การออกแบบและสร้างหุ่นยนต์แบบ Cable-Driven Robot ขนาด 20*60 เมตร

นายทองยศ ศรีเพ็ง

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

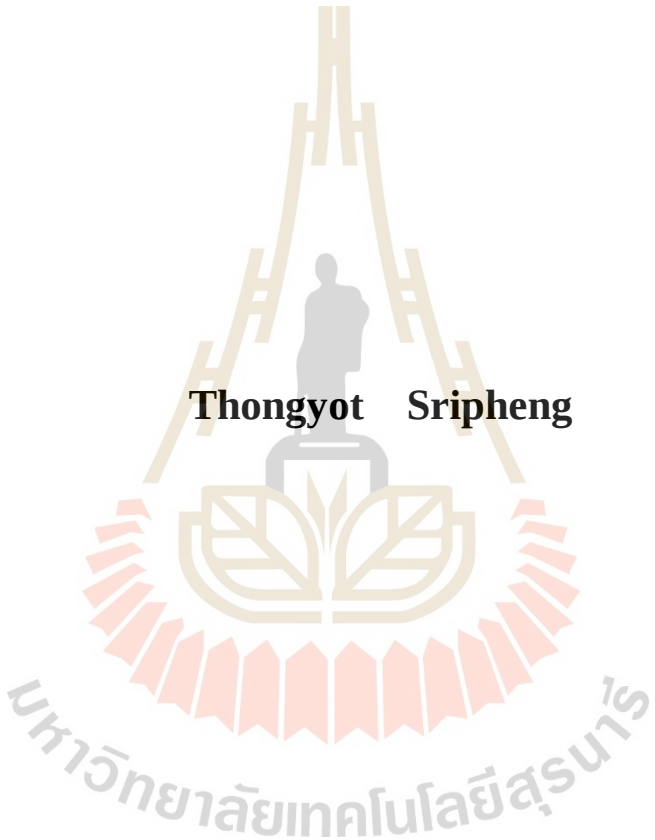
สาขาวิชาวิศวกรรมเมคคาทรอนิกส์

มหาวิทยาลัยเทคโนโลยีสุรนารี

ปีการศึกษา 2563

DESIGN AND PROTOTYPING ROBOT WITH CABLE-DRIVEN ROBOT SIZE 20*60 M

Thongyot Sripheng



A Thesis Submitted in Partial Fulfillment of the Requirements for the

Degree of Master of Engineering in Mechatronics Engineering

Suranaree University of Technology

Academic Year 2020

การออกแบบและสร้างหุ่นยนต์แบบ Cable-Driven Robot ขนาด 20*60 เมตร

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรปริญญามหาบัณฑิต

คณะกรรมการสอบวิทยานิพนธ์

(รศ. ดร.บัณฑิต ทฤตาคม)
ประธานกรรมการ

(ผศ. ดร.โศรฎา แข็งการ)
กรรมการ (อาจารย์ที่ปรึกษาวิทยานิพนธ์)

(รศ. ดร.จิระพล ศรีเสริฐผล)
กรรมการ

(รศ. ร.อ. ดร.กนต์ธร ชำนิประศาสน์)
รองอธิการบดีฝ่ายวิชาการและพัฒนาความเป็นสากล

(รศ. ดร.พรศิริ จงกล)
คณบดีสำนักวิชาวิศวกรรมศาสตร์

ทองยศ ศรีเพ็ง : การออกแบบและสร้างหุ่นยนต์แบบ Cable-Driven Robot ขนาด 20*60 เมตร (DESIGN AND PROTOTYPING ROBOT WITH CABLE-DRIVEN ROBOT SIZE 20*60 M) อาจารย์ที่ปรึกษา : ผู้ช่วยศาสตราจารย์ ดร.โศรฎา แข็งการ, 98 หน้า.

งานวิจัยนี้มีวัตถุประสงค์เพื่อศึกษา การออกแบบ การสร้าง และการควบคุมหุ่นยนต์แบบ Cable-Driven Robot ขนาด 20*60 เมตร Cable-Driven Robot เป็นหุ่นที่มีขนาดไม่ใหญ่มาก สามารถทำความเร็วได้สูง มีพื้นที่การทำงานที่ใหญ่ และสามารถประยุกต์ใช้งานได้หลากหลายในพื้นที่การทำงานของหุ่นยนต์ ด้วยการเปลี่ยนแค่ end-effector ให้ตรงกับความต้องการใช้งานในแต่ละงาน แต่ปัจจุบันยังไม่เป็นที่รู้จักกันอย่างแพร่หลาย เนื่องจากมีกลไกลการทำงานที่ยังซับซ้อนอยู่ ด้วยปัญหาและประโยชน์ดังกล่าว จึงออกแบบหุ่นยนต์แบบ Cable-Driven Robot ขนาด 20*60 เมตร ขึ้น Cable-Driven Robot เป็นหุ่นยนต์แบบขนาน เคลื่อนที่ด้วยการเปลี่ยนความยาวของสาย Cable ในแต่ละเส้น ปลายด้านหนึ่งติดกับ Winch ปลายอีกด้านติดกับ end-effector งานวิจัยนี้วิจัยแบบใช้สาย Cable 4 เส้น ด้วยพื้นที่ 20*60 เมตร การดำเนินการวิจัยแบ่งเป็น 4 ขั้นตอน ได้แก่ ขั้นที่ 1 วิเคราะห์หาความยาวในการเคลื่อนที่ของสาย Cable ในแต่ละเส้นด้วยกลศาสตร์ Inverse Kinematics ขั้นที่ 2 ออกแบบ Winch และหาขนาดของสาย Cable กับขนาดของมอเตอร์ที่ต้องใช้ ขั้นที่ 3 ออกแบบชุดความคุมใช้บอร์ด Arduino Mega 2560 Pro เป็นบอร์ดไมโครคอนโทรลเลอร์ และใช้ Protocol RS485 ในการสื่อสารระหว่างบอร์ด ขั้นที่ 4 เขียนโปรแกรมควบคุม ด้วย Arduino IDE ในการรับค่าจากรีโมตคอมนโทรล การคำนวณหาความยาวจากสมการในขั้นตอนที่ 1 และการสื่อสารระหว่างบอร์ด จากผลการวิจัยทำให้ได้หุ่นยนต์แบบ Cable-Driven Robot ขนาด 20*60 เมตร สามารถเคลื่อนที่ได้ ด้วยการสั่งงานผ่านรีโมตคอนโทรล และรับน้ำหนักได้ 20 กิโลกรัม

สาขาวิชา วิศวกรรมเมคคาทรอนิกส์ ลายมือชื่อนักศึกษา ทองยศ

ปีการศึกษา 2563 ลายมือชื่ออาจารย์ที่ปรึกษา

THONGYOT SRIPHENG : DESIGN AND PROTOTYPING ROBOT WITH CABLE-DRIVEN ROBOT SIZE 20*60 M. THESIS ADVISOR : ASST. PROF. SORADA KHAENGKARN, Ph.D., 98 PP.

FOUR CABLE-DRIVEN ROBOT/INVERSE KINEMEATICS/RS485 PROTOCAL/ ARDUINO MEGA2560 PRO

The objective of this research is to develop the design, build and control of cable-driven robot size 20*60 m. The Cable-Driven Robot is not very large., Can do high speed., Has a large working area and can be used for a wide range of applications in the working area of the robot, By changing only the end-effector to meet the needs of each job. But today is not widely known. Due to the mechanism of operation that is still complex. With such problems and benefits. This research therefore designed a cable-driven robot size 20*60 m. The cable-driven robot is a type of parallel manipulator in which flexible cables are used as actuators. The end of each cable is reeled around a rotor twisted by a motor, and the other end is connected to the end-effector. This research was done using 4 cables, with an area of 20*60 m. The research was divided into 4 steps., Analyze kinematics, Design of winch, Design of Controller and Develop program of controller. As a result of this research, cable-driven robot size 20*60 m was able to move. By operating through the remote control and can load 20 kg.

School of Mechatronic Engineering Student's Signature 9/03/64.

Academic Year 2020 Advisor's Signature

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี ทั้งนี้ผู้วิจัยขอขอบพระคุณบุลคลและหน่วยงานต่าง ๆ ที่ได้ให้คำแนะนำปรึกษา ชีแนะแนวทาง และให้ความช่วยเหลืออย่างดียิ่งเสมอมา ดังนี้

ผู้ช่วยศาสตราจารย์ ดร.โศรฎา แข็งการ อาจารย์ที่ปรึกษาวิทยานิพนธ์ที่ได้ถ่ายทอดความรู้ อบรมสั่งสอน ให้คำปรึกษา ชี้แนะข้อมูลต่าง ๆ ที่เป็นประโยชน์ในงานวิจัย และแนวทางการแก้ปัญหาต่าง ๆ ด้วยความเมตตากรุณาเสมอมา

อาจารย์ วิชัย ศรีสุรักษ์ ที่มอบโอกาส แนวคิดการออกแบบ และสนับสนุนให้เกิดงานวิจัยในครั้งนี้ รวมทั้งประสิทธิ์ประสาทวิชาความรู้ ทักษะและประสบการณ์ในการดำเนินชีวิตตลอดจนท่านเป็นต้นแบบในการดำเนินชีวิต ด้วยความเมตตากรุณมุทิตาอุเบกขา

รองศาสตราจารย์ ดร. สุรินทร บุญอนันธนสาร หัวหน้าสาขาวิชาเทคโนโลยีและนวัตกรรมทางสัตว์ ที่มอบโอกาศให้ได้ทำวิจัยในครั้งนี้ รวมทั้งอำนวยความสะดวกสถานที่ในการวิจัย

คุณศิวศิลป์ พรจำศิลป์ ผู้ร่วมในงานวิจัยนี้ ที่ได้ช่วยเหลือชี้แนะแนวทางการออกแบบและการแก้ปัญหาต่าง ๆ ในการทำวิจัยนี้ จนสำเร็จลุล่วงไปด้วยดี

คุณสุนัย พลายมี นักวิชาการเกษตร ฟาร์มมหาวิทยาลัยเทคโนโลยีสุรนารี ที่คอยประสานงานในการใช้สถานที่วิจัย ในครั้งนี้

คุณพรสวรรค์ เหม็นเณร คุณพีรพงษ์ พิมพ์อบ ที่คอยให้ความร่วมมือสนับสนุนผู้วิจัย สละเวลา แรงกาย แรงใจ ให้ความช่วยเหลือเกื้อกูลและเป็นกำลังใจที่ดีเสมอมา

มหาวิทยาลัยเทคโนโลยีสุรนารี รวมถึงบุคลากรประจำศูนย์เครื่องมือ และบุคลากรประจำสำนักวิชาวิศวรรมศาสตร์ (หลักสูตรนอกเวลา) ที่ได้ให้ความช่วยเหลือในการใช้อุปกรณ์ เครื่องมือต่าง ๆ และประสารงานเรื่องเอกสารต่าง ๆ ในการทำงานวิจัยนี้

ท้ายนี้ ขอกราบขอบพระคุณบิดา มารดา ที่ให้การเลี้ยงดูอบรม เข้าใจ เป็นกำลังใจและส่งเสริมการศึกษาเป็นอย่างดีมาโดยตลอด แม้จะด้วยความลำบากสักเพียงใด จนทำให้ผู้วิจัยประสบผลสำเร็จในชีวิตได้ ขอขอบพระคุณครับ

ทองยศ ศรีเพ็ง

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของงานวิจัย

ปัจจุบันเทคโนโลยีเข้ามามีบทบาทสำคัญกับการดำเนินชีวิตประจำวันของมนุษย์มากขึ้นเรื่อย ๆ ไม่ว่าจะเป็น ภาคอุตสาหกรรม ภาคการเกษตร ฯลฯ ทุกภาคส่วนมีการปรับตัวและพัฒนาเทคโนโลยีต่าง ๆ มาใช้ในงานมากขึ้น รวมถึงการพัฒนาและเริ่มนำเทคโนโลยีด้านหุ่นยนต์เข้ามาประยุกต์ใช้งานกันมากขึ้นเรื่อย ๆ เพื่อเป็นเครื่องทุ่นแรงคน การลดต้นทุนในการผลิต และเพื่อให้ทันต่อสถานการณ์ปัจจุบัน และ Cable-Driven Robot เป็นหุ่นยนต์อีกชนิดที่มีการวิจัยและพัฒนากันมาอย่างต่อเนื่องตั้งแต่อดีตจนถึงปัจจุบัน เนื่องจาก Cable-Driven Robot เป็นหุ่นยนต์ ที่มีข้อดีหลายอย่าง สามารถประยุกต์ใช้งานได้หลากหลาย เช่น เป็นหุ่นยนต์ที่มีน้ำหนักเบากลไหลการทำงานมีขนาดไม่ใหญ่มากเมื่อเทียบกับพื้นที่การทำงานของหุ่นยนต์ สามารถทำความเร็วได้สูง สะดวกต่อการเคลื่อนย้าย และยังสามารถประยุกต์ให้เข้ากับอุปกรณ์ต่าง ๆ ตามความต้องการ การใช้งานและการทำงานต่าง ๆ ที่หลากหลายในพื้นที่การทำงานของหุ่นยนต์

Cable-Driven Robot เป็นหุ่นยนต์แบบขนาน (Parallel Robot) เคลื่อนที่ด้วยสาย Cable 4 เส้น โดยปลายด้วยหนึ่งของสาย Cable ติดกับ Winch ปลายอีกด้านติดกับ End-effector เคลื่อนที่ End-effector ด้วยการเปลี่ยนความยาของสาย Cable ในแต่ละเส้น Cable-Driven Robot ใช้สาย Cable ในการเคลื่อนที่พื้นที่การทำงานจึงขึ้นกับความยาวของสาย Cable ทำให้หุ่นยนต์มีขนาดพื้นที่การทำงานที่ใหญ่ สาย Cable มีน้ำหนักเบาและมีความเฉื่อยน้อย ทำให้หุ่นยนต์มีความเร็วในการเคลื่อนที่สูง หุ่นยนต์ Cable-Driven ใช้ Winch 4 จุดในการเปลี่ยนความยาวของของสาย Cable 4 เส้น กลไกลของหุ่นยนต์สามารถแยกชิ้นได้ทำให้หุ่นยนต์มีขนาดไม่ใหญ่ สะดวกต่อการขนส่ง และบำรุงรักษา ตัวของหุ่นยนต์ Cable-Driven ที่เห็นได้ชัดเจนในปัจจุบัน คือ Spidercam เป็นหุ่นยนต์ที่ใช่ในการถ่ายทอดสดการแข่งขันฟุตบอลโลก ลักษณะของหุ่นยนต์ คือ ติดตั้ง Winch 4 ตัวไว้ที่ขอบสนาม 4 จุด ใช้กล้องและอุปกรณ์ที่ใช้ในการถ่ายทดสด ติดไว้ที่ End-effector แล้วควบคุมการเคลื่อนที่ตามความต้องการ แต่ทว่าต้นทุนในสร้างหุ่นยนต์ชนิดนี้ค่อนข้างสูง มีกลไกลการทำงานที่ยังซับซ้อน และการควบคุมหุ่นยนต์ให้ไปยังพิกัดที่ต้องการยังยากอยู่ หุ่นยนต์ชนิดนี้จึงยังไม่เป็นที่รู้จักในกันในเชิงลึก ส่วนใหญ่จะรู้จักกันเฉพาะวงการอุตสาหกรรมขนาดใหญ่ และนักวิจัยเท่านั้น

จากงานวิจัยของ Cable-Driven Robot ส่วนใหญ่เป็นจะเป็นการวิจัยเชิงการทดลอง และที่มีการสร้างจริงก็ยังมีต้นทุนในการผลิตสูง ทำให้คนส่วนใหญ่เข้าถึงยาก ดังนั้นวิทยานิพนธ์เล่มนี้จึงเป็นงานวิจัยเพื่อการออกแบบและสร้างต้นแบบหุ่นยนต์แบบ Cable-Driven Robot ขนาด 20*60 เมตร ให้สามารถควบคุมการเคลื่อนที่ของหุ่นยนต์ได้ด้วยรีโมตคอนโทรล และประยุกต์ใช้วัสดุที่หาได้ง่ายและต้นทุนในการผลิตไม่สูงมาก ตลอดเป็นเพื่อเป็นต้นแบบให้นักพัฒนาหรือผู้ที่สนใจนำไปพัฒนาต่อยอดในอนาคตข้างหน้าได้เป็นอย่างดี อันจะเป็นการช่วยเพิ่มขีดความสามรถทางด้านวิศวกรรมและเทคโนโลยีอย่างไร้ขีดจำกัด เป็นการพัฒนาประเทศให้ทัดเทียมกับนานาประเทศทั่วโลกอีกต่อไป

1.2 วัตถุประสงค์การวิจัย

1.2.1 การออกแบบและสร้างหุ่นยนต์แบบ Cable-Driven Robot ขนาด 20*60 เมตร

1.3 ขอบเขตของการวิจัย

1.3.1 หุ่นยนต์เคลื่อนที่ได้ในพื้นที่ขนาด 20*600 เมตร

1.3.2 บรรทุกน้ำหนักได้ไม่เกิน 20 กิโลกรัม

1.3.3 ออกแบบระบบคอนโทรลเลอร์โดยใช้ว Remote RC[A9] + Arduino

1.4 วิธีการดำเนินวิจัย

1.4.1 ศึกษาค้นคว้าองค์ความรู้ ทฤษฎี และเอกสารที่เกี่ยวข้อง

1.4.2 วิเคราะห์หลักการทางจลศาสตร์เพื่อใช้ในการออกแบบ

1.4.3 ออกแบบโครงสร้างของ CABLE-DRIVEN ROBOT

1.4.4 คำนวณหาขนาดของมอเตอร์และสาย Cable

1.4.5 ออกแบบระบบควบคุมการทำงานของคอนโทรลเลอร์

1.4.6 จัดหาวัสดุและอุปกรณ์ในการจัดสร้าง

1.4.7 ดำเนินการสร้างต้นแบบ

1.4.8 ทดสอบและปรับปรุงแก้ไข

1.4.9 วิเคราะห์ข้อมูลและสรุปผลที่ได้

1.4.10 เขียนรายงานฉบับสมบูรณ์

1.5 สถานที่ทำงานวิจัย

1.5.1 อาคารเครื่องมือวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยเทคโนโลยีสุรนารี

1.5.2 ฟาร์ม (ประมง) มหาวิทยาลัยเทคโนโลยีสุรนารี

1.6 ประโยชน์ที่คาดว่าจะได้รับ

1.6.1 ได้เรียนรู้วิธีออกแบบและสร้างหุ่นยนต์แบบ Cable-Driven Robot ขนาด 20*60 เมตร

1.6.2 ได้เรียนรู้วิธีการเขียนโปรแกรมและสร้าง Controller สำหรับการควบคุมหุ่นยนต์แบบ Cable-Driven Robot ขนาด 20*60 เมตร

1.6.3 ได้ต้นแบบออกแบบและสร้างหุ่นยนต์แบบ Cable-Driven Robot ขนาด 20*60 เมตร เพื่อนำไปพัฒนาต่อยอด

1.7 การจัดรูปเล่มวิทยานิพนธ์

วิทยานิพนธ์นี้ประกอบด้วย 7 บท 1 ภาคผนวก ซึ่งมีรายละเอียดโดยย่อดังนี้

บทที่ 1 กล่าวถึงบทนำซึ่งจะกล่าวถึงความสำคัญของปัญหาวัตถุประสงค์และเป้าหมายของงานวิจัยวิทยานิพนธ์ตลอดจนขอบเขตและประโยชน์ที่คาดว่าจะได้รับจากงานวิจัยนี้

บทที่ 2 เป็นการสำรวจปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง เพื่อให้ทราบถึงแนวทางและระเบียบวิธีการวิจัยที่เกี่ยวข้อง โดยผลจากการสำรวจสืบค้น จะใช้เป็นแนวทางสำหรับการประยุกต์และพัฒนาเข้ากับงานวิจัยวิทยานิพนธ์นี้

บทที่ 3 นำเสนอขั้นตอนการวิจัย ขั้นตอนการออกแบบ การศึกษาข้อมูลโครงสร้างก่อนการทดลอง และขั้นตอนการปฏิบัติงานในการสร้างหุ่นยนต์แบบและสร้างหุ่นยนต์แบบ Cable-Driven Robot ขนาด 20*60 เมตร

บทที่ 4 เป็นการศึกษาความสามารถและทดสอบการทำงานของอุปกรณ์ที่ใช้ทดลอง ทดสอบการส่งข้อมูลระหว่างอุปกรณ์ตามระยะการสื่อสารที่ใช้จริงด้วย Protocol Modbus RS485 ทดสอบการส่งข้อมูลผ่านชุดรีโมทคอนโทรล และผลการทดสอบการเคลื่อนที่ของหุ่นยนต์

บทที่ 5 เป็นบทสรุปผลการวิจัยและข้อเสนอแนะของการวิจัยการสร้างหุ่นยนต์แบบและสร้างหุ่นยนต์แบบ Cable-Driven Robot ขนาด 20*60 เมตร

บทที่ 2
ปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

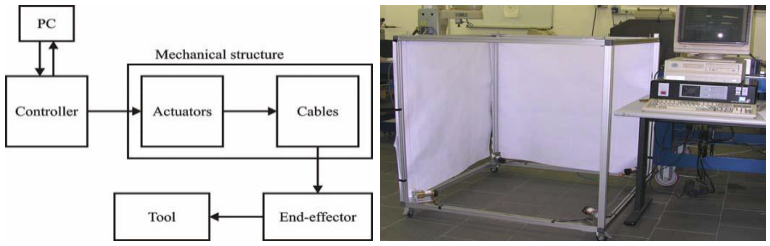
2.1 บทนำ

วรรณกรรมและงานวิจัยในอดีตที่เกี่ยวข้องกับหุ่นยนต์แบบ Cable-Driven Robt มีอยู่มากมาย ตั้งแต่การออกแบบ การวิเคราะห์หลักการทางจลนศาสตร์ การสร้างหุ่นยนต์แบบ Cable-Driven Robot รวมถึงการทดการทำงานของหุ่นยนต์แบบ Cable-Driven Robt ซึ่งงงานวิจัยเหล่านี้ได้ถูกใช้เป็นแนวทางในการทำวิจัยของผู้วิจัยเพื่อออกแบบ และสร้างหุ่นยนต์แบบ Cable-Driven Robot ขนาด 20*30 เมตร ซึ่งเป็นหุ่นยนต์ที่สร้างขึ้นโดยคาดหวังว่าจะสามารถเป็นต้นแบบในการพัฒนา หุ่นยนต์แบบ Cable-Driven ให้ใกล้เคียงกับความต้องการการใช้งานจริง และประหยัดต้นทุนในการสร้าง โดยในบทนี้ได้รวบรวมงานวิจัยต่าง ๆ ที่เกี่ยวข้องกับ การออกแบบและสร้างหุ่นยนต์แบบ Cable-Driven ในอดีตนำเสนอ เพื่อใช้เป็นแนวทางสำหรับการประยุกต์ และพัฒนาเข้ากับงานวิจัยที่จะดำเนินการต่อไป

2.2 งานวิจัยที่เกี่ยวข้อง

Erika Ottaviano (2005) [1] ได้ออกแบบและวิเคราะห์ จลนศาสตร์ ของหุ่นยนต์แบบคู่ขนาน Cable-driven robot แบบประหยัด ด้วยสาย Cable 4 เส้น มีการคำนวณหาขนาดกำลังของมอเตอร์ที่ใช้ขับเคลื่อนด้วยสมการ $\tau_{max} = r\frac{mg}{2\sqrt{2}sin\alpha_{min}}$ คำนวณหาแรงตึงสูงสุดของ Cable ด้วยสมการ $F_{max} = \frac{mg}{2\sqrt{2}sin\alpha}$ หาความยาวของสาย Cable เพื่อขับเคลื่อนหุ่นยนต์ด้วยการใช้สมการ Inverse Kinematics รวมถึงการเขียนโปรแกรมเพื่อทดสอบการเคลื่อนที่และทดสอบความเป็นไปได้ในการสร้างหุ่นยนต์แบบ Cable-Driven Robot ด้วยการสร้างต้นแบบหุ่นยนต์ แบบ Cable-Driven robot

Erika Ottaviano (2008) [2] ในบทความนี้ได้เสนอการวิเคราะห์พื้นที่การทำงานของการออกแบบ Neuron design ของ KNTU CDRPM KNTU CDRPM คือ หุ่นยนต์ที่ขับเคลื่อนด้วยสาย Cable ทดลองแบบ virtual reality ใช้ความเร็วสูงสุดที่เป็นไปได้ในการวิเคราะห์ความเป็นไปได้ของพื้นการทำงานของหุ่นยนต์



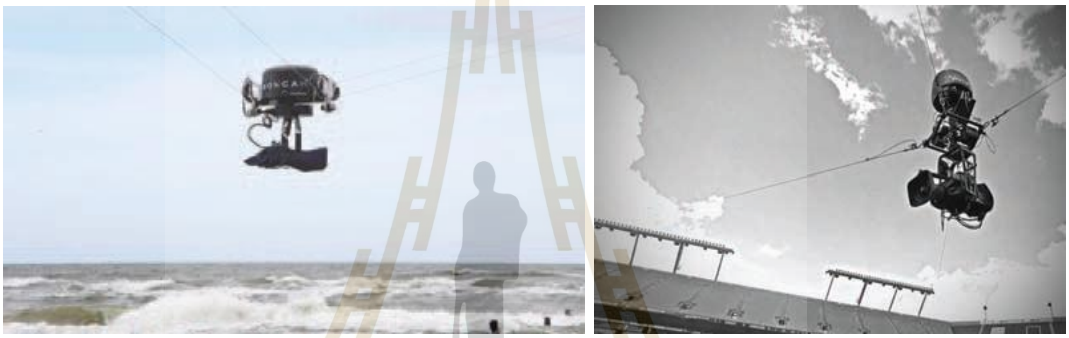
รูปที่ 2.1 A scheme and A built prototype

Jean-Pierre Merlet (2010) [3] นำเสนอการพัฒนา Modular parallel wire crane แบบพกพา เพื่อใช้ในการกู้ภัย จากการทดลองแสดงให้เห็นถึงประสิทธิภาพในการยกน้ำหนักของรอก สามารถยกน้ำหนักได้ไม่น้อยกว่า 100 กิโลกรัม แต่ยังต้องมีการปรับปรุงความแม่นยำของจลนศาสตร์เพิ่มเติมในอนาคต

รูปที่ 2.2 Modular parallel wire crane

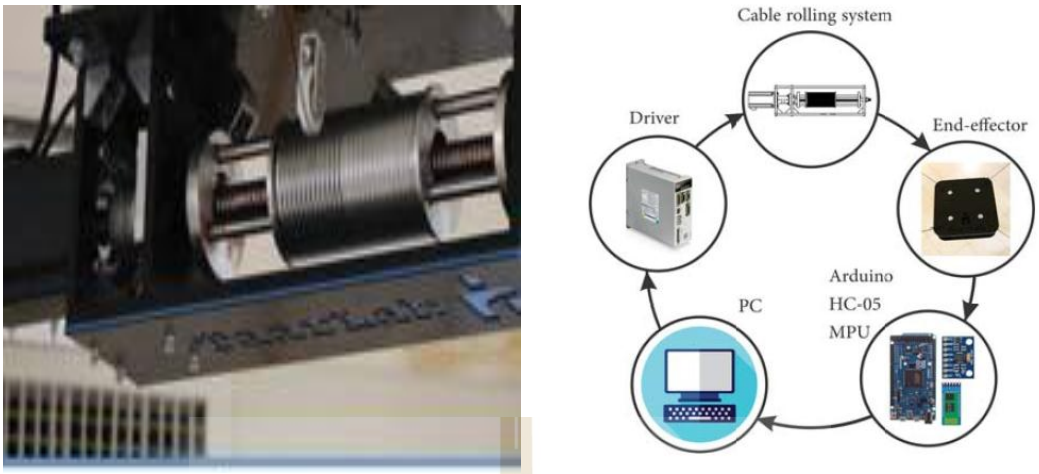
Xiaoqiang Tang (2014) [4] ศึกษาภาพรวมของการพัฒนา Cable-Driven Robot ความก้าวหน้าของทฤษฎี และการใช้งานตัวอย่างของ Cable-Driven Robot

รูปที่ 2.3 ROBOCRANE, 9-Cable-Driven Parallel Manipulator, LAR Telescope



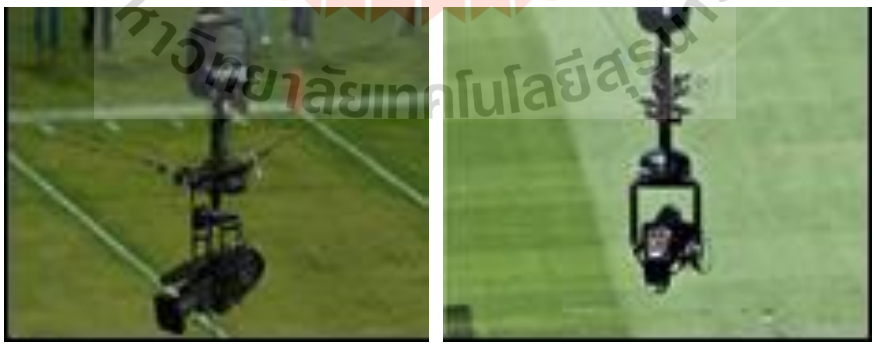
รูปที่ 2.4 Spidercam and Skycam Robot

Ramin Mersi (2018) [5] นำเสนอการออกแบบและการควบคุมหุ่นยนต์ Cable-Driven Parallel Robot ด้วย 4 Cable ภายใต้ข้อจำกัด (Under-Constrained) เพื่อปรับปรุงการพันกันของสาย Cable ที่ loller ด้วยการนำสกรูลูกกลิ้งมาใช้เพื่อเลื่อน Winch ในแนวตรง และยังมีการคำนวณค่าแบบ Real-time ของ Gyro sensor ที่ติดตั้งไว้กับ end-effector ข้อมูลที่ได้จาก Sensor นำมาใช้เพื่อแก้ปัญหา Geometric-Static และปรับปรุงสมการในการเคลื่อนที่



รูปที่ 2.5 Winch and General view of experimental setup

Yupeng Deng (2018) [6] นำเสนอการวิจัยเกี่ยวกับ Cable-driven Robots จากการค้นคว้าพบว่า ข้อดีของ Cable-driven Robots คือ ความเฉื่อยต่ำ เสียงเบา ความแม่นยำสูง และมีข้อดีที่เป็นเอกลักษณ์ นักวิจัยหลายคนได้ทำการวิจัยเชิงลึกและอภิปรายเกี่ยวกับ Cable-driven Robots ที่ได้ผลลัพธ์ที่เป็นผลสำเร็จ อย่าเช่น Skycam camera robot สามารถเคลื่อนย้ายและค้นหาตำแหน่งได้อย่างรวดเร็วเคลื่อนที่ได้เกือบทุกมุมสร้างที่ ประเทศ เยอรมณี Skycam ตัวนี้ไม่เพียง แต่สามารถเคลื่อนที่ได้ แต่ยังสามารถบินไปยังที่ใดก็ได้ในสถานที่ที่บินในระยะที่เข้าถึงได้และไม่มีข้อ จำกัดด้านความยาวของแขน โครงสร้างทางกลยังเรียบง่ายและถอดออกได้มีความยืดหยุ่นในการใช้งาน



รูปที่ 2.6 Spidercam and Skycam Robot

จากงานวิจัยที่กล่าวมาข้างต้นแสดงให้เห็นถึงน่าสนใจของ Cable-driven robot ที่มีการวิจัยมาอย่าต่อเนื่องตั้งแต่อดีตจนถึงปัจจุบัน ด้วยการประยุกต์ใช้งานได้อย่างหลากหลาย และข้อดีต่าง ๆ

ของ Cable-driven Robots ที่เหล่านักวิจัยพยายามวิจัยเพื่อที่จะนำมาประยุกต์ใช้งานในด้านต่าง ๆ ในการทำงานจริง และเพื่อให้ง่ายต่อการใช้งาน ความแม่นยำในการเคลื่อนที่ และการลดต้นทุนในการผลิต ใช้คนเข้าถึงกันมากขึ้น

2.3 ไมโครคอนโทรลเลอร์ (Microcontroller)

ไมโครคอนโทรลเลอร์ คือ อุปกรณ์ที่สามารถสร้างระบบควบคุมได้ โดยอุปกรณ์นี้มีขนาดเล็ก และเป็นอุปกรณ์ประเภทสารกึ่งตัวนำที่มีการรวมเอาฟังก์ชันการทำงานต่าง ๆ ไว้ในตัวเอง ลักษณะคล้ายกับคอมพิวเตอร์ ซึ่งในที่นี่หมายถึงอุปกรณ์ภายใน ที่ประกอบด้วย หน่วยประมวลผลกลาง, พอร์ตในการเชื่อมต่อแบบต่าง ๆ

2.3.1 ส่วนประกอบทั่วไปของไมโครคอนโทรลเลอร์

1. หน่วยประมวลผลกลาง (Control Processing Unit)
2. หน่วยความจำ ประกอบไปด้วย RAM (Random Access Memory) และ EEPROM/EPROM/PROM/ROM
3. หน่วยรับและแสดงผลข้อมูล (Input/Output) ซึ่งมีพอร์ตขยายแบบขนาน (Parallel) และอนุกรม (Serial)
4. ตัวนับเวลา (Timer)
5. หน่วยควบคุมการอินเตอร์รับ (Interrupt Controller)

ส่วนประกอบเหล่านี้เป็นเพียงส่วนประกอบพื้นฐานของไมโครคอนโทรลเลอร์ นอกจากนี้ยังมีส่วนประกอบอย่างอื่นเพื่อเพิ่มเติมความสามารถ เช่น

- ADC (Analog to Digital) ส่วนภาครับสัญญาณอนาล็อกแปลงไปเป็นสัญญาณดิจิตอล
- DAC (Digital to Analog) ส่วนภาคส่งสัญญาณดิจิตอลแปลงไปเป็น สัญญาณอนาล็อก
- I2C (Inter Integrate Circuit Bus) เป็นการสื่อสารอนุกรม แบบ ซิงโครนัส (Synchronous) เพื่อใช้ติดต่อสื่อสารระหว่าง ไมโครคอนโทรลเลอร์ (MCU) กับอุปกรณ์ภายนอกซึ่งถูกพัฒนาขึ้นโดยบริษัท Philips Semiconductors โดยใช้สายสัญญาณเพียง 2 เส้น เท่านั้นคือ serial data (SDA) และสาย serial clock (SCL)
- SPI (Serial Peripheral Interface) เป็นการเชื่อมต่อกับอุปกรณ์เพื่อรับส่งข้อมูลแบบซิงโครนัส (Synchronize) มีสัญญาณนาฬิกาเข้ามา เกี่ยวข้องระหว่างไมโครคอนโทรลเลอร์ หรือจะเป็นอุปกรณ์ภายนอกที่มีการรับส่งข้อมูลแบบ SPI

- PWM (Pulse Width Modulation) การสร้างสัญญาณพัลส์แบบสแควร์เวฟที่สามารถปรับเปลี่ยนความถี่และ Duty Cycle ได้เพื่อนำไปควบคุมอุปกรณ์ต่าง ๆ เช่น มอเตอร์

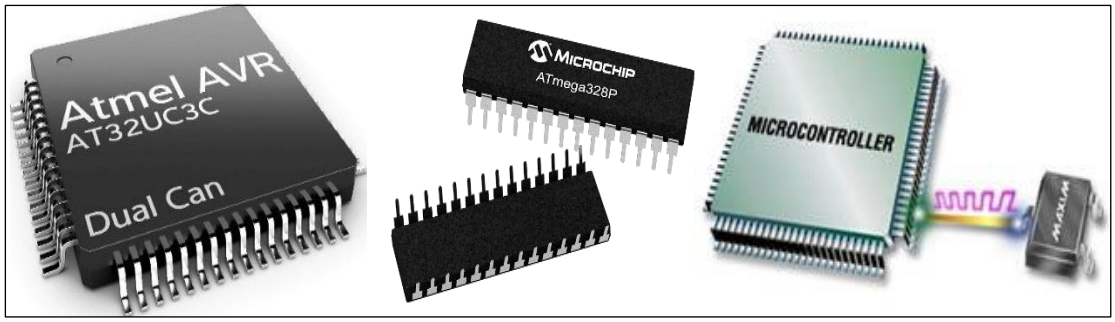
- UART (Universal Asynchronous Receiver Transmitter) ทำหน้าที่รับส่งข้อมูลแบบอะซิงโครนัสสำหรับมาตรฐานการรับส่งข้อมูลแบบ RS-232

2.3.2 ประเภทของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ มีด้วยกันหลายประเภทแบ่งตามสถาปัตยกรรม การผลิต และกระบวนการทำงานระบบการประมวลผล ที่มีใช้ในปัจจุบัน ดังนี้

1. ไมโครคอนโทรลเลอร์ตระกูล PIC (บริษัทผู้ผลิต Microchip)
2. ไมโครคอนโทรลเลอร์ตระกูล MCS51 (บริษัทผู้ผลิต Atmel, Philips)
3. ไมโครคอนโทรลเลอร์ตระกูล AVR (บริษัทผู้ผลิต Atmel)
4. ไมโครคอนโทรลเลอร์ตระกูล Arduino (บริษัทผู้ผลิต Atmel) เป็นการนำ AVR มาเขียน firmware Bootloader คำสั่งใช้งานใหม่ ข้อแตกต่างของบอร์ด Arduino กับบอร์ดทดลองทั่วไปคือเป็น Open source หมายถึงการเปิดเผยวิธีการสร้างทั้งในส่วนของ Hardware ไม่ว่าจะเป็นวงจรต่าง ๆ ภายในบอร์ด และตัว Tool Software ที่เป็นเครื่องมือใช้ในการเขียนคำสั่งต่าง ๆ ลงบอร์ด ส่งผลให้นักวิจัยมีสิทธิ์ดาวน์โหลดโปรแกรมต่าง ๆที่เกี่ยวกับการพัฒนา Arduino ไปใช้ฟรี และสามารถสร้างวงจรจากต้นแบบใช้ฟรีได้เช่นกัน ข้อดีคือเป็นการสร้างมาตรฐานในการพัฒนาจะเป็นไปในทิศทางเดียวกันเนื่องจากนักพัฒนาทั่วโลกนิยมใช้งานบอร์ดชนิดนี้ ดังนั้นไม่ว่าตัวอย่าง Source code ตลอดจนปัญหาเรื่องการใช้งาน หรือตัวอย่างการต่อวงจรเพื่อการใช้งานในรูปแบบต่างๆจะถูกพัฒนาภายใต้ Open source ของบอร์ดนี้
5. ไมโครคอนโทรลเลอร์ตระกูล ARM7, ARM9 (บริษัทผู้ผลิต Atmel, Phillips, Analog Device, Sumsung, STMicroelectronics)
6. ไมโครคอนโทรลเลอร์ตระกูล Basic Stamp (บริษัทผู้ผลิต Parallax)
7. ไมโครคอนโทรลเลอร์ตระกูล PSOC (บริษัทผู้ผลิต CYPRESS)
8. ไมโครคอนโทรลเลอร์ตระกูล MSP (บริษัทผู้ผลิต Taxas Instrument)
9. ไมโครคอนโทรลเลอร์ตระกูล 68HC (บริษัทผู้ผลิต MOTOROLA)
10. ไมโครคอนโทรลเลอร์ตระกูล H8 (บริษัทผู้ผลิต Renesas)
11. ไมโครคอนโทรลเลอร์ตระกูล RABBIT (บริษัทผู้ผลิต RABBIT SEMI-CONDUCTOR)
12. ไมโครคอนโทรลเลอร์ตระกูล Z80 (บริษัทผู้ผลิต Zilog)

(อ้างอิงจาก : ธนภัทร พรหมวัฒนภักดี, 2008)



รูปที่ 2.7 ไมโครคอนโทรลเลอร์

2.4 อาดูโน่ (Arduino)

Arduino เป็นภาษาอิตาลี โดยเป็นโครงการพัฒนาไมโครคอนโทรลเลอร์ตระกูล AVR แบบ Open Source ที่ได้รับการปรับปรุงมาจากโครงการพัฒนา Open Source ของ AVR อีกโครงการหนึ่งโดยใช้ชื่อว่า Wiring แต่เนื่องจากโครงการ Wiring เลือกใช้ AVR เบอร์ ATmega128 เป็นชิพที่มีตัวถังแบบ SMD จึงทำให้อุปสรรคสำคัญ สำหรับผู้ที่เริ่มต้นใช้ ในการสร้างบอร์ดและการต่อวงจรที่มีขนาดใหญ่เกินความจำเป็น สำหรับผู้ที่เริ่มต้น จึงไม่ค่อยเป็นที่นิยม แต่หลังจากที่Arduino ได้นำ Source code ของ Wiring มาทำการปรับปรุงและพัฒนาใหม่ ทำให้ มีขนาดเล็กลง เช่น Mega8 และ Mega168 จึงทำให้วงจรของบอร์ดมีขนาดเล็กลงกว่า Wiring มากและยังใช้อุปกรณ์น้อยชิ้น ทำให้ง่ายต่อการต่อวงจรใช้งานกันเอง ประหยัดต้นทุนในการสร้างบอร์ดไปได้มากด้วยเหตุผลนี้เอง ที่ทำให้ Arduino ได้รับความนิยมจากผู้คนทั่วโลกเป็นอย่างมากในระยะเวลาอันรวดเร็ว

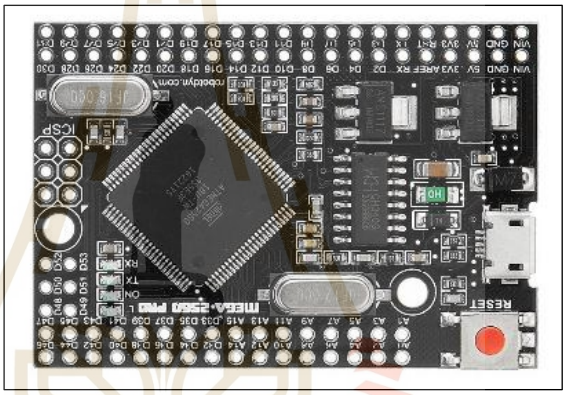
Arduino มีจุดเด่นในเรื่องความง่ายในการเรียนรู้และการใช้งาน เนื่องจากมีคำสั่งที่สนับสนุนการใช้งาน ด้วยรูปแบบที่ง่ายไม่ซับซ้อน ถึงแม้ว่ารูปแบบการใช้งานจะคล้ายกับไมโครคอนโทรลเลอร์อย่าง Basic Stamp ของ Parallax, BX-24 ของ Net medias และ Handy Board ของ MIT แต่มีจุดเด่นกว่าของรายอื่น ๆ หลายอย่าง เป็นต้นว่า

- ราคาไม่แพงเนื่องจากเป็น Open Source หมายถึงโปรแกรมคอมพิวเตอร์ที่มีลักษณะเข้าถึงได้โดยสาธารณะ เพื่อใช้หรือเปลี่ยนแปลงแก้ไขจากต้นฉบับ โค้ดโอเพนซอร์ซ ถูกสร้างมาเพื่อให้นักเขียนโปรแกรมสามารถร่วมมือกันพัฒนาซอร์ซโค้ด และแบ่งปันการเปลี่ยนแปลงกับเหล่านักพัฒนา ซึ่งโค้ดที่ถูกเผยแพร่อยู่ภายใต้เงื่อนไขของสัญญาอนุญาตซอฟต์แวร์ คนอื่นสามารถดาวน์โหลด เผยแพร่ และสร้างวงจรขึ้นมาใช้ได้เอง
- โปรแกรมที่ใช้พัฒนาของ Arduino IDE รองรับระบบปฏิบัติการทำงานทั้ง Windows Linux , Macintosh OSX รวมถึง Android บางรุ่นก็สามารถเชื่อมต่อเขียนโปรแกรมได้

- มีรูปแบบคำสั่งที่ง่ายต่อการใช้งาน แต่สามารถนำไปใช้งานจริง ๆ ที่มีความซับซ้อนได้ และยังสามารถสร้างคำสั่งและ Library ใหม่ ๆ ขึ้นมาใช้งานได้เอง

- มีการเปิดเผยวงจร ทั้งหมดทำให้สามารถนำไปพัฒนาต่อยอดเพิ่มเติมได้ตามความต้องการทั้ง Hardware และ Software

Arduino เป็นบอร์ดไมโครคอนโทรเลอร์โดยใช้ AVR ขนาดเล็กมาเป็นตัวประมวลผลและสั่งงาน เหมาะกับการนำไปประยุกต์ในการควบคุม INPUT และ OUTPUT ต่าง ๆ ได้มากมาย ทั้งเป็นการเชื่อมต่อในรูปแบบอิสระเดียวหรือการเชื่อมต่อร่วมกับอุปกรณ์อื่นยกตัวอย่าง เช่น เครื่องคอมพิวเตอร์ PC หรือการเชื่อมต่อแบบ Digital และแบบ Analog เช่นการรับค่าจากสวิตช์หรืออุปกรณ์ตรวจจับ (Sensor) แบบต่าง ๆ รวมถึงการควบคุมอุปกรณ์แบบ OUTPUT



รูปที่ 2.8 Arduino Mega 2560 Pro

การเปรียบเทียบภาษาซีกับ Arduino โดยในภาษาของ Arduino จะใช้ภาษา C++ ซึ่งเป็นรูปแบบโปรแกรมภาษาซีประยุกต์ ที่มีโครงสร้างของตัวภาษาโดยใกล้เคียงกับกับภาษาซีมาตรฐาน ANSI-C และได้มีการปรับปรุงรูปแบบในการเขียนโปรแกรมบางส่วนที่ผิดเพี้ยนไปจาก ANSI-C เล็กน้อย เพื่อใช้ในการลดความยุ่งยากในการเขียนโปรแกรม โดยที่สามารถเขียนโปรแกรมได้ง่ายขึ้นและสะดวกมากยิ่งขึ้นกว่าการเขียนภาษาซีแบบมาตรฐาน ANSI-C โดยตรงภาษาซี ซึ่งมีความได้เปรียบและมีความอ่อนตัวในการใช้งาน เหนือกว่าการใช้งานเหนือกว่าภาษาอื่น ๆ

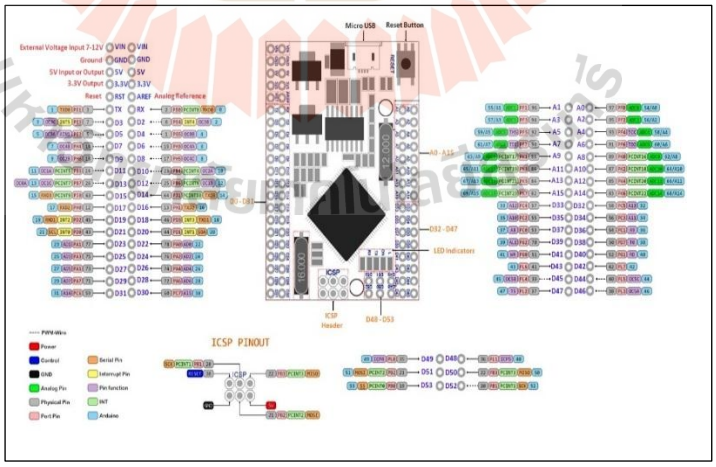
กล่าวคือ ภาษาซี สามารถนำไปประยุกต์ใช้งานในระบบฮาร์ดแวร์ที่มีความแตกต่างกันได้หลากหลาย โดยผู้ใช้เพียงแต่เลือกใช้ตัวแปรคำสั่ง (C-Compiler) ให้ตรงกับระบบฮาร์ดแวร์ ที่ใช้งาน ส่วนรูปแบบการเขียนการเขียนจะเป็นมาตรฐานเดียวกัน แต่รูปแบบภาษาซีที่มีรูปแบบที่ใช้งานง่าย แต่มีข้อกำหนดในการใช้งานหรือ Syntax แต่ไม่มีฟังก์ชั่นสำเร็จรูป (Built-in Function)ใด ๆ รวมอยู่ในตัวภาษาด้วย โดยส่วนที่เป็นฟังก์ชั่นการใช้งานต่าง ๆ เช่นการดำเนินการเกี่ยวกับ Input/Output

การจองหน่วยความจำ (Memory Allocation) เป็นหน้าที่ของผู้ใช้ที่จะต้องสร้างขึ้นมาเอง หรือในบางครั้งจะต้องใช้วิธีการเรียกใช้ฟังก์ชั่นที่ ผู้ผลิตตัวแปลคำสั่ง (C-Compiler) ซึ่งได้สร้างเตรียมไว้ในรูปแบบคำสั่ง Library Function โดยคำสั่งหรือฟังก์ชั่นนี้เองเป็นสิ่งที่ทำให้ภาษาซีมีความแตกต่าง จนบางครั้งผู้ใช้แถบไม่รู้เลยว่าอันไหนคำสั่งมาตาฐาน อันไหนผู้ใช้สร้างขึ้น

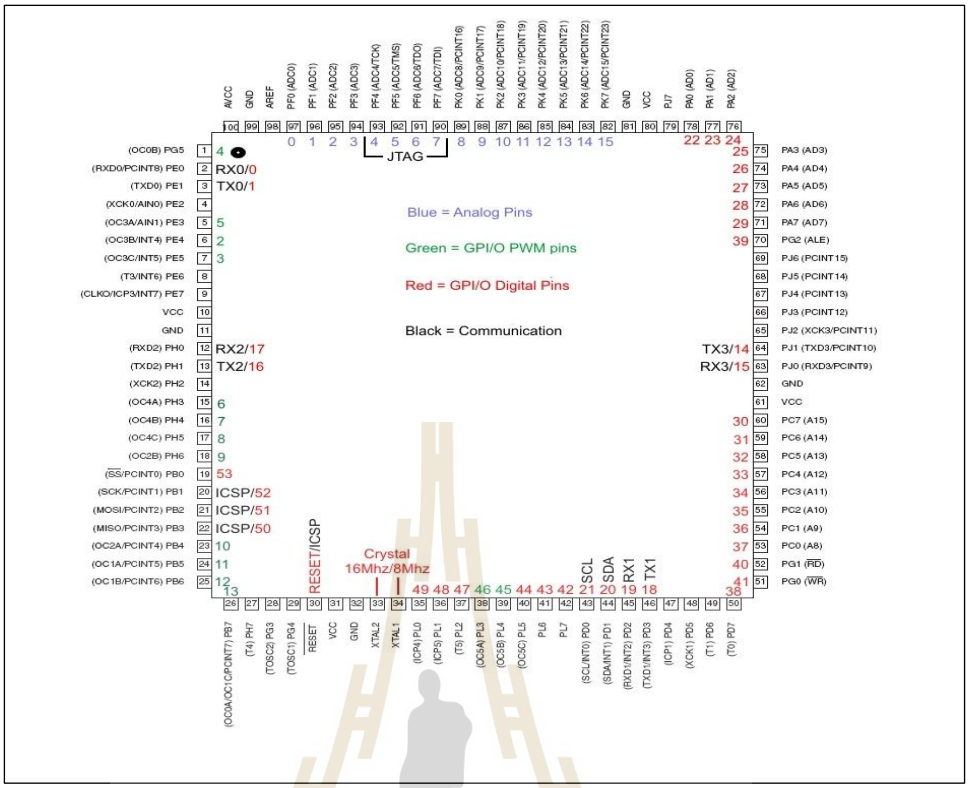
สำหรับการเขียนโปรแกรม Arduino นั้นจะใช้ภาษา C++ ซึ่งเป็นรูปแบบภาษาซีประยุกต์แบบหนึ่งแต่ มีโครงสร้างการทำงานของตัวภาษาโดยรวมจะคล้ายกับภาษาซีมาตรฐาน ANSI-C เพียงแต่มีการปรับปรุงในส่วนความยุ่งยากให้น้อยลง เพื่อสามารถให้ผู้ใช้สามารถใช้งานและเขียนโปรแกรมได้ง่ายขึ้นสะดวกกว่าภาษาซีแบบมาตรฐาน

ในการต่ออุปกรณ์เสริมต่าง ๆ คือผู้ใช้งานสามารถต่อวงจรอิเล็กทรอนิกส์จากภายนอกแล้วเชื่อมต่อเข้ามาที่ขา I/O ของบอร์ดได้

Arduino Mega 2560 Pro เป็นบอร์ด Mega2560 ตัวเดียวกับรุ่นมาตรฐานทั่วไป การอัพโหลดโค้ดใช้งานเหมือนกันทุกอย่าง แต่ย่อขนาดให้เล็กลง เหมาะกับการลดขนาดกล่องวงจร แต่ยังมีความสามารถเท่าเดิม มีคุณสมบัติ หน่วยความจำแฟลช 256 KB แรม 8 KB ใช้ไฟเลี้ยง 7 ถึง 12 V แรงดันของระบบอยู่ที่ 5 V มี Digital Input/ Output 54 ขา (which 15 provide PWM output) มี Analog Input 16 ขา Serial UART 4 ชุด เขียนโปรแกรมบนArduino IDE ผ่าน USB เหมาะสำหรับผู้ที่พัฒนาไมโครคอนโทรลเลอร์ ที่ต้องการ บอร์ด Arduino ที่มีหน่วยความจำและขาสัญญาณต่าง ๆ ให้ต่อใช้งานได้อย่างหลากหลาย



รูปที่ 2.9 PIN INPUT-OUTPUT ของ MEGA 2560 PRO ATmega2560



รูปที่ 2.10 ตำแหน่งพอร์ตของ Arduino ATmega2560

ตารางที่ 2.1 คุณสมบัติของไมโครคอนโทรลเลอร์ Arduino ATmega2560

Microcontroller	**ATmega2560**
Serial to USB Converter	CH340
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16 Port
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB
SRAM	8 KB
EEPROM	4 KBytes
Clock Speed	16 MHz

2.5 รีโมทโครคอนโทรลเลอร์ (Remote Controller)

ไมโครคอนโทรลเลอร์รีโมตคอนโทรล (อังกฤษ: remote control) คือเครื่องมืออิเล็กทรอนิกส์ชนิดหนึ่ง ใช้สำหรับควบคุมการดำเนินการของสิ่งประดิษฐ์หรือเครื่องจักรต่าง ๆ โดยเฉพาะเครื่องใช้ไฟฟ้าภายในบ้านเช่น โทรทัศน์ เครื่องเสียง เครื่องเล่นดีวีดี จากระยะไกล โดยไม่ใช้สายไฟเป็นตัวมีขนาดเหมาะมือ และมีปุ่มฟังก์ชันต่าง ๆ รีโมตคอนโทรล เป็นการเรียกย่อมาจาก รีโมตคอนโทรลเลอร์ (remote controller) อีกต่อหนึ่ง และสามารถเรียกย่อลงได้อีกเหลือเพียง รีโมต แทนได้ว่า เครื่องควบคุมระยะไกลหรืออุปกรณ์ควบคุมระยะไกล รีโมทคอนโทรลจะสามารถสั่งงานได้ ต้องประกอบด้วย 2 สิ่งนี้คือ รหัส และตัวส่งสัญญาณ 1. รหัส (Code) เป็นระบบสัญญาณจะนำออกได้ต้องมีตัวคลื่นพานำออกไป 2. ตัวส่งสัญญาณ(Carrier) ตัวรับสัญญาณ เพื่อถอดหรือรับรหัสที่ถูกส่งมาใช้ควบคุมอุปกรณ์ต่าง ๆ

2.5.1 รีโมท RadioLink AT9-S

รูปที่ 2.11 วิทยุบังคับ RadioLink AT9S

คุณสมบัติเบื้องต้น

- ใช้ได้ทั้งเครื่องบิน เครื่องร่อน มัลติโรเตอร์ เฮลิคอปเตอร์120 degree and 90 degree swashplate
- ระบบ DSSS spread spectrum มีความเสถียรของสัญญาณสูงมากไร้กังวล

- วิทยุจอเป็นสีทันสมัยสวยงามสุด ๆ หน้าจอขนาด 2.8 นิ้วระบบ 2.4G. ใช้ firmware V 1.1.8
- มีระบบแจ้งเตือนผู้เล่นแบบ Vibration alarm สามารถร้องเตือนและสั่นได้
- มีระบบ Telemetry สามารถมอนิเตอร์รับข้อมูลต่าง ๆ บนเครื่องบินจากระยะไกลได้ เช่นแรงดันไฟ ของแบตเตอรี่, ความเร็วรอบมอเตอร์, GPS บนหน้าจอวิทยุ
- สามารถตั้งเวลาถอยหลังมีเสียงเตือนเมื่อครบเวลาที่ตั้งไว้
- วิทยุรับไฟได้ 7.4V. ถึง 1.8V. มี Anti-reverse ป้องกันการเสียบแบ็ตผิดขั้ว
- มีช่อง USB สามารถ update ข้อมูลล่าสุดจากทางผู้ผลิตแบบ onlineได้
- ระยะส่งบนพื้นประมาณ 900 เมตร บนอากาศประมาณ 1.5 กิโล

Features : Reciver RadioLink AT9S

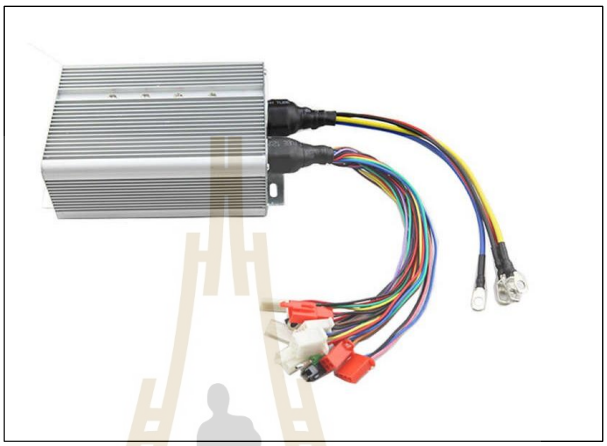
1) Size: 183*193*100 mm
2) Weight: 0.88 kg
3) Frequency: 2.4 GHz ISM band (2400 MHz to 2485 MHz)
4) Modulation mode: QPSK
5) Channel bandwidth: 5.0 MHz
6) Spread spectrum: DSSS
7) Adjacent channel rejection: > 38 dbm
8) Transmitter power: <100 mW(PCB testing), < 20 dbm (3 meter air testing)
9) Operating Current: < 105 mA
10) Operating Voltage: 7.4~15V
11) Control distance: more than 900 meters ground, 1.5 kms air
12) Channel: 9 channels, 5~9 channels are customizable
13) Simulator model: under the simulator model the transmitter action turn off
14) Screen: 2.8 inches 16 colorful screen, 240*320 pixels
15) Compatible model: Include all 120 degree and 90 degree swashplate

2.6 ชุดขับมอเตอร์ (Driver Motor)

สำหรับหุ่นยนต์ (Robot) สิ่งที่เป็นพลังขับเคลื่อนหลักให้กับกลไกต่าง ๆ คือมอเตอร์ ซึ่งมอเตอร์ต้องการการควบบคุมจากชุดไดร์เวอร์มอเตอร์ (Motor Driver) เพื่อที่จะให้พลังงาน เพื่อที่จะกำหนด ทิศทางการหมุน ความเร็วรอบของชุดมอเตอร์

2.6.1 กล่องคอนโทรล BLDC Motor รุ่น BLDC-750W-48V

กล่องคอนโทรล BLDC Motor รุ่น BLDC-750W-48V เป็นกล่องคอนโทรลสำหรับคอนโทรลมอเตอร์ Brushless DC ขนาด 48 V 750 W ใช้งานได้กับ บัสเลสมอเตอร์รอบสูง, บัสเลสมอเตอร์เกียร์ และฮับมอเตอร์

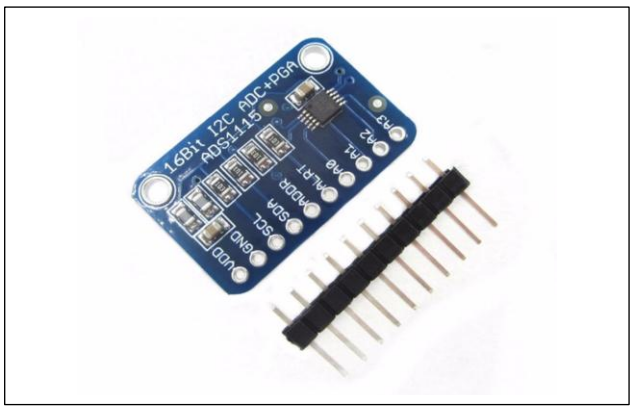


รูปที่ 2.12 กล่องคอนโทรล BLDC Motor รุ่น BLDC-750W-48V

2.6.2 โมดูล ADS1115 I2C ADC

โมดูล ADS1115 I2C ADC 4 Channel 16-Bit with Programmable Gain Amplifier Module Analog to Digital I2C สำหรับรับและแปลงสัญญาณอนาล็อกเป็นดิจิตอลความละเอียด 16-bit สั่งงานผ่านบัส I2C เปลี่ยนแอดเดรสเชื่อมต่อกันได้สูงสุด 4 โมดูลในบัสเดียว ไฟเลี้ยงโมดูล 2 ถึง 5 โวลต์ กำหนดให้ทำงานแยกเป็น 4 ช่องแบบ Single-EndedInput เหมาะสำหรับไมโครคอนโทรลเลอร์ที่ไม่มี ADC (Raspberry Pi) หรือต้องการ ADC ความละเอียดสูงขึ้น (Arduino 10-bit ADC) หรือเป็น 2 ช่องแบบ Differential Input

โมดูล ADS1115 I2C ADC 4 Channel 16-Bit with Programmable Gain Amplifier Module Analog to Digital I2C สำหรับรับและแปลงสัญญาณอนาล็อกเป็นดิจิตอลความละเอียด 16-bit สั่งงานผ่านบัส I2C เปลี่ยนแอดเดรสเชื่อมต่อกันได้สูงสุด 4 โมดูลในบัสเดียว ไฟเลี้ยงโมดูล 2 ถึง 5 โวลต์ กำหนดให้ทำงานแยกเป็น 4 ช่องแบบ Single-EndedInput เหมาะสำหรับไมโครคอนโทรลเลอร์ที่ไม่มี ADC (Raspberry Pi) หรือต้องการ ADC ความละเอียดสูงขึ้น (Arduino 10-bit ADC) หรือเป็น 2 ช่องแบบ Differential Input



รูปที่ 2.13 โมดูล ADS1115 I2C ADC 4 Channel 16-Bit

คุณสมบัติ

- WIDE SUPPLY RANGE: 2.0V to 5.5V
- LOW CURRENT CONSUMPTION: Continuous Mode: Only 150A Single-Shot Mode : Auto Shut-Down
- PROGRAMMABLE DATA RATE: 8SPS to 860SPS
- INTERNAL LOW-DRIFT VOLTAGE REFERENCE
- INTERNAL OSCILLATOR
- INTERNAL PGA
- I2C INTERFACE: Pin-Selectable Addresses
- FOUR SINGLE-ENDED OR TWO DIFFERENTIAL INPUTS
- PROGRAMMABLE COMPARATOR
- This board/chip uses I2C 7-bit addresses between 0x48-0x4B, selectable with jumpers.

2.7 มอเตอร์บัสเลส เกียร์ 750W 48V BLDC

มอเตอร์ BLDC เป็นชนิด Brushless DC แบบไม่มีแปรงถ่าน ด้านหน้าแกนมอเตอร์ มีเกียร์ทดรอบแบบซันเกียร์ ทดรอบเฟืองประมาณ 1:5 ได้รอบประมาณ 500 รอบต่อนาที ตัวเรือนมอเตอร์ทำด้วยอลูมิเนียม มีครีบระบายความร้อนรอบตัวเรือน สามารถใช้งานได้นานกว่ามอเตอร์ DC ทั่วไป ด้านหลังมอเตอร์ มีพัดลมระบายความร้อนติดตั้งมาสำเร็จรูปกับมอเตอร์ การขับเคลื่อนมอเตอร์

มอเตอร์ BLDC (Brushless DC) จะมีกล่องควบคุมในการหมุนคอนโทรลสนามแม่เหล็ก S-N ดูด และผลักกัน สลับกันไปในแต่ละจังหวะ ทำให้มอเตอร์หมุนเร็วช้าได้ตามต้องการ

รูปที่ 2.14 มอเตอร์บัสเลส เกียร์ 750W 48V BLDC

คุณสมบัติ

- กำลังวัตต์: 750w
- แรงดันไฟ DC : 48v
- กระแสสูงสุดFull Load: ≤ 19.0A
- กระแสไม่มีโหลด: ≤3A
- ทอร์ค : 2.56 N.m
- อัตราเกียร์ทด : 6:1
- ประสิทธิภาพ : ≥ 80%
- ความเร็วรอบ (มีโหลด): ประมาณ 2700 รอบต่อนาที
- ความเร็วรอบ (ไม่มีโหลด): ประมาณ 2900 รอบต่อนาที
- ทดเกียร์แล้วจะหมุนจริงประมาณ 450 รอบ/นาที
- น้ำหนัก 5.6kg

2.8 Modbus Protocol

การสื่อสารตามมาตรฐาน Modbus เป็นหนึ่งในมาตรฐานการสื่อสารแบบอนุกรม (Serial Communications protocol) ที่ใช้งานอย่างแพร่หลายในระบบอัตโนมัติอุตสาหกรรม (Industrial Automation Systems : IAS) เพื่อสร้างการเชื่อมโยงข้อมูลระหว่างอุปกรณ์ต่าง ๆ เช่น อุปกรณ์ควบคุมพีแอลซี (Programmable Logic Controllers : PLC) หรือ Arduino อุปกรณ์ตรวจวัด

(Sensor) อุปกรณ์เครื่องกล อุปกรณ์ขับเร้า (Actuator) หน่วยตรวจวัดระยะไกล (Remote Terminal Unit : RTU) รวมถึงระบบคอมพิวเตอร์ที่ใช้ในการควบคุมและแสดงสถานะของอุปกรณ์ต่าง ๆ (Supervisory control and Data acquisition : SCADA)

Modbus ถูกพัฒนาขึ้นในปี ค.ศ. 1979 โดยบริษัท Modicon (ปัจจุบันคือ Schneider Electric) เป็นโพรโทคอลที่ถูกใช้กันอย่างกว้างขวางในงานอุตสาหกรรมเนื่องจากความง่ายในการใช้งานและมีความน่าเชื่อถือ ในปัจจุบันนี้การสื่อสารสามารถแบ่งได้เป็น 2 ระบบคือ Modbus RTU และ Modbus TCP โดยความแตกต่างอยู่ที่โพรโทคอลการสื่อสารที่ใช้ ในการวิจัยนี้ผู้วิจัยใช้ Modbus RTU (RS-485)

2.8.1 RS-485

RS485 (ย่อมาจาก: Recommended Standard no. 485) คือมาตรฐานการสื่อสารข้อมูลดิจิตอลแบบอนุกรม (serial communication) ซึ่งถูกกำหนดขึ้นครั้งแรกในปี ค.ศ. 1998 โดยความร่วมมือของ TIA (Telecommunications Industry Association) และ EIA (Electronic Industries Association) มาตรฐาน RS485 ถูกใช้อย่างแพร่หลายในโรงงานอุตสาหกรรม เนื่องจากสามารถส่งสัญญาณได้ไกลและยังสามารถส่งพร้อม ๆ กันได้หลายจุด

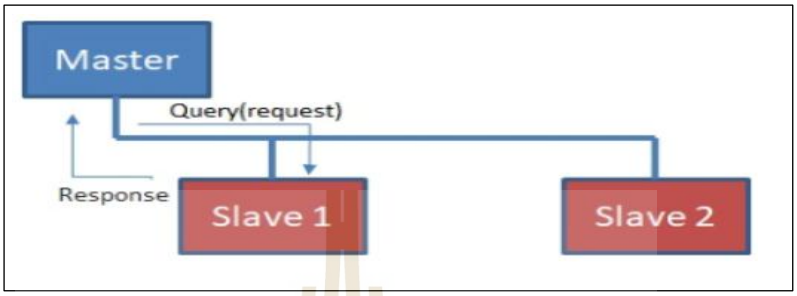
ปกติแล้ว EIA จะตั้งชื่อมาตรฐานของตัวเองโดยการใช้คำนำหน้าว่า "RS" (Recommended Standard) แต่เนื่องจากมาตรฐานนี้เป็นความร่วมมือระหว่าง 2 หน่วยงาน คือ TIA และ EIA ทั้งสองหน่วยงานจึงตกลงเปลี่ยนจากคำว่า "RS" เป็น "TIA/EIA" แทนอย่างเป็นทางการเพื่อระบุถึงแหล่งที่มาของมาตรฐานอย่างชัดเจน โดยต่อมาทาง EIA ก็ได้ยกเลิกมาตรฐานนี้และมาตรฐาน RS485 นี้ก็ได้ถูกพัฒนาอย่างต่อเนื่องจนถึงปัจจุบันโดย TIA ทำให้มาตรฐาน RS485 ถูกเปลี่ยนชื่อเป็น "TIA-485" อย่างเป็นทางการ แต่สุดท้ายเพราะความเคยชินทำให้วิศวกรทั่วโลกยังเรียกมาตรฐานการสื่อสารนี้ว่า RS485 เหมือนเดิม

หลักการทำงานของ RS485

มาตรฐาน RS485 เป็นมาตรฐานที่รับ/ส่งข้อมูลในแบบที่เรียกว่า Half duplex คือสามารถรับ และส่งข้อมูลได้ทีละอย่างเท่านั้น ไม่สามารถทำทั้งสองอย่างได้ในเวลาเดียวกัน ถ้าจะให้พูดแล้วเห็นภาพก็คงคล้าย ๆ ลักษณะของวิทยุสื่อสารที่ต้องคอยสลับกันพูดทีละครั้ง สำหรับการรับ/ส่งข้อมูลดิจิตอลแบบ RS485 นั้น จะส่งข้อมูลโดยใช้สายไฟเพียงแค่ 2 เส้นคือ A และ B เป็นตัวบอกค่ารหัสดิจิตอล (Digital code) โดยใช้ความแตกต่างของแรงดันไฟฟ้าระหว่างขั้ว A และ B เป็นตัวบอก

มาตรฐาน RS485 สามารถเชื่อมต่อการรับส่งข้อมูลแบบเครือข่าย (Network) โดยมีอุปกรณ์ในเครือข่ายได้สูงสุดถึง 32 ตัว ซึ่งในเครือข่ายนั้น จะต้องมีอุปกรณ์อยู่ 1 ตัว ทำหน้าที่คอยจัดคิวการสื่อสารในเครือข่าย ซึ่งเราจะเรียกอุปกรณ์ตัวนี้ว่า Master และอุปกรณ์ส่วนที่เหลือเราจะเรียกว่า Slave โดยที่ Slave แต่ละตัวจะมีหมายเลข Address ของตัวเอง และเมื่อตัว Master ต้องการ

สั่งการตัว Slave ตัว Master จะส่งชุดคำสั่งพร้อมระบุหมายเลข Address ไปยังอุปกรณ์ Slave ทุกตัว เมื่ออุปกรณ์ Slave ได้รับคำสั่งและคำสั่งนั้นมีหมายเลข Address ตรงกับตัวเอง อุปกรณ์ Slave ถึงจะทำตามคำสั่งของ Master เป็นลำดับไป



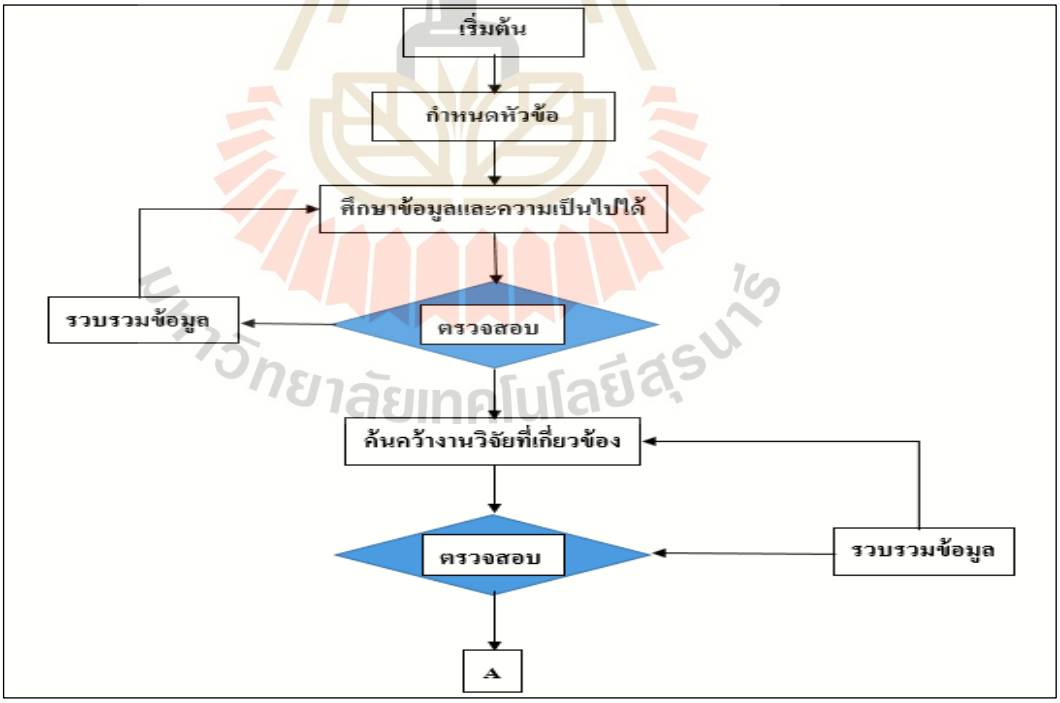
รูปที่ 2.15 การสื่อสารแบบอนุกรมด้วย RS-485 สำหรับ Modbus RTU

บทที่ 3
วิธีดำเนินงานวิจัย

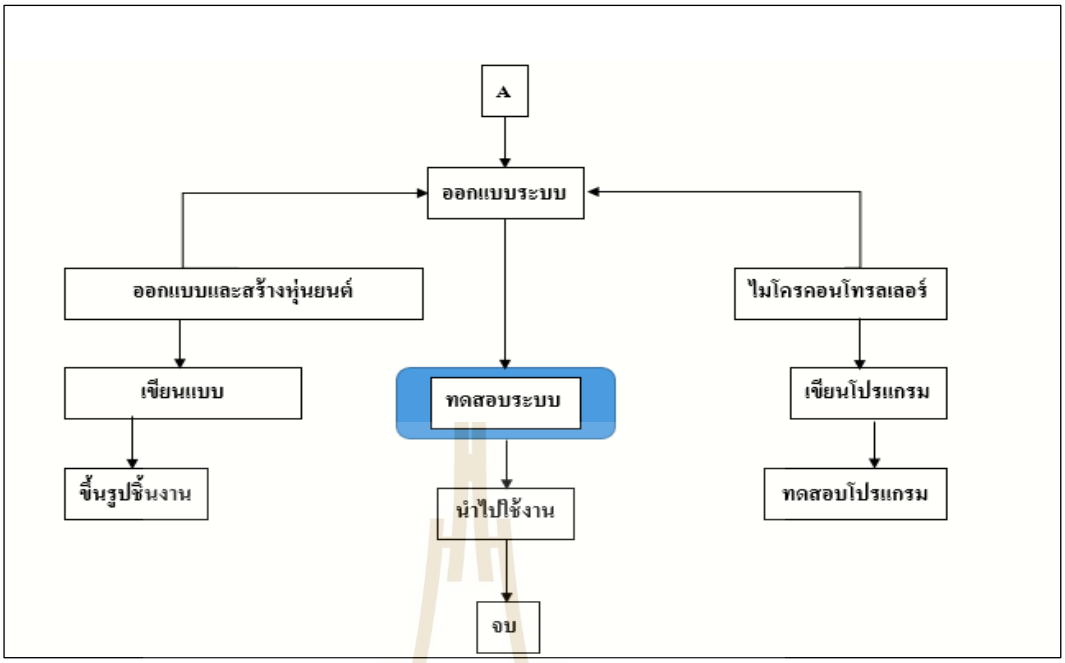
3.1 กล่าวนำ

งานวิจัยนี้เป็นส่วนหนึ่งในการศึกษามีความต้องการออกแบบและสร้างหุ่นยนต์แบบ Cable-Driven Robot ขนาด 20*60 เมตร นำไปพัฒนาและประยุกต์ใช้งานในด้านต่างได้ตามความต้องการการใช้งานได้อย่างหลากหลาย กล่าวคือในส่วนของวิธีการดำเนินการซึ่งมีด้วยกัน 2 ส่วน โดยแบ่งเป็นส่วนที่ 1 คือการออกแบบขั้นตอนการวิจัย และส่วนที่ 2 คือการออกแบบขั้นตอนการสร้าง ซึ่งขั้นตอนในการดำเนินงานจะมีดังต่อไปนี้

3.2 การออกแบบขั้นตอนการวิจัย



รูปที่ 3.1 แผนผังแสดงขั้นตอนการดำเนินงานวิจัย



รูปที่ 3.2 แผนผังแสดงขั้นตอนการดำเนินงานวิจัย (ต่อ)

3.3 การออกแบบขั้นตอนการสร้าง

ในส่วนของขั้นตอนการสร้างหุ่นยนต์นั้น ได้แบ่งส่วนการดำเนินงานออกเป็น 3 ส่วน ดังนี้

1. ศึกษาพื้นที่ที่ใช้ในการสร้างหุ่นยนต์

ในส่วนนี้เป็นการลงพื้นที่สำหรับใช้ในการออกแบบ เก็บข้อมูลต่าง ๆ ที่จำเป็นสำหรับการออกแบบหุ่นยนต์ อย่างเช่นข้อมูลทางไฟฟ้าที่ใช้อยู่ ลักษณะของพื้นที่ต่าง ๆ พื้นที่สำหรับติดตั้งส่วนประกอบของหุ่นยนต์ เนื่องจากหุ่นยนต์เป็นหุ่นยนต์ที่มีพื้นที่การทำงานขนาดใหญ่ใช้ไฟฟ้าเป็นต้นกำลังในการทำงาน และหุ่นยนต์ต้องทำงานสัมพันธ์กันกับอุปกรณ์ที่อยู่ห่างกัน การลงพื้นที่เก็บข้อมูลก่อนการออกแบบและทดลองจึงมีความสำคัญอย่างมาก

2. ออกแบบโครงสร้างของหุ่นยนต์

ในการออกแบบจะเลือกวัสดุที่ที่มีอยู่ทั่วไปก่อนแล้วค่อยนำมาต่อเติมแก้ไขเพื่อให้ได้ตามแบบที่เขียนไว้ เนื่องจากจะทำให้ลดต้นทุนในการสร้างหุ่นยนต์ ให้คนทั่วไปสามารถเข้าถึงในการพัฒนาต่อยอดได้ง่าย ภายหลังการออกแบบด้วยโปรแกรมแล้ว ทำให้ทราบขนาดของโครงสร้างที่ที่ต้องการ เพื่อจัดเตรียมวัสดุอุปกรณ์สำหรับโครงสร้าง และทำการประกอบต่อเติมให้ตรงตามแบบที่วางแผนไว้ โดยในงานวิจัยนี้ผู้พัฒนาได้สร้างหุ่นยนต์ออกมาหลายโมเดลเพื่อใช้ทดสอบความสามารถดึงสาย Cable ให้ได้ความเร็วและแรงดึงที่ต้องการ หลังจากนั้นทำการติดตั้งอุปกรณ์อิเล็กทรอนิกส์ที่ใช้ในการควบคุม ซึ่งในการติดตั้งอุปกรณ์อิเล็กทรอนิกส์นี้จะต้องคำนึงถึงพื้นที่ที่

เหมาะสมและสามารถปรับปรุงแก้ไขได้ง่าย รวมไปถึงการหากล่องกันฝนสำหรับใส่อุปกรณ์คอนโทรลเพื่อกันไม่ให้ชุดคอนโทรลโดนฝน เพื่ออายุการใช้งานที่ยาวนาน

3. เขียน Software

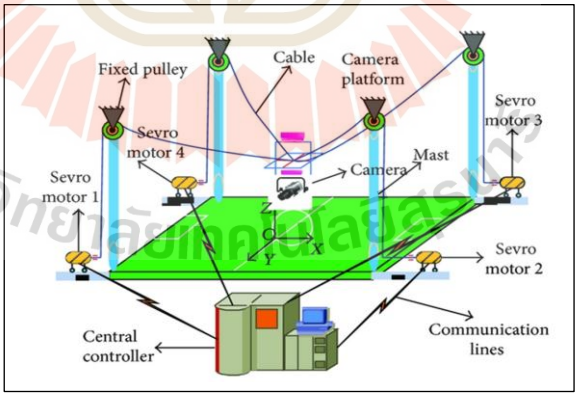
ในส่วนนี้เป็นขั้นตอนการออกแบบโปรแกรมเพื่อใช้ในการควบคุมระบบการทำงาน โดยผู้พัฒนาจะเรียงลำดับการทดสอบการทำงานแยกออกจากกันโมดูลย่อย เมื่อสมบูรณ์ทีละส่วนแล้วจึงนำเข้ามารวมกัน

3.4 ศึกษาข้อมูลโครงสร้างก่อนดำเนินการ

ในส่วนนี้เป็นดำเนินการค้นคว้าหาตัวอย่างที่เหมาะสมที่สุด เพื่อเป็นแม่แบบในการใช้สร้าง โดยมีการกำหนดพารามิเตอร์ต่าง ๆ ว่าด้วยขนาดของ Robot ความเร็วที่ต้องการ ระยะเวลาการใช้งานในแต่ละภารกิจ รวมถึงงบประมาณในการทำหุ่นยนต์

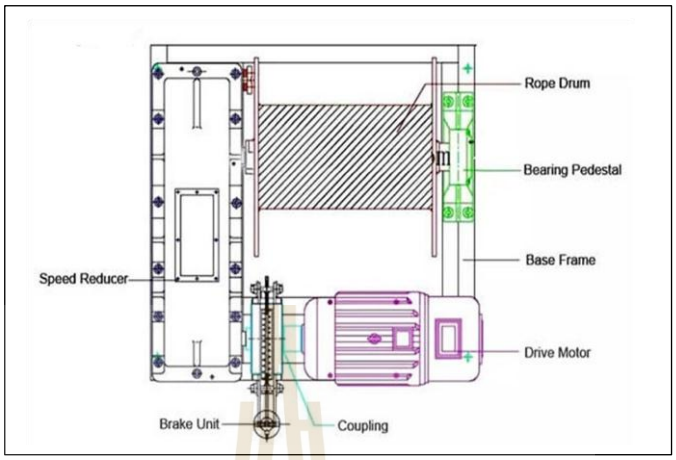
หลังจากนั้นดำเนินการสรุปผลเก็บบันทึกข้อมูลตรวจสอบแนวทางการสร้างต้นแบบ โดยข้อมูลที่ได้ทำการเก็บและนำมาวิเคราะห์สรุปผล เพื่อเป็นประโยชน์ในการสร้างหุ่นยนต์ตลอดจนการแก้ไขข้อบกพร้องที่เกิดขึ้นระหว่างการสร้างตัวชิ้นงานและทดสอบ จากนั้นทำการบันทึกและสรุปผลในขั้นตอนถัดไป

3.4.1 ข้อมูลโครงสร้าง ต้นแบบ Robot แบบ Overview



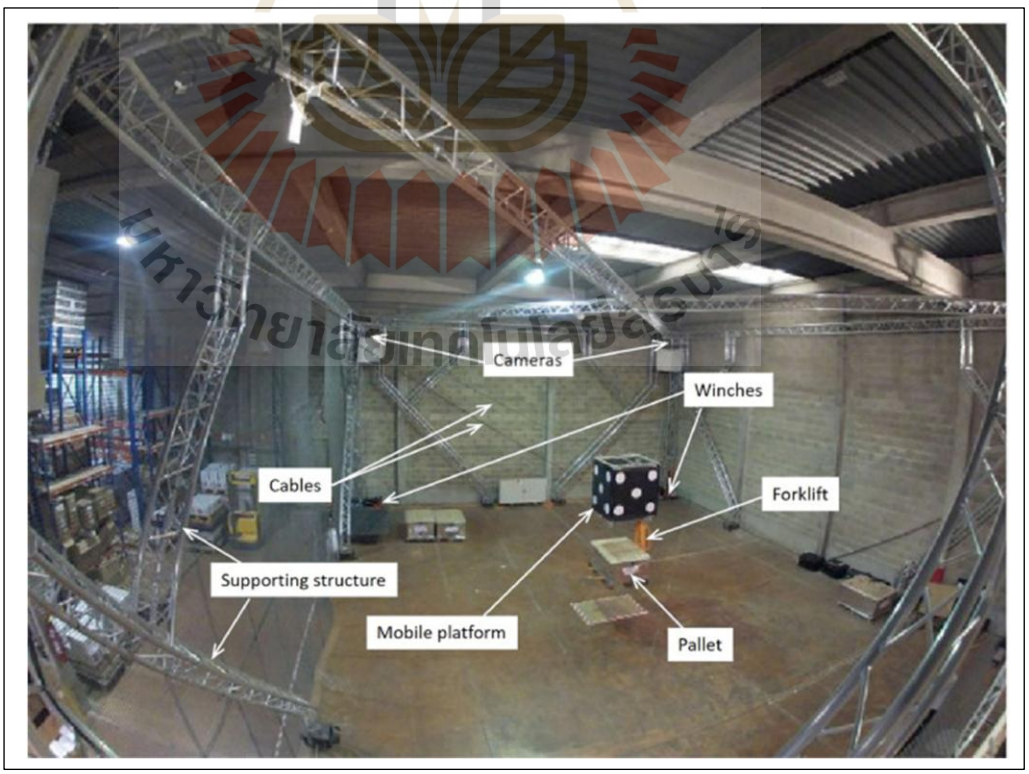
รูปที่ 3.3 แบบ Overview ของ Robot ที่ศึกษาข้อมูล

3.4.2 ข้อมูลโครงสร้าง ต้นแบบ Winch



รูปที่ 3.4 แบบ Winch ของ Robot ที่ศึกษาข้อมูลโครงสร้างภายใน

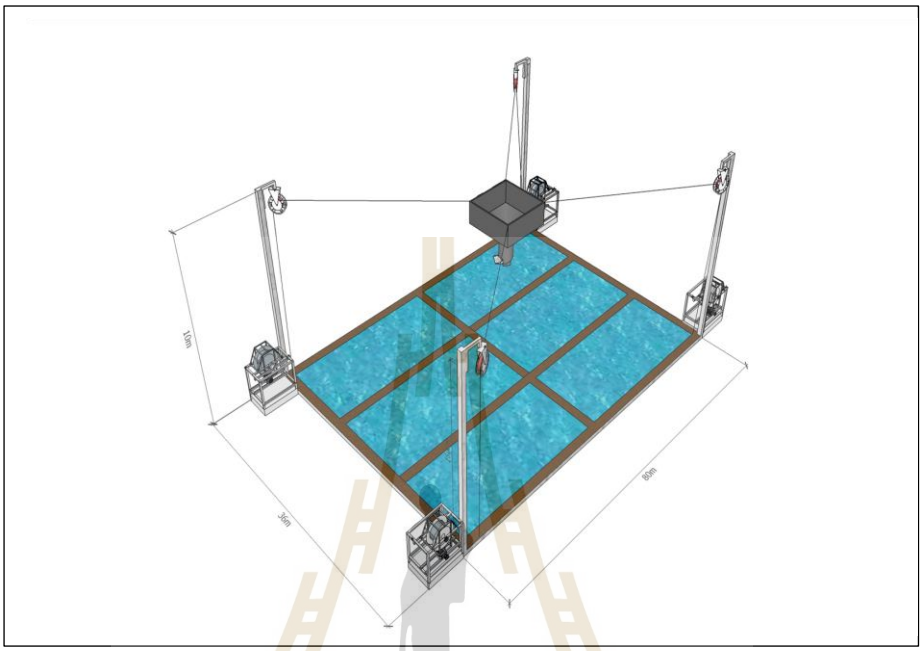
3.4.3 ข้อมูลโครงสร้าง ต้นแบบ Structure install



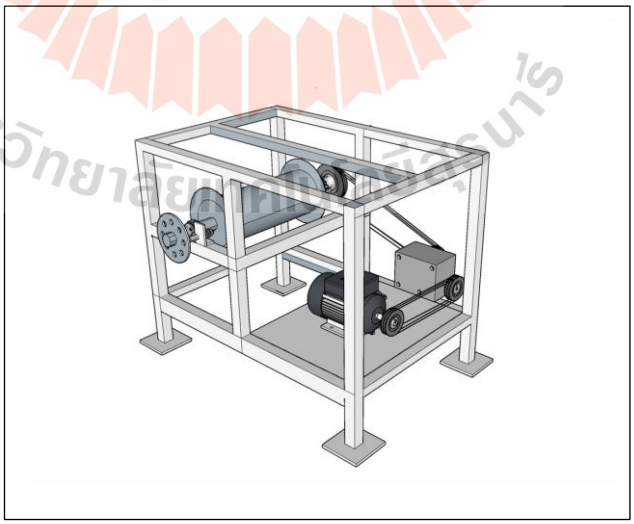
รูปที่ 3.5 แบบติดตั้งอุปกรณ์ ของ Robot ที่ศึกษาข้อมูล

3.5 ขั้นตอนการปฏิบัติงาน

Step 1/6 : เขียนแบบ Overview ของระบบหุ้นยนต์ กำหนดระยะตามพื้นที่จริงที่ใช้ทดลอง

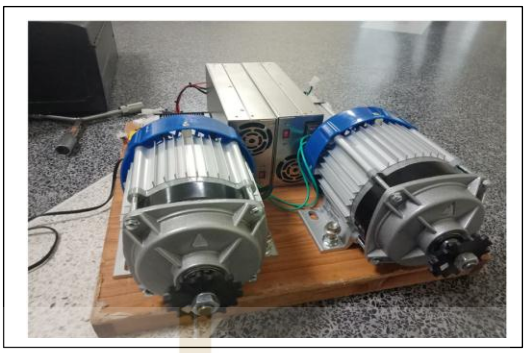


รูปที่ 3.6 Overview ของ หุ่นยนต์แบบ Cable-Driven Robot

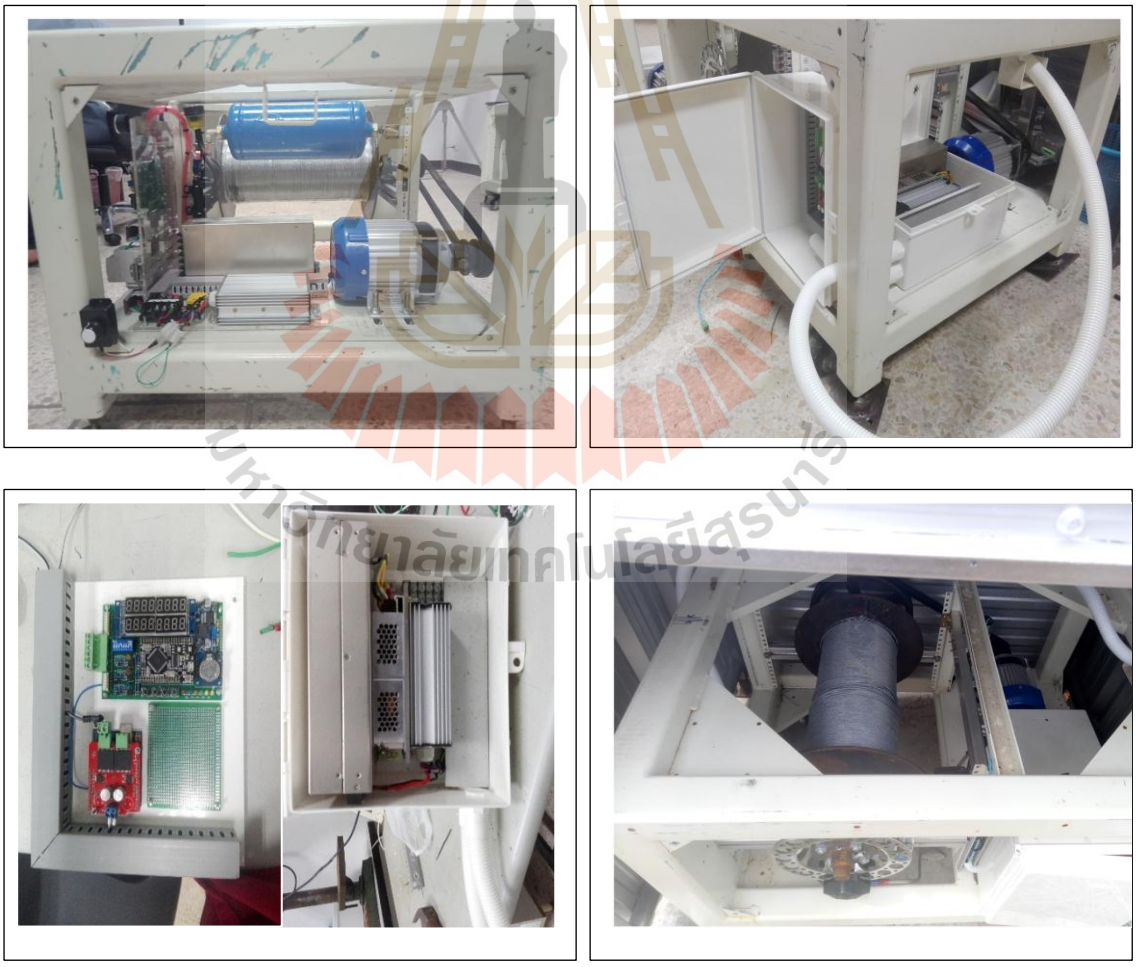


รูปที่ 3.7 แบบ Winch ของหุ่นยนต์แบบ Cable-Driven Robot

Step 2/6 : สร้าง Winch โดยเลือกใช้วัสดุที่มีอยู่จะไม่พยายมสร้างขึ้นมาใหม่

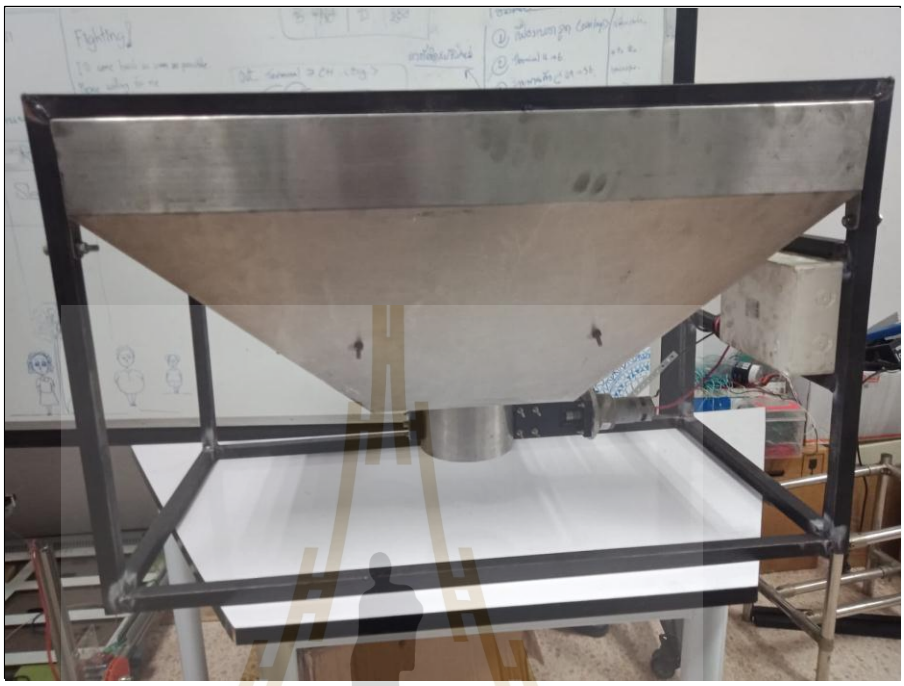


รูปที่ 3.8 ทดสอบ Motor

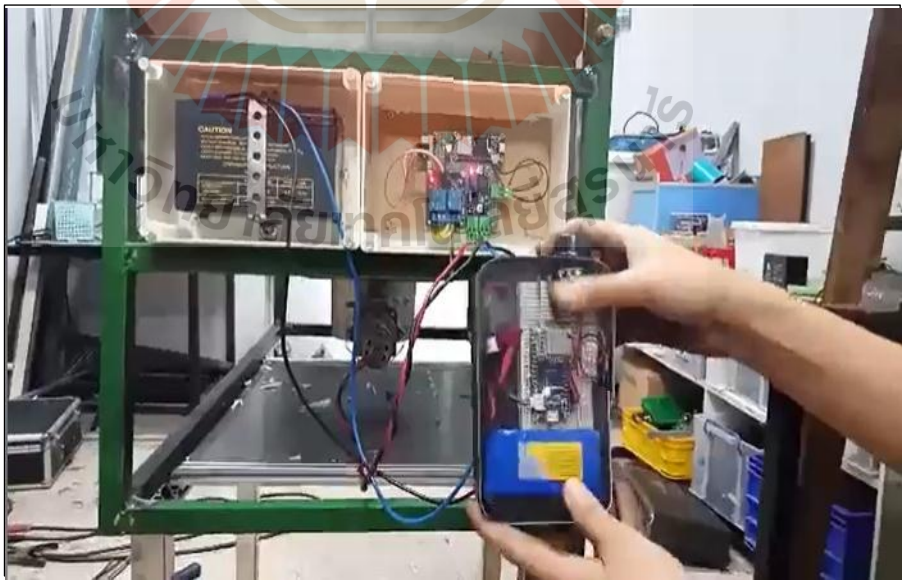


รูปที่ 3.9 สร้าง Winch และติดตั้งอุปกรณ์ที่ต้องใช้

Step 3/6 : สร้าง End-effector ของ Robot



รูปที่ 3.10 End-effector



รูปที่ 3.21 Test End-effector

Step 4/6 : เตรียมสถานที่ติดตั้งหุ่นยนต์

รูปที่ 3.12 ตั้งเสา

รูปที่ 3.13 เดินสายไฟและสายสื่อสาร

รูปที่ 3.14 ติดตั้งเบรกเกอร์สำหรับ Winch แต่ละตัว

รูปที่ 3.15 ต่อไฟฟ้าเข้าระบบ

Step 5/6 : ติดตั้งหุ่นยนต์

รูปที่ 3.16 ติดตั้ง Winch

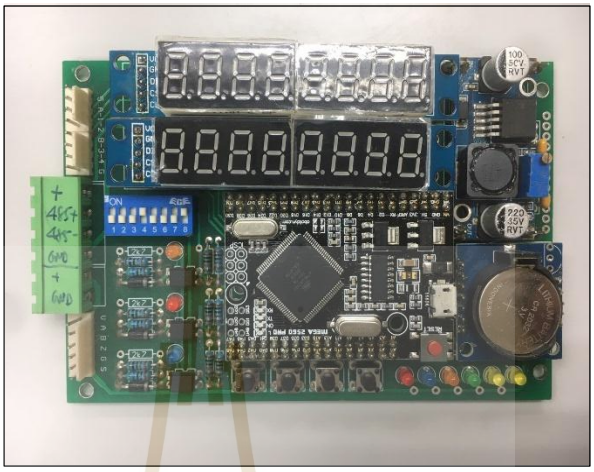
รูปที่ 3.18 ติดตั้งตัวนำ Cable

รูปที่ 3.18 ต่อไฟฟ้าให้ Winch

รูปที่ 3.19 ติดตั้ง End-effecter

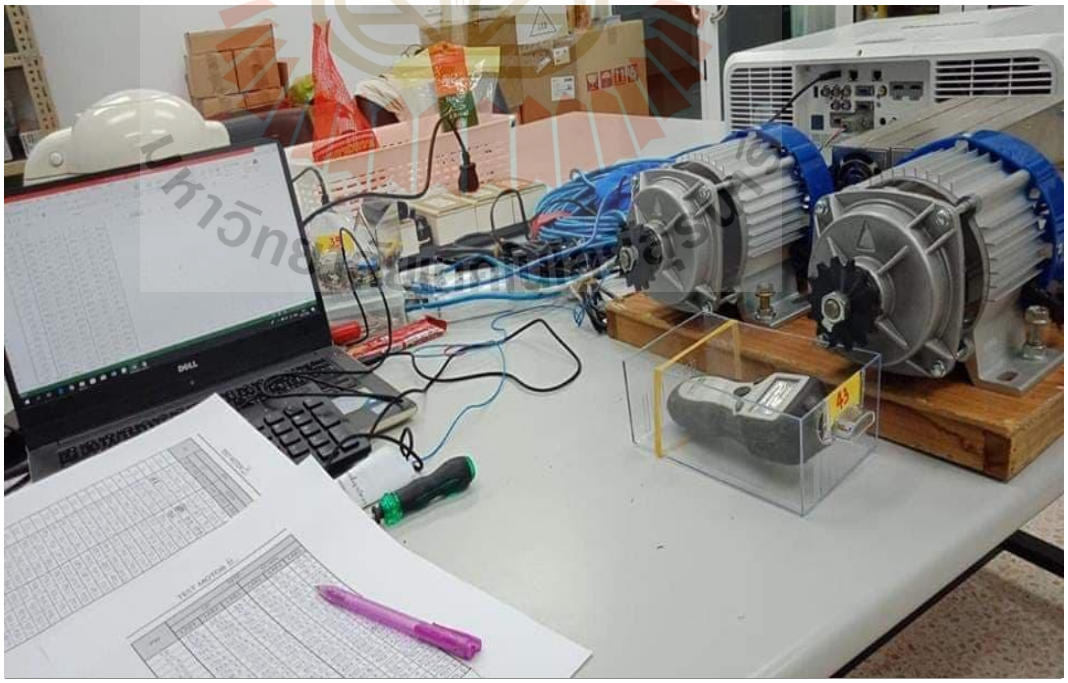
Step 5/6 : สร้างระบบควบคุมหุ่นยนต์และทำการเชื่อมต่อกับชุดรีโมทคอนโทร

Sub 1/4 : สร้างแผงวงจรควบคุม



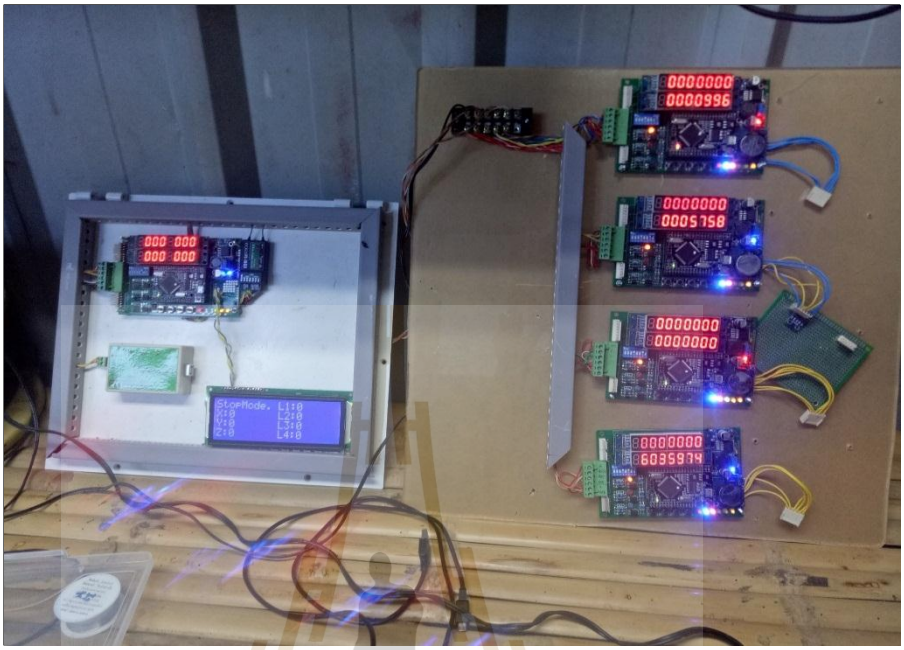
รูปที่ 3.20 ชุดคอนโทรล

Sub 2/4 : ทดสอบความเร็วและแรงดันที่ใช้ขับ Motor



รูปที่ 3.21 Test Speed Motor

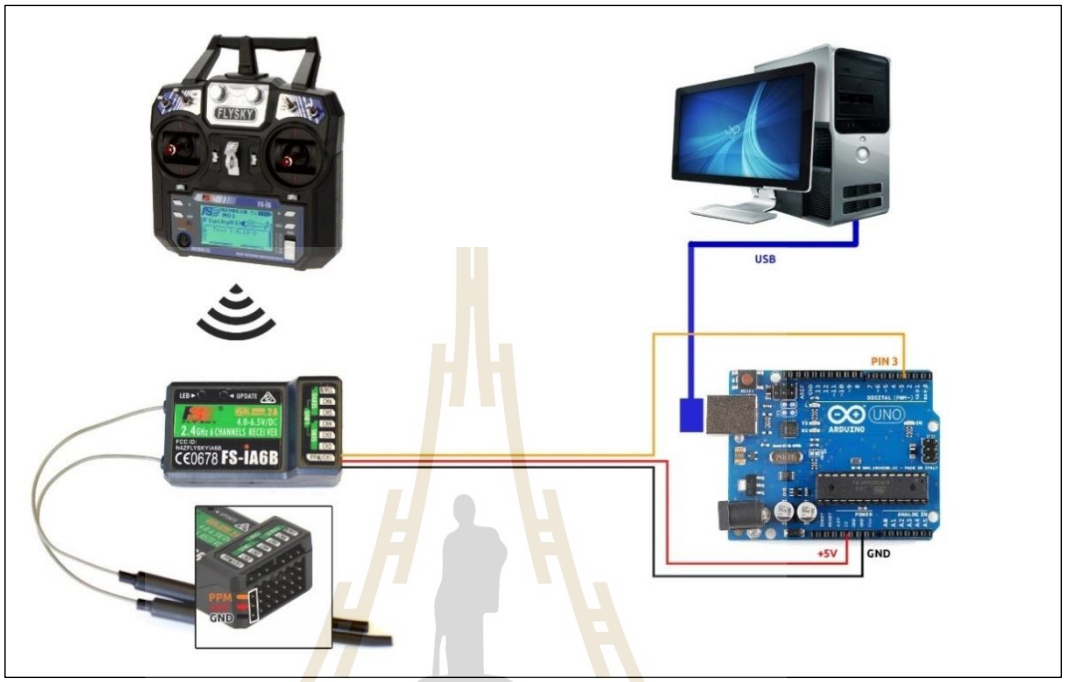
Sub 3/4 : ทดสอบการสื่อสารของระบบ



รูปที่ 3.22 Test Communication

รูปที่ 3.23 Test Slave Communication

Sub 4/4 : ทดสอบการเชื่อมต่อระหว่าง Arduino กับ Receiver และทดสอบรับ Parameters



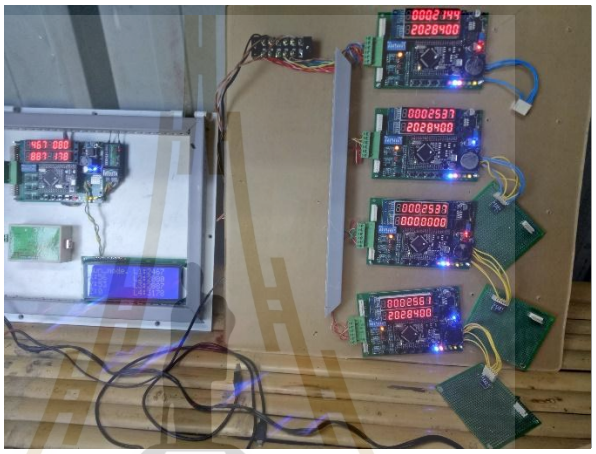
รูปที่ 3.24 Test Connect Receiver

รูปที่ 3.25 Test Blink LED By RC Radio

จากการทดลองขั้นนี้ pulseIn จะส่งค่าความกว้างของ pulse ออกมา หน่วยเป็น millisec ซึ่งอยู่ในช่วง 1,000 ถึง 2,000 ซึ่งค่ากลางของมันจะอยู่ที่ 1,500 ค่าได้ที่ออกมา ซึ่งสามารถแปลงให้อยู่ในช่วง -500 ถึง 500 ได้ด้วยวิธี map (x, 1000, 2000, -500, 500) ซึ่งจะ map ตัวเลขในช่วง 1000 ถึง 2000 ให้เป็น -500 ถึง 500 หลังจากนั้นทดสอบควบคุม แสดงผลออกทาง LED และ Serial Monitor

Step 6/6 : ทดสอบการทำงานของหุ่นยนต์

Sub 1/2 : การสื่อสาร



รูปที่ 3.26 ชุดคอนโทรล

Sub 2/2 : การเคลื่อนที่ของหุ่นยนต์

รูปที่ 3.27 Test Cable

บทที่ 4
ผลการศึกษาและวิเคราะห์ผล

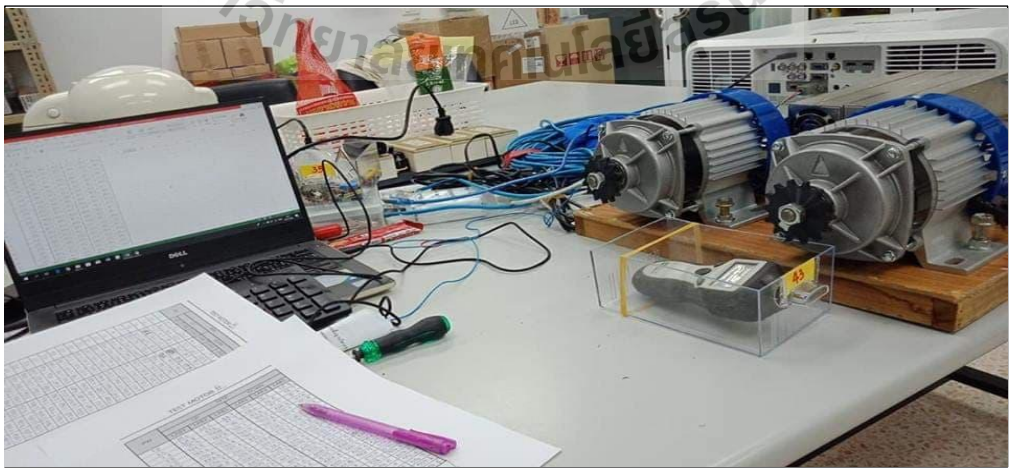
4.1 กล่าวนำ

งานวิจัยนี้ได้ทำการสร้างต้นแบบหุ่นยนต์แบบ Cable-Driven Robot ขนาด 20*60 เมตร เพื่อศึกษา วิเคราะห์ และแก้ไขปัญหาการออกแบบหุ่นยนต์ประเภทนี้ รวมถึงการเป็นต้นแบบในการศึกษาและพัฒนาต่อยอดให้หุ่นยนต์มีประสิทธิภาพสูง สามารถใช้งานได้จริงในอนาคต โดยทางผู้วิจัยมีแนวคิดที่จะสร้างต้นแบบหุ่นยนต์โดยประกอบด้วยการนำข้อมูลที่สังเคราะห์แล้วในบทที่ 3 มาสร้างเพื่อให้ได้มาซึ่งชิ้นงานที่สมบูรณ์ โดยบทนี้ในส่วนแรกผู้วิจัยจะขอกล่าวถึงประสิทธิภาพของอุปกรณ์ที่ได้เลือกใช้ และในส่วนถัดไปจะเป็นการอภิปรายเกี่ยวกับการทดสอบความเข้ากันได้ของอุปกรณ์

งานวิจัยในครั้งนี้ได้ดำเนินการสร้าง และเก็บข้อมูลต่าง ๆ ภายฟาร์มประมง มหาวิทยาลัยเทคโนโลยีสุรนารี และได้นำข้อมูลทั้งหมดมาทำการวิเคราะห์เพื่อหาแนวทางในการสร้าง และเพิ่มประสิทธิภาพของหุ่นยนต์ โดยผลการศึกษามีรายละเอียดดังต่อไปนี้

4.2 การคำนวณค่าฟังก์ชันวัตถุประสงค์

4.2.1 ทดสอบความเร็วของ Motor และช่วงแรงดันที่ที่ป้อน



รูปที่ 4.1 ชุดทดสอบ Motor

4.2.1.1 ผลการทดสอบ

ตารางที่ 4.1 ผลการทดสอบ Motor ชุด A

PW	Motor A							
	UP				DOWN			
	LAB 1	LAB 2	LAB 3	AVG	LAB 1	LAB 2	LAB 3	AVG
0.0	0	0	0	0	0	0	0	0
0.5	0	0	0	0	0	0	0	0
1.0	0	0	0	0	0	0	0	0
1.5	3281	3269	3514	111.822	3520	3543	3699	119.578
2.0	8470	7818	8182	271.889	6926	8481	8243	262.778
2.5	13840	13546	13364	452.778	13402	12977	13248	440.3
3	17270	17203	17494	577.411	17203	17384	17482	578.544
3.5	17906	18449	18180	605.944	18861	19038	19250	634.989
4	0	0	0	0	0	0	0	0
4.5	0	0	0	0	0	0	0	0
5.0	0	0	0	0	0	0	0	0

ตารางที่ 4.2 ผลการทดสอบ Motor ชุด B

PW	Motor B							
	UP				DOWN			
	LAB 1	LAB 2	LAB 3	AVG	LAB 1	LAB 2	LAB 3	AVG
0.0	0	0	0	0	0	0	0	0
0.5	0	0	0	0	0	0	0	0
1.0	0	0	0	0	0	0	0	0
1.5	3698	3545	3658	121.122	3401	3398	3530	114.767
2.0	6126	6418	6256	208.889	7009	7038	6860	232.3
2.5	8342	7947	8410	274.433	8054	8579	8719	281.689
3	7445	7130	7418	244.367	7712	7543	7279	250.378
3.5	7367	6869	6711	232.744	5778	6536	6263	206.411
4	0	0	0	0	0	0	0	0
4.5	0	0	0	0	0	0	0	0
5.0	0	0	0	0	0	0	0	0

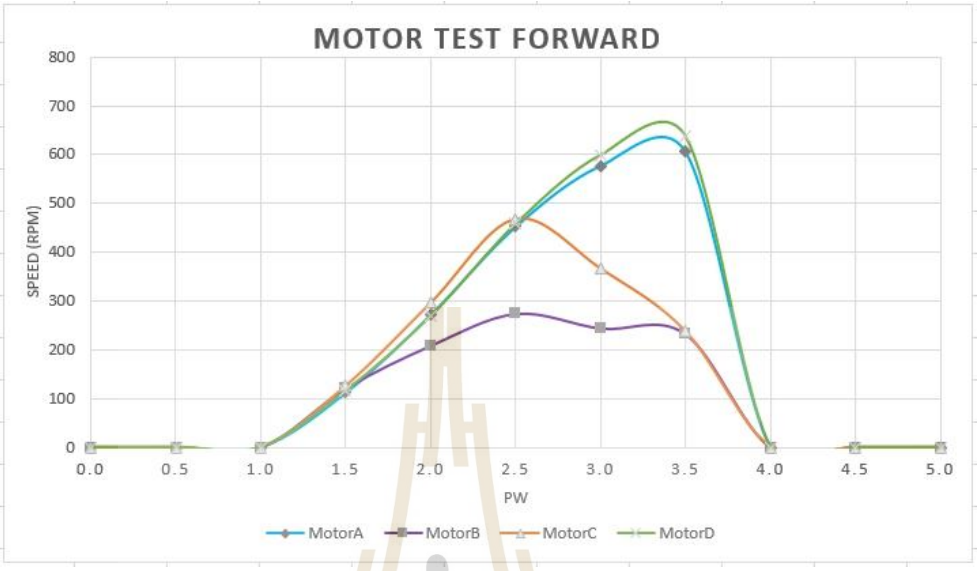
ตารางที่ 4.3 ผลการทดสอบ Motor ชุด C

PW	Motor C							
	UP				DOWN			
	LAB 1	LAB 2	LAB 3	AVG	LAB 1	LAB 2	LAB 3	AVG
0.0	0	0	0	0	0	0	0	0
0.5	0	0	0	0	0	0	0	0
1.0	0	0	0	0	0	0	0	0
1.5	3739	3757	3888	126.489	3848	3818	4371	133.744
2.0	9125	8420	9149	296.6	8534	8819	8325	285.311
2.5	14456	13612	14060	468.089	13449	14033	13927	460.1
3	11513	11243	10175	365.9	8688	10744	10432	331.822
3.5	7111	6270	7995	237.511	6427	6525	6889	220.456
4	0	0	0	0	0	0	0	0
4.5	0	0	0	0	0	0	0	0
5.0	0	0	0	0	0	0	0	0

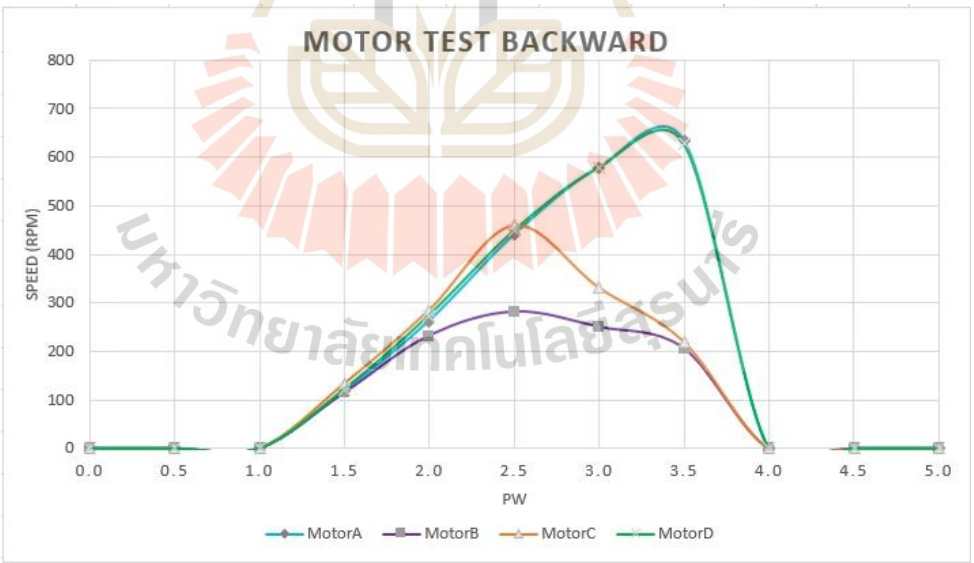
ตารางที่ 4.4 ผลการทดสอบ Motor ชุด D

PW	Motor D							
	UP				DOWN			
	LAB 1	LAB 2	LAB 3	AVG	LAB 1	LAB 2	LAB 3	AVG
0.0	0	0	0	0	0	0	0	0
0.5	0	0	0	0	0	0	0	0
1.0	0	0	0	0	0	0	0	0
1.5	3715	3614	3161	116.556	3700	3554	3709	121.811
2.0	7743	8030	8460	269.256	7782	8361	8542	274.278
2.5	14101	13765	13393	458.433	13580	13187	13582	448.322
3	18055	17757	18157	599.656	17206	17453	17400	578.433
3.5	19233	18717	19419	637.433	18554	18894	19022	627.444
4	0	0	0	0	0	0	0	0
4.5	0	0	0	0	0	0	0	0
5.0	0	0	0	0	0	0	0	0

4.2.1.2 การฟผลการทดสอบ



รูปที่ 4.2 กราฟทดสอบแบบ Forward

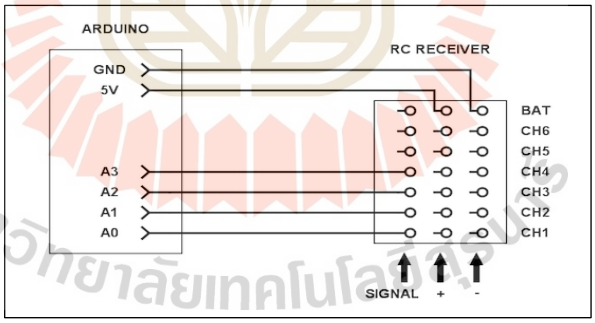


รูปที่ 4.3 กราฟทดสอบแบบ Backward

4.2.2 ทดสอบชุด Control

รูปที่ 4.4 ชุด Adiolink-at9s-10ch

ทำการเชื่อมต่อวงจร ดังรูปที่ 4.8 ในส่วนของภาครับสัญญาณและชุดควบคุม (Arduino)



รูปที่ 4.5 เชื่อมต่อ Arduino กับ Receiver

(-) เชื่อมต่อ Arduino GND กับหมุดในแถว GND ของเครื่องรับ

(+) เชื่อมต่อ Arduino V + กับหมุดในแถว V + ของเครื่องรับ

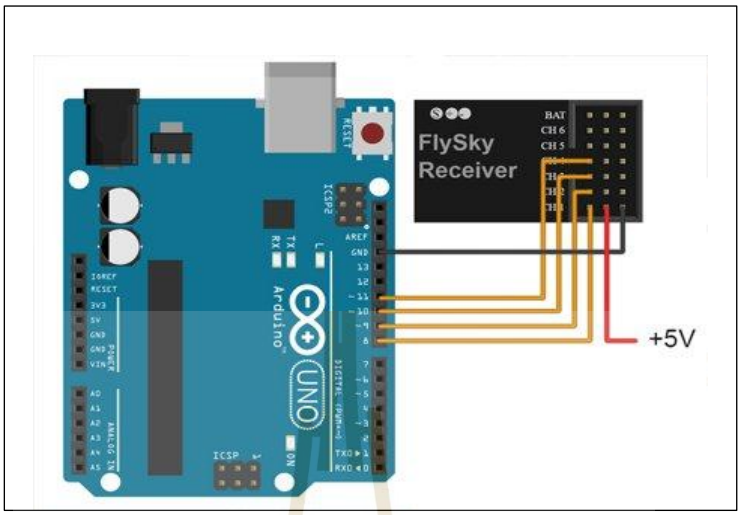
(Signal) สายสัญญาณของเครื่องรับ สำหรับป้อนให้ Arduino ใช้สั่งงานต่อไป

สำหรับการทดลองนี้ใช้วิธีรับค่าด้วย Interrupts โดยฟังก์ชันนี้จะถูกเรียกเมื่อมีสัญญาณสูง-ต่ำเข้ามารบกวนในระบบหรือมื่อใดมีการขัดจังหวะการทำงานใน Loop จะทำให้ได้ค่าสัญญาณออกมา

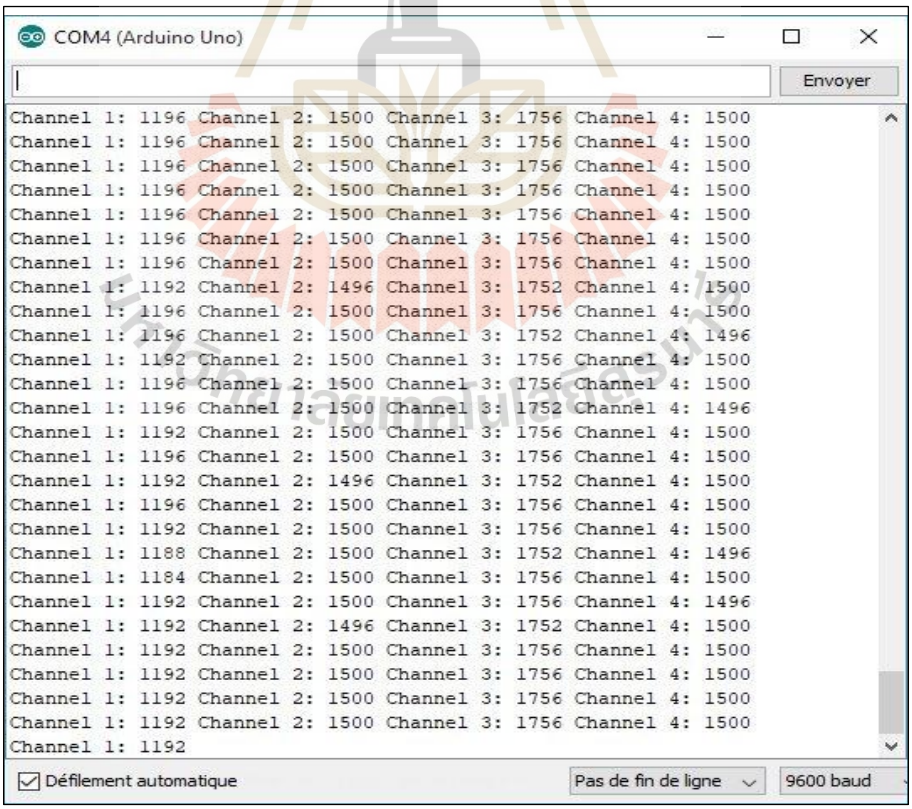
```
#include <EnableInterrupt.h> #define SERIAL_PORT_SPEED 57600
#define RC_NUM_CHANNELS  4   #define RC_CH1  0   #define RC_CH2  1
#define RC_CH3  2            #define RC_CH4  3   #define RC_CH1_INPUT  A0
#define RC_CH2_INPUT  A1     #define RC_CH3_INPUT  A2
#define RC_CH4_INPUT  A3
uint16_t rc_values[RC_NUM_CHANNELS]; uint32_t rc_start[RC_NUM_CHANNELS];
volatile uint16_t rc_shared[RC_NUM_CHANNELS];
void rc_read_values() {  noInterrupts();  memcpy(rc_values, (const void *)
rc_shared, sizeof(rc_shared));  interrupts(); }
void calc_input(uint8_t channel, uint8_t input_pin) {
  if (digitalRead(input_pin) == HIGH) {
    rc_start[channel] = micros();  } else {
    uint16_t rc_compare = (uint16_t)(micros() - rc_start[channel]);
rc_shared[channel] = rc_compare;  }}
void calc_ch1() { calc_input(RC_CH1, RC_CH1_INPUT); }
void calc_ch2() { calc_input(RC_CH2, RC_CH2_INPUT); }
void calc_ch3() { calc_input(RC_CH3, RC_CH3_INPUT); }
void calc_ch4() { calc_input(RC_CH4, RC_CH4_INPUT); }
void setup() {  Serial.begin(SERIAL_PORT_SPEED);
  pinMode(RC_CH1_INPUT, INPUT);  pinMode(RC_CH2_INPUT, INPUT);
  pinMode(RC_CH3_INPUT, INPUT);  pinMode(RC_CH4_INPUT, INPUT);
enableInterrupt(RC_CH1_INPUT, calc_ch1, CHANGE);
enableInterrupt(RC_CH2_INPUT, calc_ch2, CHANGE);
enableInterrupt(RC_CH3_INPUT, calc_ch3, CHANGE);
enableInterrupt(RC_CH4_INPUT, calc_ch4, CHANGE);}
void loop() {  rc_read_values();  Serial.print("CH1:");
Serial.print(rc_values[RC_CH1]); Serial.print("\t");  Serial.print("CH2:");
Serial.print(rc_values[RC_CH2]); Serial.print("\t");  Serial.print("CH3:");
```

รูปที่ 4.6 ทดสอบรับค่าด้วย Interrupt

4.2.3 ทดลองการรับค่าจาก Remote



รูปที่ 4.7 วงจร Arduino & Receiver



รูปที่ 4.8 ค่าจาก Remote ที่ Arduino รับได้รูปแบบการใช้งานคำสั่ง

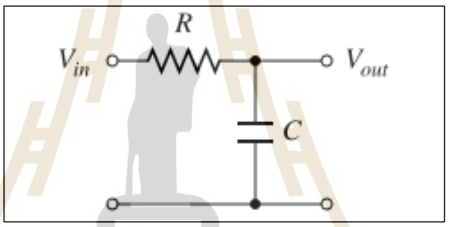
pulseIn (pin, value ,timeout)

โดยที่ pin คือ ขาดิจิตอลอินพุตที่ต้องการอ่านค่า

value คือ ลอจิกที่จะอ่านค่าความกว้างพัลส์ออกมา ในที่นี้จะอ่านที่ลอจิก 1

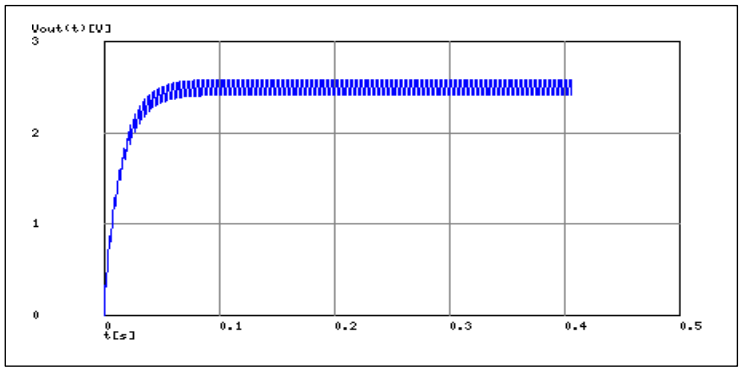
timeout คือ ช่วงเวลาที่กำหนดว่าถ้าไม่เกิดพัลส์ขึ้นมาในช่วงเวลากี่ไมโครวินาทีให้ไปทำงานคำสั่งถัดไป

ซึ่งในการทดสอบผู้พัฒนาอยากทราบว่า ถ้าต้องการนำเอาสัญญาณจากชุดรับสัญญาณนำไปใช้งานเลย โดยที่ไม่ผ่านชุดไมโครคอนโทรเลอร์ ด้วยวงจรรูปที่ 4.9 ใช้ capacitor 2.2 uF ตัวต้านทานขนาด 15K โอห์ม ทำการป้อนสัญญาณและวัดค่าที่ได้



รูปที่ 4.9 วงจร RC Low Pass

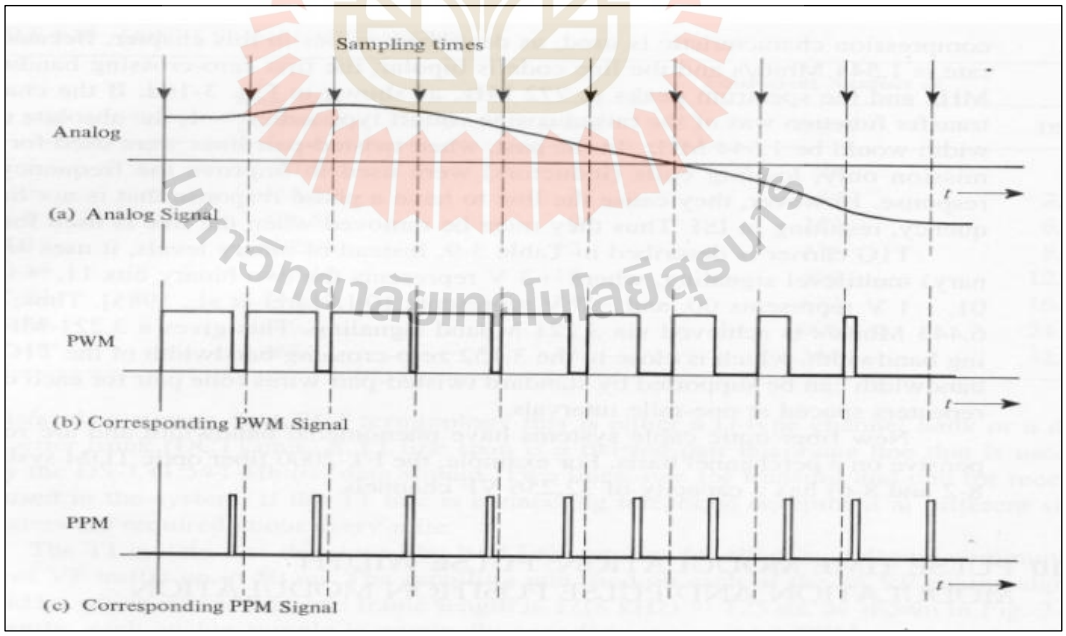
จากการทดลองดังกล่าวพบว่าค่าที่ออกมามีการ Swing ขึ้นลงอยู่ตลอดเวลาอันเป็นผลมาจากภาครับวิทยุ มีความถี่และคลื่นรบกวนในอากาศแทรกเข้ามาด้วยซึ่งถ้าเปรียบเทียบในรูปที่ 4.10 และค่าที่อ่านได้ในรูปที่ 4.9 จะเห็นความคลาดเคลื่อนของช่วงสัญญาณที่มีการเปลี่ยนแปลง ซึ่งส่งผล ต่อระบบการความคุมเพียงเล็กน้อย ดังนั้นค่าความคลาดคลื่นที่เกิดขึ้นจึงไม่ส่งผลกระทบต่อระบบหรือถ้าหากสัญญาณแกว่งขณะที่ระบบ Standby เราสามาเขียนโปแกรมเพื่อรองรับ Gap ได้



รูปที่ 4.10 สัญญาณเมื่อผ่านวงจร RC Low Pass

4.2.3.1 วิเคราะห์ค่าที่รับเข้ามาได้

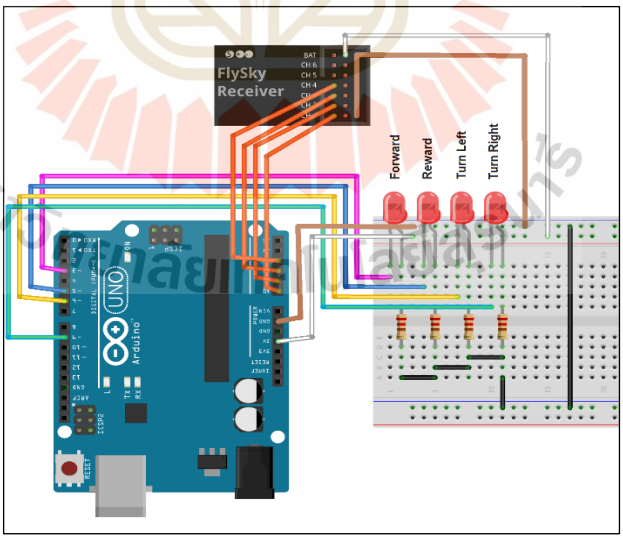
รูปแบบสัญญาณที่รับได้เป็นลักษณะของ PWM ซึ่งย่อมาจาก Pulse Width Modulation และ PPM ย่อมาจาก Pulse Position Modulation PWM เป็นเทคนิคที่ใช้ในการส่งข้อมูลในรูปแบบของความกว้างชีพจร (Pulse) ที่แตกต่างกันออกไปในแต่ละช่องสัญญาณ (Channel)



รูปที่ 4.11 Signal Analysis

รูปที่ 4.12 Signal from Oscillposcope PWM

4.2.3.2 ทดลองเขียนโปรแกรมคุม LED



รูปที่ 4.13 วงจรทดสอบด้วยหลอด LED

สรุป - อุปกรณ์ที่ใช้ทดลอง ได้แก่ Arduino, Adiolink at9s 10 ch, LED 4 หลอด

- การทดลองนี้สามารถ ควบคุมสถานะ LED จาก PWM ที่รับเข้ามาได้จากการดักค่าในแต่ละ Channel เพื่อนำออกไปควบคุม Output ที่ต้องการ

โปรแกรมตัวอย่างที่ใช้ในการดักค่า Pulse จากภาครับสัญญาณ

```
int ch1,ch2,ch3;
int count=0;
void setup() {
  pinMode(5, INPUT);
  pinMode(6, INPUT);
  pinMode(7, INPUT);
  Serial.begin(9600);
}
void loop() {
  ch1 = pulseIn(5, HIGH, 25000);
  ch2 = pulseIn(6, HIGH, 25000);
  ch3 = pulseIn(7, HIGH, 25000);
  Serial.print("No. ");
  Serial.print(count);
  Serial.print(" Channel 1 : ");
  Serial.print(ch1);
  Serial.print(" Channel 2 : ");
  Serial.print(ch2);
  Serial.print(" Channel 3 : ");
  Serial.print(ch3);
  Serial.println();
  count++;
```

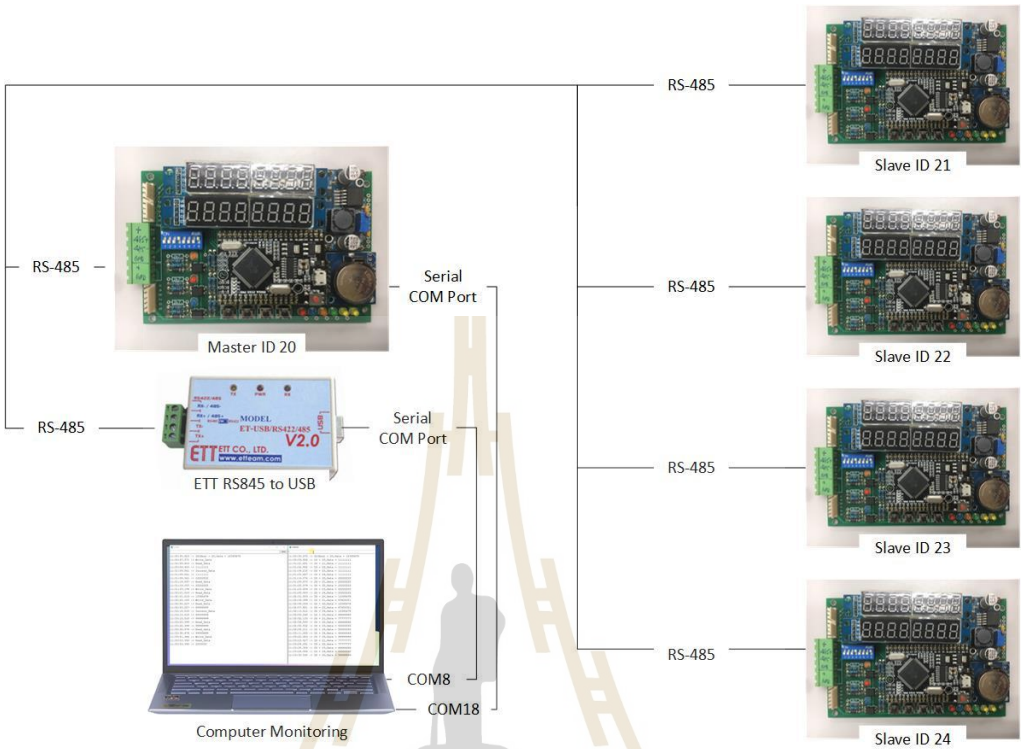
รูปที่ 4.14 โปรแกรมอ่านค่าจาก Remote

```
int ch1;
int ch2;
int ch3;
int count = 0;
   void setup() {
     pinMode(5, INPUT);
     pinMode(6, INPUT);
     pinMode(7, INPUT);
    Serial.begin(9600);
   }
   void loop() {
     ch1 = pulseIn(5, HIGH, 25000);
     ch2 = pulseIn(6, HIGH, 25000);
     ch3 = pulseIn(7, HIGH, 25000);
       if(ch1>1000){
       Serial.println("Left Switch: Engaged");
       }
       if(ch1<1000){
       Serial.println("Left Switch:Disengaged");
       }
     Serial.print("Right Stick X:");
     Serial.println(map(ch3, 1000,2000,-500,500));
     Serial.print("Right Stick Y:");
     Serial.println(map(ch2, 1000,2000,-500,500));
     Serial.println();
```

รูปที่ 4.15 โปรแกรมทดลองใช้งาน Data ที่รับเข้ามาจาก Remote

หลังจากที่สามารถเรียกใช้สัญญาณและนำไปควบคุม LED และแสดงออกหน้า Serial Monitor ได้แล้ว ลำดับต่อไปคือการสร้าง Algorithm สำหรับการควบคุมหุ่นยนต์ซึ่งจะแสดงในภาคผนวกต่อไป

4.2.4 ทดสอบการสื่อสารระหว่างอุปกรณ์

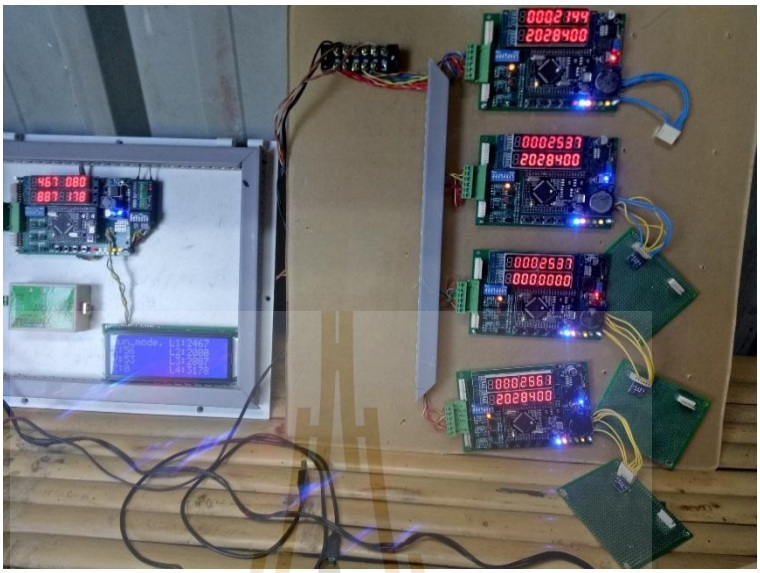


รูปที่ 4.16 Diagram Communication test

จากรูปที่ 4.8 ระบบประกอบด้วย ชุดแปลง RS485 เป็น USB 1 ชุด Master Control 1 ชุด และ Slave Control 4 ชุด ทดสอบการรับ-ส่งข้อมูลด้วยการ

- ส่งข้อมูลจาก Computer Monitoring แบบ Serial ไปยังชุด Master Control
- Master Control ส่งข้อมูลต่อไปยัง Slave Control
- ทดสอบรับ-ส่งข้อมูล ด้วยชุด ETT
- ทดสอบรับ-ส่งข้อมูล ด้วยชุด Master Control
- ทดสอบการสื่อสารด้วยระยะไกล้
- ทดสอบการสื่อสารด้วยระยะใช้งานจริง

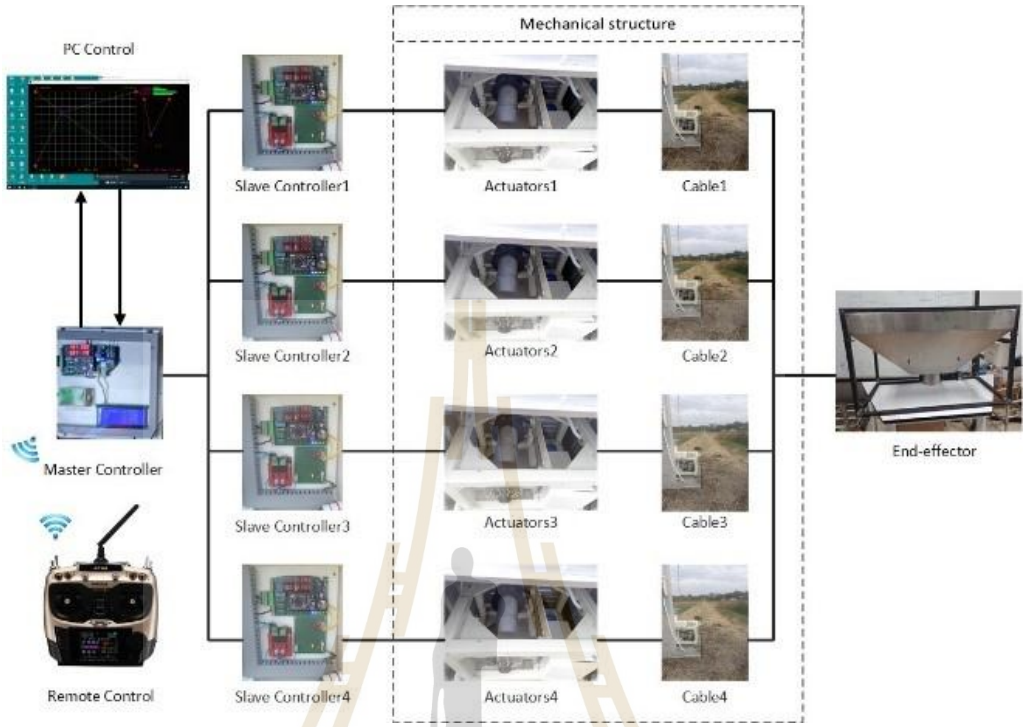
ผลการทดสอบ



รูปที่ 4.17 ทดสอบแบบภาพรวม

รูปที่ 4.18 ทดสอบกับระยะจริง

4.3 ทดสอบการเคลื่อนจริงของหุ่นยนต์



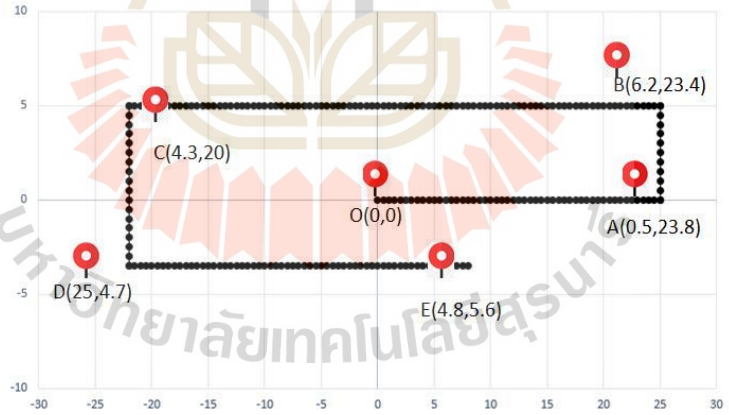
รูปที่ 4.19 Diagram Communication Controller Test

จากรูปที่ 4.11 การทำงานของระบบประกอบไปด้วย

- Remote Control ส่งคำสั่ง (x, y) ไป ให้ Master Control หรือ
- PC Control ส่งคำสั่ง (x, y) ไปให้ Master Control
- Master Control ทำการคำนวณหาความยาวของ Cable แล้วส่งความยาวไปยัง Slave Control
- Slave Control รับคำสั่งจาก Master Control แล้วสั่งให้ Motor ทำงาน
- Slave Control อ่านค่าจาก Proximity Sensor ที่ติดไว้กับ Winch เพื่อดูระยะที่ต้องสั่งงานให้ Motor เคลื่อนที่
- **ทดสอบการเคลื่อนที่**

ผลการทดสอบ

รูปที่ 4.20 หุ่นยนต์ที่ทดสอบ



รูปที่ 4.21 ผลการทดสอบ

ตารางที่ 4.5 ผลการทดสอบ

จุด	ระยะที่สั่งงาน (เมตร)	เคลื่อนที่จริง (เมตร)	ความคลาดเคลื่อน (เมตร)
A	A(0, 25)	A(0.5, 23.8)	(0.5, 1.2)
B	B(5, 25)	B(6.2, 6.23.4)	(1.2, 1.6)
C	C(5, -22)	C(4.3, -20)	(0.7, 2)
D	D(-4, -22)	D(4.7, -25)	(0.4, 3)
E	E(4, 8)	E(4.8, 5.6)	(0.8, 1.4)

4.4 ปัญหาที่พบ

1. หาวัสดุอุปกรณ์ข้อนข้างยาก เนื่องจากใช้ทำงานในสเกลที่ใหญ่
2. งานที่ต้องขึ้นรูปใหม่มีค่าใช้จ่ายเยอะ
3. Cable มีการพันกันเนื่องจากน้ำหนักในการเคลื่อนที่กับความเร็วของมอเตอร์ไม่สม่ำเสมอ ส่งผลให้เกิดความคาดเคลื่อนในการเคลื่อนที่มากขึ้น
4. มอเตอร์รับโหลดไม่ไหว

4.5 แนวทางการแก้ไข

ข้อที่ 1 หาวัสดุที่ใช้ทดแทนกัน

ข้อที่ 2 หาร้านเล็ก ๆ และพยายามทำเองให้แล็ปมหาวิทยาลัยยุบตัว รวมไปถึงมีความพยายามขัดผิดชั้นในลดน้ำหนังของ Robot ตัวในลงอีก

ข้อที่ 3 ในอนาคตต้องติด Sensor ตรวจจับให้มากขึ้น และปรับปรุงการออกแบบ Winch

ข้อที่ 4 เพิ่มเกียร์บล็อกมาใส่ในระบบ ทำให้การเคลื่อนที่ดีขึ้น

บทที่ 5
สรุปผลและข้อเสนอแนะ

5.1 สรุปงานวิจัย

งานวิจัยนี้เป็นการออกแบบและสร้างหุ่นยนต์แบบ Cable-Driven Robot ขนาด 20*60 เมตร สามารถเคลื่อนที่ได้ตามคำสั่งที่ป้อนให้กับหุ่นยนต์ด้วยรีโมทคอนโทร และสามารถทำงานได้ตามที่ผู้วิจัยต้องการในส่วนของการดำเนินการตรวจวัดและเก็บข้อมูลนำมาวิเคราะห์เพื่อหาความสามารถทำให้ได้ทราบถึงสมรรถนะ และประสิทธิภาพการใช้งานหุ่นยนต์ที่สร้างขึ้น ทำให้รู้ว่าสิ่งใดควรปรับปรุงสิ่งใดควรใส่ใจเป็นพิเศษ ไปถึงการเลือกใช้วัสดุอุปกรณ์ที่เหมาะสม อีกทั้งยังใช้ความรู้ที่เรียนมาได้ตรงตามวัตถุประสงค์ที่ตั้งไว้ได้อย่างชัดเจน

เมื่อทำการวิเคราะห์ข้อมูลการสร้างหุ่นยนต์แบบ Cable-Driven Robot ขนาด 20*60 เมตร จากทดลองสร้างด้วยข้อจำกันด้านงบประมาณ ทำให้มีข้อจำกัดในการหาวัสดุ และอุปกรณ์ที่ใช้ในการทดลองมีข้อจำกัดตามไปด้วยทำให้ผู้วิจัยต้องวัสดุที่มีอยู่ส่วนใหญ่ โดยแบบระบบออกเป็น 3 ส่วน คือ สารสื่อสารข้อมูลระหว่างอุปกรณ์ การเขียนโปรแกรมควบคุม การสร้างชุด Winch (เป็นชุด Actuator ของ Cable จำนวน 4 ชุด) และการวิเคราะห์ Kinematic ของหุ่นยนต์

การสื่อสารข้อมูลระหว่างอุปกรณ์ ในกาทดลงนี้ใช้การสื่อสารด้วยโปรโตคอล Modbus RS485 ในการสื่อสาร เนื่องจากเป็นโปรโตคอลที่ภาคอุตสาหกรรมยอมรับ และสื่อสารข้อมูลได้ในระยะไกล ผลการทดสอบส่งข้อมูลระหว่างชุด Control สามารถสื่อสารข้อมูลกันได้ และใช้ Remote Control ในการสื่อสารสั่งงานผ่าน User Interface ผลการทดลอง สามารถสื่อสารได้ทุกจุดในพื้นที่การทำงานของหุ่นยนต์

การเขียนโปรแกรมควบคุม เขียนด้วยภาษา C บนโปรแกรม Arduino IDE การเขียนโปรแกรมประกอบด้วยฟังก์ชัน การสื่อสารด้วย RS485 การสื่อสารผ่าน Receiver การคำนวณหาความยาวในการเคลื่อนที่ของสาย Cable ในแต่ละเส้น ไฟ Status การทำงานของอุปกรณ์ การบันทึกข้อมูลลง EPROM เป็นฟังก์ชันหลักในการเขียนโปรแกรมสำหรับการทดลองในครั้งนี้

การสร้างชุด Winch ด้วยงบประมาณที่จำกัด การสร้างชุด Winch ตึงต้องหาของที่มีอยู่มาตัดแต่งต่อเติมให้ได้ตามแบบท่าวไว้ ชุดWinch ประกอบด้วย Motor ใช้ Motor brushless dc 48v เกียร์-

บล็อกทดรอบ 1:10 ชุดเบรกใช้เบรกมอเตอร์ไซตมาทำ และ Sensor สำหรับการวัดระยะ Cable ใช้ Poximity sens จากการที่ได้ทำการออกแบบและทดลองสร้างหุ่นยนต์ลักษณะนี้ หุ่นยนต์สามารถเคลื่อนที่ได้ด้วยการสั่งงานผ่าน Remote Control การรับ-ส่งข้อมูลระหว่างบอร์ดทำได้ดี

และรับน้ำหนักได้ 20 กิโลกรัม แต่ระยะทางที่เคลื่อนที่เทียบกับที่คำนวณยังมีความคลาดเคลื่อนสูงอยู่มาก เนื่องจากการออกแบบ Winch ยังไม่ดีพอ และตัว End-effector ไม่ได้ติดตั้งชุด Sensor สำหรับการระบุตำแหน่งของหุ่นยนต์ การออกแบบและสร้างหุ่นยนต์ลักษณะนี้ยังขึ้นอยู่กับการเลือกใช้วัสดุอุปกรณ์ที่มีประสิทธิภาพ การติดตั้งชุดขับเคลื่อนในตำแหน่งใหม่ ๆ รวมถึงการเพิ่มลูกเล่นให้หุ่นยนต์นี้มีความน่าสนใจมากยิ่งขึ้น ตลอดจนชุดอุปกรณ์เสริมที่สามารถอ่านค่าและป้อนกลับมายังผู้ควบคุม (Monitor) ก็มีผลสำคัญที่จะช่วยประเมินการควบคุม ได้ดียิ่งขึ้นซึ่งจะช่วยเพิ่มประสิทธิภาพในการทำงานของหุ่นยนต์นี้ รวมไปถึงสามารถประยุกต์ใช้กับงานอื่น ๆ ที่เกี่ยวข้องได้ต่อไป

5.2 ข้อเสนอแนะ

จากวิจัยในครั้งนี้ ทำให้ได้ทราบถึงวิธีการเลือกหาอุปกรณ์ที่เหมาะสมที่สุดเพื่อใช้สร้างหุ่นยนต์แบบ Cable-Driven Robot ขนาด 20*60 เมตร ว่าผลลัพธ์ที่ได้นั้นจะมีรูปแบบเป็นไปตามที่ผู้วิจัยได้ออกแบบไว้ก่อนสร้างเพียงใด และมีข้อเสนอแนะเพื่อนำไปศึกษาต่อไป ดังนี้

- ในอนาคตต้องสร้างระบบควบคุมอัตโนมัติสามารถทำงานได้เองโดยไม่ต้องมีผู้ควบคุม
- ในการพัฒนาต่อต้องออแบบชุด Winch ให้สามารถให้มีความเป็นระเบียบในการม้วนเก็บในแต่ละรอบของการเคลื่อนที่
- ต้องมีการติดตั้งชุด Sensor เพิ่มเติม ที่ใช้ในการระบุตำแหน่งและตวามสูงของหุ่นยนต์เพื่อความละเอียดและความแม่นยำในการเคลื่อนที่

การศึกษาเกี่ยวกับการสร้างหุ่นยนต์แบบ Cable-Driven Robot ขนาด 20*60 เมตร ต้นแบบนี้ผู้วิจัยหวังเป็นอย่างยิ่งว่าจะเป็นประโยชน์ต่อการพัฒนาวงการหุ่นยนต์ และอุตสาหกรรมในประเทศไทยตลอดจน นักเรียนนิสิตนักศึกษาและผู้ที่มีความสนใจ เนื่องจากจะเป็นอีกแนวทางหนึ่งที่สำคัญซึ่งจะมีส่วนในการช่วยในการพัฒนาไอเดียการออกแบบ และการเลือกใช้วัสดุอุปกรณ์ที่เหมาะสมในการสร้างชิ้นงานสร้างหุ่นยนต์ หรือสิ่งประดิษฐ์อื่น ๆ และเป็นประโยชน์ต่อการพัฒนาต่อยอดที่ดีได้ อีกต่อไป

รายการอ้างอิง

Erika Ottaviano., Marco Ceccarelli., Alessio Paone., Giuseppe Carbone. (2005). "A Low-Cost Easy Operation 4-Cable Driven Parallel Manipulator", **IEEE International Conference on Robotics and Automation ICRA2005, Spain**. 4019-4024.

Mohammad M. Aref., Hamid D. Taghirad. (2008). "Geometrical Workspace Analysis of a Cable-Driven RedundantParallel Manipulator: KNTU CDRPM", **2008 IEEE/RSJ International Conference on Intelligent Robots and SystemsAcropolis Convention Center, France, Sept.** 22-26**:** 1958-1963.

Jean-Pierre Merlet., David Daney. (2010). "A portable, modular parallel wire crane for rescue operations", **IEEE International Conference on Robotics and Automation, ICRA 2010, Anchorage, Alaska, USA.** 01-10.

Ebert-Uphoff I., Voglewede P. A. (2004). "On the Connections Between Cable-Driven Robots, Parallel Manipulators and Grasping", **IEEEInternational Conference on Robotics and Automation ICRA'04, New Orleans.** 4521-4526.

Verhoeven R., Hiller M., Tadokoro S. (1998). "Workspace, Stiffness, Singularities and Classification of Tendon-Driven Stewart-Platforms", **International Symposium on Advances in Robot Kinematics ARK '98**, Strobl, Austria. 105-114.

Riechel A. T., Ebert-Uphoff I. (2004). "Force-Feasible Workspace Analysis for Underconstraiend, Point-Mass Cable Robots. "**IEEE International Conference on Robotics and Automation ICRA'04**, New Orleans. 4956-4962.

Fattah A., Agrawal, S.K. (2002). "Workspace and Design Analysis of Cable- Suspended Planar Parallel Robots", **ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference**, Montreal, paper MECH-34330.

Oh S.-R., Agrawal S.K. (2003). "Cable-Suspended Planar Parallel Robots with Redundant Cables : Controllers with Positive Cable Tensions", **IEEE International Conference on Robotics & Automation ICRA'03**, Taipei. 3023-3028.

Kossowski C., Notash L. (2002). "CAT4 (Cable Actuated Truss-4 Degrees of Freedom): A Novel 4 DOF Cable Actuated ParallelManipulator". **Journal of Robotic Systems**, Vol.19, n.12. 605-615.

Williams R.L.II, Gallina P., Vadia J. (2003). "Planar Translational Cable- Direct-Driven Robots", **Journal of Robotic Systems**, Vol. 20, n.3. 107-120.

Lytle A.M., Saidi K.S., Bostelman R.V., Stone W.C., Scott N.A. (2004). "Adapting a Teleoperated Device for Autonomous Control Using Three-Dimensional Positioning Sensors: Experiences with the NIST RoboCrane", **Automation in Construction**, Vol. 13, n.1. 101-118.

Kamamura, S., Kino, H., Won, C. (2000). "High-speed Manipulation by Using Parallel Wire-drive Robots", **Robotica**, Vol. 18, n.1. 13-21.

Morizono, T., Kurahashi, K., Kamamura, S. (1998). "Analysis and Control of a Force Display System Driven by Parallel Wire Mechanism", Robotica, Vol. 16. 551-563.

Jeong, J. W., Kim, S. H. and Kwak, Y. K. (1999). "Kinematics and Workspace Analysis of a Parallel Wire Mechanism for Measuring a Robot Pose", **Mechanism and Machine Theory**, Vol. 34, n.6. 825-841,

Ottaviano E., Ceccarelli M., Toti M., Avila Carrasco C. (2002). "CaTraSys (Cassino Traking System): A Wire System for Experimental Evaluation of Robot Workspace", **Journal of Robotics and Mechatronics**, Vol. 14, n.1. 78-87.

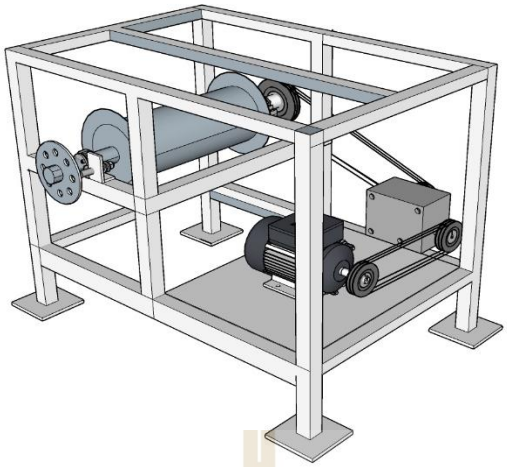
Pott, A., C. Meyer, and A. Verl. Large-scale assembly of solar power plants with parallel cable robots. (2010). In **Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)**. VDE.

Dietmaier, P. (1998). The stewart-gough platform of general geometry can have 40 real postures. In : **Advances in Robot Kinematics**. Kluwer Academic Publishers, Salzburg and Austria. 7-16.

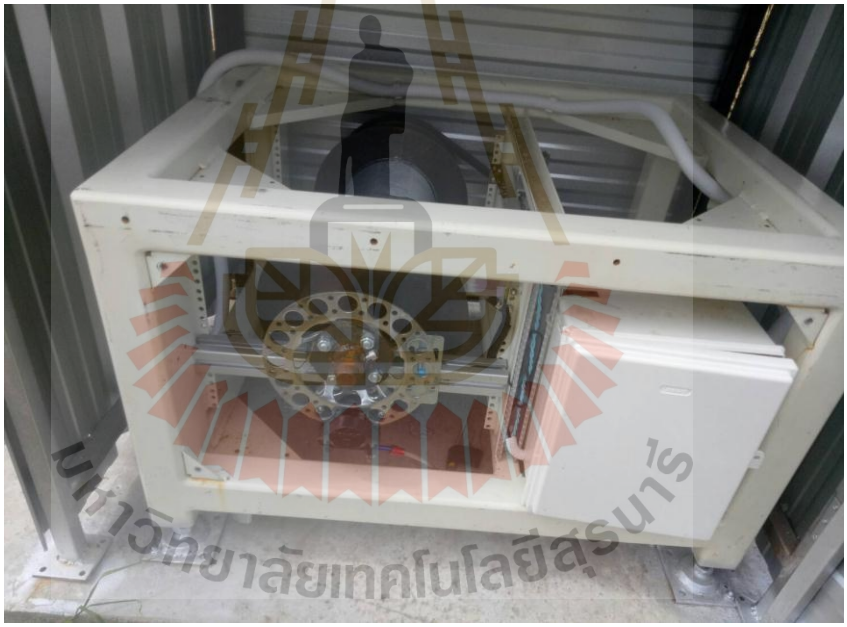
Husty, M.L. (1996). An algorithm for solving the direct kinematic of stewart-gough-type platforms. In : **Mechanism and Machine Theory**, vol. 31: 365-380

ภาคผนวก ก

Hardware & design



รูปที่ ก.1 Design winch of cable-driven



รูปที่ ก.2 Winch of cable-driven

ภาคผนวก ข

รายละเอียดโปรแกรมควบคุม

Slave Control

```
#define Pin_DIR A1
#define Pin_BRK A0

#define Port_ID1 24
#define Port_ID2 26
#define Port_ID3 28
#define Port_ID4 30
#define Port_ID5 25
#define Port_ID6 27
#define Port_ID7 29
#define Port_ID8 31

#define Port_Sw1 46
#define Port_Sw2 44
#define Port_Sw3 42
#define Port_Sw4 40

#define Port_LED1 43
#define Port_LED2 41
#define Port_LED3 39
#define Port_LED4 37
#define Port_LED5 35
#define Port_LED6 33

#define Port_UpDIN  23
#define Port_UpCLK  8
#define Port_UpEN   10
#define Port_DwDIN  22
#define Port_DwCLK  4
#define Port_DwEN   6

#define Pin_APWM 3
#define Pin_AIN1 5
#define Pin_AIN2 7

#define Pin_EncoderA  18    // INT3 > PD3 >> 18 (interrupt 5),
#define Pin_EncoderB  19    // INT2 > PD2 >> 19 (interrupt 4),
#define Pin_EncoderZ  38    // T0
```

รูปที่ ข.1 การประกาศตัวแปรในการควบคุม

```
#define Port_UpDIN  23
#define Port_UpCLK  8
#define Port_UpEN   10
#define Port_DwDIN  22
#define Port_DwCLK  4
#define Port_DwEN   6

#define Pin_APWM 3
#define Pin_AIN1 5
#define Pin_AIN2 7

#define Pin_EncoderA  18    // INT3 > PD3 >> 18 (interrupt 5),
#define Pin_EncoderB  19    // INT2 > PD2 >> 19 (interrupt 4),
#define Pin_EncoderZ  38    // T0

#define RS485_Ctrl 14
#define RS485_RxPin 17
#define RS485_TxPin 16
```

รูปที่ ข.2 การประกาศตัวแปรในการควบคุม (ต่อ)

```
void Port_Initial(void) {
  pinMode(Port_ID1, INPUT_PULLUP);
  pinMode(Port_ID2, INPUT_PULLUP);
  pinMode(Port_ID3, INPUT_PULLUP);
  pinMode(Port_ID4, INPUT_PULLUP);
  pinMode(Port_ID5, INPUT_PULLUP);
  pinMode(Port_ID6, INPUT_PULLUP);
  pinMode(Port_ID7, INPUT_PULLUP);
  pinMode(Port_ID8, INPUT_PULLUP);

  pinMode(Port_Sw1, INPUT_PULLUP);
  pinMode(Port_Sw2, INPUT_PULLUP);
  pinMode(Port_Sw3, INPUT_PULLUP);
  pinMode(Port_Sw4, INPUT_PULLUP);

  pinMode(Port_LED1, OUTPUT);
  pinMode(Port_LED2, OUTPUT);
  pinMode(Port_LED3, OUTPUT);
  pinMode(Port_LED4, OUTPUT);
  pinMode(Port_LED5, OUTPUT);
  pinMode(Port_LED6, OUTPUT);

  pinMode(Pin_EncoderA, INPUT);
  pinMode(Pin_EncoderB, INPUT);
  pinMode(Pin_EncoderZ, INPUT);

  pinMode(Pin_APWM, OUTPUT);          pinMode(RS485_Ctrl, OUTPUT);
  pinMode(Pin_AIN1, OUTPUT);          pinMode(Pin_DIR, OUTPUT);
  pinMode(Pin_AIN2, OUTPUT);          pinMode(Pin_BRK, OUTPUT);
}
```

รูปที่ ข.3 ฟังก์ชันระบุ ID Slave

```
void RS485_TxMode(void) {
  digitalWrite(RS485_Ctrl, HIGH);  // 0 = Rx
  delay(50);
}

void RS485_RxMode(void) {
  digitalWrite(RS485_Ctrl, LOW);  // 0 = Rx
  delay(50);
}
```

รูปที่ ข.4 ฟังก์ชันสื่อสาร RS485

```
int Read_myID(void) {
  int xAddress = 0;
  xAddress = xAddress * 2 + digitalRead(Port_ID1);
  xAddress = xAddress * 2 + digitalRead(Port_ID2);
  xAddress = xAddress * 2 + digitalRead(Port_ID3);
  xAddress = xAddress * 2 + digitalRead(Port_ID4);
  xAddress = xAddress * 2 + digitalRead(Port_ID5);
  xAddress = xAddress * 2 + digitalRead(Port_ID6);
  xAddress = xAddress * 2 + digitalRead(Port_ID7);
  xAddress = xAddress * 2 + digitalRead(Port_ID8);
  return xAddress ^ 0xff;
}
```

รูปที่ ข.5 ฟังก์ชันระบุ ID Slave

```
void ModeDisplay(void) {
  int Temp = abs(RunSpeed);
  modeDisp.setRow(0, 7, 0B10000001);
  modeDisp.setRow(0, 6, 0);
  modeDisp.setDigit(0, 5, my_ID / 16, false);
  modeDisp.setDigit(0, 4, my_ID % 16, true);
  modeDisp.setDigit(0, 0, Temp % 10, false);  Temp /= 10;
  modeDisp.setDigit(0, 1, Temp % 10, false);  Temp /= 10;
  modeDisp.setDigit(0, 2, Temp % 10, false);}
```

รูปที่ ข.6 ฟังก์ชันแสดง Mode การทำงานบนจอ Display 7 segment

```
void nRunCountDisplay(void) {
  long int Temp = nNowCounter;
  nCOuntDisp.setRow(0, 7, 0);
  if (Temp < 0) {
    Temp = Temp * -1;
    nCOuntDisp.setRow(0, 7, 1); }
  nCOuntDisp.setDigit(0, 0, Temp % 10, false);  Temp /= 10;
  nCOuntDisp.setDigit(0, 1, Temp % 10, false);  Temp /= 10;
  nCOuntDisp.setDigit(0, 2, Temp % 10, false);  Temp /= 10;
  nCOuntDisp.setDigit(0, 3, Temp % 10, false);  Temp /= 10;
  nCOuntDisp.setDigit(0, 4, Temp % 10, false);  Temp /= 10;
  nCOuntDisp.setDigit(0, 5, Temp % 10, false);  Temp /= 10;
  nCOuntDisp.setDigit(0, 6, Temp % 10, false);}
```

รูปที่ ข.6 ฟังก์ชันแสดง ตำแหน่ง การทำงานบนจอ Display 7 segment (ต่อ)

```
void Save_NVRAM(void) {
  long int  Temp = nNowCounter;
  byte TT = Temp % 100;    Temp = Temp / 100;
  byte UU = Temp % 100;    Temp = Temp / 100;
  byte SS = Temp % 100;    Temp = Temp / 100;
  byte XX = Temp % 100;
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.write(10); // Start Address Save
  Wire.write(XX); // set X
  Wire.write(SS); // set S
  Wire.write(UU); // set U
  Wire.write(TT); // set T
  Wire.endTransmission();}
```

รูปที่ ข.7 ฟังก์ชัน Save ลง EEPROM

```
void Read_NVRAM(void) {
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.write(10); // Start Address Save
  Wire.endTransmission();
  Wire.requestFrom(DS1307_I2C_ADDRESS, 4);
  byte XX = Wire.read(); // read X
  byte SS = Wire.read(); // read S
  byte UU = Wire.read(); // read U
  byte TT = Wire.read(); // read T
  nNowCounter = XX * 1000000 + SS * 10000 + UU * 100 + TT;
  Serial.print("Read = ");
  Serial.println(nNowCounter);
  Serial.print(" >> "); Serial.print(XX);
  Serial.print("-");    Serial.print(SS);
  Serial.print("-");    Serial.print(UU);
  Serial.print("-");    Serial.println(TT);
}
```

รูปที่ ข.8 ฟังก์ชัน Read EEPROM

```
void DC_MotorControl(void) {
 if (RunDirection == 2)  {
   digitalWrite(Pin_BRK, LOW);
   digitalWrite(Pin_DIR, LOW);
   if (counter > map(RunSpeed, 0, 500, 0, 3000))
      counter=counter-50;
      dac.setVoltage(counter, false);
 }
 if (RunDirection == 0)  {
   digitalWrite(Pin_BRK, HIGH);
   digitalWrite(Pin_DIR, HIGH);
   if (counter < map(RunSpeed, 0, 500, 0, 3000))
      counter=counter+50;
      dac.setVoltage(counter, false);

 }

if (RunDirection == 1) {
   digitalWrite(Pin_BRK, HIGH);
   digitalWrite(Pin_DIR, LOW);
  if (counter < map(RunSpeed, 0, 500, 0, 3000))
      counter=counter+50;
      dac.setVoltage(counter, false);
 }
}
```

รูปที่ ข.9 ฟังก์ชัน สั่งงาน Motor

```
void direction_run() {
  if (RunDirection == 1) {
    directionCount = +1;
    modeDisp.setRow(0, 3, B10011100);
    digitalWrite(Pin_DIR, LOW);
    digitalWrite(Port_LED5, LOW);
    digitalWrite(Port_LED6, HIGH);
  }
  if (RunDirection == 0) {
    directionCount = -1;
    modeDisp.setRow(0, 3, B11100010);
    digitalWrite(Pin_DIR, HIGH);
    digitalWrite(Port_LED5, HIGH);
    digitalWrite(Port_LED6, LOW);

  }


  digitalWrite(Port_LED4, HIGH);
  digitalWrite(Pin_BRK, HIGH);


}
if (RunDirection == 1) {
    digitalWrite(Pin_BRK, HIGH);
    digitalWrite(Pin_DIR, LOW);
   if (counter < map(RunSpeed, 0, 500, 0, 3000))
```

รูปที่ ข.10 ฟังก์ชัน เชค Forward or Backward

```
void setup() {
  Port_Initial();                          dac.begin(0x60);
  digitalWrite(Pin_BRK, LOW);
  Serial.begin(115200);                    Serial2.begin(9600);
  delay(1000);
  Serial.println("1/3 DS1307 Connecting .....");
  Wire.begin();
  setDS1307time(1, 2, 3, 4, 11, 12, 18);  //SS,MM,HH,Dy,Dt,Mt,Yr
  Start_DS1307();                          // Start DS1307-RTC
  Read_NVRAM();                            // nNowCounter
  Serial.println("2/3 Display .....");
  modeDisp.shutdown(0, false);
  modeDisp.setIntensity(0, 6);
  modeDisp.clearDisplay(0);
  nCOuntDisp.shutdown(0, false);
  nCOuntDisp.setIntensity(0, 6);
  nCOuntDisp.clearDisplay(0);
  my_ID = Read_myID();
  RunDirection = 2;
  ModeDisplay();
  nRunCountDisplay();
  attachInterrupt(5, EncoderCount, RISING);
  Serial.println("3/3 Ready .....");
  RS485_RxMode();
  directionCount = +1;}

  digitalWrite(Port_LED4, HIGH);
  digitalWrite(Pin_BRK, HIGH);


}
if (RunDirection == 1) {
    digitalWrite(Pin_BRK, HIGH);
    digitalWrite(Pin_DIR, LOW);
   if (counter < map(RunSpeed, 0, 500, 0, 3000))
       counter=counter+50;
       dac.setVoltage(counter, false);
  }
}
```

รูปที่ ข.11 ฟังก์ชัน Setup Arduino ใช้เรียกฟังก์ชันใช้งาน

```
void loop() {
  if (Serial.available()) {
    for (int i = 0; i < nBuffer; i++)
      ArrDataCMD[i] = ArrDataCMD[i + 1];
    ArrDataCMD[nBuffer] = Serial.read();
    if (ArrDataCMD[nBuffer] == '?') Help();
    if (ArrDataCMD[nBuffer] == '+') {up(10);Clear_RxBuff();}
    if (ArrDataCMD[nBuffer] == '-') {down(10);Clear_RxBuff();}
    if (ArrDataCMD[nBuffer] == 'w') {
      RunSpeed = 200; start = 1;
      Clear_RxBuff();
    }
    if (ArrDataCMD[nBuffer] == 'q') {
      start = 1;
      DC_MotorControl();
      ModeDisplay();
      RunSpeed = 0;
      counter = 0;
      directionCount = +1;
      Clear_RxBuff();
    }
  }
  if (Serial2.available()) {
    for (int i = 0; i < nBuffer; i++)
      ArrDataCMD[i] = ArrDataCMD[i + 1]; }}
    ArrDataCMD[nBuffer] = Serial2.read();
  }
  digitalWrite(Port_LED4, HIGH);
  digitalWrite(Pin_BRK, HIGH);

}
if (RunDirection == 1) {
    digitalWrite(Pin_BRK, HIGH);
    digitalWrite(Pin_DIR, LOW);
   if (counter < map(RunSpeed, 0, 500, 0, 3000))
        counter=counter+50;
        dac.setVoltage(counter, false);
  }
}
```

รูปที่ ข.12 ฟังก์ชัน Loop Arduino ใช้เรียกฟังก์ชันที่ทำงานซ้ำ ๆ

```
if (Serial2.available()) {
   for (int i = 0; i < nBuffer; i++)
     ArrDataCMD[i] = ArrDataCMD[i + 1];
   ArrDataCMD[nBuffer] = Serial2.read();
  }
  if (ArrDataCMD[nBuffer] == 0xD) {
   ArrDataCMD[nBuffer] = 0;
   cmd_Reboot();
   cmd_Liset();
   cmd_AStop();
   cmd_Start();
   cmd_LMove();
   cmd_ERead();
   cmd_CMove();
   cmd_PMove();
   cmd_SHome();
   cmd_GHome();
   ModeDisplay();
   nTargetDisplay();
   digitalWrite(Port_LED1, SrlBeat);
   SrlBeat = !SrlBeat;
  }
```

รูปที่ ข.13 ฟังก์ชัน Loop Arduino ใช้เรียกฟังก์ชันที่ทำงานซ้ำ ๆ (ต่อ)

```
if (nNowCounter == nTargetCounter) {
   RunDirection = 2;
   start = 0;
   DC_MotorControl();
   ModeDisplay();
   RunSpeed = 0;
   counter = 0;
   directionCount = +1;
  }
  if (digitalRead(Port_Sw1) == LOW)  {
   nTargetCounter--;
   RunSpeed = 200;
   nTargetDisplay();
  }
  if (digitalRead(Port_Sw2) == LOW)  {
   nTargetCounter++;
   RunSpeed = 200;
   nTargetDisplay();
  }
  if (digitalRead(Port_Sw3) == LOW)  start = 1;
  if (digitalRead(Port_Sw4) == LOW)  start = 0;
  if (loopCnt > 40000)
  {
   digitalWrite(Port_LED2, HeartBeat);
   if (RunDirection == 2)
     ModeDisplay();
   else
     nTargetDisplay();
   nRunCountDisplay();
   HeartBeat = !HeartBeat;
   loopCnt = 0;
  }
  loopCnt++;
  nRunCountDisplay();
}
```

รูปที่ ข.14 ฟังก์ชัน Loop Arduino ใช้เรียกฟังก์ชันที่ทำงานซ้ำ ๆ (ต่อ)

```
void up(int xxx) {
  nTargetCounter = nNowCounter + xxx;
  RunSpeed = 200;
  nTargetDisplay();
  start = 1;
}
void down(int xxx) {
  nTargetCounter = nNowCounter - xxx;
  RunSpeed = 200;
  nTargetDisplay();
  start = 1;
}
//#####################################################################
void EncoderCount() {
  cli();  //stop interrupts happening before we read pin values
  nNowCounter = nNowCounter + directionCount;
  sei();  //restart interrupts
}
void softwareReboot() {
  wdt_enable(WDTO_15MS);
  while (1) {}
}
```


รูปที่ ข.15 ฟังก์ชัน Loop Arduino ใช้เรียกฟังก์ชันที่ทำงานซ้ำ ๆ (ต่อ)

Master Control

```
#define Pin_DIR A1
#define Pin_BRK A0

#define Port_ID1 24
#define Port_ID2 26
#define Port_ID3 28
#define Port_ID4 30
#define Port_ID5 25
#define Port_ID6 27
#define Port_ID7 29
#define Port_ID8 31

#define Port_Sw1 46
#define Port_Sw2 44
#define Port_Sw3 42
#define Port_Sw4 40

#define Port_LED1 43
#define Port_LED2 41
#define Port_LED3 39
#define Port_LED4 37
#define Port_LED5 35
#define Port_LED6 33
```

รูปที่ ข.16 การประกาศตัวแปรในการควบคุม

```
#define Port_UpDIN  23
#define Port_UpCLK  8
#define Port_UpEN   10
#define Port_DwDIN  22
#define Port_DwCLK  4
#define Port_DwEN   6

#define Pin_APWM 3
#define Pin_AIN1 5
#define Pin_AIN2 7

#define Pin_EncoderA  18    // INT3 > PD3 >> 18 (interrupt 5),
#define Pin_EncoderB  19    // INT2 > PD2 >> 19 (interrupt 4),
#define Pin_EncoderZ  38    // T0

#define RS485_Ctrl 14
#define RS485_TxPin 16
#define RS485_RxPin 17
```

รูปที่ ข.17 การประกาศตัวแปรในการควบคุม

```
#include "_portAssign.c"
#include "LedControl.h"
#include "Wire.h"
#define DS1307_I2C_ADDRESS 0x68
#include <Adafruit_MCP4725.h>
uint32_t counter;
Adafruit_MCP4725 dac;
float ccounter = 0;
#define PMove "PMove"  // PMove +ID 255 99999999<Cr>
#define CMove "CMove"  // CMove +ID 255<Cr>
#define LMove "LMove"  // LMove ID 255 99999999<Cr>
#define GHome "GHome"  // GHome ID 255<Cr>
#define SHome "SHome"  // SHome ID<Cr>
#define ERead "ERead"  // ERead ID<Cr>
#define AStop "AStop"  // AStop<Cr>
#define Start "Start"  // Start<Cr>
#define nBuffer 23

int t3 = 0;
float k = 0, kt1 = 0, kt2 = 0, xx, xx1;

float val = 0;
float amplitude = 3000;
float angle = 0;
const float pi = 3.14159265;

LedControl modeDisp = LedControl(Port_UpDIN, Port_UpCLK, Port_UpEN, 1);
LedControl nCOuntDisp = LedControl(Port_DwDIN, Port_DwCLK, Port_DwEN, 1);
```

รูปที่ ข.18 การประกาศตัวแปรและเรียกใช้ฟังก์ชัน

```
int counterx = 0;
volatile long int nNowCounter = 0;
volatile int directionCount = 0;
long int nTargetCounter, nRunStep;
int RunDirection = 0, my_ID, Ctrl_ID = 0, RunSpeed = 0;
char ArrDataCMD[nBuffer + 1];

bool HeartBeat, SrlBeat;
unsigned int loopCnt;
```

รูปที่ ข.19 การประกาศตัวแปรและเรียกใช้ฟังก์ชัน

```
void Save_NVRAM(void) {
  long int  Temp = nNowCounter;
  byte TT = Temp % 100;    Temp = Temp / 100;
  byte UU = Temp % 100;    Temp = Temp / 100;
  byte SS = Temp % 100;    Temp = Temp / 100;
  byte XX = Temp % 100;
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.write(10); // Start Address Save
  Wire.write(XX); // set X
  Wire.write(SS); // set S
  Wire.write(UU); // set U
  Wire.write(TT); // set T
  Wire.endTransmission();
}
```

รูปที่ ข.20 ฟังก์ชัน Save_NVRAM

```
void Read_NVRAM(void) {
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.write(10); // Start Address Save
  Wire.endTransmission();
  Wire.requestFrom(DS1307_I2C_ADDRESS, 4);
  byte XX = Wire.read(); // read X
  byte SS = Wire.read(); // read S
  byte UU = Wire.read(); // read U
  byte TT = Wire.read(); // read T
  nNowCounter = XX * 1000000 + SS * 10000 + UU * 100 + TT;
  Serial.print("Read = ");
  Serial.println(nNowCounter);
  Serial.print(" >> "); Serial.print(XX);
  Serial.print("-");    Serial.print(SS);
  Serial.print("-");    Serial.print(UU);
  Serial.print("-");    Serial.println(TT);
}
```

รูปที่ ข.21 ฟังก์ชัน Read_NVRAM


```
void Start_DS1307(void) {
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.write(7); // set next input to start at the seconds register
  Wire.write(0xff); // set data
  Wire.endTransmission();
}
```

รูปที่ ข.22 ฟังก์ชัน Start_DS1307

```
void readDS1307time(byte *SS, byte *MM, byte *HH, byte *Dy, byte *Dt, byte *Mt,
byte *Yr) {
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.write(0);
  Wire.endTransmission();
  Wire.requestFrom(DS1307_I2C_ADDRESS, 7);
  *SS = bcdToDec(Wire.read() & 0x7f); // read seconds
  *MM = bcdToDec(Wire.read());  // read minutes
  *HH = bcdToDec(Wire.read() & 0x3f); // read hours
  *Dy = bcdToDec(Wire.read());  // read day of week (1=Sunday, 7=Saturday)
  *Dt = bcdToDec(Wire.read());  // read date (1 to 31)
  *Mt = bcdToDec(Wire.read());  // read month
  *Yr = bcdToDec(Wire.read());  // read year (0 to 99)
}
```

รูปที่ ข.23 ฟังก์ชัน readDS1307time

```
void printDateTime(void) {
  char datestring[20];
  byte SS, MM, HH, Dy, Dt, Mt, Yr;
  readDS1307time(&SS, &MM, &HH, &Dy, &Dt, &Mt, &Yr);
  snprintf_P(datestring, countof(datestring),
            PSTR("%02u/%02u/%04u %02u:%02u:%02u"),
            Dt, Mt, 2000 + Yr, HH, MM, SS);
  Serial.println(datestring);
}
```

รูปที่ ข.24 ฟังก์ชัน printDateTime

```
void displayTime() {
  byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
  readDS1307time(&second, &minute, &hour, &dayOfWeek, &dayOfMonth, &month,
&year);
  Serial.print(hour, DEC);
  Serial.print(":");
  if (minute < 10)     Serial.print("0");
  Serial.print(minute, DEC);                    Serial.print(":");
  if (second < 10)                              Serial.print("0");
  Serial.print(second, DEC);                    Serial.print(" ");
  Serial.print(dayOfMonth, DEC);                Serial.print("/");
  Serial.print(month, DEC);                     Serial.print("/");
  Serial.print(year, DEC);                      Serial.print(" Day of week: ");
  switch (dayOfWeek) {
    case 1:
      Serial.print("Sunday");          break;
    case 2:
      Serial.print("Monday");          break;
    case 3:
      Serial.print("Tuesday");         break;
    case 4:
      Serial.print("Wednesday");       break;
    case 5:
      Serial.print("Thursday");        break;
    case 6:
      Serial.print("Friday");          break;
    case 7:
      Serial.print("Saturday");        break;
  } }  Serial.println();}
}
  Serial.println();
}
```

รูปที่ ข.25 ฟังก์ชัน displayTime

```
void Port_Initial(void) {
  pinMode(Port_ID1, INPUT_PULLUP);
  pinMode(Port_ID2, INPUT_PULLUP);
  pinMode(Port_ID3, INPUT_PULLUP);
  pinMode(Port_ID4, INPUT_PULLUP);
  pinMode(Port_ID5, INPUT_PULLUP);
  pinMode(Port_ID6, INPUT_PULLUP);
  pinMode(Port_ID7, INPUT_PULLUP);
  pinMode(Port_ID8, INPUT_PULLUP);
  pinMode(Port_Sw1, INPUT_PULLUP);
  pinMode(Port_Sw2, INPUT_PULLUP);
  pinMode(Port_Sw3, INPUT_PULLUP);
  pinMode(Port_Sw4, INPUT_PULLUP);
  pinMode(Port_LED1, OUTPUT);
  pinMode(Port_LED2, OUTPUT);
  pinMode(Port_LED3, OUTPUT);
  pinMode(Port_LED4, OUTPUT);
  pinMode(Port_LED5, OUTPUT);
  pinMode(Port_LED6, OUTPUT);
  pinMode(Pin_EncoderA, INPUT);
  pinMode(Pin_EncoderB, INPUT);
  pinMode(Pin_EncoderZ, INPUT);
  pinMode(Pin_APWM, OUTPUT);
  pinMode(Pin_AIN1, OUTPUT);
  pinMode(Pin_AIN2, OUTPUT);
  pinMode(RS485_Ctrl, OUTPUT);
  pinMode(Pin_DIR, OUTPUT);
  pinMode(Pin_BRK, OUTPUT);
}
```

รูปที่ ข.26 ฟังก์ชัน Port_Initial

```
void RS485_TxMode(void)
{ digitalWrite(RS485_Ctrl, HIGH);  // 0 = Rx
  delay(50);
}

void RS485_RxMode(void)
{ delay(50);
  digitalWrite(RS485_Ctrl, LOW);  // 0 = Rx
}
```

รูปที่ ข.27 ฟังก์ชัน RS485_TxMode และ RS485_RxMode

```
int Read_myID(void) {
  int xAddress = 0;
  xAddress = xAddress * 2 + digitalRead(Port_ID1);
  xAddress = xAddress * 2 + digitalRead(Port_ID2);
  xAddress = xAddress * 2 + digitalRead(Port_ID3);
  xAddress = xAddress * 2 + digitalRead(Port_ID4);
  xAddress = xAddress * 2 + digitalRead(Port_ID5);
  xAddress = xAddress * 2 + digitalRead(Port_ID6);
  xAddress = xAddress * 2 + digitalRead(Port_ID7);
  xAddress = xAddress * 2 + digitalRead(Port_ID8);
  return xAddress ^ 0xff;
}
```

รูปที่ ข.28 ฟังก์ชัน RS485_TxMode และ Read_myID

```
void cmd_LMove(void) {// LMove ID 255 99999999<Cr>
  if (IS_Command(LMove) != 0) {
    Serial.println("cmd_LMove"); int j = IS_Command(LMove);
    Ctrl_ID = (ArrDataCMD[j + 6] - '0') * 16;
    Ctrl_ID = Ctrl_ID + (ArrDataCMD[j + 7] - '0');
    bool bStatus = (Ctrl_ID == my_ID);
    bStatus = bStatus && (ArrDataCMD[j + 5] == ' ');
    bStatus = bStatus && (ArrDataCMD[j + 8] == ' ');
    bStatus = bStatus && (ArrDataCMD[j + 12] == ' ');
    if (bStatus == true) {
      RunSpeed  = ArrDataCMD[j + 9] - '0';
      RunSpeed  = RunSpeed * 10 + ArrDataCMD[j + 10] - '0';
      RunSpeed  = RunSpeed * 10 + ArrDataCMD[j + 11] - '0';
      nRunStep = ArrDataCMD[j + 13] - '0';
      nRunStep = nRunStep * 10 + ArrDataCMD[j + 14] - '0';
      nRunStep = nRunStep * 10 + ArrDataCMD[j + 15] - '0';
      nRunStep = nRunStep * 10 + ArrDataCMD[j + 16] - '0';
      nRunStep = nRunStep * 10 + ArrDataCMD[j + 17] - '0';
      nRunStep = nRunStep * 10 + ArrDataCMD[j + 18] - '0';
      nRunStep = nRunStep * 10 + ArrDataCMD[j + 19] - '0';
      nRunStep = nRunStep * 10 + ArrDataCMD[j + 20] - '0';
      nTargetCounter = nRunStep;
      if (nNowCounter == nRunStep) RunDirection = 0;
      if (nNowCounter > nRunStep)  RunDirection = 10;
      if (nNowCounter < nRunStep)  RunDirection = 01;
    }
    Display_RxBuff(j); Clear_RxBuff();
  }
}
```

รูปที่ ข.29 ฟังก์ชัน cmd_Lmove

```
void cmd_ERead(void) {//=== ERead ID<Cr> ===
  if (IS_Command(ERead) != 0) {
    Serial.println("cmd_ERead");
    int j = IS_Command(ERead);
    Ctrl_ID = (ArrDataCMD[j + 6] - '0') * 16;
    Ctrl_ID = Ctrl_ID + (ArrDataCMD[j + 7] - '0');
    bool bStatus = (Ctrl_ID == my_ID);
    bStatus = bStatus && (ArrDataCMD[j + 5] == ' ');
    if (bStatus == true) {
      Read_NVRAM();
      Serial.print("Enc.Read = ");
      Serial.print(nNowCounter);
      Serial.print(",");
      if (RunDirection < 10) Serial.print("0");
      Serial.println(RunDirection);
      RS485_TxMode();
      Serial2.print("$");
      Serial2.print(my_ID, HEX);
      Serial2.print(":");
      Serial2.print(nNowCounter);
      Serial2.print(":");
      if (RunDirection < 10) Serial2.print("0");
      else Serial2.print(RunDirection);
      Serial2.print(":");Serial2.print(1);
      Serial2.print(":");Serial2.print(2);
      Serial2.print(":");      Serial2.println(3);
      RS485_RxMode();
    } Display_RxBuff(j);    Clear_RxBuff();
  }
}
```

รูปที่ ข.30 ฟังก์ชัน cmd_Eread

```
//=== SHome ID<CR> ===
void cmd_SHome(void) {
  if (IS_Command(SHome) != 0) {
    Serial.println("cmd_SHome");
    int j = IS_Command(SHome);
    Ctrl_ID = (ArrDataCMD[j + 6] - '0') * 16;
    Ctrl_ID = Ctrl_ID + (ArrDataCMD[j + 7] - '0');
    bool bStatus = (Ctrl_ID == my_ID);
    bStatus = bStatus && (ArrDataCMD[j + 5] == ' ');
    if (bStatus == true) {
      nRunStep = 0;
      nNowCounter = 0;
      nTargetCounter = 0;
      Save_NVRAM();
    }
    Display_RxBuff(j);
    Clear_RxBuff();
  }
}
```

รูปที่ ข.31 ฟังก์ชัน SHome

```
void cmd_Start(void) {
  if (IS_Command(Start) != 0)
  { DC_MotorControl();
    Save_NVRAM();
    Serial.println("cmd_Start");
    Clear_RxBuff();
  }}
```

รูปที่ ข.32 ฟังก์ชัน cmd_Start

```
void cmd_AStop(void) {
  if (IS_Command(AStop) != 0)  {
    RunDirection = 0;
    dac.setVoltage(0, false);
    Save_NVRAM();
    Serial.println("cmd_AStop");
    Clear_RxBuff();
    //while(1);
  }
}
```

รูปที่ ข.33 ฟังก์ชัน cmd_AStop

```
int IS_Command(String searchString) {
  int result = 0;
  for (int i = 0; i < nBuffer; i++) {
    bool bCheck = true;
    bCheck = bCheck && (searchString[0] == ArrDataCMD[i]);
    bCheck = bCheck && (searchString[1] == ArrDataCMD[i + 1]);
    bCheck = bCheck && (searchString[2] == ArrDataCMD[i + 2]);
    bCheck = bCheck && (searchString[3] == ArrDataCMD[i + 3]);
    bCheck = bCheck && (searchString[4] == ArrDataCMD[i + 4]);
    if (bCheck == true) {
      result = i;
      i = nBuffer + 5;
    }
  }
  return result;
}
```

รูปที่ ข.34 ฟังก์ชัน IS_Command

```
void Help(void) {
  Serial.println("================================");
  Serial.println("--------------------------------");
  Serial.println(" PMove +ID 255 99999999<Cr>");
  Serial.println(" CMove +ID 255<Cr>");
  Serial.println(" LMove ID 255 99999999<Cr>");
  Serial.println(" GHome ID 255<Cr>");
  Serial.println(" SHome ID<Cr>");
  Serial.println(" ERead ID<Cr>");
  Serial.println(" AStop<Cr>");
  Serial.println(" Start<Cr>");
  Serial.println("--------------------------------");
  Clear_RxBuff();
}
```

รูปที่ ข.35 ฟังก์ชัน Help

```
void cmd_AStop(void) {
  if (IS_Command(AStop) != 0)  {
    RunDirection = 0;
    dac.setVoltage(0, false);
    Save_NVRAM();
    Serial.println("cmd_AStop");
    Clear_RxBuff();
    //while(1);
  }
```

รูปที่ ข.36 ฟังก์ชัน cmd_Astop

```
void setup() {
  Port_Initial();
  dac.begin(0x60);
  digitalWrite(Pin_BRK, LOW);
  Serial.begin(115200);
  Serial2.begin(9600);
  Serial.println("1/3 DS1307 Connecting .....");
  Wire.begin();
  setDS1307time(1, 2, 3, 4, 11, 12, 18);  //SS,MM,HH,Dy,Dt,Mt,Yr
  Start_DS1307();   // Start DS1307-RTC
  Read_NVRAM();     // nNowCounter
  Serial.println("2/3 Display .....");
  modeDisp.shutdown(0, false);
  modeDisp.setIntensity(0, 6);
  modeDisp.clearDisplay(0);
  nCOuntDisp.shutdown(0, false);
  nCOuntDisp.setIntensity(0, 6);
  nCOuntDisp.clearDisplay(0);
  my_ID = Read_myID();
  RunDirection = 0;
  DC_MotorControl();
  ModeDisplay();
  nRunCountDisplay();
  attachInterrupt(5, EncoderCount, RISING);
  Serial.println("3/3 Ready .....");
  RS485_RxMode();
}
```

รูปที่ ข.37 เรียกใช้ฟังก์ชัน Setup

```
void loop() {
  nRunCountDisplay();
  if (Serial.available()) {
    for (int i = 0; i < nBuffer; i++)
      ArrDataCMD[i] = ArrDataCMD[i + 1];
    ArrDataCMD[nBuffer] = Serial.read();
    if (ArrDataCMD[nBuffer] == '?') Help();
  }
  if (Serial2.available()) {
    for (int i = 0; i < nBuffer; i++)
      ArrDataCMD[i] = ArrDataCMD[i + 1];
    ArrDataCMD[nBuffer] = Serial2.read();
  }
  if (ArrDataCMD[nBuffer] == 0xD) {
    ArrDataCMD[nBuffer] = 0;
    cmd_AStop();
    cmd_Start();
    cmd_CMove();
    cmd_PMove();
    cmd_LMove();
    cmd_SHome();
    cmd_GHome();
    cmd_ERead();
    ModeDisplay();
    digitalWrite(Port_LED1, SrlBeat);
    SrlBeat = !SrlBeat;
  }
```


รูปที่ ข.38 เรียกใช้ฟังก์ชันในส่วนของ Void loop

```
Serial.print("Now > ");
Serial.print(nNowCounter);
Serial.print(" Tar > ");
Serial.print(nTargetCounter);
Serial.print(" Dir > ");
Serial.print(RunDirection);
Serial.print(" Spd > ");
speed_fade(abs(abs(nTargetCounter) - abs(nNowCounter)));
Serial.println();

if ((RunDirection == 10) && (nNowCounter <= nTargetCounter)) {
  RunDirection = 0;


  Serial.println(nNowCounter);
  Serial.println(nTargetCounter);

  RunDirection = 0;
  digitalWrite(Pin_BRK, LOW);
  counter = 0;
  dac.setVoltage(ccounter, false);
  //  directionCount = 0;


  Save_NVRAM();


  Serial.println("-Dir, and Over");
}
```

รูปที่ ข.39 เรียกใช้ฟังก์ชันในส่วนของ Void loop

```
if ((RunDirection == 01) && (nNowCounter >= nTargetCounter)) {
   RunDirection = 0;
   Serial.println(nNowCounter);
   Serial.println(nTargetCounter);    RunDirection = 0;
   digitalWrite(Pin_BRK, LOW);    counter = 0;
   dac.setVoltage(ccounter, false);
   // directionCount = 0;
  Save_NVRAM();
   Serial.println("+Dir, and Over");
  }
  if (loopCnt > 40000)
  { digitalWrite(Port_LED2, HeartBeat);
   if (RunDirection == 0)
     ModeDisplay();
   else
     nTargetDisplay();
   nRunCountDisplay();
   HeartBeat = !HeartBeat;
   loopCnt = 0;
   Save_NVRAM();
  }
  if (digitalRead(Port_Sw4) == LOW)  Go_Move();
  if (digitalRead(Port_Sw1) == LOW)  Back_Move();
  if (digitalRead(Port_Sw3) == LOW)  Sp_UP();
  if (digitalRead(Port_Sw2) == LOW)  Sp_DW();
  loopCnt++;
  //Save_NVRAM();
}
```

รูปที่ ข.40 เรียกใช้ฟังก์ชันในส่วนของ Void loop

ภาคผนวก ค

บทความทางวิชาการที่ได้รับการตีพิมพ์เผยแพร่

รายชื่อบทความที่ได้รับการตีพิมพ์เผยแพร่ในระหว่างการศึกษา

Thongyot Sripheng., Wichai Srisuruk., and Sorada khangkaarn., (2020). **DESIGN AND PROTOTYPING OF CABLE-DRIVEN ROBOT SIZE 20*60 M FOR FISHING FARM.** The 398th International Conference on Robotics, Aeronautics, Mechanics and Mechatronics (ICRAMM). 6th-7th November, 2020 at Phuket, Thailand. 14-19.

DESIGN AND PROTOTYPING OF CABLE-DRIVEN ROBOT SIZE 20*60 M FOR FISHING FARM

[1]SORADA KHAENGKARN, [2]WICHAI SRISURUK, [3]THONGYOTSRIPHENG

[1]School of Mechanical Engineering, Suranaree University of Technology, 111 University Avenue, Muang District, Nakhon Ratchasima, 30000, Thailand
[2]School of Computer Engineering, Suranaree University of Technology, 111 University Avenue, Muang District, Nakhon Ratchasima, 30000, Thailand
[3]Institute of Engineering, Suranaree University of Technology, 111 University Avenue, Muang District, Nakhon Ratchasima, 30000, Thailand
E-mail: [1]soradak@sut.ac.th,[2]wichai@sut.ac.th, [3]thongyotThailand@gmail.com

Abstract - This paper presents the design and control for a cable-driven robotic size 20*60 m.with four cables.The cable-driven robot isa type of parallel manipulatorin which flexible cables are used as actuators.The end of each cable is reeled around a rotor twisted by a motor, and the other end is connected to the end-effector.The cable-driven parallelrobot (CDPR) manipulatorproduces a planar motion, including two translational and one rotationaldegree of freedom. The end-effector of CDPR motion, controller design, andthe kinematic structure is analyzed, and the inverse kinematicsis formulated in the closed-form solution.The resultsshowed that theeach cable must have a positive tension and the torque of winch must be greater than the maximum tension of cable.

Keywords - Inverse Kinematics, Modbus RS485 Protocol, Cable-driven Robot, Processing Software.

I. INTRODUCTION

Cable-driven robots are a type of parallel manipulators in which flexible cables are used as actuators. The end of each cable is reeled around a rotor twisted by a motor, and the other end is connected to the end-effector.Cables are much lighter than rigid linkages of a serial or parallel robot, and very long cables can be used without making the mechanism massive.The workspace and controllability of cable robots can be enhanced by adding cables to the structure of the robot. Consequently, the workspace analysis and designare different from those that can be referredto in this research [1-3].Only a few works are related to planar [4-7], or spatial cable-driven robot [8-10], or so-calledtendon driven Stewart platforms [11]. A class of measuringsystems has been based on the structures of parallelmanipulators with cables [12,13].

The cable-driven robots can be classified as "fully constrained" and "under constrained" based on the extent to winch end-effector is constrained by the cable.Inter first-class, the post of the end-effector can be completely determined as a function of the cable lengths. In the second class of cable-driven manipulators, the end-effector's pose is not completely determined by the cables' lengths. End-effector,Determined by the gravitational phenomenon
In this paper, we present the design and prototyping of a cable-driven robot size 20*60 meter.A prototype has been built, and tests have experienced the feasibility of the cable system design and its operation for planar and spatial tasks.

II. DETAILS EXPERIMENTAL

The cable-driven robot size 20*60 meter has been conceived atFishing farmin SuranareeUniversity of Technology. It composed by mechanical structure and diagram controller as show in Fig. 1.
The overview design and prototype of cable-driven robot size 20*60 meter. as shows in Fig.2-4.

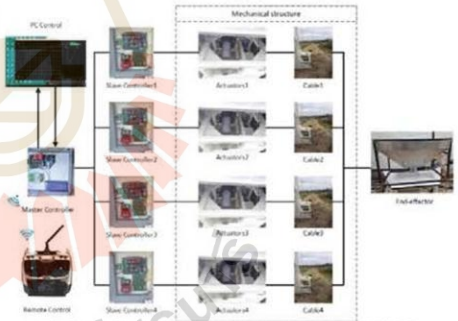


Fig.1. A scheme for cable-driven robot size 20*60 M

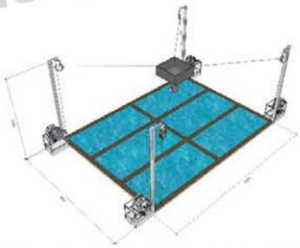


Fig.2.Overviewof cable-driven robot size 20*60 M

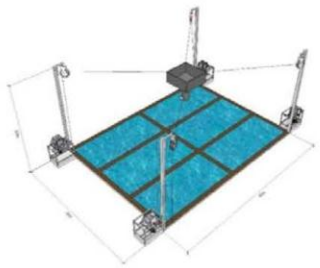


Fig.3.A built prototype of winch for Cable-Driven Robot Size 20*60 M

Fig.4.A built prototype of Cable-Driven Robot Size 20*60 M

2.1 Materials and Procedures

In this paper, we explain a cable-driven robot size 20*60 meter, whose specification are given as in Table 1. The prototype presented in this work is used to verify the algorithms such as inverse/forward kinematics, workspace analysis. Tension

Size of Frame	20 m x 60 m x 12 m
Speed	1 m/s
Payload	20 kg
DOF	3 DOF

Table1: Specification of cable-driven robot size 20*60 meter

2.1.1 System Configuration

Fig.5.System configuration of cable-driven robot size 20*60 M

Fig.4 Show the cable-driven robot size 20*60 mater is composed of a rigid frame, four winches that control the cable lengths, A master Controller calculate the length of the cable and send a cable length to the winch 1-4., an end-effector contains a fish feeder, as

2.1.2 Design of controller

Use an Arduino MEGA 2560 PRO as a controller.The controller contains, RTC Module, rs485 Communication Module, DIF switch for control slave number, Push button for manual test, and MAX7219 7 segment display Module for showcoordinates, as shown in Fig.6.

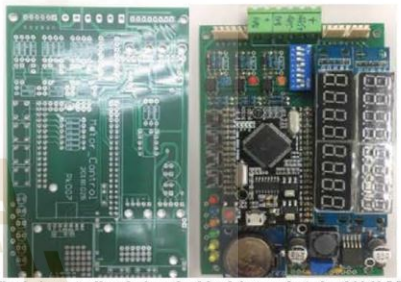


Fig.6. A controller designof cable-driven robot size 20*60 M

2.1.3 Design of Materials

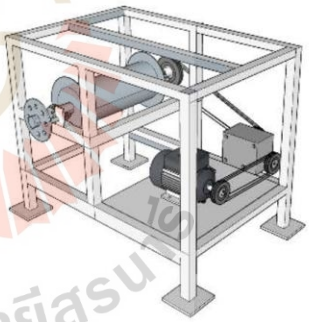


Fig.7.Assembly of the winch

Fing.7 show assembly of the winch, whose basic concept is same as the winch explained in [14]. A drum is supported by two bearings, One end is connected to a gearbox, the other is for measure the length of the cable with proximity sensor.Powered by motor brushless dc via gearbox with gear ratio 1:10.

2.1.4Choice of the Motors and Cables

A static analysis has been carried out in order to properly size actuators and cables of the proposed manipulator. In particular, two cables are connected to one point at the end-effector, as shown in the

simplified scheme of Fig.8, where m is half of the mass of the end-effector plus the payload.

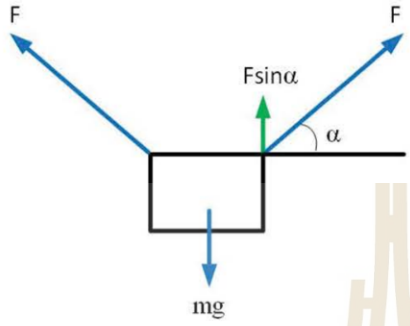


Fig.8.A 2D simplified scheme for statics evaluation.

For these conditions, each cable is loaded by a tension Fwhose direction plane. Thus, one can write respect to the horizontal plane. Thus, one can write

$$2Fsin\alpha = mg + ma \tag{1}$$

A maximum required force F_{req}con be calculated as in (2) to satisfy the specification in Table1.

$$F_{req} = \frac{m(g+a_{max})}{2sin\alpha_{min}} \tag{2}$$

The required torque for the actuator can be calculate as in (3), considering the drum radius and the gear reduction.

$$\tau_{req} = \frac{F_{req}\, r}{G} \tag{3}$$

The maximum acceleration a_{max} and the maximum payload are assumed to be 1 m/s^2 and 20 kg, respectively. The drum radius and gear reduction ratio are selected to r = 0.15 m and G = 10, respectively.

In is worth noting that if α becomes 0 the actuation torque becomes infinite. In fact, the configuration where tension of horizontal is singular, in which all the tension of the cables are orthogonal with respect to the force give by gravity (mg). Assuming α_{min} = 15 degree, we can get F_{req} = 425 N and τ_{req} = 21 Nm.

Thus, for the above-mentioned considerations each cable should yield a force higher than about 425 N and each actuator has a nominal torque of about 21 Nm.

2.2 Kinematics

2.2.1 Inverse Kinematics

A kinematic scheme of the cable-driven robot is show in Fig.9. two reference frames have been considered, namely Oxy is the fixed reference frame and Ox'y' is the moving reference frame. Points A_i,(for i = 1,2,3,4) lie on upper face, which has a square shape with dimension L, as shown in Figs. 3

According to the proposed scheme, the four cable lengths have been indicated with l_i(i = 1,2,3,4).

Cables are connected to the end-effector through two attachment points A and B, whose coordinates, with respect to fixed frame are given by (X_A, Y_A) and (X_B, Y_B). Point G is located on the center of mass of the end-effector.

The end-effector pose of the planar cable driven robot is given by the position of point G and θ angle, which is the angle between x and x'.

The Inverse Kinematics Problem of the planar cable driven robot can be formulated as finding the cable lengths l_i as function of the end-effector pose.

Suitable constrains are due to the position of point G, whose coordinates X_G and Y_G must to be within the limits h<X_G<L-h and 0<=Y_G<=L; and maximum value for θ for a given position of point G, [15]. The limitation for θ angle is related to cables. Which can only pull the end-effector but cannot push it.

The Inverse Kinematics Problem of the planar cable driven robot can be formulated as

$$\begin{aligned} l_1 &= \sqrt{x_A^2+y_A^2} \\ l_2 &= \sqrt{x_A^2 + (L-y_A)^2} \\ l_3 &= \sqrt{(L-x_B)^2 + y_B^2} \\ l_4 &= \sqrt{(L-x_B)^2 + (L-y_B)^2} \end{aligned} \tag{4}$$

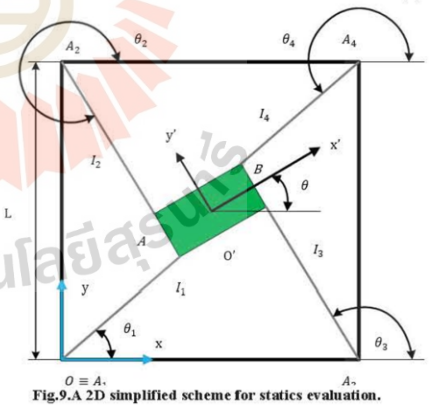


Fig.9.A 2D simplified scheme for statics evaluation.

In which

$$\begin{aligned} x_A &= x_G + hcos(\theta+\pi) \\ y_A &= y_G + hsin(\theta+\pi) \\ x_B &= x_G + hcos(\theta) \\ y_A &= y_G + hsin(\theta) \end{aligned} \tag{5}$$

The angular configuration $\theta_i$of the cables shown in Fing.9 can be evaluated as

$$\theta_1 = arctan\frac{y_A}{x_A}$$
$$\theta_2 = 2\pi - arctan\frac{L - y_A}{x_A}$$
$$\theta_3 = \pi - arctan\frac{y_B}{L - x_B}$$
$$\theta_4 = \pi + arctan\frac{L - y_B}{L - x_B} \quad (6)$$

2.2.2 Forward Kinematics

Finding the Cartesian position of the end-effector when joint variable are give is called forward kinematics. The problem of forward kinematics of Cable-Driven robot is one of highly complicated issues and cannot be solved in a closed form It is also an area of consistent research for parallel robot in general. In fact, for the general case with 6 degrees of freedom up to 40 solutions may exist for the forward kinematic problem [15]. Husty proposed a method using a univariate polynomial of degree 40 finding all these solutions [16]. This would be very impractical to implement. In this research, the cables tension forces is used as an extra sensory data in the solution of forward kinematics. Since only the inverse kinematics is needed for a position control, the forward one is not preformed in this wark.

2.3 Communications

The RS-485 is a protocol used for communication. this research becauseRS-485 supports inexpensive local networks and multidrop communications links, And can transmit data over long distances.

III. RESULTS AND DISCUSSION

A prototype of cable-driven robot size20*60M
It has the following Composition.

3.1 Mechanical

Mechanical comprising, Brushless dc motor driving power. Transmit power for gearblock. To increase driving torque Drum, The gear block tranmit power to the drum for make the drum actuator cable, A proximity sensor is attached for counting drum rounds.

Fing.10 show speeds and volts(V) testbackward, forward of motor brushless dc 750W 48V for cable-driven robot size 20*60 M.

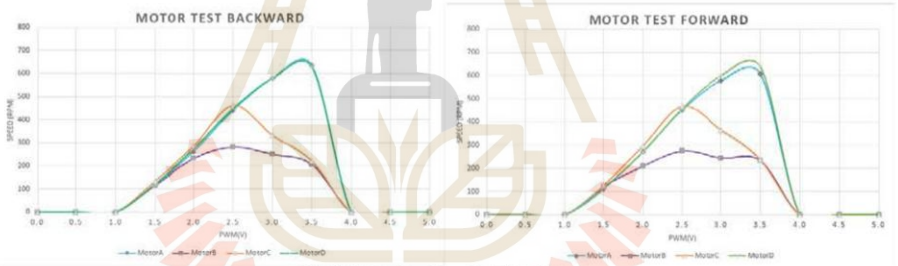


Fig.10. Motor brushless dc 750W 48V test

3.2 Controller

Controller comprising master controller and slave controller., Mastercontroller acts as l_1-l_4 receiver and calculators then sends l_1-l_4 to slave controller 1-4, Slave controller acts as winch1-winch4 control as master controller sends it.

Fing.11-12 show communications rs485 modbus protocol test of cable-driven robot size 20*60 M., Show master send data to slave and master read data from slave.

Fig.11.Serial monitor of master controller and Serial monitor of ETT(Model usb to rs485)

Fig.12.Communications of slave controller

3.3 Kinematic

In Cable-driven robot size 20*60 meter, inverse kinematics makes use of the kinematics equations to determine the joint parameters that provide a desired configuration for the robot end-effectors.Determining the movement of a robot so that its end-effectors move from an initial configuration to a desired configuration is known as motion planning.
Fing.11-12 show master controller calculate of length cable and send data to slave controller

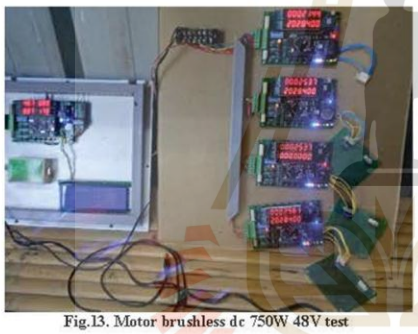


Fig.13. Motor brushless dc 750W 48V test

3.4 Prototyping

Test the movement of the cable-driven robot size 20*60 meter by setting the origin pointO(0,0). Test its with the remote control, enter the x,y coordinates. To move the cable-driven robot go to position A. When robot reach the A position, stop measuring the actual area relative to the usable distance.as shows in Fig.14.

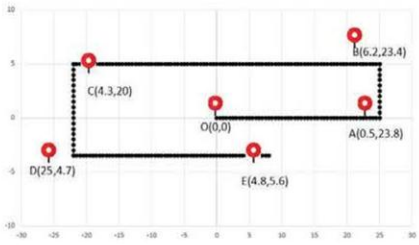


Fig.14.Test the cable-driven robot movement

IV. CONCLUSION

A suitable Kinematics analysis of cable-driven robot size 20*60 meter. architecture has given the possibility to conceive low cost easy operation design of a cable-driven robot size 20*60 meter. Basic performances have been simulated for design purposes and they have been experienced in successful test for validation purposes. But there are still high movement errors due to the winch design and the gps end-effector is not yet installed.The proposed cable-driven robot has been used in fishing farmat SuranareeUniversity of Technology both for under constrained and fully constrained application that have outlined the possibility to extend, and also can combine with fish feeder end-effecter in the future, the design concepts for low cost of cable-driven robot size 20*60 meter, That can be used on the real area.

ACKNOWLEDGMENTS

This work is supported by the Institute of Engineering(MechatronicsEngineering)at Suranaree University of Technology, funded by Institute of Research and Development (Science Program in Animal Technology and Innovation) at SuranareeUniversity of Technology.

REFERENCES

[1] Ebert-Uphoff I., Voglewede P. A., "On the Connections BetweenCable-Driven Robots, Parallel Manipulators and Grasping", IEEEInternational Conference on Robotics and Automation ICRA'04,New Orleans, 2004, pp. 4521-4526.

[2] Verhoeven R., Hiller M., Tadokoro S., "Workspace, Stiffness,Singularities and Classification of Tendon-Driven Stewart-Platforms", International Symposium on Advances in RobotKinematics ARK '98, Strobl, Austria, 1998, pp. 105-114.

[3] Riechel A. T., Ebert-Uphoff I.," Force-Feasible Workspace Analysisfor Underconstraiend, Point-Mass Cable Robots",IEEE InternationalConference on Robotics and Automation ICRA'04, New Orleans,2004, pp. 4956-4962.

[4] [4] Fattah A., Agrawal, S.K., "Workspace and Design Analysis of Cable-Suspended Planar Parallel Robots", ASME Design EngineeringTechnical Conferences and Computers and Information inEngineering Conference, Montreal, paper MECH-34330, 2002.

[5] Oh S.-R., Agrawal S.K.," Cable-Suspended Planar Parallel Robotswith Redundant Cables: Controllers with Positive Cable Tensions",IEEE International Conference on Robotics & Automation ICRA'03,Taipei, 2003, pp. 3023-3028.

[6] Kossowski C., Notash L., "CAT4 (Cable Actuated Truss—4 Degreesof Freedom): A Novel 4 DOF Cable Actuated ParallelManipulator",Journal of Robotic Systems, Vol.19, n.12, , 2002, pp. 605-615.

[7] Williams R.L II, Gallina P., Vadia J., "Planar Translational Cable-Direct-Driven Robots", Journal of Robotic Systems, Vol. 20, n.3,2003, pp. 107-120.

[8] Lytle A.M., Saidi K.S., Bostelman R.V., Stone W.C., Scott N.A.,"Adapting a Teleoperated Device for Autonomous Control UsingThree-Dimensional Positioning Sensors: Experiences with the NISTRoboCrane", Automation in Construction, Vol. 13, n.1, 2004, pp.101-118.

[9] Kamamura, S., Kino, H., Won, C., "High-speed Manipulation byUsing Parallel Wire-drive Robots", Robotica, Vol. 18, n.1, 2000, pp.13–21.

[10] Morizono, T., Kurahashi, K., Kamamura, S., "Analysis and Controlof a Force Display System Driven by Parallel Wire Mechanism",Robotica, Vol. 16, 1998, pp. 551–563.

[11] Ferraresi C., Paoloni M., Pastorelli S., Pescarmona F., "Studio di unMaster per teleoperazione a sei gradi di libertàazionato a cordini", 1aConferenza Nazionale ed ExhibitionsuSistemiAutonomiIntelligentie RoboticaAvanzata ENEA, Frascati, 2002.

[12] Jeong, J. W., Kim, S. H. and Kwak, Y. K., "Kinematics and Workspace Analysis of a Parallel Wire Mechanism for Measuring aRobot Pose", Mechanism and Machine Theory,Vol. 34, n.6 1999. ,pp.825–841,

[13] Ottaviano E., Ceccarelli M., Toti M., Avila Carrasco C., "CaTraSys(Cassino Traking System): A Wire System for ExperimentalEvaluation of Robot Workspace", Journal of Robotics andMechatronics, Vol. 14, n.1, 2002, pp.78-87.

[14] Pott, A., C. Meyer, and A. Verl. Large-scaleassembly of solar power plants with parallel cablerobots. in Robotics (ISR), 2010 41st InternationalSymposium on and 2010 6th German Conferenceon Robotics (ROBOTIK). 2010. VDE.

[15] Dietmaier, P.: The stewart-gough platform ofgeneral geometry can have 40 real postures. In:Advances in Robot Kinematics, pp. 7–16. Kluwer Academic Publishers, Salzburg and Austria (1998)

[16] Husty, M.L.: An algorithm for solving the directkinematic of stewart-gough-type platforms. In: Mechanism and Machine Theory, vol. 31, pp. 365–380 (1996)

★★★

ประวัติผู้เขียน

นายทองยศ ศรีเพ็ง เกิดเมื่อวันที่ 11 ธันวาคม 2538 ที่อำเภอเมือง จังหวัดนครพนม เริ่มการศึกษาระดับอนุบาลที่โรงเรียนบ้านกล้วย ระดับประถมศึกษาปีที่ 1-6 ที่โรงเรียนบ้านกล้วย และมัธยมศึกษาปีที่ 1-3 ที่โรงเรียนคำเตยอุปถัมภ์ มัธยมศึกษาปีที่ 4-6 ที่โรงเรียนนครพนมวิทยาคม ที่อำเภอเมือง จังหวัดนครราชสีมา สำเร็จการศึกษาวิศวกรรมศาสตร์บัณฑิต (วิศวกรรมคอมพิวเตอร์) สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี จังหวัดนครราชสีมา เมื่อ พ.ศ. 2560 และได้ศึกษาต่อระดับวิศวกรรมศาสตร์มหาบัณฑิต สาขาวิชาวิศวกรรมแมคคาทรอนิกส์ ขณะศึกษาได้เป็นผู้สอนรายวิชาปฏิบัติกา สาขาวิชาวิศวกรรมการผลิต สาขาวิชาวิศวกรรมแมคคาทรอนิกส์ และสาขาวิศวกรรมคอมพิวเตอร์ จำนวน 7 รายวิชาได้แก่

(1) ปฏิบัติการระบบควบคุมและอัตโนมัติ
(2) ปฏิบัติการวิศวกรรมแมคคาทรอนิกส์ 2
(3) ปฏิบัติการอิเล็กทรอนิกส์สำหรับวิศวกรรมคอมพิวเตอร์
(4) ปฏิบัติการระบบดิจิทัล
(5) ปฏิบัติการไมโครโพรเซสเซอร์
(6) ปฏิบัติการระบบฝังตัว (EMBEDDED SYSTEMS)
(7) ปฏิบัติการอินเทอร์เน็ตทุกสรรพสิ่ง (Internet of Things)

ระหว่างการศึกษาระดับปริญญาโทได้นำเสนอผลงานทางวิชาการ เรื่อง DESIGN AND PROTOTYPING OF CABLE-DRIVEN ROBOT SIZE 20x60 M FOR FISHING FARM ในการประชุมนานาชาติ 398th International Conference on Robotics, Aeronautics, Mechanics and Mechatronics (ICRAMM) Symposium 2020 แบบ Virtual presentation ณ จังหวัดภูเก็ต ประเทศไทย