CACC Truck Instrumentation and Software Development

PATH Research Report for

FHWA Exploratory Advanced Research Program

Cooperative Agreement DTFH61-13-H00012

Task 2.3 – Equip Trucks for Experiments

John Spring, David Nelson, and Xiao-Yun Lu

February 2018

Table of Contents

List of Figures

List of Tables

1 Introduction

This report documents the hardware and software development of the truck CACC system. Although, the Volvo trucks are controlled almost as it is in the sense that no sensors or control actuators have been added, several components are still necessary for the implementation of Connected Automated Vehicle technology which do not exist on the manufacturer's product. Those components include the following:

- Industrial PC-104 computer
- A laptop interfacing with the PC-104 computer for system development purpose
- Power supply system for the operation of all the added components
- DSRC communication units and the dual antennas
- Touch-screen tablet computer for driver vehicle interface (DVI)
- A network switch for timing and routing the interface of all the external components with PC-104 computer
- Safety Interrupt (or Emergency) Switch
- A 5 Hz DGPS (Global Positioning System) unit
- An X-PC box for data processing to fuse the radar and video camera data for front target detection and tracking

The functionality and installation of these components will be briefly introduced individually in this report.

The software development mainly includes the interfaces between those hardware components and the central PC-104 computer for real-time data exchanges: the PC-104 reading information from those components and sending back the commands, if applicable, to those components. It also includes the timing and scheduling of the processes running on the PC-104 in real-time like a director of an orchestra: which process needs to run earlier, which one to run later and how long it is expected to run. It is also responsible for the management of the data exchange by temporarily storing the data in a database and feeding the data to the control system. All those functionalities of the software have been built in submodules which are running in the Real-Time Operation System (RTOS) QNX-6. The QNX system is the real RTOS which has been used at PATH for nearly 30 years. It was initially developed for real-time and embedded control purposes, with a small kernel for high speed and reliable command execution. It has similar origins to the UNIX operating system. Those sub-modules will be described in more detail later.

2 Hardware System Development

2.1 PC-104 Housing

Although the PC-104 computer could be built up like a toy stack by mounting different boards with one on top of the other, an integrated housing is a neater solution, with convenient interface of the PC-104 with all other components connecting with it. Such integration also makes the physical connections more reliable. The following Figure 2-1 is an assembled PC-104 computer with mother board, solid state drive, CAN card and interface ports (serial and USB ports etc).

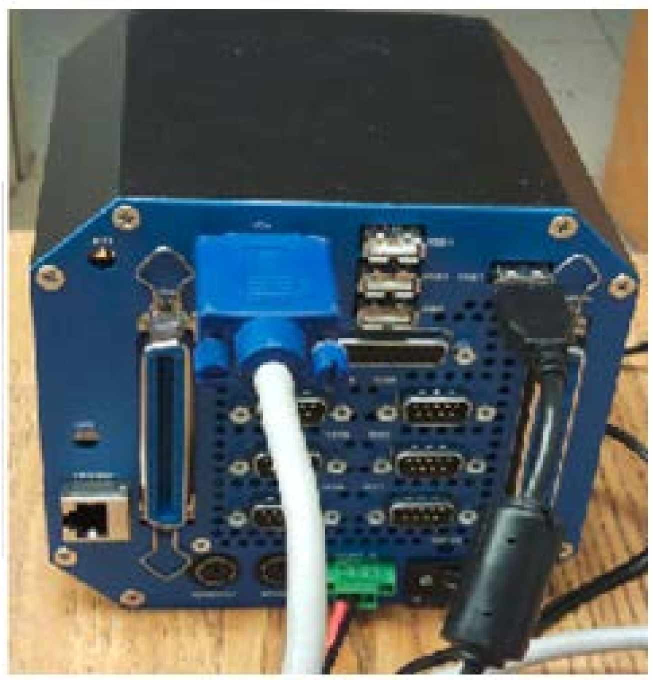
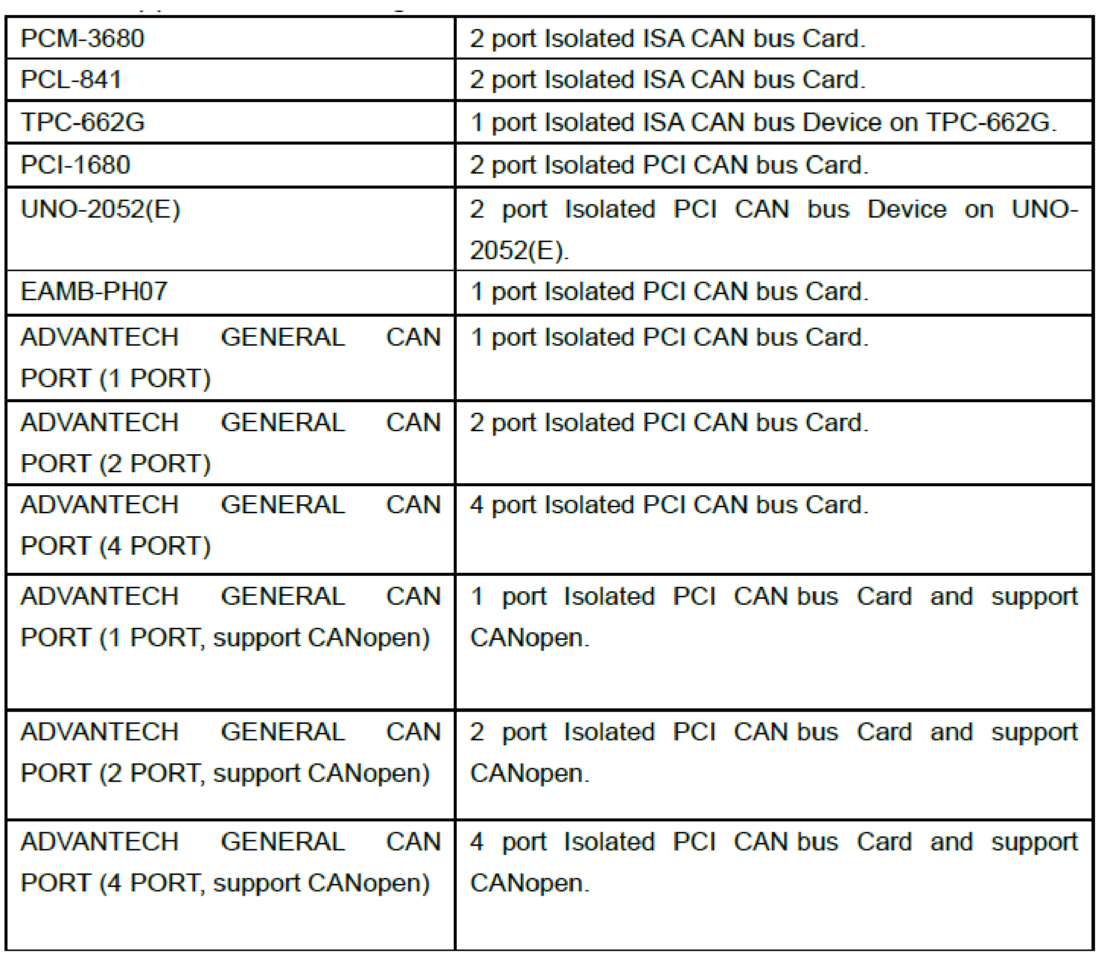


Figure 2-1 Assembled PC-104 and solid drive etc. with integrated CAN output interface port

2.2 Main Processor Board

Advantech PCM-3362 is a PC/104 plus form factor single board computer. Based on the Intel Atom N450 processor with computation speed at 1.66 GHz, single core capable of 2 threads and 512 kb of L2 core. The system was populated with 2GB of DDR2 (second generation double data rate) 667 5DRAM (Dynamic Random Access Memory). The PCM-3362 supports two RS-232 and one RS422/485 serial port and one Ethernet port at 10/100/1000 Mbps. The sinloge SATA port is populated with a Samsung SSD model 840 for 728 GB of storage. This system also supports four USB 2.0 ports and a video port supporting up to SXGA 1400-1060 @ 60 Hz. It is important that the Real-time Operating System QNX is compatible with this processor board. This driver supports QNX 6.5.X Intel x86 hardware platform. Table 2.1 lists the details of the CAN card used. Figure 2-2 is the Intel CPU (Central Processor Unit) of the PC-104 computer.

Table 2.1**.** Advcan QNX driver design for ADVANTECH CAN device in QNX OS.



PCM-3680	2 port Isolated ISA CAN bus Card.
PCL-841	2 port Isolated ISA CAN bus Card.
TPC-662G	1 port Isolated ISA CAN bus Device on TPC-662G.
PCI-1680	2 port Isolated PCI CAN bus Card.
UNO-2052(E)	2 port Isolated PCI CAN bus Device on UNO-2052(E).
EAMB-PH07	1 port Isolated PCI CAN bus Card.
ADVANTECH GENERAL CAN PORT (1 PORT)	1 port Isolated PCI CAN bus Card.
ADVANTECH GENERAL CAN PORT (2 PORT)	2 port Isolated PCI CAN bus Card.
ADVANTECH GENERAL CAN PORT (4 PORT)	4 port Isolated PCI CAN bus Card.
ADVANTECH GENERAL CAN PORT (1 PORT, support CANopen)	1 port Isolated PCI CAN bus Card and support CANopen.
ADVANTECH GENERAL CAN PORT (2 PORT, support CANopen)	2 port Isolated PCI CAN bus Card and support CANopen.
ADVANTECH GENERAL CAN PORT (4 PORT, support CANopen)	4 port Isolated PCI CAN bus Card and support CANopen.

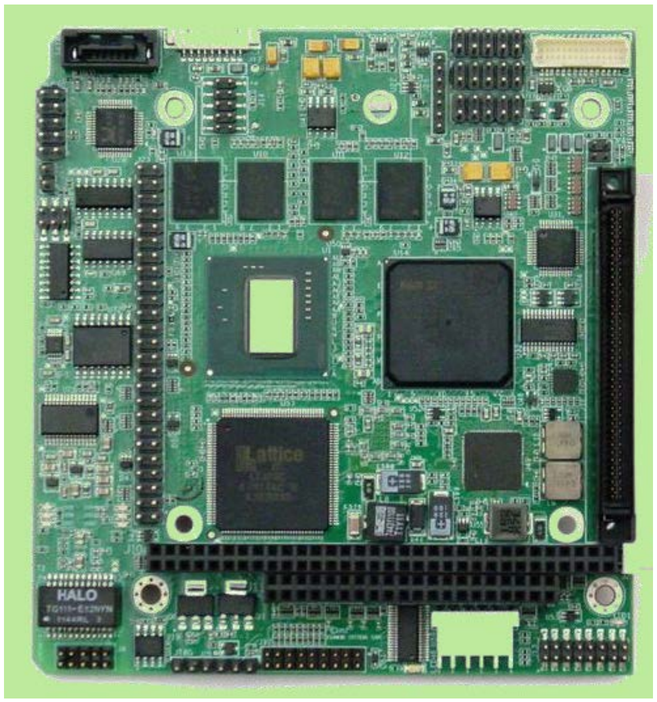


Figure 2-2. Intel Atom E640T CPU and Integrated Autocalibrating Data Acquisition

2.3 Controller Area Network Interface board

Advantech PCM-3680 is a PC/104 form factor interface card that operates two separate Controller Area Network (CAN) networks simultaneously. The separate parts have optical isolation up to 2,500 Vdc. The networked CAN ports are supported by the SSA-1000 CAN controller chip and operate CAN 2.0A/B standard up to 1 Mbps software programmable transfer rate.

2.4 Main system Power

The 5 volts required by the processor and interface board is powered through the M2-ATX_HV DC to DC power converter (Figure 2-3) supplied by mini-box. It is a 6V to 32V input Intelligent Automotive ATX power supply with programmable startup and shutdown characteristics. The main function is to maintain clean 5 volts over a variable and noisy input voltage range including down to 7.5 volts encountered during engine start phase.

Figure 2-3. M2-ATX_HV DC to DC power converter

2.5 Vehicle to Vehicle communications

The Industrial Technology Research Institute (ITRI) provided the ITRI WAVE/DSRC (Dedicated Short Range Communication) units (Figure 2-4). These units support communication packets that conform to the DSRC standards. The Dual Antenna Diversity enables broadcast/receive using 9 dB gain antennas placed on the right and left outside mirrors to maintain line of sight reception during turning maneuvers with trailers attached. The ITRI WAVE/DSRC communication unit (IWCU) operates over a wide range of automotive input voltages, contains a network port, single CAN port, serial port for console support, one USB 2.0 port, VGA, Dual 5.9 GHz antennas and 1 GPS antenna. The control computer communicated with the IWCU over the Ethernet port. To avoid blocking field of view between the antennas of the different trucks, dual antenna units were mounted on both side mirrors as shown in (Figure 2-5)**.** This required the ITRI DSRC unit to have two antenna outputs. This split was built internally to avoid conflict of output signals from the two outputs as shown in (Figure 2-5).

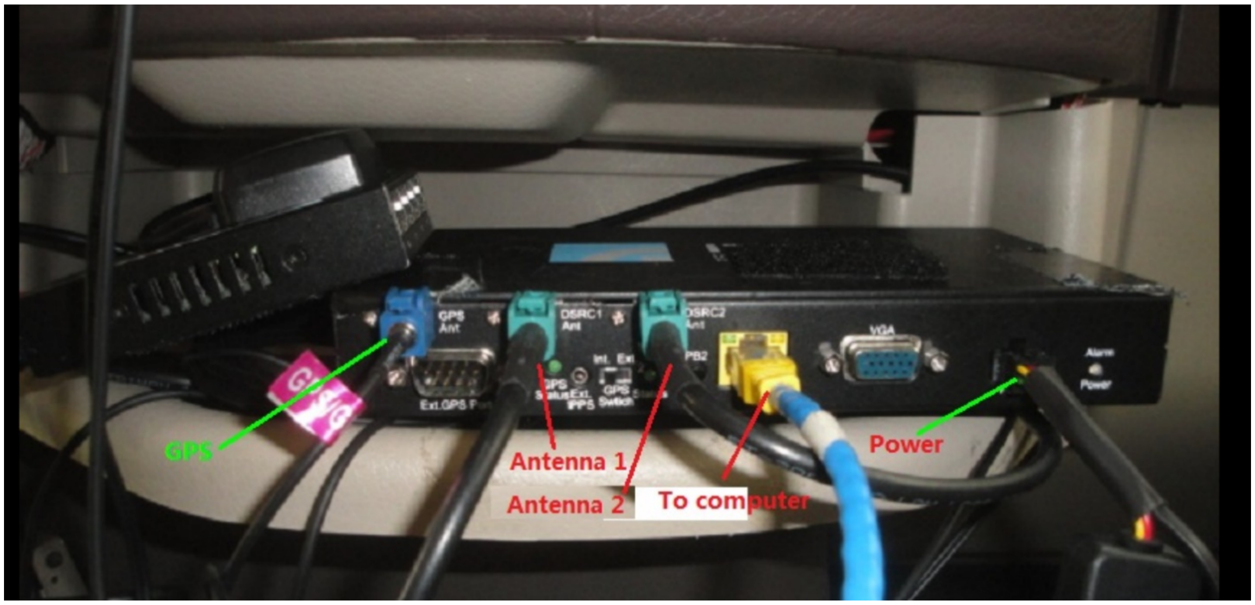


Figure 2-4. ITRI DSRC interface connections

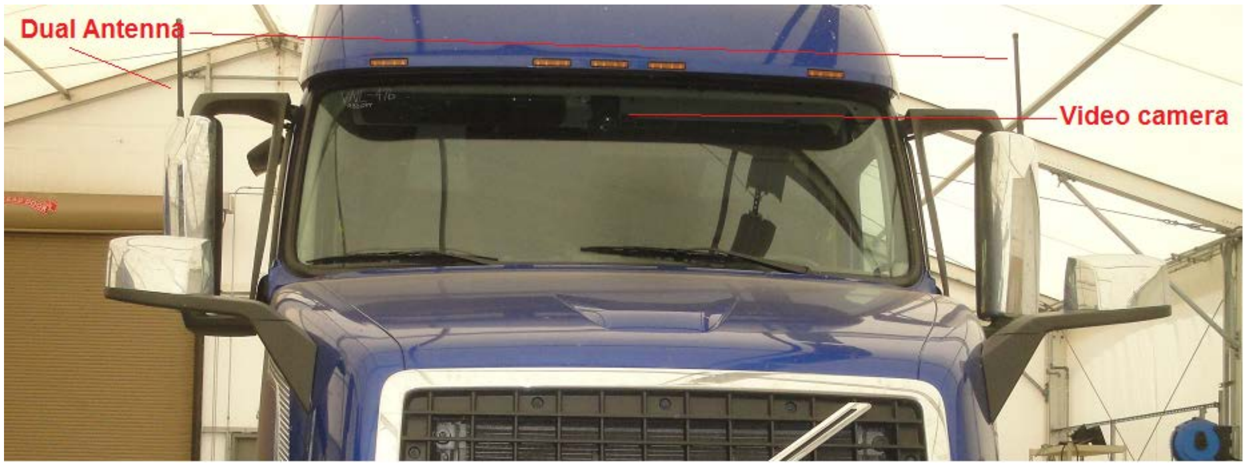


Figure 2-5. DSRC dual antenna installation on left and right mirrors.

2.6 In CAN Wi-Fi +DVI

A D-Link model AC1200 DIR-822 Wi-Fi router (Figure 2-6) was installed to provide WiFi access to the tablet computer used for Driver Vehicle Interface, a Samsung Galaxy Tab 52, This 8” tablet (Figure 2-7) mounted off the dashboard on the right side near the driver functioned as the driver vehicle interface. This DVI relays platoon status information collected over DSRC to the truck driver and sends change requests to the system over the touch screen interface.

Figure 2-6. D-Link model AC1200 DIR-822 Wi-Fi router

2.7 Tablet Computer for Driver Vehicle Interface (DVI)

The DVI is a Samsung Galaxy Tab S2 8” Tablet with 32 GB SD and WIFI capability. It has touch screen function for convenience of driver operation. It is connected with PC-104 computer through the D-Link WIFI router as in Figure 2-6. The DVI on each vehicle has its own local IP address to avoid the WIFI signal messing with DVI unit on other truck in short distance following.

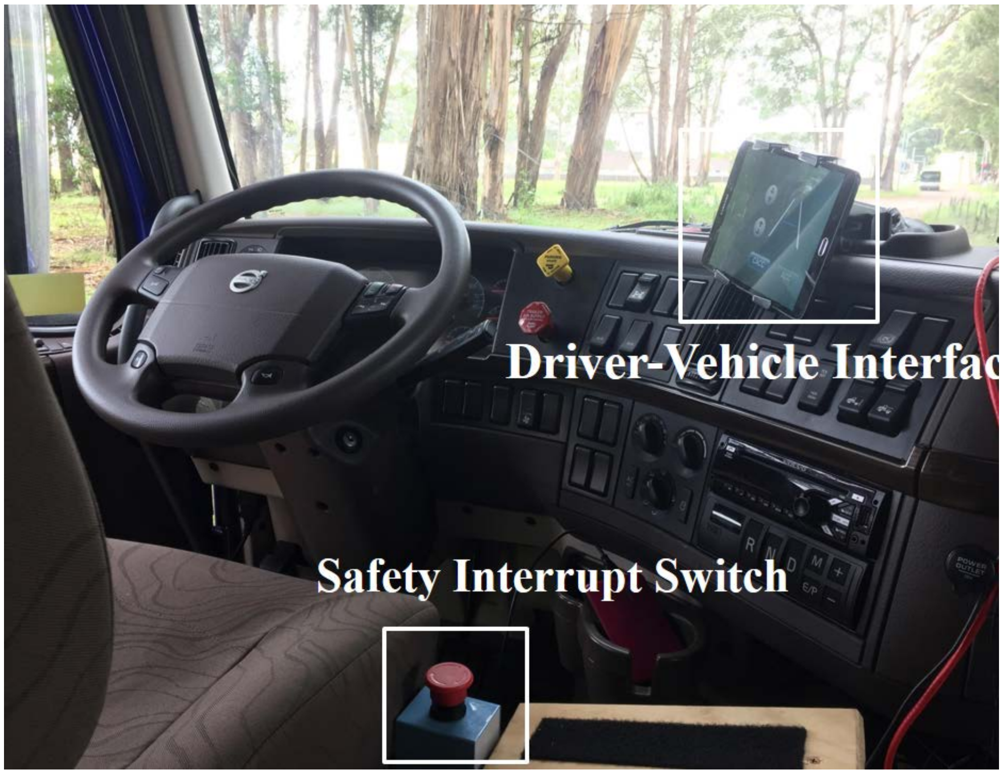


Figure 2-7. Safety Interrupt Switch and Driver-Vehicle Interface (DVI) Tablet

2.8 Network Switch

An 8-port industrial Gigabit Ethernet Switch (Figure 2-8) with hardened temperature range added connectivity for several devices. This switch was responsible for network connections among the control computer, the ITRI DSRC unit, the Wi-Fi router, and a laptop running Linux used to collect test data after runs and log into the control computer to start system processes remotely.

Figure 2-8 Black Box network switch

2.9 Safety Interrupt Switch

This was added to the right of the driver's seat to allow for a hard interrupt of the CAN based controller signals to the engine and braking control modules. This allowed for a relay in the normally open mode to connect the control computer to the truck control modules. The normally open mode when power was activated allowed for power faults or broken wires to default to the unactuated or manual control mode. The switch was installed near the driver seat for easy access as shown in (Figure 2-7).

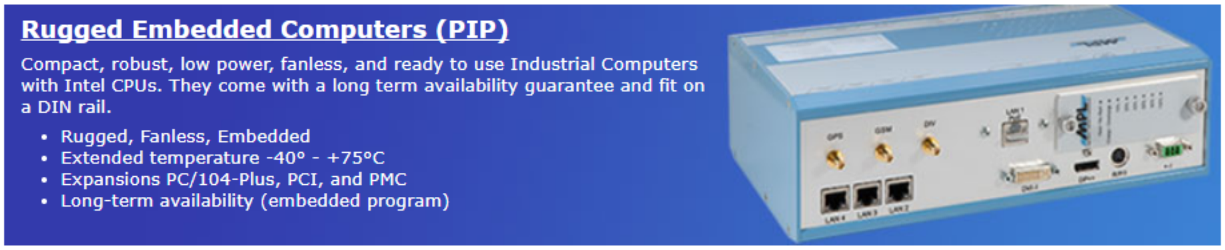
2.10 DGPS unit

A Garmin 18x-5hz GPS unit (Figure 2-9) was added to the computer through a serial port. It provides 5 position updates per second for use in position determination and time synchronization across trucks. The Differential effect comes from the WAAS (Wide Area Augmentation System) corrections.

Figure 2-9. Garmin 18x-5hz GPS unit

2.11 X-PC Box for Target Detection and Perception

X-PC is an industrial PC-104 computer produced by MPL (Figure 2-10) which runs Volvo proprietary software signal processing fusing the outputs from a Doppler radar and video camera (Figure 2-11) for front target detection and perception. The outputs include one detected (confirmed) front target with the following estimations: (a) target availability: 1: there is a front target; 0: no front target; (b) target distance; (c) target speed; and (d) estimated target acceleration. Targets in the adjacent lane and static targets such as guardrails and traffic signs, trees etc. are ignored. The target detection and distance, speed and acceleration estimation are mainly based on the Doppler radar measurement. The video data are mainly used for improvement of detection and perception with radar signals. For example, if the front target is a tractor only, the video camera data can be used to make sure that the front range of the subject vehicle is with respect to the end of the tractor instead of the back of the driver compartment; or when the truck is on a curve, the video camera data is used to help identify if the front target is in the same lane as the subject vehicle to reduce false positives.


Rugged Embedded Computers (PIP)

Compact, robust, low power, fanless, and ready to use Industrial Computers with Intel CPUs. They come with a long term availability guarantee and fit on a DIN rail.

- Rugged, Fanless, Embedded
- Extended temperature -40° - +75°C
- Expansions PC/104-Plus, PCI, and PMC
- Long-term availability (embedded program)

Figure 2-10. Industrial PC-104 computer by MPL-High Tech for sensor fusion

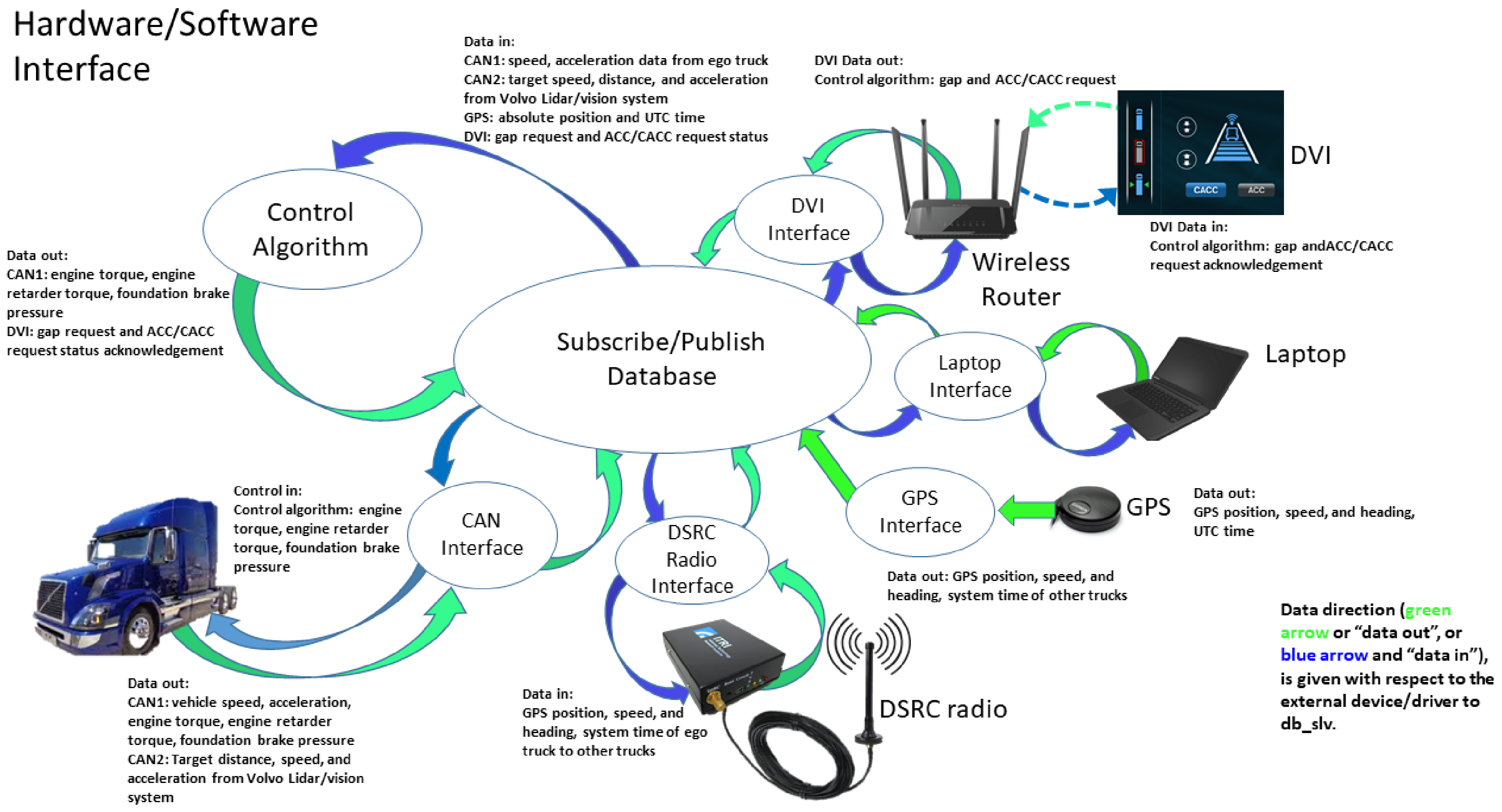
(a) Volvo truck built-in microwave radar

(b) Volvo truck built-in video camera

Figure 2-11**.** Volvo truck built-in radar (a) under middle the front bumper; Volvo truck built-in video camera (b) looking ahead inside the driver cabin at upper edge of the front window

3 Software System Development

Software control and data acquisition for the Cooperative Adaptive Cruise Control (CACC) system was constructed in three levels, described below. A schematic of the control system is depicted in Figure 3-1, in which the high-level software comprises the control algorithm and DVI, and the middle and low-level software comprises the subscribe/publish database and interfaces.



Hardware/Software Interface
Data in:
CAN1: speed, acceleration data from ego truck
CAN2: target speed, distance, and acceleration from Volvo Lidar/vision system
GPS: absolute position and UTC time
DVI: gap request and ACC/CACC request status
DVI Data out:
Control algorithm: gap and ACC/CACC request
DVI
CACC
ACC
DVI Data in:
Control algorithm: gap andACC/CACC request acknowledgement
Control Algorithm
DVI Interface
Wireless Router
Data out:
CAN1: engine torque, engine retarder torque, foundation brake pressure
DVI: gap request and ACC/CACC request status acknowledgement
Subscribe/Publish Database
Laptop Interface
Laptop
Control in:
Control algorithm: engine torque, engine retarder torque, foundation brake pressure
CAN Interface
DSRC Radio Interface
GPS Interface
GPS
Data out:
GPS position, speed, and heading, UTC time
Data out: GPS position, speed, and heading, system time of other trucks
Data direction (green arrow or “data out”, or blue arrow and “data in”), is given with respect to the external device/driver to db_slv.
Data out:
CAN1: vehicle speed, acceleration, engine torque, engine retarder torque, foundation brake pressure
CAN2: Target distance, speed, and acceleration from Volvo Lidar/vision system
Data in:
GPS position, speed, and heading, system time of ego truck to other trucks
DSRC radio

Figure 3-1 Schematic of Truck CACC data acquisition hardware/software interface on the Volvo VNL trucks.

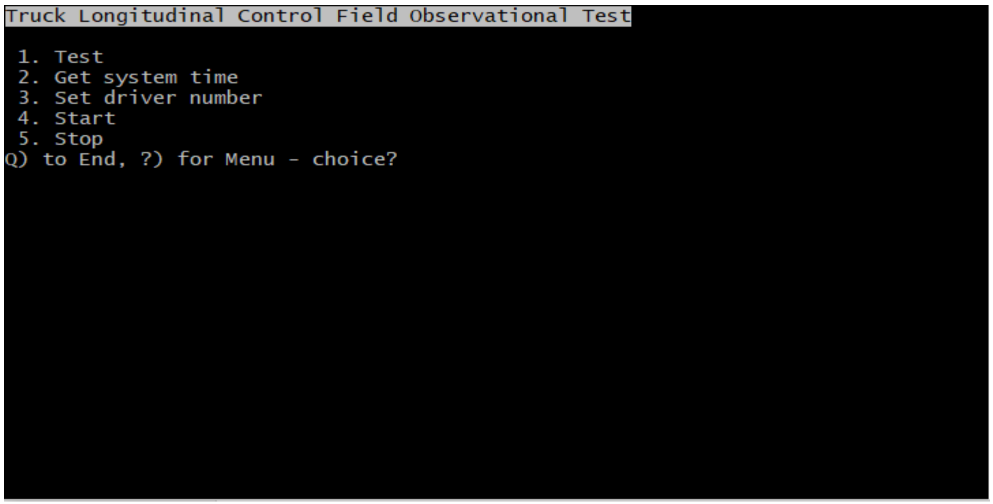
Is the schematic of CACC data acquisition hardware/software interface on the Volvo VNL trucks. The human truck driver sends control signals from the DVI to the control algorithm, which calculates the appropriate torque to send to the engine control module. The control algorithm also sends engine braking torque commands to the engine retarder and deceleration commands to the service brakes if needed. In this schematic, the software modules labeled "Interface" are a combination of a low-level driver and middle level database clients.

High-level software comprises the consumers of sensor data and control sources for the truck control modules and DSRC communication.

Middle-level software comprises the program used for inter-process communications (IPC), db_slv, and various programs used for reading sensor data from, and sending control messages to, the low-level drivers.

Low-level software comprises the hardware drivers necessary for communicating with hardware such as the truck Controller Area Network (CAN), GPS, Dedicated Short Range Communications (DSRC) radio, Driver Vehicle Interface (DVI) via a wireless router, and a laptop that was used for monitoring system status and starting it up.

The PATH software is started up in reverse order of the above list, with the low-level software being started first, then the middle-level software, and finally the high-level software. It is started either at boot time with a shell script, or more usually, by a trained truck driver or passenger using a simple menu program called simple_menu (see Figure 3-2).



```
Truck Longitudinal Control Field Observational Test

 1. Test
 2. Get system time
 3. Set driver number
 4. Start
 5. Stop
Q) to End, ?) for Menu - choice?
```

Figure 3-2 Simple menu script for testing, starting, and stopping PATH CACC software.

Figure 3-2 above is a screen shot of a simple menu script for testing, starting, and stopping PATH CACC software. The operator presses a '1' to run a series of tests on the CAN buses, DVI communication, and GPS system time. The control software is started by pressing '4', and stopped after the run by pressing ‘5’.

After starting the truck, the operator types a ‘1’ at the menu prompt, and the script runs a series of tests for basic operation of the CAN bus, DVI communication, and GPS system time. If there is an error, the operator checks hardware configuration such as position of the kill switch in the absence of CAN traffic, or connection with the wireless router if the DVI does not change its display state. (Troubleshooting tips are suggested on the Simple Menu screen if there is an error.)

Once system testing is completed successfully, regular operations may begin. The operator presses a ‘4’ to start up the control software and data logging. Software startup is described below, in the “Middle-level Software” section. The truck driver starts driving manually, and once the truck reaches minimum cruise control speed (at least 15 mph), he/she enables cruise control and the system begins active control. When the run is completed, the truck driver taps the service brake to disable cruise control, and the operator types a ‘5’ on the Simple Menu display to stop the software and complete data logging.

3.1 Brief Description of Single and Multiple Vehicle Control

For the purposes of this section, a brief description of how a single truck is normally controlled, and the addition of communication for multiple vehicle cooperative control, is in order here.

3.1.1Manual and Cruise Control (CC)

There are five hardware Electronic Control Modules (ECM) that directly pertain to truck control: engine, engine retarder, transmission, braking, and instrument control panel. An Electronic Control Unit (ECU) serves as the “brain” that coordinates the other control modules. They all must interact with each other for proper control. For instance, if the driver sets a cruise control speed on the instrument panel, messages must be conveyed to the engine to provide the correct positive torque to match the set speed, or negative torque to the engine retarder to slow the truck to that speed. The transmission must choose the correct gear and gear progression to reach the set speed given the torque changes. These messages are transmitted on the truck's Controller Area Network (CAN), and the values of the various physical parameters are calculated by firmware in the ECU.

3.1.2Adaptive Cruise Control (ACC)

For Adaptive Cruise Control (ACC), Volvo has added a video + radar subsystem that searches for targets in front of the vehicle, as well as the firmware needed to control the ECMs via the truck CAN network (CAN1). This system is contained in a Volvo Enhanced Control unit (VECU). If a target is acquired, the cruise control system is notified, and the truck modifies its speed to the minimum of either the set speed, or the preceding target speed, given a time gap that is also set by the driver. Volvo engineers also added the transmission of target parameters ("target acquired", target distance, speed, and acceleration) on a separate CAN bus (CAN2) to the PATH control computer so that PATH could use that data in its Cooperative Adaptive Cruise Control (CACC) system.

3.1.3Cooperative Adaptive Cruise Control (CACC)

PATH engineers have augmented Volvo's VEC with inter-truck communication among multiple trucks. Each truck transmits its own CAN1 and CAN2 data, as well as its GPS position, to other similarly-equipped trucks This allows for multiple truck platooning, since each truck immediately knows the position, braking status, engine torque, engine retarder torque, etc. of the other trucks, thus allowing for much faster response times. Communication is done wirelessly, via DSRC radios, using the SAE J2735 Basic Safety Message (BSM) that contains GPS position, vehicle speed and acceleration. We also added CACC-specific fields beyond the standard BSM message contents.

3.2 High-level Software

High-level software comprises the control sources and consumers of sensor data for the truck control modules and DSRC communication. These programs are the control algorithm, long_trk, and the driver-vehicle interface, comprising the Galaxy Tab S2 tablet computer running the Qt program PATH_App_Berkeley, written by Volvo and customized by PATH.

3.2.1Path_App_Berkeley

PATH_App_Berkeley is a Qt program running on a Samsung Galaxy Tab S2 that serves as the Driver-Vehicle Interface (DVI) between the human truck driver and the PATH CACC system. The DVI (see figure 3) is a touchscreen display that allows the driver to decrease/increase the time gap between his truck and the preceding truck (the two sets of arrows) and set the following algorithm to either Adaptive Cruise Control (ACC) or Cooperative Adaptive Cruise control (CACC) by pressing the appropriate button on the DVI. A button press triggers a UDP (User Datagram Protocol) message that is sent wirelessly from the tablet to the control computer via a dLink Model 822 wireless router. Status messages (e.g. control status of the ego truck or other trucks, braking, target availability, communication status, time gap) are sent by the control computer to the DVI.

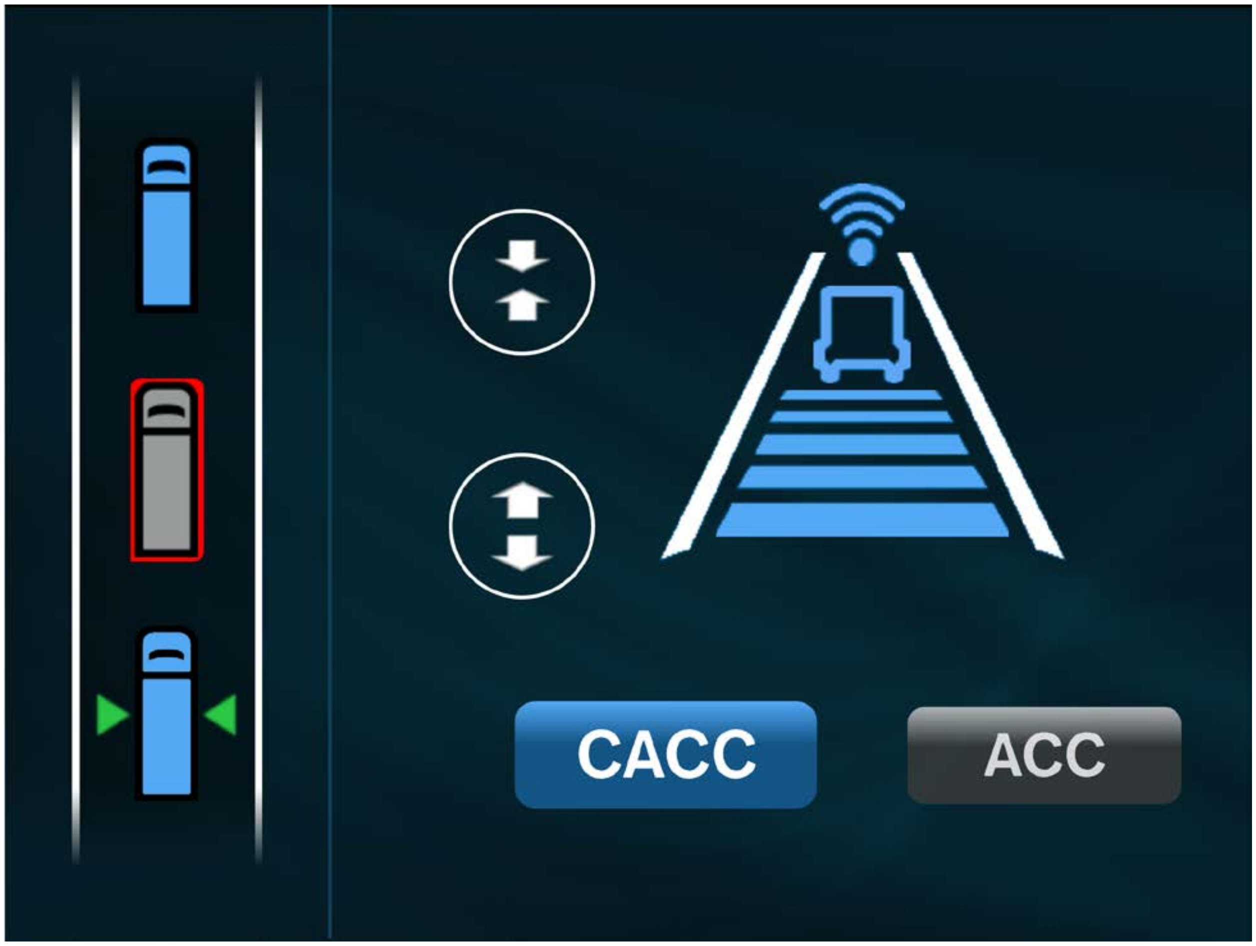


Figure 3-3. Screenshot of the Driver Vehicle Interface (DVI)

Figure 4.1 is a screen shot of the Driver Vehicle Interface (DVI). Its main functions include: (a) for the driver to learn the current status of several critical components such as vehicle position in the platoon, driving mode (manual, CC ACC or CACC), DSRC health, service brake use of all the vehicles in the platoon; and (b) for the driver to select drive mode between ACC or CACC (for the following trucks, since the lead truck is always in CC or ACC) and Time-Gap for ACC and CACC driving mode. More detailed DVI descriptions were presented in a separate report. UDP messaging is used to send/receive messages from/to the control algorithm. The two sets of arrows send time gap requests, and the CACC/ACC radio button sends CACC or ACC control mode requests. The current status of the control system is contained in UDP messages received from the control computer.

3.2.2long_trk

long_trk is the control program for CACC control of all three of the trucks. It receives data from the host truck CAN bus and data communicated from the other trucks and calculates the correct engine/engine retarder torque and braking pressure to send to the engine, transmission, and engine retarder control modules. It also determines the status messages to send to the DVI.

3.3   Middle-level Software

Middle-level software comprises the **database server** program used for inter-process communications (IPC), db_slv, and various client programs used for reading sensor data from, and sending control messages to, the low-level drivers. The order in which the middle-level software is started is important; db_slv is started first, and then the **database clients** are started up in the following order: trk_cr, can_man, rdj1939, jbussend, jbussendGPS, gpsdb, trk_wr, veh_snd, veh_rcv, dvi_snd, and dvi_rcv. long_trk is also a database client, and it is started last. The interface schematic of Figure 3-1 is repeated in Figure 3-4 with the middle-level software and low-level drivers distinguished by the outline color and including the program names of the software used.

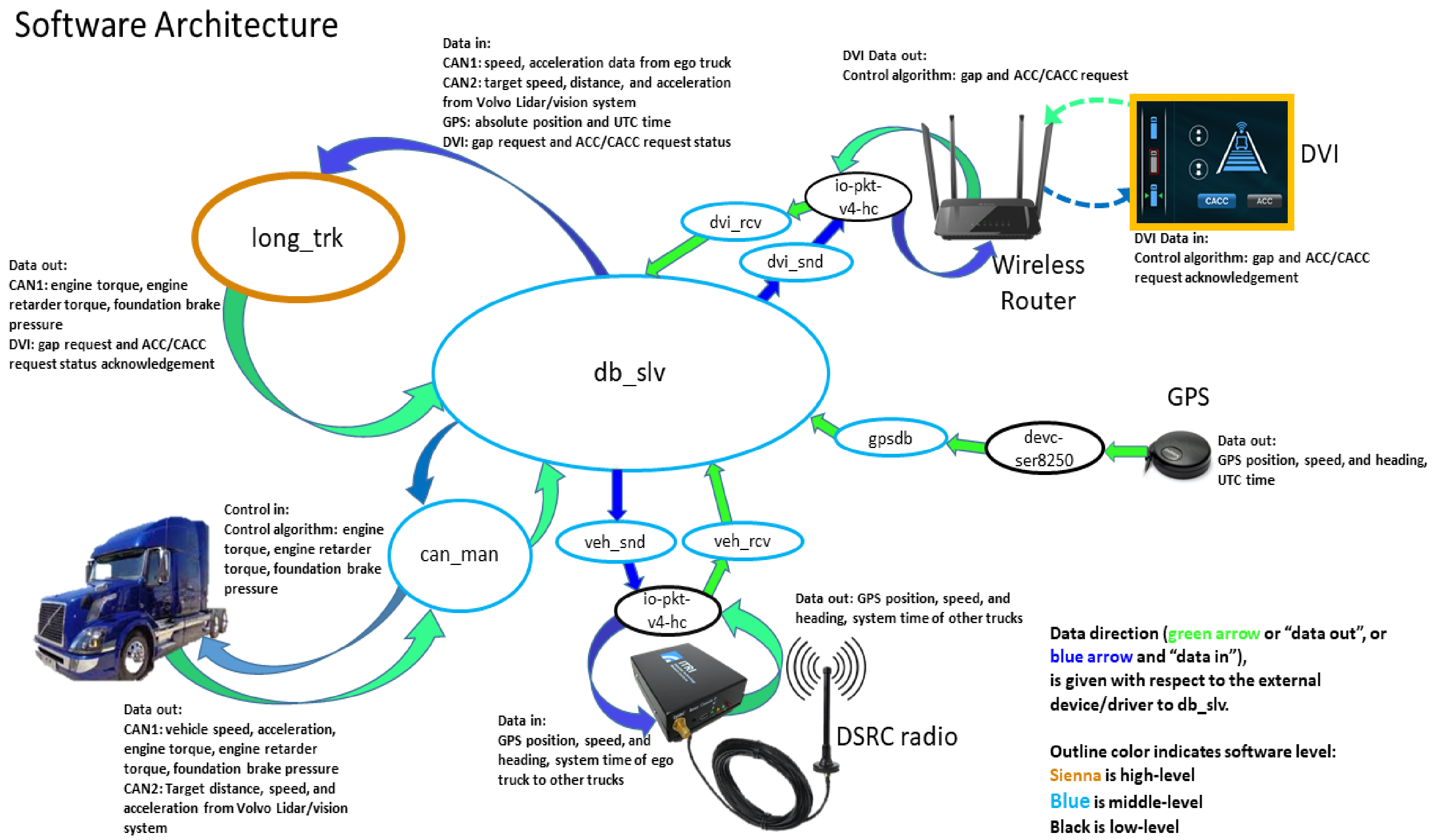

Software Architecture
Data in:
CAN1: speed, acceleration data from ego truck
CAN2: target speed, distance, and acceleration from Volvo Lidar/vision system
GPS: absolute position and UTC time
DVI: gap request and ACC/CACC request status
DVI Data out:
Control algorithm: gap and ACC/CACC request
DVI
DVI Data in:
Control algorithm: gap and ACC/CACC request acknowledgement
long_trk
io-pkt-v4-hc
dvi_rcv
dvi_snd
Wireless Router
Data out:
CAN1: engine torque, engine retarder torque, foundation brake pressure
DVI: gap request and ACC/CACC request status acknowledgement
db_slv
GPS
gpsdb
devc-ser8250
Data out:
GPS position, speed, and heading, UTC time
Control in:
Control algorithm: engine torque, engine retarder torque, foundation brake pressure
can_man
veh_snd
veh_rcv
io-pkt-v4-hc
Data out: GPS position, speed, and heading, system time of other trucks
Data direction (green arrow or "data out", or blue arrow and "data in"), is given with respect to the external device/driver to db_slv.
Data out:
CAN1: vehicle speed, acceleration, engine torque, engine retarder torque, foundation brake pressure
CAN2: Target distance, speed, and acceleration from Volvo Lidar/vision system
Data in:
GPS position, speed, and heading, system time of ego truck to other trucks
DSRC radio
Outline color indicates software level:
Sienna is high-level
Blue is middle-level
Black is low-level

Figure 3-4 Copy of figure 1 with the middle and low-level software separated and with all programs named.

The above *Figure 5.1* is similar to figure 1 with the middle and low-level software separated and with all programs named. It shows the logical relationships of all the programs. The functionalities of major named programs are described in more detail in the following subsections.

3.3.1db_slv

db_slv is a publish/subscribe database server. It is the hub for data and control exchanges among the high and middle-level programs. When started up, db_slv registers the database with the QNX operating system and returns a channel ID so there is a target for clients to attach to. It then sets up synchronous message handling so that when a message is received, new messages are blocked until the first one is dealt with.

When a database client connects to db_slv, the client sends a numerical name (a **database variable**), a pointer to a C structure, and the size of that structure, and db_slv allocates memory for it and sets up the messaging system that will be used for further communication. The client may also request that a trigger be associated with this database variable, so that if the values of one of the C structure members changes, the client will be notified with a message, and it can then deal with the new data.

3.3.2 trk_cr

trk_cr creates and initializes all the database variables that the database clients will use. This program was created because if a client tries to read or write a nonexistent database variable, an error is generated, and that client may terminate. Also, it’s just easier to have all the database variable creates and initializations in one place.

3.3.3 can_man

can_man is the CAN manager. It serves as both the low-level driver and medium-level database program for reading data from the two CAN networks on the truck and writing this raw binary data to the db_slv database. It also reads control signals from the database and writes them onto the CAN buses. Triggering for certain database variables was enabled so that whenever control signals are written to the database, can_man will be triggered to read them and forward them to the CAN buses. It was written by PATH to configure and communicate with two different chipsets: Phillips SJA1000 and Intel i82527, before those companies provided QNX drivers of their own. can_man may be configured for either chipset. We use the RTD ECAN527D Dual-port CAN card, so can_man is configured for the Intel chipset.

Since the ECN527D CAN card communicates on the older ISA bus, it is not a plug-and-play device. So, its configuration parameters (viz. interrupt request line and base address) are set with jumpers on the card (see Figure 3-5). The base address of CAN1 is set on jumper JP3 for 0xda000. The base address of CAN2 is automatically assigned to an offset of 0x200 above CAN1, so its base address is 0xd200. The ECAN527D interrupt outputs are connected to the interrupt lines on the ISA bus using jumpers JP1 for CAN1 and JP2 for CAN2. In the figure, CAN1 is set to IRQ10 and CAN2 is set to IRQ7.

When can_man is started up, the same configuration set with the jumpers on the CAN card must be conveyed to the software. This is done with options sent to can_man. For CAN1, can_man is started with the following command:

```
can_man -n /dev/can1 -s 250 -i 10 -p 0xda000 -e 1
```

For CAN2, can_man is started with the following command:

```
can_man -n /dev/can2 -s 500 -i 7 -p 0xda200 -e 1
```

As may be surmised from the above hardware description, -n gives the device name as assigned by the QNX operating system, -i gives the interrupt line to be used by the software, and -p gives the base address of the I/O map into system memory. The -s option gives the speed of the particular CAN bus; 250,000 bits/second for CAN1 and 500,000 bits per second for CAN2. The -e option tells can_man whether the standard length of the CAN message header (11 bits) or the extended length (29 bits) is being used. Both CAN buses are using the extended header length.

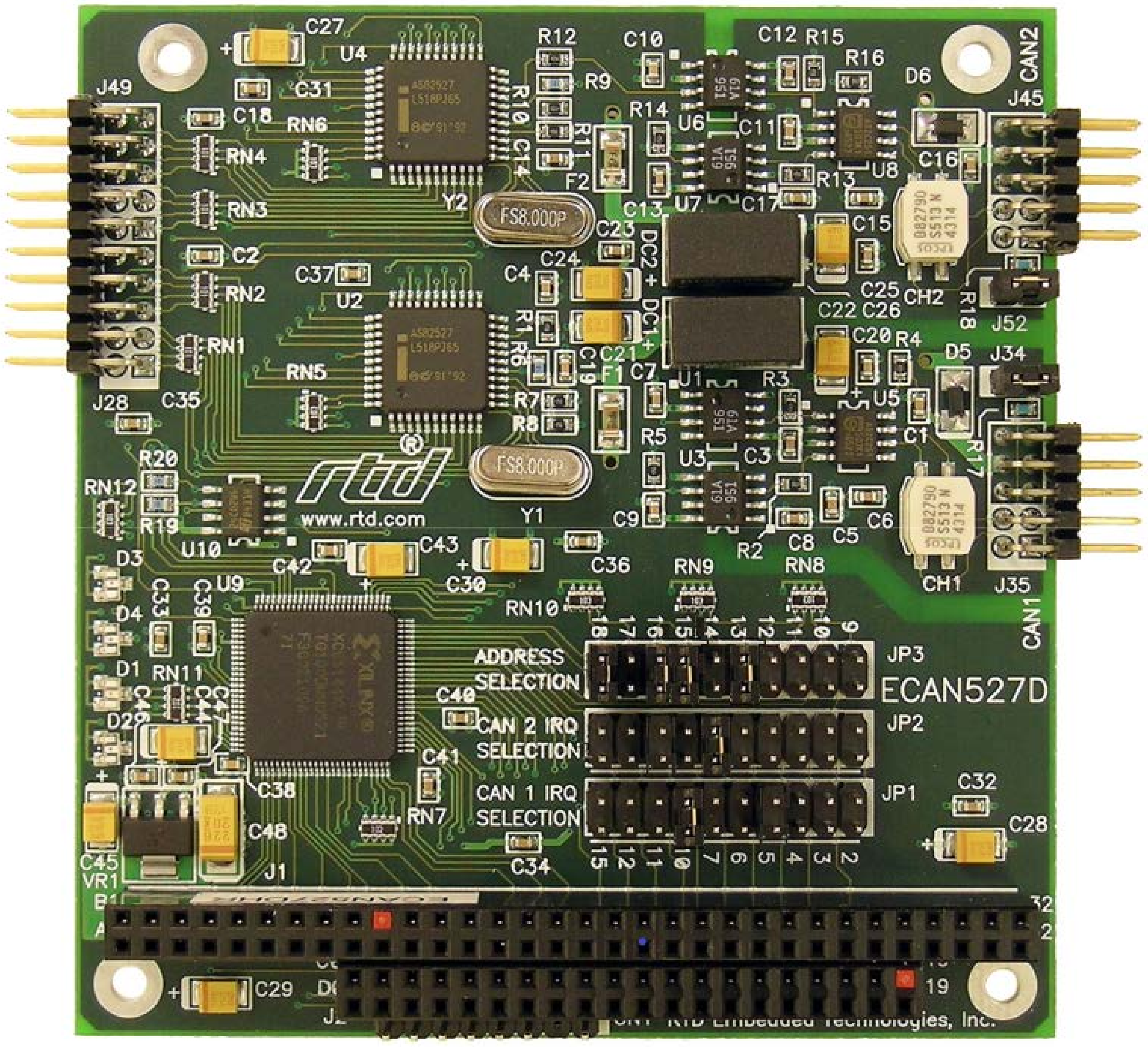


Figure 3-5 RTD ECAN527D CAN card configured for Volvo truck control.

Figure 3-5 is a photo of the RTD ECAN527D CAN card configured for Volvo truck CACC of the project. Jumper JP3 is configured for a base address of 0xDA000 for CAN1 and 0xDA200 for CAN2. Jumper JP1 configures CAN1 for IRQ10, and jumper JP2 configures CAN2 for IRQ7.

3.3.4 *rdj1939*

rdj1939 reads the SAE J1939-formatted binary CAN messages that were written to the database by can_man, parses the binary data into the appropriate integers, floating points, and flags, and writes the results back into the database.

For the truck CAN, CAN1, there is a standard set of messages defined by SAE International for automotive uses. Information such as engine torque, brake pedal position, brake pressure, engine retarder torque, cruise control switch status, cruise control set speed, and so on, are all transmitted on CAN1 periodically, at intervals ranging from 10 ms to 100 ms. Control signals from long_trk must also be transmitted periodically, with a bus error resulting from missed messages.

For the Volvo target acquisition system on CAN2, a small set of J1939-formatted messages was agreed to among Volvo and PATH engineers. One message provides target availability and target distance, speed, and acceleration. Another message provides ego (self) vehicle speed and acceleration.

3.3.5 jbussend and jbussendGPS

jbussend and jbussendGPS are essentially the same program. They both read control signals from the database, convert them into J1939 messages, and periodically write the binary data back to the database. For jbussend, database write to certain database variables will trigger database reads of those same variables by can_man, which then forwards the messages onto the appropriate CAN bus. jbussendGPS writes directly to its CAN bus. The main difference between jbussend and jbussendGPS is that jbussend writes control variables associated with the truck CAN and jbussendGPS directly writes CAN messages to the Volvo targeting system CAN.

As its name implies, jbussendGPS reads GPS position and UTC time data that was written to the database by gpsdb, packages it into the custom CAN message agreed to by PATH and Volvo engineers, and writes the data directly to CAN2. jbussendGPS also reads the control variables for engine and engine retarder torque and writes those to CAN2 in the same message. All of these data are needed for the Volvo targeting system for predicting truck trajectory.

3.3.6 gpsdb

gpsdb is the database client that opens a connection to one of the serial ports that has been configured by ser-8250, the serial port driver. A Garmin 18x-5 Hz GPS is connected to that serial port, and outputs GPS position, speed, altitude, heading, number of satellites, UTC time, and horizontal degree of precision. gpsdb reads and parses the NMEA sentence and writes the data to the database. GPS data are used for various purposes: truck positioning in platoon, system time synchronization (and therefore data synchronization on different trucks), and trajectory prediction, among other things.

3.3.7 *trk_wr*

trk_wr is the main data logging program for the control system. It collects all the database variables at 100 ms intervals and writes them to a growing buffer in memory. When a run is finished and the operator hits a ‘5’ on the simple_menu screen, the collected data is saved to a data file named with the timestamp of the start of the run. (There are also debugging files for the other database clients that may be optionally saved. If there are other files, they are also named with the same startup timestamp.)

3.3.8 *veh_snd and veh_rcv*

veh_snd and veh_rcv are the database clients that send and receive BSM messages to and from the ITRI DSRC radio, respectively.

veh_snd reads the current GPS coordinates and UTC time; truck CAN data such as vehicle speed and acceleration, brake switch status and brake pressure; Volvo targeting data such as preceding vehicle distance, velocity, and acceleration; and necessary control parameters such as CC/ACC/CACC status and requested engine retarder torque from the database.   veh_snd then adds the ego truck ID, packages the data into the appropriate fields in the BSM message, and broadcasts the BSM message at 100 ms intervals via the ITRI DSRC radio.

veh_rcv performs the reverse operation of veh_snd. It waits to receive a BSM message, unpacks the message, identifies which truck the message is from, and writes the data to the appropriate database variables. These data are used by long_trk for ACC/CACC control and by the DVI to indicate ego truck position; and preceding and following truck communication status, braking, and Manual/ACC/CACC status.

3.3.9 *dvi_snd and dvi_rcv*

dvi_snd and dvi_rcv are the database clients that send and receive UDP messages to and from PATH_App_Berkeley, the Qt application running on the Galaxy tablet that is the DVI, respectively.

dvi_rcv receives UDP messages sent by PATH_App_Berkeley, parses them into the appropriate database variables, and writes the data to the database. The messages sent include control requests that are sent by touchscreen presses by the truck driver. They are: increase time gap, decrease time gap, enable CACC control, and enable ACC control. A button press on the touchscreen is indicated to the truck driver by a momentary enlargement of an arrow or a "glow" enhancement to the ACC or CACC control button. However, *actually* changing the control status of the truck depends on long_trk, the control algorithm. It is possible, for instance, that CACC control cannot be enabled because there is no communication with the preceding truck. If a driver attempts to enable CACC under that condition, the CACC button will glow but the display of the ego truck and will remain gray, indicating ACC control. The actual status of the control system is contained in messages sent by dvi_snd.

dvi_snd reads control status database variables, packages them into UDP messages, and sends them to the DVI via a dlink wireless router. The DVI display status is controlled by several inputs: the control algorithm sets the actual time gap and ACC/CACC status; braking status is read directly from the truck CAN bus; target availability is read directly from the Volvo target acquisition system CAN bus; communication status with other trucks is calculated within dvi_snd using the time since last message reception from a given truck.

3.4 Low-level Control

Low-level control comprises the software drivers that communicate directly with hardware. Except for can_man (described above), they are standard QNX drivers used for ethernet and serial devices.

3.4.1 io-pkt-v4-hc

io-pkt-v4-hc is the ethernet driver for QNX 6. It is used for communication with the LAN ports on the wireless router (DVI communication) and DSRC radio (inter-truck communication).

3.4.2 devc-ser8250

devc-ser8250 is the serial port driver for QNX 6. It is used for the 5 Hz input from the Garmin GPS.

4 Summary and Conclusion

This section documents the hardware and software development for truck CACC. The hardware components include: PC-104 computer, DSRC communication, interface with J-1939 Bus, GPS, driver-vehicle interface (DVI), power supply, GPS, emergency switch, network switch, WiFi router. Accordingly, software has been developed to link all those components with the PC-104 computer which runs long_trk for CACC control. PATH team intended to control the Volvo truck as it is without any additional actuators. The PC-104 computer runs QNX 6 as the Real-Time Operating System (RTOS) which is small kernel high efficient able to run with very high frequency with reliability. PATH have been using QNX over 25 years. Therefore, it is highly recommended to use QNX as the RTOS for any safety critical control system. The software structure PATH has developed over the past thirty years has also been proved to be reliable and versatile in the sense that it can accommodate multiple interface with any process scheduling problem. One lesson was that the DVI tablet would be better linked with PC-104 with cable instead of WIFI for convenience for software development and better reliability. We originally selected WiFi as the connection was mainly for demo purpose – the guest on in the truck could hold the second DVI in hand for better view.