CAN Bus for Software Developers

Controller Area Network (CAN) is a common messaging system in embedded systems. It's widely used in automotive applications for monitoring and controlling vehicle systems. This overview focuses on the software perspective of CAN bus.

by Dave

CAN Bus Basics

1

Message-Centric

CAN bus focuses on messages, not nodes. It's a cloud where messages appear and are acted upon.

2

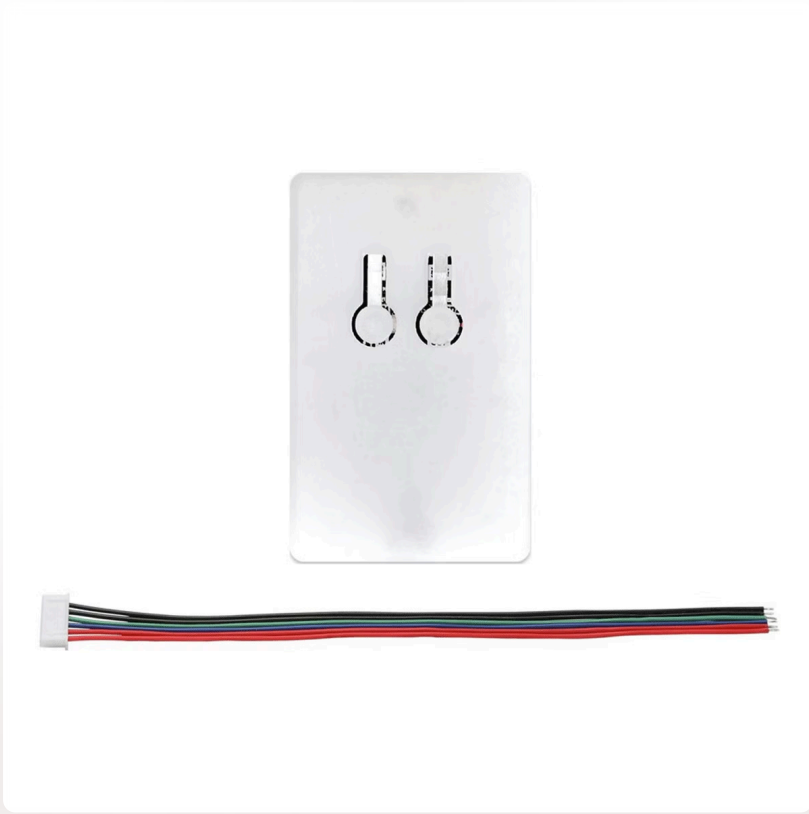
Pub/Sub Architecture

Nodes can publish messages and subscribe to receive them. It's not master/slave or client/server.

3

Short Messages

Message format fits within 8 bytes. IDs are unique and determine priority.



Message ID and Priority

1

ID Assignment

Message IDs are unique across sending nodes. Lower ID value means higher priority.

2

Arbitration

Hardware handles bus arbitration using message IDs. Lower IDs get priority.

3

Transmission

Messages flow according to ID priority. Higher ID messages wait for transmission.

CAN Controllers

Functionality

Modern CAN controllers offer advanced features like error handling and message filtering.

Complexity

Controller datasheets can be extensive. The TI CAN controller 4550 datasheet is 150 pages.

Setup

Basic CAN bus systems can be easily set up in Linux with proper kernel configuration.

Higher Level Protocols

Ad-Hoc Systems

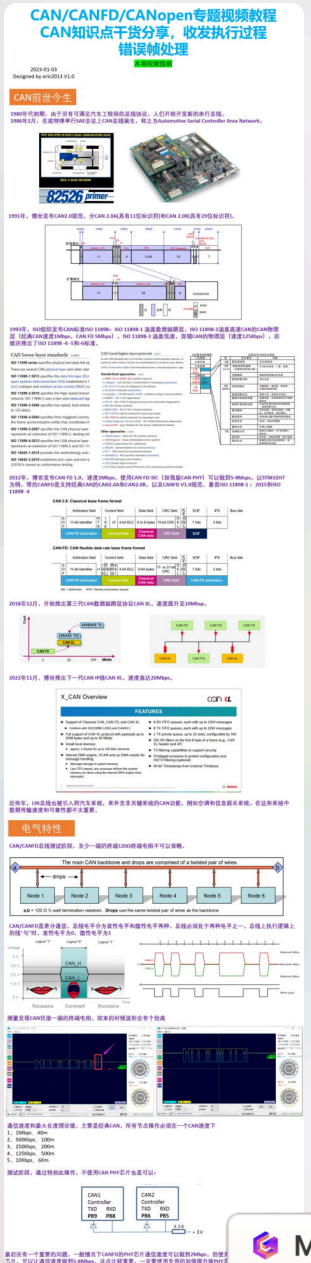
Simplest systems assign message IDs and formats to components.

CANopen

Used in automation, attempts to turn CAN bus into a node-based system.

Automotive Standards

Includes SAE J1939 and ISO 11898 series specifications.



Software Architecture

1

Message Graph

Design CAN systems using a graph of messages and state machines.

2

Node Independence

Architecture can be done without specifying physical nodes or CPUs.

3

Message Allocation

Use spreadsheets to allocate outgoing messages to nodes in the system.

Simple Can Bus App

Virtual CAN Bus

Setting up a virtual CAN bus for Linux.

Libraries

Simple libraries supporting CAN bus access in C and Go.