Using a CAN Bus for Control of an All-Terrain Vehicle

Joshua R. Henderson*, James M. Conrad*, Craig Pavlich†
*Electrical and Computer Engineering Department, University of North Carolina at Charlotte, NC, USA
†Argon National Laboratory, Lemont, IL, USA
{jhender7, jmconrad}@uncc.edu, cpavlich@anl.gov

***Abstract*—When designing an autonomous All-terrain Vehicle (ATV) one must ensure that the communications paths to all major subsystems use a robust electrical signaling protocol. One such protocol is the CAN bus. This paper describes an effort to utilize a CAN bus to connect existing and new hardware on an ATV. The work involved a redesign of the braking control system as well as the throttle controls. The redesign was necessary to replace obsolete or broken components, reduce the amount of extra wiring on board the ATV, and create a robust infrastructure for future expansion. A CAN bus control node was installed for each major component on the ATV as well as a master control node. The master control node now reads in signals from a remote control receiver and interprets them into commands for the other nodes on the CAN bus. Each node takes the commands received from the CAN bus and drives their respective components. A control node was developed to control the throttle and braking system. A second node was developed to control the steering control system. New control systems were developed for driving the brakes and the steering. The ATV can now be driven using a RC controller to operate the brakes, steering, and throttle, and is poised to accept new hardware to make it truly autonomous.**

***Index Terms*—CAN Bus, ATV, Remote Control, RX63N, GR-Sakura**

I. Introduction

The end goal of this work is to create a fully autonomous ATV. Before making the ATV autonomous, a remote control system needed to be developed. The ATV originally worked by remote control, but many of the original components had been salvaged for other projects. The original microcontroller had to be replaced and all of the wiring had to be redesigned. A CAN bus was also installed to reduce the amount of wiring on board the ATV. The CAN bus also makes for easier addition and removal of new components to the system. The steering system had also been redesigned and needed to be integrated into the new control system. The initial wiring is shown in Figure 2. The control system was broken down into multiple sections on the CAN bus to make the system more modular. Each component was assigned to one of three microcontrollers which interpret commands to and from the CAN bus. The control system was seperated into a master control node, a steering control node, as well as a steering and throttle control node.

CAN bus is mostly used in the automotive industry and was developed specifically with cars in mind. CAN bus was designed to reduce the amount of cable running between all of the sensors and controls inside of a vehicle. Autonomous vehicles typically require a large number of sensors for navigation, which makes CAN well suited to this application.

Fig. 1. Zapatabot ATV [5]

II. CAN Bus

A. CAN Bus

Since one of the future goals of this work is to add sensors to make the ATV fully autonomous, a Control Area Network (CAN) bus was chosen[4]. A CAN bus allows for the easy addition and removal of components to the system without disturbing the rest of the bus. A CAN bus requires only two wires and a ground for communication between all nodes on the bus. This greatly reduces the amount of cabling running throughout the ATV. When a new node needs to be added to the bus, it simply needs to be connected to the two wires already running throughout the ATV. This simplifies addition and removal of nodes over the initial setup. A master control node was developed which interprets the RC controller signals into CAN messages and then transmits them to the other nodes. Another node was also made for controlling the brakes and the throttle. This node reads messages from the CAN bus and interprets them into directions for the brakes and throttle. The steering control node works the same way by interpreting CAN messages into commands to the steering system. A diagram of the overall plan for the CAN bus is

978-1-4799-6585-4/14/$31.00 ©2014 IEEE

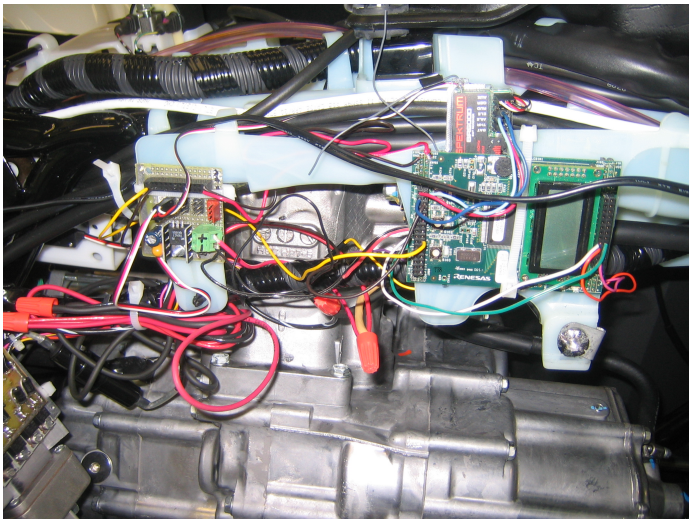

Fig. 2. Initial ATV Wiring [5]

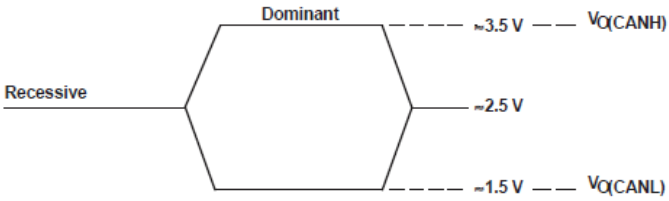

Fig. 3. CAN Transmission[1]

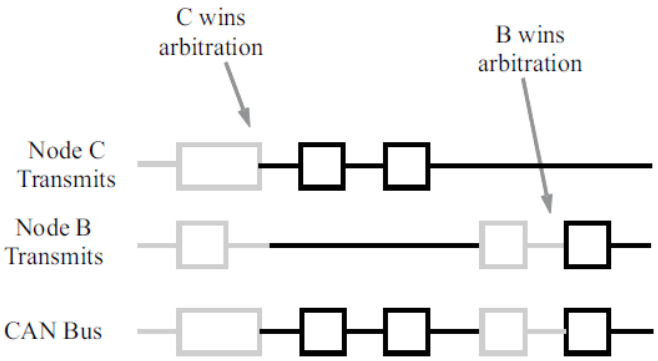

Fig. 4. CAN Arbitration[1]

shown in Figure 5.

CAN is an asynchronous TDM protocol. It has a relatively slow data rate, but makes up for it with very high reliability. CAN messages can be heard by all nodes on the bus. This allows for easy debugging of multiple components and reduces the amount of wiring needed to connect each of the nodes for communication. CAN operates using a three wire system. The wires are CAN High, CAN Low, and Ground. Typically CAN devices operate at 5V, but 3.3V devices are also used. In order to communicate via CAN, a CAN module and a CAN transceiver are required. The CAN module is typically embedded into the microcontroller, while the transceiver is an external component. The CAN module is in charge of reading and sending signals to the transceiver and placing valid messages into their respective mailboxes. The CAN transceiver is responsible for converting bits received into a signal across the CAN High and CAN Low wires.

When a 1 is being sent across the bus, both lines are set to 2.5V. When a 0 is sent across the bus, the CAN High line goes to 3.5V and the CAN Low line goes to 1.5V. The transceivers on the bus then use the voltage difference across the line to determine whether a 1 or 0 is being sent. This is shown in Figure 3. CAN is unique in the way that it handles bus arbitration. When two nodes attempt to transmit at the same time, zeros will overwrite ones. When a node sees that one of its bits has been overwritten by another node, it will back off and wait until the message is finished to begin transmitting again. Since the ID of a message is broadcast first, the first few bits are often used to indicate the priority of the message. Since zeros overwrite ones, the ID with the lowest value always has a higher priority. An example is shown in Figure 4.

The SAE J1939 Standard was used in setting up the steering control node. Since the SAE J1939 standard is common and was already implemented in the steering node, this protocol was used for the rest of the system as well. The SAE J1939 standard runs at 250kbps. The node for the steering system was developed using a PIC Microchip microcontroller[6]. The steering control node interprets the first two data bytes into a steering angle for the wheels and then tricks the steering assist into turning the desired direction.

In current network the extended ID mode is used which uses 29 bit identifiers instead of the standard 11 bits. In CAN, each message is given an ID. This ID used to indicate the message type instead of the destination. Each device will have multiple mailboxes with different IDs in order to read in multiple types of messages. Mailboxes are also given a mask so that they can ignore certain bits such as the priority bits. This also means a mailbox could be set up to ignore all of the bits in an ID and receive all messages sent on the bus. In the current network, the first 3 bits are used to indicate priority. The last 8 bits are used to indicate the source address of the message. The source address mostly goes unused for now, but is useful for debugging and error messages. The rest of the bits are used for the ID and are the only bits not masked out. Each type of message will have its own unique idea. For now, the only types of commands used are steering, acceleration, braking, and error messages.

A Renesas RX63N microcontroller was used as the master node for giving the rest of the system commands. The master node is also responsible for interpreting signals from the remote control. The RX63N development board was chosen because it already has both a CAN module as well as a CAN transceiver on board. The board also has CAN APIs available, which simplified programming of the CAN bus. The master node is responsible for most of the messages on the CAN bus. The other two nodes only transmit startup complete and

978-1-4799-6585-4/14/$31.00 ©2014 IEEE

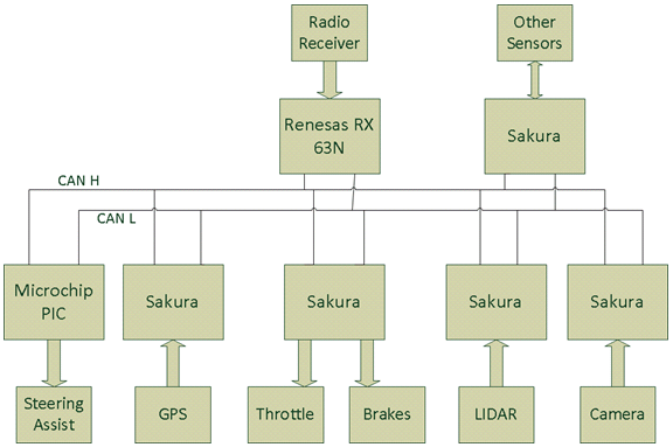

Fig. 5. CAN Design Plan

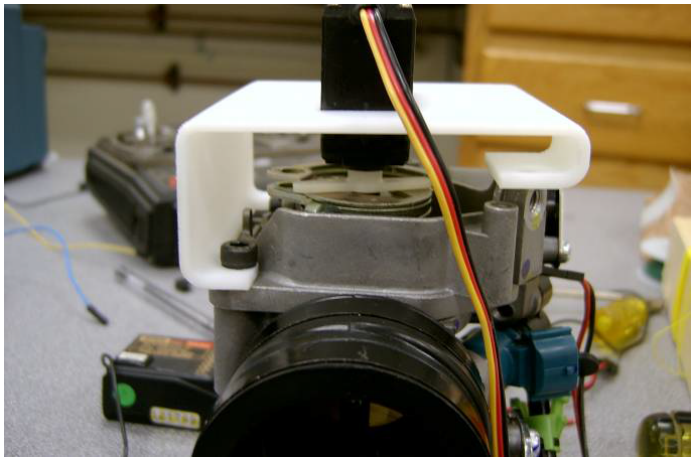

Fig. 6. Throttle Servo Mount[5]

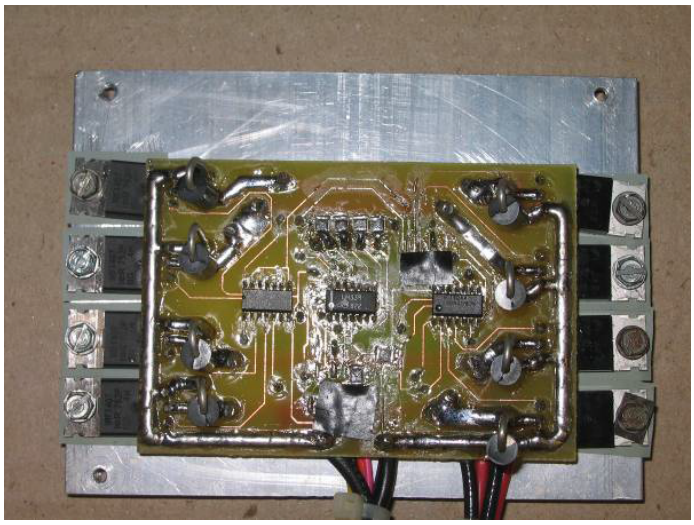

Fig. 7. Braking H-Bridge[5]

error messages. This will change in the future, after sensors are added to the ATV.

The largest disadvantage to using a CAN bus is that each component now needs an individual microcontroller to control it. The microcontroller is mainly used to interpret the CAN messages into control signals for the devices it is responsible for. The steering and throttle were originally going to be treated as two seperate nodes. They were combined into one node since neither device has complex controls and neither device will be removed from the system since throttle and braking are required for the ATV to drive. The brakes and throttle are both controlled by a Renesas GR-Sakura microcontroller. Since the Sakura board uses an RX63N microcontroller, it already has a CAN module built in. For the transceiver, a Texas Instruments SN65HVD251 Industrial CAN Transceiver was used. Even though the Sakura uses an RX63N, the CAN APIs could not be used, because the APIs were designed specifically with the RX63N development board in mind.

III. Other Components

A. Throttle and Brakes

As stated previously, both the throttle and the brakes were controlled using a Sakura microcontroller. The throttle control on the Honda ATV was developed by a previous group and worked well so no modifications were made other than switching control to the Sakura[**?**]. The throttle is controlled by a standard Parallax servo attached to the throttle via a rapid prototyped frame. Since the ATV has a spring return on the throttle, it will automatically shut off the throttle when the servo returns to the idle state. The servo and frame are shown attached to the throttle in Figure 6

The ATV has a hand brake as well as a foot brake. The hand brake is left alone so that the ATV may still be driven by a person without the electronics. A previous group created a frame and attached a linear actuator to the foot brake. The linear actuator was controlled by a custom built H-bridge. The linear actuator and the H-bridge can both be seen in Figures 8 and 7 respectively. After testing the braking system, a short was found in the H-bridge. Since the H-bridge was large and also had problems with overheating in the past, it was replaced with a Pololu motor controller. The new motor controller was chosen because it has much higher voltage and current tolerances. The motor controller also does not require a heat sync unless using more than 15A, whereas the H-bridge required a large heat sync just to operate at 8A. The Pololu motor controller is also much smaller, allowing it to fit in the new enclosure easily. The original H-bridge is shown in Figure 7.

B. Steering

The first steering system had severe overheating problems[2]. One of the key components of the Electric Power Steering system's operation is the torque sensor. When a difference in the angle of the wheels and the operator's handle bar is detected, a change in resistance is output by the sensor. The sensor has two separate resistive components. One indicates angles to the right, while the other indicates

978-1-4799-6585-4/14/$31.00 ©2014 IEEE

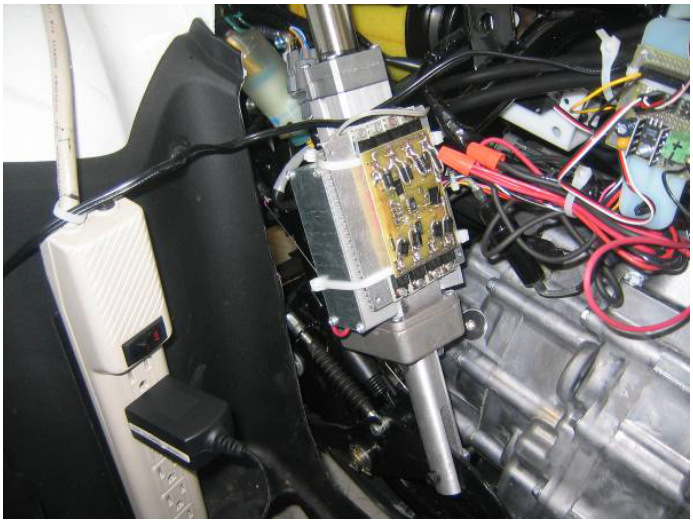

Fig. 8. Braking Linear Actuator[5]

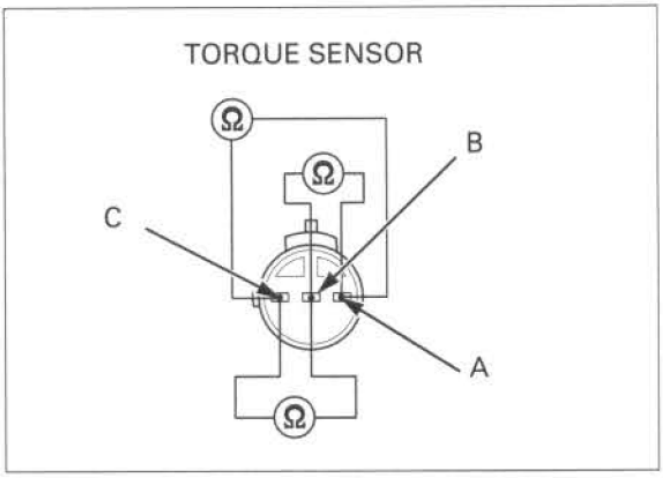

Fig. 9. Torque Sensor Schematic[6]

angles to the left. The third wire is connected to ground. A picture of this is shown in Figures 9 and 10.

From experiments using a multi-meter, the following torque sensor system function was discovered[6]:

- Pin A: W/Bu wire = Right Turn. Varies from 13W at neutral (no steering effort) to 9.5W at maximum steering effort (meter connected A to C).
- Pin B: Y/R wire =: Left Turn. Same specification as Pin A (connected B to C).
- Pin C: G/O wire = Ground. Continuity with Chassis.

A circuit of parallel resistors was developed with switches to add and remove resistors. PWM is used to emulate intermediate resistances. Since very low resistances were being used, Maxim ICs MAX4668 Dual SPST CMOS Analog Switch was selected to switch resistors on and off. This switch has a low on resistance of 2.5Ω, making it ideal for this application. A US Digital MAE3 Digital Magnetic Encoder was attached to the steering column to determine the current steering angle. The encoder is shown in Figure 11. The encoder is used for feedback in the closed loop control system.

The new steering system only has overheating problems when attempting to steer while completely stationary. While stationary, the steering system is over damped, causing it to steer slowly and overheat. While moving, the over damping problems should be reduced since the motor will have less resistance from friction. During the steering redesign, the steering controller was also turned into a CAN bus node. This allowed easy integration of the steering controller into the new system. The steering control node is shown in Figures 12 and 13.

C. Remote Control

A Spektrum DX6 transmitter and a BR6000 receiver were used to control the ATV. Theses are typically used for controlling model airplanes. Since this controller is normally used for yaw, pitch, and roll, most of the channels were not

Fig. 10. Torque Sensor Pluge[6]

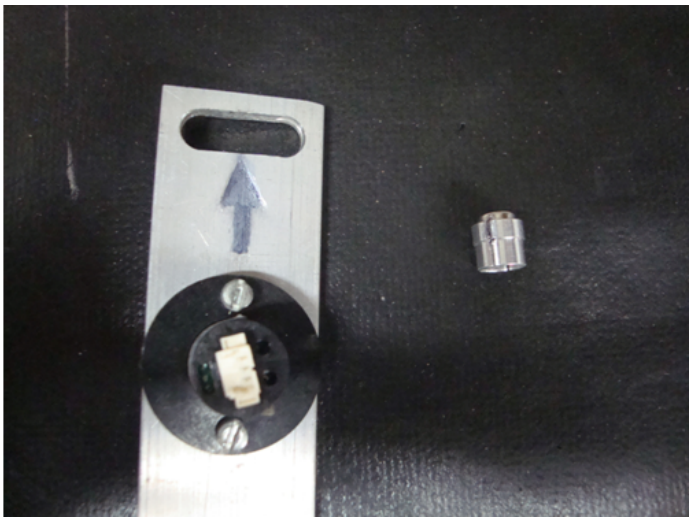

Fig. 11. Steering Angle Encoder[6]

978-1-4799-6585-4/14/$31.00 ©2014 IEEE

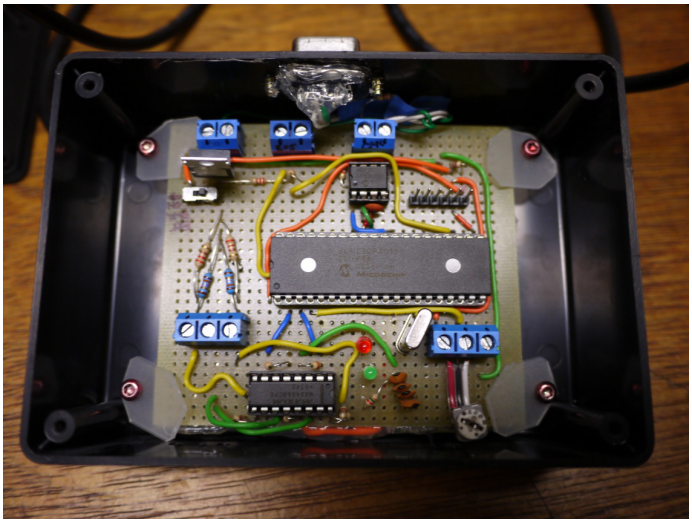

Fig. 12. Steering Control Node[6]

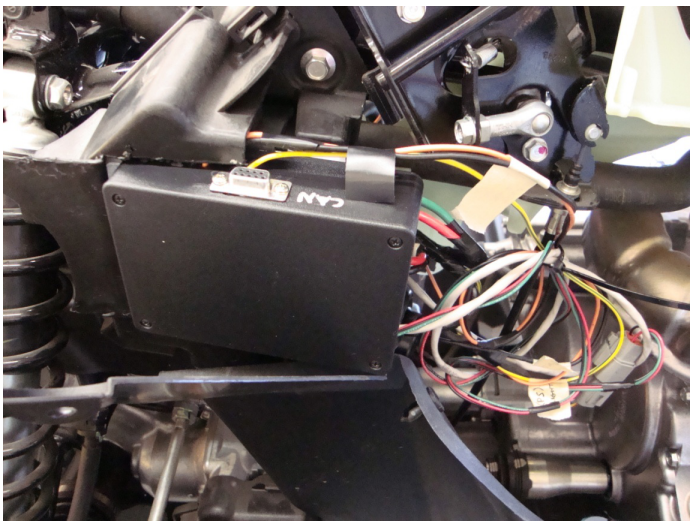

Fig. 13. Installed Steering Control Node[6]

needed. Only one of the joysticks on the controller. Forward and backwards were interpreted into acceleration or braking respectively, since the ATV cannot currently go into reverse on its own. The RX63N microcontroller read the pulses coming from the receiver and interpreted them into commands for each of the nodes on the bus.

IV. Conclusion

The ATV is now fully operational using an RC transmitter. A CAN bus network has been successfully installed on board the ATV. A new control node has been developed for the steering control system. Another node was developed for controlling the throttle and brakes. The remote control functionality as well as the new steering control node have been reintegrated into the system. Also, a new master control node has been completed which takes the radio receiver's signals and interprets them into commands across the CAN

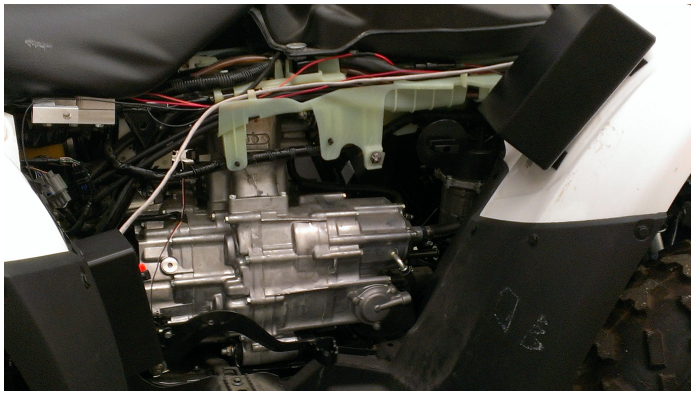

Fig. 14. Final ATV Wiring

bus. The steering, throttle, and brakes can all be controlled across the CAN bus now. All of the wiring has been placed into enclosures with their respective nodes. The two new nodes are shown in Figure 14. The master control node is in the top right enclosure and the braking and throttle node is in the bottom left.

V. Future Work

A kill switch still needs to be added to the system so that the ATV can be shut off in case of an emergency. The kill switch can also act as a simple on off switch. Currently the wiring is disconnected whenever testing is completed and a switch would simplify this process as well as make it safer. The actuator needs some sort of feedback system to the Sakura. Currently, the Sakura works with the brake by assuming it is in the raised position at startup. Some possible options include adding an LVDT or potentiometer to the actuator. In the future sensors will be added to the ATV and integrated into previous work on path planning algorithms to make it fully autonomous[3].

References

[1] S. Corrigan. *Introduction to the Control Area Network (CAN)*, 2008.

[2] A. Cortner, J.M. Conrad, and N.A. BouSaba. Autonomous all-terrain vehicle steering. In *Southeastcon, 2012 Proceedings of IEEE*, pages 1–5, 2012.

[3] S. Ghangrekar and J.M. Conrad. Modeling and simulating a path planning and obstacle avoidance algorithm for an autonomous robotic vehicle. In *Modeling, Analysis Simulation of Computer and Telecommunication Systems, 2009. MASCOTS '09. IEEE International Symposium on*, pages 1–3, 2009.

[4] S.K. Gurram and J.M. Conrad. Implementation of CAN bus in an autonomous all-terrain vehicle. In *Southeastcon, 2011 Proceedings of IEEE*, pages 250–254, 2011.

[5] R.A. McKinney, M.J. Zapata, J.M. Conrad, T.W. Meiswinkel, and S. Ahuja. Components of an autonomous all-terrain vehicle. In *IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the*, pages 416–419, 2010.

[6] C. Pavlich. Design of a CAN-enabled steering control node for an autonomous ATV. Technical report, UNC Charlotte, 2012.

978-1-4799-6585-4/14/$31.00 ©2014 IEEE