# Jarvis Desktop Robot

## Interactive Desktop Companion for Stress Relief

ECE 445 Senior Design • Spring 2026

---

Team 6: Jiajun Gao, Yuchen Shih, Zichao Wang

# Project Introduction

## What is Jarvis?

A compact, interactive desktop assistant built on an ESP32-S3.

## Voice Interaction

Natural language processing for seamless user engagement.

## Dynamic UI

Expressive digital face providing emotive visual feedback.

## Connectivity Hub

Integrated wireless stack for smart environment control.

# Objective & Problem Statement

## The Problem

Prolonged seated work and sustained screen exposure lead to severe mental fatigue, accumulated stress, and significantly reduced cognitive efficiency over long hours.

## Current Limitations

Existing passive desk toys offer zero contextual awareness. Current consumer robots are usually too large, mechanically complex, or lack strict safety constraints for desks.

## The Solution

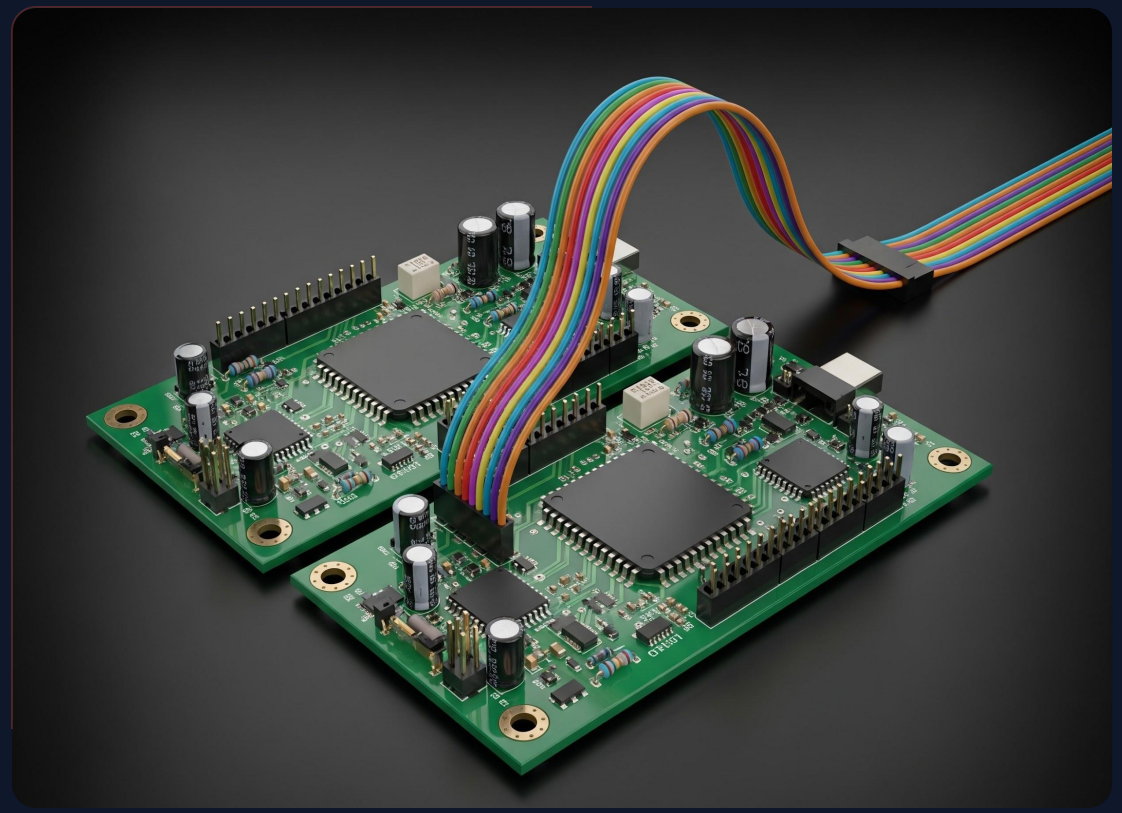
A lightweight, self-aware companion providing low-effort conversational interaction and physical feedback

# Design Evolution

## Dual-PCB Modular Architecture

We are using a separate "Face Board" and "Body Board" to optimize hardware distribution.

These two distinct modules were designed to be connected via **dupont cable** to isolate subsystem development and debugging.



# Custom PCB Architecture

## Face Board

### Integrated Audio

Combines an ADC microphone on GPIO4 with a speaker and power amplifier for high-fidelity audio handling.

### Visual & Expansion

Directly drives the ST7735 80x160 SPI LCD

# Custom PCB Architecture

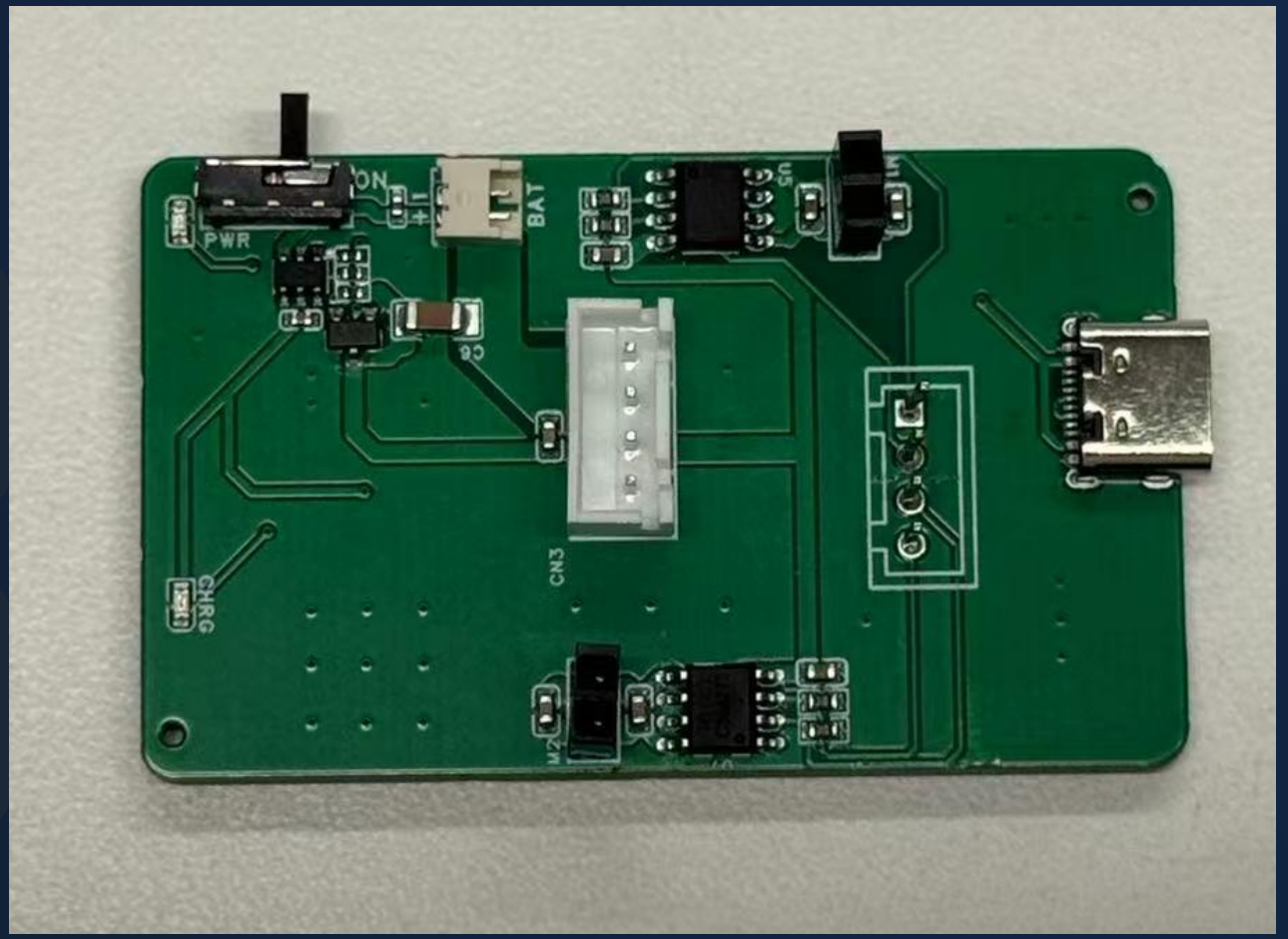
## Body Board

### IMotion Control & Power Isolation

The Body Board serves as the drive hub for the robot's physical movement.

### Seamless Signal Interconnect

Communication between the two modules is handled via a single physical link: ESP32 PWM control signals transmitted through Dupont wiring.



# Safety & Edge Detection (IR)

## Hardware Setup

Infrared (IR) sensors utilized for both forward obstacle and downward cliff detection.

## Implementation Pros

- Extremely fast response times
- Simple digital GPIO interface
- Cost-effective implementation

## Implementation Cons

- Susceptible to ambient light interference
- Varying surface reflectivities (e.g., fails on matte black)

## Control Priority & Safety

IR flags are continuously polled to allow immediate override of motor PWM outputs, ensuring the highest level of safety during operation.

# High Level Requirements

### HR1 – Cloud Connectivity

Robot connects to Wi-Fi and communicates with the cloud AI server within **5 seconds**.

### HR2 – Voice Interaction

Robot supports voice conversation with user, average response latency **under 6 seconds**.

### HR3 – Lift / Edge Safety

Robot detects when lifted or no longer on desk using the bottom IR sensor.

### HR4 – Front Obstacle

Robot detects obstacles ahead using front IR sensor with **90% stop success rate**.

### HR5 – Emergency Stop

Robot stops motors immediately when hazard detected, within **50 ms**.

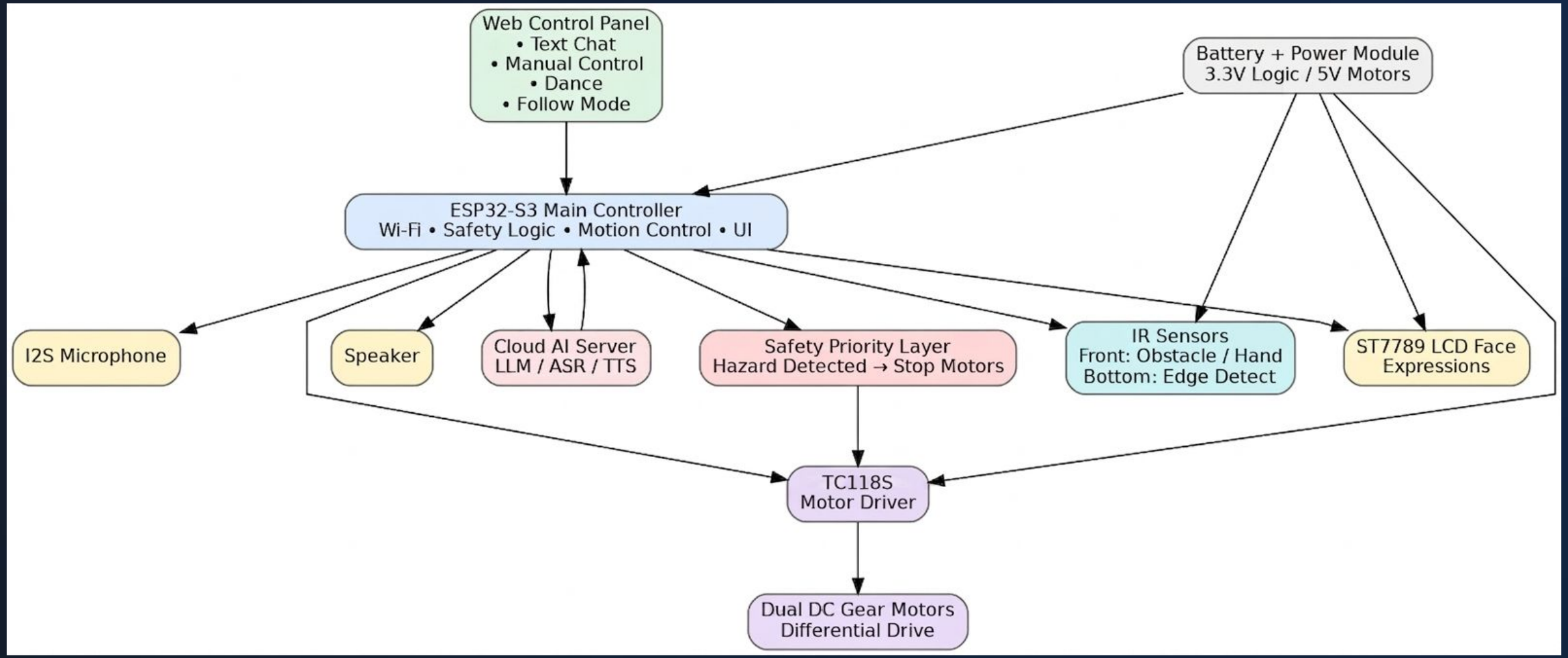
### HR6 – LCD Expression

Robot displays facial expressions and system states correctly in all operating modes.

### HR7 – Battery Stability

Robot runs stably on battery power with no reset during a **30-minute operation**.

# Block Diagram



# Software Architecture

**Device Layer (Firmware)** : The robot listens for the wake word and sends compressed audio to the backend through WebSocket.

**Server Layer (Mac Backend)** : A local Mac backend receives the audio and uses voice activity detection to separate speech from noise.

**Fast-Path ASR & LLM :** Speech is converted into text, then sent to Gemini 2.5 Flash to understand the user's command.

**Streaming TTS Output** : The response is converted back into speech and streamed to the robot in small chunks to reduce latency.

# Model Context Protocol: Interaction & Control

## MCP Bridge

- Connects LLM to robot hardware
- Uses WebSocket
- Sends JSON-RPC tool calls

## Motion Tools

- robot.move
- Direction + speed input
- Returns IR / obstacle safety status

## UI & Mode Tools

- Syncs face with activity
- Switches AI / human control
- Supports Wi-Fi driving

# Visual Expression Subsystem


## Geometric Minimalism

We use a simple geometric style to design the robot's face, ensuring a clean aesthetic and efficient rendering.

### Design Principles

- Basic shapes: circles, arcs, and lines
- No complex images or emoji graphics

### Smooth Animations

- Transition speed: 200–500 ms
- Performance: 30–60 FPS via double buffering (LVGL)

### State-Driven Expressions

Matches state: **Listening, Thinking, Speaking, Moving, Idle**.

# Safety & Sensing Logic

**2 mm**
Max Travel Between Polls

**20 ms**
Sampling Interval

**0.1 m/s**
Max Robot Velocity

## Fast IR Safety Check

- IR sensor checks every **20 ms**
- Robot speed: **0.1 m/s**
- Max movement per check: **2 mm**

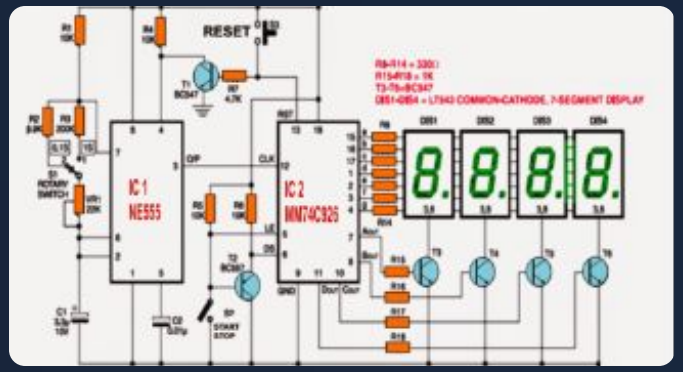
$$d = v_{max} \cdot t = 0.1\ m/s \cdot 0.020\ s = 0.002\ m$$

## Hazard Override

- IR sensor runs at **high RTOS priority**
- If desk edge or obstacle is detected:

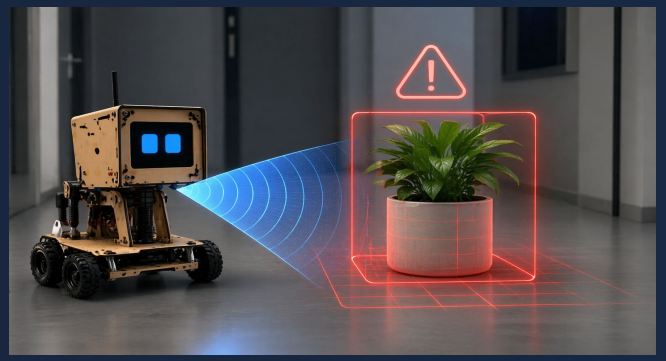
• Motor PWM immediately set to **0**
• Robot stops before falling or hitting objects

# Functional Test Results




## Conversational Latency

Consistent sub-4s end-to-end latency achieved on typical campus Wi-Fi networks, ensuring natural interaction flow.



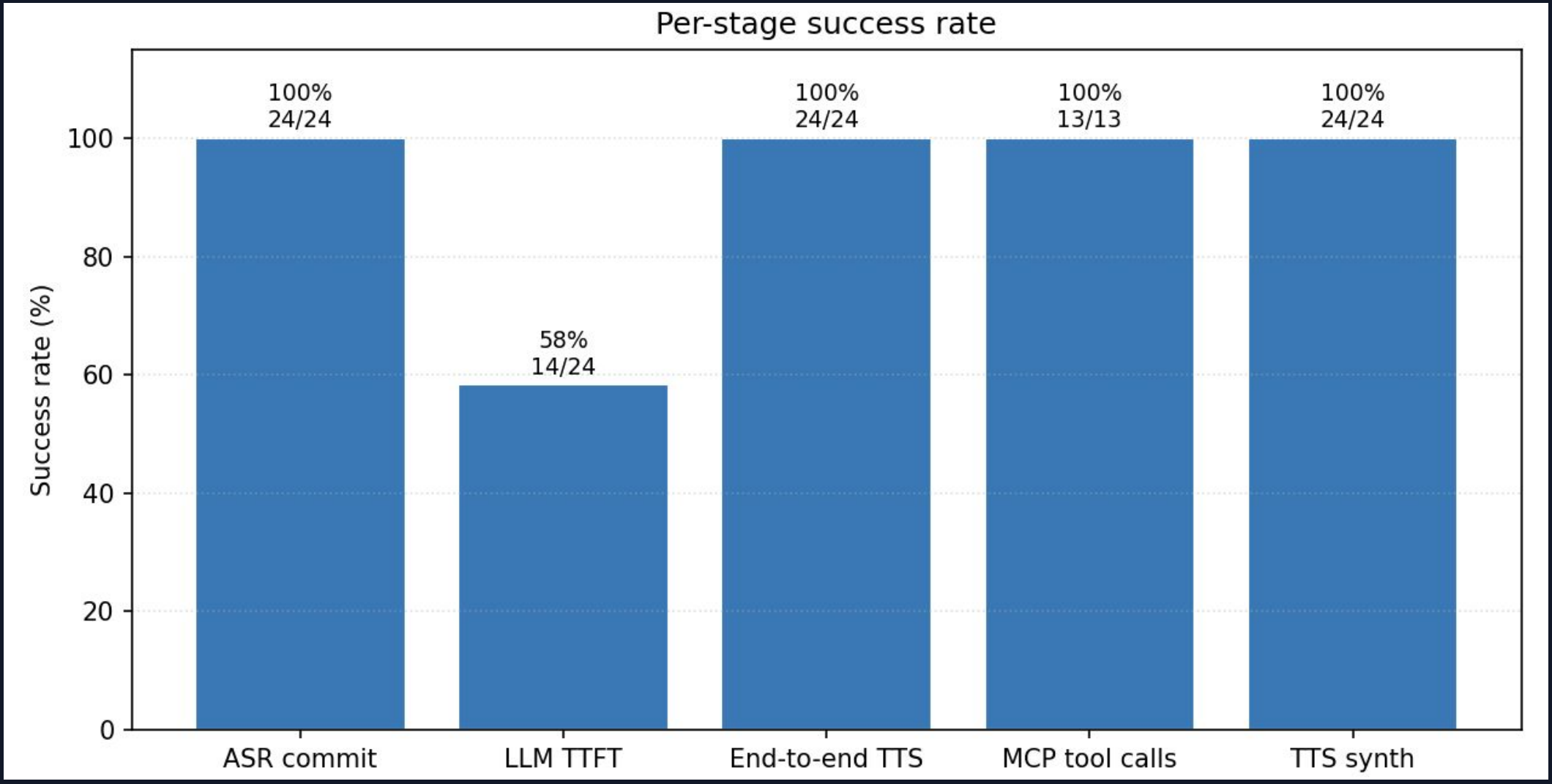
## Safety Detection

100% cliff detection success rate over multiple 10-minute continuous motion tests. Obstacle avoidance functions flawlessly.

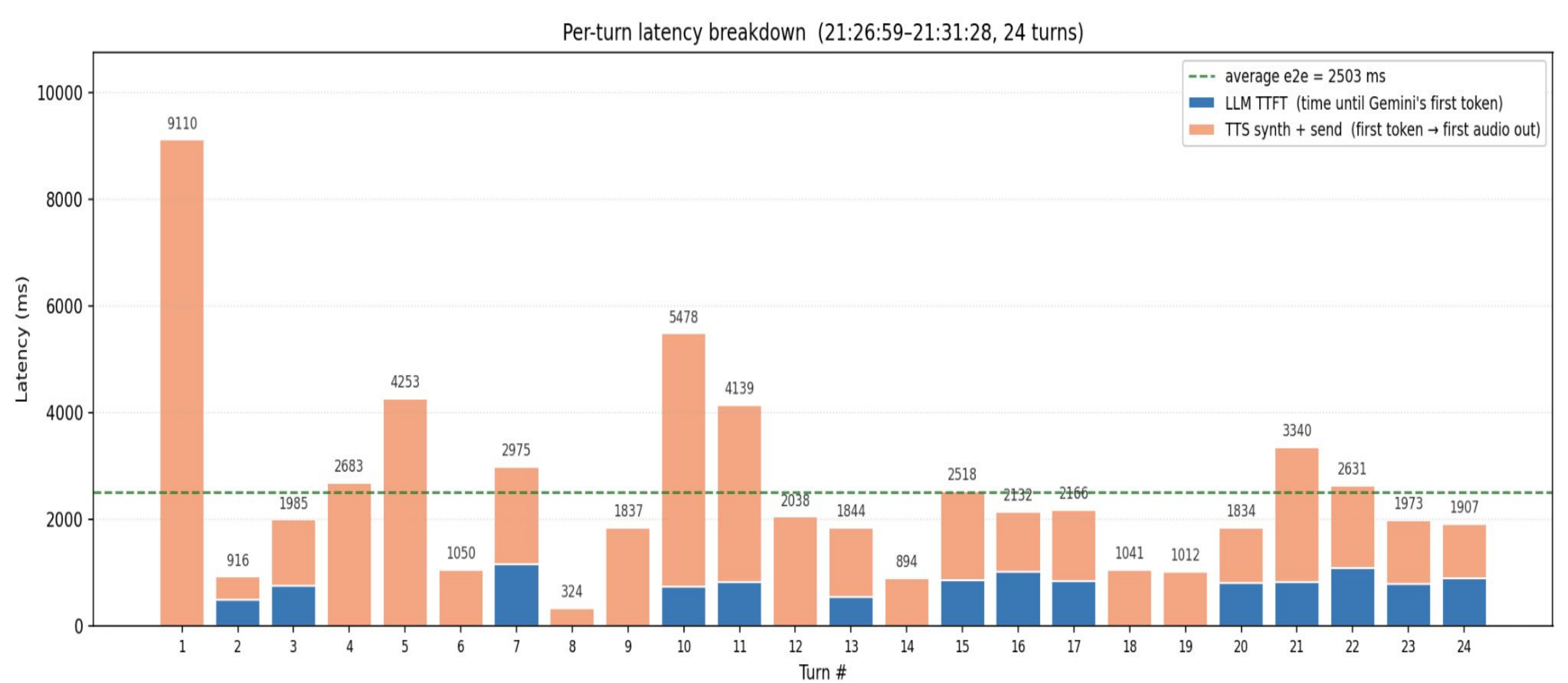

## Human Control Mode

Verified 'Human Control' mode successfully blocks AI voice-motion, enabling safe, manual Wi-Fi driving via browser.

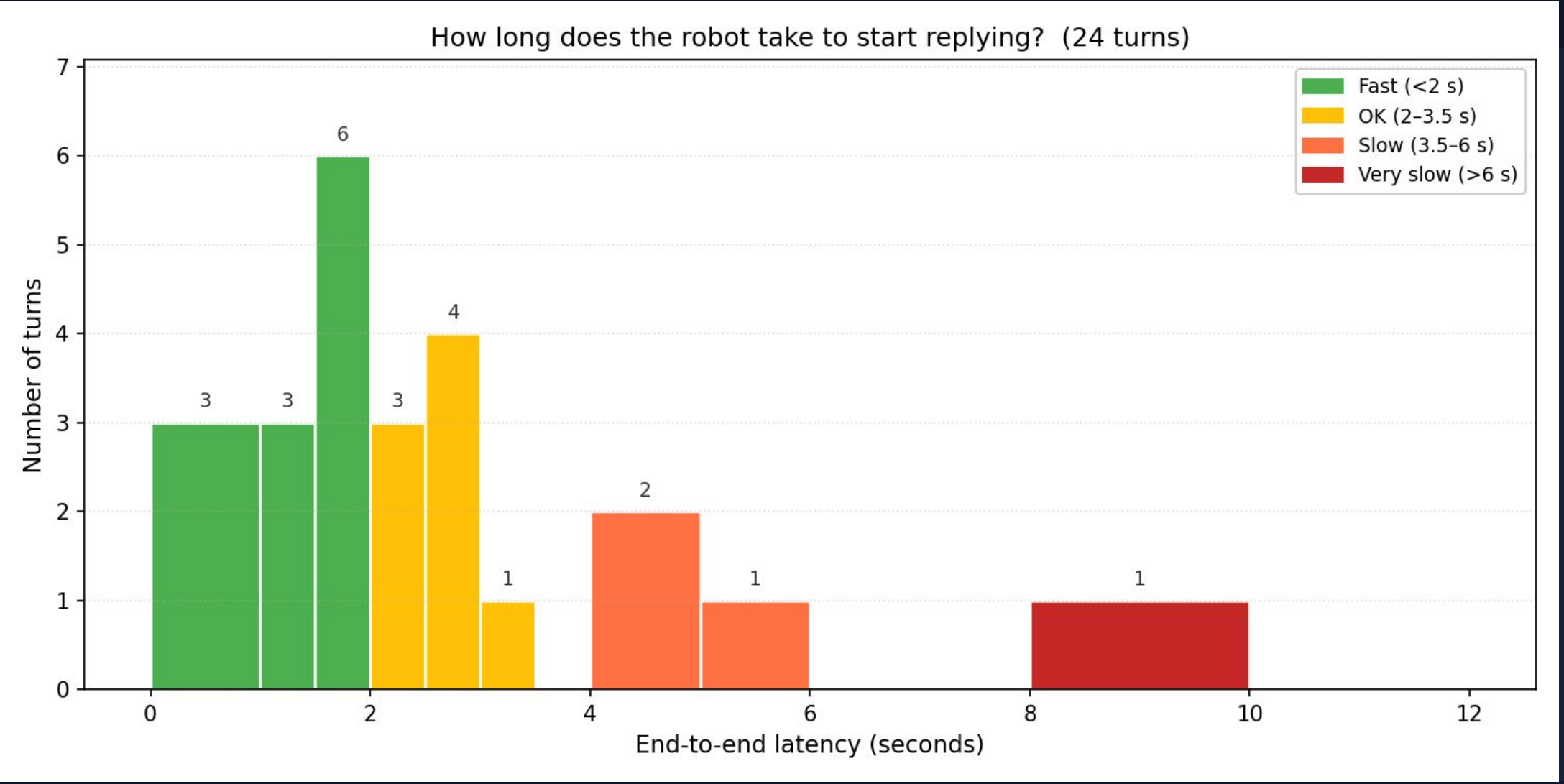
# System Performance



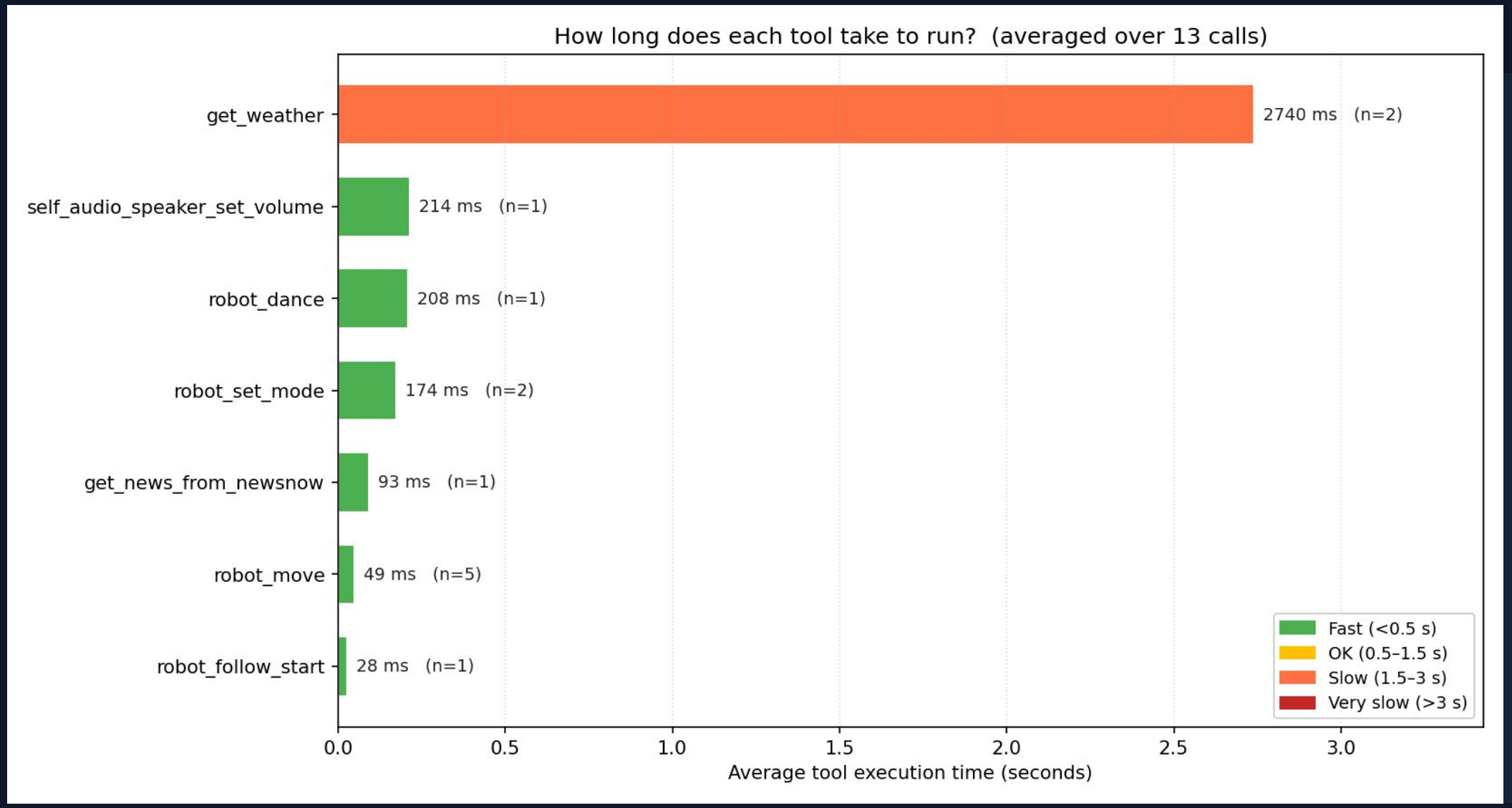
# System Performance: Latency Analysis



# System Performance: Response Latency



# System Performance: Tool Execution Latency



# Success Rates & Limiting Factors

## Software-Level Software Reliability: 100%

- **LLM API:** 90% Success Rate (no HTTP 4xx errors or null responses)
- **MCP Tooling:** 100% Invocation accuracy (JSON-RPC)
- **TTS Engine:** 100% Reliability (Chunked streaming)

## Primary Performance Bottleneck

- **Automatic Speech Recognition (ASR) Accuracy:**

**~55% (Battery) vs. 95% (Wall Power)**

## Root Cause Analysis

- Hardware Limitation: Supply rail Voltage Sag
- Module Affected: Analog Front-End (AFE)
- Signal Impact: Reduced Acoustic SNR
- System Result: Downstream ASR performance loss

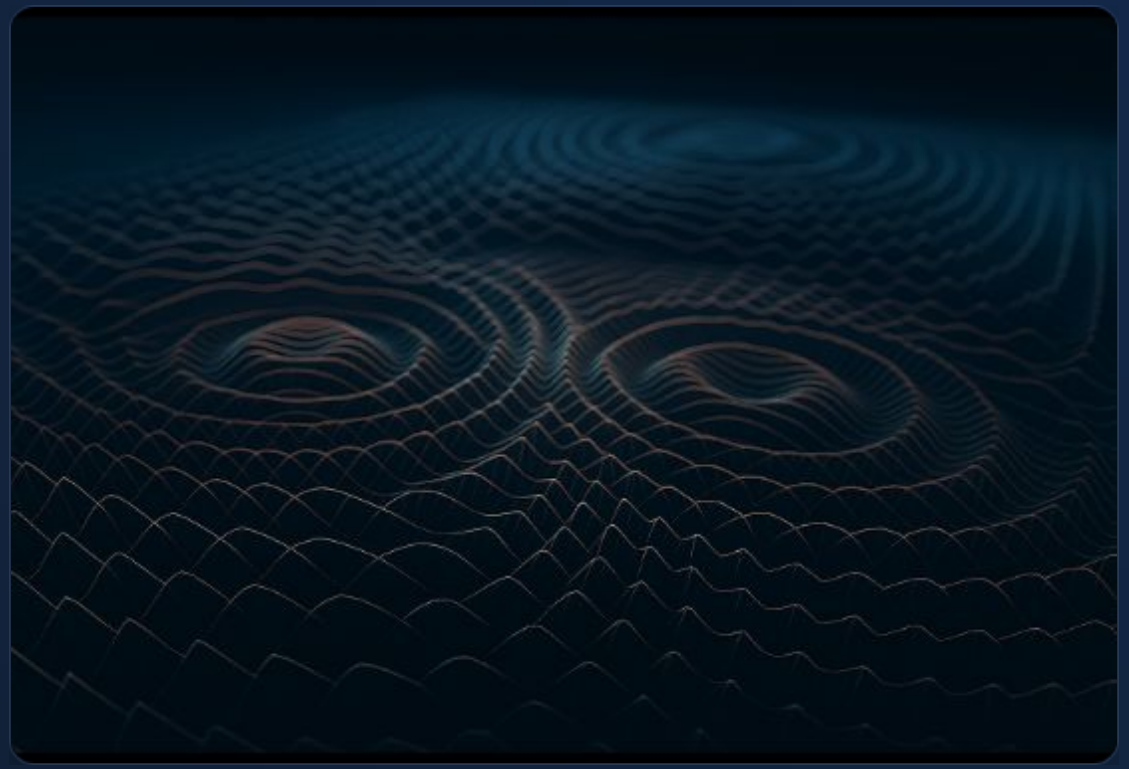
# Challenge: Acoustic Echo & Feedback Loop

## The Failure

- Physical Coupling: Speaker & Mic proximity
- TTS Self-Capture: Loopback of playback signal
- Result: False ASR triggers & unstable loops

## Mitigation Strategy

- **Software-First Approach**: Compensating for lack of dedicated AEC DSP
- **Server-Side "Echo Guard"**: Time-windowing filter for reflection suppression
- **Firmware Grace Period**: Post-TTS delay to block mic reactivation



# Technical Challenge: AI Latency vs. Real-Time Safety

AI commands are slow and sometimes uncertain, but motor safety must be immediate.

## Problem

- Speech + tool calls have seconds of jitter
- AI may misread user intent
- Motors need fast, repeatable control

## Solution

- ESP32 makes final safety decisions
- IR sensors run continuously
- Emergency stop overrides all commands
- Forward motion blocked if obstacle detected
- Watchdog prevents unsupervised motion

**Key Takeaway: The LLM can suggest actions, but the ESP32 decides whether motion is safe.**

# System Outcomes & Key Learnings

### System Integration Outcome

- Unified E2E Architecture
- WebSocket Audio Streaming
- LLM-MCP Orchestration
- RTOS-based LVGL UI

### Power Integrity

- Voltage Sag: Critical AFE drop
- Transient Noise: Unstable DC rail
- SNR Loss: Direct signal interference
- ASR Failure: Battery-mode reliability gap

### Latency Optimization

- Audio Streaming: Chunk-based delivery
- WebSocket Link: Persistent E2E connection
- JSON Payloads: Structured data exchange
- Real-time Flow: Minimized system overhead

# Future Work & System Roadmap

## Sensing System Upgrade

- IR to ToF Migration
- Surface-Independent Sensing
- Eliminate Reflectivity Bias

## Hardware Acoustic Echo Cancellation (AEC)

- Dedicated AEC DSP
- Full-Duplex Audio
- Reliable Barge-in Support

## Vision-Based Context Awareness

- OV5640 Camera Integration
- VLM-Driven Understanding
- Contextual Desktop Interaction

## Locomotion & Mechanical Optimization

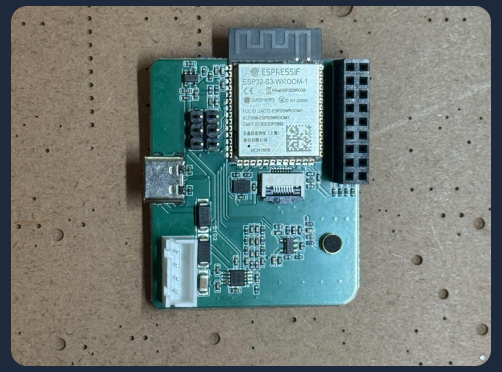
- Tracked Chassis Evaluation
- Surface Adaptability Testing
- Enhanced Traction & Stability

# Thanks for Listening

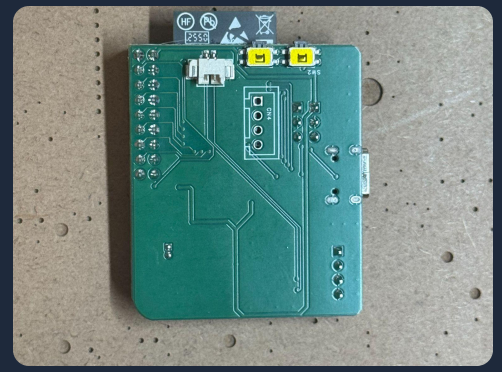
## Q & A

# Hardware Components Overview

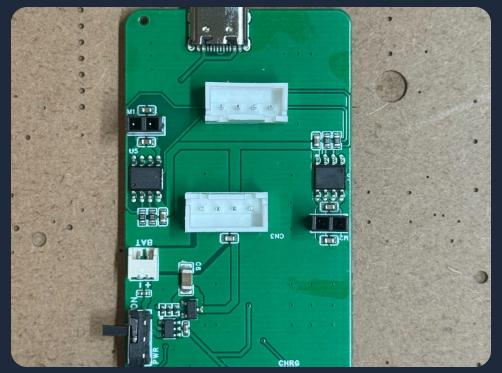
## ESP32-S3 Main Board (Front)



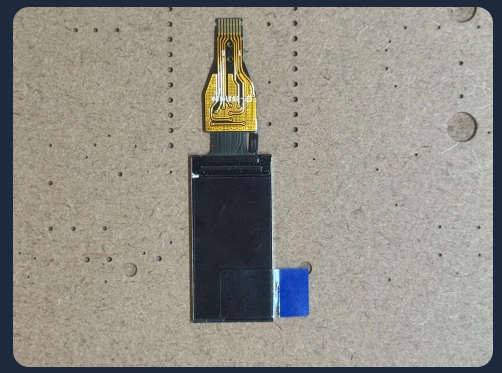
## ESP32-S3 Main Board (Back)



## Power & Motor Driver Board

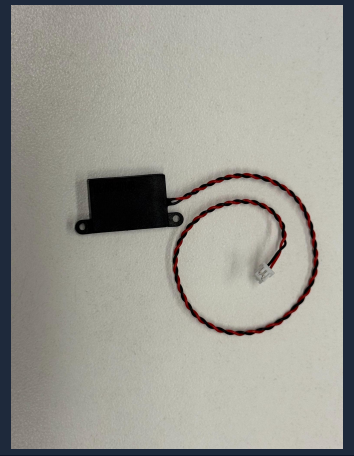


## ST7735 LCD Module

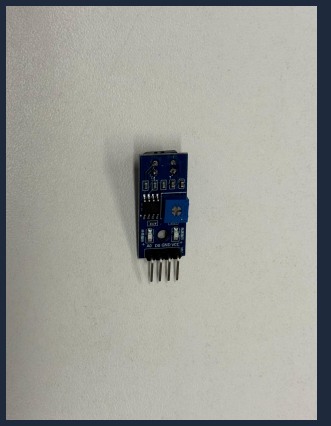


# Hardware Components Overview

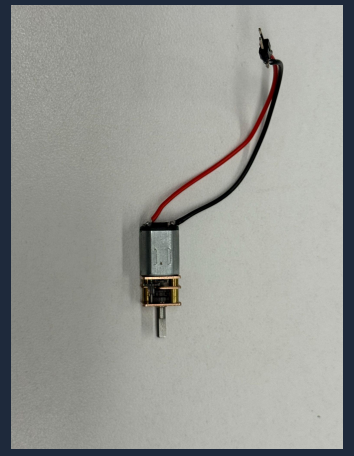
## Speaker



## IR

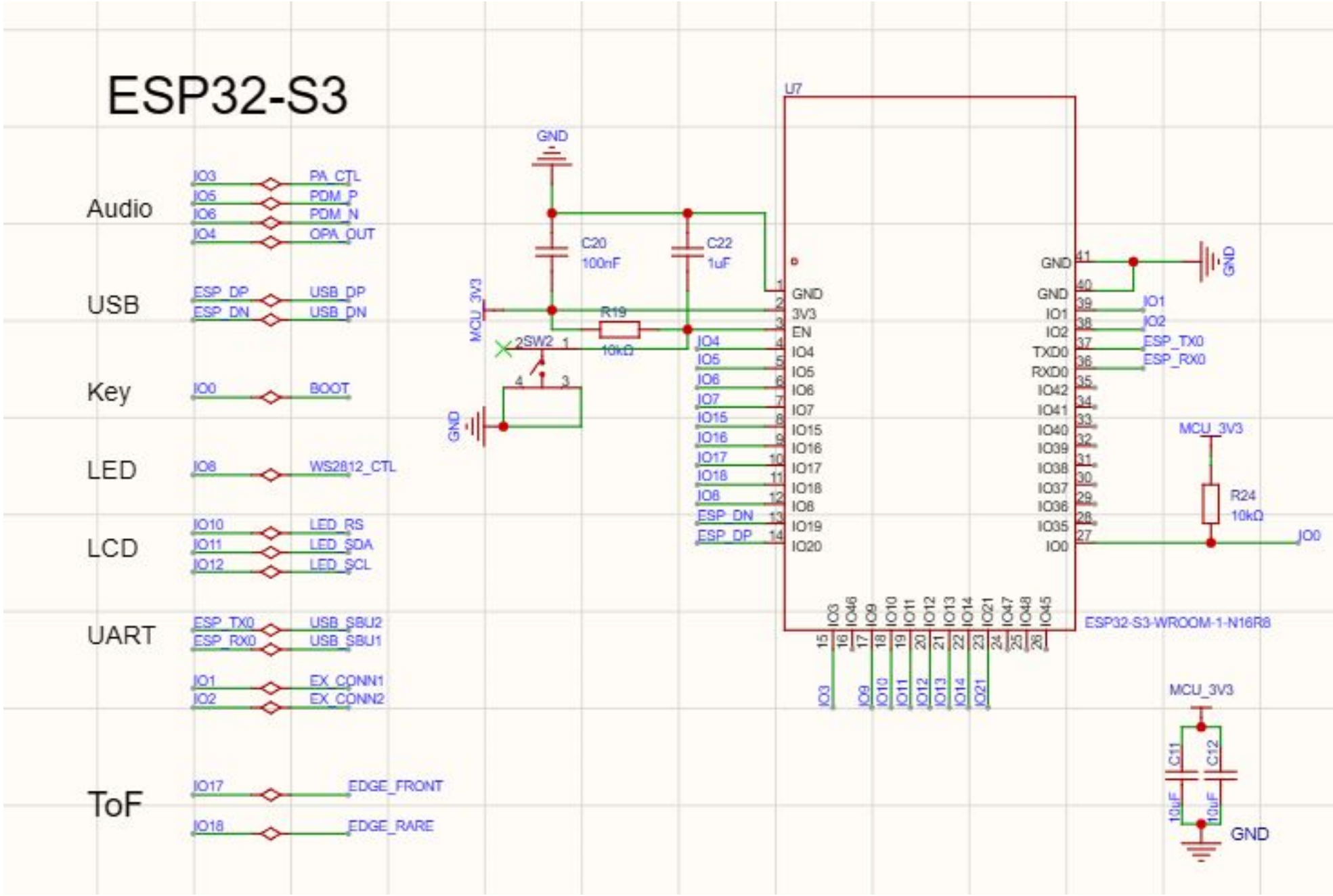


## Motor

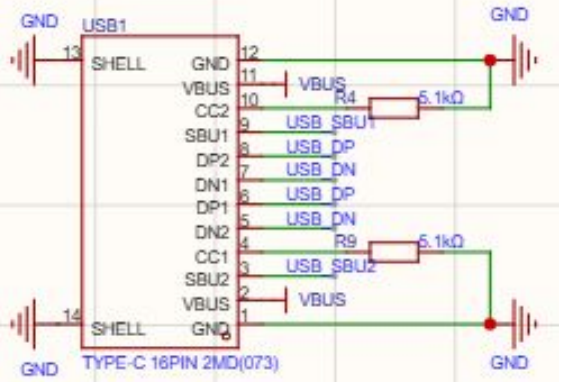


## Battery

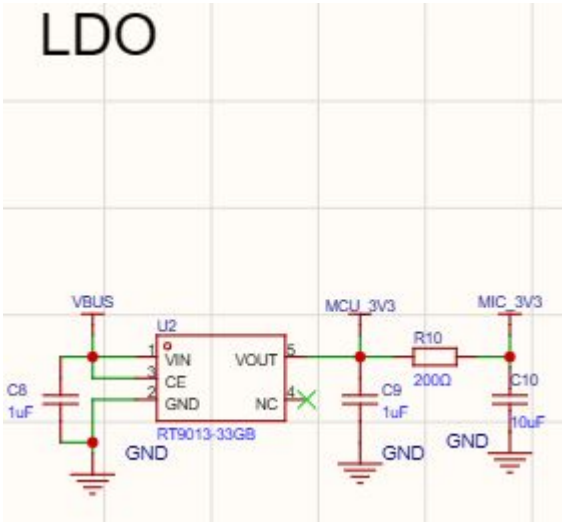
# Appendix:


ESP32-S3
Audio
IO3 PA_CTL
IO5 PDM_P
IO6 PDM_N
IO4 OPA_OUT
USB
ESP_DP USB_DP
ESP_DN USB_DN
Key
IO0 BOOT
LED
IO8 WS2812_CTL
LCD
IO10 LED_RS
IO11 LED_SDA
IO12 LED_SCL
UART
ESP_TX0 USB_SBU2
ESP_RX0 USB_SBU1
IO1 EX_CONN1
IO2 EX_CONN2
ToF
IO17 EDGE_FRONT
IO18 EDGE_RARE
GND
C20
100nF
C22
1uF
MCU_3V3
R19
10kΩ
2SW2
U7
GND
3V3
EN
IO4
IO5
IO6
IO7
IO15
IO16
IO17
IO18
IO8
ESP_DN
ESP_DP
IO19
IO20
IO1
IO2
ESP_TX0
ESP_RX0
R24
10kΩ
IO0
ESP32-S3-WROOM-1-N16R8
C11
C12
10uF

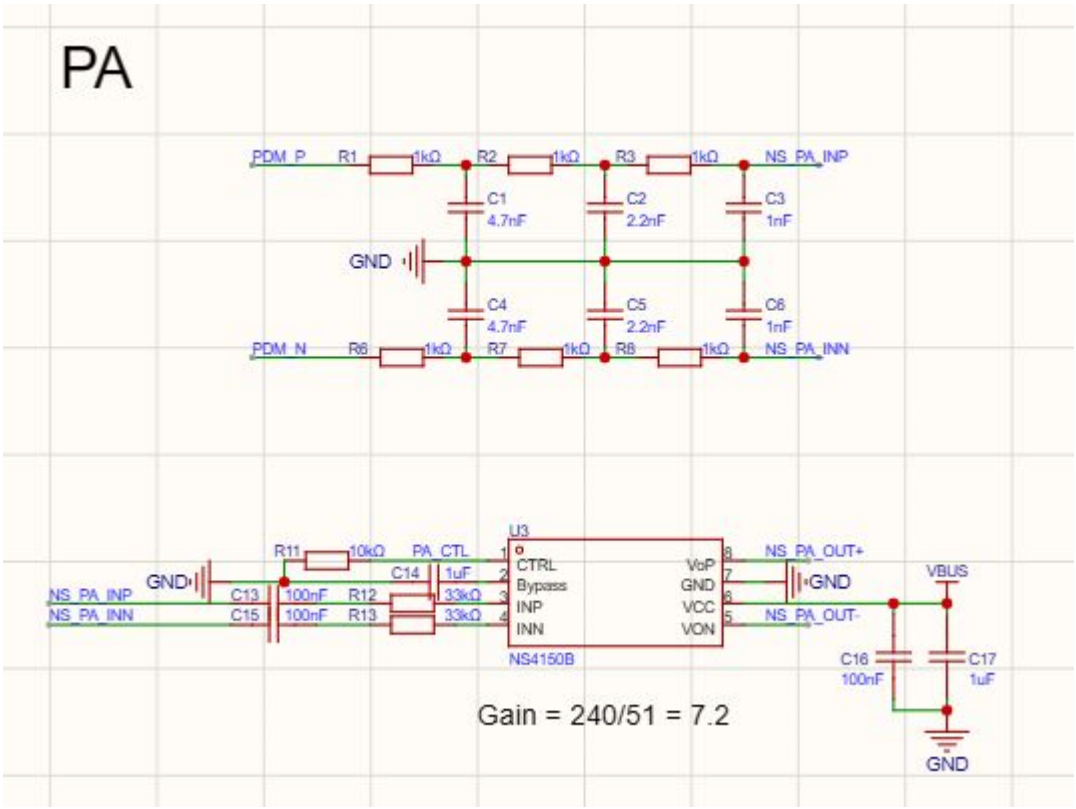
## USB



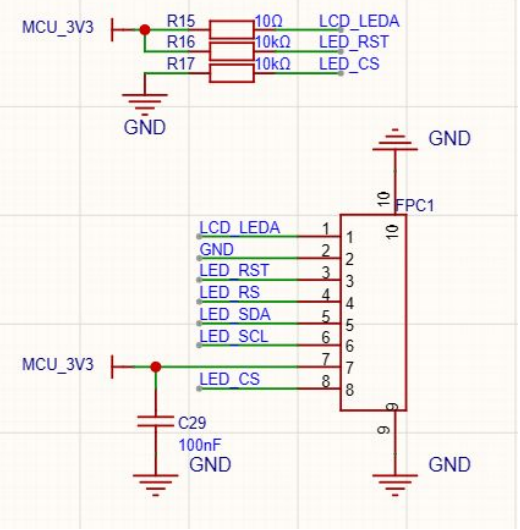

## LDO



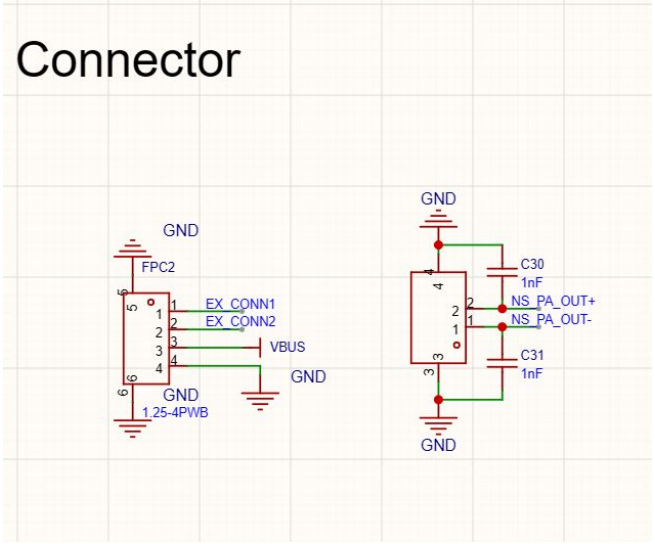

## PA



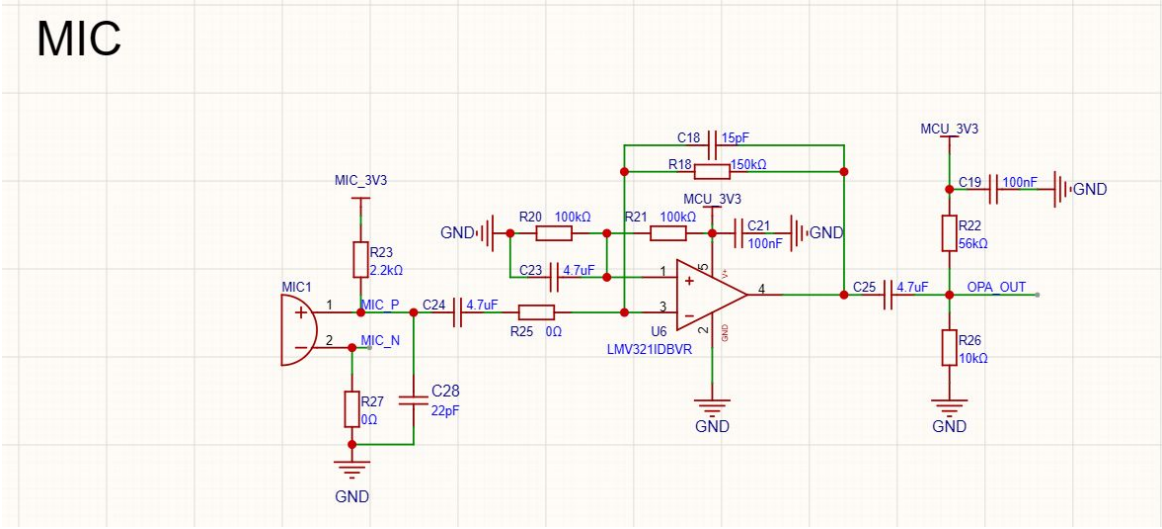

## LCD



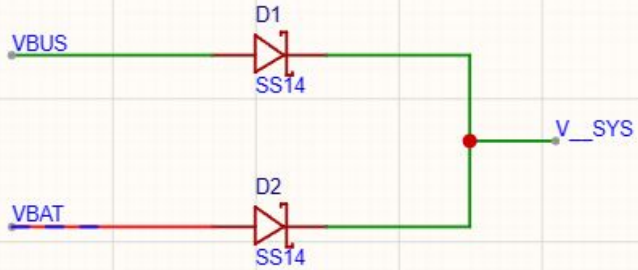

## Connector



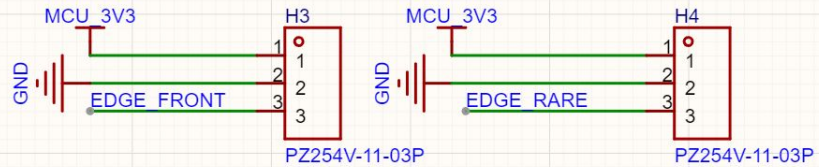

## MIC



# V_SYS




# IR




# Motor Driver Circuit: