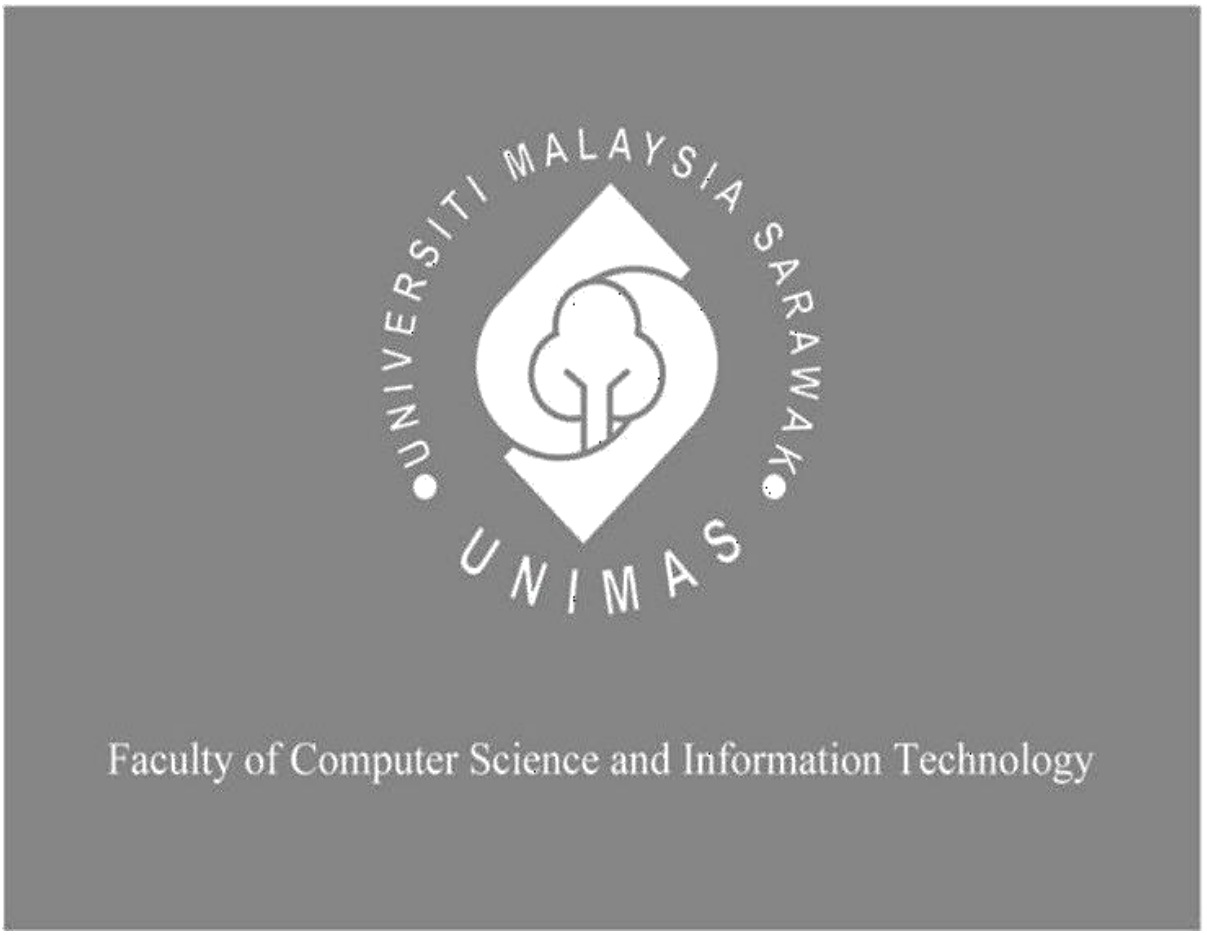



# Gesture-Controlled Robotic Arm for Kids

**MUHAMMAD ZIKRY BIN MOHD ZAMBRI (78277)**

**Bachelor of Computer Science with Honour's**

**(Software Engineering)**

Faculty of Computer Science and Information Technology UNIVERSITI MALAYSIA SARAWAK
2025

**UNIVERSITI MALAYSIA SARAWAK**

**THESIS STATUS ENDORSEMENT FORM**

**TITLE** **Gesture-Controlled Robotic Arm for Kids**

**ACADEMIC SESSION: 2024/2025**

MUHAMMAD ZIKRY BIN MOHD ZAMBRI

**(CAPITAL LETTERS)**

hereby agree that this Thesis* shall be kept at the Centre for Academic Information Services, Universiti Malaysia Sarawak, subject to the following terms and conditions:

1. The Thesis is solely owned by Universiti Malaysia Sarawak
2. The Centre for Academic Information Services is given full rights to produce copies for educational purposes only
3. The Centre for Academic Information Services is given full rights to do digitization in order to develop local content database
4. The Centre for Academic Information Services is given full rights to produce copies of this Thesis as part of its exchange item program between Higher Learning Institutions [ or for the purpose of interlibrary loan between HLI ]
5. ** Please tick ( √ )

| | | |
|---|---|---|
| ☐ | CONFIDENTIAL | (Contains classified information bounded by the OFFICIAL SECRETS ACT 1972) |
| ☐ | RESTRICTED | (Contains restricted information as dictated by the body or organization where the research was conducted) |
| ☑ | UNRESTRICTED | |

Validated by

(AUTHOR'S SIGNATURE) (SUPERVISOR'S SIGNATURE)

Permanent Address

Date: 17/01/2025 Date:

Note * Thesis refers to PhD, Master, and Bachelor Degree
** For Confidential or Restricted materials, please attach relevant documents from relevant organizations / authorities

## DECLARATION

I hereby declare that this project and its contents are my original work except for that information which have been cited and quoted are extracted from other sources with the provided references.

(MUHAMMAD ZIKRY BIN MOHD ZAMBRI) 17th JANUARY 2025

# ACKNOWLEDGEMENT

In this project, I would like to express my gratitude to Faculty of Computer Science and Information Technology, and special thanks to my supervisor, Dr. Lau, who gave me the golden opportunity to do this wonderful project as my Final Year Project on the topic - The Development of Gesture-Controlled Robotic Arm for Kids. I am sincerely delivering my appreciation to Dr. Lau for his truthful advice, patience, knowledge, guidance, and support to complete this project on time with high productivity.

Apart from that, I also would like to acknowledge, with much appreciation, the crucial roles of all my lecturers throughout my studies since my first year, who had contributed to teaching me with all the skill set and knowledge for me to be able to execute this project.

Finally, I would also like to thank my family and friends who helped me a lot, provided me with spiritual support and guidance through the journey to complete this Final Year Project of mine.

**ABSTRACT**

STEM education fosters critical thinking and problem-solving, but traditional methods often fail to engage young learners. Many educational robotics kits are too complex for children, relying on joysticks or programming interfaces. This project, *Gesture-Controlled Robotic Arm for Kids*, introduces a user-friendly robotic arm operated through simple hand gestures, making STEM learning more accessible and interactive. Using the Agile Software Development Life Cycle, the system integrates a gesture detection module, a robotic arm, and control interface. Hand gestures captured via a smartphone camera are processed in real time and translated into movements using an Arduino and servomotors. The design is affordable, portable, and ideal for schools, workshops, or home use. This project enhances children's cognitive and motor skills while introducing them to technologies like embedded systems and computer vision. By bridging theory and hands-on experience, it inspires curiosity and innovation in robotics.

**ABSTRAK**

Pendidikan STEM menggalakkan pemikiran kritikal dan penyelesaian masalah, tetapi kaedah tradisional sering gagal untuk menarik minat pelajar muda. Banyak kit robotik pendidikan terlalu kompleks untuk kanak-kanak, bergantung kepada joystick atau antaramuka pengaturcaraan. Projek ini, "*Gesture-Controlled Robotic Arm for Kids"*, memperkenalkan lengan robot yang mesra pengguna yang dikendalikan melalui isyarat tangan yang mudah, menjadikan pembelajaran STEM lebih mudah diakses dan interaktif. Menggunakan Kitaran Hayat Pembangunan Perisian Agile, sistem ini mengintegrasikan modul pengesanan isyarat, lengan robot, dan antaramuka kawalan. Isyarat tangan yang dirakam melalui kamera telefon pintar diproses dalam masa nyata dan ditransformasikan menjadi gerakan menggunakan Arduino dan servomotor. Rekabentuknya yang mampu milik, mudah alih, dan ideal untuk sekolah, bengkel, atau penggunaan di rumah. Projek ini meningkatkan kemahiran kognitif dan motor kanak-kanak sambil memperkenalkan mereka kepada teknologi seperti sistem terbenam dan penglihatan komputer.Dengan menghubungkan teori dan pengalaman praktikal, ia menggalakkan rasa ingin tahu dan inovasi dalam robotik.

.

Table of Contents

**List of Tables**

**List of Figures**

# Chapter 1

## 1.1. Background/Introduction

Robotics has emerged as a key component of educational innovation in recent years, offering kids engaging chances to investigate STEM (Science, Technology, Engineering, and Mathematics) ideas through interactive instruction. (Huda, 2023) A decline in STEM enrolment could hinder the nation's progress in these areas. Among these developments, gesture-controlled systems have become popular instruments that facilitate natural human-machine interaction. The goal of this project, "Gesture-Controlled Robotic Arm for Kids," is to create a robotic arm that reacts to hand movements in order to improve kids' hands-on experience and comprehension of robotics and human-computer interaction.

By utilizing gesture-based controls, the suggested robotic arm makes operation more convenient and appealing for younger users. Children can control the robotic arm naturally to carry out a variety of tasks, including moving objects, picking up objects, and sketching shapes, by using sensor technologies to detect and interpret hand gestures. In August 2023, Vidyakar Children gain a lot from robotics because it fosters their ability to be creative, solve problems, and think critically.

The development of such a system offers multiple educational benefits, including exposure to coding, electronics, and physics, while promoting motor skills and cognitive development through interactive play. This project aims to create a user-friendly and safe environment that encourages children to learn through exploration, ultimately making learning fun and inspiring the next generation of young innovators.

## 1.2. Problem Statement

1. Traditional teaching methods often do not capture children's interest in STEM subjects. Without hands-on, interactive experiences, many kids lose interest in science and technology early on.
2. Many educational robotics kits are too complex for younger children, relying on difficult controls like joysticks or programming, which can make it harder for them to learn and enjoy robotics.
3. Children have limited opportunities to experience modern technologies, like gesture control, which are becoming more common in today’s world. Early exposure to these technologies can better prepare them for the future.
4. There is a custom Arduino robotic arm in the maker space that uses a joystick for control. This project seeks an alternative remote control method for the robotic arm, making it easier and more accessible for children to operate.

## 1.3. Objectives

1. To develop a gesture recognition system using a smartphone camera to detect and interpret hand movements.
2. To map the recognized gestures from the smartphone camera to specific commands for controlling the Arduino robotic arm movement
3. To integrate smartphone-based gesture recognition with Arduino robotic arm via a mobile phone application.

### 1.4. Procedure/Methodologies

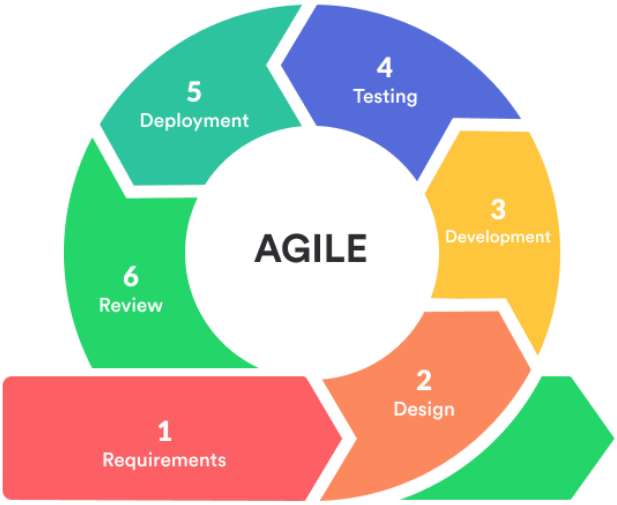



Figure 1: Agile SDLC. (Adapted from Jayathilaka, 2020)

For the development of the robotics, we will be using Arduino by following the Agile Software Development Life Cycle (SDLC) Model as such as figure 1 where it includes 6 main phases: Requirement Analysis, Design, Development, Testing, Deployment and Review. The Agile Model itself implies adaptability. This is because we can make adjustments when we use this model to implement our system because some of its requirements may change as it is being developed. Additionally, it works well for small projects with small teams. Unknown risks could arise at any point while our project is being developed, and they could have a significant impact. With options, they can postpone crucial decisions until better or more data, or even whole hosting programs, become available. This way, the project can proceed without worrying about coming to a sudden halt (Jayathilaka, 2020). The first phase is dedicated to requirements gathering and analysis, in which all-critical information is collected and analysed. It is intended to resolve all project-related ambiguities. The next phase is Design. This is where the design process for the robotic arm begins. This project will design the robotic arm modularly to ensure flexibility, scalability, and ease of use. The design phase entails creating detailed schematics for hardware and software components. The hardware design focuses on developing the mechanical structure, using lightweight and child-friendly materials, and ensuring that all moving parts operate safely. For the design, we will develop a control system that combines gesture recognition input with robotic arm movements. This includes creating algorithms for

gesture processing, mapping gestures to specific arm actions (e.g., gripping, rotating), and designing a user-friendly interface if necessary. The Development Phase follows, during which the hardware and software components are built and assembled. The Testing Phase entails meticulous verification to ensure that each component and the system as a whole meet the necessary functionality, performance, and safety requirements. Deployment occurs after successful testing. This phase focuses on making the gesture-controlled robotic arm available to its intended users, which include educators, parents, and children. Documentation, user guides, and tutorials will be created to make it easier to use and more educationally valuable. The final Review Phase will be used to assess the project's overall success and effectiveness. User feedback will be collected to help identify areas for improvement. This review may result in additional iterations to refine the system further, ensuring that the project is constantly adapting to user needs and technological advancements.

### 1.5. Scope

The project focuses on increasing children's interest and engagement in STEM (Science, Technology, Engineering, and Mathematics) by providing an intuitive and interactive learning tool. The aim is to create a robotic arm that children can control using simple hand gestures, making robotics and technology fun and accessible. By eliminating complex programming or controls, the project ensures that even young learners can easily understand and use the system. Through hands-on interaction, children will gain exposure to modern technologies like gesture recognition, develop critical skills such as problem solving and creativity, and enhance their fine motor coordination. Designed with child safety and accessibility in mind, the robotic arm will be suitable for use in schools, workshops, and home learning environments, supported by user guides and educational materials to maximize its impact. This project seeks to bridge the gap between theoretical STEM learning and real-world applications, sparking curiosity and inspiring the next generation of innovators.

## 1.6. Significance of the project

This project is significant because it helps spark children's interest in STEM (Science, Technology, Engineering, and Mathematics) by making learning fun and hands-on. By using simple hand gestures to control a robotic arm, kids can easily understand and explore robotics, motion, and technology. This project introduces children to modern technologies like gesture recognition and robotics, giving them valuable skills and preparing them for a tech-driven future.

## 1.7. Project Schedule

| Name | Duration | Start | Finish |
|---|---|---|---|
| **Final Year Project Sched** | 6 days | **10/18/24 8:00 AM** | **10/25/24 5:00 PM** |
| Determine Supervisor and F | 6 days | 10/18/24 8:00 AM | 10/25/24 5:00 PM |
| | | | |
| **Requirement Analysis** | 20 days? | **10/25/24 8:00 AM** | **11/21/24 5:00 PM** |
| Project Working Progress | 7 days | 10/25/24 8:00 AM | 11/4/24 5:00 PM |
| Determine the scope, objec | 6 days | 11/4/24 8:00 AM | 11/11/24 5:00 PM |
| Determine Hardware Requi | 3 days | 11/11/24 8:00 AM | 11/13/24 5:00 PM |
| Determine Software Requir | 3 days? | 11/13/24 8:00 AM | 11/15/24 5:00 PM |
| Chapter 1 Submission | 5 days? | 11/15/24 8:00 AM | 11/21/24 5:00 PM |
| | | | |
| | | | |
| **Chapter 2: Literature Rev** | 27 days | **11/21/24 8:00 AM** | **12/27/24 5:00 PM** |
| Design Electrical Circuit Diag | 10 days | 11/21/24 8:00 AM | 12/4/24 5:00 PM |
| System Design | 10 days | 12/4/24 8:00 AM | 12/17/24 5:00 PM |
| Design App interface | 5 days | 12/17/24 8:00 AM | 12/23/24 5:00 PM |
| chapter 2 Submission | 5 days | 12/23/24 8:00 AM | 12/27/24 5:00 PM |
| | | | |
| **Chapter 3: Requirement .** | 16 days? | **12/27/24 8:00 AM** | **1/17/25 5:00 PM** |
| Design System and Impleme | 10 days | 12/27/24 8:00 AM | 1/9/25 5:00 PM |
| Draw prototype and circuit | 5 days | 1/9/25 8:00 AM | 1/15/25 5:00 PM |
| Submission FYP 1 | 3 days? | 1/15/25 8:00 AM | 1/17/25 5:00 PM |
| | | | |
| **Chapter 4: Development** | 13 days | **4/10/25 8:00 AM** | **4/28/25 5:00 PM** |
| Prepare test result | 5 days | 4/10/25 8:00 AM | 4/16/25 5:00 PM |
| Build the Hardware (Electric | 5 days | 4/16/25 8:00 AM | 4/22/25 5:00 PM |
| Build App Interface | 5 days | 4/22/25 8:00 AM | 4/28/25 5:00 PM |
| Alpha Testing | 5 days | 4/10/25 8:00 AM | 4/16/25 5:00 PM |
| Connect the hardware with | 5 days | 4/16/25 8:00 AM | 4/22/25 5:00 PM |
| Chapter 4 Submission | 5 days | 4/22/25 8:00 AM | 4/28/25 5:00 PM |
| | | | |
| **Chapter 5: Testing and Ev** | 14 days | **4/28/25 8:00 AM** | **5/15/25 5:00 PM** |
| Build the system | 10 days | 4/28/25 8:00 AM | 5/9/25 5:00 PM |
| Define Project Achievemen | 5 days | 5/9/25 8:00 AM | 5/15/25 5:00 PM |
| | | | |
| **Chapter 6: Conclusion an** | 10 days | **8/25/25 8:00 AM** | **9/5/25 5:00 PM** |
| Submission of First Draft fo | 10 days | 8/25/25 8:00 AM | 9/5/25 5:00 PM |
| | | | |

Figure 2: Gant Chart Table

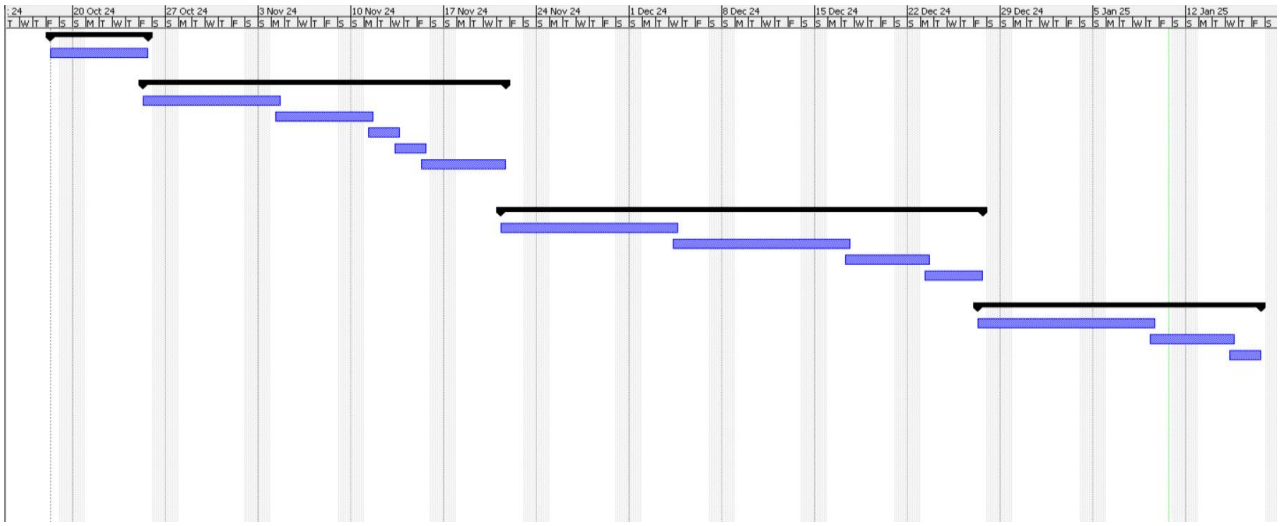



Figure 3: Gant Chart Graph

## 1.8. Expected Outcome

The "Gesture-Controlled Robotic Arm for Kids" project aims to achieve a range of significant outcomes that enhance children's educational experiences and foster deeper engagement in STEM (Science, Technology, Engineering, and Mathematics). By interacting with the robotic arm, children will develop a greater interest in STEM disciplines through hands-on, interactive learning. The project will produce a user-friendly robotic arm that can be intuitively controlled using hand gestures, eliminating complex controls and making robotics accessible to young users. Additionally, children will gain early exposure to advanced technologies such as robotics and gesture recognition, equipping them with practical knowledge that reflects today's technological landscape.

Through experiential play with the robotic arm, children will learn essential concepts related to robotics, motion control, and sensors while improving their problem-solving, logical reasoning, and creativity. Gesture-based control will further help enhance fine motor skills, hand-eye coordination, and cognitive development. The project will also serve as a comprehensive educational resource, complete with user manuals, training materials, and guides, making it easy for parents, teachers, and facilitators to integrate into educational settings. Furthermore, the robotic arm will demonstrate real-world applications of gesture recognition and robotics, helping children understand and appreciate the relevance of these technologies in everyday life. Overall, the expected outcome is an engaging, educational, and user-friendly gesture-controlled robotic arm that inspires children to learn, explore, and innovate within the STEM field, preparing them for future technological challenges and opportunities.

# Chapter 2

## 2.1 Overview

In this chapter, we will review three similar studies from previous research and the tools we plan to use to develop our system.

## 2.2 Background

Gestures, defined as deliberate and expressive bodily motions involving the hands, fingers, face, arms, body, or head, are a prominent form of non-verbal communication in human interaction. Among these, hand gestures are particularly significant as they offer an alternative mode of communication. The ability of machines to comprehend human gestures enables a more natural and intuitive means of interaction. In the early stages of machine development, manual interfaces such as buttons and joysticks were commonly used to control a machine's circuitry and transmit commands through mechanical transmissions (A. Osman Hashi et al., 2013).

## 2.3 Review of Existing Solutions

### 2.3.1 Controlling a Robot Using Leap Motion

Controlling a robot using Leap Motion involves utilizing the Leap Motion Controller to interpret hand and finger movements as commands for the robot. Leap Motion is a highly sensitive motion-tracking device that detects gestures, enabling intuitive and precise control over robotic systems.

Figure 4: SCARA robotic arm

In the study "*Controlling a Robot Using Leap Motion*" by Chen et al. (2017), the authors explore the Leap Motion controller which detects gestures and hand movement, and then the data are delivered through USB to a SCARA robotic arm. This innovative approach emphasizes simplifying robot control mechanisms and making them more intuitive, bridging the gap between technology and usability. Traditional robotic systems often require intricate programming or physical controllers, which can pose challenges for users without technical expertise. As detailed in the study, Gesture-based control introduces a more accessible interface by leveraging the Leap Motion controller to capture and interpret real-time hand movements (Guna et al., 2014). The controller's precision, coupled with the intuitive nature of gestures, facilitates a seamless interaction process, making robot operation efficient and user-friendly. The primary objective of this research was to develop a model where users could teach robots tasks using predefined gestures. By employing a K-Nearest Neighbor (KNN) algorithm for gesture recognition, the system processes hand and finger movements to command the robotic

arm in real time. For example, an open palm signifies an object placement, while a fist halts motion, ensuring a robust and versatile system for various tasks (Casiez et al., 2012).

The robotic system consists of two main modules: the Leap Motion-based control module and the SCARA robotic arm as the controlled module. The researchers conducted experiments with novice operators performing pick-and-place operations to validate its effectiveness. The results highlighted a 100% success rate, with an average positional error of less than 1.5mm and task completion times averaging 72.38 seconds, demonstrating the system's reliability and ease of use (Sathiyanarayanan et al., 2014). However, limitations such as a 0.5-second latency and sensitivity to environmental factors require attention to improve precision and performance in dynamic settings. Enhancing gesture recognition algorithms with advanced machine learning models, expanding multi-robot capabilities, and improving hardware precision are key areas for future exploration. These advancements could further broaden the applicability of gesture-based control in fields like healthcare, education, and remote assistance (Brogårdh, 2007).

By integrating Leap Motion technology with robotics, this research underscores the transformative potential of gesture-based systems in advancing natural user interfaces. The findings demonstrate that such systems simplify robot operations and pave the way for innovations in HCI, fostering more intuitive, responsive, and accessible interactions between humans and machines (Guna et al., 2014).

### 2.3.2 Accelerometer and Gyroscope-Based Wearable

Wearable systems have gained prominence in gesture-controlled robotics due to their adaptability, compact design, and real-time interaction capabilities. These systems utilize Micro-Electro-Mechanical System (MEMS) sensors, including accelerometers, gyroscopes, and magnetometers, to detect and interpret human gestures. In the study *"Hand Motion Controlled Robotic Arm Based on MEMS Sensors"* by Mohamad et al. (2017), the authors explore the architecture, sensor fusion techniques, and practical applications of these systems, particularly in child-friendly robotic interfaces.

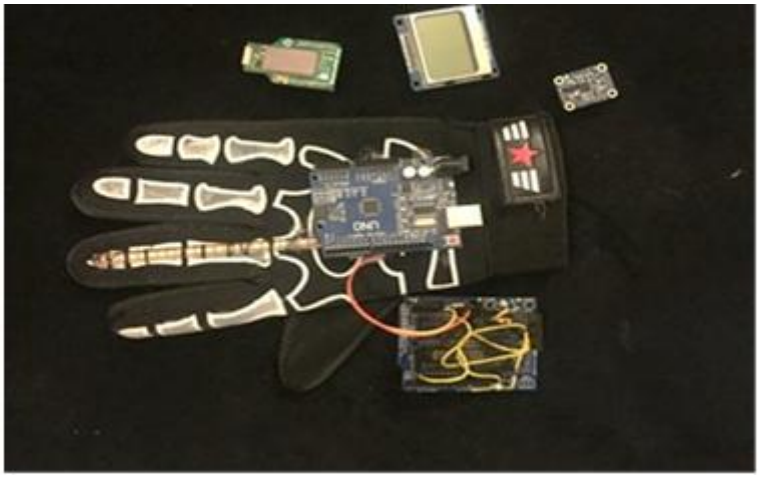


Figure 5: Robotic Arm Based on MEMS Sensors

The proposed system features a dual-component architecture comprising a sender side and a receiver side. The sender captures hand movements using sensors and transmits the data wirelessly via Bluetooth. The receiver decodes the transmitted data and commands a robotic arm to replicate the gestures. Key components include accelerometers for measuring tilt and linear acceleration, gyroscopes for tracking angular velocity, and magnetometers for orientation detection. Additionally, flex resistors enhance the system's ability to capture finger movements, improving the robotic arm's dexterity. The sender uses an Arduino Uno to process sensor data, while the receiver employs an Arduino Mega to control servo motors for precise arm movements. Bluetooth facilitates wireless communication between the sender and receiver, while an LCD monitors system status and movement angles.

The system workflow involves capturing real-time hand motion data with sensors, integrating the data using a complementary filter to reduce noise and enhance accuracy, and transmitting the processed information via Bluetooth. The receiver interprets this data and commands the robotic arm to replicate the user's gestures. Sensor fusion is crucial in ensuring accuracy by combining accelerometer and gyroscope data to estimate roll, pitch, and yaw angles. While accelerometers provide tilt information, they are sensitive to noise, whereas gyroscopes offer smooth rotational data but are prone to drift. The complementary filter balances these limitations, and magnetometers are integrated to correct yaw errors, although careful calibration is required to mitigate interference from external magnetic fields.

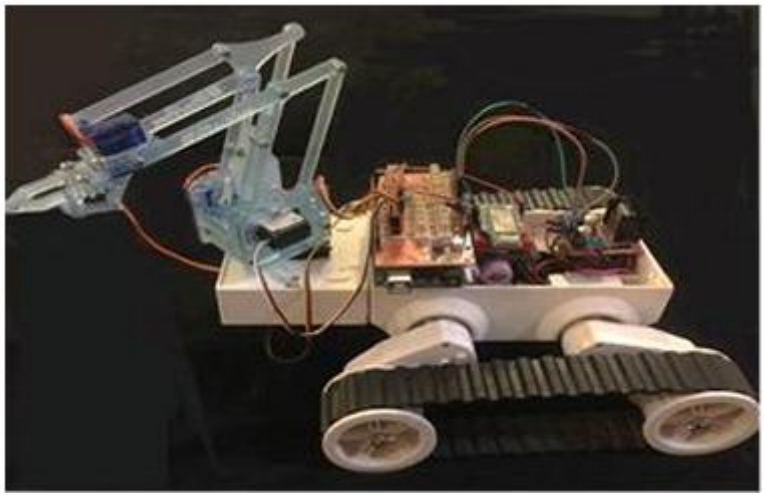


Figure 6: Robot Model Based on MEMS Sensors

Practical implementation involves hardware such as hand-mounted sensors on a glove equipped with MEMS components and a robotic arm mounted on a rover platform. The system is programmed using Arduino IDE, enabling functionalities like data acquisition, noise reduction, angle calculation through sensor fusion, data transmission, and real-time robotic arm control. Performance tests revealed high accuracy in tracking hand movements, low latency for real-time responsiveness, and ease of use, requiring minimal training for intuitive operation.

### 2.3.3 Microsoft Kinect for Gesture Recognition


Figure 7: System Architecture for Gesture Control of a Mobile Robot using Kinect Sensor

The Microsoft Kinect sensor, originally designed for gaming, has emerged as a valuable tool in gesture recognition and robotics. The paper *"Gesture Control of a Mobile Robot using Kinect Sensor"* by Cekova et al. (2016) explores its potential to enhance human-robot interaction.

Kinect-based gesture control systems rely on several interconnected components. The Kinect sensor captures user gestures by tracking the skeletal structure of the human body in 3D space. This data is processed by a computer, which interprets the gestures into corresponding commands. These commands are then sent via Bluetooth to an Arduino Uno controller, which translates them into motor instructions for a mobile robot. This system architecture ensures seamless communication between hardware and software. The Kinect sensor transmits skeleton data, which is processed through algorithms to identify gestures. Recognized gestures are mapped to predefined commands, enabling intuitive robot control.

The Kinect sensor offers impressive features such as depth perception, skeletal tracking of 25 joints, 3D motion capture, a wide field of view, and an effective range of 0.7 to 6 meters. These features make it suitable for small-scale robotics applications. Gesture recognition is based on predefined gestures, such as moving elbows forward to command "Move Forward" or raising one elbow to trigger a "Turn" action. The system processes gestures in under 150

milliseconds, ensuring the responsiveness required for real-time interaction. Communication and control are facilitated through Bluetooth connectivity, with commands transmitted to a mobile robot featuring differential drive wheels powered by DC motors. The Arduino microcontroller interprets these commands to adjust motor speeds.

The robot’s movement is governed by its kinematic equations. Using differential drive, the motion is defined by the linear and angular velocities, wheel velocities, and the distance between the wheels. These equations determine the robot's navigation and position over time, ensuring precise movement.

The Kinect sensor-based system boasts several advantages. Its high accuracy in tracking multiple joints provides detailed gesture recognition, while its ease of use allows natural interactions with minimal learning curve. The system can track up to six skeletons simultaneously, enabling multi-user scenarios. Additionally, real-time performance ensures rapid gesture recognition and command execution. However, there are limitations, such as environmental constraints like lighting and reflective surfaces affecting performance, reduced accuracy at extreme ranges, and reliance on hardware like computers and Bluetooth communication, which can introduce latency. Predefined gesture sets also limit customization.

Applications of Kinect sensor-based systems are diverse. They have been used in rehabilitation to assist patients with movement therapy, in education for interactive learning, and in industrial automation for contactless machinery operation. Experimental evaluations by Cekova et al. demonstrated the system's effectiveness, with over 90% recognition accuracy for gestures, 100% execution success for most commands, and a command latency of 50-150 milliseconds, making it suitable for real-time applications.

In conclusion, the Microsoft Kinect sensor has proven to be a powerful tool in robotics, enabling intuitive, accurate, and real-time gesture-based control, despite some challenges. Its versatility and effectiveness make it a valuable asset in various fields.

### 2.3.4 Comparison Table Between Existing and Proposed Systems

Table 1: Comparison Existing and Proposed System

| **Feature** | **Controlling a Robot Using Leap Motion** | **Accelerometer & Gyroscope-Based Wearables** | **Microsoft Kinect-Based System** | **Proposed Gesture-Controlled Robotics Arm for Kids** |
|---|---|---|---|---|
| **Tracking Technology** | Infrared Sensors | MEMS Sensors (Accelerometer, Gyroscope) | Depth Camera with Infrared Projection | Vision-Based (Webcam or Smartphone Camera) |
| **Accuracy** | High for finger-level precision | High for dynamic gestures | High for full-body gestures | Moderate (optimized for simple hand gestures) |
| **Cost** | Expensive | Moderate | High | Low (child-friendly and affordable hardware) |
| **Portability** | Compact but requires connection to PC | Wearable | Bulky and requires a PC connection | Highly portable and standalone |
| **Environmental Sensitivity** | Affected by ambient lighting | Independent of lighting conditions | Affected by lighting and reflective surfaces | Robust in varied conditions |
| **Ease of Use** | Requires familiarization | Simple but may require calibration | Intuitive but limited by predefined gestures | Designed for intuitive child interaction |
| **Setup Complexity** | Moderate | Low to Moderate | High | Low |
| **User Comfort** | High (non-contact) | Moderate (wearables can cause fatigue) | High (non-contact) | High (non-contact and child-friendly) |
| **Applications** | Precision tasks, gaming | Medical, industrial, and educational | Rehabilitation, industrial automation | Education and STEM learning |

### 2.3.5 Analysis

The comparison table highlights various systems' unique strengths and limitations, offering insights into their performance across different scenarios. This detailed analysis evaluates tracking technology, accuracy, cost, environmental sensitivity, ease of use, and applications, focusing on their implications for a child-friendly gesture-controlled robotic arm.

Regarding tracking technology, Leap Motion employs infrared sensors for highly precise finger-level gestures. Still, it is limited by its reliance on line-of-sight and sensitivity to ambient lighting. MEMS-based wearables use accelerometers and gyroscopes, excelling in dynamic gesture tracking but lacking fine-grained accuracy. Microsoft Kinect provides full-body gesture tracking using depth cameras, ideal for large-scale gestures but less suitable for compact systems. The proposed system leverages vision-based tracking optimized for simplicity, focusing on basic hand gestures to align with children's cognitive and motor skills, making it robust in diverse environments.

Regarding accuracy, both Leap Motion and Kinect offer high precision, though their complexity might be excessive for child-oriented systems. MEMS-based wearables provide reliable results for dynamic movements but often require calibration to maintain consistency. The proposed system balances simplicity and accuracy, ensuring effective gesture recognition without demanding excessive precision, which is ideal for young users.

In terms of cost and portability, Leap Motion and Kinect are expensive and rely on bulky hardware connected to PCs, limiting portability. MEMS-based wearables are moderately priced and reasonably portable but may feel cumbersome due to the wearable equipment. The proposed system stands out with its low cost and high portability, making it well-suited for educational purposes where affordability and ease of deployment are essential.

Environmental sensitivity is another key factor. Leap Motion and Kinect are highly susceptible to lighting conditions, affecting performance in varied settings. MEMS-based wearables operate independently of lighting but may encounter interference from magnetic fields. The proposed system demonstrates resilience in diverse conditions, ensuring consistent performance even in less controlled environments like classrooms or homes.

Ease of use and user comfort are critical considerations for child-friendly systems. Leap Motion and Kinect, while intuitive, require some familiarization, which can be challenging for younger users. MEMS-based wearables may cause discomfort during prolonged use due to

their wearable nature. The proposed system prioritizes non-contact interaction and intuitive design, enabling children to engage comfortably and without frustration.

Finally, applications vary across systems. Leap Motion is well-suited for precision tasks like gaming and virtual reality but lacks applicability in educational robotics. MEMS-based wearables have diverse applications in medical, industrial, and educational contexts, though their child-specific usability is limited. Kinect's full-body tracking excels in rehabilitation and industrial automation but is less relevant for compact, child-oriented systems. The proposed system aligns seamlessly with STEM education, offering an engaging platform for children to learn robotics and programming concepts in a practical and accessible manner.

## 2.4 Tools for Proposed System

The proposed system aims to be simple, portable, and cost-effective while being engaging and educational for children

### Hardware Component

### 2.4.1 Camera

Figure 8: Camera

The camera serves as the primary input device for capturing real-time hand gestures that guide the operation of the robotic arm. This can be a standard webcam connected to a computer or the built-in camera of a smartphone, chosen for its affordability and widespread availability. The camera streams live video, which is processed to recognize specific hand gestures or positions using computer vision or machine learning techniques. By detecting gestures, the system can translate them into commands for the robotic arm, such as moving to a specific location or picking up an object. A webcam or smartphone camera offers the advantage of high resolution and portability, making it suitable for both prototype development and field applications. Moreover, the use of a standard camera helps in maintaining cost efficiency while ensuring reliable gesture recognition in real-time.

### 2.4.2 Microcontroller

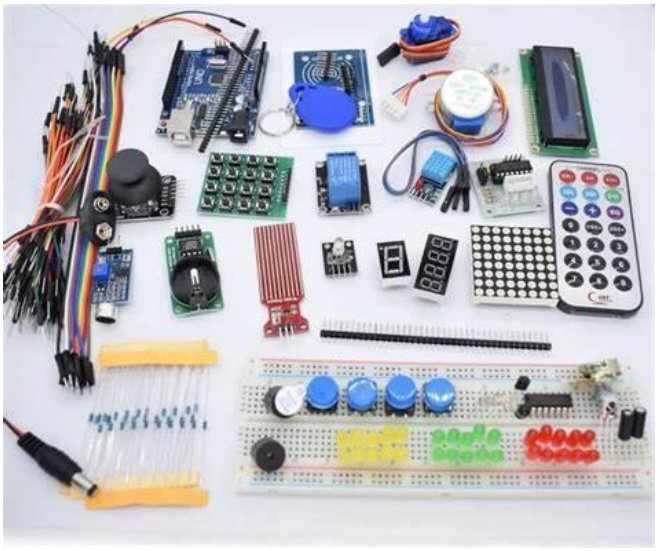


Figure 9: Arduino Set

The microcontroller is the brain of the robotic system, responsible for processing input signals and controlling the robotic arm's movements. An Arduino board is often used for this purpose due to its cost-effectiveness, ease of programming, and extensive community support. Similar to any microcontroller, an Arduino is a circuit board with a chip that can be configured to perform a wide range of tasks. To carry out a specific command, information is sent from the computer program to the Arduino microcontroller and then to the particular circuit or machine with multiple circuits (Badamasi, 2014). The Arduino's versatility and compatibility with various sensors and actuators make it ideal for managing tasks such as motor control, data communication, and input processing. For the robotic arm, the Arduino reads gesture-based commands from the computer or smartphone and translates them into precise motor actions. Its GPIO (General Purpose Input/Output) pins can directly interface with servo motors, enabling the microcontroller to manage joint angles and perform complex movements efficiently. The Arduino's simplicity and wide range of compatible libraries also reduce development time and effort, making it an excellent choice for both educational and professional robotics projects.

### 2.4.3 Robotic Arm

The robotic arm is the mechanical component of the system, designed to mimic human arm movements with precision and flexibility. It is equipped with multiple servo motors, which serve as actuators to control the joints. Servo motors are preferred because of their ability to rotate to specific angles, providing precise control over the robotic arm’s movements. This enables the arm to perform actions such as gripping, lifting, placing objects, and other intricate manoeuvres. The robotic arm's structure typically includes segments and joints that replicate the degrees of freedom found in a human arm, allowing for a wide range of motion. The combination of servo motors and lightweight materials ensures that the arm is both powerful and efficient, capable of handling delicate tasks as well as heavier loads. With its modular design, the robotic arm can be customized to suit various applications, from industrial automation to gesture-controlled interactive systems.

## Software Tools

### 2.4.4 OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source library for real-time computer vision and machine learning applications. For real-time image and video processing, including gesture detection, recognition, and analysis. While primarily written in C++, OpenCV provides bindings for Python, Java, and other languages, allowing developers to create cross-platform applications. OpenCV has numerous modules, such as those for object detection, image processing, and machine learning (Sharma et al, 2021). Its capabilities include image transformation, object tracking, and integration with deep learning frameworks such as TensorFlow and PyTorch.

### 2.4.5 Programming Environment

The Arduino Integrated Development Environment (IDE) is a software platform used to write, compile, and upload code to microcontroller boards such as Arduino. It provides a user-friendly interface for programming and debugging, making it an essential tool for controlling hardware components like motors and sensors in robotics. Processing, a crucial open-source programming language, forms the foundation of the Arduino programming language (Pramoto & Perdana, 2017). The IDE supports a wide range of Arduino compatible boards and features built-it libraries for functions like motor control, sensor data processing, and communication protocols. For robotic arm projects, the Arduino IDE facilitates precise motor control by allowing developers to write custom code for task like positioning the arm, adjusting grip strength, synchronizing movement, Its simplicity, combined with extensive community support, makes the Arduino IDE a popular choice for both beginners and advanced robotics engineers.

### 2.4.6 Mediapipe

Google developed MediaPipe, a cross-platform machine-learning library that simplifies complex vision-based tasks, making them easier to implement. MediaPipe provides pre-built solutions for real-time applications such as pose estimation, facial recognition, hand gesture detection, and more. In the context of robotics, MediaPipe can be utilized to detect hand movements and control a robotic arm through gestures. For example, gestures like pointing, waving, or forming specific finger shapes can be mapped to robotic arm actions such as picking up an object or moving to a designated position. One of MediaPipe's key advantages is its extensibility, which allows it to be adapted for use across various programming languages and operating systems, making it one of the most versatile and sought-after toolkits available today (Chunduru et al., 2021). Its real-time processing capabilities and modular design make it particularly effective for low-latency tasks, such as interactively directing a robotic arm. By abstracting much of the complexity involved in training and deploying machine learning models, MediaPipe enables developers to streamline the integration of AI into robotics applications, significantly reducing development time and effort.

### 2.4.7 Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android app development, developed by Google. It is built on IntelliJ IDEA and provides an extensive set of tools and resources for creating, debugging, and optimizing Android applications for various devices, such as smartphones, tablets, TVs, and wearables. Its intelligent code editor supports languages like Kotlin, Java, and C++, offering features like real-time error checking, code completion, and refactoring. Android Studio also includes a visual Layout Editor, which simplifies the design of user interfaces with drag-and-drop components and XML editing, allowing developers to preview layouts for different screen sizes and resolutions. One of Android Studio's key strengths is its Gradle build system, which automates app builds and handles dependencies and configurations. It also comes with a powerful emulator that allows developers to test their applications on virtual devices with different configurations and Android versions, eliminating the need for physical devices during development. Furthermore, it offers profiler tools to analyze app performance in real time, debugging tools like Logcat to fix runtime issues, and seamless integration with version control systems like Git. Developers

can also leverage Jetpack libraries and tools for modern Android development, along with extensive support for testing using JUnit and Espresso frameworks. Android Studio is particularly suitable for integrating a gesture-controlled robotic arm project for kids due to its comprehensive support for creating interactive and user-friendly mobile applications. The IDE's visual tools make it easy to design an intuitive interface where kids can control the robotic arm using gestures, ensuring simplicity and engagement. Additionally, its support for advanced frameworks and libraries allows seamless integration with gesture recognition technologies, sensors, and Bluetooth connectivity to interact with the robotic arm. With its debugging tools and emulators, developers can test and refine the application to ensure smooth functionality and a delightful user experience tailored to children. This makes Android Studio an ideal choice for developing innovative and educational projects like gesture-controlled robotics.

## 2.5 Critical Review

The three existing solutions—Leap Motion-based systems, MEMS-based wearable systems, and Microsoft Kinect-based systems—each offer distinct advantages and drawbacks. A comparison with the proposed Gesture-Controlled Robotic Arm for Kids highlights how this system is uniquely designed to meet the needs of children in educational settings.

Leap Motion-based systems are highly precise, capable of tracking finger movements with millimetre accuracy, making them ideal for tasks requiring fine motor control, such as virtual reality, gaming, or precision robotics. They are non-contact and easy to use, eliminating the need for wearables or physical attachments. Additionally, their compact design makes them portable, though they require a computer for operation. However, their reliance on line-of-sight tracking and sensitivity to ambient lighting reduce effectiveness in uncontrolled environments. The high cost and dependence on a PC further limit their suitability for budget-conscious or portable applications.

MEMS-based wearable systems use sensors like accelerometers and gyroscopes to reliably track dynamic movements, even in challenging lighting or environmental conditions. These wearables are cost-effective and versatile, finding applications in robotics control, medical rehabilitation, and industrial automation. Despite these strengths, wearables can be uncomfortable for extended use, particularly for children, and require regular calibration to prevent sensor drift, which may affect accuracy over time.

Microsoft Kinect-based systems, equipped with depth-sensing technology, excel in full-body gesture recognition and spatial tracking, making them ideal for collaborative or large-scale environments. With their wide field of view and ability to track multiple users simultaneously, Kinect systems are highly versatile for tasks such as collaborative robotics and rehabilitation. However, their bulky design, reliance on controlled lighting, and high cost limit portability and adaptability. Additionally, predefined gesture sets make them less suitable for customization in educational or child-focused applications.

The Gesture-Controlled Robotic Arm for Kids overcomes the challenges of these systems while aligning with the needs of young learners. By focusing on simplicity, the system achieves sufficient accuracy for basic hand gestures, eliminating the complexity of high-precision systems like Leap Motion and Kinect. This makes it intuitive and user-friendly for

children without requiring extensive training. The system also prioritizes affordability by utilizing inexpensive components such as webcams or smartphone cameras for gesture detection and Arduino boards for control. This cost-effective approach ensures greater accessibility for schools and families, particularly where budget constraints are a concern.

# Chapter 3

## 3.1 Introduction

In this chapter, we will be discussing more on the requirement analysis and design. To fulfil the first objective, a questionnaire was conducted to gather user requirements on the design development. We will also discuss on the methodology used for the development of our proposed project here.

## 3.2 Methodology

The "Gesture-Controlled Robotic Arm for Kids" was developed by integrating hardware and software, with a focus on young users' safety, ease of use, and engagement. The project uses the Agile Software Development Life Cycle (SDLC), which offers an iterative and flexible framework, to accomplish these objectives. Agile's adaptability makes it ideal for projects with changing needs and ongoing user input.

Establishing the project's vision and overarching goals is the first step in the technique. The objective is to combine children's enjoyment and education by developing an interactive robotic arm that reacts to simple hand movements. High-priority capabilities include the ability to recognize motions like pointing or waving and convert them into actions like turning the arm or picking up objects. These goals direct the development of a product backlog that include activities including safety features, robotic arm control, and gesture recognition.

The project is broken up into digestible sprints, each of which focuses on a different module, per Agile's iterative planning method. Early sprints, for instance, focus on creating gesture recognition algorithms that use an Arduino and camera combination to drive robotic arm movements. The system is improved with more gestures, safety features, and user calibration choices in later sprints. This tiered strategy guarantees ongoing development and frequent input from all parties involved, including teachers and students.

### 3.3 Requirement Planning

According to 3.2 Methodology, requirement planning is a phase in SDLC where the requirements or goals are defined. For the proposed system, a gesture-controlled robotic arm, the requirements are gathered through discussions with someone are very familiar with the robot arm, and system requirements are based on the literature review that was conducted in the previous chapter.

#### 3.3.1 System Requirements

According to Yahya et al (2021), through an interview, researchers can delve into the specifics and comprehend the thoughts, feelings, and experiences of others to examine the world from the respondents' point of view. From the discussion, we found out that the current robotic arm is less interesting among kids as it is very difficult to control with a joystick. Also, the previous design of the interface of its control was poor and needed to be improved with gesture hand.

With the current issues that we have, the proposed project was designed to meet the user requirements. The user requirements of this project are stated below:

1. The user required a  better control interface as a platform for the user to control the robotic arm
2. The user requires certain hand gestures to control the movement of the robot's arm.

#### 3.3.2 Project Requirements

Functional requirement:

Functional requirements are referred to as the functions and behaviors of the project.

- The control interface shall allow the user to move the robot moves according to the user’s hand.
- It also can allow the user to act and pick up objects, as user according to users will.
- It also be able to function from a distance via the connection through Bluetooth.
- The system can detect hand gestures.

Non-functional requirement

Non-functional requirements are referred to as the quality requirements of the overall project.

- The gesture detected must be an appropriate gesture when commanding the robotic arm
- When controlled by the user, the robotic arm should be able to move smoothly in any direction.

Hardware requirement:

The hardware requirements for the development of the project.

Table 2: Hardware requirement

| Hardware Component | | Requirement |
|---|---|---|
| Laptop | Processor (CPU) | Intel® Core ™ i3 and above |
| | Memory (RAM) | 4 GB and above |
| | Graphic Card | 750 GB and above |
| | OS Platform | 32-bit and above |
| Mobile Phone | Android Platform | Android Version 7.0 and above |
| | Processor (CPU) | 1.6 GHz and above |
| | Internal Memory (RAM) | At least 2 GB |
| | External Memory (ROM) | At least 1 GB |

Software requirement:

The software requirements for the development of the project.

Table 3: Software requirement

| **Software** | **Descriptions** |
|---|---|
| Windows 10 | Window 10 will be used during development to carry out the proposed project until its completion. |
| C++ program | C++ program is a general-purpose computer programming language. In the proposed project, C++ program is the language supported by the Arduino Studio IDE. |
| Arduino Studio IDE | Arduino Studio IDE is a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It will be used to write the Arduino coding for the robotic arm to function. |

## 3.4 System Design

In this section, the project's system design is discussed, and it is required during system implementations as a guideline. The system design consists of system architecture, use case diagram, use case description, flow chart, context diagram, data flow diagram, entity relationship diagram, data dictionary, small arm robot design, and user interface design.

### 3.4.1 System Architecture Design

The figure 10 shows the overall system architecture of the project, the proposed system will begin with a mobile device equipped with a camera, a Bluetooth communication module, and a robotic arm. The system starts with the mobile device, which serves as the central interface and processing hub. Using its camera, the device captures visual inputs such as images or video of objects or gestures in its environment. Through on-device processing powered by computer vision algorithms or AI models, the mobile device interprets the captured data to identify objects, recognize gestures, or scan codes. It translates these inputs into actionable commands that are transmitted via Bluetooth. Additionally, the mobile device provides a user-friendly interface for system configuration, monitoring, and feedback. The Bluetooth communication module acts as a bridge, ensuring seamless and secure data transfer between the mobile device and the robotic arm. It handles command transmission efficiently, mitigating data loss and latency to ensure precise coordination. Commands received from the mobile device are interpreted and executed with precision, enabling the arm to respond dynamically to its environment.

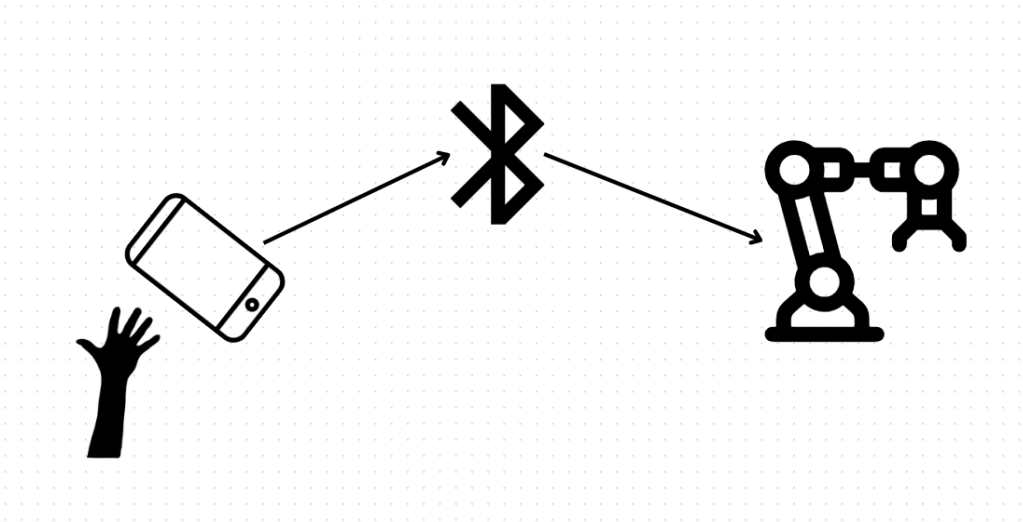


Figure 10: System Architecture Design

### 3.4.2 Use Case Diagram

The use case diagram illustrates the interaction between the child user and the robotic arm. Key actions include gesture recognition, robotic arm movement, object pickup, and rotation

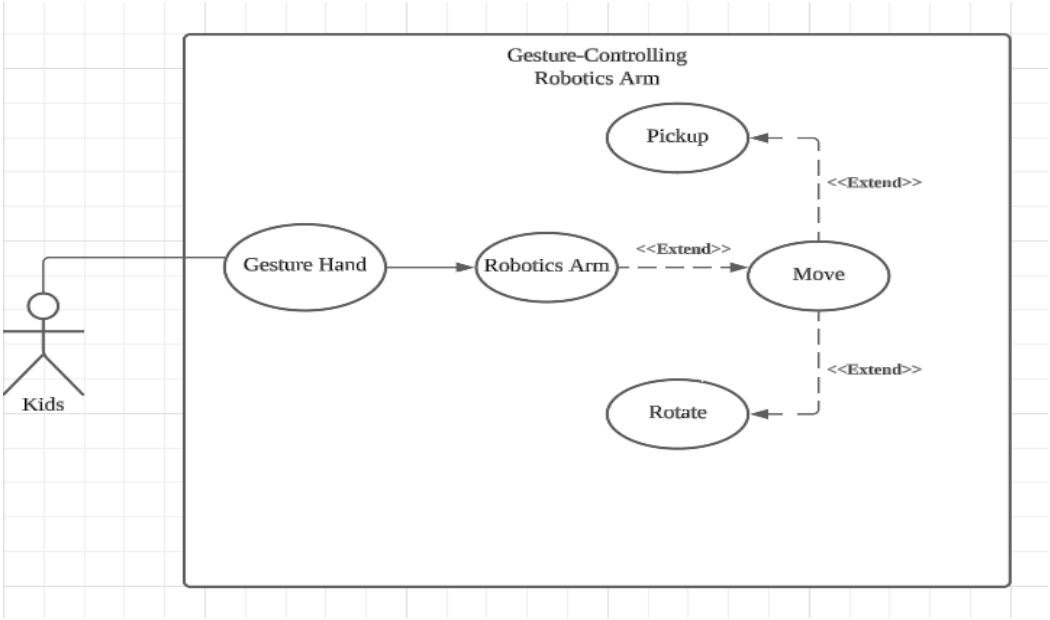



Figure 11: Use Case Diagram

### 3.4.3 Use Case Description

Table 4: Use Case Description

| Actors | Brief Description |
|---|---|
| Kids | The kids will perform hand gestures, and the robotic arm will respond accordingly. Specific gestures will prompt the robotic arm to perform actions such as moving, picking up objects, or rotating. |

| Use Case | Brief Description |
|---|---|
| Gesture Hand | The child performs specific hand gestures that are captured by the system. These gestures act as commands for the robotic arm to execute various actions. |
| Robotics Arm | The robotic arm processes the received gesture commands and performs mechanical actions, such as movement, object pickup, or rotation. |
| Move | A basic command where the robotic arm moves to a specified position. This movement serves as the foundation for other actions. |
| Pickup | An extension of the **Move** use case, where the robotic arm picks up an object from the current location after moving into position. |
| Rotate | Another extension of the **Move** use case, enabling the robotic arm to rotate its base or gripper to the desired angle. |

### 3.4.4 Flow Chart

Figure 12 the flow chart shows how the application using Media pipe and Arduino works. When the system starts, it will begin with an open application reading the hand gesture. After capturing the hand gesture data, it will transferred to the robot arm.

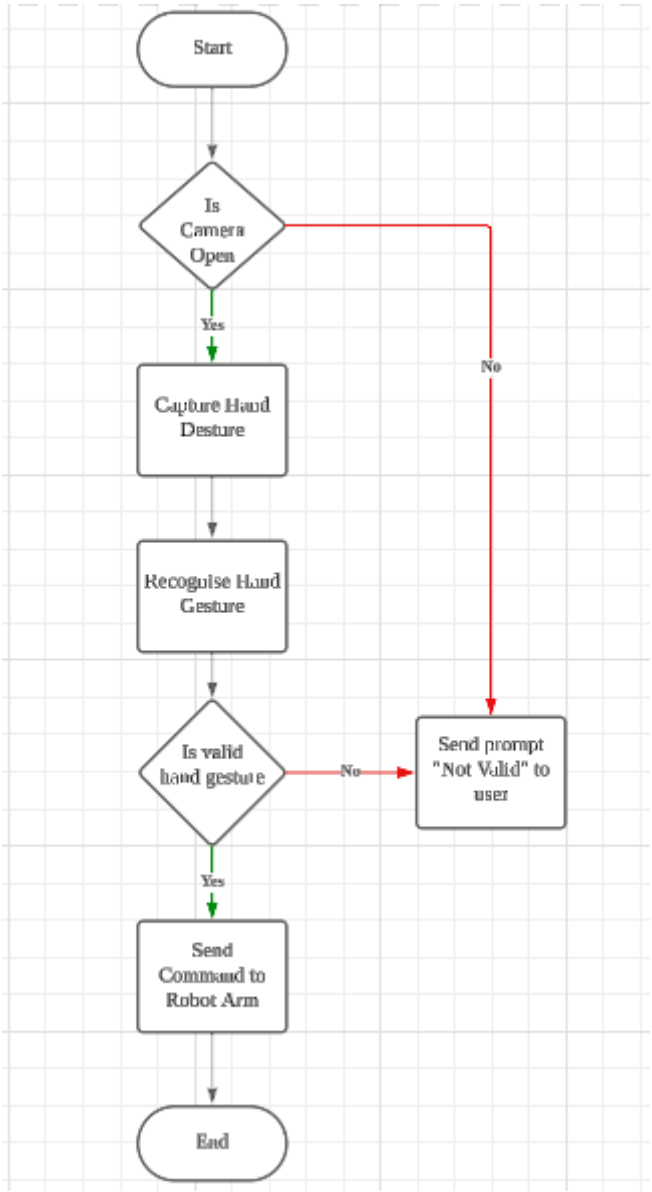



Figure 12: Flow Chart

### 3.4.5 Context Diagram

A context diagram is used to illustrate the relationship between a system and external entities involved in the system. It is used to represent the high-level process of the Data Flow Diagram.

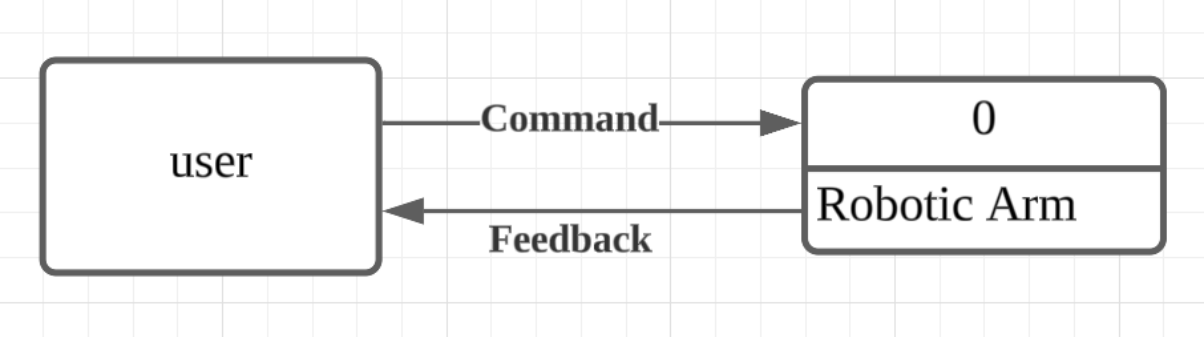
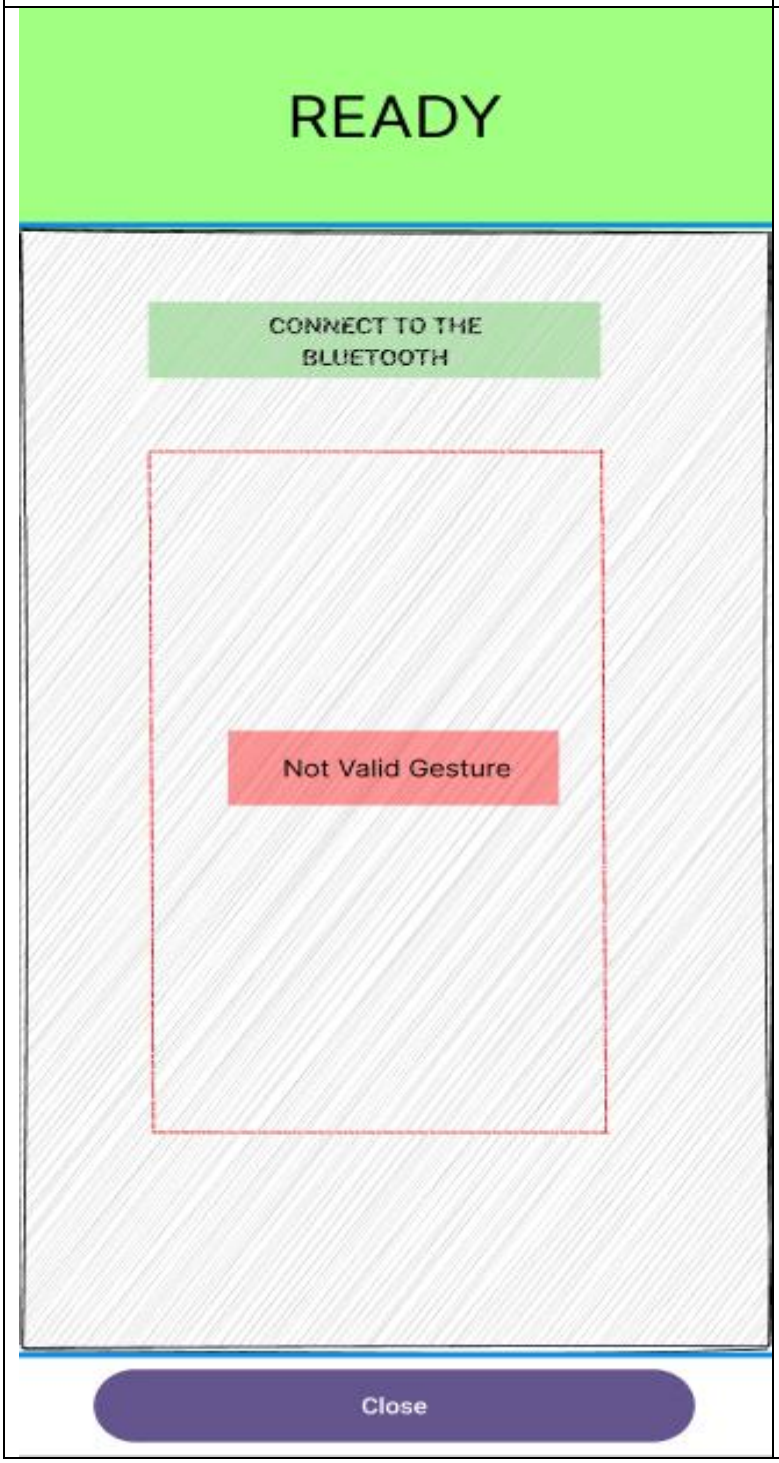


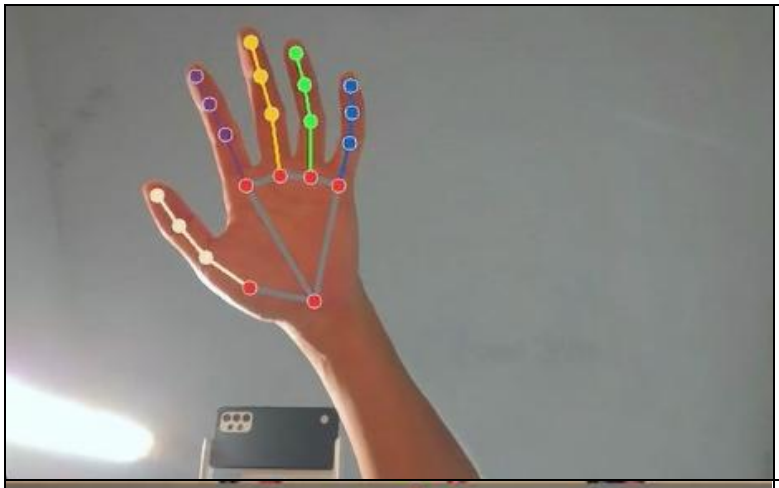
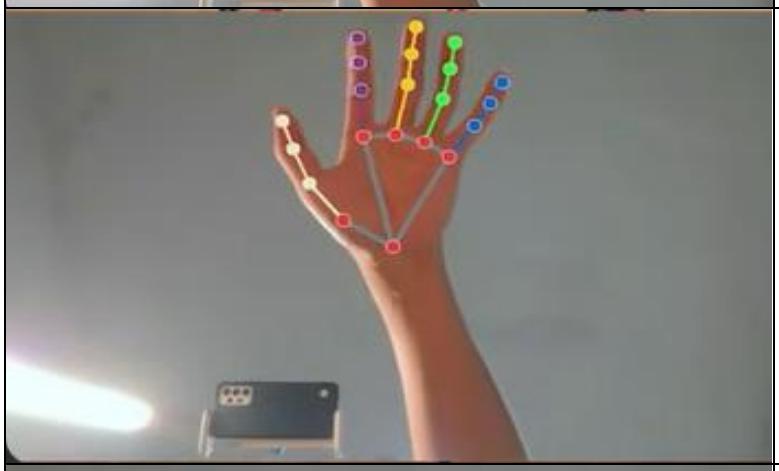
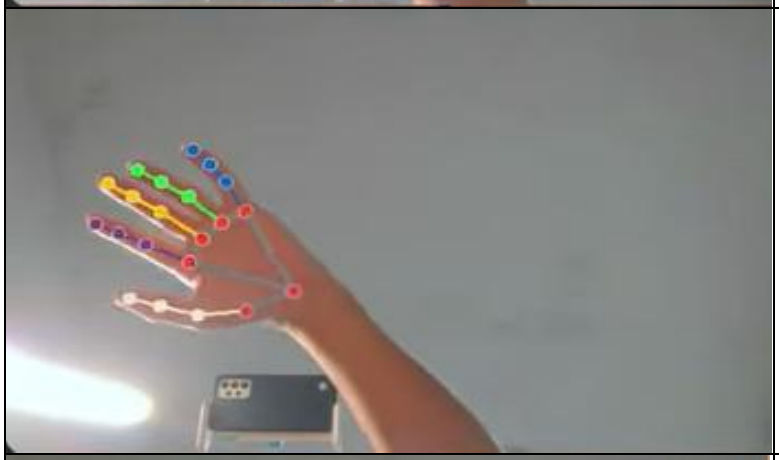
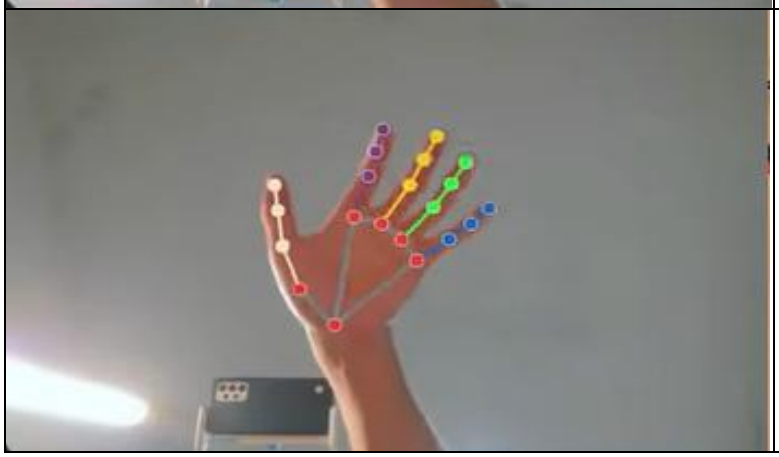

Figure 13: Context Diagram

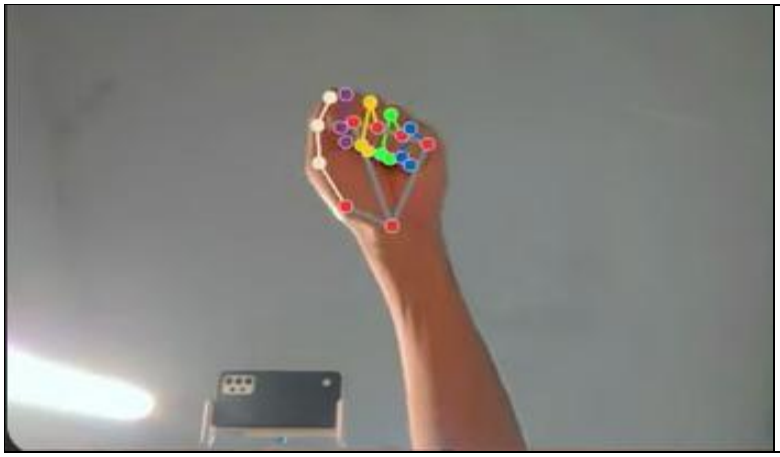
## 3.5 Control Interface Design

The robotic arm will have a simple and easy-to-understand design of its control interface. The user interface design of the project is as follows:

Table 5: User Interface Design

| Design | Description |
|---|---|
|  | This is a full view of the interface that detects hand gestures. When the hand is not detected, it will notify the user "Not Valid Gesture". There is a close button after using the robot arm. |

| | |
|---|---|
|  | This is a hand gesture for the robot arm to stay still |
|  | This is a hand gesture to move forward to extend the robot arm extension. |
|  | This is a hand gesture to move them rotating anticlockwise in a 180-degree rotation. |
|  | This is a hand gesture to move them rotating clockwise in a 180-degree rotation. |

| | |
|---|---|
|  | This is a hand gesture for grab action. |

## 3.6 Summary

This chapter discussed the requirement analysis and system design of the project. In the requirement analysis phase, it has been observation and gathering of the user requirements and system requirements of the existing system. Used to construct the user and system requirements for the proposed system. The user requirements are been gathered based on analysis and questionnaires related to the system. Moreover, the system requirements contain all the details of how the system shall operate and behave. In the system design phase, there is an outline of the use case diagram, a use case description, a flowchart, a context diagram of the robotic arm, a data flow diagram, and a user interface design for the control interface.

# Chapter 4

## 1.1 Introduction

Following the design and analysis phase described in Chapter 3, this chapter outlines the implementation of the Gesture-Controlled Robotic Arm for Kids. The implementation includes the setup of development environments, component testing, integration of gesture recognition, and communication between the mobile application and the Arduino-controlled robotic arm using MediaPipe, OpenCV, and Bluetooth.

## 1.2 Environment Setup and Installation

To ensure smooth development, two main software environments were used: Arduino IDE for embedded programming and Android Studio for mobile application development.

### 1.2.1 Arduino IDE

The Arduino Integrated Development Environment (IDE) is an open-source platform used for writing and uploading code to Arduino-compatible boards. It served as the main development tool to control the servo motors of the robotic arm.

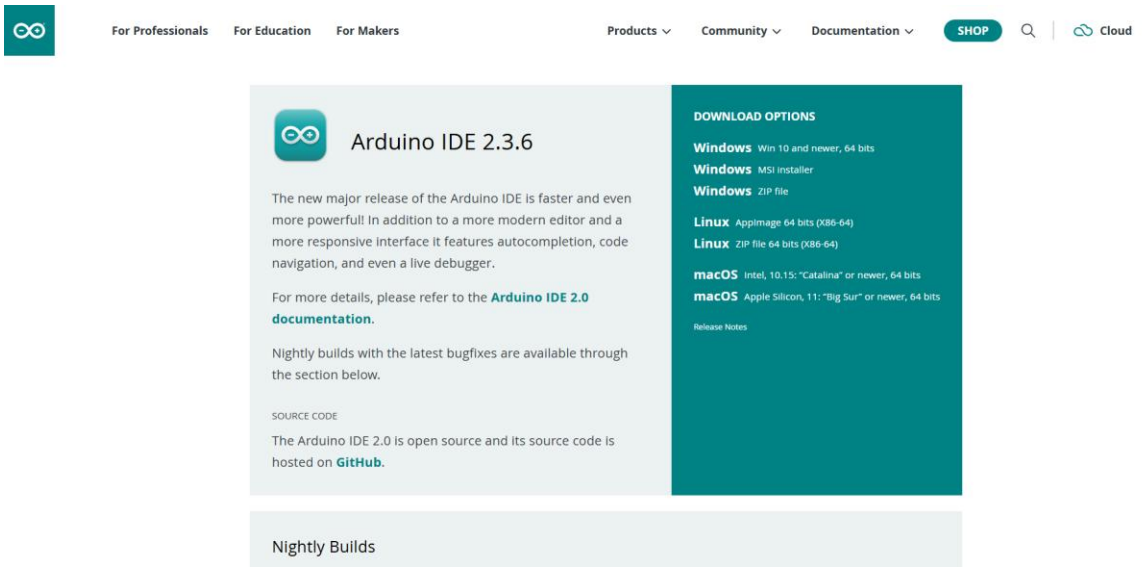



Figure 14: Arduino Official Website

The Arduino IDE is a well-known software that many users prefer for uploading programs and communicating with them. Arduino IDE can be downloaded freely at https://www.arduino.cc/en/software. The official Arduino IDE website is shown in the figure, and the user ID is required to select and install the Arduino IDE version that is compatible with the operating system, by referring to the Download Options section.

After the installation, the Arduino IDE can be run automatically or manually clicked depending on the user's preference. Here is an overview of the installed Arduino IDE software.

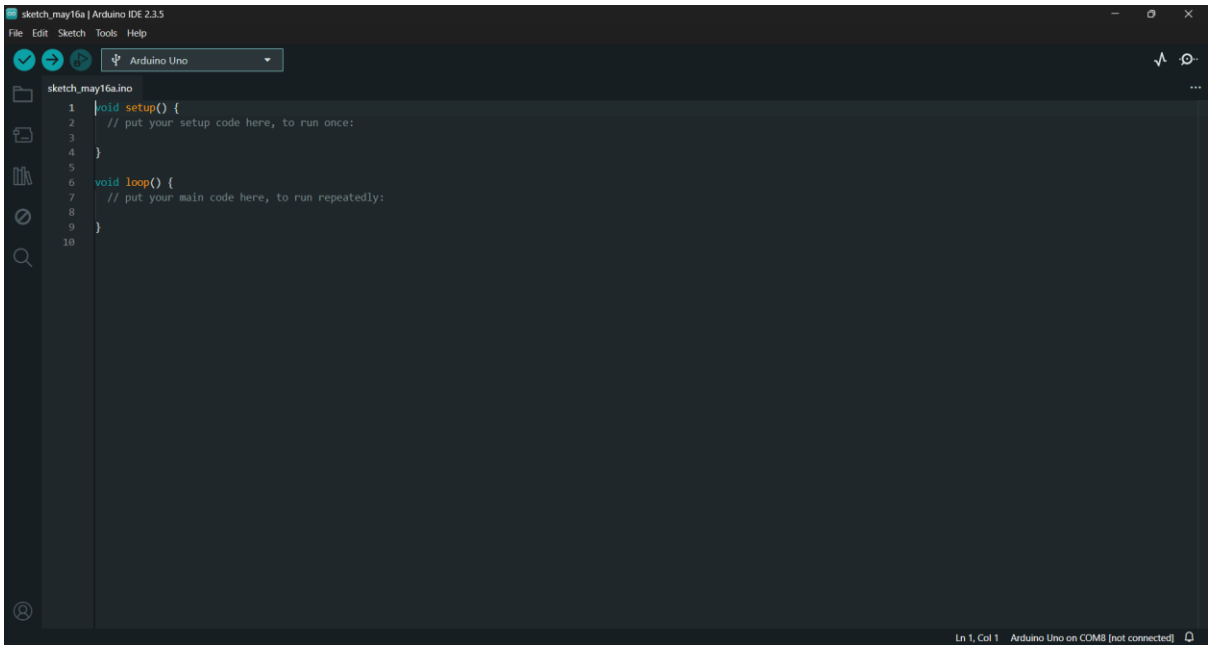



Figure 15: Arduino IDE after complete installation

Based on the figure above, there are two required functions in an Arduino sketch: setup() and loop(). Other optional functions must be created outside the brackets of those two functions.

### 1.2.2 Android Studio

Android Studio is the official integrated development environment (IDE) for the Google Android operating system and is designed specifically for Android development. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native (local) Android application development. In this project, Android Studio was used to develop the mobile application that communicates with the Arduino via Bluetooth to control the robot arm.

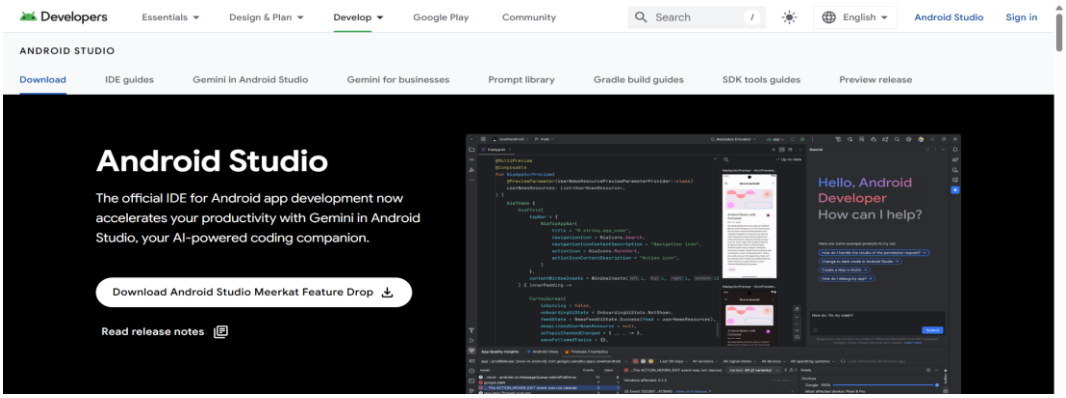



Figure 16: Android Studio Official Website

The Android Studio website provides installation files for different operating systems. Once installed, developers can create and test apps that interact with hardware components, such as Bluetooth modules.

### 1.3 Test-run before Development

Before integrating the entire system, preliminary testing was carried out to validate servo motor functionality and gesture recognition components.

#### 1.3.1 Servo Testing

To validate the servo hardware, six servos (MG996R and SG90 models) were programmed using basic Arduino code:

```
#include <Servo.h>
Servo s1, s2, s3, s4, s5, s6;

void setup() {
  s1.attach(0); s2.attach(1); s3.attach(2);
  s4.attach(3); s5.attach(4); s6.attach(5);
                                                }

void loop() {
  s1.write(0); s2.write(0); s3.write(0);
  s4.write(0); s5.write(0); s6.write(0);
  delay(1000);

  s1.write(180); s2.write(180); s3.write(180);
  s4.write(180); s5.write(180); s6.write(180);
```

```
  delay(1000);

}
```

Figure 17: Source code for servo testing

Each of the servo motors was declared as s1, s2, s3, s4, s5, and s6, respectively. In the setup section, the servo motors are attached according to the pins they are connected to on the board shield.

As for the loop section, the write function was used repeatedly to determine the position of the motors. The value 0 means that the servo motors started from their initial position, whilst the value 180 means that they move about 180 degrees before returning to their original position. The delay indicates the time interval taken for the loop to execute continuously.

### 1.3.2 MediaPipe and OpenCV

MediaPipe, developed by Google, is a cross-platform framework used for processing perceptual data such as video and images. In this project, it was used for hand tracking and gesture recognition. OpenCV (Open Source Computer Vision Library) was employed to handle real-time camera input and image processing.

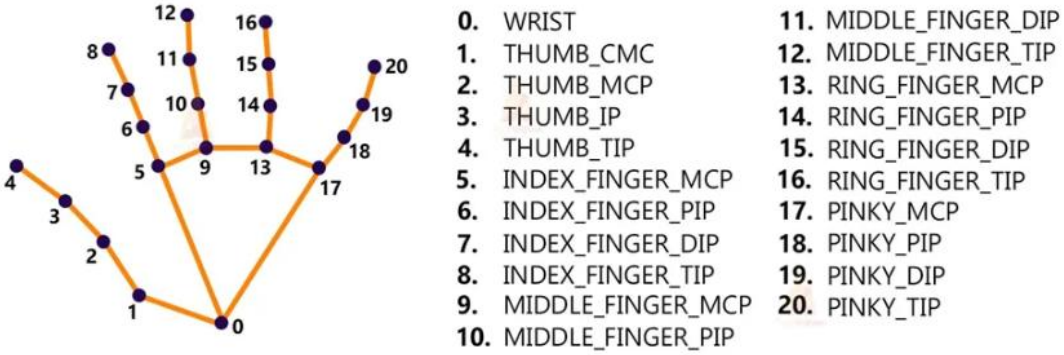
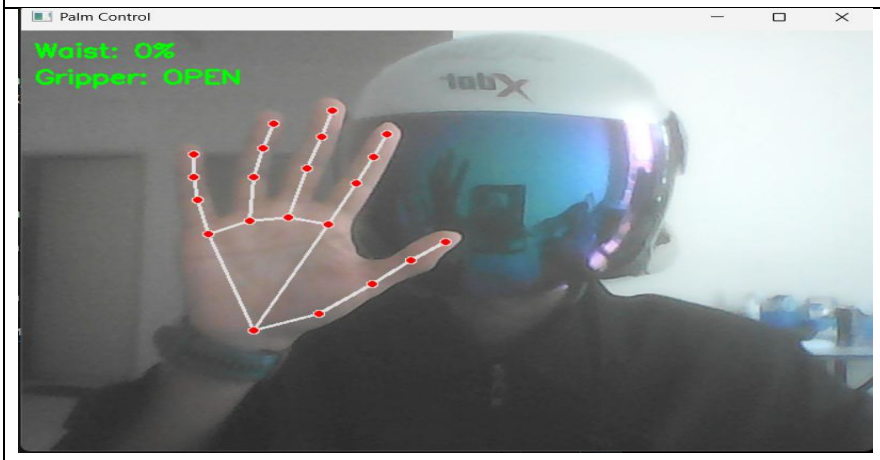
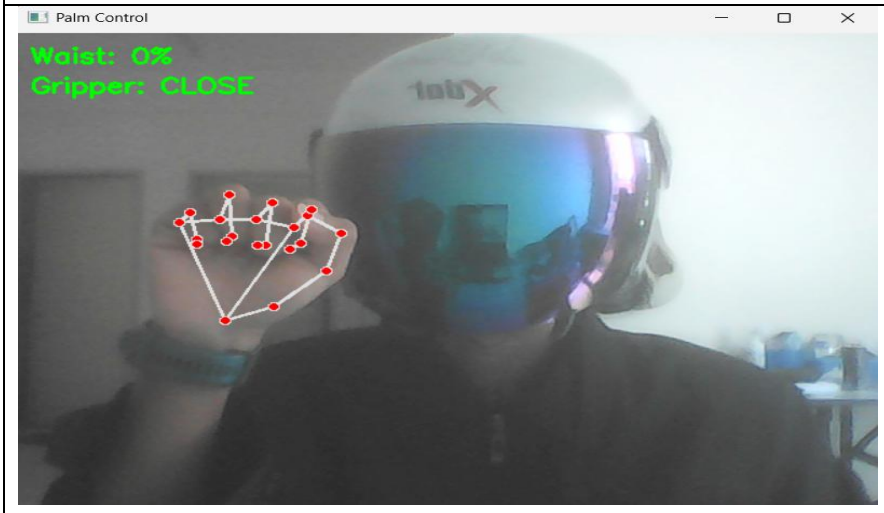
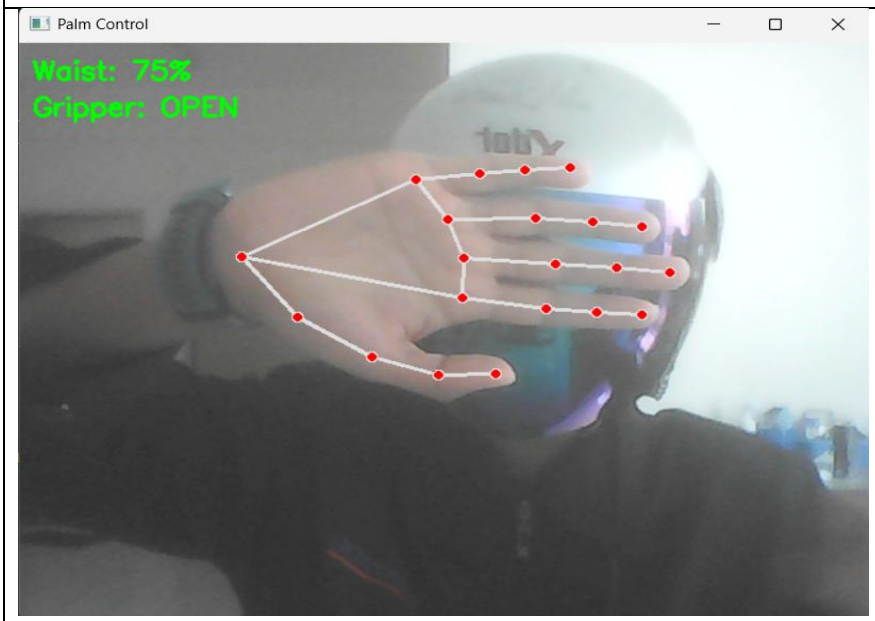


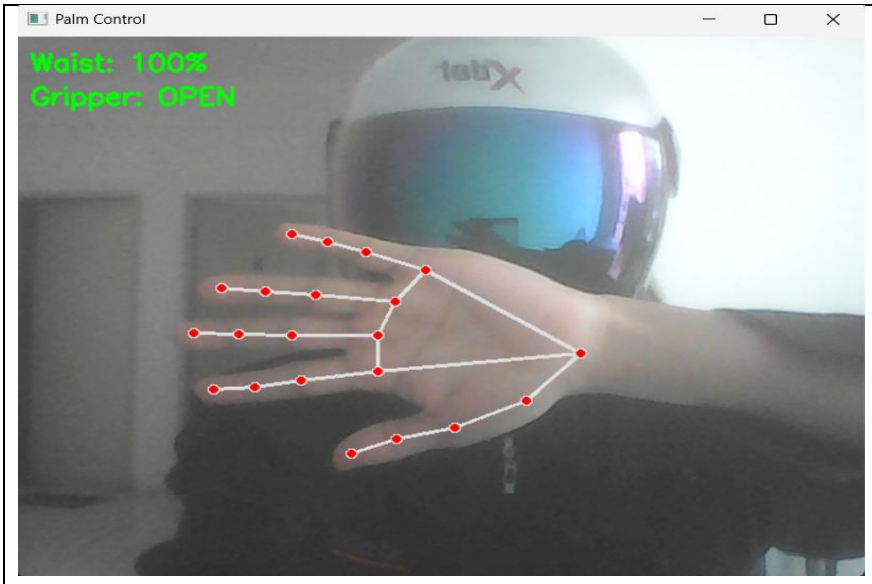

Figure 18: MediaPipe Hand Tracking Module

According to Gautam (2024), the MediaPipe Hand module detects 21 key landmarks on the hand (e.g., fingertips, wrist, knuckles). These landmarks are analyzed in real time to determine gestures such as an open palm, closed fist, or directional tilt. OpenCV assists with capturing camera frames, converting them to RGB, and overlaying results.

By integrating both tools, the system was able to extract meaningful hand data—such as palm orientation, finger distance, and wrist height—to generate control signals. These were then transmitted via Bluetooth to the Arduino for motor control. This combination of computer vision and hardware integration mirrors applications in sign language recognition, prosthetics, and robotics.

Table 6: Sample Hand Gestures and Their Mapped Robot Commands

| Hand Gesture | Description |
|---|---|
|  | This hand gesture command the robot arm to open the gripper. |
|  | This hand gesture command the robot arm to close the gripper. |
|  | This hand gesture command the robot arm to move right. |

| | |
|---|---|
|  | This hand gesture command the robot arm to move left |

## 1.4 Development

After confirming the functionality of individual components, the Arduino code and mobile app were integrated to finalize the robotic arm system.

### 1.4.1 Arduino Motor Control (RobotArm.ino)

The core Arduino code (RobotArm.ino) is responsible for interpreting Bluetooth signals and controlling the servo motors. When a gesture is detected on the mobile app, a corresponding command is sent via Bluetooth to the HC-05 module, which is connected to the Arduino.

```
 #include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
#include <SoftwareSerial.h>
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();
// Servo channels
#define WAIST_PIN 0
#define SHOULDER_PIN 1
#define ELBOW_PIN 2
#define HOLDER_PIN 3
#define HOLDERGRIPPER_PIN 4
#define GRIPPER_PIN 5
// Limits
const int WAIST_MIN = 150, WAIST_MAX = 600;
const int SHOULDER_MIN = 300, SHOULDER_MAX = 600;
```

```
const int ELBOW_MIN = 400, ELBOW_MAX = 550;
const int GRIPPER_OPEN = 150, GRIPPER_CLOSE = 600;
int waistPos = (WAIST_MIN + WAIST_MAX) / 2;
int shoulderPos = SHOULDER_MAX;
int elbowPos = ELBOW_MAX;
int holder = 100;
int holdergripper = 150;
const int STEP = 20;
String inputBuffer = "";
SoftwareSerial BT_Serial(3, 2); // RX=3, TX=2
void setup() {
  Serial.begin(38400);
  BT_Serial.begin(38400);
  pwm.begin();
  pwm.setPWMFreq(60);
  pwm.setPWM(WAIST_PIN, 0, waistPos);
  pwm.setPWM(SHOULDER_PIN, 0, shoulderPos);
  pwm.setPWM(ELBOW_PIN, 0, elbowPos);
  pwm.setPWM(HOLDER_PIN, 0, holder);
  pwm.setPWM(HOLDERGRIPPER_PIN, 0, 375);
  pwm.setPWM(GRIPPER_PIN, 0, GRIPPER_OPEN);
  Serial.println("Device is ready");
}
void loop() {
```

```
  while (BT_Serial.available()) {
    char c = BT_Serial.read();
    Serial.print(c); // Debug
    if (c == '\n') {
      processCommand(inputBuffer);
      inputBuffer = "";
    } else if (isPrintable(c)) {
      inputBuffer += c;
    }
  }
  // Optional: echo PC serial commands to Bluetooth
  while (Serial.available()) {
    char c = Serial.read();
    BT_Serial.write(c);
  }
}
void processCommand(String command) {
  command.trim();
  Serial.println("Processing: " + command);  // Debug log
  int colonIndex = command.indexOf(':');
  if (colonIndex <= 0 || colonIndex >= command.length() - 1) {
    Serial.println("Malformed command: " + command);
    return;
  }
```

```
char cmd = command.charAt(0);
char dir = command.charAt(colonIndex + 1);
static char lastGripState = '-';
switch (cmd) {
  case 'W':
    waistPos += (dir == '1') ? STEP : -STEP;
    waistPos = constrain(waistPos, WAIST_MIN, WAIST_MAX);
    pwm.setPWM(WAIST_PIN, 0, waistPos);
    break;
  case 'S':
  case 'E':
    // Ignore: handled together with G below
    break;
  case 'G':
if (dir != lastGripState) {
  lastGripState = dir;
  if (dir == '1') {  // Fist gesture
    // Step 1: Move elbow and shoulder to MIN (grabbing position)
    shoulderPos = SHOULDER_MIN;
    elbowPos = ELBOW_MIN;
    pwm.setPWM(SHOULDER_PIN, 0, shoulderPos);
    pwm.setPWM(ELBOW_PIN, 0, elbowPos);
    delay(400);  // Allow time to reach
    // Step 2: Close gripper
```

```
      pwm.setPWM(GRIPPER_PIN, 0, GRIPPER_CLOSE);
      // Step 3: Wait 1 seconds
      delay(1000);
      // Step 4: Move elbow and shoulder to MAX (lift while holding)
      shoulderPos = SHOULDER_MAX;
      elbowPos = ELBOW_MAX;
      pwm.setPWM(SHOULDER_PIN, 0, shoulderPos);
      pwm.setPWM(ELBOW_PIN, 0, elbowPos);
    } else if (dir == '0') {  // Open hand
      // Just open the gripper (keep arm in current position)
      pwm.setPWM(GRIPPER_PIN, 0, GRIPPER_OPEN);
    }
  }
  break;
    default:
      Serial.println("Unknown command: " + command);
      break;
  }
}
```

Figure 19: Source code for RobotArm.ino

The purpose of this code was to act as the main code, which received the transmitted data to control the servo motors of the robotic arm. With the HC-05 module already paired with the Android application, any data from the application will be received wirelessly to the robotic

arm whenever a gesture is displayed. Also, the value of the gesture displayed can be viewed on the Serial Monitor of the Arduino.

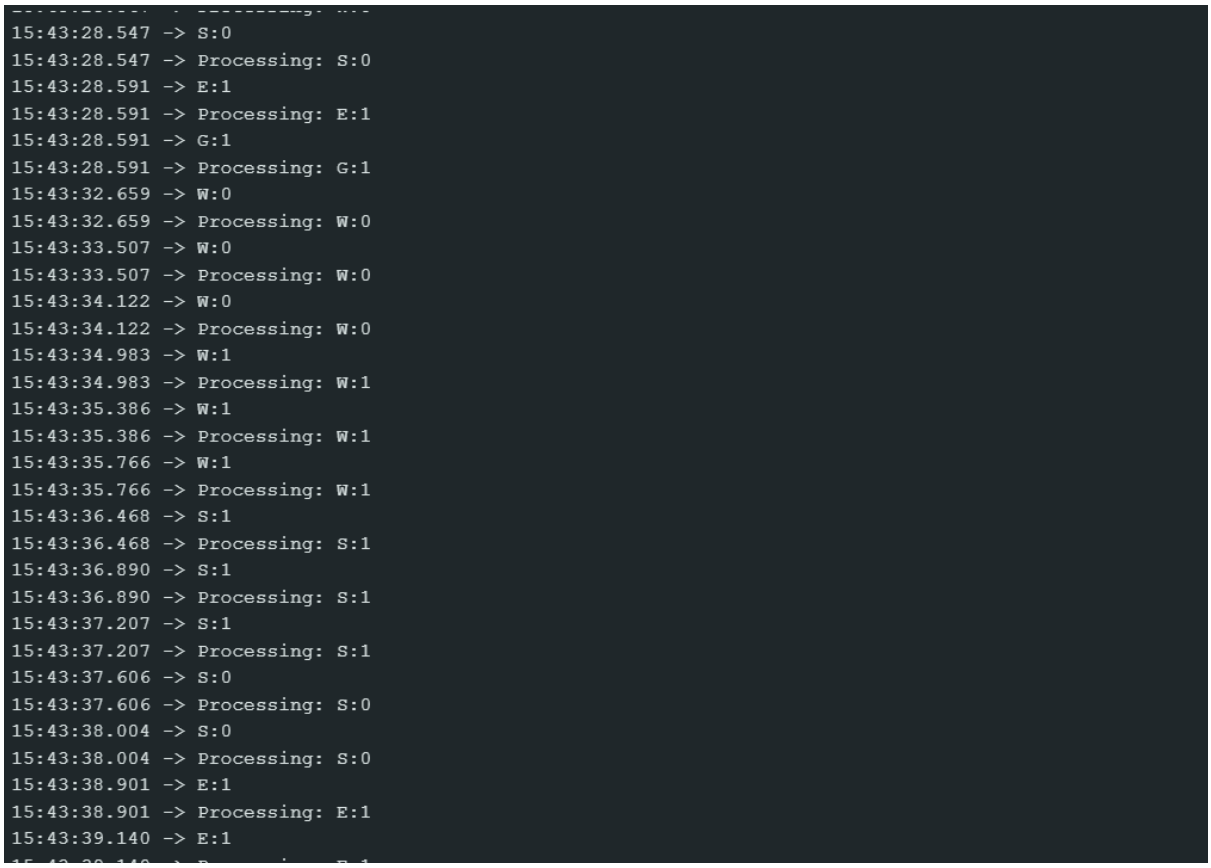



Figure 20: Serial Monitor for RobotArm.ino code

The figure 4.7 are example when the Arduino received data or command from the application.

### 1.4.2 Android Application

The Android application named RobotArm was developed using Android Studio. It has two modes: gesture control (via phone camera and MediaPipe) and manual control (via button presses). It continuously transmits gesture-derived data over Bluetooth using standard I/O streams. Integration testing confirmed that the application could detect gestures in real time and successfully send them to the robot arm.

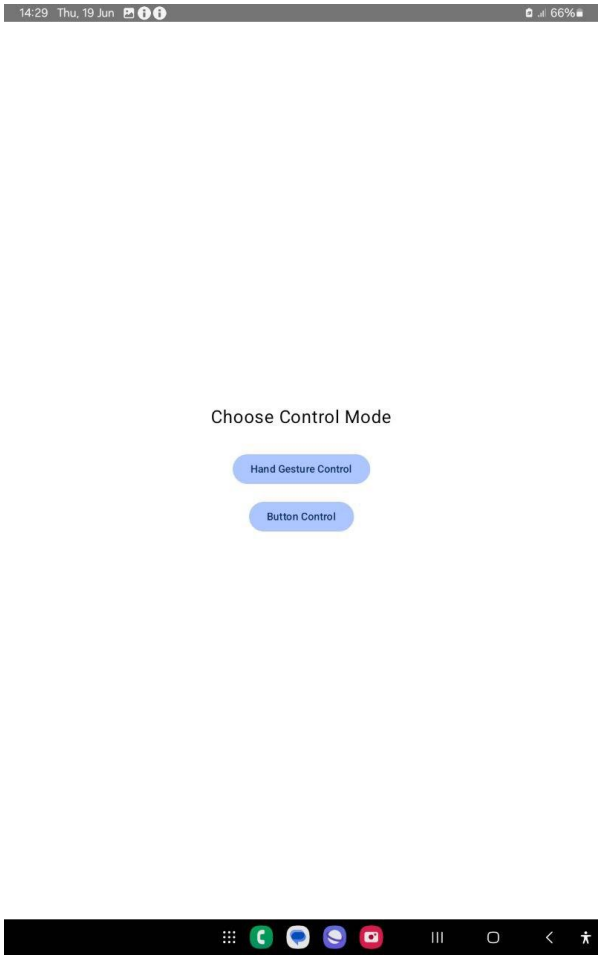
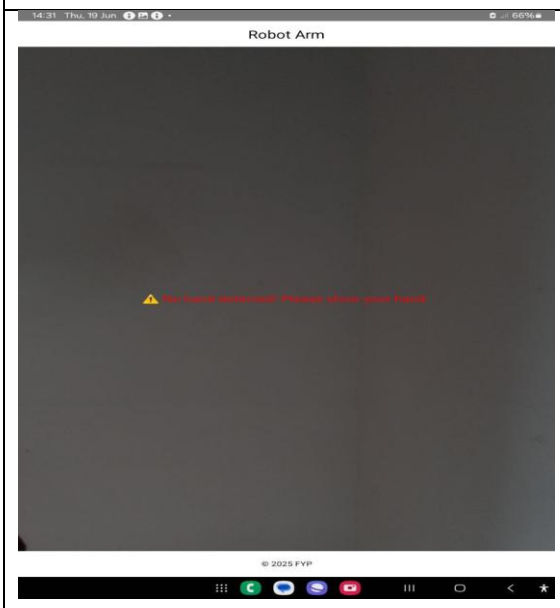
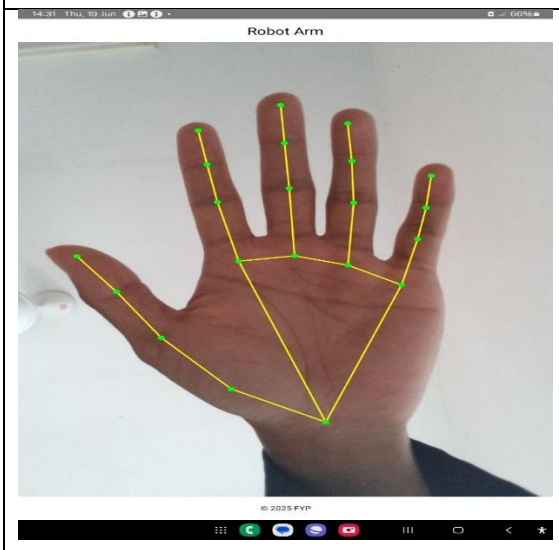
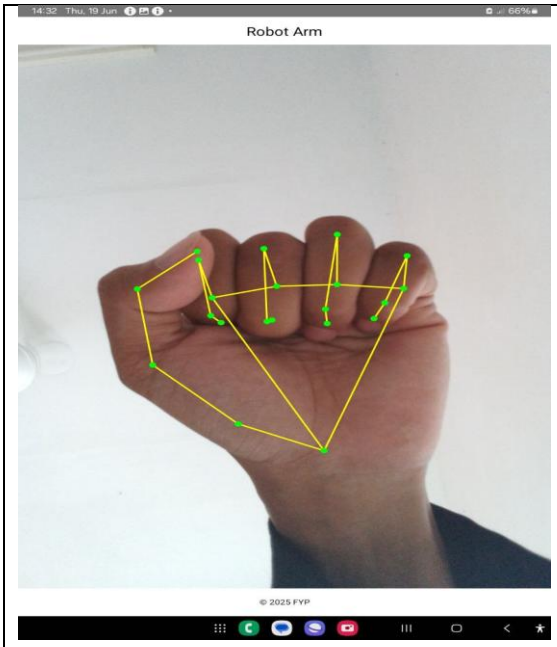
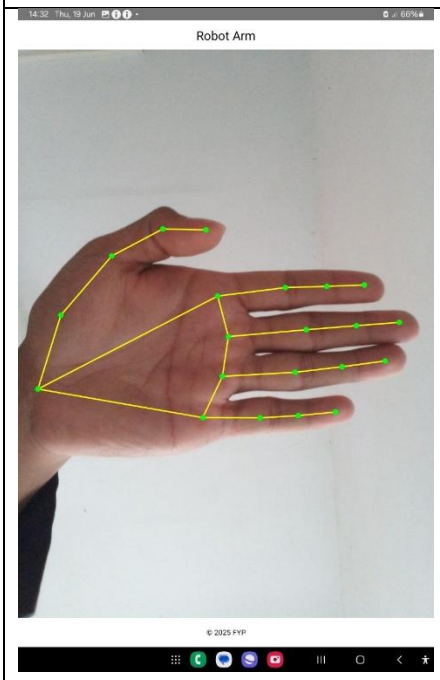


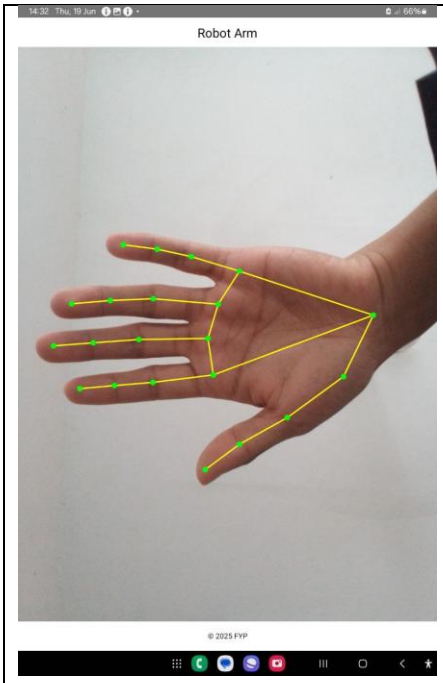

Figure 21: Main Page of RobotArm Application

Figure 4.8 shows the main page of the application. The user can choose two modes to control the robot arm by Hand Gesture or by button control.

Table 7: Gesture-Based Commands in Android App

| Hand Gesture | Description |
| --- | --- |
|  | When there are no hands shown in the screen, there will be a pop-up informing the user to show their hand |
|  | This hand gesture commands the robot arm to opening the gripper |

| | |
|---|---|
|  | This hand gesture commands the robot arm to pick up things and close the gripper. |
|  | This hand gesture commands the robot arm to move the base angle to the right |

| | |
|---|---|
|  | This hand gesture commands the robot arm to move the base angle to the left. |

## 1.5 Summary

This chapter presented the complete implementation of the Gesture-Controlled Robotic Arm for Kids. From setting up development environments to testing individual components and integrating software and hardware, the system was successfully implemented. Tools such as Arduino IDE, Android Studio, MediaPipe, and OpenCV enabled seamless interaction between the mobile device and the robot arm. The system was able to reliably detect hand gestures and execute mechanical movements through Bluetooth communication.

# Chapter 5

## 2.1 Introduction

This chapter presents the testing and evaluation phase of the Gesture-Controlled Robotic Arm for Kids. Testing is a crucial step to verify that the system meets both functional requirements, such as gesture recognition and motor response, and non-functional expectations like responsiveness, compatibility, and ease of use. The testing phase involved iterative debugging, performance measurement, and validation with real users.

## 5.2 Functional Testing

Functional testing verifies whether the robotic arm responds correctly to gestures and whether communication between the Android app and Arduino board is successful.

### 5.2.1 Application Testing

The RobotArm Android application was tested to ensure accurate Bluetooth transmission to the robotic arm. Initial tests revealed that sending gesture data too frequently caused the HC-05 Bluetooth module to become unresponsive. To solve this, a delay threshold of 300 milliseconds was implemented between transmissions to prevent buffer overflow.

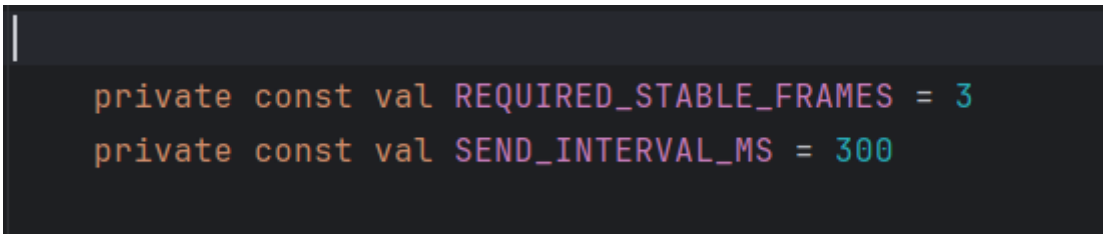



Figure 22: Delay Interval Implemented

This solution improved the stability and ensured reliable command delivery from the app to the robot.

### 5.2.2 Main Code Testing

The final version of the Arduino code was tested under various gesture scenarios. The following table summarizes the test cases and outcomes:

Table 8: Test Case

| **Project name:** | The Development of *Gesture-Controlled Robotic Arm for Kids* | **Designed By:** | Muhammad Zikry bin Mohd Zambri | | |
|---|---|---|---|---|---|
| **Module:** | Motor Control Arduino | **Design Date:** | 16/6/2025 | | |
| **Test Title:** | Robot movement test | **Prerequisite:** | Users understand each gesture to control the robot arm | | |
| **No** | **Test Case** | **Test Procedure** | **Expected Result** | **Pass (Y/N)** | **Remarks** |
| 1 | Make a fist gesture | Show a closed fist in front of the phone camera | The robot gripper closes | Y | |
| 2 | Open palm gesture | Show an open palm to the camera | Robot gripper opens | Y | |
| 3 | Palm tilted to the right | Tilt open palm to the right (fingers point to right) | The robot base rotates right | Y | |

| | | | | | |
|---|---|---|---|---|---|
| 4 | Palm tilted to the left | Tilt open palm to the left (fingers point to the left) | The robot base rotates left | Y | |
| 5 | No hand shown | Do not show hand in front of the camera | Application displays "Please show hand" warning | Y | Useful for user prompt verification |
| 6 | Rapid fist-open cycle | Alternate fist and open palm gestures within 2 seconds | Gripper alternates open and close quickly, without lag | N | Tests responsiveness and speed |
| 7 | Fist + move right | Make a fist and tilt to the right | Gripper remains closed, arm base rotates to right | N | Composite gesture test |
| 8 | Fist + move left | Make a fist and tilt to the left | Gripper remains closed, arm base rotates to left | N | Composite gesture test |
| 9 | Loss of Bluetooth | Turn off Bluetooth or move the phone out of range | The application shows an error or a reconnect prompt | N | Needs auto-reconnect |
| 10 | Reconnect sequence | Restart the app and reconnect | App reconnects and resumes | N | Reconnection unstable |

| | | | | | |
|---|---|---|---|---|---|
| | | to the robot via Bluetooth | gesture detection | | |

From the observation, the application performs well under standard conditions. However, error handling for Bluetooth disconnections should be improved.

### 5.3 Non-Functional Testing

This section evaluates performance, usability, reliability, and compatibility.

Table 9: Non-Functional Testing

| Types of Non-Functional Testing | Description |
|---|---|
| **Performance** | Measures the system's response time and execution speed during real-time gesture control. The system detects gestures and responds within ~100–200 ms. Robot movement is smooth without noticeable lag. |
| **Usability** | Evaluates the ease of use, especially for children. The interface is simple with two control modes (gesture and manual). |
| **User Satisfaction** | Captures user feedback on the experience. Test users appreciated the novelty of controlling the robot with hand gestures and found it engaging and easy to use. |
| **Reliability** | Assesses the stability of the system over time. Continuous operation for 30+ minutes showed no crashes or servo failures. Gesture detection remained consistent without false triggers. |
| **Compatibility** | Confirms that the application works on different Android devices. The system was tested on several phones with different screen sizes and Android versions, all of which performed correctly. |

From this result, it could be concluded that all project objectives have been achieved and the development of the Arduino-based robotic arm has been completed successfully.

## 5.4 Summary

The testing phase confirmed that both software and hardware components function reliably under intended conditions. Functional testing ensured gesture recognition and robotic motion worked as expected, while non-functional testing confirmed usability, responsiveness, and Bluetooth range stability. These results verify that the project meets its goals and is ready for further enhancements or deployment.

# Chapter 6

## 6.1 Introduction

This chapter presents the overall achievements of the project titled "Development of Gesture-Controlled Robotic Arm for Kids", along with limitations encountered and suggestions for future enhancements. It also includes a comparison between the initial project objectives and the outcomes. Lastly, recommendations for future improvement and the overall contribution of the project are outlined.

## 6.2 Project Achievement

The main goal of this project was to develop an interactive robotic arm controlled via hand gestures, targeting children and young learners. The project aimed to serve as an educational tool, promoting early interest in robotics and technology.

The table below presents a comparison between the project objectives and the actual achievements.

Table 10: Project Achievement

| Objectives | Achievement |
|---|---|
| To develop a gesture recognition system using a smartphone camera to detect and interpret hand movements. | A gesture recognition system was successfully developed using MediaPipe and OpenCV within an Android application. Real-time hand landmarks were detected through the smartphone's camera. |
| To map the recognized gestures from the smartphone camera to specific commands for controlling the Arduino robotic arm movement | Gestures such as open palm, closed fist, and palm tilt were mapped to robot commands (e.g., gripper open/close, base rotation), which were sent as serial data via Bluetooth. |
| To integrate smartphone-based gesture recognition with Arduino robotic arm via a mobile phone application. | Integration was completed using Android Studio. The mobile app connected via Bluetooth (HC-05) to the Arduino board and successfully transmitted control commands to the robotic arm. |

All three objectives were achieved successfully, validating the functionality and usability of the developed system.

### 6.3 Project Limitation

While the project successfully achieved its primary objectives, several limitations impacted the overall system performance and flexibility. One notable constraint was the limited gesture complexity—the system faced challenges in accurately mapping intricate gestures to multiple robotic arm movements, such as simultaneously extending both the elbow and shoulder with a single gesture. Additionally, Bluetooth connection stability proved to be a concern; the system experienced disruptions when the device moved out of range or disconnected unexpectedly, and there was no automatic reconnection mechanism in place. Another issue arose under poor lighting conditions, where the camera and MediaPipe model exhibited reduced gesture recognition accuracy, sometimes resulting in incorrect interpretations. Moreover, hardware limitations affected performance; the servos used were prone to slight jitter, and their response lacked smoothness, which could potentially be improved with more advanced components or signal filtering techniques. While these limitations did not critically hinder the project's success, they present valuable opportunities for future refinement and enhancement.

## 6.4 Future Works

Several enhancements can be introduced to improve the system's overall performance, scalability, and user experience. One key improvement would be to expand the gesture recognition logic to support multi-joint movements, allowing more complex actions—such as simultaneous elbow and shoulder extensions—to be executed with a single gesture. To address connectivity issues, implementing a Bluetooth auto-reconnect feature within the Android app would ensure smoother operation by automatically recovering from disconnections without requiring a manual restart. For improved accessibility, voice control integration could complement gesture inputs, offering an inclusive solution for users with limited mobility. On the hardware side, upgrading to more precise servo drivers or applying signal filtering techniques could help eliminate jitter and enhance motion smoothness. Lastly, introducing an educational, gamified interface would make the system more engaging, particularly for children, by offering a fun and interactive way to learn robotics while controlling the robotic arm.

## 6.5 Contribution

This project offers several meaningful contributions across educational, technological, and social domains. As an educational tool, it provides an engaging and interactive way for children and beginners to explore the fundamentals of robotics, gesture recognition, and Arduino programming. By demonstrating real-world applications in a simplified and hands-on manner, it effectively promotes interest in STEM (Science, Technology, Engineering, and Mathematics) fields. The project's foundation on open-source technologies such as MediaPipe, OpenCV, and Arduino enhances its accessibility and adaptability, allowing students, educators, and hobbyists to build upon it for their own purposes. Moreover, it serves as a prototype for low-cost assistive systems, laying the groundwork for future development of gesture-controlled devices aimed at supporting individuals with physical disabilities.

## 6.5 Conclusion

In conclusion, the **Gesture-Controlled Robotic Arm for Kids** successfully met all intended objectives, combining software and hardware into an interactive, Bluetooth-controlled educational tool. Through real-time gesture recognition and servo motor response, users were able to manipulate the robotic arm effectively. Although there were limitations in gesture complexity and Bluetooth stability, the project proved to be a strong prototype that can be expanded upon. Future work may focus on enhancing the control methods, improving system robustness, and widening accessibility. This project not only achieved its goals technically but also holds significant value in promoting early STEM education and low-cost robotics.

## Reference


A. B. Pratomo and R. S. Perdana, "Arduviz, a visual programming IDE for arduino," *2017 International Conference on Data and Software Engineering (ICoDSE)*, Palembang, Indonesia, 2017, pp. 1-6, https://doi.org/10.1109/ICODSE.2017.8285871.

A.Sharma, J. Pathak, M. Prakash and J. N. Singh, "Object Detection using OpenCV and Python," *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, Greater Noida, India, 2021, pp. 501-505, https://doi.org/10.1109/ICAC3N53548.2021.9725638.

Brogårdh, T. (2007). Present and Future robot control Development—An industrial Perspective, Annual Reviews in Control. *ScienceDirect, Volume 31*(Issue 1), 69–7. https://doi.org/10.1016/j.arcontrol.2007.01.002

C. Chen, L. Chen, X. Zhou and W. Yan, "Controlling a robot using leap motion," *2017 2nd International Conference on Robotics and Automation Engineering (ICRAE)*, Shanghai, China, 2017, pp. 48-51, https://doi.org/10.1109/ICRAE.2017.8291351.

Casiez, Géry & Roussel, Nicolas & Vogel, Daniel. (2012). 1€ Filter: A Simple Speed-based Low-pass Filter for Noisy Input in Interactive Systems. Conference on Human Factors in Computing Systems - Proceedings. https://doi.org/10.1145/2207676.2208639

Cekova, K., Koceska, N., & Koceski, S. (2016). Gesture control of a mobile robot using Kinect sensor. *Proceedings of the AIIT 2016 Conference*, 251–258. https://doi.org/10.20544/AIIT2016.31

Gautam, A. (2024, November 26). Hand Detection Tracking in Python using OpenCV and MediaPipe. *Medium*. https://gautamaditee.medium.com/hand-recognition-using-opencv-a7b109941c88

Guna, J., Jakus, G., Pogačnik, M., Tomažič, S., & Sodnik, J. (2014). An analysis of the precision and reliability of the LeAP Motion sensor and its suitability for static and dynamic tracking. *Sensors*, *14*(2), 3702–3720. https://doi.org/10.3390/s140203702

Hashi, A. O., Hashim, S. Z. M., & Asamah, A. B. (2024). A systematic review of hand gesture recognition: An update from 2018 to 2024. IEEE Access, 12, 143599–143626. https://doi.org/10.1109/ACCESS.2024.3421992

Huda, N. (2023, October 1). *Malaysia is facing a decline in STEM intake*. Business Today. Retrieved October 8, 2024, from https://www.businesstoday.com.my/2023/10/16/malaysia-is-facing-a-decline-in-stem-intake/

Jayathilaka, C. (2020, July 30). *Agile Methodology*. Medium. Retrieved October 10, 2024, from https://medium.com/@chathmini96/agile-methodology-30ec4cdf3fc

Mohamad, A., Abdulbaqi, B., & Jumaa, N. (2017). Hand motion controlled robotic arm based on micro-electro-mechanical-system sensors: Gyroscope, accelerometer, and magnetometer. *Communication on Applied Electronics, 7.* https://doi.org/10.5120/cae2017652621

Pititheeraphab, Y., Chotikunnan, P., Thongpance, N., Kullathum, K., & Pintavirooj, C. (2016). Robot-arm control system using LEAP motion controller. In *Proceedings of BME-HUST 2016* (pp. 109–112). https://doi.org/10.1109/BME-HUST.2016.7782091

Sathiyanarayanan, Mithileysh & Azharuddin, Syed & Kumar, Santhosh & Khan, Gibran. (2014). GESTURE CONTROLLED ROBOT FOR MILITARY PURPOSE. International Journal For Technological Research In Engineering (IJTRE). 1. 2347-4718.https://www.researchgate.net/publication/265425730_GESTURE_CONTROLLED_ROBOT_FOR_MILITARY_PURPOSE

V. Chunduru, M. Roy, D. R. N. S and R. G. Chittawadigi, "Hand Tracking in 3D Space using MediaPipe and PnP Method for Intuitive Control of Virtual Globe," *2021 IEEE 9th*

*Region 10 Humanitarian Technology Conference (R10-HTC)*, Bangalore, India, 2021, pp. 1-6, https://doi.org/10.1109/R10-HTC53172.2021.9641587.

Vidyakar, V. (2023, August 1). *Robotics for kids*. Playto Labs. Retrieved October 8, 2024, from https://www.playtolabs.com/blog/robotics-for-kids

Y. A. Badamasi, "The working principle of an Arduino," *2014 11th International Conference on Electronics, Computer and Computation (ICECCO)*, Abuja, Nigeria, 2014, pp. 1-4, https://doi.org/10.1109/ICECCO.2014.6997578.

Yahya, Norzariyah & Maidin, Sarah & Safari, Muhammad. (2021). The Development of Interview Protocol to Explore Hybrid Agile Software Development Phases. International Journal of Advanced Trends in Computer Science and Engineering. 1639-1645. https://doi.org/10.30534/ijatcse/2021/221032021.