ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมเครื่องกล)

ปริญญา

วิศวกรรมเครื่องกล	วิศวกรรมเครื่องกล
สาขา	**ภาควิชา**

เรื่อง การออกแบบหุ่นยนต์สำรวจท่อแอร์

Design of Air-duct Surveying Robot

นามผู้วิจัย นายกาญจนะ ชาวบ้านเกาะ

ได้พิจารณาเห็นชอบโดย

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก .....

(อาจารย์วิทิต ฉัตรรัตนกุลชัย, Ph.D.)

อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม .....

(อาจารย์คุณยุต เอี่ยมสอาด, Ph.D.)

หัวหน้าภาควิชา .....

(รองศาสตราจารย์ชวลิต กิตติชัยการ, Ph.D.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

.....

(รองศาสตราจารย์กัญจนา ธีระกุล, D.Agr.)

คณบดีบัณฑิตวิทยาลัย

วันที่ ..... **เดือน** ..... **พ.ศ.** .....

วิทยานิพนธ์

เรื่อง

การออกแบบหุ่นยนต์สำรวจท่อแอร์

Design of Air-duct Surveying Robot

โดย

นายกาญจนะ ชาวบ้านเกาะ

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์
เพื่อความสมบูรณ์แห่งปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมเครื่องกล)
พ.ศ. 2552

กาญจนะ ชาวบ้านเกาะ 2552: การออกแบบหุ่นยนต์สำรวจท่อแอร์
ปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมเครื่องกล) สาขาวิศวกรรมเครื่องกล
ภาควิชาวิศวกรรมเครื่องกล อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก:
อาจารย์วิทิต ฉัตรรัตนกุลชัย, Ph.D. 103 หน้า

วิทยานิพนธ์นี้ได้ทำการออกแบบและสร้างหุ่นยนต์สำรวจท่อแอร์ โดยการประยุกต์ระบบขับเคลื่อนแบบสายพานเข้ากับระบบขับเคลื่อนแบบรถยนต์ โดยใช้มอเตอร์ขับเคลื่อน 2 ตัว แยกขับล้อซ้าย-ขวา และ RC Servo Motor 1 ตัว ในการบังคับมุมเลี้ยว ใช้การรับส่งข้อมูลแบบไร้สาย 3 ชุด ในการควบคุมการเคลื่อนที่ ส่งข้อมูลลักษณะการเคลื่อนที่ และส่งข้อมูลภาพ การควบคุมการทำงานของหุ่นยนต์ใช้ไมโครคอนโทรลเลอร์ dsPIC30F2010 เป็นตัวประมวลผลในหุ่นยนต์ มีการทำงาน 2 ลักษณะ คือ การทำงานแบบอัตโนมัติ และการควบคุมด้วยตนเอง สำหรับแบบอัตโนมัติใช้ข้อมูลจาก Infrared Measuring Distance Sensor 3 ตัว เป็นข้อมูลในการควบคุมการเคลื่อนที่ของหุ่นยนต์ และส่งข้อมูลลักษณะการเคลื่อนที่ผ่าน RS232 ไร้สายความถี่ 2.4 GHz มายังคอมพิวเตอร์ เพื่อใช้สร้างแผนที่แบบใช้ข้อกำหนดตายตัว ในโหมดการควบคุมหุ่นยนต์ด้วยตนเอง ใช้ข้อมูลปริมาณเพียง 2 แกน ของวิทยุบังคับ คือ ขึ้นลงและซ้ายขวาในการควบคุมการเคลื่อนที่ทั้งหมด

ผลการทดสอบหุ่นยนต์สามารถขึ้นทางชันได้สูงสุด 45 องศา สำหรับผิวพื้นเอียงที่แปะกระดาษทราย ความเร็วสูงสุดในการเคลื่อนที่ 0.342 m/s การทำงานในโหมดอัตโนมัติเมื่อทดลองกับท่อแอร์ของระบบปรับอากาศจำลองที่สร้างขึ้น หุ่นยนต์สามารถเลือกเส้นทางการเคลื่อนที่ได้ตามหลักการที่ใช้ในการเขียนโปรแกรม และสามารถกลับมายังจุดเริ่มต้นได้ การควบคุมหุ่นยนต์ในโหมดควบคุมด้วยตนเองทำได้ง่ายไม่ซับซ้อน สามารถควบคุมหุ่นยนต์ให้เคลื่อนที่ในพื้นที่จำกัดได้ดี มีความคล่องตัวในการเคลื่อนที่สูง ระยะการรับส่งข้อมูลไกลที่สุดในพื้นที่เปิดโล่งประมาณ 35 m โดยอุปกรณ์ไร้สายทุกตัวทำงานปกติ การสร้างแผนที่แบบใช้ข้อกำหนดตายตัวสามารถบอกลักษณะเส้นทางการเคลื่อนที่ได้ใกล้เคียงกับเส้นทางการเคลื่อนที่จริงของหุ่นยนต์ แต่ไม่เป็นไปตามสัดส่วนที่แท้จริง เนื่องจากไม่ได้นำข้อมูลของระยะทาง เวลา หรือทิศทางการเคลื่อนที่มาประกอบในการสร้างแผนที่

_____ _____ ___ / ___ / ___

ลายมือชื่อนิสิต ลายมือชื่ออาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

Kanjana Chawbankor 2009: Design of Air-duct Surveying Robot. Master of Engineering (Mechanical Engineering), Major Field: Mechanical Engineering, Department of Mechanical Engineering. Thesis Advisor: Mr. Withit Chatlatanagulchai, Ph.D.
103 pages.

In this work, we design and build an air-duct surveying robot. We apply the belt-driven system to the car-like driven system. Two motors are used to drive the right and left wheels separately, and an RC servo motor is used for turning. There are three sets of wireless communication hardwares for controlling robot movement, transmitting the position data, and transmitting the video image. The robot is controlled by a microcontroller modeled dsPIC30F2010 as the robot's central processor. There are two control modes, which are the automatic mode and the human controlled mode. In the automatic mode, signals from three infrared distance measuring sensors are used in controlling the robot movement. The robot position data are transmitted to a computer via a wireless RS232 port using 2.4 GHz frequency signal to generate the map of the air duct system. In the human controlled mode, a radio control is used in the robot movement control by controlling the two axes, horizontally and vertically.

Experimental result shows that the robot can climb a slope of forty five degrees when a sand paper is used to cover the slope's surface. The maximum ground speed is 0.342 m/s. In the automatic mode, when the robot is placed in an air duct system model, the robot is able to follow its designed path according to the navigation algorithm and to come back to the origin. In the human controlled mode, the movement control is not complicated, and the robot can move quite well in limited space with good mobility. The maximum wireless transmission distance in an open area is thirty five meters. All wireless devices are operated normally. The generated map from the robot position signals is close to the actual robot's path. However, the map is in the wrong scale since distance, time, or direction data are excluded in the map generation.

_____ _____ ___ / ___ / ___

Student's signature Thesis Advisor's signature

กิตติกรรมประกาศ

ข้าพเจ้าขอขอบพระคุณ อาจารย์วิทิต ฉัตรรัตนกุลชัย ที่ได้ให้โอกาส ความรู้ ประสบการณ์ และคำแนะนำต่างๆ ในการทำงานวิจัยนี้ให้ประสบความสำเร็จไปด้วยอย่างดี

ขอขอบพระคุณ อาจารย์คุณยุต เอี่ยมสอาด ที่ได้ให้คำแนะนำ คำปรึกษาอย่างใกล้ชิดและร่วมตรวจสอบวิทยานิพนธ์นี้

ขอขอบพระคุณอาจารย์ และเจ้าหน้าที่ทุกท่าน ในภาควิชาวิศวกรรมเครื่องกล มหาวิทยาลัยเกษตรศาสตร์ ที่ให้วิชาความรู้ ตลอดจนช่วยเหลือในด้านต่างๆ ตลอดระยะเวลาที่ทำการศึกษา

สุดท้ายนี้ขอขอบพระคุณครอบครัวที่ให้โอกาสในการศึกษา ให้ความไว้วางใจ และให้กำลังใจเสมอมา ขอบคุณเพื่อน พี่ น้อง ทุกคนที่ให้ความช่วยเหลือ และคำแนะนำต่างๆ แก่ข้าพเจ้า

กาญจนะ ชาวบ้านเกาะ

กันยายน 2552

สารบัญ

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

ตารางผนวกที่

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

ภาพผนวกที่ **หน้า**

คำอธิบายสัญลักษณ์และอักษรย่อ

คำอธิบายสัญลักษณ์

F_{app}	=	แรงขับเคลื่อนของหุ่นยนต์ (N)
μ	=	สัมประสิทธิ์แรงเสียดทานระหว่างพื้นกับล้อหุ่นยนต์
m	=	มวลของหุ่นยนต์ (kg)
g	=	ความเร่ง (9.81 m/s^2)
α	=	มุมที่พื้นเอียง (องศา)
v	=	อัตราเร็ว (m/s)
P	=	กำลังของมอเตอร์ขับแต่ละตัว (W)
N	=	ค่าความปลอดภัย
N_m	=	จำนวนมอเตอร์ขับ
T	=	แรงบิดของมอเตอร์ (Nm)
V_m	=	อัตราเร็วการหมุนของมอเตอร์ (rpm)
F_{osc}	=	ความถี่สัญญาณนาฬิกา
F_{cy}	=	ความถี่การทำงานของไมโครคอนโทรเลอร์
T_{cy}	=	คาบการทำงานของไมโครคอนโทรเลอร์

คำอธิบายอักษรย่อ

Ton	=	ช่วงที่สัญญาณ Output เป็น +5V
Toff	=	ช่วงที่สัญญาณ Output เป็น 0V
ESC	=	Electronic Speed Control
XTAL	=	ความถี่ของตัวกำเนิดสัญญาณนาฬิกา
PLL	=	ตัวคูณสัญญาณนาฬิกา

การออกแบบหุ่นยนต์สำรวจท่อแอร์

Design of Air-duct Surveying Robot

คำนำ

การทำความสะอาดหรือการซ่อมแซมท่อลมของบบปรับอากาศขนาดใหญ่ จำเป็นต้องใช้เวลาและเงินทุนจำนวนมาก การทำความสะอาดแต่ละครั้งจึงต้องทำตามกำหนดเวลา หรือทำเมื่อสังเกตเห็นความผิดปกติในระบบปรับอากาศ การทำความสอาดท่อลมตามกำหนดเวลาอาจพบปัญหาได้ล่าช้า แต่การสังเกตก็อาจไม่ครอบคลุมทั่วถึงเช่นกัน

ดังนั้น จึงมีแนวความคิดที่จะสร้างหุ่นยนต์ที่ใช้ในการสำรวจท่อลมของระบบปรับอากาศขึ้น เพื่อเป็นตัวตรวจสอบความสะอาด ความปลอดภัย รวมไปถึงความเสียหายหรือ ความผิดปกติภายในท่อลมของระบบปรับอากาศ ก่อนที่จะดำเนินการทำความสะอาด ปรับปรุงแก้ไข หรือซ่อมแซมระบบปรับอากาศต่อไป ซึ่งการใช้หุ่นยนต์ดังกล่าวจะทำให้ทราบถึงชนิดของปัญหา ตำแหน่งที่เกิดปัญหา และสามารถเลือกวิธีในการแก้ปัญหาได้อย่างถูกต้องเหมาะสม รวดเร็ว และลดต้นทุนในการทำงาน รวมถึงสามารถหาวิธีป้องกันปัญหาที่อาจเกิดขึ้นในอนาคตได้อีกด้วย

การออกแบบโครงสร้างหุ่นยนต์จะใช้การประยุกต์ระหว่างระบบสายพานขับเคลื่อน เข้ากับระบบบังคับเลี้ยวแบบรถยนต์ เพื่อความคล่องตัวในการทำงานภายใต้พื้นที่จำกัด ใช้การส่งข้อมูลไร้สาย ในการควบคุมการทำงาน ส่งสัญญาณภาพ และส่งข้อมูลการทำงาน สร้างแผนที่แสดงเส้นทางการเคลื่อนที่ โดยใช้การสร้างแผนที่แบบมีข้อกำหนดตายตัว

วัตถุประสงค์

1. ออกแบบโครงสร้างหุ่นยนต์ให้เหมาะสมกับการสำรวจท่อลมของระบบปรับอากาศ

2. ออกแบบระบบควบคุมหุ่นยนต์ให้สามารถสำรวจท่อลมของระบบปรับอากาศได้เองโดยอัตโนมัติ

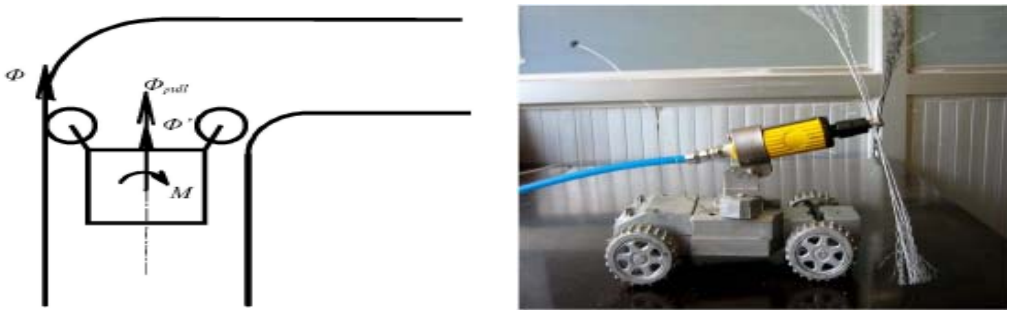
3. ประยุกต์ใช้การรับส่งข้อมูลแบบไร้สายในการควบคุมการทำงานของหุ่นยนต์

4. สร้างแผนที่แสดงเส้นทางการเคลื่อนที่ของหุ่นยนต์ โดยใช้ข้อมูลที่ได้รับจากตัวหุ่นยนต์

5. บันทึกข้อมูลภาพในขณะที่ทำการสำรวจ

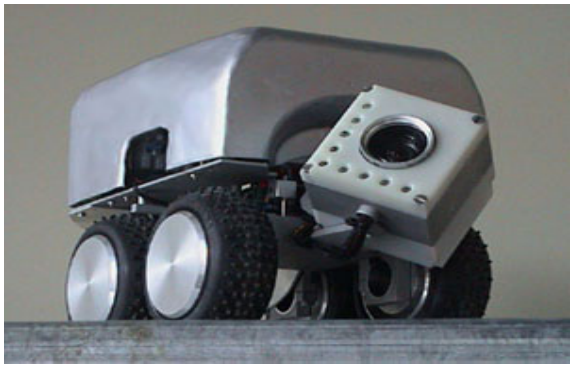
การตรวจเอกสาร

Wang and Zhang (2006) สร้างหุ่นยนต์ทำความสะอาดท่อลมของระบบปรับอากาศแบบอัตโนมัติ โดยใช้ AC Motor 1 ตัว ในการขับเคลื่อน หุ่นยนต์สามารถเคลื่อนที่ทำความสะอาดในท่อตรงและทางเลี้ยวได้เอง การควบคุมจากภายนอก จะควบคุมให้หุ่นยนต์เดินหน้าหรือถอยหลังเท่านั้น สำหรับหุ่นยนต์ต้นแบบนั้น การเคลื่อนที่ของหัวแปลงทำความสะอาดจะควบคุมด้วยมือ



ภาพที่ 1 Autonomous Air Duct Cleaning Robot (Prototype)

Autonomus System Lab of Eidgenössische Technische Hochschule Zürish (n.d.) สร้างหุ่นยนต์สำรวจท่อแอร์ โดยใช้ระบบขับเคลื่อน 4 ล้อ มีกล้อง 2 ตัว บริเวณหน้าและหลัง ใช้สายไฟในการส่งข้อมูลการควบคุมและข้อมูลภาพ โดยมีม้วนสายไฟภายในตัวหุ่นยนต์ ขณะเคลื่อนที่ไปข้างหน้าหุ่นยนต์จะปล่อยสายไฟไปตามพื้น และเมื่อเคลื่อนที่กลับหุ่นยนต์จะม้วนสายไฟกลับเข้าไปในตัวหุ่นยนต์ สายส่งข้อมูลมีความยาว 30 เมตร ควบคุมการทำงานด้วยเครื่องคอมพิวเตอร์แบบพกพา



ภาพที่ 2 Air-duct Inspection Robot

Triventek Inc. (n.d.) เป็นหุ่นยนต์สำรวจท่อแอร์ ของบริษัท TRIVENTEK ใช้ระบบขับเคลื่อน 4 ล้อ สามารถขึ้นทางชันได้ 40 องศา ตัวหุ่นยนต์หนัก 2 กิโลกรัม สามารถทำงานในท่อที่มีขนาดตั้งแต่ 160 mm ขึ้นไป ส่งข้อมูลการควบคุมและข้อมูลภาพผ่านสายไฟยาว 15 เมตร แสดงผลภาพผ่านจอ LCD ควบคุมการเคลื่อนที่ด้วย Joy-stick

ภาพที่ 3 Jetvent Inspector Robot

อุปกรณ์และวิธีการ

อุปกรณ์

การออกแบบหุ่นยนต์เพื่อให้สามารถทำงานตามต้องการได้อย่างมีประสิทธิภาพ จำเป็นต้องใช้ความรู้ความเข้าใจในหลายสาขา เช่น สาขาวิศวกรรมเครื่องกล วิศวกรรมไฟฟ้า วิทยาศาสตร์คอมพิวเตอร์ และความรู้ด้านปัญญาประดิษฐ์ (Artificial Intelligence A.I.) โดยความรู้ในแต่ละด้านจะนำไปสู่การออกแบบ ส่วนต่างๆของหุ่นยนต์

การออกแบบหุ่นยนต์ในหัวข้ออุปกรณ์ จะแบ่งการออกแบบออกเป็น 2 ส่วน คือ การออกแบบโครงสร้าง และการออกแบบระบบไฟฟ้า การออกแบบทั้ง 2 ส่วนเป็นการนำเสนอหลักการที่ใช้ในการออกแบบหุ่นยนต์ ซึ่งนำไปสู่การเลือกใช้วัสดุอุปกรณ์ที่สอดคล้องกับข้อกำหนดในการออกแบบต่อไป

1. การออกแบบโครงสร้าง

หุ่นยนต์ถูกแบ่งออกเป็น 2 ประเภทตามลักษณะการใช้งาน คือ

1. หุ่นยนต์ชนิดที่ติดตั้งอยู่กับที่ (Fixed Robot) เป็นหุ่นยนต์ที่ไม่สามารถเคลื่อนที่ได้ด้วยตัวเอง มีลักษณะเป็นแขนกล สามารถขยับและเคลื่อนไหวได้เฉพาะแต่ละข้อต่อ ภายในตัวเองเท่านั้น มักนำไปใช้ในโรงงานอุตสาหกรรม เช่น โรงงานประกอบรถยนต์

2. หุ่นยนต์ชนิดที่เคลื่อนที่ได้ (Mobile Robot) หุ่นยนต์ประเภทนี้จะแตกต่างจากหุ่นยนต์ที่ติดตั้งอยู่กับที่ เพราะสามารเคลื่อนที่ได้ด้วยตัวเอง โดยการใช้ล้อหรือการใช้ขา ซึ่งหุ่นยนต์ประเภทนี้ปัจจุบันยังเป็นงานวิจัยที่ทำการศึกษาอยู่ภายในห้องทดลอง เพื่อพัฒนาออกมาใช้งานในรูปแบบต่าง ๆ เช่น หุ่นยนต์สำรวจดาวอังคาร ขององค์การ NASA

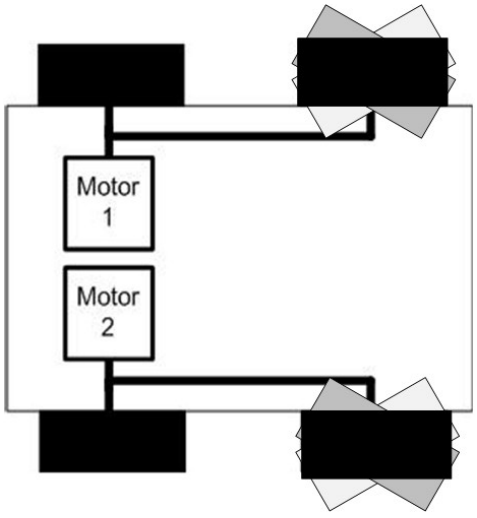
หุ่นยนต์ชนิดที่เคลื่อนที่ได้มีหลายแบบ แต่ละแบบเหมาะสมกับงานที่แตกต่างกัน ดังนั้นการเลือกชนิดของระบบขับเคลื่อนจึงเป็นสิ่งสำคัญในการออกแบบ สามารถแบ่งระบบขับเคลื่อนของหุ่นยนต์ตามลักษณะการขับเคลื่อนออกได้เป็น 3 แบบ คือ

1. ระบบที่ใช้ล้อขับเคลื่อน
2. ระบบสายพานขับเคลื่อน
3. ระบบที่ใช้ขาในการขับเคลื่อน

ซึ่งมีลักษณะดังตัวอย่างของระบบขับเคลื่อนในภาคผนวก ก

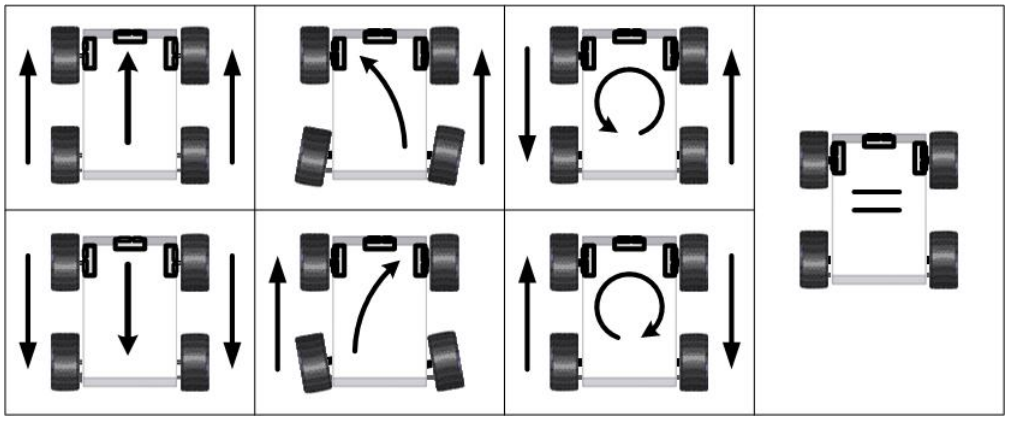
ในการสำรวจท่อลมของระบบปรับอากาศ จำเป็นต้องใช้ระบบขับเคลื่อนที่มีความคล่องตัวสูง สามารถเลี้ยวในที่แคบได้ดี แรงในการขับเคลื่อนดี ซ่อมบำรุงได้ง่าย จึงเลือกใช้ระบบสายพานขับเคลื่อน (Tracked drive) ประยุกต์เข้ากับระบบขับเคลื่อนแบบรถยนต์ (Ackermann Steering)

ระบบสายพานขับเคลื่อนมีความคล่องตัวในการเคลื่อนที่และ การเลี้ยวดี แต่ใช้พลังงานในการเลี้ยวสูง เนื่องจากผลของแรงเสียดทานที่สายพานขับเคลื่อน ดังนั้น จึงใช้ล้อแทนสายพานเพื่อลดแรงเสียดทานลง และเพิ่มระบบบังคับเลี้ยวแบบรถยนต์ เพื่อช่วยลดแรงเสียดทานในการเลี้ยวมุมกว้างลงอีก จึงได้ระบบขับเคลื่อนที่เหมาะสมกับการใช้งาน ในการสำรวจท่อลมของระบบปรับอากาศ ดังภาพที่ 4



ภาพที่ 4 ระบบขับเคลื่อนของหุ่นยนต์สำรวจท่อแอร์

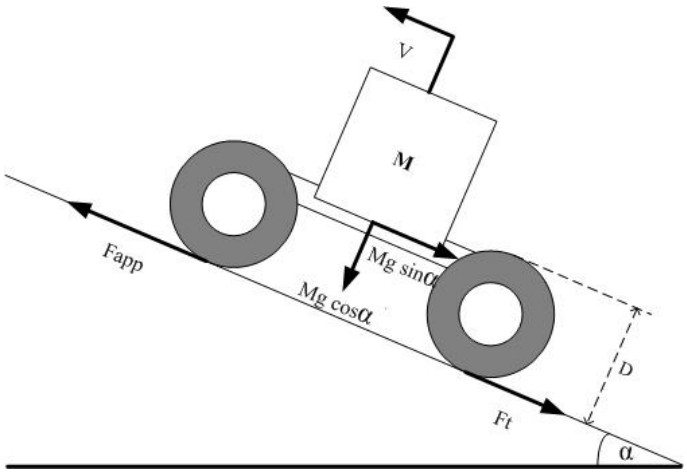
จากภาพที่ 4 ระบบขับเคลื่อนที่ได้จากการประยุกต์ ทำให้ได้ลักษณะการเคลื่อนที่หลายรูปแบบดังแสดงในภาพที่ 5



ภาพที่ 5 การเคลื่อนที่ในรูปแบบต่างๆ ของหุ่นยนต์

มอเตอร์ขับ

สำหรับหุ่นยนต์ที่มีมีลักษณะการเคลื่อนที่คล้ายรถยนต์ การคำนวณจะใช้หลักการเดียวกับการคำนวณแรงขับเคลื่อนของรถยนต์ (Jones *et al*., 1999) ดังภาพที่ 6 และสมการที่ 1, 2, 3 และ 4



ภาพที่ 6 Free Body Diagram of Robot

$$F_{app} = \mu mg\cos\alpha + mg\sin\alpha \tag{1}$$

จาก $P = Fv$ และ $T = Fr$ จะได้

$$P = \frac{F_{app} vN}{N_m} \tag{2}$$

$$T = \frac{F_{app} N r}{N_m} \quad (3)$$

$$V_m = \frac{30 v}{\pi r} \quad (4)$$

โดยที่ F_{app} คือ แรงขับเคลื่อนของหุ่นยนต์ (N)

μ คือ สัมประสิทธิ์แรงเสียดทานระหว่างพื้นกับล้อหุ่นยนต์

m คือ มวลของหุ่นยนต์ (kg)

g คือ ความเร่ง (9.81 m/s^2)

α คือ มุมที่พื้นเอียง (องศา)

v คือ อัตราเร็ว (m/s)

P คือ กำลังของมอเตอร์ขับแต่ละตัว (W)

N คือ ค่าความปลอดภัย

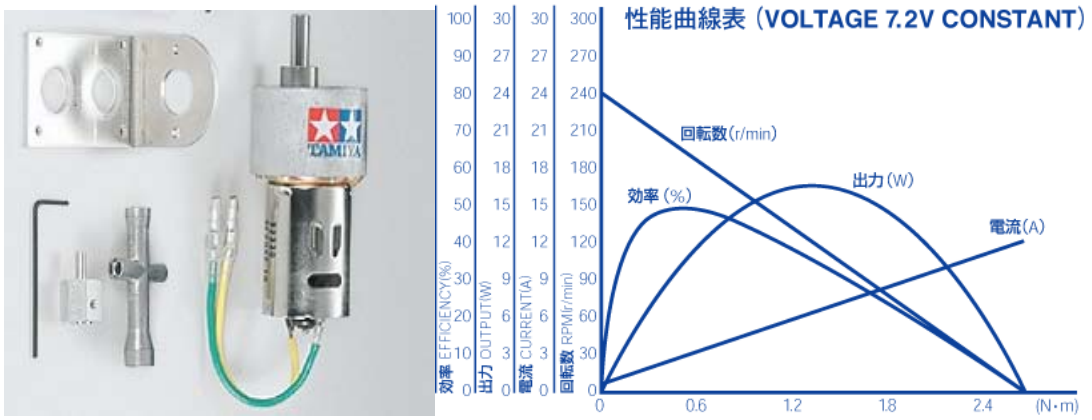
N_m คือ จำนวนมอเตอร์ขับ

T คือ แรงบิดของมอเตอร์ (Nm)

V_m คือ อัตราเร็วการหมุนของมอเตอร์ (rpm)

จากสมการที่ 1-4 เมื่อทำการคำนวณหาขนาดของมอเตอร์ขับ โดยกำหนดค่าของตัวแปลต่างๆ ดังนี้ μ = 0.3, m = 5 kg, g = 9.81 m/s^2, α = 30 องศา, v = 0.15 m/s และ N = 3 จะได้ F_{app} = 37.27 N, P = 4.19 W, T = 6.71 Nm และ V_m = 23.87 rpm

ระบบขับเคลื่อนที่ทำการออกแบบใช้มอเตอร์ 2 ตัวในการขับเคลื่อน ดังนั้น แต่ละตัวจะต้องมีขนาด P = 2.1 W, T = 3.35 Nm และ V_m = 23.87 rpm จึงเลือกใช้มอเตอร์ TAMIYA ดังภาพที่ 7 ร่วมกับชุดเกียร์ ที่มีอัตราทด 34/11

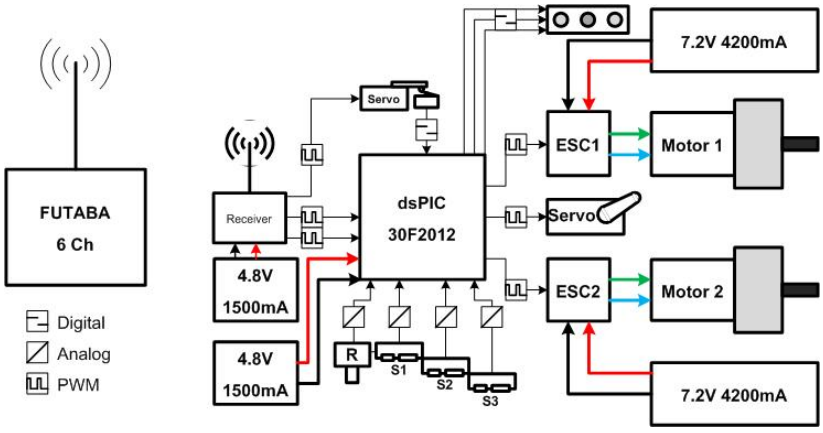


ภาพที่ 7 มอเตอร์พร้อมชุดเกียร์ของ TAMIYA

จากภาพที่ 7 จุดที่ Power สูงสุด คือ 16.5 W มีค่า Efficiency 55%, Current 16.5 A, Speed 165 rpm และ T 1.3 Nm เมื่อรวมอัตราทดของห้องเกียร์ 34/11 จะได้ T = 4.02Nm และ Speed 53.4 rpm ซึ่งมีค่ามากกว่าค่าที่คำนวณจากการเลือกขนาดมอเตอร์ขับ

2. การออกแบบระบบไฟฟ้า

การออกแบบระบบไฟฟ้า คือ การเลือกอุปกรณ์ไฟฟ้า ที่ใช้ในหุ่นยนต์ หลังจากเลือกระบบขับเคลื่อนของหุ่นยนต์ สามารถออกแบบวงจรควบคุมการทำงานของหุ่นยนต์ได้ดังภาพที่ 8



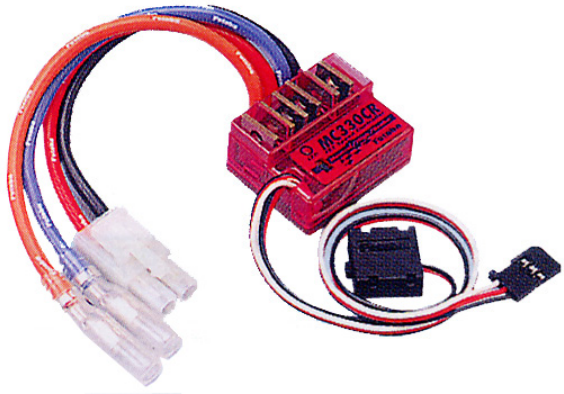
ภาพที่ 8 วงจรควบคุมการทำงานของหุ่นยนต์

จากการออกแบบวงจรควบคุมการทำงานของหุ่นยนต์ อุปกรณ์ที่ใช้ในหุ่นยนต์ ประกอบด้วย

1. Driver Motor
2. Distance Sensor
3. วิทยุบังคับ
4. Controller Board
5. อุปกรณ์ส่งข้อมูลไร้สาย
6. กล้อง Video
7. แบตเตอร์รี่

Driver Motor

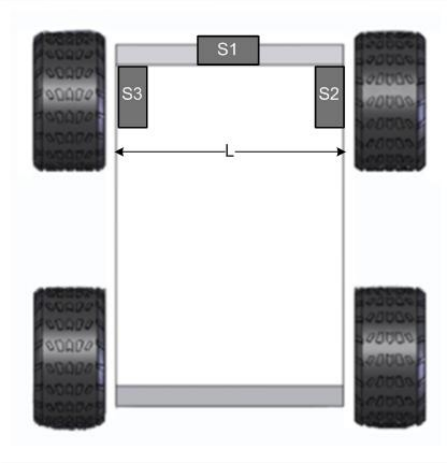
ระบบขับเคลื่อนของหุ่นยนต์ที่ทำการออกแบบใช้มอเตอร์ขับจำนวน 2 ตัว ดังภาพที่ 4 จากคุณสมบัติของมอเตอร์ สามารถใช้กระแสสูงสุดที่ 14-15A ที่ 7.2V ดังนั้น จำเป็นต้องใช้ Driver Motor ที่สามารถขับมอเตอร์ที่กระแสมากกว่า 15 A ได้ ดังนั้น จึงเลือกใช้ ESC (Electronic Speed Control) FUTABA รุ่น MC330CR จำนวน 2 ตัว ทำงานที่แรงดัน 7.2-8.4 V ขับกระแสเดินหน้าสูงสุด 200 A ถอยหลังสูงสุด 100 A ที่ความถี่ 1.5 kHz ดังภาพที่ 9



ภาพที่ 9 FUTABA ESC รุ่น MC330CR

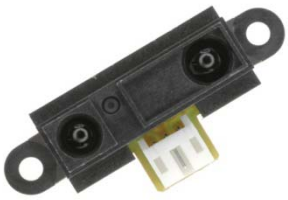
Distance sensor

การเลือก Sensor วัดระยะทาง (Distance sensor) จะต้องคำนึงถึงขนาดของท่อแอร์ในระบบปรับอากาศ รุปแบบการใช้งาน และลักษณะของสัญญาณที่ออกมาเป็นสำคัญ ท่อแอร์ในระบบปรับอากาศมีขนาดความกว้างประมาณ 50-120 cm และจากตำแหน่งการติดตั้ง Sensor ดังภาพที่ 10



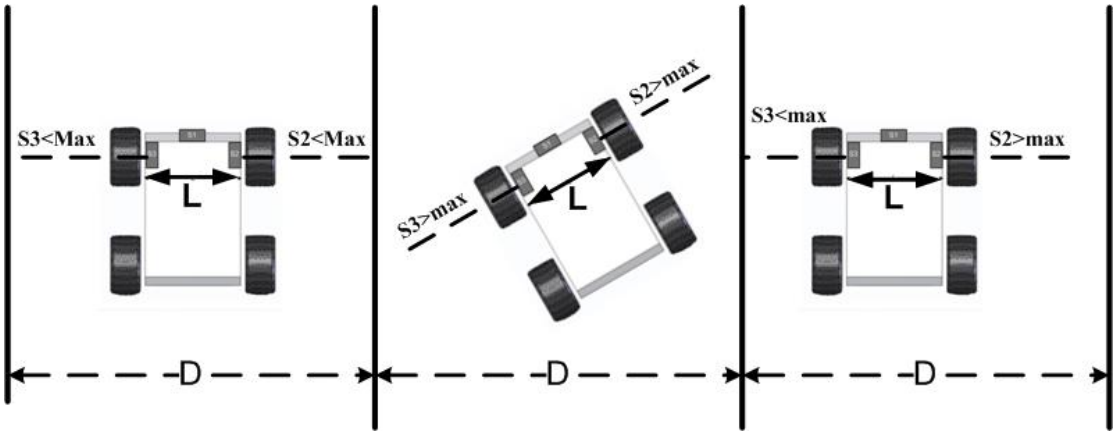
ภาพที่ 10 ตำแหน่งการติดตั้ง Distance Sensor

จากภาพที่ 10 ระยะการทำงานของSensor ทั้ง 2 ด้าน รวมกับ L ควรมีค่ามากกว่าขนาดของท่อที่ต้องการทำงาน จึงเลือกใช้ GP2Y0A21YK0F Distance Sensor ของ SHARP วัดระยะได้ 10-80 cm ให้ค่าออกมาเป็นแรงดัน (V) ที่แปลผันตามระยะทาง ซึ่งเหมาะสมที่จะใช้งานกับไมโครคอนโทรลเลอร์ เนื่องจากสามารถแปลงสัญญาณ Analog เป็น Digital (ADC) ได้ ซึ่งเป็นคุณสมบัติโดยทั่วไปของไมโครคอนโทรลเลอร์ Sensorมีลักษณะดังภาพที่ 11



ภาพที่ 11 Distance Sensor GP2Y0A21YK0F

ขนาดความกว้างของท่อแอร์ที่สามารถทำงานได้ คือ 80+L+80 เมื่อ L = 21 cm ความกว้างท่อเป็น 181 cm ขนาดของท่อที่เล็กที่สุดเป็น 10+L+10=41 cm ดังนั้น หุ่นยนต์ควรทำงานได้ในท่อขนาด 41-181 cm แต่เนื่องจากขณะที่หุ่นยนต์เอียงทำมุมกับผนังท่อ หรือหุ่นยนต์ชิดผนังท่อฝั่งใดฝั่งหนึ่งมากเกินไป ทำให้ระยะเพิ่มมากขึ้น จนอาจออกนอกช่วงการทำงานของSensor จึงไม่ควรกำหนดให้ขนาดของท่อในการทำงานให้พอดีกับระยะการทำงานของ Sensor ดังภาพที่ 12 จึงควรใช้ค่าโดยประมาณ 50-150 cm การพิจารณาข้างต้นใช้เฉพาะการทำงานในโหมดอัตโนมัติเท่านั้น สำหรับในโหมดควบคุมด้วยตนเอง หุ่นยนต์สามารถทำงานได้ในท่อขนาด 40 cm ขึ้นไป โดยพิจารณาจากขนาดของตัวหุ่นยนต์



ภาพที่ 12 แสดงขนาดของท่อกับระยะทางสูงสุดของ Sensor ในตำแหน่งต่างๆ

วิทยุบังคับ

วิทยุบังคับที่ใช้จะต้องมีจำนวนช่องสัญญาณที่เพียงพอกับการใช้งาน หุ่นยนต์ที่ทำการออกแบบ ต้องการใช้การส่งข้อมูล 3 ช่องสัญญาณ คือ ส่งข้อมูลควบคุมโหมดการทำงาน ส่งข้อมูลเดินหน้าถอยหลัง และส่งข้อมูลเลี้ยวซ้ายเลี้ยวขวา จึงเลือกใช้วิทยุบังคับของ FUTABA รุ่น 6EXAP ซึ่งมี 6 ช่องสัญญาณ พร้อม RC Servo Motor 2 ตัว ดังภาพที่ 13

ภาพที่ 13 วิทยุบังคับ FUTABA รุ่น 6EXAP

Controller Board

การเลือก Controller Board ต้องมีจำนวนช่องสัญญาณ Input และ Output เพียงพอกับความต้องการใช้งาน และต้องมีฟังก์ชันการทำงานเฉพาะที่สอดคล้องกับการควบคุมอุปกรณ์ต่างๆ ของหุ่นยนต์ จากการออกแบบการทำงานของหุ่นยนต์สามารถกำหนดจำนวนช่องสัญญาณต่างๆ ได้ดังตารางที่ 1

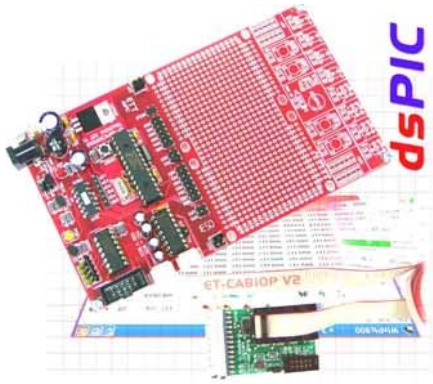
ตารางที่ 1 ชนิดและจำนวนสัญญาณ Input และ Output ของหุ่นยนต์

ลำดับที่	ชื่อสัญญาณ	ชนิดของสัญญาณ	จำนวน
1	Mode Input	Digital Input	1
2	สัญญาณ เดินหน้า-ถอยหลัง	PWM Input	1
3	สัญญาณ เลี้ยวซ้าย-ขาว	PWM Input	1
4	สัญญาณ ควบคุม ESC1และ ESC2	PWM Output	2
5	สัญญาณ ควบคุม บังคับเลี้ยว	PWM Output	1
6	สัญญาณ distance sensor 3 ตัว	Analog Input	3
7	สัญญาณ ควบคุมความเร็วในโหมดอัตโนมัติ	Analog Input	1
8	สัญญาณควบคุมไฟแสดงสถานะของการทำงาน	Digital Output	3
รวม			13

จากข้อมูลในตารางที่ 1 สามารถสรุปจำนวณช่องสัญญาณต่างได้ดังนี้ PWM Input 2 Ch, PWM Output 3 Ch, Analog Input 4 Ch, Digital Input 1 Ch, Digital Output 3 Ch จึงเลือกใช้ Controller Board ของ ETT รุ่น ET-dsPIC30F2010 V1 ซึ่งใช้ MCU เบอร์ dsPIC30F2010 ขนาด 28 ขา ของบริษัท Microchip ดังภาพที่ 14 โดยมีคุณสมบัติ ดังนี้

หน่วยความจำโปรแกรมแบบ FLASH 12KBYTE (4KWORD)

RAM ขนาด 512 BYTE	EEPROM ขนาด 1KBYTE
TIMER 16 BIT 3 CH	INPUT CAP 4 CH (PWM Input)
A TO D ขนาด 10 BIT 6 CH	MOTOR CONTROL PWM 6 CH



ภาพที่ 14 Controller Board ET-dsPIC30F2010 V1

จากคุณสมบัติของ ET-dsPIC30F2010 V1 นั้น จะใช้การเขียนข้อมูลผ่านพอร์ตขนานของคอมพิวเตอร์ เพื่อความสะดวกในการทำงานกับคอมพิวเตอร์แบบพกพา ซึ่งไม่มีพอร์ตขนาน จึงต้องใช้เครื่องโปรแกรมผ่านพอร์ต USB ของ ETT รุ่น ET-PGM PIC USB ดังภาพที่ 15

ภาพที่ 15 ET-PGM PIC USB V1, V1 Plus

อุปกรณ์ส่งข้อมูลไร้สาย

อุปกรณ์ส่งข้อมูลไร้สาย จะทำหน้าที่ส่งข้อมูลระยะทาง สภาวะการทำงาน หรือข้อมูลต่างๆ ที่ใช้ในการตรวจสอบการทำงานของหุ่นยนต์ สำหรับไมโครคอนโทรลเลอร์นั้น จะมีช่องการส่งข้อมูล ผ่านพอร์ตอนุกรม (RS232) ของคอมพิวเตอร์อยู่แล้ว ดังนั้น จึงเลือกใช้ ET-RF24G V1 ดังภาพที่ 16 จำนวน 2 ตัว (สำหรับส่ง 1 ตัว และสำหรับรับ 1 ตัว) ซึ่งเป็นอุปกรณ์ส่งข้อมูลผ่าน RS232 แบบไร้สายที่ความถี่ 2.4 GHz มีการเข้ารหัสเพื่อแยกสัญญาณที่ไม่เกี่ยวข้องออก ดังนั้น จึงสามารถใช้พร้อมกันได้หลายชุด โดยการตั้งรหัสที่แตกต่างกัน

ภาพที่ 16 อุปกรณ์ส่งข้อมูลผ่าน RS232 ไร้สาย ET-RF24G V1

กล้อง Video

การส่งข้อมูลภาพ จะใช้กล้อง Video วงจรปิดไร้สายขนาดเล็ก เพื่อความสะดวกในการติดตั้ง และมีน้ำหนักเบา โดยตัวกล้องเป็นทั้งกล้องและตัวส่งสัญญาณภาพแบบไร้สาย และมีตัวรับสัญญาณภาพเพื่อแปลงสัญญาณ และแสดงภาพผ่านระบบ AV ดังภาพที่ 17

ภาพที่ 17 กล้อง Video ไร้สายขนาดเล็ก

สัญญาณ AV ที่รับจากกล่องรับสัญญาณของกล้อง สามารถแสดงภาพได้โดยใช้ TV ทั่วไป แต่เนื่องจากในการทำงานจะใช้เครื่องคอมพิวเตอร์ในการสื่อสารกับ Controller Board และจำเป็นต้องบันทึกข้อมูลภาพในการสำรวจ ดังนั้น จึงต้องใช้อุปกรณ์รับสัญญาณ AV ให้กับคอมพิวเตอร์ คือ UTV330+ USB 2.0 TV BOX ของ DTECH ซึ่งสามารถรับสัญญาณ TV AV และยังสามารถบันทึกข้อมูลภาพของสัญญาณที่รับเข้ามาได้ ทั้งในแบบภาพนิ่งและภาพเคลื่อนไหว ดังภาพที่ 18

ภาพที่ 18 DTECH UTV330

แบตเตอร์รี่

การเลือกแบตเตอร์รี่ พิจราณาจากขนาดแรงดันไฟฟ้า จำนวนแหล่งจ่ายไฟ น้ำหนักโดยรวมของหุ่นยนต์ และระยะเวลาในการทำงานที่ต้องการ ดังนั้น จึงจำเป็นต้องทราบข้อมูลอุปกรณ์ไฟฟ้าทั้งหมดที่ใช้ในหุ่นยนต์เพื่อเป็นข้อมูลในการเลือกแบตเตอร์รี่ ซึ่งมีข้อมูลดังตารางที่ 2

ตารางที่ 2 คุณสมบัติของอุปกรณ์ไฟฟ้าในหุ่นยนต์

ลำดับที่	รายละเอียด	แรงดัน (V)	กระแส
1	มอเตอร์ขับ 2 ตัว	7.2	0-15 A/ตัว
2	Controller Board	5	~25mA
3	RC servo Motor S3003	4.8	7.2mA
4	Distance Sensor 3 ตัว	5	33mA/ตัว
5	Receiver	4.8	9.5mA
6	กล้อง Video	9	-
7	Micro Servo Motor	4.8	~10mA
8	ไฟส่องสว่าง (LED) 8 ดวง	5	20mA/ดวง

เนื่องจากมอเตอร์ขับจะใช้กระแสไฟฟ้าสูง ในขณะที่เปลี่ยนแปลงลักษณะการเคลื่อนที่ทำให้เกิดการกระชากกระแสไฟ ดังนั้น ถ้าใช้แหล่งจ่ายไฟเดียวกันกับ Controller Board หรือ Receive จะทำให้เกิดการ Reset หรือเกิดการกระตุกของ Servo Motorได้ ซึ่งจะทำให้การทำงานคลาดเคลื่อน หรืออาจเกิดความเสียหายกับ RC Servo Motor ได้ จึงต้องแยกแหล่งจ่ายไฟออกเป็นส่วนๆ ดังนี้

1. แหล่งจ่ายไฟ 4.8V 2500 mA 1ชุด ใช้กับ Controller Board, Distance sensor 3 ตัว และ RC Servo Motor S3003 กระแสรวม คือ 25+33X3+7.2 = 131.2mA ดังนั้น จะใช้ได้ประมาณ 2500/131.2 = 19.05 ชั่วโมง

2. แหล่งจ่ายไฟ 4.8V 1200 mA 1ชุด ใช้กับ Receiver 1 ตัว, Micro Servo Motor 1 ตัว และไฟส่องสว่าง 8 ดวง กระแสรวม คือ 9.5+10+20X8 = 139.5 mA ดังนั้นจะใช้ได้ประมาณ 1200/139.5 = 8.6 ชั่วโมง

3. แหล่งจ่ายไฟ 7.2V 4200 mA 2 ชุด ใช้กับมอเตอร์ขับ 2 ตัว กระแสที่ได้จากการวัดในขณะเคลื่อนที่ปกติประมาณ 1-1.8A (วัดจาก Power Supply ที่อ่านค่ากระแสไฟฟ้าได้) จึงใช้ค่าคำนวณที่ 2 A ดังนั้นจะใช้ได้ประมาณ 4200/2000 = 2.1 ชั่วโมง

4. แหล่งจ่ายไฟ 9 V จำนวน 1 ชุด สำหรับ Video กล้องไร้สาย ใช้งานได้ประมาณ 1 ชั่วโมง (จากการทดลอง) แต่ถ้าใช้แบตเตอร์รี่ที่มีความจุมากกว่านี้ก็จะใช้ได้นานกว่า 1 ชั่วโมง

3. รายการอุปกรณ์ที่ใช้ในการออกแบบ

จากการออกแบบโครงสร้างและ ระบบไฟฟ้าของหุ่นยนต์สำรวจท่อแอร์ สามารถเขียนเป็นรายการของอุปกรณ์ที่จำเป็นต้องใช้ พร้อมทั้งรายละเอียดของราคาอุปกรณ์ต่างๆ ได้ดังตารางที่ 3

ตารางที่ 3 รายการอุปกรณ์ที่ใช้ในขั้นตอนการออกแบบ

ลำดับที่	อุปกรณ์	ราคา	จำนวน	รวม
1	Tamiya Motor 72102	3,800	1	3,800
2	ชุดเกียร์ 34/11	750	2	1,500
3	ESC FUTABA รุ่น MC330CR	1,900	2	3,800
4	GP2Y0A21YK0F Distance Sensor	590	3	1,770
5	วิทยุบังคับ FUTABA รุ่น 6EXAP	4,500	1	4,500
6	ETT รุ่น ET-dsPIC30F2010 V1	890	1	890
7	ET-PGM PIC USB V1	790	1	790
8	ET-RF24G V1	1,390	2	2,780
9	กล้อง Video	1,100	1	1,100
10	UTV330+ USB 2.0 TV BOX	1,200	1	1,200

ตารางที่ 3 (ต่อ)

ลำดับที่	อุปกรณ์	ราคา	จำนวน	รวม
11	4.8V 2500 mA	320	1	320
12	4.8V 1200 mA	280	1	280
13	7.2V 4200 mA	1,500	2	3,000
	รวม			25,730

วิธีการ

หลังจากเลือกอุปกรณ์ต่างๆ ในหัวข้ออุปกรณ์ จะทำการดำเนินงานด้านการออกแบบโครงสร้างของหุ่นยนต์อย่างละเอียด สร้างหุ่นยนต์ต้นแบบ เขียนโปรแกรมควบคุมการทำงาน เขียนโปรแกรมสร้างแผนที่ และทดสอบการทำงานต่อไป โดยจะอธิบายเป็นหัวข้อ ดังนี้

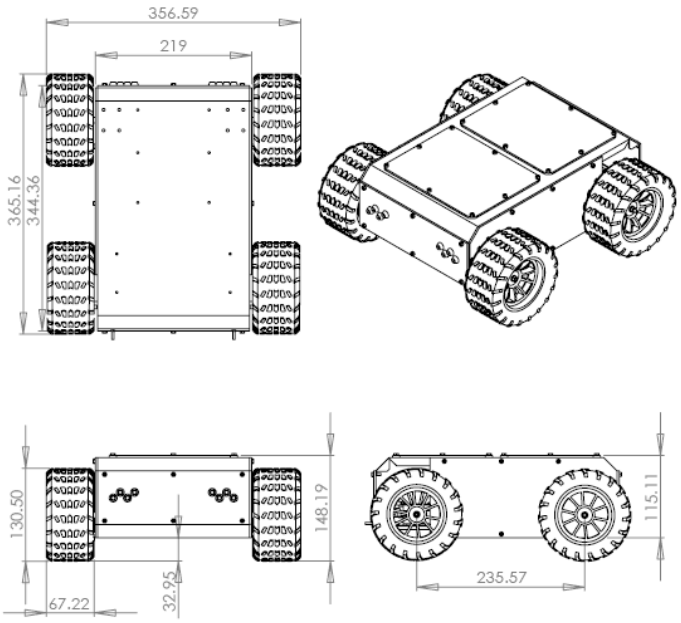
1. ออกแบบและสร้างหุ่นยนต์ต้นแบบ
2. เขียนโปรแกรมควบคุมการทำงานของหุ่นยนต์
3. เขียนโปรแกรมสร้างแผนที่
4. ออกแบบการทดลอง

1. การออกแบบและสร้างหุ่นยนต์ต้นแบบ

การออกแบบหุ่นยนต์ จะเน้นให้สามารถซ่อมแซมได้ง่ายไม่ยุ่งยาก ดังนั้น อุปกรณ์ที่ใช้ในหุ่นยนต์ส่วนใหญ่จะเป็นอุปกรณ์ที่หาซื้อได้ตามท้องตลาดทั่วไป โดยส่วนใหญ่แล้วเป็นอุปกรณ์ของรถบังคับวิทยุ เพื่อให้ง่ายแก่การซ่อมแซม และลดต้นทุนในการสร้างชิ้นส่วนต่างๆ

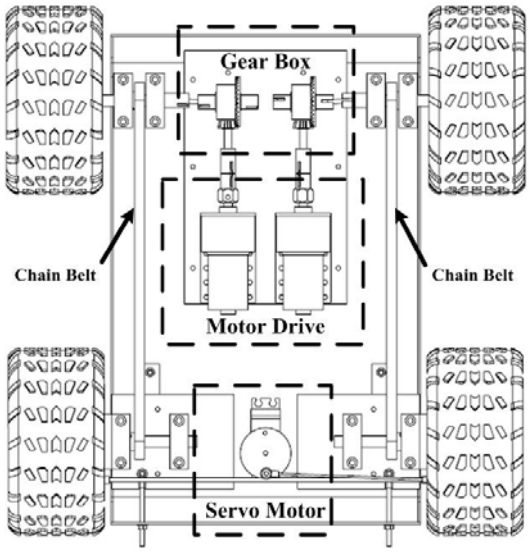
ข้อกำหนดที่ใช้ในการออกแบบ คือ สามารถขึ้นทางชันได้ 30 องศา ความเร็วในการเคลื่อนที่ 0.15 m/s ซึ่งข้อกำหนดนี้ส่วนหนึ่งได้ถูกออกแบบไปแล้วในหัวข้อการเลือกมอเตอร์ขับ ดังนั้น จึงต้องออกแบบความสูงของหุ่นยนต์ ขนาดของล้อและระยะห่างของล้อเพื่อให้สามารถขึ้นทางชัน 30 องศา ได้โดยไม่ติดขัด รวมไปถึงการจัดวางระบบส่งกำลังของระบบขับเคลื่อนแบบผสมของระบบสายพาน กับระบบขับเคลื่อนของรถยนต์ จากข้อกำหนดในการออกแบบ และอุปกรณ์ต่างๆ

ที่เลือกใช้ในหัวข้ออุปกรณ์ จึงออกแบบหุ่นยนต์ได้ดังภาพที่ 19 โดยมีลักษณะการส่งกำลังของระบบขับเคลื่อนดังภาพที่ 20



ภาพที่ 19 ลักษณะภายนอกของหุ่นยนต์ที่ทำการออกแบบ

โครงสร้างของหุ่นยนต์ใช้แผ่นอลูมิเนียมหนา 2 mm พับขึ้นรูป การใช้อลูมิเนียมแผ่น 2 mm เป็นการลดนำหนัก และการพับขึ้น รูปจะช่วยให้โครงสร้างแข็งแรงมากขึ้น ส่วนประกอบบางส่วนจะใช้แผ่นพลาสติกใสหนา 3 mm เพื่อให้สามารถมองเห็นภายในได้ ขนาดของหุ่นยนต์ กว้าง 357 mm ยาว 365 mm สูง 148 mm ความสูงจากพื้น 33 mm ระยะห่างระหว่างล้อหน้าและล้อหลัง 235 mm จากความสูงนี้หุ่นยนต์สามารถขึ้นทางชันได้ 30 องศา ได้โดยไม่ติดขัด ความสูงของหุ่นยนต์ดังกล่าวนั้นเป็นความสูงขั้นต่ำที่สุดที่เป็นไปได้ เนื่องจากต้องการให้ได้ปริมาตรในหุ่นยนต์มากที่สุดเท่าที่จะเป็นไปได้



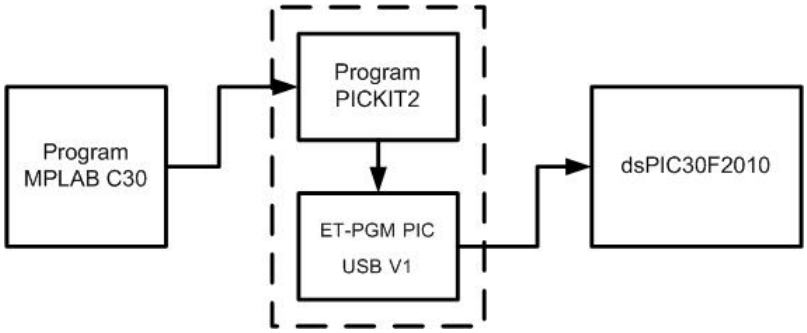
ภาพที่ 20 ระบบส่งกำลังและกลไกการบังคับเลี้ยวของหุ่นยนต์

ระบบขับเคลื่อนของหุ่นยนต์ จะใช้มอเตอร์ขับ 2 ตัวส่งกำลังผ่านชุดเกียร์ที่มีอัตราทด 34/11 แยกกันขับล้อด้านซ้าย และด้านขาว ใช้โซ่ส่งกำลังจากล้อหน้าไปเพลาหลังของแต่ละด้าน ที่ล้อหลังจะมีระบบบังคับเลี้ยวโดยใช้ Servo Motor 1 ตัว บริเวณล้อหลังจึงใช้ข้อต่ออ่อนส่งกำลังจากเพลาหลังไปยังล้อหลังแต่ละด้าน เพื่อให้สามารถเปลี่ยนมุมการส่งกำลังได้ จากการออกแบบทำการสร้างหุ่นยนต์ต้นแบบได้ดังภาพที่ 21

ภาพที่ 21 หุ่นยนต์ต้นแบบ

2. การเขียนโปรแกรมควบคุมการทำงานของหุ่นยนต์

ในการเขียนโปรแกรม จะใช้โปรแกรม MPLAB C30 เขียน Code ภาษา C (นคร และชัยวัฒน์, ม.ป.ป.) และ PICKIT 2 กับ ET-PGM PIC USB V1 ในการเขียนข้อมูลลงในหน่วยความจำของ dsPIC30F2010 ผ่าน ICD2 ดังภาพที่ 22



ภาพที่ 22 ขั้นตอนการเขียนโปรแกรม

จากภาพที่ 22 แสดงขั้นตอนการเขียนโปรแกรมไมโครคอนโทรลเลอร์ โดยเริ่มจากการเขียน Code ภาษา C ในโปรแกรม MPLAB C30 ทำการ Compile เพื่อตรวจสอบความถูกต้องของ Code เมื่อไม่มีข้อผิดพลาด โปรแกรม MPLAB C30 จะสร้างไฟล์มีนามสกุล hex ขึ้นมา เพื่อเป็นข้อมูลในการเขียน Code ลงไปในหน่วยความจำของ Chip ใช้โปรแกรม PICKIT 2 อ่านข้อมูลนามสกุล hex และทำการเขียนข้อมูลดังกล่าว ลงในหน่วยความจำของ Chip เมื่อเขียนข้อมูลเสร็จสมบูรณ์ สามารถทดสอบการทำงานได้ทันที

การเขียนโปรแกรมเพื่อใช้ควบคุมนั้น จำเป็นต้องมีการรับข้อมูลจาก Sensor ต่างๆ เพื่อนำไปประมวลผล และทำการส่งสัญญาณควบคุมไปยังอุปกรณ์ที่ต้องการควบคุม จึงจำเป็นต้องมีความรู้ความเข้าใจเกี่ยวกับอุปกรณ์ ทั้งด้าน Input และ Output เพื่อให้ทราบถึงหลักการทำงาน และลักษณะของสัญญาณที่ใช้สั่งงานอุปกรณ์ หรือลักษณะของสัญญาณที่อ่านได้จาก Sensor เพื่อให้สามารถเขียน Code ได้อย่างถูกต้อง จึงแบ่งการอธิบายออกเป็น 2 ส่วน คือ

1. หลักการทำงานและการเขียน Code ของอุปกรณ์ต่างๆ ในหุ่นยนต์สำรวจท่อแอร์
2. ปัญญาประดิษฐ์ของหุ่นยนต์สำรวจท่อแอร์

2.1 หลักการทำงานและการเขียน Code ของอุปกรณ์ต่างๆ ในหุ่นยนต์สำรวจท่อแอร์

สำหรับหุ่นยนต์สำรวจท่อแอร์นั้นจะแบ่งอุปกรณ์ออกเป็น 2 ส่วน คือ Input และ Output ซึ่งมีรายละเอียดดังนี้

Input

- Receiver
- Distance Sensor GP2Y0A21YK0F
- Potentiometer
- Limit switch

Output

- ESC
- RC SERVO MOTOR
- LED

การอธิบายถึงลักษณะการทำงาน และการเขียนโปรแกรมควบคุมอุปกรณ์ต่างๆ นั้น เพื่อให้เข้าใจได้ง่าย จะอธิบายตามลักษณะของสัญญาณที่คล้ายคลึงกันไว้ด้วยกัน โดยจะแบ่งออกเป็นกลุ่มตามลักษณะสัญญาณ ดังนี้

- Digital Input, Digital Output คือ Limit switch และ LED
- Analog Input คือ Potentiometer และ Distance Sensor
- PWM Input, PWM Output คือ Receiver, RC Servo Motor และ ESC

2.1.1 Digital IO

Digital IO เป็นการเขียนโปรแกรมพื้นฐาน โดยมีการทำงาน 2 ลักษณะ คือ เปิด-ปิด ใช่-ไม่ใช่ ถูก-ผิด 1-0 สัญญาณ Digital จะมีข้อมูลเพียงเท่านี้ เช่น อุปกรณ์จำพวก หลอดไฟ Switch ปิด-เปิด เป็นต้น อย่างไรก็ตามในการทำงานที่มีความซับซ้อนเช่น การสร้าง PWM การส่งข้อมูลผ่าน RS232 การแสดงผล 7 Segment หรือ จอ LCD ล้วนใช้พื้นฐานของ Digital IO ทั้งสิ้น เพียงแต่เป็นการทำงานของ Digital IO ในรูปแบบที่ใช้ฐานเวลาเข้ามาควบคุม ทำให้เกิดเป็นรูปแบบการส่งข้อมูลในแบบต่างๆ ขึ้น

การเขียนCode Digital IO จะมี 2 ส่วน คือ การกำหนดคุณสมบัติการทำงาน และการใช้งาน การกำหนดคุณสมบัติการทำงาน ทำได้โดยการเขียนคำสั่งดังต่อไปนี้

- TRISFbits.TRISF2 = 0; // Configuration for RF2 to output
- TRISFbits.TRISF3 = 1; // Configuration for RF3 to input

จากตัวอย่าง เป็นการกำหนดคุณสมบัติการทำงานของขา RF2 และ RF3 โดยกำหนดให้ ขาRF2 ทำงานเป็น Output Digital และขา RF3 ทำงานเป็น Input Digital

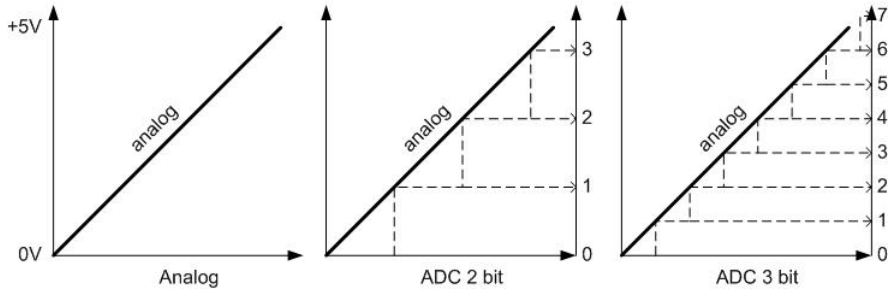
การใช้งาน Input Digital จะเป็นการตรวจสอบว่าที่ขา Input มีแรงดันเป็น +5V หรือ 0V ถ้าเป็น +5V สถานะของขาถือเป็นจริง แต่ถ้าเป็น 0V สถานะของขาถือเป็นเท็จ การเขียน Code Input Digital จึงเป็นการเขียนเพื่อตรวจสอบสถานะของขา Input ว่าเป็นจริงหรือเท็จ

การใช้งาน Output Digital จะเป็นการกำหนดให้ขาที่ทำงานเป็น Output Digital มีสถานะเป็น +5V หรือ 0V ดังตัวอย่างที่ 1 ในภาคผนวก ข

จากตัวอย่างที่ 1 ในภาคผนวก ข เป็นการกำหนดให้ RF2 เป็น Output Digital เทียบได้กับ LED และ RF3 เป็น Input digital เทียบได้กับ Limit Switch โปรแกรมจะทำการตรวจสอบว่าถ้า Limit Switch ถูกกด (0V) ให้ขา RF2 มีสถานะเป็น +5V LED ติด แต่ถ้า Limit Switch ไม่ถูกกด (+5V) ให้ขา RF2 มีสถานะเป็น 0V LED ดับ การตรวจสอบว่า Limit Switch ถูกกดจะใช้ if (!PORTEbits.RF3) และไม่ถูกกดจะใช้ if (PORTEbits.RF3) สั่ง Output ด้วย PORTEbits.RF2 = 1; และ PORTEbits.RF2 = 0; (! ในภาษา C หมายถึง not)

2.1.2 Analog Input

Analog เป็นค่าที่บอกปริมาณมากน้อย โดยอาจมีค่าแรงดันตั่งแต่ 0V จนถึง +5V การอ่านค่า Analog จะใช้โมดูล ADC (Analog to Digital Converter) ในการอ่านค่าแรงดัน ความละเอียดในการอ่านค่า ADC จะระบุด้วยจำนวน bit ใน dsPIC30F2010 มี ADC ขนาด 10 bit จำนวน 6 ช่อง



ภาพที่ 23 ลักษณะของสัญญาณ Analog และหลักการทำงานของ ADC

จากภาพที่ 23 จะเห็นได้ว่า ADC คือ การแบ่งสัญญาณ Analog ออกเป็นช่วงๆ และระบุค่าออกมาเป็นตัวเลขเพื่อใช้บอกปริมาณ โดยยิ่งถ้าช่วงมากก็จะมีความละเอียดมากขึ้น ซึ่งจำนวนช่วงนั้นจะขึ้นกับค่า bit เช่น 2 bit มี $2^2 = 4$ ช่วง 3 bit มี $2^3 = 8$ ช่วง สำหรับ dsPIC30F2010 โมดูล ADC ขนาด 10 bit มี 2^{10}=1024 ช่วง แต่ละช่วงห่างกัน 4.8828125 mV (ที่ Vref เป็น 0V และ +5V)

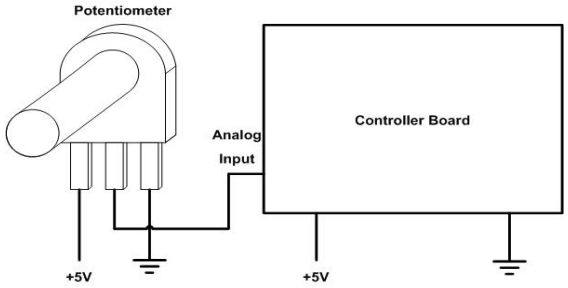
การเขียน Code เริ่มจากการกำหนดคุณสมบัติการทำงานในโมดูล ADC และเขียน Code อ่านค่า ADC ใน While Loop เพื่อให้ทำงานแบบต่อเนื่อง ดังตัวอย่างที่ 2 ในภาคผนวก ข

จากตัวอย่างที่ 2 ในภาคผนวก ข. init_adc(); คือ ฟังก์ชันที่ใช้ในการกำหนดคุณสมบัติการทำงานของโมดูล ADC จากตัวอย่างดังกล่าวกำหนดให้ AN0-AN3 ทำงานเป็น Analog Input ให้ Vref เป็น Vdd และ Vss ซึ่งเป็นแรงดันเดียวกับที่จ่ายให้กับ Controller Board โดยที่ Vdd = +5V และVss = 0V ใน Main Program จะเป็นการวนอ่านค่า ADC ของ AN0-AN3 ซ้ำวนไปเรื่อยๆ และเก็บค่าไว้ใน Array ชื่อ adc_buff[]

Vref คือ ค่าแรงดันที่ใช้ในการเปรียบเทียบ ถ้ากำหนด Vref เป็น 0V และ +5V ที่ความละเอียด 10 bit และอ่านค่า ADC ได้ 512 สามารถคำนวณค่าแรงดัน Analog Input ได้เป็น 512X(5/1024)=2.5V แต่ถ้ากำหนด Vref เป็น 0V และ +12V แรงดัน Analog Input จะเป็น 512X(12/1024)=6V

Potentiometer

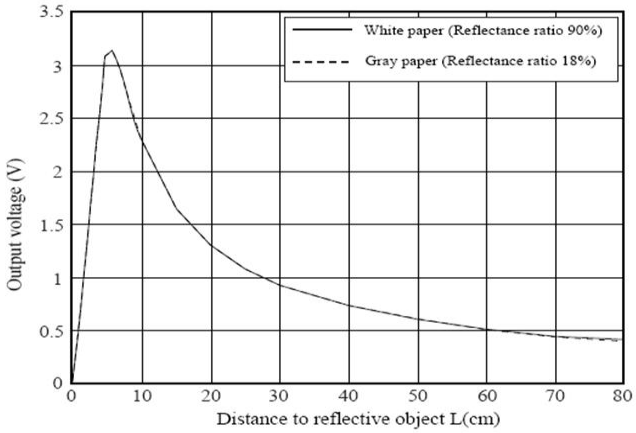
การอ่านค่าจาก Potentiometer นั้น สามารถใช้การอ่านค่าดังตัวอย่างข้างต้นได้ เนื่องจาก Potentiometer คือ ตัวต้านทานปรับค่าได้ ดังนั้น จึงสามารถต่อวงจรเปลี่ยนความต้านทานให้เป็นสัญญาณ Analog ซึ่งเป็นการเปลี่ยนจากการหมุนเป็นความต้านทาน และจากความต้านทานเป็นแรงดันที่ใช้เป็น Input ให้กับ Controller Board ดังภาพที่ 24



ภาพที่ 24 การเปลี่ยนค่าความต้านทานเป็นสัญญาณ Analog Input

Distance Sensor (GP2Y0A21YK0F)

เนื่องจากสัญญาณที่ออกมาจาก GP2Y0A21YK0F เป็นแรงดัน (Analog) จึงใช้โมดูล ADC ในการอ่านค่า แต่จากการอ่านค่า ADC เป็นการอ่านค่าของแรงดัน ไม่ใช่ระยะทางที่ Sensor อ่านได้ จึงต้องสร้างฟังก์ชั่นในการคำนวณระยะ จากค่า ADC 10 bit ให้เป็นระยะทาง

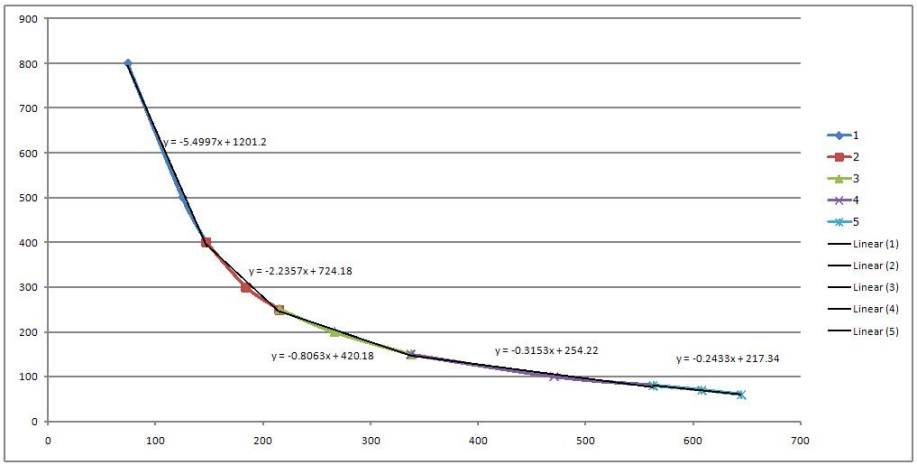


ภาพที่ 25 ความสัมพันธ์ของ Output Voltage (V) กับ Distance (cm)

จากภาพที่ 25 แสดงความสัมพันธ์ระหว่างแรงดันเป็น V กับระยะเป็น cm เพื่อความสะดวกในการใช้งาน จะทำการเขียนกราฟใหม่ โดยเปลี่ยนหน่วยจาก V เป็นค่า ADC 10 bit และจาก cm เป็น mm

ADC 10 bit คือ การเปลี่ยนค่าแรงดันจาก 0-5V (Vref เป็น 0V และ +5V) ให้มีค่าเป็น 10 bit หรือเท่ากับ 0- 2^{10} หรือ 0-1023 ดังนั้น จะได้ความสัมพันธ์ดังสมการที่ 5 ส่วนการเปลี่ยนจาก cm เป็น mm ทำได้โดยเอา 10 คูณ Cm จะได้ mm นำค่า ADC และ mm ไปสร้างกราฟ จะได้กราฟดังภาพที่ 26

$$ADC = V \times \frac{1024}{5} \tag{5}$$



ภาพที่ 26 ความสัมพันธ์ระหว่างค่า ADC กับระยะทางเป็น mm

จากภาพที่ 26 เป็นการพอร์ตกราฟโดยในแนวแกน X เป็นระยะทาง (mm) และในแนวแกน Y เป็นค่า ADC ขนาด 10 bit เนื่องจากไมโครคอนโทรลเลอร์ไม่สามารถคำนวณสมการที่มีความซับซ้อนมากได้ จึงได้ทำการแบ่งกราฟออกเป็นช่วงสั้นๆ และสร้างเป็นสมการเส้นตรงออกมาเป็น 5 ช่วง ดังตารางที่ 4

ตารางที่ 4 ช่วงของค่า ADC และสมการของแต่ละช่วง

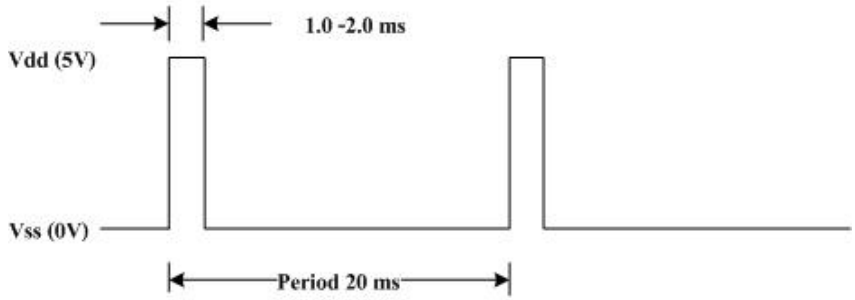
ช่วงที่	ค่า ADC	สมการ
1	74-147	1201.2-(5.5 X ADC)
2	147-215	724.18-(2.23 X ADC)
3	215-338	420.18-(0.8 X ADC)
4	338-563	254.2-(0.315 X ADC)
5	563-640	217.34-(0.24 X ADC)

จากข้อมูลในตารางที่ 4 สามารถเขียนฟังก์ชันคำนวณ ค่าระยะได้ดังตัวอย่างที่ 3 ในภาคผนวก ข

ตัวอย่างที่ 3 ในภาคผนวก ข z คือ ADC ดังนั้น ถ้า z มีค่าน้อยกว่า 74 จะให้ค่าระยะ เป็น 1000 ซึ่งถือว่าระยะเป็นระยะอนันต์ (นอกรัศมีทำงานของ Sensor) และถ้า z มากกว่า 640 จะให้ค่าระยะเป็น 0 คือ ใกล้กว่าระยะที่ Sensor จะอ่านค่าได้ สามารถเรียกใช้งานได้ดังนี้ Dis1 = distance(adc_buff[0]); โดยค่า Dis1 เป็นตัวแปรที่ใช้เก็บค่าระยะของ Sensor 1 และ adc_buff[0] เป็นค่า ADC ของ Analog Input ช่อง 0

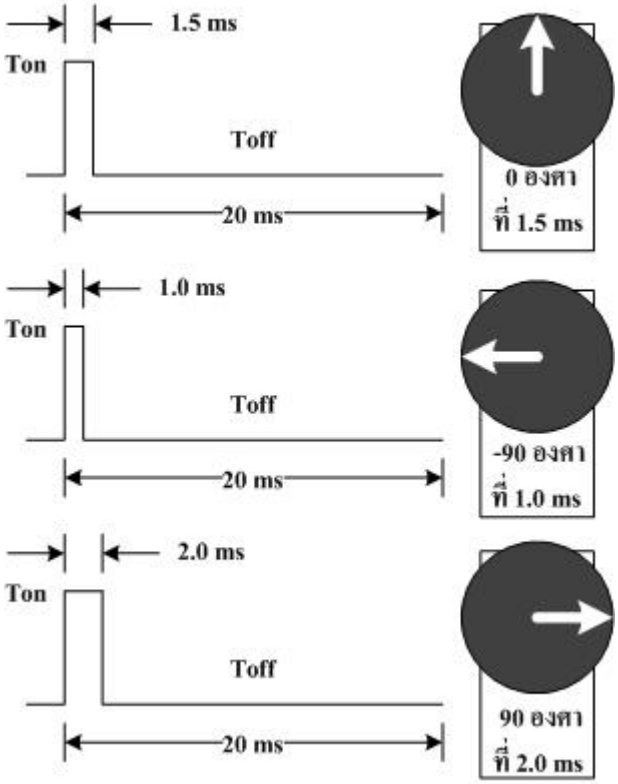
2.1.3 PWM Input และ PWM Output

Receiver เป็นอุปกรณ์สำหรับ รับข้อมูลตำแหน่งของวิทยุบังคับ FUTABA รุ่น 6EXAP และส่งสัญญาณควบคุมไปยัง RC Servo Motor เพื่อให้ RC Servo Motor เคลื่อนที่ไปยังตำแหน่งที่ต้องการ สำหรับหุ่นยนต์สำรวจท่อแอร์นั้น ใช้ Receiver เป็นสัญญาณ Input ให้กับไมโครคอนโทรลเลอร์ เพื่อใช้ควบคุมการเคลื่อนที่ของหุ่นยนต์ โดยมีลักษณะสัญญาณเป็น PWM ที่มี Period คงที่ที่ 20 ms และช่วง Ton จะมีค่าอยู่ระหว่าง 1.0-2.0 ms ดังภาพที่ 27



ภาพที่ 27 ลักษณะของสัญญาณควบคุม Servo Motor

RC Servo Motor เป็นมอเตอร์ที่หมุนได้ 180 องศา โดยที่ตำแหน่งของ Servo Motor จะถูกกำหนดด้วย ช่วง Ton คือ มีค่าตังแต่ 1.0-2.0 ms โดยมีลักษณะการเคลื่อนที่และ ลักษณะของสัญญาณสั่งงานดังภาพที่ 28



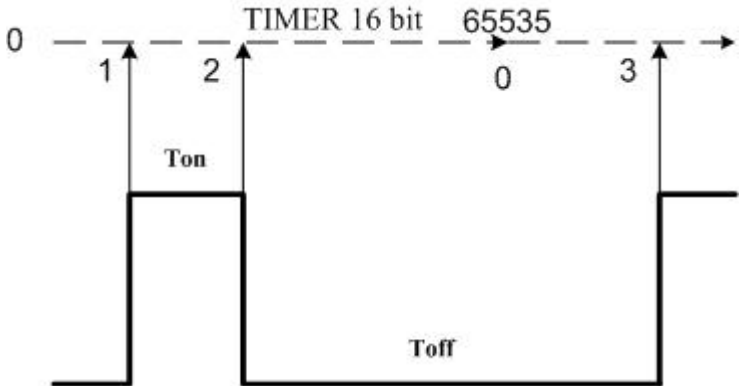
ภาพที่ 28 แสดงตำแหน่งของ Servo Motor ที่ช่วง Ton ต่างๆ

จากภาพที่ 27 และ 28 การใช้ Receiver เป็น Input ดังนั้น จึงใช้การทำงานในโมดูลตรวจจับสัญญาณนาฬิกา เพื่ออ่านค่าความแตกต่างในช่วง Ton ที่มีการเปลี่ยนแปลงตั้งแต่ 1.0-2.0 ms และต้องใช้การทำงานในแบบตรวจจับสัญญาณทั้งขอบขาขึ้นและขาลงของสัญญาณ Input

PWM Input

หลักการทำงานของโมดูลตรวจจับสัญญาณนาฬิกาจะทำการกำหนดฐานเวลาที่ทำงานควบคู่ไปกับการตรวจจับสัญญาณ เมื่อตรวจจับสัญญาณได้จะทำการอ่านค่าเวลาเพื่อนำไปคำนวณช่วงเวลา Ton ต่อไป

ฐานเวลาที่ใช้มีขนาด 16 บิต ซึ่งจะนับตั้งแต่ 0 ถึง 65535 และเมื่อ ถึง 65535 จะกลับไปเริ่มที่ 0 ใหม่วนไปเรื่อยๆ เมื่อมีการตรวจจับสัญญาณนาฬิกาได้จะทำการอ่านค่าเวลาในปัจจุบันเก็บไว้ เมื่อได้ค่าเวลาจากการตรวจจับสัญญาณนาฬิกาครบ 3 จุด จะสามารถหาค่าความแตกต่างของสัญญาณนาฬิกา โดยมีทั้งช่วง Ton และ Toff ความแตกต่างของทั้งสองช่วง คือ Ton<Toff ดังนั้น เราสามารถหาได้ทั้ง 2 ค่า ดังตัวอย่างต่อไปนี้



ภาพที่ 29 หลักการทำงานของโมดูลตรวจจับสัญญาณนาฬิกา

จากภาพที่ 29 กำหนดให้มีการตรวจจับสัญญาณ 3 ครั้ง ค่าที่อ่านได้เป็น ครั้งที่ 1 = 15000 ครั้งที่ 2 = 30000 และ ครั้งที่ 3 = 10000 ทำให้สามารถหาช่วงเวลาได้ 2 ช่วง คือ ช่วงเวลาระหว่างค่าที่ 1 กับ ค่าที่ 2 และ ระหว่างค่าที่ 2 กับค่าที่ 3 เนื่องจากค่าของครั้งที่ 2 มากกว่าครั้งที่ 1 ดังนั้น ช่วงแรกจึงเป็น 30000-15000 = 15000 แต่สำหรับช่วงที่ 2 จะเห็นได้ว่าค่าของครั้งที่ 3 น้อยกว่าครั้งที่ 2 แสดงว่ามีการนับเวลาจนครบ แล้วเริ่มใหม่ ดังนั้น การคำนวณในช่วงที่ 2 จึงต้องพิจารณา

การเริ่มต้นนับใหม่ของ Timer โดยช่วงที่ 2 จึงเป็น (65535-30000)+(10000+1) = 45536 เมื่อนำค่าของทั้ง 2 ช่วงมาเปรียบเทียบกัน จะเห็นว่า 15000 < 45536 ดังนั้น Ton = 15000 และ Toff = 45536 จากหลักการดังกล่าวนี้สามารถเขียนCode ภาษา C ได้ดังตัวอย่างที่ 4 ในภาคผนวก ข

จากตัวอย่างที่ 4 ในภาคผนวก ข จะใช้ Timer3 ทำงานในโหมด 16 bit ควบคู่ไปกับการตรวจจับสัญญาณ เมื่อมีการตรวจจับสัญญาณ 3 ครั้ง จะเก็บค่าไว้ในตัวแปร Capture_Value2_1, Capture_Value2_2 และ Capture_Value2_3 และเมื่อ Capture_Interrupt_Count = 0 จะนำค่าของตัวแปรทั้ง 3 ไปหาช่วงเวลา 2 ช่วง และเปรียบเทียบกัน ท้ายที่สุดจะได้ Ton = Period2_1 และ Toff = Period2_2 เพื่อใช้เป็นค่าในการเขียนโปรแกรมต่อไป

PWM Output

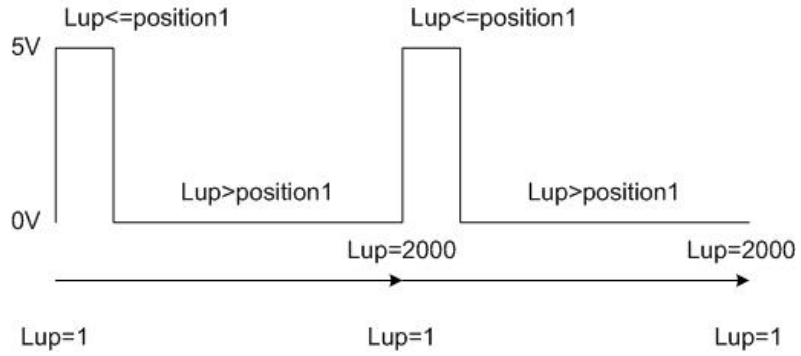
ก. RC Servo Motor

สัญญาณของ Receiver เป็นสัญญาณที่ใช้ควบคุมการทำงานของ RC Servo motor ดังนั้น ถ้าต้องการควบคุมการทำงานของ RC Servo Motor จึงต้องเขียน Code สร้างสัญญาณดังภาพที่ 28 การเขียน Code สร้าง PWM โดยทั่วไปจะมีโมดูลในการสร้าง PWM อยู่แล้ว แต่สามารถสร้างได้จำนวนจำกัด สำหรับ dsPIC30F2010 นั้น สามารถสร้างได้ 3 ช่อง แต่ละช่องจะมีคู่การทำงานเป็น H และ L ซึ่งจะใช้เมื่อต้องการขับ Motor 3 เฟส ดังนั้น เพื่อความยืดหยุ่นในการทำงานในกรณีที่ต้องการเพิ่ม Servo Motor จึงใช้วิธีการสร้างฟังก์ชันขึ้นมาเอง เนื่องจากสามารถสร้าง PWM ได้ตามจำนวนที่ต้องการ โดยการสร้าง PWM จะใช้ Interrupt Timer ช่วยสร้าง ซึ่งการทำงานจะคล้ายกับการสร้างด้วยโมดูลของไมโครคอนโทลเลอร์เอง

หลักการทำงานของ Interrupt Timer คือ ให้ Timer นับจากศูนย์จนถึงค่าที่กำหนดจะเกิด Interrupt ขึ้น เนื่องจาก Interrupt เป็นการทำงานที่มีความสำคัญมากกว่า Main Program ดังนั้น ทุกครั้งที่เกิด Interrupt ขึ้น ไม่ว่าโปรแกรมจะทำงานอยู่จุดใดก็ตาม จะเข้ามาทำงานในฟังก์ชัน Interrupt ก่อนเสมอ ดังนั้น การใช้ฟังก์ชัน Interrupt Timer สร้าง PWM นั้น จะได้ฐานเวลาที่คงที่ เพียงแต่ต้องใช้ Oscilloscope ในการตรวจสอบความถูกต้องของสัญญาณก่อนที่จะนำไปใช้งานจริง

หลักการเขียนโปรแกรมสร้าง PWM โดยใช้ Interrupt Timer จะใช้ Interrupt Timer 1 สร้างฐานเวลา และเมื่อเกิด Interrupt โปรแกรมจะเข้าไปทำงานในฟังก์ชั่น Interrupt โดยเพิ่มค่าเวลาที่เป็นฐานเวลาในการสร้าง PWM อีกที ซึ่งอาจเปรียบเทียบได้กับนาฬิกาในหน่วยวินาที กับนาที โดยที่การเกิด Interrupt แต่ละครั้งจะเป็นวินาที และตัวแปรเวลาในฟังก์ชันจะเป็นนาที ในขณะเดียวกันก็จะทำการตรวจสอบฐานเวลาในฟังก์ชั่นเพื่อ Output เป็น +5V หรือ 0V ตามเวลาที่ต้องการ โดยเมื่อได้คาบเวลาตามต้องการแล้ว ก็ทำการเปลี่ยนค่าให้เป็นค่าเริ่มต้นใหม่ ดังตัวอย่างที่ 5 ในภาคผนวก ข

จากตัวอย่างจะใช้ Interrupt Timer1 ซึ่งเป็น Timer ขนาด 16bit เมื่อเกิด Interrupt 1 ครั้ง ในฟังก์ชัน Interrupt ค่า lup จะมีค่าเพิ่มขึ้นทีละ 1 (lup = lup+1;) และมีการตรวจสอบค่า lup กับ position1 ถ้า lup<=position1 จะ Output +5V ออกไปที่ขา RF2 แต่ถ้า lup>position1 จะ Output 0V และเมื่อ lup มีค่าเท่ากับ 2000 ก็จะทำการ Reset ค่า lup กลับไปเป็น 1 ซึ่งค่า lup นี้เองที่เป็นตัวกำหนดคาบหรือความถี่ของ PWM dki ทำซ้ำเช่นนี้จะได้สัญญาณที่คงที่ ดังภาพที่ 30



ภาพที่ 30 หลักการทำงานของฟังก์ชันสร้าง PWM

จากภาพที่ 30 ถ้าต้องการเปลี่ยนแปลงช่วง Ton ซึ่งเป็นค่าที่ใช้ควบคุมตำแหน่ง Servo ให้ทำการเปลี่ยนค่าของ position1 จะทำให้ Ton เปลี่ยนแปลง ในฟังก์ชันนี้ สามารถเพิ่มช่อง Output เข้าไปได้อีก โดยการกำหนดตัวแปร position เพิ่ม และตรวจสอบค่ากับ lup และทำการ Output ออกไปยังขาที่ต้องการใช้งาน

หลักการทำงานของ Interrupt Timer ในการเขียนCode จะใช้ Interrupt Timer1 และให้ Timer1 ทำงานในโหมด 16 bit ซึ่งจะนับค่าตังแต่ 0-65535 และจะนับขึ้นทุกๆ Cycle การทำงาน (ถ้าไม่มีการลดทอนค่าใดๆ) โดยที่เวลาของ Cycle การทำงานหาได้จากหาได้จากสมการที่ 9

$$F_{\mathrm{OSC}} = XTAL \times PLL \tag{6}$$

$$F_{cy} = F_{OSC}/4 \tag{7}$$

$$T = 1/F \tag{8}$$

$$T_{cy} = 4/F_{OSC} \tag{9}$$

ทำให้สามารถกำหนดให้เกิด Interrupt ในเวลาที่ต้องการได้ โดยกำหนดค่าของตัวเลขที่ Timer1 จะนับไปถึง และเกิด Interrupt ซึ่งในตัวอย่างคือค่า match_value โดยมีค่าอยู่ระหว่าง 0-65535 ซึ่งสามารถคำนวณได้จากสมการ 10

$$match\_value = T \times T_{cy} \tag{10}$$

เพื่อให้สอดคล้องการใช้งานจริงที่ต้องการให้เกิด Interrupt 2000 ครั้งใน 20ms ดังนั้น การเกิด Interrupt 1 ครั้งต้องใช้เวลา 20 ms/2000 = 10 us โดยที่ XTAL เป็น 7.3728MHz PLL เป็น 8 (ค่าของ XTAL และ PLL เป็นค่าที่ใช้จริงในการเขียนโปรแกรม) ดังนั้น จึงคำนวณค่า match_value ได้เป็น ((7372800X8)/4)X10X10^{-6} = 147.5 (ในโปรแกรมจริงใช้ค่า 146.0) ค่า mach value ที่ได้จะนำไปใช้ในการกำหนดค่าการทำงานของ Timer 1 เพื่อให้เกิด Interrupt ในเวลาที่ต้องการ

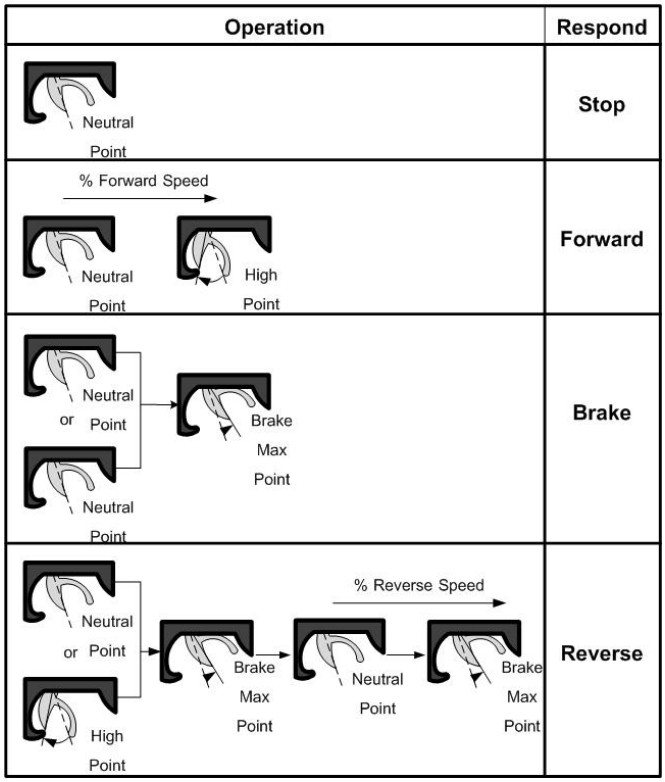
ข. ESC Futaba MC330CR

MC330CR เป็นอุปกรณ์ควบคุมความเร็วของมอเตอร์กระแสตรง (DC Motor) สำหรับรถบังคับวิทยุ โดยจะรับสัญญาณสั่งงาน เหมือนกับ RC Servo Motor เพื่อใช้ควบคุมความเร็วของมอเตอร์

เนื่องจากของเล่นประเภทรถบังคับวิทยุนั้น มีการเคลื่อนที่หลายรูปแบบ (ยกเว้นเลี้ยวซ้าย-ขวา เนื่องจากไม่ได้ใช้ ESC ควบคุม) ได้แก่ เดินหน้า ถอยหลัง และเบรก แต่ในการใช้งานจริงนั้นจะเน้นที่การเดินหน้าและ การเบรกเป็นสำคัญ ดังนั้น ESC ที่สามารถถอยหลังจะมีพิกัดกระแส

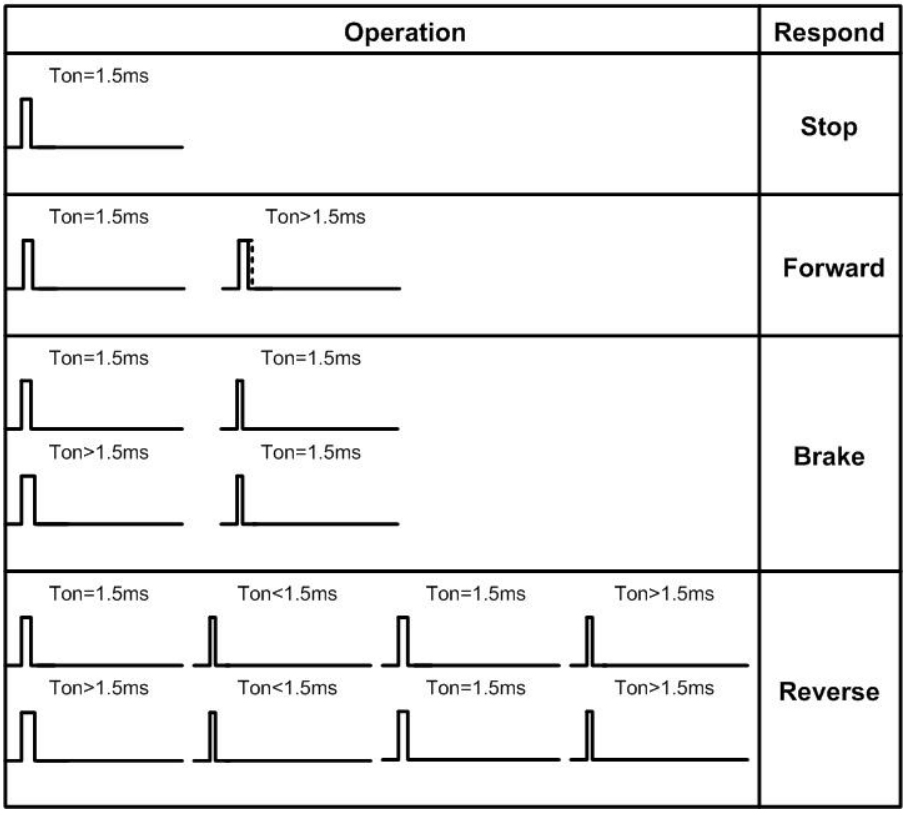
ในการถอยหลังต่ำกว่าการเดินหน้า สำหรับ ESC ระดับสูงบางรุ่นอาจมีแต่ฟังก์ชั่นการเดินหน้ากับการเบรกเท่านั้น สำหรับ MC330CR นั้นมีโหมดเดินหน้า เบรก และถอยหลัง แต่สามารถขับกระแสในการเดินหน้าได้มากกว่ากระแสในการถอยหลัง คือ กระแสในการเดินหน้า/กระแสในการถอยหลัง 200A/100A การควบคุมการเคลื่อนที่ทั้ง 3 รูปแบบ ใช้สายสัญญาณเพียงเส้นเดียว

ESC MC330CR มีลักษณะการทำงาน 4 รูปแบบ ได้แก่ หยุด เดินหน้า เบรกและถอยหลัง โดยมีลักษณะการสั่งงานจากวิทยุบังคับแบบไกปืน ดังภาพที่ 31



ภาพที่ 31 แสดงลักษณะการสั่งงาน และการตอบสนองของ MC330CR

จากภาพที่ 31 เนื่องจากการสั่งงานด้วยวิทยุบังคับแบบไกปืนโดยตรงเป็นข้อมูลจากคู่มือ MC330CR จึงไม่สามารถแสดงให้เห็นลักษณะของสัญญาณควบคุมจริง ดังนั้น จึงต้องแปลงข้อมูลดังกล่าวให้อยู่ในรูปสัญญาณควบคุม เพื่อนำไปใช้เขียน Code ควบคุมการทำงานของ MC330CR โดยสามารถเปลี่ยนลักษณะสัญญาณได้ดังภาพที่ 32



ภาพที่ 32 ลักษณะของสัญญาณในการสั่งงาน

จากภาพที่ 32 เป็นลักษณะของสัญญาณในการสั่งงาน 4 แบบ ถ้าต้องการสั่งให้ Motor หมุนในทิศทางต่างๆ จะต้องสร้าง PWM เป็นสัญญาณดังตารางที่ 5

ตารางที่ 5 การสั่งงานในแต่ละรูปแบบการเคลื่อนที่

ลำดับที่	ทิศทางการเคลื่อนที่	การสั่งงาน
1	หยุด	PWM Ton=1.5ms.
2	เดินหน้า	PWM 2.0>Ton>1.5ms.
3	ถอยหลัง	PWM 1.5>Ton=1.5ms.→Ton<1.5 ms→ Ton=1.5 ms→2.0>Ton>1.5ms.

จากตารางที่ 5 การสั่งให้เคลื่อนที่ถอยหลังนั้น จำเป็นต้องส่งสัญญาณออกไปเป็นชุด โดยเครื่องหมาย (⟶) ในตารางเป็นการหน่วงเวลาก่อนส่งสัญญาณลักษณะต่อไป ดังนั้น เพื่อความสะดวกในการใช้งาน ESC จึงนำข้อมูลในตารางมาเขียนเป็นฟังก์ชั่นควบคุมการทำงานของ ESC ได้ดังตัวอย่างที่ 6 ในภาคผนวก ข จากตัวอย่างโปรแกรมที่ 6 ในภาคผนวก ข เป็นการสร้าง PWM ควบคุมการทำงานของ ESC MC330CR โดยเพิ่มฟังก์ชั่น delay(count1); และ esc_control(m,s); เข้าไปในโปรแกรมสร้าง PWM สำหรับควบคุม Servo เดิม

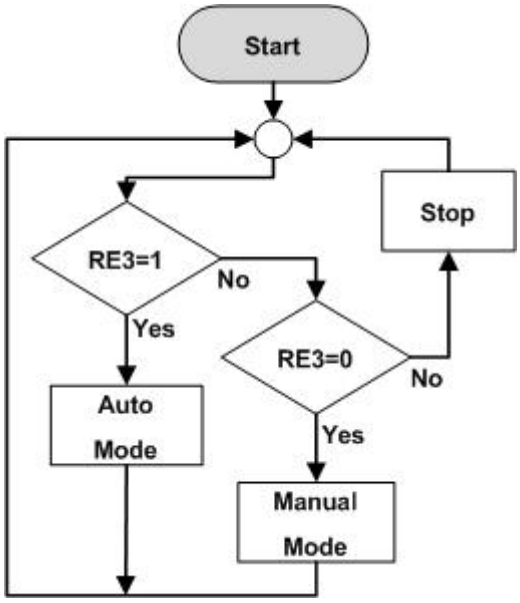
delay(count1); เป็นฟังก์ชันหน่วงเวลา มีค่าตั้งแต่ 1-4294967296 ค่า 1 ค่าหมายถึง 1 Cycle การทำงาน 67.8168 ns ดังนั้นการ delay(40000); เป็นการหน่วงเวลาให้ทำคำสั่งดังกล่าวเป็นเวลา 40000X67.8168 ns = 2.7 ms

esc_control(m,s); เป็นฟังก์ชันสั่งงาน ESC โดยที่ m คือ Motor ที่ต้องการควบคุม มีค่าเป็น 1 หรือ 2 (ใน Code จริงใช้ควบคุม Motor 2 ตัว) s เป็นความเร็วและทิศทางที่ต้องการ มีค่าตั้งแต่ 100-200 ถ้า s เป็น 150 Motor จะหยุดหมุน, s<150 motor หมุนไปข้างหน้า และ s>150 motor หมุนกลับหลัง โดยที่ s = 100 และ s = 200 เป็นความเร็วสูงสุดของแต่ละทิศทาง

จากตัวอย่างโปรแกรมที่ 6 ในภาคผนวก ข Motor 1 หยุดหมุนเป็นเวลา 100000X67.8168 ns = 0.0474 s หลังจากนั้นจะหมุนไปข้างหน้าด้วยความเร็วสูงสุด เป็นเวลา 0.0474 s และหมุนกลับหลังด้วยความเร็วสูงสุด เป็นเวลา 0.0474 s ทำซ้ำเช่นนี้ไปเรื่อยๆ

2.2 ปัญญาประดิษฐ์ของหุ่นยนต์สำรวจท่อแอร์

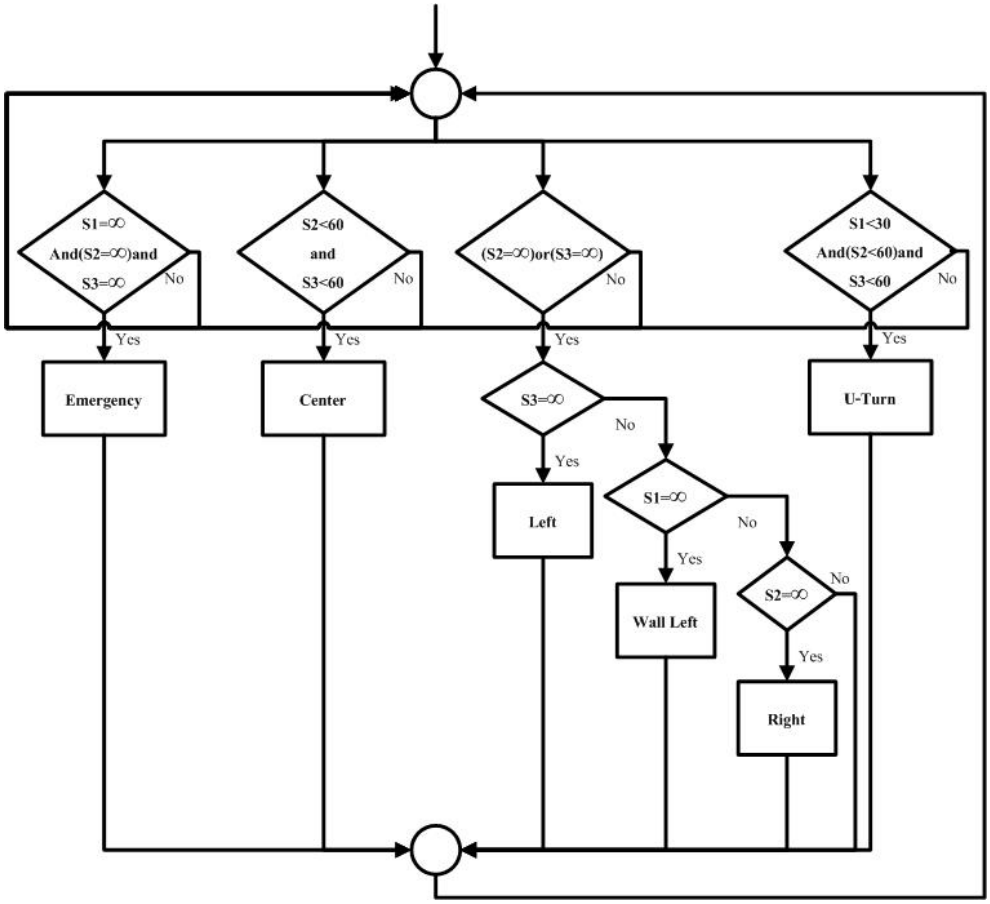
การทำงานของหุ่นยนต์จะแบ่งออกเป็น 2 โหมดการทำงาน โดยมีการตรวจสอบสัญญาณ Digital Input ที่ขา RE3 ถ้า RE3 = 1 เป็น โหมด AUTO ถ้า RE3 = 0 เป็นโหมด Manual ดังภาพที่ 33



ภาพที่ 33 Flow Chart การตรวจสอบโหมดการทำงานของหุ่นยนต์

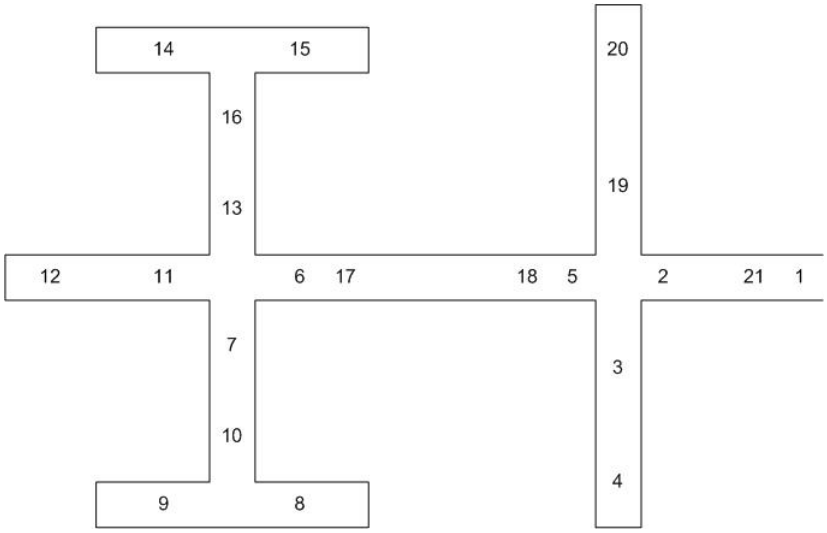
2.2.1 การเขียนโปรแกรมในโหมด AUTO

ในโหมด Auto จะรับสัญญาณจาก Distance sensor 3 ตัว เป็น Input ให้กับ Controller Board เพื่อเป็นข้อมูลในการเลือกโหมดการทำงานย่อยของแต่ละลักษณะการเคลื่อนที่ และจะทำงานในโหมดการทำงานย่อยนั้น จนกว่าจะเป็นไปตามเงื่อนไขจำกัดของแต่ละโหมดการทำงานย่อยดังกล่าว จากนั้นจึงออกจากโหมดและตรวจสอบสภาวะการทำงานต่อไปอีก หลักการทำงานในโหมดอัตโนมัติ มีข้อกำหนดในการทำงาน คือ เมื่ออยู่ในท่อตรงหุ่นยนต์จะเคลื่อนที่บริเวณกลางท่อ, เมื่อพบทางตันจะกลับตัว, เมื่อถึงทางแยก ถ้ามีแยกทางซ้ายจะไปทางซ้ายก่อนเสมอถ้าไม่มีแยกทางซ้ายจะไปทางตรงถ้าไม่มีแยกทางซ้ายและทางตรงจะไปทางขาว สามารถเขียนเป็น Flowchart การทำงานได้ดังภาพที่ 34 โดยมีลักษณะการติดตั้ง Distance Sensor 3 ตัว เป็นดังภาพที่ 10



ภาพที่ 34 Flow Chart การทำงานในโหมด Auto

จากหลักการเขียนโปรแกรมดัง Flowchart หุ่นยนต์สามารถทำการสำรวจได้ทั่วบริเวณ. ท่อแอร์ และกลับออกมาทางจุดเดิมได้ดังตัวอย่างท่อแอร์จำลองภาพที่ 35 ในการเขียนโปรแกรมควบคุมการทำงานในโหมดอัตโนมัติ จะแบ่งการทำงานออกเป็นโหมดการทำงานย่อย 6 โหมดการทำงาน คือ Center Mode, Left Mode, Left wall Mode, Right mode, U-Turn mode และ Emergency Mode โดยมีหลักการเขียนโปรแกรม ดังนี้

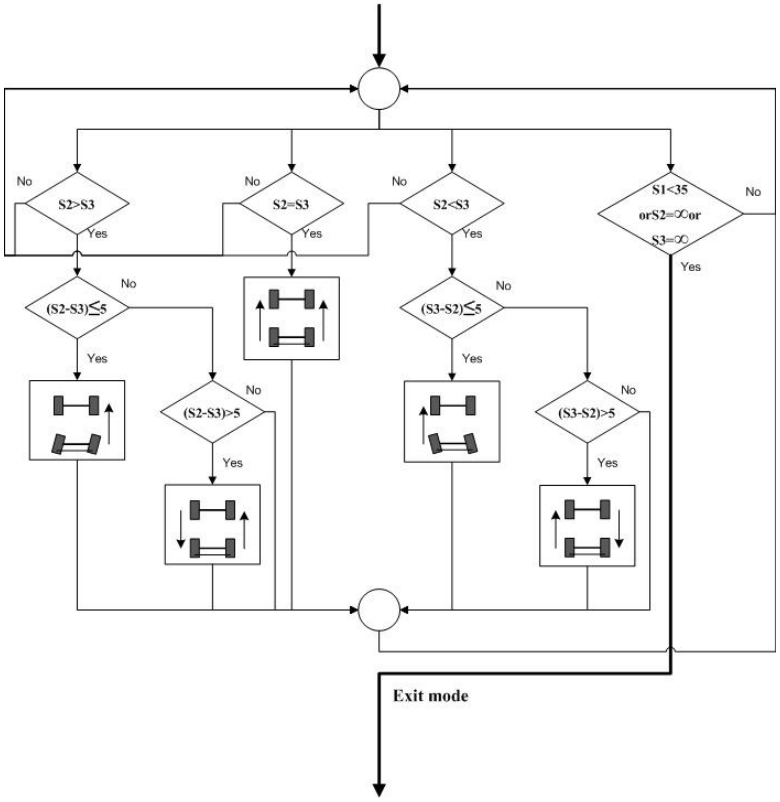


ภาพที่ 35 ลำดับการเคลื่อนที่ของหุ่นยนต์ตามหลักการเขียนโปรแกรมในแผนที่จำลอง

ก. Center Mode

โหมดการทำงานย่อย Center Mode เป็นโหมดการทำงานในท่อตรง โดยให้หุ่นยนต์พยายามเคลื่อนอยู่กลางท่อเสมอ เพื่อให้ง่ายต่อการทำงานในช่วงตรวจสอบทางแยก การเลี้ยวและการกลับตัวในทางตัน

การทำงานในโหมดนี้จะใช้ระยะ ที่อ่านได้จาก Sensor 2 และ Sensor 3 ในการควบคุมการเคลื่อนที่ของหุ่นยนต์ โดยใช้หลักการทำงานดัง Flowchart ในภาพที่ 36

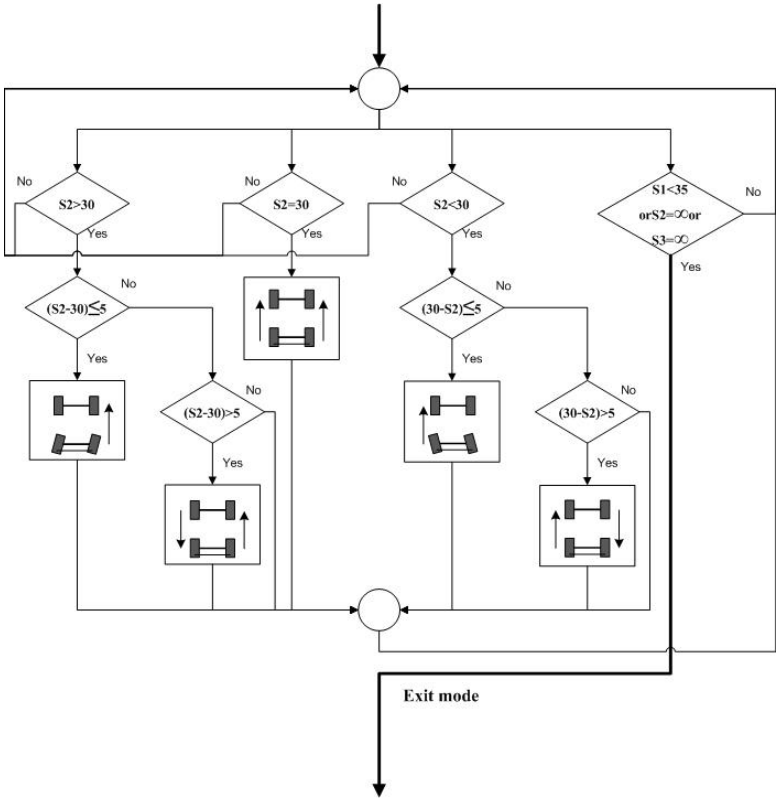


ภาพที่ 36 Flow Chart การทำงานในโหมดการทำงานย่อย Center Mode

หลักการทำงาน เมื่อระยะจาก Sensor 2 = Sensor 3 ให้หุ่นยนต์เคลื่อนที่ตรงไป และถ้าระยะทางของ Sensor ฝั่งใดมากว่า ให้หุ่นยนต์เลี้ยวไปในทิศทางนั้น โดยมีการเลี้ยว 2 ลักษณะขึ้นกับความแตกต่างของระยะห่างของ Sensor ทั้ง 2 ตัว

ข. Left and Left wall Mode

โหมดการทำงานย่อย Left Mode และ Left wall Mode (Jones, 2004) เป็นโหมดการทำงานที่หุ่นยนต์จะรักษาระยะห่างทางด้านซ้ายให้คงที่ เท่ากับค่าที่กำหนด ในที่นี้ระยะห่างทางด้านซ้าย คือ ค่าระยะของ Sensor 2 การทำงานในโหมดการทำงานย่อยนี้จะใช้เมื่อต้องการให้หุ่นยนต์เลี้ยวซ้าย หรือ ต้องการให้หุ่นยนต์ตรงผ่านทางแยกที่เป็น 3 แยก ซึ่งมีหลักการทำงานดัง Flowchart ในภาพที่ 37

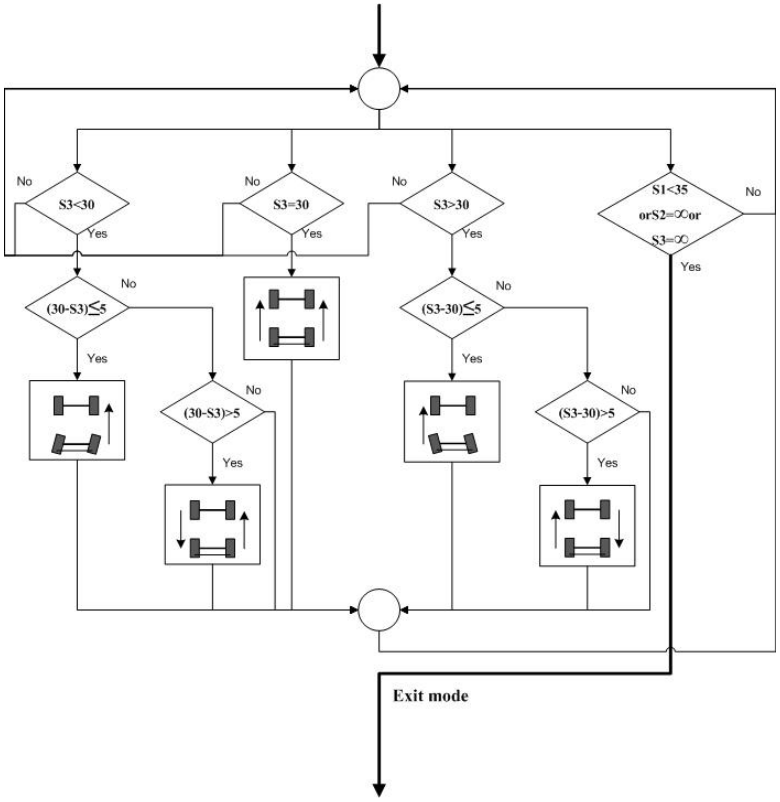


ภาพที่ 37 Flow Chart การทำงานในโหมดการทำงานย่อย Left and Left wall

จากภาพที่ 37 เมื่อ S2 = 30 หุ่นยนต์จะเคลื่อนที่ตรงไป ถ้า S2>30 หุ่นยนต์จะเลี้ยวเข้าหาผนัง และถ้า S2<30 จะเลี้ยวออกจากผนัง โดยมีระดับการเลี้ยว 2 ระดับ ขึ้นอยู่กับระยะห่างจากจุดที่ต้องการถึงค่าที่อ่านได้จริงจาก Sensor 2

ค. Right Mode

โหมดการทำงานย่อย Right Mode เป็นโหมดการทำงานที่จะให้หุ่นยนต์รักษาระยะห่างทางด้านขวาให้คงที่ เท่ากับค่าที่กำหนด ในที่นี้ระยะห่างทางด้านขวา คือ ค่าระยะของ Sensor 3 การทำงานในโหมดย่อยนี้จะใช้เมื่อต้องการเลี้ยวขวา โดยมีหลักการทำงานดัง Flowchart ในภาพที่ 38

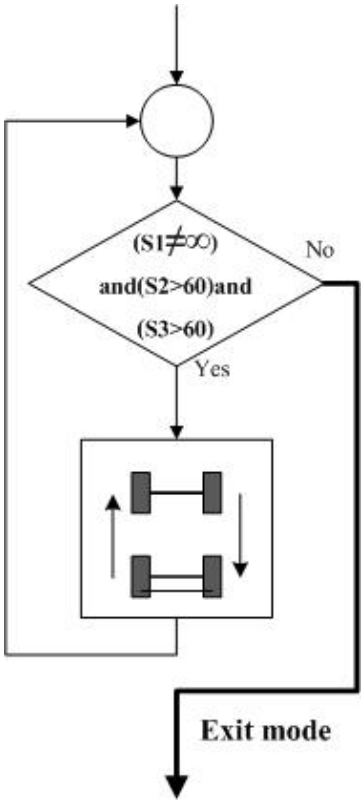


ภาพที่ 38 Flow Chart การทำงานในโหมดการทำงานย่อย Right

จากภาพที่ 38 เมื่อ S3=30 หุ่นยนต์จะเคลื่อนที่ตรงไป ถ้า S3>30 หุ่นยนต์จะเลี้ยวเข้าหาผนัง และถ้า S3<30 จะเลี้ยวออกจากผนัง โดยมีระดับการเลี้ยว 2 ระดับ ขึ้นอยู่กับระยะห่างจากจุดที่ต้องการถึงค่าที่อ่านได้จริงจาก Sensor 3

ง. U-Turn Mode

โหมดการทำงานย่อย U-Turn Mode เป็นโหมดการทำงานที่ใช้เมื่อพบท่อตัน เพื่อให้หุ่นยนต์หมุนตัว และกลับมาทำงานในโหมดการทำงานย่อยอื่นต่อไปได้ โดยมีหลักการทำงานดัง Flowchartในภาพที่ 39



ภาพที่ 39 Flow Chart การทำงานในโหมดการทำงานย่อย U-Turn

จากภาพที่ 39 หุ่นยนต์จะหมุนตัวไปจนกว่า Sensor 1 = ∞ และ Sensor 2, Sensor 3 มีระยะน้อยกว่า 60 จึงจะหยุด กล่าวคือ หุ่นยนต์จะหมุนตัวไปจนกว่าจะเข้าสู่เงื่อนไขการทำงานใน Center Mode

จ. Emergency Mode

โหมดการทำงานย่อย Emergency Mode เป็นโหมดการทำงานที่แสดงว่ามีสิ่งผิดปกติเกิดขึ้น ด้วยการหยุดการเคลื่อนที่ของหุ่นยนต์ และส่งสัญญาณความผิดปกติออกมา หุ่นยนต์จะอยู่ในโหมดนี้จนกว่าความผิดปกติดังกล่าวถูกแก้ไข ในการเขียนโปรแกรมในที่นี้ความผิดปกติที่กล่าวถึงคือ Sensor ทั้ง 3 ตัว ไม่สามารถวัดระยะทางได้ มีค่าเป็นอนันต์ หรือมีค่าน้อยมาก ซึ่งจะเกิดในกรณีที่มีการชนขึ้น

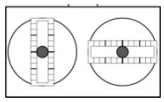
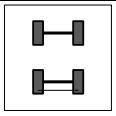
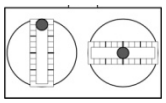
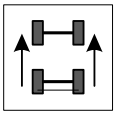
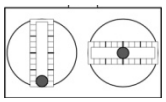
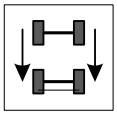
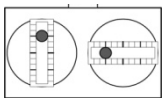
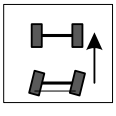
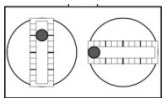
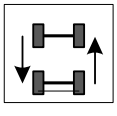
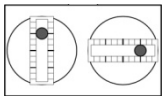
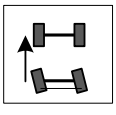
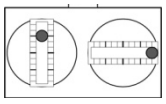
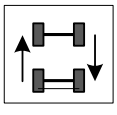
2.2.2 การเขียนโปรแกรมในโหมด Manual

การทำงานในโหมด Manual เป็นโหมดการทำงานที่ควบคุมการเคลื่อนที่ของหุ่นยนต์ด้วยตนเองผ่านวิทยุบังคับ เนื่องจากระบบขับเคลื่อนของหุ่นยนต์เป็นแบบผสมระหว่าง Tracked drive และ Ackerman Steering จึงทำให้เกิดการเคลื่อนที่ได้หลายรูปแบบ ดังภาพที่ 5 การควบคุมการเคลื่อนที่ของหุ่นยนต์จะรับข้อมูลจากวิทยุบังคับเพื่อกำหนดรูปแบบการเคลื่อนที่ โดยมีรูปแบบการควบคุมและการเคลื่อนที่ของหุ่นยนต์ดังตารางที่ 6 โดยค่า SP และ LR คือ ค่าที่รับจาก Receiver โดยใช้โมดูลตรวจจับสัญญาณนาฬิกาหาช่วงเวลา Ton ของทั้ง 2 ช่องสัญญาณ ออกมาเป็น SP และ LR

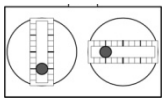
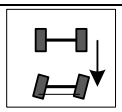
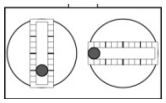
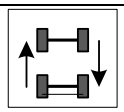
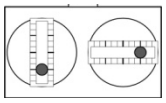
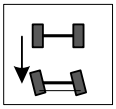
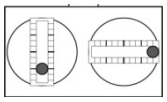
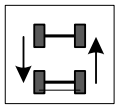
SP คือ ค่าความเร็ว ซึ่งมีค่าระหว่าง 100-200 ถ้า SP = 150 คือ หุ่นยนต์จะหยุด 150>SP>100 หนุ่นยนต์จะเดินหน้า 150<SP<200 หนุ่นยนต์จะถอยหลัง

LR คือ ปริมาณทางซ้ายขวา ใช้บังคับเลี้ยว ซึ่งมีค่าระหว่าง 100-200 ถ้า LR=150 คือ ไม่มีการเลี้ยว 150>LR>100 เลี้ยวซ้าย 150<LR<200 เลี้ยวขวา

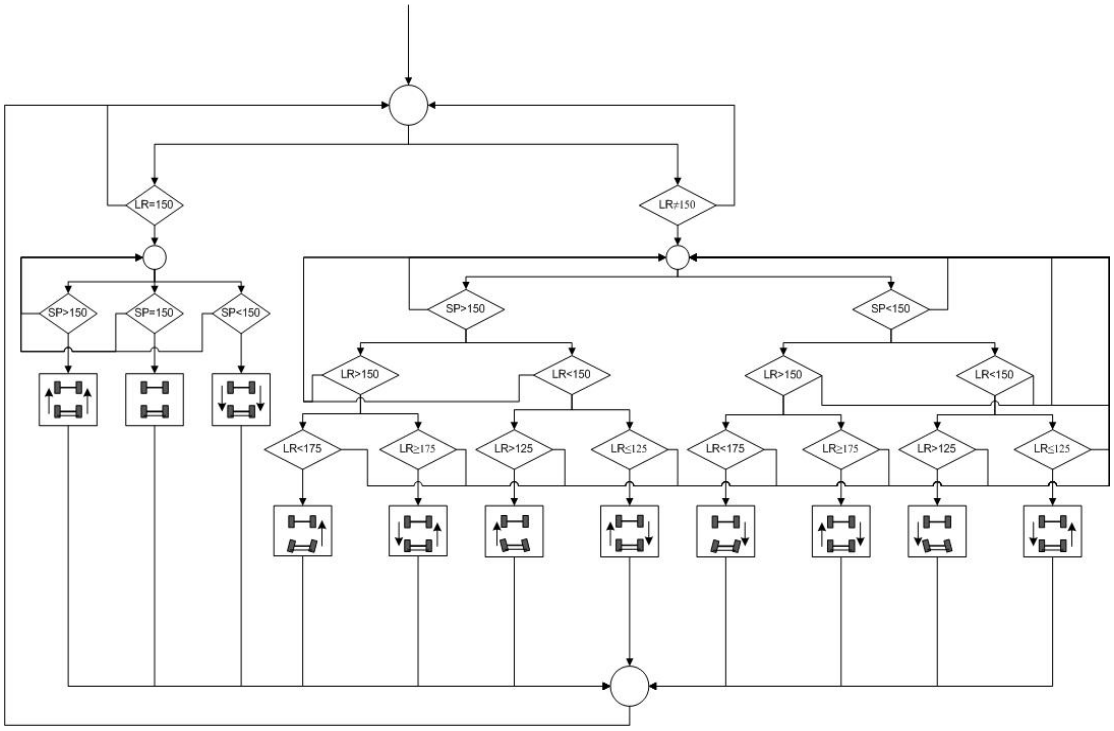
ตารางที่ 6 ลักษณะการควบคุมและการเคลื่อนที่ของหุ่นยนต์

ลำดับที่	ลักษณะการควบคุม	การเคลื่อนที่ของหุ่นยนต์	ลำดับที่	ลักษณะการควบคุม	การเคลื่อนที่ของหุ่นยนต์
1					
2			3		
4			5		
6			7		

ตารางที่ 6 (ต่อ)

ลำดับที่	ลักษณะการควบคุม	การเคลื่อนที่ของหุ่นยนต์	ลำดับที่	ลักษณะการควบคุม	การเคลื่อนที่ของหุ่นยนต์
8			9		
10			11		

จากข้อมูลดังตารางที่ 6 สามารถเขียนเป็น Flowchart การทำงานได้ดังภาพที่ 40 หลักการทำงานจะแบ่งการตรวจสอบออกเป็น 2 ชุด คือ ตรวจสอบว่า LR = 150 หรือไม่ กับ LR ≠ 150 ถ้า LR = 150 จะแบ่งเป็น 3 กรณี คือ หุ่นยนต์จะเดินหน้าตาม SP ที่ต้องการ, หยุด และหุ่นยนต์จะถอยหลังตาม SP ที่ต้องการ ถ้า LR ≠ 150 แสดงว่ามีการบังคับเลี้ยว จะแยกออกเป็น 2 กรณี คือ เลี้ยวขณะเดินหน้า และ เลี้ยวขณะถอยหลัง เนื่องการลักษณะการเลี้ยวของหุ่นยนต์ในขณะเดินหน้าและถอยหลังแตกต่างกัน โดยตรวจสอบค่า SP ถ้า SP>150 เป็นการเลี้ยวขณะเดินหน้า SP<150 เป็นการเลี้ยวขณะถอยหลัง การเลี้ยวจะมีการเลี้ยว 2 ระดับ คือ การเลี้ยวมุมกว้าง จะใช้การบังคับมุมเลี้ยวที่ล้อหลัง และขับล้อฝั่งที่ทำให้เกิดการเลี้ยวในทิศทางที่ต้องการ ส่วนการเลี้ยวมุมแคบจะใช้การเลี้ยวแบบหมุนตัว โดยการบังคับล้อตรงและขับล้อทั้งสองฝั่งสวนทางกัน ทำให้เกิดการหมุนตัว



ภาพที่ 40 Flow Chart การทำงานในโหมด Manual

การเขียน Code รวมของหุ่นยนต์สำรวจท่อแอร์จะแสดงในภาคผนวก ค

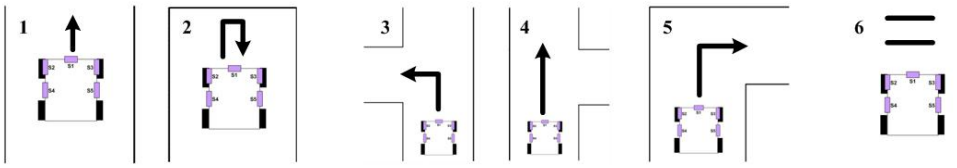
3. การสร้างแผนที่

การสร้างแผนที่เพื่อบอกเส้นทางการเคลื่อนที่ของหุ่นยนต์ จะใช้การหาตำแหน่งของหุ่นยนต์ในขณะที่หุ่นยนต์เคลื่อนที่ และนำข้อมูลตำแหน่งมาใช้ในการสร้างแผนที่ โดยอาจใช้ข้อมูลของสภาพแวดล้อม เวลา ความเร็ว ระยะทางและอื่นๆ ร่วมในการสร้างแผนที่ด้วย

การหาตำแหน่งของหุ่นยนต์มีหลายวิธี (Bräunl, 2006) เช่น วิธีการหาตำแหน่งจากการหมุนของล้อ ซึ่งวิธีนี้มีความซับซ้อนในการคำนวณ และมีความคลาดเคลื่อนสะสม เนื่องจากล้ออาจมีการลื่นไถลได้ ทำให้ได้แผนที่มีความคลาดเคลื่อนสูง วิธีการหาตำแหน่งจากจุดอ้างอิง เช่น GPS เป็นวิธีที่ใช้ได้ดีในพื้นที่ที่มีขนาดใหญ่ และมีต้นทุนสูง การหาตำแหน่งของหุ่นยนต์ด้วยวิธีนี้จึงไม่เหมาะที่จะใช้กับการการสร้างแผนที่ในพื้นที่ที่มีขนาดเล็ก

สำหรับหุ่นยนต์สำรวจท่อแอร์มีระบบขับเคลื่อนคล้ายกับระบบสายพานขับเคลื่อน การหาตำแหน่งของหุ่นยนต์จากการหมุนของล้อ มีความคลาดเคลื่อนสูง เนื่องจากพื้นฐานของระบบขับเคลื่อนจะมีการลื่นไถลในขณะเลี้ยว และในขณะกลับตัวเสมอ วิธีการหาตำแหน่งของหุ่นยนต์จากการหมุนของล้อจึงไม่เหมาะสม การหาตำแหน่งจากจุดอ้างอิง ใช้ได้ดีในพื้นที่ที่มีขนาดใหญ่ จึงไม่เหมาะสมกับการใช้งานภายในอาคาร วิธีการสร้างแผนที่แบบใช้ข้อกำหนดตายตัว เป็นตัวเลือกที่ดี ไม่ยุ่งยากและต้นทุนต่ำ ถึงแม้จะมีความแม่นยำต่ำ แต่ยังเพียงพอที่จะบอกตำแหน่งของหุ่นยนต์ได้เช่นกัน

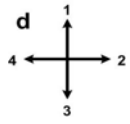
สำหรับหุ่นยนต์สำรวจท่อแอร์นั้น จะสร้างแผนที่เฉพาะในโหมดการเคลื่อนที่แบบ Auto เท่านั้น เนื่องจากกระบวนการสร้างแผนที่จะใช้ข้อมูลโหมดการทำงานย่อย จากการทำงานในโหมด Auto เป็นข้อมูลในการสร้างแผนที่ โดยจะกำหนดเป็นค่าคงที่ให้สอดคล้องกับโหมดการทำงานย่อยแต่ละโหมด เพื่อใช้เป็นข้อมูลในการสร้างแผนที่ ดังนั้น จำเป็นต้องทราบถึงหลักการทำงานในโหมด Auto ก่อน ดังภาพที่ 41



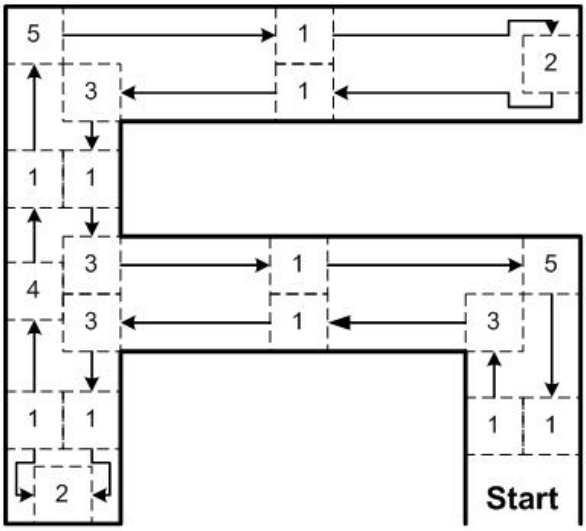
ภาพที่ 41 ลักษณะการเคลื่อนที่ในโหมด Auto

จากโหมดการทำงานย่อยทั้ง 6 โหมดดังภาพที่ 41 สามารถสร้างแผนที่ในพิกัด (X,Y) ได้ด้วยการกำหนดพิกัดเริ่มต้นของ X,Y และทิศทางเริ่มต้น (d) เป็นค่าคงที่ เมื่อมีการเปลี่ยนโหมดการเคลื่อนที่ ทำการบวกค่า X,Y ที่อ้างอิงจากทิศทางของการเคลื่อนที่เดิม และทำการเปลี่ยนค่าทิศทางการเคลื่อนที่เพื่อใช้อ้างอิงในจุดต่อไป โดยมีค่าการเปลี่ยนแปลงของ X,Y และ d ดังตารางที่ 7

ตารางที่ 7 ความสัมพันธ์ของโหมดการเคลื่อนที่กับการเปลี่ยนแปลงค่าพิกัด (X, Y) และ d

Mode	ลักษณะการเคลื่อนที่ 	ค่าปรับพิกัด(X,Y) และ d											
		d1			d2			d3			d4		
		X	Y	d	X	Y	d	X	Y	d	X	Y	d
1	ตรงในท่อตรง	0	10	1	10	0	2	0	-10	3	-10	0	4
2	กลับตัวในทางตัน	2	0	3	0	-2	4	-2	0	1	0	2	2
3	เลี้ยวซ้ายในทางแยก	-2	0	4	0	2	1	2	0	2	0	-2	3
4	ตรงในทางแยก	0	6	1	0	-4	2	0	-4	3	4	0	4
5	เลี้ยวขวาในทางแยก	2	0	2	0	-2	3	-2	0	4	0	2	1
6	หยุดเมื่อเกิดความผิดปกติ	0	0	0	0	0	0	0	0	0	0	0	0

จากข้อมูลในตารางที่ 7 สามารถสร้างแผนที่ในพิกัด (X, Y) ได้ เช่น กำหนด (X, Y) เริ่มต้นที่ (100,100) และd = 1 เมื่อมีการเปลี่ยนโหมดเป็น 1 ค่า X,Y จะเปลี่ยน โดย X=100+0,Y=100+10 และ d = 1 เมื่อโหมดเปลี่ยนเป็น 3 ค่าของ X=100-2, Y=110+0 และ d=4 จะได้ (X, Y) จำนวน 3 จุด คือ (100, 100), (100, 110) และ (98.110) ส่วนค่า d ใช้เป็นค่าอ้างอิงในการบอกทิศทาง เพื่อหาค่าการเปลี่ยน (X, Y) ในแต่ละโหมด ดังตัวอย่างของแผนที่ ดังภาพที่ 42



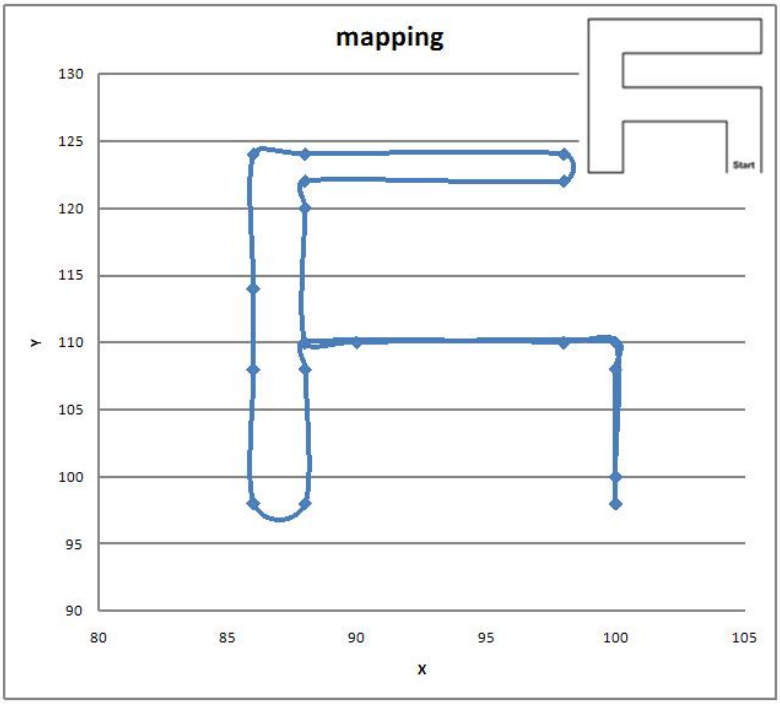
ภาพที่ 42 ตัวอย่างแผนที่และโหมดการเคลื่อนที่

จากภาพที่ 42 เป็นแผนที่จำลองเพื่อใช้ทดสอบกระบวนการสร้างแผนที่จากจุด Start หุ่นยนต์จะเคลื่อนที่ตามหัวลูกศร โดยหุ่นยนต์จะส่งข้อมูลโหมดการเคลื่อนที่กลับมา เมื่อมีการเปลี่ยนแปลงโหมดการทำงาน ซึ่งจากภาพจะมีการเปลี่ยนโหมด 19 ครั้ง ดังนี้ 1-3-1-3-1-2-1-4-1-5-1-2-1-3-1-3-1-5-1 และจะได้ พิกัด (X, Y) จำนวน 20 จุด มีค่าดังตารางที่ 8

ตารางที่ 8 ค่าพิกัด (X, Y) และ d ที่เปลี่ยนแปลงไปตามโหมดการเคลื่อนที่

ลำดับ	mode	dir	X	Y	ลำดับ	Mode	dir	X	Y
Start	-	1	100	100	10	5	2	88	124
1	1	1	100	110	11	1	2	98	124
2	3	4	98	110	12	2	4	98	122
3	1	4	88	110	13	1	4	88	122
4	3	3	88	108	14	3	3	88	120
5	1	3	88	98	15	1	3	88	110
6	2	1	86	98	16	3	2	90	110
7	1	1	86	108	17	1	2	100	110
8	4	1	86	114	18	5	3	100	108
9	1	1	86	124	19	1	3	100	98

จากข้อมูลดังตารางที่ 8 สามารถนำพิกัด (X, Y) ทั้ง 20 จุด (รวมจุดเริ่มต้น) ไปเขียนกราฟจะได้แผนที่ดังภาพที่ 43 ซึ่งมีลักษณะใกล้เคียงกับแผนที่จริง แต่มีข้อจำกัดในเรื่องลักษณะของสนามและมาตราส่วนของแผนที่ คือ ลักษณะของแผนที่จะต้องเลี้ยวทำมุม 90 องศา และเนื่องจากไม่มีการรับข้อมูลความเร็ว เวลา หรือระยะทางในการเคลื่อนที่ แผนที่ที่สร้างขึ้นจึงไม่เป็นไปตามสัดส่วนที่แท้จริง



ภาพที่ 43 แผนที่ที่ได้จากพิกัด (X, Y)

4. ออกแบบการทดลอง

4.1 การทดสอบคุณสมบัติของหุ่นยนต์

4.1.1 การขึ้นทางชัน

ทำการทดสอบโดยการสร้างพื้นเอียงที่สามารถปรับมุมเอียงได้ โดยทดลองให้หุ่นยนต์เคลื่อนที่ขึ้นพื้นเอียงในแต่ละมุมเอียง บันทึกผลมุมเอียงสูงสุดที่หุ่นยนต์สามารถเคลื่อนที่ขึ้นได้

4.1.2 ความเร็วในการเคลื่อนที่

ทำการทดสอบโดยการวัดระยะทางบนพื้น และทำเครื่องหมายบอกระยะทาง ให้หุ่นยนต์เคลื่อนที่ไปบนพื้นที่มีเครื่องหมายบอกระยะทาง โดยถ่าย Video การเคลื่อนที่ไว้ คำนวณหาความเร็วการเคลื่อนที่จากเวลาและระยะทาง ด้วยการดูจากไฟล์ Video

4.1.3 ระยะไกลที่สุดในการรับส่งข้อมูลของอุปกรณ์ไร้สาย

ทำการทดสอบโดยการเปิดระบบส่งสัญญาณไร้สายทั้งหมด และให้หุ่นยนต์เคลื่อนที่ออกจากจุดรับส่งสัญญาณในพื้นที่เปิดโล่งด้วยโหมด Manual (วิทยุบังคับมีระยะรับส่งข้อมูลไกลที่สุด ตามคู่มือการใช้งาน) สังเกตการส่งสัญญาณของอุปกรณ์ทุกตัว เมื่ออุปกรณ์ตัวใดเริ่มส่งสัญญาณไม่ได้ หรือสัญญาณได้ไม่สมบูรณ์ ให้ตำแหน่งดังกล่าวเป็นระยะการส่งสัญญาณที่ไกลที่สุดของอุปกรณ์นั้น วัดระยะทางจากจุดส่งสัญญาณไปยังจุดสูงสุดของแต่ละอุปกรณ์และบันทึกผล

4.2 การทดลองการทำงานในโหมดอัตโนมัติ

ทำการทดลองโดยการสร้างท่อแอร์ของระบบปรับอากาศจำลอง และให้หุ่นยนต์ทำการสำรวจท่อแอร์ของระบบปรับอากาศจำลองที่สร้างขึ้น ด้วยโหมดอัตโนมัติ สังเกตการเคลื่อนที่ของหุ่นยนต์

4.3 การทดลองการทำงานในโหมดควบคุมด้วยตนเอง

ทำการทดลองโดยการสร้างท่อแอร์ของระบบปรับอากาศจำลอง และให้หุ่นยนต์ทำการสำรวจท่อแอร์ของระบบปรับอากาศจำลองที่สร้างขึ้น ด้วยโหมดควบคุมด้วยตนเอง และสังเกตการเคลื่อนที่ของหุ่นยนต์

4.4 การทดลองสร้างแผนที่

ทำการทดลองโดยการให้หุ่นยนต์เคลื่อนที่ในโหมด AUTO ในท่อแอร์ระบบปรับอากาศจำลองที่สร้างขึ้น และรับสัญญาณโหมดการทำงานย่อยของหุ่นยนต์ นำข้อมูลโหมดการทำงานย่อยที่ได้รับมาสร้างเป็นแผนที่

ผลและวิจารณ์

1. ผลการทดสอบคุณสมบัติของหุ่นยนต์

1.1 การทดสอบการขึ้นทางชัน

ตารางที่ 9 ผลการทดลองการเคลื่อนที่ขึ้นทางชัน

ครั้งที่	มุมทดสอบ (องศา)	ผลการทดสอบ (ผ่าน/ไม่ผ่าน)
1	20	ผ่าน
2	21	ผ่าน
3	22	ผ่าน
4	23	ผ่าน
5	24	ผ่าน
6	25	ผ่าน
7	26	ไม่ผ่าน

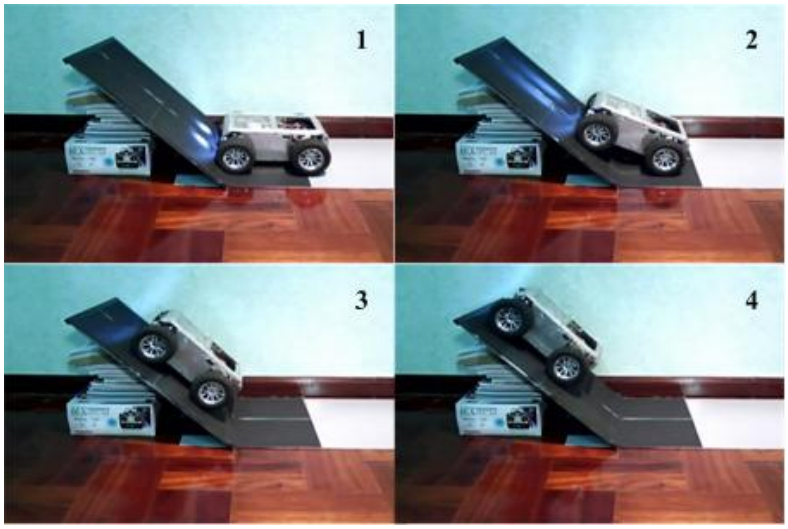
จากข้อมูลดังตารางที่ 9 หุ่นยนต์สามารถเคลื่อนที่ขึ้นทางชันสูงสุดได้ 25 องศา ค่าที่ใช้คำนวณในการออกแบบ คือ 30 องศา และค่าความปลอดภัยที่ 3 จากการสังเกตการณ์เคลื่อนที่ขึ้นทางชันของหุ่นยนต์ที่ 26 องศา ล้อทั้ง 4 ล้อของหุ่นยนต์มีการลื่นไถลบริเวณทางขึ้น สาเหตุที่ทำให้หุ่นยนต์ไม่สามารถเคลื่อนที่ขึ้นทางชันที่ 26 องศาได้นั้น อาจมาจากแรงยึดเกาะระหว่างล้อกับพื้นไม่เพียงพอ เนื่องจากในการทดลองวัสดุที่ใช้ทำพื้นเอียงเป็นแผ่นอลูมิเนียมหนา 4 mm

ในขั้นตอนการออกแบบจะคำนวณแรง F_{app} จากแรงเสียดทานรวมกับแรงที่เกิดจากมวลของตัวหุ่นยนต์ ดังสมการที่ 1 แต่จะสามารถส่งแรงลงพื้นได้มากน้อยเพียงใด ขึ้นอยู่กับแรงแรงยึดเกาะระหว่างพื้นกับล้อ ซึ่งถ้าแรงยึดเกาะสูงสุดระหว่างพื้นกับล้อน้อยกว่าแรงที่จำเป็นต้องใช้ในการขึ้นทางชัน (F_{app}) ก็จะเกิดการลื่นไถลก่อน จึงไม่สามารถเคลื่อนที่ขึ้นทางชันได้ ในการออกแบบคำนวณค่า F_{app} = 37.27 N แรงสูงสุดที่สามารถส่งลงสู่พื้นได้คำนวณจากสามการที่

$$F_{\max} = \mu mg \tag{11}$$

จากสมการที่ 11 คำนวณค่า F_{max} โดยกำหนดให้ $\mu = 0.3$, $m = 5$ kg และ $g = 9.81$ m/s^2 (ค่าที่กำหนดเป็นค่าเดียวกับที่ใช้ในการคำนวณ P_{app}) จะได้ $F_{max} = 14.72$ N ซึ่งมีค่าน้อยกว่า F_{app} ดังนั้น หุ่นยนต์จึงเกิดการลื่นไถลก่อน

จากข้อสันนิฐานข้างต้น ทำการทดลองอีกครั้งด้วยการเพิ่มแรงยึดเกาะระหว่างพื้นกับล้อของหุ่นยนต์ โดยการแปะกระดาษทรายที่พื้นเอียง ทำให้หุ่นยนต์เคลื่อนที่ขึ้นทางชันสูงสุดได้ 45 องศา ดังภาพที่ 44 และหยุดทำการทดลองเนื่องจากโซ่ของล้อฝั่งขวามีการหลุดข้ามร่องโซ่ อาจเกิดจากโซ่ส่งกำลังไม่ตึงพอ



ภาพที่ 44 การเคลื่อนที่ขึ้นทางชันที่ 45 องศา

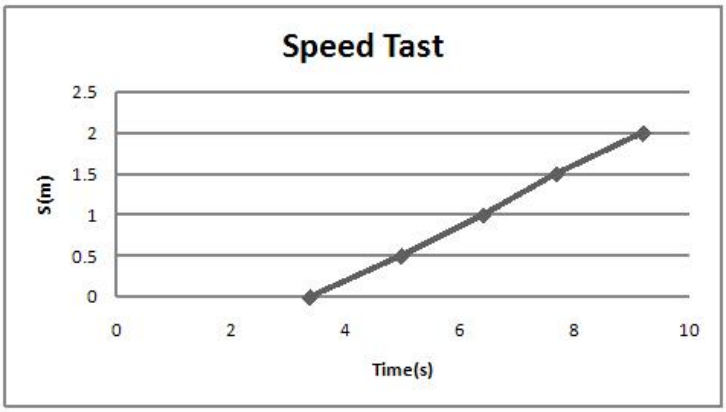
1.2 ความเร็วในการเคลื่อนที่

จากการทดลองจับเวลาการเคลื่อนที่ของหุ่นยนต์ด้วยการแตกไฟล์ Video ตั้งแต่ระยะ 0-2 m ได้เวลาดังตารางที่ 10

ตารางที่ 10 เวลาการเคลื่อนที่ของหุ่นยนต์ในระยะทาง 2 เมตร

ระยะทาง (m)	เวลา (s)
0	3.36
0.5	4.96
1	6.4
1.5	7.68
2	9.2

เขียนกราฟระหว่างระยะทางกับเวลาได้ดังภาพที่ 45



ภาพที่ 45 ความสัมพันธ์ของระยะทางกับเวลาในการเคลื่อนที่

จากภาพที่ 45 เป็นกราฟเส้นตรงแสดงให้เห็นว่าหุ่นยนต์เคลื่อนที่ด้วยความเร็วคงที่ เมื่อนำมาคำนวณความเร็วในการเคลื่อนที่ได้เป็น 2/(9.2-3.36) = 0.342 m/s ซึ่งค่าที่ใช้ออกแบบ คือ 0.15 m/s ความเร็วการหมุนของมอเตอร์เป็น 23.87 rpm เนื่องจากมอเตอร์ที่เลือกใช้รวมกับห้องเกียร์ ทำให้มอเตอร์หมุนที่ความเร็ว 53.4 rpm ที่จุดประสิทธิภาพสูงสุด ความเร็วที่ได้ จึงเป็นไปตามการออกแบบ

1.3 ระยะการส่งข้อมูลไร้สาย

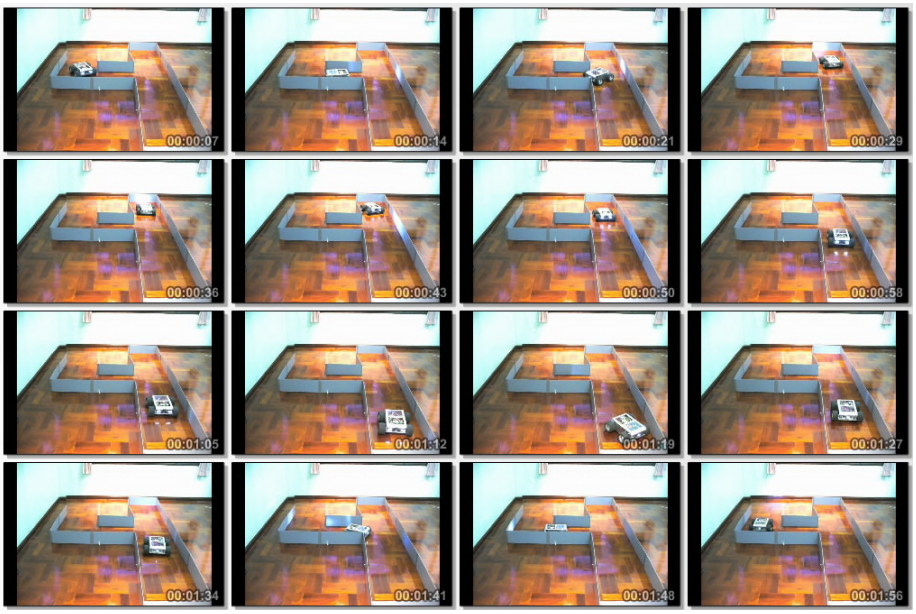
ตารางที่ 11 ระยะทางสูงสุดในการรับส่งข้อมูลของแต่ละอุปกรณ์ไร้สาย

ลำดับที่	ชนิดอุปกรณ์	ระยะการส่งข้อมูลสูงสุดในที่โล่ง
1	วิทยุบังคับ	45 m.
2	RS232 2.4GHz	37 m.
3	กล้อง Video	35 m.

ระยะทางในการรับส่งข้อมูลของอุปกรณ์ไร้สายนั้น สั้นกว่าระยะที่กำหนดไว้ในคู่มือเล็กน้อย เนื่องจากเสารับสัญญาณของระบบที่อยู่ในตัวหุ่นยนต์นั้น ไม่ได้ยืดตรงออกมานอกตัวหุ่นยนต์ เพื่อไม่ให้เสารับสัญญาณต่างๆ กีดขวางการทำงานของหุ่นยนต์ จึงทำให้การรับส่งสัญญาณ มีประสิทธิภาพลดลง

2. การทดลองการทำงานในโหมดอัตโนมัติ

เมื่อหุ่นยนต์ทำการสำรวจท่อแอร์ของระบบปรับอากาศจำลอง หุ่นยนต์สามารถเคลื่อนที่บริเวณกลางท่อได้ดี สามารถเลือกเส้นทางการเคลื่อนที่ได้ถูกต้อง ตามหลักการที่ใช้ในการเขียนโปรแกรม และสามารถเคลื่อนที่กลับมายังจุดเดิมได้ ดังภาพที่ 46 ที่แสดงให้เห็นตำแหน่งของหุ่นยนต์ที่เวลาต่างๆ



ภาพที่ 46 การเคลื่อนที่ในโหมด Auto

อย่างไรก็ตามจากการสังเกตลักษณะการเคลื่อนที่ของหุ่นยนต์ในโหมดกึ่งกลางท่อ และการกลับรถนั้น ยังไม่ราบรื่นเท่าที่ควร กล่าวคือ ในโหมดกึ่งกลางท่อ ถึงแม้หุ่นยนต์จะสามารถเคลื่อนที่กึ่งกลางท่อได้ แต่มีลักษณะส่ายไปมาไม่ราบรื่น สาเหตุอาจมาจากการใช้ความเร็วในการเคลื่อนที่สูงเกินไป หรือค่าของ Sensor ในขณะเคลื่อนที่ตอบสนองไม่ทันกับการเคลื่อนที่ รวมไปถึงการสั่งงาน Driver Motor ที่ต้องสั่งงานเป็นชุดคำสั่ง และต้องมีการหน่วงเวลาการเคลื่อนที่เอาไว้ ซึ่งอาจทำให้การตอบสนองช้า

โหมดกลับตัว ในขณะที่หมุนตัวกลับมีบางครั้งที่ล้อหลังของหุ่นยนต์โดนผนังของท่อแอร์จำลอง เนื่องจากในขณะที่เข้าสู่โหมดกลับตัวนั้นไม่มีการตรวจสอบระยะทางด้านหลัง ประกอบกับจุดศูนย์กลางในการหมุนตัวของหุ่นยนต์ไม่ได้อยู่กึ่งกลาง เนื่องจากการกระจายน้ำหนักไม่เท่ากันในแต่ละล้อ

3. การทดลองการทำงานในโหมดควบคุมด้วยตนเอง

หุ่นยนต์สามารถเคลื่อนที่ในท่อแอร์จำลองได้อย่างราบรื่น มีความคล่องตัว และความเร็วในการเคลื่อนที่ดี การควบคุมการเคลื่อนที่ผ่านวิทยุบังคับทำได้ดี

เนื่องจากสามารถควบคุมความเร็วในการเคลื่อนที่ และรัศมีวงเลี้ยวได้ ประกอบกับมีรูปแบบการเคลื่อนที่หลายรูปแบบ ภายใต้การควบคุมที่ไม่ยุ่งยาก ทำให้สามารถควบคุมการเคลื่อนที่ของหุ่นยนต์ได้ตามต้องการ

4. การสร้างแผนที่

เมื่อทดลองให้หุ่นยนต์ทำการสำรวจท่อแอร์ของระบบปรับอากาศจำลอง ขณะที่ทำการสำรวจหุ่นยนต์ได้ส่งข้อมูลของโหมดการเคลื่อนที่ ผ่าน RS232 ไร้สายที่ความถี่ 2.4 GHz. และใช้โปรแกรม Hyper Terminal ในการรับข้อมูล ได้ข้อมูลโหมดการทำงานย่อยดังตารางที่ 12

ตารางที่ 12 ข้อมูลโหมดการทำงานย่อยในขณะสำรวจท่อแอร์ด้วยโหมด Auto

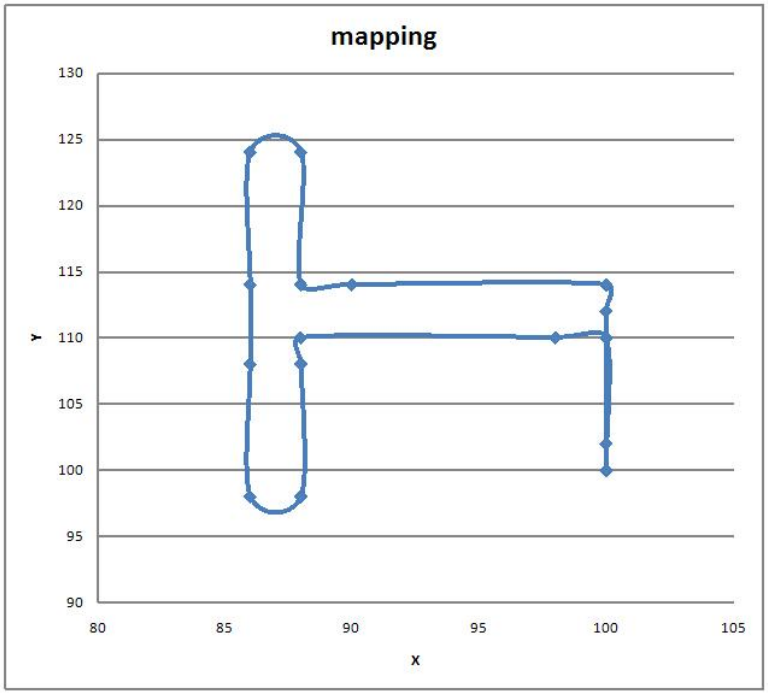
ครั้งที่	Mode	ครั้งที่	Mode
1	1	9	1
2	3	10	2
3	1	11	1
4	3	12	3
5	1	13	1
6	2	14	5
7	1	15	1
8	4	16	6

จากข้อมูลที่ได้เกิดจากการเปลี่ยนโหมดการทำงานย่อยดังตารางที่ 12 นำข้อมูลดังกล่าวผ่านกระบวนการคำนวณเพื่อสร้างแผนที่ ดังที่อธิบายไว้ข้างต้นได้ค่าการคำนวณพิกัด (X, Y) ดังตารางที่ 13

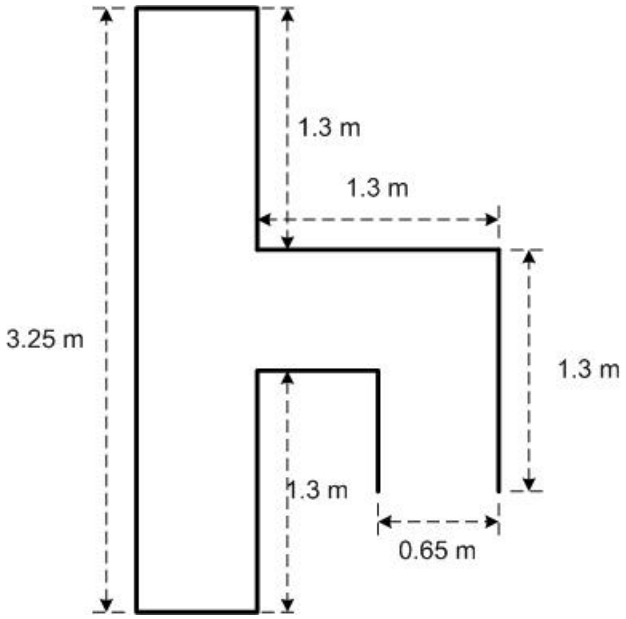
ตารางที่ 13 การเปลี่ยนแปลงของค่า (X, Y) และ d ในแต่ละโหมดการเคลื่อนที่

ลำดับ	mode	dir	X	Y
Start	-	1	100	100
1	1	1	100	110
2	3	4	98	110
3	1	4	88	110
4	3	3	88	108
5	1	3	88	98
6	2	1	86	98
7	1	1	86	108
8	4	1	86	114
9	1	1	86	124
10	2	3	88	124
11	1	3	88	114
12	3	2	90	114
13	1	2	100	114
14	5	3	100	112
15	1	3	100	102

จากข้อมูลในตารางที่ 13 เขียนกราฟของ (X, Y) ได้เป็นแผนที่ดังภาพที่ 47



ภาพที่ 47 แผนแสดงเส้นทางการเคลื่อนที่ของหุ่นยนต์ขณะเคลื่อนที่ในโหมดอัตโนมัติ



ภาพที่ 48 ลักษณะของท่อแอร์จำลอง

แผนที่แสดงการเคลื่อนที่ของหุ่นยนต์ มีลักษณะใกล้เคียงกับท่อแอร์ของระบบปรับอากาศจำลองที่สร้างขึ้น แต่มีขนาดไม่เป็นไปตามสัดส่วน เนื่องจากไม่มีการรับข้อมูลระยะทางหรือเวลาในการเคลื่อนที่ในแต่ละโหมด จึงไม่สามารถกำหนดระยะทางในการเคลื่อนที่ได้ตรงตามสัดส่วนการเคลื่อนที่จริงของหุ่นยนต์

สรุปและข้อเสนอแนะ

สรุป

1. ระบบขับเคลื่อนของหุ่นยนต์ที่ใช้การประยุกต์ ระบบขับเคลื่อนแบบสายพานเข้ากับการบังคับเลี้ยวแบบรถยนต์ ทำให้หุ่นยนต์มีความคล่องตัวในการเคลื่อนที่สูง ทำงานในพื้นที่จำกัดได้ดี ควบคุมการเคลื่อนที่ได้ง่าย

2. คุณสมบัติของหุ่นยนต์ที่ได้คือ ขึ้นทางชันได้ 26 องศา สำหรับพื้นผิวพื้นเอียงที่เป็นอลูมิเนียม และ 45 องศา สำหรับพื้นผิวที่เป็นกระดาษทราย ความเร็วสูงสุดในการเคลื่อนที่ 0.342 m/s ซึ่งคุณสมบัติต่างๆ นั้นเป็นไปตามข้อกำหนดในการออกแบบที่กำหนดไว้ คือ สามารถขึ้นพื้นเอียงได้ 30 องศา และความเร็วสูงสุดที่ 0.15 m/s

3. การส่งข้อมูลไร้สายของแต่ละอุปกรณ์ทำได้ดี สามารถส่งข้อมูลควบคุมการเคลื่อนที่ ข้อมูลภาพ และโหมดการเคลื่อนที่ได้ตามต้องการ ระยะการส่งข้อมูลไร้สายที่สั้นที่สุด เป็นกล้อง Video ไร้สาย โดยมีระยะการส่งข้อมูลที่ 35 m ในพื้นที่เปิดโล่ง

4. แผนที่ที่สร้างขึ้นจากกระบวนการสร้างแผนที่ตามการทดลอง สามารถบอกถึงเส้นทางการเคลื่อนที่ของหุ่นยนต์ ได้ใกล้เคียงกับการเคลื่อนที่ของหุ่นยนต์จริง

5. ข้อมูลภาพที่ได้จากการบันทึก มีคุณภาพพอสมควร ทั้งนี้ขึ้นอยู่กับคุณภาพของตัวกล้อง Video และสามารถบันทึกได้ทั้งภาพนิ่งและภาพเคลื่อนไหว ระยะเวลาในการบันทึกขึ้นอยู่กับปริมาณหน่วยความจำของเครื่องคอมพิวเตอร์

ข้อเสนอแนะ

1. การออกแบบหุ่นยนต์ ควรคำนึงถึงค่าแรงเสียดทานระหว่างพื้นกับล้อ เพื่อไม่ให้เกิดการลื่นไถลในขณะขึ้นทางชัน โดยในเบื้องต้นอาจคำนวณแรงที่สูงสุดที่สามารถส่งกำลังลงพื้นก่อน จากนั้นตรวจสอบว่ามีแรงเพียงพอที่จะนำตัวหุ่นยนต์ขึ้นทางชันในมุมที่ต้องการได้หรือไม่ ถ้าไม่เพียงพออาจเปลี่ยนยางหรือ แก้ไขระบบขับเคลื่อน ก่อนที่จะเลือกมอเตอร์ขับต่อไป

2. เนื่องจากข้อกำหนดในการออกแบบที่ตั้งไว้ ทำให้หุ่นยนต์มีขนาดใหญ่และน้ำหนักมาก ดังนั้น หากมีการออกแบบหรือปรับปรุง ควรศึกษาเลือกการใช้วัสดุนำหนักเบา และการลดขนาดของหุ่นยนต์ลง หรือลดข้อกำหนดในการออกแบบ เพื่อให้หุ่นยนต์มีความคล่องตัวในการเคลื่อนที่และลดการใช้พลังงาน

3. การทำงานของระบบไร้สายที่เลือกใช้นั้นเป็นการทำงานที่แยกอิสระต่อกัน ดังนั้น จึงอาจพัฒนาใช้การรับส่งข้อมูลไร้สายเพียงระบบเดียว โดยอาจใช้การส่งข้อมูลไร้สายความถี่สูง 2.4 GHz ซึ่งอาจเพิ่มระยะการส่งข้อมูลได้โดยการติดตั้ง Access Point เป็นตัวขยายสัญญาณ รวมไปถึงในปัจจุบันมีกล้อง Video ไร้สายที่ส่งข้อมูลที่ความถี่ 2.4 GHz. ด้วยเช่นกัน

4. การสร้างแผนที่แบบใช้ข้อกำหนดตายตัว อาจไม่ครอบคลุมการทำงานในทุกสภาพของพื้นที่ ดังนั้น หากมีการศึกษาหรือพัฒนาเพิ่มเติม โดยรับค่าความเร็ว, เวลา หรือทิศทางการเคลื่อนที่ จะทำให้แผนที่มีความใกล้เคียงกับการเคลื่อนที่จริงมากขึ้น

5. หลักการในการออกแบบหุ่นยนต์สำรวจท่อแอร์ในวิทยานิพนธ์ฉบับนี้ใช้หลักการเดียวกับการออกแบบหุ่นยนต์โดยทั่วไป เพียงแต่ให้ความสำคัญกับการประยุกค์ใช้งานในด้านการสำรวจท่อแอร์ภายในอาคารเป็นหลัก จึงมีลักษณะเฉพาะบางประการที่ต้องสอดคล้องกับสภาพการทำงานในการสำรวจท่อแอร์ เช่น ขนาดของท่อแอร์ หรือ ระบบขับเคลื่อนที่ต้องการความคล่องตัวสูง ดังนั้น หลักการในการออกแบบหุ่นยนต์ในวิทยานิพนธ์ฉบับนี้จึงสามารถนำไปประยุกค์ใช้กับการออกแบบหุ่นยนต์สำหรับทำงานในด้านอื่นๆ ได้เช่นกัน

6. การทดสอบการทำงานของหุ่นยนต์สำรวจท่อแอร์ในวิทยานิพนธ์ฉบับนี้ ได้ทำการทดสอบกับท่อแอร์ของระบบปรับอากาศจำลองที่สร้างขึ้นเอง เพื่อทดสอบหลักการทำงานของหุ่นยนต์ โดยอาจมีลักษณะหรือคุณสมบัติบางประการที่ไม่เหมือนกับท่อแอร์ของระบบปรับอากาศจริง หากต้องการพัฒนาหุ่นยนต์ให้สามารถใช้งานในท่อแอร์ของระบบปรับอากาศจริง ควรมีการทดสอบกับท่อแอร์ของระบบปรับอากาศจริง และทำการปรับปรุงระบบต่างๆ ให้มีความเหมาะสมก่อนนำไปใช้งาน

เอกสารและสิ่งอ้างอิง

นคร ภักดีชาติ และ ชัยวัฒน์ ลิ้มพรจิตรวิไล. ม.ป.ป. **คู่มือการทดลอง dsPIC Microcontroller เบื้องต้น ด้วยโปรแกรมภาษา C กับ MPLAB C30**. บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด, กรุงเทพฯ.

Autonomus System Lab of Eidgenössische Technische Hochschule Zürich. n.d. **Air-duct Inspection Robot**. Institute of Robotics and Intelligent Systems. Available Source: http://as1.epfl.ch/index.html?content=research/systems/Scrappy.php, May 10, 2008.

Bräunl, T. 2006. **Embededed Robotics.** 2 ed. Springer-Verlag, Germany.

Jones, J.L. 2004. **Robot Programming**. McGraw-Hill, New York.

Jones, J.L., A.M. Flynn and B.A. Seiger. 1999. **Mobile Robots**. 2nd ed. AK Peters, Massachusetts.

Triventek Inc. n.d. **Jetvent Inspector**. Duct Cleaning Equipment. Available Source: http://www.triventek-ductcleaning.com/, June 10, 2008.

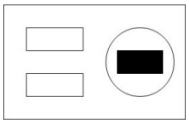
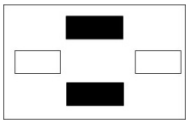
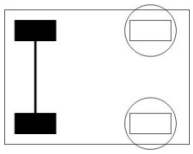
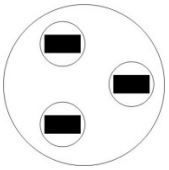
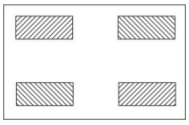
Wang, Y. and J. Zhang. 2006. Autonomous Air Duct Cleaning Robot System. **Ciruitsand and Systems** (2): 510-513.

ภาคผนวก

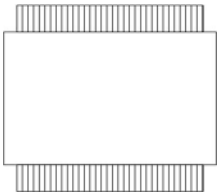
ภาคผนวก ก
ระบบขับเคลื่อนของหุ่นยนต์

ระบบขับเคลื่อนของหุ่นยนต์

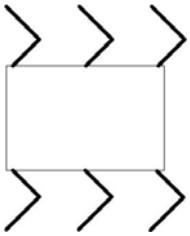
ตารางผนวกที่ ก1 ระบบขับเคลื่อนแบบใช้ล้อขับเคลื่อน

ชื่อระบบขับเคลื่อน	คำอธิบาย
Single Wheel Drive 	เป็นระบบขับเคลื่อนที่ใช้ล้อขับเพียงล้อเดียว การเลี้ยวทำได้โดยการหมุนล้อขับไปในทิศทางที่ต้องการ ทำให้มีความคล่องตัวสูง
Differential Drive 	เป็นระบบขับเคลื่อนที่มีล้อขับ 2 ล้อ การควบคุมการเคลื่อนที่ในทิศทางต่างๆทำได้ด้วยการควบคุมความเร็ว และทิศทางการหมุนของมอเตอร์ของล้อขับทั้งสอง
Ackermann Steering 	เป็นระบบการเคลื่อนที่เหมือนกับระบบของรถยนต์ โดยจะใช้ล้อคู่หลังในการขับเคลื่อน และควบคุมทิศทางที่ล้อคู่หน้า ทำให้ระบบนี้เลี้ยวได้ง่ายคล่องตัว แต่รัศมีการเลี้ยวค้อนข้างกว้าง
Synchro Drive 	เป็นระบบขับเคลื่อนที่คล้ายกับ Single drive โดยที่ล้อทุกล้อจะใช้มอเตอร์ขับตัวเดียวกัน และการควบคุมทิศทางของแต่ละล้อก็จะใช้มอเตอร์ตัวเดียวกัน ถ้าต้องการให้หุ่นยนต์ไปในทิศทางใด ก็ให้หันล้อทุกล้อไปในทิศทางนั้น
Omni-Directional Drive 	เป็นระบบขับเคลื่อนที่จะต้องใช้ล้อแบบพิเศษที่เรียกว่า Mecanum Wheel สามารถเคลื่อนที่ได้หลายรูปแบบ และมีคล่องตัวในการเคลื่อนที่และการเลี้ยวสูง แต่ต้นทุนในการสร้างสูง ใช้มอเตอร์ขับเท่าจำนวนล้อที่ใช้

ตารางผนวกที่ ก2 ระบบขับเคลื่อนแบบสายพานขับเคลื่อน

ชื่อระบบขับเคลื่อน	คำอธิบาย
ระบบสายพานขับเคลื่อน (Tracked Drive) 	เป็นระบบขับเคลื่อนที่มีลักษณะคล้ายกับแบบ differential drive การควบคุมการเคลื่อนที่ทำได้โดยการควบคุมความเร็วและทิศทางการหมุนของมอเตอร์แต่ละด้าน ทำให้สามารถเลี้ยวมุมแคบได้ดี ความสามารถในการเคลื่อนที่ได้ในทุกๆ พื้นผิว มีแรงขับเคลื่อนดี แต่มีข้อเสีย คือ ในขณะที่เลี้ยวจะการสูญเสียพลังงานมาก

ตารางผนวกที่ ก3 ระบบขับเคลื่อนแบบใช้ขาขับเคลื่อน

ชื่อระบบขับเคลื่อน	คำอธิบาย
Walking Robot 	เป็นระบบขับเคลื่อนที่เลียนแบบการเคลื่อนที่ของมนุษย์ หรือสัตว์จำพวก ปู แมงมุม สุนัข เป็นต้น เนื่องจากใช้ขาในการเคลื่อนที่ทำให้สามารถขึ้นบันได ลงทางชั้น ข้ามเครื่องกีดขวาง ฯลฯ แต่การเคลื่อนที่ในระบบนี้จะมีความยุ่งยากซับซ้อนในการออกแบบการสร้าง รวมไปถึงการควบคุม ทั้งยังมีต้นทุนสูง เนื่องจากใช้มอเตอร์ขับมาก

ภาคผนวก ข

ตัวอย่างการเขียนโปรแกรมอุปกรณ์ต่างๆ

ตัวอย่างการเขียนโปรแกรมอุปกรณ์ต่างๆ

ตัวอย่างที่ 1 การเขียนโปรแกรม Digital IO

```
int main (void)
{
TRISFbits.TRISF2 = 0;                   // Configuration for RF2 to output
TRISFbits.TRISF3 = 1;                   // Configuration for RF3 to input
While(1)
        {
if(!PORTEbits.RF3)
PORTEbits.RF2 = 1;
if(PORTEbits.RF3)
            PORTEbits.RF2 = 0;
}
}
```

ตัวอย่างที่ 2 การเขียนโปรแกรม Analog Input

```
void init_adc()
{
unsigned int Channel, PinConfig, Scanselect, Adcon3_reg, Adcon2_reg,Adcon1_reg;
ADCON1bits.ADON = 0;      // Turn off ADC
Channel   =   ADC_CH0_POS_SAMPLEA_AN0 &        // Channel 0 positive input select AN0
ADC_CH0_POS_SAMPLEA_AN1 &               // Channel 0 positive input select AN1
ADC_CH0_POS_SAMPLEA_AN2 &               // Channel 0 positive input select AN2
ADC_CH0_POS_SAMPLEA_AN3 &               // Channel 0 positive input select AN3
ADC_CH0_NEG_SAMPLEA_NVREF ;             // Channel 0 negative VREF
SetChanADC10(Channel);                  // Set channel configuration
ConfigIntADC10(ADC_INT_DISABLE);        // Disable interrupt for ADC
```

```
PinConfig = ENABLE_AN0_ANA & ENABLE_AN1_ANA & ENABLE_AN2_ANA &
ENABLE_AN3_ANA ;                                    // Enable AN0-AN3 analog
portScanselect = SKIP_SCAN_AN4 & SKIP_SCAN_AN5 & SKIP_SCAN_AN6 &
SKIP_SCAN_AN7;                                      // Scan for AN0-AN3
Adcon3_reg = ADC_SAMPLE_TIME_10 &                   // Sample for 10 time
ADC_CONV_CLK_INTERNAL_RC &                          // Internal Clock
            ADC_CONV_CLK_13Tcy;
Adcon2_reg = ADC_VREF_AVDD_AVSS &                   // Vref at Vdd and Vss
ADC_SCAN_ON &                                       // Enable scan for ADC
ADC_ALT_BUF_OFF &                                   // Disable alternate buffer
ADC_ALT_INPUT_OFF &                                 // Disable alternate input
ADC_CONVERT_CH0&                                    // Select CH0 convert
ADC_SAMPLES_PER_INT_16;                             // 16 sample between interrupt
Adcon1_reg = ADC_MODULE_ON &                        // Enable module ADC
ADC_IDLE_CONTINUE &                                 // ADC run on idle mode
ADC_FORMAT_INTG &                                   // Output value integer format
ADC_CLK_MANUAL &                                    // ADC manual clock
ADC_SAMPLE_SIMULTANEOUS &                           // ADC sampling simultaneous
ADC_AUTO_SAMPLING_ON;                               // ADC auto sampling
OpenADC10(Adcon1_reg, Adcon2_reg,  Adcon3_reg,PinConfig, Scanselect);          //
Turn on ADC
}
int main(void)
{
init_adc();
while(1)
{
for(adc_count=0;adc_count<3;adc_count++)        // Loop Read ADC[0..3]
            {
            ADCON1bits.SAMP = 1;
```

```
            while(!ADCON1bits.SAMP);                    // Wait ADC Sampling Complete
            ConvertADC10();
             adc_buff[adc_count]= ReadADC10(adc_count);      // Save Result to Buffer
            }
        }
}
```

ตัวอย่างที่ 3 ฟังก์ชันหาค่าระยะทางจากค่า ADC

```
int distance(int z)
{
if(z<74)
return(1000);
if(z<147)
return(1201.2-(5.5*z));
if(z<215)
return(724.18-(2.23*z));
if(z<338)
return(420.18-(0.8*z));
if(z<563)
return(254.2-(0.315*z));
if(z<640)
return(217.34-(0.24*z));
if(z>=640)
return(0);
}
```

ตัวอย่างที่ 4 การเขียนโปรแกรมตรวจจับสัญญาณ PWM

```
void _ISR _IC1Interrupt(void)
{
IFS0bits.IC2IF = 0;                                 // Reset Capture1 Interrupt Flag
```

```
Capture_Interrupt_Count++;                          // Count Edge Detect
if (Capture_Interrupt_Count==1)                     // First Edge Detect
        ReadCapture2(&Capture_Value2_1);            // Save 1st Edge Capture Value
if (Capture_Interrupt_Count==2)                     // First Edge Detect
        ReadCapture2(&Capture_Value2_2);            // Save 2st Edge Capture Value
if (Capture_Interrupt_Count==3)                     //Second Edge Detect
{
        ReadCapture2(&Capture_Value2_3);            // Save 3nd Edge Capture Value
        Capture_Interrupt_Count=0;
}
if (Capture_Interrupt_Count==0)                     //Second Edge Detect
{
if(Capture_Value2_2 > Capture_Value2_1)
Period2_1 = Capture_Value2_2 - Capture_Value2_1;
if(Capture_Value2_1 > Capture_Value2_2)
Period2_1 = (65535-Capture_Value2_1)+(Capture_Value2_2+1);
if(Capture_Value2_3 > Capture_Value2_2)
Period2_2 = Capture_Value2_3 - Capture_Value2_2;
if(Capture_Value1_2 > Capture_Value1_3)
Period2_2 = (65535-Capture_Value2_2)+(Capture_Value2_3+1);
if(Period2_1>Period2_2)
{
Period2=Period2_1;
Period2_1=Period2_2;
Period2_2=Period2;
}
}
}
int main(void)
{
```

```
init_capture();
While(1)
{
        Period2_1= Period2_1/10;
Period2_2= Period2_2/10;
}
}
void init_capture(void)
{
CloseCapture1();                                    // Disable Capture1 Befor New Config
CloseTimer3();                                      // Disable Timer3 Befor New Config
ConfigIntCapture1(IC_INT_ON &                       // Enable Capture Interrupt
IC_INT_PRIOR_6);       // Capture Interrupt Priority = 6
OpenTimer3(T2_ON &                                  // ON Timer3
T2_IDLE_STOP &         // Disable Timer2 in IDLE Mode
T2_32BIT_MODE_OFF &          // Timer3 = 16 Bit Timer
T2_GATE_OFF &                // Disable Timer2 Gate Control
T2_PS_1_1 &            // Timer3 Prescale = 1:1
T2_SOURCE_INT ,        // Timer3 Clock Source = Internal
65535);                      // Timer3 Match Value
WriteTimer3(0);                                     // Reset Timer2 Count
OpenCapture1(IC_IDLE_STOP &                         // Disable Capture1 in IDLE Mode
IC_TIMER3_SRC &              // Used Timer3 = Source Clock Capture1
IC_INT_1CAPTURE &             // 1 Capture / Interrupt
IC_EVERY_EDGE);               // Capture1 Start on Rising Edge
}
```

ตัวอย่างที่ 5 โปรแกรมสร้างสัญญาณ PWM สำหรับควบคุม RC Servo Motor

```
void _ISR _T1Interrupt(void)                    // pwm for servo motor 3 ch  timer1
{
IFS0bits.T1IF = 0;                          // Clear Timer interrupt flag
lup=lup+1;
if(lup<=position1)
PORTFbits.RF2 = 1;              // turn on  RF2   //Pwm CH1
if(lup>position1)
PORTFbits.RF2 = 0;              // turn off RF2     //Pwm CH1
if(lup == 2000)                                 //2000  Pwm 20ms macth_value cpu
lup=1;
}
int main(void)
{
unsigned int match_value;                        // for time loop
match_value = 146;                              // set time
ConfigIntTimer1(T1_INT_PRIOR_5 &            // Timer1 interrupt priority 1
        T1_INT_ON);                 // Enable interrupt for timer1
WriteTimer1(0);                                 // Clear count value at TMR1 register
OpenTimer1(     T1_ON &                         // Start timer1
T1_GATE_OFF &                   // Disable gate pin for timer1
T1_IDLE_STOP &                  // Stop timer in idle mode
T1_PS_1_1 &             // Presceler 1:1
T1_SYNC_EXT_OFF &           // Disable sync external source
T1_SOURCE_INT, match_value);// Wait till the timer matches with the period value
while(1)// Loop Continue
{
position1=150;
}
}
```

ตัวอย่างที่ 6 โปรแกรมควบคุมการทำงานของ ESC MC330CR

```
void _ISR _T1Interrupt(void)                    // pwm for servo mortor 3 ch  timer1
{
IFS0bits.T1IF = 0;                          // Clear Timer interrupt flag
  lup=lup+1;
  if(lup<=position1)
     PORTFbits.RF2 = 1;                             // turn on  RF2   //Pwm CH1
  if(lup>position1)
     PORTFbits.RF2 = 0;                             // turn off RF2     //Pwm CH1
  if(lup == 2000)                           //2000  Pwm 20ms macth_value cpu
     lup=1;
}
void delay(unsigned long int count1)
{
  while(count1 > 0) {count1--;}                     // Loop Decrease Counter
}

void esc_control(unsigned int m,unsigned int s)     //ESC futaba contraller
{
if(m==1)
    {
if(s<150)
{position1=s;   mc1=1;}
if(s==150)
{position1=150;   mc1=1;}
if(s>150)
{
                if(mc1==1)
                {
```

```
                position1=145;          delay(40000);
                position1=160;          delay(40000);
                position1=150;          delay(40000);
                mc1=2;
            }
if(mc1==2)
position1=s;
}
  }
}

int main(void)
{
 unsigned int match_value;                      // for time loop
 match_value = 146;                             // set time
  ConfigIntTimer1(T1_INT_PRIOR_5 &                      // Timer1 interrupt priority 1
            T1_INT_ON);                 // Enable interrupt for timer1
  WriteTimer1(0);                               // Clear count value at TMR1 register
  OpenTimer1(T1_ON &                            // Start timer1
       T1_GATE_OFF &                            // Disable gate pin for timer1
        T1_IDLE_STOP &                          // Stop timer in idle mode
       T1_PS_1_1 &                      // Presceler 1:1
       T1_SYNC_EXT_OFF &                        // Disable sync external source
       T1_SOURCE_INT, match_value);             // Wait till the timer matches with the period
value
while(1)// Loop Continue
{
esc_control(1,150);
delay(700000);
esc_control(1,100);
```

```
delay(7000);
esc_control(1,200);
delay(700000);
        }
}
```

ภาคผนวก ค

Code โปรแกรมของหุ่นยนต์สำรวจท่อแอร์

Code โปรแกรมของหุ่นยนต์สำรวจท่อแอร์

```
/*********************************************************/
/* Project     : Air-duct Surveying Robot                */
/* By          : Kanjana Chawbankor                      */
/* Target MCU     : dsPIC30F2010                         */
/*             : X-TAL : 7.3728 MHz                      */
/*             : Run 58.9824 MHz                         */
/*             : Selec OSC Mode = XT w/PLL 8x            */
/* Compiler       : MPLAB + C30 V1.33                    */
/*********************************************************/
/* Used RE3 = Digital Input                              */
/* Used RB0..RB3 = ADC Input                             */
/* Used RD0,RD1 = PWM Input                              */
/* Used RE0..RE2 = Digital Output                        */
/* Used RF2,RF3,RE8 = PWM Output                         */
/* Display Result to UART1 : 4800,N,8,1                  */
/*********************************************************/
#include "p30f2010.h"           // For dsPIC30F2010 MPU Register
#include "uart.h"               // Used UART Library Function
#include "adc10.h"              // Used ADC Library Function
#include "stdio.h"              // Used "sprintf" Function
#include "timer.h"              // Module function for Timer
#include "incap.h"              // Used Input Capture Library Function
/* Setup Configuration For ET-dsPIC30F2010 */
_FOSC(CSW_FSCM_ON & XT_PLL8);       // Enable Clock Switching,Enable Fail-Salf
Clock
                                    //Closk Source = Primary XT + (PLL x 8)
_FWDT(WDT_OFF);                     // Disable Watchdog
```

```
_FBORPOR(PBOR_ON & BORV_45 & PWRT_64 & MCLR_EN);          //Enable Brown-Out
= 4.5V,
_FGS(CODE_PROT_OFF);                                      //Code Protect OFF
/* End   Configuration For ET-dsPIC30F2010 */
unsigned char uart_buf[40];                    // "sprintf" Buffer
unsigned int adc_buff[4],adc_buff1[4],adc_buff2[4],adc_buff3[4],adc_buff4[4],adc_buff5[4];// 4
Channel ADC
unsigned int adc_count;
unsigned int timer_value;
unsigned int lup,mode,n;        //for counter timer
unsigned int position1,positionset,position2,position3;     //servo 1ms-2ms(0-7373)
unsigned int speed,lr_position;
unsigned int Capture_Interrupt_Count1,Capture_Interrupt_Count2;                    // Capture
Edge Count
unsigned int Capture_Value1_1,Capture_Value1_2,Capture_Value1_3;                   // 1st
Capture Value Save
unsigned int Capture_Value2_1,Capture_Value2_2,Capture_Value2_3;                   // 2nd
Capture Value Save
unsigned int Period1_1,Period1_2;                  // Period Value
unsigned int Period2_1,Period2_2;                  // Period Value
unsigned int mc1,mc2;
unsigned int kp_dir,sub_mode,speed_at;
unsigned int oldsub_mode;
unsigned int pe1_1,pe1_2,pe2_1,pe2_2;
/* pototype  section */
void init_uart(void);                   // Initial UART1 Function
void init_adc(void);                    // Initial ADC Function
void init_capture(void);
//------------------------------------ Interrupt service routine for timer1----------------------------------
--------//
```

```
void _ISR _T1Interrupt(void)                    // pwm for servo motor  3 ch  timer1
{
IFS0bits.T1IF = 0;                              // Clear Timer interrupt flag
lup=lup+1;
if(lup == 2000)        //2000  Pwm 20ms macth_value cpu
{lup=1;}
if(lup<=position1)
{PORTFbits.RF2 = 1;}    // turn on  RF2   //Pwm CH1
if(lup>position1)
{PORTFbits.RF2 = 0;}    // turn off RF2    //Pwm CH1
if(lup<=position2)
{PORTFbits.RF3 = 1;}    // turn on  RF3   //Pwm CH2
if(lup>position2)
{PORTFbits.RF3 = 0;}    // turn off RF3   //Pwm CH2
if(lup<=position3)
{ PORTEbits.RE8 = 1;}  // turn on RE8     //Pwm CH3
if(lup>position3)
{ PORTEbits.RE8 = 0;}  // turn off RE8   //Pwm CH3
}
/* Capture1 Interrupt Service */
void _ISR _IC1Interrupt(void)
{
IFS0bits.IC1IF = 0;                     // Reset Capture1 Interrupt Flag
Capture_Interrupt_Count1++;                     // Count Edge Detect
if (Capture_Interrupt_Count1==1)                // First Edge Detect
ReadCapture1(&Capture_Value1_1);        // Save 1st Edge Capture Value
if (Capture_Interrupt_Count1==2)                // First Edge Detect
ReadCapture1(&Capture_Value1_2);        // Save 1st Edge Capture Value
if (Capture_Interrupt_Count1==3)                // Second Edge Detect
```

```
{ReadCapture1(&Capture_Value1_3); Capture_Interrupt_Count1=0; }        // Save 2nd Edge
Capture Value
if (Capture_Interrupt_Count1==0)                    // Second Edge Detect
{
if (Capture_Value1_2 > Capture_Value1_1)
Period1_1 = Capture_Value1_2 - Capture_Value1_1;
if (Capture_Value1_1 > Capture_Value1_2)
Period1_1 = (65535-Capture_Value1_1)+(Capture_Value1_2+1);
if (Capture_Value1_3 > Capture_Value1_2)
Period1_2 = Capture_Value1_3 - Capture_Value1_2;
If (Capture_Value1_2 > Capture_Value1_3)
Period1_2 = (65535-Capture_Value1_2)+(Capture_Value1_3+1);
If (Period1_1>Period1_2)
Period1_1=Period1_2;
Period1_1=Period1_1/10;
}
}
//for IC2 capture 2
void _ISR _IC2Interrupt(void)
{
IFS0bits.IC2IF = 0;                                 // Reset Capture1 Interrupt Flag
Capture_Interrupt_Count2++;                         // Count Edge Detect
if (Capture_Interrupt_Count2==1)                    // First Edge Detect
ReadCapture2(&Capture_Value2_1);       // Save 1st Edge Capture Value
if (Capture_Interrupt_Count2==2)                    // First Edge Detect
ReadCapture2(&Capture_Value2_2);       // Save 1st Edge Capture Value
if (Capture_Interrupt_Count2==3)                    // Second Edge Detect
{
ReadCapture2(&Capture_Value2_3); Capture_Interrupt_Count2=0;}          // Save 2nd Edge
Capture Value
```

```
if (Capture_Interrupt_Count2==0)                    // Second Edge Detect
{
if (Capture_Value2_2 > Capture_Value2_1)
Period2_1 = Capture_Value2_2 - Capture_Value2_1;
if (Capture_Value2_1 > Capture_Value2_2)
Period2_1 = (65535-Capture_Value2_1)+(Capture_Value2_2+1);
if (Capture_Value2_3 > Capture_Value2_2)
Period2_2 = Capture_Value2_3 - Capture_Value2_2;
if (Capture_Value1_2 > Capture_Value1_3)
Period2_2 = (65535-Capture_Value2_2)+(Capture_Value2_3+1);
if (Period2_1>Period2_2)
Period2_1=Period2_2;
Period2_1=Period2_1/10;
}
}
/***********************/
/* Delay Time Function */
/*    1-4294967296     */
/***********************/
void delay(unsigned long int count1)
{while(count1 > 0) {count1--;}}                     // Loop Decrease Counter
void esc_control(unsigned int m,unsigned int s)     //ESC futaba contraller
{
if (m==1)
{
if (s<150)
{position1=s;   mc1=1;}
if (s==150)
{position1=150;   mc1=1;}
if (s>150)
```

```
{
if (mc1==1)
{
position1=145;     delay(40000);
position1=160;     delay(40000);
position1=150;     delay(40000);
mc1=2;
}
if (mc1==2)
position1=s;
}
}
if (m==2)
{
if (s<150)
{position2=s;   mc2=1;}
if (s==150)
{position2=150;              mc2=1;}
if (s>150)
{
if (mc2==1)
{
position2=146; delay(40000);
position2=160;     delay(40000);
position2=150;     delay(40000);
mc2=2;
}
if (mc2==2)
position2=s;
}
```

```
}
}
//Futaba ESC controller
void move_robot(unsigned int sp,unsigned int lr)
{
if (lr==150)
{
if (sp==150)
{esc_control(1,150);   esc_control(2,150);   position3=150;}
if (sp<150)
{esc_control(1,sp);         esc_control(2,sp);          position3=150;}
if (sp>150)
{esc_control(1,sp);         esc_control(2,sp);          position3=150;}
}
if (lr!=150)
{
if (sp>150)
{
if (lr>150)
{
if (lr<175)
{esc_control(1,sp);         esc_control(2,150);         position3=lr;}
if (lr>=175)
{esc_control(1,sp);         esc_control(2,150-(sp-150));        position3=150;}
}
if (lr<150)
{
if (lr>125)
{esc_control(1,150);        esc_control(2,sp);          position3=lr;}
if (lr<=125)
```

```
{esc_control(1,150-(sp-150));      esc_control(2,sp);          position3=150;}
}
}
if (sp<150)
{
if (lr>150)
{
if (lr<175)
{esc_control(2,sp);         esc_control(1,150);         position3=lr;}
if (lr>=175)
{esc_control(2,sp);         esc_control(1,150-(sp-150));       position3=150;}
}
if (lr<150)
{
if (lr>125)
{esc_control(2,150);        esc_control(1,sp);          position3=lr;}
if (lr<=125)
{esc_control(2,150-(sp-150));      esc_control(1,sp);          position3=150;}
}
}
}
}
//Calculate distance
int distance(int z)
{
        if (z<74)
            return(1000);
        if (z<147)
            return(1201.2-(5.5*z));
        if (z<215)
```

```
            return(724.18-(2.23*z));
        if (z<338)
            return(420.18-(0.8*z));
        if (z<563)
            return(254.2-(0.315*z));
        if (z<640)
            return(217.34-(0.24*z));
        if (z>=640)
            return(0);
}
//Main loop
int main(void)
{
unsigned int match_value;                   // for time loop
match_value = 146;                  // set time
init_uart();                        // Initial UART = 9600,N,8,1
init_adc();                         // Initial ADC[0..3]
init_capture();
//Set IO
TRISFbits.TRISF2 = 0;                       // Configuration for RF2 to output
TRISFbits.TRISF3 = 0;               // Configuration for RF3 to output
TRISEbits.TRISE8 = 0;               // Configuration for RE8 to output
TRISEbits.TRISE0 = 0;                       // Configuration for RE0 to output
TRISEbits.TRISE1 = 0;                       // Configuration for RE1 to output
TRISEbits.TRISE2 = 0;               // Configuration for RE2 to output
TRISEbits.TRISE3 = 1;                       // Configuration for RE3 to mode input
//Set timer1 for servo
ConfigIntTimer1(T1_INT_PRIOR_5 &            // Timer1 interrupt priority 1
T1_INT_ON);         // Enable interrupt for timer1
WriteTimer1(0);                             // Clear count value at TMR1 register
```

```
OpenTimer1(T1_ON &                      // Start timer1
T1_GATE_OFF &                           // Disable gate pin for timer1
T1_IDLE_STOP &                          // Stop timer in idle mode
T1_PS_1_1 &                     // Presceler 1:1
T1_SYNC_EXT_OFF &                       // Disable sync external source
T1_SOURCE_INT, match_value);            // Wait till the timer matches with the period value
sub_mode=0;                             //constant sub mode
oldsub_mode=0;
move_robot(150,150);
sprintf(uart_buf,"start program\n\r\0 ");               // Print Message String
putsUART1((unsigned int *)uart_buf);                    // Print uart_buff to UART1
while(BusyUART1());                     // Wait putsUART1 Complete
while(1)// Loop Continue
{
        //Mode control
if (!PORTEbits.RE3)
                mode=1;
if (PORTEbits.RE3)
                mode=3;
//mode 1 for manual mode
if (mode==1)
{
PORTEbits.RE0 = 0;PORTEbits.RE1 = 0;PORTEbits.RE2 = 0;
if ((Period1_1>=2160)&&(Period1_1<=2250))
pe1_1=150;
if ((Period1_1<2160)&&(Period1_1>1660))
pe1_1=150+((2160-Period1_1)/10);
if ((Period1_1>2250)&&(Period1_1<2800))
pe1_1=150-((Period1_1-2250)/11);
if ((Period1_1<=2800)&&(Period1_1>=1660))
```

```
{
speed=pe1_1;
pe1_2=pe1_1;
}
if ((Period1_1>2800)||(Period1_1<1660))
speed=pe1_2;
if ((Period2_1>=2233)&&(Period2_1<=2273))
pe2_1=150;
if ((Period2_1<2233)&&(Period2_1>1683))
pe2_1=150-((2233-Period2_1)/11);
if ((Period2_1>2273)&&(Period2_1<2823))
pe2_1=150+((Period2_1-2273)/11);
if ((Period2_1<=2823)&&(Period2_1>=1683))
{
lr_position=pe2_1;
pe2_2=pe2_1;
}
if ((Period2_1>2823)||(Period2_1<1683))
lr_position=pe2_2;
move_robot(speed,lr_position);
delay(50);
//Show capture parameter
/*
sprintf(uart_buf,"Period1 = %5u",speed);              // Print Message String
putsUART1((unsigned int *)uart_buf);                  // Print uart_buff to UART1
while(BusyUART1());                                   // Wait putsUART1 Complete
sprintf(uart_buf,"Period2 = %5u \r\0",lr_position); // Print Message String
putsUART1((unsigned int *)uart_buf);                 // Print uart_buff to UART1
while(BusyUART1());
delay(800000);
```

```
*/
        }   //end Mode 1
//Mode 3 Auto mode
if (mode==3)
{
for (adc_count=0;adc_count<4;adc_count++)        // Loop Read ADC[0..3]
{
ADCON1bits.SAMP = 1;
while(!ADCON1bits.SAMP);                  // Wait ADC Sampling Complete
ConvertADC10();
adc_buff1[adc_count]= ReadADC10(adc_count);   // Save Result to Buffer
}
for (adc_count=0;adc_count<4;adc_count++)        // Loop Read ADC[0..3]
{
ADCON1bits.SAMP = 1;
while(!ADCON1bits.SAMP);                  // Wait ADC Sampling Complete
ConvertADC10();
adc_buff2[adc_count]= ReadADC10(adc_count);  // Save Result to Buffer
}
for (adc_count=0;adc_count<4;adc_count++)        // Loop Read ADC[0..3]
{
ADCON1bits.SAMP = 1;
while(!ADCON1bits.SAMP);                  // Wait ADC Sampling Complete
ConvertADC10();
adc_buff3[adc_count]= ReadADC10(adc_count);  // Save Result to Buffer
}
for (adc_count=0;adc_count<4;adc_count++)        // Loop Read ADC[0..3]
{
ADCON1bits.SAMP = 1;
while(!ADCON1bits.SAMP);                  // Wait ADC Sampling Complete
```

```
ConvertADC10();
adc_buff4[adc_count]= ReadADC10(adc_count);  // Save Result to Buffer
}
for (adc_count=0;adc_count<4;adc_count++)        // Loop Read ADC[0..3]
{
ADCON1bits.SAMP = 1;
while(!ADCON1bits.SAMP);                  // Wait ADC Sampling Complete
ConvertADC10();
adc_buff5[adc_count]= ReadADC10(adc_count);        // Save Result to Buffer
}
adc_buff[0]=(distance((adc_buff1[0]+adc_buff2[0]+adc_buff3[0]+adc_buff4[0]+adc_buff5[0])/5
))/10;
adc_buff[1]=(distance((adc_buff1[1]+adc_buff2[1]+adc_buff3[1]+adc_buff4[1]+adc_buff5[1])/5
))/10;
adc_buff[2]=(distance((adc_buff1[2]+adc_buff2[2]+adc_buff3[2]+adc_buff4[2]+adc_buff5[2])/5
))/10;
adc_buff[3]=(adc_buff1[3]+adc_buff2[3]+adc_buff3[3]+adc_buff4[3]+adc_buff5[3])/100;
speed_at=150+adc_buff[3];
if (sub_mode==0)                    //standby mode
{
PORTEbits.RE0 = 0; PORTEbits.RE1 = 0; PORTEbits.RE2 = 0;
if ((adc_buff[1])<60&&(adc_buff[2]<60)&&(adc_buff[0]>45))
sub_mode=1;
if (adc_buff[1]>=60||adc_buff[2]>=60)
{
move_robot(speed_at,150);
delay(10000);
if(adc_buff[2]>=60)
sub_mode=3;
if((adc_buff[0]>60)&&(adc_buff[1]>60)&&(adc_buff[2]<60))
```

```
sub_mode=4;
if((adc_buff[0]<70)&&(adc_buff[1]>60)&&(adc_buff[2]<60))
sub_mode=5;
}
if ((adc_buff[1])<60&&(adc_buff[2]<60)&&(adc_buff[0]<25))
sub_mode=2;
if ((adc_buff[0]<=10)||((adc_buff[0]>100)&&(adc_buff[1]>100)&&(adc_buff[2]>100)))
{move_robot(150,150); sub_mode=6;}

}
if (sub_mode==1)      //Forward mode
{
                PORTEbits.RE0 = 0; PORTEbits.RE1 = 1; PORTEbits.RE2 = 0;
if ((adc_buff[1]==adc_buff[2])&&(adc_buff[0]>35))
{move_robot(speed_at,150); delay(50);}
if ((adc_buff[1]>adc_buff[2])&&(adc_buff[0]>10))
{
kp_dir=adc_buff[1]-adc_buff[2];
if (kp_dir<=5)
{move_robot(speed_at,170); delay(50);}
if (kp_dir>5)
{move_robot(speed_at,200); delay(50);}
                        }
if ((adc_buff[1]<adc_buff[2])&&(adc_buff[0]>10))
{
kp_dir=adc_buff[2]-adc_buff[1];
if (kp_dir<=5)
{move_robot(speed_at,130); delay(50);}
if (kp_dir>5)
{move_robot(speed_at,100); delay(50);}
```

```
}
if ((adc_buff[1]>60)||(adc_buff[2]>60)||(adc_buff[0]<35))   //exit sub_mode 1
sub_mode=1;
if ((adc_buff[0]<=10)||((adc_buff[0]>100)&&(adc_buff[1]>100)&&(adc_buff[2]>100)))
{move_robot(150,150); sub_mode=6;}
}                          //exit sub_mode1
if ((sub_mode==3)||(sub_mode==4))         // left and wall left
{
PORTEbits.RE0 = 0; PORTEbits.RE1 = 0; PORTEbits.RE2 = 1;
if (adc_buff[2]==30)
{move_robot(speed_at,150); delay(200);}
if (adc_buff[2]>25)
{
kp_dir=adc_buff[2]-25;
if (kp_dir<=5)
{move_robot(speed_at,130); delay(5000);}
if (kp_dir>5)
{move_robot(speed_at,100); delay(2000);}
}
if (adc_buff[2]<25)
{
kp_dir=25-adc_buff[2];
if (kp_dir<=5)
{move_robot(speed_at,170); delay(5000);}

if (kp_dir>5)
{move_robot(speed_at,200); delay(5000);}
}
if ((adc_buff[1])<=50&&(adc_buff[2]<=50)&&(adc_buff[0]>70))
sub_mode=1;                              //exit sub_mode3
```

```
if ((adc_buff[0]<=10)||((adc_buff[0]>100)&&(adc_buff[1]>100)&&(adc_buff[2]>100)))
{move_robot(150,150); sub_mode=6;}
}
if (sub_mode==5)// right
{
PORTEbits.RE0 = 1; PORTEbits.RE1 = 0; PORTEbits.RE2 = 0;
if (adc_buff[1]<25)
{
kp_dir=25-adc_buff[1];
if (kp_dir<=5)
{move_robot(speed_at,150); delay(5000);}
if (kp_dir>5)
{move_robot(speed_at,150); delay(5000);}
}
if (adc_buff[1]==30)
{move_robot(speed_at,150);}
if (adc_buff[1]>25)
{
kp_dir=adc_buff[1]-25;
if (kp_dir<=5)
{move_robot(speed_at,170); delay(5000);}
if (kp_dir>5)
{move_robot(speed_at,200); delay(2000);}
}
if ((adc_buff[1])<=50&&(adc_buff[2]<=50)&&(adc_buff[0]>85))
sub_mode=1;                          //exit sub_mode5
if ((adc_buff[0]<=10)||((adc_buff[0]>100)&&(adc_buff[1]>100)&&(adc_buff[2]>100)))
{move_robot(150,150); sub_mode=6;}
}
        if (sub_mode==2)                    //กลับรถ
```

```
                {
PORTEbits.RE0 = 1; PORTEbits.RE1 = 1; PORTEbits.RE2 = 1;
if ((adc_buff[0]<70)&&(adc_buff[1]<=50)&&(adc_buff[2]<=50))
{move_robot(165,200); delay(100);}
else
sub_mode=1;      //exit mode2
}

          if (sub_mode!=oldsub_mode)    //show sub_mode when sub_mode change
                {
                sprintf(uart_buf,"sub mode %4d \r\n",sub_mode);           // 4 Digit Decimal[0..1023]
Display
                putsUART1((unsigned int *)uart_buf);                 // Print uart_buff to UART1
                while(BusyUART1());                        // Wait putsUART1 Complete
                oldsub_mode=sub_mode;
                }
          if (sub_mode==6)
                {
                sprintf(uart_buf,"Emergency Mode\n\r\0 ");      // Print Message String
putsUART1((unsigned int *)uart_buf);                 // Print uart_buff to UART1
while(BusyUART1());                        // Wait putsUART1 Complete
}
//show ADC in rs232 for check parameter
/*
//Display ADC[0..3] Result on UART1
sprintf(uart_buf,"\rADC[0..3] = ");              // Print Message String
putsUART1((unsigned int *)uart_buf);                 // Print uart_buff to UART1
while(BusyUART1());                        // Wait putsUART1 Complete
//Display Result ADC0
sprintf(uart_buf,"%4d : ",adc_buff[0]);             // 4 Digit Decimal[0..1023] Display
```

```
putsUART1((unsigned int *)uart_buf);                // Print uart_buff to UART1
while(BusyUART1());                         // Wait putsUART1 Complete
//Display Result ADC1
sprintf(uart_buf,"%4d : ",adc_buff[1]);          // 4 Digit Decimal[0..1023] Display
putsUART1((unsigned int *)uart_buf);                // Print uart_buff to UART1
while(BusyUART1());                         // Wait putsUART1 Complete
//Display Result ADC2
sprintf(uart_buf,"%4d : ",adc_buff[2]);          // 4 Digit Decimal[0..1023] Display
putsUART1((unsigned int *)uart_buf);                // Print uart_buff to UART1
while(BusyUART1());                         // Wait putsUART1 Complete
//Display Result ADC3
sprintf(uart_buf,"%4d \r\0",speed_at);          // 4 Digit Decimal[0..1023] Display
putsUART1((unsigned int *)uart_buf);                // Print uart_buff to UART1
while(BusyUART1());
//mode
sprintf(uart_buf,"sub mode %4d \r\n",sub_mode);          // 4 Digit Decimal[0..1023] Display
putsUART1((unsigned int *)uart_buf);                // Print uart_buff to UART1
while(BusyUART1());                         // Wait putsUART1 Complete
*/
}//End Mode 3
}//end while
}//end main

/* Initial UART for dsPIC30F2010  4800,N,8,1 */
void init_uart()
{
CloseUART1();                               // Disable UART1 Before New Config
//Config UART1 Interrupt Control
ConfigIntUART1(UART_RX_INT_DIS &                // Disable RX Interrupt
UART_RX_INT_PR2 &                   // RX Interrupt Priority = 2
```

```
UART_TX_INT_DIS &                  // Disable TX Interrupt
UART_TX_INT_PR3);                  // TX Interrupt Priority = 3
//Open UART1 = Mode,Status,Baudrate
OpenUART1(UART_EN   &                        // Enable UART(UART Mode)
UART_IDLE_STOP &                   // Disable UART in IDLE Mode
UART_ALTRX_ALTTX &                       // Select U1TX=RC13,U1RX=RC14
UART_DIS_WAKE &                    // Disable Wake-Up
UART_DIS_LOOPBACK &                      // Disable Loop Back
UART_DIS_ABAUD &                   // Disable Auto Baudrate
UART_NO_PAR_8BIT &                   // UART = 8 Bit, No Parity
UART_1STOPBIT,                     // UART = 1 Stop Bit
// Config UART1 Status
UART_INT_TX &                      // Select Interrupt After TX Complete
UART_TX_PIN_NORMAL &                     // Normal U1TX Mode
UART_TX_ENABLE &                   // Enable U1TX
UART_INT_RX_BUF_FUL &                    // Flasg Set After RX Complete
UART_ADR_DETECT_DIS &                        // Disable Check Address
UART_RX_OVERRUN_CLEAR,                   // Clear Overrun Flag
191);
}
/* Initial ADC for dsPIC30F2010 */
void init_adc()
{
unsigned int Channel, PinConfig, Scanselect, Adcon3_reg, Adcon2_reg,Adcon1_reg;
ADCON1bits.ADON = 0;       // Turn off ADC
Channel   =   ADC_CH0_POS_SAMPLEA_AN0 &   // Channel 0 positive input select AN0
ADC_CH0_POS_SAMPLEA_AN1 &   // Channel 0 positive input select AN1
ADC_CH0_POS_SAMPLEA_AN2 &   // Channel 0 positive input select AN2
ADC_CH0_POS_SAMPLEA_AN3 &   // Channel 0 positive input select AN3
ADC_CH0_NEG_SAMPLEA_NVREF ;   // Channel 0 negative VREF
```

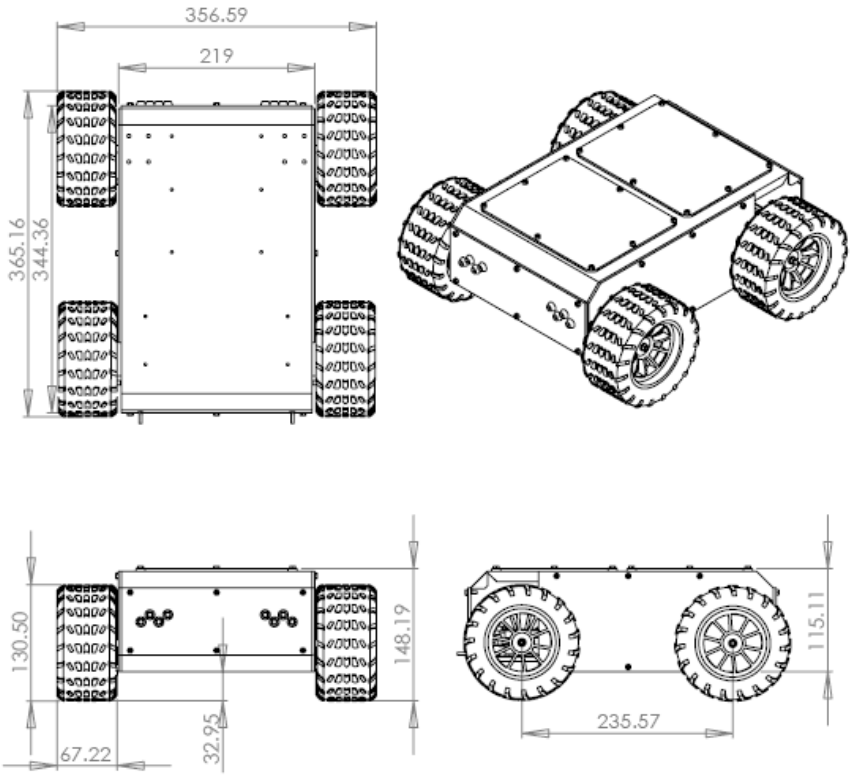
```
SetChanADC10(Channel);              // Set channel configuration
ConfigIntADC10(ADC_INT_DISABLE);        // Disable interrupt for ADC
PinConfig   =   ENABLE_AN0_ANA &     // Enable AN0-AN3 analog port
ENABLE_AN1_ANA &
ENABLE_AN2_ANA &
ENABLE_AN3_ANA ;
Scanselect   =   SKIP_SCAN_AN4 &        // Scan for AN0-AN3
SKIP_SCAN_AN5 &
SKIP_SCAN_AN6 &
SKIP_SCAN_AN7;
Adcon3_reg   =   ADC_SAMPLE_TIME_10 &        // Sample for 10 time
ADC_CONV_CLK_INTERNAL_RC &    // Internal Clock
ADC_CONV_CLK_13Tcy;
Adcon2_reg   =   ADC_VREF_AVDD_AVSS &     // Vref at Vdd and Vss
ADC_SCAN_ON &           // Enable scan for ADC
ADC_ALT_BUF_OFF &        // Disable alternate buffer
ADC_ALT_INPUT_OFF &      // Disable alternate input
ADC_CONVERT_CH0&        // Select CH0 convert
ADC_SAMPLES_PER_INT_16;   // 16 sample between interrupt
Adcon1_reg   =   ADC_MODULE_ON &        // Enable module ADC
ADC_IDLE_CONTINUE &      // ADC run on idle mode
ADC_FORMAT_INTG &        // Output value integer format
ADC_CLK_MANUAL &        // ADC manual clock
ADC_SAMPLE_SIMULTANEOUS &   // ADC sampling simultaneous
ADC_AUTO_SAMPLING_ON;   // ADC auto sampling
OpenADC10(Adcon1_reg, Adcon2_reg,   Adcon3_reg,PinConfig, Scanselect);    // Turn on ADC
module
}
/* Initial Timer for dsPIC30F2010 */
void init_capture(void)
```

```
{
CloseCapture1();                    // Disable Capture1 Befor New Config
CloseCapture2();                    // Disable Capture1 Befor New Config
CloseTimer3();                    // Disable Timer2 Befor New Config
// Config Capture1 Interrupt Control
ConfigIntCapture1(IC_INT_ON &              // Enable Capture Interrupt
IC_INT_PRIOR_6);          // Capture Interrupt Priority = 6
ConfigIntCapture2(IC_INT_ON &              // Enable Capture Interrupt
IC_INT_PRIOR_6);          // Capture Interrupt Priority = 6
ConfigIntTimer3(T3_INT_OFF &              // Disable Timer2 Interrupt
T3_INT_PRIOR_7);          // Timer2 Interrupt Priority = 7
OpenTimer3(T2_ON &                  // ON Timer2
T2_IDLE_STOP &                  // Disable Timer2 in IDLE Mode
T2_32BIT_MODE_OFF &             // Timer2 = 16 Bit Timer
T2_GATE_OFF &                   // Disable Timer2 Gate Control
T2_PS_1_1 &                   // Timer2 Prescale = 1:1
T2_SOURCE_INT ,                 // Timer2 Clock Source = Internal
65535);                   // Timer2 Match Value
WriteTimer2(0);                   // Reset Timer2 Count
OpenCapture1(IC_IDLE_STOP &              // Disable Capture1 in IDLE Mode
IC_TIMER3_SRC &                 // Used Timer2 = Source Clock Capture1
IC_INT_1CAPTURE &               // 1 Capture / Interrupt
IC_EVERY_EDGE);           // Capture1 Start on Rising Edge
OpenCapture2(IC_IDLE_STOP &              // Disable Capture1 in IDLE Mode
IC_TIMER3_SRC &                 // Used Timer2 = Source Clock Capture1
IC_INT_1CAPTURE &               // 1 Capture / Interrupt
IC_EVERY_EDGE);           // Capture1 Start on Rising Edge
}
```

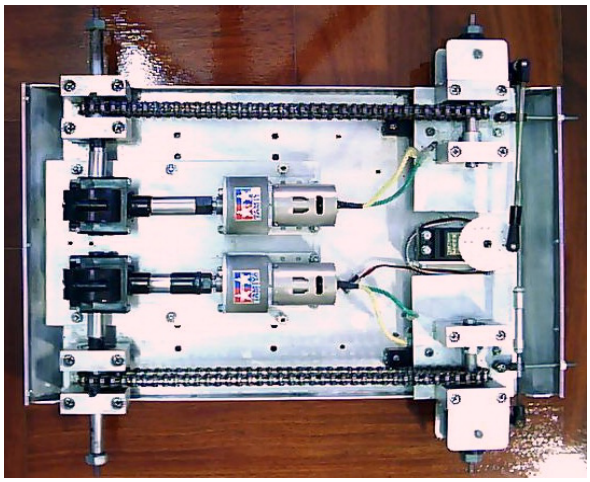
ภาคผนวก ง

ภาพหุ่นยนต์สำรวจท่อแอร์

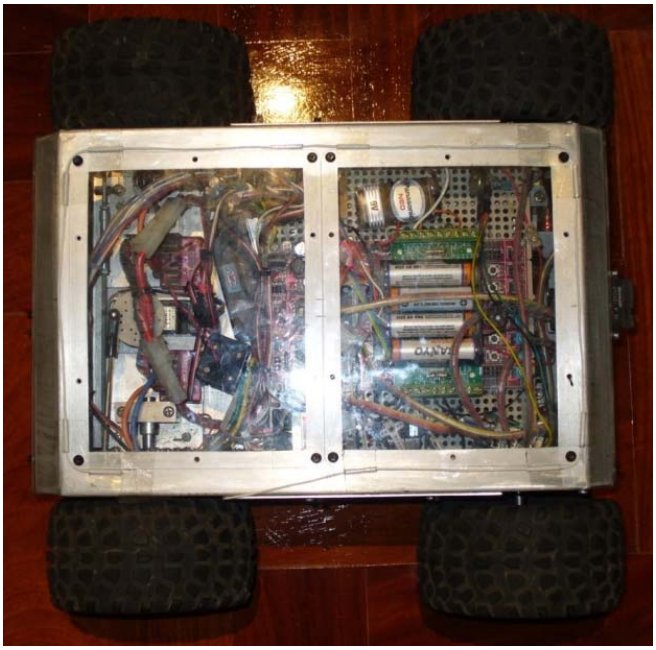
ภาพหุ่นยนต์สำรวจท่อแอร์



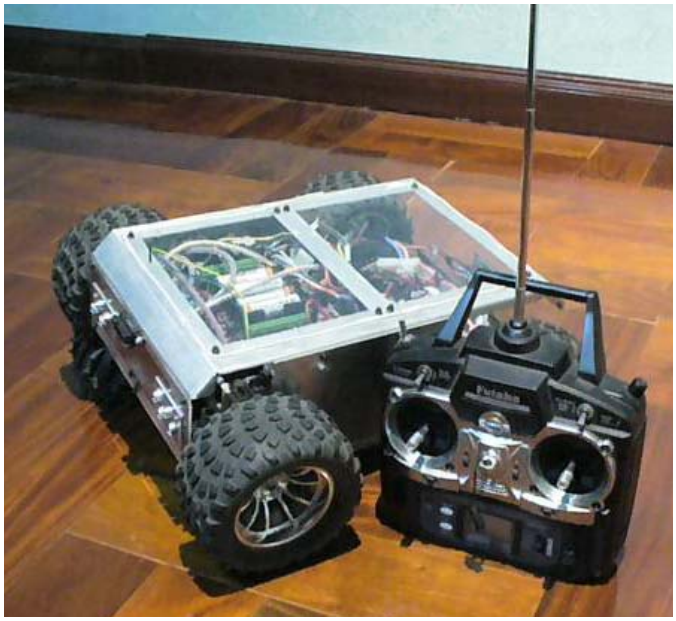
ภาพผนวกที่ ง1 ขนาดของหุ่นยนต์สำรวจท่อแอร์ที่ทำการออกแบบ



ภาพผนวกที่ ง2 ระบบส่งกำลังของหุ่นยนต์สำรวจท่อแอร์



ภาพผนวกที่ ง3 หุ่นยนต์สำรวจท่อแอร์



ภาพผนวกที่ ง4 หุ่นยนต์สำรวจท่อแอร์และวิทยุบังคับ

ประวัติการศึกษา และการทำงาน

ชื่อ –นามสกุล	นายกาญจนะ ชาวบ้านเกาะ
วัน เดือน ปี ที่เกิด	26 มกราคม พ.ศ. 2526
สถานที่เกิด	อำเภอเมือง จังหวัดสมุทรสาคร
ประวัติการศึกษา	วิศวกรรมศาสตรบัณฑิต (เครื่องกล) มหาวิทยาลัยเกษตรศาสตร์ (กำแพงแสน) พ.ศ. 2549
ตำแหน่งหน้าที่การงานปัจจุบัน	-
สถานที่ทำงานปัจจุบัน	-
ผลงานดีเด่นและรางวัลทางวิชาการ	-
ทุนการศึกษาที่ได้รับ	ทุนสนับสนุนคุณภาพงานวิจัยระดับบัณฑิตศึกษา ประจำปีงบประมาณ 2551 บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์