Technical Documentation: Dual-Microcontroller Autonomous Sensor Vehicle

Project Name: Automated Guided Furnace Trolley / Sensor Car

Version: 1.0

Date: January 13, 2026

Repository: https://github.com/ArnawMahato/adruniosensor

1. System Overview

This document outlines the technical specifications, hardware architecture, and firmware logic for the Dual-Microcontroller Sensor Vehicle. The system utilizes a distributed processing architecture to ensure real-time responsiveness and efficient sensor data aggregation.

The system is divided into two primary processing nodes:

- **Sensor Hub (Bottom Layer):** An **Arduino Uno R4** is responsible for interfacing with environmental sensors, processing raw signals, and transmitting a consolidated telemetry packet.
- **Main Controller (Top Layer):** An **ESP32** serves as the central processing unit. It receives telemetry data, executes navigation algorithms, controls the traction motors, and manages wireless communication (Wi-Fi/HMI).

2. Hardware Architecture

2.1 Component List

- **Primary Controller:** ESP32 Development Board (3.3V Logic)
- **Secondary Controller:** Arduino Uno R4 (5V Logic)
- **Sensors:**
  - 2x Ultrasonic Distance Sensors (HC-SR04)
  - 1x Laser Distance Sensor (ToF/Analog)
  - 1x IR Speed Sensor (Encoder)
  - 1x IMU (Inertial Measurement Unit)
- **Actuation:** Dual DC Motors via Motor Driver Module (L298N/TB6612)

- **Power System:** 12V DC Battery Source with 5V Voltage Regulator Module.

2.2 Electrical Connections

The two microcontrollers communicate via a UART Serial Bridge. Due to voltage logic discrepancies (5V vs 3.3V), a voltage divider is implemented on the transmission line from the Arduino R4 to the ESP32.

Wiring Interconnect Table:

Signal Description	**Source (Arduino R4)**	**Destination (ESP32)**	**Notes**
Common Ground	GND	GND	**Critical:** Reference ground for signals.
Data Transmission	Pin 1 (TX)	Pin 16 (RX2)	**Via Voltage Divider (1kΩ/2kΩ)**
Data Reception	Pin 0 (RX)	Pin 17 (TX2)	Direct Connection (Safe).

Voltage Divider Schematic:

- **R1 (1kΩ):** Series connection between R4 TX and ESP32 RX.
- **R2 (2kΩ):** Parallel connection from ESP32 RX to GND.

3. Communication Protocol

The system utilizes a custom simplex UART protocol operating at **115200 baud**.

- **Packet Frequency:** 20 Hz (Every 50ms)
- **Data Format:** ASCII CSV (Comma Separated Values)
- **Terminator:** Newline character (\n)

Packet Structure:

[FrontDistance],[RearDistance],[LaserDistance],[Speed],[Angle]

Data Dictionary:

1. **FrontDistance:** Integer (cm) - Obstacle distance forward.
2. **RearDistance:** Integer (cm) - Obstacle distance rear.
3. **LaserDistance:** Integer (cm) - Precision distance measurement.
4. **Speed:** Float (m/s or RPM) - Current velocity.
5. **Angle:** Float (Degrees) - Heading or tilt from IMU.

4. Firmware Logic Description

4.1 Sensor Hub Logic (Arduino R4)

The firmware operates on a non-blocking loop using millis() timers.

1. **Initialization:** Sets up sensor I/O pins and initiates UART1 serial communication.
2. **Data Acquisition:**
   - Triggers ultrasonic pulses and calculates time-of-flight.
   - Reads analog voltage from the laser sensor.
   - Reads IMU data via I2C.
   - Calculates speed based on interrupt pulses from the IR sensor over a fixed time window.
3. **Transmission:** Formats all readings into the standard CSV packet and transmits via Hardware Serial.

4.2 Main Controller Logic (ESP32)

The firmware functions as a state machine driven by incoming serial data.

1. **Data Parsing:** Buffer reads incoming serial stream until a newline is detected. The string is split by delimiters to extract sensor values.
2. **Safety Layer:**
   - *Condition:* If FrontDistance < SafetyThreshold (20cm).
   - *Action:* Override motor PWM to 0 (Emergency Stop).
3. **Navigation Layer:**

    - If safety checks pass, the PID or open-loop controller adjusts motor PWM based on target speed or position.
4. **Actuation:** Writes PWM signals to the Motor Driver pins.

5. Source Code & Resources

The complete source code for both the Sensor Hub (Arduino R4) and the Main Controller (ESP32) is version-controlled and available online.

You can look into the code by this GitHub repo:

https://github.com/ArnawMahato/adruniosensor

This repository contains:

- SensorHub_R4.ino: Firmware for the Arduino R4 (Sensor data collection).
- MainController_ESP32.ino: Firmware for the ESP32 (Motor control and Logic).
