โครงงานทางวิศวกรรม

เรื่อง

ระบบหุ่นยนต์หลายตัวสำหรับการควบคุมวัตถุ

Multi-Robot System for Object Manipulation

โดย

นายรุ่งโรจน์ จินตเมธาสวัสดิ์ 5031061821

อาจารย์ที่ปรึกษาโครงงาน

ผู้ช่วยศาสตราจารย์ ดร.อรรถวิทย์ สุดแสง

อาจารย์ ดร.นัททีนิภานันท์

รายงานนี้เป็นส่วนหนึ่งของการศึกษาวิชา โครงงานวิศวกรรมคอมพิวเตอร์

หลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2553

บทคัดย่อ

ปัจจุบัน หุ่นยนต์ได้เข้ามามีบทบาทต่อชีวิตเรามากขึ้น อย่างไรก็ดี หุ่นยนต์ที่มีประสิทธิภาพมากขึ้นก็ย่อมมีราคาแพงขึ้น แต่ก็ไม่ได้ทำงานเร็วขึ้นมากนัก และถ้าหุ่นยนต์ตัวนั้นเกิดขัดข้อง งานที่เหลือที่หุ่นยนต์ยังทำค้างอยู่ ก็จะดำเนินต่อไปไม่ได้เลย ผู้พัฒนาพบว่า ปัญหาเรื่องค่าใช้จ่าย เวลาในการทำงาน และความทนทานต่อความผิดพลาด เป็นปัญหาที่สำคัญ และยังคงเกิดขึ้นกับระบบหุ่นยนต์ในปัจจุบัน ดังนั้น ผู้พัฒนาจึงได้ศึกษาและพัฒนาระบบหุ่นยนต์ ที่สามารถแก้ไขปัญหาเหล่านี้ได้

ระบบหุ่นยนต์ที่พัฒนาขึ้น ประกอบด้วยหุ่นยนต์มากกว่า 1 ตัว และทุกตัวต้องเป็นหุ่นยนต์แบบเดียวกัน (Homogeneous) โดยผู้พัฒนาจะใช้หุ่นยนต์ยี่ห้อ Surveyor SRV-1 การทำงานของระบบหุ่นยนต์นี้ คือ หุ่นยนต์ภายในกลุ่มต้องทำหน้าที่เคลื่อนย้ายวัตถุ 3 ชิ้น คือ ทรงกระบอกสีน้ำเงิน ทรงสี่เหลี่ยมสีเขียว และปริซึมหน้าตัดรูปสามเหลี่ยมสีแดง ไปยังเป้าหมายที่กำหนดไว้ให้ นอกจากนี้ หุ่นยนต์ภายในกลุ่มจะทำงานภายในสนามพื้นสีขาว และต้องติดต่อสื่อสารกันภายในเครือข่ายไร้สาย (Wireless LAN) เดียวกัน

จากการทดสอบระบบหุ่นยนต์หลายตัวที่พัฒนาขึ้น ผู้พัฒนาพบว่า ระบบหุ่นยนต์หลายตัว สามารถแก้ปัญหาได้ในเรื่องของค่าใช้จ่าย เวลาในการทำงาน รวมทั้งความทนทานต่อความผิดพลาด อย่างไรก็ตาม ความถูกต้องและประสิทธิภาพของการทำงาน ยังไม่เป็นที่น่าพอใจมากนัก เนื่องจากมีปัญหาทางกายภาพที่เกิดจากตัวหุ่นยนต์เอง และปัญหาที่เกิดจากสภาพแวดล้อมภายนอกของหุ่นยนต์

โครงงานวิศวกรรมคอมพิวเตอร์ ปีการศึกษา 2553
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ลายมือชื่อนิสิต_____
ลายมือชื่ออาจารย์ที่ปรึกษาโครงงาน_____
ลายมือชื่ออาจารย์ที่ปรึกษาโครงงาน_____

Abstract

Nowadays, robots deal with people's daily life more. However, spending much more money to invent a new robot does not always result in a smarter robot. Moreover, single robot can fail easily, and if it fails, the remaining tasks will be ignored. We noticed that budget, execution time, and fault tolerance are the critical problems, and they still occur in many robot systems. Therefore, we would like to study for implementing a new robot system, where they could provide solutions for these problems.

The developed robot system consists of more than one homogeneous Surveyor SRV-1 robot. The multi-robot system must bring 3 kinds of objects, a blue cylinder, a green cube, and a red triangle prism, to the specified destination. Besides, robots have to work in a white field, and communicate among themselves in group via the same wireless local area network (Wireless LAN).

The experimental results show us that multi-robot system can solve budget, execution time, and fault tolerance problems. However, the accuracy and efficiency are not considered to be good enough because of the physical problems of robots themselves and the harsh environment outside the robots.

Senior Project, Academic Year 2010

Department of Computer Engineering

Faculty of Engineering, Chulalongkorn University

Student's Signature_____

Project Advisor's Signature_____

Project Advisor's Signature_____

กิตติกรรมประกาศ

ในการจัดทำโครงงานครั้งนี้ ผู้พัฒนาได้รับความอนุเคราะห์จาก ผศ.ดร.อรรถวิทย์ สุดแสง และ อ.ดร.นัททที นิภานันท์ ในการให้คำแนะนำที่เป็นประโยชน์ต่อการทำโครงงาน การนำเสนอโครงงาน รวมทั้งการต่อยอดโครงงานไปสู่งานวิจัยอื่นๆ

ขอขอบคุณสำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ ที่ได้มอบทุนวิจัยระดับปริญญาตรีให้แก่ผู้พัฒนาตลอดระยะเวลาการศึกษา

ขอขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ที่ได้ประสิทธิ์ประสาทวิชาความรู้ตลอดระยะเวลาการศึกษา

ขอขอบคุณพี่ๆ จากห้องวิจัย ISL2 ที่ได้เอื้อเฟื้อสถานที่ และอุปกรณ์ต่างๆ ที่เกี่ยวข้องกับโครงงาน อีกทั้งยังให้คำแนะนำที่เป็นประโยชน์ และวิธีการแก้ไขปัญหาที่เกิดขึ้นในระหว่างการทำโครงงาน

สุดท้ายนี้ ขอขอบคุณบิดา มารดา ที่คอยให้กำลังใจเสมอมา และคอยจัดหาอุปกรณ์ในการทำโครงงานให้อย่างครบถ้วน

รุ่งโรจน์ จินตเมธาสวัสดิ์

สารบัญ

สารบัญรูป

สารบัญตาราง

บทที่ 1 บทนำ

1.1 ความเป็นมาของโครงการ

ทุกวันนี้ หุ่นยนต์ได้เข้ามามีบทบาทในการดำรงชีวิตของมนุษย์มากขึ้น ดังจะเห็นได้จากการนำหุ่นยนต์มาใช้งานในหลายด้านด้วยกัน เช่น ในงานการผลิตแบบอุตสาหกรรม งานช่วยเหลือผู้ประสบภัย งานลาดตระเวนพื้นที่เสี่ยงภัย ในภัตตาคาร หรือแม้กระทั่งนำหุ่นยนต์มาใช้ในบ้าน เพื่อให้ทำงานบ้านเล็กๆ น้อยๆ อีกด้วย อย่างไรก็ตาม หากต้องการให้หุ่นยนต์มีความสามารถหลากหลาย ราคาของตัวหุ่นยนต์เองก็ย่อมแพงขึ้นไปด้วย เพราะการประดิษฐ์หุ่นยนต์แบบนี้ อาจจำเป็นต้องใช้อุปกรณ์ชนิดพิเศษ และเมื่อหุ่นยนต์เกิดความผิดพลาด งานที่เหลือที่กำลังดำเนินอยู่นั้น ก็จะหยุดชะงักทันที

ในทางกลับกัน หากกระจายงานให้หุ่นยนต์หลายๆ ตัว โดยที่แต่ละตัวรับผิดชอบงานเพียงบางส่วนตามความสามารถของหุ่นยนต์ตัวนั้น โอกาสที่งานจะสำเร็จลุล่วงไปได้ก็ย่อมมีมากขึ้น เพราะหากมีหุ่นยนต์ตัวใดตัวหนึ่งเกิดขัดข้อง หุ่นยนต์ตัวอื่นก็ยังสามารถทำงานที่เหลือต่อไปได้ และเนื่องจากหุ่นยนต์แต่ละตัวถูกออกแบบมาให้มีความสามารถเท่าที่จำเป็น ดังนั้น ชิ้นส่วนที่เป็นองค์ประกอบของตัวหุ่นยนต์เอง ก็ย่อมมีราคาถูก เพราะสามารถหาซื้อได้ตามท้องตลาด

1.2 วัตถุประสงค์

เพื่อสร้างระบบหุ่นยนต์หลายตัว ที่ช่วยกันนำวัตถุ 3 ชิ้น คือ ทรงกระบอก ทรงสี่เหลี่ยม และปริซึมหน้าตัดรูปสามเหลี่ยม ไปรวมกัน ณ ตำแหน่งปลายทางที่ได้กำหนดไว้ให้

1.3 ขอบเขตของโครงงาน

1. ระบบหุ่นยนต์สามารถขนย้ายวัตถุได้ 3 แบบ คือ ทรงกระบอก ทรงสี่เหลี่ยม และปริซึมหน้าตัดรูปสามเหลี่ยม โดยวัตถุแต่ละชิ้นมีสีน้ำเงิน เขียว และแดงตามลำดับ
2. หุ่นยนต์แต่ละตัว สามารถรับรู้รายละเอียดของวัตถุทั้ง 3 แบบได้ โดยใช้กล้องที่ติดตั้งอยู่ด้านหน้าของตัวหุ่นยนต์ และสามารถค้นหาวัตถุได้ โดยการหมุนรอบตัวเองแล้วคอยตรวจสอบภาพที่ได้รับจากกล้องมาเรื่อยๆ จนกว่าจะเจอภาพของวัตถุที่ต้องการ หุ่นยนต์จึงหยุดหมุนรอบตัวเอง
3. หุ่นยนต์แต่ละตัว ไม่รู้ว่าตัวเองอยู่ ณ ตำแหน่งใดในสนาม หุ่นยนต์รู้เพียงแค่ว่า วัตถุแต่ละชิ้นเป้าหมายที่จะนำวัตถุไปวาง รวมทั้งหุ่นยนต์ตัวอื่นในสนาม อยู่ใกล้ หรือไกลจากตัวมันเองมากแค่ไหน
4. หุ่นยนต์ทุกตัว จะต้องรู้สถานะการทำงานของตนเองและหุ่นยนต์ตัวอื่น โดยอาศัยการส่งข้อความติดต่อกันเป็นระยะๆ ภายในเครือข่ายไร้สาย (Wireless LAN)
5. เมื่อมีหุ่นยนต์ตัวใดตัวหนึ่งเกิดขัดข้อง หุ่นยนต์ตัวอื่นจะต้องรับรู้ และเข้าไปทำงานแทนหุ่นยนต์ตัวนั้น

1.4 แนวทางการดำเนินงาน

1. **ศึกษาค้นคว้าข้อมูลที่เกี่ยวข้อง**

ในขั้นตอนนี้ จะทำการศึกษาการตั้งค่า IP ให้หุ่นยนต์ การเขียนโปรแกรมลงบนหุ่นยนต์ และการใช้กล้องของหุ่นยนต์

2. **จัดเตรียมวัสดุที่เกี่ยวข้องกับการทำโครงงาน**

ในขั้นตอนนี้ จะทำการเตรียมอุปกรณ์ที่เกี่ยวข้อง ได้แก่ การเตรียมสนาม วัตถุ และหุ่นยนต์ Surveyor SRV-1 รวมทั้งออกแบบแขนสำหรับจับวัตถุ ซึ่งจะถูกนำไปติดตั้งที่ด้านหน้าของหุ่นยนต์ในภายหลัง

3. **พัฒนาโปรแกรมสำหรับจัดการระบบหุ่นยนต์หลายตัว**

ในขั้นตอนนี้ จะทำการศึกษา Library ที่ใช้สำหรับการควบคุมหุ่นยนต์ และนำ Library มาพัฒนาต่อยอดให้เป็นโปรแกรมที่ใช้ควบคุมระบบหุ่นยนต์ หรือที่เรียกว่า Console อีกทีหนึ่ง

4. **พัฒนาระบบตรวจจับวัตถุ**

ในขั้นตอนนี้ จะทำการพัฒนาส่วนที่ใช้ในการตรวจจับชนิดของวัตถุ โดยพิจารณาจากสีของวัตถุในภาพ ถ้าตรวจพบสีแดง แสดงว่าพบวัตถุปริซึมหน้าตัดรูปสามเหลี่ยม ถ้าตรวจพบสีเขียว แสดงว่าพบวัตถุทรงสี่เหลี่ยม และถ้าตรวจพบสีน้ำเงิน แสดงว่าพบวัตถุทรงกระบอก นอกจากนี้ ยังทำการพัฒนาส่วนที่ใช้ในการหาทิศทางของวัตถุเทียบกับตัวหุ่นยนต์อีกด้วย โดยพิจารณาจากตำแหน่งของวัตถุที่ปรากฏในภาพ

5. **ออกแบบวิธีการเคลื่อนที่หุ่นยนต์**

ในขั้นตอนนี้ จะทำการออกแบบวิธีการเคลื่อนที่ของหุ่นยนต์ ในสถานการณ์ต่างๆ เพื่อให้หุ่นยนต์เคลื่อนที่ไปยังตำแหน่งปลายทางได้ถูกต้องและไม่ชนสิ่งกีดขวาง

6. **พัฒนาหุ่นยนต์หนึ่งตัว เพื่อให้สามารถนำวัตถุไปวางไว้ ณ ตำแหน่งที่ถูกต้องได้**

ในขั้นตอนนี้ จะทำการทดลองเพื่อให้หุ่นยนต์เพียงหนึ่งตัว นำวัตถุไปวางไว้ ณ ตำแหน่งปลายทางได้ถูกต้อง

7. **พัฒนาระบบหุ่นยนต์หลายตัว เพื่อให้สามารถนำวัตถุไปวางไว้ ณ ตำแหน่งที่ถูกต้องได้**

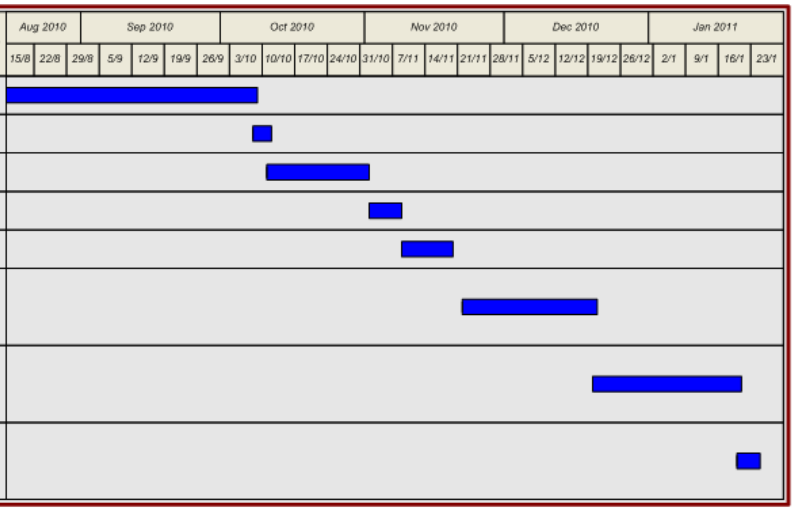
เมื่อหุ่นยนต์หนึ่งตัวทำงานได้ถูกต้องแล้ว ก็จะทำการเพิ่มจำนวนหุ่นยนต์ให้มีตั้งแต่ 2 ตัวขึ้นไป โดยทำการพัฒนาให้หุ่นยนต์มีการประสานงานกัน เพื่อช่วยกันนำวัตถุไปวางไว้ในตำแหน่งปลายทางได้ถูกต้อง

8. **เปรียบเทียบผลการทำงานที่ได้ ระหว่างระบบหุ่นยนต์หลายตัว กับหุ่นยนต์ตัวเดียว**

ทำการเปรียบเทียบประสิทธิภาพที่ได้ ระหว่างหุ่นยนต์ตัวเดียว กับระบบหุ่นยนต์หลายตัว และทำการเปรียบเทียบเวลาที่ใช้เคลื่อนย้ายวัตถุทั้งหมดในสนามไปยังตำแหน่งปลายทาง ระหว่างหุ่นยนต์ตัวเดียว กับหุ่นยนต์หลายตัว

1.5 ขั้นตอนและระยะเวลาการดำเนินงาน

ID	Task Name	Start	Finish	Duration
1	ศึกษาค้นคว้าข้อมูลที่เกี่ยวข้อง	8/16/2010	10/8/2010	8w
2	จัดเตรียมวัสดุที่เกี่ยวข้องกับการทำโครงงาน	10/8/2010	10/11/2010	.4w
3	พัฒนาโปรแกรมสำหรับจัดการระบบหุ่นยนต์หลายตัว	10/11/2010	11/1/2010	3.2w
4	พัฒนาระบบตรวจจับวัตถุ	11/2/2010	11/8/2010	1w
5	ออกแบบวิธีการเคลื่อนที่หุ่นยนต์	11/9/2010	11/19/2010	1.8w
6	พัฒนาหุ่นยนต์หนึ่งตัว เพื่อให้สามารถนำวัตถุไปวางไว้ ณ ตำแหน่งที่ถูกต้องได้	11/22/2010	12/20/2010	4.2w
7	พัฒนาระบบหุ่นยนต์หลายตัว เพื่อให้สามารถนำวัตถุไปวางไว้ ณ ตำแหน่งที่ถูกต้องได้	12/20/2010	1/20/2011	4.8w
8	เปรียบเทียบผลการทำงานที่ได้ ระหว่างระบบหุ่นยนต์หลายตัว กับหุ่นยนต์ตัวเดียว	1/20/2011	1/24/2011	.6w



รูปที่ 1 Gantt chart แสดงขั้นตอนและระยะเวลาการดำเนินงาน

1.6 ประโยชน์ที่คาดว่าจะได้รับ

1. ระบบหุ่นยนต์หลายตัวสามารถลดข้อผิดพลาดที่เกิดจากการทำงานของหุ่นยนต์เพียงตัวเดียวได้
2. ระบบหุ่นยนต์หลายตัวสามารถแก้ปัญหาข้อจำกัดทางด้านฮาร์ดแวร์ที่มีอยู่ในปัจจุบันได้
3. ระบบหุ่นยนต์หลายตัวสามารถลดค่าใช้จ่ายในการออกแบบและประดิษฐ์หุ่นยนต์ได้
4. ระบบหุ่นยนต์หลายตัว มีความยืดหยุ่นสูง กล่าวคือ สามารถเพิ่มจำนวนของหุ่นยนต์ภายในกลุ่มได้ โดยไม่จำเป็นต้องแก้ไขระบบมากนัก

บทที่ 2 ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 งานวิจัยเบื้องต้น

2.1.1 Heterogeneous Multi-Robot Cooperation [2]

งานวิจัยนี้ ได้ทำการวิเคราะห์ข้อผิดพลาดของระบบหุ่นยนต์หลายตัว อันเกิดจากการที่หุ่นยนต์เกิดขัดข้องในระหว่างการทำงาน หุ่นยนต์ไม่มีความสามารถเพียงพอในการทำงาน หรือสภาพแวดล้อมได้เปลี่ยนแปลงไปจากเดิม งานวิจัยนี้ได้ทำการพัฒนาสถาปัตยกรรมที่ชื่อว่า ALLIANCE ซึ่งเป็นสถาปัตยกรรมของระบบหุ่นยนต์หลายตัว ที่ออกแบบมาเพื่อให้สามารถตรวจสอบและแก้ไขข้อผิดพลาดที่เกิดขึ้นได้ โดยอาศัยหลักการของการเผยแพร่ข้อมูลสถานะการทำงานของหุ่นยนต์แต่ละตัว ไปให้หุ่นยนต์ทุกตัวในกลุ่มรับทราบ นอกจากนี้ งานวิจัยยังได้พัฒนาสถาปัตยกรรมหุ่นยนต์หลายตัวที่ชื่อว่า L-ALLANCE ซึ่งเป็นสถาปัตยกรรมที่เปิดโอกาสให้หุ่นยนต์เรียนรู้จากประสบการณ์ที่พบก่อนหน้านั้นได้ ทำให้หุ่นยนต์ทำงานได้มีประสิทธิภาพมากขึ้น และลดปริมาณงานในการตั้งค่าพารามิเตอร์ให้กับหุ่นยนต์แต่ละตัว

2.1.2 Cooperative Localization and Control for Multi-Robot Manipulation [3]

งานวิจัยนี้ ได้ออกแบบโมเดลในการหาตำแหน่ง และทิศทางของหุ่นยนต์ภายในกลุ่ม โดยให้หุ่นยนต์ตัวหนึ่งเป็นจุดอ้างอิงของหุ่นยนต์ทั้งหมด การหาตำแหน่งของหุ่นยนต์ จะมีประเด็นหลักอยู่ 2 อย่าง คือ

1. *การหาตำแหน่ง และทิศทางของหุ่นยนต์แต่ละตัว* หุ่นยนต์แต่ละตัวจะรับภาพจากกล้องที่สามารถมองเห็นได้โดยรอบ (Omni-Camera) ที่ติดตั้งบนตัวหุ่นยนต์ จากนั้นภาพที่รับมา จะถูกนำไปประมวลผลเพื่อหาตำแหน่ง และทิศทางของหุ่นยนต์ตัวอื่นอีกทีหนึ่ง
2. *การรักษาการจัดวางตัวของหุ่นยนต์แต่ละตัว* หุ่นยนต์แต่ละตัวจะต้องอยู่ ณ ตำแหน่งที่เหมาะสม เพื่อให้การจัดวางตัวของหุ่นยนต์ภายในกลุ่มไม่ผิดเพี้ยนจนเกินไป เช่น ถ้าหุ่นยนต์ทั้ง 3 ตัว จะต้องจัดวางตัวกันเป็นรูปสามเหลี่ยมหน้าจั่ว หุ่นยนต์แต่ละตัวจะต้องอยู่ ณ ตำแหน่งที่เป็นจุดยอดของรูปสามเหลี่ยม

2.1.3 Multi-Robot Manipulation via Caging in Environments with Obstacle [4]

งานวิจัยนี้ ได้ทำการออกแบบอัลกอริทึมในการเคลื่อนย้ายวัตถุโดยใช้ระบบหุ่นยนต์หลายตัว โดยหุ่นยนต์แต่ละตัว จะไม่รู้ขนาดของกลุ่ม และความสามารถของหุ่นยนต์ตัวอื่นเลย รวมทั้งจะต้องมีการสื่อสารกันภายในกลุ่มให้น้อยที่สุด การทำเช่นนี้ ทำให้ระบบหุ่นยนต์หลายตัวมีความทนต่อข้อผิดพลาดสูง ง่ายต่อการควบคุม และเสียเวลาน้อยในการสื่อสาร หุ่นยนต์แต่ละตัวภายในระบบนี้ จะมีสถานะอยู่ 3 สถานะด้วยกัน คือ

1. *Approach* เป็นสถานะที่ทำการค้นหาวัตถุ ในขณะเดียวกันก็ต้องหลีกเลี่ยงการชนสิ่งกีดขวาง และหุ่นยนต์ตัวอื่นด้วย
2. *Surround* เป็นสถานะที่หุ่นยนต์ตรวจพบวัตถุ และกำลังจะไปควมคุมวัตถุชิ้นนั้น ในขณะเดียวกันก็ต้องหลีกเลี่ยงการชนสิ่งกีดขวาง และหุ่นยนต์ตัวอื่นด้วย
3. *Transport* เป็นสถานะที่หุ่นยนต์กำลังเคลื่อนย้ายวัตถุไปยังตำแหน่งเป้าหมาย

2.1.4 Abstractions and Algorithms for Cooperative Multiple Robot Planar Manipulation [5]

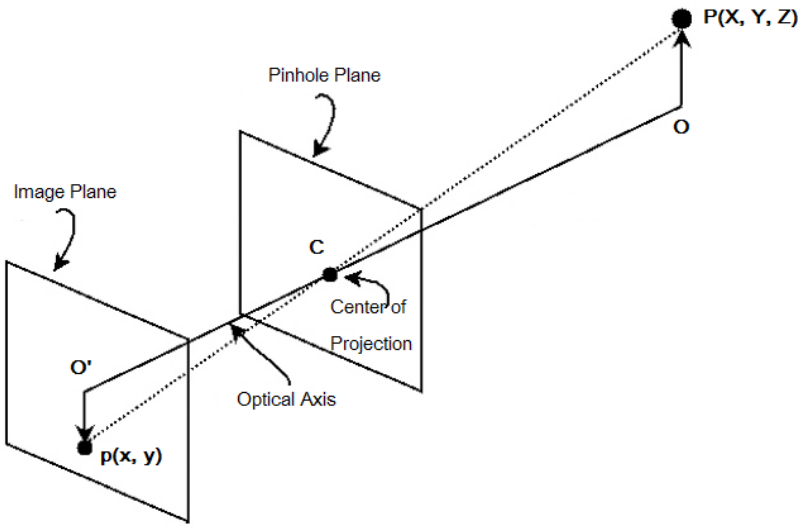
งานวิจัยนี้ ได้ทำการออกแบบอัลกอริทึมอย่างง่าย ในการเคลื่อนย้ายวัตถุบนระนาบ 2 มิติ ที่มีสิ่งกีดขวางอยู่ โดยใช้หุ่นยนต์ 2 ตัว อัลกอริทึมนี้มีความทนทานต่อสภาวะการณ์ที่ไม่ปกติ เช่น แรงเสียดทานบนพื้น หรือข้อจำกัดของเซนเซอร์บนตัวหุ่นยนต์

2.1.5 Behavior-Based Multi-Robot Collaboration for Autonomous Construction Tasks [6]

งานวิจัยนี้ ได้ทำการออกแบบระบบ RCC (Robot Construction Crew) สำหรับระบบหุ่นยนต์หลายตัว โดยที่หุ่นยนต์แต่ละตัวมีหน้าที่แตกต่างกัน RCC เป็นระบบที่มีพื้นฐานมาจากการแบ่งงานเป็นลำดับชั้น งานระดับล่าง จะถูกควมคุมโดยงานระดับบนอีกทีหนึ่ง เช่น การสั่งให้หุ่นยนต์เคลื่อนแขน (งานระดับบน) จะทำให้มอเตอร์หมุน (งานระดับล่าง) แต่งานวิจัยนี้ได้ใช้วิธีลดความผิดพลาดที่เกิดขึ้นในระบบโดยการควบคุมความเร็วของมอเตอร์ให้เหมาะสม

2.2 การฉายภาพแบบทัศนมิติ (Perspective Projection)

หลักการของการได้ภาพมานั้น แสดงได้ดังรูปที่ 2



รูปที่ 2 การเกิดภาพในกล้องรูเข็ม (ที่มา: http://jde.gsyc.es/index.php/JDEROBOT_4.3.0:Manual)

จากรูป จะพบว่า รังสีของแสง (Optical Ray) ทุกเส้น จะผ่านจุดศูนย์กลางของการฉาย (Center of Projection) และจะไปตกกระทบที่ฉากรับภาพ (Image Plane) โดยภาพที่ได้จากการฉายภาพเช่นนี้ จะเป็นภาพหัวกลับ และเราสามารถคำนวณหาขนาดของภาพได้ โดยใช้หลักการของสามเหลี่ยมคล้าย ดังนี้

$$\frac{O'C}{O'P} = \frac{OC}{OP}$$

โดยที่ O'C แทนความยาวโฟกัสของกล้อง OC แทนระยะห่างระหว่างวัตถุกับเลนส์กล้อง

O'P แทนขนาดของภาพ OP แทนขนาดของวัตถุ

จากความสัมพันธ์ดังกล่าว จะพบว่า เมื่อวัตถุอยู่ไกลกล้อง หากเลื่อนวัตถุให้เข้าใกล้กล้อง ขนาดของภาพวัตถุจะเพิ่มไม่มากนัก แต่ถ้าวัตถุอยู่ใกล้กล้อง หากเลื่อนวัตถุให้ใกล้กล้องเข้ามาอีก ขนาดของภาพจะใหญ่ขึ้นมาก ดังนั้น เราสามารถคำนวณได้ว่า วัตถุอยู่ห่างจากหุ่นยนต์เท่าใด โดยคำนวณระยะห่าง OC ตามรูปด้านบน และดูอัตราการเปลี่ยนแปลงของขนาดของภาพ

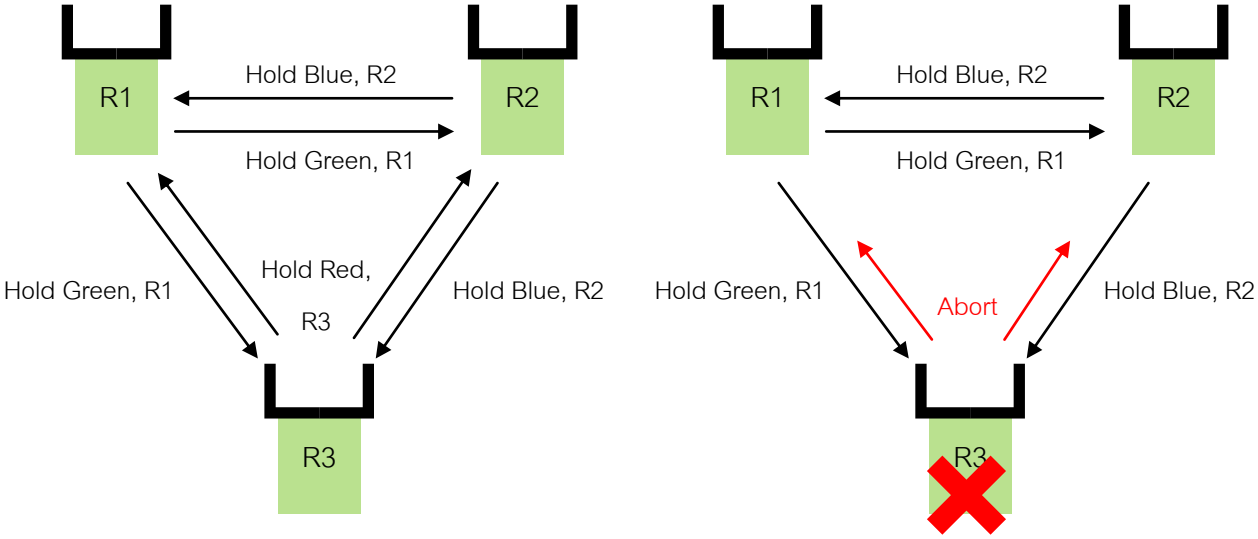
อย่างไรก็ตาม เมื่อนำวิธีการนี้ไปใช้จริง กลับไม่ได้ผลตามที่คาดไว้ ทั้งนี้เพราะเมื่อหุ่นยนต์เคลื่อนที่เข้าใกล้วัตถุมากขึ้น สีของวัตถุก็ย่อมเปลี่ยนไป ทำให้หุ่นยนต์ไม่สามารถตรวจจับวัตถุได้อย่างถูกต้อง ดังเช่นรูปที่ 3 เป็นผลทำให้เกิดความคลาดเคลื่อนในการหาวัตถุ ดังนั้น ผู้พัฒนาจึงใช้วิธีการหาความกว้างโดยประมาณของวัตถุ ควบคู่ไปกับขนาดโดยประมาณของวัตถุ เพื่อให้มีความแม่นยำมากขึ้น การหาความกว้างและขนาดโดยประมาณของวัตถุ จะกล่าวถึงในหัวข้อที่ 2.4 การตรวจหาวัตถุ (Object Detection)

รูปที่ 3 (ซ้าย) เมื่อหุ่นยนต์อยู่ไกลวัตถุ วัตถุจะมีสีสด

(ขวา) เมื่อหุ่นยนต์อยู่ใกล้วัตถุ วัตถุจะมีสีซีดลง

2.3 การทำความตกลงแบบประชามติ (Consensus Agreement) [1]

ระบบหุ่นยนต์หลายตัวที่ออกแบบ สามารถทำงานได้แม้กระทั่งในกรณีที่เกิดข้อผิดพลาดกับหุ่นยนต์ภายในกลุ่ม กล่าวคือ หากหุ่นยนต์ตัวใดตัวหนึ่งเกิดขัดข้องแบบที่หุ่นยนต์ตัวอื่นไม่สามารถติดต่อกับหุ่นยนต์ตัวนี้ได้ หุ่นยนต์ตัวที่เหลือจะต้องไปช่วยทำหน้าที่แทนหุ่นยนต์ตัวที่ขัดข้อง การตรวจสอบว่าหุ่นยนต์ตัวใดในระบบขัดข้องในระหว่างการทำงาน สามารถทำได้โดยใช้วิธีการที่เรียกว่า การทำความตกลงแบบประชามติ (Consensus Agreement)

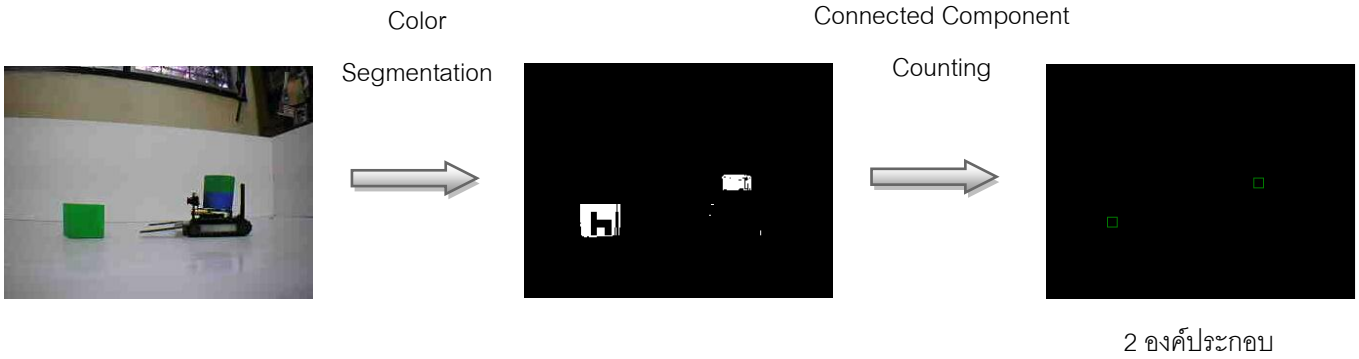


รูปที่ 4 (ซ้าย) การทำความตกลงแบบประชามติ (ขวา) แสดงกรณีที่มีหุ่นยนต์ภายในกลุ่มเกิดขัดข้อง

จากรูป ให้ R1, R2 และ R3 แทนหุ่นยนต์ทั้ง 3 ตัว หุ่นยนต์แต่ละตัว จะส่งข้อความกระจายไปให้หุ่นยนต์ที่เหลือภายในกลุ่มรับทราบ ทั้งข้อมูลของงานที่ทำ และสถานะการทำงานที่บอกว่าตัวมันเองยังสามารถทำงานได้อยู่ ดังรูปที่ 4 ทางด้านซ้าย หุ่นยนต์แต่ละตัว จะส่งข้อความไปบอกหุ่นยนต์ทุกตัวในกลุ่ม ว่าหุ่นยนต์ตัวนั้นกำลังจะไปหยิบวัตถุสีอะไร เพื่อที่ว่าหุ่นยนต์ตัวอื่นจะได้ไม่ทำหน้าที่ซ้ำกับหุ่นตัวนั้น แต่หากมีหุ่นยนต์ตัวใดตัวหนึ่งเกิดขัดข้อง (จากรูปที่ 4 ทางด้านขวา คือหุ่นยนต์ R3) หุ่นยนต์ตัวนั้นก็จะไม่สามารถส่งข้อความกระจายไปให้หุ่นยนต์ตัวที่เหลือได้ เมื่อหุ่นยนต์ตัวที่เหลือ ไม่ได้รับข้อความจากหุ่นยนต์ตัวนั้นภายในเวลาที่กำหนด ก็จะทราบทันทีว่าหุ่นยนต์ตัวนั้นเกิดข้อผิดพลาด หุ่นยนต์ตัวที่ว่างงานอยู่ ก็จะไปรับหน้าที่แทนหุ่นยนต์ตัวที่เกิดขัดข้อง

2.4 การตรวจหาวัตถุ (Object Detection)

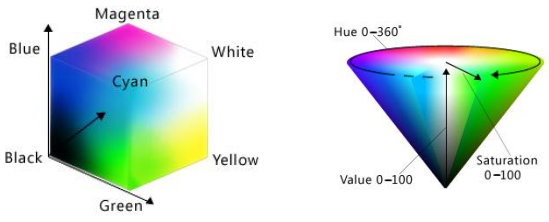
ขั้นตอนการตรวจหาวัตถุ สามารถแบ่งได้เป็น 2 ขั้นตอนใหญ่ๆ คือ ขั้นตอนการแบ่งแยกสี (Color Segmentation) และขั้นตอนการนับจำนวนองค์ประกอบในรูปภาพ (Connected Component Counting)



รูปที่ 5 ขั้นตอนการตรวจหาวัตถุ

2.4.1 การแบ่งแยกสี (Color Segmentation)

เนื่องจากวัตถุแต่ละชนิด มีสีที่แตกต่างกัน และสนามของหุ่นยนต์มีพื้นและขอบเป็นสีขาวทั้งหมด ดังนั้น ผู้พัฒนาจึงใช้วิธีการจำแนกวัตถุ โดยใช้สีของวัตถุเป็นเกณฑ์ในการแบ่ง อย่างไรก็ตาม ภาพที่รับเข้ามาจากกล้อง จะเป็นภาพที่อยู่ในปริภูมิสี RGB โดยที่ R (Red) แทนความเข้มของสีแดง G (Green) แทนความเข้มของสีเขียว และ B (Blue) แทนความเข้มของสีน้ำเงิน ซึ่งปริภูมิสี RGB ไม่เหมาะสำหรับใช้จำแนกสี ผู้พัฒนาจึงต้องแปลงปริภูมิสีให้เป็นแบบ HSV แทน โดยที่ H (Hue) แทนค่าสี S (Saturation) แทนความอิ่มตัวของสี และ V (Value) แทนความสว่างของสี ปริภูมิ RGB และ HSV แสดงได้ดังรูปด้านล่าง

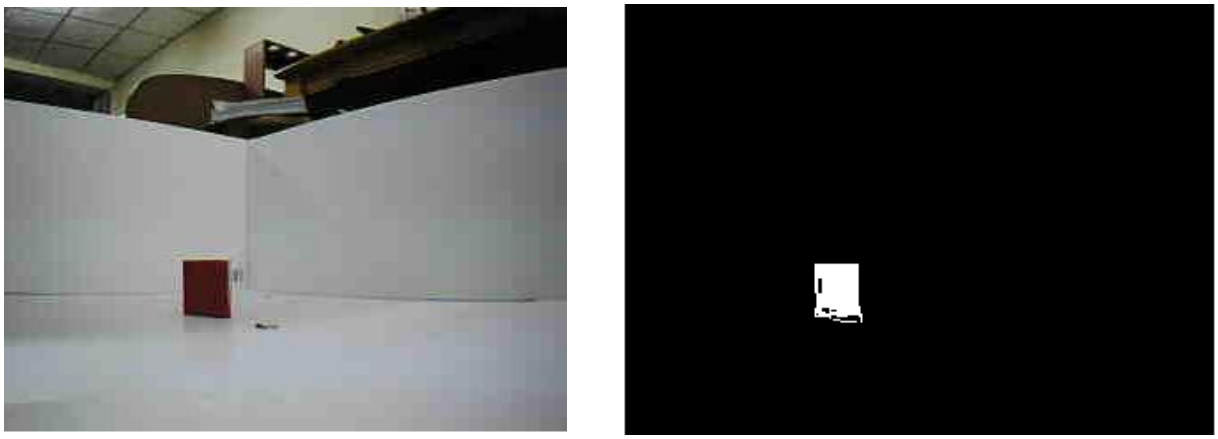


รูปที่ 6 (ซ้าย) ปริภูมิสี RGB (ขวา) ปริภูมิสี HSV

(ที่มา: http://msdn.microsoft.com/en-us/library/aa511283.aspx)

ในทางทฤษฎี ค่า H เพียงอย่างเดียว ก็เพียงพอสำหรับการบอกสีของวัตถุ แต่เมื่อนำมาใช้จริง กลับไม่ได้ผลตามที่คาดหวังไว้ เพราะบริเวณพื้นหรือขอบสนามที่อยู่ใกล้ๆ วัตถุมาก ก็จะมีค่า H ที่ใกล้เคียงกับวัตถุด้วย ดังนั้น ผู้พัฒนาจึงต้องใช้ทั้งค่า H, S และ V ในการจำแนกวัตถุ

การแยกภาพวัตถุออกจากพื้นหลัง ทำได้โดยพิจารณาว่าจุดภาพ (Pixel) ใดในภาพ เป็นจุดภาพของวัตถุที่กำลังพิจารณาอยู่ โดยดูว่าจุดภาพที่เลือกมานั้น มีค่า H, S และ V อยู่ในช่วงเดียวกับค่า H, S และ V ของวัตถุนั้นหรือไม่ ถ้าค่าทั้งสามอยู่ในช่วงเดียวกัน แสดงว่าจุดภาพนั้นเป็นส่วนหนึ่งของภาพวัตถุ ให้แทนด้วยจุดภาพสีขาว ในทางกลับกัน ถ้ามีค่าใดค่าหนึ่งไม่อยู่ในช่วง ก็แสดงว่าจุดภาพนั้นไม่ได้เป็นส่วนหนึ่งของภาพวัตถุ ให้แทนด้วยจุดภาพสีดำ รูปที่ 7 แสดงการแยกวัตถุออกจากพื้นหลังด้วยวิธีการแบ่งแยกสี



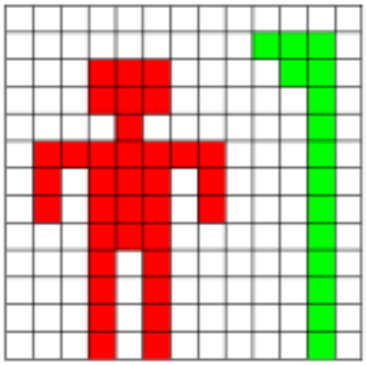
รูปที่ 7 (ซ้าย) ภาพวัตถุสีแดง

(ขวา) ภาพวัตถุหลังจากผ่านกระบวนการแบ่งแยกสี ซึ่งเป็นภาพขาวดำ

2.4.2 **การนับจำนวนองค์ประกอบในรูปภาพ** (Connected Component Counting)

หลังจากที่ได้ภาพขาวดำจากการบวนการแบ่งแยกสีแล้ว กระบวนการต่อไป คือ การนับจำนวนองค์ประกอบภายในภาพเดียวกัน การหาจำนวนองค์ประกอบ ทำได้โดยการกวาดจุดภาพทุกจุดภาพ จากซ้ายไปขวา และบนลงล่าง เมื่อใดที่เจอจุดสีขาว ก็จะพิจารณาจุดภาพรอบๆ จุดนั้น ทั้ง 4 ทิศ ว่าจุดภาพทิศใดบ้างที่มีสีขาว ถ้าจุดภาพในทิศใดมีสีขาว ก็จะทำการแผ่ขยายไปต่อในทิศทางนั้น ทำเช่นนี้ไปเรื่อยๆ จนกระทั่งไม่

สามารถแผ่ขยายไปได้อีก (จุดภาพทั้ง 4 ทิศเป็นสีดำหมด) การแผ่ขยายจึงหยุด และจะได้ว่าในองค์ประกอบนั้นมีจุดภาพใดบ้าง และมีจำนวนจุดภาพกี่จุด ข้อมูลเหล่านี้ ทำให้สามารถหาตำแหน่งจุดศูนย์กลาง ขนาด และมิติขององค์ประกอบนั้นได้ รูปที่ 8 แสดงจำนวนองค์ประกอบในรูปภาพ โดยที่จุดภาพสีเดียวกัน จะอยู่ในองค์ประกอบเดียวกัน



รูปที่ 8 ภาพขนาด 13 x 13 pixels ที่มี 2 องค์ประกอบ

(ที่มา: http://en.wikipedia.org/wiki/File:Two-pass_connected_component_labeling.svg)

ข้อมูลของแต่ละองค์ประกอบ จะประกอบด้วย ตำแหน่งจุดศูนย์กลาง (cx, cy) ขนาด (A) และมิติ (ความกว้าง (w), ความสูง (h)) ให้ C เป็นองค์ประกอบ และให้ I[r, c] เป็นจุดภาพ ณ ตำแหน่ง (r, c) จะได้ว่า

$$A = n(C)$$

$$cx = \frac{\sum_{I[r,c] \in C} c}{A} \qquad cy = \frac{\sum_{I[r,c] \in C} r}{A}$$

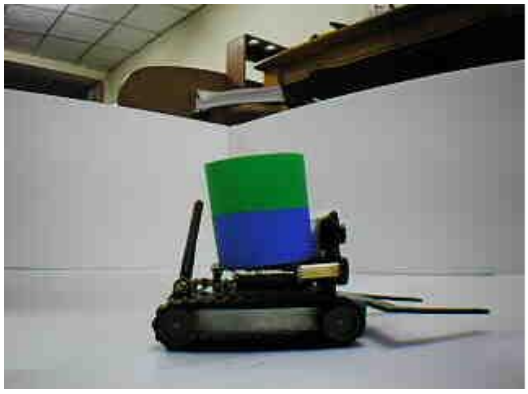
$$w = \max_{I[r,c] \in C} c - \min_{I[r,c] \in C} c$$

$$h = \max_{I[r,c] \in C} r - \min_{I[r,c] \in C} r$$

ข้อมูลของแต่ละองค์ประกอบ จะถูกเก็บอยู่ในแถวลำดับความสำคัญ (Priority Queue) ซึ่งจัดลำดับความสำคัญตามขนาดขององค์ประกอบ กล่าวคือ ข้อมูลที่จะถูกดึงออกมาจากแถวลำดับก่อน จะเป็นข้อมูลขององค์ประกอบที่มีขนาดใหญ่ที่สุด ที่ทำเช่นนี้เพราะองค์ประกอบที่ใหญ่ที่สุด มีแนวโน้มที่จะเป็นวัตถุที่

ต้องการมากที่สุด และสำหรับองค์ประกอบที่มีขนาดเล็กเกินไป ก็จะไม่ถูกเก็บเข้าแถวลำดับเลย เพราะมีแนวโน้มสูงมากที่จะเป็นสิ่งรบกวนจากภายนอก

นอกจากนี้ ระบบหุ่นยนต์หลายตัว ต้องมีการจัดการไม่ให้หุ่นยนต์เคลื่อนที่ชนกัน ผู้พัฒนาจึงเลือกสีเขียว และสีน้ำเงิน มาประกอบกันเป็นแถบสี และนำไปติดตั้งบนหุ่นยนต์ทุกตัว ดังรูปที่ 9 เพื่อเป็นสัญลักษณ์แทนตำแหน่งของหุ่นยนต์



รูปที่ 9 แถบสีเขียว-น้ำเงิน ที่ติดตั้งบนตัวหุ่นยนต์

ดังนั้น เมื่อหุ่นยนต์ตรวจพบวัตถุทั้งสีเขียวและสีน้ำเงิน จะต้องทำการตรวจสอบว่าเป็นวัตถุที่หุ่นยนต์จะต้องไปเก็บ หรือเป็นเพียงแค่แถบสีที่ใช้แทนตำแหน่งหุ่นยนต์เท่านั้น วิธีการตรวจสอบว่า ภาพวัตถุที่มองเห็นนั้นเป็นแถบสี ทำได้ดังนี้

ให้ G(x1, y1) และ B(x2, y2) เป็นจุดศูนย์กลางขององค์ประกอบสีเขียว และสีสีน้ำเงิน ตามลำดับ

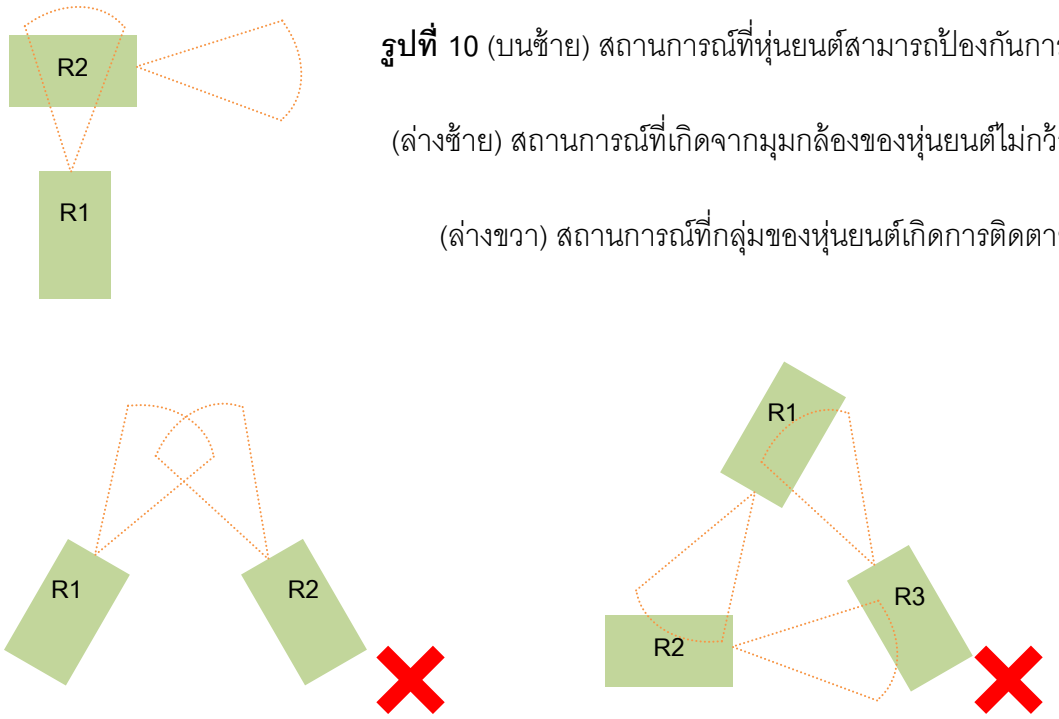
ถ้า $$\left|\tan^{-1}\frac{y2-y1}{x2-x1} - \frac{PI}{2}\right| < \varepsilon$$

หรือ $$\left|\tan^{-1}\frac{y2-y1}{x2-x1} + \frac{PI}{2}\right| < \varepsilon$$

หรือกล่าวอีกนัยหนึ่ง คือ พบองค์ประกอบสีน้ำเงิน อยู่บนองค์ประกอบสีเขียว หรือพบองค์ประกอบสีเขียว อยู่บนองค์ประกอบสีน้ำเงินนั่นเอง แสดงว่า องค์ประกอบสีเขียวและสีน้ำเงินที่เห็นนั้น เป็นแถบสีบนตัวหุ่นยนต์ นั่นคือ มีหุ่นยนต์ตัวอื่นกำลังขวางแนวการเดินของหุ่นยนต์ตัวนี้อยู่

2.5 การหลีกเลี่ยงการชนกันของหุ่นยนต์

ระบบหุ่นยนต์ที่ออกแบบนี้ สามารถป้องกันไม่ให้หุ่นยนต์เคลื่อนที่ชนกันเองได้ แต่ได้เพียงบางกรณีเท่านั้น ดังรูปที่ 10



รูปที่ 10 (บนซ้าย) สถานการณ์ที่หุ่นยนต์สามารถป้องกันการชนได้ (ล่างซ้าย) สถานการณ์ที่เกิดจากมุมกล้องของหุ่นยนต์ไม่กว้างพอ (ล่างขวา) สถานการณ์ที่กลุ่มของหุ่นยนต์เกิดการติดตาย

จากรูปบนซ้าย หุ่นยนต์ R1 เห็นแถบสีของหุ่นยนต์ R2 แต่หุ่นยนต์ R2 ไม่เห็นแถบสีของหุ่นยนต์ R1 ดังนั้น หุ่นยนต์ R1 จะหยุดการเคลื่อนที่ และให้หุ่นยนต์ R2 เคลื่อนที่ผ่านไปจนกว่าจะพ้นมุมกล้องของหุ่นยนต์ R1 หุ่นยนต์ R1 จึงจะกลับมาเคลื่อนที่ใหม่อีกครั้ง

รูปล่างซ้าย เกิดจากการที่มุมกล้องของหุ่นยนต์ไม่กว้างพอ ทำให้หุ่นยนต์ทั้ง 2 ตัวไม่เห็นแถบสี และทำให้หุ่นยนต์เคลื่อนที่ต่อไปเรื่อยๆ จนมาบรรจบกันและชนกันในที่สุด ส่วนรูปล่างขวา เกิดจากการที่หุ่นยนต์แต่ละตัว ต่างก็เห็นแถบสีของหุ่นยนต์ตัวอื่นซึ่งกันและกัน ทำให้หุ่นยนต์ทุกตัวในกลุ่มนี้หยุดการเคลื่อนที่ และจะหยุดรอไปเรื่อยๆ ไม่มีที่สิ้นสุด สถานการณ์เช่นนี้ เรียกว่า การติดตาย (Deadlock)

2.6 การควบคุมหุ่นยนต์ Surveyor SRV-1 (Controlling Surveyor SRV-1 Robot)

เนื่องจากหุ่นยนต์ Surveyor SRV-1 เป็นหุ่นยนต์ Open Source ที่ใช้ระบบปฏิบัติการ Linux ดังนั้น ผู้พัฒนาทุกคนจึงมีสิทธิ์พัฒนาโปรแกรมที่ใช้ควบคุมหุ่นยนต์ หรือที่เรียกว่า Console หรือพัฒนา Library เพื่อใช้ควบคุมหุ่นยนต์ได้เอง นอกจากนี้ยังสามารถแก้ไข Firmware ภายในตัวหุ่นยนต์ได้อีกด้วย

การสั่งงานหุ่นยนต์ Surveyor SRV-1 สามารถทำได้ 2 แบบ คือ

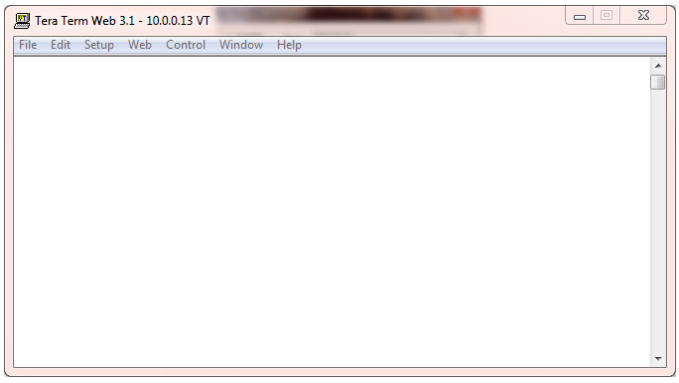
1. โดยการส่งคำสั่งไปในรูปแบบของ Byte Array ผ่านโปรโตคอล TCP หรือ UDP สำหรับรูปแบบของคำสั่ง สามารถดูได้ที่ http://www.surveyor.com/SRV_protocol.html [9]

2. โดยการเขียนโปรแกรมลงบนตัวหุ่นยนต์โดยตรง โดยใช้ภาษา PicoC [12]

การส่งโปรแกรมจากคอมพิวเตอร์ไปเก็บลงในหน่วยความจำแฟลชของหุ่นยนต์ สามารถทำได้ 2 วิธีหลัก คือ

1. ใช้โปรโตคอล XMODEM [10] ซึ่งเป็นโปรโตคอลอย่างง่ายที่ใช้ในการส่งไฟล์ และมีความสามารตรวจสอบข้อผิดพลาดที่เกิดจากการส่ง โปรแกรมที่รองรับการส่งไฟล์ผ่านโปรโตคอล XMODEM เช่น โปรแกรม TeraTerm [11] ดังแสดงในรูปที่ 11

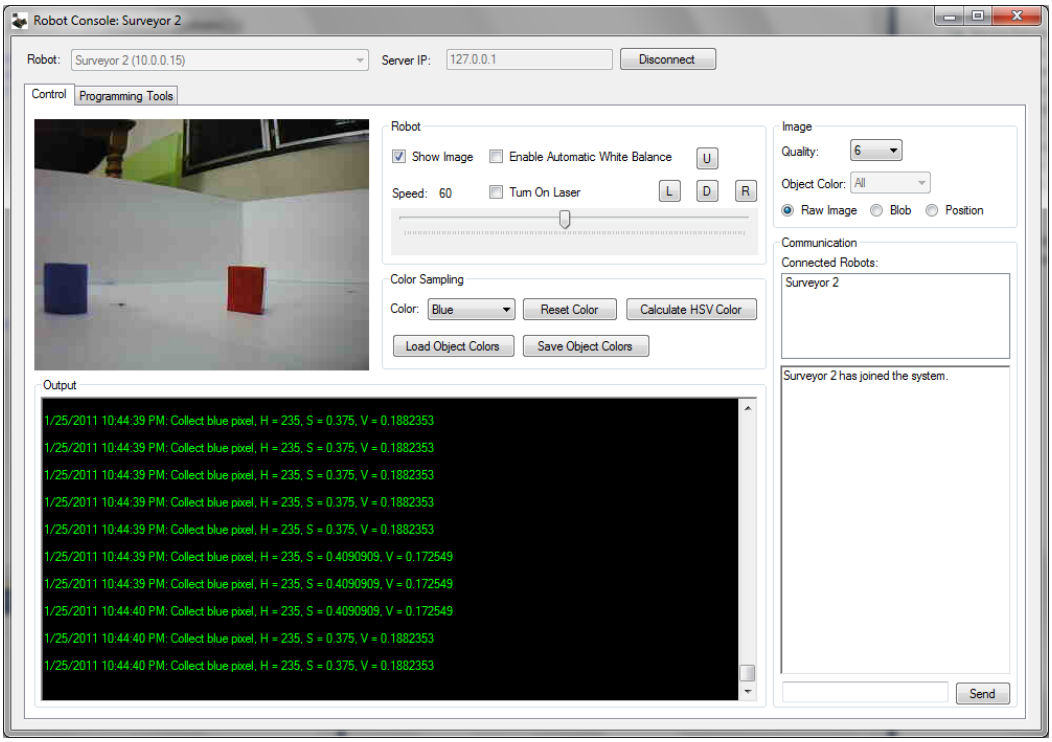
2. ใช้วิธีการสร้าง Console โดย Console จะเรียกใช้โปรโตคอลของหุ่นยนต์ Surveyor SRV-1 เพื่อทำการส่ง Byte Array ไปบอกหุ่นยนต์ที่จะโหลดโปรแกรมใส่ก่อน จากนั้น Console จึงค่อยส่งโปรแกรมไป

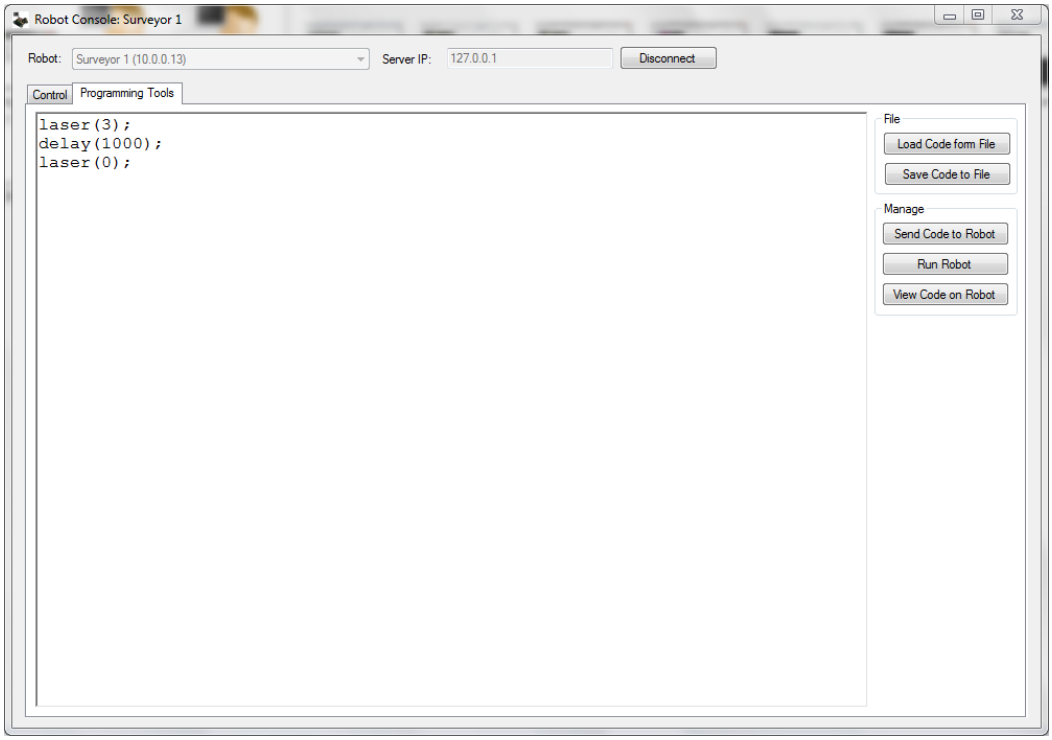


รูปที่ 11 โปรแกรม TeraTerm ที่รองรับโปรโตคอล XMODEM

เนื่องจากโครงงานของผู้พัฒนาเป็นงานเฉพาะทาง ที่จำเป็นต้องใช้ Console ที่มีรูปแบบการทำงานแตกต่างจาก Console ที่มีอยู่แล้ว กล่าวคือ Console ต้องสามารถประสานงานกับหุ่นยนต์ตัวอื่นๆ ได้ อีกทั้งยังมี Library เป็นจำนวนมาก ที่สามารถนำมาใช้สร้าง Console ได้อย่างสะดวก และนอกจากนี้ ผู้พัฒนาพบว่าการใช้โปรแกรม TeraTerm ในการส่งไฟล์ มีความยุ่งยากกว่าการสร้าง Console ขึ้นมาใช้เอง ทั้งนี้เนื่องจากการส่งโปรแกรมผ่านโปรแกรม TeraTerm จะต้องนำโปรแกรมเก็บลงไฟล์ก่อน แล้วจึงค่อยส่งไฟล์ไป และหากต้องการให้โปรแกรมบนหุ่นยนต์ทำงาน จะต้องเรียกใช้โปรแกรม TeraTerm อีกครั้งหนึ่ง แต่ถ้าเป็น Console ที่เขียนขึ้นเอง ผู้พัฒนาสามารถออกแบบ Console ให้มีที่ว่างสำหรับใช้แก้ไขโปรแกรม และมีปุ่มสำหรับโหลดโปรแกรมลงบนตัวหุ่นยนต์ รวมทั้งมีปุ่มสั่งให้โปรแกรมบนตัวหุ่นยนต์ทำงานอีกด้วย ผู้พัฒนาจึงตัดสินใจสร้าง Console ใช้เอง โดยอ้างอิงจากซอร์สโค้ดของ Console ที่มีอยู่แล้ว

ผู้พัฒนาจะใช้ Library ของ AForge.NET [7] เพื่อพัฒนา Console และ Console ที่พัฒนาขึ้น จะมีความสามารถในการสั่งงานหุ่นยนต์เบื้องต้น เช่น การควบคุม Motor การสั่งงาน Laser การรับภาพจากกล้องบนตัวหุ่นยนต์มาแสดงผลและประมวลผล เป็นต้น และมีความสามารถในการเขียนโปรแกรมลงในหน่วยความจำแฟลชของตัวหุ่นยนต์ หน้าตาของ Console ที่พัฒนาขึ้น แสดงดังรูปที่ 12





รูปที่ 12 (บน) Console ที่พัฒนาขึ้น เพื่อใช้สั่งงานพื้นฐานให้กับหุ่นยนต์

(ล่าง) Console ที่พัฒนาขึ้น เพื่อใช้สำหรับโหลดโปรแกรมลงบนหน่วยความจำแฟลช

อย่างไรก็ตาม ผู้พัฒนาไม่สามารถนำโปรแกรมการตรวจหาวัตถุ ตามหัวข้อ 2.4 ที่เขียนด้วยภาษา PicoC ไปใส่ไว้ในหน่วยความจำแฟลช และให้หน่วยประมวลผล (Processor) บนตัวหุ่นยนต์ทำการประมวลผลได้ ทั้งนี้เนื่องจากหน่วยความจำบนตัวหุ่นยนต์น้อยเกินไป เป็นผลทำให้การประมวลผลผิดพลาด และหากนำ Library สำหรับการตรวจหาวัตถุที่มีอยู่แล้วในภาษา PicoC มาใช้ ก็ไม่สามารถทำได้ เพราะเนื่องจากจำนวนองค์ประกอบในภาพมากเกินไป (อ้างอิงจากบริษัทผู้ผลิตหุ่นยนต์ Surveyor SRV-1) และต้องมีการคำนวณเพิ่มเติม จึงจะใช้ Library ได้ แต่ก็ยังทำให้การประมวลผลช้าลงไปมาก ปัญหาเหล่านี้ สามารถดูรายละเอียดเพิ่มเติมได้ในบทที่ 7 ปัญหาและอุปสรรค

ดังนั้น ผู้พัฒนาจึงตัดสินใจออกแบบให้ Console กลายเป็นหน่วยประมวลผลของหุ่นยนต์แทน โดย Console หนึ่ง Console สามารถเชื่อมต่อกับหุ่นยนต์ได้เพียงตัวเดียวเท่านั้น และจะทำหน้าที่ประมวลผลทุกอย่างเกี่ยวกับหุ่นยนต์ตัวที่เชื่อมต่อกับ Console นี้อยู่ รวมทั้งการประมวลผลภาพด้วย และ Console แต่ละ

Console สามารถติดต่อสื่อสารกับ Console อื่นๆ ได้ ทั้งนี้เพื่อให้เกิดการประสานงานกันระหว่างหุ่นยนต์ภายในกลุ่ม สำหรับตัวหุ่นยนต์เอง ก็เพียงแต่รับคำสั่งที่ได้รับการประมวลผลแล้วจาก Console เท่านั้น

กรณีที่ Console ต้องทำการติดต่อกับหุ่นยนต์ มีอยู่ 2 กรณี ดังนี้

1. **การสั่งงานทั่วไป**

การสั่งงานทั่วไป สามารถทำได้โดยการส่งข้อความในรูปแบบที่กำหนดให้ ผ่านโปรโตคอล TCP หรือ UDP เช่น หากต้องการสั่งให้ Motor เดินหน้าเป็นเวลา 0.7 วินาที จะต้องส่ง Byte Array เป็น 0x4D 0x32 0x46 0x14 (หรือเขียนอยู่ในรูปข้อความอักขระได้เป็น “M22F”) หรือหากต้องการสั่งเปิด Laser จะต้องส่ง Byte Array เป็น 0x6C (หรือเขียนอยู่ในรูปข้อความอักขระได้เป็น “l”) เป็นต้น

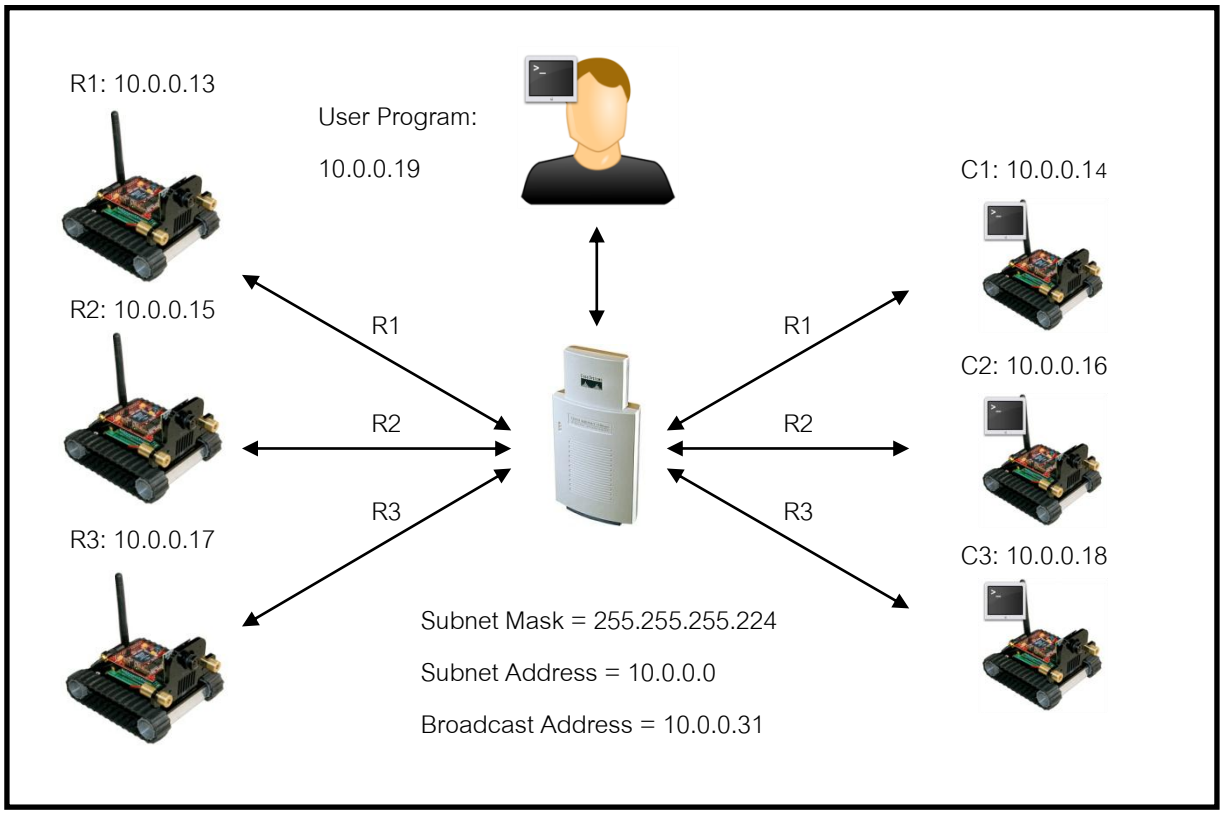
2. **การเขียนโปรแกรมลงบนหน่วยความจำแฟลซ**

การเขียนโปรแกรมลงบนหน่วยความจำแฟลซ สามารถทำได้โดยมีขั้นตอนดังนี้

1. ส่งข้อความ “zc” เพื่อทำการล้างข้อมูลเดิมในหน่วยความจำแฟลซออกให้หมด
2. ส่งข้อความ “E” เพื่อเข้าสู่โหมด Editor สำหรับแก้ไขข้อมูลในหน่วยความจำแฟลซ
3. สำหรับแต่ละบรรทัดของโปรแกรม ให้ส่งข้อความ “I” ไปบอกหุ่นยนต์ก่อน เพื่อบอกว่าจะทำการเพิ่มข้อมูลลงในหน่วยความจำแฟลซ
4. ส่งโปรแกรมบรรทัดนั้นเข้าไป
5. ส่งข้อความ 0x1B หรือเป็นข้อความแสดงการกด Esc Key เพื่อออกจากโหมดการเพิ่มข้อมูล
6. ทำขั้นตอนที่ 3-5 เรื่อยๆ จนส่งโปรแกรมครบทุกบรรทัด

2.7 การตั้งค่า IP และการแบ่ง Subnet (IP Assignment and Subnetting)

หุ่นยนต์ Surveyor SRV-1 รองรับการติดต่อสื่อสารผ่าน Wireless LAN 802.11g ดังนั้น เพื่อที่จะให้หุ่นยนต์ติดต่อสื่อสารกันได้ภายในกลุ่ม ผู้พัฒนาจำเป็นต้องออกแบบระบบเครือข่าย รูปที่ 13 แสดงออกแบบระบบเครือข่าย โดยการตั้งค่า IP ให้กับแต่ละส่วน



รูปที่ 13 การออกแบบระบบ เพื่อให้หุ่นยนต์แต่ละตัวสามารถประสานงานกันได้

จากการตั้ง Subnet Mask ทำให้สามารถกำหนด IP ในเครือข่ายนี้ได้แตกต่างกัน 32 แบบ แต่เครือข่ายนี้ จะต้องเสียไป 2 IP สำหรับ Subnet Address และ Broadcast Address ดังนั้น ในเครือข่ายนี้ จะมีหุ่นยนต์ได้มากที่สุด 28 ตัว โดยกรณีที่สามารถใช้หุ่นยนต์ได้ 28 ตัว คือ ให้ Console และโปรแกรมผู้ใช้ ทำงานในเครื่องคอมพิวเตอร์เครื่องเดียวกันให้หมด

Console หนึ่ง Console ทำหน้าที่ควบคุมหุ่นยนต์ 1 ตัว ดังที่ได้กล่าวไปแล้วในหัวข้อ 2.5 จากรูปที่ 13 C1, C2 และ C3 ทำหน้าที่ควบคุมหุ่นยนต์ R1, R2 และ R3 ตามลำดับ Console ต่างๆ รวมทั้งโปรแกรมที่ใช้เริ่มระบบการทำงานหุ่นยนต์ (User Program) สามารถทำงานบนเครื่องคอมพิวเตอร์เครื่องเดียวกัน หรือต่างเครื่องกันก็ได้ โดย Console และ User Program จะติดต่อสื่อสารกันด้วยโปรโตคอล TCP (Transmission Control Protocol)

บทที่ 3 รายละเอียดของการพัฒนา

3.1 เครื่องมือที่ใช้ในการพัฒนา

3.1.1 ฮาร์ดแวร์ (Hardware)

- หุ่นยนต์ Surveyor SRV-1 จำนวน 2 ตัวขึ้นไป
- Cisco Aironet 1100 Series Wireless Access Point สำหรับใช้เป็นศูนย์รวมการรับ-ส่งข้อมูลทั้งหมด

3.1.2 ซอฟต์แวร์ (Software)

- ระบบปฏิบัติการ Windows 7
- Microsoft Visual Studio 2010 สำหรับใช้เขียนภาษา C#
- AForge.NET Framework เวอร์ชัน 2.1.5 สำหรับใช้ติดต่อกับหุ่นยนต์ Surveyor SRV-1

3.1.3 อุปกรณ์เสริมอื่น ๆ

- สนามฟิวเจอร์บอร์ดสีขาว ที่มีความกว้าง และความยาว มากกว่า 1 เมตรขึ้นไป แต่ไม่เกิน 2 เมตร
- วัตถุทรงสี่เหลี่ยม ทรงกระบอก และปริซึมหน้าตัดรูปสามเหลี่ยม ที่มีขนาดที่บรรจุได้พอดีในทรงลูกบาศก์ขนาด 6 x 6 x 6 ลูกบาศก์เซนติเมตร
- วัตถุทรงกระบอก ที่จะใช้เป็นเป้าหมายให้หุ่นยนต์นำวัตถุมาวาง และมีขนาดเส้นผ่านศูนย์กลาง 10 เซนติเมตร สูง 20 เซนติเมตร

3.2 Functional Specification

Functional Specification ของระบบหุ่นยนต์หลายตัว สามารถแบ่งได้ออกเป็น 3 ส่วน ตามหน้าที่การทำงาน คือ Functional Specification ของ Console ของโปรแกรมผู้ใช้ และของกลุ่มหุ่นยนต์

3.2.1 Functional Specification ของ Console

1. Console สามารถสั่งงานพื้นฐานแก่หุ่นยนต์ได้ เช่น สั่งให้มอเตอร์หมุน สั่งเปิด Laser หรือสั่งให้รับภาพจากกล้องที่ติดตั้งบนตัวหุ่นยนต์ และ Console ต้องสามารถรองรับการเขียนโปรแกรมภาษา PicoC ลงหน่วยความจำแฟลชบนตัวหุ่นยนต์ได้
2. ผู้ใช้สามารถกำหนดค่าสีของวัตถุต่างๆ โดยการเก็บจุดภาพตัวอย่างของวัตถุ จากภาพใน Console รวมทั้งนำเข้า หรือจัดเก็บค่าสีของวัตถุได้
3. Console ต้องมีความสามารถในการประมวลผลภาพ (Image Processing) เพื่อใช้ในการตรวจหาวัตถุ
4. ผู้ใช้สามารถเขียนโปรแกรมสั่งงานหุ่นยนต์ได้ โดยการเขียนโปรแกรมภาษา C# ในไฟล์ RobotExecutionCode.cs ซึ่งเป็นไฟล์เก็บซอร์สโค้ดการทำงานของหุ่นยนต์ และเป็นส่วนหนึ่งของซอร์สโค้ดทั้งหมดของโปรแกรม Console ดังนั้น เมื่อมีการแก้ไขไฟล์นี้ จะต้องทำการคอมไพล์โปรแกรม Console ใหม่ทั้งหมด
5. Console ที่พัฒนาขึ้นมา รวมทั้งซอร์สโค้ดในไฟล์ RobotExecution.cs สามารถนำไปใช้กับหุ่นยนต์ตัวใดก็ได้ (Write once, run anywhere)

3.2.2 Functional Specification ของโปรแกรมผู้ใช้

1. ผู้ใช้สามารถเรียกดูสถานะการทำงานของระบบหุ่นยนต์ได้
2. ผู้ใช้สามารถเริ่มการทำงาน หรือหยุดการทำงานของระบบหุ่นยนต์หลายตัวได้ทุกเมื่อ

3.2.3 Functional Specification ของกลุ่มหุ่นยนต์

1. ในกรณีที่มีหุ่นยนต์ภายในกลุ่มเกิดข้อผิดพลาด ระบบจะต้องรายงานให้ผู้ใช้ทราบ และหุ่นยนต์ตัวที่สามารถทำงานได้ จะต้องมาทำงานแทนที่หุ่นยนต์ตัวที่ขัดข้อง
2. หุ่นยนต์ภายในสนาม สามารถหลบหลีกหุ่นยนต์ตัวอื่นได้ ตามรูปแบบที่ได้อธิบายไว้ในหัวข้อ 2.4
3. ระบบหุ่นยนต์หลายตัว จะต้องรองรับการเพิ่ม-ลดจำนวนหุ่นยนต์ได้ โดยไม่จำเป็นต้องปรับแก้ระบบแต่อย่างใด อย่างไรก็ตาม ระบบหุ่นยนต์หลายตัวสามารถรองรับจำนวนหุ่นยนต์ได้ไม่เกิน 5 ตัว

3.3 ขั้นตอนการทำงานของระบบหุ่นยนต์หลายตัว

ขั้นตอนการทำงานของระบบหุ่นยนต์ สามารถแบ่งได้เป็น 7 ขั้นตอน ดังนี้

1. ผู้ใช้สั่งเริ่มระบบหุ่นยนต์หลายตัว โดยผ่านทางโปรแกรมผู้ใช้
2. หุ่นยนต์แต่ละตัว หมุนรอบตัวเองเพื่อหาวัตถุที่ตัวมันเองจะต้องไปจับ
3. เมื่อหุ่นยนต์พบวัตถุที่ต้องการแล้ว หุ่นยนต์ก็จะมุ่งหน้าไปยังวัตถุชิ้นนั้น
4. เมื่อหุ่นยนต์จับวัตถุได้แล้ว หุ่นยนต์จะต้องหาตำแหน่งเป้าหมายที่จะนำวัตถุไปวางต่อ
5. เมื่อหุ่นยนต์พบตำแหน่งเป้าหมายแล้ว หุ่นยนต์จะต้องนำวัตถุไปวางไว้ ณ ตำแหน่งเป้าหมาย
6. เมื่อหุ่นยนต์ทำงานเสร็จแล้ว หุ่นยนต์ตัวนั้นจะรายงานให้หุ่นยนต์ตัวอื่นทราบ
7. เมื่อหุ่นยนต์ทุกตัวทำงานเสร็จสิ้นแล้ว ก็จะแจ้งให้ผู้ใช้ทราบผ่านทางโปรแกรมผู้ใช้

ขั้นตอนที่ 1 และ 2 สามารถแสดงได้โดยผังการทำงานตามรูปที่ 14

ขั้นตอนที่ 3 สามารถแสดงได้โดยผังการทำงานตามรูปที่ 15

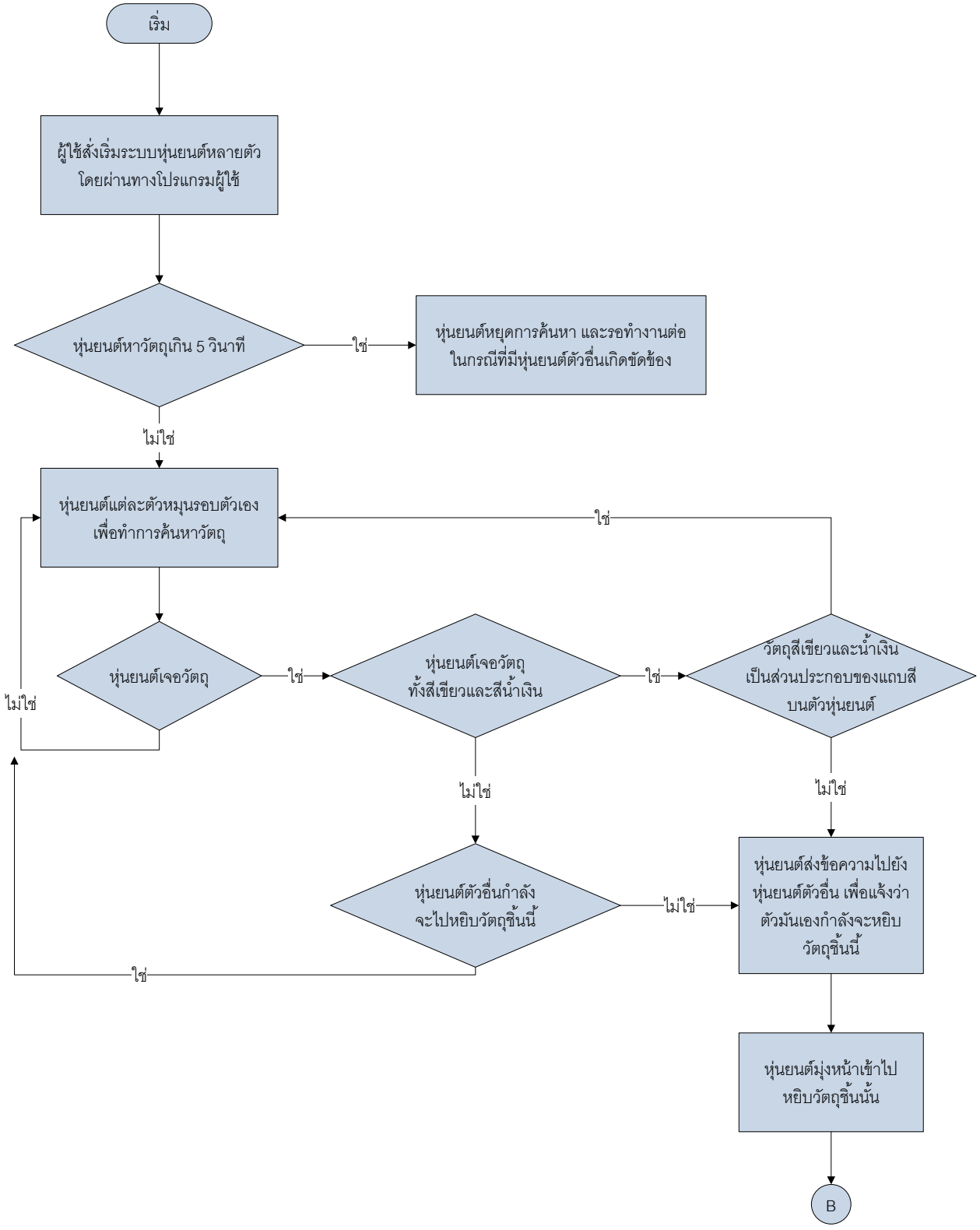
ขั้นตอนที่ 4 สามารถแสดงได้โดยผังการทำงานตามรูปที่ 16

ขั้นตอนที่ 5, 6 และ 7 สามารถแสดงได้โดยผังการทำงานตามรูปที่ 17

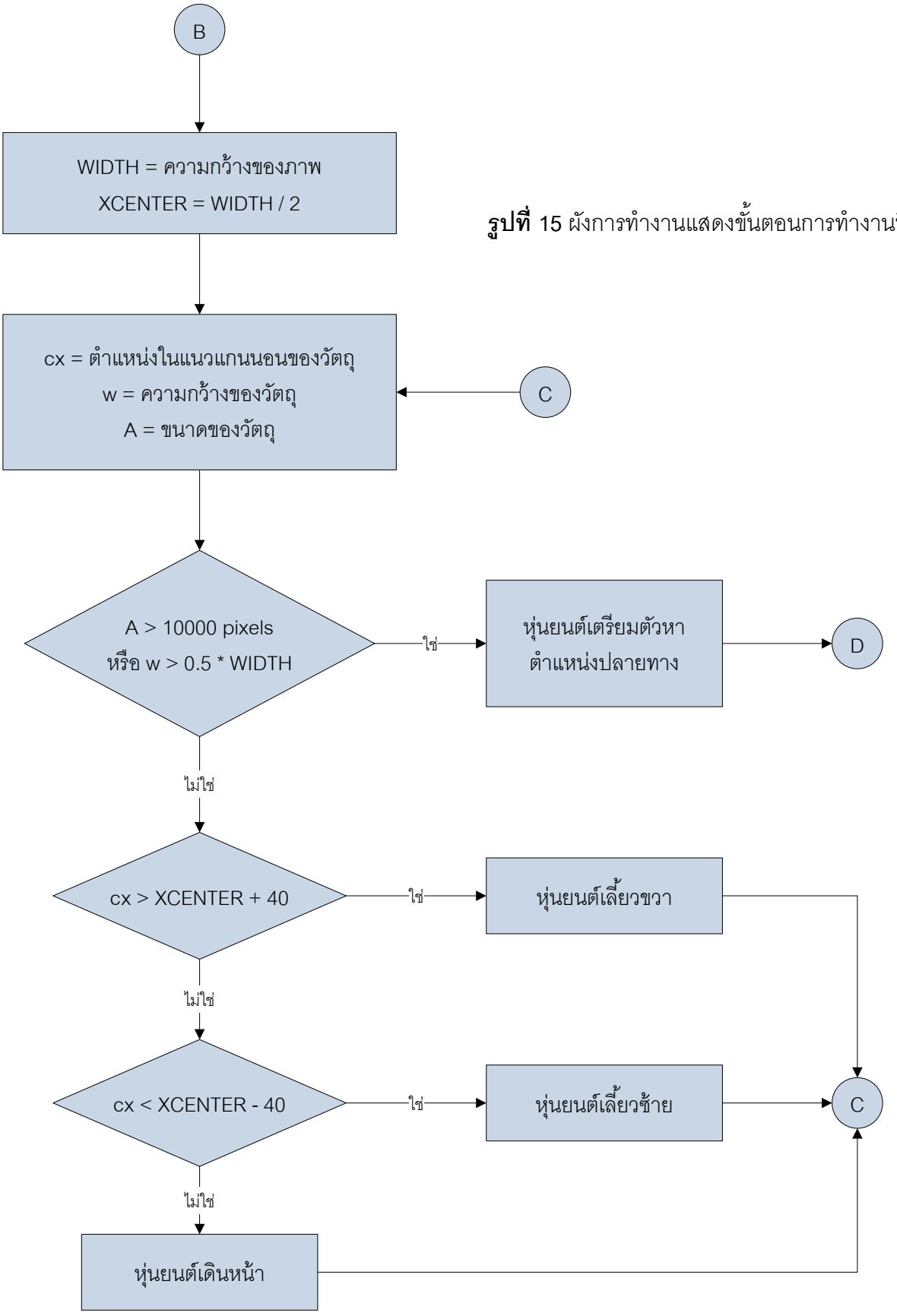
สำหรับกรณีเฉพาะ ที่ทำให้ขั้นตอนการทำงานของระบบหุ่นยนต์หลายตัว ไม่ตรงตามที่ได้เขียนไว้ มีอยู่ 2 กรณี คือ

1. หุ่นยนต์ไม่สามารถหาวัตถุที่จะไปจับได้ภายในระยะเวลาที่กำหนด ทั้งที่เกิดจากการที่หุ่นยนต์ตัวอื่นบังวัตถุ และการที่ไม่มีวัตถุเหลือให้หุ่นยนต์ตัวนี้เข้าไปจับเลย โดยหุ่นยนต์ตัวนี้จะหยุดการค้นหา และจะไม่ทำอะไรต่อ แต่จะรอทำงานในกรณีที่มีหุ่นยนต์ตัวอื่นเกิดข้อผิดพลาด
2. หุ่นยนต์ภายในกลุ่มตรวจพบว่า มีหุ่นยนต์ภายในกลุ่มเกิดขัดข้อง ทำให้หุ่นยนต์ตัวที่ทำงานเสร็จแล้ว หรือตัวที่ทำงานได้ ต้องเข้ามารับงานแทนที่หุ่นยนต์ที่ขัดข้อง

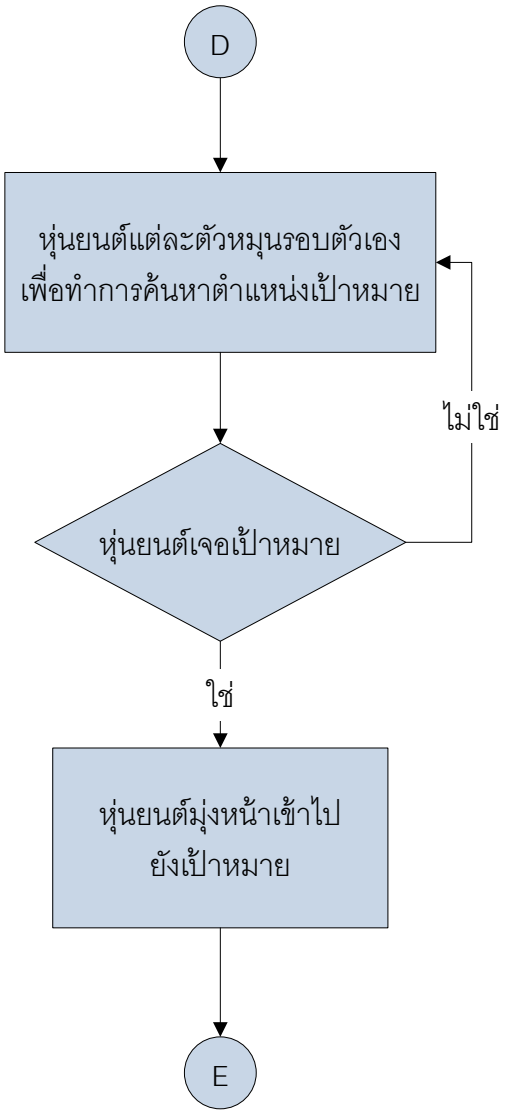
กรณีพิเศษทั้ง 2 กรณี สามารถแสดงได้โดยผังการทำงานตามรูปที่ 18



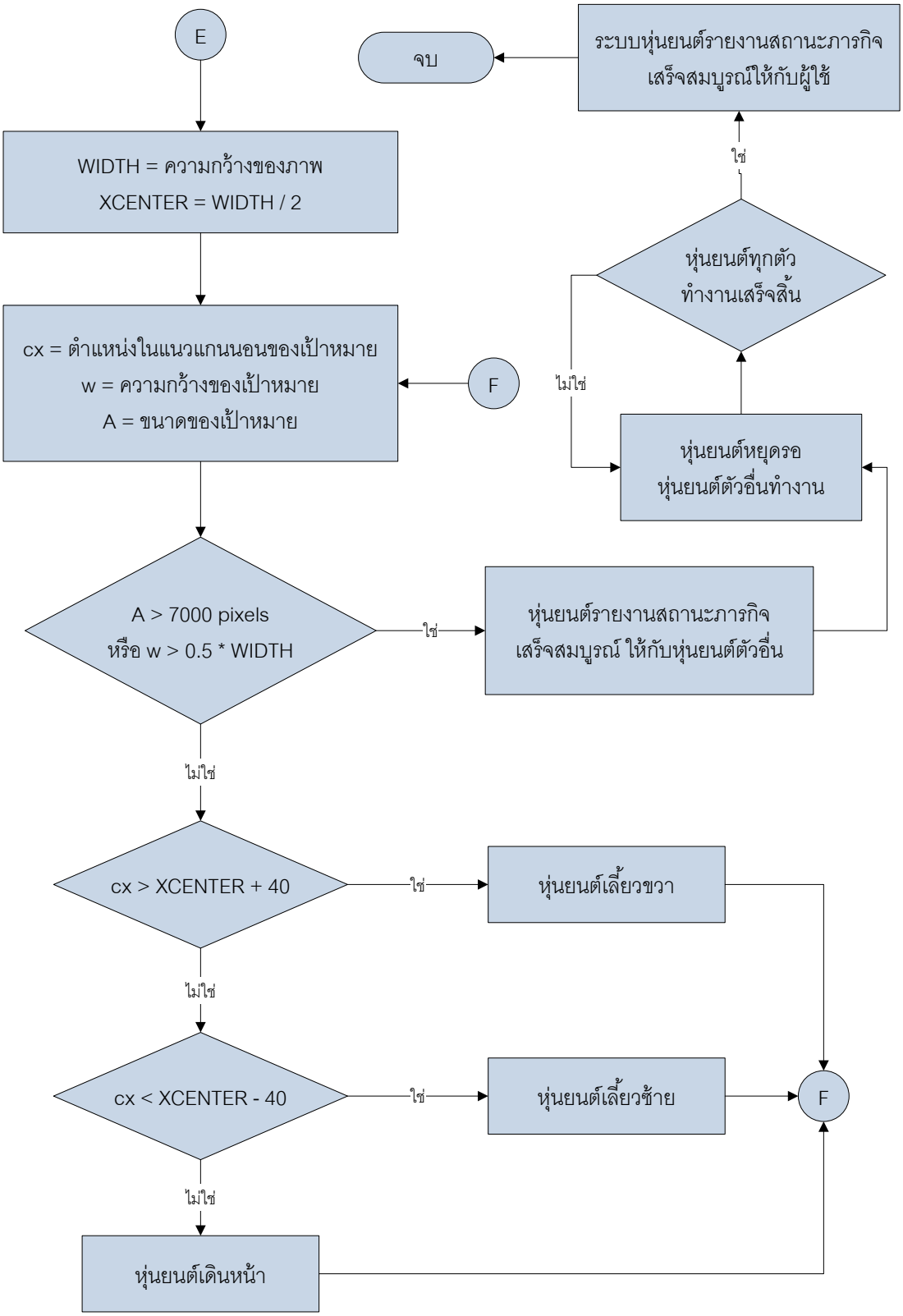
รูปที่ 14 ผังการทำงานแสดงขั้นตอนการทำงานที่ 1 และ 2



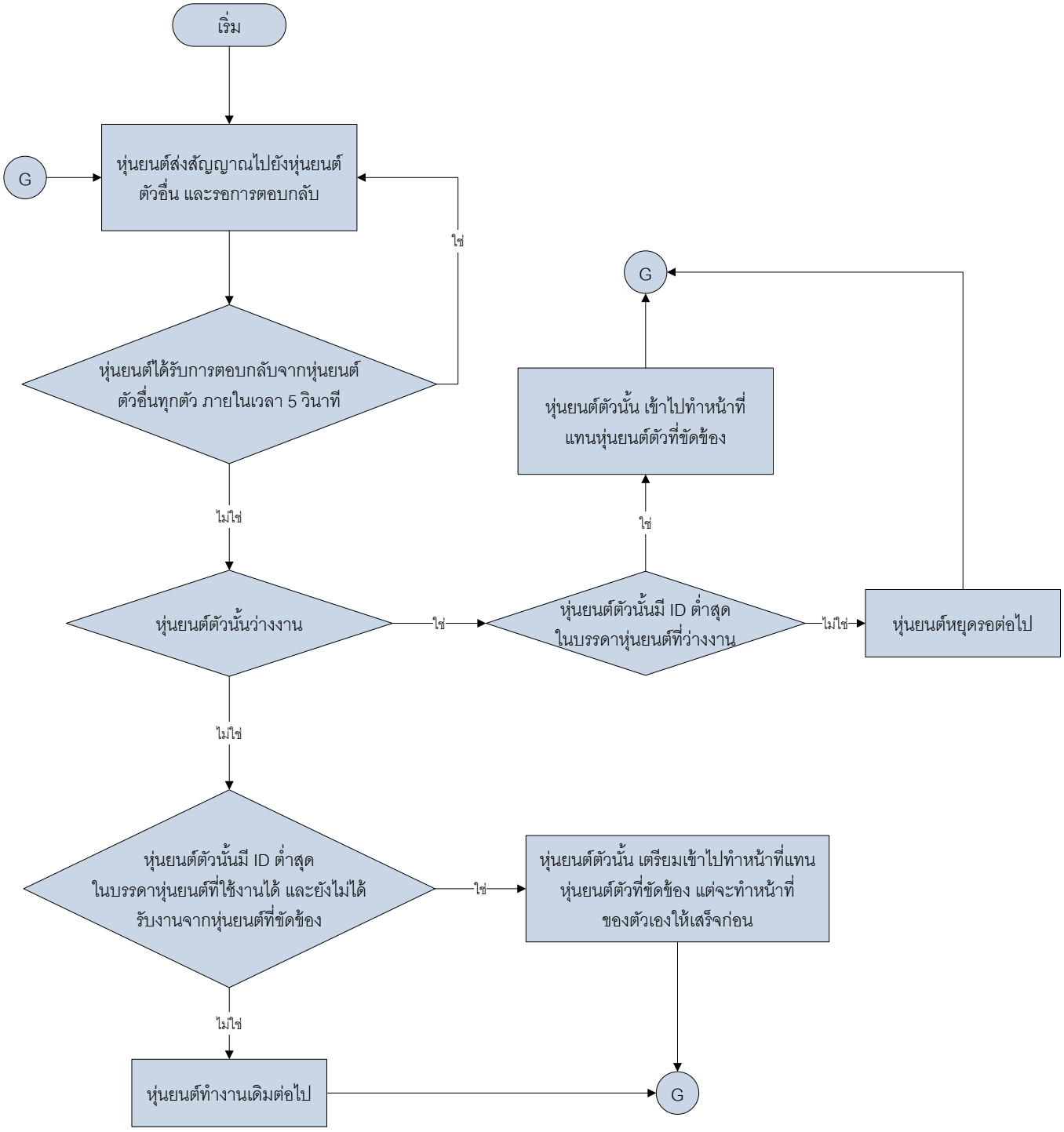
รูปที่ 15 ผังการทำงานแสดงขั้นตอนการทำงานที่ 3



รูปที่ 16 ผังการทำงานแสดงขั้นตอนการทำงานที่ 4



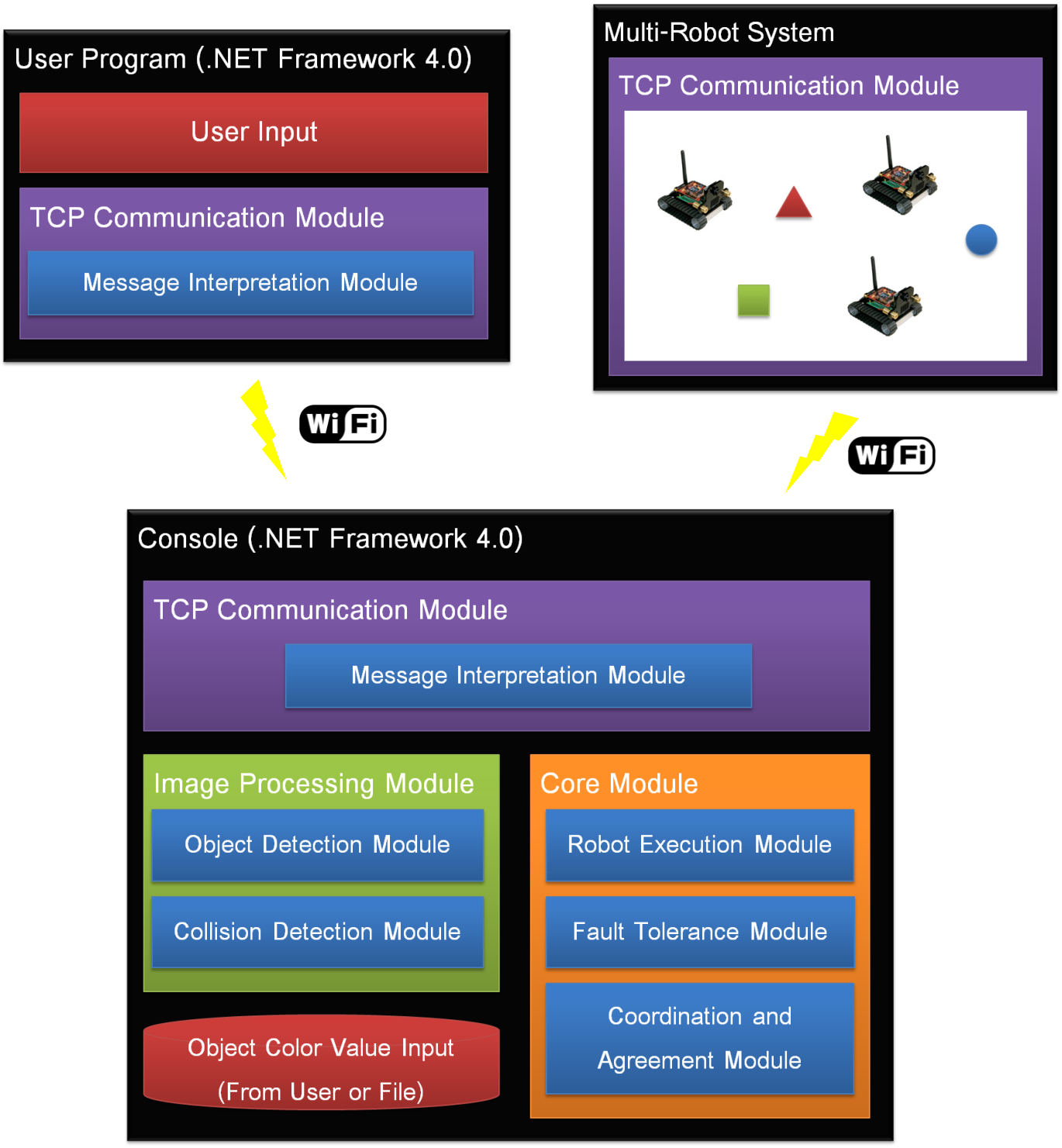
รูปที่ 17 ผังการทำงานแสดงขั้นตอนการทำงานที่ 5, 6 และ 7



รูปที่ 18 ผังการทำงานแสดงขั้นตอนการตรวจสอบความผิดพลาดของหุ่นยนต์ในระบบ

ที่จะทำให้เกิดกรณีพิเศษขึ้นมา

3.4 โครงสร้างของซอฟต์แวร์



รูปที่ 19 โครงสร้างของซอฟต์แวร์ของ Console, โปรแกรมผู้ใช้ และกลุ่มหุ่นยนต์

โครงสร้างของซอฟต์แวร์ (Software Architecture) ของระบบหุ่นยนต์หลายตัว ดังรูปที่ 19 ประกอบด้วยโครงสร้างซอฟต์แวร์ของ 3 ส่วนย่อย คือ

1. โครงสร้างซอฟต์แวร์ของ Console
2. โครงสร้างซอฟต์แวร์ของโปรแกรมผู้ใช้
3. โครงสร้างซอฟต์แวร์ของกลุ่มหุ่นยนต์

โดยที่แต่ละส่วน ประกอบด้วยโมดูล (Module) ต่างๆ ดังนี้

3.4.1 **โครงสร้างซอฟต์แวร์ของ** Console

1. TCP Communication Module

TCP Communication Module เป็นโมดูลที่ใช้ในการรับ-ส่งข้อความสั่งงาน หรือข้อความรายงานสถานะการทำงานต่างๆ กับระบบหุ่นยนต์ และโปรแกรมผู้ใช้ ภายในโมดูลนี้ จะมีโมดูลย่อยที่ชื่อว่า Message Interpretation Module สำหรับใช้ตีความข้อความ ที่ถูกส่งมาในรูปแบบต่างๆ ว่าเป็นคำสั่ง หรือสถานะการทำงานประเภทใดนั่นเอง

2. Image Processing Module

Image Processing Module เป็นโมดูลที่ทำหน้าที่รับภาพจากกล้องของหุ่นยนต์ มาทำการประมวลผล โดยเริ่มแรก ผู้ใช้จะต้องกำหนดค่าสีของวัตถุให้กับโปรแกรมก่อน โดยผ่านทางไฟล์ข้อความ (Text File) หรือผ่านทาง Console โมดูลนี้ ประกอบด้วย 2 โมดูลย่อยคือ

a. *Object Detection Module* เป็นโมดูลที่ทำหน้าที่หาตำแหน่ง ทิศทาง และขนาดของวัตถุและปลายทาง
b. *Collision Detection Module* เป็นโมดูลที่ทำหน้าที่คอยตรวจสอบว่า มีหุ่นยนต์ตัวอื่นขวางแนวการเดินหรือไม่ โดยใช้วิธีการหาตำแหน่ง ทิศทาง และขนาดของแถบสีที่ติดอยู่บนตัวหุ่นยนต์

3. Core Module

Core Module เป็นโมดูลที่ทำหน้าที่รับข้อมูลจากโมดูล TCP Communication และ Image Processing มาทำการประมวลผลต่อ โดยข้อมูลที่รับเข้ามา คือ คำสั่งที่ใช้สั่งงานหุ่นยนต์ สถานะการทำงานของหุ่นยนต์ภายในกลุ่ม และข้อมูลตำแหน่ง ทิศทาง และขนาดของทั้งวัตถุและปลายทาง โมดูลนี้ ประกอบด้วย 3 โมดูลย่อย คือ

a. *Robot Execution Module* เป็นโมดูลที่กำหนดขั้นตอนการทำงานของหุ่นยนต์ และเป็นโมดูลที่สามารถแก้ไขได้ เพื่อให้หุ่นยนต์มีการทำงานเป็นไปตามที่ผู้ใช้ต้องการ
b. *Fault Tolerance Module* เป็นโมดูลที่คอยจัดการเกี่ยวกับความผิดพลาดที่เกิดขึ้นกับหุ่นยนต์ภายในระบบ และทำหน้าที่กู้คืนระบบให้กลับสู่สภาพปกติเท่าที่จะเป็นไปได้
c. *Coordination and Agreement Module* ทำหน้าที่เก็บสถานะการทำงานของหุ่นยนต์ทุกตัวภายในกลุ่ม และถ้าหากเกิดการเปลี่ยนแปลงกับระบบหุ่นยนต์ โมดูลนี้ จะต้องปรับปรุงค่าสถานะให้เป็นสถานะล่าสุดของระบบด้วย

3.4.2 โครงสร้างซอฟต์แวร์ของโปรแกรมผู้ใช้

1. TCP Communication Module

TCP Communication Module เป็นโมดูลที่ใช้ในการรับ-ส่งข้อความสั่งงาน หรือข้อความรายงานสถานะการทำงานต่างๆ กับ Console ภายในโมดูลนี้ จะมีโมดูลย่อยที่ชื่อว่า Message Interpretation Module สำหรับใช้ตีความข้อความ ที่ถูกส่งมาในรูปแบบต่างๆ ว่าเป็นคำสั่ง หรือสถานะการทำงานประเภทใดนั่นเอง นอกจากนี้ โมดูลนี้ยังทำหน้าที่รับคำสั่งจากผู้ใช้ มาทำการวิเคราะห์ และสั่งงานให้ Console รวมทั้งกลุ่มหุ่นยนต์ ทำงานตามที่ผู้ใช้ต้องการ

3.4.3 โครงสร้างซอฟต์แวร์ของกลุ่มหุ่นยนต์

1. TCP Communication Module

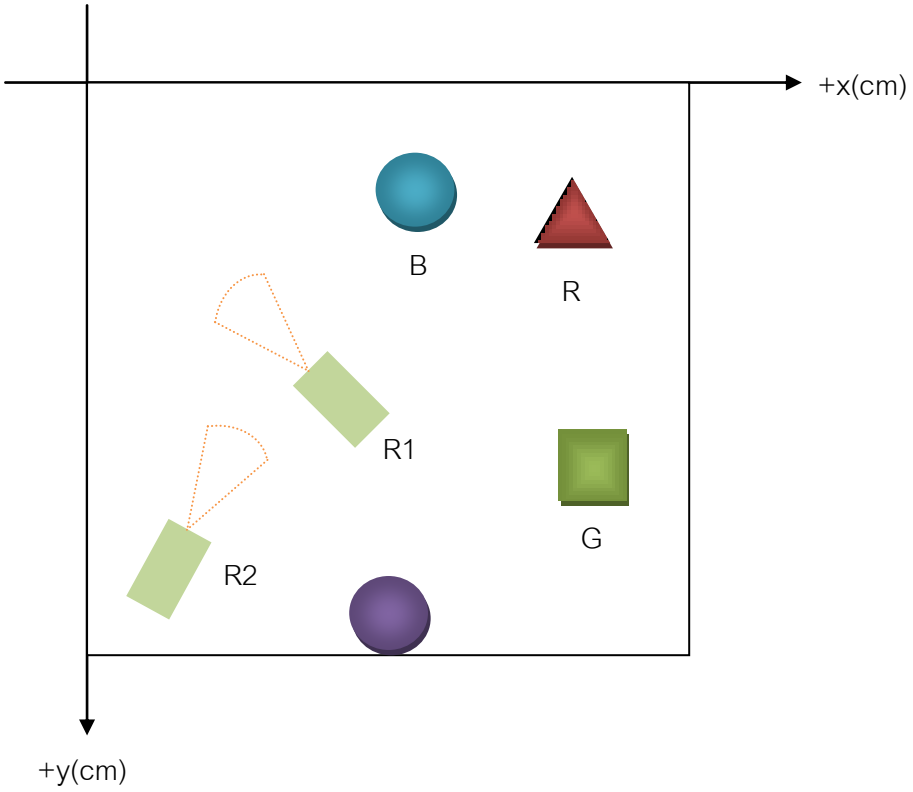
TCP Communication Module เป็นโมดูลที่ใช้ในการรับข้อความสั่งงานจาก Console และส่งข้อมูลภาพให้กับ Console

3.5 ขอบเขตละข้อจำกัดของระบบที่พัฒนา

1. ระบบหุ่นยนต์หลายตัวที่พัฒนาขึ้นมานี้ สามารถทำงานได้ภายในสนาม ที่มีพื้นและขอบสนามเป็นสีขาวเท่านั้น หากทำงานภายในสภาวะแวดล้อมอื่น ระบบหุ่นยนต์จะทำงานผิดพลาด เพราะมีแสงรบกวนจากภายนอกมาก
2. ระบบหุ่นยนต์หลายตัว สามารถป้องกันการชนได้บางกรณีเท่านั้น กล่าวคือ หุ่นยนต์สามารถป้องกันการชนได้ ก็ต่อเมื่อหุ่นยนต์ต้องเห็นแถบสีที่ติดตั้งบนหุ่นยนต์ตัวอื่น การเกิดมุมอับของกล้อง ทำให้หุ่นยนต์ไม่สามารถตรวจสอบตำแหน่งของหุ่นยนต์ตัวอื่นได้
3. ระบบหุ่นยนต์หลายตัว สามารถทนต่อความผิดพลาดแบบแครช (Crash) ได้เท่านั้น ไม่สามารถทนต่อความผิดพลาดแบบอาร์บิทรารี่ (Arbitrary) ได้ กล่าวคือ หุ่นยนต์ภายในกลุ่ม สามารถเข้าไปช่วยเหลือหุ่นยนต์ที่หยุดการตอบสนองได้ แต่จะไม่สามารถเข้าไปช่วยเหลือหุ่นยนต์ที่ทำงาน แต่ทำงานผิดพลาดได้
4. หุ่นยนต์สามารถทำงานรวมกันได้ เมื่อหุ่นยนต์อยู่ภายในเครือข่ายที่กำหนดไว้ให้เท่านั้น
5. ระบบหุ่นยนต์สามารถรองรับหุ่นยนต์ได้ไม่เกิน 5 ตัว

บทที่ 4 ผลการทดสอบระบบ

การทดสอบระบบ จะเริ่มจากการทดสอบความถูกต้องของหน่วยการทำงานย่อยๆ ก่อน แล้วจึงค่อยทดสอบระบบโดยรวมในภายหลัง เพื่อเปรียบเทียบความถูกต้อง ระยะเวลาที่ใช้ และประสิทธิภาพระหว่างหุ่นยนต์ตัวเดียว กับระบบหุ่นยนต์หลายตัว ดังนั้น การทดสอบระบบจะแบ่งออกเป็น 4 ชุดการทดสอบ แต่ก่อนที่จะกล่าวถึงรายละเอียดของการทดสอบ ผู้พัฒนาขอนิยามการระบุตำแหน่งและวัตถุในสนาม โดยใช้ Configuration ซึ่งการหา Configuration มีขั้นตอนดังนี้



รูปที่ 20 รูปแสดงการบอกพิกัดของวัตถุ และหุ่นยนต์ในสนาม

ให้ R, G, B เป็นตำแหน่งของวัตถุสีแดง, สีเขียว, สีน้ำเงิน, และ Ri เป็นตำแหน่งของหุ่นยนต์ตัวที่ i Configuration ของสนามนี้ คือ (R, G, B, R1, R2, …) และหากวัตถุใดหรือหุ่นยนต์ตัวใดไม่มีในสนาม ก็ใช้ค่า null แทน

4.1 การทดสอบความแม่นยำในการจับวัตถุ

Configuration	ครั้งที่	ผลการทดสอบ
((90, 10), null, null, (10, 90), null)	1	หุ่นยนต์หาวัตถุไม่พบ
(null, (90, 10), null, (10, 90), null)	2	หุ่นยนต์หาวัตถุไม่พบ
(null, null, (90, 10),(10, 90), null)	3	หุ่นยนต์หาวัตถุไม่พบ
((80, 30), null, null, (10, 90), null)	4	หุ่นยนต์หาวัตถุไม่พบ
(null, (80, 30), null, (10, 90), null)	5	หุ่นยนต์หยิบวัตถุสีเขียวได้
(null, null, (80, 30),(10, 90), null)	6	หุ่นยนต์หาวัตถุไม่พบ
((85, 50), null, null, (10, 90), null)	7	หุ่นยนต์หยิบวัตถุสีแดงได้
(null, (85, 50), null, (10, 90), null)	8	หุ่นยนต์หยิบวัตถุสีเขียวได้
(null, null, (85, 50),(10, 90), null)	9	หุ่นยนต์หาวัตถุไม่พบ
((50, 50), null, null, (10, 90), null)	10	หุ่นยนต์หยิบวัตถุสีแดงได้
(null, (50, 50), null, (10, 90), null)	11	หุ่นยนต์หยิบวัตถุสีเขียวไม่ได้
(null, null, (50, 50),(10, 90), null)	12	หุ่นยนต์หยิบวัตถุสีน้ำเงินได้
((40, 75), null, null, (10, 90), null)	13	หุ่นยนต์หยิบวัตถุสีแดงได้
(null, (40, 75), null, (10, 90), null)	14	หุ่นยนต์หยิบวัตถุสีเขียวได้
(null, null, (40, 75),(10, 90), null)	15	หุ่นยนต์หยิบวัตถุสีน้ำเงินได้

ตารางที่ 1 ผลการทดสอบความแม่นยำในการจับวัตถุ

4.2 การทดสอบความสามารถในการป้องกันการชน

Configuration	ครั้งที่	ผลการทดสอบ
(null, (30, 0), (100, 70)), (20, 70), (30, 80))	1	หุ่นยนต์ R1 และ R2 ชนกันตั้งแต่เริ่มการทำงาน
	2	หุ่นยนต์ R1 และ R2 ชนกันตั้งแต่เริ่มการทำงาน
(null , (30, 0), (100, 50), (20, 50), (30, 80))	3	หุ่นยนต์ R2 หยุดรอหุ่นยนต์ R1 ให้เคลื่อนที่ผ่านไปก่อน
	4	หุ่นยนต์ R2 หยุดรอหุ่นยนต์ R1 ให้เคลื่อนที่ผ่านไปก่อน
null , (30, 0), (100, 30), (20, 30), (30, 80))	5	หุ่นยนต์ R2 ไม่เห็นหุ่นยนต์ R1 แต่ไม่เกิดการชนกัน
	6	หุ่นยนต์ R2 ไม่เห็นหุ่นยนต์ R1 แต่ไม่เกิดการชนกัน
(null , (50, 0), (100, 70), (20, 70), (50, 80))	7	หุ่นยนต์ R2 หยุดรอหุ่นยนต์ R1 ให้เคลื่อนที่ผ่านไปก่อน
	8	หุ่นยนต์ R2 หยุดรอหุ่นยนต์ R1 ให้เคลื่อนที่ผ่านไปก่อน
(null , (50, 0), (100, 50), (20, 50), (50, 80))	9	หุ่นยนต์ R1 และ R2 ชนกันตรงกลางสนาม
	10	หุ่นยนต์ R1 และ R2 ชนกันตรงกลางสนาม
(null , (50, 0), (100, 30), (20, 30), (50, 80))	11	หุ่นยนต์ R2 หยุดรอหุ่นยนต์ R1 ให้เคลื่อนที่ผ่านไปก่อน
	12	หุ่นยนต์ R2 หยุดรอหุ่นยนต์ R1 ให้เคลื่อนที่ผ่านไปก่อน
(null , (70, 0), (100, 70), (20,70), (70, 80))	13	หุ่นยนต์ R2 วิ่งเข้าหาหุ่นยนต์ R1 เพราะคิดว่าเป็นวัตถุสีน้ำเงิน
	14	หุ่นยนต์ R2 วิ่งเข้าหาหุ่นยนต์ R1 เพราะคิดว่าเป็นวัตถุสีน้ำเงิน
(null , (70, 0), (100, 50), (20, 50), (70, 80))	15	หุ่นยนต์ R2 หยุดรอหุ่นยนต์ R1 ให้เคลื่อนที่ผ่านไปก่อน
	16	หุ่นยนต์ R2 หยุดรอหุ่นยนต์ R1 ให้เคลื่อนที่ผ่านไปก่อน
(null , (70, 0), (100, 30), (20, 30), (70, 80))	17	หุ่นยนต์ R1 และ R2 ชนกันตรงบริเวณวัตถุ
	18	หุ่นยนต์ R1 และ R2 ชนกันตรงบริเวณวัตถุ

ตารางที่ 2 ผลการทดสอบความสามารถในการป้องกันการชน

4.3 การทดสอบความทนทานต่อความผิดพลาด โดยทดลองปิดการทำงานหุ่นยนต์ 1 ตัว

การทดสอบนี้ จะทดลองปิดสวิตซ์ที่หุ่นยนต์ตัวที่ 2 ณ สถานะการทำงานต่างๆ ของระบบหุ่นยนต์ จากนั้นจึงสังเกตการตอบสนองของหุ่นยนต์ตัวที่ 1

สถานะการทำงานของหุ่นยนต์	ครั้งที่	ผลการทดสอบ
หุ่นยนต์ตัวที่ 2 ยังหาวัตถุที่จะเข้าไปจับไม่เจอ	1	หุ่นยนต์ตัวที่ 1 ทำงานต่อจนเสร็จ
หุ่นยนต์ตัวที่ 2 หาวัตถุที่จะเข้าไปจับได้แล้ว และหุ่นยนต์ตัวที่ 1 กำลังทำงานอยู่	2	เมื่อหุ่นยนต์ตัวที่ 1 ทำงานเสร็จแล้ว จึงจะไปช่วยหุ่นยนต์ตัวที่ 2 ทำงาน
หุ่นยนต์ตัวที่ 2 หาวัตถุที่จะเข้าไปจับได้แล้ว แต่หุ่นยนต์ตัวที่ 1 ว่างงาน	3	หุ่นยนต์ตัวที่ 1 เริ่มการทำงานอีกครั้ง โดยจะไปช่วยหุ่นยนต์ตัวที่ 2 ทำงาน
หุ่นยนต์ตัวที่ 2 ว่างงาน	4	หุ่นยนต์ตัวที่ 1 ทำงานต่อจนเสร็จ

ตารางที่ 3 ผลการทดสอบความทนทานต่อความผิดพลาด

4.4 การทดสอบความถูกต้องโดยรวม และเวลาที่ใช้

Configuration	ครั้งที่	ผลการทดสอบ
((40, 60), (50, 30), (70, 60), (30, 80), null)	1	หุ่นยนต์ R1 ทำงานเสร็จในเวลา 37.3 วินาที
	2	หุ่นยนต์ R1 ทำงานเสร็จในเวลา 38.7 วินาที
((40, 60), (50, 30), (70, 60), (30, 80), (80, 80))	3	หุ่นยนต์ R1 และ R2 ชนกันเพราะเกิดจากมุมอับกล้อง
	4	หุ่นยนต์ R2 หยิบวัตถุสีน้ำเงินไม่ได้
((40, 60), (50, 30), (70, 60), (30, 80), (80, 80), (20, 20))	5	หุ่นยนต์ R2 และ R3 ชนปะทะกัน เพราะแผงตั้งกล้องบนตัวหุ่นบดบังแถบสี
	6	หุ่นยนต์ R2 จับวัตถุสีน้ำเงินไม่ได้
((40, 60), (50, 30), (70, 60), (20, 80), null)	7	หุ่นยนต์ R1 ทำงานเสร็จในเวลา 35.2 วินาที
	8	หุ่นยนต์ R1 ทำงานเสร็จในเวลา 38.1 วินาที
((40, 60), (50, 30), (70, 60), (20, 80), (80, 80))	9	หุ่นยนต์ R2 ไม่ยอมปล่อยวัตถุสีน้ำเงิน
	10	หุ่นยนต์ทั้งสองตัวช่วยกันทำงานเสร็จในเวลา 34.0 วินาที
((40, 60), (50, 30), (70, 60), (20, 80), (80, 80), (80, 20))	11	หุ่นยนต์ R2 และ R3 ชนปะทะกัน แต่ก็เคลื่อนที่ต่อไปได้ ทำให้หุ่นยนต์ทั้งสามตัว ทำงานเสร็จในเวลา 24.3 วินาที
	12	หุ่นยนต์ R2 ถูก R1 ขวางอยู่ แต่ R2 ไม่สามารถตรจวจสอบได้เพราะมุมอับของกล้อง
((40, 60), (50, 30), (70, 60), (60, 50), null)	13	หุ่นยนต์ R1 ทำงานเสร็จในเวลา 37.0 วินาที
	14	หุ่นยนต์ R1 ทำงานเสร็จในเวลา 34.5 วินาที
((40, 60), (50, 30), (70, 60), (60, 50), (70, 50))	15	หุ่นยนต์ R2 ขวางทาง R1 ในณะที่หุ่นยนต์ R1 จะไปหยิบวัตถุสีแดง ทำให้เกิดการชนกัน
	16	หุ่นยนต์ R2 ขวางทาง R1 ในณะที่หุ่นยนต์ R1 จะไปหยิบวัตถุสีแดง ทำให้เกิดการชนกัน
((40, 60), (50, 30), (70, 60), (60, 50), (70, 50), (40, 50))	17	หุ่นยนต์ R3 ไม่สามารถเคลื่อนที่ต่อไปได้ เพราะติดแขนของหุ่นยนต์ R1
	18	หุ่นยนต์ทั้งสามตัวช่วยกันทำงานเสร็จในเวลา 18.5 วินาที

Configuration	ครั้งที่	ผลการทดสอบ
((80, 50), (50, 20), (50, 50), (30, 80), null)	19	หุ่นยนต์ R1 ทำงานเสร็จในเวลา 34.2 วินาที
	20	หุ่นยนต์ R1 ทำงานเสร็จในเวลา 42.2 วินาที
((80, 50), (50, 20), (50, 50), (30, 80), (80, 80))	21	หุ่นยนต์ R1 จับวัตถุสีเขียวไม่ได้
	22	หุ่นยนต์ R1 จับวัตถุสีน้ำเงินไม่ได้ และหุ่นยนต์เกิดการชนกัน
((80, 50), (50, 20), (50, 50), (30, 80), (80, 80), (20, 20))	23	หุ่นยนต์ R1 ถอยหลังชน R3
	24	หุ่นยนต์ R2 ติดขอบสนาม
((80, 50), (50, 20), (50, 50), (20, 80), null)	25	หุ่นยนต์ R1 จับวัตถุสีเขียวไม่ได้
	26	หุ่นยนต์ R1 ทำงานเสร็จในเวลา 35.4 วินาที
((80, 50), (50, 20), (50, 50), (20, 80), (80, 80))	27	หุ่นยนต์ R1 จับวัตถุสีเขียวไม่ได้
	28	หุ่นยนต์ทั้งสองตัวช่วยกันทำงานเสร็จในเวลา 33.9 วินาที
((80, 50), (50, 20), (50, 50), (20, 80), (80, 80), (80, 20))	29	หุ่นยนต์ R2 และ R3 ชนปะทะกัน เพราะแผงตั้งกล้องบนตัวหุ่นบดบังแถบสี
	30	หุ่นยนต์ R2 และ R3 ชนปะทะกัน เพราะแผงตั้งกล้องบนตัวหุ่นบดบังแถบสี
((80, 50), (50, 20), (50, 50), (60, 50), null)	31	หุ่นยนต์ R1 จับวัตถุสีเขียวไม่ได้
	32	หุ่นยนต์ R1 ทำงานเสร็จในเวลา 30.6 วินาที
((80, 50), (50, 20), (50, 50), (60, 50), (70, 50))	33	หุ่นยนต์ R1 และ R2 ชนกันตั้งแต่ตอนเริ่มงาน
	34	หุ่นยนต์ R1 และ R2 ชนกันตั้งแต่ตอนเริ่มงาน
((80, 50), (50, 20), (50, 50), (60, 50), (70, 50), (40, 50))	35	หุ่นยนต์ทั้งสามตัวหยิบวัตถุได้ แต่วิ่งชนกันทุกตัว
	36	หุ่นยนต์ทั้งสามตัวหยิบวัตถุได้ แต่วิ่งชนกันทุกตัว

Configuration	ครั้งที่	ผลการทดสอบ
((30, 70), (70, 30), (50, 50), (30, 80), null)	37	หุ่นยนต์ R1 ทำงานเสร็จในเวลา 36.6 วินาที
	38	หุ่นยนต์ R1 ทำงานเสร็จในเวลา 42.1 วินาที
((30, 70), (70, 30), (50, 50), (30, 80), (80, 80))	39	หุ่นยนต์ R1 และ R2 ชนกันเพราะเกิดจากมุมอับกล้อง
	40	หุ่นยนต์ R1 ถูกขวางโดยหุ่นยนต์ R2 หุ่นยนต์ R1 จึงต้องรอเป็นเวลานาน ทำให้หุ่นยนต์ทั้งสองตัวช่วยกันทำงานเสร็จในเวลา 122.3 วินาที
((30, 70), (70, 30), (50, 50), (30, 80), (80, 80), (20, 20))	41	หุ่นยนต์ทั้งสามตัวช่วยกันทำงานเสร็จในเวลา 25.2 วินาที
	42	หุ่นยนต์ R1 ไม่สามารถเคลื่อนที่ต่อไปได้ เพราะติดแขนของหุ่นยนต์ R3
((30, 70), (70, 30), (50, 50), (20, 80), null)	43	หุ่นยนต์ R1 ทำงานเสร็จในเวลา 40.9 วินาที
	44	หุ่นยนต์ R1 ทำงานเสร็จในเวลา 45.6 วินาที
((30, 70), (70, 30), (50, 50), (20, 80), (80, 80))	45	หุ่นยนต์ทั้งสองตัวช่วยกันทำงานเสร็จในเวลา 27.8 วินาที
	46	หุ่นยนต์ทั้งสองตัวช่วยกันทำงานเสร็จในเวลา 26.9 วินาที
((30, 70), (70, 30), (50, 50), (20, 80), (80, 80), (80, 20))	47	หุ่นยนต์ R1 และ R2 ชนกันเพราะเกิดจากมุมอับกล้อง
	48	หุ่นยนต์ R2 ไม่สามารถเคลื่อนที่ต่อไปได้ เพราะติดแขนของหุ่นยนต์ R3
((30, 70), (70, 30), (50, 50), (60, 50), null, null)	49	หุ่นยนต์ R1 จับวัตถุสีเขียวไม่ได้
	50	หุ่นยนต์ R1 ทำงานเสร็จในเวลา 40.6 วินาที
((30, 70), (70, 30), (50, 50), (60, 50), (70, 50), null)	51	หุ่นยนต์ R1 และ R2 ชนกันเพราะเกิดจากมุมอับกล้อง
	52	หุ่นยนต์ R1 และ R2 ชนกันเพราะเกิดจากมุมอับกล้อง
((30, 70), (70, 30), (50, 50), (60, 50), (70, 50), (40, 50))	53	หุ่นยนต์ R1 และ R3 ชนปะทะกัน เพราะแผงตั้งกล้องบนตัวหุ่นบดบังแถบสี
	54	หุ่นยนต์ R1 ไม่สามารถเคลื่อนที่ต่อไปได้ เพราะติดแขนของหุ่นยนต์ R3

ตารางที่ 4 ผลการทดสอบความถูกต้องโดยรวม และเวลาที่ใช้

จำนวนหุ่นยนต์	ผลการทดสอบ จำนวน 18 ครั้ง	เวลาที่ใช้โดยเฉลี่ย (กรณีที่ทำงานถูกต้อง)	ส่วนเบี่ยงเบนมาตรฐาน
1	ทำงานสำเร็จ 15 ครั้ง	37.9	3.71
2	ทำงานสำเร็จ 5 ครั้ง	30.6*	3.32*
3	ทำงานสำเร็จ 3 ครั้ง	22.7	2.97

*คิดจากค่าการทดสอบ 4 ครั้ง และตัดผลการทดสอบ 1 ครั้งที่มีค่าสูงผิดปกติออกไป

ตารางที่ 5 ตารางการเปรียบเทียบความถูกต้อง
และเวลาในการทำงาน ของจำนวนหุ่นยนต์ในกรณีต่างๆ

4.5วิเคราะห์ผลการทดสอบ

จากการทดสอบระบบหุ่นยนต์หลายตัวทั้ง 4 หัวข้อนั้น ผู้พัฒนาพบว่า ประสิทธิภาพ และระยะเวลาการทำงานของระบบหุ่นยนต์หลายตัว ไม่เป็นไปตามที่ผู้พัฒนาได้ตั้งสมมติฐานไว้ ผู้พัฒนาจึงนำผลการทดสอบมาวิเคราะห์ และพบว่าปัญหาเหล่านี้ ทำให้ระบบหุ่นยนต์ไม่สามารถทำงานได้ถูกต้อง

1. เมื่อวัตถุอยู่ไกลจากหุ่นยนต์มากเกินไป หุ่นยนต์จะมองไม่เห็นวัตถุ ทำให้หุ่นยนต์หาวัตถุไม่พบ (อ้างอิงจากการทดสอบที่ 4.1)
2. ขณะที่หุ่นยนต์ต้องการจะเข้าไปหยิบวัตถุ อาจเกิดความล่าช้าของระบบเครือข่าย หรือแสงของวัตถุเปลี่ยนไปก็ได้ ทำให้หุ่นยนต์ประมวลผลภาพผิดพลาด หรือประมวลผลไม่ทัน ทำให้หุ่นยนต์ไม่สามารถเข้าไปจับวัตถุได้อย่างแม่นยำ (อ้างอิงจากการทดสอบที่ 4.1)
3. หุ่นยนต์สามารถป้องกันการชนได้เพียงบางกรณีเท่านั้น ตามที่ได้อธิบายในหัวข้อ 2.3.2 นอกจากนี้ หุ่นยนต์อาจเกิดข้อผิดพลาด โดยการมองแถบสีเป็นวัตถุก็ได้ (อ้างอิงจากการทดสอบที่ 4.2)

4. ระบบหุ่นยนต์สามารถทนต่อความผิดพลาดแบบแครชเท่านั้น ดังนั้น หากเกิดความผิดพลาดแบบอื่น ระบบหุ่นยนต์จะทำงานแบบผิดพลาดต่อไป (อ้างอิงจากการทดสอบที่ 4.3)
5. การวางของวัตถุ มีผลต่อความถูกต้องของระบบ และจากการทดสอบ ผู้พัฒนาพบว่า การวางวัตถุทั้ง 3 ชิ้น ตามแนวทแยง ทำให้ระบบหุ่นยนต์หลายตัวมีประสิทธิภาพโดยรวมมากที่สุด (อ้างอิงจากการทดสอบที่ 4.4)

บทที่ 5 ปัญหาและอุปสรรค

ในระหว่างการพัฒนาระบบ ผู้พัฒนาได้พบปัญหามากมาย ซึ่งทำให้การพัฒนาระบบไม่เป็นไปตามที่คาดหวังไว้ และผู้พัฒนาต้องเสียเวลาส่วนใหญ่ไปกับการแก้ปัญหา ปัญหาบางปัญหา ผู้พัฒนาสามารถแก้ไขได้ ในขณะที่บางปัญหา ก็ไม่สามารถแก้ไขได้ เนื่องจากข้อจำกัดทางฮาร์ดแวร์ วิธีการที่ใช้ รวมทั้งความรู้ ความเข้าใจระบบของผู้พัฒนาเอง ปัญหาต่างๆ ที่ผู้พัฒนายังแก้ไขไม่ได้ จะถูกเก็บรวบรวมไว้ เพื่อนำไปแก้ไขต่อในอนาคต สำหรับรายละเอียดของปัญหาต่างๆ ที่ผู้พัฒนาพบ มีดังนี้

5.1 ขอบเขตของโครงงานไม่ชัดเจน

ในช่วงแรกของการพัฒนาระบบหุ่นยนต์หลายตัว ผู้พัฒนาได้ศึกษางานวิจัยที่ผ่านมา และพบว่าหุ่นยนต์แต่ละตัว จะมีวิธีระบุตำแหน่งของตัวมันเอง แต่ระบบหุ่นยนต์หลายตัวที่ผู้พัฒนาจัดทำขึ้นนั้น หุ่นยนต์แต่ละตัว จะไม่รู้ตำแหน่งและทิศทางของตนเอง ดังนั้น ผู้พัฒนาจึงต้องออกแบบสนาม และสภาพแวดล้อมของหุ่นยนต์เอง เพื่อให้หุ่นยนต์สามารถทำงานได้ โดยที่ไม่จำเป็นต้องรู้ว่า วัตถุและหุ่นยนต์ตัวอื่นอยู่ ณ ตำแหน่งใดในสนาม อย่างไรก็ตาม การออกแบบระบบในช่วงแรก ผู้พัฒนาก็ยังไม่สามารถกำหนดของเขตของโครงงานได้อย่างชัดเจน เพราะผู้พัฒนาได้พบปัญหาต่างๆ เป็นจำนวนมาก โดยสิ่งที่ผู้พัฒนาคิดว่าสามารถทำได้ แท้ที่จริงแล้วกลับทำไม่ได้อย่างที่คาดหวังไว้ จนกระทั่งผู้พัฒนาได้ทำการพัฒนาระบบไประยะหนึ่ง จึงสามารถกำหนดขอบเขตของโครงงานได้ชัดเจนขึ้น

5.2 ผู้พัฒนาขาดความรู้ในการเขียนโปรแกรมบนตัวหุ่นยนต์

การเขียนโปรแกรมบนหุ่นยนต์ จำเป็นต้องอาศัยโปรโตคอลที่มีชื่อว่า XMODEM ในการส่งโปรแกรมไปเก็บยังหน่วยความจำแฟลชบนตัวหุ่นยนต์ ซึ่งผู้พัฒนายังขาดความรู้เกี่ยวกับการติดต่อสื่อสารระหว่างคอมพิวเตอร์กับหุ่นยนต์โดยใช้โปรโตคอลนี้ และหลังจากที่ผู้พัฒนาสามารถส่งข้อมูลไปยังหุ่นยนต์ได้แล้ว ผู้พัฒนาก็ต้องศึกษาภาษา PicoC เพิ่มเติมอีก ซึ่งเป็นภาษาที่ใช้ในการสั่งงานหุ่นยนต์ Surveyor SRV-1

5.3 ฟังก์ชันการประมวลผลภาพในหุ่นยนต์ ไม่สามารถทำงานได้ ในกรณีที่มีสิ่งรบกวนในภาพมาก [8]

หุ่นยนต์ Surveyor SRV-1 มีฟังก์ชันในการตรวจจับวัตถุที่สามารถเขียนได้ด้วยภาษา PicoC แต่ผู้พัฒนาพบว่า การประมวลผลภาพ เพื่อทำการตรวจจับวัตถุ โดยใช้ฟังก์ชันของตัวหุ่นยนต์ มีปัญหาเป็นอย่างมาก กล่าวคือ ภาพนำเข้า ที่จะนำมาใช้ในการตรวจจับวัตถุ ต้องเป็นภาพที่ไม่มีสิ่งรบกวน แต่จากสภาพการทำงานจริงของหุ่นยนต์ เมื่อหุ่นยนต์เคลื่อนที่ ย่อมทำให้ภาพนำเข้าเกิดการเปลี่ยนแปลง และเกิดสิ่งรบกวนในภาพมากขึ้น ทำให้ฟังก์ชันตรวจจับวัตถุของหุ่นยนต์ ไม่สามารถทำงานได้ เมื่อมีสิ่งรบกวนในภาพมากเกินไปไป

ผู้พัฒนาได้พยายามศึกษาวิธีการกำจัดสิ่งรบกวนในภาพ แต่ผู้พัฒนาก็พบว่า ฟังก์ชันการตรวจจับวัตถุมีความเร็วในการทำงาน 150 มิลลิวินาทีต่อเฟรม และหากเพิ่มส่วนกำจัดสิ่งรบกวน ก็ยิ่งทำให้การตรวจหาวัตถุช้ามากขึ้นไปอีก ในทางกลับกัน หากนำภาพไปประมวลผลบนคอมพิวเตอร์ ก็จะได้ความเร็ว และความยืดหยุ่นในการประมวลผลมากกว่านี้ ด้วยเหตุนี้ ผู้พัฒนาจึงเลือกที่จะพัฒนาส่วนประมวลผลภาพใน Console แทน

5.4 แสงของวัตถุเปลี่ยนไป เมื่อหุ่นยนต์เคลื่อนที่เข้าใกล้วัตถุ

ในระยะแรก ผู้พัฒนาได้ใช้ทฤษฎีของการฉายภาพแบบทัศนคติ มาใช้ในการคำนวณระยะห่างระหว่างหุ่นยนต์กับวัตถุ แต่เมื่อหุ่นยนต์เคลื่อนที่เข้าใกล้วัตถุมากขึ้น สีของวัตถุจะซีดลง ทำให้หุ่นยนต์ตรวจจับวัตถุได้แย่ลง ส่งผลให้ขนาดของวัตถุที่หาได้มีความถูกต้องน้อยลง ดังนั้น ผู้พัฒนาจึงเลือกใช้ความกว้าง และขนาดของวัตถุโดยประมาณแทน แต่อย่างไรก็ตาม ความกว้าง และขนาดของวัตถุที่หามาได้ ก็ยังมีความคลาดเคลื่อนเป็นอย่างมาก เพราะความเข้มของวัตถุลดลงจนแทบจะเป็นสีขาว

ผู้พัฒนาพยายามศึกษาวิธีการปรับค่าสีของวัตถุ ให้เข้ากับสภาพแวดล้อมที่เปลี่ยนไป แต่วิธีการนี้ไม่ได้ผล เพราะหุ่นยนต์จะจำค่าสีค่าล่าสุดที่หุ่นยนต์เห็นเสมอ ตัวอย่างเช่น หากหุ่นยนต์เคลื่อนที่ไปใกล้ตำแหน่งปลายทางสีม่วงมากๆ หุ่นยนต์จะปรับค่าสีของตำแหน่งเป้าหมายให้เป็นสีที่มันเห็น ซึ่งก็คือสีที่ใกล้สีขาวมากๆ นั่นเอง และจากนั้น หุ่นยนต์ก็เคลื่อนที่ไปหยิบวัตถุอื่น และก็ทำการหาตำแหน่งปลายทางอีกครั้ง ซึ่งอยู่ไกลจากตัวมัน ทำให้ในช่วงนี้ หุ่นยนต์จะไม่สามารถหาตำแหน่งปลายทางสีม่วงได้อีกต่อไป เพราะค่าสีม่วงได้ถูกเปลี่ยนเป็นสีอื่นที่ใกล้เคียงกับสีขาวไปแล้ว ปัญหานี้ ผู้พัฒนายังไม่สามารถแก้ไขได้ แต่ก็เป็นปัญหาที่ผู้พัฒนาจะพยายามแก้ไขในอนาคต

5.5 ไม่มีวิธีการหลบหลีกหุ่นยนต์ตัวอื่นที่ดี

ถึงแม้ว่าระบบหุ่นยนต์หลายตัว จะมีความสามารถในการป้องกันการชนกัน แต่จากการทดสอบระบบ ผู้พัฒนาพบว่า วิธีการป้องกันการชนที่พัฒนาขึ้นมา แทบจะไม่สามารถทำงานได้เลย เนื่องมาจากสาเหตุ 2 ประการดังนี้

1. หุ่นยนต์มีแนวโน้มที่จะเคลื่อนที่เข้าหาตำแหน่งเป้าหมายพร้อมๆ กัน แต่เนื่องจากมุมอับของกล้อง ทำให้หุ่นยนต์แต่ละตัวไม่เห็นแถบสีที่ติดอยู่บนหุ่นยนต์ตัวอื่น ทำให้หุ่นยนต์แต่ละตัวไม่มีทางรู้เลยว่ากำลังจะมีหุ่นยนต์ตัวอื่นเคลื่อนที่มาชน และทำให้หุ่นยนต์เคลื่อนที่ต่อไป จนมาบรรจบกันและชนกันในที่สุด
2. เมื่อหุ่นยนต์พบหุ่นยนต์ตัวอื่น หุ่นยนต์จะหยุดรอ แต่หากหุ่นยนต์ตัวอื่นไม่เคลื่อนที่ หุ่นยนต์ตัวนั้นจะหยุดรอไปเรื่อยๆ ทำให้เกิดการติดตาย แต่อย่างไรก็ตาม เมื่อหุ่นยนต์หยุดรอนานถึงช่วงเวลาหนึ่ง หุ่นยนต์จะเบี่ยงซ้าย และพยายามเคลื่อนที่ออกไป แต่สุดท้าย ก็จะมีโอกาสสูงมากที่หุ่นยนต์จะหันมาเจอหุ่นยนต์ตัวนั้นอีก และก็จะวนการทำงานเช่นนี้ไปเรื่อยๆ ไม่มีที่สิ้นสุด

ปัญหานี้ ผู้พัฒนายังไม่สามารถหาวิธีการที่ดีกว่านี้ได้ แต่ก็เป็นปัญหาที่ผู้พัฒนาจะพยายามแก้ไขในอนาคต

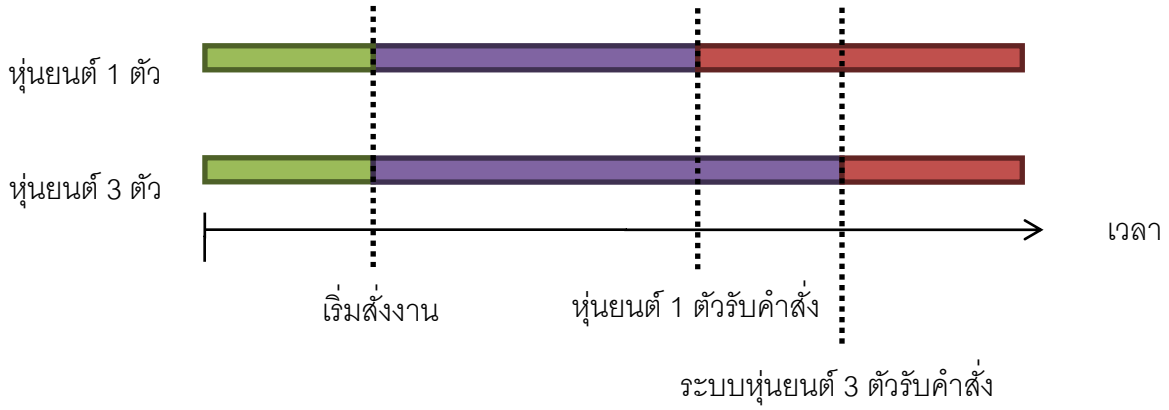
5.6 ลักษณะทางกายภาพของหุ่นยนต์แต่ละตัวไม่เหมือนกัน

หุ่นยนต์แต่ละตัวที่นำมาใช้งาน ถึงแม้จะเป็นรุ่นเดียวกัน ก็มีความสามารถในการทำงานแตกต่างกัน เนื่องจากความสึกหรอที่แตกต่างกัน ความแตกต่างที่ผู้พัฒนาพบ มีทั้งความเร็วของมอเตอร์ ความชัดของกล้อง รวมทั้งความเร็วในการส่งข้อมูล ความแตกต่างทางกายภาพ ทำให้หุ่นยนต์แต่ละตัว ทำงานไม่เหมือนกัน แม้ว่าจะใช้โปรแกรมชุดเดียวกันก็ตาม และอาจทำให้ระบบหุ่นยนต์ทำงานผิดพลาดในที่สุด

5.7 ระบบเครือข่ายที่ใช้มีความล่าช้า

ระบบเครือข่ายไร้สายที่ใช้ ถึงแม้ว่าจะเป็นแบบ 802.11g ที่มีความเร็วในการส่งข้อมูล 54Mbps ซึ่งจากการคำนวณพบว่า หุ่นยนต์สามารถส่งภาพมาให้ยังคอมพิวเตอร์ได้ 30 ภาพต่อวินาที แต่จริงๆ แล้ว ข้อมูลที่ส่งไป จะต้องถูกเติมด้วยข้อมูลอีกชุดหนึ่งที่เรียกว่า Header ทำให้อัตราการรับ-ส่งข้อมูล ช้ากว่าที่ได้คำนวณ ไว้ และอัตราการรับ-ส่งภาพ ที่ช้ากว่า 30 ภาพต่อวินาที ก็อาจทำให้การประมวลผลไม่ทันท่วงที

นอกจากนี้ ถึงแม้ว่าระบบเครือข่าย จะสามารถรองรับการส่งข้อมูลในปริมาณมากได้ แต่การส่งภาพโดยหุ่นยนต์หลายตัว จะทำให้เกิดความล่าช้า ทั้งในการส่งภาพจากตัวหุ่นยนต์ไปยังคอมพิวเตอร์ และการส่งคำสั่งจากคอมพิวเตอร์ไปยังหุ่นยนต์ ดังรูปที่ 21



รูปที่ 21 ความล่าช้าในการสั่งงานระบบหุ่นยนต์หลายตัว เทียบกับหุ่นยนต์ตัวเดียว

บทที่ 6 แนวทางการพัฒนาต่อในอนาคต

เนื่องด้วยระยะเวลาที่จำกัด และความไม่คุ้นเคยของผู้พัฒนาต่อระบบหุ่นยนต์หลายตัว ทำให้มีหลายๆ ส่วนในโครงงาน ไม่สำเร็จตามที่ผู้พัฒนาได้คาดหวังไว้ อย่างไรก็ตาม ผู้พัฒนาได้เสนอแนวทางการพัฒนาต่อในอนาคต เพื่อเพิ่มประสิทธิภาพให้กับระบบหุ่นยนต์หลายตัว ดังนี้

1. พัฒนาวิธีการหลีกหุ่นยนต์ตัวอื่นภายในสนาม ซึ่งอาจใช้วิธีการลบภาพพื้นหลังของสนามและวัตถุออก ภาพที่เหลือคือ ภาพของหุ่นยนต์เท่านั้น ทำให้สามารถหาตำแหน่งของหุ่นยนต์ได้
2. พัฒนาวิธีการตรวจจับวัตถุ แทนที่จะพิจารณาจากขนาดของวัตถุ ก็ใช้วิธีการตรวจหาขอบของวัตถุแทน
3. พัฒนาหุ่นยนต์ให้สามารถตรวจจับการทำงานแบบอาร์บิทรารี่ (Arbitrary) ได้ ซึ่งสามารถใช้วิธีการกำหนดระยะเวลามากสุด ที่ยอมให้หุ่นยนต์ทำงานก็ได้
4. พัฒนาระบบหุ่นยนต์หลายตัว เพื่อให้หุ่นยนต์สามารถทำนายการเคลื่อนที่ล่วงหน้าได้ ทำให้หุ่นยนต์ไม่จำเป็นต้องติดต่อกับ Console ตลอดเวลา
5. นำแขนจับของวัตถุออก และพัฒนาให้หุ่นยนต์เรียนรู้วิธีการจับวัตถุที่ถูกต้องได้

บทที่ 7 ข้อสรุปและข้อเสนอแนะ

หุ่นยนต์แต่ละตัวภายในระบบหุ่นยนต์หลายตัวที่พัฒนาขึ้น สามารถแบ่งงานกันทำได้จริง แต่หุ่นยนต์ 2 ตัวจะ ได้รับปริมาณงานที่ไม่เท่ากัน และถึงแม้ว่าหุ่นยนต์ 3 ตัว สามารถแบ่งงานกันทำได้ดีกว่าหุ่นยนต์ 2 ตัว แต่ยิ่งเพิ่มจำนวนหุ่นยนต์ ก็จะยิ่งจะเกิดปัญหามาก เมื่อหุ่นยนต์แต่ละตัวต้องทำการประสานงานกัน และเมื่อหุ่นยนต์กำลังจะชนกัน ส่งผลให้เวลาในการทำงานของระบบหุ่นยนต์หลายตัว น้อยกว่าหรือไม่ต่างจากหุ่นยนต์ตัวเดียวมากนัก ยิ่งไปกว่านั้น แม้ว่าระบบหุ่นยนต์จะทนต่อวามผิดพลาด แต่ก็ทนได้เฉพาะความผิดพลาดแบบแครช (Crash) เท่านั้น ไม่สามารถทนต่อความผิดพลาดแบบอาร์บิทรารี่ (Arbitrary) ได้ ดังนั้น ระบบหุ่นยนต์หลายตัว จึงไม่สามารถทำงานต่อไปได้ เมื่อเกิดการชนกัน หรือหยิบวัตถุผิด

จากผลการทดสอบ ผู้พัฒนายังพบอีกด้วยว่า ระบบหุ่นยนต์หลายตัวที่ดี หุ่นยนต์แต่ละตัว จะต้องได้รับปริมาณงานที่เท่าๆ กัน โดยที่แต่ละงานต้องแยกกันอย่างอิสระ และหุ่นยนต์ภายในกลุ่ม จะต้องมีความสามารถในการจัดการความผิดพลาดที่ดี เพื่อให้ระบบสามารถทำงานต่อไปได้อย่างราบรื่น นอกจากนี้ ระบบหุ่นยนต์ที่ดี หุ่นยนต์ภายในกลุ่ม ควรจะต้องเรียนรู้ความสามารถและการกระทำของตัวมันเองได้ด้วย เพื่อที่ว่าหุ่นยนต์จะได้ทำงานได้แม่นยำมากขึ้น

บทที่ 8 เอกสารอ้างอิง

[1] รศ.ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา. **การประสานงานและการทำความตกลงระหว่างโพรเซส**. จุฬาลงกรณ์มหาวิทยาลัย, 2006.

[2] Lynne E. Parker, “Heterogeneous Multi-Robot Cooperation,” Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy at the MASSACHUSETTS INSTITUTE OF TECHNOLOGY, Feb 1994.

[3] J.Spletzer, A.K.Das, R.Fierro, C.J.Taylor, V.Kumar, and J.P.Ostrowski, “Cooperative Localization and Control for Multi-Robot Manipulation,” GRASP Laboratory – University of Pennsylvania, Philadelphia.

[4] Jonathan Fink, M.Ani Hsieh, and Vijay Kumar, “Multi-Robot Manipulation via Caging in Environments with Obstacles,” GRASP Laboratory – University of Pennsylvania, Philadelphia.

[5] Guilherme A.S.Pereira[a], Vijay Kumar[b], Mario F.M.Campos[a], “Decentralized Algorithms for Multi-Robot Manipulation via Caging,” [a]VERLab, DCC, Universidade Federal de Minas Gerais, Av. Antonio Carlos 6627, Belo Horizonte, MG 31270-010 Brazil, [b]GRASP Lab., University of Pennsylvania, 3330 Walnut Street, Philadelphia, PA 19104-6228 USA.

[6] Peng Cheng, Jonathan Fink, Vijay Kumar, “Abstractions and Algorithms for Cooperative Multiple Robot Planar Manipulation,” GRASP Lab., University of Pennsylvania, 3330 Walnut Street, Philadelphia, PA 19104 USA.

[7] Ashley Stroupe, Terry Huntsberger, Avi Okon, Hrand Aghazarian, Matthew Robinson, “Behavior-Based Multi-Robot Collaboration for Autonomous Construction Tasks,” Jet Propulsion Laboratory / California Institute of Technology 4800 Oak Grove Drive, Pasadena, CA 91 109.

[8] AForge.NET. **AForge.NET:: Framework Features – Surveyor Robotics.** [Online] Available from: http://www.aforgenet.com/framework/features/surveyor_robotics.html [2010].

[9] Surveyor Corporation. **Surveyor SRV-1 Blackfin Setup.** [Online] Available from: http://www.surveyor.com/blackfin/SRV_setup_bf.html [2010, Apr 24].

[10] Surveyor Corporation. **Definition of the SRV-1 Control Protocol (Blackfin Version).**
[Online] Available from: http://www.surveyor.com/SRV_protocol.html [2010, Apr 24].

[11] Webopedia. **What's Xmodem?.**
[Online] Available from: http://www.webopedia.com/TERM/X/Xmodem.html [2010].

[12] Wikipedia. **Tera Term.**
[Online] Available from: http://en.wikipedia.org/wiki/Tera_Term [2010, Sep 8].

[13] Surveyor Corporation. **SRV-1 C Interpreter.**
[Online] Available from: http://www.surveyor.com/C.html [2010, May 7].

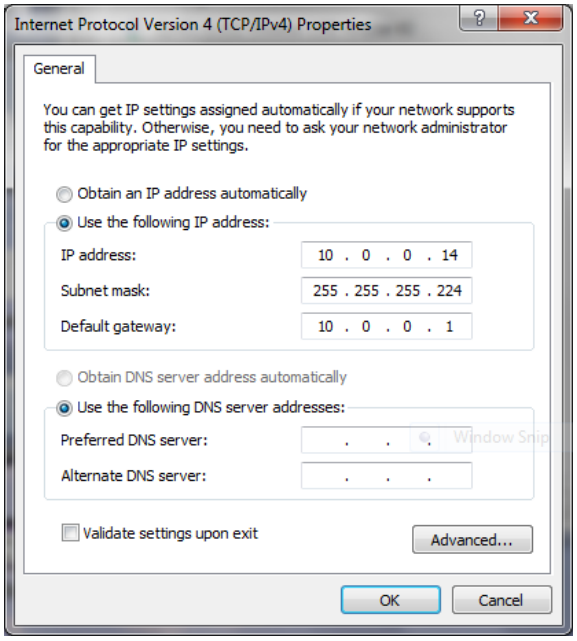
[14] Surveyor Robotics Forum. **Using the vblob function in PicoC.**
[Online] Available from: http://www.surveyor.com/cgi-bin/yabb2/YaBB.pl?num=1250473596/0
[2010, Aug 16].

บทที่ 9 ภาคผนวก

9.1 วิธีการตั้งค่าระบบเครือข่ายไร้สาย

9.1.1 การตั้งค่า IP ให้กับเครื่องคอมพิวเตอร์

1. เข้าไปที่ Wireless Network Properties และเลือกที่แถบ Internet Protocol Version 4 (IPv4) จากนั้นให้คลิกปุ่ม Properties จะปรากฏหน้าต่างขึ้นมาดังรูป

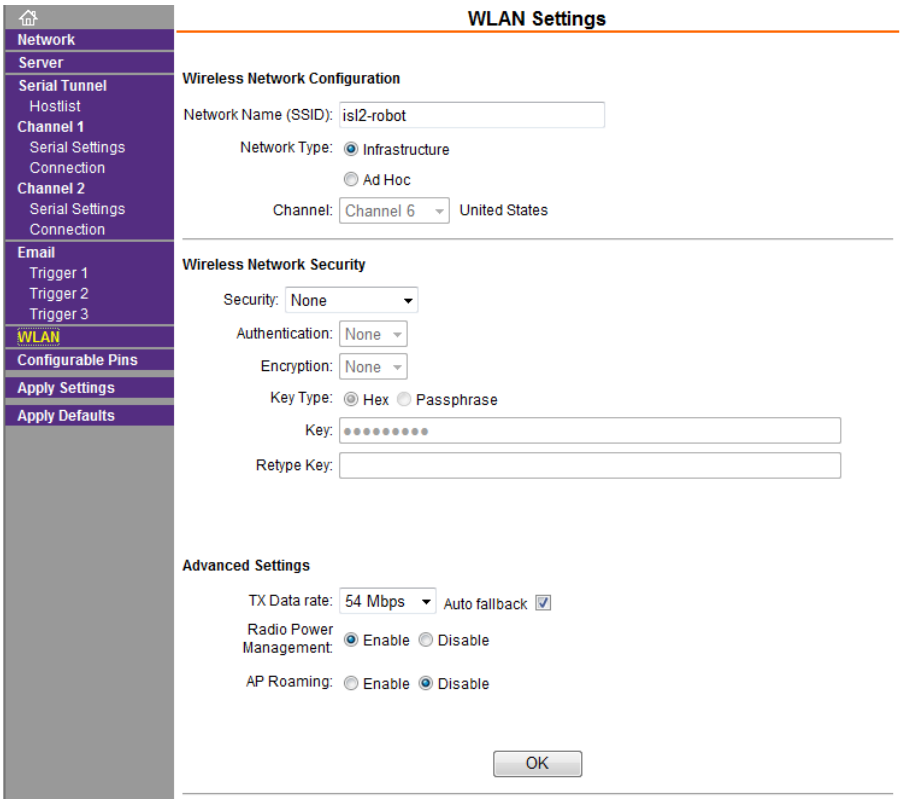


รูปที่ 22 วิธีการตั้งค่า IP ให้กับเครื่องคอมพิวเตอร์

2. ตั้งค่า IP address, Subnet mask และ Default gateway ให้ถูกต้องตามที่ได้ออกแบบไว้

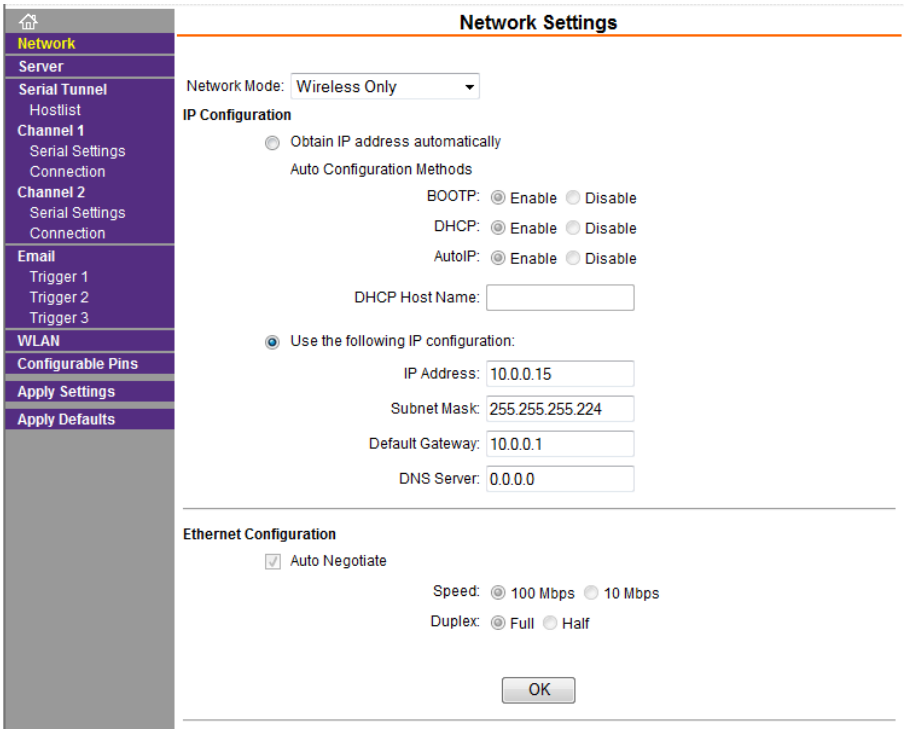
9.1.2 การตั้งค่า IP ให้กับหุ่นยนต์ Surveyor SRV-1 [8]

1. เปิดโปรแกรม Internet Explorer จากนั้น ให้ใส่ค่า IP ของหุ่นยนต์ที่ต้องการตั้งค่า IP ใหม่ (ค่าเริ่มต้นคือ 169.254.0.10) จะปรากฏเมนูทางด้านซ้าย สำหรับตั้งค่าหุ่นยนต์ ให้เลือกเมนู WLAN จะปรากฏหน้าต่างดังรูป ในช่อง Network Name (SSID) ให้ตั้งชื่อ Access point ที่เราต้องการให้หุ่นยนต์เชื่อมต่อ และเลือก Network Type เป็น Infrastructure



รูปที่ 23 การตั้งค่า Network ให้กับหุ่นยนต์ Surveyor SRV-1

2. จากนั้นเลือกเมนู Network จะปรากฏหน้าต่างดังรูป ในช่อง Network Mode ให้เลือกเป็น Wireless Only จากนั้น ให้เลือก IP Configuration เป็น Use the following IP configuration แล้วจึงตั้ง IP address, Subnet mask และ Default gateway ให้ถูกต้องตามที่ได้ออกแบบไว้



รูปที่ 24 การตั้งค่า IP ให้กับหุ่นยนต์ Surveyor SRV-1

9.1.3 การตั้งค่า IP ของหุ่นยนต์ ให้กับ Console

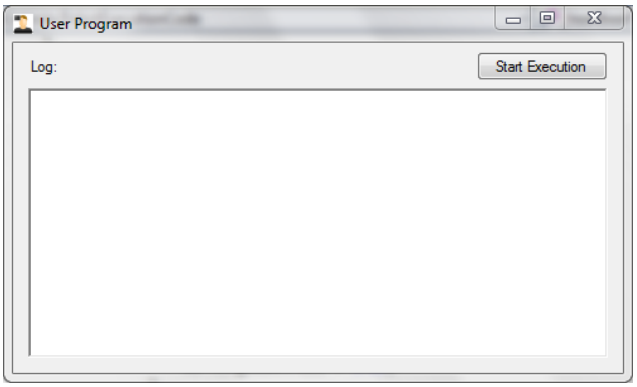
เปิดไฟล์ชื่อ RobotInfo.txt และทำการเพิ่มรายชื่อหุ่นยนต์ กับ IP ของหุ่นยนต์นั้น โดยมีรูปแบบการเพิ่มดังนี้

<ชื่อหุ่นยนต์> <หมายเลขหุ่นยนต์>

<IP ของหุ่นยนต์>

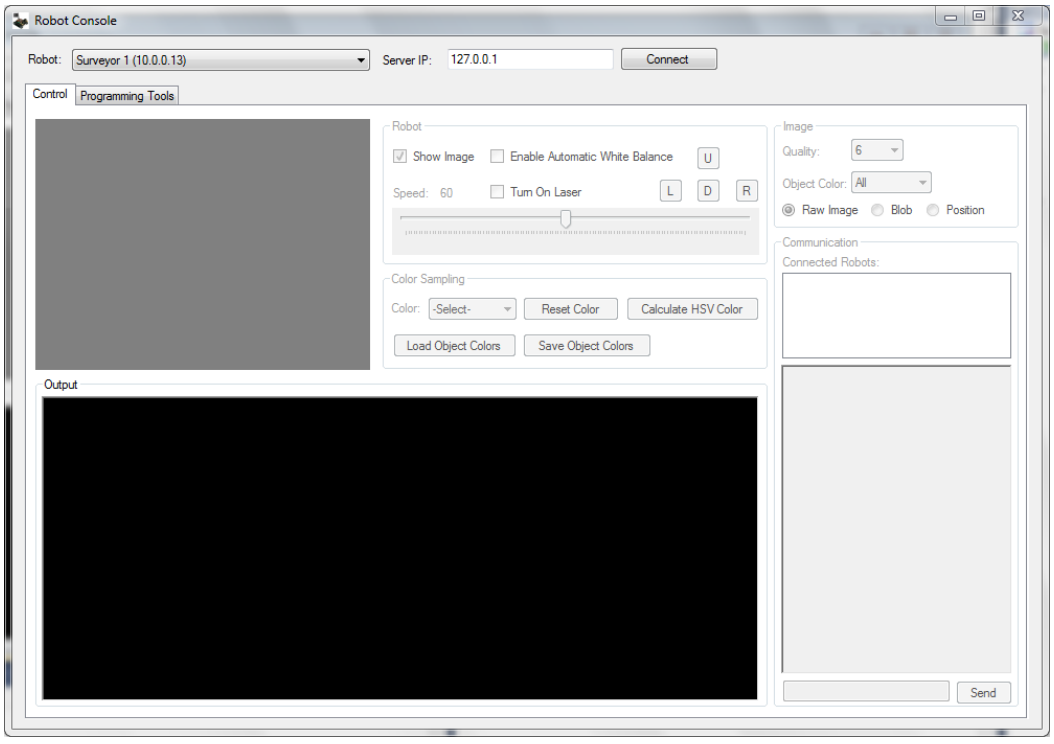
9.2วิธีการใช้งานระบบเบื้องต้น

1. ดับเบิลคลิกที่ไอคอน  เพื่อเปิดโปรแกรมผู้ใช้ โปรแกรมจะถูกเปิดขึ้นมาดังรูป



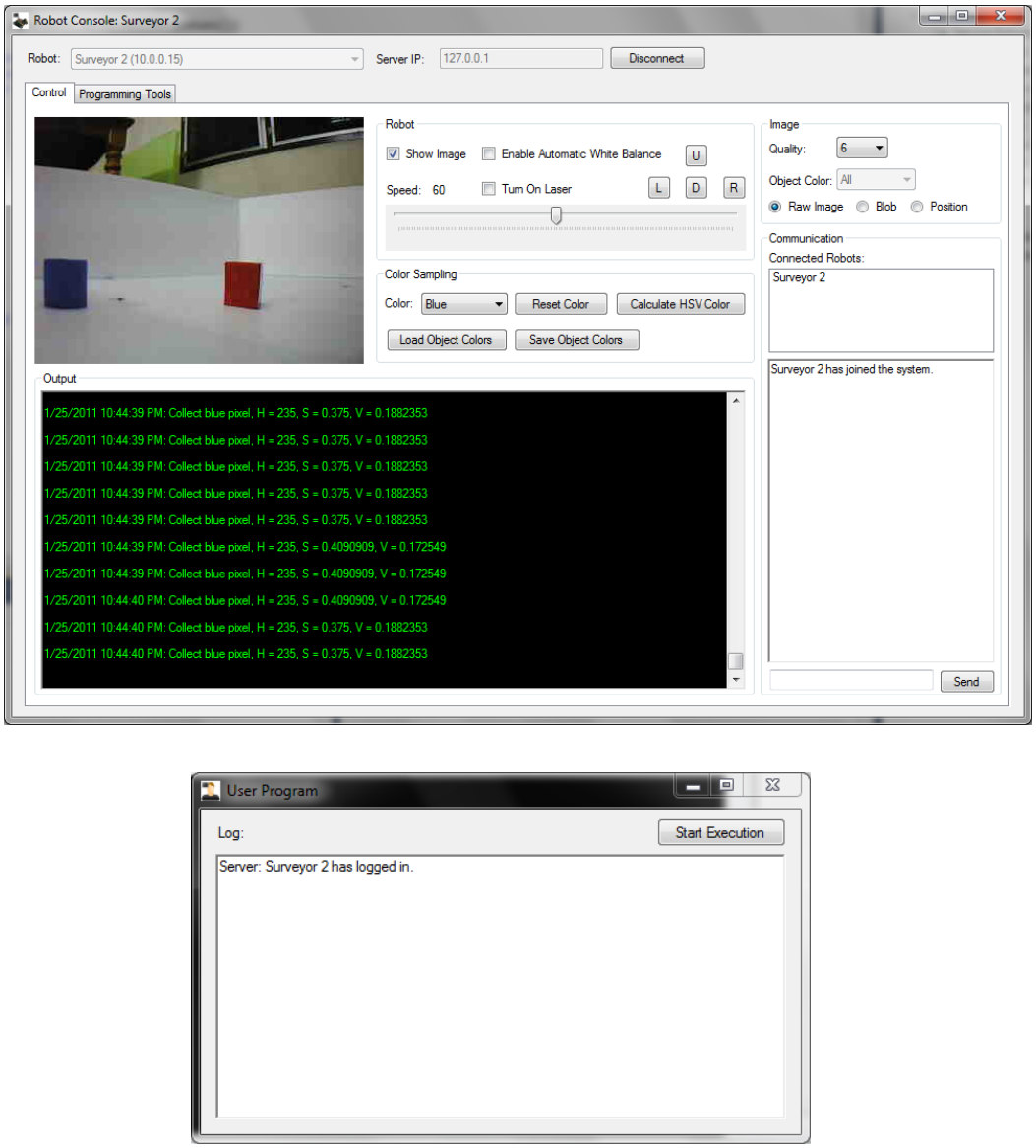
รูปที่ 25 หน้าตาเริ่มต้นของโปรแกรมผู้ใช้

2. ดับเบิลคลิกที่ไอคอน  เพื่อเปิด Console โปรแกรมจะถูกเปิดขึ้นมาดังรูป



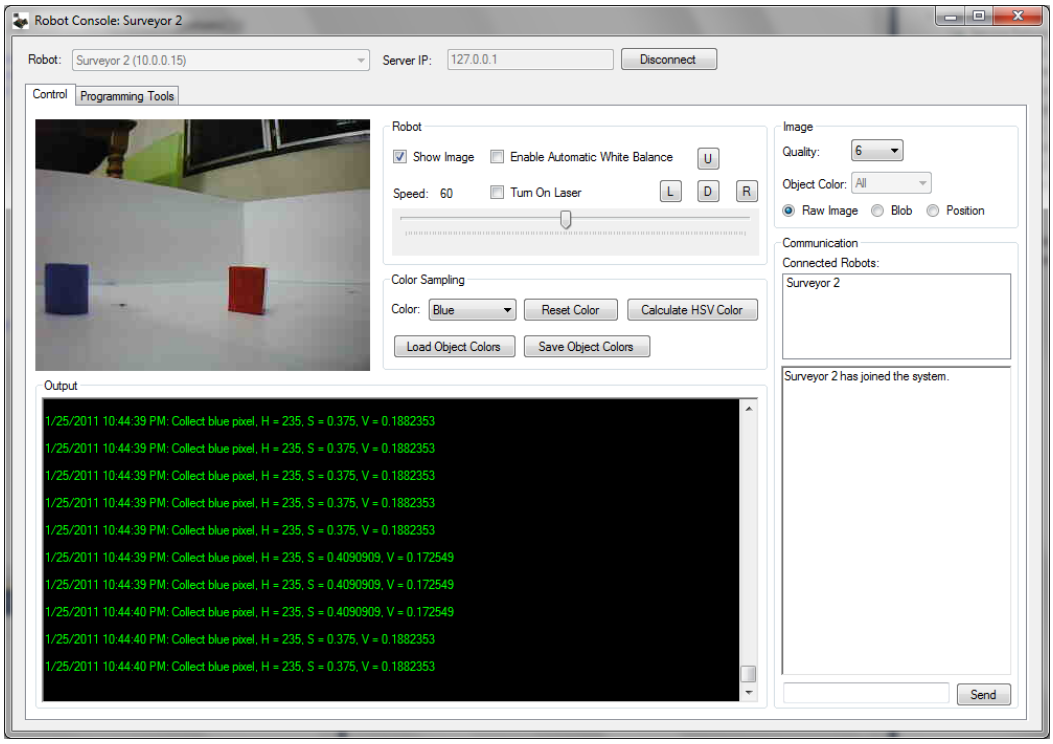
รูปที่ 26 หน้าตาเริ่มต้นของ Console

3. เลือกหุ่นยนต์ที่ต้องการเชื่อมต่อ จากนั้นคลิกปุ่ม Connect โปรแกรมจะทำการเชื่อมต่อเข้ากับโปรแกรมผู้ใช้และหุ่นยนต์ เมื่อทำการเชื่อมต่อได้แล้ว ภาพจากกล้องของหุ่นยนต์จะถูกนำมาแสดงบนหน้าจอของโปรแกรมนี้ และผู้ใช้สามารถส่งคำสั่งพื้นฐานไปควบคุมได้ เช่น สั่งให้มอเตอร์หมุน หรือเปิดเลเซอร์ เป็นต้น ส่วนฝั่งผู้ใช้โปรแกรม จะมีการรายงานว่าหุ่นยนต์ตัวนี้ ได้ถูกเพิ่มเข้ามาในระบบหุ่นยนต์หลายตัว



รูปที่ 27 การเปลี่ยนแปลงในฝั่ง Console (บน) และโปรแกรมผู้ใช้ (ล่าง)

4. ขั้นตอนนี้จะเป็นการหาค่าสีของวัตถุ ในส่วนของ Color Sampling ให้เลือกสีของวัตถุที่ต้องการเก็บค่าตามรูป คือ สีน้ำเงิน จากนั้น ให้กดเมาส์ปุ่มซ้ายค้างไว้ และกวาดเมาส์ไปยังบริเวณที่มีสีน้ำเงิน เพื่อทำการเก็บตัวอย่างจุดภาพสีน้ำเงิน สังเกตตรง Output จะขึ้นค่าสี HSV ของจุดภาพที่ทำการเก็บตัวอย่าง ทำเช่นนี้ไปเรื่อยๆ จนครบทั้ง 4 สี

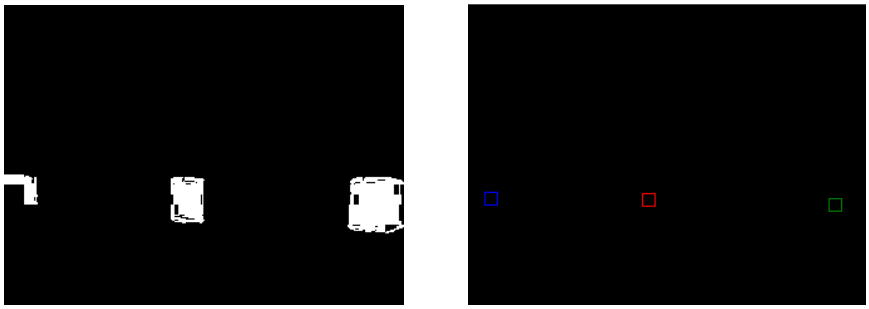


รูปที่ 28 Output แสดงค่าสีของจุดภาพที่เก็บมา

5. คลิกปุ่ม Calculate HSV Color ตรงช่อง Output จะแสดงค่าสีของวัตถุ ที่คำนวณมาจากจุดภาพหลายๆ จุดที่เลือกมา

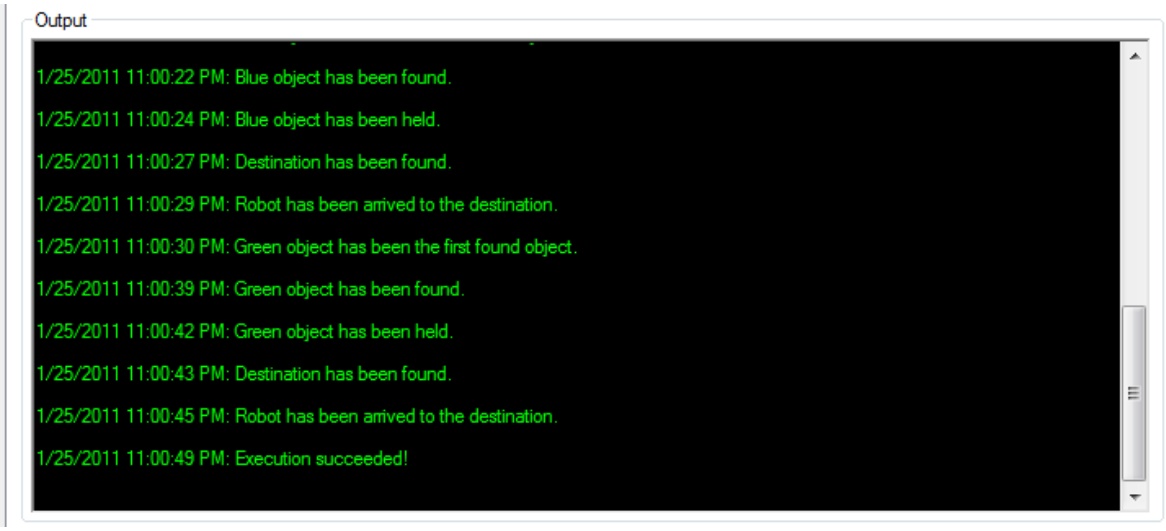
รูปที่ 29 Output แสดงค่าสีของวัตถุที่คำนวณได้

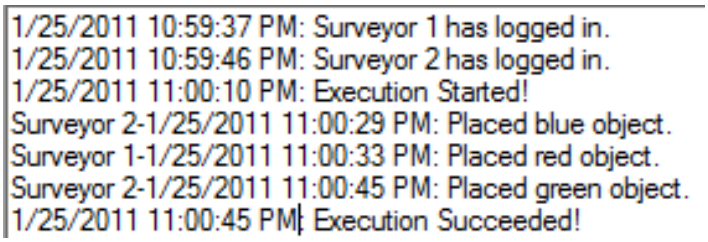
6. ในส่วนของ Image สามารถเลือกได้ว่า จะให้แสดงภาพจากกล้อง (Raw Image) ภาพขาวดำที่แสดงถึงจำนวนองค์ประกอบ (Blob) หรือตำแหน่งของวัตถุได้ (Position)



รูปที่ 30 ส่วนแสดงภาพที่สามารถเปลี่ยนรูปแบบการแสดงผล
ไปเป็นแบบจำนวนองค์ประกอบ (ซ้าย) หรือตำแหน่งของวัตถุ (ขวา) ได้

7. คลิกปุ่ม Start Execution ที่โปรแกรมผู้ใช้ ระบบหุ่นยนต์จะเริ่มทำงาน และในระหว่างที่ระบบหุ่นยนต์ทำงาน ก็จะมีการรายงานสถานะเป็นระยะๆ ทั้งในโปรแกรมผู้ใช้ และ Console จนกระทั่งหุ่นยนต์ทำงานเสร็จทุกตัว โปรแกรมผู้ใช้ และ Console จึงแจ้งสถานะการทำงานว่าทำงานเสร็จแล้ว

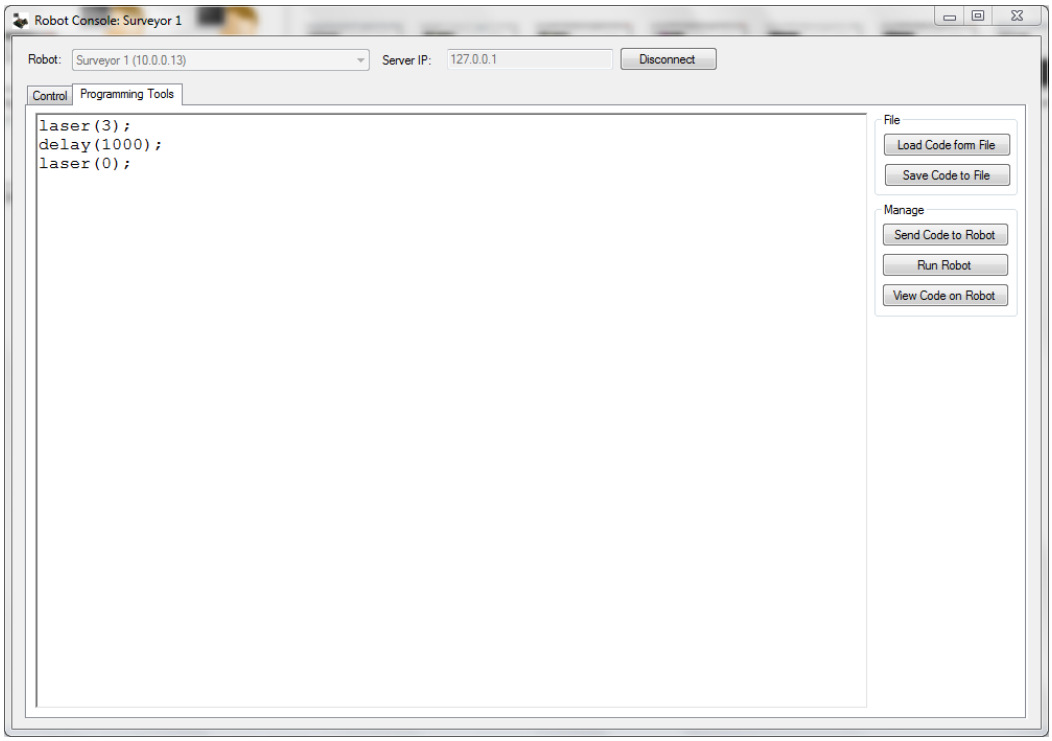




รูปที่ 31 สถานะการทำงานของระบบหุ่นยนต์จะถูกรายงาน

ทั้งฝั่ง Console (ซ้าย) และฝั่งโปรแกรมผู้ใช้ (ขวา)

หมายเหตุ: หากผู้ใช้ต้องการเขียนโปรแกรมลงบนหน่วยความจำแฟลชของหุ่นยนต์ ก็สามารถทำได้โดยใช้เมนู Programming Tools ใน Console ซึ่งมีหน้าตาดังรูป



รูปที่ 32 แสดงหน้าต่างของ Console ที่ใช้สำหรับการเขียนโปรแกรม

ลงบนหน่วยความจำแฟลชของหุ่นยนต์