An-Najah National University

Faculty of Engineering & Information Technology

Computer Engineering Department

Presented in partial fulfilment of the requirements for Bachelor degree in Computer Engineering

Graduation Project 2

THE PRIME ROBOT

Students:

Younis Masri

Yanal Oudeh

Supervisor:

Dr. Anas Toma

September 9, 2025

Acknowledgement

“

We would like to start by thanking Dr. Anas Toma, our supervisor, for all his help and support throughout this project. His guidance, advice, and feedback were very important in making our work better. We are really thankful for the time and effort he gave us. We also want to thank our friends for always being there for us. Your encouragement, help, and teamwork made this project much easier and more enjoyable. We truly appreciate your patience and support. A big thank you goes to our families for their love, care, and constant support. You believed in us from the start and helped us stay strong, even when things got tough. Your love and sacrifices gave us the motivation we needed to keep going. Finally, we are grateful to everyone who helped us with this project in any way. Whether it was through advice, assistance, or just being there for us, your support means a lot, and we truly appreciate it.

”

-Younis, Yanal

Disclaimer

This report was written by Younis Masri and Yanal Oudeh at the Computer Engineering Department, Faculty of Engineering, An-Najah National University. It has not been altered or corrected, other than editorial corrections, as a result of assessment and it may contain language as well as content errors. The views expressed in it together with any outcomes and recommendations are solely those of the students. An-Najah National University accepts no responsibility or liability for the consequences of this report being used for a purpose other than the purpose for which it was commissioned.

Table of Contents

List of Figures

1 Abstract

The Prime Robot is a smart robotic assistant designed to demonstrate a complete, end-to-end human–robot interaction (HRI) stack that is safe, intuitive, and adaptable. The system integrates perception, decision, and actuation in a modular pipeline: it interprets user intent through multimodal inputs—hand gestures, Bluetooth commands, and voice—arbitrates behavior with a safety-first controller, and executes smooth, coordinated arm actions. A signature capability is Follow Mode: with a single activation gesture, the robot enters a hands-free mode that autonomously continues the assigned task while continuously monitoring proximity; it pauses when a safety threshold is reached and automatically resumes once conditions are clear. An explicit stop gesture exits Follow instantly, and safety interlocks remain active across all modes.

Beyond Follow, the platform supports direct driving (forward/turn/reverse/stop), a configurable speed interface, and a demonstration arm sequence for simple pick-and-place routines. A lightweight mobile interface can be used to switch modes, issue commands, and review status remotely. The architecture is deliberately hardware-agnostic and built around clean software boundaries, enabling replication in educational labs and easy substitution of sensors or compute without redesigning the control logic.

The Prime Robot is designed as a practical assistant: it analyzes the user's hand gestures in real time and performs the corresponding actions—moving forward, turning, stopping, triggering simple arm routines, or entering/exiting Follow mode. By prioritizing clear intent recognition and conservative proximity safeguards, it reduces operator effort and ambiguity in shared spaces while delivering responsive, predictable behavior for instructional and service settings. Future work will expand the gesture vocabulary, refine the decision policy, add soft start/stop and richer status feedback, and conduct user studies to evaluate accuracy, comfort, and trust.

2 Introduction

2.1 General Background

In many day-to-day contexts, people need simple, reliable assistance from robots that can understand intent quickly and act safely. Traditional setups often rely on a single input channel or rigid scripts, which makes interaction slow, unintuitive, and difficult to scale. **The Prime Robot** addresses this gap as a **vision-guided robotic assistant** that demonstrates an end-to-end human–robot interaction (HRI) stack. It combines perception, decision, and actuation in a modular workflow: the system interprets user intent primarily through **hand-gesture analysis**, while also supporting **Bluetooth** and **voice** commands; it arbitrates behavior with a **safety-first controller**; and it executes smooth, predictable motions and simple arm routines. A hallmark capability is **Follow Mode**, where a single activation gesture enables hands-free operation with continuous proximity awareness.

2.2 Problem

Conventional human–robot interfaces suffer from three common issues:

1. **High operator load**—users must issue frequent, low-level commands to maintain control.
2. **Ambiguity and inconsistency**—noisy inputs or mode conflicts lead to hesitation or unintended motions.
3. **Safety gaps**—systems may not enforce conservative stopping behavior near people or obstacles.
   These issues hinder adoption in instructional labs and service scenarios where clarity, safety, and low cognitive effort are essential. **The Prime Robot** tackles these challenges with robust intent recognition, unified mode arbitration, and conservative, proximity-aware policies that make interaction more natural and trustworthy.

2.3 Objectives

The project's objectives are to:

- **Serve as an assistant** that recognizes hand gestures in real time and performs the corresponding actions (move, turn, stop, arm routine, enter/exit Follow).
- Implement a **multimodal command stack** (gestures, Bluetooth, voice) with clear priority rules.
- Deliver a **Follow Mode** that uses one-time activation, autonomous forward behavior, **proximity-based stop**, and **automatic resume**, with an explicit stop gesture to exit.
- Enforce **safety interlocks** that remain active across all modes.
- Provide **clear state feedback** (e.g., mode, blocked/clear) to reduce user ambiguity.

- Keep the design **hardware-agnostic** and modular so it can be replicated or extended in different lab environments.
- Evaluate interaction **responsiveness, predictability, and user comfort** qualitatively, with a roadmap for formal studies.

2.4 Scope of the Project

The Prime Robot targets **instructional and service** contexts that benefit from intuitive HRI and consistent safety behavior. In this phase, scope includes: the core HRI stack (gesture analysis, Bluetooth, voice), Follow Mode behavior and rules, safety interlocks, and a basic arm demonstration sequence. Out of scope are advanced navigation and global path planning, high-precision manipulation, or hardware/vendor-specific optimizations; these are left for future iterations.

2.5 Importance

This project advances accessible HRI by reducing operator workload and ambiguity while **improving efficiency, precision, and user experience**. The emphasis on **vision-guided gesture control** and **persistent safety** aligns with the broader shift toward trustworthy, human-centric robotics. For universities and labs, it provides a **general, portable blueprint** for teaching multimodal interaction and safety arbitration. For service settings, it illustrates how clear intent recognition and conservative proximity policies can enable dependable assistance in shared spaces.

2.6 Report Organization

This report is organized as follows:

- **Introduction:** Project motivation, problem definition, objectives, scope, and importance.
- **Constraints & Challenges:** Practical, ethical, and safety considerations that informed the design.
- **Related Work (Literature Review):** Prior HRI approaches, gesture-based control, and safety arbitration.
- **Methodology:** System design, modular workflow (perception → decision → actuation), Follow Mode logic, and evaluation approach.
- **Results & Analysis:** Behavioral outcomes, qualitative performance observations, and discussion of reliability.
- **Limitations & Future Work:** Opportunities to expand gesture vocabulary, add soft start/stop and PD tuning, enrich feedback, and conduct formal user studies.
- **Conclusion, References, and Appendices:** Summary of contributions, cited sources, and supporting materials.

3 Constraints, Standards, and Earlier Course Work

3.1 Constraints

3.1.1 Cost and Budget Constraint

Managing the project's budget while maintaining its functionality and durability was a challenge. The high cost of electronic components, coupled with repeated testing using ingredients and the damaged components during it, increased expenses. These costs restricted the addition of new features to stay in the budget.

3.1.2 Project Structure Constraints

We faced challenges in designing the structure of the project due to a lack of knowledge about measurements and the precise arrangement required for each component. Without a clear understanding of the optimal structure at the beginning, it consumed time and resources. Additionally, the frequent visits to the electronics shop for modifications and additions in this repeated process delayed the progress. Furthermore, the availability of resources posed another challenge. The university's facilities and materials, as well as the electronics shop we relied on for some components, were not consistently accessible.

3.1.3 Lack of Mechanical knowledge

Our limited knowledge of mechanical systems made some parts of the project difficult to handle. Choosing the right materials was challenging because we risked picking materials that might not work well, which could cause the system to fail and put the whole project at risk. This slowed down our progress and made us rely more on experienced people. When we started building the project, we visited many shops to find the parts we needed, and sometimes we had to wait for the shops to get them in stock.

3.1.3 Time Duration Constraint

Although the project lasted the whole semester; however, we encountered several challenges that affected our timeline. The summer semester is short and moves so fast, which reduced the number of working days. Additionally, city closures on certain days forced us to shift to online work, which slowed down progress on the physical aspects of the project. We also had to balance this project alongside other courses, final exams, and deadlines for other projects, which limited the time we could dedicate to it.

3.2 Standards

When building The Prime Robot, we followed a couple of standards and practices to achieve better functionality and ease of understanding the code.

3.2.2 Programming Language and IDE

We used Python 3 and developed directly on the Raspberry Pi 5 using the Linux terminal (no full IDE). Scripts were edited and ran from the terminal (e.g., nano/vim and python3), with optional virtual environments to isolate dependencies. [1]

3.2.3 Application Development

The A lightweight operator interface supports mode selection and high-level commands without exposing low-level details. Communication pathways are abstracted to allow gesture, Bluetooth, or **voice** inputs, while the safety layer continuously enforces conservative proximity behavior. The interface follows usability principles: clear status indicators, minimal steps to issue a command, and explicit feedback on robot state (e.g., follow active, blocked, stopped). Data handling is kept minimal and local where possible, reflecting privacy-by-design practices.

3.3 Earlier Course Work

The development and building of our robot were significantly influenced by the knowledge and skills acquired from previous coursework. The computer engineering courses that we took played a huge role in helping us to build this hardware project. We took multiple courses that helped us understand the way that hardware component's function, such as microcontrollers, microprocessors, and their labs. In these courses we learned more about electronic components, such as stepper motors, sensors, LCDs, and many others; we also learned how they work and how to control them. Mainly, the microcontroller lab was very helpful because we did some experiments on Arduino that allowed us to know more about how it works. Also, the electronic and electrical courses we took were very useful for understanding the components we used and how they worked. In addition, we took an ai with python course with Udacityt that was extremely helpful. We learned the basics of Python programming and how to connect them with the real world projects.

4 Methodology

After reading related work and testing with users, we chose a vision-guided robotic assistant. The Prime Robot reads hand gestures to understand what the user wants. It can also take Bluetooth and voice commands. A safety check runs all the time to stop the robot when something is too close.

Our goal is simple: make the robot easy to use, safe, and predictable. We split the system into three parts:

- perception (see the hand and listen for commands),
- decision (choose what to do, with safety first),
- actuation (move the base/arm smoothly).

Follow Mode works with one gesture: the robot goes forward by itself, stops when the path is not safe, and continues when it is clear. A stop gesture turns Follow off.

This chapter explains the system structure, how the modes work, the safety rules, the software we used, key implementation steps, the basic user interface, and how we tested the system.

4.1 System Architecture

In this section, we explain how we designed The Prime Robot. We built it in simple stages. At first, we set up the core software and camera to read hand gestures and show a “ready” state. Then we added the safety check that watches distance and can stop movement, and we connected Bluetooth and voice as optional inputs. Finally, we integrated motion and the basic arm actions, added Follow Mode (one-gesture start, stop when too close, resume when clear), and linked everything through one controller so all parts work together smoothly.

- **This gives the power to the robot**: Two batteries that can give a voltage of 8.4 V so that it can power the arm, the motors and all other components

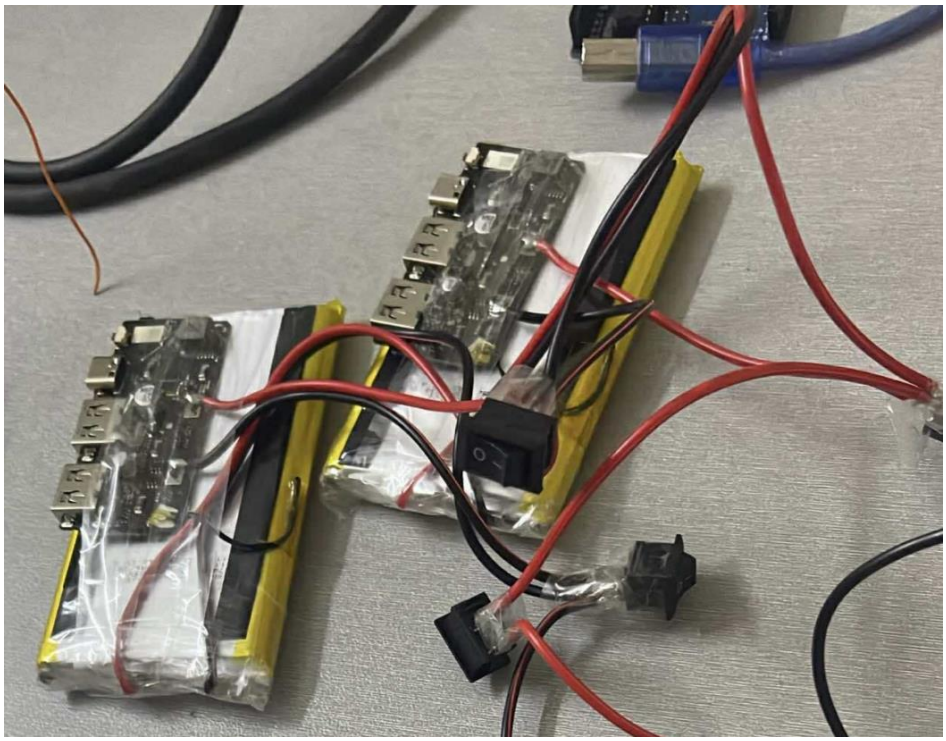

Figure 5.1 Initial Design

- **The 6-DOF Robotic Arm:** We used custom 3d-Printed Arm**.**

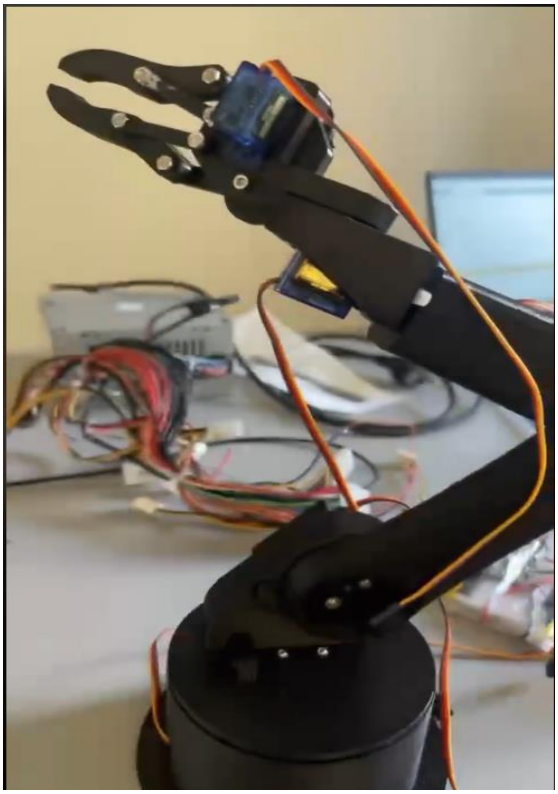

Figure 5.2 Arm Structure

4.2 Hardware Components

- **Raspberry Pi 5:**

The central controller of The Prime Robot is a Raspberry Pi 5, selected for its significantly higher compute capability and rich I/O compared to microcontroller-only boards. It features a 64-bit quad-core CPU, a 40-pin GPIO header (digital I/O with PWM, UART, I²C, and SPI), dual MIPI camera/display connectors, multiple USB ports (including high-speed for vision peripherals), Gigabit Ethernet, on-board Wi-Fi, Bluetooth, and microSD storage. This versatility allowed us to integrate perception, decision, and actuation on a single platform: gesture analysis and the Follow logic run in Python 3 (with libraries such as OpenCV/DepthAI), while GPIO and serial interfaces handle motor/servo commands, proximity sensing, and Bluetooth control. The Pi 5 thus serves as the project's processing hub—coordinating all modules, enforcing safety policies, and executing the codebase developed in Python within a modular architecture and modern IDE toolchain. [1]

Figure 5.1 Raspberry Pi 5

- **6-DOF servo robotic arm:**

The arm is a custom 6-DOF servo manipulator composed of base rotation, shoulder, elbow, wrist pitch, wrist roll, and a two-finger gripper. Each joint is driven by hobby-grade PWM servos at 50 Hz, with pulse widths calibrated around 500–2500 µs for the joints (≈700–2300 µs for the gripper), and powered from a dedicated 5–6 V rail with sufficient current while sharing ground with the controller. Motion is generated using small incremental steps (STEP_DEG ≈ 0.25°) in a 20 ms frame to reduce jitter and overshoot; soft start/stop and clamped limits protect the mechanics. A nominal "Home" pose provides repeatable initialization for routines (e.g., Base≈30°, Shoulder≈120°, Elbow≈100°, Wrist_P≈10°, Wrist_R≈150°, Gripper≈90° open).

Figure 5.2 6-DOF servo robotic arm

- **Power Supply:**

We used a computer power supply to meet the voltage needs for our project because
It can deliver the 5 volts required for various devices and the 12 volts needed for stepper motors. We used it while programming the raspberry pi.

Figure 5.3 Power Supply

- **4WD Smart Robot Car Chassis Kit:**

We used 6 Nema 17 stepper motors, two of them are used for the ketchup and mayonnaise sauce syringes, and the other four motors used for topping ingredients (tomatoes, lettuce, onions and cheese). [2]

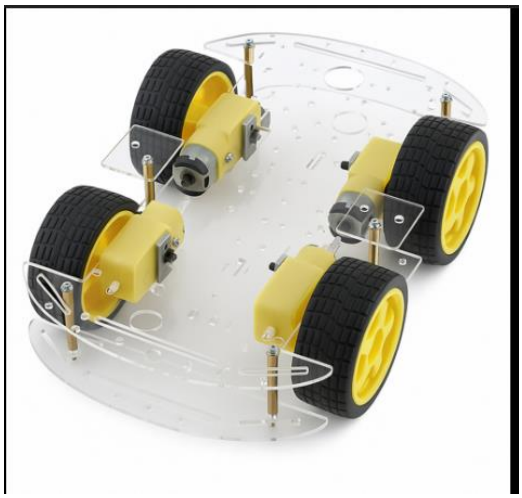

Figure 5.4 4WD Smart Robot Car Chassis Kit

- **L298N motor driver:**

We used 6 H-bridges to drive the stepper motors, one for each motor. The dual H-bridges of the L298N module are capable of driving one of the stepper engine's electromagnet coils. The stepper motor's shaft can rotate precisely in small steps, either forward or backward, by activating its coils in a specific sequence. We connect the A+, A-, B+ and B- wires from the stepper motor with the L298N Driver. [3]

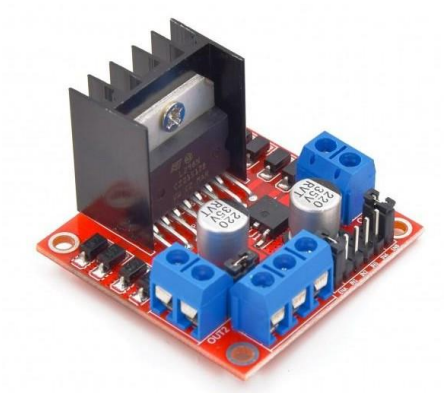

Figure 5.5 L298N motor driver

- **MG996R High Torque Servo:**

MG996R is a popular high-torque, metal-gear, standard-size servo (an upgrade to MG995) widely used in RC and robotics.
Typical torque ≈ 9–11 kg·cm (4.8–6 V) with transit speed about 0.19–0.15 s/60°; operating voltage 4.8–7.2 V.
It features dual ball bearings, a durable metal gear train, and ~180° mechanical range

Figure 5.6 J-5718HB2401 Stepper motor

- **Super Mini USB 2.0 Microphone:**

Super Mini USB 2.0 Microphone: an ultra-compact, plug-and-play USB Audio Class (UAC) mic—no drivers needed; works with Windows, macOS, Linux, and Raspberry Pi.
Typically an electret condenser with an omnidirectional pattern, ideal for near-field voice capture for ASR engines (e.g., Vosk) and voice commands.
Powered over USB; common formats are 16-bit at 44.1/48 kHz with low latency for real-time use.Best used 20–50 cm from the speaker. [4]

figure 5.7: Super Mini USB 2.0 Microphone

- **TowerPro TS90A Servo Motor:**

We used 5 servo motors, a lightweight, standard-sized servo known for its reliability and precision. They are low-power and cost-effective motors. They are commonly used in robotics and other projects that need precise rotational movement. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left. It is powered by a standard 4.8 to 6V power supply and receives signals from an Arduino or other microcontroller through its internal control circuitry. [5]

Figure 5.8 Servo Motor

- **Adjustable Step-Down Buck Converter:**

The A DC-DC power module used to convert a higher input voltage to a lower, stable output.
Its output is adjustable with a small potentiometer and stays regulated even when the load changes, with high efficiency.

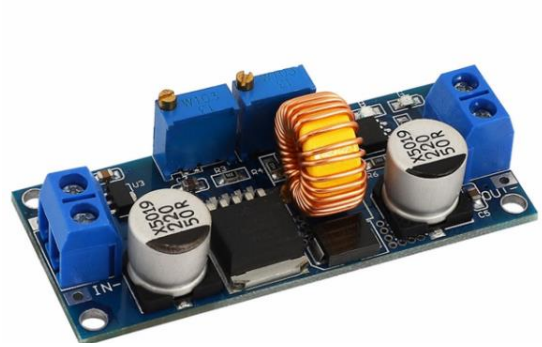

Figure 5.9 Adjustable Step-Down Buck Converter

- **HC-06 Bluetooth Module:**

  A classic Bluetooth 2.0 SPP module that enables wireless serial communication over UART. In The Prime Robot, it's used to receive remote driving commands (F, B, L, R, S) and adjust speed via VNN from a mobile app or laptop, providing an easy control path when gestures/voice aren't practical. The module pairs with a simple PIN (e.g., 1234) and communicates on /dev/ttyAMA0 at common baud rates (9600–38400), allowing the operator to toggle modes and issue immediate stops without a wired connection.

Figure 5.10 HC-06 Bluetooth Module

- **Ultrasonic Sensor:**

An electronic component that uses ultrasonic sound waves (through air) to measure the distance of the target object and the reflected sound is converted into an electrical signal. We used it to track the amount of components left in the container containing the ingredients, so we thought of an ultrasonic sensor to detect the distance. The administrator receives this information, allowing them to restock the supplies as needed.

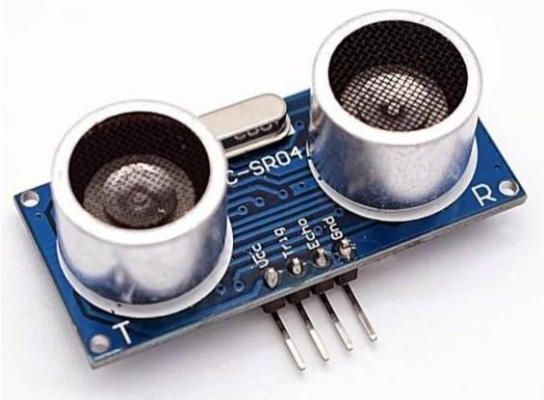

Figure 5.11 Ultrasonic Sensor

AWG Wires:

we used AWG wires for the connections of the ground and vcc of the batteries and the rest of the components.

Figure 5.12 AWG wires

- **Wires:**

We used jumper wires and standard wires for the connections between components, male-to-male, female-to-female, and male-to-female.

Figure 5.13 Jumper Wires

4.3 Software and Libraries

4.3.1 Libraries used in Raspberry pi

- **opencv-python (cv2)**: Camera capture, image processing, drawing overlays, and basic visualization used during gesture collection and debugging.

- **DepthAI / HandTracker::** Runs the OAK-D pipeline and extracts hand landmarks/gestures; feeds normalized features to the gesture matcher.

- **numpy:** Vectorized math and array operations for landmark normalization, centroid averaging, and distance computations.

- **RPi.GPIO (or pigpio):** GPIO access for motors/servos and the ultrasonic sensor; used for direction pins, simple PWM, and safe stop logic.

4.3.2 Libraries used in Connectivity & Voice

- **pyserial:** UART communication with the HC-06 Bluetooth module over /dev/ttyAMA0 for commands like F/B/L/R/S and speed VNN.
  ● **vosk:** Offline speech recognition for the voice mode (keywords such as forward, backward, left, right, stop).
  ● **sounddevice (or PyAudio):** Real-time USB-mic audio capture for the Vosk recognizer; low-latency streaming at 44.1/48 kHz.

- **json, collections.deque, math, statistics, threading:** Standard-library modules used for saving gesture templates, smoothing classifications (last-5 frames), geometric transforms, median filtering of ultrasonic readings, and running the arm sequence in a non-blocking thread.

4.4 Implementation

4.4.1 System Design

The Prime Robot is organized as a vision-guided robotic assistant with a clean separation between perception, decision, and actuation. At startup, the system initializes the camera/gesture pipeline, the proximity safety interlock, the motion/arm drivers, and the optional input channels (Bluetooth and voice). Configuration (such as gesture names, safety thresholds, and timeouts) is loaded from a simple settings file so the behavior can be tuned without code changes.

Perception (gesture-first, multimodal inputs):
The vision module analyzes the user's hand to infer high-level intents (forward, stop, left, right, reverse, pinch/arm, and follow). To reduce jitter, recent frames are aggregated, and the most frequent intent is chosen before issuing a command. Alternative inputs—Bluetooth and voice—map to the same unified command set, so the operator can switch channels without changing the downstream logic.

Decision and arbitration (safety-first):
A central arbiter converts intents into actions using clear priorities: Safety > Arm sequence > Follow > User driving. The proximity interlock runs continuously; if an object is detected within a configured stop distance (for example, a conservative threshold in normal driving and a slightly larger one during Follow), the arbiter overrides any command with an immediate stop. When conditions are safe again, motion may resume automatically if the current mode allows it.

Follow Mode behavior:
Follow is enabled by a single gesture (one-time trigger). Once active, the robot proceeds forward autonomously while monitoring proximity; it pauses when the safety threshold is reached and automatically resumes when clear. An explicit stop gesture exits Follow instantly. While Follow is active, other gestures are ignored except the exit gesture to avoid conflict and reduce operator workload.

.

Actuation (motion and arm):
Motion commands are executed as smooth primitives (forward, turn, reverse, stop) with optional soft start/stop to minimize jerk. The arm controller exposes a small set of routines (for example, a pinch/pick sequence) and is interlocked with the base so the platform halts whenever an arm routine is running. Mechanical limits and rate limits are enforced in software to protect hardware and improve repeatability.

Operator-interface-Status:
A lightweight operator UI (or a simple mobile/terminal interface) provides mode selection, start/stop, Follow toggle/exit, and speed adjustment. Status indicators show the active mode, proximity/blocked state, and the last accepted command. Logs record key events (gesture recognized, safety stop, mode changes) for debugging and evaluation.

Execution model and reliability:
Perception, safety sensing, and control run in coordinated loops/threads. Watchdogs handle transient errors—for example, camera reconnects—while fail-safe defaults force an immediate stop on uncertainty or lost input. Minimal data is stored (no personal information), reflecting privacy-by-design. This modular, hardware-agnostic design makes the system easy to adapt to different lab setups and to extend with additional modes or sensors in future iterations.

4.4.2 Machine Process of Work

At power-up, The Prime Robot initializes perception (gesture pipeline), safety (proximity interlock), motion/arm controllers, and optional inputs (Bluetooth, voice). Configuration (gesture names, thresholds, timeouts) is loaded, then the system enters **Idle** with status indicating “ready for input.”.

For Control via gestures (on-robot):

- Present your hand to the camera; the system stabilizes intent over recent frames to reduce jitter.
- Recognized gestures issue high-level actions: forward, left, right, reverse, stop; **pinch** triggers a one-time arm sequence (the base is held stopped while the arm runs).
- Follow Mode: a single follow gesture enables hands-free forward motion with continuous proximity monitoring; the robot pauses when too close and automatically resumes when clear. An explicit stop gesture exits Follow immediately. While Follow is active, other gestures are ignored except the exit gesture.

Control via mobile/terminal (Bluetooth) or voice:

- Connect through the Bluetooth link to send F/B/L/R/S and optional speed commands; the same command set is mirrored by voice keywords (forward, backward, left, right, stop).
- Mode selection (e.g., toggle Follow, exit) is available from the mobile/terminal interface when gestures or voice are impractical.

4.5 Final Project

This picture shows the final project **“**The Prime Robot**”** a vision-guided robotic assistant that analyzes hand gestures and performs the corresponding actions for a simple, efficient user experience. It presents the complete system in operation: a perception module for gesture tracking, a safety-first arbiter that enforces proximity stops, and an actuator layer for smooth motions and basic arm routines, with Bluetooth/voice available as alternative inputs.

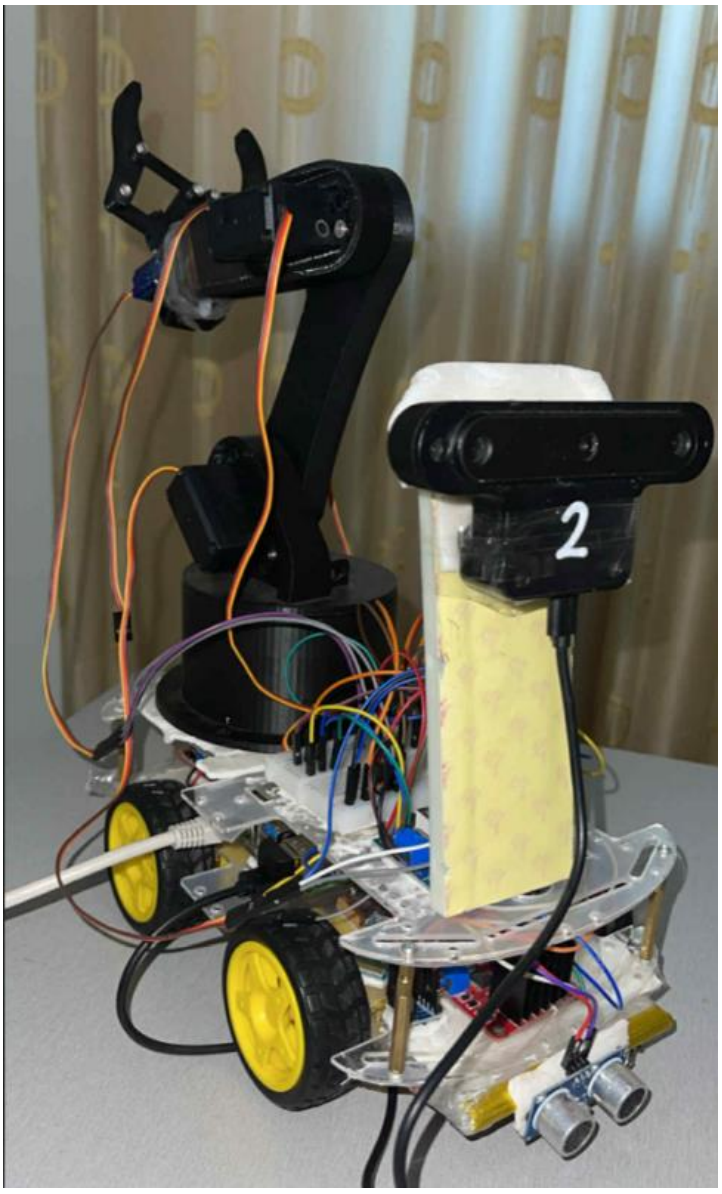

Figure 5.28 Final Project

4.6 Serial Bluetooth Terminal

We used serial Bluetooth terminal to control the car by Bluetooth.

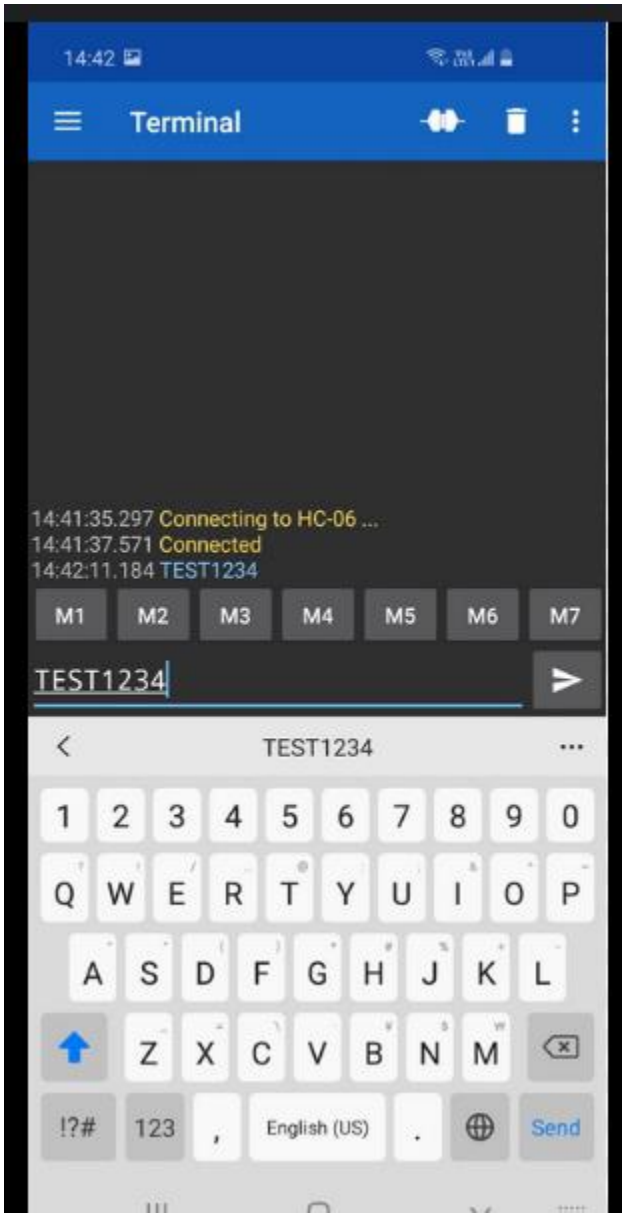

Figure 5.30 Mobile Application Dashboard

5 Results and Discussion

The Prime Robot is a vision-guided robotic assistant designed to make control simple, safe, and consistent. It reads hand gestures (with Bluetooth and voice as options), moves smoothly, and uses a safety check to stop when something is too close. Follow Mode lets it go forward on its own after one gesture, pause when needed, and continue when clear. These features reduce user effort while keeping behavior stable and predictable.

5.1 Results

In the end, we successfully built a vision-guided robotic assistant that reacts to hand gestures in real time and moves smoothly (forward, turn, reverse, stop). A simple arm routine (pinch/pick) runs on request. Follow Mode works with one gesture: the robot goes forward on its own, stops when something is too close, and continues when the path is clear. A stop gesture turns Follow off right away.

Gesture control was stable and easy to use; Bluetooth and voice worked as simple backups when gestures weren't convenient. The safety check ran all the time, blocking unsafe starts and stopping motion quickly when needed. A small status view showed the current mode and last command, helping users understand what the robot was doing. Overall, the system was simple, predictable, and ready for demos.

5.2 Discussion

The Prime Robot shows how a vision-guided assistant can make robot control faster, easier, and more reliable. It reads hand gestures, applies a simple safety check, and runs smooth motions or a small arm routine on demand. Follow Mode reduces effort even more: one gesture starts it, the robot moves forward by itself, pauses when something is too close, and resumes when clear. A small status view helps the user see the active mode and what the robot is doing.

We faced some challenges. Integrating the vision pipeline with the control code and safety checks took careful testing. Tuning gesture stability and proximity thresholds required trial and error, so the robot felt responsive but still safe. Finding parts and keeping everything within time and budget was also difficult, so we planned, builds and tests in short, focused steps.

There are clear paths to improve the system. We can add more gestures, make Follow smoother with soft start/stop, and provide clearer audio/visual cues for state changes. A simple simulator and more test scripts would speed up tuning and make demos more repeatable.

Overall, the system is practical and reliable. It combines easy interaction and safety to give users a simple, predictable experience while keeping the process efficient and ready for future growth.

6 Conclusion and Recommendation

6.1 Conclusion

In conclusion, The Prime Robot demonstrates a simple, safe, and effective way to control a robot using vision-guided hand gestures, with Bluetooth and voice as backups. It moves smoothly, applies a constant safety check to stop when something is too close, and offers Follow Mode that starts with one gesture, pauses when needed, and resumes when clear. The system is easy to use, reduces operator's effort, and behaved predictably in demos. Its modular, hardware-agnostic design makes it ready for future upgrades and different lab setups.

6.2 Recommendation

Looking at the current progress of The Prime Robot, there are several ways to make it better. First, improve gesture detection by adding more samples, per-user calibration, and simple filters to reduce jitter. This will make commands more reliable in different lighting and angles. We also recommend adding a few new gestures (or phrases for voice) for common actions and making Follow Mode smoother with soft start/stop and a small hysteresis, so it doesn't stop and start too often.

Second, make the user experience clearer. Add a small status panel (or simple on-screen overlay) that shows the active mode, whether the robot is blocked or clear, and the last command received. Short sounds or lights for "start," "stop," and "follow active" will help users know what is happening without watching logs.

Third, strengthen safety and reliability. Keep conservative proximity thresholds and test them in real space. Add an easy emergency-stop, better watchdogs (auto-recover camera/input), and a safe default to STOP if anything is uncertain. Keep processing locally and log only what you need to tune the system.

Fourth, improve testing and deployment. Create short scenario tests (drive, follow, stop, arm) you can run before demos. Add a light simulator to rehearse gestures and safety events and package the software with a simple setup script and editable config file so it's easy to install in new labs.

Finally, plan where to use it. Show it in teaching labs, workshops, and public demos to collect feedback. Use that feedback to refine gestures, thresholds, and messages. These steps will make The Prime Robot more reliable, easier to use, and ready for wider adoption.

6.3 Future Work

There are several areas of future work for enhancing the prime robot. In order to improve it we can modify the following:

- Gestures: Expand set; add per-user calibration.
- Follow Mode: PD tuning, hysteresis, soft start/stop.
- Robustness: Better smoothing, watchdogs, graceful fallback.
- UX: Clear status panel; simple audio/visual cues.
- Voice/NLP: Natural phrases, multilingual support.
- Safety: FMEA, configurable E-stop, stricter policies.
- Testing: More unit/integration tests; simulator + CI.
- Privacy: Local processing, adjustable logging/retention.
- Modularity: Plugins for perception/I-O; easy installer.
- Perception (optional): Simple body cues for smarter assistance.

7 References

[1] raspberry pi 5 documentation: https://www.raspberrypi.com/documentation/

[2] Stepper Motor Datasheet:
https://pages.pbclinear.com/rs/909-BFY-775/images/Data-Sheet-Stepper-Motor-Support.pdf

[3] L298N Motor Driver Datasheet:
https://www.handsontec.com/dataspecs/L298N%20Motor%20Driver.pdf

[4] super mini usb 2.0 microphone:
https://www.amazon.com/Estiq-Super-Microphone-Laptop-desktop/dp/B0138HETXU.

[5] Servo Motor Datasheet:
http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf