PROTEUS CAN BUS

Write the firmware for one ECU on a bus full of behavioural car-part models, with full traffic visibility and realistic fault injection.

Teaching Automotive CAN Bus

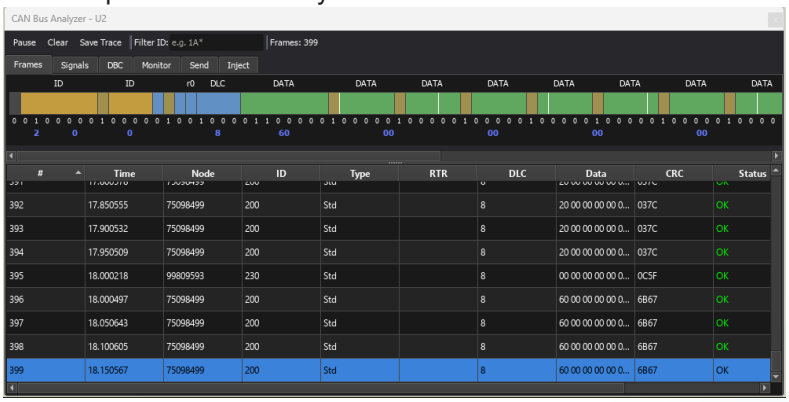
CAN bus is a cornerstone of every automotive and embedded systems curriculum, but teaching it well is hard. A meaningful exercise needs a multi-node network, and provisioning benches of ECUs, transceivers, wiring and analysers is expensive, fragile and time-consuming. Too often, lab hours are spent debugging connectors rather than firmware, and students never see a realistically busy bus at all.

Proteus VSM solves this at the system level, not just the protocol level. Students write real firmware (in C or MicroPython, on STM32 or pyboard targets) and run it against a complete simulated vehicle network inside the schematic. They single-step from a line of transmit code to the resulting frame on the wire, watch arbitration happen, and debug receive handlers with breakpoints. Students gain insight early: the brake pedal is not wired to the lamp; it sets a bit in a broadcast message, and another ECU lights the lamp.

The key to making this work is Proteus' twenty-plus behavioural car-part nodes. The Instrument Cluster, Engine ECU, Body Control Module, Vehicle Dynamics and Lighting Clusters each join the bus as a single component with a live popup window: gauges sweep, lamps illuminate, sliders inject stimulus. The student's microcontroller is wired exactly as it would be on a real board, and any Proteus VSM microcontroller can take part: the STM32F103 and pyboard use their on-chip CAN controller, while a PIC, AVR or Cortex-M0 uses an MCP2515 or MCP2518FD over SPI, each reaching the wire through a transceiver such as the MCP2551. Because the car-part nodes are behavioural models, not extra microcontrollers, the whole network can be simulated efficiently and advance responsively.

The CAN Bus Analyser is the student's inspection and analysis tool. It captures and timestamps every frame, decodes it by name from the loaded DBC, and opens any frame, classical CAN or CAN FD, down to its individual bits. It also transmits frames on demand, reports each node's error state, injects faults to force bus-off, and exports industry-standard Vector and PEAK trace files.

Deployment is trivial. Ready-made sample designs are included (one student MCU, a few behavioural nodes and the instruments), covering message reception, lighting control, gateways, fault recovery and security. Every student runs the identical laboratory at home: nothing to wire, nothing to break and no safety concerns.



"The whole Proteus suite has been invaluable to us in delivering our advanced BTEC Electronic Courses. Proteus VSM processor simulation is excellent and very easy to implement."

Alan Duffy
B-Met – Sutton Coldfield Campus

CAN BUS SIMULATION

Student firmware in C or MicroPython runs live on the simulated bus.

Behavioural car-part nodes deliver a busy vehicle network at single-MCU simulation speed.

TEACH THE PROTOCOL

Students learn CAN frame anatomy and arbitration by watching it happen. The Analyser exposes the identifier, DLC, data and acknowledgement of every message, and opens any frame down to its bits, with start, arbitration, CRC and stuff bits laid out in order.

Priority becomes concrete when two nodes transmit at once: arbitration plays out bit by bit and the lowest identifier wins. Standard and extended identifiers and bus loading are visible and measurable, and the same view breaks down diagnostic transport: a request to 0x7DF, replies on 0x7E8, and a multi-frame ISO-TP VIN reassembled from the log. The simulation is deterministic and pausable, so the lecturer can freeze the bus at any moment.

TEACH REAL-TIME MESSAGING

Real in-vehicle networks are exercises in scheduling, and the behavioural nodes provide authentic traffic. The Vehicle Dynamics node publishes wheel speeds at periodic rates, the Engine ECU streams powertrain data, and the student's firmware must transmit on time. Students implement a non-blocking timer loop, a clean indicator flash, brake-over-tail priority and input debounce, then build a receive-process-retransmit bridge that uses interrupt-driven reception. Diagnostic exchanges interleave with the periodic broadcasts, so students see how an ECU answers a scan tool without missing deadlines. They change a timing constant and verify it against the Analyser's timestamps.

TEACH SIGNAL DATABASES

Industry builds CAN applications on signal databases, and so does Proteus. Every node ships with a DBC file defining its messages and signals, so students author firmware against a genuine signal specification. Exercises move from decoding a single bit through scale-and-offset numeric signals (raw count to RPM, °C, km/h) to diagnostic PIDs via a scan tool. They predict a payload, then verify it in the Analyser's Signals view, which decodes values into engineering units. A reverse-engineering exercise withholds the DBC entirely: students recover an unknown node's signals and document them, the skill aftermarket engineers are paid for.

TEACH FAULT FINDING AND SECURITY

Deliberately corrupting a real bus is impractical in a student lab; in Proteus it is a menu click. Inject bit, stuff, form or CRC errors, or drive a victim to bus-off, and watch its error counters climb from error-active to error-passive to bus-off. Any node can also take a node-level fault, from mute or dead to flooding, each mapped to a trouble code, and students write firmware that returns it to a safe state. Faults also surface as diagnostics: a broken bulb or leaky tyre logs a code, a scan tool reads and clears. Security matters too: a rogue node can spoof, replay or flood messages, caught by a passive intrusion-detection node, while authenticated messaging and seed-and-key access show the defences, all sim-safe.

CAN Analyser decodes every frame by name and exports Vector and PEAK traces.

Inject bit, CRC and bus-off faults or rogue-node attacks, then diagnose them, all sim-safe.

Labcenter Electronics Ltd.
The Stables, Hartlington Hall Estate, Near Burnsall, N. Yorks, BD23 6BY, UK
Web: www.labcenter.com
Email: info@labcenter.com

- CAN Bus Intro
- CAN Analyser
- CAN Fault Finding
- CAN Bus Safety