Controller Area Network (CAN)

- CAN is a real-time, serial-communication, multi-master vehicle network
  - CAN specification defines the Data Link Layer,
- Developed by Robert Bosch, Gmbh (Germany) in 1986 to provide high-speed, robust communications in automotive applications (between three ECUs (electronic control units) in vehicles by Mercedes.
  - UART was limited to point-to-point communication.
  - They needed need a multi-master communication system.
  - First CAN silicon fabricated in 1987 by Intel
- 1st release CAN 1.2, current standard CAN 2.0 (1991)
- Automotive applications:
  - low speed CANbus to operate window and seat controls.
  - high speed CANbus for engine management or brake control.
  - other applications: engine sensors, anti-skid systems, etc.

CAN and related standards

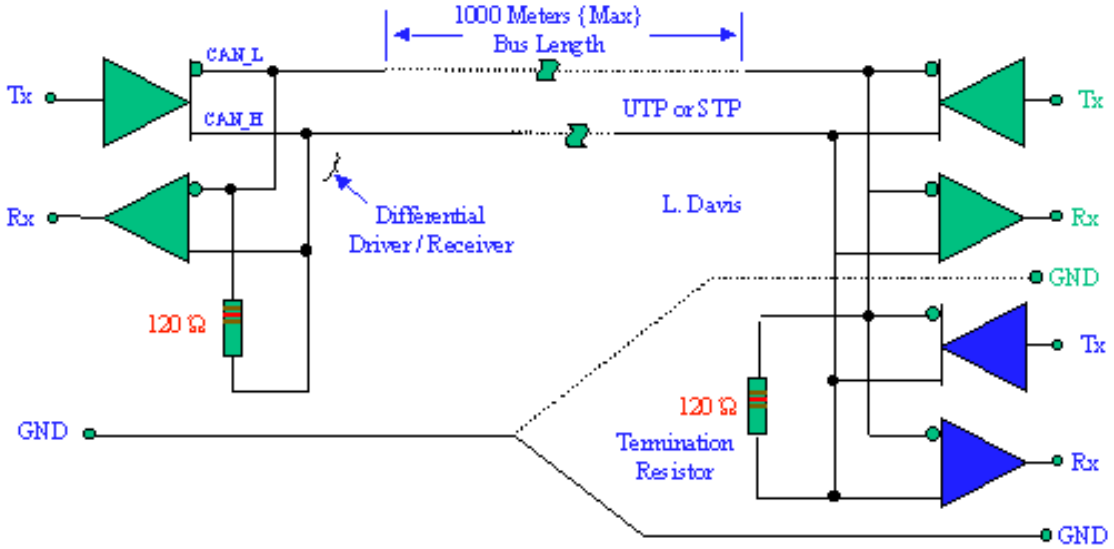
- **CAN Bus Specification** Version 2.0 (data link layer)
- **ISO/DIS 11898 (physical layer)**
  - **Low speed Class A bus (<10 kbps)**
  - **Class B bus (10 kbps – 125 kbps)**
  - **High speed bus (125 kbps – 1 Mbps)**
- **ISO/DIS 11898-1:** Road vehicles -- Controller area network (CAN) -- Part 1: Data link layer and physical signaling
  ISO/DIS 11898-2: Road vehicles -- Controller area network (CAN) -- Part 2: High-speed medium access unit
  ISO/CD 11898-3: Road vehicles -- Controller area network (CAN) -- Part 3: Low-speed fault tolerant medium dependent interface
  ISO/CD 11898-4: Road vehicles -- Controller area network (CAN) -- Part 4: Time triggered communication

Three types of CAN controllers:

- ***2.0A controllers*** transmit and receive only Standard format messages
  - # of unique identifiers available to users, on a single 2.0A network, is 2,032 ($2^{11} - 2^4$).
- ***2.0B passive controllers*** will receive Extended format messages but then ignore them.
- ***2.0B controllers*** can send and receive messages in both formats.
  - # of unique identifiers available on a 2.0B network is in excess of 500 million!
  - 2.0B and 2.0B passive controllers can coexist on a network
  - If 29 bit identifiers are used on a bus which contains 2.0A controllers, the bus will not work!!!

CAN Bus physical level

- Physical layer is not part of the Bosch CAN standard
- CAN bus consists of two wires: CAN-High and CAN-Low.
  - Differential mode decreases noise interference
  - Voltage level in recessive state for idle bus
- Non return to zero (NRZ) signaling with “bit stuffing”:
  - Recessive high and dominant low states



ISO bus/transceiver standards

- **ISO 11898**
  - Twisted pair, shielded or unshielded.
  - Impedance of the cable = 120 12 ohms
  - Signal recessive state: CAN-high = 2.5v, CAN-low = 2.5v
  - Signal dominant state: CAN-high = 3.5v, CAN-low = 1.5v
    - for the recessive state, nominal voltage for the two wires is the same to decreases the power drawn from the nodes through the termination resistors.
  - 120 ohm termination resistors on each end of the wires.
- **ISO 11519**
  - Signal recessive state: CAN-high = 1.76v, CAN-low = 3.25v
  - Signal dominant state: CAN-high = 4.0v, CAN-low = 1.0v
  - No termination resistors required (due to lower bit rates)

CAN bus lengths

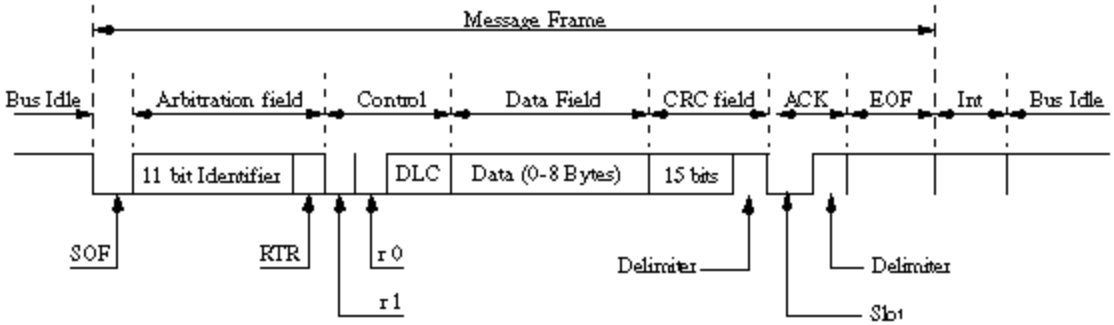
- The maximum bus length for a CAN network depends on bit rate.
  - Wave front of the bit signal needs time to travel to the most remote node and back again before the bit is sampled.

Bus length (meters)	Max bit rate
40	1 Mbit/s
100	500 kbit/s
200	250 kbit/s
500	125 kbit/s
6 km	10 kbit/s

CAN message frame types

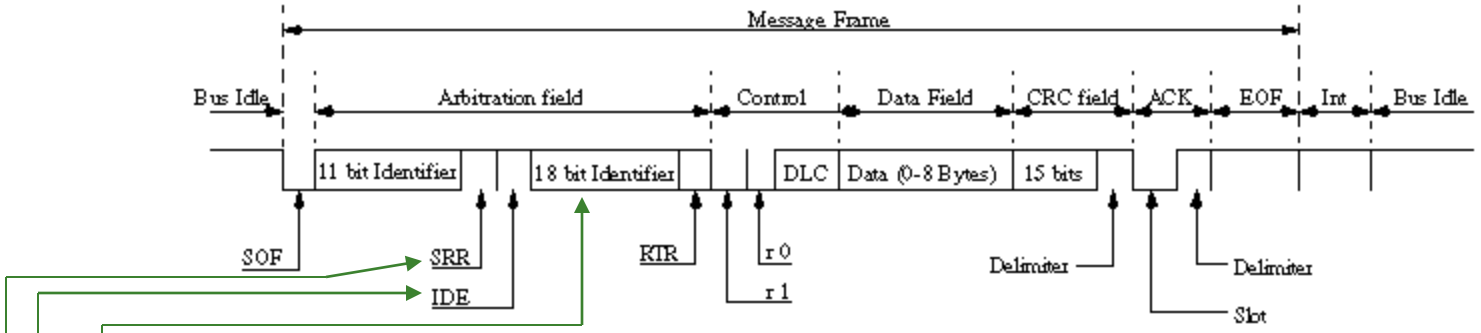
- Data frame used to transmit data
- Remote frame: information request
  - Transmitting node wants information of type given by identifier
  - Node with the information puts it into the data field
    - send at a later time if unable to respond immediately (BasicCAN)
- RTR bit in the message frame identifies the frame as data or remote

CAN 2.0A message frame



- Control field (6 bits)
  - * two dominant bits (r0 and r1) reserved for future use
  - *4-bit Data Length Code (DLC) = # bytes in Data Field
- Data field (0-8 bytes)
- CRC field (16 bits) – 15-bit cyclic redundancy check code and a recessive delimiter bit
- ACKnowledge field (2 bits).
  - Slot bit : transmitted as a recessive bit, but subsequently over written by dominant bits transmitted from all other nodes that successfully receive the message.
  - Recessive delimiter bit
- EOF (End of Frame) - 7 recessive bits
- Following EOF is an INTermission field (3 recessive bits)
  - After INTermission period the bus is recognised to be free.
  - Bus Idle time may be of any arbitrary length including zero.

CAN 2.0B message frame



- Extended format compatible with 2.0A format, with two main differences:
- Arbitration field contains two identifier bit fields.
  - Base ID: 11 bits for compatibility with Version 2.0A
  - ID extension: 18 bits (to give total identifier length of 29 bits)
  - IDE (Identifier Extension) bit indicates the format
- SRR (Substitute Remote Request) bit in the Arbitration Field
  - transmitted as a recessive bit to ensure that, in the case of arbitration between a Standard Data Frame and an Extended Data Frame, the Standard Data Frame will always have priority if both messages have the same base (11 bit) identifier.

Access Mechanism

- CSMA/CD+AMP – Carrier Sense Multiple Access with Collision Detection, with Arbitration on Message Priority
  - Highest priority message guaranteed access
    - Lower priority messages retransmitted later
  - Nodes transmit 11-bit IDs simultaneously
    - ID's prioritized - low #'s highest priority
  - On concurrent dominant/recessive bits, transmitter of recessive bit backs off
    - Similar to I^2C
  - By end of 11-bit address, all but one node should have backed off

Error detection and fault confinement

- The error detection, signaling and fault confinement defined in the CAN standard makes the CAN bus very reliable.
  - The built in error detection and signaling ensure correct and consistent information.
  - Faulty nodes go to modes where they do not disturb the traffic on the bus.
- **The CAN error process**
  - The error is detected by the a CAN controller (a transmitter or a receiver).
  - An error frame is immediately transmitted.
  - The message is cancelled at all nodes (exceptions exist).
  - The status of the CAN controllers are updated.
  - The message is re-transmitted. If several controllers have messages to send, normal arbitration is used.

Error detection by CAN controller

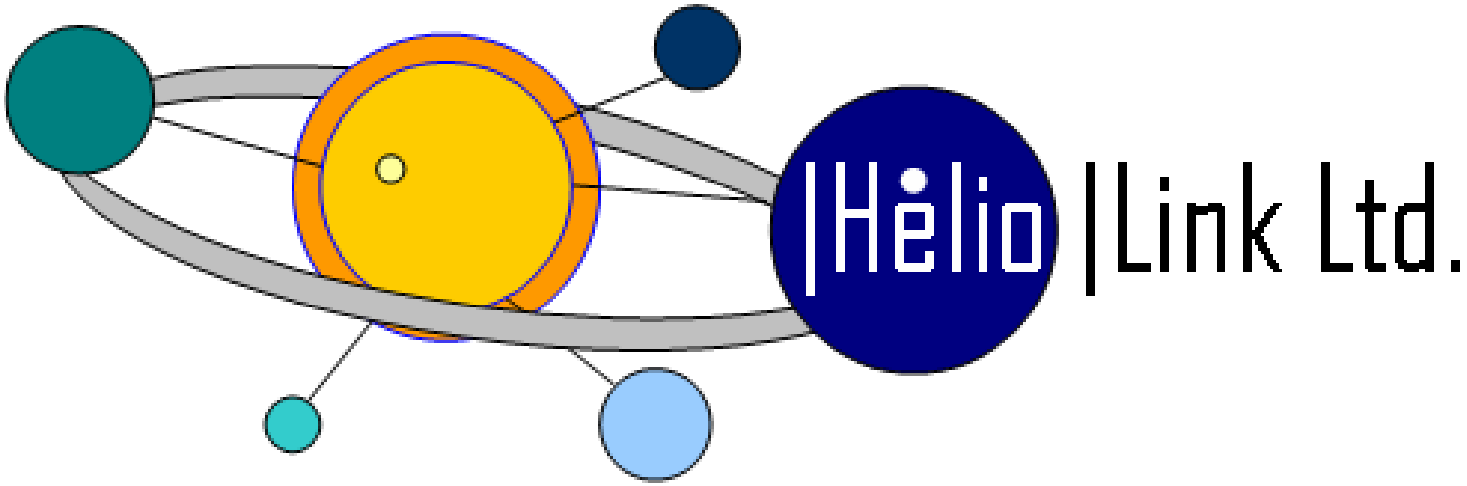
- Bit errors:
  - **Bit stuffing** error - transmitting node inserts a 1 after 5 consecutive 0s (and 0 after 5 consecutive 1s). Receiving node detects more than 5 consecutive bits as an error.
  - **Bit error**: Transmitting node reads back the message as it is sending. Error signaled if a different bit value from that sent (other than in the arbitration or acknowledgement fields).
- Message errors:
  - **Checksum error** – Checked by receiving node.
    - coefficients are generated modulo-2:
      X15 + X14 + X10 + X8 + X7 + X4 + X3 + 1.
  - **Frame error** - Receiver detects invalid bit in one of the defined frame positions.
  - **Acknowledgement Error** - Transmitter determines that a message has not been ACKnowledged.

CAN implementations

- BasicCAN
  - inexpensive microcontrollers with CAN modules
  - 2 receive buffers/1 transmit buffer
  - “interesting” messages filtered out by IDs
    - CPU must do final sorting of messages
  - no auto-answering remote frames
- FullCAN
  - high-performance controllers
  - message mailboxes supported
    - minimizes CPU work to sort messages
  - support auto-answering of remote frames

References

- Controller Area Network – *CAN Specification Version 2.0 Part A, Part B*, Robert Bosch GmbH, 1991.
- ISO International Standard – *ISO 11898 First Edition 1993 Road Vehicles – Interchange of Digital Information – Controller Area Network (CAN) for high-speed Communication*.
- *"Controller Area Network (CAN), an overview"* http://www.can-cia.org/can/ (CAN in Automation home page, with other CAN resources)
- *"Controller Area Network - CAN Information"* by Staffan Nilsson (Embedded Developers Pages) http://www.algonet.se/~staffann/developer/frames.htm



Solar Car CAN Development

Senior Design Project

Mike Cornelison
Beau Eckermann
David Last
Aaron Steiner
Luke Stewart
Brian Whitehousev

Sol of Auburn

- Student built and maintained vehicle
- Runs completely of solar energy
- Races long competitions

Problem Description

- Improve method of communication between subsystems of AU solar car
- Allow driver access to new features such as turn indicators, trip odometer, cruise control, etc.
- Provide driver and chase vehicle (via wireless modem) with real-time system information
- Improve vehicle safety by implementing a “safe mode” for powering down in the event of a system fault
- Reduce size, weight, power consumption

Systems

Motor Controller:

- Powers the motor
- Breaks the vehicle
- Monitors the primary electrical systems

Steering and Throttle:

- Acceleration
- Regenerative breaking
- Turing and breaking signals

Display and Power Controls:

- Speed
- Currents and voltages
- Switch bank

The Design Problem

- Bulky wire harness
  - Adds weight
  - Reliability and packaging problems
  - Noise due to other voltage systems

Solution Idea

Implement a Controller Area Network:

- Connect system devices
  - Motor controller
  - Steering and throttle
  - Display
- CAN Advantages:
  - Fewer wire count
  - Failure detection and safe mode
  - Weight loss
  - Easier installation and maintanance
  - Driver safety

Requirements

- Data handling Speed (1 Mbs)
- Safe Mode for driver safety
- Must run and not exceed 48 watts

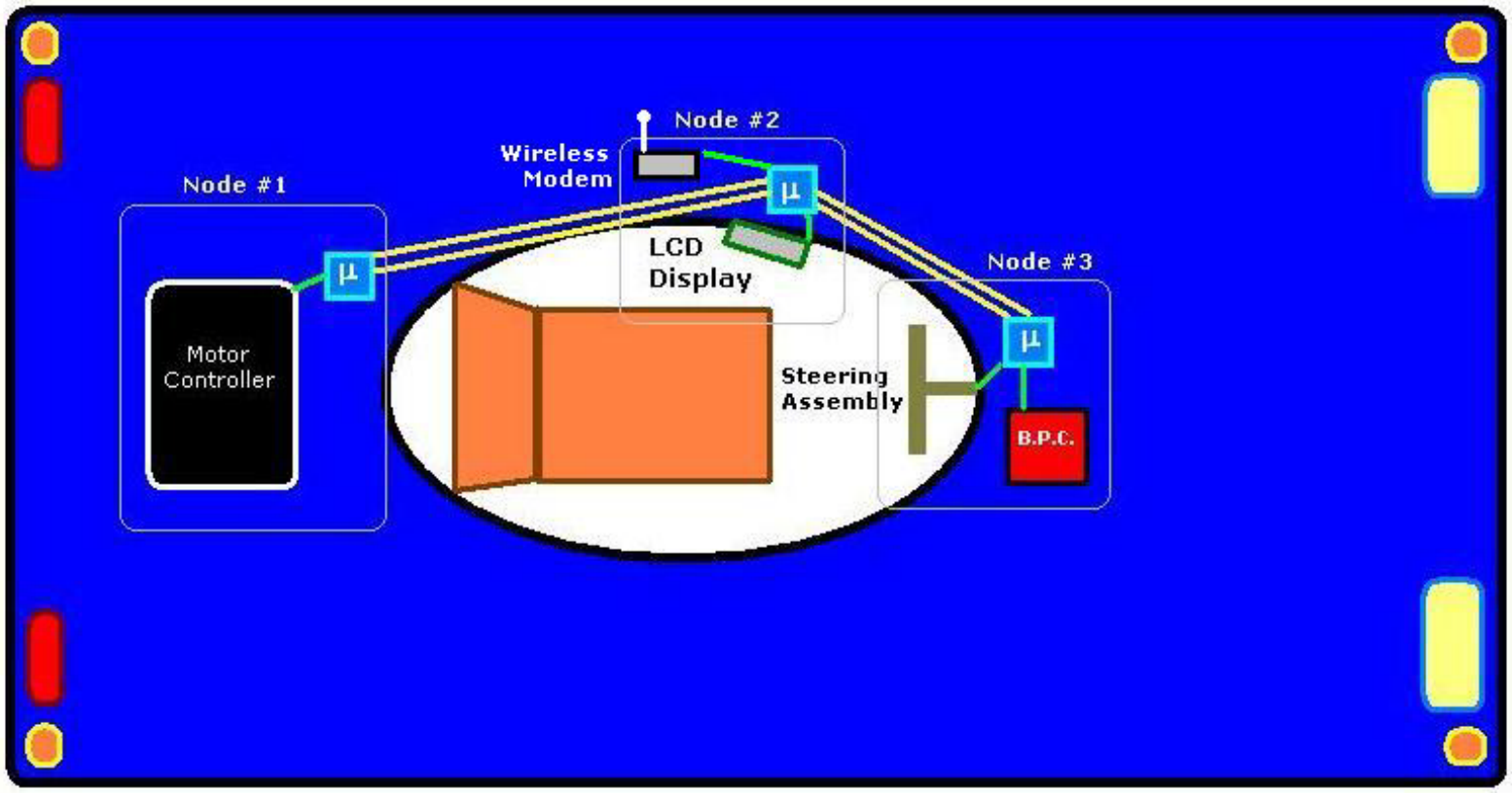
Constraints

- Low current limits
- Low power consumption
- Lightweight
- Functionality
- Cheap

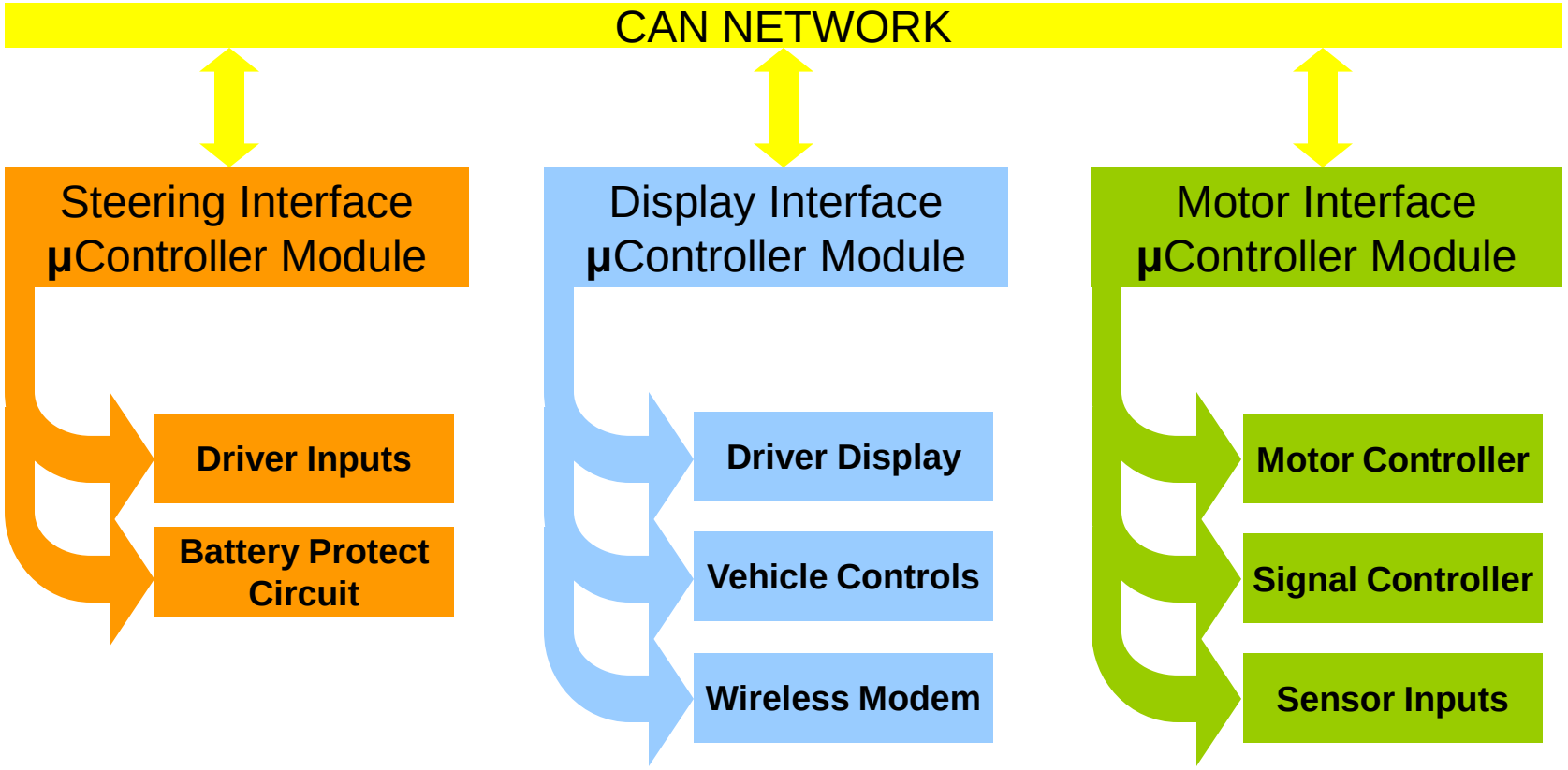
The Approach - Overview

- The Controller Area Network
  - Localized network of independent node devices
    - Most commonly microcontrollers
  - Standardizes communication format, arbitration, and addressing
    - Other network layers (i.e. Physical Layer) can be implemented as the designer sees fit
  - Hardware and Software must be implemented

Overhead Car Diagram



Network Architecture



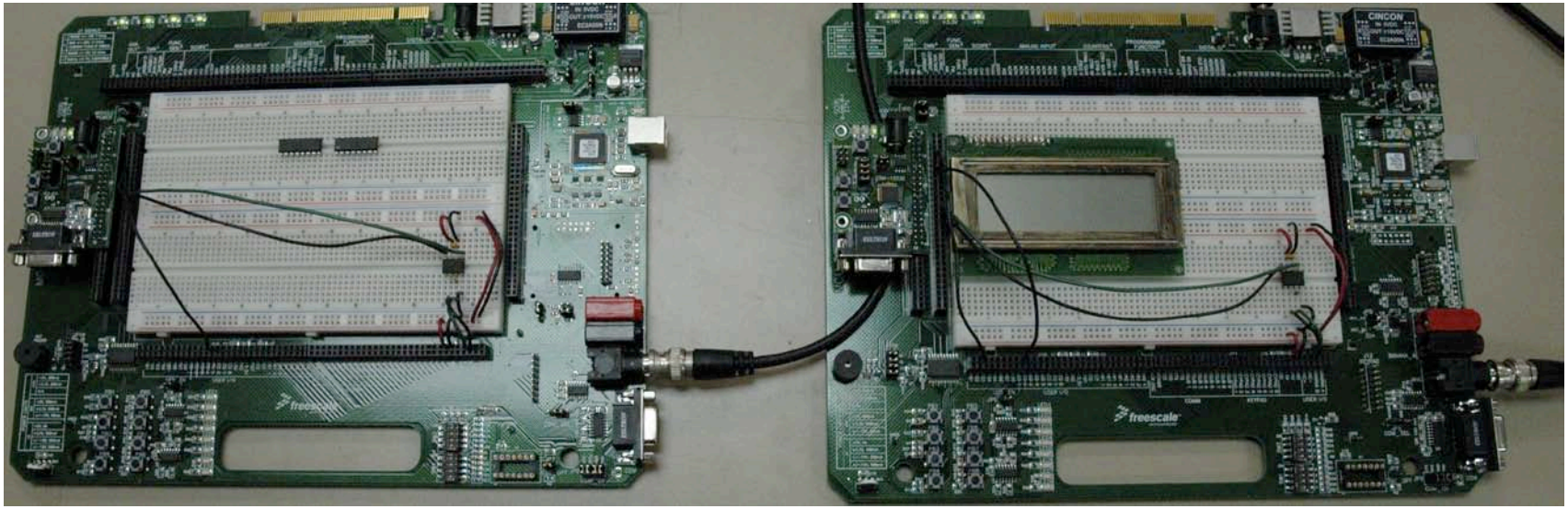
I/O Summary

I/O Name	I/O Label	Type	Source No	Source De	Destinatior	Destinatior
Driver Display	display	Digital	Display	-	-	-
Direction Control	dir	Digital	Display	Switch	Motor	Motor Cont
Hazard Lights	hazard	Digital	Display	Switch	Motor	Signal Con
Ignition Control	ignition	Digital	Display	Switch	Motor	Motor Cont
Mph/kph Toggle	mph/kph	Digital	Display	Switch	Display	Internal
Throtle Enable	threnable	Digital	Display	Switch	Motor	Motor Cont
Auxilary Battery Volta	aux_volt	Analog	Motor	Sensor	Display	Display
Break	brake	Digital	Motor	Motor Con	?	?
Main Battery Voltage	main_volt	Analog	Motor	Sensor	Display	Display
Solar Array Current	array_amp	Analog	Motor	Sensor	Display	Display
Solar Array Voltage	array_volt	Analog	Motor	Sensor	Display	Display
Speed Pulse	spdpulse	PWM	Motor	Motor Con	Display	Display
State of Charge	SOC	PWM	Motor	Motor Con	Display	Display
Break Light	brake_light	Digital	Steering	Switch	Motor	Signal Con
Cruise Control -	cc_down	Digital	Steering	Switch	Motor	Internal
Cruise Control +	cc_up	Digital	Steering	Switch	Motor	Internal
Cruise Control Set	cc_set	Digital	Steering	Switch	Motor	Internal
Display Control	disp_toggle	Digital	Steering	Switch	Display	Display
Left Turn Signal	left_turn	Digital	Steering	Switch	Motor	Signal Con
Regen	rgn	Analog	Steering	5K POT	Motor	Motor Cont
Right Turn Signal	right_turn	Digital	Steering	Switch	Motor	Signal Con
Throttle	thr	Analog	Steering	5K POT	Motor	Motor Cont

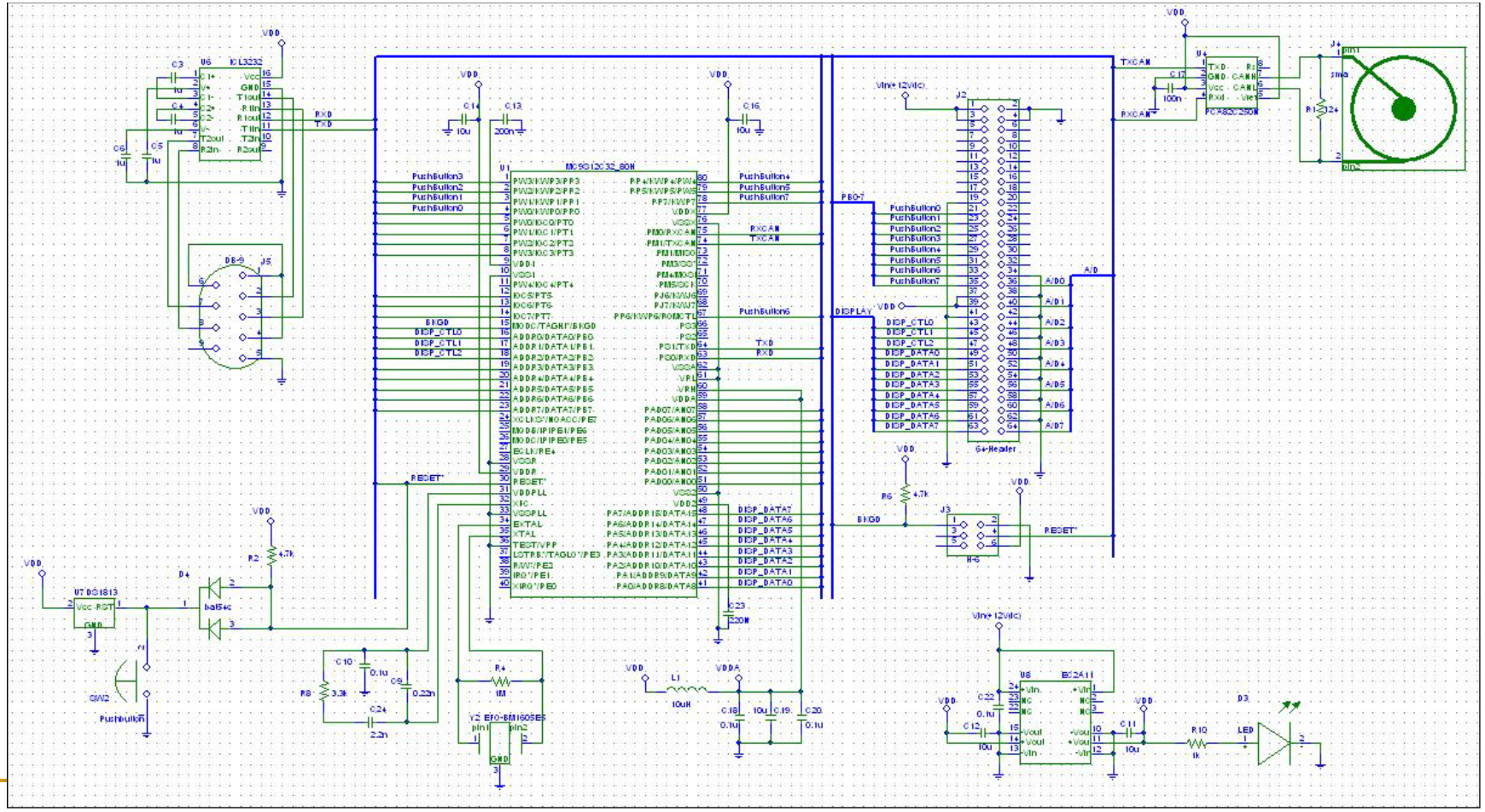
Hardware

- Freescale's MC9S12C32 microcontroller
  - Versatile microcontroller core with CAN interface
  - Plenty of input and output pins
  - Development hardware (MCU-SDK)
  - CAN Transceiver (Phillips PCA82C250N)
- Three modules throughout the car
  - Steering, Motor and Display Interfaces

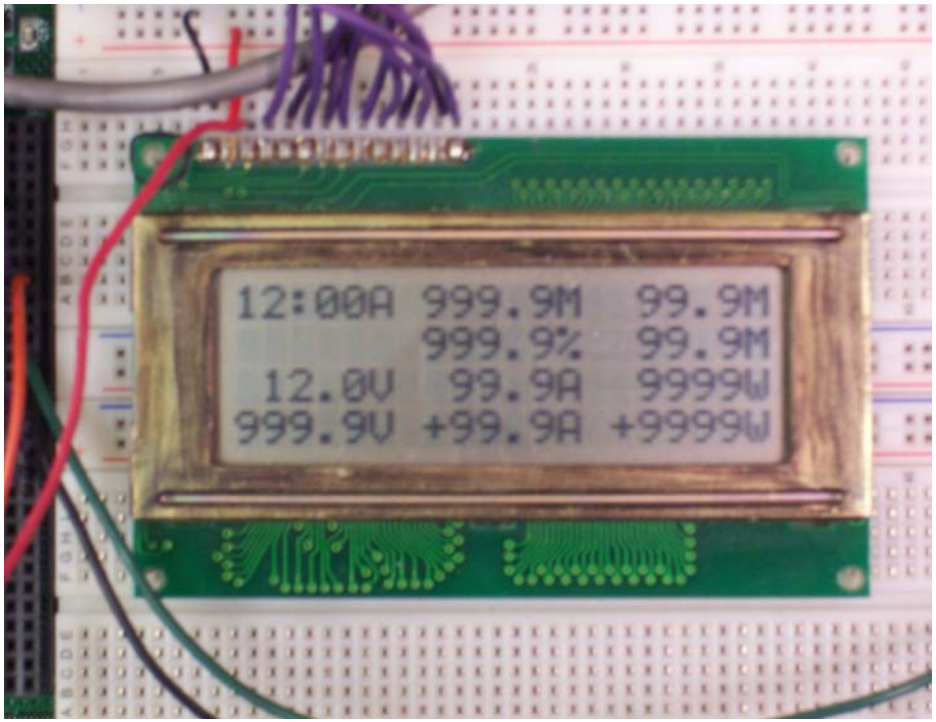
Development setup



Hardware Schematic



Optrex 20x4 LCD display



Software

- CAN communication
- Analog-to-Digital conversion
- Push button interrupts
- RS-232 communication
- Timers
- Display

Steering Interface Module (SIM)

- Collects driver inputs
  - Acceleration, deceleration, brake rate and turn signals
- Analog inputs (ADC)
  - Two potentiometers and one brake rate sensor
- Momentary switches (Interrupts)
  - Cruise control, turn signals, brake pedal, display toggle

Motor Interface Module (MIM)

- Analog outputs
  - Acceleration, deceleration, breaking rate
- Digital outputs
  - Cruise control, turn signals, brake lights
- Pulse width modulated input communicates motor speed and current
- Analog inputs
  - Battery and solar array current, battery voltages

Display Interface Module (DIM)

- Input from other modules
  - Analyzes and outputs information to an LCD display
- Input from bank of eight switches
  - Switches do not yet have a function

Software

- Sends and receives data over the network
- Collects data from car components
  - Decide priority structure of data
- Forces car into shut-down “safe mode” if network connectivity has been lost

Future Work

- Node formation for each subsystem
- CAN message hierarchy
- RS-232 packetization of streaming data
- Safe Mode implementation