ECE445

SENIOR DESIGN LABORATORY

DESIGN DOCUMENT

A World-Model based Infant Interaction Robot

Team #44

Ruijin Xu
Sylvia Chung Yan Shan
Artha Amanda Sidauruk
Zishuo Feng

Supervisor: Gaoang Wang

April 08, 2025

Table of Contents

1. Introduction

1.1 Problem Statement

In close human-robot interaction scenarios, especially involving infants or small children, safety is critical. Traditional robots rely on **reactive behavior**, meaning they only respond after contact or near-contact occurs. This can lead to unsafe interactions, delayed responses, and poor user experience.

Additionally, many low-cost robots lack:

- predictive awareness of approaching objects
- smooth interaction behavior
- integration of perception and motion

This project aims to develop a **low-cost mobile robot** that can:

- interact safely with nearby humans
- predict short-term contact risk
- proactively adjust its motion to avoid unsafe interactions

1.2 Solution Overview & Visual Aid

The proposed system consists of:

- **Vision system (external camera + laptop)**
  detects human/hand movement and predicts risk
- **Mobile robot (ESP32-based)**
  executes movement and interaction behaviors
- **Communication system**
  sends commands from vision → robot

System Flow (conceptual)

```
Camera → Vision Model → Risk Prediction → Command → ESP32 → Motors/LED/Buzzer
```

The robot operates in three modes:

- **Engage mode** (safe interaction)
- **Caution mode** (approach detected)
- **Escape mode** (high risk → move away)

1.3 High-Level Requirements

(1) Safety & Avoidance

- Achieve quantifiable improvement over a reactive baseline in at least one metric (e.g., lower contact rate, higher avoidance success rate, faster response, or greater maintained separation distance)
- Maintain minimum safe distance (>20–30 cm)

(2) Prediction Performance

- Predict contact risk within 0.5–1.0 seconds
- ≥70% correct prediction of approaching motion
- Model must be lightweight and capable of real-time inference on embedded platforms

(3) System Responsiveness

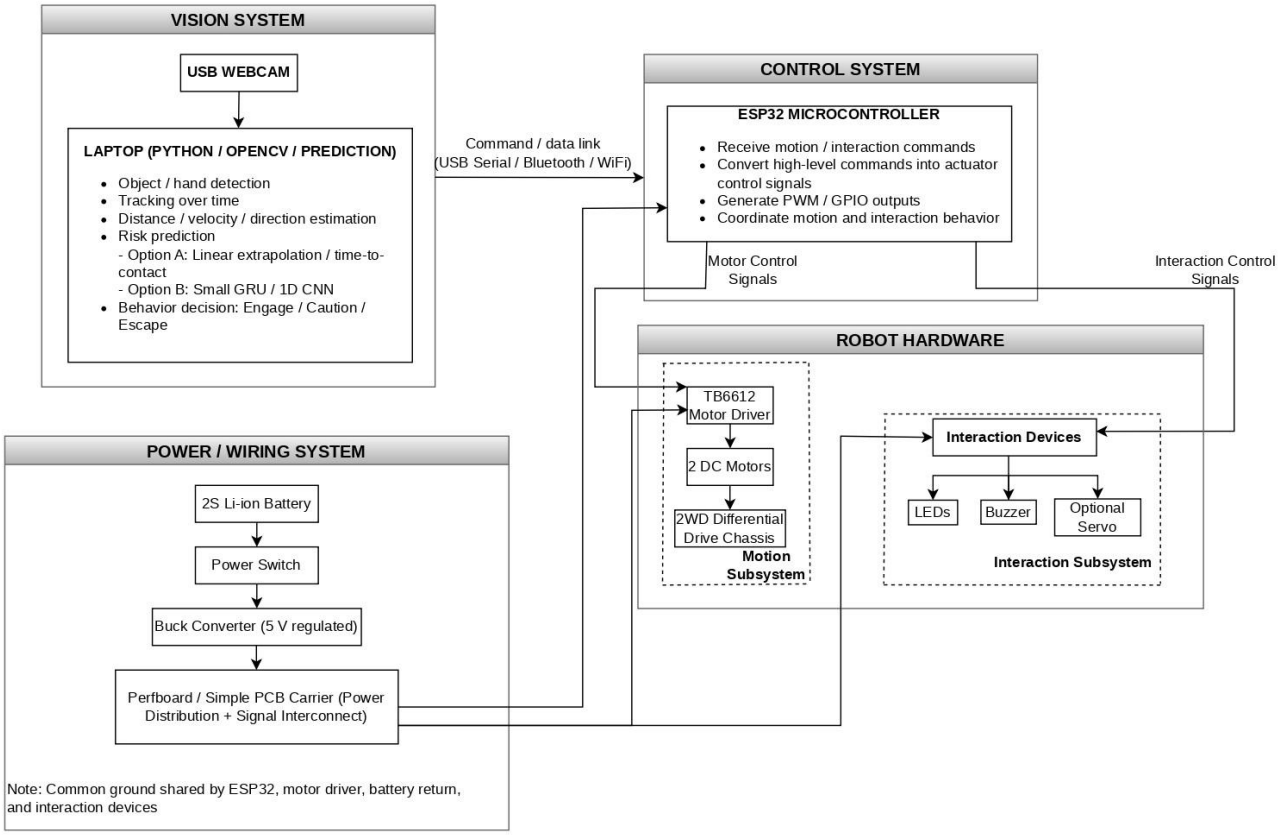
- End-to-end latency ≤200 ms
- Control update rate ≥10 Hz
- Robot must execute proactive avoidance maneuvers upon prediction trigger (not purely reactive braking/retreat)

(4) Interaction Behavior

- Robot provides visible feedback:
  - LED signals
  - Buzzer with different frequencies for warning vs. avoidance
  - simple motion gestures (e.g., slight retreat or turn to signal avoidance behavior)

2. Design

2.1 Block Diagram



The block diagram above shows the overall architecture of the infant interaction robot with predictive avoidance. The system is divided into four main parts: the vision system, control system, robot hardware, and power/wiring system. The USB webcam sends visual data to a laptop for detection, tracking, and short-term risk prediction, and the resulting commands are transmitted to the ESP32 through USB serial, Bluetooth, or WiFi. The ESP32 then generates control signals for the motion subsystem, consisting of the TB6612 motor driver, two DC motors, and a 2WD differential-drive chassis, as well as for the interaction subsystem, which includes the LEDs, buzzer, and optional servo. The power and wiring system supplies regulated power to the onboard electronics and provides the necessary interconnections for coordinated system operation. This integrated design allows the robot to detect approaching objects, assess potential contact risk, and respond through

motion and interaction behaviors, which directly supports the project's goal of achieving safe and proactive interaction between the robot and nearby users.

2.2 Vision System Design

2.2.1 Purpose

The vision system serves as the robot's high-level perception and behavior engine. Its role is to observe the environment through the front-facing camera, detect and follow a person, estimate how that person is moving relative to the robot, and determine the appropriate behavioral response. Rather than directly controlling low-level actuators, it produces high-level decisions such as whether the robot should engage, remain cautious, or retreat.

2.2.2 Hardware

The vision subsystem requires a USB webcam, a laptop running the perception software, and a stable mounting frame that places the camera at the front of the robot with a clear forward-facing view. A camera capable of 720p or 1080p resolution at around 30 fps is sufficient for the initial prototype. A wide-angle field of view in the range of about 70 to 100 degrees is preferred, so the robot can detect users over a wider frontal area, and the camera should perform reasonably well under variable indoor lighting.

2.2.3 Software Stack

The software stack is centered on Python 3.x with OpenCV and NumPy forming the base for image capture, processing, and geometric calculations. Depending on the final detection approach, MediaPipe may be added for pose or hand detection, while YOLO or MobileNet SSD can be used for person detection. PySerial is recommended for communication with the ESP32 over USB serial, and PyTorch or ONNX Runtime may be included if a neural network model is deployed locally.

Human and Object Detection

The primary goal of the detection module is to identify a person in the robot's field of view, with optional hand detection if gesture-based interaction is needed later. A practical detection hierarchy is to first detect the person, then refine localization using face or upper-body cues, and finally detect hands only when required for play or interaction gestures. Each

detection should produce at minimum a bounding box (`x, y, w, h`), a confidence score, and, if tracking is enabled, a target ID.

Tracking

After a person has been detected, the system should maintain a stable track of that same target across consecutive frames. This prevents erratic switching and allows motion trends to be estimated over time. The tracking layer can be implemented using centroid tracking, SORT or DeepSORT, OpenCV's CSRT tracker, or Kalman-filter-based smoothing depending on the desired balance between simplicity and robustness. The output of tracking should include a stable target center, an estimate of the target's image-plane velocity, and a persistent identity for the selected target.

Distance Estimation

Distance estimation is needed to decide when the robot should approach, hold position, caution, or escape, and also to adjust music or interaction intensity. For the first prototype, the simplest method is to use the apparent height of the person's bounding box as an inverse proxy for distance, where a larger box indicates the person is closer and a smaller box indicates they are farther away. This approach is easy to implement and requires no additional hardware, although it is sensitive to posture and camera angle. A more refined alternative is a calibrated monocular estimate using known camera parameters and approximate body dimensions, while stereo or depth sensing can remain a future upgrade path. For initial deployment, bounding-box height with a calibration lookup table is recommended. The interaction space can be divided into practical distance zones such as far for distances above 2.0 m, medium for 1.0 to 2.0 m, near for 0.5 to 1.0 m, and too close for less than 0.5 m.

Direction and Approach Estimation

In addition to estimating distance, the vision system should determine whether the user is approaching, retreating, remaining relatively static, or moving laterally across the scene. This can be inferred from the time history of bounding-box size, optical flow, and frame-to-frame displacement of the tracked target. A simple rule-based implementation is sufficient for the first version: if the bounding box grows consistently over time, the user is approaching; if it shrinks, the user is moving away; and if the target center shifts significantly left or right, the user is crossing laterally relative to the robot.

Risk Prediction

The risk prediction layer decides whether the interaction should remain playful or transition into caution or escape behavior. Risk should increase when the person approaches too quickly, comes excessively close, moves suddenly, thrusts a hand rapidly toward the camera, or blocks the robot's ability to retreat safely. A three-level risk model is appropriate for this application. Low risk corresponds to normal engagement, medium risk triggers caution behavior, and high risk causes the robot to enter escape mode.

2.2.4 Vision State Machine

The vision system should operate through a behavior-oriented state machine so that perception results translate into clear robot responses. In the **Idle Search** state, no valid target is present, and the robot remains in a passive ambient mode, possibly with soft music or a low-intensity light pulse. In **Target Acquired**, a person has been detected and the robot begins orienting toward them. In **Engage**, the target is within a comfortable interaction range, and the robot is allowed to move gently while playing interactive music. In **Caution**, the target is too close or approaching too quickly, so the robot slows, alters its sound and lighting behavior, and prepares to retreat if needed. In **Escape**, the risk signal is strong enough that the robot backs away and plays a clear alert pattern. In **Lost Target**, the tracked person temporarily disappears, so the robot stops advancing, optionally performs a limited scan, and then returns to idle if the target is not reacquired.

Behavioral Output from the Vision Layer

The output of the vision system should remain abstract and behavior-oriented rather than directly specifying raw PWM values. In practice, the vision process should transmit fields such as whether a target is visible, the horizontal position of the target within the image, the current distance zone, the estimated approach state, the current risk level, the active behavior state, a motion command, an interaction mode, and a normalized speed scale. A representative output might specify that a target is visible, located on left of center, currently near, approaching, associated with medium risk, and that the robot should enter caution mode while turning left slowly and switching to an alert rhythm.

Serial Command Protocol

Communication from Python to the ESP32 should use a simple and robust line-based protocol. A compact text packet is usually the easiest approach for early development because it is lightweight and straightforward to parse on the microcontroller. For example,

commands may encode state, linear motion, angular motion, interaction mode, tempo, LED mode, and sound mode in a single delimited line. JSON can also be used if readability is more important than parser simplicity, but for a small ESP32-based controller, compact CSV-like or tokenized packets are generally more reliable and easier to debug.

Vision Processing Rate

The camera should ideally capture frames at about 20 to 30 fps. Depending on the chosen model and hardware, person detection may run effectively at around 10 to 20 fps, which is sufficient for this interaction scenario. Command updates sent to the ESP32 do not need to occur at video frame rate; a rate of roughly 10 to 15 Hz is normally adequate for smooth behavior. To reduce jitter, tracking outputs should be smoothed over a short temporal window of approximately 3 to 8 frames.

Failure Handling

The vision system must be robust to common real-world failures such as loss of target, multiple visible people, unstable lighting, and temporary occlusion. The control strategy should fail safely: if the target is lost, the robot should stop forward motion immediately, avoid aggressive movement, and, if necessary, continue only with slow turning or scanning behavior. If the target is not reacquired within a timeout period, the interaction layer should revert the robot to its idle mode. This ensures that perception uncertainty does not produce unsafe or confusing behavior.

2.3 ESP32 Control System Design

The ESP32 functions as the real-time execution layer of the robot, bridging high-level decisions from the vision system with low-level hardware actuation. It receives abstract commands and translates them into coordinated control of the differential drive motors, LED-based interaction signals, audio output through a buzzer or speaker, and optional servo movements. In addition, it ensures system safety by enforcing timeouts and preventing unsafe motion when communication is lost or invalid.

The ESP32 is well suited for this role due to its enough GPIO and PWM channels for motor and interaction control, as well as its built-in Bluetooth and Wi-Fi capabilities for flexible communication. Its timing performance is adequate for concurrent motor control and synchronized interaction behaviors, and it integrates easily with common development environments such as Arduino and ESP-IDF.

2.3.1 Responsibilities

At runtime, the ESP32 must continuously receive and parse incoming commands, maintain awareness of the communication link state, and convert high-level motion directives into appropriate motor signals. It is also responsible for running the interaction subsystem independently, ensuring that LED patterns, sound generation, and servo gestures remain smooth and synchronized even between command updates. A critical responsibility is enforcing safety behavior, including stopping motion if communication fails and managing concurrent timing demands across motors, audio, and lighting.

2.3.2 Software Architecture

The control firmware is best structured as a set of concurrent tasks. A communication task handles incoming serial, Bluetooth, or Wi-Fi packets and updates the current command state. A motion control task converts linear and angular velocity commands into left and right motor PWM signals while applying acceleration limits for smooth motion. An interaction task manages LED animations, generates tones or rhythmic sequences, and updates any servo gestures. Overseeing all of these is a safety supervisor, which monitors a heartbeat signal and triggers an emergency stop if commands are not received within a defined timeout period.

The robot uses a TB6612 motor driver connected to two DC motors in a differential drive configuration. Each motor is controlled by two digital direction pins and one PWM signal, typically mapped as AIN1, AIN2, and PWMA for the left motor, and BIN1, BIN2, and PWMB for the right motor, with a shared standby pin. Motion commands are expressed in terms of linear velocity, which are converted into individual wheel speeds using a simple differential mapping where the left wheel speed is v-kw and the right wheel speed is v+k·w, followed by clamping to valid PWM ranges. This enables forward and reverse motion, in-place rotation, curved trajectories, and full stops. To maintain both safety and expressiveness, motor commands should be smoothed by gradually ramping PWM values, avoiding abrupt reversals, and enforcing a maximum acceleration step per control cycle.

Commands from the vision system should arrive at a rate of approximately 10 to 15 Hz, which is sufficient for responsive behavior without overloading the microcontroller. The ESP32 should monitor the timing of incoming packets and, if no valid command is received within roughly 500 milliseconds to one second, it must immediately stop the motors, transition to a safe idle interaction mode, and optionally indicate the fault through a status LED. This ensures that communication loss does not result in uncontrolled motion.

Internally, the ESP32 maintains a state machine that mirrors the behavior state received from the vision system while also enforcing its own safety constraints. Typical states include BOOT, IDLE, ENGAGE, CAUTION, ESCAPE, COMMS_LOST, and ESTOP. Under normal operation, valid commands transition the system into states such as ENGAGE or CAUTION, while a high-risk command triggers ESCAPE. If communication is lost, the system enters COMMS_LOST, and in cases such as low battery or critical faults, it transitions into ESTOP or a reduced-power mode. This layered state handling allows the robot to remain responsive while still protecting itself and the user.

To support the music wheel concept, the ESP32 should include a centralized timing scheduler that coordinates all interaction outputs. At each update cycle, the system selects the appropriate LED pattern, note sequence, and servo position, ensuring that all outputs remain synchronized. This synchronization is typically driven by a shared tempo parameter, such as a BPM value, which governs the timing of beats, light transitions, and motion accents. By aligning all modalities to the same temporal structure, the robot achieves a cohesive and expressive behavior rather than disjointed outputs.

2.4 Motion System

The motion system is responsible for executing the behavioral logic of the toy car: moving away when the infant approaches, and moving closer when the infant retreats or remains stationary. This behavior is achieved through differential steering using two independent DC motors, each driving one rear wheel. The front wheel is an omnidirectional ball caster that provides stability without restricting movement.

Hardware Components:

2x DC Geared Motors (6V, 200 RPM)

2x Motor Drivers (TB6612FNG or L298N)

1x Omnidirectional Ball Caster

1x Ultrasonic Sensor (HC-SR04) or LiDAR (TF-Luna) for distance measurement

1x Microcontroller (ESP32)

Motion Logic:

Let d(t) be the distance between the car and the infant at time t, and $d_{threshold}$ be the desired following distance. The motion is governed by:

$$v(t) = \begin{cases} +v_{max}, & if\ d(t) < d_{threshold} - \delta \\ -v_{max}, & if\ d(t) < d_{threshold} + \delta \\ 0, & otherwise \end{cases}$$

where v_{max} is maximum speed and δ is a hysteresis band to prevent oscillation.

Differential Steering:

The car rotates by running the two rear motors in opposite directions:

Turn left: Right motor forward, left motor backward

Turn right: Right motor backward, left motor forward

Move forward/backward: Both motors forward/backward

The direction toward the infant is assumed to be straight ahead, as the ultrasonic sensor is forward-facing. If a multi-directional sensor array is used, the car can also turn to face the infant before moving.

2.5 Interaction System

The interaction subsystem transforms the robot from a purely functional machine into an expressive, socially engaging entity. Rather than only reacting through motion, the robot communicates intent, emotion, and state through coordinated sound, light, and movement, creating the impression of a playful “music wheel” that responds dynamically to human presence.

2.5.1 Interaction Hardware

The core interaction hardware consists of LEDs, either simple or RGB, and a buzzer or small speaker driven by the ESP32’s PWM or tone-generation capabilities. For enhanced expressiveness, a small servo can be added to drive a decorative or gestural element, and a circular LED arrangement such as a WS2812 (Neopixel) ring is highly recommended to visually reinforce the music wheel concept. A 12- or 16-LED ring is particularly effective for creating rotating or rhythmic light patterns.

2.5.2 Music Wheel Interaction Concept

The defining idea behind the music wheel robot is that it behaves like a mobile musical instrument. Its expression emerges from three tightly coupled channels: sound, light, and motion. Sound consists of tones, short melodies, and rhythmic patterns; light consists of rotating, pulsing, or color-changing LED effects; and motion includes wheel-based movement, directional adjustments, and optional servo gestures. These three channels operate together to convey the robot's internal state and intent in an intuitive and engaging way.

2.5.3 Interaction Modes

In the idle ambient mode, when no person is detected, the robot produces soft, infrequent tones, slow LED pulses, and minimal movement, giving a sense of presence without demanding attention. When a person is detected, the greeting or acquisition mode activates, characterized by a short rising tone sequence, a brightening or expanding LED pattern, and a reorientation of the robot toward the user, signaling awareness. During playful engagement, when the user is at a comfortable distance, the robot generates a rhythmic melody loop, synchronizes its wheel motion to the beat, and displays rotating or alternating LED patterns, optionally accompanied by servo movements that reinforce the rhythm. If the user comes too close or approaches too quickly, the system enters a caution mode, where tones become sharper and faster, LED blinking intensifies, and forward motion is reduced or replaced by slight retreat, signaling discomfort. In the escape mode, triggered by high risk, the robot produces a more urgent or descending alert sequence, displays strong flashing or rapidly rotating light patterns, and actively moves away from the user, clearly communicating a boundary.

2.5.4 Sound Design

Sound output can be implemented using a passive buzzer for simplicity, which allows tone generation via PWM but has limited timbral richness, or a small, amplified speaker for more expressive audio. For the initial prototype, a passive buzzer is sufficient. Musical behavior should be parameterized so that distance, approach speed, and behavior state influence the sound. For example, distant users correspond to slower and softer notes, while closer users result in faster or brighter tones, and high-risk situations trigger alert motifs. Similarly, slow approaches can produce welcoming rhythms, while rapid approaches produce more tense patterns. Distinct tone sets can be assigned to each state, such as simple pulses for idle, a rising triad for greeting, a repeating pentatonic loop for playful interaction, and sharper or descending patterns for caution and escape. Tempo should also vary by state, ranging from

slow ambient tempos around 60–80 BPM in idle mode to faster tempos up to 200 BPM in escape scenarios.

2.5.6 LED Design

The LED subsystem plays a central role in visual expression. While simple LEDs can be used, an addressable LED ring greatly enhances the music wheel effect by enabling dynamic patterns such as rotation, pulsing, and synchronized flashes. Different lighting modes should map directly to robot states, with slow breathing effects for idle, expanding brightness for greeting, rotating circular patterns for engagement, rapid pulses or warm colors for caution, and fast red flashing for escape. These patterns should be synchronized with the audio tempo to reinforce the sense of rhythm.

2.5.7 Optional Servo Gesture

Adding a servo introduces a physical gesture layer that can significantly improve emotional readability. The servo can drive a decorative element or pointer that moves in rhythm with the music, nods, or performs small expressive motions. In idle mode, the motion may be a slow sweep, while in engagement it can bounce to the beat. In caution mode, the movement may become tense or jittery, and in escape mode it can switch to rapid, sharp motions that emphasize urgency.

2.5.8 Coordinated Interaction Engine

To achieve a coherent “musical robot” behavior, all interaction outputs must be synchronized through a shared internal timing system. A simple implementation uses a base time tick, for example every 50 milliseconds, with a beat interval derived from the current tempo in BPM. LED updates, sound triggers, and servo movements are all scheduled relative to this shared beat structure. By aligning all modalities to the same timing reference, the robot produces unified, rhythmic expressions rather than disconnected signals, reinforcing the identity of the system as a music-driven interactive agent.

2.6 Power System

The power subsystem supplies electrical power to all onboard hardware and ensures that each subsystem receives a stable and suitable supply during operation. Its main components are a 2S Li-ion battery, a master power switch, a DC-DC buck converter, and the power distribution network implemented through the onboard wiring and carrier board. Together, these components provide power to the ESP32 control board, the TB6612 motor

driver, the two DC motors in the motion subsystem, and the interaction devices, including the LEDs, buzzer, and optional servo. In this way, the power subsystem acts as the electrical foundation of the entire robot and directly supports coordinated sensing, control, motion, and interaction behavior.

The 2S Li-ion battery serves as the main onboard energy source for the robot during mobile operation. Its output first passes through the master power switch, which allows the system to be turned on and off safely during testing and demonstration. After the switch, power is divided into two paths. One path supplies the TB6612 motor driver, which drives the two DC motors of the 2WD differential-drive chassis. The other path goes through the buck converter, which steps the battery voltage down to a regulated 5 V supply for the ESP32 and the interaction devices.

The motion subsystem places the highest demand on the power system because the DC motors draw relatively large current during startup, turning, and rapid escape maneuvers. For this reason, the battery and power distribution path must provide enough current to the motor driver without causing excessive voltage drop. At the same time, the regulated 5 V rail must remain stable for the ESP32, since the controller is responsible for receiving commands from the vision system and generating outputs for both the motion and interaction subsystems. The buzzer, LEDs, and optional servo also depend on this regulated supply to produce clear and reliable feedback behaviors.

Another important requirement is proper electrical integration among all powered modules. The battery, ESP32, TB6612 motor driver, DC motors, and interaction devices must share a common ground reference so that control signals can be interpreted correctly, and actuator behavior remains stable. The onboard wiring and carrier board therefore function not only as physical connections, but also as the main distribution path for power and ground throughout the robot. Since the camera and laptop-based vision system are external to the robot, the onboard power subsystem only needs to support the mobile platform electronics, which helps keep the design compact and practical while still meeting the operating requirement of at least ten minutes of continuous use.

(1) Requirement 1: Stable regulated supply for control and interaction electronics

The power subsystem shall provide a stable regulated supply for the ESP32 and onboard interaction devices during both idle and active robot operation. Voltage fluctuation must not cause controller reset, communication failure, or unintended interaction behavior when the motors start, stop, or reverse direction.

(2) Requirement 2: Sufficient current capability for motion loads

The 2S Li-ion battery and power distribution path shall support the current demand of the TB6612 motor driver and the two DC motors during startup, turning, and rapid escape motion without excessive voltage sag or loss of motor response.

(3) Requirement 3: Reliable integration of all powered modules

The battery, power switch, buck converter, ESP32, motor driver, motors, LEDs, buzzer, and optional servo shall be connected through a reliable power distribution network with a shared electrical ground to ensure correct signal reference and stable overall system behavior.

(4) Requirement 4: Minimum continuous operating time

The onboard power system shall support continuous robot operation for at least 10 minutes under normal testing conditions, including repeated motion commands and interaction feedback events.

2.7 PCB / Wiring System

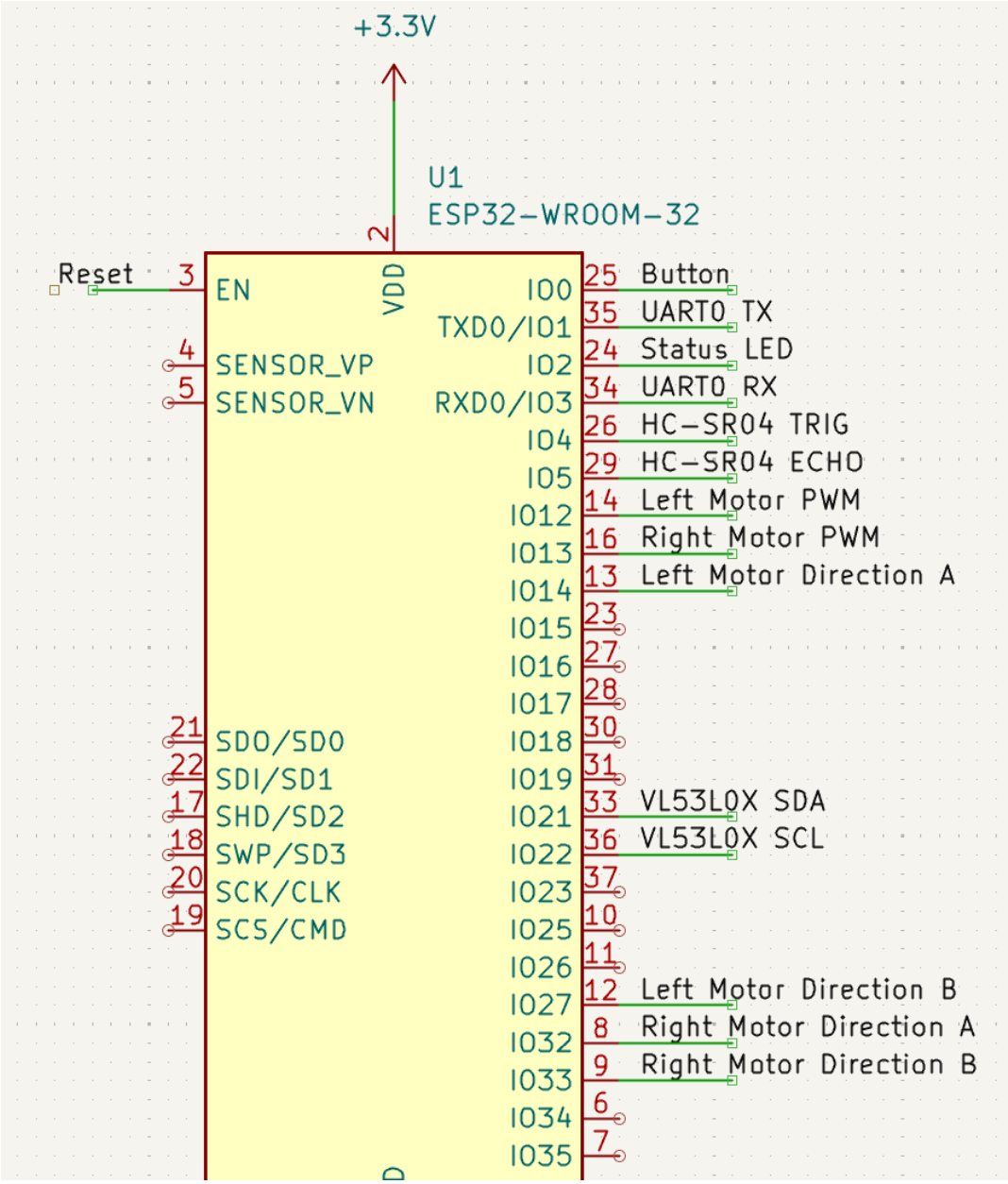


Figure 2: ESP32 Connection

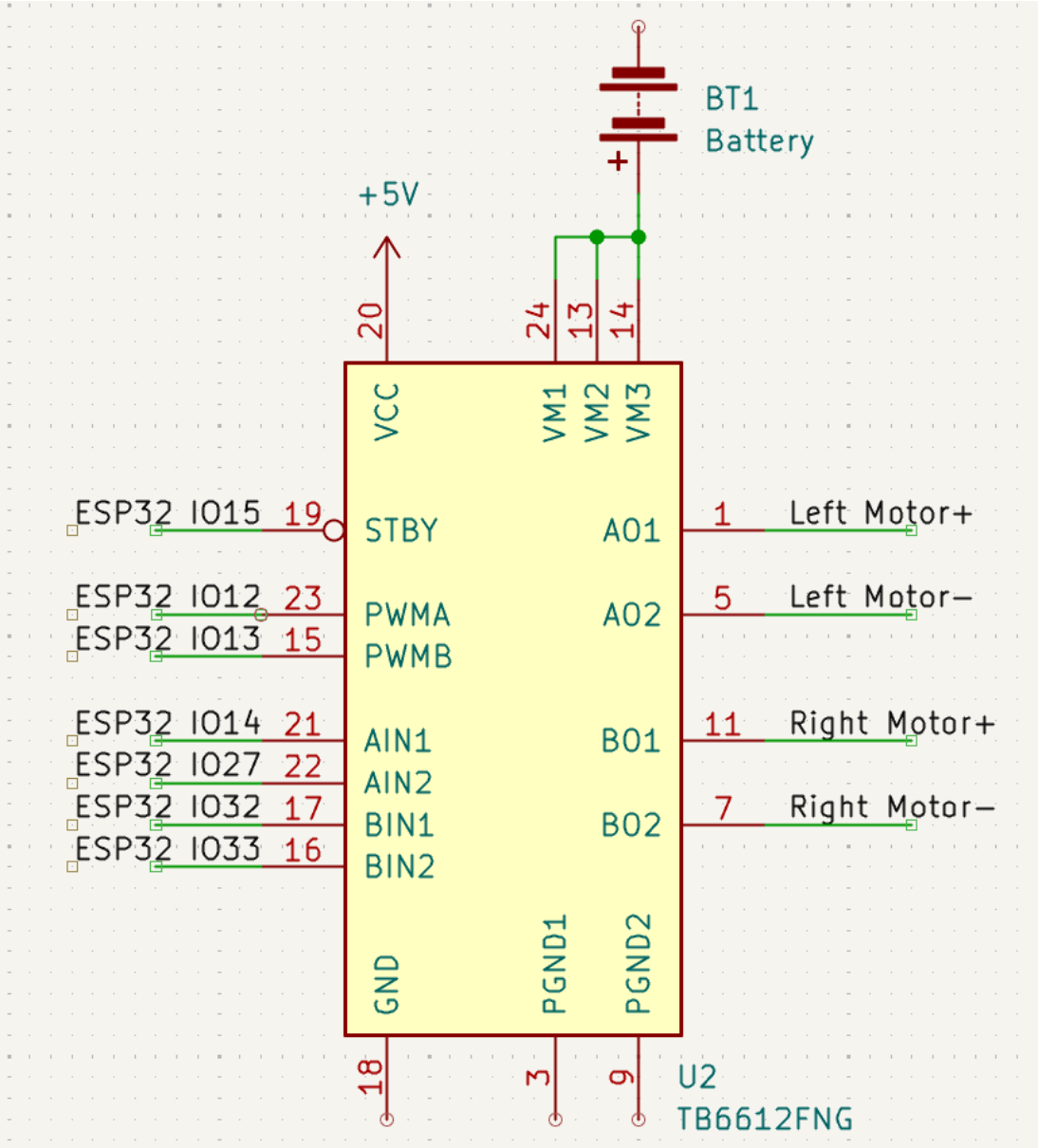


Figure 3: Motor Driver Connection

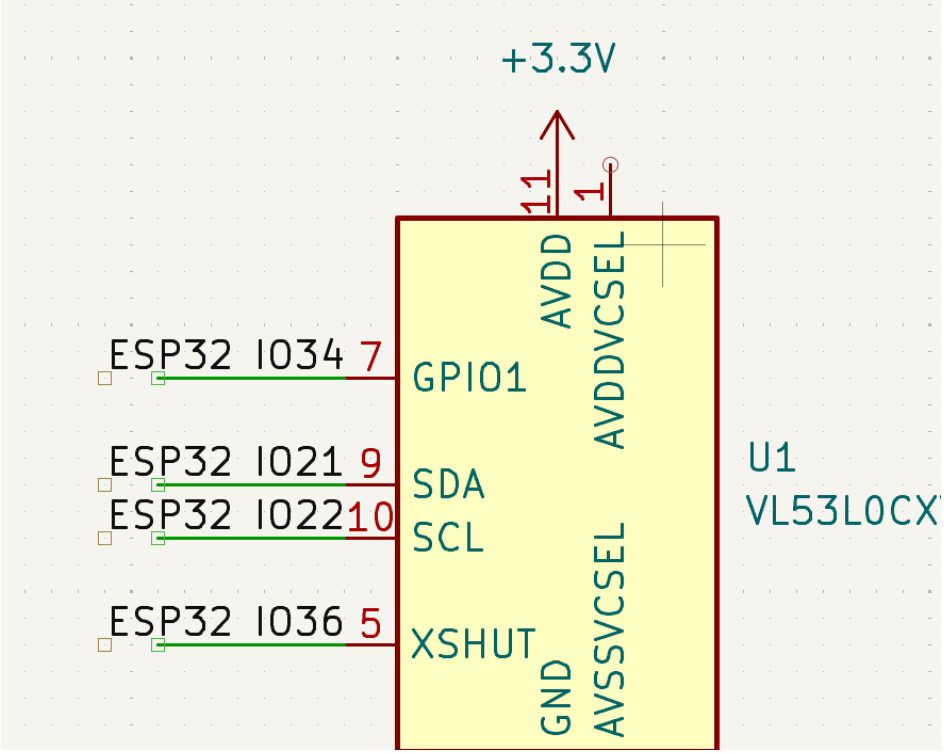


Figure 4: ToF Sensor Connection

2.8 Tolerance Analysis / Risks

The most critical subsystem in the infant-following toy car is the distance measurement and motion response loop. A failure here could cause the car to either collide with the infant or fail to engage, reducing safety and interactivity. This analysis focuses on sensor accuracy, motor response variability, and control loop timing.

(1) Design-Level Risks

Sensor Accuracy: Ultrasonic sensors have beam width (±15°) and noise (≤1 cm), which may cause false distance readings, especially near walls or soft surfaces (clothing).

Motor Variability: DC motors have ±10% RPM tolerance, leading to uneven movement and unintended turning.

Control Loop Latency: Total loop time (sensor read + processing + motor update) must be ≤100 ms to maintain smooth interaction.

Floor Surface Friction: Carpet vs. hardwood changes effective speed and stopping distance.

(2) Distance Measurement Tolerance

Using HC-SR04, typical error model:

$$d_{measure} = d_{true} + \epsilon_{noise} + \epsilon_{angle}$$

where:

$\epsilon_{noise} \sim \mathrm{N}(0,1)$

ϵ_{angle} = up to 5cm if sensor axis is not aligned with infant's reflective surface

(3) Motion Response Tolerance

Let t_{loop} = total feedback loop time. The car's position change during one loop is:

$$\Delta x = v_{max} \cdot t_{loop}$$

To prevent overshoot beyond the desired following range (±5 cm), we require:

$$v_{max} \cdot t_{loop} \leq \delta_{hysteresis}$$

Given δ = 5 cm, this inequality holds with margin.

(4) Simulation and Validation

A MATLAB/Simulink simulation was conducted with:

Distance sensor noise: 1 cm RMS

Motor response delay: 50 ms

Infant movement: ±0.1 m/s sinusoidal motion

Results show the car maintains distance within $d_{threshold}$ ±6cm under typical conditions, meeting safety and interaction requirements.

(5) Conclusion

The motion system design is feasible within the stated tolerances. Sensor noise and motor variability are within acceptable limits if median filtering and hysteresis control are implemented. Future hardware testing will validate real-world performance on carpet and hardwood floors.

3. Cost Estimate

Item	Model/Spec	Qty	Cost (CNY)
ESP32	ESP32-C3 SuperMini	1	30
Motor driver	TB6612FNG	1	25
Chassis	Includes 2 motors + 2 wheels + 1 caster wheel	1	38
Battery + power	3.7V 2000mAh	2	30
LEDs +buzzer + servo	/	/	50
Camera (or phone)			0–150
Misc materials			100
PCB	JLCPCB, 10cm×8cm	5	25
Lovely appearances	toys	1	20

Total: ~300–500 CNY

4. Timeline

Week	Tasks
Week 1	Order all components (ESP32-C3, TB6612, chassis, battery, MT3608, buzzer, LEDs, resistors, SG90, CH340, soldering kit, webcam) Design schematic and PCB layout Export Gerber and order PCB Install Arduino IDE, Python, OpenCV Deliverables: Components ordered, PCB ordered, dev environment ready.
Week 2	ESP32: motor control (forward/back/left/right/stop) ESP32: buzzer, LEDs, servo control ESP32: serial command parser (F, B, L, R, S, BEEP, LED_ON, WIGGLE, etc.) Python: OpenCV webcam capture, object/color detection Python: send serial commands to ESP32

	Deliverables: Complete robot.ino and vision.py code.
Week 3	Solder PCB and mount onto 2WD chassis Connect motors, battery, and all components Flash ESP32 code, test all functions Position overhead webcam, run Python script Implement reactive baseline: if distance < threshold → backward() Deliverables: Fully assembled robot, reactive avoidance working.
Week 4	Implement predictive controller: predict hand position in 0.5-1 sec Compare reactive vs predictive (contact rate, response time) Record demo video (2-3 minutes) Prepare documentation (block diagram, BOM, results table)

5. Ethics and Safety

Ethics

- The robot operates at low voltage (3.7V – 5V), eliminating electrical shock hazards. All electronic components are enclosed within the chassis, preventing accidental contact with live circuits. The 18650 batteries are protected against over-discharge and short circuits. Moving parts (wheels, servo) are designed to avoid pinching or entrapment.
- The primary safety feature is the predictive avoidance algorithm. Unlike reactive systems that only respond after contact, the robot predicts hand position 0.5–1 second in advance and retreats before physical contact occurs. This prevents collisions, falls, or accidental impacts that could harm a child or damage the robot. All movements are smooth and predictable, avoiding sudden jerks that could startle a user.

Safety

- This robot is designed as a research prototype for studying predictive human-robot interaction. It is clearly presented as a mechanical system, not a social companion. It does not simulate emotions, express attachment, or attempt to deceive users about its capabilities. The system is not intended to replace human caregiving or supervision.
- The overhead camera performs real-time object detection only. No video frames are stored, and no personally identifiable information is captured or logged. The camera is positioned to view only the robot's workspace, not faces or private areas.

- This project complies with the IEEE Code of Ethics, which states: "to hold paramount to the safety, health, and welfare of the public." All design decisions prioritize user safety over performance or cost. The robot is intended to supplement, not replace, responsible for adult supervision during any interaction with infants or children.