การสำรวจและการสร้างแผนที่ในพื้นที่ปิดด้วย ROS

นายสวาส อาจสาลี

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมเมคคาทรอนิกส์

มหาวิทยาลัยเทคโนโลยีสุรนารี

ปีการศึกษา 2561

ROS BASED LOCALIZATION AND MAPPING FOR CLOSED AREA SURVEYING

Swas Oajsalee

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Mechartronic Engineering

Suranaree University of Technology

Academic Year 2018

การสำรวจและการสร้างแผนที่ในพื้นที่ปิดด้วย ROS

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

คณะกรรมการสอบวิทยานิพนธ์

(อ. ดร.ธีทัต คลวิชัย)
ประธานกรรมการ

(อ. ดร.โศรฎา แข็งการ)
กรรมการ (อาจารย์ที่ปรึกษาวิทยานิพนธ์)

(อ. ดร.วิทูรย์ เห็มสุวรรณ)
กรรมการ

(ศ. ดร.สันติ แม้นศิริ)
รองอธิการบดีฝ่ายวิชาการและพัฒนาความเป็นสากล

(รศ. ร.อ. ดร.กนต์ธร ชำนิประศาสน์)
คณบดีสำนักวิชาวิศวกรรมศาสตร์

สวาส อาจสาลี : การสำรวจและการสร้างแผนที่ในพื้นที่ปิดด้วย ROS (ROS BASED LOCALIZATION AND MAPPING FOR CLOSED AREA SURVEYING)
อาจารย์ที่ปรึกษา : อาจารย์ ดร.โศรฏา แข็งการ, 99 หน้า

งานวิจัยนี้นำเสนอหุ่นยนต์สำรวจในพื้นที่ปิดตามระบบ Turtlebot และ Robot Operating System (ROS) ในการสำรวจและทำแผนที่พื้นที่ที่ไม่สามารถเข้าถึงได้ เช่น อาคารเก่า พื้นที่ภัยพิบัติ อุโมงค์และถ้ำ เป็นต้น โดยการใช้เลเซอร์สแกนเนอร์เพื่อตรวจจับสภาพแวดล้อมรอบ ๆ หุ่นยนต์สำรวจและ ROS ถูกใช้เพื่อสร้างแผนที่

การทดลองสภาพแวดล้อมจริงที่ได้จำลองขึ้นเอง ในการทดลองแบ่งออกเป็นสองส่วนคือ ส่วนแรกจะการสำรวจและสร้างแผนที่และส่วนที่สองจะใช้อัลกอริธึม SLAM เพื่อระบุการเคลื่อนที่แบบอัตโนมัติของหุ่นยนต์ ซึ่งหุ่นยนต์สามารถสำรวจพื้นที่แคบและที่พื้นที่ปิดได้ตลอดจนสภาพแวดล้อมที่เป็นอุปสรรคและสามารถระบุการเคลื่อนที่ที่มีประสิทธิภาพของหุ่นยนต์สำรวจเมื่อการทำแผนที่เสร็จสิ้น

จากการทดลองการเคลื่อนที่ในท่อที่เป็นแบบจำลองที่สร้างขึ้นเองโดยผู้วิจัยนั้น ในการทดลองที่ 1 การสำรวจและการสร้างแผนที่ ผลที่ได้คือหุ่นยนต์สำรวจสามารถสำรวจและสร้างแผนที่ได้ตามที่ต้องการ ส่วนการทดลองที่ 2 การระบุตำแหน่งการเคลื่อนที่อัตโนมัติของหุ่นยนต์สำรวจพบว่าหุ่นยนต์สำรวจสามารถเคลื่อนที่เข้าไปในท่อได้นั้นจะต้องมีการปรับค่าพารามิเตอร์ Inflation Radius และ Cost Scaling Factor เพื่อเพิ่มประสิทธิภาพของการเคลื่อนที่แบบอัตโนมัติ

สาขาวิชา วิศวกรรมเครื่องกล ลายมือชื่อนักศึกษา _____

ปีการศึกษา 2561 ลายมือชื่ออาจารย์ที่ปรึกษา _____

SWAS OAJSALEE : ROS BASED LOCALIZATION AND MAPPING FOR CLOSED AREA SURVEYING. THESIS ADVISOR : SORADA KHAENGKARN, Ph.D., 99 PP.

ROS/SLAM/SURVEYING ROBOT

This research introducestherobots for closed areas survey based on the Turtlebot and Robot Operating System (ROS) to survey and createmap inaccessible areas such as old buildings, disaster areas, tunnels and caves using laser scanner to detect surround environment. The survey robot and ROS is used to create the map.

Experiment in simulating real-life environment, the experiment was divided into two parts; the first part explored and created maps, and the second used the SLAM algorithm to identify the robot's automatic movement. The robot can explore narrow and closed areas as well as harsh environments, and can identify the effective movement of the robots when the map is completed.

From the movement experiments in sewer modelcreated by the researcher; experiment 1 - surveying and mapping: the robot can survey and create map as needed. Experiment 2 - autonomous location of the robot: the robot can move into the tube with adjustment of the Inflation Radius and Cost Scaling Factor parameters to optimize the autonomous motion.

School of Mechanical Engineering Student's Signature _____

Academic Year 2018 Advisor's Signature _____

กิตติกรรมประกาศ

วิทยานิพนธ์นี้สำเร็จลุล่วงด้วยดีเนื่องจากได้รับความช่วยเหลืออย่างดียิ่ง ทั้งด้านวิชาการและด้านการดำเนินงานวิจัยจากบุคคลและกลุ่มบุคคลต่าง ๆ ได้แก่

อาจารย์ ดร.โศรฎา แข็งการและ อาจารย์ ดร.สุรเดช ตัญตรัยรัตน์ อาจารย์สาขาวิชาวิศวกรรมเครื่องกล มหาวิทยาลัยเทคโนโลยีสุรนารีอาจารย์ที่ปรึกษาวิทยานิพนธ์ที่ให้โอกาสทางการศึกษาให้คำแนะนำปรึกษาช่วยแก่ปัญหาและให้กำลังใจแก่ผู้วิจัยมาโดยตลอด รวมทั้งช่วยตรวจทานและแก้ไขวิทยานิพนธ์เล่มนี้จนเสร็จสมบูรณ์

ท้ายนี้ ผู้วิจัยขอกราบขอบพระคุณบิดา มารดา ที่ให้การอุปการะอบรมเลี้ยงดู ตลอดจนส่งเสริมการศึกษา และให้กำลังใจเป็นอย่างดีเสมอมาตลอดจนครูอาจารย์ที่เคารพทุกท่านที่ได้ประสิทธิ์ประสาทวิชาความรู้และถ่ายทอดประสบการณ์ที่ดีให้แก่ผู้วิจัยตลอดมาจนทำให้ประสบความสำเร็จในชีวิตจนกระทั่งวิทยานิพนธ์ฉบับนี้สำเร็จ

สวาส อาจสาลี

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

คำอธิบายสัญลักษณ์และคำย่อ

SLAM	=	Simultaneous Localization and Mapping
AMCL	=	Adaptive Monte Carlo Localization
IMU	=	Inertial Measurement Unit
ROS	=	Robot Operating System
Lidar	=	Light Detection and Ranging
C	=	ความเร็วของแสง เมื่อทราบระยะทางก็จะสามารถหาระยะทางได้ หาร 2
S	=	ระยะทางของสิ่งกีดขวางเทียบกับเซนเซอร์ Lidar
D	=	ระยะห่างระหว่างล้อกับรัศมีของล้อ
T_e	=	การเคลื่อนที่ในระยะเวลาสั้น ๆ ในช่วงเวลา
v_l	=	ความเร็วในการหมุนล้อซ้าย
v_r	=	ความเร็วในการหมุนล้อขวา
v_k	=	ความเร็วเชิงเส้น
w_k	=	ความเร็วเชิงมุม
Odometry	=	การวัดการเคลื่อนที่ของหุ่นยนต์ในระบบพิกัด

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของปัญหา

ในอดีตหุ่นยนต์ที่ใช้ในการสำรวจมีหลากหลายลักษณะ ส่วนมากเป็นหุ่นยนต์แบบควบคุมสั่งการด้วยรีโมทคอนโทรล ไม่สามารถทำงานได้อย่างอัตโนมัติ เนื่องจากพื้นที่สำรวจมักเป็นพื้นที่ที่ไม่มีแผนที่ ทำให้หุ่นยนต์เข้าไปในพื้นที่ที่ไม่รู้จักทำได้ยาก จึงต้องทำการควบคุมด้วยมนุษย์ และหุ่นยนต์ที่เคลื่อนที่อัตโนมัติ เช่น หุ่นยนต์เดินตามเส้น เป็นต้น จะใช้ได้ก็ต่อเมื่อรู้เส้นทางการเคลื่อนที่ของหุ่นยนต์และทำเส้นทางการเคลื่อนที่ของหุ่นยนต์ไว้ก่อน นอกจากนี้หุ่นยนต์ยังไม่สามารถหลบหลีกสิ่งกีดขวางได้โดยอัตโนมัติ ทำได้เพียงหยุดเคลื่อนที่เมื่อมีสิ่งกีดขวาง ปัจจุบันมีวิธีการที่ถูกพัฒนาขึ้นใหม่และได้รับความนิยมมากในงานวิจัยที่เกี่ยวข้องกับหุ่นยนต์ที่เคลื่อนที่แบบอัตโนมัติ คือ SLAM ย่อมาจาก Simultaneous Localization and Mapping โดยหุ่นยนต์จะสร้างแผนที่ด้วยเซนเซอร์ต่าง ๆ เช่น เซนเซอร์ Sonar, Lidarหรือ kinect[1] พร้อมกับจดจำตำแหน่งของตนเองในแผนที่ เมื่อหุ่นยนต์สำรวจพื้นที่และสร้างแผนที่แล้วเสร็จ หุ่นยนต์จะสามารถเดินทางไปยังตำแหน่งต่าง ๆ ในแผนที่ที่สร้างขึ้นได้โดยอัตโนมัติพร้อมกับสามารถหลบหลีกสิ่งกีดขวางได้

ROS [2] ,[3] เป็นระบบปฏิบัติการคล้ายกับ Window หรือ Linux แต่เป็นระบบปฏิบัติการสำหรับหุ่นยนต์เป็น Middleware ที่มี Library และ Tool สำหรับการทำหุ่นยนต์ไว้ค่อนข้างครบมีทั้ง Driver ของอุปกรณ์ต่าง ๆ รวมไปถึง Algorithm และที่สำคัญเป็น Open source ภาษาที่ใช้ในการเขียนโปรแกรมจะใช้ C++ และ Python ซึ่งเป็นระบบปฏิบัติการหุ่นยนต์ที่ได้รับพัฒนาและเป็นที่นิยมมากที่สุดในปัจจุบัน การทำงานของซอฟท์แวร์หุ่นยนต์ในระบบ ROS จะประกอบไปด้วย Node ต่าง ๆ ที่ทำงานเกี่ยวข้องกัน มีการสื่อสาร แลกเปลี่ยนข้อมูลระหว่างกัน ซึ่งข้อดีของ ROS ก็คือเป็น Open source ประกอบไปด้วย ซอฟท์แวร์, ไลบรารี, หน่วยโปรแกรมต่าง ๆ สามารถทำงานร่วมกันได้ โดยกลไกสำคัญในการสื่อสารระหว่าง Node เพื่อให้ Node อื่นสามารถรับข้อมูลได้ โดยที่ Node แต่ละ Node อาจจะพัฒนาภาษาที่แตกต่างกัน

การที่จะทำให้หุ่นยนต์สามารถเคลื่อนที่ไปในตำแหน่งต่าง ๆ ที่ต้องการได้ โดยไม่ชนกับสิ่งกีดขวางจะต้องอาศัยการนำทางหรือ Navigation ซึ่งมีอยู่ในระบบปฏิบัติการของ ROS แต่ ROS Navigation [4] จะสามารถทำงานได้นั้นจำเป็นที่จะต้องมีฮาร์ดแวร์ที่สำคัญได้แก่ เซนเซอร์ Lidarและส่วนของการขับเคลื่อนหุ่นยนต์ซึ่งขึ้นอยู่กับประเภทของหุ่นยนต์ที่ใช้ได้แก่ มอเตอร์กระแสตรง, เซอร์โว

มอเตอร์ ฯลฯ และส่วนที่เป็นบอร์ดวงจรอิเล็กทรอนิกส์ต่าง ๆ ที่เอาไว้ควบคุมการทำงานของหุ่นยนต์[3]

ในบทความนี้ต้องการพัฒนาหุ่นยนต์สำรวจในพื้นที่ปิด ที่สามารถทำงานได้อย่างอัตโนมัติ ซึ่งสามารถนำไปประยุกต์ใช้ในการสำรวจพื้นที่ประสพภัย อาคารร้าง หรือในถ้ำที่คนไม่สามารถเข้าไปสำรวจได้ ด้วยวิธี SLAM โดยใช้ระบบปฏิบัติการหุ่นยนต์ ROS ร่วมกับเซนเซอร์ Lidarในการตรวจจับสิ่งกีดขวางหรือวัตถุที่อยู่รอบ ๆ โดยการสร้างแผนที่นั้นจะใช้วิธีการ SLAM [4] ของ ROS ซึ่งจะสร้างแผนที่โดยการส่งเป็น Topic แล้วทำการจัดเก็บไว้ใน Map Server ในส่วนของการรู้ตำแหน่งหรือ Localization สำหรับ ROS มีอัลกอริทึมที่เรียกว่า AMCL (Adaptive Monte Carlo Localization) ซึ่งจะทำให้รู้ว่าหุ่นยนต์อยู่จุดใดในแผนที่ [3] ในการทำงานของหุ่นยนต์สำรวจอัตโนมัติด้วยวิธี SLAM นั้น แบ่งออกเป็นสองส่วนคือ ส่วนแรกการสำรวจและสร้างแผนที่ และส่วนที่สองระบุตำแหน่งเคลื่อนที่อัตโนมัติของหุ่นยนต์สำรวจ

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 นำเสนอหุ่นยนต์สำรวจและสร้างแผนที่

1.2.2 นำเสนอการระบุตำแหน่งการเคลื่อนที่อัตโนมัติของหุ่นยนต์สำรวจในพื้นที่ปิด

1.3. ขอบเขตของการวิจัย

1.3.1 การพัฒนาหุ่นยนต์สำรวจในพื้นที่ปิดที่สามารถทำงานได้อย่างอัตโนมัติ ซึ่งสามารถนำไปประยุกต์ใช้ในการสำรวจพื้นที่ประสพภัย อาคารร้าง หรือในถ้ำที่คนไม่สามารถเข้าไปสำรวจได้ ด้วยวิธี SLAM โดยใช้ระบบปฏิบัติการหุ่นยนต์ ROS โดยมีเป้าหมายงานวิจัยสามารถสรุปได้ดังนี้

1.ใช้เซนเซอร์ Lidarในการสำรวจและตรวจสอบสิ่งกีดขว้าง

2.ใช้เซนเซอร์ IMUในการวัดความเร่งจากการเคลื่อนที่ของหุ่นยนต์สำรวจ

3.ใช้เซนเซอร์ Gyroscopeในการวัดความเร็วเชิงมุมจากการเคลื่อนที่ของหุ่นยนต์สำรวจ

4.ระบุตำแหน่งพร้อมกับการสร้างแผนที่สิ่งแวดล้อมแบบปิดเป็น 2 มิติ

5.ระบุตำแหน่งการเคลื่อนที่อัตโนมัติของหุ่นยนต์สำรวจ

6.พื้นที่สำรวจเป็นแบบพื้นราบ

7.หุ่นยนต์ขับเคลื่อนด้วยมอเตอร์ DYNAMIXEL XM series

8.รับส่งข้อมูลระหว่างหุ่นยนต์และคอมพิวเตอร์เป็นแบบไร้สาย

9.มีการรับส่งข้อมูลและแสดงผลบนแผนที่แบบเวลาจริง (Real time)

1.4. วิธีการดำเนินการของงานวิจัย

1.4.1 ศึกษางานวิจัยที่เกี่ยวข้อง

1.4.2 ศึกษารูปแบบของโครงสร้างและองค์ประกอบส่วนต่างๆ ของหุ่นยนต์สำรวจ

1.4.3 ศึกษาและทดลองใช้อุปกรณ์วัดระยะทางด้วยเซนเซอร์ Lidar

1.4.4 ศึกษาทฤษฎีพื้นฐานของ SLAMในการใช้ระบบปฏิบัติการหุ่นยนต์ ROS

1.4.5 ศึกษาทฤษฎีพื้นฐานของ SLAM Navigation ในการใช้ระบบปฏิบัติการหุ่นยนต์ ROS

1.4.6 ออกแบบและสร้างหุ่นยนต์สำรวจ

1.4.7 ทดสอบการควบคุมการเคลื่อนที่ของหุ่นยนต์สำรวจในการเคลื่อนที่แบบไร้สาย

1.4.8 ทดลองและวัดผลการทดลอง

1.4.9 สรุปผลการทดลอง

1.4.10 จัดทำรูปเล่มและเนื้อหาวิทยานิพนธ์

1.5. ประโยชน์ที่คาดว่าจะได้รับ

1.5.1ได้ออกแบบและสร้างหุ่นยนต์สำรวจด้วยวิธี SLAMโดยใช้ระบบปฏิบัติการหุ่นยนต์ ROS

1.5.2ได้พัฒนาหุ่นยนต์สำรวจในพื้นที่ปิดที่สามารถทำงานได้อย่างอัตโนมัติ

1.5.3ได้สร้างแผนที่ในแบบของ SLAM โดยใช้อุปกรณ์วัดระยะทางด้วยเซนเซอร์ Lidar

1.5.4สามารถระบุตำแหน่งพร้อมกับการสร้างแผนที่เป็นแบบ 2 มิติ

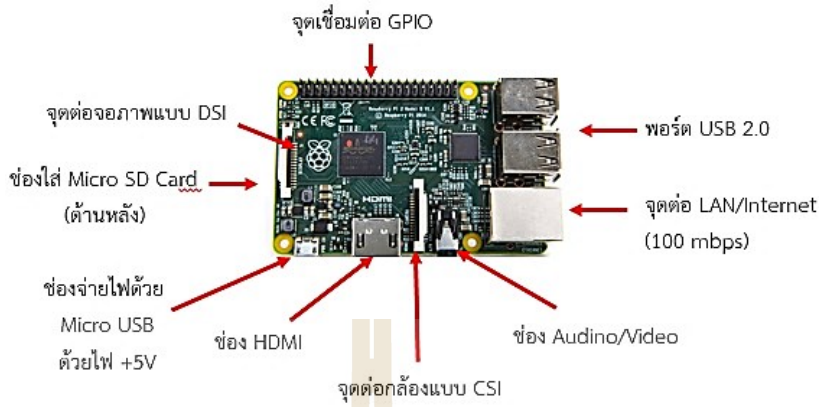
1.5.5ได้เรียนรู้หลักการทำงานของอัลกอริทึมที่เป็นแบบ Real-Time

บทที่ 2
ทฤษฎีเกี่ยวข้อง

การพัฒนาหุ่นยนต์สำรวจที่จะนำมาใช้กับROS จำเป็นที่จะต้องศึกษาความรู้พื้นฐานที่สำคัญต่างๆ ได้แก่ ระบบพิกัด เฟรม อุปกรณ์ต่างๆ ที่มีความจำเป็น เซนเซอร์ เป็นต้น เนื่องจากการพัฒนาหุ่นยนต์สำรวจเกี่ยวกับROS เป็นเรื่องใหญ่และมีความซับซ้อน ยุ่งยากมากกว่าการพัฒนาซอฟท์แวร์อื่นๆ นอกจากทักษะทางด้านการเขียนโปรแกรมแล้ว ยังต้องมีทักษะด้านฮาร์ดแวร์ ได้แก่ ฟิสิกส์ อิเล็กทรอนิกส์ คอมพิวเตอร์และระบบการปฏิบัติการ เป็นต้น ซึ่งROS เป็นชุดของซอฟท์แวร์ชุดหนึ่งที่มีวัตถุประสงค์สำหรับงานทางด้านหุ่นยนต์ ซึ่งชุดซอฟท์แวร์นี้ทำงานบนระบบปฏิบัติการสำหรับคอมพิวเตอร์ เช่น Linux, Windows, MacOS เป็นต้น กลไกในการทำงานของ ROS จะมีส่วนของ Master ซึ่งเป็น Node หลักที่ควบคุม Node อื่นๆ ทั้งหมดในระบบ ดังนั้นในบทนี้จะเป็นการกล่าวถึงทฤษฎีที่ใช้ในการออกแบบและสร้างหุ่นยนต์สำรวจในการทำงานร่วมกับ ROS

2.1 บอร์ด Raspberry Pi

บอร์ดราสเบอร์รี่พาย (Raspberry Pi) Raspberry Pi เป็นคอมพิวเตอร์ในบอร์ดเดียว (Single Board Computer) พัฒนาขึ้นในประเทศสหรัฐอเมริกา โดย Raspberry Pi Foundation ภายใต้การทำงานขององค์ไม่หวังผลกำไรสร้างคอมพิวเตอร์ในบอร์ดเดียวเพื่อใช้สำหรับการสอนและสำหรับเรียนของนักศึกษาในสาขา Computer Science ซึ่ง Raspberry Pi นั้นมีขนาดเล็กมาก มีความสามารถในการรองรับระบบปฏิบัติการ Linux ที่เรียกว่า Raspbian ที่บรรจุลงใน SD การ์ดสำหรับการพัฒนาไปสู่บอร์ด Embedded Linux ซึ่งจากรูปที่ 2.1 นั้นจะเห็นส่วนประกอบต่างๆ ของบอร์ด Raspberry Pi ที่มีจุดเชื่อมต่ออุปกรณ์อินพุตเอาต์พุตทั้งผ่านพอร์ต USB, LAN, HDMI, CSI, DSI, AUDIO, ช่องสัญญาณภาพและ GPIO (General Purpose Input Output) สำหรับต่อกับวงจรหรืออุปกรณ์อิเล็กทรอนิกส์ต่างๆ ซึ่งอุปกรณ์ Raspberry Pi ยังมีคุณสมบัติทางเทคนิคดังต่อไปนี้



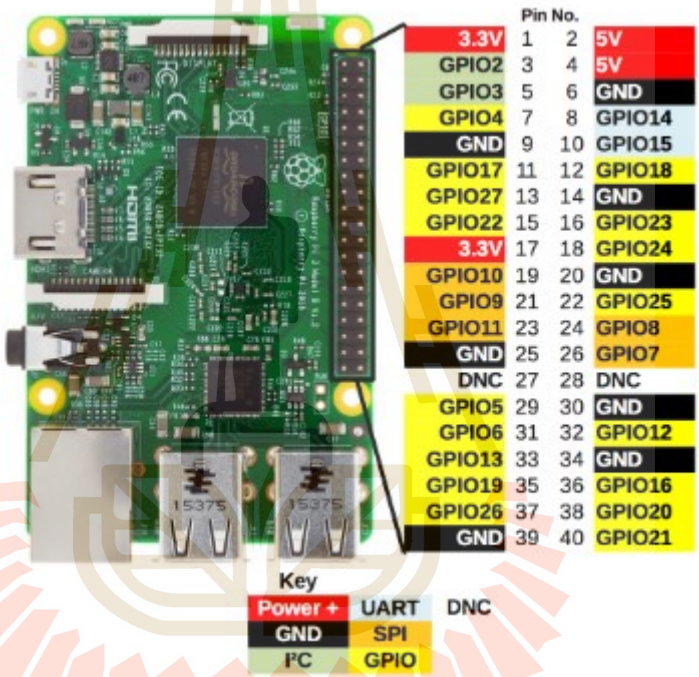
รูปที่ 2.1 ส่วนประกอบของบอร์ด Raspberry Pi

2.1.1 คุณสมบัติทางเทคนิค

- Chip ควบคุมหลัก Broadcom BCM2835 เทียบเท่าซึ่งรวม CPU, หน่วยประมวลกราฟิกหรือ GPU และหน่วยความจำ SD RAM ไว้ภายในตัวเดียวกัน
- หน่วยประมวลผลกลางหรือ CPU ARM11 Core ARM1176JZF-S ความเร็ว 700MHz
- หน่วยประมวลกราฟิกหรือ GPU Broadcom Video core IV หรือเทียบเท่ารองรับการแสดงผลผ่านจอภาพที่ใช้จุดต่อแบบ HDMI
- หน่วยความจำ SDRAM 512 MB
- USB 2.0 (4 พอร์ต)
- เอาต์พุต RCA และ HDMI เอาต์พุตสัญญาณวีดีโอสาหรับต่อกับโทรทัศน์หรือจอแสดงผล
- เอาต์พุตเสียงแจ๊คหูฟังขนาด 3.5 มิลลิเมตร
- พอร์ต Ethernet หรือ LAN
- พอร์ตอินพุตเอาต์พุต GPIO(General Purpose Input/Output) ที่มีขาต่อแบบบัส SPI (Serial Peripheral Interface Bus), I2C, I2S
- ขาสัญญาณรับส่งข้อมูลอนุกรมหรือ UART
- Socket ของ SD การ์ด
- ไฟเลี้ยง 5 โวลต์700 มิลลิแอมป์
- ขนาด 85.60 x 53.93 มิลลิเมตร

2.1.2 จุดเชื่อมต่อแบบ GPIO

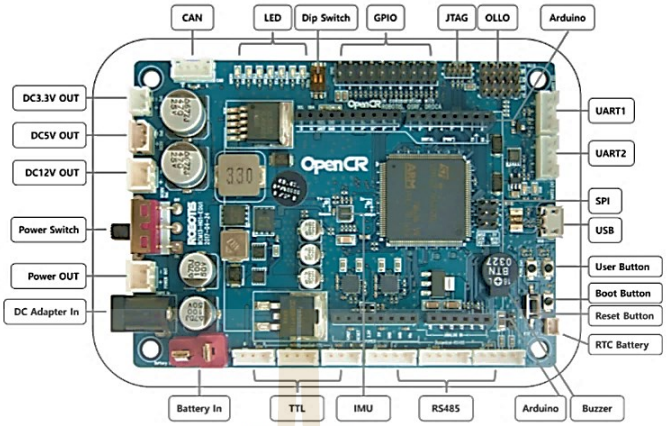
Raspberry Pi สามารถเชื่อมต่อกับอุปกรณ์อิเล็กทรอนิกส์ได้ผ่าน GPIO (General Purpose Input Output) ซึ่งประกอบด้วย UART, SPI, PWM, I2C และอื่นๆ เพื่อใช้ในการควบคุมและสื่อสารกับอุปกรณ์อิเล็กทรอนิกส์ ซึ่งพอร์ต GPIO เป็นพอร์ตอินพุตเอาต์พุตอเนกประสงค์สามารถนำไปใช้งานได้ 21 ขา (ในเวอร์ชั่น Rev.2 512MB) จึงสามารถเชื่อมต่อกับอุปกรณ์อิเล็กทรอนิกส์ได้หลากหลายชนิดด้วยกันรวมทั้งมีพอร์ตสำหรับจ่ายไฟให้กับอุปกรณ์อิเล็กทรอนิกส์ด้วยซึ่งมีทั้งขนาด 3V, 5V และกราวด์ ซึ่งได้แสดงไว้ในรูปที่ 2.2



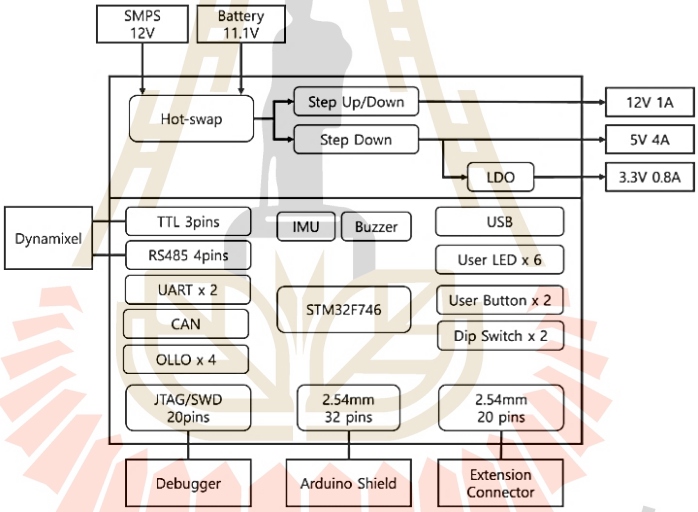
รูปที่ 2.2 จุดเชื่อมต่อแบบ GPIO ของ Raspberry Pi Model B

2.2 บอร์ด OpenCR

OpenCR (โมดูลควบคุมโอเพนซอร์สสำหรับ ROS) เป็นบอร์ดที่ทำงานร่วมกับระบบ ROS ได้ดีและใช้เป็นตัวควบคุมหลักของหุ่นยนต์สำรวจ ซึ่งข้อมูลของฮาร์ดแวร์ของ OpenCR จะประกอบไปด้วย เฟิร์มแวร์ ข้อมูล Gerber และ Source code



รูปที่ 2.3 ส่วนประกอบของบอร์ด OpenCR



รูปที่ 2.4 Block diagram ของบอร์ด OpenCR

2.2.1 องค์ประกอบของบอร์ด OpenCR

2.2.2.1 High PerformanceSTMicroelectronics STM32F7 ไมโครคอนโทรลเลอร์

STM32F7 ไมโครคอนโทรลเลอร์เป็น ARM Cortex-M7 ที่เหนือกว่า cores รุ่นก่อนหน้านี้ เช่น Cortex-M4 รุ่น STM32F7 MCU คือมีขีดความสามารถประมวลผลสัญญาณดิจิตอล (DSP) ได้มากขึ้นประมาณสองเท่า ทำให้ MCUs เหมาะสำหรับการใช้งานที่

ต้องการความเร็วสูง หรือระบบเสียงแบบหลายช่องสัญญาณ วีดีโอ ไวร์เลส การจดจำการเคลื่อนไหวหรือการควบคุมมอเตอร์ นอกจากนี้การนำข้อดีของกลไกประมวลผลกราฟิก ST's advanced ART Accelerator™ และแคช L1 on-chip ของ Cortex-M7 มาใช้ STM32F7 MCUs ยังช่วยส่งประสิทธิภาพการทำงานตามทฤษฎีสูงสุดของ Cortex-M7 และนำเสนอ 1082 CoreMark/462 DMIPS ที่ 216 MHz ไม่ว่าโค้ดจะถูกดำเนินการจากแฟลชภายในหรือหน่วยความจำภายนอกก็ตาม ที อุปกรณ์มีประสิทธิภาพการประมวลผลสูง 6 CoreMark/mW ที่ 1.8 V และประสิทธิภาพการทำงานที่ยอดเยี่ยมในการประหยัดพลังงาน ด้วยการใช้พลังงานปกติในปัจจุบันแค่เพียง 100 μA ในโหมด Stop หรือที่ต่ำที่สุดเท่าแค่ 1.7 μA ในโหมด Standby

การนำเสนอสถาปัตยกรรมที่มีความยืดหยุ่นสูง ในซีรี่ย์ประกอบด้วย AXI และ multi-AHB bus matrix สำหรับการเชื่อมต่อกับ core อุปกรณ์ต่อพ่วงและหน่วยความจำ การคัดเลือกอุปกรณ์ต่อพ่วงขั้นสูงมีคุณสมบัติต่างๆ เช่น รางพลังงานเฉพาะสำหรับ USB OTG ซึ่งจะช่วยให้การเชื่อมต่อ USB ทำงานต่อเนื่องในขณะที่ส่วนที่เหลือของชิปใช้พลังงานอยู่ที่ 1.8V เพื่อช่วยประหยัดพลังงาน สัญญาณนาฬิกาแบบคู่ช่วยสนับสนุนให้อุปกรณ์ต่อพ่วงต่างๆ ลดความเร็วของ CPU ให้ใช้พลังงานน้อยที่สุด ในขณะที่ยังรักษาความถี่นาฬิกาไม่ให้เปลี่ยนแปลงไปในการสื่อสารกับอุปกรณ์ต่อพ่วง

STM32F746 จะมี Cortex-M7 core พร้อมกับหน่วย floating-point และส่วนขยาย DSP ที่ทำงานได้สูงถึง 216MHz ชุดSTM32F745, MCU มีหน่วยความจำแฟลชในชิปมากถึง 1MB รวมทั้ง RAM 320KB, Ethernet, QSPI และการเชื่อมต่อกล้องถ่ายรูป และตัวควบคุมหน่วยความจำที่มีความยืดหยุ่น(FMC) STM32F746MCUs ขยายการทำงานด้วยตัวควบคุมจอแสดงผล TFT-LCD on-chip, MCU ของ STM32F7 มีพร้อมอยู่ในของแพ็คเกจต่างๆ รวมถึง: 14x14 มม. LQFP100; 28x28 มม. LQFP208; 10x10 มม. 0.65 มม.-pitch UFBGA176; 13x13 มม. 0.8mm-pitch TFBGA216; และ 5.9x4.6 มม. WLCSP143.

2.2.2.2 Arduino UNO Support

Arduino (อ่านว่า : อาร์ดุยโน่) คือเครื่องมือที่จะทำให้คอมพิวเตอร์สามารถรับสัญญาณจากภายนอกและส่งสัญญาณไปควบคุมอุปกรณ์ภายนอกได้อย่างมีประสิทธิภาพมากกว่าใช้เครื่องพีซีตั้งโต๊ะตัวบอร์ดออกแบบจากไมโครคอมพิวเตอร์ชิพเดี่ยวและมีโปรแกรมพัฒนาสำหรับเขียนโปรแกรมให้บอร์ดทำงาน Arduino สามารถประยุกต์ทำเครื่องใช้อัจฉริยะรับสัญญาณจากสวิทซ์หรือเซนเซอร์และควบคุมหลอดไฟมอเตอร์หรืออุปกรณ์อื่นๆ Arduino เป็นได้ทั้งแบบทำงานอิสระหรือทำงานติดต่อกับโปรแกรมที่ทำงานบนเครื่องพีซีหรืออุปกรณ์อิเล็กทรอนิกส์ตัว

บอร์ดสามารถประกอบขึ้นใช้เองหรือจะซื้อสำเร็จที่มีขาย ส่วนโปรแกรมพัฒนา Arduino สามารถดาวน์โหลดได้ฟรี

ไมโครคอนโทรลเลอร์มีตัวเลือกมากมาย เช่น Parallax Basic Stamp, Netmedia's BX-24, Pidgets, MIT's Handyboard, และอีกมากมายที่มีคุณสมบัติใกล้เคียงกันคือการทำงานให้สามารถใช้งานง่ายและเน้นการโปรแกรมไมโครคอนโทรลเลอร์เป็นหลัก Arduino ก็เช่นเดียวกันแต่มีข้อแตกต่างที่เห็นได้ชัดคือ

1.ราคาไม่แพงราคาของ Arduino บอร์ดไม่แพงเมื่อเทียบกับบอร์ดอื่น บอร์ด Arduino ที่ราคาถูกสุดสามารถทำใช้เองได้หรือซื้อสำเร็จ

2.ทำงานได้หลายแพลตฟอร์มโปรแกรมพัฒนา Arduino ทำงานได้ทั้งบนวินโดวส์, Macintosh OSX, และบนลีนุกซ์ในขณะที่บอร์ดอื่นทำงานได้เฉพาะบนวินโดวส์

3.ใช้งานง่ายมีโปรแกรมพัฒนาที่ไม่ซับซ้อน โปรแกรมพัฒนา Arduino ใช้งานง่ายสำหรับมือใหม่และมีความสามารถครบตามความต้องการของนกพัฒนามืออาชีพ

4.เปิดเผยซอร์สโค้ดและนำไปพัฒนาต่อยอดได้ โปรแกรม Arduino ตีพิมพ์แบบเปิดเผยซอร์สโค้ดและสามารถเพิ่มเติมความสามารถผ่าน C++ library ได้

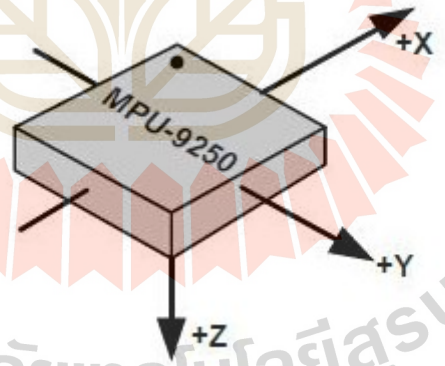
5.เปิดเผยวงจรและสามารถนำไปพัฒนาขยายฮาร์ดแวร์ได้ Arduino ใช้ Microcontroller ของ Atmel เบอร์ ATMEGA8 และ ATMEGA168 วงจรของบอร์ดตีพิมพ์แบบเปิดเผยวงจรภายใต้ Creative Commons License สามารถนำไปดัดแปลงต่อขยายและเพิ่มประสิทธิภาพเพื่อศึกษาการทำงานของมันได้แบบฟรี

บอร์ด Arduino ใช้พัฒนาการใช้งาน Microcontroller ในตระกูล AVR ที่กำลังได้รับความนิยมอย่างสูงทั่วโลกเพราะว่าเป็น Open Source สามารถดัดแปลงไปใช้งานได้ทั้งฮาร์ดแวร์และซอฟต์แวร์ได้ทันที ภาษาที่ใช้กับบอร์ดนี้จะเป็นลักษณะของ C/C+ โดยจัดให้มีไลบรารีต่างๆ ให้พร้อมให้เรียกใช้งานได้ทันทีมากมาย ครอบคลุมการติดต่อกับ I/O ต่างๆ ได้กว้างมากมีไอซี FT232 ที่ใช้สำหรับแปลง USB เป็น Serial ทำให้เมื่อเสียบบอร์ดใช้งานจะมีพอร์ต Serial เพิ่มขึ้นมาอีกพอร์ต ในส่วนของโปรแกรม Arduino มีไลบรารีที่ใช้ในการติดต่อกับอุปกรณ์มากมายเช่น ติดต่อกับ sensor ต่างๆ และสามารถนำมาใช้ได้โดยไม่ต้องเขียนโปรแกรม เพื่อใช้ติดต่อกับอุปกรณ์เหล่านั้นเพียงแต่เรียกใช้ฟังชั่นก็สามารถใช้งานอุปกรณ์เหล่านั้นได้ ภาษาในการเขียนโปรแกรมจะเป็นลักษณะของ C/C+ โดยจัดให้มีไลบรารีต่างๆ ให้พร้อมเรียกใช้งานได้ทันทีมากมายครอบคลุม

การติดต่อกับอินพุตและเอาท์พุตต่างๆ โดยไมโครคอนโทรเลอร์ต้องทำการโหลดข้อมูลผ่าน boot loadder

2.2.2.3 IMU (Inertial Measurement Unit)

IMU (Inertial Measurement Unit) เป็นระบบหนึ่งซึ่งบรรจุในระบบนำทางด้วยแรงเฉื่อย (Inertial Navigation System : INS) มีส่วนประกอบหลักที่สำคัญคือ เครื่องวัดความเร่ง ซึ่งวัดทั้งความเร่งเชิงมุมและความเร่งเชิงเส้น(สำหรับการเปลี่ยนแปลงตำแหน่ง) และไจโรสโคป(เพื่อรักษาให้อยู่ในแนวอ้างอิงที่ถูกต้อง) ซึ่งโดยปกติจะต้องมี Sensor ตรวจวัดอย่างน้อยหนึ่งตัวในแต่ละแนวแกนลักษณะการออกแบบโดยทั่วไป IMU จะมีลักษณะเป็นกล่องภายในบรรจุเครื่องวัดความเร่ง 3 ตัวและ ไจโรสโคปอีก 3 ตัว เครื่องวัดความเร่งจะถูกติดตั้งเข้าไปในแต่ละแนวแกนที่ต้องการวัดโดยที่แต่ละแนวแกนตั้งฉากกันไจโรสโคปอีก 3 ตัว จะถูกติดตั้งในลักษณะตั้งฉากเช่นเดียวกัน ทำการวัดอาการหมุนที่เบี่ยงเบนไปจากแนวอ้างอิงในระบบพิกัดที่กำหนดไว้ จากที่กล่าวมาจะเห็นได้ว่าเซนเซอร์ตรวจวัดหลักๆ ของระบบ IMU เซนเซอร์สำหรับตรวจวัดความเร่งและ เซนเซอร์สำหรับตรวจวัดอาการเอียงในแนวแกนต่างๆ ซึ่งก็คือ เครื่องวัดความเร่ง (Accelerometer) และไจโรสโคป (Gyroscope) โดยในที่นี้จะกล่าวถึงชนิดและหลักการทำงานของอุปกรณ์ทั้งสองต่อไป

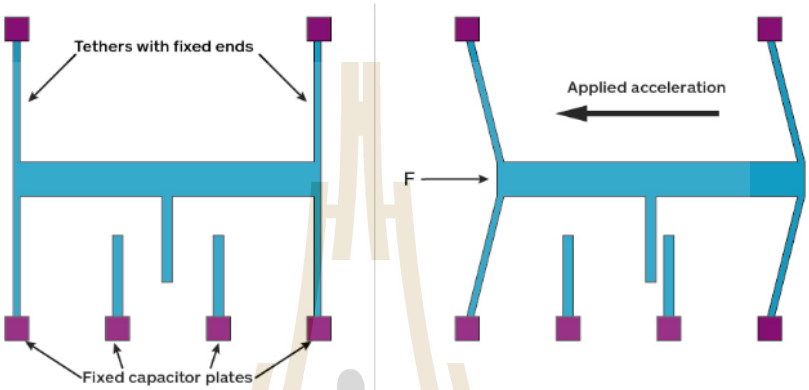


รูปที่ 2.5 เซนเซอร์ IMU (Inertial Measurement Unit)

2.2.2.4 เซนเซอร์วัดความเร่ง(Accelerometer)

ในปัจจุบันอุปกรณ์ Sensor ต่างๆ เริ่มมีการติดตั้งอยู่ในอุปกรณ์ทั่วไป เช่น GPS ในรถยนต์หรือโมดูล GPRS ใช้ทำโทรศัพท์หรือว่าจะเป็น Ultrasonic ที่ใช้ทำ Sensor จับวัตถุที่ท้ายรถยนต์มี Sensor ชนิดหนึ่ง ที่ใช้วัดความเอียงนั้นก็คือ Accelerometer และ Gyroscope หลักการทำงานโดยทั่วไปของ Accelerometer

หากแปลตามตัวอักษรแล้ว Accelerometer มาจาก Acceleration + Meter หรือมิเตอร์ความเร่งตามนิยามก็คือ Sensor วัดความเร่งเพิ่มขึ้นหรือลดลง (ในหน่วย m/s^2) เช่นตัวอย่างความเร่งของแรงโน้มถ่วงก็คือ $9.8m/s^2$ หรือ a (มาจาก Acceleration) นั่นเอง



รูปที่ 2.6 หลักการทำงานของ Accelerometer

โดยหลักการทำงานให้นึกถึงห้องสี่เหลี่ยมเล็กๆ ที่ทุกด้านของกำแพงจะมีสปริงติดอยู่เวลาที่ห้องนี้เอียงไปทางใดทางหนึ่ง สปริงก็จะยุบไปด้านนั้นๆ โดยสมมติว่าแรงดันของสปริงมีน้อยกว่าแรงโน้มถ่วงของโลกและใช้วงจรไฟฟ้าในการดึง Output Analog ออกมาใช้งาน (หรือ Output Digital ซึ่งก็แล้วแต่ตัว Sensor เอง) เราจะใช้ Accelerometer สำหรับเป็นตัวชี้ว่าอยู่ในสถานะ Static(นิ่งเฉย) หรือ Dynamic(เคลื่อนไหวทันทีทันใดหรือหยุดทันทีทันใด) นั่นทำให้Accelerometer เป็น Sensor สำหรับบอกสถานะการเอียงได้เป็นอย่างดี(Tilt Sensor) Sensor ยกตัวอย่างเช่นในโทรศัพท์สมาร์ทโฟนในปัจจุบันก่อนอื่นต้องทำความเข้าใจในเรื่องของอัตราเร่งกันก่อน

อัตราเร่งและการตรวจวัดอัตราเร่ง

อัตราเร่งคือ อัตราการเปลี่ยนแปลงของความเร็วเทียบกับเวลาเครื่องมือที่ใช้วัดอัตราเร่งก็คือมิเตอร์วัดอัตราเร่งหรือแอกเซเลอโรมอเตอร์ (accelerometer) โดยที่สามารถแบ่งลักษณะการตรวจวัดได้ 2 ลักษณะ

1. การตรวจวัดการช็อก(Shock) และการสั่นสะเทือน(Vibration) ซึ่งการช็อกคือ อัตราเร่งขนาดมหาศาลที่เกิดขึ้นในช่วงเวลาสั้นๆ เช่น การสั่นสะเทือนคือ อัตราเร่งขนาดเล็กที่เกิดขึ้นซ้ำกันไปเรื่อยๆ

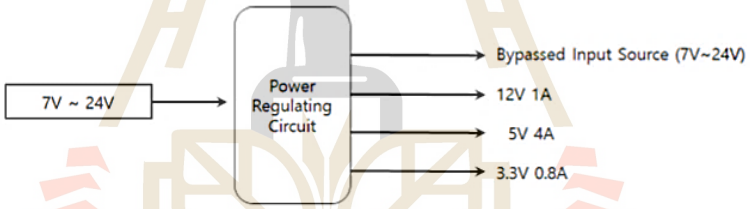
2. การตรวจวัดอัตราเร่งของวัตถุเพื่อนำข้อมูลไปใช้ในการระบุตำแหน่ง ความเร็วและระยะทางที่ได้จากการเคลื่อนที่

2.2.2.5 Various Interfaces

OpenCR เป็นบอร์ดที่supports การสื่อสารด้วยTTL และ RS458 ซึ่งเป็นการอินเทอร์เฟซเริ่มต้นสำหรับ Dynamixel จากภายนอกและบอร์ดยังรองรับ UART, SPI, I2C, CAN interface สำหรับการติดต่อสื่อสารและจุเชื่อมต่อGPIO เพิ่มเติม นอกจากนี้ยังสามารถพัฒนาและแก้ปัญหาเฟิร์มแวร์โดยใช้อุปกรณ์ JTAG9 เช่น STLink หรือ JLink สำหรับนักพัฒนาเนื่องจากรองรับพอร์ต JTAG

2.2.2.6 Power Output

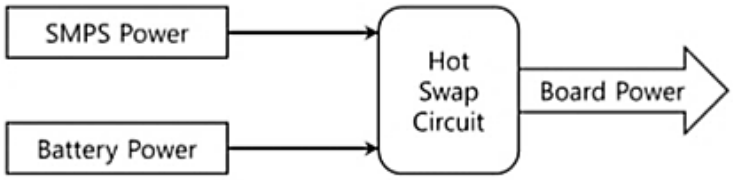
OpenCR ใช้พลังงานจากแหล่งจ่ายไฟ 7V ~ 24V เอาต์พุต 12V (1A), 5V (4A) และ 3.3V (800mA) และยังสามารถใช้เป็นแหล่งพลังงานของ SBC และเซ็นเซอร์ เช่น Raspberry Pi, กล้อง USB เนื่องจากรองรับได้ถึง 5V / 4A



รูปที่ 2.7 Power output Diagram

2.2.2.7 Power Hot Swap

เมื่อ SMPS (Switched-Mode Power Supply หรือที่เรียกว่าอุปกรณ์แปลงกระแสไฟฟ้ากระแสสลับไปเป็น DC) แหล่งสัญญาณจะเชื่อมต่อกับบอร์ด OpenCR ทำการสวิตช์แหล่งจ่ายไฟจะเปลี่ยนจากแบตเตอรี่เป็น SMPS โดยอัตโนมัติ ในทำนองเดียวกันถ้าแบตเตอรี่เชื่อมต่อขณะใช้ SMPS และ SMPS ถูกตัดการเชื่อมต่อบอร์ดจะทำงานโดยใช้พลังงานจากแบตเตอรี่อาจได้รับการแปลงสัญญาณหรือ SMPS ที่ไม่มีในขณะที่กำลังทำงานเมื่อเปิดเครื่อง



รูปที่ 2.8 Hot-swap configuration

2.2.2.8 โมดูลมอเตอร์

โมดูลมอเตอร์ของบอร์ด OpenCR สามารถรับข้อมูลเชิงพื้นฐานได้อย่างแม่นยำ โดยการใช้ 2 DYNAMIXELs ของล้อทั้งสองข้างรวมกัน มอเตอร์ DYNAMIXEL XM series สามารถทำงานได้ตั้งแต่ 1 ถึง 6 โหมดของโหมดการทำงาน เช่น โหมดควบคุมความเร็วล้อ, โหมดควบคุมตำแหน่ง เป็นต้น DYNAMIXEL สามารถใช้ในการทำหุ่นยนต์ที่มีน้ำหนักเบา สามารถควบคุมได้อย่างแม่นยำ โดยการควบคุมความเร็ว แรงบิด และการควบคุมตำแหน่ง DYNAMIXEL เป็นต้น ซึ่งเป็นส่วนประกอบหลักในการทำให้หุ่นยนต์สามารถทำงานได้อย่างสมบูรณ์แบบ การตั้งค่าและการดูแลรักษาง่าย

2.2.2.9 การควบคุมบอร์ดจาก ROS

การควบคุมบอร์ดจะมี Open-sourced ของฮาร์ดแวร์และซอฟแวร์จาก ROS ซอฟแวร์ในการควบคุมบอร์ด OpenCR ไม่ได้มีหน้าที่เพียงเพื่อการควบคุมการเคลื่อนที่ แต่ยังสามารถนำมาเพื่อการควบคุมเซนเซอร์ของหุ่นยนต์เพื่อให้มีประสิทธิภาพมากขึ้น เช่น เซ็นเซอร์สัมผัส, เซ็นเซอร์สี, เซ็นเซอร์แสง เป็นต้น โดยบอร์ดมีขาต่อไฟเลี้ยง 3.3V, 5V, 12V เพื่อรองรับอุปกรณ์ในหลากหลายรูปแบบ

2.2.2.10 Source code

ฮาร์ดแวร์ เฟิร์มแวร์และซอฟแวร์ของหุ่นยนต์สามารถดาวน์โหลดและพัฒนาได้จากเว็บไซต์และส่วนประกอบทั้งหมดของหุ่นยนต์ทำมาจากพลาสติก

- https://github.com/ROBOTIS-GIT/OpenCR
- https://github.com/ROBOTIS-GIT/OpenCR-Hardware

2.3 มอเตอร์ DYNAMIXEL

DYNAMIXEL เป็นดิจิตอลเซอร์โวมอเตอร์สำหรับหุ่นยนต์โดยเฉพาะ ซึ่งประกอบด้วยมอเตอร์กระแสตรงชุดเฟืองมอเตอร์ มีไดรเวอร์คอนโทรลเลอร์ที่สามารถติดต่อสั่งงานแบบเครือข่ายได้อีกด้วย ซีรีส์MX เป็นแนวคิดใหม่ของ DYNAMIXEL ที่มีฟังก์ชั่นขั้นสูงเช่น การควบคุมที่แม่นยำการควบคุมแบบPID ควบคุมตำแหน่งใน Actuator Mode ได้ 360 องศา อีกทั้งยังสามารถสื่อสารด้วยความเร็วขั้นสูง

คุณลักษณะเด่น

- มีความทนทานต่อการควบคุมตำแหน่งและองศาที่แม่นยำ
- การควบคุมตำแหน่งด้วย Absolute Encoder
- ควบคุมตำแหน่งใน Actuator Mode ได้ 360 องศาโดยไม่มีจุดบอด
- ควบคุมตำแหน่งได้ละเอียดถึง 4,096 ตำแหน่ง (ค่าความคลาดเคลื่อน 0.088 องศา)
- สามารถควบคุมความเร็วได้ในโหมด Endless Turn
- มีความแม่นยำในการควบคุมตำแหน่ง ด้วยการควบคุมแบบ PID
- ความเร็วในการรับส่งข้อมูลสูดสุด 4.5 Mbps
- สื่อสารด้วย : TTL Multi Drop Bus (daisy chain type connector)

ข้อควรระวัง

- MX-28T รองรับการสื่อสารแบบ TTL Multi Drop Bus (daisy chain type connector)
- แรงดันไฟฟ้าที่แนะนำของ MX-28 จะอยู่ที่ 10~14.8V (แนะนำที่แรงดัน 12 V) ซึ่งจะแตกต่างกับของ RX-28 แรงดันไฟฟ้าในการใช้งาน 12~18.5V

ตารางที่ 2.1 องค์ประกอบของ DYNAMIXEL

Description		Qty
DYNAMIXEL	MX-28T	1
HORN	RX28 HN07-N101	1
WASHER	Thrust Washer	1
CABLE	3P Cable 200mm	1
BOLT/NUT	Wrench Bolt M2.5*6	16
	Wrench Bolt M2.5*8	1
	Wrench Bolt M2*3	10
	Nut M2.5	18

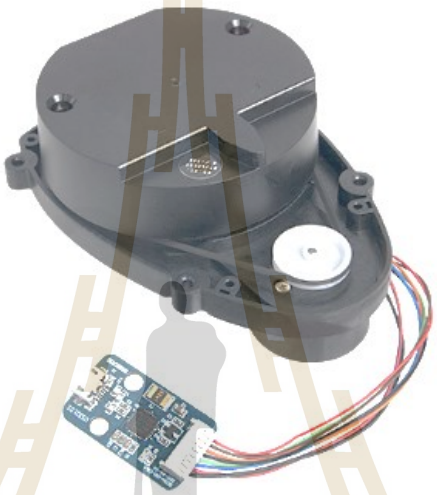
คุณสมบัติ

- น้ำหนัก : 72 กรัม
- ขนาด : 35.6มม. * 50.6มม. * 35.5มม.
- ความละเอียด : 0.088°x 4,096
- อัตราทดเกียร์ : 193 : 1
- แรงบิด : 3.1 นิวตันเมตร (ที่แรงดัน 14.8 โวลท์, 1.7 แอมป์)
- ความเร็วรอบสูงสุดขณะไม่มีโหลด : 67 รอบต่อนาที (Wheel Mode ,ที่แรงดัน 14.8 โวลท์)
- รูปแบบการทำงาน
    - Actuator Mode : 0° ~ 360°
    - Wheel Mode : Endless Turn
- อุณหภูมิการใช้งาน: -5°C ~ +80°C
- แรงดันไฟฟ้าที่ใช้ : 10 ~ 14.8 โวลท์ (แรงดันไฟฟ้าที่แนะนำ 12 โวลท์)
- สัญญาณข้อมูลแบบดิจิตอลแพ็กเกจ
- รูปแบบโปรโตคอล Half duplex Asynchronous Serial Communication (8bit,1stop,No Parity)
- รูปแบบการติดต่อสื่อสาร : TTL Multi Drop Bus (daisy chain type connector)
- ID : 254 ID (0~253)
- ความเร็วในการสื่อสาร : 8,000 bps ~ 4.5 Mbps
- การป้อนกลับ : ตำแหน่ง, อุณหภูมิ, แรงโหลด, แรงดันไฟฟ้าอินพุต ฯลฯ
- วัสดุที่ใช้ผลิตเฟือง : โลหะ
- เซ็นเซอร์ที่ใช้วัดตำแหน่ง : Absolute Encoder

รูปที่ 2.9 มอเตอร์ DYNAMIXEL

2.4 เซนเซอร์ Lidar (Light Detection and Ranging)

เป็นเซนเซอร์ที่มีความจำเป็นอย่างยิ่งสำหรับการใช้งาน ROS SLAM การทำงานของ Lidar คือการใช้แสงเลเซอร์ในการวัดระยะทางของสิ่งกีดขวางโดยรอบ 360 องศา มีทั้งแบบ 2 มิติและ 3 มิติ ซึ่งจะมี DC Motor ทำให้เลเซอร์หมุนกวาดไปโดยรอบแล้วทำการอ่านค่าระยะทางที่ได้ส่งไปทางพอร์ท USBภาพที่ 2.10 Lidar



รูปที่ 2.10 เซนเซอร์ Lidar

อุปกรณ์วัดระยะทางด้วยเลเซอร์ (LRF: Laser Range Finder, LADAR: Laser Detection and Ranging, LIDAR: Light Detection and Ranging) เป็นเทคโนโลยีการวัดระยะจากตัวอุปกรณ์ถึงวัตถุด้วยการปล่อยลำแสงเลเซอร์ขนาดเรียวเล็กออกไปเป็นจังหวะสั้น ๆ และวัดพลังงานที่วัตถุสะท้อนกลับมา อุปกรณ์วัดระยะทางด้วยเลเซอร์ถูกใช้งานอย่างแพร่หลาย เช่น การวัดและสำรวจ การสร้างโมเดลสามมิติ ฯลฯ อุปกรณ์วัดระยะทางด้วยเลเซอร์แต่ละรุ่นจะใช้แสงเลเซอร์ที่ย่านความถี่ต่าง ๆ กัน ส่วนใหญ่จะอยู่ในช่วงของแสงอินฟราเรด (600-1000 นาโนเมตร) วัตถุที่สามารถวัดได้มีความหลากหลายสูง เช่น เสื้อผ้า ผิวหนังโต๊ะไม้ ก้อนหิน พื้นกระเบื้อง แผ่นโฟม กล่องพลาสติก เป็นต้น การคำนวณระยะของวัตถุแบ่งออกเป็นสองวิธีหลัก คือ วิธีที่หนึ่งแบบ “Incoherent” ซึ่งวัดระยะทางจากปริมาณพลังงานที่สะท้อนกลับมาและวิธีที่สองแบบ “Coherent” ซึ่งวัดความเหลื่อมล้ำของคาบซึ่งจะถูกรบกวนได้ง่ายกว่าแบบแรก วิธีการวัดทั้งสองวิธีจะต้องอาศัยการปล่อยลำแสงเลเซอร์ออกไปซึ่งแบ่งเป็นลำแสงพลังงานต่ำ และลำแสงพลังงานสูง สำหรับลำแสงพลังงานต่ำนั้นจะใช้พลังงานน้อยมาก (ระดับหนึ่งไมโครจูล) และส่วนใหญ่จะปลอดภัยต่อ

สายตา ส่วนลำแสงพลังงานสูงมันจะถูกใช้ในงานวิจัยด้านชั้นบรรยากาศ อุปกรณ์วัดระยะทางด้วยเลเซอร์ประกอบไปด้วยส่วนประกอบหลัก 3 ส่วน คือ

1.แสงเลเซอร์มีช่วงความยาวคลื่น 600-1000 นาโนเมตร อุปกรณ์ที่ปล่อยลำแสงเลเซอร์ในช่วงความยาวคลื่นนี้มีราคาถูกแต่จะส่งผลกระทบต่อสายตาทำให้ต้องจำกัดปริมาณพลังงานที่เลเซอร์ปล่อยออกมา ในบางงานเลเซอร์ที่ใช้จะมีความยาวคลื่นเกินกว่า 1000 นาโนเมตรซึ่งปลอดภัยต่อสายตาที่ระดับพลังงานที่สูงกว่าเนื่องจากตาคนไม่รับแสงที่ความยาวคลื่นนี้ แต่อุปกรณ์ที่จะตรวจวัดคลื่นที่ความยาวคลื่นนี้ยังมีประสิทธิภาพไม่สูงนักทำให้มีความแม่นยำต่ำ

2.อุปกรณ์ปล่อยลำแสงเลเซอร์ซึ่งประกอบไปด้วยกระจกตัวหมุนซึ่งการหมุนของกระจกทำให้แนวการตรวจวัดของเลเซอร์เปลี่ยนทิศไปเรื่อย ๆ และเรียงตัวกันเป็นระนาบ การเลือกอุปกรณ์ปล่อยลำแสงและฐานหมุนนี้จะส่งผลต่อความละเอียดของข้อมูลที่อุปกรณ์รับรู้จะตรวจวัดได้

3.อุปกรณ์รับแสงโดยความเร็วและความละเอียดในการตรวจวัดแสงจะส่งผลต่อคุณภาพโดยรวมของอุปกรณ์วัดระยะทางด้วยเลเซอร์

อุปกรณ์วัดระยะทางด้วยเลเซอร์ Lidar ทำงานแบบ step-by-step สามารถออกแบบให้ทำงานได้ 360 องศา เป็นอุปกรณ์วัดระยะทางด้วยเลเซอร์สำหรับการทำแผนที่กลางแจ้งหรือในห้องซึ่งทำงานแบบ real-time และสามารถติดตามตำแหน่งบนแผนที่นั้นๆ ได้ เรียกว่า 'SLAM' (simultaneous localization and mapping)

คุณสมบัติ Lidar

- ระยะที่สามารถวัดได้ 0-40 เมตร
- ใช้แรงดันขนาด 4.75-5 โวลต์
- มีการส่งออกค่าเป็น PWM หรือ I2C
- ใช้กระแสไฟฟ้า 105-130 mA
- อัลตร้าโซนิคใช้ความถี่ 1-500 kHz

Lidarคือการส่งแสงเลเซอร์ออกไปและจับเวลาที่แสงสะท้อนกลับมา เพื่อนำมาหาระยะทางของสิ่งกีดขวางตามสมการ

$$C = s/t \tag{2.1}$$

$$S = c/t \tag{2.2}$$

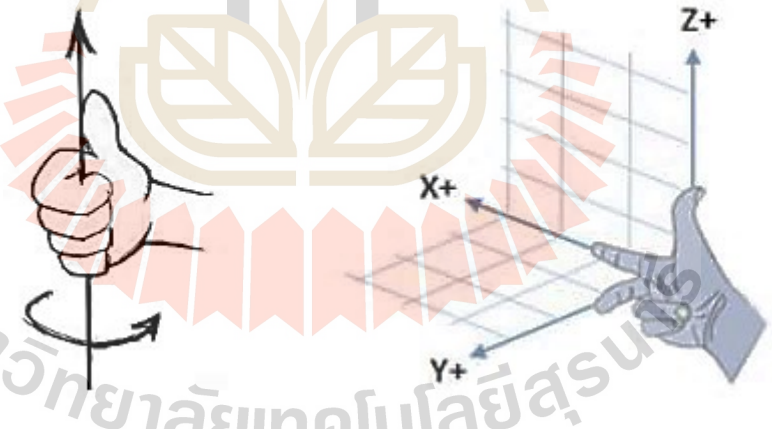
$$s = s/2 = c.t \tag{2.3}$$

C คือความเร็วของแสง เมื่อทราบระยะทางก็จะสามารถหาระยะทางได้ หาร 2 เนื่องจากการเดินทางของแสงไปและกลับ

sคือระยะทางของสิ่งกีดขวางเทียบกับเซนเซอร์ Lidar

2.5 ระบบพิกัด

ระบบพิกัด (coordinate) เป็นหลักการที่สำคัญสำหรับหุ่นยนต์ โดยหลักการของระบบพิกัดจะเกี่ยวข้องกับเรื่องของ Transform Configuration (TF) สำหรับการใช้ ROS ซึ่งความหมายของระบบพิกัดคือ ระบบแกน เช่น X,Y,Zที่สามารถระบุตำแหน่งของวัตถุที่ต้องการอ้างอิงถึงได้ในระบบพิกัดนั้นๆ ในปัจจุบันระบบ ROS นิยมใช้แกน X,Y,Z เป็นระบบพิกัดของการระบุตำแหน่งต่างๆ ดังรูปที่ 2.11 ซึ่งเป็นหลักการเดียวกับวิชาคณิตศาสตร์ คือ จุดต่างๆ ในระบบ 3 มิติ สามารถอ้างถึงได้จาก (X,Y,Z) ซึ่งใช้ระบบของกฎมือขวาดังรูปที่ 2.11



รูปที่ 2.11 ระบบพิกัด ROS

2.6 การเคลื่อนที่ของหุ่นยนต์สำรวจ

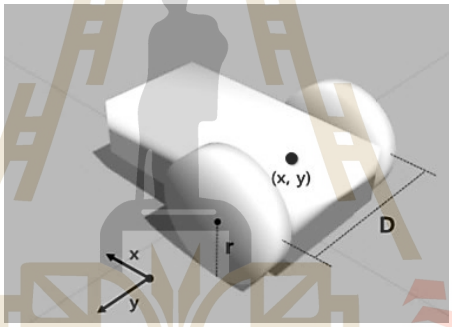
การเคลื่อนที่ของหุ่นยนต์โดยหลักแล้วจะพิจารณาออกแบบตามวัตถุประสงค์ การใช้และสภาพการทำงานของหุ่นยนต์เป็นสำคัญหากหุ่นยนต์นั้นถูกใช้ในโรงงานอุตสาหกรรม ซึ่งงานส่วนใหญ่จะเป็นงานที่ทำในขอบเขตจำกัด การเคลื่อนที่ของหุ่นยนต์จึงไม่มีความจำเป็น ดังนั้นหุ่นยนต์

จึงถูกออกแบบให้มีลักษณะเป็นแขนกลชนิดติดตั้งอยู่กับที่ แต่หากการทำงานเป็นไปในเชิงสำรวจตรวจการณ์หรืองานที่มีขอบเขตการทำงานที่กว้าง จำเป็นที่หุ่นยนต์ต้องสามารถเคลื่อนที่ไปอยู่ในจุดต่างๆได้ หุ่นยนต์จะถูกออกแบบให้สามารถเคลื่อนที่ได้ซึ่งการเคลื่อนที่ของหุ่นยนต์จะมีอยู่ 2 แบบดังต่อไปนี้

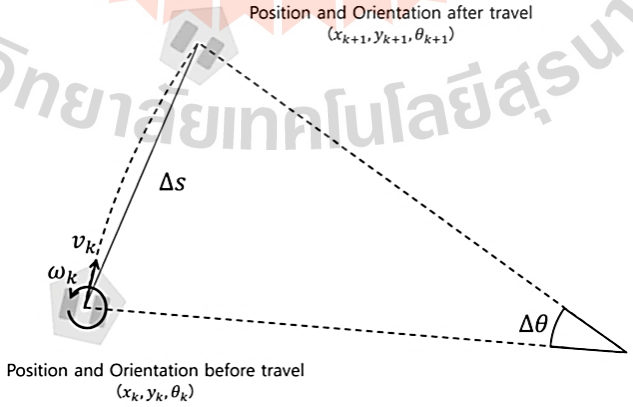
• การเคลื่อนที่โลโคโมชั่น (locomotion) หมายถึงเป็นการกระทำด้วยกำลังเพื่อให้เกิดการเคลื่อนที่จากที่หนึ่งไปอีกที่หนึ่ง

• ความสามารถในการเคลื่อนที่โมบิลิตี้ (mobility) หมายถึงความสามารถของระบบขับเคลื่อนที่จะนำพาหุ่นยนต์ให้เคลื่อนที่ไปในพื้นผิวและสิ่งกีดขวางต่างๆ

หลักการในการเคลื่อนที่ของหุ่นยนต์สำรวจ ดังรูปที่ 2.12



รูปที่ 2.12 หลักการในการเคลื่อนที่ของหุ่นยนต์สำรวจ



รูปที่ 2.13 ตำแหน่งและการเคลื่อนที่ของหุ่นยนต์สำรวจ

จากรูปที่ 2.13 เมื่อมีหุ่นยนต์เคลื่อนที่ กำหนดให้ D เป็นระยะห่างระหว่างล้อกับรัศมีของล้อ สมมติว่าหุ่นยนต์เกิดการเคลื่อนที่ในระยะเวลาสั้น ๆ ในช่วงเวลา T_e กำหนดให้ความเร็วในการหมุน (V_l, V_r) ของล้อซ้ายและขวาจะสามารถหาได้จากสมการที่ (4) และ (5) พร้อมกับจำนวนรอบมอเตอร์ซ้ายและขวา (จุดเริ่มต้นของการเคลื่อนที่ของหุ่นยนต์ E_lc, E_rc และจุดที่หุ่นยนต์เคลื่อนที่ผ่าน E_lp, E_rp ดังสมการต่อไปนี้

$$V_l = (\frac{E_lc - E_lp}{T_e}).\frac{\pi}{180} \ (\text{radian/sec}) \tag{2.4}$$

$$V_r = (\frac{E_rc - E_rp}{T_e}).\frac{\pi}{180} \ (\text{radian/sec}) \tag{2.5}$$

จากสมการ 6 และ 7 สามารถคำนวณหาความเร็วของล้อซ้ายและขวา (V_l, V_r) จากความเร็วของล้อซ้ายและขวาจะเป็นความเร็วเชิงเส้น (V_k) และความเร็วเชิงมุม (W_k) ของหุ่นยนต์สามารถหาได้ดังแสดงในสมการที่ 8 และสมการที่ 9

$$V_l = V_l \cdot r \ (\text{meter/sec}) \tag{2.6}$$

$$V_r = V_r \cdot r \ (\text{meter/sec}) \tag{2.7}$$

$$V_k = (V_r + V_l)/2 \ (\text{meter/sec}) \tag{2.8}$$

$$W_k = (V_r + V_l)/D \ (\text{meter/sec}) \tag{2.9}$$

จากที่ได้กล่าวมาสามารถหาตำแหน่งในการเคลื่อนที่ได้จาก ($X_{(k+1)}, Y_{(k+1)}$) และมุมที่เกิดจากการเคลื่อนที่ของหุ่นยนต์คือ ($\theta_{(k+1)}$) ดังนั้นสมการในการเคลื่อนที่ของหุ่นยนต์สามารถหาได้จากสมการที่ 10 ไปจนถึงสมการที่ที่ 12

$$\Delta S = V_k T_e \ \Delta\theta = W_k T_e \tag{2.10}$$

$$X_{(k+1)} = X_k + \Delta S \cdot \cos(\theta_k + \frac{\Delta\theta}{2}) \tag{2.11}$$

$$Y_{(k+1)} = Y_k + \Delta S \cdot \sin(\theta_k + \frac{\Delta\theta}{2}) \quad (2.12)$$

$$\theta_{(k+1)} = \theta_k + \Delta\theta \quad (2.13)$$

2.7 Odometry

Odometry คือ การวัดการเคลื่อนที่ของหุ่นยนต์ในระบบพิกัดเฉพาะของตัวหุ่นยนต์เอง ระบบ ROS ซึ่งจะทำการเก็บข้อมูลการเคลื่อนที่ของหุ่นยนต์ดังนี้

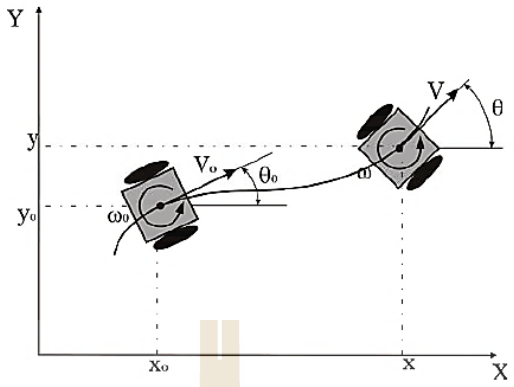
- ตำแหน่งของหุ่นยนต์ในพิกัด (Odometryframe) ได้แก่ตำแหน่ง X,Y,Z ของหุ่นยนต์ที่ได้เคลื่อนที่ไปตั้งแต่จุดเริ่มต้น (0,0,0)

- การวางมุมของหุนยนต์เทียบกับแกน Z (Orientation) ได้แก่ มุมที่หุ่นยนต์กระทำเทียบกับแกน X ของระบบพิกัดของหุ่นยนต์

- ความเร็วเชิงเส้นของหุ่นยนต์ ได้แก่ ความเร็วในแนวแกน X,Y,Z ตามลำดับ มีหน่วยเป็นเมตรต่อวินาที ความเร็วมีค่าเป็นบวก หมายความว่าหุ่นยนต์เคลื่อนที่ไปทางแกนฝั่งบวก ตรงกันข้ามความเร็วมีค่าเป็นลบเมื่อหุ่นยนต์เคลื่อนที่ไปทางแกนฝั่งลบ

- ความเร็วเชิงมุมของหุ่นยนต์ ได้แก่ความเร็วเชิงมุมโดยเทียบกับแกน Z ของขนาดของหุ่นยนต์จะมีค่าเป็นบวกเมื่อหุ่นยนต์หมุนไปทิศทวนเข็มนาฬิกาและตรงกันข้ามจะเป็นลบเมื่อหุ่นยนต์หมุนไปในทิศตามเข็มนาฬิกา

รูปที่ 2.14อธิบายหลักการของ Odometry ซึ่งการทำงานของหุ่นยนต์จะเคลื่อนที่ไปตามแกน X,Y ด้วยความเร็วเชิงเส้นและความเร็วเชิงมุมที่แตกต่างกันไปในแต่ละเวลา ซึ่งระบบ Odometry จะเก็บข้อมูลเหล่านี้ไว้ทั้งหมด



รูปที่ 2.14 อธิบายหลักการของ Odometry

เมื่อหุ่นยนต์เคลื่อนที่ไปในตำแหน่งต่างๆ จำเป็นที่จะต้องรู้ตำแหน่งที่หุ่นยนต์เคลื่อนที่ไป ซึ่งระบบการรู้ตำแหน่งมีหลายวิธีแต่วิธีของ Odometry เป็นวิธีการที่เป็นพื้นฐานที่สุดได้แก่ การคำนวณจากการหมุนไปของมอเตอร์ ซึ่งได้มาจาก Encoder ของมอเตอร์ สามารถเขียนเป็นสมการได้ดังนี้

ระยะทาง = (2 x Pi x รัศมีของล้อ) / จำนวนพัลส์ในการหมุน 1 รอบ (2.14)

สำหรับการคำนวณความเร็วของมอเตอร์จะใช้การนับพัลส์จากนั้นนำมาคำนวณหาระยะทางและหารด้วยเวลาที่ใช้ในการนับพัลส์

ความเร็ว = ระยะ / เวลา (2.15)

2.8 โปรแกรม rviz

โปรแกรม rviz นั้นเป็นโปรแกรมที่มาพร้อมกับการติดตั้ง ROS มีประโยชน์มากสำหรับการดูค่า Topic ต่างๆ เช่น ข้อมูลจาก Lidar, ค่า frame ต่างๆ แผนที่ที่สร้างขึ้นมา จึงต้องมีการแสดงรายละเอียดต่างๆ โดยมีหลักการใช้งานดังต่อไปนี้

2.8.1 การ Add ส่วนของ Topic เข้าไป

สามารถใช้ปุ่ม Add แล้วเลือกประเภทของข้อมูลที่จะ Add เข้าไป เช่น Topic ชื่อว่า scan จาก Lidarก็สามารถ Add เข้าไปได้

2.8.2 ขนาดของ Cell size ใน Grid ควรปรับให้เหมาะสมกับพื้นที่จริง

ตัวอย่างเช่น 50 ซ.ม. ตอนกำหนดจะเป็น 0.5 ม. สำหรับงานภายในห้องควรจะมีการปรับไว้ที่ 0.5 ม. แต่หากเป็นงานที่มีขนาดพื้นที่ใหญ่ขึ้นมีการสร้างแผนที่ขนาดใหญ่ก็อาจจะปรับขนาดให้เล็กลง เป็นต้น

2.8.3 Topic ไหนไม่ใช้ไม่ควร Add เข้ามา

เนื่องจากเป็นการเพิ่มภาระให้การทำงานกับคอมพิวเตอร์โดยไม่จำเป็น

2.8.4 การกำหนด Fix frame มีความสำคัญ

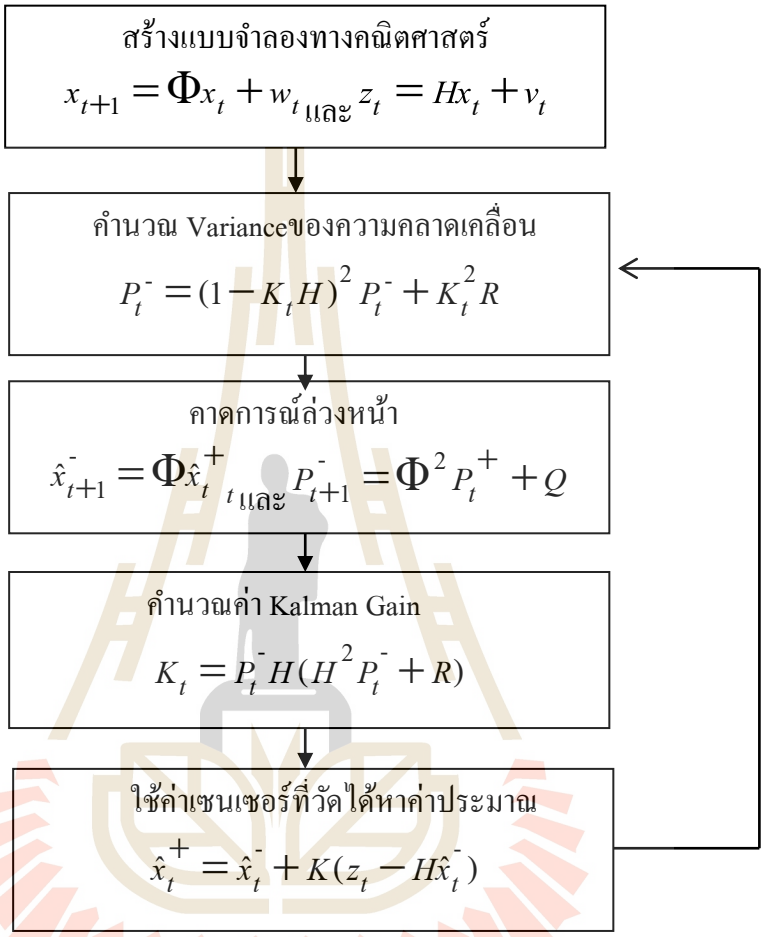
Fix frame คือ frame ที่จะอยู่นิ่งตรงกึ่งกลางของจอภาพการใช้งานขึ้นอยู่กับความต้องการ เช่น ต้องการดูการเคลื่อนที่ว่าหุ่นยนต์สามารถเคลื่อนที่ไปไกลจากจุดเริ่มต้นเท่าไร ก็ควรใช้ Fix frame เป็น map แต่ถ้าต้องการดูว่าหุ่นยนต์อยู่ตำแหน่งไหนของแผนที่ก็ควรกำหนด Fix frame เป็น base_link เป็นต้น

รูปที่ 2.15 ตัวอย่างโปรแกรม rviz

2.9 เทคนิค Kalman Filter

ตัวกรองคาลมานเป็นแบบจำลองทางคณิตศาสตร์ซึ่งถูกตั้งชื่อตาม รูดอล์ฟ อี คาลมาน (Rudolf E.Kalman) มีจุดประสงค์เพื่อใช้ในการประมาณค่าจากการวัดที่ถูกสังเกตในการประมาณค่าจากช่วงระยะเวลาที่มีสัญญาณรบกวนในรูปแบบสุ่ม และมีค่าความไม่แม่นยำต่าง ๆ และค่าที่ผลิตได้จากการกรองแบบคาลมานจะมีความใกล้เคียงค่าที่แท้จริงมากกว่าค่าที่ได้จากการวัดและค่าที่คำนวณเกี่ยวข้องต่าง ๆ ของการวัด ตัวกรองคาลมานมีการนำไปประยุกต์ใช้ได้หลายรูปแบบ เช่น การลดสัญญาณรบกวน และอื่น ๆ เทคนิคการกรองสัญญาณแบบคาลมานแบบสามารถอธิบายได้

ด้วยแผนภาพเชื่อมโยงดังในรูปที่ 2.16เทคนิค Kaman Filter จะประกอบด้วยสองส่วนหลัก คือ การคาดการณ์และการแก้ไขค่า



รูปที่ 2.16 ขั้นตอนของการประมวลผลด้วย Kalman Filter ที่เวลา t ใด ๆ

2.9.1 การคาดการณ์

การคาดการณ์นั้นเป็นขั้นตอนแรกของเทคนิค Kalman Filter ตัวแปรสถานะจากการคาดการณ์(Predicted State) หรือที่เรียกว่าตัวแปรสถานะเบื้องต้น (Priori State) ถูกคำนวณโดยไม่ได้สนใจสัญญาณรบกวนทางพลศาสตร์และแก้สมการเชิงอนุพันธ์ที่อธิบายแบบจำลองทางพลศาสตร์ของระบบ

$$\dot{x}^-(t) = F \cdot x^-(t) \tag{2.16}$$

เวกเตอร์ตัวแปรสถานะที่เวลา tสามารแสดงโดยอนุกรม Taylor เมื่อเทียบกับตัวแปรสถานะ $\dot{x}^{-}(t_0)$

$$x^{-}(t)=x^{-}(t_0)+\dot{x}^{-}(t_0)(t-t_0)+\frac{1}{2}\ddot{x}^{-}(t_0)(t-t_0)^2+.....\tag{2.17}$$

โดยใช้สมการที่ 17นี้ แก้สมการด้านบนสามารถเขียนได้ดังนี้

$$x^{-}(t)=(t_0)+\dot{x}^{-}(t_0)+F\cdot x^{-}(t_0)(t-t_0)+\frac{1}{2}F\cdot x^{-}(t_0)(t-t_0)^2+.....\tag{2.18}$$

ดังนั้นผลลัพธ์ $x^{-}(t)$ ของสมการเชิงอนุพันธ์หรืออีกนัยหนึ่งก็คือตัวแปรสถานะจากการคาดการณ์ที่แท้จริงเป็นการรวมกันแบบเชิงเส้นของตัวแปรสถานะเริ่มต้นที่ $\dot{x}^{-}(t_0)$

$$x^{-}(t)=\Phi_0^t\cdot x^{-}(t_0)\tag{2.19}$$

Φ_0^t ถูกเรียกว่าเมตริกซ์การถ่ายโอนตัวแปรสถานะ ซึ่งแปลงตัวแปลสถานะใด ๆ $x(t)$ ไปเป็นตัวแปรสถานะที่สอดคล้องกัน $x(t)$ ที่เวลา t

จากสมการที่ 19และ 20จะได้

$$x^{-}(t)=F\cdot x^{-}(\mathrm{t})=F\cdot\Phi_0^t\cdot x^{-}(t)\tag{2.20}$$

และการใช้ในสมการที่ 17อีกครั้งก็จะได้

$$\dot{x}^{-}(t)=\frac{d}{dt}x^{-}(t)=\frac{d}{dt}[\Phi_0^t\cdot x^{-}(t_0)]=\left[\frac{d}{dt}\Phi_0^t\right]\cdot x^{-}(t)\tag{2.21}$$

โดยเปรียบเทียบสมการที่ 19และ 20จะได้

$$\frac{d}{dt}\Phi_0^t=F\cdot\Phi\neq\Phi_0^t\tag{2.22}$$

โดยใช้เมตริกซ์เริ่มต้นที่ $\Phi_0^t=I$ ได้มาเนื่องจาก $x(t)=I\cdot x(t_0)$

และเมตริกซ์แปรปรวน (Covariance Matrix) $P^{-}(t_i)$ ของเวกเตอร์ตัวแปรสถานะจากการคาดการณ์จะได้ตามกฎของค่าการแพร่กระจายของค่าความผิดพลาด

$$P^{-}(t_i) = \Phi_{t_{i-1}}^{t_i} \cdot P(t_i - 1) \cdot \left(\Phi_{t_{i-1}}^{t_i}\right)^T + Q \tag{2.23}$$

ในรูปแบบทั่วไปแล้วเมตริกซ์ความแปรปรวนของสัญญาณรบกวน σ เป็นฟังก์ชั่นของเวลา ดังนั้นเมตริกซ์ความแปรปรวนจะได้ดังนี้

$$P^{-}(t_i) = \Phi_{t_{i-1}}^{t_i} \cdot P(t_i - 1) \cdot \left(\Phi_{t_{i-1}}^{t_i}\right)^T + \int_{t_i}^{t_1} Q(t)\,dt \tag{2.24}$$

2.9.2 การแก้ไขค่า

ในขั้นตอนการแก้ไขค่าเวกเตอร์ตัวแปรสถานะจากการคาดการณ์ $x^{-}(t_i)$ ถูกปรับปรุงด้วยค่าสังเกตการณ์ที่วัดได้ในช่วงเวลา t ดังนั้นตัวแปรสถานะหลังการแก้ไข (Posterior State) มีรูปแบบเป็น

$$x^{+}(t_i) = x^{-}(t_i) \tag{2.25}$$

โดยใช้เมตริกซ์ความแปรปรวน

$$P^{+}(t_i) = P^{-}(t_i) + \Delta P(t_i) \tag{2.26}$$

เหมือนที่กล่าวไปแล้ว Kalman Filter เป็นตัวกรองสัญญาณที่เหมาะสมที่สุดนี้หมายความว่า การแปรปรวนของตัวแปรสถานะในเมตริกซ์ความแปรปรวนของตัวแปรสถานะ P^{+} จะมีค่าที่น้อยที่สุดขณะที่ P^{-} ได้คำนวณมาจากขั้นตอนจากการคาดการณ์จะได้ ΔP ที่เป็นค่าน้อยที่สุด

$$\Delta P(t_i) = E[\Delta x(t_i) \Delta x(t_i)^T] \tag{2.27}$$

ซึ่งเงื่อนไขนี้จะสอดคล้องกับ

$$\Delta x(t_i) = P^{-} H^T (HP - H^T + R(t_i))^{-1} (l(t_i) - Hx^{-}(t_i))$$

$$\Rightarrow \Delta x(t_i) = K(t_i)(l(t_i) - l^{-}(t_i)) \tag{2.28}$$

$$\text{และ } K(t) = P^{-} H^{T} (HP - H^{T} + R(t_i))^{-1} \tag{2.29}$$

K ถูกเรียกเป็นเมตริกซ์อัตราขยาย (Gain Matrix) ค่าความแตกต่างของ $l(t_i) - l^{-}(t_i)$ นั้น ถูกเรียกว่าค่าที่เหลือจากการวัด (Measurement Residual) ซึ่งมันแสดงให้เห็นถึงความแตกต่างระหว่างการวัดที่ได้จากการคาดการณ์ $l(t_i) - Hx^{-}(t_i)$ และการวัดที่ได้จริง $l(t_i)$ สุดท้ายแล้วตัวแปรสถานะที่ถูกแก้ไขจะได้จาการคำนวณดังนี้

$$x^{+}(t_i) = x^{-}(t_i) + K(t_i) \cdot (l(t_i) - l^{-}(t_i)) \tag{2.30}$$

ในสมการนี้ตัวแปรสถานะที่ถูกประมาณค่าและค่าที่วัดได้นั้นจะถูกแบ่งความสำคัญตามน้ำหนักแล้วนำมารวมกันเพื่อคำนวณตัวแปรสถานะที่ถูกแก้ไขค่าซึ่งหมายความว่า ถ้าค่าความแปรปรวนจากการวัดนั้นมีค่าน้อยกว่าค่าความแปรปรวนของตัวแปรสถานะที่ได้จากการณคาดการณ์น้ำหนักของการวัดจะสูงกว่าและตัวแปรสถานะที่ได้จากการณ์คาดการณ์จะต่ำ และดังนั้นความไม่แน่นอนจะสามารถลดลงได้ด้วยเมตริกซ์ความแปรปรวนของตัวแปรสถานะหลังแก้ไข (Posteriori) จะได้จากกฎของค่าแพร่กระจายของค่าความผิดพลาดดังนี้

$$P^{+}(t_i) = P^{-}(t_i) - K(t_i) HP^{-}(t_i) = (I - K(t_i) H) P^{-}(t_i) \tag{2.31}$$

2.10 เทคนิค ExtendedKalman Filter

เทคนิค Kalman Filter นั้นจะใช้กับระบบเชิงเส้นเพียงอย่างเดียวเท่านั้น แต่ในทางปฏิบัติแบบจำลองทางพลศาสตร์หรือการสังเกตการณ์จะเป็นแบบไม่เชิงเส้น หนึ่งในเทคนิคของ Kalman Filter สำหรับปัญหาที่ไม่เป็นเชิงเส้นนั้น จะถูกเรียกว่า ExtendedKalman Filter ซึ่งถูกค้นพบโดย Stanley F.Schmidt หลังจากที่ Kalman ได้นำเสนอผลลัพธ์ที่ได้จาก Kalman Filtering หลังจากนั้น Schmidt ได้เริ่มนำเทคนิคนี้ไปประยุกต์ใช้สำหรับปัญหาการนำทางอวกาศในทันทีสำหรับโครงการ Apollo เพื่อส่งมนุษย์ไปทำการสำรวจโลกพระจันทร์และเขาได้คิดค้น ExtendedKalman Filter ในเทคนิค Kalman Filter จะประมาณค่าแบบเชิงเส้นรอบ ๆ ตัวแปรสถานะปัจจุบัน ดังนั้นระบบจะจำเป็นต้องถูกแสดงด้วยฟังก์ชันเชิงอนุพันธ์แบบต่อเนื่อง

ข้อเสียหนึ่งของเทคนิคนี้ของ Kalman Filter สำหรับระบบที่ไม่เชิงเส้นมันต้องใช้การคำนวณที่ใช้เวลานานการดำเนินการสำหรับระบบเชิงเส้น จะสามารถทำให้มีประสิทธิภาพมากขึ้น

โดยการคำนวณทางเมตริกซ์ทางพลศาสตร์ (F) ไว้ล่วงหน้าเมตริกซ์การถ่ายโอนตัวแปรสถานะ (Q) และเมตริกซ์การสังเกตการณ์ (H) แต่สำหรับระบบไม่เชิงเส้นเมตริกซ์เหล่านี้จะเป็นฟังก์ชันของตัวแปรสถานะและจะมีการเปลี่ยนแปลงทุกๆ ช่วงเวลาและจะไม่สามารถคำนวณไว้ล่วงหน้าก่อนได้

2.10.1 การคาดการณ์

ในกรณีที่เป็นระบบไม่เป็นเชิงเส้นเมตริกซ์ (F) จะเป็นฟังก์ชันของตัวแปรสถานะซึ่งจะถูกประมาณค่า ดังนั้นตัวแปรสถานะจากการคาดการณ์จะถูกคำนวณโดยการแก้สมการเชิงอนุพันธ์ด้านล่างนี้

$$\dot{\bar{x}}(t_i) = f(\bar{x}(t_i)) \tag{2.32}$$

โดยการแทนสมการนี้ด้วยอนุกรม Taylor เมื่อเทียบกับ xแล้วแทนค่าตัวแปรสถานะที่ได้จากการคาดการณ์ $\dot{\bar{x}}(t_i)$ และสมมติว่าพจน์ที่มีอันดับสูงนั้นสามารถตัดทิ้งได้ เมตริกซ์ทางพลศาสตร์ที่เวลา t หรือ $F(t_i)$ จะสามารถถูกคำนวณโดย

$$F(t_i) = \frac{\partial f(x)}{\partial x}_{x=\bar{x}(t_i)} \tag{2.33}$$

และขั้นตอนอื่นๆ ของการคาดการณ์จะสามารถถูกคำนวณได้แต่ควรจะสังเกตว่าเมตริกซ์ที่ถูกใช้ไม่เป็นค่าคงที่เหมือนในกรณีระบบเชิงเส้น แต่จะขึ้นอยู่กับช่วงเวลา

$$\frac{d}{dt}\Phi_{t_i}^{t_i} = F(t_i)\cdot\Phi_{t_i}^{t_i} - 1 \tag{2.34}$$

$$\bar{P}(t_i) = \Phi_{t_{i-1}}^{t_i}\cdot P(t_i - 1)\cdot(\Phi_{t_{i-1}}^{t_i})^T + \int_{t_{i-1}}^{t_i} Q(t_i)dt \tag{2.35}$$

2.10.2 การแก้ไขค่า

เหมือนกับสมการเชิงอนุพันธ์ในขั้นตอนจากการณคาดการณ์สมการการสังเกตการณ์ที่ไม่เป็นเชิงเส้นที่สอดคล้องกัน จะถูกทำให้เป็นเชิงเส้นโดยใช้อนุกรม Taylor รอบๆ ตัวแปรสถานะที่ได้จากการคาดการณ์ $\dot{\bar{x}}(t_i)$ และพจน์อันดับสูงจะถูกตัดทิ้งไปดังนั้นเมตริกซ์การสังเกตการณ์ที่ถูกประมาณจะได้เป็น

$$H(t_i) = \frac{\partial h(x)}{\partial x}_{x=x^-(t_i)} \tag{2.36}$$

ในกรณีนี้การวัดจากการณ์คาดการณ์ $l^-(t_i)$ สำหรับการคำนวณค่าที่เหลือจากการวัด $l(t_i) - l^-(t_i)$ เป็น

$$l^-(t_i) = h(x^-(t_i)) \tag{2.37}$$

นอกจากนี้เราสามารถใช้สูตรเดียวกันอีกครั้งในการคำนวณตัวแปรสถานะจากการแก้ไขค่าและเมตริกซ์ความแปรปรวนได้เหมือนในกรณีแบบเชิงเส้นแต่เมตริกซ์เหล่านี้จะเป็นฟังก์ชันของเวลา

$$x^+(t_i) = x^-(t_i) + K(l(t_i) - l^-(t_i)) \tag{2.38}$$

และ $$P^+(t_i) = (I - K(t_i)H(t_i))P^-(t_i) \tag{2.39}$$

กับ $$K(t_i) = P^-H(t_i)^T(H(t_i)P^-H(t_i)^T + R(t_i))^{-1} \tag{2.40}$$

2.11 การระบุตำแหน่งพร้อมกับการสร้างแผนที่ (SLAM)

การระบุตำแหน่งพร้อมกับการสร้างแผนที่ (SLAM) เป็นกระบวนการที่หุ่นยนต์จะสร้างแผนที่ของสภาพแวดล้อมในขณะที่กำลังเคลื่อนที่และระบุตำแหน่งของตัวเองในเวลาพร้อม ๆ กัน โดยที่หุ่นยนต์นั้นไม่มีข้อมูลของสิ่งแวดล้อมมาก่อน เนื่องจากว่าการใช้งาน SLAM มักจะเกี่ยวข้องกับอุปกรณ์วัดค่าต่าง ๆ (Sensors) ซึ่งค่าที่วัดได้จากอุปกรณ์ต่าง ๆ นั้นย่อมมีความคลาดเคลื่อนของการวัดค่าเกิดขึ้น จึงทำให้แผนที่และตำแหน่งของหุ่นยนต์ที่ประมาณได้มีความไม่แน่นอนเกิดขึ้น ดังนั้นหลักการทั่วไปของ SLAM จะใช้วิธีการทางความน่าจะเป็น (probabilistic methods) มาจัดการความไม่แน่นอนเหล่านี้ให้มีค่าต่ำสุด

จุดประสงค์ของ SLAM คือการประมาณการกระจายความน่าจะเป็นของตำแหน่งหุ่นยนต์และแผนที่ เมื่อกำหนดค่าการวัด (Measurement) จากอุปกรณ์วัดค่า (Sensor) และคำสั่งควบคุมหุ่นยนต์ (Control)

$$p(x_{t:1}, m / z_{t:1}, u_{t:1}) \tag{2.41}$$

โดยกำหนดให้

$x_{t:1} = \{x_1, x_2, ...x_t\}$ เป็นเซตของสถานะของหุ่นยนต์ตั้งแต่เวลา 1 ถึงเวลา t

$m = \{m_1, m_2, ...m_n\}$ เป็นเซตของแผนที่ซึ่ง m_i อาจแทนจุดสังเกตที่ใช้อธิบายแผนที่

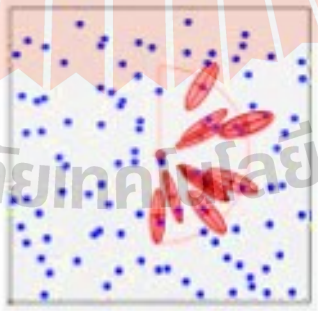
$z_{t:1} = z^t = \{z_1, z_2, ...z_t\}$ เป็นเซตของสถานะของค่าการวัด (Measurement) ตั้งแต่เวลา 1 ถึงเวลา t

$u_{t:1} = u^t = \{u_1, u_2, ...u_t\}$ เป็นเซตของสถานะของคำสั่งควบคุมหุ่นยนต์ (Control) ตั้งแต่เวลา 1 ถึงเวลา t

2.11.1 รูปแบบการการอธิบายแผนที่ของ SLAM

การอธิบายแผนที่ของ SLAM สามารถแบ่งออกเป็นประเภทใหญ่ ๆ ได้สองประเภทหลักด้วยกันคือ Feature-based SLAM และ View-based SLAM

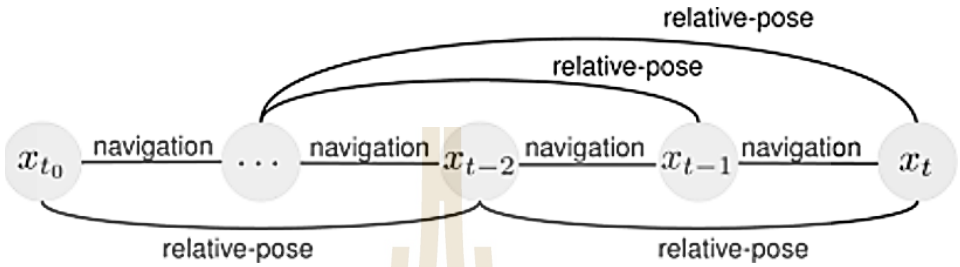
Feature-based SLAM จะอธิบายแผนที่ด้วยจุดสังเกตจำนวนมากซึ่งในที่นี้จุดสังเกตอาจแทนด้วย ตำแหน่งของวัตถุ, จุดกลุ่มหมอก (Point Cloud), เส้นตรงในสิ่งแวดล้อมหรืออื่น ๆ ที่มีลักษณะเด่นแผนที่แบบ Feature-basedเป็นการอธิบายแผนที่ที่มีความนิยมสูงในการอธิบายสิ่งแวดล้อมได้อย่างตรงตัว ตามรูปที่ 2.17



รูปที่ 2.17 Feature-based representation

View-based SLAM จะอธิบายแผนที่ด้วยความสัมพันธ์ของมุมมองของหุ่นยนต์ในอดีต ถึงแม้ว่าแผนที่แบบ View-based จะไม่สามารถอธิบายลักษณะของสิ่งแวดล้อมได้โดยตรงแต่เราก็สามารถสร้างแผนที่แบบ Feature-based ในบริเวณที่สนใจได้จากการส่งค่าของการวัด ณ ขณะที่

หุ่นยนต์อยู่ใน View-based นั้น ๆ ลงบนแผนที่ข้อดีของแผนที่แบบ View-based ก็คือแผนที่แบบ View-based มีประสิทธิภาพในการคำนวณสูง, มีความซับซ้อนต่ำตามรูปที่ 2.18



รูปที่ 2.18 View-based representation

2.11.2 รูปแบบการประมาณคำตอบของ SLAM

รูปแบบการประมาณคำตอบของ SLAM แบ่งออกได้เป็นสองประเภทคือ Full SLAM และ Online SLAM

Full SLAM นั้นจะประมาณสถานะทั้งหมดของ SLAM ได้แก่แผนที่ทั้งหมด ตำแหน่งของหุ่นยนต์ตั้งแต่เวลาเริ่มต้นจนถึงปัจจุบันโดยการกระจายความน่าจะเป็นสามารถเขียนได้ดังนี้

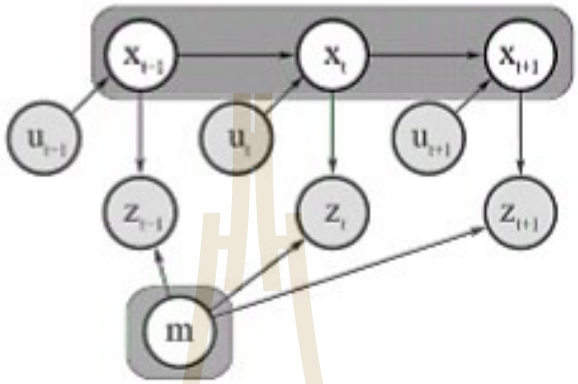
$$p(x_{t:1}, m / z_{t:1}, u_{t:1}) \tag{2.42}$$

การประมาณคำตอบแบบ Full SLAM จะมีความแม่นยำสูงเพราะประมาณสถานะทั้งหมด แต่ก็ใช้เวลาในการประมวลผลนานเช่นกัน ดังนั้นโดยทั่วไปแล้ว Full SLAM มักจะไม่ทำงานแบบ Real Time

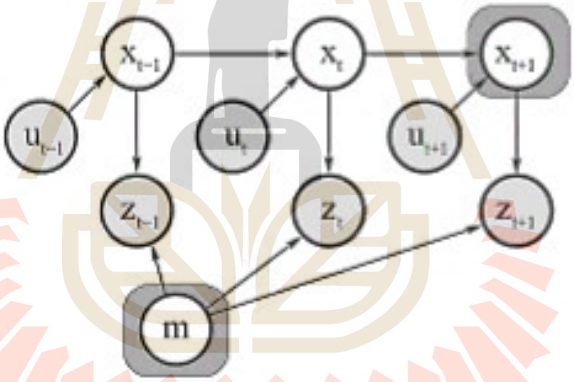
Online SLAM นั้นจะประมาณสถานะของหุ่นยนต์ ณ เวลาปัจจุบันและสถานะของแผนที่ทั้งหมดทำให้เวลาที่ใช้ในการประมวลผลไม่นานเกินไป และสามารถทำงานแบบ Real Time ได้ ณ เวลาหนึ่ง ๆ การประมาณการกระจายความน่าจะ เป็นของ Online SLAM สามารถทำได้โดยการ Marginalize Out สถานะของหุ่นยนต์ก่อนหน้าทำให้ Online SLAM สามารถลดจำนวนสถานะลงได้

$$p(x_{t:1}, m / z_{t:1}, u_{t:1}) = \int .....\iint px_{t:1}, m / z_{t:1}, u_{t:1}) dx_1 dx_2 .....dx_{t-1} \tag{2.43}$$

แผนภาพ Graphical Model ของ Full SLAM และ Online SLAM แสดงได้ ดังรูปที่ 2.19 และรูปที่ 2.20ตามลำดับโดย Random Variable ที่อยู่ในกรอบสีเทาจะเป็นสถานะที่ถูกประมาณโดย SLAM



รูปที่ 2.19 แผนภาพ Graphical Model ของ Full SLAM



รูปที่ 2.20 แผนภาพ Graphical Model ของ Online SLAM

2.12 ROS Navigation

การที่หุ่นยนต์สำรวจจะสามารถทำงานได้ขั้นตอนแรกจะต้องสามารถขับเคลื่อนให้หุ่นยนต์สำรวจไปในตำแหน่งที่ต้องการได้อย่างถูกต้อง ปลอดภัยและไม่ชนกับสิ่งกีดขวางที่อยู่ระหว่างทาง ซึ่งในการทำงานลักษณะนี้เรียกว่า การนำทาง (Robot Navigation) ซึ่งเป็นการพัฒนาการนำทางให้กับหุ่นยนต์สำรวจได้นั้น จะต้องมีปัจจัยหลายอย่าง ทั้งฮาร์ดแวร์และซอฟท์แวร์ แต่ปัจจุบัน ROS ได้พัฒนาไลบรารีที่อยู่ในรูปแบบของ ROS Stack โดยจะเรียกว่า Navigation Stack

2.12.1 องค์ประกอบของ Navigation Stack

การทำให้หุ่นยนต์สามารถเคลื่อนที่ไปในตำแหน่งต่างๆ ที่ต้องการได้โดยไม่ชนกับสิ่งกีดขวางมีความจำเป็นที่จะต้องอาศัยการนำทางหรือ Navigation ซึ่งในระบบ ROS ได้มี ROS Navigation Stack เอาไว้ให้ สามารถปรับใช้ได้ตามลักษณะของงานโดยมีองค์ประกอบของ ROSNavigation Stack มีดังต่อไปนี้

2.12.1.1 ฮาร์ดแวร์

การที่ ROS Navigation Stack จะสามารถทำงานได้ จำเป็นจะต้องมีฮาร์ดแวร์ที่เฉพาะเจาะจง ได้แก่ เซนเซอร์ Lidarและส่วนของการขับเคลื่อนหุ่นยนต์สำรวจ โดยจะขึ้นอยู่กับประเภทของหุ่นยนต์ที่ใช้ ซึ่งในการทำงานของหุ่นยนต์สำรวจจะใช้ DYNAMIXEL ในการขับเคลื่อนและบอร์ด Raspberry Pi,บอร์ด OpenCR นอกจากนั้นก็ยังมีเรื่องของเซนเซอร์ต่างๆ ที่มีความจำเป็นในการทำงานร่วมกับหุ่นยนต์สำรวจ

2.12.1.2 ระบบการแปลงพิกัดหรือ TF (Transform Configuration)

การใช้งาน ROS Navigation จะต้องเข้าใจหลักการทำงานของ TF เนื่องจากมีความสำคัญในทุกๆ ส่วนการที่มี TF ทำให้ข้อมูลเรื่องตำแหน่งต่างๆ ในแต่ละมุมมองหรือระบบพิกัด (Coordinate) สามารถเชื่อมโยงเข้าหากันได้ เช่น ค่าที่วัดได้จากเซนเซอร์ Lidarเมื่อวัดจากศูนย์กลางของตัวหุ่นยนต์สำรวจก็จะต้องมีการแปลง โดยการแปลงนั้นจะได้แก่ระยะทางระหว่างเซนเซอร์ Lidar เทียบกับตัวหุ่นยนต์สำรวจ

2.12.1.3 แผนที่ (Map)

สำหรับเรื่องของการนำทาง Navigation จะไม่สามารถทำงานได้หากไม่มีแผนที่เข้ามาเกี่ยวข้อง ซึ่งแผนที่สามารถสร้างได้จากการใช้เซนเซอร์ Lidar หรืออุปกรณ์อื่นๆ เช่น Kinect เป็นต้น ในการสร้างแผนที่จะมีอัลกอริทึมที่สำคัญคือ SLAM (Simultaneous Localization and Mapping) ซึ่งการสร้างแผนที่จะส่งข้อมูลเป็น Topic แล้วทำการจัดเก็บไว้ใน Map Server

2.12.1.4 การรู้ตำแหน่งของหุ่นยนต์สำรวจ (Locallization)

การรู้ตำแหน่งของหุ่นยนต์สำรวจ หรือ Locallization เป็นสิ่งที่มีความสำคัญอีกเรื่องหนึ่งในการทำงานROS Navigationเพราะว่าหุ่นยนต์สำรวจไม่สามรถรู้ตำแหน่งและทิศทางของหุ่นยนต์สำรวจเอง หุ่นยนต์สำรวจก็ไม่สามารถเคลื่อนที่ไปในตำแหน่งที่ต้องการได้อย่างถูกต้อง เปรียบกับการเคลื่อนที่ที่ไม่มีทิศทางและจุดหมายปลายทาง สำหรับ ROS มีอัลกอริทึมที่เรียกว่า เรียกว่า AMCL (Adaptive Monte Carlo Localization) ซึ่งจะสามารมช่วยให้หุ่นยนต์สำรวจกำลังอยู่ตำแหน่งใดในแผนที่

2.12.1.5 แผนที่ Costmap

แผนที่ Costmap ต่างจากแผนที่ (Map) เนื่องจาก Costmap จะทำการเก็บข้อมูลที่ใช้สำหรับหุ่นยนต์สำรวจที่จะสามารถเคลื่อนที่ผ่านไปในจุดใดก็ได้ โดยไม่ติดขัด ซึ่งสามารถปรับค่าพารามิเตอร์ของ Costmap เพื่อให้หุ่นยนต์สามารถเคลื่อนที่เข้าไปเบียดหรือใกล้กับวัตถุได้ ซึ่งการมี Costmap จะช่วยให้การนำทางของหุ่นยนต์สำรวจสามารถทำงานได้อย่างราบรื่น

2.12.1.6 การสร้างแผนการเดิน (Path Planning)

เป็นอัลกอริทึมที่สร้างชุดของเส้นทาง (Path) เพื่อให้ไปถึงเป้าหมายได้ ซึ่งแผนในการเคลื่อนทีจะมีทั้งแบบระยะใกล้ (Local Path Planning) และระยะไกล (Global Path Planning) โดยมีขอบเขตในการทำงานแตกต่างกันคือ ระยะไกลจะสร้างเส้นทางที่จะเคลื่อนที่ไปสู่จุดหมาย ส่วนระยะใกล้เป็นอัลกอริทึมที่จะหาเส้นทางที่จะเคลื่อนที่ไปตามเส้นทางที่ได้จากแผนที่ระยะไกล

2.13 งานวิจัยที่เกี่ยวข้อง

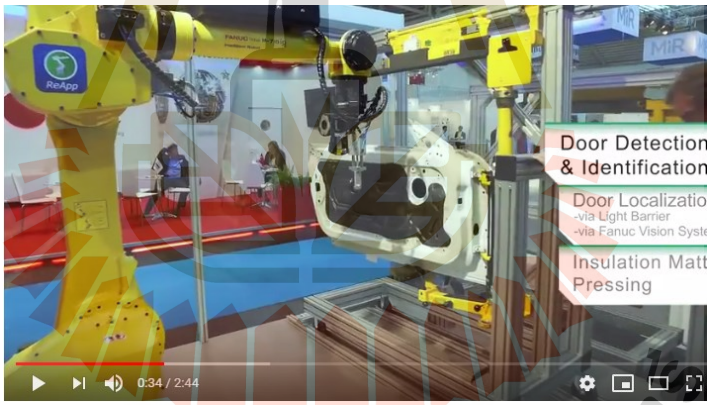
ในอดีตการพัฒนาหุ่นยนต์ยังถือว่าเป็นการเริ่มต้นเนื่องจากปัญหาทางด้านคอมพิวเตอร์และเซนเซอร์ แต่ปัจจุบันสิ่งต่างๆ เหล่านี้มีการพัฒนาไปมาก เครื่องคอมพิวเตอร์มีประสิทธิภาพสูงขนาดเล็กและราคาถูก จึงทำให้เกิดการพัฒนาเทคโนโลยีของเครื่องจักรให้มีประสิทธิภาพตามไปด้วย แต่ก็ยังมีส่วนที่สำคัญในการพัฒนาหุ่นยนต์ได้แก่การพัฒนาทางด้านของซอฟท์แวร์

ในการพัฒนาซอฟท์แวร์สำหรับหุ่นยนต์นั้นส่วนใหญ่แล้วจะเป็นการพัฒนาแบบแยกส่วนและเน้นไปในทิศทางทิศทางหนึ่ง เช่นระบบควบคุมอัตโนมัติต่างๆ เฉพาะด้าน เป็นต้น และเป็นการพัฒนาจากผู้ที่มีความเชี่ยวชาญทางด้านนั้นๆ จึงทำให้ซอฟท์แวร์ของหุ่นยนต์มีข้อจำกัดในการทำงาน แต่เมื่อมีการนำ ROS เข้ามาพัฒนาซอฟท์แวร์ทางด้านหุ่นยนต์แล้วทำให้เปิดกว้างในทุกๆ ด้านของซอฟท์แวร์ เช่น รหัสคำสั่งโปรแกรม ฮาร์ดแวร์ต่างๆ การแบ่งปันความรู้ ดังนั้น ROS จึงเพิ่มผู้พัฒนาหุ่นยนต์ทั่วโลกได้อย่างมาก โดยสามารถเปลี่ยนแปลง แบ่งปัน สิ่งต่าง ๆ ให้แก่กันได้อย่างเปิดกว้างจึงทำให้ซอฟท์แวร์ของ ROS ได้รับความนิยมอย่างแพร่หลายในตอนนี้

YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, TaeHoon Lim [3]ได้กล่าวไว้ว่าหุ่นยนต์ส่วนบุคล (Personal Robot) จะเป็นสิ่งที่เกิดขึ้นในอนาคต คนส่วนใหญ่จะมีหุ่นยนต์เป็นของตัวเองและสามารถสั่งงาน ควบคุม เขียนโปรแกรม ดัดแปลงหุ่นยนต์ได้ตามที่ต้องการ เช่นการใช้เครื่องใช้ไฟฟ้าที่มีอยู่ภายในบ้าน ซึ่ง ROS เป็นซอฟท์แวร์หุ่นยนต์ที่มีความพร้อมทางด้านนี้ จึงเป็นข้อได้เปรียบในการศึกษาและใช้งาน ROS สำหรับผู้ที่ต้องการพัฒนาหุ่นยนต์

ROS เกิดจากองค์กรที่ชื่อว่า Willow Garage ซึ่งได้พัฒนา ROS ขึ้นมาในรูปแบบปิดเผย (Open Source)[13] โดยใช้รูปแบบลิขสิทธิ์ของ BSD ทำการผลิตหุ่นยนต์ที่ใช้ ROS จำหน่าย เช่น TurtleBot, PR2 เป็นต้น ซึ่ง ROS เป็นที่ยอมรับในทุกระดับทั้งอุตสาหกรรม การศึกษา จึงทำให้ ROS ได้รับความนิยมเป็นอย่างมากรูปที่ 2.21แสดงตัวอย่างที่ใช้กับ ROS ที่ทำงานอยู่และรูปที่ 2.22 แสดงหุ่นยนต์อุตสาหกรรมที่ใช้กับ ROS

รูปที่ 2.21 ตัวอย่างที่ใช้กับ ROS ที่ทำงานอยู่ [3]



รูปที่ 2.22 หุ่นยนต์อุตสาหกรรมที่ใช้กับ ROS [13]

ROS เป็นซอฟท์แวร์หุ่นยนต์ที่ได้รับความนิยมมากที่สุดในปัจจุบันแล้ว เนื่องจากROS มีข้อดีหลายอย่าง โดยที่ ROS เป็น Open source ได้แก่ ซอฟท์แวร์, ไลบรารี, หน่วยโปรแกรมต่าง ๆ ช่วยให้นักวิจัยสามารถทำการจำลองและการทดลองได้อย่างรวดเร็วด้วยโปรแกรมGazebo และโปรแกรม Rviz ที่มีอยู่ในระบบปฏิบัติการ ROS สำหรับงานวิจัยที่หุ่นยนต์ที่มีการเคลื่อนที่แบบอัตโนมัติจะมีข้อจำกัดเกี่ยวกับเซ็นเซอร์ เนื่องจากมีราคาค้อนข้างแพงมีงานวิจัยของ Safdar Zaman, Wolfgang Slany, Gerald Steinbauer[5] ได้ใช้หุ่นยนต์ Pioneer 3-Dx ร่วมกับROS ซึ่งในงานวิจัยได้

นำเสนอพื้นฐานของหุ่นยนต์ในการสร้างแผนที่และระบุตำแหน่งพร้อมทั้งกำหนดทิศทางการเคลื่อนที่ของหุ่นยนต์แบบอัตโนมัติด้วยกล้อง Pioneer 3-Dx นอกจากนี้ยังได้มีการนำเสนอปัจจัยที่อาจจะทำให้เกิดข้อผิดพลาดในการทำงานของหุ่นยนต์ในการสร้างแผนที่ได้ เนื่องจากปัจจัยที่เกิดจากเซนเซอร์ เช่น ตำแหน่งในการติดตั้งเซนเซอร์ที่อยู่บนตัวหุ่นยนต์อาจจะมีผลในการเคลื่อนที่สำรวจในพื้นที่ที่มีความสูงต่ำไม่เท่ากัน ปัจจัยต่อมาเกี่ยวกับล้อของหุ่นยนต์ซึ่งอาจจะส่งผลกระทบในการเคลื่อนที่ของหุ่นยนต์ทำให้เกิดความล่าช้าและเกิดปัญหาในการเลื่อนไถล และปัจจัยของสภาพพื้นผิวที่มีลักษณะที่มีความสูงของพื้นผิวหรือมีความลาดเอียง อาจจะทำให้การทำงานของเซนเซอร์เกิดข้อผิดพลาดได้ จึงมีการแก้ปัญหาในการสร้างแผนที่เป็นแบบสามมิติ [6] รูปที่ 2.23 แสดงหุ่นยนต์Pioneer 3-Dx

รูปที่ 2.23 หุ่นยนต์ Pioneer 3-Dx [6]

SLAM ย่อมาจาก Simultaneous Localization and Mapping ซึ่งหมายถึงการสร้างแผนที่และการค้นหาตำแหน่งของตนเอง ซึ่งส่วนใหญ่แล้วหุ่นยนต์สร้างแผนที่จะสร้างแผนที่ด้วยเซนเซอร์ต่าง ๆ เช่น เซนเซอร์ Lidar หรือ กล้อง Stereo kinect โดย ROS มีไลบรารีที่เป็น SLAM หลายชุด เช่น slam gmapping, hector slam, slam karto เป็นต้น [3] hector slam [7], [8], [9] เป็นงานวิจัยที่ใช้ข้อมูลจากเซนเซอร์ Lidar ทำงานร่วมกับหุ่นยนต์ UGV เป็นการนำเสนองานวิจัยที่เกี่ยวกับหุ่นยนต์ที่มีการเคลื่อนที่แบบอัตโนมัติโดยการใช้วิธีของ Hector SLAM ในรูปแบบเลเซอร์สแกน โดยใช้อัลกอริทึมที่สามารถสร้างแผนที่เป็นแบบ 2D และการระบุตำแหน่งได้อย่างแม่นยำสำหรับการออกแบบหุ่นยนต์ให้มีการเคลื่อนที่อัตโนมัตินั้นB. L. E. A. Balasuriya et al. [4]การใช้อัลกอริทึม Gmapping ของ SLAM เป็นงานวิจัยที่สามารถตรวจสอบสิ่งกีดขว้างต่าง ๆ โดยใช้ระบบ

ROS ในการระบุตำแหน่งและการสร้างแผนที่ในห้องที่มีขนาดกว้าง ได้นำเสนอการสร้างแผนที่ของพื้นที่ไม่รู้จักและการระบุตำแหน่งของหุ่นยนต์ เป็นวิธีของหุ่นยนต์เคลื่อนที่แบบอัตโนมัติ มีงานวิจัยจำนวนมากมุ่งเน้นไปที่ปัญหาเกี่ยวกับการสร้างแผนที่ และการระบุตำแหน่ง ซึ่งหุ่นยนต์ SLAM ในการสร้างแผนที่และการระบุตำแหน่งของหุ่นยนต์จะขึ้นอยู่กับเซ็นเซอร์ต่างๆ เช่น kinects, เลเซอร์สแกนเนอร์หรือเซ็นเซอร์อัลตราโซนิค [10], [11], [12] เป็นต้น

งานวิจัยด้านการระบุตำแหน่งพร้อมกับการสร้างแผนที่ (Simultaneous Localization and Mapping: SLAM) เป็นงานที่สามารถนำมาใช้ในการสร้างหุ่นยนต์เคลื่อนที่ (Mobile Robot) เนื่องจากหุ่นยนต์มีความจำเป็นต้องรับรู้ถึงข้อมูลตำแหน่งของสิ่งแวดล้อมต่าง ๆ ในบริเวณโดยรอบเพื่อที่จะทำการสร้างแผนที่รอบ ๆ ร่วมถึงการระบุตำแหน่งการทำงานของหุ่นยนต์ได้ด้วย

การใช้งานอุปกรณ์รับรู้ เช่น เซนเซอร์ โดยทั่วไปข้อมูลที่ได้จะมีค่าความไม่แน่นอนเนื่องจากการวัดระยะทางของอุปกรณ์รับรู้เองหรือเกิดจากสิ่งรบกวน (Noise) จึงต้องมีการจัดการกับความไม่แน่นอนเหล่านี้ วิธีที่นิยมใช้ในการจัดการกับสิ่งที่ไม่แน่นอนเหล่านี้มีหลายงานวิจัยได้แก่ งานวิจัยที่ได้เสนอการใช้แบบจำลองต่าง ๆ ในการประมาณตำแหน่งของหุ่นยนต์ เช่น Kalman Filter, Extended Kalman Filter, Particle Filter, MHT(Multiple Hypothesis Tracking) [14], [15],[16] เป็นต้น ในสภาพแวดล้อมที่หุ่นยนต์ไม่สามารถที่จะเคลื่อนที่แบบไร้ทิศทาง (Brownian Motion) แต่จะมีการเคลื่อนที่แบบมีเส้นทางการเคลื่อนที่ที่ค่อนข้างแน่นอนและมีเป้าหมายที่หุ่นยนต์ต้องการจะเคลื่อนที่

อัลกอริทึมที่ใช้ในการแก้ปัญหา SLAM นั้นมีหลายวิธีแต่ที่เป็นที่นิยมนั้นจะมีอยู่ด้วยกันแต่ 3 วิธีคือ Fast SLAM, EKF-SLAM และ Information-Based SLAM

Fast SLAM [17], [18]นั้นจะแก้ปัญหา SLAM โดยใช้ Particle Filter โดยแต่ละ Particle จะเก็บค่าสถานะของแผนที่ไว้ แต่ในระยะหลังนี้ความนิยมในการใช้งาน Fast SLAM เริ่มลดลงเนื่องจากมีวิธีอื่นทำงานได้เร็วกว่าและมีประสิทธิภาพที่มากกว่า

EKF-SLAM [19], [20], [21]จะแก้ปัญหา SLAM โดยใช้ Extended Kalman Filter ซึ่งเป็นวิธีในการประมาณสถานะของระบบแบบ Dynamic จากข้อมูลการวัดที่มีความคลาดเคลื่อนสำหรับการแก้ปัญหาแบบ EKF-SLAM นั้นจะกำหนดให้แผนที่และตำแหน่งของหุ่นยนต์เป็นสถานะของระบบโดยจะอธิบายในรูปของ Covariance Form จากนั้นเมื่อได้รับข้อมูลการวัดจากอุปกรณ์วัดค่าจะนำเอาข้อมูลการวัดมาปรับแก้สถานะของระบบ EKF-SLAM นั้นถูกนำมาใช้งานอย่างกว้างขวางเนื่องจากมีประสิทธิภาพในการประมาณสถานะค่อนข้างดีและมีความเร็วสูง สำหรับสิ่งแวดล้อมที่มี

ขนาดไม่ใหญ่นัก แต่เนื่องจาก EKF-SLAM มีประสิทธิภาพเป็นโดยที่เป็นขนาดของแผนที่ทำให้ EKF-SLAM อาจจะเกิดปัญหาด้านเวลาในการคำนวณสิ่งแวดล้อมต่างๆ

Information-Based SLAM [22]จะมีขั้นตอนในการประมาณสถานะของระบบคล้ายกับ EKF-SLAM แต่จะอธิบายสถานะของระบบด้วย Information Vector และ Information Matrix ซึ่งเป็น Dual ของ Covariance Form แต่ในระยะหลังนี้ได้มีงานวิจัยเกี่ยวกับปัญหา SLAM ที่อยู่ในรูป Information-Based จำนวนมากทั้งนี้เนื่องจากข้อดีของ Information-Based ซึ่งเวลาในการ Predict และ Update สถานะของ SLAM เป็น O(1) และ O(m) ตามลำดับเมื่อ m เป็นขนาดของข้อมูลการวัด (measurement) อย่างไรก็ดีสถานะของระบบในรูป Information Form นั้นอยู่ในรูปที่หุ่นยนต์นำไปใช้ไม่ได้ จึงต้องทำการ State Recovery เพื่อแปลงสถานะให้อยู่ในรูป Covariance Form โดยเวลาที่ใช้ในการ Recovery เป็น O(n3) ซึ่งก็เทียบเท่ากับ EKF-SLAM อย่างไรก็ดี Thrun et al. [23] และ Hirzinger [24]ได้พบว่าสำหรับ Feature-based SLAM นั้น ค่าส่วนใหญ่ใน Information Matrix มีค่าเข้าใกล้ศูนย์

จากงานวิจัยที่กล่าวมาข้างต้นวิทยานิพนธ์ฉบับนี้จะเลือกใช้แนวทางในการสร้างแผนที่และการค้นหาตำแหน่งของหุ่นยนต์สำรวจด้วยวิธี SLAM โดยใช้ROS ร่วมกับเซนเซอร์ Lidarในการตรวจจับสิ่งกีดขวางหรือวัตถุที่อยู่รอบ ๆ โดยการสร้างแผนที่นั้นจะใช้วิธีการ SLAM Gmapping ของ ROS ซึ่งจะสร้างแผนที่ จะเป็นการส่งเป็นข้อมูลแล้วทำการสร้างแผนที่จัดเก็บไว้ใน Map Server ในส่วนของการรู้ตำแหน่งสำหรับ ROS มีอัลกอริทึมที่เรียกว่า AMCL (Adaptive Monte Carlo Localization) ซึ่งจะทำให้รู้ว่าหุ่นยนต์อยู่จุดใดในแผนที่ ในการทำงานของหุ่นยนต์สำรวจอัตโนมัติด้วยวิธี SLAM นั้น แบ่งออกเป็นสองส่วนคือ ส่วนแรกการสำรวจและสร้างแผนที่และส่วนที่สองระบุตำแหน่งเคลื่อนที่อัตโนมัติของหุ่นยนต์

บทที่ 3
วิธีดำเนินงานวิจัย

3.1 ขั้นตอนในการดำเนินงานวิจัย

สำหรับการสร้างหุ่นยนต์สำรวจนี้ได้มีการวางแผนการดำเนินงาน ซึ่งได้สรุปไว้ในตารางแสดงแผนการดำเนินงานของงานวิจัยดังนี้

ตารางที่ 3.1 แผนงานการดำเนินงานวิจัย

กิจกรรม / ขั้นตอนการดำเนินงานวิจัย	**2560**						**2561**				
	ม.ค. ก.พ.	มี.ค. เม.ย.	พ.ค. มิ.ย.	ก.ค. ส.ค.	ก.ย. ต.ค.	พ.ย. ธ.ค.	ม.ค. ก.พ.	มี.ค. เม.ย.	พ.ค. มิ.ย.	ก.ค. ส.ค.	ก.ย. ต.ค.
1.ศึกษาค้นคว้าองค์ความรู้/ทฤษฎีและเอกสารที่เกี่ยวข้อง											
2.ศึกษารูปแบบของโครงสร้างและองค์ประกอบส่วนต่างๆ ของหุ่นยนต์สำรวจ											
3.ศึกษาและออกแบบการควบคุมหุ่นยนต์สำรวจด้วยระบบปฏิบัติการหุ่นยนต์(ROS)											

ตารางที่ 3.1 แผนงานการดำเนินงานวิจัย (ต่อ)

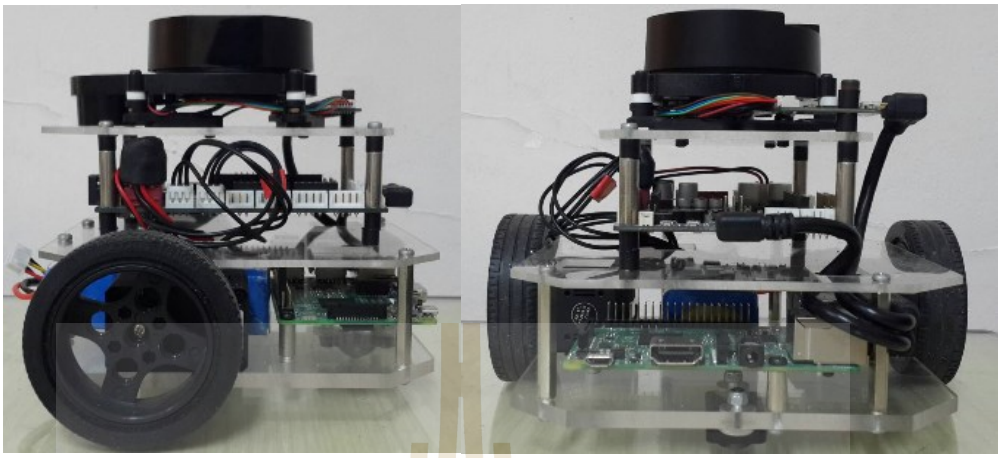
กิจกรรม / ขั้นตอนการดำเนินงานวิจัย	2560						2561				
	ม.ค. ก.พ.	มี.ค. เม.ย.	พ.ค. มิ.ย.	ก.ค. ส.ค.	ก.ย. ต.ค.	พ.ย. ธ.ค.	ม.ค. ก.พ.	มี.ค. เม.ย.	พ.ค. มิ.ย.	ก.ค. ส.ค.	ก.ย. ต.ค.
4.ศึกษาและทดลองใช้อุปกรณ์วัดระยะทางด้วยเซนเซอร์ Lidar											
5.ศึกษาและทดลองใช้อุปกรณ์วัดระยะทางด้วยเซนเซอร์ Gyro, Accelometer,											
6.ศึกษาทฤษฎีพื้นฐานของ slam ในการใช้ระบบปฏิบัติการหุ่นยนต์ ROS											
7.ศึกษาทฤษฎีพื้นฐานของ slam navigationในการใช้ระบบปฏิบัติการหุ่นยนต์ ROS											
8.ออกแบบและสร้างหุ่นยนต์สำรวจ											

ตารางที่ 3.1 แผนงานการดำเนินงานวิจัย (ต่อ)

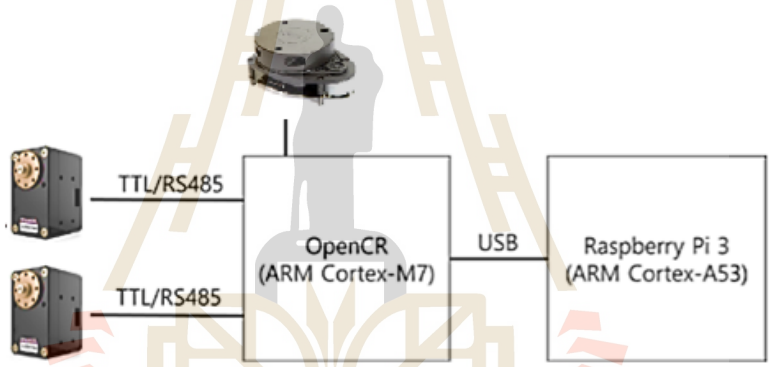
กิจกรรม / ขั้นตอนการดำเนินงานวิจัย	2560						2561				
	ม.ค. ก.พ.	มี.ค. เม.ย.	พ.ค. มิ.ย.	ก.ค. ส.ค.	ก.ย. ต.ค.	พ.ย. ธ.ค.	ม.ค. ก.พ.	มี.ค. เม.ย.	พ.ค. มิ.ย.	ก.ค. ส.ค.	ก.ย. ต.ค.
9.ทดสอบการควบคุมการเคลื่อนที่ของหุ่นยนต์สำรวจในการเคลื่อนที่แบบไร้สาย											
10.ทดลองและวัดผลการทดลอง											
11.สรุปผลการทดลอง											
12.จัดทำรูปเล่มและนำเสนอวิทยานิพนธ์ / แก้ไขวิทยานิพนธ์และเผยแพร่ผลงานวิจัย											

3.2 เครื่องมือที่ใช้ในการทำงานวิจัย

โครงสร้างทั้งหมดของหุ่นยนต์สำรวจตัวนี้ได้ทำการออกแบบตามลักษณะของท่อ ซึ่งเกณฑ์การออกแบบได้เน้นไปที่ความกะทัดรัด ขนาดเล็ก น้ำหนักเบา โดยโครงสร้างหุ่นยนต์สำรวจจะเป็นแผ่นพลาสติกมีทั้งหมด 3 ชั้น ในแต่ละชั้นประกอบไปด้วยชั้นบนสุดจะเป็นส่วนของเซนเซอร์ Lidarชั้นที่ 2 จะเป็นส่วนของบอร์ด OpenCR และชั้นที่ 1 จะเป็นส่วนของบอร์ด Raspberry Pi, มอเตอร์ DYNAMIXE และแบตเตอรี่ รูปที่ 3.1แสดงถึงภาพของหุ่นยนต์สำรวจหลังจากที่ได้ทำการประกอบสมบรูณ์แล้วและรูปที่ 3.2แผนภาพโครงสร้างของหุ่นยนต์สำรวจ



รูปที่ 3.1 หุ่นยนต์สำรวจที่ประกอบเสร็จ



รูปที่ 3.2 แผนภาพโครงสร้างของหุ่นยนต์สำรวจ

คุณสมบัติทางกายภาพของหุ่นยนต์สำรวจได้แสดงไว้ในตารางที่ 3.2 ซึ่งมอเตอร์ที่ใช้เป็นแบบ DYNAMIXEL จำนวน 2 ตัว เป็นตัวขับเคลื่อนให้หุ่นยนต์สำรวจทำงานส่วนของเซนเซอร์ Lidarใช้ในการตรวจสอบสิ่งกีดขวางและหลบหลีกสิ่งกีดขวางส่วนของบอร์ด OpenCR และบอร์ด Raspberry Pi ทำหน้าที่ในการประมวลผลและควบคุมในการทำงานของระบบ

ตารางที่ 3.2 คุณสมบัติทางกายภาพของหุ่นยนต์สำรวจ

ข้อมูล	หุ่นยนต์สำรวจ
ความเร็วสูงสุดในการเคลื่อนที่	0.22 m/s
ความเร็วสูงสุดในการหมุน	2.84 rad/s
ขนาด (กว้าง x ยาว x สูง)	138 mm x 175 mm x 135 mm
น้ำหนัก	1 kg
เวลาในการทำงาน	2 h 30 m
เวลาในการชาร์ตแบตเตอรี่	2 h 30 m
หน่ายประมวลผล	32-bit ARM Cortex-M7 with FPU (216 MHz, 462 DMIPS)
เซนเซอร์	Gyroscope 3 Axis Accelerometer 3 Axis Magnetometer 3 Axis
พลังงานที่ใช้	3.3 V / 800 mA 5 V / 4A 12 V / 1A
ขาที่รองรับการใช้งาน	GPIO 18 pins Arduino 32 pin
อุปกรณ์ต่อพ่วง	UART x3,CAN x1,SPI x1,I2C x1,ADC x5,5pin OLLO x4
Dynamixelพอร์ต	RS458 x3, TTL x3
สถานะ LEDs	Board status LED x1 Arduino LED x1 Power LED x1
ปุ่มกดและสวิทช์	Push buttons x2,Reset button x1,Dip switch x2
แบตเตอรี่	Lithium polymer 11.1V 1800mAh/19.98Wh 5C
การเชื่อมต่อ	USB
เฟิร์มแวร์	Via USB /via JTAG
พลังงาน Adapter	Input : 100-240V,AC 50/60Hz,1.5A@max Output :12V DC,5A

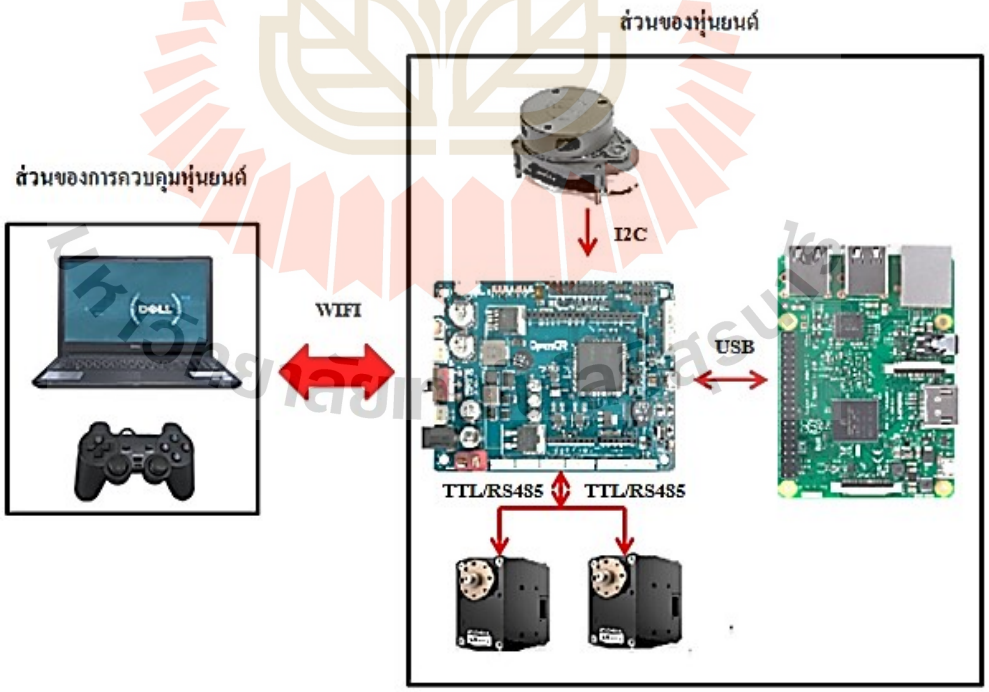
3.3 ระบบการทำงานของหุ่นยนต์สำรวจ

ส่วนประกอบของระบบการควบคุมหุ่นยนต์สำรวจจะมีอยู่ด้วยกัน 3 ส่วน ประกอบด้วย

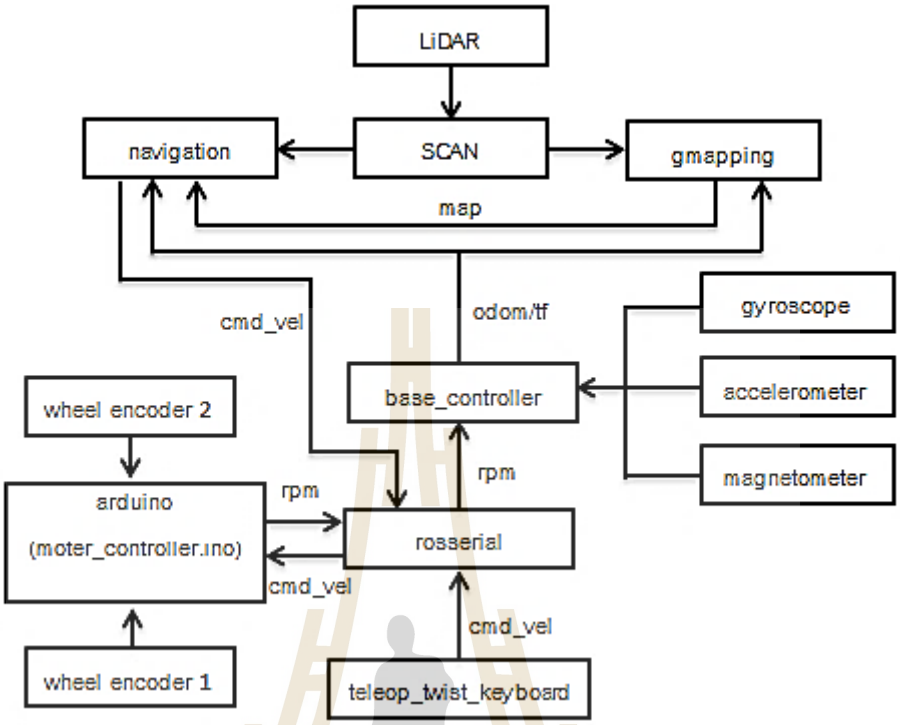
1. ส่วนของหุ่นยนต์สำรวจ จะประกอบไปด้วย เซนเซอร์ Lidar, บอร์ด OpenCR,บอร์ด Raspberry Pi, มอเตอร์ Dynamixel และแบตเตอรี่การทำงานของหุ่นยนต์สำรวจจะใช้เซนเซอร์ Lidarในการตรวจสอบสิ่งกีดขวางและทำการสำรวจพื้นที่ที่ต้องการสำรวจแล้วทำการสร้างแผนที่ โดยในส่วนของหุ่นยนต์สำรวจนี้ จะใช้บอร์ด OpenCRและบอร์ด Raspberry Pi ทำหน้าที่ในการประมวลผลพร้อมทั้งควบคุมการทำงานให้หุ่นยนต์เกิดการเคลื่อนที่ ซึ่งในการเคลื่อนที่จะใช้มอเตอร์Dynamixel ในการขับเคลื่อน

2. ส่วนของการควบคุมหุ่นยนต์สำรวจ จะประกอบไปด้วยคอมพิวเตอร์ใช้ในการควบคุมการทำงานของหุ่นยนต์สำรวจ โดยการใช้แป้นคีย์บอร์ดของคอมพิวเตอร์หรือการใช้จอยสติ๊กในการควบคุม ซึ่งในส่วนนี้จะทำการสร้างแผนที่จากข้อมูลของสิ่งแวดล้อมที่ได้รับมาเซนเซอร์ Lidarในการสร้างแผนที่จะใช้โปรแกรม Rvizที่มีอยู่ในระบบ ROS

3. ส่วนของการรับ-ส่งข้อมูลระหว่างคอมพิวเตอร์กับหุ่นยนต์สำรวจซึ่งการรับ-ส่งข้อมูลจะใช้ระบบ WIFI เป็นสื่อกลางในการรับ-ส่งข้อมูล



รูปที่ 3.3 การทำงานในส่วนต่างๆ ของหุ่นยนต์สำรวจ



รูปที่ 3.4 โฟร์ชาร์ตการทำงานของหุ่นยนต์สำรวจ

จากรูปที่ 3.4 โฟร์ชาร์ตการทำงานของหุ่นยนต์สำรวจจะมีอยู่ 2 ส่วน ในส่วนที่ 1 จะเป็นส่วนของการทำงานของระบบแบบบังคับด้วยแป้นคีย์บอร์ด ซึ่งจะเป็นการควบคุมหุ่นยนต์สำรวจผ่านผู้ใช้งาน เพื่อทำการสำรวจและการสร้างแผนที่ ในการสำรวจนั้นจะรับข้อมูลการสำรวจมาจากเซนเซอร์ Lidarที่ทำการสแกนวัตถุหรือสิ่งกีดขวางที่อยู่รอบๆ ของตัวหุ่นยนต์สำรวจ แล้วใช้โปรแกรม Rvizที่มีอยู่ในระบบ ROS สร้างการสร้างแผนที่หรือที่เรียกว่า SLAM โดยมีการควบคุมผ่านแป้นคีย์บอร์ดของคอมพิวเตอร์ ซึ่งมีการควบคุมการทำงานอยู่ 5 แบบคือ หุ่นยนต์สำรวจเดินหน้าจะกดปุ่ม W, หุ่นยนต์สำรวจถอยหลังจะกดปุ่ม X, หุ่นยนต์สำรวจเลี้ยวซ้ายจะกดปุ่ม D, หุ่นยนต์สำรวจเลี้ยวขวาจะกดปุ่ม A และสั่งให้หุ่นยนต์สำรวจหยุดการทำงานจะกดปุ่ม S

ในส่วนที่ 2 จะเป็นส่วนของการทำงานของระบบแบบอัตโนมัติอย่างย่อๆ เริ่มต้นการทำงานนั้นจะต้องมีเป้าหมายที่ต้องการเป็นลำดับแรก หลังจากนั้นทำการตรวจสอบระบบการทำงานของหุ่นยนต์สำรวจแล้วจึงปล่อยให้หุ่นยนต์สำรวจให้เคลื่อนที่ โดยระหว่างการเคลื่อนที่ของหุ่นยนต์สำรวจจะมีการทำงานอยู่ 3 ส่วนหลัก คือ 1) ตรวจหาสิ่งกีดขวาง 2) หาเส้นทางการเคลื่อนที่ 3) สั่งงานและรับค่าป้อนกลับ ทั้ง 3 ส่วนจะทำงานพร้อมกันซึ่งระบบตรวจสอบสิ่งกีดขวางไม่พบสิ่งกีดขวางตัวหุ่นยนต์สำรวจก็จะมุ่งหน้าไปยังเป้าหมายที่ต้องการเพียงอย่างเดียว แต่เมื่อไรที่ตรวจพบสิ่งกีดขวางระบบจะทำการคำนวณและหาค่าสั่งที่เหมาะสมซึ่งอาจจะเป็นการเดินหน้า ถอยหลัง

เลี้ยวซ้ายหรือเลี้ยวขวาก็ได้ขึ้นอยู่กับข้อมูลที่ได้รับมารวมทั้งหาเส้นทางการเคลื่อนที่ที่จะสามารถไปถึงเป้าหมายที่ต้องการได้สำเร็จอีกด้วย โดยการที่จะนำทางหรือหลบหลีกสิ่งกีดขวางแบบอัตโนมัติได้สำเร็จจะต้องมีตัวรับรู้ข้อมูลอย่างน้อยที่สุดคือเซนเซอร์ Lidar, เซนเซอร์ gyroscope, เซนเซอร์ accelerometer, เซนเซอร์ magnetometer,เซนเซอร์ encoder เป็นต้น

การทำงานในส่วนของการควบคุมหุ่นยนต์สำรวจด้วยคอมพิวเตอร์นั้นจะต้องทำการติดตั้ง Software ของ ROS เนื่องจากการเรียนรู้ในการทำงานของ ROS จะต้องใช้โปรแกรม Rvizและสั่ง Node ต่างๆ ค่อนข้างมาก โดยวิธีในการติดตั้งมีดังต่อไปนี้

3.3.1. การติดตั้ง PC Software

รูปที่ 3.5 การเชื่อมต่อของหุ่นยนต์สำรวจกับคอมพิวเตอร์

เป็นการเชื่อมต่อกับคอมพิวเตอร์ระยะไกล ซึ่งการควบคุมหุ่นยนต์สำรวจในรูปแบบสายจะไม่สามารถใช้งานคำสั่งในรูปแบบนี้ได้ การติดตั้ง Ubuntu บนคอมพิวเตอร์ (Desktop or Laptop PC) จะสามารถทำการควบคุมระยะไกลได้ส่วนวิธีการติดตั้งสามารถดาวน์โหลด Ubuntu 16.04 และติดตั้งได้จากลิงค์ดังนี้

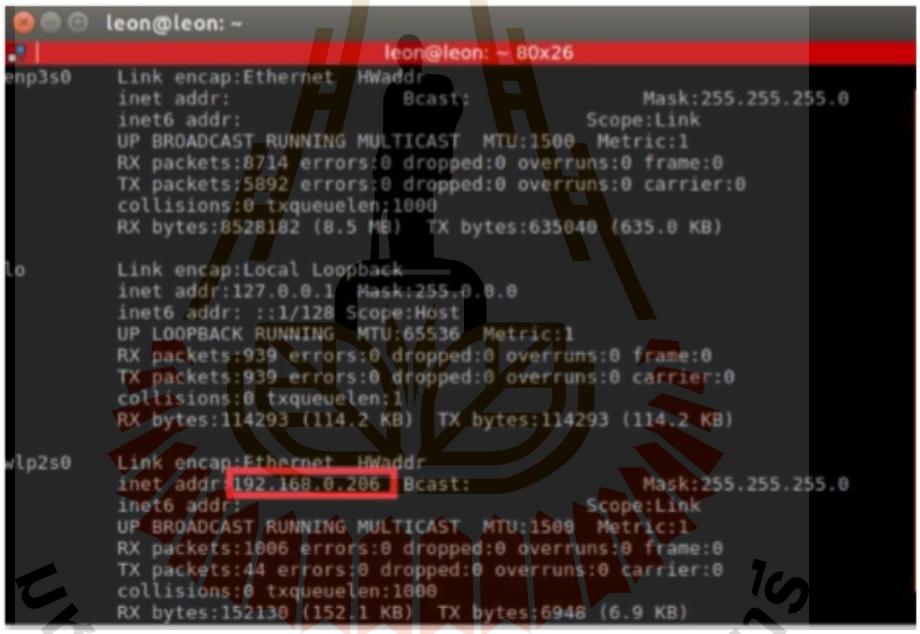
- https://www.ubuntu.com/download/desktop

3.3.2 การตั้งค่า Network

ROS จะต้องมีที่อยู่ IP เพื่อใช้ในการสื่อสารระหว่างหุ่นยนต์สำรวจกับคอมพิวเตอร์

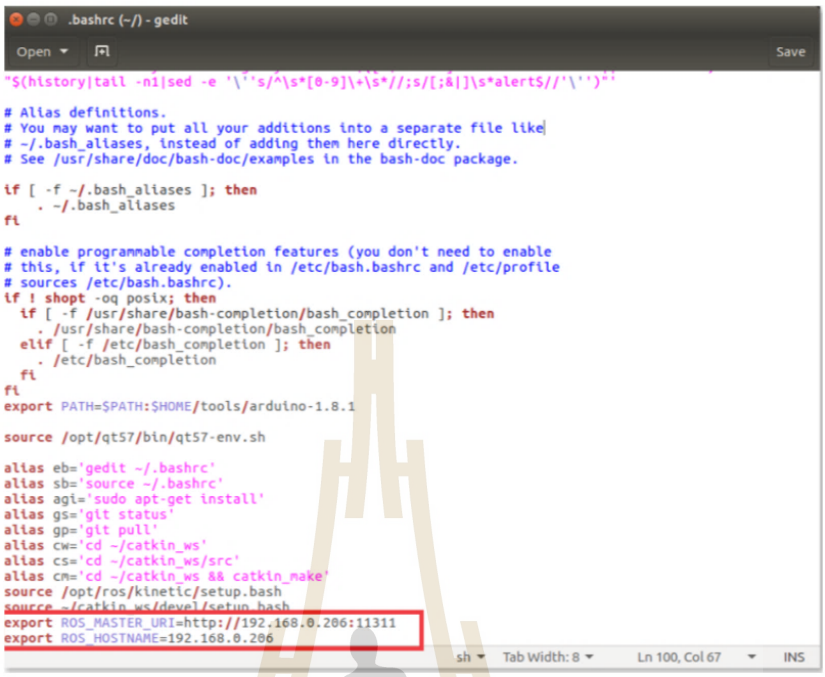
รูปที่ 3.6 การตั้งค่า Network ของหุ่นยนต์สำรวจกับคอมพิวเตอร์

จากนั้นทำการพิมพ์คำสั่งifconfigจะมีหน้าจอแสดง IP address บนจอ



รูปที่ 3.7 หน้าจอแสดง IP address

จากนั้นทำการพิมพ์คำสั่งgedit ~/.bashrcแล้วทำการปรับเปลี่ยนที่อยู่ของ Localhostกับที่อยู่ IP ที่ได้จากหน้าจอแสดง IP address



รูปที่ 3.8 Localhostกับที่อยู่ IP address

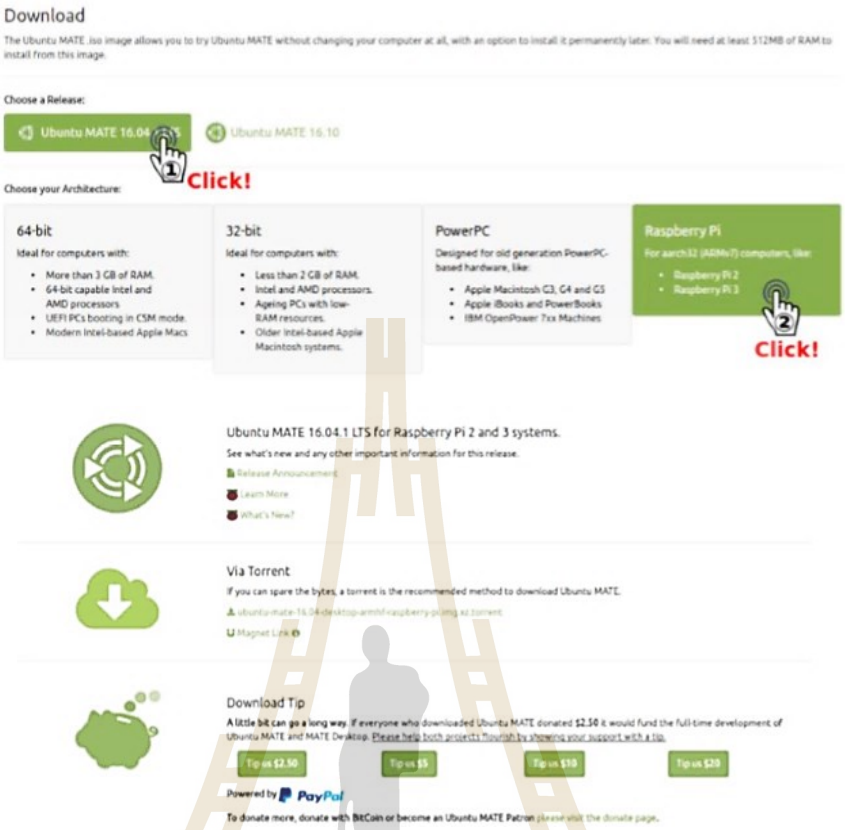
หุ่นยนต์สำรวจจะแทนที่ Localhostใน ROS_MASTER_URI ด้วย IP address จาก Remote PC Network Configuration และทำการแทน Localhostใน ROS_HOSTNAME ด้วย IP address จาก Terminal window ของ IP address ของหุ่นยนต์สำรวจ จากนั้นระบุคำสั่ง source ~/.bashrc เพื่อทำการเซฟข้อมูล IP address ของหุ่นยนต์สำรวจ

3.3.3 การติดตั้ง Software สำหรับหุ่นยนต์สำรวจ

การติดตั้ง Linux บนหุ่นยนต์สำรวจ (Raspberry Pi 3) SD card ควรมีพื้นที่ว่างอย่างน้อย 8 GB ในการติดตั้ง Linux บนหุ่นยนต์สำรวจ ซึ่งสามารถดาวโหลด Ubuntu MATE 16.04 บน Raspberry Pi 3 จากลิ้งดังต่อไปนี้

- https://ubuntu-mate.org/download/

แล้วทำตามขั้นตอนดังต่อไปนี้



รูปที่ 3.9 Ubuntu MATE 16.04 บน Raspberry Pi 3

3.3.3.1 การติดตั้ง ROS และ Packagesสำหรับหุ่นยนต์สำรวจ

การติดตั้งส่วนนี้จะสอดคล้องกับหุ่นยนต์สำรวจ (Raspberry Pi 3or Intel® Joule™) ซึ่งคำสั่งต่อไปนี้ไม่สามารถใช้กับคอมพิวเตอร์หรือ Laptop ได้ในการติดตั้งอาจใช้เวลานานถึง 2 ชั่วโมง เนื่องจากการติดตั้ง ROS และ Packages สำหรับหุ่นยนต์สำรวจจะขึ้นอยู่กับเครือข่ายที่ใช่ด้วยเช่นกันโดยการการติดตั้ง ROS สำหรับหุ่นยนต์สำรวจจะมีด้วยกัน 2 วิธีดังต่อไปนี้

วิธีที่ 1 การติดตั้ง ROS สำหรับหุ่นยนต์สำรวจสามารถทำได้ด้วยการติดตั้งด้วยไฟล์ Script ที่ได้ทำการดาวโหลดมา หลังจากทำการติดตั้ง ROS เสร็จสิ้นให้ทำการรีบูท Raspberry Pi or Intel® Joule™.

วิธีที่ 2 สามารถเริ่มจากหัวข้อ “1.2 Setup your sources.list” จนถึงหัวข้อ “1.7 Getting rosinstall” จากลิงค์การติดตั้ง ROS ดังต่อไปนี้

- http://wiki.ros.org/kinetic/Installation/Ubuntu

3.3.3.2 การตั้งค่า USB สำหรับหุ่นยนต์สำรวจ

สำหรับคำสั่งของหุ่นยนต์สำรวจที่ทำให้พอร์ต USB ทำงานร่วมกับบอร์ด OpenCRโดยการอนุญาตให้ได้รับการเข้าใช้สิทธิ์จาก Root เพื่อให้การสื่อสารส่งข้อมูลได้ด้วยความรวดเร็ว

```
$ cd ~/catkin_ws/src/turtlebot

$ sudocp ./99-turtlebot-cdc.rules /etc/udev/rules.d/

$ sudoudevadm control --reload-rules

$ sudoudevadm trigger
```

3.3.4 การติดตั้ง Software OpenCR

บอร์ด OpenCRเป็นซอฟแวร์ที่จำเป็นในการทำงานร่วมกับหุ่นยนต์สำรวจ ซึ่งคำสั่งต่อไปนี้จะใช้งานบน Ubuntu 16.04, ROS Kinetic Kame และทำการติดตั้งซอฟแวร์ OpenCRได้สมบูรณ์ซึ่งบอร์ด OpenCRจะใช้ในการควบคุม DYNAMIXELs การติดตั้งสามารถทำได้ตามขั้นตอนต่อไปนี้

3.3.4.1 การตั้งค่า ArduinoIDEจาก OpenCR

ขั้นตอนการติดตั้ง OpenCRบนคอมพิวเตอร์มีวิธีการดังต่อไปนี้

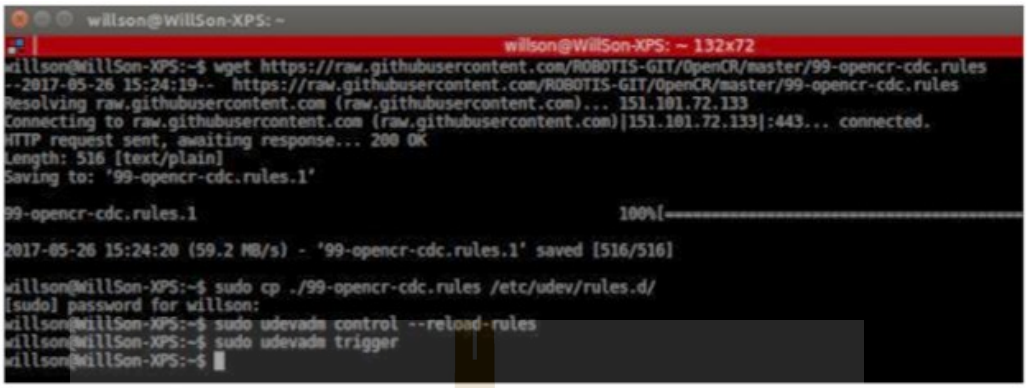
การตั้งค่า USB Port เพื่อให้ USB Port สามารถทำงานร่วมกับบอร์ด Raspberry Pi วิธีการระบุคำสั่งให้ทำการใช้งานพอร์ต OpenCR USB port สำหรับการอัพโหลด Arduino IDE program จะต้องทำผ่าน Root โดยมีวิธีดังต่อไปนี้

```
$ wget https://raw.githubusercontent.com/ROBOTIS-GIT/OpenCR/master/99-opencr-cdc.rules

$ sudocp ./99-opencr-cdc.rules /etc/udev/rules.d/

$ sudoudevadm control --reload-rules

$ sudoudevadm trigger
```



รูปที่ 3.10 การใช้งานพอร์ต OpenCR USB port สำหรับการอัพโหลด Arduino IDE program

3.3.4.2 การตั้งค่า Compiler

บอร์ดOpenCR library จะมีอยู่ในรูปแบบ 32-bit และ 64-bit แต่คอมพิวเตอร์ต้องการ 32-bit สำหรับ Compiler เพื่อใช้งานร่วมกับบอร์ดของ Arduino IDEโดยมีวิธีดังต่อไปนี้

$ sudo apt-get install libncurses5-dev:i386

3.3.4.3 การติดตั้ง Arduino IDE

ทำการดาวน์โหลด Arduino IDE จากเว็บไซต์ของ Arduinoแล้วทำการติดตั้งโปรแกรม OpenCRเพื่อให้ซัปพอร์ต Arduino IDE 1.6.12 หรือ (OpenCRมีการทดสอบบน Arduino IDE 1.8.1.) จากเว็บไซต์ดังต่อไปนี้

- https://www.arduino.cc/en/Main/Software

เมื่อทำการดาวน์โหลดไฟล์เสร็จสิ้นจากนั้นทำการระบุคำสั่งดังต่อไปนี้ลงในโฟลเดอร์ Arduino IDE เพื่อทำการลงโปรแกรม

$ cd ~/tools/arduino-1.8.1

$ ./install.sh

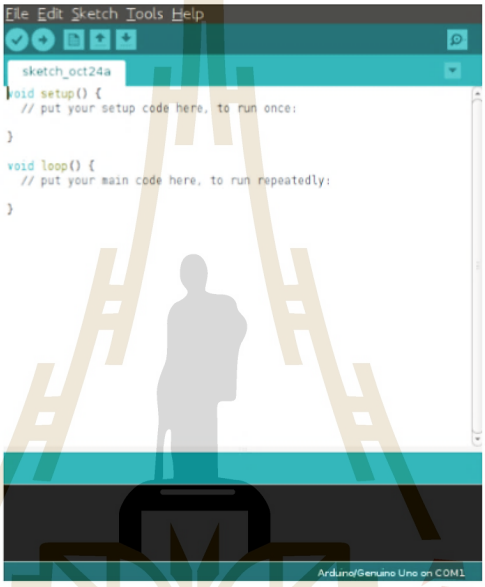
เมื่อลงโปรแกรม Arduino IDE แล้วทำการระบุคำสั่ง gedit ~/.bashrcเพื่อทำการ Bashrcในไฟล์ด้านล่างสุดของ Code ดังต่อไปนี้

$ export PATH=$PATH:$HOME/tools/arduino-1.8.1

หลังจากเสร็จสิ้นให้ทำการ Source ไฟล์ Bashrcด้วย source ~/.bashrcเพื่อให้โปรแกรมสามารถทำงานได้

3.3.4.4 วิธีการ RUN Arduino IDE

สำหรับวิธีการ RUN Arduino IDE เพื่อการใช้งาน Arduino IDE ให้ป้อนคำสั่ง Arduino บน Terminal หากโปรแกรมติดตั้งถูกต้องจะมี GUI ดังรูปที่ 3.11ดังต่อไปนี้

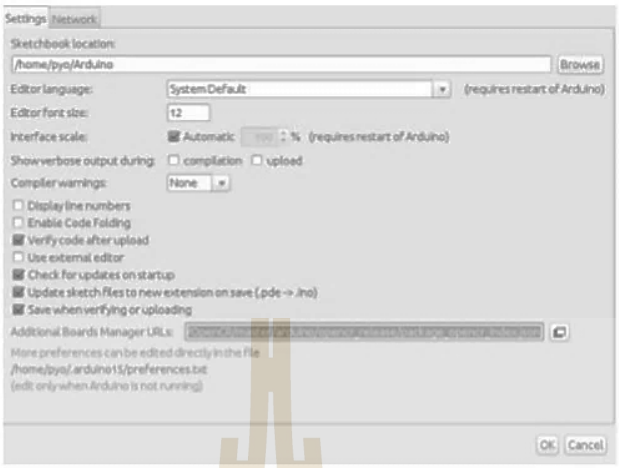


รูปที่ 3.11 GUI ของArduino IDE

3.3.4.5 วิธีการ Porting OpenCR to Arduino IDE

การตั้งค่าเมื่อเปิดโปรแกรม Arduino IDE ให้ทำงานแล้วกดไฟล์ File ⟶ Preferences จาก Menu ของโปรแกรมให้คัดลอกลิงค์ URLs จากด้านล่างดังต่อไปนี้ (อาจใช้เวลาประมาณ 20 นาที)จะได้ดังรูปที่ 3.12

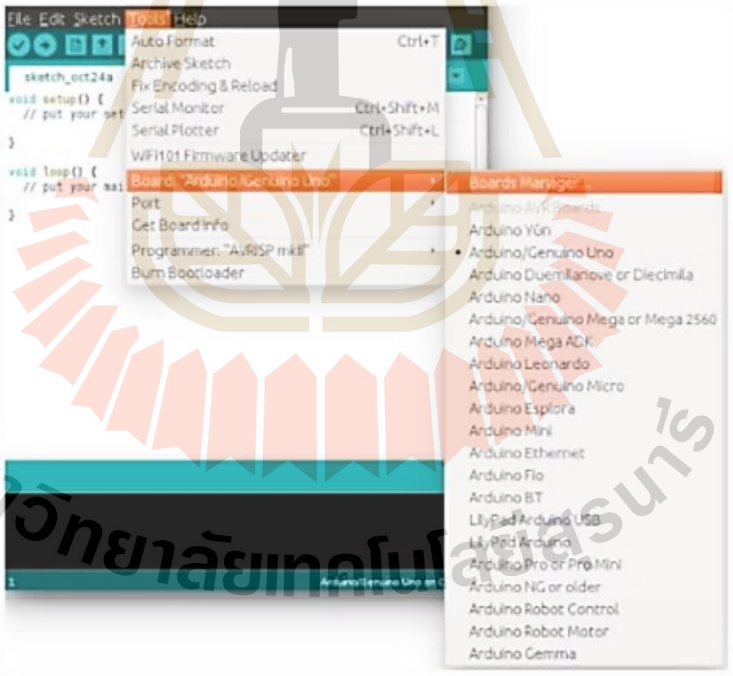
https://raw.githubusercontent.com/ROBOTIS-GIT/OpenCR/master/arduino/opencr_release/package_opencr_index.json



รูปที่ 3.12 ลิงค์ URLs ของ Arduino IDE

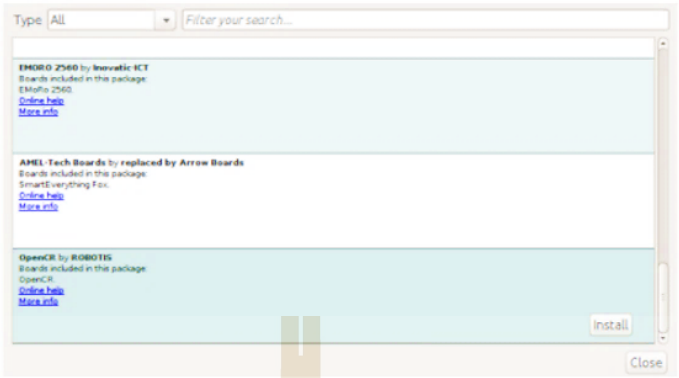
3.3.4.6 การติดตั้ง OpenCR Package จาก Boards Manager

เมื่อเปิดโปรแกรม Arduino IDE ให้ทำงานแล้วไปที่ Tools ⟶ Board ⟶ Boards Manager



รูปที่ 3.13 การติดตั้ง OpenCR Package จาก Boards Manager

จะได้ดังรูปที่ 3.14แล้วทำการเลือกประเภท OpenCRในกรอบที่ปรากฏถ้าพบเจอแพ็คเกจแล้วทำการติดตั้งทันทีหากติดตั้งแพ็คเกจเสร็จสิ้นและมีแพ็คเกจอื่นจะปรากฏขึ้นต่อไปสามารถลงเพิ่มเติมได้



รูปที่ 3.14 การติดตั้ง OpenCR Package จาก Boards Manager

3.3.4.7 การตั้งค่า Port

ส่วนนี้จะแสดงวิธีการติดตั้งพอร์ต USB สำหรับอัพโหลดโปรแกรมบนบอร์ด OpenCRเพื่อเชื่อมต่อกับคอมพิวเตอร์ โดยการกดเลือก Tools ⟶ Port ⟶ /dev/ttyACM0 หากค่าที่แสดงเป็นศูนย์ /dev/ttyACM0 อาจมีความแตกต่างระหว่าง USB หรือการเชื่อมต่อที่ผิดพลาดดังรูปที่ 3.15

รูปที่ 3.15 วิธีการติดตั้งพอร์ต USB

3.3.5 การตั้งค่าบอร์ด OpenCR Firmware สำหรับ ROS

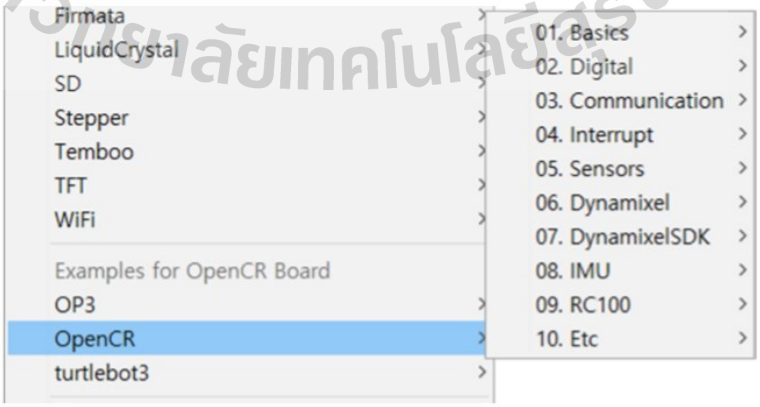
การตั้งค่าบอร์ด OpenCR Firmware สำหรับ ROS มีความจำเป็นที่จะต้องเพิ่ม Firmware สำหรับการทำงานของหุ่นยนต์สำรวจในบอร์ด OpenCR โดยการใช้เฟิร์มแวร์OpenCR

สำหรับ ROS เพื่อทำการควบคุม DYNAMIXEL และ Sensors ใน ROS จะมีเฟิร์มแวร์ หากเกิดข้อผิดพลาดขณะอัพโหลดเฟิร์มแวร์ให้ไปที่ Tools ⟶ Port และตรวจสอบว่ามีการเลือกพอร์ตที่ถูกต้องหรือไม่ แล้วทำการกดปุ่ม Reset บนบอร์ด OpenCRแล้วลองอัพโหลดเฟิร์มแวร์อีกครั้งดังรูปที่ 3..16

รูปที่ 3.16 อัพโหลด Firmware ของหุ่นยนต์สำรวจในบอร์ด OpenCR

3.3.5.1 การทดสอบบอร์ด OpenCR

เมื่อเพิ่มแพ็คเกจสำหรับบอร์ด OpenCRเข้ากับ Arduino IDE วิธีการทำงานของบอร์ด Arduinoสามารถใช้ตัวอย่างซอฟแวร์ที่มีอยู่ในบอร์ด OpenCRโดยไปที่เมนู File ⟶ Examples มีตัวอย่างมากมายที่สามารถใช้ในการควบคุมและเรียนรู้วิธีการใช้ฮาร์ดแวร์ที่เชื่อมต่อกับบอร์ด OpenCRดังรูปที่ 3.17ตัวอย่างการใช้งานบอร์ด OpenCR



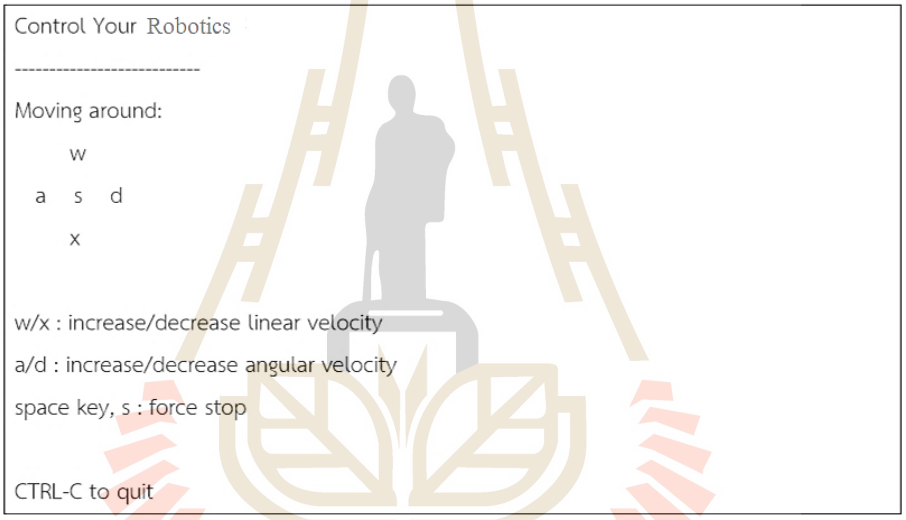
รูปที่ 3.17 ตัวอย่างการใช้งานบอร์ด OpenCR

3.4 การควบคุมการทำงานของหุ่นยนต์สำรวจ

การควบคุมการทำงานของหุ่นยนต์สำรวจจะทำงานบนเครื่องคอมพิวเตอร์สำหรับหุ่นยนต์สำรวจสามารถควบคุมการทำงานโดยอุปกรณ์ต่างๆ มีการทดสอบอุปกรณ์ไร้สายต่างๆ เช่น PS3, XBOX 360, ROBOTIS RC100 เป็นต้น ซึ่งในการควบคุมการทำงานของหุ่นยนต์สำรวจจะใช้ Keyboard ในการควบคุมการเคลื่อนที่โดยการพิมพ์คำสั่งดังต่อไปนี้

```
$ roslaunchturtlebot_teleopturtlebot_teleop_key.launch
```

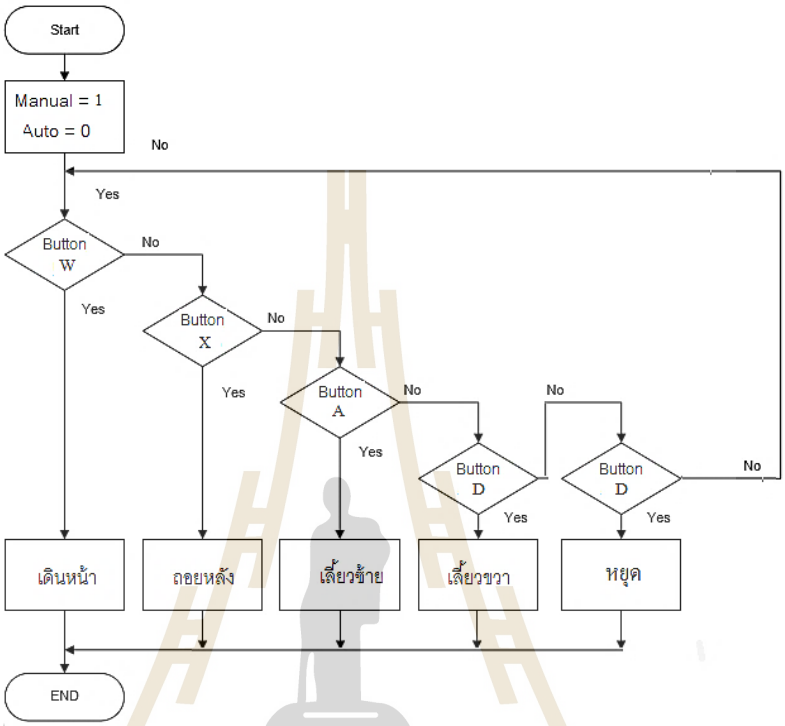
การเปิดไฟล์สำหรับการทดสอบการทำงาน teleoperationหากโปรแกรมถูกเปิดใช้งานคำสั่งแล้วจะปรากฏขึ้นที่หน้าต่าง Terminal ดังรูปที่ 3.18



รูปที่ 3.18 การควบคุมการทำงานของหุ่นยนต์สำรวจ

สำหรับการควบคุมการทำงานของหุ่นยนต์สำรวจจะมีรูปแบบการควบคุมอยู่ 2 แบบ คือ แบบบังคับด้วยผู้ใช้งานและแบบอัตโนมัติในส่วนของการควบคุมแบบบังคับด้วยผู้ใช้งานจะเป็นส่วนการควบคุมหุ่นยนต์สำรวจในการสร้างแผนที่ ซึ่งจะเป็นการควบคุมผ่านผู้ใช้งานผ่านแป้นคีย์บอร์ดของคอมพิวเตอร์ โดยการควบคุมการทำงานของหุ่นยนต์สำรวจจะมีอยู่ 5 แบบคือ หุ่นยนต์สำรวจเดินหน้าจะกดปุ่ม W, หุ่นยนต์สำรวจถอยหลังจะกดปุ่ม X, หุ่นยนต์สำรวจเลี้ยวซ้ายจะกดปุ่ม D, หุ่นยนต์สำรวจเลี้ยวขวาจะกดปุ่ม A และสั่งให้หุ่นยนต์สำรวจหยุดการทำงานจะกดปุ่ม S ส่วนของการควบคุมแบบอัตโนมัติจะเป็นส่วนการควบคุมหุ่นยนต์สำรวจโดยการใช้แผนที่ที่สร้างเสร็จแล้วมาทำการควบคุมแบบอัตโนมัติโดยการใช้โปรแกรม Rvizที่มีอยู่ในระบบ ROS ด้วยการกดปุ่ม 2D Navigation จะเห็นเป็นภาพลูกศรอยู่ด้านบนของโปรแกรมแล้วนำเมาส์ไปคลิ๊กยังตำแหน่งและ

ทิศทางที่ต้องการในแผนที่ที่ได้สร้างไว้แล้วหุ่นยนต์สำรวจจะทำการเคลื่อนที่แบบอัตโนมัติไปยังตำแหน่งและทิศทางตามที่ต้องการ



รูปที่ 3.19 โฟร์ชาร์ตการสั่งงานด้วยแป้นคีย์บอร์ด

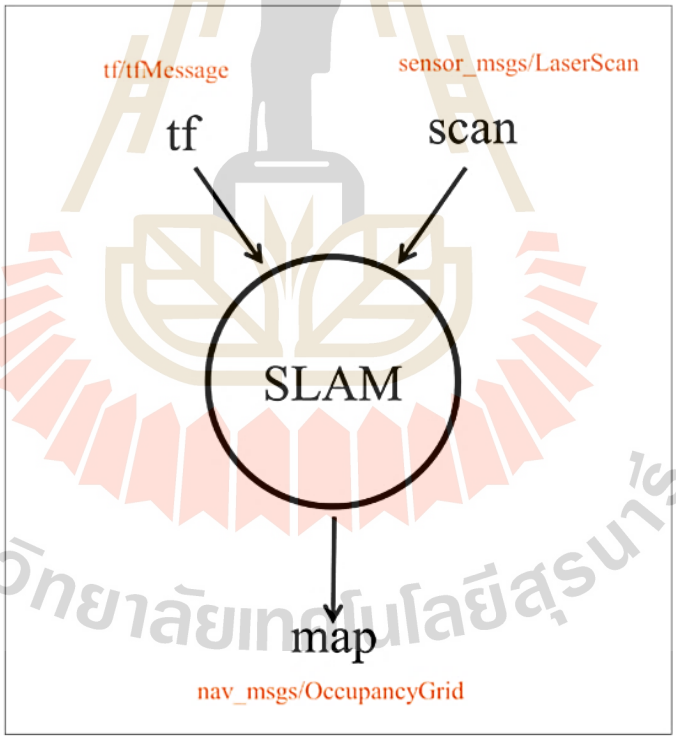
รูปที่ 3.20 การสั่งงานด้วยกดปุ่ม 2D Navigation

3.5 วิธีการสร้าง SLAM สำหรับหุ่นยนต์สำรวจ

สำหรับ ROS มีไลบรารีที่เป็น SLAM หลายชุด เช่น Slam Gmapping, Hector slam, Slam kartoเป็นต้น สำหรับวิทยานิพนธ์เล่มนี้จะใช้ Slam Gmapping โดยหลักการสร้างแผนที่ด้วย Gmappingพบว่าไลบรารี Gmapping ของ ROS คือการสร้างการห่อหุ้มไลบรารีที่ชื่อว่า Gmapping จาก OpenSLAM

3.5.1 ขั้นตอนในการทำ SLAM ของหุ่นยนต์สำรวจ

สำหรับหุ่นยนต์สำรวจจะใช้หลักการในการแปลงข้อมูลของจุดที่ต้องการ เช่น ระยะทางของสิ่งกีดขวางที่วัดได้จากเซนเซอร์ Lidarจุดหนึ่ง ซึ่งระยะทางที่วัดด้วยเซนเซอร์ Lidar เรียกว่า Scan ใน ROS และข้อมูลตำแหน่ง (position + orientation) จะเรียกว่า tf (transform) ดังแสดงในรูปที่ 3.21วิธีการเรียกใช้ SLAM โดยใช้ข้อมูลสองอย่างคือ Scan และ tfสำหรับสร้างแผนที่ที่เราต้องการ



รูปที่ 3.21 ข้อมูล Scan และ tf สำหรับการสร้างแผนที่

หุ่นยนต์สำรวจจะเริ่มทำการสร้างแผนที่ Slam_gmappingโดยมีขั้นตอนการทำงานดังต่อไปนี้

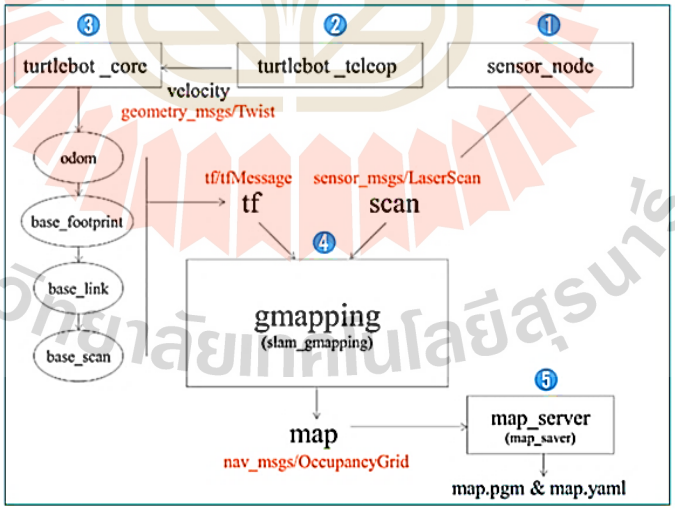
1. ข้อมูลที่ได้จากเซนเซอร์ Lidarได้แก่ Topic ที่ชื่อว่า Scan โดยมี Message เป็นประเภท Sensor_msgs/LidarScanจะทำการสแกนสิ่งแวดล้อมที่อยู่รอบๆ ตัวหุ่นยนต์สำรวจ เพื่อเป็นข้อมูลในการสร้างแผนที่

2. การควบคุมการทำงานหุ่นยนต์สำรวจให้เคลื่อนที่ สามารถกำหนดความเร็วและทิศทางในการเคลื่อนที่ของหุ่นยนต์สำรวจได้จากแป้นคีย์บอร์ดของคอมพิวเตอร์

3. เมื่อได้ข้อมูลการเคลื่อนที่ของหุ่นยนต์สำรวจก็จะทำการส่งข้อมูลของการเคลื่อนที่ เพื่อเป็นข้อมูลในการสร้างแผนที่ โดยมี Messageประเภท tfซึ่งจะเป็นการอธิบาย frame ต่างๆ ของหุ่นยนต์และเซนเซอร์ Lidar

4. ในการสร้างแผนที่จะมี Node ที่ชื่อ Rvizที่มีอยู่ในระบบ ROS เพื่อใช้ในการจำลองการทำงานของหุ่นยนต์สำรวจ เมื่อได้ข้อมูลจากการเคลื่อนที่ของหุ่นยนต์สำรวจและข้อมูลจากการสแกนสิ่งแวดล้อมที่อยู่รอบๆ ตัวหุ่นยนต์สำรวจ ตั้งแต่เริ่มต้นในการสำรวจจนกระทั้งทำการสำรวจเสร็จก็จะได้แผนที่ ซึ่งแผนที่ที่ได้จะเป็นแบบ Gmapping

5. เมื่อทำการสำรวจเสร็จก็จะทำการเซฟไฟล์แผนที่ โดยไฟล์จะมี 2 ไฟล์ ไฟล์จะมีนามสกุล pgmและ yamlซึ่งไฟล์ pgmจะเป็นรูปภาพส่วนไฟล์ yamlจะเป็นการอธิบายแผนที่



รูปที่ 3.22 โฟร์ชาร์ตสำหรับหุ่นยนต์สำรวจในการสร้างแผนที่

สำหรับสิ่งที่ได้จากการทำงานของ Gmappingจะประกอบไปด้วย Topic ต่างดังต่อไปนี้

1.Topic ที่ชื่อว่า map_metadataโดยมี Message เป็นประเภท nav_msgs/MapMetaData

2.Topic ที่ชื่อว่า map โดยมี Message ประเภท nav_msgs/OccpancyGrid

3.Topic ที่ชื่อว่า entropy มีประเภทเป็น std_msgs/Float64 ซึ่งเป็นค่าที่บอกถึงความไม่แน่นอนของข้อมูล ค่ายิ่งมากยิ่งมีความไม่แน่นอนสูง

4.ค่าตำแหน่งของหุ่นยนต์สำรวจใน Map frame ซึ่งได้มาจากการแปลงจาก frame ชื่อ map ไปสู่ odom

นอกจากนั้น Gmappingยังมี service ได้แก่ service ชื่อ dynamic_mapซึ่งจะใช้ Message เป็นประเภท nav_msgs/GetMap และยังมี Node และ Topic ต่างๆ ดังรูปที่ 3.23

รูปที่ 3.23 Nodes และ topics สำหรับหุ่นยนต์สร้างแผนที่

3.5.2 การหาตำแหน่งในแผนที่ด้วย AMCL

เมื่อหุ่นยนต์สำรวจสร้างแผนที่เสร็จแล้ว จะทำการเก็บไฟล์ไว้และให้ Map Server เรียกไฟล์แผนที่ที่ต้องการขึ้นมาใช้งาน โดยข้อมูลที่ผู้ใช้ต้องการจาก Map Server สามารถเรียกใช้แผนที่ไปที่ Server จากนั้น Map Server จะทำการจัดส่งข้อมูลแผนที่ไปให้ สำหรับไลบรารี Slam_Gmappingมีจุดเด่นในการสร้างแผนที่ ซึ่งในการเคลื่อนที่ของหุ่นยนต์สำรวจในท่อจะใช้ SLAM เพียงอย่างเดียวไม่พอสำหรับการนำทางให้หุ่นยนต์สำรวจ สิ่งที่ต้องมีเพิ่มเติมในการทำงานของระบบ ROS SLAM คือการรู้ตำแหน่งของหุ่นยนต์สำรวจด้วยไลบรารี AMCL ซึ่งจะทำให้รู้ตำแหน่งในการเคลื่อนที่ของหุ่นยนต์สำรวจและทิศทางของหุ่นยนต์สำรวจในแผนที่ สำหรับการนำทางด้วยไลบรารี AMCL มีวิธีการดังต่อไปนี้

3.5.2.1 AMCL

AMCL (Adaptive Monte Carlo Localization) เป็นไลบรารีที่ทำหน้าที่ในการหาตำแหน่ง ณ ปัจจุบันของหุ่นยนต์ Dieter Fox ได้นำเสนอด้วยอัลกอริทึมที่ดัดแปลงมาจาก

Monte Carlo Localizationซึ่งใช้หลักการทำงานเชิงสถิติ ที่เรียกว่า Kalman Filter ซึ่งใช้หลักการของ Genetic Algorithm คืออัลกอริทึมที่เรียนแบบหลักการคัดสรรของธรรมชาติ

ไลบรารีAMCL มีองค์ประกอบต่างๆ ดังต่อไปนี้

1. แผนที่ จะขึ้นอยู่กับพารามิเตอร์ชื่อว่าuse_map_topicหากเป็น True จะใช้ Topic ถ้าหากเป็น False จะใช้การเรียกไปที่ Map Server

2.Topic ที่ Subscribe ได้แก่

-Scan ซึ่งเป็นข้อมูลของระยะสิ่งกีดขวางจากเซนเซอร์ Lidar

- tf คือตัว TF ที่บอกความสัมพันธ์ระหว่างระบบพิกัดต่างๆ เช่น base_link, laser_frameเป็นต้น

- initialposeคือจุดเริ่มต้นของหุ่นยนต์สำรวจในแผนที่เพื่อที่จะทำการเริ่มต้นค่าของ Kalman Filter

- map ใช้ในกรณีที่กำหนดค่าพารามิเตอร์ชื่อ use_map_topicให้เป็น True เท่านั้น

3.Topicที่จะ Publisher ได้แก่

- amcl_poseได้แก่ตำแหน่งของหุ่นยนต์สำรวจในแผนที่ที่เกิดจากการคาดการณ์ AMCL

- Kalman cloud คือเซ็ตของค่าตำแหน่งของหุ่นยนต์ที่เกิดจากการสุ่มค่าของ Filter

- tfจะ broadcast ความสัมพันธ์ระหว่าง frame ของ odomกับ map

4. Service ที่เรียกใช้ได้

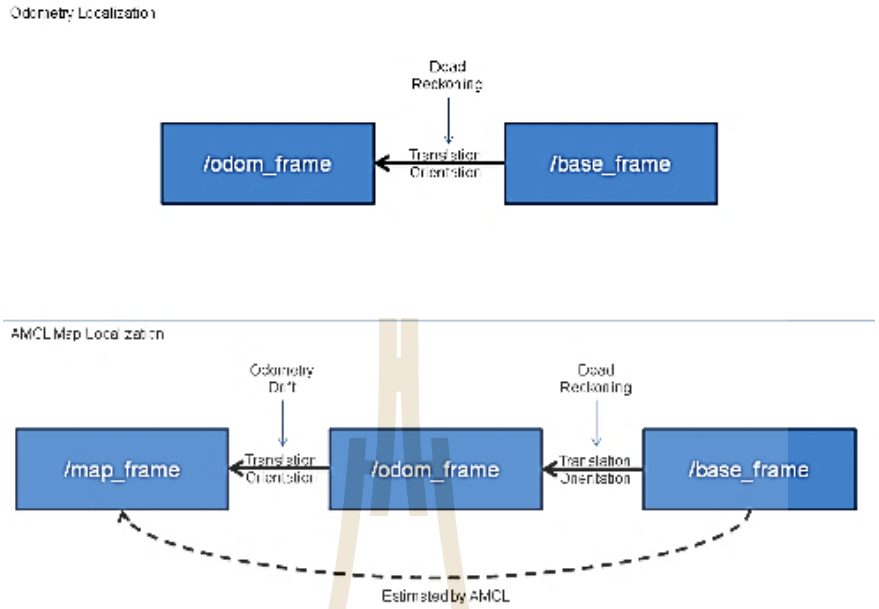
- global_localizationใช้สำหรับการเริ่มต้นการหาตำแหน่งในแบบ Global

- request_nomotion_updateใช้ในกรณีที่ต้องการกำหนดค่าของ Kalman

5. Service ที่ Node จะเรียก

- static_mapของแผนที่โดยที่ Node ต้องการเรียกใช้แผนที่จาก Map Server ผ่านทาง Service

โดยมีวิธีการแปลง frame ที่ AMCL เมื่อทำการประมวลผลจะทำการ broadcast ตัว TF เพื่อแสดงความสัมพันธ์ระหว่าง /base_frameกับ /map_frame ดังรูปที่ 3.24



รูปที่ 3.24 การแปลง frame ที่ได้จาก AMCL

ในส่วนของ Odometryการ transform ของ frame จะได้เพียงความสัมพันธ์ระหว่าง /base_frameกับ /odom_frame ซึ่งเป็นเรื่องปกติในการทำงานของระบบเพราะบอร์ด OpenCRและ มอเตอร์ DYNAMIXE นั้นสามารถทำได้เพียงการเปลี่ยนแปลงตำแหน่งของตัวหุ่นยนต์สำรวจ แต่ AMCL จะทำหน้าที่เชื่อมต่อกับ frame ของ map ดังนั้นข้อดีของ AMCL คือการแก้ความผิดพลาดที่เกิดจาก Odometry Drift ทำให้รู้ตำแหน่งและทิศทางของหุ่นยนต์สำรวจในแผนที่ได้อย่างถูกต้องซึ่งจะมีความสำคัญในการทำ Navigation ของหุ่นยนต์สำรวจต่อไป

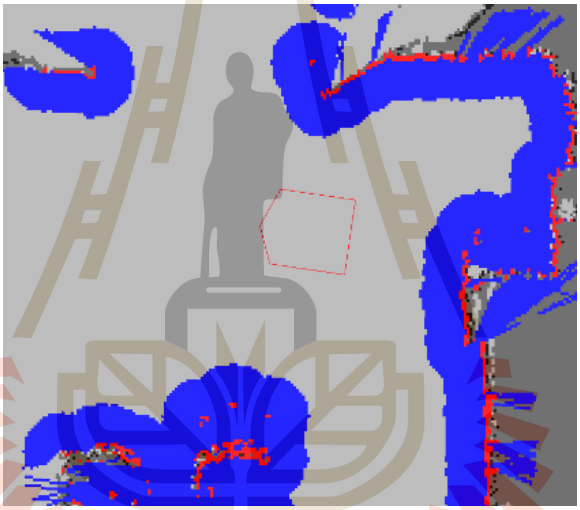
3.6 วิธีการ Navigation สำหรับหุ่นยนต์สำรวจ

เมื่อสามารถรู้ตำแหน่งของหุ่นยนต์สำรวจแล้ว ขั้นตอนของ ROS Navigation คือการรู้ถึงลักษณะของสิ่งกีดขวางโดยรอบ เพื่อให้หุ่นยนต์สำรวจนั้นสามารถค้นหาเส้นทางในการเคลื่อนที่โดยไม่ติดขัดหรือชนกับสิ่งกีดขวางและสามารถไปถึงเป้าหมายได้ตามที่ต้องการ การที่หุ่นยนต์สำรวจจะรู้ถึงลักษณะและสภาพของสิ่งกีดขวางนั้น ROS จะใช้เทคนิคที่เรียกว่า Costmapหรือแผนที่ที่ได้ทำการสร้างไว้แล้วซึ่งจะทำให้สามารถเคลื่อนที่ไปจุดต่างๆ ได้โดยการใช้เวลาที่น้อยลงและสามารถหาเส้นทางการเคลื่อนที่ที่เร็วขึ้น และเมื่อรู้ถึงตำแหน่งของสิ่งกีดขวางโดยรอบแล้ว สามารถวางแผนในการเคลื่อนที่สำรวจแบบอัตโนมัติได้ด้วย อัลกอริทึมการวางแผนหรือ Planner ใน Move Base โดยมีขอบเขตของการการทำงานอยู่ 2 ระดับได้แก่ระดับภาพรวม Global Planner และระดับใกล้ Local Planner โดยที่ Planner ทั้งสองประเภทจะทำการคำนวณเพื่อสร้างแผนของเส้นทางแล้ว

ทำการส่ง Topic ที่ชื่อว่า cmd_velออกมาเพื่อนำไปควบคุม Base Controller หรือหุ่นยนต์สำรวจให้เกิดการเคลื่อนที่

3.6.1 หลักการของ Costmap

การเคลื่อนที่ของหุ่นยนต์สำรวจที่สามารถหลบหลีกสิ่งกีดขวางได้นั้นเป็นปัจจัยพื้นฐานในการทำงานของหุ่นยนต์สำรวจอย่างหนึ่ง ซึ่งอัลกอริทึมในการหลบหลีกสิ่งกีดขวางสำหรับ ROS จะใช้หลักการทำงานของ Costmapโดยการคำนวณ ระยะอันตรายที่ยื่นออกมาของสิ่งกีดขวาง (inflate) การคำนวณนี้จะมีแบบ 2 มิติ และ 3 มิติ ซึ่งในวิทยานิพนธ์นี้จะใช้การคำนวณแบบ 2 มิติ ดังรูปที่ 3.25หลักการคำนวณของ Costmapโดยส่วนที่ยื่นออกมาจากสิ่งกีดขวางคือส่วนของ Inflation ซึ่งจะใช้ในการคำนวณระยะอันตรายหรือระยะที่ปลอดภัยของหุ่นยนต์สำรวจ



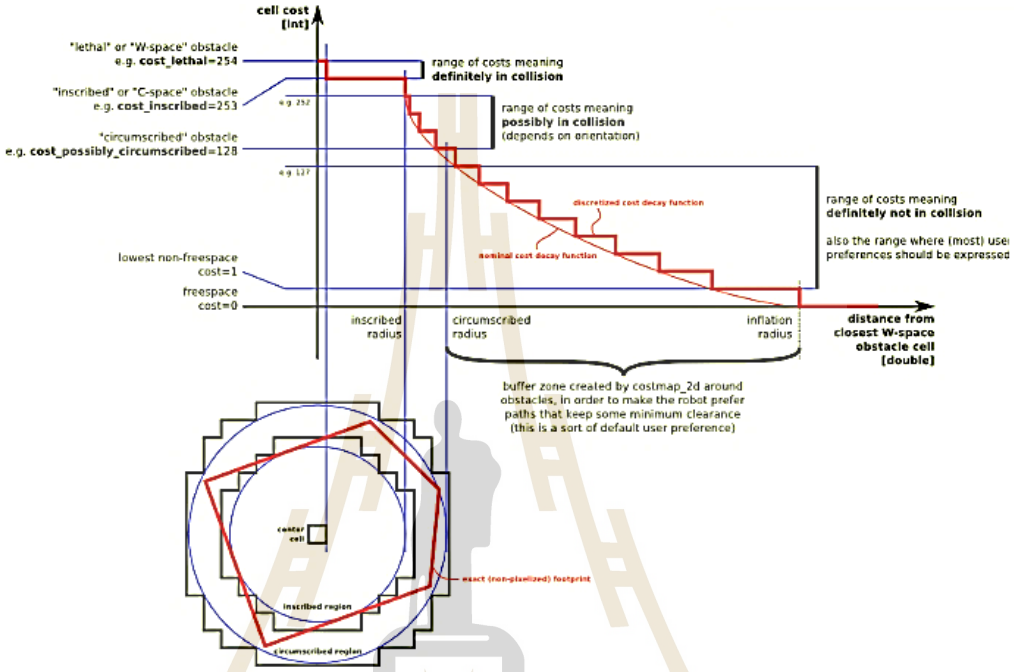
รูปที่ 3.25 หลักการคำนวณของ Costmap

จากรูปที่ 2.23 ส่วนที่เป็นรูป 5 เหลี่ยมหรือ Polygon นั่นคือส่วนล่างสุดของหุ่นยนต์สำรวจ โดยที่ Costmapคือบริเวณโยรอบ ส่วน Inflation คือรูปคล้ายก้อนเมฆที่ยื่นออกมา สีแดงหมายถึงระยะอันตรายที่ไม่สามารถให้หุ่นยนต์สำรวจผ่านเข้าไปได้ ส่วนของสีน้ำเงินเป็นส่วนที่พอรับได้และส่วนที่ไม่ใช้ศูนย์กลางยังพอสามารถผ่านไปได้ การทำงานของ Costmapจะอ่านค่าจาก Topic ของเซนเซอร์ Lidarเพื่อนำข้อมูลของเซนเซอร์ที่อ่านได้มาคำนวณหาเส้นทางในการเคลื่อนที่ในแผนที่

3.6.1.1 การ Inflation

การเลือกเส้นทางของ Planner จะนำข้อมูล Inflation จาก Costmapเข้ามาคำนวณ โดยเส้นทางที่หุ่นยนต์สำรวจเคลื่อนที่จะทำการเลือกเส้นทางที่ไม่ทำให้จุดศูนย์กลางของหุ่นยนต์

สำรวจผ่านไปในส่วนของ Inflation แต่หากต้องการเคลื่อนที่เข้าไปใกล้หรือลักษณะทางที่แคบก็สามารถให้บางส่วนของหุ่นยนต์สำรวจสามารถเคลื่อนที่ผ่านไปได้ ดีกว่าไม่สามารถเคลื่อนที่ผ่านเข้าไปได้ ดังภาพที่ 3.26อธิบายหลักการของ Inflation



รูปที่ 3.26 อธิบายหลักการของ Inflation

จากรูปที่ 3.24 กราฟในแนวแกน Y คือค่า Cost ที่สามารถคำนวณได้ โดยใช้เป็นฐานในการคำนวณได้แก่ cost_lethal, cost_inscribed, cost_possibly_circumscribed, freespaceโดยมีความหมายดังต่อไปนี้

- cost_lethalคือ ค่าตั้งแต่ cost_inscribed ถึงค่า cost_lethal จะเป็นค่าที่อันตรายที่หุ่นยนต์สำรวจห้ามผ่านโดยเด็ดขาด ถ้าหากมีการเคลื่อนที่เข้าไปของหุ่นยนต์สำรวจจะทำให้ไม่สามารถเคลื่อนที่ไปต่อได้

- cost_inscribedคือ ค่าตั้งแต่ cost_possibly_circumscribed ถึงค่า cost_inscribed อาจจะมีโอกาสเกิดการชนกับสิ่งกีดขวางของหุ่นยนต์สำรวจหรืออาจจะทำให้ติดขัดในการเคลื่อนที่ โดยขึ้นอยู่กับทิศทางในการเคลื่อนที่ของหุ่นยนต์สำรวจ

- cost_possibly_circumscribedคือ ค่าตั้งแต่freespaceถึงค่าcost_possibly_circumscribedจะไม่เกิดการชนของหุ่นยนต์สำรวจอย่างแน่นอน

โดยทั่วไปแล้วการทำงานของหุ่นยนต์สำรวจจะอาศัยหลักการของ Planner ในการพยายามหาเส้นทางการเคลื่อนที่ที่ใกล้ที่สุดและไกลจากส่วนที่อาจจะทำให้หุ่นยนต์สำรวจติดขัดซึ่งเรียกการเคลื่อนที่แบบนี้ว่าเหมาะสมที่สุด

3.6.1.2 การ Mark และ Clear

การเก็บข้อมูลของ Costmapจะเป็นในรูปแบบของ cell ขนาดเล็กที่อยู่ในแผนที่ ก่อนที่ Costmapจะสร้าง Inflation จะมีขั้นตอนในการ Clear และทำการ Mark โดยรับข้อมูลจากเซนเซอร์ Lidarที่อยู่บนตัวหุ่นยนต์สำรวจ หากพบสิ่งกีดขวางที่อยู่รอบๆ ตัวหุ่นยนต์สำรวจก็จะทำการ Mark ส่วนในการ Clear จะทำการตรวจสอบบริเวณรอบๆ แล้วจะมีเวลา Clear ซึ่งในการ Clear จะไม่ได้ทำบ่อยเท่ากับการ Mark และถ้าหากพบว่าตำแหน่งเดิมที่เคยมีสิ่งกีดขวางหรือไม่มีสิ่งกีดขวางก็จะทำการ Clear ซึ่งระยะในการ Clear และ Mark สามารถกำหนดได้ในพารามิเตอร์การทำงานของหุ่นยนต์สำรวจ

3.6.1.3 ข้อจำกัดการทำงานของ TF

Costmapต้องการข้อมูลที่มีความจำเป็นของ frame ได้แก่ map, base_link และข้อมูลแบบ Real time (Broadcast) ซึ่งหาก Broadcast ไม่เป็นแบบ Real time ตามที่กำหนดการเคลื่อนที่ Navigation Stack จะหยุดการทำงานของหุ่นยนต์สำรวจ โดยการส่งข้อมูลหรือการทำงานของหุ่นยนต์ขึ้นอยู่กับเสปกของคอมพิวเตอร์ที่นำมาใช้ในการทำงานก็มีผลต่อเวลาในการส่งข้อมูล

3.6.1.4 การใช้ Layer ของ Costmap

Costmapจะมีการใช้งาน Layer ที่ว่างทับซ้อนกันได้ ในการทำงานของแต่ละ Layer จะมีหน้าที่ในการเก็บข้อมูลและทำการประมวลผลที่แต่ต่างกัน ดังตัวอย่างต่อไปนี้

- static map layer เป็น layer ที่สร้างจากแผนที่ตายตัว เช่นแผนที่ที่เก็บไว้จาก SLAM ส่วนใหญ่มักใช้เป็น Global Costmap
- obstacle layer เป็น layer ที่เก็บข้อมูลของสิ่งกีดขวางในแผนที่
- Inflation layer เป็น layer ที่เก็บข้อมูลการ Inflation จากจุด lethal ของสิ่งกีดขวาง
- layerอื่นๆ นอกเหนือจากนี้จะอยู่ใน pluginlibเช่น Range Sensor Layer เป็นต้น

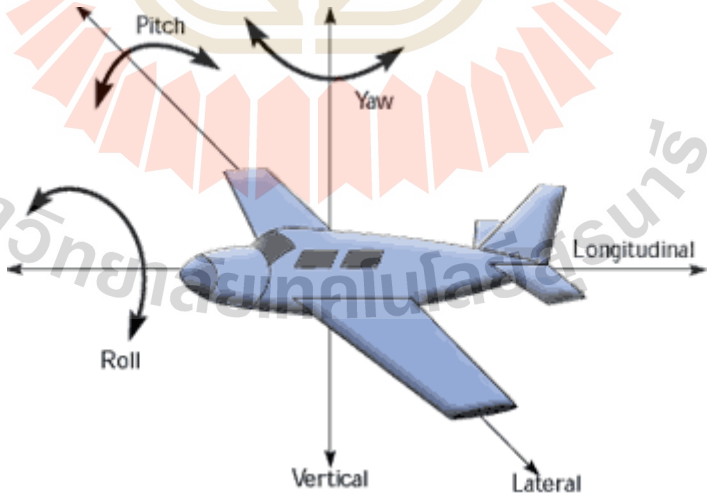
3.7 Move Base และแผนที่ Costmap

เมื่อสามารถรู้ตำแหน่งของหุ่นยนต์แล้ว ขั้นตอนต่อไปของ ROS Navigation คือการรู้ถึงลักษณะของสิ่งกีดขวางโดยรอบ เพื่อให้หุ่นยนต์ค้นหาเส้นทาง โดยไม่ติดขัดหรือชนและสามารถไป

ถึงเป้าหมายได้ตามที่ต้องการ สภาพแวดล้อมที่มีสิ่งกีดขวางนั้น ROS จะใช้เทคนิคของ Costmapเพื่อเป็นตัวกำหนดเส้นทางในการเคลื่อนที่ และเมื่อทราบว่ามีสิ่งกีดขวางอยู่นั้น สามารถวางแผนเส้นทางในการเคลื่อนที่ได้ด้วย อัลกอริทึมการวางแผนหรือ Planner ใน Move Baseจะมีขอบเขตอยู่ 2 ระดับ คือ Global Planner และ Local Planner ซึ่งทั้งสอง Planner จะทำการคำนวณเพื่อสร้างเส้นทางให้หุ่นยนต์สามารถเคลื่อนที่ไปยังเป้าหมาย

3.7.1 เป้าหมาย Move Base

การกำหนดจุดหมายปลายทางให้กับการเคลื่อนที่ของหุ่นยนต์สำรวจ ซึ่งถ้าไม่มีความสามารถนี้หุ่นยนต์สำรวจก็จะทำได้เพียงเคลื่อนที่สำรวจไปมาหรือสามารถหลบหลีกสิ่งกีดขวางได้เท่านั้น สำหรับ ROS Navigation จะใช้ Massage ที่ชื่อว่า geometry_msgs/Posestampedโดยมีอยู่ในรูปแบบ std_msgs/Header header, geometry_msgs/Pose pose ซึ่งข้อมูลในส่วนของ std_msgs/Header header จะประกอบไปด้วย Meta-data ต่างๆ เช่น ชื่อ frame เป็นต้น ส่วนของ geometry_msgs/Pose Pose จะประกอบไปด้วย geometry_msgs/Point position, geometry_msgs/Quaterninon orientation ซึ่ง position มีประเภทเป็น Point หมายถึง จุดใดๆ ในแผนที่ได้แก่ X, Y, Z ที่ต้องการกำหนดให้หุ่นยนต์สำรวจเคลื่อนที่ไปสู่จุหมาย ส่วนค่า orientation คือทิศทางในการหมุนของตัวหุ่นยนต์สำรวจจะมีค่าเป็น Quaterninonได้แก่ค่าที่เกิดจากการหมุนทั้งสามแกน (row-pitch-yaw) โดยหุ่นยนต์สำรวจจะมีการหมุนเพียงแกนเดียวคือแกน Z (yaw) เท่านั้น ภาพที่ 3.27แสดงการหมุนของแกน row-pitch-yaw



รูปที่ 3.27 แสดงการหมุนของแกน row-pitch-yaw

การสั่งให้หุ่นยนต์สำรวจเคลื่อนที่ไปสู่เป้าหมายในระบบ ROS สามารถทำได้ 2 วิธีได้แก่

1.สั่งการทำงานในโปรแกรม Rvizที่มีอยู่ในระบบ ROS โดยการกดปุ่มที่ชื่อว่า 2D Nav Goal เมื่อคลิ๊กที่ปุ่มแล้วไปทำการคลิ๊กและลากใน Grid ของ Map โดยเลือกทิศทางไปในทิศทางที่ต้องการแล้วจะปรากฏลูกศรของปลายทางที่ต้องการให้หุ่นยนต์สำรวจเคลื่อนที่ไปในโปรแกรม Rviz

2.การใช้ Action ที่ทำงานตามรูปแบบของ MoveBaseActionคือการเขียนโปรแกรมเข้าไปใน Node โดยตรงเนื่องจากการใช้งานโปรแกรม Rvizเป็นเพียงการใช้งานเพื่อการทดสอบการทำงานของการเคลื่อนที่แบบอัตโนมัติเท่านั้น แต่ในการทำงานจริงของหุ่นยนต์สำรวจ เช่น การเคลื่อนที่ไปยังตำแหน่งต่างๆ ของหุ่นยนต์สำรวจ จำเป็นที่ต้องเขียนโปรแกรมก็สามารถใช้ MoveBaseAction ได้

3.7.2 สร้างไฟล์ที่เป็น Configuration ต่างๆ สำหรับการทำงานของ Costmap Planner

1. base_local_planner_params.yamlเป็นการตั้งค่าเกี่ยวกับ Local Planner

2. costmap_common_params.yamlเป็นการตั้งค่าเกี่ยวกับ Costmapของทั้ง Global และ Local

3. global_costmap_params.yamlเป็นการตั้งค่าเกี่ยวกับ Global Costmap

4. local_costmap_params.yamlเป็นการตั้งค่าเกี่ยวกับ Local Costmap

ตัวอย่างการตั้งค่าไฟล์ base_local_planner_params.yaml ที่ได้ทำการทดสอบมีดังนี้

```
TrajectoryPlannerROS:

# Robot Configuration Parameters

max_vel_x: 0.18

min_vel_x: 0.08

max_vel_theta:  1.0

min_vel_theta: -1.0

min_in_place_vel_theta: 1.0
```

acc_lim_x: 1.0

acc_lim_y: 0.0

acc_lim_theta: 0.6

Goal Tolerance Parameters

xy_goal_tolerance: 0.10

yaw_goal_tolerance: 0.05

Differential-drive robot configuration

holonomic_robot: false

Forward Simulation Parameters

sim_time: 0.8

vx_samples: 18

vtheta_samples: 20

sim_granularity: 0.05

ค่า max_vel_x และ min_vel_x คือความเร็วสูงสุดและต่ำสุดในการเคลื่อนที่ของหุ่นยนต์สำรวจ ค่า max_vel_theta และ min_vel_thetaคือความเร็วสูงสุดและต่ำสุดในการหมุนของหุ่นยนต์สำรวจ(หน่วยเป็น radian/sec)ค่า min_in_place_vel_theta คือความเร็วต่ำสุดของการหมุนหุ่นยนต์สำรวจค่า acc_lim_xและacc_lim_yคือความเร่งสูงสุดของการเคลื่อนที่แบบเชิงเส้นค่าacc_lim_theta คือความเร่งสูงสุดของการหมุนของหุ่นยนต์สำรวจ ค่า xy_goal_toleranceคือระยะตำแหน่งเป้าหมายของหุ่นยนต์สำรวจค่าyaw_goal_tolerance คือการหมุนหาตำแหน่งทิศทางเป้าหมายของหุ่นยนต์สำรวจค่า holonomic_robot คือการเคลื่อนที่แบบ holonomicคือการเคลื่อนที่ไปได้แบบทุกทิศทุกทาง สำหรับหุ่นยนต์สำรวจเป็นแบบ Differential Drive ค่าที่กำหนดจึงเป็น false ค่า sim_timeคือเวลาที่เคลื่อนที่ไปข้างหน้าของหุ่นยนต์สำรวจค่าvx_samples คือความเร็วในแนวแกน x ค่า vy_samples คือความเร็วในแนวแกน y และค่า vtheta_samplesคือคามเร็วในการหมุน

ตัวอย่างการตั้งค่าไฟล์ costmap_common_params.yamlที่ได้ทำการทดสอบมีดังนี้

obstacle_range: 3.0

raytrace_range: 3.5

footprint: [[-0.105, -0.105], [-0.105, 0.105], [0.041, 0.105], [0.041, -0.105]]

#robot_radius: 0.105

inflation_radius: 1.0

cost_scaling_factor: 3.0

map_type: costmap

observation_sources: scan

scan: {sensor_frame: base_scan, data_type: LaserScan, topic: scan, marking: true, clearing: true}

obstacle_rangeคือระยะที่ใช้ในการตรวจสอบสิ่งกีดขวาง raytrace_range คือระยะที่ใช้ในการแสกนหาพื้นที่ว่าง footprint คือจุด x,yของตำแหน่งหุ่นยนต์สำรวจโดยวัดจากขนาดรูปร่างของตัวหุ่นยนต์สำรวจซึ่งจุต่างๆ จะเชื่อมเข้าหากันแบบ Polygon ดังที่ได้กล่าวมาinflation_radiusคือระยะการ inflation ของCostmapและ observation_sources คือแหล่งข้อมูลของเซนเซอร์ที่ใช้ทำ Costmap

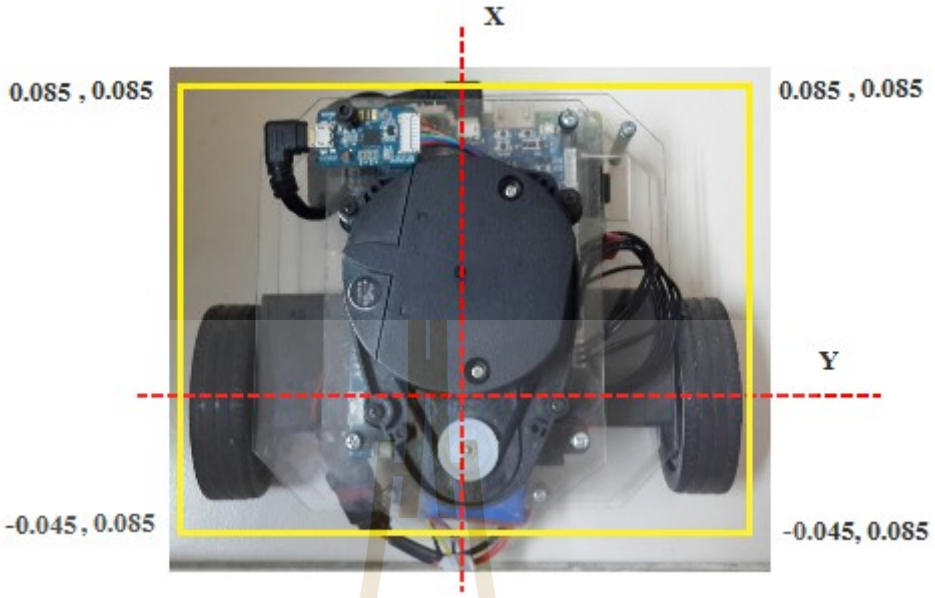
หลักการในการกำหนดค่า Polygon ให้กับหุ่นยนต์สำรวจ

1.วัดระยะความกว้างและความยาวของหุ่นยนต์สำรวจโดยการวัดจากจุดศูนย์กลาง (หน่ายเป็นเมตร)

2.ส่วนด้านหน้าของหุ่นยนต์สำรวจจะตรงกับแกน x ของ base_link

3.การกำหนดตำแหน่งของความกว้างและความยาวควรเกินจริงเล็กน้อย (ระยะเผื่อ)

ในการกำหนดค่า Polygon อาจจะเกิดความสับสนจึงได้แสดงการกำหนดวิธีการคำนวณหา Polygon ดังรูปที่ 3.28



รูปที่ 3.28 วิธีการคำนวณหา Polygon

ตัวอย่างการตั้งค่าไฟล์ global_costmap_params.yamlที่ได้ทำการทดสอบมีดังนี้

global_costmap:

global_frame: /map

robot_base_frame: /base_footprint

update_frequency: 10.0

publish_frequency: 10.0

transform_tolerance: 0.5

static_map: true

global_frameคือชื่อของ frame ที่ Global ใช้แผนที่ map ส่วนค่า robot_base_frame คือชื่อ frame ของหุ่นยนต์สำรวจซึ่งชื่อว่า base_footprint และค่าupdate_frequency คือความถี่ของการอัพเดทตัว Costmapค่า publish_frequency คือความถี่ในการ publish ตัว Costmap ค่า transform_tolerance คือการเคลื่อนที่ในการเคลื่อนที่ไปให้ถึงจุดหมาย ค่า static_map คือเป็นค่าการบอกว่า Global Costmapมีการใช้แผนที่แบบตายตัวหรือไม่ (ในงานวิทยานิพนธ์เล่มนี้เป็น true เพราะเป็นแผนที่จาก Map Server)

ตัวอย่างการตั้งค่าไฟล์ local_costmap_params.yamlที่ได้ทำการทดสอบมีดังนี้

local_costmap:

global_frame: /odom

robot_base_frame: /base_footprint

update_frequency: 10.0

publish_frequency: 10.0

transform_tolerance: 0.5

static_map: false

rolling_window: true

width: 3

height: 3

resolution: 0.05

ค่าglobal_frame คือชื่อ frame ของ Global ของ Local Costmapซึ่งส่วนใหญ่จะใช้เป็น /odom เนื่องจากมีความสอดคล้องกับค่าจากระบบ odometryของหุ่นยนต์สำรวจ ค่า robot_base_frame คือ frame ของตัวหุ่นยนต์สำรวจในที่นี้คือ /base_footprint ค่า update_frequency คือความถี่ในการอัพเดทค่า Local Costmap ค่าpublish_frequency คือความถี่ในการ publish ตัว Costmap ค่า static_map ในที่นี้เป็น false เนื่องจาก Local Costmap จะใช้ map จากตัวเซนเซอร์ LiDARโดยตรงไม่ได้ใช้จาก Map Server ค่า rolling_window ถ้าหากว่าเป็น true จะทำการสร้าง Costmap รอบๆ ตัวของหุ่นยนต์สำรวจไม่ว่าจะเคลื่อนที่ไปยังตำแหน่งไหนก็ตาม ส่วน width, height คือความกว้างและความสูงของ Costmapตามลำดับส่วนค่า resolution คือความละเอียดของการสร้าง Costmap ในที่นี้คือ 5 เซนติเมตร ยิ่งละเอียดยิ่งดีแต่เปลืองพื้นที่มาก

3.7.3 launch file ของ Move base

move_base.launch คือคำสั่งการเปิดการทำงานของ Map Server โดยการส่งไฟล์แผนที่ที่จัดเก็บไว้จากการทำ SLAM เช่นแผนที่ที่ได้สร้างเก็บไว้ในที่นี้คือไฟล์ที่จะใช้จะลงท้ายด้วย

นามสกุล yamlแต่ในความเป็นจริงจะมี 2 ไฟล์ดังที่กล่าวถึงไปแล้วส่วนคำสั่งอื่นๆ คือการเปิดการทำงานของ move_base โดยการกำหนดค่าพารามิเตอร์ต่างๆ ให้กับ Node ได้แก่ไฟล์ Configuration ต่างๆ ที่ได้กล่าวมาแล้ว

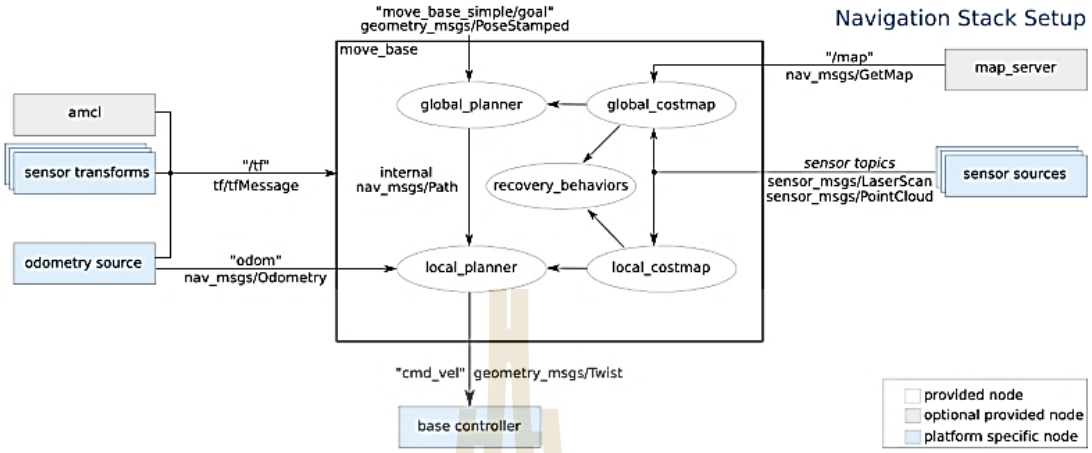
ทดสอบการทำงานของ Global Costmapและ Local Costmapโดยการเรียกใช้คำสั่ง roslaunchmove_base.launchโดยยังไม่ต้องสั่งเป้าหมายในการเคลื่อนที่ให้กับหุ่นยนต์สำรวจและทำการเพิ่มแผนที่เข้าไปในโปรแกรม Rvizซึ่งการกำหนดค่าในโปรแกรม Rviz นั้นจะต้องเพิ่ม Topic ประเภท /map โดยการเข้าไปกำหนดชื่อ Topic เป็น / move_base/ local_costmap/ costmapสำหรับดู local costmap และ move_base/ global_costmap/ costmap สำหรับ global costmap ซึ่งการทำงานในส่วนนี้มีความสำคัญต่อการ Navigation เนื่องจากต้องใช้ Costmapในการคำนวณหาเส้นทางการเคลื่อนที่ จะเห็นว่าเมื่อมีสิ่งกีดขวางเข้าไปในระยะการเคลื่อนที่ของหุ่นยนต์สำรวจเซนเซอร์ LiDARจะเกิดการ inflation ใหม่ตามข้อมูลจริงที่เซนเซอร์ LiDARตรวจพบ จึงเรียกว่า Local Costmap สำหรับส่วน Global Costmap จะเป็นค่านิ่งเนื่องจากเป็นการกำหนดให้ใช้แผนที่แบบตายตัว ซึ่งหลักการทำงานของ Costmap ทำงานได้อย่างถูกต้องแสดงว่าพร้อมสำหรับการทำงานในส่วนของ Planner

3.8 Planner

ในหัวข้อที่ผ่านมาได้แยก launch file ออกเป็น 2 ไฟล์ เพื่อให้ง่าย แต่สำหรับการสั่งการหุ่นยนต์สำรวจให้ทำงานตาม Navigation Stack จริงๆ ควรนำมารวมใน launch file อันเดียว แต่จะกล่าวถึง Planner เพื่อความเข้าใจก่อน เนื่องจากเป็นอัลกอริทึมที่สำคัญสำหรับ ROS Navigation Stack

Planner คืออัลกอริทึม ภายใน Node ที่ทำหน้าที่ค้นหาเส้นทางและการสั่งการหุ่นยนต์สำรวจให้เคลื่อนที่ไปสู่เป้าหมายที่ต้องการ Planner มี 2 ประเภท ได้แก่

1.Global Planner คืออัลกอริทึมที่จะสร้างเส้นทางให้หุ่นยนต์สำรวจไปสู่เป้าหมายที่ต้องการ โดยจะสร้างเส้นทางตั้งแต่เริ่มต้นทางไปจนถึงปลายทาง Global Planner สามารถใช้แผนที่ได้ทั้งจากแผนที่แบบตายตัวหรือจาก Topic ก็ได้ ขึ้นอยู่กับลักษณะของสิ่งแวดล้อม หากเป็นสิ่งแวดล้อมที่มีการเปลี่ยนแปลงไม่บ่อยครั้ง ก็สามารถใช้แผนที่แบบตายตัวจาก Map Server ได้ (ซึ่งสร้างจาก SLAM แล้วทำการเก็บไว้) สำหรับ Global Plannerจะอยู่ใน Package ชื่อว่า nav_coreซึ่งประกอบไปด้วย Interface สำหรับการวางแผนเส้นทางระดับ Global ได้แก่ BaseGlobalPlanner, BaseLocalPlannerและ RecoveryBehaviorดังรูปที่ 3.29



รูปที่ 3.29 อธิบาย nav_corePackage

สำหรับ Planner ที่จะมาทำงานตาม lnterfaceจะสร้างเป็นรูปแบบของ Plugin ที่ชื่อ move_base และสำหรับ lnterface ที่ชื่อว่า nav_core:: BaseGlobalPlannerเป็น lnterfaceหลักสำคัญ GlobalPlanner หมายความว่าจะสร้างคลาสที่สามารถ Implement ตัว lnterface นั้น ตัวอย่างของคลาสที่ว่าได้แก่

- global_plannerเป็น global_planner/GlobaPlanner

- navfnทำงานแบบ Grid-based ชื่อ Plugin คือ navfn/NavfnROS

- carrot_ planner เป็น Global Planner ที่ง่ายที่สุด คือพยายามเคลื่อนที่ไปให้ใกล้เป้าหมายมากที่สุด แม้เป้าหมายจะอยู่ในสิ่งกีดขวางก็จะเคลื่อนที่ไปให้ใกล้มากที่สุด

2.Local Planner คืออัลกอริทึมที่จะสร้างเส้นทางให้หุ่นยนต์ไปสู่เป้าหมายในระดับย่อย หรือเส้นทางที่ได้จาก Global Planner ซึ่ง Local Planner จะทำหน้าที่หลบหลีกสิ่งกีดขวางที่อยู่ในระยะใกล้ และพยายามเคลื่อนที่ไปตามเส้นทางด้วยระยะที่สั้นที่สุด โดยปกติแล้ว Local Planner จะไม่ใช่แผนที่แบบตายตัว เนื่องจากเน้นการหลบสิ่งกีดขวางที่ทำการเคลื่อนที่ แต่จะสร้างแผนที่ขึ้นตามเวลาจริง จากเซนเซอร์ LiDARเช่นเดียวกับ Global Planner ในส่วนของ Local Planner จะใช้ lnterface ที่ชื่อว่า nav_core:: BasLocalPlanner โดยคลาสที่ Implement ตัว lnterfaceได้แก่

- base_local_planner ใช้การทำงานแบบหน้าต่างหรือ Dynamic Window และการสร้างเส้นทางสุ่มขึ้นมา (Trajectory Rollout) เพื่อให้เดินไปตามเส้นทางแบบ Local ได้

- ebanb_local_plannerใช้วิธีที่เรียกว่า Elastic Band

- teb_local_ planner ใช้วิธีของ Time_Elastic-Band ในการปรับเส้นทางเดิน

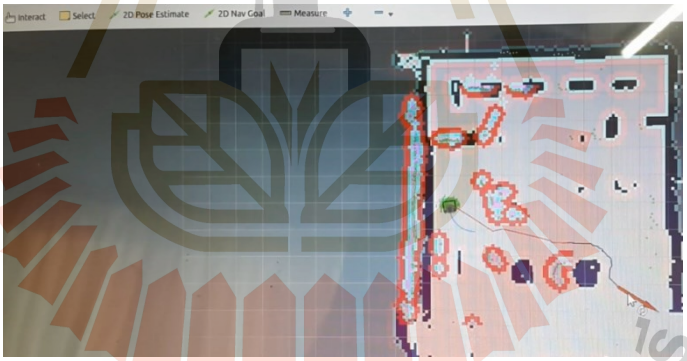
3.Recovery Behavior บางครั้งการเคลื่อนที่ของหุ่นยนต์สำรวจตาม GlobalPlanner หรือ Local Planner นำไปสู่การติดขัดภายใน Costmapทำให้หุ่นยนต์ต้องการหาเส้นทางใหม่ๆ เพื่อจะหลุดออกจากการติดขัดนั้น เรียกว่าการ Recovery ซึ่งใช้ lnterface ชื่อว่า nav_core::RecoveryBehavior และมีคลาสที่เกี่ยวข้องได้แก่

- clear_costmap_recovery พยายามไปใช้แผนที่แบบตายตัว ที่อยู่ภายนอกขอบเขตที่ผู้ใช้กำหนด

- rotate_recoveryทำการหมุน 360 องศา เพื่อปรับหรือรับค่าใหม่

3.9 วิธีในการทดสอบสู้เป้าหมายของหุ่นยนต์สำรวจ

ในการทดสอบนี้จะใช้โปรแกรม Rvizที่มีอยู่ในระบบ ROS โดยการกดปุ่มในโปรแกรม Rviz หลังจากนั้นหุ่นยนต์สำรวจจะทำการเคลื่อนที่ไปทันทีดังรูปที่ 3.30การทดสอบ Navigation ด้วยโปรแกรมRviz



รูปที่ 3.30 การทดสอบ Navigation ด้วยโปรแกรมRviz

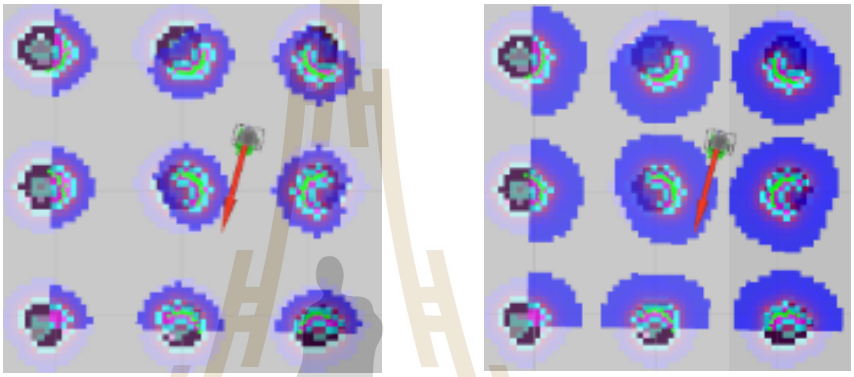
3.10 วิธีTuning GuideROS Navigation

หุ่นยนต์สำรวจที่ใช้ระบบ ROS Navigationจะมีประสิทธิภาพมากสำหรับการเคลื่อนที่จากที่หนึ่งไปอีกที่หนึ่ง แต่ Navigationstack คือการสร้างเส้นทางที่มีความปลอดภัยสำรับหุ่นยนต์สำรวจ โดยการประมวลผลข้อมูลจากการเคลื่อนที่ของหุ่นยนต์ เซนเซอร์ต่างๆ และแผนที่สภาพแวดล้อมที่อยู่รอบๆ หุ่นยนต์สำรวจการเพิ่มประสิทธิภาพของ ROSNavigation นั้นจะต้องมีการปรับพารามิเตอร์บางอย่างให้ละเอียดและไม่ใช่เรื่องง่ายอย่างในการปรับพารามิเตอร์ สำหรับ

การปรับค่าพารามิเตอร์ที่สำคัญ หากต้องการเปลี่ยนการแปลงโดยขึ้นอยู่กับสภาพแวดล้อมต่างๆ จะมีวิธีในการปรับค่าพารามิเตอร์ดังต่อไปนี้

3.10.1 Inflation_Radius

พารามิเตอร์นี้จะทำให้พื้นที่สีน้ำเงินเพิ่มหรือลดเป็นส่วนสำคัญในการกำหนดเส้นทางจะมีการวางแผนเพื่อไม่ให้ข้ามพื้นที่สีน้ำเงินนี้ เพื่อความปลอดภัยที่จะทำการตั้งค่านี้ให้ใหญ่กว่ารัศมีหุ่นยนต์สำรวจ



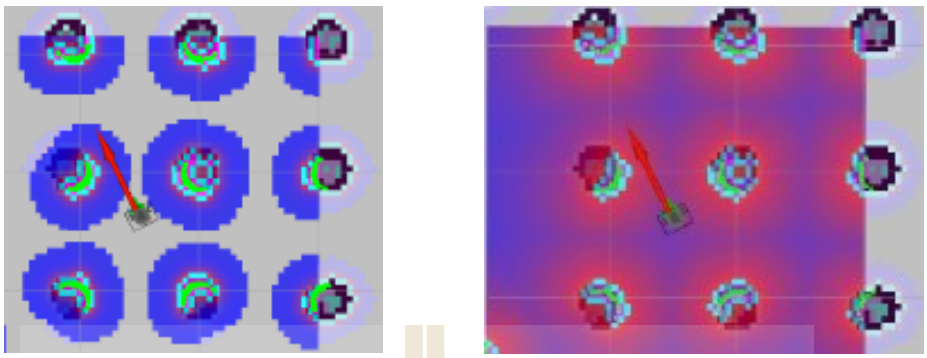
ก) inflation_radius =0.2 ข) inflation_radius =1.0

รูปที่ 3.31 การปรับค่าinflation_radius

3.10.2 Cost_scaling_factor

ค่า factorจะแปลผันตามค่า Costเนื่องจากเป็นข้อดีซึ่งกันและกันของพารามิเตอร์นี้ซึ่งถ้ากำหนดค่าพารามิเตอร์มากค่าความปลอดภัยก็จะเพิ่มมากขึ้นค่าพารามิเตอร์น้อยค่าความปลอดภัยจะลดลงดังนั้นควรเลือกเส้นทางที่ดีที่สุดคือให้หุ่นยนต์สำรวจสามารถผ่านจุดกึ่งกลางระหว่างอุปสรรคและทำการตั้งค่าพารามิเตอร์นี้ให้เหมาะสมเพื่อให้ห่างไกลจากอุปสรรคและป้องกันการชนกับสิ่งกีดขวาง ซึ่งตัวแปรในการปรับพารามิเตอร์ที่ใช้ในการคำนวณ costmapสูตรคำนวณมีดังนี้

exp(-1.0 * cost_scaling_factor *(distance_from_obstacle - inscribed_radius)) *(254 - 1) (3.1)



ก)cost_scaling_factor = 20 ข)cost_scaling_factor = 2.0

รูปที่ 3.32 การปรับค่าcost_scaling_factor

3.10.3 ความเร็วและความเร่ง

ความเร็วสูงสุดและอัตราเร่งเป็นสองพารามิเตอร์พื้นฐานสำหรับการเคลื่อนที่ของหุ่นยนต์สำรวจ การตั้งค่าเหล่านี้อย่างถูกต้อง จะมีประโยชน์อย่างยิ่งกับการวางแผนในการเคลื่อนที่ของหุ่นยนต์สำรวจให้ได้ดีที่สุด ในระบบROS Navigationเราจำเป็นต้องทราบความเร็วในการเคลื่อนที่และการหมุนและการอัตราเร่งความเร็วต่าง ๆ ดังต่อไปนี้

max_vel_x คือการกำหนดค่าความเร็วสูงสุดในการเคลื่อนที่หุ่นยนต์สำรวจ

min_vel_x คือการกำหนดค่าความเร็วต่ำสุดในการเคลื่อนที่หุ่นยนต์สำรวจ และหากตั้งค่าเป็นค่าลบนี้หุ่นยนต์สำรวจจะเคลื่อนที่ไปข้างหลังได้ด้วย

max_trans_vel คือ ค่าที่แท้จริงของความเร็วสูงสุดในการเคลื่อนที่หุ่นยนต์สำรวจ ซึ่งหุ่นยนต์สำรวจจะไม่สามารถเคลื่อนที่ได้เร็วกว่านี้

min_trans_vel คือ ค่าที่แท้จริงของความเร็วต่ำสุดในการเคลื่อนที่หุ่นยนต์สำรวจ ซึ่งหุ่นยนต์สำรวจจะไม่สามารถเคลื่อนที่ได้ช้ากว่านี้

max_rot_vel คือ การกำหนดค่าความเร็วสูงสุดในการหมุนของหุ่นยนต์สำรวจ

min_rot_vel คือ การกำหนดค่าความเร็วต่ำสุดในการหมุนของหุ่นยนต์สำรวจ

acc_lim_xคือ การกำหนดค่าความเร่งสูงสุดในการหมุนของหุ่นยนต์สำรวจ

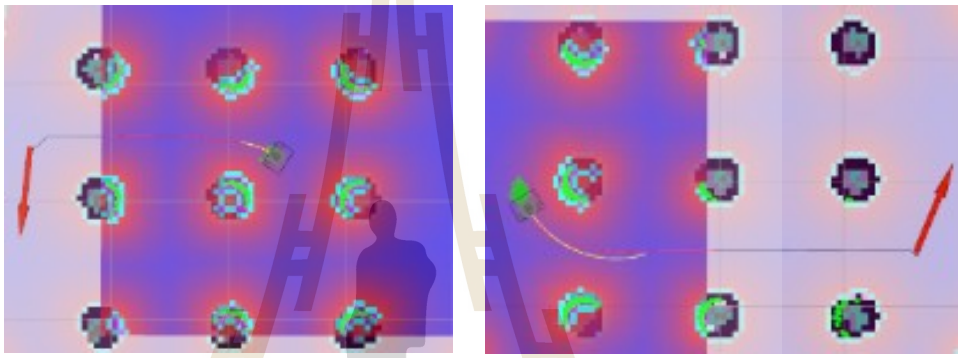
acc_lim_theta คือ การกำหนดค่าความเร่งสูงสุดในการหมุนของหุ่นยนต์สำรวจ

xy_goal_tolerance คือ ระยะทาง x, y เมื่อหุ่นยนต์สำรวจถึงจุดเป้าหมาย

yaw_goal_tolerance คือ มุมด้านหน้าของหุ่นยนต์สำรวจเมื่อหุ่นยนต์สำรวจถึงจุดเป้าหมาย

3.10.4 Sim_Time

การตั้งค่าพารามิเตอร์ที่เป็นแบบจำลองของเวลาไม่กี่วินาที ค่าพารามิเตอร์ที่กำหนดต่ำเกินไปหรืออยู่ในช่วงเวลาที่เหมาะสม เพื่อให้หุ่นยนต์สำรวจเคลื่อนที่ผ่านพื้นที่แคบและค่าพารามิเตอร์ที่สูงมากเกินไปไม่สามารถหมุนหรือเคลื่อนที่อย่างรวดเร็วได้สามารถสังเกตได้จากความยาวของเส้นสีเหลืองในรูปที่ 3.33



ก)sim_time = 1.5 ข)sim_time= 5.0

รูปที่ 3.33 การปรับค่าsim_time

3.10.5 Dynamic Window Approach (DWA)

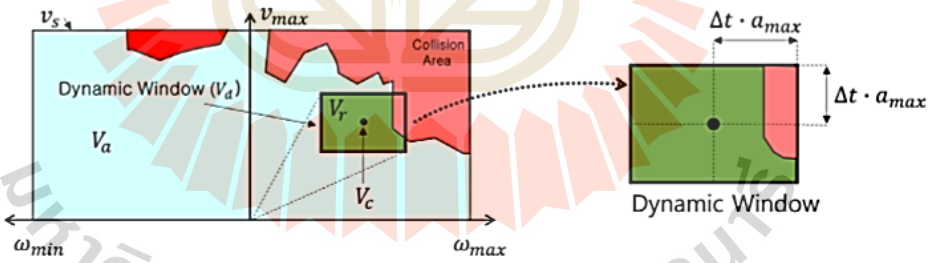
วิธีแบบDynamic Window Approach(DWA) เป็นวิธีที่ได้รับความนิยมสำหรับการวางแผนในการหลบหลีกสิ่งกีดขวางและหลีกเลี่ยงจากการชนซึ่งเป็นวิธีการเลือกความเร็วที่สามารถเข้าถึงจุดเป้าหมายได้อย่างรวดเร็วโดยไม่ต้องเจอกับสิ่งกีดขวางที่อาจจะทำให้ชนกับหุ่นยนต์สำรวจในพื้นที่ความเร็วในระบบROS สามารถวางแผนเส้นทางจะถูกใช้สำหรับการวางแผนเส้นทางระยะใกล้ แต่มีการแทนที่ DWA เนื่องจากมีประสิทธิภาพมากกว่า

อันดับแรกหุ่นยนต์ไม่อยู่ในพิกัดแกน Xและแกน Yแต่ในพื้นที่ค้นหาความเร็วที่มีความเร็วในการเคลื่อนที่เป็น Vและความเร็วในการหมุนของแกนเป็น ω เป็นดังแสดงในรูปที่ 2.12 ในพื้นที่นี้หุ่นยนต์สำรวจจะมีความเร็วสูงสุดที่สามารถเคลื่อนที่ได้เนื่องจากข้อจำกัดของฮาร์ดแวร์และเรียกว่า Dynamic Windowโดยมีค่าตัวแปลต่างๆ ดังต่อไปนี้

v: Translational velocity (meter/sec)
ω: Rotational velocity (radian/sec)
V_s: Maximum velocity area
V_a: Permissible velocity area
V_c: Current velocity
V_r: Speed area in Dynamic Window
a_{max}: Maximum acceleration / deceleration rate
$G_{(v,\omega)} = \sigma(\alpha \cdot heading(v,\omega) + \beta \cdot dist(v,\omega) + \gamma \cdot velocity(v,\omega))$: Objective function
$heading(v,\omega)$: 180 - (difference between the direction of the robot and the direction of the target point)
$dist(v,\omega)$: Distance to the obstacle
$velocity(v,\omega)$: Selected velocity
α, β, γ: Weighting constant
$\sigma(x)$: Smooth Function

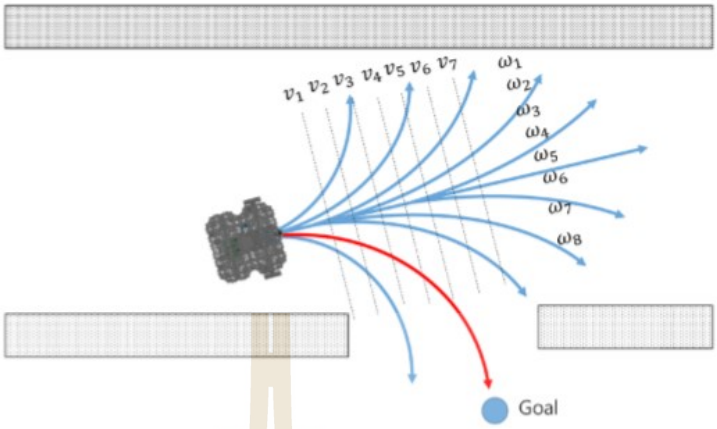
รูปที่ 3.34 ตัวแปรของ Dynamic Window

ในหน้าต่างแบบDynamic Windowนี้ฟังก์ชันวัตถุประสงค์ G (V, ω) ใช้ในการคำนวณ หาความเร็วในการเคลื่อนที่Vและความเร็วในการหมุนเป็น ωซึ่งเพิ่มฟังก์ชันในการพิจารณาทิศทาง ความเร็วและการชนของหุ่นยนต์สำรวจ ถ้าเราพล็อตกราฟจะสามารถหาความเร็วที่ดีที่สุดระหว่าง ความเร็วVและความเร็วในการหมุนωจากตำแหน่งเริ่มต้นจนไปถึงตำแหน่งเป้าหมายตามที่ได้กล่าว ไว้ในรูปที่ 2.13



รูปที่ 3.35 ความเร็วในการเคลื่อนที่แบบ Dynamic Window

การใช้แอพพิเคชั่นและทฤษฎี SLAM และROS Navigation แม้ส่วนใหญ่จะเป็นการอธิบาย รูปแบบในการเคลื่อนที่ของหุ่นยนต์สำรวจแต่ก็สามารถใช้กับหุ่นยนต์ตัวอื่นได้เช่นเดียวกัน



รูปที่ 3.36 ความเร็วในการเคลื่อนที่เป็น V และความเร็วในการหมุนเป็น ω

บทที่ 4

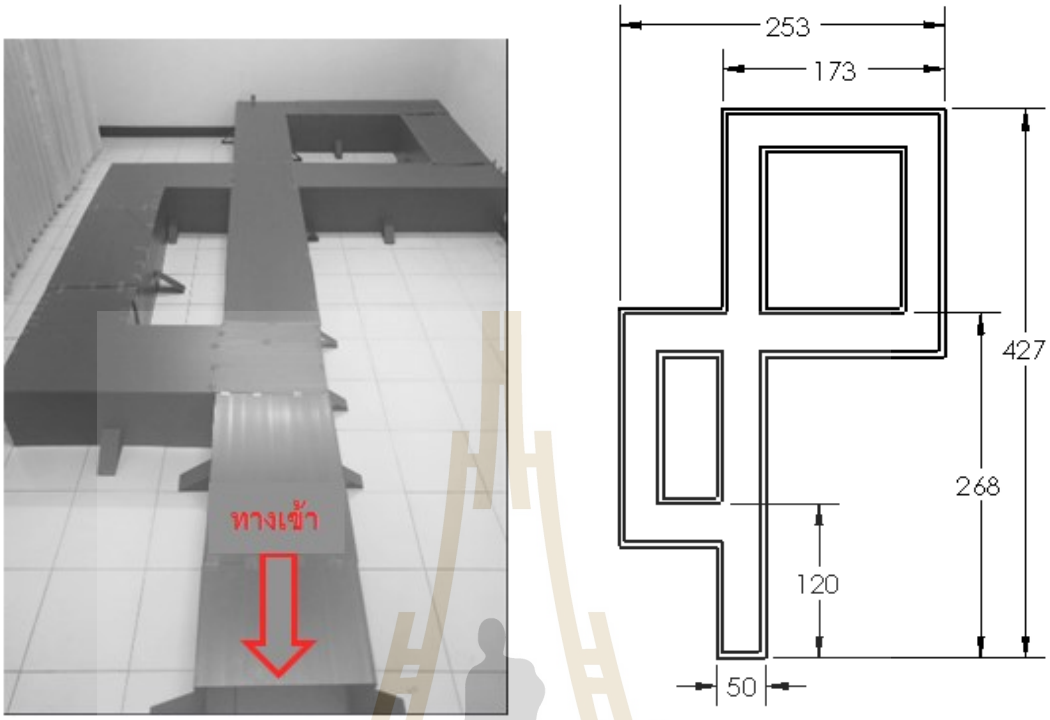
การทดลองและผลการทดลอง

หลังจากที่ได้ทดสอบการทำงานและตรวจสอบวิธีของอัลกอริทึมต่างๆ ที่ใช้ในงานวิจัยในครั้งนี้แล้วในบทนี้จะเป็นการแสดงการทดลองและผลการทดสองในสภาพแวดล้อมจริงที่ได้จำลองขึ้นเอง โดยการใช้เซนเซอร์ Lidarที่ได้กล่าวมาในการรับรู้ข้อมูลของสิ่งกีดขวางที่อยู่รอบๆ สำหรับการทดลองจะแบ่งออกเป็น 3 ส่วนคือ (1) การสำรวจและสร้างแผนที่ (2) การระบุตำแหน่งเคลื่อนที่อัตโนมัติของหุ่นยนต์สำรวจ(3) การหลบหลีกสิ่งกีดขว้างในพื้นที่แคบ

4.1 การทดลองการสำรวจและสร้างแผนที่

สำหรับแบบจำลองที่ใช้ในการทดลองในงานวิจัยนี้เป็นแบบจำลองที่สร้างขึ้นเองโดยผู้วิจัย ซึ่งในแบบจำลองนี้จะมีลักษณะพื้นผิวของสภาพแวดล้อมเป็นแบบพื้นราบ พื้นที่แคบและพื้นที่ปิดตลอดจนสภาพแวดล้อมที่เป็นอุปสรรคต่าง ๆ ซึ่งลักษณะของแบบจำลองจะประกอบด้วยทางแยกที่มีแยกสามแยกและทางแยกที่เป็นสี่แยก ซึ่งลักษณะของแบบจำลองจะมีดังต่อไปนี้

- ความกว้างของท่อมีขนาด 50 เซนติเมตร ความยาว 420 เซนติเมตร ความสูง 30 เซนติเมตร
- ระยะทาง 90 เซนติเมตร จากทางเข้าจะเป็นทางสามแยก
- ระยะทาง 246 เซนติเมตร จากทางเข้าจะเป็นทางสี่แยก
- ระยะทางในการสำรวจทั้งหมดอยู่ที่ 1,204 เซนติเมตร
- ทางเข้า-ออกเป็นทางเดียว

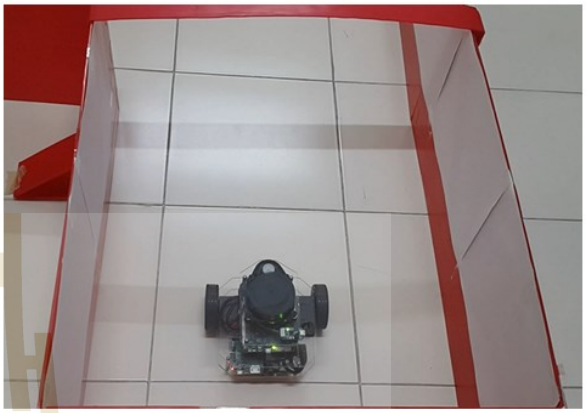


(ก)แบบจำลองที่สร้างขึ้น (ข)ขนาดของแบบจำลอง

รูปที่ 4.1 แบบจำลองที่ใช้ในการทดลอง

4.1.1 การทดลองการสำรวจและสร้างแผนที่

สำหรับการทดลองในงานวิจัยนี้จะเป็นการสำรวจและสร้างแผนที่โดยการเริ่มต้นบังคับหุ่นยนต์สำรวจแบบบังคับด้วยแป้นคีย์บอร์ดให้เคลื่อนที่เข้าไปในท่อจากปากทางเข้าของท่อเพื่อเริ่มทำการสำรวจพร้อมกับการสร้างแผนที่ซึ่งจะใช้เซนเซอร์ Lidar ในการรับข้อมูลของสิ่งแวดล้อมที่อยู่รอบๆ ตัวหุ่นยนต์สำรวจ โดยที่ Localization จะเป็นตัวที่บอกให้ทราบว่าหุ่นยนต์สำรวจอยู่ตรงไหนในแผนที่ที่สร้าง ซึ่งในการเคลื่อนที่ของหุ่นยนต์สำรวจจะใช้การวัดระยะทางของเซนเซอร์ Lidarเพื่อนำมาสร้างเป็นแผนที่ เมื่อเคลื่อนที่เข้าไปในท่อได้ระยะทาง 90 เซนติเมตร จะพบกับทางแยกที่เป็นสามแยกและเมื่อหุ่นยนต์สำรวจเคลื่อนที่เข้าไปได้ระยะทาง 246 เซนติเมตร จะพบกับทางแยกที่เป็นสี่แยก โดยมีระยะทางที่ทำการสำรวจและสร้างแผนที่ตั้งแต่ทางเข้าของท่อจนกระทั่งกลับมาอยู่ ณ ตำแหน่งเริ่มต้นระยะทางทั้งหมดอยู่ที่ 1,204 เซนติเมตรและเมื่อทำการทดลองเสร็จจะได้แผนที่ดังรูปที่ 4.2(ก)

(ก) แผนที่ที่สร้างขึ้นด้วย SLAM Gmapping (ข)จุดเริ่มต้นบังคับหุ่นยนต์สำรวจ

รูปที่ 4.2 การสำรวจและสร้างแผนที่

จากการทดลองในการสำรวจและสร้างแผนที่ด้วยการใช้หลักการในการแปลงข้อมูลของจุดที่ต้องการคือ ระยะทางของสิ่งกีดขวางที่วัดได้จากเซนเซอร์ Lidar ซึ่งระยะทางที่วัดด้วยเซนเซอร์ Lidarเรียกว่าสแกนข้อมูลของตำแหน่งการเคลื่อนที่เรียกว่า Transformในระบบ ROS ดังแสดงในรูปที่ 4.2 (ก) สีเขียวคือระยะของเซนเซอร์ที่ได้ทำการสแกนซึ่งในการสแกนของหุ่นยนต์สำรวจจะทำให้ทราบถึงตำแหน่งของหุ่นยนต์สำรวจด้วยอัลกอริทึม AMCL ซึ่ง SLAM จะใช้ข้อมูลสองอย่างคือข้อมูลในการสแกนสิ่งกีดขวางตำแหน่งการเคลื่อนที่ของหุ่นยนต์สำรวจ เพื่อใช้สำหรับสร้างแผนที่ที่เราต้องการและตำแหน่งของหุ่นยนต์สำรวจการสร้างแผนที่จะใช้โปรแกรม Rviz ที่มีอยู่ในระบบ ROS เพื่อใช้ในการจำลองการสร้างแผนที่ของหุ่นยนต์สำรวจ เมื่อได้ข้อมูลจากการเคลื่อนที่ของหุ่นยนต์สำรวจและข้อมูลจากการสแกนสิ่งแวดล้อมที่อยู่รอบ ๆ ตัวหุ่นยนต์สำรวจ ตั้งแต่เริ่มต้นในการสำรวจจนกระทั้งทำการสำรวจเสร็จก็จะได้แผนที่ ซึ่งแผนที่ที่ได้จะเป็นแบบ Gmapping เมื่อทำการสำรวจเสร็จก็จะทำการเซฟไฟล์แผนที่ โดยไฟล์จะมี 2 ไฟล์ ไฟล์จะมีนามสกุล pgm และ yaml ซึ่งไฟล์ pgm จะเป็นรูปภาพส่วนไฟล์ yaml จะเป็นการอธิบายแผนที่ซึ่งผลจากการทดลองพบว่าหุ่นยนต์สำรวจสามารถสำรวจและสร้างแผนที่ได้อย่าถูกต้องตามที่ได้สร้างแบบจำลอง

4.2 การระบุตำแหน่งการเคลื่อนที่อัตโนมัติของหุ่นยนต์สำรวจ

เมื่อทำการสำรวจและสร้างแผนที่เสร็จเรียบร้อยแล้ว ต้องการที่จะให้หุ่นยนต์สำรวจเคลื่อนที่แบบอัตโนมัติจะต้องทำการเรียก AMCL ที่ถูกเก็บไว้ที่ Map Serverc หลังจากทำการเรียกไฟล์ที่สร้างมาจาก SLAM แล้วจะต้องเปิดการทำงานของ amclจะพบว่าโปรแกรมจะอ่านข้อมูลจาก

แผนที่ที่ได้จากการเรียก Launch File ที่ได้สร้างไว้ให้ทำงานในโปรแกรม Rviz ซึ่งจะมีข้อมูลที่สำคัญในการทำให้หุ่นยนต์สำรวจเคลื่อนที่แบบอัตโนมัติ ภาพลูกศรจะใช้ในการกำหนดให้หุ่นยนต์สำรวจทำงาน โดยการกดปุ่ม 2D Nav Goal ซึ่งจะอยู่ด้านบนของโปรแกรม Rviz แล้วนำเมาส์ไปลากที่ Grid ในโปรแกรม Rvizโดยการชี้ไปยังตำแหน่งและทิศทางที่ต้องการ ซึ่งลูกศรคือตำแหน่งของหุ่นยนต์สำรวจที่จะเคลื่อนไปสู่เป้าที่ได้กำหนด

สำหรับการทดลองการเคลื่อนที่อัตโนมัติของหุ่นยนต์สำรวจจะมีการทดลองอยู่ 2 ส่วนคือ ส่วนที่หนึ่งเป็นการทดลองแบบทางตรงและส่วนที่สองเป็นการทดลองการเลี้ยวซ้าย-ขวาของหุ่นยนต์สำรวจแต่การเคลื่อนที่แบบอัตโนมัติในระบบROS จะสามารถเคลื่อนที่เข้าไปในพื้นที่แคบหรือท่อได้อย่างมีประสิทธิภาพโดยที่หุ่นยนต์สำรวจเคลื่อนที่จากที่หนึ่งไปยังอีกที่หนึ่งได้อย่างถูกต้องและแม่นยำจะต้องมีการปรับค่าพารามิเตอร์ในการเคลื่อนที่ เพื่อทำการสร้างเส้นทางหรือค้นหาเส้นทางใหม่ที่มีความปลอดภัยสำหรับหุ่นยนต์สำรวจ โดยการประมวลผลของข้อมูลจากเซ็นเซอร์และแผนที่ที่อยู่รอบๆ ตัวหุ่นยนต์สำรวจซึ่งเป็นการเพิ่มประสิทธิภาพของการเคลื่อนที่แบบอัตโนมัตินี้จะต้องมีการปรับพารามิเตอร์ของ Inflation Radius และ Cost Scaling Factorให้มีความละเอียดและการปรับค่าพารามิเตอร์ในพื้นที่แคบหรือท่อนั้นจะขึ้นอยู่กับความกว้างของการเคลื่อนที่เขาไปในพื้นที่นั้นๆ

ในการทดลองนี้มีจุดประสงค์เพื่อปรับค่าพารามิเตอร์ในการเคลื่อนที่ในท่อที่มีขนาดความกว้างอยู่ที่ 50 เซนติเมตรให้สามารถเคลื่อนได้อย่างมีประสิทธิภาพโดยที่ไม่ชนกับสิ่งกีดขวางที่อยู่รอบๆ ตัวหุ่นยนต์สำรวจ และสามารถนำไปเป็นข้อมูลอ้างอิงในการปรับค่าพารามิเตอร์ในการเคลื่อนที่แบบอัตโนมัติของหุ่นยนต์ที่ต้องการเคลื่อนที่ในพื้นที่แคบหรือท่อให้มีประสิทธิภาพเพิ่มมากขึ้นและในการทดลองนี้ได้ทำการทดลองในการปรับพารามิเตอร์ Inflation Radius และการทดลองการปรับพารามิเตอร์ Cost Scaling Factor เพื่อให้สามารถเคลื่อนที่ในท่อได้อย่างมีประสิทธิภาพดังการทดลองต่อไปนี้

4.2.1 การทดลองในการปรับพารามิเตอร์Inflation Radius

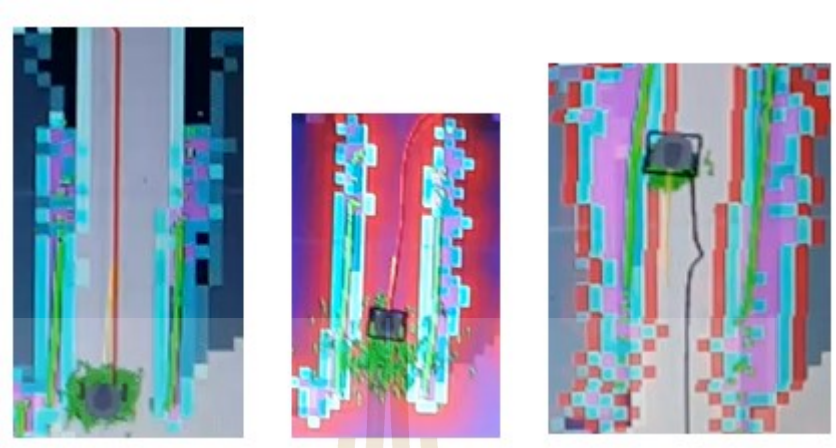
พารามิเตอร์นี้จะทำให้พื้นที่รอบ ๆ ที่ได้จากการสร้างแผนที่ที่อยู่รอบๆ ตัวหุ่นยนต์สำรวจเพิ่มหรือลดขนาดจากการปรับค่าพารามิเตอร์ InflationRadius ซึ่งเป็นส่วนสำคัญในการกำหนดเส้นทางในการเคลื่อนที่ของหุ่นยนต์แบบอัตโนมัติและป้องกันไม่ให้ข้ามพื้นที่นี้ เพื่อความปลอดภัยในการเคลื่อนที่ของหุ่นยนต์สำรวจ ในการตั้งค่านี้จะกำหนดใหญ่กว่ารัศมีหุ่นยนต์สำรวจแต่ระยะของ InflationRadiusจะเป็นการป้องกันปัญหาที่เกิดจากการชนกับสิ่งกีดขวางที่อยู่รอบๆ ตัวของหุ่นยนต์สำรวจ ซึ่งอาจจะทำให้หุ่นยนต์สำรวจเคลื่อนที่เข้าใกล้สิ่งกีดขวางมากเกินไป แต่ในทาง

ตรงกันข้ามถ้าเผื่อระยะมากเกินไปก็อาจจะทำให้หุ่นยนต์สำรวจไม่สามารถเคลื่อนที่ผ่านไปได้ แม้ในความเป็นจริงจะสามารถเคลื่อนที่ผ่านไปได้ก็ตามดังรูปที่4.3(ก)การเผื่อระยะ InflationRadius และ(ข)หุ่นยนต์สำรวจที่อยู่ในสภาพแวดล้อมจริง

(ก) การเผื่อระยะ InflationRadius (ข)หุ่นยนต์สำรวจที่อยู่ในสภาพแวดล้อมจริง

รูปที่ 4.3 การทดลองในการปรับค่า InflationRadius มากเกินไป

จากรูปที่ 4.3(ก) ค่าพารามิเตอร์ของ InflationRadius = 0.35 ซึ่งสิ่งกีดขวางที่อยู่รอบๆ ตัวของหุ่นยนต์สำรวจมีขนาดความกว้างที่ 50 เซนติเมตรจากผลของการทดสอบของการปรับค่าพารามิเตอร์ InflationRadius จะเห็นได้ว่าสีแดงจะเป็นค่าพารามิเตอร์ของ InflationRadiusที่ได้ทำการปรับค่ามากเกินไปจากความเป็นจริงหรือการเผื่อระยะ InflationRadius จึงทำให้หุ่นยนต์สำรวจไม่สามารถเคลื่อนที่ผ่านเข้าไปยังเป้าหมายที่ต้องการได้ เนื่องจากค่าพารามิเตอร์ของ InflationRadius เป็นค่าพารามิเตอร์ที่ป้องกันปัญหาที่เกิดจากการชนกับสิ่งกีดขวางของหุ่นยนต์สำรวจ ดังนั้นในการปรับค่าพารามิเตอร์ที่มากเกินไปจึงทำให้ค่าของ InflationRadius ก็จะมากตามพารามิเตอร์ที่ได้ทำการปรับ จึงทำให้พื้นที่ในการเคลื่อนที่ของหุ่นยนต์สำรวจนั้นแคบทำให้หุ่นยนต์สำรวจไม่สามารถเคลื่อนที่ผ่านไปได้ เพราะหากหุ่นยนต์สำรวจเคลื่อนที่เข้าไปอาจจะทำให้ติดขัดหรือชนกับสิ่งกีดขวางได้ แต่ในความเป็นจริงแล้วหุ่นยนต์สำรวจสามารถเคลื่อนที่ผ่านไปได้ จากรูปที่ 4.4การปรับค่าพารามิเตอร์ของ InflationRadius ให้มีความเหมาะสมในการเคลื่อนที่ในขนาดความกว้าง 50 เซนติเมตร



(ก)InflationRadius = 0.1(ข) InflationRadius = 0.30(ค)InflationRadius = 0.55

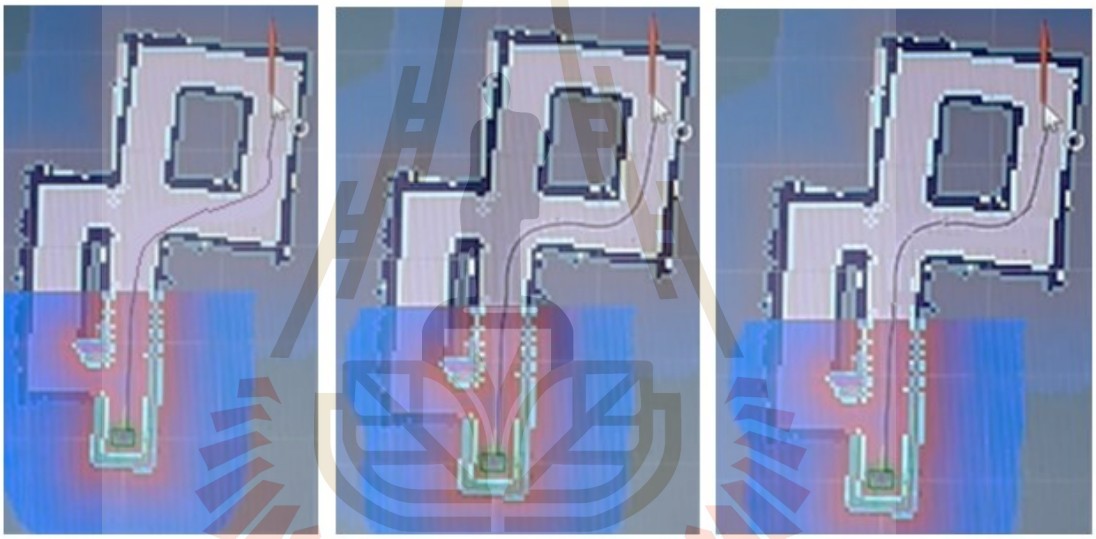
รูปที่ 4.4 การปรับค่าพารามิเตอร์ของ InflationRadius

รูปที่ 4.4 การปรับค่าพารามิเตอร์ InflationRadius ในพื้นที่ขนาดความกว้าง 50 เซนติเมตร จากผลของการทดสอบการปรับค่าพารามิเตอร์ InflationRadius = 0.1จะทำให้หุ่นยนต์สำรวจสามารถเคลื่อนที่เข้าไปถึงเป้าหมายได้ แต่มีโอกาสทำให้ติดขัดหรือชนกับสิ่งกีดขวางได้สูง และการปรับค่าพารามิเตอร์ InflationRadius = 0.30 จะทำให้หุ่นยนต์สำรวจสามารถเคลื่อนที่เข้าไปถึงเป้าหมายได้อย่ามีประสิทธิภาพและมีความปลอดภัยสูง โอกาสที่จะทำให้ติดขัดหรือชนกับสิ่งกีดขวางน้อยมาก ซึ่งเป็นค่าพารามิเตอร์ที่มีความเหมาะสมในการเคลื่อนที่เข้าไปยังท่อที่มีขนาดความกว้าง 50 เซนติเมตรและการทดสอบการปรับค่าพารามิเตอร์ InflationRadius>= 0.55 จะทำให้หุ่นยนต์สำรวจไม่สามารถเคลื่อนที่เข้าไปถึงเป้าหมายได้ ดังที่ได้กล่าวไว้ในรูปที่ 4.3

4.2.2 การทดลองการปรับพารามิเตอร์ Cost Scaling Factor

การทดลองการปรับค่าพารามิเตอร์ Cost Scaling Factor การปรับค่าพารามิเตอร์ต่ำหรือสูงเกินไปจะทำให้ลดคุณภาพของเส้นทางในการเคลื่อนที่ของหุ่นยนต์สำรวจ เส้นทางในการเคลื่อนที่ของหุ่นยนต์สำรวจที่ถูกสร้างขึ้นจะไม่ผ่านจุดกึ่งกลางของอุปสรรคหรือสิ่งกีดขวางที่อยู่รอบ ๆ ในแต่ละด้านและมีความโค้งค่อนข้างน้อย ดังนั้นค่าพารามิเตอร์ที่เป็นกลางจะเป็นค่าที่เหมาะสม สำหรับการปรับค่าพารามิเตอร์ที่อาจจะทำให้เกิดอันตรายต่อหุ่นยนต์สำรวจ คือการปรับค่าพารามิเตอร์ที่ต่ำอาจทำให้เกิดการสร้างเส้นผิดพลาดได้ แม้ว่าเส้นทางที่ได้ทำการสร้างนั้นจะเป็นเส้นทางที่สามารถทำให้หุ่นยนต์สำรวจสามารถเคลื่อนที่ผ่านไปได้ก็ตาม ดังรูปที่ 4.5แสดงถึงการสร้างเส้นทางในการเคลื่อนที่ของหุ่นยนต์สำรวจในท่อที่มีขนาด 50 เซนติเมตรซึ่งเป็นการวางแผนในการสร้างเส้นทางของอัลกอริทึม Global Planner โดยเป็นการปรับค่าค่าพารามิเตอร์ของ Cost

Scaling Factor เส้นสีดำเป็นเส้นทางที่ Global Planner กำหนดเส้นทางในการเคลื่อนที่ของหุ่นยนต์สำรวจ จากการทดลองพบว่าการปรับค่าพารามิเตอร์Cost Scaling Factor = 0.3จะเป็นการสร้างเส้นทางที่ไม่มีความปลอดภัยในการเคลื่อนที่ของของหุ่นยนต์สำรวจ เนื่องจากจะทำให้หุ่นยนต์สำรวจติดขัดหรือชนกับสิ่งกีดขวางได้สูงและการปรับค่าพารามิเตอร์ที่Cost Scaling Factor = 0.85 จะเป็นการสร้างเส้นทางที่มีความปลอดภัยในการเคลื่อนที่ของหุ่นยนต์สำรวจมากที่สุด เนื่องจากจะทำให้หุ่นยนต์สำรวจเคลื่อนที่ไปถึงเป้าหมายได้อย่างถูกต้องแม่นยำและไม่ชนกับสิ่งกีดขวางตลอดการเคลื่อนที่แล้วยังเป็นค่าที่เหมาะสมอีกด้วย ในส่วนของการปรับค่าพารามิเตอร์Cost Scaling Factor = 0.95 จะเป็นการสร้างเส้นทางที่มีความปลอดภัยในการเคลื่อนที่ของของหุ่นยนต์สำรวจ แต่อาจจะทำให้หุ่นยนต์สำรวจมีโอกาสติดขัดหรือชนกับสิ่งกีดขวางได้เช่นกัน



(ก)Cost Scaling Factor = 0.3(ข) Cost Scaling Factor = 0.85 (ค)Cost Scaling Factor = 0.95

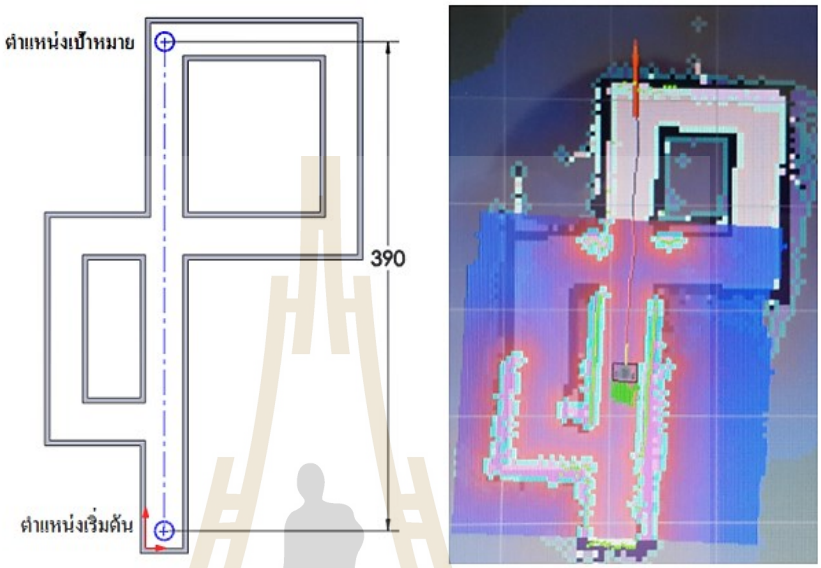
รูปที่ 4.5 การปรับค่าพารามิเตอร์Cost Scaling Factor

4.3 การทดลองการเคลื่อนที่อัตโนมัติจากการปรับพารามิเตอร์ InflationRadius และ Cost Scaling Factor ในท่อ

4.3.1 การทดลองการระบุตำแหน่งการเคลื่อนที่อัตโนมัติของหุ่นยนต์สำรวจแบบทางตรง

สำหรับการทดลองจะกำหนดให้หุ่นยนต์สำรวจเคลื่อนที่แบบอัตโนมัติ ในท่อที่ได้ทำการสร้างแผนที่ไว้แล้วดังรูปที่ 4.2(ก)โดยที่หุ่นยนต์สำรวจจะเริ่มจากตำแหน่งเริ่มต้นจนถึงจุดเป้าหมายที่กำหนด โดยการทดลองจะทำการทดลองการเคลื่อนที่เป็นแบบทางตรงในท่อ ซึ่งมี

ระยะทาง 390 เซนติเมตร ดังรูปที 4.6 การเคลื่อนที่แบบทางตรงในท่อโดยมีระยะทาง 390 เซนติเมตร

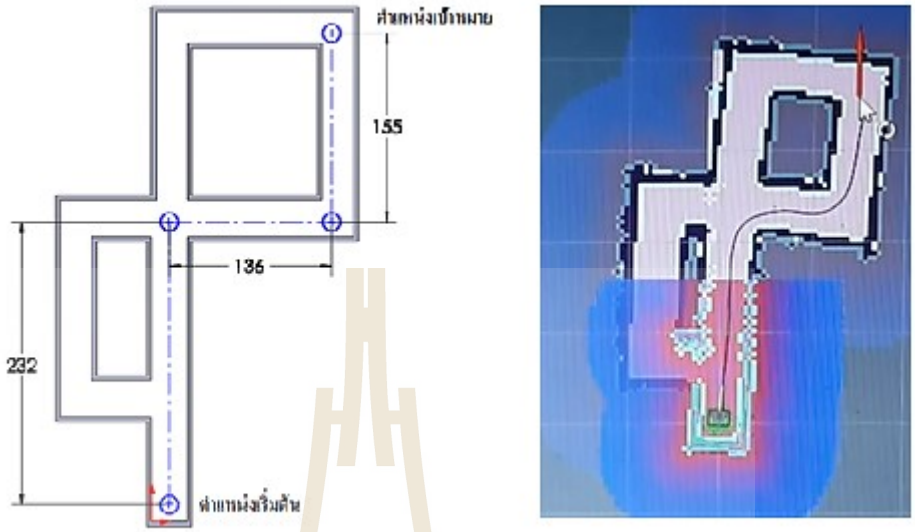


รูปที่ 4.6 การเคลื่อนที่แบบทางตรงในท่อโดยมีระยะทาง 390 เซนติเมตร

จากรูปที่ 4.6 จากการทดลองพบว่าการเคลื่อนที่แบบทางตรงในท่อโดยมีระยะทาง 390 เซนติเมตร ซึ่งมีขนาดความกว้างของท่อ 50 เซนติเมตรและได้ทำการปรับค่าพารามิเตอร์ Inflation Radius = 0.30และ Cost Scaling Factor = 0.85 สามารถทำให้หุ่นยนต์สำรวจเคลื่อนที่แบบอัตโนมัติได้อย่างมีประสิทธิภาพมากที่สุด และได้ทำการทดลองในการเคลื่อนที่จำนวน 8 ครั้ง สามารถเคลื่อนที่ไปยังเป้าหมายที่ระยะทาง 390 เซนติเมตรได้อย่างแม่นยำ โดยที่หุ่นยนต์สำรวจไม่ชนกับสิ่งกีดขวางที่อยู่รอบๆ ของตัวหุ่นยนต์สำรวจ

4.3.2 การทดลองการเคลื่อนที่แบบเลี้ยวซ้าย-ขวาในท่อ โดยมีระยะทางทั้งหมด 523 เซนติเมตร

สำหรับการทดลองจะกำหนดให้หุ่นยนต์สำรวจเคลื่อนที่แบบอัตโนมัติ ในท่อที่ได้ทำการสร้างแผนที่ไว้แล้วดังรูปที่ 4.2(ก)โดยที่หุ่นยนต์สำรวจจะเริ่มจากตำแหน่งเริ่มต้นจนถึงจุดเป้าหมายที่กำหนด โดยการเคลื่อนที่เป็นแบบทางตรง ซึ่งมีระยะทาง 232 เซนติเมตร เลี้ยวขวาเคลื่อนที่เป็นแบบทางตรงด้วยระยะทาง 136 เซนติเมตร และเลี้ยวซ้ายเคลื่อนที่เป็นแบบทางตรงด้วยระยะทาง 155 เซนติเมตร ซึ่งรวมระยะทางในการเคลื่อนที่ทั้งหมดเป็นระยะทาง 523 เซนติเมตร ดังรูปที 4.7 การเคลื่อนที่แบบเลี้ยวขวา-ซ้ายในท่อระยะทาง 523 เซนติเมตร



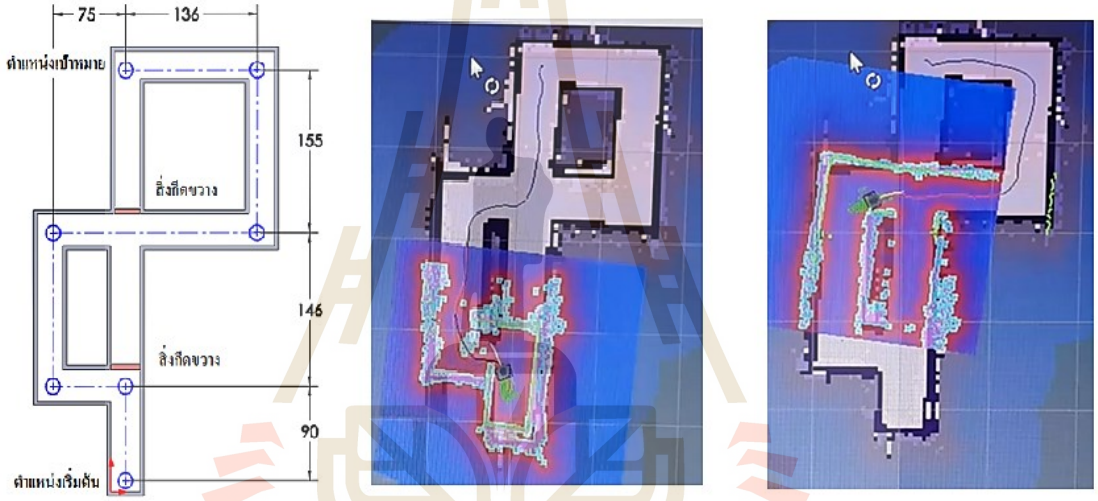
รูปที่ 4.7 การเคลื่อนที่แบบเลี้ยวขวา-ซ้ายในท่อระยะทาง 523เซนติเมตร

จากรูปที่ 4.7จากการทดลองพบว่าการทดลองการเคลื่อนที่แบบเลี้ยวขวา-ซ้ายในท่อโดยมีระยะทาง 523 เซนติเมตร ซึ่งมีขนาดความกว้างของท่อ 50 เซนติเมตรและได้ทำการปรับค่าพารามิเตอร์ Inflation Radius = 0.30 และ Cost Scaling Factor = 0.85 สามารถทำให้หุ่นยนต์สำรวจเคลื่อนที่แบบอัตโนมัติได้อย่างมีประสิทธิภาพแต่เส้นทางในการเคลื่อนที่แบบเลี้ยวขวา-เลี้ยวซ้ายอาจจะทำให้หุ่นยนต์สำรวจเกิดการเคลื่อนที่ช้าและเร็วตามเส้นทางที่ได้ทำการสร้างไว้ ซึ่งผลจากการทดลองในการเคลื่อนที่จำนวน 8 ครั้ง สามารถเคลื่อนที่ไปยังเป้าหมายที่ระยะทาง 523 เซนติเมตรได้อย่างแม่นยำ โดยที่หุ่นยนต์สำรวจไม่ชนกับสิ่งกีดขวางที่อยู่รอบๆ ของตัวหุ่นยนต์สำรวจ

จากผลการทดลองการเคลื่อนที่ของหุ่นยนต์สำรวจจากตำแหน่งเริ่มต้นไปยังตำแหน่งเป้าหมาย ระยะทาง 390 เซนติเมตรและระยะทาง 523 เซนติเมตร โดยการเคลื่อนที่ในท่อแบบทางตรงและการเคลื่อนที่แบบเลี้ยวซ้าย-ขวา แบบไม่มีสิ่งกีดขวางในการเคลื่อนที่แสดงดังรูปที่ 4.6 และรูปที่ 4.7 พบว่าสามารถเคลื่อนที่ไปยังเป้าหมายได้อย่างถูกต้องโดยที่ลูกศรสีแดงคือตำแหน่งเป้าหมายในการเคลื่อนที่ของหุ่นยนต์สำรวจ เส้นทึบสีดำคือเส้นทางที่สร้างขึ้นด้วยโปรแกรม Rviz ที่อยู่ในระบบ ROS เป็นเส้นทางที่หุ่นยนต์สำรวจใช้ในการเคลื่อนที่ไปยังตำแหน่งเป้าหมายที่ต้องการ

4.4 การหลบหลีกสิ่งกีดขว้างในพื้นที่แคบ

สำหรับการทดลองนี้จะกำหนดให้หุ่นยนต์สำรวจเคลื่อนที่แบบอัตโนมัติ ในท่อที่ได้ทำการสร้างแผนที่ไว้แล้วดังรูปที่ 4 (ก)โดยที่หุ่นยนต์สำรวจจะเริ่มเคลื่อนที่จากตำแหน่งเริ่มต้นจนถึงจุดเป้าหมายที่กำหนด โดยเส้นทางในการเคลื่อนที่ของหุ่นยนต์สำรวจนั้น จะทำการปิดทางสามแยกของท่อที่ระยะทาง 115 เซนติเมตรและทำการปิดเส้นทางสี่แยกของท่อที่ระยะทาง 275 เซนติเมตร ซึ่งเส้นทางในการเคลื่อนที่รวมระยะทางที่ของหุ่นยนต์สำรวจจะต้องเคลื่อนที่ไปให้ถึงป้าหมายทั้งหมดเป็นระยะทาง 813 เซนติเมตรดังรูปที่ 4.8 การเคลื่อนที่ของหุ่นยนต์สำรวจในการหลบหลีกสิ่งกีดขวางในท่อ



รูปที่ 4.8 การเคลื่อนที่ของหุ่นยนต์สำรวจในการหลบหลีกสิ่งกีดขวางในท่อ

รูปที่ 4.8 จากการทดลองพบว่าการหลบหลีกสิ่งกีดขว้างในพื้นที่แคบในท่อโดยมีระยะทาง 813 เซนติเมตร ซึ่งมีขนาดความกว้างของท่อ 50 เซนติเมตร และได้ทำการปรับค่าพารามิเตอร์ Inflation Radius = 0.30และ Cost Scaling Factor = 0.85 สามารถทำให้หุ่นยนต์สำรวจเคลื่อนที่แบบอัตโนมัติได้อย่างมีประสิทธิภาพ และเมื่อหุ่นยนต์สำรวจตรวจพบกับสิ่งกีดขวางที่ทำให้หุ่นยนต์สำรวจไม่สามารถเคลื่อนที่ผ่านไปได้ หุ่นยนต์สำรวจจะทำการสร้างเส้นทางที่สามารถทำให้ไปถึงยังเป้าหมายได้ ซึ่งเส้นทางในการเคลื่อนที่หลบหลีกสิ่งกีดขวางในท่อจะทำให้หุ่นยนต์สำรวจเกิดการเคลื่อนที่ช้าและเร็วตามเส้นทางที่ได้ทำการสร้างไว้ จากการทดลองในการเคลื่อนที่จำนวน 5ครั้ง สามารถเคลื่อนที่ไปยังเป้าหมายที่ระยะทาง 813 เซนติเมตรได้อย่างแม่นยำ โดยที่หุ่นยนต์สำรวจไม่ชนกับสิ่งกีดขวางที่อยู่รอบๆ ของตัวหุ่นยนต์สำรวจและสามารถสร้างเส้นทางใหม่เมื่อตรวจพบว่ามีสิ่งกีดขวางอยู่ในท่อได้

จากผลการทดลองการเคลื่อนที่ของหุ่นยนต์สำรวจจากตำแหน่งเริ่มต้นไปยังตำแหน่งเป้าหมายในการหลบหลีกสิ่งกีดขว้างในพื้นที่แคบด้วยระยะทาง 813 เซนติเมตร โดยเส้นทางในการเคลื่อนที่ของหุ่นยนต์สำรวจนั้น ปิดทางสามแยกของท่อที่ระยะทาง 115 เซนติเมตร และปิดเส้นทางสี่แยกของท่อที่ระยะทาง 275 เซนติเมตร ซึ่งในการเคลื่อนที่แสดงดังรูปที่ 4.8 โดยลูกศรสีแดงคือตำแหน่งเป้าหมายในการเคลื่อนที่ของหุ่นยนต์สำรวจ เส้นทึบสีดำคือเส้นทางที่สร้างขึ้นด้วยโปรแกรม Rviz ที่อยู่ในระบบ ROS เป็นเส้นทางที่หุ่นยนต์สำรวจทำการเคลื่อนที่ไปยังตำแหน่งเป้าหมายที่ต้องการ จะเห็นได้ว่าเมื่อหุ่นยนต์สำรวจเคลื่อนที่เข้าใกล้สิ่งกีดขวางนั้น การควบคุมการทำงานของ Global Costmap และ Local Costmap ที่ได้ออกแบบเส้นทางในการเคลื่อนที่ของหุ่นยนต์สำรวจ จะทำหน้าที่ค้นหาเส้นทางและสร้างเส้นทางในการหลบหลีกสิ่งกีดขวางเพื่อไปยังตำแหน่งเป้าหมาย โดยที่หุ่นยนต์สำรวจสามารถสร้างเส้นทางการเคลื่อนที่ใหม่เมื่อมีสิ่งกีดขวางอยู่เพื่อไปยังเป้าหมายตามเส้นทางที่ต้องการได้

บทที่ 5
สรุปผลการทดลองและข้อเสนอแนะ

5.1 สรุปวัตถุประสงค์ของงานวิจัยและวิธีการดำเนินการวิจัย

ในงานวิจัยนี้ต้องการพัฒนาหุ่นยนต์สำรวจในพื้นที่ปิด ที่สามารถทำงานได้อย่างอัตโนมัติ ซึ่งสามารถนำไปประยุกต์ใช้ในการสำรวจพื้นที่ประสพภัย อาคารร้าง หรือในถ้ำที่คนไม่สามารถเข้าไปสำรวจได้ ด้วยวิธี SLAM โดยใช้ระบบปฏิบัติการหุ่นยนต์ ROS ร่วมกับเซนเซอร์ Lidar ในการตรวจจับสิ่งกีดขวางหรือวัตถุที่อยู่รอบ ๆ โดยการสร้างแผนที่นั้นจะใช้วิธีการ SLAM ของ ROS ซึ่งจะสร้างแผนที่โดยการส่งเป็น Topic แล้วทำการจัดเก็บไว้ใน Map Server ในส่วนของการรู้ตำแหน่งหรือ Localization สำหรับ ROS มีอัลกอริทึมที่เรียกว่า AMCL (Adaptive Monte Carlo Localization) ซึ่งจะทำให้รู้ว่าหุ่นยนต์อยู่จุดใดในแผนที่ ในการทำงานของหุ่นยนต์สำรวจอัตโนมัติด้วยวิธี SLAM นั้น แบ่งออกเป็นสองส่วนคือ ส่วนแรกการสำรวจและสร้างแผนที่ และส่วนที่สองระบุตำแหน่งการเคลื่อนที่อัตโนมัติของหุ่นยนต์สำรวจ

หุ่นยนต์สำรวจได้ทำการติดตั้ง เซนเซอร์ Lidar, บอร์ด OpenCR, บอร์ด Raspberry Pi, มอเตอร์ Dynamixel และแบตเตอรี่ เพื่อให้สามารถทำงานได้ตามวัตถุประสงค์ จากนั้นได้ทำการศึกษาอัลกอริทึมต่างๆ และได้มีการจำลองการทำงานในระบบของ ROS ที่มีโปรแกรม Rviz พร้อมทั้งหาค่าตัวแปรที่สำคัญของเซนเซอร์ที่ใช้งานอีกด้วย เมื่อทำการตรวจสอบการทำงานของระบบจนครบถ้วนสมบรูณ์แล้ว จึงได้มีการทดสอบการทำงานของระบบ ROS เพื่อใช้ในการตรวจสอบสิ่งกีดขวางและทำการสำรวจพื้นที่ที่ต้องการสำรวจแล้วทำการสร้างแผนที่ในสภาพแวดล้อมที่ได้จำลองในการทำงานและปรับปรุงให้มีประสิทธิภาพต่อไป

5.2 สรุปผลการทดลองและอภิปรายผล

ในการทดลองนี้จะแบ่งออกเป็นสองส่วนคือ ส่วนแรกจะเป็นการสำรวจและสร้างแผนที่และส่วนที่สองจะเป็นการระบุตำแหน่งการเคลื่อนที่อัตโนมัติของหุ่นยนต์สำรวจในส่วนแรกจะเป็นการสำรวจและสร้างแผนที่จากแบบจำลองของท่อที่ได้ทำการสร้างขึ้นเอง โดยการใช้ระบบปฏิบัติการหุ่นยนต์ ROS ร่วมกับเซนเซอร์ Lidar ในการตรวจจับสิ่งกีดขวางหรือวัตถุที่อยู่รอบ ๆ โดยการสร้างแผนที่นั้นจะใช้วิธีการ SLAM Gmappingผลที่ได้คือสามารถสำรวจ และสร้างแผนที่ได้ตรงตามตำแหน่งที่แท้จริง จากนั้นได้มีการทดสอบการระบุตำแหน่งการเคลื่อนที่

อัตโนมัติของหุ่นยนต์สำรวจด้วยอัลกอริทึมที่เรียกว่า AMCL (Adaptive Monte Carlo Localization) ซึ่งจะทำให้รู้ว่าหุ่นยนต์สำรวจอยู่จุดใดในแผนที่ และสามารถเคลื่อนที่ไปได้ทุกที่ที่ต้องการในแผนที่ขั้นตอนในการเคลื่อนที่แบบอัตโนมัติของหุ่นยนต์สำรวจคือจะต้องรู้ถึงลักษณะของสิ่งกีดขวางหรือวัตถุที่อยู่รอบ ๆ เพื่อให้หุ่นยนต์สำรวจสามารถค้นหาเส้นทางในการเคลื่อนที่โดยไม่ติดขัดหรือชนกับสิ่งกีดขวาง และสามารถทำให้หุ่นยนต์สำรวจไปถึงเป้าหมายตามที่ต้องการได้อย่างถูกต้องแม่นยำ ซึ่งลักษณะของสิ่งกีดขวางหรือวัตถุที่อยู่รอบ ๆ ตัวหุ่นยนต์สำรวจนั้น ROS จะใช้เทคนิคที่เรียกว่า Costmap หรือแผนที่ที่ได้ทำการสร้างไว้แล้ว เพื่อใช้ในการเคลื่อนที่แบบอัตโนมัติ ดังนั้นเมื่อหุ่นยนต์สำรวจรู้ถึงสภาพแวดล้อมที่อยู่รอบๆ แล้วก็สามารถวางแผนมนการในการเคลื่อนที่แบบอัตโนมัติได้ด้วยอัลกอริทึมการวางแผนหรือที่เรียกว่า Planner ที่มีอยู่ใน Move Base ในการวางแผนการเคลื่อนที่จะมีอยู่ 2 ระดับ ได้แก่ การวางแผนในภาพรวม (Global Planner) และการวางแผนระยะใกล้ (Local Planner) โดยที่ Planner ทั้งสองประเภทนี้จะทำการคำนวณเพื่อสร้างแผนของเส้นทาง แล้วทำการส่งข้อมูลเพื่อนำไปควบคุมตัวหุ่นยนต์สำรวจในการเคลื่อนที่แบบอัตโนมัติ

จากการทดลองพบว่าการเคลื่อนที่ในท่อที่เป็นแบบจำลองที่สร้างขึ้นเองโดยผู้วิจัยนั้น ในการทดลองที่ 1 การสำรวจและการสร้างแผนที่ ผลที่ได้คือหุ่นยนต์สำรวจสามารถสำรวจและสร้างแผนที่ได้ตามที่ต้องการ โดยการทำงานจะใช้โปรแกรม Rvizที่อยู่ในระบบ ROS เป็นตัวสร้างแผนที่และใช้แป้นคีย์บอร์ดในการบังคับควบคุมสั่งงานให้หุ่นยนต์สำรวจทำงาน ซึ่งในการสำรวจและสร้างแผนที่นั้นจะทำงานได้เร็วหรือช้าขึ้นอยู่กับผู้ที่ใช้งาน ส่วนการทดลองที่ 2 การระบุตำแหน่งการเคลื่อนที่อัตโนมัติของหุ่นยนต์สำรวจจากการทดลองพบว่าหุ่นยนต์สำรวจจะสามารถเคลื่อนที่เข้าไปในท่อได้นั้นมีความจำเป็นที่จะต้องมีการปรับค่าพารามิเตอร์Inflation Radius และ Cost Scaling Factor เพื่อเพิ่มประสิทธิภาพของการเคลื่อนที่แบบอัตโนมัติ เนื่องจากหุ่นยนต์สำรวจเคลื่อนที่เข้าไปในท่อ พื้นที่ประสพภัย อาคารร้าง หรือในถ้ำที่คนไม่สามารถเข้าไปสำรวจได้นั้น มีความเสี่ยงหรืออาจจะทำให้หุ่นยนต์สำรวจเกิดการเสียหายได้และเป็นการป้องกันในการชนกับสิ่งที่กีดขวางที่อยู่รอบๆ ตัวหุ่นยนต์สำรวจด้วย ซึ่งผลที่ได้จากการทดลองในการเคลื่อนที่ของหุ่นยนต์สำรวจในท่อที่มีขนาดความกว้าง 50 เซนติเมตรนี้พบว่าการปรับค่าพารามิเตอร์Inflation Radius ที่มีค่าน้อยกว่า 0.10 นั้นจะทำให้เกิดความเสี่ยงในการเคลื่อนที่ของหุ่นยนต์สำรวจเพราะอาจจะทำให้หุ่นยนต์สำรวจนั้นชนกับสิ่งกีดขวางได้โดยง่ายและอาจจะทำให้หุ่นยนต์สำรวจเกิดความเสียหายได้ และการปรับค่าพารามิเตอร์Inflation Radius ที่มีค่ามากกว่า 0.50 นั้นจะทำให้หุ่นยนต์สำรวจไม่สามารถเคลื่อนที่ผ่านเข้าไปได้ เพราะว่าหุ่นยนต์สำรวจจะคิดว่าเป็นทางที่แคบไม่สามารถเคลื่อนที่เข้าไปได้ แต่ในความเป็นจริงแล้วสามารถเคลื่อนที่ผ่านเข้าไปได้ก็ตาม จากการทดลองในการปรับ

ค่าพารามิเตอร์ Inflation Radius ให้ได้มีประสิทธิภาพมากที่สุดในการเคลื่อนที่ในท่อขนาด 50 เซนติเมตร จะอยู่ในช่วงระหว่าง 0.1< 0.3 > 0.5 ซึ่งค่า Inflation Radius= 0.3 จะเป็นค่าที่มีประสิทธิภาพมากที่สุด ในส่วนของการปรับค่าพารามิเตอร์Cost Scaling Factor จะเป็นค่าพารามิเตอร์ที่สร้างเส้นทางในการเคลื่อนที่แบบอัตโนมัติของหุ่นยนต์สำรวจ เพื่อให้ไปถึงเป้าหมายที่ต้องการได้อย่างปลอดภัยโดยที่ไม่ชนกับสิ่งกีดขวาง จากการทดลองพบว่าค่าพารามิเตอร์ Cost Scaling Factor ที่น้อยกว่า0.3 จะเป็นการสร้างเส้นทางที่ไม่มีความปลอดภัยในการเคลื่อนที่ของของหุ่นยนต์สำรวจ เนื่องจากการเคลื่อนที่ตามเส้นทางที่ถูกสร้างนั้น ตัวหุ่นยนต์สำรวจอาจจะติดขัดหรือชนกับสิ่งกีดขวางได้สูง และการปรับค่าพารามิเตอร์ Cost Scaling Factor ที่มากกว่า0.95 จะเป็นการสร้างเส้นทางที่มีความโค้งมากเกินไป อาจจะทำให้ไม่มีความปลอดภัยในการเคลื่อนที่ของของหุ่นยนต์สำรวจได้เช่นกัน จากการทดลองในการปรับค่าพารามิเตอร์Cost Scaling Factor ให้ได้มีประสิทธิภาพมากที่สุดในการเคลื่อนที่ในท่อขนาด 50 เซนติเมตร จะอยู่ในช่วงระหว่าง 0.3< 0.85 > 0.95 ซึ่งค่า Cost Scaling Factor = 0.85 จะเป็นค่าที่มีประสิทธิภาพมากที่สุดในการเคลื่อนที่ในท่อที่มีขนาดความกว้าง 50 เซนติเมตร

ในส่วนของการทดลองในการเคลื่อนที่ในท่อระยะทาง 390 เซนติเมตรแบบเลี้ยวขวา-ซ้าย ในท่อระยะทาง 523 เซนติเมตรและการเคลื่อนที่ของหุ่นยนต์สำรวจในการหลบหลีกสิ่งกีดขวางในท่อระยะทาง 813 เซนติเมตร ผลที่ได้คือ เมื่อทำการปรับค่าพารามิเตอร์ของ Inflation Radius = 0.3 และ Cost Scaling Factor= 0.85 ทำให้การเคลื่อนที่ในท่อของหุ่นยนต์สำรวจสามารถเคลื่อนที่ได้อย่างมีประสิทธิภาพมากขึ้นและสามารถค้นหาเส้นทางและสร้างเส้นทางในการหลบหลีกสิ่งกีดขวางเพื่อไปยังตำแหน่งเป้าหมายได้

จากผลการทดลองนี้สามารถทำได้ตามวัตถุประสงค์และขอบเขตที่วางไว้ แต่เพื่อให้มีประสิทธิภาพมายิ่งขึ้นไป จะต้องมีการพัฒนาต่อทั้งในด้านของระบบประมวลผลให้มีความไวมากยิ่งขึ้น หลังจากนั้นอาจจะต้องมีการทดลองในสภาพแวดล้อมที่มีความซับซ้อนมากขึ้น พื้นผิวที่ไม่เรียบหรือสภาพแวดล้อมจริง เพื่อตรวจสอบว่าอัลกอริทึมหรือค่าพารามิเตอร์ที่ได้ทำการทดลองสามารถทำงานได้อย่างถูกต้องหรือไม่ และสุดท้ายจากการทดลองคือหุ่นยนต์สำรวจสามารถเคลื่อนที่เข้าไปในท่อได้

5.3 ปัญหาและข้อเสนอแนะ

ปัญหาในงานวิจัยครั้งนี้ประกอบไปด้วย

- ข้อมูลที่ได้จากการเคลื่อนที่มีความคลาดเคลื่อนไม่ตรงกับหุ่นยนต์สำรวจ โดยเฉพาะเมื่อหุ่นยนต์สำรวจเคลื่อนที่หลายๆ รอบในแผนที่ ซึ่งเป็นปัญหาในการเคลื่อนที่ของหุ่นยนต์สำรวจแบบอัตโนมัติ สามารถแก้ปัญหาได้ด้วยการใช้ Package ในการกรองข้อมูลที่ดีกว่า เช่น EKF เป็นต้น ซึ่งจะรวมถึงเซนเซอร์ที่เกี่ยวข้องในการเคลื่อนที่ของหุ่นยนต์สำรวจด้วย เช่น เซนเซอร์ในการเคลื่อนที่ของล้อหรือตัวหุ่นยนต์สำรวจ เซนเซอร์ IMU เป็นต้น

- ลักษณะของแผนที่ Costmapมีความพอดีกับขนาดของตัวหุ่นยนต์สำรวจมากเกินไปทำให้การเคลื่อนที่เข้าไปของหุ่นยนต์สำรวจเกิดการติดขัดได้ ในกรณีที่ทางแคบไปหุ่นยนต์สำรวจก็จะไม่สามารถผ่านไปได้หรือกว้างมากเกินไปหุ่นยนต์สำรวจก็จะสามารถผ่านไปได้ แต่ในกรณีที่ความกว้างมีความพอดีกับขนาดของหุ่นยนต์สำรวจอาจจะเกิดปัญหานี้ได้

- ความผิดพลาดอื่นๆ เช่น เซนเซอร์ต่างๆ, เซนเซอร์ Lidar, บอร์ด OpenCR, บอร์ด Raspberry Pi หรือคอมพิวเตอร์ ทำการประมวลผลไม่ทัน ซึ่งวิธีในการแก้ไขนั้นจะต้องดูไปตามกรณี ซึ่งเป็นข้อดีของ ROS Navigation ซึ่งจะมีข้อความในการเตือนขึ้นมาในกรณีที่หุ่นยนต์สำรวจนั้นมีความเสี่ยงต่อความผิดพลาดในการทำงาน ซึ่งจำเป็นที่จะต้องอ่านข้อความที่มีการแจ้งเตือนต่างๆ ทีปรากฏขึ้นมา เพื่อป้องกันปัญหาหรือความผิดพลาดที่อาจเกิดขึ้น

ข้อเสนอแนะ

- การกำหนดค่าพารามิเตอร์ต่างๆ ในไฟล์ Configuration มีความสำคัญมาก จะต้องกำหนดค่าให้ตรงกับความเป็นจริงของหุ่นยนต์สำรวจ เช่น ค่าความเร็วสูงสุด, ค่าความเร็วต่ำสุด, ค่าความเร่งสูงสุด, ค่าความเร็วเชิงมุมเป็นต้น

- ในการกำหนดค่าบางอย่างในช่วงแรกควรกำหนดเผื่อไว้ เช่น การกำหนด Footprint ของตัวหุ่นยนต์สำรวจ, ระยะของ Inflation เพื่อป้องกันปัญหาที่เกิดจากการชนกับสิ่งกีดขวางที่อยู่รอบ ๆ ตัวของหุ่นยนต์สำรวจ หรืออาจทำให้หุ่นยนต์สำรวจเคลื่อนที่เข้าใกล้มากเกินไป แต่ในทางตรงกันข้ามถ้าเผื่อระยะมากเกินไปก็อาจจะทำให้หุ่นยนต์สำรวจไม่สามารถเคลื่อนที่ผ่านไปได้ แม้ในความเป็นจริงจะสามารถเคลื่อนที่ผ่านไปได้ก็ตาม

- การเปิดและปิด roscore ใหม่สามารถแก้ปัญหาบางอย่างในระหว่างการทำงานได้ เช่น การจดจำค่าเก่าซึ่งทำให้โปรแกรมที่ทำงานใหม่จะทำงานไม่ถูกต้องตรงตามเวลาจริง

- การปิดโปรแกรม Rvizจะช่วยลดการประมวลผลของคอมพิวเตอร์ได้ และความเร็วหรือเสปกของคอมพิวเตอร์มีผลต่อการทำงานของระบบ ถ้าเป็นไปได้ควรจัดหาคอมพิวเตอร์ที่มีเสปกที่สูงเพื่อให้การทำงานของระบบนั้นสามารถประมวลผลได้อย่างมีประสิทธิภาพและทันต่อเวลาจริง

รายการอ้างอิง

[1] X. Li, X. Huang, and M. Wang. Robot Map Building From Sonar Sensors and DSmT. Information & Security. An International Journal, 20:104–121, 2006

[2] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: an open-source Robot Operating System. In ICRA Workshop on Open Source Software, 2009

[3] YoonSeokPyo, HanCheol Cho, RyuWoon Jung, TaeHoon Lim, “ROS Robot Programming,” Republic of Korea, ROBOTIS Co.,Ltd, 2017

[4]B. L. E. A. Balasuriya et al., "Outdoor robot navigation using Gmapping based SLAM algorithm," 2016 Moratuwa Engineering Research Conference (MERCon), Moratuwa, 2016, pp. 403-408.

[5] S. Zaman, W. Slany and G. Steinbauer, "ROS-based mapping, localization and autonomous navigation using a Pioneer 3-DX robot and their relevant issues," 2011 Saudi International Electronics, Communications and Photonics Conference (SIECPC), Riyadh, 2011, pp. 1-5.

[6] M. P. Imthiyas. Indoor Environment Mobile Robot Localization International Journal on Computer Science and Engineering, 2(3):714– 719, 2010

[7] S. Kohlbrecher, O. von Stryk, J. Meyer and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation," 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics, Kyoto, 2011, pp. 155-160.

[8] K. O. Arras, S. Grzonka, M. Luber and W. Burgard, "Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities," 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, 2008, pp. 1710-1715.

.[9] I. Z. Ibragimov and I. M. Afanasyev, "Comparison of ROS-based visual SLAM methods in homogeneous indoor environment," 2017 14th Workshop on Positioning, Navigation and Communications (WPNC), Bremen, 2017, pp. 1-6.

[10] Rui-Jun Yan; Jing Wu; Sung-Jin Lim; Ji-Yeong Lee; Chang-Soo Han, " Natural corners-based SLAM in unknown indoor environment," Ubiquitous Robots and Ambient Intelligence (URAI), vol., no., pp.259,261, 26-28 Nov. 2012

[11] Yang, P., " Efficient particle filter algorithm for ultrasonic sensor based 2D range-only simultaneous localization and mapping application, " Wireless Sensor Systems, IET, vol.2, no.4, pp. 394,401, December 2012

[12] Z. Zhang, H. Guo, G. Nejat and P. Huang, "Finding Disaster Victims: A Sensory System for Robot-Assisted 3D Mapping of Urban Search and Rescue Environments," Proceedings 2007 IEEE International Conference on Robotics and Automation, Roma, 2007, pp. 3889-3894.

[13] http://http://www.willowgara.com/robot/overview

[14] K.O. Arras, S.Grzonka, M. Luber, and W. Burgard, "Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities," in Proc. IEEE Int. Conf. Robotics and Automation ICRA 2008, 2008, pp. 1710-1715

[15] N. A. Tsokas and K. J. Kyriakopoulos, "A multiple hypothesis people tracker for teams of mobile robots," in Proc. IEEE Int Robotics and Automation (ICRA) Conf, 2010, pp. 446-451

[16] M. Luber, G. D. Tipaldi, and K. O. Arras, "Better models for people tracking," in Robotics and Automation (ICRA), 2011 IEEE International Conference on, 2011, pp. 854-859

[17] M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," Proceedings of the National Conference on Artificial Intelligence, 2002.

[18] M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, "FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges," In Proc. of the Int. Conf. on Artificial Intelligence (IJCAI), 2003.

[19] J. Weingarten, "EKF-based 3D SLAM for Structured Environment Reconstruction," in Proceedings of IROS, 2005.

[20] A. J. Davison and D. W. Murray, "Simultaneous localization and map-building using active vision," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002.

[21] J. Leonard and P. Newman, "Consistent, convergent, and constant-time slam," International Joint Conference on Artificial Intelligence (IJCAI), 2003.

[22] M. R. Walter, R. M. Eustice and J. J. Leonard, "Exactly Sparse Extended Information Filters for Feature-based SLAM," " The International Journal of Robotics Research, 2007.

[23] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani and H. F. Durrant-Whyte, "Simultaneous Localization and Mapping with Sparse Extended Information Filters," International Journal of Robotics Research, 2004.

[24] U. Frese and G. Hirzinger, "Simultaneous localization and mapping – a discussion," Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robotics, 2001.

ประวัติผู้เขียน

นายสวาส อาจสาลี เกิดเมื่อวันที่ 6 กุมภาพันธ์ พ.ศ. 2527 เริ่มศึกษาชั้นประถมที่โรงเรียนบ้านโนนสูง ชั้นมัธยมศึกษาที่โรงเรียนภู่วิทยาสำเร็จการศึกษาวิชาชีพ ปวส. ที่วิทยาลัยเทคโนโลยีช่างกลพาณิชยการนครราชสีมาจังหวัดนครราชสีมาและสำเร็จการศึกษาระดับปริญญาตรี สาขาวิชาวิศวกรรมเมคคาทรอนิกส์ มหาวิทยาลัยวงษ์ชวลิตกุล จังหวัดนครราชสีมา เมื่อปี พ.ศ.2552 โดยหลังจากสำเร็จการศึกษาได้เริ่มทำงานที่มหาวิทยาลัยวงษ์ชวลิตกุล ตำแหน่งอาจารย์ปฏิบัติการสาขาวิศวกรรมเมคคาทรอนิกส์และเป็นผู้ช่วยสอน คณะวิศวกรรมศาสตร์ มหาวิทยาลัยวงษ์ชวลิตกุล

ปี พ.ศ.2557 เข้าศึกษาต่อในระดับปริญญาโทสาขาวิชาวิศวกรรมเมคคาทรอนิกส์ มหาวิทยาลัยเทคโนโลยีสุรนารีโดยขณะศึกษาได้รับทุนผู้ช่วยสอนจากมหาวิทยาลัยวงษ์ชวลิตกุล