Cross-Platform UDS Flasher & Diagnostics Suite for Agricultural ECUs

Overview

Customer: Belgium-based Agricultural OEM (Name withheld for confidentiality)

Industry: Agriculture | Embedded Systems | Automotive Electronics

Modern agricultural machinery depends on Electronic Control Units (ECUs) to manage engine control, hydraulics, and sensor-based automation.

The client required a robust UDS (Unified Diagnostic Services) flashing and diagnostics tool tailored for in-house Bosch and TTC-based controllers.

We developed a cross-platform, production-grade UDS Flasher suite with:

• GUI-based Windows application (Avalonia UI)

• CLI-based Linux tool

The solution enables full ECU lifecycle support including flashing, diagnostics, and calibration using PEAK and Vector CAN interfaces.

Problem Statement

• Lack of a unified tool for ECU flashing, diagnostics, and calibration.

• Existing tools limited to Windows; lack of Linux CLI support for field teams.

• Heavy dependency on third-party tools such as Bosch Service Tool.

• Complex ECU ecosystem involving Bosch and TTC controllers requiring custom handling.

• Need for full UDS protocol support over ISO-TP including multi-frame communication and grouped DID parsing.

• Challenges in decoding and data handling (padding/stripping bytes like 0x55, 0x00).

• Security requirements such as XTEA-based SecurityAccess for TTC ECUs.

• Field usability needs including logging, monitoring, uptime tracking, and real-time feedback.

Our Approach

• Designed a cross-platform solution with Windows GUI (Avalonia) and Linux CLI.

• Built a centralized UdsSession engine to handle reusable UDS protocol logic.

• Implemented modular architecture with custom handlers for Bosch and TTC ECUs using Dependency Injection.

• Used Strategy Pattern for CAN interface abstraction supporting PEAK, Vector, and SocketCAN.

• Enabled multi-dongle compatibility including Vector CAN (VN1610, VN1611) and PEAK CAN (PCAN-USB).

• Developed advanced flashing mechanisms including multi-region memory support and raw binary streaming.

• Integrated diagnostics framework with Read/Write DIDs, session control, and routine control.

• Added real-time monitoring using SignalR (GUI) and console logs (CLI).

Implementation Details

Dual Platform Support

• Windows application with Avalonia GUI

• Linux Command-Line Interface

Multi-Dongle Support

• Vector CAN (VN1610, VN1611)

• PEAK CAN (PCAN-USB)

Protocol Support

• Full UDS protocol over ISO-TP

• Custom implementations for Bosch and TTC ECUs

ECU Flashing

• Multi-region memory block flashing

• XTEA-based SecurityAccess for TTC ECUs

• Flash driver management

• Raw binary streaming

Diagnostics

• Read/Write DIDs (Device Identification Data)

• Session control, SecurityAccess, RoutineControl

• Multi-frame communication handling

• Grouped DID parsing for Bosch ECUs

• Data decoding with stripping/padding (0x55, 0x00)

Calibration & Logging

• Custom parameter calibration post-flash

• UDS logs, voltage monitoring

• Uptime and operating hours decoding

• Real-time feedback via SignalR and CLI console

Technology Stack

• .NET Core / ASP.NET Core

• Avalonia UI

• SocketCAN, PCAN-Basic, Vector XL Driver

• SignalR

• Visual Studio Installer Project

Architectural Highlights

• Strategy Pattern for CAN abstraction

• Centralized UdsSession Engine
• Modular ECU handlers using Dependency Injection
• Multi-threaded backend launcher tied to GUI lifecycle

Business Impact

• Enabled field teams to reprogram ECUs using Linux CLI tools
• Reduced dependency on third-party tools like Bosch Service Tool
• Reusable solution across multiple ECU variants
• Improved operational efficiency and flexibility