SENIOR DESIGN 2

MDAS.ai Drive-By-Wire Using V2X for Enhanced Safety

LOW LEVEL DESIGN SPECIFICATION

VERSION 2 / 8 FEBRUARY 2019

WINTER 2019

Team #7 - Classified:
Athanasios Argyris, Kayleigh James, Sarah Overbeck, Joshua Quejadas

Contact Information: aargyris@umich.edu, jamesmk@umich.edu, soverbec@umich.edu, jquejada@umich.edu
Advisor: Dr. Mike Putty

Table of Contents

1. INTRODUCTION

1.1 Purpose

This document define the low level design modules for the MDAS.ai Drive-by-Wire system using V2X for Enhanced Safety. This document will expand on all hardware and software modules of our design in detail and outline, again, their traceability to system requirements.

For this project, we will be designing a system that will convert a traditionally operated vehicle into a Drive-By-Wire (DbW) vehicle. We will be taking three main functions usually operated by a driver and computerizing the control of each as well as the communication between them. In addition, we will be incorporating Dedicated Short-Range Communication (DSRC) as a form of Vehicle-to-Everything (V2X) communication for further safety of the system as a whole. This V2X communication will notify the vehicle of an external event, such as, a pedestrian in a crosswalk. Then, the vehicle will come to a safe stop until the crosswalk has been cleared, or the pedestrian in the crosswalk message has stop being sent by the DSRC radio. The ***motivations for this project*** is to create a reliable and safe DbW system and enhancing the aforementioned safety by incorporating V2X communication. An ***overall benefit*** is learning how both these technologies work and to show that autonomous vehicles could be enhanced with this outside information to make better decisions in various environments and conditions. This vehicle will be a ***working prototype*** that the MDAS.ai group can take and turn into an autonomous vehicle that will be driven around the University of Michigan - Dearborn campus.

Image 1: Pedestrians in crosswalk; example for DSRC external event

Image 2: MDAS.ai Vehicle in IAVS High Bay

1.2 Definitions

MDAS.ai: Michigan Dearborn Autonomous Shuttle
DSRC: Dedicated Short Range Communication
V2X: Vehicle to X (Infrastructure, Vehicle, etc.)
CAN: Controller Area Network
DbW: Drive-by-Wire
Drive PX2 : The Nvidia Drive PX2 is a computer that is aiming to provide autonomous functionality to MDAS.ai using machine learning.
RSU: Road Side Unit; DSRC module placed outside of the vehicle used to alert the vehicle of a road hazard.
OBU: On Board Unit; DSRC module placed in the vehicle used to receive alerts from the RSU.
RSSI: Received Signal Strength Indicator
GPIO: General Purpose Input Output
E-STOP: A switch, that when pressed, will notify the boards via an interrupt on a GPIO pin. This switch will be an analog device, either providing 0V or 3.3V into the microcontroller.
DAC: Digital-to-Analog Converter; a piece of hardware that converts a digital signal to an analog signal.

1. 3 System Description

MDAS.ai (Michigan Dearborn Autonomous Shuttle) is a research project dedicating to the design, construction, test, and implementation of an autonomous shuttle vehicle. Before that can be accomplished, however, there are several smaller projects that must be completed. For this project, we will be designing a system that will convert a traditionally operated vehicle into a Drive-By-Wire (DbW) vehicle. We will be taking three main functions usually operated by a driver and computerizing the control of each as well as the communication between them. In addition, we will be incorporating Dedicated Short-Range Communication (DSRC) as a form of Vehicle-to-Everything (V2X) communication for further safety of the system as a whole. The primary objectives of this system are to be a reliable and safe DbW system and enhancing the aforementioned safety by incorporating V2X communication. The major goal of using this V2X communication is to provide the vehicle with information about the outside world that could alter the state of the vehicle's operation.

The DbW system will include the following modules: RF, steering, braking and throttle, and joystick. As mentioned before, all modules will be communicating via a Controlled Area Network (CAN). Presently, MDAS.ai only has pieces of two of the modules which have been implemented in an Ad-Hoc fashion: the steering module and the joystick module. These two are currently interfaced over CAN but have no defined message set. This is because they primarily work together for demonstration purposes. Previously, the vehicle utilized a rudimentary throttle module, however, it was never interfaced over CAN and was activated by flipping a toggle switch. The brake and DSRC modules have yet to be developed and will be fully developed by our team. The individual modules that are partially developed have been tested, but never integrated to communicate as an operational system with a defined message set. Our goal is to integrate all of these modules, use CAN as our main communication protocol, build a message set and introduce V2X using DSRC to provide the vehicle with vital information that could alter the way the vehicle will operate.

The RF module will be composed of two parts, the On-Board Unit (OBU) located on the vehicle and the Roadside Unit (RSU) located on a crosswalk, intersection, or other infrastructure. In the event of a traffic incident (e.g. reduced speed ahead or a crash) the RSU will send a message to the OBU in order to alert the vehicle of the incident ahead. The OBU will interpret the RSU's messages in order to generate a CAN message. This CAN message will be sent to the modules that are affected by the incident. For example, if a pedestrian is inside the crosswalk, The RSU should send a message to the OBU indicating this event. The CAN message sent by the OBU should trigger a change in all modules in order to properly respond to the pedestrian. The brake module will bring the vehicle to a stop and the other modules will enter a safety state. Once the OBU stops receiving this message, the safety state will be lifted and the vehicle will resume normal operation.

The next module in our system is the joystick module. The joystick will be capable of controlling the vehicle meaning it will have control over steering, brakes, and throttle modules. We will be utilizing CAN to communicate between the joystick module and all other modules. This device will have a dead man switch in order to ensure safe operation of the vehicle. This module will serve two additional purposes in our system: the primary purpose being a diagnostic tool and our secondary purpose being system verification. Due to the safety concerns that come with working on a vehicle, the joystick will be our primary method for demonstrating the vehicle operating as intended.

The brake module in our system will be controlled via a PI controller (shown on the right). This means it will be receiving a demanded "Setpoint" pressure via CAN. It will be receiving a feedback "Output" pressure via the brake pressure sensor, and apply the correct amount to achieve the setpoint. It will apply the pressure by generating a CAN message and sending it to the brake actuator. For this module, we will have to define a message set for both receiving and sending CAN data.

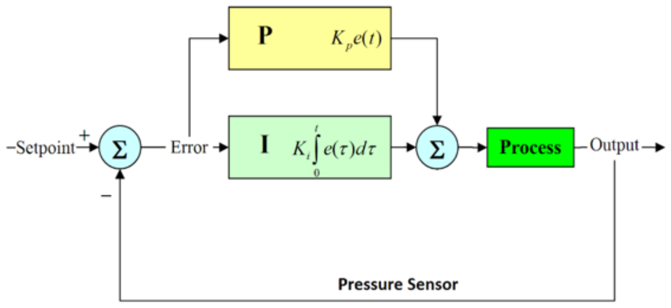


Figure: PI Controller with Pressure Sensor output along the feedback path

The steering module will receive a CAN message and do meaningful scaling. After, it will use this scaling and generate a CAN message that will be sent to the steering actuator. It will also be actively listening for a message from the input torque sensor. If there is an input torque applied during DbW mode, it will safely deactivate DbW and notify the other modules. The other modules will be notified via CAN and enter a safety mode that will not be deactivated until the vehicle's power is cycled. A well-defined message set will be in order, to ensure proper and safe communication for the DbW system.

Figure: E-Stop Button

Another example of our intended functionality is when the DbW modules receive a signal indicating an emergency. The manual emergency stop button, if pressed, will trigger the brake board to stop the vehicle in a quick but safe manner. Simultaneously, the steering and throttle boards will go into safety mode such that they do not accept messages from any other sources. To ensure that the E-stop is not easily disengaged, the button will have to be released and the vehicle power- cycled before DbW can be re-engaged.

As part of our message set and safety features of our DbW system, we will implement a "Keep Alive" check. Meaning, if our modules don't see a message within a set amount of time, they will enter a safety mode and disable DbW mode until communication is re-established. On the system requirements documents, the details and specific numbers will be presented.

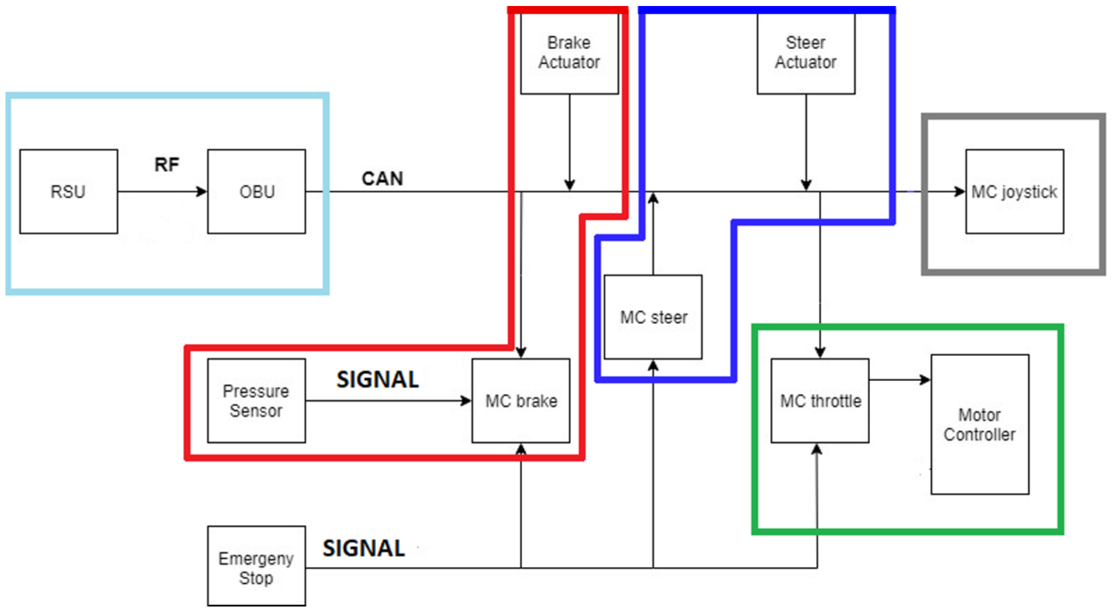


Figure 1: MDAS.ai Drive-by-Wire System Block Diagram
DSRC module (light blue), Brake Module (red), Steering Module (dark blue), Throttle Module (green), and Joystick Module (gray)

1.4. References:

1. Bertoluzzo, M., et al. “Drive-by-Wire Systems for Ground Vehicles.” *2004 IEEE International Symposium on Industrial Electronics*, 2004, doi:10.1109/isie.2004.1571893.

2. Kenney, John B. “Dedicated Short-Range Communications (DSRC) Standards in the United States.” *IEEE*, Proceedings of the IEEE, July 2011, ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5888501.

3. TIVA C Series Microcontroller Data Sheet: http://www.ti.com/lit/ds/symlink/tm4c123gh6pm.pdf

4. CAN ISO Standard 11898: https://www.iso.org/obp/ui/#iso:std:iso:11898:-1:ed-2:v1:en

5. Technical Specification for DSRC: https://www.imda.gov.sg/-/media/imda/files/regulation-licensing-and-consultations/ict-standards/telecommunication-standards/radio-comms/imda-ts-dsrc.pdf?la=en

6. IEEE 1609.2-2016 Spec for DSRC: https://standards.ieee.org/standard/1609_2-2016.html

7. IEEE 802.11-2016 Spec for wireless communication: https://standards.ieee.org/standard/802_11-2016.html

8. I^2C Standard: https://www.nxp.com/docs/en/user-guide/UM10204.pdf

2. CONSTRAINTS

2.1 Environmental Constraints

The system will be developed in a laboratory environment during this project. It will not be exposed to significant temperature changes, vibration, or impact.

2.2 MDAS.ai Vehicle Constraints

The system must be able to run off of the MDAS.ai vehicle's power supply while the vehicle is operating. The system's size and weight shall not degrade the operational characteristics of the shuttle. The system (excluding the RSU and the antennas for the OBU) must be able to fit within the shuttle.

2.3 Reliability and Safety Constraints

The system must maintain safe operation in the event of component failure or communication failure. The software must maintain reliability in the event of software errors.

2.4 MDAS.ai Development Site Constraint

The system will be developed in a laboratory environment (IAVS high bay) as a test bed and the vehicle will be up on blocks so that movement from test area is prevented. This project is also not easily mobile, preventing the ability to work in other labs.

2.5 MDAS.ai Other Project Teams Considerations Constraint

The vehicle is part of a large research project with several other teams working on it at once. It must be considered that at times, components may or may not be worked on while other teams are completing a demonstration or working on the vehicle at the same time. This will constrain time while completing the project on the vehicle.

3. LOW LEVEL SYSTEM DESIGN

3.1 DSRC - [Trace to requirements 3.1]

- 3.1.1 DSRC Radio - Hardware component [Traces to requirements: 3.1.1, 3.1.2, 3.1.3]
  This component was selected because it is part of the DSRC research project and would be very costly to purchase separately.
  - Vendor: Cohda Wireless
  - Cost: $2000 each
  - Operating System: Modified Radio Linux
  - Automotive 12V power supply
  - Built in CAN Transceiver
  - The IEEE 1609.2-2016 (DSRC) communication standard inherited the IEEE 802.11-2016 communication protocol stack, this means that the sending radio will use Carrier-Sense Multiple access with Collision Avoidance (CSMA/CA) via the IEEE 802.11-2016 communication stack.

- 3.1.2 Receiver Radio Software - Software component [Traces to requirement: 3.1.4, 3.1.5]
  - Generate CAN message via received RSU DSRC message.
  - Latency measurement will be done by using previnent software on the DSRC radio that will subtract the receiving radio GPS time from the sending radio GPS time and printed to a terminal via the UDP pre-existing socket interface on port 9001.

- 3.1.3 sender Radio Software - Software component [Traces to requirement: 3.1.6]
  - Send a message via socket interface to OBU
  - Socket Port number chosen: 10000 UDP
  - UDP port selected because of its superior performance via RF.

<u>DSRC Message Sets:</u>
2nd lowest priority since all the module and actuator message sets takes precedence.

<u>DSRC to ALL</u>
Used to send messages from RSU to OBU.

CAN ID	**DATA**							
	Byte 1	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**
0x1BDB0000	SRC ID	EVENT TYPE	DSRC MESSAGE		Res	Res	Res	Res

<u>DSRC hardware & Software diagram:</u>

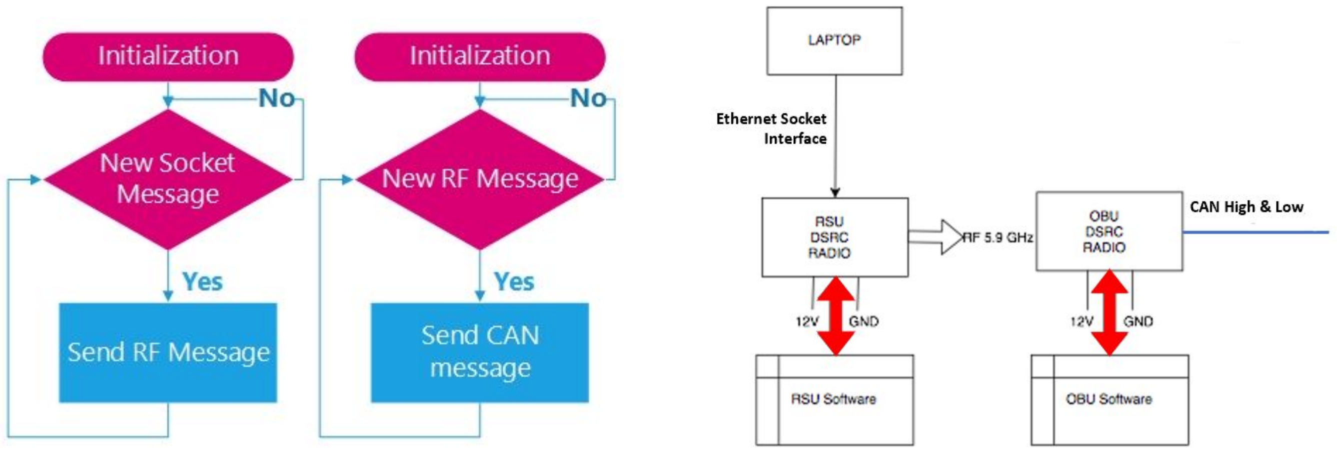


Figure 2: Hardware & Software Diagram for DSRC Module

<u>3.2 Throttle Module - [Trace to requirements 3.2]</u>

- 3.2.1 Tiva Launchpad - Hardware Component [Trace to requirements: ALL 3.2]
  This microcontroller was chosen because our team members were familiar with using this board from previous courses such as ECE 473 and ECE 4951. Additionally, this board meets all of the processing, timing, and power requirements needed to support the DbW system while being relatively inexpensive.
  - Vendor: Texas Instruments
  - Cost: $13.49
  - In-charge of all computation/scaling to control the throttle of the vehicle
  - E-Stop switch analog input
  - Interfaced with DAC's (I^{2}C Interface Protocol)
  - CAN transceiver (CAN Tx & Rx Interface Protocol)
  - Runs C software component to control throttle via CAN & DAC's
  - The Tiva requires an input voltage of 4.75 VDC to 5.25 VDC
  - The Tiva can provide an output voltage of 3.3 VDC (300 mA max) or 5.0 VDC (depends on 3.3 VDC usage, 23 mA to 323 mA) to the DACs

- 3.2.2 Two DAC's - Hardware Components [Trace to requirement 3.2.5, 3.2.1, 3.2.3]
  - Vendor: Microchip
  - Cost: $2.49
  - These DAC's will provide input analog signals to motor controller from the digital output of the Tiva.
  - I^{2}C interface to the microcontroller
  - DAC2 will be ½(DAC1) at all times.
  - DAC1 operation range is 1-4V, DAC2 operation range is 0.5-2V
  - The DACs will have 12 bits of resolution
  - The Vss pins on both DACs will be connected to ground
  - The Vdd pins on both DACs are used to power the device with the output voltage from the Tiva. These DACs can accept anywhere between 3-5V.

- The Vout pin will be an analog output of a voltage between 0 and Vdd.
- The SDA pin will be receiving an digital input high voltage of 0.7Vdd and an input low voltage of 0.3Vdd
- The SCL pin will receive the I2C clock from the Tiva

- 3.2.3 CAN Transceiver - Hardware component [Trace to requirement 3.2.4, 3.2.8]
  We had previous experience integrating this specific CAN transceiver with the Tiva Launchpad. We decided to use it for our project in order to mitigate risk.
  - Vendor: Texas Instruments
  - Cost: $2.07
  - Interfaced to Tiva Microcontroller CAN Controller
  - Signal interface CAN RX/TX
  - Requires an input voltage of 3.3V from the Tiva
  - Voltage at CANH or CANL pins can be between -14V and 14V

- 3.2.4 E-STOP switch - Hardware component [Trace to requirement 3.2.3, 3.2.6]
  We selected this E-Stop switch as its voltage, current, and temperature ratings met our requirements and operating conditions.
  - Vendor: IDEC
  - Cost: $51.90
  - Switch used to notify Microcontroller of an emergency
  - Roughly ~3.3V input into Tiva for normal operation, ~0V for emergency
  - Analog signal
  - Connected to each module's Tiva board.

- 3.2.5 Throttle Position Sensor (TPS) - Hardware component [Trace to requirement 3.2.7]
  This throttle position sensor came preinstalled from the OEM.
  - Vendor: Polaris GEM
  - Cost: $124.99
  - Analog signal output depending on how far the throttle is pressed down
  - Default Configuration of the vehicle would feed this directly into the motor controller.
  - Input into the Tiva for controlling the vehicle's acceleration depending on the mode of operation.

- 3.2.6 Throttle Code - Software component [Trace to requirements: ALL 3.2]
  - C Language code
  - Receive CAN messages, scale and provide throttle accordingly
  - If in DbW mode, the throttle will be controlled via CAN, otherwise via TPS
  - Scale code such that output voltages of DAC's are respective limits (.5-2V & 1-4V)
  - Count CAN messages, if no message is received within 300 ms, kill DbW mode
  - If E-Stop analog signal is received, kill DbW mode
  - Analog-to-Digital conversion of TPS Sensor and sent onto the CAN Bus
  - CAN message set for the throttle code can be seen in the Interface Description later in the next section of this document.

<u>Throttle module Message Sets:</u>

Throttle module will use two message sets. Both brake and steering module uses two message sets hence the throttle message set falls at fifth priority.

<u>Joystick/PX2 to Throttle Module</u>

Joystick/PX2 to Throttle module message set is used to receive y axis data from the joystick.

CAN ID	**DATA**							
	Byte 1	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**
0x1ADB0000	SRC ID	EVENT TYPE	THROTTLE DATA (12 Bit) MSB = 0		Res	Res	Res	Res

<u>Module Feedback</u>

The module feedback message set will be used to send and receive data from the throttle sensor.

CAN ID	**DATA**							
	Byte 1	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**
0x1CDBFFFF	SRC ID	SENSOR FEEDBACK (12 Bit)		RES	RES	RES	RES	RES

Throttle module software flow chart:

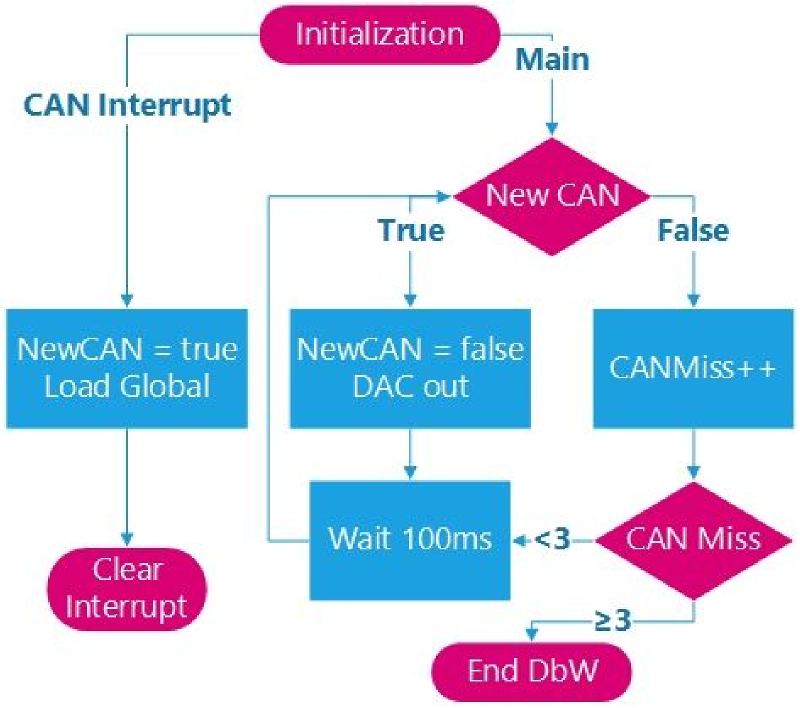


Figure 3: Software Flowchart for Throttle Module.

Throttle module hardware diagram:

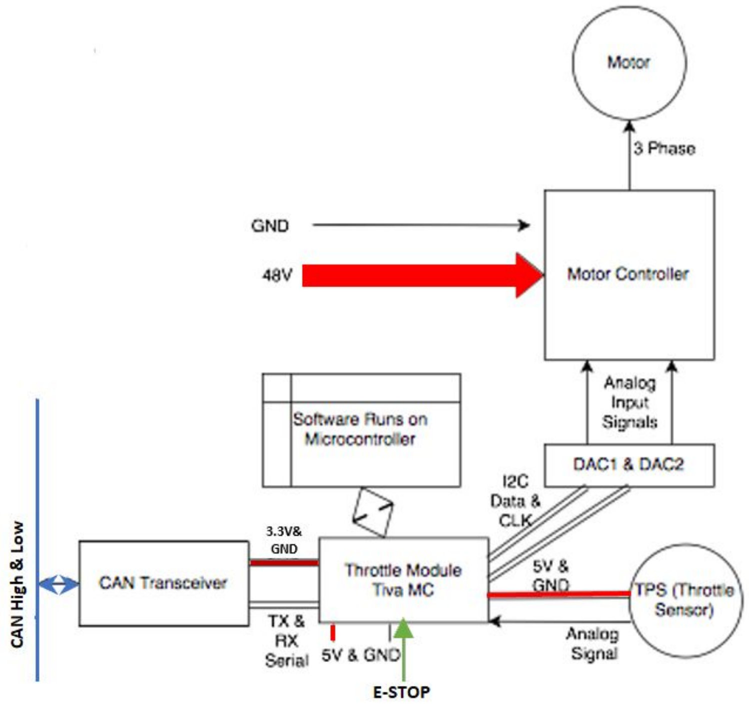


Figure 4: One Wire Hardware diagram for Throttle Module.

3.3 Steering Module - [Trace to requirements 3.3]

- 3.3.1 Tiva Launchpad - Hardware Component [Trace to requirements: ALL 3.3]
  This microcontroller was chosen because our team members were familiar with using this board from previous courses such as ECE 473 and ECE 4951. Additionally, this board meets all of the processing, timing, and power requirements needed to support the DbW system while being relatively inexpensive.
  - Vendor: Texas Instruments
  - Cost: $13.49
  - In-charge of all computation/scaling to control the steering of the vehicle
  - E-Stop switch analog input
  - CAN transceiver (CAN Tx & Rx Interface Protocol)
  - Runs C software component to control steering via CAN
  - The Tiva requires an input voltage of 4.75 VDC to 5.25 VDC

- 3.3.2 Electronic Steering Actuator - Hardware component [Trace to requirements: ALL 3.3]
  Steering Actuator came preinstalled on the vehicle by the OEM.
  - Vendor: Global Motors
  - Cost: $1499.99
  - Steering actuator that is controlled via manual input or CAN message
  - Outputs position of electronic actuator onto the CAN Bus
  - Outputs input torque of electronic actuator onto the CAN Bus

- 3.3.3 CAN Transceiver - Hardware component [Trace to requirement 3.3.2, 3.3.4, 3.3.5]
  We had previous experience integrating this specific CAN transceiver with the Tiva Launchpad. We decided to use it for our project in order to mitigate risk.
  - Vendor: Texas Instruments
  - Cost: $2.07
  - Interfaced to Tiva Microcontroller CAN Controller
  - Signal interface CAN RX/TX
  - Requires an input voltage of 3.3V from the Tiva
  - Voltage at CANH or CANL pins can be between -14V and 14V

- 3.3.4 E-STOP switch - Hardware component [Trace to requirement 3.3.3]
  We selected this E-Stop switch as its voltage, current, and temperature ratings met our requirements and operating conditions.
  - Vendor: IDEC
  - Cost: $51.90
  - Switch used to notify Microcontroller of an emergency
  - Roughly ~3.3V input into Tiva for normal operation, ~0V for emergency
  - Analog signal

- 3.3.5 Steering code - Software component [Traces to requirement: ALL 3.3]
  - C Language code
  - Receive CAN messages, scale and provides steering accordingly
  - Scale such that we can go from Center to Max turn left or right, within 2 seconds
  - If in DbW mode, steering will be controlled via CAN, otherwise via manual input
  - CAN Control messages to steering actuator will be generated and sent every 100 ms

- If Input Torque on steering wheel exceeds +7 Nm or less than -7 Nm, disengage DbW
- Count CAN messages, if no message is received within 300 ms, kill DbW mode
- If E-Stop analog signal is received, kill DbW mode
- CAN message set for the steering code can be seen in the Interface Description later in the next section of this document.

Steering module Message Sets:

Steering module will use three message sets. Its priority comes second between the three modules. Steering actuator takes precedence over joystick hence steering module to steering actuator is third priority and joystick/pxs to steering module is fourth priority.

Steering Module to Steering Actuator

Used to send steering data to steering actuator.

CAN ID	**DATA**							
	Byte 1	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**
0x18FF00F9	0x05	0x00	Steer Data	Steer Data	Steer Data	Steer Data	0x75	0x00

Joystick/PXS to Steering Module

Used to received x axis data from the joystick module.

CAN ID	**DATA**							
	Byte 1	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**
0x19DB0000	SRC ID	EVENT TYPE	STEERING DATA (12 Bit) MSB = 0		Res	Res	Res	Res

Module Feedback

Used to send and received data from the steering sensor.

CAN ID	**DATA**							
	Byte 1	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**
0x1CDBFFFF	SRC ID	SENSOR FEEDBACK (12 Bit)		RES	RES	RES	RES	RES

Steering module software flow chart:

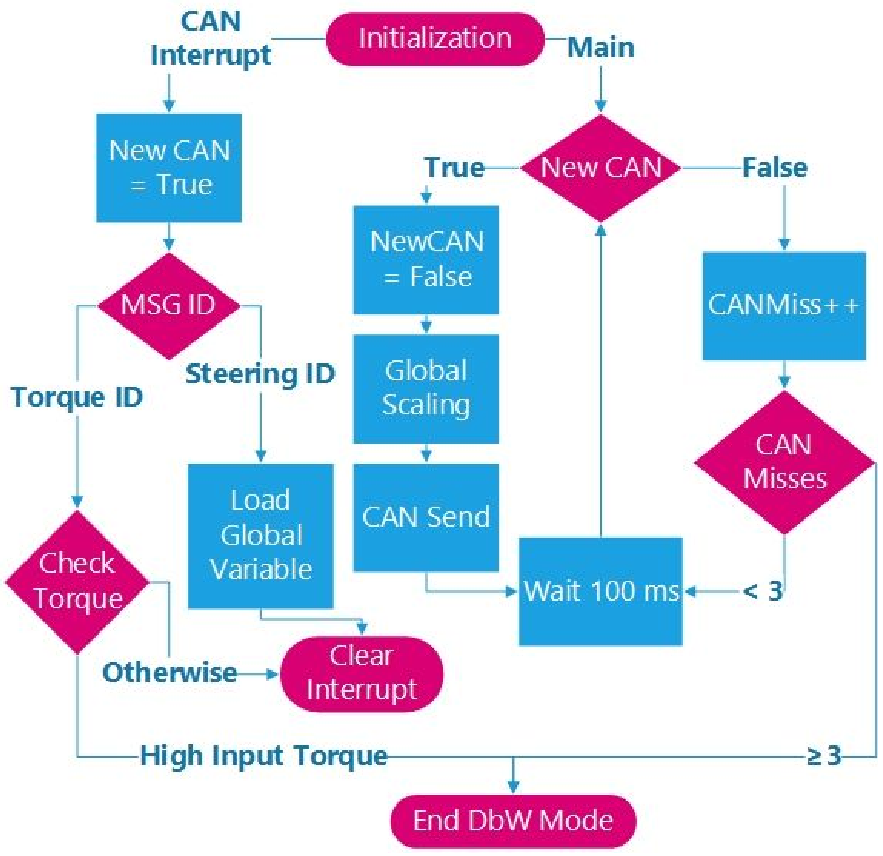


Figure 5: Software Flowchart for Steering Module.

Steering module hardware diagram:

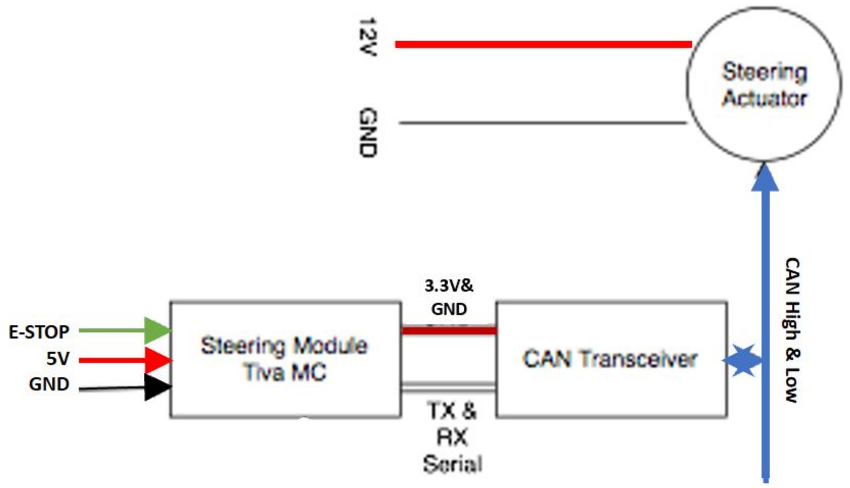


Figure 6: One Wire hardware diagram for Steering Module.

<u>3.4 Brakes Module - [Trace to requirements 3.4]</u>

- 3.4.1 Tiva Launchpad - Hardware Component [Trace to requirements: ALL 3.4]
  This microcontroller was chosen because our team members were familiar with using this board from previous courses such as ECE 473 and ECE 4951. Additionally, this board meets all of the processing, timing, and power requirements needed to support the DbW system while being relatively inexpensive.
  - Vendor: Texas Instruments
  - Cost: $13.49
  - In-charge of all computation/scaling to control the braking of the vehicle
  - E-Stop switch analog input
  - Brake Pressure Sensor analog input
  - CAN transceiver (CAN Tx & Rx Interface Protocol)
  - Runs C software component to control braking via CAN
  - The Tiva requires an input voltage of 4.75 VDC to 5.25 VDC

- 3.4.2 Electronic Linear Actuator - Hardware component [Trace to requirements: ALL 3.4]
  Linear actuator was installed in the vehicle by the MDAS.ai research team because the brake actuator installed by the OEM was inaccessible. This component, as such, was included before the start of the project.
  - Vendor: Kartec
  - Cost: $1499.99
  - Linear actuator that receives commanded actuation positions via CAN
  - Actuates to setpoint and requires a message every 100ms to hold setpoint

- 3.4.3 CAN Transceiver - Hardware component [Trace to requirement 3.4.2, 3.4.7, 3.4.8]
  We had previous experience integrating this specific CAN transceiver with the Tiva Launchpad. We decided to use it for our project in order to mitigate risk.
  - Vendor: Texas Instruments
  - Cost: $2.07
  - Interfaced to Tiva Microcontroller CAN Controller
  - Signal interface CAN RX/TX
  - Requires an input voltage of 3.3V from the Tiva
  - Voltage at CANH or CANL pins can be between -14V and 14V

- 3.4.4 E-STOP switch - Hardware component [Trace to requirement 3.4.3]
  We selected this E-Stop switch as its voltage, current, and temperature ratings met our requirements and operating conditions.
  - Vendor: IDEC
  - Cost: $51.90
  - Switch used to notify Microcontroller of an emergency
  - Roughly ~3.3V input into Tiva for normal operation, ~0V for emergency
  - Analog signal

- 3.4.5 Brake Pressure Sensor - Hardware component [Trace to requirement 3.4.1, 3.4.6, 3.4.8]
  Brake Pressure Sensor was preinstalled by the OEM.
  - Vendor: Race Technology
  - Cost: $33

- Reads pressure of Brake lines
- Outputs Analog signal between 0V to 5V depending on the pressure in the lines
- Analog voltage read by PI control loop and further actuates or reduces actuation based on setpoint

- 3.4.6 Brake Module code - Software component [Traces to requirement: ALL 3.4]
  - C Language code
  - Receive CAN messages, scale and provides braking accordingly
  - Software-defined PI loop that will use the output of the brake pressure sensor to achieve the desired setpoint.
  - If in DbW mode, braking will be controlled via CAN, otherwise via manual input
  - The CAN control messages received will have a setpoint in the data fields that will specify from 0% to 100% braking
  - CAN Control messages to brake actuator will be generated and sent every 100 ms
  - Count CAN messages, if no message is received within 300 ms, kill DbW mode
  - If E-Stop analog signal is received, kill DbW mode
  - Transmit brake pressure on vehicle CAN Bus every 100ms.

Brake module Message Sets:

Brake module will use three message sets. It is the highest priority. Actuators takes priority over joystick/PX2 to brake module to prioritise the brake actuator activating. Brake actuator is the highest priority and the joystick/PX2 to brake module being 2nd highest priority.

Brake module to brake actuator

Used to send brake data from brake module to brake actuator.

CAN ID	DATA							
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x0FF00000	Brake Data	Brake Data	Brake Data	Brake Data	essure (12 bit)		Res	Res

Joystick/PX2 to Brake Module

Used to send negative y axis to the brake module.

CAN ID	DATA							
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x18DB0000	SRC ID	EVENT TYPE	DATA (12 Bit)		Res	Res	Res	Res

Module Feedback

Used to send and received data from the brakes.

CAN ID	DATA							
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x1CDBFFFF	SRC ID	SENSOR FEEDBACK (12 Bit)		RES	RES	RES	RES	RES

Brake module software flow chart:

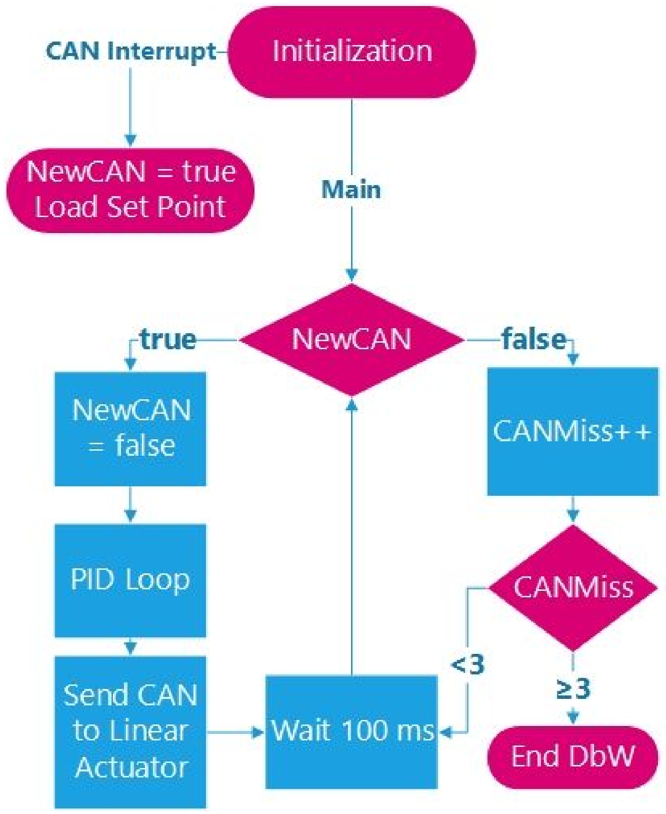


Figure 7: Software Flowchart for Brake Module.

Brake module hardware diagram:

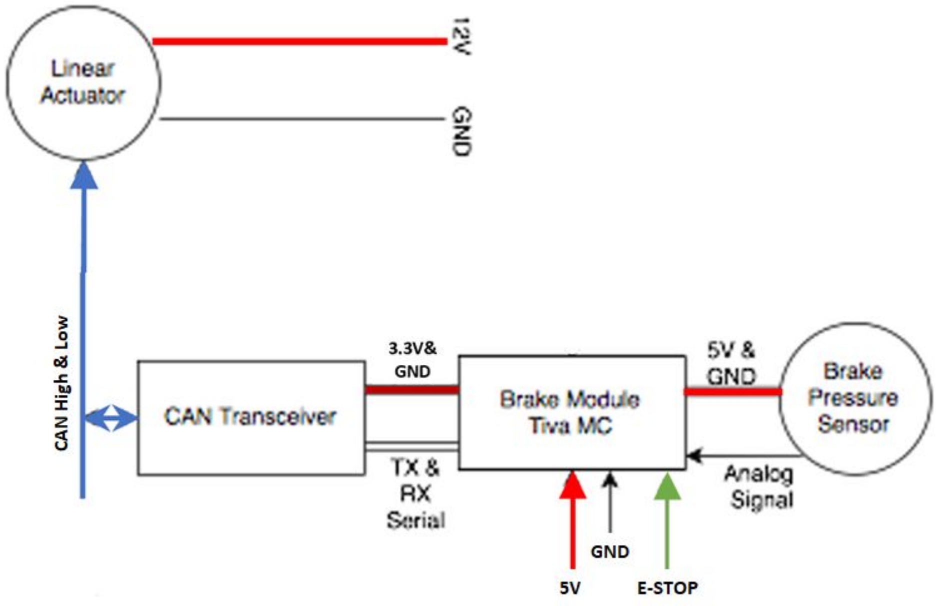


Figure 8: One Wire Hardware Flowchart for Brake Module

3.5 Joystick Module - [Trace to requirements 3.5]

- 3.5.1 Tiva Launchpad - Hardware Component [Trace to requirements: ALL 3.5]
  This microcontroller was chosen because our team members were familiar with using this board from previous courses such as ECE 473 and ECE 4951. Additionally, this board meets all of the processing, timing, and power requirements needed to support the DbW system while being relatively inexpensive.
  - Vendor: Texas Instruments
  - Cost: $13.49
  - CAN transceiver (CAN Tx & Rx Interface Protocol)
  - In-charge of Analog-to-Digital conversion of Joystick
  - Transmit digital positions of X & Y Axis of the joystick on CAN Bus
  - Pushbutton used as a deadman switch
  - Runs C software component to control vehicle via CAN
  - The Tiva requires an input voltage of 4.75 VDC to 5.25 VDC

- 3.5.2 Analog Joystick board - Hardware component [Trace to requirements: ALL 3.5]
  - Vendor: Texas Instruments
  - Cost: $29.99
  - Analog output signals, one signal for X-axis, one signal for the Y axis
  - Interfaced to Tiva via analog signaling
  - Pinouts
    - X axis: J1.2
    - Y axis: J3.26
    - Pushbutton: J1.5

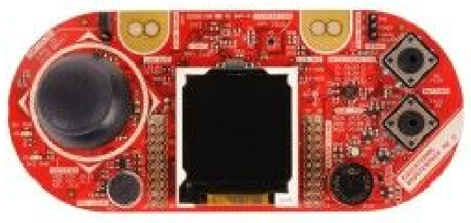


Figure 9: Joystick that is socketed to Tiva

- 3.5.3 CAN Transceiver - Hardware component [Trace to requirement ALL 3.5]
  We had previous experience integrating this specific CAN transceiver with the Tiva Launchpad. We decided to use it for our project in order to mitigate risk.
  - Vendor: Texas Instruments
  - Cost: $2.07
  - Interfaced to Tiva Microcontroller CAN Controller
  - Signal interface CAN RX/TX
  - Requires an input voltage of 3.3V from the Tiva
  - Voltage at CANH or CANL pins can be between -14V and 14V

- 3.5.4 Joystick Module code - Software component [Traces to requirement: ALL 3.5]
    - C Language code
    - Analog-to-Digital conversion of the joystick, Two Analog signals: one for X, one for Y
    - Check if the Deadman switch is pressed, if not, do not send CAN messages
    - Send three messages on the CAN Bus
        - X-Axis message for steering
        - Positive Y-Axis for Throttle
        - Negative Y-Axis for Brake
    - Each message sent at 100ms on CAN Bus

<u>Joystick module Message Sets:</u>

Joystick/PX2 to module priority takes after their respective tiva to actuator message sets. Actuators take priority to ensure the actuators properly activate.

<u>Joystick/PX2 to brake module</u>

CAN ID	DATA							
	Byte 1	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**
0x18DB00 00	SRC ID	EVENT TYPE	DATA (12 Bit)		Res	Res	Res	Res

<u>Joystick/PX2 to steering module</u>

CAN ID	DATA							
	Byte 1	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**
0x19DB00 00	SRC ID	EVENT TYPE	STEERING DATA (12 Bit) MSB = 0		Res	Res	Res	Res

<u>Joystick/PX2 to throttle module</u>

CAN ID	DATA							
	Byte 1	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**
0x1ADB00 00	SRC ID	EVENT TYPE	THROTTLE DATA (12 Bit) MSB = 0		Res	Res	Res	Res

<u>Joystick module hardware diagram:</u>

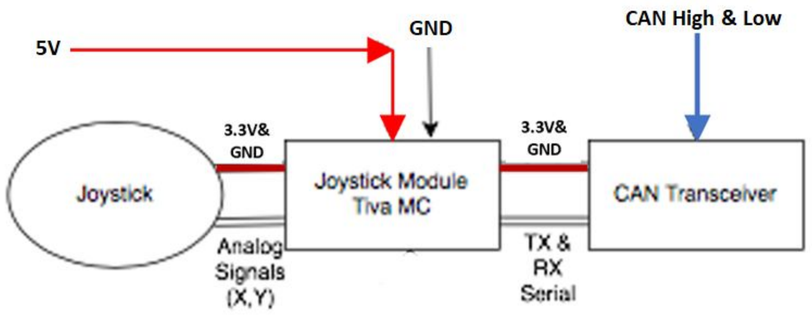


Figure 10: Block Diagram for Joystick Module.

<u>Joystick module software diagram:</u>

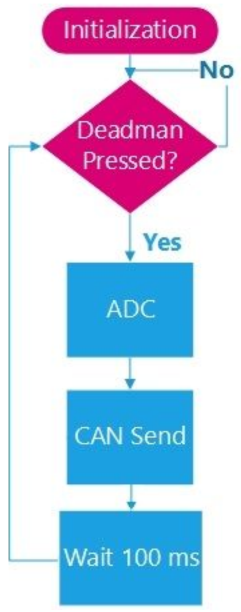


Figure 11: Software Flowchart for Joystick Module.

4. INTERFACE DESCRIPTION

The majority of this project will be focused on creating a software interface between steering, brake, throttle, and joystick modules. This will come in the form of a CAN message set that we will create. CAN will be the communication protocol between the Onboard Unit and the brake actuator, steering actuator, brake module, steering module, throttle module, and joystick. We will be utilizing the CAN ISO Standard 11898 to construct our message set. There will also have to be an interface between the Roadside Unit and Onboard Unit in the form of DSRC in the 5.9 GHz band. For this communication standard, we will be utilizing the IEEE 1609 Spec for DSRC. There will also be I^2C communication between the DACs and throttle module. We will be using the I^2C Standard from NXP. All of these documents have links associated with them in the References section of this report. Finally, the communication between the Emergency Stop and the brake, steering, and throttle modules will be an analog signal.

4.1 DSRC module Interface Description

The system will use DSRC V2X communication to alter the system's state in order to enhance the safety of the DbW system by relaying information about the surroundings. The OBU radio will receive a message from the RSU radio via RF communication and generate a CAN message to update the other modules on the vehicle state.

Interfaces for this module:

- 12V power supply for DSRC Radios
- Sockets interface with a laptop for RSU
- 5.9 GHz RF Communication
- CAN communication for OBU to talk to rest of the vehicle

4.2 Throttle module Interface Description

The system has a throttle module in order to control the state of the motor controller. It will read the throttle pedal analog sensor and listen a for CAN message for control information. We will also publish the Throttle Position Sensor (TPS) sensor reading onto the CAN bus such that the neural network in the MDAS vehicle can be trained. This is a requirement from the neural team that is working on the MDAS vehicle. The message will be parsed and control two I^2C DAC's that will provide two analog signals to control the vehicles speed via the motor controller. If the E-Stop is engaged, the module will not provide any analog signals to the motor controller. After the E-Stop is disengaged and the vehicle's power is cycled, the modules will return to their normal operating state.

Interfaces for this module:

- I^2C communication protocol from Tiva to DAC's
- Analog Signals from DAC's to the motor controller
- Analog Signal from E-Stop & TPS to Tiva
- CAN communication from Tiva to rest of the modules
- 5V power supply for Tiva & TPS Sensor
- 3.3V power supply for CAN Transceiver

4.3 Steering module Interface Description

The system will have a steering module to provide steering actuation for a safe DbW system. The steering module will respond to CAN communication from the OBU and the joystick module. When the steering module responds to notifications from the OBU or the joystick module, meaningful scaling will be performed and a CAN message will be sent to the steering actuator. The steering module will also be actively listening for a message from the input torque sensor. If there is an input torque applied to the steering wheel by the safety driver during DbW mode, it will safely deactivate DbW and notify the other modules.

Interfaces for this module:

- Analog Signal from E-Stop to Tiva
- CAN communication from Tiva to rest of the modules
- 5V power supply for Tiva
- 3.3V power supply for CAN Transceiver

4.4 Brake module Interface Description

This module of the system will control the braking of the vehicle. It will receive a message via CAN communication requesting a set position (i.e. brake pressure) and the PI loop will handle brake actuation in order to achieve the set position. It will do this by reading the pressure of the Brake Pressure Sensor (BPS) and comparing the difference between the setpoint pressure and the current pressure. In the event the E-STOP is pressed, the brake module will bring the vehicle to a safe stop and disengage DbW mode. This safe mode will not be disengaged until the E-STOP is no longer being depressed and the power on the vehicle is cycled.

Interfaces for this module:

- Analog Signal from E-Stop & BPS to Tiva
- CAN communication from Tiva to rest of the modules
- 5V power supply for Tiva & BPS
- 3.3V power supply for CAN Transceiver

4.5 Joystick module Interface Description

The last module of the system will be the joystick module. This will be used as a way to simulate autonomy in the vehicle, in addition, being a debugging tool. It will also be used to verify the operational integrity of the DbW system. This control will also take priority over PX2 for safety and testing purposes. The joystick will have two analog signal outputs that are fed into the Tiva and then will be converted using an ADC and sent to other modules via CAN communication.

Interfaces for this module:

- Two Analog Signals from Joystick to Tiva
- CAN communication from Tiva to rest of the modules
- 5V power supply for Tiva & Joystick
- 3.3V power supply for CAN Transceiver

Low Level design diagram:

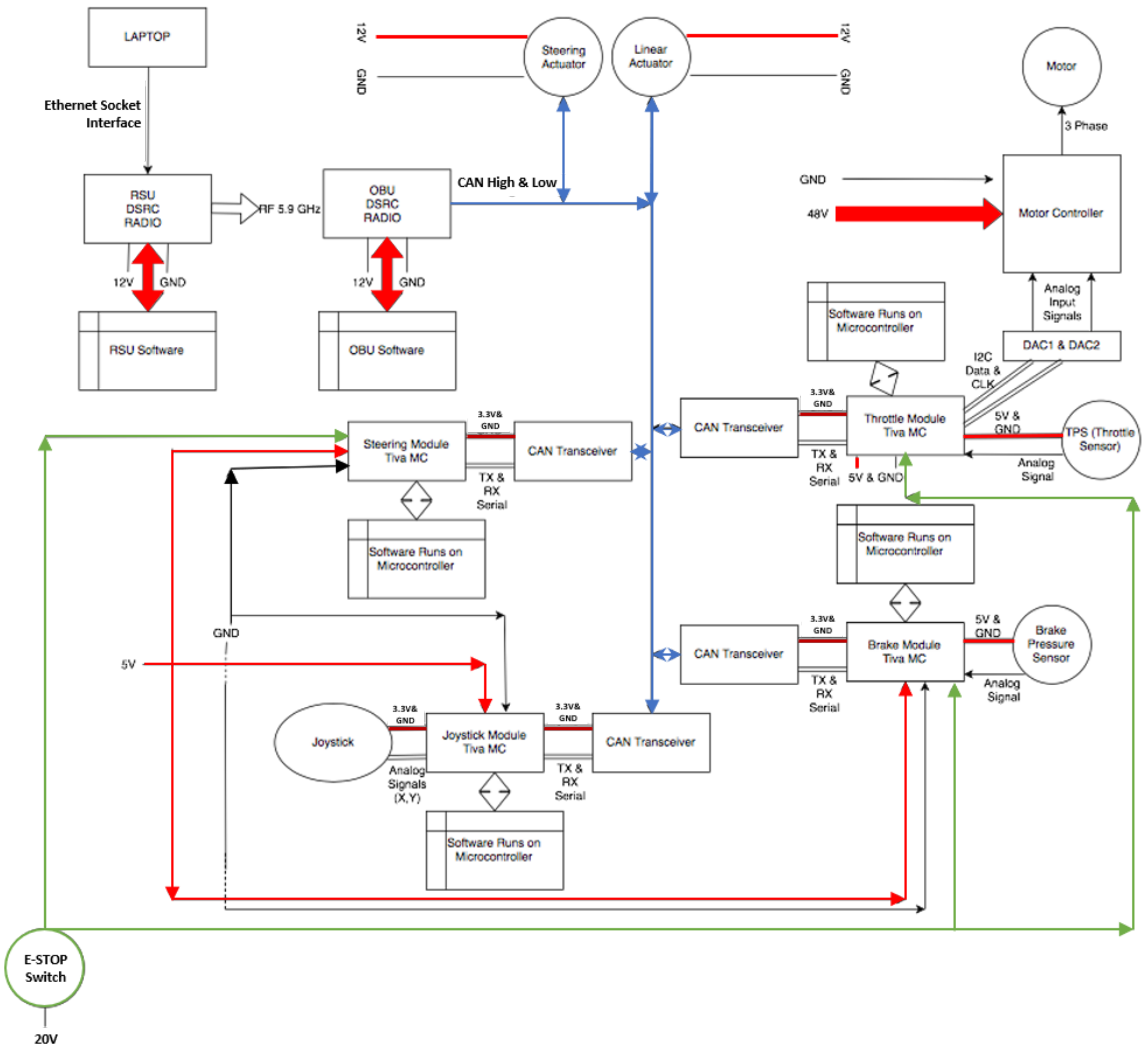


Figure 11: MDAS.ai Drive-by-Wire System Low Level Diagram

● Vehicle power source to Tiva:	5V
● Tiva power source to I2C DAC's:	5V
● Tiva power source to CAN Transceiver:	3.3V
● Tiva power source to Joystick:	3.3V
● Tiva power source to sensors:	5V
● DSRC Radio's power source:	12V
● Linear & Steering Actuator power source:	12V

- Communication Interface includes: DSRC, I2C, CAN

E-Stop Step-down Circuit

As illustrated in the one-wire diagram, a large voltage source 20-24V which was stepped down from the car battery 48V is connected to the E-Stop Switch. Since this button is just a single, normally-closed relay, the voltage into the switch must be reduced before being input into the Tiva microcontrollers because they are not rated for that degree of voltage. The following figure displays the design for this voltage step-down. Resistor values in the diagram are general and will be adjusted based on actual voltage and current outputs from the car source. The output of the comparator will be an analog voltage of about 3.3V routed to a single pin on each of the module Tiva microcontrollers, acting as an interrupt to normal operation and shut down Drive-by-Wire mode. The brake module will respond by slowing the car to a stop as quickly and safely as possible before shutting off as well.

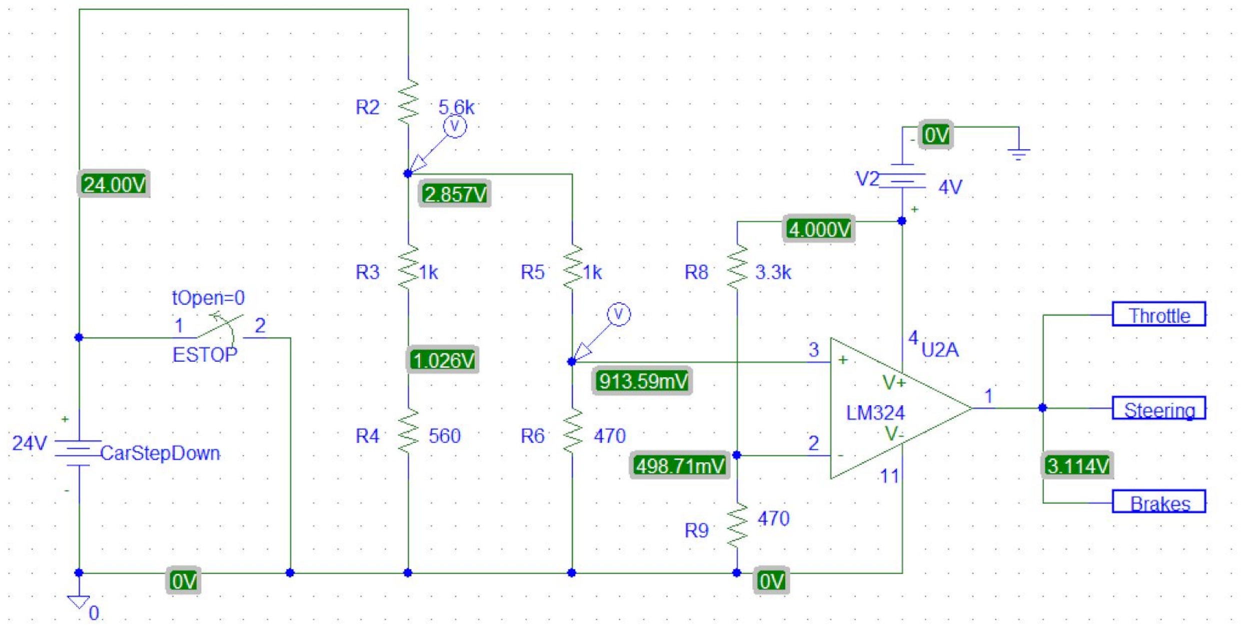
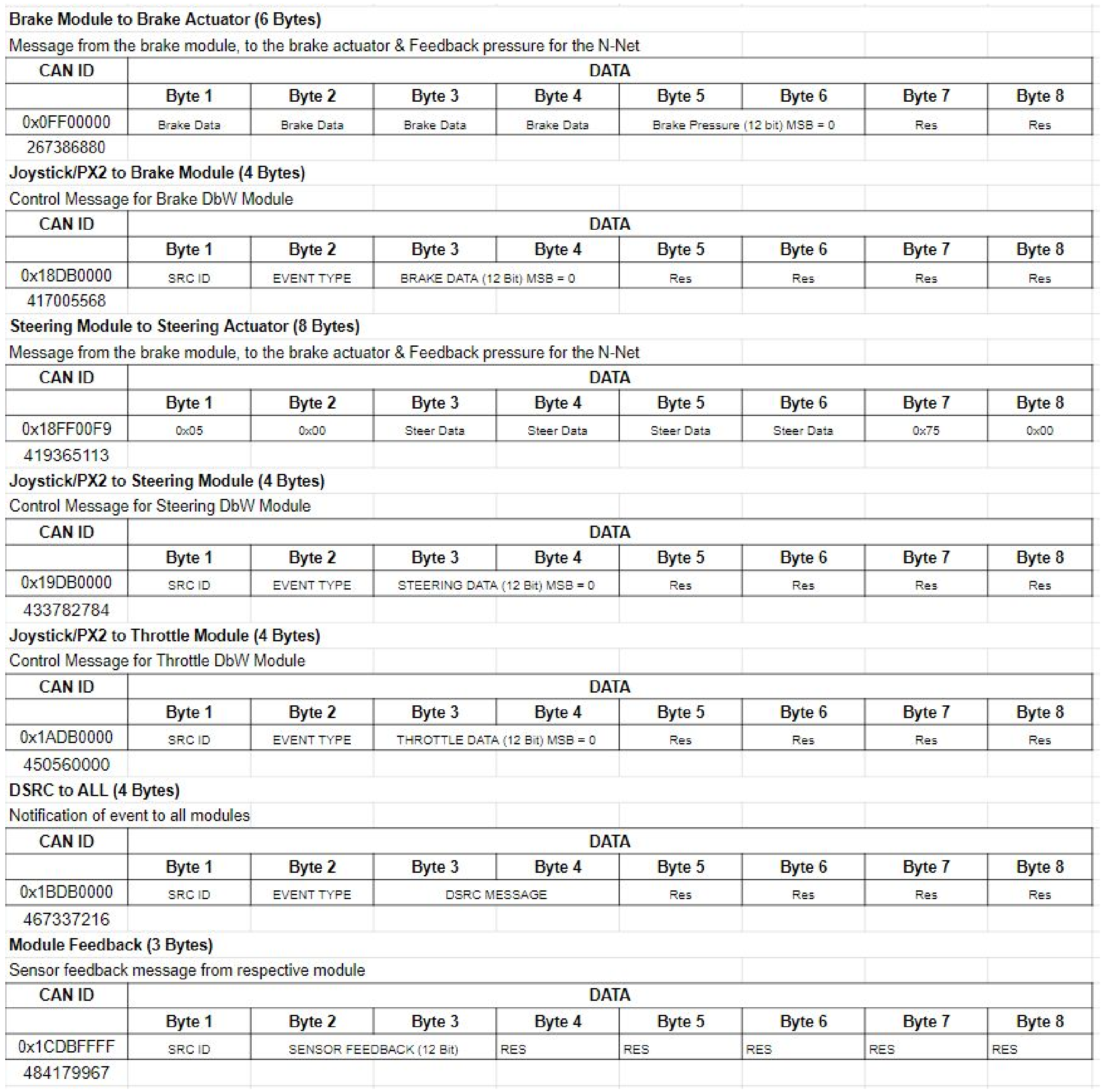


Figure 12: E-Stop Voltage Divider Circuit

Priority Ordered Defined message set:



Brake Module to Brake Actuator (6 Bytes)

Message from the brake module, to the brake actuator & Feedback pressure for the N-Net

CAN ID	DATA							
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x0FF00000	Brake Data	Brake Data	Brake Data	Brake Data	Brake Pressure (12 bit) MSB = 0		Res	Res
267386880								

Joystick/PX2 to Brake Module (4 Bytes)

Control Message for Brake DbW Module

CAN ID	DATA							
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x18DB0000	SRC ID	EVENT TYPE	BRAKE DATA (12 Bit) MSB = 0		Res	Res	Res	Res
417005568								

Steering Module to Steering Actuator (8 Bytes)

Message from the brake module, to the brake actuator & Feedback pressure for the N-Net

CAN ID	DATA							
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x18FF00F9	0x05	0x00	Steer Data	Steer Data	Steer Data	Steer Data	0x75	0x00
419365113								

Joystick/PX2 to Steering Module (4 Bytes)

Control Message for Steering DbW Module

CAN ID	DATA							
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x19DB0000	SRC ID	EVENT TYPE	STEERING DATA (12 Bit) MSB = 0		Res	Res	Res	Res
433782784								

Joystick/PX2 to Throttle Module (4 Bytes)

Control Message for Throttle DbW Module

CAN ID	DATA							
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x1ADB0000	SRC ID	EVENT TYPE	THROTTLE DATA (12 Bit) MSB = 0		Res	Res	Res	Res
450560000								

DSRC to ALL (4 Bytes)

Notification of event to all modules

CAN ID	DATA							
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x1BDB0000	SRC ID	EVENT TYPE	DSRC MESSAGE		Res	Res	Res	Res
467337216								

Module Feedback (3 Bytes)

Sensor feedback message from respective module

CAN ID	DATA							
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x1CDBFFFF	SRC ID	SENSOR FEEDBACK (12 Bit)		RES	RES	RES	RES	RES
484179967								

Figure 13: MDAS.ai Drive-by-Wire System CAN message set

5. RISK ASSESSMENT AND RISK MITIGATION

5.1 Functional and Performance Risks

A. Improper Communication Protocols (6)
B. Noisy Environment (8)

5.2 Programmatic Risks

A. Loss of data (computer/document files) (4)

5.3 Safety and Reliability

A. Faulty Wiring/Wire Harness (5)
B. Lack of signal from RSU (low RSSI) (7)

5.4 Knowledge Base and Uncertainty

A. Lack of Subject Matter Knowledge (9) - All team members lack some of the information necessary to complete this project. This will mostly be alleviated by excursions.
   a. 9.1 Current MDAS.ai systems
   b. 9.2 Digital to Analog Converters and Integration with Microcontrollers
   c. 9.3 Create a CAN message set
   d. 9.4 Sending a CAN message using DSRC radios
   e. 9.5 Closing PI control loops for the braking system

Risk Assessment Table:

🟥 red = alert
🟨 yellow = warning
🟩 green = caution

Penalty of Occurrence 10	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
9	🟨	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
8	🟨	🟨	🟨	🟥	🟥	🟥	🟥	🟥	🟥	🟥
7	🟨	🟨	🟨	🟨	🟥	🟥	🟥	🟥	🟥	🟥
6	🟨	🟨	🟨 5	🟨	🟨	🟥	🟥	🟥	🟥	🟥
5	🟩 7	🟨	🟨	🟨	🟨	🟨	🟥	🟥	🟥	🟥
4	🟩 6	🟩	🟨	🟨	🟨	🟨	🟨	🟥	🟥	🟥
3	🟩 4	🟩 9.3	🟩 9.5	🟨	🟨	🟨	🟨	🟨	🟥	🟥
2	🟩 8	🟩	🟩	🟩	🟨	🟨	🟨	🟨	🟨	🟥 9.4
1	🟩 9.2	🟩	🟩	🟩	🟩	🟩 9.1	🟨	🟨	🟨	🟨
	1	2	3	4	5	6	7	8	9	10
	Risk of Occurrence									
	very unlikely									very likely

Table: Risk Assessment (8 February 2019)

5.5 Risk Mitigation Steps for any Warning or Alert Level Risks:

Risk Mitigation Steps

- 7,8,9: Excursions and component-testing before implementation and connection with other components
- 6: Detailed documentation
- 4: Redundant storage of all documents and project files in Google Drive Cloud
- 5: Consistent wiring color index to identify the type of signal

Most risks outlined in previous documents pertinent to this project have been mitigated as this project has progressed. Further experience with the components in use as well as individual and team research, study, and experimentation have reduced or eliminated risks that were originally in the warning or alert levels.

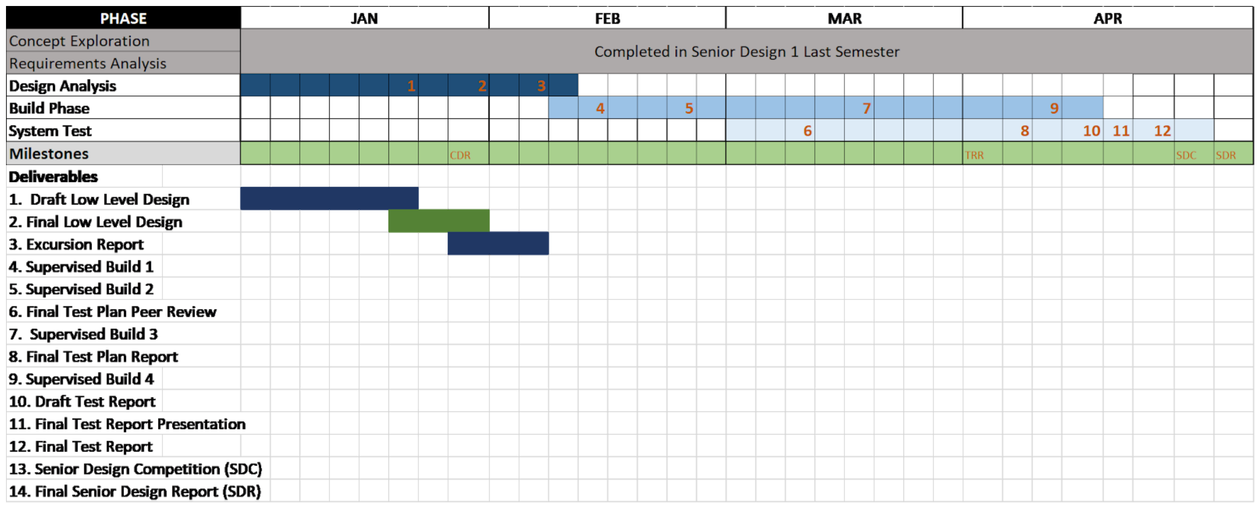
6. BUDGET INFORMATION

Cost Estimate (Version 1, Low Level Design Specification, 8 February 2019)

Module	#NO	Item	Vendor	Model/Part No.	Unit Cost	Qty	Sub Total Cost
RF Module	1	DSRC Radio	Cohda	MK5 OBU/RSU	$2000.00	2	$4000.00
Microcontroller	1-5	Tiva Launchpad	TI	TM4C123GXL	$13.49	5	$67.45
Digital-to-Analog	1-2	DAC	Microchip	MCP4725	$2.49	2	$4.99
Steering Actuator	1	Steering Actuator	Global Motors	GMA12	$1499.99	1	$1499.99
Linear Actuator	1	Linear Actuator	Kartec	1A0011BJ	$1499.99	1	$1499.99
CAN Transceiver	1-5	CAN Transceiver	TI	TCAN332DR	$2.07	5	$10.35
E-Stop	1	E-Stop Switch	IDEC	AVD301N-R	$51.90	1	$51.90
Electronic Throttle	1	TPS	Polaris GEM	4014989	$124.99	1	$124.99
Brake Pressure Sensor	1	BPS	Race Tech.	M10X1	$33	1	$33
Joystick MK2	1	Joystick	TI	EDUMKII	$29.99	1	$29.99
Total Cost Estimates							**$7323.00***

* All components provided by MDAS.ai and DSRC budget

7. MASTER SCHEDULE (Low Level Design Specification) - 8 February 2019



7.1 PROJECT STATUS (8 February 2019)

Concept exploration phase was completed following completion of the concept description and presentation at which point requirements analysis began. The requirements analysis phase is also complete following the system requirements review. Excursions definitions and budget have been developed. High-level design was completed in the previous semester. Currently, we are in the low-level design analysis phase. The test plan is also in progress which is part of this phase. We are currently on track to maintain the schedule depicted above.

PARTS STATUS			SEPT								OCT								NOV								DEC							
module	no																																	
RF Module	1	DSRC Radio	R																															
Microcontroller	5	Tiva Launch Pad	R																															
Digital-to-Analog	2	DAC	R																															
Steering Actuator	1	Steering Actuator	R																															
Linear Actuator	1	Linear Actuator	R																															
CAN Transceiver	5	CAN Transceiver	R																															
E-Stop	1	E-Stop Switch	R																															
Electronic Throttle	1	TPS	R																															
Brake Pressure Sensor	1	BPS	R																															
Joystick MK2	1	Joystick	R																															
APPROVED	A																																	
ORDERED	O																																	
Projected Delivery	R																																	
Received	R																																	