SENIOR DESIGN 1

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

HIGH LEVEL DESIGN SPECIFICATION

VERSION 2 / 14 DECEMBER 2018

FALL 2018

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

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

Table of Contents

1. INTRODUCTION

1.1 Purpose

This document define the high level design modules for the MDAS.ai Drive-by-Wire system using V2X for Enhanced Safety. This document will explain the hardware and software modules of our design and 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.

Figure: Pedestrians in Crosswalk example for DSRC external event

Figure: 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, ect)
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

module 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). The derivative piece "D" in full PID feedback controllers is neglected because exact position is not required by our system. 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.

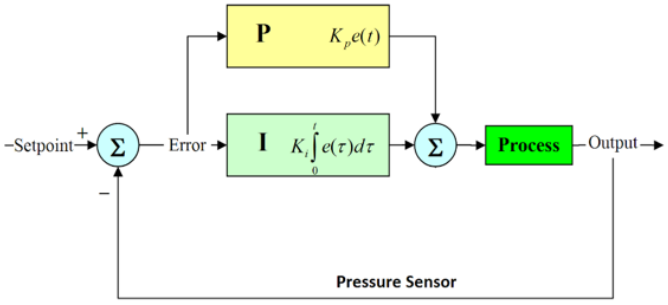


Figure: PI Controller with Pressure Sensor output along the feedback path

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 vehicles 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.

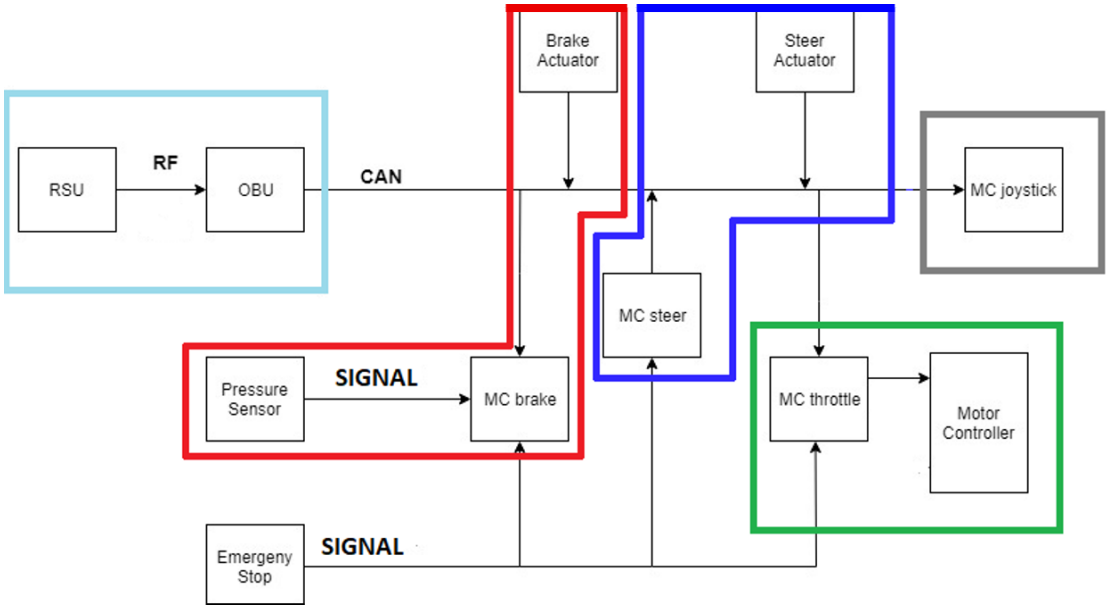


Figure 1: MDAS.ai Drive-by-Wire System Block Diagram
DSRC module (light blue), Brake Module (red), Steering Module (dark blue), Throttle Module (green), and Joystick Module (gray)

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

7. 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 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. HIGH 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]
    - Vendor: Cohda Wireless
    - Cost: $2000 each
    - Operating System: Modified Radio Linux

- 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

- 3.1.3 sender Radio Software - Software component [Traces to requirement: 3.1.6]
    - Send message via socket interface to OBU
    - Socket Port number chosen: 10000

DSRC hardware & Software diagram:

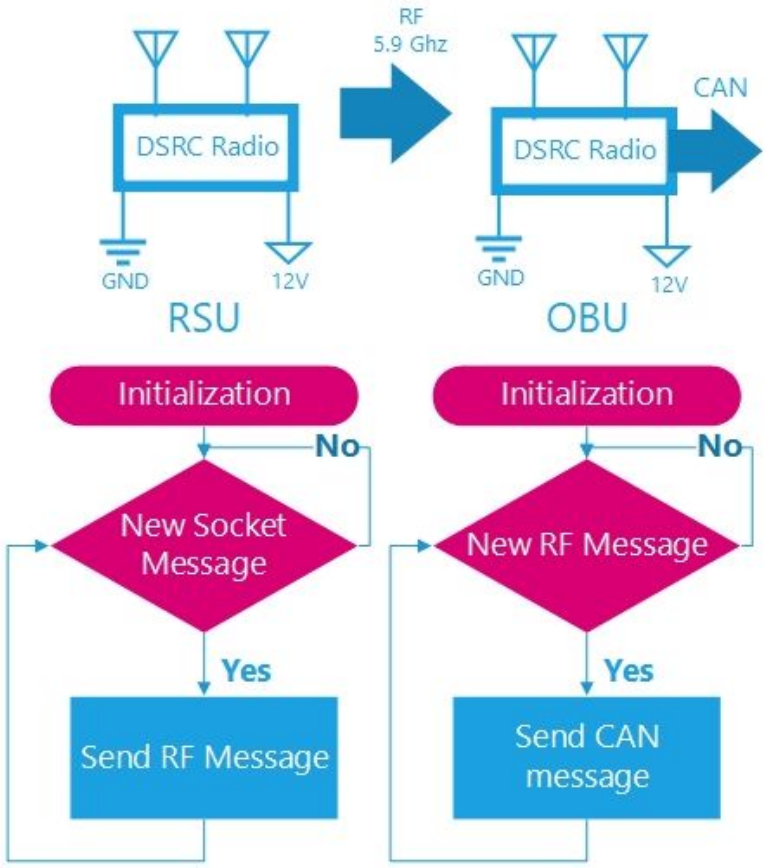


Figure 2: Hardware & Software Diagram for DSRC Module

3.2 Throttle Module - [Trace to requirements 3.2]

- 3.2.1 Tiva Launchpad - Hardware Component [Trace to requirements: ALL 3.2]
    - 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 (I2C Interface Protocol)
    - CAN transceiver (CAN Tx & Rx Interface Protocol)
    - Runs C software component to control throttle via CAN & DAC's

- 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
    - I2C interface to microcontroller
    - DAC2 will be ½(DAC1) at all times.
    - DAC1 operation range is 1-4V, DAC2 operation range is .5-2V

- 3.2.3 CAN Transceiver - Hardware component [Trace to requirement 3.2.4, 3.2.8]
    - Vendor: Texas Instruments
    - Cost: $2.07
    - Interfaced to Tiva Microcontroller CAN Controller
    - Signal interface CAN RX/TX

- 3.2.4 E-STOP switch - Hardware component [Trace to requirement 3.2.3, 3.2.6]
    - 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

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

Throttle module software flow chart:

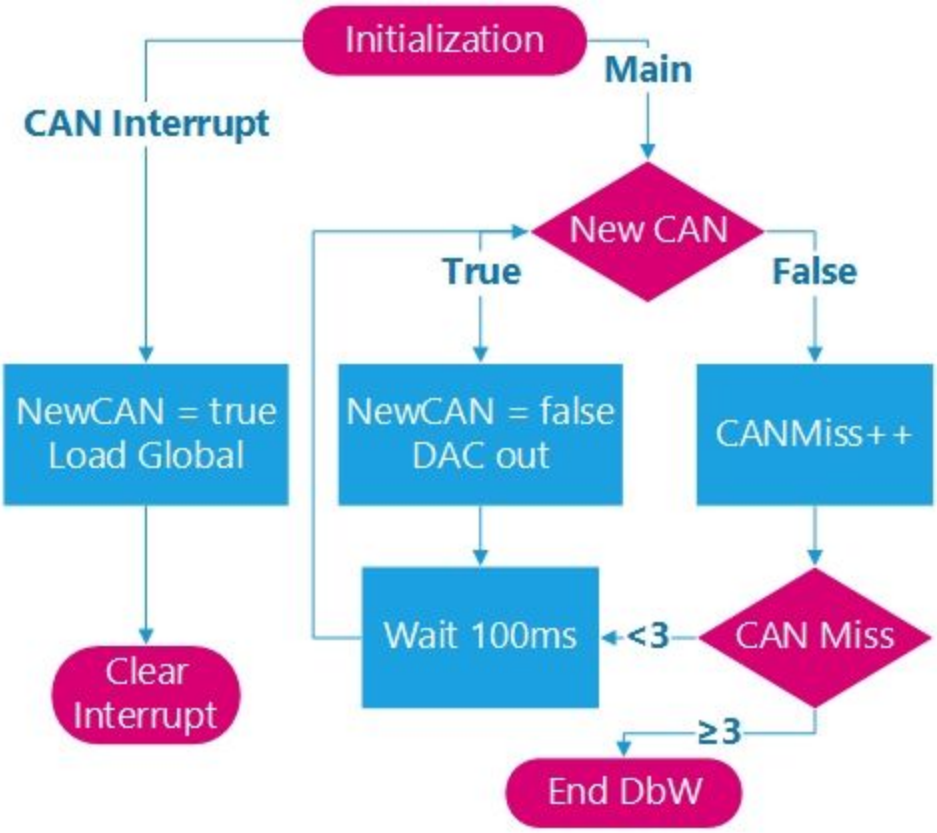


Figure 3: Software Flowchart for Throttle Module.

Throttle module hardware diagram:

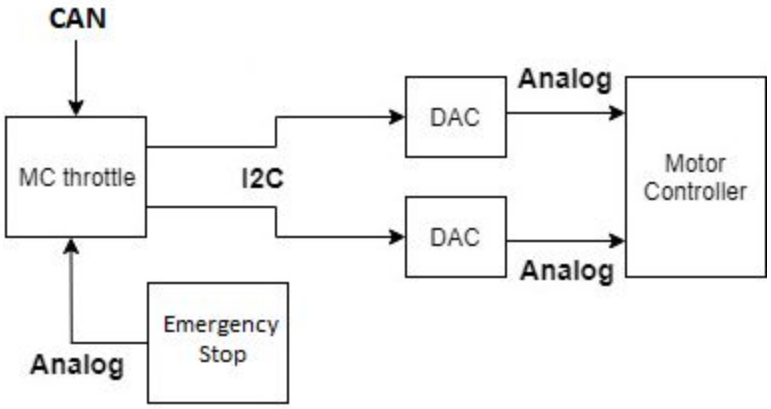


Figure 4: Hardware diagram for Throttle Module.

3.3 Steering Module - [Trace to requirements 3.3]

- 3.3.1 Tiva Launchpad - Hardware Component [Trace to requirements: ALL 3.3]
    - 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

- 3.3.2 Electronic Steering Actuator - Hardware component [Trace to requirements: ALL 3.3]
    - 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]
    - Vendor: Texas Instruments
    - Cost: $2.07
    - Interfaced to Tiva Microcontroller CAN Controller
    - Signal interface CAN RX/TX

- 3.3.4 E-STOP switch - Hardware component [Trace to requirement 3.3.3]
    - 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

Steering module software flow chart:

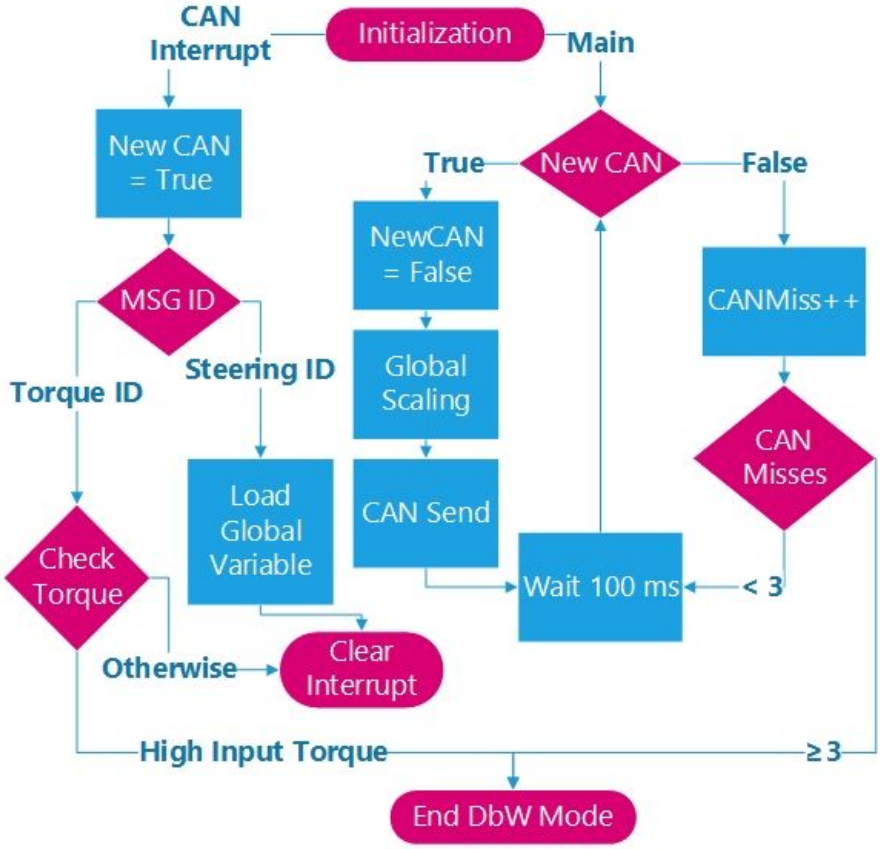


Figure 5: Software Flowchart for Steering Module.

Steering module hardware diagram:

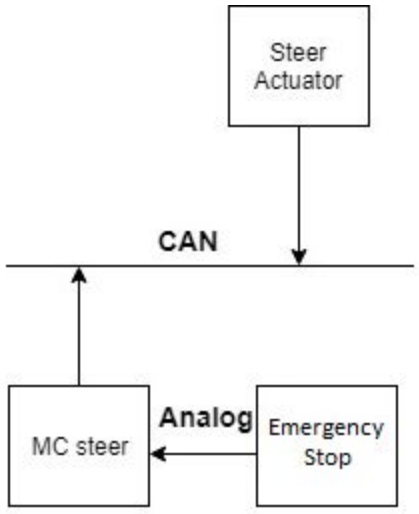


Figure 6: Hardware diagram for Steering Module.

3.4 Brakes Module - [Trace to requirements 3.4]

- 3.4.1 Tiva Launchpad - Hardware Component [Trace to requirements: ALL 3.4]
    - 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

- 3.4.2 Electronic Linear Actuator - Hardware component [Trace to requirements: ALL 3.4]
    - Vendor: Kartec
    - Cost: $1499.99
    - Linear actuator that receives requested actuation 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]
    - Vendor: Texas Instruments
    - Cost: $2.07
    - Interfaced to Tiva Microcontroller CAN Controller
    - Signal interface CAN RX/TX

- 3.4.4 E-STOP switch - Hardware component [Trace to requirement 3.4.3]
    - 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]
    - Vendor: Race Technology
    - Cost: $33
    - Reads pressure of Brake lines
    - Outputs Analog signal between 0V to 5V depending on the pressure in the lines

- 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 software flow chart:

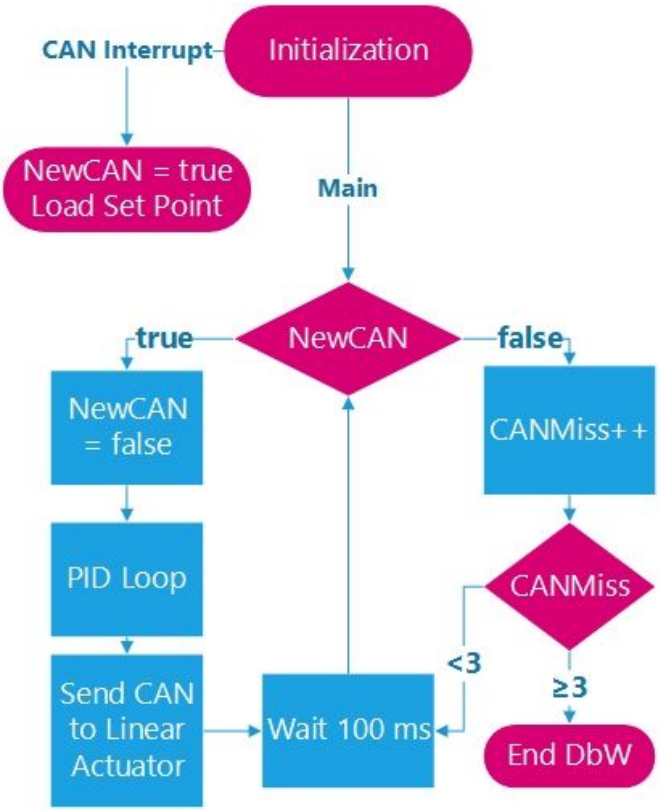


Figure 7: Software Flowchart for Brake Module.

Brake module hardware diagram:

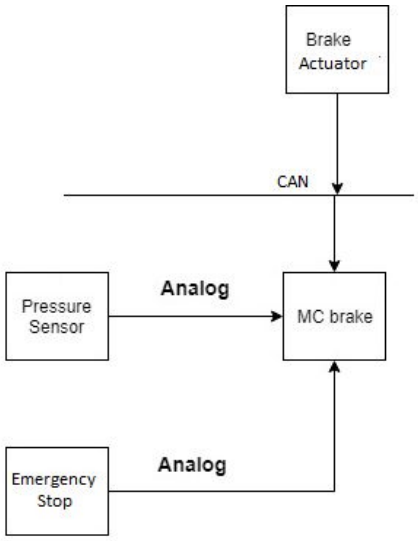


Figure 8: Software Flowchart for Brake Module.

3.5 Joystick Module - [Trace to requirements 3.5]

- 3.5.1 Tiva Launchpad - Hardware Component [Trace to requirements: ALL 3.5]
    - 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 joystick on CAN Bus
    - Pushbutton used as deadman switch
    - Runs C software component to control vehicle via CAN

- 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 Y axis
    - Interfaced to Tiva via analog signaling

- 3.5.3 CAN Transceiver - Hardware component [Trace to requirement ALL 3.5]
    - Vendor: Texas Instruments
    - Cost: $2.07
    - Interfaced to Tiva Microcontroller CAN Controller
    - Signal interface CAN RX/TX

- 3.5.4 Joystick Module code - Software component [Traces to requirement: ALL 3.5]
    - C Language code
    - Analog-to-Digital conversion of 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

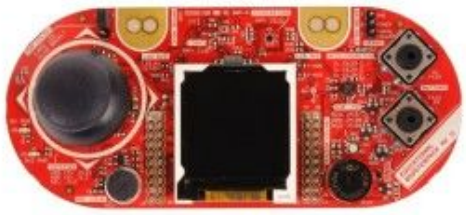


Figure 9: Joystick that is socketed to Tiva

Joystick module software diagram:

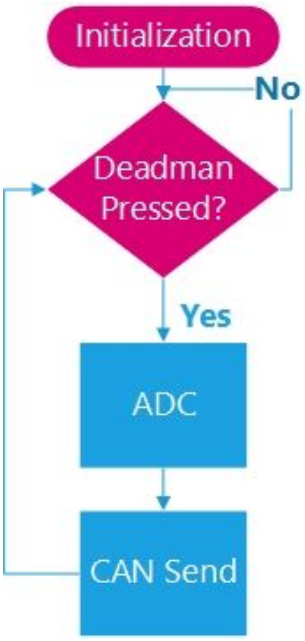


Figure 10: Software Flowchart for Joystick Module.

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.

Interface Description for this module:

- 12V power supply for DSRC Radios
- Sockets interface with 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 have 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.

Interface Description for this module:

- I^2C communication protocol from Tiva to DAC's
- Analog Signals from DAC's to 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.

Interface Description 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 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.

Interface Description 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.

Interface Description 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

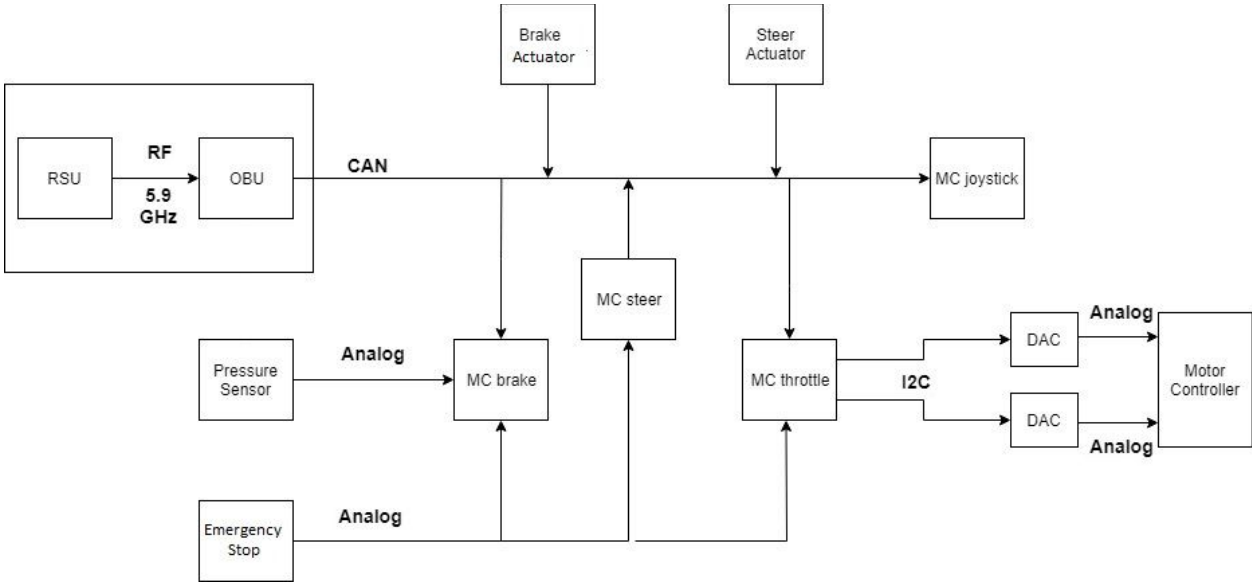


Figure 11: MDAS.ai Drive-by-Wire System Block Diagram With Interfaces Between Components.

- 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, CAN TX & RX

5. RISK ASSESSMENT AND RISK MITIGATION

5.1 Functional and Performance Risks

Improper Communication Protocols (6)

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 PID control loops for the braking system

Risk Assessment Table:

red = alert
yellow = warning
green = caution

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

Table: Risk Assessment (10 December 2018)

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

Risk Mitigation Steps

- 1,3: Creating a project plan to decrease likelihood of falling behind or missing deliverables
- 2: Miscommunication can be mitigated by creating a dependable communication channel for group meetings and assignment of tasks.
- 7,8,9: Excursions and component-testing before implementation and connection with other components
- 3,6: Detailed documentation
- 4: Redundant storage of all documents and project files in Google Drive Cloud
- 5: Consistent wiring color index to identify type of signal

Excursions

Excursion 1: Exploration of the Current Drive Systems in the MDAS.ai Vehicle

(1) Description of Risk Area:

The MDAS.ai vehicle was, originally, a traditionally operated car which is being converted into an autonomous shuttle. In order to translate the existing system to Drive-by-Wire, the team must have an understanding of what systems and components present in the vehicle. This excursion mitigates the risk of the

1. Lack of familiarity with electric vehicle systems
2. Lack of knowledge of what functions Drive-by-Wire controls

Current Risk Assessment

- Risk 9.1: Lack of Subject Matter Knowledge - Current MDAS.ai systems
  - Risk of Occurrence: 6
  - Penalty of Occurrence: 1

(2) Objective of Excursion

The objective of this excursion is to perform a set of small experiments on elements of the vehicle in order to reduce the risk of potentially disastrous mistakes while adapting traditional function to DbW. Lack of experience with vehicle systems will be mitigated by exploring all relevant systems in their entirety.

(3) Relevant Requirements

- In-depth mapping of the current drive system in the MDAS.ai vehicle
  - Mapping the 48 Volt system to the 12 Volt system
  - Signals that controls the vehicles motor controller & steering actuator

(4) Excursion Definition:

Description of Steps for Excursion

1. Draw a system block diagram on a whiteboard with the use of the vehicle data sheets.
2. Understand each of the blocks that are relevant to this project.

Possible Outcomes

1. Familiarize team with all vehicle motion systems and see what functionality is already in place.
2. Decide the scope of the project or the system description needs to be altered.

Excursion 2: Integrating a Digital-Analog Converter (DAC) with the Microcontroller

(1) Description of Risk Area:

This project involves conversion of a traditional vehicle to a Drive-by-Wire (DbW) system. One part of this project involves controlling the throttle of the vehicle. One possible approach to control this function is to interface a DAC with a microcontroller which will operate the motor controller already in the vehicle. This excursion mitigates, in part, multiple risks within our project:

1. Faulty wiring and wire harnesses
2. Improper Communication between the microcontroller and a DAC
3. Lack of knowledge of DACs

Current Risk Assessment

- Risk 5: Faulty Wiring/Wire Harness
  - Risk of Occurrence: 7
  - Penalty of Occurrence: 9
- Risk 6: Improper Communication Protocols
  - Risk of Occurrence: 1
  - Penalty of Occurrence: 7
- Risk 9.2: Lack of Subject Matter Knowledge - Digital to Analog Converters and Integration with Microcontrollers
  - Risk of Occurrence: 4
  - Penalty of Occurrence: 6

(2) Objective of Excursion

Perform an experiment which integrates the DAC with the microcontroller to reduce the above risks to ensure that the system performance requirements can be met. Additionally, this experiment will provide the opportunity for the team to learn about Digital-to-Analog converters.

(3) Relevant Requirements

- Control the vehicle speed from 0-100%
- Appropriate analog voltages output given digital input.
- Correct wire connections between the DAC and the microcontroller
- Investigate I2C as a communication protocol for the DAC
- Verify voltage output is corresponding with the 12 bit digital value (0 to 4095 in decimal)

(4) Excursion Definition:

Description of Steps for Excursion

1. Obtain microcontroller, DAC, and connectors
2. Integrate DAC to microcontroller using I2C
3. Provide digital values to the DAC and verify the output voltages

Completion Date and Time: October 3rd, 11:30 – 1:15 pm

Completion Status: **100%**

Results:

For this excursion, we examined, loaded, and ran code for the throttle board. We then proceeded to connect the throttle board to the MDAS.ai vehicle, successfully interfacing with the DAC on board. We also examined steering control already present in the vehicle. There was one issue with connection to

actual throttle which requires an intermediary between the power source and the DAC at initial startup of vehicle power. This will be investigated in another excursion to be completed before the build stage.

Excursion 3: Investigate CAN to create a proper message set

(1) Description of Risk Area:

This project involves modules communicating with one another via CAN (Controller Area Network). In order to build a safe Drive-by-Wire system for the MDAS.ai vehicle, the team will have to develop a CAN message set. This requires an in-depth understanding of the CAN protocol and how to use it to build our message set. This excursion mitigates the risk of:

1. Creating an improper message set
2. Using correct communication protocols for CAN
3. Lack of subject matter knowledge involving CA

Current Risk Assessment

- Risk 6: Improper Communication Protocols
  - Risk of Occurrence: 1
  - Penalty of Occurrence: 7
- Risk 9.3: Lack of Subject Matter Knowledge - Create a CAN message set
  - Risk of Occurrence: 4
  - Penalty of Occurrence: 3

(2) Objective of Excursion

Perform research and learn about CAN protocol in order to reduce the above risks to ensure that the system performance requirements can be met.

(3) Relevant Requirements

- Learn how to assign the CAN ID such that the priority is as intended
- Verify that we have 8 Bytes of data per message and 29 bit addressing

(4) Excursion Definition:

Description of Steps for Excursion

1. Research what CAN is from a high level of abstraction.
2. Read more in depth papers and resources in order to grasp the underlying details of how CAN works.
3. Explore code examples of CAN protocol.

Possible Outcomes

1. Learn enough about CAN through our research and exploration in order to perform a future excursion where we work with CAN ourselves.
2. Decide we need to seek help from our faculty advisor so that we can learn enough to perform a future excursion where we work with CAN ourselves.

Excursion 4: Use the DSRC radio to generate a CAN message

(1) Description of Risk Area:

A major goal of our project is to integrate V2X communication between the vehicle and infrastructure. We are going to use Dedicated Short Range Communication (DSRC) as our V2X protocol. The idea is to receive a message on an onboard DSRC radio and generate a CAN message that will notify the DbW system of an external event. This excursion mitigates the risk of the

1. Lack of familiarity with DSRC radios
2. Lack of knowledge with DSRC radio & CAN
3. Lack of Understanding with communication protocols for V2X

Current Risk Assessment

- Risk 6: Improper communication protocols
  - Risk of Occurrence: 1
  - Penalty of Occurrence: 7
- Risk 9.4: Lack of Subject Matter Knowledge - Sending a CAN message using DSRC radios
  - Risk of Occurrence: 10
  - Penalty of Occurrence: 2

(2) Objective of Excursion

The objective of this excursion is to investigate the ability of the DSRC radio to generate a CAN message to provide the vehicle with external data.

(3) Relevant Requirements

- Integrate SocketCAN into the DSRC radio
- Generate a CAN message and verify that it is being transmitted properly
- Investigate the J2735 Standard for DSRC radio communication

(4) Excursion Definition:

Description of Steps for Excursion

1. Setup and connect the DSRC radio.
2. Do research on SocketCAN and write a basic application to transmit CAN messages.
3. Verify it is sending CAN message by using a CAN-to-USB device.
4. Pick out a message from the J2735 standard and modify the application to only transmit a CAN message for a particular DSRC message.
5. Verify it is sending CAN message by using a CAN-to-USB device.

Possible Outcomes

1. Having a DSRC radio that can generate CAN messages.
2. Being able to parse DSRC messages and generating unique CAN messages.
3. CAN may not work on the DSRC radio as it has never been tested.

Excursion 5: GitHub as a Collaboration and Repository Tool

(1) Description of Risk Area:

The source code will be modified many times at different sections by multiple people. This could lead to unintended overlaps and/or overwrites. There is also the possibility of losing or damaging personal computers thus preventing access to the modified code. The proposed approach is to learn to use Git. GitHub will be used to host the source code. This mitigates the risk of losing data.

Current Risk Assessment

- Risk 4: Loss of Data/Project Files
  - Risk of Occurrence: 1
  - Penalty of Occurrence: 8

(2) Objective of Excursion

Learn to use Git commands to maintain a centralized source code in order to improve collaboration and reduce the risk of loss of project files.

(3) Relevant Requirements

- Organized code with comments and updates for each revision

(4) Excursion Definition:

Description of Steps for Excursion

1. To create a local repositories
2. Branch the master code
3. Learn to push and pull source code from GitHub.

Possible Outcomes

1. A centralized source code in GitHub.
2. A centralized source code in the personal computer.

Excursion 6: Closing a PID loop for brake system

(1) Description of Risk Area:

The brake system in the MDAS.ai vehicle does not have a proper control system in place to govern the braking of the vehicle by wire. One approach to alleviate this issue is to use a PID loop to provide a regulated amount of actuation to the brake pressure sensor (BPS) output as compared to the demanded BPS value. This excursion mitigates the risk of:

1. Lack of subject matter knowledge in PID controllers
2. Lack of experience with brake pressure sensors and brake actuators

Current Risk Assessment

- Risk 9.5: Lack of Subject Matter Knowledge - Closing PID control loops for the braking system
  - Risk of Occurrence: 3
  - Penalty of Occurrence: 4

(2) Objective of Excursion

Our primary objective for this excursion is to learn how to close a PID loop. Lack of experience in PID controllers makes separate experimentation with them essential to mitigate risk. This will also improve the team's knowledge of vehicle braking systems.

(3) Relevant Requirements

- Learning to close a PID loop & Closing the loop
- Apply 0 to 100% brake given the input to the loop

(4) Excursion Definition

Description of Steps for Excursion

1. Research PID controllers
2. Close the PID loop
3. Tune the proportional and integral (PI) parts of the loop
4. Verify that the setpoint and actuation is stable

Completion Date and Time: November 16, 12:00-7:30pm; November 30, 1:00-3:00pm

Completion Status: 80%

Results:

To determine slew rate and the control parameters, a short program was written to actuate the linear actuator within the brake system of MDAS.ai. An indicator LED would be lit for 5 seconds indicating a HIGH state and then go off for 5 seconds indicating a LOW state of actuation. These two states had to be adjusted several times in order to establish a suitable margin for measurement and one that would not bind the actuator, tainting the results. The final numbers that controlled actuation produced a set voltage which could be measured by a multimeter:

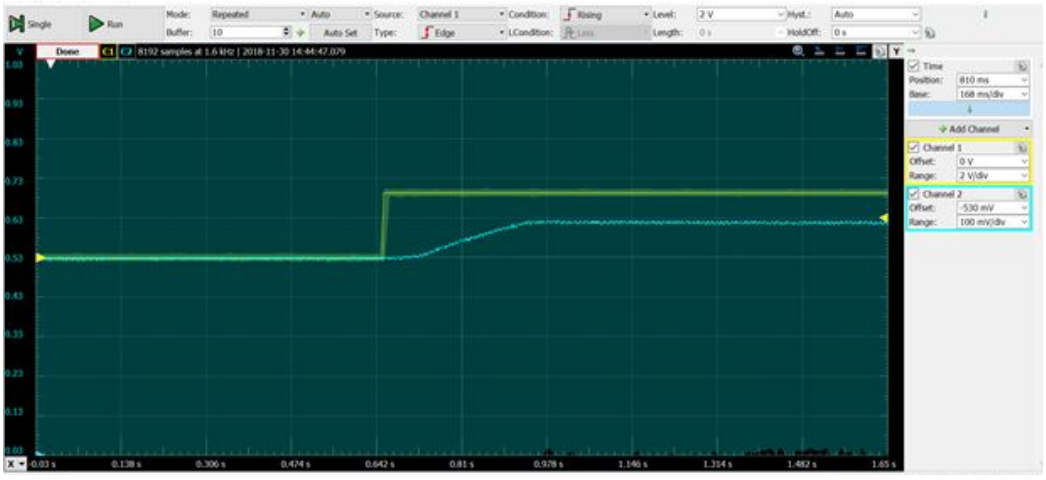
- Minimum Actuation amt: 1900 (V = 0.531)
- Maximum Actuation amt: 2200 (V = 0.628)

Here is the code used to execute the actuation:

```
int main()
{
	uint32_t state =  0;
	int counter = 0;
	int actuationAmt = 1900;//.53v
	initialization();
	while(1)
	{
		if (g_tick_flag == true) //check tick happened ( set true every 10hz)
		{
			en_clutch_lin_act();
			counter++;
			g_tick_flag = false;   //clear tick_flag

			if(counter >= 50) // every 5 seconds toggle LED & Act setpoint
			{
			PF3 ^= 0x08;
			counter = 1;
			if(actuationAmt == 2200)
			actuationAmt = 1900;
			else
			actuationAmt = 2200;//.63v
			}
			moveto_lin_act(actuationAmt);
		}
	}
}
```

Then, using an PC-based oscilloscope, a waveform was collected and used to determine the control parameters. The waveform is displayed in the following image:



This is the exact results intended by the experiment. The slope of the blue line is the actuation voltage produced by the brake pressure sensor inside the vehicle. The calculations for all necessary constants and control parameters are shown in the next page. Once the loop is closed, they will be analyzed and adjusted as needed to produce a smooth braking control loop.

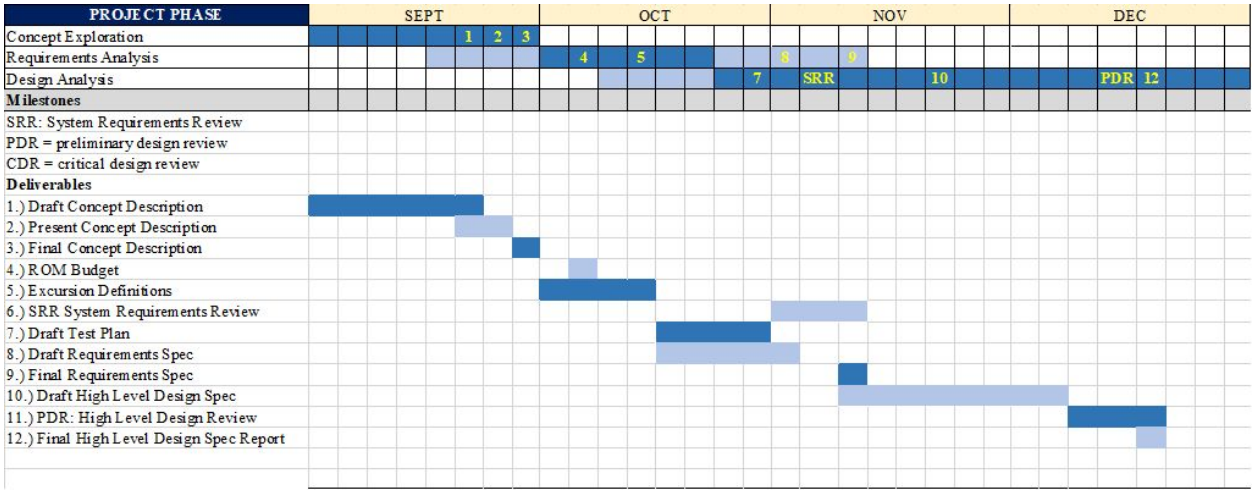
6. BUDGET INFORMATION

Cost Estimate (Version 1, High Level Design Specification, 10 December 2018)

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	TM4C123G XL	$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 (High Level Design Specification) - 8 December 2018



7.1 PROJECT STATUS (8 December 2018)

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. The design analysis phase is in progress as the high level design is being defined in this document. 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					