The Evolution of a Supervisory Controller for a Split Parallel Plug-in Hybrid Electric Vehicle

by

Radhika Kartha

A thesis

presented to the University of Waterloo

in fulfilment of the

thesis requirement for the degree of

Master of Applied Science

in

Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2017

AUTHOR’S DECLARATION

Abstract

The growing global acceptance of alternative fuels and electrified vehicles, accelerated by stricter emission rules, has accentuated the demand for more engineers who are adept in advanced vehicle technologies, particularly in the field of hybrid electric vehicles. Through the EcoCAR 3 challenge, the University of Waterloo Alternative Fuels Team (UWAFT) has the opportunity to produce a split parallel plug-in hybrid electric vehicle (PHEV) that runs on an alternative fuel that has 85% ethanol and 15% gasoline (E85). With the avant-garde design of a snowmobile engine (850cc turbocharged, two-cylinder) with pre and post transmission motors and an 8-speed automatic transmission, there is a need to build an equally innovative hybrid supervisory controller (HSC).

This research focuses on the process followed to build pragmatic controls for its HSC that aims to optimise vehicle performance while ensuring safe operation, lowering emissions, and maximising fuel economy. The question of how an HSC built in a student team environment, which transitions through various drive modes, can ensure safety coverage has not yet been sufficiently addressed. Along with documentation that has traceability between requirements and testing of the HSC code, UWAFT's HSC is intelligently structured for manageability, serviceability and ultimately safety.

This investigation starts with methodologies adopted to ascertain traceability between requirements developed and testing done for validation and verification of the developed HSC code for automotive safety. UWAFT's HSC is a rule-based controller developed through model-based design. Following regression testing in various environments such as SIL and HIL, the integrated HSC code is then tested in vehicle. For the integrated code to be cohesive, the rules for code partitioning to maintain its readability and how the model is managed are discussed.

The preliminary results of the HSC code prove its practicality by showing effective transitions and torque splitting. Thus, by providing an HSC with incorporated safety and intelligent structure for rapid code development, a new benchmark is set for HSC developed within student team environments.

Acknowledgements

The hard work of the Controls team of Year 3, including Cole Powers, Michael Wu, Jake McGrory and myself, has resulted in the code development leading to the fruition of this thesis. Without Cole's daily motivation and the collective readiness of UWAFT's members to support me wherever possible, the journey of writing this thesis would have been very strenuous. The guidance provided by my supervisor Dr Roydon Fraser, not just for this thesis, but for the duration of my time with the team, has been nothing short of priceless. I would also like to acknowledge the invaluable advice provided by Dr Michael Fowler throughout my UWAFT experience.

My gratitude extends to every UWAFT member. The past ones for creating a legacy I could be part of and the present ones for helping me learn so much in just over a year; including the coops, the students who continue to take UWAFT courses, and other volunteers. Thanks will not make the cut for the core members of UWAFT, who are my chosen family. With the competition of EcoCAR 3 being so intertwined with our research, we have had our ups and downs together. Thank you for making the tough days bearable, especially Ramin Shaikhi, Paul McInnis, Patrick Digioacchino, John Catton and Sid Kakodkar. Being UWAFT's Project Manager and having worked along such intrinsically motivated peers will forever be a highlight in my life.

My sincere gratitude towards Argonne National Laboratories (ANL) for managing EcoCAR 3 along with all the sponsors for making this research possible and providing us with this platform that allows us to learn while having fun. Special thanks to Laura Barlow for being our GM mentor and Jane Russwurm (from the Writing Centre) for providing me with valuable writing advice. I am grateful for the boundless inspiration from my mother, Susobhini K. and for the constant encouragement from my father, Rajan Kartha. Thank you Rejeesh Kartha for all the times you dragged me to stare at cars and in turn exposed me to the joy of the automotive world.

Last but not the least, thank you for reading this. A lot of good intention and thought has been put to make it as easy-to-follow as possible. If you are a UWAFT member, I hope this aids to familiarize you with all the amazing tools at UWAFT!

Dedication

To my Ma who kept her chin up and head high through it all.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AFMEA	Algorithm Failure Modes Effects Analysis
ANL	Argonne National Laboratory
APP	Accelerator Pedal Position
ASIL	Automotive Safety Integrity Level
AVTC	Advanced Vehicle Technology Competition
BCM	Battery Control Module
BEV	Battery Electric Vehicle
BPP	Brake Pedal Position
CAN FD	Controller Area Network Flexible Data-rate
CAN	Controller Area Network
CD	Charge Depleting
CDNG	ControlDesk Next Generation
CIL	Component/Processor-in-the-Loop
CR	Collision Resolution
CS	Charge Sustaining
CSMA	Carrier Sense Multiple Access
CSMS	Controls and System Modelling and Simulations
dbc	Database container
DFMEA	Design Failure Modes Effects Analysis
DOE	Department of Energy
DTC	Diagnostic Trouble Codes

ECM	Engine Control Module
ECU	Electronic Control Unit
eLSD	electronic Limited Slip Differential
EPA	Environmental Protection Agency
EPO	Emergency Power Off
ESS	Energy Storage System
FMEA	Failure Modes Effects Analysis
FTA	Failure Tree Analysis
GHG	Greenhouse gases
GM	General Motors
HARA	Hazard Analysis and Risk Assessment
HIL	Hardware-in-the-Loop
HSC	Hybrid Supervisory Controller
I/O	Inputs or Outputs
ICE	Internal Combustion Engine
ID	Identifier
IO	Inputs and Outputs
ISO	International Organisation for Standardisation
LIN	Local Interconnect Network
LSD	Limited Slip Differential
MABx	MicroAutoBox
MAP	Manifold Air Pressure
MIL	Model-in-the-Loop
NDA	Non-Disclosure Agreement

OEM	Original Equipment Manufacturer
PHEV	Plug-in Hybrid Electric Vehicle
PRNDL	Park, Reverse, Neutral, Drive, Low (transmission gear selector positions)
RPN	Risk Priority Number
RTI	Real-time Interface
RTICANMM	Real-time Interface Controller Area Network Multi-message
RWD	Rear Wheel Drive
Rx	Receive
SAE	Society of Automotive Engineers
SIL	Software-in-the-Loop
SOC	State of Charge
TCM	Transmission Control Module
Tx	Transmit
UWAFT	University of Waterloo Alternative Fuels Team
VIL	Vehicle-in-the-Loop
VPA	Vehicle Propulsion Architecture
VPC	Vehicle Propulsion Controller
VTS	Vehicle Technical Specifications

Chapter 1
Introduction

With the energy crisis and marked increase of 1.38 degree Celsius in the overall global temperature since the 19th century, it is important that there is reduced use of petroleum products and reduced production of greenhouse gases (GHG) [1] [2]. According to a recent US Environmental Protection Agency (EPA) report, the transportation sector is the second largest contributor of the total GHG emissions in the US and about half of the 27% of GHG is produced by passenger vehicles [3]. This emphasises the need for the transportation sector to continue using advanced vehicle technologies to produce more electrified vehicles which have been identified as an effective solution.

1.1 Motivation

The interest in electrified vehicles is at an all-time high with governments and automakers making announcements alike that benefit a surge in their production. One of the most ambitious plans for electrified vehicles was announced by France to eradicate conventional vehicles by 2040. This was immediately after Volvo's pledge to only produce hybrid and electric vehicles starting 2019 [4]. Such declarations are a consequence of one of the most transformative and disruptive times that the automotive world is currently in. In contrast to GM's EV-1 that could not make sales a century back, the present global sales of electrified vehicles is 1% with commercialised vehicles like Tesla's Model S, the Nissan Leaf, GM's Volt and Bolt, and BMW's i3, to name a few [5]. As more competent engineers will be needed for this exploding industry, it is important to educate the future engineers with the technologies necessary for electrified vehicles.

An initiative in this direction of educating future engineers in advanced technologies is the Advanced Vehicle Technology Competition (AVTC) series. The University of Waterloo Alternative Fuels Team (UWAFT) competes in the latest AVTC called EcoCAR 3; sponsored by General Motors (GM) and US Department of Energy (DOE). It is a four-year challenge running from 2014 until 2018 (with annual competitions) challenging 16 student teams across North America to convert a 2016 Chevrolet Camaro into a Plug-in Hybrid Electric Vehicle (PHEV). The students are to accomplish this feat by maintaining the iconic Camaro body. Through the use of advanced technologies, teams strive to lower the emissions of the vehicle without compromising the Camaro's legendary performance, safety or consumer acceptability [6]. Since its commencement in 1996, UWAFT has completed five AVTC series and is currently on its sixth. This can be attributed to the unparalleled experiential learning the AVTCs provide. Students are exposed to the latest automobile industry fads as well as standards that are used. The hands-on experience achieved through being part of the team makes members better engineers and team players. Benefits of design-and-build competitions, particularly in effectively teaching engineering concepts has been established in the engineering education world [7].

Specifically, AVTCs provide undergraduate and graduate students the unique opportunity to mimic the automobile industry practices closely. Acknowledging the complexity of the work undertaken by the teams, a lot of guidance and support is provided by the industry sponsors which is assimilated and managed by Argonne National Laboratory (ANL). These efforts result in student teams like UWAFT (shown in Figure 1) being capable of making a vehicle as close to an industry development vehicle as possible [6].

In EcoCAR 3, due to the highly functional and challenging architecture chosen by UWAFT, there is significant Controls and System Modelling and Simulations (CSMS) work. This work involves understanding complex concepts that pose a steep learning curve and culminates in making a successful hybrid supervisory controller (HSC) that is both reliable and safe.

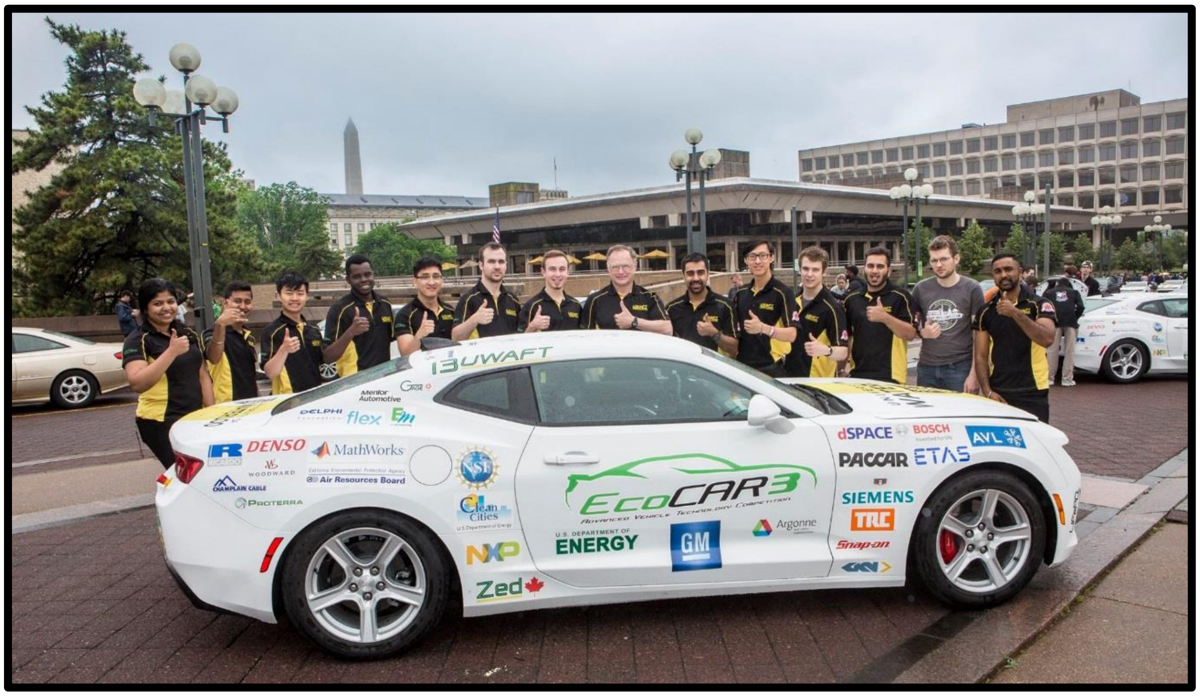


Figure 1: Image of UWAFT outside the US DOE in Year 3 of EcoCAR3

Photographed by Myles Reagan (May 2017)

In this thesis, a simple rule-based controller is proposed and the evolution of this HSC is walked through by providing details of the methodologies used by UWAFT. The intrinsic relationship of controls development and safety as well as the actual code developed are covered, concluding with drive logs exhibiting the practicality of the developed work.

1.2 Thesis Objectives

The main objective of this thesis is to design a hybrid supervisory controller within a student team environment. Through the discussion provided, it meets the following sub-objectives:

(1) Develop pragmatic controls for the HSC with a working torque control strategy and ability to transition through different drive modes
(2) Document the work done by CSMS members for the rule-based HSC with a focus on Year 3 efforts (when the author was part of the team).
(3) To provide new CSMS team members with context of the work done so far for the evolution of the HSC code. By reading this thesis, the new recruits will understand how their contribution fits into the bigger picture of HSC code development and UWAFT as a whole.

With the competition deliverables becoming more and more vehicle centric, there is lesser documentation of the work done by UWAFT for its safety focused HSC. Likewise, experienced CSMS members are running short of time to train new members since the HSC code development takes priority. The additional strain due to the mandatory coop program within the Faculty of Engineering at the University of Waterloo results in lesser time for knowledge transfer, training or documentation. Addressing these challenges, this thesis also serves to be a training tool for new recruits and future student teams working with an HSC.

By keeping the audience in mind, parts of the thesis are kept instructional and simple [1] in order to aid new members. In order to make sense of the different teams mentioned throughout the thesis, UWAFT's organizational structure that existed in Year 3 is provided in Appendix A.

1.3 Thesis Outline

Chapter 1 establishes the research territory proposed by showing the relevance of educating new engineers and introducing the EcoCAR 3 competition.

[1] Italicised text refers to the names provided by UWAFT. This could be for the different subsystems in the code or signals used at UWAFT. The blocks that are generic to Simulink [43] are presented in single quotation marks to assist students who are not familiar with the software.

Chapter 2 involves apposite literature review and the processes used to establish automotive safety within the software development of HSC. Also, briefly discussed are the topics of CAN, the need for an HSC and the testing platforms available for evaluation of the HSC code for proposed safety requirements.

Chapter 3 details UWAFT-specific processes by first describing the UWAFT PHEV and its available drive modes, followed by explaining the methodology used to ascertain that the HSC meets proposed safety requirements through testing of code produced and how maintaining traceability between all safety documentation used is established. This chapter is concluded with a discussion on achieving baseline functionality of components followed by diagnostics and remediation for safety.

Chapter 4 provides an overview of the HSC code development after establishing the rules for managing the code and the tools used to do the same. Subsequently, parts of the code are described with particular attention given to how the torque control strategy was developed and the partitioning that helped in building the code that incorporates safety.

Chapter 5 displays the drive logs from on-road testing for validation of the HSC code developed. It also gives a quick introduction to the tools used to interface CAN for monitoring and logging data.

Chapter 6 presents the conclusions from the thesis and proposes some recommendations and suggestions for future work.

Chapter 2
Background and Literature Review

The sales of electric vehicles (includes BEVs and PHEVs) in Canada surpassed 32,000 in total by March of this year [8] [9]. On assessing the automotive trends both nationally as seen in Figure 2 and internationally, it is very promising for PHEVs, and providing hands-on experience to the next generation of automotive engineers is important.

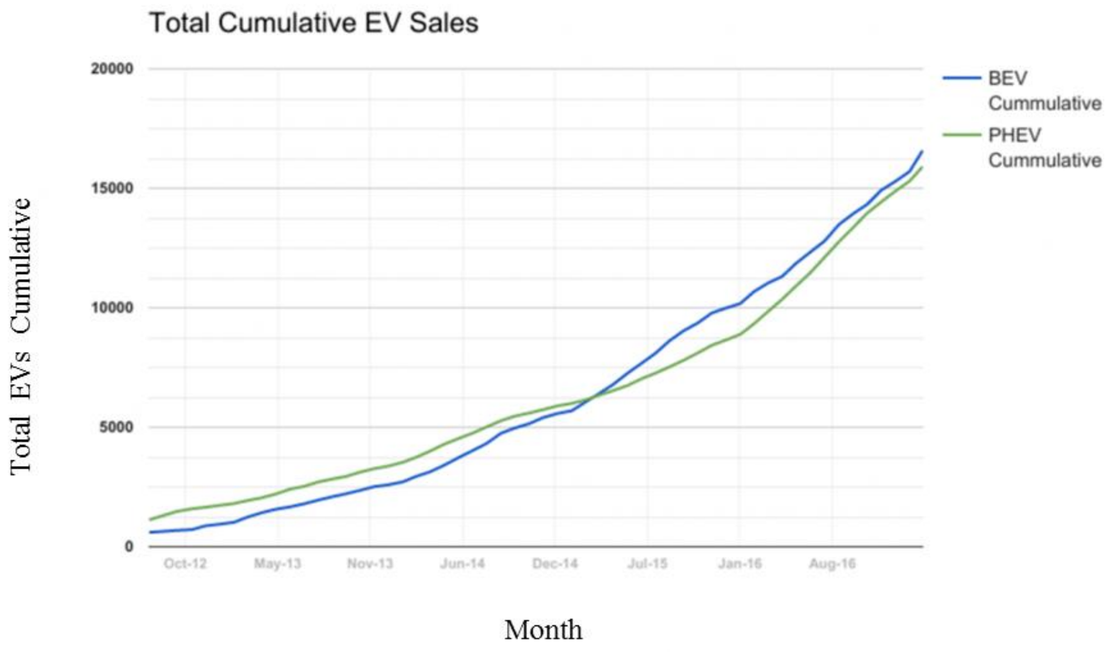


Figure 2: Graphs showing the rising sales in PHEVs and BEVs in Canada [8]

There has been sufficient research done in categorising the various hybrid electric vehicles and Alexander Koch’s master’s thesis is a great starting point for being introduced to the classification scheme [10]. However, this thesis is looking at a practical rule-based controller developed in a student setting and hence reviewed available sources for the HSC. There have been innumerable publications in the form of papers, presentations and theses that have been developed as a direct result of AVTCs. On reviewing above 30 SAE papers produced by the different student teams, over the course of three AVTCs (EcoCAR series), it can be seen that most of them are related to modelling and controls work. This can be through discussion of the architecture selection design process through comparison [11] [12] [13] or elaboration of the selected architecture with the decision-making reasons [14] [15] [16]. Some of them discuss strategies used for controlling a component [17] or compound all the components to discuss vehicle strategy [18] [19]. The papers that summarise the implementation of the work done for the various AVTC briefly mentions the controls and modelling efforts along with the mechanical integration [20] [21]. Similarly, the theses produced from two student teams (UWAFT and Virginia Tech) which were affiliated to AVTC work was examined. These two were chosen since they have been amongst those participating in the AVTCs for the longest duration. The theme of the theses followed those in the papers. Although, there were publications that discussed the controls development using model-based design, none of them discussed how the safety aspect of the controls work developed was ensured. Some papers mentioned methods like Design of Failure Modes Effects Analysis (DFMEA) and Fault Tree Analysis (FTA). One thesis in particular, by Michael Giannikouris, elaborated these methodologies, but did not sufficiently show how the student team ensured the necessary safety coverage [22].

This thesis elaborates on how UWAFT manages to develop safety requirements for the HSC development that help prioritise and ensure test coverage. The framework of the safety documentation and test cases used to implement and keep the development work updated is provided. This documentation also helps with planning the work that can be done. Since the HSC code development relies on a number of people, how it is partitioned for rapid code development without compromising safety is also discussed. The following sections in this chapter sufficiently introduce the background knowledge required to understand the subsequent chapters.

2.1 Need for Hybrid Supervisory Controller (HSC)

In a conventional vehicle with an internal combustion engine (ICE), the driver torque request, which is interpreted in the form of accelerator pedal position, is mapped to the only torque-producing component in the vehicle – the engine. With hybrid vehicles that have more than one torque-producing component, this singular mapping is no longer possible. There is a requirement for a controller that can intelligently and efficiently decide how the torque request must be split between the available torque-producing components (for example, engine and electric motors) to deliver requested resultant torque at the wheels. Figure 3 provides a partial representation of the control modules that interact with the HSC for the algorithm used to produce the required torque at the wheels.

The component that decides how the torque request will be met is called a Hybrid Supervisory Controller (HSC). Along with this function, the HSC is tasked to monitor the vehicle as a whole. This means that faults and failures from the different lower-level controllers such as the Engine Control Module (ECM) and Battery Control Module (BCM) needs to be detected and mitigated, if necessary.

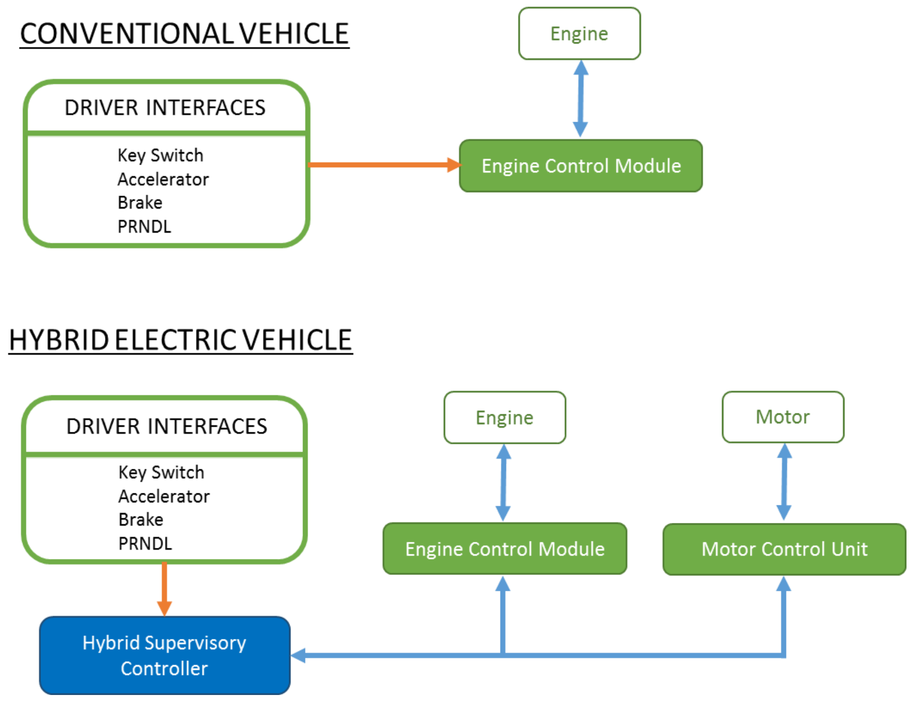


Figure 3: Torque request control flow in conventional vs hybrid vehicle

The mitigation decision will rely on the degree of importance of the component (or sub-system) and fault detected as well as the vehicle's overall condition. The degree of importance ties to the degree of safety related with the fault and/or component as is identified by the risk assessment detailed in section 2.2. The HSC interprets the different driver inputs such as accelerator pedal position (APP), brake pedal position (BPP) and gear shifter (PRNDL) position at the very minimum. Additionally, in student environments the HSC doubles as controller for actuating components that lack dedicated controllers. Evidently, the HSC is a critical component in a hybrid vehicle and it is essential that it is rigorously evaluated and reviewed before being deemed safe to be used in a vehicle.

2.2 Automotive Safety

In order to ensure the safe operation of a vehicle, risk assessments and other analyses of hazards must be conducted. Once these have been completed for the different vehicle systems and the hazards identified, the mitigation strategies can be found. Concurrently, the requirements that need to be met by the systems including the ones for HSC can be determined. In the following sections, the different methodologies used to identify failures and faults have been discussed.

2.2.1 Design Failure Modes Effects Analysis (DFMEA)

DFMEA is a fundamental tool in the design of vehicle systems. It is an industry wide standard that is applied to all types of systems - mechanical, electrical, and software. When used early in the design process before the hardware is built, it helps in identifying potential failure modes and can be designed out of the system where possible. Thus, it guides in the selection of the design that has least failure modes considering the amount of information available at that initial stage. Failure Modes Effect Analysis (FMEA) is continued throughout the vehicle development process so that any changes made later on can be reassessed to find the impacts of failure (risk assessment). When these are identified and are no longer avoidable by design, detection and mitigation strategies are defined and developed for corrective action. Generally, risk priority number (RPN) is used to guide the design of appropriate diagnostics. An example of DFMEA is provided in Appendix C.

Severity rating refers to the extent of how damaging a failure can be. Occurrence rating, also known as, risk of occurrence is defined as the probability of the incident that is being assessed actually occurring. Detection rating refers the probability or extent to which the incident of the failure mode can be determined. The RPN can then be calculated by using equation 1.

$$\boldsymbol{RPN} = \begin{pmatrix} Severity \\ Rating \end{pmatrix} * \begin{pmatrix} Occurrence \\ Rating \end{pmatrix} * \begin{pmatrix} Detection \\ Rating \end{pmatrix} \quad \text{Equation 1}$$

These numbers can be used to identify which assessments are more critical and appropriately more diagnostic techniques can be developed for those incidents for redundancy. Higher the RPN, higher the risk of failure and hence they are proportionally prioritized for implementing corrective actions.

DFMEA is a bottom-up inductive analysis of risk. An inductive analysis is one where the causes are known but the effects are unknown. Assignment of scores for detectability, severity and occurrence needs to be consistent for the RPN to be useful. Rules for the assessment need to be developed and documented by the team so that comparisons between designs are possible. And again, this consistency will help in prioritising development of diagnosis and remediation (See section 3.6). Using FMEA improves overall cross-functional communication and learning. Due to the documentation done during FMEA, it can double up as a space for documenting lessons learned. Proactive risk assessment is encouraged when routine FMEA is conducted. System verification test plan (See section 3.3.3) that has test cases that need to be run for a component or feature needs to have DFMEA as a critical input so that there is a link to the remedial actions in the event of a failure. Software generally does not tend to “break”, however, there is a possibility of hardware failure or for the system to crash. Some teams employ Algorithm Failure Mode Effects Analysis (AFMEA) to deal with software development failures [23].

2.2.2 Fault Tree Analysis (FTA)

Fault tree analysis is a top-down approach that analyses an undesirable top event and traces it all the way to lower level component failure. All the in between steps are documented in order for the team to identify if there are different faults at a system level that can be boiled to a single point of failure. It also helps to see multiple sources of failure that can initiate other faults in the system. Hence, detection and mitigation of these failures are prioritised. This is a deductive analysis of risk since the causes are unknown but the effects are known. FTA provides a logical framework and estimates of the probability of undesired events occurring using Boolean algebra. There is documentation of all the failure scenarios that have been considered. It supports engineering and management decisions since it is a visual tool and can help identify critical paths. The FTA tool can be used for optimizing redundancies, inserting diagnostic algorithms, and identifying proper fault provisions [23].

2.2.3 Hazard Analysis and Risk Assessment (HARA)

Hazard Analysis and Risk Assessment (HARA) is a technique that aids in identifying hazards associated with a particular job, operation, or task before they occur. The analysis focuses on the relationship between the worker, the task, the tools, and the work environment. Upon identification, the hazards are given an appropriate risk rating and subsequent steps are taken to eliminate or reduce them to an acceptable risk level [24]. When applied to the automotive context, proper safety protocols need to be determined for possible scenarios of hazards. However, defining these requires experienced personnel since there are chances of hazards being missed due to lack of prior exposure. In a student setting, there is a risk of underestimation of hazard and probability of failure. This can be reduced by doing extensive literature review and research to maximise understanding of the operation of each component in the vehicle and possible interactions. Any hazard that has not been identified will not pass through the scrutiny subject to identified hazards. The process of performing HARA is represented in Figure 4. By using this engineering tool a thorough assessment of the possible hazards and their associated risks can be developed. This will prompt alternative design solutions to mitigate or control the risks to an acceptable level. Until this acceptable level is found, the process is repeated.

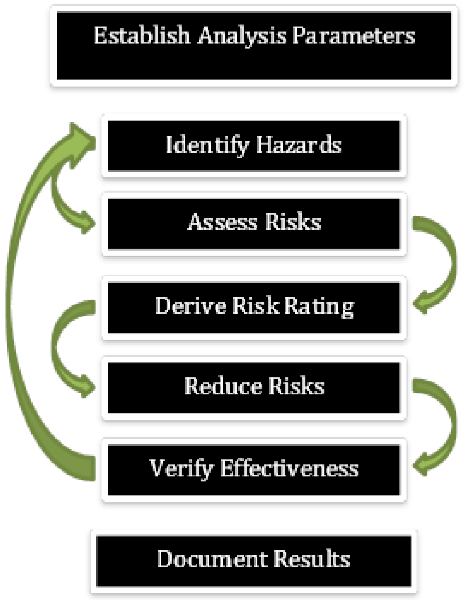


Figure 4: Flowchart for HARA

2.2.4 Automotive Safety Integrity Level (ASIL)

ASIL is a risk analysis methodology defined by ISO 26262 -Functional Safety for Road Vehicles standard. This can be used as the step for deriving risk rating in HARA as shown in Figure 4. Severity, Exposure and Controllability are used as parameters to do the hazard analysis and risk assessment to define the strategies to be used to keep the passengers and pedestrians safe. These parameter specifications have been provided in Table 1. Their product helps find ASIL rating.

Table 1: ASIL parameter specification rating

Parameter	Limit	Description
Exposure	E0	Incredibly unlikely
	E1	Very low probability (injury could happen only in rare operating conditions)
	E2	Low probability
	E3	Medium probability
	E4	High probability (injury could happen under most operating conditions)
Severity	S0	No Injuries
	S1	Light to moderate injuries
	S2	Severe to life-threatening (survival probable) injuries
	S3	Life-threatening (survival uncertain) to fatal injuries
Controllability	C0	Controllable in general
	C1	Simply controllable
	C2	Normally controllable (most drivers could act to prevent injury)
	C3	Difficult to control or uncontrollable

ASIL A to ASIL D are the four levels of standards identified by this risk assessment with ASIL D equating to highest integrity requirements and diagnostic capabilities and ASIL A with the lowest as shown in Table 2. ASIL D generally means that the occurrence of this potential hazard could result in fatal injuries. ASIL is different from risk because it takes into account how much a typical driver can react to prevent injury in the event the hazard occurs whereas risk only looks at the probability of the hazard occurring and its impact. Depending on the ASIL, remedial actions such as propulsion shutdown, decreased functionality and doing diagnostics to detect faults and mitigate them appropriately are followed [25] [26].

Table 2: ASIL rating levels

Risk of Hazard	**LOW** $\longrightarrow$			**HIG**
Rating	**A**	**B**	**C**	**D**
ASIL Range	**(1-19)**	**(20-39)**	**(40-59)**	**(60-80)**

2.3 Software Development

2.3.1 Development process layouts

The V-model shown below in Figure 5, also known as verification and validation model, is a derivation of the waterfall model. It represents the methodology used for general software development plan and although represented with a 'V', it is executed in a sequential fashion. This is represented by the direction around the 'V' in Figure 5 that moves linearly.

In a typical software project, the initial phase consists of customers providing their requirement which is followed by a back and forth to understand the exact requirements that are designed in detail so that the coders can be provided with exact details of expectations from system and software. During this phase of detailed design, evaluations of what requirements can be met and expectations are set. During implementation, the code is written and then subsequently tested (isolated testing followed by integrated testing and system level testing).

Although linear, the model is provided with a V to visually represent how the testing phases are closely related to the initial phases of development. This V-model is adopted and used in different mannerisms throughout this thesis. Testers are trying to implement a new methodology and consider the V-model archaic. However, the V-model is sufficiently simple to understand and implement for student teams [27].

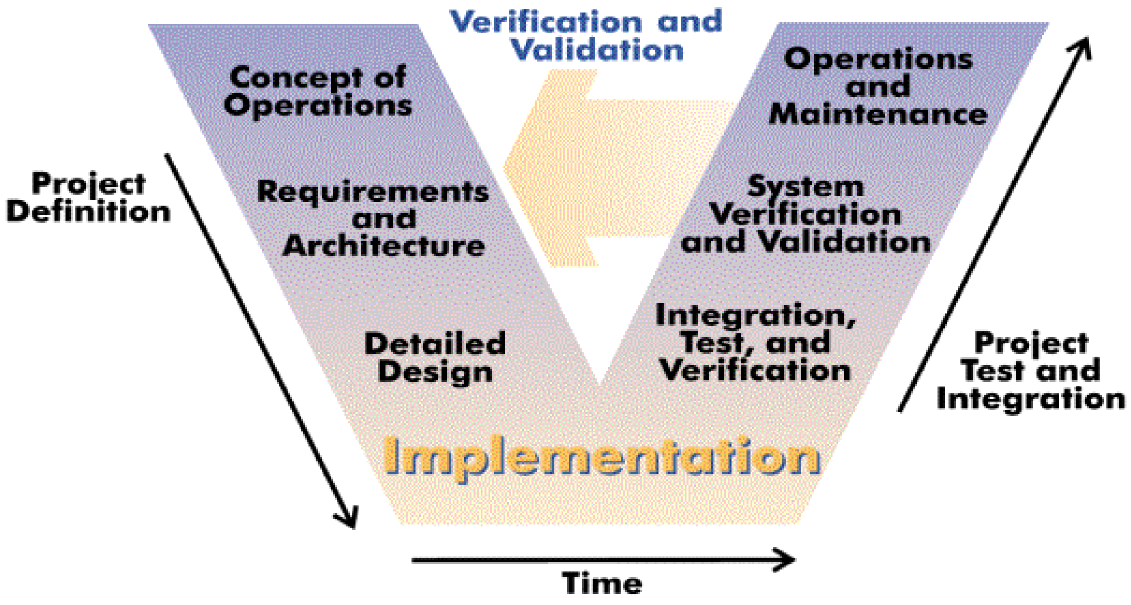

Verification and Validation
Concept of Operations
Requirements and Architecture
Detailed Design
Implementation
Integration, Test, and Verification
System Verification and Validation
Operations and Maintenance
Project Definition
Project Test and Integration
Time

Figure 5: V-model software development [28]

2.3.2 Defining model requirements

Following the selection of vehicle architecture, the hardware and software requirements for the different components (or subsystems) is defined. This is so that the components and the controllers can be virtually represented so that subsequently the HSC requirements can be found and code developed appropriately. By making the component models and soft ECUs more realistic, the full vehicle model can produce more reliable results with the simulations that are run.

Hardware requirements like how the component is powered and the types of inputs and outputs involved to establish communication with the HSC is researched along with consolidating as much information as possible to adequately ascertain safety requirements. After that, software requirements like which CAN signals are needed (if at all) or other analog and digital signals needed along with information on start-up sequences for control algorithms to be written are collected and documented. This is done by defining a set of inputs and outputs for a particular function of the investigated component as described in Table 3 shown below.

Table 3: Examples for how input and output signals are defined

DEFINITION OF INPUT/OUTPUT	Examples
Type of variable	Unsigned, integer, double

Range of variable values	Lower and upper bound, cannot be zero if used as dividend
Output range	Usually dependent on input so provide acceptable values for corresponding inputs
Units for variable	N, m/s
Signal nomenclature	Engine_Speed[RPM]

The function of the component is then broken into testable smaller sub-functions and requirements are created. For each requirement, corresponding code is developed, tested and documented. The methodology of detailed documentation for both requirements and tests is time-consuming yet very effective. Moreover, the benefits of well documented software development are manifold, especially for troubleshooting when the code breaks during system level interactions; following integration of piece codes that passed tests for lower level requirements. When new problems are identified in the light of new information obtained or due to behavioural differences at a system level, requirements are written for those as well and the iterative process of code development follows.

In addition to mimicking the methodology used in the automotive industry, this process of finding the software requirements is an effective way for students to engage in thinking what is being worked on before the actual work begins. Also, executing test cases is a good training tool for on-boarding new CSMS members. Although different from component model requirements, controls testing requirements follow similar methodology for documentation.

All documentation should be written in simple language that is grammatically correct and with the right terminology. It is best if the statements are not vague and anybody with rudimentary knowledge can follow test procedures or write code after reading requirements. When an easy and organized methodology for traceability exists between requirements and test cases, it is possible to note which higher level requirement is met once all the lower level requirement is successfully verified and validated for the safe operation of the vehicle.

2.3.3 Testing platforms

The pragmatic controls being developed for the car are tested for safety and reliability using several environments. It is not necessary that the following test platforms have tests performed sequentially. In fact, they are bidirectional. It is also not necessary that all tests pass through every single environment before it can be tested in the vehicle. The platforms that different tests need to be run in is decided by the teams such that they sufficiently meet the requirements and provide enough information about their system-level interactions. This is done to prevent as many damaging hazards during VIL tests as possible. It is advisable to pass at least through one other platform of testing before using it on the on-board HSC.

Model-In-the-Loop (MIL)

This is usually used only to test single algorithms and not for the full vehicle model. It usually contains idealized versions of the models and may contain interfaces that do not exist in reality. There is no real-time signal being sent and the entire model is in one single computer. Hence, the computer's memory and processor speed limits the simulation fidelity similar to the case in SIL below. MIL is suitable for testing enabling sequences to components/subsystems before it is sent over physical signals so that the algorithms can be fixed. MIL does not target to test integrated HSC code.

Software-In-the-Loop (SIL)

In this closed-loop environment the entire model is in one computer but contains the full vehicle model which is compiled to an executable. SIL is used to test the HSC code that has fault detection and mitigation strategies. The signals routed exist in the actual vehicle and are in engineering units. The simulation time can be sped up or slowed down as is considered beneficial. To use SIL effectively, a thorough understanding of the inputs and outputs is expected. Tests such as sending signals that are out of range is performed in SIL (as well as HIL) to check the effects on a system level. It is possible to use this environment for automated testing.

Hardware-In-the-Loop (HIL)

This environment consists of signals that are sent real-time and has physical interfaces for the simulated components. Due to the existence of physical interfaces, it is possible to generate electrical faults in this testing environment such as shorts and loss of communication. Communication in HIL environment accounts for latency and makes evident any physical lag and other bus dynamics not evident from virtual testing. Latency is the amount of time it takes for an application to respond to a request. HIL is capable of carrying out most of the validation and verification necessitated by the test cases. However, there are things like the need to make separate mounts and other setups that can hinder it from being the preferred choice. Also, training to configure and use the HIL effectively is quite challenging. If the HIL simulator is used to simulate the vehicle model, it will interface with actual HSC hardware for tests. However, it is not necessary for the HIL to run the entire vehicle model and can run pieces of code in instances like when baseline functionality of the components is to be achieved before system-level testing can be conducted. In turn, the HIL simulator can also be used to test different control systems. This process is called rapid prototyping and since actual components are used, it helps in accurate code development. Typically, real-time simulation and using real environment to run tests results in better tuned controllers. Similar to SIL, HIL is also suitable for automated and regression testing.

Component/Processor-In-the-Loop (CIL)

Using the HIL, 12V is supplied to the component and/or controller in an attempt to establish initial communication and for achieving baseline functionality. The HIL now simulates the HSC and by using constant blocks, different command messages are provided until the controller starts transmitting CAN messages. If the right documentation is not provided for enabling the component, trial and error method or engineering intuition is used.

Vehicle-In-the-Loop (VIL)

The actual harness in the vehicle is used in this testing environment when all the components have been integrated into the car. Generally, the vehicle itself is used to rerun all the system-level tests to ensure that all the tests are passed. This makes it possible for testing noise and vibrations along with other drive quality parameters which cannot be simulated in HIL testing. This is also the final level of testing and since the code is intended to work safely in the vehicle, it is the most crucial. Generally, VIL tests start in controlled environments within a facility (such as on the lift or rolling dynamometer), then are taken to low speed closed course spaces (could be deserted parking lots marked by pylons for safety), followed by high speed testing tracks with no other traffic before it is taken on the public roads. This final step is accompanied with a requirement of certification post safety inspection from appropriate personnel.

2.4 In-vehicle communication

For the safe communication between electronic control units (ECUs) in a vehicle, different automotive networks using embedded systems have developed over the years. The point-to-point communication method from the earlier days had to be replaced because the number of ECUs in a car began increasing which resulted in more wires than could be managed and also resulted in increased weight of the vehicle. The solution found was multiplexed systems and the specific requirements for the various car domains resulting in networks like CAN, CAN FD, LIN, and FlexRay to name a few due to the different protocols that were developed. These different protocols developed are standardised by SAE and ISO. Present day vehicles may use a combination of the available automotive networks to meet their requirements. However, CAN is the most widely used communication network in vehicles due to their high bandwidth, low cost and robustness. It is detailed in the next section.

2.4.1 Controller Area Network (CAN)

CAN is a serial communication protocol developed by Bosch that supports distributed real-time control and multiplexing with high level of safety and signal integrity. Most of the vehicle data communication is passed via CAN bus and when ECUs and other modules like data loggers that meet the termination requirements of 120 ohms are placed on that network, they are called nodes of that CAN bus. In Figure 6, the nodes are represented as SG1 to SGn. These nodes are allowed to transmit and/or receive information on these serial data networks. Although this methodology increases network loading, it enables configuration flexibility and real-time network access. If a node needs to be added to an existing CAN channel, as long as it follows the termination requirements and can transmit and/or receive at set network speed following the protocol, it can be physically added to the network.

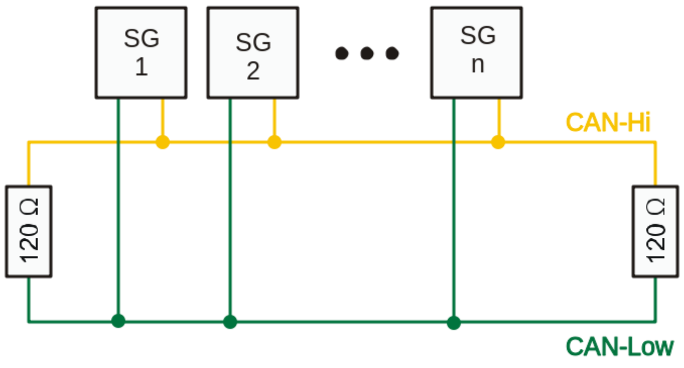


Figure 6: CAN linear topology with terminators [29]

CAN communication is message-centric and not node-centric. What this means is that there is no more important node (multi-master architecture), but prioritization of messages exists so that when different frames need to be sent on the same channel simultaneously, the more important one is given access. This is done using identifiers (IDs) which could be 11-bits or 29-bits (extended IDs). The frame with higher priority is allowed to occupy the CAN bus and the node with the lower priority, sensing that the communication medium is occupied, refrains from sending its data. This method is called Carrier Sense Multiple Access / Collision Resolution (CSMA / CR). These IDs also help nodes with a filtering process to decide if they want to be a receiver or not. The protocol for CAN is ISO 11896 which has developed several extensions over the years. The standard CAN is represented as CAN 2.0 and has two versions of the protocol depending on the ID size: CAN 2.0A for 11-bit ID and CAN 2.0B for 29-bit ID. Generally, CAN 2.0A is sufficient for all the communication in a typical vehicle (number of frames is less than 2^{11}).

Present day cars generally make use of class B and C networks (as defined by SAE) for CAN communication. The de-facto twisted pair of wires is generally for class C CAN that allows for data transmission by differential voltage transmission. They have transmission rates of 250 kbps – 1Mbps and typically connect to high performance, safety-critical control modules in the powertrain and chassis domains. The two wires are twisted to minimize magnetic field effects and have termination requirements of 120 ohms. The class B network for CAN uses single wire and is of low speed with a transmission rate of 30 – 125 kbps. Low performance, non-critical devices such as mirror adjusters and seatbelts are used in this network. With a single wire, there is no resistor and the voltage sent is relative to the vehicle ground [30] [31].

Chapter 3

UWAFT PHEV and Controls Development

3.1 UWAFT Vehicle Architecture

In Year 1 when UWAFT was tasked with selecting an architecture to meet the competition goals, Autonomie models which were validated by the sponsors were used as a starting point for vehicle modelling. Autonomie is a GUI wrapper on Simulink that is used for basic effort and flow modelling. For baseline vehicle modelling, the donated models of powertrain components that was developed by Argonne National Laboratory were put together and simulations run for different architectures by the Systems Modelling and Simulations team. This provided UWAFT with insight for the preferred architectures and the priority of preference. Meanwhile, the Controls team trained to employ model-based design for controls development of soft ECUs and the Hybrid Supervisory Controller (HSC); for the different architectures being simulated. Refer Appendix A and Appendix B for further information about the architecture selection process.

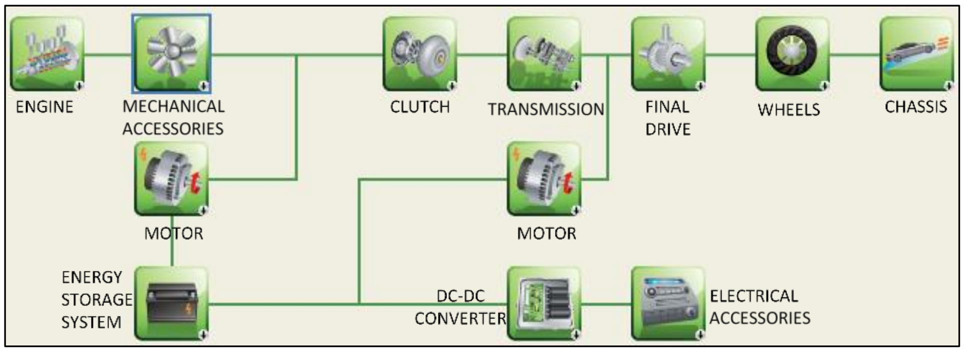


Figure 7: Basic effort and flow model of UWAFT's split-parallel architecture

Through consumer market research, UWAFT identified its target market. This meant that UWAFT found the niche UWAFT's PHEV would cater to. It was scoped to be young, urban professionals who are tech savvy, subconscious environmentalists and are inspired by the legacy that the Camaro represents. Although the target market was not immediately recognized as a demographic that looked at fuel economy as an important factor, they are interested in vehicles that produce lesser emissions. Fuel economy is one of the most important Vehicle Technical Specification (VTS) for UWAFT. The VTS is provided in Appendix B for reference. Following architecture approval, UWAFT began re-engineering the 2016 Chevrolet Camaro into a PHEV with a split-parallel architecture. The basic effort and flow model of the UWAFT vehicle is shown in Figure 7.

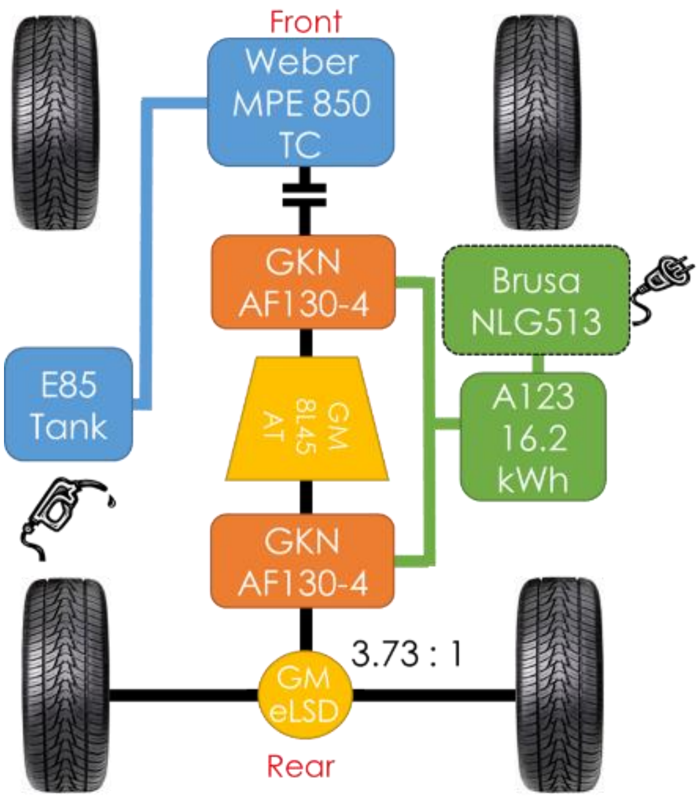


Figure 8: Diagram showing UWAFT's powertrain architecture

The diagrammatic powertrain layout shown in Figure 8 consists of a Weber MPE 850cc turbo-charged engine typically featured in snowmobiles, two GKN AF130-4 motors, and a 16.2 kWh A123 Energy Storage System (ESS). One motor is located pre-transmission (designated P2) while the other motor is located post-transmission (designated P3). The transmission is a stock component and is an 8-speed automatic transmission. UWAFT integrated a custom designed racing clutch between the engine and P2 motor to allow for more flexibility in the powertrain architecture. This chosen architecture allows for a series, parallel, or all-electric mode of operation, enabling the control system to select the mode which is most efficient given the drive profile or performance demand [32]. The UWAFT PHEV was maintained as a rear wheel drive (RWD) but the stock limited slip differential (LSD) was replaced with an electronic limited-slip differential (eLSD) from the Cadillac ATS-V which has a higher gear ratio; stock 2.77 to UWAFT 3.73. The engine runs on an alternative fuel of E85 (composed of 85% ethanol and 15% gasoline) which contributes to lesser greenhouse gas emissions than the conventional gasoline. Brusa NLG513 is the on-board charger for the ESS that has level one and two (J1772 compliant) charging by plugging capabilities [33]. Further information about the architecture is given in Appendix B.

3.2 Vehicle Modes and Operations

During EcoCAR 3 evaluations and competitions, the UWAFT PHEV is tested with respect to two high level control strategies or vehicle modes - Charge Depleting (CD) or blended mode and Charge Sustaining (CS) mode. This does not mean that the UWAFT PHEV is limited to just two modes of operation. For clarity, the numerous operation modes (called drive modes in the code and in sections ahead) with respect to the two high-level vehicle modes have been explained in detail. Understanding the modes of operation helps to better associate the drive mode names (provided in italics) used in the HSC code and their associated constraints and conditions. Depending on factors like ESS state of charge (SOC) and torque demanded, a drive mode is selected for vehicle operation by the HSC.

3.2.1 Charge Depleting Mode

When energy for propulsion of a vehicle relies on the ESS (battery pack), it is said to be in CD mode. If an engine is used to augment performance during very high torque demands, this form of charge depletion is called blended mode. In CD mode, the SOC on average decreases. The UWAFT PHEV can operate in three drive modes within the CD mode –

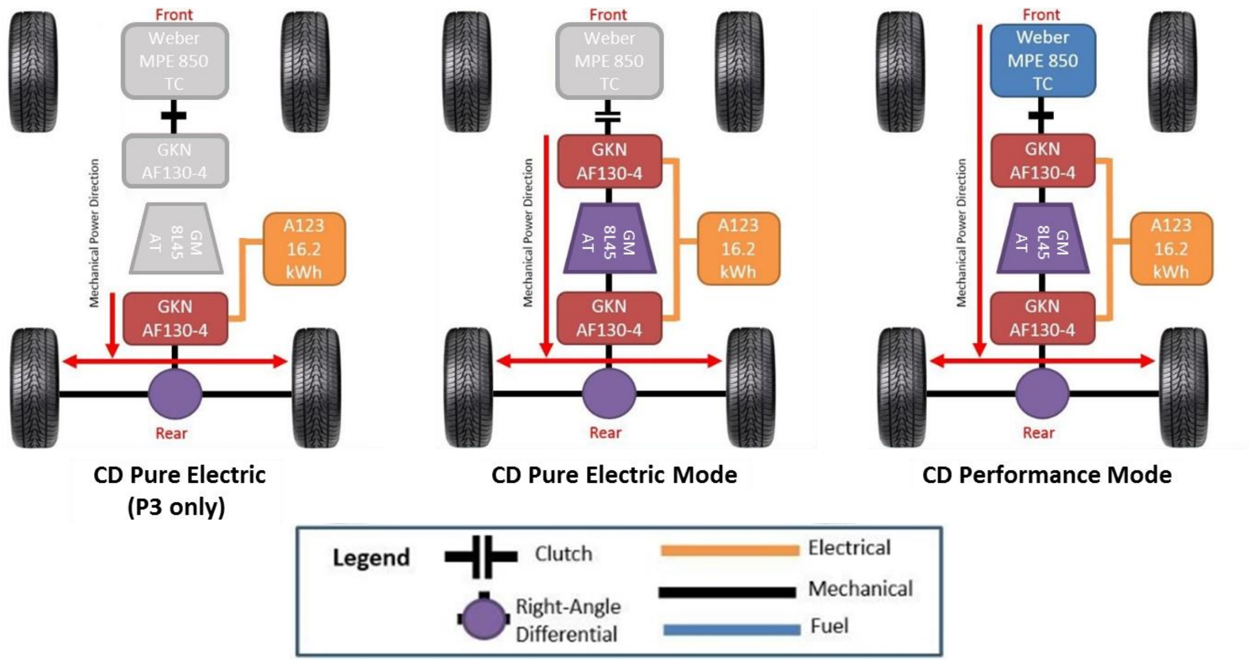


Figure 9: Power flow of charge depleting drive modes

(1) CD Pure Electric: all-electric operation using only P3 motor (*SingleEV or P3Only*)
(2) CD Pure Electric: all-electric operation using both P2 and P3 motors (*FullEV*)
(3) CD Performance: blended mode (*FullDrive*)

The UWAFT PHEV is capable of all-electric operations when only one or both the motors are used for meeting the torque requirements. The CD mode using only P3 motor is operated when the SOC is high and the P3 motor alone can meet the drive cycle requirements. It is also used when the vehicle is placed in reverse. When the racing clutch is disengaged, it allows the engine to be decoupled from the rest of the powertrain and allows both the motors to power the wheels.

However, if the motor duo cannot achieve the torque requirements, the clutch is engaged so that the engine can work in conjunction with the motors to power the wheels. This blended mode of operation is called the performance mode of the UWAFT vehicle. It has been previously established using simulations that the UWAFT vehicle is capable of 232kW in performance mode and can hence meet all the drive cycles and requirements necessitated by UWAFT targets and EcoCAR 3 competition. (The names of these drive modes in the HSC code have been provided in brackets above). The power flows in the different drive modes possible in CD have been shown in Figure 9 using red arrows.

3.2.2 Charge Sustaining Mode

PHEVs in general switch to a CS mode on reaching a certain SOC so that on average it can maintain the said SOC even while driving. With the UWAFT PHEV, CS mode is achieved with the help of the engine that can be mechanically coupled to the rest of the powertrain with the help of the integrated racing clutch. There are three drive modes that can be operated within CS mode –

(1) CS Series: Series operation (*P3 Hybrid*)
(2) CS Performance: Parallel operation (*FullDrive*)
(3) CS Pure Engine: Engine only (*EngineOnly*)

In CS series mode, the transmission is shifted into neutral to work like a clutch that disengages pre-transmission from the rest of the driveline, allowing the engine and P2 motor to work as a generator set. Due to overlapping sections on efficiency curves for the Weber engine and the AF130 motors, maximum efficiency while generating electricity to sustain the SOC is desired.

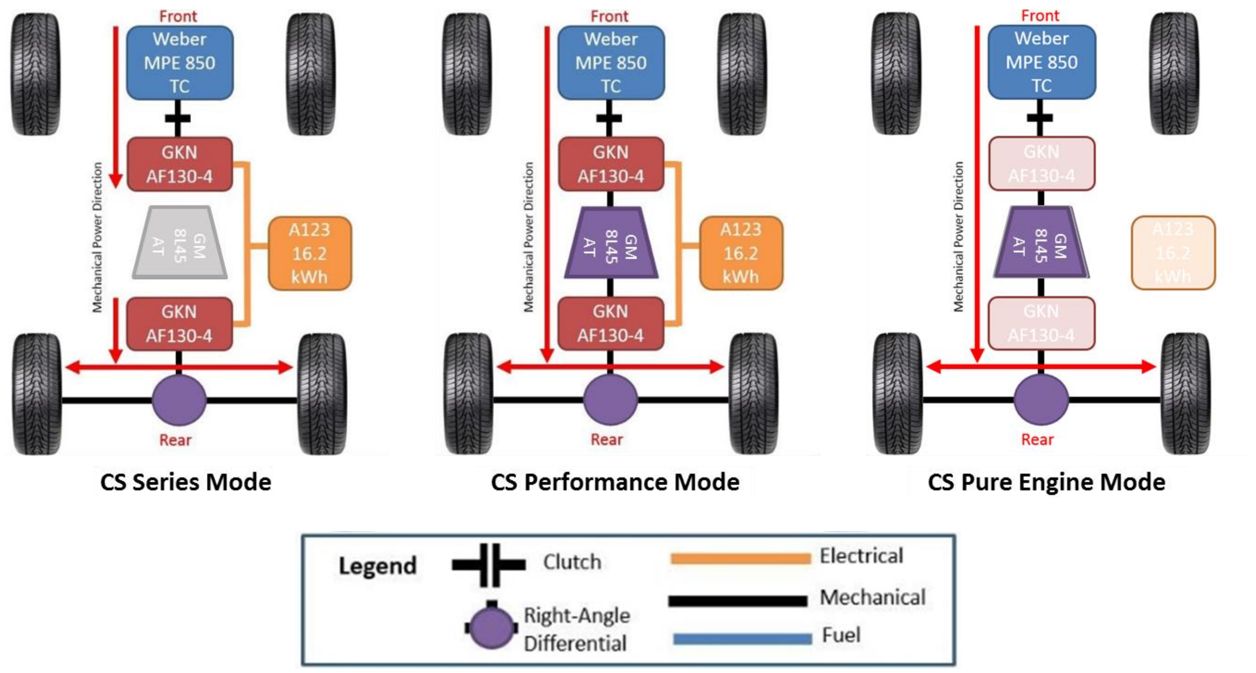


Figure 10: Power flow of charge sustaining drive modes

In series operation, only the P3 motor drives the wheels. If P3 cannot provide sufficient torque, the vehicle will go into a parallel mode. This mode of operation is similar to CD performance mode since the transmission gets engaged to an optimal gear and all the torque producing components are capable of driving the wheels. However, it differs since a portion of the supplied torque from engine is used for power generation to sustain the SOC. This is possible because the GKN motors used are four-quadrant motor-generators. When all the torque produced by the engine (run at its most efficient speed) is not required at the wheels, the P2 motor applies a slight braking torque (by turning backwards in simple terms) to charge the ESS from the surplus power.

In the engine only CS mode, the vehicle functions as a conventional vehicle. This drive mode can be used when the electric powertrain fails. The explained CS drive modes have their power flows represented in Figure 10 to facilitate understanding.

3.3 UWAFT Software development

Although a great starting point, UWAFT had to later migrate to Simulink from Autonomie in order to further develop the vehicle model and the vehicle propulsion controller (VPC) to make it more realistic. This also reduced the time needed for compiling the model, allowed for testing with the help of MATLAB scripts, using RTICANMM blocks for testing on the Hardware-In-the-Loop (HIL) platform and using other features of the software [34].

As briefly discussed in section 2.3.2, soft ECUs and component plant models are developed so that the vehicle functionality at a system level can be simulated and analysed. At UWAFT, the powertrain components were prioritised, followed by other safety-critical components and features. These models would be used to test the developed HSC code.

To recap, the HSC code is developed from the HSC requirements which are defined through processes explained in automotive safety (section 2.2) and further refined with the help of plant model and soft ECU requirements. Additionally, to validate the HSC code, a number of test cases were to be designed and algorithms implemented to ensure that there were planned operations in the event of a malfunction of components and/or controllers. The simplistic model in Figure 11 shows an overview of the development process followed by UWAFT.

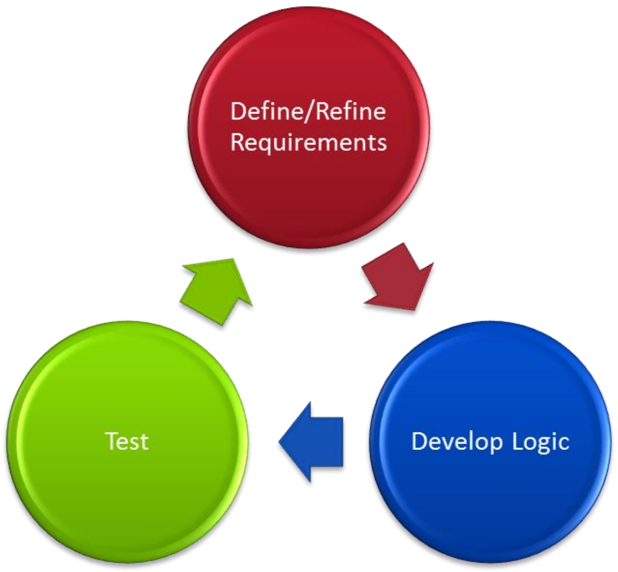


Figure 11: Development process layout

As shown, this is an iterative process that allows for more requirements to be defined during the exploratory analysis of the system (development vehicle here) and when more real data was collected. Hence, all the logic for a requirement need not necessarily be developed at once. As the code or logic develops from the most simplistic solution, it will be tested to check if it meets the safety requirements that have been identified. The following sections go into detail of how the important steps for development of integrated HSC code are followed.

3.3.1 Developing plant models and soft ECUs

In the context explained here, the virtual representation of a component in the vehicle is called plant model and that of component controllers is called soft ECUs. The benefit of developing these are that controls algorithm development for HSC need not absolutely rely on the availability of the vehicle (that may be getting integrated mechanically and electrically). The development work for plant models and soft ECUs started simple with 1D or 2D efficiency maps. Simple diagnostics for HSC were initially tested in isolated models (MIL) until the need to implement more features were pressing. This helps optimise work since isolated testing can be done in parallel (at least initially for verification). It is important to note that the best model is not the one with the highest accuracy, but the simplest one that can most effectively help test the HSC code.

Increasing understanding of the functioning of plant models is beneficial if modifications are to be made in the future. Since models are needed for testing, the naming convention and placement of signals were kept uniform for testers to find them easily. Another good practice UWAFT strived to follow was to not make a model of very high fidelity, unless it was demanded by the HSC code. Since actual component data can be collected for comparison, simple models using look-up tables (that use empirical data) are sufficient for UWAFT's use. This is better than a highly parametrised model that can no longer be analysed or evaluated.

The primary functionality expected from a soft ECUs is that it interfaces with the HSC. Soft ECUs generally consist of component start up and shutdown sequences, CAN Tx and Rx for communication, different I/Os as well as different component modes of operation (normal, limited functionality, emergency power off-EPO). As with development of plant models, the signal names and format of signals were captured correctly. Override switches were placed in an agreed upon spot for testing and/or generating faults. The practice of using them on every signal during initial stages of development is not uncommon, specifically, when it is hard for members to gauge where it is necessary and where it is not. Within the existing UWAFT vehicle model, some soft ECUs were inherited from EcoCAR 2 and others developed. Again, an evaluation of how reasonable these are and understanding them for easy manipulation is recommended. Further practical tips for development of soft ECUs have been provided in Appendix D.

3.3.2 Developing HSC requirements

For the development of the HSC code for the UWAFT PHEV, the V-model seen in

Figure 12 was used as a guideline. Each action performed by a driver is considered as a use case and corresponding expectation is documented as the customer use case requirement. For example, when the driver puts the key to crank, the expectation of the engine getting ready for propulsion is a use case requirement. In order to meet this requirement, certain sub requirements called feature requirements need to be met. For a hybrid vehicle, the HSC has to send commands to the engine controller for it to actuate the engine. All such features that require initiation from HSC for devices that the driver interacts with is noted along with the controllers that are necessary for these features. The exact steps to meet each feature by the HSC is noted as subsystem requirements. These are extremely specific and can be plentiful as the information to successfully build HSC algorithms need to be captured. The subsystem requirements include information such as inputs and subsystem outputs necessary for the algorithm, how the HSC can determine if the controller is switched on to initialise communication, and then determine if HSC commands have been honoured by the controller.

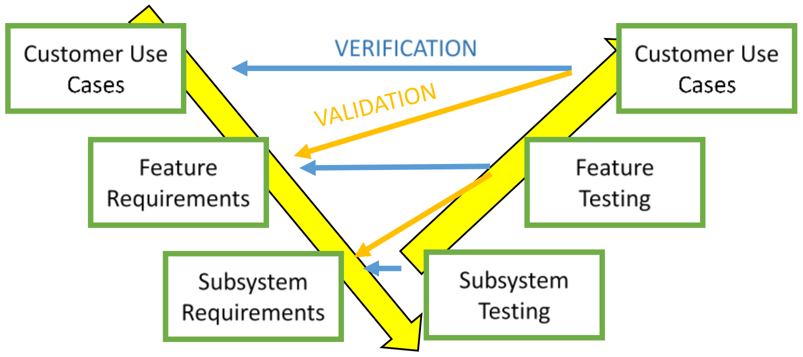


Figure 12: V-model for documentation of requirements and test cases [23]

Each requirement was to be documented adequately so that a person who has not researched the component could read the requirement and develop corresponding code. Once the code was developed, a senior CSMS member would review the code for a static test. Static testing is when the code is inspected to find errors without executing any test cases. These errors would include non-functional parameters such as readability of code, naming convention and any mistakes that an experienced pair of eyes could catch. The developed piece of code would then go for functional testing. If new requirements were found as a result of testing or if at any point the developed code was found not to meet the requirement, it would follow the above steps in the development flow (Figure 11) for verification of any changes made to the piece of code.

All the requirements have corresponding test cases that are designed, evaluated and documented. These test cases, like the requirements, have unique identifiers, documented revision dates and refers to the relevant requirement ID. This gives the team the ability to have traceability between the different levels of requirements as well as corresponding test cases. Refer Appendix C for further details on the documentation of HSC requirements. By maintaining this organized documentation trail, UWAFT's safety team could note which higher level requirement is met on the successful verification and validation of lower level requirements as shown in

Figure 12. Similarly, a lower level requirement can check which higher level requirement drives it. All this documentation also provides reference for students in the later years to develop HSC code with safety incorporated.

The developed integrated HSC code has been discussed in detail in Chapter 4. The step after development of HSC code is to develop tests and that methodology is described next.

3.3.3 Test Case Development

The verification and validation of all the algorithms (especially safety-critical) that get flashed on to the on-board supervisory controller is crucial and is done through a systematic effort of the CSMS and safety teams. The safe operation of the vehicle is necessary for the safety of the passengers, the people around the vehicle as well as the components itself. This is methodically done with the help of test cases that are verified and then integrated into the main HSC code in a step-by-step process for validation before being used in the vehicle. Verification and validation are both testing of a unique algorithm. However, verification involves only a single test that confirm a corresponding single requirement works as intended, whereas validation confirms that the algorithm works in all the test cases it was intended to work in. Validation involves a number of requirements that needs passing of a number of tests as shown in Figure 12.

After reading the function described by subsystem requirement, research on how exactly the function can be detected so that a flag can be set is conducted. Depending on how important the flag is, it is decided if the flag gets latched into a fault. Subsequent iterations would see the evaluation of the fault to be resettable or not (depending on the criticality of the component as well as time in hand for testing) and the conditions for allowing this reset. The validity of these flags and faults are evaluated depending on the reliability of signals that are used. Phenomena such as sensor drifts and that thresholds in different testing environments can be different needs to be accounted for when writing algorithms for diagnostics of signal validity. In preliminary development of the HSC code, usually if it is a powertrain component, safe shutdown of the component was initialised.

After the algorithm development, a fault is injected into the model in the SIL or HIL environment and isolated subsystem testing ensues using constant blocks, override switches and function generators as inputs. This is so that the algorithms developed for HSC can be stressed with different faults and is followed by feature testing and system-level testing as the ascending requirements are met. There could be new requirements that are generated in this phase since there are system level interactions that can result in unexpected results. It is best not to immediately implement the remedial actions from the HSC in the vehicle until it is ascertained that the faults and flags are not false positives and all related test cases have been implemented. This is discussed further in section 3.6.

The template for documenting these test cases along with other test case documentation are provided in Appendix C. By organized documenting, the test cases could successfully be reproduced when there is a necessity to repeat verification and validation of the modified HSC code. This also forms as a method to establish test coverage for different algorithms and a tangible way to prove that the code has undergone rigorous testing for safety. It is better to write repeated, concise test procedures that are requirement-focused than a complex large test case that is poorly worded and causes confusion during execution. This would allow segue in to automated testing that will be expected to be implemented in Year 4 of the competition.

Since the testing is done ideally by someone who has neither written the requirements nor developed the code, the pass and fail requirements also need to be adequately specified in the test case documentation. Log files that are used to maintain the inputs and outputs from various tests are also to be dated so that during post processing on a later date, the code used can be deciphered by comparing dates. According to the specified criteria, the tester assigns an overall pass or fail and saves the documents. If passed, the safety team is notified and the corresponding requirement is updated. Maintaining a high-level test plan can help with the process for traceability and quick access. This helps in scheduling tests to be performed as well as checking status of the test cases.

3.4 Real-time Interface (RTI)

dSPACE platforms are capable of carrying out fast design iterations once the setup is properly done. The software from dSPACE, ControlDesk Next Generation (CDNG) helps to seamlessly implement Simulink and Stateflow models onto the real time hardware such as HIL simulator and MicroAutoBox II. This is done by extending C code generator Simulink Coder that translate C code to object code using shell programs that include single-step compiling and assembling. During compiling and building of the code, errors are identified due to the checks automatically done and the user gets notified so that any issues due to wrong parametrization can be avoided while testing on the hardware.

RTI CAN Multi Message (RTICANMM) Blockset is an extension of the RTI Blockset but can handle a larger number of CAN messages. Its capabilities are several such as CAN communication networks can be configured and messages can be transmitted over another CAN bus with the help of RTICANMM settings and RTICANMM Gateway block [35]. There are modules available to expedite training and understanding of using these useful tools. HelpDesk is the go-to friend for explanations (like MATLAB's help section) as well as how to interface with the platforms and provide platform pinouts. Experiential learning is the best way to go in order to understand the workings of these complex modules.

3.4.1 Hardware-In-the-Loop (HIL) simulator

As has been discussed, several test cases are tested using HIL platform for the HSC code that controls the powertrain as well as other body ECUs. The compact, mid-size simulator with DS 2202 processor board is used at UWAFT for HIL testing. The simulation model runs on the powerful processor board and the HIL simulator generates and measures I/O signals via integrated I/O boards. It also consists of failure insertion units that can simulate electrical failures on all the ECU output pins connected to the HIL I/O board. However, this is a feature that needs to be used with caution, especially when ECUs are attached to the HIL during the phase of achieving baseline functionality since it can potentially burn up ECUs if short circuits are simulated. It is capable of shorts to the ground or battery voltage along with cross-wired shorts; open circuit as well as activation of multiple failures [36]. The simulator can also simulate complete loss of bus to see the effects of signals that do not update on HSC. RTICANNMM has an option to stop all messages to a bus and another for not sending any messages from a particular controller to simulate a non-functional controller. Bus overloading can be simulated by sending dummy messages at high speeds with high cyclic rate and high priority. These are all tests that are hard to simulate in a SIL environment.

3.4.2 HSC hardware

The Hybrid Supervisory Controller (HSC) used by UWAFT is dSPACE's MicroAutoBox II (MABx). It can connect to 6 CAN channels; all of which are being used for development work by UWAFT as seen in Figure 14. With its operating range of -40 to 85 degree Celsius and number of analog and digital I/Os, the 1513 variant is ideal for UWAFT's application.

However, UWAFT also understand the limitations of this device. The device is best suited for low current applications (analog outputs). To illustrate, due to UWAFT's vehicle architecture, it is necessary that the HSC intervenes before the driver inputs from the gear selection (PRNDL) reaches the transmission control module (TCM) for it to honour the requests. The HSC would evaluate the vehicle condition and send fitting PRNDL selection for the TCM to honour. These requests are sent via analog lines with differential voltage that was held throughout the vehicle's operation. UWAFT observed that within 2 or 3 hours of testing, the transmission would slip into a faulty mode and was no longer safe or reliable. While conducting root cause analysis, one of the helpful indicators that pointed towards MABx was it overheating. The ability to monitor the processor and board temperature via CDNG helped to further investigate the problem.

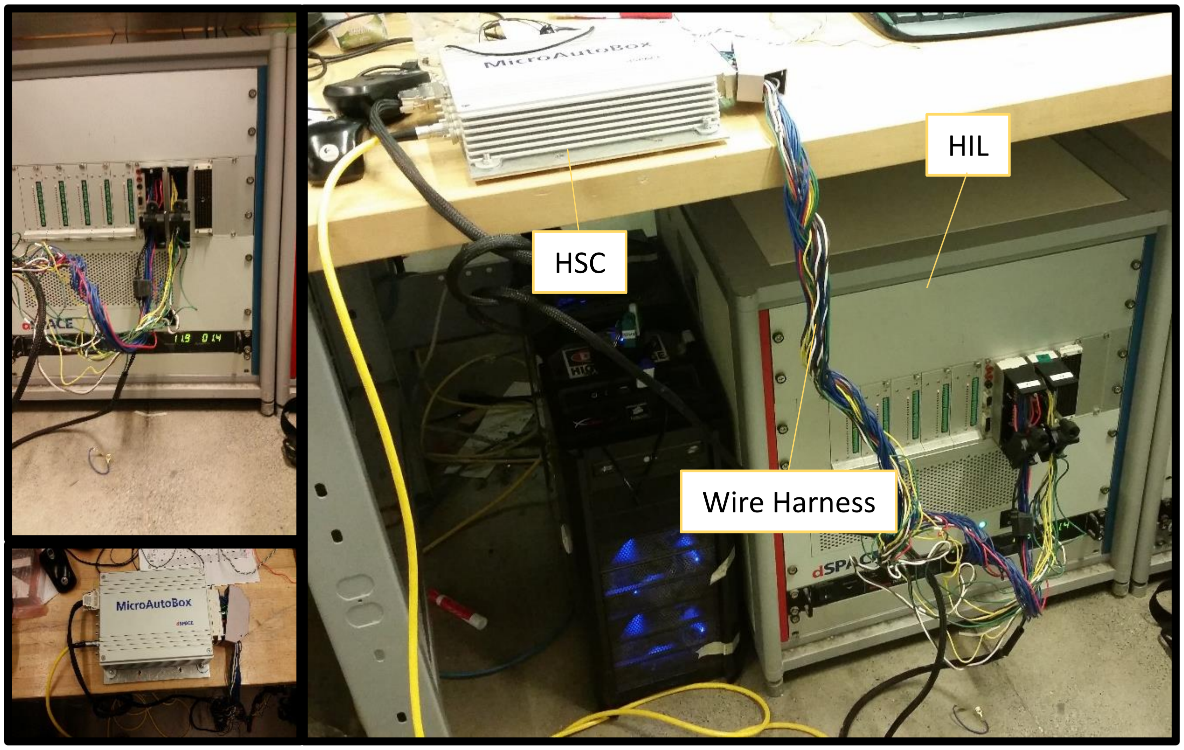


Figure 13: Physical HSC-HIL setup for testing and individual hardware photographs

Photographed by author in UWAFT office space (November 2016)

It was later discovered that the MABx was power limited and that currents of above 15 mA for this application was much above its rated capacity. UWAFT subsequently worked on a solution where digital outputs from the HSC would pass through an external controller (distributed controller) and that would drive the TCM instead. This remedial action would then help the transmission work as a safe and reliable component.

3.5 Baseline functionality of components

Following the software development for component start-up and preliminary communication along with the availability of those corresponding components and/or controllers, the HSC was interfaced with the different hardware parts to achieve baseline functionality. When the messages from the electronic control unit (ECU) were on the bus and the component was ready to receive its enable message, baseline functionality of that component was said to be achieved. This is closely related to bench testing which would not only include components and controllers being powered and initial communication with powertrain controllers being established, but also checking to see if the expected benchmarks (understood from requirements) as a result of that communication can be successfully achieved.

While waking the lower-level controllers (component controllers), they need to be provided with 12V power. This 12V power can be provided through an external source, from within the vehicle or with the help of the HIL simulator. Many of these controllers stay awake only as long as it sees a particular signal at high, occurrence of specific messages or if a bus is alive. The HSC communicates through CAN, analog or digital signals. This communication information is obtained from the data provided by the manufacturer in the form of interface control documents, CAN specifications and/or database container (.dbc) files. A .dbc file specifies message composition such as signal names, bit length and order, and cycle time. After the initial handshaking, the HSC sends requests such as request for engine torque to engine controller. By HSC sending a request to a controller, it is not always guaranteed that it will be honoured. Hence, feedback from component controllers is important, especially if subsequent decision making by the HSC relies on that information or it is safety-critical.

Achieving baseline functionality of the components was not always easy. In fact, at times controlling the individual components were not as hard as establishing that initial communication. This is because there is a level of understanding of the component that is required to correctly wire the component and troubleshoot, which can be quite challenging. However, the sequence for writing algorithms to start-up and control the component and/or controller is quite often given, in the mentioned interface documents. In Appendix D, some troubleshooting methods have been mentioned. It can be noted that in this phase, since only certain components (mostly powertrain components) were being tested, the entire HSC model was not necessary.

Certain components needed to handshake with other control modules before it could enable component functionality while some others needed more than one wake line in addition to power. Due to the limitations of time, it was always a challenge as to whether the baseline functionality of the component needed to be achieved using the HIL simulator or VIL. This would depend on things like if the component required mounts, or auxiliary systems for functioning and if the benefits outweigh the additional work for set up by decoupled testing from the vehicle. Other factors like the availability of the component and schedule conflicts with integration would also contribute to the kind of testing performed.

3.6 Diagnostics and remediation

Achieving baseline functionality of the components does not guarantee that the components and controllers will always operate correctly or without failure. Since UWAFT has a single hybrid supervisory controller, a number of diagnostics have to be performed in order to ascertain the status of different ECUs and also the integrity of important signals like accelerator pedal, and PRNDL. Diagnostics is the process of assessing performance of the vehicle at lower levels such as signal, component and controller to the system level and then vehicle as a whole in order to set appropriate flags or faults. UWAFT has successfully developed diagnostics for a number of safety-critical components and will continue to develop them in Year 4 for more component coverage. However, be wary of developing unnecessary diagnostics.

Parallel to UWAFT's own diagnostics' development, UWAFT also understands the diagnostic trouble codes (DTCs) generated by the stock car and clears them wherever possible. The DTCs are found with the help of the Snap-On car diagnostics tool - SOLUS™ EDGE full-function scan tool and software. The touch screen with an intuitive interface is easy to use and complete the code scanning process [37]. Many DTCs are produced as a result of missing components and in context to the development vehicle are false positives. As an illustration, the stock V-6 engine has been removed from the UWAFT PHEV but there are messages from the stock engine control module (ECM) that are required for other stock components to work. When these are not received as expected, there are other controllers that will produce DTCs since in the context of the stock car, this could be interpreted as non-functioning engine. However, in the context of UWAFT PHEV, these are false positives since there are other functioning components fulfilling the purpose of the stock engine.

It is equally possible for UWAFT to write code that can generate false positives since an algorithm can work correctly but can be limited by the thresholds applied. For example, assume a flag is generated for unacceptable ranges of ESS cell temperatures, however, it is not taken into account that before closing contactors, this signal is always maintained at unrealistically high numbers in order to distinctly mark when the HV bus is alive. This would result in generating flags (false positives) when the bus is offline, although in reality, there is no issue. Another example is diagnostics for checking whether requested motor torque equals actual torque produced by the motor. The feedback signals from torque measurements are usually transient and susceptible to a margin of error since it is hard to measure torque accurately. However, if a fault is generated with no or little bands of tolerance, it can result in false positives. It would be worse if remedial actions are practised without appropriate testing. Remediation is the assessment of the flags and faults generated as a result of diagnostics and executing algorithms to alter the vehicle's behaviour.

Remedial actions like shutting down propulsive power or changing motor behaviour must therefore be implemented only after sufficient testing.

As mentioned, certain ECUs like the TCM and other manufacturer built controllers perform their own diagnostics immaterial of the CSMS team's requirements or the lack there of. In many occasions, this information is proprietary and has to be treated as a black box. If component behaviour seems abnormal or if DTCs are being produced, there is a need to try to understand and observe what inputs result in this behaviour. This information may be available on the online service manual availed to UWAFT members who have signed Non-Disclosure Agreements (NDAs). If not, information may be requested from respective manufacturers (GM mentor for GM requests). It is to the company's discretion if they want to part with the information that is requested for. Some of the stock or other OEM component diagnostics and remediation which are also emulated by UWAFT include but are not limited to:

- Normal behaviour if flags generated are of low priority, but can be counted to respond differently if enough number of flags are generated within a period of time.
- Actions are implemented so that system level behaviour changes to acknowledge and alleviate issue. It may not result in change in the specific component's behaviour.
- Functionality of component is temporarily disabled.
- Component functionality is reduced.
- Component is not allowed to be used until the fault clears.

Unlike the component controllers and control modules developed by manufacturers, for the UWAFT HSC, additional diagnostics for the important signals as well as system level diagnostics are required. Some of the signal diagnostics is to ensure the signals are within acceptable ranges or valid to be used for decision making by the HSC. System level diagnostics ensure that a system is capable of functioning given that some components are not functional. For example, the electric powertrain cannot function if the ESS has an Emergency Power Off (EPO) and now the vehicle has to run in an engine-only mode. Hence the UWAFT HSC is responsible for disabling certain modes as deemed by component diagnostics and other things like limiting allowed maximum and safe shutdown of vehicle if diagnostics are critical.

3.7 Overview of CAN

Section 2.4 gives a brief explanation on CAN which is the main communication means between the ECUs in the vehicle. UWAFT makes use of high-speed dual CAN as well as low-speed single wire CAN. In discussions here, mostly the high-speed CAN is referred to. Although quite robust and capable of tolerating a high number of messages, to avoid bus errors and failures it is best to load it to a maximum of 80 % of its capacity for development work. Popular literature recommends to keep it to 50% of loading capacity for commercialised products. If not absolute failure, there would be tell-tale signs like disabled features or delayed response to driver input showing that the bus is overloaded [30] [38]. For example, one of the six CAN channels used at UWAFT that holds nodes such as the engine controller and two motor controllers was not functioning properly. This had a transmission rate of 500 kbps. On troubleshooting, UWAFT identified it as a bus loading problem. There were a couple of possible solutions -

(i) Increase the cycle time of the CAN messages (for example, from every 100 ms to 500 ms) on the channel so that lesser number of messages are transmitted on the bus from each node within a period of time. In other words, the values are updated slower. However, the powertrain components are safety-critical components and this is not advisable since faster updates of their condition is crucial.

(ii) Migrate node to another bus that is not as loaded. However, this is feasible only in the initial design phase since it would require rewiring in this phase. UWAFT was on time crunch with competition around the corner. Also, there were reasons such as certain control modules that have to be isolated occupying a number of available CAN buses.

(iii) Increase the transmission rate of the network. UWAFT went with this option and began transmitting that CAN channel at 1 Mbps. This solved the problem.

Another problem that is often encountered is incorrect prioritization of messages that results in critical messages not being sent out first. At the same time, an unnecessarily high cycle time for a high priority message is not advisable either. Other messages on this channel with lower priority may not get an opportunity to get transmitted. Certain hardware considerations would help maintain the CAN bus integrity such as appropriate termination, understanding electromagnetic compatibility issues with high voltage components and taking appropriate steps to shield from them when encountered. Also, repeated removing and reinserting of connectors can result in weak connection; high quality connectors and wires that are appropriately crimped are used by UWAFT to avoid issues.

Improper hardware integration can result in hours of troubleshooting. One such example is of the time the clutch actuator was not communicating and showed a fatal error flag on inspection of the bus. The suggestion on the clutch manual was to take the component in for service. This was just about a week before the competition and was not a viable option since the location of the manufacturer was in the US. UWAFT relentlessly went on to troubleshoot and after a number of times of powering the component and trying different things, it was found that the ground was not properly connected. Once this was relocated, all the problems were fixed. The only reason a lot of time was not spent pouring through the code was because it was tested and validated; the clutch actuator had worked without incident for six months prior to this. Also, one of the members remembered that the ground had been changed by another member. This incident highlights the importance of communication along with completeness of hardware integration.

While dealing with a development vehicle, there are many points in time when the CAN bus needs to be modified. Before modifying the existing vehicle CAN bus, it is important to have a practical understanding of the implications of the actions. First the topology of the CAN bus must be understood. On removing or adding nodes to the bus, it will change the length between nodes and this could affect signal stability in some cases. It is important to note if other nodes on the CAN bus were receivers of the removed node and what this removal would mean to them. By addition of a node, usually a data logger, check if it meets the termination requirements. There will also be changes in the bus loading by addition of a node such as a new controller.

The UWAFT CAN architecture has changed over its three years of development as new information was gathered. The present CAN architecture is presented in Figure 14. UWAFT uses dual CAN and generally has a linear topology. However, one of the CAN buses (CAN 2 in Figure 14) now has a star topology which the team is considering to rewire.

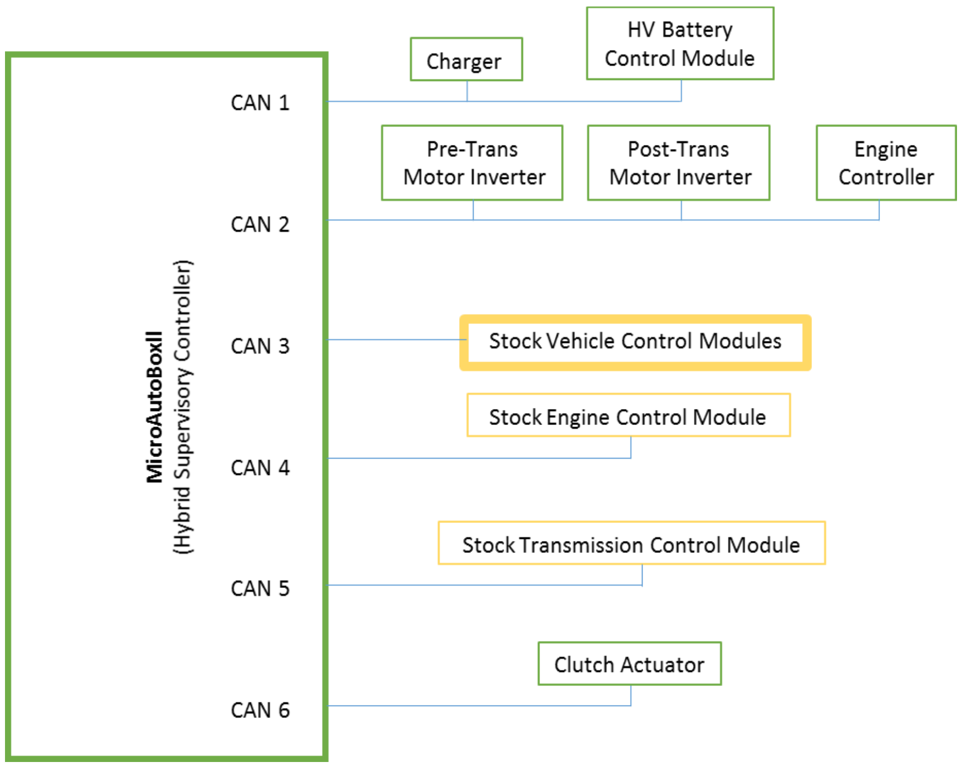

MicroAutoBoxII
(Hybrid Supervisory Controller)
CAN 1
CAN 2
CAN 3
CAN 4
CAN 5
CAN 6
Charger
HV Battery Control Module
Pre-Trans Motor Inverter
Post-Trans Motor Inverter
Engine Controller
Stock Vehicle Control Modules
Stock Engine Control Module
Stock Transmission Control Module
Clutch Actuator

Figure 14: UWAFT CAN architecture in Year 3

The reason for initial changes in the CAN architecture was based on the fact that the stock Engine Control Modules (ECM) and Transmission Control Module (TCM) had to be isolated and only necessary messages needed to be transmitted and modified for the proper functioning of the other stock control modules. Messages had to be spoofed so that although the components were not available (stock components like engine not being present), the control modules were not capable of limiting other body ECUs to either shutting or functioning in limp mode. This would also ensure safe, predictable working of the developed car. The clutch actuator got its own CAN bus since the sourced part utilizes standard SAE J1939 with a lower transmission rate. J1939 is generally defined for heavy-duty vehicles like buses and trucks.

The near-future addition of CAN enabled relay controllers and a controller for the eLSD will necessitate the re-evaluation of the CAN architecture according to the requirements that need to be met. There could also be other controllers that the team wants to add to the architecture as time and vehicle development progresses.

Chapter 4
HSC Code Development

4.1 Managing HSC development

At UWAFT, the development of the Hybrid Supervisory Controller (HSC) is a joint effort by the CSMS team members. With a code that is being written by a number of people, structuring the layout and establishing conventions for nomenclature is essential. The structure should be such that the work can be divided intelligently amongst members and the end result is organized and functional. By partitioning the model, not only is parallel code development by multiple team members possible, but it also makes subsequent code development clear. This clarity is important not just for the development of the code and testing but also so that troubleshooting can be logical as well. Organization of the model can follow different variations based on functionality, powertrain and logical flow of the signals. However, choice of the layout has an impact on the efficiency of code execution. For a model as large as the one that UWAFT has developed, the effects can compound and build time can increase significantly.

4.1.1 Naming convention and signal routing

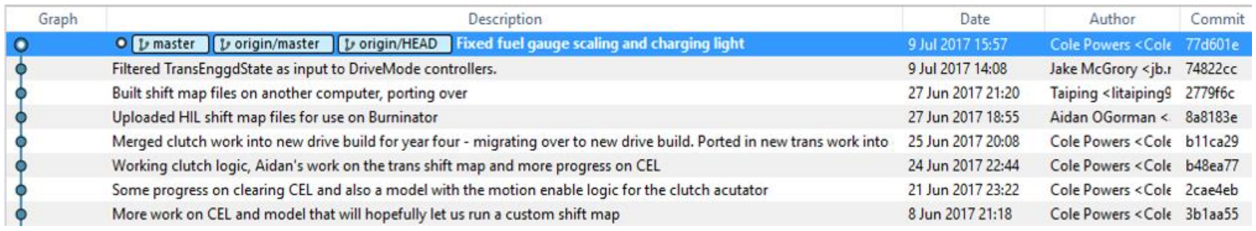
As for the naming convention, it is possible to name the signals whatever the team pleases, however a convention that is adhered to can make code development and subsequent testing more streamline. The uniformity in naming will increase readability of the integrated code. The naming convention at minimum should comprise of the component name, signal name and unit associated. For example, *bcm_soc[%]* is the name given to the signal that indicates in percentage the state of charge (SOC) of the ESS. The battery control module (BCM) is the dedicated controller for the ESS. Where available and possible, it is best to follow the CAN format as given in the dbc files. This is barring instances when the given signal names are ambiguous and unintuitive. As an illustration, signal in the CAN format that reads NLG5_C_C_EL gives no particular information of the data carried by that signal. Hence, it was renamed in the UWAFT HSC.

In certain cases, the data type of the signal is also mentioned in the name of the signal since it is an important entity. There can be data type conflicts that cause errors in the code and prevent the entire code from being compiled. UWAFT gets around this by making use of 'Data Type Conversion' blocks on the signals. Proper signal routing is another important aspect of the code development. UWAFT prefers using buses over mux for signal routing, especially because not obeying routing rules can result in errors.

4.1.2 *Tracking changes and version control*

Once the structure and signal conventions of the HSC code is understood, next it is key to understand how the HSC code model management is done. Some teams employ custom libraries in the HSC code because any modification to a linked model in the HSC code is propagated to all the other models that are linked to this library. Model referencing is another method employed by teams. Developed models are introduced into the HSC model as Simulink blocks (like function calling) with the inputs and outputs of the referenced model used as the I/O ports. Employing model referencing allows for separate files that can be worked on and post validation, the model can be updated with the correct model. UWAFT uses custom libraries but does not use model referencing since it was found that all blocks could not be employed; with notable trouble using RTI blocks (necessary for HIL testing). However, there could be software patches that have since developed.

As discussed earlier, it is imperative that all tests related to a particular code change is run before that code is deemed VIL "worthy". Being VIL worthy means that the HSC executable will be flashed on to the on-board MABx and subsequently used for driving the car. Hence, these modifications need to be tracked. UWAFT, like many other software development teams, uses the version control software – GitHub [39]. This program not only helps with version control of the code but also helps track which team member has made a particular change. UWAFT makes use of the Atlassian tool SourceTree [40] for cloning the code locally from its Git repository. It is a Git client that the team members have learned to use effectively and is shown in Appendix F.



Graph	Description	Date	Author	Commit
	master origin/master origin/HEAD Fixed fuel gauge scaling and charging light	9 Jul 2017 15:57	Cole Powers <Cole	77d601e
	Filtered TransEnggdState as input to DriveMode controllers.	9 Jul 2017 14:08	Jake McGrory <jb.r	74822cc
	Built shift map files on another computer, porting over	27 Jun 2017 21:20	Taiping <litaiping9	2779f6c
	Uploaded HIL shift map files for use on Burninator	27 Jun 2017 18:55	Aidan OGorman <	8a8183e
	Merged clutch work into new drive build for year four - migrating over to new drive build. Ported in new trans work into	25 Jun 2017 20:08	Cole Powers <Cole	b11ca29
	Working clutch logic, Aidan's work on the trans shift map and more progress on CEL	24 Jun 2017 22:44	Cole Powers <Cole	b48ea77
	Some progress on clearing CEL and also a model with the motion enable logic for the clutch acutator	21 Jun 2017 23:22	Cole Powers <Cole	2cae4eb
	More work on CEL and model that will hopefully let us run a custom shift map	8 Jun 2017 21:18	Cole Powers <Cole	3b1aa55

Figure 15: UWAFT master branch

Some teams follow the method of placing the tested (verified and validated) integrated HSC code on the master branch and the developing pieces of code on new branches. However, while committing and merging updates from branches to master branch there is a high probability of conflicts. At UWAFT, the most up-to-date code is placed on the main master branch as shown in Figure 15. The unverified code that is being tested is placed in separate files (separate names) also on the master branch. Once verified, it gets merged manually to the main integrated HSC code by the members to avoid any issues and the old code is archived.

The core CSMS members would be aware of the name of the main integrated HSC code that is VIL ready and the ones that are being verified. The reason this kind of model management can be employed by UWAFT is due to the relatively small size of code contributors.

It would also make sense to archive the code in a separate space (such as a team server) or tag the code that has undergone significant changes within a particular time period and/or meets a significant milestone requirement. Tagging the code is possible while using Git. This way, if a problem with the code is identified only after subsequent code changes are made, there is a way to track back to the nearest tagged code to iterate the process of testing. Commenting elaborately during each code commit and push is encouraged so that the node on the master branch to revert to can easily be located. The faster the pace of code development, the more there are chances of something unforeseen happening. Even with existing measures, code conflicts when changes are made to the same piece of code is likely. It is important that whatever the methodology used for code management, it is one where a single person cannot "break" all the code that has been developed.

4.2 Overview of UWAFT HSC

The HSC code started very simple with initially only the structure of the code in place. In fact, in many parts of the code, 'Terminator' and 'Ground' blocks were used as placeholders for inputs and outputs before the algorithms were fully developed and tested. The EcoCAR 2 control strategy was used as reference and in some places adopted before fine tuning. Once the components for the UWAFT PHEV were confirmed, corresponding start-up algorithms were developed. The electric powertrain strategy and development was prioritised over ICE or hybrid. Through parallel development, different start up sequences for available components were tested and CAN communication established for those components. Around then CSMS members noted that the code structure for HSC that was developed had limitations in terms of manageability. This resulted in the first restructure of the HSC code. Following the baseline functionality of the powertrain components, other components were bench tested while simultaneously the integration of the components in the vehicle progressed. Once the essential components were successfully integrated, the vehicle start-up sequence was tested with the help of overrides. This was to be followed by mapping vehicle acceleration to vehicle actuation and required torque-producing components to power the wheels.

Immediately after the ESS was commissioned at Year 2 competition and the main contactors for the ESS could close, attempts to spin the wheels using P3 motor ensued. UWAFT was triumphant in powering the wheels by the beginning of Year 3. By this time all the signals in the code matched the actual vehicle signals. The next milestone was the on-board charging system being functional. Since a lot of development work occurred at different paces and in parallel, it is hard to recount exact details in a chronological fashion.

As development work continued and Year 3 progressed, most of the hardware, including cabin lights and auxiliary boards/controllers, associated with the HSC were operational. The motors and engine were successfully receiving torque requests and delivering torques as requested. The transmission was operational and initially functioned in a limp mode. The isolated powertrains were deemed functional through pre-competition safety inspection. It was during the HSC code development for this safety inspection that the need for a second restructure of the code was found. The HSC code that existed at that time could not support development work for all the drive modes for the UWAFT PHEV. Naturally, the CSMS team began visualising a new structure that could help with this refinement and optimization of the code and began developing the structure explained in section 4.3. Subsequently, the software architecture and code development process was tuned for rapid code development and helped keep the code more serviceable and manageable. Despite the black box approach required, the transmission could beautifully transition through all 8 gears. The component interplay was assessed and the hybrid control strategy including torque blending implemented. However, UWAFT is still collecting more vehicle data in order to optimise developed code and develop appropriate system level test cases. Additionally, some hardware that had functional limitations were being improved. For instance, the AC compressor could not function reliably and the thermal system was not evaluated.

4.3 Description of UWAFT HSC

The functional code structure that is followed in this work is the model – *UWAFT_Camaro_Y3Comp_Win* - that was developed by Year 3 competition. The HSC code at the highest level has been partitioned into three subsystems as shown in Figure 16.

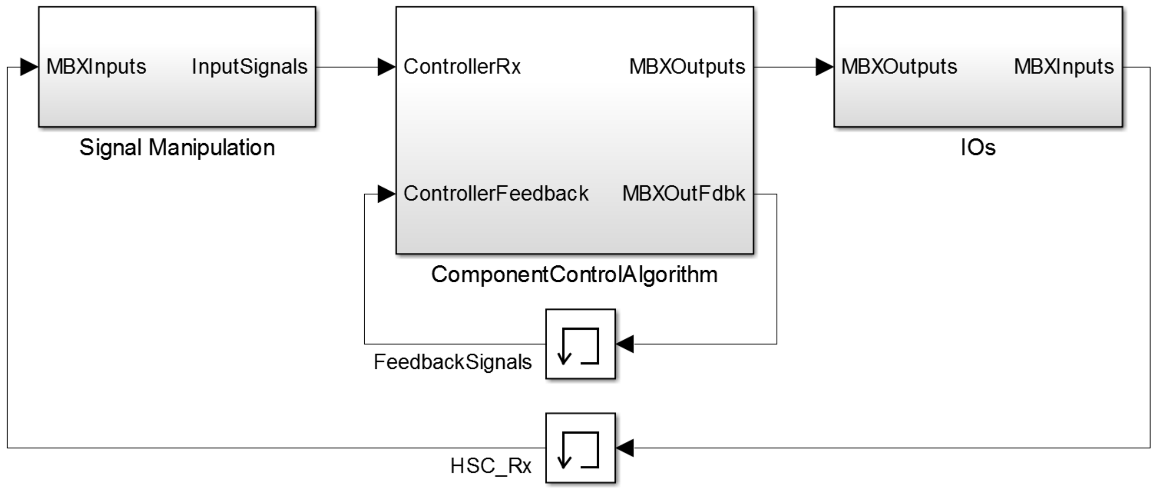


Figure 16: Highest level of representation of HSC code

The ‘Memory’ blocks shown in Figure 16 delay the provided input by a step. The outputs from ‘Memory’ blocks have been labelled *Feedback Signal* and *HSC_Rx*. MBx or HSC are the different notations used to identify the hybrid supervisory controller (hardware name – MicroAutoBox) in the code developed. Within the *Signal Manipulation* subsystem, the inputs to the MABx are renamed to make it easier to run through the code. UWAFT has made consistent efforts to maintain the code as readable as possible. The *Signal Manipulation* block is divided into subsystems of different components and have corresponding signals manipulated in them. Manipulation can be in the form of scaling, filtering and other basic operations to ensure that the units are correct for the actual processing that is done in the *ComponentControlAlgorithm* subsystem. It also ensures that the component controller communicating with the HSC can send the expected proper CAN messages. Additionally, a subsystem of raw signals has also been directly passed through (*Bypass* subsystem) within the *Signal Manipulation* block for access to all received inputs being received by the MABx.

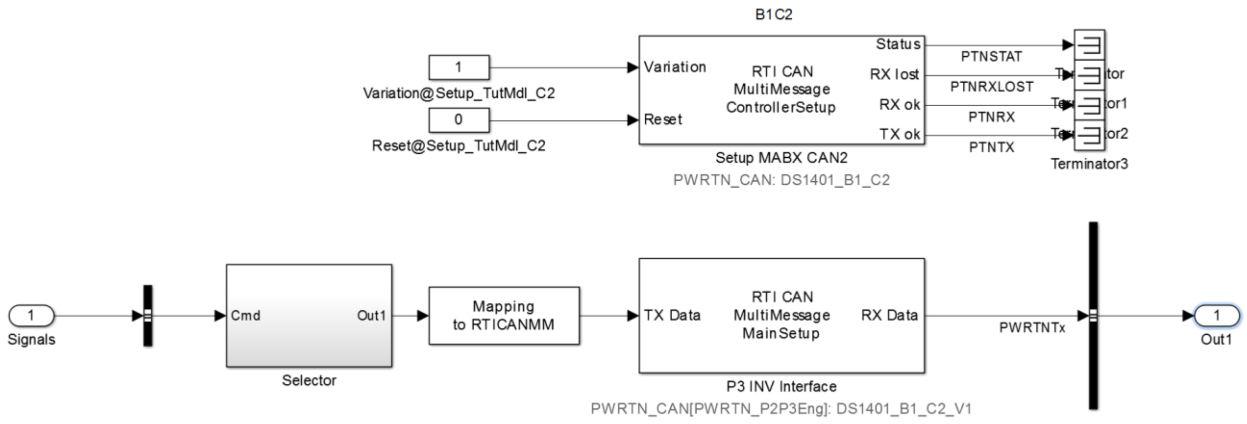


Figure 17: Representation of RTICANMM Controller and Main Setup

The *IOs* subsystem consists of the different interface setups for MABx to establish communication with the rest of the vehicular components. This can be using CAN, digital or analog IOs. Within the *RTICANMM setup* subsystem, the six channels for MABx that are used for CAN communication have been setup with the correct baud rate on the controller as well as channel. Figure 17 displays a lower level subsystem of one of the CAN channel setups. Within the ‘Main Setup RTICANMM’ block, the correct .dbc file for that CAN channel along with suitable settings is loaded so that all necessary information about the CAN communication is readily available and initialised. These settings can be loaded only if they have been previously developed and saved. Otherwise, the settings need to be manually selected. Further explanation on setup and how to use the various blocks is not within the scope of this thesis. Within the ‘Selector’ subsystem shown in Figure 17, the signals are renamed to their corresponding names as seen in the .dbc files and bundled to the matching message names. Caution is to be exercised while routing the different CAN signals and messages using ‘Bus Selectors’ and ‘Bus Creators’.

Similarly, for the Analog and Digital IOs, the correct pin is initialized for interfacing. Signal manipulation by scaling using ‘Gain’ blocks is visible. There are override switches placed within these blocks to aid testing. ‘Data type Conversion’ blocks are abundantly used at both ends of signal manipulation so that the code compiling is not interrupted by wrong data type. These blocks work as sources and prevent complications while compiling if signal names are not matching.

The middle subsystem *Component Control Algorithm* in Figure 16 and Figure 18 holds the bulk of the HSC code and is where most of the tested code gets integrated to. It determines which vehicle drive mode needs to be implemented along with fault mitigation and other optimizations. For functionality, it has been partitioned into four parts – *Fault Detection, Vehicle Commands, Optimizations* and *Executive*.

The outputs from *Fault Detection*, *Vehicle Commands*, and *Optimization Actions* are then sent to the *Executive* block where the individual operations of the components are commanded. This segregation benefits the code development since once the start-up and shutdown sequence is achieved for each component, the change in the rest of the code does not interfere with the integrity of the components’ operation. At UWAFT, the control strategies for the different drive modes were developed in detail only later and the segregation of optimization from executive came handy. The *Executive* subsystem is paramount to ensure torque safety. Throughout processing the driver torque request, if a fault is detected, the remediation can be initiated in the appropriate components.

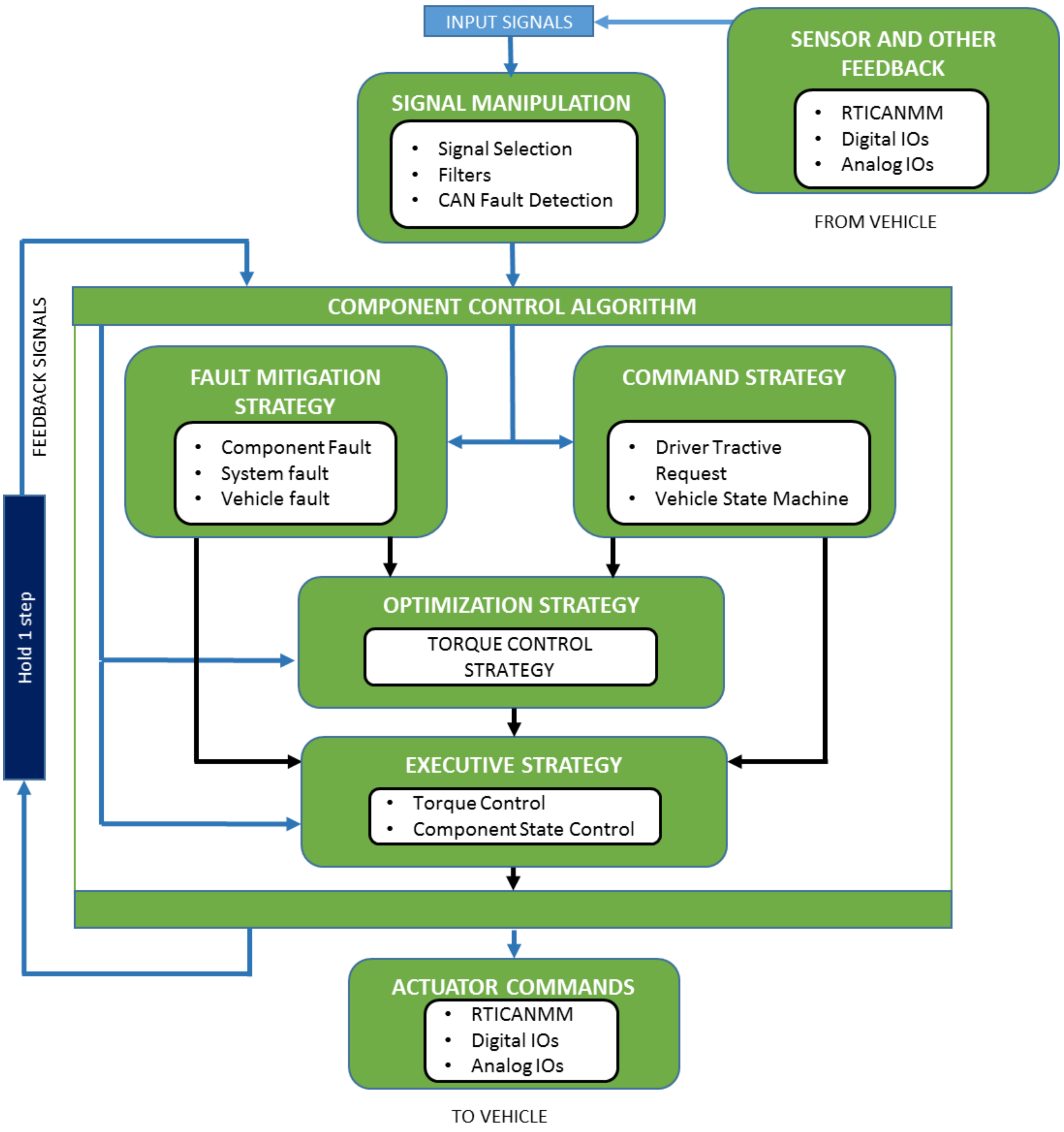


Figure 18: HSC Control Strategy

4.3.1 Torque Control Strategy

During Year 3, one of the most prominent changes in the HSC architecture took place in the *OptimizationActions* block. The finite state machines moved from being predominantly Stateflows to more Simulink-based; consisting of 'If' blocks and 'Switch' cases. This prominent change involved further partitioning the code and Figure 19 represents these sections of the optimization calculations, namely, *Torque Split, Drive Mode Aim, Drive Mode Parsing, Pre-Trans Torque Control, Wheel Torque State Control,* and *Parameter Determination.* These calculations are performed to find appropriate values (such as set-points, minimum and maximum values) to effectively control the different torque-producing components depending on the drive mode the vehicle is in. This torque control strategy is one of the HSC's most important functions. In the sections ahead, each of the aforementioned subsystems are elaborated.

The input signals on the right-side of Figure 19 include the signals received by the HSC, the feedback from the HSC, the fault status determined at the component, system and vehicle levels as well as the interpretation of driver commands. For effective torque splitting between the three torque-producing components, different algorithms for different drive modes are run. These result in parameters like torque values to be achieved by P3 motor and what percentage of the maximum allowable torque value of P3 motor it corresponds to. Similar values are calculated for the P2 motor to determine target torque value and corresponding percentage of maximum P2 torque. For the engine, by finding the target torque value and corresponding percentage of maximum engine torque capacity, the set point for manifold air pressure (MAP) is found using a look up table. MAP control is used for commanding the engine appropriately by the engine controller. All these calculations are done within the '*Torque Split'* subsystem.

For example, the preliminary algorithm for torque splitting in *FullEV* drive mode, where both the P2 and P3 motors can produce torque to power the wheels, is elaborated below. Within the code, the demanded tractive torque corresponds to what cumulative torque is to be supplied at the output shaft of the P3 motor (before differential). This is calculated with the help of a look up table with inputs of vehicle speed and accelerator pedal position. To further the understanding of the algorithm, Table 4 shows the mathematical representation of the algorithm.

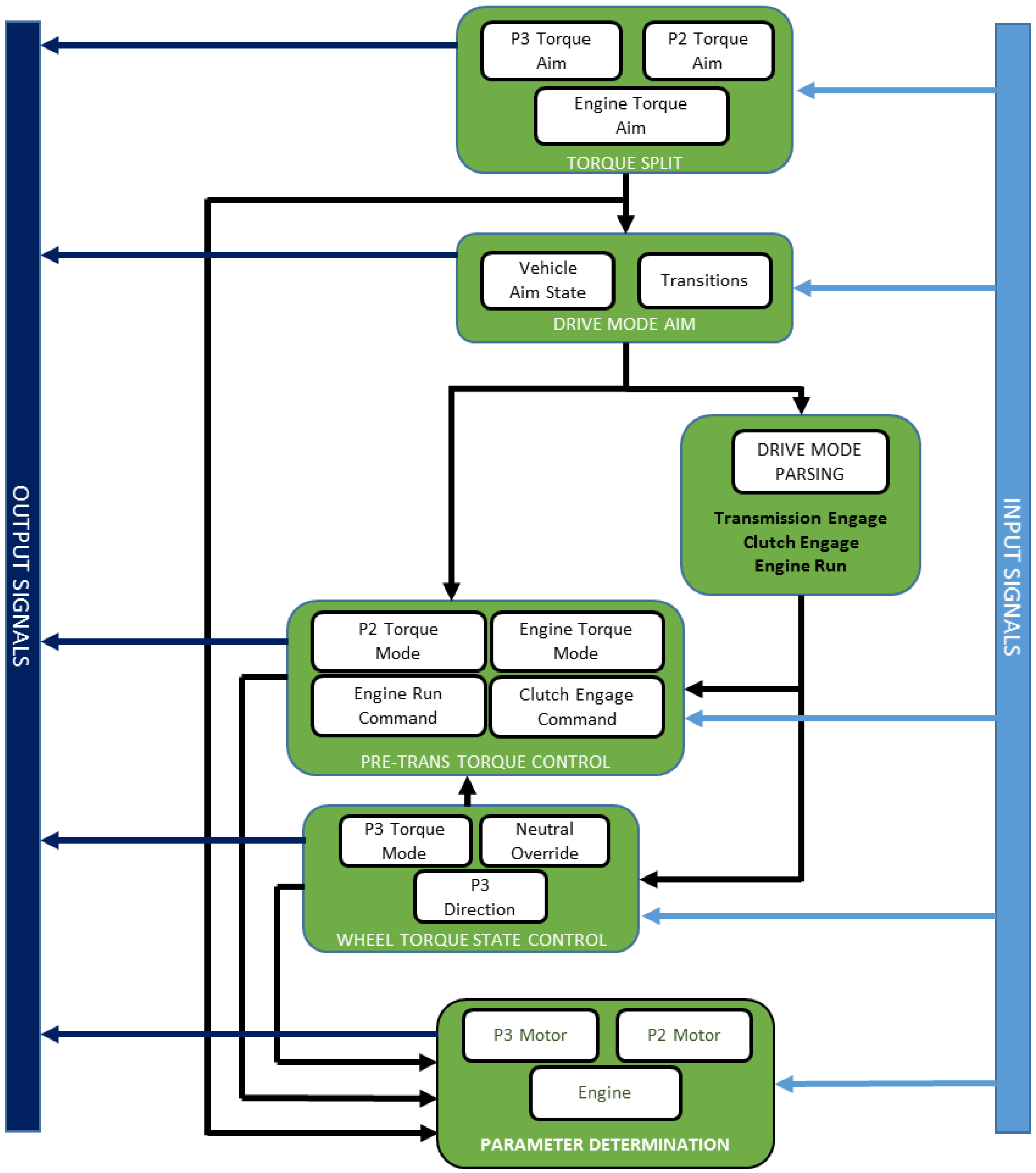


Figure 19: Flowchart in *Optimization Actions* Block

Table 4: Torque split in FullEV drive mode

Conditions	P2 torque aim [Nm]	P3 torque aim [Nm]
CST > $\tau_{P3max} + (\tau_{P2max}/GR)$	τ_{P2max}	τ_{P3max}
CST < 5 [Nm]		CST
If P3 motor speed ≤ 1500 rpm AND		
CST ≤ $0.15\tau_{P3max}$		CST
CST ≤ $(0.15\tau_{P3max} + (\tau_{P2max}/GR))$	$(CST-0.15\tau_{P3max})/GR$	$0.15\tau_{P3max}$
CST > $(0.15\tau_{P3max} + (\tau_{P2max}/GR))$	τ_{P2max}/GR	CST – (τ_{P2max}/GR)
If P3 motor speed > 1500 rpm AND		
CST ≤ $0.7\tau_{P3max}$		CST
CST ≤ $(0.7\tau_{P3max}+ 0.7\tau_{P2max})$	$(CST - 0.7\tau_{P3max})/GR$	$0.7\tau_{P3max}$
CST > $(0.7\tau_{P3max}+ 0.7\tau_{P2max})$	τ_{P2max}/GR	τ_{P3max}

Note: CST: Calculated Shaft Torque; τ_{P3max}: P3 maximum torque; τ_{P2max}: P2 maximum torque; GR: corresponding transmission gear ratio

1. If calculated shaft torque is greater than what can be provided, P2 and P3 motors target to output their maximum torque respectively.

2. If calculated shaft torque is less than 5Nm, only the P3 motor aims to provide torque to the wheels.

3. Given the condition P3 motor speed is less than or equal to 1500 rpm, and calculated shaft torque is less than or equal to 15 percent of the P3 motor's maximum torque capacity, only P3 motor aims for providing torque.

4. Given the condition P3 motor speed is less than or equal to 1500 rpm, and calculated shaft torque is less than or equal to the combined torques of 15 percent of the P3 maximum torque capacity and P2 maximum torque capacity corresponding to the gear ratio the transmission is in, the torque is split such that the P3 motor aims for 15 percent of its maximum torque capacity and P2 motor aims to produce the remainder of the requirement divided by the gear ratio of the transmission.

5. Given the condition P3 motor speed is less than or equal to 1500 rpm, and calculated shaft torque is more than the combined torques of 15 percent of the P3 maximum torque capacity and P2 maximum torque capacity corresponding to the gear ratio the transmission is in, the torque is split such that the P2 motor aims for its maximum torque capacity divided by gear ratio and the remainder torque is aimed by P3 motor.
6. Given the condition P3 motor speed is greater than 1500 rpm, and calculated shaft torque is less than or equal to 70 percent of the P3 motor's maximum torque capacity, only P3 motor aims for providing torque.
7. Given the condition P3 motor speed is greater than 1500 rpm, and calculated shaft torque is less than or equal to combined torques of 70 percent of the P3 motor's and P2 motor's maximum torque capacity, the torque is split such that the P3 motor aims for 70 percent of its maximum torque capacity and P2 motor aims to produce the remainder of the requirement divided by the gear ratio of the transmission.
8. Given the condition P3 motor speed is greater than 1500 rpm, and calculated shaft torque more than the combined torques of 70 percent of the P3 motor's and P2 motor's maximum torque capacity, the torque is split such that the P2 motor aims for its maximum torque capacity divided by gear ratio and the P3 motor aims for its maximum.

As shown above, the torque splitting is based not only on the torque capacities of the two motors but also the gear ratio of the transmission as well as the P3 motor speed. Similarly, pertinent factors and parameters are used for the preliminary torque-splitting in every drive mode. These factors are found with the help of manufacturer's data sheets and through UWAFT's observations. Currently, the torque control is based on an open loop estimate of calculated shaft torque that depends on an acceleration map. It does not look at the torque feedback and UWAFT strives for closed loop controls in the future. As more data is obtained and further calibrations are done, there will be further iterations and modifications run before the optimised torque control strategy is finalised. However, for now, the structure for a dynamic HSC is in place that equips UWAFT with the ability to clearly visualise how further development must proceed as well as gives room for future improvements.

Further, the *Drive Mode Aim* subsystem calculates the desired drive mode the vehicle must achieve given the information of the faults, the previous drive mode aim, computed torque splitting between the components and other input signals. The transitions between the different drive modes are also handled here as shown in Figure 20.

Table 5: Drive mode transition numbers with their meanings

Transition number	Corresponding transition initiated	
	From	To
1	P3EV	FullEV
2	FullEV	P3EV
3	EV	P3Hybrid
4	P3Hybrid	EV
5	EV	FullDrive
6	FullDrive	EV
7	P3Hybrid	FullDrive
8	FullDrive	P3Hybrid

In order to find the suitable transition, certain calculations based on calculated shaft torque request is processed. For example, if the UWAFT vehicle must undergo transition 1, which corresponds to shifting from drive mode *P3EV* to *FullEV* as shown in Table 5, certain conditions that are ascertained by algorithms within the HSC code are met.

- There is no transmission fault that withholds the transmission from getting engaged AND
- There is no fault stopping the P2 motor from functioning AND
- The calculated shaft torque request for more than 2s is above 70% of P3 maximum torque capacity OR
- The calculated shaft torque request is above 85% of the P3 maximum torque capacity

Like the torque-splitting strategy that is in the preliminary stage, even the numbers provided here can change as more tests are run and new information is processed by UWAFT. In order to help with testing, the HSC code makes use of overrides and this is sustained within drive mode transitions as well. The Stateflow chart in Figure 20 is a better graphical representation tool than their Simulink counterparts. It can also be noticed that transition 5 is not in its intended spot of moving from *EV* to *FullDrive,* instead that transition has been intentionally not been permitted. This is because sufficient testing of the code for clutch rev-matching with a moving P2 motor had not been conducted.

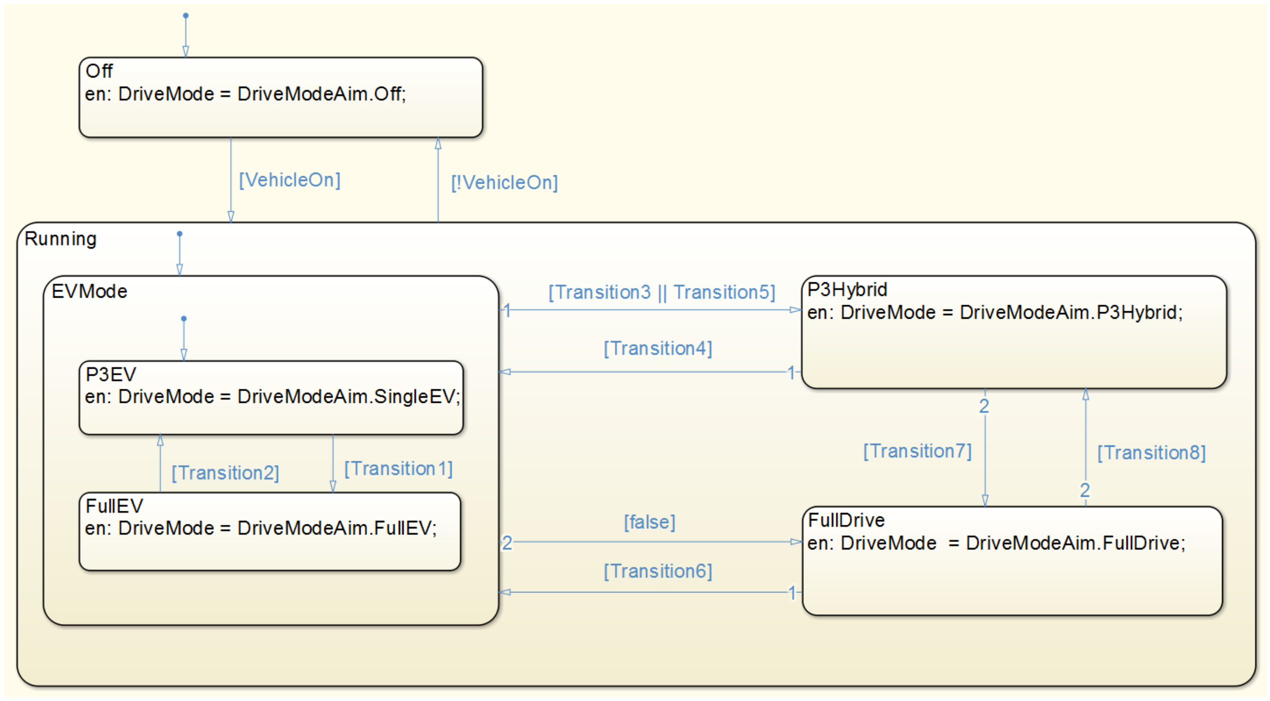

Off
en: DriveMode = DriveModeAim.Off;
[VehicleOn]
[!VehicleOn]
Running
EVMode
P3EV
en: DriveMode = DriveModeAim.SingleEV;
[Transition2]
[Transition1]
FullEV
en: DriveMode = DriveModeAim.FullEV;
[Transition3 || Transition5]
[Transition4]
[false]
[Transition6]
P3Hybrid
en: DriveMode = DriveModeAim.P3Hybrid;
[Transition7]
[Transition8]
FullDrive
en: DriveMode = DriveModeAim.FullDrive;

Figure 20: Stateflow chart representing the transitions of the different drive modes

'*Drive Mode Parsing*' subsystem uses the computed drive mode aim state to translate it to states for each of the three components that are key for the interplay between the powertrain components in the UWAFT architecture – Transmission, Clutch, and the Engine. Commands in the form of flags are sent to engage the transmission, engage the clutch or request to run the engine.

'*Wheel Torque State Control*' subsystem uses information about the transmission state to suitably find the torque mode and direction the P3 motor must follow. It also commands for a neutral override to disengage pre-transmission components from the rest of the driveline and also protects the transmission from getting damaged. The different P3 torque modes range from producing no torque to producing torque for forward propulsion or reverse propulsion depending on the shift lever position (PRNDL). Torque to reverse the vehicle is produced only by the P3 motor (of the three torque-producing components) since in all intended modes, the P3 motor stays on and there was no customer use requirement that required more torque than that can be produced by the P3 motor. This way inefficiencies from the transmission are reduced during reverse and there is overall robustness in operation.

The torque modes followed by the pre-transmission powertrain components P2 motor and engine is determined by the '*PreTrans Torque Control*' subsystem. A number of input signals are required to compute these torque modes which include but are not limited to the status of the engine, the clutch and the transmission. Depending on the torque modes each of the components are operating in, the '*Parameter Determination*' subsystem provides values for the maximum and minimum torque for P3 and P2 motors. Additionally, values for the P2 PID controller are computed. These include set-point speed, maximum and minimum values for rate limiters and the different proportional, integral and derivative terms in the PID. Similarly, values for the PID on the engine controller with corresponding values for MAP is also determined.

Chapter 5
On-Road Testing

The initial chapters in the thesis explains the theory and methodology for developing the HSC code along with some background on the UWAFT PHEV. The previous chapter discusses the actual HSC code developed by UWAFT. The validation of all the developed and integrated HSC code culminates in on-road testing. Since the UWAFT PHEV can drive, it is possible to collect data from the vehicle not just from within the controlled space of the garage (spinning wheels on the hoist) but also by driving on the road. In order to be able to analyse the functioning of the vehicle, it is important to be able to log data since all the details cannot be monitored real-time. The collection of vehicle data is also relevant for the purpose of comparing one run from another when the code has been modified to recognise and track particular differences. Many changes that are made to the code rely on engineering intuition and information processed until that point and data alone can confirm if the hunch is right or wrong. Given below is the hardware that is used by UWAFT to interface and log data.

5.1 CAN interfaces

During on-road testing, the main interface used by UWAFT is the data logger and real-time CAN bus monitor – Vector CANcaseXL log [41]. This data logger enables users to interface with up to two CAN buses per device via DB9 connectors. These can be plugged directly into the CAN bus either using the provision of connectors to each of the six channels (under the passenger seat) or with the help of an OBD-II to DB9 adapter cable. The user can record the data either directly on to a removable SD card mounted in the data logger itself or via a USB connected to a host PC. The latter requires the help of the software Vector CANoe [42]. This software is capable of producing real-time graphs along with their values in numbers for the selected signals that are being monitored on the CAN bus. This visual representation of signals simplifies the understanding of trends to make quick decisions if variables need to be manipulated. Although the CANcaseXL log is able to send and receive CAN signals, UWAFT mainly uses it for data logging and supervising interesting signals.

Another interface often used during these VIL runs is real-time CAN bus monitoring via CDNG. This addition during on-road testing is particularly prevalent when certain overrides are to be used during testing. The dSPACE software CDNG is interfaced to the host PC via an Ethernet cable and is connected to the MABx from where the CAN buses are observed. This setup requires the dSPACE license keys' dongles to run the software on the PC. Both the aforementioned interfaces require setting the devices up for proper functioning. There is more information that can be found for installation, training and using the interfaces mentioned, but this is not within the scope of this thesis.

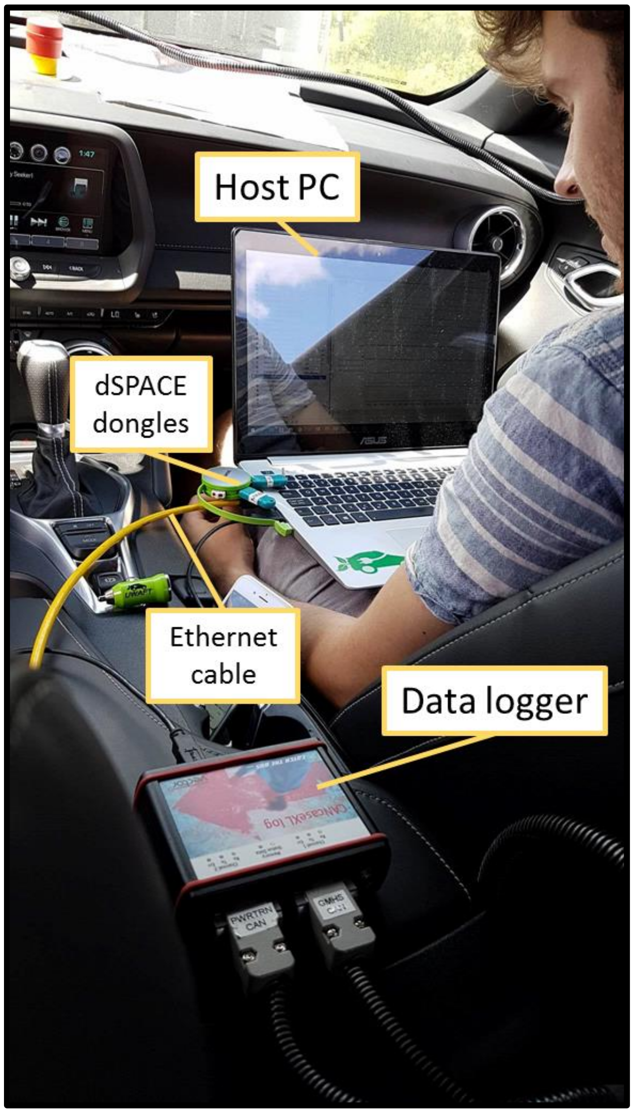


Figure 21: The setup used by UWAFT for on-road testing.

Photographed by Cole Powers in UWAFT PHEV (July 2017)

UWAFT routinely logs drive data in order to better understand the capabilities of the powertrain at a system level, to deem if a test is successful and to ascertain subsequent repeatability of that success. Figure 21 shows the setup with the data logger connected to the DB9 connectors (labelled GMHS and PWRTRN) and the host PC along with the MABx itself (Ethernet cable connects to MABx). In the examples given below for validation of the HSC code, the same drive log for the same duration of time has been used.

5.2 Validation of developed HSC code

5.2.1 Transitions in drive modes

As explained in sections 3.2 and 4.3.1, the UWAFT PHEV transitions through a number of drive modes. The partial drive log from one of the tests run is shown in Figure 22 to show in part the successful implementation of the developed HSC code. The first signal represented in the log is the *Drive Mode Aim* which as the name suggests is the drive mode the vehicle targets to reach. From the log, it can be seen that the vehicle is ready to be driven after 15s of the data logging being started when the drive mode aim reaches *P3EV*. Once the vehicle aims for *P3EV* drive mode, the neutral override command is passed. This helps disengage the powertrain pre-transmission from the rest of the driveline and allows only the P3 motor to deliver necessary torque to the wheels. At around 21s, the calculated shaft torque request, as a consequence of driver request for acceleration, increases. The vehicle takes an extra second before slowly moving.

The driver requests for more torque to speed the vehicle up, and the drive mode shifts to *FullEV* drive mode to meet the torque requirements which the P3 motor alone cannot satisfy. This is transition 1 and all the conditions mentioned in section 4.3.1 are met for it to occur. Since the transmission needs to get engaged at this stage for both the motors to deliver torque, the neutral override command is no longer high. Although the shaft torque request falls in about 4s of the drive mode aim being in *FullEV*, the drive mode aim is maintained for the 6s when the demand is less than 60% of the P3 motor maximum torque capacity (condition for transition 2) before returning to *P3EV*.

At 36s when shaft torque request increases again, the drive mode aim moves back to *FullEV* and then *P3Hybrid* drive modes. The condition for transition specifies that it occurs only when the ESS SOC falls below 20% (which it does not from the drive log) or if the CS switch is turned on. However, neither of this was the case. However, as revealed in Figure 20, when this code was run, the vehicle was not allowed to transition from *FullEV* to *FullDrive* (transition 5) directly. Hence, when the driver demand resulted in shaft torque request more than the total of 80% of P3 and 15% of P2 maximum torque capacities, transition 5 conditions were met for the drive mode to move to *P3Hybrid*. However, before conditions for transition 7 could be met (for *P3Hybrid* to FullDrive), the shaft torque request dwindled and transitions 4 and 2 could be met to arrive back at *P3EV*.

From a snippet of drive log of about a minute that displays only a tiny fraction of the total signals, the preceding information could be inferred. This reflects the importance of data logging and post processing not only to validate the developed code but also to imbibe more information to provide insight for future modifications. Within this minute of the log, the ESS SOC falls by barely a percentage.

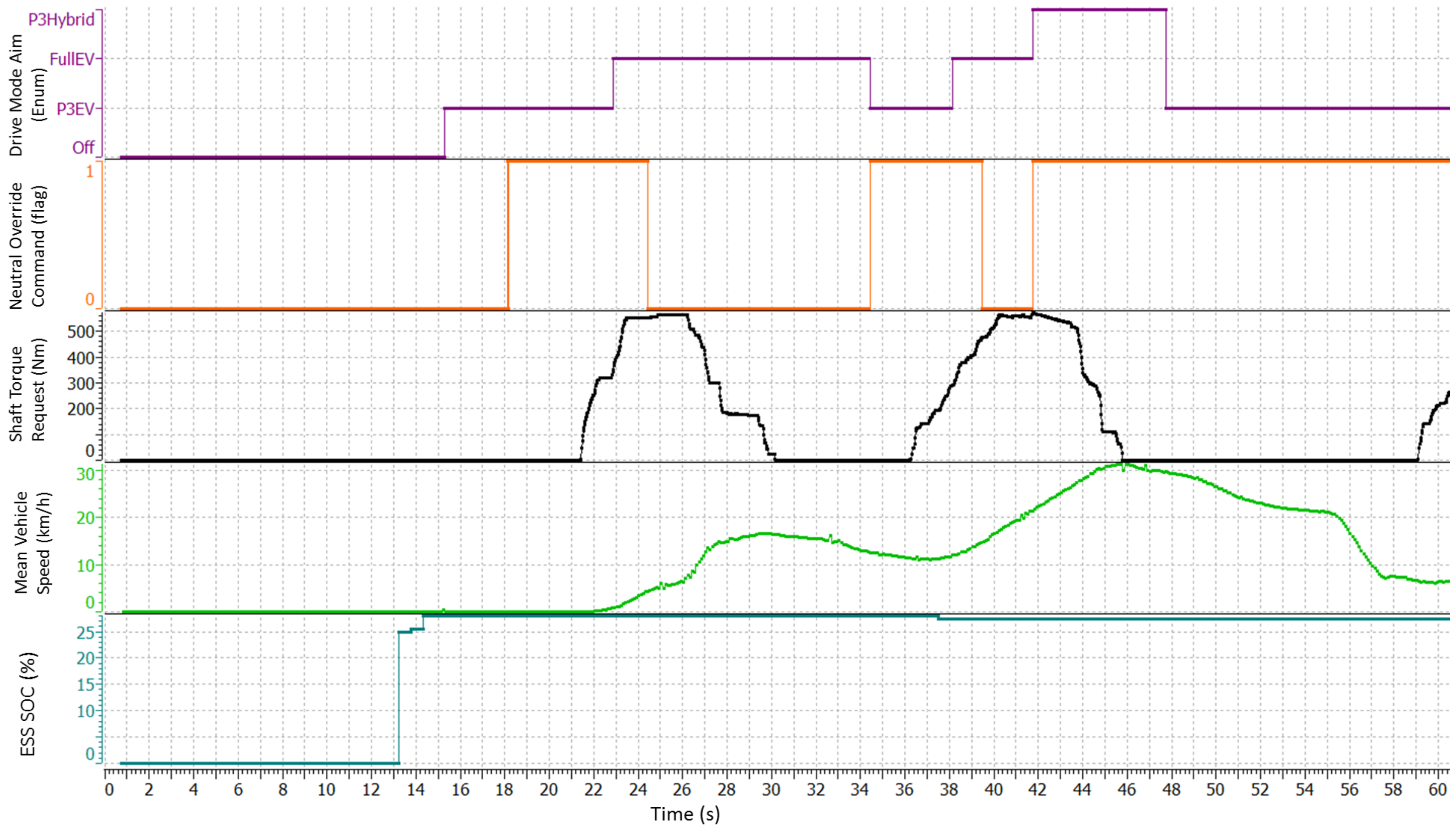


Figure 22: Recorded drive log[1] indicating transitions of the drive mode

1 Corresponding names of signals in the code from top to bottom: *DriveModeAim*, *Neutral_Override_Cmd*, *CalcShaftTrqReq*, *VehSpdAvgDrvn*,and *bcm_soc*.

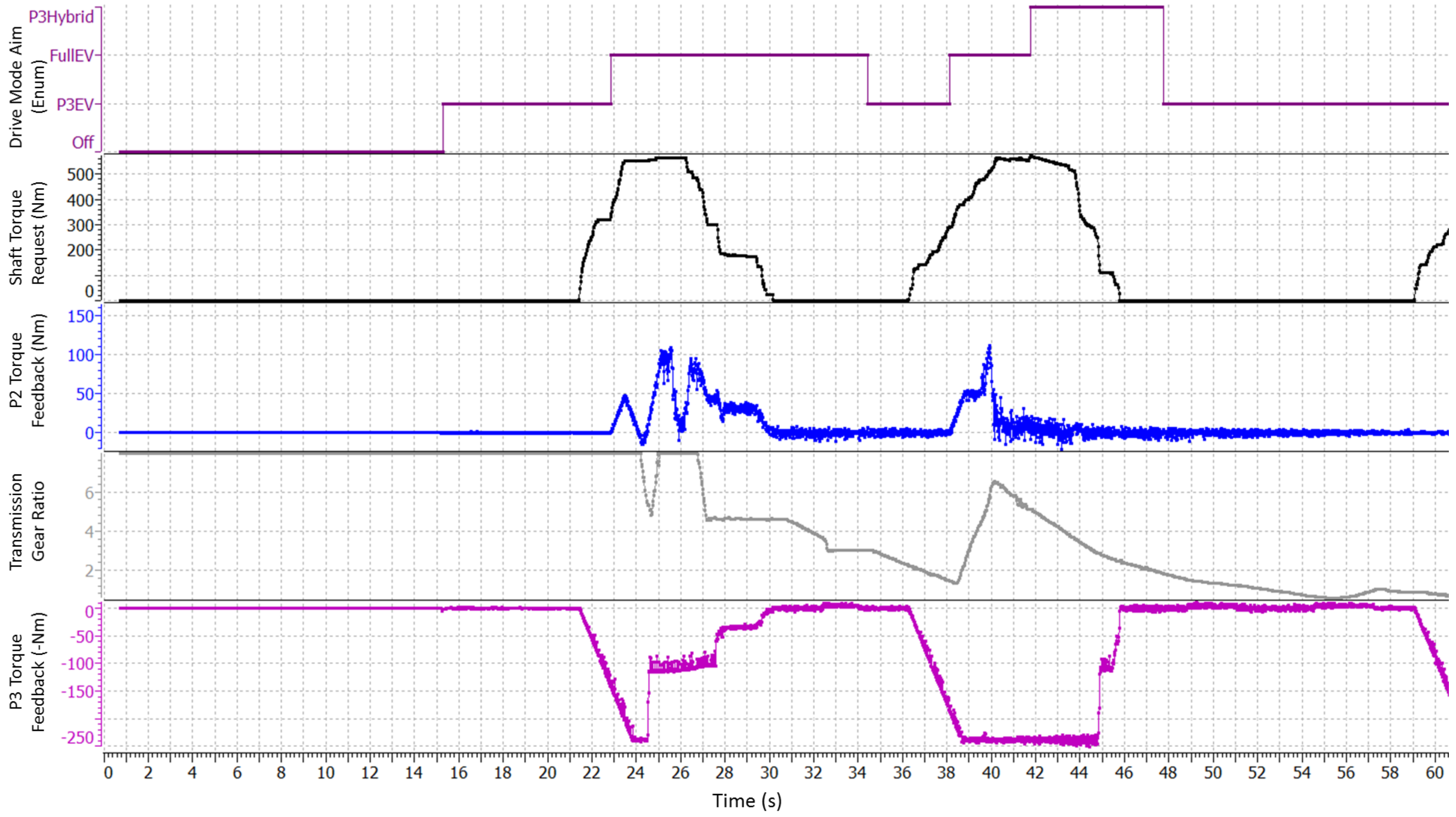


Figure 23: Drive log[2] representing torque split between P2 and P3 motors

2 Corresponding names of signals in the code from top to bottom: *DriveModeAim*, *CalcShaftTrqReq*, *P2_D2_Torque_Feedback*, *TranGrRat*, and *P3_D2_Torque_Feedback*.

5.2.2 Torque splitting

Although the torque splitting done in the HSC is in the preliminary phase, Figure 23 establishes that the algorithm works sufficiently to provide torque to the wheels. The rules for torque splitting are based on the data sheets provided by manufacturers or bench tested by UWAFT for individual torque producing components as mentioned in section 4.3.1. Further refining of the code requires more testing and tweaks. In Figure 23, P3 torque feedback still represents produced torque. The P3 motor is mounted opposite to how the P2 motor has been mounted and hence the negative values in the P3 torque feedback graph. As explained in several sections, shaft torque request provides the basis for the HSC to calculate the drive mode aim as well as the torque targets for the P2 and P3 motors. Once the *Executive* subsystem actuates the motors appropriately, the torque produced powers the wheels. The feedback from the motors along with the vehicle speeds that are attained due to these torques have been shown above.

In *P3EV* when the shaft torque request increases, the P3 motor ramps its torque until it gets saturated at around 240 Nm. As soon as the drive mode reaches *FullEV*, P2 motor too progressively increases the torque delivered. However, the variation in the P2 torque delivered as seen in Figure 23 is attributed to the variation in the gear ratio of the transmission which affects the P2 torque aim as shown in Table 4. Thus, the drive log snippet proves that the developed HSC code is valid according to preliminary analysis. Due to the similar slopes each time the P3 motor ramps, it can be inferred that a rate limiter on the P3 torque delivered exists. With experience, analysis of data becomes easier and trends can be identified such that more than obvious information can be deduced.

Drive logs of pre-determined drive cycles is not available at the moment but can be collected in the near future. By running the same drive cycles within the different testing environments such as SIL and HIL, the expected vehicle behaviour from each environment can be compared for further analysis. This also helps to understand the factor to which the tests run in other environments is valid. UWAFT needs to observe how closely the actual vehicle follows drive cycles, especially during the Emissions and Energy Consumption drive cycle provided by EcoCAR 3 competition.

Chapter 6
Conclusions and Recommendations

This thesis describes how the University of Waterloo Alternative Fuel's Team (UWAFT) develops its hybrid supervisory controller (HSC) for its re-engineered Plug-in Hybrid Electric Vehicle (PHEV) with a focus to incorporate safety. UWAFT's choice of a split parallel architecture in the EcoCAR 3 competition has its benefits. This can be in the form of options provided for finding a drive mode that fulfils the driver demand while prioritising energy efficiency and minimising emissions or meeting driver's requirement of enhanced performance so that the thrill of driving is prioritised. Another benefit of choosing this relatively challenging architecture is that it has been an amazing teaching tool and very eloquently meets UWAFT's mission and vision for its team members. The struggles for integration and testing has given its members not only hands-on experience but an appreciation for other PHEVs and vehicles. This thesis helps understand a student team's journey in building the code for its HSC using model-based design. The language is kept basic enough for someone new to the topic of HSC to understand the nuances discussed yet introduces tools used in the process and explains sufficiently the lessons learned. UWAFT members can, in particular, use this as a training tool for future students participating in Year 4 as well as in Advanced Vehicle Technology Competitions to come. From the different methodologies used to incorporate safety in the development work for the HSC, the Failure Modes Effects Analysis (FMEA) and Fault Tree Analysis (FTA) were found to be useful in the design phase. As the development progressed, Hazard Analysis and Risk Assessment (HARA) methodology with Automotive Safety Integrity Level (ASIL) as the risk rating technique was used more frequently.

Some suggestions for bettering the modeling and controls development in the future are:

- The control strategy for thermal management is quite basic at the moment. It needs to be developed such that the HSC diagnoses the thermal management system and has been tested enough for remedial actions to be confidently implemented. This will ensure that the UWAFT PHEV meets the lower temperature requirements of the power electronics and other electric components involved as compared to a conventional car. Establishing a baseline to better control and implement strategies once the AC compressor works reliably can help quantify engineering decisions. Since the vehicle performance and fuel economy is coupled to the thermal management, this work needs to be prioritised.
- Update the existing complex vehicle model. It is unable to satisfactorily account for interactions or handle transients. Although more data is available now to update the different plant models, certain

components are far from realistic (referring *UWAFT_Master* model) and sometimes even incorrect. Hence, it is unable to satisfactorily contribute to further develop the HSC code.

- Maintain and update safety documents for test case development. Test! Test! And Test! Data collected can be used for comparison to ensure that the goals for the vehicle controls are met. Implement automated testing so that more test coverage can be accomplished and it can help with regression testing for when code changes are made. The groundwork for this has been laid but test case development and documentation needs more oversight, especially to ensure system safety.
- Develop more distributed controllers to offload the HSC and for reliable functioning of interfaces through CAN-enabled boards such as CAN-enabled relay boards. This way the IOs on HSC can be used for more development work. When the IOs are not overloaded, safety and good performance of the vehicle can be ensured.
- Model a simplified full vehicle model for the team to run iterations of optimisation and perform thematic analysis as the HSC code develops to efficiently optimise vehicle performance, lower emissions and maximise fuel economy. From purely the competition perspective, optimization work for HSC code needs to be streamlined for lower emissions.

All the work stated is possible only through good project management and human resource management. On-boarding more CSMS members is an obvious thing to do. However, effectively training them by ably maintaining their interest takes skill and dedicated time. This can be done through on-going tutorials or workshops and other structured meetings run by senior CSMS members. The benefits from the extra effort can be reaped during crunch time when testing, analysing, re-coding and documenting work for deliverables are to occur simultaneously. Having more CSMS members work on the HSC development introduces challenges around maintaining the integrated HSC code and ensuring effective version control. Appropriate project management to make processes as seamless as possible will help with time management to test newer innovative solutions and prevent overall burning out of the team members.

Bibliography

[1] R. B. R. W. Robert L. Hirsch, “PEAKING OF WORLD OIL PRODUCTION: IMPACTS, MITIGATION, & RISK MANAGEMENT,” DOE, 2005.

[2] O. Milman, “Climate Change,” theguardian, 30 August 2016. [Online]. Available: https://www.theguardian.com/environment/2016/aug/30/nasa-climate-change-warning-earth-temperature-warming. [Accessed 10 July 2017].

[3] United States Environmental Protection Agency, “Inventory of US Greenhouse Gas Emissions and Sinks,” EPA, 2017.

[4] A. V. Angelique Chrisafis, “Automotive Industry,” theguardian, 6 July 2017. [Online]. Available: https://www.theguardian.com/business/2017/jul/06/france-ban-petrol-diesel-cars-2040-emmanuel-macron-volvo. [Accessed 10 July 2017].

[5] E. Shaal, “7 Automakers with Major Electric Vehicle Programs in the Works,” FleetCarma, 20 September 2016. [Online]. Available: http://www.fleetcarma.com/7-automakers-electric-vehicle-programs/. [Accessed 10 July 2017].

[6] Argonne National Laboratories, “Sponsors,” [Online]. Available: http://ecocar3.org/wp-content/uploads/2015/01/Sponsor-Brochure_1-28-15.pdf. [Accessed 2nd April 2016].

[7] S. Kundu and M. W. Fowler, “Use of Engineering Design Competitions for Undergraduate and Capstones Projects,” *Chemical Engineering Education,* vol. 43, no. 2, pp. 131-136, 2009.

[8] E. Schmidt, “Electric Vehicle Sales in Canada: Q1 2017,” 25 May 2017. [Online]. Available: http://www.fleetcarma.com/electric-vehicle-sales-canada-q1-2017/. [Accessed 12 July 2017].

[9] S. Trochaniak, “Electric Vehicle Sales In The United States: 2016 Final Update,” 19 January 2017. [Online]. Available: http://www.fleetcarma.com/ev-sales-usa-2016-final/. [Accessed 19 July 2017].

[10] A. K. Koch, “Hybrid Controls Development and Optimization of a Fuel Cell Hybrid Powertrain ,” University of Waterloo, 2012.

[11] J. K. E. W. D. N. Robert Jesse Alley, “Hybrid Architecture Selection to Reduce Emissions and Petroleum Energy Consumption,” *SAE 2012 World Congress & Exhibition,* no. 10.4271/2012-01-1195, 2012.

[12] M. D. R. X. Y. A. I. U. D. S. J. C. K. Zhuoran Zhang, “Hybrid Electric Vehicle Architecture Selection for EcoCAR 3 Competition,” *SAE 2015 World Congress & Exhibition,* no. 10.4271/2015-01-1228, 14 April 2015.

[13] E. W. P. C. M. A. K. L. S. D. N. David Ord, “Powertrain Design to Meet Performance and Energy Consumption Goals for EcoCAR 3,” *SAE 2014 World Congress & Exhibition,* no. 10.4271/2014-01-1915, 2014.

[14] Z. Z. H. W. K. d. P. J. C. K. Miriam Di Russo, “Modeling, Simulation and Control Development of a Pre-Transmission Parallel E85 PHEV for Year-1 of EcoCAR 3 Competition,” *SAE 2016 World Congress and Exhibition,* no. 10.4271/2016-01-1256, 2016.

[15] M. L. Jonathan D. Cox, “Specification of a P3 Parallel Hybrid Electric Vehicle Architecture for the EcoCAR 3 Competition,” *SAE 2016 World Congress & Exhibition,* no. 10.4271/2016-01-1245, 2016.

[16] A. K. J. W. M. Y. S. M.-M. Sam Yacinthe, “Development of the Design of a Plug-In Hybrid-Electric Vehicle for the EcoCAR 3 Competition,” *SAE 2016 World Congress & Exhibition,* no. 10.4271/2016-01-1257, 2016.

[17] J. C. J. W. Alex K. Gibson, “A Comparison of the Mississippi State University EcoCAR 3 Team Vehicle Architecture Motor Control Strategies,” *SAE 2016 International Powertrains, Fuels & Lubricants Meeting,* no. 10.4271/2016-01-2217, 2016.

[18] D. K. K. A. S. K. Z. D. C. C. Daniel Prescott, “Implementation of Series-Parallel Multiple-Regime Vehicle Architecture Using 2013 Chevrolet Malibu Platform,” *SAE/KSAE 2013 International Powertrains, Fuels & Lubricants Meeting,* no. 10.4271/2013-01-2493, 2013.

[19] A. H. M. Y. M. Y. M. O. B. H. J. W. Katherine Bovee, “Plant Modeling and Software Verification for a Plug-in Hybrid Electric Vehicle in the EcoCAR 2 Competition,” *SAE 2015 Congress & Exhibition,* no. 10.4271/2015-01-1229, 2015.

[20] M. R. T. C. M. P. D. I. D. E. S. Ryan Howell, “The University of Tennessee's EcoCAR 2 Final Design Report,” *SAE International Journal of Alternative Powertrains,* no. 10.4271/2012-01-1771, 2012.

[21] M. C. E. E. J. L. M. K.-L. T. S. B. W. R. A. F. M. F. Gurhari Preet Singh, “The University of Waterloo Alternative Fuels Team's Approach to EcoCAR 2,” *SAE 2012 International Powertrains, Fuels & Lubricants,* no. 10.4271/2012-01-1761, 2012.

[22] M. Giannikouris, “Design and Control of a Unique Hydrogen Fuel Cell Plug-In Hybrid Electric Vehicle ,” University of Waterloo, Waterloo, 2014.

[23] Argonne Laboratories, *EcoCAR 3 workshops,* 2014-2017.

[24] U.S. Department of Labor Occupational Safety and Health Administration, “Job Hazard Analysis OSHA 3071,” 2002. [Online]. Available: https://www.osha.gov/Publications/osha3071.pdf. [Accessed 17 10 2016].

[25] Wikipedia, “Automotive Safety Integrity Level,” [Online]. Available: 25.

[26] S. K. a. P. G. P. Metz, “ISO 26262 in Practice – Resolving Myths with Hazard & Risk Analyses,” [Online]. Available: https://www4.in.tum.de/lehre/vorlesungen/funktionale_sicherheit/WS14/04_%5bMKG13%5d.pdf. [Accessed 12 July 2017].

[27] B. Marick, “Mew models for test development,” [Online]. Available: http://www.exampler.com/testing-com/writings/new-models.pdf. [Accessed 25 June 2017].

[28] J. B. R. H. M. (. Z. P. P. S. C. Leon Osborne1, “Clarus Concept of Operations, Intelligent Transportation System, US Department of Transportation,” October 2005. [Online]. Available: https://ntl.bts.gov/lib/jpodocs/repts_te/14158.htm. [Accessed 22nd June 2017].

[29] “CAN bus,” Wikipedia. [Online]. [Accessed 29 June 2017].

[30] N. N. a. F. Simonot-Lion, “In-vehicle communication networks - a historical perspective and review,” University of Luxembourg, 2013.

[31] National Instruments, “Controller Area Network (CAN) Overview,” 01 August 2014. [Online]. Available: http://www.ni.com/white-paper/2732/en/. [Accessed 23 June 2017].

[32] R. F. M. F. D. V. B. G. Patrick Ellsworth, “Control Analysis for Efficiency Optimization of a High Performance Hybrid Electric Vehicle with Both Pre and Post Transmission Motors,” *SAE International,* no. 2016-01-1253, p. 4, 2016.

[33] Brusa, [Online]. Available: http://www.brusa.biz/fileadmin/template/Support-Center/Datenbl%C3%A4tter/BRUSA_DB_EN_NLG513.pdf. [Accessed 29 June 2017].

[34] UWAFT, *Deliverables submitted to competition,* UWAFT, 2014 - 2017.

[35] dSPACE, “RTI CAN MultiMessage Blockset,” [Online]. Available: https://www.dspace.com/en/inc/home/products/sw/impsw/rti_can_multimessage_blockset.cfm. [Accessed 25 June 2017].

[36] dSPACE, “dSPACE Simulator Mid-Size,” [Online]. Available: https://www.dspace.com/en/inc/home/products/hw/simulator_hardware/dspace_simulator_mid_size.cfm. [Accessed 25 June 2017].

[37] Snap-On, “SOLUS™ EDGE User Manual,” [Online]. Available: https://www1.snapon.com/display/3871/SOLUSEdgeUserManual.pdf. [Accessed 04 July 2017].

[38] CANbus Academy, “CAN Bus Load Calculator,” [Online]. Available: https://www.canbusacademy.com/resources/can-bus-load-calculator/. [Accessed 28 June 2017].

[39] “GitHub,” [Online]. Available: https://github.com/. [Accessed 10 July 2017].

[40] “SourceTree,” [Online]. Available: https://www.sourcetreeapp.com/. [Accessed 10 July 2017].

[41] Vector Informatik GmbH, “Manual, CANCaseXL log,” [Online]. Available: https://vector.com/portal/medien/cmc/manuals/CANcaseXL_Manual_EN.pdf. [Accessed 04 July 2017].

[42] Vector Informatik GmbH, [Online]. Available: https://vector.com/vi_canoe_en.html. [Accessed 07 July 2017].

[43] MathWorks, “Simulation and model based design,” MathWorks, [Online]. Available: https://www.mathworks.com/products/simulink.html. [Accessed 10 July 2017].

Appendix A – UWAFT specific

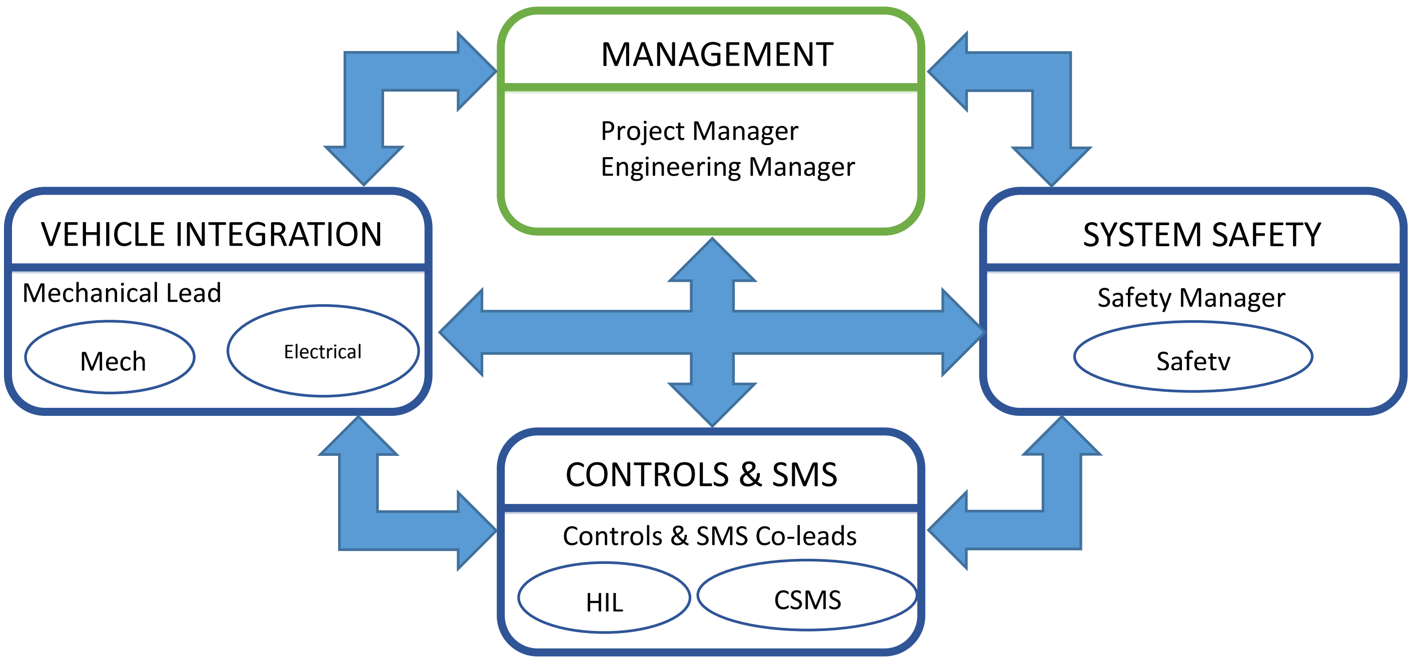


Figure 24: Team structure limited to subteams directly involved with vehicle development process

It is important to note that this representation pertains to only the sub-teams that are directly related to the development of the vehicle until Year 3 of the four-year challenge. UWAFT also comprises of the Advanced Driver Assistance Systems (ADAS) team, Communications team and Innovation team, none of which have been represented as they do not concern this thesis.

Table 6: Table of references for further reading

Year of Competition	**Name of Deliverable/Resource**	**Quick Description**
Y1	Architecture Selection Report	Explains the 4 architectures that were suggested and analysis using basic flow and effort modeling (Autonomie).
Y2	Vehicle Design Report	Pages 19 – 36 to know how CSMS things were mid Y2; explains start-up algorithms for certain components
Y2	Final Technical Report	First round of engine dynamometer test
Y3	Fall Swimlane Report	SMS part covers test case development and SIL-HIL runs
	AAE modules	Basics and buzzwords to increase theoretical knowledge
Y1-Y4	Workshops	Presentations – CSMS; Material provided by sponsors
	Document in Resources on Git	Diagnostics written up by Michael Wu
Y1 – Y4	Papers and theses	Produced by UWAFT members, faculty advisors will have an exhaustive list
	Model-Based Design for Vehicle Powertrains	https://www.nterlearning.org/web/guest/course-details?cid=3728
	Mathworks resources	
	dSPACE webinars	

The order of the resources is not in order of importance or chronology and is provided only as an asking point to senior members for resources. They can choose the best supporting material for your target work.

Appendix B – UWAFT architecture

Table 7: Vehicle Technical Specification (VTS) for UWAFT

Specification	Units	Targets	Requirement	Team VTS
Acceleration, IVM-60mph	sec	5.82	7.9	5.6
Acceleration, 50-70mph (Passing)	sec	6.6	9.9	3.2

Braking, 60-0mph	ft.	121.4	135	121
Acceleration Events Torque Split (Fr/Rr	%	0/100	49/51	0/100
Lateral Acceleration, 300ft. Skid Pad	G	0.84	0.80	N/A
Double Lane Change	mph	54.4	52	N/A
Highway Grade ability, @60 mph for 20 mins	%	6%	6	7%
Cargo Capacity	ft^3	2.4	2.4	2.4
Passenger Capacity	#	4	2	4
Curb Mass	kg		1938	1824
Starting Time	s	5	15	N/A
Total Vehicle Range	mi	187	150	188.78
CD Mode Capability	Blended/EV	EV Only	N/A	EV Only
CD Mode Range	mi	22.37	N/A	33.42
CD Mode Total Energy Consumption	Wh/km	267.8	N/A	223.44
CS Mode Fuel Consumption	Wh/km	668.7	N/A	467.75
UF-Weighted Fuel Energy Consumption	Wh/km	736.6	N/A	232.71
UF-Weighted AC Electric Energy Consumption	Wh/km	23.8	N/A	116.87
UF-Weighted Total Energy Consumption	mpgge	26.46	25	57.39
UF-Weighted WTW Petroleum Energy Use	Wh PE/km	621	750	67.92
UF-Weighted WTW Greenhouse Gas Emissions	g GHG/km	222.6	250	113.93
UF-Weighted Criteria Emissions	g/km	2.64	-	0.402

Table 8: Powertrain Components

Component	Manufacturer/Model	Specifications
Engine	Weber MPE 850cc Turbo	PP: 92 kW
P2/3 Motor	GKN AF130-4	PP: 70 kW (300 V)
		PP: 70 kW (300 V)
Battery	A123 6x15s3p	Cap: 16.2 kWh
		N Volt: 292 V
Transmission	GM AT	8 Gear, Longitudinal Auto
Final Drive	GM 3.73 Final Drive, eLSD	3.73 : 1 Final Drive Ratio

Note: PP: Peak Power, PC: Peak Current, CP: Continuous Power, Cap: Capacity, N: Nominal, eLSD: Electronic Limited-slip differential

Table 9: UWAFT ECUs

ECU	Description
P2/P3 Inverter	Converts torque demands into three phase power for P2 and P3 motors
MoTeC M400	Controls engine and monitors all parameters
Clutch Actuator	Controls position of custom clutch using J1939 CAN protocol
MicroAutoBox II	Provides overall control of all added components
BCM (Battery Control Module)	Monitors battery and high voltage bus status, controlled via CANBUS
Denso AC Compressor	Uses PWM signal to control AC compressor
Relay Box	Controls power to components with inputs from the MicroAutoBox II
Brusa NLG5	Commands and monitors power to the battery when charging

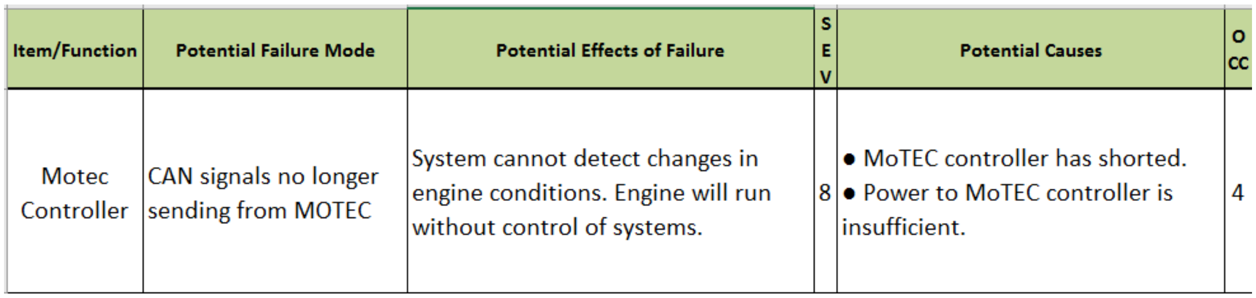
Table 10: GM ECUs

ECU	Description
TCM	Controls when transmission will shift, and provides protection
ECM	Used to control stock V6 engine

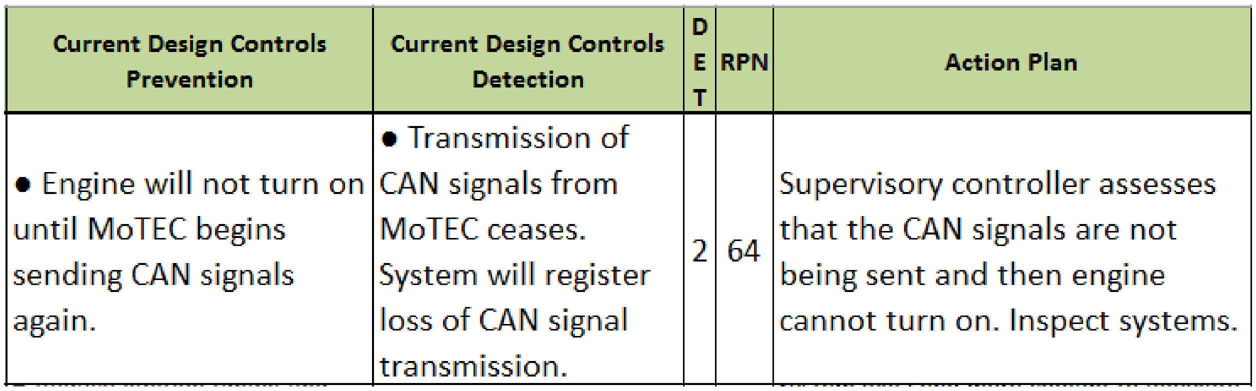
BCM (Body Control Module)	Allows accelerator pedal position and other body components to be interfaced with on CAN bus
EPS	Controls power steering pump

Appendix C – Safety documentation

Table 11: Example DFMEA for one item showing RPN and remedial action



Item/Function	Potential Failure Mode	Potential Effects of Failure	SEV	Potential Causes	OCC
Motec Controller	CAN signals no longer sending from MOTEC	System cannot detect changes in engine conditions. Engine will run without control of systems.	8	● MoTEC controller has shorted. ● Power to MoTEC controller is insufficient.	4



Current Design Controls Prevention	Current Design Controls Detection	DET	RPN	Action Plan
● Engine will not turn on until MoTEC begins sending CAN signals again.	● Transmission of CAN signals from MoTEC ceases. System will register loss of CAN signal transmission.	2	64	Supervisory controller assesses that the CAN signals are not being sent and then engine cannot turn on. Inspect systems.

Table 12: Customer Use Case Requirement

Requirement ID	Requirement	Revision Date
1000	The customer shall be able to drive away within 10 seconds of entering the vehicle.	
1001	The customer shall be able to accelerate when the accelerator pedal is pushed down.	
1002	The customer shall be able to recharge the battery when plugged into the charging port.	

Table 13: Feature Requirement

Requirement ID	Requirement	Revision Date	Customer Use Case ID
2000	The instrument panel shall display "Apply Brake Pedal and Push Run to Start" within 10 seconds of the recognizing vehicle key.		1000
2001	The instrument panel shall illuminate the vehicle ready indicator when the vehicle is able to provide axle torque.		1000
2002	The accelerator pedal will depress as force is applied by the driver to increase the speed of the vehicle.		1001
2003	The instrument panel shall indicate the speed of the vehicle as the customer accelerates.		1001

Table 14 : Subsystem Requirement

Requirement ID	Requirement	Revision Date	Feature Requirement ID	Feature Requirement Revision Date	Requirement Updated	Feature	System

This test case is missing the corresponding requirement ID because that documentation was not maintained or cross-referenced. Another test case for accelerator pedal template is available in *Vehicle Design Report*.

Table 15: Test case example

UWAFT EcoCAR 3 Test Case Outline		
1.0	**Overview**	
1.1	Test Name	EPO torque restriction
1.2	Test Goal	To verify that torque demands are ramped down when an emergency power-off (EPO) is triggered.
1.3	Version	1.0
1.4	Last Revised	Nov. 6, 2016
1.5	Author(s)	Cole Powers & Jake McGrory
1.6	Description	The software model of the vehicle plant will trigger an EPO (ESS over temperature), the supervisory controller must respond such that the vehicle torque commands are removed.

1.7	Test Class	System Function
1.8	Environment	SIL
2.0	**Procedure**	
2.1	Initialization	An ESS_fault_insert block exists in the ESS SIL plant model. By setting the temp_fault constant to 1, a thermal runaway scenario is injected. Run the sch_cycle as usual.
2.2	Body	Inject this temperature fault at various times in the drive cycle.
2.3	Completion	Vehicle speed and torque demands to each powertrain component should be recorded. The temp_fault constant should be reset to zero after the test.
3.0	**Summary**	
3.1	Generated plot	See APPENDIX of *Fall Swimlane Report* [Y3 deliverable]
3.2	Expected result	The ESS plant should transmit a bcm_epo signal when its maximum temperature threshold is exceeded, and torque demand to the powertrain components should be ramped down or cut.
3.3	Actual result	When the maximum temperature threshold is exceeded, the torque and vehicle speed demands to the powertrain components are reduced significantly.
3.4	Consistency	N/A
3.5	Limitations	Once the temperature fault is injected, it takes roughly 40 seconds (simulation time) before the ESS temperature exceeds the threshold. Since it takes roughly 16% of the total simulation time before the effect propagates, it is difficult to gauge how the model behaves at the very beginning and very end of the simulation.
4.0	**Suggestions**	
4.1	Test-specific	Modify the ESS temperature so that it increases at a faster rate to allow for testing when T is less than 50 (simulation time).
4.2	Other	None.

Table 16: Safety level description for test cases

Safety Level	Level Description
Level 1	A level 1 safety requirement specifies that a fault has occurred that the vehicle will have to be shut-down.
Level 2	A level 2 safety requirement specifies that a fault has occurred and the 1 or more components will have to be shut-down.

Level 3	A level 3 safety requirement specifies that a fault has occurred and the MABx will have to limit the extent of component functionality. The fault cannot be removed until a key-cycle has been performed.
Level 4	A level 4 safety requirement specifies that a fault has occurred and the MABx will have to limit the extent of component functionality until a re-pass threshold has been met, at which point the component fault will be removed.
Level 5	A level 5 safety requirement specifies that a fault has occurred and certain non-powertrain features will be degraded and/or disabled.

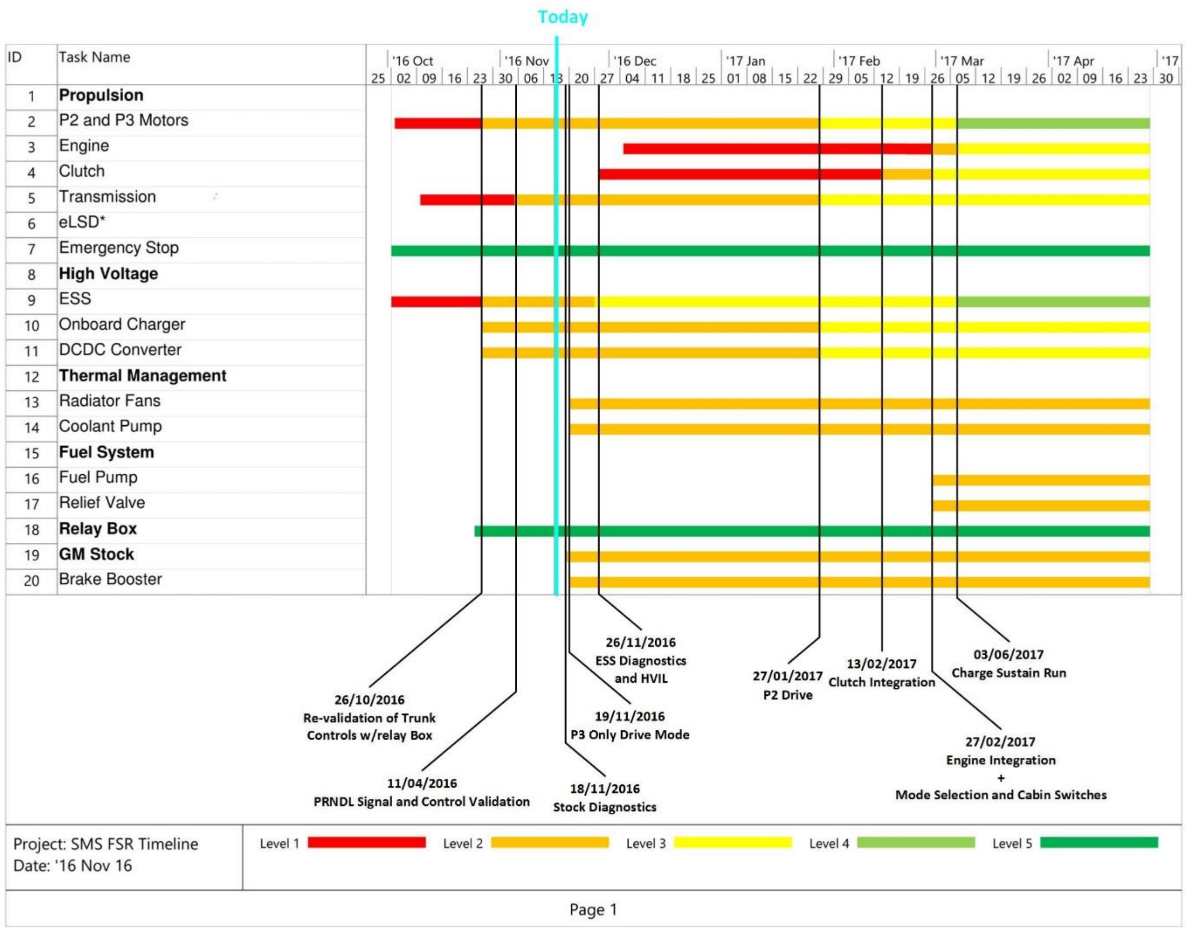


Figure 25: Year 3 timeline for test case development

Appendix D – Tips [23]

Troubleshooting tips

- If component does not function, check all the connections
  - Connectors can get pulled out of housing resulting in open circuits
  - Male pins can get bent out of shape
  - Repeated insertion and removal loosen connections
  - Loose grounds
  - Terminal connections not tight due to fatigue
  - Improper crimping of wires
- Continuity tests
  - Probe between power to chassis
  - Read wake line voltage when controllers ON
- Check if bus is functioning
  - Use Vector tool (CANcaseXL)
  - Check on CDNG
- Check bus integrity
  - by measuring bus resistance between CAN High and Low
  - by checking between component and HSC
- Ensure all wires on HIL bench are fused appropriately
- Heat fatigue or piercing resulting in shorting of wires is a real thing

Tips to develop soft ECUs

- Keep simple with room for future development. Requirements:
  - What functionality is being aimed for?
  - What faults should ECU be capable of injecting and how?
  - Does this ECU have to communicate with other ECUs?
  - Need for interfacing to test with HIL?
  - Do the signals originating from ECU need to be saturated? Any ranges?
- Is the developed code readable by other members?
- Test after each modification and document if possible
  - Use real data to compare the soft ECU's performance
  - If soft ECU does not have necessary functionality, do not be afraid to take the even simpler route and use constant blocks that can be manually manipulated
  - What are the outputs HSC is looking for? Sufficient to provide in Boolean with a flag or is the value important for HSC to make subsequent decisions?

Appendix E –

1 Powertrain Failure Scenario

1.1 Failure Scenario Definition

The failure scenario being investigated is a scenario that is covered in the requirements listed in NYSR, section B.5.1.11 in regards to the torque direction in a regenerative braking event. The system safety team has investigated the requirement and has defined the function of the feature and possible failure cases. This requirement details that the vehicle hybrid supervisory controller (HSC) must first register that a regenerative event is intended in the vehicle, after which the vehicle should attempt to use the motor to recover electrical energy by applying a negative torque in the braking event.

From this requirement, a failure scenario can be suggested, in which during a regen braking event any of the driving motors can apply torque in the accelerating direction. The HSC shall sufficiently detect if the torque developed on the motor or powertrain is positive. Due to the design of the vehicle architecture, the P3 motor is directly coupled to the driven rear wheels which will be the main target of this failure case. However, the P2 motor is not excluded from the possibility of being in a regen event, but this development is less prioritized compared to the P3 motor.

1.2 Code Development

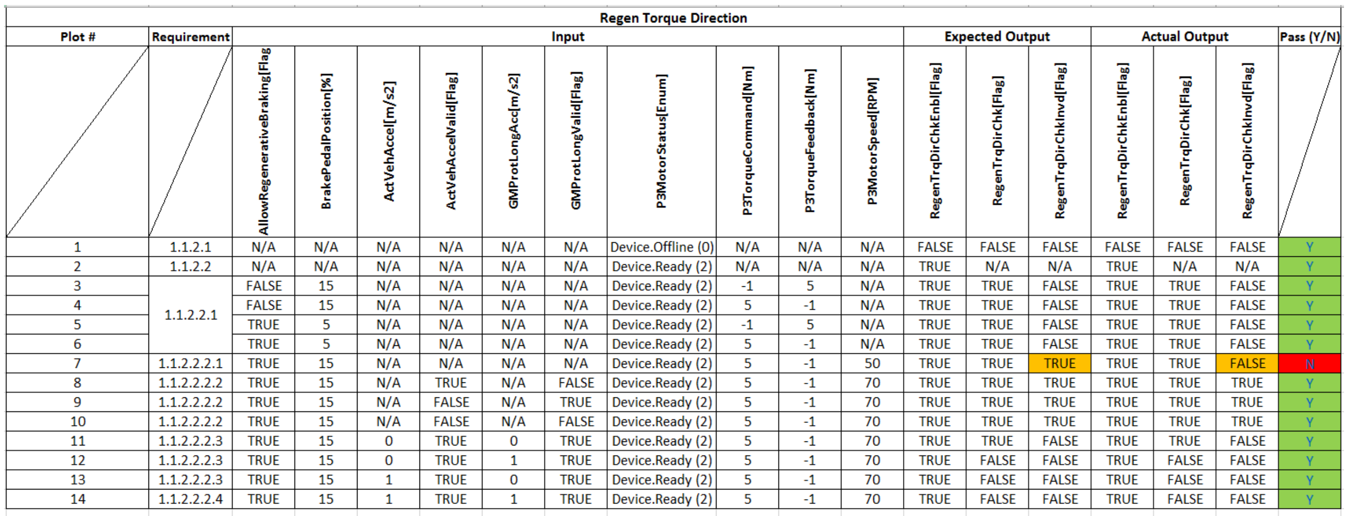
The failure case diagnostic algorithm has been through an initial iteration of development and testing, and the mitigation strategy is in the process of being transitioned from strategy to algorithm. The current diagnostic algorithm creates coverage on the case of a failure in the P3 self reported torque value during a regen braking event. The diagnostic checks will feed into a maturation algorithm that allows fault to be matured such that a valid mitigation action can be taken. The "detailed" software design for diagnosing this error as well as more detailed software views are attached in Figure 26, Figure 27and Figure 28.

More diagnostic algorithms need to be developed to detect cases such as the motors' self reporting torque mismatches, the commanded torque direction, as well as accounting for P2 regen torque, and the overall powertrain braking torque with consideration to grade. Additionally, further design failure mode effect analyses (DFMEAs) are needed to investigate the failure cases that can cause the failure scenario in question more specifically.

UWAFT is currently working on developing the mitigation actions if such fault scenario was to occur. Development of a vehicle limp mode is underway to address the degradation stage before removing drivetrain functionality, and to reflect the vehicle limp mode, a component limp mode scheme.

1.3 SIL HIL Analysis

The software that has already been developed has been unit tested within SIL and HIL. A sample of the SIL test results is shown Figure 26. The test cases document records the algorithm input values, desired and actual output values used when running such that the tests are replicable. The cases developed during the SIL stage are aimed specifically at each of the software design requirements and the tests indicate which of requirements of the diagnostic is not being met.

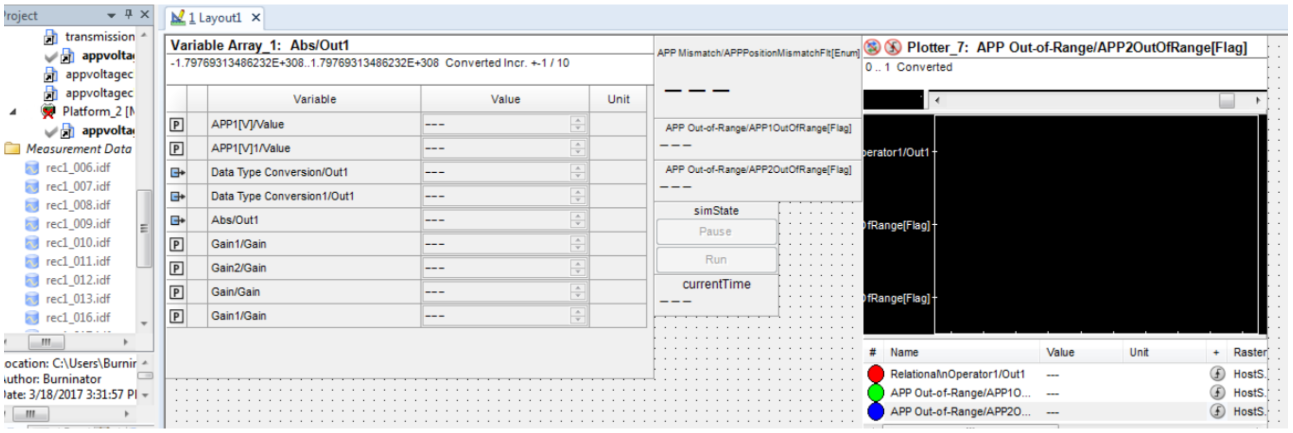


Regen Torque Direction																		
Plot #	Requirement	Input										Expected Output			Actual Output			Pass (Y/N)
		AllowRegenerativeBraking[Flag	BrakePedalPosition[%]	ActVehAccel[m/s2]	ActVehAccelValid[Flag]	GMProtLongAcc[m/s2]	GMProtLongValid[Flag]	P3MotorStatus[Enum]	P3TorqueCommand[Nm]	P3TorqueFeedback[Nm]	P3MotorSpeed[RPM]	RegenTrqDirChkEnbl[Flag]	RegenTrqDirChk[Flag]	RegenTrqDirChkInvd[Flag]	RegenTrqDirChkEnbl[Flag]	RegenTrqDirChk[Flag]	RegenTrqDirChkInvd[Flag]	
1	1.1.2.1	N/A	N/A	N/A	N/A	N/A	N/A	Device.Offline (0)	N/A	N/A	N/A	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	Y
2	1.1.2.2	N/A	N/A	N/A	N/A	N/A	N/A	Device.Ready (2)	N/A	N/A	N/A	TRUE	N/A	N/A	TRUE	N/A	N/A	Y
3	1.1.2.2.1	FALSE	15	N/A	N/A	N/A	N/A	Device.Ready (2)	-1	5	N/A	TRUE	TRUE	FALSE	TRUE	TRUE	FALSE	Y
4		FALSE	15	N/A	N/A	N/A	N/A	Device.Ready (2)	5	-1	N/A	TRUE	TRUE	FALSE	TRUE	TRUE	FALSE	Y
5		TRUE	5	N/A	N/A	N/A	N/A	Device.Ready (2)	-1	5	N/A	TRUE	TRUE	FALSE	TRUE	TRUE	FALSE	Y
6		TRUE	5	N/A	N/A	N/A	N/A	Device.Ready (2)	5	-1	N/A	TRUE	TRUE	FALSE	TRUE	TRUE	FALSE	Y
7	1.1.2.2.2.1	TRUE	15	N/A	N/A	N/A	N/A	Device.Ready (2)	5	-1	50	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	N
8	1.1.2.2.2.2	TRUE	15	N/A	TRUE	N/A	FALSE	Device.Ready (2)	5	-1	70	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	Y
9	1.1.2.2.2.2	TRUE	15	N/A	FALSE	N/A	TRUE	Device.Ready (2)	5	-1	70	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	Y
10	1.1.2.2.2.2	TRUE	15	N/A	FALSE	N/A	FALSE	Device.Ready (2)	5	-1	70	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	Y
11	1.1.2.2.2.3	TRUE	15	0	TRUE	0	TRUE	Device.Ready (2)	5	-1	70	TRUE	TRUE	FALSE	TRUE	TRUE	FALSE	Y
12	1.1.2.2.2.3	TRUE	15	0	TRUE	1	TRUE	Device.Ready (2)	5	-1	70	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	Y
13	1.1.2.2.2.3	TRUE	15	1	TRUE	0	TRUE	Device.Ready (2)	5	-1	70	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	Y
14	1.1.2.2.2.4	TRUE	15	1	TRUE	1	TRUE	Device.Ready (2)	5	-1	70	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	Y

Figure 26: Sample documentation of SIL test results

As seen in the SIL tests, algorithmic bugs have been found that either mean the software implementation is incorrect, or the design of the software contains a fault and does not meet the system level requirements.

The same test cases created in SIL has been re-run in the HIL environment as seen in below in Figure 27 for the software unit. Development for hardware specific tests are under way, introducing extra tests such as the responses to loss of communication or out of range data. These results might push for considerations to account for extra enabling conditions for the diagnostic and changes in mitigation strategies. Once the hardware considerations are accounted for, full controller tests are on the schedule to verify the code before deployment to the vehicle.



ocation: C:\Users\Burnir
uthor: Burninator
ate: 3/18/2017 3:31:57 PI

Variable Array_1: Abs/Out1
-1.79769313486232E+308..1.79769313486232E+308 Converted Incr. +-1 / 10

Variable	Value	Unit
APP1[V]/Value	---	
APP1[V]1/Value	---	
Data Type Conversion/Out1	---	
Data Type Conversion1/Out1	---	
Abs/Out1	---	
Gain1/Gain	---	
Gain2/Gain	---	
Gain/Gain	---	
Gain1/Gain	---	

APP Mismatch/APPPositionMismatchFlt[Enum]
– – –
APP Out-of-Range/APP1OutOfRange[Flag]
– – –
APP Out-of-Range/APP2OutOfRange[Flag]
– – –

Plotter_7: APP Out-of-Range/APP2OutOfRange[Flag]
0 .. 1 Converted

#	Name	Value	Unit	+	Raster
●	RelationalnOperator1/Out1	---			HostS.
●	APP Out-of-Range/APP1O...	---			HostS.
●	APP Out-of-Range/APP2O...	---			HostS.

Figure 27: Screenshot of sample HIL environment for test cases

System safety requirements:

- Regenerative braking shall not apply negative torque when the vehicle speed is below 7mph.

Table 17: ASIL evaluation for system safety requirement

Fault Cases	Severity	Exposure	Controllability	ASIL
Regen Torque Direction	S3	E2	C2	B

Requirements enabling Regen Braking:

- Brake Pedal > 7%
- Accelerator Pedal < 7%
- Vehicle Speed > 7MPH
- Shift Lever in Drive
- Regen disable switch is OFF

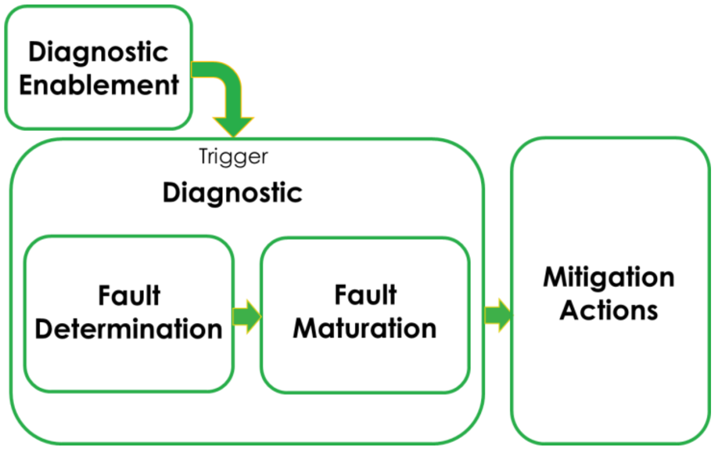
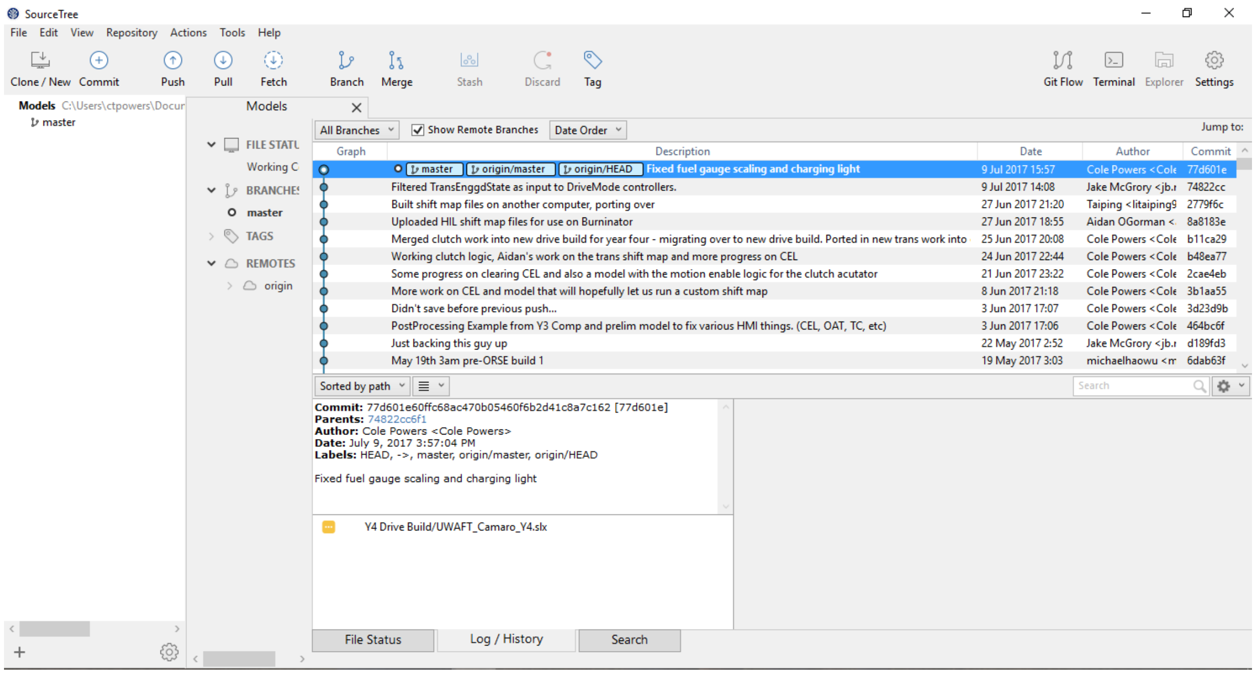


Figure 28: Control strategy for fault detection and failure mitigation

Appendix F -


SourceTree
File Edit View Repository Actions Tools Help
Clone / New Commit Push Pull Fetch Branch Merge Stash Discard Tag Git Flow Terminal Explorer Settings
Models C:\Users\ctpowers\Docur
master
Models
FILE STATU
Working C
BRANCHES
master
TAGS
REMOTES
origin
All Branches
Show Remote Branches
Date Order
Jump to:
Sorted by path
Search
File Status Log / History Search

Graph	Description	Date	Author	Commit
	master origin/master origin/HEAD Fixed fuel gauge scaling and charging light	9 Jul 2017 15:57	Cole Powers <Cole	77d601e
	Filtered TransEnggdState as input to DriveMode controllers.	9 Jul 2017 14:08	Jake McGrory <jb.r	74822cc
	Built shift map files on another computer, porting over	27 Jun 2017 21:20	Taiping <litaiping9	2779f6c
	Uploaded HIL shift map files for use on Burninator	27 Jun 2017 18:55	Aidan OGorman <	8a8183e
	Merged clutch work into new drive build for year four - migrating over to new drive build. Ported in new trans work into	25 Jun 2017 20:08	Cole Powers <Cole	b11ca29
	Working clutch logic, Aidan's work on the trans shift map and more progress on CEL	24 Jun 2017 22:44	Cole Powers <Cole	b48ea77
	Some progress on clearing CEL and also a model with the motion enable logic for the clutch acutator	21 Jun 2017 23:22	Cole Powers <Cole	2cae4eb
	More work on CEL and model that will hopefully let us run a custom shift map	8 Jun 2017 21:18	Cole Powers <Cole	3b1aa55
	Didn't save before previous push...	3 Jun 2017 17:07	Cole Powers <Cole	3d23d9b
	PostProcessing Example from Y3 Comp and prelim model to fix various HMI things. (CEL, OAT, TC, etc)	3 Jun 2017 17:06	Cole Powers <Cole	464bc6f
	Just backing this guy up	22 May 2017 2:52	Jake McGrory <jb.r	d189fd3
	May 19th 3am pre-ORSE build 1	19 May 2017 3:03	michaelhaowu <m	6dab63f

Commit: 77d601e60ffc68ac470b05460f6b2d41c8a7c162 [77d601e]
Parents: 74822cc6f1
Author: Cole Powers <Cole Powers>
Date: July 9, 2017 3:57:04 PM
Labels: HEAD, ->, master, origin/master, origin/HEAD

Fixed fuel gauge scaling and charging light

Y4 Drive Build/UWAFT_Camaro_Y4.slx