

**CABLE-DRIVEN PARALLEL ROBOT
FOR SMART FARMING**

PHIROM SASOMSAP A/L EHDI

UNIVERSITY SAINS MALAYSIA

2021

CABLE-DRIVEN PARALLEL ROBOT FOR SMART FARMING

by

PHIROM SASOMSAP A/L EHDI

**Thesis submitted in partial fulfilment of requirements
for the degree of
Bachelor of Mechatronic Engineering**

July 2021

ACKNOWLEDGEMENT

A student can only be successfully navigated with strong support. I would like to sincerely thank all the people who have helped me throughout this process and made my time at the University of Science Malaysia memorable.

I would like to say thank you and sincere gratitude to my supervisor, Dr. Abdul Sattar Bin Din for giving me the opportunity to do research and providing invaluable guidance throughout this research. His motivation and discussion for this project have deeply inspired me. He has guided me and give the idea to carry out the research and to present the research works as very clear. I also would like to show appreciation to him for his empathy, friendship and good sense of humor.

After that, I would like to say thank to my examiner, Assoc. Prof. Ir. Dr. Rosmiwati Mohd Mokhtar, for giving the information about the final year project particularly the guidance for preparation. In addition, I also would like to express my deep gratitude for her for giving the motivation about the final report improvement during pre-viva session. Other than that, special appreciation to my family for giving encouragement, enthusiasm and invaluable assistance to me. I am very grateful to my family who are willing to spend their precious time in participating for my data collection. Without all of these, I might not be able to finish this project properly.

Finally, I would like to give special thanks to my friends and everyone past and present at the Mechatronic Lab for making the place that I worked in a genuinely fun and productive workspace. I appreciate everyone who have supported me to complete the research work directly or indirectly

TABLE OF CONTENTS

ACKNOWLEDGEMENT	ii
TABLE OF CONTENTS.....	iii
LIST OF TABLES.....	v
LIST OF FIGURES	vi
LIST OF ABBREVIATIONS.....	vii
MATHEMATICAL NOTATIONS	viii
ABSTRAK.....	xi
ABSTRACT.....	x
CHAPTER 1 : INTRODUCTION.....	1
1.1 Research Background	1
1.2 Problem Statement	2
1.3 Objectives.....	2
1.4 Scope of Project	2
1.5 Thesis Outline.....	2
CHAPTER 2 : LITERATURE REVIEW	4
2.1 Introduction	4
2.2 Related Works.....	5
2.3 Theoretical Analysis of CDPR.....	7
2.3.1 Forward Kinematics	7
2.3.2 Inverse Kinematics.....	8
2.4 Derivation.....	8
2.5 Advantages and Drawbacks of CDPR	10
2.6 Summary	11
CHAPTER 3 : METHODOLOGY	12
3.1 Introduction	12
3.2 Hardware and Components	13
3.3 Software and Application.....	15
3.3.1 Arduino IDE.....	15

3.3.2 Fritzing	15
3.3.3 AutoCAD	16
3.3.4 Little Stars	17
3.4 Block Diagram and Flowchart	18
3.5 Schematic Diagram	18
3.6 Conceptual Design of CDPR	20
3.7 System Dimensional Optimization of CDPR.....	21
3.8 Motion Test	22
3.9 Summary	22
CHAPTER 4 RESULTS AND DISCUSSION	23
4.1 Introduction	23
4.2 Overall Design.....	23
4.2.1 Mechanical Structure.....	24
4.2.2 End-Effector.....	24
4.2.3 Center of Mass Balancing.....	25
4.2.4 Water Tube Position	25
4.2.5 Circuit Connection	26
4.3 Motion Generation	27
4.3.1 Motion Speed.....	28
4.3.2 Association Between the Motion of End-Effector and Cables Length.....	29
4.4 Remote Monitoring System	30
4.5 Summary	31
CHAPTER 5 : CONCLUSION	32
5.1 Conclusion.....	32
5.2 Future Development.....	33
REFERENCES	34
APPENDIX A: CODING	36

LIST OF TABLES

Table 2.1: Advantages and Drawbacks of CDPRs	10
Table 3.1 List of Electronic Components of CDPR	13
Table 3.2 List of Mechanical Components of CDPR	14
Table 3.3 Pin Connection.....	19
Table 3.4 Full-Scale Parameters of CDPR	21

LIST OF FIGURES

Figure 2.1 4 CDPM Phenotypic.....	4
Figure 2.2 Different CDPR Classification.....	5
Figure 2.3 The Dimensional of CDPR	6
Figure 3.1 A4988 Stepper Driver Installation.....	15
Figure 3.2 The single connection of stepper motor	16
Figure 3.3 CAD Modeling Stepper Motor	16
Figure 3.4 Little Stars IP Camera Application	17
Figure 3.5 Little Stars Application Setup	17
Figure 3.6 Block diagram for an operation of CDPR.....	18
Figure 3.7 Flowchart diagram for an operation of CDPR	19
Figure 3.8 Schematic Diagram of CDPR	20
Figure 3.9 CAD Modeling Structure of CDPR: (a) 3-Dimensional View (b) Front View	21
Figure 3.10 The Modeling of CDPR: (c) Top View (d) Bottom View.....	21
Figure 3.11 The Dimensions of CDPR	22
Figure 4.1 Overall Design Of CDPR For Smart Farming.....	24
Figure 4.2 Mechanical Structure of CDPR.....	25
Figure 4.3 The Design Of End-Effector Of CDPR	25
Figure 4.4 Center of Mass Balancing of the End-Effector.....	26
Figure 4.5 The Position of Water Tube for Irrigation Process	26
Figure 4.6 Circuit Connection	26
Figure 4.7 The Direction of the End-Effector to Target Location	27
Figure 4.8 Curvilinear Translational motion	27
Figure 4.9 Rotation about a fixed axis	28
Figure 4.10 The Movement of the End-Effector.....	29
Figure 4.11 Graph of Cables Length (cm) Versus Time (s)	30
Figure 4.12 The Remote Monitoring System	30

LIST OF ABBREVIATIONS

CDPR	Cable Driven Parallel Robot
DoF	Degree of Freedom
SCDR	Single-link Cable Driven Parallel Robot
MCDR	Multi-link Cable Driven Parallel Robot
EoM	Equation of Motion
CoM	Centre of Mass
IK	Inverse Kinematics
FK	Forward Kinematics
ID	Inverse Dynamics
CRM	Cable Routing Matrix
SEC	Static Equilibrium Condition
SW	Static Workspace
WCC	Wrench Closure Condition
WCW	Wrench Closure Workspace
WFC	Wrench Feasibility Condition
WFW	Wrench Feasible Workspace
DFC	Dynamic Feasibility Condition
DW	Dynamic Workspace
CASPR	Cable-robot Analysis and Simulation Platform for Research
ROS	Robotics Operating System

MATHEMATICAL NOTATIONS

n	speed of the actuators
μ	number of cables
η	number of operational space DoFs
r	number of rigid links
ρ	number of end effectors
w	joint space wrench vector
x	body space pose vector
F	body space force vector
l	cable lengths
f	vector of cable forces
r	linear distance of end-effector
y	operational space pose vector
τ	operational space force vector
J	joint-to-operational space Jacobian matrix
L	the length of end-effector from a fixed point
sps	steps per second
spv	steps per revolution
G	gravitational force vector

CABLE-DRIVEN PARALLEL ROBOT FOR SMART FARMING

ABSTRAK

Robot selari yang didorong oleh kabel (CDPR) membentuk kelas manipulator robot di mana penggerak dihantar melalui kabel. Oleh kerana ciri-ciri dan kelebihan transmisi kabel yang unik, CDPR menjadi semakin popular pada masa kini. Tujuan projek ini adalah untuk membina CDPR yang dapat dikendalikan dari segi pemantauan jarak jauh dan proses pengairan dan untuk menyiasat pergerakan robot yang bergerak semasa proses penghantaran. Skala kecil CDPR telah dibina yang terdiri dari struktur mekanik, mikrokontroler, pemacu, komputer untuk pengaturcaraan, dan efektor yang dipasang dengan muncung air dan kamera untuk pengairan dan pemantauan jarak jauh. Dalam reka bentuk ini, pose efektor tidak dapat dikawal sepenuhnya oleh panjang kabel dan akan bergantung kepada kehadiran graviti (jenis yang tidak terkawal). Prestasi CDPR dari segi kedudukan ketepatan efektor akhir mengikut urutan pergerakan menegak, mendatar dan melengkung telah diperhatikan. Walau bagaimanapun, terdapat tiga faktor yang boleh dipengaruhi pada ketidakseimbangan efektor seperti kelajuan penggerak, jisim efektor dan beban luaran. Oleh itu, kaedah untuk mengatasi masalah telah dikembangkan. Kelajuan penggerak dan jisim efektor akhir telah dijatuhkan dengan alasan untuk mengurangkan inersia dan untuk mengelakkan kendur kabel semasa sistem penghantaran. Efektor menjadi tidak stabil disebabkan oleh adanya tiub air (beban luaran). Oleh itu, pusat pengimbangan jisim telah dirancang dalam robot.

CABLE-DRIVEN PARALLEL ROBOT FOR SMART FARMING

ABSTRACT

Cable-driven parallel robot or CDPR is the robot that the actuation is transmitted through cables to the end-effector. Based on the specific qualification and the utility of cable transmission, the CDPR is become popular in this modern day. The purposes of this project are to construct the CDPR which can be operated in term of remote monitoring and irrigation process and to investigate the motion generation of robot during the transmission process. The small scale of CDPR has been constructed which consists of a mechanical structure, microcontroller, driver, computer for programming, and the end-effector that was attached with water nozzle and camera for irrigation and remote monitoring respectively. In this design, the pose of the end-effector is not completely controlled by the cables length and it will depend on the presence of gravity (under-constrained type). The performance of CDPR in term of the accuracy position of the end-effector in order of vertical, horizontal and curvilinear movement has been observed. However, there three factors that can be affected on the unbalancing of the end-effector such as actuators speed, mass of the end-effector and external load. Therefore, the ways to overcome the problems were developed. The speed of the actuators and the mass of the end-effector have been dropped by the reason to reduce the inertia and to avoid the sagging of cable during the transmission system. The unstable of the end-effector is caused by the presence of water tube (external load). Therefore, the center of mass balancing has been designed in the robot.

CHAPTER 1 : INTRODUCTION

1.1 Research Background

The research presented in this report was about the design of a cable-driven parallel robot for smart farming in agriculture field. The research emphasizes on the modeling, design and optimization of the cable system along with the development of a stabilization system for suspended payloads. Cable-driven parallel robot (CDPR) is a type of parallel manipulators that is the end-effector is supported in parallel by multiple cables that are controlled by multiple tensioning actuators. The CDPR is a robot whose end-effector pose is controlled by winding and unwinding independent cables connecting the end effector to the fixed base [1].

The CDPRs have a large potential in various applications, such as agriculture irrigation, remote monitoring, clean-up of the dirty area, manipulation of heavy payloads, and rescue systems. There are consists of the end-effector which connected by multiple cables. Besides, the end-effector is operated by actuators which can extend or retract cables. Cable-driven manipulators are structurally that like the parallel ones and possess desirable characteristics, such as a large workspace, when compared to the workspace of the classical parallel manipulators.

Furthermore, CDPR is the robot that structure is light. This will give a good dynamic property, can hold the high load, can be used as a transport and etc. However, the robot has the limitation due to the main characteristic of the cables. The cables can only pull the end-effector but cannot push it. Therefore, the analysis and design of workspace are different from its which can be referred to parallel manipulators. The workspace is restricted by the requirement that having the positive tension in all cables. For example, Pinto et al. [2] proposed SPIDERobot, a 4 degree of freedom (DOF) robot, for architectures projects. However, the CDPR need for efficient tension distribution algorithms to improve the positioning accuracy. Such algorithms have been tested in simulations but remain to be used for the real-time control of a prototype [3].

1.2 Problem Statement

The problem statement of this project was based on the agriculture sector. Many of agriculture industries incur large costs to pay more salaries to the workers due to inefficiency of current technology. The workers were struggling hard to manage works in agriculture sector by the reason of the large size of workspace and insufficient of workers. Therefore, the cable driven parallel robot (CDPR) for smart farming was introduced in this project. The purposes of this robot are to reduce the costs of an agriculture industries and reduce the energy of workers. The robot will be operated as remote monitoring and perform the irrigation process. The system of robot can be controlled by either one or more person.

1.3 Objectives

The main objectives of this project are listed as below:

1. To develop the cable driven parallel robot which capable in remote monitoring and irrigation system.
2. To investigate the motion generation of robot during the transmission process.

1.4 Scope of Project

This project proposed the cable-driven parallel robot for smart farming. The consideration of robot mechanism based on the hardware and components should be thrifed. Therefore, this project is constructed which making scale down version in term of prototype (small size) to fit within the budget. There are many tasks in farming that can benefit from this type of robot. In this project, the remote monitoring and irrigation system will be developed on the CDPR. The scope of research will be based on the working principle, control system, and limitations of CDPR. The analysis of project will be focused on the performance of motion generation of robot.

1.5 Thesis Outline

This section gives an overview of the thesis structure and a brief introduction to each chapter. In short, this thesis consists of five main chapters:

- **Chapter 1: Introduction**

Chapter one focused on the introduction of the project. It was more focused on the purposes of this project. This chapter consists of research background, problem statement, objectives and scope of project.

- **Chapter 2: Literature Review**

Chapter two presents the literature review of completed previous work that related to this project. The section in this chapter is about the theoretical analysis of cable-driven parallel robot including an inverse and forward kinematics system. This chapter also discuss the advantages and drawbacks, dimension layout and the derivation of CDPR.

- **Chapter 3: Research Methodology**

Chapter three describes the methodology of robot construction. It includes the hardware/components and software used in this project. Furthermore, this chapter is indicated the system layout of robot such as block diagram, schematic diagram, CAD modeling and system dimensional of robot. The motion test of the robot was described on this chapter.

- **Chapter 4: Result and Discussion**

Chapter four presents the results and discussion of the project. It indicates the overall design of robot. The performance of CDPR in term of the accuracy position of the end-effector in order of vertical and horizontal movement were analyzed. Besides, the factors that can be affected on the unbalancing of the end-effector such as external load and actuator speed which having the correlation with the torque and inertia were observed in this chapter.

- **Chapter 5: Conclusion and Future Development**

Chapter five presents the summary from overall of this project. The future development of robot was suggested on this chapter.

CHAPTER 2 : LITERATURE REVIEW

2.1 Introduction

Cable-driven parallel robot represents a term of mechanical system capable of independently manipulating the surrounding of an environment by using the end-effector which is actuated through kinematics chain of the typical rigid link. By the reason of the advantages of robot, applications of CDPR can be found in many fields such as warehousing in logistics [4], contour crafting system in the construction industry, camera system in the stadium, etc. Thus, lots of them have redundant actuators and this kind of structure produces an over-constrained system [5]. These robots have been commonly classified in term of their kinematics structure which having two major class: serial manipulators and parallel manipulators. The Figure 2.1 shown the four-cable driven phenotyping parallel manipulator (4CDPM). The purpose of this system is to scan a one-acre maize field. The field contains multiple plots arranged in a grid, and the 4CDPM will be programmed to visit the plots to acquire phenotypic and environmental data based on a predetermined schedule [6].

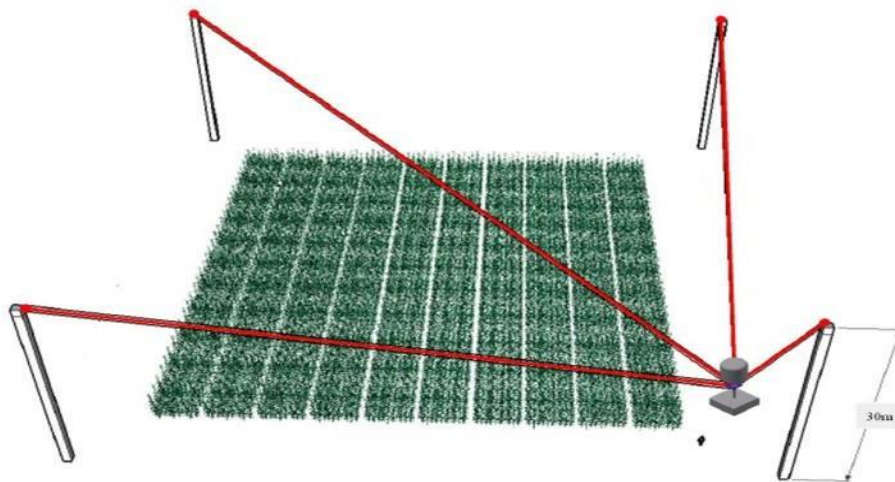


Figure 2.1: 4 CDPM Phenotypic [6]

The links of the parallel manipulator form a closed kinematic chain [7]. This means that parallel manipulators may distribute the actuation of a single DoF across multiple actuators/links, resulting in a typically higher payload to weight ratio when compared to serial manipulators. The closed kinematic chain also increases the rigidity of the mechanism by providing additional constraints. The outcome of class of the robots was in term of cable-driven parallel robots (CDPRs). The use of cables in place of rigid links results in several

unique benefits for CDPRs. First, the actuating cables possess a much lower inertia. This results in lower overall system inertia, thereby further increasing the payload to weight ratio and allowing for higher speeds of operation.

The important factors to achieve a suitable performance in controlling the robot is the accuracy of the kinematic parameters. This will be dealt with the two main approaches. The first category tries to cope with it through applying robust controller schemes [8]. Even though this method requires no meticulously accurate installation procedure, their performance and accuracy are bounded and the impact of uncertainty cannot be completely eliminated. In addition, the second category has been tried to eliminate the sources of an uncertainty all at once via performing calibration routines on the robot [9]

2.2 Related Works

CDPRs are typically further classified based upon the number of rigid links that they possess. Single-link CDPRs (SCDRs) represent a single link that is actuated by a parallel configuration of cables [10]. These manipulators can be seen as parallel manipulators in which all rigid links with the exception of the end effector have been replaced by cables. As a result, SCDRs are typically described in terms of the number of cables and the joint types. Cable-driven robot also can be classified as under-constrained and fully-constrained and under-constrained, as shown in Figure 2.2 (a) and (b) [11]. In the fully constrained class, the pose of the end-effector was completely determined as the function of the length of cable. In the under-constrained type, the pose of the end-effector was not completely determined by the lengths of the cable. The under-constrained type was also depended on the gravity.

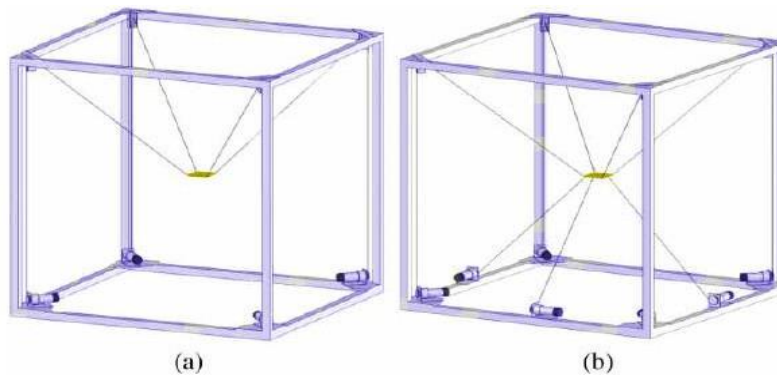


Figure 2.2 Different CDPR Classification

For the basic of the CDPR control system, the microcontroller receives an input from the PC. It receives a destination or position in the workspace which can move the end-effector by writing the program through a PC connected to the USB port of the Arduino, or it will receive a preference velocity vector on the end-effector. The controller processes with the data and determines the speed of each winch that required to move and guide the end-effector along the target position. The speed was then transmitted to the winches or pulleys that the return messages contained of the length of the cables which can be allowed the controller to estimate the current position of the end-effector.

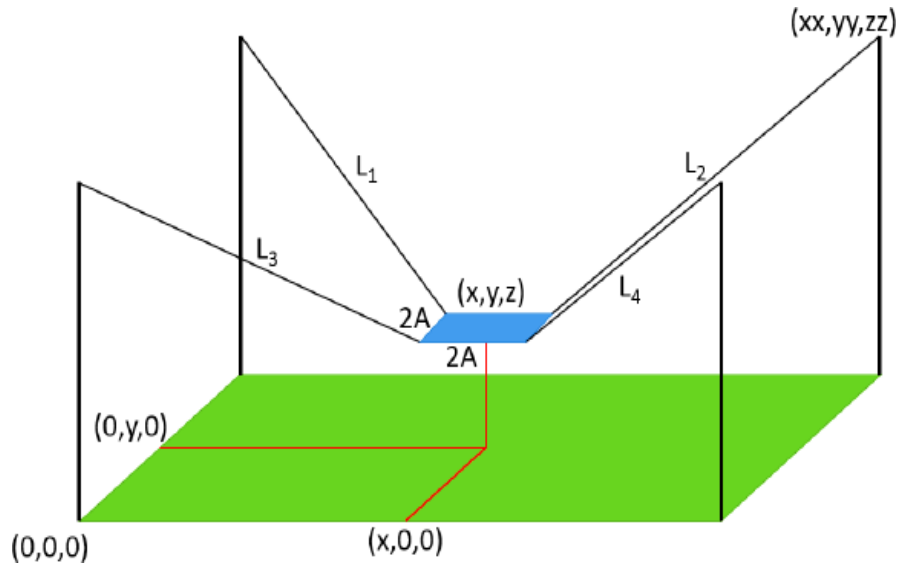


Figure 2.3: The Dimensional of CDPR [12]

The x and y-axis (or horizontal and vertical) positions from the left correlate to the motion speed of cables one and two, respectively whereas the x and y-axis signals from the right correlate to the motion speed of cables three and four. In this mode, the controller simply reads the four analog inputs, maps them to desired speeds for the winches, and transmits the speed to the appropriate winch. This mode is normally used for the tuning of cable lengths during the initial setup or in the situation when one cable becomes sagging [12].

2.3 Theoretical Analysis of CDPR

Kinematic analysis can be performed by considering the movement of the end-effector in term of planar and spatial or the vertical and horizontal movement. At normally, the controllable workspace or the predict force feasible can be calculated by using the Jacobian matrix. This case will presence the mapping of kinetostatic between end-effector and the fixed point [13]. The formulation will be determined by considering of the geometry of the robot. Since, the planar motion will be located as a part of possible configurations based on the spatial motion. The Analysis on the fixed point will be observed to construct the constraints equations. The replenishment of the workspace is a part of the configurations kinematic. There were violated the positive cables' tension and discharged.

2.3.1 Forward Kinematics

For the forward kinematics, there are contain of the pose of the end-effector for a given set of cable lengths [14]. However, when the CDPR has performed in the gravitational field, the situations that can be occurred are:

- 1) the given set of cable lengths is not compatible with the geometry of the manipulator and the forward kinematic problem has no feasible solutions.
- 2) there is a feasible solution, but it may not be in the plane of motion.
- 3) there was a feasible solution and it is on the plane of the end-effector's motion.

The situation (2) refers to the operation of the manipulator as an under-constrained cable-driven parallel manipulator. The situation (3) refers to the operation of the manipulator as a fully-constrained cable-driven manipulator. It is worthy to note that the situation (3) represents a subset of the feasible solutions of situation (2). Since, to solve the FKP of the four-cable-driven parallel manipulator, the cable mass is negligible with respect to the manipulator's end-effector and the cable lengths should be compatible with the geometry of the manipulator's structure.

2.3.2 Inverse Kinematics

Based on the Figure 2.3, the cables are connected to the end-effector through two attachment points A and B, whose coordinates with respect to the fixed frame are given by (x_A, y_A, z_A) and (x_B, y_B, z_B) , respectively. The end-effector length is determined by $2h$ and its center of mass G is located at point O'. The fixed frame at the beginning point, O can be chosen as concurrent with the midpoint of the end-effector and the beginning point of the can be chosen as concurrent with the midpoint of the end-effector, whereas the position are (x_G, y_G, z_G) . The load can be suspended on the center of mass of the end-effector. The end-effector pose of the four-cable-driven parallel manipulator is given by the position of point G and θ angle, which is the angle between x and x' axes. The inverse kinematics problem (IKP) of the four cable-driven manipulator can be formulated as finding the cable lengths l_i as a function of the end-effector pose [14].

2.4 Derivation

The CDPR system can be tracked and navigated by knowing only the cable lengths, the cable lengths and the Cartesian coordinate system should be able to convert. By assuming of the end-effector was always level, the length of each cable will be calculated in terms of the end-effector position (x, y, z) by using the Pythagorean theorem [15] based on Figure 2.3.

$$L_1 = \sqrt{(x - A)^2 + (yy - y - A)^2 + (zz - z)^2} \quad (1)$$

$$L_2 = \sqrt{(xx - x - A)^2 + (yy - y - A)^2 + (zz - z)^2} \quad (2)$$

$$L_3 = \sqrt{(x - A)^2 + (yy - A)^2 + (zz - z)^2} \quad (3)$$

$$L_4 = \sqrt{(xx - x - A)^2 + (yy - A)^2 + (zz - z)^2} \quad (4)$$

Solves the three equations for x , y and z by using given formula:

$$x = \frac{-L_1^2 + L_2^2 - xx^2 + 2 * A * xx}{4 * A - 2 * xx} \quad (5)$$

$$y = \frac{L_1^2 - L_3^2 - yy^2 + 2* A * yy}{4*A - 2*yy} \quad (6)$$

$$z = zz - \sqrt{L_3^2 - (x - A)^2 - (y - A)^2} \quad (7)$$

With this, the administrators or users are able to track the position of a suspended payload by tracking cable feed with actuators. In the previous sections it was stated that the cables are actuated based on a preset speed of the end effector. By taking the derivatives of (5) - (7) with respect to time, the equations of cable velocities will be created based on the velocities of the end-effector:

$$\frac{dL_n}{dt} = \frac{\partial L_n}{\partial x} \frac{dx}{dt} + \frac{\partial L_n}{\partial y} \frac{dy}{dt} + \frac{\partial L_n}{\partial z} \frac{dz}{dt} \quad (8)$$

$$\frac{dL_1}{dt} = \frac{(x-A)\dot{x} + (y-yy+A)\dot{y} + (z-zz)\dot{z}}{\sqrt{(x-A)^2 + (y-yy+A)^2 + (z-zz)^2}} \quad (9)$$

$$\frac{dL_2}{dt} = \frac{(x-xx+A)\dot{x} + (y-yy+A)\dot{y} + (z-zz)\dot{z}}{\sqrt{(x-xx+A)^2 + (y-yy+A)^2 + (z-zz)^2}} \quad (10)$$

$$\frac{dL_3}{dt} = \frac{(x-A)\dot{x} + (y-A)\dot{y} + (z-zz)\dot{z}}{\sqrt{(x-A)^2 + (y-A)^2 + (z-zz)^2}} \quad (11)$$

$$\frac{dL_4}{dt} = \frac{(x-xx+A)\dot{x} + (y-A)\dot{y} + (z-zz)\dot{z}}{\sqrt{(x-xx+A)^2 + (y-A)^2 + (z-zz)^2}} \quad (12)$$

The speed of the actuators can be calculated as a unit of revolution per minute (RPM). The formula to calculate the speed of actuators are stated as below [16]:

$$\text{Speed of actuator, } n = \frac{\text{Steps per second}}{\text{Steps per revolution}} \quad (13)$$

The velocity in a unit of meter per second (m/s) and angular velocity in radian per second (rad/s) can be calculated by using the formula:

$$\text{The velocity of end-effector, } v = r \times n \times \frac{2\pi}{60} \quad (14)$$

$$\text{Angular velocity of the end-effector, } \omega = \frac{v}{L} \quad (15)$$

where,

A = Center origin of moving platform

L = Cable length

x, y and z = Axis value

n = Speed of actuator

v = Linear velocity

ω = Angular velocity

r = Linear distance of the end-effector

2.5 Advantages and Drawbacks of CDPR

The CDPR have several advantages and disadvantages based on their specific characteristics. In normally, the CDPR has been used the large workspace. The table below shown the advantages and the drawbacks of the system operation of CDPR that has been used in real life.

Table 2.1: Advantages and Drawbacks of CDPRs [17]

Advantages	Drawbacks
<ul style="list-style-type: none">• Large workspace• High payload-to-weight ratio• Low inertia of the mobile parts• Energy efficiency• Low cost• Low maintenance effort• Deplorability and transportability• Possible reconfigurability	<ul style="list-style-type: none">• Mono-directional cable transmission of forces to the platform• Cable interferences• Collisions between the cables and the environment• Cable deformations

2.6 Summary

In this literature review, the current status of CDPR research has been presented. Using cable-based transmission, CDPRs have been shown to possess an increased payload to weight ratio, reduced mechanism inertia, potentially larger operating workspace, increased reconfigurability and higher transportability when compared to traditional serial or parallel manipulators. The CDPRs can be defined in many tasks such as a desired set of mechanism kinematics. Compared to traditional manipulators, the study of CDPR motion generation is complicated by the constraints of operation. Kinematic characteristics such as workspace must be analyzed in a different manner with respect to classic parallel manipulators since the intrinsic characteristics of cables is that it must be in tension but with limited forces' values. A formulation is presented as based on the geometry of the manipulator. The length of each cable can be calculated in terms of the end-effector position by using the Pythagorean Theorem. From the overall, this chapter was more focus on the mechanism of robot, working principle, derivation, advantages and the limitation of robot.

CHAPTER 3 : METHODOLOGY

3.1 Introduction

The performance of CDPR can be affected by their construction or model. The option of a relevant model such as full-constrained and under-constrained can predict the feature of a robot in term of moving platform positioning accuracy. The under-constrained cable robot has been chosen in this project. The pose of the end-effector of this robot is not completely determined by the system transmission from cable and they rely on the presence of gravity. CDPR for smart farming is composed of the major parts such as rigid frame, four pulleys lifting and actuators, cables, and the end-effector that attached with camera and water spray nozzle. The system of the robot will be controlled by the PC.

The CDPR for smart farming is suitable to operate in the large workspace. However, the construction robot in this project has been scaled down in terms of prototype and others relevant hardware and components. Furthermore, this robot was needed to attach the external load or water tube as an irrigation system that can lead to the unbalancing of the end-effector. Therefore, the design of the end-effector has the significant role to make the end-effector balancing during the irrigation process. However, the speed and mass of the end-effector also can affect the stability of the effector. Therefore, all these factors will be considered in this project. Furthermore, all hardware and components to construct the robot have been listed and their functional was explained. The block diagram, schematic diagram and 3D modeling have been designed to virtualize the mechanism of the robot.

3.2 Hardware and Components

There are several hardware and components used to construct the CDPR for smart framing. The Table 3.1 and 3.2 shown the details of hardware and components and their function that have been used in the project.

Table 3.1 List of Electronic Components of CDPR


















Electronic Components	Name & Function
	Arduino Mega – Used to read the input from stepper driver and turn into the output
	Bipolar Stepper Motor (NEMA-17HS4401) – Used as an actuator to retract and extend the cable.
	Stepper Motor Driver (A4988) – Microstepping driver for controlling bipolar stepper motors which has built-in translator for easy operation.
	Capacitor 16V 100uF – Used for protecting the driver board from voltage spikes.
	Resistor 100kΩ – Used to avoid the floating input of stepper driver
	A9 Mini Wi-fi Camera – Attached to the end-effector for monitoring system.
	Power Source – Used to supply the voltage (12v) to the actuators.

Table 3.2 List of Mechanical Components of CDPR

Mechanical Components	Name & Function
	Water Spray Nozzle (Knapsack Sprayer) – Attached to the end-effector for irrigation and pest control.
	Wood Planks – Used as a frame of CDPR.
	Pulley Lifting – Used for transmission system from stepper motor to the end-effector.
	Waxed Cotton Cable – Attached with actuator to drive the end-effector.
	PVC Fitting Connector – Used as the end-effector of CDPR.
	3D Printed Plate – Acts as a pulley, attached to the shaft of stepper motor.
	Pipe PVC – Used for balancing of the end-effector during irrigation process or pest control.
	Water tube – The water tube was connected to the end-effector for irrigation process.
	Catapult Rubber – Used for balancing of the end-effector during irrigation process or pest control.
	Cardboard – A part of the end-effector

3.3 Software and Application

There are several software and application that have been used in this project which consist of Arduino IDE, Fritzing, AutoCAD and Little Stars.

3.3.1 Arduino IDE

The Arduino IDE is the software that has been used to write the code as a purpose to control the speed and direction of an actuators. The program code subsequently uploaded to the microcontroller. However, the A4988 driver should be installed before start writing the code. The Figure 3.1 shown the installation of stepper driver from the library of Arduino IDE.

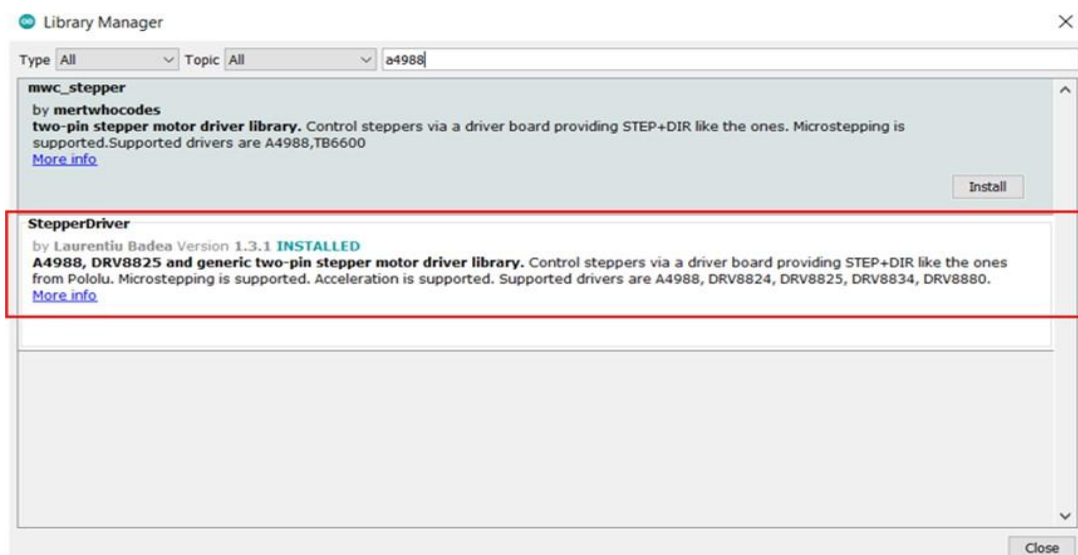


Figure 3.1 A4988 Stepper Driver Installation

3.3.2 Fritzing

The schematic diagram or circuit connection has been drawn by using Fritzing software. This software can design an electronic circuit by using standard schematic symbols from the schematic interface. This software has a wide library hardware or components such as Arduino board, stepper motor, resistor, capacitor and etc. The Figure 3.2 shown the schematic of single connection of stepper motor.

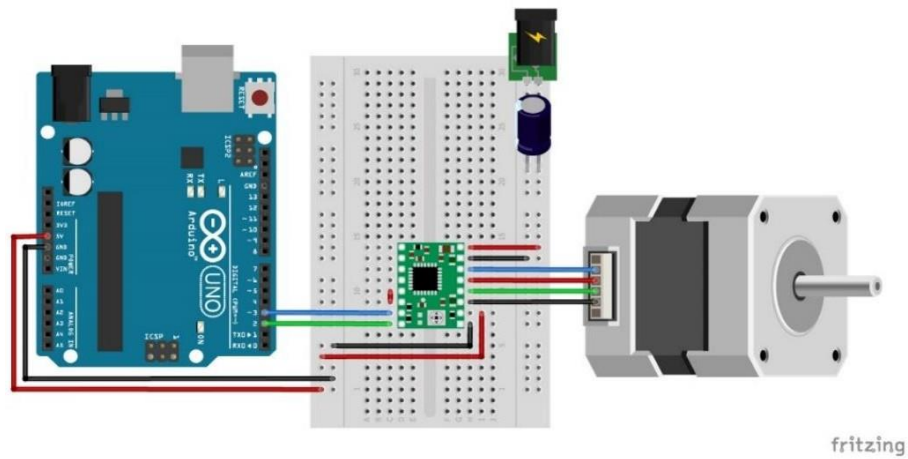


Figure 3.2 The single connection of stepper motor

3.3.3 AutoCAD

The AutoCAD is the software which has numerous capabilities that can be applied to the array of projects in various fields. The AutoCAD was used in this project to draw the modeling of CDPR. The Figure 3.3 below shown a part of drawing from AutoCAD.

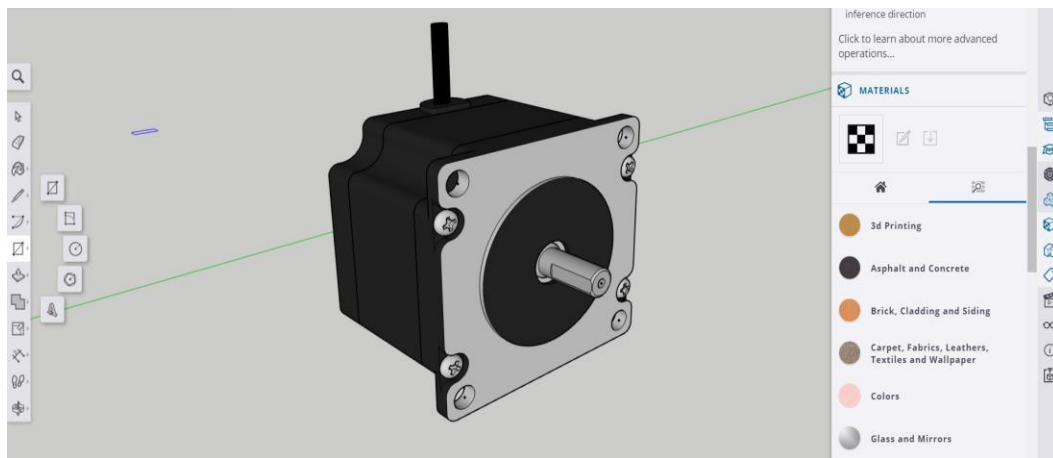


Figure 3.3 CAD Modeling Stepper Motor

3.3.4 Little Stars

Little Star is the application that use to display the live image or video through the PC. This application was used as remote monitoring by connecting with the internet protocol (IP) Camera. The Figure 3.4 shown the application of IP Camera.

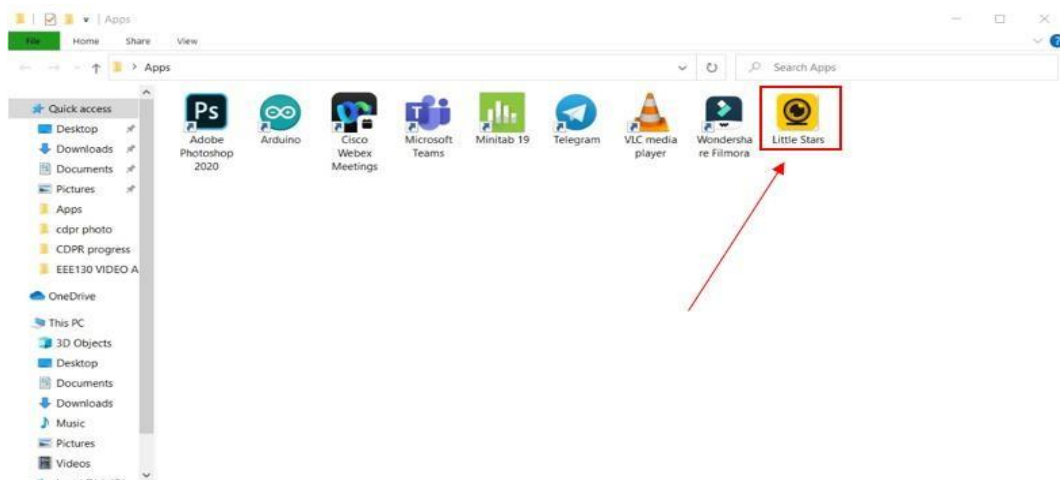


Figure 3.4 Little Stars IP Camera Application

After the installation of the application is finished, the IP camera should be turned on and the wi-fi of remote camera will be displayed on the PC. However, the ID and password should be changed to another one by the reason to avoid the other user can connect through this wi-fi. The Figure 3.5 shown image displayed through the application.

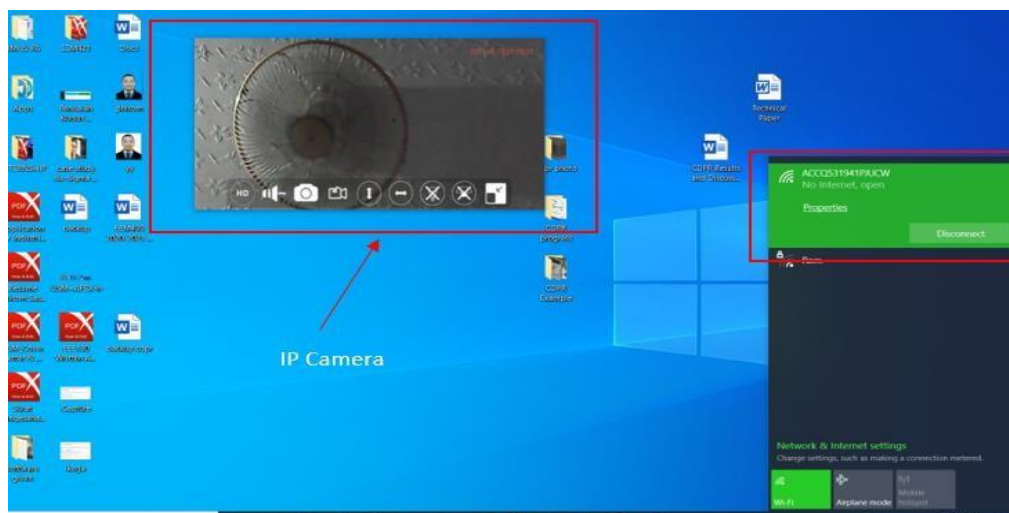


Figure 3.5 Little Stars Application Setup

3.4 Block Diagram and Flowchart

The block diagram of cable driven parallel robot for smart farming has been developed. As a control system, the program code has been created through the PC and uploaded to the microcontroller which acts as a brain of CDPR. At the same time, the data from microcontroller will be transferred to stepper driver to initialize the actuators. The transmission system has been designed as composed of lifting pulleys and cables to drive the end-effector from one position to another position. The Figure 3.6 shown the block diagram of the CDPR.

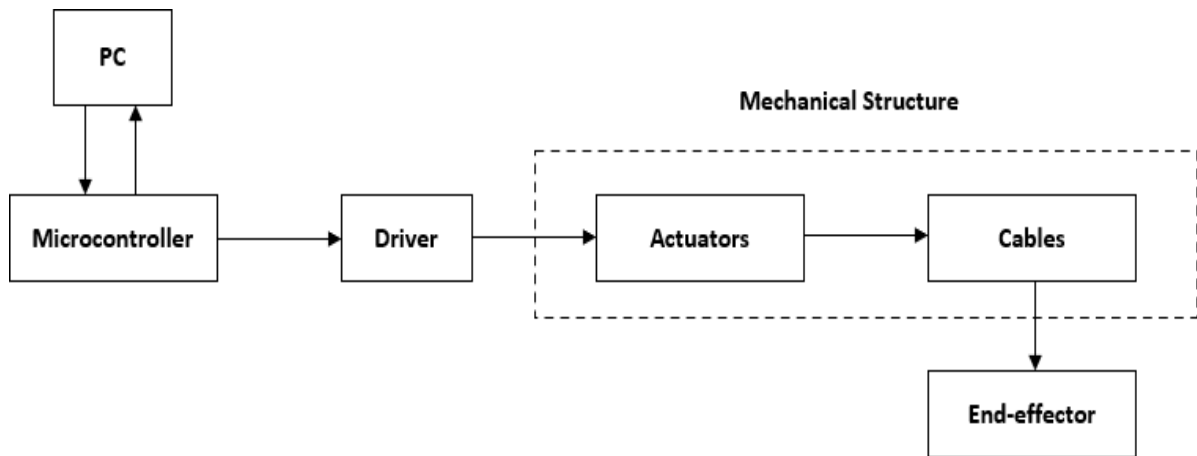


Figure 3.6: Block diagram for an operation of CDPR

For the beginning of the system, the microcontroller, IP camera and water nozzle are switched on. The IP camera will record and display the image through the PC. When the water pump is opened, the water will flow through the water nozzle and start for irrigation process. The signal from microcontroller will be sent via the stepper drivers to each the stepper motors. Then, the stepper motors will release or pull the cables based on the preset program. The end-effector start moving for implement the irrigation and remote monitoring system. The Figure 3.7 show the flowchart of the CDPR for smart farming.

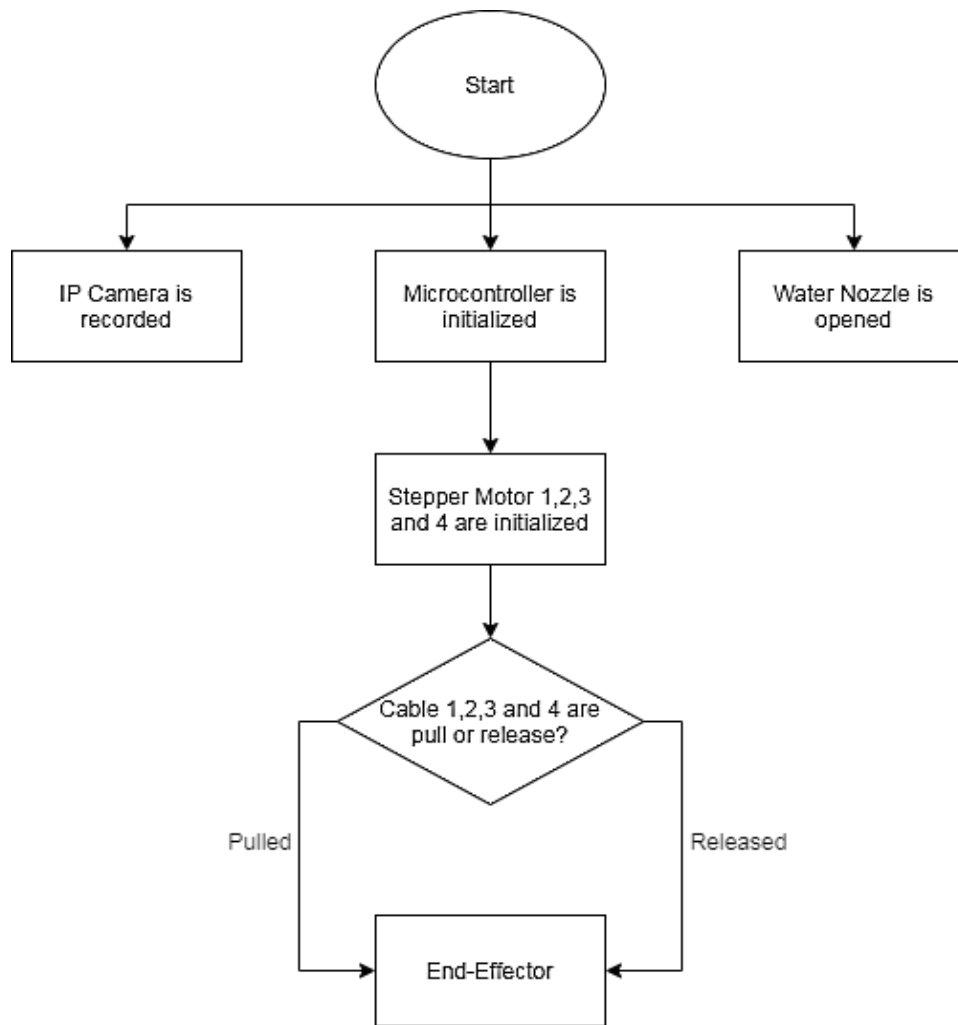


Figure 3.7: Flowchart for an operation CDPR

3.5 Schematic Diagram

This section explained the methods of the circuit connection of this project. The hardware and components for wiring are composed of Arduino Mega, NEMA 17 stepper motor, A4988 driver, resistors, capacitors and power source. The voltage supply to the stepper motor is 12v. The 10000k Ω resistors have been used to avoid the floating input of stepper driver. Furthermore, the 100uF capacitors have been used to protect the driver board from voltage spikes. The Figure 3.8 and Table 3.3 shown the schematic diagram and their pin connection respectively.

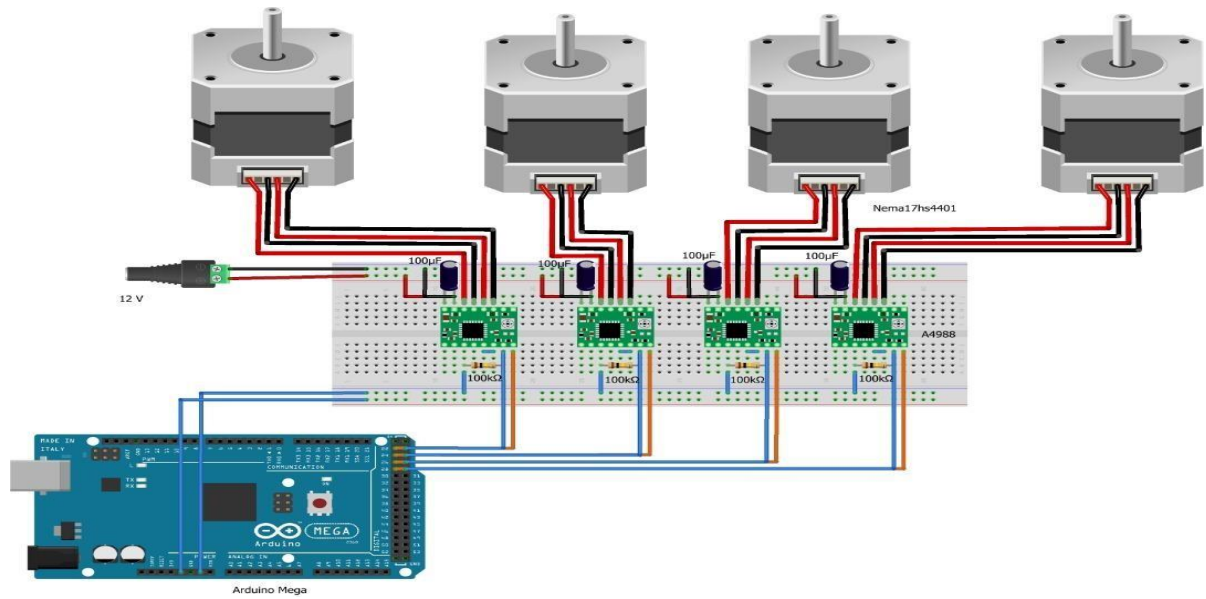


Figure 3.8 Schematic Diagram of CDPR

Table 3.3 Pin Connection [18]

INPUT/OUTPUT	ARDUINO MEGA PIN	CONNECTION
Output from Arduino Mega to Stepper Driver	Digital Pin 22 & 23 Digital Pin 24 & 25 Digital Pin 26 & 27 Digital Pin 28 & 29	DIR & STEP Pin on Driver 1 DIR & STEP Pin on Driver 2 DIR & STEP Pin on Driver 3 DIR & STEP Pin on Driver 4
Input power from Arduino Mega to Stepper Driver	5V Pin GND Pin	VDD Pin on Driver 1,2,3&4 GND Pin on Driver 1,2,3&4
INPUT/OUTPUT	NEMA 17 STEPPER MOTOR	CONNECTION
Output from NEMA 17 Stepper Motor to Stepper Driver	Motor1 Pin 2B, 2A, 1A & 1B Motor2 Pin 2B, 2A, 1A & 1B Motor3 Pin 2B, 2A, 1A & 1B Motor4 Pin 2B, 2A, 1A & 1B	2B, 2A, 1A & 1B Pin on Driver 1 2B, 2A, 1A & 1B Pin on Driver 2 2B, 2A, 1A & 1B Pin on Driver 3 2B, 2A, 1A & 1B Pin on Driver 4
INPUT/OUTPUT	A4988 STEPPER DRIVER	CONNECTION
Output from Stepper Driver to Resistor	Diver 1,2,3&4 STEP Pin	100kΩ from Driver 1, 2, 3 & 4 to GND
Output from Stepper Driver to Capacitor	Diver 1,2,3&4 VMOT Pin Diver 1,2,3&4 GND Pin	100μF Positive Pin 100μF Negative Pin
RST & SLP from Stepper Driver	Diver 1,2,3&4 RST & SLP Pin	RST to SLP Pin
INPUT/OUTPUT	POWER SUPPLY	CONNECTION
Input from Power Supply to Stepper Driver	12V GND	VMOT Pin from Driver 1,2,3&4 GND Pin from Driver 1,2,3&4

3.6 Conceptual Design of CDPR

The main goal of this work is to design the cable-driven parallel robot for smart farming. In this design, the four cables are needed to suspend the end-effector in which to swing the water nozzle and camera that located at the end-effector to the target position. There were several parts that attached on the end-effector such as water nozzle, camera, load, and water tube. The 3D modeling has been drawn to virtualize the construction of CDPR. This drawing has been considered the system balancing of the end-effector during the transmission process. The Figure 3.9 and 3.10 shown the CAD modeling structure of the robot (under-constrained).

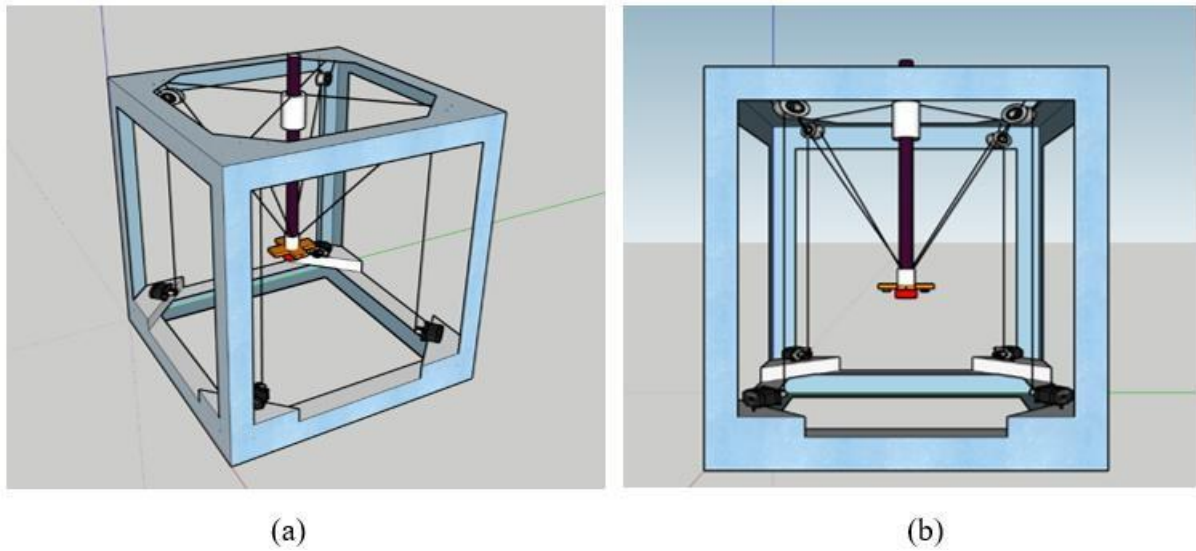


Figure 3.9 CAD Modeling Structure of CDPR: (a) 3-Dimensional View (b) Front View

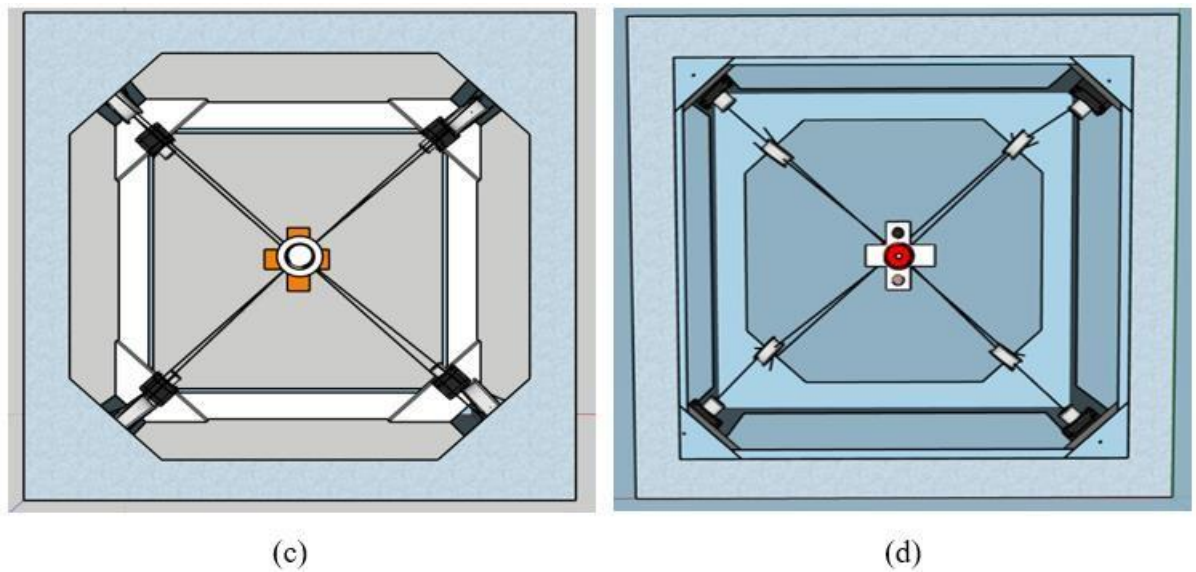


Figure 3.10 The Modeling of CDPR: (c) Top View (d) Bottom View

3.7 System Dimensional Optimization of CDPR

Modeling CDPR required the knowledge of several key system parameters such as width, depth, density, height and mass. The robot for smart farming has been scaled down in term of prototype. The Figure 3.11 shown the dimensions of frame and the end-effector. The full scale of CDPR's frame was 1:1.

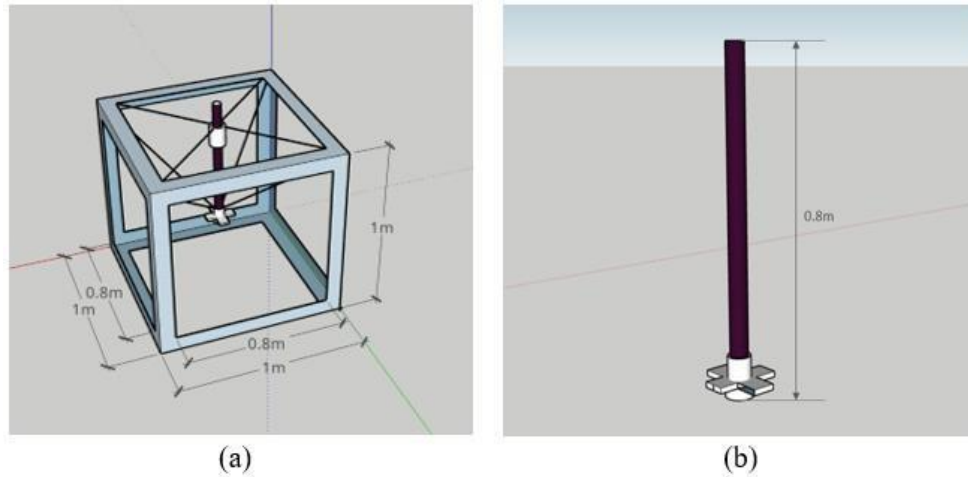


Figure 3.11: The Dimensions of CDPR

The dimensional of each part of robot should be considered based the capability or power of the actuators to suspend the load. In this design, the end-effector was attached with external load by the reason for balancing during the motion generation. The table below shown the full-scale system parameters of CDPR for smart farming.

Table 3.4 Full-Scale Parameters of CDPR

Parameters	Dimensions
Field width	1m
Field depth	0.8
Tower height	1m
Cable density	0.15 g/m
End-effector height	0.8
End-effector mass	300g

3.8 Motion Test

The motion test of the end-effector has been implemented after the robot has been built and all hardware and components were setup. The end-effector can be moved fast without the presence of camera for remote monitoring and water tube for irrigation process. However, after the camera and water tube were attached, the end effector was loss the stability during the transmission process. There are three factors that affected to the unbalancing of the end-effector have been listed: 1) Speed of the actuators, 2) Mass of the end- effector, and 3) External load (water tube).

Based on this observation, the ways to overcome the problem were developed. The speed of the actuators and the mass of the end-effector have been dropped by the reason to reduce the inertia and to avoid the sagging of cable during the transmission system. However, the unbalancing of the end-effector can be caused by the presence of water tube. Therefore, the center of mass balancing has been designed in this project.

3.9 Summary

In short, this chapter discussed the methodology of cable-driven parallel robot for smart farming. There were completely discussed the details of hardware and components and their functions as well. The type of application and software used to construct the robot such as Arduino IDE, Fritzing, AutoCAD and Little Stars were explained on this chapter. The block diagram of CDPR was created to describe the system's function of robot. The schematic diagram has been drawn and the pin connection has been shown on this chapter. Furthermore, 3D modeling of robot was drawn to virtualize the CDPR construction. The system dimensional of robot including the system parameters such as width, depth, height, density and mass were described. Lastly, the motion test of the end-effector was implemented and described on this chapter.

CHAPTER 4 RESULTS AND DISCUSSION

4.1 Introduction

The CDPR for smart farming was completely developed after the troubleshooting. In this chapter will be focused on the outcomes of CDPR for smart farming based on the methodology or experiment setup from chapter 3. The overall design of robot will be shown as well as their explanation of the design. The system of robot was demonstrated by inserted some plants inside the CDPR. As a smart farming system, the robot should be able to operate as remote monitoring, then at the same time implement the irrigation process. The analysis of this chapter will be focused on the performance of CDPR in term of the accuracy position of the end-effector in order of vertical and horizontal movement. In addition, the motion speed of the actuators and the end-effector will be calculated and analyzed on this chapter. The performance of remote monitoring system during the irrigation will be recorded as well.

4.2 Overall Design

The CDPR for smart farming has been demonstrated after the completely of troubleshooting. Some plants were put inside the CDPR for implementing the irrigation process and remote monitoring system. The Figure 4.1 shown the overall design of CDPR for smart farming.



Figure 4.1 Overall Design Of CDPR For Smart Farming

4.2.1 Mechanical Structure

The transmission system was performed via the mechanical structure of robot. The mechanical structure of CDPR was developed which composed of pulleys lifting and cables that attached between the stepper motor and the end-effector. The Figure 4.2 shown the mechanical structure of CDPR.



Figure 4.2 Mechanical Structure of CDPR

4.2.2 End-Effector

The end-effector was designed by using the cardboard and pipe connector. At the end-effector was located with the IP camera for remote monitoring and the water spray nozzle for irrigation process. The end-effector also was attached with the load by the reason to balance the end-effector during free motion. The Figure 4.3 shown the design of the end-effector



Figure 4.3 The Design Of End-Effector Of CDPR

4.2.3 Center of Mass Balancing

The center of mass balancing has been constructed to make the end-effector become more stable during the transmission system. The water tube was not able to connect with the end-effector directly by the reason of the unbalancing of end-effector during irrigation process. The designed was consists of rope, rubber, and PVC pipe. From this design, the end-effector can be moved smoothly along x, y and z plane. The figure below shown the design of mass balancing of the end-effector.

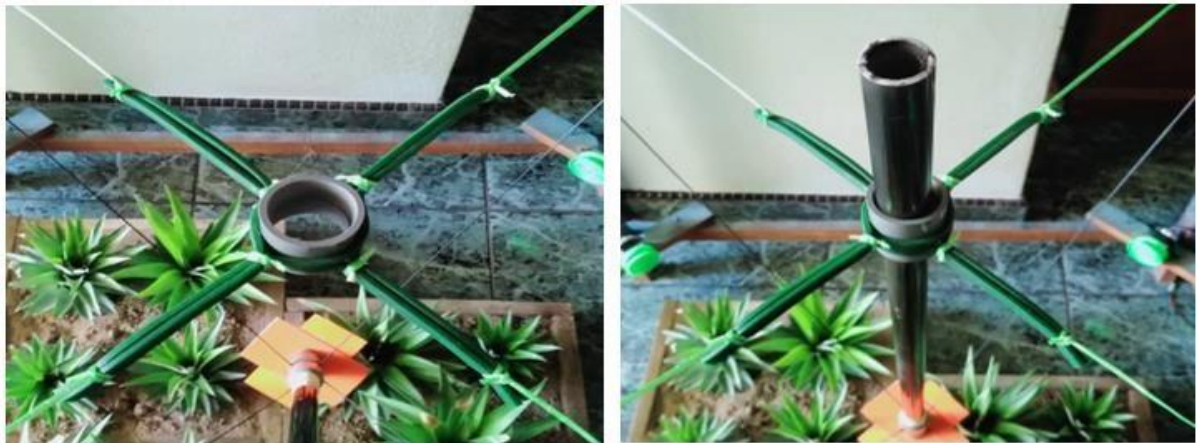


Figure 4.4 Center of Mass Balancing of the End-Effector

4.2.4 Water Tube Position

The water tube act as a source of the water flow. It was directly connected to the water pump and located on the center of mass balancing of the end-effector. The figure below shown the position of water tube for irrigation process.



Figure 4.5 The Position of Water Tube for Irrigation Process

4.2.5 Circuit Connection

The circuit connection has been developed based on the schematic diagram from Chapter 3. The Figure 4.6 shown the circuit connection of the robot.

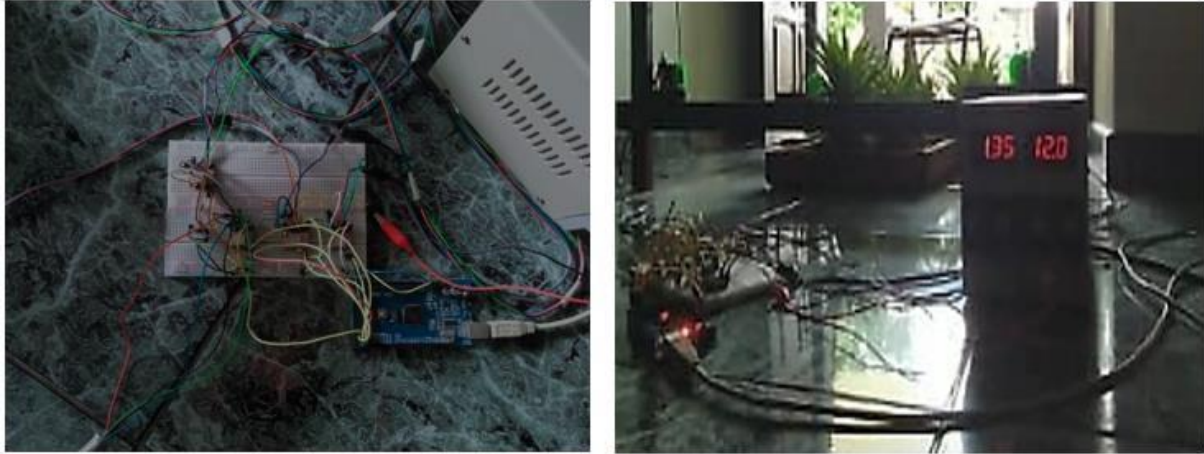


Figure 4.6 Circuit Connection

4.3 Motion Generation

The performance of CDPR in term of the accuracy position of the end-effector during the transmission process were recorded. The end-effector was able to follow the preset program (vertical, horizontal and curvilinear movement). Based on the robot design, the end-effector was able to move on the x, y and z-plane as well. The Figure 4.7 shown the direction of the end-effector to the target location.



Figure 4.7 The Direction of the End-Effector to Target Location

The body of the end-effector was suspended with the fix point as known as the center of mass balancing. Therefore, the movement of the end-effector during the transmission system would be followed on the curve line or in other words is curvilinear motion. The Figure 4.8 shown the movement of the end effector from the fix point.

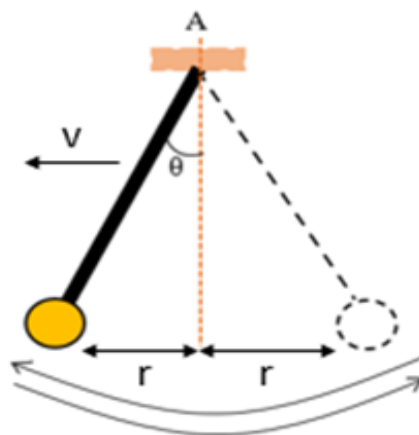


Figure 4.8 Curvilinear Translational motion

4.3.1 Motion Speed

The stepper motor that has been used as the actuator is a hybrid stepping motor with a 1.8 step angle or 200 steps per revolution. However, the type of stepper driver A4998 that has been used was set to quarter which mean the stepper motor need 800 steps for one revolution. Furthermore, each movement of the end-effector from one position to another position having 500 microsecond delays to avoid the cable sagging and reduce the inertia of the end-effector. In this section, the speed of actuator and the end-effector have been calculated by using the formula from Chapter 2:

$$800 \text{ steps} = 1 \text{ rev, for } 2.47 \text{ seconds}$$

$$\text{Steps per second} = 800/2.47 = 323.89 \approx 324$$

$$\text{Speed of actuator, } n = \frac{\text{Steps per second}}{\text{Steps per revolution}} \times 60 = \frac{324}{800} \times 60 = 24.3 \text{ RPM}$$

$$\text{Linear distance (in a second), } r = 7.2 \text{ cm} = 0.072 \text{ m}$$

$$\text{The linear velocity of end-effector, } v = r \times \frac{n \times 2\pi}{60} = 0.072 \times \frac{24.3 \times 2\pi}{60} = 0.18 \text{ m/s}$$

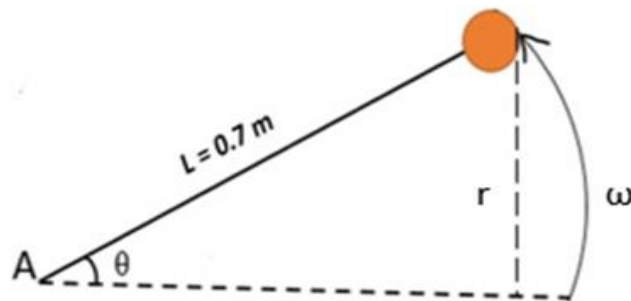


Figure 4.9 Rotation about a fixed axis

$$\text{Angular velocity of the end-effector, } m = \frac{v}{L} = \frac{0.18}{0.7} = 0.26 \text{ rad/s}$$

4.3.2 Association Between the Motion of End-Effector and Cables Length

In this section, the length of each cable has been recorded as shown in the Table 4.1 while the end effector was moved from one position to another position as shown in the Figure 4.10. The graph of cables length versus times have been plotted and the observation were recorded.

Based on the graph that shown in the Figure 4.11, the rising and falling point are not constant which caused from the time delay in the preset program. The control system of robot was set with the time delay to reduce inertia and avoid the cable sagging during the motion generation of the end-effector.

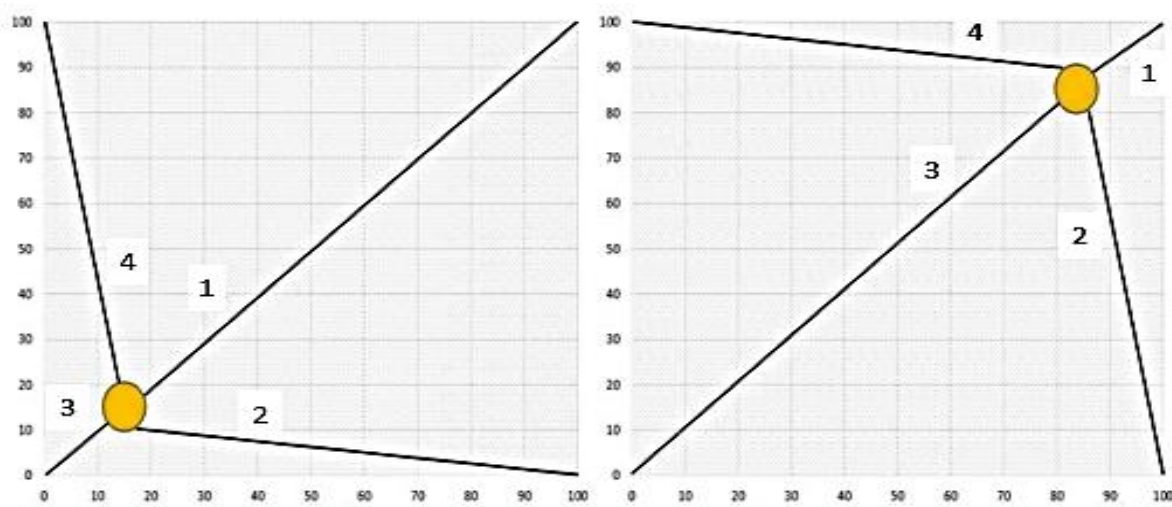


Figure 4.10 The Movement of the End-Effector

Table 4.1 The Measurement of Time (s) and Cables Length during the Motion Generation

Cable 1		Cable 2		Cable 3		Cable 4	
Time (s)	Length (cm)	Time (s)	Length (cm)	Time (s)	Length (cm)	Time (s)	Length (cm)
0	112	0	78	0	25	0	78
1	108	1	75	1	30	1	76
2	82	2	60	2	50	2	73
3	56	3	48	3	78	3	82
4	38	4	50	4	91	4	94
5	30	5	52	5	98	5	98

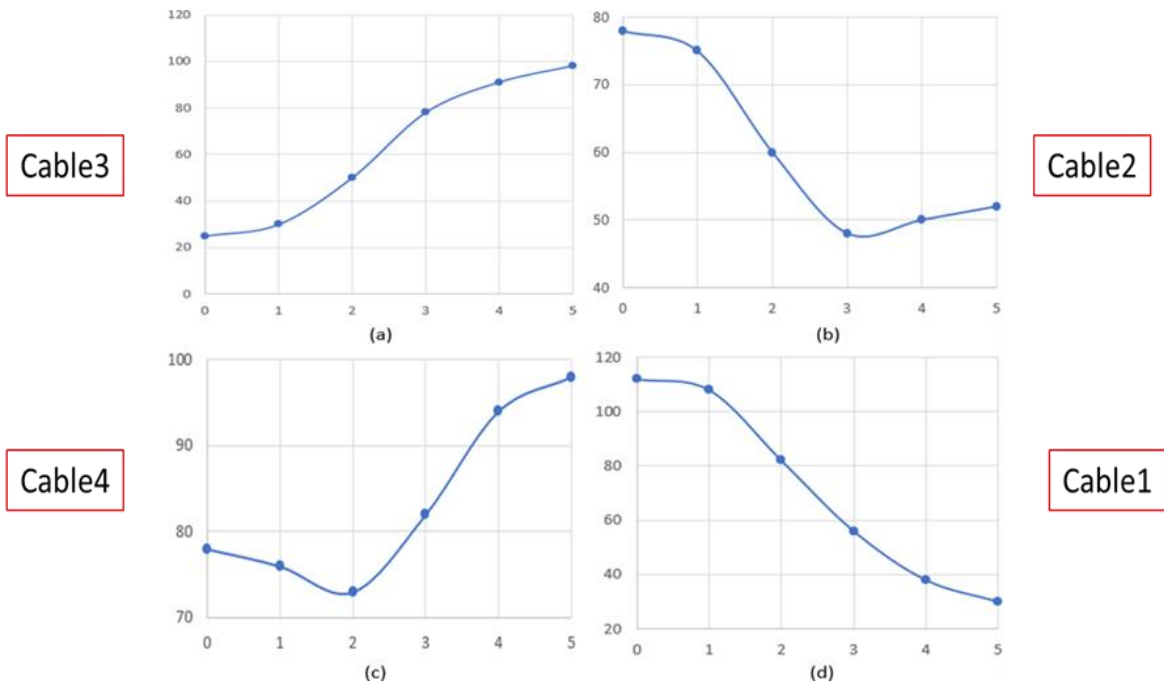


Figure 4.11 Graph of Cables Length (cm) Versus Time (s)

4.4 Remote Monitoring System

In remote monitoring system is use of the internet protocol between the camera and PC. The live video can be displayed through the PC during the irrigation or surveillance. The maximum distance of the camera from the monitor can be exceeded to 15 meters. The remote monitoring was operated as well during the irrigation process as show in the Figure 4.12.

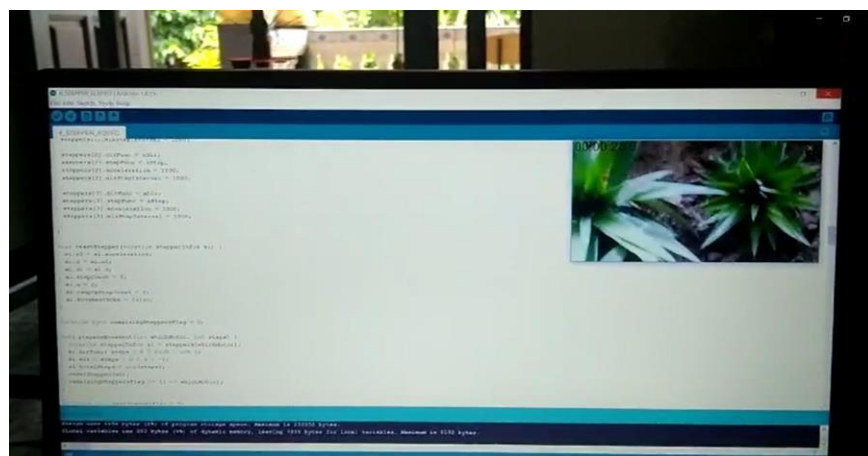


Figure 4.12 The Remote Monitoring System

4.5 Summary

Based on this chapter, the objectives of this project were achieved. The cable driven parallel robot for smart farming has been constructed. The system overall of robot has been shown as well as the explanation of their functions of each part such as mechanical structure design, end-effector, center of mass balancing, water tube position and circuit connection. The irrigation process and remote monitoring have been demonstrated and the results of robot performance were recorded. The motion generation of the robot have been analyzed including the speed of the actuators, linear and angular velocity of the end-effector and the association between the motion of the end-effector and cables length. Lastly, the remote monitoring was operated as well during the irrigation process.

CHAPTER 5 : CONCLUSION

5.1 Conclusion

In conclusion, the study of CDPR for smart farming which can operate as a remote monitoring and irrigation system were achieved. Based on the chapter 2, the current status of CDPR research has been presented including the mechanism of robot, working principle, derivation, advantages and disadvantages of robot.

Furthermore, based on the idea from research paper, the methods to construct the robot were successful described in Chapter 3. There were completely explained the details of hardware and components and their functions as well as the application and software used to construct the robot. In the chapter methodology also was successful described the block diagram, schematic diagram, 3D modeling of robot construction. There are three factors that affected to the unbalancing of the end-effector: 1) Speed of the actuators, 2) Mass of the end-effector, and 3) External load (water tube). The ways to overcome the problems were developed by the speed of the actuators and the mass of the end-effector have been dropped by the reason to reduce the inertia and to avoid the sagging of cable during the transmission system. However, the unbalancing of the end-effector can be caused by the presence of water tube. Therefore, the center of mass balancing has been designed to make the end-effector become more stable during transmission system.

Based on the Chapter 4, the cable driven parallel robot for smart farming has been constructed. The system of robot was demonstrated by inserted some plants inside the CDPR. The robot was successfully operated as remote monitoring, at the same time implemented the irrigation process. The analysis of this chapter will be focused on the performance of CDPR in term of the accuracy position of the end-effector in order of vertical, horizontal and curvilinear movement. In addition, the motion speed of the actuators and the end-effector during the transmission system have been calculated on this chapter. The motion generation of the robot have been analyzed including the speed of the actuators, linear and angular velocity of the end -effector and the graph of cables length versus time has been plotted.

Lastly, the CDPRs have a large potential in various applications, such as agriculture irrigation, remote monitoring, clean-up of the dirty area, manipulation of heavy payloads, and rescue systems. However, this type of robot (under-constrained) have the limitation due to the main characteristics of the cables. The actuators are only can pull and release the end-effector through cables but cannot push it.

5.2 Future Development

The system design of CDPR can be further optimized to make it more efficiency and can be used in real life. Based on the current CDPR for smart farming in this project, the small workspace of robot cannot be used in real life or agriculture field. Firstly, CDPR should be increased in term of robot's size including the efficiency of the hardware and components for used in the large workspace or in agriculture field.

Secondly, the actuators should have the higher value of torque by the reason to be able to hold the higher mass of the end-effector that attached with external load (e.g., camera, water nozzle, water tube and etc.) without tilted or unbalance during the transmission system.

Lastly, the GUI system should be created to control the motion of the end-effector. This will allow the administrators or users to be able to control the motion of the end-effector without the needed of writing program too many times. This will be able to save the times of the users or administrators.

REFERENCES

- [1] T. Tjahjowidodo, K. Zhu, W. Dailey, E. Burdet, D. Campolo, Multi-source micro-friction identification for a class of cable-driven robots with passive backbone, in: *Mechanical Systems and Signal Processing*, Elsevier, 2016, pp. 152–165, vol. 80.
- [2] A.M. Pinto , E. Moreira , J. Lima , J.P. Sousa , P. Costa , A cable-driven robot for architectural constructions : a visual-guided approach for motion control and path-planning, *Auton Robots* 41 (7) (2017) 1487–1499.
- [3] D. Lau, D. Oetomo and S. K. Halgamuge, "Inverse Dynamics of Multilink Cable-Driven Manipulators With the Consideration of Joint Interaction Forces and Moments," in *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 479-488, April 2015, doi: 10.1109/TRO.2015.2394498.
- [4] P. Miermeister, M. Lächele, R. Boss, C. Masone, C. Schenk, J. Tesch, M. Kerger, H. Teufel, A. Pott, and H. H. Bühlhoff, "The cablerobot simulator large scale motion platform based on cable robot technology," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2016, pp. 3024–3029.
- [5] J. P. Sousa, C. G. Palop, E. Moreira, A. M. Pinto, J. Lima, P. Costa, P. Costa, G. Veiga, and A. P. Moreira, "The spiderobot: A cable-robot system for on-site construction in architecture," in *Robotic Fabrication in Architecture, Art and Design*, 2016, pp. 230–239.
- [6] D. Lau, J. Eden, D. Oetomo, and S. K. Halgamuge, "Musculoskeletal static workspace of the human shoulder as a cable-driven robot," *IEEE Trans. on Mechatronics*, vol. 20, no. 2, pp. 978–984, 2015.
- [7] D. Lau, D. Oetomo and S. K. Halgamuge, "Generalized Modeling of Multilink Cable-Driven Manipulators With Arbitrary Routing Using the Cable-Routing Matrix," in *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1102-1113, Oct. 2013, doi: 10.1109/TRO.2013.2264866.
- [8] C. Alias, I. Nikolaev, E. G. Correa Magallanes and B. Noche, "An Overview of Warehousing Applications based on Cable Robot Technology in Logistics," 2018 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI), 2018, pp. 232-239, doi: 10.1109/SOLI.2018.8476760.
- [9] El-Ghazaly G., Gouttefarde M., Creuze V. (2015) Adaptive Terminal Sliding Mode Control of a Redundantly-Actuated Cable-Driven Parallel Manipulator: CoGiRo. In: Pott A., Bruckmann T. (eds) *Cable-Driven Parallel Robots. Mechanisms and Machine Science*, vol 32. Springer, Cham. https://doi.org/10.1007/978-3-319-09489-2_13.

- [10] Sun Y., Newman M., Zygielbaum A., Terry B. (2019) Active Vibration Damping of a Cable-Driven Parallel Manipulator Using a Multirotor System. In: Pott A., Bruckmann T. (eds) Cable-Driven Parallel Robots. CableCon 2019. Mechanisms and Machine Science, vol 74. Springer, Cham. https://doi.org/10.1007/978-3-030-20751-9_34.
- [11] H. Wang, J. Kinugawa and K. Kosuge, "Exact Kinematic Modeling and Identification of Reconfigurable Cable-Driven Robots With Dual-Pulley Cable Guiding Mechanisms," in IEEE/ASME Transactions on Mechatronics, vol. 24, no. 2, pp. 774-784, April 2019, doi: 10.1109/TMECH.2019.2899016.
- [12] SA Khalilpour, A Bourbour, R Khorrambakht, S Kariminasab, and HD Taghirad. Forward kinematics resolution of a deployable cable robot. In 2017 5th RSI International Conference on Robotics and Mechatronics (ICRoM), pages 27–32. IEEE, 2017.
- [13] T. Zheng et al., "Self-Calibration of Cable Driven Continuum Robot," 2018 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2018, pp. 2498-2503, doi: 10.1109/ROBIO.2018.8664763.
- [14] SA Khalilpour, R Khorrambakht, HD Taghirad, and Philippe Cardou. Robust cascade control of a deployable cable-driven robot. Mechanical Systems and Signal Processing, 127:513–530, 2019.
- [15] Preiss JA, Hausman K, Sukhatme GS, Weiss S. Simultaneous self-calibration and navigation using trajectory optimization. The International Journal of Robotics Research. 2018;37(13-14):1573-1594. doi:10.1177/0278364918781734.
- [16] Furet M., Lettl M., Wenger P. (2019) Kinematic Analysis of Planar Tensegrity 2-X Manipulators. In: Lenarcic J., Parenti-Castelli V. (eds) Advances in Robot Kinematics 2018. ARK 2018. Springer Proceedings in Advanced Robotics, vol 8. Springer, Cham. https://doi.org/10.1007/978-3-319-93188-3_18.
- [17] Baklouti, S., Courteille, E., Caro, S., and Dkhil, M. (October 12, 2017). "Dynamic and Oscillatory Motions of Cable-Driven Parallel Robots Based on a Nonlinear Cable Tension Model." ASME. J. Mechanisms Robotics. December 2017; 9(6): 061014. <https://doi.org/10.1115/1.4038068>.
- [18] Coordinated stepper motor control (Published on 20/2/2020).
<https://www.cuidevices.com/product/resource/nema17-amt112s.pdf>

APPENDIX A: CODING

```
#define X_DIR_PIN      22
#define X_STEP_PIN     23
#define Y_DIR_PIN      24
#define Y_STEP_PIN     25
#define Z_DIR_PIN      26
#define Z_STEP_PIN     27
#define A_DIR_PIN      28
#define A_STEP_PIN     29
```

```
#define X_STEP_HIGH    PORTA |= 0b00000010;
#define X_STEP_LOW     PORTA &= ~0b00000010;
#define Y_STEP_HIGH    PORTA |= 0b00001000;
#define Y_STEP_LOW     PORTA &= ~0b00001000;
#define Z_STEP_HIGH    PORTA |= 0b00100000;
#define Z_STEP_LOW     PORTA &= ~0b00100000;
#define A_STEP_HIGH    PORTA |= 0b10000000;
#define A_STEP_LOW     PORTA &= ~0b10000000;
```

```
#define TIMER1_INTERRUPTS_ON  TIMSK1 |= (1 << OCIE1A);
#define TIMER1_INTERRUPTS_OFF TIMSK1 &= ~(1 << OCIE1A);
```

```
struct stepperInfo {
    // externally defined parameters
    float acceleration;
    volatile unsigned int minStepInterval; // ie. max speed, smaller is faster
    void (*dirFunc)(int);
    void (*stepFunc)();

    // derived parameters
    unsigned int c0;           // step interval for first step, determines acceleration
    long stepPosition;         // current position of stepper (total of all movements taken so far)

    // per movement variables (only changed once per movement)
    volatile int dir;          // current direction of movement, used to keep track of
    position
    volatile unsigned int totalSteps; // number of steps requested for current movement
    volatile bool movementDone = false; // true if the current movement has been completed
    (used by main program to wait for completion)
    volatile unsigned int rampUpStepCount; // number of steps taken to reach either max
    speed, or half-way to the goal (will be zero until this number is known)

    // per iteration variables (potentially changed every interrupt)
```



```

    volatile unsigned int n;          // index in acceleration curve, used to calculate next
interval
    volatile float d;                 // current interval length
    volatile unsigned long di;        // above variable truncated
    volatile unsigned int stepCount;  // number of steps completed in current movement
};

void xStep() {
    X_STEP_HIGH
    X_STEP_LOW
}
void xDir(int dir) {
    digitalWrite(X_DIR_PIN, dir);
}

void yStep() {
    Y_STEP_HIGH
    Y_STEP_LOW
}
void yDir(int dir) {
    digitalWrite(Y_DIR_PIN, dir);
}

void zStep() {
    Z_STEP_HIGH
    Z_STEP_LOW
}
void zDir(int dir) {
    digitalWrite(Z_DIR_PIN, dir);
}

void aStep() {
    A_STEP_HIGH
    A_STEP_LOW
}
void aDir(int dir) {
    digitalWrite(A_DIR_PIN, dir);
}

void resetStepperInfo( stepperInfo& si ) {
    si.n = 0;
    si.d = 0;
    si.di = 0;
    si.stepCount = 0;
    si.rampUpStepCount = 0;
    si.totalSteps = 0;
    si.stepPosition = 0;
    si.movementDone = false;
}

```

```

}

#define NUM_STEPPERS 4
volatile stepperInfo steppers[NUM_STEPPERS];

void setup() {
  pinMode(X_STEP_PIN, OUTPUT);
  pinMode(X_DIR_PIN, OUTPUT);
  pinMode(Y_STEP_PIN, OUTPUT);
  pinMode(Y_DIR_PIN, OUTPUT);
  pinMode(Z_STEP_PIN, OUTPUT);
  pinMode(Z_DIR_PIN, OUTPUT);
  pinMode(A_STEP_PIN, OUTPUT);
  pinMode(A_DIR_PIN, OUTPUT);

  noInterrupts();
  TCCR1A = 0;
  TCCR1B = 0;
  TCNT1 = 0;
  OCR1A = 1000; // compare value
  TCCR1B |= (1 << WGM12); // CTC mode
  TCCR1B |= ((1 << CS11) | (1 << CS10)); // 64 prescaler

  interrupts();

  steppers[0].dirFunc = xDir;
  steppers[0].stepFunc = xStep;
  steppers[0].acceleration = 1000;
  steppers[0].minStepInterval = 1000;

  steppers[1].dirFunc = yDir;
  steppers[1].stepFunc = yStep;
  steppers[1].acceleration = 1000;
  steppers[1].minStepInterval = 1000;

  steppers[2].dirFunc = zDir;
  steppers[2].stepFunc = zStep;
  steppers[2].acceleration = 1000;
  steppers[2].minStepInterval = 1000;

  steppers[3].dirFunc = aDir;
  steppers[3].stepFunc = aStep;
  steppers[3].acceleration = 1000;
  steppers[3].minStepInterval = 1000;
}

```

```

void resetStepper(volatile stepperInfo& si) {
    si.c0 = si.acceleration;
    si.d = si.c0;
    si.di = si.d;
    si.stepCount = 0;
    si.n = 0;
    si.rampUpStepCount = 0;
    si.movementDone = false;
}

volatile byte remainingSteppersFlag = 0;

void prepareMovement(int whichMotor, int steps) {
    volatile stepperInfo& si = steppers[whichMotor];
    si.dirFunc( steps < 0 ? HIGH : LOW );
    si.dir = steps > 0 ? 1 : -1;
    si.totalSteps = abs(steps);
    resetStepper(si);
    remainingSteppersFlag |= (1 << whichMotor);
}

volatile byte nextStepperFlag = 0;
volatile int ind = 0;
volatile unsigned int intervals[100];

void setNextInterruptInterval() {

    bool movementComplete = true;

    unsigned int mind = 999999;
    for (int i = 0; i < NUM_STEPPERS; i++) {
        if ( ((1 << i) & remainingSteppersFlag) && steppers[i].di < mind ) {
            mind = steppers[i].di;
        }
    }

    nextStepperFlag = 0;
    for (int i = 0; i < NUM_STEPPERS; i++) {
        if ( ! steppers[i].movementDone )
            movementComplete = false;

        if ( ((1 << i) & remainingSteppersFlag) && steppers[i].di == mind )
            nextStepperFlag |= (1 << i);
    }

    if ( remainingSteppersFlag == 0 ) {
        OCR1A = 65500;
    }
}

```

```

OCR1A = mind;
}

ISR(TIMER1_COMPA_vect)
{
    unsigned int tmpCtr = OCR1A;

    OCR1A = 65500;

    for (int i = 0; i < NUM_STEPPERS; i++) {

        if ( !( (1 << i) & remainingSteppersFlag ) )
            continue;

        if ( !( nextStepperFlag & (1 << i) ) ) {
            steppers[i].di -= tmpCtr;
            continue;
        }

        volatile stepperInfo& s = steppers[i];

        if ( s.stepCount < s.totalSteps ) {
            s.stepFunc();
            s.stepCount++;
            s.stepPosition += s.dir;
            if ( s.stepCount >= s.totalSteps ) {
                s.movementDone = true;
                remainingSteppersFlag &= ~(1 << i);
            }
        }

        if ( s.rampUpStepCount == 0 ) {
            s.n++;
            s.d = s.d - (2 * s.d) / (4 * s.n + 1);
            if ( s.d <= s.minStepInterval ) {
                s.d = s.minStepInterval;
                s.rampUpStepCount = s.stepCount;
            }
            if ( s.stepCount >= s.totalSteps / 2 ) {
                s.rampUpStepCount = s.stepCount;
            }
        }
        else if ( s.stepCount >= s.totalSteps - s.rampUpStepCount ) {
            s.d = (s.d * (4 * s.n + 1)) / (4 * s.n + 1 - 2);
            s.n--;
        }

        s.di = s.d; // integer
    }

    setNextInterruptInterval();
}

```

```

    TCNT1 = 0;
}

void runAndWait() {
    setNextInterruptInterval();
    while ( remainingSteppersFlag );
}

void loop() {

    TIMER1_INTERRUPTS_ON

    ////////////x-plane & y-plane//////////

    for (int i = 0; i < NUM_STEPPERS; i++)
        prepareMovement( 0, -300 );
        prepareMovement( 1, -300 );
        prepareMovement( 2, -300 );
        prepareMovement( 3, -300 );
    runAndWait();
    delay(1000);

    //////////////////////////////////////
    for (int i = 0; i < NUM_STEPPERS; i++)
        prepareMovement( 0, 400 );
        prepareMovement( 1, 400 );
        prepareMovement( 2, -400 );
        prepareMovement( 3, -400 );
    runAndWait();
    delay(500);

    //////////////////////////////////////
    for (int i = 0; i < NUM_STEPPERS; i++)
        prepareMovement( 0, -300 );
        prepareMovement( 1, 300 );
        prepareMovement( 2, 300 );
        prepareMovement( 3, -300 );
    runAndWait();
    delay(500);

    for (int i = 0; i < NUM_STEPPERS; i++)
        prepareMovement( 0, 600 );
        prepareMovement( 1, -600 );
        prepareMovement( 2, -600 );
        prepareMovement( 3, 600 );
    runAndWait();
    delay(500);

    for (int i = 0; i < NUM_STEPPERS; i++)
        prepareMovement( 0, -300 );

```

```

    prepareMovement( 1, 300 );
    prepareMovement( 2, 300 );
    prepareMovement( 3, -300 );
    runAndWait();
    delay(500);

```

```

////////////////////////////////////
for (int i = 0; i < NUM_STEPPERS; i++)
    prepareMovement( 0, -200 );
    prepareMovement( 1, -200 );
    prepareMovement( 2, 200 );
    prepareMovement( 3, 200 );
    runAndWait();
    delay(500);

```

```

////////////////////////////////////
for (int i = 0; i < NUM_STEPPERS; i++)
    prepareMovement( 0, -300 );
    prepareMovement( 1, 300 );
    prepareMovement( 2, 300 );
    prepareMovement( 3, -300 );
    runAndWait();
    delay(500);

```

```

for (int i = 0; i < NUM_STEPPERS; i++)
    prepareMovement( 0, 600 );
    prepareMovement( 1, -600 );
    prepareMovement( 2, -600 );
    prepareMovement( 3, 600 );
    runAndWait();
    delay(500);

```

```

for (int i = 0; i < NUM_STEPPERS; i++)
    prepareMovement( 0, -300 );
    prepareMovement( 1, 300 );
    prepareMovement( 2, 300 );
    prepareMovement( 3, -300 );
    runAndWait();
    delay(500);

```

```

////////////////////////////////mid////////////////////////////////
for (int i = 0; i < NUM_STEPPERS; i++)
    prepareMovement( 0, -200 );
    prepareMovement( 1, -200 );
    prepareMovement( 2, 200 );
    prepareMovement( 3, 200 );
    runAndWait();
    delay(500);

```

```

////////////////////////////////////
for (int i = 0; i < NUM_STEPPERS; i++)

```

```

    prepareMovement( 0, -300 );
    prepareMovement( 1, 300 );
    prepareMovement( 2, 300 );
    prepareMovement( 3, -300 );
runAndWait();
delay(500);

for (int i = 0; i < NUM_STEPPERS; i++)
    prepareMovement( 0, 600 );
    prepareMovement( 1, -600 );
    prepareMovement( 2, -600 );
    prepareMovement( 3, 600 );
runAndWait();
delay(500);

for (int i = 0; i < NUM_STEPPERS; i++)
    prepareMovement( 0, -300 );
    prepareMovement( 1, 300 );
    prepareMovement( 2, 300 );
    prepareMovement( 3, -300 );
runAndWait();
delay(500);

////////////////////////////////////
for (int i = 0; i < NUM_STEPPERS; i++)
    prepareMovement( 0, -200 );
    prepareMovement( 1, -200 );
    prepareMovement( 2, 200 );
    prepareMovement( 3, 200 );
runAndWait();
delay(500);

////////////////////////////////////
for (int i = 0; i < NUM_STEPPERS; i++)
    prepareMovement( 0, -300 );
    prepareMovement( 1, 300 );
    prepareMovement( 2, 300 );
    prepareMovement( 3, -300 );
runAndWait();
delay(500);

for (int i = 0; i < NUM_STEPPERS; i++)
    prepareMovement( 0, 600 );
    prepareMovement( 1, -600 );
    prepareMovement( 2, -600 );
    prepareMovement( 3, 600 );
runAndWait();
delay(500);

for (int i = 0; i < NUM_STEPPERS; i++)
    prepareMovement( 0, -300 );

```

```

    prepareMovement( 1, 300 );
    prepareMovement( 2, 300 );
    prepareMovement( 3, -300 );
    runAndWait();
    delay(500);

```

```

////////////////////////////////////

```

```

    for (int i = 0; i < NUM_STEPPERS; i++)
        prepareMovement( 0, -200 );
    prepareMovement( 1, -200 );
    prepareMovement( 2, 200 );
    prepareMovement( 3, 200 );
    runAndWait();
    delay(500);

```

```

////////////////////////////////////

```

```

    for (int i = 0; i < NUM_STEPPERS; i++)
        prepareMovement( 0, -300 );
    prepareMovement( 1, 300 );
    prepareMovement( 2, 300 );
    prepareMovement( 3, -300 );
    runAndWait();
    delay(500);

```

```

    for (int i = 0; i < NUM_STEPPERS; i++)
        prepareMovement( 0, 600 );
    prepareMovement( 1, -600 );
    prepareMovement( 2, -600 );
    prepareMovement( 3, 600 );
    runAndWait();
    delay(500);

```

```

    for (int i = 0; i < NUM_STEPPERS; i++)
        prepareMovement( 0, -300 );
    prepareMovement( 1, 300 );
    prepareMovement( 2, 300 );
    prepareMovement( 3, -300 );
    runAndWait();
    delay(500);

```

```

////////////////////////////////////

```

```

    for (int i = 0; i < NUM_STEPPERS; i++)
        prepareMovement( 0, 400 );
    prepareMovement( 1, 400 );
    prepareMovement( 2, -400 );
    prepareMovement( 3, -400 );
    runAndWait();
    delay(1000);

```

```

//////////z-axis//////////

```

```

    for (int i = 0; i < NUM_STEPPERS; i++)

```



```
    prepareMovement( 0, 300 );  
    prepareMovement( 1, 300 );  
    prepareMovement( 2, 300 );  
    prepareMovement( 3, 300 );  
runAndWait();  
  
while (true);  
  
}
```