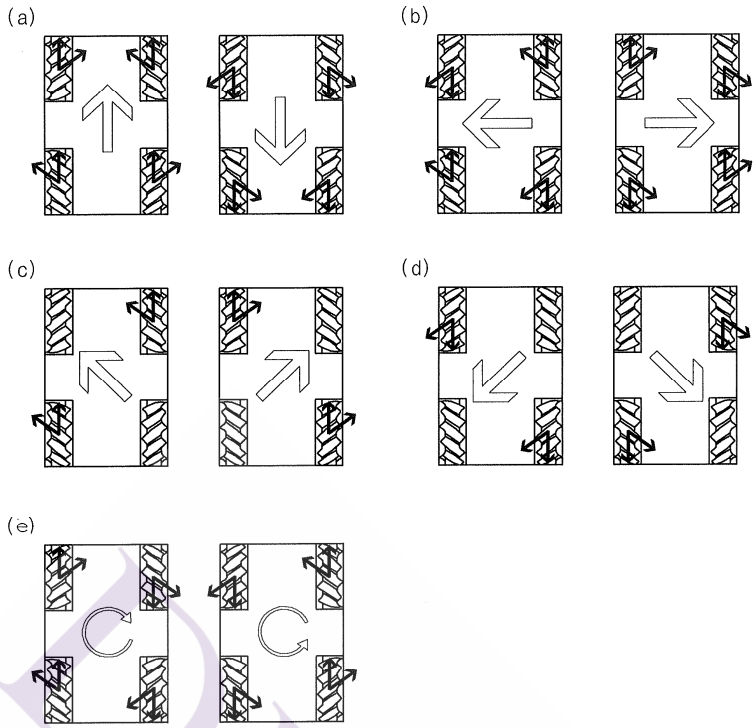
V_y คือ ความเร็วเชิงเส้นในแนวแกน Y

ω_{R0} คือ ความเร็วเชิงมุมในการเคลื่อนที่หมุนรอบตัวเอง

R_0 คือ รัศมีจากจุดศูนย์ของหุ่นยนต์

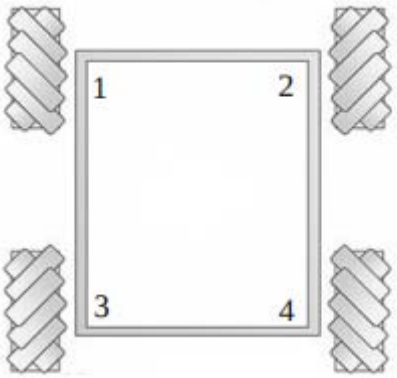
จากสมการการเคลื่อนที่ดังกล่าว เป็นการเคลื่อนที่แบบให้หุ่นยนต์ขับเคลื่อนและหมุนตัวไปด้วย (Translate and Rotate) โดยในสมการ $\omega_{R0} R_0$ เป็นเทอมที่กำหนดความเร็วเชิงมุมในการหมุนของหุ่นยนต์แต่ละตัว

3.7.2 การเคลื่อนที่แบบ Mecanum Directional



ภาพที่ 3.16 ลักษณะการเคลื่อนที่ในทิศทางต่างๆ ของ Mecanum Wheel

ในการควบคุมและสั่งการให้เคลื่อนที่จะมีหลักการคล้าย Omni Directional ซึ่งในการที่หุ่นยนต์จะเคลื่อนที่ไปในทิศทางใดๆ จะต้องอาศัยความสัมพันธ์ของตำแหน่งและทิศทางการวางตำแหน่งล้อ ทำให้ได้ทิศทางเวกเตอร์ของแต่ละล้อ



ภาพที่ 3.17 Mecanum Wheel

$$V1 = Vd \sin(\theta d + \pi/4) + V\theta \quad (3.4)$$

$$V2 = Vd \cos(\theta d + \pi/4) - V\theta \quad (3.5)$$

$$V3 = Vd \cos(\theta d + \pi/4) + V\theta \quad (3.6)$$

$$V4 = Vd \sin(\theta d + \pi/4) - V\theta \quad (3.7)$$

โดยที่

V_1 คือ ความเร็วเชิงเส้นของล้อ 1

V_2 คือ ความเร็วเชิงเส้นของล้อ 2

V_3 คือ ความเร็วเชิงเส้นของล้อ 3

V_4 คือ ความเร็วเชิงเส้นของล้อ 4

V_d คือ ความเร็วของหุ่นยนต์ [-1, 1]

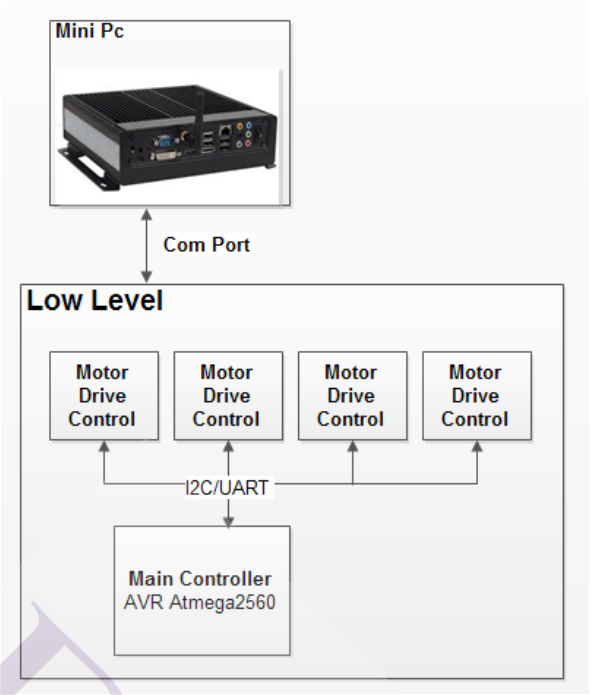
θ_d คือ ทิศทางของหุ่นยนต์ [0, 2 π]

V_θ คือ ความเร็วในการหมุนตัว [-1, 1]

รูปแบบของ Mecanum Directional จะวางในรูปแบบสี่เหลี่ยมจัตุรัส โดยให้แกนหมุนของลูกล้อแต่ละจุดไปยังศูนย์กลางของตัวหุ่นยนต์

3.8 ออกแบบพัฒนา libraries และคำสั่งในการสั่งงาน

ในส่วนของ Libraries เพื่อให้ง่ายต่อการเรียกใช้งานจะแบ่งรูปแบบการทำงานเป็น 2 ส่วน คือ รับคำสั่งจากส่วน High Level และการทำงานเมื่อไม่มี High Level โดย Libraries จะสามารถติดตั้งกับ Arduino IDE ในส่วนของ Core และ libraries ซึ่งจะมี Function ที่เรียกใช้ Motor Drive Control และ รูปแบบ การเคลื่อนที่เข้าไว้ด้วยกัน



ภาพที่ 3.18 ภาพรวมการเชื่อมต่อ ระหว่างส่วน Low Level กับ High Level

คำสั่ง Libraries การเรียกใช้งาน Motor Drive Control ที่ถูกเชื่อมต่อกับ Main Controller โดยการเชื่อมต่อ แบบ serial ด้วยความเร็ว 115200 ใน Motor Drive Control จะมี โปรแกรมที่รอรับคำสั่งจาก Main Controller ซึ่งจะมองตัว Motor Drive Control เป็น Object ที่สามารถสั่งงานผ่าน Libraries ที่สร้างขึ้น โดยสามารถกำหนดให้ Motor Drive Control แต่ละตัวจะมี ID เป็นของตัวเอง ในการสั่งงานจะต้องกำหนด ID ของ Motor Drive Control ไปด้วยทุกครั้ง

ตัวอย่างการเรียกใช้งาน Motor Drive Control

```
#include "iDrive.h"
iDrive_results allDriveResults;
iDrive allDrive(MODE_UART);

allDrive.InitDrive(&allDriveResults);
```

Function ของ Motor Drive Control

```
class iDrive
{
          public:
          iDrive(uint8_t mode);
          uint8_t InitDrive(iDrive_results *drive);
          int32_t Get_Encoder(uint8_t ID);
          uint16_t Get_Speed(uint8_t ID);
          uint16_t Get_MaxSpeed(uint8_t ID);
          uint8_t * GET_Buffer();
          uint8_t SET_SPEED(uint8_t ID,uint8_t DIRECTION,uint16_t SPEED,uint8_t
waitRes);
    uint8_t SET_DVI_TIME(iDrive_results *drive);
          uint8_t SET_PID(iDrive_results *drive);
          private:
          uint8_t Wait_Res();
          uint8_t TxPacket(uint8_t bID, uint8_t ins, uint8_t *Parameter,
uint8_t bParameterLength);
          void Send_Com(uint8_t *data,uint8_t length);
          uint8_t RxPacket(uint8_t bRxPacketLength);
          uint8_t checkSum(uint8_t start,uint8_t stop);
};
```

Function ของ Middle Level

```
class iDrive
{
          public:
          MobileRobot();
          void SetDrive(uint8_t ID,float P,float I,float D,uint8_t TIME,uint8_t
ENCODE_DIV);
          void SetAllDrive(float P,float I,float D,uint8_t TIME,uint8_t
ENCODE_DIV);
          //void SetSpeedSetSpeed(uint8_t ID,uint8_t DIRECTION,uint8_t SPEED);
          uint8_t SetSpeed(uint8_t ID,uint8_t DIRECTION,uint8_t SPEED);
          uint16_t TestSpeedMotor(uint8_t ID);
          int32_t ReadEncode(uint8_t ID);
          uint16_t SpeedMaxMotor(uint8_t ID);
          uint16_t ReadCurrentSpeed(uint8_t ID);

          void ReadAllCurrentSpeed( MotorData *motorValue);
          void SynchronousSpeedMotor(MotorData *motorValue,double gaint);
          void SetAllSpeed(MotorData *motorValue,uint8_t directionMode);
          void ReadAllEncode(MotorData *motorValue);

          void MecanumVector(MotorData *motorValue,double speed ,double angle,
double rotation);
          void MecanumRotation(MotorData *motorValue,double speed , double
rotation);
          void OmniVector( MotorData *motorValue,int cTheta, int direction_X ,
int direction_Y ,int rotate,int speed /* 1*/);
```

3.9 ออกแบบพัฒนาระบบระบุตำแหน่ง

การออกแบบส่วนที่ใช้ในการระบุตำแหน่งของหุ่นยนต์ การที่จะทำให้หุ่นยนต์สามารถระบุตำแหน่งของตัวเองภายในห้องได้มีหลายวิธี เช่น การเคลื่อนที่ตามเส้น การใช้เซนเซอร์ แบบ Absolute position อย่างเช่น Stargazer, GPS หรือการสร้างแผนที่ Robot Mapping งานวิจัยนี้สร้างระบบที่ทำให้หุ่นยนต์สามารถระบุตำแหน่งของตัวเองภายในห้อง โดยใช้ Monte Carlo algorithm ซึ่งเป็น Randomized algorithm เพื่อระบุตำแหน่งของหุ่นยนต์จากแผนที่ที่มีอยู่แล้ว หรือแผนที่ที่จัดทำขึ้น

3.9.1 Lidar Laser เซนเซอร์[1]

Lidar Laser เซนเซอร์ ที่ใช้ จะเป็นเซนเซอร์ ประเภท distance scanner แบบ 360 องศา หลักการทำงาน คือ ตัวเซนเซอร์ จะมีส่วนที่หมุนอยู่ตลอดเวลา ในสวนที่หมุนนั้นจะมีเซนเซอร์ วัดระยะทาง วัดระยะในมุมที่หมุนไป โดยใช้หลักการสะท้อนของแสง Laser ระยะที่สามารถวัดได้คือ 0.2 – 6 เมตร เมื่อหมุนครบหนึ่งรอบ จะให้ output เป็น ระยะระหว่างตัว เซนเซอร์ กับผนังหรือวัตถุที่ laser ถูกสะท้อนมา ในแต่ละองศา ตั้งแต่ 0 - 360 องศา

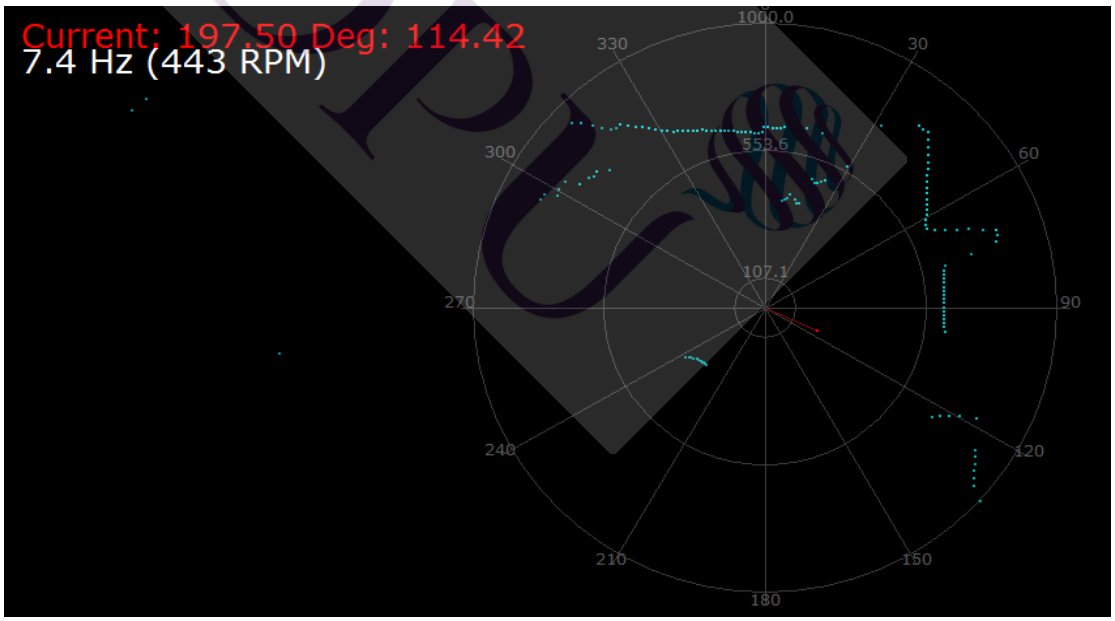
ภาพที่ 3.19 Lidar Laser

[1] RPLIDAR 360° Laser Scanner, robotshop [ออนไลน], สืบค้นเมื่อ 7 เมษายน 2558, จาก http://www.robotshop.com/en/rplidar-360-laser-scanner.html

Item	Unit	Min	Typical	Max	Comments
Distance Range	Meter(m)	TBD	0.2 - 6	TBD	White objects
Angular Range	Degree	n/a	0-360	n/a	
Distance Resolution	mm	n/a	<0.5 <1% of the distance	n/a	<1.5 meters All distance range*
Angular Resolution	Degree	n/a	≤1	n/a	5.5Hz scan rate
Sample Duration	Millisecond(ms)	n/a	0.4	n/a	
Sample Frequency	Hz	n/a	≥2000	2010	
Scan Rate	Hz	1	5.5	10	Typical value is measured when RPLIDAR takes 360 samples per scan

ภาพที่ 3.20 Lidar Laser Specifications

ข้อมูลของ Lidar Laser จะถูกส่งมาประมวลผลที่ Mini PC หน่วยประมวลผลที่อยู่ในตัวของหุ่นยนต์ ด้วยความเร็วในการ Scan ประมาณ 5Hz-6Hz ส่งข้อมูลผ่าน COM Port ความเร็ว 115200bps ข้อมูลจะถูกเข้ากระบวนการของการระบุตำแหน่งของหุ่นยนต์



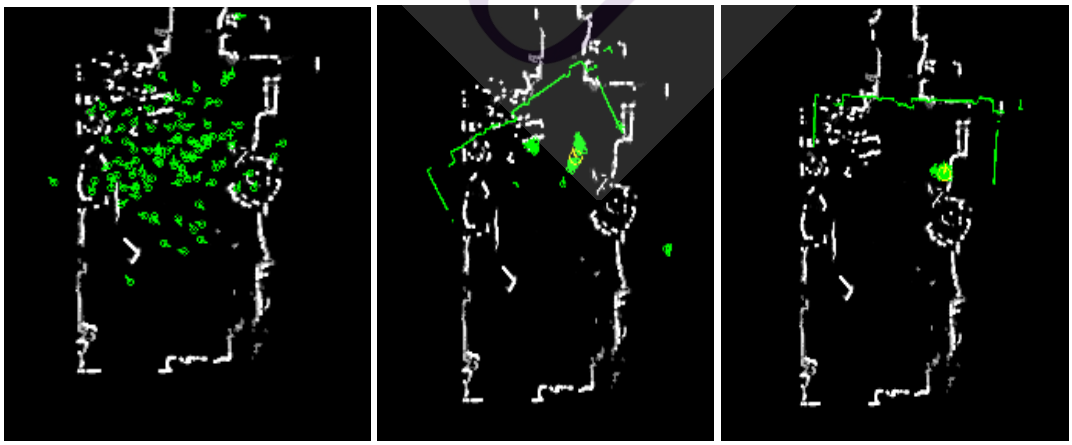
ภาพที่ 3.21 ภาพที่ได้จาก Demo Application

```
 1  #RPLIDAR SCAN DATA
 2  #COUNT=220
 3  #Angule Distance    Quality
 4  359.1563 3275.3 13
 5  0.3906 3273.3 14
 6  1.6719 3246.3 13
 7  2.9063 3274.3 13
 8  4.1719 3246.3 12
 9  5.4219 3275.3 13
10  6.6563 3304.8 12
11  7.9063 3307.5 13
12  9.1563 3307.5 12
13  10.4219 3340.3 12
14  11.6875 3314.5 12
15  12.9375 3318.3 13
16  14.1875 3378.0 13
17  15.4375 3381.5 13
18  16.6406 3384.8 12
19  17.8906 3450.0 13
20  19.1250 3454.8 12
```

ภาพที่ 3.22 ตัวอย่างข้อมูลที่ได้จาก Lidar Laser

ข้อมูลของ Lidar Laser เป็นข้อมูลของระยะระหว่าง Sensor กับสิ่งกีดขวางในแต่ละองศาที่ Sensor หมุน ค่าที่ได้หน่วยเป็นมิลลิเมตรค่าความผิดพลาดอยู่ที่ 1 เปอร์เซ็นต์

3.9.2 Monte Carlo localization

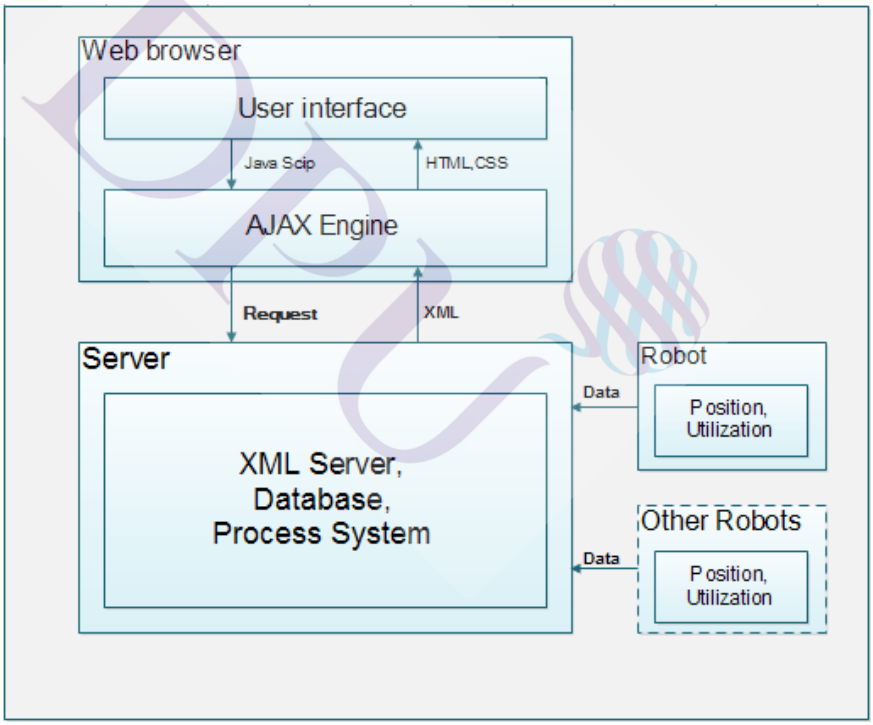


ภาพที่ 3.23 ระบบระบุตำแหน่งที่ได้ออกแบบขึ้นจาก Monte Carlo localization

ขั้นตอนของวิธีการระบุตำแหน่งของ Monte Carlo localization ระบบระบุตำแหน่งที่ทำขึ้นจะทำการสุ่มจุดที่คาดว่าจะเป็นตำแหน่งของหุ่นยนต์ขึ้นมา 100 จุด แต่จะจุดจะทำการหาค่าความน่าจะเป็นโดยนำค่าของเซ็นเซอร์ที่อ่านได้มาเทียบกับแผนที่ ที่ถูกสร้างไว้ตั้งแต่ต้น จุดที่มีการทับของเซ็นเซอร์กับแผนที่มากที่สุดจะมีความน่าจะเป็นที่เป็นตำแหน่งของหุ่นยนต์

3.10 การออกแบบพัฒนา ระบบสารสนเทศ

เมื่อหุ่นยนต์ สามารระบุตำแหน่งของตัวเองได้แล้ว ข้อมูล ภาพแผนที่และตำแหน่งจะถูกส่งไปแสดงผลที่ระบบสารถสนเทศ โดยข้อมูลทั้งสองจะถูกเก็บไว้ยัง Server ที่เขียนโดย Node.js โดยจะทำการส่งข้อมูล จาก Lab Top ไปยัง Local Server ที่จำลองขึ้น ระบบสารสนเทศจะนำข้อมูลที่ถูกบันทึกไว้ในฐานข้อมูลมาแสดงผล ผ่าน Web Browser



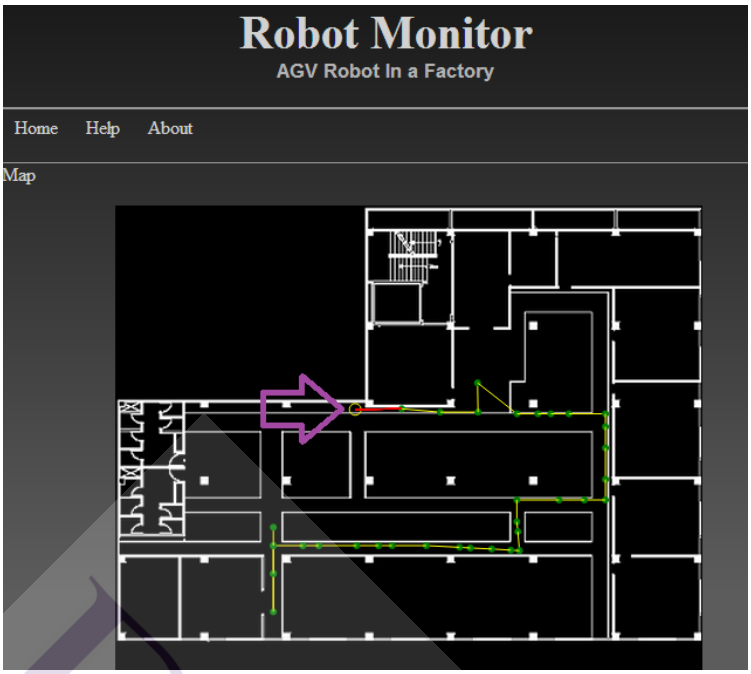
ภาพที่ 3.24 ภาพการติดต่อข้อมูลหุ่นยนต์

จากภาพ ข้อมูลตำแหน่งของหุ่นยนต์จะถูกส่งมายัง Server โดยจะมีข้อมูลตำแหน่งของหุ่นยนต์ และข้อมูลตามตารางที่ 3.5 ซึ่งจะจำลองฐานข้อมูลอยู่บน Notebook โดยจะมีทั้ง Database, Web Server

ตารางที่ 3.5 ข้อมูลที่ส่งไปเก็บไว้ในฐานข้อมูล

Table Name	Attribute Name	Attribute Type	Attribute Size
TblRobot	ID	AutoNumber	Long Integer
	robotID	Text	4
	mapID	Text	30
	PositionX	Text	30
	PositionY	Text	30
	Time	Date/Time	30

ดูข้อมูลตำแหน่งของหุ่นยนต์ สามารถเรียกดูได้จาก Web Browser เมื่อมีการเรียกใช้งาน ข้อมูลที่เก็บอยู่ในฐานข้อมูลจะถูกเลือกขึ้นมา ในรูปแบบ XML ไฟล์ ข้อมูลที่แสดงบนหน้า Web Browser จะถูกอัพเดททุกๆ 1 วินาที โดยเทคนิคการรับส่งข้อมูล AJAX (Asynchronous JavaScript and XML) เป็นการรับส่งข้อมูลแบบ Asynchronous ของ JavaScript และ XML มีข้อดีคือ ผู้ใช้ไม่จำเป็นต้องโหลดข้อมูลใหม่เมื่อมีการ อัพเดท ข้อมูลตำแหน่งของหุ่นยนต์, ลดปริมาณการรับส่งข้อมูล โดยข้อมูลจะเปลี่ยนเฉพาะส่วนที่ต้องการ ทำให้ทำงานได้เร็วขึ้น



ภาพที่ 3.25 ภาพตำแหน่งหุ่นยนต์จากระบบสารสนเทศ

หน้าข้อมูลที่ถูกแสดงผลบน Web Browser ข้อมูลแผนที่จะถูกจัดเก็บในรูปแบบของรูปภาพ ซึ่งจะต้องจัดเตรียมไว้ล่วงหน้า ส่วนตำแหน่งของหุ่นยนต์จะถูกสร้างจากข้อมูลที่ได้รับจากการใช้เทคนิค AJAX

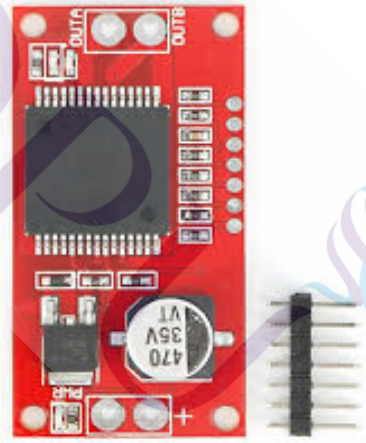
บทที่ 4
ผลการศึกษา

เนื้อหาในบทนี้จะกล่าวถึงผลการศึกษาวิจัยผลการวิจัย ผลการศึกษาวิจัยได้ผลการจำลองระบบดังต่อไปนี้

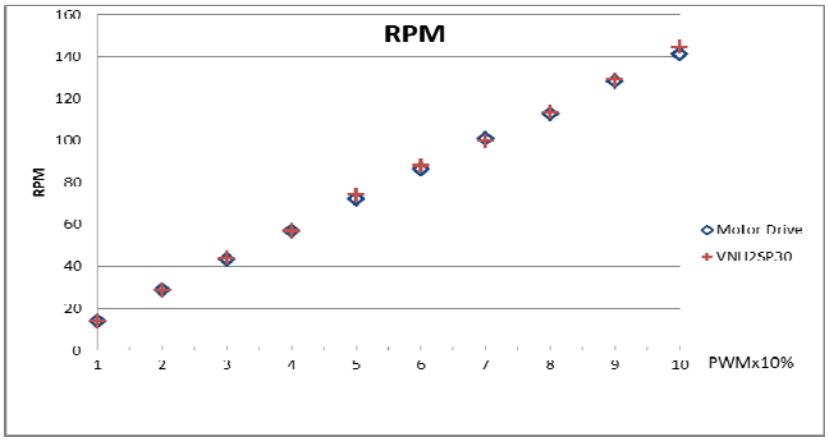
4.1 การทดสอบ Motor Drive Control

4.1.1 การทดสอบวงจร H-Brige

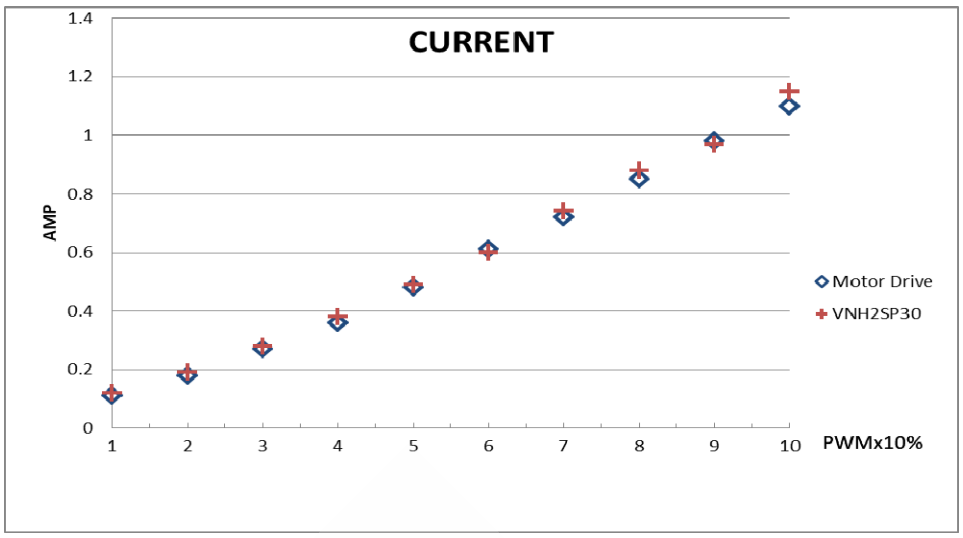
วงจรขับมอเตอร์ที่นำมาทดสอบร่วมกันคือ H-Brige VNH2SP30



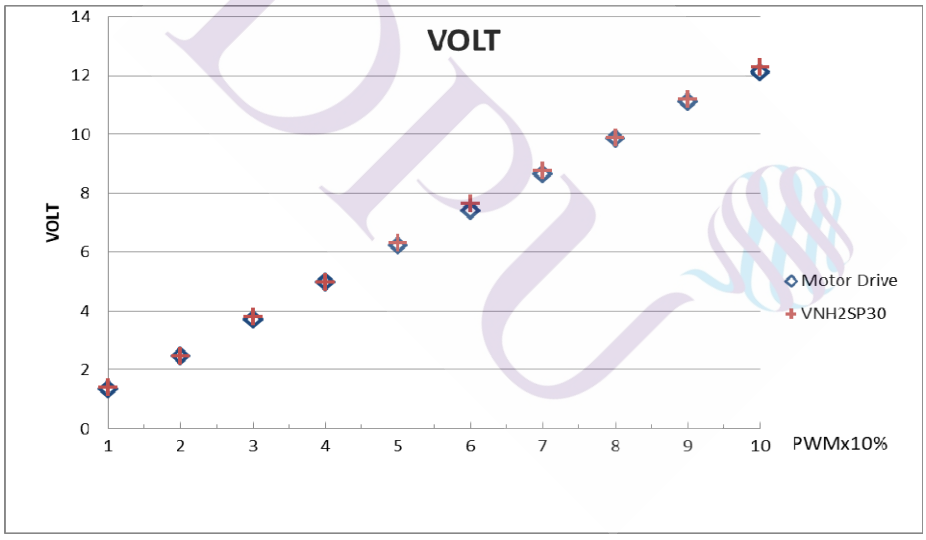
ภาพที่ 4.1 วงจร VNH2SP30



ภาพที่ 4.2 ผลของความเร็วมอเตอร์ ที่ได้จากการสั่งงานวงจรทั้งสอง



ภาพที่ 4.3 ผลของกระแสที่ใช้ เปรียบเทียบจากการสั่งงานวงจรทั้งสอง



ภาพที่ 4.4 ผลของแรงดันไฟฟ้า เปรียบเทียบจากการสั่งงานวงจรทั้งสอง

ทดสอบการทำงานของวงจร โดยทดสอบกับมอเตอร์ Namiki ในสภาวะไม่มีโหลด จากผลตามภาพที่ 4.3 ถึง 4.4 เมื่อสั่งงานมอเตอร์ด้วย PWM (Pulse Width Modulation) จาก 0 เปอร์เซ็นต์ จนถึง 100 เปอร์เซ็นต์ วงจรสามารถจ่ายแรงดันและกระแสให้กับมอเตอร์ได้ ผลใกล้เคียงกับวงจรที่นำมาทดสอบร่วมกัน

4.1.2 การทดสอบความเร็วที่เหมาะสมกับการสั่งงาน Motordrive control

ทดสอบการสั่งงานวงจร Low level ทดสอบโดยส่งคำสั่งให้กับวงจรโดยเมื่อสั่งงานสำเร็จจะมีการตอบกลับคำสั่งมาให้ Main controller เพื่อหาอัตราความเร็วที่วงจรสามารถรองรับการสั่งงานได้

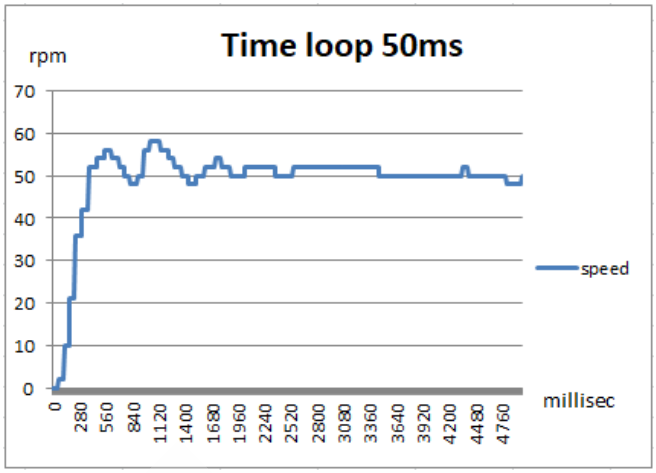
ตารางที่ 4.1 สั่งงาน Motordrive control ที่ ความเร็ว 115200 bps

Motor drive	Loop Time(ms)	Frequency(Hz)
1	8.08	123.7
2	9.47	105.56
3	10.86	92.09
4	12.47	80.19

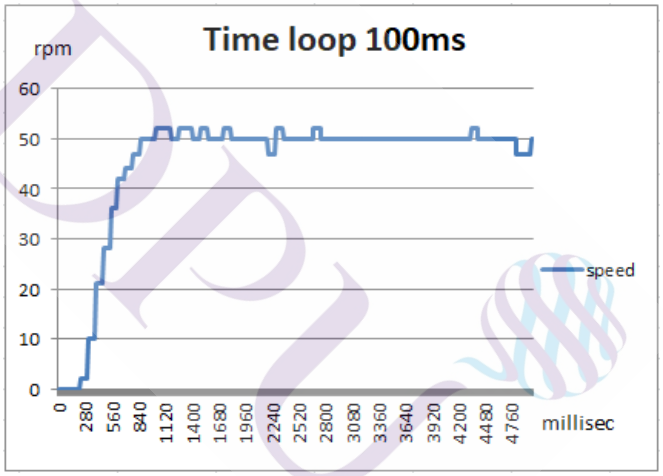
ตารางที่ 4.2 สั่งงาน Motordrive control ที่ ความเร็ว 250000 bps

Motor drive	Loop Time(ms)	Frequency(Hz)
1	6.31	158.47
2	7.21	128.69
3	8.13	123
4	9.52	105

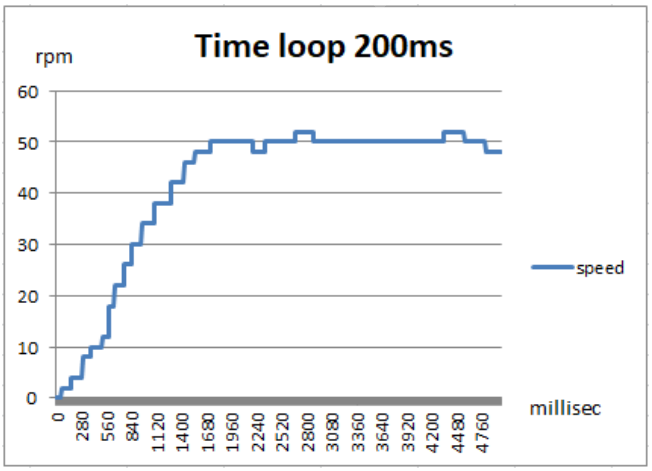
ในการทดสอบทำการส่งข้อมูลจำนวน 1000 ครั้ง อัตราในการส่งข้อมูลสำเร็จของการส่งข้อมูลเป็น 100 เปอร์เซ็นต์ จากทั้ง 2 ตารางวิเคราะห์ได้ว่าเมื่อจำนวน Motor Drive มากขึ้นเวลาที่ใช้ในการสั่งงานก็จะมากขึ้นเช่นกัน ที่ความเร็ว 115200 bps เมื่อจำนวน Motor Drive เพิ่มจาก 1 ตัว เป็น 2 ตัว ใช้เวลาเพิ่มขึ้น 1.39ms และเพิ่มขึ้นเป็น 3 ตัว เวลาเพิ่มขึ้น 1.39ms เช่นกัน จำนวน Motor Drive ที่เพิ่มขึ้นจะทำให้จำนวนครั้งในการสั่งงาน Motor Drive ต่อวินาทีลดลง แต่เมื่อใช้ความเร็วในการส่งข้อมูลมากขึ้นทำให้จำนวนครั้งต่อวินาทีที่สามารถสั่งงานได้มากขึ้น การเลือกความเร็วที่เหมาะสมสามารถกำหนดได้จากความเหมาะสมในการสั่งงานได้โดยการทดสอบหาได้จากเวลาในการคำนวณ PID



ภาพที่ 4.5 ทดสอบเปลี่ยนความเร็วในการคำนวณ PID โดย SET POINT = 50rpm TIME = 50ms



ภาพที่ 4.6 ทดสอบเปลี่ยนความเร็วในการคำนวณ PID โดย SET POINT = 50rpm TIME = 100ms



ภาพที่ 4.7 ทดสอบเปลี่ยนความเร็วในการคำนวณ PID โดย SET POINT = 50rpm TIME = 200ms

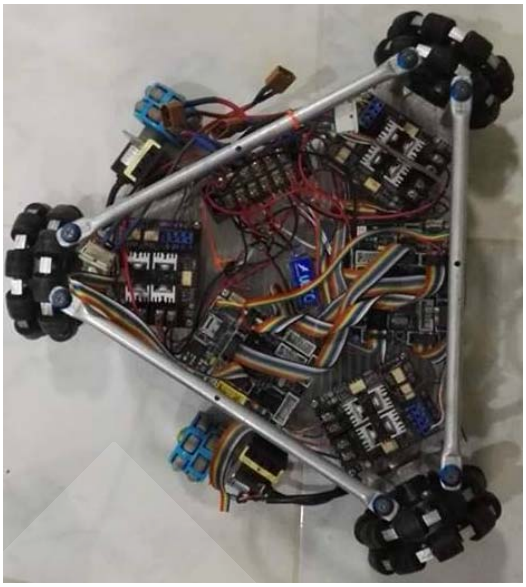
ทดสอบการปรับเปลี่ยนเวลาในการคำนวณ PID ในแต่ละครั้งเพื่อหาความเร็วที่เหมาะสมในการสั่งงานวงจร Low level ภาพที่ 4.5 ถึง 4.7 ทดปรับเปลี่ยนเวลา 50ms 100ms 200ms เท่ากับ ความถี่ 20Hz 10Hz 5Hz ผลการทดสอบช่วงเวลาที่เหมาะสมสำหรับมอเตอร์ Namiki อยู่ในช่วงเวลาประมาณ 10Hz เป็นเวลาที่เหมาะสมถ้าหากเวลาคำนวณ PID เร็วไปมอเตอร์ไม่สามารถตอบสนองกับ OUT PUT คำนวณได้ หากช้าไป OUT PUT ก็จะไปถึง SET POINT ช้าเกินไป จากการทดสอบการสั่งงานวงจร วงจรสามารถรองรับการสั่งงานได้ที่ความเร็ว 115200bps ได้ 80Hz และที่ความเร็ว 250000 ได้ 150Hz ซึ่งความเร็วทั้งสองสามารถรองรับการสั่งงานได้มากกว่า 10Hz

ตารางที่ 4.3 ตารางเปรียบเทียบวงจรที่จัดทำขึ้นกับวงจรในท้องตลาด

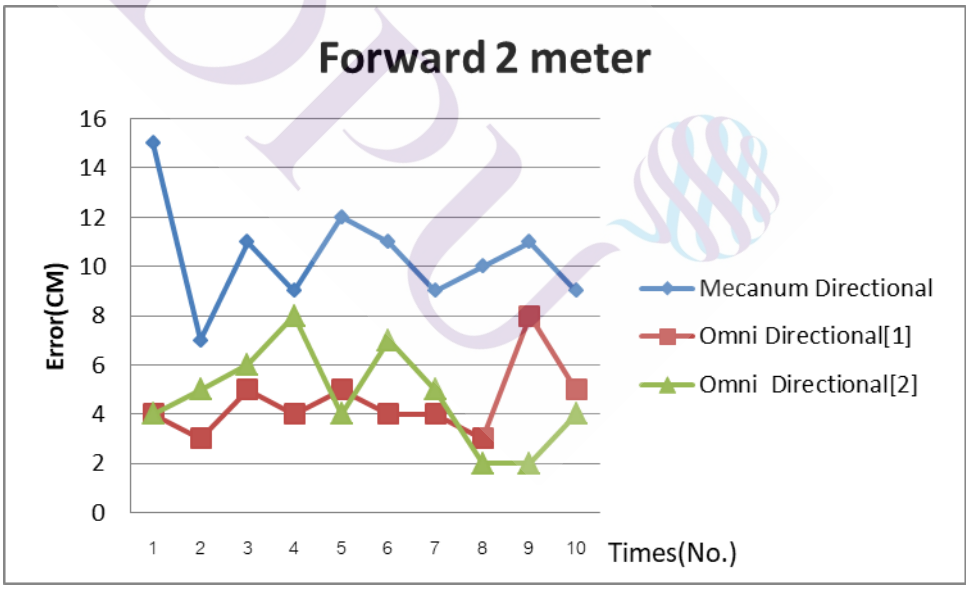
	วงจรที่จัดทำขึ้น	VNH5019 Motor Driver	Cytron SmartDrive Duo-10	DC servo motor drive module PID
กระแสที่รองรับ	10A ,20A	12A	10A 2CH	5A 2Ch
โวลต์ที่รองรับ	12v - 30v	5.5v - 24v	7v - 35v	7v - 12v
PID Control	รองรับ	ไม่รองรับ	ไม่รองรับ	รองรับ
รองรับ Encoder	รองรับ	ไม่รองรับ	ไม่รองรับ	รองรับ
สั่งงานผ่าน Protocol	รองรับ	ไม่รองรับ	รองรับ	รองรับ
ตั้งค่า ID บอร์ด	รองรับ	ไม่รองรับ	ไม่รองรับ	ไม่รองรับ
ราคา	1500	1250	1650	2000

4.2 การสั่งงานการเคลื่อนที่

การสั่งงานหุ่นยนต์ 2 รูปแบบด้วยคำสั่งเดียวกันจาก Hi Level สั่งงานไปยังหุ่นยนต์ทั้ง 2 ตัว โดยทดสอบเพิ่มเติมกับหุ่นยนต์ Omni Directional[2] ที่มีอยู่แล้วเพื่อทดสอบการการเคลื่อนที่กับวงจร Low Level



ภาพที่ 4.8 หุ่นยนต์ Omni Directional[2] ที่นำมาทดสอบเพิ่มเติม



ภาพที่ 4.9 ความคลาดเคลื่อนในการเคลื่อนที่ โดยเคลื่อนเป็นเส้นตรงยะทาง 2 เมตร

จากภาพที่ 4.8 ได้ใช้คำสั่งเดียวกันจาก Hi level สั่งงานให้หุ่นยนต์เคลื่อนที่ไปด้านหน้า 2 เมตร โดยทดสอบและวัดการคลาดเคลื่อนของหุ่นยนต์ทั้งสองรูปแบบจำนวน 10 ครั้ง ในการทดสอบหุ่นยนต์ Omni Directional[1] มีความคลาดเคลื่อน 3-8 เซนติเมตรจากเป้าหมาย และในหุ่นยนต์ Mecanum Directional มีความคลาดเคลื่อน 5-15 เซนติเมตร ความคาดเคลื่อนของหุ่นยนต์

Mecanum Directional มีมากว่าหุ่นยนต์เราพบว่าในการเคลื่อนที่ของหุ่นยนต์ ทั้งสองรูปแบบล้อจะต้องสัมผัสพื้นตลอดเวลาขณะเคลื่อนที่หากล้อใดล้อหนึ่งลอยขึ้นจากพื้นจะมีผลกับการเคลื่อนที่ไปยังเป้าหมาย ซึ่งในหุ่นยนต์ Mecanum Directional มีล้อสี่ล้อเมื่อเคลื่อนที่ไปยังพื้นผิวที่ไม่เรียบทำให้ล้อบางล้อไม่สัมผัสพื้นทำให้มีความผิดพลาดของตำแหน่งมากกว่ารูปแบบ Omni Directional[1] และ Omni Directional[2] ที่ล้อสัมผัสพื้นตลอดเวลา

4.3 การเปลี่ยนรูปแบบของหุ่นยนต์

ทดสอบโดยการกำหนดโจทย์การเคลื่อนที่ให้กับหุ่นยนต์ที่มีระบบ วงจรและโปรแกรมส่วนกลาง กับหุนยนต์ที่ไม่มีวงจรและโปรแกรมส่วนกลางช่วยจัดการคำสั่ง โดยทำการเขียนโปรแกรมและทดสอบหุ่นยนต์ให้ได้ตามโจทย์ที่กำหนด

ตารางที่ 4.4 เวลาในการทดสอบการเปลี่ยนแพลตฟอร์ม

รูปแบบการเคลื่อนที่	เขียนโปรแกรมหุ่นยนต์ครั้งแรก (นาที)	ไม่มีแพลตฟอร์ม (นาที)	มีแพลตฟอร์ม (นาที)
เส้นตรง (2 เมตร)	5	4	2
สามเหลี่ยมด้านเท่า (1 เมตร)	15	12	5
สี่เหลี่ยมจัตุรัส (1 เมตร)	10	8	3
วงกลม (เส้นผ่านศูนย์กลาง 1 เมตร)	15	12	4

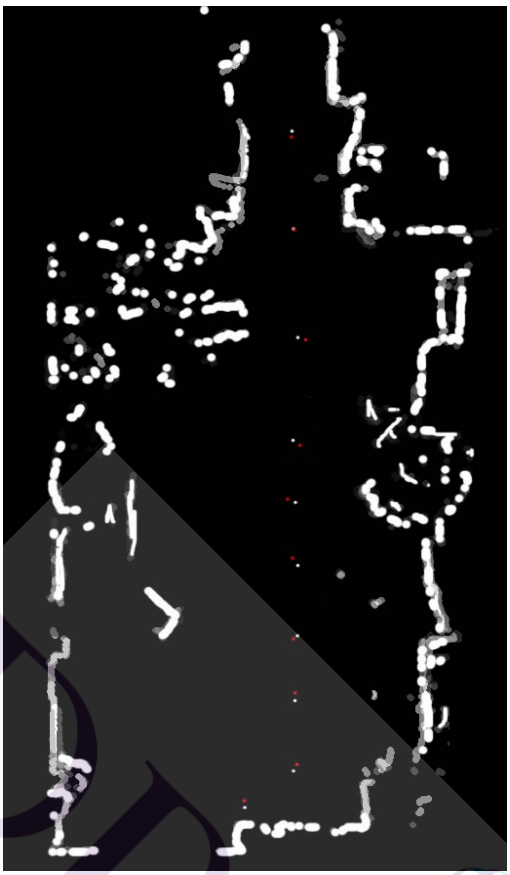
ทดสอบการเปลี่ยนรูปแบบของหุ่นยนต์โดยกำหนดโจทย์การเคลื่อนที่ขึ้นมา 4 แบบ โดยในหุ่นยนต์ที่ได้พัฒนา โปรแกรมส่วนกลางช่วยจัดการคำสั่งจะเขียนคำสั่งการสั่งงานจาก Hi level เมื่อเปลี่ยนรูปแบบของหุ่นยนต์จึงไม่จำเป็นต้องเขียนโปรแกรมการเคลื่อนที่ใหม่อีกครั้ง โดยเมื่อเปลี่ยนแพลตฟอร์มจำเป็นต้องแก้ไขค่าของ Encoder ขนาดของล้อ ความเร็วของมอเตอร์ที่สามารถหมุนได้สูงสุด ต่างกับหุ่นยนต์ที่ไม่มีโปรแกรมช่วยจัดการ จะต้องทำการหาค่า Encoder ขนาดของล้อ ความเร็วของมอเตอร์ เช่นเดียวกับ หุ่นยนต์แพลตฟอร์มแต่จะต้องเขียนโปรแกรมในส่วนของ Hi level ใหม่ลงไปในตัวหุ่นยนต์ ซึ่งการทดสอบนี้ไม่รวมเวลาในการสร้างหุ่นยนต์และการเชื่อมต่อวงจรอิเล็กโทรนิค

4.4 การ Monitor หุ่นยนต์

เปรียบเทียบพิกัดจริงของหุ่นยนต์ กับระบบระบุตำแหน่งและระบบสารสนเทศที่แสดงตำแหน่งทดสอบโดยวางหุ่นยนต์ตามตำแหน่งที่กำหนดไว้ระบบระบุตำแหน่งจะทำการส่งตำแหน่งไปยังฐานข้อมูล แล้วจึงแสดงตำแหน่งผ่านระบบสารสนเทศ

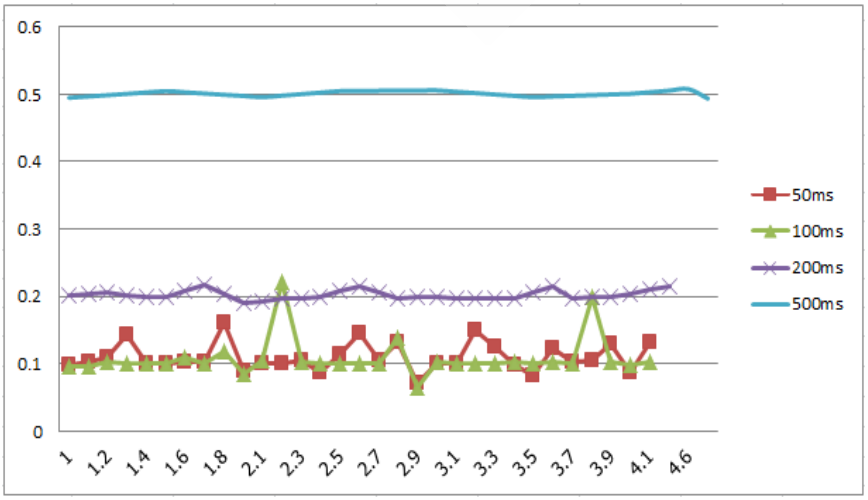
ตารางที่ 4.5 ความถูกต้องของตำแหน่งของหุ่นยนต์ กับระบบตรวจสอบการทำงานของหุ่นยนต์

	robot position	Localization	web monitor
1	1000 , 2316	999 , 2280	999 , 2280
2	1152 , 2328	1157 , 2318	1157 , 2318
3	1144 , 2248	1142 , 2238	1142 , 2238
4	1136 , 2104	1145 , 2095	1145 , 2095
5	1144 , 2016	1147 , 1995	1147 , 1995
6	1160 , 1880	1157 , 1180	1157 , 1180
7	1160 , 1752	1156 , 1750	1156 , 1750
8	1168 , 1648	1160 , 1647	1160 , 1647
9	1168 , 1528	1165 , 1526	1165 , 1526
10	1168 , 1456	1165 , 1570	1165 , 1570



ภาพที่ 4.10 ภาพแสดงตำแหน่งของหุ่นยนต์จากโปรแกรมระบุตำแหน่งกับตำแหน่งจริง

ผิดพลาดที่เกิดขึ้นเกิดจากระบบระบุตำแหน่ง ซึ่งเป็นความคลาดเคลื่อนที่เกิดจากกระบวนการระบุตำแหน่ง +- 20 เซนติเมตร



ภาพที่ 4.11 กราฟแสดง เวลาในการดึงข้อมูลจากฐานข้อมูล

ในการทดสอบ เราทดสอบโดยให้โปรแกรมจำลองการทำงานของหุ่นยนต์ เขียนข้อมูลสถานะปัจจุบันทุกๆ 0.1 วินาที แต่ในส่วนของระบบสารสนเทศ เราต้องการหาช่วงเวลาที่เหมาะสมในการอ่านข้อมูลจากฐานข้อมูล โดยเราได้ให้ระบบสารสนเทศอ่านข้อมูลและนำมาแสดงผลเป็นช่วงๆ และได้ทำการทดสอบ 4 ช่วงเวลาคือ 0.05, 0.1,0.2 และ 0.5 วินาที

ตารางที่ 4.6 ช่วงเวลาที่ใช้ในการอ่านข้อมูลและเวลาเฉลี่ยที่อ่านข้อมูลสำเร็จ

ช่วงเวลาที่ทดสอบ	เวลาเฉลี่ยในการอ่านข้อมูลสำเร็จ
0.05	0.109
0.1	0.108
0.2	0.191
0.5	0.500

จากภาพที่ 4.10 และตารางที่ 4.6 พบว่า ถ้าช่วงเวลาในการอ่านข้อมูลมากกว่าเวลาที่ใช้ในการเขียนข้อมูล (0.1 วินาที) เวลาเฉลี่ยที่อ่านข้อมูลสำเร็จจะใกล้เคียงกับเวลาที่ใช้เขียนข้อมูล ดังนั้นค่าช่วงเวลาอ่านข้อมูลที่เหมาะสมคือการอ่านข้อมูลทุกๆ 0.2 วินาที เนื่องจากถ้าเวลาในการดึงข้อมูลช้าไปก็จะมีผลต่อการแสดงผล ทำให้การเคลื่อนที่ของหุ่นยนต์ไม่ราบรื่น ถ้าอ่านข้อมูลเร็วไปจะเป็นการเพิ่มภาระการทำงานให้ตัวระบบฐานข้อมูล

บทที่ 5
บทสรุป และข้อเสนอแนะ

ในบทนี้จะเป็นการอภิปรายเพื่อสรุปผลที่ได้จากการทดสอบงานวิจัย รวมทั้งข้อกำจัดของอุปกรณ์ และข้อเสนอแนะสำหรับแนวทางในการพัฒนางานวิจัยนี้ต่อไปเพื่อแก้ข้อบกพร่องของงานวิจัยให้มีประสิทธิภาพมากขึ้น

5.1 สรุปผลการวิจัย

การเคลื่อนที่ของ Mobile Robot มีหลายรูปแบบต้องอาศัยวงจรควบคุมที่สามารถเพิ่มลดจำนวนและควบคุมมอเตอร์ได้แม่นยำการนำการควบคุม PID มอเตอร์แต่ละตัวโดยในหัวข้อที่ 4.1 ผลการทดลองวงจรมีประสิทธิภาพมากกว่าความต้องการของระบบในการควบคุมโดยใช้ PID โดยสามารถสั่งงานวงจรที่ความเร็ว 115200 bps ที่ความถี่ 80 Hz สูงกว่าความต้องการของระบบ PID มีการคำนวณอยู่ที่ 10 Hz

จากผลการวิจัยการที่มีระบบควบคุมส่วนกลางสามารถ จัดการกับคำสั่งที่ถูกสั่งมาจากส่วน Hi level ได้ จากการทดสอบการสั่งงานการเคลื่อนที่หัวข้อ 4.2 และการมีระบบควบคุมส่วนกลางสามารถลดเวลาในการเปลี่ยนรูปแบบการเคลื่อนที่ของหุ่นยนต์ได้ โดยอาศัยการทำงานร่วมกันระหว่างฮาร์ดแวร์และระบบควบคุม เมื่อเปลี่ยนรูปแบบการเคลื่อนที่ จะต้องมีการเชื่อมต่อวงจรและเขียนโปรแกรมในส่วนสั่งงานการเคลื่อนที่ใหม่ การมีระบบควบคุมส่วนกลางจะช่วยลดเวลาในส่วนสั่งงาน Hi level

งานวิจัยนี้แสดงให้เห็นถึงข้อดีของการมีระบบควบคุมส่วนกลางที่คอยจัดการคำสั่งในการเคลื่อนที่ของ Mobile Robot ที่มีล้อไม่เกิน 4 ล้อ ทำให้สามารถลดเวลาในการพัฒนา Mobile Robot โดยอาศัยการทำงานร่วมกันของวงจรที่สามารถลดความซับซ้อนในการเชื่อมต่อวงจรอิเล็กโทรนิค มีระบบตรวจสอบการทำงานเพื่อตรวจสอบการเคลื่อนที่ของหุ่นยนต์

สรุปผลตามวัตถุประสงค์ของงานวิจัย สามารถสรุปผลได้ดังนี้

1) สามารถออกแบบและสร้างวงจรที่ใช้ในการขับเคลื่อน และโครงสร้างหุ่นยนต์ที่สามารถประยุกต์ใช้ได้สองรูปแบบ Omni Directional Drive และ Mecanum drive

2) สามารถสร้างระบบจัดการคำสั่งในการเคลื่อนที่ Middleware เพื่อสั่งงาน Low Level โดยรองรับ คำสั่งความเร็วและทิศทางการเคลื่อนที่ของหุ่นยนต์ Omni Directional Drive และ Mecanum drive

3) สามารถสร้างระบบที่คอยตรวจสอบ สถานะของการทำงาน และแสดงตำแหน่งของหุ่นยนต์ผ่านระบบ ระบบสารสนเทศโยดึงข้อมูลมาแสดงผลทุกๆ 200ms

5.2 ข้อจำกัดและแนวทางแก้ไขของงานวิจัย

หุ่นยนต์ที่ใช้ในการทดสอบ รูปแบบ Mecanum Wheel ในการเคลื่อนที่ล้อทั้ง 4 ล้อ จำเป็นจะต้องสัมผัสพื้นตลอดเวลา แต่หุ่นยนต์ที่ได้ออกแบบไม่มีระบบกันสะเทือนทำให้ล้อทั้ง 4 สัมผัสพื้นไม่พร้อมกันขณะเคลื่อนที่ไปบนพื้นผิวที่ไม่เรียบ ทำให้เมื่อสั่งงานการเคลื่อนที่ไม่สามารเคลื่อนที่ได้เต็มประสิทธิภาพ

ส่วนของ Middle level ได้พัฒนาส่วนของการสั่งงานความเร็วและทิศทางการเคลื่อนที่เป็นหลัก ยังไม่สามารถสั่งงานการเคลื่อนที่ไปตามระยะทางที่กำหนดได้อย่างแม่นยำ และยังมีรูปแบบการเคลื่อนอีกหลายแบบที่ไม่ได้ทดสอบ

วงจรที่ใช้ขับมอเตอร์เนื่องจากวงจรที่ออกแบบมาให้อยู่ในหุ่นยนต์ Mobie Robot ในงานวิจัยนี้มีขนาดเล็กทำให้ไม่สามารถขับมอเตอร์ที่ใช้กระแสสูงกว่า 20 แอมแปร์ได้ แต่สามารถเปลี่ยนวงจรขับมอเตอร์ที่ออกแบบขึ้นเป็นวงจรที่มีขายทั่วไป ซึ่งจำเป็นต้องเชื่อมต่อวงจรเพิ่มเติม

5.3 ข้อเสนอแนะและแนวทางการพัฒนาในอนาคต

5.3.1 ควรจะทดสอบวงจรขับมอเตอร์เทียบกับวงจรในท้องตลาดมากขึ้น

5.3.2 ทดสอบหา เวลาที่ใช้ในการสั่งการ PID แต่ละครั้งให้ละเอียดมากขึ้น

5.3.3 เพิ่มจำนวนรูปแบบของหุ่นยนต์ที่ใช้ในการทดสอบ

5.3.4 ศึกษา อุปกรณ์ หรือ เซนเซอร์ ที่จำเป็นสำหรับหุ่นยนต์ พัฒนาวงจรโปรแกรมเพื่อรองรับอุปกรณ์ หรือ เซนเซอร์ รุ่นใหม่ๆ

บรรณานุกรม

บรรณานุกรม

ภาษาต่างประเทศ

Dong-Eon Kim, Ha-Neul Yoon, Ki-Seo Kim, Sreejith M.S1 and Jang-Myung Lee. (June 28 - July 1, 2017). “Using Current Sensing method and Fuzzy PID Controller for Slip Phenomena Estimation and Compensation of Mobile Robot.” 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), pp. 397-401.

Guo Suoli, Liu Yanfei & Li Qi. Electronic Design. (2008). “Simulation and Integrated Application Based on Multisim9 [M].” Beijing: Post and Telecom Press

Innovation brings rewards for Siasun. China Daily [ออนไลน]. สืบค้นเมื่อ 7 เมษายน 2558. จาก http://africa.chinadaily.com.cn/weekly/2015-04/17/content_20454360.htm

J.A. Janet, R. Gutierrez-Osuna, T.A. Chase, M. White, R.C. Luo. (1995). "Global self-localization for autonomous mobile robots using region- and feature-based neural networks." IEEE IECON 21st International Conference, pp. 1142-1147.

Jefri Efendi Mohd Salih, Mohamed Rizon, Sazali Yaacob, Abdul Hamid Adom and Mohd Rozailan Mamat. (2006). “Designing Omni-Directional Mobile Robot with Mecanum Wheel” American Journal of Applied Sciences, pp. 1831-1835.

Keiji Nagatani, Seiga Kiribayashi, Yoshito ,Satoshi Tadokoro, Takeshi Nishimura, Tomoaki Yoshida, Eiji Koyanagi and Yasushi Hada (1-5 Nov. 2011). “Redesign of rescue mobile robot Quince.” IEEE International Symposium on Japan, pp. 13-15.

Meet the drone that already delivers your packages. Kiva robot teardown. Robohub [ออนไลน]. สืบค้นเมื่อ 1 กุมภาพันธ์ 2559. จาก http://robohub.org/meet-the-drone-that-already-delivers-your-packages-kiva-robot-teardown/

Mobile robot. wikipedia [ออนไลน]. สืบค้นเมื่อ 1 มกราคม 2560. จาก https://en.wikipedia.org/wiki/Mobile_robot

Peter R. Wurman, Raffaello D’Andrea, and Mick Mountz. (2007, July 13–17). “Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses.” AAAI 2007, pp. 1752-1759.

Wai Phyo Aung . (Dec 2007). “Analysis on Modeling and Simulink of DC Motor and its Driving System Used for Wheeled Mobile Robot”. Proceedings of World Academy of Science: Engineering & Technolog, pp.229

Wheel Control Theory, Robotplatform [ออนไลน], สืบค้นเมื่อ 26 เมษายน 2559. จาก http://www.robotplatform.com/knowledge/Classification_of_Robots/wheel_control_theory.html

ประวัติผู้เขียน

ชื่อ-นามสกุล	พีรเดช เปรมใจ
ประวัติการศึกษา	วิศวกรรมศาสตร์บัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยธุรกิจบัณฑิตย์