# หุ่นยนต์เคลื่อนที่ตามพิกัดโดยใช้การประมวลผลจากกล้อง
# COORDINATE CONTROLLED TRACKING ROBOT WITH IMAGE PROCESSING


เพชรมงคล งามแสงพฤกษ์
มธุรินทร์ เลาะวิธี
วสุท ช่างเรือ


ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมแมคคาทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2559

# หุ่นยนต์เคลื่อนที่ตามพิกัดโดยใช้การประมวลผลจากกล้อง

# COORDINATE CONTROLLED TRACKING ROBOT WITH IMAGE PROCESSING

เพชรมงคล งามแสงพฤกษ์

มธุรินทร์ เลาะวิธี

วสุท ช่างเรือ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมแมคคาทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2559

# COORDINATE CONTROLLED TRACKING ROBOT WITH IMAGE PROCESSING

PHETMONGKOL NGAMSAENGPRUEK

MATHURIN LOHVITHEE

WASUT CHANGRUA

THIS THESIS IS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN MECHATRONICS ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2016

**ปริญญานิพนธ์ปีการศึกษา 2559**

ภาควิชาวิศวกรรมการวัดและควบคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง หุ่นยนต์เคลื่อนที่ตามพิกัดโดยใช้การประมวลผลจากกล้อง

COORDINATE CONTROLLED TRACKING ROBOT WITH IMAGE PROCESSING


ผู้จัดทำ นายเพชรมงคล งามแสงพฤกษ์ 56010892

นางสาวมธุรินทร์ เลาะวิธี 56010965

นายวสุท ช่างเรือ 56011100


.....อาจารย์ที่ปรึษา

(ผู้ช่วยศาสตราจารย์สุมิตร พนาอุดมทรัพย์)

# หุ่นยนต์เคลื่อนที่ตามพิกัดโดยใช้การประมวลผลจากกล้อง

โดย

นายเพชรมงคล งามแสงพฤกษ์ 56010892

นางสาวมธุรินทร์ เลาะวิธี 56010965

นายวสุท ช่างเรือ 56011100

อาจารย์ที่ปรึกษา

ผู้ช่วยศาสตราจารย์สุมิตร พนาอุดมทรัพย์

ปีการศึกษา 2559

## บทคัดย่อ

วิทยานิพนธ์นี้มีวัตถุประสงค์เพื่อแสดงการศึกษา และการวิจัยเกี่ยวกับการควบคุมหุ่นยนต์ด้วยการประมวลผลภาพจากกล้อง โดยโครงงานนี้จะต้องสร้างหุ่นยนต์ และระบบขนาดเล็กที่เปรียบเสมียนระบบขนส่งแบบจำลอง จะแบ่งการใช้งานเป็น 4 ส่วนใหญ่ๆ 1. การประมวลผลภาพ โครงงานนี้ใช้การประมวลผลภาพเป็นการจับสีเป็นหลัก เพื่อระบุตำแหน่งของหุ่นยนต์ และให้หุ่นยนต์เคลื่อนที่ไปตามพิกัดที่ระบุไว้ โดยมีขอบเขตพิกัดที่สามารถระบุได้ ให้อยู่ในระยะฉายของกล้อง 2. ตัวหุ่นยนต์ ด้านการออกแบบหุ่นยนต์จะออกแบบให้มีขนาดเล็ก เนื่องจากสามารถควบคุมได้ง่ายโดยใช้กล้องตัวเดียว อีกทั้งใช้เซอร์โวมอเตอร์เป็นตัวขับเคลื่อนเพราะสามารถกำหนดการหมุนได้แม่นยำกว่ามอเตอร์ธรรมดา 3. การสร้างหน้าจอคำสั่ง เพื่อใช้ในการควบคุมระบบการขนส่งแบบจำลองนี้ และ 4. การสื่อสาร การส่งข้อมูลระหว่างการประมวลผลภาพในคอมพิวเตอร์ไปยังตัวหุ่นยนต์จะใช้ระบบไร้สายเป็นตัวกลาง โครงงานนี้ทางผู้จัดทำคาดหวังว่าจะสามารถเป็นต้นแบบ เพื่อใช้พัฒนาระบบขนส่งอัตโนมัติด้วยกล้องในอนาคตอันใกล้

# COORDINATE CONTROLLED TRACKING ROBOT WITH IMAGE PROCESSING

By

Mr. Phetmongkol Ngamsaengpruek

Ms. Mathurin Lohvithee

Mr. Wasut Changrua

Advisor

Asst. Prof. Sumit Panaudomsup

Academic Year 2016

## ABSTRACT

The thesis is written to demonstrate the theory and research for Coordinate Tracking Robot by using image processing. This project intends to be a small model for industrial transport system. In image processing section, color tracking is used to determine the actual coordinate of the robot and also controls the robot to the given position. The scope of coordinate tracking depends on the pixel of the Webcam. Robot is designed to be a model of transport in order to conveniently controlled by a single Webcam. Therefore, it is designed into a small-size robot using servo motor to drive it. The system can be controlled by an interface which can give a specific coordinate as a target of the system. Moreover, the communication between the robot, the program is through the wireless system or via Bluetooth HC-05. This project can hopefully be applied for an automatic transport system that controlled be image processing and can be useful for further studies.

# กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้และโครงงานชิ้นนี้ดำเนินการสำเร็จลุล่วงไปได้ด้วยดีด้วยความช่วยเหลือจากอาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์สุมิตร พนาอุดมทรัพย์ ซึ่งท่านได้ให้ข้อคิดเห็น และคำแนะนำต่างๆ อันเป็นประโยชน์ในการดำเนินการทำโครงงานนี้เป็นอย่างมาก อีกทั้งยังช่วยคิดแนวทางแก้ปัญหาต่างๆ ที่เกิดขึ้นระหว่างทำโครงงานชิ้นนี้

ขอขอบคุณคณะอาจารย์ และเจ้าหน้าที่ภาควิชาวิศวกรรมการวัด และควบคุม เพื่อนพี่น้อง ณ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง รวมถึงผู้ปกครองและครอบครัวที่ให้การสนับสนุนในด้านต่างๆ อีกทั้งยังคอยเป็นกำลังใจมาโดยตลอดจนสำเร็จการศึกษา คณะผู้จัดทำรู้สึกประทับใจ และซาบซึ้งเป็นอย่างยิ่งสำหรับทุกความช่วยเหลือ

ผู้จัดทำ

นายเพชรมงคล งามแสงพฤกษ์

นางสาวมธุรินทร์ เลาะวิธี

นายวสุท ช่างเรือ

# สารบัญ

# สารบัญ (ต่อ)

# สารบัญรูป

# สารบัญรูป(ต่อ)

# สารบัญตาราง

# บทที่ 1
# บทนำ

## 1.1 ความสำคัญและที่มาของโครงงาน

ปัจจุบันนี้ถือได้ว่าเป็นสังคมแห่งการพัฒนา และการเรียนรู้แบบเปิด ซึ่งมีหลากหลายภาคแขนงวิชา เป็นการบูรณาการองค์ความรู้ทุกอย่างเข้าด้วยกันทั้งด้านเทคโนโลยี การสื่อสาร การคมนาคม ฯลฯ เพื่อทำให้เกิดสิ่งใหม่เป็นประโยชน์ต่อการเรียนรู้ และพัฒนาต่อไปในภายหน้า ปัจจุบันเทคโนโลยีของหุ่นยนต์เจริญก้าวหน้าอย่างรวดเร็ว และเริ่มเข้ามามีบทบาทกับชีวิตของมนุษย์ในด้านต่างๆ เช่น ด้านการแพทย์ ด้านอุตสาหกรรมการผลิต ด้านงานสำรวจ ด้านการเกษตร และในอนาคตอาจมีการนำหุ่นยนต์มาใช้แทนแรงงานคนมากขึ้น อาจไปถึงขั้นเกิดระบบการผลิตในอุตสาหกรรมที่ปราศจากมนุษย์ ทั้งหมดเพื่อเพิ่มความแม่นยำ ลดระยะเวลาในการดำเนินงาน และเพื่อทำงานที่หนักเกินกำลังมนุษย์

การขนส่งภายในโรงงานนั้นถือเป็นสิ่งที่เกิดขึ้นบ่อยครั้งในปัจจุบัน หลักใจความสำคัญที่ทำให้เกิดโครงงานนี้ขึ้นมาคือ การมองเห็นปัญหาการจราจรในการขนส่งในอุตสาหกรรม ซึ่งปัญหาไม่ได้เกิดมาจากปริมาณของรถ แต่เกิดจากความไม่เป็นระเบียบของตัวบุคคล สาเหตุที่ใช้กล้องเป็นตัวรับตำแหน่ง และควบคุมรถส่งของนั้น เนื่องจากปัจจุบันทุกๆ ที่ล้วนมีกล้องวงจรปิด ซึ่งสามารถนำมาประยุกต์ใช้ได้เพียงการเขียนโปรแกรมเพื่อเพิ่มความสามารถในการใช้งาน การใช้กล้องยังเปรียบเสมือนมีคนดูแลควบคุมการจราจรภายในเพียงคนเดียว ทำให้ประหยัดทั้งเวลา และแรงงาน

ด้วยเหตุนี้โครงงานนี้จึงเปรียบเสมือนศูนย์กลางการควบคุมเพียงคนเดียว ถือเป็นการแก้ปัญหาที่ตรงจุด

## 1.2 วัตถุประสงค์ในการทำโครงงาน

1. เพื่อศึกษา และทำความเข้าใจเกี่ยวกับการสร้างโค้ด เพื่อควบคุมเซอร์โวของหุ่นยนต์ให้ไปตามทิศทางที่ต้องการ โดยใช้โปรแกรม Arduino ในการเขียน
2. เพื่อศึกษาและทำความเข้าใจเกี่ยวกับการประมวลผลภาพด้วยการตรวจจับสี ซึ่งเขียนในโปรแกรมโดยใช้ภาษา Python
3. เพื่อสร้างระบบต้นแบบของรถส่งของอัตโนมัติภายในอุตสาหกรรม ที่สามารถควบคุมด้วยกล้องที่ฉายภาพจากมุมบน ซึ่งสามารถแยกแยะสิ่งกีดขวางกับหุ่นยนต์ และยังสามารถควบคุมการจราจรของรถส่งของได้

## 1.3 ขอบเขตของโครงงาน

1. โครงงานนี้ใช้กล้องจับสี ทิศทาง และการเคลื่อนไหวเพียงตัวเดียวเท่านั้น ทำให้ครอบคลุมพื้นที่ได้จำกัด

2. ในส่วนของการประมวลผลภาพจะใช้การแบ่งสีเป็นตัวระบุตัวหุ่นยนต์ พื้นที่สามารถวิ่งไปตามพิกัดที่ต้องการได้

3. โครงงานนี้เป็นขนาดจำลองของรถส่งอัตโนมัติภายในอุตสาหกรรม

## 1.4 รายละเอียดของรายงาน

เนื้อหาที่จะกล่าวในรายงานฉบับนี้ประกอบด้วย

บทที่ 1 บทนำ กล่าวถึง วัตถุประสงค์ ขั้นตอนการศึกษา ขอบเขตของการศึกษา รายละเอียดของรายงานในแต่ละบท ประโยชน์ที่คาดว่าจะได้รับ และแผนการดำเนินงาน

บทที่ 2 ทฤษฎี และความรู้ที่เกี่ยวข้องของหลักการ และทฤษฎีที่เกี่ยวข้องกับอุปกรณ์ที่ใช้ในโครงงาน การคำนวณ และโปรแกรมในการประมวลผลภาพ

บทที่ 3 หลักการออกแบบ และควบคุมการเคลื่อนที่ของหุ่นยนต์ และหลักการเขียนโปรแกรมที่ใช้ในการในการประมวลผลภาพ

บทที่ 4 การทดลอง

บทที่ 5 บทวิจารณ์ สรุปผลการดำเนินงาน ปัญหาที่เกิดขึ้น และแนวทางการปรับปรุง

## 1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้ใช้ความคิดในการทำโครงงาน เพื่อเป็นการฝึกให้มีการทำงานเป็นระบบ มีความสามัคคี และมีความรับผิดชอบมากขึ้น

2. เข้าใจการเขียนโปรแกรมด้วยคำสั่งภาษา Python และการควบคุมการเคลื่อนที่ของหุ่นยนต์ด้วย Arduino มากยิ่งขึ้น

3. เข้าใจการตรวจจับสีด้วยการใช้ Image Processing ในการใช้ชุดคำสั่งของ OpenCV มากยิ่งขึ้น

4. รู้ปัญหา และแนวทางในการพัฒนาระบบขนส่งในอุตสาหกรรมเพื่อให้สอดคล้องกับการใช้งานจริง

## 1.6 แผนการดำเนินโครงงาน

**ตารางที่ 1.1** แผนการดำเนินโครงงานระหว่างเดือนสิงหาคม 2559 ถึงเดือนเมษายน 2560

| ขั้นตอนการดำเนินโครงงาน | 2559 | | | | | 2560 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | ส.ค. | ก.ย. | ต.ค. | พ.ย. | ธ.ค. | ม.ค. | ก.พ. | มี.ค. | เม.ย. |
| 1. ศึกษาทฤษฎีการประมวลผลภาพ | ← | — | → | | | | | | |
| 2. ศึกษาการใช้คำสั่งภาษา Python | ← | — | → | | | | | | |
| 3. ศึกษาข้อมูลและทฤษฎีที่เกี่ยวข้อง | ← | — | → | | | | | | |
| 4. ออกแบบและจัดซื้ออุปกรณ์ที่ต้องใช้ | | ← | — | → | | | | | |
| 5. จัดเตรียมและทำการประกอบหุ่นยนต์ | | | ← | — | — | → | | | |
| 6. ทดสอบการจับสีของกล้อง | | | | ← | — | → | | | |
| 7. เขียนโปรแกรมและทำการทดสอบ | | | | ← | — | — | — | → | |
| 8. ตรวจและแก้ไขปัญหา | | | | | | ← | — | — | → |
| 9. สรุปผลการดำเนินงาน | | | | | | | | ← | — → |

# บทที่ 2
# หลักการและทฤษฎีที่เกี่ยวข้อง

## 2.1 การเขียนโปรแกรม

### 2.1.1 OpenCV

OpenCV เป็น Library ในภาษา C++ และPython สำหรับการพัฒนาโปรแกรมที่เกี่ยวข้องกับ Image Processing และComputer Vision โดยสามารถพัฒนาได้ทั้งในระบบปฏิบัติการ Window และระบบปฏิบัติการ Linux

Colors หรือ Object จะประกอบไปด้วย Histograms ของสีต่างๆ จะเก็บลักษณะของ Histograms เอาไว้เช็คข้อมูลที่อยู่ในภาพที่เปลี่ยนแปลงอยู่ตลอดเวลา หากมีลักษณะที่ใกล้เคียงก็สามารถที่จะระบุตำแหน่งของ Object หรือ Color

### 2.1.2 Python

Python คือ ชื่อภาษาที่ใช้ในการเขียนโปรแกรมภาษาหนึ่ง ซึ่งถูกพัฒนาขึ้นมาโดยไม่ยึดติดกับแพลตฟอร์ม กล่าวคือสามารถรันภาษา Python ได้ทั้งบนระบบ Unix, Linux, Windows NT, Windows 2000, Windows XP หรือแม้แต่ระบบ FreeBSD อีกอย่างหนึ่งภาษาตัวนี้เป็น Open Source เหมือนอย่าง PHP ทำให้ทุกคนสามารถที่จะนำ Python มาพัฒนาโปรแกรมได้แบบฟรีๆ โดยไม่ต้องเสียค่าใช้จ่าย และความเป็น Open Source ทำให้มีคนเข้ามาช่วยกันพัฒนา Python ให้มีความสามารถสูงขึ้น และใช้งานได้ครอบคลุมกับทุกลักษณะงาน

โค้ดของ Python ถูกสร้างขึ้นมาจากภาษา C การประมวลผลจะทำในแบบอินเทอร์พรีเตอร์ คือจะประมวลผลไปทีละบรรทัด และปฏิบัติตามคำสั่งที่ได้รับ Python เวอร์ชั่นแรกคือ เวอร์ชั่น 0.9.0 ออกมาเมื่อปี 2533 และเวอร์ชั่นปัจจุบันคือ 2.5.2

**คุณลักษณะเด่นของภาษา Python**

1. สนับสนุนแนวคิดแบบ OOP (Object Oriented Programming)
2. เป็น Open Source
3. โค้ดที่เขียนด้วย Python สามารถนำไปรันบนระบบปฏิบัติการได้หลากหลาย
4. สนับสนุนเทคโนโลยี COM ของ Ms-windows
5. Python รวมมาตรฐานการอินเตอร์เฟส Tkinter ซึ่งสนับสนุนบนระบบ X Windows, Ms-windows และ Macintosh การใช้คำสั่ง Tkinter API ช่วยให้โปรแกรมเมอร์ไม่ต้องแก้ไขโค้ดเมื่อนำไปรันบนระบบปฏิบัติการอื่นๆ
6. เป็น Dynamic Typing คือ สามารถเปลี่ยนชนิดข้อมูลได้ง่ายและสะดวก
7. มี Build-in Object Types คือ โครงสร้างของข้อมูลที่สามารถใช้ได้ในภาษา Python ประกอบด้วย ลิสต์, ดิกชันนารี, สตริง ที่ง่ายต่อการใช้งาน และมีประสิทธิภาพสูง
8. มีเครื่องมือต่างๆ มากมาย เช่น การประมวลผลเท็กซ์ไฟล์ การเรียงข้อมูล การเชื่อมต่อสตริง การตรวจสอบเงื่อนไขของข้อความ การแทนคำ เป็นต้น

9. มีโมดูลสำหรับจัดการ Regular Expression

10. มีโมดูลที่สร้างขึ้นจากนักพัฒนาสนับสนุนมากมาย ได้แก่ COM, Image, CORBA, ORBs, XML เป็นต้น

11. จัดการหน่วยความจำอย่างอัตโนมัติ สามารถจัดการพื้นที่หน่วยความจำที่ไม่ต่อเนื่องให้ทำงานได้อย่างมีประสิทธิภาพ

12. อนุญาตให้ฝังชุดคำสั่งของ Python เอาไว้ภายในโค้ดภาษา C/C++ ได้

13. อนุญาตให้โปรแกรมเมอร์สร้าง Dynamic Link Library (DLL) เพื่อใช้ร่วมกับ Python

14. มีโมดูลสนับสนุนเกี่ยวกับเน็ตเวิร์ก โปรเซส เธรด Regular, Expression, Xml, GUI และอื่นๆ

15. ประกอบด้วยโมดูลสำหรับสร้าง Internet Script และติดต่อกับอินเทอร์เน็ตผ่าน Sockets, และทำหน้าที่เป็น CGI Script ตลอดจนใช้งานคำสั่ง FTP, Glopher, XML และอื่นๆ อีกมากมาย

16. สามารถประมวลผลทางด้านวิทยาศาสตร์ และวิศวกรรมศาสตร์ได้อย่างมีประสิทธิภาพ

17. มีฟังก์ชันสนับสนุนฐานข้อมูล เช่น MySQL, Sybase, Oracle, Informix, ODBC และอื่นๆ

18. มีไลบรารีสนับสนุนด้านการสร้างภาพกราฟฟิก เช่น ทำภาพเบลอ ทำภาพชัด หรือเขียนข้อความบนภาพ ตลอดจนบันทึกไฟล์ในรูปแบบต่างๆ ได้อย่างสะดวกและมีประสิทธิภาพ

19. มีไลบรารีสนับสนุนด้านปัญญาประดิษฐ์

20. มีไลบรารีสำหรับสร้างเอกสาร PDF โดยไม่ต้องติดตั้ง Acrobat Writer

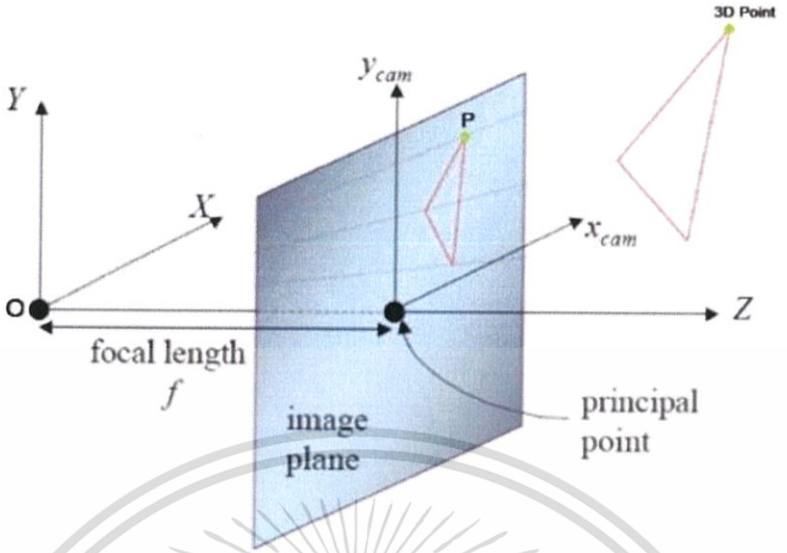
21. มีไลบรารีสำหรับสร้าง Shockwaves Flash (SWF) โดยไม่ต้องติดตั้ง Macromedia Flash

## 2.2 กล้อง

### 2.2.1 แบบจำลองของกล้อง (Camera Model)

แบบจำลองของกล้องที่นิยมใช้กันโดยทั่วไปจะเป็นแบบจำลองกล้องรูเข็ม (Pin-Hole Camera Model) โดยประกอบด้วยจุดศูนย์กลางการฉาย O (Center of Projection), ระนาบรับภาพ P (Image Plane), จุดพิกัดใน 3 มิติ (3 D Point) และจุดบนระนาบรับภาพ ดังรูปที่ 1.2 และรูปที่ 2.2


**รูปที่ 2.1** กล้องเว็ปแคม




**รูปที่ 2.2** แบบจำลองกล้องรูเข็ม

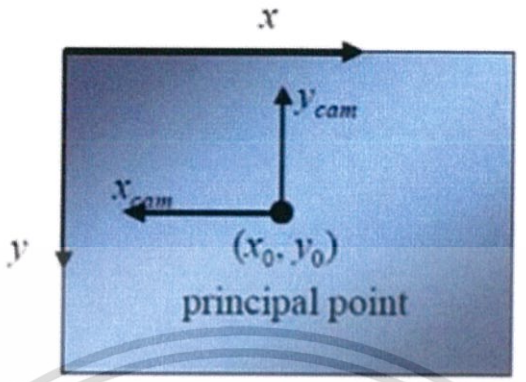
โดยการประมาณพิกัดของจุดบนระนาบรับภาพ P สามารถคำนวณได้จากหลักการของสามเหลี่ยมคล้ายดังสมการที่ (2.1)

$$x_{cam} = f\frac{x}{z}$$
$$y_{cam} = f\frac{y}{z} \qquad (2.1)$$

โดย Xcam, Ycam เป็นพิกัดของระนาบภาพในแนวแกน X และแกน Y, f คือ ความยาวโฟกัส ซึ่งสามารถเขียนจุดพิกัดใน 3 มิติที่ถูกแปลงเป็น 2 มิติบนระนาบภาพได้ใหม่ในรูปแบบของพิกัดเอกพันธุ์ (Homogeneous Coordinate) ซึ่งการแปลงพิกัดด้วยผลการแปลงเชิงเส้นนี้จะช่วยแก้ปัญหาการที่ระบบพิกัดของโลกภายนอกกับตัวกล้องมักไม่สอดคล้องกันโดยตรงดังสมการที่ (2.2)

$$\begin{bmatrix} x_{cam} \\ y_{cam} \\ f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad (2.2)$$

เมื่อได้พิกัดของจุดบนระนาบภาพแล้ว แต่เนื่องจากแกนของภาพทั่วไปกับแกนของพิกัดของจุดบนระนาบภาพมีความแตกต่างกันดังรูปที่ 2.3




**รูปที่ 2.3** ความแตกต่างระหว่างตำแหน่งแกนของพิกัดจุดบนระนาบกับพิกัดบนภาพทั่วไป

จึงต้องมีการปรับเพื่อให้แกนทั้งภาพตรงกันดังสมการที่ (2.3)

$$x = -k_x x_{cam} + x_0 \qquad (2.3)$$
$$x = -k_x x_{cam} + x_0$$

โดยกำหนดให้ $\alpha x = -fkx$ และ $\alpha y = -fky$ ซึ่งสมการข้างต้น สามารถแปลงเป็นพิกัดเอกพันธุ์ได้ดังสมการที่ (2.4)

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \frac{1}{f}\begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x_{cam} \\ y_{cam} \\ f \end{bmatrix}$$
$$\begin{bmatrix} x_{cam} \\ y_{cam} \\ f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} \qquad (2.4)$$
$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} R^T & -t \\ 0^T & 1 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

ซึ่งสมการที่ได้มาเป็นสมการที่ใช้สำหรับการหาพิกัดของจุดบนระนาบภาพ และสมการอันถัดลงมาเป็นสมการการหาพิกัด 3 มิติของกล้อง โดยการนำตำแหน่ง 3 มิติของวัตถุจริงคูณกับค่าการหมุน (R) และการเคลื่อนที่ของตำแหน่งพิกัด (T) เพื่อความเข้าใจที่ง่ายขึ้น สามารถจัดรูปได้ดังสมการที่ (2.5)

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R^T | -t \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$x = K \begin{bmatrix} R^T | -t \end{bmatrix} X$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R^T & -t \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (2.5)$$

$$K = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

จากสมการด้านบน K เป็นเมทริกซ์การสอบเทียบกล้อง (Camera Calibration Matrix) ขนาด 3x3 ที่มีความสัมพันธ์โดยตรงกับพารามิเตอร์ภายในทั้ง 4 ตัวของกล้อง โดย $\alpha x$, $\alpha y$ เป็นค่าคงที่ซึ่งเกี่ยวข้องกับคุณสมบัติของเลนส์ และความละเอียดของอุปกรณ์รับรู้ภาพ x0, y0 เป็นตำแหน่งจุดศูนย์กลางของภาพซึ่งคือจุดกำเนิดของแกนอ้างอิงภาพ และเป็นจุดตัดของเลนส์มุขสำคัญของเลนส์ (Optical Axis) กับระนาบรับภาพ R T |-t เป็นพารามิเตอร์ภายนอกมีขนาด 3x4 โดยแบ่งเป็นส่วนของการหมุน และส่วนของการเลื่อนที่ซึ่งมีองศาอิสระเท่ากับ 6 (องศาอิสระการเลื่อนเท่ากับ 3 และองศาอิสระการหมุนเท่ากับ 3) เมทริกซ์กล้องจะมีองศาอิสระรวม 11 จะสอดคล้อง กับการที่เมทริกซ์กล้องมีสมาชิก 12 ตัวซึ่งมีค่าขึ้นอยู่กับสเกล (Up To Scale) ดังนั้นองศาจะเหลือเพียง 11 เท่านั้นและ X เป็นพิกัด 3 มิติ ซึ่งจากสมการนี้สามารถนำมาใช้สำหรับการปรับปรุง และแก้ปัญหาความผิดเพี้ยนของกล้องที่เกิดจากการ Misalignment ของตัวเซนเซอร์กับศูนย์กลางของเลนส์ และความผิดเพี้ยนของตัวเลนส์เอง ในขั้นตอนของการผลิต และการประกอบกล้อง ซึ่งสามารถแบ่งความผิดเพี้ยนของเลนส์เป็น 3 ชนิดดังนี้

1. ความผิดเพี้ยนในแนวรัศมี (Radial Distortion) เกิดจากความบกพร่องของเลนส์ทำให้จุดภาพมีการเคลื่อนที่เข้าหรือออกมาข้างนอกจากตำแหน่งที่ถูกต้อง ซึ่งสามารถแสดงให้อยู่ในรูปสมการ ดังสมการที่ (2.6)

$$x_{rd} = a_1 x_d p^2 + a_2 x_d p^4$$
$$y_{rd} = a_1 y_d p^2 + a_2 y_d p^4 \qquad (2.6)$$
$$p = \sqrt{(X_{d^2} + Y_{d^2})}$$

เมื่อ $x_{rd}, y_{rd}$ คือ ความผิดเพี้ยนในแนวรัศมีตามแนวแกน x และ y
$a_1, a_2$ คือ สัมประสิทธิ์ความผิดเพี้ยน
$X_d, Y_d$ คือ พิกัดของจุดภาพบนระนาบภาพในแนวแกน x และ y

2. ความผิดเพี้ยนไม่ตรงศูนย์กลาง (Decentering Distortion) เกิดจากศูนย์กลางทางเดินแสงของเลนส์ไม่ได้อยู่ที่จุดเดียวกัน ดังสมการที่ (2.7)

$$x_{dd} = 2b_1 X_d Y_d + b_2 (3X_{d^2} + Y_{d^2})$$
$$y_{dd} = b_1 (X_{d^2} + 3Y_{d^2}) + 2b_2 X_d Y_d \qquad (2.7)$$

เมื่อ $x_{dd}, y_{dd}$ คือ ความผิดเพี้ยนไม่ตรงศูนย์กลางในแนวแกน x และ y
$b_1, b_2$ คือ สัมประสิทธิ์ความผิดเพี้ยน

3. ความผิดเพี้ยนจากความหนาของเลนส์ (Thin Prism Distortion) ดังสมการที่ (2.8)

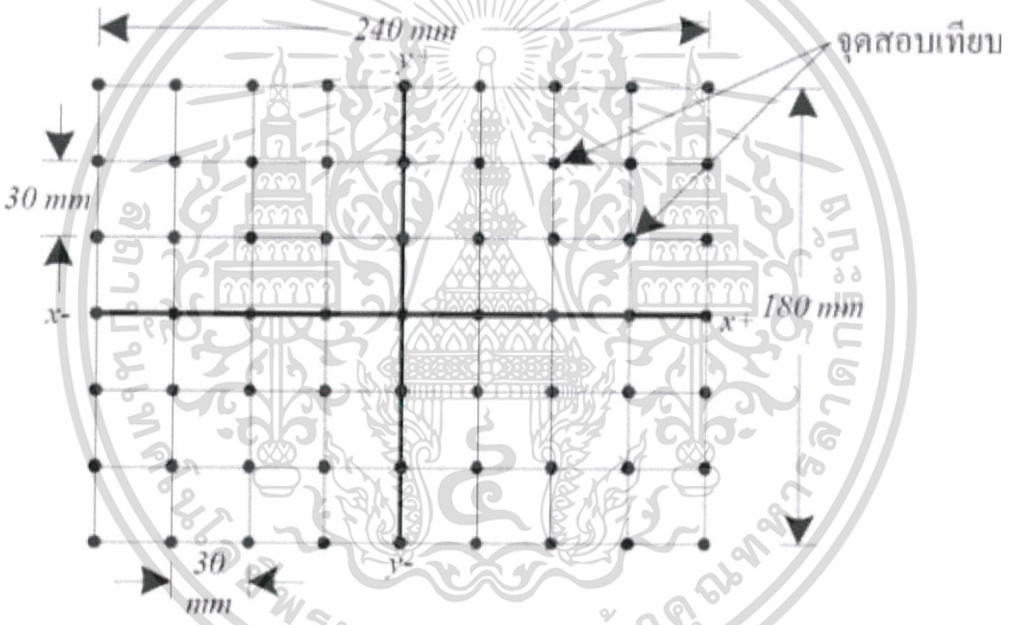
$$x_{pd} = c_1 p^2$$
$$y_{pd} = c_2 p^2 \qquad (2.8)$$

เมื่อ $x_{pd}, y_{pd}$ คือ ความผิดเพี้ยนจากความหนาของเลนส์ในแนวแกน x และ y
$c_1, c_2$ คือ สัมประสิทธิ์ความผิดเพี้ยน

ดังนั้นจึงนำไปสู่กระบวนการสอบเทียบกล้อง เพื่อจะแก้ไขปัญหาความผิดเพี้ยนของกล้องด้วยการคำนวณภาพใหม่ที่มีความถูกต้องมากขึ้น

**การสอบเทียบกล้อง (Camera Calibration)**

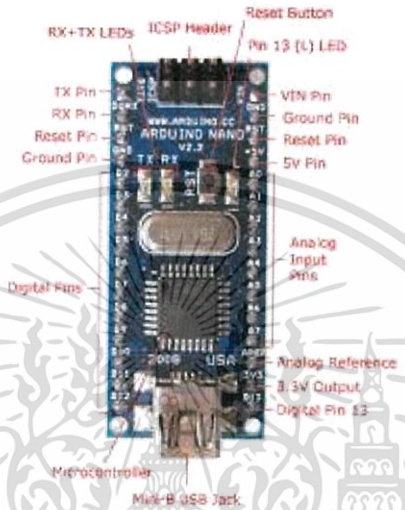
ในการสอบเทียบสิ่งที่ต้องการหาคือ พารามิเตอร์ภายใน และภายนอกของกล้อง การสอบเทียบทำได้โดยการถ่ายภาพ Calibration Chart ดังรูปที่ 2.4 เนื่องจากจุดสอบเทียบทุกจุดบน Calibration Chart มีระยะห่างทั้งแนวตั้ง และแนวนอนเท่ากับทุกๆ จุดเมื่อใช้กล้องที่ต้องการสอบเทียบถ่ายภาพ Calibration Chart จะได้ระยะห่างระหว่างจุดสอบเทียบทั้งแนวนอน และแนวตั้งใกล้เคียงหรือเท่ากับที่บริเวณจุดศูนย์กลางของภาพ แต่จะต่างกันมากขึ้นเมื่อจุดสอบเทียบอยู่ที่เส้นรอบวงภายนอก ด้วยเหตุนี้จะสมมติได้ว่าระยะทางระหว่างจุดสอบเทียบที่อยู่ใกล้บริเวณศูนย์กลางของภาพเป็นค่าที่ถูกต้อง และนำค่าระยะทางระหว่างจุดสอบเทียบที่นำมาคำนวณหาพิกัดของจุดสอบเทียบอื่นๆ ได้ ซึ่งจุดพิกัดของจุดสอบเทียบที่คำนวณได้นี้เป็นพิกัดภาพของจุดสอบเทียบบนระนาบภาพที่ไม่มีความผิดพลาด พิกัดของจุดสอบเทียบเหล่านี้จะถูกนำมาใช้เพื่อคำนวณหาค่า Element ของเมทริกซ์การหมุน Element ของ Translation Vector และความยาวโฟกัส f




**รูปที่ 2.4** Calibration Chart

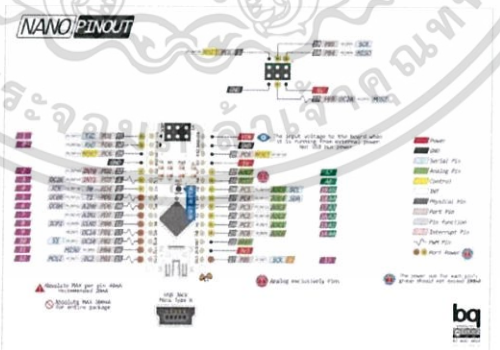
## 2.3 Microcontroller (Arduino)

Arduino คือ บอร์ดไมโครคอนโทรลเลอร์สำเร็จรูปที่รวมเอาตัวไมโครคอนโทรลเลอร์ และอุปกรณ์อื่นๆ ที่จำเป็นมาในบอร์ดเดียว มีการเปิดเผยข้อมูลทุกๆ อย่าง ทั้งลายวงจรและตัวอย่างโปรแกรม ทำให้ผู้ใช้สามารถนำไปพัฒนาต่อได้ เพียงแค่มีบอร์ด Arduino กับคอมพิวเตอร์ ดังรูปที่ 2.5




**รูปที่ 2.5** โครงสร้างภายนอกของ Arduino Nano

Arduino Nano มีขนาดเพียง 1.8 x 4.8 เซนติเมตร ซึ่งถือว่ามีขนาดเล็กมาก เมื่อเทียบกับบอร์ดไมโครคอนโทรลเลอร์อื่นบนบอร์ด Arduino Nano นั้นมีวงจรสำหรับปรับแรงดันไฟฟ้าให้เหมาะสมเป็น Microcontroller ดังรูปที่ 2.6

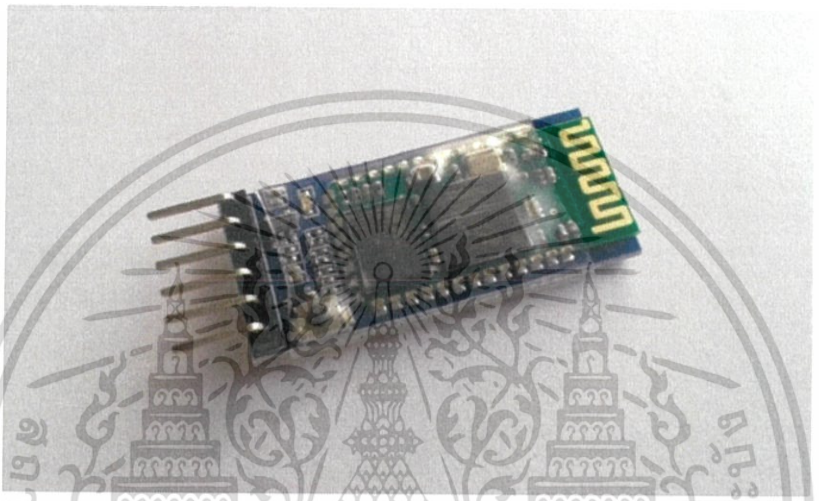



**รูปที่ 2.6** ภาพรายละเอียด Arduino Nano 3.0 ATMEGA328P

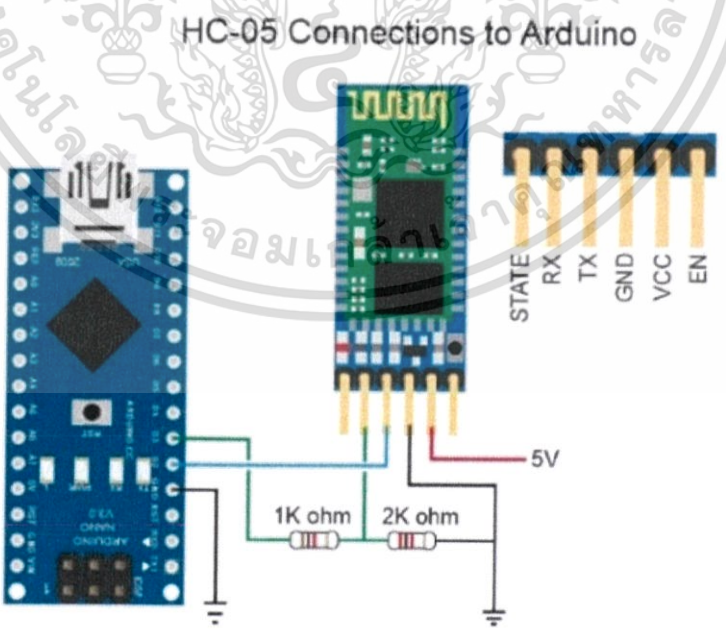
Board (MCU) ที่ใช้ ATmega328p เป็น MCU หลักที่รวบรวมอุปกรณ์สนับสนุนการทำงานของ CPU ไว้มากมาย อาทิเช่น Analog to Digital, SPI, UART, Timer, Counter และ PWM ซึ่งอุปกรณ์สนับสนุนการทำงานเหล่านี้ทำให้ MCU สามารถทำงานได้กว้างขวาง

## 2.4 Module Bluetooth HC-05

HC05 เป็นโมดูล Bluetooth ที่ใช้งานในการเชื่อมต่อกับสมาร์ทดีไวซ์ต่างๆ ให้สมาร์ทดีไวซ์สามารถสื่อสารกับไมโครคอนโทรลเลอร์ (Arduino AVR PIC etc.) ได้ ผ่าน Serial Port โมดูลรุ่น HC05 สามารถตั้งให้ใช้งานได้ทั้งโหมด Master (ให้อุปกรณ์อื่นมาเชื่อมต่อ) และโหมด Slave (เชื่อมต่อกับอุปกรณ์อื่น) การตั้งค่าต่างๆ เช่น ชื่ออุปกรณ์ รหัสผ่าน ทำได้ผ่าน AT Command ซึ่งจะต้องมีการต่อขาพิเศษเพื่อให้โมดูลเข้าโหมดการตั้งค่า หรือกดปุ่มบนโมดูลค้างไว้ ดังรูปที่ 2.7 และรูปที่ 2.8



**รูปที่ 2.7** ภาพ Module Bluetooth HC-05




**รูปที่ 2.8** ภาพ Module Bluetooth HC-05 ต่อกับ Arduino Nano

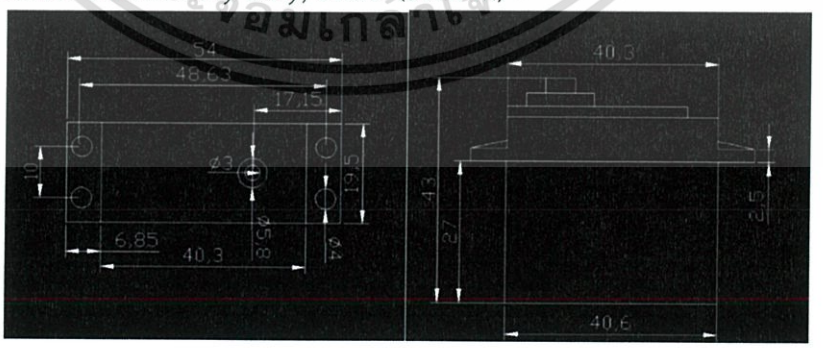
## 2.5 High Torque Servo Motor MG995 Tower Pro (360 Degree)

Servo Motor ตัวนี้หมุนได้ต่อเนื่อง 360 องศา ควบคุมความเร็วรอบได้ทั้งทวนเข็ม และตามเข็มนาฬิกา โดยใช้สัญญาณควบคุมจากไมโครคอนโทรลเลอร์ ให้ทอร์กสูง เฟืองทำจากทองเหลืองทำให้มีความทนทานต่อการใช้งาน ดังรูปที่ 2.9

**รูปที่ 2.9** ภาพ High Torque Servo Motor MG995 Tower Pro (360 Degree)

**Specification**

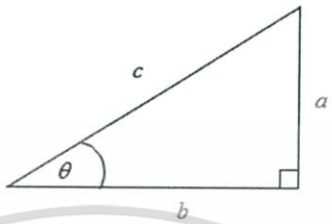
- Dimension : 40mm x 19mm x 43mm
- Weight : 55g
- Operating Speed : 0.17sec /60 degrees (4.8V no load)
- Operating Speed : 0.13sec /60 degrees (6.0V no load)
- Stall Torque : 9 kg-cm (180.5 oz.-in) at 4.8V
- Stall Torque : 12 kg-cm (208.3 oz.-in) at 6V
- Operation Voltage : 4.8-7.2Volts
- Gear Type : All Metal Gears
- Original Box : NO
- Color : Black
- Connector Wire : Heavy Duty, 11.81" (300mm)




**รูปที่ 2.10** ภาพ High Torque Servo Motor MG995 Design

## 2.6 ตรีโกณมิติของรูปสามเหลี่ยมมุมฉาก

มุมมองแบบที่สองของฟังก์ชันตรีโกณมิติคือ มุมมองจากรูปสามเหลี่ยมมุมฉาก พิจารณารูปสามเหลี่ยมมุมฉากรูปหนึ่งที่มีมุมแหลมมุมหนึ่งเป็น $\theta$ ดังรูปที่ 2.11 ด้านของรูปสามเหลี่ยมที่เกี่ยวข้องกับมุม $\theta$ ได้แก่ ด้านตรงข้ามมุมฉาก ด้านตรงข้ามมุม $\theta$ และด้านประชิดมุม $\theta$




รูปที่ 2.11 ภาพสามเหลี่ยมมุมฉาก

ด้วยการใช้ความยาวของด้านทั้งสามนี้ สามารถสร้างฟังก์ชันตรีโกณมิติทั้งหกฟังก์ชันของมุมแหลม $\theta$ ได้ ดังนิยามต่อไปนี้

$$\sin\theta = \frac{a}{c} \text{ (ข้าม/ฉาก)} \qquad \cos ec\,\theta = \frac{1}{\sin\theta} = \frac{c}{a}$$

$$\cos\theta = \frac{b}{c} \text{ (ชิด/ฉาก)} \qquad \sec\theta = \frac{1}{\cos\theta} = \frac{c}{b}$$

$$\tan\theta = \frac{a}{b} \text{ (ข้าม/ชิด)} = \frac{\sin\theta}{\cos\theta} \qquad \cot\theta = \frac{1}{\tan\theta} = \frac{b}{a} = \frac{\cos\theta}{\sin\theta}$$

# บทที่ 3
# หลักการการออกแบบ และการเขียนโปรแกรม

ในการจัดทำโครงงานนี้นอกจากจะอาศัยทฤษฎีข้างต้นดังที่ได้กล่าวมาแล้ว ยังต้องอาศัยการเขียนโปรแกรมด้วยภาษา Python ในการควบคุมการเคลื่อนที่ของหุ่นยนต์ให้ทำงานได้ตรงตามเป้าหมาย ซึ่งรายละเอียดขั้นตอนทั้งหมดมีดังนี้

## 3.1 การออกแบบหุ่นยนต์

ขั้นตอนแรกของการทำโครงงานคือ การประกอบหุ่นยนต์สองล้อขึ้นมาเพื่อใช้เป็นต้นแบบหรือแบบจำลองของรถส่งของในอุตสาหกรรม โดยการประกอบหุ่นยนต์มีขั้นตอนดังนี้

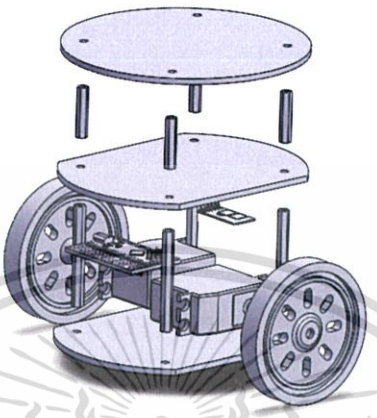
### 3.1.1 อุปกรณ์ที่ต้องใช้สำหรับการประกอบหุ่นยนต์

อุปกรณ์ที่ใช้ประกอบหุ่นยนต์ 2 ล้อมีดังต่อไปนี้

1. แผ่นอะคริลิก หนา 3 มม.
2. ไมโครคอนโทรลเลอร์ Arduino Nano 3.0
3. High Torque Servo Motor MG-995
4. Bluetooth Module HC-05
5. Buck Converter Module
6. ล้อ Servo ขนาดเส้นผ่าศูนย์กลาง 70 มม.
7. Power Bank (Battery)
8. Sticker สีแดง และสีน้ำเงิน

### 3.2.2 ขั้นตอนการประกอบหุ่นยนต์

1. ออกแบบลักษณะ และกำหนดขนาดของหุ่นยนต์ โดยใช้โปรแกรม CAD ในการออกแบบ ดังรูปที่ 3.1



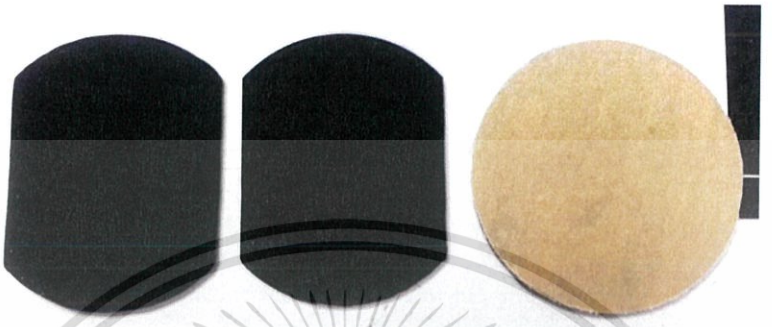
**รูปที่ 3.1** ออกแบบหุ่นยนต์ด้วยโปรแกรม SolidWorks

**รูปที่ 3.2** การออกแบบหุ่นยนต์

เนื่องจากโครงงานนี้จะใช้กล้องประมวลผลภาพเพียงตัวเดียว ดังนั้นขนาดของหุ่นยนต์จึงมีขนาดเล็กเพียง 12x12x9 ซม. โดยออกแบบให้มีทั้งหมด 2 ชั้น โดยชั้นที่ 1 จะเป็นส่วนที่ตั้งของ Servo Motor 2 ตัว คือ Arduino Nano และ Bluetooth ส่วนชั้นที่ 2 จะเป็นส่วนที่เอาไว้สำหรับวาง Power bank เพื่อเพิ่มความง่ายในการจัดระเบียบของหุ่นยนต์ อีกทั้งยังมีหลังคาที่ติดเทปกาวสีเพื่อให้กล้องประมวลผลจากสีที่ถ่ายไว้ ดังรูปที่ 3.2

2. นำแผ่นอะคริลิกหนา 3 มม. มาตัดตามขนาดที่ออกแบบไว้ ซึ่งมีลักษณะดังรูปที่ 3.3 โดยขนาดดังนี้

- อะคริลิกขนาดรัศมี 60 มม. 1 ชิ้น
- อะคริลิกขนาดรัศมี 60 มม. พร้อมตัดข้าง 2 ชิ้น



**รูปที่ 3.3** ตัดแผ่นอะคริลิก

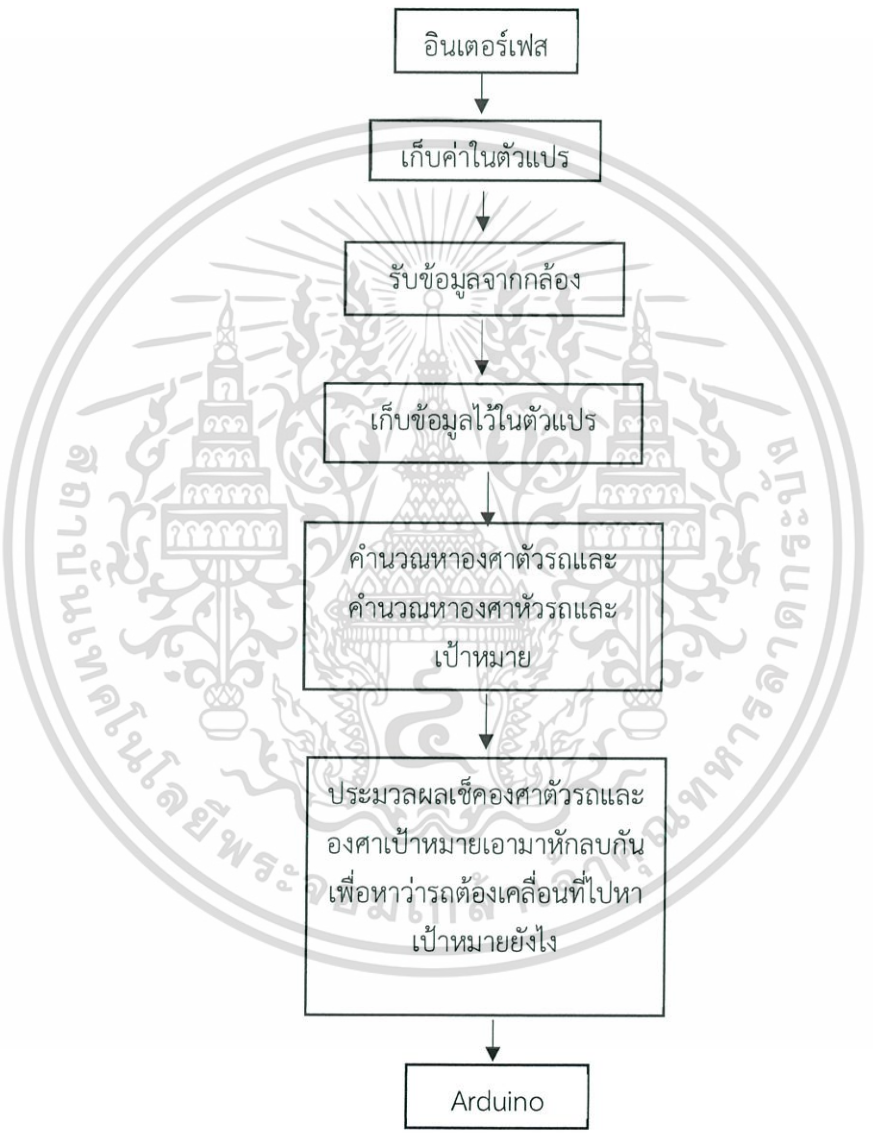
จากนั้นเจาะรูบนอะคริลิกเพื่อค้ำ และประกอบทั้งสามชิ้นเข้าด้วยกันเป็นชั้นดังที่ออกแบบไว้

3. จัดวาง Servo Motor ตามตำแหน่งที่ต้องการ และประกอบให้เข้ากับล้อ โดยแกนแนวของทั้งสองล้อจะต้องตรงกันเพื่อความสมดุลในการเคลื่อนที่ของหุ่นยนต์ จากนั้นเชื่อมต่อ Arduino Nano, Servo และ Bluetooth Module เข้าด้วยกัน ดังรูปที่ 3.4

**รูปที่ 3.4** การประกอบหุ่นยนต์

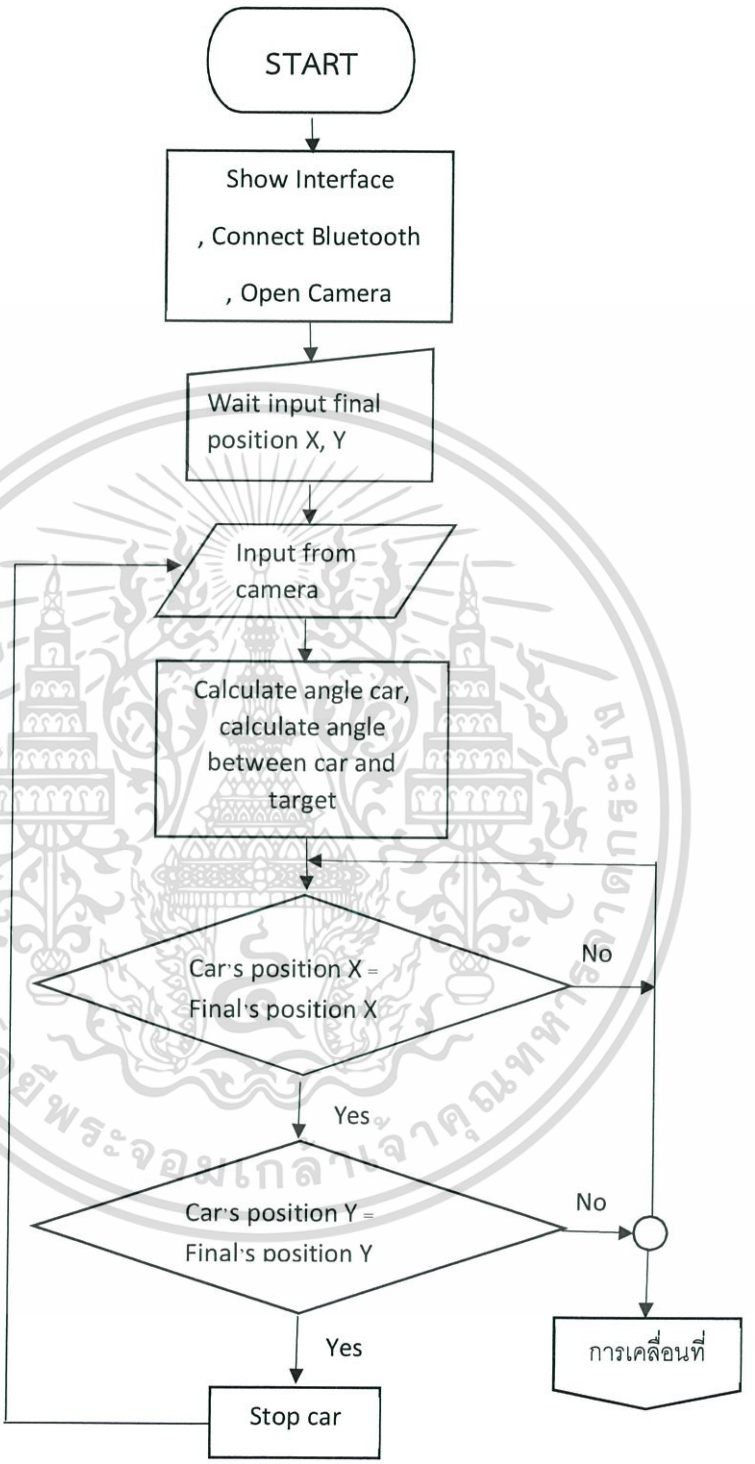
## 3.2 หลักการการควบคุมหุ่นยนต์ด้วยประมวลผลภาพ

หลักการเขียนโปรแกรมการควบคุมหุ่นยนต์ด้วยการประมวลผลภาพมีหลักการทำงานใหญ่ๆ เป็นสามขั้นตอนคือ การสร้างอินเตอร์เฟสไว้รอรับค่า สร้างตัวแปรไว้เก็บค่า ส่งต่อไปยังโปรแกรมย่อย เพื่อประมวลผล และส่งข้อมูลผ่าน Bluetooth ไปยังโปรแกรมใน Arduino ที่ทำการควบคุมรถ ดังรูปที่ 3.5

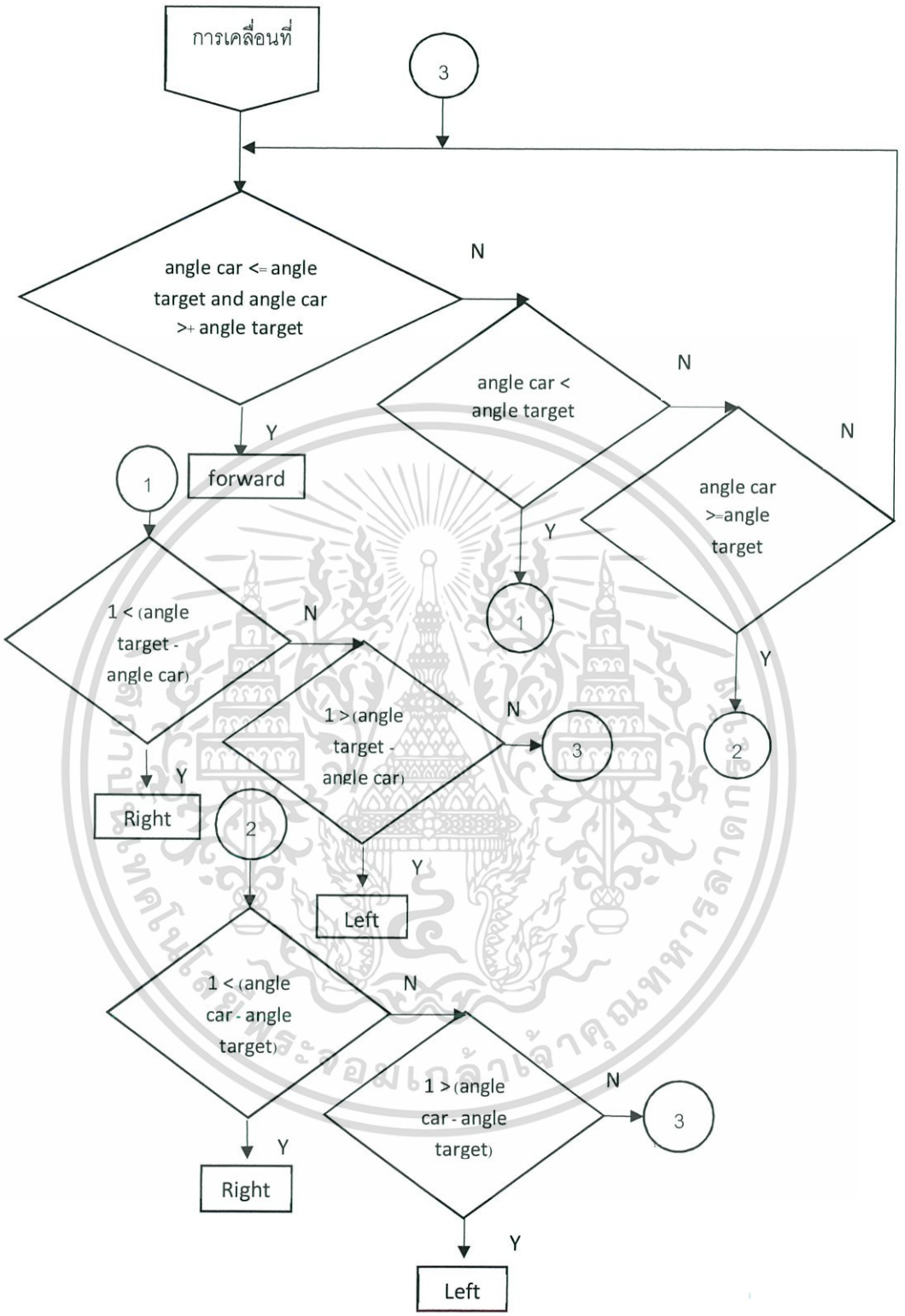



**รูปที่ 3.5** ลำดับขั้นตอนการทำงานโปรแกรม

Flowchart โปรแกรมควบคุมหุ่นยนต์ด้วยการประมวลผลภาพ




**รูปที่ 3.6** รูป Flowchart โปรแกรมควบคุมหุ่นยนต์ด้วยการประมวลผลภาพตอนที่ 1




**รูปที่ 3.7** รูป Flowchart โปรแกรมควบคุมหุ่นยนต์ด้วยการประมวลผลภาพตอนที่ 2

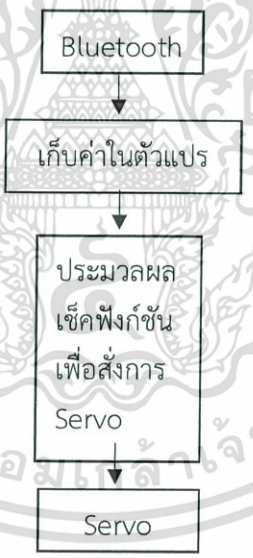
## 3.3 เขียนโปรแกรมควบคุมหุ่นยนต์

การควบคุมการทำงานของหุ่นยนต์ใช้ Microprocessor เช่น Arduino เป็นตัวควบคุม โดยจะสามารถเขียนโปรแกรมควบคุมทิศทางการวิ่งของหุ่นยนต์ได้ ด้วยการบังคับทิศทางการหมุนของเซอร์โวมอเตอร์ทั้งสองข้างที่ต่อกับล้อของหุ่นยนต์ หากล้อหมุนไปทางเดียวกันหุ่นยนต์จะวิ่งไปข้างหน้าหรือข้างหลัง หากล้อหมุนทิศทางตรงข้ามหรือข้างใดข้างหนึ่งหมุน หุ่นยนต์จะเลี้ยวไปตามทิศทางที่ล้อหมุน โดยสามารถเขียนโปรแกรมได้ดังนี้

ส่วนแรกของโปรแกรมจะเป็นการประกาศตัวแปร และเลือกการจ่ายไฟให้เซอร์โวมอเตอร์จากช่องจ่ายไฟขาออกของ Microprocessor

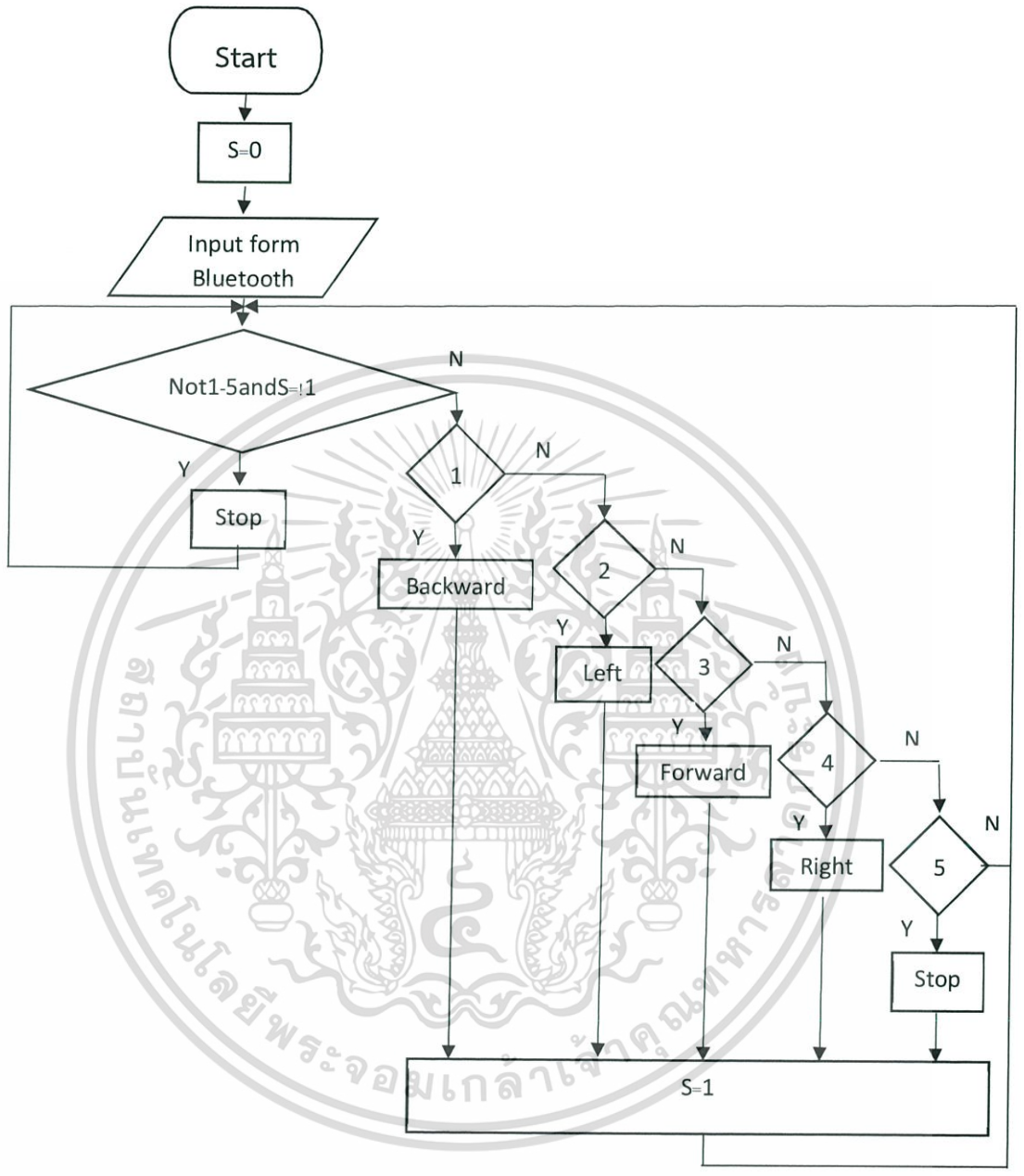
การกำหนดลูปรับค่ามาจาก Bluetooth คำสั่งมาใช้เป็นตัวควบคุมทิศทางการเคลื่อนที่ของหุ่นยนต์ โดยขั้นแรกหุ่นยนต์จะหยุดอยู่กับที่

ค่าคำสั่งที่ได้รับจะเป็นค่า Serial Numbers หากการคำนวณในโปรแกรมส่งค่า 1 มา หุ่นยนต์จะวิ่งถอยหลัง หากส่งค่า 2 มา หุ่นยนต์จะเลี้ยวไปทางขวา หากค่าส่ง 3 มา หุ่นยนต์จะวิ่งตรงไปข้างหน้า และค่า 4 สำหรับเลี้ยวซ้าย เมื่อหุ่นยนต์ถึงจุดหมายปลายทางแล้ว โปรแกรมจะส่งค่า 5 มา ซึ่งหมายถึงการบังคับให้หุ่นยนต์หยุดวิ่ง




**รูปที่ 3.8** ลำดับขั้นตอนการทำงานโปรแกรมที่ตัว Arduino

Flowchart's Car

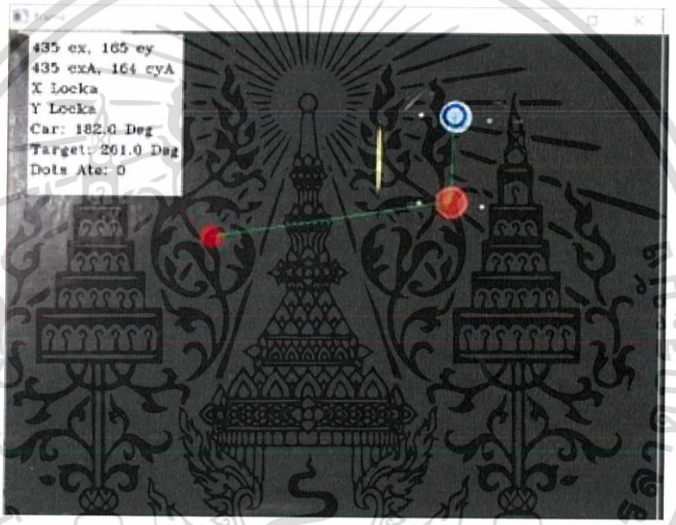



**รูปที่ 3.9** รูป Flowchart's Car

# บทที่ 4
# การทดลองและผลการทดลอง

## 4.1 การประมวลผลภาพและการจับสี

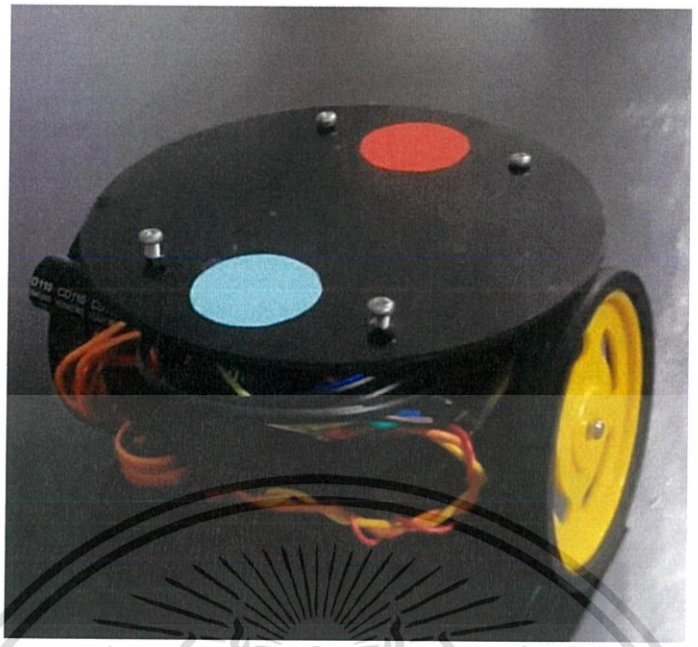
การทดสอบการประมวลผลภาพด้วยกล้อง Webcam OKER พบว่าการเขียนโปรแกรมด้วยภาษา Python และการใช้ OpenCV ในการเขียนคำสั่งประมวลผลภาพ โดยในโครงงานนี้จะใช้ความสามารถในการจับสีมาเป็นตัวช่วยในการระบุตำแหน่ง และควบคุมหุ่นยนต์ โปรแกรมที่เขียนไปจะสามารถตรวจจับช่วงโทนสีแดง และสีน้ำเงินตามที่ได้กำหนดค่าไว้ อีกทั้งยังสามารถระบุพิกัดจุดทั้งสอง และคำนวณการทำมุมระหว่างจุดทั้งสองกับพิกัดเป้าหมายที่ตั้งค่าไว้ ดังรูปที่ 4.1




**รูปที่ 4.1** การประมวลผลภาพจากกล้อง

## 4.2 โครงสร้างหุ่นยนต์

การออกแบบหุ่นยนต์ให้มีสองล้อจะทำให้การเคลื่อนที่ และหมุนตัวสามารถดำเนินการได้ง่าย โดยจะใช้การควบคุมการหมุนของมอเตอร์เซอร์โวทั้งสองตัว หากหมุนไปหน้าหรือหลังพร้อมกันทั้งสองข้าง หุ่นยนต์จะเคลื่อนที่ไปข้างหน้าหรือไปข้างหลังตามลำดับ แต่หากมีข้างใดข้างหนึ่งหยุดจะทำให้เกิดจุดหมุน ทำให้หุ่นยนต์สามารถเลี้ยวซ้ายหรือเลี้ยวขวาได้ตามที่ต้องการ อย่างไรก็ตามหุ่นยนต์สองล้อจำเป็นต้องมีการใช้ล้อลูกกลิ้งในการช่วยประคอง เนื่องจากหุ่นยนต์สองล้อนี้ไม่สามารถประคองตัวเองให้สมดุลได้ โดยจะเอนไปด้านที่ติดอุปกรณ์ที่มีน้ำหนักมากกว่า การติดล้อลูกกลิ้งจะช่วยค้ำหุ่นยนต์ไม่ให้ล้ม ซึ่งจะไม่ก่อให้เกิดข้อผิดพลาดระหว่างการดำเนินงาน



**รูปที่ 4.2** หุ่นยนต์เคลื่อนที่ตามพิกัด

## 4.3 การทดสอบการควบคุมหุ่นยนต์ด้วยระบบไร้สาย Bluetooth

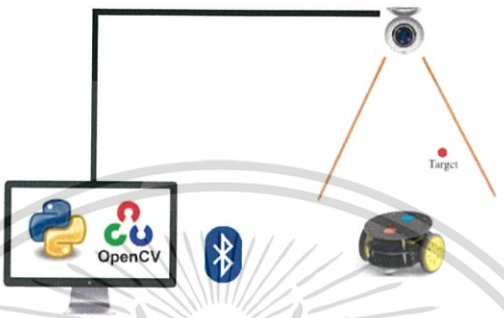
### 4.3.1 การควบคุมโดยระบบ Android ผ่าน Bluetooth

จากรูปที่ 4.3 พบว่าโปรแกรมสำหรับควบคุมมอเตอร์เซอร์โวที่เขียนขึ้นในโปรแกรม Arduino สามารถควบคุมหุ่นยนต์โดยใช้ Smart Phone ที่เป็นระบบ Android ให้เคลื่อนที่ไปในทิศทางที่ต้องการได้ โดยผ่านระบบ Bluetooth ซึ่งการทดลองนี้จะสามารถบ่งบอกประสิทธิภาพของการรับค่าคำสั่งต่างๆ ผ่านระบบไร้สาย ค่าคำสั่งที่ส่งไปจะส่งไปเป็น Serial Numbers โดยแต่ละตัวเลขจะเป็นเหมือนชุดคำสั่งที่มีสำหรับการควบคุมการเคลื่อนที่ และทิศทางของหุ่นยนต์

**รูปที่ 4.3** ควบคุมหุ่นยนต์ระบบไร้สายผ่านมือถือระบบ Android

### 4.3.2 การควบคุมด้วยระบบไร้สายผ่านทาง Computer

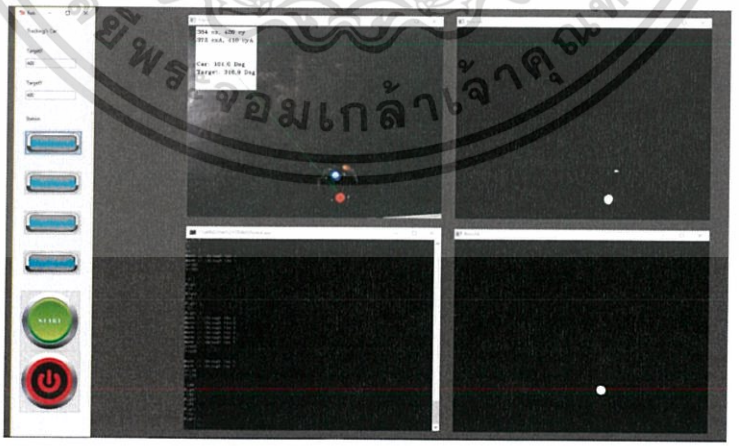
การทดสอบนี้สามารถควบคุมหุ่นยนต์ได้ผ่านโปรแกรมในคอมพิวเตอร์ โดยจะต้องเชื่อมต่อผ่านระบบไร้สาย และส่งค่าเป็น Serial Numbers ซึ่งเป็นคำสั่งในการควบคุมหุ่นยนต์ ค่าที่ได้รับจะถูกนำไปประมวลผลใน Microprocessor ซึ่งได้มีการเขียนโปรแกรมการเคลื่อนที่ของหุ่นยนตไว้แล้ว เมื่อคำสั่งถูกป้อนในคอมพิวเตอร์หุ่นยนต์จะพยายามเคลื่อนที่ไปตามจุดที่ต้องการได้ดังรูปที่ 4.4




รูปที่ 4.4 การรับค่าผ่านระบบไร้สาย

## 4.4 การสร้างแผงควบคุมสำหรับการระบุพิกัดเป้าหมาย

การสร้างเครื่องหมายหรือสัญลักษณ์ที่ทำหน้าที่เชื่อมประสานระหว่างผู้สั่งการ และหุ่นยนต์ ในโครงงานนี้สร้างขึ้นมาเพื่อระบุพิกัดเป้าหมายที่ต้องการให้หุ่นยนต์เดินทางไปถึง โดยจะตั้งค่าพิกัดบนระนาบ X-Y ที่แตกต่างไว้ตามปุ่มต่างๆ หากกดปุ่มเหล่านั้น จุดพิกัดจะที่ตั้งค่าไว้จะกลายเป็นจุดหมายปลายทางที่หุ่นยนต์จะต้องเดินทางไป ดังรูปที่ 4.4 นอกจากปุ่มที่บันทึกค่าตำแหน่งอย่างถาวรแล้ว ยังสามารถกรอกค่าพิกัด X และ Y ลงไปได้ อย่างไรก็ตามค่าพิกัดที่สามารถระบุได้จะต้องอยู่ในขอบเขตระนาบที่กล้องสามารถบันทึกได้ หากเกินไปกว่านั้นระบบจะไม่ทำงาน



**รูปที่ 4.5** การใช้แผงควบคุมกำหนดพิกัดเป้าหมาย

# บทที่ 5
# สรุปและวิจารณ์ผลการทดลอง

## 5.1 สรุปผลการทดลอง

จากการทดสอบการวิ่งตามพิกัดของหุ่นยนต์ พบว่าโปรแกรมสามารถจับสีน้ำเงิน และสีแดงได้โดยจะระบุพิกัด และจุดศูนย์กลางของทั้งสองสีได้ดี โปรแกรมสามารถคำนวณหามุม และระยะห่างของหุ่นยนต์ที่ทำต่อพิกัดเป้าหมาย ซึ่งพิกัดเป้าหมายสามารถกำหนดลงบนแผงควบคุมอินเตอร์เฟส (Interface) ที่เขียนโดยภาษา Python จากการทดสอบการวิ่งของหุ่นยนต์พบว่า หุ่นยนต์สามารถวิ่งไปถึงจุดพิกัดได้อย่างแม่นยำ โดยหุ่นยนต์จะหยุดก็ต่อเมื่อจุดสีแดงที่ตำแหน่งหน้ารถ ไปถึงจุดพิกัดที่กำหนดไว้แล้ว

## 5.2 ปัญหาและแนวทางแก้ไขปัญหา

### 5.2.1 ปัญหาที่พบ

จากการทำโครงงานพบว่า ปัญหาหลักๆ ที่พบเจอคือ ปัญหาเกี่ยวข้องกับการใช้กล้องในการทำหน้าที่จับสี เนื่องจากสีที่ทำการจับจากวัตถุนั้น ไม่สามารถทำการกำหนดตายตัวให้เป็นสีใดสีหนึ่งได้ เนื่องจากสภาพแวดล้อมมีผลต่อแสงที่ส่องลงไปยังวัตถุนั้นๆ จึงทำให้สีที่ทำการกำหนดนั้นไม่มีความแน่นอน อาจกลายเป็นโทนเข้มหรือโทนอ่อนไปเลยก็ได้ อาทิเช่น สีแดง ที่ใช้เป็นตัวบอกตำแหน่งหัวรถนั้น หากโดนแสงสว่างมากๆ อาจทำให้ภาพที่กล้องจับได้กลายเป็นสีชมพูอ่อนไปเลยก็เป็นได้ หรือหากเป็นที่มืดก็จะกลายเป็นสีแดงออกโทนน้ำตาลเข้ม ซึ่งทำให้ค่าสีในโปรแกรมที่กำหนดสีเอาไว้ตรวจจับสีนั้นๆ ไม่ได้ ทำให้โปรแกรมเกิดการผิดพลาดไม่สามารถดำเนินการได้ อีกปัญหาที่รองลงมาของตัวโครงงานคือ การเชื่อมต่อระหว่างโปรแกรมที่เขียนขึ้นกับตัวรถ เนื่องจากได้ใช้การสื่อสารผ่านระบบไร้สายระหว่างตัวรถกับตัวคอมที่กำลังประมวลผลอยู่ ทำให้บางทีไม่สามารถเชื่อมต่อได้ ในส่วนของปัญหาที่เหลือก็คงจะเป็นปัญหาในระหว่างการประกอบรถ อาทิเช่น ไฟไม่เพียงพอ ไฟกระชากเวลามอเตอร์เซอร์โวทำงาน

### 5.2.2 แนวทางการแก้ไขปัญหา

การแก้ปัญหาในโครงงานนี้ต้องกล่าวก่อนว่าในกรณีข้างต้น ปัญหาได้แบ่งออกเป็นสามส่วน ซึ่งการแก้ปัญหานั้นมีดังนี้

1. ในส่วนของการที่กล้องไม่สามารถจับสีวัตถุได้คงที่ทำให้โปรแกรมไม่สามารถทำงาน จึงได้กำหนดค่าช่วงความเข้มของสีไว้ โดยให้ตัวโปรแกรมนั้นจับสีเป็นโทนแทน อาทิเช่น สีแดง ก็ให้จับทุกสีที่มีค่าเป็นโทนแดงที่อยู่ในช่วงที่ทดลองมาแล้วว่าโทนสีที่นำมาใช้นั้นสามารถจับสีแดงของหัวรถได้ทุกสถานการณ์

2. ส่วนของการเชื่อมต่อที่พบว่าบางทีไม่สามารถทำการเชื่อมต่อได้นั้น เนื่องมาจากมีโปรแกรมอื่นๆ หรือคอมพิวเตอร์ทำการพักการเชื่อมต่อไว้ จึงสั่งให้โปรแกรมที่เขียนขึ้นมานั้นเป็น

โปรแกรมที่ทำงานหลักโดยไม่สนโปรแกรมอื่น ซึ่งคอมพิวเตอร์ได้ใช้ระบบปฏิบัติการ Windows 10 การแก้ปัญหาคือการสั่งเปิดโปรแกรมด้วยคำสั่ง Run As Administrator

3. ทางด้านตัวรถที่มีปัญหาเกี่ยวกับไฟฟ้าที่ทำการจ่ายเข้าสู่ตัวรถผ่านแบตเตอรี่ จึงทำการติดตั้ง Buck Converter Module เพื่อกำหนดความต่างศักย์ และกระแสให้คงที่

## 5.3 ข้อเสนอแนะ

ตัวโครงงานนี้ตั้งใจไว้ว่าจะเป็นโมเดลของระบบขนส่งที่สามารถควบคุมผ่านกล้องโดยตัวคนทำหน้าที่กำหนดสถานที่ปลายทาง ในส่วนของตัวโปรแกรมที่ตัวคอมพิวเตอร์นั้น สามารถเพิ่มให้ควบคุมรถมากกว่า 1 คันได้ ส่วนโค้ดที่เขียนให้รถตรวจจับสี ก็อาจกำหนดให้คันที่ 2, 3, ..., n เป็นสีต่างๆ กันได้ หรืออาจเปลี่ยนเป็นการจับรูปภาพแทนก็ได้เช่นกัน โดยจะใช้ QR Code ซึ่งในส่วนของ OpenCV มีคำสั่งรองรับอยู่แล้ว โดย QR Code นั้นสามารถกำหนดในตัวโค้ดได้เลยไม่ต้องอาศัยจับหัวรถกับท้ายรถ ในการคำนวณหาทิศของตัวรถ และ QR Code สามารถทำได้อีกหลายๆ อย่างไม่ว่าจะเป็น การกำหนดชื่อตัวรถ หรือการระบุข้อมูลของรถคันนั้นๆ ได้ ส่วนตัวกล้องนั้นสามารถรับกล้องได้มากกว่า 1 ตัว โดยนำมาประกอบภาพกัน โดยเอาพิกัดพิกเซลมารวมกัน ทำให้สามารถสร้างแผนที่ขนาดใหญ่ได้ ในส่วนของการหลบหลีกสิ่งกีดขวางนั้น ต้องกำหนดพื้นสนามที่จะทำการวิ่ง โดยสร้างตัวแปรไว้คอยรับค่าพิกัดของสีแปลกปลอมนอกเหนือจากสีของพื้นสนาม แล้วเพิ่มโค้ดในส่วนการประมวลผลการวิ่ง ส่วนเรื่องการเชื่อมต่อ สามารถนำเอา Wi-Fi มาใช้ส่งคำสั่งไปยังตัวรถแทน Bluetooth ได้ ซึ่งจะทำให้การเชื่อมต่อสามารถควบคุมได้ไกลมากยิ่งขึ้น และเสถียรมากยิ่งขึ้น เนื่องจาก Wi-Fi Router ในปัจจุบันสามารถส่งสัญญาณคลื่นความถี่ได้มากกว่า 1 ช่วง ซึ่งเพียงแค่ช่วงเดียวก็สามารถรองรับการเชื่อมต่อได้มากถึง 50 คู่ได้อย่างไม่มีปัญหา ในส่วนของหน้าต่างควบคุมรูปร่างหน้าตาโปรแกรมก็สามารถปรับแต่งไปตามความเหมาะสมของผู้ใช้งานได้

# เอกสารอ้างอิง

[1] Alexander Mordvintsev & Abid K., **"OpenCV-Python Tutorials."**, January 2016 [Online] Available from: https://opencv-python-tutroals.readthedocs.io/en /latest/

[2] Arduino.cc, **"Arduino Nano."**, 2016 [Online] Available from: https://www.arduino.cc/en/uploads/Main/ArduinoNanoManual23.pdf

[3] Electronicaestudio, **"HC-05"**, January 2016 [Online] Available from: http://www.electronicaestudio.com/docs/istd016A.pdf

[4] HENRYDANGPRG., **"Color Detection in Python with OpenCV"**, [Online] Available from: https://henrydangprg.com/2016/06/26/color-detection-in-python-with-opencv/

[5] PythonProgramming.net, **"OpenCV with Python Intro and loading Images"**, [Online] Available from: https://pythonprogramming.net/loading-images-python-opencv-tutorial/

[6] Texas instruments, **"LM2621 Step-Up Converter"** [Online] Available from: http://www.ti.com/lit/ds/symlink/lm2621.pdf

[7] Texas instruments, **"LM2621 Step-Up Converter"** [Online] Available from: http://www.ti.com/lit/ds/symlink/lm2621.pdf

# ภาคผนวก

# ภาคผนวก ก

# ชุดคำสั่งควบคุม

เป็นการเรียกใช้ชุดคำสั่งใน library

```
from Tkinter import *
from ttk import *
import ttk
import cv2
import numpy as np
import serial
from time import sleep
import threading
import math
from math import atan2, degrees, pi
```

ประกาศตัวแปล

```
global X
X=''
global Y
Y=''
```

ประกาศตัวแปลที่เชื่อมต่อผ่าน serial

```
ser = serial.Serial('COM5', 9600)
```

ประกาศโปรแกรมย่อยเกี่ยวกับการประมวลผลภาพและการคำนวณ

```
def OpenCv(X,Y):
cap = cv2.VideoCapture(0)**********กำหนดช่องการรับค่าวีดีโอจากกล้อง**********
```

ประกาศตัวแปรย่อยในโปรแกรมย่อย

```
global cxAvga
global cxFounda
global cxAvgb
global cxFoundb
global iFrame
newTarget = "Yes"
cxAvga = 0      cyAvga = 0
xOlda = 0       yOlda = 0
cxAvgb = 0      cyAvgb = 0
xOldb = 0       yOldb = 0
```

ลูปการทำงานหลักของโปรแกรมย่อยประมวลผลภาพ และสั่งการเคลื่อนที่ไปยังตัวรถ

```
while(True):

#Strings to hold the "Target Lock" status.
        stringXOka = " "
        stringYOka = " "
#Strings to hold the "Target Lock" status.
        stringXOkb = " "
        stringYOkb = " "
        ret, frame = cap.read()
```

จับสีแดง

```
        hsva = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
        thresha = cv2.inRange(hsva,np.array((0, 30, 20)), np.array((10, 245, 230)))
        thresha2 = thresha.copy()
        contoursa,hierarchy =
        cv2.findContours(thresha,cv2.RETR_LIST,cv2.CHAIN_APPROX_SIMPLE)
```

จับสีน้ำเงิน

```
        hsvb = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
        threshb = cv2.inRange(hsvb,np.array((70, 10, 120)), np.array((200,150,150)))
        threshb2 = threshb.copy()
        contoursb,hierarchy =
        cv2.findContours(threshb,cv2.RETR_LIST,cv2.CHAIN_APPROX_SIMPLE)
```

คำนวณหากึ่งกลางที่ดีที่สุดของการจับสีหัวรถ

```
        max_areaa = 0
        for cnta in contoursa:
            areaa = cv2.contourArea(cnta)
            if areaa > max_areaa:
                max_areaa = areaa
                best_cnta = cnta
```

คำนวณหากึ่งกลางที่ดีที่สุดของการจับสีท้ายรถ

```
        max_areab = 0
        for cntb in contoursb:
            areab = cv2.contourArea(cntb)
            if areab > max_areab:
                max_areab = areab
                best_cntb = cntb
        Ma = cv2.moments(best_cnta)
        cxa = int(Ma['m10']/Ma['m00'])
        cya = int(Ma['m01']/Ma['m00'])
```

```
            cv2.circle(frame,(cxa,cya),10,(63,72,204),2)

            Mb = cv2.moments(best_cntb)
            cxb = int(Mb['m10']/Mb['m00'])
            cyb = int(Mb['m01']/Mb['m00'])
            cv2.circle(frame,(cxb,cyb),10,(237,28,36),2)
#A
            if cxAvga < (cxa + 5) and cxAvga > (cxa - 5):
                        xOlda == cxAvga
                        stringXOka = "X Locka"
            if cyAvga < (cya + 5) and cyAvga > (cya - 5):
                        yOlda == cyAvga
                        stringYOka = "Y Locka"
#B
            if cxAvgb < (cxb + 5) and cxAvgb > (cxb - 5):
                        xOldb == cxAvgb
                        stringXOkb = "X Lockb"
            if cyAvgb < (cyb + 5) and cyAvgb > (cyb - 5):
                        yOldb == cyAvgb
                        stringYOkb = "Y Lockb"
#A
#X
            cxAvga = cxAvga + cxa
            cxAvga = cxAvga / 2
#Y
            cyAvga = cyAvga + cya
            cyAvga = cyAvga / 2

#B
#X
            cxAvgb = cxAvgb + cxb
            cxAvgb = cxAvgb / 2
#Y
            cyAvgb = cyAvgb + cyb
            cyAvgb = cyAvgb / 2

            if newTarget == "Yes":
                        tX = X
                        tY = Y
                        newTarget = "No"
            if tX > cxAvga -20 and tX < cxAvga + 20:
                        print "Made it through the X"
                        if tY > cyAvga -20 and tY < cyAvga + 20:
                                    print "Made it through the Y"
```

```
                    ser.write(b'5')
                    ser.flushOutput()
                    break
#Slope main
        dxa = cxAvga - tX
        dya = cyAvga - tY

#Sub Slope
        dxb = cxAvgb - cxAvga
        dyb = cyAvgb - cyAvga
#Main
#Quad I -- Good
        if tX >= cxAvga and tY <= cyAvga:
                radsa = atan2(dya,dxa)
                degsa = degrees(radsa)
                degsa = degsa - 90
#Quad II -- Good
        elif tX >= cxAvga and tY >= cyAvga:
                radsa = atan2(dxa,dya)
                degsa = degrees(radsa)
                degsa = (degsa * -1)
#Quad III
        elif tX <= cxAvga and tY >= cyAvga:
                radsa = atan2(dxa,-dya)
                degsa = degrees(radsa)
                degsa = degsa + 180
#degsa = 3
        elif tX <= cxAvga and tY <= cyAvga:
                radsa = atan2(dxa,-dya)
                degsa = degrees(radsa) + 180
#degsa = 4

#Sub
#Quad I -- Good
        if cxAvga >= cxAvgb and cyAvga <= cyAvgb:
                radsb = atan2(dyb,dxb)
                degsb = degrees(radsb)
                degsb = degsb - 90
#Quad II -- Good
        elif cxAvga >= cxAvgb and cyAvga >= cyAvgb:
                radsb = atan2(dxb,dyb)
                degsb = degrees(radsb)
                degsb = (degsb * -1)
#Quad III
```

```
            elif cxAvga <= cxAvgb and cyAvga >= cyAvgb:
                    radsb = atan2(dxb,-dyb)
                    degsb = degrees(radsb)
                    degsb = degsb + 180
    #degsb = 3
            elif cxAvga <= cxAvgb and cyAvga <= cyAvgb:
                    radsb = atan2(dxb,-dyb)
                    degsb = degrees(radsb) + 180
    #degsb = 4

            targetDegsa = int(math.floor(degsa))
            strTargetDegsa = " "
            strTargetDegsa = str(math.floor(degsa))
            targetDegsb = int(math.floor(degsb))
            strTargetDegsb = " "
            strTargetDegsb = str(math.floor(degsb))
```

ทางเลือกคำสั่งเกี่ยวกับการส่งค่าไปยังหุ่นยนต์โดยจะส่งเป็นตัวเลขไปควบคุมการเคลื่อนที่หุ่น

```
            if targetDegsb <= (targetDegsa + 35) and targetDegsb >+ (targetDegsa - 35):
                ser.write(b'3')
                ser.flushOutput()
            else:
                if targetDegsb < targetDegsa:
                        if 1 < (targetDegsa - targetDegsb):
                    #abs(intHeadingDeg - targetDegs) >= 180:
                                ser.write(b'2')
                                ser.flushOutput()
                                print (targetDegsb - targetDegsa)
                                print "Right 1"
                            elif 1 > (targetDegsa - targetDegsb):
                #abs(intHeadingDeg - targetDegs) < 180:
                                ser.write(b'4')
                                ser.flushOutput()
                                print (targetDegsb - targetDegsa)
                                print "Left 1"
                    elif targetDegsb >= targetDegsa:
                            if 1 < (targetDegsb - targetDegsa):
                            #abs(intHeadingDeg - targetDegs) <= 180:
                                    ser.write(b'2')
                                    ser.flushOutput()
                                    print (targetDegsa - targetDegsb)
                                    print "Right 2"
                            elif 1 > (targetDegsb - targetDegsa):
```

```
                #abs(intHeadingDeg - targetDegs) > 180:
                    ser.write(b'4')
                    ser.flushOutput()
                    print (targetDegsb - targetDegsa)
                    print "Left 2"
```

สร้างภาพวาดภาพรวมถึงการสร้างข้อความ

```
        #Target circle
            cv2.circle(frame, (tX, tY), 10, (0, 0, 255), thickness=-1)
        #Background for text.
            cv2.rectangle(frame, (18,2), (170,160), (255,255,255), -1)
        #Target angle.
            cv2.line(frame, (tX,tY), (cxAvga,cyAvga),(0,255,0), 1)
        #car angle.
            cv2.line(frame, (cxAvga,cyAvga), (cxAvgb,cyAvgb),(0,255,0), 1)
            cv2.putText(frame,str(cxa)+" cx, "+str(cya)+"
        cy",(20,20),cv2.FONT_HERSHEY_COMPLEX_SMALL,.7,(0,0,0))
            cv2.putText(frame,str(cxAvga)+" cxA, "+str(cyAvga)+"
        cyA",(20,40),cv2.FONT_HERSHEY_COMPLEX_SMALL,.7,(0,0,0))
        cv2.putText(frame,stringXOka,(20,60),cv2.FONT_HERSHEY_COMPLEX_SMALL,.7,(0,0,0))
        cv2.putText(frame,stringYOka,(20,80),cv2.FONT_HERSHEY_COMPLEX_SMALL,.7,(0,0,0))
            cv2.putText(frame,"Car: "+strTargetDegsb+"
        Deg",(20,100),cv2.FONT_HERSHEY_COMPLEX_SMALL,.7,(0,0,0))
            cv2.putText(frame,"Target: "+strTargetDegsa+"
        Deg",(20,120),cv2.FONT_HERSHEY_COMPLEX_SMALL,.7,(0,0,0))
            cv2.imshow('frame',frame) #Color image
        cv2.imshow('thresha',thresha2) #Black-n-White Threshold image
            cv2.imshow('threshb',threshb2) #Black-n-White Threshold image
            if cv2.waitKey(33)== 27          cv2.destroyAllWindows()
                cap.release()
                ser.write(b'5')
                ser.flushOutput()
                break
```

โปรแกรมย่อยต่าง ๆ ที่ทำการเก็บค่าและส่งไปยังโปรแกรมประมวลผลภาพ

```
def PA():
  X=150
  Y=150
  print("X=%s\nY=%s\n" %(X, Y))
  OpenCv(X,Y);

def PB():
  X=550
```

```
    Y=200
    print("X=%s\nY=%s\n" % (X, Y))
    OpenCv(X,Y);

def PC():
    X=200
    Y=400
    print("X=%s\nY=%s\n" % (X, Y))
    OpenCv(X,Y);

def PD():
    X=500
    Y=400
    print("X=%s\nY=%s\n" % (X, Y))
    OpenCv(X,Y);

def GO():
    X= e1
    Y= e2
    print("X=GoAround" % (X, Y))
    OpenCv(X,Y);
```

โปรแกรมย่อยสั่งปิดโปรแกรม

```
def ExitPro():
        cv2.destroyAllWindows()
        master.quit()
```

เกี่ยวกับชุดคำสั่ง interface อยู่แกนหลักของโปรแกรม

```
        master = Tkmaster.geometry('200x1080'master.title('Robot Controller')
        Label(master, text="Tracking's Car").place(x= 27, y=20)
        Label(master, text="TargetX").place(x= 27, y=70)
        Label(master, text="TargetY").place(x= 27, y=150)
        Label(master, text="Station").place(x= 27, y=240)
        e1 = Entry(master)
        e1.place(x=27, y=100) )
        e2 = Entry(master)
        e2.place(x=27, y=180)
        X = e1
        Y = e2
```

สร้างปุ่มกด

```
        #A
        A=Button( master, text='Go to StationA', command = PA)
        photoA=PhotoImage(file="C:\Users\phetmongkol\Desktop\projectPIC\StationA.gif")
        LogoA = photoA.subsample(3, 3)
        A.config(image = LogoA)
        A.place(x=27,y=280)
        #B
        B=Button( master, text='Go to StationB', command = PB)
        photoB=PhotoImage(file="C:\Users\phetmongkol\Desktop\projectPIC\StationB.gif")
        LogoB = photoB.subsample(3, 3)
        B.config(image = LogoB)
        B.place(x=27,y=380)
        #C
        C=Button( master, text='Go to StationC', command = PC)
        photoC=PhotoImage(file="C:\Users\phetmongkol\Desktop\projectPIC\StationC.gif")
        LogoC = photoC.subsample(3, 3)
        C.config(image = LogoC)
        C.place(x=27,y=480)
        #D
        D=Button( master, text='Go to StationD', command = PD)
        photoD=PhotoImage(file="C:\Users\phetmongkol\Desktop\projectPIC\StationD.gif")
        LogoD = photoD.subsample(3, 3)
        D.config(image = LogoD)
        D.place(x=27,y=580)


        #exit button
        E=Button( master, text='exit', command = ExitPro)
        photoE=PhotoImage(file="C:\Users\phetmongkol\Desktop\projectPIC\exit.gif")
        LogoE = photoE.subsample(4, 4)
        E.config(image = LogoE)
        E.place(x=20,y=830)

        #go button
        G=Button( master, text="Start", command = GO)
        photoG=PhotoImage(file="C:\Users\phetmongkol\Desktop\projectPIC\start.gif")
        LogoG = photoG.subsample(4, 4)
        G.config(image = LogoG)
        G.place(x=20,y=680)
mainloop( )
```