DEVELOPMENT AND ANALYSIS OF AN EMBEDDED SOFTWARE FOR CILI-PADI PICKING ROBOT FRAMEWORK USING MATLAB

DANIAL 'AFIF BIN MOHD ZAKI

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

DEVELOPMENT AND ANALYSIS OF AN EMBEDDED SOFTWARE FOR CILI-PADI PICKING ROBOT FRAMEWORK USING MATLAB WHICH HAS BEEN APPROVED BY FACULTY OF ELECTRONIC AND COMPUTER ENGINEERING (FKEKK)

DANIAL 'AFIF BIN MOHD ZAKI

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

This report is submitted in partial fulfilment of the requirements for the degree of Bachelor of Electronic Engineering with Honours

Faculty of Electronic and Computer Engineering
Universiti Teknikal Malaysia Melaka

2023

DECLARATION

I declare that this report entitled "**DEVELOPMENT AND ANALYSIS OF AN EMBEDDED SOFTWARE FOR CILI-PADI PICKING ROBOT FRAMEWORK USING MATLAB**" is the result of my own work except for quotes as cited in the references.

UTeM
اونيۏرسيتي تيکنيکل مليسيا ملاک
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Signature : .....

Author : DANIAL 'AFIF BIN MOHD ZAKI

Date : 13/1/2023

APPROVAL

I hereby declare that I have read this thesis and in my opinion this thesis is sufficient in terms of scope and quality for the award of Bachelor of Electronic Engineering with Honours.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Signature : .....

Supervisor Name : Zarina Mohd Noh

Date : 23rd January 2023

DEDICATION

I would want to thank everyone who helped make this project a reality, whether they were directly or indirectly involved. It's possible that I wouldn't have been able to finish all of this work without their help and encouragement. To express my gratitude for Dr. Zarina binti Mohd Noh's trust in me to complete my assigned thesis, I am happy to do so on this occasion. In addition, I would want to express my gratitude to Dr. Wira Hidayat bin Mohd Saad and all of the lecturers that helped me complete this PSM. I'd also like to thank everyone who helped make this assignment possible, especially my lecturer, Dr. Zarina binti Mohd Noh, for the guidance she provided over the course of this assignment and also other lecture who helped me with the PSM and this dissertation are also mentioned. I can't forget to thank my fellow soldiers, who have been a tremendous source of support and collaboration during the course of our mission. Finally, I'd want to thank all of my friends who helped me complete this project. Thanks a lot.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

ABSTRACT

The primary focuses of this research are the development, implementation, and verification of embedded software for robotic systems. The Cili-Padi Picking Robot places a significant amount of attention on being able to function all actuators and sensors by making use of MATLAB and having the ability to communicate with a GUI. However, in order to reach its full potential, the robot needs to be equipped with AI in its visual system so that it can recognise mature chilies. However, the focus of this project is on embedded systems for sensors and actuators in robots. This is accomplished by connecting MATLAB and Arduino so that they are capable of communicating with each other thanks to the utilisation of serial communication, which enables both programmes to send and receive data. Prior to that, it needs to be done for the purpose of robot analysis, which includes determining what actuators and sensors are utilised as well as what data will be used regardless of whether or not the parameters are modified. Additionally, this results in the formation of many subdivisions for the system that is integrated into the robot. Systems comparable to these include a manual navigation system, a Robot Arm system, a visual robot system to detect Chile, as well as other systems of a similar nature.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

ABSTRAK

Fokus utama penyelidikan ini ialah pembangunan, pelaksanaan dan pengesahan perisian terbenam untuk sistem robotik. Robot Pemetik Cili Padi memberi perhatian yang besar kepada keupayaan untuk berfungsi semua penggerak dan penderia dengan menggunakan MATLAB dan mempunyai keupayaan untuk berkomunikasi dengan GUI. Bagaimanapun, untuk mencapai potensi sepenuhnya, robot itu perlu dilengkapi dengan AI dalam sistem visualnya supaya ia dapat mengenali cili matang. Walau bagaimanapun, tumpuan projek ini adalah pada sistem terbenam untuk penderia dan penggerak dalam robot. Ini dicapai dengan menyambungkan MATLAB dan Arduino supaya mereka mampu berkomunikasi antara satu sama lain berkat penggunaan komunikasi bersiri, yang membolehkan kedua-dua program menghantar dan menerima data. Sebelum itu, ia perlu dilakukan untuk tujuan analisis robot, termasuk menentukan apakah penggerak dan penderia yang digunakan serta apakah data yang akan digunakan tanpa mengira sama ada parameter diubah suai atau tidak. Selain itu, ini mengakibatkan pembentukan banyak subbahagian untuk sistem yang disepadukan ke dalam robot. Sistem yang setanding dengan ini termasuk sistem navigasi manual, sistem Lengan Robot, sistem robot visual untuk mengesan Chile, serta sistem lain yang serupa.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

ACKNOWLEDGEMENTS

I would like to thank everyone who contributed directly or indirectly to the accomplishment of this work. Without their assistance and support, it may not have been feasible to complete all of these responsibilities. With this occasion, I am pleased to convey my gratitude to Dr. Zarina binti Mohd Noh, the principal Supervisor of PSM, for placing her confidence in me to complete the assigned thesis. I would also like to thank everyone who worked tirelessly to assist in the completion of this assignment, especially my professor, Dr. Zarina binti Mohd Noh, for the direction she provided during the period of this assignment and also other professors who assisted me in completing the PSM and this dissertation are also acknowledged. Not forgetting my comrades-in-arms, as they have provided a great deal of assistance and cooperation in completing this mission effectively. Many thanks.

TABLE OF CONTENTS

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

LIST OF FIGURES

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

LIST OF TABLES

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

LIST OF SYMBOLS AND ABBREVIATIONS

ROS	:	Robot Operating System
GUI	:	Graphical User Interface
TICS	:	Task-Instruction-Contingency-Shaping

UTeM
اونيۏرسيتي تيکنيکل مليسيا ملاک
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

LIST OF APPENDICES

UTeM
اونيۏرسيتي تيکنيکل مليسيا ملاك
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

CHAPTER 1

INTRODUCTION

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

This chapter describes the idea general idea for this project which includes the background of the project, problem statement, objectives, scope of the project, and the report outline.

1.1 Project Background

"Industry 4.0" refers to the transition to cyber-physical systems, which marks the fourth major industrial revolution [1]. In the forth industrial revolution, the application of robot had been widely accepted to ease the work in the industry such as for agriculture propose. Because of this, the goal of this project is to find solutions to the challenges faced in the agriculture by constructing a robot structure with the capability of assisting farmer in tasks that can reducing their workload and reducing the amount of time they spend on such tasks. Therefore, this project is about how to build a robot framework through embedded software. By using the appropriate software to integrate the software for agriculture purpose, specifically for Cili Padi Picking Robot.

Selection of the right framework and middleware for software and hardware integration is a difficult process that can have far-reaching consequences for any robotics framework. In most cases, computational and memory resources are limited in the robotics software. The goal of this project is to create a framework through which embedded software like MATLAB can communicate to hardware like Arduino and demonstrate its application for robot's framework that can be used in agriculture. This project is also to run simulations with software and hardware also involve software simulation and hardware demonstration in real-world environment.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

1.2 Problem Statement

Malaysia should encourage rice and chilli farming due to its struggling economy [2]. An increasing population and worldwide market demand agricultural products. Malaysia needs a more advanced and reliable food production system to address food sustainability and minimize dependence on imported basic food supply.

Upgrade seeds, pesticides, insecticides, fertilizers, and farming procedures. After locating the chili, the robot arm can pick up the plants one by one. The robot's durability and endurance will outperform a human's speed. Preventing chilies from bruising and lowering crop quality depends on the robot arm and fruit picking process.

Robots are made up of numerous functional components and require a variety of specific talents. These algorithms may be tested in either a simulated or real-world environment. An experimental setup with real parts can be used to test a design, but this approach is both costly and limiting in terms of exploratory study. In addiction it is performed time-consuming to design a robot since each component has a distinct function, such as developing architecture robots, creating visual images, and creating arm robots. Hence, a simulation framework can cut down on the time, effort, and money needed to demonstrate working prototype of robot software [3].

In this project, the simulate framework will be integrated hardware involved in an embedded software to keep track of robot's system. The robotic system demonstrated of this prototype with utilized the Cili Padi Picking Robot, as an example one of the systems that can be used in agriculture.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

1.3 Objectives

The objectives of this project are to:

1. Analyze and broken down a Cili-Padi Picking robot framework large systems into smaller and more manageable pieces.

2. Develop an embedded software part of the Cili-Padi Picking robot function using MATLAB software (Simulink and App Designer) that can be interfaced with Graphical User Interface (GUI).
3. Demonstrate the functionality of the embedded software in real time.

1.4 Scope of Project

The scope of this project to analyses a robotic framework of a Analysis part of Robot Cili-Padi Picking Robot and develop its framework through the use of MATLAB Simulink software communicate the Arduino using serial communication and freeRTOS. The hardware prototype of the robot demonstration within the Arduino board as its base platform.

1.5 Report Outline

The information presented in this report on the project is broken up into five chapters which are the introduction, the background study, the methodology followed by the results and the discussion, and the conclusion. In chapter 1, the description of the ideal and concept for robot agriculture and explanation step by step to analysis and develop the robot framework. This is followed the project background, problem statement, objective and scope of project.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

In Chapter 2 focused on the literature review or background study on robotic framework and embedded software using MATLAB software. In explanation for this comparison of previous research papers concerning the framework and application.

In chapter 3 is a focus in methodology content to achieve the objective project. This is followed by objective 1, the explanation on how analysis of Cili-Padi picking robot framework. In objective 2 show the step to develop embedded software and

hardware interface using MATLAB. And objective 3 is the final step how embedded software can run in real time in hardware.

Chapter 4, as where the result show achieves of the objective project. Result in objective 1 with analysis of Cili-Padi picking robot framework and can defined which actuators and sensor has been used and parameter has been used. Next, the objective 2, result show the develop embedded software interfaced with hardware and software communicate to produce GUI. Lastly objective 3 show the result functionality of embedded software in real time.

Finally, chapter 5. conclusion and future works towards the project. At conclusion will explained the objective project has been successful. This also explained the limitation that project face and can explained the solve at future work.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

CHAPTER 2

BACKGROUND STUDY

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

This chapter focuses on research on the development and analysis of embedded software for a robot. First, research about robotic frameworks and their applications Second, the basic embedded software for robots and robot functionality. Finally, a robot's software architecture and programming.

2.1 Robotic Framework and Its Application

The picking process has a significant impact on gripper designs for robotic harvesting; therefore, this section summaries the primary strategies currently used with the goal of identifying the gaps where soft robotics might make the largest contributions.

An integration framework is an essential tool for managing data flow across software programmed that interact with each other. An integration framework provides an abstraction for how information flows across applications and organizations. For example, research deep-learning-based object-detection framework is used in the Human Support Robot (HSR) platform to recognize the door handle. The framework proposed in this paper is built to essentially identify and classify types of door handles. The base framework used in this paper is YOLO V3 built using Dark Flow in Python. [4]

It's imperative that efforts like the Surgeon–Robot paper research by Prokhorenko and Klimov, based on specialist software or results of open-source solution integration, address the issue of translating surgeon orders into automated procedure execution processes. That translation is provided by a hardware and software system that interfaces between the surgeon and the robot to construct this type of application software, presents a generalized framework architecture Both OP:Sense and ARAKNES have been examined as the best existing frameworks for medical robotics. Two proprietary (and consequently non-modifiable) solutions are currently in development in this group [5].

An architecture for a socially assistive robot system for cardiac rehabilitation based on model-controller structure and finite-state machine and behaviour module

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

has been presented in prior work by Casas J, etc. al [15]. The platform's purpose is to increase the quality of the service offered, as well as the patient's involvement and performance, by providing social support and aid during the therapeutic process. Both the Robot Controller and the Robot Model provide the basis for this paper's work. Robots are able to interface with humans and other systems using this framework, which is made up of two parts: an Application Layer and a SARI (Self-Aware Robotics Interface). To access the robot's resources through TCP/IP, the interface employs the NAOqi Framework. The interface has two modules, Camera & Audio Manager, which manages camera, microphone, and speaker administration. The rest of the robot's resources, including as sensors, actuators, and the board built in, are managed by the Device Communication Manager (DCM) module [6].

An indoor robotic framework for human-robot interaction is presented by Ranieri C. and Nardari G. in their article. This study focuses on a robot framework, dubbed LARa, that was developed as part of the referred project. The LARa robot and the LARa library make up this framework. A robot platform for doing experiments indoors and competing in robot contests, it was created for this purpose. Pioneer P3DX robot, additional sensors, and an embedded laptop make up the LARa robot, which costs less than $1,000. The laptop is responsible for providing high-performance computation and visual feedback, including a robotic face. The LARa robot was designed to serve as a testing ground for research on human-robot interaction, with a particular emphasis on service robots. Computer vision was made easier because to the LARa library's beneficial behaviours for human-robot interaction applications These modules were made available as ROS topics so that they may be used in a variety of contexts [7].

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Titled Teaching a Robot with Unlabeled Instructions: The TICS Architecture, Najar A Sigaud O's study discusses how to teach robots without labelling their instructions. Humans may teach robots new tasks by giving them unlabeled instructions that they can follow in real time. Task Models, Contingency Models, Instruction Models, and Shaping Components make up the four primary components of the TICS architecture. Novel robotic task learning framework that combines the advantages of autonomous and interactive learning systems. To teach a robot new skills, this system uses unlabeled human instructions. Through the use of unlabeled instruction signals, a task-learning process is simplified. To speed up the learning process, these signals are analysed and utilised concurrently by the robot [8].

Cloud Robotics (CR) platform for data centre environmental monitoring is presented in this work by Ben Amor A, with the title Robotics based Solution for Data Center E-Monitoring. Because data centres use a lot of power, it's essential to keep an eye on the temperature in each one. In a data centre room, a mobile robot can autonomously navigate using an already-created map to accurately monitor critical measurements, localise itself, and perform an array of measurements that the user provides via a Graphical User Interface (GUI). This platform is based on the Robot Operating System (ROS) . Robot Operating System (ROS) middleware is the foundation of our system, which is built on cloud robotics and ROS middleware, which includes mapping and navigation algorithms [9].

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

In the publication "Synchronous Robotic Framework," written by Balaji N and Kilaru J Morales-Ponce O, the authors offer a synchronous robotic testbed known as SyROF that enables the rapid development of robotic swarms. Describe the design of the SyROF testbed, which comprises of many mobile robots of varying

types, such as omnidirectional robots, drones, and rovers, in this paper. Each robot possesses a CPU, a flow sensor, a sensor that functions similarly to a GPS, a sensor that detects gyroscopes and accelerometers, and a Bluetooth chip. Create and put into action a real-time publish-and-subscribe system as the foundational platform [10].

In conclusion, comparison of the framework and its application by all of the mentioned research is provided in Table 2.1. In the Robotic Framework, the implementation of robotic must depend on something. The specific use of the robot will determine the kinds of inferences that can be drawn from the robot's architecture.

Table 2.1 Research Robotic Framework

Robotic Framework	Platform	Application	Year
ARAKNES , Op:sense [5]	Multiplatform	Medical Robotics	2019
NAOqi [6]	Multiplatform	Social Robotic	2018
LARa [7]	Multiplatform	Robotic Competitions , Robotic Framework	2018
TiCS , Novel [8]	Multiplatform	Education Robotic, Teaching Robotic	2021
ROS [9]	Ubuntu	Cloud Robotic	2019
Testbed [10]	Ubuntu	Robotic Swarms	2020

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2.2 Basic Embedded Software for Robot

An embedded system is a computer system in which software was developed specifically for the hardware it runs on [11]. This system could be part of a bigger system or it could be self-contained. ROM allows software to be permanently stored within a memory module, eliminating the need for additional computer memory. Some examples of where embedded systems are put to use are in telecommunications, smart cards, missiles, computer networks, digital consumer electronics, and satellites.

Through task allocation, algorithm computation, and data analysis, the robot firmware implemented on PC or embedded microcontroller plays a significant role in creating the required actuation signals. transmission. The software programmed into the embedded microcontroller decides the control and action of each component of the robotic system so that the system can carry out the duties that have been previously stated [11].

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Applications of embedded systems include Robotics, digital cameras, multi-tasking toys, cooking and washing systems, biomedical systems, key-board controllers, mobile & smart phones, computing systems, electronic smart weight display system, and entertainment systems including videos, games, music systems, and video games, etc. The embedded system design process might begin with simulation, which is used to test the circuit because it is difficult to replace hardware if the circuit malfunctions. If the findings match the desired ones, sequential wafer procedures will be used to design the process permanently.

2.2.1 Embedded System Hardware & Software

A hardware platform is necessary for an embedded system in order for it to interface with a variety of real-time inputs and outputs or variables. The term "controller" can refer to a microcontroller or a microprocessor. Other components of the hardware include memory modules, I/O interfaces, display systems, communication modules, and so on. In paper by M. Dasygenis et al [12]. The primary hardware components of our autonomous vehicle implementation are a LIDAR sensor and an embedded system, with the project's overarching goal being collision avoidance. A Raspberry Pi and an RPLidar A2 Light Detection and Ranging (LIDAR) is a surveying technique that constantly emits pulsed light in all directions (360 degrees) and measures the reflected pulses at all angles using a sensor. Received pulses are analysed for their timing and wavelength in order to determine how far away an obstruction is. A thorough analysis of these readings is required for detection of obstacles and navigation. A Raspberry Pi 3 is utilised for the processing. Thanks to its 40 GPIO (General Purpose Input Output) pins, the Raspberry Pi 3 is a highly configurable, lightweight, capable, and inexpensive Single Board Computer (SBC) [13].

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Embedded System Software makes it possible to programme in any way that is wanted, allowing it to control a variety of different processes. Following its compilation into code and subsequent dumping into hardware controllers, it is written in a high-level format.

2.2.2 Real-Time Embedded Systems

Computer systems that operate in real time are known as real-time embedded systems, and they are responsible for tasks including monitoring, reacting, and

controlling external motion. Sensors, actuators, and input/output (I/O) interfaces allow the external environment to communicate with the computer system. Real-time embedded system refers to a computer system that is physically integrated with another computer system. Embedded real-time systems are employed in many different industries, including the medical, government, and military sectors [14].

Numerous autos, robotics, industries, etc., can benefit from Robotic Real Time Projects in Embedded Systems. Some current initiatives that make use of robotics technology are listed below.

2.2.3 Model Based Design in MATLAB

Testing for Model In the Loop (MIL), Software In the Loop (SIL), Processor In the Loop (PIL), and Hardware In the Loop (HIL) occurs during the verification phase of the Model-Based Design process. This phase occurs after the requirement of the component or system that is being developed has been recognized, and after the component or system has been modelled at the simulation level. There are a few verification procedures that take place before the model is delivered to the hardware for production. These verification steps are given below. Using the design of a controller for a DC motor as an example, we will now place the code that was created from the model of the Controller in a System-on-Chip that is supported [16].

A technique called Model-in-the-Loop (MIL) simulation, also known as Model-Based Testing, is used to abstract the behavior of a system or sub-system in such a way that this model can be used to test, simulate, and validate that model. This technique is sometimes referred to as "MIL simulation." Testing and simulation according to MIL standards is the starting point for software verification and validation. It involves testing individual model modules or combined model modules

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

in a development environment such as MATLAB Simulink from MathWorks or ASCET from ETAS. For instance, once a plant model has been established, MIL can be used to validate whether or not the controller module is able to control the plant in the correct manner. It ensures that the controller logic is responsible for producing the necessary functionality [17].

Software in the Loop, also known as SIL, is a process that creates code from the controller model and then uses that code instead of the controller block. Verifying the functionality of embedded software, algorithms, control loops, and other system parts on a personal computer or another platform apart from the actual control hardware is what SIL testing entails. It involves modelling the physical plant as a software simulation and the software itself as C code. Imagine that there are some strange outcomes from the SIL. In this scenario, it may be essential to return to the MIL stage, make adjustments as indicated by the SIL results, reaffirm that the alterations continue to give acceptable results from the MIL testing, and then repeat the process of SIL verification. When dealing with more complicated systems, it is often required to go through this circular procedure multiple times. Following the successful completion of the MIL and SIL verification processes, the following step is the PIL verification [18].

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Putting the Processor in the Loop PIL testing might alternatively be called FIL testing, which stands for "FPGA-in-the-loop testing," depending on the architecture of the system. At this stage, the embedded software, algorithms, control loops, and other components of the system are executed as a closed-loop simulation directly on the physical hardware of the processor (or FPGA), as opposed to being run on a personal computer or some other platform. This guarantees that the controller

software will work smoothly on the hardware, and that the outputs will be accurate and as anticipated. If there are problems at this level, it may be possible to determine that the required tasks cannot be performed by the processor, or it may be possible to determine that the programme has to be modified. In the event that software changes are required, it is possible that it will be essential to return to the SIL or even the MIL testing phase before returning to the PIL phase. In the same time that the control software is being developed and verified through the use of MIL, SIL, and PIL testing, the various hardware systems and subsystems are being modelled and validated through the utilisation of MBSE tools and methodologies. HIL testing is often the final validation and verification stage that takes place just before starting the system integration process [19].

Before attaching the embedded processor to the actual hardware, HIL testing is performed by running a simulated plant model on a real-time system such as Typhoon, OPAL-RT, dSPACE, Speedgoat, and NI. This is done before the embedded processor is connected to the hardware. The majority of these tools are compatible with simulation models that were developed on platforms such as MATLAB Simulink or ASCET. Deterministic simulations and real physical connections to the embedded processor are both components of HIL. These connections can take the form of analogue inputs and outputs, communications interfaces, and other similar elements. It does this by recording the interaction that takes place in real time between the control software and the actual hardware environment. This allows it to discover glitches and other issues that could otherwise go undetected by simulations. It is possible, for instance, to add delays and attenuations to an analogue channel, both of which might generate instabilities in the control loop. HIL testing is necessary in the aerospace and automotive industries,

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

particularly for operations that are considered to be safety-critical. In the event that an issue arises during the process of integrating the different subsystems, it is possible that HIL testing will need to be repeated. In the most severe circumstances, it may be required to revert back to an even earlier stage in the process of MIL, SIL, or PIL[20].

Table 2.2 show the different of model design in MATLAB. For each Model has prototype limitation and according to the suitability of the prototype to be built.

Table 2.2 Model Based Design in MATLAB

Model Based Design	**Description**	**Example**	**Prototype Limitation**
Model In the Loop (MIL)	MIL testing uses a software environment like MATLAB to perform the simulation model and interface the physical system or component.	Utilize an interface, such as an Arduino board, to link the physical system or component to the software environment. Utilize the simulation model to transmit inputs to and receive outputs from the physical system.	MIL testing necessitates a sophisticated relationship between the simulation model and the physical system or component being evaluated.
Software In the Loop (SIL)	SIL simulation model is executed in	Using an interface, such as a function	SIL testing needs a sophisticated link

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

	a software environment, such as MATLAB, and the software or algorithms being tested are connected to the simulation model via an interface.	call, connect the under test software or algorithm to the simulation model. Send inputs to the software or algorithms and receive their outputs using the simulation model.	between the simulation model and the software or algorithms being evaluated.
Processor In the Loop (PIL)	PIL testing involves running the simulation model in a software environment, such as MATLAB, and connecting the control system or process to the software environment via an interface.	Using an interface, such as an Arduino board, link the control system or process to the software environment. Utilize the simulation model to transmit inputs to and receive outputs from the control system.	PIL testing needs a connection between the simulation model and the being tested control system or process, which can be difficult to establish and maintain.
Hardware In the Loop (HIL)	HIL testing, the simulation model is executed in a	Using an interface, such as an ADC/DAC (analog-	HIL testing necessitates a sophisticated

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

	software environment, such as MATLAB, and the hardware or physical system being tested is interfaced with the simulation model.	to-digital/digital-to-analog converter), connect the hardware or physical system under test to the simulation model. Send inputs to the hardware or physical system and obtain outputs from it using the simulation model.	relationship between the simulation model and the hardware or physical system being tested.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2.3 Basic Structure of Robot

Fruits can be detached in one of two ways mechanically, wherein pieces of fruit are removed from the tree branch using a machine or mechanical mechanism, or (ii) manually, wherein pieces of fruit are extracted from the tree branch using the human hand. The methods of mechanical fruit harvesting are categorized as follows in : Both I those who utilise air blasting, canopy shaking, limb shaking, or trunk shaking to take the fruits from the entire plant and (ii) those that use automatic robotic picking robots that require minimal or no human intervention to harvest the ripe fruits [21].

A robot is an autonomous mechanical system that can carry out tasks and interact with its surroundings using preprogrammed instructions. The study and practice of robotics includes all aspects of the robot life cycle from conception to operation. A robot consists of a frame or body, a control system, manipulators, and a drivetrain. No restrictions on size or shape are placed on the body or frame. To put it simply, the robot's body or frame is what holds everything together. In popular culture, robots typically resemble humans in size and shape, yet most actual robots look nothing like humans. Most robots prioritise functionality above aesthetics.

Many robots rely on interacting with their surroundings and the outside world to carry out their tasks. Robots can be used in situations where humans can't because they need to move or rearrange objects without touching them. As opposed to the Body/frame and the Control System, which are required for the robot to function, the Manipulator is not required for the robot to function. In particular, Unit 6 of this programme emphasises the use of manipulatives.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

While some robots can accomplish their goals without ever leaving a single spot, mobility is typically essential for robots. They need a drivetrain in order to complete this task. A powered means of transport is what drivetrains are all about. Legs are used by humanoid robots, while wheels are used by the vast majority of other robots [13].

2.4 Functionality of Robot

It is not sufficient to classify robots based on their generation (first, second, or third) or the distinction between autonomous and non-autonomous robots. Robotics is a very complex system.

To comprehend the typology and models of robots currently available, it is necessary to comprehend the robots' underlying structure and primary functions. Essentially, they consist of four "functional units" and must be viewed as complex systems with various "functional organs" (mechanical organs, sensory organs, control organs, governing, and calculation organs).

Before viewing each of these organs, it is important to recall the Robotic Institute of America's definition of robotics (RIA). "A robot is a programmable, multifunctional manipulator capable of performing a variety of tasks. A robot also gathers data from its surroundings and moves intelligently in response."

Fundamentally, robots perform three functions — "sense," "think," and "act" — which form the basis of their autonomy. They "sense" environmental stimuli, "think" in terms of predetermined planning algorithms, and "act" based on these algorithms, which define their reactions and overall behavior.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

This three-function process drives actions such as increasing pneumatic power to orient a picking limb in order to pick and place a component on a circuit board or lowering a tray onto a patient's side table. As depicted in Figure 2.1, these three functions define the primary technologies used in robotics [22].

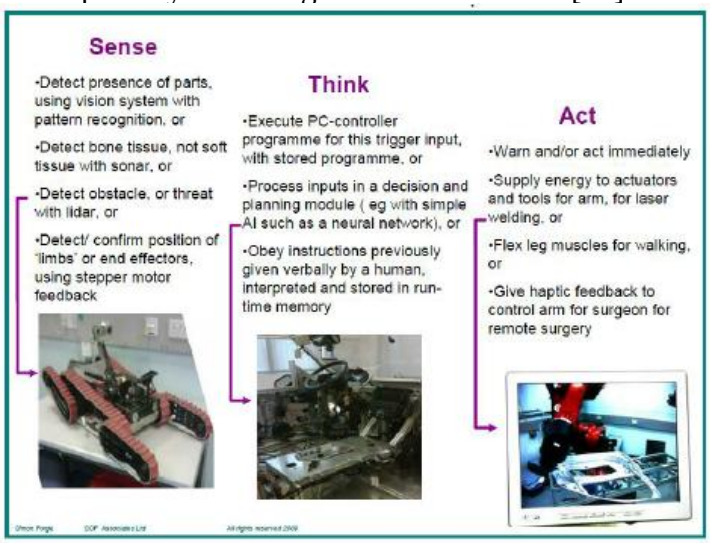


Figure 2.1 Fundamental Robotic Operations and Actions

2.4.1 Mechanical Structure Function

The functional organs of a robot will be explained independently in the follow sub-section. The controller, robot body, robotic arm, sensors, and end-effector are the primary parts of an industrial robot. Each of these parts plays a crucial part in an industrial robot's functioning as a whole. Mechanical components may not be as visible as the rest of the robot, but they are essential to its operation. Industrial robots are able to move and do their intended tasks because of the mechanical parts that make them work.

For example, in paper “Lower Limb Rehabilitation Exoskeleton Robots” Rehabilitating the Lower Limbs For exoskeleton robots to be able to transmit force and energy through the wearable link, they need a mechanical construction similar to human lower limbs. These results are attainable through the development of a suitable robot mechanism and actuation [23].

2.4.2 Sensory Structure Function

The robotic systems have the ability to "perceive" their surroundings and react accordingly. Of course, it's not about how the robot feels, but rather how the robot is equipped with sensors that allow it to gather information about its internal mechanical status. variables (such as coordinates and velocity) and external factors (such as weather and terrain)

2.4.3 Control Structure (Robot Control Organs)

The systems that ensure the robot can carry out the tasks for which it was designed are located in the control organs, which serve as bridges between perception and action. Actuators (electric motors, hydraulic or pneumatic systems, etc.) and control algorithms for driving the actuators provide the control framework.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2.4.4 Governance Structure (Memorization and Calculation Organs) Function

In this context, "systems" refers to the means through which robotic machines may be programmed, calculated, and monitored. Hardware (microprocessors, memory, etc.) and software (controllers, algorithms, etc.) typically make up the governance and calculation structure (application programs, calculation algorithms coded in programming languages, standard or dedicated). Table 2.3 show the short explanation for each function of robot and include the example.

Table 2.3 Functionality of Robot

Function	Description	Example
Mechanical Structure	It gives the robot a physical form as well as the capacity to move and interact with its surroundings.	Bridges sustain vehicles and people and transfer their weight to the ground or other supporting structure. A bridge must be strong and stiff enough to withstand the loads it will face and made of materials that can withstand these stresses.
Sensory Structure	In a robot, a sensory structure is a device or component that is used to detect and respond to stimuli from the environment, and	Robots utilise cameras to get visual data. Robot cameras detect light and convert visual data into electronic impulses. These signals are

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

	its function is to collect information about the environment and transmit it to the robot's processing unit, so that the robot can perceive and respond to its surroundings.	relayed to the robot's processing unit, which interprets and displays the environment.
Control Structure	A robot's control structure is a system or component that regulates the robot's behaviour in response to input from sensory structures and other sources. A control structure processes the input and generates commands or signals that direct the robot's motions, activities, and functions.	The control system is responsible for processing data from sensors and other sources and creating commands or signals that direct the movements, activities, and functions of the robot.
Governance Structure	A system or collection of processes used inside an organisation or group for making and implementing decisions. A governance structure for a robot would be a system or set of processes used to make and implement decisions	A firm that develops and manufactures industrial robots may create a governance structure to ensure that the robots are safe, ethical, and in line with the company's beliefs and aims. Processes like:

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

	regarding the development, deployment, and use of the robot.	Reviewing and approving robots in production, considering their potential effects on people and the workplace. Establishing robot maintenance and operation rules, including worker safety and training.

2.5 Software Architecture and Programming of a Robot

The robot system's operations are managed by the governance unit using a combination of sensor data and an internal model of the automaton's mechanical structure. Algorithms that decide the actuator signals should be placed at the highest level of the control architecture's hierarchy. Within such a hierarchical framework, the output of one level's computation is transmitted to the level below it, where it is, however, affected by the previous level's output in a backwards fashion.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Looking at robotic system code can be done in one of three ways:

1. Teaching by showing: the robot is led along a path, and it learns the positions achieved thanks to the sensors; afterwards, it merely copies that sequence of locations.
2. A high-level programming language with intricate data structures, variables, and routines is available for use with robots.

3. Object-oriented: the same as the first, but the language is object-oriented.

In summary a robot is an intricate mechanical device with moving parts. But may move and perform its functions suitably in order to fulfil the roles for which it has been designed. It needs to be modelled mathematically, with the interrelationships between its parts accounted for (mechanical organs, sensory organs, control organs).

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

CHAPTER 3

METHODOLOGY

UNIVERSITI TEKNIKAL MALAYSIA MELAKA
UTeM
أونيۏرسيتي تيكنيكل ملسيا ملاك
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

This chapter explains how the project was finished. First, analyses the framework of the robot. Next, describe how Embedded Software for Requirements is developed. Cili-Padi Picking Robot that adheres to embedded sub-part development software. Lastly, explain the demonstration hardware and software's functionality.

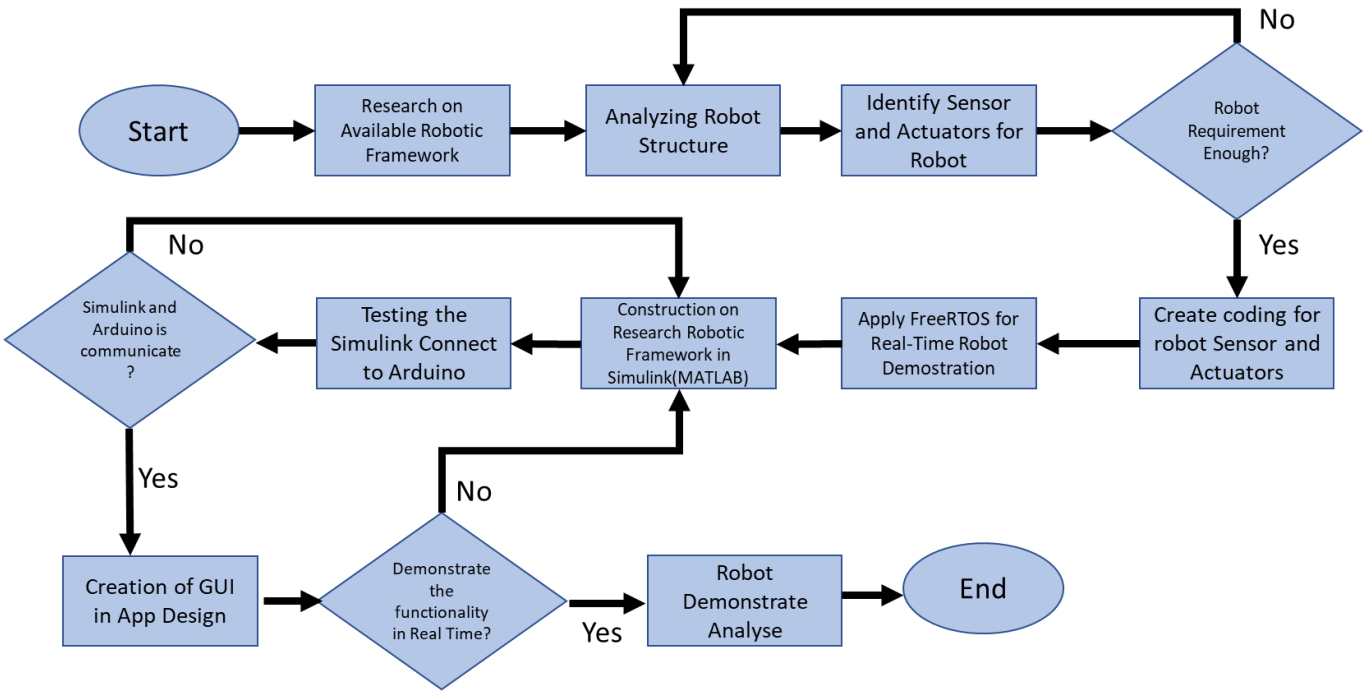


Figure 3.1 Flowchart of Project

Figure 3.1 show the flowchart of the project execution. The first step in the project execution is Research on available robotic framework. Then analyzing robot structure to identify sensor and actuators for robot. After that identify the robot requirement enough or not. If not enough requirement robot, analyzing robot structure again and if requirement enough the objective one in this project is accomplished. Next, to achieve objective two create coding for robot sensor and actuators and apply the FreeRTOS coding for real-time robot demonstration. After that research the robotic framework in Simulink to connect Simulink and Arduino. If Arduino and Simulink communicate the objective to is achieve. If not communicate research again about Simulink and Arduino to communicate. Lastly, for achieve objective three , first step creates the Graphical User Interface(GUI) in App Design at MATLAB. After the demonstrate the functionality in Real Time is successfully or not. If not function , research again Robotic framework in Simulink. If demonstrate functionality in real time the objective three is accomplished.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

3.1 Analyzing of Robot Framework for Cili-Padi Picking Robot

Figure 3.2 Architecture of Cill-Padi Picking Robot

In order for the input and output systems to communicate well with one another one, must first comprehend the purpose of the robot and be familiar with its internal components. This requirement analyzes each individual component of the robot's interior, one at a time, in order to build a framework node for the robot.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

In this project, framework of a Cili-Padi Picking Robot has chosen as one of the examples of robot that can be applied in agriculture robot. The Cili-Padi Picking Robot is a machine that can be used to harvest mature chilli peppers. The agricultural sector is the one that will be most benefit from this technology, in some cases will ne more benifit there is insufficient labour available in the agriculture sector. Therefore, research has been done in order to find a solution to the challenge of developing robots that can assist farmer in the agricultural industry. The robot architecture for harvesting ripe rice chillies is depicted in the picture labelled Figure 3.2. In the cili-padi picking robot system, the robot framework was analyzed based on this cili-padi picking robot figure. Two different kinds of cameras have been utilised in this

automated system in order to monitor the maturation of the chilies. Which are tracking cameras and lidar cameras. The lidar camera will identify the mature chilies that are ready to be picked up at this point. The gripper delta arm robot will then automatically go to the position where the ripe chilli is located after being guided there by the tracking camera after it has detected its location. Gripper delta arm utilises three motors in order to move the delta arm in an easy and convenient manner to the box. Regarding the mobility, will be using six Flex sensors. The flex sensor function allows the robot to move in response to the tension and force that is applied by our hand grip. In addition, there is both a left-hand and a right-hand condition for the command.

In order for the input and output system to communicate well with one another of each individual component of the robot's interior, one at a time, in order to build a framework node for the robot. The sensor and actuator after analysis of the cili-padi picking robot its show in framework consist of several sensor and actuators such as IMU sensor, flex sensor, stepper motor and servo motor. The architecture and component used analysis of in the robot is show in Figure 3.3.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

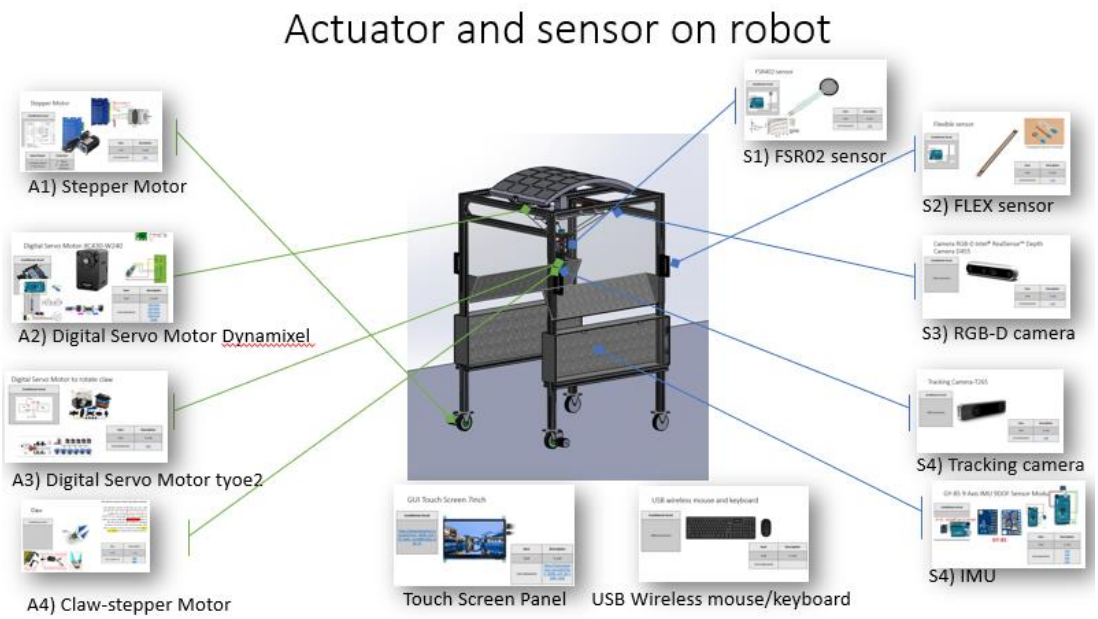


Figure 3.3 Cili-Padi Picking Robot Architecture and Components.

3.2 Development an Embedded Software for Requirement Cili-Padi Picking Robot

To perform as intended, a robotic system requires integration of specialised hardware and software. In this section, the system's requirements, which include a description of the desired behaviour and limitations of the cili-padi picking robot will be detailed respectively.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

3.2.1 Navigation System Requirement

In this cili-padi picking robot, two differential drive kinematic. Stepper motor required as the navigation system. The two-stepper motor are required on the left and right of the motor for navigation system requirement is show in Figure 3.4 waymove the robot which are by :

1. Using gui – Example Joystick, or button navigation
2. Using handler have 4 sensors around the handler – like using simple fuzzy logic for predict direction

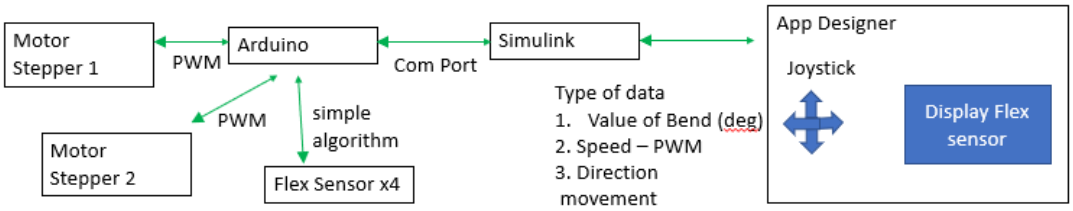


Figure 3.4 Navigation System Requirement

3.2.2 Delta Robot Arm Requirement

The cili-padi picking robot in this project consists of a delta robot arm. The robot delta arms are robotic arms that operate in a forward or reverse kinematic motion using three motor servos. As they can move rapidly and precisely, they are frequently employed for pick-and-place applications. An IMU (Inertial Measurement Unit) sensor controls the arm's motion by measuring the arm's angle along the x, y, and z axes. The IMU sensor is utilized to determine the arm's position and change its motion accordingly. The delta robot requirement as a shown Figure 3.5.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

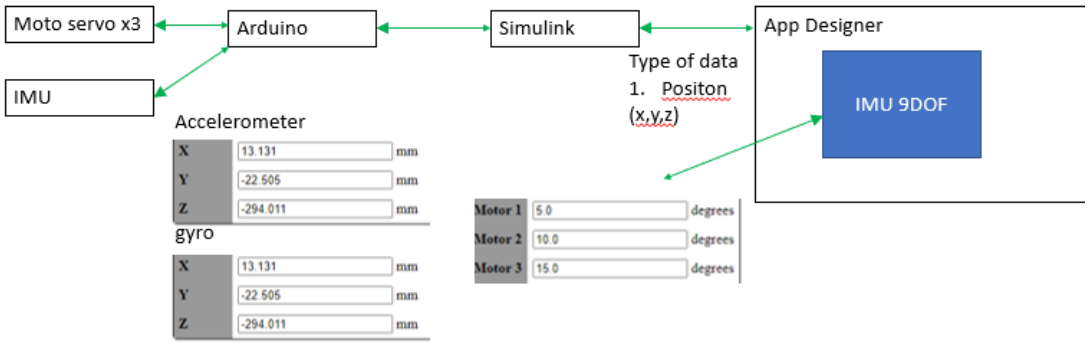


Figure 3.5 Delta Robot Arm Requirement

3.2.3 Robot Gripper Requirement

To control a robot gripper with one servo that rotates the entire gripper 360 degrees with an accuracy of 1 degree, has need a servo that is capable of rotating to a specific angle with high precision.

For the second servo that is used to open and close the claw gripper, will again need a servo that has high precision and can move to a specific angle with accuracy. Also need to consider the torque of the servo, as it will need to be able to generate enough force to open and close the gripper. The control system for this servo will need to be able to send precise control signals to the servo in order to achieve the desired angle of rotation. Robot Gripper requirement shown Figure 3.6. In app design had knob rotate control and can edit manually data using edit text. For open closed claw using the toggle button.

For gripper have 2 actuators :

1.Analog servo 1 is to rotate the entire gripper 360 deg Accuracy 1 deg.

2.Analog servo 2 is to open and closed the gripper.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

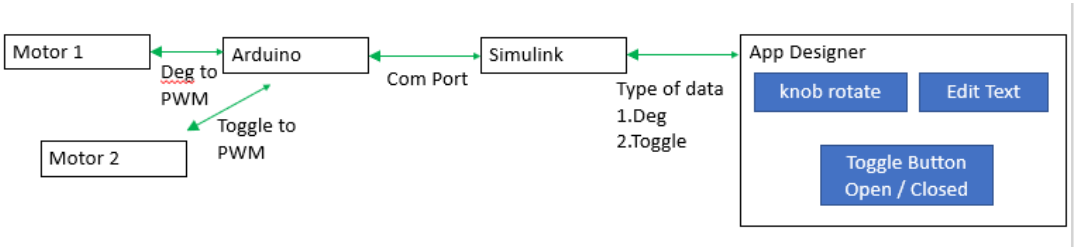


Figure 3.6 Robot Gripper Requirement

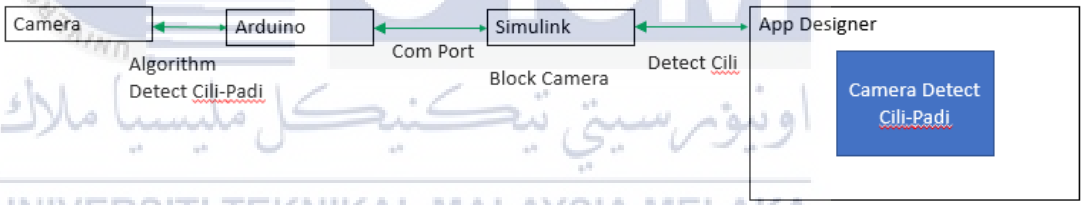
3.2.4 Robot Vision Requirement

To detect ripe chilies using a camera and an algorithm it shows in Figure 3.7, a camera able to capture high-resolution photos or video of the chilies is required.

Additionally, the camera should have a broad field of vision in order to capture a bigger portion of the chiles. The algorithm used to detect ripe chiles will depend on the particular features being sought. The size, shape, colour, or texture of chilies could be used to determine whether they are ripe.

Once the camera and algorithm are in place, they must be integrated into a graphical user interface (GUI) that displays the camera feed and the algorithm's results. The user interface should allow the user to alter camera settings and manipulate the algorithm as necessary. Possible additions to the GUI include the possibility to save photos or movies, as well as the display of statistical data regarding the detected chilies.

1. Camera – GUI display image



UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Figure 3.7 Robot Vision Requirement

3.3 Real-time robot operating system using FreeRTOS

FreeRTOS is a real-time, open-source operating system for microcontrollers that simplifies the development of multitasking applications [24]. FreeRTOS supports numerous threads or tasks, mutexes, semaphores, and software timers. For low-power applications, a mode without ticks is offered. Priorities on threads are supported. Multiple processes or threads can apparently run concurrently on the majority of operating systems. The term for this is multitasking. Each CPU core can

only run one application at any given time. A component of the operating system known as the scheduler is responsible for determining which application will execute when and creates the illusion of simultaneous execution by frequently switching between programmes. The scheduler of a Real Time Operating System (RTOS) is intended to offer a predictable (often termed deterministic) execution pattern. This is especially relevant for embedded systems, such as the Arduino devices, because embedded systems frequently have real-time requirements. Traditional real time schedulers, such as FreeRTOS's scheduler, accomplish determinism by permitting the user to specify a priority to each thread of execution. The scheduler use the priority to determine which thread of execution to execute next. Task is the term for an execution thread in FreeRTOS [24].

In this project that implementation that have been apply are FreeRTOS to get real-time robot operating. According to all data running the most important data selected will be put at higher priority. In Cili-Padi Picking Robot project, it will priorities the Inertial Measurement Unit (IMU) sensor. This is due to the fact that the coordinate Cili-Padi Picking Robot and position arm delta robot must always transmit data for the navigation system, Robot Gripper system, and Robot vision system to operate effectively.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

This is 5 step to create FreeRTOS:

1. Include the Arduino library FreeRTOS header file follow in Figure 3.8.

```
#include <Arduino_FreeRTOS.h>
```

Figure 3.8 Library FreeRTOS

2. Provide the function prototype of all functions are developing for execution, which is represented as Figure 3.9.

```
void Task1( void *pvParameters );
……..
…….
void Task2( void *pvParameters );
```

Figure 3.9 FreeRTOS Task

3. Create tasks and launch the task scheduler in the void setup() function. In the setup method, the xTaskCreate() API is invoked with particular parameters/arguments to create a task. The explanation detail in Table 3.1.

Table 3.1 Description Each Function in Coding FreeRTOS

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

`xTaskCreate(`
`  Task1 // Task function`
`  , "task1"  // Task name`
`  , 128  // Stack size`
`  , NULL`
`  , 1  // Priority`
`  , NULL ); // Task handler`
`xTaskCreate(`
`  Task2 // Task function`
`  , "task2"  // Task name`
`  , 128  // Stack size`
`  , NULL // Parameter`
`  , 1  // Priority`
`  , NULL ); // Task handler`
	Task Function
	The only thing it is is a pointer to the function that does the work (in effect, just the name of the function).
	Task Name
	a task name that is clear and concise. FreeRTOS does not make advantage of this. It is only present for debugging needs.
	Stack Size
	When a task is created, the kernel allots a stack to the task that is unique to that task. The value indicates the amount of words that can be stored on the stack, not the number of bytes. If the stack is 32 bits wide and usStackDepth is 100, for instance, 400 bytes of stack space will be allocated in RAM (100 * 4 bytes). Arduino Uno has only 2K bytes of RAM, so utilise it with caution.
	Parameter
	Input Parameters for a Task (can be NULL).
	Priority
	Task for priority. 0 is lowest priority
	Task Handler
	It can be used to pass out a handle to the task being created. This handle can then be used to reference the task in API calls that, for example, change the task priority or delete the task (can be NULL).

4. Start the scheduler in a void setup using the " vTaskStartScheduler(); " After creating the task.
5. The loop() method will be left empty so that no task will be run manually and indefinitely. Due to the fact that task execution is now handled by Scheduler.
6. The next step show the Figure 3.10 is to construct task functions and write the desired logic within them. The function name must match the first argument of the xTaskCreate() API call.

```
void task1(void *pvParameters)
{
while(1) {
..
..//your logic
}
}
```

Figure 3.10 void Task Coding and Loop

7. The Figure 3.11 show majority of codes require the delay function to halt a running job, but it is not recommended to use the Delay() method in RTOS because it stops the CPU and, in turn, RTOS. Therefore, FreeRTOS provides a kernel API to pause a job for a predetermined amount of time.

```
vTaskDelay( const TickType_t xTicksToDelay );
```

Figure 3.11 Codes Require Delay

This API can be used for delay-related functions. In Figure 3.12 show API delays a job by a specified number of ticks. The actual amount of time the task is blocked is dependent on the tick rate. The portTICK PERIOD MS constant can be used to derive real-time from the tick rate. This means that if want a 200ms delay, simply type this line:

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

```
vTaskDelay( 200 / portTICK_PERIOD_MS );
```

Figure 3.12 API Delays

This example to use these FreeRTOS APIs to implement three tasks:

1. xTaskCreate();
2. vTaskStartScheduler();
3. vTaskDelay();

3.4 Arduino coding for each part sensor for Cili-Padi Picking Robot

This section describes the Chili-Padi Picking Robot's operational sensors. Each sensor has a unique function. Some sensors collect data that is unnecessary for this project. Consequently, this research investigates which data should be collected and which should not be collected. As an illustration, the IMU sensor - GY85 has nine Data, including three accelerometer data, compass data, and gyro data. Therefore, each data has x, y, and z data of varying forms. for Chili-Padi Picking Robot the suitable data used is Accelerometer data and Gyro data. Therefore, it is essential to identify the sensors utilised to choose relevant data for this robot.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

3.4.1 Part of Navigation system Arduino coding

Using an Arduino, a Flex Sensor, and a Motor, this is the component that makes up our part of the Cili-Padi Picking Robot. This component is needed while manually navigating the Cili-Padi Picking Robot in order to move it. Using of the FreeRTOS library that implement a real-time operating system kernel for embedded devices in the Cili-Padi Picking Robot. Figure 3.13 show the connection with arduno and Figure 3.14 is Coding. Output will show condition flex sensor in degree and value voltage is produce at Figure 3.15.

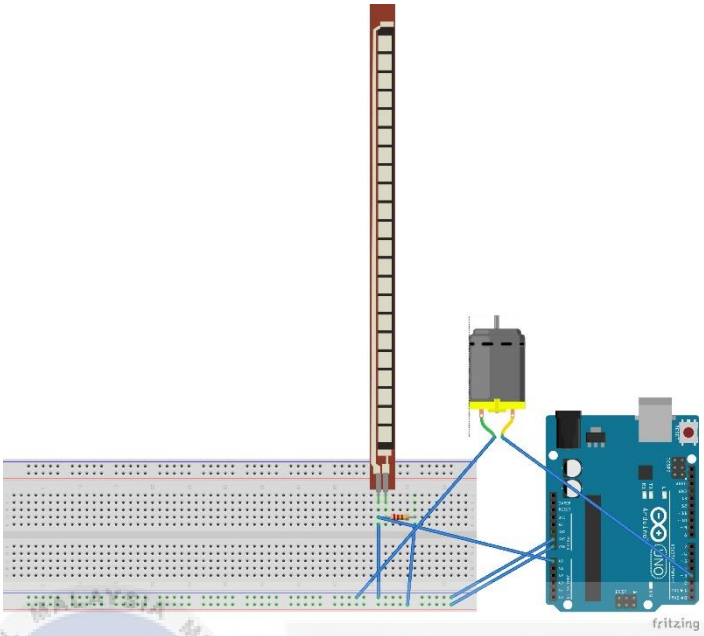


Figure 3.13 Circuit Flex Sensor and Motor

```
const int flexPin = A0;      // Pin connected to voltage divider output
// Change these constants according to your project's design
const float VCC = 5;       // voltage at Ardunio 5V line
const float R_DIV = 47000.0;  // resistor used to create a voltage divider
const float flatResistance = 25000.0; // resistance when flat
const float bendResistance = 100000.0;  // resistance at 90 deg
void setup() {
  Serial.begin(9600);
  pinMode(flexPin, INPUT);
}

void loop() {
  // Read the ADC, and calculate voltage and resistance from it
  int ADCflex = analogRead(flexPin);
  float Vflex = ADCflex * VCC / 1023.0;
  float Rflex = R_DIV * (VCC / Vflex - 1.0);
 // Serial.println("Resistance: " + String(Rflex) + " ohms");
   Serial.println("Voltage: " + String(Vflex) + " Volt");
  float angle = map(Rflex, flatResistance, bendResistance, 0, 90.0);
  Serial.println("Flex sensor: " + String(angle) + " degrees");
  Serial.println();
  delay(500);
}
```

Figure 3.14 Coding Flex Sensor to Apply at Motor

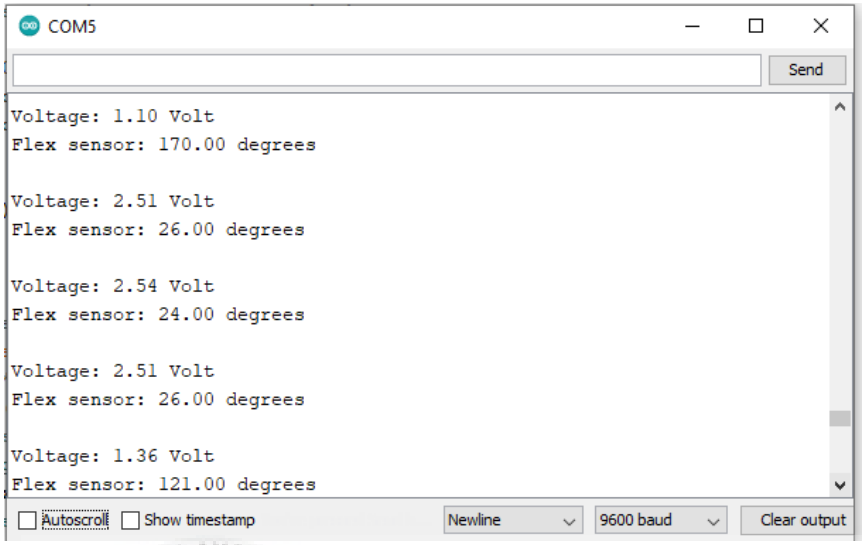


Figure 3.15 Output from Coding Flex

3.4.2 Part of Delta Robot Arm and gripper system Arduino coding

A member of the sensor family known as an inertial measurement unit (IMU) is a piece of electrical equipment. In order to measure the acceleration, angular velocity, and orientation of the sensor, it utilises a combination of a gyroscope, an accelerometer, and a magnetometer. An accelerometer and a gyroscope are the two components that make up a type I inertial measurement unit. A magnetometer is included as part of a type II IMU as well.

All three types of sensors—accelerometers, gyroscopes, and magnetometers—collect information along a single axis (X: pitch, Y: roll, Z: yaw). For a type II inertial measurement unit (IMU), need to integrate three components for each axis (an accelerometer, gyro, and magnetometer) in order to acquire data for all three axes. The conventional inertial measurement unit (IMU) sensor comprises nine degrees of freedom (DoF), which are comprised of three accelerometers, three

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

gyroscopes, and three magnetometers. There are only two types of sensors that can be used for the robotic arm on the Chili Padi Picking robot: there are three accelerometers and three gyroscopes.

3.5 Development Functionality Robot using MATLAB software

This section describes how Arduino and MATLAB interact with external hardware via serial communication. Numerous external hardware devices are intended to connect to a PC via its Serial Port. This project will connect Arduino to MATLAB through serial communication using Simulink block.

To create a Graphical User Interface (GUI) utilising the SIMULINK platform as a function to adjust Arduino input and receive output. Using App Design, to create a graphical user interface with input and output buttons. App Design has alter and show the value from the Simulink block that was developed as a communication medium between MATLAB and Arduino.

اونيۏرسيتي تيکنيکل ملسيا ملاک
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Before connect the embedded processor to the actual hardware, and can run the simulated plant model on a real-time system such as Speedgoat. This is what is known as a Hardware-in-the-Loop, or HIL, Simulation. The real-time system is capable of carrying out deterministic simulations and has real-world, physical connections to the embedded processor. These connections can take the form of analogue inputs and outputs as well as communication interfaces such as CAN and UDP. This will assist in determining problems that are associated with the communication channels and the I/O interface. One such problem is attenuation and delay, both of which are brought about by an analogue channel and can cause the controller to become unstable. These behaviours are not amenable to being modelled or simulated. HIL testing is frequently carried out for safety-critical applications, and

the validation requirements for the automobile industry and the aerospace industry necessitate it.

3.5.1 Serial Communication MATLAB (Simulink) connect in Arduino

At this point in the process, the hardware and software will both communicate with Simulink over the serial port. To make use of the SEND block in order to transfer data from Simulink to Arduino, and have to make use of the RECEIVE block in order to transfer data from Arduino to Simulink. In Figure 3.17 this is block from Simulink to Arduino with single data send and receive for communicate using serial communication.

This section will begin transmitting and receiving single data using serial communication between Arduino and Simulink as part of an effort to send and receive data from Arduino. The coding and simulink block has been configured as depicted in Figure 3.18 and Figure 3.19. Set the Arduino's coding to "writeToMatlab(myVal*2);" to display the data received and transmitted. myVal is derived from simulink block data and arduino contains data for times 2.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Referring to Figure 3.13, the data 4 is located to the input block, and when the block is transferred, the output data is 8. In this case, the receiving block has received data from the Arduino that is multiplied by two and added to the input data, which is four. Consequently, 4x2 equals 8 is evidence that arduino and simulink successfully send and receive data.

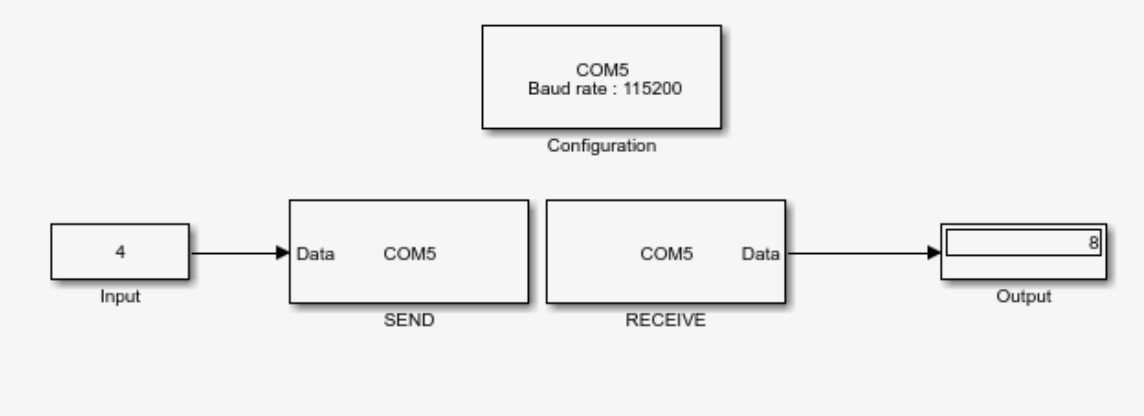


Figure 3.17 Block in Simulink for Single Data For Send Data And Receive Data

```
union Btof
{
  byte b[16];
  float fval;

} u;

const int BUFFER_SIZE = 16;
byte buf[BUFFER_SIZE];

float myVal;

void setup()
{
  Serial.begin(115200, SERIAL_8N1);
}

void loop()
{
  if(Serial.available()>0)
  {
    myVal = readFromMatlab();
    delay(0.1);
      writeToMatlab(myVal*2);

  }
}
```

```
float readFromMatlab()
{
   int rlen = Serial.readBytesUntil('\n', buf, BUFFER_SIZE);
  for (int i=0; i<BUFFER_SIZE; i++)
  {
    u.b[i] = buf[i];

  }
  float output = u.fval;
  return output;
}


void writeToMatlab (float number)
{
  byte *b = (byte* ) &number;

  Serial.write(b,4);
  Serial.write(13); // "\r" CR
  Serial.write(10); // "\r" LF
}
```

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Figure 3.18 Coding Single Data

It is possible to send and receive by sending and receiving many data sets. This is because this project exclusively utilises serial data transfer through cable. Therefore, it is necessary to utilise block multiplexer to transmit data from simulink to Arduino and demux to receive data from Arduino to Simulink.

As shown in Figure 3.19, the code to receive and transmit numerous data from Simulink to Arduino is provided. The Simulink in Figure 3.20 show which sends and receives a single piece of data. Multiple data must contain both mux and demux blocks. This segment in Figure 3.20 has three inputs and three outputs. myValue1, myValue2 and myValue3 are data from simulink. send1, send2, and send3 are used to transmit data from Arduino to Simulink. In order to see the difference between output and input in simulink, the Arduino output code is input1+input2. While output1 is for input2 - input 1, output2 is for input2 times input1. Consequently, this demonstrates that numerous receive and send operations can be performed utilising mux and demux. To be successful with the mux and demux block functions, however must persist in coding using the check sum method.

```
// Create a union to easily convert float to byte
typedef union{
  float number;
  uint8_t bytes[4];
} FLOATUNION_t;

// Create the variables you want to receive
FLOATUNION_t myValue1;
FLOATUNION_t myValue2;
FLOATUNION_t myValue3;


// Create the variables to send
FLOATUNION_t send1;
FLOATUNION_t send2;
FLOATUNION_t send3;

void setup() {
  // initialize serial, use the same boudrate in the Simulink Co
  Serial.begin(115200);
}
void loop(){
  // Get the floats from serial
  myValue1.number = getFloat(); // Give your float a value
  myValue2.number = getFloat(); // Give your float a value
  myValue3.number = getFloat(); // Give your float a value
  // Do whatever you want here

  // Send some variables back
  send1.number = myValue1.number+myValue2.number;
  send2.number = myValue2.number-myValue1.number;
  send3.number = myValue3.number*myValue1.number;

  // Print header: Important to avoid sync errors!
  Serial.write('A');
```

```
  // Print float data
  for (int i=0; i<4; i++){
    Serial.write(send1.bytes[i]);
  }
  for (int i=0; i<4; i++){
    Serial.write(send2.bytes[i]);
  }
  for (int i=0; i<4; i++){
    Serial.write(send3.bytes[i]);
  }
  // Print terminator
  Serial.print('\n');

  // Use the same delay in the Serial Receive block
  delay(500);
}
float getFloat(){
    int cont = 0;
    FLOATUNION_t f;
    while (cont < 4 ){
        f.bytes[cont] = Serial.read() ;
        cont = cont +1;
    }
    return f.number;
}
```

Figure 3.19 Coding for Multiple Data

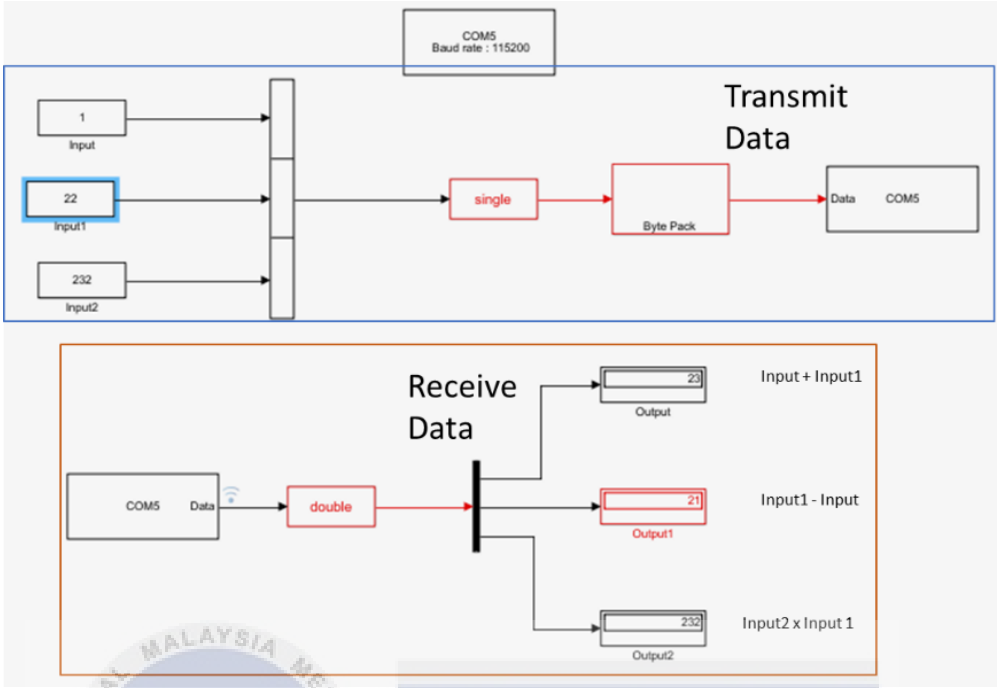


Figure 3.20 Simulink Multiple Data

Data is transmitted and received in sequences of bytes (1 byte = 8 bits or 0 - 255) through a single pin of this wire, hence the name "serial message.". Similar to how a period at the conclusion of a phrase in English denotes the end of the thought, a "terminator" at the end of a string of bytes signifies the end of a message. As long as all parties agree, a carriage return (r) is usually used as the message termination. The concept of a buffer is fundamental to comprehending serial data transmission. Let's pretend a sensor is constantly feeding data back to app, probably more often than app can process it. The computer's buffer is where the information is kept until it is read. One way to understand a buffer is as a list:

- As new data values are received, they are appended to the end of the list (most recent data)

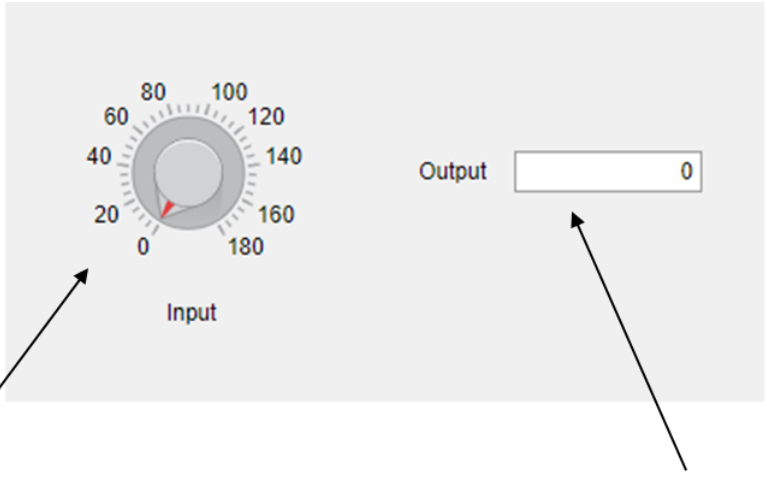
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

- When a programme reads a value from the buffer, it begins from the beginning of the list (oldest data). Once a byte of data has been read, it is removed from the buffer and the data at the second position on the list moves to the top of the list, etc.
- The buffer's length is finite. This signifies there is a maximum length for the list. What happens when the sensor tries to send new data to the buffer once it is completely full? To make room for new data at the bottom of the list, the oldest data (at the top of the list) is destroyed forever and all other items are moved upward.

3.5.2 Simulink to App Design for GUI

Understanding the types of parameters is crucial for designing buttons and displays that function properly in an app. This project uses simulink to send and receive data to generate a graphical user interface in App Designer. When utilising the GUI in App Designer by altering the value and continuing to alter the value in simulink, the result is identical to Figure 3.22. There, can also see simulink output in App Designer. To update the data value in a Simulink function in the button designer, the code must be written as shown in Figure 3.21, using "set param" to alter the input block's value. To display output from simulink, "get param" must be used.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA



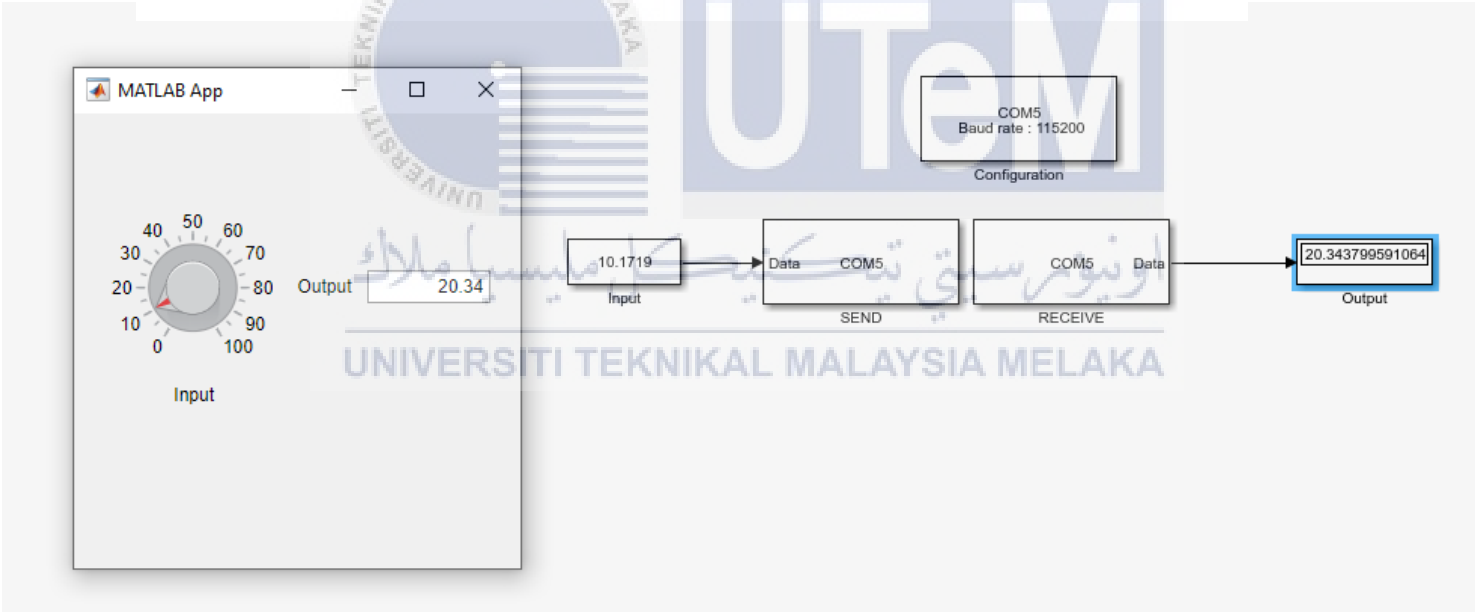

```
% Value changing function: InputKnob
  function InputKnobValueChanging(app, event)

      changingValue = event.Value;

      set_param([bdroot,'/Input'],'Value',
num2str(changingValue));
```

```
function OutputEditFieldValueChanged(app, event)
rto = get_param('test1single/Output','RuntimeObject');
str = double(rto.InputPort(1).Data);
app.OutputEditField.Value = str;
end
end
```

Figure 3.21 App Design Coding



UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Figure 3.22 GUI with Simulink

3.6 Demonstration Hardware and Software for Functoriality

For this project's results and simulations, only existing sensors and actuators will be used. This is because there are insufficient components, and it is just a test to ensure that the hardware and software are functioning properly. Referring to Figure 3.23, a circuit has been constructed based on the analysis of the robot. This project uses only 5 actuators and 2 sensors simulations and results are sufficient proof that the hardware and software are functioning properly.

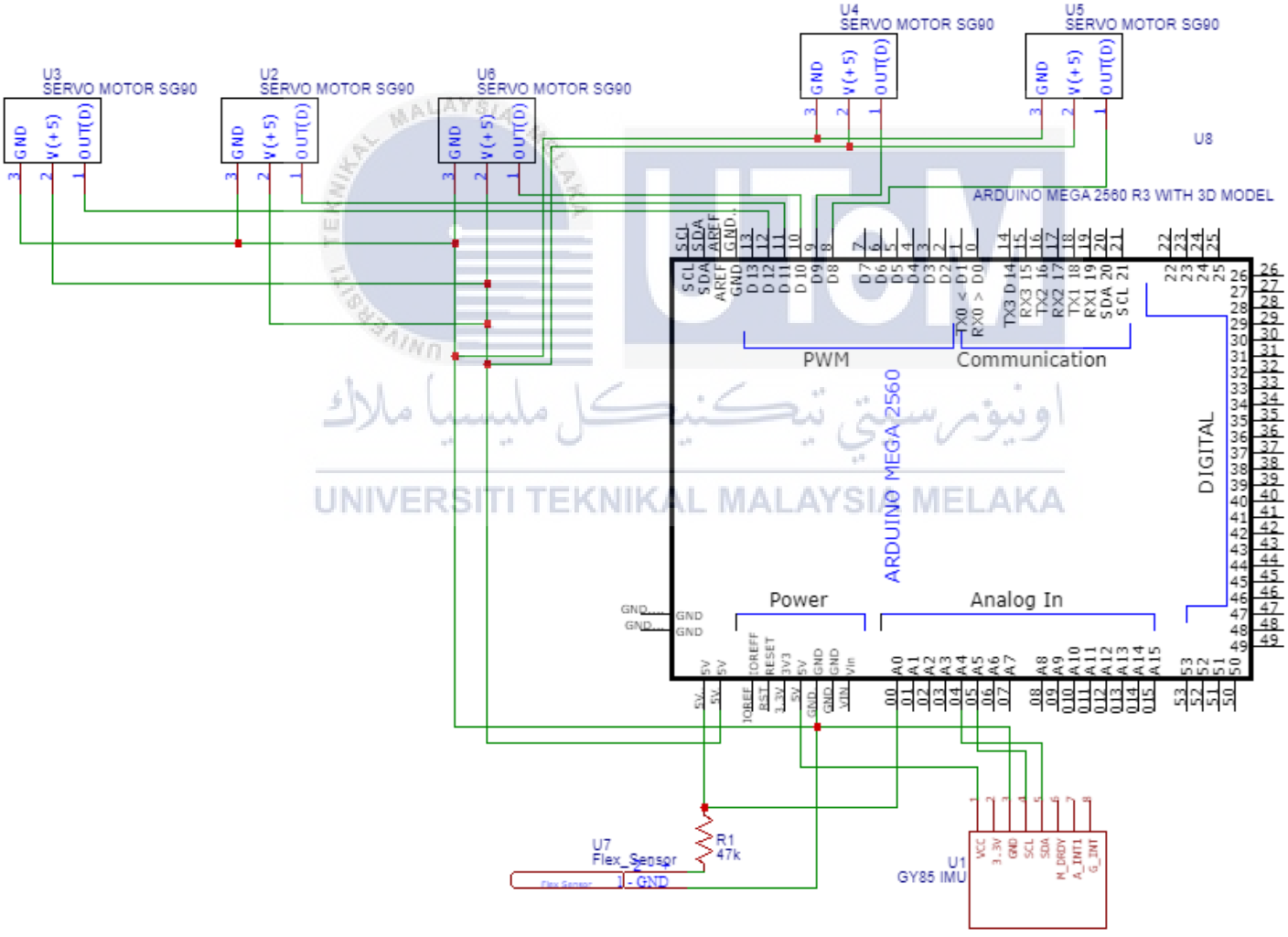


Figure 3.23 Circuit Prototype Cili Padi Picking Robot

CHAPTER 4

RESULTS AND DISCUSSION

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

This chapter presents the results and discussion of the analysis and development of the Cili-Padi Picking Robot Framework. Focuses on the implementation of an embedded software interfaced with a graphical user interface (GUI) utilizing both Arduino and MATLAB for communication. Further, it discusses the design and simulation of the GUI using Simulink and the functionality of the embedded software in real-time.

4.1 Analysis of Cili-Padi Picking Robot Framework

A framework for the Cili-Padi Picking Robot, it is necessary to analyse the robot's structure and the parameters used by the Cili-Padi Picking Robot. Below are the parameters utilised by the Cili-Padi Picking Robot. List table 4.1 is a list sensor and actual with data parameter type.

Table 4.1 Sensor and Actuator Setup Check List

No	Component	Type	Data Parameter	Quantity	Checklist/ comment
1	Stepper Motor	Actuator	PWM	2	For navigation
2	Analog Servo Motor	Actuator	PWM	2	1. To rotate the end effector 2. For gripping mechanism
3	Digital servo Motor	Actuator	PWM	3	Delta robot arm
4	FSR/Flex Sensor	Sensor	1. Voltage 2. Bend (degree)	8	For navigation system
5	IMU (9DOF)	Sensor	1. Accelerator 2. Gyro	1	Delta robot arm
6	Camera	Sensor/ MATALB		1	Picking mechanism

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

There are three components that make up the Cili-Padi Picking Robot's overall system. The very first navigation system, which allowed the user to control the movement of the robot by manual navigation. In the navigation system, the sensors are comprised of 8 Flex Sensor/FSR, the actuators are comprised of 1 stepper motor, and the data parameter transmission is PWM. The Delta Root Arm system makes up the second component of the systems. In the Delta robot arm, there is one IMU sensor and three digital servos that operate as actuators. In order to solve the Direct Kinematics Problem of Parallel Mechanisms, the data from the Accelerator and the Gyroscope in the IMU sensor will be employed. A servo motor

will be used as an actuator for the mechanism's ability to rotate and grip. In the final step of the Vision system, a camera and an algorithm are used to identify Cili-Padi.

4.2 Develop Embedded Software Interfaced with Graphical User Interface (GUI).

To demonstrate the outcome in embedded software, there will be two sections shown. First portion, the outcome was that MATLAB and Arduino were able to connect with each other and send and receive data. In the second section, App Design and Simulink collaborate to design graphical user interfaces by communicating with one another (GUI). To demonstrate that the result is correct, both App Design and Simulink will display the same data receive, and users will be able to switch the data displayed in the GUI between App Design and Simulink.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

4.2.1 Arduino and MATLAB communicate for Cili-Padi Picking Robot

Figure 4.1 provides an overview of the results obtained by having Simulink and Arduino communicate with one another using serial communication. The data that has to be sent from Simulink to Arduino can be found in the left block.

Figure 4.2 shows a multiplexer being used in the zoomed-in photo on the block's left. This allows for the transmission of several data streams. This is due to the fact that the robot Cili Padi Picking Robot has a large number of inputs to control and an interface with GUI. Block for transmitting and receiving data from Arduino can be found in the middle of Figure 4.3.

Using the same COM Port with Arduino requires setting up the appropriate configuration and checking in the device manager to see what number COM port is connected. In this case, COM Port 4 serves as an interface, and will need to configure Simulink accordingly. A block that may receive data from Arduino can be seen in Figure 4.4. Naturally, Simulink will get multiple data at the same time. A demultiplexer is needed to receive the several data received by the COM integration.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

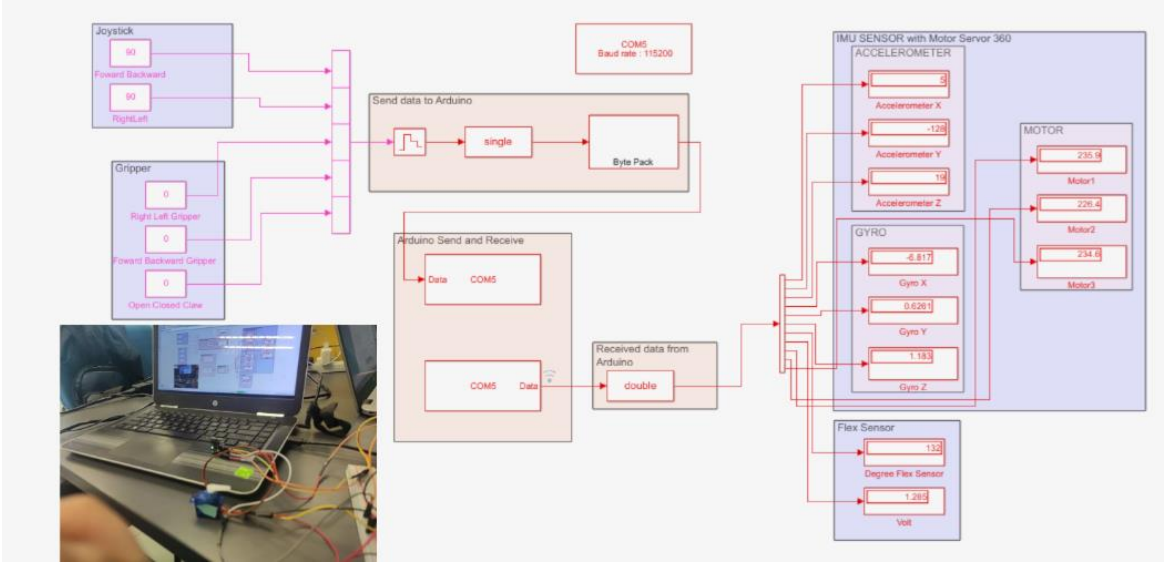


Figure 4.1 Arduino Communicate with MATLAB

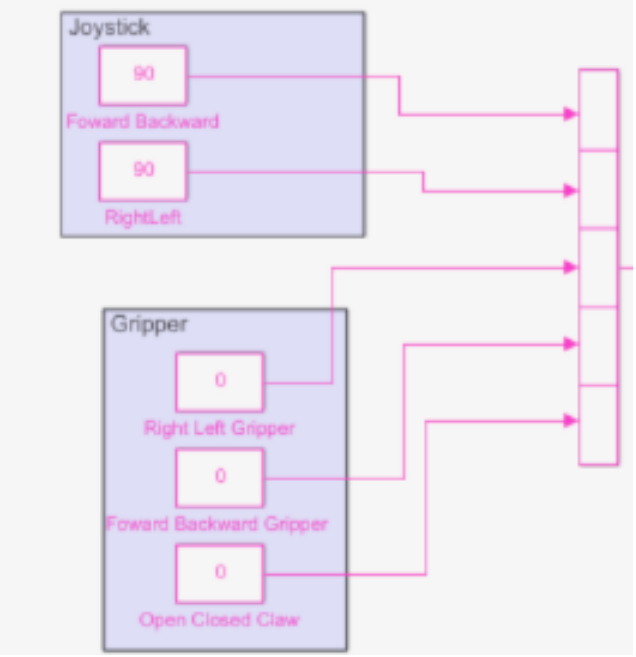


Figure 4.2 Send data from MATLAB to Arduino

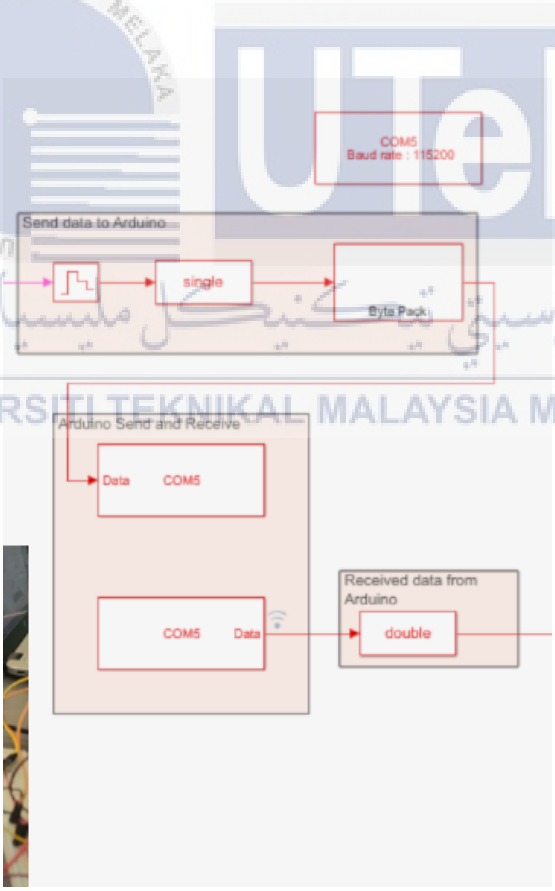


Figure 4.3 Block Send And Receive Data In

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

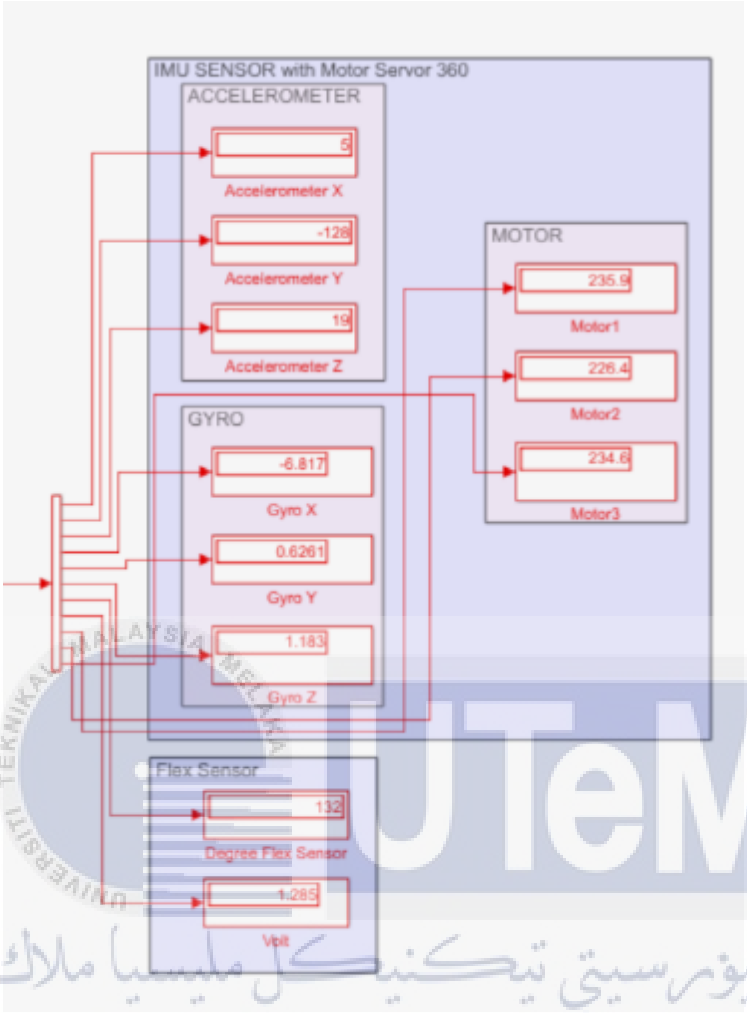


Figure 4.4 Receive Data from Arduino

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

4.2.2 App Design and Simulink communicate for GUI Cili-Padi Picking Robot

Table 4.2 illustrates the GUI's overall design. In left side bar have a menu. In menu have 4. Navigation, Delta Robot Arm, Gripper arm, and finally Robot Vision are accessible via four menus.

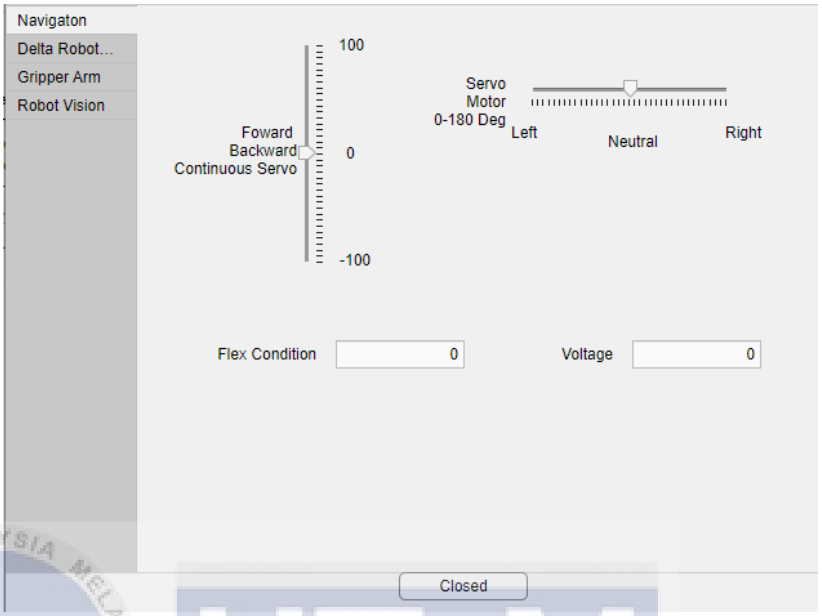
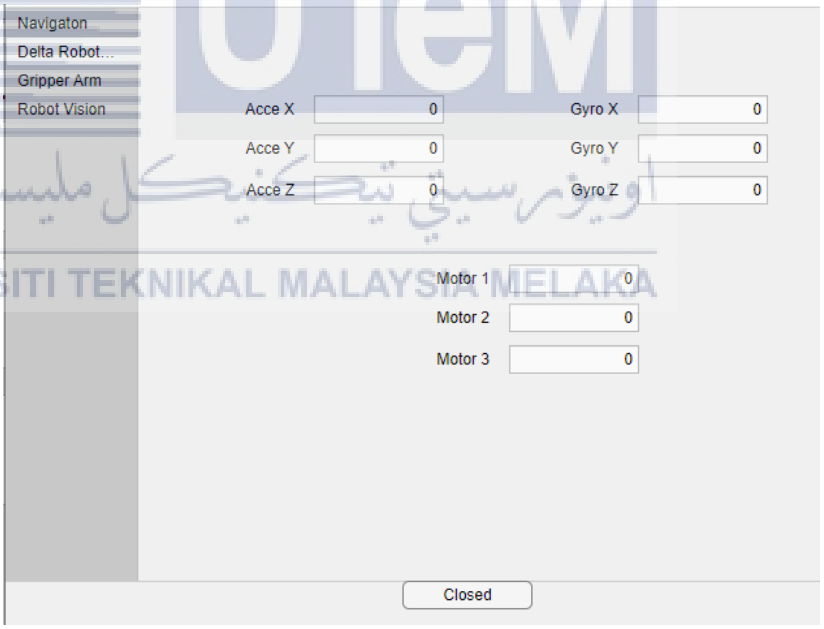
There are two as an input and two as an output in navigation. In Forward-backward, and left-right inputs operate the robot similarly to a joystick. And for output, display the Flex condition value and the Voltage produced by the Flex sensor.

In the Delta arm area, there is simply output, but a three-group Display. group 1 and group 2 for reading from sensor IMU, Sensor IMU displays Accelerometer and Gyroscope readings, however each reading has three output axes: X, Y, and Z.

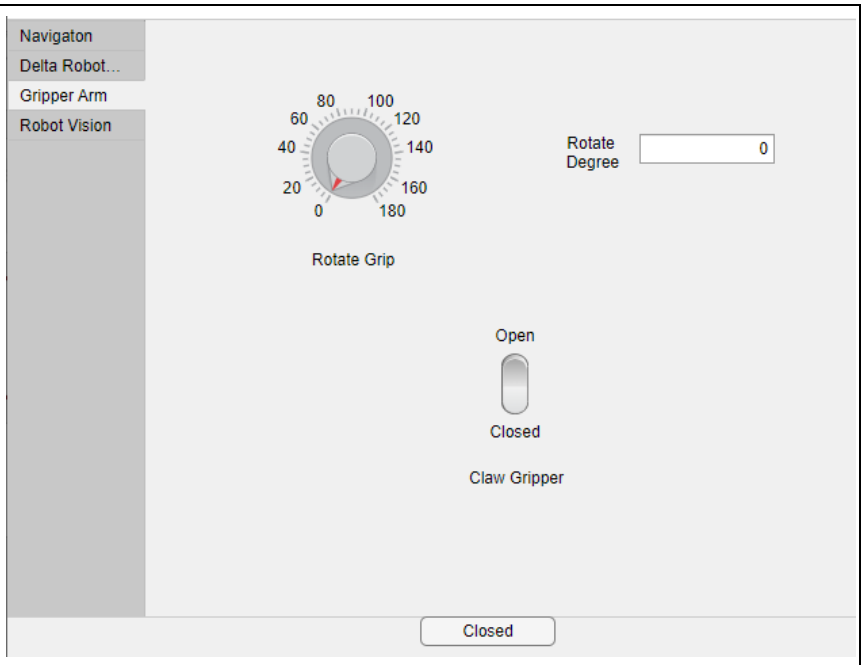
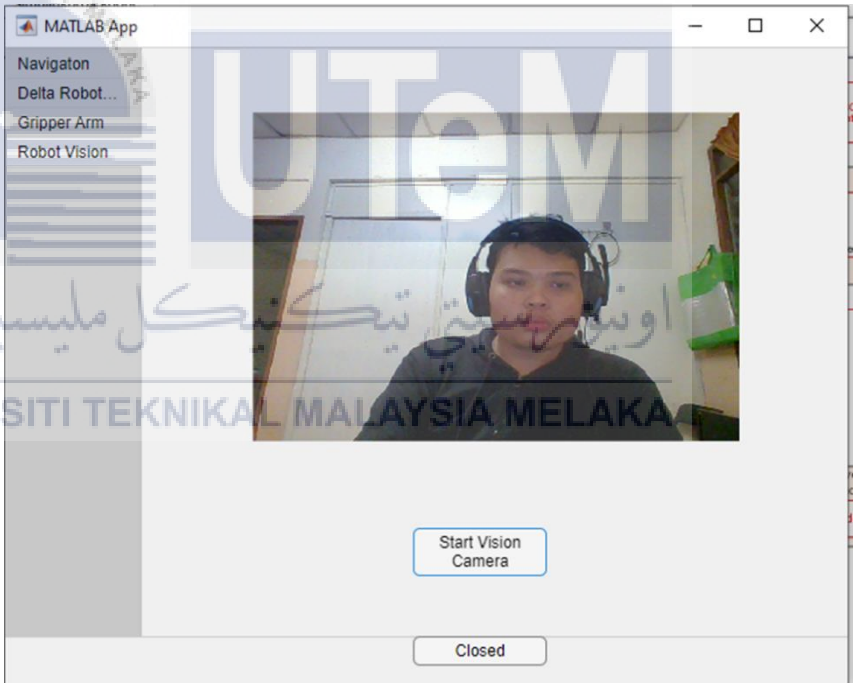
In Gripper Arm, there are only 3 inputs. 3 inputs are knob rotation and edit field. Rotate the grip and then press the button to open and close the claw gripper. Robot Vision has two First have a recognition button and one display camera.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

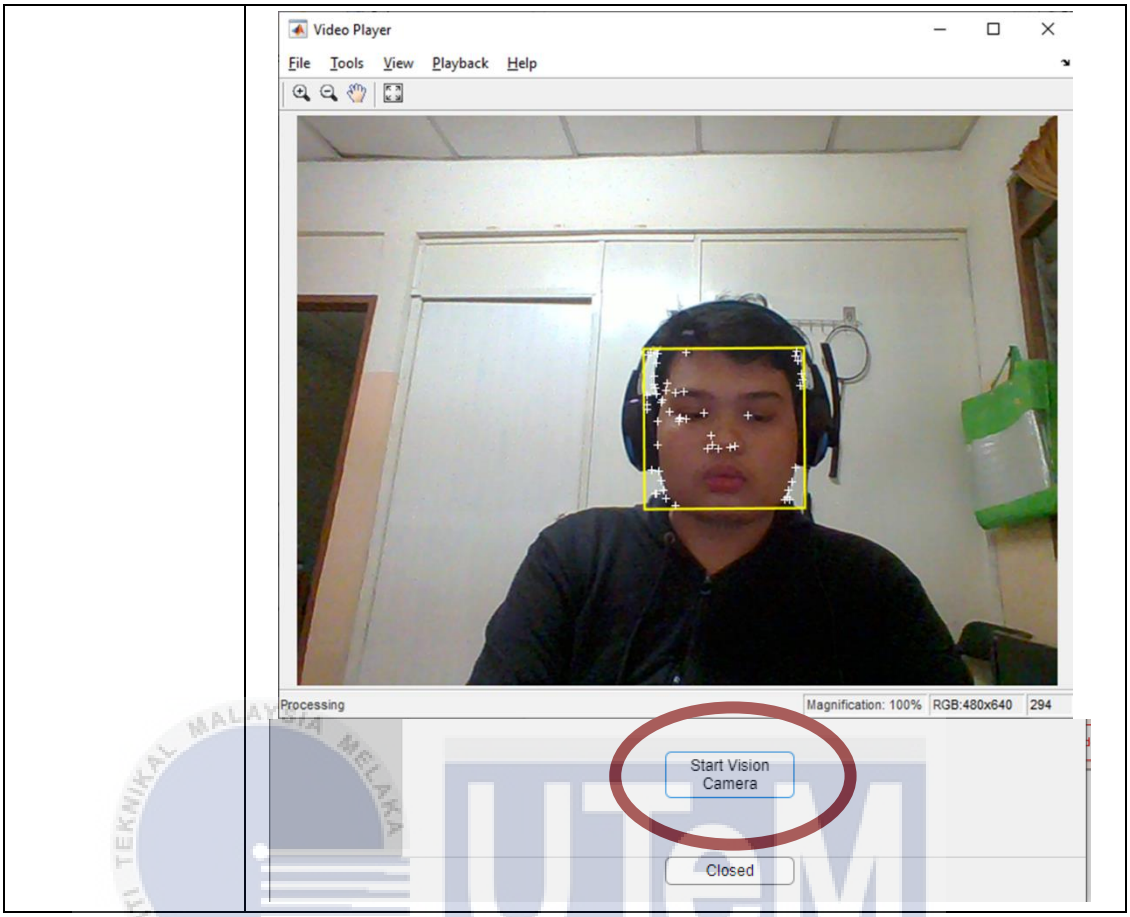
Table 4.2 Overall Design in GUI

Menu in GUI	Part of Menu Interface
Navigation	
Delta Robot Arm	

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Gripper Arm	
Robot Vision	

UNIVERSITI TEKNIKAL MALAYSIA MELAKA



UNIVERSITI TEKNIKAL MALAYSIA MELAKA
اونيۏرسيتي تيکنيکل مليسيا ملاک

Figure 4.5 demonstrates that App Design as a GUI may access the Simulink output value. This is an example showing simultaneous Accelerometer and Gyroscope readings with Simulink. And as illustrated in Figure 4.6, both the button and input can be modified simultaneously in Simulink.

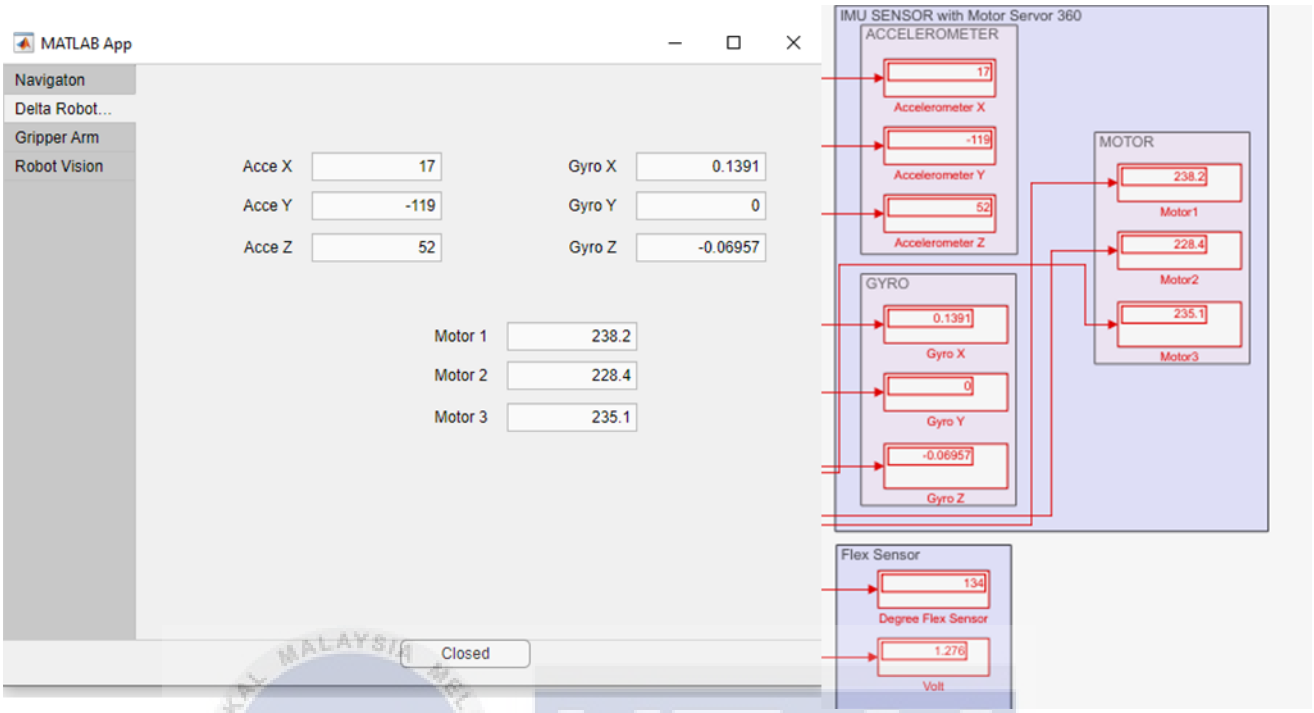


Figure 4.5 App Design Read Output Simulink

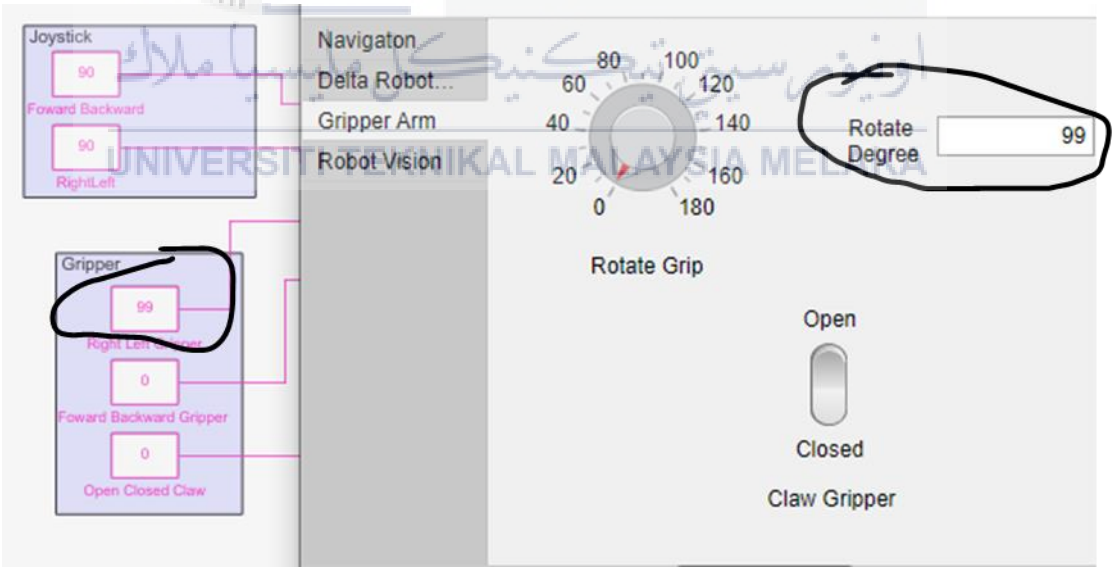


Figure 4.6 App Design Changed Value in Simulink Input.

4.3 Functionality of the Embedded Software in Real Time

According to the graphs in Figures 4.7 and 4.8, Arduino and IMU sensor readings are depicted. This demonstrates Simulink ability to manipulate and receive data in real time and without delay. Here can see that the Gyro and Accelerometer data changes are received simultaneously and without any data latency, as depicted by the graph. Using only serial transmission, the received data is not lost even if it is delivered concurrently.

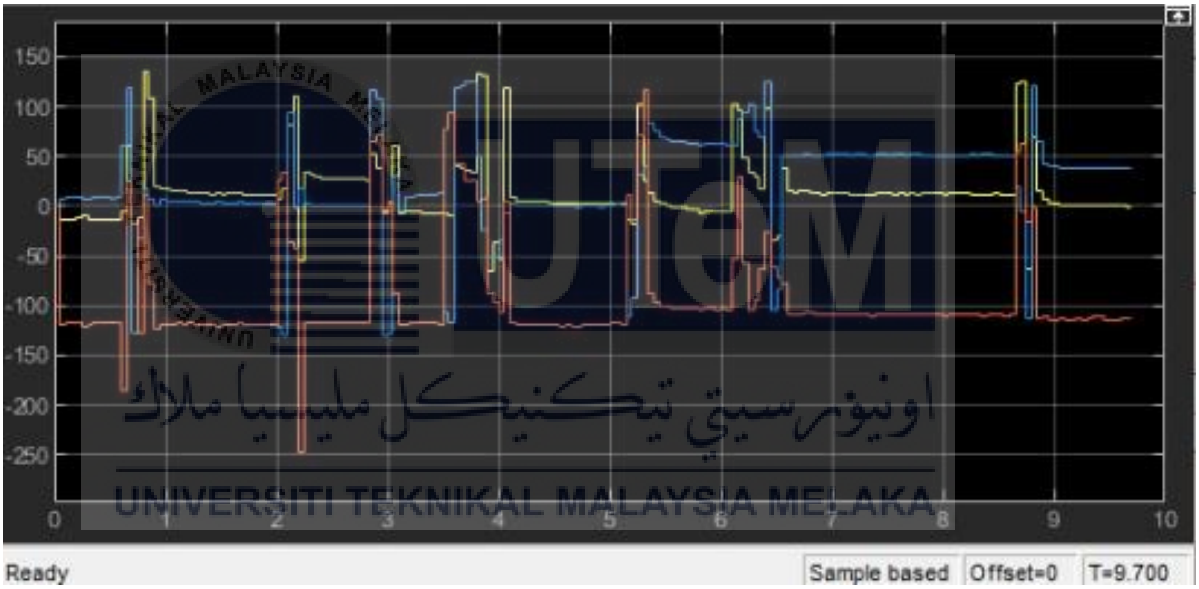


Figure 4.7 Reading Accelerometer in Simulink

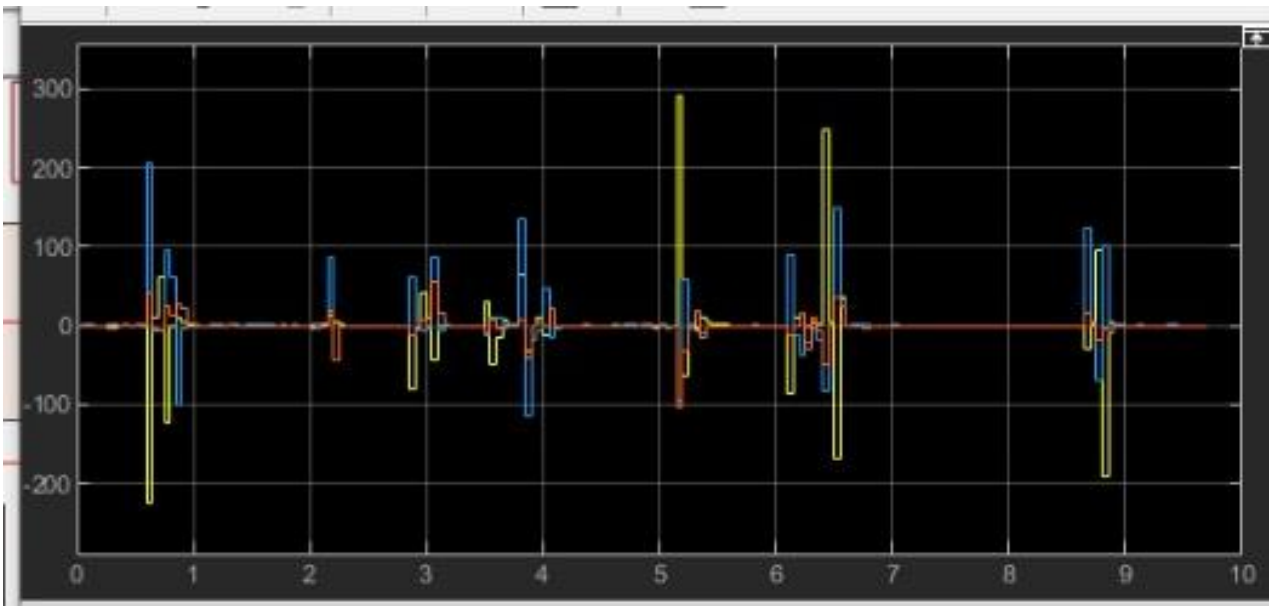


Figure 4.8 Reading Gyro in Simulink

Two sensors send data from Arduino to Simulink in Figure 4.9. From left side and middle side is a gyro sensor. In right side is flex sensor. When the IMU sensor is moved, both the Gyroscope and accelerometer graphs will change concurrently. And when the Flex sensor is bend, the graph will immediately and responsively shift. Figure 4.10 show the hardware for embedded software and GUI testing.

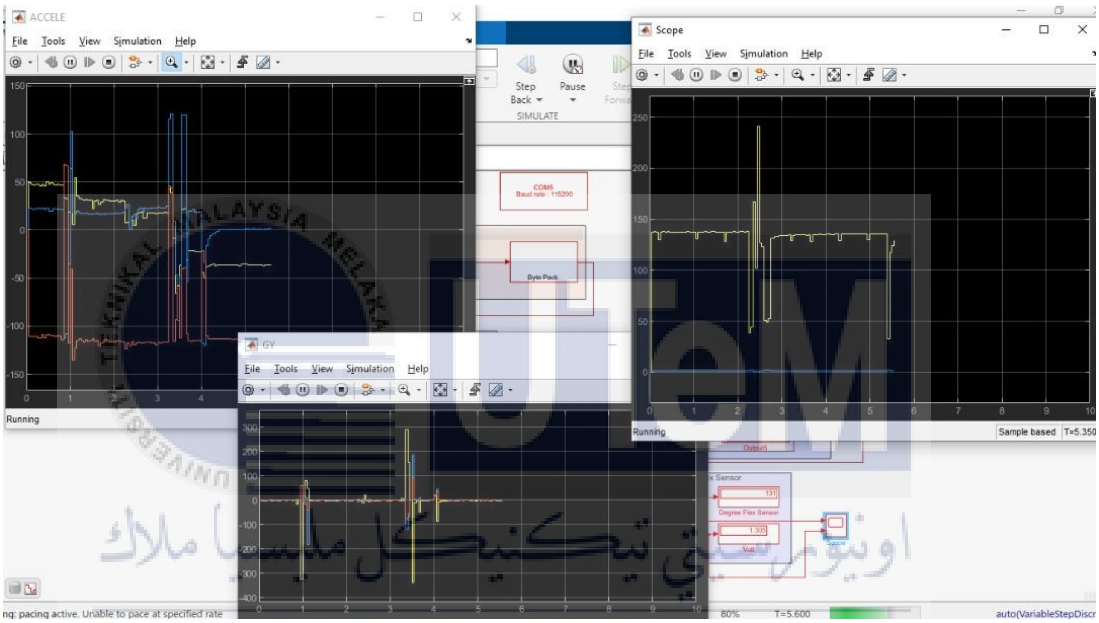


Figure 4.9 Graph Flex Sensor and IMU Sensor

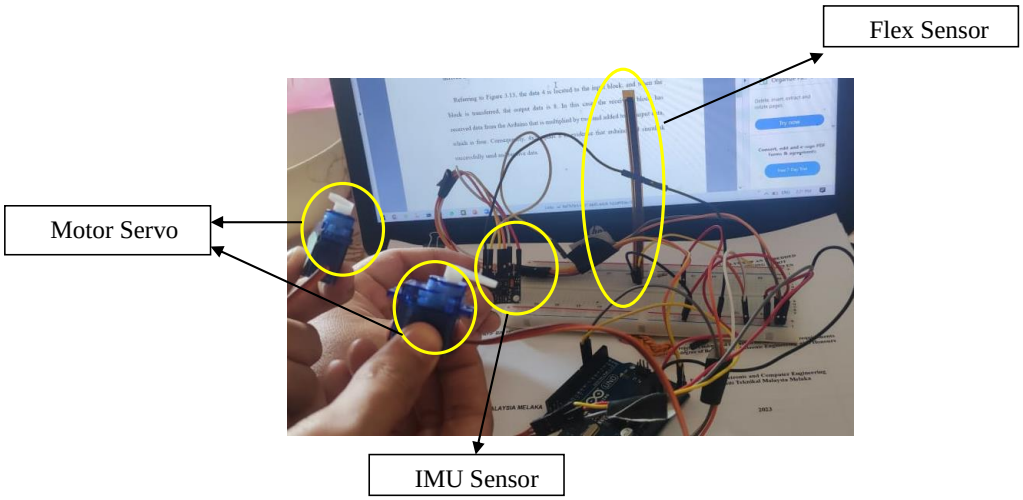


Figure 4.10 Hardware for Testing

CHAPTER 5

CONCLUSION AND FUTURE WORKS

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

5.1 Conclusion

As a conclusion, the main achievement of this project was to Development and Analysis of An Embedded Software for Cili-Padi Picking Robot Framework Using MATLAB. The project successfully defined the system requirements and developed software that was able to communicate with the hardware and control the actuators. However, there was a limitation in the data transmission between MATLAB and Arduino, which resulted in occasional data crashes due to the delay in the coding. Despite this limitation, the hardware robot was able to function in real-time with a fast response from the software. Overall, the project successfully achieved its

objectives and demonstrated the functionality of the embedded software in a real-time setting.

5.2 Future Works

It is recommended that in future work, a sufficient number of sensors and actuators be used, and that real vision be used for the Cili Padi picking function of this robot, so that the robot algorithm can think like a human. This will allow the Cili Padi picking function of this robot to be more accurate.

Fixing these and other flaws can inspire future GUI development that is more effective and suitable. Therefore, for future development on this project, it is suggested that an IOT graphical user interface (GUI) with several customizable buttons and an adaptable display be used. For instance, suggested advocate IoT via a web-based application. This is because "App Design" in MATLAB is limited in its ability to edit relevant buttons and provides a limited number of buttons and displays.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

REFERENCES

[1] X. Xu, Y. Lu, B. Vogel-Heuser, and L. Wang, "Industry 4.0 and Industry 5.0—Inception, conception and perception," *J Manuf Syst,* vol. 61, 2021, doi: 10.1016/j.jmsy.2021.10.006.

[2] Z. Saidah, Harianto, S. Hartoyo, and R. W. Asmarantaka, "Change on Production and Income of Red Chili Farmers," in *IOP Conference Series: Earth and Environmental Science,* 2020, vol. 466, no. 1. doi: 10.1088/1755-1315/466/1/012003.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

[3] M. L. Rajaram, E. Kougianos, S. P. Mohanty, and U. Choppali, "Wireless Sensor Network Simulation Frameworks: A Tutorial Review: MATLAB/Simulink bests the rest," *IEEE Consumer Electronics Magazine,* vol. 5, no. 2, 2016, doi: 10.1109/MCE.2016.2519051.

[4] B. Ramalingam *et al.*, "A human support robot for the cleaning and maintenance of door handles using a deep-learning framework," *Sensors (Switzerland),* vol. 20, no. 12, 2020, doi: 10.3390/s20123543.

[5] L. Prokhorenko, D. Klimov, D. Mishchenkov, and Y. Poduraev, "Surgeon–robot interface development framework," *Comput Biol Med*, vol. 120, 2020, doi: 10.1016/j.compbiomed.2020.103717.

[6] J. Casas *et al.*, "Architecture for a Social Assistive Robot in Cardiac Rehabilitation," in *2018 IEEE 2nd Colombian Conference on Robotics and Automation, CCRA 2018*, 2018. doi: 10.1109/CCRA.2018.8588133.

[7] C. M. Ranieri, G. Nardari, A. H. M. Pinto, D. C. Tozadore, and R. A. F. Romero, "LARa: A robotic framework for human-robot interaction on indoor environments," in *Proceedings - 15th Latin American Robotics Symposium, 6th Brazilian Robotics Symposium and 9th Workshop on Robotics in Education, LARS/SBR/WRE 2018*, 2018. doi: 10.1109/LARS/SBR/WRE.2018.00074.

[8] A. Najar, O. Sigaud, and M. Chetouani, "Teaching a robot with unlabeled instructions: The TICS architecture," in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, 2021, vol. 3.

[9] "Proceedings of International Conference on Advanced Systems and Emergent Technologies, IC_ASET 2019," *Proceedings of International Conference on Advanced Systems and Emergent Technologies, IC_ASET 2019*. 2019.

[10] "Proceedings - 16th Annual International Conference on Distributed Computing in Sensor Systems, DCOSS 2020," *Proceedings - 16th Annual*

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

International Conference on Distributed Computing in Sensor Systems, DCOSS 2020. 2020.

[11] W. Liu, H. Wu, Z. Jiang, Y. Gong, and J. Jin, "A robotic communication middleware combining high performance and high reliability," in *Proceedings - Symposium on Computer Architecture and High Performance Computing*, 2020, vol. 2020-September. doi: 10.1109/SBAC-PAD49847.2020.00038.

[12] J. Nowell, J. Connor, B. Champion, and M. Joordens, "Coaxial magnetic drivetrain for robotic stingrays," in *World Automation Congress Proceedings*, 2021, vol. 2021-August. doi: 10.23919/WAC50355.2021.9559500.

[13] J. Nowell, J. Connor, B. Champion, and M. Joordens, "Coaxial magnetic drivetrain for robotic stingrays," in *World Automation Congress Proceedings*, 2021, vol. 2021-August. doi: 10.23919/WAC50355.2021.9559500.

[14] C. Aakash and V. Manoj Kumar, "Path Planning of an UAV with the Help of Lidar for Slam Application," in *IOP Conference Series: Materials Science and Engineering*, 2020, vol. 912, no. 6. doi: 10.1088/1757-899X/912/6/062013.

[15] N. Baras, G. Nantzios, D. Ziouzios, and M. Dasygenis, "Autonomous Obstacle Avoidance Vehicle Using LIDAR and an Embedded System," in *2019 8th International Conference on Modern Circuits and Systems Technologies, MOCAST 2019*, 2019. doi: 10.1109/MOCAST.2019.8742065.

[16] "2021 Index IEEE Transactions on Industrial Informatics Vol. 17," *IEEE Trans Industr Inform*, vol. 17, no. 12, 2022, doi: 10.1109/tii.2021.3138206.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

[17] A. Vidanapathirana, S. D. Dewasurendra, and S. G. Abeyaratne, “Model in the loop testing of complex reactive systems,” in *2013 IEEE 8th International Conference on Industrial and Information Systems, ICIIS 2013 - Conference Proceedings*, 2013. doi: 10.1109/ICIInfS.2013.6731950.

[18] D. Mustafa, “A Survey of Performance Tuning Techniques and Tools for Parallel Applications,” *IEEE Access*, vol. 10, 2022, doi: 10.1109/ACCESS.2022.3147846.

[19] Y. Tang, “Research on medical device software development and design based on CMMI model,” *International Journal Bioautomation*, vol. 23, no. 4, 2019, doi: 10.7546/ijba.2019.23.4.000625.

[20] I. Nastjuk, B. Herrenkind, M. Marrone, A. B. Brendel, and L. M. Kolbe, “What drives the acceptance of autonomous driving? An investigation of acceptance factors from an end-user’s perspective,” *Technol Forecast Soc Change*, vol. 161, 2020, doi: 10.1016/j.techfore.2020.120319.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

[21] E. Navas, R. Fernández, D. Sepúlveda, M. Armada, and P. Gonzalez-De-santos, “Soft grippers for automatic crop harvesting: A review,” *Sensors*, vol. 21, no. 8. 2021. doi: 10.3390/s21082689.

[22] T. Schneider *et al.*, “Maplab: An Open Framework for Research in Visual-Inertial Mapping and Localization,” *IEEE Robot Autom Lett*, vol. 3, no. 3, 2018, doi: 10.1109/LRA.2018.2800113.

[23] D. Shi, W. Zhang, W. Zhang, and X. Ding, “A Review on Lower Limb Rehabilitation Exoskeleton Robots,” *Chinese Journal of Mechanical*

Engineering (English Edition), vol. 32, no. 1. 2019. doi: 10.1186/s10033-019-0389-8.

[24] P. Hambarde, R. Varma, and S. Jha, "The survey of real time operating system: RTOS," in *Proceedings - International Conference on Electronic Systems, Signal Processing, and Computing Technologies, ICESC 2014*, 2014. doi: 10.1109/ICESC.2014.15.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

APPENDICES

Appendix A Coding in Arduino

```
#include <Arduino_FreeRTOS.h>
#include "GY_85.h"
#include <Wire.h>
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
Servo myservo1;  // create servo object to control a servo
Servo servo0;
Servo servo1;
Servo servo2;
Servo servo3;

int button;
int servovalue;

int minVal = 265;
int maxVal = 402;

double x;
double y;
double z;
```

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

```
////////////////////////////FLEX SENSOR/////////////////////////////////
const int flexPin = A0;       // Pin connected to voltage divider output

// Change these constants according to your project's design
const float VCC = 5;       // voltage at Ardunio 5V line
const float R_DIV = 47000.0;  // resistor used to create a voltage divider
const float flatResistance = 25000.0; // resistance when flat
const float bendResistance = 100000.0;  // resistance at 90 deg
////////////////////////////////////////////////////////////////////////////

GY_85 GY85;     //create the object
void Tasksimulink( void *pvParameters );
void TaskImusensor( void *pvParameters );
void TaskFlexsensor( void *pvParameters );
void TaskMotorcontrolfb( void *pvParameters );
//void TaskClaw( void *pvParameters );

typedef union {
  float number;
  uint8_t bytes[4];
} FLOATUNION_t;
// Create the variables you want to receive
FLOATUNION_t myValue1;
FLOATUNION_t myValue2;
FLOATUNION_t myValue3;
FLOATUNION_t myValue4;
FLOATUNION_t myValue5;
FLOATUNION_t myValue6;

// Create the variables to send
FLOATUNION_t send1;
FLOATUNION_t send2;
FLOATUNION_t send3;
FLOATUNION_t send4;
FLOATUNION_t send5;
FLOATUNION_t send6;
FLOATUNION_t send7;
FLOATUNION_t send8;
FLOATUNION_t send9;
FLOATUNION_t send10;
FLOATUNION_t send11;
FLOATUNION_t send12;
```

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

```
void setup() {
  Serial.begin(115200);

  xTaskCreate(Tasksimulink, "Task1", 128, NULL, 1, NULL);
  xTaskCreate(TaskImusensor, "Task2", 128, NULL, 1, NULL);
  xTaskCreate(TaskFlexsensor, "Task3", 128, NULL, 1, NULL);
  xTaskCreate(TaskMotorcontrolfb, "Task4", 128, NULL, 1, NULL);
  //  xTaskCreate(TaskClaw, "Task5", 128, NULL, 1, NULL);

  vTaskStartScheduler();

}


void loop() {

}

void Tasksimulink(void *pvParameters)
{

  while (1)
  {
    myValue1.number = getFloat(); // Give your float a value
```

UTeM
اونيۏرسيتي تيکنيکل مليسيا ملاک
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

```
myValue2.number = getFloat(); // Give your float a value
myValue3.number = getFloat(); // Give your float a value
myValue4.number = getFloat(); // Give your float a value
myValue5.number = getFloat(); // Give your float a value
myValue6.number = getFloat(); // Give your float a value

//  send1.number = 1;
//  send2.number = 2;
//  send3.number = 3;
//  send4.number = 4;
//  send5.number = 5;
//  send6.number = 6;
//  send7.number = 7;
//  send8.number = 8;

// Print header: Important to avoid sync errors!
Serial.write('A');

// Print float data
for (int i = 0; i < 4; i++) {
  Serial.write(send1.bytes[i]);
}
for (int i = 0; i < 4; i++) {
  Serial.write(send2.bytes[i]);
}
for (int i = 0; i < 4; i++) {
  Serial.write(send3.bytes[i]);
}
for (int i = 0; i < 4; i++) {
  Serial.write(send4.bytes[i]);
}
for (int i = 0; i < 4; i++) {
  Serial.write(send5.bytes[i]);
}
for (int i = 0; i < 4; i++) {
  Serial.write(send6.bytes[i]);
}
for (int i = 0; i < 4; i++) {
  Serial.write(send7.bytes[i]);
}
for (int i = 0; i < 4; i++) {
  Serial.write(send8.bytes[i]);
}
for (int i = 0; i < 4; i++) {
  Serial.write(send9.bytes[i]);
}
```

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

```
    for (int i = 0; i < 4; i++) {
      Serial.write(sendl0.bytes[i]);
    }
    for (int i = 0; i < 4; i++) {
      Serial.write(sendl1.bytes[i]);
    }
    for (int i = 0; i < 4; i++) {
      Serial.write(sendl2.bytes[i]);
    }
    // Print terminator
    Serial.print('\n');

    // Use the same delay in the Serial Receive block
    delay(800);
  }
}

void TaskImusensor(void *pvParameters)

{
  servo0.attach(9);
  servo1.attach(10);
  servo2.attach(11);
  Wire.begin();
  delay(10);
  GY85.init();
  delay(10);

  while (1)

  {

    int ax = GY85.accelerometer_x( GY85.readFromAccelerometer() );
    int ay = GY85.accelerometer_y( GY85.readFromAccelerometer() );
    int az = GY85.accelerometer_z( GY85.readFromAccelerometer() );

    float gx = GY85.gyro_x( GY85.readGyro() );
    float gy = GY85.gyro_y( GY85.readGyro() );
    float gz = GY85.gyro_z( GY85.readGyro() );

    int xAng = map(ax, minVal, maxVal, -90, 90);
    int yAng = map(ay, minVal, maxVal, -90, 90);
    int zAng = map(az, minVal, maxVal, -90, 90);
```

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

```
    x = RAD_TO_DEG * (atan2(-yAng, -zAng) + PI);
    y = RAD_TO_DEG * (atan2(-xAng, -zAng) + PI);
    z = RAD_TO_DEG * (atan2(-yAng, -xAng) + PI);
    int servo0Value = map(x, 0, 360, 0, 180);
    int servo1Value = map(y, 0, 360, 0, 180);
    int servo2Value = map(z, 0, 360, 0, 180);

    send1.number = (ax);
    send2.number = (ay);
    send3.number = (az);

    send4.number = (gx);
    send5.number = (gy);
    send6.number = (gz);

    send9.number = (x);
    send10.number = (y);
    send11.number = (z);
    delay(500);

  }

}
void TaskFlexsensor(void *pvParameters)

{
  pinMode(flexPin, INPUT);


  while (1)

  {
    int ADCflex = analogRead(flexPin);
    float Vflex = ADCflex * VCC / 1023.0;
    float Rflex = R_DIV * (VCC / Vflex - 1.0);

    // Serial.println("Resistance: " + String(Rflex) + " ohms");

    // Use the calculated resistance to estimate the sensor's bend angle:
    float angle = map(Rflex, flatResistance, bendResistance, 0, 90.0);
    //Serial.println("Bend: " + String(angle) + " degrees");
    //Serial.println();
    send7.number = (angle);
    send8.number = (Vflex);

    delay(500);

  }

}
```

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

```
void TaskMotorcontrolfb(void *pvParameters)  {

  myservo.attach(6);
  myservo1.attach(5);
  while (1)
  {
    myservo.write(myValue1.number);
    myservo1.write(myValue2.number);
    delay(100);
   send12.number =   myValue1.number;

  }


}


float getFloat() {
  int cont = 0;
  FLOATUNION_t f;
  while (cont < 4 ) {
    f.bytes[cont] = Serial.read() ;
    cont = cont + 1;
  }
  return f.number;
}
```

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Appendix B Coding of App Design(MATLAB)

```
classdef Gui < matlab.apps.AppBase
    % Properties that correspond to app components
    properties (Access = public)
        simulinkprojekdone              matlab.ui.Figure
        ClosedButton                    matlab.ui.control.Button
        TabGroup                        matlab.ui.container.TabGroup
        NavigatonTab_3                  matlab.ui.container.Tab
        NeutralLabel                    matlab.ui.control.Label
        LeftLabel                       matlab.ui.control.Label
        RightLabel                      matlab.ui.control.Label
        VoltageEditField                matlab.ui.control.NumericEditField
        AcceYLabel_3                    matlab.ui.control.Label
        FlexconditionField              matlab.ui.control.NumericEditField
        FlexConditionLabel              matlab.ui.control.Label
        StopLabel                       matlab.ui.control.Label
        BackwardLabel                   matlab.ui.control.Label
        FowardLabel                     matlab.ui.control.Label
        ServoMotor0180DegSlider_3       matlab.ui.control.Slider
        ServoMotor0180DegSlider_3Label  matlab.ui.control.Label
        FowardBackwardContinuousServoSlider_3  matlab.ui.control.Slider
        FowardBackwardContinuousServoSlider_3Label  matlab.ui.control.Label
        DeltaRobotArmTab                matlab.ui.container.Tab
        Motor3EditField                 matlab.ui.control.NumericEditField
        Motor3EditFieldLabel            matlab.ui.control.Label
        Motor2EditField                 matlab.ui.control.NumericEditField
        Motor2EditFieldLabel            matlab.ui.control.Label
        Motor1EditField                 matlab.ui.control.NumericEditField
        Motor1EditFieldLabel            matlab.ui.control.Label
        GyroZEditField                  matlab.ui.control.NumericEditField
        GyroZEditFieldLabel             matlab.ui.control.Label
        GyroYEditField                  matlab.ui.control.NumericEditField
        GyroYEditFieldLabel             matlab.ui.control.Label
        GyroXEditField                  matlab.ui.control.NumericEditField
        GyroXEditFieldLabel             matlab.ui.control.Label
        ZAcceEditField                  matlab.ui.control.NumericEditField
        AcceYLabel_2                    matlab.ui.control.Label
        YAcceEditField                  matlab.ui.control.NumericEditField
        AcceYLabel                      matlab.ui.control.Label
        XAcceEditField                  matlab.ui.control.NumericEditField
        AcceXEditFieldLabel             matlab.ui.control.Label
        GripperArmTab                   matlab.ui.container.Tab
        ClawGripperSwitch               matlab.ui.control.RockerSwitch
        ClawGripperSwitchLabel          matlab.ui.control.Label
        RotateDegreeEditField           matlab.ui.control.NumericEditField
        RotateDegreeEditFieldLabel      matlab.ui.control.Label
        RotateGripKnob                  matlab.ui.control.Knob
        RotateGripKnobLabel             matlab.ui.control.Label
        RobotVisionTab                  matlab.ui.container.Tab
        StartVisionCameraButton         matlab.ui.control.Button
        UIAxes                          matlab.ui.control.UIAxes
    end
```

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

```
properties (Access = private)
    timerObj %timer
end

methods (Access = public)

    function readacceX(app)

        rto = get_param([bdroot,'/Accelerometer X'],'RuntimeObject');
        app.XAcceEditField.Value = double(rto.InputPort(1).Data);
    end

    function readacceY(app)

        rto = get_param([bdroot,'/Accelerometer Y'],'RuntimeObject');
        app.YAcceEditField.Value = double(rto.InputPort(1).Data);
    end

     function readacceZ(app)

        rto = get_param([bdroot,'/Accelerometer Z'],'RuntimeObject');
        app.ZAcceEditField.Value = double(rto.InputPort(1).Data);
     end

     function readgyroX(app)

        rto = get_param([bdroot,'/Gyro X'],'RuntimeObject');
        app.GyroXEditField.Value = double(rto.InputPort(1).Data);
    end

    function readgyroY(app)

        rto = get_param([bdroot,'/Gyro Y'],'RuntimeObject');
        app.GyroYEditField.Value = double(rto.InputPort(1).Data);
    end

     function readgyroZ(app)

        rto = get_param([bdroot,'/Gyro Z'],'RuntimeObject');
        app.GyroZEditField.Value = double(rto.InputPort(1).Data);
     end

     function readflexsensor(app)

        rto = get_param([bdroot,'/Degree Flex Sensor'],'RuntimeObject');
        app.FlexconditionField.Value = double(rto.InputPort(1).Data);
     end
```

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

```
        function readvoltage(app)

            rto = get_param([bdroot,'/Volt'],'RuntimeObject');
            app.VoltageEditField.Value = double(rto.InputPort(1).Data);
        end

        function readmotor1(app)

            rto = get_param([bdroot,'/Motor1'],'RuntimeObject');
            app.Motor1EditField.Value = double(rto.InputPort(1).Data);
        end

        function readmotor2(app)

            rto = get_param([bdroot,'/Motor2'],'RuntimeObject');
            app.Motor2EditField.Value = double(rto.InputPort(1).Data);
        end

        function readmotor3(app)

            rto = get_param([bdroot,'/Motor3'],'RuntimeObject');
            app.Motor3EditField.Value = double(rto.InputPort(1).Data);
        end

    end

    properties (Access = private)
        webcamObject;
        imageObject;
end

    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)
 app.webcamObject = webcam;
        app.imageObject = image(app.UIAxes);
        axis(app.UIAxes,'ij')
        res = split(app.webcamObject.Resolution,'x');
        app.UIAxes.XLim = [0,str2double(res{1})];
        app.UIAxes.YLim = [0,str2double(res{2})];
        app.webcamObject.preview(app.imageObject);

set_param('simulinkprojekdone','SimulationCommand','start')
pause (10);

            app.timerObj = timer('TimerFcn',@(~,~) readacceX(app), 'Period',0.5, 'ExecutionMode', 'fixedSpacing', 'BusyMode','drop');
start(app.timerObj);
app.timerObj = timer('TimerFcn',@(~,~) readacceY(app), 'Period',0.5, 'ExecutionMode', 'fixedSpacing', 'BusyMode','drop');
start(app.timerObj);
 app.timerObj = timer('TimerFcn',@(~,~) readacceZ(app), 'Period',0.5, 'ExecutionMode', 'fixedSpacing', 'BusyMode','drop');
start(app.timerObj);

 app.timerObj = timer('TimerFcn',@(~,~) readgyroX(app), 'Period',0.5, 'ExecutionMode', 'fixedSpacing', 'BusyMode','drop');
start(app.timerObj);
app.timerObj = timer('TimerFcn',@(~,~) readgyroY(app), 'Period',0.5, 'ExecutionMode', 'fixedSpacing', 'BusyMode','drop');
start(app.timerObj);
 app.timerObj = timer('TimerFcn',@(~,~) readgyroZ(app), 'Period',0.5, 'ExecutionMode', 'fixedSpacing', 'BusyMode','drop');
start(app.timerObj);

app.timerObj = timer('TimerFcn',@(~,~) readflexsensor(app), 'Period',0.5, 'ExecutionMode', 'fixedSpacing', 'BusyMode','drop');
start(app.timerObj);
 app.timerObj = timer('TimerFcn',@(~,~) readvoltage(app), 'Period',0.5, 'ExecutionMode', 'fixedSpacing', 'BusyMode','drop');
start(app.timerObj);

app.timerObj = timer('TimerFcn',@(~,~) readmotor1(app), 'Period',0.5, 'ExecutionMode', 'fixedSpacing', 'BusyMode','drop');
start(app.timerObj);
app.timerObj = timer('TimerFcn',@(~,~) readmotor2(app), 'Period',0.5, 'ExecutionMode', 'fixedSpacing', 'BusyMode','drop');
start(app.timerObj);
app.timerObj = timer('TimerFcn',@(~,~) readmotor3(app), 'Period',0.5, 'ExecutionMode', 'fixedSpacing', 'BusyMode','drop');
start(app.timerObj);
```

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

```
        end

        % Value changing function: RotateGripKnob
        function RotateGripKnobValueChanging(app, event)
            changingValue = event.Value;
            set_param([bdroot,'/Right Left Gripper'],'Value', num2str(changingValue));
        end

        % Value changed function: RotateDegreeEditField
        function RotateDegreeEditFieldValueChanged(app, event)
          changingValue = event.Value;
          set_param([bdroot,'/Right Left Gripper'],'Value', num2str(changingValue));
        end

        % Button pushed function: StartVisionCameraButton
        function StartVisionCameraButtonPushed(app, event)
            faceDetector = vision.CascadeObjectDetector();

% Create the point tracker object.
pointTracker = vision.PointTracker('MaxBidirectionalError', 2);

% Create the webcam object.
cam = webcam('HP Wide Vision HD');

% Capture one frame to get its size.
videoFrame = snapshot(cam);
frameSize = size(videoFrame);

% Create the video player object.
videoPlayer = vision.VideoPlayer('Position', [100 100 [frameSize(2), frameSize(1)]+30]);

%% Detection and Tracking
% Capture and process video frames from the webcam in a loop to detect and
% track a face. The loop will run for 400 frames or until the video player
% window is closed.

runLoop = true;
numPts = 0;
frameCount = 0;
```

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

```
while runLoop && frameCount < 400

    % Get the next frame.
    videoFrame = snapshot(cam);
    videoFrameGray = rgb2gray(videoFrame);
    frameCount = frameCount + 1;

    if numPts < 10
        % Detection mode.
        bbox = faceDetector.step(videoFrameGray);

        if ~isempty(bbox)
            % Find corner points inside the detected region.
            points = detectMinEigenFeatures(videoFrameGray, 'ROI', bbox(1, :));

            % Re-initialise the point tracker.
            xyPoints = points.Location;
            numPts = size(xyPoints,1);
            release(pointTracker);
            initialise(pointTracker, xyPoints, videoFrameGray);

            % Save a copy of the points.
            oldPoints = xyPoints;

            % Convert the rectangle represented as [x, y, w, h] into an
            % M-by-2 matrix of [x,y] coordinates of the four corners. This
            % is needed to be able to transform the bounding box to display
            % the orientation of the face.
            bboxPoints = bbox2points(bbox(1, :));

            % Convert the box corners into the [x1 y1 x2 y2 x3 y3 x4 y4]
            % format required by insertShape.
            bboxPolygon = reshape(bboxPoints', 1, []);

            % Display a bounding box around the detected face.
            videoFrame = insertShape(videoFrame, 'Polygon', bboxPolygon, 'LineWidth', 3);

            % Display detected corners.
            videoFrame = insertMarker(videoFrame, xyPoints, '+', 'Color', 'white');
        end
```

UTeM
اونيۏرسيتي تيکنيکل مليسيا ملاک
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

```
    else
        % Tracking mode.
        [xyPoints, isFound] = step(pointTracker, videoFrameGray);
        visiblePoints = xyPoints(isFound, :);
        oldInliers = oldPoints(isFound, :);

        numPts = size(visiblePoints, 1);

        if numPts >= 10
            % Estimate the geometric transformation between the old points
            % and the new points.
            [xform, inlierIdx] = estimateGeometricTransform2D(...
                oldInliers, visiblePoints, 'similarity', 'MaxDistance', 4);
            oldInliers(inlierIdx, :);
            visiblePoints = visiblePoints(inlierIdx, :);

            % Apply the transformation to the bounding box.
            bboxPoints = transformPointsForward(xform, bboxPoints);

            % Convert the box corners into the [x1 y1 x2 y2 x3 y3 x4 y4]
            % format required by insertShape.
            bboxPolygon = reshape(bboxPoints', 1, []);

            % Display a bounding box around the face being tracked.
            videoFrame = insertShape(videoFrame, 'Polygon', bboxPolygon, 'LineWidth', 3);

            % Display tracked points.
            videoFrame = insertMarker(videoFrame, visiblePoints, '+', 'Color', 'white');

            % Reset the points.
            oldPoints = visiblePoints;
            setPoints(pointTracker, oldPoints);
        end

    end

    % Display the annotated video frame using the video player object.
    step(videoPlayer, videoFrame);

    % Check whether the video player window has been closed.
    runLoop = isOpen(videoPlayer);
end

% Clean up.
clear cam;
release(videoPlayer);
release(pointTracker);
release(faceDetector);
        end

        % Value changed function: ClawGripperSwitch
        function ClawGripperSwitchValueChanged(app, event)

        end

        % Button pushed function: ClosedButton
        function ClosedButtonPushed(app, event)
            app.delete
            set_param('simulinkprojekdone','SimulationCommand','stop')

        end

        % Value changing function: FowardBackwardContinuousServoSlider_3
        function FowardBackwardContinuousServoSlider_3ValueChanging(app, event)
            changingValue = event.Value;
            set_param([bdroot,'/Foward Backward'],'Value', num2str(changingValue));
        end

        % Value changing function: ServoMotor0180DegSlider_3
        function ServoMotor0180DegSlider_3ValueChanging(app, event)
            changingValue = event.Value;
            set_param([bdroot,'/RightLeft'],'Value', num2str(changingValue));
        end
    end
```

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

```
% Component initialization
methods (Access = private)

    % Create UIFigure and components
    function createComponents(app)

        % Create simulinkprojekdone and hide until all components are created
        app.simulinkprojekdone = uifigure('Visible', 'off');
        app.simulinkprojekdone.Position = [100 100 630 476];
        app.simulinkprojekdone.Name = 'MATLAB App';

        % Create TabGroup
        app.TabGroup = uitabgroup(app.simulinkprojekdone);
        app.TabGroup.TabLocation = 'left';
        app.TabGroup.Position = [1 35 631 442];

        % Create NavigatonTab_3
        app.NavigatonTab_3 = uitab(app.TabGroup);
        app.NavigatonTab_3.Title = 'Navigaton';

        % Create FowardBackwardContinuousServoSlider_3Label
        app.FowardBackwardContinuousServoSlider_3Label = uilabel(app.NavigatonTab_3);
        app.FowardBackwardContinuousServoSlider_3Label.HorizontalAlignment = 'right';
        app.FowardBackwardContinuousServoSlider_3Label.Position = [24 307 101 42];
        app.FowardBackwardContinuousServoSlider_3Label.Text = {'Foward '; 'Backward'; 'Continuous Servo'};

        % Create FowardBackwardContinuousServoSlider_3
        app.FowardBackwardContinuousServoSlider_3 = uislider(app.NavigatonTab_3);
        app.FowardBackwardContinuousServoSlider_3.Limits = [0 180];
        app.FowardBackwardContinuousServoSlider_3.MajorTicks = [];
        app.FowardBackwardContinuousServoSlider_3.MajorTickLabels = {'-100', '100'};
        app.FowardBackwardContinuousServoSlider_3.Orientation = 'vertical';
        app.FowardBackwardContinuousServoSlider_3.ValueChangingFcn = createCallbackFcn(app, @FowardBackwardContinuousServoSlider_3ValueChanging, true);
        app.FowardBackwardContinuousServoSlider_3.Position = [131 242 3 168];
        app.FowardBackwardContinuousServoSlider_3.Value = 90;

        % Create ServoMotor0180DegSlider_3Label
        app.ServoMotor0180DegSlider_3Label = uilabel(app.NavigatonTab_3);
        app.ServoMotor0180DegSlider_3Label.HorizontalAlignment = 'right';
        app.ServoMotor0180DegSlider_3Label.Position = [225 345 62 42];
        app.ServoMotor0180DegSlider_3Label.Text = {'Servo'; 'Motor'; '0-180 Deg'};

        % Create ServoMotor0180DegSlider_3
        app.ServoMotor0180DegSlider_3 = uislider(app.NavigatonTab_3);
        app.ServoMotor0180DegSlider_3.Limits = [0 180];
        app.ServoMotor0180DegSlider_3.MajorTicks = [];
        app.ServoMotor0180DegSlider_3.ValueChangingFcn = createCallbackFcn(app, @ServoMotor0180DegSlider_3ValueChanging, true);
        app.ServoMotor0180DegSlider_3.Position = [308 374 150 3];
        app.ServoMotor0180DegSlider_3.Value = 90;
```

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

```
% Create FowardLabel
app.FowardLabel = uilabel(app.NavigatonTab_3);
app.FowardLabel.Position = [157 399 46 22];
app.FowardLabel.Text = 'Foward';

% Create BackwardLabel
app.BackwardLabel = uilabel(app.NavigatonTab_3);
app.BackwardLabel.Position = [158 232 58 22];
app.BackwardLabel.Text = 'Backward';

% Create StopLabel
app.StopLabel = uilabel(app.NavigatonTab_3);
app.StopLabel.Position = [163 315 30 22];
app.StopLabel.Text = 'Stop';

% Create FlexConditionLabel
app.FlexConditionLabel = uilabel(app.NavigatonTab_3);
app.FlexConditionLabel.HorizontalAlignment = 'right';
app.FlexConditionLabel.Position = [58 159 82 22];
app.FlexConditionLabel.Text = 'Flex Condition';

% Create FlexconditionField
app.FlexconditionField = uieditfield(app.NavigatonTab_3, 'numeric');
app.FlexconditionField.Editable = 'off';
app.FlexconditionField.Position = [155 159 100 22];

% Create AcceYLabel_3
app.AcceYLabel_3 = uilabel(app.NavigatonTab_3);
app.AcceYLabel_3.HorizontalAlignment = 'right';
app.AcceYLabel_3.Position = [324 159 46 22];
app.AcceYLabel_3.Text = 'Voltage';

% Create VoltageEditField
app.VoltageEditField = uieditfield(app.NavigatonTab_3, 'numeric');
app.VoltageEditField.Editable = 'off';
app.VoltageEditField.Position = [385 159 100 22];

% Create RightLabel
app.RightLabel = uilabel(app.NavigatonTab_3);
app.RightLabel.Position = [457 331 34 22];
app.RightLabel.Text = 'Right';

% Create LeftLabel
app.LeftLabel = uilabel(app.NavigatonTab_3);
app.LeftLabel.Position = [291 331 26 22];
app.LeftLabel.Text = 'Left';
```

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

```
% Create NeutralLabel
app.NeutralLabel = uilabel(app.NavigatonTab_3);
app.NeutralLabel.Position = [366 324 44 22];
app.NeutralLabel.Text = 'Neutral';

% Create DeltaRobotArmTab
app.DeltaRobotArmTab = uitab(app.TabGroup);
app.DeltaRobotArmTab.Title = 'Delta Robot Arm ';

% Create AcceXEditFieldLabel
app.AcceXEditFieldLabel = uilabel(app.DeltaRobotArmTab);
app.AcceXEditFieldLabel.HorizontalAlignment = 'right';
app.AcceXEditFieldLabel.Position = [77 352 44 22];
app.AcceXEditFieldLabel.Text = 'Acce X';

% Create XAcceEditField
app.XAcceEditField = uieditfield(app.DeltaRobotArmTab, 'numeric');
app.XAcceEditField.Editable = 'off';
app.XAcceEditField.Position = [136 352 100 22];

% Create AcceYLabel
app.AcceYLabel = uilabel(app.DeltaRobotArmTab);
app.AcceYLabel.HorizontalAlignment = 'right';
app.AcceYLabel.Position = [78 322 43 22];
app.AcceYLabel.Text = 'Acce Y';

% Create YAcceEditField
app.YAcceEditField = uieditfield(app.DeltaRobotArmTab, 'numeric');
app.YAcceEditField.Editable = 'off';
app.YAcceEditField.Position = [136 322 100 22];

% Create AcceYLabel_2
app.AcceYLabel_2 = uilabel(app.DeltaRobotArmTab);
app.AcceYLabel_2.HorizontalAlignment = 'right';
app.AcceYLabel_2.Position = [78 290 43 22];
app.AcceYLabel_2.Text = 'Acce Z';

% Create ZAcceEditField
app.ZAcceEditField = uieditfield(app.DeltaRobotArmTab, 'numeric');
app.ZAcceEditField.Editable = 'off';
app.ZAcceEditField.Position = [136 290 100 22];

% Create GyroXEditFieldLabel
app.GyroXEditFieldLabel = uilabel(app.DeltaRobotArmTab);
app.GyroXEditFieldLabel.HorizontalAlignment = 'right';
app.GyroXEditFieldLabel.Position = [327 352 43 22];
app.GyroXEditFieldLabel.Text = 'Gyro X';
```

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

```
% Create GyroXEditField
app.GyroXEditField = uieditfield(app.DeltaRobotArmTab, 'numeric');
app.GyroXEditField.Editable = 'off';
app.GyroXEditField.Position = [385 352 100 22];

% Create GyroYEditFieldLabel
app.GyroYEditFieldLabel = uilabel(app.DeltaRobotArmTab);
app.GyroYEditFieldLabel.HorizontalAlignment = 'right';
app.GyroYEditFieldLabel.Position = [327 322 43 22];
app.GyroYEditFieldLabel.Text = 'Gyro Y';

% Create GyroYEditField
app.GyroYEditField = uieditfield(app.DeltaRobotArmTab, 'numeric');
app.GyroYEditField.Editable = 'off';
app.GyroYEditField.Position = [385 322 100 22];

% Create GyroZEditFieldLabel
app.GyroZEditFieldLabel = uilabel(app.DeltaRobotArmTab);
app.GyroZEditFieldLabel.HorizontalAlignment = 'right';
app.GyroZEditFieldLabel.Position = [328 290 42 22];
app.GyroZEditFieldLabel.Text = 'Gyro Z';

% Create GyroZEditField
app.GyroZEditField = uieditfield(app.DeltaRobotArmTab, 'numeric');
app.GyroZEditField.Editable = 'off';
app.GyroZEditField.Position = [385 290 100 22];

% Create Motor1EditFieldLabel
app.Motor1EditFieldLabel = uilabel(app.DeltaRobotArmTab);
app.Motor1EditFieldLabel.HorizontalAlignment = 'right';
app.Motor1EditFieldLabel.Position = [225 222 46 22];
app.Motor1EditFieldLabel.Text = 'Motor 1';

% Create Motor1EditField
app.Motor1EditField = uieditfield(app.DeltaRobotArmTab, 'numeric');
app.Motor1EditField.Editable = 'off';
app.Motor1EditField.Position = [286 222 100 22];

% Create Motor2EditFieldLabel
app.Motor2EditFieldLabel = uilabel(app.DeltaRobotArmTab);
app.Motor2EditFieldLabel.HorizontalAlignment = 'right';
app.Motor2EditFieldLabel.Position = [225 192 46 22];
app.Motor2EditFieldLabel.Text = 'Motor 2';

% Create Motor2EditField
app.Motor2EditField = uieditfield(app.DeltaRobotArmTab, 'numeric');
app.Motor2EditField.Editable = 'off';
app.Motor2EditField.Position = [286 192 100 22];

% Create Motor3EditFieldLabel
app.Motor3EditFieldLabel = uilabel(app.DeltaRobotArmTab);
```

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

```
            app.Motor3EditFieldLabel.HorizontalAlignment = 'right';
            app.Motor3EditFieldLabel.Position = [225 160 46 22];
            app.Motor3EditFieldLabel.Text = 'Motor 3';

            % Create Motor3EditField
            app.Motor3EditField = uieditfield(app.DeltaRobotArmTab, 'numeric');
            app.Motor3EditField.Editable = 'off';
            app.Motor3EditField.Position = [286 160 100 22];

            % Create GripperArmTab
            app.GripperArmTab = uitab(app.TabGroup);
            app.GripperArmTab.Title = 'Gripper Arm';

            % Create RotateGripKnobLabel
            app.RotateGripKnobLabel = uilabel(app.GripperArmTab);
            app.RotateGripKnobLabel.HorizontalAlignment = 'center';
            app.RotateGripKnobLabel.Position = [121 253 67 22];
            app.RotateGripKnobLabel.Text = 'Rotate Grip';

            % Create RotateGripKnob
            app.RotateGripKnob = uiknob(app.GripperArmTab, 'continuous');
            app.RotateGripKnob.Limits = [0 180];
            app.RotateGripKnob.ValueChangingFcn = createCallbackFcn(app, @RotateGripKnobValueChanging, true);
            app.RotateGripKnob.Position = [123 309 60 60];

            % Create RotateDegreeEditFieldLabel
            app.RotateDegreeEditFieldLabel = uilabel(app.GripperArmTab);
            app.RotateDegreeEditFieldLabel.HorizontalAlignment = 'right';
            app.RotateDegreeEditFieldLabel.Position = [306 329 45 28];
            app.RotateDegreeEditFieldLabel.Text = {'Rotate '; 'Degree'};

            % Create RotateDegreeEditField
            app.RotateDegreeEditField = uieditfield(app.GripperArmTab, 'numeric');
            app.RotateDegreeEditField.Limits = [0 180];
            app.RotateDegreeEditField.ValueChangedFcn = createCallbackFcn(app, @RotateDegreeEditFieldValueChanged, true);
            app.RotateDegreeEditField.Position = [366 335 100 22];

            % Create ClawGripperSwitchLabel
            app.ClawGripperSwitchLabel = uilabel(app.GripperArmTab);
            app.ClawGripperSwitchLabel.HorizontalAlignment = 'center';
            app.ClawGripperSwitchLabel.Position = [237 92 76 22];
            app.ClawGripperSwitchLabel.Text = 'Claw Gripper';

            % Create ClawGripperSwitch
            app.ClawGripperSwitch = uiswitch(app.GripperArmTab, 'rocker');
            app.ClawGripperSwitch.Items = {'Closed', 'Open'};
            app.ClawGripperSwitch.ValueChangedFcn = createCallbackFcn(app, @ClawGripperSwitchValueChanged, true);
            app.ClawGripperSwitch.Position = [264 150 20 45];
            app.ClawGripperSwitch.Value = 'Closed';
            % Create RobotVisionTab
            app.RobotVisionTab = uitab(app.TabGroup);
            app.RobotVisionTab.Title = 'Robot Vision ';

            % Create UIAxes
            app.UIAxes = uiaxes(app.RobotVisionTab);
            title(app.UIAxes, 'Camera')
            xlabel(app.UIAxes, 'X')
            ylabel(app.UIAxes, 'Y')
            zlabel(app.UIAxes, 'Z')
            app.UIAxes.Position = [59 128 394 272];

            % Create StartVisionCameraButton
            app.StartVisionCameraButton = uibutton(app.RobotVisionTab, 'push');
            app.StartVisionCameraButton.ButtonPushedFcn = createCallbackFcn(app, @StartVisionCameraButtonPushed, true);
            app.StartVisionCameraButton.Position = [204 46 100 36];
            app.StartVisionCameraButton.Text = {'Start Vision'; 'Camera'};

            % Create ClosedButton
            app.ClosedButton = uibutton(app.simulinkprojekdone, 'push');
            app.ClosedButton.ButtonPushedFcn = createCallbackFcn(app, @ClosedButtonPushed, true);
            app.ClosedButton.Position = [306 14 100 22];
            app.ClosedButton.Text = 'Closed';

            % Show the figure after all components are created
            app.simulinkprojekdone.Visible = 'on';
        end
    end
```

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

```
    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = Gui

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.simulinkprojekdone)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.simulinkprojekdone)
        end
    end
end
```

Appendix C Gantt Chart of Project Planning

A. PERANCANGAN PROJEK
PROJECT PLANNING (GANTT CHART)

Senaraikan aktiviti-aktiviti yang berkaitan bagi projek yang dicadangkan dan nyatakan jangka masa yang diperlukan bagi setiap aktiviti.
List all the relevant activities of the proposed project and mark the period of time that is needed for each of the activities.

	SEM I																			SEM BREAK					SEM II															
Aktiviti Projek Project Activities	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
PSM 1 Title Registration	X																																							
Background Study / Literature Review		X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X																	
Proposal Submission and Defense				X																																				
Evaluate Proposed Project					X	X	X	X	X	X		X	X																												
Analyze and broken down a Cili-Padi Picking robot framework					X						SEMINAR PSM I																												SEMINAR PSM II		
Develop an embedded software					X	X	X	X	X	X																														
Apply FreeRTOS												X	X																												
GUI design													X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X													
Interface GUI with simulink																												X	X	X	X	X								
Fully testing and updating on GUI running.																																X	X	X	X					
Documentation					X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X				