Robotics 2

ARI3215

Assignment Report

Amy Spiteri, Clyde Vella, Daniel Pace

January 24, 2026

Contents

1 Introduction

In this assignment, we were tasked with designing, implementing, and demonstrating an autonomous robot or simulation that can sense, reason, and act autonomously, demonstrating key aspects of embodied Artificial Intelligence (AI). A hardware implementation using the Elegoo Smart Robot Car V4 was chosen. Our robot is a gesture-controlled robot that responds to hand gestures in real-time, captured by a webcam. The system uses computer vision techniques to recognize specific gestures made by the user and translates them into movement commands for the robot. The robot also incorporates an ultrasonic sensor to detect nearby obstacles and prevent collisions and a gyroscope to maintain accuracy with turning.

All of the project code can be found on the following GitHub repository: https://github.com/spiteriamy/ari3215-robotics2. The final working python code can be found in the subdirectory `python/`. The final working Arduino code can be found in the subdirectory `arduino/robot/`.

2 System Design

3 Implementation

To implement the system described above, several components were integrated, including hardware setup, computer vision for gesture recognition, sensor fusion for autonomous behaviour, communication protocols, and control algorithms.

3.1 Hardware Integration & Setup

The hardware setup involved configuring the Elegoo Smart Robot Car V4 with an ultrasonic sensor for obstacle detection and a gyroscope for orientation tracking. The robot's motors were connected to the Arduino board, which served as the central controller. The ultrasonic sensor was wired to the appropriate pins on the Arduino to enable distance measurement, while the gyroscope was set up to provide real-time orientation data. Serial communication between the Arduino and a Raspberry Pi was established to facilitate command transmission based on gesture recognition.

Aside from the default components of the Elegoo Smart Robot Car V4, an additional ultrasonic sensor was added to extend obstacle detection to rear obstacles. A buzzer was also integrated to provide auditory feedback, and a LED strip was included for visual feedback based on the robot's actions.

On the side of the Raspberry Pi, a webcam was connected to capture real-time video feed for gesture recognition. The Raspberry Pi ran a Python script that processed the video input, recognized hand gestures, and sent corresponding movement commands to the Arduino via serial communication. The Raspberry Pi was configured to automatically run the gesture recognition script on startup, ensuring that the system was ready for operation as soon as it was powered on.

3.2 Computer Vision Pipeline

The computer vision pipeline was implemented using the MediaPipe library for hand detection and tracking. A webcam connected to the Raspberry Pi captured real-time video feed, which was processed to identify hand gestures. The MediaPipe Hands solution provided 21 3D landmarks for each detected hand, which were used to recognize specific gestures such as open hand, closed fist, and pointing. Custom algorithms were developed to interpret these landmarks and classify them into predefined gestures. The recognized gestures were then translated into movement commands for the robot.

An experimental alternate method of decoding hand gestures, where the angle of the pointing finger is read, and the robot is directed accordingly, was eventually abandoned due to instability and inaccuracy in gesture recognition.

3.3 Sensor Fusion & Autonomous Behaviour

The robot's autonomous behaviour was achieved by fusing data from the ultrasonic sensors, gyroscope, and gesture recognition system. The ultrasonic sensor monitored the distance to nearby obstacles only when necessary and not continuously, allowing the robot to make real-time decisions to avoid collisions. When an obstacle was detected within a predefined threshold, the robot would halt or change direction based on the last received gesture command. The gyroscope provided orientation data, which was used to enhance the robot's turning accuracy. When a turn command was issued via gesture recognition, the gyroscope data ensured that the robot executed the turn precisely, compensating for any drift or inaccuracies in movement.

3.4 Communication Protocol

The communication between the Raspberry Pi and the Arduino was established using serial communication over USB. The Raspberry Pi sent encoded movement commands to the Arduino based on the recognized gestures. A simple protocol was designed where each command corresponded to a specific action, such as moving forward, backward, turning left, or turning right, and by how much. The Arduino decoded these commands and executed the corresponding motor actions. Feedback from the ultrasonic sensor and gyroscope was also sent back to the Raspberry Pi when necessary to inform decision-making processes. The serial outputs from the Arduino were read and printed to the console on the Raspberry Pi for real-time monitoring and debugging purposes.

3.5 Control Algorithms

Control of the movements was handled through a custom intermediary library built specifically for the Elegoo Smart Robot Car V4 for our purposes. The main functions included in this library were to move forwards, backwards, turn left, turn right, and stop. Each function took parameters to specify speed and duration where applicable. The obstacle avoidance was handled inside the main control loop, where the ultrasonic sensor readings were checked using custom functions that limited the wait duration for a reading to avoid blocking the main loop. If an obstacle was detected within a certain distance, the robot would stop and wait for a new gesture command before proceeding. The gyroscope data was used to ensure accurate turning by adjusting the turn duration based on the current orientation of the robot.

3.6 Efficiency & Robustness

To ensure efficient hand detection and communication between the Raspberry Pi and Arduino, several optimizations were implemented. The gesture recognition algorithm is invoked at a controlled frame rate to balance responsiveness and computational load. The serial communication protocol was designed to minimize latency by using concise command encoding, and a command is only sent when a new gesture is detected, reducing unnecessary data transmission.

Error handling mechanisms were incorporated to manage potential issues such as incorrect gesture execution, communication failures, and sensor malfunctions. The system includes checks to verify the integrity of received commands and sensor data, displaying error messages on the console, and displaying error codes via the LED strip on the robot.

4 Testing & Evaluation

4.1 Limitations & Future Improvements

5 Ethical Considerations

The MediaPipe library was used for hand detection and tracking. Key ethical issues relevant to the use of MediaPipe and similar computer vision libraries include data privacy, surveillance, bias, and discrimination. Because the system relies on a live webcam feed, privacy and surveillance are concerns that should be considered. Individuals in the background may be captured unintentionally without their knowledge or

consent. To reduce privacy risks, users should be clearly informed when the camera is active (for example through an LED indicator). Additionally, informed consent should be obtained before recording or testing around other people. Privacy concerns can also be mitigated by avoiding the collection of unnecessary data, such as performing all video processing locally and avoiding storage of footage. If any footage must be stored, users should be informed about retention time, storage location, who has access, and the intended use of the data. Another important ethical concern is bias, fairness, and discrimination. Hand tracking accuracy can perform differently depending on a number of factors. This includes lighting conditions, skin tone, hand size, and anatomical differences. These performance differences may disadvantage users from underrepresented groups if tracking is less reliable for certain types of hands. Additionally, gesture-based interaction may also introduce accessibility issues. For example, users with limited hand mobility, missing fingers, or one-handed users may not be able to use the system effectively. Since the system controls a physical robot in the real world, physical safety is another ethical concern to consider. The use of robots in the real world can lead to robots colliding with objects, people, or pets. Misclassification of gestures can also lead to unintended movements. To mitigate these risks, safety mechanisms such as an emergency stop and safe speed limits should be included.

References