Distributed control for unmanned vehicles

Jeffrey N. Callen
RedZone Robotics, Inc.

Remotely operated or computer-controlled vehicles have been developed for dealing with hazardous waste or other materials; for military, police, and fire-fighting operations; for undersea operations; and for automated highway driving. With improvements in sensors and computing capability, the trend is to put more and more control on board the vehicle.

All robotic vehicles must turn intent into action. This is the function of the onboard controller, which translates higher-level vehicle commands into vehicle motion or activation of the vehicle's equipment. As research into and use of robotic vehicles increases, so does the need for the efficient development and application of controller technology.

Toward that end, RedZone Robotics has developed a prototype *Distributed Control System*. The DCS provides cost-effective, flexible control and monitoring of a variety of robotic vehicle functions.

The DCS

RedZone's experience with centralized control systems indicated that they were expensive and had limited usability (see the sidebar, "A brief controller history"). Also, a feasibility study of single-board controller implementations identified the distributed approach as optimal for vehicle control. This led us to develop the DCS. The DCS consists of a network of small, intelligent nodes mounted throughout the vehicle. The nodes interface with and control a variety of device types. The nodes communicate via a *controller area network* (CAN) bus (basically a serial LAN), with a master unit to oversee the network operation. Remote control, autonomous navigation, or other higher-level functions can be layered on top of the DCS through an external gateway.

Why distributed control?

Such a distributed vehicle-control architecture has six main advantages:

Optimization

The local node cards are small by design and mounted next to the vehicle sensors and actuators. There is no large control enclosure with a multiple slotted card cage and power conditioning. The bulk of the controller is distributed throughout the vehicle, allowing maximum use of available space. Power consumption and dissipation are also distributed among the nodes, particularly those that drive high-current applications such as actuator motors.

Robustness

Distributing intelligence throughout the vehicle enhances the vehicle's fail-safe operation. Each node that controls a vehicle actuator can have specific operational instructions in its software to accommodate a variety of failure modes. If the vehicle loses communications or control from the directing authority or the outside world, the nodes can bring the vehicle to an orderly, safe stop on their own. Likewise, periodic diagnostics to monitor the health of the sensors or actuators can be run more often, at the local level. The nodes can report hard or impending failures and can implement alternate control schemes to effect fault-tolerant or "limp home" operation, depending on the nature of the failure.

Simplified maintenance

Every node has identical hardware, allowing for easy substitution of failed nodes. Software is configurable over the CAN to initialize replacement nodes to their specific application.

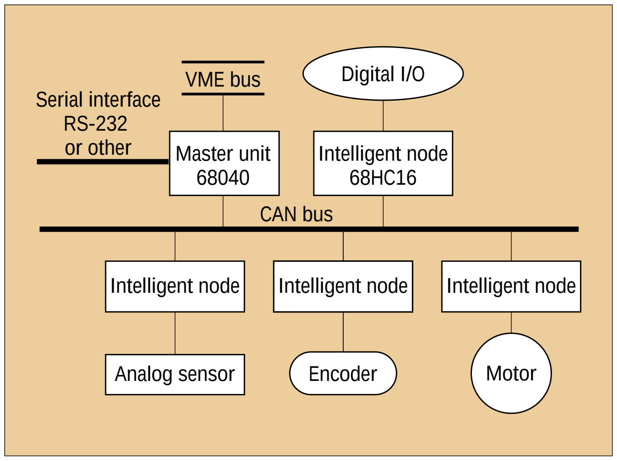

Figure 1. A system block diagram. The DCS consists of a multidrop *controller area network* bus that serially links a master unit with several intelligent nodes. (Multidrop means that several nodes can be placed in parallel anywhere on the network cable.) The CAN bus allows nodes to be added to the network at any location, supports transmission speeds up to 1 Mbit per second, and is becoming an automotive industry standard for control systems in vehicles.

Standardization
The building-block approach allows many components, such as the master unit and the CAN hardware and protocols, to be off-the-shelf standard items. The node hardware is custom but follows industry-standard guidelines. Interface and control code for the node hardware is built into the controller and as such is transparent to users. They can treat the entire control system as a single device.

Economy
Multiple, identical nodes enable an economy of scale not present in centralized custom controllers. For example, reducing the lengths of multiconductor cabling throughout the vehicle reduces the wiring and installation costs significantly (see the sidebar, "Distributed control reduces wiring costs"). The distributed controller requires only that the power wiring and the CAN wiring extend throughout the vehicle. Very short cables connect the nodes to the actuators or sensors.

Comprehensiveness
The number of functions available on the vehicle is limited only by the number of nodes, which in turn is limited by the capabilities of the CAN and onboard power. A node can interface with multiple sensors and control an axis of motion at the same time.

Hardware design

Figure 1 shows a schematic of the DCS network. An external controller addresses the DCS through either the RS-232 serial or the VME parallel interface on the master unit. The external controller can be another computer that has navigation and mission responsibility, or an electronic link to a human operator controlling the vehicle via telepresence. Controlling software can reside on the master unit or, depending on the application, on a more powerful computer interfaced with the DCS.

The intelligent nodes typically receive analog sensor, digital switch or contact, encoder, and sensor or standalone serial device inputs. Nodes typically output

A brief controller history

When RedZone started developing the Distributed Control System in late 1994, most control systems for remotely controlled vehicles and unmanned ground vehicles employed a centralized design that interfaced all vehicle-sensing and servo-control systems individually to a central computer. These computers consisted of multiple CPUs in standard equipment racks in a large central enclosure. They were designed to read sensors and close control loops on several actuators simultaneously and in real time. Cards in the same racks handled the I/O interface, meaning that each sensor and actuator on the vehicle was wired back to the central computer.

RedZone Robotics developed some controllers of this type (the University of Massachusetts at Amherst research testbed, *http://vis-www.cs.umass.edu/~bdraper/ugv.html*, and the internal Non-Line-of-Sight Leader/ Follower, sponsored by the US Army, *http://www.redzone.com*). Other centralized-control implementations included

- Lockheed Martin Astronautics' four Surrogate Semiautonomous Vehicles, developed as part of the Unmanned Ground Vehicle Demo II program, sponsored by DARPA and the Office of the Secretary of Defense (*http://www.cis.saic.com/previous/demoII/index.html*);
- the National Institute for Standards and Technology's Mustang vehicle, built to demonstrate teleoperation for military applications; and
- Carnegie Mellon University's NavLab platforms, which demonstrated early autonomous vehicle operation (*http://www.ri.cmu.edu/ri-home/projects.html*).

Because projects such as these used a centralized-control approach, each new application required a complete redevelopment of the controller technology. A dedicated low-level technology to control motions, monitor health, and integrate the control system with position sensors was not available. This approach carried high development, installation, and maintenance costs, and resulted in a product not easily adaptable to other platforms or tasks.

motor-driver signals to control different actuators, relay drivers, and serial devices. One node can control multiple devices, depending on their location and the nature of their interfaces. An onboard power amplifier provides motor-drive capability for most vehicle-actuator systems. The nodes have identical hardware (see Figure 2).

The master unit (see Figure 3) oversees external communications, monitors system health, and passes commands from the external controller to the appropriate nodes. It also coordinates and controls higher-level vehicle functions that require the action of multiple nodes, and directs the configuration of the software in the nodes.

Software configuration

Although the intelligent nodes contain identical software, they can control different types of actuators and interface with different I/O devices. The user accomplishes this by writing generic software that uses database records that are configured at startup. The database records contain all the information needed to customize a particular input or output.

The nodes share the database records. High-level configuration files stored on the master hard drive define the system inputs and outputs. One file lists all the inputs in the system, and one lists all the outputs. Each device connected to the hardware is referenced by a variable name in the configuration files, and is identified by the node address and the I/O-channel number of the node where the device is connected.

Control- and monitor-configuration files assign actions to the variables. The control-configuration file has a standard format for describing a control loop that links an output with a feedback device. This format uses the variable names defined in the configuration files. The monitor-configuration file allows any input to be tied to any output through a variety of conditional tests.

When the system is powered up, the master unit reads the configuration files from its hard drive. The master software looks at the configuration files and determines the nodes where the variable names will reside. For each variable, the software defines one node as the producer of that information. Any other node that uses that variable is defined as a consumer of that information. The master unit packages this information, along with the control and monitor rules from the configuration files. It then transmits these packets over the CAN to the proper nodes. The nodes read the information and place it in the appropriate databases.

Each node has three databases. The *variable database* contains the name, location, and value of each variable defined in the input- and output-configuration files. The *control database* contains the input and output variables and the motion-control parameters for each control loop. The *monitor database* contains the conditions and constraints to monitor, as well as the actions to take when constraints are violated. Once these databases are set up, the system is ready to operate.

Because the nodes have identical hardware and software, users can easily substitute any node for any other node in the field.

Software operation

The master software runs three tasks simultaneously to process information to and from the controller and to monitor the system's operation. The *main task* executes all the Master User Interface commands, which are a set of standard high-level commands from outside the DCS, such as requests for I/O data or commands for motion. The *heartbeat task* is a background task that runs continuously on the master to detect communication failures. The *message-processing task* continuously reads and processes CAN messages from the network.

Each node's software runs five tasks concurrently at rates of 10 to 100 Hz. The *command task* continuously processes messages while performing overhead housekeeping. The *variable update task* keeps the node-variable database up to date, and sends new values to other nodes over the CAN bus. The *control task* performs closed-loop servo control of an actuator by comparing the current position with the reference position and acting on the position error. The *node I/O task* keeps all variable values up to date by polling the I/O devices. The *monitor task* checks the conditions and constraints, and takes action if necessary. A scheduler determines the task priorities.

High-level commands pass to the master unit through the RS-232 or VME interface. These commands follow a standard format that allows reading node inputs, activating node outputs, or commanding motion from node-driven actuators. The master sends the command or query over the CAN bus to the appropriate node or nodes. The affected nodes reply or carry out the requested action as needed. Commands can be strung together into script files so that they will occur at a single command. When monitor constraints are violated, the system takes action that was preprogrammed by the user into the configuration files, the script files, or both. This action can include everything from notifying the user to sending actuators to certain positions.

High-level programming and standardized components add versatility

Both the master and the node software are written in C. However, this is transparent to the users. To configure the controller for any applicable system and to operate it, users do not need to write, edit, or compile any C code. They program the system by editing the text of the high-level configuration and script files. A standard format for commands lets users access all the I/O in the system, once the configuration files are set up. They need only choose the variable names and actions to be carried out.

This gives users a great deal of flexibility in configuring and operating a system. They can easily add a new device to a node by wiring it to an open I/O channel and editing the text in the configuration files to define the device and what it does. Users can also easily add nodes to the system. The hardware distributes the CAN connections and power leads through junction boxes. Users can add a new node to any open connector on a junction box. They add the node and its devices to the software by editing the configuration files. The entire system can likewise be moved to another application with only installation costs and rewritten configuration files. No custom programming is needed. This feature benefits researchers who need the ability to change and expand a system without the time and cost of a complete custom design.

This type of controller also provides operational benefits. Because the nodes have identical hardware and software, users can easily substitute any node for any other node in the field. Jumpers in the wiring harness determine the node address, so the address resides in the installation and not in the node. Because the DCS configures the nodes over the network at startup, replacing a node is as simple as connecting the new one in place of the old and rebooting the system. Identical hardware reduces the spare parts count, a benefit for military and other users in remote locations where logistics is an issue.

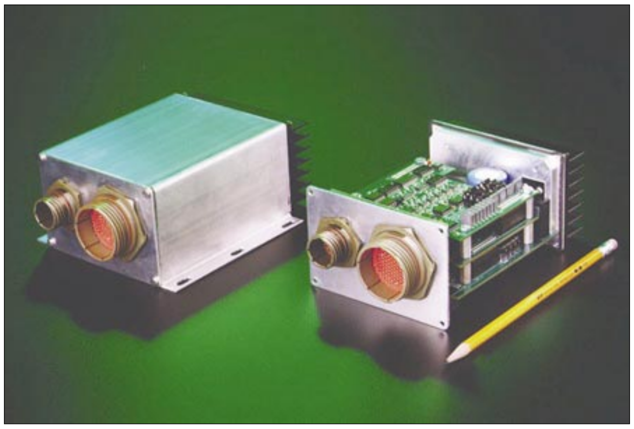
Figure 2. DCS nodes are small and compact and can interface to a variety of devices. Identical hardware simplifies installation and field replacement. (photo by Matt Bulvony, courtesy of RedZone Robotics)

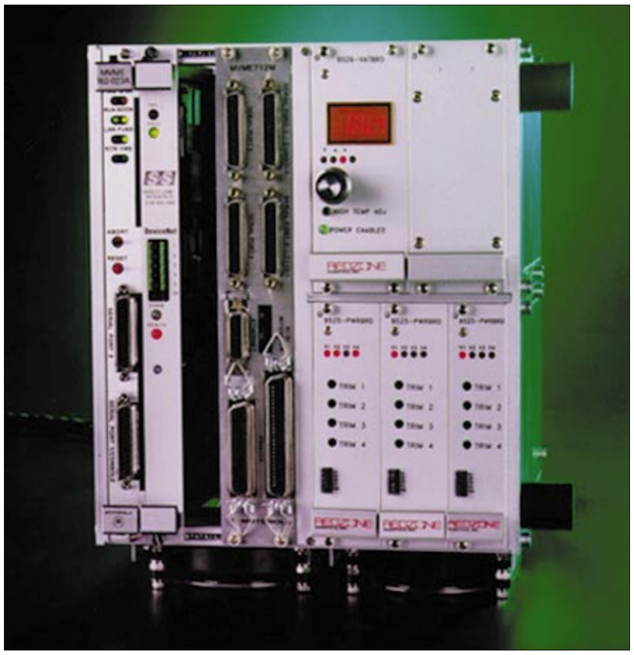
Figure 3. The master unit is a VME cage that contains a 68040 processor card, an interface card to the CAN, and a hard disk to store master programming. The cage also includes power converters for the VME cage and the nodes, and a power-monitoring module. (photo by Matt Bulvony, courtesy of RedZone Robotics)

Testing the DCS

We implemented the DCS on a US Army HMMWV (a Humvee—High Mobility Multipurpose Wheeled Vehicle), to demonstrate full control of the vehicle. The DCS controlled the Humvee's throttle, brake, and steering through electric actuators. For the testing and demonstration, a laptop PC supplied driving commands to the DCS master unit through the RS-232 serial link. The master software interpreted certain keystrokes on the PC keyboard as commands to steer left or right, or move the throttle or brake by preset increments. A researcher in the front passenger seat operated the PC. A safety driver sat in the driver's seat but did not touch the vehicle controls unless there was an emergency.

To date, the vehicle has been driven in this manner for approximately 17 hours and over 50 miles. Because of extensive bench testing before installation, there were surprisingly few integration issues. The majority of changes resulting from the testing were to the laptop PC user interface—primarily selecting the proper keystrokes and actuator increments. Most of the rest of the testing was to build operator confidence in the nonintuitive driving style. Maximum speed for the vehicle was 33 mph, limited by the constraints of the testing sites rather than by the DCS.

The vehicle had no trouble maneuvering on public roads with varying levels of traffic, although in these circumstances, the safety driver was especially vigilant. Because of safety concerns, we did not take the vehicle onto high-speed roads or out with heavy traffic. We demonstrated the safety, monitoring, and maintenance features of the DCS by simulating failures and showing recovery by the intelligent nodes, and by swapping nodes in the field. The project's scope limited the extent of the testing we could do, but the results were very positive.

Where intelligent vehicles are headed

While this project successfully developed and demonstrated a controller for remotely operated vehicles, the resultant product will work on a wide range of applications. Because the programming is at a high level, this controller is appropriate for intelligent vehicles of all sorts, industrial processes or factory automation, commercial building automation, and motion-control applications, such as aircraft simulators (Stewart platforms). Any application that has remote sensing and control points would benefit from distributed control. However, hardware cost would be a significant issue for commercial applications.

Other intelligent-vehicle projects have demonstrated the benefits of distributed control. Recent military projects include

- Omnitech Robotics' Standard Teleoperation System (*http://www.omnitech.com/sts.html*),
- Robotics Systems Technology's Mobile Detection Assessment Response System-Exterior (MDARS-E) program (*http://www.nosc.mil/robots/land/mdars/mdars.html*), and
- a controller developed by SAIC (Science Applications Int'l Corp.) for the Unmanned Ground Vehicles/Systems Joint Program Office, sponsored by the United States Army and Marine Corps (*http://www.redstone.army.mil/ugvsjpo*).

The next generation of intelligent vehicles will likely use distributed control exclusively.

Acknowledgments
This work was funded under DARPA's Small Business Innovation Research program and was sponsored by the Robotics Office of the US Army Tank-Automotive Research, Development, and Engineering Center (Tardec) in Warren, Michigan. For more about RedZone Robotics, access http:// www.redzone.com. RedZone products, including the DCS, are listed in the *Aviation Week & Space Technology—Assoc. for Unmanned Vehicle Systems Int'l 1997–98 Int'l Guide to Unmanned Vehicles* (McGraw-Hill, 1997).

Jeffrey Callen is an electrical engineer and the program manager for intelligent vehicles at RedZone Robotics. He currently is managing the development of an automated transport system to move large, sophisticated modular optics units for the National Ignition Facility laser-fusion project being built at Lawrence Livermore National Laboratory. He previously managed RedZone's Non-Line-of-Site Leader/Follower project, which demonstrated convoying of a human-driven lead vehicle and a computer-driven follower vehicle for the US Army TACOM (Tank-automotive and Armaments Command). He has a BS from Carnegie Mellon University and an MS from the University of Pittsburgh, both in electrical engineering. He is a member of the IEEE and the Association for Unmanned Vehicle Systems International. Contact him at RedZone Robotics, Inc., 2425 Liberty Ave., Pittsburgh, PA 15222-4639; jcallen@redzone.com.

Distributed control reduces wiring costs

Lockheed Martin Astronautics studied comparative wiring costs for their Surrogate Semiautonomous Vehicles, which were based on a US Army HMMWV (High-Mobility Multipurpose Wheeled Vehicle—a Humvee). The study compared the wiring costs for centralized control with an estimate of wiring costs for a distributed controller design. The results (see Table A) show important savings in total wire weight, total connections, and labor to fabricate and install wire harnesses into the vehicle. An 85% savings of installation labor costs is significant when constructing multiple vehicles.

Table A. Comparison of wiring for centralized and distributed control of the Lockheed Martin Surrogate Semiautonomous Vehicle (presented by Roy Greunke, Lockheed Martin Astronautics, at the DARPA Unmanned Ground Vehicle/Demo II Workshop, Feb. 22, 1995, Killeen, Texas).

	Centralized control	Distributed control (estimated)	Savings (%)
No. of Cables	33	14	57.6
Cable length (ft.)	117	54.5	53.4
Total wire weight (lb.)	30.56	7.76	74.6
No. of connections	1,004	536	46.6
Fabrication time (hrs.)	98.3	44.6	54.6
Installation time (hrs.)	88	13	85.2