CODEN:LUTEDX/(TEIE-5432)/1-79(2020)

Industrial Electrical Engineering and Automation

Electronic Control Unit for Automatic Control of Torque Vectoring

John Huzell

Johan Göransson

Division of Industrial Electrical Engineering and Automation
Faculty of Engineering, Lund University

Electronic Control Unit for Automatic Control of Torque Vectoring

John Huzell and Johan Göransson

DIVISION OF INDUSTRIAL ELECTRICAL ENGINEERING AND AUTOMATION
FACULTY OF ENGINEERING LTH | LUND UNIVERSITY
2020

MASTER THESIS

BorgWarner

This thesis was conducted at BorgWarner Sweden AB for the Division of Industrial Electrical Engineering and Automation, Faculty of Engineering, Lund University. The division of labor was evenly distributed between the two authors.

Supervised by Daniel Blom and Pontus Månsson from BorgWarner Sweden AB.
Supervised by Assoc. Prof. Gunnar Lindstedt from the Faculty of Engineering.
Examined by Researcher Johan Björnstedt from the Faculty of Engineering.

For the Master of Science in Engineering Degree

Division of Industrial Electrical Engineering and Automation
Box 118, 221 00 Lund
Sweden
www.iea.lth.se

Abstract

Torque vectoring for automobiles is used to increase cornering speeds and improve road safety by manipulating the car's wheel-speeds. One method of torque vectoring for a rear-wheel-drive Formula Student racecar is using a dual-clutch to slip the inner wheel thus create a yaw moment during cornering.

The goal of this thesis was to develop and design an electronic control unit to automatically control the two clutches to achieve the target yaw moment. By building on the previous development of a dual-clutch and a theoretical control model, this thesis was delimited to the implementation of a control algorithm on the hardware used to control the dual-clutch.

Ulrich and Eppinger's product development method was used to develop an initial alpha-prototype for proof-of-concept development. A beta-prototype was then developed to control the dual-clutch via a Controller Area Network (CAN), before being packaged into a final product, the Torque Vectoring Electronic Control Unit (TVECU).

The final product resulted in an electronic control unit capable of being mounted to a mechanical dual-clutch and regulating two Gen6 clutch actuators, thereby controlling the yaw moment. It can then be used to test, validate and further develop a dual-clutch torque vectoring system, and lays the foundations for a next-generation commercial model.

Keywords: torque vectoring, electronic control unit, automatic control, yaw moment, CAN, PCB, Formula Student

Sammanfattning

Momentvektorering uppnås genom att manipulera bilens hjulhastigheter, i syfte att öka kurvhastigheterna och förbättra trafiksäkerheten. En dubbelkoppling på bakaxeln är en möjlig vektoreringsmetod för en bakhjulsdriven Formula Student-bil. Dubbelkopplingen gör att det inre hjulet kan glida och därmed skapa ett vridmoment kring bilens kropp i kurvans färdriktning.

Målet med examensarbetet var att undersöka och utveckla en elektronisk styrenhet för att automatiskt styra de två kopplingarna i syfte att uppnå önskat vridmoment. Genom att bygga vidare på tidigare utveckling av en dubbelkoppling och en teoretisk styrmodell avgränsades examensarbetet till implementeringen av en kontrollalgoritm på en elektronisk styrenhet.

Ulrich och Eppingers produktutvecklingsmetod användes för att utveckla en initial alfa-prototyp för konceptutveckling. En beta-prototyp utvecklades sedan för att styra dubbelkopplingen via ”Controller Area Network” (CAN), innan den förpackades i en slutprodukt, kallad ”Torque Vectoring Electronic Control Unit” (TVECU).

Slutprodukten resulterade i en elektronisk styrenhet designad för att monteras på en mekanisk dubbelkoppling och styra två Gen6-aktuatorer. Den kan sedan användas för att testa, verifiera och vidareutveckla ett momentvektoreringssystem och lägger grunden för nästa generations kommersiella modell.

Nyckelord: vridmomentvektorering, elektronisk styrenhet, automatisk styrning, kurvtagning, CAN, kretskort, Formula Student

Acknowledgements

We have received input and support from various people at BorgWarner and LTH throughout the thesis. We are especially grateful to Daniel Blom, Team Leader for Vehicle Dynamics and Simulations at BorgWarner, for his continuous support. Blom has been very helpful with his technical expertise and administration and has been invaluable in finding the right employees for several administrative issues along the way.

We would like to thank Mia Grahovic for all the help and input regarding the control model and Pierre Peterson for his help regarding the electronic control unit selection. Without their input, the master thesis would not have progressed as smoothly as it did.

We are also grateful to our supervisor at BorgWarner, Pontus Månsson and our supervisor at LTH, Gunnar Lindstedt for their help regarding administration and planning.

Finally, we are appreciative of David Bäcklund and Jakob Wilson who have made our time at BorgWarner fun and competitive.

Contents

List of Figures

List of Tables

List of Acronyms

ABS Anti-lock Braking System

TVDC Torque Vectoring Dual Clutch

TVECU Torque Vectoring Electronic Control Unit

ECU Electronic Control Unit

FS Formula Student

CAN Controller Area Network

PCB Printed Circuit Board

RPM Revolutions Per Minute

FS CAN CAN to/from car

TVDC CAN CAN for TVDC

COG Center of Gravity

List of Nomenclature

Automatic Control

Ψ	Yaw Rate
Ψ_{des}	Desired Yaw Rate
ζ	Anti Windup
D_t	Derivative Term
e	Yaw Rate Error
e_{old}	Previous Yaw Rate Error
I_t	Integral Term
K_d	Derivative Gain in PID Controller
K_I	Integral Gain in PID Controller
K_p	Proportional Gain in PID Controller
P_t	Proportional Term
u	Control Signal

Vehicle Dynamics

δ_{st}	Steering Wheel Angle
δ_w	Wheel Angle
ϕ	Roll
ψ	Yaw
τ_l	Torque Provided, Left Wheel
τ_r	Torque Provided, Right Wheel
$\tau_{slip,l}$	Slip Torque, Left Wheel
$\tau_{slip,r}$	Slip Torque, Right Wheel
τ_{tot}	Total Torque
θ	Pitch
A	Driver Preference

a	Vehicle Body Acceleration
a_x	Vehicle Body Acceleration x-Axis
a_y	Vehicle Body Acceleration y-Axis
$C_{\alpha,f}$	Lateral Tire Stiffness for the Front Wheels
$C_{\alpha,r}$	Lateral Tire Stiffness for the Rear Wheels
f_t	Tire Friction
g	Gravity Acceleration
I	Vehicle Moment of Inertia
l_f	Distance from COG to Front Axle
l_r	Distance from COG to Rear Axle
L_{tw}	Track Width
L_{wb}	Wheel Base
m	Vehicle Mass
m_{distr}	Mass Distributed
r_t	Tire Radius
v	Average Vehicle Velocity
v_{fl}	Front Left Wheel Speed
v_{fr}	Front Right Wheel Speed

Chapter 1

Introduction

This chapter deals with the background, objectives, and limitations of the development of an electronic control unit for torque vectoring. It proceeds to present the method of development, risks and the organization of the report.

Ever since the dawn of man, we have yearned to compete and go fast. From sailing and horseback riding during the antiquities to jets and cars in the modern era. The endeavor to be the quickest and the quest to break boundaries has always driven mankind in the pursuit of technology. Today, car racing has progressed to a point where the cars are pushing man's limits, and rules are in effect for both the safety of drivers and the fairness of competitions. Their efforts to break records have technologies spilling into the civilian market.

One such game-changing technology is the invention of torque vectoring, used to increase the performance of vehicles such as cars and fighter jets during cornering. In their own attempt to break records, BorgWarner is developing a torque vectoring system for Formula Student racing cars in cooperation with the Lund Formula Student racing team.

A previous master thesis has completed the mechanical side of the torque vectoring system, and another has investigated a method of automatic control. However, work remains to be done on the electronics and the implementation of the control algorithms.

1.1 Background

1.1.1 Lund Formula Student

Formula Student is one of the largest and most renowned engineering design competitions for university students worldwide. Students spend nine months pushing their skills in the design and manufacture of top-class single-seat open-wheel formula racing cars. The teams compete in both static and dynamic events, where they score points in different categories such as design, cost, and business as well as acceleration, cornering, and endurance. Over 600 universities compete in combustion, electric and driverless classes increasing their engineering skills and making them better prepared for the future [1].

Lund Formula Student, previously LURacing, is Lund University's contribution to the Formula Student (FS) competitions. Founded in 2006, the team is now (2019-2020) producing their 12th and 13th cars simultaneously. The former being the next iteration of a series of successful combustion engine cars, and the latter being the team's first iteration of a fully electric car.

Figure 1.1: The most recent FS-car, the LFS19 [1].

1.1.2 Thesis One: Torque Vectoring Dual Clutch (TVDC)

Sven Kalkan investigated the possibility of torque vectoring for the FS-cars by replacing the limited-slip differential (LSD) with an electro-hydraulically actuated dual-clutch [2]. His thesis culminated in a "Torque Vectoring Dual Clutch" (TVDC) comprised of two independently controlled lamellar clutches (Figures 1.2-1.5 below).

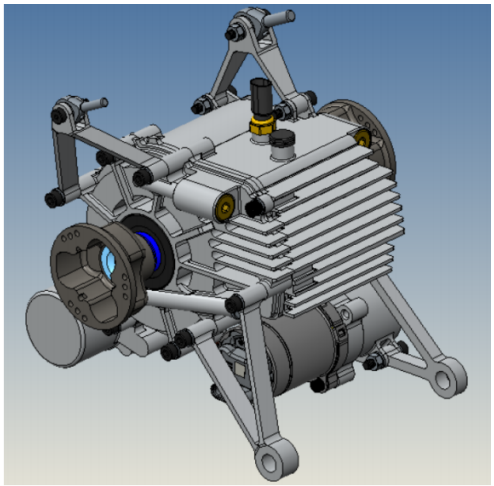


Figure 1.2: Front view of the TVDC [2].

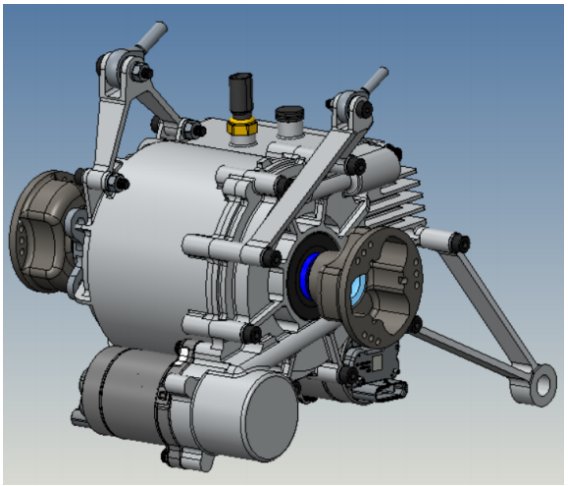


Figure 1.3: Rear view of the TVDC [2].

In order to provide some level of function upon failure, the TVDC is designed to work normally-closed, i.e. the clutch is normally a fixed axle and torque vectoring is achieved through slip from actuated clutches. Normally-closed is a relatively unusual design choice and most clutches operate as normally-open. However, since Formula Student competitions press the cars to their limits, they are prone to breakdowns. As such, normally-closed is a safer albeit more complex alternative to normally-open clutches.

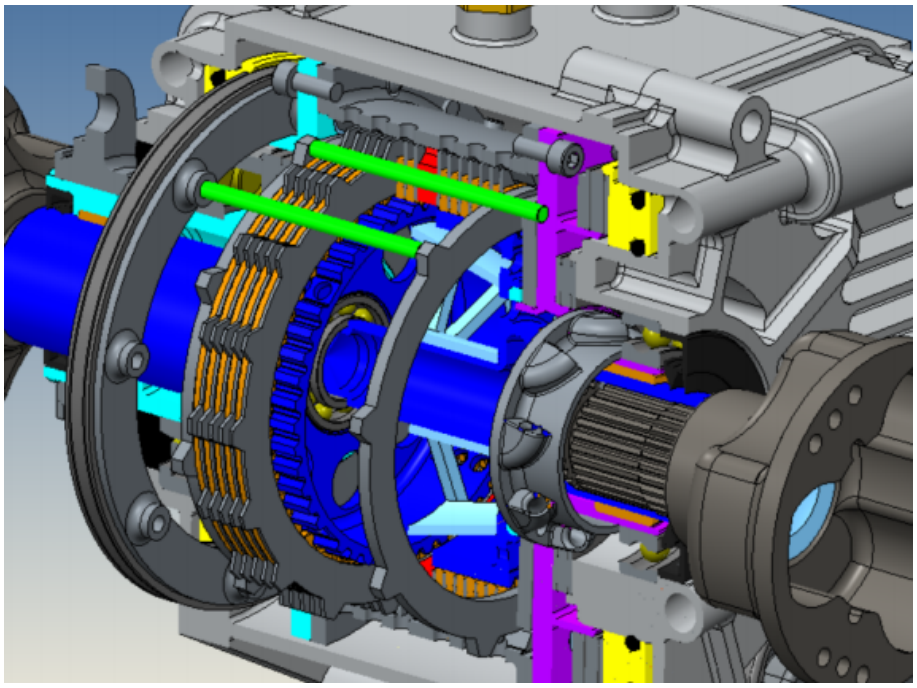


Figure 1.4: Cross-section of the TVDC [2].

Two Generation 6 clutch actuators from BorgWarner, controlled by control signals via a Controller Area Network (CAN), provide the torque vectoring moment yaw on the car. Neither the controller, i.e. the electronic control unit (ECU), nor the control algorithm were implemented in Kalkan's thesis as it was delimited to the mechanical function of the TVDC. For more information on differentials and torque vectoring, see *2.1.1 Differentials and Torque Vectoring*.

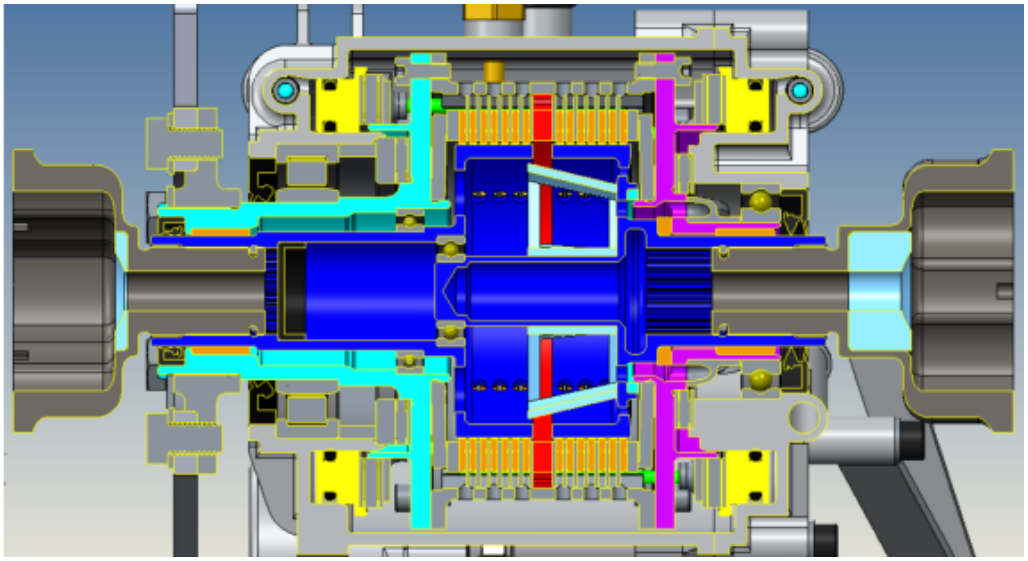


Figure 1.5: Cross-section of the TVDC [2].

1.1.3 Thesis Two: Torque Vectoring Control Algorithm

Continuing the development of Kalkan's work, Mia Grahovic and Madeleine Rosicki investigated and developed the theoretical control algorithm required to achieve torque vectoring with the TVDC [3]. The algorithm, developed in Simulink, uses the FS-car's yaw rate as a control signal and a vehicle model supplied by BorgWarner to simulate the physical car. During assessment it was determined that the nonlinear model using a simulated TVDC performed better than a limited-slip differential (LSD), which is currently used by Lund Formula Student. The model is summarized in the signal overview in Figure 1.6 below.

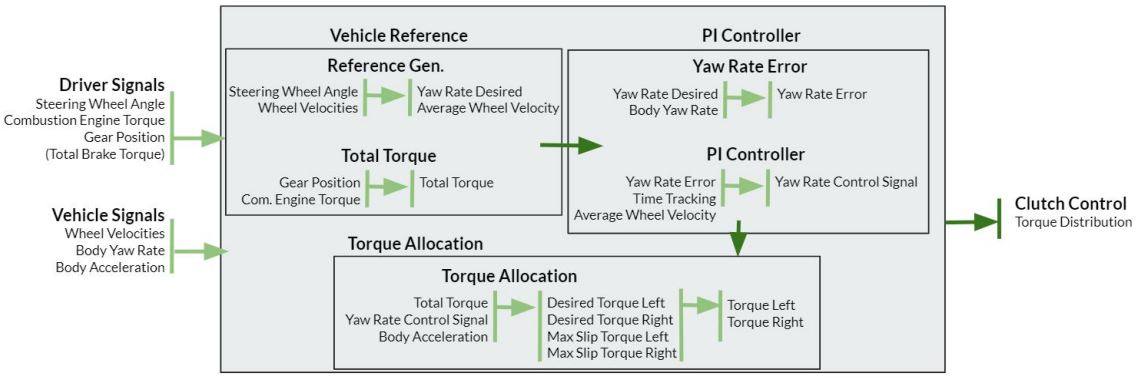


Figure 1.6: Overview of signals in theoretical control model.

1.2 Mission Statement

The objective of this thesis was to design and develop an electronic control unit, running a control algorithm, to control the first generation Torque Vectoring Dual Clutch (TVDC) on a Formula Student car.

1.2.1 Tiered Ambition Level

The objective was divided into three-tier goals; acceptable, desired and optimal. An acceptable tier goal was a completed electronic control unit (ECU) prototype, both hardware and software, capable of receiving vehicle data and generating a control signal for the TVDC. The desired tier goal was a fully developed ECU implemented onto the TVDC and tested on a test bench/test rig. The optimal tier goal was the design and development of hardware, with a complete implementation as well as practical tests in a Formula Student car.

1.3 Assumptions, Limitations and Delimitations

At the start of the thesis, it was assumed that the TVDC is manufactured and available in a reasonable time-frame within the scope of the thesis. It was also assumed that the theoretical automatic control functions without major fault. Nevertheless, the untimely manufacture of the TVDC may become a limitation, and inhibiting the desired and optimal tier goals.

The thesis was delimited to the implementation and development of an ECU. While the algorithm was rewritten and modified to be usable in practice, no consideration was taken whether the control theory and method is the most suitable or efficient. The thesis was also delimited to not include neither complete failure and error handling nor an exhaustive debug logging. In other words, if a sensor fails or the TVDC breaks the system will not attempt to compensate or automatically execute failure handling. The lack of exhaustive debug logging will also make error analysis harder.

Furthermore, the packaging of the ECU was primarily designed to withstand testing and development only and works as the first prototype for a weather-proof system. Neither Ingress Protection certification nor a final packaging solution was in the scope of the thesis, although an equivalent Ingress Protection test was conducted.

Finally, as the ECU outputs a control signal it may function with other torque-vectoring methods. However, the primary design was to be used with the TVDC and therefore it is not guaranteed to function with other systems.

1.4 Method

The development of the ECU, henceforth known as the "Torque Vectoring Electronic Control Unit" (TVECU), followed the product development methodology laid out by Karl T. Ulrich and Steven D. Eppinger. The Ulrich and Eppinger method is a wide-reaching product development method which at times reaches further and more in-depth than necessary. Therefore, the method was adapted to this project's needs and requirements with variations in the method as a result.

The method was comprised of 5 of Ulrich and Eppinger's 6 phases [4]:

Phase 0 Planning - The planning phase in which the project was evaluated and prioritized, time and resources were allocated as well as a concretization of the project in terms of mission statements, assumptions and constraints as well as vision.

Phase 1 Concept Development - The concept development phase where customer needs were identified, target specifications were established and concepts were generated. Literature studies and analyses of related works were conducted, and required hardware was ordered after concept selection.

Phase 2 System-Level Design - The preliminary design phase where various designs, based on the selected concept, were laid out and evaluated. Alpha-prototyping started in this phase and continued on through phase 3.

Phase 3 Detail Design - The main development phase, conducted in parallel with phase 4. The alpha-prototype was tested and the development of the beta-prototype, and final product, started.

Phase 4 Testing and Refinement - Evaluation, improvements and further developments to the beta-prototype.

Phase 5 Production Ramp-Up - This phase was stricken from the project and replaced with a conversion from beta-prototype to the final product. It also included a final evaluation of the final product, the control method and the thesis as a whole.

1.4.1 Risk Analysis

Anything that could affect the development or function of the TVECU in a non-insignificant way was evaluated using risk matrices. The risk matrices were divided into two categories; "Risk Matrix - Technical" and "Risk Matrix - Administration". The former evaluated factors related to the feasibility of the thesis out of a technical and "project administrative" perspective. The latter evaluated factors related to the feasibility of the thesis out of a university-administrative and examinatorial perspective, for example, licenses and ownership. It is considered both sensitive information and outside the scope of this thesis and is therefore not included.

The risk matrices were sorted by probability versus consequence. If the risk was high, ie. both the probability and the consequences were high - the risk was eliminated. If the risk was low, ie. both the probability and the consequences were low, the risk was observed for potential changes. Finally, if the risk was medium changes were made to minimize the risk and observed for potential changes. Although risk analysis was conducted continuously throughout the thesis, the bulk of "Risk Matrix - Administration" was evaluated during the Planning phase while the majority of "Risk Matrix - Technical" was determined during the Concept phase, shown in *3.1 Risk Analysis and Strategy*.

1.4.2 Organization of Report

Chapter I: Introduction This chapter deals with the background, objectives, and limitations of the development of an electronic control unit for torque vectoring. It proceeds to present the method of development, risks and the organization of the report.

Chapter II: Theory This chapter explains the theory needed to understand the report, divided into four sections; Vehicle Dynamics, Automatic Control, Electronic Control Unit, and the Controller Area Network. Vehicle Dynamics aims gives an understanding of dynamics, yaw rate and the basic physics of a moving car. Automatic Control examines the continuous PID control and conversion to discrete control. Electronic Control Unit briefly describes what ECUs and PCBs are while Controller Area Network explains what it is and how it functions.

Chapter III: Concept Development This chapter analyzes possible hazards in the development of the product through risk and Automotive Safety and Integrity Level (ASIL) analysis. It proceeds to determine the needs and requirements of the TVECU in order to define the target specifications. Finally, hardware and software concepts are screened and selected according to the target specifications.

Chapter IV: System-Level Design This chapter explains the high-level hardware design of the TVECU and its additional components. It describes supporting development systems used for CAN communication, and tests the alpha-prototype against a commercial engine ECU. Finally, it evaluates software deployment methods and describes the general process of the TVECU software.

Chapter V: Detail Design This chapter develops the hardware configuration and wiring of the TVECU beta-prototype. It proceeds to address packaging and the Gen6 connectors, and examines the TVECU software and control algorithm in detail. Ultimately, it presents the final assembled product.

Chapter VI: Testing and Refinement This chapter describes the various tests done in parallel with the development of the prototypes and final product. It outlines the verification of the practical control algorithm and then depicts the TVECU system tests conducted to verify communication, control and sensors.

Chapter VII: Evaluation and Conclusion This chapter evaluates the TVECU against the target specifications and identifies design elements not yet implemented. It reviews potential issues moving forward and presents items for future development such as CAD and PCB design, before concluding the thesis.

Chapter 2

Theory

This chapter explains the theory needed to understand the report, divided into four sections; Vehicle Dynamics, Automatic Control, Electronic Control Unit, and the Controller Area Network. Vehicle Dynamics aims gives an understanding of dynamics, yaw rate and the basic physics of a moving car. Automatic Control examines the continuous PID control and conversion to discrete control. Electronic Control Unit briefly describes what ECUs and PCBs are while Controller Area Network explains what it is and how it functions.

2.1 Vehicle Dynamics

2.1.1 Differentials and Torque Vectoring

When a car corners, the outer wheel will travel further than the inner wheel as seen in Figure 2.1. Consequently, the outer wheel has to travel faster than the inner wheel in order to turn. If the wheels are being driven by the drive shaft on a fixed axle and thus not able to rotate at different speeds then the wheel with the least ground traction will lose grip and skid along the ground. Therefore a mechanism is required to allow the wheels to rotate at different speeds while still being connected to the same driveshaft.

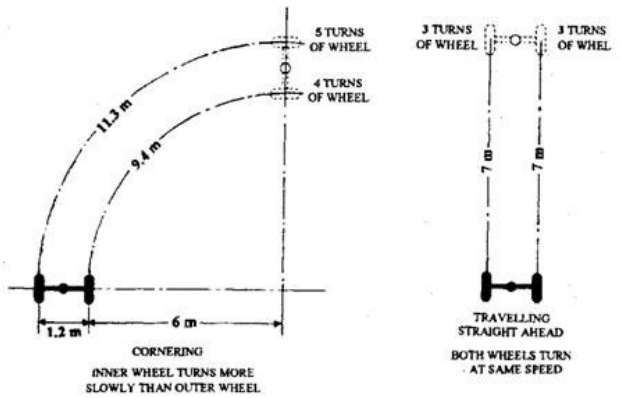


Figure 2.1: Illustration of different wheel speeds [5].

This is conventionally achieved by using a differential. Several different types of differentials exist, but they all serve the same primary function. In the open differential in Figure 2.2, torque is delivered from the engine to the crown wheel via the driveshaft and pinion. The rotating cage, attached to the crown wheel, holds and rotates small gears that in turn rotate both wheel axles driving the car forward in a straight line. In this sense, it functions similarly to a fixed axle. However, spinning the right wheel forward when the driveshaft is locked as in Figure 2.3 will cause the small rears to rotate the left wheel backward in the opposite direction.

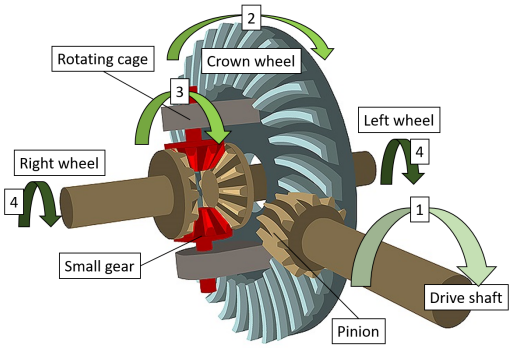


Figure 2.2: Differential when driving straight [6].

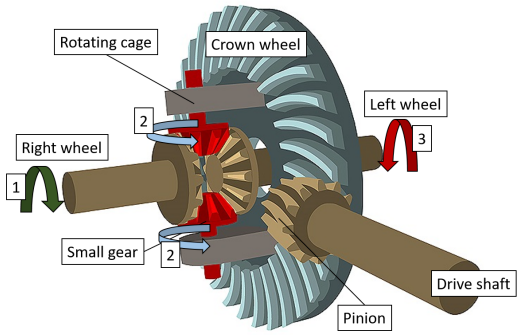


Figure 2.3: Differential with locked drive shaft [6].

During normal use the differential will function like Figure 2.2 when driving straight. During cornering it will combine the two use cases into Figure 2.4, where the outer wheel is allowed to rotate faster than the inner wheel. Several types of differentials exist such as the open differential, locked differential, spool differential, the limited-slip differential (LSD) currently used on the Lund Formula Student's car, and many others. While their purpose is the same, they utilize different methods with different benefits and drawbacks.

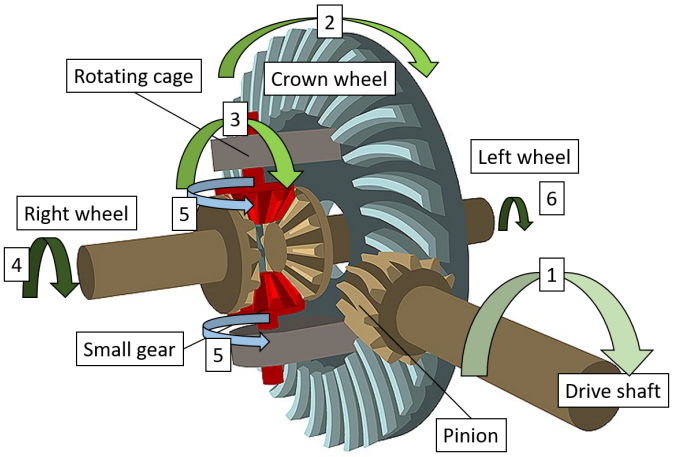


Figure 2.4: Differential during cornering [6].

Torque Vectoring

With a regular differential, the difference in wheel speeds occurs due to the different ground speeds of the wheels as a *result* of the car turning. Torque vectoring can be summarized as the opposite; changing the torque on each wheel, and consequently the wheel speeds, in order to create a turning moment and turn the car, as shown in Figure 2.5. Consequently, the car can corner at a steeper angle and at higher speeds than a car without torque vectoring.

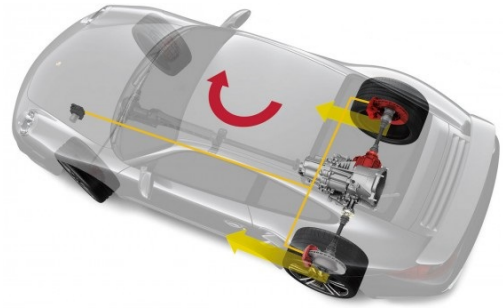


Figure 2.5: Different wheel torque creates a turning moment on the car [7].

Torque vectoring can be achieved by several means. Each motor in an electric powertrain, for example, can be controlled individually and generate a turning moment. Many modern production cars create torque vectoring by braking the inner wheels during cornering. Other possible torque vectoring devices are active differentials, torque-vectoring differentials, and dual clutches, among others.

2.1.2 Vehicle Axis System

The vehicle axis system is a three-dimensional system with its origin at the center of gravity (COG) of the vehicle. The longitudinal axis is described as the x-axis and is the axis in which the vehicle travels along when the wheel angle is at zero. The lateral axis is set as the y-axis and the vertical axis is described as z. All linear motions or accelerations from the COG is along one or more of the three axes.

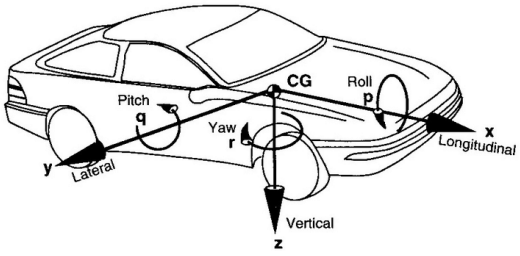


Figure 2.6: Vehicle axis system, where gravity is defined as positive [8].

A rotational moment creates a rotational acceleration around the COG. If the moment is centered around the x-axis it is known as a roll and if it is centered around the y-axis it is instead known as a pitch. If the rotation is around the z-axis it is known as yaw [9]. It is the centripetal acceleration from the yaw moment that the driver experiences when cornering.

2.1.3 Yaw Stability Control Systems

There are three different yaw control systems, also known as electronic stability control systems. The purpose of an electronic stability control system is, as the name implies, to stabilize the yaw rate and adjust it to a nominal value that is as close to what the driver expects as possible [10]. The three different control systems are:

- Steer-by-Wire
- Differential Braking
- Active Torque Distribution

Steer-by-Wire systems remove the steering wheel's physical connection to the wheels of the vehicle and instead control the wheels of the vehicle electronically [11]. The systems stabilize by modifying the steering wheel angle to compensate for unwanted disturbances [10]. Differential braking systems use the braking system to apply yaw stability. The brakes are normally distributed equally on the left and right wheels [10].

The differential brake system stabilizes by distributing the brake torque to each wheel independently, keeping the car on the target path as in Figure 2.7 [10, 12]. Active torque distribution systems control the torque applied on the wheels using, for example, an active differential to maintain the desired yaw rate [10].

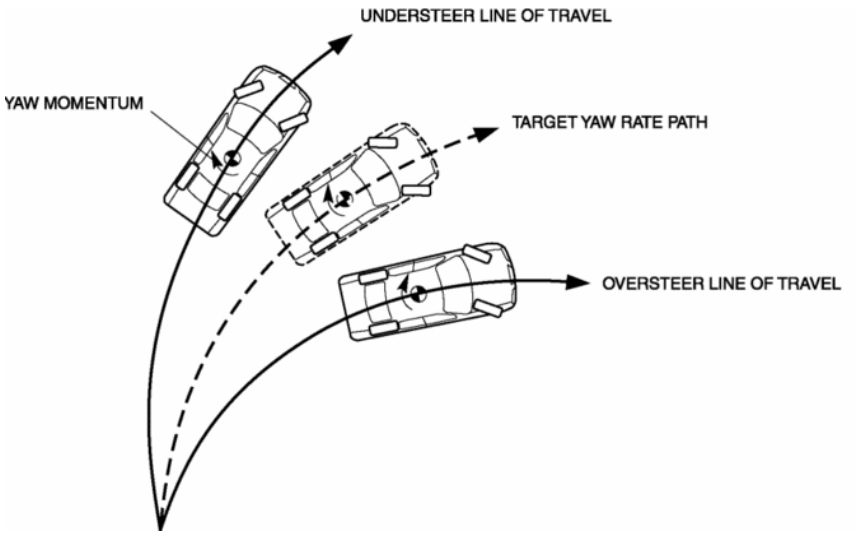


Figure 2.7: Visualization of understeering, oversteering and target yaw [13].

Torque Vectoring, Active Torque Distribution

Torque vectoring is similar to the stabilization systems, but its purpose is to increase the yaw moment instead of stabilizing it. For the active torque distribution system to work there must be an electronic control unit controlling the specific torque distributed to the wheels at all times. The torque distributed depends on the control signal from the ECU, which is in turn affected by a changing desired yaw rate. The desired yaw rate can, according to [10], be described as:

$$\psi_{des} = \frac{V_x}{R} \tag{2.1}$$

Where V_x is the speed of the vehicle along the x-axis and R is the cornering radius [3]. It follows that the cornering radius can be described by:

$$R = \frac{L + K_v \cdot V^2}{\delta_{st}} \tag{2.2}$$

Where V is the global velocity, K_v is the under-steer gradient, δ_{ss} is the steering wheel angle and L is the wheel base [3]. Rewritten, the yaw rate can be described as:

$$\psi_{des} = \frac{V_x \cdot \delta_{st}}{L + \frac{m \cdot V_x (l_r \cdot C_{\alpha,r} - l_f \cdot C_{\alpha,f})}{2 \cdot C_{\alpha,r} \cdot C_{\alpha,f} \cdot L}} \tag{2.3}$$

Where ψ_{des} is the desired yaw rate, v_x is the velocity along the x-axis, δ_{st} is the steering wheel angle and m is the vehicle mass. l_f and l_r are the front and rear axle distances to the COG, while $C_{\alpha,f}$ and $C_{\alpha,r}$ are the front and rear lateral tire stiffness [3, 10]. The desired yaw rate can be compared to the actual yaw rate at a specific point in time and the error can be calculated. This error is what the electronic control unit is controlling and trying to set to zero.

2.2 Automatic Control

The following theory concerning automatic control is all derived from Real-Time Control Systems *by Karl-Erik Årzén. It is provided to give an understanding of the control algorithm in the final design. Årzén provides theory concerning both the control theory and the link between the theory and the implementation [14].*

2.2.1 Continuous Control in PID Controller

The PID-type controllers are the most widely used controllers in industries. The controller consists of three parts; the proportional (P) term, the integral (I) term and the derivative (D) term. The proportional term attempts to remove the error. The integral term is based on the error history and attempts to close the stationary error that the proportional term cannot account for. The derivative term predict the coming error to prevent oscillations, overshoot, and instability.

The proportional controller can be described in the continuous state-space form with:

$$P(s) = K_c \cdot E(s) \tag{2.4}$$

Transforming the proportional term to the time-domain yields:

$$P(t) = K_c(y_{sp} - y) + u_b = K_c \cdot e + u_b \tag{2.5}$$

Where K_c is the proportional gain, y_{sp} is the reference value, y is the current value and u_b is the reset term. The purpose of the reset term is to maintain a control signal that is different from zero even when the control signal is zero. Solely implementing a proportional term can be problematic as it gives rise to a steady-state error. To avoid a steady-state error an integral term can be implemented. The state-space form of the integral term is described by:

$$I(s) = \frac{K_c}{s \cdot T_i} \cdot E(s) \tag{2.6}$$

Where T_i is the integral constant, s is the Laplace variable and $E(s)$ is the current error, described in 2.5 as $y_{sp} - y$. The time-domain controller is described by:

$$U(t) = P(t) + I(t) = K_c\big(e(t) + \frac{1}{T_i}\int_0^t e(s)\, ds\big) \tag{2.7}$$

Where T_i is the integral constant. The controller in Equation 2.7 is called a PI controller. A derivative term is used to anticipate the coming behavior and avoid overshoots. The derivative term predicts the future by extrapolating the error curve along its tangent, yielding the state-space design shown in Equation 2.8:

$$D(s) = K_c \cdot T_d \cdot s \cdot E(s) \tag{2.8}$$

Where T_d is the derivative constant. This yields a controller in the time-domain with the following characteristics:

$$U(t) = P(t) + I(t) + D(t) = K_c\Big(e(t) + \frac{1}{T_i} \cdot \int_0^t e(s)\, ds + T_d\, \frac{de}{dt}\Big) \tag{2.9}$$

It is important to note that the implementation of a pure derivative is not wanted in a controller. Typically the derivative amplifies the measurement noise at specific frequencies, which can be avoided by replacing the transfer function $s \cdot T_d$ with:

$$\frac{s \cdot T_d}{1 + \frac{s \cdot T_d}{N}} \tag{2.10}$$

Where N is generally around 10-20. This will yield the derivative term:

$$D(s) = \frac{s \cdot T_d}{1 + \frac{s \cdot T_d}{N}} \cdot E(s) \tag{2.11}$$

2.2.2 Discrete Implementation of PID Controller

Using a controller designed for continuous-time is not applicable when implementing a controller in an embedded system. The continuous-time domain expects the controller's sample-time to be continuous, i.e. infinitely small.

This is not the case in embedded systems, as there will always be a discrete sample-time. This phenomenon requires a controller design in the derivative time-domain. Discretizing a continuous control system has several challenges, the most significant being sampling and quantization.

Discretizing a continuous system demands approximating the Laplace operator, s, as a discrete shift operator, z, which is a specific time-step. There are three main methods used for this transformation; forward-difference, backward-difference, and Tustin's approximation.

A proportional controller simply observes and corrects the error at a specific time, as shown in Equation 2.12.

$$P(t_k) = K\Big(b \cdot y_{sp}(t_k) - y(t_k)\Big) \tag{2.12}$$

Where t_k is a specific time-step, y is the actual value at a specific time, y_{sp} is the reference value and b is a variable gain.

Forward-Difference

The approximation of s using the forward-difference method is:

$$s = \frac{z-1}{h} \tag{2.13}$$

Where h is the sample-time and z the shift operator. As the name suggests, this approximation is the equivalent of replacing all the derivatives with a forward-difference.

$$\frac{dx(t)}{dt} = \frac{x(t+h) - x(t)}{h} \tag{2.14}$$

Approximating the integral term in Equation 2.6 with the forward-difference method yields:

$$\frac{I(t_k+1) - I(t_k)}{h} = \frac{K}{Ti} \cdot e(t_k) \tag{2.15}$$

Where $I(t_k+1)$ is the integral term at the actual time-step plus one.

$$I(t_k+1) = I(t_k) + K \cdot h \cdot \frac{e(t_k+1)}{T_i} \tag{2.16}$$

Approximating the derivative term in Equation 2.8 with the forward-difference method yields:

$$\frac{T_d}{N} \cdot \frac{D(t_k+1) - D(t_k)}{h} + D(t_k) = -K \cdot T_d \cdot \frac{y(t_k+1) - y(t_k)}{h} \tag{2.17}$$

$$D(t_k+1) = \big(1 - \frac{N \cdot h}{T_d}\big) \cdot D(t_k) - K \cdot N\big(y(t_k+1) - y(t_k)\big) \tag{2.18}$$

Where t_k is a specific time-step and $t_k + 1$ is one time-step forward. In other words, the forward-difference is an approximation on the next derivative or integral term.

Backward-Difference

The approximation of s using the backward-difference method is:

$$s = \frac{z-1}{z_h} \tag{2.19}$$

Once again as the name suggests, this approximation is the equivalent of replacing all the derivatives with a backward-difference.

$$\frac{dx(t)}{dt} = \frac{x(t) - x(t-h)}{h} \tag{2.20}$$

Approximating the integral term with backward-difference yields:

$$\frac{I(t_k) - I(t_k - 1))}{h} = \frac{K}{Ti} \cdot e(t_k) \tag{2.21}$$

$$I(t_k) = I(t_k - 1) + \frac{K \cdot h}{Ti} \cdot e(t_k) \tag{2.22}$$

Approximating the derivative term with backward-difference yields:

$$\frac{T_d}{N} \cdot \frac{D(t_k) - D(t_k - 1)}{h} + D(t_k) = -K \cdot T_d \cdot \frac{y(t_k) - y(t_k - 1)}{h} \tag{2.23}$$

$$D(t_k) = \frac{T_d}{T_d + N \cdot h} \cdot D(t_k - 1) - \frac{K \cdot T_d \cdot N}{T_d + N \cdot h}\big(y(t_k) - y(t_k - 1)\big) \tag{2.24}$$

This is the method used in Grahovic and Rosicki's Simulink model.

Tustin's Approximation

The discretization of the frequency using Tustin's Approximation yields:

$$s = \frac{2}{h} \cdot \frac{z-1}{z+1} \tag{2.25}$$

Approximating the integral term with Tustin's approximation yields:

$$I(t_k + 1) = I(t_k) + K \cdot h \cdot \frac{e(t_k + 1) + e(t_k)}{2 \cdot T_i} \tag{2.26}$$

Approximating the derivative term with Tustin's approximation yields:

$$D(t_k) = \frac{2 \cdot T_d - N \cdot h}{2 \cdot T_d + N \cdot h} \cdot D(t_k - 1) - \frac{2 \cdot K \cdot T_d \cdot N}{2 \cdot T_d + N \cdot h} \cdot \big(y(t_k) - y(t_k - 1)\big) \tag{2.27}$$

2.2.3 Integrator Windup

Phenomena

A controller will always strive to find the output that will yield the correct results, i.e. zero error. Although the controller can output any value, there are always physical limitations to the output such as the maximum torque from a motor. Consequently, the output must be saturated against that physical limitation. A system with both an actuator that saturates the system and an integral term will give rise to integrator windup.

Integrator windup is the phenomenon when the error is so large that the actuator saturates the output and the integrator saturates the feedback loop and builds up a large accumulated error over time. The error could become large enough to inhibit the system from adjusting to a change fast enough. In other words, the system's "reflex" is sluggish and the system is slow to react to a rapid change in the opposite direction.

To impede integrator windup, the system could either be tuned to avoid saturation or by disabling its integral term when the system is far away from steady-state. The method used in Grahovic and Rosicki's Simulink model is the back-calculation method, also known as the tracking method.

Tracking Method

The tracking method recalculates the integral term if the controller is saturated. If the controller saturates, the integral term will adjust itself to the difference between the input value and the saturation limit. This method counters the integrator windup and ensures that the integrator does not integrate up to very large numbers.

2.3 Electronic Control Unit - ECU

An electronic control unit, more commonly known as an ECU, is a device responsible for regulating an electronic system in, for example, a vehicle [15]. The general function of an ECU is to collect input signals from various sensors to process and control one or more outputs.

The outputs can be signals to actuators, pumps, motors to control windows, anti-skid systems, and engines, to name a few examples. The engine control unit is often colloquially referred to as ECU as well. As the name implies, it is an electronic control unit that controls the engine such as fuel injection and the ignition timing [16].

ECUs are comprised of printed circuit boards, or PCBs. A PCB is the mechanical support which houses and connects an electronic system [17]. The mechanical base structure of a printed circuit board consists of layers of conductive material, such as copper, and sheet layers of a non-conductive substrate [18].

2.3.1 Embedded Components - Micro-Controllers

A micro-controller is a compact integrated circuit designed to govern a specific operation in an embedded system [19]. The microcontroller consists of three main elements:

Central Processing Unit (CPU) The central processing unit is the brain of the controller, where all logic actions take place and where all the functions are executed.

Memory The memory is split into the program memory and the data memory.

- The program memory is the long term memory, the memory concerning the different instructions that the CPU performs.
- The data memory is a temporary memory used as a hold while different instructions are being executed.

I/O peripherals The input/output peripherals is the interface for the processor with the outside world.

2.4 Controller Area Network - CAN

The Controller Area Network, more commonly known as CAN, is a network used for safe and efficient communication between nodes such as ECUs, sensors and actuators [20, 21]. The CAN-bus allows communication without having a host computer or a wire-to-wire network. All communication is sent via the CAN-bus, using a specific message format and identification number, to all nodes regardless of the message's intended recipient.

Each node listens and based on the sender's ID determines whether or not the message is interesting or not. The CAN-bus is kept functioning and collision-free by using a "carrier-sense, multiple-access protocol with collision detection and arbitration on message priority (CSMA/CD+AMP)" [22].

2.4.1 CAN Function

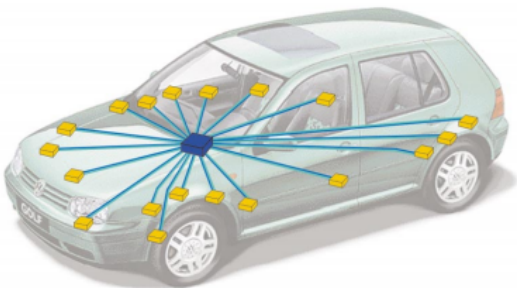


Figure 2.8: One ECU network [23].

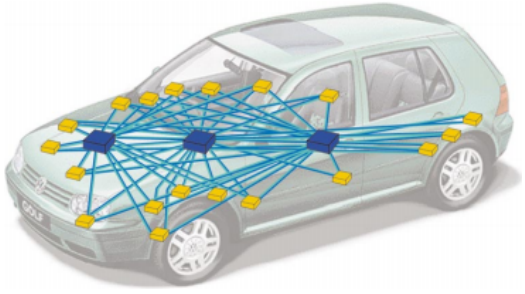


Figure 2.9: Three ECU network [23].

A single ECU system with several sensors requires a wire from each sensor, as shown in Figure 2.8. The number of wires are tripled if three ECUs require the same sensors since each node/ECU requires an individual connection to each sensor (Figure 2.9). Instead, a CAN-bus system can be used where each sensor only connects to the

nearest ECU, but all ECUs are connected to each other via a CAN-bus and share information. By using the CAN-bus "highway" system the number of wires can be kept to a minimum while ensuring robust and prioritized communication between all systems as shown in Figure 2.10 [20, 21, 22].

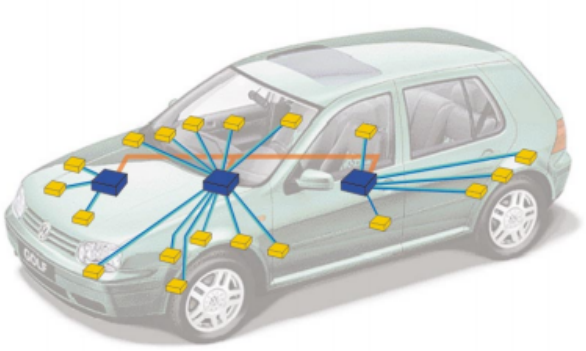


Figure 2.10: Three ECU CAN-bus [23].

Sending Messages

The send/receive process using CAN, shown in Figure 2.11, is consequently more complicated than in a wire-to-wire network. Each node must follow a strict communication protocol in order to, for example, display the car's RPM on the dashboard. The step-by-step process would begin with the engine ECU continuously checking the engine RPM from the engine speed sensor. The engine ECU subsequently packages the information as a CAN message with (a simplified) ID "engine_1" and places the message in a transmit mailbox.

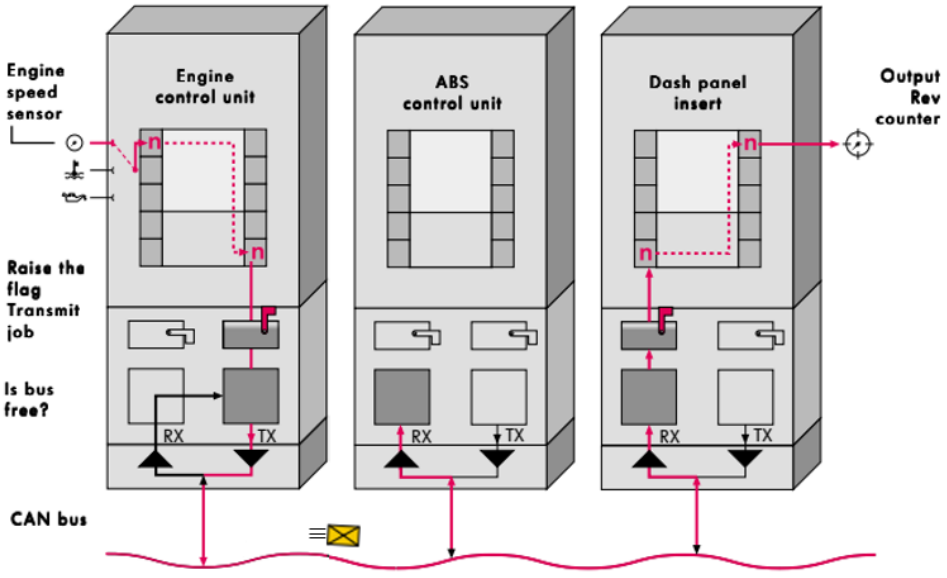


Figure 2.11: Example of CAN message transmission and reception [23].

The message containing the RPM is put on hold in the transmit mailbox until no ongoing transmissions are present on the CAN-bus before sending. Should a collision occur and two messages are sent simultaneously they will arbitrate and the message with the lowest priority will cease its send attempt [20, 21, 23].

Receiving Messages

Meanwhile, all other ECUs listen to the CAN-bus for incoming transmissions. Although it has no use for the engine RPM, the ABS ECU still receives the CAN message. Upon receiving the CAN message the ABS ECU initially checks the message for errors before checking the message ID.

Since the ID "engine_1" is not relevant for the ABS ECU it ignores the message. Meanwhile, the dashboard ECU also checked for errors and determined that the message was relevant and placed the message in a receive mailbox. There, it decodes the message and finally displays the RPM on the dashboard for the driver to see [20, 21, 23].

2.4.2 The CAN Message

There are three conventional types of CAN messages, each with a unique function:

- **Data frame** "Hey everyone, here is this information."
- **Remote frame** "Hey all, I'd like to have this information please."
- **Error frame** "Oops something went wrong, I didn't understand. Can you say again?"

Both the data frame and the remote frame follow the framing rules of a CAN message as described below. Although rarely used, a remote frame allows for a request-response type of system where the nodes must request the data in order to receive it. Unlike the other two, the error frame is a special message that does not follow the message format and is transmitted once a node detects an error [24, 25]. It interrupts bus traffic and causes all other nodes to discard the message upon which the transmitting node will resend the CAN message.

An overflow frame also exists but has largely been discontinued. It was similar to an error frame but used as a "busy" signal when a node was overrun with messages. Modern CAN handles the messages more efficiently and the overflow frame has, for the most part, become obsolete [24].

A CAN message, shown in Figure 2.12, is comprised of four main fields; the arbitration field, the data field, the Cyclic Redundancy Check (CRC) field and the acknowledge field [26]. The arbitration field is mainly comprised of the message identification number and is used to both identify and prioritize the message. After the arbitration field comes four Data Length Code (DLC) bits which determine the data length between 0 to 8 bytes, followed by the actual data.

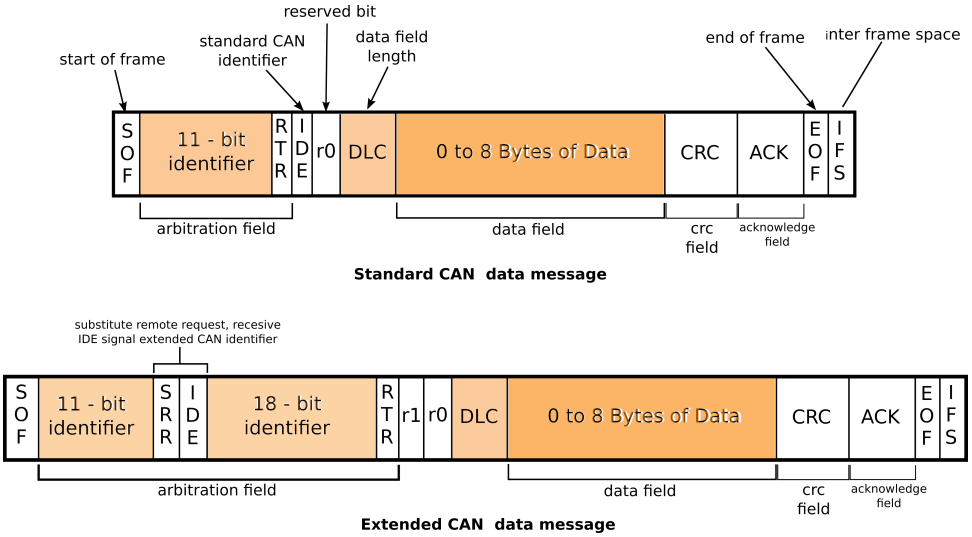


Figure 2.12: Standard and Extended CAN messages [25].

A checksum is calculated from the message data and sent with the message in the Cyclic Redundancy Check (CRC) field. The receiving node uses the same method to calculate a checksum from the received message and compares the two checksum values. If they are the same then the data is intact, if they differ then an error occurred. The subsequent acknowledge field is then used to acknowledge to the transmitting node that the message was received without errors. Each message begins with a dominant (0) Start of Frame (SOF) bit and an End of Frame (EOF) bit. At least three recessive (1) bits, known as the Inter Frame Space (IFS), are appended after the EOF-bit to separate each CAN message [25, 26].

Arbitration and Message Priority

The arbitration field varies in length between Standard and Extended CAN formats. The Standard CAN contains an 11-bit identifier used to identify the message, allowing for 2048 unique identification numbers. It ends with a bit called Remote Transmission Request (RTR) used to broadcast that a message is a data frame or a remote frame. Immediately following the RTR bit is the Identifier Extension (IDE) bit which determines if the CAN message is in the Standard format or the Extended format [24, 25].

An Extended CAN message is similar to the Standard CAN, but appends an 18-bit identifier after the Standard CAN-like arbitration field, as shown in Figure 2.12. A Standard CAN message will during arbitration always have priority over an Extended CAN message, which is determined by the recessive (1) Substitute Remote Request (SRR) bit. Consequently, the RTR-bit is moved to the end of the Extended arbitration field [27].

Between one and two reserved and unused bits separate the arbitration field from the data field. If two messages collide the CAN will automatically arbitrate and send the most prioritized message as determined by the message ID's dominant bits. In other words, the lowest CAN ID is the most prioritized.

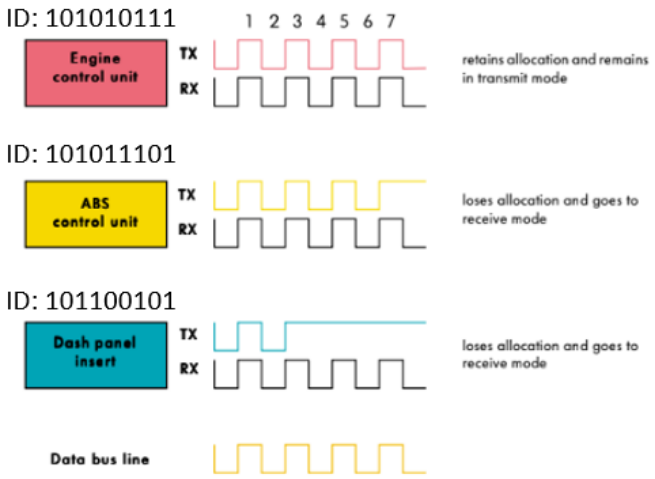


Figure 2.13: Example of arbitration based on CAN identification number [23].

The identification numbers are compared bit by bit, as shown in Figure 2.13, and as soon as a node outputs a bit that is recessive while it detects a bit that is dominant it loses allocation and stops transmission. Since each ID is unique, only one can "finish" broadcasting its identifier and therefore only one can transmit its message at a time [21, 22].

Error Frame and Error Detection

Each CAN node will check the message for errors. If a node detects an error it will send an error frame causing the transmitted message to be discarded by all nodes and the message resent by the transmitting node. Without some secondary error handling, however, a faulty node that continuously destroys each message due to error would inhibit all traffic on the CAN-bus. Therefore, each node contains two error counters in case a faulty node keeps failing; the Transmit Error Counter and the Receive Error Counter [27, 28].

In short, each error that occurs during transmission or reception will increase the corresponding counter by 8 points and 1 point respectively. The transmit error counter increments at a faster rate as it is most likely that the transmitter is at fault. Once the counter reaches 127 the node will switch from "Error Active" mode - where it throws message-destroying dominant Error frames - to "Error Passive". In "Error Passive" the node will start sending error frames but with recessive bits. Consequently, the error counter keeps incrementing if something fails but the faulty node does not destroy communication on the CAN-bus. After reaching 255 the node will switch off and no longer attempt to transmit anything. Each successful message will decrease its respective counter by one [28]. The CAN protocol uses at least five ways of detecting errors [27, 28]:

Bit monitoring - Each node will return the currently transmitted bit. If the received and transmitted bits differ, the node throws a Bit Error frame. No Bit Error is thrown during arbitration.

Bit stuffing - If five consecutive bits have the same level (high or low) the transmitting node will append one bit of the opposite level (low or high). The

appended bit is ignored by the receiving nodes. It is used to detect DC faults and provides another error check. Errors throw a Stuff Error.

Frame check - Each node checks that the CAN message follows the correct format as described in Figure 2.12. An incorrect message format throws a Form Error.

Acknowledgment check - Every receiving node will transmit a dominant bit in the ACK-slot of the message, regardless of whether or not the node was "interested" in the message. The transmitting node will transmit a recessive ACK and will throw an ACK Error if it does not detect a returned dominant bit.

Cyclic Redundancy Check (CRC) - Each node will individually calculate a checksum from the CAN message. If the message checksum is not equal to the calculated checksum, then a CRC Error is thrown.

2.4.3 The Physical Layer

There are several physical standards for CAN. The most common is the "High Speed CAN," ISO 11898-2. It utilizes a two-wire CAN-bus with a standard maximum speed of 1 Mbit/s. The maximum cable length is limited by the speed of light, as the forward part of the signal wave must be able to travel up and down the entire CAN-bus before the next bit is sent. A CAN system at a bit rate of 500 kbit/s equates to a maximum cable length of around 100 m [26, 29].

Termination

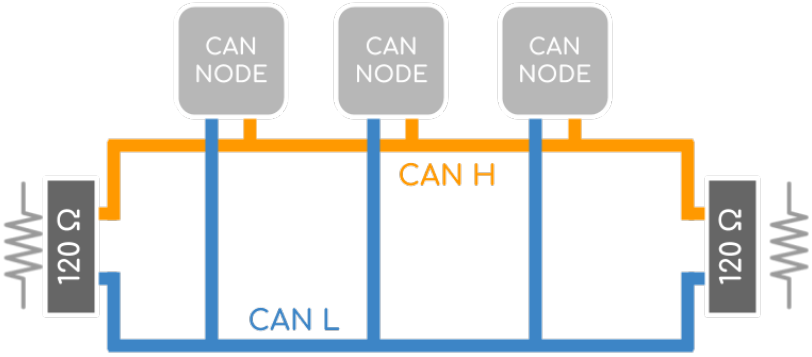


Figure 2.14: A CAN-bus with two termination resistors [30].

Due to the two-way communication on a CAN-bus, it can be interfered with or disrupted when CAN signals reflect from the cable ends. Consequently, a termination resistor is placed at both cable ends to absorb the energy, as shown in Figure 2.14. Longer cables and higher bit rates have a larger issue with reflections. A termination resistor of 120Ω is the most common, as the "termination [resistor] should match the nominal impedance of the cables" [30] as defined by ISO 11898-2 for High Speed CAN.

Chapter 3

Concept Development

This chapter analyzes possible hazards in the development of the product through risk and Automotive Safety and Integrity Level (ASIL) analysis. It proceeds to determine the needs and requirements of the TVECU in order to define the target specifications. Finally, hardware and software concepts are screened and selected according to the target specifications.

3.1 Risk Analysis and Strategy

The technical risk analysis was a continuous and iterative process that looked at both areas related to our own design and work as well as areas in which we had no control over. It also took the tiered ambition level (see *1.2.1 Tiered Ambition Level*) into consideration where the scope would be adjusted based on the availability of time and resources.

For example, the optimal situation was to have an FS-car available to test the TVECU+TVDC system on but it was not feasible if the TVDC was not produced or no FS-car was available. Therefore the system needed to be designed and tested with the assumption of a worst-case scenario. Furthermore, it was decided that anything that could be incorporated into the TVECU should be implemented to avoid relying too much on the FS-team. The iterated "Risk Matrix - Technical" is shown in Appendix A.

A notable decision was to use two separate CAN-buses in the system. Both the FS-car and the Gen6 actuators use CAN to communicate. Using the same CAN-bus as the rest of the FS-car is advantageous as it simplifies data logging and reduces the complexity of the TVECU.

Nevertheless, it opened up for compatibility issues and function failures as a result of the TVDC operating on the same CAN-bus as the ever-changing FS CAN. By using two different CAN-buses this problem can be avoided altogether. It is also possible to go from two CAN-buses to one with fairly small modifications.

3.1.1 Automotive Safety and Integrity Level (ISO 26262)

The Automotive Safety and Integrity Level (ISO 26262) is a risk-based safety standard used in the automotive industry. It applies to all electronic systems in the production vehicle and is used to ensure sufficient safety margins relative to the risk (probability and consequence) of a potential failure [31]. Although the ASIL is not formally required for the TVECU, an ASIL rating gives a sense of safety level ambition.

There are three factors that determine the ASIL of a component; severity, exposure, and controllability. Severity is the possible personal and property damage that can occur during system failure and is ranked S0-S3, i.e. no injuries to life-threatening. Exposure is the likelihood that an injury would occur given that the system has failed and ranges from E0-E4, i.e. incredibly unlikely to high probability. Controllability is the likelihood that an accident can be avoided when a hazardous system-failure has occurred. The controllability is ranked from C0-C3, i.e. controllable in general to difficult to control or uncontrollable.

It was determined that there are potential light to moderate injuries during a hazardous system failure (S1). It was also determined that the likelihood of injury during a hazardous system failure can only occur in rare circumstances (E1). The ability to avoid harm during a failure is at worst considered normally controllable (C2) as most drivers could take measures to avoid harm.

Table 3.1: ASIL table for the TVECU system.

Factor	Class/Grade	Class Description
Severity	S1	Light to moderate injuries
Exposure	E1	Very low probability
Controllability	C2	Normally controllable
ASIL	QM	Quality Management, no requirements

ASIL has four possible values ranging from A, the minimum risk with fewest compliance measures, to D with the greatest risk and strictest compliance. The combination S1 E1 C2 does not generate an ASIL value, and the ASIL is instead noted as Quality Management signifying there is no safety requirement.

3.2 Identifying Customer Needs

As the TVDC is designed for Formula Student, their demands and needs for the technical specifications were the foundation for the project. Interviews were held with representatives from BorgWarner as well as applicable students from Lund Formula Student. A list of customer needs was developed in cooperation with BorgWarner, as well as latent needs identified during internal discussions. The needs were expanded into system requirements and detailed needs required to fulfill the system requirement. The needs were then grouped into their relevant subsystem and the importance evaluated.

3.2.1 Categorized Product Needs and Requirements

The identified list of needs for the TVECU is shown below in Table 3.2. The importance denotes how the need relates to the basic function of the TVECU and its use, ranked 1-3, with 3 denoting critically important needs.

Table 3.2: Product needs and requirements categorized into packaging, hardware and software.

Nº	Subsystem			Need	Importance
1		H/W	S/W	Easy to connect to FS CAN-bus	3
2		H/W	S/W	Robust connection to TVDC CAN-bus	3
3		H/W	S/W	Accurate PI controller	3
4		H/W	S/W	Runs at high speed	3
5			S/W	Protected from wind-up	3
6			S/W	Protected from saturation errors	3
7			S/W	Handles locked wheels	3
8	PACK	H/W	S/W	Inhibited from damaging TVDC	3
9	PACK	H/W		User is protected from electrical shocks	3
10	PACK	H/W	S/W	Unit operates in all temperatures	3
11		H/W	S/W	Fast start up time	3
12	PACK	H/W	S/W	Unit is easy to reprogram/configure	2
13			S/W	Unit handles over-steering	2
14		H/W	S/W	Runs or easy transfer from Simulink	2
15	PACK	H/W		Local sensors prioritized	2
16			S/W	Contains fail-safe for TVDC failure	2
17			S/W	Contains fail-safe for electrical failure	2
18	PACK	H/W		Light	1
19	PACK	H/W		Efficient size	1
20		H/W	S/W	Automatic reboot watchdog	1
21	PACK			Weather resistant	1
22	PACK	H/W		Adequate life expectancy	1
23		H/W	S/W	Logs data locally	1
24	PACK	H/W	S/W	Data log easily accessible	1

3.3 Target Specifications

The customer needs in Table 3.2 were converted into measurable metrics in order to establish a target specification for the TVECU, shown in Table 3.3. Some needs fit into a single metric, while some needs reflect more than one or a certain range the TVECU has to function within. The metrics were also divided into acceptable and ideal goals, in accordance with the tiered goals method described in *1.2.1 Tiered Ambition Level*.

Table 3.3: List of target metrics as developed from the product needs.

Nº	Metric	Imp.	N.Nº	Optimal	Acceptable
1	Simple FS CAN	3	1	Y	Y
2	Robust TVDC CAN				
	Robust CAN-bus	3	2	Y	N
	Shock protection	3	9	Y	Y
	Local sensors prioritized	2	15	Y	Y
3	High Speed	3	4	>50 Hz	>10 Hz
4	Safety Handles				
	No TVDC damage	3	8	Y	N
	Shock protection	3	9	Y	Y
5	Operating Temperature	3	10	-20 − 80°C	-5 − 60°C
6	Fast Startup				
	Fast Startup Time	3	11	<1 s	<3 s
	Reboot Watchdog	1	20	Y	N
7	Simple Configuration				
	Easy to reprogram	2	12	Y	N
	Simulink compatible	2	14	Y	N
8	Small Control Error	3	3	± 5%	± 15%
9	Failure Handles				
	TVDC failure	2	16	Y	N
	Electrical failure	2	17	Y	N
10	Special Case Handles				
	Wind-up	3	5	Y	Y
	Saturation errors	3	6	Y	Y
	Locked wheels	3	7	Y	N
	Over-steering errors	2	13	Y	N
11	Low weight	1	18	<25 g	<100 g
12	Efficient Packaging				
	Efficient size	1	19	Y	N
	Weather resistant	1	21	Y	N
13	Data Logging				
	Local data log	2	16	Y	N
	Accessible data log	2	17	Y	N
14	Lifespan	1	22	>10 yrs	>2 yrs

3.3.1 Competitive Benchmarking and Patent Research

In order to evaluate the target specifications, they were compared with other similar commercial systems available on the market. Although no systems were identical in use and needs as the TVECU, Bosch sells an engine ECU for automotive use that controls the engine and communicates via CAN. Their ECU was used to get a feel for the metrics in an automotive application and is shown in Table 3.4.

Table 3.4: Engine ECU specifications from Bosch [32].

CPU Speed	80 – 300	MHz
Flash Memory	1.5 - 8	MB
RAM	72 - 628	KB

Patent research and analysis was also conducted via the European Patent Office (EPO) and the US Patent and Trademark Office (USPTO). The goal was to see what innovations exist and are patented, as well as to get inspiration for non-patent-infringing ideas. Nevertheless, given the specific nature of the TVECU, no such systems were found that were adequately similar to the needs and specs of the TVECU.

3.4 Concept Generation

Concept Generation aimed to develop and screen hardware and software design ideas. The software's main signals were to be properly designed and the hardware determined and ordered as soon as possible to avoid consequences from supplier delays. It was conducted iteratively and in parallel with *4 System-Level Design* to ensure that the design was in line with the *3.3 Target Specifications*.

3.4.1 Hardware

The TVECU system was initially to be developed using existing hardware before creating a proprietary printed circuit board (PCB) for the TVECU, in accordance with *1.2.1 Tiered Ambition Level*. The hardware selection process involved creating a wide list of possible commercial hardware that can be used to develop the TVECU hardware.

Hardware Screening

The concept screening pertained to evaluating the hardware and eliminating unfavorable candidates in an effort to determine which is most relevant to our needs, and which has the greatest potential to meet our requirements. The hardware shortlist was initially screened on two criteria. First, whether or not there are duplicate or similar hardware, e.g. hardware based off of the open-source Arduino. Second, whether or not we had experience with the hardware or similar.

The PocketBeagle, STM32 Discovery, Teensy++, and the two Raspberry Pis are relatively similar and were considered "duplicates". Since we had experience with Raspberry Pi, the other three micro-controllers/micro-processors were stricken from the list. The remaining hardware were screened based on development (alpha-prototype) and final product (assembled and completed beta-prototype) needs.

For example, stackable headers and pins mean faster development during early prototyping, while the size is of little importance. Conversely, size is important for the beta-prototype/final product and the availability of stackable headers and pins are not as the wires are soldered onto the PCB. Special consideration was also taken in regards to four risks:

- CAN communication inhibits serial communication (metric 1 and 2)
- CAN availability/compatibility (metric 1 and 2)
- One processor not enough to handle control software (metric 3, 6 and 8)
- Memory not enough for control software (general function)

An attempt was made during pre-development to deploy the theoretical Simulink model created by Grahovic and Rosicki to an Arduino Uno. The model failed to deploy as the Simulink generated code requires more memory than what is available on an Arduino Uno. While deployment methods and software generation were further evaluated in *4.2.1 Software Deployment Methods*, the large size of the Simulink model showed a potential limitation of the hardware. Consequently, the list was further narrowed down by removing the Arduino Uno and Pro Mini as well as the SparkFun RedBoard and Pro Micro. The hardware boards that passed screening are:

- Arduino Mega
- Raspberry Pi 3 B+
- Raspberry Pi Zero W
- SparkFun RedBoard Artemis
- SparkFun AST-CAN485

Hardware Selection

The selection process involved reasoning around the hardware specs and needs of the system, as well as the needs during development and final assembly. The most important specs are outlined in Table 3.5.

While all hardware that passed screening were adequate, different micro-controllers and micro-processors have different advantages. Despite being the largest of the bunch, the Arduino Mega provides the most headers and four serial connections for dedicated communication. However, the Arduino Mega does not have a dedicated CAN PCB (called Shield, Hat or Breakout) and would have to be modified manually.

Table 3.5: Hardware specifications of hardware under evaluation.

	Ard. Mega	Pi 3 B+	Pi Zero	Artemis	AST-CAN485
Memory	256 KB	Ext μSD	Ext μSD	1 MB	128 KB
RAM	8 KB	1 GB SDRAM	512 MB	384 KB	4 KB
Clock Speed	16 MHz	1.4 GHz	1 GHz	96 MHz	16 MHz
Size [mm]	102x54	85x56	65x30	69x53	33x18
CAN	Separate	Shield	Shield	Shield	Built-In
Serial	4	2	1	2	2

The Raspberry Pi 3 B+ is a multi-core micro-processor that is by a large margin the most powerful of the screened hardware. It would allow multi-threaded programming at a very high speed and smaller size-factor than the Mega. The Raspberry Pi Zero offers an even smaller size-factor while retaining most of the Pi 3 B+'s impressive clock speed and memory. Both of the Raspberry Pis have CAN Shields available, but they both run Raspian OS. Consequently, their use in embedded systems is unusual and their Simulink compatibility limited.

The RedBoard Artemis is somewhat of a compromise. as it can be programmed using an Arduino programmer and is thus compatible with Simulink. On the other hand, it is less powerful and also relies on a CAN Shield. The last alternative is the AST-CAN485 which is the least powerful of the shortlist, but is by far the smallest and has built-in CAN hardware.

Conclusion

None of the micro-controllers have the same specifications as Bosch's engine ECU in Table 3.4, but their ECU has to handle more data and is not an adequate "one-to-one" comparison. The hardware in the final shortlist were all deemed acceptable with different pros and cons.

Two AST-CAN485 were chosen for the beta-prototype, and final product, as the size and CAN advantages were overwhelmingly positive. Two RedBoard Artemis were also ordered to allow for rapid prototyping during the development phase. Although there are several different CAN Shields on the market it was decided to order the CAN Shield from the same supplier as the AST-CAN485 and the RedBoard Artemis. The Raspberry Pis were kept in reserve, but not ordered, should problems arise, however unlikely, with regards to the risks described in *3.4.1 Hardware Screening*.

3.4.2 Software

The concept generation started with a thorough analysis of the theoretical model provided by Grahovic and Rosicki, and a rework of the signal overview shown in Figure 3.1 for practical purposes. The specifics around the model, in-signal types and sources were modified but, the essence of the control method was kept unchanged.

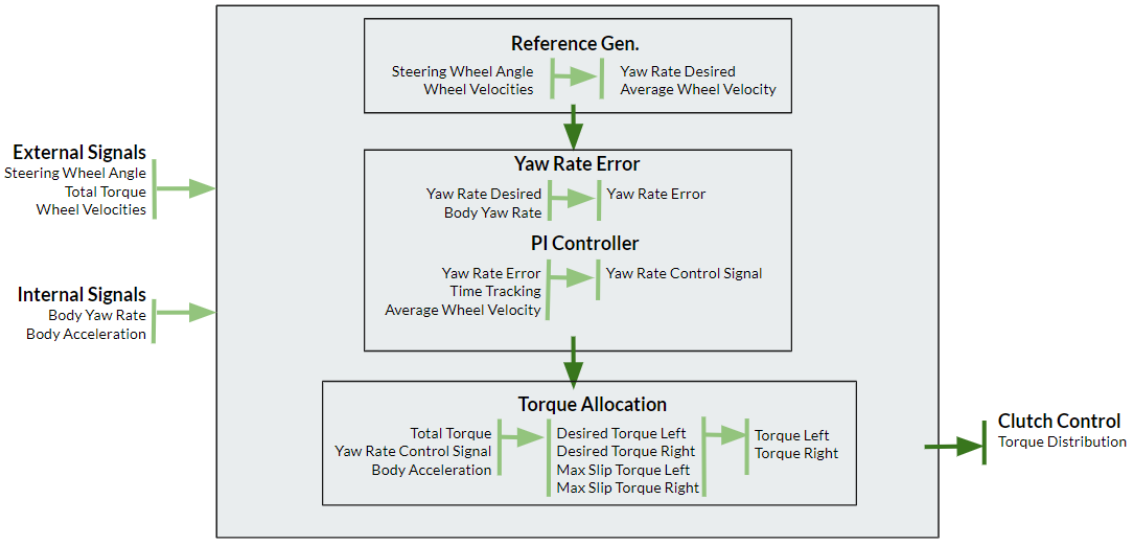


Figure 3.1: Overview of signals for practical control reworked from the theoretical model.

Chapter 4

System-Level Design

This chapter explains the high-level hardware design of the TVECU and its additional components. It describes supporting development systems used for CAN communication, and tests the alpha-prototype against a commercial engine ECU. Finally, it evaluates software deployment methods and describes the general process of the TVECU software.

The resulting torque vectoring system, shown in Figure 4.1, is comprised of an ECU (i.e. the TVECU) to control the TVDC, which receives all external signals from the FS CAN-bus. After processing the TVECU sends a control signal to the TVDC actuators/clutches via the TVDC CAN-bus. The TVECU would also contain an accelerometer and gyroscope to ensure function, as decided in *3.1 Risk Analysis and Strategy*.

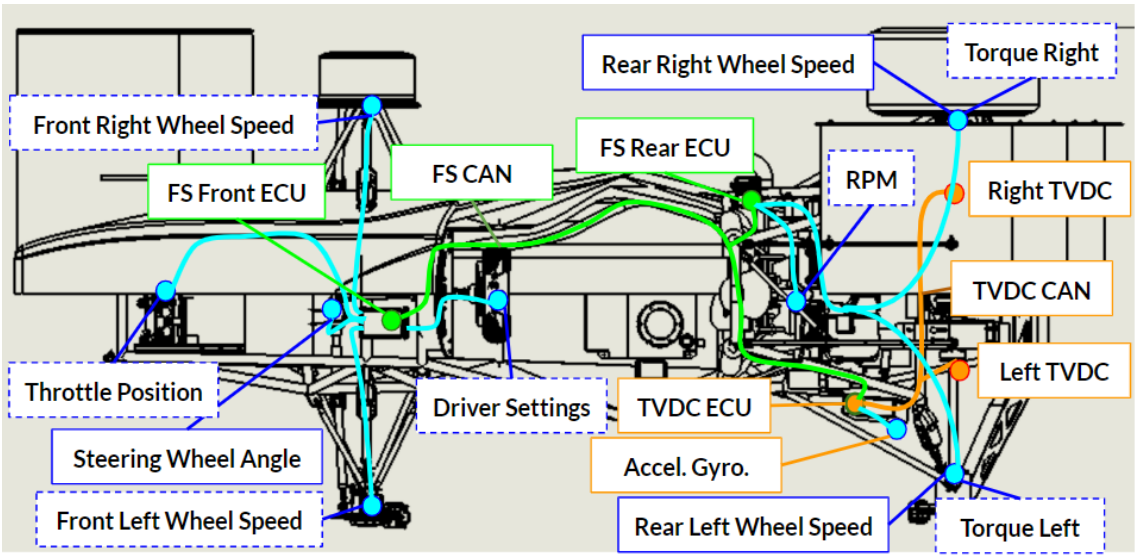


Figure 4.1: Chapter summary; Overview of signals on FS-car. In orange; the TVECU+TVDC system. In green; the FS system. In blue; sensor data. In dashed blue; optional but recommended sensor data.

4.1 Hardware Selection and System Design

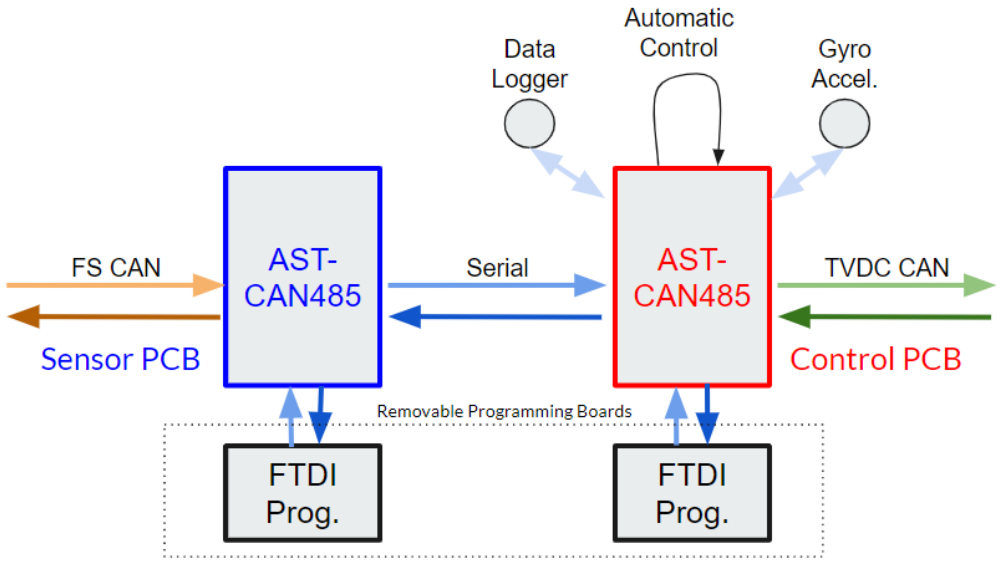


Figure 4.2: Overview of TVECU system.

The SensorPCB, whose function is to connect to the FS CAN-bus, relays its collected information to the ControlPCB, as shown in Figure 4.2. The ControlPCB, in turn, uses its sensors and the parameters from the SensorPCB to determine and output a control signal on the TVDC CAN-bus. The AST-CAN485 requires a programming tool (FTDI Breakout) which is removable to minimize the size of the final TVECU.

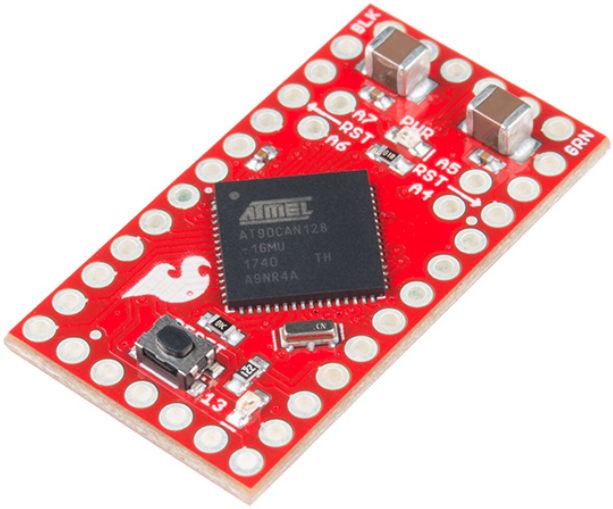


Figure 4.3: AST-CAN485 [33].

4.2 Prototyping and Development

An alpha-prototype system was first developed to simplify the development of the TVECU system in Figure 4.2. Two ECUs were created using a CAN Shield and a micro-controller. The micro-controller was initially comprised of the RedBoard Artemis, but due to the lack of compatibility with Simulink, they were initially abandoned in favor of compatible Arduino Unos, as detailed below.

Out of simplicity, the Arduino Unos were kept during the remainder of the alpha-prototype development. Their lacking technical specifications were not an issue as they were only used to test and develop small sub-problems. It was additionally decided, also outlined below, to write our own C/Arduino-code based on the Simulink model. As the AST-CAN485 is Arduino compatible we were also able to use Arduino's many libraries during development.

4.2.1 Software Deployment Methods

Three methods for deploying the control model were identified:

- Deploying the Simulink model directly
- Generating C-code with Simulink
- Writing C-code based on the Simulink model

Deploying Simulink Model Directly

Deploying the model directly from Simulink onto the hardware was the first method examined. The model was Simulink based with a sample time of 0.01 seconds. Deploying a model directly onto a micro-controller requires a compiler to convert the code into binary.

The "Simulink Support Package for Arduino Hardware" was the only hardware support package available from Simulink that worked with our hardware. Deploying simple Simulink models onto an Arduino board worked well, could be programmed with different tasks such as flashing an LED. The support package had two main functions; creating model blocks, shown in Figure 4.4, that could toggle different Arduino pins and the compilation and deployment onto an Arduino board.

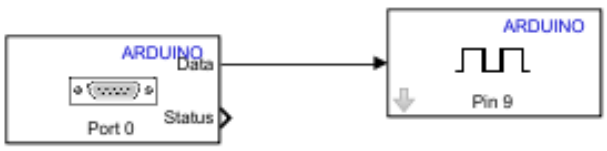


Figure 4.4: Example of Arduino blocks in the Simulink model.

A limitation of the support package was the lacking function of the CAN communication blocks. The communication did not work or failed to deploy entirely. Another limitation of the support package is that it only supports specific Arduino boards.

It was initially assumed the support package would deploy a Simulink model onto any Atmel processor. This was not the case, and the method of deploying the model directly onto the hardware was abandoned.

Generating C Code with Simulink

The second method investigated was generating C-code from Simulink instead of a Simulink block model and consisted of two parts. First, generating C or C++ code directly from a Simulink with the Embedded Coder [34]. Second, deploying the code using a programmer and Atmel Studio. Atmel Studio is an integrated development platform (IDP) for developing and debugging AVR and SAM micro-controller applications [35]. C code can be developed, compiled and uploaded onto any Atmel processor.

Without access to an external programmer required the development to proceed with using Atmel Studios' External Tool function. The external tool is written with a command and an argument. The same command used in an Arduino boot-loader was used since it functioned properly when deploying from the Arduino IDE.

Unfortunately, the generation of C-code from Simulink also generated an abundance of Simulink support files and functions. Each code file generated was complex and not intuitive which resulted in difficulties during adjustment and optimization. They also took up a disproportionate and mind-boggling amount of memory. Additionally, the CAN model blocks were not usable with this method either.

Finally, Atmel Studio was not able to build the different files together as a solution and could not compile and upload the code to one micro-controller. Given the complexity of the solution, the memory overshoot and overall non-functionality this method was deemed both to time-consuming and disproportionally difficult.

Writing C++ Code Based on Simulink Model

The final method examined was writing the C++ code from scratch. This method consists of two parts; writing C++ code based on the Simulink model and then using the Arduino boot-loader to upload the code to the micro-controller. Writing the C++ code from scratch gives the opportunity for both memory and execution efficiency, as well as creating a solid framework for reworking the model. As such, this method was deemed the most efficient, feasible and least time-consuming.

4.2.2 CAN Send/Receive

The CAN system was developed by using two development systems comprised of a CAN Shield and an Arduino Uno. The initial development started with modifications of the CAN Shields from the OBD-II pin format to raw CAN pin format, as well as the construction of a High Speed CAN cable with two 120Ω termination resistors. The two units, shown in Figure 4.5, were then connected and basic CAN send/receive test scrips were written in order to test and develop the CAN communication.

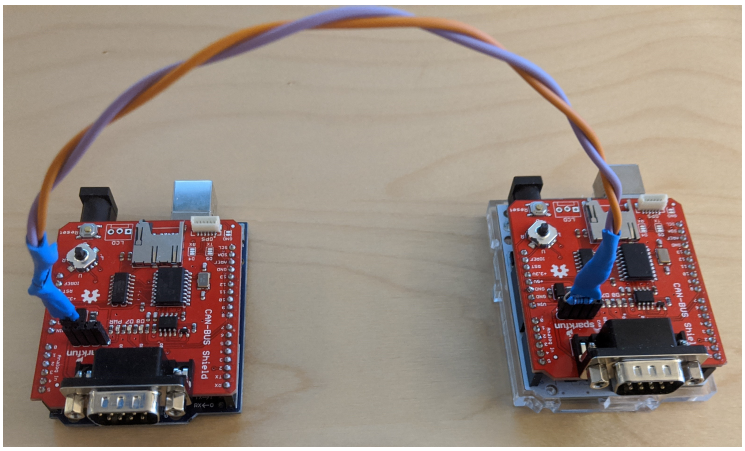


Figure 4.5: Two CAN development systems communicating via CAN-bus.

Since the scripts only tested the CAN functionality and sent the same message over and over again, the CAN send/receive was also tested using a BorgWarner proprietary engine ECU, shown in Figure 4.6. The ECU, once connected, automatically sends vehicle parameter data over CAN. This allowed us to test the receive functionality in a data-intensive environment with varying messages and message IDs. The CAN messages were also sent to the CAN software CANalyzer. The integrity of the CAN system could then be ascertained by comparing the messages received by both interfaces.

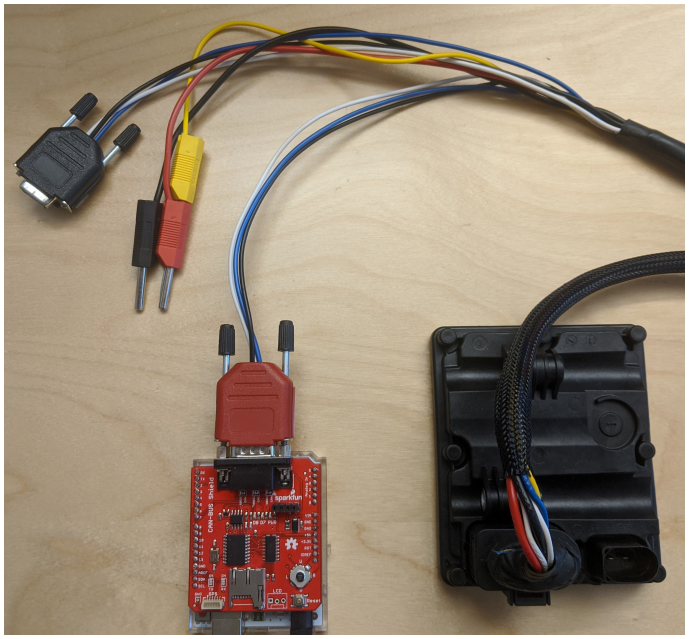


Figure 4.6: Alpha-prototype with CAN Shield tested with a BorgWarner proprietary engine ECU.

The CAN send/receive system was subsequently developed on the AST-CAN485, which varied slightly in programming commands as a result of different CAN hardware components. The tests were repeated on the AST-CAN485.

4.2.3 Interrupt Flags or Serial Flags

The ControlPCB (AST-CAN485) requires its vehicle parameters immediately when needed in order to maintain a minimal sample time. A potential solution was using interrupts to trigger data transmission from the SensorPCB. The SensorPCB loops and collects the vehicle data from the FS CAN-bus. When the ControlPCB needs the data it triggers an interrupt in the SensorPCB, which proceeds to send the vehicle data over serial.

Initially, the interrupt-routines functioned well, but as a result of limited-resource prototyping, the system was put together with a breadboard and a loose contact with the AST-CAN485 PCBs. Consequently, the movement of the loose contacts triggered unwanted interrupts. Although this would not be an issue in the final soldered product, it became a problem during development.

As a stand-in development solution, the data request from the ControlPCB to the SensorPCB was sent via serial instead of triggered via an interrupt. This method ended up working better than expected, and the need to return to the interrupt system diminished. The interrupt system works better in the sense that the ControlPCB gets its data as soon as possible, yet the control algorithm did not seem to be affected by the slightly slower serial method. As a result, Control-Sensor communication occurs exclusively via serial but is designed to also function with interrupts should the need arise in the future.

4.2.4 Accelerometer and Gyroscope

Figure 4.7: MPU-9250 IMU Breakout [33].

The MPU-9250 is used as an accelerometer, gyroscope and temperature sensor. The communication between the sensor and the micro-controller is through the I^2C interface. The sensor runs on 3.3V, requiring a voltage divider to function with the 5V TVECU.

4.2.5 Level Shifting microSD

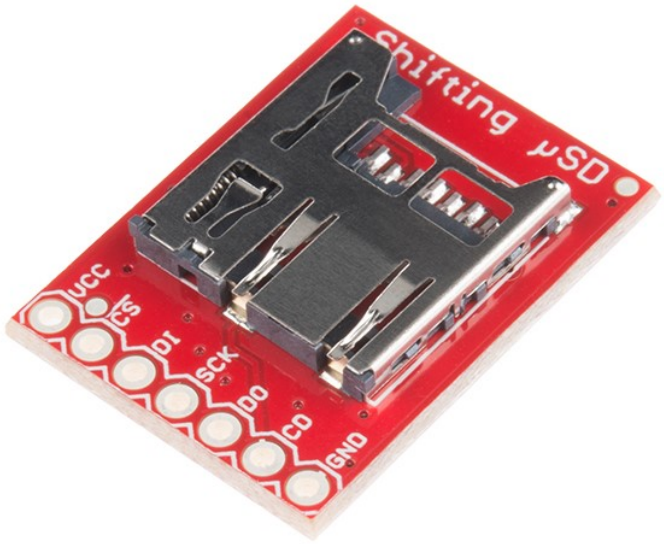


Figure 4.8: Level Shifting microSD Breakout [33].

The Level Shifting microSD adds internal data logging functionality through serial Peripheral Interface to the TVECU and is compatible with both 3.3V and 5V systems. The TVECU can also be modified to transmit its data over the FS CAN-bus, and as such allows for external data logging. Although the complete implementation of a data logging system is not within the scope of the thesis, it is vital to facilitate the future development of data logging. Tracking temperature, pressure and general functionality of the TVECU/TVDC is important during both testing and racing.

4.3 Software System Design

The SensorPCB collects the vehicle parameters via the FS CAN-bus. Upon request, it sends the vehicle parameters via serial to the ControlPCB. The ControlPCB, in turn, processes the vehicle parameters and accelerometer/gyroscope data and runs the control algorithm. It outputs a control signal via the TVDC CAN-bus to the two Gen6 actuators controlling the clutch. In order to alleviate the pressure of the serial communication line, the ControlPCB contains the accelerometer, gyroscope and data logger. Additionally, the TVECU allows "switching" the SensorPCB and ControlPCB should the speed of the ControlPCB software become an issue.

The TVECU system also contains an error handling and tracking system. Upon discovery of certain issues, such as high TVECU temperature, it will throw a warning that can be sent via the FS CAN-bus to the driver. Upon another threshold, it will throw an error and, for example, disable the TVDC until the temperature is low enough again. Finally, the TVECU allows internal data logging; after small tweaks, it is also compatible with external data loggers via the FS CAN-bus.

Chapter 5

Detail Design

This chapter develops the hardware configuration and wiring of the TVECU beta-prototype. It proceeds to address packaging and the Gen6 connectors, and examines the TVECU software and control algorithm in detail. Ultimately, it presents the final assembled product.

5.1 Hardware

The hardware system has three waterproof Gen6 connectors. The bottom right connection in Figure 5.1 leads to the TVDC torque and temperature sensors used to detect possible faults. The bottom left connection leads to a fork in the TVDC CAN-bus and supplies power to both Gen6 actuators. The CAN-bus is terminated at each actuator end, and each actuator uses an IGN cable as specified by Gen6. The IGN cable is held high and signals the Gen6 to start-up and function.

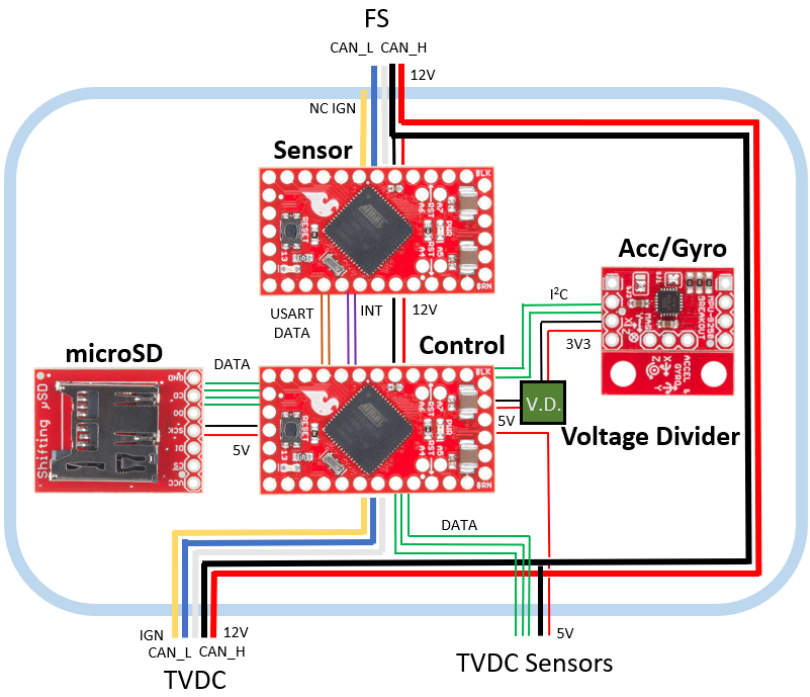


Figure 5.1: TVECU connections.

Both of the bottom connections lead to the ControlPCB, which running the control algorithm. It is also connected to the microSD for internal data logging and to the MPU-9250 sensor for acceleration and gyroscope data. A 5V pin from the ControlPCB powers both the 5V microSD and the 3.3V MPU-9250. The latter uses a voltage divider, shown in Figure 5.2 to step down the voltage. The 5V TVDC sensors are likely comprised of two 3-wire pressure sensors and one 2-wire temperature sensor but are yet to be decided by the TVDC team. As such, the pin layout is yet to be determined.

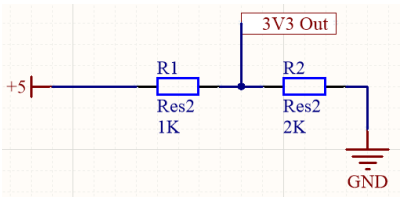


Figure 5.2: The 5V to 3.3V voltage divider.

The ControlPCB is in turn connected to the SensorPCB via USART Tx/Rx, as well as two interrupt enable pin connections, as shown in Figure 5.3. The interrupt pins are currently unused, but as discussed in *4.2.3 Interrupt Flags or Serial Flags*, they may be in the future. Both the ControlPCB and SensorPCB are connected to 12V supplied by the FS connection and use their internal voltage regulator to step down the voltage. The FS connection also contains the FS CAN-bus and a No Connection (NC) IGN cable. Although unused, the NC IGN cable is there to allow the reversal of the TVECU system by having the ControlPCB run Sensor software, and SensorPCB run Control software. In other words, it allows switching the FS and TVDC connections and consequently making the modules be attached to the SensorPCB instead of ControlPCB.

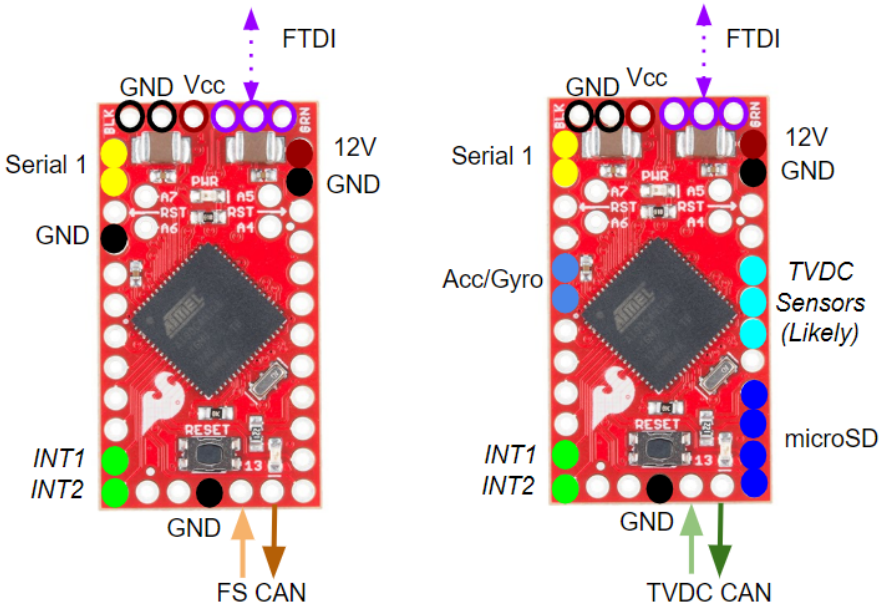


Figure 5.3: Pinouts of SensorPCB and ControlPCB.

5.1.1 Electro-Magnetic Interference and Gen6 Actuators

The ground cable from the TVDC CAN cabling is not directly connected to the ControlPCB to avoid a ground loop inside the TVECU. There is also the potential risk of electro-magnetic interference as a result of the relatively high power drawn by the Gen6 actuators. Each actuator can potentially draw 12.5A current, resulting in the possible, albeit unusual, circumstance of 25A passing near the TVECU. Although the TVECU is deemed resilient enough, it is something worth keeping in mind should problems arise.

5.2 Packaging and External Connections

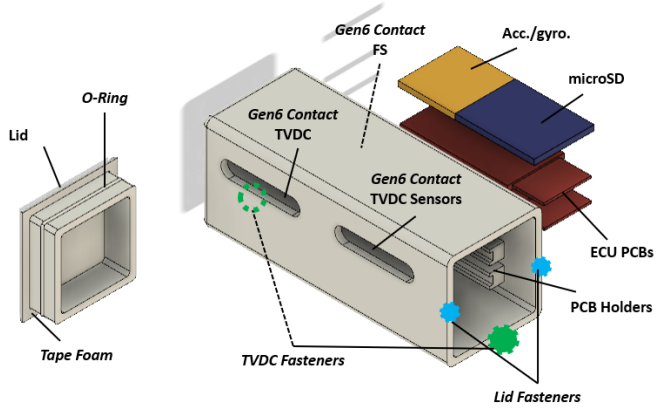


Figure 5.4: CAD of the TVECU box. Figure labels in italics are not shown.

The TVECU box is a nylon waterproof casing with slots for the PCBs, shown in Figures 5.4 and 5.5. The TVECU box is closed by a cap, sealed by an O-ring and locked in place. Three holes allow access to, and are sealed by, the waterproof Gen6 connectors in Figure 5.6. Two fasteners attach the TVECU box to the TVDC top side using vibration dampening spacers, which also distance the TVECU from the hot TVDC.

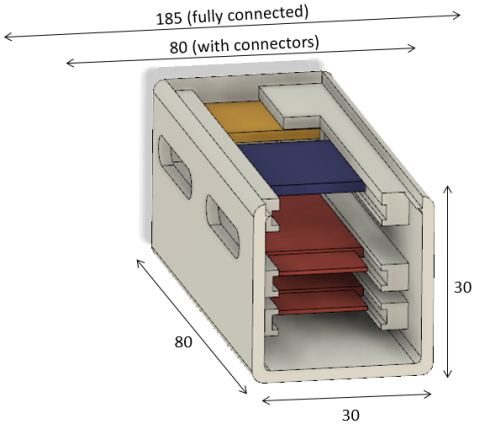


Figure 5.5: CAD of the TVECU box with top removed and PCBs in place.

Discussions with BorgWarner regarding placement of the TVECU resulted in a cleared area on the TVDC the size of the TVECU box. Nevertheless, some details regarding the exact location of the spacers and fasteners are yet to be determined after further development on the TVDC. As such, the current TVECU box lacks both lid fasteners and fasteners to the TVDC.

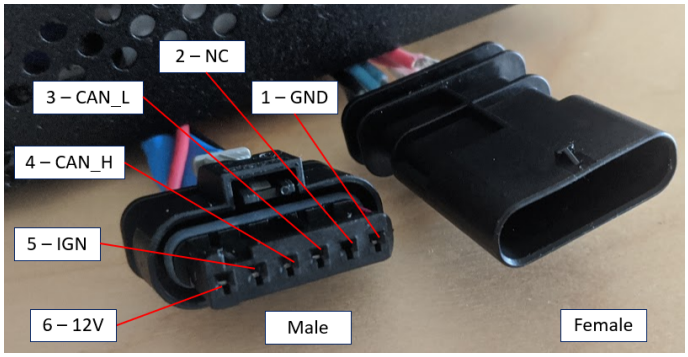


Figure 5.6: Male and female Gen6 connectors.

The potential difference of 12V, a maximum (albeit unusual) current of 25A and a cable length of 2 m necessitate a power cable thickness of at least 4 mm^2, given a 5% loss. A 3% loss requires a cable thickness of 6 mm^2.

The TVECU should never be powered by more than one source. When powered by the car, the TVECU system should only be programmed by non-powered FTDI programmers. If not powered by the car and both PCBs are attached to programmers, only one programmer should be powered. Irreversible damage to the system can occur if there are several power sources.

5.3 Main Algorithm Loop

The main continuous loop program that runs the TVECU is located on the ControlPCB. The loop begins with requesting a variable refresh from the SensorPCB to update the parameters from the FS-car. It then polls the accelerometer, gyroscope and the TVDC sensors. Using the data available, the algorithm analyzes the parameters and handles potential warnings and errors. Given no errors, the program uses the control algorithm to derive the required torque on each wheel and commands the Gen6 actuators to the correct torque via CAN.

5.3.1 Supporting Software

Sensor-Control PCB Communication

In order to request a data refresh, the ControlPCB sends an error byte (described below) to the SensorPCB via serial. Once the SensorPCB detects an incoming message it syncs the error byte and returns all the FS vehicle parameters. Each variable is packaged with start and stop markers to ensure complete data transmission.

A potential issue with the serial wake up method is the delay between variable request and delivery. Although the speed of the ControlPCB algorithm can be increased with interrupts, there are no indications of any time-related issues. Even in

the worst conditions, the algorithm runs fast enough. However, there is no certainty until the speed has verified via practical tests on an FS-car.

Error Handling

The main program loop keeps track of warnings and errors through an error byte. The algorithm throws a warning to the driver and data logger upon a certain sensor threshold. If conditions worsen and a new threshold is passed, the algorithm will throw an error and lock the TVDC until the error conditions are no longer valid. The error byte is synced across the SensorPCB and ControlPCB and also used as the wake-up signal between the two during data exchange.

CAN Messages

Table 5.1: Example IDs of the Gen6 actuators used to develop the TVECU.

Destination	Identification
Gen6 Right	0x1AFB5F10
Gen6 Left	0x1AFB5F20

The CAN message format is separate in the two categories, the TVDC CAN and the FS CAN. The TVDC CAN conforms to the commercial Gen6 actuators message format shown in Figure 5.7. The 0th byte sends the "method of function" command to the actuators whilst the 1st and 2nd bytes demand the torque required. Each actuator is individually controlled by a unique CAN ID, as shown in Table 5.1. The 3rd and 4th bytes are unused.

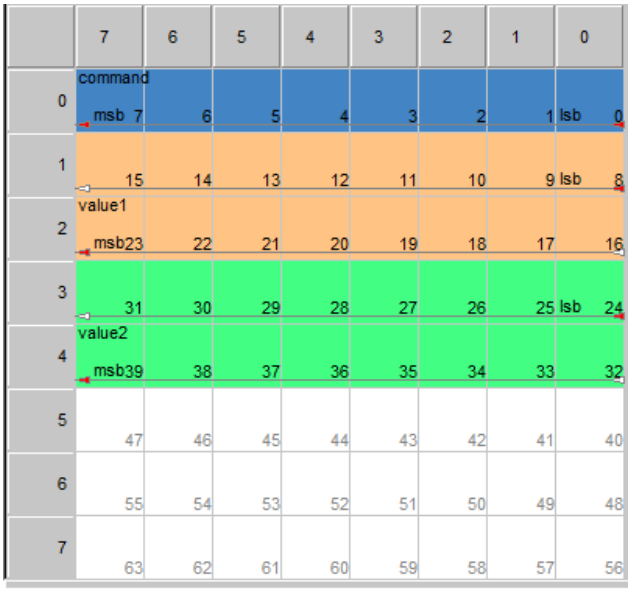


Figure 5.7: The CAN message format required by the Gen6 actuators.

While the TVDC CAN format is given, the FS CAN is able to conform to any format. Two methods of definition are available in relation to Lund Formula Student, bilateral and unilateral definitions. A unilateral definition is when the message formats are defined with regards to the TVECU only. It simplifies development, standardizes the system and allows the product to be used by other teams.

On the other hand, it requires Lund Formula Student to conform to the TVECUs (FS CAN) message format, which might result in CAN ID collisions or the need to completely revamp their CAN system. A bilateral method involves cooperating with Lund Formula Student when defining the message formats. Although more work for the TVECU development, it likely minimizes the total workload required to develop, implement and test the TVECU in the car. As such, the exact formats of the FS CAN messages are not conclusively defined. Table 5.2 shows the development definitions of the CAN messages.

Table 5.2: FS CAN message IDs and formats. Odd bytes used as high bytes and even bytes as low bytes during bit manipulation.

Message Name	ID	7_H	6_L	5_H	4_L	3_H	2_L	1_H	0_L
WheelSpeed	0x100	FL	FL	FR	FR	RL	RL	RR	RR
TorqueSteer (1)	0x101	TQ	TQ	ST	ST	-	-	-	-
TorqueSteer (2)	0x101	TQR	TQR	ST	ST	TQL	TQL	$PARA$	ERR

The WheelSpeed message is comprised of the four-wheel speeds in 16-bits split into 8-bit chunks via bit manipulation. The initial development of the TorqueSteer message contained total torque and steering wheel angle. The continued development of the TorqueSteer message splits the total torque into left and right and adds both the driver parameter byte and the error byte. The driver parameter value ranges from 0.5 to 1.5 and affects the performance of the TVDC, for more information see *5.4.3 Vehicle Reference*.

5.4 Control Algorithm

The control code is written as C++ libraries to modularize the control algorithm and simplify the implementation. The different libraries were created using a .cpp source file together with a .h header file.

The algorithm initially used the same sample time implemented in the theoretical Simulink model. The sample time of the system was updated to 12 ms after testing in *6.2 TVECU System Tests* based on the loop time of the system.

5.4.1 Controller

Theoretical Simulink Control Model

The theoretical control model designed by Grahovic and Rosicki was a PI controller with a time-tracking anti-windup system, shown in Figure 5.8 [3]. The integral and proportional gains were dynamic and based on the speed of the vehicle. The yaw rate error was the basis of the controller.

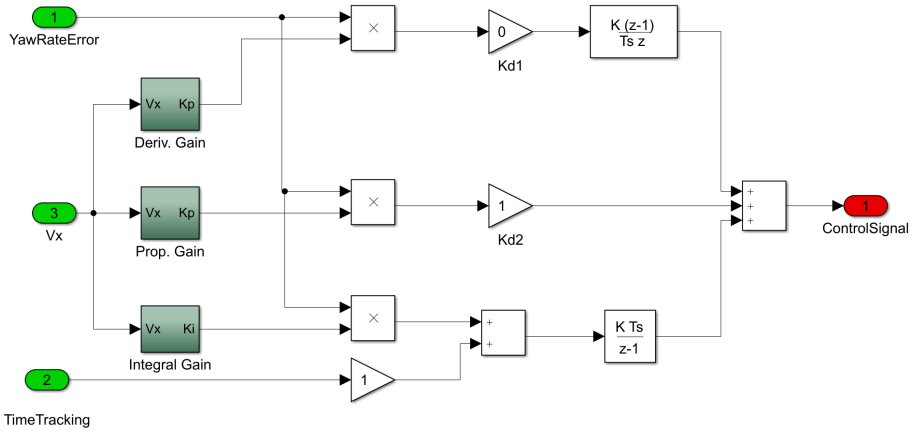


Figure 5.8: The theoretical Simulink control model.

Practical Control Algorithm

The controller implemented in C++ uses the same gain for the proportional term and the integral term. The software is based on a three attribute input; the yaw rate error, the velocity, and the anti-windup. The output from the controller is the control signal.

The proportional gain was based on the velocity of the vehicle. As such, the proportional term was implemented as the proportional gain multiplied by the error at the specified time. Equation 5.1 shows the proportional term implemented in the software.

$$P = K_p \cdot e \tag{5.1}$$

Where K_p is the proportional gain and the error e is calculated once per sample. In order to implement the integral term, the design must consider the sample time. The speed of the controller is directly connected to the sample time and the integral term of the controller is therefore directly affected by the sample time of the system. Equation 5.2 shows the integral term implemented in the software.

$$I = I_{old} + h(K_I \cdot e + \zeta) \tag{5.2}$$

Where ζ is the anti-windup and h is the sample time. Although the integral gain in *2.2.2 Discrete Implementation of PID Controller* is $K_p \cdot T_i$, the design uses a single integral gain K_I. The need for a derivative term arose after the system tests described in *6.1 Simulink Model Code Test* to account for high-speed changes and oscillations at high velocities. Equation 5.3 shows the derivative term implemented in the software, where K_d is the derivative gain.

$$D = \frac{K_d}{h}(e - e_{old}) \tag{5.3}$$

5.4.2 Torque Allocation

Theoretical Simulink Control Model

The control signal generated in the control model shown in Figure 5.9 was used to calculate the wanted torque on both the left and the right clutches. The total torque provided by the engine was subsequently split up according to the control algorithm. To avoid wheel slip the torque provided was saturated against the slip limit of the wheels.

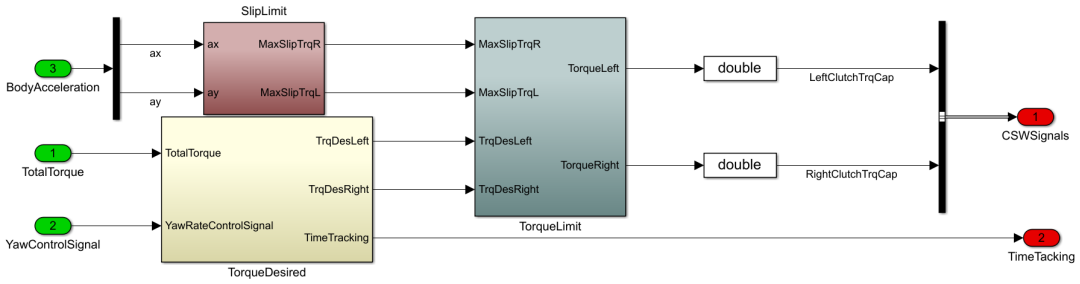


Figure 5.9: The theoretical Simulink torque allocation model.

Practical Control Algorithm

As the slip limit coefficients are only based on static vehicle parameters and the acceleration in the x and y-directions, the coefficients are dynamic only in regards to the acceleration. The static coefficients are determined through implementation and fine-tuning, and as such generally not altered. The slip torque is updated once per sample time and the coefficients are implemented in the source file according to Equations 5.4-5.6.

$$Tq = \frac{v \cdot g \cdot m_{distr} \cdot L_{wb}}{2 \cdot L_{wb}} \tag{5.4}$$

$$\tau_{slip,l,max} = \left(Tq + \frac{a_x \cdot m \cdot CoG_z}{2 \cdot L_{wb}} - \frac{a_y \cdot m \cdot CoG_z}{2 \cdot L_{tw}}\right) \cdot r_t \cdot f_t \tag{5.5}$$

$$\tau_{slip,r,max} = \left(Tq + \frac{a_x \cdot m \cdot CoG_z}{2L_{wb}} + \frac{a_y \cdot m \cdot CoG_z}{2 \cdot L_{tw}}\right) \cdot r_t \cdot f_t \tag{5.6}$$

Where v is the vehicle velocity, m_{distr} is the mass distributed to the front axle and L_{wb} is the wheel base. L_{tw} is the track width, r_t and f_t are tire radius and friction respectively and a_x and a_y the longitudinal and lateral accelerations.

The desired torques on the left and right clutch have two dynamic coefficients, the total torque provided by the engine and the control signal. If the control signal is at zero the torque will be evenly divided to the two clutches. If the control signal differs from zero it is added to and subtracted from the left and right torques respectively. The value is shown in Equation 5.7.

$$\Delta = I \cdot \frac{r_t}{L_{tw}} \cdot u \tag{5.7}$$

The difference Δ is saturated against half of the total torque to ensure that the distributed torque is always positive and does not exceed the total torque.

Thus, the distributed torque on the two axes can be described as:

$$\tau_l = \frac{\tau_{tot}}{2} - \Delta \tag{5.8}$$

$$\tau_r = \frac{\tau_l}{2} + \Delta \tag{5.9}$$

Where τ_l and τ_r are the distributed torques on the left and right wheels, and τ_{tot} is the total torque provided by the engine. The distributed torque is finally saturated against the slip torque calculated in Equation 5.5 and 5.6.

5.4.3 Vehicle Reference

Theoretical Simulink Control Model

The vehicle reference block in the Simulink model, shown in Figure 5.10, collects the inputs from the vehicle model and distributes them to the controller and torque allocation-blocks.

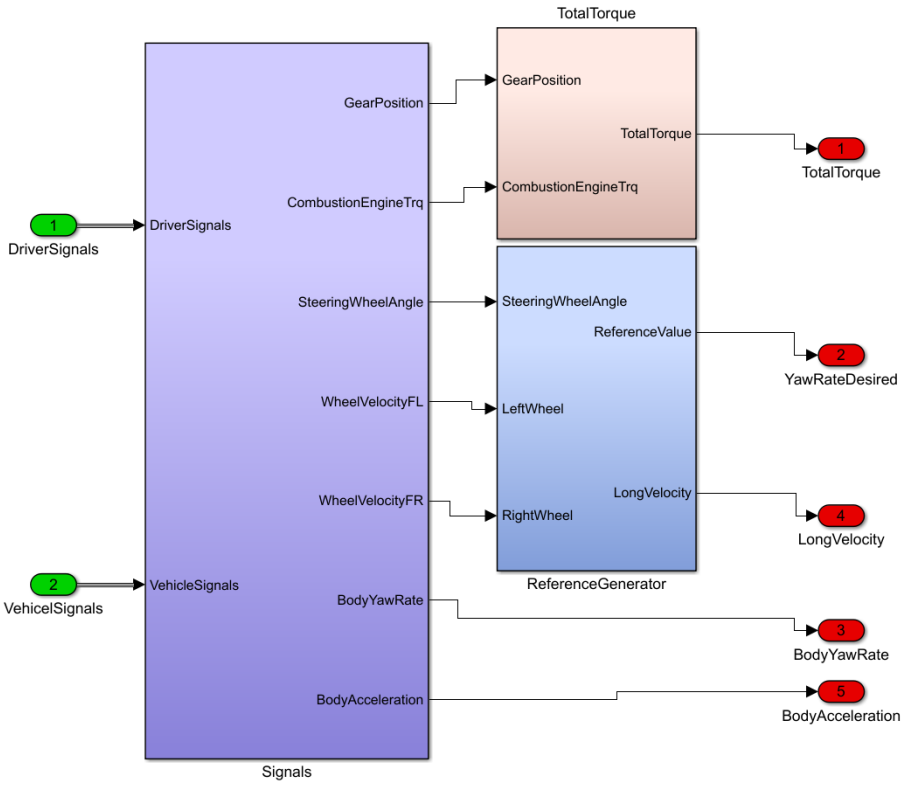


Figure 5.10: The theoretical Simulink vehicle reference model.

Practical Control Software

The vehicle reference part of the control algorithm acts as a reference to the different inputs from the vehicle. The reference inputs generated are body acceleration, body yaw rate, vehicle velocity, desired yaw rate, and the total torque. All of these references are quantified as doubles. The body yaw rate and the body acceleration is expected to come from direct sensors connected to the electronic control unit.

Vehicle velocity is calculated based on the input from the wheel velocities from the FS CAN-bus. The body velocity is set as the average of the two front wheels, but the averaging method (two-wheel front, two-wheel rear or four-wheel) is changeable in the settings file.

The yaw rate desired is based on two inputs; the vehicle velocity and the steering wheel angle. The steering wheel angle δ_{st} is initially converted to wheel angle δ_w by dividing with the vehicle's conversion rate CR, i.e. the relationship between the steering wheel angle and the tire angles. The yaw rate desired is then calculated with Equation 5.10:

$$\delta_w = \frac{\delta_{st}}{CR} \tag{5.10}$$

$$\psi_{des} = \frac{v \cdot \delta_w \cdot 2 \cdot C_{\alpha,r} \cdot C_{\alpha,f} \cdot (l_r + l_f) + (l_r + l_f)}{\sqrt{v} \cdot m \cdot (l_r \cdot C_{\alpha,r} - l_f \cdot C_{\alpha,f})} \tag{5.11}$$

Where $C_{\alpha,f}$ and $C_{\alpha,r}$ are the lateral tire stiffness for the front and rear wheels, and l_f and l_r are the distances from the COG to the front and rear axles respectively. A driver parameter A is added in order to allow customization of the TVDC in regards to over- and understeering. One driver may prefer the vehicle to oversteer at all times and another driver may prefer the vehicle to understeer. As such, the yaw rate desired is set to vary depending on the driver input and is adjusted according to Equation 5.12.

$$\psi_{des} = \psi_{des} \cdot A \tag{5.12}$$

Depending on the speed and the tire friction, the tires may slip before the desired yaw rate is achieved. Consequently, the tire slip upper and lower limits in Equations 5.13 and 5.14 are used to saturate the desired yaw rate.

$$\overline{\lim} = \frac{g \cdot f_t \cdot 0.85}{v} \tag{5.13}$$

$$\underline{\lim} = \frac{-g \cdot f_t \cdot 0.85}{v} \tag{5.14}$$

Total Torque or Torque Left and Torque Right

The total torque was first implemented as a combination of the combustion engine torque and the gear ratio depending on the gear position. This choice was made to make the total torque source flexible and can be determined by the FS-team as they see fit. However, if the total torque calculation is poor then the control algorithm is poor.

Consequently, it is advisable to use torque sensors on the axles as it provides an immediate and accurate torque result. The NCTE - FS Driveshaft Torque Sensor, for example, is an affordable and easy-to-mount sensor for FS-cars that uses the driveshaft's magnetic field to determine the torque. It allows for a higher degree of torque vs. slip control, as well as a better control algorithm in general.

5.4.4 Manual Settings

As the control algorithm depends on several different vehicle parameters and driver preferences, it is important to have a simple tuning interface. As such a settings-file was constructed which allows the tuning parameters to be changed without going into the critical and sensitive control algorithm. The settings-file initializes the different parameters as public. The three different control modules then reference the file and collect the parameters as constant values.

5.5 The TVECU Final Product

The final assembled product, shown in Figure 5.11, weighs 68 g, of which only 10 g are electronics. Both the TVECU and the TVDC CAN cables are waterproof reinforced, and contain invisible termination bits at each actuator end. The product enables the continued development and use of the TVDC+TVECU system.

Figure 5.11: The final assembled TVECU, with connections to FS CAN, TVDC CAN and simulated TVDC sensors.

5.5.1 Ingress Protection

A complete ingress test was conducted, where the TVECU laid underwater at 1 m for 30 minutes. Although a formal Ingress Protection certification test was not conducted, the test showed an equivalent certification level of IPx7. Although no dust tests were conducted, it is likely the TVECU provides a dust-tight environment that gives an equivalent certification level of IP67.

5.5.2 Actuator Control

The setup, shown in 5.12, represents the implementation in the TVDC and is similar to the setup in *6.2.2 Control Algorithm Tests*. The FS-car, simulated using the development system on the left, sends parameters to the TVECU. In response, the TVECU sends commands to the two Gen6 actuators (bottom). The development system on the right simulates the TVDC sensors.

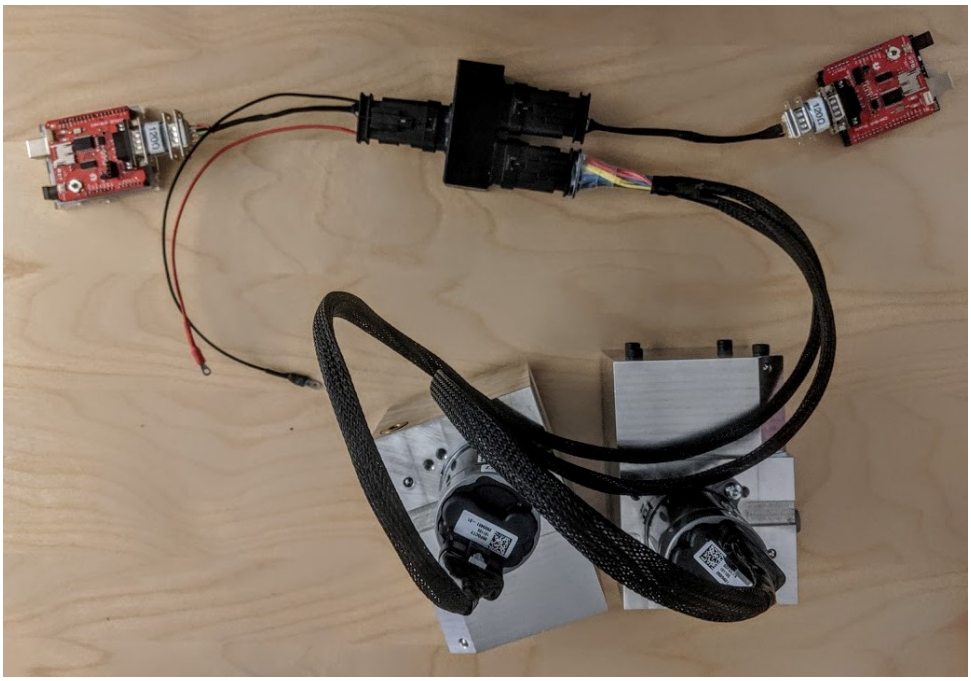


Figure 5.12: The final assembled TVECU connected to two Gen6 actuators, and two systems simulating the FS-car and the TVDC sensors.

Chapter 6

Testing and Refinement

This chapter describes the various tests done in parallel with the development of the prototypes and final product. It outlines the verification of the practical control algorithm and then depicts the TVECU system tests conducted to verify communication, control and sensors.

6.1 Simulink Model Code Test

The first tests conducted were to ensure that the control code written in C was functioning the same as the control model created in Simulink. A Simulink block was created from the practical control algorithm. The block consisted of the same parameters as the controller block and was connected to the vehicle model. The block system was run in parallel with the original theoretical Simulink model, as shown in Figure 6.1.

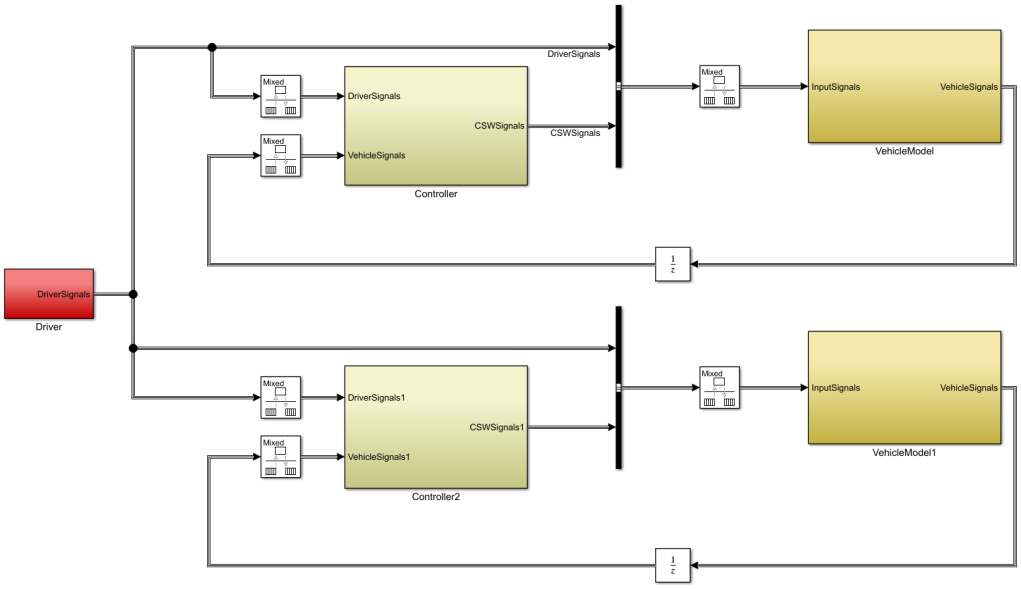
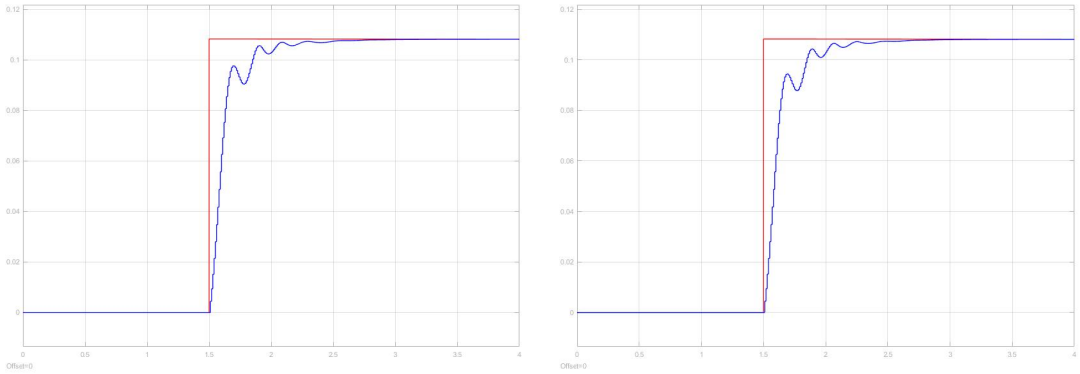


Figure 6.1: The theoretical model (top) and control algorithm (bottom) in parallel.

The Simulink function "coder.ceval" was used to insert and execute the external C/C++ function. A Simulink scope block, a block that displays time-domain signals, was used to visualize the results. The results were first analyzed to find any difference between the two systems and further examined to find any flaws in the control sequence, and what possible measures could be taken to adjust those flaws.

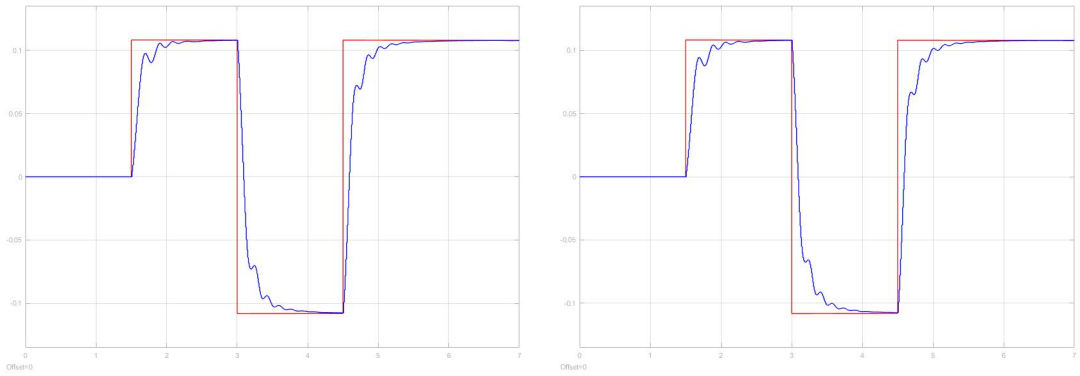
6.1.1 Verification of Control Algorithm Behavior

As shown in Figures 6.2-6.4, the yaw rate response behaves the same in the control algorithm as in the Simulink model, given the same input parameters. This leads to the conclusion that the control algorithm correctly represents the Simulink model and can be used to implement the control model onto the ECU.



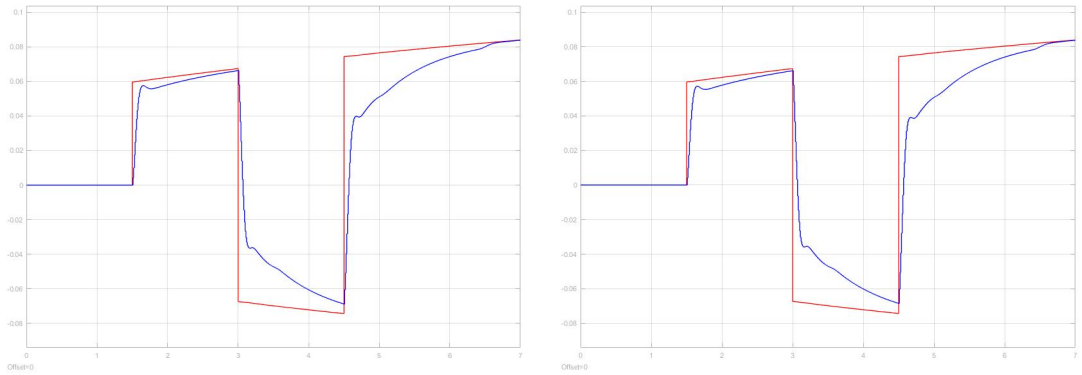
(a) Simulink Model (b) Control Algorithm

Figure 6.2: Angular velocity [rad/s] vs time [s] after an 18° steering wheel angle, 20 m/s initial velocity and a total torque of 60 Nm. In red, the desired yaw rate. In blue, the body yaw rate.



(a) Simulink Model (b) Control Algorithm

Figure 6.3: Angular velocity [rad/s] vs time [s] after three 18° steering wheel angles, 20 m/s initial velocity and a total torque of 60 Nm. In red, the desired yaw rate. In blue, the body yaw rate.



(a) Simulink Model (b) Control Algorithm

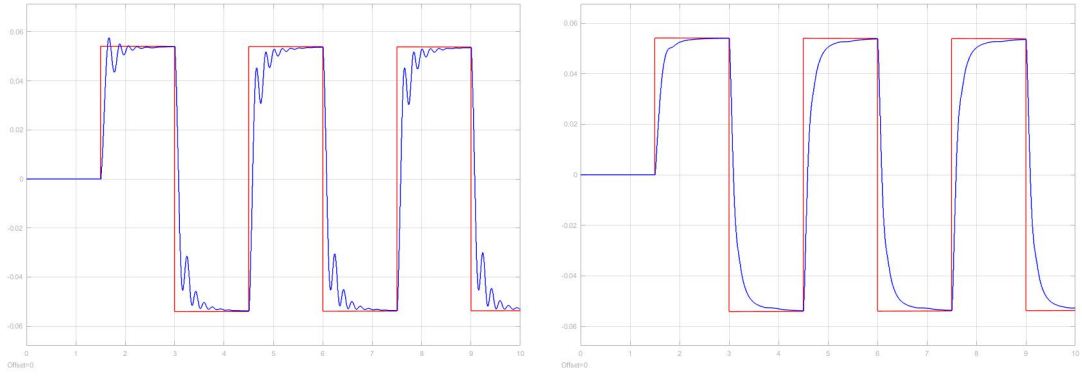
Figure 6.4: Angular velocity [rad/s] vs time [s] after three 18° steering wheel angles, 5 m/s initial velocity and a total torque of 60 Nm. In red, the desired yaw rate. In blue, the body yaw rate.

6.1.2 Verification Test Evaluation

Although the control algorithm adequately represents the theoretical model, the test discovered oscillations that occurred at speeds higher than 15 m/s regardless of how small the turn was.

In the model, a steering wheel angle at 0° will be applied as exactly 0°, whereas the driver's steering wheel angle won't. There will always be a small oscillating steering wheel angle as a result of the human factor, uneven track or stones, among others. The TVDC can be disabled at very high speeds, but the elimination of this problem was crucial for the product to function as intended during mid-range velocities.

This prompted the development of the derivative term. The derivative term dampens the aggressive behavior of the proportional and integral terms, i.e. dampens the oscillations. Since the oscillations occur at high speeds the derivative gain is set to 0 at velocities below 15 m/s. The difference is shown in Figure 6.5.

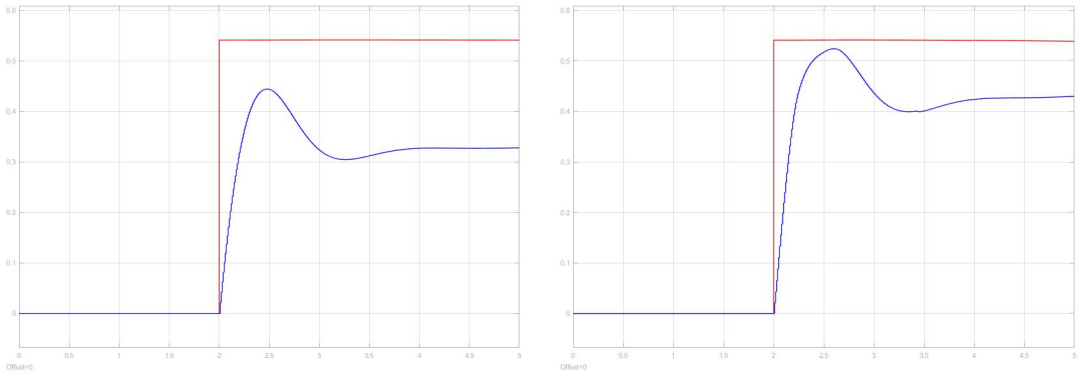


(a) Simulink Model (b) Control Algorithm

Figure 6.5: Angular velocity [rad/s] vs time [s] after six 9° steering wheel angles, 20 m/s initial velocity and a total torque of 60 Nm. In red, the desired yaw rate. In blue, the body yaw rate.

6.1.3 Active Torque Distribution Evaluation

As shown in Figure 6.6, the active torque distribution implemented has a clear positive effect on the yaw rate. The yaw rate response in the relatively extreme scenario is not optimal in either model, however, the response in the system with the active torque distribution performs more closely to the optimal value.



(a) Simulink Model (b) Control Algorithm

Figure 6.6: Angular velocity [rad/s] vs time [s] after a 90° steering wheel angle, 20 m/s initial velocity and a total torque of 60 Nm. In red, the desired yaw rate. In blue, the body yaw rate.

6.2 TVECU System Tests

6.2.1 Communication Loop-Back Test

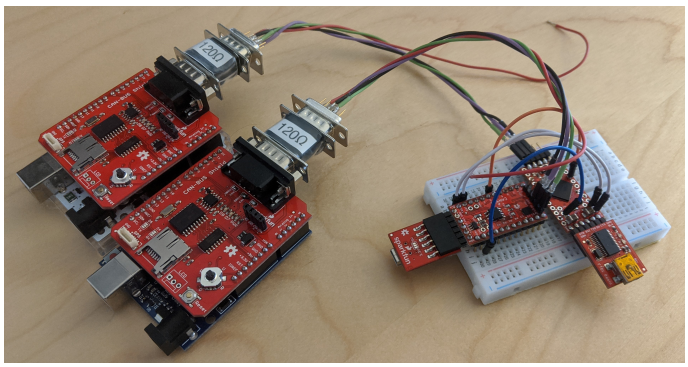


Figure 6.7: Development system (left) connected to TVECU beta-prototype (right) during a loop-back test.

The first test of the TVECU system was conducted by generating CAN messages, modified by a joystick, via a development system. The data was received by the SensorPCB, sent via serial to ControlPCB and then sent via CAN to a second development system. The sample time of the TVECU loop including the control algorithm was also measured. During 1000 iterations the sample time varied evenly between 11-13 ms, about 80 times per second. For perspective, a regular Anti-lock Braking System (ABS) runs at around 15 times per second [36].

The system worked without fault during the loop-back test. An important milestone in the development of both the communication and the processing of the TVECU system. It proved the serial variable refresh request method works sufficiently fast and that interrupts may just introduce more potential issues at a low reward.

6.2.2 Control Algorithm Tests

Manual Error Control Manipulation

The manual error control manipulation test involved recreating the previous setup from the loop-back test and adding a serial communication line between a computer and the sending development system. The error value in the control algorithm was then externally modified and the response measured through the receiving development system. Upon the creation of an error, the response would quickly grow towards infinity. This is expected, as the control algorithm cannot close an error it cannot affect.

Gen6 Actuator Pump Control Test

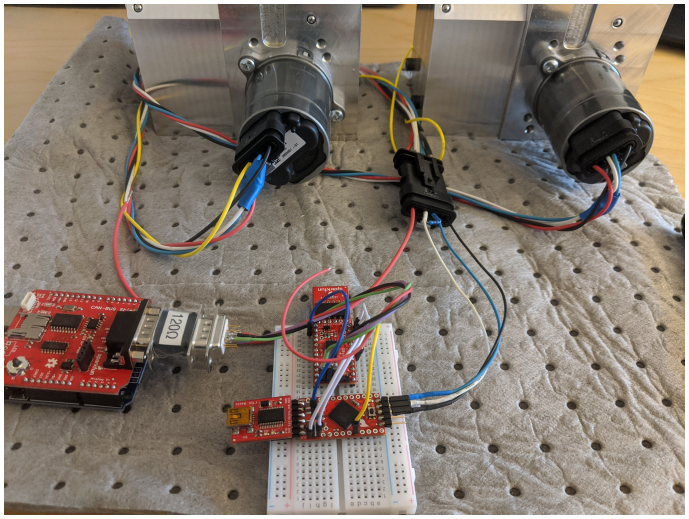


Figure 6.8: TVECU beta-prototype connected to 2 Gen6 actuators for a pump test.

To visualize the error response as well as test the control of the Gen6 actuators, the test was repeated with two oil-filled Gen6 actuators instead of the receiving development system. Upon the introduction of an error, the "inner wheel pump" engaged and attempted to close the error, as expected from the previous tests. The communication with the Gen6 actuators functioned properly, and the test provided a more concrete visual and audible feedback of the control algorithm.

6.2.3 Sensors Test with Volvo XC90

To test the sensors in the TVECU, it was connected to the CAN-bus of a Volvo XC90. The purpose was to ensure proper CAN function and that the Volvo messages could be interpreted and handled as parameters in the TVECU. The CAN IDs of the XC90 were known.

CAN Message Receiver

A splice into the XC90 CAN-bus allowed access to the CAN messages and the raw CAN data could be read. Using the same CAN interpreting algorithm written for the TVECU made the values that were transmitted on CAN bus readable. The loop time for the messages ranged between 11-13 ms which is the same as the previous sample time. This further supports that the TVECU will be able to receive and process CAN messages, as well as run its control algorithm at an adequate speed while transmitting its own CAN messages.

Accelerometer and Gyroscope

The gyroscope and accelerometer in the XC90 were also captured via CAN in order to compare with the results yielded from the TVECU sensors. A source of error is the fact that the TVECU sensor was not completely locked into place during the test, but held as firmly as possible. Nevertheless, it can be concluded that the values of the acceleration and yaw rate correlate with the XC90 sensors and that the differences between the TVECU sensor values and the XC90 sensor values are negligible.

Chapter 7

Evaluation and Conclusion

This chapter evaluates the TVECU against the target specifications and identifies design elements not yet implemented. It reviews potential issues moving forward and presents items for future development such as CAD and PCB design, before concluding the thesis.

7.1 Evaluation of TVECU

The final specifications are presented and compared to the target specifications in Table 7.1. All in all, the TVECU scores optimal in 15/23 needs, acceptable in 8/23 needs and failed zero needs. Certain results marked with a * are partial results.

Partial Result Metrics

Despite being unlikely to ever damage the TVDC, the possibility cannot and has not been precluded. As such, no TVDC damage in metric 4 cannot be ascertained. There could exist certain cases where the TVECU control runs haywire due to TVDC issues since investigations into possible hazards were outside the scope of the thesis.

Similar reasoning is applicable to failure handles in metric 9. Although the TVDC defaults to closed when an issue arises, neither investigations were conducted nor fail-safes implemented for when, for example, the TVDC breaks. However, the Gen6 actuators need continuous contact with the TVECU to run or they will close the TVDC, in case of TVDC CAN failure.

The Simulink compatibility was stricken from the target specifications after the deployment alternatives in *4.2.1 Software Deployment Methods*. Finally, method and compatibility work has been completed for the locked wheels special case in metric 10 and local data log in metric 13, both described in *7.1.1 Trivial and Potential Issues in Further Development*.

Table 7.1: List of target metrics as developed from the product needs and the result. Partial results are denoted with a *.

№	Metric	Imp.	N.№	Optimal	Acceptable	Result
1	Simple FS CAN	3	1	Y	Y	Y
2	Robust TVDC CAN					
	Robust CAN-bus	3	2	Y	N	Y
	Shock protection	3	9	Y	Y	Y
	Local sensors priority	2	15	Y	Y	Y
3	High Speed	3	4	>50 Hz	>10 Hz	80 Hz
4	Safety Handles					
	No TVDC damage	3	8	Y	N	N*
	Shock protection	3	9	Y	Y	Y
5	Operating Temp.	3	10	-20 − 80°C	-5 − 60°C	-40 − 80°C
6	Fast Startup					
	Fast Startup Time	3	11	<1 s	<3 s	<1 s
	Reboot Watchdog	1	20	Y	N	N
7	Simple Configuration					
	Easy to reprogram	2	12	Y	N	Y
	~~*Simulink compatible*~~	~~2~~	~~14~~	~~Y~~	~~N~~	~~N*~~
~~8~~	~~Small Control Error~~	~~3~~	~~3~~	~~± 5%~~	~~± 15%~~	~~??~~
9	Failure Handles					
	TVDC failure	2	16	Y	N	N*
	Electrical failure	2	17	Y	N	N*
10	Special Case Handles					
	Wind-up	3	5	Y	Y	Y
	Saturation errors	3	6	Y	Y	Y
	Locked wheels	3	7	Y	N	N*
	Over-steering errors	2	13	Y	N	N
11	Low weight	1	18	<25 g	<100 g	68 g
12	Efficient Packaging					
	Efficient size	1	19	Y	N	Y
	Weather resistant	1	21	Y	N	Y
13	Data Logging					
	Local data log	2	16	Y	N	Y*
	Accessible data log	2	17	Y	N	Y
14	Lifespan	1	22	>10 yrs	>2 yrs	>2 yrs

Evaluation Results

A notable decision was the removal of the "Small Control Error" (metric 8) from the evaluation table. Since there was no TVDC or FS-car to test on, it was difficult to quantify the control error. Nevertheless, it can be established that there is considerable room for optimization of the control algorithm parameters.

Although no reboot watchdog was implemented, the TVECU auto-initializes upon reboot and can recover from a power failure or engine restart. It is simple to configure via a settings-file and has a modular software to simplify changes in the code. The

TVECU electronics weigh 10 g and are comprised of components with a long lifespan, despite safeguarding the durability to more than 2 years.

Most importantly, the TVECU runs at a significantly higher speed than in the target specifications. It has a fast start-up time, can operate in most temperature and weather conditions, and is robust despite its modular software. In conclusion, the TVECU fulfills the required specifications set out in the early stages of development.

7.1.1 Trivial and Potential Issues in Further Development

Trivial Issues and Development

Vibration dampeners, screws and the lid and TVDC fasteners on the TVECU box also remain to be developed in cooperation with the TVDC engineers. Although a design suggestion is presented in Figure 7.1, it cannot be decided until a design freeze on the TVDC ensures proper placement and ample measurements.

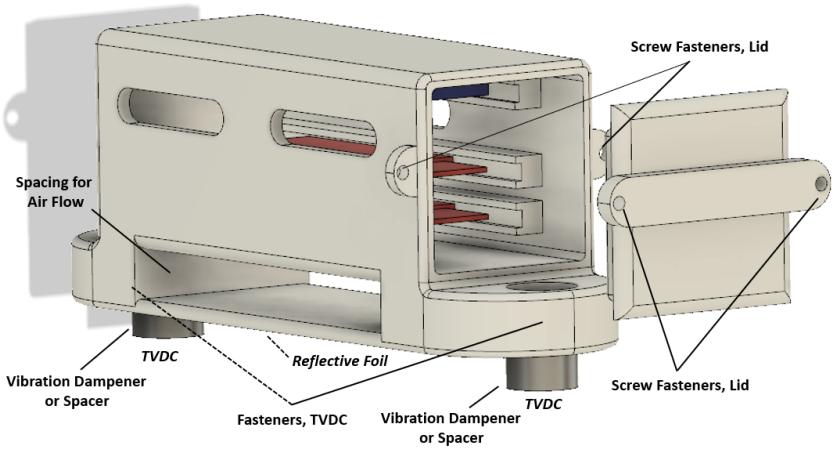


Figure 7.1: An example solution of the lid and TVDC fastening method.

The ControlPCB code should be converted to a concurrent program to ensure a stable sample time, and the locked wheels special case is easily solved by disabling the TVDC when braking hard. Furthermore, the CAN message ID and format remain to be specified bilaterally with Lund Formula Student. As such, the driver parameter reception, as well as the transmission of the error byte to the driver via FS CAN, are not implemented. Finally, what parameters Lund Formula Student want to be logged is not determined and remains to be implemented.

Potential Issues and Problems

A potential problem with the TVECU box is the durability of the fastening method between the Gen6 connectors and the TVECU box. The Gen6 connector attachment point may need to be redesigned and reinforced for both further waterproofing and durability.

Another potential problem is the gyroscope's placement on the TVDC and not on the car's COG. Although the tests in *6.2.3 Sensors Test with Volvo XC90* did not indicate an issue, it needs to be documented and kept in mind during FS-car tests.

Finally, the electromagnetic interference (EMI) from the Gen6 power cable cannot be ruled out. Further testing should be conducted "on the bench" and in an FS-car to check for noise and function issues.

7.1.2 Future Development

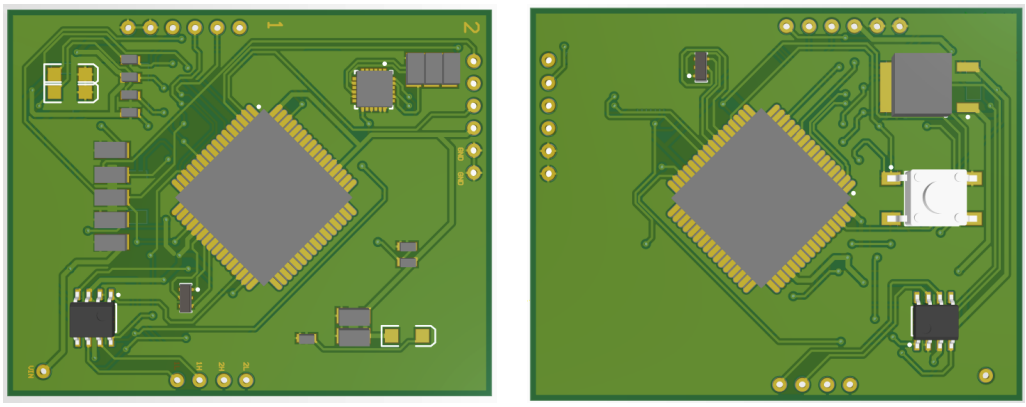
The TVECU has certain special-case handling in the control algorithm, but further investigation into a potential over-steer error is required. The hypothetical error can arise when the car over-steers despite a fully open inner clutch. This can be solved by slipping the outer-wheel when over-steering with maximally clutched inner wheel.

Furthermore, the testing, fine-tuning and implementation of the TVECU+TVDC system must be completed on an FS-car. Since the proper function of the complete system cannot be ascertained without an FS-car, the implementation and testing might reveal minor or major flaws in the system. Even if the system works as expected, the parameters must be fine-tuned on each FS-car as well as a great deal of track testing to ensure optimum performance.

Next Generation TVECU - PCB Design

Given the proof-of-concept of the TVECU, it is also possible to design one PCB with all components required. Creating a TVECU PCB would increase robustness and safety whilst minimizing size and weight. It also allows for complete freedom when deciding specifications of components and functions of modules.

A demo TVECU PCB schematic for the TVECU is shown in Appendix B and a design is shown in Figure 7.2. The design contains two AT90CAN128 micro-controllers, the same as in the AST-CAN485, two CAN transceivers, an accelerometer, a gyroscope, and a voltage regulator. Although the demo TVECU PCB contains everything required for the development and replacement of the current TVECU, it should serve as an example and proof of concept only.



(a) Front (b) Back

Figure 7.2: The design of the demo TVECU PCB.

7.2 Conclusion

The final TVECU is designed to be used on the first iteration of the Torque Vectoring Dual Clutch and has reached the "desired ambition" level according to *1.2.1 Tiered Ambition Level*. Despite the fact that the TVDC was not produced on time, the development of the TVECU went beyond its initial objective by including TVDC sensors, system modularity as well as data logging and error management.

Although the implementation work remains to be done once the TVDC is produced, the primary development of the TVECU is considered completed. The groundwork for the next generation TVECU has also been laid, enabling production development when the system is ready. All in all, the torque vectoring ECU is ready to be the stepping stone into Lund Formula Students torque vectoring era, allowing the team to push limits in both electric and combustion vehicle classes.

Figure 7.3: Racing into the future [37].

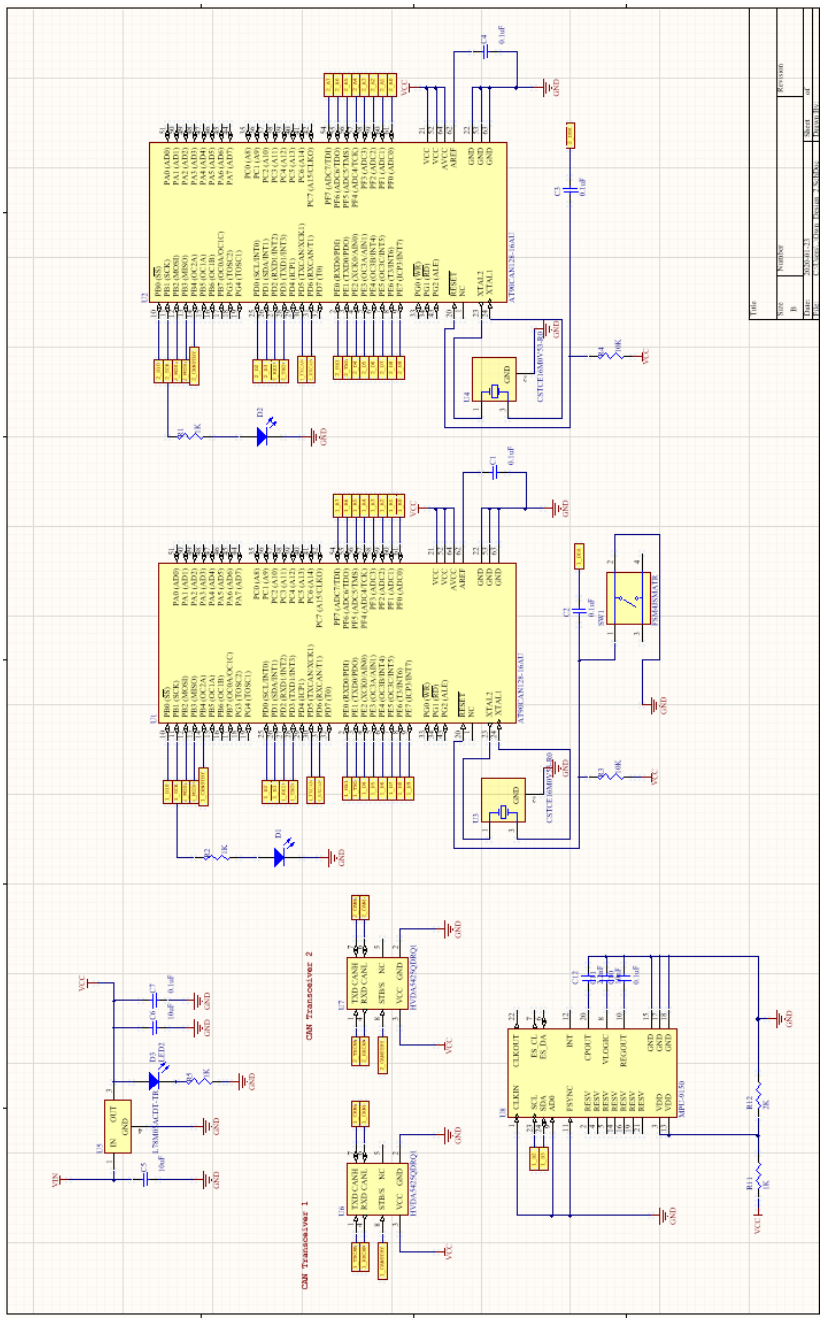
Appendix A

The "Risk Matrix - Technical" used to identify risks which resulted in the system decision of keeping the two CAN separate and including an accelerometer and gyroscope.

Item	Probability	Consequence	Resulting Risk	Measure
FS signals interfere with TVDC signals on CAN-Bus	High	High	High - Eliminate	Use a second private CAN-bus for TVDC control.
Accel./gyro. missing or not functioning on FS-car	High	High	High - Eliminate	Incorporate accel./gyro. in system.
TVDC late/not produced	High	Low	Medium - Minimize	System designed/thesis delimited to not require TVDC mechanical impl.
FS-car not available for final tests	High	Low	Medium - Minimize	System designed/thesis delimited to not require FS-car tests.
Time in bench	High	Low	Medium - Minimize	System designed/thesis delimited to not require bench tests.
One processor not enough	High	Low	Medium - Minimize	Order extra hardware/alternative hardware ready to order.
Memory not enough	Low	High	Medium - Minimize	Order extra hardware/alternative hardware ready to order.
Processor/system not fast enough	Low	High	Medium - Minimize	Order extra hardware/alternative hardware ready to order.
Simulink model only functions in theory	Low	Low	Low - Observe	Observe. Control theory concept outside scope of thesis.
Ordered hardware delayed	Low	Low	Low - Observe	Order hardware early.
Time in BorgWarner car	Low	Low	Low - Observe	System designed/thesis delimited to not require BorgWarner car tests.

Appendix B

The demo TVECU PCB schematic used in the design and manufacture of the demo TVECU PCB.



Bibliography

[1] Lund Formula Student. Last day of testing before we head to fs netherlands!, 2019. URL http://lundformulastudent.se/. Available 2019-12-13.

[2] Sven Kalkan. *A final drive device for torque vectoring in small race cars*. Master thesis, Lund University, Unpublished.

[3] Mia Grahovic and Madeleine Rosicki. *Development and Evaluation of a Torque-Vectoring Algorithm on RWD Racing Cars using a Dual Clutch*. Master thesis, Department of Automatic Control, TFRT-6081, Lund University, 2019.

[4] Steven D. Eppinger Karl T. Ulrich. *Product Design And Development*. McGraw-Hill, New York, NY, 5 edition, 2012.

[5] What When How. Differential (automobile), Publication Date Unknown. URL http://what-when-how.com/automobile/differential-automobile/. Available 2019-12-13.

[6] BMW. The Active M Differential, 2014. URL https://www.bmw-m.com/en/topics/magazine-article-pool/the-active-m-differential.html. Available 2019-12-13.

[7] Belisso. Porsche torque vectoring, 2009. URL http://torque-vectoring.belisso.com/. Available 2019-12-13.

[8] Reasearch Gate. Sae vehicle axis system, 2015. URL https://www.researchgate.net/figure/1-SAE-Vehicle-Axis-System-2_fig94_313475487. Available 2019-12-13.

[9] Michael Sayers. Standard terminology for vehicle dynamics simulations, 1996. URL http://hosting.umons.ac.be/html/mecara/grasmech/standardterminologyforvehicledynamicssimulation.pdf. Available 2019-12-13.

[10] R. Rajamani. *Vehicle Dynamics and Control*. Springer Science & Business Media, New York, USA, 2011.

[11] Adam Kader. Steer-by-wire control system, 2006. URL https://www.swarthmore.edu/sites/default/files/assets/documents/engineering/AK_Final.pdf. Available 2020-01-23.

[12] D.Y. Batyshchev V.A. Ryzhikov. Differential braking device, 2017. URL https://www.sciencedirect.com/science/article/pii/S187770581735364X. Available 2020-01-23.

[13] Mazda. Mazda cx-5 service & repair manual: Dsc control, 2016. URL http://www.mcx5.org/dsc_control-875.html. Available 2019-12-16.

[14] Karl-Erik Årzén. *Real-Time Control Systems*. Lund University, Lund, 2014.

[15] Lewis Kingston. What is an electronic control unit?, 2018. URL https://www.pistonheads.com/features/ph-features/what-is-an-electronic-control-unit-ph-explains/37771. Available 2019-12-16.

[16] ECU Testing. Ecu explained, 2020. URL https://www.ecutesting.com/categories/ecu-explained/. Available 2020-01-13.

[17] Altium. What is a printed circuit board?, 2019. URL https://www.altium.com/solution/what-is-a-pcb/. Available 2019-12-16.

[18] Embedded Artistry. Printed circuit board (pcb), 2019. URL https://embeddedartistry.com/fieldmanual-terms/printed-circuit-board/. Available 2019-12-16.

[19] Margaret Rouse. Definition of microcontroller, 2012. URL https://internetofthingsagenda.techtarget.com/definition/microcontroller. Available 2019-12-16.

[20] CSS Electronics. Can bus explained - a simple intro, 2019. URL https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/language/en. Available 2019-12-10.

[21] Bosch. Can specification, 1991. URL http://esd.cs.ucr.edu/webres/can20.pdf. Available 2019-12-05.

[22] Steve Corrigan. Introduction to the controller area network (can), 2016. URL http://www.ti.com/lit/an/sloa101b/sloa101b.pdf. Available 2019-12-10.

[23] Wolfsburg VOLKSWAGEN AG. Data exchange on the can bus i, Publication Date Unknown. URL http://www.volkspage.net/technik/ssp/ssp/SSP_238.pdf. Available 2019-12-05.

[24] Kvaser. Can messages, 2019. URL https://www.kvaser.com/about-can/the-can-protocol/can-messages-23/. Available 2019-12-06.

[25] DEWEsoft. Automotive buses - can measurement, 2019. URL https://training.dewesoft.com/online/course/automotive-buses-can-measurement. Available 2019-12-10.

[26] National Instruments. Controller area network (can) overview, 2019. URL https://www.ni.com/sv-se/innovations/white-papers/06/controller-area-network--can--overview.html. Available 2019-12-05.

[27] Staffan Nilsson. Controller area network - can information, 1997. URL http://hem.bredband.net/stafni/developer/CAN.htm. Available 2019-12-05.

[28] Kvaser. Can bus error handling, 2019. URL https://www.kvaser.com/about-can/the-can-protocol/can-error-handling/. Available 2019-12-05.

[29] Kvaser. Can physical layers, 2019. URL https://www.kvaser.com/about-can/the-can-protocol/can-physical-layers/. Available 2019-12-05.

[30] CSS Electronics. Terminal resistor (120 ohm, db9, can bus), 2019. URL https://www.csselectronics.com/screen/product/terminal-resistor-can-bus/language/en. Available 2019-12-17.

[31] Perforce. What is iso 26262?, 2019. URL https://www.perforce.com/blog/qac/what-iso-26262-overview. Available 2019-12-13.

[32] Bosch. Electronic engine control unit for commerical vehicles, 2019. URL https://www.bosch-mobility-solutions.com/en/products-and-services/commercial-vehicles/powertrain-systems/natural-gas/electronic-engine-control-unit/. Available 2019-12-13.

[33] Sparkfun. Sparkfun, 2019. URL https://www.sparkfun.com/. Available 2019-12-19.

[34] Mathworks. Embedded coder, 2019. URL https://www.mathworks.com/products/embedded-coder.html. Available 2019-12-18.

[35] Microchip Technology. Atmel studio 7, 2019. URL https://www.microchip.com/mplab/avr-support/atmel-studio-7. Available 2019-12-18.

[36] KARIM NICE. How anti-lock brakes work, 2019. URL https://auto.howstuffworks.com/auto-parts/brakes/brake-types/anti-lock-brake1.htm. Available 2020-01-13.

[37] Formula Student Germany. Photos hockenheim, 2019. URL https://media.formulastudent.de/2019/Hockenheim. Available 2020-01-23.