OBD-II BASED EV MONITORING AND EMULATION SYSTEM

CONTROL SYSTEMS AND INSTRUMENTATION ENGINEERING PROGRAM

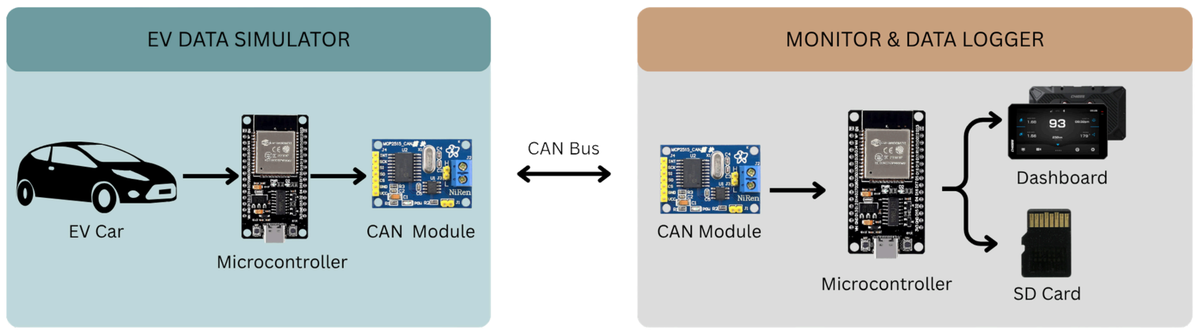
AUTHORS : THANATORN SARIKAKET, BANWITIT RUNGRUENG
ADVISOR : WUTTHICHAI POLWISATE

ABSTRACT

This project, the “OBD-II Based EV Monitoring and Enulation System,” addresses a fundamental limitation of modern electric vehicles, where closed-architecture CAN networks severely restrict access to critical kinematic parameters. Without this system, vital data such as precise steering angles, throttle and braking responses, and vehicle speeds—which are the absolute core for advancing Autonomous Vehicles research—would remain entirely inaccessible. To overcome this barrier, a decentralized system architecture was developed to ensure maximum stability. The system integrates a Dashboard and Datalogger module capable of extracting, decoding, visualizing, and recording high-resolution real-time data via OBD-II, alongside an OBD-II Simulator designed to emulate CAN ID traffic. This simulator facilitates safe and repeatable Hardware-in-the-loop testing without the risks associated with relying on actual vehicles. The project achieved complete functional success, which was rigorously validated through the Reverse engineering of commercially available EV platforms. By effectively unlocking these hidden data layers, the system establishes a powerful, real-time data infrastructure that serves as a crucial foundation for next-generation automotive innovation and intelligent mobility systems.

SYSTEM OVERVIEW

This block diagram illustrates an EV data simulation and logging system consisting of two main sections connected via a CAN Bus. The "EV DATA SIMULATOR" utilizes a microcontroller to generate vehicle telemetry, transmitting it through a CAN Module. The data travels across the CAN Bus to the "MONITOR & DATA LOGGER," where a receiving microcontroller processes it to simultaneously display real-time status on a Dashboard and record the data onto an SD Card for future analysis.

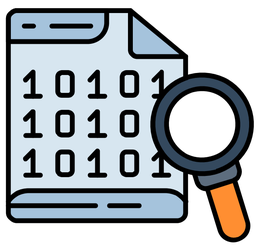


METHODOLOGY

The project implements a Decentralized Embedded System architecture, centered around the high-performance ESP32 microcontroller. To ensure ultra-low latency and maximum data security, the system utilizes Local Processing, eliminating dependencies on IoT or Cloud infrastructures. The methodology is divided into four critical engineering layers:

1. Hardware Architecture & Communication Protocols

- CAN Bus Interface: Integration of the MCP2515 CAN Controller interfaced with the ESP32 via SPI protocol. This setup drives the CAN Transceiver to establish a robust link with the vehicle's OBD-II port.
- Signal Integrity: The custom PCB design features Differential Pair routing and is equipped with 120Ω termination resistors to maintain impedance control and prevent signal reflection.
- HMI Integration: Real-time data visualization is achieved through a 7-inch DWIN HMI Touchscreen via UART communication at a baud rate of 115,200 bps.



2. Reverse Engineering & Data Decoding

Due to the proprietary and closed nature of EV data architectures, the system employs advanced bit-level interpretation techniques:

- BYD Dynamics Analysis: Deciphering steering angle parameters (CAN ID: 0x0002) utilizing Two’s Complement arithmetic and Parity Check for data validation.
- MG ZS EV Battery Monitoring: Extracting High-Voltage (HV) system parameters through Passive Observer Mode to circumvent the vehicle’s security safeguards .
- Standard PID Translation: Applying algebraic transformation algorithms to convert raw response packets into physical units, including Speed (0x0D), RPM (0x0C), and Throttle Position (0x11).

3. Real-Time OS & High-Speed Data Logging

- Buffer Overflow Prevention: Implementation of FreeRTOS (Real-Time Operating System) to manage multi-tasking, message queuing, and dynamic memory allocation, ensuring stability during high-density CAN traffic.
- High-Speed Datalogging: Utilizing RAM Buffering (Batch Writing) techniques to record data onto FAT32-formatted SD cards, effectively minimizing latency caused by frequent file I/O operations.

4. EV Data Simulator (OBD-II Emulation)

- ECU Emulation: Operates as a synthetic signal source that reverse-maps engineering physical values back into raw byte sequences for CAN Bus injection.
- Hardware-in-the-Loop (HIL): Provides a safe, controlled, and repeatable environment for testing Dashboard and Datalogger stability without the need for a physical electric vehicle.

RESULTS & DISCUSSION

CAN Bus Decoding and Parameter Extraction

From real-world testing on an electric vehicle platform, the proposed system successfully interfaced with the in-vehicle communication network via the CAN Bus protocol operating at a baud rate of 500 kbps. The system demonstrated the capability to accurately identify, decode, and interpret multiple critical parameters. The decoded signals are categorized as follows:

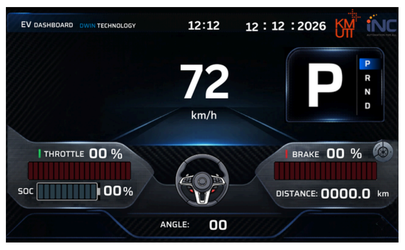
CAN Bus Decoding and Parameter Mapping Table

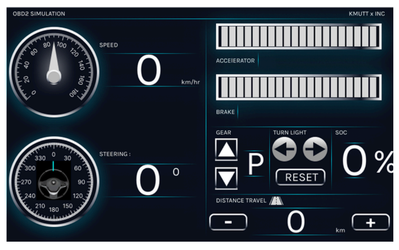
Category	Signal Name	CAN ID (Hex)	Data Length	Description	Unit	Remarks
Vehicle Kinematics	Vehicle Speed	0x0292	2 Bytes	Extracted from raw CAN data and converted to physical speed	km/h	High accuracy, matches dashboard
	Motor Speed (RPM)	0x01DA	2 Bytes	Measures motor rotational speed with fast response	RPM	Faster than dashboard update
	Steering Angle	0x0002	2 Bytes	Decoded using signed integer (Two’s Complement)	Degree	Indicates left/right direction
Energy & Drive	Battery SoC	0x055B	1 Byte	Real-time battery charge level monitoring	%	Stable and continuous
	Gear Position	0x0421	1 Byte	Identifies gear state (P, R, N, D)	-	No latency observed
Driver Inputs	Accelerator Position	0x011A	1 Byte	Represents throttle input level	%	Used for driving behavior analysis
	Brake Status	0x01CA	1 Byte	Detects braking action	Boolean	ON/OFF state
	Turn Signals	0x060D	1 Byte	Indicates left/right turn signal status	Boolean	Event-based logging

2. System Stability and Data Logging Performance

The system exhibited high reliability and deterministic performance during continuous operation:

- Zero Packet Loss:
- By integrating the MCP2515 CAN controller with the ESP32 microcontroller, along with an optimized data queue management mechanism, the system maintained zero packet loss even under high CAN Bus load conditions.
- Deterministic Data Logging:
- The Data Logger module consistently recorded data to the SD card at a stable sampling rate defined by the LOG_INTERVAL parameter (approximately 20–25 Hz). This ensured temporal consistency and suitability for downstream data analytics and visualization.
- Human-Machine Interface (HMI) Responsiveness:
- Real-time data transmission to the DWIN HMI display via UART communication (115200 bps) using the Variable Pointer (VP Address) protocol resulted in smooth and responsive graphical updates. The speedometer and numerical indicators showed minimal latency and high visual fidelity.





3 Simulator and Hardware-in-the-Loop (HIL) Validation

To verify system robustness and readiness for real-world deployment, simulation and validation were conducted:

- ECU Signal Emulation:
- The developed simulator successfully generated CAN messages (e.g., IDs 0x0292 and 0x055B) that closely replicated real vehicle behavior, enabling controlled testing conditions.
- Hardware-in-the-Loop (HIL) Testing:
- The dashboard system accurately received and processed simulated CAN data. The displayed outputs were consistent with the input signals, confirming system correctness and integration reliability.

These results demonstrate that the proposed system is fully capable of operating both in simulated environments and real vehicle conditions, ensuring safe and effective deployment.