

COLLEGE OF ENGINEERING

BELLS UNIVERSITY OF TECHNOLOGY-NEW HORIZONS

# GESTURE CONTROLLED ROBOT

BY

ELECTRICAL/ELECTRONICS ENGINEERING

Group 10

20TH OF JANUARY 2025.

ROBOTICS 1

(ICT 215)

SUBMITTED TO

AYUBA MUHAMMAD

Group Members:

OGEDEGBE VICTORY OSOSE 2023/12159

ADEGBAMI FREDRICK 2023/12641

AKHABUE OSEMUDIAMHEN 2023/12755

AJAERO DANIEL 2023/12386

OLUWUNMI DAVID 2023/12615

DECLARATION

We hereby declare that this is our group original work of the project design reflecting the knowledge acquired from research on my robotics project about “Gesture controlled robot”. I therefore declare that the information in this report is original and has never been submitted to any other institution, university or college for any award.

Name: …………………………………………………………………………………………………………………

Matric No: …………………………………………………………………………………………………………..

Signature: ……………………………………………………………………………………………………………

Date: …………………………………………………………………………………………………………………..

Name: …………………………………………………………………………………………………………………

Matric No: …………………………………………………………………………………………………………..

Signature: ……………………………………………………………………………………………………………

Date: …………………………………………………………………………………………………………………..

Name: …………………………………………………………………………………………………………………

Matric No: …………………………………………………………………………………………………………..

Signature: ……………………………………………………………………………………………………………

Date: …………………………………………………………………………………………………………………..

Name: …………………………………………………………………………………………………………………

Matric No: …………………………………………………………………………………………………………..

Signature: ……………………………………………………………………………………………………………

Date: …………………………………………………………………………………………………………………..

Name: …………………………………………………………………………………………………………………

Matric No: …………………………………………………………………………………………………………..

Signature: ……………………………………………………………………………………………………………

Date: …………………………………………………………………………………………………………………..

APPROVAL

I have heard and hereby recommended this project design entitled “Gesture controlled robot” acceptance of Bells University of Technology in the partial fulfillment of my group new horizon project.

………………………………………………………………………………….

Ayuba Muhammad

Lecturer

ACKNOWLEDGEMENT

We would like to thank our project supervisor for his guidance Mr. Ayuba Muhammad for his enormous co-operation and guidance.

We have no words to express our gratitude for a person who wholeheartedly supported the project and gave freely of his valuable time while making this project. The technical guidance provided by him was more than useful and made the project successful. We also thankful to you for guiding us to develop a very good project idea. Finally, we would also like to thank our dear classmates of my college and friends who guided and helped while working on our project.

DEDICATION

We dedicate this project to our project supervisor and diligent lecturer who gives us the reason to work harder and think outside of what we are studying. He has been and will always been a thoughtful lecturer. We also dedicate the project to LORD almighty because without God we wouldn’t have been able to go through the whole process of the project.

TABLE OF CONTENTS

DECLARATION……………………………………………………………………………………………………2

APPROVAL………………………………………………………………………………………………………….3

ACKNOWLEDGEMENT………………………………………………………………………………………..4

DEDICATION……………………………………………………………………………………………………….5

LIST OF FIGURES…………………………………………………………………………………………………9

ABSTRACT…………………………………………………………………………………………………………..10

CHAPTER ONE.........................................................................................................11

1. Introduction…………………………………………………………………………………………………11
   1. Background of the project……………………………………………………………………………12
   2. Problem Statement………………………………………………………………………………………13
   3. The first usage of the project…………………………………………………………………...….14
   4. Technological advancement on project till date…………………………………………..15
   5. Objective of the project……………………………………………………………………………….16
      1. Main objectives…………………………………………………………………….……….16
      2. Specific objectives………………………………………………………………………...16
   6. Research question……………………………………………………………………….………………16
   7. Significance of the project…………………………………………………………………………...16
   8. Scope of the study……………………………………………………………………………………….17
      1. Context scope……………………………………………………………………………….17
      2. Geographical scope……………………………………………………………………….17
      3. Time scope……………………………………………………………………………………17

CHAPTER TWO…………………………………………………………………………………………………..18

1. Introduction………………………………………………………………………………………………….18
   1. Arduino uno as a board……………………………………………………………………………….19-22
   2. Arduino uno as an IDE…………………………………………………………………………………23-25
   3. L298 motor………………………………………………………………………………………………….26-28
   4. Resistor (Potentiometer)……………………………………………………………………………..29
   5. DC motor…………………………………………………………………………………………………….30-31

CHAPTER THREE…………………………………………………………………………………………………..32

METHODOLOGY…………………………………………………………………………………………………..32-33

3.0 Arrangements of components……………………………………………………………………….34

3.1 Connections….……………………………………………………………………………………………….35-37

3.2 Arduino uno code for project………………………………………………………………………...38-39

CHAPTER FOUR…………………………………………………………………………………………………...40

4.0 Results of the system…………………………………………………………………………………….40

4.1 Stimulation log of system……………………………………………………….........................41

4.2 Common errors that can be encountered in a gesture controlled robot………..42-45

4.3 Solution to the common errors……………………………………………………………………..45-54

CHAPTER FIVE………………………………………………………………………………………………………55

5.0 Conclusion……………………………………………………………………………………………………..55

5.1 Recommendation…………………………………………………………………………………………..55

5.2 References……………………………………………………………………………………………………..55

LIST OF FIGURES

Figure 1: Arduino uno board……………………………………………………………21

Figure 2: L298 motor……………………………………………………………………….27

Figure 3: Potentiometer………………………………………………………………….28

Figure 4: DC motor………………………………………………………………………….30

Figure 5: Circuit connection…………………………………………………………….37

ABSTRACT

This work proposes constructive design of hand gesture control robot. This system acts as a channel between the human and the robot through physical change such as tilting of hand which is just using hand gesture to control the robot instead of using objects to control the robot. The robot operates in multiple modes, forward, backward, left, right and stop, ensuring smooth control.

1.0 INTRODUCTION

Wireless communication system form the backbone of model-day robotics control. The main reason wireless control possess over wired control is that they provide a much broader range for the robot to interacts with its environment. External peripherals are usually required in order to wirelessly transmit data to the robot; however control schemes that don’t require the use of any external device are yet to be brought into the main stream by the robotics community at large.

We purpose the implementation of robot that can be physically controlled via hand gesture. It uses highly sensitive sensors, such as accelerometers and cameras, which made it possible to interpret and detect human gestures accurately. Gesture controlled robot can be used in different fields such as medical fields for precise operations. It can also be used for disaster responses like in hazardous environment where humans cannot physically intervene. Gesture recognition is a touchless technology that allows devices to understand and respond to human movements as commands. Gesture recognition technology is integrated into a variety of appliances and devices, most commonly known are - the Microsoft Kinect for Xbox and Play Station games have used gesture recognition. In this article, we are going to discuss gesture recognition.

We intend on accomplishing this through a proteus software for designing the circuit and an arduino software for coding the project.

1.1 BACKGROUND OF PROJECT

Gesture controlled robots is a major human-computer interaction innovation that gives users more intuitive and natural ways to operate machines. Gesture control is the interpretation of human hands or body movement as commands that guide the action of a robot. It opens up a lot of opportunities and can be utilized in many fields, such as robotics, remote controlled devices, assistive technology, industrial automation, consumer robots, and much more.

There was an innovation in sensor, image processing and machine learning algorithms which have enabled the concept of gesture control. These mainly involve protyping using accelerometers, gyroscopes, infrared sensors as well as cameras with computer vision capabilities. These innovations allow robots to understand gestures in a very precise and responsive manner, adding levels of adaptability to the movement of machines in changing environments.

This project involves designing a robot that responds to the user commands in real time using gestures. The robot interprets specific gestures, such as movement or task execution, by interpreting input from sensors or cameras. The goal is to show the capabilities of gesture control on robotics projects where usability, accuracy, and efficiency play a major role in complex robotic implementations.

This report covers the process of designing and creating the robot, especially highlighting the hardware and software aspects. The paper also addresses the challenges and limitations associated with gesture recognition technology and proposes potential enhancements for future applications.

1.2 PROBLEM STATEMENT

Traditionally robots were controlled with manual input devices such as keyboards, joysticks or remote controls. These methods were too stressful, cumbersome and challenging to use in dynamic or high-stress environments.

To address this, the gesture controlled robot was introduce which leverages human gestures as a natural form of communication. It enhances users experience, improve control efficiency and open up new possibilities in applications like healthcare, industrial automation and so on.

However, developing this robust gesture-controlled system comes with different challenges, including accurately interpreting gestures in real-time, minimizing errors in diverse environments, ensuring seamless communication between the user and the robot, and achieving reliable performance across varying user profiles and lighting conditions. This project aims to design and implement a gesture-controlled robot that addresses these challenges while offering a user-friendly and efficient control interface.

1.3 FIRST USAGE OF GESTURE CONTROLLED ROBOT

One of the notable early instances was in the 1980s and 1990s when researchers started developing systems for robotic arms in industrial applications, using gestures to simplify tasks like assembly or material handling. Another milestone was the integration of glove-based input devices, such as the "DataGlove" developed in 1987, which allowed users to control robotic systems through hand and finger movements.

The history of hand gesture recognition for computer control started with the invention of glove-based control interfaces. Researchers realized that gestures inspired by sign language can be used to offer simple commands for a computer interface. This gradually evolved with the development of much accurate accelerometers, infrared cameras and even fibreoptic bend-sensors (optical goniometers). Some of those developments in glove based systems eventually offered the ability to realize computer vision based recognition without any sensors attached to the glove. These early efforts laid the foundation for modern gesture controlled robots, which now use advanced technologies like computer vision, machine learning, and wearable devices to achieve greater accuracy and functionality.

1.4 TECHNOLOGICAL ADVANCEMENT ON GESTURE CONTROLLED ROBOT TILL DATE.

Gesture-controlled robots have seen significant advancements over the years, driven by innovation in sensor technology, human- machine interaction and artificial intelligence. Early gesture-controlled robot used sensors like accelerators and gyroscope to detect hand movement. These has been advanced into wearable devices like gloves.

Modern robots now utilize cameras and vision-based algorithms to recognize gestures without requiring wearables. Advanced algorithms train robots to interpret gestures in real-time, making systems adaptable to different users and environments.

1.5 OBJECTIVE OF PROJECT

1.5.1 Main objectives

The main objectives of this project is to design and implement a gesture controlled robot using various form of technology such as Arduino uno, DC motor driver, Potentiometer, L298 motor.

1.5.2 Specific objectives

1. To carry out the benefit of using gesture control robot.

2. To simplify robotic operation, especially for non-technical users.

3. To foster human-robot communication for a variety of applications.

4. To allow users to perform specific tasks e.g., picking objects.

1.6 Research Questions

1. How can gesture recognition be implemented effectively using sensors or cameras?

2. What is the most suitable combination of hardware and software for gesture detection and processing?

3. What are the challenges in real-time recognition, and how can they be migrated?

1.7 Significance of project

Since the project lies with controlling robots with gestures. If we design and introduce more robots that can be controlled by gestures we would be able to reduce stress and make the usage of robot accessible to non- technical users. Therefore, this system/project would be easy to maintain and promote human- robot relationship.

1.8 Scope of study

1.8.1 Context scope

The study will cover the implementation of gesture controlled robot using proteus software to design it

The function of the project is to improve human- robot relationship by having a robot being controlled by human gestures.

1.8.2 Geographical scop

The study would be conducted in any offices, hospitals, homes, business areas etc. around Nigeria.

1.8.3 Time scope

The project is based on theoretical and methodological data, thus it is approximated to take a maximum of 1 month.

CHAPTER TWO

2.0 Introduction

This chapter delves into the fundamental components and tools utilized in the development of a gesture-controlled robot. The discussion begins with the Arduino Uno, highlighting its role as a versatile microcontroller board and is Integrated Development Environment (IDE) for programming. Key components such as L298 motor driver, potentiometers, and DC motors are also explored, providing insights into their functionality and integration. By understanding these elements, we establish a solid foundation for designing and implementing the robot’s control and movement mechanisms.

2.1 Arduino uno as a board

The Arduino Uno is one of the most popular microcontroller boards in the Arduino family, designed to provide a simple yet powerful platform for prototyping and development. It is built around the ATmega328P microcontroller, which combines processing capabilities with a range of interfacing options, making it ideal for beginners and advanced users alike.

Features and Specifications

The Arduino Uno features:

14 digital input/output pins, 6 of which support PWM (Pulse Width Modulation). These pins are commonly used to control devices like LEDs, servo motors, and communication modules.

6 analog input pins, used to read analog signals from sensors such as potentiometers, temperature sensors, or gesture recognition modules.

A USB interface for uploading code and communication with a computer.

A DC power jack and an onboard voltage regulator to ensure stable operation when powered externally (7–12V recommended).

Communication protocols: UART (TX/RX), SPI, and I2C, enabling the board to communicate with other devices such as motor drivers, sensors, or wireless modules.

Pin Configuration in the Project

In the gesture-controlled robot, the Arduino Uno pins are allocated as follows:

Digital Pins:

Pins 2 and 3: Configured for interrupt-based inputs (e.g., for gesture recognition or motion sensors).

Pins 9–11: Used for PWM control of motors through the L298 motor driver.

Analog Pins:

A0: Connected to the potentiometer for fine-tuning motor speed or gesture calibration.

A1–A3: Connected to additional sensors (e.g., ultrasonic or IR sensors for obstacle detection).

Communication Pins (TX/RX): Used to establish communication with wireless modules like Bluetooth or RF transceivers if remote control is incorporated.

Libraries Used

To simplify development and ensure efficient integration of the Arduino Uno with other components, several libraries are employed:

Servo.h: For controlling servo motors, which might be used for precise gesture-based actions.

AccelStepper.h: For smoother motor control when working with stepper motors.

AFMotor.h: For controlling DC motors through the L298 motor driver.

Wire.h: To facilitate I2C communication with other devices such as accelerometers or gyroscopes used for gesture detection.

SoftwareSerial.h: For creating additional serial communication ports, especially for Bluetooth or wireless modules.

Role in the Gesture-Controlled Robot

In this project, the Arduino Uno acts as the brain of the gesture-controlled robot. It reads data from input devices like accelerometers or gesture recognition modules and processes this information using pre-written algorithms. Based on the processed data, it generates appropriate control signals to drive output devices such as DC motors, LEDs, or buzzers.

The Arduino Uno’s robust design, extensive library support, and ease of use make it an essential part of the project, providing real-time control and facilitating seamless interaction between the hardware components.

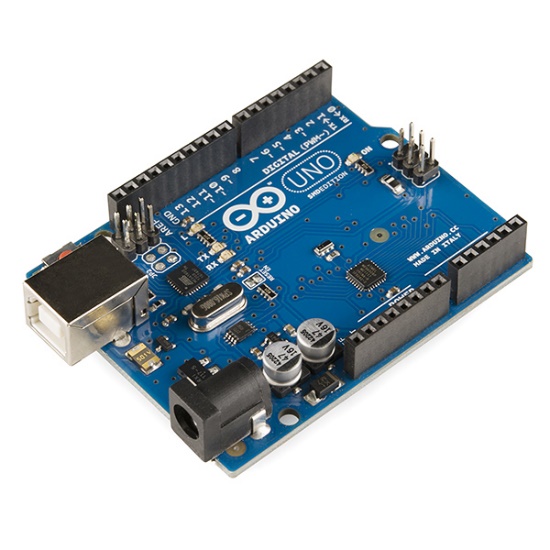


Figure 1: Arduino uno board

2.2 Arduino uno as an IDE

The Arduino Integrated Development Environment (IDE) is an essential tool for programming and interacting with the Arduino Uno board. It serves as a user-friendly platform for writing, compiling, and uploading code to the microcontroller. The Arduino IDE combines simplicity with powerful features, making it ideal for both beginners and advanced users.

Key Features of the Arduino IDE

1. Cross-Platform Compatibility:

The IDE is compatible with Windows, macOS, and Linux operating systems, allowing developers to work seamlessly across various platforms.

2. Simple Text Editor:

The IDE provides a straightforward editor where code is written in a simplified version of C/C++ using the Arduino programming language. This abstraction reduces the complexity of traditional microcontroller programming.

3. Built-In Compiler and Debugger:

The IDE includes a built-in compiler that converts user-written sketches into machine-readable code, along with basic debugging features to identify and fix errors.

4. Code Examples and Libraries:

The IDE includes a vast collection of built-in libraries and example codes for different sensors, modules, and components. These resources save development time and simplify coding for hardware interaction.

5. Serial Monitor:

The IDE provides a Serial Monitor tool, enabling developers to communicate with the Arduino board in real-time. It is useful for debugging and monitoring data from connected sensors or modules.

6. Board and Port Selection:

The IDE allows users to select the specific Arduino board (e.g., Arduino Uno) and the correct communication port for uploading code.

Arduino IDE Workflow

The Arduino IDE simplifies the process of programming the Arduino Uno board through the following workflow:

1. Writing the Code:

The user writes a program (called a “sketch”) in the editor. For example, code to control a DC motor based on gesture input is written in the IDE.

2. Compiling the Code:

The IDE compiles the sketch to check for errors and converts it into a format that the Arduino Uno can understand.

3. Uploading the Code:

Using a USB cable, the compiled code is uploaded to the Arduino Uno board. The board executes the uploaded program to perform specific tasks.

4. Debugging and Monitoring:

The Serial Monitor allows the user to receive real-time data from sensors or to send commands to the board for testing purposes.

Libraries Used in This Project

In the gesture-controlled robot project, the Arduino IDE is used to integrate and manage several libraries that simplify hardware interaction:

Wire.h: Facilitates communication with I2C-based components such as accelerometers.

AccelStepper.h: Used for controlling stepper motors if precise motion is required.

Servo.h: Controls servo motors for additional functionalities like arm movements.

AFMotor.h: For motor driver communication.

SoftwareSerial.h: Enables communication with external modules like Bluetooth or RF modules.

Role in the Gesture-Controlled Robot

The Arduino IDE plays a critical role in the development of the gesture-controlled robot by providing an efficient platform for writing and managing the code that controls the robot. It enables real-time programming of the Arduino Uno, simplifying tasks such as reading input from gesture sensors, processing the data, and controlling the robot’s actuators and motors.

By leveraging the Arduino IDE, developers can focus on algorithm development and hardware interaction without worrying about low-level programming complexities. Its ease of use, combined with extensive library support, ensures the smooth development and operation of the robot.

2.3 L298 motor

The L298 motor driver is a high-power dual H-bridge motor driver designed to control the speed and direction of DC motors and stepper motors. It is widely used in robotics and automation projects due to its ability to handle high currents and its compatibility with microcontrollers like the Arduino Uno.

Features and Specifications

1. Dual H-Bridge Design:

The L298 contains two H-bridges, enabling it to control two DC motors independently or one stepper motor. Each H-bridge allows for bidirectional motor control.

2. Voltage and Current Ratings:

Input voltage: 5V to 46V.

Maximum output current: Up to 2A per channel (with appropriate heatsinking).

Logic voltage: 5V, making it compatible with the Arduino Uno.

3. Built-In Protection:

Integrated diodes for back-emf protection to safeguard the circuit and connected components.

Overheating protection (requires an external heatsink for high-current applications).

4. Control Pins:

Enable Pins: Used to turn the motor on/off.

Input Pins: Control the direction of motor rotation.

PWM Capability: Allows motor speed control by supplying a pulse-width modulation (PWM) signal.

Working Principle

The L298 motor driver works on the H-bridge principle, which allows current to flow in both directions through a motor, enabling forward and reverse motion. The input pins of the driver receive logic signals from the Arduino Uno, which determine the direction and speed of the motor. By varying the PWM signal, the motor's speed can be adjusted.

Pin Configuration

A typical L298 motor driver module includes the following pins:

Logic Pins (IN1, IN2, IN3, IN4): Used to control the direction of two motors.

Enable Pins (ENA, ENB): Used to enable/disable the motors. These pins also accept PWM signals for speed control.

Power Pins (Vcc, GND): Provide power to the motors and logic circuitry.

Output Pins (OUT1, OUT2, OUT3, OUT4): Connected to the motor terminals.

Role in the Gesture-Controlled Robot

In the gesture-controlled robot, the L298 motor driver serves as the interface between the Arduino Uno and the DC motors. The Arduino sends control signals to the driver, which then:

Determines the motor's direction by setting the appropriate logic on the input pins (e.g., IN1 and IN2).

Adjusts the motor’s speed by providing a PWM signal to the enable pins (ENA or ENB).

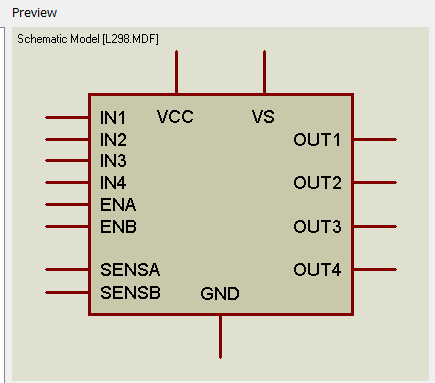


Figure 2: L298 motor

2.4 Resistor (Potentiometer)

A potentiometer is a type of variable resistor. It allows for adjustable resistance and is often used in circuit to vary voltage levels. It can be applied as sensors for control panels. The potentiometer can be used to adjust certain parameters, such as controlling the sensitivity of the gesture inputs or calibrating the response of robot.

By adjusting the wiper position, the resistance can be varied, which may influence how the robot interprets input signals or controls its motion. The potentiometer is connected to the Arduino Uno board, allowing for real-time adjustments in the robot's operation. The Arduino reads the variable resistance as an analog input, converting it into a digital signal to control the robot. It is used for tasks like adjusting the speed of the DC motor or fine-tuning other motor control parameters via the L298 motor driver.

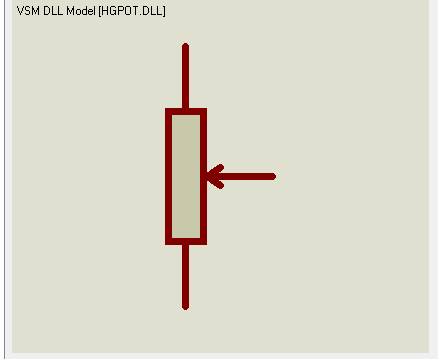


FIGURE 3: POTENTIOMETER

2.5 DC MOTOR

A DC motor is an electric motor that runs on direct current (DC) electricity. It converts electrical energy into mechanical energy, producing rotational motion. DC motors are commonly used in various applications, including robotics, due to their simplicity, ease of control, and reliability.

Components of a DC Motor:

1. Stator: The stationary part of the motor that provides a magnetic field. It usually contains magnets or windings that create a magnetic field when energized.

2. Rotor (Armature): The rotating part of the motor. It consists of windings through which current flows, generating a magnetic field that interacts with the stator's magnetic field.

3. Commutator: A mechanical switch that reverses the direction of current in the rotor windings, ensuring continuous rotation.

4. Brushes: Conductive contacts that transfer current to the rotor through the commutator.

Working Principle:

When direct current flows through the motor's windings, a magnetic field is created around the rotor.

This field interacts with the stator's magnetic field, causing the rotor to rotate.

The commutator and brushes ensure the current direction in the rotor windings changes at the right time, maintaining continuous rotation.

DC motors can be easily controlled for speed and direction using PWM or an H-bridge circuit like the L298 motor driver

In gesture controlled robot, the DC motor is used to move the robot or perform task in response to gestures. Its speed and direction could be adjusted using signals from the Arduino board, providing precise control over the robot’s movements.

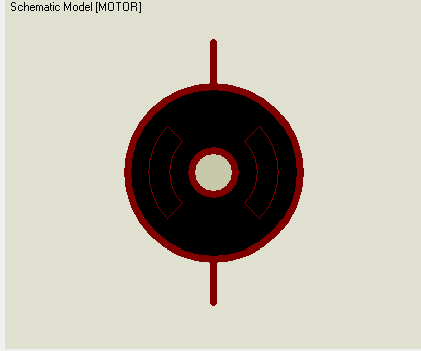
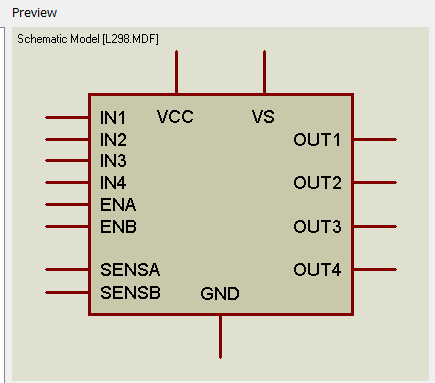
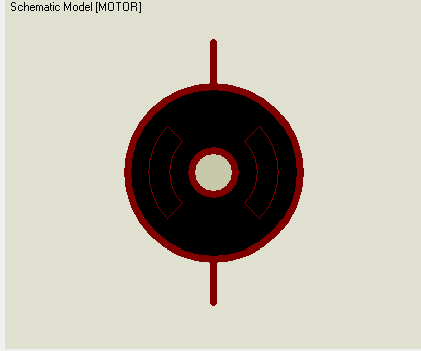


FIGURE 4: DC Motor

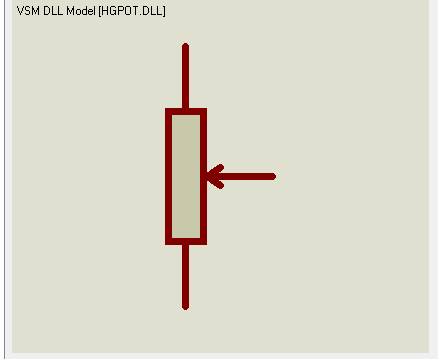
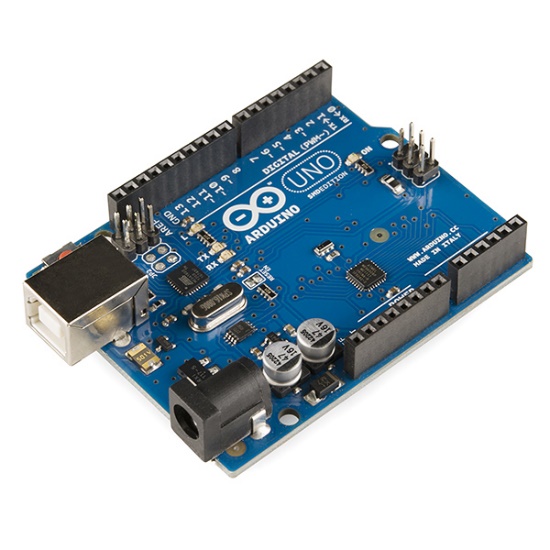
CHAPTER THREE

METHODOLOGY

The robot that can be controlled using human gesture is designed and built. The robot is intended to solve assistive technology, remote control, or automation. The application program was developed using C++ programming language. The component used consist of L298, Motor, Arduino uno board, POT-HG. They can all be shown in the figure below.

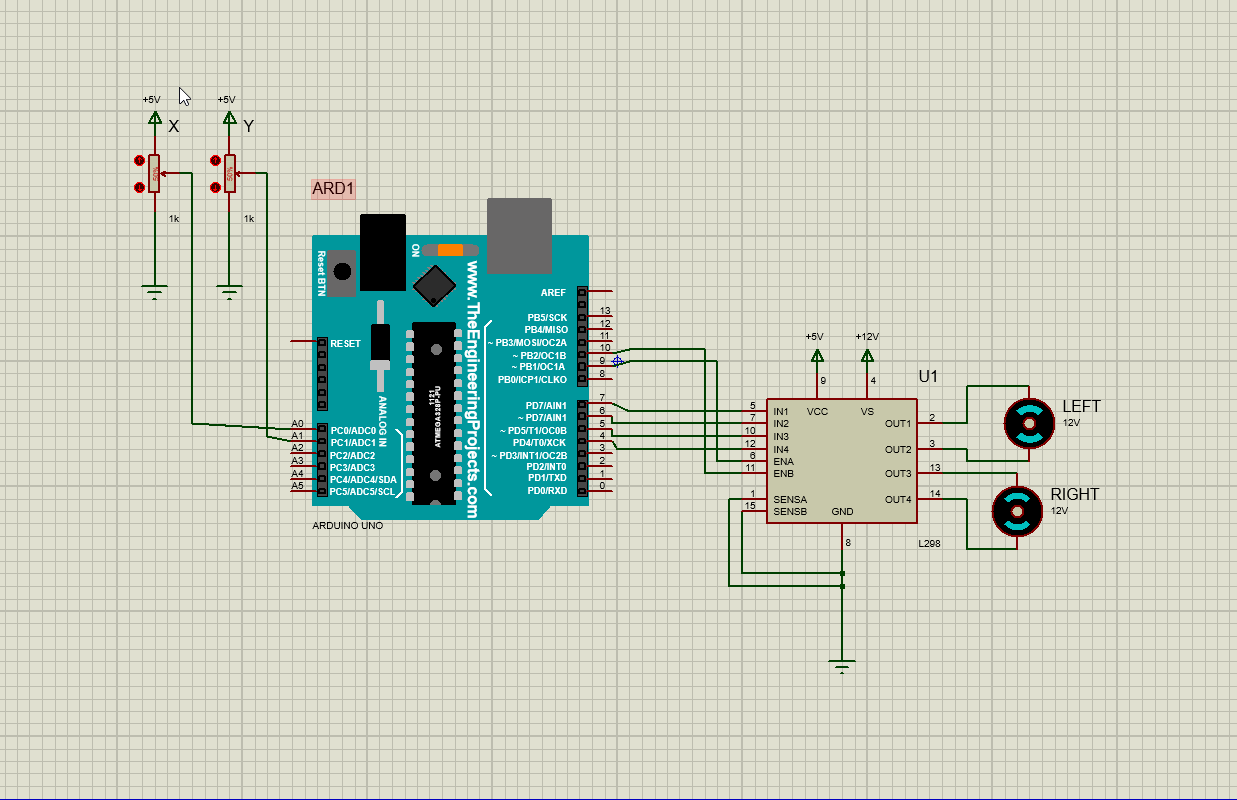
 

L298 Motor driver

POT-HG Arduino uno board

CIRCUIT DIAGRAM OF THE GESTURE CONTROLLED ROBOT

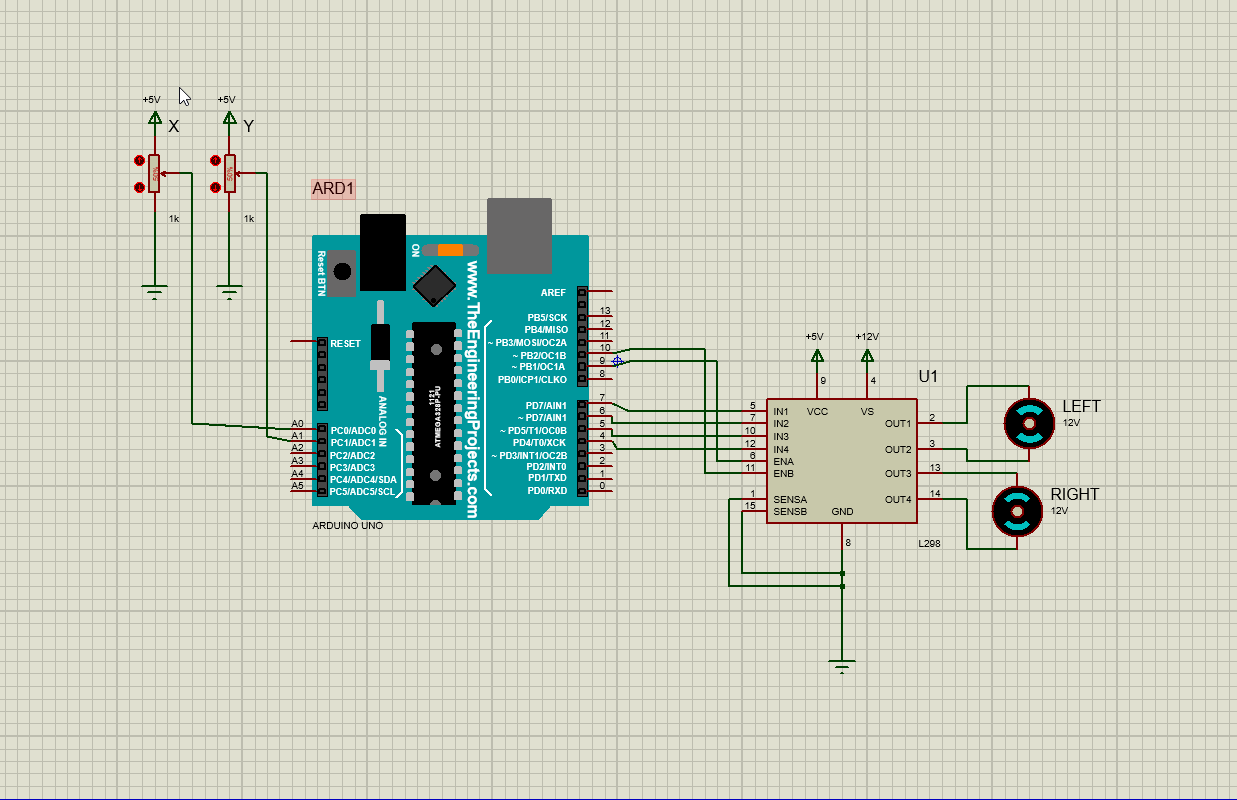


3.0 ARRANGEMENT OF COMPONENTS

In the proteus library, we go to pick parts and select all the necessary components which are the of L298, Motor, Arduino uno board, POT-HG. The Arduino uno is then placed in the work space the L298 is then placed next to the board. After the DC motor driver is placed close to the L298. The potentiometer (POT HG) is then positioned at the left side of the Arduino uno board along with two grounds. Then another ground is then placed at the buttom of L298. Although, some Arduino board as ground in it but the one used in the project doesn’t have a ground that why a ground was introduced.

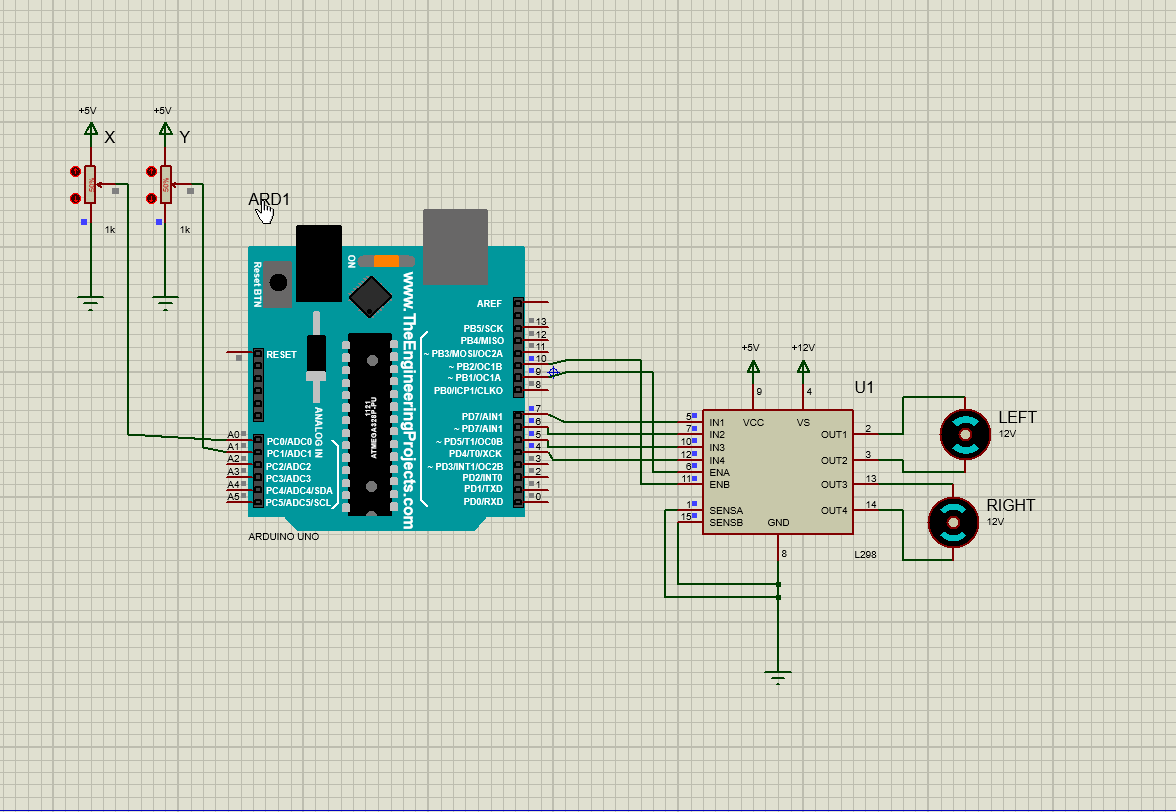
3.1 CONNECTIONS/PROCESS OF PROJECT

The Arduino uno is placed anywhere in the work space. The potentiometer is then placed at left top corner of the Arduino board the wiper of the potentiometers x and y is then connected to A0 and A1 of the arduino board then the bottom side potentiometer x and y are connected to grounds each while the top side of the potentiometer is connected to an output labelled x and y with voltage source 5V each. L298 is placed at the right side of the Arduino uno. IN1, IN2, IN3, IN4, ENA and ENB is connected to pin 7, 6, 5, 4, 10 and 9 of the Arduino board respectively. An external ground is then connected to the L298 ground. SENSA and SENSB are then connected to the grounds. Connect outputs to VCC and VS with voltage source 5V and 12V respectively. Then two motor drivers of 12V are placed at the right side of the L298 with the driver at the top labelled left and the bottom driver labelled right. Then out 1 is connected to the top of the left driver and out 2 was connected the bottom of the left driver. Out 3 and Out 4 are connected to the top and bottom of the right driver respectively.



END RESULT OF CIRCUIT CONNECTION

The changing of the percentage of the wiper controls the movement of the motor driver. It sense the movement or gesture and the motor driver move in the direction of the gesture. If POT Y gesture move upward and POT X is still in It position the motor will turn in the right direction. The more POT Y is increased the more the speed of the motor in the right direction. If POT Y is in 50% and POT X is reduced the right motor would turn in the left direction and the left motor would remain stagnant and if POT X is increased the right motor would remain stagnant and the left motor would turn in the left direction. If POT X and POT Y is increased the both motor would move in the right direction while if POT X and Y are decreased both motors would move in left direction but the right motor would be faster.



3.2 Arduino code of the project

#define ENA 9   // PWM pin for left motor

#define ENB 10  // PWM pin for right motor

#define IN1 7   // Left motor direction 1

#define IN2 6   // Left motor direction 2

#define IN3 5   // Right motor direction 1

#define IN4 4   // Right motor direction 2

#define POT\_X A0 // POT-HG for forward/backward

#define POT\_Y A1 // POT-HG for left/right

void setup() {

  pinMode(IN1, OUTPUT);

  pinMode(IN2, OUTPUT);

  pinMode(IN3, OUTPUT);

  pinMode(IN4, OUTPUT);

  pinMode(ENA, OUTPUT);

  pinMode(ENB, OUTPUT);

}

void loop() {

  int xVal = analogRead(POT\_X); // Read forward/backward input

  int yVal = analogRead(POT\_Y); // Read left/right input

  int motorSpeedX = map(abs(xVal - 512), 0, 512, 0, 255); // Map X-axis to speed

  int motorSpeedY = map(abs(yVal - 512), 0, 512, 0, 255); // Map Y-axis to speed

  // Forward/backward motion (both motors)

  if (xVal > 512) { // Forward

    digitalWrite(IN1, HIGH);

    digitalWrite(IN2, LOW);

    analogWrite(ENA, motorSpeedX); // Left motor PWM

    digitalWrite(IN3, HIGH);

    digitalWrite(IN4, LOW);

    analogWrite(ENB, motorSpeedX); // Right motor PWM

  } else if (xVal < 512) { // Backward

    digitalWrite(IN1, LOW);

    digitalWrite(IN2, HIGH);

    analogWrite(ENA, motorSpeedX); // Left motor PWM

    digitalWrite(IN3, LOW);

    digitalWrite(IN4, HIGH);

    analogWrite(ENB, motorSpeedX); // Right motor PWM

  } else { // Stop

    analogWrite(ENA, 0);

    analogWrite(ENB, 0);

  }

  // Left/right turning adjustments

  if (yVal > 512) { // Turn right

    analogWrite(ENA, motorSpeedY); // Left motor keeps moving

    analogWrite(ENB, 0);          // Right motor stops

  } else if (yVal < 512) { // Turn left

    analogWrite(ENA, 0);          // Left motor stops

    analogWrite(ENB, motorSpeedY); // Right motor keeps moving

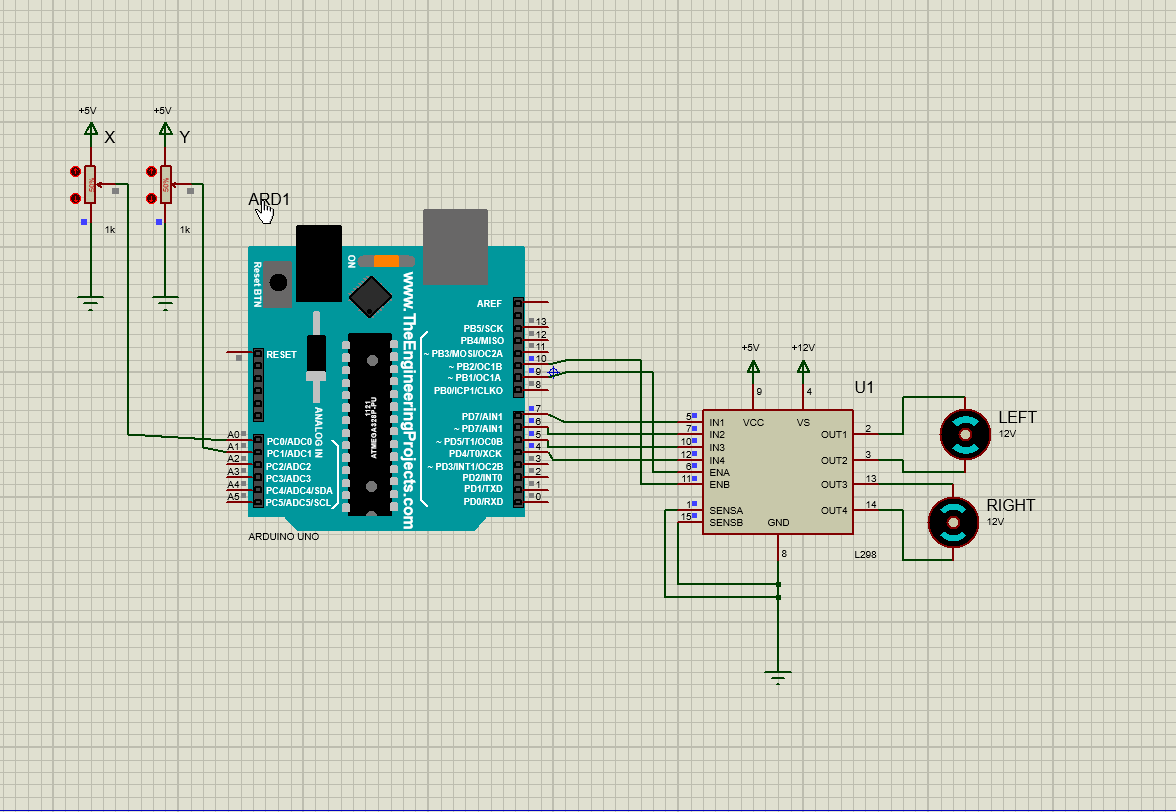
  }

}

CHAPTER FOUR

4.0 RESULT OF THE PROJECT

At the end of the project the motor is meant to move as the potentiometer change position. The potentiometer acts as the sensor. The result of the project can be shown in the image below.



4.1 STIMULATION LOG OF PROJECT

PROSPICE 8.13.00 (Build 32709) (C) Labcenter Electronics 1993-2023.

Loaded netlist 'C:\Users\ACER\AppData\Local\Temp\LISA8933.SDF' for design 'Gesture Controlled Robot.pdsprj'

AVR Release 8.3SP0 build 33337 for ATMEGA328P. [ARD1]

Loading HEX file 'Gesture Controlled Robot Ino files\gesturecontrolledrobot\build\arduino.avr.uno\gesturecontrolledrobot.ino.hex'. [ARD1]

Read total of 1338 bytes from file 'Gesture Controlled Robot Ino files\gesturecontrolledrobot\build\arduino.avr.uno\gesturecontrolledrobot.ino.hex'. [ARD1]

4.2 COMMON ERRORS THAT CAN BE ENCOUNTED WHILE DESIGNING A GESTURE CONTROLLED ROBOT

1. Sensor Errors

Inaccurate Sensor Readings: Due to noise, interference, or poor sensor quality.

Calibration Errors: Miscalibrated sensors leading to incorrect data interpretation.

Limited Range: Sensors failing to detect gestures outside their field of view or range.

2. Gesture Recognition Errors

Misinterpretation of Gestures: Similar gestures being confused or incorrectly recognized.

Incomplete Gesture Detection: Failure to recognize gestures due to incomplete or partial movements.

Slow Recognition: High latency in processing gestures causing delays in robot response.

3. Hardware Errors

Actuator Malfunction: Failure in robot actuators leading to incorrect or incomplete movements.

Power Supply Fluctuations: Unstable power supply causing erratic robot behavior.

Mechanical Wear and Tear: Degradation of components over time affecting performance.

4. Software Errors

Algorithmic Bugs: Errors in the gesture recognition algorithms causing incorrect responses.

Communication Lag: Delays in communication between sensors, processors, and actuators.

Overfitting/Underfitting: Machine learning models failing to generalize gestures properly.

5. Environmental Errors

Lighting Conditions: Poor or variable lighting affecting vision-based gesture recognition.

Obstructions: Physical objects blocking sensors and disrupting gesture detection.

Background Noise: Interference in audio-based gesture systems due to ambient noise.

6. User Errors

Inconsistent Gesture Execution: Variability in how users perform gestures leading to recognition issues.

Unintended Gestures: Accidental or unintended movements being misinterpreted as commands.

User Fatigue: Leading to incomplete or incorrect gesture execution.

7. Integration Errors

Component Compatibility Issues: Incompatibility between different hardware and software components.

Synchronization Problems: Lack of synchronization between gesture input and robot actions.

8. System Stability Errors

System Crashes: Software crashes causing the robot to stop responding.

Overheating: Prolonged operation leading to overheating of components.

Memory Overload: Excessive data processing leading to system slowdowns or crashes.

4.3 SOLUTION TO THIS ERRORS

1. Sensor Errors

Inaccurate Sensor Readings:

Use higher-quality sensors with better accuracy.

Implement noise filtering techniques to reduce interference.

Calibration Errors:

Regularly calibrate sensors before each use.

Use automated calibration routines to maintain accuracy.

Limited Range:

Use wide-range or multiple sensors to cover a larger detection area.

Optimize sensor placement to maximize field of view.

2. Gesture Recognition Errors

Misinterpretation of Gestures:

Improve gesture recognition algorithms by training on a larger, more diverse dataset.

Implement redundancy checks to verify gesture accuracy.

Incomplete Gesture Detection:

Use predictive algorithms to infer incomplete gestures.

Guide users with visual or auditory feedback to complete gestures correctly.

Slow Recognition:

Optimize algorithm efficiency to reduce processing time.

Use faster hardware or parallel processing to improve response time.

3. Hardware Errors

Actuator Malfunction:

Perform regular maintenance and diagnostics on actuators.

Use robust, high-quality actuators designed for the specific robot's requirements.

Power Supply Fluctuations:

Use a stable and regulated power supply with surge protection.

Include backup power solutions like batteries or capacitors.

Mechanical Wear and Tear:

Implement regular maintenance schedules to check and replace worn parts.

Use durable materials for critical mechanical components.

4. Software Errors

Algorithmic Bugs:

Conduct thorough testing and debugging of the software.

Use version control and code reviews to catch and fix bugs early.

Communication Lag:

Optimize communication protocols to reduce latency.

Use high-speed communication channels or dedicated hardware for critical data transfer.

Overfitting/Underfitting:

Train machine learning models on diverse and representative datasets.

Regularly update models with new data to improve generalization.

5. Environmental Errors

Lighting Conditions:

Use infrared sensors or other technologies less affected by lighting variations.

Implement adaptive algorithms that can adjust to changing light conditions.

Obstructions:

Use multiple sensors or cameras from different angles to mitigate obstructions.

Implement obstacle detection and avoidance systems.

Background Noise:

Use noise-canceling techniques or directional microphones for audio-based systems.

Implement signal processing to filter out background noise.

6. User Errors

Inconsistent Gesture Execution:

Provide user training or guidance on how to perform gestures correctly.

Use adaptive learning systems that can adjust to individual user variations.

Unintended Gestures:

Implement confirmation steps or feedback loops before executing critical commands.

Use context-aware systems to differentiate between intentional and accidental gestures.

User Fatigue:

Design gestures to be simple and less physically demanding.

Provide regular rest breaks or alternative input methods.

7. Integration Errors

Component Compatibility Issues:

Ensure compatibility of hardware and software components during the design phase.

Use standardized protocols and interfaces for easier integration.

Synchronization Problems:

Implement synchronization mechanisms to ensure coordinated operation.

Use real-time operating systems (RTOS) for better timing control.

8. System Stability Errors

System Crashes:

Conduct thorough testing under various conditions to identify and fix stability issues.

Implement robust error-handling and recovery mechanisms.

Overheating:

Use proper cooling systems, such as fans or heat sinks, to manage heat.

Monitor temperature and implement automatic shutdowns if overheating is detected.

Memory Overload:

Optimize code and data processing to reduce memory usage.

Use memory management techniques, such as garbage collection, to prevent overload.

CHAPTER FIVE

5.0 CONCLUSION

In conclusion, the development of the gesture-controlled robot demonstrates significant advancements in human-machine interaction, offering a more intuitive and natural way of controlling robotic systems. The project successfully integrates gesture recognition technology with robotic control, showcasing the potential for applications in various fields such as assistive technology, industrial automation, and entertainment.

The implementation of gesture control provides users with a hands-free, efficient means of communication with robots, reducing the need for traditional input devices. The system's accuracy and responsiveness highlight the effectiveness of the chosen sensors and algorithms, though further improvements can be made to enhance its robustness and adaptability in diverse environments.

Future work may focus on expanding the range of recognized gestures, improving system adaptability to different users, and integrating machine learning to enable continuous learning and adaptation. This project lays a solid foundation for further exploration and development in the realm of gesture-controlled robotics, paving the way for more sophisticated and user-friendly robotic systems.

5.1 RECOMMENDATION

To improve the accuracy and responsiveness of the robot, it is recommended to refine the gesture recognition algorithms. Incorporating advanced machine learning techniques could enable the system to adapt to different users and environments more effectively. Also, to improve the accuracy and responsiveness of the robot, it is recommended to refine the gesture recognition algorithms. Incorporating advanced machine learning techniques could enable the system to adapt to different users and environments more effectively.

5.2 REFERENCES

1. Arduino Official Website

Arduino. (n.d.). Arduino Uno. Retrieved January 20, 2025, from https://www.arduino.cc/en/Guide/ArduinoUno

2. L298 Motor Driver Information

Texas Instruments. (n.d.). L298 Dual Full-Bridge Driver. Retrieved January 20, 2025, from https://www.ti.com/product/L298

3. Gesture Recognition Overview

SparkFun Electronics. (n.d.). How to Use an Accelerometer to Control Your Robot. Retrieved January 20, 2025, from https://learn.sparkfun.com/tutorials/accelerometer-basics/all

4. Gesture-Controlled Robotics

Robotics Today. (n.d.). Gesture Control in Robotics. Retrieved January 20, 2025, from https://www.robotics.org/journals/gesture-control-in-robotics