หุ่นยนต์เลียนแบบการว่ายน้ำของปลาอย่างง่าย

A SIMPLE MIMIC FISH-BEHAVIOR ROBOT

นายตรีพัน ปุณกะบุตร
นางสาวพนิดา ดวงษ์วัฒนาสรี
นางสาวพิมพ์ชนก ยิ้มอยู่

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมการวัดคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2553

A SIMPLE MIMIC FISH-BEHAVIOR ROBOT

TREEPAN PUNKABUT

PANIDA KUWONGWATTANASAREE

PIMCHANOK YIMYOO

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2010

สาขาวิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

.....

หัวข้อปริญญานิพนธ์ หุ่นยนต์เลียนแบบการว่ายน้ำของปลาอย่างง่าย

A SIMPLE MIMIC FISH-BEHAVIOR ROBOT

นักศึกษาผู้จัดทำ	นายตรีพัน ปุณกะบุตร	รหัสนักศึกษา 50010541
	นางสาวพนิดา คูวงษ์วัฒนาเสรี	รหัสนักศึกษา 50011029
	นางสาวพิมพ์ชนก ยิ้มอยู่	รหัสนักศึกษา 50011106
ปริญญา	วิศวกรรมศาสตรบัณฑิต	
สาขา	วิศวกรรมการวัดคุม	
ปีการศึกษา	2553	

อาจารย์ผู้ควบคุมปริญญานิพนธ์	ลายมือชื่อ
ผู้ช่วยศาสตราจารย์ ดร. พงษ์ชัย นิลาศ	(แทน)

หัวข้อปริญญานิพนธ์ หุ่นยนต์เลียนแบบการว่ายน้ำของปลาอย่างง่าย

A SIMPLE MIMIC FISH-BEHAVIOR ROBOT

นักศึกษาผู้จัดทำ นายตรีพัน ปุณกะบุตร รหัสนักศึกษา 50010541

นางสาวพนิดา คูวงษ์วัฒนาเสรี รหัสนักศึกษา 50011029

นางสาวพิมพ์ชนก ยิ้มอยู่ รหัสนักศึกษา 50011106

อาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ ดร. พงษ์ชัย นิลาศ

ปีการศึกษา 2553

บทคัดย่อ

ในโครงงานนี้ เป็นการวิเคราะห์ถึงลักษณะการเคลื่อนที่ของปลาในระดับพื้นฐาน เพื่อนำมาสู่การออกแบบและผลิตโครงสร้างของ "หุ่นยนต์เลียนแบบพฤติกรรมการว่ายน้ำของปลาอย่างง่าย" โดยภายในโครงงานจะประกอบไปด้วยการศึกษา การทดลองจัดทำโครงสร้างของหุ่นยนต์และการเขียนโปรแกรมเพื่อ ทำการควบคุมการเคลื่อนที่โดยพื้นฐานของหุ่นยนต์ เลียนแบบการว่ายน้ำของปลา ผ่านไมโครอนโทรลเลอร์ชนิด ARM 7 Model ET-ARM BASE2103 ซึ่งทำหน้าที่ควบคุมการทำงานของเซอร์โวมอเตอร์ซึ่งเป็นส่วนประกอบหลักของการเคลื่อนที่ใน 2 ส่วนคือ ส่วนของโคนหางและครีบหาง ให้มีการเคลื่อนที่ในลักษณะของการสะบัดไป มา คล้ายการสะบัดของหางและการยืด หด ของกล้ามเนื้อลำตัวปลา นอกจากนั้นยังมีการรับค่าตัวแปรจากโฟโตเซนเซอร์เพื่อตรวจจับสิ่งกีดขวาง และสามารถเคลื่อนที่เลี้ยวหนีสิ่งกีดขวางดังกล่าวได้

Thesis Title	A Simple Mimic Fish-Behavior Robot
Authors	Mr.Treepan Punkabut
	Miss Panida Kuwongwattanasaree
	Miss Pimchanok Yimyoo
Thesis Advisor	Asst. Prof. Dr. Phongchai Nilas
Year	2010

ABSTRACT

This project analyzes the characteristics motion of the fish in a basic level for designing and manufacturing of "A Simple Mimic Fish Behavior Robot". The project will consist of studies and preparation of the structure of the robots and write programs to control the movement by microcontroller ARM 7 Model ET-ARM BASE2103 which controls the operation of the servo motor, a main component of motion in two parts of the peduncle-tail and fin-tail. To be moving in the nature of the flick back and forth like the tail flick and retraction of the muscular body fish. In addition, the parameters of the photoelectric sensors to detect obstacles and moving obstacles such turn away.

กิตติกรรมประกาศ

การจัดทำปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี เพราะได้รับคำปรึกษาและความอนุเคราะห์จาก ผศ.ดร.พงษ์ชัย นิลาศ ซึ่งเป็นอาจารย์ผู้ควบคุมปริญญานิพนธ์ฉบับนี้

ขอขอบพระคุณอาจารย์ และบุคลากรภาควิชาวิศวกรรมการวัดคุมทุกๆท่าน ที่เอื้อเฟื้ออุปกรณ์และสถานที่ในการทำโครงงาน ตลอดจนให้ความช่วยเหลือ และคำปรึกษาในด้านต่างๆ

ขอขอบพระคุณ นายณัฐพงษ์ สุวรรณจิตต์ ผู้ให้คำปรึกษาทางด้านซอฟแวร์และพร้อมทั้งให้คำแนะนำในโครงงานนี้อย่างเต็มที่

ขอขอบพระคุณชุมนุมโรบอท และ ชมรมออโตเมทีฟ ซึ่งได้ให้คำปรึกษาและให้ความช่วยเหลือในด้านเทคนิคในการทำหุ่นยนต์ อีกทั้งยังเอื้อเฟื้ออุปกรณ์ และเครื่องมือในการทำงานต่างๆ

และสุดท้ายนี้คณะผู้จัดทำขอกราบขอบพระคุณ บิดา มารดา และครอบครัว เป็นอย่างสูงสำหรับความรัก กำลังใจ คำปรึกษา และการสนับสนุนในด้านต่างๆ ที่มอบให้อยู่อย่างสม่ำเสมอจนเกิดเป็นแรงผลักดัน ที่ทำให้โครงงานนี้ประสบความสำเร็จและผ่านลุล่วงมาได้อย่างสมบูรณ์

คณะผู้จัดทำ

สารบัญ

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

สารบัญภาพ (ต่อ)

หน้า

บทที่ 1

บทนำ

1.1 ความสำคัญของปริญญานิพนธ์

ในโลกปัจจุบันเทคโนโลยีมีการพัฒนาไปอย่างรวดเร็ว ซึ่งสามารถสังเกตได้จากสิ่งรอบตัว ที่ไม่ว่าอะไรก็ล้วนแต่สะดวกสบายมากขึ้นด้วยศักยภาพของการพัฒนาของเทคโนโลยี การพัฒนาของเทคโนโลยีนั้นก็เกิดขึ้นได้ด้วยปัจจัยหลายอย่าง อาจจะเพื่อตอบสนองความต้องการทางด้านอุตสาหกรรม เพื่ออำนวยความสะดวกให้แก่ชีวิตของมนุษย์ เพื่อการประหยัดพลังงาน หรืออาจจะแค่เพื่อการสร้างสรรค์และตอบสนองแนวคิดส่วนบุคคล ซึ่งหากจะเปรียบเทียบแล้วหุ่นยนต์ก็คงเป็นเทคโนโลยีที่ถูกพัฒนาขึ้นมาจากแนวคิดอันสร้างสรรค์ของมนุษย์ที่ต้องการพัฒนาโครงสร้างให้มีกลไกการเคลื่อนที่อัตโนมัติที่ถูกควบคุมด้วยสมองกลหรือที่เรียกอีกอย่างหนึ่งว่าไมโครคอนโทรลเลอร์(Microcontroller)

โดยปริญญานิพนธ์นี้เป็นการนำเสนอเทคโนโลยีหุ่นยนต์ในอีกรูปแบบหนึ่งโดยเป็นการศึกษาและทดลองจัดทำโครงสร้างของหุ่นยนต์ที่มีกลไกการเคลื่อนที่และลักษณะของโครงสร้างเลียนแบบลักษณะการว่ายน้ำอย่างง่ายของปลา ซึ่งการจะจัดทำได้นั้นจำเป็นต้องมีการศึกษาและค้นคว้าเป็นอย่างดี โดยเฉพาะในส่วนของการควบคุมการเคลื่อนที่ ที่ต้องมีความเข้าใจในลักษณะของการว่ายน้ำของปลา และปัจจัยที่เกี่ยวข้องต่างๆ ดังนั้นโครงงานฉบับนี้จึงได้ถูกจัดทำขึ้นเพื่อเป็นแนวทางในการศึกษาและพัฒนาต่อไป

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1. ศึกษาโครงสร้างและลักษณะการทำงานของไมโครคอนโทรลเลอร์ พร้อมทั้งสามารถเขียนโปรแกรมผ่านไมโครคอนโทรลเลอร์ได้

2. ศึกษาถึงพฤติกรรมและโครงสร้างของปลาเพื่อนำไปสู่การออกแบบและจัดทำโครงสร้างของหุ่นยนต์ที่มีการเลียนแบบพฤติกรรมการว่ายน้ำของปลาในเบื้องต้นได้

3. ศึกษาลักษณะการทำงานของเซอร์โวมอเตอร์เพื่อนำไปสู่การควบคุมด้วยไมโครคอนโทรลเลอร์ได้

1.3 ขอบเขตของปริญญานิพนธ์

1. ออกแบบและจัดทำโครงสร้างของหุ่นยนต์เลียนแบบการว่ายน้ำของปลาอย่างง่าย

2. สามารถเขียนโปรแกรมควบคุมการทำงานของเซอร์โวมอเตอร์ที่เป็นส่วนประกอบหลักของการเคลื่อนที่ของโครงสร้าง เพื่อให้โครงสร้างเกิดการเคลื่อนที่ในลักษณะของการเลียนแบบการว่ายน้ำของปลาโดยพื้นฐาน

1.4 ขั้นตอนการศึกษา

1. ศึกษาถึงลักษณะการว่ายน้ำของปลาโดยทั่วไป และนำข้อมูลที่ได้มาวิเคราะห์เพื่อนำมาใช้กับการเคลื่อนที่ของหุ่นยนต์เลียนแบบการว่ายน้ำของปลาให้มีความเหมาะสม

2. รวบรวมผลการวิเคราะห์ที่ได้ เพื่อนำมาใช้ในการออกแบบและควบคุมโครงสร้างของหุ่นยนต์เลียนแบบการว่ายน้ำของปลา โดยมีการคำนึงถึงสภาวะการนำไปเคลื่อนที่จริง เช่น ความสมดุล ความสามารถในการกันน้ำได้

3. ทำการทดลองเพื่อตรวจสอบลักษณะการทำงาน และผลลัพธ์ที่ได้จากการศึกษาโดยรวม

1.5 ประโยชน์ที่คาดว่าจะได้รับ

เป็นการประยุกต์นำความรู้ที่ได้จากการศึกษาในด้านต่างๆ มาใช้ เพื่อทดสอบถึงปฏิภาณและความสามารถในการนำความรู้มาใช้ในการปฏิบัติจริง รู้จักการวิเคราะห์ดึงนำข้อมูลความรู้ต่างๆมาใช้ให้เกิดประโยชน์ ไม่ว่าจะด้วยจากการศึกษาด้วยตนเองหรือจากการถามผู้รู้ พร้อมทั้งมีไหวพริบในการปรับปรุงแก้ไขการทำงานเมื่อเกิดปัญหาขึ้น อีกทั้งยังเป็นการฝึกฝนตนเองให้รู้จักที่จะปฏิบัติงานร่วมกับผู้อื่น มีการรับฟังความคิดเห็นและรับผิดชอบในการปฏิบัติงานร่วมกันจนเกิดความสำเร็จ ซึ่งจะก่อให้เกิดเป็นประสบการณ์และความภาคภูมิใจในการที่จะปฏิบัติงานต่อไปในภายภาคหน้า

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ความรู้พื้นฐานในการสร้างหุ่นยนต์

ความหมายของ "หุ่นยนต์" โดยสถาบันหุ่นยนต์อเมริกา (The Robotics Institute of America) ได้ให้ความหมายไว้ ดังนี้ "น้ำเชื้อคือส่วนประกอบของหุ่นยนต์ประกอบต่างๆ เครื่องมือหรืออุปกรณ์พิเศษ ตลอดจนการเคลื่อนที่ได้หลากหลาย ตามที่ตั้งลำดับการทำงาน เพื่อสำหรับใช้งานหลากหลายประเภท" (A robot reprogrammable, multifunctional manipulator designed to move materials, parts, tools or specialized devices through various programmed motions for the performance of a variety of tasks.)

นิยามดังกล่าว อีกนัยหนึ่งก็คือ เครื่องจักรกลทุกชนิดที่สามารถปฏิบัติงานแทนมนุษย์ได้ทุกประเภท ทั้งทางตรงและทางอ้อม รวมทั้งในงานที่เสี่ยงอันตรายโดยที่มนุษย์ไม่สามารถปฏิบัติงานได้ ตลอดจนการทำงานที่เป็นอัตโนมัติโดยตนเองหรือถูกควบคุมโดยมนุษย์ และสามารถปรับเปลี่ยนรูปแบบการทำงานได้หลากหลาย

2.1.1 ประเภทของหุ่นยนต์

ประเภทของหุ่นยนต์ สามารถแบ่งแยกได้หลากหลายรูปแบบตามลักษณะเฉพาะของการใช้งาน ได้แก่การแบ่งประเภทตามการเคลื่อนที่ นอกจากนี้อาจจำแนกตามรูปลักษณ์ภายนอกด้วยก็ได้

2.1.1.1 การแบ่งประเภทตามการเคลื่อนที่ได้

1. หุ่นยนต์ที่ติดตั้งอยู่กับที่ ไม่สามารถเคลื่อนที่ได้หุ่นยนต์ที่ติดตั้งอยู่กับที่สามารถ เคลื่อนไหวไปมาแต่ไม่สามารถเคลื่อนที่ได้ หุ่นยนต์ในประเภทนี้ได้แก่ แขนกลของหุ่นยนต์ที่ใช้ในงานด้านอุตสาหกรรมต่าง ๆ เช่นงานด้านอุตสาหกรรมผลิตรถยนต์แขนกลของหุ่นยนต์ที่ใช้งานในด้านการแพทย์ เช่นแขนกลที่ใช้ในการผ่าตัด หุ่นยนต์ประเภทนี้จะมีลักษณะโครงสร้างที่ใหญ่โต เทอะทะและมีน้ำหนักมาก ใช้พลังงานให้สามารถเคลื่อนไหวได้จากแหล่งจ่ายพลังงานภายนอก และจะมีการกำหนดขอบเขตการเคลื่อนไหวของหุ่นยนต์เอาไว้ ทำให้หุ่นยนต์สามารถเคลื่อนไหวไปมาได้ในเฉพาะที่ที่กำหนดเอาไว้เท่านั้น

2. หุ่นยนต์ที่สามารถเคลื่อนไหวและเคลื่อนที่ได้

หุ่นยนต์สามารถเคลื่อนไหวร่างกายไปมาได้อย่างอิสระ หมายความถึงหุ่นยนต์ที่สามารถเคลื่อนย้ายตัวเองจากตำแหน่งหนึ่งไปยังอีกตำแหน่งหนึ่งได้อย่างอิสระ หรือมีการเคลื่อนที่ไปมาในสถานที่ต่าง ๆ เช่น หุ่นยนต์ที่ใช้ในการสำรวจดวงจันทร์ขององค์กรนาซ่า หุ่นยนต์สำรวจใต้พิภพหรือ

หุ่นยนต์ที่ใช้ในการขนถ่ายสินค้า ซึ่งหุ่นยนต์ที่สามารถเคลื่อนไหวได้นี้ ถูกออกแบบลักษณะของโครงสร้างให้มีขนาดเล็กและมีระบบเคลื่อนที่ไปมา รวมทั้งมีแหล่งจ่ายพลังสำรองภายในร่างกายของตนเอง แตกต่างจากหุ่นยนต์ที่ไม่สามารถเคลื่อนที่ไปมา ได้ ซึ่งจะต้องมีแหล่งจ่ายพลังงานอยู่ภายนอก แหล่งจ่ายพลังสำรองภายในร่างกายของหุ่นยนต์ที่สามารถเคลื่อนไหวร่างกาย และสามารถเคลื่อนที่ไปมาได้นั้น โดยปกติแล้วจะถูกออกแบบลักษณะของโครงสร้างให้มีขนาดเล็ก รวมทั้งมีปริมาณน้ำหนักไม่มาก เพื่อไม่ให้เป็นอุปสรรคต่อการปฏิบัติงานของหุ่นยนต์หรืออุปสรรคในการเคลื่อนที่

2.1.1.2 การแบ่งประเภทตามลักษณะรูปร่างภายนอก

โดยทั่วไป หุ่นยนต์ยังถูกจำแนกตามลักษณะรูปลักษณ์ภายนอก และมีคำศัพท์เฉพาะเรียกต่างๆกันไป ได้แก่

1. หุ่นยนต์ฮิวแมนนอยด์ (Humanoid Robot) เป็นลักษณหุ่นยนต์ที่เหมือนกับมนุษย์

2. แอนดรอยด์ (Android) เป็นการเรียกหุ่นยนต์คล้ายมนุษย์ที่สามารถแสดงออกเหมือนมนุษย์แม้ว่ารากศัพท์ภาษากรีกของคำนี้หมายถึงเพศชาย แต่การใช้ในบริบทภาษาอังกฤษมักไม่ได้มีความหมายเจาะจงว่าเป็นเพศใด

3. จีนอยด์ (Gynoid) เป็นการเรียกหุ่นยนต์คล้ายมนุษย์เพศหญิง

4. แอ็คทรอยด์ (Actriod) เป็นหุ่นยนต์ที่เลียนแบบพฤติกรรมมนุษย์ เช่น กระพริบตา หายใจ เริ่มพัฒนาโดย มหาวิทยาลัยโอซาก้าและบริษัทโคโคโระ

5. ไซบอร์ก (Cyborg) เป็นหุ่นยนต์ที่เชื่อมต่อกับสิ่งมีชีวิต หรือ ครึ่งคนครึ่งหุ่น เริ่มปรากฏครั้งแรกในเรื่องแต่งปี 1960

6. นาโนโรบอท (Nano robot) เป็นหุ่นยนต์ขนาดเล็กมาก ขนาดประมาณ 0.5-3 ไมครอน

2.1.2 ส่วนประกอบของหุ่นยนต์

ในหุ่นยนต์นั้นจะประกอบไปด้วยส่วนประกอบหลายส่วน ดังต่อไปนี้

1. มานิพูเลเตอร์ (manipulator) หรือ โรเวอร์ (rover) เป็นส่วนประกอบหลักของหุ่นยนต์ประกอบด้วยชิ้นส่วน (1inks) ข้อต่อ (joints) และโครงสร้างของตัวหุ่นยนต์

2. ผลสิ้นสุด (end effecter) เป็นส่วนปลายสุดท้ายต่อกับข้อตอส่วนสุดท้ายของแขนกล โดยปกติจะให้หยิบจับวัตถุ หรือทำงานเฉพาะทาง เช่น ที่ติดตั้งเครื่องเชื่อม หรือเครื่องพ่นสี โดยปกติการควบคุมโดยใช้ชุดควบคุม PLC (programmable 1ogic contro11er)

3. ชุดขับ(actuators) เปรียบเสมือนกล้ามเนื้อ หรือชุดขับเคลื่อนของหุ่นยนต์ รูปแบบที่ใช้โดยทั่วไป ได้แก่ มอเตอร์ กระบอกลม กระบอกไฮดรอลิก

4. เซนเซอร์ (sensors) คือ อุปกรณ์หัววัด เพื่อใช้วัดข้อมูลของหุ่นยนต์ในการรับรู้สภาพแวดล้อม เช่น ชุดควบคุมต้องการที่จะทราบตำแหน่งของชิ้นส่วนของแขน ว่าอยู่ที่ ณ ตำแหน่งใด เทียบกับมนุษย์ที่สามารถหยั่งรู้ได้ว่าแขนหรือขาของเราอยู่ที่ใดแม้ในที่มืด เพราะชุดเครื่องมือวัดย้อนกลับของมนุษย์ที่อยู่ในกล้ามเนื้อ แล้วส่งข้อมูลผ่านระบบประสาทไปยังสมองโดยสมองรับรู้ได้ว่าขณะนี้แขนหรือขาอยู่ที่ใด หลักการเหมือนในหุ่นยนต์ โดยชุดเครื่องมือวัดป้อนกลับ (feedback sensors) ส่งข้อมูลกลับแต่ละข้อต่อไปยังชุดควบคุมต่อไป

5. ชุดควบคุม (processor and controller) คล้ายสมองในมนุษย์โดยได้รับข้อมูลจากเซนเซอร์(sensor) และสั่งการไปยังชุดขับ (actuators) เพื่อให้เคลื่อนที่ตามที่ได้ตั้งโปรแกรมไว้ โปรแกรม (software) แบ่งออกเป็น 3 กลุ่ม

- ระบบปฏิบัติการพื้นฐาน (operating systems)
- โปรแกรมของหุ่นยนต์ (robotic software) ซึ่งทำการควบคุมการเคลื่อนที่ของข้อต่อจากสมการจลน์ศาสตร์ (kinetic equation) จากนั้นข้อมูลจะถูกส่งต่อไปชุดควบคุมจลศาสตร์
- เป็นโปรแกรมควบคุมที่ใช้งานเฉพาะทาง เช่น ระบบวิทัศน์ (vision systems) เป็นต้น

6. แหล่งพลังงาน (power supply) เป็นส่วนสำคัญที่จ่ายพลังงานไปชุดควบคุม และ มานิพูเลเตอร์ (manipulator) แบ่งออกเป็น 2 ส่วนใหญ่

- แหล่งจ่ายไฟฟ้า
- แหล่งจ่ายพลังงานในการขับเคลื่อน เช่น ในหุ่นยนต์ที่ใช้ระบบนิวแมติกส์ (pneumatic)จะต้องมีแหล่งจ่ายลมอัด เป็นต้น

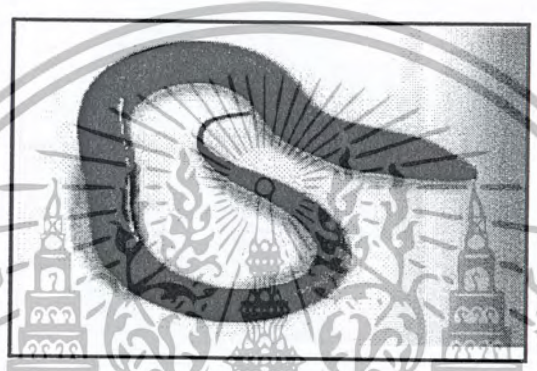
ที่กล่าวมาเป็นองค์ประกอบขั้นพื้นฐานที่สำคัญที่หุ่นยนต์ทุกตัวต้องมี ยังมีนอกเหนือจากที่กล่าวมาอีกมากเพื่อนำไปใช้ในงานต่าง ๆ ตามแต่ลักษณะของงาน หุ่นยนต์ในอนาคตอาจจะไปถึง มีความฉลาด และการมีความคิดเป็นของมันเอง

ซึ่งการสร้างหุ่นยนต์สร้างตามลักษณะที่ต้องการใช้งานนั้น หุ่นยนต์แต่ละแบบอาจจะมีความแตกต่างกัน หุ่นยนต์แต่ละชนิดจะเคลื่อนที่ไปตามโครงสร้างที่เป็นไปตามกฎของฟิสิกส์ (physic) คือมีการเคลื่อนไหวตามโครงสร้างที่ออกแบบ โดยตัวขับให้หุ่นยนต์ได้เคลื่อนไหว เช่น มอเตอร์ มีระบบเซนเซอร์ตรวจจับการทำงาน และมีสมองกลคอยควบคุมการทำงานของชิ้นส่วนทั้งหมด เป็นต้น

2.2 ลักษณะการเคลื่อนที่ของปลาตามธรรมชาติ

โดยปลานั้นเป็นสิ่งมีชีวิตที่มีการเคลื่อนที่ โดยอาศัยการหดตัวของกล้ามเนื้อลำตัวและการเคลื่อนไหวของครีบในส่วนต่างๆ เช่น การสะบัดของครีบหาง ซึ่งการเคลื่อนที่ของปลานั้นก็มีลักษณะการเคลื่อนที่หลายแบบแตกต่างกันไป ซึ่งจากการศึกษาคุณลักษณะการเคลื่อนที่ของปลานั้นสามารถแบ่งออกได้เป็น 3 ประเภทหลัก คือ

1. เคลื่อนที่แบบปลาไหล (anguilliform หรือ eel - like) เป็นการเคลื่อนที่ที่มีลักษณะเหมือนการเลื้อยของงู ซึ่งเกิดจากการหดตัวของมัดกล้ามเนื้อสลับกันไปในแต่ละข้างของลำตัว เช่น ปลาไหล

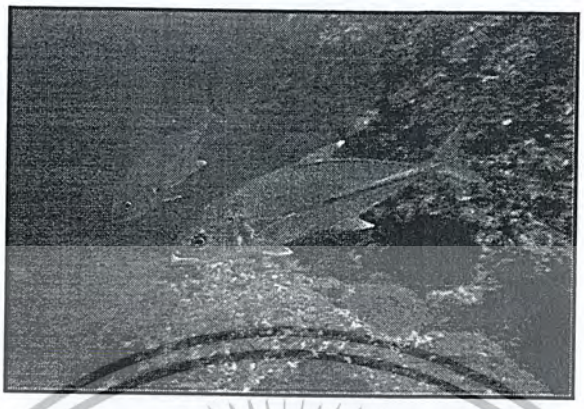


ภาพที่ 2.1 การเคลื่อนที่แบบปลาไหล

2. การเคลื่อนที่แบบแจวเรือ (ostraciform หรือ trunk fish - like) เกิดจากการหดตัวของมัดกล้ามเนื้อแต่ละตอนในแต่ละข้างของลำตัวสลับกันไปมาทำให้เกิดลักษณะวิกแวก(wigwag motion) ที่เห็นได้ชัดคือ การโบกพัดครีบหาง พบในพวก เช่น ปลาวัว ปลาสี่เหลี่ยม

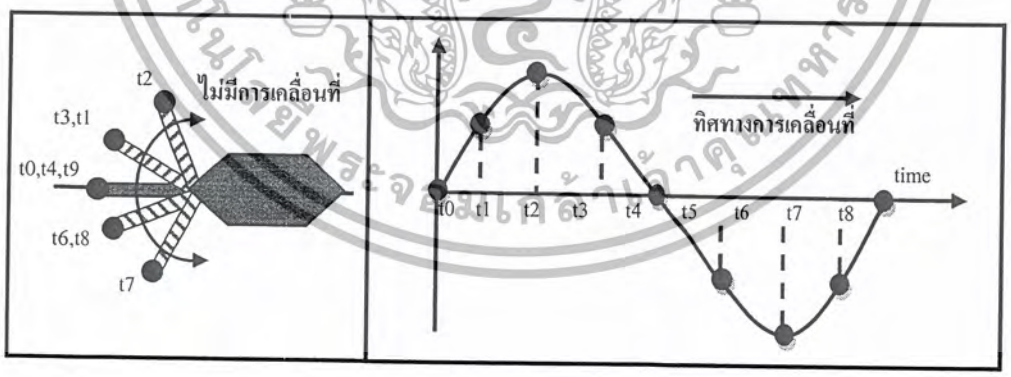
ภาพที่ 2.2 การเคลื่อนที่แบบแจวเรือ

ต้นลำตัวจะเคลื่อนไปก่อนแล้วไล่ตามลงมาทางครีบหาง พบได้ในการว่ายน้ำของปลาทั่ว ๆ ไป เช่น ปลาหางแข็ง ปลาทู ปลาสีกุน ปลาพวกนี้ว่ายน้ำได้รวดเร็ว



ภาพที่ 2.3 การเคลื่อนที่แบบคาร์แรงกิฟอร์ม

จากการที่ได้กล่าวมานั้นจะเห็นได้ว่าปลาเป็นสิ่งมีชีวิตที่มีการเคลื่อนที่ที่ซับซ้อนและหลากหลาย จึงเป็นการยากที่จะมีการเจาะจงถึงคุณลักษณะการเคลื่อนที่โดยชัดเจน ดังนั้นจึงได้มีการตัดสินใจที่จะเลียนแบบลักษณะการเคลื่อนที่โดยพื้นฐานเท่านั้น โดยจากการศึกษาถึงลักษณะการเคลื่อนที่ที่เราได้ศึกษามาทั้งหมดนั้น ทำให้ทราบว่าปลาโดยส่วนใหญ่จะใช้การหดตัวของกล้ามเนื้อข้างลำตัวสลับกันไป ซึ่งจะก่อให้เกิดการบิดของลำตัวซ้าย-ขวาสลับกันและการสะบัดของครีบหาง



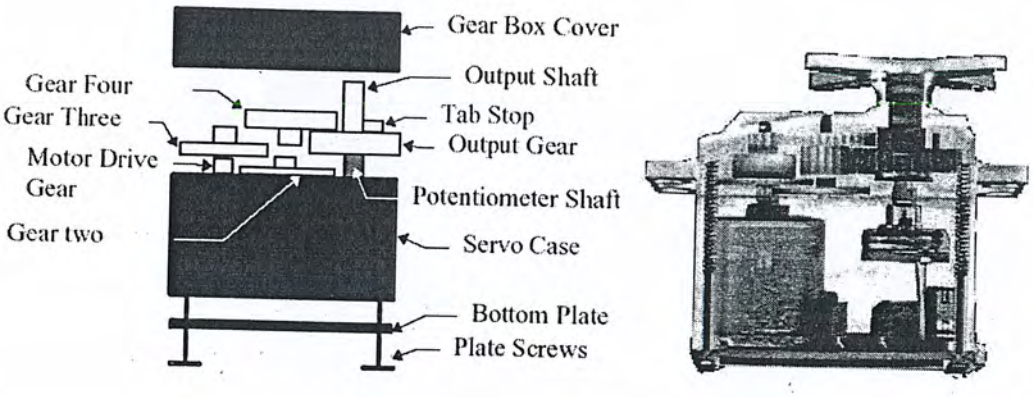
ภาพที่ 2.4 ตำแหน่งการเคลื่อนที่ของครีบหางปลา

จากภาพที่ 2.4 เป็นการจำลองตำแหน่งการเคลื่อนที่ของครีบหางปลาจากมุมมองด้านบน โดยเปรียบจุดสีดำเป็นตำแหน่งปลายครีบหางของปลา รูปทางซ้ายมือเป็นการจำลองลักษณะการสะบัดของครีบหางปลาเมื่อปลาไม่มีการเคลื่อนที่ จะได้ว่าหางปลาจะเคลื่อนที่ในลักษณะของการสะบัดขึ้น

ลงเป็นเส้นโค้งกลับไปกลับมาในตำแหน่งเดิม ในขณะที่รูปทางซ้ายมือนั้นเป็นการจำลองลักษณะการสะบัดของครีบหางปลาเมื่อปลามีการเคลื่อนที่ไปตามทิศทางของลูกศร ซึ่งจะได้ว่าเมื่อครีบหางปลามีลักษณะการเคลื่อนที่ในลักษณะของการสะบัดขึ้นนั้นไปพร้อมๆ กับการเคลื่อนที่ของตัวปลานั้น จะทำให้ตำแหน่งของครีบหางปลามีการเปลี่ยนแปลงไปในลักษณะของการโค้งขึ้นเป็นพาราโบลา และเมื่อมีการสะบัดกลับลงมาก็จะมีการเปลี่ยนแปลงไปในลักษณะของการโค้งลงในลักษณะเช่นเดียวกัน แต่มีทิศทางตรงกันข้ามกัน ซึ่งเมื่อพิจารณาจากลักษณะการเคลื่อนที่โดยรวมแล้วแล้วจะทำให้สังเกตได้ว่าครีบหางปลามีตำแหน่งการเคลื่อนที่ในลักษณะใกล้เคียงกับคลื่นซายน์(sine wave) ในขณะเดียวกันถ้าเปรียบเทียบจุดสีดำแทนตำแหน่งหนึ่งบนลำตัวของปลา ที่มีการเคลื่อนที่ในลักษณะบิดไป-บิดมาเพื่อให้เกิดการเคลื่อนที่ ก็จะได้ว่าตำแหน่งการเคลื่อนที่ของลำตัวของปลานั้นก็จะมีลักษณะการเคลื่อนที่เช่นเดียวกันกับตำแหน่งของครีบหางปลา

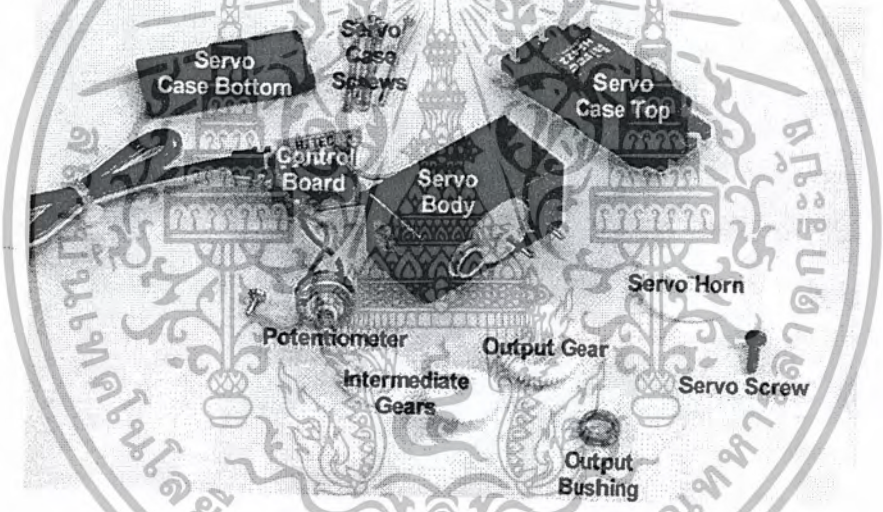
2.3 เซอร์โวมอเตอร์(Servo Motor)

Servo motor คือ มอเตอร์ไฟฟ้ากระแสตรง(DC motor) ที่ถูกประกอบรวมกับ ชุดเกียร์ และส่วนควบคุมต่างๆ ไว้ในโมดูลเดียวกัน หรือ ภายในกล่องพลาสติกเดียวกัน โดยมอเตอร์ชนิดนี้จะมีสายต่อใช้งานเพียง 3 เส้นเท่านั้น คือ VCC , GND , และสายสัญญาณควบคุม (Control Line) ซึ่งสามารถควบคุมให้มอเตอร์หมุนซ้ายหรือขวา ได้จากสายสัญญาณเพียงเส้นเดียวโดยสัญญาณที่ใช้ควบคุมนี้จะเป็นสัญญาณ พัลส์วิดมอด (PWM) แบบ TIL Level ระดับแรงดันที่จ่ายให้มอเตอร์นี้จะมีขนาดเล็กน้ำหนักเบา,ให้แรงบิดสูง,กินพลังงานน้อย และ สามารถควบคุม ด้วยแรงดันลอจิกที่เป็น TIL ได้โดยตรงไม่จำเป็นต้องต่อวงจรขับ(Driver) อื่นๆ เพราะ มอเตอร์ชนิดนี้จะมีวงจรควบคุมบรรจุไว้ภายในอยู่แล้ว ซึ่งมอเตอร์ชนิดนี้สามารถควบคุมให้หมุนไปในตำแหน่ง หรือ ทิศทางองศาที่ต้องการได้ โดยอาศัยสัญญาณความกว้างพัลส์ (pulse width) ที่ป้อนให้มอเตอร์ แต่เซอร์โวมอเตอร์นี้จะหมุนได้แค่เพียงในช่วงประมาณ 180 องศา หรือ ครึ่งรอบเท่านั้น หรือ บางรุ่นอาจหมุนได้ถึง 210 องศา แต่จะไม่สามารถหมุนเป็นวงรอบได้ เนื่องจากโครงสร้างภายในจะประกอบด้วยตัวต้านทานชนิดปรับค่าได้ (VR) ที่ทำหน้าที่ตรวจสอบตำแหน่งการหมุนของมอเตอร์ และ ตัวต้านทานนี้จะถูกยึดติดกับแกนหมุนของมอเตอร์ ซึ่งจากการที่ตัวต้านทานปรับค่านี้ไม่สามารถหมุนเป็นวงรอบได้ ดังนั้น เซอร์โวมอเตอร์จึงถูกออกแบบให้หมุนได้เพียงแค่ประมาณ 180 องศา หรือประมาณครึ่งรอบเท่านั้น เพื่อป้องกันความเสียหายที่จะเกิดกับตัวต้านทานปรับค่าได้ แต่ถ้าหากเราต้องการให้มอเตอร์หมุนเป็นวงรอบ (360 องศา) นั้นก็สามารถทำได้ โดยจะต้องทำการปรับแต่ง (Modified)ดัดแปลงชิ้นส่วนบางอย่างของมอเตอร์ ซึ่งวิธีการต่างๆจะได้กล่าวถึงในภายหลัง



ภาพที่ 2.5 ส่วนประกอบของเซอร์โวมอเตอร์

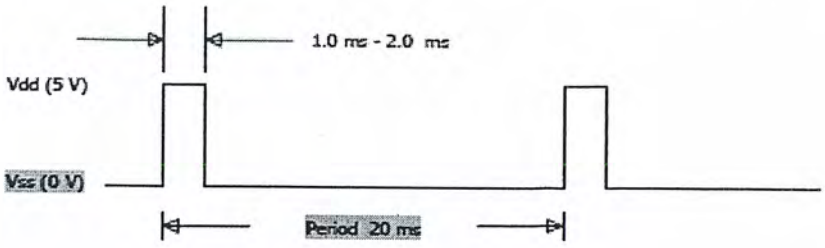
2.3.1 ส่วนประกอบต่างๆของ Servo Motor

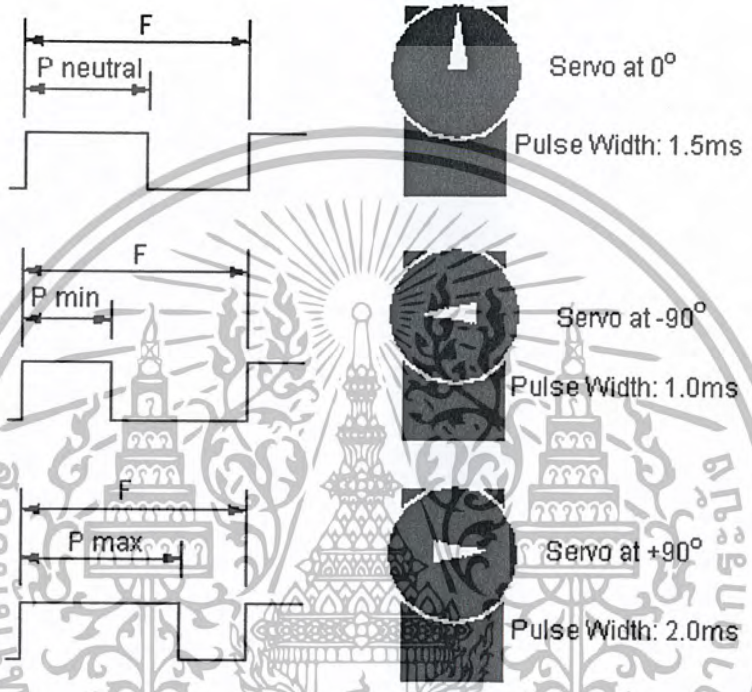


ภาพที่ 2.6 ส่วนประกอบของเซอร์โวมอเตอร์

2.3.2 หลักการทำงานของ Servo Motor

การควบคุมการทำงานของ เซอร์โวมอเตอร์ ทำได้โดย การป้อนสัญญาณความกว้างพัลส์ให้กับมอเตอร์ซึ่งตำแหน่งและทิศทางการหมุนของมอเตอร์นี้ขึ้นกับขนาดของความกว้างของพัลส์นั้นๆ โดยทั่วไปแล้วความกว้างของสัญญาณพัลส์จะมีจุดให้อ้างอิง 3 จุด ดังรูป คือ





ภาพที่ 2.7 สัญญาณพัลส์ควบคุมเซอร์โวมอเตอร์

- สัญญาณความกว้างพัลส์ 1.5 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม 0 องศา หรือ จุดกึ่งกลางของมอเตอร์

- สัญญาณควบคุมความกว้างพัลส์ขนาด 1 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม -90 องศา หรือ ทวนเข็มนาฬิกา

- สัญญาณความกว้างพัลส์ขนาด 2 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปที่ตำแหน่งมุม +90 องศา หรือ ในทิศทางตามเข็มนาฬิกา

ค่าความกว้างพัลส์ และ ระยะองศาการหมุนของมอเตอร์ที่อธิบายด้านบน นั้นเป็นเพียงค่าประมาณเท่านั้น ทั้งนี้ระยะการหมุน และ ขนาดของพัลส์ที่ควบคุมการทำงานของมอเตอร์ในแต่ละยี่ห้ออาจจะไม่เท่ากัน ดังนั้นในการใช้งานจึงควรศึกษารายละเอียดของมอเตอร์ในแต่ละรุ่นที่นำมาใช้ ซึ่งโดยปกติแล้วรายละเอียดต่างๆของมอเตอร์ มักจะมีติดมากับตัวมอเตอร์นั้นๆอยู่แล้ว

ส่วนการที่จะควบคุมให้มอเตอร์หมุนเป็นมุมอื่นๆ นั้นก็สามารถทำได้โดยการป้อนสัญญาณพัลส์เป็นระดับความกว้างต่างๆ โดยอ้างอิงจากจุด ทั้ง 3 จุดที่กล่าวมานี้ ตัวอย่างเช่น ถ้าต้องการให้มอเตอร์หมุนไปที่มุม -45 องศา เราก็จะต้องป้อนสัญญาณพัลส์ที่มีความกว้าง 1.25 ms เป็นต้น และสัญญาณพัลส์นี้จะต้องจ่ายให้มอเตอร์ทุกๆ 20 ms (Period) เพื่อรักษาสภาพตำแหน่งของมอเตอร์ไว้

โดยหลักการคือ จะอาศัยการเปรียบเทียบช่วงเวลาของความกว้างพัลส์ที่จ่ายให้กับมอเตอร์ทางขาสัญญาณควบคุมกับค่าเวลาของวงจร RC ภายในบอร์ดควบคุมในตัวของมอเตอร์ ซึ่งค่าเวลาของวงจร RC นี้จะมีการเปลี่ยนแปลงตามการหมุนของมอเตอร์ เนื่องจากตัวต้านทานปรับค่าจะถูกยึดติดอยู่กับแกนหมุนของมอเตอร์ ซึ่งการหมุนของมอเตอร์จะทำให้ค่าความต้านทานของตัวต้านทานปรับค่า (VR) เปลี่ยนแปลงไป เป็นผลทำให้ค่าเวลาของวงจร RC เปลี่ยนแปลงตามไปด้วย โดยในขณะที่เราป้อนสัญญาณความกว้างพัลส์ให้กับมอเตอร์ทางขาสัญญาณควบคุม สัญญาณนี้จะถูกนำไปเปรียบเทียบกับค่าเวลาของวงจร RC หากค่าทั้ง 2 ไม่เท่ากันมอเตอร์ก็จะหมุนทำให้ค่าเวลาของวงจร RC เปลี่ยนแปลงจนกระทั่งค่าเวลาความกว้างพัลส์ของ วงจร RC เปลี่ยนแปลงจนเท่ากับสัญญาณพัลส์ทางขาควบคุม (Control line) มอเตอร์จึงจะหยุดหมุน

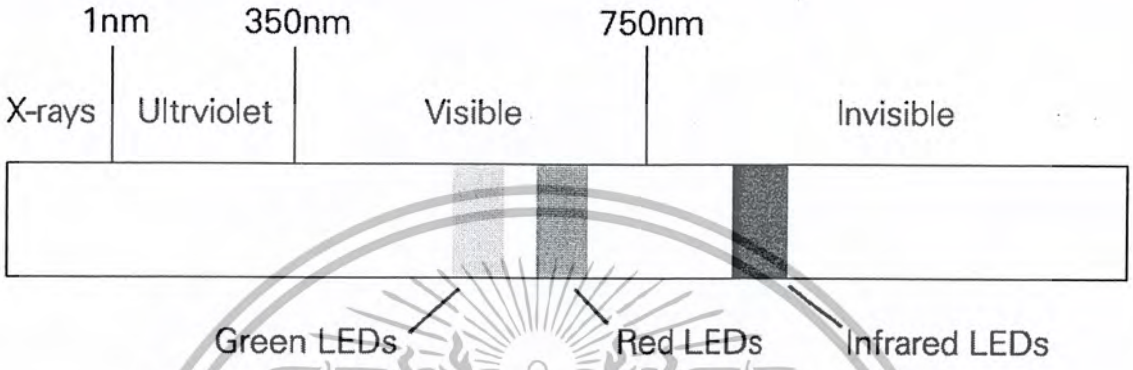
2.4 เซนเซอร์(Photo Sensor)

Proximity Sensor หรือ Proximity Switch เป็นอุปกรณ์การตรวจจับว่ามีวัตถุอยู่ใกล้กับบริเวณที่เราติดตั้ง Sensor ไว้หรือไม่ โดยทั่วไป sensor ประเภทนี้จะไม่มีการสัมผัสกับวัตถุที่มันตรวจจับโดยตรง แต่ proximity sensor จะอาศัยหลักการอื่นในการตรวจจับ โดยไม่ต้องมีการสัมผัสกับวัตถุโดยตรง สำหรับในอุตสาหกรรมการผลิตทั่วไป proximity sensor จะมีด้วยกันหลายแบบ แต่สำหรับที่ใช้ภายในโครงงานนี้คือประเภท Photoelectric Proximity sensor

2.4.1 Photoelectric Proximity Sensor

Photoelectric Proximity Sensors เป็นเซนเซอร์ที่ใช้ตรวจสอบวัตถุอีกประเภทหนึ่งที่เรานิยมใช้กันอยู่ในปัจจุบัน เป็นเซนเซอร์ที่มีความแม่นยำสูงหลักการทำงานของมันก็คือ จะตรวจจับการปรากฏของวัตถุ ด้วยการที่วัตถุตัดผ่านลำแสงหรือสะท้อนแสงที่สร้างขึ้นจากเซนเซอร์นี้ ส่วนประกอบหลักของเซนเซอร์นี้จะมีสองส่วนคือ ส่วนที่กำเนิดแสง Transmitter หรือ Emitter ซึ่งอาจจะสร้างแสงในย่านที่ตาเรามองเห็นได้ จนถึงบางรุ่นที่ใช้แสง infrared ข้อสำคัญก็คือแสงที่สร้างขึ้นนี้จะเป็นแสงความถี่เดียว เพื่อให้แตกต่างจากแสงที่อยู่รอบๆตัวเรา จากนั้นแสงจะถูกส่งไปที่ตัวรับแสง Receiver ซึ่งตัวรับแสงจะทำหน้าที่แยกว่ามีแสงจากแหล่งกำเนิดมาตกกระทบหรือไม่ เพื่อใช้เป็นข้อมูลในการสั่งการทำงานของวงจร output ในเซนเซอร์ต่อไป โดยทั่วไปอุปกรณ์ใน Receiver จะเป็น photodiode หรือ phototransistor ซึ่งจะมีการเปิดหรือปิดวงจรตามที่มี

แสงมาตกกระทบอุปกรณ์นี้ สำหรับแสงที่ใช้ในเซนเซอร์ประเภทนี้มักจะส่งออกจาก transmitter ออกเป็นสัญญาณ pulse ที่มีความถี่ประมาณ 5 and 30 KHZ และแสงที่ใช้มักจะมีความถี่เดียว ตามที่บอกไปแล้วโดยแสงนิยมให้ Light – emitting diode (LED) เป็นแหล่งกำเนิดแสงและสีของแสงจะเป็นตัวกำหนดความถี่หรือความยาวของแสงด้วย ตามที่แสดงในภาพ

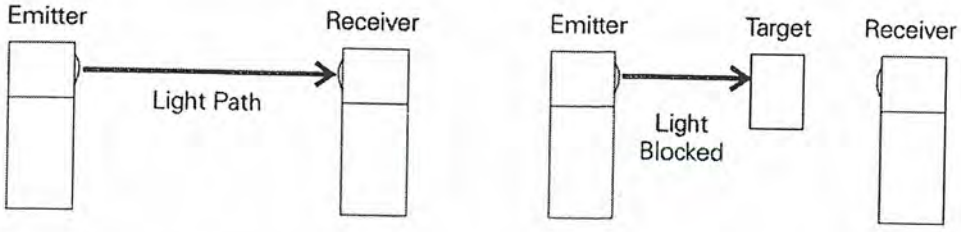


ภาพที่ 2.8 ความยาวคลื่นของแสงที่ใช้ใน Photoelectric Sensor

2.4.2 การตรวจจับวัตถุ

การตรวจจับวัตถุของ photoelectric ที่นิยมใช้มีอยู่หลายวิธี ขึ้นกับสภาพสิ่งแวดล้อม ความสะดวกในการติดตั้ง หรือชนิดของวัสดุที่ตรวจจับ โดยมีการแบ่งการติดตั้งออกเป็น 3 แบบ ใหญ่ๆ คือ แบบแรกเป็นแบบ Thru – beam Scan แบบที่สองเป็นแบบ Reflective Scan และแบบสุดท้ายเป็นแบบ Diffusive Scan ซึ่งทั้งสามแบบจะมีรายละเอียดดังนี้

1. การตรวจจับแบบ Thru-beam Scan การตรวจจับนี้ Emitter และ Receiver จะอยู่คนละด้านกัน โดยสภาวะปกติแสงจาก Transmitter จะตกกระทบ Receiver ตลอดเวลา เมื่อวัตถุที่ต้องการตรวจจับเคลื่อนที่มาตัดลำแสง แสงที่ตกกระทบ Receiver จะหายไป และทำให้ sensor ตรวจจับการมาของวัตถุได้ ดังที่แสดงในรูป

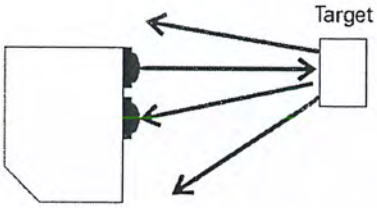


ภาพที่ 2.9 การติดตั้ง Photoelectric Sensor แบบ Thru- Beam

2. การตรวจจับแบบ Reflective Scan การติดตั้งแบบนี้ Emitter จะอยู่ด้านเดียวกันกับ Receiver โดย Emitter จะยิงแสงไปกระทบกับตัวสะท้อน (Reflector) และสะท้อนกลับมาตกกระทบที่ Receiver เมื่อวัตถุเคลื่อนที่เข้ามาตัดลำแสง แสงก็จะไม่สามารถสะท้อนกลับไปตกกระทบที่ Receiver ได้ ทำให้เซนเซอร์สามารถรับรู้ได้ว่ามีวัตถุเคลื่อนที่เข้ามาตัดลำแสง การติดตั้งประเภทนี้แสดงในรูป ข้อสำคัญของการติดตั้งประเภทนี้ วัตถุที่ตัดลำแสงควรเป็นวัตถุที่มีคุณสมบัติในการดูดกลืนแสงสูงและสะท้อนแสงต่ำ เพื่อไม่ให้วัตถุสะท้อนแสงกลับไปตกกระทบที่ Receiver ทำให้เซนเซอร์เกิดการเข้าใจผิดว่าไม่มีวัตถุมาขวางลำแสงได้ แต่ถ้ามีความจำเป็นต้องติดตั้งอาจจะมีการใช้อุปกรณ์กรองแสงแบบต่างๆเข้ามาช่วย ซึ่งจะไม่ขอกล่าวถึงรายละเอียดในที่นี้

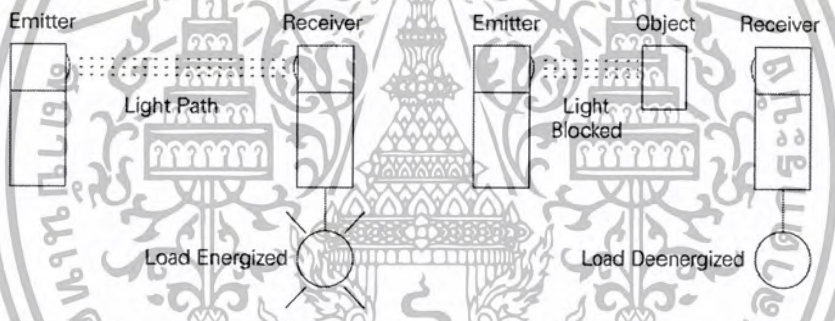
ภาพที่ 2.10 การติดตั้ง Photoelectric Sensor แบบ Reflective Scan

3. การตรวจจับแบบ Diffuse Scan วิธีการนี้ Emitter และ Receiver จะวางอยู่ติดกันเหมือนวิธีที่ผ่านมาแต่ไม่มีแผ่นสะท้อนแสงที่ฝั่งตรงข้าม ซึ่งจะทำให้ไม่มีแสงตกกระทบที่ Receiver เมื่อไม่มีวัตถุผ่านมา และเมื่อมีวัตถุที่ผิวมันพอสมควรผ่านมา มันจะทำหน้าที่สะท้อนแสงบางส่วนกลับไปตกกระทบที่ Receiver ทำให้เซนเซอร์ทราบว่ามีวัตถุเคลื่อนที่ผ่านเข้ามาในบริเวณนั้น การติดตั้งประเภทนี้แสดงในรูป ข้อสำคัญของการตรวจจับนี้ วัตถุควรจะสามารถสะท้อนแสงได้ดีระดับหนึ่ง เพื่อให้แสงที่สะท้อนกลับไปตกกระทบที่ตัวรับแสงมีความเข้มสูงพอที่เซนเซอร์จะตรวจจับได้

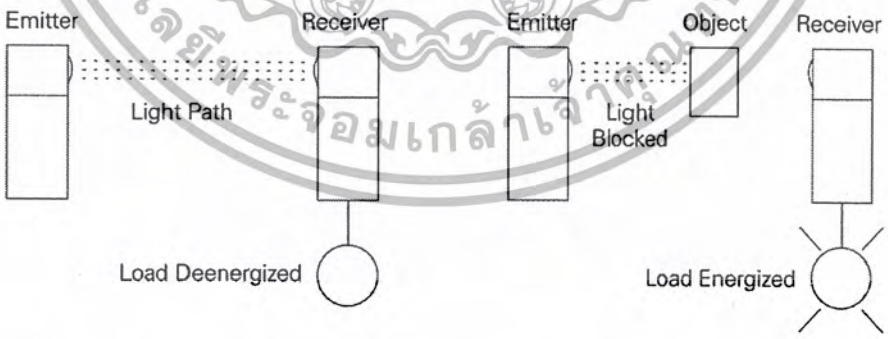


ภาพที่ 2.11 การติดตั้ง Photoelectric Sensor แบบ Diffusive Scan

นอกเหนือจากวิธีการตรวจจับแล้ว เซนเซอร์ประเภทนี้ยังมีลักษณะการทำงานอีกแบบหนึ่ง คือ การกำหนดว่าถ้ามีแสงมาตกกระทบที่ตัวรับแล้วจะให้เซนเซอร์นี้ทำงานหรือไม่ ซึ่งสามารถแบ่งออกได้เป็นสองลักษณะคือ Light On และ Dark On โดยแบบ Light On นี้หากมีแสงมาตกกระทบตัวรับเซนเซอร์นี้จะทำงานหรือส่งสัญญาณออกมา ส่วนแบบ Dark on นี้หากไม่มีแสงมาตกกระทบตัวรับแสงเซนเซอร์นี้ก็จะมีสภาวะเป็น on โดยทั้งสองแบบแสดงในภาพที่ 2.12 และภาพที่ 2.13



ภาพที่ 2.12 Photoelectric Proximity Sensor แบบ Light On ติดตั้งแบบ Thru-Beam



ภาพที่ 2.13 Photoelectric Proximity Sensor แบบ Dark On ติดตั้งแบบThru- beam

2.5 ไมโครคอนโทรลเลอร์ (Microcontroller)

ET-ARM BASE2103 เป็นบอร์ดไมโครคอนโทรเลอร์ในตระกูล ARM7TDMI-S Core ซึ่งเลือกใช้ไมโครคอนโทรเลอร์ 16/32-Bit ขนาด 48 Pin แบบใช้พลังงานต่ำเป็น MCU ประจำบอร์ด ซึ่งบอร์ดนี้เลือกใช้MCU เบอร์ LPC2103/2106 ของ Philips โดยการออกแบบโครงสร้างของบอร์ดนั้นจะเน้นเรื่องการจัดวางบอร์ดให้มีขนาดเล็กเพื่อให้ง่ายต่อการนำไปประยุกต์ใช้งาน โดยได้นำ MCU มาจัดวงจรร่วมกับอุปกรณ์พื้นฐานที่จำเป็นและจัดเรียง Port แบบ 10PIN ของ ETT ตัวบอร์ดใช้ไฟ +5V นอกจากนั้น GPIO ยังสามารถรองรับสัญญาณที่เป็น 5V ได้ มี Connector UART0 (RS-232) จำนวน 2 Port สำหรับทำการ Download Hex File และใช้งานในการสื่อสาร RS232 ในโปรแกรม (Application) ที่เขียนขึ้นเอง

คุณสมบัติของบอร์ดในกรณีใช้ LPC2103

1. ใช้MCUตระกูล ARM7TDMI-S เบอร์LPC2103ของPhilips ซึ่งเป็นMCUขนาด16/32-Bit
2. ใช้ Crystal 19.6608 MHz โดย MCU สามารถประมวลผลด้วยความเร็วสูงสุดที่ 58.9824 MHzเมื่อใช้งานร่วมกับ Phase-Locked Loop (PLL) ภายในตัว MCU เอง
3. รองรับการโปรแกรมแบบ In-System Programming (ISP) และ In-Application Programming (IAP) ผ่านทาง On-Chip Boot-Loader Software ทางพอร์ต RS232-1
4. พอร์ต JTAG 20PIN สำหรับ Real Time Debugging จำนวน 1 พอร์ต
5. พอร์ต LCD มาตรฐาน ETT 14PIN จำนวน 1 พอร์ต
6. พอร์ต GPIO ขนาด 10PIN จำนวน 4 พอร์ต มาตรฐาน ETT
7. หน่วยความจำโปรแกรมแบบ Flash 32KB และ RAM 8KB
8. RTC (Real Time Clock) 32.768KHz พร้อม Battery Backup +3V
9. ใช้แหล่งจ่ายไฟ +5V Power Supply
10. จำนวน GPIO สูงสุดถึง 32 I/O Pins (เฉพาะ GPIO รองรับสัญญาณที่เป็น 5V ได้) ซึ่งขาสัญญาณ GPIO จะมีการใช้งานร่วมกันของ Function อื่นๆอีกดังนี้
   - SPI จำนวน 2 ช่อง
   - I2C จำนวน 2 ช่อง
   - 8-Channel 10 Bit A/D Converter
   - UART แบบ Full-Duplex จำนวน2ช่อง คือ RS232-1, RS232-2 มาตรฐาน 4 Pin ETT
   - Timer 32-bit จำนวน 2 ช่อง (7 Input Capture / 7 Output Compare)
   - Timer 16-bit จำนวน 2 ช่อง (3 Input Capture / 7 Output Compare)
   - Watchdog Timer, PWM Output

2.6 โปรแกรมวิชวลเบสิก (Visual Basic)

โปรแกรม Visual Basic (VB) เป็นโปรแกรมสำหรับพัฒนาโปรแกรมประยุกต์ที่กำลังเป็นที่นิยมใช้อยู่ในปัจจุบัน โปรแกรม Visual Basic เป็นโปรแกรมที่ได้เปลี่ยนรูปแบบการเขียนโปรแกรมใหม่ โดยมีชุดคำสั่งมาสนับสนุนการทำงาน มีเครื่องมือต่าง ๆ ที่เรียกกันว่า คอนโทรล(Control) ไว้สำหรับช่วยในการออกแบบโปรแกรม โดยเน้นการออกแบบหน้าจอแบบกราฟฟิก หรือที่เรียกว่า Graphic User Interface (GUI) ทำให้การจัดรูปแบบหน้าจอเป็นไปได้ง่าย และในการเขียนโปรแกรมนั้นจะเขียนแบบ Event - Driven Programming คือ โปรแกรมจะทำงานก็ต่อเมื่อเหตุการณ์ (Event) เกิดขึ้น ตัวอย่างของเหตุการณ์ได้แก่ ผู้ใช้เลื่อนเมาส์ ผู้ใช้กดปุ่มบนคีย์บอร์ด (keyboard) ผู้ใช้กดปุ่มเมาส์ (mouse) เป็นต้น

เครื่องมือ หรือ คอนโทรล ต่าง ๆ ที่ Visual Basic ได้เตรียมไว้ให้ ไม่ว่าจะเป็น Form Textbox Label ฯลฯ ถือว่าเป็นวัตถุ (Object ในที่นี้ขอใช้คำว่า ออบเจ็กต์) นั่นหมายความว่า ไม่ว่าจะเป็นเครื่องมือใด ๆ ใน Visual Basic จะเป็นออบเจ็กต์ทั้งสิ้น สามารถที่จะควบคุมการทำงาน แก้ไขคุณสมบัติของออบเจ็กต์นั้นได้โดยตรง ในทุกๆ ออบเจ็กต์จะมีคุณสมบัติ (properties) และเมธอด (Methods) ประจำตัว ซึ่งในแต่ละออบเจ็กต์ อาจจะมีคุณสมบัติและเมธอดที่เหมือน หรือต่างกันก็ได้ขึ้นอยู่กับชนิดของออบเจ็กต์

ในการพัฒนาโปรแกรมประยุกต์ด้วย Visual Basic การเขียนโค้ดจะถูกแบ่งออกเป็นส่วนๆ เรียกว่า โพรซีเดอร์ (procedure) แต่ละโพรซีเดอร์จะประกอบไปด้วย ชุดคำสั่งที่พิมพ์เข้าไปแล้ว ทำให้คอนโทรลหรือออบเจ็กต์นั้น ๆ ตอบสนองการกระทำของผู้ใช้ ซึ่งเรียกว่าการเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming-OOP) แต่ตัวภาษา Visual Basic ยังไม่ถือว่าเป็นการเขียนโปรแกรมแบบ OOP อย่างแท้จริง เนื่องจากข้อจำกัดหลายๆ อย่างที่ Visual Basic ไม่สามารถทำได้

โดยโปรแกรม Visual Basic นั้น เป็นโปรแกรมที่ถูกนำมาใช้สำหรับสร้างการติดต่อกับผู้ใช้งาน สำหรับปรับค่าตัวแปรควบคุมต่างๆ แทนการแก้ค่าในโปรแกรมภาษาซีที่ค่อนข้างมีความยุ่งยาก

บทที่ 3

โครงสร้างและการออกแบบ

3.1 พื้นฐานการออกแบบ

จากการศึกษาและรวบรู้ความรู้ทั้งหมด จึงนำมาสู่การออกแบบโครงสร้างของหุ่นยนต์เลียนแบบการว่ายน้ำปลา โดยอ้างอิงมาจากลักษณะโครงสร้างพื้นฐานของปลาตามธรรมชาติ โครงสร้างของหุ่นยนต์นั้น เบื้องต้นควรมีความเรียวบางคล้ายลักษณะของปลาตามธรรมชาติที่ลักษณะของตัวจะค่อนข้างบางเพื่อลดแรงต้านการเคลื่อนที่ที่เกิดจากภายในน้ำมากที่สุด และจำนวนข้อต่อการเคลื่อนที่ของโครงสร้างนั้น ยิ่งมีจำนวนข้อต่อมากเท่าไหร่การเคลื่อนที่ของโครงสร้างก็จะยิ่งมีความราบเรียบมากขึ้นเท่านั้น แต่ด้วยความจำกัดของโครงสร้างภายในโครงงานนี้จึงได้มีการกำหนดให้หุ่นยนต์เลียนแบบการว่ายน้ำของปลามีจำนวนข้อต่อการเคลื่อนที่ 2 ข้อต่อ คือ ช่วงกลางลำตัวและครีบหาง โดยเซอร์โวมอเตอร์เป็นอุปกรณ์ที่จะถูกนำมาใช้ในการทำข้อต่อ เนื่องจากคุณสมบัติที่สามารถควบคุมการหมุนเป็นมุมองศา

โดยการจะเลือกชนิดของเซอร์โวมอเตอร์และขนาดของตัวปลาจะต้องมีการอ้างอิงมาจากแรงบิดสูงสุดที่เซอร์โวมอเตอร์สามารถรองรับได้ โดยมีการคำนวณค่าแรงบิด (Torque) มาจาก

$$T = F * r \qquad (1.1)$$

โดย T ทอร์กหรือค่าแรงบิด (N.m)

F แรงที่กระทำกับจุดหมุน (N)

r รัศมีการหมุน (m)

จากสมการที่ 1.1 ค่าที่มีผลต่อแรงบิดก็คือ F และ r ซึ่ง F หรือแรงที่กระทำกับจุดหมุน ที่จะถูกนำมาคำนวนนี้ก็คือค่าแรงที่เกิดขึ้นเนื่องจากน้ำหนักและค่าแรงโน้มถ่วงของโลก โดยเซอร์โวมอเตอร์ที่ใช้ภายในโครงงานนี้นั้นมีค่าแรงบิดสูงสุดอยู่ที่ 5.5 kgf.m ดังนั้นการออกแบบจึงต้องมีการคำนึงถึงค่ามวลของโครงสร้างที่จะติดตั้งเข้ากับจุดหมุน และระยะในการติดตั้ง ไม่ให้มีค่าเกินกว่าที่เซอร์โวมอเตอร์จะสามารถรับแรงได้

3.1 การควบคุมและรูปแบบการเคลื่อนที่

เซอร์โวมอเตอร์เป็นอุปกรณ์ที่ทำหน้าที่ขับเคลื่อนการเคลื่อนที่ของหุ่นยนต์ โดยเซอร์โวมอเตอร์จะถูกควบคุมการทำงานโดยไมโครคอนโทรลเลอร์ ซึ่งจะทำหน้าที่ประมวลผลโปรแกรมการทำงาน และมีการสั่งการทำงานเซอร์โวมอเตอร์ ให้เกิดการเคลื่อนที่ตามที่ได้กำหนดไว้ โดยการเขียนโปรแกรมเพื่อควบคุมการทำงานของเซอร์โวมอเตอร์นั้น จำเป็นต้องมีการกำหนดรูปแบบการเคลื่อนที่ของหุ่นยนต์ปลาขึ้นมาก่อน โดยรูปแบบการเคลื่อนที่ของหุ่นยนต์ปลานั้นจะอ้างอิงมาจากสมการการเคลื่อนที่ของหุ่นยนต์ปลา อันมีพื้นฐานมาจากลักษณะการเคลื่อนที่ของปลาที่มีตำแหน่งการเคลื่อนที่ใกล้เคียงกับคลื่นซายน์ (Sine wave) ดังนั้นจากคุณลักษณะดังกล่าวจึงได้มีการกำหนดสมการการเคลื่อนที่ของหุ่นยนต์ปลาขึ้น โดยมีพื้นฐานมาจากสมการของคลื่นซายน์

สมการคลื่นซายน์

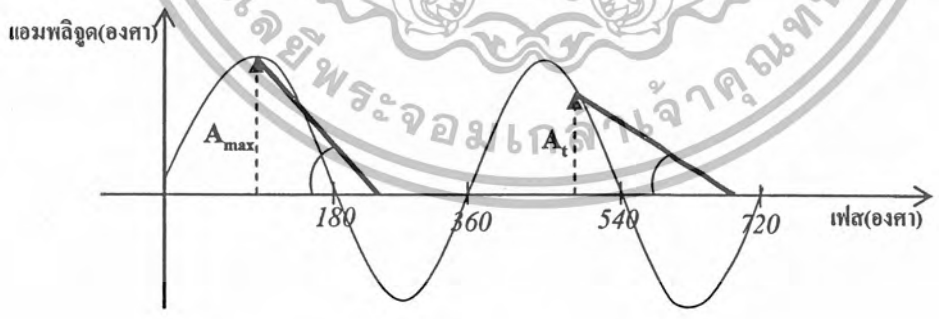
$$A_t = A_m * \sin(2\pi ft) \tag{1.2}$$

โดย A_t แอมพลิจูดของคลื่น ณ เวลาใดๆ *(องศา)*

A_m แอมพลิจูดสูงสุดของคลื่น *(องศา)*

f ความถี่ของคลื่น *(1/วินาที)*

สมการการเคลื่อนที่ของหุ่นยนต์ปลา



ภาพที่ 3.1 ลักษณะการเคลื่อนที่ของหุ่นยนต์ปลา

$$A_t = K_a * A_{max} * \sin(2\pi ft) \tag{1.3}$$

โดย A_t ค่าแอมพลิจูดการเคลื่อนที่ของครีบหางปลา ณ เวลาใดๆ (องศา)

K_a ค่าอัตราส่วนของค่าแอมพลิจูดสูงสุดที่ต้องการกับค่าแอมพลิสูงสุดที่เป็นไปได้

A_{max} แอมพลิจูดการเคลื่อนที่สูงสุดของครีบหางปลา (องศา)

สมการการเคลื่อนที่ของหุ่นยนต์ปลานี้ถูกกำหนดขึ้นมา เพื่อใช้จำลองและควบคุมการเคลื่อนที่ของหุ่นยนต์เลียนแบบการว่ายน้ำของปลา โดยจากลักษณะการเคลื่อนที่ของปลานั้น โครงสร้างของหุ่นยนต์เลียนแบบการว่ายน้ำปลาจะมีการเคลื่อนที่ในลักษณะของมุม(องศา) ดังนั้น ค่าแอมพลิจูดที่ถูกกำหนดภายในสมการนี้ จะถูกพิจารณาในรูปแบบของหน่วยองศา

ค่า K_a : ถูกกำหนดขึ้นมาเพื่อเป็นค่าตัวแปรที่นำมาใช้ปรับค่ามุมการเคลื่อนที่สูงสุดของหุ่นยนต์เลียนแบบการว่ายน้ำของปลา คือ จากการจำลองลักษณะการเคลื่อนที่นั้นทำให้ทราบว่าปลาจะมีการเคลื่อนที่ในลักษณะบิดไป-บิดมาเป็นค่ามุม ดังนั้นในการควบคุมการเคลื่อนที่ของหุ่นยนต์เลียนแบบการว่ายน้ำของปลา จะต้องมีการกำหนดค่าแอมพลิจูด (Amplitude) การเคลื่อนที่สูงสุดที่โครงสร้างสามารถเคลื่อนที่ไปได้ และเมื่อต้องการลดค่ามุมการเคลื่อนที่สูงสุดของโครงสร้างให้มีค่าต่ำลงอยู่ในค่าที่ต้องการ ค่า K_a จะเป็นตัวแปรที่ถูกนำมาใช้ในการปรับค่ามุมการเคลื่อนที่สูงสุดที่ต้องการดังกล่าว แทนการเปลี่ยนค่า A_{max} เพื่อป้องกันการปรับค่าที่ทำให้โครงสร้างเกิดการเคลื่อนที่มากเกินกว่าที่โครงสร้างจะรับได้

จาก $$K_a = \frac{A_{mw}}{A_{max}} \quad (1.4)$$

$$A_{mw} = K_a * A_{max} \quad (1.5)$$

โดย A_{mw} ค่าแอมพลิจูดการเคลื่อนที่สูงสุดที่ต้องการให้โครงสร้างเกิดการเคลื่อนที่ (องศา)

A_{max} ค่าแอมพลิจูดการเคลื่อนที่สูงสุดที่ถูกจำกัดไว้ (องศา)

หมายเหตุ : ค่า A_{mw} จะมีค่าสูงสุดได้ไม่เกินค่า A_{max} ดังนั้นค่า $0 \leq K_a \leq 1$

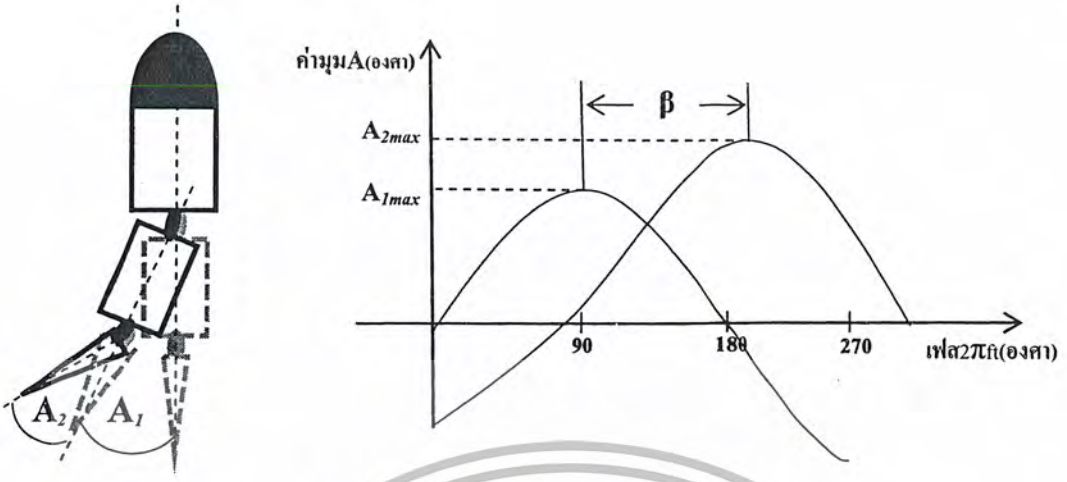
ดังนั้น จะสามารถเขียนสมการการเคลื่อนที่ของหุ่นยนต์ปลาในอีกรูปแบบได้ว่า

$$A_t = A_{mw} * \sin(2\pi ft) \quad (1.6)$$

จากสมการการเคลื่อนที่ของหุ่นยนต์ปลาในสมการที่ *1.3* นั้น เป็นสมการการเคลื่อนที่ของโครงสร้างเพียงส่วนเดียวเท่านั้น แต่ภายในโครงงานนี้กำหนดให้โครงสร้างของหุ่นยนต์ปลามีการเคลื่อนที่ได้ *2* ส่วน คือ ส่วนครีบหางและกลางลำตัว ดังนั้นสมการการเคลื่อนที่จึงต้องมี *2* สมการ ดังนี้

$$A_1 = K_a * A_{1max} * \sin(2\pi ft) \quad (1.7)$$

$$A_2 = K_a * A_{2max} * \sin(2\pi ft) \quad (1.8)$$



ภาพที่ 3.2 เส้นกราฟแสดงตำแหน่งการเคลื่อนที่ของครีบหางและโคนหาง (กลางลำตัว)

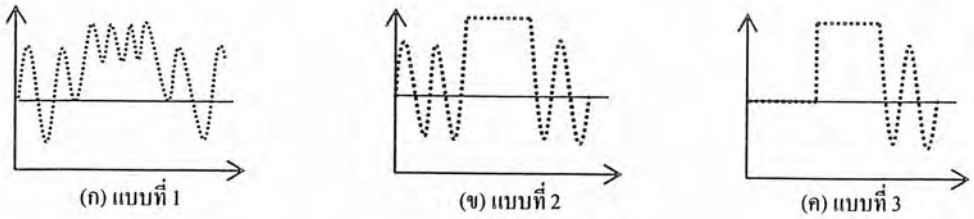
โดย A_1 และ A_2 ค่าแอมพลิจูดการเคลื่อนที่ของจุดข้อต่อโคนหางกับครีบหาง ณ เวลาใดๆ ตามลำดับ(องศา)

A_{1max} และ A_{2max} ค่าแอมพลิจูดการเคลื่อนที่สูงสุดของจุดข้อต่อโคนหางกับครีบหาง ณ เวลาใดๆ ตามลำดับ โดยค่าแอมพลิจูดสูงสุดนี้จะพิจารณามาจากค่ามุมการเคลื่อนที่สูงสุดที่โครงสร้างจะสามารถเคลื่อนที่ไปได้(องศา)

β ค่ามุมต่างเฟสระหว่างการเคลื่อนที่ของจุดข้อต่อโคนหางกับครีบหาง ซึ่งค่ามุมต่างเฟสนี้ จะเป็นค่าที่ถูกกำหนดเพื่อระบุค่าความต่างของเวลาการเคลื่อนที่ว่าต้องการให้มีการเริ่มต้นการเคลื่อนที่ต่างกันเท่าไหร่(องศา)

จากสมการการเคลื่อนที่ทั้งหมดที่กล่าวมาเบื้องต้นนั้น เป็นสมการที่กำหนดรูปแบบการเคลื่อนที่ในลักษณะของการว่ายตรงไปด้านหน้าเท่านั้น ซึ่งยังไม่รวมถึงลักษณะการเคลื่อนที่แบบเลี้ยว

โดยการเลี้ยวของปลาตามธรรมชาตินั้น ไม่ได้ใช้แค่การสะบัดครีบหางเพื่อช่วยในการเคลื่อนที่เท่านั้น ยังต้องประกอบไปด้วยครีบอกและครีบท้องด้วย แต่ภายในโครงงานนี้มีการใช้เพียงการสะบัดของครีบหางที่มาช่วยในการเลี้ยวเป็นหลัก และมีครีบอกมาเป็นตัวช่วยเสริม ซึ่งในส่วนของการควบคุมลักษณะการเลี้ยว ได้มีการแบ่งรูปแบบการเลี้ยวออกเป็นหลายรูปแบบ แต่ภายในปริญญานิพนธ์นี้จะขอกล่าวถึงรูปแบบการเลี้ยวโดยพื้นฐาน 3 ลักษณะ ดังนี้



ภาพที่ 3.3 จำลองการเลี้ยวในลักษณะต่างๆ

แบบที่ 1

ภาพที่ 3.3 (ก) เป็นการแสดงการเลี้ยวในรูปแบบที่ 1 โดยในรูปแบบนี้เมื่อมีการเลี้ยวครีบหางปลาจะมีการสะบัดไป-มาเพียงเฉพาะด้านตรงข้ามกับด้านที่ต้องการเลี้ยวเท่านั้น ซึ่งจากลักษณะการเคลื่อนที่ดังกล่าวจะทำให้โครงสร้างเกิดการผลักตัวเคลื่อนที่เลี้ยวในที่สุด ซึ่งการเคลื่อนที่ในลักษณะนี้จะสามารถปรับค่ารัศมีและความเร็วการเลี้ยวได้หลากหลาย

แบบที่ 2

ภาพที่ 3.3 (ข) เป็นการแสดงการเลี้ยวในรูปแบบที่ 2 ซึ่งในลักษณะนี้นั้นเมื่อต้องการเลี้ยวในช่วงแรกหุ่นยนต์ปลาจะต้องว่ายตรงมาก่อน จากนั้นจึงมีการสะบัดหางไปในทิศทางด้านตรงข้ามกับด้านที่ต้องการเลี้ยว และหยุดการเคลื่อนที่ไว้ ณ ตำแหน่งดังกล่าว แล้วปล่อยให้โครงสร้างเกิดการเลี้ยวด้วยแรงของกระแสน้ำ ซึ่งการเลี้ยวในลักษณะนี้นั้นจะให้รัศมีในการเลี้ยวน้อยกว่าในแบบที่ 1

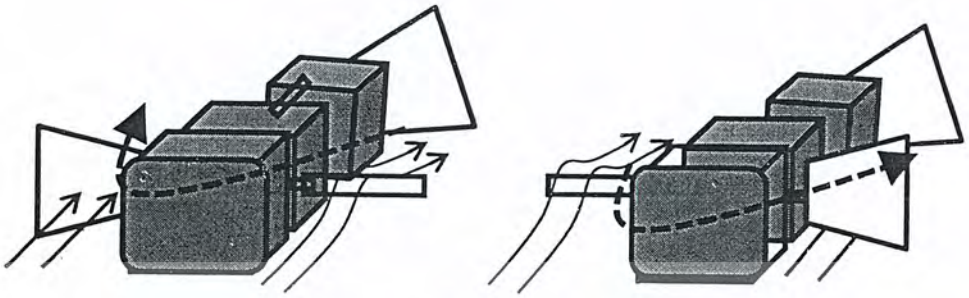
แบบที่ 3

ภาพที่ 3.3 (ค) เป็นการแสดงการเลี้ยวในรูปแบบที่ 3 ซึ่งในลักษณะนี้นั้นเมื่อต้องการเลี้ยวหุ่นยนต์ปลาจะอยู่ในสภาวะที่หยุดนิ่ง และมีการสะบัดหางอย่างรวดเร็วในเวลาต่อมา ซึ่งแรงเฉื่อยและแรงเสียดทานที่เกิดขึ้นจากการสะบัดจะส่งผลให้หุ่นยนต์ปลาเกิดการเคลื่อนที่เลี้ยว

โดยการเลี้ยวในรูปแบบที่ 3 นี้ ถือว่าเป็นรูปแบบการเลี้ยวที่ให้คุณลักษณะการเลี้ยวที่ดีเยี่ยม เพราะสามารถที่จะเลี้ยวจากสภาวะหยุดนิ่ง และให้ค่ารัศมีการเลี้ยวน้อยที่สุดจากแบบทั้งหมดที่กล่าวถึง แต่ไม่ว่าอย่างไรก็ตามการเลี้ยวในลักษณะนี้นั้น เป็นลักษณะการเลี้ยวที่ยากต่อการควบคุมความเร็วและมุมในการเลี้ยว นอกเหนือจากนั้นการจะเลี้ยวในลักษณะนี้มีความจำเป็นที่ต้องใช้แหล่งพลังงานที่ให้แรงบิดที่มีค่าสูงในการจะสะบัดครีบหาง

จากรูปแบบการเลี้ยวทั้งหมดที่กล่าวมานั้น ภายในปริญญานิพนธ์นี้ได้มีการกำหนดให้หุ่นยนต์เลียนแบบการว่ายน้ำของปลามีรูปแบบการเลี้ยว ตามรูปแบบที่ 1 เพราะสามารถควบคุมลักษณะการเลี้ยวได้หลากหลาย แต่จากที่ได้กล่าวถึงการเลี้ยวในรูปแบบที่ 1 ให้รัศมีการเลี้ยวที่

ค่อนข้างสูงกว่ารูปแบบอื่นดังนั้นจึงได้มีการนำครีบอกมาเป็นตัวช่วยเสริมให้การเลี้ยวมีประสิทธิภาพมากยิ่งขึ้น



(ก) (ข)

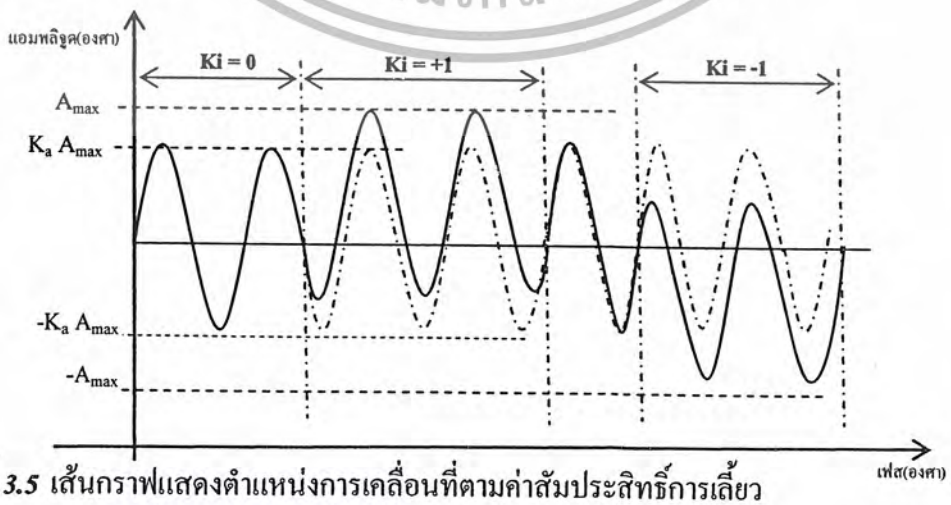
ภาพที่ 3.4 รูปแบบการเลี้ยวด้วยครีบอก

โดยครีบอกของหุ่นยนต์เลียนแบบการว่ายน้ำปลานั้นจะสามารถเคลื่อนที่อยู่ ณ สองตำแหน่ง คือในแนวราบกับแนวตั้ง ในสภาวะการเคลื่อนที่ปกตินั้นครีบอกทั้งสองด้านจะอยู่ในตำแหน่งของแนวราบ แต่เมื่อต้องการเคลื่อนที่เลี้ยวในทิศทางใด ครีบอกของทางด้านเดียวกับทิศทางการเลี้ยวนั้นจะถูกตั้งขึ้นต้านการเคลื่อนที่ ซึ่งจะส่งผลให้ด้านทั้งสองของตัวปลาเกิดการเคลื่อนที่ได้ไม่เท่ากัน จึงส่งผลช่วยให้เกิดการเคลื่อนที่เลี้ยวมากขึ้น

จากรูปแบบการเคลื่อนที่ที่ต้องมีการเลี้ยวมาเกี่ยวข้อง สมการการเคลื่อนที่ของหุ่นยนต์ปลาจึงต้องมีการเพิ่มตัวแปรที่เกี่ยวข้องกับการเลี้ยว ซึ่งจะได้สมการการเคลื่อนที่ของหุ่นยนต์ปลา ดังนี้

$$A_t = K_a * A_{max} * \sin(2\pi ft) - (1- K_a) * K_i * A_{max} \qquad (1.9)$$

โดย K_i ค่าสัมประสิทธิ์การเลี้ยว โดยจะมีค่าอยู่ในช่วง *-1* ถึง *1*



ภาพที่ *3.5* เส้นกราฟแสดงตำแหน่งการเคลื่อนที่ตามค่าสัมประสิทธิ์การเลี้ยว

จากกราฟภาพที่ *3.5* เมื่อหุ่นยนต์ปลามีการว่ายน้ำตรงปกติ ค่า $K_i = 0$ นั่นก็คือ $A_t = K_a * A_{max} * sin(2\pi ft)$ แต่เมื่อค่า $K_i = +1$ สมการจะถูกบวกเพิ่มด้วยค่ามุม $(1- K_a) * A_{max}$ ซึ่งจะทำให้ตำแหน่งการเคลื่อนที่มีการเคลื่อนที่ไปทางด้านบวกมากขึ้นและมีการเคลื่อนที่ไปทางลบน้อยลง ซึ่งจะส่งผลให้หุ่นยนต์ปลาจะเกิดการเลี้ยวมาทางด้านบวก ในทางตรงข้ามหากค่า $K_i = -1$ ตำแหน่งการเคลื่อนที่มีการเคลื่อนที่ไปทางด้านลบมากขึ้นและมีการเคลื่อนที่ไปทางบวกน้อยลง และส่งผลให้หุ่นยนต์ปลาจะเกิดการเลี้ยวมาทางด้านลบนั่นเอง

จากที่ได้กล่าวมาทั้งหมดจะสามารถสรุปเป็นสมการการเคลื่อนที่ของหุ่นยนต์ปลา ทั้งในส่วนของครีบหางและกลางลำตัว ได้ดังนี้

กลางลำตัว; $A_1 = K_a * A_{1max} * \sin(2\pi ft) - (1- K_a) * K_i * A_{1max}$ (2.0)

ครีบหาง ; $A_2 = K_a * A_{2max} * \sin(2\pi ft - \beta) - (1- K_a) * K_i * A_{2max}$ (2.1)

3.3 โครงสร้างหุ่นยนต์

โครงสร้างของหุ่นยนต์ถูกออกแบบให้เป็นลักษณะของกล่อง ที่มีการเชื่อมต่อกันเป็นข้อต่อที่มีอุปกรณ์ทำหน้าที่เป็นจุดหมุนคือ เซอร์โวมอเตอร์ โดยอะคริลิคเป็นวัสดุที่ถูกนำมาใช้ในการทำกล่องซึ่งจะนำมาใช้ในการบรรจุอุปกรณ์ไฟฟ้าต่างๆที่เกี่ยวข้องกับการทำงานของหุ่นยนต์ ซึ่งกล่องอะคริลิคที่นำมาใช้ในการบรรจุนั้นจะต้องมีความสามารถในการกันน้ำได้

โครงสร้างของหุ่นยนต์นั้นจะประกอบไปด้วย

1. ไมโครคอนโทรลเลอร์ ชนิด ARM7 รุ่น ET-ARM BASE2103 ทำหน้าที่เป็นหน่วยประมวลผลซึ่งหากเปรียบเทียบก็คือเป็นสมองของหุ่นยนต์ ที่จะคอยควบคุมและสั่งการทำงานในส่วนต่างๆ โดยจะถูกติดตั้งไว้ที่ส่วนหัวของโครงสร้าง

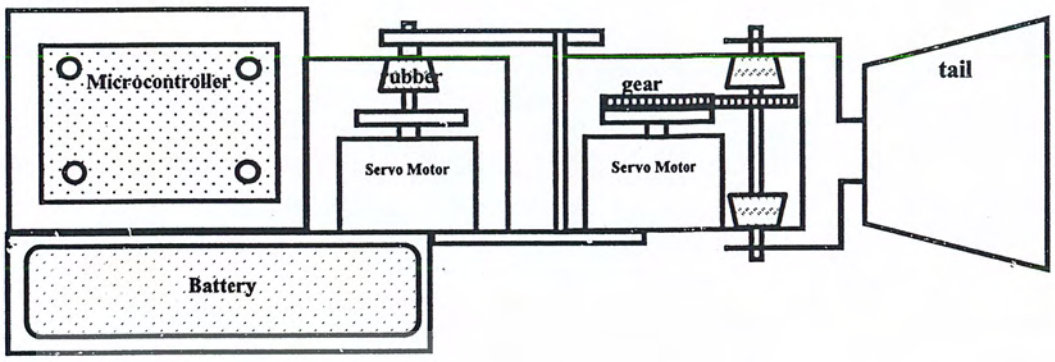
2. เซอร์โวมอเตอร์ ทำหน้าที่เป็นจุดหมุน ณ ส่วนกลางลำตัว ครีบหาง และครีบอกทั้งสองด้าน

3. เซนเซอร์(Photoelectric) ทำหน้าที่เป็นตัวตรวจจับสิ่งกีดขวางการเคลื่อนที่ของหุ่นยนต์ โดยเมื่อเจอสิ่งกีดขวาง เซนเซอร์จะมีการส่งสัญญาณมายังไมโครคอนโทรลเลอร์ เพื่อสั่งให้เกิดการเลี้ยวต่อไป โดยจะถูกติดตั้งไว้ที่ด้านข้างและด้านหน้าส่วนหัวของหุ่นยนต์

4. แบตเตอรี่ (Battery) ทำหน้าที่เป็นแหล่งจ่ายพลังงานแก่อุปกรณ์ต่างๆ ไม่ว่าจะเป็นไมโครคอนโทรลเลอร์ เซอร์โวมอเตอร์ และเซนเซอร์ แบตเตอรี่ที่เลือกใช้เป็นแบตเตอรี่ชนิดลิเที่ยมโพลิเมอร์ ขนาด 12 โวลต์

ตารางที่ 3.1 รายละเอียดส่วนประกอบของโครงสร้าง

ไมโครคอนโทรลเลอร์	ARM7 ET-ARM BASE2103
จำนวนข้อต่อ	2
เซอร์โวมอเตอร์	
จำนวน	2
แรงบิดสูงสุด	5.5 kg-cm (4.8 โวลต์) , 6.5 kg-cm (6 โวลต์)
น้ำหนัก	38 กรัม
มินิเซอร์โวมอเตอร์	
จำนวน	2
แรงบิดสูงสุด	1.5 kg-cm
น้ำหนัก	10 กรัม
เซนเซอร์	โฟโต้เซนเซอร์ ตรวจจับแบบ Diffuse Scan ทำงานแบบ Dark On
แบตเตอรี่	ลิเทียมโพลิเมอร์ ขนาด 12 โวลต์



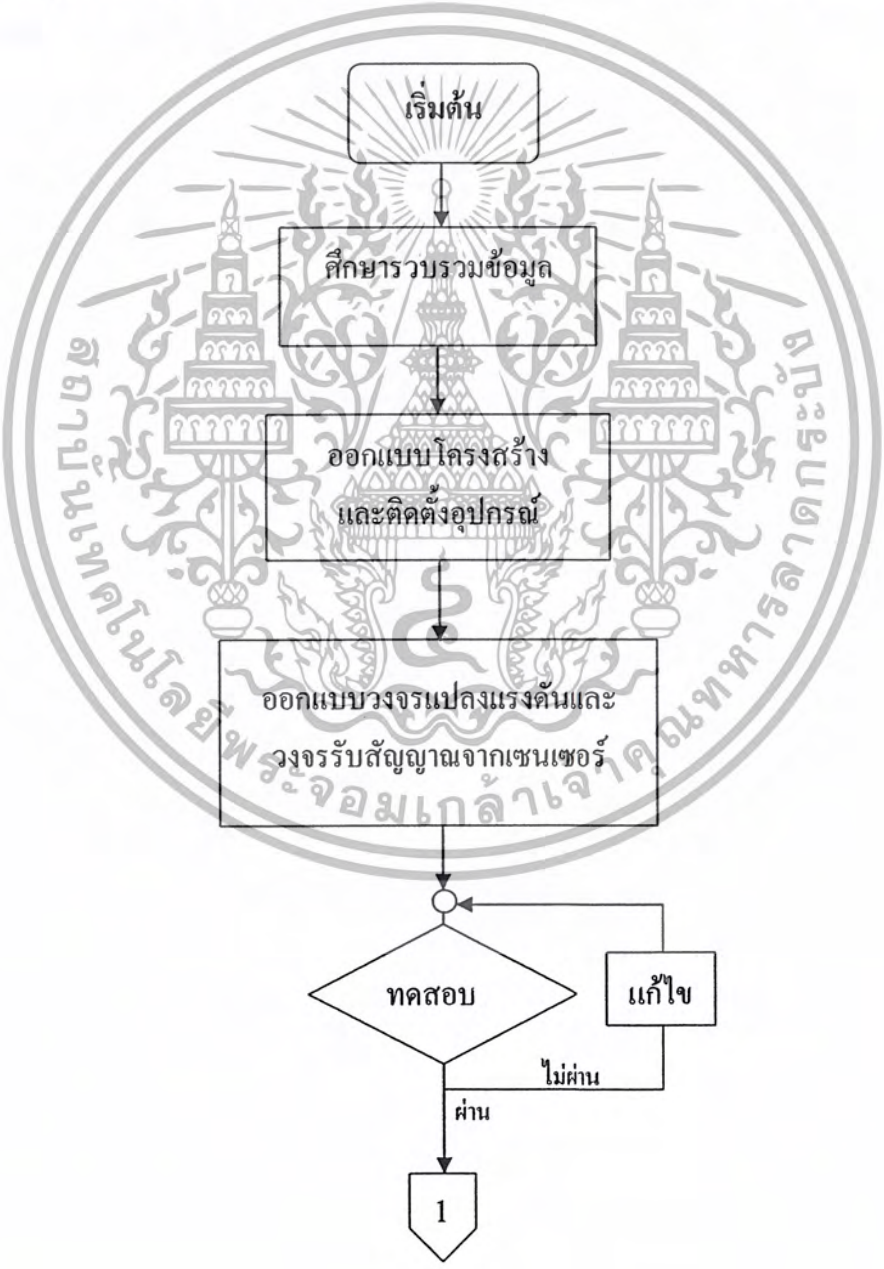
ภาพที่ 3.6 โครงสร้างของหุ่นยนต์เลียนแบบการว่ายน้ำของปลา

โดยข้อต่อการเคลื่อนที่นั้นจะใช้เป็นลักษณะของการใช้แกนหมุนเพื่อลดปัญหาการรั่วซึมอันเนื่องมาจาการเคลื่อนที่ โดยจะต้องมีการนำยางกันรั่วมาติดตั้ง ณ ตำแหน่งช่องที่แกนดังกล่าวสอดผ่านทุกจุด เพื่อป้องกันการรั่วซึม

บทที่ 4

ขั้นตอนและวิธีดำเนินการ

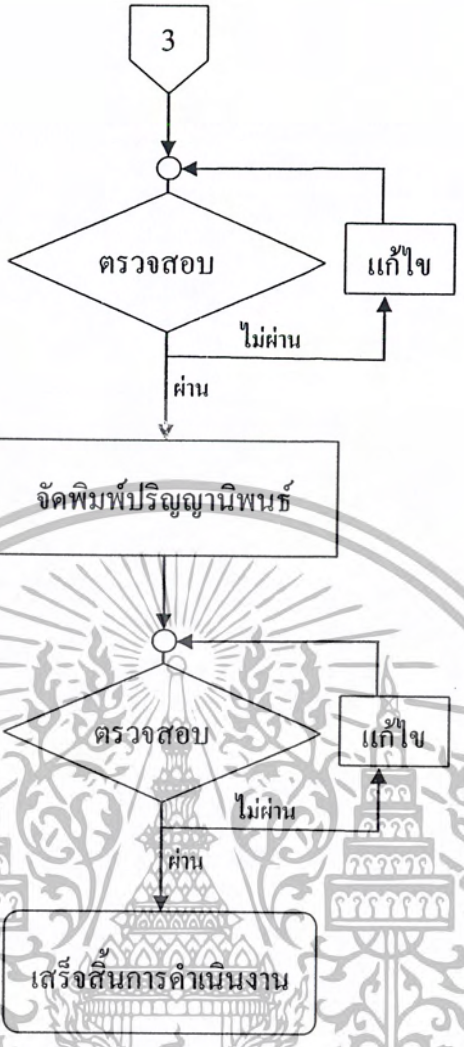
ในการดำเนินการนั้นเราจำเป็นที่จะต้องมีการวางแผนการทำงานเพื่อตรวจสอบการทำงานว่าการทำงานในแต่ละช่วง เวลาเท่าไหร่จึงจะมีความเหมาะสม และนำมาตรวจสอบการทำงานว่ามีความช้าหรือเร็วเกินไปหรือไม่ เพื่อจะได้มีการจัดการเวลาการทำงานที่เหมาะสมและทำให้งานที่ได้ออกมานั้นมีความสมบูรณ์มากที่สุด ในการดำเนินงานของโครงงานมีโฟลว์ชาร์ตดังต่อไปนี้



ภาพที่ 4.1 Flow Chart แสดงการทำงาน

ภาพที่ 4.2 Flow Chart แสดงการทำงาน (ต่อ)

ภาพที่ 4.3 Flow Chart แสดงการทำงาน (ต่อ)



ภาพที่ 4.4 Flow Chart แสดงการทำงาน (ต่อ)

ขั้นตอนในการดำเนินงาน สามารถแบ่งเป็นขั้นตอนต่างๆ ได้ดังนี้

4.1 ศึกษาและรวบรวมข้อมูล

4.2 ออกแบบโครงสร้างและวงจร

4.3 ศึกษาและเขียนโปรแกรมสร้าง PWM

4.4 ศึกษาและเขียนโปรแกรมสั่งการทำงานด้วย Visual Basic

4.5 ประกอบทุกส่วนเข้าด้วยกันและทดลอง

4.6 จัดทำปริญญานิพนธ์

4.1 ศึกษาและรวบรวมข้อมูล

ผู้จัดทำได้มีการศึกษาข้อมูลที่มีความเกี่ยวข้องกับโครงงานในส่วนต่างๆ โดยเรียงลำดับดังต่อไปนี้

- ความรู้พื้นฐานในการสร้างหุ่นยนต์
- ลักษณะการเคลื่อนที่ของปลาตามธรรมชาติ
- สมการการเคลื่อนที่ของหุ่นยนต์ปลา
- เซอร์โวมอเตอร์ (Servo Motor)
- เซนเซอร์(Photoelectric sensor)
- ไมโครคอนโทรลเลอร์ (Microcontroller)
- โปรแกรมวิชวลเบสิก (Visual Basic)

4.2 ออกแบบโครงสร้างและวงจร

ก่อนการจัดทำโครงสร้างของหุ่นยนต์นั้น ต้องมีการออกแบบโครงสร้างของหุ่นยนต์และวงจรที่ควบคุมการทำงานของอุปกรณ์ต่างๆภายในหุ่นยนต์ก่อน โดยในส่วนของวงจรนั้นภายในวงจรนี้ประกอบไปด้วยวงจร 2 ส่วน คือ วงจรเซนเซอร์ ที่ทำหน้าที่จ่ายพลังงานและรับค่าสัญญาณจากเซนเซอร์มายังไมโครคอนโทรลเลอร์ กับ วงจรปรับค่าแรงดัน ที่ทำหน้าที่ปรับค่าแรงดันให้อยู่ในค่าที่เหมาะสมต่อการใช้งาน เมื่อขั้นตอนของการออกแบบเสร็จสิ้นก็ต้องมีการตรวจสอบถึงความถูกต้องก่อน จากนั้นจึงทำการติดตั้งอุปกรณ์ตามแบบต่อไป

4.3 ศึกษาและเขียนโปรแกรมสร้าง PWM

สัญญาณ PWM (Pulse Width Modulation) เป็นสัญญาณที่ใช้ในการควบคุมการทำงานของเซอร์โวมอเตอร์ โดยเราจะต้องทำการเขียนโปรแกรมสร้างสัญญาณ PWM สำหรับไมโครคอนโทรลเลอร์ขึ้นมา ไมโครคอนโทรลเลอร์จะทำการสร้างสัญญาณพัลส์ออกมาควบคุม

เซอร์โวมอเตอร์ตามโปรแกรมดังกล่าว ซึ่งก่อนจะทำการจะเขียนโปรแกรมเหล่านี้ได้นั้น จึงจำเป็นต้องมีการศึกษาถึงองค์ประกอบของการเขียนโปรแกรมก่อน เช่น การเขียนภาษาซี การเขียนโปรแกรมควบคุมเซอร์โวมอเตอร์ เป็นต้น

ซึ่งหุ่นยนต์นั้นประกอบไปด้วยเซอร์โวมอเตอร์ 4 ตัว ซึ่งการเคลื่อนที่ของหุ่นยนต์เองก็มีอยู่หลายลักษณะ ซึ่งก็ทำให้เซอร์โวมอเตอร์แต่ละตัวทำงานต่างกันออกไป ดังนั้นการเขียนโปรแกรมนอกจากจะสามารถสั่งการทำงานของเซอร์โวมอเตอร์ได้แล้ว จำเป็นต้องสามารถเชื่อมโยงสั่งการทำงานของเซอร์โวมอเตอร์แต่ละตัวให้มีความสัมพันธ์กันด้วย

4.4 ศึกษาและเขียนโปรแกรมสั่งการทำงานด้วย Visual Basic

หุ่นยนต์เลียนแบบการว่ายน้ำของปลาเป็นหุ่นยนต์ที่ต้องมีการเคลื่อนที่ภายใต้น้ำตลอดเวลา โดยในการทดลองการควบคุมการทำงานของหุ่นยนต์ด้วยค่าตัวแปรต่างๆ นั้น จำเป็นต้องมีการลงโปรแกรมเพื่อปรับค่าให้แก่ไมโครคอนโทรลเลอร์ ซึ่งด้วยโครงสร้างของหุ่นยนต์ที่ต้องมีการปิดผนึกเพื่อกันน้ำนั้นจึงถือว่าค่อนข้างลำบากที่จะต้องแกะกล่องเพื่อลงโปรแกรมเข้าไปใหม่ ดังนั้นจึงได้มีการศึกษาการทำงานของโปรแกรม Visual Basic ขึ้น เพื่อสร้าง GUI (Graphics User Interface) สำหรับการเปลี่ยนตัวแปรต่างในโปรแกรม โดยไม่จำเป็นต้องลงโปรแกรมเข้าไปใหม่

4.5 ประกอบทุกส่วนเข้าด้วยกันและทำการทดลอง

ประกอบการทำงานทุกส่วนเข้าด้วยกันทั้งส่วนของโครงสร้างและโปรแกรม ทดลองผลการทำงานให้หุ่นยนต์มีการเคลื่อนที่ในน้ำ ทำการปรับค่าตัวแปรต่างๆ โดยพิจารณาหาค่ารัศมีการเลี้ยวให้มีค่ารัศมีการเลี้ยวน้อยที่สุด สังเกตพร้อมเก็บบันทึกผล

4.6 จัดทำปริญญานิพนธ์

บทที่ 5
การทดลองและผลการทดลอง

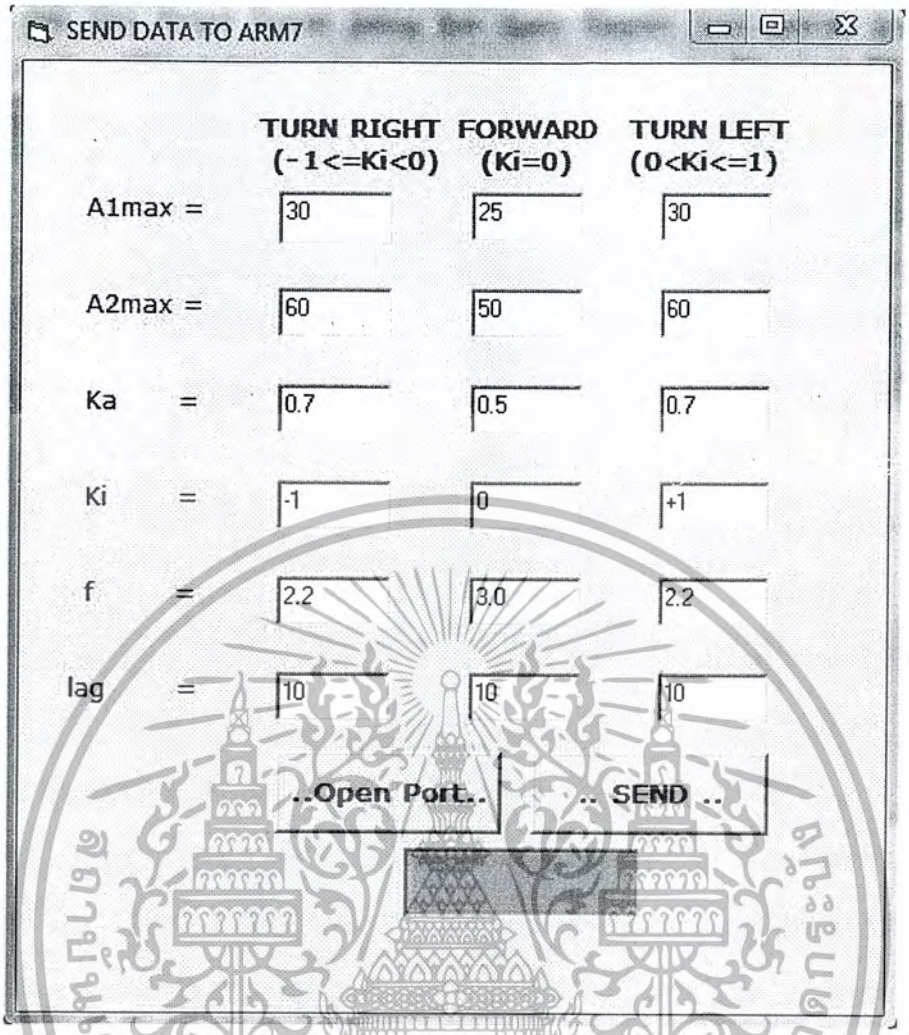
5.1 กล่าวนำ

จากที่ได้กล่าวถึงสมการการเคลื่อนที่ของปลา ที่เป็นสมการที่กำหนดลักษณะการเคลื่อนที่ของหุ่นยนต์ปลา โดยมีการกล่าวถึงความสัมพันธ์ของตัวแปรต่างๆ ที่มีผลต่อการเคลื่อนที่ ซึ่งหากเรามีการกำหนดค่าตัวแปรที่ต่างออกไป ลักษณะการเคลื่อนที่ก็จะต่างออกไปตาม

จากหลักการ การให้หุ่นยนต์ปลาเกิดการเคลื่อนที่ไปข้างหน้าเป็นเส้นตรงนั้น ตัวแปรK_i จะไม่มีผลต่อการเคลื่อนที่ในลักษณะนี้ คือ ต้องกำหนดให้ K_i มีค่าเท่ากับ 0 ดังนั้นตัวแปรที่เข้ามาเกี่ยวข้องก็จะมีเฉพาะ K_a และ f เท่านั้น และค่าผลลัพธ์ของการปรับค่าตัวแปรทั้งสองในกรณีนี้ ก็คือจะมีผลต่อความเร็วการเคลื่อนที่เท่านั้น ซึ่งโดยจุดประสงค์การจัดทำหุ่นยนต์ปลาภายใต้โครงงานนี้นั้น ไม่ได้มีการกำหนดให้ปลาต้องมีการเคลื่อนที่ด้วยความเร็ว แต่เพียงต้องสามารถเคลื่อนที่ไปข้างหน้าได้และเคลื่อนที่เลี้ยวหลบสิ่งกีดขวางได้นั่นเอง ซึ่งในส่วนของการเคลื่อนที่ไปข้างหน้าได้นั้น จะเป็นส่วนของการถ่วงน้ำหนักทั้งสองด้านของตัวหุ่นยนต์ปลาให้มีความสมดุล เมื่อหุ่นยนต์มีความสมดุลแล้วเพียงมีการบังคับให้เกิดการสะบัดการเคลื่อนที่ทั้งสองด้านที่มุมเท่ากัน หุ่นยนต์ก็จะสามารถเคลื่อนที่ไปข้างหน้าได้นั่นเอง ดังนั้นกรณีการเคลื่อนที่ที่ต้องทำการทดลองเพื่อหาลักษณะการเคลื่อนที่ที่เหมาะสมก็คือ กรณีการเลี้ยว ว่าตัวแปรการเคลื่อนแต่ละตัวนั้น ควรมีค่าตัวแปรในสมการการเคลื่อนที่เท่าไหร่จึงจะสามารถทำให้โครงสร้างเลี้ยวพ้นสิ่งกีดขวางได้

5.2 ขั้นตอนการทดลอง

1. สร้างหน้าจอ Graphics User Interface โดยโปรแกรมวิชวลเบสิก(Visual Basic) เพื่อการปรับค่าตัวแปรต่างๆ โดยไม่จำเป็นต้องเปิดกล่องบรรจุไมโครคอนโทรลเลอร์เพื่อลงโปรแกรมลงไปใหม่ คือ ไมโครคอนโทรลเลอร์ที่บรรจุภายในโครงสร้างของหุ่นยนต์ปลา จะถูกโยงสาย RS-232 เชื่อมต่อมายังคอมพิวเตอร์ ซึ่งจะทำการส่งข้อมูลจากหน้าจอ Graphics User Interface ส่งมาเปลี่ยนค่าในโปรแกรมควบคุมการเคลื่อนที่ของหุ่นยนต์ปลา



ภาพที่ 5.1 หน้าจอ Graphics User Interface

2. ทดลองเปลี่ยนค่าตัวแปร สังเกตและบันทึกผล โดยนำโครงสร้างของหุ่นยนต์ปลาลงในน้ำ แล้วทดลองปรับค่าตัวแปรต่างๆ บนหน้าจอ Graphics User Interface โดยมีลักษณะการปรับดังนี้

- ปรับค่า K_a และ K_i โดยให้ $A_{1max} = 0$, $A_{2max} = 60$ องศา สังเกตและบันทึกผลค่ารัศมีการเลี้ยว

- ปรับค่า K_a และ K_i โดยให้ $A_{1max} = 60$, $A_{2max} = 75$ องศา สังเกตและบันทึกผลค่ารัศมีการเลี้ยว

- ปรับค่า K_a และ f โดยให้ $A_{1max} = 60$, $A_{2max} = 75$ องศา สังเกตและบันทึกผลค่ารัศมีการเลี้ยว

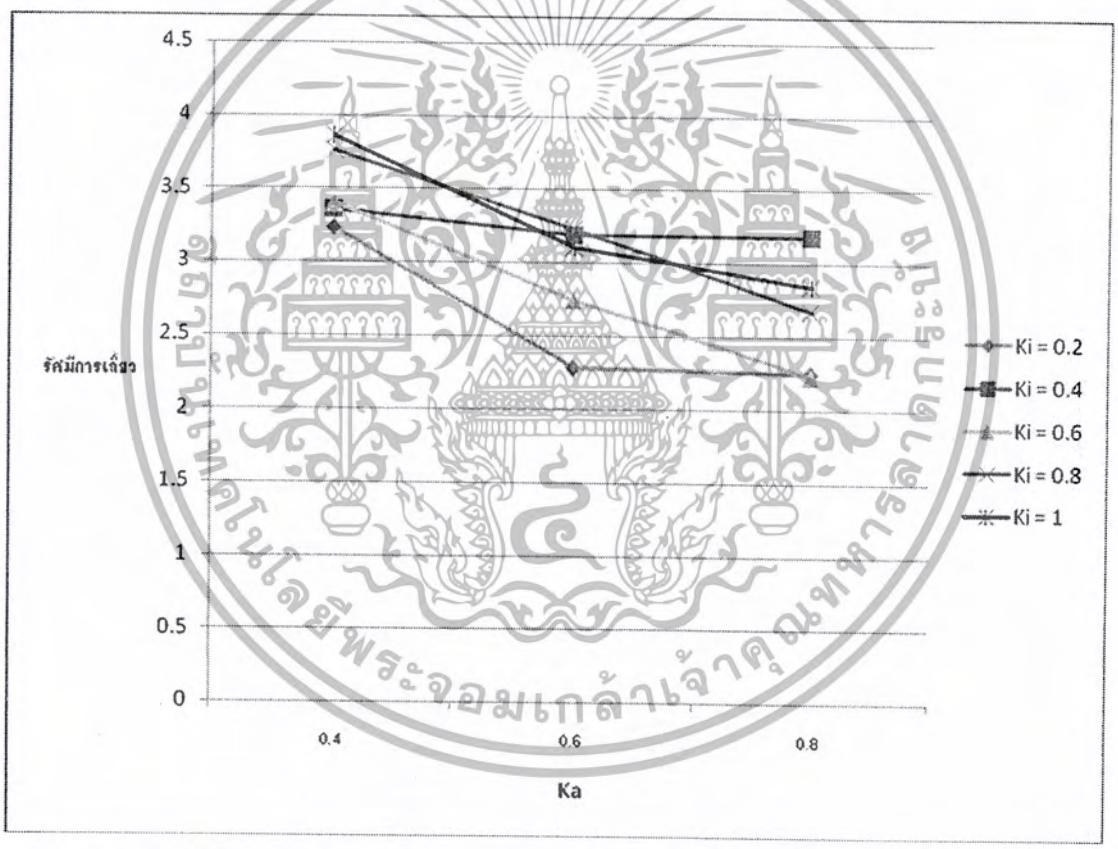
5.3 ผลการทดลอง

5.3.1 การทดลองครั้งที่ 1

เมื่อ A_{1max} = 0 องศา , A_{2max} = 60 องศา , f = 1.3 เฮิร์ต และ β = 10 องศา ปรับค่า K_a และ K_i

ตารางที่ 5.1 ผลการทดลองครั้งที่ 1

Ka \ Ki	0.2	0.4	0.6	0.8	1
0.4	3.23 นิ้ว	3.36 นิ้ว	3.40 นิ้ว	3.76 นิ้ว	3.86 นิ้ว
0.6	2.28 นิ้ว	3.19 นิ้ว	2.74 นิ้ว	3.23 นิ้ว	3.1 นิ้ว
0.8	2.25 นิ้ว	3.18 นิ้ว	2.23 นิ้ว	2.67 นิ้ว	2.84 นิ้ว



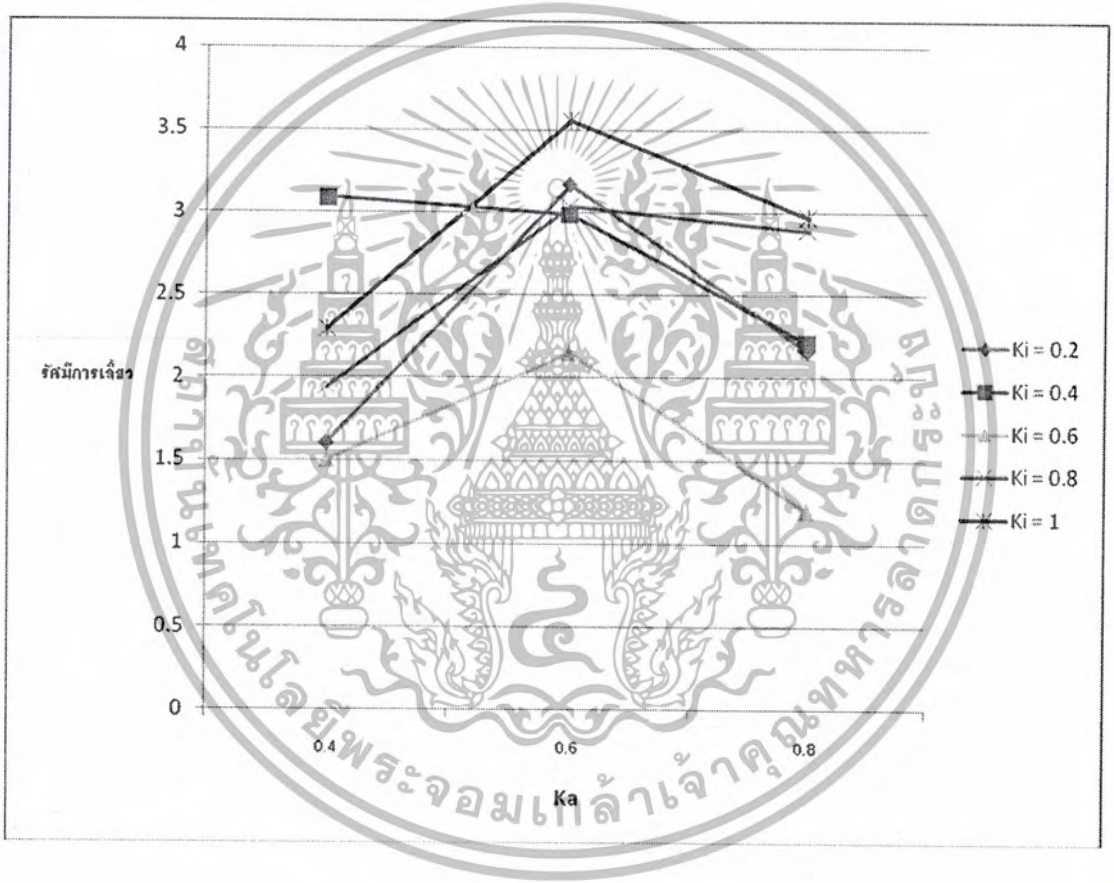
ภาพที่ 5.2 กราฟแสดงผลการทดลองครั้งที่ 1

5.3.2 การทดลองครั้งที่ 2

เมื่อ A_{1max} = 60 องศา, A_{2max} = 75 องศา, f = 1.3 เฮิร์ต และ β = 10 องศา ปรับค่า K_a และ K_i

ตารางที่ 5.2 ผลการทดลองครั้งที่ 2

Ka \ Ki	0.2	0.4	0.6	0.8	1
0.4	1.00 นิ้ว	3.09 นิ้ว	0.96 นิ้ว	1.94 นิ้ว	2.29 นิ้ว
0.6	3.17 นิ้ว	2.99 นิ้ว	2.15 นิ้ว	3.04 นิ้ว	3.56 นิ้ว
0.8	2.09 นิ้ว	2.21 นิ้ว	3.59 นิ้ว	2.89 นิ้ว	2.97 นิ้ว



ภาพที่ 5.3 กราฟแสดงผลการทดลองครั้งที่ 2

5.3.3 การทดลองครั้งที่ 3

เมื่อ A_{1max} = 60 องศา, A_{2max} = 75 องศา, K_i = 1.0 และ β = 10 องศา ปรับค่า K_a และ f

ตารางที่ 5.3 ผลการทดลองครั้งที่ 3

Ka \ f	1.2	1.3	1.4	1.5
0.4	1.94 นิ้ว	2.59 นิ้ว	1.9 นิ้ว	1.42 นิ้ว
0.6	2.81 นิ้ว	2.38 นิ้ว	1.85 นิ้ว	1.7 นิ้ว
0.8	3.88 นิ้ว	2.4 นิ้ว	1.3 นิ้ว	1.32 นิ้ว

ภาพที่ 5.4 กราฟแสดงผลการทดลองครั้งที่ 3

บทที่ 6
บทสรุป

6.1 สรุปผลการทดลอง

6.1.1 สรุปผลการทดลองครั้งที่ 1

ผลการทดลองครั้งที่ 1 เป็นผลจากการทดลองที่กำหนดให้ข้อโคนหางไม่มีการเคลื่อนที่ และทำการเปลี่ยนค่า K_i ณ ค่าK_a ต่างๆ โดยกำหนดให้ค่าตัวแปรอื่นๆคงที่ที่ $A_{1max} = 0$ องศา, $A_{2max} = 45$ องศา, $f = 1.3$ เฮิร์ต และ $\beta = 10$ องศา

ซึ่งจากผลการทดลองจะเห็นว่า ที่ค่า K_i น้อย K_a มาก ให้ค่ารัศมีการเลี้ยวที่น้อยกว่า โดยค่ารัศมีการเลี้ยวที่น้อยที่สุดจะอยู่ที่ ค่า $K_a = 0.6$ และ $K_i = 0.2$ ซึ่งจะให้ค่ารัศมีการเลี้ยวอยู่ที่ 2.28 นิ้ว

6.1.2 สรุปผลการทดลองครั้งที่ 2

ผลการทดลองครั้งที่ 2 เป็นผลจากการทดลองที่กำหนดให้มีการเคลื่อนที่ทั้งสองข้อต่อ และทำการเปลี่ยนค่า K_i ณ ค่า K_a ต่างๆ โดยกำหนดให้ค่าตัวแปรอื่นๆคงที่ที่ $A_{1max} = 60$ องศา, $A_{2max} = 75$ องศา , $f = 1.3$ เฮิร์ต และ $\beta = 10$ องศา

ซึ่งจากผลการทดลองจะได้ว่า โดยเฉลี่ยที่ค่า K_i น้อย ให้ค่ารัศมีการเลี้ยวที่น้อยกว่าเช่นกัน แต่เมื่อข้อโคนหางมีการเคลื่อนที่ด้วย จะให้ค่ารัศมีการเลี้ยวที่น้อยกว่าผลการทดลองครั้งก่อน คือ ค่ารัศมีการเลี้ยวที่น้อยที่สุดจะอยู่ที่ค่า $K_a = 0.8$ และ $K_i = 0.6$ ซึ่งจะให้ค่ารัศมีการเลี้ยวอยู่ที่ 1.2 นิ้ว

6.1.3 สรุปผลการทดลองครั้งที่ 3

ผลการทดลองครั้งที่ 3 เป็นการเปลี่ยนค่า f ณ ค่า K_a ต่างๆ โดยกำหนดให้ค่าตัวแปรอื่นๆ คงที่ที่ $A_{1max} = 60$ องศา , $A_{2max} = 75$ องศา , $K_i = 1$ และ $\beta = 10$ องศา

ซึ่งจากผลการทดลองจะได้ว่า ที่ค่า f น้อย จะให้ค่ารัศมีการเลี้ยวที่น้อยที่สุด โดยค่ารัศมีการเลี้ยวที่น้อยที่สุดจะอยู่ที่ ค่า $K_a = 0.8$ และ $f = 1.5$ เฮิร์ต ซึ่งจะให้ค่ารัศมีการเลี้ยวอยู่ที่ 1.32 นิ้ว

จากการทดลองทั้งหมดที่กล่าวมา จึงได้ตัวแปรที่เหมาะสมโดยประมาณที่จะให้ค่าการเลี้ยวน้อยที่สุด คือ $K_a = 0.8$, $K_i = 0.6$ และ $f = 1.5$ เฮิร์ต ดังนั้นเมื่อโครงสร้างพบสิ่งกีดขวางและต้องเลี้ยวหนี ตัวแปรดังกล่าวจึงเป็นตัวแปรโดยประมาณที่จะถูกกำหนดให้โครงสร้างเกิดการเคลื่อนที่ได้รัศมีการเลี้ยวที่น้อยมากที่สุด

6.2 สรุปผลโครงงาน

หุ่นยนต์เลียนแบบการว่ายน้ำของปลาอย่างง่ายจะประกอบไปด้วยข้อต่อ 2 จุดคือ ข้อโคนหางและครีบหาง และมีการเสริมประสิทธิภาพการเลี้ยวด้วยครีบอกที่ถูกติดอยู่ 2 ข้างลำตัวปลา โครงสร้างการทำงานของหุ่นยนต์นั้น จะมีโฟโต้เซนเซอร์เป็นตัวตรวจจับสิ่งกีดขวางส่งสัญญาณมายังไมโครคอนโทรลเลอร์เพื่อประมวลผล และสั่งการการทำงานที่เหมาะสมไปยังเซอร์โวมอเตอร์ โดยสมการการเคลื่อนที่ของปลาจะเป็นสมการที่ถูกนำมาใช้ในการควบคุมการเคลื่อนที่ โดยจะมีตัวแปรที่เกี่ยวข้อง อยู่ 3 ตัวแปรหลักๆ คือ K_a, K_i และ f ซึ่งตัวแปรทั้งหลายเหล่านี้จะต้องถูกทำการทดลองเพื่อหาค่าที่เหมาะสมที่สุด และถูกนำมากำหนดใช้ในการควบคุมการเคลื่อนที่ของหุ่นยนต์เลียนแบบการว่ายน้ำของปลา ซึ่งค่าตัวแปรที่เหมาะสมโดยประมาณที่ได้จากการทดลองคือ $K_a = 0.8$, $K_i = 0.6$ และ f =1.5 เฮิร์ต

สุดท้ายแล้วหุ่นยนต์ปลาจะสามารถเกิดการเคลื่อนที่ในลักษณะของการสะบัดของหางและบิดงอของลำตัว จนเกิดการเคลื่อนที่ในลักษณะคล้ายคลึงกับการว่ายน้ำของปลา และสามารถตรวจจับสิ่งกีดขวางและเลี้ยวหลบได้ด้วยตัวเอง โดยเมื่อข้อต่อในส่วนครีบหางและโคนหางมีการเคลื่อนที่ในด้านขวามากขึ้น หุ่นยนต์ปลาก็จะมีการเคลื่อนที่ไปทางด้านขวามากขึ้น(เลี้ยวขวา) และในทางตรงกันข้ามกันเมื่อข้อต่อในส่วนครีบหาง และโคนหางมีการเคลื่อนที่ในด้านซ้ายมากขึ้น หุ่นยนต์ปลาก็จะมีการเคลื่อนที่ไปทางด้านซ้ายมากขึ้น(เลี้ยวซ้าย) แต่ไม่ว่าอย่างไรก็ตามการเคลื่อนที่อย่างสมบูรณ์แบบของหุ่นยนต์เป็นเรื่องที่ซับซ้อนและมีรายละเอียดปัจจัยต่างๆมาเกี่ยวข้องมากมาย เช่น เรื่องของกระแสน้ำ ขนาดของโครงสร้าง แรงลอยตัว และอื่นๆอีกมากมาย ซึ่งยังเป็นเรื่องที่ต้องมีการศึกษา ทดลอง และพัฒนาต่อไป

บรรณานุกรม

[1] Koichi HIRATA,Tadanori TAKIMOTO and Kenkichi TAMURA, "STUDY ON TURNING PERFORMANCE OF A FISH ROBOT" Arctic Vessel and Low Temperature Engineering Division , Ship Research Institute Japan Marine Science and Technology Center

[2] Koichi HIRATA, "Development of Experimental Fish Robot"

[3] http://ptkrobot.blogspot.com/2009/04/blog-post_115.html

[4] http://www.nmri.go.jp/eng/khirata/fish/experiment/pf300/pf300e.htm

[5] http://www.robotic~fish.net/index.php?lang=en&id=robots

ภาคผนวก ก

ไมโครคอนโทรลเลอร์

คู่มือการใช้งานบอร์ดไมโครคอนโทรลเลอร์รุ่น ET-ARMBASE2103

ET-ARM BASE2103 เป็นบอร์ดไมโครคอนโทรเลอร์ในตระกูล ARM7TDMI-S Core ซึ่งเลือกใช้ไมโครคอนโทรเลอร์ 16/32-Bit ขนาด 48 Pin แบบใช้พลังงานต่ำเป็น MCU ประจำบอร์ด ซึ่งบอร์ดนี้เลือกใช้MCU เบอร์ LPC2103ของ Philips โดยการออกแบบโครงสร้างของบอร์ดนั้นจะเน้นเรื่องการจัดวางบอร์ดให้มีขนาดเล็กเพื่อให้ง่ายต่อการนำไปประยุกต์ใช้งาน โดยได้นำ MCU มาจัดวงจรร่วมกับอุปกรณ์พื้นฐานที่จำเป็นและจัดเรียง Port แบบ 10PIN ของ ETT ตัวบอร์ดใช้ไฟ +5V นอกจากนั้น GPIO ยังสามารถรองรับสัญญาณที่เป็น 5V ได้ มี Connector UART0 (RS-232) จำนวน 2 Port สำหรับทำการDownload Hex File และใช้งานในการสื่อสาร RS232 ในโปรแกรม Application ที่เขียนขึ้นเอง

คุณสมบัติของบอร์ดในกรณีใช้ LPC2103

1. ใช้ MCU ตระกูล ARM7TDMI-S เบอร์ LPC2103 ของ Philips ซึ่งเป็น MCU ขนาด 16/32-Bit

2. ใช้ Crystal 19.6608 MHz โดย MCU สามารถประมวลผลด้วยความเร็วสูงสุดที่ 58.9824 MHzเมื่อใช้งานร่วมกับ Phase-Locked Loop (PLL) ภายในตัว MCU เอง

3. รองรับการโปรแกรมแบบ In-System Programming (ISP) และ In-Application Programming (IAP) ผ่านทาง On-Chip Boot-Loader Software ทางพอร์ต RS232-1

4. พอร์ต JTAG 20PIN สำหรับ Real Time Debugging จำนวน 1 พอร์ต

5. พอร์ต LCD มาตรฐาน ETT 14PIN จำนวน 1 พอร์ต

6. พอร์ต GPIO ขนาด 10PIN จำนวน 4 พอร์ต มาตรฐาน ETT

7. หน่วยความจำโปรแกรมแบบ Flash 128KB และ RAM 64KB

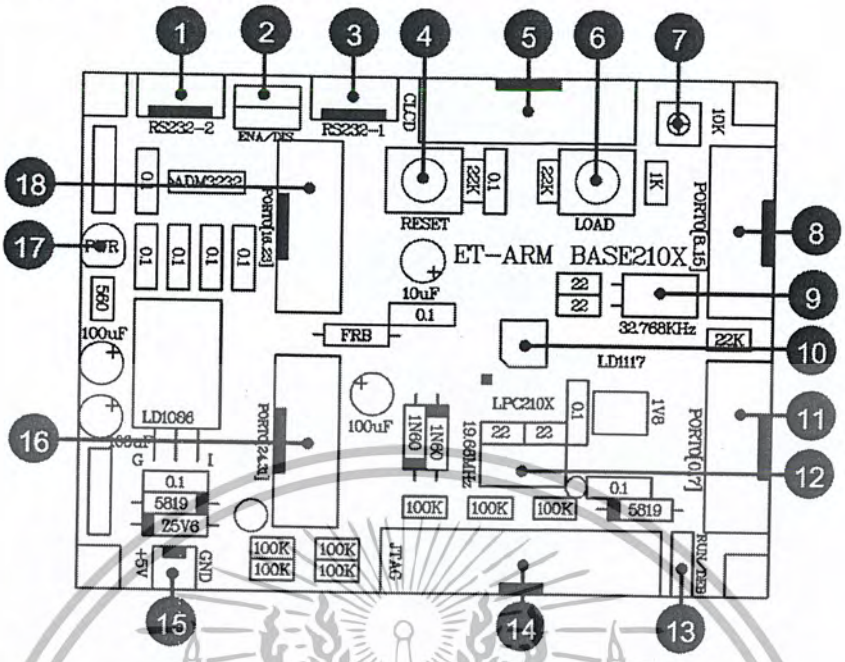
8. RTC (Real Time Clock) 32.768KHz พร้อม Battery Backup +3V

9. ใช้แหล่งจ่ายไฟ +5V Power Supply

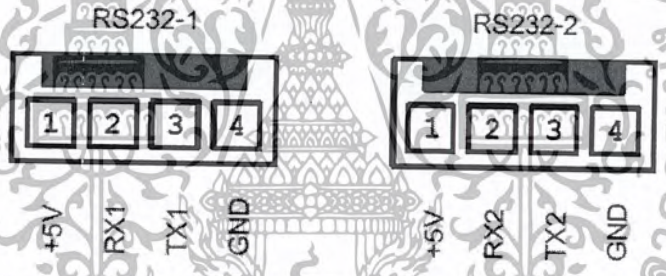
10. จำนวน GPIO สูงสุดถึง 32 I/O Pins (เฉพาะ GPIO รองรับสัญญาณที่เป็น 5V ได้) ซึ่งขาสัญญาณ GPIO จะมีการใช้งานร่วมกันของ Function อื่นๆอีกดังนี้

- SPI จำนวน 2 ช่อง
- I2C จำนวน 2 ช่อง
- 8-Channel 10 Bit A/D Converter
- UART แบบ Full-Duplex จำนวน 2 ช่อง คือ RS232-1, RS232-2 มาตรฐาน 4 Pin ETT
- Timer 32-bit จำนวน 2 ช่อง (7 Input Capture / 7 Output Compare)
- Timer 16-bit จำนวน 2 ช่อง (3 Input Capture / 7 Output Compare)
- Watchdog Timer
- PWM Output

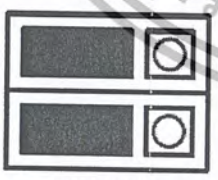
โครงสร้างบอร์ด ET-ARM BASE2103



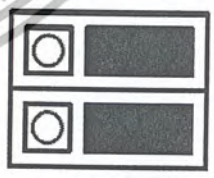
หมายเลข 1 และ 3 คือ พอร์ตการสื่อสาร RS232-2 และ RS232-1 ตามลำดับ



หมายเลข 2 คือ Jumper สำหรับเลือก ใช้งาน (Enable) หรือ ไม่ใช้งาน (Disable) สัญญาณ GPIO ของ P0.8 และ P0.9 ให้เป็นสัญญาณ RS232-2 หรือ GPIO



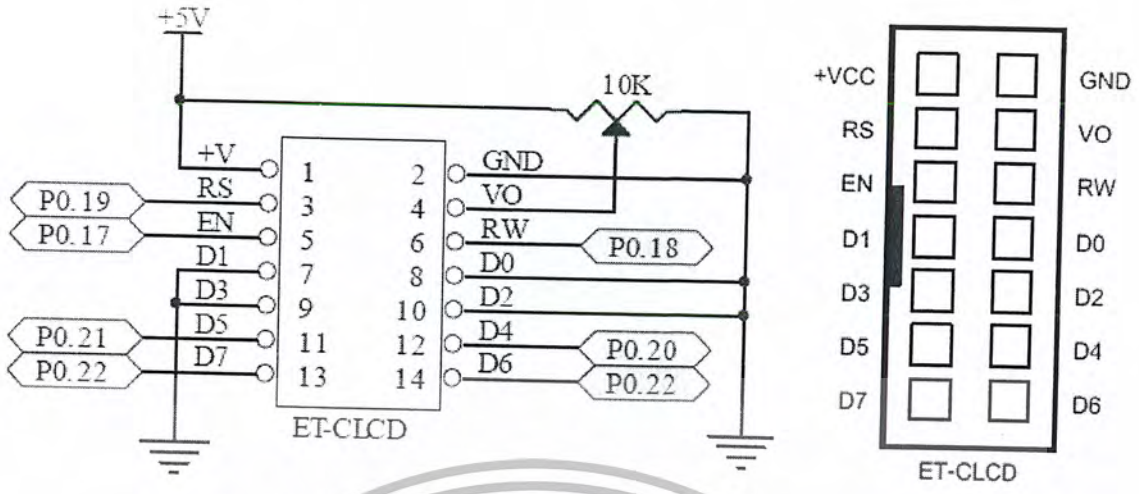
การเลือกกำหนดเป็น RS232-2



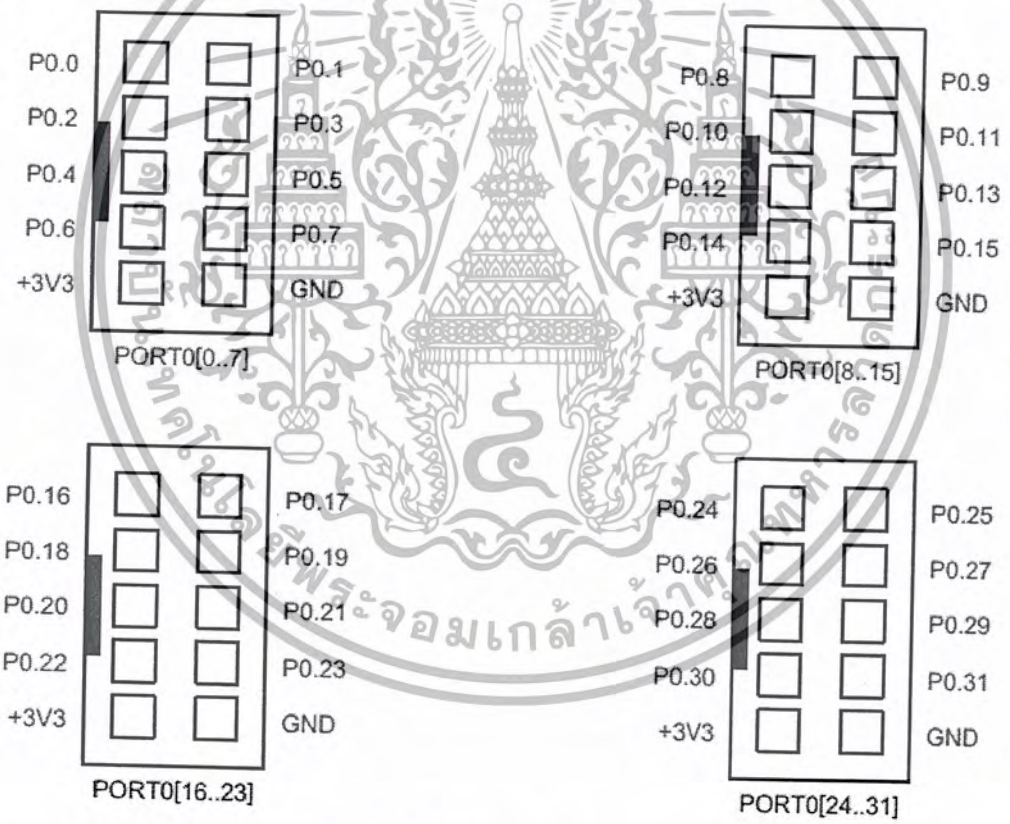
การเลือกกำหนดเป็น GPIO

หมายเลข 4 และ 6 คือ สวิตช์กด RESET และ LOAD ตามลำดับ

หมายเลข 5 และ 7 คือ LCD Connector 14PIN และ VR-10K ปรับความสว่างให้จอ LCD ตามลำดับ โดยมีการจัดวงจรและสัญญาณดังนี้



หมายเลข 8, 11, 16, และ 18 คือ GPIO0 แบ่งเป็น 4 พอร์ต ๆ ละ 8Bit รวมทั้งหมด 32Bit (เฉพาะ GPIO สามารถรองรับสัญญาณที่เป็น 3.3V และ 5V ได้)



รูปแสดงการจัดเรียงสัญญาณของ GPIO ทั้ง 4 ชุด

หมายเลข 9 คือ Crystal Oscillator 32.768 KHz สำหรับ RTC (มีเฉพาะ LPC2103)

หมายเลข 10 คือ MCU ARM7TDMI-S LPC2103 หรือ LPC2106 ของ Philips

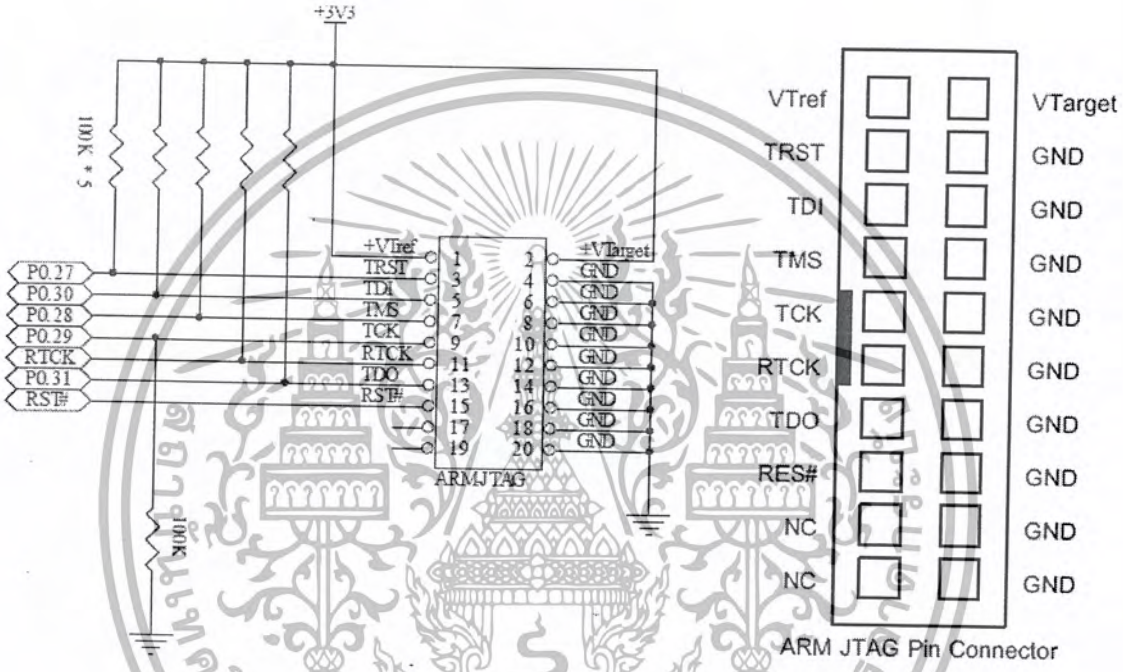
หมายเลข 12 คือ Crystal 19.6608 MHz สำหรับ MCU

หมายเลข 13 คือ Jumper สำหรับเลือกโหมด RUN Mode หรือ Debugging Mode

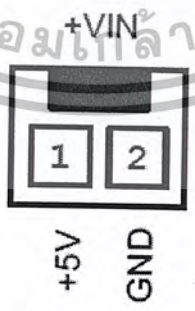
การเลือกกำหนด Jumper เป็น Run Mode

การเลือกกำหนด Jumper เป็น Debug Mode

หมายเลข 14 คือ JTAG Connector 20PIN สำหรับ Interface กับ JTAG Debugger โดยมีการจัดวงจรและสัญญาณดังนี้



หมายเลข 15 คือ จุดต่อไฟเลี้ยงบอร์ด +5V และ GND ซึ่งใช้สำหรับเป็นแหล่งจ่ายไฟให้กับบอร์ด



โดยต้องป้อนเป็นไฟกระแสตรง DC ขนาด +5V เท่านั้น โดยมีการจัดขั้วดังนี้

หมายเลข 17 คือ LED สีแดง แสดงสถานะในการทำงานของ Power Supply

การ Download Hex file ให้กับ MCU ของบอร์ด

การ Download Hex File ให้กับหน่วยความจำ Flash ของ MCU ในบอร์ดนั้น จะใช้โปรแกรมชื่อLPC2000 Flash Utility V2.2.3 ของ Philips ซึ่งจะติดต่อกับ MCU ผ่าน Serial Port ของคอมพิวเตอร์ PC โดยโปรแกรมดังกล่าวสามารถดาวน์โหลดฟรีได้ที่ www.semiconductors.philips.com

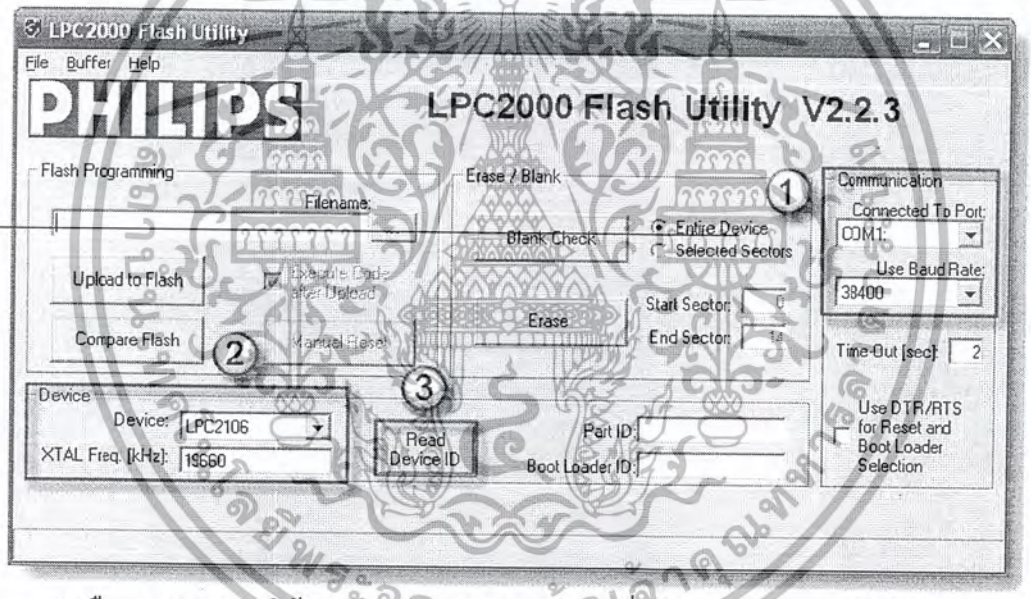
ขั้นตอนการ Download HEX File ให้กับ MCU

1. ต่อสายสัญญาณ RS232 จากพอร์ตสื่อสารอนุกรม RS232 ของ PC เข้ากับบอร์ด (RS232-1)

2. จ่ายไฟเลี้ยง +5V ให้กับบอร์ด ซึ่งจะสังเกตเห็น LED สีแดง (PWR) ติดสว่างให้เห็น

3. สั่ง Run โปรแกรม LPC2000 Flash Utility ของ Philips ซึ่งจะได้ผลดังรูป

4. เริ่มต้นกำหนดค่าตัวเลือกต่าง ๆ ของโปรแกรมให้กับบอร์ด ET-ARM BASE210x ของอีทีที ให้เลือกกำหนดค่าต่าง ๆ ตามรูป ดังตัวอย่าง



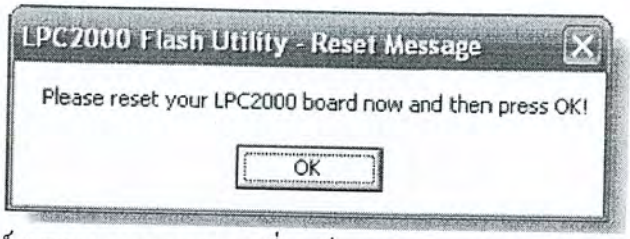
1) เลือก COM Port ให้ตรงกับหมายเลข COM Port ที่ใช้งานจริง (ในตัวอย่างใช้ COM1)

2) ตั้งค่า Baud Rate อยู่ที่ระหว่าง 4800 - 38400 ซึ่งเป็นค่าที่ทดสอบแล้วใช้ได้โดยไม่เกิดปัญหา หรือใช้ค่าความเร็วมาตรฐานคือ 9600

3) สำหรับช่อง Device ไม่สามารถเลือก MCU ได้ โปรแกรมจะแสดงเบอร์ของ LPC2103 หรือ LPC2106 ขึ้นมาเอง (ขึ้นอยู่กับ MCU ประจำบอร์ด) เมื่อสั่ง Read Device ID ได้สำเร็จ

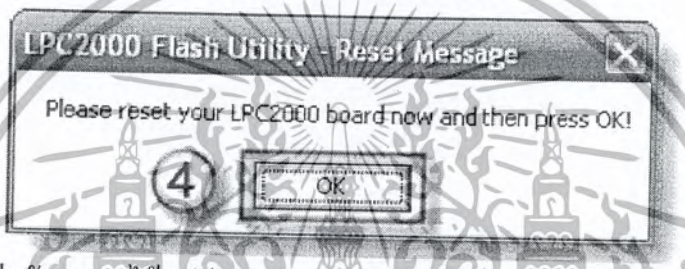
4) กำหนดค่าคริสตอลออสซิลเลเตอร์ ให้ตรงกับที่ใช้ในจริงภายในบอร์ด โดยกำหนดให้มีหน่วยเป็น KHz และห้ามใส่ค่าเกิน 5 หลัก ในที่นี้ใช้ค่า 19.6608MHz ซึ่งเท่ากับ 19660

5) คลิกเมาส์ที่ปุ่มคำสั่ง Read Device ID เพื่อติดต่อกับ CPU ซึ่งจะมีข้อความขึ้นมาเตือนให้เข้าสู่ Boot Mode ดังแสดงในรูป

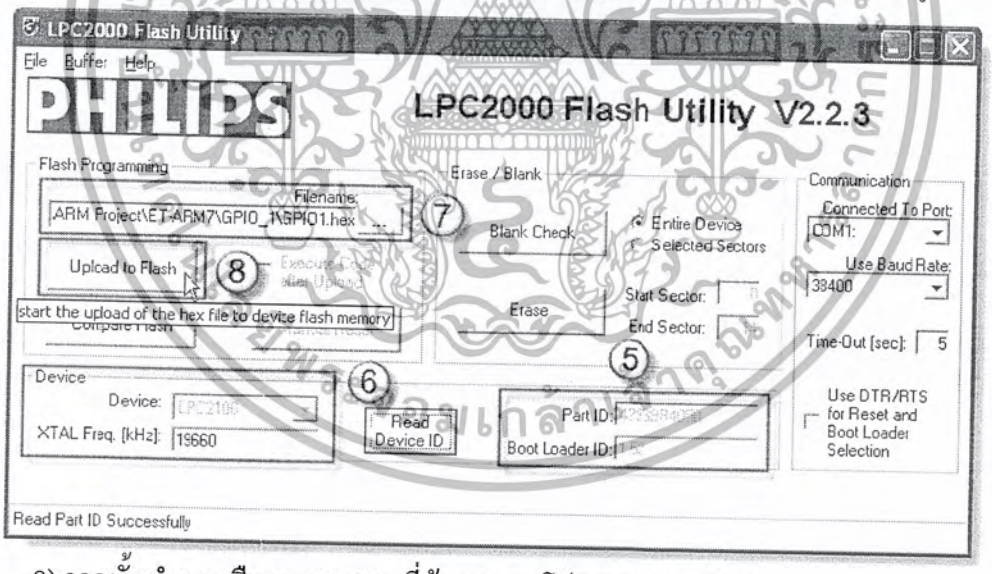


6) ให้กดสวิตช์ RESET และ LOAD ที่บอร์ด ET-ARM BASE210X เพื่อทำการ Reset ให้ MCU ทำงานในโหมด Boot Loader ตามขั้นตอนดังต่อไปนี้

- กดสวิตช์ LOAD ค้างไว้
- กดสวิตช์ RESET โดยที่สวิตช์ LOAD ยังกดค้างอยู่
- ปล่อยสวิตช์ RESET โดยที่สวิตช์ LOAD ยังกดค้างอยู่
- ปล่อยสวิตช์ LOAD เป็นลำดับสุดท้าย เสร็จแล้วจึงคลิกเมาส์ที่ "OK"

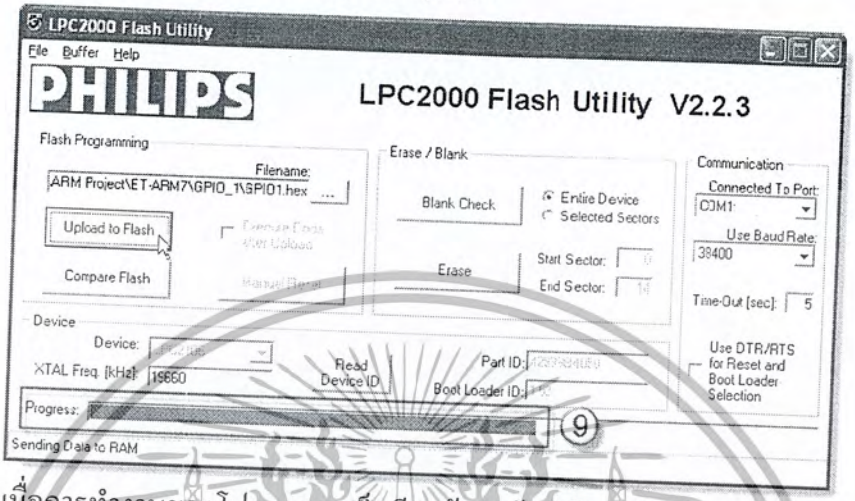


7) เมื่อติดต่อกับ CPU ได้ จะปรากฏรายละเอียด Part ID และ Boot Loader ID ดังรูป

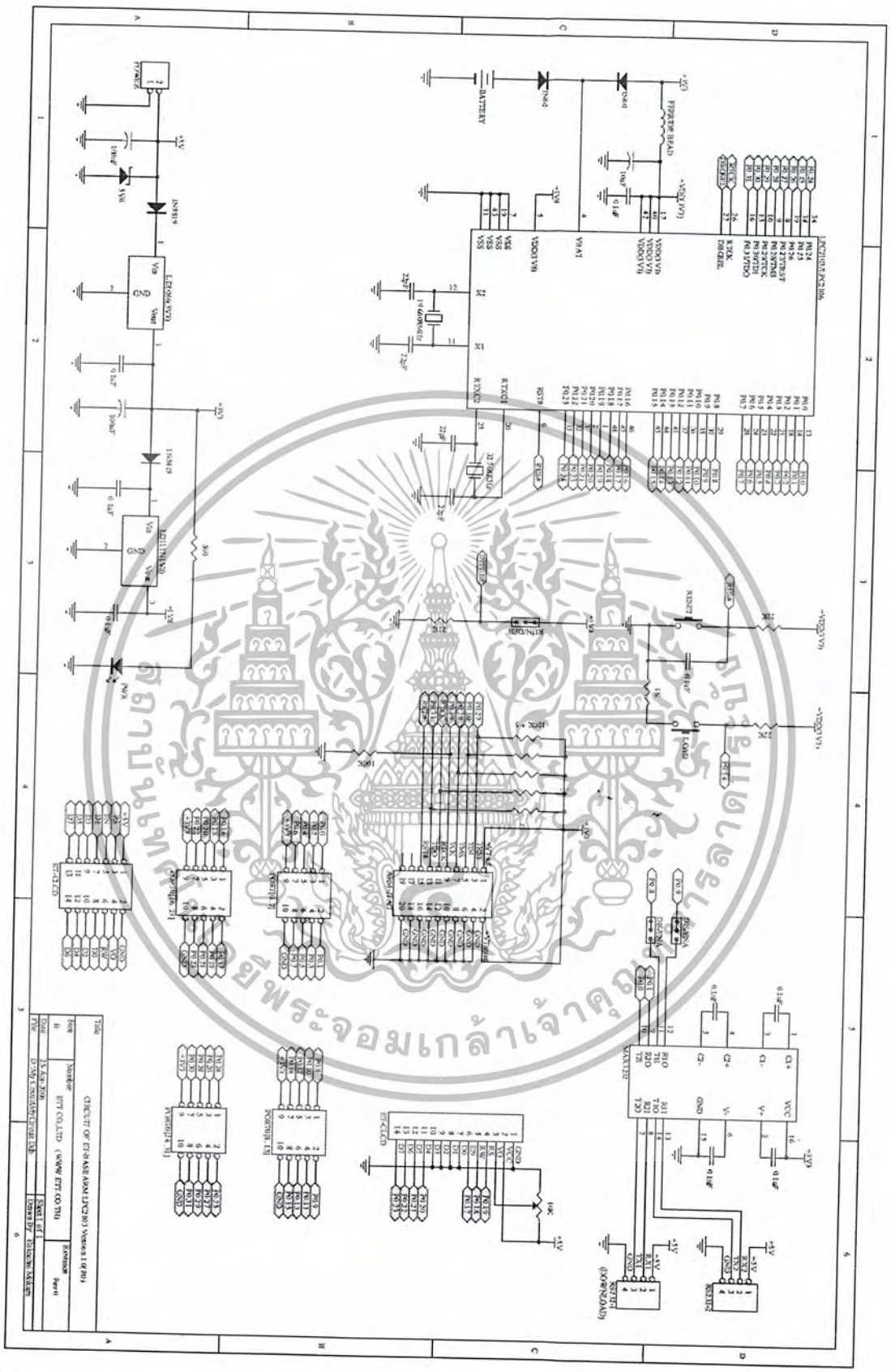


8) จากนั้นทำการเลือก HEX File ที่ต้องการจะโปรแกรมลง MCU

9) แล้วคลิกเมาส์ที่ **"Upload to Flash"** ซึ่งโปรแกรม LPC2000 จะเริ่มการDownload ข้อมูลให้กับ MCU ทันที โดยสังเกตที่ Status **"Uploading to LPC2000 RAM and Copying to Flash Memory"** ดังรูป โดยในขั้นตอนนี้ให้รอจนกว่าการทำงานของโปรแกรมจะเสร็จสมบูรณ์ ซึ่งให้สังเกตที่ Status **"File Upload Successfully Completed"**



10) เมื่อการทำงานของโปรแกรมเสร็จเรียบร้อยแล้ว ให้กดสวิตช์ Reset ที่บอร์ด ซึ่ง MCU จะเริ่มต้นทำงานตามโปรแกรมที่สั่ง Download ให้ทันที



ภาคผนวก ข.

โปรแกรม

โปรแกรมที่ใช้ในการรับค่าตัวแปรจากหน้าจอ Graphics User Interface

```
#include <LPC2103.H>                    // LPC2103 MPU Register
#include<math.h>                        //math function
#include <stdio.h>                      // For Used Function printf

/*pragard function*/

void init_serial0 (void);               // Initial UART-0
int getchar (void);                     // Get Char From Uart-0
double sin(double  x) ;
void delay(long int ms);                // Delay Time Function
void forward(void);
void right(void);
void left(void);
void clear(void);
void swim(unsigned int t);
float A1maxR,A2maxR,KaR,KiR,fR,lagR;
float A1maxF,A2maxF,KaF,KiF,fF,lagF;
float A1maxL,A2maxL,KaL,KiL,fL,lagL;
float A1max,A2max,Ka,Ki1,Ki2,f,lag;
float A1,A2;
unsigned int t,i,h;

/*Start ja*/
void main(void)
{
        h=0     ;
        t=0     ;
        i=0     ;
        init_serial0()  ;

// Select RS 232 UART1
 PINSEL0 |= 0x00010000;                 // Select P0.8 = TxD(UART1)
 PINSEL0 |= 0x00040000;                 // Select P0.9 = RxD(UART1)

// Select P0.5 0.6 0.7 I/O
 PINSEL0 &= 0xFFFF03FF;                 // 1111 1111 1111 1111 0000 0011 1111 1111
 PINSEL0 |= 0x00000000;                 // 0000 0010 0000 0000 0000 0000 0000 0000

// Initial Timer1 Pin Connect
PINSEL1 &= 0xFFFFFF3F;                  // 1111 1111 1111 1111 1111 1111 xx11 1111 =
Reset P0.19 Function
PINSEL1 |= 0x00000080;                  // 0000 0000 0000 0000 0000 0000 1000 0000 =
P0.19 -> MAT1.2
```

```
PINSEL1 &= 0xFFFFFCFF;          // 1111 1111 1111 1111 1111 11xx 1111 1111 =
Reset P0.20 Function

// Initial Timer3 Pin Connect
PINSEL1 &= 0xFFFFF3FF;          // 1111 1111 1111 1111 1111 1111 xx11 1111 =
Reset P0.21 Function
PINSEL1 |= 0x00000800;          // 0000 0000 0000 0000 0000 0000 1000 0000 =
P0.21 -> MAT3.0
PINSEL0 &= 0xFFFFFFFC;          // 1111 1111 1111 1111 1111 1111 1111 11xx =
Reset P0.0 Function
PINSEL0 |= 0x0000002;           // 0000 0000 0000 0000 0000 0000 0000 0010 =
P0.0 -> MAT3.1
PINSEL0 &= 0xFFFFFFF3;          // 1111 1111 1111 1111 1111 1111 1111 xx11 =
Reset P0.1 Function
PINSEL0 |= 0x0000008;           // 0000 0000 0000 0000 0000 0000 0000 1000 =
P0.1 -> MAT3.2
PINSEL1 &= 0xCFFFFFFF;          // 1111 1111 1111 1111 1111 11xx 1111 1111 =
Reset P0.30 Function

// All of Timer1
 // Initial Timer Operate
  T1TCR  &= 0xFFFFFFFC;          // Timer1 = Timer Mode Count By Rising PCLK
Edge
  T1TC   = 0x00000000;           // Timer Start = 0
  T1PR   = 0x00000000;           // Prescale = 0
  T1PC   = 0x00000000;           // Prescale Count = 0

 // Initial Timer1 Match Operate
  T1MCR  &= 0xFFFE;          // Disable Interrupt on Match-0 (1111 1111 1111 111x)
  T1MCR  &= 0xFFFD;          // Disable Reset TC on Match-0(1111 1111 1111 11x1)
  T1MCR  &= 0xFFFB;          // Disable Stop TC on Match-0(1111 1111 1111 1x11)
  T1MCR  &= 0xFFF7;          // Disable Interrupt on Match-1(1111 1111 1111 x111)
  T1MCR  &= 0xFFEF;          // Disable Reset TC on Match-1(1111 1111 111x 1111)
  T1MCR  &= 0xFFDF;          // Disable Stop TC on Match-1(1111 1111 11x1 1111)
  T1MCR  &= 0xFFBF;          // Disable Interrupt on Match-2(1111 1111 1x11 1111)
  T1MCR  &= 0xFF7F;          // Disable Reset TC on Match-2(1111 1111 x111 1111)
  T1MCR  &= 0xFEFF;          // Disable Stop TC on Match-2 (1111 111x 1111 1111)
  T1MCR  &= 0xFDFF;          // Disable Interrupt on Match-3(1111 11x1 1111 1111)
  T1MCR  |= 0x0400;          // Enable Reset TC on Match-3(0000 0x00 0000 0000)
  T1MCR  &= 0xF7FF;          // Disable Stop TC on Match-3 (1111 x111 1111 1111)

  // Initial PWM Function
  // Set   Output By Timer Match
  // Reset Output By TC Reset
  T1PWMCON |= 0x00000001;          // Enable MAT1.0 = PWM
  T1PWMCON |= 0x00000002;          // Enable MAT1.1 = PWM
  T1PWMCON |= 0x00000004;          // Enable MAT1.2 = PWM
```

```
// Initial Timer1 Interrupt
T1IR &= 0xFE;                    // Disable MAT0 Interrupt (1111111x)
T1IR &= 0xFD;                    // Disable MAT1 Interrupt (111111x1)
T1IR &= 0xFB;                    // Disable MAT2 Interrupt (11111x11)
T1IR &= 0xF7;                    // Disable MAT3 Interrupt (1111x111)
T1IR &= 0xEF;                    // Disable CAP0 Interrupt (111x1111)
T1IR &= 0xDF;                    // Disable CAP1 Interrupt (11x11111)
T1IR &= 0xBF;                    // Disable CAP2 Interrupt (1x111111)
T1IR &= 0x7F;                    // Disable CAP3 Interrupt (x1111111)

// Initial Timer-1 Control
T1TCR |= 0x01;                   // Timer-1 Enable       (0000000x)
T1TCR |= 0x02;                   // Timer-1 Press Reset          (000000x0)
T1TCR &= 0xFD;                   // Timer-1 Release Reset (111111x1)

// All of Timer3
// Initial Timer3 Operate
T3TCR  &= 0xFFFFFFFC;    // Timer3 = Timer Mode Count By Rising PCLK Edge
T3TC   = 0x00000000;     // Timer Start = 0
T3PR   = 10;             // Prescale = 10
T3PC   = 0x00000000;     // Prescale Count = 0

// Initial Timer3 Match Operate
T3MCR  &= 0xFFFE;        // Disable Interrupt on Match-0 (1111 1111 1111 111x)
T3MCR  &= 0xFFFD;        // Disable Reset TC on Match-0  (1111 1111 1111 11x1)
T3MCR  &= 0xFFFB;        // Disable Stop TC on Match-0   (1111 1111 1111 1x11)
T3MCR  &= 0xFFF7;        // Disable Interrupt on Match-1  (1111 1111 1111 x111)
T3MCR  &= 0xFFEF;        // Disable Reset TC on Match-1  (1111 1111 111x 1111)
T3MCR  &= 0xFFDF;        // Disable Stop TC on Match-1   (1111 1111 11x1 1111)
T3MCR  &= 0xFFBF;        // Disable Interrupt on Match-2  (1111 1111 1x11 1111)
T3MCR  &= 0xFF7F;        // Disable Reset TC on Match-2  (1111 1111 x111 1111)
T3MCR  &= 0xFEFF;        // Disable Stop TC on Match-2   (1111 111x 1111 1111)
T3MCR  &= 0xFDFF;        // Disable Interrupt on Match-3  (1111 11x1 1111 1111)
T3MCR  |= 0x0400;        // Enable Reset TC on Match-3   (0000 0x00 0000 0000)
T3MCR  &= 0xF7FF;        // Disable Stop TC on Match-3   (1111 x111 1111 1111)
```

```
// Initial PWM Function
// Set   Output By Timer Match
// Reset Output By TC Reset
T3PWMCON |= 0x00000001;     // Enable MAT1.0 = PWM
T3PWMCON |= 0x00000002;     // Enable MAT1.1 = PWM
T3PWMCON |= 0x00000004;     // Enable MAT1.2 = PWM

// Initial Timer3 Interrupt
T3IR &= 0xFE;          // Disable MAT0 Interrupt (1111111x)
T3IR &= 0xFD;          // Disable MAT1 Interrupt (111111x1)
T3IR &= 0xFB;          // Disable MAT2 Interrupt (11111x11)
T3IR &= 0xF7;          // Disable MAT3 Interrupt (1111x111)
T3IR &= 0xEF;          // Disable CAP0 Interrupt (111x1111)
T3IR &= 0xDF;          // Disable CAP1 Interrupt (11x11111)
T3IR &= 0xBF;          // Disable CAP2 Interrupt (1x111111)
T3IR &= 0x7F;          // Disable CAP3 Interrupt (x1111111)

// Initial Timer-3 Control
T3TCR |= 0x01;         // Timer-1 Enable      (0000000x)
T3TCR |= 0x02;         // Timer-1 Press Reset   (000000x0)
T3TCR &= 0xFD;         // Timer-1 Release Reset (111111x1)

// setting Timer
//T1MR2 = 554000;      // MAT1.2 P0.19 Body
T1MR3 = 589824;        // MAT1.3

//T3MR0 = 55400;       // MAT1.0 P0.21 Tail
T3MR1 = 55600;         // MAR1.1 P0.0
T3MR2 = 54000;         // MAT1.2 P0.1
T3MR3 = 58982;         // MAT1.3

str=getchar();         //A1maxL
a=(str-48)*10;
str1=getchar();
b=(str1-48);
A1maxR=a+b;

str=getchar();         //A2maxL
a=(str-48)*10;
str1=getchar();
b=(str1-48);
A2maxR=a+b;

str=getchar();         //KaL
a=(str-48);
str1=getchar();
b=(str1-46);
```

```
str2=getchar();
c=(str1-48)/10;
KaR=a+b+c;

str=getchar();          //KiL
a=(str-43);
str1=getchar();
b=(str1-48);
str2=getchar();
c=(str2-46);
str3=getchar();
d=(str3-48)/10;
KiR=(1-a)*(b+c+d);

str=getchar();          //fL
a=(str-48);
str1=getchar();
b=(str1-46);
str2=getchar();
c=(str2-48)/10;
fR=a+b+c;

str=getchar();          //lagL
a=(str-48)*10;
str1=getchar();
b=(str1-48);
lagR=a+b;

str=getchar();          //A1maxF
a=(str-48)*10;
str1=getchar();
b=(str1-48);
A1maxF=a+b;

str=getchar();          //A2maxF
a=(str-48)*10;
str1=getchar();
b=(str1-48);
A2maxF=a+b;

str=getchar();          //KaF
a=(str-48);
str1=getchar();
b=(str1-46);
str2=getchar();
c=(str2-48)/10;
KaF=a+b+c;
```

```
str=getchar();          //KiF
a=(str-43);
str1=getchar();
b=(str1-48);
str2=getchar();
c=(str2-46);
str3=getchar();
d=(str3-48)/10;
KiF=(1-a)*(b+c+d);

str=getchar();          //fF
a=(str-48);
str1=getchar();
b=(str1-46);
str2=getchar();
c=(str1-48)/10;
fF=a+b+c;

str=getchar();          //lagF
a=(str-48)*10;
str1=getchar();
b=(str1-48);
lagF=a+b;

str=getchar();          //A1maxR
a=(str-48)*10;
str1=getchar();
b=(str1-48);
A1maxL=a+b;

str=getchar();          //A2maxR
a=(str-48)*10;
str1=getchar();
b=(str1-48);
A2maxL=a+b;

str=getchar();          //KaR
a=(str-48);
str1=getchar();
b=(str1-46);
str2=getchar();
c=(str2-48)/10;
KaL=a+b+c;

str=getchar();          //KiR
a=(str-43);
```

```
str1=getchar();
b=(str1-48);
str2=getchar();
c=(str2-46);
str3=getchar();
d=(str3-48)/10;
KiL=(1-a)*(b+c+d);

str=getchar();          //fR
a=(str-48);
str1=getchar();
b=(str1-46);
str2=getchar();
c=(str2-48)/10;
fL=a+b+c;

str=getchar();          //lagR
a=(str-48)*10;
str1=getchar();
b=(str1-48);
lagL=a+b;

delay(5000);
while(1)
{

if((IOPIN & 0x00000020)!=0&&(IOPIN & 0x00000080)!=0&&(IOPIN
&0x00000040)!=0)
{
        for(i=0;i<=10;i++)
        {
                t=t+1;
                forward()  ;
                swim(t)    ;
                delay(1000);
        if(t == 65535 )
                {
                t = 0 ;
                }
        }
        clear();
}
else if((IOPIN & 0x00000020)==0&&(IOPIN & 0x00000040)!=0&&(IOPIN &
0x00000080)!=0)
{
        for(i=0;i<=10;i++)
        {
```

```
                t=t+1;
                left()    ;
                swim(t)          ;
                delay(700);
        if(t == 65535 )
                {
                t = 0 ;
                }
        }
        clear();
}
else if((IOPIN & 0x00000020)!=0&&(IOPIN & 0x00000040)==0&&(IOPIN &
0x00000080)!=0)
{
        for(i=0;i<=10;i++)
        {
                t=t+1;
                right()   ;
                swim(t)          ;
                delay(700);
        if(t == 65535 )
                {
                t = 0 ;
                }
        }
        clear();
}
else if((IOPIN & 0x00000020)!=0&&(IOPIN & 0x00000040)!=0&&(IOPIN &
0x00000080)==0)
{
        for(i=0;i<=10;i++)
        {
                t=t+1;
                left()    ;
                swim(t)          ;
                delay(700);
        if(t == 65535 )
                {
                t = 0 ;
                }
        }
        clear();
}
else if((IOPIN & 0x00000020)==0&&(IOPIN & 0x00000040)==0&&(IOPIN &
0x00000080)!=0)
{
        for(i=0;i<=10;i++)
```

```
        {
                t=t+1;
                right()  ;
                swim(t)          ;
                delay(1000);
        if(t == 65535 )
                {
                t = 0 ;
                }
        }
        clear();
}
else  if((IOPIN  &  0x00000020)==0&&(IOPIN  &  0x00000040)!=0&&(IOPIN  &
0x00000080)==0)
{
        for(i=0;i<=10;i++)
        {
                t=t+1;
                left()   ;
                swim(t)          ;
                delay(1000);
        if(t == 65535 )
                {
                t = 0 ;
                }
        }
        clear();
}
else  if((IOPIN  &  0x00000020)==0&&(IOPIN  &  0x00000040)==0&&(IOPIN  &
0x00000080)==0)
{
        for(i=0;i<=10;i++)
        {
                t=t+1;
                right()  ;
                swim(t)          ;
                delay(1000);
        if(t == 65535 )
                {
                t = 0 ;
                }
        }
        clear();
}
else
{
        for(i=0;i<=10;i++)
```

```
        {
                t=t+1;
                forward()       ;
                swim(t)         ;
                delay(1000);
        if(t == 65535 )
                {
                t = 0 ;
                }
        }
        clear();
}
 }//while(1)
 }//void main
void delay(long int ms)
{
        long int i,j;
        for(i=0;i<ms;i++)
        for(j=0;j<7371;j++);
}

void swim(unsigned int t)
{

        A1=Ka*A1max*sin((44/7)*f*t)+(1-Ka)*Ki1*A1max;
        A2=Ka*A2max*sin(((44/7)*f*t)-lag* 22 / (7 * 180))+(1-Ka)*Ki2*A2max ;
        T3MR2=55600 +(A1)*(23.33);
        T1MR1=554000+(A2)*(377.77);
}
void left(void)
        {
                A1max=A1maxR;
                A2max=A2maxR;
                Ka=KaR;
                Ki1=KiR;
                Ki2=KiL;
                f=fR;
                lag=lagR;

        }

void right(void)
        {
                A1max=A1maxL;
                A2max=A2maxL;
                Ka=KaL;
                Ki1=KiL;
```

```
                Ki2=KiR;
                f=fL;
                lag=lagL;

        }

void forward(void)
        {
                A1max=A1maxF;
                A2max=A2maxF;
                Ka=KaF;
                Ki=KiF;
                f=fF;
                lag=lagF;

        }
void clear(void)
        {
                A1max=0;
                A2max=0;
                Ka=0;
                Ki=0;
                f=0;
                lag=0;

        }
/*****************************/
/* Initial UART0 = 9600,N,8,1 */
/* VPB(pclk) = 29.4912 MHz    */
/*****************************/
void init_serial0 (void)
{
  U0LCR &= 0xFC;                    // Reset Word Select(1:0)
  U0LCR |= 0x03;                    // Data Bit = 8 Bit
  U0LCR &= 0xFB;                    // Stop Bit = 1 Bit
  U0LCR &= 0xF7;                    // Parity = Disable
  U0LCR &= 0xBF;                    // Disable Break Control
  U0LCR |= 0x80;                    // Enable Programming of Divisor Latches

  // U0DLM:U0DLL = 29.4912MHz / [16 x Baud]
  //          = 29.4912MHz / [16 x 9600]
  //          = 192 = 0x00C0
  U0DLM = 0x00;                     // Program Divisor Latch(192) for 9600 Baud
  U0DLL = 0x10;
  U0LCR &= 0x7F;                    // Disable Programming of Divisor Latches
  U0FCR |= 0x01;                    // FIF0 Enable
  U0FCR |= 0x02;                    // RX FIFO Reset
```

```
  U0FCR |= 0x04;                    // TX FIFO Reset
  U0FCR &= 0x3F;
}

/*****************************/
/* Read Character From UART0 */
/*****************************/
int getchar (void)
{
  while (!(U0LSR & 0x01));          // Wait RXD Receive Data Ready
  return (U0RBR);                   // Get Receive Data & Return
}
```

โปรแกรมที่ใช้ในการควบคุมการเคลื่อนที่ของหุ่นยนต์

```
#include <LPC2103.H>                    // LPC2103 MPU Register
#include<math.h>                        //math function
#include <stdio.h>                      // For Used Function printf

/*pragarm function*/

void init_serial0 (void);               // Initial UART-0
int getchar (void);                     // Get Char From Uart-0
double sin(double  x) ;
void delay(long int ms);                // Delay Time Function
void forward(void);
void right(void);
void left(void);
void clear(void);
void swim(unsigned int t);
float A1maxR,A2maxR,KaR,KiR,fR,lagR;
float A1maxF,A2maxF,KaF,KiF,fF,lagF;
float A1maxL,A2maxL,KaL,KiL,fL,lagL;
float A1max,A2max,Ka,Ki1,Ki2,f,lag;
float A1,A2;
unsigned int t,i,h;

/*Start */
void main(void)
{
        h=0     ;
        t=0     ;
        i=0     ;
        init_serial0()  ;

// Select RS 232 UART1
 PINSEL0 |= 0x00010000;                 // Select P0.8 = TxD(UART1)
 PINSEL0 |= 0x00040000;                 // Select P0.9 = RxD(UART1)

// Select P0.5 0.6 0.7 I/O
 PINSEL0 &= 0xFFFF03FF;                 // 1111 1111 1111 1111 0000 0011 1111 1111
 PINSEL0 |= 0x00000000;                 // 0000 0010 0000 0000 0000 0000 0000 0000

// Initial Timer1 Pin Connect
PINSEL1 &= 0xFFFFFF3F;                  // 1111 1111 1111 1111 1111 1111 xx11 1111 =
Reset P0.19 Function
PINSEL1 |= 0x00000080;                  // 0000 0000 0000 0000 0000 0000 1000 0000 =
P0.19 -> MAT1.2
```

```
PINSEL1 &= 0xFFFFFCFF;          // 1111 1111 1111 1111 1111 11xx 1111 1111 =
Reset P0.20 Function

// Initial Timer3 Pin Connect
PINSEL1 &= 0xFFFFF3FF;          // 1111 1111 1111 1111 1111 1111 xx11 1111 =
Reset P0.21 Function
PINSEL1 |= 0x00000800;          // 0000 0000 0000 0000 0000 0000 1000 0000 =
P0.21 -> MAT3.0
PINSEL0 &= 0xFFFFFFFC;          // 1111 1111 1111 1111 1111 1111 1111 11xx =
Reset P0.0 Function
PINSEL0 |= 0x0000002;           // 0000 0000 0000 0000 0000 0000 0000 0010 =
P0.0 -> MAT3.1
PINSEL0 &= 0xFFFFFFF3;          // 1111 1111 1111 1111 1111 1111 1111 xx11 =
Reset P0.1 Function
PINSEL0 |= 0x0000008;           // 0000 0000 0000 0000 0000 0000 0000 1000 =
P0.1 -> MAT3.2
PINSEL1 &= 0xCFFFFFFF;          // 1111 1111 1111 1111 1111 11xx 1111 1111 =
Reset P0.30 Function

// All of Timer1
 // Initial Timer Operate
  T1TCR  &= 0xFFFFFFFC;         // Timer1 = Timer Mode Count By Rising PCLK
Edge
  T1TC   = 0x00000000;          // Timer Start = 0
  T1PR   = 0x00000000;          // Prescale = 0
  T1PC   = 0x00000000;          // Prescale Count = 0

  // Initial Timer1 Match Operate
  T1MCR  &= 0xFFFE;       // Disable Interrupt on Match-0 (1111 1111 1111 111x)
  T1MCR  &= 0xFFFD;       // Disable Reset TC on Match-0(1111 1111 1111 11x1)
  T1MCR  &= 0xFFFB;       // Disable Stop TC on Match-0(1111 1111 1111 1x11)
  T1MCR  &= 0xFFF7;       // Disable Interrupt on Match-1(1111 1111 1111 x111)
  T1MCR  &= 0xFFEF;       // Disable Reset TC on Match-1(1111 1111 111x 1111)
  T1MCR  &= 0xFFDF;       // Disable Stop TC on Match-1(1111 1111 11x1 1111)
  T1MCR  &= 0xFFBF;       // Disable Interrupt on Match-2(1111 1111 1x11 1111)
  T1MCR  &= 0xFF7F;       // Disable Reset TC on Match-2(1111 1111 x111 1111)
  T1MCR  &= 0xFEFF;       // Disable Stop TC on Match-2 (1111 111x 1111 1111)
  T1MCR  &= 0xFDFF;       // Disable Interrupt on Match-3(1111 11x1 1111 1111)
  T1MCR  |= 0x0400;       // Enable Reset TC on Match-3(0000 0x00 0000 0000)
  T1MCR  &= 0xF7FF;       // Disable Stop TC on Match-3 (1111 x111 1111 1111)

  // Initial PWM Function
  // Set   Output By Timer Match
  // Reset Output By TC Reset
  T1PWMCON |= 0x00000001;           // Enable MAT1.0 = PWM
  T1PWMCON |= 0x00000002;           // Enable MAT1.1 = PWM
  T1PWMCON |= 0x00000004;           // Enable MAT1.2 = PWM
```

```
// Initial Timer1 Interrupt
T1IR &= 0xFE;                  // Disable MAT0 Interrupt (1111111x)
T1IR &= 0xFD;                  // Disable MAT1 Interrupt (111111x1)
T1IR &= 0xFB;                  // Disable MAT2 Interrupt (11111x11)
T1IR &= 0xF7;                  // Disable MAT3 Interrupt (1111x111)
T1IR &= 0xEF;                  // Disable CAP0 Interrupt (111x1111)
T1IR &= 0xDF;                  // Disable CAP1 Interrupt (11x11111)
T1IR &= 0xBF;                  // Disable CAP2 Interrupt (1x111111)
T1IR &= 0x7F;                  // Disable CAP3 Interrupt (x1111111)

// Initial Timer-1 Control
T1TCR |= 0x01;                 // Timer-1 Enable          (0000000x)
T1TCR |= 0x02;                 // Timer-1 Press Reset     (000000x0)
T1TCR &= 0xFD;                 // Timer-1 Release Reset   (111111x1)

// All of Timer3
// Initial Timer3 Operate
T3TCR  &= 0xFFFFFFFC;    // Timer3 = Timer Mode Count By Rising PCLK Edge
T3TC   = 0x00000000;     // Timer Start = 0
T3PR   = 10;             // Prescale = 10
T3PC   = 0x00000000;     // Prescale Count = 0

// Initial Timer3 Match Operate
T3MCR  &= 0xFFFE;     // Disable Interrupt on Match-0 (1111 1111 1111 111x)
T3MCR  &= 0xFFFD;     // Disable Reset TC on Match-0  (1111 1111 1111 11x1)
T3MCR  &= 0xFFFB;     // Disable Stop TC on Match-0   (1111 1111 1111 1x11)
T3MCR  &= 0xFFF7;     // Disable Interrupt on Match-1 (1111 1111 1111 x111)
T3MCR  &= 0xFFEF;     // Disable Reset TC on Match-1  (1111 1111 111x 1111)
T3MCR  &= 0xFFDF;     // Disable Stop TC on Match-1   (1111 1111 11x1 1111)
T3MCR  &= 0xFFBF;     // Disable Interrupt on Match-2 (1111 1111 1x11 1111)
T3MCR  &= 0xFF7F;     // Disable Reset TC on Match-2  (1111 1111 x111 1111)
T3MCR  &= 0xFEFF;     // Disable Stop TC on Match-2   (1111 111x 1111 1111)
T3MCR  &= 0xFDFF;     // Disable Interrupt on Match-3 (1111 11x1 1111 1111)
T3MCR  |= 0x0400;     // Enable Reset TC on Match-3   (0000 0x00 0000 0000)
T3MCR  &= 0xF7FF;     // Disable Stop TC on Match-3   (1111 x111 1111 1111)

// Initial PWM Function
// Set   Output By Timer Match
// Reset Output By TC Reset
T3PWMCON |= 0x00000001;    // Enable MAT1.0 = PWM
T3PWMCON |= 0x00000002;    // Enable MAT1.1 = PWM
T3PWMCON |= 0x00000004;    // Enable MAT1.2 = PWM

// Initial Timer3 Interrupt
T3IR &= 0xFE;          // Disable MAT0 Interrupt (1111111x)
T3IR &= 0xFD;          // Disable MAT1 Interrupt (111111x1)
```

```
T3IR &= 0xFB;          // Disable MAT2 Interrupt (11111x11)
T3IR &= 0xF7;          // Disable MAT3 Interrupt (1111x111)
T3IR &= 0xEF;          // Disable CAP0 Interrupt (111x1111)
T3IR &= 0xDF;          // Disable CAP1 Interrupt (11x11111)
T3IR &= 0xBF;          // Disable CAP2 Interrupt (1x111111)
T3IR &= 0x7F;          // Disable CAP3 Interrupt (x1111111)

// Initial Timer-3 Control
T3TCR |= 0x01;         // Timer-1 Enable      (0000000x)
T3TCR |= 0x02;         // Timer-1 Press Reset   (000000x0)
T3TCR &= 0xFD;        // Timer-1 Release Reset (111111x1)

// setting Timer
//T1MR2 = 554000;    // MAT1.2 P0.19 Body
T1MR3 = 589824;       // MAT1.3

//T3MR0 = 55400;      // MAT1.0 P0.21 Tail
T3MR1 = 55600;        // MAR1.1 P0.0
T3MR2 = 54000;        // MAT1.2 P0.1
T3MR3 = 58982;        // MAT1.3

while(1)
{
if((IOPIN      &      0x00000020)!=0&&(IOPIN      &      0x00000080)!=0&&(IOPIN
&0x00000040)!=0)
{
        for(i=0;i<=10;i++)
        {
                t=t+1;
                forward()      ;
                swim(t)        ;
                delay(1000);
        if(t == 65535 )
                {
                t = 0 ;
                }
        }
        clear();
}
else   if((IOPIN   &   0x00000020)==0&&(IOPIN   &   0x00000040)!=0&&(IOPIN   &
0x00000080)!=0)
{
        for(i=0;i<=10;i++)
        {
                t=t+1;
                left()    ;
                swim(t)        ;
```

```
                delay(700);
        if(t == 65535 )
                {
                t = 0 ;
                }
        }
        clear();
}
else if((IOPIN & 0x00000020)!=0&&(IOPIN & 0x00000040)==0&&(IOPIN & 0x00000080)!=0)
{
        for(i=0;i<=10;i++)
        {
                t=t+1;
                right()  ;
                swim(t)          ;
                delay(700);
        if(t == 65535 )
                {
                t = 0 ;
                }
        }
        clear();
}
else if((IOPIN & 0x00000020)!=0&&(IOPIN & 0x00000040)!=0&&(IOPIN & 0x00000080)==0)
{
        for(i=0;i<=10;i++)
        {
                t=t+1;
                left()  ;
                swim(t)          ;
                delay(700);
        if(t == 65535 )
                {
                t = 0 ;
                }
        }
        clear();
}
else if((IOPIN & 0x00000020)==0&&(IOPIN & 0x00000040)==0&&(IOPIN & 0x00000080)!=0)
{
        for(i=0;i<=10;i++)
        {
                t=t+1;
                right()  ;
```

```
                swim(t)         ;
                delay(1000);
        if(t == 65535 )
                {
                t = 0 ;
                }
        }
        clear();
}
else if((IOPIN & 0x00000020)==0&&(IOPIN & 0x00000040)!=0&&(IOPIN &
0x00000080)==0)
{
        for(i=0;i<=10;i++)
        {
                t=t+1;
                left()  ;
                swim(t)         ;
                delay(1000);
        if(t == 65535 )
                {
                t = 0 ;
                }
        }
        clear();
}
else if((IOPIN & 0x00000020)==0&&(IOPIN & 0x00000040)==0&&(IOPIN &
0x00000080)==0)
{
        for(i=0;i<=10;i++)
        {
                t=t+1;
                right()  ;
                swim(t)         ;
                delay(1000);
        if(t == 65535 )
                {
                t = 0 ;
                }
        }
        clear();
}
else
{
        for(i=0;i<=10;i++)
        {
                t=t+1;
                forward()       ;
```

```
                swim(t)          ;
                delay(1000);
        if(t == 65535 )
                {
                t = 0 ;
                }
        }
        clear();
}
 }//while(1)
 }//void main
 void delay(long int ms)
{
        long int i,j;
        for(i=0;i<ms;i++)
        for(j=0;j<7371;j++);
}


void swim(unsigned int t)
{

        A1=Ka*A1max*sin((44/7)*f*(t))+(1-Ka)*(Ki1)*A1max;
        A2=Ka*A2max*sin(((44/7)*f*(t))-lag* 22 / (7 * 180))+(1-Ka)*(Ki2)*A2max ;
        T1MR2=554000 +(A1)*(288.89);
        T3MR0=55400+(A2)*(18.89);
}
void left(void)
        {
                T3MR1=53500;
                A1max=20;
                A2max=40;
                Ka=0.7;
                Ki1=-1;
                Ki2=1;
                f=2.5;
                lag=10;
        }

void right(void)
        {
                T3MR2=57000;
                A1max=20;
                A2max=40;
                Ka=0.7;
                Ki1=1;
                Ki2=-1;
```

```
                f=2.5;
                lag=10;

        }

void forward(void)
        {
                T3MR1=55600;
                T3MR2=54000;
                A1max=25;
                A2max=50;
                Ka=0.5;
                Ki1=0;
                Ki2=0;
                f=2;
                lag=10;

        }
void clear(void)
        {
                T3MR1=55600;
                T3MR2=54000;
                A1max=0;
                A2max=0;
                Ka=0;
                Ki1=0;
                Ki2=0;
                f=0;
                lag=0;

        }

/*****************************/
/* Initial UART1 = 9600,N,8,1 */
/* VPB(pclk) = 29.4912 MHz    */
/*****************************/
void init_serial0 (void)
{
 U1LCR &= 0xFC;              // Reset Word Select(1:0)
 U1LCR |= 0x03;              // Data Bit = 8 Bit
 U1LCR &= 0xFB;              // Stop Bit = 1 Bit
 U1LCR &= 0xF7;              // Parity = Disable
 U1LCR &= 0xBF;              // Disable Break Control
 U1LCR |= 0x80;              // Enable Programming of Divisor Latches
 // U1DLM:U1DLL = 29.4912MHz / [16 x Baud]
 //          = 29.4912MHz / [16 x 9600]
 //          = 192 = 0x00C0
```

```
  U1DLM = 0x00;                 // Program Divisor Latch(192) for 9600 Baud
  U1DLL = 0x10;
  U1LCR &= 0x7F;                // Disable Programming of Divisor Latches
  U1FCR |= 0x01;                // FIFO Enable
  U1FCR |= 0x02;                // RX FIFO Reset
  U1FCR |= 0x04;                // TX FIFO Reset
  U1FCR &= 0x3F;
}

/*****************************/
/* Read Character From UART1 */
/*****************************/
int getchar (void)
{
  while (!(U1LSR & 0x01));      // Wait RXD Receive Data Ready
  return (U1RBR);               // Get Receive Data & Return
}
```