หุ่นยนต์ชาญฉลาด

(Intelligent Robot)

โดย

[illegible] รหัสประจำตัว B4500928

[illegible] รหัสประจำตัว B4511252

[illegible] รหัสประจำตัว B4513506

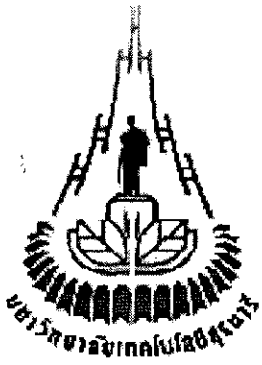
รายงานนี้เป็นส่วนหนึ่งของรายวิชา 427494 โครงงานศึกษาวิศวกรรมโทรคมนาคม

และ 427499 โครงงานวิศวกรรมโทรคมนาคม

ประจำภาคการศึกษาที่ 1 และ 3 ปีการศึกษา 2548

หลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมโทรคมนาคม หลักสูตรปรับปรุง พ.ศ. 2546

สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์



หุ่นยนต์สมองกล

(Intelligent Robot)

โดย

นางสาวชนิดา ทุ่มนอก รหัสประจำตัว B4509028

นางสาวนภาลัย วิชา รหัสประจำตัว B4511212

นางสาวสุรีย์พร มั่งมี รหัสประจำตัว B4513506

รายงานนี้เป็นส่วนหนึ่งของรายวิชา 427494 โครงการศึกษาวิศวกรรมโทรคมนาคม

และ 427499 โครงงานวิศวกรรมโทรคมนาคม

ประจำปีการศึกษาที่ 1 และ 3 ปีการศึกษา 2548

หลักสูตรวิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมโทรคมนาคม หลักสูตรปรับปรุง พ.ศ. 2541

สาขาวิชาวิศวกรรมโทรคมนาคม สำหรับวิชาวิศวกรรมศาสตร์

ใบรับรองโครงงานวิศวกรรมโทรคมนาคม

สาขาวิชาวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีสุรนารี

หัวข้อโครงงาน	หุ่นยนต์สมองกล	
นักศึกษา	นางสาวชนิดา ทุ่มนอก	B4509028
	นางสาวนภาลัย วิชา	B4511212
	นางสาวสุรีย์พร มั่งมี	B4513506
ปริญญา	วิศวกรรมศาสตร์บัณฑิต	
พ.ศ.	2548	
อาจารย์ที่ปรึกษาโครงงาน	อ.ดร.วิภาวี อุสาหะ	

กรรมการสอบ	ลงนาม
อ.ดร.วิภาวี อุสาหะ	
อ.ดร ชาญชัย ทองโสภา	

วันที่ 30 เมษายน 2549

สถานที่ ห้องปฏิบัติการโทรคมนาคม

หัวข้อโครงงาน	**หุ่นยนต์สมองกล**
นักศึกษา	**นางสาวชนิดา ทุ่มนอก**
	นางสาวนภาลัย วิชา
	นางสาวสุรีย์พร มั่งมี
ปริญญา	**วิศวกรรมศาสตร์บัณฑิต**
พ.ศ.	2548
อาจารย์ที่ปรึกษาโครงงาน	**อ.ดร.วิภาวี อุสาหะ**

บทคัดย่อ

สิ่งที่ยังคงเป็นปัญหาของหุ่นยนต์ในปัจจุบันนี้คือหุ่นยนต์ไม่สามารถเรียนรู้ได้ด้วยตนเอง ยังคงใช้การควบคุมโดยมนุษย์เป็นส่วนใหญ่ เช่น การควบคุมโดยใช้รีโมทคอนโทรล การป้อนคำสั่งให้ทำงานโดยผ่านทางคอมพิวเตอร์ ทางคณะผู้จัดทำได้เล็งเห็นถึงปัญหานี้จึงมีการทำหุ่นยนต์ต้นแบบที่สามารถเรียนรู้ที่จะหาเส้นทางได้ด้วยตนเองในสภาวะแวดล้อมที่เป็นกริดขนาด 10x10 และ 25X25 ที่มีสิ่งกีดขวางและไม่มีสิ่งกีดขวาง โดยเส้นทางที่จะไปถึงจุดมุ่งหมายนั้นต้องเป็นเส้นทางที่สั้นที่สุด โครงงานฉบับนี้ได้นำเสนอวิธีการเรียนรู้แบบรีอินเฟอร์สเมนท์ ในแบบออนโพลิซี มัลติ คาลโล (on-policy monte carlo หรือ ONMC) เพื่อประยุกต์ใช้ร่วมกับการหาเส้นทางเดินของหุ่นยนต์ ซึ่งวิธีนี้เป็นวิธีการหนึ่งของการเรียนรู้แบบรีอินเฟอร์สเมนท์ที่สามารถเรียนรู้วิธีการตัดสินใจหาเส้นทางที่สั้นที่สุดได้ ในสภาวะแวดล้อมที่มีการดำเนินไปของกระบวนการในลักษณะเป็นเอพพิโซด(episode) โดยเอพพิโซดหนึ่งสิ้นสุดลงเมื่อระบบมีการตรวจสอบพบตำแหน่งเป้าหมายที่เราต้องการโดยระบบที่ทำการศึกษาจะมีการตัดสินใจด้วยการนำผลเฉลี่ยระยะยาวของผลรางวัลสูงสุด ที่เกิดจากการตัดสินใจเลือกการกระทำเมื่ออยู่ในแต่ละสถานะมาเป็นเงื่อนไขในการตัดสินใจ และจะทำการปรับปรุงผลเฉลี่ยระยะยาวนี้ในทุกๆ เอพพิโซด (episode) เพื่อให้หุ่นสามารถตัดสินใจเลือกการกระทำที่ไปถึงจุดหมายโดยใช้เส้นทางที่สั้นที่สุด ซึ่งได้ทำการทดลอง แบ่งเป็นสองส่วนคือ การทดลองด้วยโปรแกรมจำลองแบบ(simulation)โดยมีการเปรียบเทียบผลกับกระบวนการคิดของมนุษย์ในการหาเส้นทางซึ่งขอบเขตการทดลองใช้กริดขนาด 25X25 ที่มีสิ่งกีดขวางในลักษณะที่มีสิ่งกีดขวางแตกต่างกัน และการทดลองภาคสนาม ใช้กริดขนาด 10X10 ที่มีสิ่งกีดขวาง ซึ่งผลการทดลองพบว่าวิธีการเรียนรู้แบบรีอินฟอร์ทเมนท์สามารถหาเส้นทางที่สั้นที่สุดได้ในสภาวะแวดล้อมที่มีสิ่งกีดขวาง การทำหุ่นยนต์ต้นแบบในโครงการนี้อาจเป็นแนวทางเพื่อนำไปพัฒนาเทคโนโลยีทางหุ่นยนต์ ให้สามารถทำงานแทนมนุษย์ได้อีกทั้งยังช่วยแบ่งเบาภาระของมนุษย์ในด้านที่เกี่ยวข้องกับชีวิตประจำวัน อาทิเช่น หุ่นยนต์ส่งเอกสาร เครื่องดูดฝุ่น เป็นต้น

กิตติกรรมประกาศ

คุณงามความดีอันใดที่เกิดจากโครงงานฉบับนี้ ขอมอบแด่บิดา มารดาของข้าพเจ้าที่คอยห่วงใย ให้โอกาส ให้กำลังใจ และให้การสนับสนุนทางการศึกษามาโดยตลอด

โครงงานเล่มนี้สามารถสำเร็จลุล่วงไปได้ด้วยดี เนื่องจากได้รับความกรุณาจากอาจารย์ที่ปรึกษา อ.ดร.วิภาวี อุตสาหะ ผู้ที่เป็นเจ้าของแนวคิดเริ่มแรกของหุ่นยนต์สมองกลที่สามารถหาเส้นทางเดินดีที่สุดได้เอง ที่ให้ความช่วยเหลือทางด้านแนวคิด การดูแลเอาใจใส่ติดตามงาน ชี้แนะข้อบกพร่องที่ข้าพเจ้ามองข้ามตลอดจนฝึกฝนและสนับสนุนข้าพเจ้าให้มีความสามารถในการทำโครงงานจนสามารถนำเสนอผลงานให้เป็นที่รู้จักและยอมรับได้

ขอขอบคุณผู้ที่เกี่ยวข้องอื่นๆดังนี้

ขอขอบคุณ คุณประพล จาระตะคุ วิศวกรประจำอาคารเครื่องมือ 3 ที่ช่วยเป็นธุระในการสั่งซื้ออุปกรณ์ คุณฉัตรชัย ถาจอหอ เจ้าหน้าที่ดูแลอุปกรณ์ห้องปฏิบัติการโทรคมนาคมที่ช่วยอำนวยความสะดวกในการเบิกอุปกรณ์ พี่ๆนักศึกษาปริญญาโท วิศวกรรมโทรคมนาคมทุกคนที่ให้การสนับสนุน และท้ายที่สุดเพื่อนนักศึกษาสาขาวิชาโทรคมนาคมทุกคนที่เป็นกำลังใจให้มาโดยตลอด

นางสาวชนิดา ทุ่มนอก
นางสาวนภาลัย วิชา
นางสาวสุรีย์พร มั่งมี

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

หน้า

สารบัญรูป

หน้า

สารบัญรูป (ต่อ)

หน้า

บทที่ 1
บทนำ

1.1 กล่าวนำ

ปัจจุบันเทคโนโลยีของหุ่นยนต์นั้นได้เจริญก้าวหน้าไปมาก ต่างจากเมื่อก่อนที่หุ่นยนต์จะถูกนำไปใช้ในงานอุตสาหกรรมเป็นส่วนใหญ่ แต่ปัจจุบันมีการนำมาใช้มากขึ้นไม่ว่าจะเป็นหุ่นยนต์ที่ใช้ในทางการแพทย์ หุ่นยนต์สำหรับงานสำรวจ หุ่นยนต์ที่ใช้งานในอวกาศ หรือแม้แต่หุ่นยนต์ที่ถูกสร้างขึ้นเพื่อเป็นเครื่องเล่นของมนุษย์ จนกระทั่งในปัจจุบันนี้ได้มีการพัฒนาให้หุ่นยนต์นั้นสามารถทำงานแทนมนุษย์และอาศัยอยู่ร่วมกันกับมนุษย์ได้ ซึ่งวัตถุประสงค์ของการนำหุ่นยนต์มาใช้นั้นก็เพื่อเพิ่มศักยภาพการทำงานและอำนวยความสะดวกให้แก่มนุษย์ ทางคณะผู้จัดทำได้เล็งเห็นความสำคัญของการทำงานร่วมกันระหว่างมนุษย์และหุ่นยนต์ จึงสร้างหุ่นยนต์ต้นแบบที่สามารถเรียนรู้และหาเส้นทางที่สั้นที่สุดได้ด้วยตนเอง

ดังนั้นโครงการนี้จึงได้ทำการศึกษาเกี่ยวกับหุ่นยนต์สมองกล โดยใช้ไมโครคอนโทรลเลอร์เป็นตัวควบคุมการเคลื่อนไหวของหุ่นยนต์ และใช้ชิพสมองกลซึ่งใช้หลักการการเรียนรู้แบบ ออนโพลิซี มัลติ คาลโล เป็นกลไกหลักที่ทำให้หุ่นยนต์สามารถเกิดการเรียนรู้และหาเส้นทางที่สั้นที่สุดได้ด้วยตนเอง ทำให้หุ่นยนต์มีอิสระในการเคลื่อนไหวโดยใช้ล้อในการขับเคลื่อน

จากโครงการนี้เราสามารถนำตัวหุ่นยนต์สมองกลไปเป็นพื้นฐานในการพัฒนา เพื่อสร้างนวัตกรรมใหม่ๆ ให้เกิดขึ้นในอนาคต เช่น หุ่นยนต์รับส่งเอกสารภายในสำนักงานหรือ การสร้างรถไฟฟ้าโดยไม่มีคนขับ เป็นต้น ทั้งนี้หุ่นยนต์สมองกลจะช่วยอำนวยความสะดวกให้กับมนุษย์มากขึ้น

1.2 <u>วัตถุประสงค์</u>

1. สามารถนำความรู้ที่ได้จากการศึกษาภาคทฤษฎีของวิชาไมโครคอนโทรลเลอร์ (microcontroller), ระบบควบคุม (control system), วงจรไฟฟ้า (electric circuit) ที่ได้ศึกษามาปฏิบัติและประยุกต์เพื่อใช้สร้างชิ้นงานออกมาและสามารถนำไปใช้งานจริงได้
2. ศึกษาและปรับปรุงระบบควบคุมของหุ่นยนต์ให้มีการเรียนรู้ได้ด้วยตัวเองโดยใช้หลักการคิดแบบรีอินฟอร์สเมนท์
3. ศึกษาการทำงานของวงจรอิเล็กทรอนิกส์ ในภาครับ(input) และแสดงเอ้าพุทได้ตามต้องการ
4. ศึกษาหาความรู้เพิ่มด้วยตนเองอย่างมีประสิทธิภาพ

1.3 ขอบข่ายของงาน

1. หุ่นยนต์สามารถจดจำเส้นทางในบริเวณพื้นที่สี่เหลี่ยมที่แบ่งเป็นกริดได้
2. หุ่นยนต์มีทักษะในการค้นหาเส้นทางเดินที่สั้นที่สุดได้
3. หุ่นยนต์สามารถเรียนรู้และจดจำได้ด้วยชีพสมองกล
4. สามารถที่จะประมวลผลโดยการใช้หลักการคิดแบบรีอินฟอร์สเมนท์

1.4 ขั้นตอนการดำเนินงาน

แบ่งขั้นตอนการดำเนินงานดังนี้

1. วางแผนการดำเนินงาน ศึกษาหาข้อมูลเกี่ยวกับโครงงานที่จะทำ
2. ศึกษากลไกการทำงานของหลักการคิดแบบรีอินฟอร์สเมนท์
3. ศึกษาวิธีเขียนโปรมแกรมวิชั่ว ซีพลัส พลัส
4. เขียนโปรแกรมเพื่อสร้างแบบจำลองทางคอมพิวเตอร์
5. ทดสอบโปรแกรม และวิเคราะห์ผล
6. ศึกษาและทำโปรแกรมไมโครคอนโทลเลอร์เพื่อใช้ควบคุมการเคลื่อนไหวของหุ่นยนต์
7. สร้างหุ่นยนต์และทำการโหลดโปรแกรมที่ออกแบบไว้ลงหุ่นยนต์เพื่อให้สามารถเดินได้ ทำการทดสอบและปรับปรุง
8. สรุปผลการทำงาน

บทที่ 2
กระบวนการเรียนรู้แบบรีอินฟอร์สเมนท์

2.1 กล่าวนำ

วิธีการเรียนรู้แบบรีอินฟอร์สเมนท์ เป็นวิธีการที่มีการเรียนรู้จากจุดมุ่งหมายโดยตรงแล้วจึงตัดสินใจในการทำตามจุดมุ่งหมายนั้น โดยโครงสร้างพื้นฐานเป็นการเรียนรู้ด้วยการตัดสินใจจากสภาวะแวดล้อมในลักษณะของ สถานะ (state) การกระทำ (action) และผลรางวัล (reward) ออน โพลิซี มัลติ คาลโล เป็นแบบหนึ่งของรีอินฟอร์สเมนท์ มีการทำงานที่มีลักษณะเป็นเอพพิโซด (episode) และต้องมีการปรับปรุงด้วยการเฉลี่ยผลรางวัลระยะยาวของทุก สถานะ และการกระทำอยู่โดยตลอด วิธีการเรียนรู้แบบรีอินฟอร์สเมนท์ สามารถนำมาประยุกต์ในการแก้ปัญหาได้โดยการทำการทดลองโดยใช้ วิธีการเรียนรู้แบบรีอินฟอร์สเมนท์ ในแบบออน โพลิซี มัลติ คาลโล (On-Policy Monte Carlo หรือ ONMC) ซึ่งสามารถนำมาใช้กับสถานการณ์ที่เราทราบข้อมูลของสิ่งแวดล้อมเพียงบางส่วนได้ โดยอาศัยหลักการในการตัดสินใจด้วยการนำผลเฉลี่ยระยะยาวของผลรางวัลสูงสุดที่เกิดจากการตัดสินใจเลือกการกระทำเมื่ออยู่ในแต่ละสถานะมาเป็นเงื่อนไขในการตัดสินใจ และจะกระทำการปรับปรุงผลเฉลี่ยระยะยาวนี้ทุกๆเอพพิโซด ซึ่งแต่ละเอพพิโซดจะเริ่มต้นจากจุดเริ่มต้นไปถึงจุดหมายที่เราต้องการ จึงเป็นการจบเอพพิโซด

ปัญหาสำคัญที่ยังคงเป็นอุปสรรค์ต่อการเรียนรู้ของหุ่นยนต์ คือ การที่หุ่นยนต์ยังไม่เข้าใจและไม่สามารถรับรู้ถึงสถานะของระบบ อีกทั้งยังไม่สามารถตัดสินใจเลือกการกระทำที่ดีที่สุดภายใต้สภาพแวดล้อมที่ศึกษาได้ ซึ่งจุดนี้เองเป็นเหตุผลที่เราจะต้องนำเอากระบวนการเรียนรู้แบบรีอินฟอร์สเมนท์เข้ามาช่วยแก้ปัญหา โดยการใช้อัลกอริทึมเพื่อหานโยบายที่ดีที่สุด และเราจะใช้ผลรางวัลเป็นตัวช่วยให้หุ่นยนต์สามารถรับรู้ถึงความพึงพอใจของเราต่อลักษณะการกระทำของตัวหุ่นยนต์ ซึ่งผลรางวัลนี้จะเป็นตัวชี้ทางให้กับหุ่นยนต์ในการเลือกเส้นทางที่เหมาะสมได้

2.2 วิธีการเรียนรู้แบบรีอินฟอร์สเมนท์ (Reinforcement Leaning)

การเรียนรู้(Learning) เป็นกระบวนการทางความคิดที่ทำให้เกิดการพัฒนาสติปัญญา ซึ่งจะเป็นกุญแจบ่งบอกถึงความอัจฉริยะของมนุษย์ และการเรียนรู้ยังช่วยเพิ่มความสามารถของตัวกระทำการตัดสินใจ (agent) ในการเลือกการกระทำที่เหมาะสมซึ่งจะส่งผลกระทบถึงการกระทำในอนาคตด้วย

วิธีการเรียนรู้แบบรีอินฟอร์สเมนท์ เป็นการเรียนรู้ที่ทำให้เกิดกระบวนการตัดสินใจและลงมือปฏิบัติได้ด้วยตนเอง เช่น ซอร์ฟแวร์ และหุ่นยนต์ เป็นต้น เป็นการเรียนรู้เพื่อเลือกการกระทำที่

เหมาะสมที่สุดที่จะสามารถนำพาให้บรรลุผลตามวัตถุประสงค์ แต่ในการเรียนรู้ต้องสามารถตระหนักถึงสภาพแวดล้อมและแสดงพฤติกรรมได้อย่างเหมาะสมในสภาพแวดล้อมที่ศึกษาอยู่ ตัวอย่างการประยุกต์ใช้งาน วิธีการเรียนรู้แบบรีอินฟอร์สเมนท์ มีดังนี้

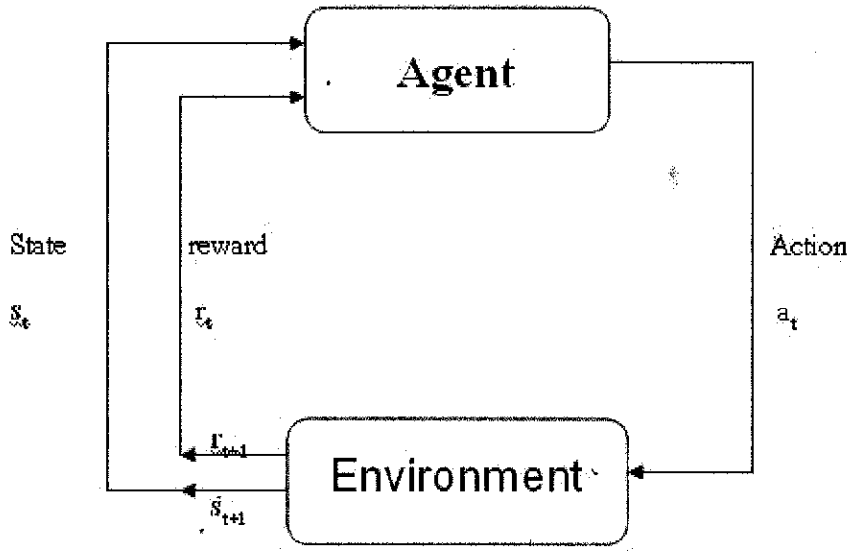
- การเรียนรู้เพื่อควบคุมการเคลื่อนที่ของหุ่นยนต์
- การเรียนรู้เพื่อให้เกิดการดำเนินงานอย่างเหมาะสมในโรงงานอุตสาหกรรม
- การเรียนรู้ในการเล่นเกมส์

กระบวนการเรียนรู้แบบรีอินฟอร์สเมนท์ เป็นวิธีที่สร้างความสามารถในการคิดและการเรียนรู้ให้เกิดขึ้นกับสิ่งประดิษฐ์หรือพวกหุ่นยนต์ต่างๆได้ โดยจะเน้นความสำคัญไปที่การเรียนรู้ด้วยตนเองภายในสิ่งแวดล้อมที่กำหนด วิธีนี้มีความโดดเด่นที่แตกต่างจากการเรียนรู้แบบดั้งเดิมซึ่งจะมุ่งแนวคิดในการเรียนรู้ด้วยการฝึกฝนจากผู้สอน หรือการโหลดแนวคิด ข้อมูล ให้ตัวหุ่นยนต์ได้ปฏิบัติตามเลย ปัจจุบันกระบวนการเรียนรู้แบบรีอินฟอร์สเมนท์ ได้รวบรวมเอาความรู้จากหลายๆ แขนงวิชามาประกอบเข้าด้วยกัน ไม่ว่าจะเป็นความรู้จากการศึกษาเชิงปฏิบัติการ อัลกอริทึมทางพันธุศาสตร์ โครงข่ายระบบประสาท จิตวิทยาด้านการศึกษาเกี่ยวกับการทำงานของจิตว่ามีผลต่อพฤติกรรมที่แสดงออกอย่างไร และวิศวกรรมควบคุม

การเรียนรู้แบบรีอินฟอร์สเมนท์จะเป็นแนวทางการคิด ให้หุ่นยนต์สามารถตัดสินใจเองได้ว่าควรปฏิบัติตัวอย่างไรเพื่อให้บรรลุจุดมุ่งหมาย โดยเราจะให้หุ่นยนต์พิจารณาว่า ณ สถานะจุดยืนปัจจุบันควรจะเลือกการกระทำอย่างไรเพื่อให้ตนได้รับผลรางวัลตอบแทนมากที่สุดด้วยกระบวนการนี้หุ่นยนต์จะมีการลองผิดลองถูกไปเรื่อย ๆ แล้วจึงค่อย ๆ เกิดพัฒนาการด้านความคิดจนหุ่นยนต์ สามารถเรียนรู้ด้วยตนเองได้ว่าเมื่อยืนอยู่ที่สถานะนี้ควรเลือกการกระทำใดในการปฏิบัติ

สิ่งหนึ่งที่เป็นตัวกระตุ้นแนวความคิดให้เกิดกับการเรียนรู้แบบรีอินฟอร์สเมนท์คือกระบวนการระหว่างการค้นหา(Exploration) และการหาประโยชน์หรือการหาผลรางวัลตอบแทน (Exploitation) ซึ่งตัวกระทำการเรียนรู้จะต้องให้ความสำคัญต่อลักษณะการกระทำเป็นอันดับแรกโดยวิเคราะห์ถึงผลรางวัลที่ได้รับต่อการกระทำต่าง ๆ

2.2.1 โครงสร้างของเรียนรู้แบบรีอินฟอร์สเมนท์



รูปที่ 2.1 โครงสร้างของเรียนรู้แบบรีอินฟอร์สเมนท์

- เวลา t สิ่งแวดล้อมในสถานะ S_t
- ตัวกระทำการตัดสินใจมีการเลือกการกระทำเป็น a_t
- ตัวกระทำการตัดสินใจมีการควบคุมโดยนโยบาย $\P = \P(S , a)$
- สิ่งแวดล้อมมีการเคลื่อนที่ที่สถานะ S_{t+1} จากจำนวนตัวอย่าง $P^{at}_{st\,st+1}$
- ตัวกระทำการตัดสินใจได้รับผลรางวัล r_{t+1}
- ความหน่าแน่น $R^{at}_{st\,st+1}$

2.2.2 ส่วนประกอบของรีอินฟอร์สเมนท์(Reinforcement Learning)

การเรียนรู้แบบรีอินฟอร์สเมนท์มีส่วนประกอบคือ ตัวกระทำการตัดสินใจ สถานะ การกระทำ ผลรางวัล นโยบาย(policy) ฟังก์ชันของผลรางวัล (reward function) ค่าของฟังก์ชัน (value function) ทางเลือก (optionally) และรูปแบบของสิ่งแวดล้อม (a model of the environment)

1. **ตัวกระทำการตัดสินใจ หมายถึง ตัวแทนที่ใช้ในการพิจารณาการทำงานของระบบ ซึ่งในที่นี้** ตัวกระทำการตัดสินใจคือ หุ่นยนต์ โดยการเลือกตัวกระทำการตัดสินใจต้องคำนึงถึงสิ่งแวดล้อมที่ศึกษาอยู่และลักษณะการทำงานของระบบ เละเป็นที่แน่นอนว่าในกระบวนการเรียนรู้ต่างๆ จะต้องมีตัวกระทำการเรียนรู้เพื่อเป็นตัวแทนในการศึกษา

2. สถานะ หมายถึง ตำแหน่งที่ตัวกระทำการตัดสินใจอยู่ในขณะนั้น เมื่อตัวกระทำตัดสินใจได้ทำการตัดสินใจเลือกการกระทำอย่างใดอย่างหนึ่งแล้ว สถานะของตัวกระทำการตัดสินใจก็จะเปลี่ยนไป โดยในการทำโครงงานนี้เรามีการกำหนดสถานะร้อยสถานะและมีลักษณะเป็นกริช
3. การกระทำ หมายถึง สิ่งที่ตัวกระทำการตัดสินใจเลือกที่จะปฏิบัติ โดยที่เราสามารถกำหนดรูปแบบของการกระทำได้ตามต้องการ เพื่อให้เหมาะสมต่อลักษณะการทำงาน
4. นโยบาย $\pi(s)$ คือ การกำหนดแนวทางการดำเนินงาน ซึ่งตัวกระทำการเรียนรู้จะต้องปฏิบัติตามเพื่อให้บรรลุจุดมุ่งหมาย และลักษณะของการกระทำได้กำหนดไว้อย่างเป็นแบบแผนแน่นอน โดยตัวกระทำการตัดสินใจจะมีการเรียนรู้พฤติกรรมในช่วงเวลาหนึ่งแล้วประเมินผลจากค่าเฉลี่ยผลรางวัลเพื่อให้ได้มาซึ่งนโยบายที่เหมาะสมที่สุด การศึกษานโยบายเป็นส่วนสำคัญของวิธีการเรียนรู้แบบรีอินฟอร์สเมนท์ ตัวกระทำการตัดสินใจจะมีการเรียนรู้พฤติกรรม โดยทั่วไปนโยบายเกี่ยวกับการตัดสินใจอาจจะเป็นค่าทางสถิติหรือค่าความน่าจะเป็นที่วัดได้
5. ฟังก์ชันของผลรางวัล R(s,a) คือ องค์ประกอบหลักที่ทำให้เกิดการเรียนรู้ เนื่องจากฟังก์ชันผลรางวัลจะเป็นตัวช่วยให้สามารถแก้ปัญหาที่เกิดจากการเลือกเส้นทางภายในสภาพแวดล้อมที่ศึกษาในกระบวนการเรียนรู้ได้ โดยเราจะกำหนดให้ผลรางวัลเป็นตัวแทนแสดงถึงความพึงพอใจของการเลือกสถานะและการกระทำต่างๆ รางวัลที่ให้ในปัญหาของวิธีการเรียนรู้แบบรีอินฟอร์สเมนท์จะเป็นสิ่งที่บอกถึงสถานะในสิ่งแวดล้อมโดยรางวัลเป็นการกระตุ้นความต้องการของสถานะ การเรียนรู้แบบรีอินฟอร์สเมนท์ มีเป้าหมายให้ได้ผลรวมรางวัลมากที่สุดที่จะได้รับในระยะยาว
6. ค่าของฟังก์ชัน V(s) จะเป็นตัวกำหนดความพึงพอใจให้กับตัวกระทำการเรียนรู้ มีลักษณะเป็นการประมาณค่าผลรวมทั้งหมดของผลรางวัลที่ตัวการะทำการเรียนรู้จะสามารถเก็บสะสมได้ ซึ่งคิดรวมตั้งแต่จุดเริ่มต้นจนถึงจุดสุดท้าย
7. รูปแบบของสิ่งแวดล้อม มีการจำลองพฤติกรรมบางอย่างของสิ่งแวดล้อม ตัวอย่างเช่นสถานะและการกระทำ รูปแบบสามารถทำนายผลของสถานะถัดไปและผลรางวัลถัดไป รูปแบบใช้สำหรับวางแผน โดยการตัดสินใจบนแนวความคิดของการกระทำโดยพิจารณาสถานการณ์ที่เป็นไปได้ในอนาคตก่อนก่อนเจอเหตุการณ์จริง การรวมกันของรูปแบบและการวางแผนในระบบการเรียนรู้แบบรีอินฟอร์สเมนท์เป็นรูปแบบการพัฒนาใหม่ๆ ระบบการเรียนรู้แบบรีอินฟอร์สเมนท์เป็นวิธีการทดลองเมื่อผิดก็ทดลองใหม่ และมีการกล่าวถึงวิธีโปรแกรมไดนามิก (Dynamic Programming) ด้วย

วิธีการเรียนรู้แบบรีอินฟอร์สเมนท์เป็นโครงสร้างของการคำนวณค่าของฟังก์ชัน ซึ่งไม่จำเป็นต้องมีความถูกต้องเมื่อใช้ในการแก้ปัญหาการเรียนรู้แบบรีอินฟอร์สเมนท์ ตัวอย่างเช่น การค้นคว้าวิธีกับขั้นตอนวิธีทางพันธุศาสตร์ โปรแกรมทางพันธุศาสตร์ การจำลองแบบอย่างง่าย และฟังก์ชันทางเลือกอื่นๆ เป็นวิธีที่ใช้ในการแก้ปัญหาการเรียนรู้แบบรีอินฟอร์สเมนท์ วิธีการเหล่านี้

เป็นการแก้ปัญหาเฉพาะหน้าของตัวกระทำการตัดสินใจส่งผลให้เกิดค่าของฟังก์ชัน เราเรียกว่า วิธีการสร้างวิวัฒนาการ (Evolutionary methods) ในที่นี้การสร้างวิวัฒนาการเป็นข้อดีของปัญหาการเรียนรู้ของตัวการตัดสินใจจะไม่แม่นยำในสถานะของสิ่งแวดล้อม

แม้ว่าวิธีการเรียนรู้แบบรีอินฟอร์สเมนท์เป็นการเรียนรู้ที่มีผลกับสิ่งแวดล้อมซึ่งวิธีการสร้างวิวัฒนาการเกิดขึ้นหรือไม่ เชื่อว่าเป็นวิธีที่สามารถนำข้อดีของรายละเอียดของผลกระทบต่อตัวบุคคลซึ่งสามารถมีประสิทธิภาพมากกว่าวิธีการสร้างวิวัฒนาการในหลายๆ กรณี วิธีการสร้างวิวัฒนาการเป็นวิธีที่ไม่รอบคอบเมื่อนำมาใช้กับโครงสร้างของปัญหาการเรียนรู้แบบรีอินฟอร์สเมนท์ เหล่านี้ไม่นำมาใช้เป็นข้อเท็จจริงของนโยบาย เราจะค้นคว้าสำหรับจากสถานะไปถึงการกระทำซึ่งสถานะของตัวบุคคลผ่านตลอดในระหว่างช่วงเวลาชีวิตของมัน หรือการกระทำที่ถูกเลือกในกรณีนี้ข้อมูลสามารถที่จะผิดได้ แต่หลายครั้งมากที่มันสามารถค้นคว้าได้อย่างมีประสิทธิภาพ แม้ว่าวิวัฒนาการและการเรียนรู้มีการแบ่งลักษณะเฉพาะและสามารถทำงานรวมกันได้

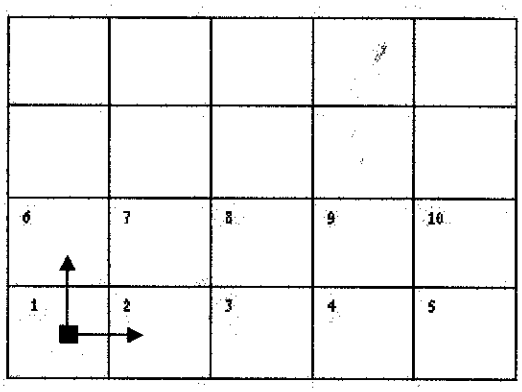
2.2.3 การสร้างหุ่นยนต์ให้เกิดการเรียนรู้

ในการสร้างหุ่นยนต์ให้เกิดการเรียนรู้จะใช้เซนเซอร์(Sensor) ช่วยในการมองเห็นสถานะของระบบแวดล้อม และให้หุ่นยนต์เป็นผู้ตัดสินใจเลือกการกระทำ เพื่อเปลี่ยนสถานะจุดยืนของตนเองภายในสภาพแวดล้อม

เซนเซอร์(Sensor) ⟶ กล้อง, รังสีโซนาร์

การกระทำ ⟶ เดินหน้า, ถอยหลัง, เลี้ยวซ้าย, เลี้ยวขวา

สถานะ ⟶ จุดยืนปัจจุบัน



รูปที่ 2.2 แสดงตัวอย่างการเลือกการกระทำ

ดังรูปที่ 1 ปัจจุบันหุ่นยนต์อยู่ที่สถานะที่1 หุ่นยนต์จะต้องเรียนรู้และตัดสินใจเลือกการกระทำ ซึ่งมีอยู่ 2ทางให้เลือก คือ เดินหน้า กับ เลี้ยวขวา

หากหุ่นยนต์เลือกเดินหน้า จุดยืนของหุ่นจะย้ายไปอยู่ที่ สถานะ 6

หากหุ่นยนต์เลือกเลี้ยวขวา จุดยืนของหุ่นจะย้ายไปอยู่ที่ สถานะ 2

การที่จะให้หุ่นยนต์ปฏิบัติงานได้บรรลุผลตามจุดประสงค์ที่ต้องการนั้น อาจจะต้องใช้เวลาในการทดลองและทดสอบการทำงานของหุ่นยนต์ภายในขอบเขตสภาพแวดล้อมที่ศึกษา เพราะขั้นแรกเราจะให้หุ่นยนต์ได้ลองผิดลองถูกไปเรื่อยๆ จนหุ่นเกิดการเรียนรู้และสะสมประสบการณ์มากขึ้น ซึ่งสิ่งที่จะเป็นผลตอบแทนที่เราให้หุ่นยนต์ ปฏิบัติงานให้บรรลุเป้าหมาย คือ รางวัล รางวัลจะเป็นผลตอบแทนที่เราให้แก่หุ่นยนต์ เมื่อหุ่นยนต์สามารถตัดสินใจเลือกการกระทำที่ถูกต้อง ตัวอย่างเช่น การที่เราพยายามสอนให้หุ่นยนต์เล่นเกมส์ หากหุ่นยนต์สามารถเล่นชนะ เราจะให้รางวัล +1 หากหุ่นยนต์เล่นเกมส์แพ้ เราจะให้ รางวัล = -1 แต่หากหุ่นยนต์อยู่ในสาถานะอื่น ๆ จะให้รางวัล = 0 การเรียนรู้เช่นนี้เป็นการเรียนรู้ทางอ้อม หุ่นต์ยนต์ต้องตัดสินใจเลือก การกระทำที่ทำให้เกิดการสะสมรางวัลค่ามากที่สุด วิธีการเรียนรู้แบบรีอินฟอร์สเมนท์ ให้ความสำคัญในแบบออน โพลิซี มัลติ คาลโล ซึ่งเป็นการเรียนรู้เพื่อให้ได้นโยบาย(policy) ที่เหมาะสมที่สุด โดยใช้รางวัล เป็นตัวกำหนดนโยบาย

เป็นที่รู้กันดีว่า รางวัลเป็นตัวกำหนดให้หุ่นยนต์สามารถปฏิบัติงานบรรลุตามจุดมุ่งหมาย การให้ความสำคัญของรางวัลจะแตกต่างกันไปในแต่ละการกระทำ และสถานะ ซึ่งหุ่นยนต์จะเกิดพัฒนาการและเรียนรู้ได้เองว่าควรจะเลือกการกระทำ และสถานะใด เพื่อให้บรรลุวัตถุประสงค์และได้รับรางวัลค่ามากที่สุด

หุ่นยนต์จะสามารถเรียนรู้เพื่อหานโยบายที่เหมาะสมได้โดย

1. หุ่นยนต์ตัดสินใจเลือกที่จะทำการกระทำ ต่างๆแบบสุ่ม
2. หุ่นยนต์จะประมวลผลรางวัลที่ได้รับเมื่อเลือกการกระทำต่างๆ
3. จะเกิดกระบวนการเรียนรู้ ทำให้หุ่นยนต์สามารถเลือกการกระทำที่ถูกต้อง โดยได้รับรางวัลมากที่สุด ทำให้ได้นโยบายที่เหมาะสมที่สุด

2.2.4 ปัญหาที่เกิดจากวิธีการเรียนรู้แบบรีอินฟอร์สเมนท์

1. ต้องใช้ระยะเวลาในการทดลองนาน กว่าที่หุ่นยนต์จะมีการเรียนรู้เพื่อเลือกเส้นทางที่ดีที่สุดเพื่อไปถึงจุดหมายที่เราต้องการได้อย่างถูกต้อง
2. การเรียนรู้พฤติกรรมของตัวกระทำการตัดสินใจที่พบในการทดลองมักเลือกตัดสินใจเดินทางผิดพลาด ภายใต้สิ่งแวดล้อมที่กำหนด
3. การเรียนรู้แบบรีอินฟอร์สเมนท์ต้องเป็นลำดับขั้น ซึ่งเราต้องมีการปรับปรุงฟังก์ชันค่า Q ให้มีค่ามากที่สุด

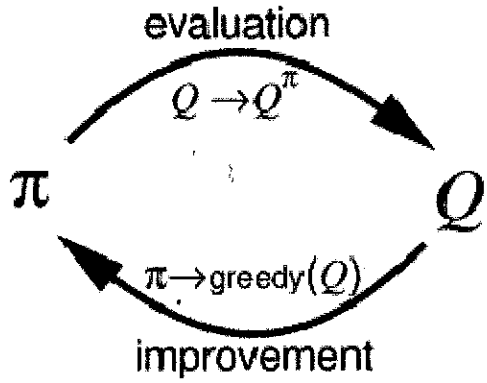
4. การประมาณค่าของฟังก์ชัน สิ่งสำคัญที่สุดของการเรียนรู้แบบรีอินฟอร์สเมนท์คือค่าของฟังก์ชัน โดยค่าของฟังก์ชันต้องแสดงอย่างถูกต้อง ตัวอย่างซึ่งเป็นส่วนที่ใช้ในระบบเครือข่ายแต่การฝึกฝนต้องใช้เวลานาน
5. การเรียนรู้แบบรีอินฟอร์สเมนท์กับการกระทำในพื้นที่ว่าง ในปัญหาหลายอย่างๆ ต้องใช้การประยุกต์ จำนวนของการกระทำมีได้หลายแบบ เรามีการศึกษาวิธีที่มีประสิทธิภาพสำหรับการเรียนรู้และการกระทำในพื้นที่ที่ว่าง ซึ่งเป็นวิธีพื้นฐานในการเปลี่ยนแปลงความน่าจะเป็นของการกระทำที่คลายกันระหว่างการเรียนรู้
6. วิธีการเรียนรู้แบบรีอินฟอร์สเมนท์ที่ไม่เป็นรูปแบบที่แน่นอน มีการศึกษากระบวนการในการสำรวจสิ่งแวดล้อม ในหลายส่วนต้องมีการกำหนดค่า อย่างไรก็ตามในระบบที่มีการออกแบบสามารถวางแผนให้อยู่ในรูปแบบของระบบ ซึ่งนโยบายต้องเป็นการหาเส้นทางที่ดีที่สุด

2.3 มัลติ คาลโล คอนโทรล (Monte Carlo Control)

มัลติ คาลโลเป็นการสุ่มตัวอย่าง เพื่อให้ได้ผลเฉลี่ยของคำตอบที่ถูกต้องจากจำนวนของตัวอย่างของคำตอบที่มีมากมาย และเป็นอัลกอริทึมที่ใช้ในการประมาณค่าของฟังก์ชัน(value function) และยังสามารถนำมาใช้ประโยชน์ในการสร้างประสบการณ์ ความชำนาญ เพื่อก่อให้เกิดกระบวนการเรียนรู้ด้วยตนเองได้ ซึ่งสิ่งเหล่านี้จะเป็นกุญแจสำคัญที่จะนำไปสู่การหานโยบายที่เหมาะสมที่สุดได้ โดยกระบวนการเรียนรู้จะค่อยๆเกิดขึ้นอย่างต่อเนื่องและสัมพันธ์กับองค์ประกอบสำคัญคือ state action reward ส่วนการประมวลผลนั้นเราจะใช้การจำลองสถานการณ์ทางคอมพิวเตอร์ โดยเราจะต้องกำหนดตัวกระทำการเรียนรู้ในโครงงานนี้คือหุ่นยนต์ ขอบเขตของสภาพแวดล้อมที่จะศึกษา และมีการกำหนดระดับความพึงพอใจให้ตัวกระทำการเรียนรู้ผ่านทางผลรางวัล กระบวนการเรียนรู้จะเกิดขึ้นอย่างเป็นระดับขั้นเนื่องจากตัวกระทำการเรียนรู้จะมีการเก็บสะสมผลรางวัลไปเรื่อยๆในทุกแอปพลิโซด เปรียบดั่งการเพิ่มประสบการณ์ให้กับตัวกระทำการเรียนรู้ทีละน้อย จนเกิดเป็นความชำนาญ

มัลติ คาลโล ซึ่งเป็นแบบหนึ่งของรีอินฟอร์สเมนท์ มีการทำงานที่มีลักษณะเป็นเอพพิโซด และต้องมีการปรับปรุงด้วยการเฉลี่ยรางวัลระยะยาวของทุก สถานะ และการกระทำ อยู่โดยตลอด ซึ่งวิธีการนี้จะมีรูปแบบเหมือนกับบทที่ว่าด้วย โปรแกรมไดนามิค

เป็นรูปแบบของอัลกอริทึมที่สามารถคำนวณหานโยบายที่เหมาะสมภายในขอบเขตแวดล้อมที่กำหนดให้ได้เช่นเดียวกับ Markov Decision Process (MDP) โดยจะนำประโยชน์จากกระบวนการของ generalized policy iteration (GPI) ซึ่งเป็นการอัพเดตค่าในทุกๆแอพพลิโซด



รูปที่ 2.3 แสดงกระบวนการเรียนรู้ในรูปแบบออน โพลิซี มัลติ คาสโล

โดย Q คือ ฟังก์ชันการเก็บค่าสถานะและการกระทำ

Q^{π} คือ ฟังก์ชันการเก็บค่าสถานะและการกระทำที่นำมาเป็นนโยบาย

π คือ นโยบายในการตัดสินใจ

ซึ่งกระบวนการทำงานของออน โพลิซี มัลติ คาสโล จะมีการนำค่าในฟังก์ชัน Q มาเป็นนโยบายในการตัดสินใจ ซึ่งจะมีการอัพเดทค่า Q ตลอดจนกว่าจะได้นโยบายในการตัดสินใจที่เหมาะสมที่สุด โดยขั้นตอนการพัฒนานโยบายการตัดสินใจและการอัพเดทค่าในฟังก์ชัน Q จะกระทำในลักษณะสลับกันไปเรื่อยๆ ดังนี้

$\pi 0 \rightarrow$ อัพเดทค่า$\rightarrow Q^{\pi o} \rightarrow$ พัฒนา$\rightarrow \pi 1 \rightarrow$ อัพเดทค่า$\rightarrow Q^{\pi 1} \rightarrow$ พัฒนา$\rightarrow \pi 2 \rightarrow \ldots \pi^* \rightarrow$ พัฒนา$\rightarrow Q^{\pi^*}$

โดยที่การอัพเดทค่าได้จากการหาค่าเฉลี่ยฟังก์ชัน Q และการพัฒนาเป็นการปรับปรุงนโยบายในการตัดสินใจเพื่อให้ได้นโยบายที่เหมาะสมที่สุด การพัฒนาในวิธีดังกล่าวนี้ควรที่จะมีการทดลองในหลายๆเอพพิโซดเพื่อที่การประมาณค่าฟังก์ชันของเราจะเข้าใกล้ค่าจริงมากขึ้น

การพัฒนานโยบายในการตัดสินใจจะใช้นโยบายในการพัฒนาแบบกรีดี้(greedy) ซึ่งจะเป็นการเลือกการกระทำที่มีความน่าจะเป็นมากที่สุด การหานโยบายที่เหมาะสมของแต่ละสถานะสามารถแสดงเป็นสมการได้ดังนี้

$$\pi(s) = \arg_a \max Q(s,a)$$

เมื่อ สถานะที่พิจารณาเป็นส่วนหนึ่งในสถานะทั้งหมด ($s \in S$)

$\pi(s)$ = นโยบายที่ใช้ในการตัดสินใจ ณ สถานะที่พิจารณา

Q(s,a) = ฟังก์ชัน Q ของสถานะและการกระทำ

$$\begin{aligned} Q^{\pi k}(s,\pi_{k+1}(s)) &= Q^{\pi k}(s,\arg\max Q^{\pi k}(s,a)) \\ &= \max Q^{\pi k}(s,a) \\ &\geq Q^{\pi k}(s,\pi_k(s)) \\ &= V^{\pi k}(s) \end{aligned}$$

โดยเราสามารถแสดงอัลกอริทึม (Algorithm) ของวิธีการคิดแบบ มัลติคาลโล คอนโทรล ได้ดังนี้

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \leftarrow$ arbitrary
$\pi(s) \leftarrow$ arbitrary
$Returns(s,a) \leftarrow$ empty list

Repeat forever:
(a) Generate an episode using exploring starts and π
(b) For each pair s,a appearing in the episode:
$R \leftarrow$ return following the first occurrence of s,a
Append R to $Returns(s,a)$
$Q(s,a) \leftarrow$ average$(Returns(s,a))$
(c) For each s in the episode:
$\pi(s) \leftarrow \arg\max_a Q(s,a)$

รูปที่ 2.4 อัลกอริทึม (Algorithm) ของวิธีการคิดแบบ มัลติคาลโล คอนโทรล

ซึ่งวิธีมัลติ คาลโล คอนโทรลนี้ สามารถแบ่งย่อยได้อีก 2 วิธี คือ แบบออน โพลิซี มัลติ คาลโล (on-policy Monte Carlo หรือ ONMC) และ แบบอ๊อฟ โพลิซี มัลติ คาลโล (off-policy Monte Carlo หรือ OFMC) ซึ่งจะกล่าวถึงรายละเอียดในหัวข้อต่อไป

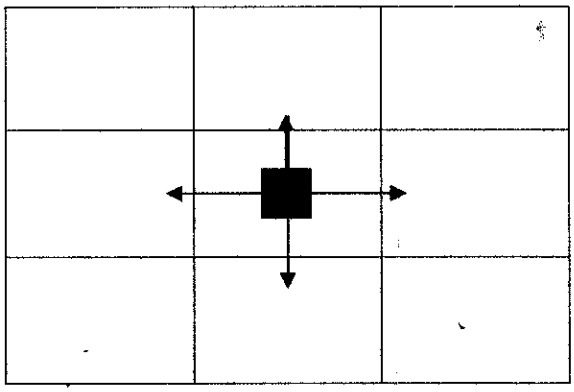
2.4 ออน โพลิซี มัลติ คาลโล (on-policy Monte Carlo หรือ ONMC)

ออน โพลิซี มัลติ คาลโล เป็นวิธีการประเมินผลที่ใช้หลักการพัฒนานโยบายเพื่อเป็นหลักในการตัดสินใจ โดยทั่วไปวิธีนี้จะใช้นโยบายแบบซอฟท์ (soft policy) นั่นก็คือจะเลือกนโยบายในการตัดสินใจที่มีค่าความน่าจะเป็นมากกว่าศูนย์ดังสมการนี้

$$\pi(s,a) > 0 \qquad \text{;สำหรับการกระทำในทุกๆสถานะ}$$

ซึ่งในที่นี้จะกล่าวถึงนโยบายแบบ ε-greedy policies ซึ่งวิธีการนี้จะเป็นการตัดสินใจเลือกการกระทำโดยเลือกความน่าจะเป็นแบบสุ่มที่มีค่ามากที่สุด โดยใช้ค่าความน่าจะเป็นแทนด้วย ε

ตัวอย่างเช่น ในบริเวณสิ่งแวดล้อมหนึ่งมีการกระทำที่สามารถเลือกได้อยู่ 4 การกระทำคือ ซ้าย ขวา บน และล่าง



รูปที่ 2.5 แสดงทิศทางการกระทำที่เป็นไปได้ทั้งหมด

ในที่นี้กำหนดให้การกระทำไปทางขวามีค่าความน่าจะเป็นแบบสุ่มมากที่สุด คือ

ε = 0.6 สำหรับค่าความน่าจะเป็นของการกระทำในทิศทางอื่นจะมีค่าเท่ากับ (1-ε) / A(S)

นั่นคือ (1-0.6)/4 = 0.1

ในที่สุดแล้วเราจะเลือกการกระทำทางขวาด้วยความน่าจะเป็น เท่ากับ ε + [(1 - ε) / A(S)]

นั่นคือ 0.6+[(1-0.6)/4] = 0.7

โดยเราสามารถแสดงอัลกอริทึม ของวิธีการคิดแบบ มัลติคาลโล คอนโทรล ได้ดังนี้

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \leftarrow$ arbitrary
$Returns(s,a) \leftarrow$ empty list
$\pi \leftarrow$ an arbitrary ε-soft policy

Repeat forever:
(a) Generate an episode using π
(b) For each pair s,a appearing in the episode:
$R \leftarrow$ return following the first occurrence of s,a
Append R to $Returns(s,a)$
$Q(s,a) \leftarrow$ average$(Returns(s,a))$
(c) For each s in the episode:
$a^* \leftarrow \arg\max_a Q(s,a)$
For all $a \in \mathcal{A}(s)$:

$$\pi(s,a) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(s)| & \text{if } a = a^* \\ \varepsilon/|\mathcal{A}(s)| & \text{if } a \neq a^* \end{cases}$$

รูปที่ 2.6 อัลกอริทึม (Algorithm) ของวิธีการคิดแบบ มัลติคาลโล คอนโทรล

แนวคิดรวบยอดของออน โพลิซี มัลติ คาลโล ยังคงใช้หลักการของ จี พี ไอ (genearalized policy iteration ,GPI) ซึ่งเป็นการอัพเดตค่าในทุกๆแอพพลิโซด และใช้วิธีการเก็บค่าการประเมินผลแบบเฟิร์ส วิสิท (First Visit)

2.4.1 วิธีการเก็บค่าการประเมินผลแบบเฟิร์ส วิสิท (First Visit)

การหาค่าเฉลี่ยของผลรางวัลแบบเฟิร์ส วิสิท คือ การคำนวณค่า $V^{\pi}(s)$ ซึ่งเป็นค่าเฉลี่ยของผลรางวัลที่คำนวณได้ในสถานะ s หากเกิดการเดินทางซ้ำรอยเดิมอีกจะไม่คิดรวมค่าผลรางวัลนั้น กล่าวคือ เป็นการเก็บค่าผลรางวัลทุกครั้งเมื่อเจอเหตุการณ์นั้นครั้งแรก โดยจะเก็บค่าลงในตารางคิว

- วิธีนี้ได้ผลค่อนข้างช้าเนื่องจากจะต้องทดลองระยะยาว คือ มีการวนซ้ำหลายๆแอปพลิโซด
- ผลลัพธ์ที่ได้ให้ความถูกต้องแม่นยำค่อนข้างสูง

บทที่ 3
การเขียนโค้ดโปรแกรม

3.1 กล่าวนำ

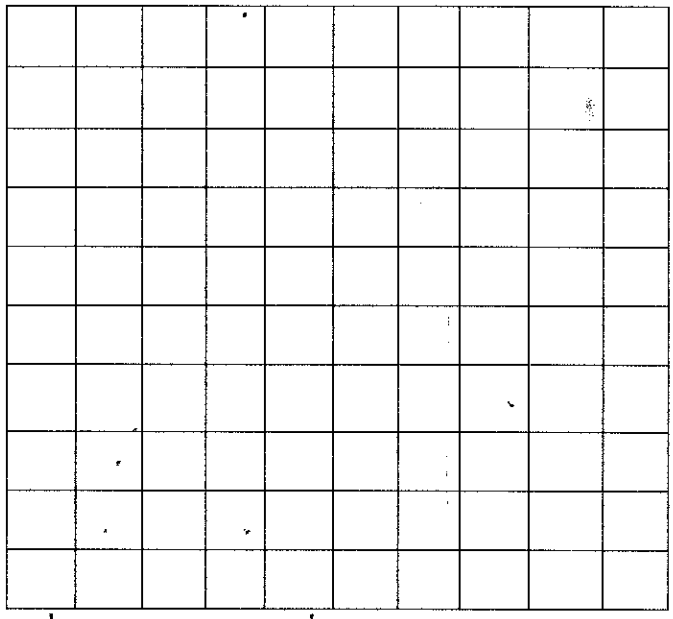
ระบบปฏิบัติการวินโดวส์เป็นระบบปฏิบัติการแบบกราฟฟิก ที่อำนวยความสะดวกในการใช้งานได้เป็นอย่างมาก เพราะรูปแบบการใช้งานจะเป็นแบบหน้าต่าง และมียูสเซอร์ อินเตอร์เฟซ (User Interface) เพื่อใช้ในการติดต่อกับผู้ใช้ในรูปแบบกราฟฟิก ปัจจุบันได้มีเครื่องมือที่ช่วยในการพัฒนาโปรแกรมบนวินโดวส์ขึ้นมามากมาย เช่น ไมโครซอร์ฟ วิชั่ว เบสิค (Microsoft Visual Basic) วิชั่ว ซี++ (Visual C++) พาวเวอร์ บิวเดอร์ (Power Builder) หรือ เดวฟี (Delphi) เป็นต้น

วิชั่ว ซี++ ก็เป็นโปรแกรมประเภท วิชั่ว อีกตัวหนึ่งจากบริษัทไมโครซอร์ฟ ผู้ผลิตระบบปฏิบัติการวินโดวส์เป็นเครื่องมือพัฒนาโปรแกรมที่มีความสามารถสูงในยุคนี้ วิชั่ว ซี++ ได้รับการพัฒนาให้มีความยืดหยุ่นและมีประสิทธิภาพสูงขึ้นมาจากภาษา ซี++ และได้สนับสนุนการพัฒนาโปรแกรมในหลายๆด้าน ไม่ว่าจะเป็นการสร้างโปรแกรมทั่วไป การสร้างโปรแกรมจัดการฐานข้อมูล การสร้างโปรแกรมระบบเครือข่าย หรือมัลติมิเดียอย่างครบครัน

วิชั่ว ซี++ ได้รับการพัฒนาขึ้นมาจาก ไมโครซอร์ฟ ซี/ซี++ ให้เป็น ไอดีอี(IDE) ที่ทำงานบนระบบปฏิบัติการวินโดวส์ได้อย่างเต็มที่ โดยไอดีอี (Integrated Development Environment) เป็นโปรแกรมซึ่งมีไว้ใช้สำหรับเพิ่มความสะดวกในการสร้างและการแก้ไขโปรเจ็กต์ให้ง่ายขึ้น โดยจะมีเครื่องมืออำนวยความสะดวกในการพัฒนาโปรแกรมต่างๆ ให้เรียกใช้งานภายใน ไอดีอี เช่น โปรแกรมที่ใช้ในการดีบัก คอมไพเลอร์ และลิงค์เกอร์ เป็นต้น และ วิชั่ว ซี++ ยังมี เอ็มเอฟซี (MFC : Microsoft Foundation Class) เป็นไลบรารีที่จะช่วยอำนวยความสะดวกในการพัฒนาโปรแกรมบนวินโดวส์อีกด้วย ซึ่งโค้ดโปรแกรมที่เขียนขึ้นโดย เอ็มเอฟซี จะมีขนาดเล็กและไม่ซับซ้อน ทำให้ง่ายต่อการพัฒนาโปรแกรม

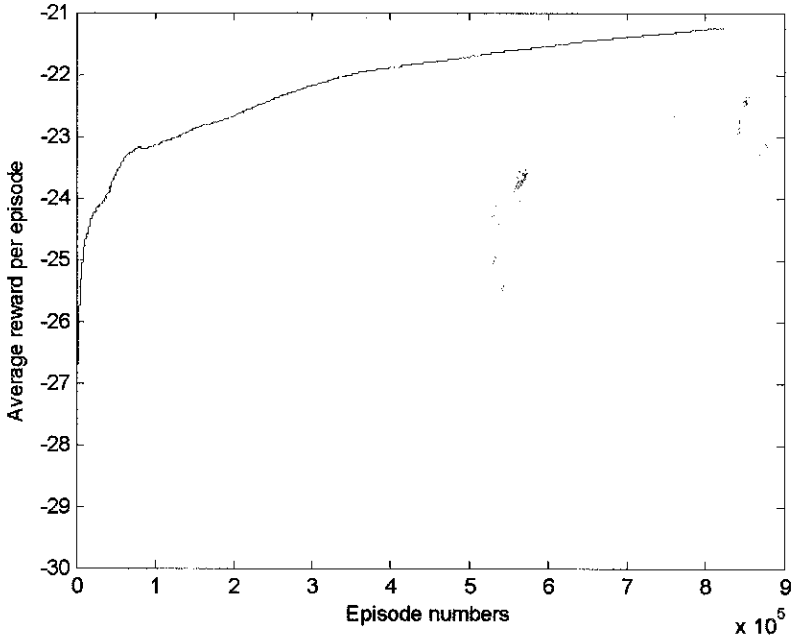
3.2 ผลการทดสอบโปรแกรม

จากผลการรันโปรแกรมเพื่อให้หุ่นยนต์ได้มีการเรียนรู้และหาประสบการณ์ในการหาเส้นที่ดีที่สุดซึ่งขนาดของขอบเขตที่เรากำหนดให้โดยใช้ขนาดของกริดเป็น 10x10 และไม่มีสิ่งกีดขวางใดๆ พบว่าผลการทดลองเป็นดังนี้



รูปที่ 3.1 รูปแบบของกริดที่ใช้ในการทดลองขนาด 10 x 10

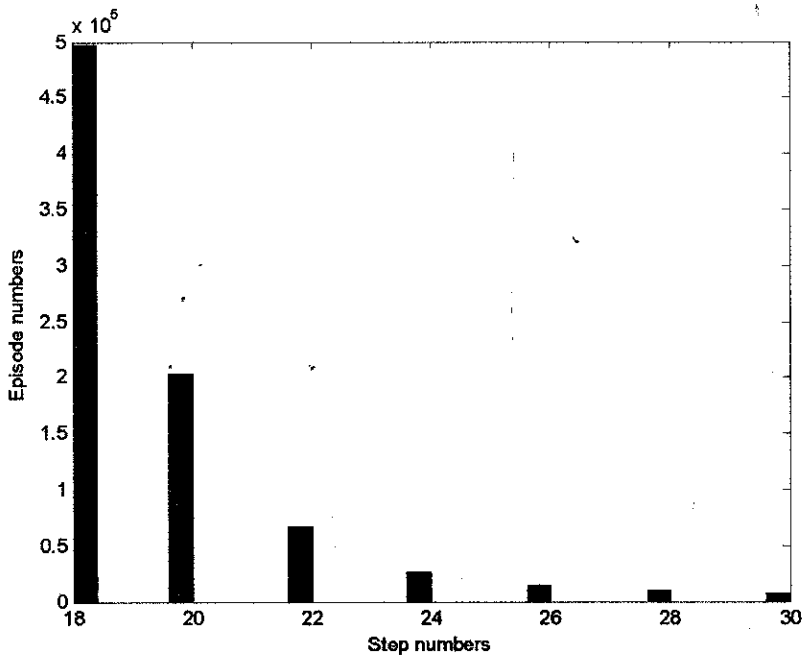
การแสดงการเปรียบเทียบระหว่างจำนวนเอพพิโซดกับผลเฉลี่ยของรางวัล



รูปที่ 3.2 การแสดงการเปรียบเทียบระหว่างจำนวนเอพพิโซดกับผลเฉลี่ยของรางวัล

จากกราฟรูปที่ 3.2 พบว่าเมื่อเราทำการรันโปรแกรมให้หุ่นยนต์ได้มีการเรียนรู้ในการหาเส้นทางโดยหุ่นยนต์ได้มีการทดลองเดินลองผิดลองถูกเพื่อหาเส้นที่สั้นที่สุดเพื่อให้ไปถึงจุดมุ่งหมายในจำนวน 1000000 เอพพิโซดพบว่าการเรียนรู้ของหุ่นยนต์มีการเรียนรู้ดีขึ้นเรื่อยๆ ซึ่งดูจากค่าเฉลี่ยของผลรางวัลที่มีค่าเพิ่มเรื่อยๆ และแนวโน้มที่จะคงที่มันสามารถบอกได้ว่าหุ่นยนต์นั้นพบเส้นทางถูกต้องและสั้นที่สุด

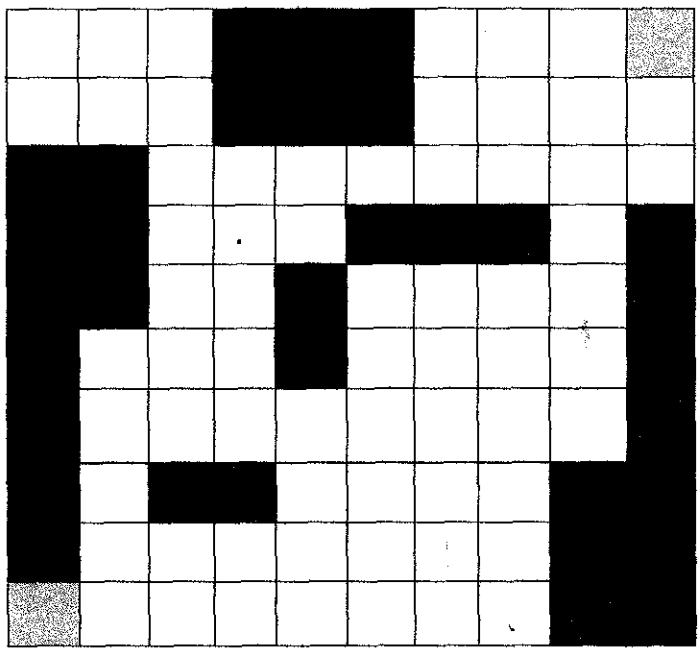
การแสดงการเปรียบเทียบระหว่างจำนวนเอพพิโซดกับจำนวนสเต็ป



รูปที่ 3.3 การแสดงการเปรียบเทียบระหว่างจำนวนเอพพิโซดกับจำนวนสเต็ป

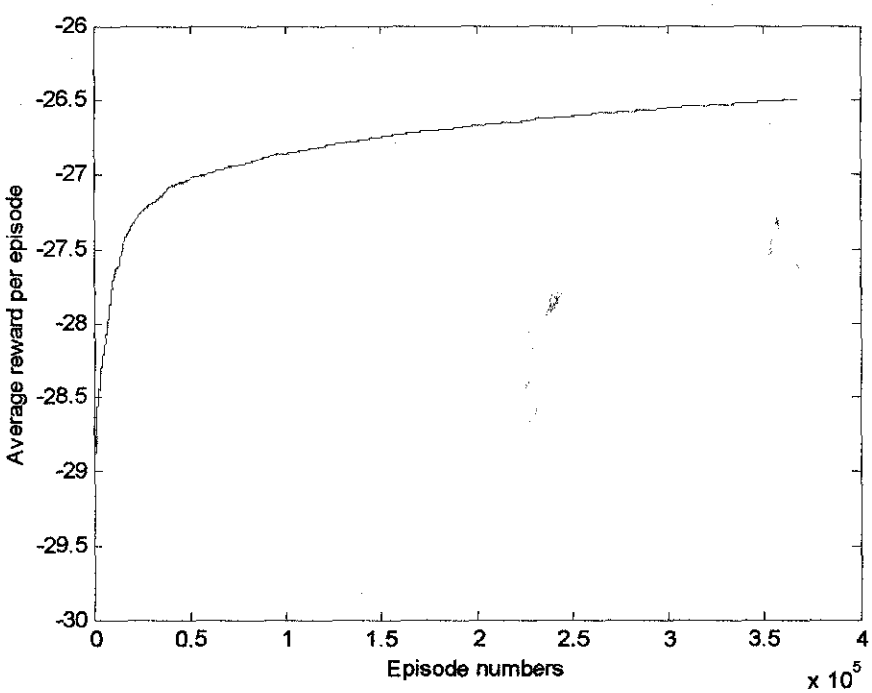
จากกราฟรูปที่ 3.3 พบว่าเมื่อหุ่นยนต์ได้มีการเรียนรู้ในการหาเส้นทางโดยการลองผิดลองถูกซึ่งในช่วงแรก ๆ นั้นหุ่นยนต์ได้เลือกเดินในเส้นทางที่ใช้จำนวนสเต็ปมากซึ่งจากการเรียนรู้และประสบการณ์ในการเลือกเส้นทางในหลายๆ เอพพิโซดพบว่ามีแนวโน้มที่ดีขึ้นเรื่อยๆ ซึ่งดูได้จากกราฟและหุ่นยนต์สามารถที่จะหาเส้นทางที่สั้นที่สุด ถูกต้องและจำนวนสเต็ปน้อยที่สุดคือจำนวน 18 ก้าวซึ่งเป็นเส้นทางที่เราต้องการ

จากผลการรันโปรแกรมเพื่อให้หุ่นยนต์ได้มีการเรียนรู้และหาประสบการณ์ในการหาเส้นที่ดีที่สุดซึ่งขนาดของขอบเขตที่เรากำหนดให้โดยใช้ขนาดของกริดเป็น 10x10 และมีสิ่งกีดขวางใดๆ พบว่าผลการทดลองเป็นดังนี้



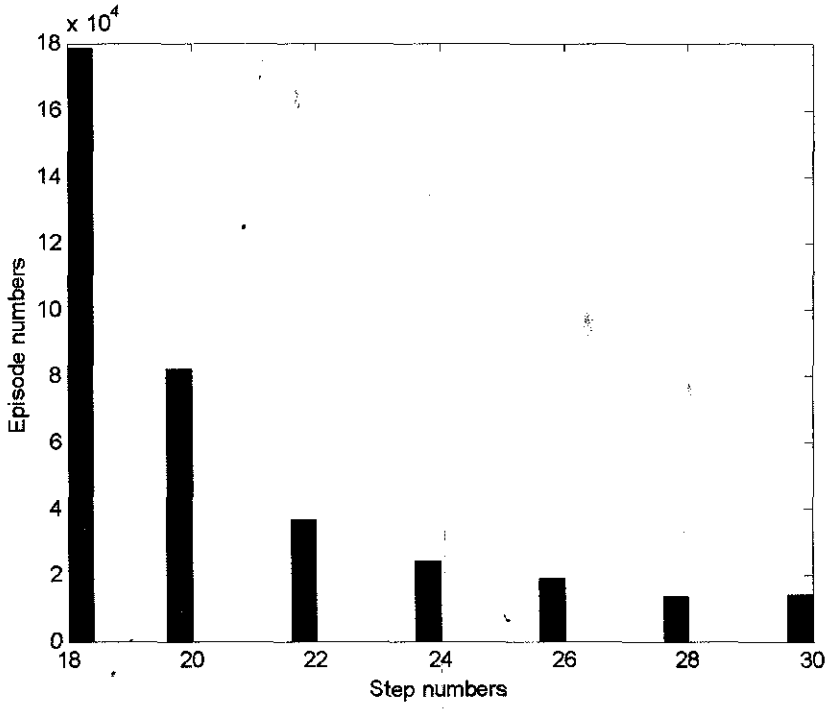
รูปที่ 3.4 รูปแบบของกริดที่ใช้ในการทดลองขนาด 10 x 10

การแสดงการเปรียบเทียบระหว่างจำนวนเอพพิโซดกับผลเฉลี่ยของรางวัล



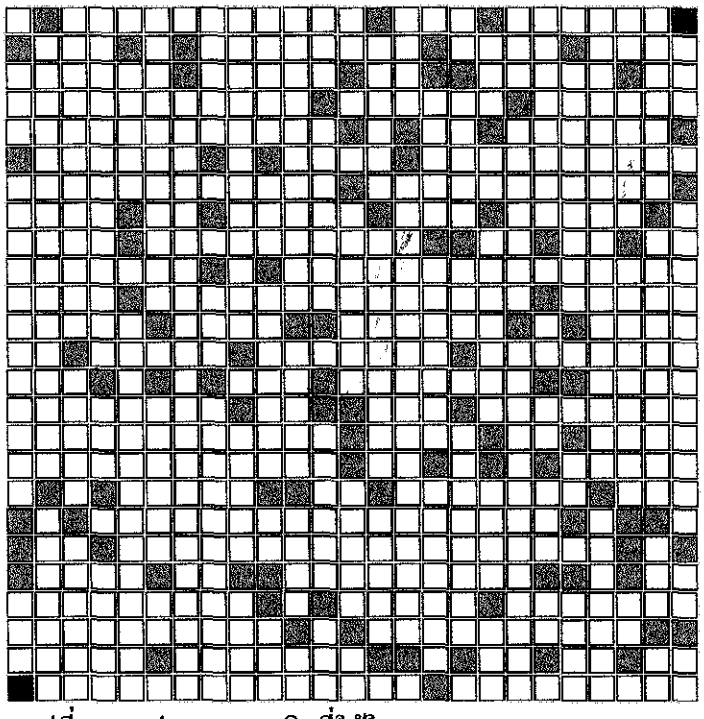
รูปที่ 3.5 การแสดงการเปรียบเทียบระหว่างจำนวนเอพพิโซดกับผลเฉลี่ยของรางวัล

กราฟแสดงการเปรียบเทียบระหว่างจำนวนเอพพิโซดกับจำนวนสเต็ป



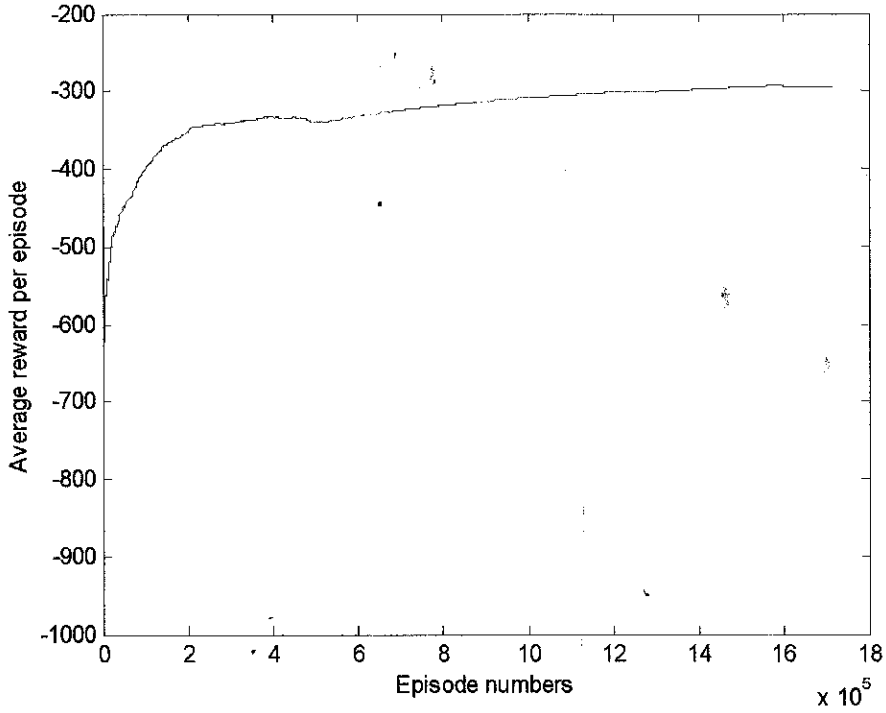
รูปที่ 3.6 การแสดงการเปรียบเทียบระหว่างจำนวนเอพพิโซดกับจำนวนสเต็ป

จากผลการรันโปรแกรมเพื่อให้หุ่นยนต์ได้มีการเรียนรู้และหาประสบการณ์ในการหาเส้นที่ดีที่สุดซึ่งขนาดของขอบเขตที่เรากำหนดให้โดยใช้ขนาดของกริดเป็น 25x25 และมีสิ่งกีดขวางใดๆ พบว่าผลการทดลองเป็นดังนี้



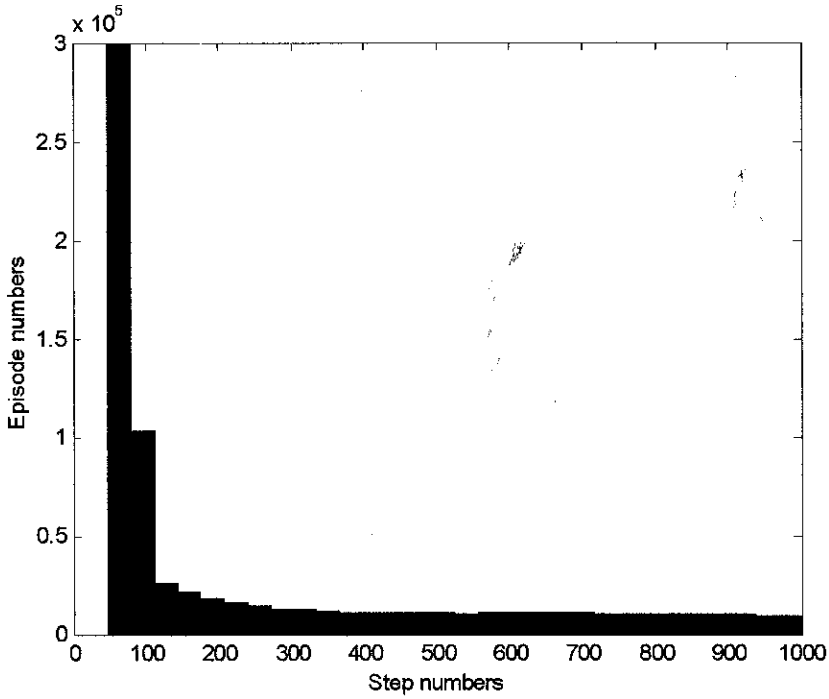
รูปที่ 3.7 รูปแบบของกริดที่ใช้ในการทดลองขนาด 25 x 25

การแสดงการเปรียบเทียบระหว่างจำนวนเอพพิโซดกับผลเฉลี่ยของรางวัล



รูปที่ 3.8 การแสดงการเปรียบเทียบระหว่างจำนวนเอพพิโซดกับผลเฉลี่ยของรางวัล

การแสดงการเปรียบเทียบระหว่างจำนวนเอพพิโซดกับจำนวนสเต็ป



รูปที่ 3.9 การแสดงการเปรียบเทียบระหว่างจำนวนเอพพิโซดกับจำนวนสเต็ป

3.3 การทดลองเปรียบเทียบผลของ การเรียนรู้แบบรีอินฟอร์สเมนท์ กับมนุษย์

รูปแบบการทดลอง

1. ออกแบบพื้นที่ โดยมีพื้นที่สี่แบบด้วยกันคือ
   - พื้นที่แบบกริดสี่เหลี่ยมขนาด 25x25 โดยมีสิ่งกีดขวาง
   - พื้นที่แบบกริดสี่เหลี่ยมขนาด 25x25 โดยมีสิ่งกีดขวาง
   - พื้นที่แบบกริดสี่เหลี่ยมขนาด 25x25 โดยมีสิ่งกีดขวาง
2. ทำการทดลองโดยกำหนดให้มนุษย์กับการเรียนรู้แบบรีอินฟอร์สเมนท์หาเส้นทางที่ดีที่สุด โดยใช้พื้นที่เดียวกันเพื่อเปรียบเทียบผลที่ออกมาว่าเป็นไปตามสมมติฐานที่ตั้งไว้หรือไม่ ที่ว่า การเรียนรู้แบบรีอินฟอร์สเมนท์สามารถที่จะหาเส้นทางการเดินที่ดีกว่ามนุษย์ได้ในกรณีที่สภาพแวดล้อมที่ใช้ในการทดลองมีความซับซ้อนมากยิ่งขึ้น ซึ่งการแสดงผลเราจะแสดงออกมาในรูปแบบของกราฟเชิงเส้น โดยการทดลองนี้กำหนดให้มี 3 คนในกานทดลอง
3. สรุป และวิเคราะห์ผล

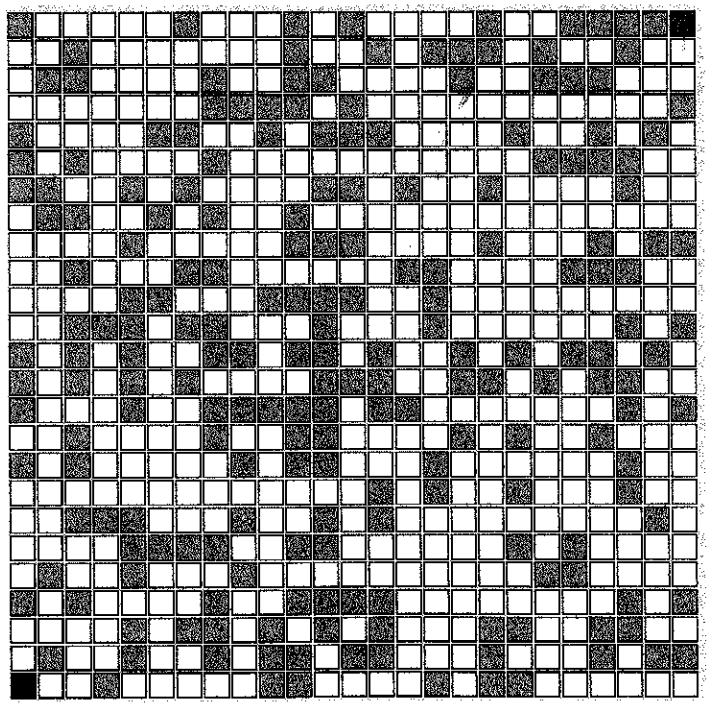
สมมติฐาน : ทำการศึกษาเปรียบเทียบกระบวนการตัดสินใจระหว่างมนุษย์กับหลักการเรียนรู้แบบรีอินฟอร์สเมนท์ว่าในสภาวะที่มีความซับซ้อน หลักการเรียนรู้แบบรีอินฟอร์สเมนท์จะมีการตัดสินใจได้ดีกว่ามนุษย์

ตัวแปรต้น : หลักการเรียนรู้แบบรีอินฟอร์สเมนท์ มนุษย์

ตัวแปรตาม : จำนวนก้าว

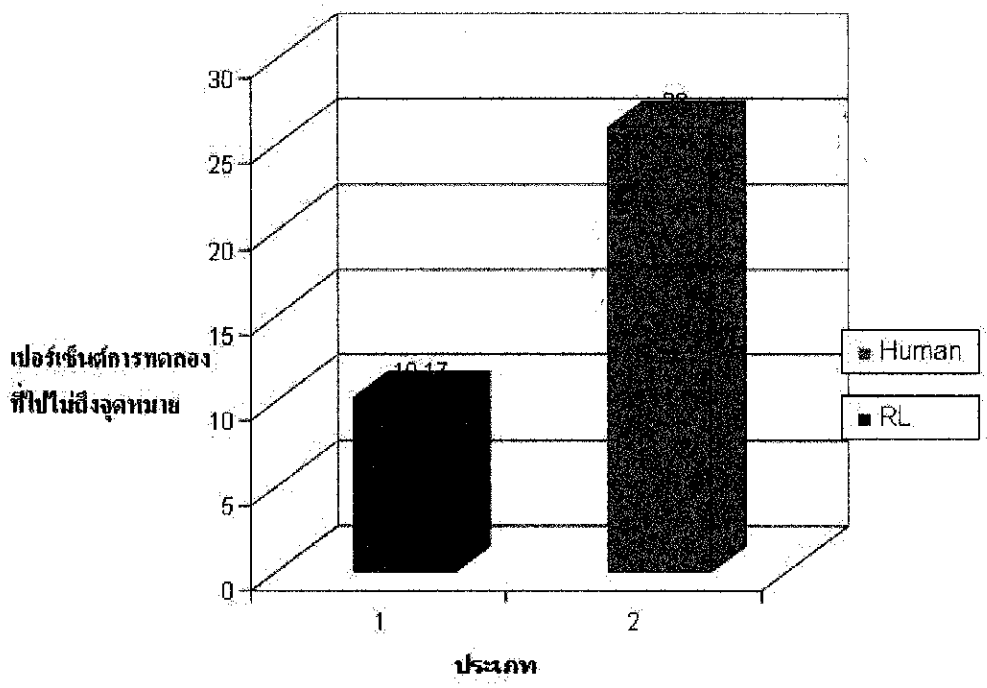
ตัวแปรควบคุม : รูปแบบพื้นที่ที่ใช้ในการทดลอง

กรณีที่1 กริด 25X25 แบบมีสิ่งกีดขวาง



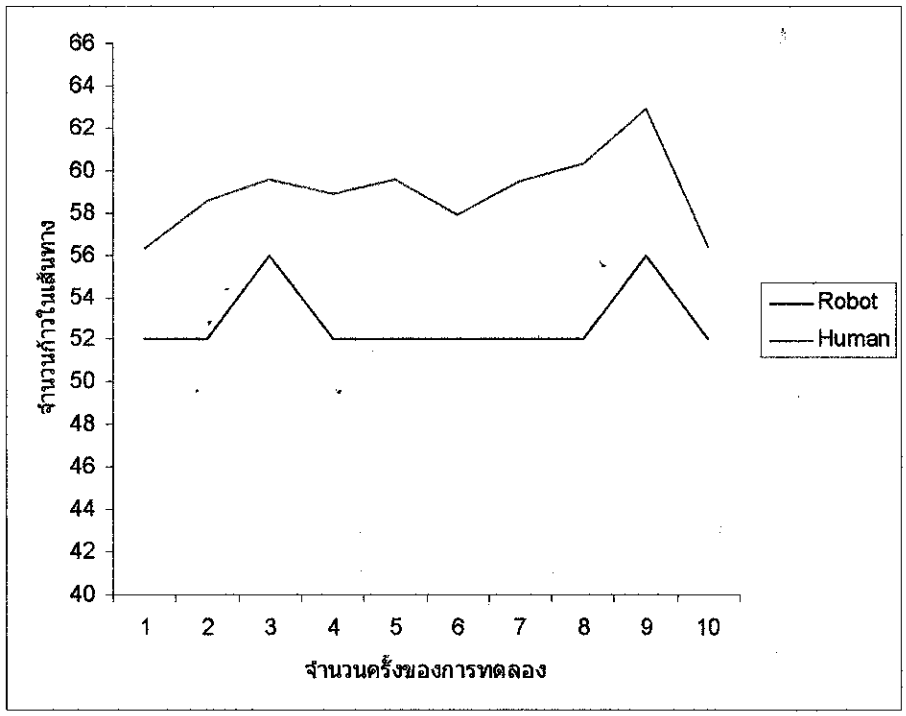
3.1 ตารางแสดงผลการทดลอง กรณีที่1 กริด 25X25 แบบมีสิ่งกีดขวาง

จำนวนครั้ง	จำนวนก้าว											
	หุ่นยนต์	มนุษย์										
		1	2	3	4	5	6	7	8	9	10	ค่าเฉลี่ย
1	52	ตัน	58	52	54	56	ตัน	ตัน	56	62	ตัน	56.33
2	52	ตัน	54	ตัน	ตัน	56	56	ตัน	58	ตัน	69	58.6
3	56	86	56	ตัน	56	56	58	58	58	56	52	59.55
4	52	55	58	60	58	56	56	62	58	ตัน	67	58.88
5	52	56	56	ตัน	58	56	ตัน	ตัน	64	56	71	59.57
6	52	56	66	52	58	56	ตัน	58	ตัน	52	65	57.85
7	52	56	68	ตัน	56	58	58	ตัน	ตัน	ตัน	61	59.5
8	52	58	68	ตัน	ตัน	56	ตัน	66	ตัน	58	56	60.33
9	56	58	70	ตัน	58	56	58	78	78	54	56	62.88
10	52	56	58	58	58	56	ตัน	ตัน	56	54	55	56.375



รูปที่ 3.10 การแสดงการเปรียบเทียบเปอร์เซ็นต์การทดลองที่ไปไม่ถึงจุดหมายกับ RLและมนุษย์

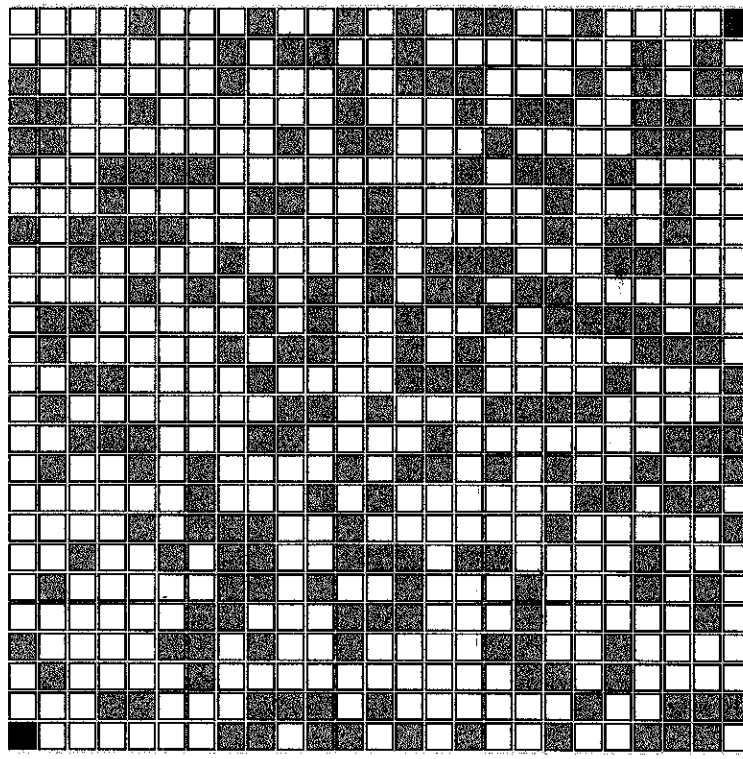
จากรูปที่ 3.10 เมื่อทำการทดลองให้ RL มีการเรียนรู้ตั้งแต่เริ่มต้นจนกระทั่งครบ 3,000 ครั้ง จากนั้นคิดเป็นเปอร์เซ็นต์การทดลองที่ไปไม่ถึงจุดหมายเทียบกับมนุษย์ที่มีการทดลองทั้งหมด 10 คน โดยใช้ กริดแบบเดียวกัน จำนวนคนละ 10 ครั้งซึ่งดูได้จากตารางแสดงผลการทดลอง กรณีที่1 กริด 25X25 แบบมีสิ่งกีดขวางพบว่าเปอร์เซ็นต์การทดลองที่ไปไม่ถึงจุดหมายของ RL นั้นน้อยกว่ามนุษย์เราสามารถอธิบายได้ว่า RL สามารถที่จะเรียนรู้และหาเส้นทางเพื่อไปถึงจุดหมายได้มากกว่ามนุษย์



รูปที่ 3.11 แสดงการเปรียบเทียบระหว่างจำนวนก้าวในเส้นทางกับจำนวนครั้งของการทดลอง

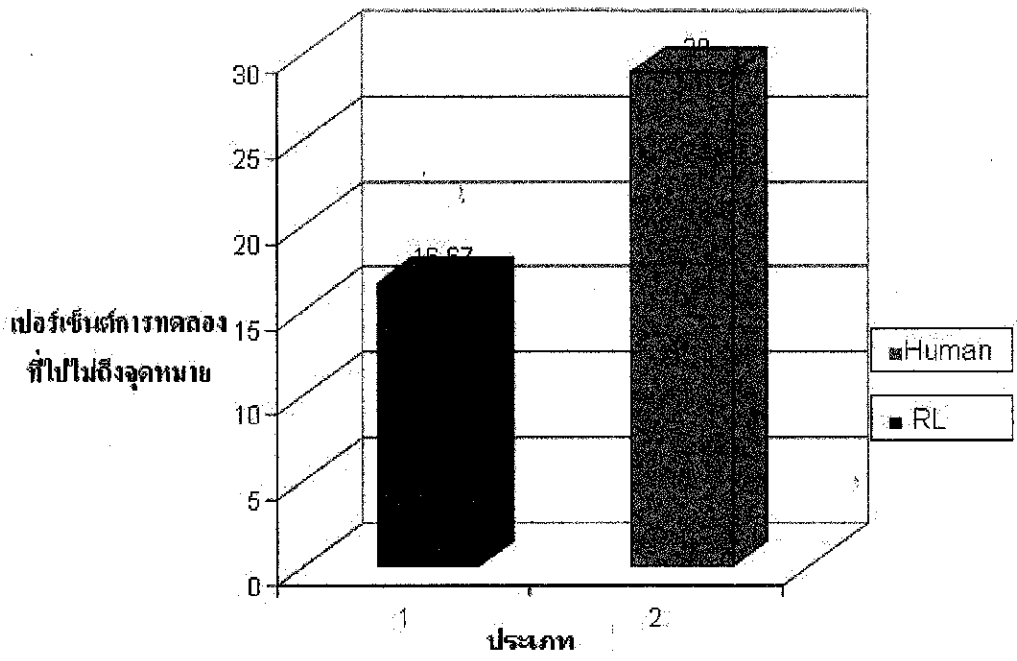
จากรูปที่ 3.11 เมื่อทำการทดลองให้ RL มีการเรียนรู้ไประยะหนึ่งประมาณ 3,000 ครั้ง จากนั้นเราเริ่มนับจาก 3,000 ถึง 3,009 ครั้งเพื่อนำมาเทียบกับผลการทดลองของมนุษย์ซึ่งดูได้จากตารางแสดงผลการทดลอง กรณีที่1 กริด 25X25 แบบมีสิ่งกีดขวาง เปรียบเทียบจำนวนก้าวที่ไปถึงจุดหมายของมนุษย์และ RL พบว่า RL สามารถที่จะไปถึงจุดหมายด้วยจำนวนก้าวที่สั้นกว่าที่มนุษย์ไปถึงบอกได้ว่า RL นั้นมีการเรียนรู้ในการหาเส้นทางที่สั้นที่สุดในสภาวะที่ซับซ้อนได้ดีกว่ามนุษย์

กรณีที่ 2 กริด 25X25 แบบมีสิ่งกีดขวาง

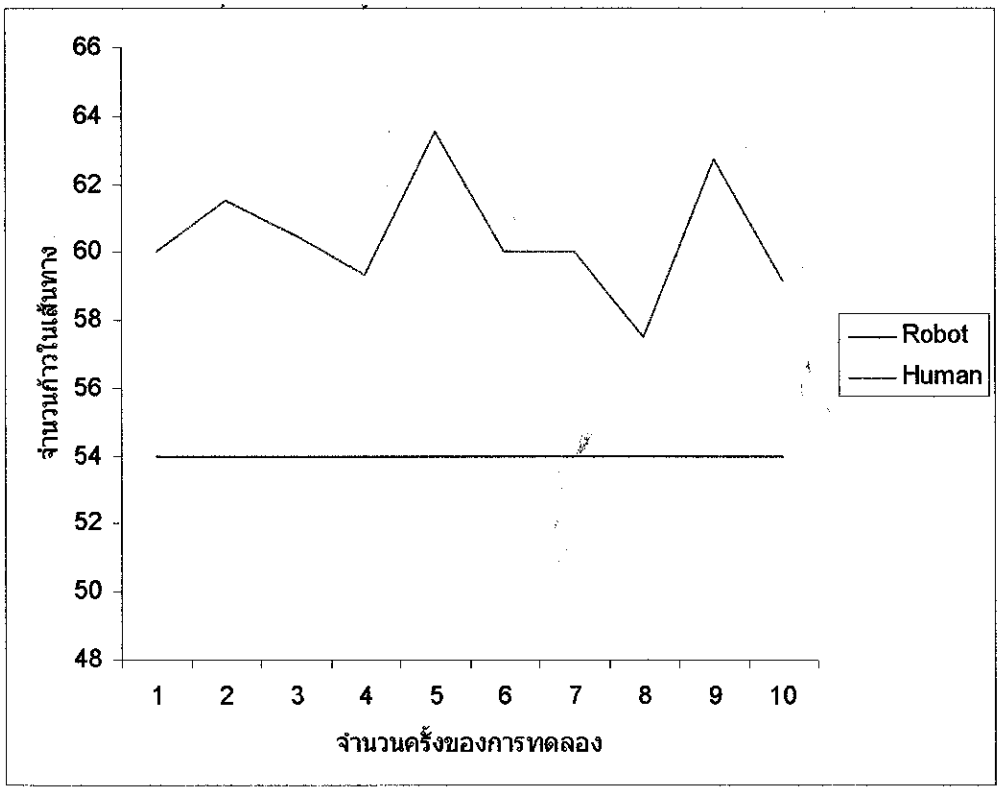


3.2 ตารางแสดงผลการทดลอง กรณีที่ 2 กริด 25X25 แบบมีสิ่งกีดขวาง

จำนวนครั้ง	จำนวนก้าว										
	หุ่นยนต์	มนุษย์									
		1	2	3	4	5	6	7	8	9	10	ค่าเฉลี่ย
1	54	62	64	ตัน	62	ตัน	58	ตัน	58	ตัน	56	60
2	54	ตัน	56	62	58	60	58	ตัน	58	86	54	61.5
3	54	72	54	ตัน	54	58	ตัน	66	ตัน	ตัน	59	60.5
4	54	54	62	58	60	58	ตัน	54	64	70	54	59.33
5	54	ตัน	ตัน	ตัน	54	58	78	63	64	64	ตัน	63.5
6	54	60	58	ตัน	ตัน	58	60	72	60	54	58	60
7	54	ตัน	78	ตัน	54	52	54	68	ตัน	ตัน	54	60
8	54	61	54	ตัน	54	56	66	ตัน	ตัน	ตัน	54	57.5
9	54	58	54	ตัน	74	54	54	54	100	ตัน	54	62.75
10	54	54	64	ตัน	ตัน	54	60	ตัน	60	58	64	59.14

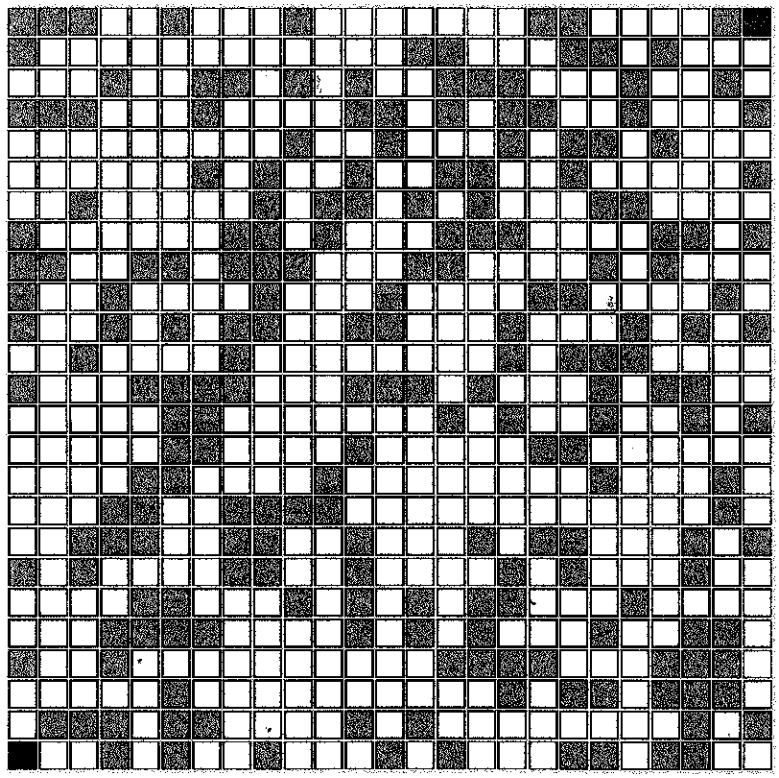


รูปที่ 3.12 การแสดงการเปรียบเทียบเปอร์เซ็นต์การทดลองที่ไปไม่ถึงจุดหมายกับ RLและมนุษย์



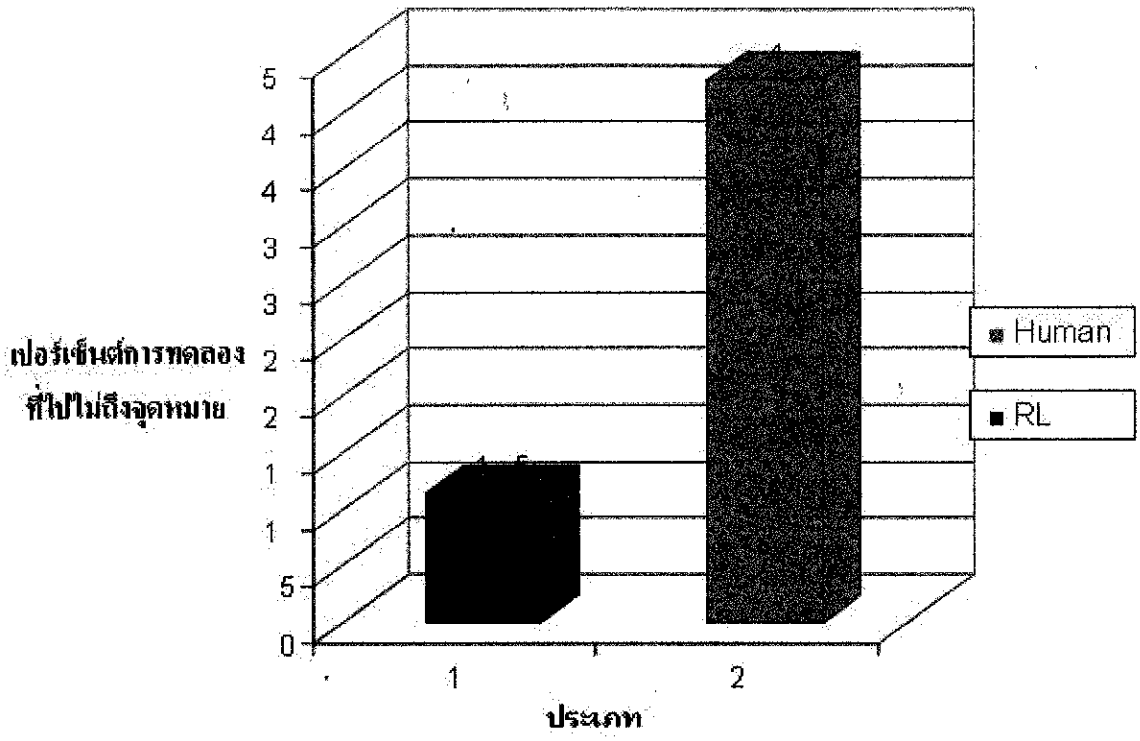
รูปที่ 3.13 แสดงการเปรียบเทียบระหว่างจำนวนก้าวในเส้นทางกับจำนวนครั้งของการทดลอง

กรณีที่ 3 กริด 25X25 แบบมีสิ่งกีดขวาง

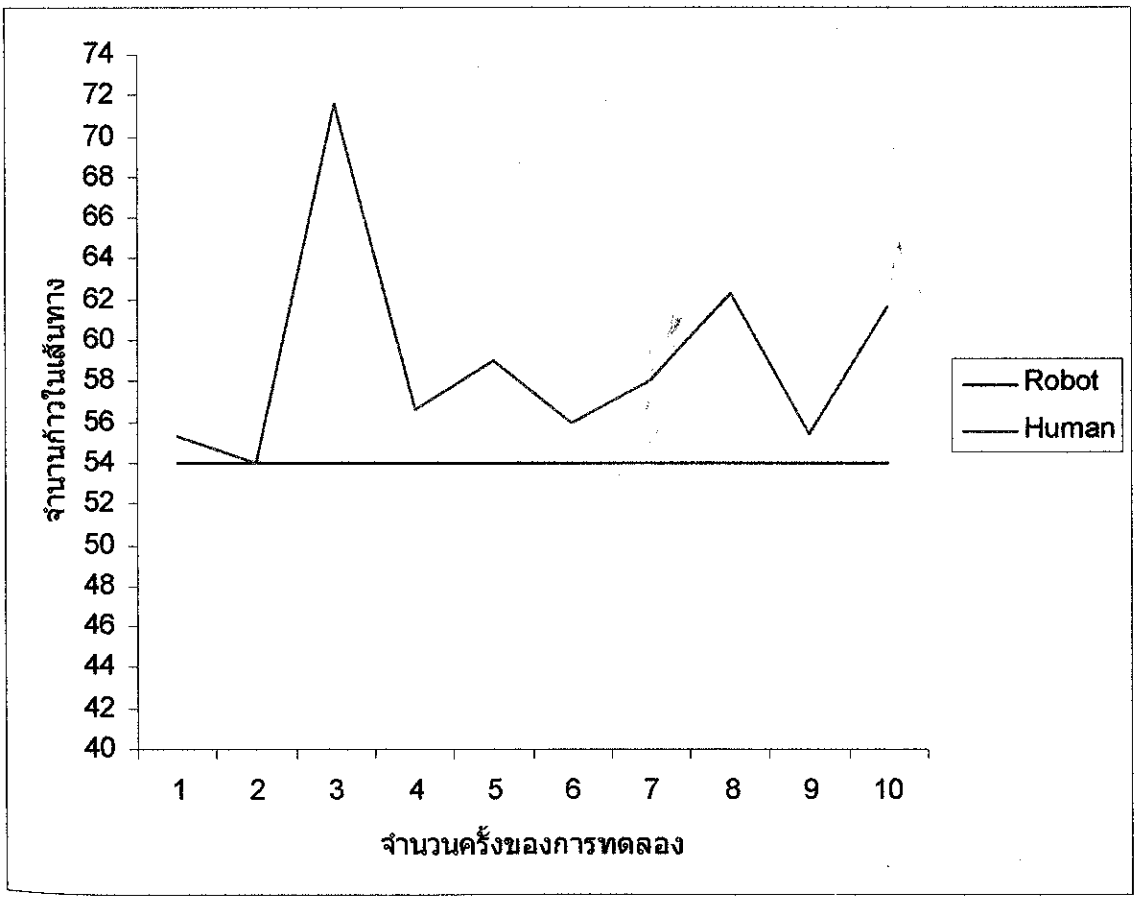


3.3 ตารางแสดงผลการทดลอง กรณีที่ 3 กริด 25X25 แบบมีสิ่งกีดขวาง

จำนวนครั้ง	จำนวนก้าว										
	หุ่นยนต์	มนุษย์									
		1	2	3	4	5	6	7	8	9	10	ค่าเฉลี่ย
1	54	ตัน	ตัน	54	54	58	ตัน	ตัน	ตัน	ตัน	ตัน	53.33
2	54	ตัน	54	ตัน	ตัน	54	ตัน	ตัน	ตัน	ตัน	54	54
3	54	ตัน	54	ตัน	58	56	ตัน	ตัน	ตัน	82	108	71.6
4	54	ตัน	ตัน	ตัน	ตัน	54	ตัน	ตัน	ตัน	58	58	56.67
5	54	ตัน	54	ตัน	56	54	60	ตัน	ตัน	62	68	59
6	54	58	58	ตัน	ตัน	54	54	ตัน	ตัน	ตัน	56	56
7	54	54	62	58	ตัน	54	ตัน	60	64	58	54	58
8	54	54	62	60	ตัน	54	ตัน	ตัน	54	60	92	62.29
9	54	54	ตัน	54	54	54	54	ตัน	ตัน	ตัน	60	55.4
10	54	54	ตัน	80	ตัน	54	ตัน	ตัน	62	54	66	61.67



รูปที่ 3.14 การแสดงการเปรียบเทียบเปอร์เซ็นต์การทดลองที่ไปไม่ถึงจุดหมายกับ RLและมนุษย์



รูปที่ 3.15 แสดงการเปรียบเทียบระหว่างจำนวนก้าวในเส้นทางกับจำนวนครั้งของการทดลอง

บทที่ 4
ปฏิบัติการสร้างหุ่นยนต์ต้นแบบ

4.1 กล่าวนำ

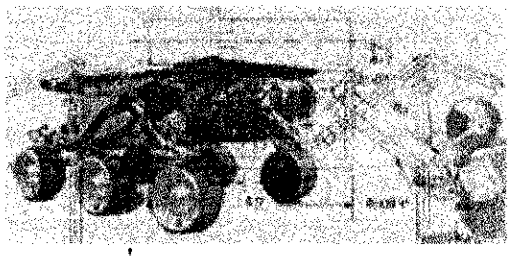
ปัจจุบันเทคโนโลยีมีการเปลี่ยนแปลงไปอย่างรวดเร็วและมากมายโดยเฉพาะด้านอิเล็กทรอนิกส์และคอมพิวเตอร์ ซึ่งเป็นเทคโนโลยีที่เข้ามามีบทบาทเกี่ยวข้องกับชีวิตประจำวันของมนุษย์เรามากยิ่งขึ้นเรื่อยๆ ซึ่งเราได้เล็งเห็นถึงความสำคัญในด้านการพัฒนาองค์ความรู้ทางด้านเทคโนโลยีอิเล็กทรอนิคส์และคอมพิวเตอร์ จึงได้ทำการสร้างหุ่นยนต์สมองกลขึ้นมาเพื่อสำหรับใช้งานจริง และยังเป็นการส่งเสริมและพัฒนาองค์ความรู้ด้านดังกล่าวให้เพิ่มมากยิ่งๆขึ้น ซึ่งการทำหุ่นยนต์ดังกล่าวหัวใจสำคัญที่ทำให้มันสามารถที่จะเคลื่อนไหวได้ คือ การเขียนคำสั่งโปรแกรมไมโครคอนโทลเลอร์ และบร์อดไมโครคอนโทลเลอร์ ซึ่งบร์อดไมโครคอนโทลเลอร์ที่ใช้นี้สามารถใช้กับไมโครคอนโทลเลอร์ตระกูล ปิ๊ก(PIC) ได้หลายเบอร์ ซึ่งถือเป็นไมโครคอนโทลเลอร์ที่มีประสิทธิภาพและใช้งานง่าย สะดวกต่อการนำไปขยายในระบบที่มีประสิทธิภาพสูงขึ้น ทั้งนี้ยังประหยัดเวลาในการสร้างและมีราคาย่อมเยาว์ นอกจากนั้นยังมีชุดบร์อดขยายและบร์อดต่อพ่วงทำให้มีการยืดหยุ่นในการทำงานได้เป็นอย่างดี

4.2 ปฏิบัติการสร้างหุ่นยนต์ต้นแบบ

ส่วนประกอบหลักของหุ่นยนต์สามารถแบ่งออกเป็น 3 ส่วนใหญ่ ๆ ดังต่อไปนี้

1. ส่วนของระบบทางกล (Mechanical part)

ในส่วนนี้จะเป็นส่วนของระบบทางกลทั้งหมดของหุ่นยนต์ เช่น ล้อ ระบบขับเคลื่อน แขนกล มือกล หรือแม้แต่ข้อพับ ข้อเหวี่ยงต่าง ๆ หรืออาจจะเรียกว่าโครง หรือเฟรมของหุ่นยนต์ก็ได้ หากจะเปรียบกับคนแล้ว ก็อาจจะเปรียบเทียบได้กับ โครงกระดูก หรือร่างกายภายนอก เช่น แขน ขา ลำตัว ฯลฯ นั่นเอง

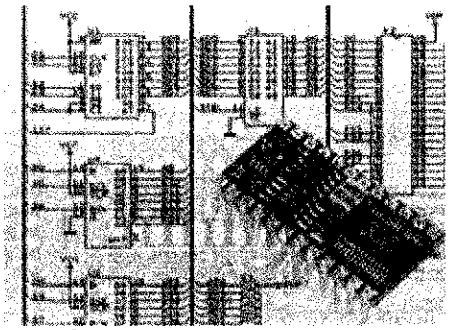


รูปที่ 4.1 ส่วนของระบบทางกล

2. ส่วนของระบบวงจรไฟฟ้า (Electrical Circuit part)

สำหรับส่วนนี้จะเป็นส่วนของระบบวงจรไฟฟ้าทั้งหมดของหุ่นยนต์ ซึ่งในหุ่นยนต์ 1 ตัวนั้น

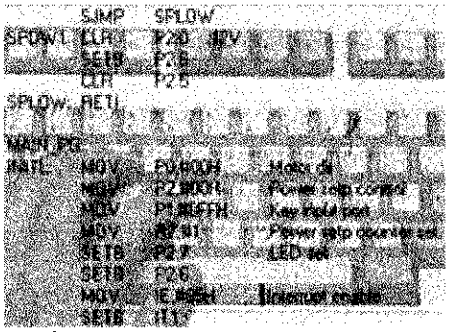
จะประกอบด้วยวงจรต่างๆ หลายวงจรทำงานร่วมกันอยู่ เช่น วงจรควบคุม (Controller circuit) วงจรเซนเซอร์(Sensor circuit) วงจรขับ(Driver circuit) วงจรอินเตอร์เฟส(Interfacing circuit) และวงจรอื่นๆ แล้วแต่ความจำเป็นสำหรับหุ่นยนต์ตัวนั้น ๆ นอกจากนี้ยังรวมไปถึง แหล่งจ่ายพลังงาน และส่วนควบคุม (Control panel) อีกด้วย หากจะว่าไปแล้ว ส่วนของวงจรไฟฟ้านี้อาจเปรียบเทียบได้กับอวัยวะภายในของร่างกายมนุษย์ เช่น หัวใจ ตับ ปอด เส้นเลือด ฯลฯ



รูปที่ 4.2 ส่วนของระบบวงจรไฟฟ้า

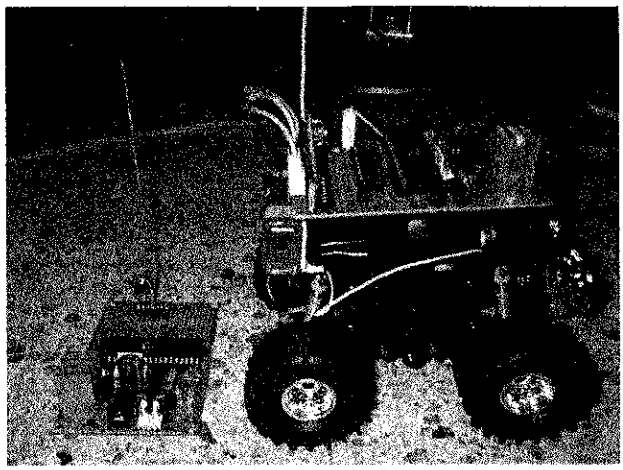
3. ส่วนของโปรแกรมปฏิบัติงาน (Software Control part)

และส่วนนี้ ก็จะเป็นส่วนของโปรแกรมปฏิบัติงาน ถึงแม้ว่าหุ่นยนต์จะได้รับการออกแบบสร้างมาอย่างดีเพียงไรก็ตาม แต่จะยังคงทำงานตามที่เราต้องการไม่ได้แน่ๆ หากยังไม่ได้มีการใส่โปรแกรมที่ถูกต้อง และเหมาะสมให้กับมัน หากจะเปรียบไปแล้วส่วนนี้ก็น่าจะเปรียบเทียบได้กับสติปัญญาของมนุษย์นั่นเอง



รูปที่ 4.3 ส่วนของโปรแกรมปฏิบัติงาน

สำหรับการออกแบบสร้างหุ่นยนต์จะต้องใช้ความรู้ความชำนาญในหลายๆด้าน เพราะเราจะเห็นได้ว่าองค์ประกอบทั้งสามส่วนนั้น มีความสำคัญไม่ยิ่งหย่อนไปกว่ากัน หุ่นยนต์ที่จะมีประสิทธิภาพได้นั้นจะต้องมีการทำงานร่วมกันของส่วนประกอบหลักทั้งสามส่วนที่ได้กล่าวมานั้น สามารถทำงานร่วมกันได้อย่างสอดคล้องและลงตัว

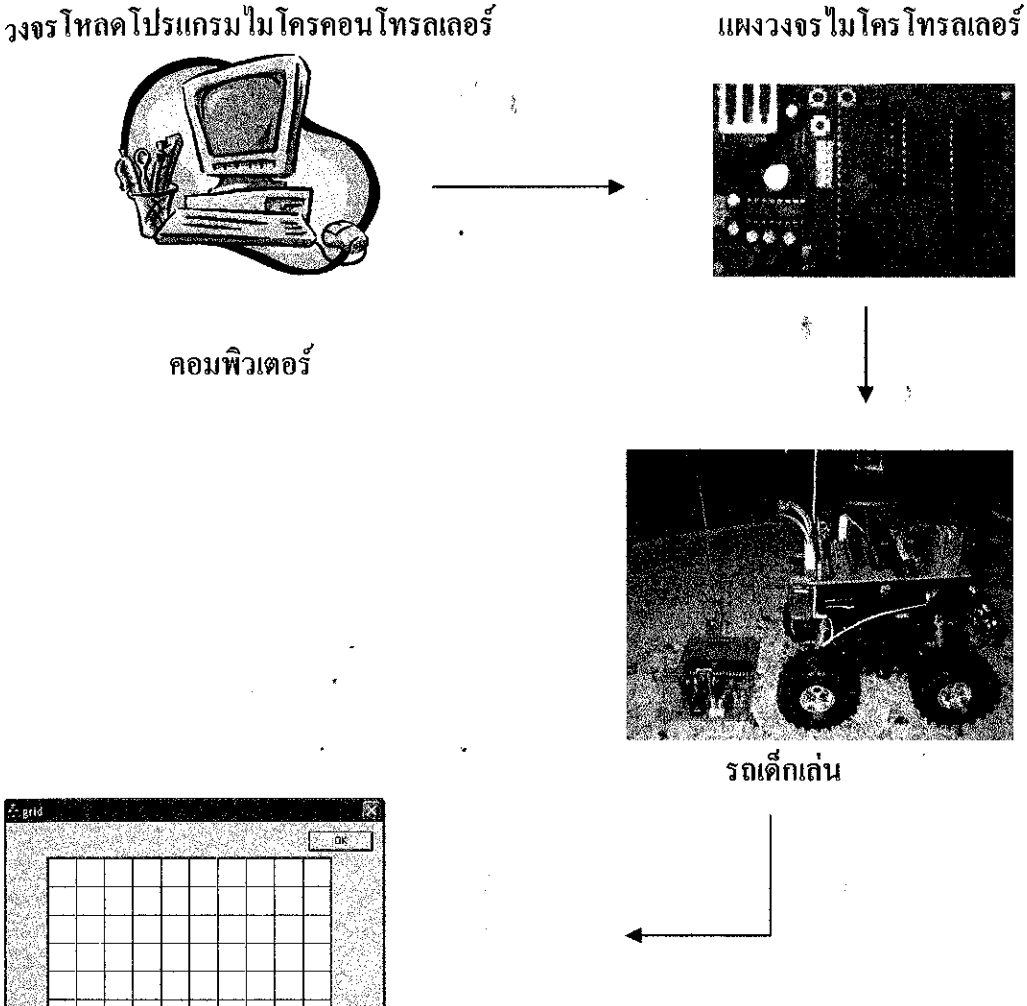


รูปที่ 4.4 รถเด็กเล่นที่ขับเคลื่อนด้วยมอเตอร์

จากรูปโครงสร้างทั่วไปของรถเด็กเล่นประกอบด้วยมอเตอร์สองตัว ตัวแรกสำหรับขับเคลื่อนล้อหลังให้ตัวรถเดินหน้าหรือถอยหลัง ส่วนมอเตอร์อีกตัวสำหรับระบบเลี้ยวซ้ายและเลี้ยวขวาของล้อหน้า ดังนั้นการออกแบบให้รถเด็กเล่นนี้เป็นหุ่นยนต์ที่มีการเคลื่อนที่ตามต้องการจึงจำเป็นจะต้องทำการดัดแปลงส่วนที่ควบคุมมอเตอร์ทั้งสองตัวนี้ ซึ่งการทำงานของหุ่นยนต์รถเด็กเล่นที่จะทำนั้นมีหลักการดังนี้

- หลักการแนวคิดไมโครคอนโทรลเลอร์เพื่อเชื่อมต่อกับรถเด็กเล่น ให้สามารถควบคุมมอเตอร์ด้วยไมโครคอนโทรลเลอร์
- หลักการเขียนโปรแกรมคำสั่งของไมโครคอนโทรลเลอร์เพื่อควบคุมรถให้สามารถวิ่งตามที่เราต้องการได้

โดยปกติแล้ว ในการสร้างวงจรเพื่อทำงานหนึ่งๆ เช่นทำหน้าที่ควบคุมและแสดงผลการใช้งานตู้ไมโครเวฟเป็นต้น วงจรดังกล่าวนี้สามารถสร้างได้จากวงจรดิจิตอลอิเล็กทรอนิกส์ ซึ่งอาจประกอบไปด้วยองค์ประกอบต่างๆ เช่นลอจิกเกต(logic gate) เป็นจำนวนสิบถึงร้อยตัว ขึ้นอยู่กับความซับซ้อนของหน้าที่ของวงจรที่ต้องการเมื่อพิจารณาใช้ไมโครคอนโทรลเลอร์ในงานดังกล่าวจะสามารถลดความยุ่งยากของลอจิกเกตนี้ได้เนื่องจากไมโครคอนโทรลเลอร์ทำงานด้วยซอฟท์แวร์ ไมโครคอนโทรลเลอร์จะทำงานตามคำสั่งในซอฟท์แวร์ซึ่งเขียนขึ้นเป็นโปรแกรมสั้นๆ โปรแกรมเหล่านี้สามารถทำงานแทนลอจิกเกตที่กล่าวมาข้างต้นได้หลายตัว ในหัวข้อต่อไปนี้จะได้กล่าวถึงรายละเอียดในการนำเอาไมโครคอนโทรลเลอร์มาใช้งาน โดยเริ่มต้นจากรายละเอียดของบอร์ดไมโครคอนโทรลเลอร์ที่จะใช้ องค์ประกอบต่างๆของไมโครคอนโทรลเลอร์ การเขียนโปรแกรมสั่งงานไมโครคอนโทรลเลอร์ ไปจนถึงการประยุกต์ใช้งานในการสร้างหุ่นยนต์จากรถเด็กเล่นให้สามารถเดินทางไปตามที่เราต้องการได้อย่างอัตโนมัติ



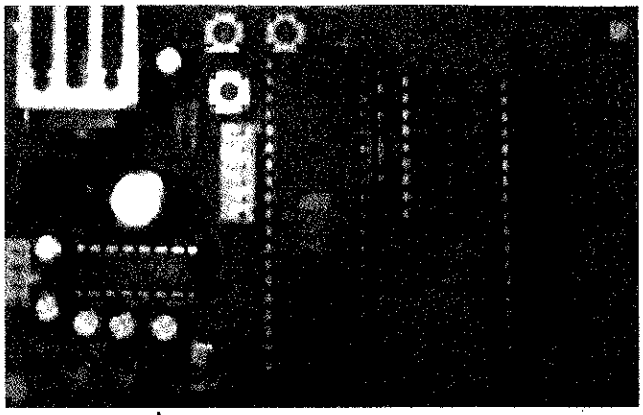
รูปที่ 4.5 โครงงานโดยรวม

4.2.1 รายละเอียดฮาร์ดแวร์

อุปกรณ์ที่ใช้งานในโครงการนี้ประกอบด้วย

1. บอร์ดไมโครคอนโทรลเลอร์
2. รถเด็กเล่นที่ขับเคลื่อนด้วยมอเตอร์

ในหัวข้อต่อไปจะได้กล่าวถึงรายละเอียดของอุปกรณ์ฮาร์ดแวร์ต่างๆ ข้างต้น เพื่อจะนำไปสู่ปฏิบัติการสร้างหุ่นยนต์ต่อไป



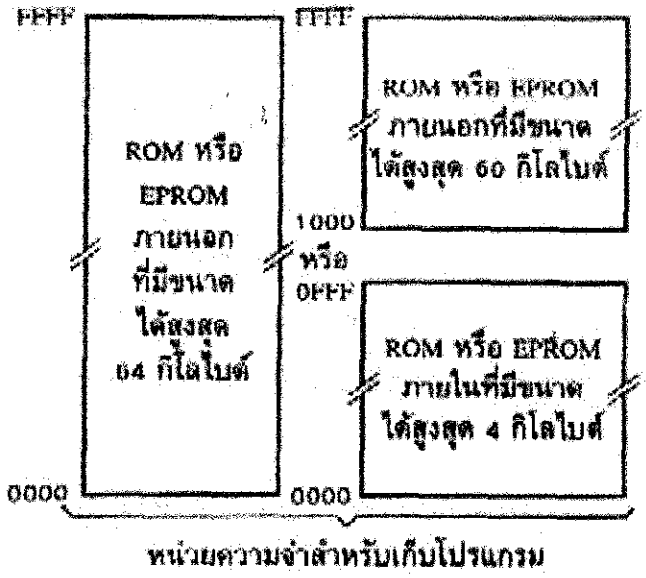
รูปที่ 4.6 บอร์ดไมโครคอนโทรลเลอร์

- บอร์ดไมโครคอนโทรลเลอร์

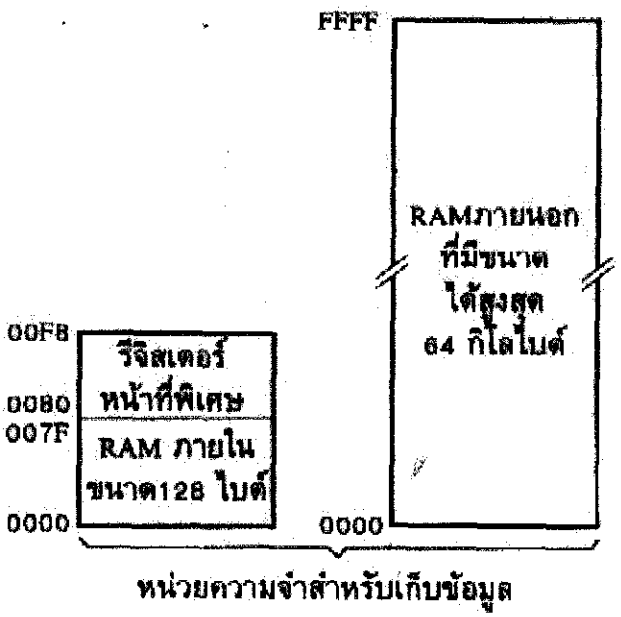
บอร์ดไมโครคอนโทรลเลอร์เป็นบอร์ดควบคุมเดี่ยว (single control boad)กล่าวคือไม่ต้องมีอุปกรณ์ต่อพ่วงใดๆ ก็สามารถใช้งานได้เลย สำหรับในการใช้ไมโครคอนโทรลเลอร์ เพื่อใช้ในงานควบคุมต่างๆนั้นจำเป็นที่จะต้องใช้ เม็มเมอร์รี่(memory) ในการเก็บข้อมูลและใช้การส่งข้อมูลผ่านทางอินพุท และเอาท์พุท (I/O) เพื่อรับข้อมูลเข้ามาประมวณผลและส่งสัญญาณออกไปควบคุมอุปกรณ์ที่ต้องการและเพื่อความสะดวกในการใช้งานตัวไมโครคอนโทรลเลอร์ จึงได้รวมเม็มเมอร์รี่ไว้ภายในตัวชิป ซึ่งจะได้อธิบายส่วนประกอบของเม็มเมอรี่ภายในชิปไมโครคอนโทรลเลอร์และการจัดสรรหน่วยความจำภายใน และ อธิบายถึงระบบการทำงานของ ทางอินพุท และเอาท์พุท โดยจะอ้างอิงถึง ไมโครคอนโทรลเลอร์ขนาด 8 บิต ที่เป็นที่นิยมใช้ในปัจจุบันคือ 8051 และสำหรับไอซีที่สำคัญอีกตัวหนึ่งบนบอร์ดของไมโครคอนโทรลเลอร์ คือไอซี แอล293 ซึ่งเป็นตัวที่ใช้ในการขับเคลื่อนมอเตอร์ นอกจากนี้ยังมีสวิตซ์รีเซตเพื่อใช้สำหรับรีเซตบอร์ดหรือให้โปรแกรมเริ่มทำงานใหม่

- ไอ ซี 8051(IC 8051)

สำหรับ ชิปเบอร์ 8051 จะมีรอม (ROM หรือ EPROM) ขนาด 4 กิโลบิต และแรม(RAM) ขนาด 128 ไบต์ ตัว 8051 สามารถทำการอ้างอิงหน่วยความจำภายนอกได้ ถ้า แรม หรือ รอมภายในนั้นมีขนาดไม่เพียงพอ โดยจะใช้ พอร์ต 2 ตัวในการทำหน้าที่อ้างอิงข้อมูลกับหน่วยความจำภายนอกซึ่งการทำเช่นนี้จะส่งผลให้พอร์ตที่ทำหน้าที่เป็น อินพุท และเอาท์พุท มีจำนวนลดลง แต่ 8051 จะมีความยืดหยุ่นในการทำงานมากขึ้น



รูปที่ 4.7 แสดงถึงหน่วยความจำสำหรับโปรแกรมที่อ่านข้อมูลได้อย่างเดียว

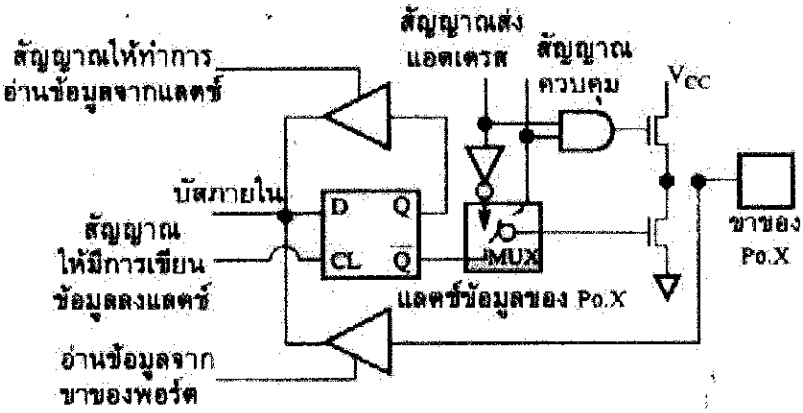


รูปที่ 4.8 แสดงถึงหน่วยความจำที่อ่านและเขียนข้อมูลลงไปได้

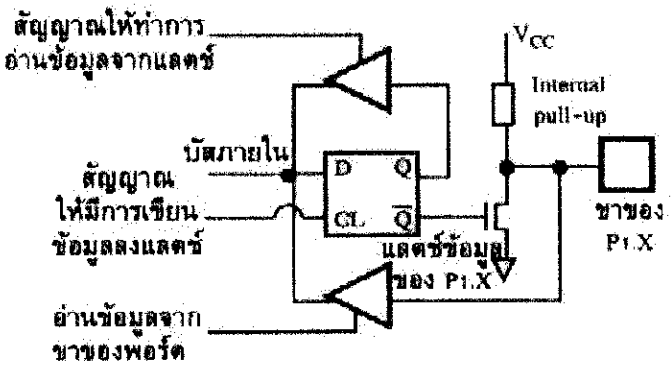
จากรูปแสดงแผนภาพการจัดสรรหน่วยความจำ-รีจิสเตอร์ (memory-register map) ของ 8051 จากรูปเราจะเห็นได้ว่า 8051 มี รีจิสเตอร์หน้าที่พิเศษรวมอยู่ภายในด้วย สำหรับส่วนของ แรม และ รอม ภายนอกนั้นเป็นส่วนที่เราต้องทำการเพิ่มเข้าไปเอง แผนภาพการจัดสรรหน่วยความจำรีจิสเตอร์ยังแสดงถึงคุณสมบัติอีกอย่างหนึ่งของ 8051 ที่ทำหน้าที่แตกต่างจาก ไมโครโปรเซสเซอร์ทั่วๆไป โดย 8051 นี้จะมีหน่วยความจำอยู่ 2 ประเภท ประเภทแรกเป็นหน่วยความจำสำหรับเก็บโปรแกรม และอีกประเภทเป็นหน่วยความจำสำหรับเก็บข้อมูลที่เปลี่ยนได้ หน่วยความจำสำหรับ

เก็บโปรแกรมนั้นจะเป็นหน่วยความจำที่เราสามารถทำการอ่านข้อมูลได้อย่างเดียว เราอ่านคำสั่งของโปรแกรมได้แต่ไม่สามารถสั่งให้หน่วยประมวณผลทำการเขียนข้อมูลลงในหน่วยความจำประเภทนี้ได้ ซึ่งเราเรียกหน่วยความจำประเภทนี้ว่า ROM โดย ROM ภายในของ 8051 ถูกกำหนดให้เป็นหน่วยความจำสำหรับเก็บโปรแกรม และ ไมโครโปรเซสเซอร์จะทำการ Fetch คำสั่งทุกคำสั่งจากหน่วยความจำประเภทนี้เท่านั้น หน่วยความจำสำหรับเก็บข้อมูลเป็นหน่วยความจำประเภทที่สามารถทำการอ่านและเขียนข้อมูลได้ หน่วยประมวลผลสามารถทำการอ่านข้อมูลและเขียนข้อมูลลงในหน่วยความจำประเภทนี้ แต่ไม่สามารถทำงานตามคำสั่งของโปรแกรมที่อยู่ในหน่วยความจำประเภทนี้ได้ เราจะพบว่าแรมภายในของ 8051 จะถูกกำหนดให้เป็นหน่วยความจำสำหรับเก็บข้อมูล

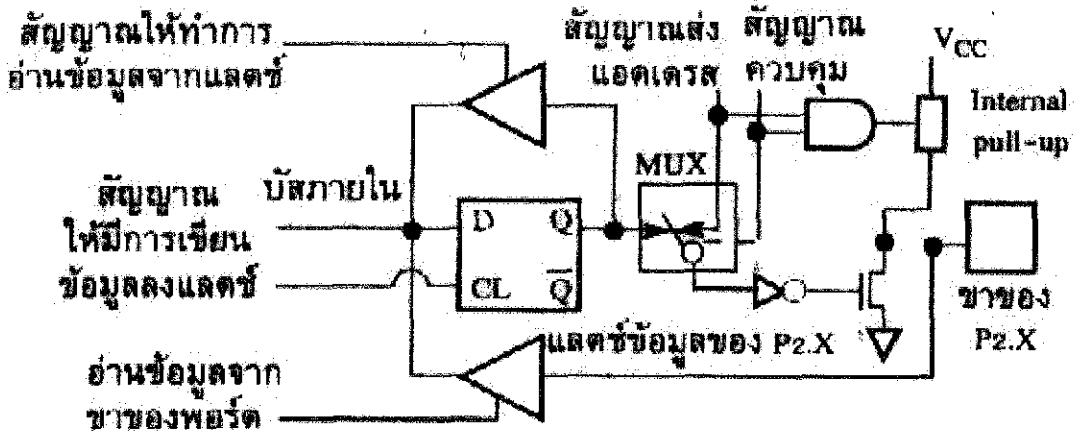
ระบบอินพุท และเอาท์พุท ของ 8051 จากองค์ประกอบของชิป 8051 เราจะพบว่ามีพอร์ตอินพุท และเอาท์พุท ที่รับส่งข้อมูลได้ 2 ทิศทางขนาด 8 บิตอยู่ 4 พอร์ต ซึ่งพอร์ตแต่ละตัวนี้จะมีคุณสมบัติพิเศษเฉพาะตัว



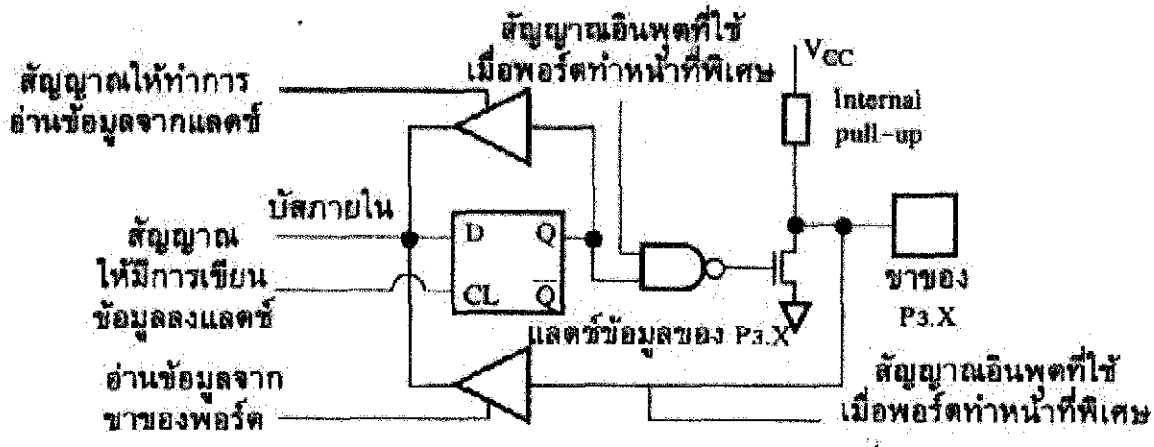
รูปที่ 4.9 แสดงแผนผังทาง ลอจิกสำหรับบิตหนึ่งๆ ในพอร์ต P0



รูปที่ 4.10 แสดงแผนผังทางลอจิก สำหรับบิตหนึ่งๆ ในพอร์ต P1



รูปที่ 4.11 แสดงแผนผังทางลอจิก สำหรับบิตหนึ่งๆ ในพอร์ต P2



รูปที่ 4.12 แสดงแผนผังทางลอจิก สำหรับบิตหนึ่งๆ ในพอร์ต P3

ดังรูปที่ 4.9 ถึง รูปที่ 4.12 ซึ่งแสดงแผนภาพทางตรรกะสำหรับบิตหนึ่งๆ ในพอร์ตแต่ละตัว เราจะเห็นได้ว่าพอร์ตเหล่านี้ (P0-P3) มีคุณสมบัติทางลอจิกและทางไฟฟ้าที่แตกต่างกันเล็กน้อย พอร์ตแต่ละตัวจะมีแลตช์ข้อมูล ซึ่งจะทำหน้าที่เก็บข้อมูลที่เข้าหรือออกจากพอร์ต โดยแลตช์ข้อมูลนี้สามารถนำข้อมูลจากขาของพอร์ตหรือจากบัสข้อมูลของไมโครโปรเซสเซอร์เข้ามาเก็บได้ และแลตช์ข้อมูลนี้ยังสามารถทกการส่งข้อมูลไปยังบัสข้อมูลของไมโครโปรเซสเซอร์หรือไปยังขาของพอร์ตได้ ถ้ามีการเชื่อมต่อ 8051 กับหน่วยความจำภายนอก (ซึ่งอาจเป็น ROM หรือ RAM) เราจะนำพอร์ตเบอร์ 0 และพอร์ตเบอร์ 2 มาใช้ในการเก็บค่าแอดเดรสที่ใช้อ้างอิงหน่วยความจำนี้ นอกจากนี้เรายังจะนำพอร์ตเบอร์ 0 มาใช้ในการแลกเปลี่ยนข้อมูลกับหน่วยความจำนี้ด้วย ซึ่งหมายความว่าพอร์ตเบอร์ 0 จะมีถึง 2 หน้าที่ หน้าที่แรกของพอร์ตเบอร์ 0 คือ ทำการส่งค่าไบต์ล่างของแอดเดรส

ขนาด 16 บิต ที่เป็นตำแหน่งของหน่วยความจำภายนอกที่ต้องการอ้างอิง หน้าที่ที่สองคือ ทำการรับข้อมูลจากหน่วยความจำหรือส่งข้อมูลไปยังหน่วยความจำที่อ้างอิง

เราสามารถนำพอร์ตเบอร์ 3 มาทำงานเป็นพอร์ตอินพุทและเอาท์พุทสองทิศทางแบบขนานได้ดังพอร์ต เบอร์ 0 และพอร์ตเบอร์ 2 และเรายังสามารถนำพอร์ตเบอร์ 3 นี้มาใช้งานในหน้าที่พิเศษอื่นๆ ได้ด้วย ดังตารางด้านล่างแสดงหน้าที่ต่างๆของแต่ละบิตในพอร์ตเบอร์ 3 ในโหมการทำงานปกติ และโหมดการทำงานพิเศษ เราสามารถนำพอร์ตเบอร์ 3 มาทำงานเป็นพอร์ต อินพุท และเอาท์พุทสองทิศทางแบบขนานได้ดังพอร์ต เบอร์ 0 และพอร์ตเบอร์ 2 และเรายังสามารถนำพอร์ตเบอร์ 3 นี้มาใช้งานในหน้าที่พิเศษอื่นๆ ได้ด้วย ดังตารางด้านล่างแสดงหน้าที่ต่างๆของแต่ละบิตในพอร์ตเบอร์ 3 ในโหมการทำงานปกติ และโหมดการทำงานพิเศษ

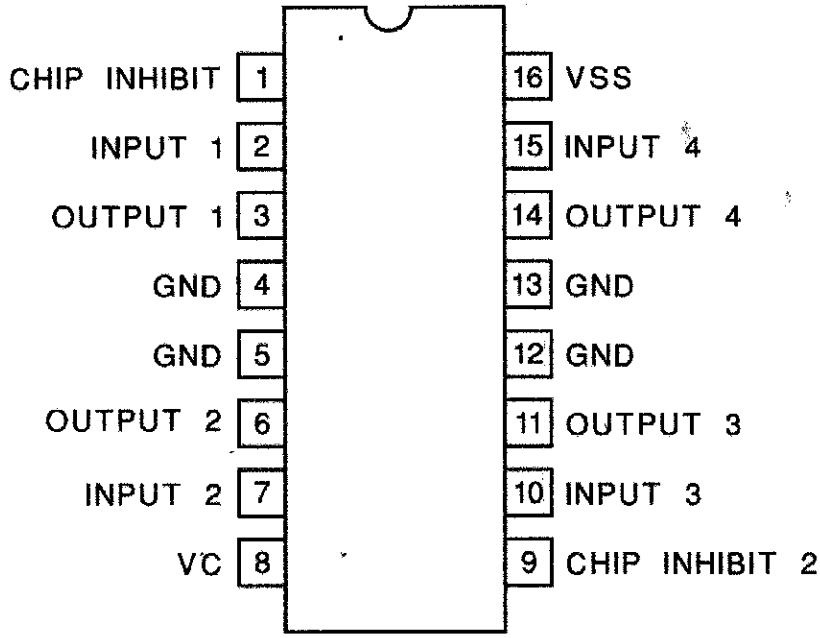
เบอร์ขาของพอร์ต	หน้าที่ปกติ	หน้าที่พิเศษ
P3.0	บิต 0	RXD (พอร์ตอนุกรมที่ทำหน้าที่รับข้อมูล)
P3.1	บิต 1	TXD (พอร์ตอนุกรมที่ทำหน้าที่ส่งข้อมูล)
P3.2	บิต 2	(INT0)' (สัญญาณอินเตอร์รัปต์ภายนอกเบอร์ 0)
P3.3	บิต 3	(INT1)' (สัญญาณอินเตอร์รัปต์ภายนอกเบอร์ 1)
P3.4	บิต 4	T0 (สัญญาณอินพุตเบอร์ 0 ที่ป้อนให้กับเคาน์เตอร์ไทเมอร์)
P3.5	บิต 5	T1 (สัญญาณอินพุตเบอร์ 1 ที่ป้อนให้กับเคาน์เตอร์ไทเมอร์
P3.6	บิต 6	(WR)' (สัญญาณให้ทำการเขียนข้อมูลลงหน่วยความจำภายนอกที่ใช้เก็บข้อมูล)
P3.7	บิต 7	(RD)' (สัญญาณให้ทำการอ่านข้อมูลจากหน่วยความจำภายนอกที่ใช้เก็บข้อมูล)

รูปที่4.13 ตารางแสดงหน้าที่พิเศษของขาต่างๆในพอร์ตเบอร์ 3

ไมโครคอนโทรลเลอร์ มีการรวมเอา รอม และแรมเข้าไปไว้ในตัวเองเพื่อความสะดวกในการใช้งานทางด้านการควบคุม โดยที่ตัวไมโครคอนโทรลเลอร์ยังสามารถที่จะต่อ รอม และแรมภายนอกได้ เพื่อให้มีความยืดหยุ่นในการทำงานมากขึ้นในกรณีที่งานที่ใช้ต้องใช้หน่วยความจำเป็นจำนวนมาก ในส่วนของ อินพุท และอาท์พุทนั้นในตัวไมโครคอนโทรลเลอร์ ได้มีจำนวนพอร์ตที่จะให้ใช้ในการควบคุมเป็นจำนวนมากพอ ดังเช่นของ 8051 จะมีให้ใช้ถึง 4 พอร์ต

- ไอซี แอล293(IC L293D)

ซึ่งไอซี แอล293เป็นอุปกรณ์ที่ใช้ในการขับมอเตอร์เพื่อให้หุ่นยนต์สามารถเดินหน้า ถอยหลัง เลี้ยวซ้าย และเลี้ยวขวา ได้ตามต้องการ



รูปที่ 4.14 ภาพประกอบ ไอซี แอล 293ดี

V_{ss} = แหล่งจ่ายแรงดัน (Logic Supply) = 4.5V to 36V

V_c = แหล่งจ่ายขับเคลื่อนมอเตอร์ (Motor Driver Supply) = V_{ss} to 36V

ค่ากำลังงานรวม (Total Power Dissipation) $T_{ground\text{-}pins}$ = 5 W

เราสามารถหาค่ากระแสสูงสุดได้จาก

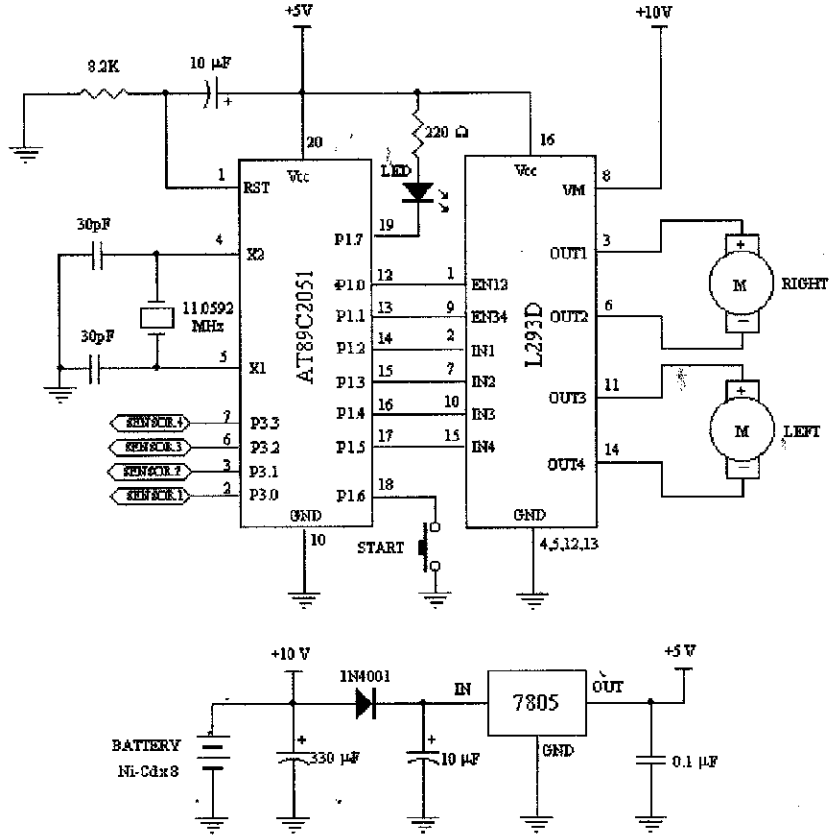
V = 3.6 Volts และ P = 5 Watts

จะได้ว่า

P = VI (Power = Volts x Current)

I = P / V = 5 / 3.6 = 1.4 Amps

เพราะฉนั้นประสิทธิภาพสูงสุดของกระแสในการขับมอเตอร์ เท่ากับ 0.8 แอมป์



รูปที่4.15 ภาพแสดงการต่อขาไอซีเพื่อควบคุมมอเตอร์

- ไมโครคอนโทรลเลอร์ เอ ที89S8252

ไมโครคอนโทรลเลอร์ เอ ที 89S8252 เป็นตัวกลางในการควบคุมสำหรับโปรแกรมที่ทำการโหลดลงไปบนบอร์ดไมโครคอนโทรลเลอร์

- ไอ ซี เร็กกูเลเตอร์ 7805

ไอ ซี เร็กกูเลเตอร์ 7805 ใช้สำหรับควบคุมแรงดัน 5โวลต์ เพื่อเป็นแหล่งจ่ายให้กับวงจร

- ไอ ซี แม็ก 232

ไอ ซี แม็ก 232 เป็นตัวกลางในการทำงานร่วมกันระหว่างสัญญาณของคอมพิวเตอร์ และของไมโครคอนโทรลเลอร์

สำหรับอุปกรณ์อิเล็กทรอนิกส์ต่างๆ เช่น ตัวเก็บประจุ ตัวต้านทาน และไดโอด เป็นต้นจะเป็นอุปกรณ์เสริมช่วยให้ ไอซี เอ ที 89S8252 สามารถทำงานได้

ตัวตรวจรู้ (sensor)

ตัวตรวจรู้ หรือ เซ็นเซอร์เป็นอุปกรณ์ที่ตรวจจับหรือรับสัญญาณ เช่น ความร้อน แสง การเคลื่อนไหว แล้วให้ผลตอบสนองเช่นกระแสไฟฟ้าหรือความต้านทาน ในลักษณะรูปแบบที่แน่นอน ตัวอย่างของเซ็นเซอร์ที่มีใช้กันทั่วไป เช่น

1.เซ็นเซอร์แสง เป็นเซ็นเซอร์ที่ใช้ตรวจจับแสง ยกตัวอย่าง เช่น โฟโตเซลล์ โฟโตไดโอด และโฟโตทรานซิสเตอร์ เป็นต้น สำหรับโฟโตไดโอดเป็นอุปกรณ์ที่คุ้นเคยกันดี นั่นคือโฟโตไดโอดหลายๆตัวถูกประกอบสร้างเป็นเซลล์แสงอาทิตย์ (sola cell) นั่นเอง โฟโตไดโอดเป็นไดโอดที่ทำงานด้วยแสง คือเมื่อมีแสงมาตกกระทบที่ตัวโฟโตไดโอด จะทำให้เกิดการนำกระแสขึ้น และถ้าไม่มีแสงมาตกกระทบจะไม่มีการไหลหรือมีการไหลของกระแสที่น้อยมากเมื่อเทียบกับการที่มีแสงตกกระทบ เช่นเดียวกับโฟโตทรานซิสเตอร์ ซึ่งก็คือทรานซิสเตอร์ที่ทำงานเมื่อมีแสงมาตกกระทบนั่นเอง

2.เซ็นเซอร์เสียง (sound sensor) เป็นที่รู้จักกันดีคือไมโครโฟน ซึ่งมีหน้าที่ในการแปลงสัญญาณเสียงเป็นสัญญาณไฟฟ้า

3.เซ็นเซอร์อุณหภูมิ (temperature sensor) เช่น เทอร์โมมิเตอร์หรือเทอร์มิสเตอร์ สำหรับเทอร์โมมิเตอร์ชนิดที่คุ้นเคยกันก็คือเทอร์โมมิเตอร์แบบปรอท ซึ่งทำงานโดยขยายหรือหดตัวของปรอทที่อยู่ในขดแก้วตามอุณหภูมิที่เปลี่ยนไป พร้อมกับทำเสกลเพื่ออ่านอุณหภูมิปัจจุบันมีเทอร์โมมิเตอร์แบบอิเล็กทรอนิกส์ที่ซึ่งทำจากวัสดุที่ให้ค่าความต้านทานที่เปลี่ยไปตามอุณหภูมิ ส่วนเซ็นเซอร์อุณหภูมิ แบบเทอร์มิสเตอร์นั้นเป็นวัสดุจำพวกตัวต้านทานที่ซึ่งมีค่าความต้านทานเปลี่ยนแปลงไปตามอุณหภูมิ ปกติแบ่งได้สองแบบ คือ ความต้านทานเพิ่มเมื่ออุณหภูมิเพิ่ม และแบบที่ความต้านทานลดเมื่ออุณหภูมิเพิ่ม

4.เซ็นเซอร์มนุษย์ (human sensor) คือประสาทของมนุษย์ อันได้แก่ การมองเห็น การรับรู้การดม การได้ยิน และการสัมผัส มนุษย์มีอวัยวะหรือเซนเซอร์ที่รับสัญญาณจากภายนอกตามประสาทสัมผัส กล่าวคือรับภาพด้วยเซ็นเซอร์ตา รับรสชาติด้วยเซ็นเซอร์ลิ้น รับเสียงด้วยเซ็นเซอร์หู และรับสัมผัสด้วยเซ็นเซอร์ผิวหนัง สัญญาณภาพ รสชาติ กลิ่น เสียง และสัมผัสจะถูกแปลงเป็นสัญญาณที่สมองรับรู้และเข้าใจต่อไป

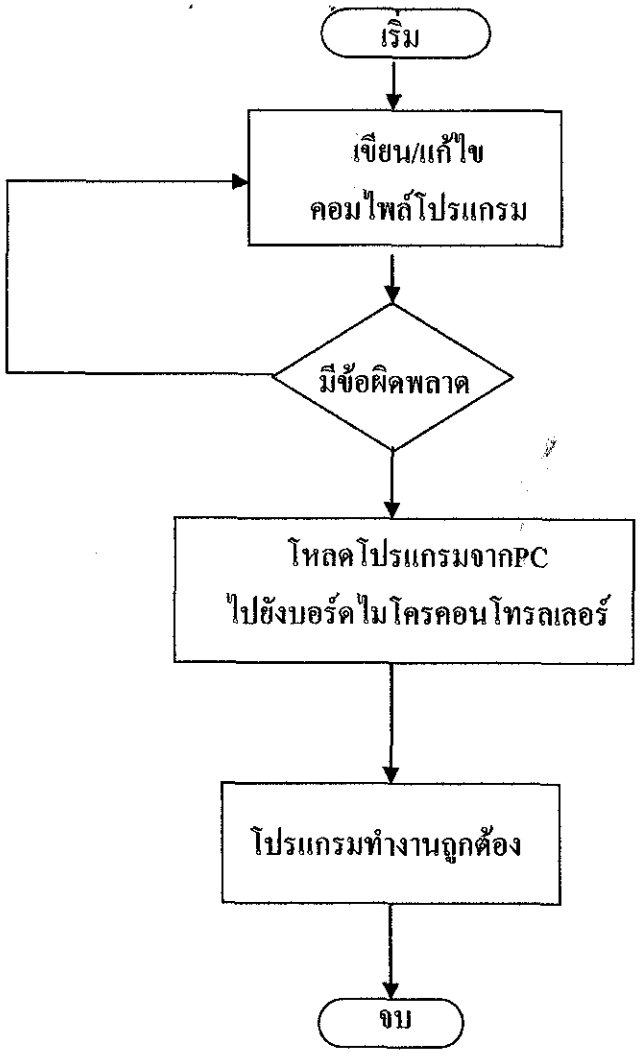
จะเห็นได้ว่าเซ็นเซอร์ก็เปรียบเสมือนกับประสาทสัมผัสของอุปกรณ์ที่เราต้องการสร้างนั่นเอง สำหรับในกรณีการสร้างหุ่นยนต์สมองกลนี้จำเป็นต้องออกแบบให้หุ่นยนต์รับรู้เส้นทางในการเดินด้วยเซ็นเซอร์ชนิดใดชนิดหนึ่ง แล้วทำการส่งข้อมูลจากเซ็นเซอร์นั้นให้กับสมอง ซึ่งก็คือไมโครโปรเซสเซอร์ทำการคำนวณ และตัดสินใจต่อไป

4.2.1 รายละเอียดซอฟแวร์

ปัจจุบันในอุปกรณ์เครื่องใช้ไฟฟ้าอีเล็กทรอนิกส์เกือบทุกชนิด ไม่ว่าจะเป็นเครื่องปรับอากาศ เครื่องซักผ้า วิทยุ โทรทัศน์ รถยนต์ ฯลฯ ต่างมีไมโครคอนโทรลเลอร์เป็นตัวควบคุมการทำงาน (Controller) ของอุปกรณ์ต่างๆ หรือขบวนการต่างๆ สำหรับโครงงานนี้สิ่งที่เป็นหัวใจสำคัญที่จะทำให้หุ่นยนต์สามารถเคลื่อนไหวได้มาจากการเขียนโปรแกรมไมโครคอนโทรลเลอร์เพื่อทำการสั่งการทำงานของหุ่นยนต์ให้เป็นไปตามที่ต้องการ

ไมโครคอนโทรลเลอร์ คือ อุปกรณ์ประเภทสารกึ่งตัวนำที่รวบรวมฟังก์ชั่นการทำงานต่างๆ ไว้ภายในตัวของมันเอง โดยมีโครงสร้างใกล้เคียงกับคอมพิวเตอร์ คือ ภายในประกอบด้วยหน่วยรับข้อมูลและโปรแกรม หน่วยประมวลผล หน่วยความจำ หน่วยแสดงผล ซึ่งส่วนประกอบเหล่านี้มีความสมบูรณ์ในตัวของมันเอง ทำให้มีขนาดเล็ก และสามารถเขียนโปรแกรมควบคุมการทำงานของอุปกรณ์ต่างๆ ที่เชื่อมต่อกับตัวมัน ง่ายต่อการนำไปประยุกต์ใช้งาน

หลักการควบคุมการทำงานของหุ่นยนต์ให้สามารถเคลื่อนที่ได้นั้นจะใช้การเขียนโปรแกรมด้วยภาษาแอสแซมบลีแล้วโหลดโปรแกรมลงบอร์ดไมโครคอนโทรลเลอร์ ซึ่งมีหลักการเขียนดังนี้



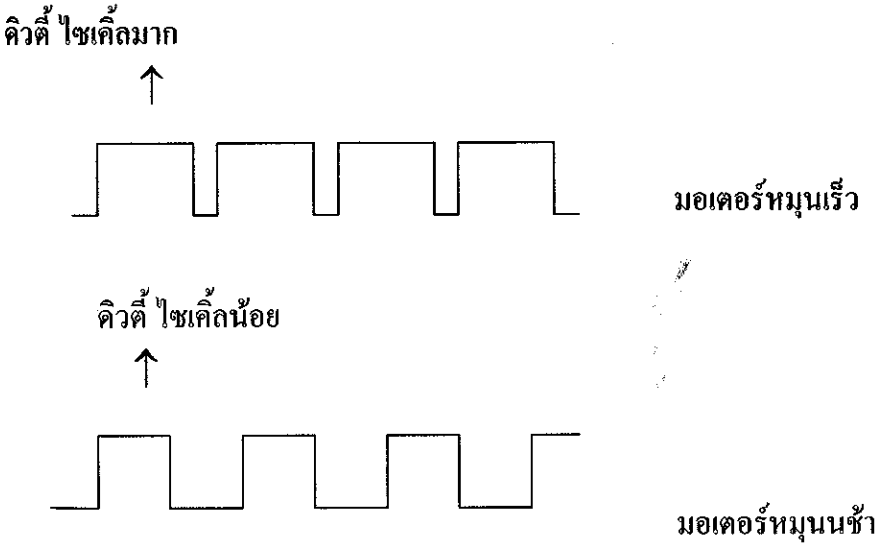
4.3 การควบคุมมอเตอร์

ในหัวข้อนี้จะกล่าวถึงการทดลองควบคุมมอเตอร์ด้วยไมโครคอนโทรลเลอร์ โดยใช้ร่วมบอร์ดขับมอเตอร์ เอ็ม ที บี ในการเชื่อมต่อกับมอเตอร์ การทดลองจะกระทำกับมอเตอร์ของรถเด็กเล่น อย่างไรก็ตาม เราสามารถใช้มอเตอร์อื่นๆ ที่ต้องใช้กระแสขับไม่เกินสองแอมแปร์ใดๆก็ได้ ฟังก์ชันการควบคุมมอเตอร์มีอยู่สองแบบคือ

1. ควบคุมทิศทางการหมุนของมอเตอร์ ซ้ายขวาหรือหน้าหลัง
2. ควบคุมความเร็วของมอเตอร์โดยใช้เทคนิคพลัสวิดท์มอดดูเลชั่น (pulse width modulation PWM)

บอร์ดขับมอเตอร์ เอ็มทีบี ใช้ไอซี แอลสองเก้าแปด ซึ่งมีฟังก์ชันการควบคุมมอเตอร์ข้างต้น ซึ่งฟังก์ชันดังกล่าวมีการควบคุมทิศทางการหมุนของมอเตอร์กำหนดด้วยลอจิกของสี่ซึ่งมีฟังก์ชันการควบคุมดังตาราง

ขา1	ขา2	การทำงานของมอเตอร์
0	0	มอเตอร์ไม่หมุน
0	1	มอเตอร์หมุนซ้ายหรือขวา
1	0	มอเตอร์หมุนขวาหรือซ้าย
1	1	มอเตอร์หยุด



รูปที่ 4.16 ผลของค่าดิวตี้ ไซเคิ้ล ต่อความเร็วของมอเตอร์

ส่วนการควบคุมความเร็วของมอเตอร์สามารถทำได้โดยง่าย พลัสวิดท์มอดดูเลชั่นให้กับ สามถ้า พลัสวิดท์มอดดูเลชั่น มีดิวตี้ ไซเคิ้ลมาก มอเตอร์จะหมุนด้วยความเร็วสูง ในมางตรงกันข้าม ถ้า

พลัลส์วิดท์มอดดูเลชั่น ที่จ่ายให้กับ สาม มีดิวตี้ ไซเคิ้ลน้อย มอเตอร์จะหมุนด้วยความเร็วต่ำเป็นสัดส่วนด้วย ดังรูปที่4.16 แสดงตัวอย่างการกำหนด ดิวตี้ ไซเคิ้ล สำหรับการควบคุมความเร็วของมอเตอร์ ดังนั้นเราจึงสามารถควบคุมความเร็วของมอเตอร์ได้ โดยให้ไมโครคอนโทรลเลอร์ส่งพลัลส์วิดท์มอดดูเลชั่น จากพอร์ตเอาท์พุตมายังบอร์ด เอ็ม ที บี เพื่อใช้ควบคุมความเร็วของมอเตอร์และส่งลอจิกออกจากพอร์ตเอาท์พุตมาควบคุมทิศทางของมอเตอร์

ซึ่งในโครงงานนี้เราจะมาทำการดัดแปลงสร้างรถเด็กเล่นให้กลายเป็นหุ่นยนต์สมองกล สำหรับรถเด็กเล่นที่จะนำมาดัดแปลงนั้นเป็นรถที่ใช้มอเตอร์เป็นตัวขับเคลื่อนทั้งการเดินหน้าถอยหลัง หรือการเลี้ยวซ้ายเลี้ยวขวา ซึ่งเราสามารถนำการออกแบบการควบคุมมอเตอร์มาประยุกต์ใช้ในที่นี้ได้ ซึ่งเราสามารถแบ่งการออกแบบสร้างหุ่นยนต์ได้ดังต่อไปนี้

1.นำรถเด็กเล่นมาถอดโครงออกให้เห็นมอเตอร์สองตัวข้างใน

2.ทำการตรวจสอบว่ามอเตอร์ตัวใดสำหรับเลี้ยวซ้ายขวา มอเตอร์ตัวใดเดินหน้าถอยหลัง

3.หลังจากนั้นก็นำสายไฟจากมอเตอร์มาต่อเข้ากับบอร์ดไมโครคอนโทรลเลอร์ และแบตเตอรี่

4.ทำการโหลดโปรแกรมลงบนหุ่นยนต์

5.ทำการติดตั้งองค์ประกอบต่างๆให้อยู่บนตัวรถ จัดสายไฟให้เรียบร้อย

6.ตรวจสอบและทดลองการทำงานของหุ่นยนต์ว่าสามารถเดินได้ตามที่เราต้องการหรือไม่

7.ตกแต่งหุ่นยนต์ให้สวยงามตามต้องการ

สรุปผลการทดลอง

จากหุ่นยนต์ที่สร้างขึ้นมาจะเห็นได้ว่าการเคลื่อนไหวของหุ่นยนต์นั้นจะขึ้นกับความแรงของแบตเตอรี่เป็นหลัก เพราะถ้าหากว่าแบตเตอรี่เต็มการเคลื่อนที่ของหุ่นยนต์ต้นแบบจะเคลื่อนที่ไปได้เร็วกว่าขณะที่แบตเตอรี่อ่อนเพราะฉะนั้นเราจะไม่สามารถควบคุมการเคลื่อนไหวของหุ่นยนต์ได้ เพราะเวลาที่แบตเตอรี่เต็มหุ่นยนต์อาจเคลื่อนที่ไปได้ระยะทางมากกว่าเวลาที่แบตเตอรี่อ่อนทำให้เรากำหนดค่าที่แน่นอนไม่ได้ว่าต้องให้ความเร็วเท่าใดหุ่นยนต์จึงจะก้าวข้ามกริชไปได้ หลักการแก้ไขจึงกระทำโดยเปลี่ยนมาใช้สเต็ปมอเตอร์แทน ซึ่งหลักการดังกล่าวจะกล่าวไว้ในบทต่อไป

บทที่ 5
ปฏิบัติการสร้างหุ่นยนต์ต้นแบบโดยใช้สเต็ปมอเตอร์

5.1 กล่าวนำ

จุดประสงค์หลักในการสร้างหุ่นยนต์ต้นแบบนี้คือ ต้องการพัฒนาหุ่นยนต์ในอนาคตให้สามารถเคลื่อนที่ได้สะดวก คล่องตัว และยังสามารถประยุกต์ใช้งานรับ-ส่งเอกสารได้ ซึ่งปัจจัยที่ส่งผลต่อการเคลื่อนที่ของหุ่นยนต์ก็คือ มอเตอร์ เนื่องจากเราใช้มอเตอร์เป็นตัวขับเคลื่อนหุ่นยนต์ ดังนั้นมอเตอร์จึงเป็นตัวแปรสำคัญที่เราต้องพิจารณาเลือกใช้ให้เหมาะสม

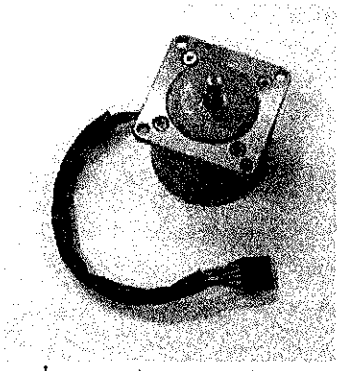
จากบทที่ 4 เราได้สร้างหุ่นยนต์ที่ใช้มอเตอร์แบบธรรมดามาเป็นตัวขับเคลื่อน และเมื่อทำการทดสอบการเคลื่อนที่ของหุ่นยนต์พบว่า หุ่นยนต์สามารถเคลื่อนที่ได้ แต่ยังบังคับการเคลื่อนไหวของหุ่นยนต์ได้ยาก อีกทั้งลักษณะการเคลื่อนที่ไม่แน่นอนคือ การหมุนแกนเพลาของล้อขึ้นอยู่กับแบตเตอรี่ที่จ่ายให้กับมอเตอร์ หากแรงดันที่จ่ายออกมาจากแบตเตอรี่มีค่ามากหุ่นยนต์ก็จะเคลื่อนที่ได้เร็ว แต่หากจ่ายแรงดันออกมาน้อยการเคลื่อนที่ก็จะช้า ทางคณะผู้จัดทำได้เล็งเห็นถึงปัญหาที่เกิดขึ้นจึงได้แก้ปัญหานี้โดยการเปลี่ยนมาใช้สเต็ปมอเตอร์เพื่อขับเคลื่อนหุ่นยนต์ ทั้งนี้เนื่องจากสเต็ปมอเตอร์เป็นมอเตอร์ที่ขับเครื่องด้วยพัลส์ ลักษณะการหมุนเป็นสเต็ป ซึ่งตรงจุดนี้เองจะช่วยให้เรากำหนดตำแหน่งการเคลื่อนที่ของหุ่นยนต์ได้แน่นอนแม่นยำมากยิ่งขึ้น

5.2 รายละเอียดพื้นฐานของสเต็ปมอเตอร์

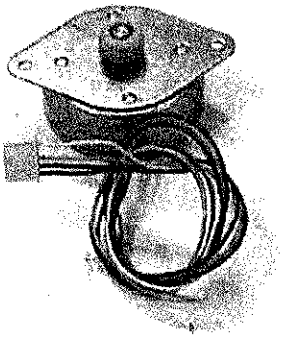
สเต็ปมอเตอร์เป็นมอเตอร์ที่ขับเครื่องด้วยพัลส์ ลักษณะการขับเคลื่อนจะหมุนรอบแกนได้ 360 องศา มีลักษณะไม่ต่อเนื่องแต่มีลักษณะเป็นสเต็ป โดยแต่ละสเต็ปจะขับเคลื่อนได้ 1, 1.5, 1.8 หรือ 2 องศา แล้วแต่โครงสร้างของมอเตอร์ ลักษณะที่นำมอเตอร์ไปใช้จะเป็นงานที่ต้องการตำแหน่งแม่นยำ เช่น ระบบขับเคลื่อนหัวแม่พิมพ์ในเครื่องพิมพ์ (PRINTER) ระบบขับเคลื่อนหัวอ่านในเครื่องอ่านบันทึกเหล็ก ระบบขับเคลื่อนตำแหน่งของปากกาใน X-Y PLOTER เป็นต้น สเต็ปมอเตอร์เป็นมอเตอร์ที่มีลักษณะเมื่อป้อนไฟฟ้าให้จะทำให้มอเตอร์หมุนเพียงเล็กน้อยตามเส้นรอบวงและหยุด ซึ่งต่างจากมอเตอร์ทั่วไปที่จะหมุนทันทีและตลอดเวลาเมื่อป้อนแรงดันไฟฟ้า ซึ่งเป็นข้อดีของสเต็ปมอเตอร์ที่ทำให้เราสามารถกำหนดตำแหน่งของการหมุนด้วยตัวเลข(องศาหรือระยะทาง) ได้อย่างละเอียด

สเต็ปมอเตอร์มีหลายขนาด หลายโครงสร้าง ดังนั้นการใช้งานในลักษณะต่างๆย่อมต้องการคุณสมบัติของมอเตอร์ที่แตกต่างกันไป เราจึงควรเลือกใช้ให้เหมาะสมกับชิ้นงาน

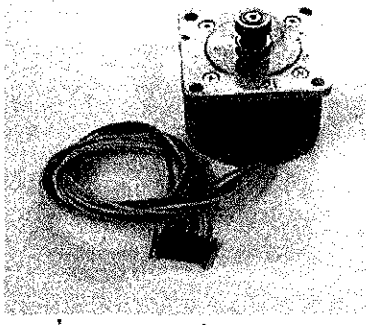
สเต็ปมอเตอร์มีลักษณะดังรูป



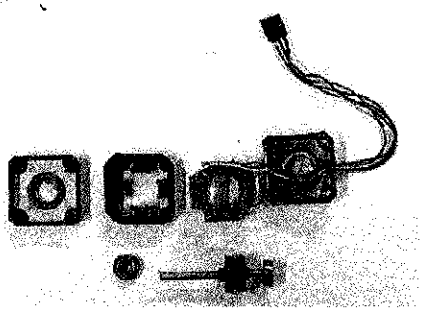
รูปที่ 5.1 สเต็ปมอเตอร์แบบมีสาย 5 เส้น ไบโพลาร์



รูปที่ 5.3 สเต็ปมอเตอร์หลายแบบ



รูปที่ 5.2 มอเตอร์แบบมีสาย 6 เส้น



รูปที่ 5.4 แสดงภาพถ่ายโครงสร้างสเต็ปมอเตอร์

สเต็ปมอเตอร์ที่พบในปัจจุบันมี 3 ลักษณะดังนี้

1. แบบแม่เหล็กถาวร (PERMANENT MAGNET_PM)

สเต็ปมอเตอร์แบบ PM จะมีสเตเตอร์ (STATOR) ที่พันขดลวดไว้หลายๆ โพล โดยมีโรเตอร์ (ROTOR) เป็นรูปทรง กระบอกฟันเลื่อย และโรเตอร์ทำด้วยแม่เหล็กถาวร เพื่อป้อนไฟกระแสตรง ให้กับขดสเตเตอร์ จะทำให้เกิดแรงแม่เหล็กไฟฟ้าผลักต่อโรเตอร์ ทำให้มอเตอร์หมุน มอเตอร์แบบ PM จะเกิดแรงฉุดยึดให้โรเตอร์หยุดอยู่กับที่ แม้จะไม่ได้ป้อนไฟเข้าขดลวด

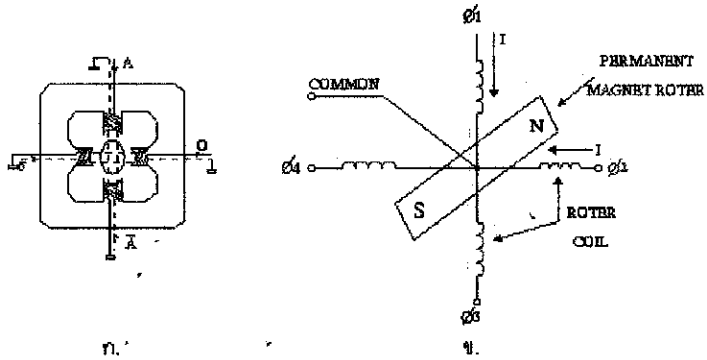
2. แบบแปรค่ารีลักแตนซ์ (VARIABLE RELUCTANCE- VR)

สเต็ปมอเตอร์แบบVR จะมีการหมุนโรเตอร์ได้อย่างอิสระ แม้จะไม่ได้จ่ายไฟให้โรเตอร์ทำจากสารเฟอร์โรแมกเนติก กำลังอ่อน มีลักษณะเป็นฟันเลื่อย รูปทรงกระบอกโดยจะมีความสัมพันธ์โดยตรงกับจำนวนโพลในสเตเตอร์ แรงบิดที่เกิดขึ้นจะไปหมุนโรเตอร์ ไปในเส้นทางของอำนาจแม่เหล็กที่มีค่ารีลักแตนท์ต่ำที่สุด ตำแหน่งที่จะเกิดแน่นอนและมีเสถียรภาพแต่จะเกิดขึ้นได้หลายๆ จุดดังนั้นเมื่อป้อนไฟเข้าขดลวดต่างๆ ในมอเตอร์แตกต่างขดกันไป ก็ทำให้มอเตอร์ หมุนไป

ตำแหน่งต่างๆ กันโรเตอร์ของ VR จะมีความเฉื่อยของโรเตอร์น้อยจึงมีความเร็วรอบสูงกว่ามอเตอร์แบบ PM

3. แบบผสม (HYBRID-H)

สเต็ปมอเตอร์แบบ H จะเป็นลูกผสมของ VR กับ PM โดยจะมีสเตเตอร์คล้ายกับที่ใช้ใน VR โรเตอร์มีหมวกหุ้ม ปลายซึ่งมีลักษณะของสารแม่เหล็กที่มีกำลังสูง โดยการควบคุมขนาดรูปร่างของหมวกแม่เหล็กอย่างดีทำให้ได้มุม การหมุนและครั้งน้อยและแม่นยำ ข้อดีก็คือ ให้แรงบิดสูงและมีขนาดกระทัดรัด และให้แรงฉุดยึดโรเตอร์นิ่งกับที่ตอนไม่จ่ายไฟ



ก. ข.

รูปที่ 5.5. แสดง (ก) โครงสร้าง (ข) วงจรเทียบเท่า (equivalent circuit) ของมอเตอร์ ชนิด 4 ขด

การสั่งงานควบคุมการหมุนของสเต็ปมอเตอร์

การควบคุมและสั่งงานให้สเต็ปมอเตอร์ทำงานไปทีละสเต็ปสามารถทำได้โดยการจ่ายกำลังไฟไปยังขดลวดในแต่ละขดบนสเตเตอร์ โดยการป้อนจะทำในลักษณะเป็นลำดับหรือเรียกว่าซีเควนเวียลในลูปที่ถูกต้อง ซึ่งจะแบบ ได้เป็น 3 รูปแบบ คือ แบบเวฟ (wave) แบบ 2 เฟส (2 phase) และแบบครึ่งสเต็ป (half step) ทั้ง 3 แบบนี้ก็จะมีข้อดีและข้อเสียต่างกันออกไป

1. แบบเวฟ (wave)

จะเป็นการกระตุ้นแบบที่ง่ายที่สุด ซึ่งจะทำการกระตุ้นขดลวดทีละขดในเวลาหนึ่งๆเรียงกันไป ตัวอย่างเช่น ขดที่ 1 , 2 , 3 , 4 ,1, 2 , 3 , 4 เป็นลำดับอย่างนี้ หรือ ขด 1 , 4 , 3 , 2 , 1 , 4 , 3 , 2 เป็นลำดับกันไป ทั้งนี้ขึ้นอยู่กับทิศทางที่เราต้องให้มอเตอร์หมุนไป วงจรที่นำมากระตุ้นนั้นจะมีราคาค่อนข้างจะถูกกว่าและง่ายกว่า

เราสามารถเขียนขั้นตอนการทำงานเป็นตารางออกมาได้ดังนี้

Step No.	Phase 1	Phase 2	Phase 3	Phase 4
1	ON			
2		ON		
3			ON	
4				ON
5	ON			
6		ON		

2. แบบ 2 เฟส(2 Phase)

แบบนี้ก็จะคล้ายกับการกระตุ้นในแบบเวฟแต่จะต่างกันตรงที่ แบบ 2 เฟส จะกระตุ้นทีละ 2 ขด ที่อยู่ใกล้กันในเวลาเดียวกัน และจะเรียงลำดับกันไป ดังเช่นแบบเดียวกับแบบเวฟ จะยกตัวอย่างการกระตุ้นขดลวดในลักษณะ ซีเควนให้ดูดังนี้ 12,23,34.41,12,23,34,41 เรียงลำดับกันไปเรื่อยๆ หรือจะเป็น 14,43,32,21,14,43,32,21 เรียงกันไปเรื่อยๆเช่นกัน ถ้าจะมากล่าวถึงข้อดีข้อเสียของแบบ 2 เฟส แล้วมีดังนี้

- ข้อดี คือ การที่เราเพิ่มจำนวนขดลวดที่ถูกกระตุ้นจะทำให้เกิดแรงบิดได้มากกว่าแบบเวฟ ซึ่งโรเตอร์จะหมุนด้วยแรงดึงแบบเต็มๆแรงจาก ทั้ง 2 ขดลวดที่กระตุ้นพร้อมกัน
- ข้อเสีย คือ การที่จะกระตุ้นขดลวดแบบ 2 เฟสนั้นต้องใช้กำลังไฟมากขึ้นเป็น 2 เท่าของแบบเวฟ

เราสามารถเขียนลำดับการกระตุ้นของขดลวดแบบ 2 เฟส ได้ดังในตารางต่อไปนี้

Step No.	Phase 1	Phase 2	Phase 3	Phase 4
1	ON	ON		
2		ON	ON	
3			ON	ON
4	ON			ON
5	ON	ON		
6		ON	ON	

3. แบบครึ่งสเต็ป

แบบนี้แบบรูปแบบผสมผสานของการกระตุ้นระหว่าง แบบเวฟ กับ แบบ 2 เฟส เพื่อให้จำนวนรอบของสเต็ปมากขึ้นเป็น 2 เท่า ซึ่งในระบบนี้จะทำการกระตุ้นขดลวดเรียงกันไปเรื่อยๆเป็นลำดับ ดังจะยกตัวอย่างต่อไปนี้ 1,12,2,23,3,34,4,41,1,12,2,23,3,34,4,41,1 เป็นลำดับอยู่อย่างนี้เรื่อยไป ถ้าเราจะกลับทิศทางการหมุนก็จะได้เป็นดังนี้ 1,41,4,43,3,32,2,21,1,41,4,43,3,32,2,21,1 เป็นลำดับกันไปเราจะมาพูดกันถึงข้อดีและข้อเสียของการกระตุ้นแบบครึ่งสเต็ปกัน

- ข้อดีของการกระตุ้นแบบนี้จะให้แรงบิดเพิ่มมากขึ้น เนื่องจากช่วงสเต็ปที่มีระยะสั้นลง อีกประการหนึ่ง คือ แต่ละสเต็ปเกิดแรงดึงจากขดลวด 2 ขดที่ถูกกระตุ้นพร้อมกันเป็นผลให้ค่าตำแหน่งความถูกต้องมากขึ้นไปด้วย
- ข้อเสีย ก็คงจะเช่นเดียวกับแบบ 2 เฟส คือ ต้องจ่ายกำลังไฟเป็น 2 เท่าของแบบเวฟหรือจะใช้เท่ากับแบบ 2 เฟส นั้นเอง

ดังนั้นเราสามารถนำลำดับการทำงานของ แบบครึ่งเฟส ในรูปของตารางได้ดังนี้

Step No.	Phase 1	Phase 2	Phase 3	Phase 4
1	ON			
2	ON	ON		
3		ON		
4		ON	ON	
5			ON	
6			ON	ON
7				ON
8	ON			ON
9	ON			
10	ON	ON		

5.3 ตัวอย่างโค้ดโปรแกรมทดสอบ

```
        org     0000h
        mov     p0,#00000000b
loop:   mov     p0,#00000111b   ------>   กระตุ้นขดลวดที่1
        acall   delay
        mov     p0,#00001110b   ------>   กระตุ้นขดลวดที่2
        acall   delay
        mov     p0,#00001101b   ------>   กระตุ้นขดลวดที่3
        acall   delay
        mov     p0,#00001011b   ------>   กระตุ้นขดลวดที่4
        acall   delay
        sjmp    loop
delay:  mov     r5,#0Cfh
delay1: mov     r6,#0ffh
        djnz    r6,$
        djnz    r5,delay1
        ret
        end
```

โค้ดโปรแกรมที่แสดงข้างต้นนี้เป็นโปรแกรมที่ใช้ทดสอบการเคลื่อนที่ของหุ่นยนต์ที่ขับเคลื่อนโดยสเต็ปมอเตอร์ โค้ดโปรแกรมเขียนด้วยภาษาแอสแซมบี ซึ่งจะเห็นว่าลักษณะการสั่งงานควบคุมการหมุนของสเต็ปมอเตอร์จะเป็นแบบเวฟ คือ จะทำการกระตุ้นขดลวดทีละขดในเวลาหนึ่งๆเรียงกันไป

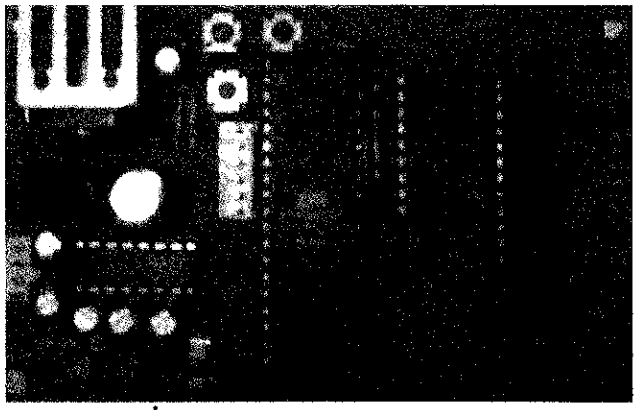
5.4 การสร้างหุ่นยนต์ต้นแบบ (สเต็ปมอเตอร์)

อุปกรณ์หลักที่ใช้ในการสร้างหุ่นยนต์มีดังนี้

1. บอร์ดไมโครคอนโทรลเลอร์	1 บอร์ด
2. สเต็ปมอเตอร์	2 ตัว
3. แบตเตอรี่ขับบอร์ดไมโคร 9V	1 ก้อน
4. แบตเตอรี่ขับสเต็ปมอเตอร์ 12V	1 ก้อน
5. แผ่นอะคีลิก	1 แผ่น
6. อะลูมิเนียมฉาก	1 เส้น
7. ล้อยาง	2 ล้อ

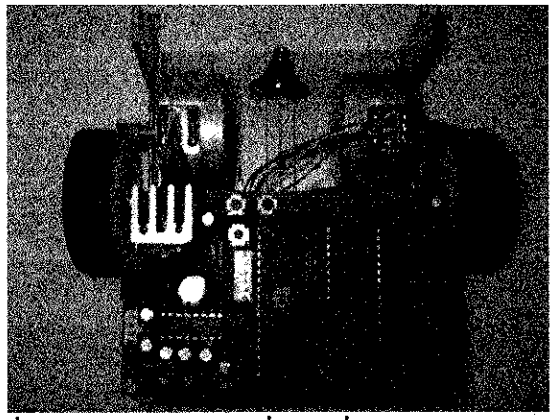
ขั้นตอนการสร้างหุ่นยนต์

1. นำไอซีต่างๆ มาต่อเข้าด้วยกันเป็นวงจรไฟฟ้า เพื่อประกอบเป็นบอร์ดไมโครคอนโทรลเลอร์ (รายละเอียดของบอร์ดไมโครคอนโทรลเลอร์ได้กล่าวไว้ในบทที่ 4)



รูปที่ 5.6 บอร์ดไมโครคอนโทรลเลอร์

2. ทำการสร้างโครงของหุ่นยนต์ โดยการนำอะลูมิเนียมฉากและแผ่นอะคีลิกมาจัดวางตามแบบโครงที่ต้องการ
3. เมื่อเราได้โครงของหุ่นยนต์แล้ว จากนั้นให้ใส่ล้อทั้ง 2 ข้างกับตัวหุ่นยนต์ และต่อสเต็ปมอเตอร์เข้ากับแกนเพลาล้อทั้ง 2ข้าง ซึ่งในการเชื่อมต่อระหว่างล้อและสเต็ปมอเตอร์จะต้องตรวจเช็คเฟืองของล้อกับเฟืองของสเต็ปมอเตอร์ให้เข้ากันได้อย่างพอดี ซึ่งตรงจุดนี้มีผลต่อการเคลื่อนที่ของหุ่นยนต์ด้วย
4. นำสายไฟมาเชื่อมบอร์ดไมโครคอนโทรลเลอร์เข้ากับสเต็ปมอเตอร์ เนื่องจากเราจะทำการสั่งงานควบคุมการหมุนของสเต็ปมอเตอร์ผ่านทางบอร์ดไมโครคอนโทรลเลอร์
5. ต่อแบตเตอรี่เข้ากับหุ่นยนต์เพื่อขับบอร์ดไมโครคอนโทรลเลอร์และสเต็ปมอเตอร์
6. เขียนโปรแกรมทดสอบการเคลื่อนที่ของหุ่นยนต์ แล้วโหลดลงบนหน่วยความจำของบอร์ดไมโครคอนโทรลเลอร์
7. ทดสอบการเคลื่อนที่ของหุ่นยนต์



รูปที่ 5.7 หุ่นยนต์ต้นแบบที่ขับเคลื่อนด้วยสเต็ปมอเตอร์

5.5 ผลการทดสอบการเคลื่อนที่ของหุ่นยนต์

เมื่อทำการทดลองให้หุ่นยนต์เคลื่อนที่ พบว่าหุ่นยนต์ของเรามีการเคลื่อนที่ได้อย่างต่อเนื่อง สะดวก คล่องตัว จังหวะการหมุนของล้อเป็นไปอย่างสม่ำเสมอ หมุนทีละสเต็ปไปเรื่อยๆ และจุดที่แตกต่างจากหุ่นยนต์ที่ใช้มอเตอร์แบบธรรมดาที่เห็นได้อย่างชัดเจน คือ ลักษณะการเลี้ยวของหุ่นยนต์ ซึ่งพบว่าหุ่นยนต์สามารถเลี้ยวซ้าย-ขวา ได้ด้วยมุม 90 องศาพอดี การเลี้ยวเป็นไปอย่างสมบูรณ์ ไม่พบการหยุดชะงักเหมือนการเลี้ยวของหุ่นยนต์ที่ขับเคลื่อนด้วยมอเตอร์แบบธรรมดา และลักษณะการเลี้ยวของหุ่นยนต์สามารถควบคุมได้ง่าย โดยเรากำหนดระยะทางการเคลื่อนที่ มุมการเลี้ยว(องศา) ได้ด้วยการนับสเต็ปการหมุนของมอเตอร์แล้วเขียนเป็นโค้ดโปรแกรมควบคุมการเคลื่อนที่

การเปลี่ยนมาใช้สเต็ปมอเตอร์ขับเคลื่อนหุ่นยนต์ ทำให้หุ่นยนต์สามารถเคลื่อนที่ได้อย่างสมบูรณ์มากยิ่งขึ้น ช่วยขจัดปัญหาที่เกิดจากการจ่ายแรงดันไฟที่ไม่สม่ำเสมอ และยังสามารถกำหนดตำแหน่ง ระยะการเคลื่อนที่ได้อย่างแน่นอนแม่นยำอีกด้วย

บทที่ 6
สรุปผลของโครงงาน

โครงงานฉบับนี้ได้นำเสนอผลการพัฒนาเทคโนโลยีหุ่นยนต์ โดยการนำหลักการเรียนรู้แบบรีอินฟอร์สเมนท์ (Reinforcement Learning หรือ RL) ในรูปของออน โพลิซี มัลติ คาลโล (On-Policy Monte Carlo หรือ ONMC) มาประยุกต์ใช้ร่วมกับหุ่นยนต์ต้นแบบ

วัตถุประสงค์ในการสร้างหุ่นยนต์ต้นแบบนี้ก็เพื่อพัฒนาเทคโนโลยีหุ่นยนต์เพื่อให้สามารถเรียนรู้การตัดสินใจได้ด้วยตนเอง เป็นการเพิ่มศักยภาพการทำงาน รวมทั้งเรายังนำหุ่นยนต์ไปประยุกต์ใช้งานอื่นๆได้ เช่น งานรับ-ส่งเอกสาร และงานสำรวจพื้นที่ที่มนุษย์ไม่สามารถเข้าถึงได้ เป็นต้น

ผลการจำลองแบบทางคอมพิวเตอร์(Simulation) จะเห็นได้จากกราฟแสดงผลว่า การเรียนรู้แบบรีอินฟอร์สเมนท์ช่วยเพิ่มทักษะการเรียนรู้การตัดสินใจเลือกกุารกระทำ จนสามารถหาเส้นทางที่สั้นที่สุดเพื่อไปยังจุดมุ่งหมายได้

เมื่อเรานำเอาหลักการเรียนรู้แบบรีอินฟอร์สเมนท์มาเปรียบเทียบกับกระบวนการหาเส้นทางของมนุษย์ภายใต้บริเวณที่ซับซ้อน เช่น มีสิ่งกีดขวาง จะพบว่าอัลกอริทึมแบบรีอินฟอร์สเมนท์สามารถค้นหาเส้นทางเพื่อไปยังจุดมุ่งหมายได้ระยะทางที่สั้นกว่ากระบวนการคิดของมนุษย์ ดังจะเห็นได้จากการทดลอง เมื่อเทียบกับสัดส่วนการเดินทางที่ไปไม่ถึงจุดหมายระหว่างมนุษย์กับกระบวนการเรียนรู้แบบรีอินฟอร์สเมนท์ จะพบว่ามนุษย์มีเปอร์เซ็นต์ที่เดินทางไปไม่ถึงจุดหมายถึง 28% ส่วนการเรียนรู้แบบรีอินฟอร์สเมนท์มีเพียง 10.7%

การทดลองภาคสนามพบว่าหุ่นยนต์ของเรามีการเคลื่อนที่ได้อย่างต่อเนื่อง สะดวก จังหวะการหมุนของล้อเป็นไปอย่างสม่ำเสมอ หมุนทีละสเต็ปไปเรื่อยๆ ลักษณะการเลี้ยวของหุ่นยนต์สามารถควบคุมได้ง่าย โดยเรากำหนดระยะทางการเคลื่อนที่ มุมการเลี้ยว(องศา) ได้ด้วยการนับสเต็ปการหมุนของมอเตอร์แล้วเขียนเป็นโค้ดโปรแกรมควบคุมการเคลื่อนที่ แต่ก็พบว่าหุ่นยนต์ต้นแบบเคลื่อนที่ได้ด้วยความเร็วที่จำกัด

ปัญหา และอุปสรรคในการทำโครงงาน

1. ลักษณะการเลี้ยวควบคุมได้ยาก

   เนื่องจากในช่วงแรกของการสร้างหุ่นยนต์ต้นแบบ เราได้ใช้มอเตอร์แบบธรรมดาเป็นตัวขับเคลื่อนหุ่นยนต์ ซึ่งส่งผลทำให้หุ่นยนต์เคลื่อนที่ได้ไม่ต่อเนื่อง มีการกระตุกขณะเคลื่อนที่ ลักษณะการหมุนเพลาของล้อขึ้นอยู่กับขนาดแบตเตอรี่ที่จ่ายให้กับมอเตอร์ และที่สำคัญคือการใช้มอเตอร์แบบธรรมดานี้จะส่งผลถึงระบบการควบคุมองศาการเลี้ยวที่เป็นไปได้ยาก

2. หุ่นยนต์เคลื่อนที่ได้ช้า

เมื่อเราสร้างหุ่นยนต์ต้นแบบขึ้นมาใหม่ โดยใช้สเต็ปมอเตอร์เป็นตัวขับเคลื่อนหุ่นยนต์ พบว่าหุ่นยนต์สามารถเคลื่อนที่ได้อย่างต่อเนื่อง คล่องตัว จังหวะการหมุนของล้อเป็นไปอย่างสม่ำเสมอ หมุนทีละสเต็ปไปเรื่อยๆ แต่ด้วยหุ่นยนต้นแบบที่สร้างมีขนาดเล็ก จึงเกิดข้อจำกัดในการเลือกใช้แบตเตอรี่ ทำให้เกิดปัญหาขึ้นมาอีก คือ แบตเตอรี่ขนาด 6 โวลต์ที่ใช้ ยังไม่เพียงพอต่อการขับสเต็ปมอเตอร์ ทำให้เราไม่สามารถขับให้สเต็ปมอเตอร์หมุนเร็วๆได้ ดังนั้นหุ่นยนต์ต้นแบบจึงเคลื่อนที่ได้ช้า

3. หุ่นยนต์ยังไม่สามารถที่จะเรียนรู้แบบออนไลน์(Online) ได้

ขณะนี้หุ่นยนต์ต้นแบบมีการเรียนรู้แบบออฟไลน์(Offline) คือ ใช้โปรแกรมการเรียนรู้แบบรีอินฟอร์สเมนต์ค้นหาเส้นทางที่สั้นที่สุดก่อน แล้วจึงนำผลลัพธ์ที่ได้มาโหลดให้กับหุ่นยนต์ต้นแบบ ซึ่งสาเหตุที่เรายังไม่สามารถสร้างหุ่นยนต์ให้เกิดการเรียนรู้แบบออนไลน์(Online) ได้นั้นเนื่องจากการมีงบประมาณในการทำโครงงานที่จำกัด จึงไม่สามารถนำอุปกรณ์จำพวกเซนเซอร์สามมิติมาช่วยให้หุ่นยนต์เกิดการเรียนรู้ด้วยตนเอง และอุปสรรคอีกข้อ คือ ขณะนี้ ยังไม่มีคอมไพล์เลอร์วิชั่ว ซี พลัส พลัส เพื่อโหลดโปรแกรมลงบอร์ดไมโครคอนโทรลเลอร์

การแก้ไขปัญหา

1. ลักษณะการเลี้ยวควบคุมได้ยาก

ปัญหานี้แก้โดยการเปลี่ยนมาใช้สเต็ปมอเตอร์ในการขับเคลื่อนหุ่นยนต์ ซึ่งจะส่งผลให้การเลี้ยวของหุ่นยนต์สามารถควบคุมได้ง่าย โดยเรากำหนดระยะทางการเคลื่อนที่ มุมการเลี้ยว (องศา) ได้ด้วยการนับสเต็ปการหมุนของมอเตอร์ แล้วเขียนเป็นโค้ดโปรแกรมควบคุมการเคลื่อนที่

2. หุ่นยนต์เคลื่อนที่ได้ช้า

เมื่อทราบแล้วว่าแบตเตอรี่ที่ใช้ยังไม่เพียงพอต่อการขับสเต็ปมอเตอร์ ดังนั้นวิธีแก้ปัญหา คือ การเพิ่มขนาดแรงดันของแบตเตอรี่ แต่หากแรงดันมากขึ้นแบตเตอรี่ก็จะก้อนใหญ่ขึ้น มีน้ำหนักมากขึ้น เราจึงต้องสร้างหุ่นยนต์ที่มีขนาดใหญ่ขึ้นเพื่อรองรับกับขนาดแบตเตอรี่ด้วย

3. หุ่นยนต์ยังไม่สามารถที่จะเรียนรู้แบบออนไลน์(Online) ได้

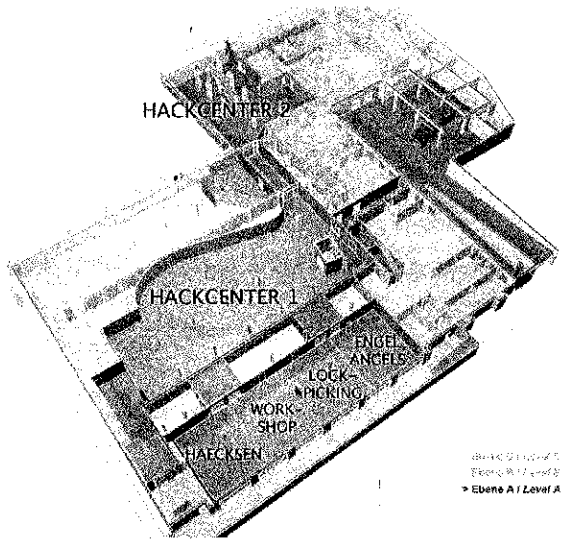
การทำให้หุ่นยนต์ต้นแบบมีการเรียนรู้แบบออนไลน์นั้นจะต้องอาศัยอุปกรณ์เสริมที่ช่วยในการตรวจจับสิ่งกีดขวางจำพวกเซนเซอร์สมามิติ\กล้อง และจะต้องหาโปรแกรมคอมไพล์เลอร์วิชั่ว ซี พลัส พลัส ลงบอร์ดไมโครคอนโทรลเลอร์ด้วย

ข้อเสนอแนะ

ส่วนนี้จะเป็นแนวทางในการพัฒนาหุ่นยนต์ต้นแบบเพื่อนำไปประยุกต์ใช้งานต่างๆได้

1. สามารถพัฒนาโปรแกรมเพื่อใช้สำหรับครอบคลุม บริเวณที่กว้างกว่าเดิมได้

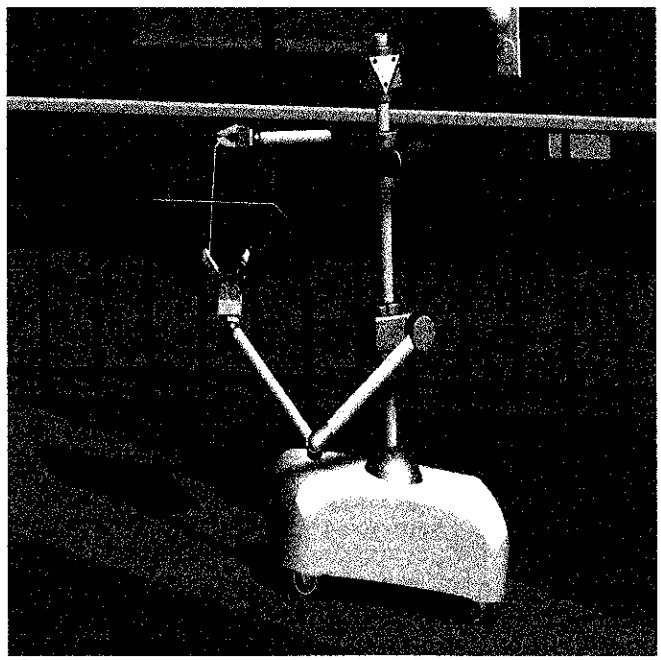
ในส่วนของโปรแกรมการเรียนรู้แบบรีอินฟอร์สเมนท์นั้นสามารถที่จะกำหนดสภาวะแวดล้อมแบบต่างๆ เพื่อให้ครอบคลุมพื้นที่ที่ซับซ้อนได้ สะดวกต่อการหาเส้นทาง



รูปที่ 6.1 แสดงสภาวะแวดล้อมที่เป็นพื้นที่ที่ซับซ้อน

2. การประยุกต์ใช้เซนเซอร์ร่วมกับหลักการคิดแบบรีอินฟอร์สเมนท์

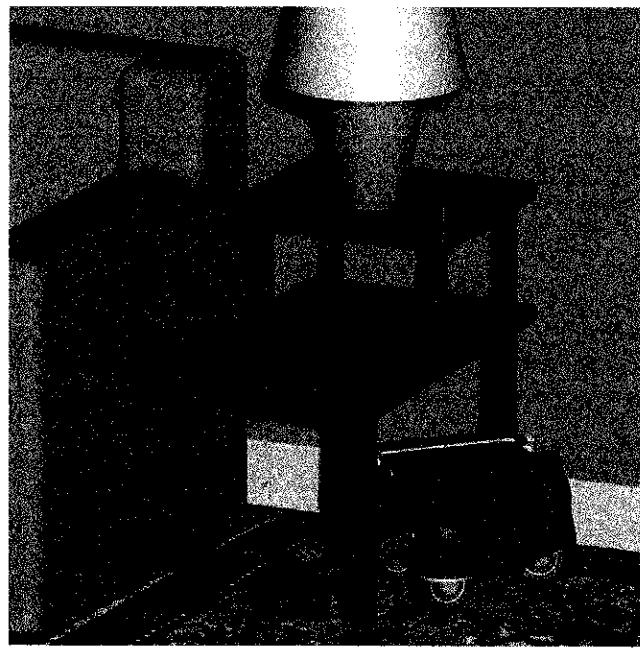
การใช้เซนเซอร์เข้ามาช่วยนั้นจะทำให้หุ่นยนต์สามารถรับรู้การเปลี่ยนแปลงสภาวะแวดล้อม ณ เวลานั้นได้เลย ซึ่งทำให้หุ่นยนต์เกิดการเรียนรู้ได้ทันที โดยที่มนุษย์ไม่ต้องเข้าไปเกี่ยวข้องด้วยเลย และเรายังสามารถนำหุ่นยนต์ต้นแบบไปใช้ในงานด้านอื่นๆได้ เช่น หุ่นยนต์ขนส่ง(Transport) หุ่นยนต์ทำความสะอาด (floor cleaning) หุ่นยนต์ส่งของ (delivery)



รูปที่ 6.2 แสดงถึงการประยุกต์ใช้งานหุ่นยนต์ต้นแบบในอนาคต

3. การนำหลักคิดแบบรีอินฟอร์สเมนท์มาหาเส้นทางพร้อมกับวัดระดับพลังงาน

เราสามารถนำหลักการเรียนรู้แบบรีอินฟอร์สเมนท์มาประยุกต์ใช้ในระดับที่สูงขึ้นอีกขั้น โดยใช้ RL เพื่อวัดระดับพลังงานไปพร้อมๆกับการหาเส้นทางที่เหมาะสม ซึ่งอาจจะกำหนดสถานะเป็นแกน x, y, z โดยแกน z จะเป็นแกนของระดับพลังงาน ตัวอย่างการนำไปใช้ก็เช่น หุ่นยนต์ดูดฝุ่น เพื่อที่จะให้เกิดประสิทธิภาพในการทำงาน หุ่นยนต์ดูดฝุ่นในอนาคตควรที่จะสามารถหลบหลีกสิ่งกีดขวาง ตัดสินใจด้วยตนเองได้ว่าควรจะใช้เส้นทางอย่างไรในการทำงาน และที่สำคัญหุ่นยนต์ดูดฝุ่นจะต้องคำนวณถึงระดับพลังงานที่เหลือว่าเพียงพอต่อการทำงานหรือไม่เพื่อป้องกันปัญหาพลังงานหมดกลางทาง



สิ่งที่ได้รับจากโครงงาน

- มีความรู้เกี่ยวกับเทคนิค Reinforcement Learning
- มีความรู้เกี่ยวกับการเขียนโปรแกรมด้วยภาษาวิชั่ว ซี พลัส พลัส (visual C++) และภาษาแอสแซมบี (Assembly)
- เพิ่มทักษะในการทำหุ่นยนต์
- สามารถคิดอย่างมีระบบและทำงานได้อย่างมีประสิทธิภาพ

ภาคผนวก

โค้ดโปรแกรมการเรียนรู้ของหุ่นยนต์

```
# include <iostream.h>      // Precompiled header
# include <stdlib.h>
# include <math.h>
# include <iomanip.h>
# include <cstdlib>
# include <fstream.h>
/*****************************************************************
                GLOBAL VARIABLES
*****************************************************************/
const int max_episode   = 1000000;
const int num_action  = 4 ;
const int num_state   = 10 ;
int   num_eps  ;
int   num_eps_visit[ num_state ][ num_state ][ num_action ] ;
double Q[ num_state ][ num_state ][ num_action ] ;
double Return[ num_state ][ num_state ][ num_action ] ;
double Returndummy[ num_state ][ num_state ][ num_action ] ;
int   iteration ;
int   i, a, action, next_action, state, next_state;
double randunif01 ;
int xstate, x_min = 0, x_max = 9 ;
int ystate, y_min = 0, y_max = 9 ;
double Inf = RAND_MAX;
int right = 0, left = 1, up = 2, down = 3 ;
int numhop = 0 ;
double reward = -1 ;
int step, step_max =  30 ;
int X = 0 , X_target = 9, X_old, a_old,  X_new ;
```

Initial Variable

Initial Variable

```
int Y = 0 , Y_target = 9, Y_old, Y_new ;
int Path[ 100 ][ 2 ] ;
int ActionHistory[ 100 ] ;                                    Initial Variable
double p_explore = 0.2 ;
double Accum_reward_eps = 0 ;
/*************************************************************************
                Declarations
*************************************************************************/
double Randunif01();
void Initialization() ;
void Reset() ;
void Writeresults() ;
/*************************************************************************
                Main Program
*************************************************************************/
void main()
{
  Initialization() ;
  double AVE_Accum_reward, Accum_reward  = 0 ;
     int flag_end  ;
     for (int episode = 1 ; episode<= max_episode ; episode++ )
      {
         Reset() ;
         for ( step = 1 ; step<= step_max ; step++ )
         {
         if ( X < 0 || X > 9 || Y < 0 || Y > 9 )                    ; Check for loops
                  int check = 0 ;
         // Check for loops
         double Q_[ num_action ];
         for (a = 0; a< num_action ; a++ )
         {
                  X_new = X ;
```

```
        Y_new = Y ;
        Q_[ a ] = Q[ X ][ Y ][ a ] ;
        if ( a == right )
                X_new = X + 1 ;
        if ( a == left )                                   } take action
                X_new = X - 1 ;
        if (a == up )
                Y_new = Y + 1 ;
        if (a == down)
                Y_new = Y - 1 ;
}
int flag_deadend = 0 ;
//if ( Q_[ 0 ] == -Inf && Q_[ 1 ] == -Inf && Q_[ 2 ] == -Inf && Q_[ 3 ] == -Inf )
//flag_deadend = 1 ;
if (flag_deadend == 0 )
{
    // For current state (X,Y), check action = arg_max Q(s,a)
        int a_max = 0 ;
        double Q_max = Q_[ a_max ] ;
        for( a = 0 ; a<4 ; a++ )
        {
          if ( Q_[ a ] >= Q_max )
            {
                a_max = a ;
                Q_max = Q_[ a ] ;
            }
         }// for a
         // Find soft policy
         double p = Randunif01() ;
         if ( p <= p_explore/(1 + double(episode)/200000 ) )
              {
                double Q_dummy = -Inf ;
```

```
                while ( Q_dummy == -Inf )
                {
                        if ( flag_deadend == 1 )
                          Q_dummy = 0 ;
                        else
                        {
                          double dummy = Randunif01() ;
                          if ( 0<= dummy && dummy < 0.25 )
                                action = 0 ;
                          else if ( 0.25 <= dummy && dummy < 0.5 )
                                action = 1 ;
                          else if ( 0.5 <= dummy && dummy < 0.75 )
                                action = 2 ;
                          else
                                action = 3 ;
                          Q_dummy = Q_[ action ] ;
                          }
                }//while
        } // if p explore
        else
                action = a_max ;
} // flag_deadend == 0 action
if ( flag_deadend ==0 )
{
   // Find new state
   X_old = X ;
   Y_old = Y ;
   if ( action == right )
        X++ ;
   if ( action == left )
        X-- ;
```

```
if (action == up )
    Y++ ;
if (action == down)
    Y-- ;
numhop++ ;
Path[ numhop ][ 0 ] = X ;
Path[ numhop ][ 1 ] = Y ;
num_eps_visit[ X_old ][ Y_old ][ action ]++ ;
ActionHistory[ numhop ] = action ;
// Immediate reward
if ( X != X_target || Y != Y_target ) //if ( X != X_target && Y != Y_target ) //
{
    Return[ X_old ][ Y_old ][ action ] += reward ;
    Accum_reward_eps += reward ;
    Returndummy[ X_old ][ Y_old ][ action ] += reward ;
}
if ( X == X_target && Y == Y_target ) //if ( X != X_target || Y != Y_target )
{
    //Return[ X_old ][ Y_old ][ action ] += 0 ;
    //Accum_reward_eps += 0 ;
    //*
    Accum_reward_eps += reward ; // 0;//
    for (i = 0 ; i<= numhop ; i++ )
    {
        X_old = Path[ i ][ 0 ] ;
        Y_old = Path[ i ][ 1 ] ;
        a_old = ActionHistory[ i ] ;
        Return[ X_old ][ Y_old ][ a_old ] += 20000/double(numhop) ;//0; //
    }//*/
  }
}// flag_deadend == 0 update
```

```
// Update Q function at end of episode)
flag_end = ( X == X_target && Y == Y_target) ;
if ( flag_end == 1 )
{
        //cout<< "Episode number =  "<< episode<< "   Step number = "<< step
        <<setw(25) << Accum_reward_eps << setw(25) << AVE_Accum_reward <<
        endl;
        ofstream outClientFile("Path.txt", ios::app );
        if ( !outClientFile ) { // overload ! operator
                cerr<<" File could not be opened"<<endl;
                exit(1);
        }
        outClientFile<< X << setw(15) << Y << setw(25) << episode<< setw(25) <<
        step  <<setw(25) << AVE_Accum_reward << endl;
//}// flagendendl;
}// flagend

if ( flag_end == 1  || flag_deadend == 1 || step == step_max )
{
   //cout<< "Episode number =  "<< episode<< "   Step number = "<< step  << endl;
   step = step_max ;
   for( xstate = x_min; xstate <= x_max ; xstate++ )
   {
       for( ystate = y_min; ystate <= y_max ; ystate++ )
       {
           for ( action = 0 ; action < 4 ; action++ )
           {
               if ( Q[ xstate ][ ystate ][ action ] != -Inf )
               {
```

```
                    if ( num_eps_visit[ xstate ][ ystate ][ action ] > 0 )
                      Q[ xstate ][ ystate ][ action ] = Q[ xstate ][ ystate ][ action ] + (
                      1/double( num_eps_visit[ xstate ][ ystate ][ action ] ) )*(Return[
                      xstate ][ ystate ][ action ] - Q[ xstate ][ ystate ][ action ]);
                    }
                    Return[ xstate ][ ystate ][ action ] = 0;
            }}}
            //Writeresults() ;
        }
        }// for step
        Accum_reward += Accum_reward_eps ;
        AVE_Accum_reward = Accum_reward/double(episode) ;
    }// for episode
    Writeresults() ;
   //system("pause");
}// end main

/*************************************************************************
            Subfunctions
*************************************************************************/
double Randunif01()
{
        randunif01 = ((double) rand())/RAND_MAX;
        return (randunif01);
}
void Initialization()
{
        for( xstate = x_min; xstate <= x_max ; xstate++ )
        {
            for( ystate = y_min; ystate <= y_max ; ystate++ )
            {
```

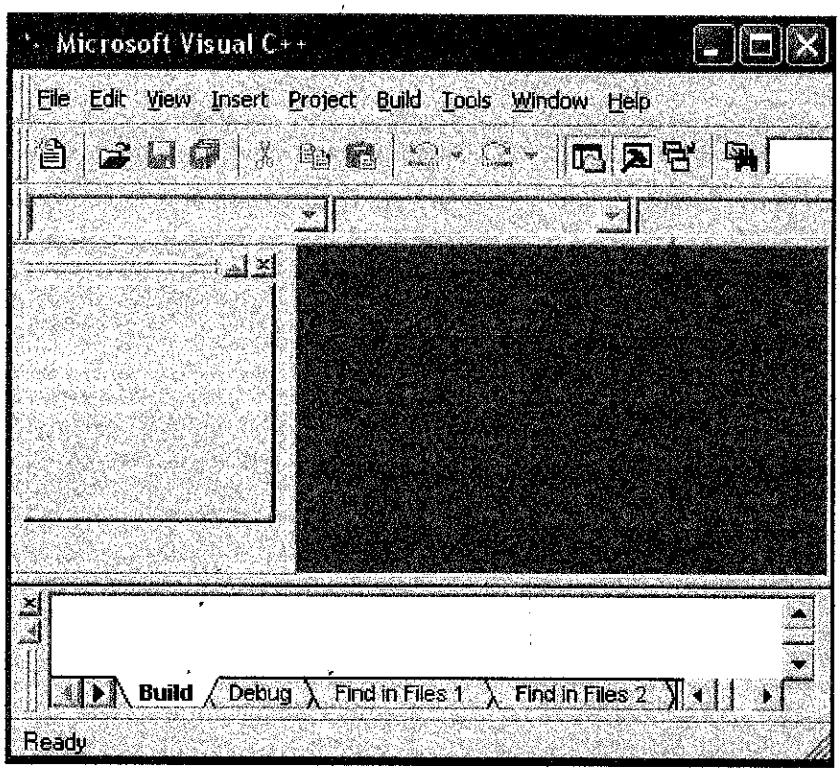
```
for ( action = 0 ; action < 4 ; action++ )
{
    Q[ xstate ][ ystate ][ action ] = -Randunif01();
    Return[ xstate ][ ystate ][ action ] = 0;
    Returndummy[ xstate ][ ystate ][ action ] = 0;
    num_eps_visit[ xstate ][ ystate ][ action ] = 0 ;

if ( xstate == x_min && ystate == y_min )
{
        Q[ xstate ][ ystate ][ down ] = -Inf ;
        Q[ xstate ][ ystate ][ left ] = -Inf ;
}
if ( xstate == x_min && ystate == y_max )
{
        Q[ xstate ][ ystate ][ up ] = -Inf ;
        Q[ xstate ][ ystate ][ left ] = -Inf ;
}
if ( xstate == x_max && ystate == y_min )
{
        Q[ xstate ][ ystate ][ down ] = -Inf ;
        Q[ xstate ][ ystate ][ right ] = -Inf ;
}
if ( xstate == x_max && ystate == y_max )
{
        Q[ xstate ][ ystate ][ up ] = -Inf ;
        Q[ xstate ][ ystate ][ right ] = -Inf ;
}
if ( x_min< xstate && xstate < x_max && ystate == y_min )
{
        Q[ xstate ][ ystate ][ down ] = -Inf ;
}
if ( x_min< xstate && xstate < x_max && ystate == y_max )
```

```
                {
                        Q[ xstate ][ ystate ][ up ] =  -Inf ;
                }
                if ( y_min< ystate  && ystate < y_max && xstate == x_max )
                {
                        Q[ xstate ][ ystate ][ right ] =  -Inf ;
                }
                if ( y_min< ystate  && ystate < y_max && xstate == x_min )
                {
                        Q[ xstate ][ ystate ][ left ] =  -Inf ;
                }
               } // for action
            }// for ystate
        }// for xstate
} // Initialization
void Reset()
{
        X = x_min ;
        Y = y_min ;
        for (int i = 0 ; i < 100 ; i++ )
        {
                Path[ i ][ 0 ] = -1 ;
                Path[ i ][ 1 ] = -1 ;
                ActionHistory[ i ] = -1;
        }
        numhop = 0 ;
        Accum_reward_eps = 0 ;
        Path[ 0 ][ 0 ] = X ;
        Path[ 0 ][ 1 ] = Y ;
}
```

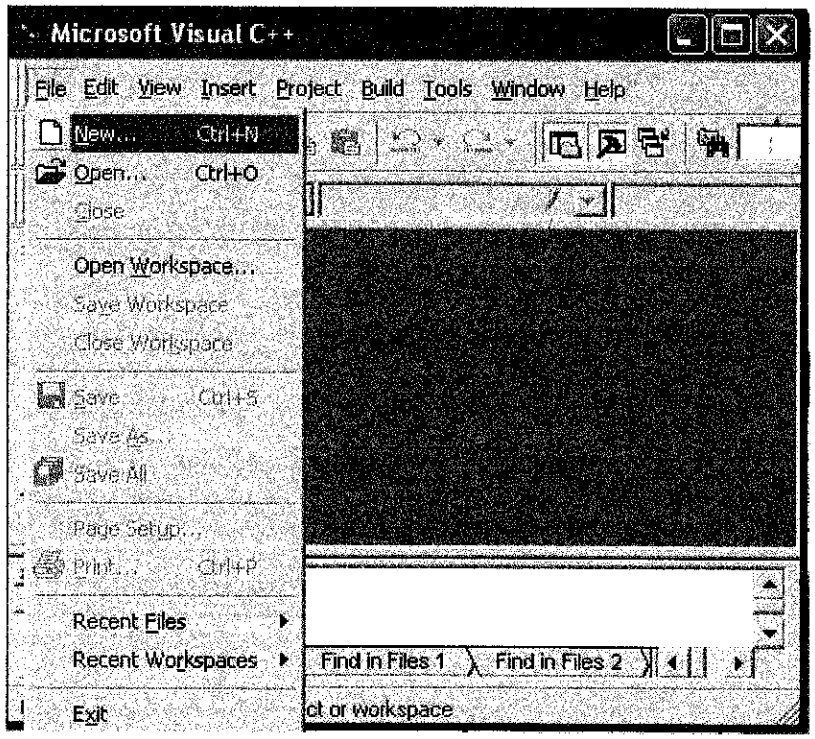
```
void Writeresults()
{
        ofstream outClientFile1("Test.txt", ios::app );
        if ( !outClientFile1 ) { // overload ! operator
                cerr<<" File could not be opened"<<endl;
                exit(1);
}
        for( xstate = x_min; xstate <= x_max ; xstate++ )
        {
           for( ystate = y_min; ystate <= y_max ; ystate++)
           {
                for ( action = 0 ; action < 4 ; action++ )
                {
                   outClientFile1<< xstate << setw(15) << ystate << setw(25) << action <<
                   setw(25) << num_eps_visit[ xstate ][ ystate ][ action ] <<setw(25) <<
                   Q[ xstate ][ ystate ][ action ]  << endl;
        }}}
}
```

ขั้นตอนการสร้าง grid เพื่อใช้ในการ simulator

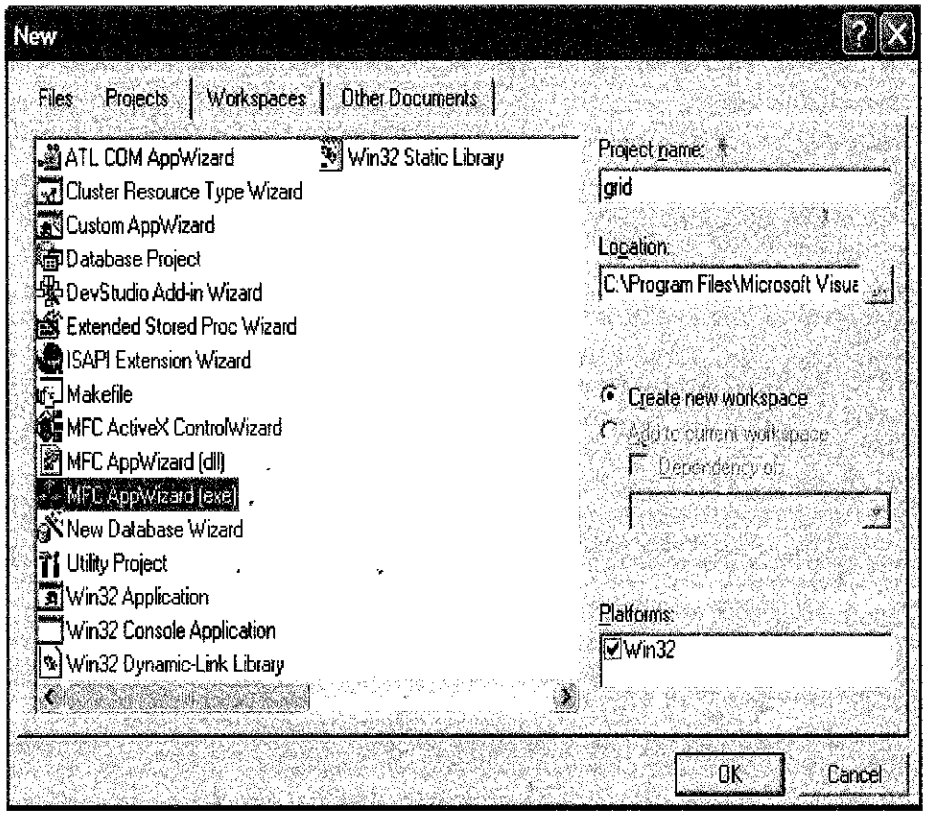
1.เปิดโปรแกรม visual c++ ขึ้นมา



2. สร้างโปรเจ็กต์เวิร์กสเปซ (Project Workspace) ขึ้นมาใหม่ซึ่งเป็นการกำหนดพื้นที่ในการเก็บโปรเจ็กที่ต้องการสร้างและกำหนดตัวเลือกต่างๆ โดยไปคลิกที่ File ⟶ New



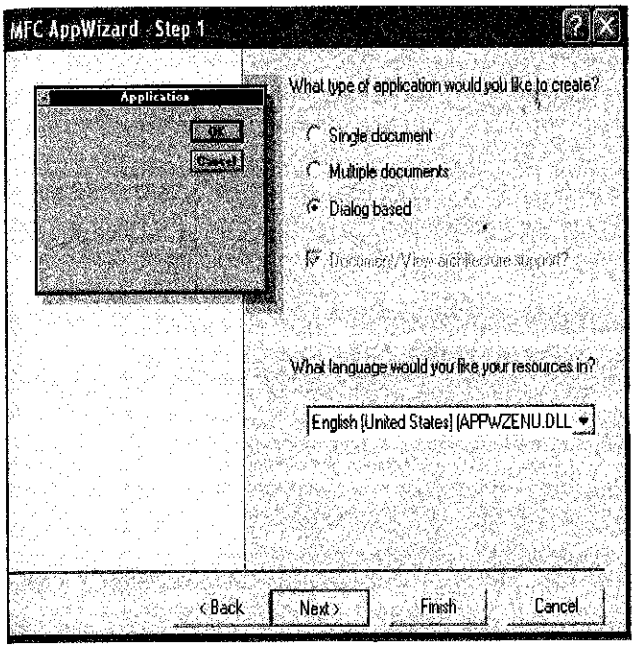
3. เมื่อไดอะล็อก New ปรากฏให้เลือกไปที่แท็บ Project และเลือกรูปแบบโปรเจ็กไปที่ MFC AppWizard (exe) จากนั้นกำหนดชื่อโปรเจ็กต์ในช่อง Project name: ในที่นี้ให้ชื่อ grid จากนั้นกดปุ่ม OK เพื่อเข้าสู่ AppWizard



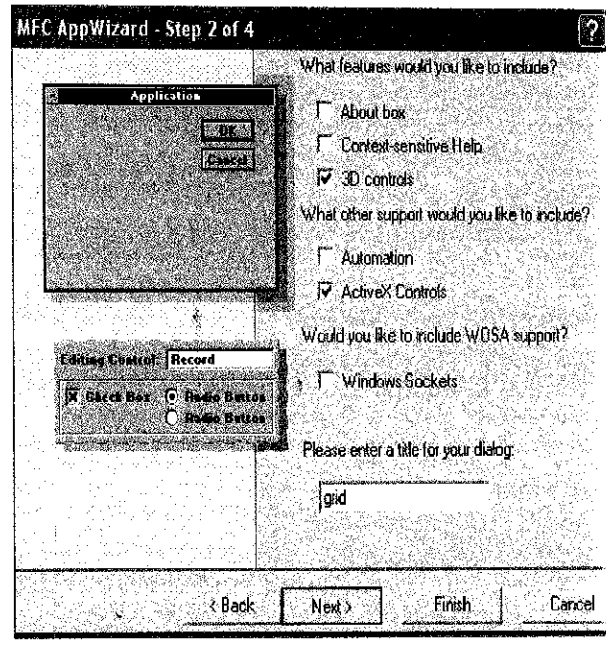
4. เมื่อเราพบกับไดอะล็อก AppWizard ให้เรากำหนดตัวเลือกต่างๆ เพื่อสร้างโปรแกรมแบบ Dialog-Based ดังนี้

   - MFC AppWizard Step 1 ให้เลือกไปที่ Dialog based และกดปุ่ม Next
   - MFC AppWizard Step 2 of 4 ให้เอาเครื่องหมายภายในช่อง About Box ออก และกดปุ่ม Next
   - MFC AppWizard Step 3 of 4 ให้คงตัวเดิมเอาไว้ และกดปุ่ม Next
   - MFC AppWizard Step 4 of 4 ให้กดปุ่ม Finish ได้ทันที แล้วเราจะพบไดอะล็อก New Project Information ให้กดปุ่ม OK เพื่อทำการสร้างโปรเจ็กต์

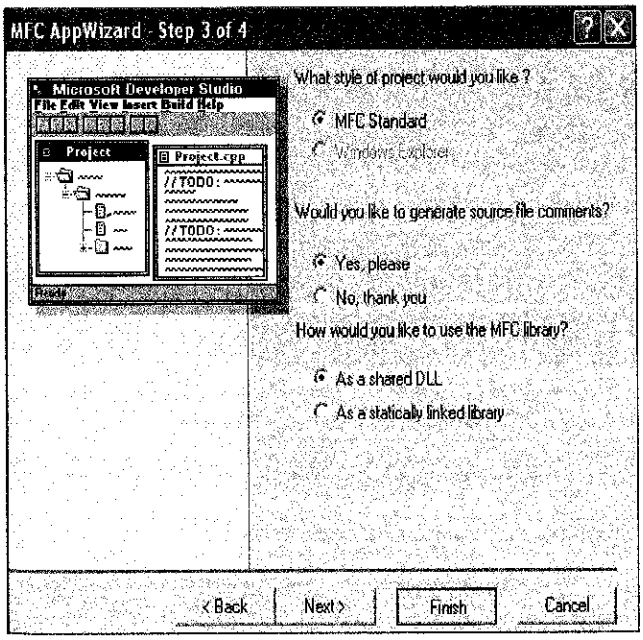
Step 1



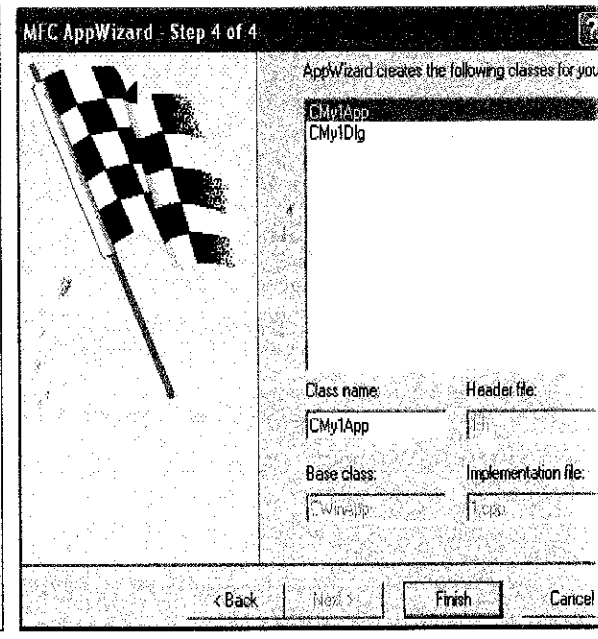
Step 2 of 4



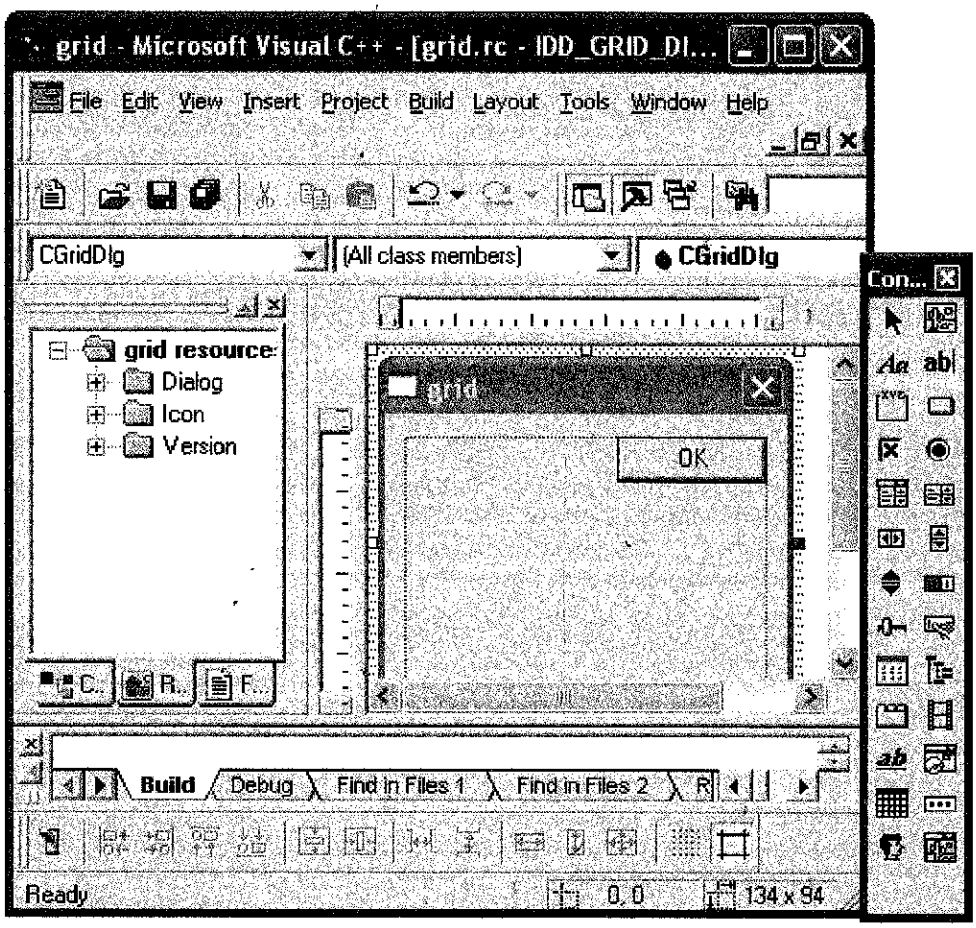
Step 3 of 4



Step 4 of 4



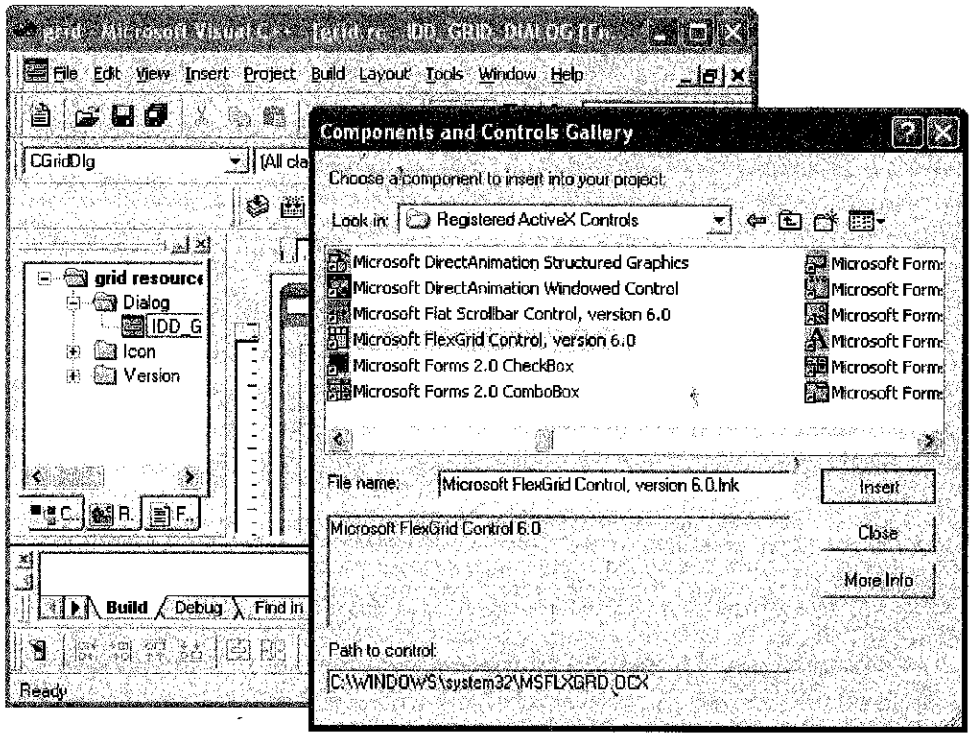
5. เราจะได้โปรเจ็กต์เวิร์กสเปซใหม่ที่ชื่อ grid พร้อมกับไดอะล็อกอิดิเตอร์ที่จะใช้สำหรับสร้างและแก้ไขไดอะล็อก



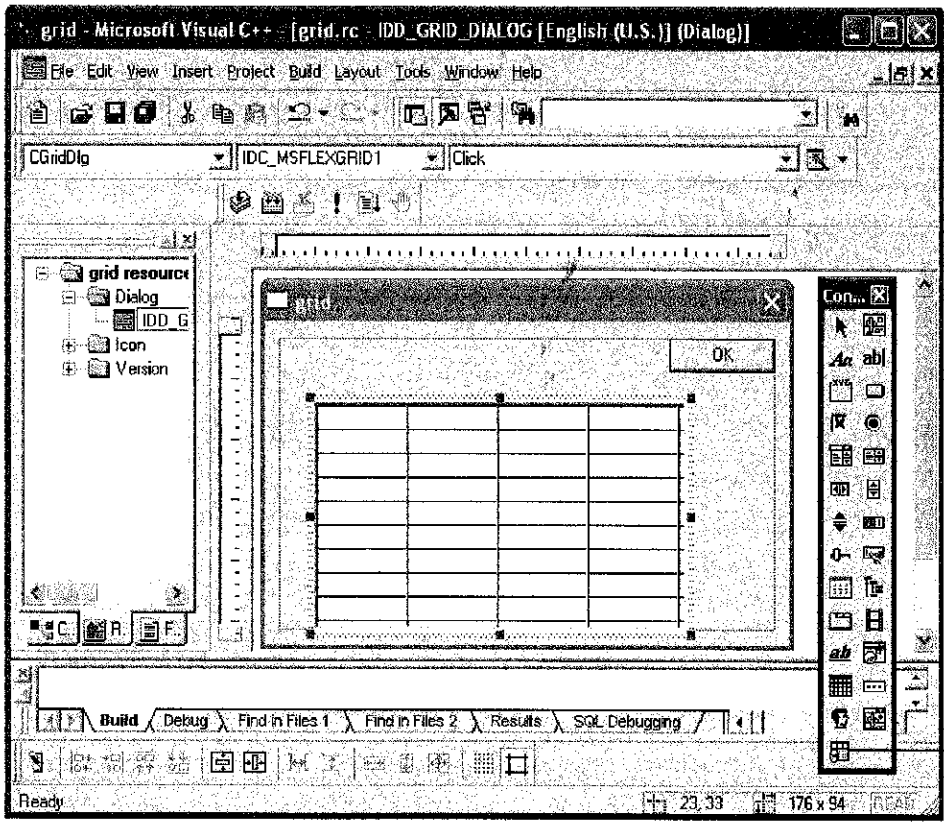
6. เราจะต้องเพิ่มไอคอน flex grid เข้ามาเพื่อใช้ในการสร้าง grid โดยสามารถปฏิบัติตามขั้นตอนดังนี้

- จากโปรเจ็กต์เวิร์กสเปซใหม่ที่ได้ให้เลือกไปที่แท็ป

  Project ⟶ Add To Project ⟶ Component and Controls...
- จากนั้นจะปรากฏหน้าต่าง Component and Controls Gallery ให้เลือกไปที่โฟลเดอร์ Registered ActiveX Controls ซึ่งในโฟลเดอร์นี้มีแอปพลิเคชั่นให้เลือกมากมาย แต่ในการสร้างกริชจะต้องเลือกใช้ Microsoft FlexGrid Control, version 6.0
- คลิก Insert เพื่อเพิ่ม flex grid ในไดอะล็อกอิดิเตอร์



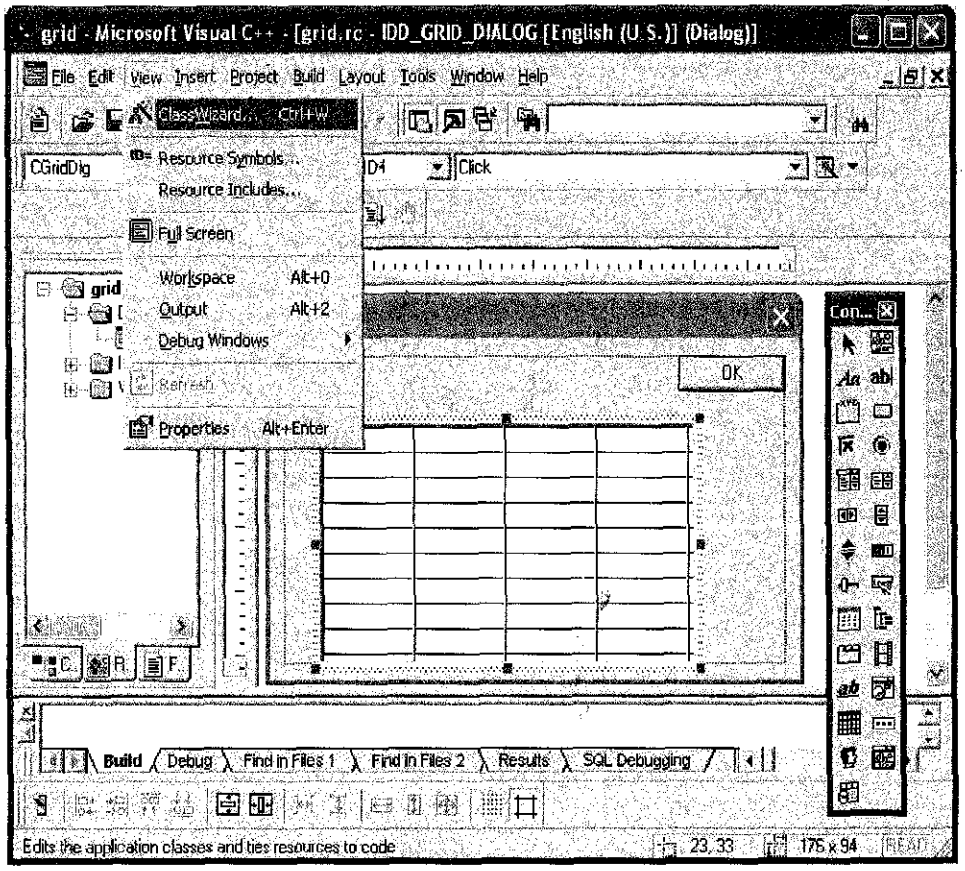
7. เราจะเห็นไอคอนเฟล็กกริช (flex grid) ปรากฏในไดอะล็อกอิดิเตอร์ จากนั้นนำไอคอนเฟล็กกริชจากคอนโทรลทูลบาร์มาวางลงในไดอะล็อก เราจะได้กริชดังภาพ

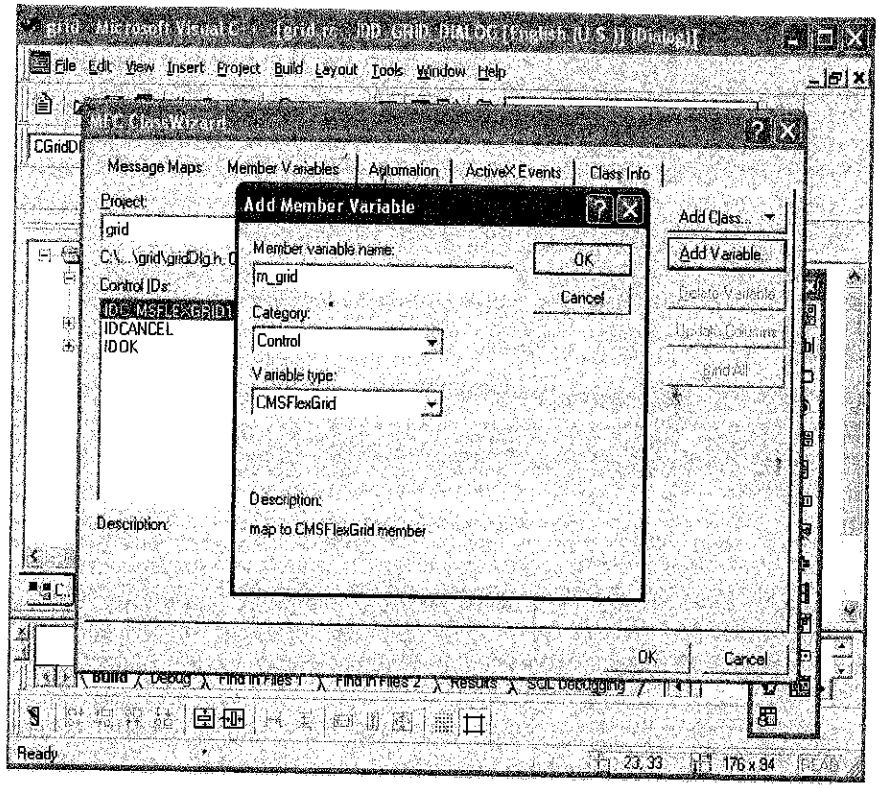


ไอคอน flex gr

8. ทำการสร้างตัวแปร m_grid เพื่อใช้ในการเขียนโค้ดโปรแกรม เพราะเราจะใช้ตัวแปรตัวนี้ในการกำหนดขนาดของกริช ซึ่งสามารถทำตามขั้นตอนดังนี้

   - เลือกไปที่ View ⟶ Class Wizard... (หรือกด Ctrl+W) ซึ่งจะปรากฏหน้าต่าง MFC Class Wizard ขึ้นมา
   - จากนั้นคลิกไปที่แท็ป Member Variables ในช่องของ Control IDs ให้คลิกที่ IDC_MSFLEXGRID1 แล้วคลิก Add Variable... เพื่อเป็นการเพิ่มตัวแปร
   - จะปรากฏหน้าต่าง Add Member Variable โดยในช่อง Member Variable name เราจะพิมพ์ชื่อตัวแปร m_grid แล้วคลิก OK
   - ตอนนี้เราจะได้ตัวแปรที่ใช้ในการเขียนโค้ดโปรแกรมแล้ว

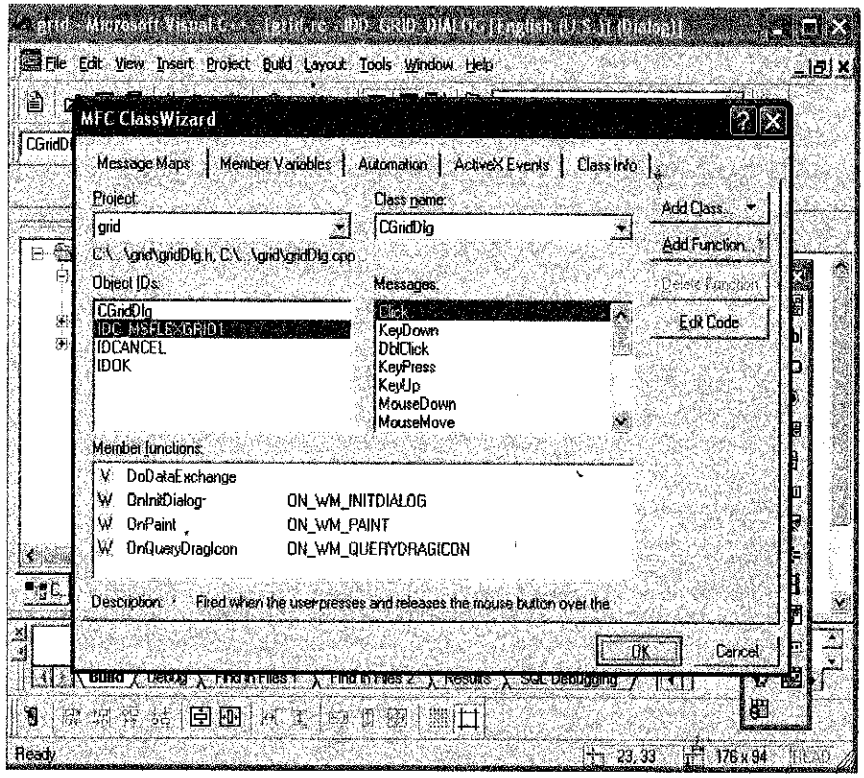




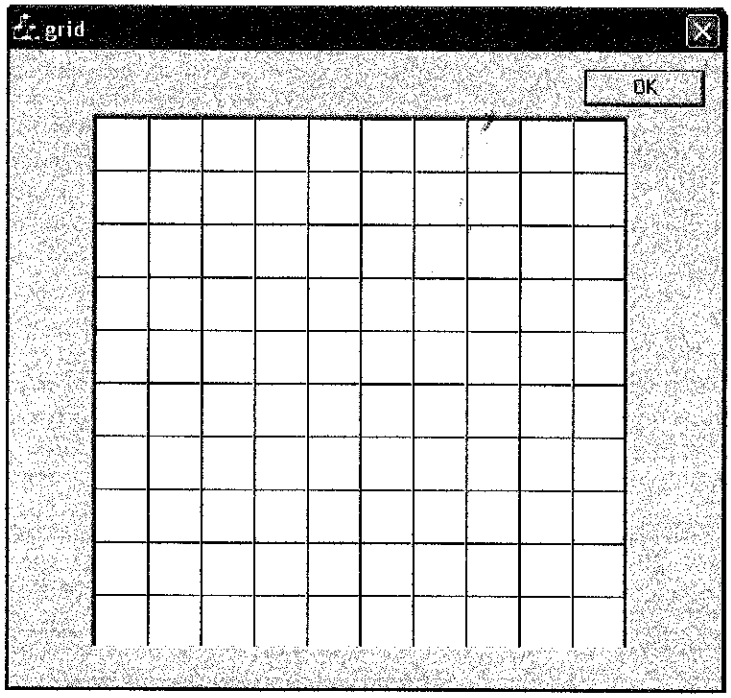
ตัวแปร m_grid ที่ใช้ในการเขียนโค้ดโปรแกรมเพื่อกำหนดขนาดของกริช

```
int i;              // counter
        m_grid.Clear();          // clear grid
        for(i=0;i<=9;i++)
        {
                m_grid.SetColWidth(i,500);
        }
        for(i=0;i<=9;i++)
        {
                m_grid.SetRowHeight(i,500);
        }
```

9. ใช้ Class Wizard แมปแมสเสจ โดยเราเรียก Class Wizard ขึ้นมา จากนั้นเลือกที่แท็ป Message Maps ในช่อง Class name จะเลือกไปที่ CGriDlg ซึ่งเราจะพบว่าในช่อง Object IDs จะปรากฏ IDs ขึ้นมา จากนั้นให้เราแมป IDC_MSFLEXGRID1 กับ Click



10. เมื่อเรา Build โปรเจ็กต์ และรันโปรแกรม เราจะพบหน้าต่างของกริช ดังรูป



บรรณานุกรม

[1] Sutton, R. S., and barto, A.G., Reinforcement Learning. Bradford Book, Cambridge, London, 1998