Senior Design: AR-based Palm-sized Robotic Assistant

Project Proposal

Team Members:

Ruixi Qin, Yuzhang Wang, Jiaqi Ding, Fengwei Yang

Course: ECE 445 Senior Design, Spring 2026

Professor: Prof. Liangjing Yang

March 24, 2026

Contents

1 Project Overview

1.1 Problem Statement

The current robot systems face core bottlenecks in achieving intuitive and widespread human-robot interaction, especially in scenarios requiring fine manipulation. Firstly, traditional remote control methods (such as programming or complex remote control) impose high cognitive load on users and are difficult to directly map human natural operation intentions, which is particularly prominent in AR/VR interactions that require real-time spatial perception. Moreover, 2D processing techniques lacking depth perception cannot meet the precise spatial requirements. Secondly, many immersive interaction solutions rely on expensive head-mounted VR devices, greatly limiting the technology's popularity in daily assistant and low-cost applications. Finally, lightweight and highly compatible control interfaces for palm-sized micro-robots (such as those based on the ESP32 platform) are still immature, hindering the effective application of these small but high-potential devices in scenarios like home assistants.

1.2 Solution Overview

This project proposes an AR-based Palm-size Robotic Assistant solution. By deeply integrating smartphone AR technology with the ESP32 microcontroller, it builds a low-cost and highly intuitive remote control platform. The core process is as follows: Firstly, the user's hand's three-dimensional posture and movement trajectory in the augmented reality (AR) environment are captured in real time through the smartphone's camera, and high-precision posture capture is achieved by using the built-in sensors of the phone (such as gyroscope and accelerometer). Secondly, these real-time collected posture data are wirelessly transmitted to the control module on the PC for intelligent conversion and calibration, mapping the user's natural gestures into kinematic instructions that the robot can execute. Finally, the control module sends precise joint angles or motion instructions to the ESP32 microcontroller via serial communication. As the core of the palm-sized robot, the ESP32 is responsible for driving the adapted micro-motors, ensuring that the robot's end effector can simulate the user's grasping and operation intentions in real time and accurately, thereby achieving a low-latency and high-fidelity virtual-real fusion interaction experience from user intention to physical execution, greatly reducing the threshold for remote fine operations.

1.3 High-Level Requirements

The system must be capable of using the camera and sensors of a smart phone to capture the user's hand or device posture in real time, and superimpose virtual information (such as robot models) onto the real environment, enabling users to express their operational intentions through natural gestures or pointing.

The interactive interface must be implemented based on smartphones to replace expensive VR headsets, ensuring that the technology is highly accessible and has a low entry barrier for a broader user base.

The entire control architecture must be seamlessly integrated with the Arduino ESP32 microcontroller to achieve low-latency wireless transmission of instructions and real-time motor control, meeting the rapid response requirements of the palm-sized robot.

The system needs to support basic operations for palm-sized robots, including but not limited to precise positioning, grasping and moving, and other delicate operations, to meet the task requirements in family assistant or specific scenarios.

2 System Architecture and Implementation

2.1 Block Diagram

The overall system architecture of *RobotPaPII* is illustrated in Figure 1. The diagram highlights the separation between the high-level software control and the low-level hardware execution, forming a cohesive closed-loop system.

2.2 Subsystem 1: Smartphone Control Layer

The smartphone acts as the high-level processing hub, responsible for environmental perception and user interaction:

- **Camera Input:** Captures real-time video frames at 30–60 FPS to provide the visual foundation for AR.
- **AR Engine (ARCore/ARKit):** Performs *Simultaneous Localization and Mapping* (SLAM) to track the robot's position relative to the environment.

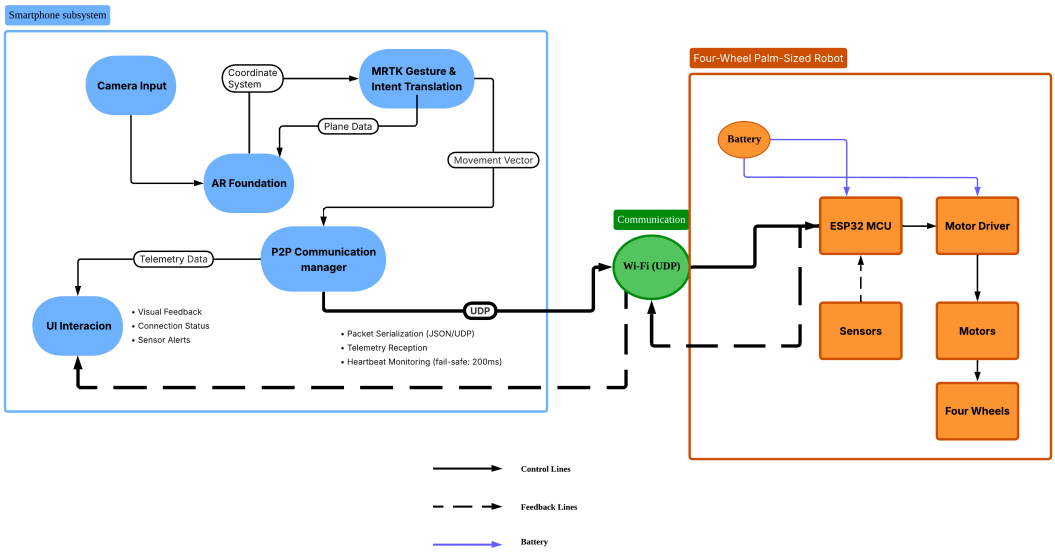

Figure 1: System Block Diagram of the AR-based Robotic Assistant.

- **UI Interaction Overlay:** A specialized interface where users can issue "Point-to-Move" commands by tapping on the AR-rendered ground plane.

2.3 Subsystem 2: Palm-sized Robot Hardware

The hardware subsystem executes physical movements and provides sensory feedback. It comprises three key components: the ESP32 microcontroller, motor drive system, and ultrasonic distance sensor.

2.3.1 ESP32 MCU

The ESP32-WROOM-32 serves as the robot's local control hub. It manages wireless communication, motion control, and real-time obstacle detection. The dual-core architecture allows concurrent execution of WiFi communication and motor control logic, ensuring deterministic response times.

Key Responsibilities:

- *Wireless Communication:* Maintains a UDP server, receiving coordinate packets from the smartphone. Packet parsing and validation occur within the communication loop.
- *Coordinate Translation:* Converts AR target coordinates into differential drive commands (left/right wheel velocities).

- *PWM Generation:* Generates PWM signals for each motor via the ESP32's LEDC peripheral.
- *Local Obstacle Avoidance:* Polls the ultrasonic sensor; if an obstacle is detected within the safety threshold, the ESP32 overrides user commands and executes an emergency stop.
- *Fail-Safe Monitoring:* Continuously monitors WiFi RSSI; if signal strength drops below -85 dBm, motor power is cut to prevent uncontrolled movement.

Technical Specifications:

Parameter	**Value**
Microcontroller	ESP32-WROOM-32
Operating Voltage	3.3 V (via on-board regulator)
Input Voltage (VIN)	5 V (from boost converter)
GPIO Used	2 for motors, 2 for ultrasonic
Communication Protocol	UDP over WiFi (2.4 GHz)
Control Loop Frequency	100 Hz

Requirements and Verification:

Requirements	Verification
R1: The ESP32 must process incoming UDP control packets and update motor PWM outputs within 50 ms of packet reception, measured from the start of UDP packet arrival to PWM signal change.	1A. Configure oscilloscope to trigger on ESP32 WiFi module's packet received interrupt pin. 1B. Send UDP packet from test smartphone while oscilloscope measures time to PWM output pin toggle. 1C. Repeat measurement 50 times; calculate average latency; requirement is met if average latency < 50 ms and maximum latency < 70 ms.
R2: Upon detecting WiFi RSSI below -85 dBm, the ESP32 must disable motor driver enable pins within 100 ms to prevent uncontrolled movement.	2A. Connect ESP32 motor enable pins to logic analyzer. 2B. Position smartphone 15 meters away with two walls between robot and phone to create signal attenuation. 2C. Monitor RSSI value via serial debug output while logging logic analyzer data; requirement is met if motor disable occurs within 100 ms of RSSI crossing -85 dBm threshold.
R3: The ESP32 must poll the ultrasonic sensor at minimum 20 Hz and override motor commands within 50 ms when obstacle distance $<$ 5 cm.	3A. Program ESP32 to toggle a debug GPIO pin at each sensor read completion. 3B. Use oscilloscope to measure period between debug pin toggles; verify frequency $\geq$ 20 Hz. 3C. Place obstacle 3 cm from sensor; use logic analyzer to measure time from Echo pin rising edge to motor enable pin state change; verify < 50 ms.

2.3.2 Motor Drive & N20 Motors

The robot employs a differential drive configuration using two N20 micro-gear motors. Each motor converts low-power control signals from the ESP32 into high-torque physical rotation for the dual-drive chassis.

Motor Selection: N20 micro-gear motors with integrated drivers are selected for their compact size, integrated driver, and simple control interface. Each motor accepts a single PWM signal that simultaneously controls speed and direction, simplifying wiring and reducing GPIO consumption.

Motor Specifications (N20 with 1:75 Gear Ratio):

Parameter	Value
Operating Voltage	3–6 V (5 V recommended)
No-Load Speed	290 rpm @ 6 V
Stall Torque	0.8 kg·cm
No-Load Current	50 mA
Stall Current	640 mA
Control Signal	PWM, 500–2500 µs pulse width
Interface	PH2.0-3P (GND / VCC / Signal)

Motor Control Logic:

- PWM pulse width of 1500 µs corresponds to motor stop.
- Pulse widths less than 1500 µs produce forward rotation (speed proportional to deviation).
- Pulse widths greater than 1500 µs produce reverse rotation.

This approach leverages the Arduino ESP32Servo library, which generates stable PWM signals with minimal CPU overhead. Differential steering is achieved by setting independent speeds for left and right motors:

$$\text{Left Speed} = \text{Base Speed} + \text{Turn Factor} \tag{1}$$

$$\text{Right Speed} = \text{Base Speed} - \text{Turn Factor} \tag{2}$$

PWM Signal Characteristics:

Parameter	Value
PWM Frequency	50 Hz (matching servo library standard)
Pulse Width Range	500–2500 µs
Resolution	16-bit (LEDC peripheral)
Idle State	1500 µs (motor stopped)

Requirements and Verification:

Requirements	Verification
R1: The motor drive system must achieve wheel speed linearly proportional to PWM pulse width deviation from 1500 µs, with speed measured at wheel circumference, maintaining ±10% linearity across 30% to 100% of maximum speed range.	1A. Mount robot on test stand with wheels elevated. 1B. Apply PWM pulse widths at 1600 µs, 1700 µs, 1800 µs, 1900 µs, and 2000 µs. 1C. Use non-contact tachometer to measure wheel RPM at each pulse width. 1D. Plot speed vs. pulse width deviation; calculate linear regression R^2 value; requirement is met if $R^2 \geq 0.95$.
R2: The differential drive system must achieve in-place rotation with turning radius $\leq$ 2 cm, measured at robot center, when left and right motors are commanded with equal magnitude and opposite direction.	2A. Place robot on flat surface marked with 1 cm grid paper. 2B. Command left motor forward at 50% speed and right motor reverse at 50% speed. 2C. Mark robot center position before and after 5 complete rotations; measure diameter of rotation circle; requirement is met if radius $\leq$ 2 cm.
R3: Motor response time from PWM signal change to measurable wheel movement must be $\leq$ 50 ms when transitioning from stop to 50% forward speed.	3A. Connect ESP32 PWM output pin to oscilloscope channel 1. 3B. Place reflective tape on wheel; position optical sensor aligned with tape; connect optical sensor output to oscilloscope channel 2. 3C. Command PWM change from 1500 µs to 1700 µs; measure time from PWM rising edge to first optical sensor pulse; requirement is met if time $\leq$ 50 ms. Repeat 10 times; report average.

2.3.3 Ultrasonic Distance Sensor

The ultrasonic distance sensor is responsible for spatial awareness by emitting acoustic pulses to detect physical obstructions. It enables obstacle avoidance, ensuring the safety of the robot without user intervention to prevent collisions.

Sensor Selection: The HC-SR04 ultrasonic sensor is selected for its low cost, ease of use, and sufficient accuracy for close-range obstacle detection. It operates by emitting a

40 kHz ultrasonic pulse and measuring the time-of-flight for the echo return.

Operating Principle:

1. ESP32 sends a 10 µs trigger pulse to the Trig pin.
2. Sensor emits 8 cycles of 40 kHz ultrasound.
3. Echo pin goes HIGH upon sound wave transmission and remains HIGH until the echo returns.
4. ESP32 measures the pulse width using pulseIn(), calculating distance as:

$$\text{Distance (cm)} = \frac{\text{Pulse Width (µs)} \times 0.034}{2} \tag{3}$$

where 0.034 cm/µs is the speed of sound in air at 20°C.

Technical Specifications:

Parameter	**Value**
Operating Voltage	5 V
Operating Current	15 mA
Detection Range	2–400 cm
Effective Range	5–100 cm (for reliable obstacle avoidance)
Accuracy	±0.5 cm (typical)
Measurement Frequency	30 Hz (achievable with optimized code)
Detection Angle	15° beam width
Interface	Trig (GPIO output), Echo (GPIO input)

Safety Logic Implementation:

Distance Range	**Action**
> 15 cm	Normal operation; execute user commands
5–15 cm	Reduce speed to 50%; prepare for stop
< 5 cm	Immediate stop; override user commands; turn away

Requirements and Verification:

Requirements	Verification
R1: The ultrasonic sensor subsystem must achieve measurement frequency of at least 20 Hz with distance measurement accuracy of ± 1 cm across 5 cm to 50 cm range when robot is stationary.	1A. Program ESP32 to toggle debug GPIO pin at start of each measurement cycle. 1B. Use oscilloscope to measure period between debug pin toggles; verify frequency $\geq$ 20 Hz. 1C. Place solid obstacle at measured distances of 5 cm, 10 cm, 20 cm, 30 cm, 40 cm, and 50 cm using ruler as reference. 1D. Record 50 distance readings at each reference distance; calculate mean and standard deviation; requirement is met if mean error $\leq$ 1 cm and standard deviation $\leq$ 0.5 cm at each test distance.
R2: The sensor must reliably detect obstacles within 5 cm to 100 cm range under normal indoor lighting conditions (200–500 lux) with detection rate $\geq$ 99% across 100 consecutive measurements.	2A. Set up robot on test bench with obstacle positioned at varying distances: 5 cm, 10 cm, 20 cm, 50 cm, 80 cm, 100 cm. 2B. Run 100 measurement cycles at each distance. 2C. Log distance readings via serial output; requirement is met if non-zero valid readings occur in $\geq$ 99 of 100 cycles at each test distance.
R3: The obstacle detection to motor stop latency must be $\leq$ 100 ms from obstacle entering 5 cm detection zone to motor disable, measured at robot maximum speed of 0.3 m/s.	3A. Program ESP32 to log timestamps via serial output at 1 ms resolution. 3B. Place robot on test track with motor speed set to maximum. 3C. Insert obstacle suddenly at 5 cm from sensor path using mechanical trigger. 3D. Record timestamp of obstacle insertion (from trigger switch) and timestamp of motor disable command; requirement is met if latency $\leq$ 100 ms. Repeat 10 times; report average and maximum latency.

Electrical Considerations:

Issue	Mitigation
Echo pin outputs 5 V	Use 1 kΩ voltage divider (1 kΩ series, 2 kΩ to ground) to step down to 3.3 V for ESP32
Multiple sensor interference	Space readings 30 ms apart; use one sensor per robot
Ambient noise	40 kHz frequency is above human hearing; minimal interference from environment

2.3.4 Chassis & Drive System

The robot adopts a four-wheel differential drive configuration to provide enhanced stability and maneuverability. The chassis is designed as a compact rectangular platform measuring approximately 10 cm × 12 cm, with four independently controlled N20 micro-gear motors driving each wheel. This four-wheel independent drive architecture offers superior traction and allows for precise control of the robot's movement.

Drive Configuration:

- Four-Wheel Independent Drive*:* Each of the four wheels is powered by its own N20 micro-gear motor, enabling independent speed control for each wheel. This configuration provides enhanced traction on various surfaces and allows for more precise turning maneuvers compared to traditional two-wheel drive systems.
- Differential Steering*:* The robot achieves steering by varying the rotational speeds of the left and right wheels. Forward movement is achieved by driving all wheels forward at equal speeds, while turning is accomplished by creating a speed differential between the left and right sides. The four-wheel configuration provides improved stability during high-speed turns and reduces the risk of tipping.
- Wheel Configuration*:* All four wheels are identical in size (diameter 32 mm) and feature rubber treads for improved grip on indoor surfaces such as carpet, hardwood, and tile. The wheelbase is 8 cm between front and rear axles, with a track width of 9 cm between left and right wheels.

Chassis Design:

The chassis is constructed from 3 mm thick acrylic material to provide structural rigidity while maintaining light weight. The total robot weight including battery, motors, and electronics is estimated to be 250–300 grams. Key chassis features include:

- Four motor mounting brackets with adjustable height to ensure all wheels maintain ground contact on uneven surfaces.
- Dedicated compartments for the ESP32 microcontroller, motor drivers, and battery management system.
- Mounting points for the ultrasonic sensor and optional micro-projector.
- Cable management channels to prevent wire entanglement with moving parts.

Technical Specifications:

Parameter	**Value**
Drive Configuration	Four-wheel independent drive
Chassis Dimensions	10 cm (L) × 12 cm (W) × 5 cm (H)
Wheel Diameter	32 mm
Wheelbase	8 cm
Track Width	9 cm
Chassis Material	3 mm acrylic / carbon fiber
Total Weight	250–300 g
Ground Clearance	10 mm

Advantages of Four-Wheel Drive:

Compared to a traditional two-wheel drive design, the four-wheel independent drive system offers several key advantages:

- Improved Traction: With all four wheels driven, the robot maintains better grip on slippery surfaces such as polished floors or low-pile carpet.
- Enhanced Stability: The four-point ground contact provides a more stable platform, reducing the likelihood of tipping during rapid acceleration or turning.
- Precise Maneuverability: Independent wheel speed control enables more precise turning and allows the robot to perform complex movements such as diagonal translation (crab walking) if programmed accordingly.
- Redundancy: In the event of a single motor failure, the remaining three motors can still provide basic mobility for recovery purposes.

2.4 ToleranceAnalysis

The most critical aspect of the *RobotPaPII* system is maintaining real-time spatial synchronization between the smartphone's AR environment and the physical robot's position. Because we rely on an edge-computing architecture with a P2P Wi-Fi UDP link, the system is highly susceptible to **network latency spikes**. If the control loop experiences severe delays, the robot may overshoot its target coordinate or fail to stop before colliding with an obstacle. Therefore, our tolerance analysis focuses on the **Total Stopping Distance** (d_{stop}) under worst-case network conditions.

2.4.1 Mathematical Model

The total stopping distance of the robot from the moment the user issues a "Stop" command (or the AR engine detects a virtual boundary) to the complete physical halt of the N20 motors is composed of three parts:

1. **Network Latency (t_{net}):** The time taken for the UDP packet to travel from the Unity app to the ESP32.
2. **Processing Latency (t_{proc}):** The time taken by the ESP32 to parse the JSON payload and drop the PWM duty cycle to 0.
3. **Mechanical Braking Time (t_{mech}):** The time required for the motors to dissipate their kinetic energy due to inertia.

Assuming the robot is traveling at its maximum design velocity v_{max}, the total stopping distance d_{stop} can be modeled as:

$$d_{stop} = v_{max} \cdot (t_{net} + t_{proc}) + \frac{1}{2} \cdot v_{max} \cdot t_{mech} \tag{4}$$

2.4.2 Worst-Case Scenario Calculation

Based on our hardware specifications, the palm-sized robot driven by N20 motors has a maximum velocity $v_{max} = 0.3$ m/s.

Under nominal operating conditions on a local 2.4 GHz Wi-Fi network, the parameters are:

- $t_{net\_nom} \approx 20$ ms $= 0.02$ s
- $t_{proc} \approx 5$ ms $= 0.005$ s
- $t_{mech} \approx 50$ ms $= 0.05$ s (due to the low inertia of the lightweight 200g chassis)

Nominal Stopping Distance:

$$d_{stop\_nom} = 0.3 \cdot (0.02 + 0.005) + 0.5 \cdot 0.3 \cdot 0.05 = 0.0075 + 0.0075 = 0.015 \text{ m } (1.5 \text{ cm}) \tag{5}$$

A stopping distance of 1.5 cm is well within our AR coordinate projection tolerance (± 2 cm).

However, in a **worst-case scenario** with heavy 2.4 GHz channel interference, UDP packets may experience significant jitter. Our system requirements state that the ESP32 will trigger a Fail-Safe halt if no packet is received for 200 ms. Thus, the absolute maximum theoretical delay before a forced shutdown is $t_{net\_worst} = 200$ ms $= 0.2$ s.

Worst-Case Stopping Distance:

$$d_{stop\_worst} = 0.3 \cdot (0.2 + 0.005) + 0.5 \cdot 0.3 \cdot 0.05 = 0.0615 + 0.0075 = 0.069 \text{ m } (6.9 \text{ cm}) \quad (6)$$

2.4.3 Conclusion and Safety Margin

The analysis demonstrates that even in the absolute worst-case network scenario (a 200 ms latency spike right before the Fail-Safe triggers), the robot will travel a maximum of **6.9 cm** before coming to a complete stop.

Since the ultrasonic sensors on the robot are configured to trigger a hardware-level override (independent of Wi-Fi) when an obstacle is detected within **10 cm**, the 6.9 cm worst-case stopping distance guarantees that the robot will not physically collide with objects. This proves that our control architecture and threshold values are well within safe engineering tolerances.

3 Ethical Issues & Safety Concerns

3.1 Ethical Issues

Our AR-based palm-sized robotic assistant combines a smartphone-controlled robot with a micro-projector. This integration raises several ethical considerations that we must address to ensure responsible design and deployment.

Data Privacy and Surveillance. The system relies on continuous camera input from the user's smartphone at 30 to 60 frames per second to perform SLAM-based localization and AR rendering. This constant visual data collection could inadvertently capture sensitive information about the user's private environment. As stated in the ACM Code of Ethics and Professional Conduct, computing professionals should respect the privacy of individuals and the confidentiality of information. To mitigate this risk, we will ensure that all visual processing happens locally on the smartphone. No raw camera frames will be transmitted to external servers. We will also implement a clear visual indicator that activates whenever the camera is actively capturing data, allowing users to remain aware

of when the system is operating.

Autonomy and Human Agency. The robot's point-to-move AR interface gives users direct control over the robot's movement. However, in shared spaces, the robot's operation might cause discomfort or anxiety to bystanders who are not interacting with the system. The IEEE Code of Ethics emphasizes the importance of avoiding harm to others. We will address this by implementing an audible status indicator to alert nearby people when the robot is active. Furthermore, the ultrasonic and Time-of-Flight sensors will serve not only for obstacle detection but also as a safety mechanism to prevent the robot from approaching individuals who do not wish to interact.

Accessibility and Usability. The primary interaction method relies on visual AR overlays displayed on a smartphone screen. This could exclude individuals with visual impairments or those who are not familiar with smartphone-based AR interfaces. Drawing from the ACM principle that computing professionals should ensure that the public good is the central concern, we will explore alternative control methods such as voice commands or physical buttons on the robot itself.

3.2 Safety Concerns

The combination of a mobile robotic chassis with a micro-projector introduces a range of physical safety concerns that we must consider carefully during design and testing.

Electrical Safety and Battery Management. The palm-sized robot is powered by a lithium-ion battery to allow untethered operation. Improper charging, over-discharge, or physical damage to the battery could lead to overheating, fire, or explosion. We will incorporate a dedicated battery management system with over-current, over-voltage, and short-circuit protection. All battery connections will be securely insulated, and the battery compartment will be designed to prevent puncture or crushing during normal operation or minor impacts.

Moving Parts and Pinch Points. The robot's dual N20 motors drive a two-wheeled chassis with exposed wheels and axles. Small fingers or loose clothing could become entangled during operation, especially around children or pets. We will enclose the robot in a smooth, seamless housing that eliminates pinch points. The housing will feature rounded edges and no accessible moving components on the exterior.

Projector Light Hazard. The onboard micro-projector provides spatial feedback but emits concentrated light that could cause eye strain or discomfort if viewed directly at close range. The robot's small size increases the chance that a user might inadvertently

look straight into the projector lens. We will use a downward-angled projection mount and limit the projector's brightness to the minimum level needed for clear visibility. We will also consider adding a diffuser or protective lens cover to reduce the intensity of direct light emission.

Laser Safety from the Time-of-Flight Sensor. The robot uses a Time-of-Flight sensor for obstacle detection, which emits infrared laser light. Off-the-shelf modules are typically certified as Class 1 laser products, meaning they are safe for normal use. We will mount and shield the sensor following the manufacturer's safety guidelines to prevent accidental direct eye exposure during normal operation.

Wireless Communication Interference. The system relies on Wi-Fi and Bluetooth UDP communication between the smartphone and the ESP32. In environments with dense wireless traffic, packet loss or latency spikes could cause the robot to receive delayed or corrupted movement commands. We will implement a watchdog timer on the ESP32 that automatically stops all motor movement if no valid command is received within a defined timeout period such as 200 milliseconds.

Mechanical Strength and Collision Risk. The robot weighs roughly 200 to 300 grams and is capable of moderate speeds. A collision could cause minor injury or damage fragile objects. We will tune the motor response to prioritize safety over speed, limiting the maximum velocity to a safe threshold. The chassis will be made from impact-resistant materials with rounded corners to minimize potential harm.

Overheating and Thermal Management. The ESP32 microcontroller, motor drivers, and micro-projector all generate heat during operation. Without adequate thermal dissipation, internal temperatures could shorten the useful life of the component or create a risk of burn. We will perform thermal simulations and incorporate passive cooling features such as ventilation slots and heat sinks to ensure external surface temperatures remain within safe limits.

Structural Integrity. The compact design of the robot requires the motors, circuit board, battery, and projector to fit tightly within a small volume. Vibration and repeated impacts could break down internal components over time. We will use secure mounting methods such as threaded inserts and strain relief for cables. We will also perform drop tests and vibration tests to validate long-term structural integrity.