# KINEMATICS SIMULATION AND EXPERIMENT FOR OPTIMUM DESIGN OF A NEW PROTOTYPE PARALLEL ROBOT

**MR. SURIN SUBSON**

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ENGINEERING IN MECHATRINICS ENGINEERING (INTERNATIONAL PROGRAM)**
**FACULTY OF TECHNICAL EDUCATION,**
**ACADEMIC YEAR 2022**
**RAJAMANGALA UNIVERSITY OF TECHNOLOGY THANYABURI,**

# KINEMATICS SIMULATION AND EXPERIMENT FOR OPTIMUM DESIGN OF A NEW PROTOTYPE PARALLEL ROBOT

MR. SURIN SUBSON

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF MASTER OF ENGINEERING IN
MECHATRINICS ENGINEERING (INTERNATIONAL PROGRAM)
FACULTY OF TECHNICAL EDUCATION,
ACADEMIC YEAR 2022
RAJAMANGALA UNIVERSITY OF TECHNOLOGY THANYABURI,

| | |
|---|---|
| **Thesis Title** | Kinematics Simulation and Experiment for Optimum Design of a Parallel Robot |
| **Name-Surname** | Mr. Surin Subson |
| **Program** | Mechatronics Engineering |
| **Thesis Advisor** | Associate Professor Dechrit Maneetham, D.Eng., Ph.D. |
| **Academic Year** | 2022 |

---

**THESIS COMMITTEE**

..... Chairman
(Professor Worawat Sa-ngiamvibool, Ph.D.)

..... Committee
(Assistant Professor Evi Triandini, Ph.D.)

..... Committee
(Assistant Professor Petrus Sutyasadi, D.Eng.)

..... Committee
(Associate Professor Ren Jean Liou, Ph.D.)

..... Advisor
(Associate Professor Dechrit Maneetham, D.Eng., Ph.D.)

Approved by the Faculty of Technical Education, Rajamangala University of Technology Thanyaburi in Partial Fulfillment of the Requirements for the Master's Degree

.....Dean of Faculty of Technical Education

(Assistant Professor Arnon Niyomphol, M.S.Tech.Ed.)

Date 19 Month April Year 2023

หัวข้อวิทยานิพนธ์ การจำลองและทดลองทางจลศาสตร์เพื่อออกแบบที่เหมาะสมที่สุดของหุ่นยนต์คู่ขนานต้นแบบใหม่
ชื่อ-นามสกุล นายสุรินทร์ ซับซ้อน
สาขาวิชา วิศวกรรมเมคคาทรอนิกส์
อาจารย์ที่ปรึกษา รองศาสตราจารย์ ดร. เดชฤทธิ์ มณีธรรม วศ.ด., ปร.ด.
ปีการศึกษา 2565

# บทคัดย่อ

วิทยาการหุ่นยนต์และระบบอัตโนมัติมีบทบาทเพิ่มขึ้นในภาคอุตสาหกรรมการผลิตทั่วโลก ในช่วงทศวรรษที่ผ่านมา หุ่นยนต์อุตสาหกรรมเป็นหน่วยย่อยในระบบอัตโนมัติที่ได้รับความนิยมและถูกเลือกใช้งานเพิ่มขึ้นอย่างรวดเร็ว ในปัจจุบันกระบวนการผลิตโดยใช้ระบบอัตโนมัติมาช่วยแก้ปัญหาในกระบวนการผลิต เช่น แก้ปัญหาความล่าช้า แก้ปัญหาของเสีย เป็นต้น แต่การใช้ระบบอัตโนมัติและหุ่นยนต์อาจส่งผลให้เกิดความผิดพลาดและความเสียหายต่อกระบวนการผลิตหากไม่มีจัดการที่ดี ดังนั้น เพื่อให้เกิดผลดีต่อผู้ผลิตจึงต้องมีการวางแผนการใช้หุ่นยนต์และระบบอัตโนมัติเพื่อเพิ่มประสิทธิภาพด้วยการวางแผนผลิตภัณฑ์และความเข้าใจการออกแบบและการควบคุมหุ่นยนต์เพื่อให้หุ่นยนต์ทำงานได้อย่างแม่นยำเหมาะสมกับสภาพและขอบเขตการทำงาน ดังนั้นจำเป็นต้องมีศึกษาและทำเข้าใจการใช้แขนกลในกระบวนการผลิตเพื่อให้ทราบถึงประสิทธิภาพและข้อจำกัดของหุ่นยนต์นั้น จุดสำคัญที่สุดคือการควบคุมแขนกลให้เคลื่อนที่ไปในพิกัดตำแหน่งที่ถูกต้อง

โปรแกรมจำลองหุ่นยนต์และระบบอัตโนมัติสามารถจำลองการเคลื่อนไหวของกระบวนการผลิตให้อยู่ในรูปของแบบจำลองทางคอมพิวเตอร์ ระบบจะทำการทดสอบแนวคิดที่ต้องการทดสอบบนคอมพิวเตอร์เพื่อศึกษาพฤติกรรมของระบบทำให้สามารถจำลองการทำงานของหุ่นยนต์และระบบอัตโนมัติก่อนที่จะทดสอบในการทำงานจริงและนำไปสู่แนวทางการวิเคราะห์เพื่อปรับปรุงข้อจำกัดของระบบอัตโนมัติและหุ่นยนต์ให้มีประสิทธิภาพในด้านเวลาและลดค่าใช้จ่ายที่เกิดจากปัญหาที่ไม่คาดคิดจากข้อจำกัดของหุ่นยนต์ได้ด้วยการแสดงภาพเคลื่อนไหวของหุ่นยนต์ 3 มิติ (Process Simulate for Robotics) ทำให้

สามารถวิเคราะห์ปัญหาที่เกิดจากการทำงานของหุ่นยนต์ เช่น ความสามารถในการเข้าถึง การเคลื่อนไหวเพื่อหลีกเลี่ยงการตรวจจับการชน สามารถจำลองการทำงานของอุปกรณ์อื่น ๆ ที่มีระบบกลไกทำงานร่วมกับหุ่นยนต์ เช่น การเปิด-ปิด ระบบจับชิ้นงานและการเคลื่อนที่ของสายพานลำเลียง ทำให้เราสามารถวางแผนและวิเคราะห์แก้ไขปัญหาที่จะเกิดขึ้นได้ตั้งแต่ในช่วงการออกแบบก่อนการติดตั้งจริงได้

ในวิทยานิพนธ์ฉบับนี้ได้นำเสนอการออกแบบโปรแกรมสำหรับจำลองรูปแบบและเส้นทางการเคลื่อนที่ของหุ่นยนต์คู่ขนาน ด้วยโปรแกรมจำลอง MATLAB โดยใช้ทฤษฎีการวิเคราะห์ทางจลนศาสตร์แบบผกผันเพื่อทดสอบตำแหน่งพิกัดของแขนกลในวิถีการเคลื่อนที่ของแบบจำลองหุ่นยนต์และเปรียบเทียบผลการจำลองกับผลการทดสอบจริงกับต้นแบบหุ่นยนต์คู่ขนาน เพื่อวิเคราะห์ความแม่นยำในพื้นที่การทำงานของหุ่นยนต์ ต้นแบบหุ่นยนต์คู่ขนานได้รับการออกแบบให้ทำงานร่วมกับระบบการมองเห็นด้วยเครื่อง (Machine Vision) ด้วยการการประมวลผลภาพและการตรวจจับสีของวัตถุเป้าหมายเพื่อให้ได้ผลลัพธ์รูปแบบพื้นที่การเคลื่อนที่ของหุ่นยนต์และนำผลลัพธ์ที่ได้จากการจำลองและการทดลองจริงมาเปรียบเทียบเพื่อวิเคราะห์และทำความเข้าใจในบริบทของการออกแบบระบบอัตโนมัติและหุ่นยนต์ให้เหมาะสมกับความต้องการของอุตสาหกรรมนั้น ๆ จากผลการจำลองจลนพลศาสตร์ผกผันเทียบกับการทดลองจริง ความคลาดเคลื่อนมุมร่วมของแขนกลในสภาวะคงที่ในต้นแบบแขนกล มุม $\theta_1$ มุม $\theta_2$ และ มุม $\theta_3$ อยู่ในช่วงตั้งแต่ 0° ถึง 4° องศา และได้ผลการทดลองของการประมวลผลภาพและการตรวจจับสีของระบบการมองเห็นด้วยเครื่องโดยโปรแกรม NI Vision LabVIEW ซึ่งผลที่ได้แม่นยำกว่า 95%

**คำสำคัญ** หุ่นยนต์คู่ขนาน จลนศาสตร์ของหุ่นยนต์ MATLAB แบบจำลองวิถีการเคลื่อนที่ การประมวลผลภาพ

| | |
|---|---|
| Thesis Title | Kinematics Simulation and Experiment for Optimum Design of a Parallel Robot |
| Name – Surname | Mr. Surin Subson |
| Program | Mechatronic Engineering |
| Thesis Advisor | Associate Professor Dechrit Manetham. D Eng.,Ph.D. |
| Academic Year | 2022 |


**Abstract**

Robotics and automation have been playing an increasing role in the global manufacturing sector over the past decade. The industrial robot is a sub-unit in automation that has attracted a lot of attention and is rapidly increasing in use. Today's production line uses automation to solve various production problems such as delays, wastage, etc. If automation is not handled well, it can also cause mistakes and damage to production lines. Therefore, in order to create positive effect on the producers, effective planning for the use of robotics and automation is required. Product planning and understanding of robot design and use are required so that the robot can work precisely to suit the working conditions. It is necessary to study the use of robotic arms in production lines. The most important point is to control the robot arm to work properly, in order to get the correct location coordinates.

Robot and automation simulator program can simulate the movement of the production process in the form of a computer model. Program will test the desired concepts on the computer. In order to study the system behavior, simulation is allowed before the real work. It leads to automation analyze and improvement to be more effective for time and cost reduction which occur in on-site problem with 3D animation display.

Process simulation of Robotics can analyze the arising operation problem such as reach ability, collision detection, operation simulation of other devices with the mechanical system working the robot such as opening and closing of the Jig Fixture or Conveyor movement. The simulation helps operator plan and fix the error during designing time before actual installation.

In this thesis, a program design for simulating the trajectory tracking of a parallel robot is proposed. With the MATLAB simulation program, the theory of the inverse kinematic

analysis is used to test the coordinate position of the robot arm and the trajectory of the robot model. The simulation results are compared between actual test result and the prototype parallel robot. In order to analyze the precision in the working area of the robot, a prototype parallel robot was designed to work with the Machine vision system to obtain the results from comparison for analysis and understanding the context of automation design. Referring to the result of the inverse kinetics simulations compared with the real experiments, robots are designed to suit the need of each industry. The steady-state joint angle tolerance of the prototype robot arm $\theta 1$, $\theta 2$, and $\theta 3$ ranged from $0^\circ$ to $4^\circ$ degrees were examined. The experimental result of image processing and color detection were obtained. According to the machine vision system with the NI Vision LabVIEW program, the result showed more than 95% accuracy.

Keywords: Parallel robot, Kinematic of Robotic, MATLAB Simulink, Trajectory Modeling, Image Processing

**Acknowledgements**

For this thesis, first of all, I would like to express my sincere gratitude to my thesis advisor Associate Professor Dr. Dechrit Manetham for the valuable of guidance and encouragement which helped me in all the time of my research.

Secondly, I would like to thank to the thesis committees, Dr. Petrus Sutyasadi, Dr. Evi Triandini and Dr. Ren Jean Liou for their valuable comments and helpful suggestions.

Thirdly, I would like to thank to all of the lecturers, Dr. Tenzin Rabgyal and Dr. Worawat Sa-ngiamvibool for their valuable lectures and experiences while I was studying.

Fourthly, I would like to thank to Dr. Myo Min Aung for helpfulness in coordination for documentations.

Finally, I would like to thank to my mother for all her love and encouragement.

Surin Subson

## Table of Contents

Page

**Table of Contents (Continued)**

## List of Table

Page

## List of Figures

Page

**List of Figures (Continued)**

Page

**List of Figures (Continued)**

Page

**List of Figures (Continued)**

Page

# CHAPTER1

# INTRODUCTION

In the field of research, theoretical information on robot selection and automated systems. Theoretical research on basic kinematics of robots. Learn about the capabilities of using robots in a variety of workflows suitable for manufacturing. There are five research methods: one is to study the theory of Homogeneous coordination transformers, the other is to study the theory of Mathematical description of the target, the third is to study the theory of Relative transformation in workspace, and the fourth is to study the theory of Transformations along kinematic chains. Fifth, study the theory of Destination kinematics of robot.

## 1.1 Study Background

In the past the functionality of the robot continues to evolve as innovation progresses. At the same time, the global manufacturing sector is increasingly adopting robots in their work processes. And the demand for robots is likely to increase exponentially. and is expected to expand above the growth rate of global demand. [1 - 3]

Robots have played a huge role in the transformation of production systems. It also helps to improve the smooth production of goods and services in factories, shops or establishments. precise Including reducing the process and increasing the speed of the production process throughout the supply chain. Reduce the cost of operators enhance consumer satisfaction and increase the productivity of the overall economy [1 - 3]

Robots and automation are similar in terms of automation machines. Robots can be called a part of automation because they have similar components and functions. But robots can operate from decision-making programs and can be programmed to perform multiple tasks. which automation cannot do. The elements that can be clearly demonstrated for the robot are the components of the robot control system, consists of three main interrelated components: a programmer or device designed to input commands from a controller or user, designed to input and process commands via a controller or component designed to receive user commands. In order to continue to control or operate the robot, the

manipulator (referred to simply as the "robot body") executes the instructions processed by the part that receives the user's command. [1 - 3]

Humans develop various forms of production. for a long time from the manufacture of parts by hand, fabrication of belts, until now most of the production is carried out with robots. The innovations that occur are not for human work. but to encourage people to work more conveniently and focus on working in an easier way
It can be said that today's robots are only a small first step in the work of humanity. But it is an important step in the manufacturing industry. Humans are entering a new industrial revolution. that will be fully automated soon. [1 - 3]

For the production of industrial plants today Automation and robotics have been introduced into some production lines to replace the overload of human operations, reduce waste, and increase accuracy in picking and inspection of workpieces. to reduce production time and increase productivity by producing quality superior to traditional production that uses only human labor [1 - 3], such as the inspection process for defects of the workpiece and sorting workpieces according to appearance, color, and size. Visually inspecting workpieces is still limited by speed. The precision and use of the human eye to perform such tasks can lead to distortion. Therefore, in order to achieve the perfect workpiece as specified by the conditions the use of automation or robots for workpiece flaw inspection and sorting workpieces according to appearance, color, and size, is able to meet the requirements for the development of industrial production [1 - 3]. Based on this problem, this study proposes a technique that enables computer vision to detect color and size differences of target objects. And use the new prototype parallel robot. Based on the data obtained from the robot simulation kinematics, the dimensions of the prototype robot are designed so that the robot can move within the desired work area. And a robotic arm control system was developed to pick up objects and place them at specific locations. The image processing system uses the NI LabVIEW program [4 -9] to identify the attributes of color and size differences of the target object, and uses the image processing method to identify and process instructions. Then send the control signal to the Arduino Uno microcontroller, which is the main control board that controls

the movement of the parallel robotic arm and the stepper motor. Use 3 sets of controllers to control the robotic arm to move it to the designated position and pick up the target object and place it at the desired position. The NI LabVIEW program adopts the data transmission format connected to the computer through the serial port, which can send and receive program running data and display the results through the computer to control analog and digital operations. And developed more software to more conveniently communicate with the microcontroller and control the operation [4 - 9]

For the simulation and analysis of the working area of parallel robots, the kinematic robotic theory is used to determine the size of the robot suitable for the desired working area. In this research, a kinematic simulation technique is presented. Using MATLAB program to simulate and analyze the workspace model of the alignment robot. and trajectories for dimensional analysis of the main components of the robot structure. to be used as information to create a parallel robot new prototype to get a robot of the right size for the experiment the purpose of selecting parallel robots for This research is due to the fact that parallel robots are robots used for pick and place tasks in industrial factories. This is a little robot. By designing a parallel robot mechanism and analyzing the workspace with the kinematics of robot theory [18, 19], the robot arm can be moved rapidly.

Computer simulations are an important step. In the analysis and design of control systems, especially industrial systems in analyzing and designing dynamic systems, we need to know their behavior. and various factors of the system that affect the operation, such as the time response and frequency response If there is a mathematical model of the system, then we can know the behavior of the system can be simulated by a computer. [18, 19],

In addition, the study of the dynamic system model can be divided into 2 main areas, namely the ease of modeling and the accuracy of the modeling system. If we want a very accurate model Modeling may be must be complex. The model will have uncomplicated structure

A linear system is a system in which the relationship between the signal and the output obeys the law of superposition, i.e. the response of multiple inputs. is equal to the sum of the responses from each input signal.

A time-independent system is a system in which the response from the input signal is delayed. is equal to the output delay. from being triggered by the original input without delay Linear and time-invariant systems It is a linear system that describes its behavior by a linear ordinary differential equation with constant coefficients.

A non-linear system is a system in which the superposition laws (superposition) is not applicable, i.e. the response from multiple input signals. The signal is not equal to the sum of the responses from each input signal. An example of a nonlinear system

For this experiment computer program used for calculations Including the program MATLAB/Simulink which has a simple and clear display tool Users can easily and flexibly interact with the program. MATLAB/Simulink [18, 19], It greatly reduces the burden of program development. Especially in learning numerical methods. and writing programs to perform computations (such as solving differential equations), allowing us to concentrate fully on the analysis and design of the control system. There is also a user interface that is easy to use and fast.

1.1.1 Benefits of Automation (Automation) [1 - 4]

1.1.1.1 Reduce operating costs, depending on the task, robots can replace 3-5 humans, in addition to saving labor costs. also save energy Public utilities in production, such as working in factories, workers have to turn on the air conditioner in case the temperature is too high inside the factory. therefore, causing additional expenses but if using robots in factories, there is no need to turn on the air conditioning because the robots can work at higher temperatures than humans. Therefore, the use of robots reduces operating costs and consumable materials.

1.1.1.2 Increased productivity, Robot can run at a constant speed unattended 24/7. That means have more productivity potential. Resulting in increased work productivity than the use of workers in production Because workers cannot work

24 hours a day, workers must take breaks. Unlike robots that can do it all the time

1.1.1.3 Better planning, Consistent production by robots helps shops reliably predict time and costs. Such predictability allows most projects to be more rigorous. And make the company's productivity meet the expected goals.

1.1.1.4 Consistency, production of parts and better quality, Automation generally runs the production process with less variance than workers. As a result, productivity from automated systems is efficient. There is more control over the quality of the product and more consistency than manual labor.

1.1.1.5 Improve labor safety, Automation takes workers out of dangerous jobs. The system uses machines to work instead of humans, such as handling hazardous materials. Working in extreme temperatures and moving heavy materials which if workers do it may cause danger and injury Using machines to help make factory employees safe. and reduce accidents in the workplace as well

**1.2 Statement of the Problem**

Parallel robot the motion of the robot is not linear, so it is difficult to control the robot. And it is a multi-joint robot, all the end joints of the robot are connected in a closed system. Make the solution endless. Therefore, it is important to be able to understand the robot's motion patterns and limitations when designing the robot's structure and control system. And must be able to design robots and control systems. In this study, a parallel robot design is presented using mathematical calculations and using MATLAB simulation program to show the boundaries of the robot's motion patterns. [18, 19], used to compare with the data reflecting the movement of the robot from the actual experiment. Collect the data of the actual experiment of the parallel robot prototype, and display the motion results of the robot through the C++ Arduino IDE program, It will be helpful to show the extent of the robot's real workspace and compare it with the simulation results of the robot's motion. The limitation of Parallel Robot is that the working area is relatively limited, obstacles cannot be bypassed, the work is relatively difficult, and multiple solutions can be derived. This study presents a synthesis of the optimal structural space

of a Parallel robot when determining the required workspace by finding the optimal relationship between the various structural spaces. of the Parallel robot which defines the working area to be in basic geometry to determine the size and installation of the parts of the Parallel robot that are suitable for a given work area.

The robot control system is usually responsible for Machin Vision detection. Motor driving and movement require complex algorithms the design of these complex systems still requires a lot of experience in remote teaching. Knowledge, expertise, and understanding of its application and technology suitable for the industrial environment. Due to the imperfection of parts used in the manufacture of robots, mechanical arms. Errors generally occur during operations and can damage the production process. This is more serious when robotic robots are used for precision applications.

The parallel robot design is powered by three Stepper motors using Forward Kinematics and Inverse Kinematic equations as mathematical models. Design and assemble the parts and use Ni Vision Builder to separate the objects. and using a microcontroller control system to control the robot the movement of the robot depends on the efficiency of the motor and the material of the robot parts.

Experimental use of a three-legged robot from the design and assembly of a robot body was tested on the movement of a robot using three Steeper motors to move the arm's triangular plates. The mover moves to the specified position along the x, y, and z axes, both positive and negative.

This project focuses on the color separation of the workpieces using the camera system and the program. for use in real-time display and recording what is needed in the system is a balance system to get a still image There is no oscillation and it is a system that allows the robot to maintain its motion characteristics. This research will study the Parallel Robot balancing system to enable the robot to function more efficiently and be able to move as we want.

## 1.3 Purpose of the Study

1.3.1 Develop a MATLAB/GUI program for designing a simulation program for the orientation of the Parallel robotic arm according to the coordinates and paths of the robot orientation workspace model, and the simulation case results can be used to analyze the robotic arm structure model.

1.3.2 Development of Machine Vision System using Ni Vision LabVIEW program to detect objects with different colors and send a signal to the microcontroller control system as a signal to control the robot

1.3.3 Design a prototype parallel robot based on a robot kinematic simulation model and design a robot control system using a microcontroller for automation in conjunction with a machine vision system.

1.3.4 To compare the results of simulations and experiments in order to analyze the results for deficiencies in order to develop a prototype parallel robot that is suitable for use in the industrial sector.

## 1.4 Research Questions and Hypothesis

1.4.1 The simulation of the parallel robotic arm orientation according to the coordinate point and the robot orientation model path by MATLAB/GUI program is effective according to the results of robotic kinematic calculations and the simulation results can be used for analyzing and designing the prototype of the robotic arm.

1.4.2 Ni Vision LabVIEW is a specification-compliant program for detecting objects of different colors and is compatible with microcontroller control systems.

1.4.3 The prototype parallel robot has a performance model in the workspace that conforms to the pattern obtained from the program MATLAB kinematics of robotic simulation results with good compatibility control with the microcontroller control system.

## 1.5 Theoretical Perspective

This thesis uses the fundamental theories concerned with the DH Parameters for Simulation, and calculation, of the kinematics of the Robot and applies geometry to the study of the movement of multi-degree-of-freedom kinematic chains that form the

structure of robotic systems.

1.5.1 Forward Kinematics Solution,

1.5.2 Inverse Kinematics Solution,

**1.6 Delimitations and Limitations**

1.6.1 Using a type of robotic arm Parallel Arm (Parallelogram)

1.6.2 Using Arduino as a controller robotic arm

1.6.3 Use LabVIEW program as User Interface

1.6.4 Use MATLAB program for Robotic Kinematic simulation to find a workspace model of the robot.

**1.7 Significance of Study**

This it aims to develop programs and kits that are commonly used in industrial applications for parallel robots with low-cost and performance functions that are suitable for the needs of small and medium-sized industrial plants.

# CHAPTER2

# REVIEW OF THE LITERATURE

## 2.1 Industrial Robotic

An industrial robot is a robot that has a structure similar to the human body, with a waist, elbow, arm and wrist. The term mechanical arm refers to the arm of an industrial robot. [10], industrial robot design It is an application of engineering in many different fields, including mechanical engineering and industrial engineering. To design and build a robot with a mechanical structure connected to each other and choose materials to be strong and durable. electrical engineering to choose the type of motor and power supply to the motor and electronic engineering. To connect the hardware and software to the microcontroller or PLC and the robot to control the movement of the robot. As shown in Figure 2.1

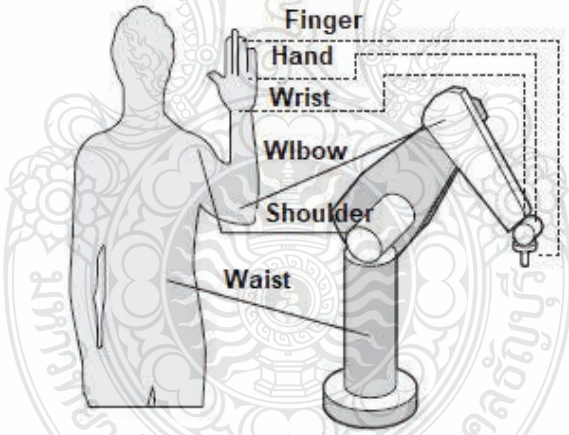



**Figure 2.1** Comparison picture of an industrial robot's arm and a human body

Types of industrial robots Classification by Arm Motion, Industrial robots are another type of automatic machine designed and built to replace people in various production processes. or used to help in the production process in a way that robots work with people. The robots that are created have many types depending on the nature of the work that needs to be applied. For industrial robots, it can be divided According to the

nature of work, there are 6 types as follows: Cartesian Robot, Cylindrical Robot, Polar Coordinate Robot, Scalar Robot, Articulate Robot, Parallel link Robot. There will be differences like movement and the ability to work differently. Including different applications, but all built on the same basic principles. [10],

2.1.1 Cartesian Coordinate Robot is a robot that has a working area in the manner, the cuboid has a joint movement of the X-axis, Y-axis, and Z-axis sliding (prismatic; P), also known as PPP robots, so it is easy to program. with precise resolution in high work, used in picking and placing workpieces Assemble the CNC machine workpiece and welding work, as shown in Figure 2.2

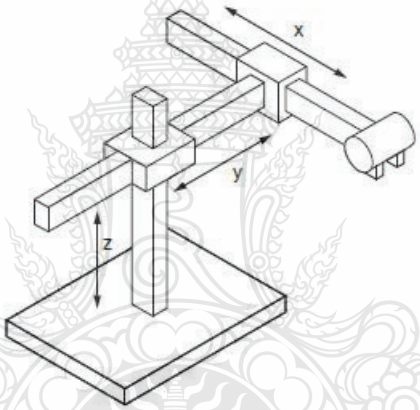



**Figure 2.2** Cartesian Robot Manipulator

2.1.1.1 Strength

1) Move in a straight line in all 3 dimensions.

2) The movement can be easily understood.

3) Simple components

4) Strong structure throughout the movement

2.1.1.2 Weakness

1) Need a lot of installation space

2) Area where the robot can work is smaller than the size of robot

3) The object cannot be reached from the southward direction.

4) The linear axis will Seal to prevent dust and liquid difficult.

2.1.1.3 Applications, because the structure is strong along the movement Therefore, it is suitable for moving heavy objects. or called work Pick-and-Place, for example, used to load workpieces into the machine (Machine loading), used to store workpieces (Stacking), can also be used in assembly work that does not require access in a rotating manner such as assembling electronic equipment and Test work

2.1.2 Spherical Robot or Polar Robot is a robot that has two joints, Rotary (Revolute; R) and 1 joint is Prismatic; P) with 2 axis of rotation and 1 axis of motion, known as RPR robot. It is commonly used in handling, lifting or moving things. Electric welding and gas welding as shown in Figure 2.3


**Figure 2.3** Spherical Robot Manipulator

2.1.2.1 Strength

1) More working volume due to the rotation of the 2nd (shoulder) axis.

2) Be able to bend down to grip the workpiece on the floor conveniently.

2.1.2.2 Weakness

1) There is a coordinate system and complex components.

2) Movement and control systems become more complex.

2.1.2.3 Applications, used in work with a little vertical movement

(Vertical), such as loading workpieces into and out of the press or may be used for spot welding.

2.1.3 Cylindrical Robot is a robot with 2 joints, sliding joints and 1 joint is a rotating type, known as the RPP robot, causing the working area to be cylindrical, commonly used in assembly work, spot welding, as shown in Figure 2.4

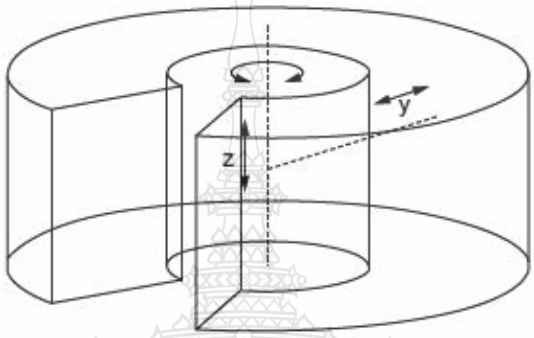



**Figure 2.4** Cylindrical Robot Manipulator

2.1.3.1 Strength

1) There are simple components.

2) The movement can be easily understood.

3) Able to access machines that are open-closed or enter into areas that are channels or holes easily (Loading), such as loading workpieces into CNC machines.

2.1.3.2 Weakness

1) Limited working space

2) The linear axis is difficult to seal to prevent dust and liquid.

2.1.3.3 Applications, It is generally used to pick up workpieces. (Pick-and-Place) or feed the workpiece into the machine. because it can easily move in and out of the area that is a small cavity

2.1.4 SCARA Robot is a robot that rotates 2 parallel axes and moves 1 axis, known as RRP robot, used to pick and place objects. assembly work and machine tools as shown in Figure 2.5

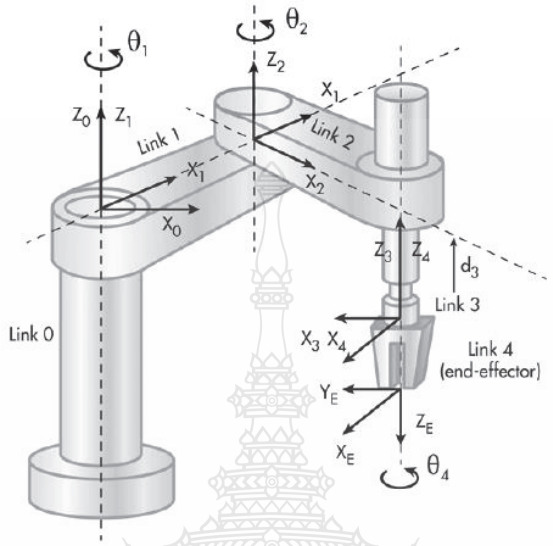



**Figure 2.5** SACARA Robot Manipulator

2.1.4.1 Strength

1) can move in a horizontal plane and go up and down quickly

2) High precision

2.1.4.2 Weakness

1) Limited working space

2) Can't rotate in various angles

3) Able to lift weights (Payload) not much

2.1.4.3 Applications, Because of its horizontal movement and fast up and down, it is suitable for electronic assembly work. Which requires speed and movement does not require much rotation. But will not be suitable for mechanical part assembly, which most of the assembly will rely on rotation in various angles. In addition, SCARA Robot is also suitable for inspection and packaging.

2.1.5 Articulated Robot (Joint Arm Robot) is a robot that consists of rotating

joints. The work of various joints is similar to the work of humans by the various joints, including the waist (Waist), shoulder (Shoulder), elbow (Elbow) and joints. Hand (Wrist), known as Robot 6R, can move up and down. and on their own side

It is more commonly used in industrial plants than other types because it is strong. and is highly flexible in work but the cost of production is high. It also requires a complex control system. As shown in Figure 2.6

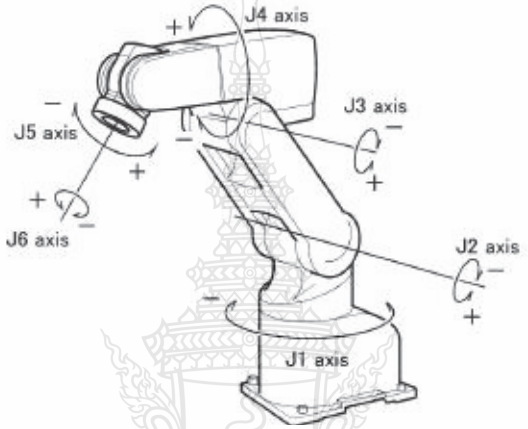



**Figure 2.6** Articulated Robot Manipulator

2.1.5.1 Strength

1) Since every axis will move in a manner of rotation makes it highly flexible in accessing various points
2) The joint area can be sealed to prevent dust, moisture or water easily.
3) There is a lot of working space.
4) Able to access the workpiece from both top and bottom
5) Suitable for electric motor use as a propulsion unit

2.1.5.2 Weakness

1) There is a complex coordinate system.
2) Movement and cleaning control system more difficult to understand
3) Difficult to control to move in a straight line (Linear)

4) The structure is unstable throughout the movement range. Because at the edge of the Work Envelope, the forearm will

5) There is a vibration, causing the accuracy to decrease.

2.1.5.3 Applications, This type of robot can be used widely because it can reach various positions well, such as spot welding, path welding, lifting, cutting, gluing, work with difficult movements such as painting and sealing.

2.1.6 Parallel Robot or Delta Robot, Parallel Robot It's a closed mechanical chain. It consists of a plate base and is sandwiched by an end effector plate on top, by means of a sliding rod-driven connecting rod of 6 through a universal joint, which the extension rod will only recognize compression or elongation without bending. as shown in Figure 2.7

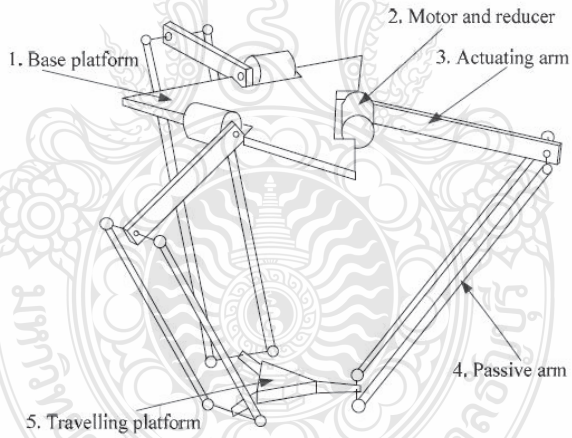



**Figure 2.7** Parallel Robot Manipulator

2.1.6.1 Strength

1) high strength because it is a closed structure

2) High precision because the tip of the arm reaches all power sources at the fast-moving base.

3) because most of the mass is at the base Therefore, the

forearm has a small mass

4) can exert a lot of force because all the power sources help each other to exert themselves.

2.1.6.2 Weakness

1) narrow working space

2) Complicated in controlling calculations

2.1.6.3 Applications, this increases the accuracy of the working position. And the structure is lightweight Triangle Robot It is commonly used in factory packaging. Medical and medicine that can work quite quickly

2.1.7 Kinematics and dynamics Kinematics and dynamics Details are as follows

2.1.7.1. Kinematics is the study of the movement of mechanical parts by the influence of force and mass on motion is not considered. Therefore, the kinetics are related to the region area. Amount of velocity and acceleration obtained by motion

2.1.7.2. Kinetics is the study of the action of forces causing mechanical parts moving which will also be caused by the influence of gravity [10, 12 – 17],

2.1.7.3 Dynamic is a combination of kinetics and kinetics, so Mechanical dynamics deals with both balanced and unbalanced forces acting on the mechanical part, taking into account the mass and acceleration of the mechanical part. As well as external forces [10, 12 – 17],

2.1.8 Degrees of Freedom, the degree of freedom of the system (DOF) is the amount of motion of the rigid body in the area of movement of the rigid Rigid Body. There are 3 types: [10, 12 – 17],

2.1.8.1 Prismatic or Translational Motion, A rigid body can move in 1 axis, 2 axes or 3 axes as shown in Figure 2.8 and is described by the name of the ship, namely

1) Throw (Heave) is a vertical linear motion (up/down).

2) Swaying (Sway) is a lateral path movement. (from side to side)

3) Surge is a horizontal linear motion. (forward/backward)

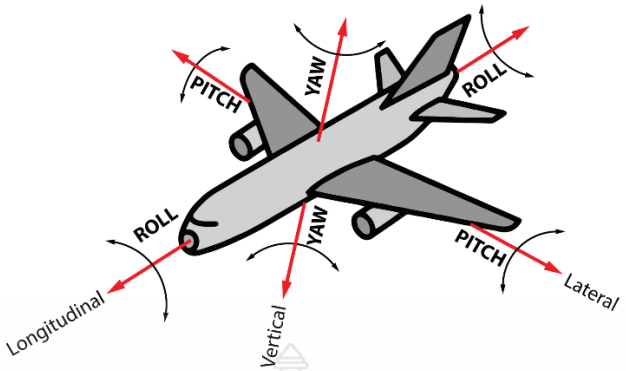



**Figure 2.8** Degrees of Freedom Chart

2.1.8.2 Rotational motion, A rigid body can move and rotate in 1, 2 or 3 axes as shown in Figure 2.2 (a) and is described by the name of the ship, namely

1) Roll is the movement around the horizontal axis.

2) Pitch is a rotation around the diagonal axis.

3) Yaw is the movement around the vertical axis.

As shown in Figure 2.9

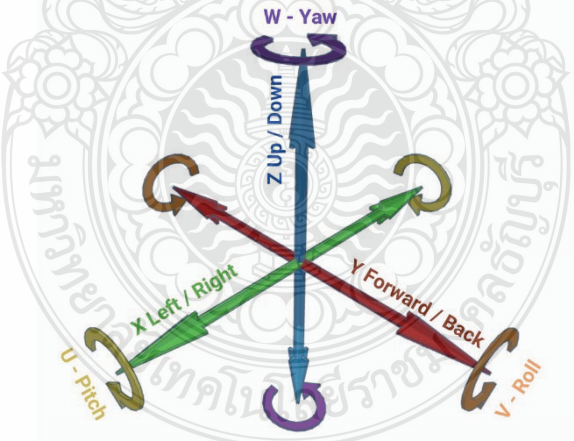
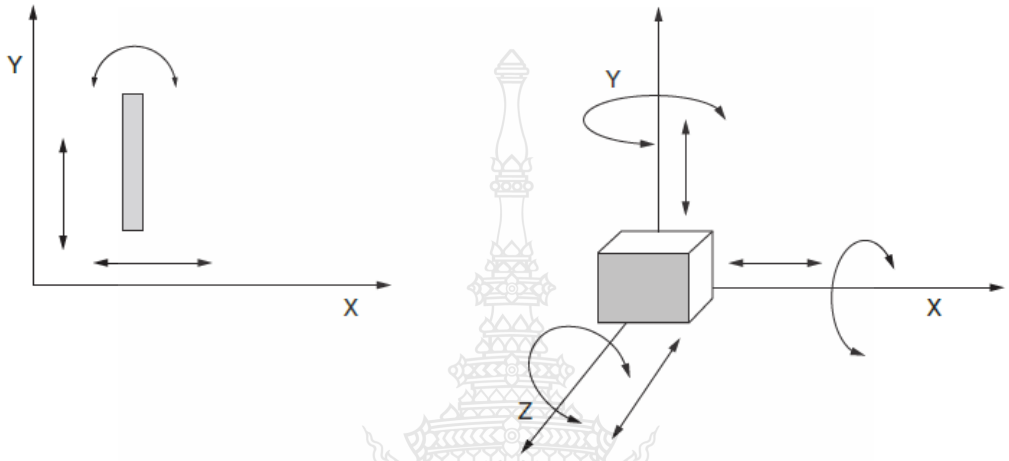



**Figure 2.9** Rotational 3 DOF motion

Figure 2.8 and Figure 2.9 Rigid body exhibits sliding motion (along the X, Y and Z axes). There are 3 DOF, and rotational motion (around X, Y, and Z axes) has 3 DOF. If the rigid body has a combined motion, it has 6 degrees of freedom (6 DOF).

Note: • A body object in a two-dimensional plane (2D) has 3 DOF, i.e. it moves along the x-axis and y-axis and rotates.

- A rigid body in the 3D plane has 6 DOF, i.e., moves along the x, y, and z axes and rotates around the x, y, and z axes, as shown in Figure 2.10




(a) 2D Rigid Body have 3 DOF (b) 3D Rigid Body have 6 DOF

**Figure 2.10** 3-DOF of objects in 2D and 3D planes

2.1.8.3 Both movements are combined (complex motion). It is a simultaneous rotational and sliding movement. The reference point on an object changes its position both linearly and angularly. And the paths on the waypoints are not parallel, and all centers of rotation are continually shifting.

2.1.9 Robot components, in one robot consists of many different devices and parts, each of which has different functions according to its characteristics. and purpose of use Selection of equipment and parts Therefore, knowledge, understanding and suitability are required. So that the robot can work efficiently, fast, durable and save energy. The robot is divided into four major components: Mechanical devices, Actuator, Electrical or electronic equipment and Control device (controller)

Mechanical devices, the structure of the robot can be divided into two main parts: the body and the arms. and wrist. Most of the torso and arms have 3 levels of

freedom, and in the wrist, there are 2-3 levels of freedom at the end of the wrist is an object that relates to the work that the robot has to do. For example, an object may be a workpiece that needs to be loaded into a machine, or it may be a tool that the robot needs to use in a certain production process. The robot's body and arms are used to provide accurate alignment of the object. and the wrist part the robot is used to orient the objects in order to position them. The body and arms of the robot must be able to move objects in the following 3 directions:

1) Vertical movement (Movement in Z axis)

2) Radial motion (in/out or movement in the Y axis)

3) Movement from left to right (moving in the X axis)

There are several ways to enable the robot to move in the above manner. depending with the types of joints used to build the robot's body and arm. which will be discussed in detail later. in order to cause proper alignment of objects We can define 3 degrees of freedom for the robot's wrist as the following example is one of the robot's wrist assembly patterns. to create a 3 degree of freedom

1) Roll This degree of freedom is achieved by using a T-joint to rotate an object. around the axis of the arm

2) Pitch related to the up-down rotation of the measuring instrument R type joint (Rotational Joint)

3) Yaw related to the left-right rotation of the object which can be done by using the R-type joint (Rotaximetal Jetset).

When considering the use of robots to assist in the production process, the nature of the work, the area, and the environment must be taken into account because the robot is a part that must be used in the work process all the time. The different job characteristics will indicate the size of the robot's structure. But the nature of the structure will have the same structure. The difference is only in the nature of the design. The key structure is designed to consist of the arm part or 'Link' and the joint part or 'Joint'.

2.1.9.1 Links, Link is the structure of the robotic arm. It is responsible for entering the working area. The length of the Link will indicate the performance of the robot. and the ability to enter the working area with Articulate Robot Industrial Robot has 2 links as follows. [10, 12-17]

1) Upper Link or Upper Link is the portion of the upper arm that enters the working area. And it is a part that connects to the Robot Hand wrist for installing the Robot Tool.
2) Lower Link or Lower Link is the part of the arm that is responsible for all the weight that occurs of the entire robot. It is the part that supports the weight of the upper arm and is connected to the base of the robot.

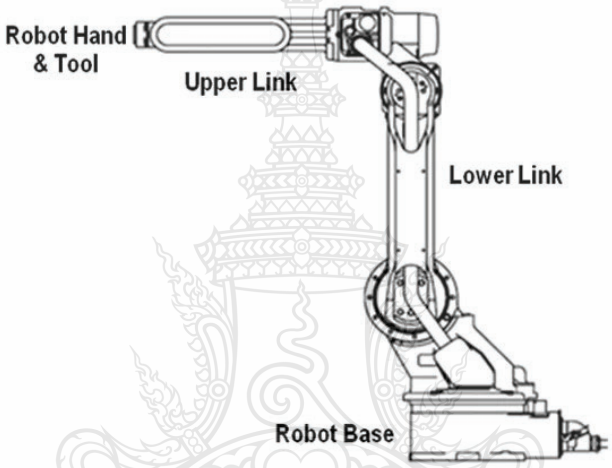



**Figure 2.11** Structure of Link or the robot's arm

**Table 2.1 Comparison table between link structure of robot and human arm**

| Robot | Human Arm |
|---|---|
| 1. Robot Base | 1. Waist |
| 2. Lower arm (Lower Link) | 2. The part of the forearm from the shoulder to the elbow. |
| **Robot** | **Human Arm** |
| 3. Upper Arm (Upper Link) | 3. The part of the forearm from the elbow to the wrist. |
| 4. Robot Hand | 4. The part from the wrist to the middle of the palm. |
| 5. Robot Tools | 5. Finger |

2.1.9.2 Joints, Joint is the structure of the robot, the joint that acts as a connection between the link of the robot and also serves to move the robot to be able to move to different positions. What we want, that is, when we program the robot to move, is to program the Joint or all joints of the robot. That means that the joint part is the part where the servo motor is installed. Normally, an articulate robot type industrial robot has 6 joints or sometimes people in the robot control industry. It is often referred to as a 6-axis robot [10, 12-17] as shown in Figure 2.12

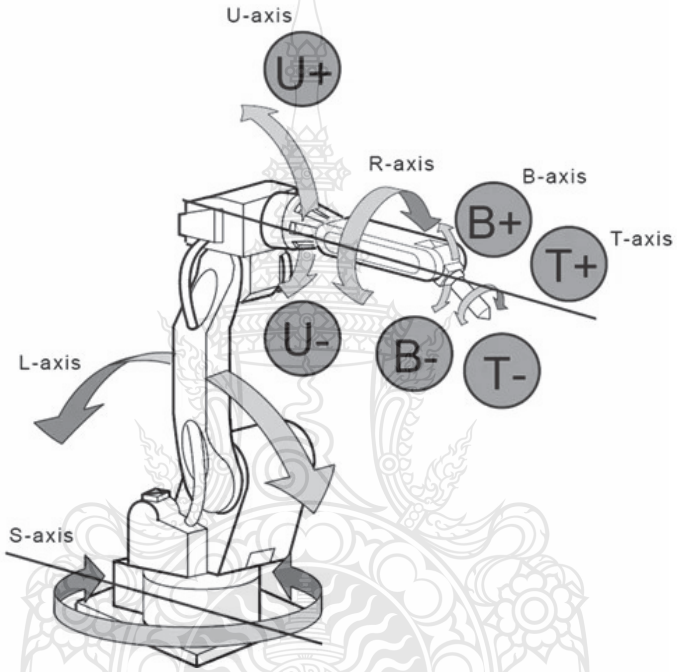



**Figure 2.12** Joint of the robot and its name

From Figure 2.12 shows the joints or joints of the robot, which are all 6 joints together, with names and functions of each joint and can be compared to human arms as shown in Table 2.2

**Table 2.2** Names and functions of each Joint and comparison with human arm

| Robots | | | Compared to |
|---|---|---|---|
| **Axis or Joint No.** | **Name** | **Description** | **Humans** |
| Axis 1 or Joint 1 | S | Rotation of the complete lumbar manipulator | waist |
| Axis 2 or Joint 2 | L | Forward and reverse movement the lower arm. | shoulder |
| Axis 3 or Joint 3 U | U | Vertical movement of the upper arm. | elbow |
| Axis 4 or Joint 4 R | R | Rotation of the complete wrist centre. | The part that rotates the upper arm to the wrist |
| Axis 5 or Joint 5 B | B | Bending of wrist around the wrist centre. | wrist part |
| Axis 6 or Joint 6 T | T | Rotation of mounting flange (turn disc) | wrist swivel |

1) joints classified by Mobility

(1) Active joint is the joint where the plant is being installed. able to control movement

(2) Passive joint is a joint that does not have an installed tree. inability to control movement the movement follows the movement of other joints that are linked to each other.

2) joints classified by movement patterns

(1) Revolute joint is a joint that can rotate (1 degree of freedom).

(2) Prismatic joint is a joint that moves in and out along a line (1 degree of freedom).

(3) Screw joint is a joint that rotates and moves in and out relative to each other (1 degree of

freedom)

(4) Spherical joint is a joint that can rotate around (2 degrees of freedom).

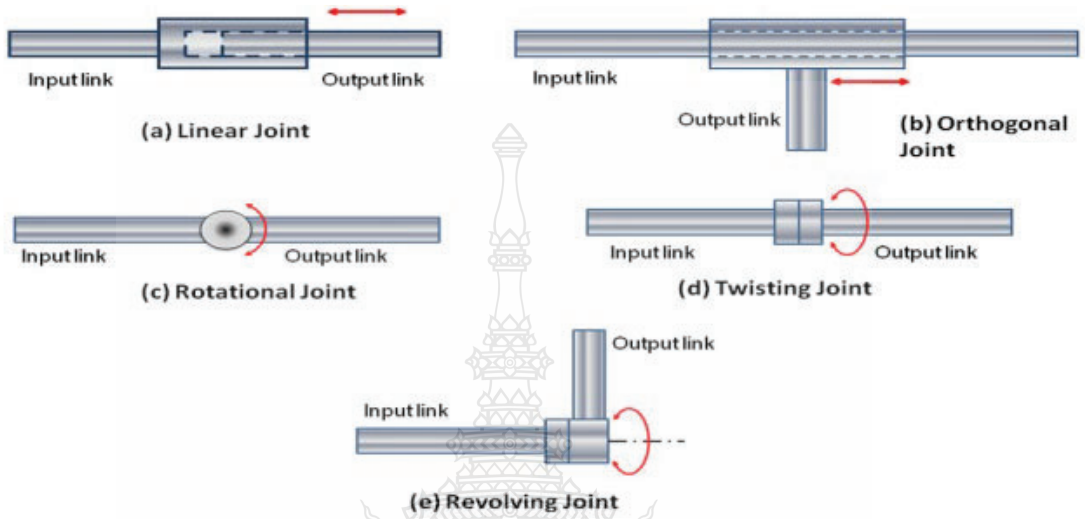



**Figure 2.13** Various joints of robot

2.1.9.3 base is the first fixed link.

2.1.9.4 wrist is a joint that does not cause movement in three dimensions but causes rotation It is often placed as the last joint on the forearm.

2.1.9.5 end-effector is a device that is installed at the forearm to use for work may be a handle Vacuum suction, drill, etc.

### *2.2* Parallel Robot

Parallel robot was developed by Raymond clavel, [19, - 22], studied and prototyped in 1980 as a parallel robot as shown in Figure 2.11 by Ramon clavel's mechanical arm developed. It is a robot with four degrees of freedom (Degree of freedoms), with three degrees of sliding. (Translation) and one degree of freedom of rotation (Rotation). This development was awarded by Ramon Clavel in 1990. The robot was then used in the packaging industry as the first due to properties of the parallel arm with precision and

high speed. In addition, plus the delta, the puppet is symmetrical between the three arms. As a result, the robot is convenient to use.

Parallel Robot or Delta Robot has its strengths in speed, accuracy, and robustness, with the sensor and vision system components built into the robot arm for extremely accurate handling and placement of moving objects. The Parallel Robot is therefore suitable for industries were goods flow along the conveyor belt. with a disorganized appearance or as we often call it, laid out directionless, formless, lightweight, bulky in conveyor belts. That comes with high speed, such characteristics are food, confectionery, medicine, cosmetics, consumer goods, electronics, auto parts, materials that are packaged in various packages, etc.

In general, a Parallel robot has a three-legged spider-like structure that is the same size. (end-effector) All arms are mounted on the circle circumference of the stationary base. And each arm is mounted at an angle of 120 degrees to each other,


**Figure 2.14** Sample of Parallel Robot

### 2.*3* Stepping Motor

Stepping Motor It is a type of electric motor that is used to control the rotation to set the position and direction by angle of the mechanism of the machine that requires high precision, such as control systems in robots, printers, conveyor belts in factories, etc., which rely on It works by using a square wave electrical signal (Pulse) to drive the motor so that the axis of rotation is non-continuous, but will drive step by step by being

able to rotate around the axis 360 degrees, causing the mechanism to move the axis at a low speed. and maintain torque instantly without damaging the motor. Depending on the motor structure of each person as well [21, 22],

Stepping motors Terminology PHASE is the part of the coil. Between the end of the cable and the CENTER TAP or both coils if there is no CENTER TAP, Phase Angle refers to the number of degrees caused by rotation in each step, FULL STEP MODE means FULL STEP rotation, i.e. rotation of each step will get angle equal to Phase Angle, HALF STEP MODE is a HALF STEP rotation. Each step rotation will have an angle of Half of Phase Angle

2.3.1 Stepper Motor Components

Stator is the part attached to the body of the motor. It is a magnetic pole with a small tooth tip wrapped with a coil to induce a change in the magnetic field.

Rotor is a pack of permanent magnets. It looks like a gear with teeth to be sucked in to match the teeth of the stator. This rotor is attached to the spindle to be used as needed. With this stepper motor has a large number of teeth. making it able to move in small steps

The working principle is the same as the DC motor. Is to supply electricity into the coil to change into a magnetic field. Then the magnet at the rotor core will be attracted to the coil. Equal to 1 Step moving motor. Example as shown in the picture below. For example, a motor with 8 Stator Coils and 6 Rotor Poles.

As already explained that the components are similar to ordinary DC motors. as shown Figure 2.15

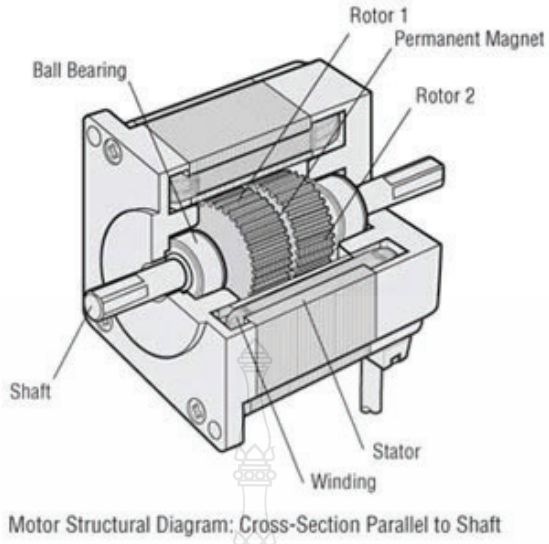



**Figure 2.15** Stepper Motor Structural Diagram

Stator is the part attached to the body of the motor. It is a magnetic pole with a small tooth tip wrapped with a coil to induce a change in the magnetic field.

Rotor is a pack of permanent magnets. It looks like a gear with teeth to be sucked in to match the teeth of the stator. This rotor is attached to the spindle to be used as needed. With this stepper motor has a large number of teeth. making it able to move in small steps

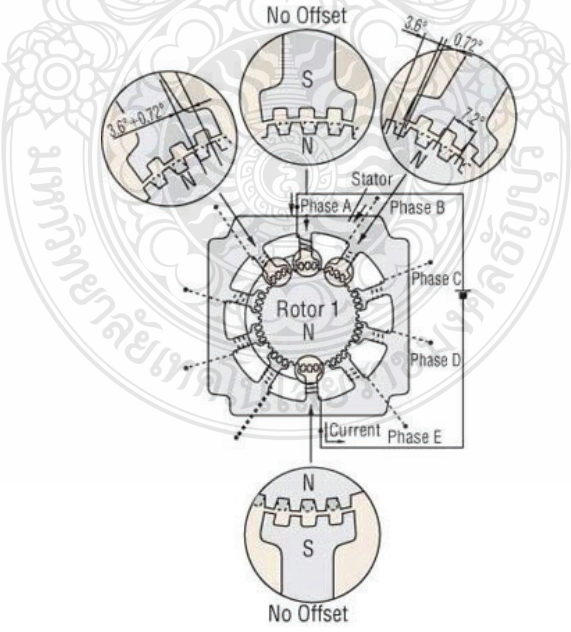



**Figure 2.16** Stepper Motor Structural Section

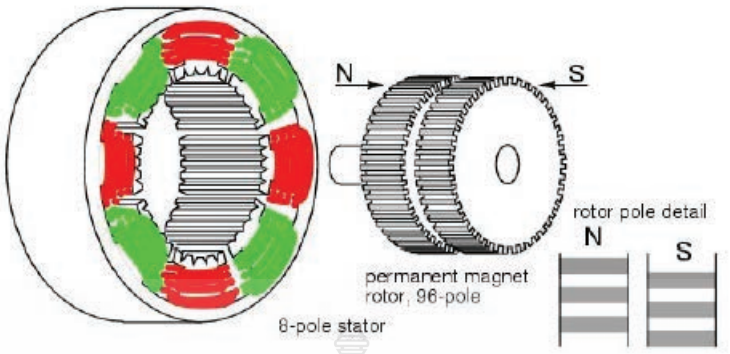



**Figure 2.17** Permanent magnet pole placement in Stepper Motor

2.3.2 Type of stepper motor

2.3.2.1 Divided by structure

1) Variable reluctance stepper motors the structure of the motor consists of a steel rotor that will be attracted to the stator when the stator receives an electric magnetic field. (The temporary magnet generated by the coil attracts the iron core itself.)

2) Permanent magnet stepper motors the motor structure has a permanent magnet rotor. which will attract or push against the stator according to the pulse supplied

3) Hybrid synchronous stepper motor the motor structure is a combination of Variable and Permanent.

2.3.2.2 Divided by type of stator

1) Bipolar Stepper Motor This type of motor has no Common Wire coil attached to the stator, making the design of the drive circuit quite complicated. MCU must be used to help and H-Bridge circuit can be used to drive.

2) Unipolar Stepper Motor (also known as 4-Phase) The stator winding of this type of motor has a Common Wire for use as Common Ground or Common Power. This type of motor can be easily built to drive circuits. easy to use

which must supply electricity (or connect to GND) to Common all the time and supply power to various coils to complete the cycle This unipolar stepper motor is divided into several types: 5-Wire type with 5 wires, Common of both sets of coils connected together as a single line, 6-Wire type with 6 wires, Common of each set separately (As shown in example Figure 2.17 below) and 8-Wire is that each coil uses its own separate Common.

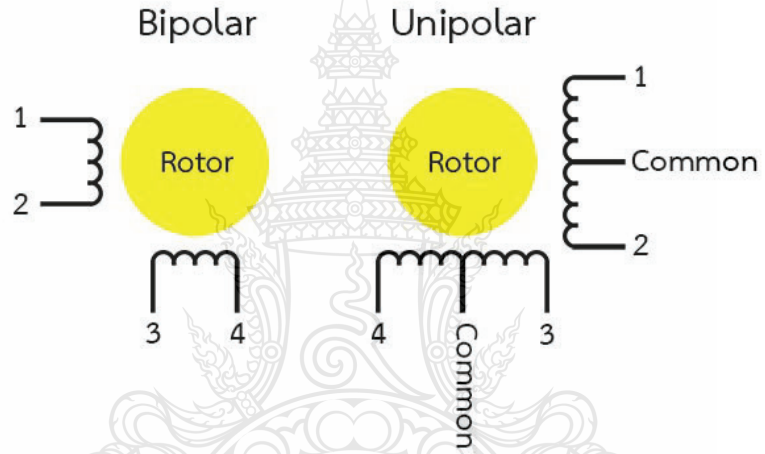



**Figure 2.18** Divided Stepper Motor by type of stator

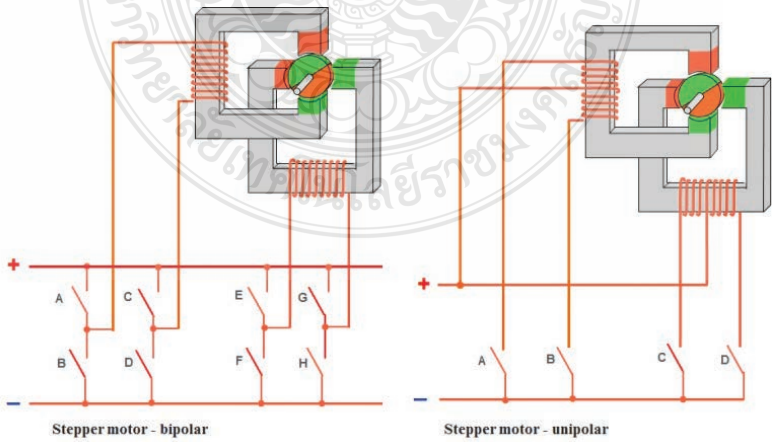



**Figure 2.19** Stepper Motor type of stator Bipolar and Unipolar

2.3.3 Stepping motor selection

various industries That require high precision in terms of the position of the workpiece movement often have a stepping motor in the production process. which if do not know how to choose the correct purchase the output may not be as efficient as it should be. So how to choose Stepping motors to suit the job description is as follows.

2.3.3.1 The number of phases of the motor: consider the work that we will use to see what type you want, such as 2 Phase, or 5 Phase.

2.3.3.2 Size of the motor (Size): The size of the motor that we will use is based on the size of the area that we will install.

2.3.3.3 Current that the motor uses: You should know the current value that the motor that we will choose to use can be used with how much current.

2.3.3.4 Motor resolution (degrees): It is a very important part when choosing a stepping motor. This degree indicates the resolution and accuracy of the work position.

2.3.3.5 Motor Torque: Torque is as important as the angle of movement of the motor. Torque value or torque is important when using a motor. If the work used is a job that requires a lot of power, should choose a motor with a high torque value.

| Series | Basic Step Angle [FULL/HALF] | Max. Holding Torque [kgf-cm] | Rotor Moment Of Inertia [g·cm²] | Winding Resistance [Ω] | Rated Current [A/Phase] |
|---|---|---|---|---|---|
| Frame Size 24mm, Shaft Type, 5-Phase Stepper Motor K Series  | 0.72°/0.36° | 0.18 | 4.2 | 1.1 | 0.75 |
| | | 0.28 | 8.2 | 1.7 | 0.75 |

**Figure 2.20** Details of Stepper Motor specification.

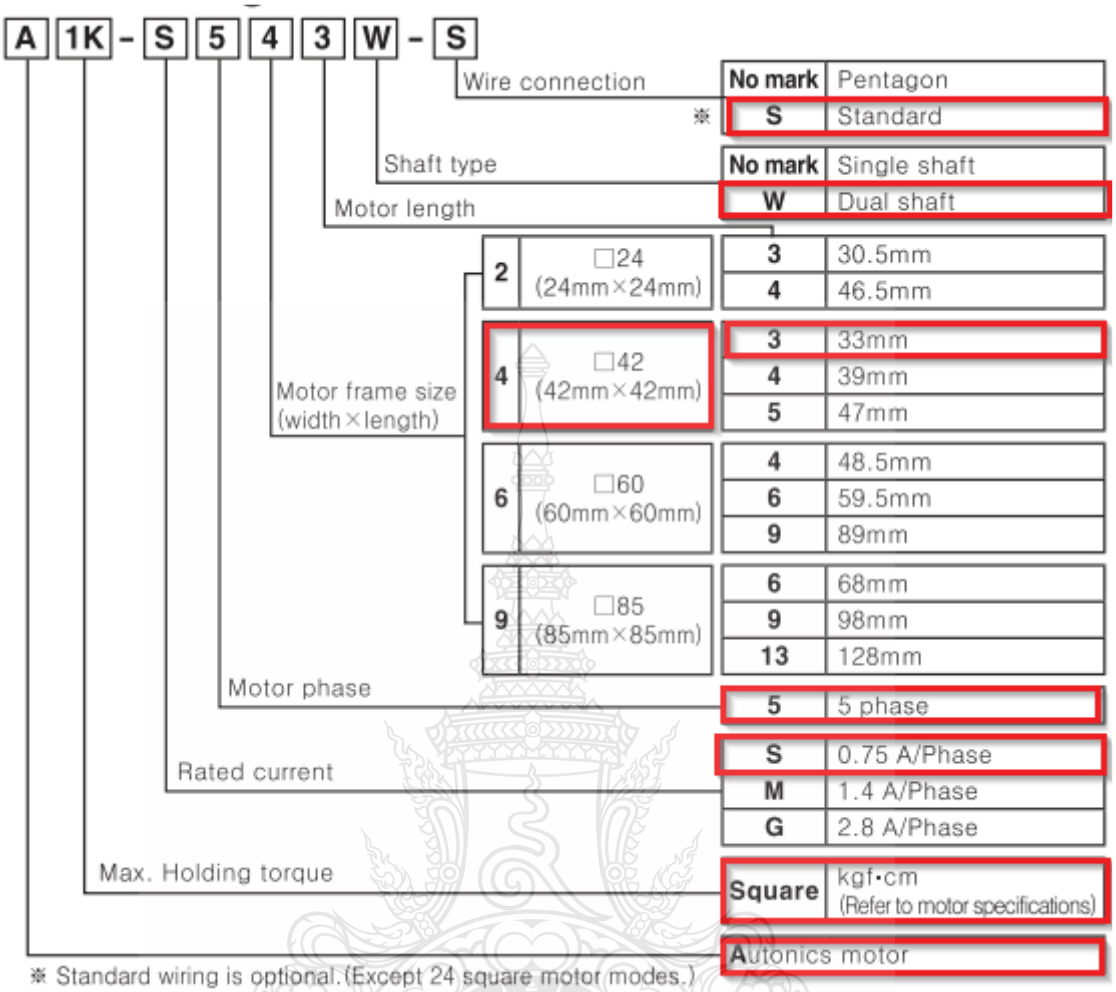



**Figure 2.21** Meaning of the model name of a stepper motor.

2.3.5 Working Principle of Stepping Motor, A Stepping Motor or Stepper Motor is a pulse driven electric motor. The internal structure consists of magnetic poles on the stator (Stator) made of steel rings. There will be protruding spokes assembled in layers. Each of the protruding teeth has a coil (coil) wrapped in it. When a current pass through the coil, an electromagnetic field occurs.

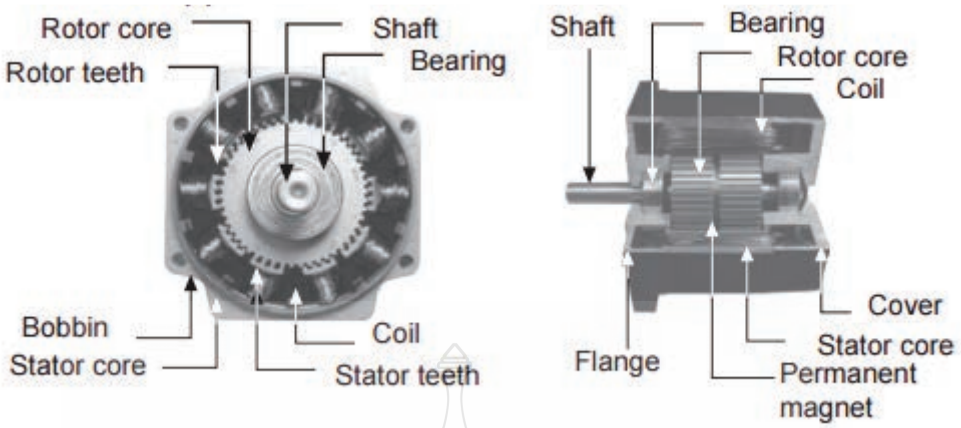



**Figure 2.22** Stepper Motor structure picture

In the operation of Stepping Motor or Stepper Motor, it cannot be driven or run by itself. It is necessary to have an electronic circuit that is used to generate a signal or supply pulses to the stepping motor drive circuit (Stepping Motor Drive). Viewing the location of the wires connected to the stepping motor

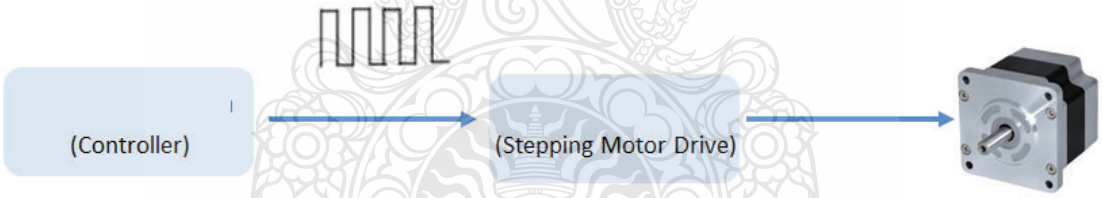



**Figure 2.23** Working Diagram of Stepping Motor

Stepping operating mode, since a Stepper Motor's stator consists of multiple windings, the electrical supply to excitation to different coil pairs will inevitably cause different work. can be separated as follows Figure 2.23 and Figure 2.24

Full Step Mode In this mode, when we supply 1 pulse, the motor will move 1 full step, for example, the NEMA 17 motor has a specification of 200 Step/Revolution. Therefore, if we want the motor to complete a complete rotation Must be paid in the amount of 200 Pulse, which will be divided into 2 types:

1-Phase on Stepping is to supply Pulse to each coil causing Rotor to rotate in different directions as shown in the Figure 2.23.

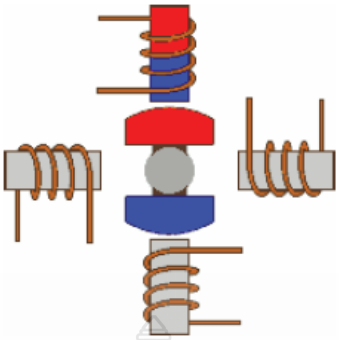


**Figure 2.24** Power supply single coils of Stepper Motor at a time

2-Phase on Stepping is similar to the 1-Phase type, except that we will supply pulses to both sets of coils simultaneously, meaning that we have to supply power to 2 coils when there is a magnetic field from 2 coils. Let's suck the rotor at the same time, making it more powerful. This method makes the motor have 30-40% more torque, but in exchange for using twice as much power as having to pay for the additional coil itself.

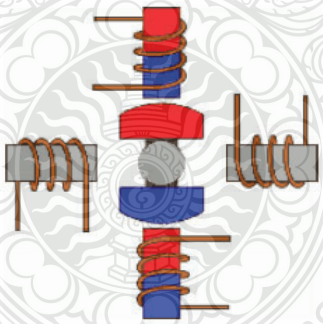


**Figure 2.25** Power supply to two coils of Stepper Motor at a time

If writing as a pulse distribution table into each step, it will be as follows Table 2.3

**Table 2.3** Power supply to 1-Phase and 2-Phase on Stepping in Full Step Mode

| 1-Phase on Stepping | | | | |
|---|---|---|---|---|
| Step | Wire 1 | Wire 2 | Wire 3 | Wire 4 |
| 1 | **1** | 0 | 0 | 0 |
| 2 | 0 | **1** | 0 | 0 |
| 3 | 0 | 0 | **1** | 0 |
| 4 | 0 | 0 | 0 | **1** |
| 2-Phase on Stepping | | | | |
| Step | Wire 1 | Wire 2 | Wire 3 | Wire 4 |
| 1 | **1** | **1** | 0 | 0 |
| 2 | 0 | **1** | **1** | 0 |
| 3 | 0 | 0 | **1** | **1** |
| 4 | **1** | 0 | 0 | **1** |

Half Step Mode Working in this Half Step Mode is the same as working in 1-Phase on Stepping and 2-Phase on Stepping combined, allowing us to get more resolution. Step itself, such as the original NEMA 17, if using Full Step, it will rotate 1.8 degrees per time, but if using Half Step, it will rotate only 0.9 degrees and of course, if you want to rotate 360 degrees, you need to pay more Pulse from 200 to 400 times that

**Table 2.4** Power supply to 1-Phase and 2-Phase on Stepping in Half Step Mode

| Step | Wire 1 | Wire 2 | Wire 3 | Wire 4 |
|---|---|---|---|---|
| 1 | **1** | 0 | 0 | 0 |
| 2 | **1** | **1** | 0 | 0 |
| 3 | 0 | **1** | 0 | 0 |
| 4 | 0 | **1** | **1** | 0 |
| 5 | 0 | 0 | **1** | **1** |
| 6 | 0 | 0 | 0 | **1** |
| 7 | **1** | 0 | 0 | **1** |
| 8 | **1** | 0 | 0 | 0 |

Micro Step Mode is the most complicated mode. But it produces fine spindle rotation, high torque and smoothest rotation. In this way, the coil is excited by two sin waves 90° apart from each other in phase. It can control the rotation of the motor axis in very fine detail (up to 0.007°/Step), which is very popular in tools such as 3D Printers, CNC machines.

**Figure 2.26** Power supply in Micro Step Mode graph

From the graph, it can be seen clearly that we supply electricity to the red coil. And the blue coil with a Sine Wave that is 90 degrees different from each other. The coil has more and less electricity, causing more and less inductance as well. More red coils, less blue hairs, the rotor core will be sucked to the red side. etc., thus allowing the stepper motor to move smoothly and precisely.

There are many ways to drive stepper motors, including full step, half step and micro step. Each drive method provides a different torque and step size that the stepper motor can use.

In full-step drive, both electromagnets are always "on". To turn the central shaft, one electromagnet is closed and the next electromagnet is open, causing the shaft to rotate 1/4 tooth (at least on hybrid stepper motors). This type has two electromagnets that are always on and has the highest torque of all types, but also the largest increment.

In the half-step drive, two electromagnets and only one electromagnet are switched on alternately. To rotate the central shaft, a first electromagnet is energized in a first step, and then a second electromagnet is energized while the first electromagnet is still energized. The third step turns off the first electromagnet, and the fourth step turns on the third electromagnet while still energizing the second electromagnet. As shown in the diagram above, this mode uses twice as many steps as a full step drive, allowing half the step length, but also less total torque because there are not always two electromagnets holding the center shaft in place.

Not surprisingly, in these styles, micro stepping has the smallest possible increments. One of the most common ways to perform micro stepping is "sin-cosine micro stepping". This means that the current flowing through each coil is manipulated to produce sine/cosine waves. The "overlap" of the waves between the two coils results in a large number of sub steps. The actual number of sub steps depends on the amount of significant variation in the current you can inject into the coil. However, of all the styles, micro stepping still has the smallest step size and therefore the most precise movement. The torque associated with this type depends on the amount of current flowing through the coil at any given time, but is always less than a full step drive.

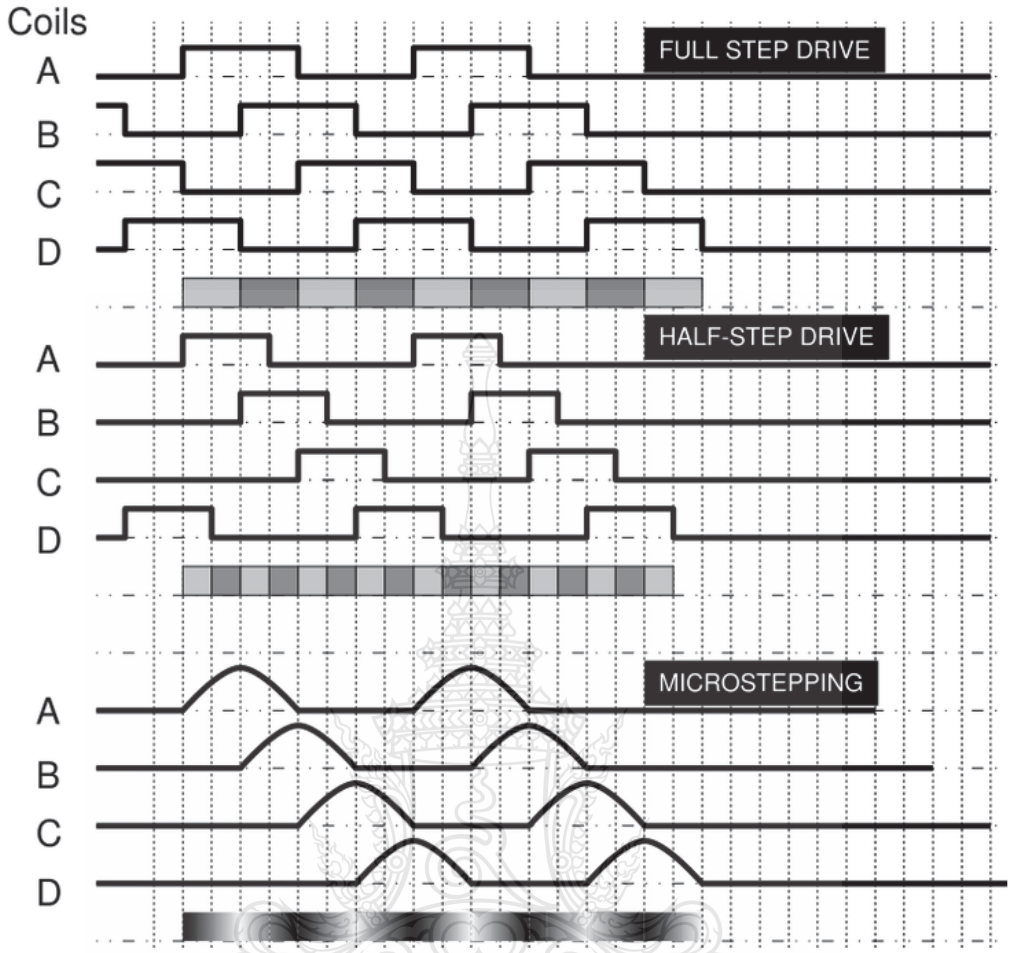



**Figure 2.27** Power input to various stepping motors.

### 2.4 Stepper Motor Driver

Stepping Motor Drive is a device that is responsible for driving the stepping motor to be able to move or work. motor to rotate Precisely moves to the desired position. By all these processes Must consist of 3 main devices: Controller, Stepping Motor Drive and Stepping Motor, Stepper Motor is a motor that has different operating characteristics from general motors. Because the pulse signal must be fed to the motor windings in a proper rhythm. And the rotation of this type of motor will rotate rhythmically according to the input pulse. If a continuous pulse signal is input the motor will be able to rotate continuously like a normal DC motor. Therefore, with the pulse input timing, the operator can select the position where the motor should stop turning.

The rotational stroke of a stepper motor is called step, hence the name of this type of motor. Motor resolution is defined in degrees of rotation in one step. If the motor has a large number of degrees per step, it means that this motor has a low rotational resolution. For example, one complete revolution is 360 degrees. If the motor has 7.5 degrees per step, the motor has 7.5 degrees per step. This motor has a rotation resolution of 48 positions, but if the rotation step is 1.8 degrees per step, the rotation resolution is 200, it can be seen that the latter motor has a much higher resolution than the first one. Makes it used in applications that require better positioning, more accurate, combined with a half-step drive circuit. The resolution of rotation is increased by 2 times, making the resolution of rotation becoming 400 positions. Therefore, Stepping Motor & Drive is an important device for Control the rotation in the movement of the rotation to rotate to the desired position. It receives signals from PLC or other automatic control devices. It is a device that controls the operation of the stepping motor to rotate to the desired position precisely. There are 2 types of circuit connection, shown in Figure 2.27 and Figure 2.28

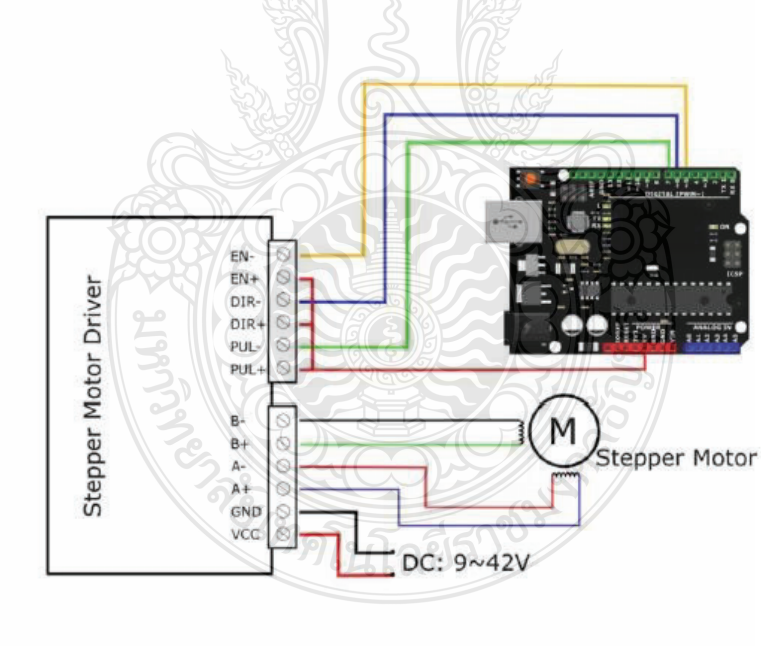



**Figure 2.28** Stepper motor driver with Microcontroller and Stepper Motor wiring diagram schematic type Common-Anode Connection

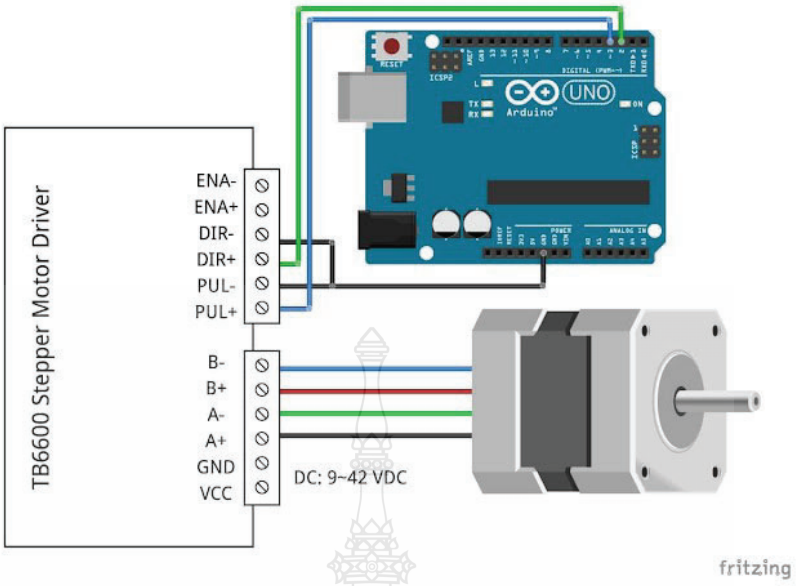



**Figure 2.29** Stepper motor driver with Microcontroller and Stepper Motor wiring diagram schematic Common-Cathode Connection

## 2.5 Encoder

Is a type of sensor that acts in encryption from the distance from the rotation itself and converted into code in the form of electrical signals, can bring these codes to convert back to find the values we want whether the rotation distance degrees of movement or speed of rotation and then brought to display the results for us to know the value through the display screen, for example, if wanting to measure the distance must be connected to the number counter To display the distance or if wanting to measure the speed around must be connected to the pulse meter by applying the encoder It can be used for a variety of tasks such as electronic assembly processes. semiconductor industry Various measuring tools, such as in length measurement or medical equipment industry, etc.

2.5.1 Rotary Encoder, it is used to measure the angle caused by the rotation of the encoder shaft. The output signal produced by the measurement has two forms.


**Figure 2.30** Rotary encoder

2.5 .2 Absolute Rotary Encoder, the working principle of this encoder is shown in the figure. The number of code bars (Digital bits coding) will be assigned to overlapping cycles in the line of Rotate with a reading head made up of a light sensor. (Photo-detector) is equal to the number of bar codes. Read the code while moving. The working principle is similar to Absolute Linear Encoder by changing the direction of motion to linear to rotation.

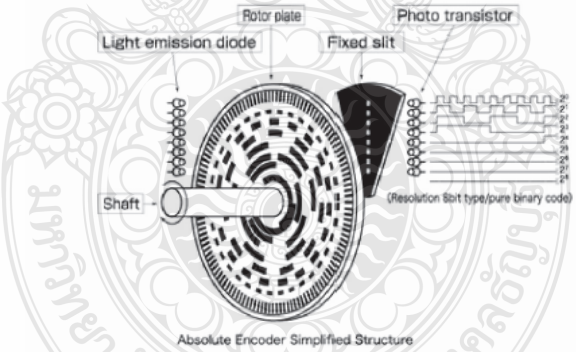



**Figure 2.31** Structure of the rotary encoder type Absolute Rotary Encoder

2.5.3 Increment Rotary Encoder, the working principle of this encoder is shown in the figure. Small straight bars are placed perpendicular to the direction of rotation. The reading head is made up of 2 optical sensors placed at an angle, so that while the rotary plate is rotated, the signal that comes out through the output in the form of a pulse (Pulse) will be phased with each other. 90 degrees, which is for checking the direction of rotation on the encoder shaft. So, resolution and accuracy So it depends on the number of small

straight lines that are placed perpendicular to the direction of rotation. This is usually stated in the form of the number of pulses per travel distance, such as 100 Pulse/rev. The working principle is similar to the Incremental Linear Encoder by changing the direction of motion to linear to rotation**.**

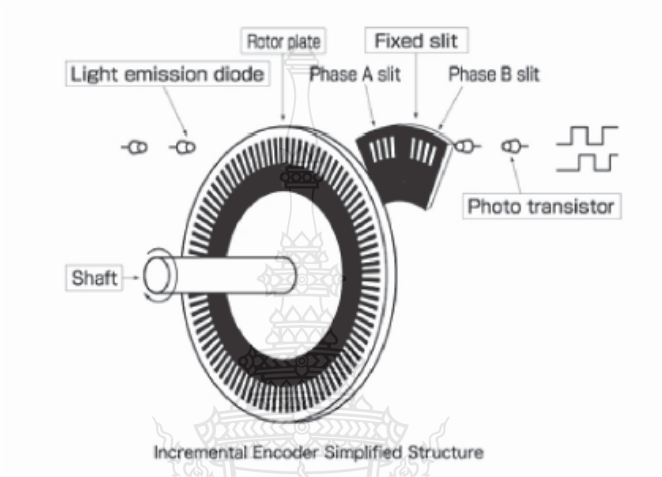



**Figure 2.32** Structure of the rotary encoder type Incremental Rotary Encoder

2.5.4 Knob Rotary Encoder module, this type of encoder works by counting by rotating forward and reverse during the rotation of the output pulse frequency.
These turns are unlimited compared to counter-rotating potentiometers. The button on the rotary encoder can be reset to the initial state, counting from 0.
Working principle: The incremental encoder converts the displacement signal of angular momentum into a series of digital rotary sensors.

These pulses are used to control angular displacement. Eltra converts angular measurements based on the photoelectric scanning principle. The reading system of alternating transmission windows and windows consists not of a rotating base of a radial index disk (codewheel), but of an infrared light source that shines light vertically onto the image of the codewheel onto the receiver. area on the surface. The receiver is covered with a diffraction grating with the same window width as the code wheel. The receiver's job is to sense the changes caused by the disk's rotation and convert the light into corresponding electrical changes. The low-level signal is then transferred to a higher

level, and there are no interfering square-wave pulses that must be processed by electronic circuits. Readout systems typically use a differential approach, where the two waveforms are roughly the same phase difference compared to the signal to improve the quality and stability of the output signal. The difference between the two signals is then formed from the difference between the two signals, thereby canceling the interference.

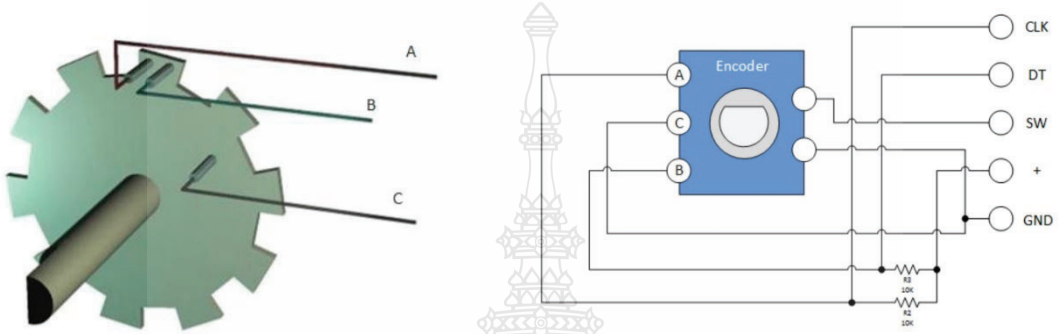



**Figure 2.33** Keyes Knob Rotary Encoder Schematic

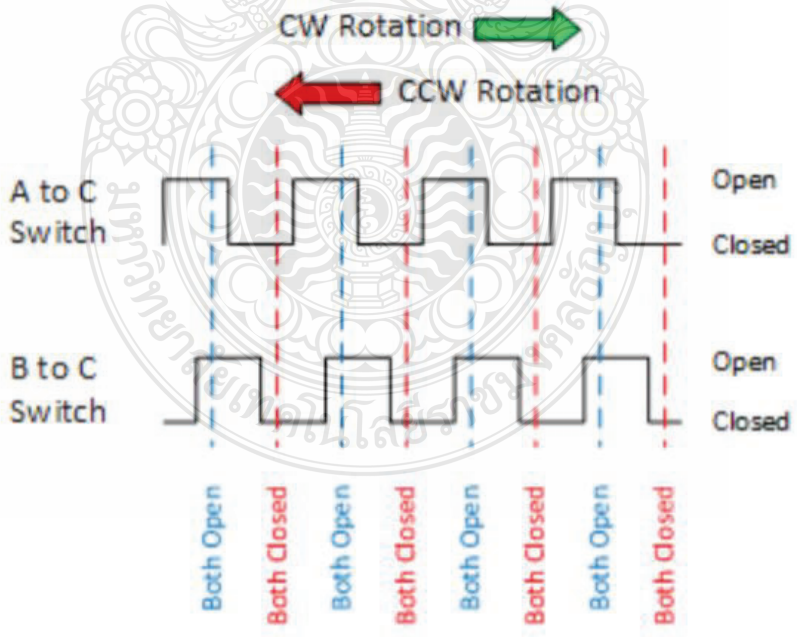



**Figure 2.34** Keyes Knob Rotary Encoder Direction and Signal Transformation Format

## 2.6 MATLAB Simulink Program

MATLAB program or Matrix Laboratory program was first developed by Dr. Cleve Molor [18 – 19], who wrote this program in Fortran language. This program was developed under the LINPACK and EISPACK projects. MATLAB program is a program designed for mathematical calculations. In general, Especially vector and matrix calculations. Both in the real number system and the complex number system This is highly suitable for use in computational analysis and system design. In all branches of engineering

A MATLAB program is a high-performance computer program used for technical computing. Programming and the display are combined in a single program effectively. In addition, the characteristics of writing equations in the program are similar to writing mathematical equations. Work that uses MATLAB programs such as general calculations. Modeling data analysis Visualization in general graphs and scientific and engineering graphs can be programmed in a graphical user interface.

The function of the MATLAB program can be run in the form of direct contact. (Interactive) is to write the commands one by one to allow MATLAB program to continue processing. Or able to compile those instructions into a program. One important aspect of the MATLAB program is that all data is stored in an array, where each variable is subdivided. In the MATLAB program, we don't need to reserve dimensions like most programming languages, so we can easily solve vector and matrix problems. This results in a significant reduction in execution time compared to programming in a C program or Fortran program.

2.6.1 MATLAB program principal system, the working principle of the program is divided into 5 main parts.

2.6.1.1 Development Environment.

2.6.1.2 MATLAB Mathematical Function Library.

2.6.1.3 MATLAB Language.

2.6.1.4 Handle Graphics

2.6.1.5 MATLAB Application Program Interface (API)

2.6.2 Development Environment, this section is a set of tools that allow us to use various functions and files. Many of these tools provide a graphical user interface, including MATLAB Desktop and Command, Windows, command history and browsers for viewing help, workspace, files and search paths.

2.6.3 The MATLAB Mathematical Function Library

In this section is the collection of parts of the program that have been compiled into sub-files. Each file. It is a file written to define characteristics in calculations or different algorithms, from simple functions such as addition, basic trigonometric functions such as sine, cosine, to complex functions. There are many computational steps, such as finding the inverse of a matrix, finding eigenvalues and eigenvectors, or fast Fourier transforms.

2.6.4 The MATLAB Language

This section is a high-level language that uses matrix or array variables that contain commands used to control program execution. Function operation the structure of variables, the input and output of the program, all of which help to make each program in MATLAB a smaller program compared to programs for the same purpose that the user has to write. Every step of the functionality comes up by itself.

2.6.5 Handle Graphics

This section is used to display graphics and images. Includes high-level commands used for rendering in two and three dimensions. Formatting in a way that image processing, animation. In addition, this section also includes a low-level language so that we can edit images. To be as we want as much as possible Including the creation of Graphic User Interface under the work of MATLAB as well.

2.6.6 The MATLAB Application Program Interface (API)

This section is a library that allows us to write programs in C language or Fortran and has a link to work with MATLAB. This section also includes programming and calling MATLAB functions to use. (dynamic linking), which allows MATLAB to function as a computational engine, including the ability to write or read MAT-files.

2.3.7 MATLAB command window used for this research

2.6.6.1 Editor/Debugger, in writing a program, known as M-file, can be written with a normal text editor such as Notepad because M-file is a program that uses letters in the style of normal ASCII Code and for MATLAB version 5 onwards, there is an editor

included. MATLAB as well, making it very convenient to use because besides being an editor, there is also a debugger to help edit the program. We are able to use Editor/Debugger To create and edit M-files, which is a program to call MATLAB commands or functions to run Editor/Debugger This will act as both a text editor for programming. And acts as a debugger, with tools that help in editing program in the event that a program error occurs Functions or commands that we write to use with MATLAB, we will call it MATLAB file or for convenience, we prefer to abbreviate it and call it M-file. It will look like the following figure 2.35.

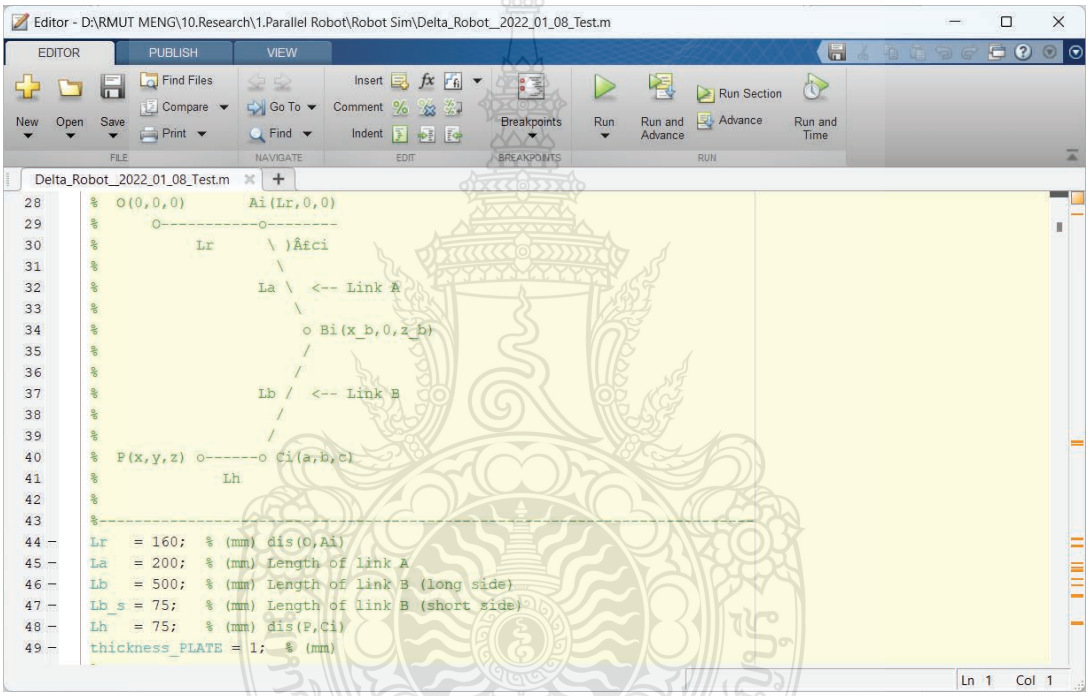



**Figure 2.35** MATLAB M-file functions

Which on Editor/Debugger This will have many tools to help us to be able to write programs more easily. However, the details of using Editor/Debugger This will be discussed in the chapter related to writing M-files.

2.6.6.2 Graphic Windows, when given a command to write a graph, MATLAB will display it on the Graphic Windows, which will start automatically. Depending on the

command assigned to MATLAB, more than one Graphic Window may appear at the same time. Menu Bar and so on. The Graphic Window will look like the following figure 2.36.

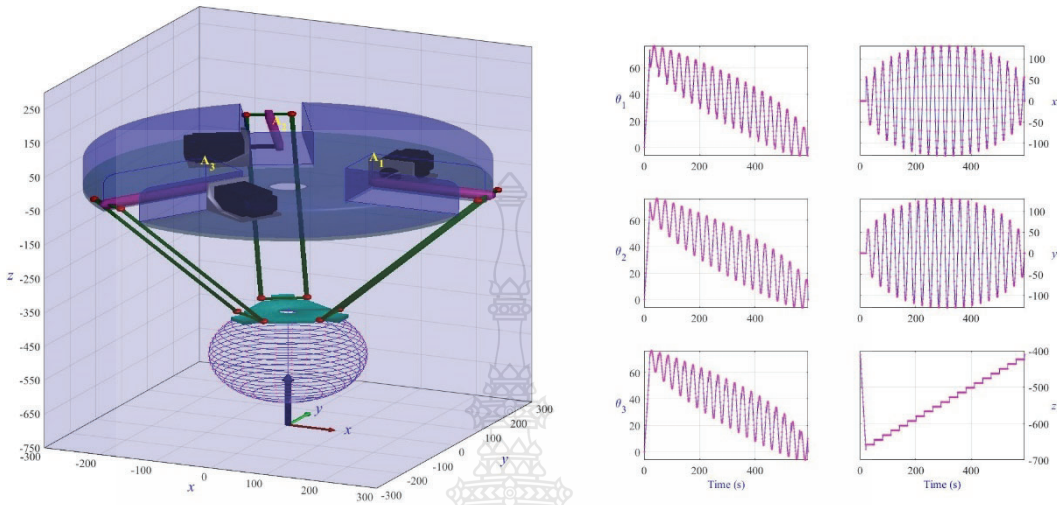



**Figure 2.36** MATLAB display it on the Graphic Windows

Which in this window, in addition to being used to display images It can also be used to create a Graphical User Interface to make programs that interact with the user using buttons. Like a program that works under general windows as well in addition, Graphic Window also has a tool that helps to write graphs more conveniently, especially in later versions of MATLAB. Add more graphs to the Graphic Window, whether adding text, adding lines, adding axis titles or graph titles. change of perspective Change the direction of the light that shines on the image. And much more, and in version 6, this is able to fit the curve that we wrote on the Graphic Window as well. For details on using various tools on the Graphic Window.

## 2.7 SolidWorks Program

SOLIDWORKS is a drawing and design program that was developed for use in the work. Product design, furniture design and 3D mechanical part design which has the following functions:

2.7.1 Part Solid construction uses the methods and technology of Surface Modeling (NURBS).

2.7.2 Assembly Modeling can assemble 3D parts faster. with the size of the file Smaller and uses less memory

2.7.3 Drawing automatically generates 2D drawings from 3D and You can save files as *dwg.

2.7.4 Simulation is used to test movement and check for conflicting parts.

2.7.5 Animator creates animations showing parts in action. or mechanical And can save files as *AVI (video files).

2.7.6 Sheet Metal can create various folding works. and can make plans to unfold sheet metal work and other application modules such as basic finite element analysis

SolidWorks program is a drawing assistant program that focuses on drawing in 3D or Parametric Solid Models. This refers to a 3D model that has a different texture than a 3D Wireframe type, which can be used with a 2D CAD program plus the imagination of the drafter can be written out. The Wireframe 3-D Model is therefore a model. 3D that consists of lines or lines continually, while the term Parametric Solid Model refers to a 3D model created by mathematical relationships which are computed within the program. This type of 3D drawing is convenient. with more authors

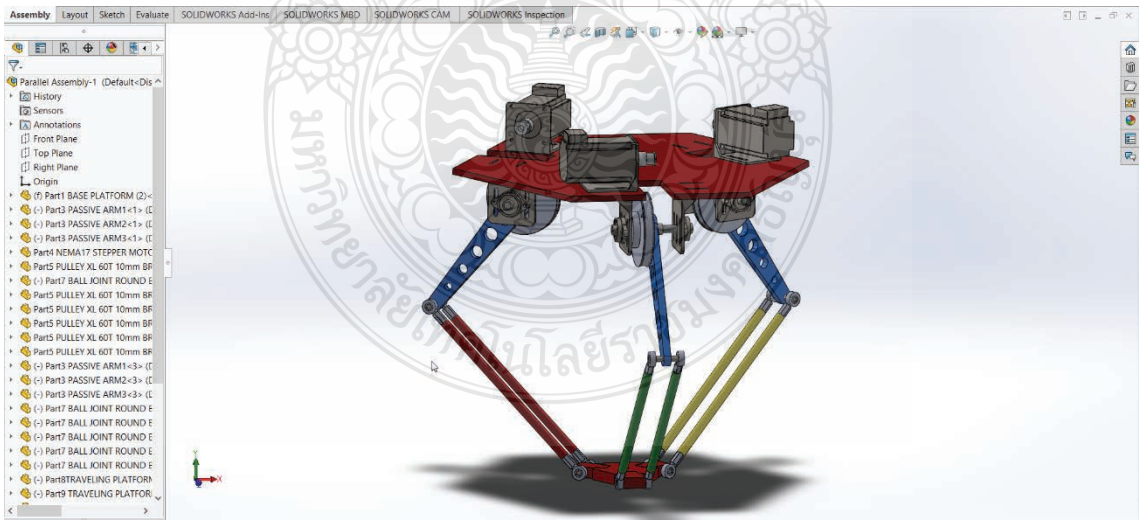



**Figure 2.37** SolidWorks program is a drawing Robotic 3-D Model

## 2.8 Arduino Microcontroller Board

Arduino is a microcontroller board that uses a small AVR microcontroller as a processor and command. Suitable for use in learning and developing microcontroller systems. And can be applied to control many different input/outputs devices, both Digital and Analog and can work both in a way that is a single independent operation or connect to work together with other devices such as PC computers. The hardware system of Arduino can be built and assembled and developed for use by yourself. The program that will be used as a development tool, can be downloaded and used for free. Arduino is an open-source microcontroller development system, making it easy to develop programs. And there are data sheets including examples. To be used as a guide to study and learn a lot. For this reason, it made people pay attention. and led to many studies and experiments has been adapted and create different types of projects together a lot Therefore, it is especially useful for beginners who can use examples as references as a guideline for studying and learning easily. [10, 11],

2.8.1 Components of the Arduino UNO R3 board

2.8.1.1 Reset switch is a Reset button for Reset Arduino Board

2.8.1.2 Port USB is a Port for connecting Arduino Board to the computer.

2.8.1.3 I/O Pin is Input and Output of Arduino Board (Digital I/O, PWM, Analog Input, Serial Port).

2.8.1.4 LED Pin13 is an LED that connects to I/O Pin 13 of Arduino Board.

2.8.1.5 LED Status TX/RX is LED Status of Port serial that is connected to Port USB.

2.8.1.6 LED Power is an LED indicating that the Arduino Board is working.

2.8.1.7 ICSP Interface is an Interface for Program Bootloader.

2.6.1.8 ATmega16U2 is a Microcontroller model ATmega16U2 that is used to control the conversion of Serial to USB Port.

2.8.1.9 ATmega328 is a Microcontroller (ATmega328) used on Arduino Board UNO, which when we write a program, it will be saved and work inside.

This microcontroller

2.8.1.10 Power Input is a port for connecting to an external power supply.

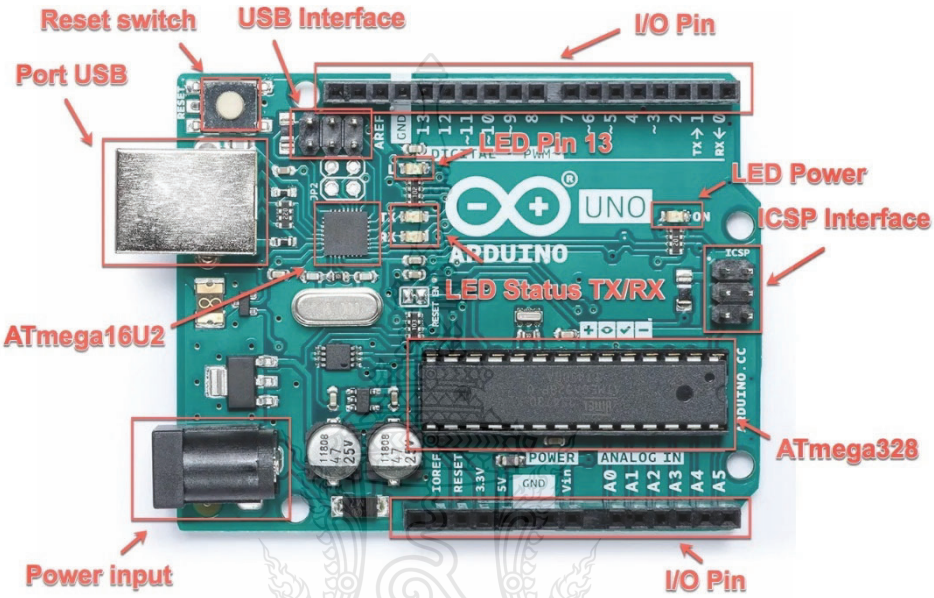



**Figure 2.38** Arduino Microcontroller Board

## 2.9 Program Arduino IDE C++

The Arduino software, also known as the Arduino IDE (Integrated Development Environment), is a tool used to develop programs on the Arduino platform and upload successfully developed programs onto the Arduino microcontroller board. And can choose to use the program as an online IDE or a desktop IDE.

2.9.1 Online IDE program (Arduino Web Editor) will be programming through the website. by various information that we have written will be stored on the Cloud, which makes it convenient to use anywhere in addition, we do not need to update programs or libraries.

2.9.2 The desktop IDE program is an offline program, that is, we have to download the program at [28] which will be available according to the usage of the OS

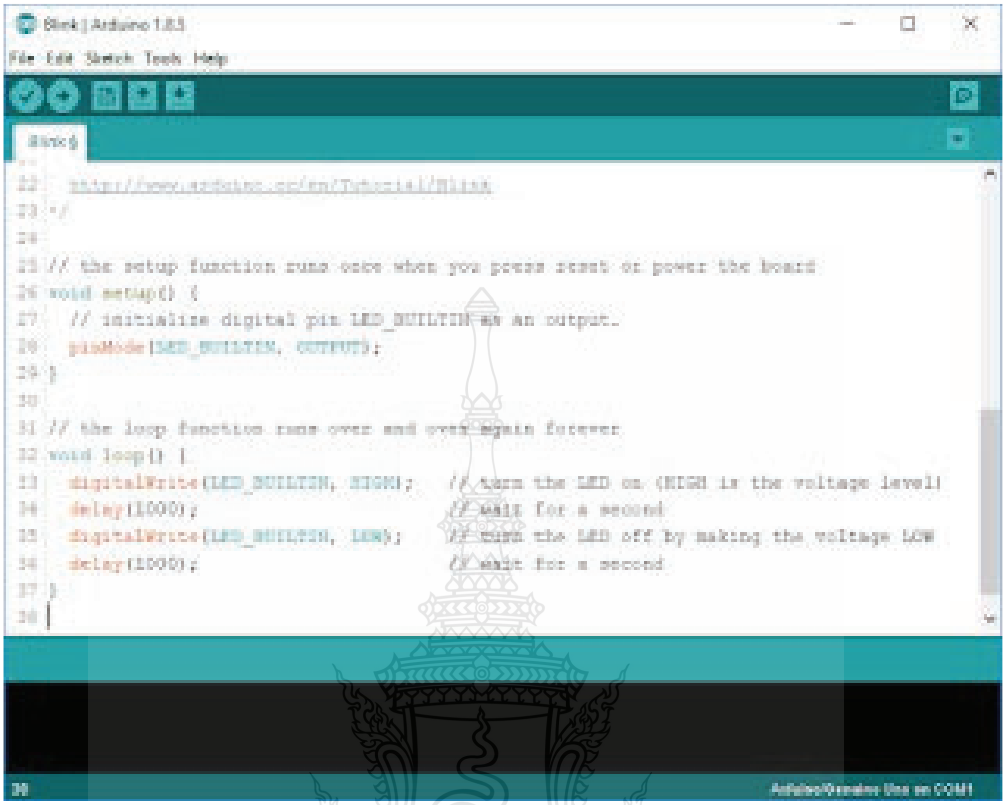



**Figure 2.39** Arduino Desktop IDE program


**Figure 2.40** Arduino Online IDE program

2.9.3 Advantages of Arduino IDE Software

It supports development in many model boards and can add other boards that are not Arduino boards to be able to develop programs and upload programs onto the

board. Within the Arduino IDE, there are various commands. program example and example libraries that has been installed and ready to use More libraries can be downloaded and installed. There is an online cloud for data storage. data can be retrieved and displayed.

2.9.4 The functions of the Arduino IDE program

2.9.4.1 Write a program in C/C++ language for Arduino.

2.9.4.2 Compile or transform C/C++ program to the microcontroller language and save it as Intel Hex File.

2.9.4.3 Upload the Intel Hex File onto the microcontroller on board Arduino via USB cable or via Programmer.

2.7.5 Components of the Arduino IDE program

2.9.5.1. The menu bar contains File, Edit, Sketch, Tools and Help.


**Figure 2.41** Arduino IDE program menu bar contains

2.9.5.2 The File menu has the following submenus:

1) New: Used for creating a new window for writing Sketch.
2) Open: Used to open the Sketch that we saved.
3) Open Recent: Used to open the Sketch that we used to open. There will be a name for us to choose from.

4) Sketchbook: is used to always open the last activated sketch file.

5) Examples: is used to open the sample program in the Arduino IDE program.

6) Close: is used to close the Arduino IDE program only for the function that we press to close.

7) Save: is used for saving the sketch that we are currently writing. If we haven't recorded before It will pop up a window for us to enter a name and select a recording area.

8) Save As: Used to save the name of a different Sketch.

9) Page Setup: is used to set the printing area of the printer.

10) Print: is used for printing in the control program area to the printer.

11) Preferences: Used to open some settings windows of the Arduino IDE program to make it suitable for use.

12) Quit: is used to close every window of the Arduino IDE program.

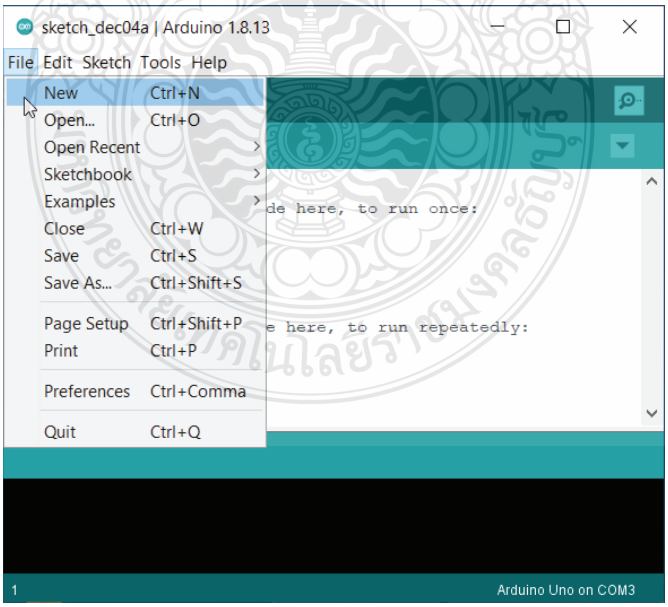



**Figure 2.42** Arduino IDE program File menu

2.9.5.3 The Edit menu has the following submenus:

1) New: Used for creating a new window for writing Sketch.1) Undo is used to go back to the previous work.
2) Redo is used for moving the operation up 1 time (This command can be used when Undo is pressed first).
3) Cut is used to delete the text that we typed in the control area. (That text should be highlighted before using this command.)
4) Copy is used for copying the text that we highlighted in the control program area.
5) Copy for Forum is used to copy all messages in the control programming area. to be placed in various document programs in the original manner
6) Copy for HTML is used to copy all text in the control area. to be placed in various document programs in the form of HTML
7) Paste is used for pasting text that we have copied. into the control programming area
8) Select All is a highlighted selection of all text.
9) Go to line is to set the position of the Keser to the position we want.
10) Comment/Uncomment Used to assign to the line where the Kaiser is located. Descriptive text or non-descriptive text?
11) Increase Font Size is used to increase the size of the text in the control program area.
12) Decrease Font Size is used to reduce the size of the text in the control program area.
13) Find is used to search for text or search for text while making changes to that text.

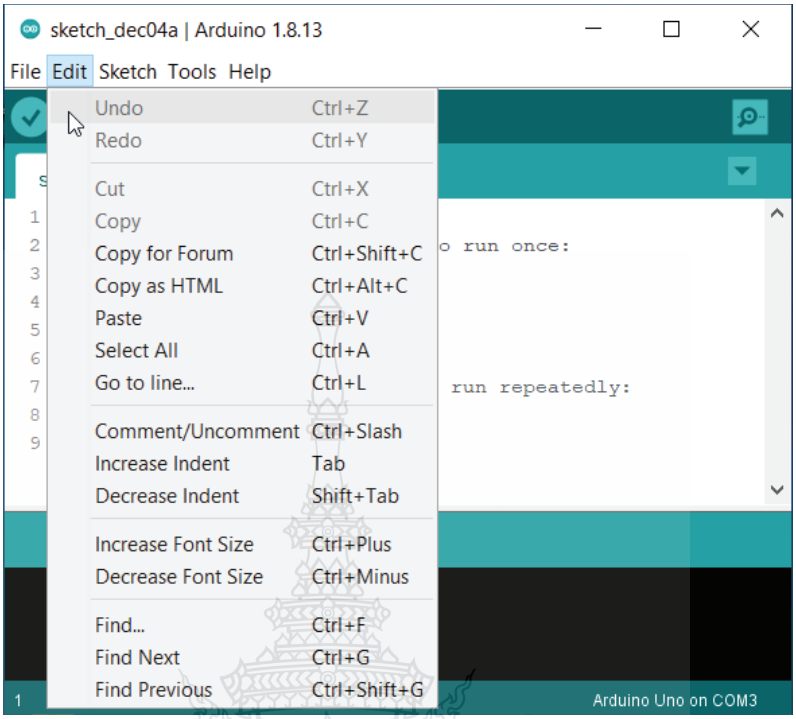



**Figure 2.43** Arduino IDE program Edit menu

2.9.5.4 The Sketch menu has the following submenus:

1) Verify/Compile used for checking the code that we have written whether there is a mistake or not along with compile
2) Upload is used to compile the code that we have written. and upload directly to the Arduino board.
3) Upload Using Programmer is used to compile the code that we have written. and uploaded to the programmer
4) Export compiled Binary is used for compiling the code that we have written. And convert it to a HEX file into the folder of the program that we saved.
5) Show Sketch Folder is used to see the location of the Sketch file that we are currently writing. at which position
6) Include Library is used to call the Library that we want to use.

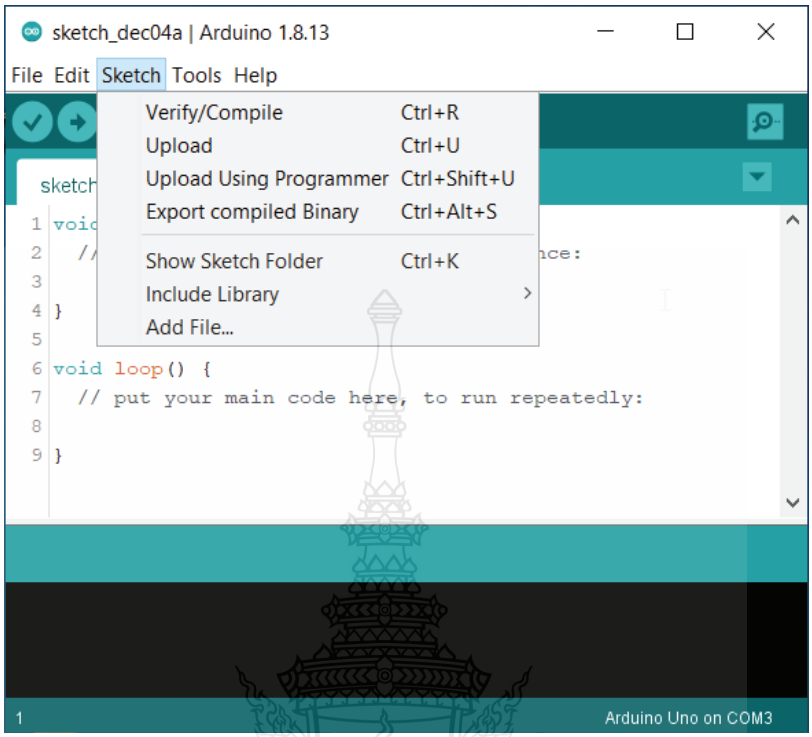



**Figure 2.44** Arduino IDE program Sketch menu

2.9.5.5 The Tools menu has the following submenus:

1) Manage Libraries Used for accessing to manage various libraries.
2) Serial Monitor Open the Serial Monitor window (only active if the board is connected to the computer)
3) Board: "……." is used to select a board to match the board we use.
4) Port is used to select Com Port that Board Arduino is connected to.
5) Burn Bootloader is used to load part of Bootloader into the Arduino board.

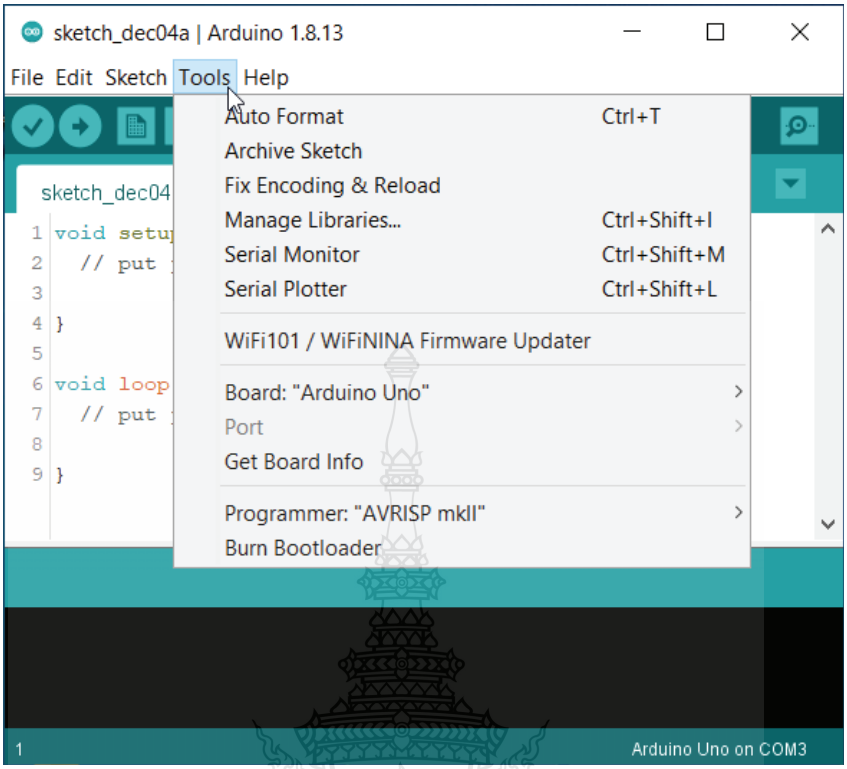



**Figure 2.45** Arduino IDE program Tools menu

2.9.5.6 The Help menu has the following sub-menus.

1) Getting Started is a link to learn how to use basics from the Arduino board, program download, program installation, and driver installation of the board, etc.

2 ) Environment This is a link to learn about using the Arduino IDE program.

3) Troubleshooting It's a link to collect problems. and solutions

4) Reference is a link to the page to learn various commands.

5 ) Find in Reference is used to find references of various commands. By placing the cursor at that command and then pressing this command

6 ) Frequently Asked Questions is a link to frequently asked questions.

7 ) Visit Arduino.cc is a link to the website. https://www.arduino.cc8 .About Arduino is used to view the version of the program Arduino IDE.


**Figure 2.46** Arduino IDE program Help menu

2.9.6 Shortcut menu

2.9.6.1) The Verify: is used to verify the code that we have written. Whether there is a mistake or not along with compile

2.9.6.2) Upload: is used for compiling the code that we have written. And upload it to the Arduino board.

2.9.6.3) New: is used to create a new window to write Sketch.

2.9.6.4) Open: used to open the Sketch we saved.

2.9.6.5) The Save: is used to save the sketch that we are currently writing. If we haven't recorded before It will pop up a window for us to enter a name and select a recording area.

2.9.6.6) The Serial Monitor: opens the Serial Monitor window (will only work if the board is connected to the computer)


**Figure 2.47** Arduino IDE Program Shortcut menu

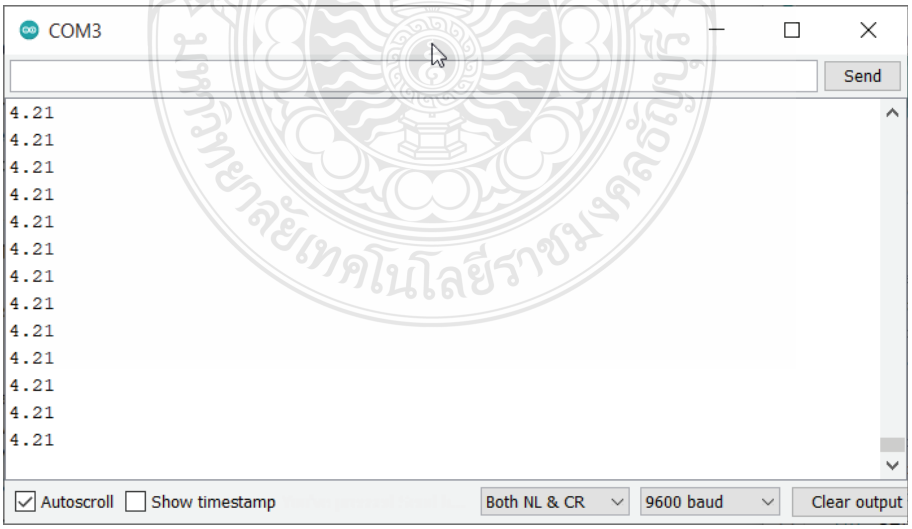



**Figure 2.48** Arduino IDE Program Serial Monitor window appearance

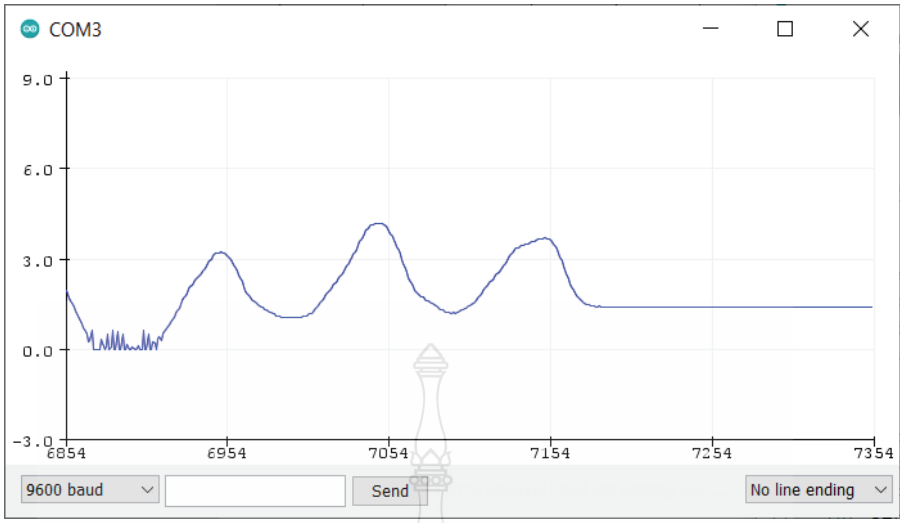



**Figure 2.49** Arduino IDE Program Serial Plotter window appearance

The Serial Monitor window plays a large role in displaying the performance of the program instead of using Other display devices because Arduino has prepared a command to display the value of the variable that you want to see the result of, that is, Serial.print. to the Arduino hardware or controller board provided Type a message and click the Send button. In order to send data, you must set the baud rate for the program in the Serial.begin command. Set the Linux operating system, the Arduino hardware resets when the serial monitor is started.

2.9.7 Basic Arduino Board Test Commands that are used in the Arduino program will be used to create a button on the toolbar so that it can be clicked. Immediately, the buttons on the toolbar: Select an Arduino board by selecting it from the Tools > Bord menu. The program will show the available boards in computer for users

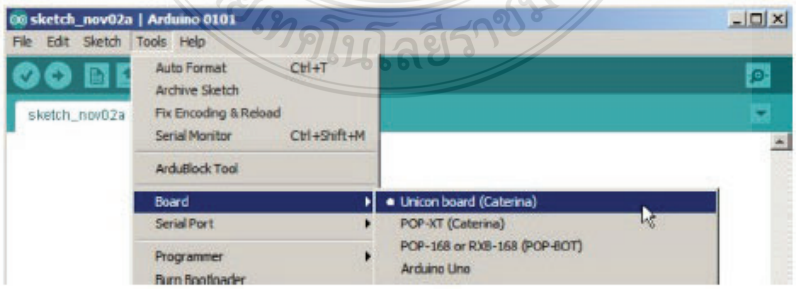



**Figure 2.50** Selecting the type of Arduino board connected

Select the COM port used to connect to the Arduino board by selecting it from the Tools > Serial port. Shows the available ports on the computer for the user to choose from in the example select COM3 port.

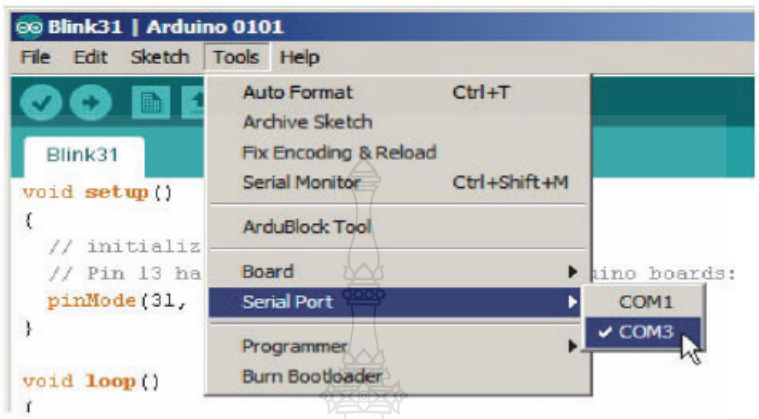



**Figure 2.51** Arduino comport selection window

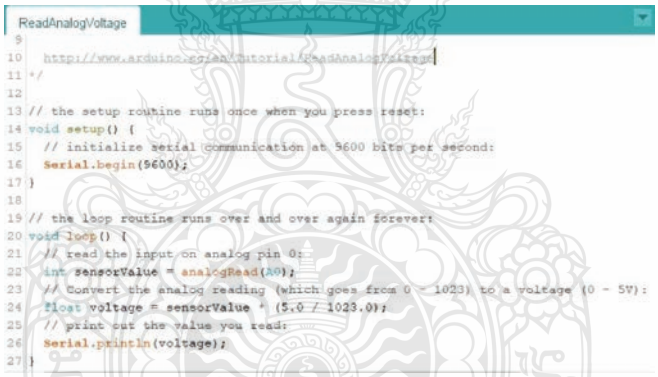



**Figure 2.52** Sample code for testing uploading program into board Arduino

Upload the program to the Arduino board by clicking the Upload button or selecting File upload menu. Wait until the upload is complete. The board will work immediately. Blinks every 1 second to complete the test. Writing and uploading the basic program with the Arduino board. If there is an error uploading the program, it will cause the Arduino board to be inoperable. You must check from the orange message below to see what is wrong.

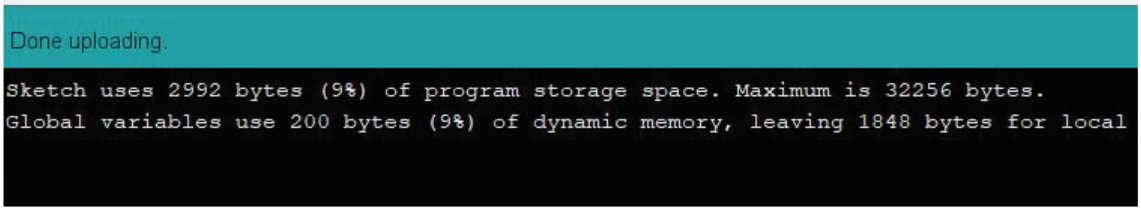



**Figure 2.53 shows that the program has been successfully uploaded**

When the program is compiled, the status bar and the compilation display window, which is a black window at the bottom of the program, the status bar will display the message Done Compiling and the display window will say Binary Sketch. Size: 2,992 bytes (of a 32,256 Byte Maximum) indicates that a machine language program as of compilation, it is 2,992 bytes out of the total usable memory capacity of 32,256 bytes.

## 2.10 Machine Vision

"Vision system" has played a huge role in the modern industrial system. Due to the need for consistent quality work Accurate and fast production process to meet consumer demand. These processes used to be tasks that required human resources to do. But there were many problems due to the uncertainty, fatigue, insufficient intelligence, inferior development, which caused the work to come out with uncertain quality and delay. Therefore, the system was invented and developed to replace it. For this reason, the vision system was born. [5 - 9],

Vision system can therefore be defined simply as follows: a system that processes data from images or videos. to obtain significant information

Because the vision system is widely used many elements therefore can be changed as appropriate but generally classified as follows:

2.10.1 The image receiving (Vision Sensor), which can receive this image in several ways, such as Load image files from the source, capture images through a camera or capture through a video camera, etc. This part is like the human eye itself.

2.10.2 Vision Lighting. This part of the lighting not only helps to see. But if installed properly and properly, it will also help the system work faster. and more accurate

2.10.3 The information processing (Vision Processor) is comparable to the

human brain. It is for thinking and analyzing to find information from what is seen. It also serves to make decisions in response to the analyzed data. Most of the processing is a computer. or a specialized microprocessor

2.10.4 Objects to process (Object of Interest) These objects are anything that we need to get some analytical data from it for further use.

**2.11 NI Vision LabVIEW Program**

LabVIEW is a program created for measurement and instrumentation in engineering. LabVIEW stands for Laboratory Virtual Instrument Engineering Workbench, which means it is a program that creates Virtual measuring instruments in engineering laboratories Therefore, the main purpose of the work of this program is to manage the measurement and measuring tool effectively And, in the program, will consist of functions used to help measure many and most certainly. This program is extremely useful when used in conjunction with engineering instrumentation. The most obvious difference between LabVIEW and other programs is that LabVIEW is a completely GUI (Graphic User Interface) program. Write any code or instructions, and most importantly, the language used in this program is called a picture language, also known as the G language (Graphical Language), which instead of programming in lines as we are familiar with. BASIC such as C, BASIC or FORTRAN with all images or symbols. which, although at first, may be somewhat confusing but once you get used to using this program, you'll find that LabVIEW is convenient and can save a lot of programming time. Especially in computer programming to connect with other devices. for use in measurement and control as a primary goal, National Instrument began to develop a program that would be used in instrumentation systems with ease of programming and functionality to assist in engineering measurements as much as possible. Instrument began to produce equipment used in engineering measurements. It's not a company that started primarily as a software production company, so it wouldn't be wrong for those who want to get the most out of LabVIEW, those who want to bring data from outside the computer. into the machine for data analysis, processing, display and in many cases used in computerized automatic control systems. The greatest advantage of LabVIEW is that our computers, combined with LabVIEW and the Data Acquisition Card, can turn

our personal computers into measuring instruments in many ways. Whether it is an Oscilloscope, Multi-meter, Function Generator, Strain meter Thermometer or other measuring instruments. as we want This makes it possible to use computers to measure and measure widely. This is where the name comes from. Virtual instrument and the advantage over using those real devices is that the Virtual Instrument can be adjusted to suit the use of each group of users. By changing the VI to suit your needs, it's not that difficult. [5 – 9, 20],

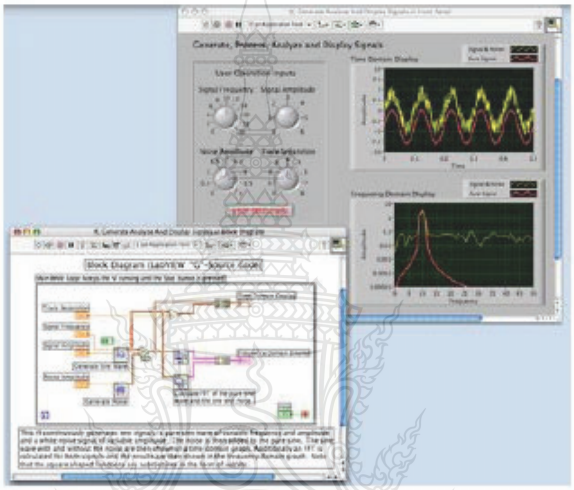


**Figure 2.54** The programming screen and the display screen

Another advantage of using a computer as a measuring tool is that Can be used as a Data Logger and PLC (Programmable Logical Controlled) at the same time, which is usually not a control system in a basic real measuring device or Data Logger, although it can store data. But the order to work with other devices It will be very difficult to order.

For those who have used text-based programs Known as Text Base, everyone would know the difficulty of dealing with the location of the data transmission by connected devices such as Port or Card, including the positioning in memory in order to be able to collect data for use. Calculate and store data to get the most out of it. Many of these problems are addressed in LabVIEW, which contains many programs or libraries to address them. Whether the connecting device is a DAQ (Data Acquisition), GPIB (General Purpose Interface Bus, formerly known as Hewlett Packard Interface Bus, HP-

IB), a serial port, or a Serial Port to communicate with the sending device. through serial instruments, as well as various data analysis methods. In addition, those libraries contain many important functions such as signal generation, signal processing, filters, statistics, algebra, and other mathematics, so LabVIEW makes measurement and instrumentation much easier. And make our personal computers into a variety of measurement tools in one device.

2.11.1 Data Flow and Programing

Since LABVIEW is an image-based program, or symbols instead of writing with characters like normal programs the first benefit is the reduction of misspelling or typographical errors. Another important difference between G programming and literal programming is that Written in G language, this is written using the principles of Data Flow, which when starting to send data into the program. We need to determine where the data flow will go. In what areas has it been evaluated and calculated? and how to show results the style of writing G language or Data Flow is similar to writing a Block Diagram, which allows programmers to pay attention to the movement and change of data without having to memorize complicated commands.

Because LabVIEW uses a block diagram style that most engineers are already familiar with. Therefore, it is easy to understand and can be further developed. And if we remember the programming process that before writing the program, the Flow Chart must be completed first. After checking the Flow Chart, we then use it to write a program. which will be more convenient If writing Flow Charts in LabVIEW is programming, this greatly simplifies the workflow. Programming in LabVIEW does not require any prior programming knowledge, but Have knowledge in programming or use other ready-made programs can be utilized as well

LabVIEW provides a front panel that is like what the user will see and control. Users can create their own designs quickly because LabVIEW includes components. Used for designing many screens such as opioid displays, knobs (Dial) and switches, etc. LabVIEW will show results and control operations via a computer.

The programming area is called Block Diagram. It is like a hardware inside a measuring instrument. LabVIEW writes a program based on images.

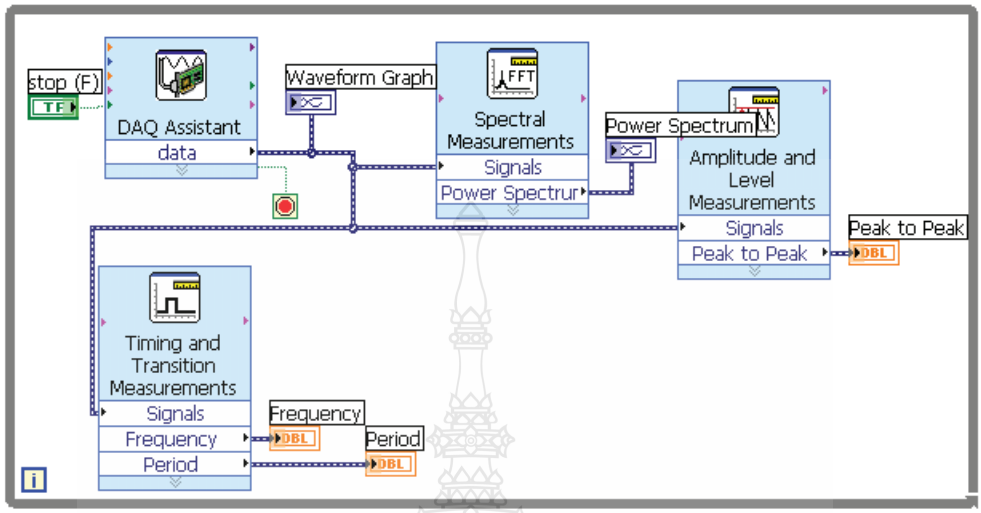



**Figure 2.55** Block Diagram of LabVIEW Program

2.11.2 LabVIEW relies on the principle of instrumentation or control, allowing users to design according to their needs. These principles are divided into three main parts:


**Figure 2.56** Block Diagram generated from LabVIEW Program

2.11.2.1 Acquisition, which is the part that receives data (Input) from the external environment into the system in this case is the computer. The data entered into this system may come from the DAQ card (for electrical signals).

2.11.2.2 Analysis After receiving the data, it may be through a function to analyze the data. This will be displayed in a meaningful form that the user can display instead of a measurable and usable medium.

2.11.2.3 Presentation is a form of display that is useful to users. It may be displayed on a computer screen such as a DMM (Digital Multimeter) showing specific results that are measured without needing to know the importance of time or Spectrum Analysis showing signals in terms of frequency or printing out a report or storing data in a hard disk

2.11.3 Components in LabVIEW

Programs written in LabVIEW are called Virtual Instruments (VI) because their appearance on the screen when the user operates is similar to that of an engineering instrument or device. At the same time, behind the scenes of those virtual devices is the execution of functions, subroutines, and the main program just like a normal language. For a VI, it consists of 3 components:

A. Front Panel
B. Block Diagram
C. Icons and Connectors

These three parts make up a virtual reality device. The characteristics and functions of the three components are as follows.

2.11.3.1 Front Panel or dial It is the part used to communicate between the user and the program. In general, it looks like the dial of the instrument or device used in general measurement, generally consisting of Toggle switches, knobs, push buttons, displays, or even user-defined values. For those who are familiar with Visual programming, it is well understood that this Front Panel is like a GUI of a program or VI.

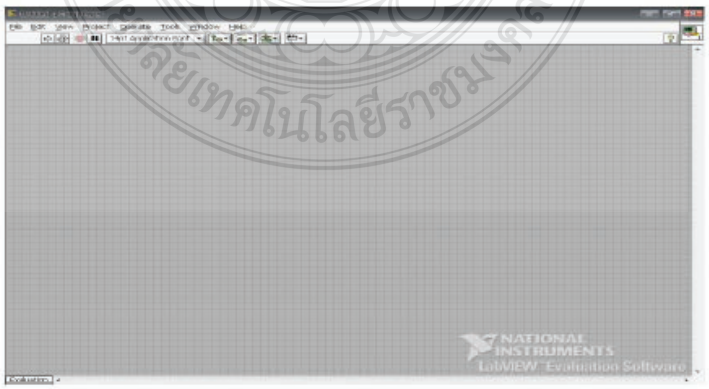



**Figure 2.57** LabVIEW Program front panel

There are three types of objects on the front panel:

1) Control is a type that receives value from the user (Input), which the user can type in. Or use the mouse to click to change values such as rotary buttons, scroll buttons, switches, etc.

2) Indicators are the type used to display various values only (Output) that cannot be modified by the user, such as graphs, meters, LEDs.

3) Decorations are objects that are not related to the program and code on the Block Diagram, but are only used for the beauty and order of the Front Panel. The Front Panel's appearance is shown in the following figure.

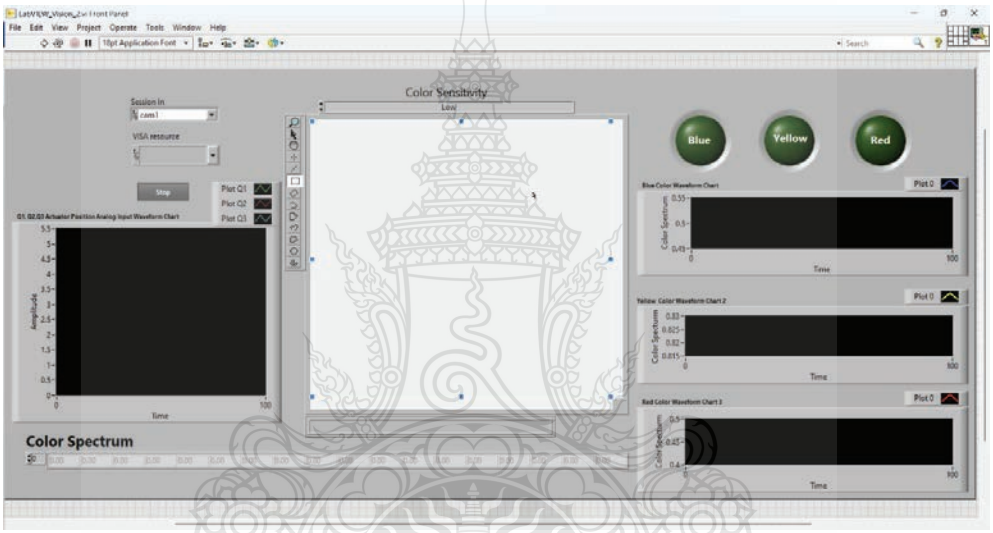



**Figure 2.58** Objects on the Front Panel of LabVIEW

2.11.3.2 Front panel design tools, the front panel design tools consist of a control palette and a tools palette. LabVIEW has a control palette used to design the front panel, shown in Figure 2.8, which is the user interface. Groups such as numbers (Numeric), within which there are various Control and Indicators related to numbers.


**Figure 2.59** Controls Palette used in Front Panel design

Tools Palette is a tool used in program development. Which will use both Front Panel design and Block Diagram. In this section, we will discuss Tools Palette for Front Panel design.

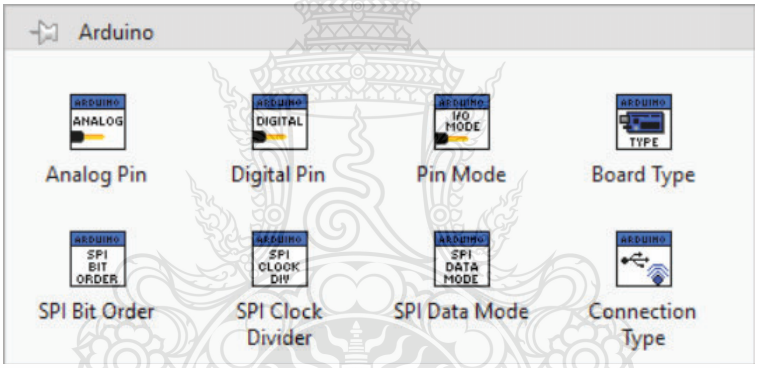



**Figure 2.60** Tools Palette used to design the Front Panel.2.4 Block Diagram

To make it easier to understand, we may look at this Block Diagram as a Source Code or LabVIEW program, which appears in the form of G language. This Block Diagram is considered an Executable Program, meaning it can be run immediately. Another advantage is that LabVIEW always checks for program errors. Makes the program work only when there are no errors in the program. Users can view the details of the error at any time, making programming easier.

Components within this Block Diagram are comprised of functions, constants, control programs or structures. Then in each of these sections This will appear in the form of a block. We will be wired together (Wire) for the appropriate block together. To define

the flow of data between those blocks, allowing the data to be processed as desired and displayed to the user.

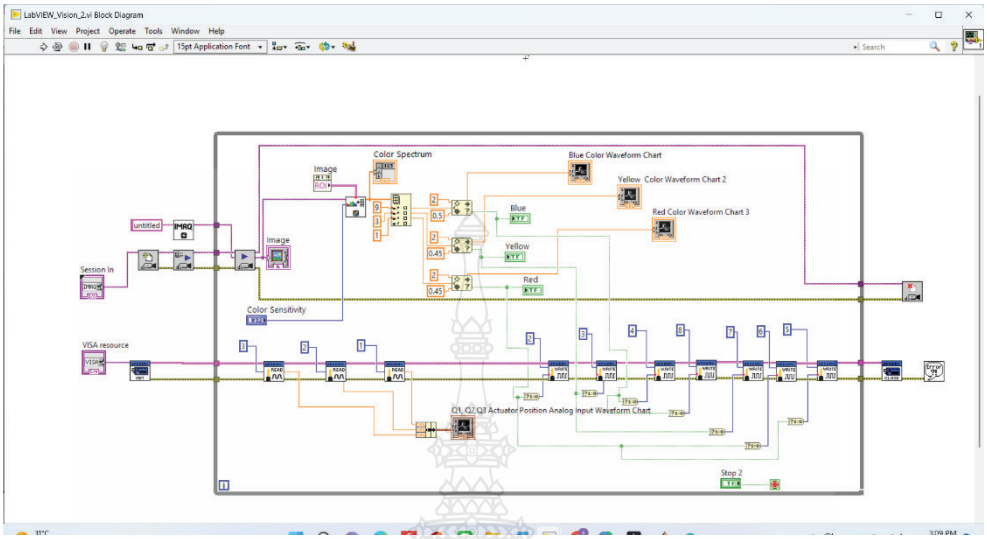



**Figure 2.61** Example Block Diagram

2.11.4 Image Processing, To identify colors in still images in LabVIEW, the visual aid tool IMAQ ColorMatch VI [25] was used. The Color Processing VIs perform basic processing of color images. Use these VIs to compute the histogram of a color image. Apply the lookup table 2 and Figure 12 to the color image. Change the brightness, contrast, and gamma information associated with color images. and Threshold:ColorImage finds a match between the color content of multiple regions in the image and that defined by the input color spectrum. Chromatograms are exported by IMAQ ColorLearn VI. [20],

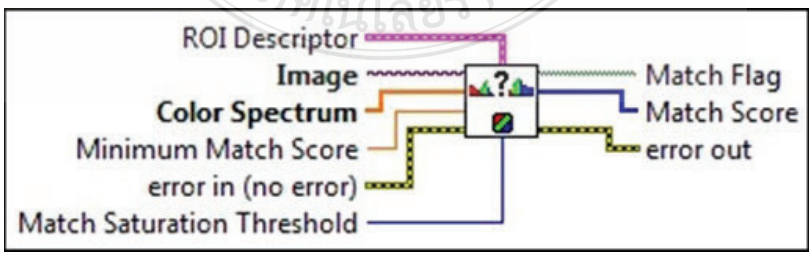



**Figure 2.62** NI Vision Development Module

**Table 2.5** IMAQ COLORMATCH VI IMPLICATION [20],

| | |
|---|---|
| ROI Descriptor | Specifies the descriptor of the region in the image whose color information is to be compared with that in the color spectrum. |
| Image | Reference to the color image from which to extract color features to compare them to the colors defined by the Color Spectrum input color features array. |
| Color Spectrum | Contains information about matching colors. During the matching phase, the color information in each specified image region is compared to the information contained in this array. |
| Color Sensitivity | Specifies the sensitivity of the color information in the image. Default is low. Set this option to High if you need to distinguish colors with similar hue values. |

## 2.12 NI LabVIEW Interface for Arduino Microcontroller

It is an add-on program for LabVIEW that is more versatile and convenient when it can be connected with the Arduino program, which will make the program development divided into two parts. The first part is to develop a program from LabVIEW called LabVIEW Interface For Arduino and the second part is to write the functionality of the program from Arduino in the wiring language which is successfully developed into a library called LIFA Base. [10, 11],

For Interface between LabVIEW and Arduino In the experiment, there will be programming in 2 parts: LabVIEW and Arduino to make development easier. Therefore, LabVIEW Interface for Arduino (LIFA) Toolkit was developed. The idea is to generate Arduino code and load it to the Arduino Board and create subvi LabVIEW to be easily run. But behind-the-scenes work is also a serial communication between Arduino and LabVIEW.

a) Arduino Code is LIFA_Base.Ide

b) Subvi LabVIEW such as Init, Close, Write Digital, Read Digital, Write Analog, Read Analog, etc.

LabVIEW Interface For Arduino installation is to install a basic program to run LabVIEW method together with using Arduino program. following

2.8.1 Install the NI-VISA Drivers so that LabVIEW can use the additional functions of the Serial Port.

2.8.2 Install JKI VI Package Manager (VIPM) Community Edition (Free) to find and install LabVIEW add-ons and Toolkits from the LabVIEW Tools Network.

2.8.3 Open VI Package Manager program and search for LabVIEW Interface For Arduino extension program and install Toolkit as shown in Figure 2.29

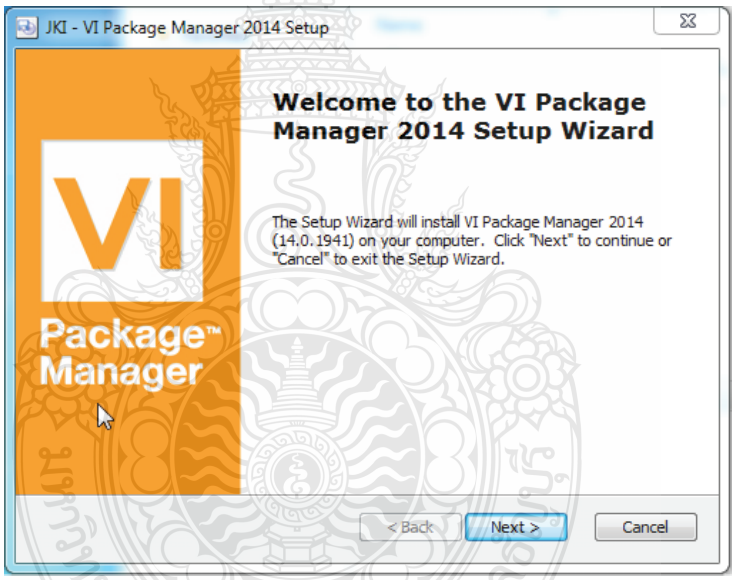



**Figure 2.63** NI-VISA Drivers Software

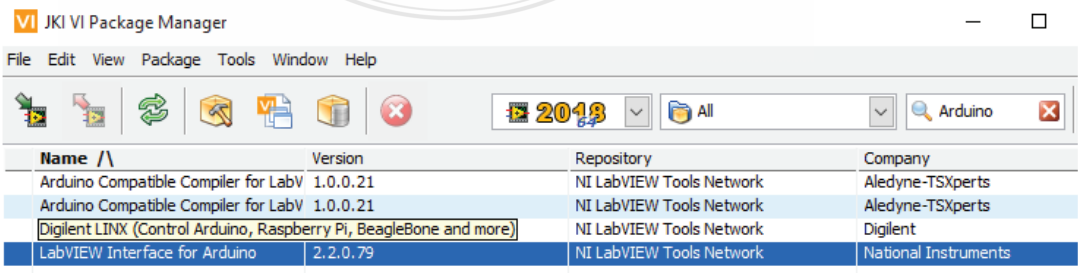



**Figure 2.64** JKI VI Package Manager (VIPM)

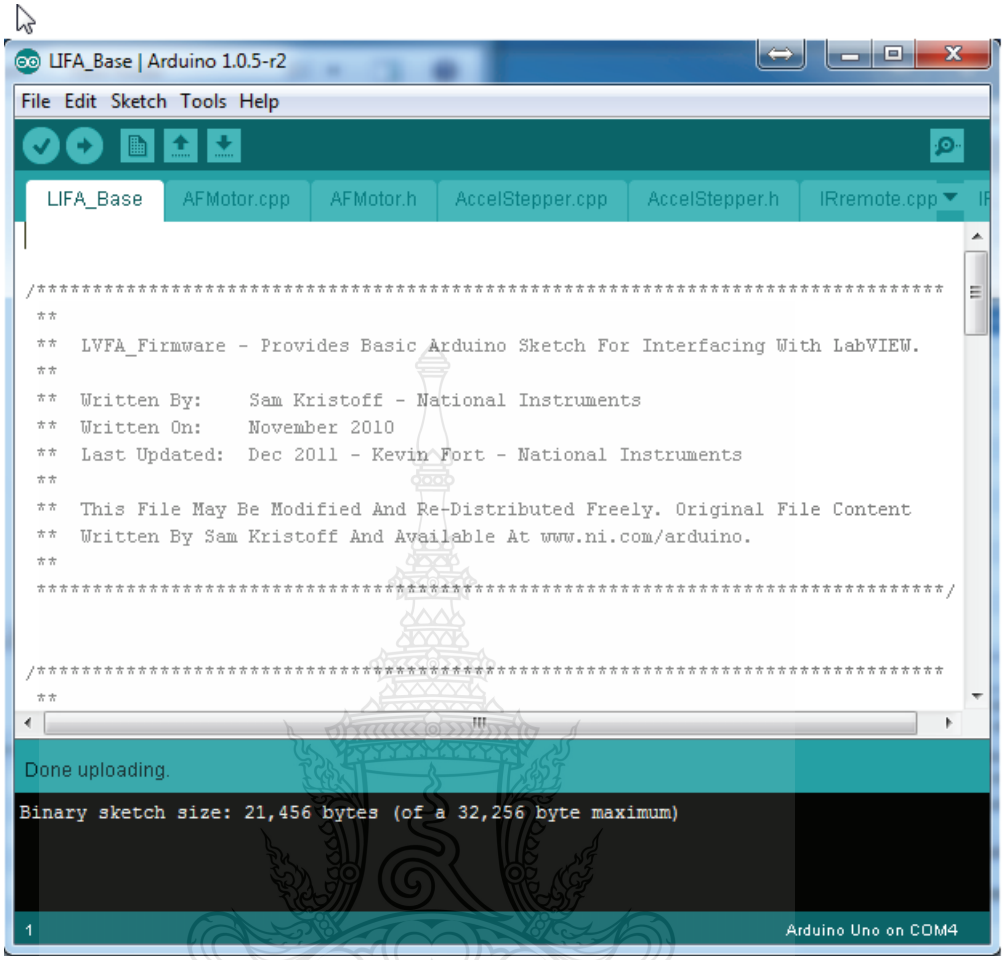



**Figure 2.65** LabVIEW Interface for Arduino (LIFA) Software

# CHAPTER3

# RESEARCH METHODOLOGY

Parallel robots are parallel structure robots. with the robot consisting of a robot base. and the forearm used to act as assigned and between the base and the end of the arm will be connected by a total of three sets of robotic arms. So, the whole structure consists of 11 arms, 12 spherical joints, and 3 rotating joints, all driven by 3 motors. independence equals three which corresponds to the number of the mechanical arm sets the nature of the robotic arm of the Parallel robot is connected in a symmetrical manner, that is, each robotic arm forms an angle of 120 degrees to each other with two ends. connected to the base and the same forearm, sometimes this structure can be called a closed structure robot and because of this making Parallel robots are more difficult and complex in many ways, for example, when moving one arm This affects the remaining two arms, making one movement of the robot necessary to supply power to all motors, or to control all motors.

## 3.1 Parallel Robot Structure

For the design of a Robot structure, the task starts with analysing the main structural dimensions of the Robot Parallel based on kinetic simulations to determine the robot's desired working area and trajectory using the MATLAB Simulink program. The structure of the robot is designed with the SOLIDWORKS program, including the design of the system and the selection of robot control devices. The analysis method is based on the factor data regarding the speed and accessibility of the desired workpiece. The parallel robot's physical structure is depicted in Figure 3.1, Figure 3.2 and Figure3.3

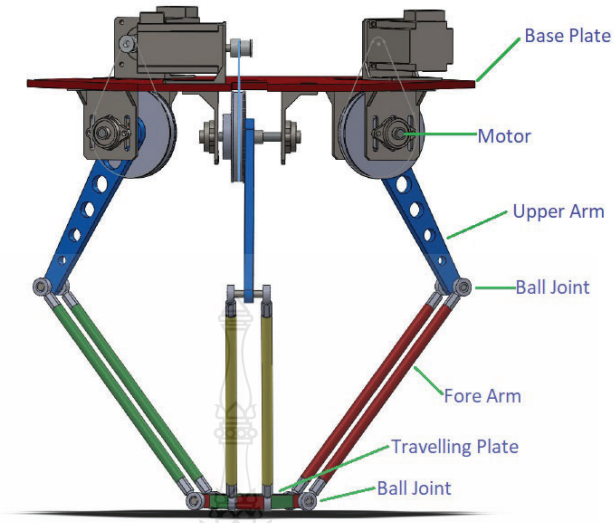
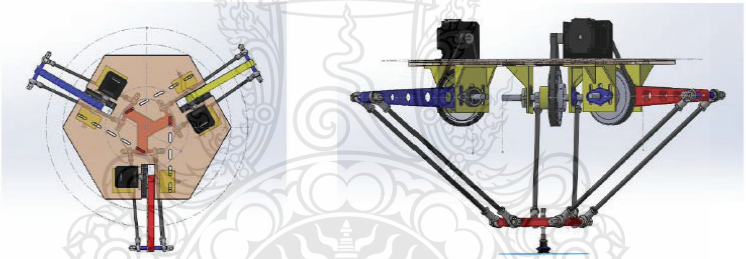



**Figure 3.1** The structure 3 DOF Parallel robot



(a) (b)

**Figure 3.2** Section view of Robot structure (a) Top View (b) assembly Section

(c) (d)

**Figure 3.3** Parallel Robot structure (c) Bottom View (d) Side View

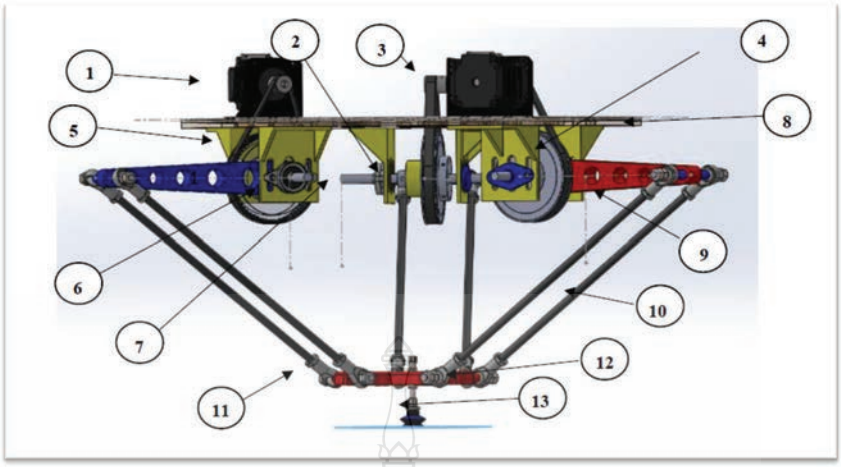



**Figure 3. 4 The Component of Parallel robot**

**Table 3.1** List of Parallel robot Component

| Item | Robots Component | Quantity |
|---|---|---|
| 1 | Stepper Motor include Encoder / Motor Driver | 3 |
| 2 | Coupling | 3 |
| 3 | Pulley XL 10T 8mm | 3 |
| 4 | Pulley XL 60T 10mm | 3 |
| 5 | Timing Belt XL 160mm | 3 |
| 6 | Bearing 10mm | 3 |
| 7 | Shaft 13mm | 3 |
| 8 | Base platform | 1 |
| 9 | Actuating arm | 3 |
| 10 | Passive arm | 3 |
| 11 | Ball Joint 8mm | 12 |
| 12 | Travelling platform | 1 |
| 13 | Vacuum Pad (Gripper) | 1 |

### 3.2 Parallel robot Arm parameters

Structural design of parallel robots Size and length of the main parts the parameters that will be used in the construction and assembly of the robot are very important because these parameters greatly affect the working area and trajectory of the robot if not properly presented. Will result in limitations that will make the robot unable to work in the area. [2,5,6,8,9] The position parameters of the main of the Parallel robot frame can be observed in Figure. 3.5

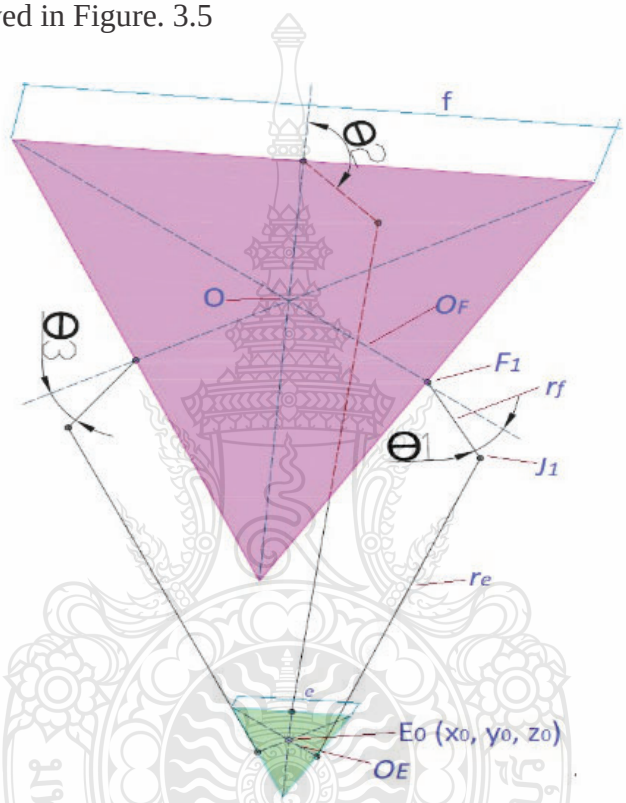



**Figure 3.5** Parallel robot mechanism diagram.

As mentioned in the mechanical structure section. The robot consists of four types of arms: base (OF), upper arm (rf), lower arm (re), and Side of the fixed triangle (f), which focus on length and weight, both of which affect the robot's performance. in the length section for the base and tip the sleeve is measured from the length of the side. and in terms of weight, Since the base is a stationary part, it makes the weight of the base has no effect on the robot in terms of performance. the total length dimension of the robot arm is shown in Table 3.2.

**Table 3.2** Parallel robot Manipulator Key component parameter and dimensions

| Component Name | Symbol | Value (mm) |
|---|---|---|
| Dimension of upper link | *rf* | 200 |
| Dimension of lower link | *re* | 500 |
| Dimension of the fixed triangle | *f* | 554.26 |
| Dimension of the end effector triangle | *e* | 259.81 |
| Dimension from center of base plate to center of motor | *OF* | 160 |
| Dimension from center of travelling plate to ball joint | *OE* | 75 |

### 3.3 Control System Design

Overview of the control system It has designed a vision system to detect the difference in position size and colour of the target workpiece sent on the conveyor belt with a vision camera detecting the position, size, and colour of the target workpiece to transmit the detected image to the NI LabVIEW Vision software for image processing for data acquisition and image processing after that, the NI Vision LabVIEW program will analyse the acquired images and send control signals to microcontroller No. 1, which is the main microcontroller board that will transmit the signal controlling the position of the robot to the microcontroller board number 2, 3 and 4 to control the stepper motor. It issues conditional control commands to the motor driver. The parallel robot is designed using stepper motors and motors. All 3 drivers work independently. Has installed a rotation sensor (Rotary Encoder) at each motor axis for detecting the rotation position of the motor and sending a detection signal to the Arduino microcontroller board No.1 to display signal effect the angle position of the robotic arm in each control condition. The control system is designed using a Vision Camera to communicate with the NI Vision program, computer, and microcontroller board to control the operation of the parallel robot and each System to communicate and send data to each other as in Figure 3.6

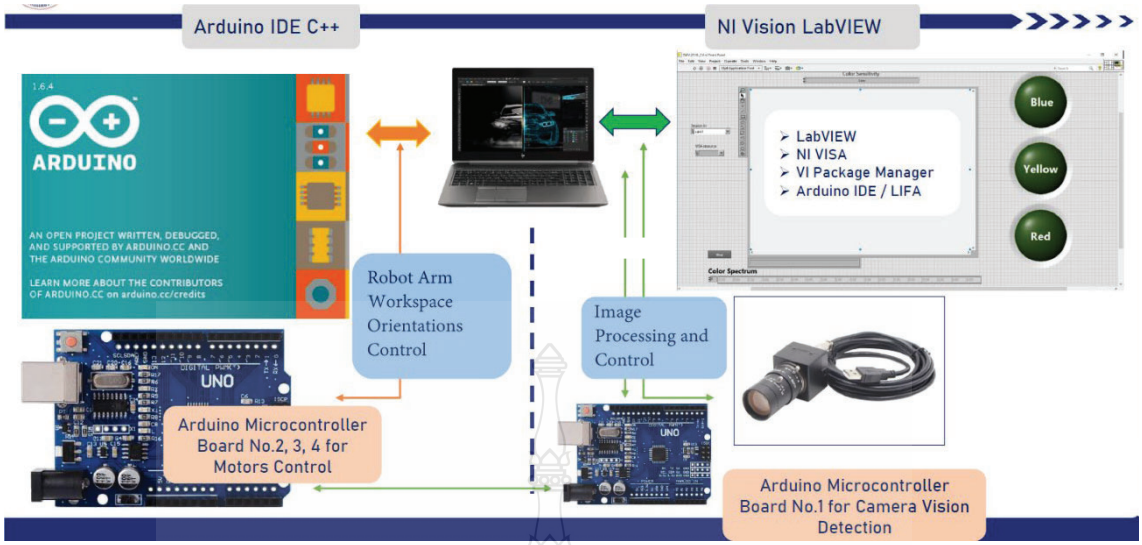



**Figure 3.6 Interface of control system architecture**

Operation process of the control system, System operation It will start by pressing the start position button switch to send a signal to microcontrollers No. 2, 3, and 4 to control the three robot arms to move to the starting position (Home Position) and send a control signal to the conveyor control system to starts the conveyor to transporting the workpiece through a vision camera system that detects the target's position, size, and colour. After that, the vision system sends the image detection signal to NI Vision LabVIEW to store the data and process the image. The NI Vision LabVIEW program analysed the acquired image signal and converted it into a digital control signal and sends the control signal to the microcontroller Board No.1 and the 1st microcontroller board will analyse the incoming digital signal to compare with the condition of the written program code. After that, it will send the digital output signal to the 2nd microcontroller Board No.3 and Board No.4 to stepper motor driver to control all of the motors to rotate the robot Link in specified direction conditions. and at the same time, of each motor Sensor detection rotation angle position (rotary encoder) sends the send the upper arm angle position to the microcontroller board No.1, which shows the position of the robot's upper arm angle on the Serial monitor program page from Arduino C++ IDE. The received signal can be compared with the dynamics simulation results of MATLAB simulation programs [24, 26] for further data analysis. The block diagram of the parallel

robot control system is shown in Figure 3.7, and Figure 3.8 shows the operation steps of the control process.

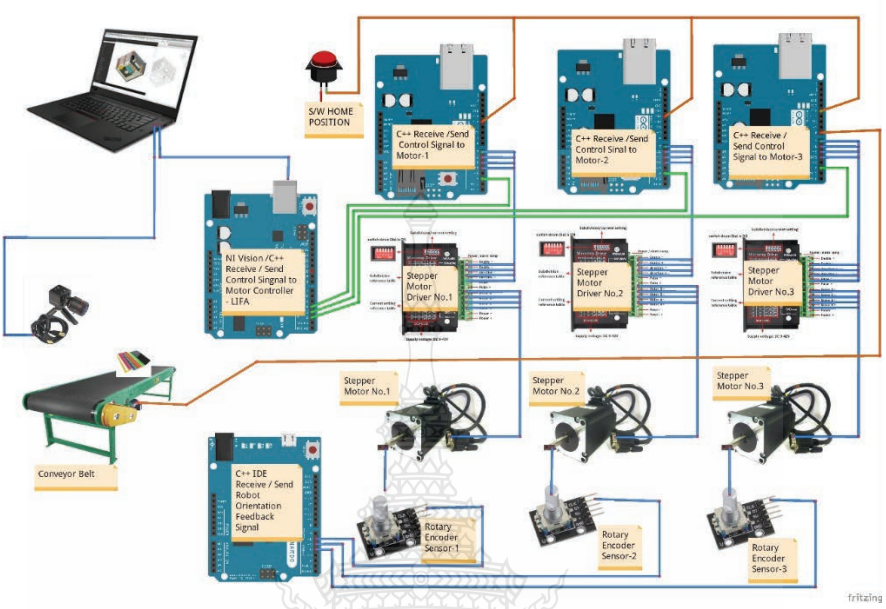



**Figure 3.7** Parallel Robot control system schematic

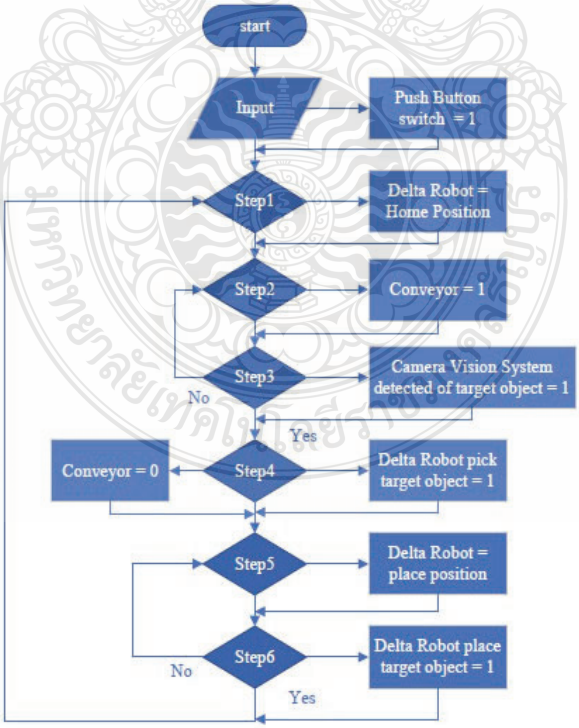



**Figure 3.8** Workflow of the Parallel Robot control process

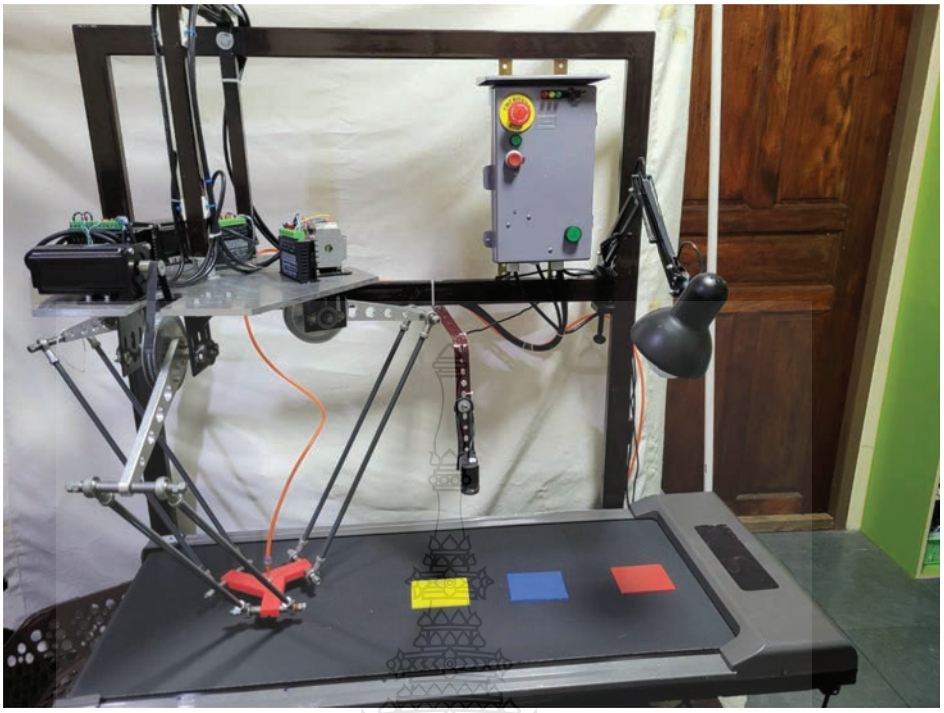


**Figure 3.9** A parallel robot mechanism connects all three motors with a microcontroller, camera view and delivery system.

### 3.4 Parallel Robot Kinematics

Overview of the control system It has designed a vision system to detect the difference in position size and colour of the target workpiece sent on the conveyor belt To control the movement of the robot's forearm. Need to rely on the mechanics of the robot, which is knowledge about mathematical equations. Because Delta robots have complex mechanics in past research, many researchers have attempted to propose different methods (19-23] in this research It will take the kinematic of robotics to find the geometry of the robot which is one of the popular methods and can be used to control the robot well. Used to solve the angle of the puppet's upper arm and forearm position as in Figure 4.1. These problems can be solved by forward kinetics and inverse kinetics which will be discussed further in this chapter. And also, in this chapter, also discusses the data of the robotic arm and robot workspace Model. The details will be described as follows.

3.4.1 Parallel Robot Forward Kinematic, Parallel Robot Forward Motion Equation A mathematical equation is used to calculate the angle of the upper arm. Is the position of the end effector of the robot when the joint angles are known. Since the three angles are θ1, θ2, and θ3 [19-23], the center position (*Xt, Yt, Zt*) of the moving plate is obtained by calculating the forward dynamics of the third axis [19-23]. Parallel robot First, the parameters of the robot can be defined. To calculate and set the lengths of the main parts of the parallel robot in mathematical equations Important variables must be configured namely the distance between the center of the base and the center of the motor (*OF*), the length of the upper link (*rf*), the distance of the lower link (*re*) and the center of the drive plate and the ball joint (*OE*). By this calculation Obtained from the overlapping of 3 spheres considering the shape of the robot. If the center of the sphere is at point J, then the radius of each sphere is equal to the length of the forearm. Then move the center of *J1*, *J*2, and *J3* to points *J1', J2',* and *J3'* in the 3D view. The length of the forearm at the origin of the three joint is expressed as

$$(x - x_j) + (y - Y_j) + (z - Z_j) = re^2 \rightarrow j = 1,2,3$$

$$OF_1 = OF_2 = OF_3 = \frac{f}{2}\tan(30°) = \frac{f}{2\sqrt{3}} \quad (1)$$

$$J1J1' = J1J1' = J1J1' = \frac{e}{2}\tan(30°) = \frac{e}{2\sqrt{3}} \quad (2)$$

$$F1J1 = rfcos(\theta_1) \quad (3)$$

$$F2J2 = rfcos(\theta_2) \quad (4)$$

$$F3J3 = rfcos(\theta_3) \quad (5)$$

$$J1' = \begin{bmatrix} 0 \\ \frac{-(f-e)}{2\sqrt{3}} - rf\cos(\theta_1) \\ -rf\sin(\theta_1) \end{bmatrix} = (x_1, y_1, z_1) \quad (6)$$

$$J2' = \begin{bmatrix} \left[\frac{f-e}{2\sqrt{3}} + rf\cos(\theta_2)\right]\cos(30°) \\ \left[\frac{f-e}{2\sqrt{3}} + rf\cos(\theta_2)\right]\cos(30°) \\ -rf\sin(\theta_2) \end{bmatrix} = (x_2, y_2, z_2) \quad (7)$$

$$J3' = \begin{bmatrix} \left[\frac{f-e}{2\sqrt{3}} + rf\cos(\theta_3)\right]\cos(30°) \\ \left[\frac{f-e}{2\sqrt{3}} + rf\cos(\theta_3)\right]\cos(30°) \\ -rf\sin(\theta_3) \end{bmatrix} = (x_3, y_3, z_3) \quad (8)$$

$$\begin{cases}(x-x_1)^2+(y-y_1)^2+(z-z_1)^2=re^2\\(x-x_2)^2+(y-y_2)^2+(z-z_2)^2=re^2\\(x-x_3)^2+(y-y_3)^2+(z-z_3)^2=re^2\end{cases} \quad (9)$$

$$\begin{cases}x^2+y^2+z^2-2y_1y-2z_1z=re^2-y_1^2-z_1^2 & [1]\\x^2+y^2+z^2-2x_2x-2y_2y-2z_2z=re^2-x_2^2-y_2^2-z_2^2 & [2]\\x^2+y^2+z^2-2x_3x-2y_3y-2z_3z=re^2-x_3^2-y_3^2-z_3^2 & [3]\end{cases}$$

(10)

$$w_i=\ x_i^2+y_i^2+z_i^2 \quad (11)$$

$$\begin{cases}x_2x+(y_1-y_2)y+(z_1-z_2)z=\frac{w_1-w_2}{2} \rightarrow [4\,]=\ [1]-[2]\\x_3x+(y_1-y_3)y+(z_1-z_3)z=\frac{w_1-w_3}{2} \rightarrow [5]=\ [1]-[3]\\(x_2-x_3)x+(y_2-y_3)y+(z_2-z_3)z=\frac{w_3-w_3}{2} \rightarrow [6]=\ [2]-[3]\end{cases} \quad (12)$$

From [4] – [5]

$$x=\ a_1z+b_1 \qquad [7] \quad (13)$$

$$a_1=\frac{1}{d}[(z_2{-}z_1)(y_3{-}y_1)-(z_3{-}z_2)(y_2{-}y_1)] \quad (14)$$

$$b_1=-\frac{1}{2d}[(w_2{-}w_1)(y_3{-}y_1)-(w_3{-}w_2)(y_2{-}y_1)] \quad (15)$$

$$d=\ (y_2-y_1\,)x_3-(y_3-y_1\,)x_2 \quad (16)$$

$$y=\ a_2z+b_2 \qquad [8] \quad (17)$$

$$a_2=\frac{-1}{d}[(z_2{-}z_1)(x_3)-(z_3{-}z_1)(x_2)] \quad (18)$$

$$b_2=-\frac{1}{2d}[(w_2{-}w_1)(x_3)-(w_3{-}w_2)(x_2)] \quad (19)$$

The latest solution and calculate in x0 and y0 from the equation [7] and [8]

The solution equation instead [7] and [8] in [1]

$$(a_1^2+a_2^2)z^2+2(a_1+a_2(b_2-y)-z_1)z+(b_1^2+(b_2-y_1)^2+z_1^2-re^2)=0 \quad (20)$$

3.4.2 Inverse Kinematic of Parallel Robot, Inverse kinematics: If the position of the end effector (*X, Y, Z*) is known [19-23], the common angles θ1, θ2, and θ3 must be determined. Need to calculate the inverse kinematics of a 3-axis parallel robot with delta. First, the robot parameters for the calculation of the mathematical equations must be determined. Then determine the side length of the upper triangle as *f*, the side length of the lower triangle as *e*, the length of the upper arm as *rf*, and the length of the lower arm as *re*.

$E = (x_{0,} y_{0,} z_0)$

(21)

$$EE_1 = \frac{e}{2}\tan(30°) = \frac{e}{2\sqrt{3}} \quad (22)$$

$$E_1E_1' = x_0 \begin{cases} E_1 = \left(x_0, y_0 - \frac{e}{2\sqrt{3}}, z_0\right) \\ E_1' = \left(0, y_0 - \frac{e}{2\sqrt{3}}, z_0\right) \end{cases} \quad (23)$$

$$F_1 = x_0, \frac{f}{2\sqrt{3}}, 0 \quad (24)$$

$$E_1'J_1 = \sqrt{(E_1J_1)^2 - (E_1E_1')^2} = \sqrt{re^2 - x_0^2} \quad (25)$$

$$\begin{cases} (y_{J1} - y_{F1})^2 + (z_{J1} - z_{F1})^2 = rf^2 \\ (y_{J1} - y_{E_1'})^2 + (z_{J1} - z_{E_1'})^2 = re^2 - x_0^2 \end{cases} \quad (26)$$

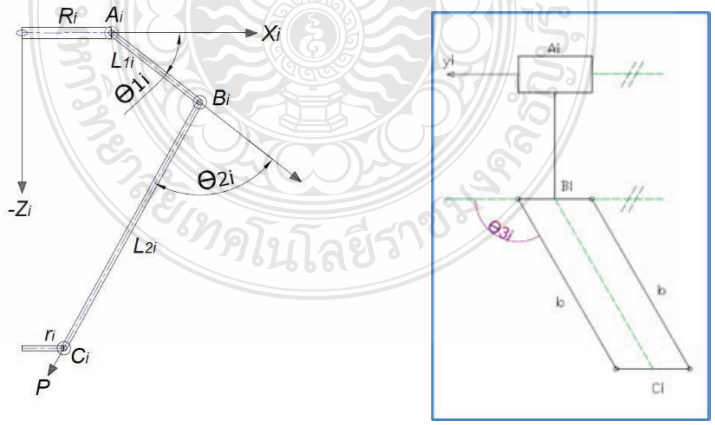
$$\begin{cases} (y_{J1} - \frac{f}{2\sqrt{3}})^2 + (z_{J1})^2 = rf^2 \\ (y_{J1} - y_0 + \frac{e}{2\sqrt{3}})^2 + (z_{J1} - z_0)^2 = re^2 - x_0^2 \end{cases} \quad (27)$$

$$J_1 = -(0, y_{J1}, z_{J11}) = f(rf, re, f, e, x_0, y_0, z_0) \quad (28)$$

$$\theta_1 = \tan^{-1}\left(\frac{z_{J1}}{y_{F1} - y_{J1}}\right) \quad (29)$$

$$\theta_2 y \theta_3 \rightarrow E_3(x_0, y_{0,} z_0) \rightarrow E_{0'}(x_0, y_{0'_,} z_{0'}) \quad (30)$$

$$\begin{cases} X_0' = (X_0 \cos(\pm 120°)) + (Y_0 \sin(\pm 120°)) \\ Y_0' = (X_0 \sin(\pm 120°)) + (Y_0 \cos(\pm 120°)) \\ Z_0' = Z_0 \end{cases} \quad (31)$$




(a) (b)

**Figure 3.10** The projection of the parallel robot kinematic

The loop that is most relevant to the desired Jacobian analysis should be chosen, which is a vector consisting of the excitatory common variable and the position vector of the moving plate. Then

$$\vec{\boldsymbol{\theta}} = \theta_{1i} = \begin{bmatrix} \theta_{11} \\ \theta_{12} \\ \theta_{13} \end{bmatrix}, \vec{\boldsymbol{p}} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} (4ac) \qquad (32)$$

**Table 3.3** Parallel robot workspace simulation configuration parameter

| ***Parameter Name*** | ***Symbol*** | ***Value (mm)*** |
|---|---|---|
| Upper link | *rf* | 200 |
| lower link | *re* | 500 |
| Distance between base center and motor center | *OF* | 160 |
| Distance from running board center to ball joint | *OE* | 75 |

### 3.5 MATLAB Program for Parallel Robotic Kinematic Simulation

To write a program to simulate the orientation and trajectory taking of the Parallel robot by specifying and referring to the size parameter of the parallel robot arm by using the MATLAB M-file function and GUI program by writing the program code that is shown in APPENDIX A.

### 3.6 NI Vision LabVIEW Program for Target Object Color Detection and interface with Arduino microcontroller Board

For programming to detect the difference in color and size of the target object using the LabVIEW NI vision Program and the communication interface to control the

operation of the robot using the LabVIEW interface with Arduino (LIFA) program by writing code. Program shown in APPENDIX B.

### 3.7 Arduino IDE C++ Program for Robotic Kinematic Detection

For writing programs to read and display the Robot arm angle position signals. of the parallel robotic arm using the Arduino IDE C++ program by writing the program code as shown in APPENDIX C.

# CHAPTER4

# RESEARCH RESULT

## 4.1 Parallel Workspace Analysis and Simulation of Manipulator based on MATLAB

Robot kinematics simulation in order to know the range of robot motion and work area suitable for designing a prototype parallel robot with MATLAB Simulink program [14-16]. Programming first needs to know the length of the mechanical arm of the parallel robot, the distance from the centre of the base to the centre of the motor (*OF*), the length of the upper joint (*rf*), the length of the lower joint (*re*). And the distance between the centre of the moving plate and the ball joint (*OE*) distance. As shown in Table 3.4 and Figure 3.5, there are main component parameters related to the movement of the manipulator.

First section, simulation programming using the inverse kinematic theory is carried out in four scenarios to prove the limitations of the motion of the parallel robot in each scenario. For [OF(Lr), rf(La), re(Lb), OE(Lh) = [160, 200, 500, 75] (mm)

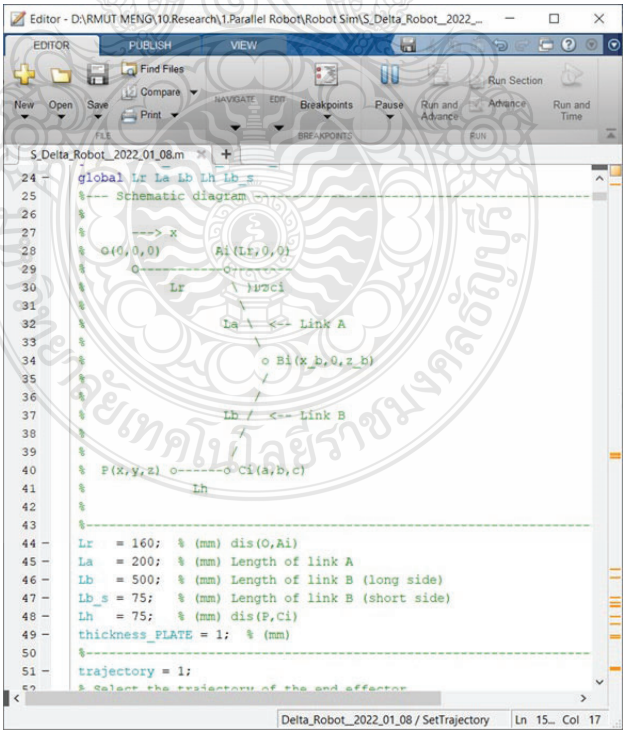



**Figure 4.1** The Parallel-Robot. Manipulator simulation parameters

4.1.1 Scenario 1, Workspace simulation and analysis for z1, Simulate the movement of the robot, and determine the limit of the movement range of the manipulator at the deepest point z1 by setting the maximum movement distance of the end effector of the manipulator in the z1 axis to a distance greater than the maximum distance from the kinematic calculations results. of the robot with parameter values of the robot arm obtained from the kinematic calculation [z1 = -200, z2 = 150, R = 250] by determining the simulation parameters of the robot arm [z1 = -210, z2 = 150, R = 250,].

From Figure 4.4 In the 3D model, it can see while setting z1 = -210 mm as a parameter for the simulation program that is a value greater than from the inverse kinematic theory, calculated at = -200 mm, use the RUN command to run in the MATLAB simulation program in front of the simulation program MATLAB will display the simulation model of exercise robotic arm Robot End-Effector in the z1 axis from the -411 mm (Home Position) position and the 3D Model program will display the message “Out of workspace!” at the -615 mm position. Thus, in Fig. 4.3, by comparing the position of the graph from the beginning of the Robot End-Effector in the z1 axis to the end at the point where the text indicates the problem point, the maximum displacement in the z1 axis is known (615mm. – 411mm. = 204 mm).  shows that the trajectory and working area of the robot obtained in the simulation are within the region obtained from the results calculated from the inverse kinematics. As shown in Table 4.1.

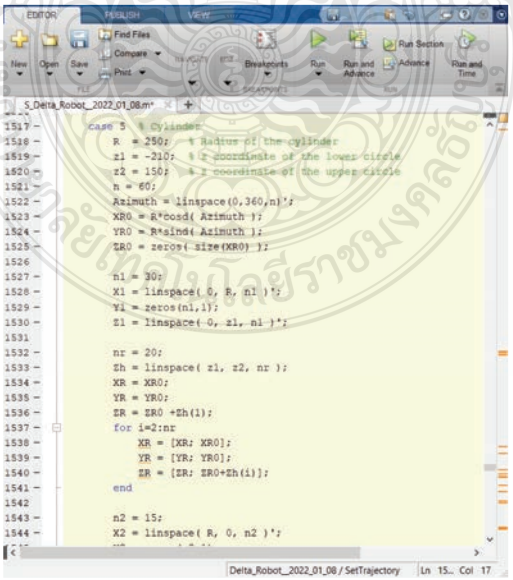



**Figure 4.2** Scenario1, Setup parameter for z1 simulation and analyzed for [z1, z2, R]

**Table 4.1** Simulation result in limit workspace point for Scenario 2,

| **Upper Arm Link Name** | **Limit of Workspace point** | |
|---|---|---|
| | End-Effector | Length from HP |
| ***x*,** | 241.5 | 241.5 |
| ***y*,** | 0 | 0 |
| ***z*$_1$,** | -615 | 204 |
| **Upper Arm Joint Name** | Joint Angles (degrees) | |
| ***Θ*$_1$** | 34.8° | |
| ***Θ*$_2$** | 95.8° | |
| ***Θ*$_3$** | 95.8° | |

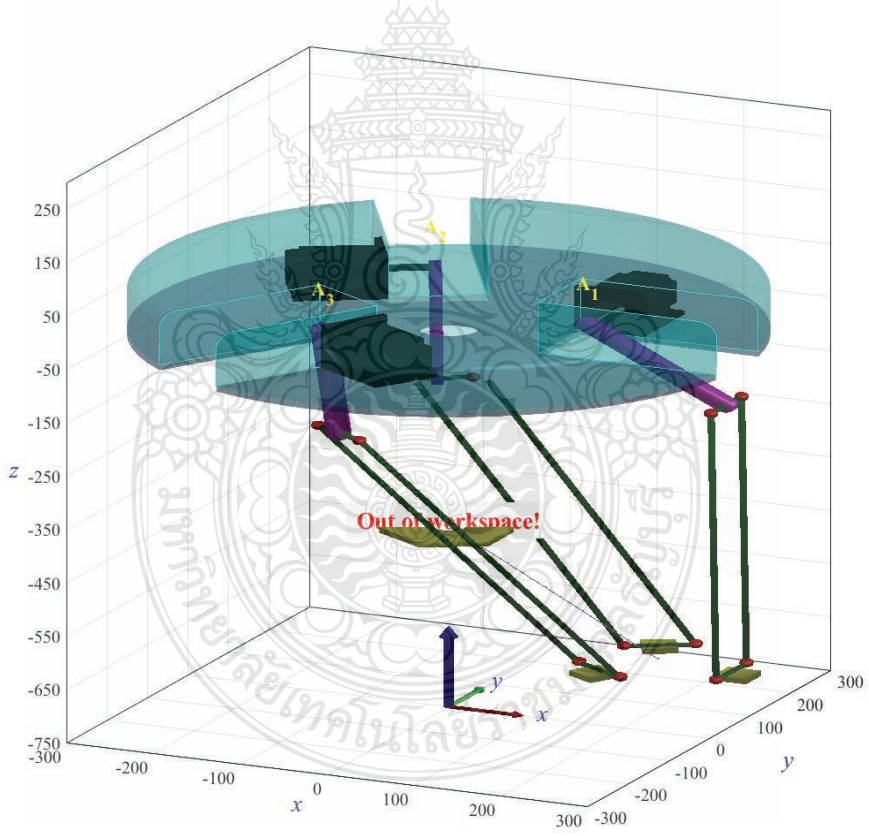



**Figure 4.3** Simulation 3D model of Maximize of lower-level workspace of z1, in Scenario1**,**

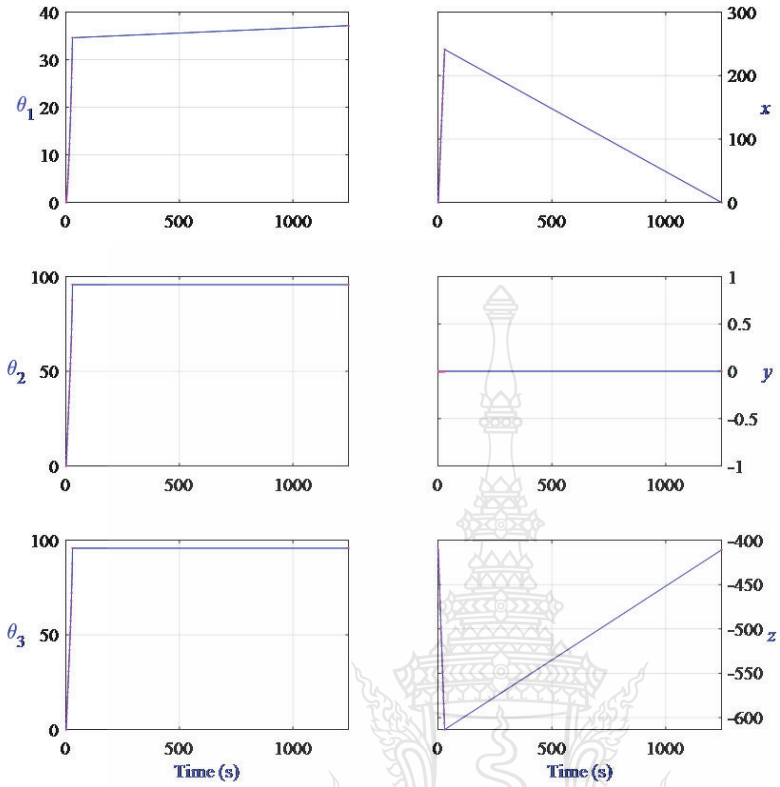



**Figure 4.4** Simulation of Maximize of trajectory tracking workspace graphs of z1, in Scenario1,

4.1.2 Scenario 2, Workspace simulation and analysis for z2, the robot's motion simulation was determining of the robot's arm motion limitation area at the upper-level point z2 by setting the movement range of the robot links at end-effector Up to a distance greater than the maximum distance in the z2 axis from the invert kinematic calculations results. of the robot with parameter values of the robot arm obtained from the invert kinematic calculation [z1= -200, z2 = 150, R = 250] by determining the simulation parameters of the robot arm [z1 = -200, z2 = 160, R = 250,].

From Figure 4.5 of In the 3D model, it can see while setting z2 = 160 mm as a parameter for the simulation program that is a value greater than from the inverse kinematic theory, calculated at = 150 mm and use the RUN command to run in the MATLAB simulation program in front of the MATLAB simulation program will display the simulation model of the movement of the robotic arm Robot End-Effector in the z2

axis from the -411 mm (Home Position) position and the 3D Model program will display the message “Out of workspace!” at the -252 mm position. Thus, in Fig. 4.4, by comparing the position of the graph from the beginning of the Robot End-Effector in the z2 axis to the end at the point where the text indicates the problem point, the maximum displacement in the z2 axis is known (411mm. – 252mm. = 159 mm).  shows that the trajectory and working area of the robot obtained in the simulation are within the region obtained from the results calculated from the inverse kinematics. As shown in Table 4.2.

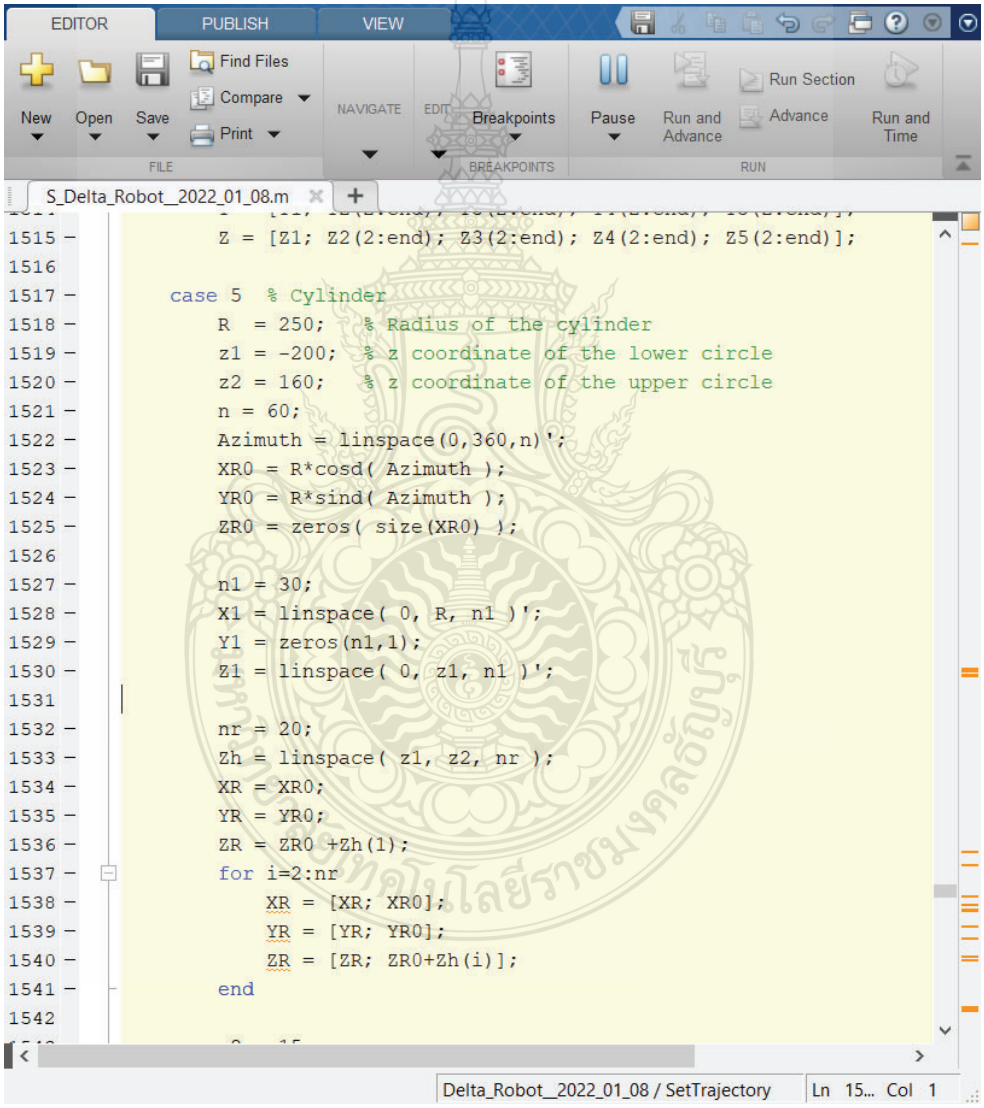



**Figure 4.5** Scenario2, Setup parameter for z2 simulation and analyzed for [z1, z2, R]

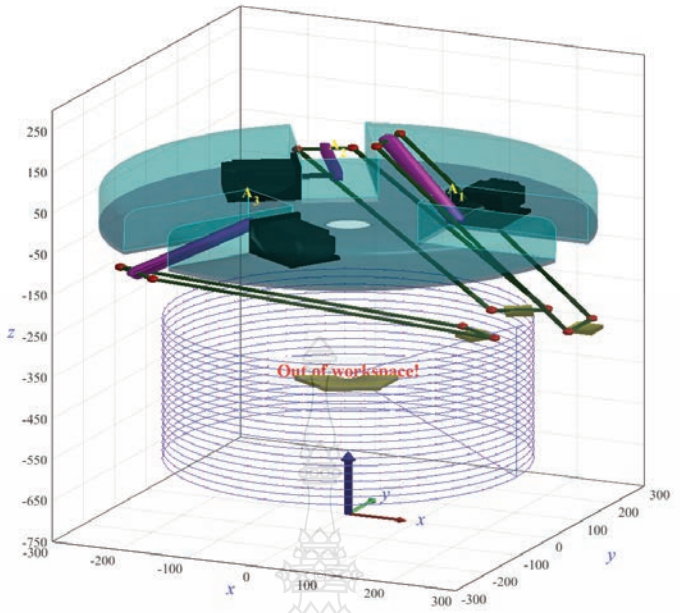



**Figure 4.6 Simulation 3D model of Maximize of Upper-level workspace of z2, in Scenario2**

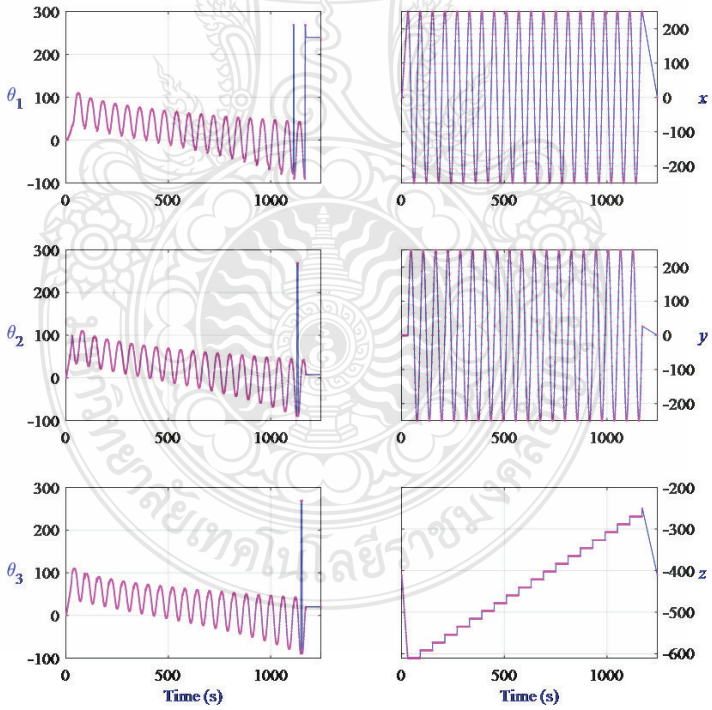



**Figure 4.7** Simulation model of Maximize of trajectory tracking workspace graphs of z2, in Scenario2,

**Table 4.2** Simulation result in limit workspace point for Scenario 2,

| **Upper Arm Link Name** | **Limit of Workspace point** | |
|---|---|---|
| | **End-Effector** | **Length from HP** |
| ***x,*** | 250 | 250 |
| ***y,*** | 27 | 27 |
| $Z_2$, | -252 | 159 |
| **Upper Arm Joint Name** | **Joint Angles (degrees)** | |
| $\Theta_1$ | -90° | |
| $\Theta_2$ | 8° | |
| $\Theta_3$ | 20° | |

4.1.3 Scenario 3, Workspace simulation and analysis for R (Radius of x-axis, and y-axis), To simulate the motion of the robot, divide the moving distance of the end effector of the robot arm by the limit of the range of motion of the robot arm at the cylinder point R (the limit radius of the working range of the X-axis and Y-axis). The distance between the axis and the Y-axis is greater than the kinematic calculation the maximum distance of the result. By determining the simulation parameters of the robot arm [z1 = -200, z2 = 150, z1 = -200, z2 = 150, R = 260]

From Figure 4.8 of the simulation in 3D model, it can be seen that when setting R = 260mm as a parameter of the simulation program, a value is greater than the value of the inverse dynamics theory calculated at = 250mm, and when using the command RUN to leave the simulation program running the MATLAB simulation program MATLAB shows the simulation model and 3D model of the robot arm "Robot End-Effector" moving in the x-axis and y-axis from the position of 0mm (initial position) and the 3D model program displays the message "Out of work area!". At 250mm position. Therefore, in Figure 4.9, by comparing the position of the robot end effector from the start of the R radius on the x-axis and y-axis to the end point of the text indicating the problem point, the maximum displacement in the x-axis is known (250 mm - 0 mm = 250mm).

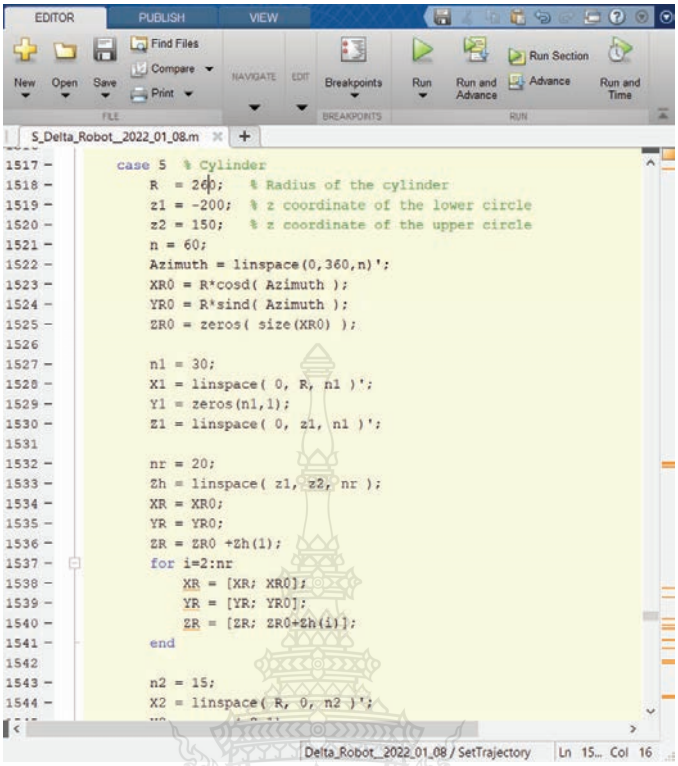



**Figure 4.8** Scenario3, Setup parameter for R simulation and analyzed for [z1, z2, R]

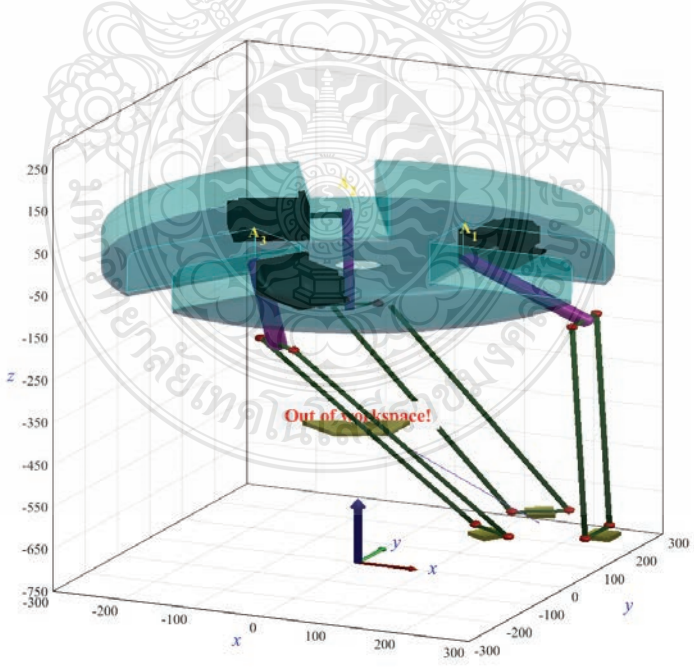



**Figure 4.9** Simulation 3D model of Maximize of radius workspace of x-axis, and y-axis, in Scenario3

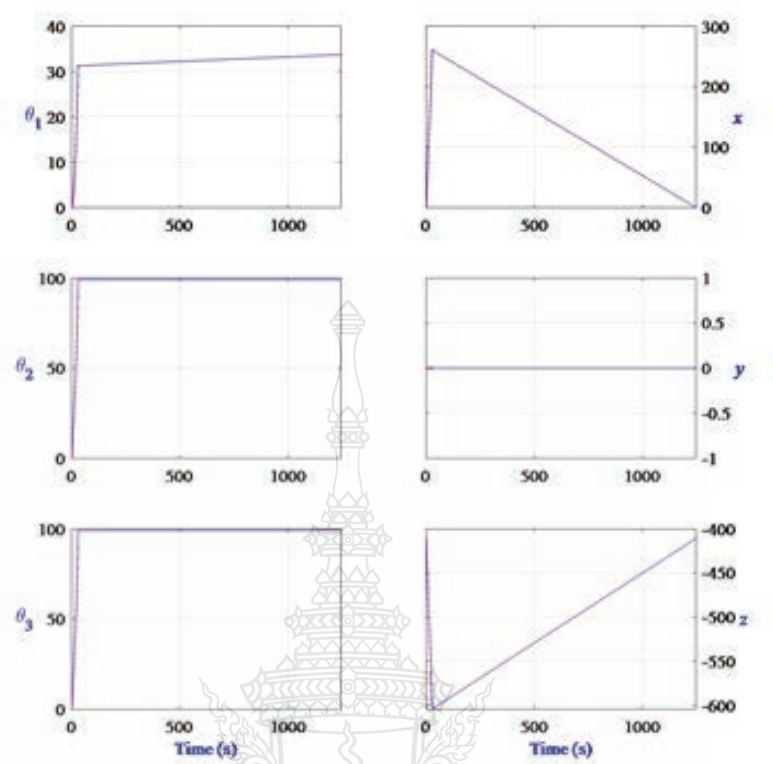



**Figure 4.10** Simulation of Maximize of trajectory tracking workspace graphs of x-axis, and y-axis, in Scenario3,

**Table 4.3** Scenario 3, limit workspace simulation

| **Upper Arm Link Name** | **Limit of Workspace point** | |
|---|---|---|
| | **End-Effector** | **Distance from HP** |
| ***x,*** | 250 | 250 |
| ***y,*** | 0 | 0 |
| ***$Z_2$,*** | -615 | 204, |
| **Upper Arm Joint Name** | **Joint Angles (degrees)** | |
| ***$\Theta_1$*** | 31.3° | |
| ***$\Theta_2$*** | 94° | |
| ***$\Theta_3$*** | 94° | |

4.1.4 Scenario 4, inverse kinematics simulation for x-axis, y-axis and z-axis, Simulate the movement of the robot, by setting the movement range limit of the robot arm at the cylinder point R (the limit radius of the X-axis, Y-axis and Z-axis work area), and set the movement distance as the end effector of the mechanical arm on the x-axis and y-axis The distance with the z-axis is greater than the maximum distance of the kinematic calculation results. By determining the simulation parameters of the robot arm [z1 = -200, z2 = 150, z1 = -200, z2 = 150, R = 250]

It can be seen from Figure 4.12 of the 3D model that when setting z1=-200mm, z2=150mm, R=250mm, that is, the calculated kinematics value, use the RUN command, MATLAB enables the simulation program to run. During the process, it simulates the movement of the mechanical arm and can work until the end of the process. From the trajectory tracking diagram in Fig. 22, the parallel robot kinematics simulation based on MATLAB Simulink can understand the spatially constrained shape of the parallel robot prototype in all workspace dimensions and summarize the values according to Table 4.4.

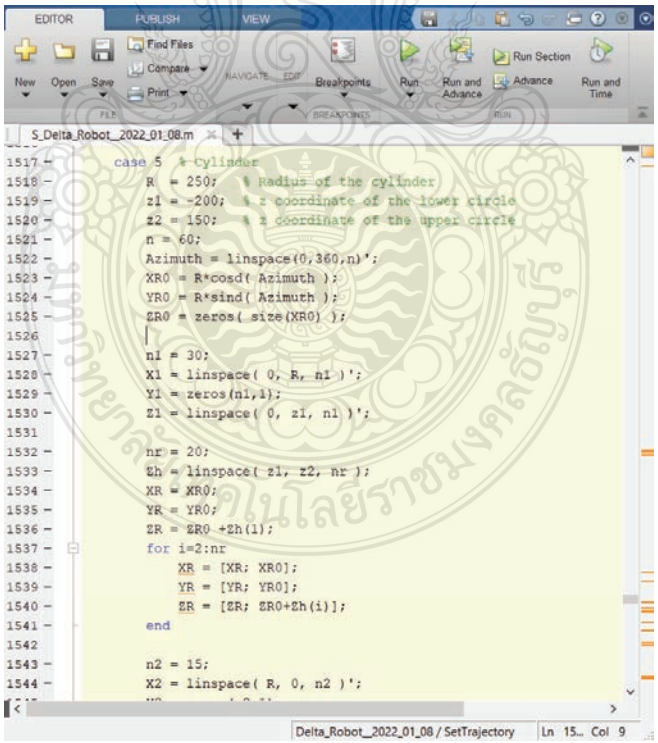



**Figure 4.11** Scenario4, Setup parameter for x-axis, y-axis and z-axis simulation and analyzed for [z1, z2, R] = [-200, 150, 250] (mm)

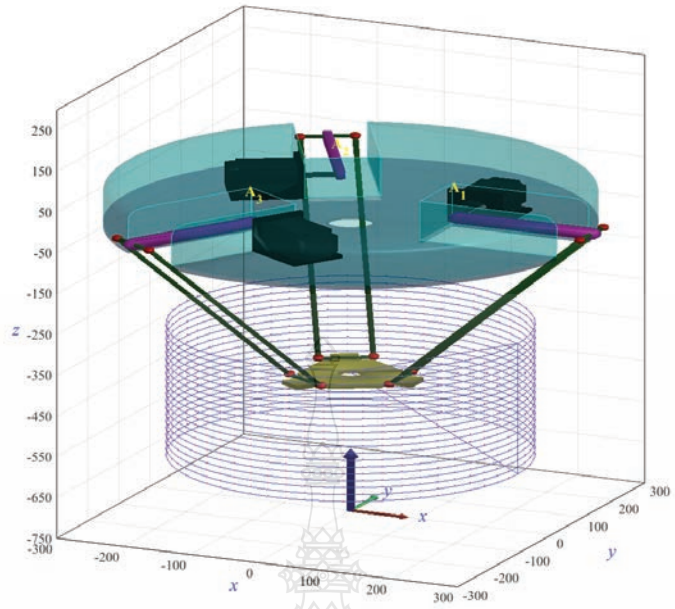



**Figure 4.12** Simulation 3D model of Maximize of radius workspace of x-axis, y-axis, and z-axis, in Scenario4 for [z1, z2, R] = [-200, 150, 250] (mm)

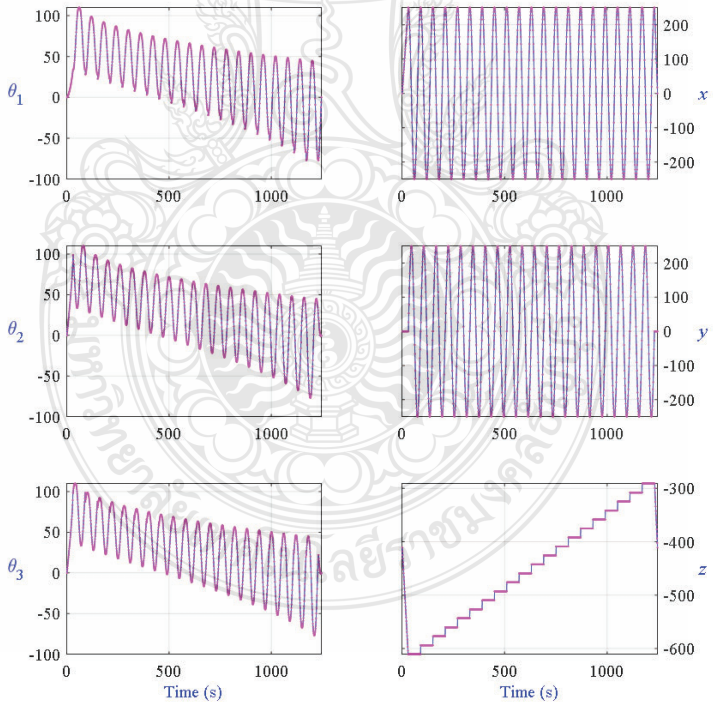



**Figure 4.13** Simulation model of Maximize workspace of x x-axis, y-axis and z-axis, in Scenario4

**Table 4.4** Scenario 4, limit workspace simulation result

| | **Value (mm) from Home position** | |
|---|---|---|
| **Upper Arm Link Name** | Min. | Max. |
| ***x,*** | -250 | 250 |
| ***y,*** | -250 | 250 |
| ***Z,*** | -200 | 150 |
| **Parameter Name** | Joint Angles (degrees) Robotic Arm orientation | |
| $\Theta_1$ | 110° | -85° |
| $\Theta_2$ | 110° | -85° |
| $\Theta_3$ | 110° | -85° |

### 4.2 Workspace and Trajectory Tracking Experiment of Prototype Parallel Robot

From determining the working range of the experimental parallel robot prototype to reaching the optimal working range. The structural size of the prototype parallel robot is determined with reference to the ratio obtained by MATLAB program simulation. Therefore, after designing and assembling the prototype parallel robot, a parallel robot experiment was carried out, using the principle of inverse kinematics to test the results of the joint angular motion position of the manipulator, to understand the kinematic characteristics of the manipulator at various positions and to detect motion. For the movement of the robot, install joint angle sensors (rotary encoders) at the ends of the three motor shafts to measure the joint angle positions (θ1, θ2, θ3) of the motors, and transmit the measured robot joint angle position feedback signals to any state under the arm to the microcontroller. The result of the angular position of the robotic arm can be displayed on the program page of the Arduino serial IDE monitor. The program page can graphically compare the angle values θ1, θ2 and θ3 using the real-time signals received from the sensors and the angle values of the robot with the reference signal according to the experimental conditions. In this experiment, according to the experimental procedure shown in Figure 23, the real-time signal data obtained from the serial monitor was plotted in a Microsoft Excel program to clearly see the signal pattern.

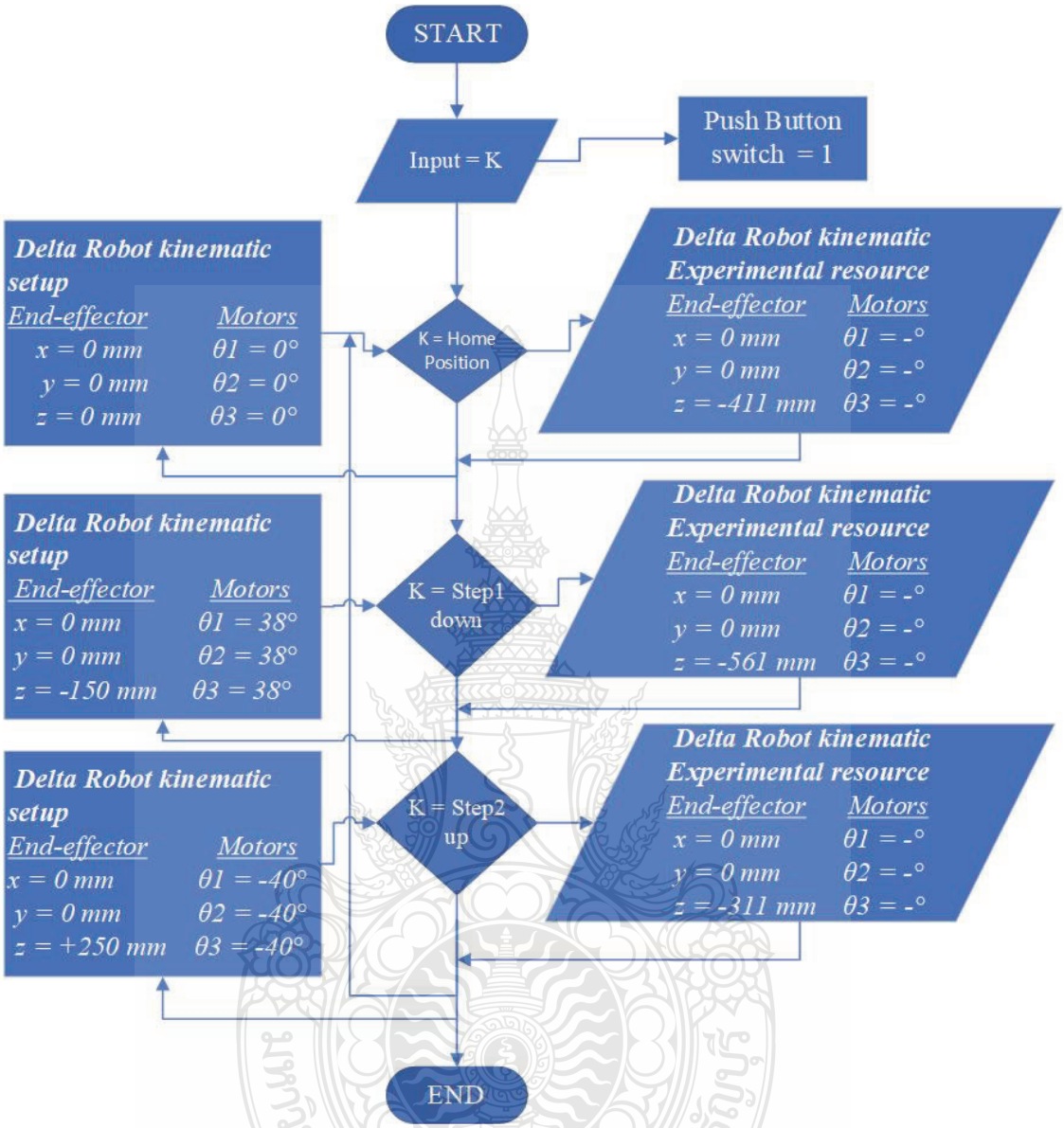



**Figure 4.14** Experimental process step of Parallel Robot

The experimental motion sequence of the prototype parallel robot uses the principle of inverse kinematics, and follows the experimental flow shown in Figure 4.14 to check the efficiency and results of the arm joint angles of the prototype parallel robot. System The joint angular position of the robot arm is moved to the home position (home position) and moved to position 1. This has positioned the end of the robot arm (end effector) to travel down the Z [x] axis = 0 mm, Y = 0 mm and Z = -150 mm]. The angles of the robot arm are set to the positions [θ1=38, θ2=38, and θ3=38] obtained by the kinematics simulation with the MATLAB Simulink program.

The position of the robotic arm (end effector) moves to the specified position according to the conditions of step 1. This step is an experiment to obtain the signal result of the angle detection of the connected encoder sensor. Placed on the motor shaft as a feedback signal for comparison with the angular position pattern of the robot arm determined by simulation. Then set the arm to move to position 2. The robotic arm (end effector) is positioned to move on the z-axis [x-axis = 0mm, y-axis = 0mm, z-axis = +250mm]. Adjust the angle of the mechanical arm to the position of [θ1=-40, θ2=-40, θ3=-40], the same as step 1. This step is the result of the experiment. The signal obtained by the angle detection of the sensor installed on the motor shaft is used as the feedback signal. It is used for comparison with the robot arm position pattern obtained from a kinematics simulation using the MATLAB-Simulink program. The total number of experiments is 5 rounds, as shown in Table 4.22, and the position of the robot arm and the angle of the robot arm are determined. The motors for each step are determined through kinematic simulations using a MATLAB Simulink program in Figure 4.15 and Figure 4.16.

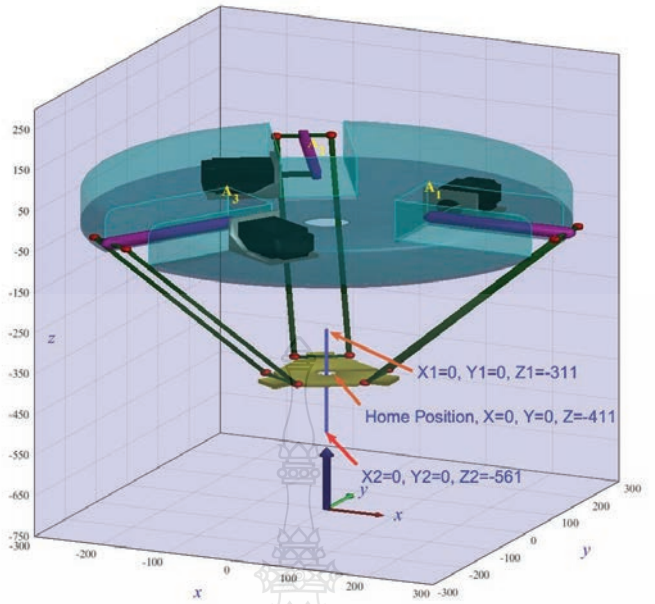



**Figure 4.15** The simulation results 3D Model for [ x, y, z ] = [0, 0, -411], [0, 0, -561], [0, 0, -311]

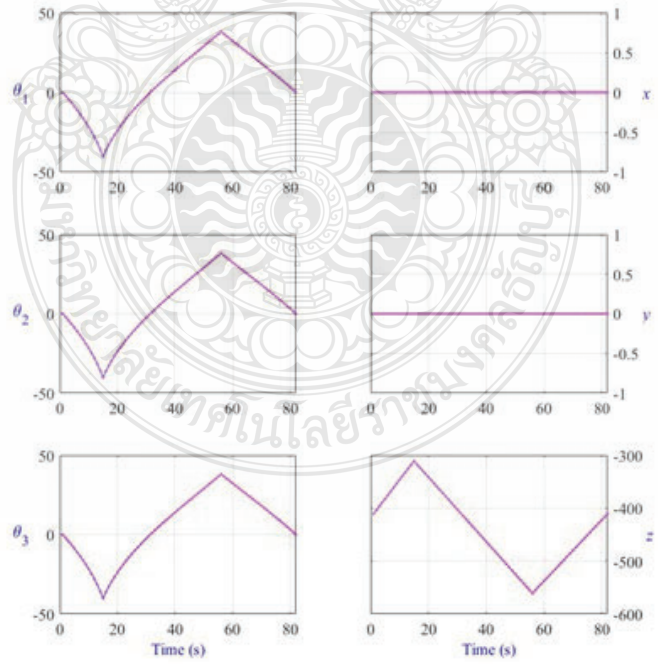



**Figure 4.16** Simulation model of Trajectory Tracking results graphs for [ x, y, z ] = [ 0, 0, -411 ], [ 0, 0, -561 ], [ 0, 0, -311 ]

**Table 4.5** Parallel Robot Experiment for joint angle orientation

| Experiment Step | Θ1 Desired Angle (Degree) | Θ2 Desired Angle (Degree) | Θ3 Desired Angle (Degree) |
|---|---|---|---|
| **Home Position** | 0º | 0º | 0º |
| **Step 1** | -40º | -40º | -40º |
| **Step 2** | 38º | 38º | 38º |
| **Step 3** | -40º | -40º | -40º |
| **Step 4** | 38º | 38º | 38º |
| **Step 5** | -40º | -40º | -40º |
| **Step 6** | 38º | 38º | 38º |
| **Step 7** | -40º | -40º | -40º |
| **Step 8** | 38º | 38º | 38º |
| **Step 9** | -40º | -40º | -40º |
| **Step 10** | 38º | 38º | 38º |
| **Home Position** | 0º | 0º | 0º |

**Table 4.6** Parallel Robot joint angle orientation results

| Experiment Step | Θ1 Detection Angle (Degree) | Θ2 Detection Angle (Degree) | Θ3 Detection Angle (Degree) |
|---|---|---|---|
| **Home Position** | 0º | 0º | 0º |
| **Step 1** | -39º | -38º | -39º |
| **Step 2** | 40º | 42º | 42º |
| **Step 3** | -39º | -39º | -38º |
| **Step 4** | 42º | 41º | 42º |
| **Step 5** | -39º | -38º | -39º |

| | | | |
|---|---|---|---|
| **Step 6** | 40º | 41º | 42º |
| **Step 7** | -39º | -39º | -39º |
| **Step 8** | 42º | 42º | 41º |
| **Step 9** | -38º | -38º | -39º |
| **Step 10** | 41º | 40º | 41º |
| **Home Position** | 0º | 0º | 0º |

**Table 4.7** Parallel Robot error signal of joint angle orientation results

| **Experiment Step** | **Θ1 Detection Angle Error (Degree)** | **Θ2 Detection Angle Error (Degree)** | **Θ3 Detection Angle Error (Degree)** |
|---|---|---|---|
| **Home position** | 0º | 0º | 0º |
| **Step-1** | 1º | 2º | 1º |
| **Step-2** | 2º | 4º | 4º |
| **Step-3** | 1º | 1º | 2º |
| **Step-4** | 4º | 3º | 4º |
| **Step-5** | 1º | 2º | 1º |
| **Step-6** | 2º | 3º | 4º |
| **Step-7** | 1º | 1º | 1º |
| **Step-8** | 4º | 4º | 3º |
| **Step-9** | 2º | 2º | 1º |
| **Step-10** | 3º | 2º | 3º |
| **HP** | 0º | 0º | 0º |

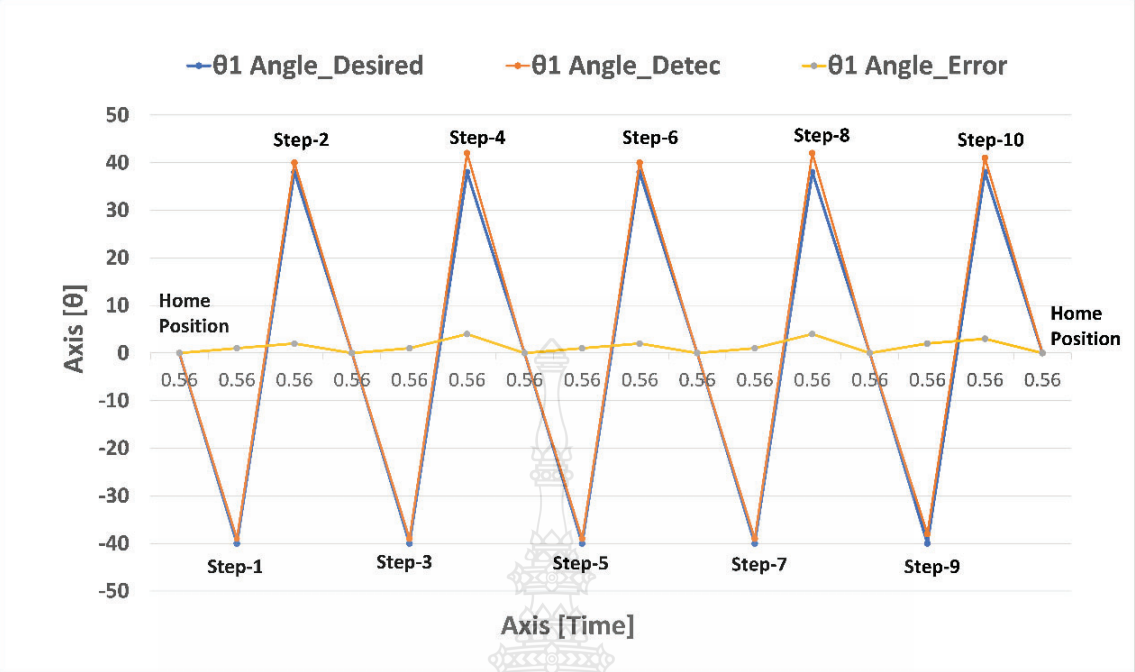



**Figure 4.17** Trajectory Tracking experiment results of θ1

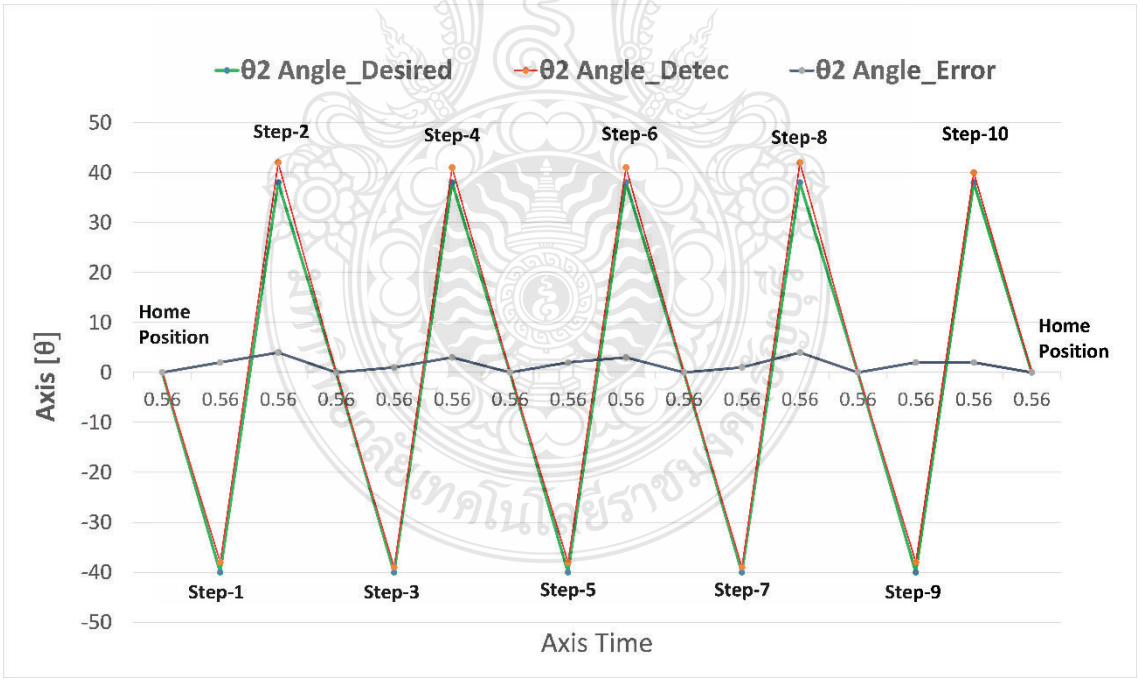



**Figure 4.18 Trajectory Tracking experiment results of θ2**

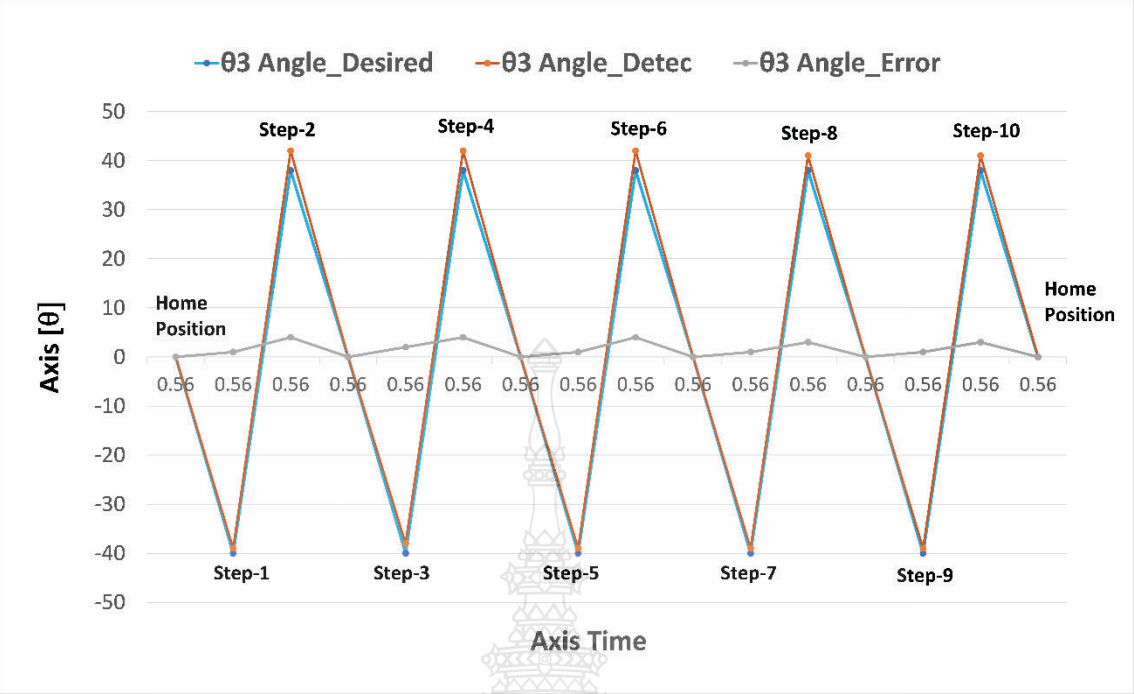



**Figure 4.19** Trajectory Tracking experiment results of θ3

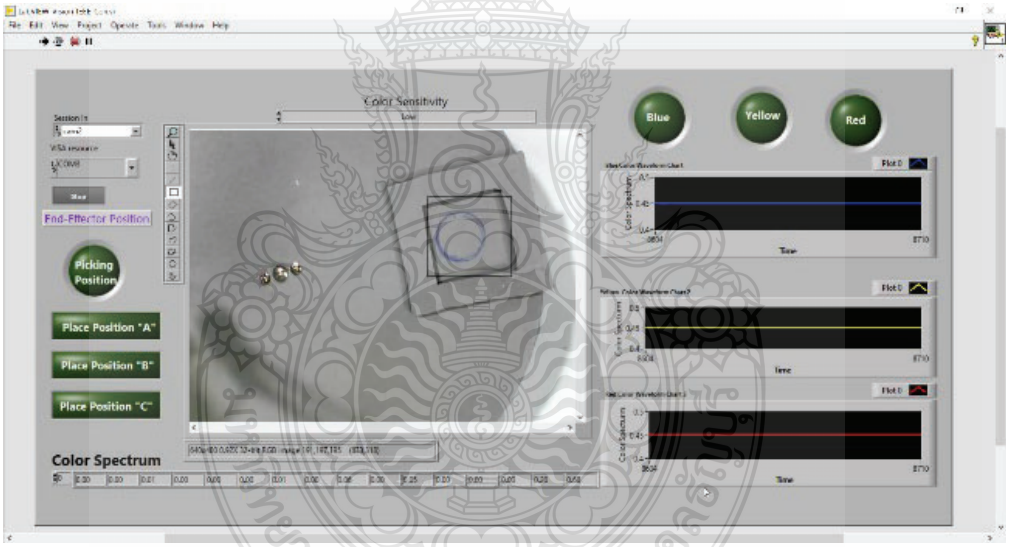
### 4.3 Experimental Results from LabVIEW Vision Control

**Table 4.8** LabVIEW Vision Experiment and Result for Color Matching Processing

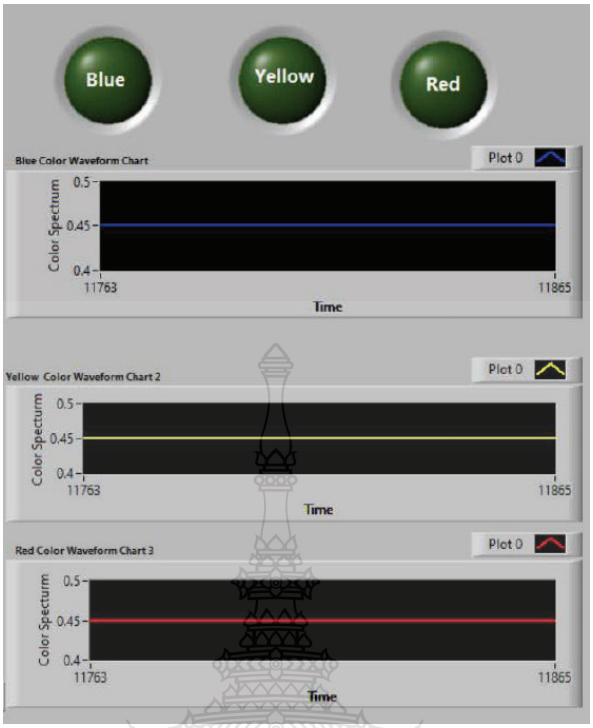
| Object | Blue Color | Yellow Color | Red Color |
|---|---|---|---|
| **Experiment Number.** | 30 | 30 | 30 |
| **Result Number.** | 30 | 30 | 30 |
| **Accuracy** | 100% | 100% | 100% |

**Table 4.9** LabVIEW Vision Experiment and Result for Color Gain

| Object | Blue Color | | Yellow Color | | Red Color | |
|---|---|---|---|---|---|---|
| | Min | Max | Min | Max | Min | Max |
| **Experiment Setpoint** | 0.45 | 1.2 | 0.45 | 1.2 | 0.45 | 1.2 |
| **Result** | 0.750 | 0.754 | 0.770 | 0.790 | 0.760 | 0.763 |
| **Accuracy** | within the specified limits | | within the specified limits | | within the specified limits | |

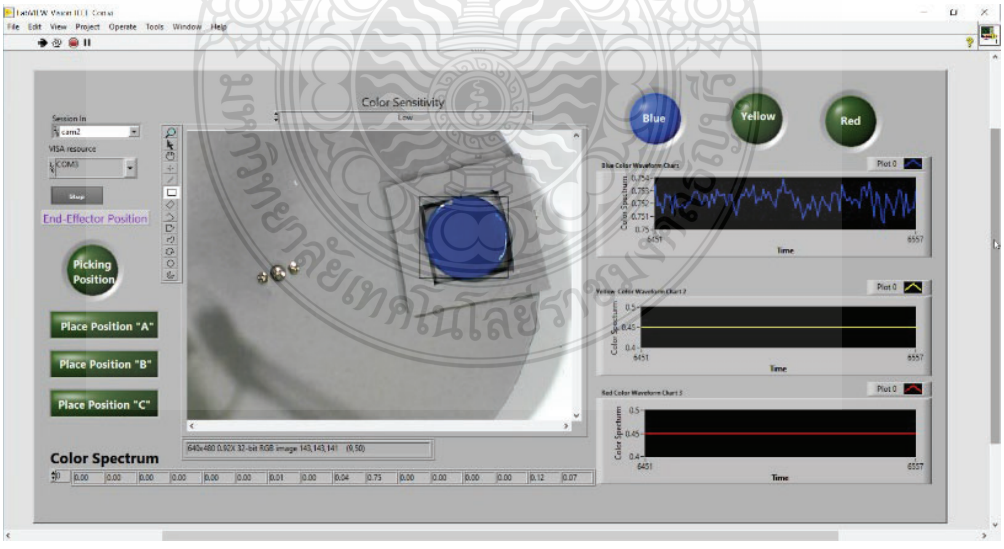



(a) LabVIEW Vision Color processing Front Panel

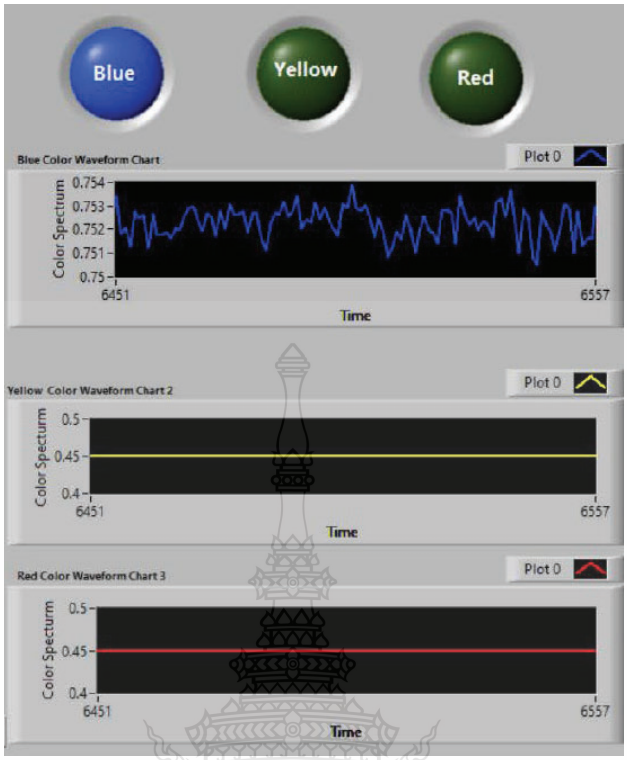



(b) Front Panel Object color matching and gain detection

**Figure 4.20 (**a), (b) Front Panel of LabVIEW Vision show Object color matching and Gain detection

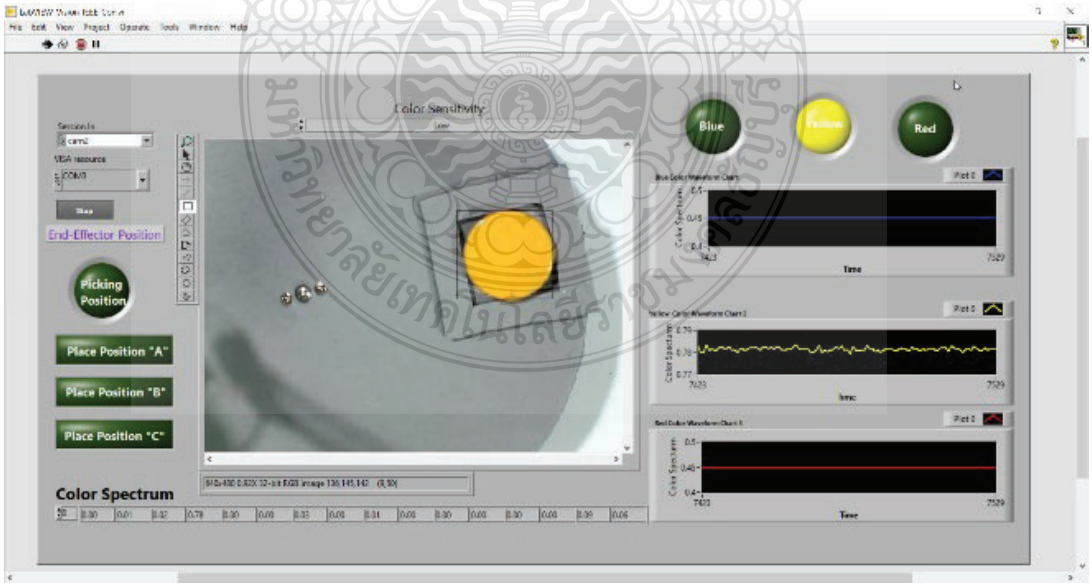



(c) LabVIEW Vision Color processing Front Panel

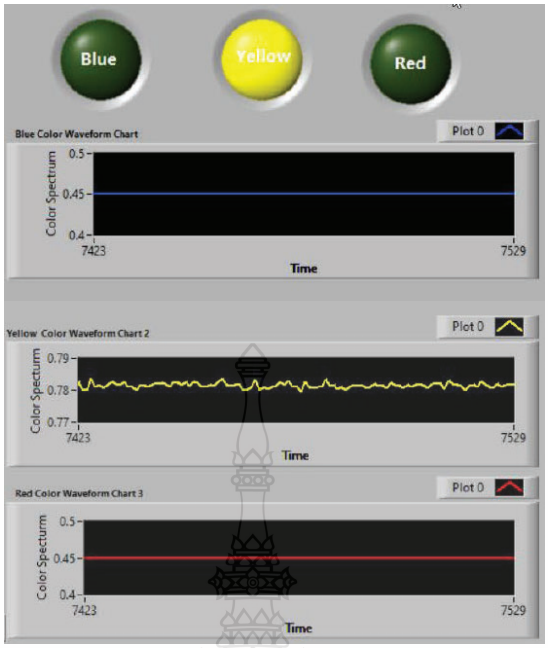



(d) Front Panel Object color matching and gain detection

**Figure 4.21** (c), (d) Front Panel of LabVIEW Vision show Object Blue color matching and Gain detection

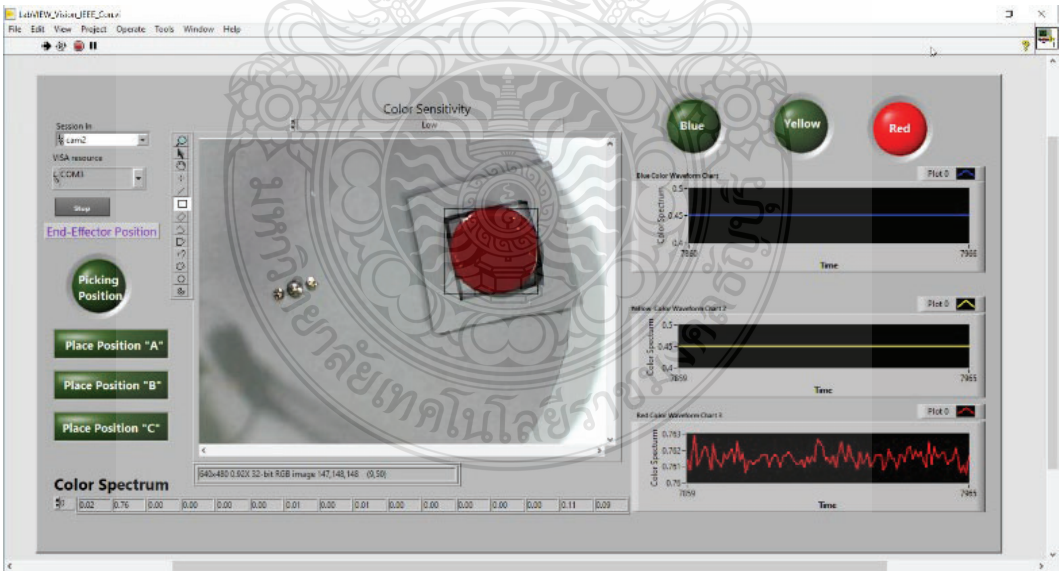



(e) LabVIEW Vision Color processing Front Panel

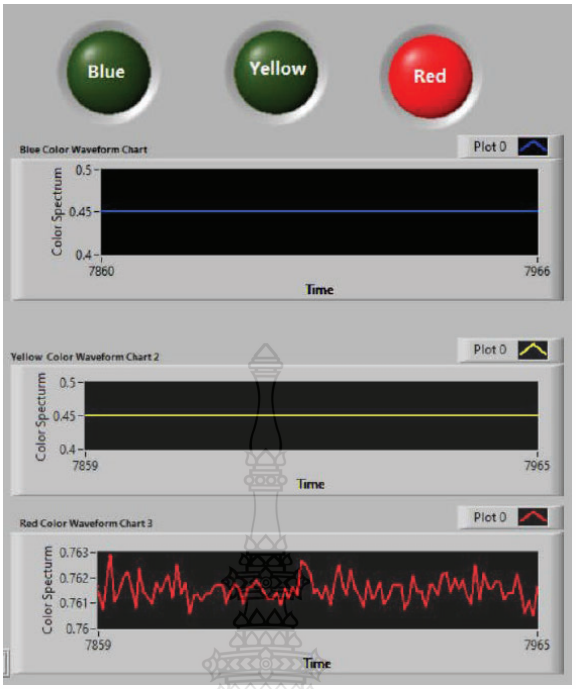



(f) Front Panel Object color matching and gain detection

**Figure 4.22** (e), (f) Front Panel of LabVIEW Vision show Object Yellow color




(g) LabVIEW Vision Color processing Front Panel




(h) Front Panel Object color matching and gain detection

**Figure 4.23** (g), (h) Front Panel of LabVIEW Vision show Object Red color matching and Gain detection

# CHAPTER5

# CONCLUSION AND RECOMMENDATION

## 5.1 Discussion and Recommendation

This research aims to design and write a simulation program for the motion of the robotic arm in order to obtain data on the working area and trajectory of the parallel robot by determining the size of the robot's structure accordingly. Based on the calculations from the kinematic theory of parallel robots and using the data obtained from simulations as a reference, designing a prototype parallel robot to work with the machine vision system and the conveyor belt so that both Collaborative systems like automation systems work with robots that are widely used in industrial plants today.

Actual experimental results from a prototype robot. Comparison with the kinematic simulation results of the MATLAB program for all 4 cases. From Figure 4.15 and Figure 4.16, it can be seen that the robot movement range of the program developed using MATLAB/GUI is Parallel robot operation x = 0 mm, y = 0 mm, z = -150 to 100 mm and θ1 = 38° to -40 °, θ2 = 38° to -40°, θ3 = 38° to -40° and the results of a 3-stage experiment using the inverse kinetic approach. The positions of the manipulator joint angles θ1, θ2, θ3 receive signals from angle sensors. It can be seen from the graphs in Figure 4.17, Figure 4.18, Figure 4.19 and Tables 4.5, 4.6, and 4.7 that the maximum error value at each step is 4°. Positional manipulation of the end effector creates positional differences because the robot has multiple joints that move during manipulation. And the control system does not take into account the inertia of the robot. Controlling the robot's inertia is a challenge. The inertia of the robot is constantly changing as the end effector is placed in different positions. Results will show slight discrepancies over time due to various inherent factors. However, the actual results are consistent with the experimental target results obtained from the kinematic simulation, which helps us to understand and know how the robot behaves in each situation. And in all dimensions of the joint motion of the robot. Experimental data ensure that the use of a kinematic simulation program to simulate the movement of the robot arm can be used as a data source already at the robot design stage, thus reducing time. and reduce the risk of problems due to the limitations of

the robot. And it can be used as information to help factories choose the use of robots to adapt them to the production conditions that the industrial factory also needs.

### 5.2 Implication for Practice and Future Research

The next step is to make robots more efficient by installing additional equipment such as IOT vision control signals and robot control signals, and using AI and IOT systems to receive signals from various sensors. Reduce the use of signal cables where installation space is limited. And increase the convenience and speed of data collection, and further optimize the control of robot motion.

## List of Bibliography


K. Oranoot, “A STUDY OF FACTORS AND EFFECTS OF INDUSTRY 4.0 POLICY ON THAI ELECTRONICS INDUSTRY,” AcademicYear2017 from http://ithesisir.su.ac.th/dspace/handle/123456789/1783

Y. Jirasak, K. Siravit, H. Natnicha Hasoontree, “READINESS OF USING AUTOMATED MANUFACTURING SYSTEM TO ENHANCE PRODUCTION CAPABILITIES FOR AUTOMOTIVE PARTS IN THAILAND,” Panyapiwat Journal Vol.10 No.3 September - December 2018

W. Thunyalak, “Automation and Required Skills for Thai Suppliers in the Automobile Industry,” Journal of Social Work Vol. 27 No.2 July-December 2019

Akekachai Pannawan, Supattarachai Sudsawat, “Automated part inspection by image processing system in vehicle part manufacturing,” The Journal of Applied Science ISSN 1513-7805 Printed in Thailand Vol. 16 No. 1: 45-59 [2017]

M. Dechrit “Industrial Robotics & mechatronics Applications” Bangkok, Thailand, SE-EDUCATION, 2018

F. Faris, D. Burhanudin, W. Putri, R. Irmawan, A. Dwi, “Vision Application of LabVIEW: IGUI for Face and Pattern Detection in Real Time,” 2020 International Conference on Information Management and Technology (ICIMTech)

Michel A. Aguilar-Torres, Amadeo J. Argüelles-Cruz, “A real time artificial vision implementation for quality inspection of industrial products,” Electronics, Robotics and Automotive Mechanics Conference 2008

Z. Yao “Dynamic Detection System of Workpiece Based on Machine Vision,” 2010 International Conference on Intelligent Computation Technology and Automation

Shrenika R M, Swati S Chikmath, Ravi Kumar A V, Divyashree Y V, Roopa K Swamy, “Non-Contact Water Level Monitoring System Implemented using LabVIEW and Arduino,” 2017 International Conference on Recent Advances in Electronics and Communication Technology

F. J. Jiménez, F. R. Lara and M. D. Redel “API for communication Between Labview

and Arduino UNO," IEEE LATIN AMERICA TRANSACTIONS, VOL. 12, NO. 6, SEPTEMBER 2014 971

Abdulghader Elfasi, M0hamed Abdussalam Shawesh, Waid.T.Shanab , Abdulaziz Khaled thabet, "Oscilloscope using Arduino interface LabVIEW," 2017 International Conference on Green Energy Conversion Systems (GECS)

Hyo-Jeong Cha; Jae-Hong Woo, Hanyang University, Ansan, Korea; Byung-Ju Yi; Chanhun Park "Workspace Analysis of the DELTA robot according to robot parameters and ball joints," 2013 10th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)

M. AFROUN, T. CHETTIBI, S. HANCHI "Planning Optimal Motions for a DELTA Parallel Robot," 2006 14th Mediterranean Conference on Control and Automation

St. Staicu;D.C. Carp-Ciocardia "DYNAMIC ANALYSIS OF CLAVEL' S DELTA PARALLEL ROBOT," 2003 IEEE International Conference on Robotics and Automation

Mostafa Mahmoodi; Mahmood Ghafouri Tabrizi; Khalil Alipour, "A New Approach for Kinematics-based Design of 3-RRR Delta Robots with a Specified Workspace," Conference 2015 AI & Robotics (IRANOPEN)

Sorawit Stapornchaisit, Chowarit Mitsantisuk, Nattapon Chayopitak, Yasuharu Koike, "Bilateral control in delta robot by using Jacobian matrix," 2015 6th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES)

P. Pradya "Workspace and Dynamic Trajectory Tracking of Delta Parallel Robot," 2014 International Computer Science and Engineering Conference (ICSEC)

Eric McCormick, Yanjun Wang, Haoxiang Lang, "Optimization of a 3-RRR Delta Robot for a Desired Workspace with Real-Time Simulation in MATLAB," The 14th International Conference on Computer Science & Education (ICCSE 2019) August 19-21, 2019. Toronto, Canada

Jan Rehbein, Tim Wrütz, Rolf Biesenbach "Model-based industrial robot programming with MATLAB/Simulink," 2019 20th International Conference on Research and Education in Mechatronics (REM)

https://www.ni.com/docs/en-US/bundle/ni-vision-labview-api-ref/page/imaqvision/imaq_colormatch.html

J. A. BorjaI, AlvaradoD, Muñoz de la Peña “Low cost two-wheels self-balancing robot for control education powered by stepper motors,” IFAC-PapersOnLine 2020

Gauri Shanker, GuptaPrabhat, Ranjan Tripathi “Prototype design for bidirectional control of stepper motor using features of brain signals and soft computing tools,” Biomedical Signal Processing and Control25 October 2021

# APPENDICES

# APPENDIX A
# MATLAB / GUI Program for Parallel Robotic Kinematic Simulation

```
%=============================================================
===========
%  S_Delta Robot
%  2022-01-8
%=============================================================
===========
function Delta_Robot__2022_01_08
close all
clear
clc
%--- Define global variables ---------------------------------
-----------
global end_program ploting k0 m_loops
global A B C
global psi M inv_M
global Xref Yref Zref Pref Line_P
global V_LINK_A0
global LINK_A  LINK_B  LINK_Bs  BALLJOINT_B  BALLJOINT_C
CONNECTER_PC
global LINK_B0 LINK_Bs0 BALLJOINT_B0 BALLJOINT_C0 CONNECTER_C0
global PLATE_C
global PLATE_C0
global thickness_PLATE
global trajectory
global Vector_Bs
global Line_Theta1 Line_Theta2 Line_Theta3
global Line_X Line_Y Line_Z
global Lr La Lb Lh Lb_s
%--- Schematic diagram ---------------------------------------
-----------
%
%      ---> x
%  O(0,0,0)       Ai(Lr,0,0)
%      O-----------o--------
%           Lr      \ )ซCi
%                    \
%                  La \  <-- Link A
%                      \
%                       o Bi(x_b,0,z_b)
%                       /
%                      /
%                  Lb /  <-- Link B
%                    /
%                   /
%  P(x,y,z) o------o Ci(a,b,c)
%              Lh
%
%-------------------------------------------------------------
-----------
Lr   = 160;  % (mm) dis(O,Ai)
La   = 200;  % (mm) Length of link A
Lb   = 500;  % (mm) Length of link B (long side)
Lb_s = 75;   % (mm) Length of link B (short side)
Lh   = 75;   % (mm) dis(P,Ci)
thickness_PLATE = 1;  % (mm)
```

```
%-------------------------------------------------------------
-----------
trajectory = 1;
% Select the trajectory of the end effector
%     (1) Vertical line
%     (2) Upper disk
%     (3) Lower disk
%     (4) Upper and lower circles
%     (5) Cylinder
%     (6) Sphere
%     (7) Trapezoidal velocity trajectory
%--- Call SET_PARAMETERS (Set robotic arm parameters) ----------
-----------
SET_PARAMETERS
[Pref, m_loops] = SetTrajectory( trajectory );

%--- Call CREAT_FIGURE1 (Creat figure 1: 3D Delta Robot) -------
-----------
CREAT_FIGURE1

%--- Call CREAT_FIGURE2 (Creat figure 2: Time Response) --------
-----------
CREAT_FIGURE2

%--- Call CREAT_FIGURE3 (Creat figure 3: UI) -------------------
-----------
CREAT_FIGURE3

%=== Main loopยก]Solve the inverse kinematicsยก^
============================
ploting = 0;
k0 = 0;
theta_A = zeros(3,1);
V_CONNECTER_C = cell(3,1);
V_LINK_B = cell(3,1);
V_LINK_Bs = cell(3,2);
V_BALLJOINT_B = cell(3,2);
V_BALLJOINT_C = cell(3,2);

while ~end_program
    if ploting
        k0 = k0+1;
        P = [ Xref(k0); Yref(k0); Zref(k0) ];
        %--- Inverse kinematics -------------------------------
-----------
        for i=1:3  % Calculate the coordinates: B{*}, C{*}
            C{i} = P +M{i}*[Lh;0;0];
            Ct = inv_M{i}*C{i};
            a  = Ct(1);
            b  = Ct(2);
            c  = Ct(3);
            alpha = (Lr-a)/c;
            beta  = 0.5*( a*a +b*b +c*c +La*La -Lb*Lb -Lr*Lr
)/c;
            gamma = Lr -alpha*beta;
            sigma = 1 +alpha^2;
```

```
            delta = gamma*gamma -sigma*(Lr*Lr+beta*beta-La*La);
            if delta>0
                x_b = ( gamma +sqrt(delta) )/sigma;  % x1>x2
ยกร?(ยข^-ยกร"ยข_)/ยขm
                z_b = alpha*x_b +beta;
                if x_b>Lr  % theta<90
                    theta = -asind( z_b/La );
                else       % theta>=90
                    theta = 180 +asind( z_b/La );
                end
            else
                set( findobj('Tag','text_error'),'Visible','on')
                k0 = m_loops;
                ploting = 0;
                set( findobj('Tag','pb_ResetRobot'),
'Enable','on' )
                set( findobj('Tag','pb_Run'), 'Enable','off' )
                break
            end
            Bt = [ Lr+La*cosd(theta); 0; -La*sind(theta) ];
            B{i} = M{i}*Bt;
            theta_A(i) = theta;
        end  % END for i=1:3  % Calculate the coordinates: B{*},
C{*}
        %-----------------------------------------------------
-----------
        for i=1:3
            %--- LINK_A ----------------------------------------
-----------
            set( LINK_A(i), 'Vertices', V_LINK_A0{i} )
            rotate( LINK_A(i),...
                Vector_Bs{i},...A{k1}-A{k2},...
                theta_A(i),...
                A{i} )
            % --- CONNECTOR_PC ---------------------------------
-----------
            V_CONNECTER_C{i} = get( CONNECTER_C0, 'Vertices' );
            set( CONNECTER_PC(i), 'Vertices', V_CONNECTER_C{i} )
            rotate( CONNECTER_PC(i), [0,0,1], psi(i), [0,0,0] )
            V_CONNECTER_C{i} = get( CONNECTER_PC(i), 'Vertices'
);
            V_CONNECTER_C{i}(:,1) = V_CONNECTER_C{i}(:,1)
+C{i}(1);
            V_CONNECTER_C{i}(:,2) = V_CONNECTER_C{i}(:,2)
+C{i}(2);
            V_CONNECTER_C{i}(:,3) = V_CONNECTER_C{i}(:,3)
+C{i}(3)...
                -0.5*thickness_PLATE;
            set( CONNECTER_PC(i), 'Vertices', V_CONNECTER_C{i} )
            %--- LINK_B ----------------------------------------
-----------
            for j=1:2
                %--- LINK_B (Parallel four-bar linkage: long
side) --------
                V_LINK_B{i,j} = get( LINK_B0, 'Vertices' );
                set(  LINK_B(i,j), 'Vertices', V_LINK_B{i,j} )
```

```
                rotate( LINK_B(i,j), [0,1,0],...
                    acosd((B{i}(3)-C{i}(3))/Lb), [0,0,0] )
                rotate( LINK_B(i,j), [0,0,1], -psi(i), [0,0,0] )
                rotate( LINK_B(i,j), [0,0,1],...
                    atan2d( (B{i}(2)-C{i}(2)), (B{i}(1)-C{i}(1))
),...
                    [0,0,0] )
                rotate( LINK_B(i,j), [0,0,1], psi(i), [0,0,0] )
                V_LINK_B{i,j} = get( LINK_B(i,j),'Vertices' );
                switch j
                    case 1
                        V_LINK_B{i,j}(:,1) =
V_LINK_B{i,j}(:,1)...
                            +C{i}(1)+Vector_Bs{i}(1);
                        V_LINK_B{i,j}(:,2) =
V_LINK_B{i,j}(:,2)...
                            +C{i}(2)+Vector_Bs{i}(2);
                        V_LINK_B{i,j}(:,3) =
V_LINK_B{i,j}(:,3)...
                            +C{i}(3)+Vector_Bs{i}(3);
                    case 2
                        V_LINK_B{i,j}(:,1) =
V_LINK_B{i,j}(:,1)...
                            +C{i}(1)-Vector_Bs{i}(1);
                        V_LINK_B{i,j}(:,2) =
V_LINK_B{i,j}(:,2)...
                            +C{i}(2)-Vector_Bs{i}(2);
                        V_LINK_B{i,j}(:,3) =
V_LINK_B{i,j}(:,3)...
                            +C{i}(3)-Vector_Bs{i}(3);
                end
                set( LINK_B(i,j), 'Vertices', V_LINK_B{i,j} )
                %--- LINK_B (Parallel four-bar linkage: short
side) -------
                V_LINK_Bs{i,j} = get( LINK_Bs0, 'Vertices' );
                set(  LINK_Bs(i,j), 'Vertices', V_LINK_Bs{i,j} )
                rotate( LINK_Bs(i,j), [0,1,0],...
                    acosd(abs(C{i}(3)-C{i}(3))/Lb), [0,0,0] )
                rotate( LINK_Bs(i,j), [0,0,1], psi(i), [0,0,0] )
                V_LINK_Bs{i,j} = get( LINK_Bs(i,j),'Vertices' );
                switch j
                    case 1
                        V_LINK_Bs{i,j}(:,1) =
V_LINK_Bs{i,j}(:,1)...
                            +B{i}(1);
                        V_LINK_Bs{i,j}(:,2) =
V_LINK_Bs{i,j}(:,2)...
                            +B{i}(2);
                        V_LINK_Bs{i,j}(:,3) =
V_LINK_Bs{i,j}(:,3)...
                            +B{i}(3);
                    case 2
                        V_LINK_Bs{i,j}(:,1) =
V_LINK_Bs{i,j}(:,1)...
                            +C{i}(1);
```

```
                        V_LINK_Bs{i,j}(:,2) =
V_LINK_Bs{i,j}(:,2)...
                            +C{i}(2);
                        V_LINK_Bs{i,j}(:,3) =
V_LINK_Bs{i,j}(:,3)...
                            +C{i}(3);
                end
                set( LINK_Bs(i,j), 'Vertices', V_LINK_Bs{i,j} )
                %--- BALLJOINT ---------------------------------
-----------
                V_BALLJOINT_B{i,j} = get(
BALLJOINT_B0,'Vertices' );
                V_BALLJOINT_B{i,j}(:,1) =
V_BALLJOINT_B{i,j}(:,1)...
                    +B{i}(1)-(-1)^j*Vector_Bs{i}(1);
                V_BALLJOINT_B{i,j}(:,2) =
V_BALLJOINT_B{i,j}(:,2)...
                    +B{i}(2)-(-1)^j*Vector_Bs{i}(2);
                V_BALLJOINT_B{i,j}(:,3) =
V_BALLJOINT_B{i,j}(:,3)...
                    +B{i}(3)-(-1)^j*Vector_Bs{i}(3);
                set( BALLJOINT_B(i,j), 'Vertices',
V_BALLJOINT_B{i,j} )
                V_BALLJOINT_C{i,j} = get(
BALLJOINT_C0,'Vertices' );
                V_BALLJOINT_C{i,j}(:,1) =
V_BALLJOINT_C{i,j}(:,1)...
                    +C{i}(1)-(-1)^j*Vector_Bs{i}(1);
                V_BALLJOINT_C{i,j}(:,2) =
V_BALLJOINT_C{i,j}(:,2)...
                    +C{i}(2)-(-1)^j*Vector_Bs{i}(2);
                V_BALLJOINT_C{i,j}(:,3) =
V_BALLJOINT_C{i,j}(:,3)...
                    +C{i}(3)-(-1)^j*Vector_Bs{i}(3);
                set( BALLJOINT_C(i,j), 'Vertices',
V_BALLJOINT_C{i,j} )
                %-------------------------------------------------
-----------
            end  % END: for j=1:2
        end  % END: for i=1:3
        %--- Plot the motion trajectory ------------------------
-----------
        addpoints( Line_P, Xref(k0), Yref(k0), Zref(k0) )
        addpoints( Line_Theta1, k0, theta_A(1) )
        addpoints( Line_Theta2, k0, theta_A(2) )
        addpoints( Line_Theta3, k0, theta_A(3) )
        addpoints( Line_X, k0, Xref(k0) )
        addpoints( Line_Y, k0, Yref(k0) )
        addpoints( Line_Z, k0, Zref(k0) )
        %--- PLATE_C (Bottom tool mounting plate) --------------
-----------
        V_PLATE_C = get( PLATE_C0,'Vertices' );
        V_PLATE_C(:,1) = V_PLATE_C(:,1) +Xref(k0);
        V_PLATE_C(:,2) = V_PLATE_C(:,2) +Yref(k0);
        V_PLATE_C(:,3) = V_PLATE_C(:,3) +Zref(k0);
        set( PLATE_C, 'Vertices', V_PLATE_C )
```

```
        %-------------------------------------------------------
-----------
    end  % END: if ploting
    if k0>m_loops-1
        k0 = 0;
        ploting = 0;
        set( findobj('Tag','pb_Run'), 'String','Run' )
    end
    drawnow limitrate
end  % END: while (k0<=m_loops-1) && (~end_program)
close all
end  % END: MAIN


%% === function: SET_PARAMETERS
=========================================
function SET_PARAMETERS
global Lr La Lb Lh Lb_s
global end_program
global N_sides
global line_width
global marker_size
global XLim YLim ZLim
global XLength YLength ZLength
global A B C
global Vector_Bs
global psi c_psi s_psi M inv_M
global Xinit Yinit Zinit
global radius_Lb_s radius_Lb radius_BALL
global thickness_PLATE
global path_color marker_color
global COVER_color
global MOTOR_color MOTOR_SUP_color
global LINK_A_color LINK_B_color BALLJOINT_color
global PLATE_A_color PLATE_C_color CONNECTER_C_color
global COVER_face_alpha face_alpha
%-------------------------------------------------------------------
-----------
end_program = 0;
N_sides = 8;
line_width = 0.5;
marker_size = 4;
%% --- The display range of the axes object ax_xyz -------------
-----------
XLim = 300*[-1,1];          % X-axis limits for the axes object
ax_xyz.
YLim = 300*[-1,1];          % Y-axis limits for the axes object
ax_xyz.
ZLim = [-750,300];          % Z-axis limits for the axes object
ax_xyz.
XLength = XLim(2) -XLim(1);
YLength = YLim(2) -YLim(1);
ZLength = ZLim(2) -ZLim(1);
%% --- Define the colors and transparency of graphics objects --
-----------
color_table{1}  = [ 255,   0,   0 ]/255;  %  1, 'r', red
```

```
color_table{2}  = [ 255, 165,   0 ]/255;  %  2, '-', orange
color_table{3}  = [ 255, 255,   0 ]/255;  %  3, 'y', yellow
color_table{4}  = [   0, 128,   0 ]/255;  %  4, 'g', green128
color_table{5}  = [   0,   0, 255 ]/255;  %  5, 'b', blue
color_table{6}  = [ 128,   0, 128 ]/255;  %  6, '-', purple
color_table{7}  = [   0, 255,   0 ]/255;  %  7, '-', green255
color_table{8}  = [   0, 255, 255 ]/255;  %  8, 'c', cyan
color_table{9}  = [ 255,   0, 255 ]/255;  %  9, 'm', magenta
color_table{10} = [  70, 130, 180 ]/255;  % 10, '-', steelblue
color_table{11} = [ 255, 255, 255 ]/255;  % 11, '-', white
color_table{12} = [ 129, 216, 208 ]/255;  % 12, '-', Tiffany
Blue
%-------------------------------------------------------------------
-----------
COVER_color = color_table{8};          %  8, 'c', cyan
MOTOR_color = 0.1*[ 1, 1, 1 ];         %          black
MOTOR_SUP_color = color_table{11};     % 11, '-', white
PLATE_A_color = color_table{10};       % 10, '-', steelblue
PLATE_C_color = color_table{3};        %  3, 'y', yellow
LINK_A_color = color_table{9};         %  9, 'm', magenta
LINK_B_color = color_table{4};         %  4, 'g', green128
BALLJOINT_color = color_table{1};      %  1, 'r', red
CONNECTER_C_color = color_table{3};    %  3, 'y', yellow
path_color = color_table{5};           %  5, 'b', blue
marker_color = color_table{9};         %  9, 'm', magenta
COVER_face_alpha = 0.3;
face_alpha = 0.4;
%-------------------------------------------------------------------
-----------
Xinit = 0;
Yinit = 0;
Zinit = -sqrt( Lb^2 -(La+Lr-Lh)^2 );
psi = [0,120,240];
c_psi = cosd( psi );
s_psi = sind( psi );
M = cell(1,3);
A = cell(1,3);
B = cell(1,3);
C = cell(1,3);
inv_M = cell(1,3);
Vector_Bs = cell(1,3);
for i=1:3
    M{i} = [ c_psi(i), -s_psi(i), 0; s_psi(i), c_psi(i), 0; 0,
0, 1 ];
    inv_M{i} = [ c_psi(i), s_psi(i), 0; -s_psi(i), c_psi(i), 0;
0, 0, 1 ];
    A{i} = M{i}*[Lr;0;0];
end
Vector_Bs{1} = 0.5*Lb_s*(A{2}-A{3})/norm(A{2}-A{3});
Vector_Bs{2} = 0.5*Lb_s*(A{3}-A{1})/norm(A{3}-A{1});
Vector_Bs{3} = 0.5*Lb_s*(A{1}-A{2})/norm(A{1}-A{2});
radius_Lb = 4;
radius_Lb_s = radius_Lb;
radius_BALL = radius_Lb*2;
thickness_PLATE = 10;
end  % END: function SET_PARAMETERS
```

```
%% === function: CREAT_FIGURE1 --> 3D Delta robot
=========================
function CREAT_FIGURE1
%% --- Declare global variables --------------------------------
-----------
global ax_xyz
global XLim YLim ZLim
global XLength ZLength
global N_sides
global line_width
global marker_size
global La Lb Lh Lr Lb_s
global thickness_PLATE
global psi M inv_M
global A B C
global theta_A
global Line_P
global text_A
global Xref Yref Zref Pref
global LINK_A  LINK_B  LINK_Bs  CONNECTER_PC
global LINK_A0 LINK_B0 LINK_Bs0 CONNECTER_C0
global BALLJOINT_B  BALLJOINT_C
global BALLJOINT_B0 BALLJOINT_C0
global V_LINK_A0 V_LINK_B0 V_LINK_Bs0 V_PLATE_C0 V_CONNECTER_PC0
global V_BALLJOINT_B0 V_BALLJOINT_C0 Vector_Bs
global PLATE_C  MOTOR_MOUNT  MOTOR
global PLATE_C0
global radius_Lb_s radius_Lb radius_BALL
global path_color marker_color
global COVER_color
global MOTOR_color MOTOR_SUP_color
global LINK_A_color LINK_B_color BALLJOINT_color
global PLATE_A_color PLATE_C_color
global COVER_face_alpha face_alpha
%% --- Set the basic drawing environment -----------------------
-----------
fig1 = figure;
set( fig1,...
    'Tag','fig1',...
    'NumberTitle','off',...
    'Name','Delta Robot',...
    'Units','normalized',...
    'OuterPosition',[ -0.006, 0.04, 0.55, 0.96 ],...
    'Color',[ 0.85, 0.95, 0.85 ] )
%--- Creat the axes object: ax_xyz ---------------------------
-----------
ax_xyz = axes(...
    'Parent',fig1,...
    'Tag','ax_xyz',...
    'XLim',XLim,...
    'YLim',YLim,...
    'ZLim',ZLim,...
    'View',[25,15],...
    'FontName','Times',...
    'FontSize',12,...
```

```
    'Clipping','off',...
    'XTick',XLim(1):100:XLim(2),...
    'YTick',YLim(1):100:YLim(2),...
    'ZTick',ZLim(1):100:ZLim(2),...
    'TickDir','in',...
    'TickLength',[0,0],...
    'Position', [0.09, 0.02, 0.84, 1],...
    'Projection','orthographic',...
    'Color',[0.8 0.8 0.9] );
Line_P = animatedline('Parent',ax_xyz,...
    'Tag','Line_P',...
    'Color',path_color,...
    'Marker','.',...
    'MarkerSize',marker_size,...
    'MarkerEdgeColor',marker_color,...
    'LineWidth',line_width );
text(0,0,-400,' Out of workspace! ',...
    'Tag','text_error',...
    'Color',[1,0,0],...
    'BackgroundColor',0.95*[1,1,1],...
    'FontName','Times New Roman',...
    'FontSize',16,...
    'FontWeight','bold',...
    'HorizontalAlignment','center',...
    'Visible','off' );
% cameratoolbar('Show')
axis square
grid on
box on
rotate3d on
%% --- Plot the x-y-z axes ---------------------------------------
-----------
text( 0.18*XLength, 0, ZLim(1)+10,...
    'x',...
    'Parent',ax_xyz,...
    'FontName','Times',...
    'FontSize', 16,...
    'Color','b',...
    'FontAngle','italic',...
    'HorizontalAlignment','center')
text( 0, 0.2*XLength, 0.0025*XLength+ZLim(1),...
    'y',...
    'Parent',ax_xyz,...
    'FontName','Times',...
    'FontSize', 16,...
    'Color','b',...
    'FontAngle','italic',...
    'HorizontalAlignment','center')
text( 0, 0, 0.22*XLength+ZLim(1),...
    'z',...
    'Parent',ax_xyz,...
    'FontName','Times',...
    'FontSize', 16,...
    'Color','b',...
    'FontAngle','italic',...
    'HorizontalAlignment','center')
```

```
arrow_x0 = CREAT_CYLINDER(...
    [0;0;ZLim(1)],...
    [ 0, 0.005, 0.005, 0.010, 0 ]*XLength,...
    [ 0, 0.125, 0, 0.02 ]*XLength,...
    32,...
    [1, 0, 0] );
set( arrow_x0, 'Parent',ax_xyz )
rotate( arrow_x0,...
    [0,1,0], 90,...
    [0;0;ZLim(1)] )
arrow_y0 = CREAT_CYLINDER(...
    [0;0;ZLim(1)],...
    [ 0, 0.005, 0.005, 0.010, 0 ]*XLength,...
    [ 0, 0.125, 0, 0.02 ]*XLength,...
    32, [0, 1, 0] );
set( arrow_y0, 'Parent',ax_xyz )
rotate( arrow_y0,...
    [1,0,0],...
    -90,...
    [0;0;ZLim(1)] )
arrow_z0 = CREAT_CYLINDER(...
    [0;0;ZLim(1)],...
    [ 0, 0.005, 0.005, 0.010, 0 ]*ZLength,...
    [ 0, 0.125, 0, 0.02 ]*ZLength,...
    32,...
    [0, 0,1] );
set( arrow_z0,'Parent',ax_xyz )
rotate( arrow_z0,...
    [0,0,1],...
    -90,...
    [0;0;ZLim(1)] )
%--- ax_xyz.XLabel ------------------------------------------
-----------
text( mean(XLim), 1.25*YLim(1), ZLim(1)-20,...
    'x',...
    'Parent',ax_xyz,...
    'FontName','Times',...
    'FontSize',16,...
    'Color','b',...
    'FontAngle','italic',...
    'HorizontalAlignment','center' )
%--- ax_xyz.YLabel ------------------------------------------
-----------
text( 1.3*XLim(2), mean(YLim), ZLim(1)-20,...
    'y',...
    'Parent',ax_xyz,...
    'FontName','Times',...
    'FontSize', 16,...
    'Color','b',...
    'FontAngle','italic',...
    'HorizontalAlignment','center' )
%--- ax_xyz.ZLabel ------------------------------------------
-----------
text( XLim(1)-40, 1.15*YLim(1), mean(ZLim),...
    'z',...
    'Parent',ax_xyz,...
```

```
    'FontName','Times',...
    'FontSize', 16,...
    'Color','b',...
    'FontAngle','italic',...
    'HorizontalAlignment','center' )
%% --- Calculate the coordinates: B{*}, C{*} ------------------
-----------
for i=1:3
    Pt = inv_M{i}*Pref(1,:)';
    a  = Pt(1) +Lh;
    b  = Pt(2);
    c  = Pt(3);
    alpha = (Lr-a)/c;
    beta = ( a*a +b*b +c*c +La*La -Lb*Lb -Lr*Lr )/c*0.5;
    gamma = Lr -alpha*alpha;
    sigma = 1 +alpha*alpha;
    x_b = ( gamma...
        +sqrt(gamma*gamma-sigma*(Lr*Lr+beta*beta-La*La)) )...
        /( 1+alpha*alpha );
    z_b = alpha*x_b +beta;
    theta = asind( -z_b/La );
    theta_A(i) = theta;
    Bt = [ Lr+La*cosd(theta); 0; -La*sind(theta) ];
    B{i} = M{i}*Bt;
    C{i} = M{i}*[ Lh; 0; 0 ] +Pref(1,:)';
end
for i=1:3
    text_A(i) = text('parent',ax_xyz,...
        'Position',A{i}+[0;0;70],...
        'HorizontalAlignment','Center',...
        'FontName','Times New Roman',...
        'FontSize',14,...
        'Color',[1,1,0],...
        'String',['\bfA_',num2str(i)] );
end
%% --- LINK_A -------------------------------------------------
-----------
Ra = 0.02*XLength;
LINK_A_height = [ 0;0.02*XLength; 0 ];
Xa = [ zeros(2*N_sides,1);
    Ra*cosd(linspace(270,90,N_sides))';...
    Ra*cosd(linspace(90,-90,N_sides))'+La*ones(N_sides,1);...
    Ra*cosd(linspace(270,90,N_sides))';...
    Ra*cosd(linspace(90,-90,N_sides))'+La*ones(N_sides,1);...
    zeros(2*N_sides,1)];
Ya = [ zeros(2*N_sides,1);
    Ra*sind(linspace(270,90,N_sides))';...
    Ra*sind(linspace(90,-90,N_sides))';...
    Ra*sind(linspace(270,90,N_sides))';...
    Ra*sind(linspace(90,-90,N_sides))';...
    zeros(2*N_sides,1);];
%-------------------------------------------------------------
-----------
LINK_A0 = CREAT_M_PLATE(...
    [ 0; 0; -0.5*LINK_A_height(2) ],...
    Xa,...
```

```
    Ya,...
    LINK_A_height,...
    LINK_A_color,...
    'off' );
rotate( LINK_A0, [1,0,0],  90, [0,0,0] )
set( LINK_A0, 'Parent',ax_xyz )
for i=1:3
    LINK_A(i) = copyobj( LINK_A0, gca );
    set( LINK_A(i), 'Visible','on' );
    rotate( LINK_A(i), [0,0,1], psi(i), [0,0,0] )
    V_LINK_A0{i} = get( LINK_A(i),'Vertices' );
    for j=1:3
        V_LINK_A0{i}(:,j) = V_LINK_A0{i}(:,j) +A{i}(j);
    end
    set( LINK_A(i), 'Vertices',V_LINK_A0{i} )
end
%% --- LINK_B (Parallel four-bar linkage: long side) -----------
-----------
LINK_B0 = CREAT_CYLINDER(...
    [ 0; 0; 0 ],...
    radius_Lb*[ 0; 1; 1; 0 ],...
    [ 0; Lb; 0 ],...
    N_sides,...
    LINK_B_color,...
    'off' );
set( LINK_B0, 'Parent',ax_xyz )
V_LINK_B = cell(3,2);
for i=1:3
    for j=1:2
        LINK_B(i,j) = copyobj( LINK_B0, gca );
        set( LINK_B(i,j), 'Visible','on' )
        rotate( LINK_B(i,j), [0,1,0],...
            acosd((B{i}(3)-C{i}(3))/Lb), [0,0,0] )
        rotate( LINK_B(i,j), [0,0,1], -psi(i), [0,0,0] )
        rotate( LINK_B(i,j), [0,0,1],...
            atan2d( (B{i}(2)-C{i}(2)), (B{i}(1)-C{i}(1)) ),
[0,0,0] )
        rotate( LINK_B(i,j), [0,0,1], psi(i), [0,0,0] )
        V_LINK_B{i,j} = get( LINK_B(i,j),'Vertices' );
        switch j
            case 1
                V_LINK_B{i,j}(:,1) =
V_LINK_B{i,j}(:,1)+C{i}(1)...
                    +Vector_Bs{i}(1);
                V_LINK_B{i,j}(:,2) =
V_LINK_B{i,j}(:,2)+C{i}(2)...
                    +Vector_Bs{i}(2);
                V_LINK_B{i,j}(:,3) =
V_LINK_B{i,j}(:,3)+C{i}(3)...
                    +Vector_Bs{i}(3);
            case 2
                V_LINK_B{i,j}(:,1) =
V_LINK_B{i,j}(:,1)+C{i}(1)...
                    -Vector_Bs{i}(1);
                V_LINK_B{i,j}(:,2) =
V_LINK_B{i,j}(:,2)+C{i}(2)...
```

```
                    -Vector_Bs{i}(2);
                V_LINK_B{i,j}(:,3) =
V_LINK_B{i,j}(:,3)+C{i}(3)...
                    -Vector_Bs{i}(3);
        end
        set( LINK_B(i,j), 'Vertices',V_LINK_B{i,j} )
    end
end
V_LINK_B0 = V_LINK_B;
%% --- LINK_Bs (Parallel four-bar linkage: short side) ---------
-----------
LINK_Bs0 = CREAT_CYLINDER(...
    [ 0; 0; -0.5*Lb_s ],...
    radius_Lb_s*[0;1;1;0],...
    [ 0; Lb_s; 0 ],...
    N_sides,...
    LINK_B_color,...
    'off' );
rotate( LINK_Bs0, [1,0,0], 90, [0,0,0] )
set( LINK_Bs0, 'Parent',ax_xyz )
V_LINK_Bs = cell(3,2);
for i=1:3
    for j=1:2
        LINK_Bs(i,j) = copyobj( LINK_Bs0, gca );
        set( LINK_Bs(i,j), 'Visible','on' )
        rotate( LINK_Bs(i,j), [0,0,1], psi(i), [0,0,0] )
        V_LINK_Bs{i,j} = get( LINK_Bs(i,j),'Vertices' );
        switch j
            case 1
                V_LINK_Bs{i,j}(:,1) =
V_LINK_Bs{i,j}(:,1)+B{i}(1);
                V_LINK_Bs{i,j}(:,2) =
V_LINK_Bs{i,j}(:,2)+B{i}(2);
                V_LINK_Bs{i,j}(:,3) =
V_LINK_Bs{i,j}(:,3)+B{i}(3);
            case 2
                V_LINK_Bs{i,j}(:,1) =
V_LINK_Bs{i,j}(:,1)+C{i}(1);
                V_LINK_Bs{i,j}(:,2) =
V_LINK_Bs{i,j}(:,2)+C{i}(2);
                V_LINK_Bs{i,j}(:,3) =
V_LINK_Bs{i,j}(:,3)+C{i}(3);
        end
        set( LINK_Bs(i,j), 'Vertices',V_LINK_Bs{i,j} )
    end
end
V_LINK_Bs0 = V_LINK_Bs;
%% --- BALLJOINT -----------------------------------------------
-----------
BJ_radius = radius_BALL*cosd( -90:10:90 );
BJ_height = radius_BALL*( sind(-80:10:90) -sind(-90:10:80) );
%-------------------------------------------------------------------
-----------
BALLJOINT_B0 = CREAT_CYLINDER(...
    [ 0; 0; -radius_BALL ],...
    BJ_radius,...
```

```
    BJ_height,...
    N_sides,...
    BALLJOINT_color,...
    'off' );
set( BALLJOINT_B0, 'Parent',ax_xyz )
%-------------------------------------------------------------------
-----------
BALLJOINT_C0 = CREAT_CYLINDER(...
    [ 0; 0; -0.5*Ra ],...
    BJ_radius,...
    BJ_height,...
    N_sides,...
    BALLJOINT_color,...
    'off');
set( BALLJOINT_C0, 'Parent',ax_xyz )
V_BALLJOINT_B = cell(3,2);
V_BALLJOINT_C = cell(3,2);
for i=1:3
    for j=1:2
        %---------------------------------------------------------
-----------
        BALLJOINT_B(i,j) = copyobj( BALLJOINT_B0, gca );
        set( BALLJOINT_B(i,j), 'Visible','on' )
        V_BALLJOINT_B{i,j} = get( BALLJOINT_B(i,j),'Vertices' );
        V_BALLJOINT_B{i,j}(:,1) =
V_BALLJOINT_B{i,j}(:,1)+B{i}(1)...
            -(-1)^j*Vector_Bs{i}(1);
        V_BALLJOINT_B{i,j}(:,2) =
V_BALLJOINT_B{i,j}(:,2)+B{i}(2)...
            -(-1)^j*Vector_Bs{i}(2);
        V_BALLJOINT_B{i,j}(:,3) =
V_BALLJOINT_B{i,j}(:,3)+B{i}(3)...
            -(-1)^j*Vector_Bs{i}(3);
        set( BALLJOINT_B(i,j), 'Vertices',V_BALLJOINT_B{i,j} )
        %---------------------------------------------------------
-----------
        BALLJOINT_C(i,j) = copyobj( BALLJOINT_C0, gca );
        set( BALLJOINT_C(i,j), 'Visible','on' )
        V_BALLJOINT_C{i,j} = get( BALLJOINT_C(i,j),'Vertices' );
        V_BALLJOINT_C{i,j}(:,1) =
V_BALLJOINT_C{i,j}(:,1)+C{i}(1)...
            -(-1)^j*Vector_Bs{i}(1);
        V_BALLJOINT_C{i,j}(:,2) =
V_BALLJOINT_C{i,j}(:,2)+C{i}(2)...
            -(-1)^j*Vector_Bs{i}(2);
        V_BALLJOINT_C{i,j}(:,3) =
V_BALLJOINT_C{i,j}(:,3)+C{i}(3)...
            -(-1)^j*Vector_Bs{i}(3);
        set( BALLJOINT_C(i,j), 'Vertices',V_BALLJOINT_C{i,j} )
        %---------------------------------------------------------
-----------
    end
end
V_BALLJOINT_B0 = V_BALLJOINT_B;
V_BALLJOINT_C0 = V_BALLJOINT_C;
```

```
%% --- PLATE_C (Bottom tool mounting plate) --------------------
-----------
TH = (30:60:330)';
Pr = (Lh-1.5*radius_BALL)/cosd(30);
Px = Pr*cosd(TH);
Py = Pr*sind(TH);
N = 6;
X1 = linspace( Px(1), Px(2), N )';
X2 = linspace( Px(2), Px(3), N )';
X3 = linspace( Px(3), Px(4), N )';
X4 = linspace( Px(4), Px(5), N )';
X5 = linspace( Px(5), Px(6), N )';
X6 = linspace( Px(6), Px(1), N )';
X7 = 0.25*Pr*cosd( linspace(30,30+360,3*(N+N)-5) )';
Y1 = linspace( Py(1), Py(2), N )';
Y2 = linspace( Py(2), Py(3), N )';
Y3 = linspace( Py(3), Py(4), N )';
Y4 = linspace( Py(4), Py(5), N )';
Y5 = linspace( Py(5), Py(6), N )';
Y6 = linspace( Py(6), Py(1), N )';
Y7 = 0.25*Pr*sind( linspace(30,30+360,3*(N+N)-5) )';
X16 = [ X1(1:end-1); X2(1:end-1); X3(1:end-1);...
    X4(1:end-1); X5(1:end-1); X6 ];
Y16 = [ Y1(1:end-1); Y2(1:end-1); Y3(1:end-1);...
    Y4(1:end-1); Y5(1:end-1); Y6 ];
XX = [ X16; X7; X7; X16; X16 ];
YY = [ Y16; Y7; Y7; Y16; Y16 ];
PLATE_C0 = CREAT_M_PLATE(...
    [ 0, 0, 0 ],...
    XX,...
    YY,...
    [ 0, -thickness_PLATE, 0, thickness_PLATE ],...
    PLATE_C_color,...
    'off' );
PLATE_C = copyobj( PLATE_C0, gca );
set( PLATE_C, 'Visible','on' );
V_PLATE_C = get( PLATE_C0,'Vertices' );
V_PLATE_C(:,1) = V_PLATE_C(:,1) +Xref(1);
V_PLATE_C(:,2) = V_PLATE_C(:,2) +Yref(1);
V_PLATE_C(:,3) = V_PLATE_C(:,3) +Zref(1);
set( PLATE_C, 'Vertices',V_PLATE_C )
V_PLATE_C0 = V_PLATE_C;
% set( PLATE_A_center, 'EdgeColor', [1,0,0] )
%% --- CONNECTOR_PC ------------------------------------------
-----------
Ph = Lh-1.5*radius_BALL;
length_PC = 0.5*Lh;
width_PC = 0.5*Lh;
CONNECTER_C0 = CREAT_CUBOID( [ Ph-Lh, -0.5*width_PC, -
0.5*thickness_PLATE ],...
    length_PC,...
    width_PC,...
    thickness_PLATE,...
    PLATE_C_color,...
    'off' );
V_CONNECTER_C = cell(3,1);
```

```
for i=1:3
    CONNECTER_PC(i) = copyobj( CONNECTER_C0, gca );
    set(CONNECTER_PC(i), 'Visible','on' );
    rotate( CONNECTER_PC(i), [0,0,1], psi(i), [0,0,0] );
    V_CONNECTER_C{i} = get( CONNECTER_PC(i),'Vertices' );
    V_CONNECTER_C{i}(:,1) = V_CONNECTER_C{i}(:,1) +C{i}(1);
    V_CONNECTER_C{i}(:,2) = V_CONNECTER_C{i}(:,2) +C{i}(2);
    V_CONNECTER_C{i}(:,3) = V_CONNECTER_C{i}(:,3) +C{i}(3)...
        -0.5*thickness_PLATE;
    set( CONNECTER_PC(i), 'Vertices', V_CONNECTER_C{i} );
end
V_CONNECTER_PC0 = V_CONNECTER_C;
%% --- MOTOR_MOUNT (Servo motor mount) -------------------------
-----------
length_motor_base    = 100;
width_motor_base     = 80;
thickness_motor_base = 0.8*thickness_PLATE;
X0 = zeros(1,9);
Y0 = zeros(1,9);
X1 = [ 0, 0,... (1,2)
    thickness_motor_base, thickness_motor_base,...(3,4)
    thickness_motor_base, thickness_motor_base,...(5,6)
    0, 0, 0 ];  % (7,8,1)
Y1 = [ -0.5*width_motor_base, -0.5*width_motor_base,...(1,2)
    -0.5*width_motor_base, -0.5*width_motor_base,...(3,4)
    0.5*width_motor_base, 0.5*width_motor_base,...(5,6)
    0.5*width_motor_base, 0.5*width_motor_base,...(7,8)
    -0.5*width_motor_base ];  % (1)
X2 = [ 0, 0,...(1,2)
    thickness_motor_base, thickness_motor_base,...(3,4)
    thickness_motor_base, thickness_motor_base,...(5,6)
    0, 0, 0 ];  % (7,8,1)
Y2 = [ -0.5*width_motor_base, -0.5*width_motor_base,...(1,2)
    -0.5*width_motor_base,...(3)
    -0.5*width_motor_base+thickness_motor_base,...(4)
    0.5*width_motor_base-thickness_motor_base,...(5)
    0.5*width_motor_base, 0.5*width_motor_base,...(6,7)
    0.5*width_motor_base, -0.5*width_motor_base ];  % (8,1)
X3a = [ 0, 0.5*width_motor_base,...(1,2)
    0.5*width_motor_base, thickness_motor_base,...(3,4)
    thickness_motor_base, 0.5*width_motor_base,...(5,6)
    0.5*width_motor_base, 0, 0 ];  % (7,8,1)
X3b = X3a;
Y3a = [ -0.5*width_motor_base, -0.5*width_motor_base,...(1,2)
    -0.5*width_motor_base+thickness_motor_base,...(3)
    -0.5*width_motor_base+thickness_motor_base,...(4)
    0.5*width_motor_base-thickness_motor_base,...(5)
    0.5*width_motor_base-thickness_motor_base,...(6)
    0.5*width_motor_base,  0.5*width_motor_base,...(7,8)
    -0.5*width_motor_base ];  % (1)
Y3b = Y3a;
X4 = [ 0, 0,...(1,2)
    length_motor_base, length_motor_base,...(3,4)
    length_motor_base, length_motor_base,...(5,6)
    0, 0, 0 ];  % (7,8,1)
Y4 = [ -0.5*width_motor_base, -0.5*width_motor_base,...(1,2)
```

```
    -0.5*width_motor_base, -0.5*width_motor_base,...(3,4)
    0.5*width_motor_base, 0.5*width_motor_base,...(5,6)
    0.5*width_motor_base, 0.5*width_motor_base,...(7,8)
    -0.5*width_motor_base ];  % (1)
X5 = zeros(size(X4));
Y5 = zeros(size(Y4));
X = [ X5, X4, X4, X3b, X3a, X2, X1, X0] +0.7*Lb_s;
Y = [ Y5, Y4, Y4, Y3b, Y3a, Y2, Y1, Y0];
HEIGHT = [ 0, thickness_motor_base, 0, 0,...
    0.5*width_motor_base-thickness_motor_base,...
    0.5*width_motor_base, 0 ];
MOTOR_SUP0 = CREAT_M_PLATE(...
    [0,0,-0.5*width_motor_base],...
    X,...
    Y,...
    HEIGHT,...
    MOTOR_SUP_color );
rotate( MOTOR_SUP0, [0,0,1], 90, [0,0,0] )
set( MOTOR_SUP0, 'Parent', ax_xyz, 'Visible','off')
V_MOTOR_BASE = cell(3,1);
for i=1:3
    MOTOR_MOUNT(i) = copyobj(MOTOR_SUP0,gca);

set(MOTOR_MOUNT(i),'Tag',['MOTOR_MOUNT',num2str(i)],'Visible','o
n')
    rotate( MOTOR_MOUNT(i), [0,0,1], psi(i), [0,0,0] )
    V_MOTOR_BASE{i} = get( MOTOR_MOUNT(i),'Vertices' );
    V_MOTOR_BASE{i}(:,1) = V_MOTOR_BASE{i}(:,1) +A{i}(1);
    V_MOTOR_BASE{i}(:,2) = V_MOTOR_BASE{i}(:,2) +A{i}(2);
    switch 1
        case 1  % Installation face down
            V_MOTOR_BASE{i}(:,3) = V_MOTOR_BASE{i}(:,3)
+A{i}(3);
        case 2  % Installation face up
            V_MOTOR_BASE{i}(:,3) = -V_MOTOR_BASE{i}(:,3) -
A{i}(3);
    end
    set( MOTOR_MOUNT(i), 'Vertices',V_MOTOR_BASE{i} )
end
%% --- MOTOR (Servo motor) ------------------------------------
-----------
shaft_radius = 6;
corner_radius = 10;

N4 = 8;
N8 = N4/2;
points_per_loop = 4*N4; % 4*100;

RL = 20;
ML = 30;

COS_M = cosd( linspace(-135, 225, points_per_loop ) );
SIN_M = sind( linspace(-135, 225, points_per_loop ) );
X0 = zeros( 1,points_per_loop );
Y0 = X0;
X1 = shaft_radius*COS_M;
```

```
Y1 = shaft_radius*SIN_M;
X2 = X1;
Y2 = Y1;
X3 = RL*COS_M;
Y3 = RL*SIN_M;
X4 = X3;
Y4 = Y3;
X5 = [ corner_radius*cosd(linspace(45,0,N8))-ML, ML-
corner_radius,...
    corner_radius*cosd(linspace(180,135,N8-1))+ML,...
    corner_radius*cosd(linspace(135,90,N8))+ML, ML,...
    corner_radius*cosd(linspace(270,225,N8-1))+ML,...
    corner_radius*cosd(linspace(225,180,N8))+ML, -
ML+corner_radius,...
    corner_radius*cosd(linspace(0,-45,N8-1))-ML,...
    corner_radius*cosd(linspace(-45,-90,N8))-ML, -ML,...
    corner_radius*cosd(linspace(90,45,N8-1))-ML ];
Y5 = [ corner_radius*sind(linspace(45,0,N8))-ML, -ML,...
    corner_radius*sind(linspace(180,135,N8-1))-ML,...
    corner_radius*sind(linspace(135,90,N8))-ML,   ML-
corner_radius,...
    corner_radius*sind(linspace(270,225,N8-1))+ML,...
    corner_radius*sind(linspace(225,180,N8))+ML,  ML,...
    corner_radius*sind(linspace(0,-45,N8-1))+ML,...
    corner_radius*sind(linspace(-45,-90,N8))+ML, -
ML+corner_radius,...
    corner_radius*sind(linspace(90,45,N8-1))-ML ];
X6 = X5;
Y6 = Y5;

X7 = [ linspace(-ML,ML,N4), ML*ones(1,N4),...
    linspace(ML,-ML,N4), -ML*ones(1,N4) ];
Y7 = [ -ML*ones(1,N4), linspace(-ML,ML,N4),...
    ML*ones(1,N4), linspace(ML,-ML,N4) ];
X8 = X7;
Y8 = Y7;
X9 = X5;
Y9 = Y5;
X10 = X5;
Y10 = Y5;
X11 = RL*COS_M;
Y11 = RL*SIN_M;
X12 = zeros(1,points_per_loop);
Y12 = zeros(1,points_per_loop);
X = [ X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12 ];
Y = [ Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8, Y9, Y10, Y11, Y12 ];
HEIGHT = [ 0,  0.7*Lb_s-0.1,  0, 3,  0,  5, 0, 80, 0,  15,  0,
0 ];
MOTOR0 = CREAT_M_PLATE(...
    [ 0, 0, 0 ],...
    X,...
    Y,...
    HEIGHT,...
    MOTOR_color );
rotate( MOTOR0, [1,0,0], -90, [0,0,0] )
set( MOTOR0, 'Parent',ax_xyz, 'Visible','off' )
```

```
V_MOTOR = cell(3,1);
for i=1:3
    MOTOR(i) = copyobj( MOTOR0, gca );
    set(MOTOR(i),'Tag',['MOTOR',num2str(i)],'Visible','on')
    rotate( MOTOR(i), [0,0,1], psi(i), [0,0,0] )
    V_MOTOR{i} = get( MOTOR(i),'Vertices' );
    V_MOTOR{i}(:,1) = V_MOTOR{i}(:,1) +A{i}(1);
    V_MOTOR{i}(:,2) = V_MOTOR{i}(:,2) +A{i}(2);
    V_MOTOR{i}(:,3) = V_MOTOR{i}(:,3) +A{i}(3);
    set( MOTOR(i), 'Vertices',V_MOTOR{i} )
end
%% --- PLATE_A ------------------------------------------------
-----------
N  = 20;
N2 = N+N;
N3 = N2+N;
N4 = N3+N;
Rmax = 1.05*norm(A{1}+A{2}-A{3});
Rmid = 0.38*Rmax;
Rmin = 0.25*Rmid;
%------------------------------------------------------------------
-----------
Pa = cell(3,1);
Pb = cell(3,1);
Pc = cell(3,1);
Pd = cell(3,1);
for i1=1:3
    Pa{i1} = M{i1}*[ Rmax;  0.7*Lb_s; 0 ];
    Pb{i1} = M{i1}*[ Rmid;  0.7*Lb_s; 0 ];
    Pc{i1} = M{i1}*[ Rmid; -0.7*Lb_s; 0 ];
    Pd{i1} = M{i1}*[ Rmax; -0.7*Lb_s; 0 ];
end
% text(Pa{1}(1),Pa{1}(2),Pa{1}(3),'a','Color','m')
% text(Pb{1}(1),Pb{1}(2),Pb{1}(3),'b','Color','m')
% text(Pc{1}(1),Pc{1}(2),Pc{1}(3),'c','Color','m')
% text(Pd{1}(1),Pd{1}(2),Pd{1}(3),'d','Color','m')
th_1a = atan2d( Pa{1}(2),  Pa{1}(1) );
th_2d = atan2d( Pd{2}(2),  Pd{2}(1) );
%------------------------------------------------------------------
-----------
X1 = Pa{1}(1)*cosd( linspace( th_2d, th_1a, N ) )';
X2 = linspace( Pb{1}(1), Pc{2}(1), N )';
X3 = X1;
X4 = X2;
Y1 = Pa{1}(1)*sind( linspace( th_2d, th_1a, N))';
Y2 = linspace( Pb{1}(2), Pc{2}(2), N )';
Y3 = Y1;
Y4 = Y2;
X = [ X1; X2; X3; X4 ];
Y = [ Y1; Y2; Y3; Y4 ];
Z = [ zeros(N2,1); thickness_PLATE*ones(N2,1) ]...
    -0.5*width_motor_base -thickness_PLATE;
F1 = [ (1:N-1)', (2:N)', (N2-1:-1:N+1)', (N2:-1:N+2)' ];
F2 = [ (1:N-1)', (N2+1:N3-1)', (N2+2:N3)', (2:N)' ];
F3 = [ (1:N-1)', (2:N)', (N2-1:-1:N+1)', (N2:-1:N+2)' ]+N2;
F4 = [ 1, N2, N4, N2+1 ];
```

```
F5 = [ N, N+1, N3+1, N3 ];
F = [ F1; F2; F3;  F4; F5 ];
%-------------------------------------------------------------------
-----------
PLATE_A0 = patch(...
    'Faces',F,...
    'Vertices',[X,Y,Z],...
    'FaceColor',PLATE_A_color,...
    'EdgeColor','none' );
V_PLATE_A0 = get( PLATE_A0, 'Vertices' );
V_PLATE_A0(:,3) = V_PLATE_A0(:,3);
set( PLATE_A0, 'Vertices', V_PLATE_A0 )
set( PLATE_A0, 'Parent',ax_xyz, 'Visible','off' )
PLATE_A = zeros(3,1);
for i1=1:3
    PLATE_A(i1) = copyobj( PLATE_A0, gca );
    set( PLATE_A(i1), 'Visible','on', 'FaceAlpha',face_alpha )
    rotate( PLATE_A(i1), [0,0,1], psi(i1), [0,0,0] )
end
%-------------------------------------------------------------------
-----------
approxi_dis = ( norm(Pb{1}-Pc{2}) +norm(Pc{2}-Pb{2}) )/N2;
Nbc = round( norm(Pb{1}-Pc{2})/approxi_dis );
Ncb = round( norm(Pc{2}-Pb{2})/approxi_dis );
X1 = linspace( Pb{1}(1), Pc{2}(1), Nbc )';
X2 = linspace( Pc{2}(1), Pb{2}(1), Ncb )';
X3 = linspace( Pb{2}(1), Pc{3}(1), Nbc )';
X4 = linspace( Pc{3}(1), Pb{3}(1), Ncb )';
X5 = linspace( Pb{3}(1), Pc{1}(1), Nbc )';
X6 = linspace( Pc{1}(1), Pb{1}(1), Ncb )';
X7 = Rmin*cosd( linspace(th_1a,th_1a+360,3*(Nbc+Ncb)-5) )';
Y1 = linspace( Pb{1}(2), Pc{2}(2), Nbc )';
Y2 = linspace( Pc{2}(2), Pb{2}(2), Ncb )';
Y3 = linspace( Pb{2}(2), Pc{3}(2), Nbc )';
Y4 = linspace( Pc{3}(2), Pb{3}(2), Ncb )';
Y5 = linspace( Pb{3}(2), Pc{1}(2), Nbc )';
Y6 = linspace( Pc{1}(2), Pb{1}(2), Ncb )';
Y7 = Rmin*sind( linspace(th_1a,th_1a+360,3*(Nbc+Ncb)-5) )';
X = [ X1(1:end-1); X2(1:end-1); X3(1:end-1);...
    X4(1:end-1); X5(1:end-1); X6 ];
Y = [ Y1(1:end-1); Y2(1:end-1); Y3(1:end-1);...
    Y4(1:end-1); Y5(1:end-1); Y6 ];
XX = [ X; X7; X7; X ];
YY = [ Y; Y7; Y7;  Y];
PLATE_A_center = CREAT_M_PLATE(...
    [0,0,-0.5*width_motor_base],...
    XX,...
    YY,...
    [0, -thickness_PLATE, 0 ],...
    PLATE_A_color );
set( PLATE_A_center, 'Visible','on', 'FaceAlpha',face_alpha )
% set( PLATE_A_center, 'EdgeColor', [1,0,0] )
%% --- COVER -------------------------------------------------------
-----------
height_cover = 1.2*width_motor_base;
r0   = 0.25*height_cover;
```

```
Rmax = Pa{1}(1);
Xin  = linspace( Pc{2}(1), Pb{1}(1), N )';
Yin  = linspace( Pc{2}(2), Pb{1}(2), N )';
COS_TH = cosd( linspace( th_2d, th_1a, N ) )';
SIN_TH = sind( linspace( th_2d, th_1a, N ) )';
X1  = Rmax*COS_TH;
deg = 0:10:90;
Ri  = Rmax-r0 +r0*cosd(deg)';
X   = X1;
for i1=1:length(deg)
    X = [ X; Ri(i1)*COS_TH ];
end
X = [ X; Xin ];
%-----------------------------------------------------------------
-----------
Y1 = Rmax*SIN_TH;
Y = Y1;
for i1=1:length(deg)
    Y = [ Y; Ri(i1)*SIN_TH ];
end
Y = [ Y; Yin ];
%-----------------------------------------------------------------
-----------
HEIGHT = [0, height_cover-r0, height_cover-r0+r0*sind(10:10:90),
height_cover ]';
Z = zeros( size(X1) );
F = [(1:N-1)',(2:N)',(N+2:N2)',(N+1:N2-1)'];
for i1=1:length(HEIGHT)-1
    Z = [ Z; HEIGHT(i1+1)*ones(N,1) ];
    Ni1 = N*(i1-1);
    F = [ F; [(1:N-1)',(2:N)',(N+2:N2)',(N+1:N2-1)']+Ni1];
end
Z = Z -0.5*width_motor_base;
for i1=2:length(HEIGHT)-1
    iN = i1*N;
    Y(iN) = Pa{1}(2);
end
dxy = 0.5*(Pb{2}-Pc{2});
xc = dxy(1);
yc = dxy(2);
XR = X(N:N:length(HEIGHT)*N);
YR = Y(N:N:length(HEIGHT)*N) -0.5*norm(Pb{1}(2)-Pc{1}(2));
ZR = Z(N:N:length(HEIGHT)*N);
for i1=1:length(XR)
    iN = i1*N;
    xy = [cosd(120),-sind(120); sind(120),cosd(120)]*[ XR(i1);
YR(i1) ];
    X(iN-N+1) = xy(1)-xc;
    Y(iN-N+1) = xy(2)-yc;
end
%-----------------------------------------------------------------
-----------
hold on
COVER_Uab1 = fill3(...
    [ XR; Pb{1}(1); XR(1) ],...
    [ YR; 0; 0 ] +Pb{1}(2)-1,...
```

```
    [ ZR; ZR(1); ZR(1) ],...
    COVER_color );
set( COVER_Uab1,...

'EdgeColor',COVER_color*0.75,...'EdgeColor','y','LineWidth',1,..
.
    'FaceColor',COVER_color,...
    'FaceAlpha',COVER_face_alpha )
COVER_Uab2 = copyobj( COVER_Uab1, gca );
rotate( COVER_Uab2, [0,0,1], 120, [0,0,0] )
COVER_Uab3 = copyobj( COVER_Uab1, gca );
rotate( COVER_Uab3, [0,0,1], -120, [0,0,0] )
%-------------------------------------------------------------------
-----------
COVER_Ucd1 = fill3(...
    [ XR; Pb{1}(1); XR(1)],...
    [ YR; 0; 0 ] +Pc{1}(2)+1,...
    [ ZR; ZR(1); ZR(1) ],...
    COVER_color );
set( COVER_Ucd1,...

'EdgeColor',COVER_color*0.75,...'EdgeColor','m','LineWidth',1,..
.
    'FaceColor',COVER_color,...
    'FaceAlpha',COVER_face_alpha )
COVER_Ucd2 = copyobj( COVER_Ucd1, gca );
rotate( COVER_Ucd2, [0,0,1], 120, [0,0,0] )
COVER_Ucd3 = copyobj( COVER_Ucd1, gca );
rotate( COVER_Ucd3, [0,0,1], -120, [0,0,0] )
%-------------------------------------------------------------------
-----------
COVER_Ubc1 = fill3(...
    [ Pb{1}(1), Pb{1}(1), Pc{1}(1), Pc{1}(1)],...
    [ Pb{1}(2), Pb{1}(2), Pc{1}(2), Pc{1}(2)],...
    [ 0, height_cover, height_cover, 0]-0.5*width_motor_base,...
    COVER_color );
set( COVER_Ubc1,...

'EdgeColor',COVER_color*0.75,...'EdgeColor','w','LineWidth',1,..
.
    'FaceColor',COVER_color,...
    'FaceAlpha',COVER_face_alpha )
COVER_Ubc2 = copyobj( COVER_Ubc1, gca );
rotate( COVER_Ubc2, [0,0,1], 120, [0,0,0] )
COVER_Ubc3 = copyobj( COVER_Ubc1, gca );
rotate( COVER_Ubc3, [0,0,1], -120, [0,0,0] )
%-------------------------------------------------------------------
-----------
COVER1 = patch(...
    'Faces',F,...
    'Vertices',[X,Y,Z],...
    'FaceColor',COVER_color,...
    'EdgeColor','none',...'EdgeColor',COVER_color,...
    'Visible','on' );
set( COVER1, 'FaceAlpha', COVER_face_alpha )
COVER2 = copyobj( COVER1, gca);
```

```
rotate( COVER2, [0,0,1],120,[0,0,0])
COVER3 = copyobj( COVER1, gca);
rotate( COVER3, [0,0,1],-120,[0,0,0])
%-------------------------------------------------------------------
-----------
X = [ Pc{1}(1); Pb{1}(1); Pc{2}(1); Pb{2}(1); Pc{3}(1);
Pb{3}(1); Pc{1}(1);...
    zeros(7,1) ];
Y = [ Pc{1}(2); Pb{1}(2); Pc{2}(2); Pb{2}(2); Pc{3}(2);
Pb{3}(2); Pc{1}(2);...
    zeros(7,1) ];
Z = (height_cover-0.5*width_motor_base)*ones( size(X) );
F = [...
    1,2,13,14;
    2,3,12,13;
    3,4,11,12;
    4,5,10,11;
    5,6, 9,10;
    6,7, 8, 9 ];
COVER0 = patch(...
    'Faces',F,...
    'Vertices',[X,Y,Z],...
    'FaceColor',COVER_color,...
    'EdgeColor','none',...'EdgeColor','g',...
    'Visible','on' );
set( COVER0, 'FaceAlpha', COVER_face_alpha )
hold off
%% --- Set 3D visual conditions ------------------------------
-----------
camlight right
material metal
lighting gouraud
set(gcf,'Renderer','OpenGL');
drawnow
end  % END: function CREAT_FIGURE1

%% === function: CREAT_FIGURE2 --> Time responses
=========================
function CREAT_FIGURE2
global Line_Theta1 Line_Theta2 Line_Theta3
global Line_X Line_Y Line_Z
global line_width
global path_color
global marker_size marker_color
fig2 = figure;
set( fig2,...
    'Tag','fig2',...
    'NumberTitle','off',...
    'Name','Time Responses',...
    'BackingStore','off',...
    'ToolBar','figure',...
    'Color',[ 0.85, 0.95, 0.85 ],...
    'Unit','Normalized',...
    'OuterPosition',[0.533, 0.04, 0.472, 0.96] );
%--- Creat axes objects ------------------------------------
-----------
```

```
ax_theta1 = subplot(321);
h321 = ylabel('\theta_{1}');
h321.FontSize = 12;
h321.FontName = 'Times New Roman';
h321.Color = 'b';
h321.Rotation = 0;
h321.HorizontalAlignment = 'right';
h321.VerticalAlignment = 'middle';
grid on
set( ax_theta1,...
    'Parent',fig2,...
    'Tag','ax_theta1',...
    'XLim',[0,inf],...
    'FontName','Times New Roman',...
    'FontSize',12,...
    'Box','on');
Line_Theta1 = animatedline(...
    'Parent',ax_theta1,...
    'Tag','Line_Theta1',...
    'Color',path_color,...
    'Marker','.',...
    'MarkerSize',marker_size,...
    'MarkerEdgeColor',marker_color,...
    'LineWidth',line_width);
ax_theta2 = subplot(323);
h323 = ylabel('\theta_{2}');
h323.FontSize = 12;
h323.FontName = 'Times New Roman';
h323.Color = 'b';
h323.Rotation = 0;
h323.HorizontalAlignment = 'right';
h323.VerticalAlignment = 'middle';
grid on
set( ax_theta2,...
    'Parent',fig2,...
    'Tag','ax_theta2',...
    'XLim',[0,inf],...
    'FontName','Times New Roman',...
    'FontSize',12,...
    'Box','on');
Line_Theta2 = animatedline(...
    'Parent',ax_theta2,...
    'Tag','Line_Theta2',...
    'Color',path_color,...
    'Marker','.',...
    'MarkerSize',marker_size,...
    'MarkerEdgeColor',marker_color,...
    'LineWidth',line_width);
ax_theta3 = subplot(325);
h325 = ylabel('\theta_{3}');
h325.FontSize = 12;
h325.FontName = 'Times New Roman';
h325.Color = 'b';
h325.Rotation = 0;
h325.HorizontalAlignment = 'right';
h325.VerticalAlignment = 'middle';
```

```
grid on
set( ax_theta3,...
    'Parent',fig2,...
    'Tag','ax_theta3',...
    'FontName','Times New Roman',...
    'XLim',[0,inf],...
    'FontSize',12,...
    'Box','on');
Line_Theta3 = animatedline(...
    'Parent',ax_theta3,...
    'Tag','Line_Theta3',...
    'Color',path_color,...
    'Marker','.',...
    'MarkerSize',marker_size,...
    'MarkerEdgeColor',marker_color,...
    'LineWidth',line_width);
h325x = xlabel('Time (s)');
h325x.FontSize = 12;
h325x.FontName = 'Times New Roman';
h325x.Color = 'b';
%-------------------------------------------------------------------
-----------
ax_x = subplot(322);
h322 = ylabel('\it{x}');
h322.FontSize = 12;
h322.FontName = 'Times New Roman';
h322.Color = 'b';
h322.Rotation = 0;
h322.HorizontalAlignment = 'left';
h322.VerticalAlignment = 'middle';
grid on
set( ax_x,...
    'Parent',fig2,...
    'Tag','ax_x',...
    'FontName','Times New Roman',...
    'YAxisLocation','right',...
    'XLim',[0,inf],...
    'FontSize',12,...
    'Box','on');
Line_X = animatedline(...
    'Parent',ax_x,...
    'Tag','Line_X',...
    'Color',path_color,...
    'Marker','.',...
    'MarkerSize',marker_size,...
    'MarkerEdgeColor',marker_color,...
    'LineWidth',line_width);
ax_y = subplot(324);
h324 = ylabel('\it{y}');
h324.FontSize = 12;
h324.FontName = 'Times New Roman';
h324.Color = 'b';
h324.Rotation = 0;
h324.HorizontalAlignment = 'left';
h324.VerticalAlignment = 'middle';
grid on
```

```
set( ax_y,...
    'Parent',fig2,...
    'Tag','ax_y',...
    'FontName','Times New Roman',...
    'YAxisLocation','right',...
    'XLim',[0,inf],...
    'FontSize',12,...
    'Box','on');
Line_Y = animatedline(...
    'Parent',ax_y,...
    'Tag','Line_Y',...
    'Color',path_color,...
    'Marker','.',...
    'MarkerSize',marker_size,...
    'MarkerEdgeColor',marker_color,...
    'LineWidth',line_width);
ax_z = subplot(326);
h326 = ylabel('\it{z}');
h326.FontSize = 12;
h326.FontName = 'Times New Roman';
h326.Color = 'b';
h326.Rotation = 0;
h326.HorizontalAlignment = 'left';
h326.VerticalAlignment = 'middle';
grid on
set( ax_z,...
    'Parent',fig2,...
    'Tag','ax_z',...
    'FontName','Times New Roman',...
    'YAxisLocation','right',...
    'XLim',[0,inf],...
    'FontSize',12,...
    'Box','on');
Line_Z = animatedline(...
    'Parent',ax_z,...
    'Tag','Line_Z',...
    'Color',path_color,...
    'Marker','.',...
    'MarkerSize',marker_size,...
    'MarkerEdgeColor',marker_color,...
    'LineWidth',line_width);
h326x = xlabel('Time (s)');
h326x.FontSize = 12;
h326x.FontName = 'Times New Roman';
h326x.Color = 'b';
%-------------------------------------------------------------------
-----------
drawnow
end  % END: function CREAT_FIGURE2

%% === function: CREAT_FIGURE3 --> UI
====================================
function CREAT_FIGURE3
global trajectory
%--- Creat UI: fig3 ----------------------------------------------
-----------
```

```
fig3 = figure;
set( fig3,...
    'Tag','fig3',...
    'NumberTitle','off',...
    'Name','Menu',...
    'Resize','off',...
    'BackingStore','off',...
    'MenuBar','none',...
    'ToolBar','none',...
    'Color',[ 0.85, 0.95, 0.85 ] +0.05,...
    'Units','normalized',...
    'OuterPosition',[ 0.45, 0.74, 0.15, 0.25 ] );
%--- (1) pb_Run ------------------------------------------------
-----------
N  = 4;                         % There are N UIControl objects
X0 = 0.05;
Y0 = 0.05;
Ygap = 0.05;
DX = 1 -2*X0;                   % Width of a UIControl object
DY = (1-(N-1)*Ygap-2*Y0)/N;     % Height of a UIControl object
my_fontsize = 12;
uicontrol( fig3,...
    'Style','pushbutton',...
    'Tag','pb_Run',...
    'Units','Normalized',...
    'Position',[X0, Y0+(N-1)*(DY+Ygap), DX, DY],...
    'FontSize',my_fontsize,...
    'String','Run',...
    'CallBack',@CB_pb_Run );
%--- (2) pb_ResetRobot -----------------------------------------
-----------
uicontrol( fig3,...
    'Style','pushbutton',...
    'Tag','pb_ResetRobot',...
    'Enable','on',...
    'Units','Normalized',...
    'Position',[X0, Y0+(N-2)*(DY+Ygap), DX, DY],...
    'FontSize',my_fontsize,...
    'String','Reset Robot',...
    'CallBack',@CB_pb_ResetRobot );
%--- (3) pop_SelectTrajectory ----------------------------------
-----------
uicontrol( fig3,...
    'Tag','pop_SelectTrajectory',...
    'Style','popupmenu',...
    'String',[...
    ' (1) Vertical line | ',...
    ' (2) Upper disk | ',...
    ' (3) Lower disk | ',...
    ' (4) Upper and lower circles | ',...
    ' (5) Cylinder | ',...
    ' (6) Sphere | ',...
    ' (7) Trapezoidal velocity trajectory' ],...
    'Value',trajectory,...
    'Units','Normalized',...
    'Position',[X0, Y0+(N-3)*(DY+Ygap), DX, DY],...
```

```
    'FontSize',my_fontsize,...
    'CallBack',@CB_pop_SelectTrajectory );
%--- (4) pb_End ------------------------------------------------
-----------
uicontrol( fig3,...
    'Tag','pb_End',...
    'Style','pushbutton',...
    'Units','Normalized',...
    'Position',[X0, Y0, DX, DY],...
    'FontSize',my_fontsize,...
    'String','End',...
    'CallBack',@CB_pb_End );
drawnow
end

%% === function: SetTrajectory
=========================================
function [Pref, m_loops]=SetTrajectory(trajectory)
global Xref Yref Zref path_color
global Zinit
set( findobj('Tag','ax_theta1'), 'XLim',[0,inf] );
set( findobj('Tag','ax_theta2'), 'XLim',[0,inf] );
set( findobj('Tag','ax_theta3'), 'XLim',[0,inf] );
set( findobj('Tag','ax_x'), 'XLim',[0,inf] );
set( findobj('Tag','ax_y'), 'XLim',[0,inf] );
set( findobj('Tag','ax_z'), 'XLim',[0,inf] );
set( findobj('Tag','Line_Theta1'), 'color',path_color )
set( findobj('Tag','Line_Theta2'), 'color',path_color )
set( findobj('Tag','Line_Theta3'), 'color',path_color )
set( findobj('Tag','Line_X'), 'color', path_color )
set( findobj('Tag','Line_Y'), 'color', path_color )
set( findobj('Tag','Line_Z'), 'color', path_color )
%-----------------------------------------------------------------
-----------
%  The default initial position of the end-effector is
%  [ Zinit, Yinit, Zinit ] = [ 0, 0, -sqrt( Lb^2 -(La+Lr-Lh)^2
].
%-----------------------------------------------------------------
-----------
switch trajectory
    case 1  % Vertical line
        z1 = 100;
        z2 = -150;

        n1 = 15;
        X1 = zeros( n1, 1 );
        Y1 = zeros( n1, 1 );
        Z1 = linspace( 0, z1, n1 )';

        n2 = 42;
        X2 = zeros( n2, 1 );
        Y2 = zeros( n2, 1 );
        Z2 = linspace( z1, z2, n2 )';

        n3 = 27;
        X3 = zeros( n3, 1 );
```

```
        Y3 = zeros( n3, 1 );
        Z3 = linspace( z2, 0, n3 )';

        X = [ X1; X2(2:end); X3(2:end) ];
        Y = [ Y1; Y2(2:end); Y3(2:end) ];
        Z = [ Z1; Z2(2:end); Z3(2:end) ];

    case 2  % Upper disk
        z  = 100;  % Z coordinate of the upper disc
        R  = 200;  % Radius of the upper disc
        ds = 20;
        dr = 20;

        n1 = ceil(abs(z/ds));
        X1 = zeros(n1,1);
        Y1 = zeros(n1,1);
        Z1 = linspace( 0, z, n1 )';

        Xr = 0;
        Yr = 0;
        for r=(R-floor(R/dr)*dr+dr):dr:R
            C = 2*pi*r;
            n = ceil(C/ds);
            TH = linspace( 0, 360, n )';
            Xr = [ Xr; r*cosd(TH) ];
            Yr = [ Yr; r*sind(TH) ];
        end
        Zr = zeros( size(Xr) ) +Z1(end);

        n2 = ceil((sqrt(R^2+z^2))/ds);
        X2 = linspace( R, 0, n2 )';
        Y2 = zeros(n2,1);
        Z2 = linspace( z, 0, n2 )';

        X = [ X1; Xr(2:end); X2(2:end) ];
        Y = [ Y1; Yr(2:end); Y2(2:end) ];
        Z = [ Z1; Zr(2:end); Z2(2:end) ];

    case 3  % Lower disk
        z  = -150;  % Z coordinate of the lower disc
        R  = 200;   % Radius of the lower disc
        ds = 20;
        dr = 20;

        n1 = ceil(abs(z/ds));
        X1 = zeros(n1,1);
        Y1 = zeros(n1,1);
        Z1 = linspace( 0, z, n1 )';

        Xr = 0;
        Yr = 0;
        for r=(R-floor(R/dr)*dr+dr):dr:R
            C = 2*pi*r;
            n = ceil(C/ds);
            TH = linspace( 0, 360, n )';
            Xr = [ Xr; r*cosd(TH) ];
```

```
            Yr = [ Yr; r*sind(TH) ];
        end
        Zr = zeros( size(Xr) ) +Z1(end);

        n2 = ceil((sqrt(R^2+z^2))/ds);
        X2 = linspace( R, 0, n2 )';
        Y2 = zeros(n2,1);
        Z2 = linspace( z, 0, n2 )';

        X = [ X1; Xr(2:end); X2(2:end) ];
        Y = [ Y1; Yr(2:end); Y2(2:end) ];
        Z = [ Z1; Zr(2:end); Z2(2:end) ];

    case 4  % Upper and lower circles
        R  = 250;   % Radius of the circles
        z1 = 100;   % Z coordinate of the upper circle
        z2 = -150;  % Z coordinate of the lower circle
        % Elevation1 = atan2d( z1, R )
        % Elevation2 = atan2d( z2, R )

        n = 60;
        Azimuth = linspace(0,360,n)';
        XR = R*cosd( Azimuth );
        YR = R*sind( Azimuth );

        n1 = 15;
        X1 = linspace( 0, R, n1 )';
        Y1 = zeros(n1,1);
        Z1 = linspace( 0, z1, n1 )';

        X2 = XR;
        Y2 = YR;
        Z2 = zeros(n,1) +z1;

        n2 = 20;
        X3 = R*ones(n2,1);
        Y3 = zeros(n2,1);
        Z3 = linspace( z1, z2, n2 )';

        X4 = XR;
        Y4 = YR;
        Z4 = zeros(n,1) +z2;

        n3 = 20;
        X5 = linspace( R, 0, n3 )';
        Y5 = zeros(n3,1);
        Z5 = linspace( z2, 0, n3 )';

        X = [X1; X2(2:end); X3(2:end); X4(2:end); X5(2:end)];
        Y = [Y1; Y2(2:end); Y3(2:end); Y4(2:end); Y5(2:end)];
        Z = [Z1; Z2(2:end); Z3(2:end); Z4(2:end); Z5(2:end)];

    case 5  % Cylinder
        R  = 260;   % Radius of the cylinder
        z1 = -200;  % z coordinate of the lower circle
        z2 = 150;   % z coordinate of the upper circle
```

```
        n = 60;
        Azimuth = linspace(0,360,n)';
        XR0 = R*cosd( Azimuth );
        YR0 = R*sind( Azimuth );
        ZR0 = zeros( size(XR0) );

        n1 = 30;
        X1 = linspace( 0, R, n1 )';
        Y1 = zeros(n1,1);
        Z1 = linspace( 0, z1, n1 )';

        nr = 20;
        Zh = linspace( z1, z2, nr );
        XR = XR0;
        YR = YR0;
        ZR = ZR0 +Zh(1);
        for i=2:nr
            XR = [XR; XR0];
            YR = [YR; YR0];
            ZR = [ZR; ZR0+Zh(i)];
        end

        n2 = 15;
        X2 = linspace( R, 0, n2 )';
        Y2 = zeros(n2,1);
        Z2 = linspace( z2, 0, n2 )';

        X = [X1; XR; X2];
        Y = [Y1; YR; Y2];
        Z = [Z1; ZR; Z2];

    case 6  % Sphere
        R  = 130;   % Radius of the sphere

        n1 = 21;
        X1 = zeros(n1,1);
        Y1 = zeros(n1,1);
        Z1 = linspace( 0, -2*R, n1 )';
        X2 = zeros(n1,1);
        Y2 = zeros(n1,1);
        Z2 = flip( Z1 );

        n = 30;
        TH = linspace( 0, 360, n )';
        XR = cosd(TH);
        YR = sind(TH);
        Xr = 0;
        Yr = 0;
        Zr = 0;
        for i=2:(n1-1)
            Xr = [ Xr; XR*sqrt( R^2 -(Z2(i)+R)^2 ) ];
            Yr = [ Yr; YR*sqrt( R^2 -(Z2(i)+R)^2 ) ];
            Zr = [ Zr; ones(n,1)*Z2(i) ];
        end
        X = [ X1; Xr(2:end); 0 ];
        Y = [ Y1; Yr(2:end); 0 ];
```

```
        Z = [ Z1; Zr(2:end); 0 ];

    case 7  % Trapezoidal velocity trajectory
        distance = 250;  % max: 385
        m0_loops = 100;
        Ta = round(m0_loops*0.3);  % ยค[(ยครฎ)ยณtยชยร—ยฐjยฐรณยผร?
        Vmax = distance/(m0_loops-Ta);
        acceleration =  Vmax/Ta;
        deceleration = -acceleration;
        Vacc = 0:acceleration:Vmax;
        Vdec = Vmax:deceleration:0;
        Vconst = Vmax*ones( 1, m0_loops -length(Vacc) -
length(Vdec) );
        Vxyz = [ Vacc, Vconst, Vdec ];
        Sxyz = zeros( length(Vxyz), 1 );
        dt = 1;     % This value must be corrected to the actual
value
        %             during the experiment. Others such as
acceleration,
        %             velocity, displacement, etc. are also the
same.
        for i=2:m0_loops
            Sxyz(i) = Sxyz(i-1) +Vxyz(i-1)*dt;
        end
        Azimuth = 240;    %   0 ~ 360
        Elevation = -45;  % -45 ~ 20
        X = cosd( Elevation )*cosd( Azimuth )*Sxyz;
        Y = cosd( Elevation )*sind( Azimuth )*Sxyz;
        Z = sind( Elevation )*Sxyz;
end
Xref = X;
Yref = Y;
Zref = Z +Zinit;
m_loops = length( Xref );
Pref = [ Xref, Yref, Zref ];
end

%% === function:
CB_pb_Run================================================
function CB_pb_Run(src,event)
global k0
global ploting
global Line_P
global Line_Theta1 Line_Theta2 Line_Theta3
global Line_X Line_Y Line_Z
set( findobj('Tag','text_error'),'Visible','off')
if ploting==1
    ploting=0;
    set(findobj('Tag','pb_Run'),'String','Continue')
else
    ploting=1;
    set(findobj('Tag','pb_Run'),'String','Pause')
end
if k0==0
    clearpoints( Line_P )
    clearpoints( Line_Theta1 )
```

```
    clearpoints( Line_Theta2 )
    clearpoints( Line_Theta3 )
    clearpoints( Line_X )
    clearpoints( Line_Y )
    clearpoints( Line_Z )
end
end

%% === function: CB_pb_ResetRobot
=======================================
function CB_pb_ResetRobot(src,event)
global k0 ploting
global Line_P
global Line_Theta1 Line_Theta2 Line_Theta3
global Line_X Line_Y Line_Z
global LINK_A LINK_B LINK_Bs PLATE_C CONNECTER_PC
global BALLJOINT_B BALLJOINT_C
global V_LINK_A0 V_LINK_B0 V_LINK_Bs0 V_PLATE_C0 V_CONNECTER_PC0
global V_BALLJOINT_B0 V_BALLJOINT_C0
k0 = 0;
ploting = 0;
clearpoints( Line_P )
clearpoints( Line_Theta1 )
clearpoints( Line_Theta2 )
clearpoints( Line_Theta3 )
clearpoints( Line_X )
clearpoints( Line_Y )
clearpoints( Line_Z )
set( findobj('Tag','text_error'),'Visible','off')
set( findobj('Tag','pb_Run'), 'string','Run','Enable','on')
for i=1:3
    %--- LINK_A ------------------------------------------------
-----------
    set( LINK_A(i), 'Vertices', V_LINK_A0{i} );
    % --- CONNECTOR_PC ----------------------------------------
-----------
    set( CONNECTER_PC(i), 'Vertices', V_CONNECTER_PC0{i} )
    %--- LINK_B (Parallel four-bar linkage) --------------------
-----------
    for j=1:2
        %--- LINK_B (Parallel four-bar linkage: long side) -----
-----------
        set( LINK_B(i,j), 'Vertices', V_LINK_B0{i,j} )
        %--- LINK_Bs (Parallel four-bar linkage: short side) ---
------------
        set( LINK_Bs(i,j), 'Vertices', V_LINK_Bs0{i,j} )
        %--- BALLJOINT ---------------------------------------
-----------
        set( BALLJOINT_B(i,j), 'Vertices', V_BALLJOINT_B0{i,j} )
        set( BALLJOINT_C(i,j), 'Vertices', V_BALLJOINT_C0{i,j} )
        %-----------------------------------------------------
-----------
    end  % END: for j=1:2
end  % END: for i=1:3
%--- PLATE_C (Bottom tool mounting plate) ---------------------
-----------
```

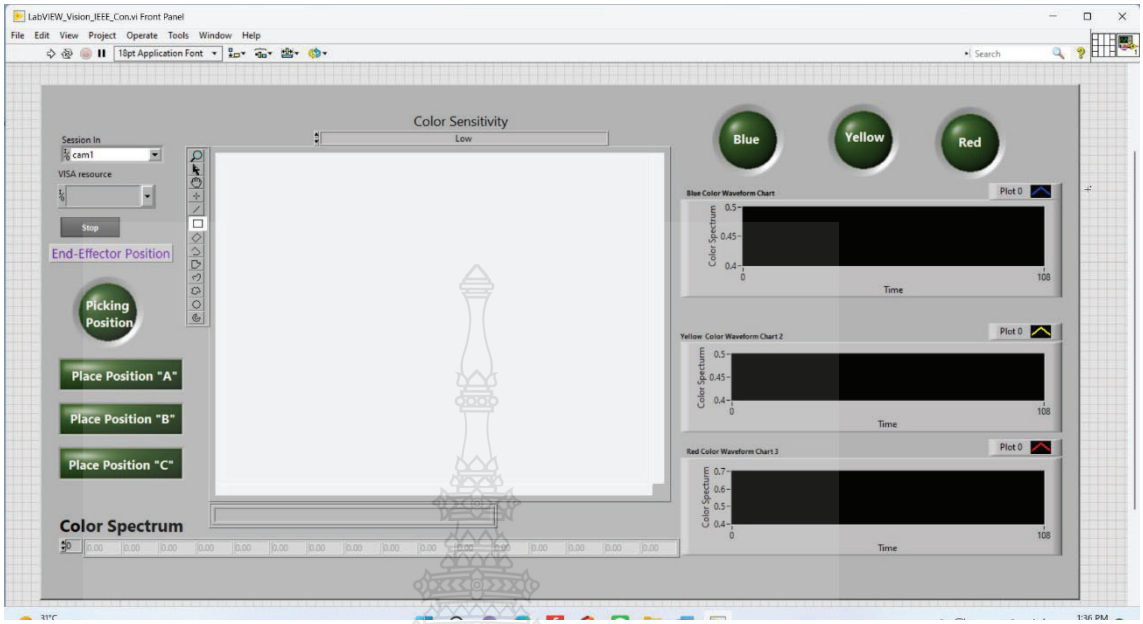
```
set( PLATE_C, 'Vertices', V_PLATE_C0 )
drawnow
end  % END function CB_ResetRobot(src,event)

%% === function: CB_pop_SelectTrajectory
==================================
function CB_pop_SelectTrajectory(src,event)
global trajectory Pref m_loops k0
trajectory = get( src,'Value' );
[Pref, m_loops] = SetTrajectory( trajectory );
k0 = 0;
end

%% === function: CB_pb_End
================================================
function CB_pb_End(src,event)
global end_program
end_program = 1;
end

%% === function: CREAT_CUBOID
============================================
function patch_obj=CREAT_CUBOID(Oxyz,Lx,Ly,Lz,...
    FaceColor,visible,EdgeColor)
% Oxyz㊣GThe coordinates of the lower left point on the bottom of
this cuboid
% Lx: The length of the side parallel to the x-axis
% Ly: The length of the side parallel to the y-axis
% Lz: The length of the side parallel to the z-axis
% FaceColor㊣GThe color of the surface of this 3D draphical
object
% visible㊣G'on' ㊣X Display the object;
%          'off' ㊣X Hide the object without deleting it.
% EdgeColor㊣GSee 'UIControl Properties'
if nargin<7  % Set the EdgeColor default value to be the same as
the cylindrical surface
    EdgeColor = 'none';
    if nargin<6  % Set the object visible by default
        visible = 'on';
        if nargin<5  % Set the default color of the object to
white
            FaceColor = ones(1,3);  % (The 3D graphics look gray
due to shadows)
        end
    end
end
x0 = Oxyz(1);
y0 = Oxyz(2);
z0 = Oxyz(3);
X  = [0, 0,Lx,Lx, 0, 0,Lx,Lx]'+x0;
Y  = [0,Ly,Ly, 0, 0,Ly,Ly,0 ]'+y0;
Z  = [0, 0, 0, 0,Lz,Lz,Lz,Lz]'+z0;
F  = [...
    1 2 3 4;
    5 1 4 8;
```

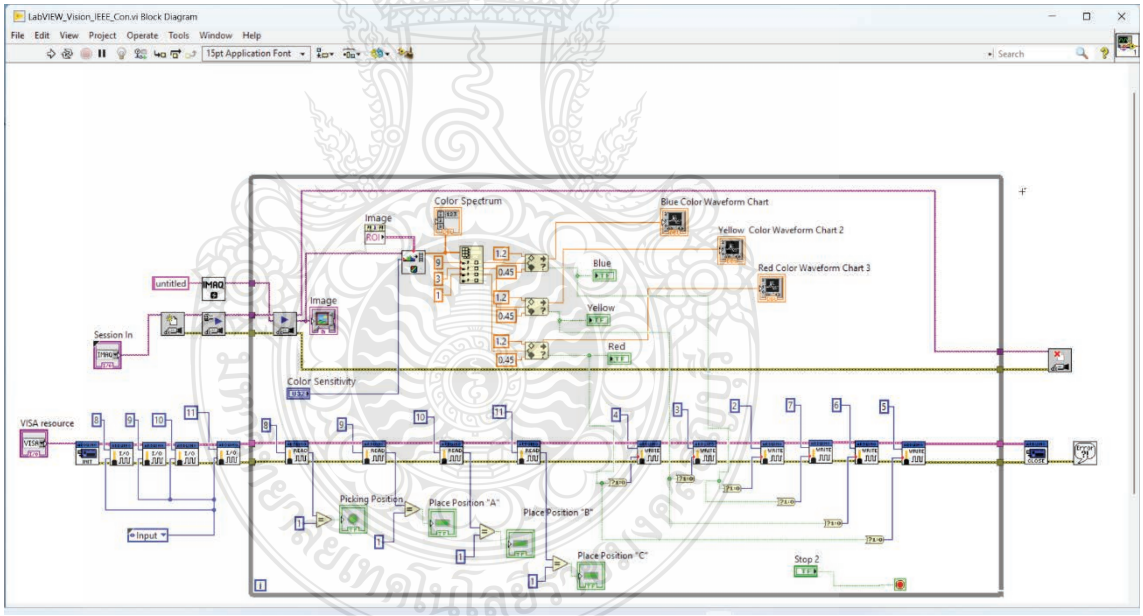
```
    5 6 2 1;
    2 6 7 3;
    4 3 7 8;
    5 6 7 8];
patch_obj = patch('Faces',F,'Vertices',[X,Y,Z],...

'FaceColor',FaceColor,'EdgeColor',EdgeColor,'Visible',visible );
end

%% === function: CREAT_CYLINDER
=========================================
function patch_obj=CREAT_CYLINDER(Oxyz,radius,height,...
    N_sides,FaceColor,visible,EdgeColor)
if nargin<7
    EdgeColor = 'none';
    if nargin<6
        visible = 'on';
        if nargin<5
            FaceColor = ones(1,3);
            if nargin<4
                N_sides = 32;
            end
        end
    end
end
Nh = length( height );
N  = Nh +1;
x0 = Oxyz(1);
y0 = Oxyz(2);
z0 = Oxyz(3);
dth = 2*pi/N_sides;
THETA = (0:dth:(N_sides-1)*dth)';
Nxm = N*N_sides;
X = zeros(Nxm,1);
Y = zeros(Nxm,1);
Z = zeros(Nxm,1);
F = zeros(Nxm-N_sides,4);
X(1:N_sides) = radius(1)*cos(THETA) +x0;
Y(1:N_sides) = radius(1)*sin(THETA) +y0;
Z(1:N_sides) = z0*ones(N_sides,1);
for i=2:N
    ixm = i*N_sides;
    X(ixm-N_sides+1:ixm) = radius(i)*cos(THETA)+x0;
    Y(ixm-N_sides+1:ixm) = radius(i)*sin(THETA)+y0;
    Z(ixm-N_sides+1:ixm) = ( z0 +sum(height(1:i-1))
)*ones(N_sides,1);
    F(ixm-N_sides-N_sides+1:ixm-N_sides,:) = [ (1:N_sides-1)',
(2:N_sides)', (N_sides+2:N_sides+N_sides)',
(N_sides+1:N_sides+N_sides-1)';...
        N_sides, 1, N_sides+1, N_sides+N_sides] +ixm-N_sides-
N_sides;
end
patch_obj = patch('Faces',F,'Vertices',[X,Y,Z],...

'FaceColor',FaceColor,'EdgeColor',EdgeColor,'Visible',visible );
end
```

```
%% === function: CREAT_M_PLATE
=========================================
function patch_obj=CREAT_M_PLATE(Oxyz,X,Y,height,...
    FaceColor,visible,EdgeColor)
if nargin<7
    EdgeColor = 'none';
    if nargin<6
        visible = 'on';
        if nargin<5
            FaceColor = ones(1,3);
        end
    end
end
[row,column] = size(X);
if row==1
    X = X';
end
[row,column] = size(Y);
if row==1
    Y = Y';
end
[row,column] = size(height);
Nh = length( height );
N  = Nh +1;
N_sides  = length(X)/N;
x0 = Oxyz(1);
y0 = Oxyz(2);
z0 = Oxyz(3);
X  = X +x0;
Y  = Y +y0;
Z  = zeros( size(X) );
Z(1:N_sides) = z0*ones(N_sides,1);
for i=2:N
    ixm = i*N_sides;
    Z(ixm-N_sides+1:ixm) =  (z0+sum(height(1:i-
1)))*ones(N_sides,1);
    F(ixm-N_sides-N_sides+1:ixm-N_sides,:) = [ (1:N_sides-1)',
(2:N_sides)', (N_sides+2:N_sides+N_sides)',
(N_sides+1:N_sides+N_sides-1)';...
        N_sides, 1, N_sides+1, N_sides+N_sides] +(i-2)*N_sides;
end
patch_obj = patch('Faces',F,'Vertices',[X,Y,Z],...

'FaceColor',FaceColor,'EdgeColor',EdgeColor,'Visible',visible );
```

# APPENDIX B

## LabVIEW NI Vision Program for Target Object Colour Detection and interface with Arduino microcontroller Board


LabVIEW_Vision_IEEE_Con.vi Front Panel
File Edit View Project Operate Tools Window Help
18pt Application Font
Search
Color Sensitivity
Low
Session In
cam1
VISA resource
Stop
End-Effector Position
Picking Position
Place Position "A"
Place Position "B"
Place Position "C"
Color Spectrum
Blue
Yellow
Red
Blue Color Waveform Chart
Yellow Color Waveform Chart 2
Red Color Waveform Chart 3
Plot 0
Color Spectrum
Time


LabVIEW_Vision_IEEE_Con.vi Block Diagram
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
Color Spectrum
Image
Blue Color Waveform Chart
Yellow Color Waveform Chart 2
Red Color Waveform Chart 3
untitled
Session In
Blue
Yellow
Red
Color Sensitivity
VISA resource
Picking Position
Place Position "A"
Place Position "B"
Place Position "C"
Stop 2
Input

# APPENDIX C
## Arduino IDE C++ Program for Robotic Kinematic Feedback Signal Detection

```
//Parallel Robot Rotation Detection

#define outputA1 2
#define outputA2 3

#define outputB1 5
#define outputB2 6

#define outputC1 8
#define outputC2 9

int counter1 = 0;
int counter2 = 0;
int counter3 = 0;

int aState1;
int aLastState1;
int aState2;
int aLastState2;
int aState3;
int aLastState3;

void setup() {
pinMode (outputA1,INPUT);
pinMode (outputA2,INPUT);
pinMode (outputB1,INPUT);
pinMode (outputB2,INPUT);
pinMode (outputC1,INPUT);
pinMode (outputC2,INPUT);

Serial.begin (9600);
Serial.print (counter1*9);
Serial.print ("\t");
Serial.print (counter2*9);
Serial.print ("\t");
Serial.println (counter3*9);

aLastState1 = digitalRead(outputA1);
aLastState2 = digitalRead(outputB1);
aLastState3 = digitalRead(outputC1);
}
```

```
void loop() {
aState1 = digitalRead(outputA1);
aState2 = digitalRead(outputB1);
aState3 = digitalRead(outputC1);

if (aState1 != aLastState1){
if (digitalRead(outputA2)!= aState1)
{
counter1 --;
}else{
counter1 ++;
}
Serial.print (counter1*9);
Serial.print ("\t");
Serial.print (counter2*9);
Serial.print ("\t");
Serial.println (counter3*9);
}
aLastState1 =aState1;

if (aState2 != aLastState2){
if (digitalRead(outputB2)!= aState2)
{
counter2 --;
}else{
counter2 ++;
}
Serial.print (counter1*9);
Serial.print ("\t");
Serial.print (counter2*9);
Serial.print ("\t");
Serial.println (counter3*9);
}
aLastState2 =aState2;

if (aState3 != aLastState3){
if (digitalRead(outputC2)!= aState3)
{
counter3 --;
}else{
counter3 ++;
}
Serial.print (counter1*9);
```

```
Serial.print ("\t");
Serial.print (counter2*9);
Serial.print ("\t");
Serial.println (counter3*9);
}
aLastState3 =aState3;
}
```

## Biography

| | |
|---|---|
| **Name - Surname** | Surin Subson |
| **Date of Birth** | December 31, 1976 |
| **Address** | 263/160, Tambon Lumpakkud, Amphoe Tunyaburi, Pathumthani, 12110, |
| **Education** | Bachelor of Business Administration (Management), (2005-2008) |
| **Experiences Work** | Project Manager<br>Maxens Co.,Ltd. (2010-2022). |
| **Telephone Number** | +668-1558-5941 |
| **Email Address** | surin_s@mail.rmutt.ac.th |

Surin Subson, Dechrit Maneetham, Myo Min Aung: Kinematics Simulation and Experiment for Optimum Design of a New Prototype Parallel Robot. International Journal of Engineering Trends and Technology, Volume 70 Issue 10, 350-362, October 2022 https://doi.org/10.14445/22315381/IJETT-V70I10P234

Surin Subson, Dechrit Maneetham, Myo Min Aung: Real-time Vision image processing based on LabVIEW and Microcontroller controlled Parallel Robot, 2022 IEEE 8th Information Technology International Seminar (ITIS) https://ieeexplore.ieee.org/document/10009950