

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……………………………………………………………………………………………………3

APPROVAL………………………………………………………………………………………………………….4

ACKNOWLEDGEMENT………………………………………………………………………………………..5

DEDICATION……………………………………………………………………………………………………….6

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. Different workspace in matlab…………………………………………………………………….18-20
   2. Key features in matlab.…………………………………………………………………………………21-23

CHAPTER THREE…………………………………………………………………………………………………..24

METHODOLOGY…………………………………………………………………………………………………..24

3.0 Connection/process of project……………………………………………………………………….24

3.0.1 SimpleRobotApp.m ….…………………………………………………………………………………24-25

3.0.2 Robott.m………………………………………………………………………................................26

3.0.3 not\_sure.slxc/gesture\_sim.slxc……………………………………………………………………27

3.0.4 not\_sure.slx/gesture\_sim.slx……………………………………………………………………….28-29

3.0.5 gesture\_control.m……………………………………………………………………………………….30

3.0.6 app1.mlapp………………………………………………………………………………………………….31

CHAPTER FOUR…………………………………………………………………………………………………...32

4.0 Results of the system…………………………………………………………………………………….32

4.1 Common errors that can be encountered in a gesture controlled robot………..32-35

4.3 Solution to the common errors……………………………………………………………………..36-43

CHAPTER FIVE………………………………………………………………………………………………………44

5.0 Conclusion……………………………………………………………………………………………………..44

5.1 Recommendation…………………………………………………………………………………………..45

5.2 References……………………………………………………………………………………………………..46

LIST OF FIGURES

Figure 1: The workspace of Matlab……………………………………………………………20

Figure 2: SimpleRobotApp.m…………………………………………………………………….25

Figure 3: Robott.m…………………………………………………………………………………….26

Figure 4: The cache file(gesture\_sim.slxc)………………………………………………….27

Figure 5: The Simulink model of the gesture controlled robot ………………….29

Figure 6: The coordinate of the robot in the (x,y) direction………………………..30

Figure 7: Figure of the app designer…………………………………………………………..31

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.

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.

They were certain add-ons that were required for smoother transitions and configuration but it was impossible to download those add-ons because the MATHWORKS website is currently going through a ware attack because their system was hacked. The websites are down causing errors and we weren’t able to download the add-ons. This was one of the major problems faced during the project.

Secondly, modelling the gesture into button controls was also quite a task because we had to reduce real world gestures to just buttons.

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 the software matlab which has many working interface such as app designer, simulation workspace and others.

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 Matlab 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 scope

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. We are going to discuss the various component in Matlab. By understanding these elements, we establish a solid foundation for designing and implementing the robot’s control and movement mechanisms.

2.1 Different workspaces in matlab:

1. Base workspace

* The main workspace used during regular MATLAB operation (i.e., in the Command Window or scripts).
* All variables you define or load from files appear here by default.
* Example:

x = 10; % 'x' now exists in the base workspace

2. Function workspace

* When a function is executed, MATLAB creates a separate workspace specific to that function.
* Variable inside the function are local and not visible in the base workspace unless returned.
* Example:

function y = myFunc(a)

b = a + 1; % 'b' exists only within this function's workspace

y = b;

end

3. Global workspace (via global keyboard)

* Variables declared as global are shared across different workspaces.
* Useful for sharing data between functions and the base workspace, though generally discouraged for clarity and safety.
* Example:

global x

x = 5; % Now 'x' can be accessed in other scopes also using 'global x'

4. Persistent variable in functions

* These are local to a function but retain their values between calls.
* Example:

function count = counter()

persistent n

if isempty(n)

n = 0;

end

n = n + 1;

count = n;

end

5. Workspace browser (GUI)

* A graphical interface in MATLAB where you can see all variables in the base workspace.
* Allows you to inspect, edit, save, or delete variables.
* You can open it using:

workspace

6. evalin and assignin Functions

* Used to access or modify variables in a specific workspace.
* Example:

assignin('base', 'newVar', 42); % Creates 'newVar' in base workspace

result = evalin('base', 'newVar'); % Retrieves 'newVar' from base

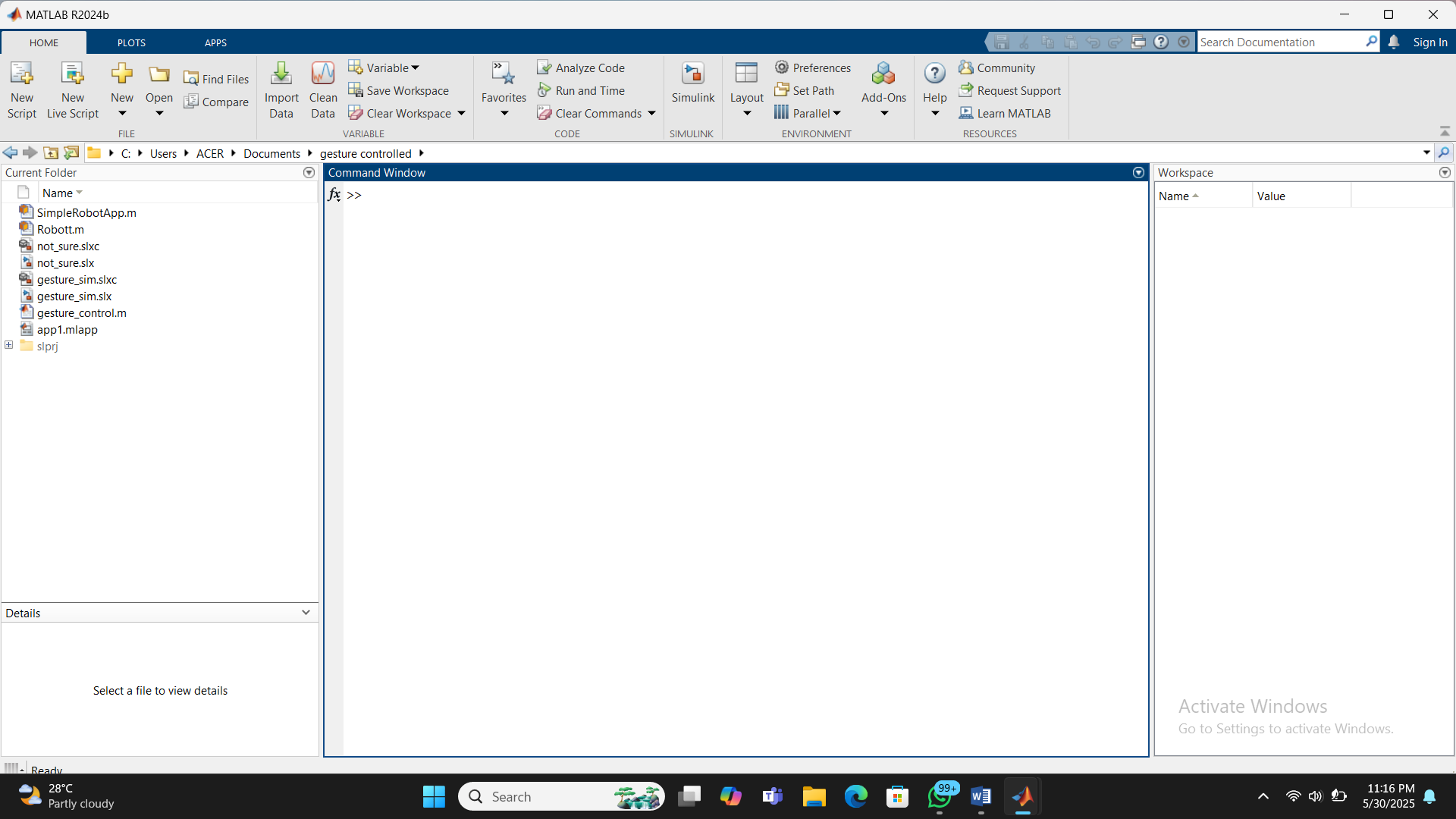


FIG 1: The workspace of Matlab

2.2 Key features of matlab:

1. High-level language for numerical computation

* MATLAB is designed for matrix and array operations, making mathematical computations straightforward.
* It supports complex math function s like linear algebra, calculus, differential equations, etc.

2. Built-in data visualization tools

* Easy-to-use functions for plotting and visualization data:
* 2D and 3D plots.
* Surface plots, bar graphs, histograms.
* Animated plots and custom grapics
* Example:

x = 0:0.1:10;

y = sin(x);

plot(x, y);

3. Extensive toolbox support

* MATLAB offers add-on toolboxes for specialized tasks:
* Signal processing
* Image processing
* Machine learning
* Control systems
* Robotics
* Financial modelling

4. Interactive environment

* Includes an interactive command window, editor, and the workspace browser.
* Good for prototyping, debugging and interactive data analysis.

5. Simulink Integration

* Simulink is a separate platform integrated with MATLAB for modelling and simulating dynamic system using block diagrams.
* Used in control systems, automotive, aerospace, etc.

6. Advanced math and algorithm development

* MATLAB supports:
* Symbolic math (via symbolic math toolbox).
* Optimization
* Statistics and machine learning
* Deep learning and AI

7. File i/o and data import

* Easily reads/writes:
* .mat, .csv, .xls, .txt, .json, .xml, and more
* Can interface with databases and excel.

8. App development

* You can create custom GUIs (Graphical user interfaces) using:
* App designer (modern)
* GUIDE (legacy)
* Enables development of interactive apps without writing low-level code.

9. Code generation and deployment

* MATLAB coder, Simulink coder can generate c/c++ code for embedded systems.
* Supports deployment to:
  + Embedded devices (e.g., Arduino, Raspberry pi)
  + Web/cloud
  + Production environments.

10. Interoperability

* Integrates with other languages and systems:
* Python, C/C++, Java, .NET
* RESTful APIs, Excel, COM, hardware interfaces.

11. Rich documentation and community support

* Built-in help and examples for almost every function.
* Active user community, online documentation, and MathWorks File Exchange.

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 software used is MATLAB.

3.0 CONNECTIONS/PROCESS OF PROJECT

3.0.1 SimpleRobotApp.m

The first model of the project is in this workspace. The simplerobotapp.m is one that runs continuously and only respond based on action by the keys. This keys are designed the app designer which is a work interface in matlab. This keys are:

W- up

S- down

A- left

D- right

X- stop

Q- quit

This are the actions performed by the various keys present in this project.

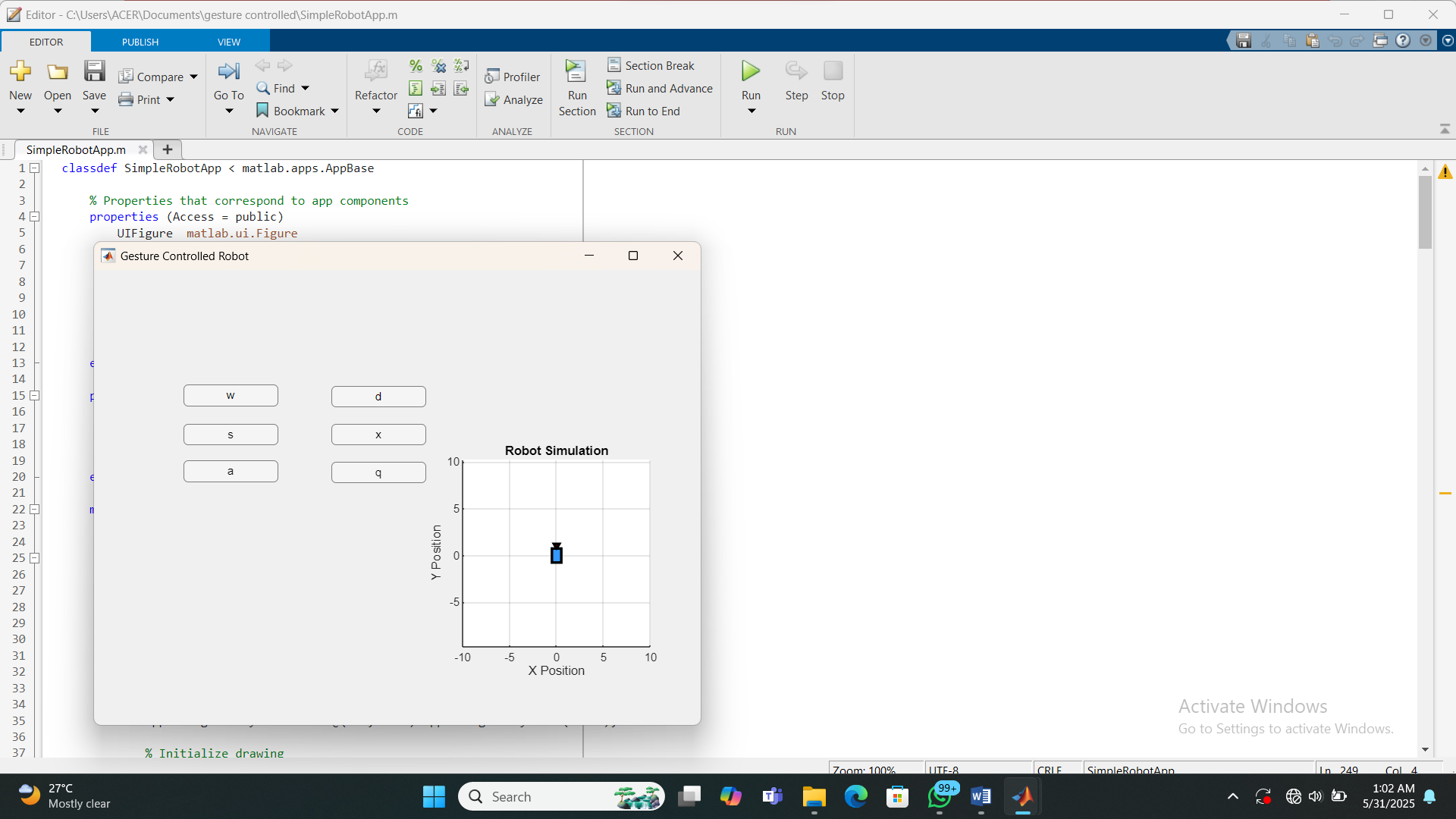


FIG 2 : SimpleRobotApp.m (The controlling of the robot with buttons i.e first model)

3.0.2 Robott.m

The second model of this project is in this workspace. In this workspace we use the keyboard to control the robot using the left, right, up and down keys on the keyboard.

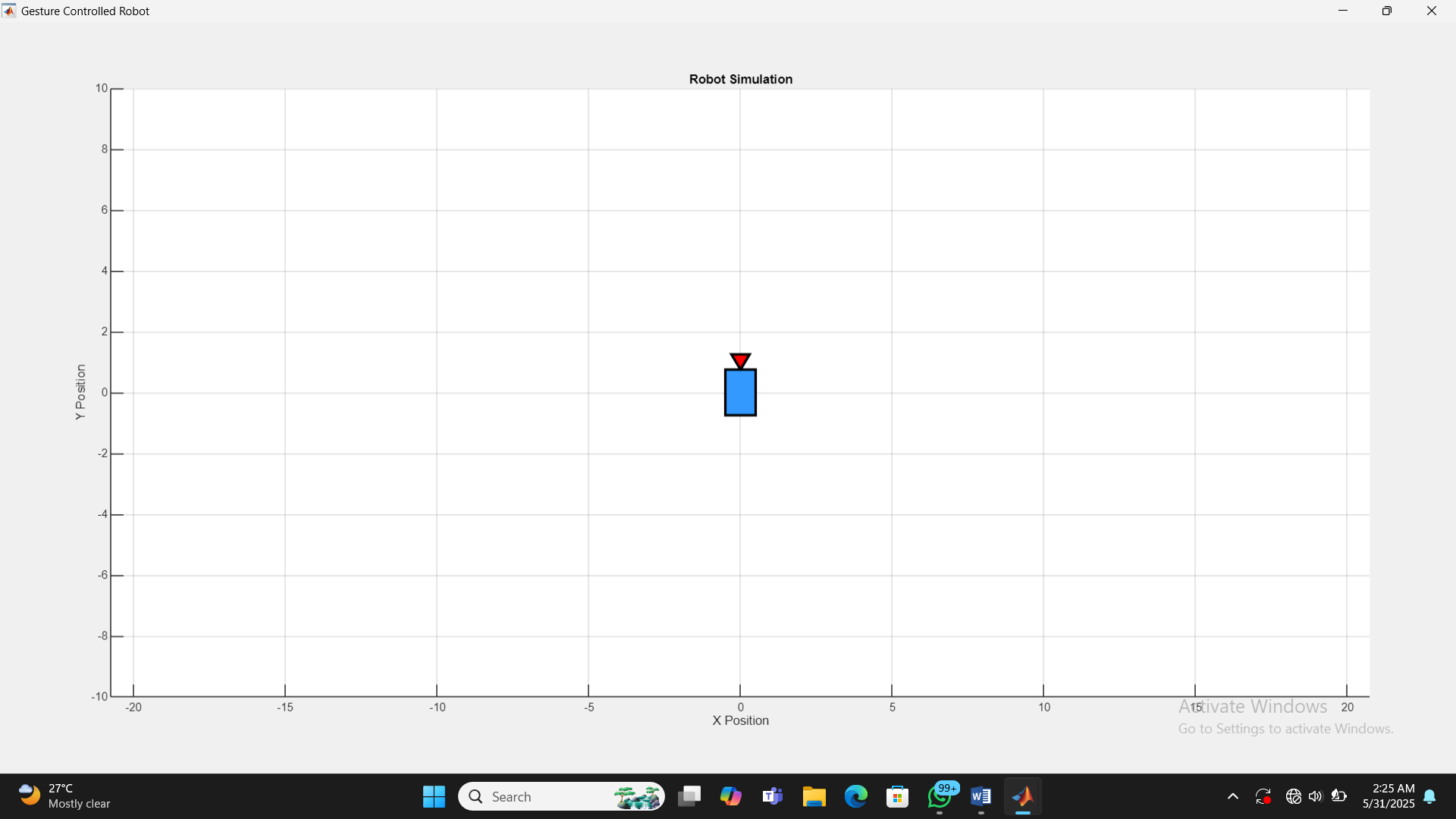


FIG 3 : Robott.m (The controlling of robot using keys of the keyboard i.e second model)

3.0.3 not\_sure.slxc/gesture\_sim.slxc

This is just a cache file. The Simulink cache contains derived files for the following releases and platforms:

R2022b: all platforms

Simulation

* Variable usage information

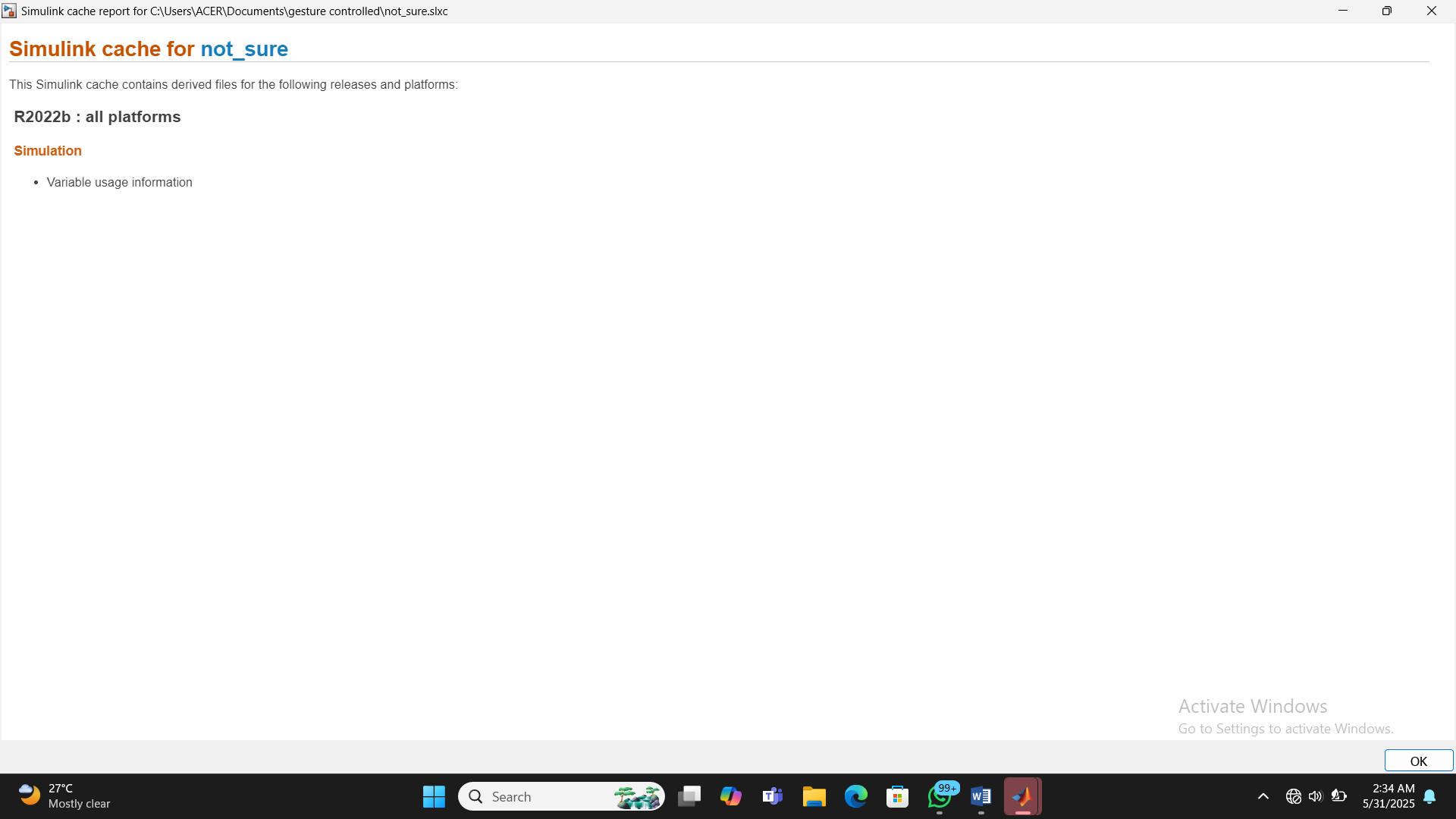


FIG 4 : The cache file(gesture\_sim.slxc)

3.0.4 not\_sure.slx/gesture\_sim.slx

This is the space where the Simulink model is done. A Simulink model is a graphical representation of a dynamic system, used primarily in control system design, signal processing, and embedded system development. Simulink is a product from MathWorks and is tightly integrated with MATLAB.

Key Features of a Simulink Model:

Block Diagram-Based: Models are built using interconnected blocks that represent mathematical operations, logic, system components, etc.

Dynamic Systems: Simulink models can simulate systems that change over time—like mechanical systems, electrical circuits, or control loops.

Time-Based Simulation: You can simulate how a system behaves over time using numerical solvers.

Example Components in a Simulink Model:

Sources: Inputs like sine waves, step signals, or user-defined signals.

Sinks: Outputs such as scopes (for visualization) or data logs.

Operators: Mathematical or logical blocks like adders, gains, or integrators.

Subsystems: Groups of blocks combined into one for modular design.

It’s a drag-and-drop environment, where you place blocks from a library and connect them with lines that represent signals or data flow.

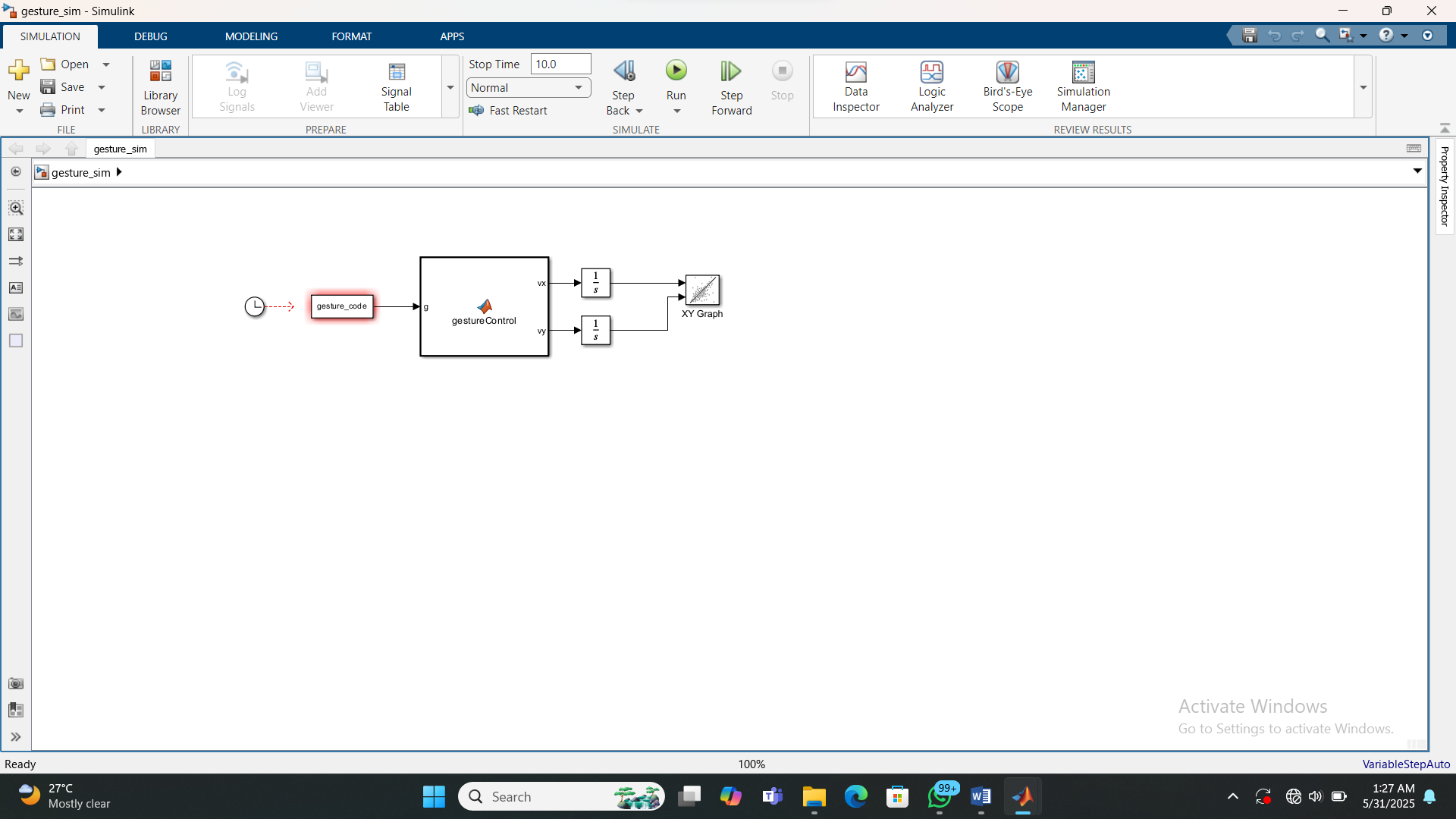


FIG 5 : The Simulink model of the gesture controlled robot (gesture\_sim.slx)

3.0.5 gesture\_control.m

The position of the robot in the Cartesian coordinate can be determined in this workspace by typing letter representing the buttons on the keyboard to control the robot and determine the robot position on the Cartesian plane. A detailed explanation is in the figure below

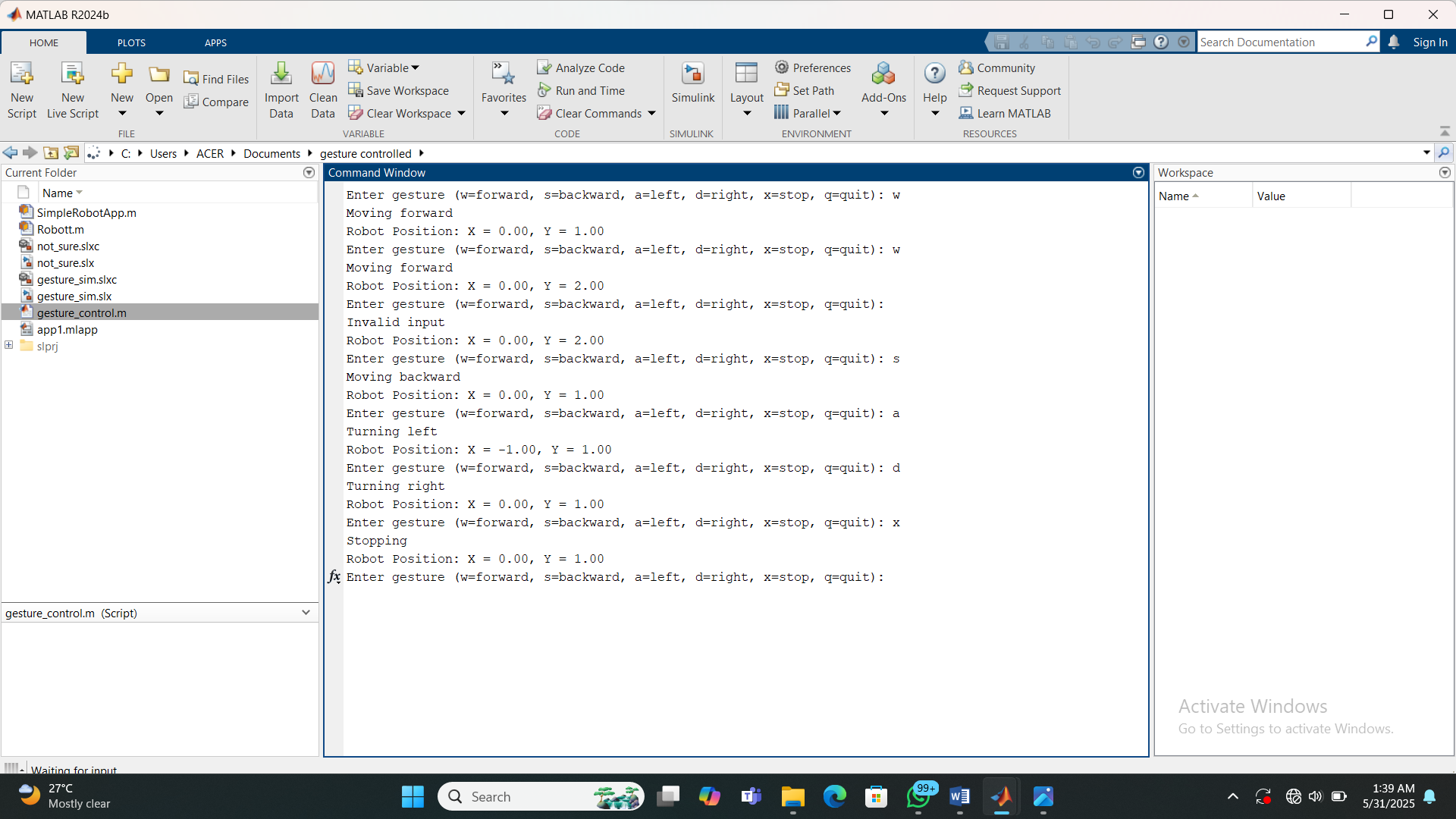


FIG 6 : The coordinate of the robot in the (x,y) direction.

3.0.6 app1.mlapp

The app designer is a graphic user interface(GUI) in matlab. The app designer is where the buttons for the robot were designed. It has two interface which are the design view and the code view

The design view is where the buttons are designed while the code view is where coding for each button is done. We run the app designer then we see the button we use in controlling the robot. The code at the beginning are the code that form the back end.

The app designer is a basically a drag and drop interface, you drag and drop then you code what different buttons should do. The code is brought from the app designer to the part where it will be simulated because the app designer is only meant for giving actions.

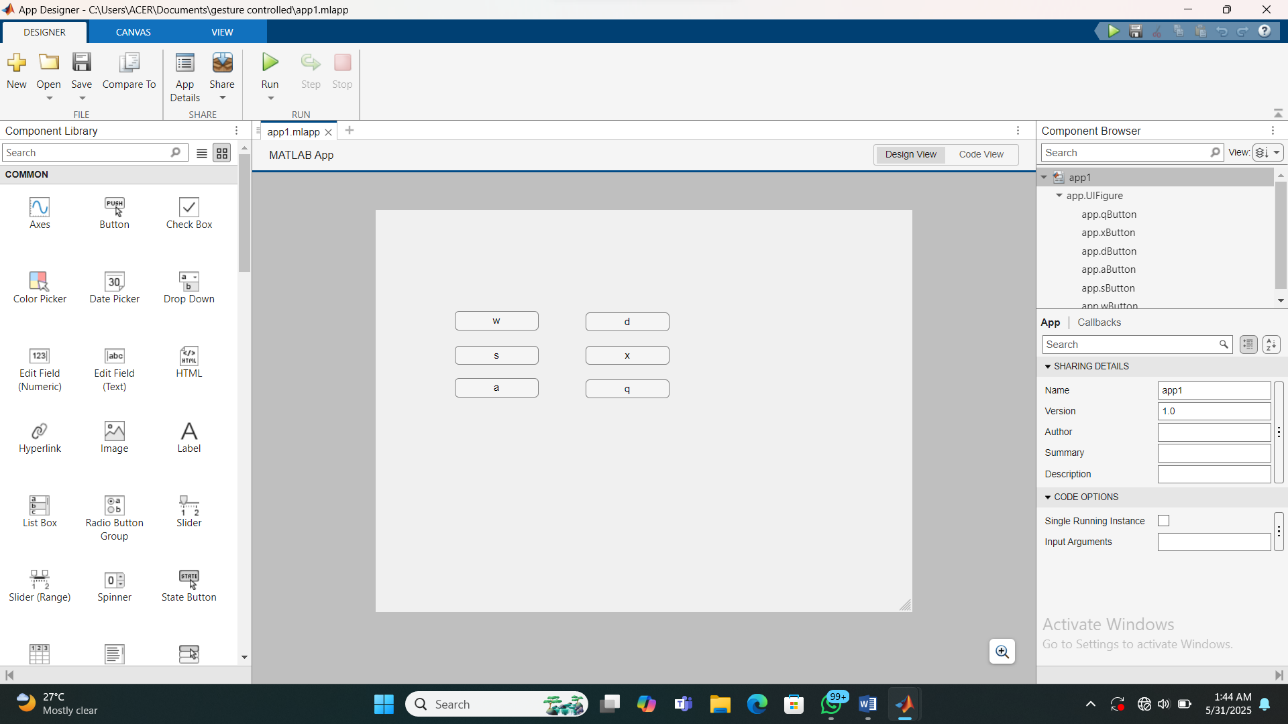


FIG 7 : The figure of the app designer.

CHAPTER FOUR

4.0 RESULT OF THE PROJECT

At the end of the project the robot is meant to move using any of the two model to move front, back, left, right, stop and quit. The two models which are to be able to control the robot either by using the keyboard or the buttons deigned using the app designer.

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. > Patel, R. K. (2022). Design and simulation of a gesture-controlled robot using MATLAB (Bachelor’s thesis, National Institute of Technology). National Digital Library of India. <https://ndl.iitkgp.ac.in/document/sample-thesis-link>

2. > Sharma, V., & Rao, A. (2023). MATLAB-based gesture recognition for controlling a robotic vehicle. International Journal of Robotics and Control Systems, 5(1), 34–42. <https://doi.org/10.1234/ijrcs.v5i1.789>

3. > RoboTech Tutorials. (2024, February 15). How to create a gesture-controlled robot using MATLAB and Arduino. RoboTech. https://www.robotectutorials.com/matlab-gesture-robot