Embedded Vehicle Black Box for Telematics and Parameter Logging

ThingsBoard

presented by-

Kolamala Srivikas
Rahul Avhad
Abhishek Kalyankar
Dipendra Bodke

7th Aug, 2025

Introduction

What is a Vehicle Black Box?

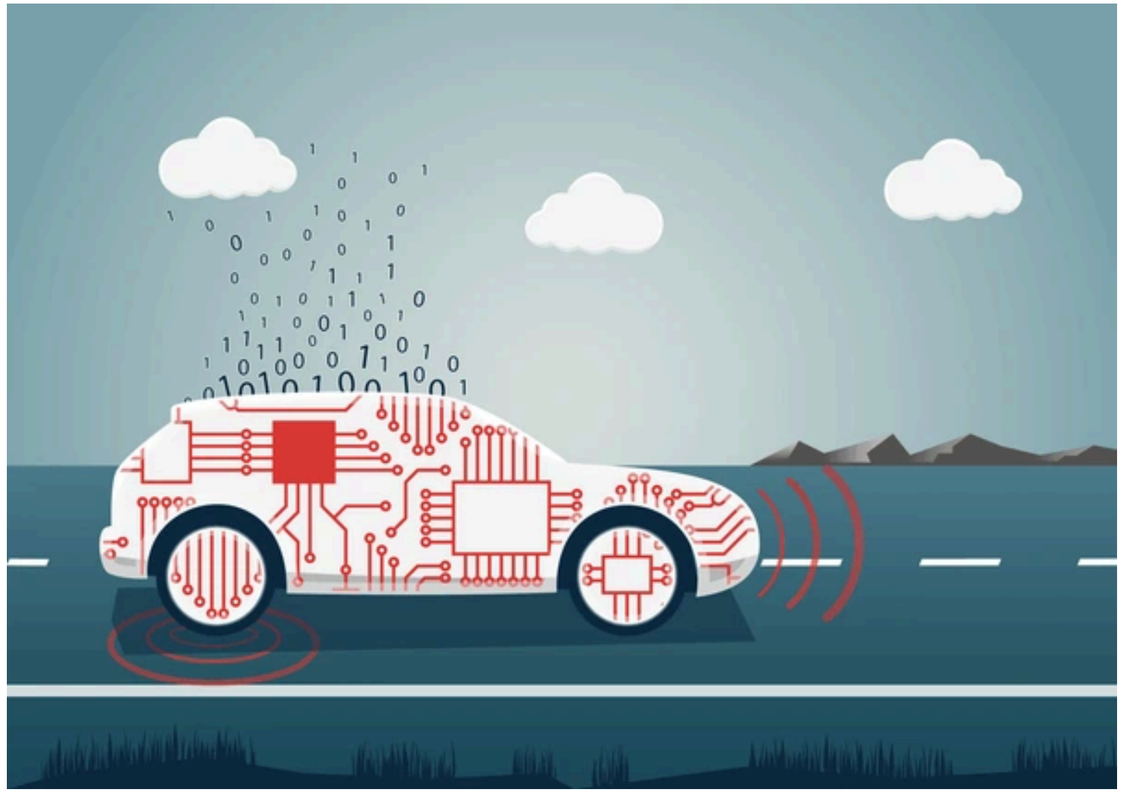
Inspired by aviation's Flight Data Recorders (FDRs) and Cockpit Voice Recorders (CVRs).
Records critical vehicle data before, during, and after an event.

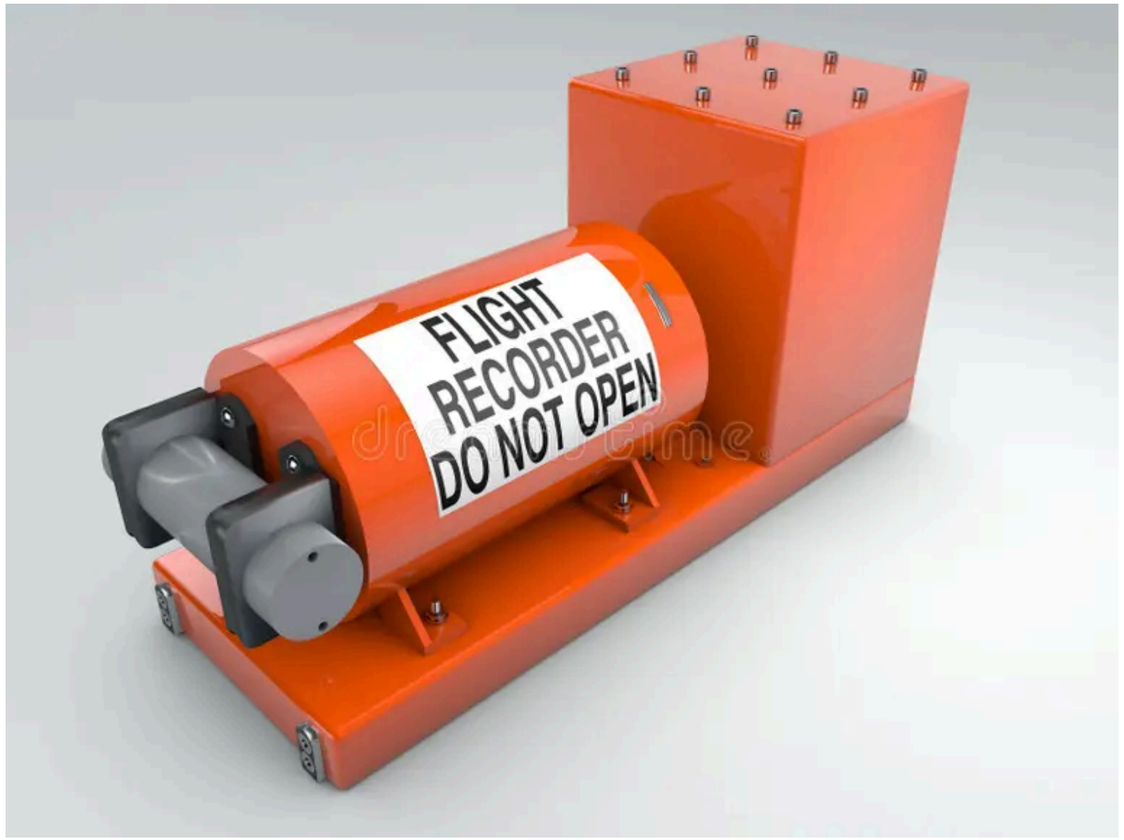
Why is it Important?

Enhanced Road Safety: Provides data for accident reconstruction and prevention.
Fleet Management: Optimizes operations, monitors driver behavior, and aids diagnostics.
Insurance & Legal: Offers objective data for claims and disputes.

Our Project's Goal:

To design and implement an embedded black box system.
Remotely transmit critical vehicle parameters using LoRa for telematics and accident analysis.
Collects, logs, and transmits data like speed, location, acceleration, braking, and crash information.




FLIGHT
RECORDER
DO NOT OPEN

Brief History – From Aviation to Automotive

Aviation Industry (Origins):
1950s: First flight recorders developed after major air disasters.
Purpose: To understand causes of accidents, improve aircraft design, and enhance safety protocols.
Components: Flight Data Recorder (FDR) for flight parameters, Cockpit Voice Recorder (CVR) for audio.

Evolution to Automotive Industry:
Early 2000s: Event Data Recorders (EDRs) began appearing in vehicles, primarily for crash data.
Today's Telematics: Driven by IoT, connectivity, and data analytics.

Beyond Accidents: Modern systems offer real-time monitoring, predictive maintenance, driver scoring, and fleet optimization.

Key Enablers: Miniaturization of electronics, widespread wireless connectivity (Wi-Fi, Cellular, LoRa), and powerful cloud computing platforms.

System Architecture

A distributed, modular system designed for robust data acquisition, communication, and cloud integration.

Vehicle Unit (Node 2 - Main Black Box):

Sensor Layer: Arduino Uno interfaces with basic sensors (DHT11, IR, Sound, Buzzer).

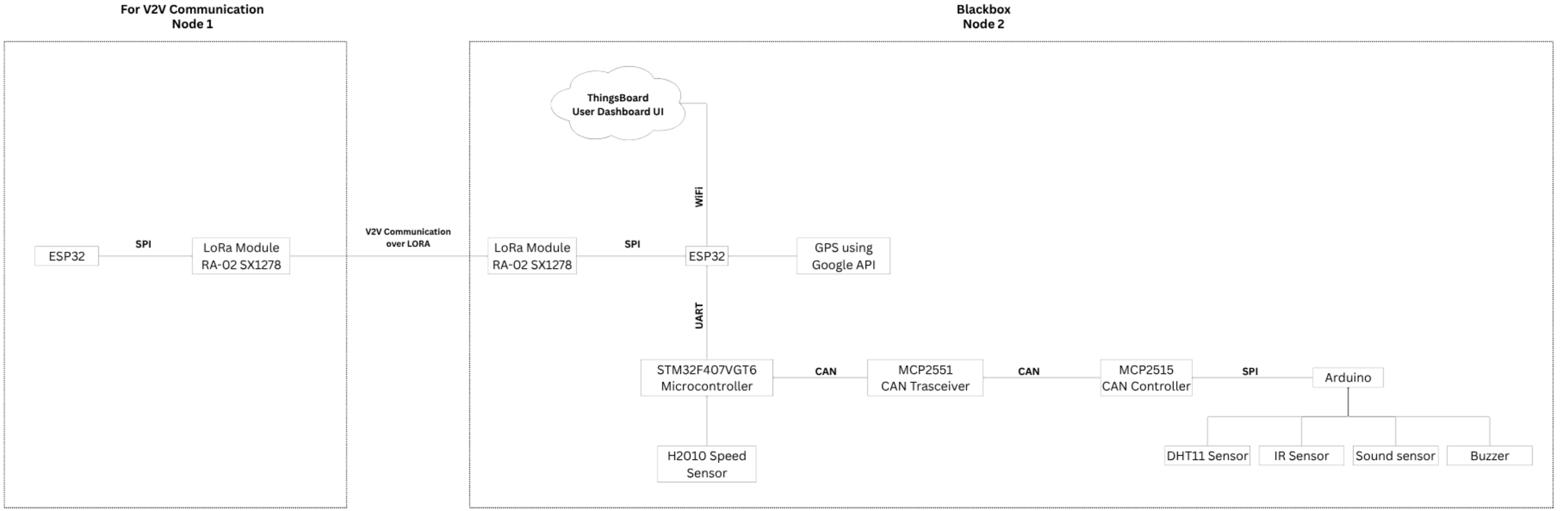
Data Aggregation & CAN Interface: STM32 collects data from Arduino Uno (UART) and CAN Bus (MCP2515/MCP2551), then sends to ESP32.

Communication Gateway: ESP32 handles Wi-Fi (ThingsBoard Cloud, Google Geolocation API) and LoRa (Node 1).

Proximity Monitor (Node 1 - LoRa Receiver):
Dedicated ESP32 with LoRa module for receiving critical alerts from Node 2 and sending proximity messages.

Cloud Platform (ThingsBoard Cloud):
Receives telemetry, stores data, processes rules, and manages alarms.
User Dashboard: Web interface for real-time visualization and historical analysis.

Block Diagram



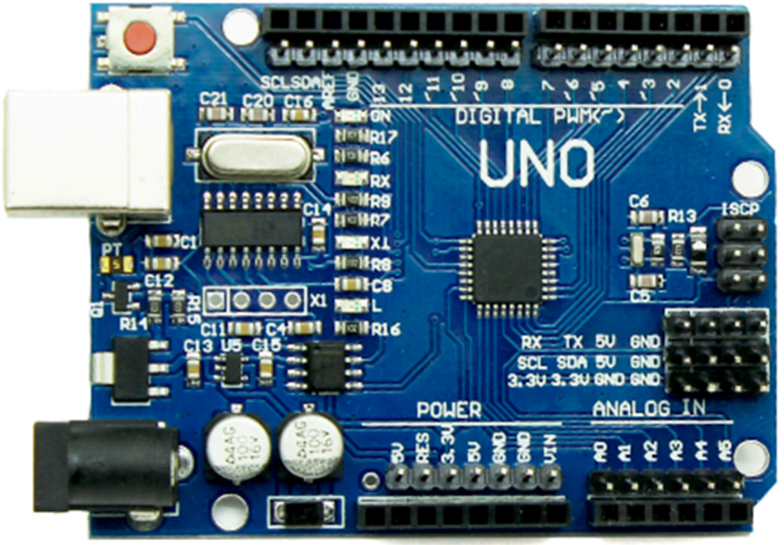
Block Diagram

Hardware Components

Microcontrollers:
ESP32 DevKit (x2): High-performance, Wi-Fi/Bluetooth enabled MCU for cloud communication and LoRa.

STM32F407VGT6 Microcontroller (x1): Powerful ARM Cortex-M4 MCU for CAN bus interface and data aggregation.

Arduino Uno R3 (x1): Easy-to-use microcontroller for basic sensor interfacing.



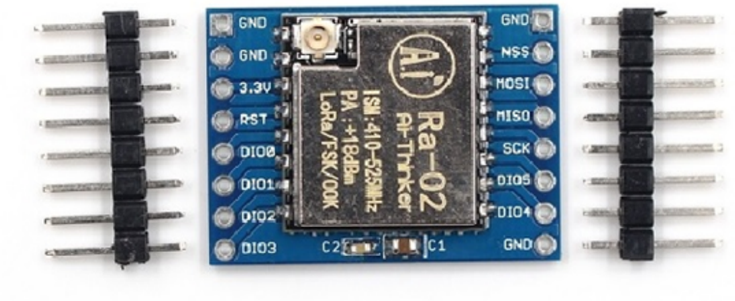
Communication Modules:

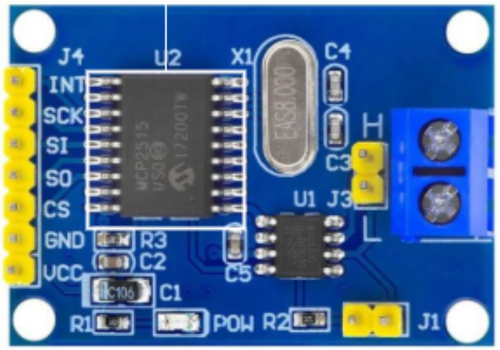
LoRa SX1278 Module (x2): Long-range, low-power wireless communication.

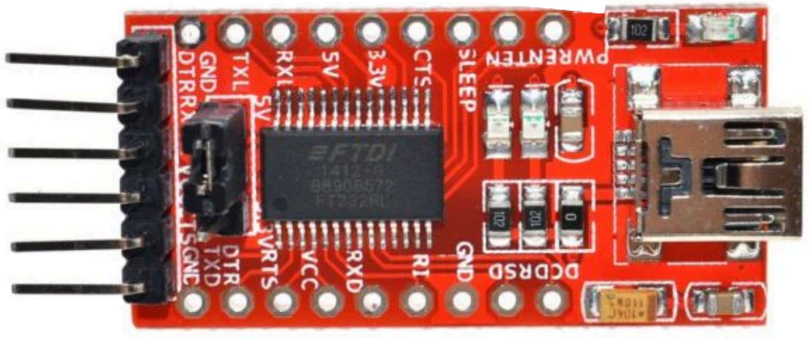
MCP2515 CAN Controller (x1): SPI-based CAN protocol controller.

MCP2551 CAN Transceiver (x1): Physical layer interface for CAN bus.

FTDI232 USB-Serial Converter (x1): For STM32 debugging via UART.



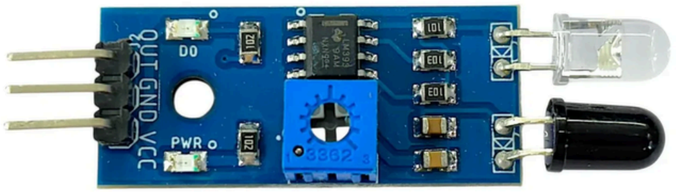




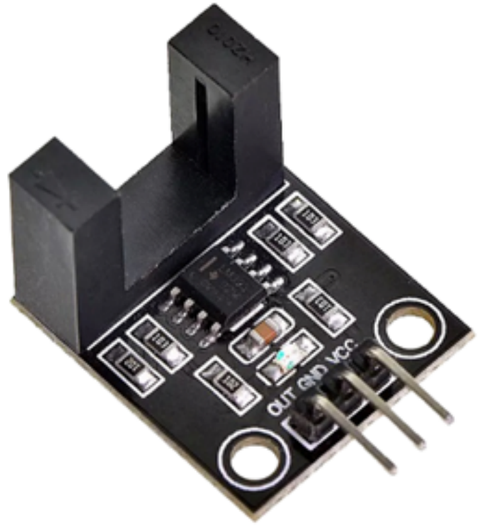
Sensors:

DHT11: Temperature & Humidity sensor

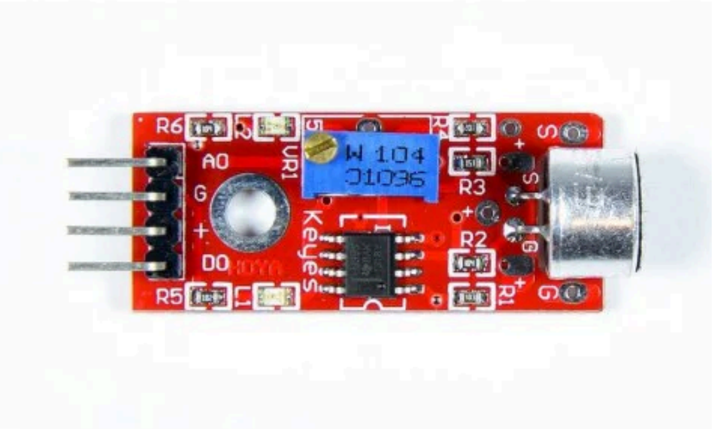
IR Sensor: Obstacle detection



H2010 Speed Sensor: Vehicle speed measurement

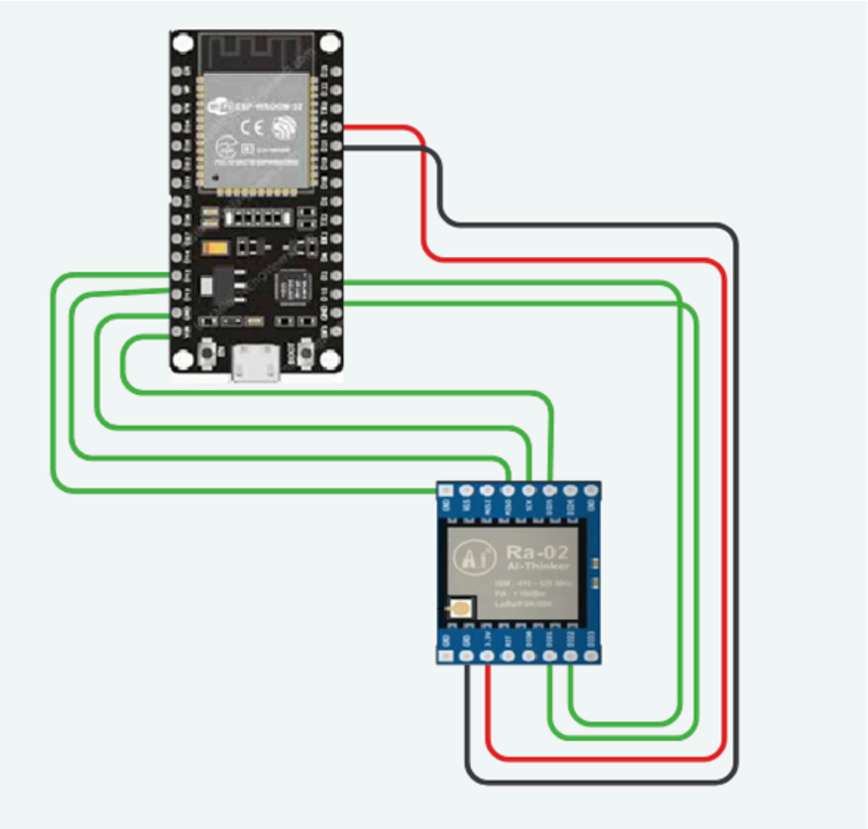


LM393 Sound Sensor: Crash detection (loud noise)

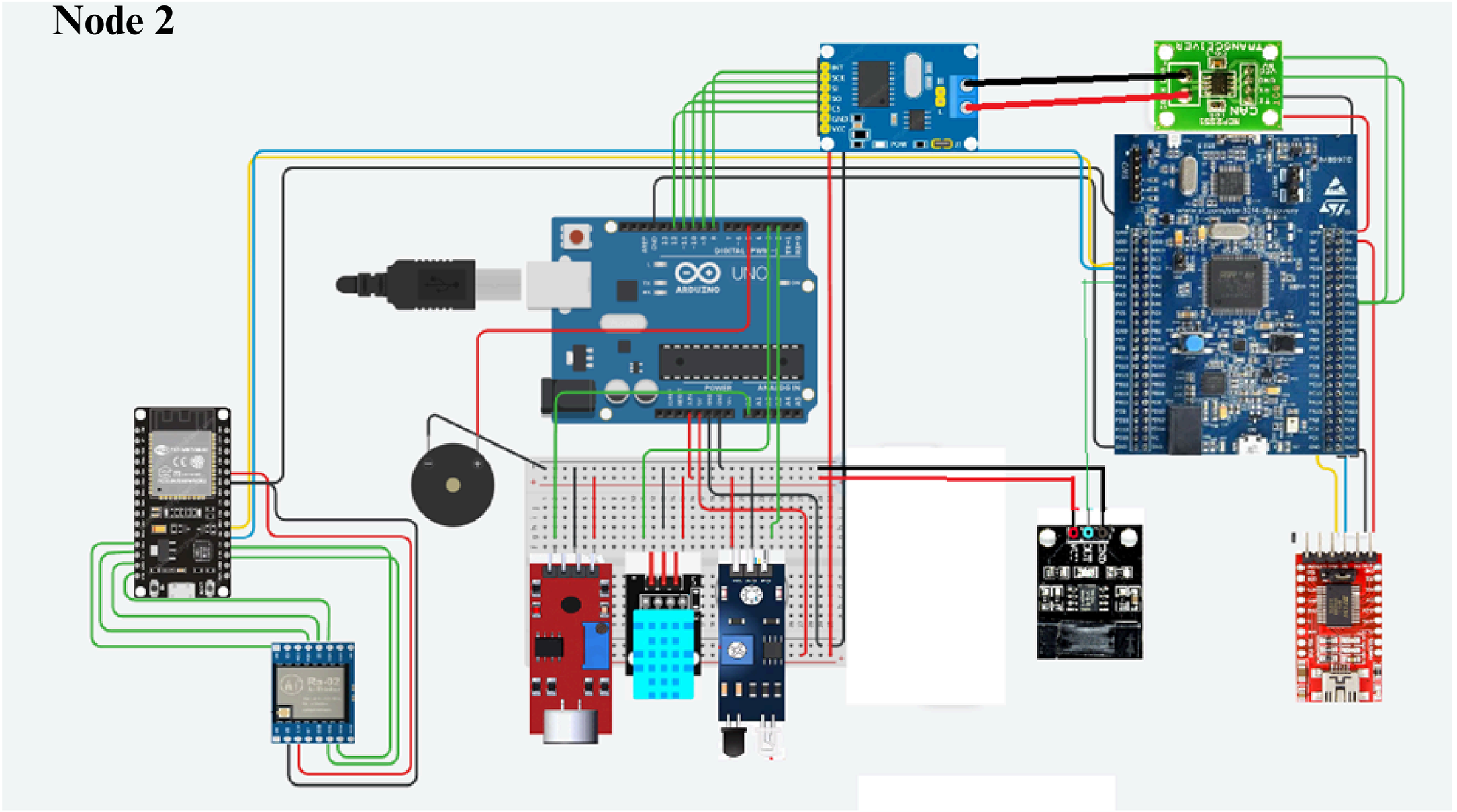


Buzzer

Node 1



Node 2



Software Design: Code Structure

Arduino Uno Firmware (Sensor Interface Layer):

Role: Reads raw data from DHT11, IR, Sound sensors. Controls the Active Buzzer**.**

Function: Simplifies sensor integration.

Output: Transmits processed sensor readings to the STM32 via UART.

STM32 Firmware (Data Aggregation & CAN Layer):

Role: Receives sensor data from Arduino Uno (UART). Interfaces directly with the vehicle's CAN bus (MCP2515/MCP2551) to read vehicle parameters (speed, RPM, acceleration).

Function: Aggregates all data, performs preliminary calculations (e.g., idling time), and formats a consolidated data string.

Output: Sends the formatted data string to the ESP32 via UART. Provides debug output via FTDI.

ESP32 Firmware (Communication & Cloud Gateway Layer):

Role: The main application. Receives data from STM32. Handles Wi-Fi, MQTT (ThingsBoard Cloud), HTTP (Google Geolocation API), and LoRa communication.

Key Functions:
setup(): Initializes Wi-Fi, LoRa, MQTT.
loop(): Main loop for continuous operation, calling MQTT client loop, checking LoRa, reading STM32 serial, publishing telemetry, updating geolocation.
parseStm32Data(): Extracts values from STM32 string.
checkCriticalStatus(): Determines vehicleStatus (SAFE/CRITICAL) and triggers LoRa alerts.
updateGeolocation(): Fetches location from Google API.
publishTelemetry(): Constructs JSON payload and sends to ThingsBoard Cloud.
onReceive(): LoRa interrupt handler.
Output: JSON telemetry to ThingsBoard Cloud, LoRa alerts to Node 1, LoRa proximity messages to Node 2.

Project Flow

Sensor Data Collection:
DHT11, IR, Sound, Buzzer data read by Arduino Uno.
H2010 Speed Sensor data and CAN Bus data read by STM32.

Data Aggregation & Processing:
Arduino Uno sends sensor data to STM32 (UART).
STM32 combines Arduino data with CAN bus data, processes it, and sends a consolidated string to ESP32 (Node 2) (UART).

Cloud Telematics (Node 2):
ESP32 (Node 2) fetches geolocation (Lat, Long, Acc) from Google Geolocation API (via Wi-Fi).
ESP32 (Node 2) constructs a JSON payload (all sensor data, vehicle status, geolocation) and sends it to AWS Cloud (MQTT/HTTP over Wi-Fi).

Real-time Alerting (LoRa V2V):
If ESP32 (Node 2) detects critical conditions (high temp, obstacle, crash), it sends a critical alert via LoRa to ESP32 (Node 1).
ESP32 (Node 1) receives critical alerts and sends back proximity messages via LoRa.

Cloud Visualization & Alarms:
ThingsBoard Cloud stores telemetry, processes data through rules, and triggers alarms (e.g., VehicleStatus = CRITICAL).
A User Dashboard displays real-time data, historical trends, map location, and an alarms table.

Key Achievements

Multi-Microcontroller Integration: Seamless data flow between Arduino Uno, STM32, and ESP32.

CAN Bus Data Acquisition: Successfully interfaced with vehicle's CAN bus for native parameter logging.

Bi-directional LoRa Communication: Implemented robust long-range alerts and proximity messages.

Real-time Geolocation: Integrated Google Geolocation API for continuous location tracking.

Comprehensive Telemetry: Collected and transmitted diverse sensor data (Temp, Hum, IR, Sound, Speed, Accel, RPM, Idle).

Dynamic Vehicle Status: Developed logic to classify and report vehicle status (SAFE/CRITICAL).

Cloud Telematics: Established reliable data streaming to ThingsBoard Cloud for storage and analysis.

Interactive Dashboard: Created a custom dashboard with real-time visualization and automated alarms.

Robustness: Overcame challenges like hardware connection issues, MQTT authentication, and payload sizing.

Applications

Accident Reconstruction & Analysis:
Provides objective data (speed, acceleration, braking, crash indicators, location) for post-incident investigations.
Aids insurance claims and legal proceedings.

Emergency Response:
Automatic critical alerts (crash, high temperature) with location data for rapid deployment of emergency services.

Vehicle Security:
Monitoring unauthorized vehicle movement or unusual activity.

Conclusion

The Embedded Vehicle Black Box project successfully developed a comprehensive and functional system for vehicle telematics and parameter logging.
It effectively integrates diverse hardware platforms and communication protocols to deliver real-time data and critical alerts.
The modular design and cloud integration provide a scalable and robust solution for modern vehicle monitoring needs.
This project serves as a strong foundation for future advancements in automotive IoT, paving the way for safer roads and more efficient vehicle operations.


Thank
you!