CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Engine synchronous sensor processing in the transmission control unit

Master's thesis in Embedded Electronic System Design

APARNA RAM SURESH SARITHA KUMARI
ELIZABETH SWATHIKA AZARIAH

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2022

Master's thesis 2022

Engine synchronous sensor processing in the transmission control unit

Aparna Ram Suresh Saritha Kumari
Elizabeth Swathika Azariah

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2022

Engine synchronous sensor processing in the transmission control unit

Aparna Ram Suresh Saritha Kumari
Elizabeth Swathika Azariah

Supervisor: Lena Peterson, CSE
Advisors: Ayron Catteau & Andrei Kovács, Volvo GTT
Examiner: Per Larsson-Edefors, CSE

Master's Thesis 2022
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2022

Engine synchronous sensor processing in the transmission control unit
Aparna Ram Suresh Saritha Kumari
Elizabeth Swathika Azariah
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

The electronic control units (ECUs) of the modern-day vehicle have evolved significantly over the years, with the increasing range of sensors used. Some of the sensors in the transmission ECU of a conventional diesel truck, such as the inclination sensor and the clutch position sensor, are affected by the vibrations caused by engine rotation, which can result in noisy sensor measurements. The purpose of this master thesis project is to collect sensor information synchronous to the engine rotation in the transmission ECU, in order to filter out the engine vibration from the raw sensor measurements, resulting in better vehicle control. We evaluate the possibility of engine-synchronous sensor acquisition by developing a proof of concept method to communicate the engine-synchronous signal from the engine ECU to the transmission ECU. The two communication methods identified and explored are, first, the dedicated direct-line communication method and, second, the controller area network (CAN) communication method which were implemented in a test-rig setup. An engine-synchronous signal transmitted using the direct-line communication was then used to schedule the filtering software at the transmission control unit for an inclination sensor to filter the vibration noise from the raw inclination sensor values using a simple moving average filter. The developed the synchronous filtering was also implemented and tested on a moving truck and the results were analysed. We were able to transmit a signal engine-synchronously from the engine ECU to the transmission ECU by using the direct-line communication method. This signal was used to engine-synchronously filter the sensor noise using a simple moving average filter in the transmission control unit.

Acknowledgements

We would like to thank the following people, without whom the completion of this thesis work would not be possible.

First and foremost, we thank **Volvo Group Trucks Technology** and **Johan Fries** for providing us the opportunity to carry out this thesis project.

We want to thank **Lena Peterson**, our supervisor at Chalmers for all her support and guidance throughout this thesis work.

We wish to express our gratitude to our supervisors **Andrei Kovács** and **Ayron Catteau** at Volvo GTT for mentoring and guiding us in all the technical aspects as well as in report writing, from start till end.

We also specially thank **Therese Mäsak** (for all her help with the truck testing), **Sara Gothäll** (for all her help with the CAN communication), **Hans Bernler**, **Simon Lillskog** and **Igor Lumpus** at Volvo GTT for all their help and support in various aspects of this thesis work.

We learned a lot from all of you!

Finally, we would like to thank our **friends** and **families** for all their motivation and moral support.

Aparna Ram Suresh Sarita Kumari
Elizabeth Swathika Azariah
Gothenburg, July 2022

Abbreviations

Denotation	Declaration
ALU	Arithmetic Logic Unit
ARU	Advanced Router Unit
CAN	Controller Area Network
CCP	CAN Calibration Protocol
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DMA	Direct Memory Access
ECU	Electronic Control Unit
EOF	End-of-Frame
ECM	Engine Control Module
EECU	Engine Electronic Control Unit
EMS	Engine Management System
GTM	Generic Timer Module
ICU	Interrupt Control Unit
ID	Identity
IP	Intellectual Protocol
IR	Interrupt Router
I/O	Input/Output
LIN	Local Interconnect Network
LPF	Low Pass Filter
MISO	Master-In/Slave-Out
MOSI	Master-Out/Slave-In
MOST	Media Oriented Systems Transport
PCB	Printed Circuit Board
SPI	Serial Peripheral Interface
QSPI	Queued Synchronous Peripheral Interface
SOF	Start Of Frame
SRN	Service Request Node
TECU	Transmission Electronic Control Unit
TIM	Timer Input Module
USB	Universal Serial Bus
VR	Variable Reluctance

Contents

1

Introduction

The modern-day motor vehicle is not just made of mechanical components, but also contains a diverse range of electronic components to control the vehicle. The electronic control unit (ECU) is one of the most advanced embedded systems designed to control the electrical systems in a modern vehicle [1] . Many parts in a conventional truck such as the engine and transmission have an ECU connected to them. The embedded software in the ECUs has undergone many design upgrades over the years, which has made it possible for them to handle complex controls within the vehicle.

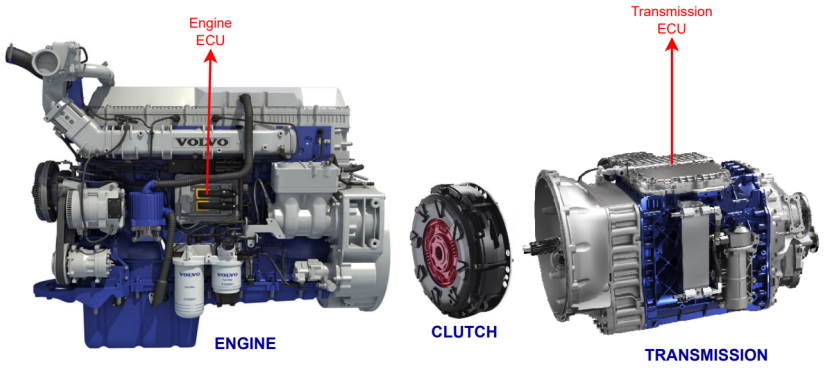

Figure 1.1: The engine and transmission of a conventional Volvo diesel truck

The driveline of a conventional diesel truck includes a diesel engine rigidly connected to a transmission through the clutch as shown in Figure 1.1. Both the engine and the transmission have a wide range of sensors and actuators, and an ECU which communicates with and processes data from the sensors [2]. The combustion and rotation of the engine is a significant source of vibration that can affect sensors on both the engine and transmission, due to their rigid connection, which results in noisy measurements obtained from the sensors connected to both the ECUs. The engine-synchronous signals such as the engine rotation position, the cylinder firing order or the cam/crank tooth times can be used to filter the engine vibration noise from the sensor data. This is currently possible in the engine ECU, but not in the transmission ECU. Having an engine-synchronous signal in the transmission ECU would make it possible to schedule software functions with simple and robust filters whose effect varies implicitly with engine rotation and therefore help to filter

out the noise. We aim to solve this in our thesis work by developing a method to communicate the engine-synchronous data from the Engine ECU to the transmission ECU which would result in more accurate sensor data, thereby enabling the use of simpler and more reliable control software.

1.1 Goals

The goal of this thesis work is to develop a proof of concept solution for executing engine-synchronous software functions in the transmission ECU. This involves finding a suitable method to communicate the change in engine rotation position from the engine to the transmission and schedule software functions to be run engine-synchronously in the transmission. This work also includes setting up a simple hardware test rig to implement, test, and demonstrate these methods using our physical ECUs. The final objective is to implement and demonstrate the software function developed on-board in a running truck.

1.2 Challenges

- Identifying and implementing a communication system for communicating the engine-synchronous signal from the engine ECU to the transmission ECU.
- Developing and scheduling a filtering software engine-synchronously for filtering the noise from the raw sensor measurements for some of the sensors (such as the inclination sensor) affected by the engine vibration in the transmission ECU.

1.3 Limitations

- There are interfering processes scheduled on the ECUs but these are not within the scope of this thesis.
- The CAN transmission function is triggered in a 1.25ms transmission task at Volvo GTT which would cause latency in the data transmitted. This is not a limitation of the CAN protocol itself, but a limitation due to software configuration at Volvo GTT.
- To verify engine-synchronous scheduling in the transmission ECU, a simple filter is used to engine-synchronously filter the vibration noise. Creating the most optimal filtering solution is not the focus of this thesis project.

1.4 Thesis Outline

The thesis report follows the below structure:

- **Chapter 2: Theory** describes the theory required for understanding the components used in the project.

- **Chapter 3: Design and Implementation** explains the high-level design and gives a technical overview of the implemented system.
- **Chapter 4: Results** explains the output obtained from the thesis work.
- **Chapter 5: Conclusion** discusses the results and provides a summary of the thesis work.

2

Theory

In this chapter, the following sections describe the theory required for understanding the different components and concepts used in this thesis work.

2.1 Vehicle drive system

A background of the vehicle drive system comprising of the engine and the transmission components is discussed in the sections below.

2.1.1 Engine

An internal combustion engine (ICE) is a type of heat engine in which the fuel is combusted inside the engine's combustion chambers. The different engine parts work together and convert chemical energy into mechanical energy. The high pressure and force generated by the fuel-air mixture inside the cylinder causes the piston inside the cylinder to move upwards and downwards. The crankshaft converts the pistons' upward and downward movements into rotary motion which drives the vehicles' wheels. Figure 2.1 shows the internal and external views of the combustion engine of a Volvo truck which comprises of six cylinders. For each rotation of the crankshaft, three cylinders are fired and it therefore takes two rotations for all six cylinders to be fired.

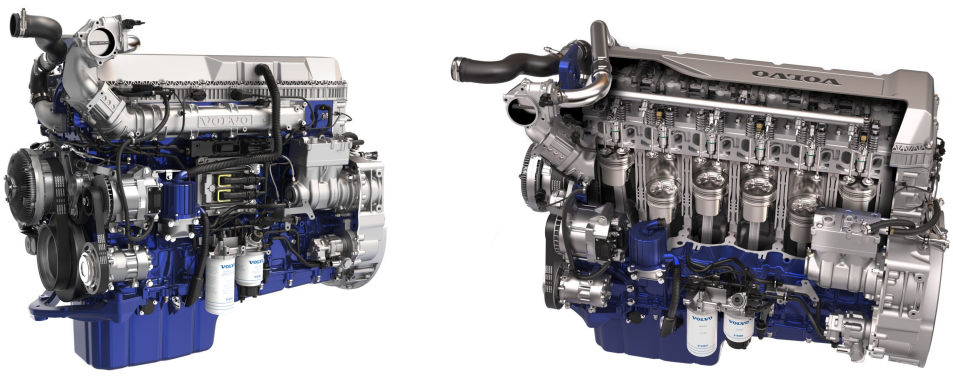

Figure 2.1: Engine of a Volvo truck - external view (left) and internal view (right)

2.1.2 Transmission

The transmission of a vehicle consists of the gearbox and it is the gateway between the engine and the other components of the vehicle. The main function of the transmission is to use the input power from the engine and navigate it through the gearbox to maintain an optimal efficiency by switching between different gears. The clutch is used to engage and disengage a gearbox in order to enable the gears to be changed without causing any damage and to prevent the engine from stalling. The gearbox is an assembly of gears used to transform the speed and torque of a vehicle and to change the direction of the vehicle to either forward or reverse. Figure 2.2 shows the external and internal view of the transmission of a Volvo truck.

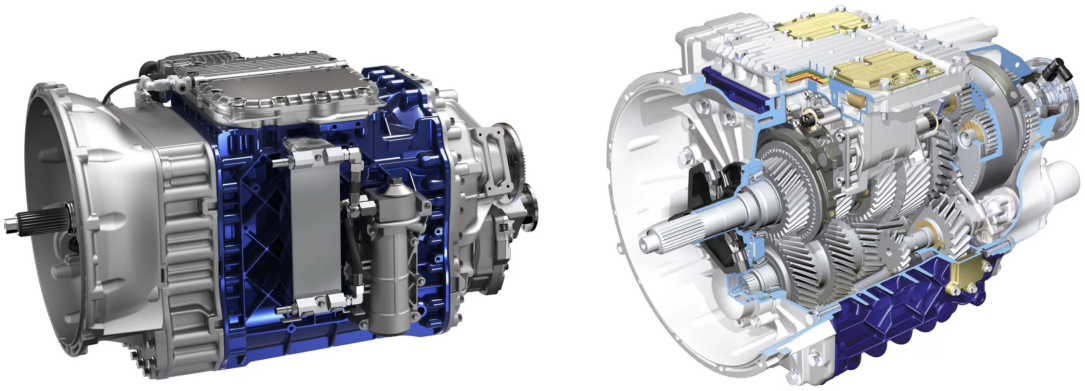

Figure 2.2: Transmission of a Volvo truck - external view (left) and internal view (right)

2.2 Electronic Control Unit

An electronic control unit (ECU) is a real-time embedded system that controls the operation of an electronic sub-system in a vehicle. Using the data from various sensors inside the vehicle, the ECU controls and manages different functions in a vehicle such as steering, anti-lock braking, controlled fuel injection, and so on through its actuators [3, 4, 5]. Figure 2.3 shows an ECU with the inputs from various sensors and the outputs sent to the different actuators.

An ECU consists of the following elements:

- Micro-controller
- Memory (Flash [7], SRAM [8], EEPROM [9])
- Inputs/Outputs (analog and digital)
- Communication Interfaces
- Connectors

The ECU hardware includes electronic components on a printed circuit board (PCB) and the software is stored in the key elements in an ECU such as the micro-controller,

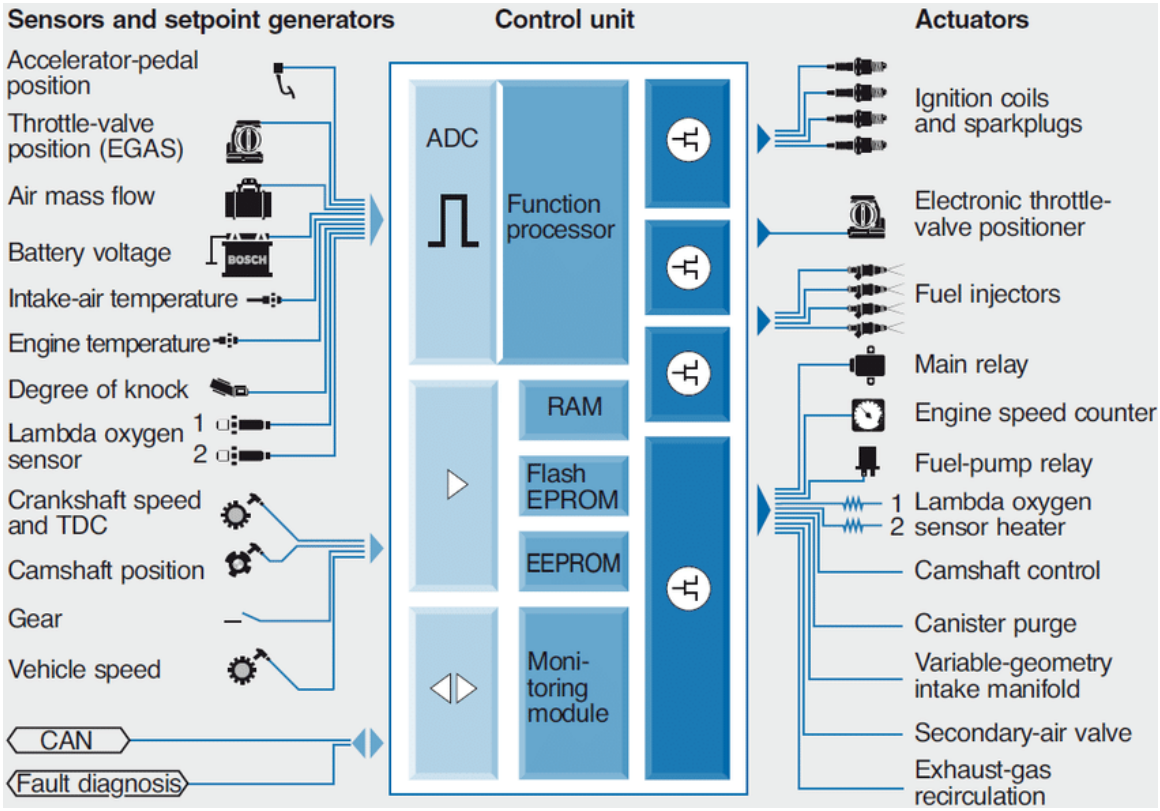

Figure 2.3: Electronic control unit - inputs and outputs, Source: [6]

SRAM and EEPROM/Flash [10]. The signals from the sensors are processed in real-time by the micro-controller [11].

There are different types of ECUs inside a vehicle and they are classified based on the functions they perform. In this thesis work, we mainly focus on the ECUs of the engine and the transmission.

- **Engine Electronic Control Unit:** The Engine ECU (EECU) is also known as the Engine Control Module (ECM) and is used to control engine operations such as fuel injection, emissions control, and engine actuator control. It also communicates with the other ECUs in the vehicle. The EECU is commonly mounted on the engine. Using the measurements recorded by the camshaft and crankshaft position sensors, it is possible to identify the identity of the cylinder fired, the engine rotation position, the firing order of the pistons in the engine, the rate at which the engine rotates and so on.

- **Transmission Electronic Control Unit:** The Transmission Electronic Control Unit (TECU), also known as transmission control module (TCM) is used in the vehicle for various control functions in the transmission component. The TECU performs functions such as clutch control and gear selections. The TECU enables the correct gear to be chosen at the right time to optimize the driving performance and fuel usage.

2.3 Inter-ECU communication

Near real-time communication between the different ECUs located within a vehicle is one of the most important aspects which impacts the functionality and overall safety of the system [12]. A simple block diagram of inter-ECU communication is shown in Figure 2.4. The ECUs can communicate with its sensors and actuators as well as with each other through different communication protocols [13] such as:

- A direct dedicated line
- Controller Area Network (CAN) [14]
- Ethernet [15]
- FlexRay [16]
- Media Oriented Systems Transport (MOST) [17]
- Local Interconnect Network (LIN) [18].

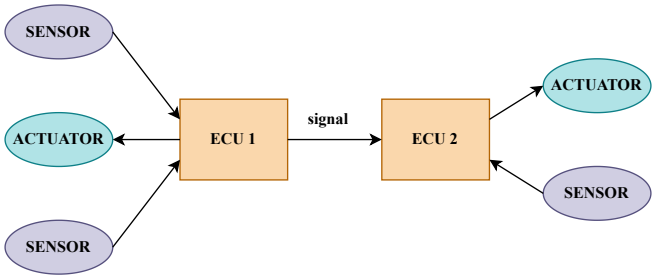

Figure 2.4: Inter-ECU communication, Source: Adapted from [19]

Figure 2.5 shows inter-ECU connections with multiple ECUs connected using direct dedicated lines (on the left) and using CAN bus (on the right).

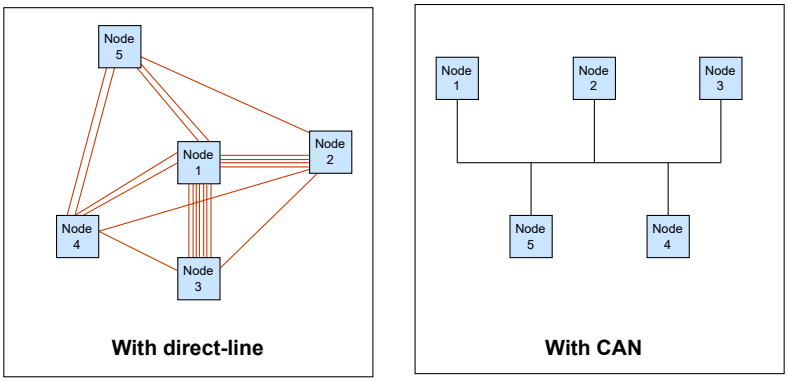

Figure 2.5: Inter-node communication with Direct-line communication (left) and CAN Bus (right)

2.3.1 Communication via direct dedicated line

The communication between the ECUs is set up using a direct dedicated line between an output port of the EECU to an input port in the TECU. Since only the

time between the change in cylinder firing is required by the TECU, using a direct line between the two ECUs allows us to send a pulse with a rising edge to be triggered whenever the engine rotation position changes. The Operating system (OS) configuration is updated to generate an interrupt whenever a change occurs in the signal, thus enabling a task to be run at the TECU engine-synchronously. A direct dedicated line helps in near real-time transfer of data especially used for sending pulse signals to trigger software tasks from one ECU to another. However, the direct-line communication results in an increase in wiring harness in a truck and hence may not be the most suitable as a stand-alone method and can instead be used in combination with other communication methods such as CAN and Ethernet. With direct-line communication it is not possible to do error detection and auto re-transmission of lost messages. This can however be rectified using additional error detection and re-transmission within the ECU software functions.

2.3.2 Communication via CAN bus

One of the main communication systems used for inter ECU communication in a conventional truck is using the Controller Area Network (CAN) protocol [14]. CAN supports both periodic as well as aperiodic data transmission.

The CAN protocol is a serial bus communication protocol which enables a robust communication between the ECUs. It is a standard used for efficient and reliable communication between sensors, actuators, controllers and other nodes in real-time applications for use in the automotive industry [20]. The usage of a common CAN bus has helped significantly in reducing the wiring harness size in automobiles, thereby making it one of the common communication systems in vehicles. It has a data rate of up to one megabit per second and twisted pair cable is used to connect between devices. The benefits of using CAN are prioritization of messages, system-wide data consistency, error detection, re-transmission of messages, multicast reception with time synchronization, possibility of setting a device to sleep mode and acknowledgement of received messages. However, CAN only supports a limited number of nodes and has a high maintenance cost.

There are different types of CAN messages with different functions:

- Data Frame
- Remote Frame
- Error Frame
- Overload Frame

The data frame of the signals sent over the CAN bus is as shown in Figure 2.6, which is the most common type of frame where the Message ID field consists of the priority of the message transmitted and the Data field consists of the data being transmitted.

The data frame has seven fields which are Start of Frame (SOF), Arbitration, Control, Data, Cyclic Redundancy Check (CRC), Acknowledgement (Ack), and End of Frame (EOF). Start of Frame consists of one bit which represents the beginning of

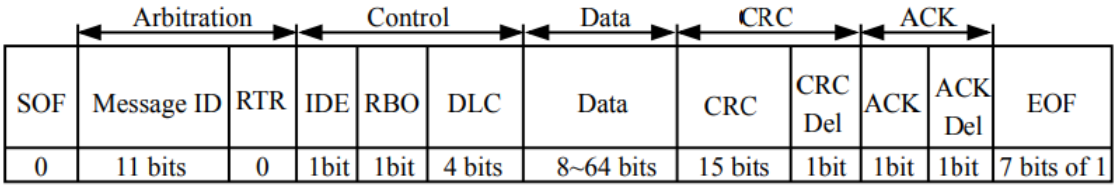

Figure 2.6: CAN Message Frame, Source: [21]

CAN message and End-of-Frame field is a 7 bit field which marks the end of the frame. The data field can hold up to 64 bits of data which allows multiple signals to be transferred in a single frame.

2.3.3 Comparison between CAN and direct-line communication

Table 2.1 shows the comparison between the CAN and direct-line communication systems with their advantages and disadvantages.

Table 2.1: Comparison between CAN and direct-line communication

Communication system	**Advantages**	**Disadvantages**
CAN	- Low implementation cost - High speed (upto 1 Mbit/sec) - Error detection - Reduced wiring harness - Auto-retransmission of lost messages	- Limited number of nodes - High maintenance cost - May have signal integrity issues - Interference from other processes
Direct dedicated line	- Near real-time transfer of data - Suitable for sending pulse signals - No interference from other processes since it is a dedicated line	- Increased wiring harness - Re-transmission of lost data not possible - Error detection not possible

2.4 Sensors

A vehicle has different types of sensors such as position sensors, speed and velocity sensors, acceleration sensors, pressure sensors and temperature sensors. The values measured by the sensors are used in the different ECUs of the vehicle to perform various functions to control the vehicle. Figure 2.7 shows some of the sensors used in a motor vehicle. In this thesis report we mainly focus on the engine speed sensor in the engine and the inclination sensor in the transmission.

2.4.1 Engine speed sensor

The engine speed is measured using a variable reluctance (VR) sensor. The VR sensor mainly consists of a transducer used to measure changes in the magnetic

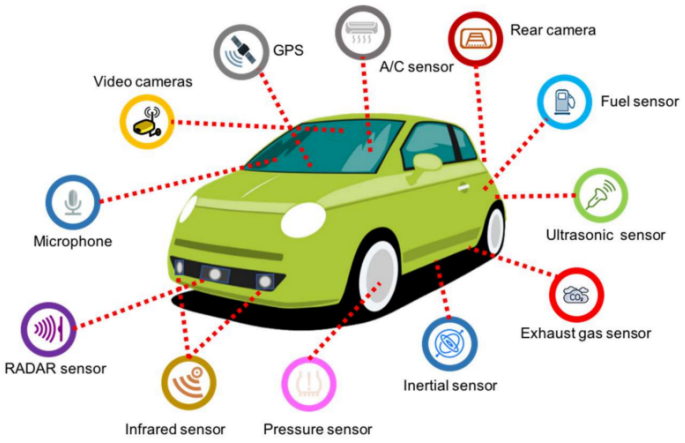

Figure 2.7: In-vehicle sensors, Source: [22]

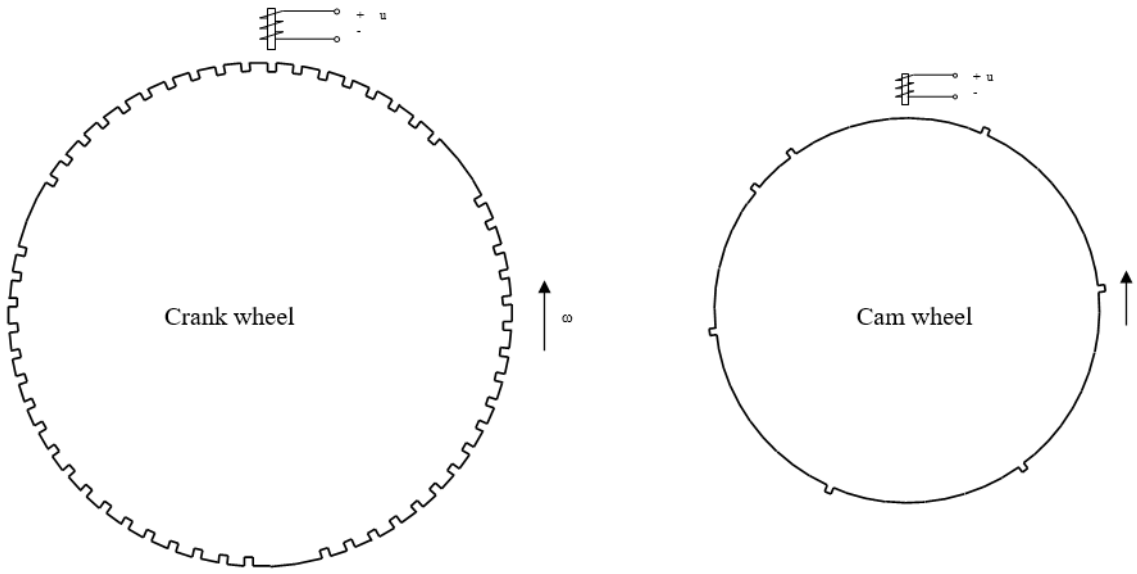

Figure 2.8: VR sensors used in the crank and cam wheels

reluctance of the engine cam and crank shaft rotation. A camshaft consists of six egg shaped cams which are used to open and close engine valves, while a crankshaft converts the upwards and downward motion of the cylinder pistons to rotational motion. The VR sensors count the number of teeth on the camshaft and crankshaft passing the sensors and the sensor measurements are sent to the engine ECU in which software functions are used to calculate various parameters such as tooth times, engine rotation angular position, engine speed, cylinder ID and so on. Figure 2.8 shows the image of the crank and cam wheels with the VR sensors. The crank wheel of a Volvo truck engine can be seen in Figure 2.9.

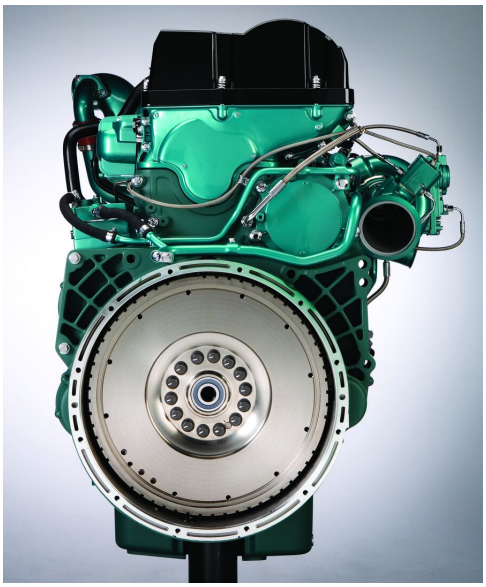

Figure 2.9: Crank wheel of a Volvo truck engine

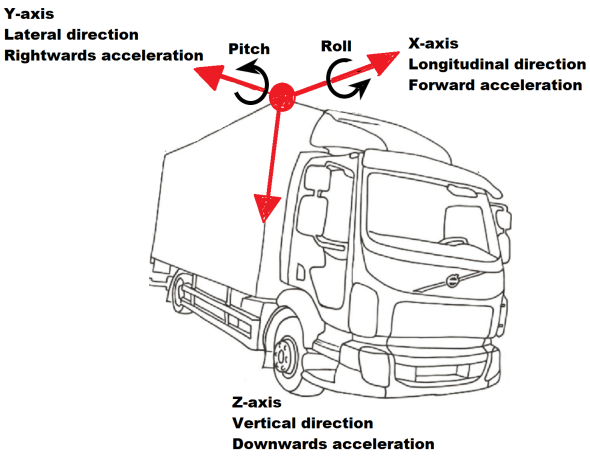

Figure 2.10: The 3 axes of the inclination sensor depicted on a Volvo truck

2.4.2 Inclination sensor

The inclination sensor is present in the TECU and consists of a 3-axis accelerometer, which is an electronic sensor used to detect the the acceleration and orientation of an object in space by measuring the acceleration forces that act on the object and uses Serial Peripheral Interface (SPI) for data transfer. Figure 2.10 shows the 3 axes of the inclination sensor depicted on a Volvo truck where the X axis refers to the roll and the Y axis refers to the pitch or the inclination.

2.5 Microcontroller

A microcontroller unit (MCU) is an integrated circuit designed to perform tasks in an embedded system. A micro-controller consists of Central Processing Unit (CPU), memory, Input/Output (I/O) ports and Serial communication interface. There are various categories of micro-controllers depending on the memory, architecture, word-length and instruction sets.

The CPU, memory and I/O peripherals are the main three elements of microcontroller. The CPU is called the brain of the microcontroller. It comprises of an arithmetic and logic Unit (ALU) which performs all the arithmetic and logical operations, a control unit (CU) which controls all the operations, a memory unit (MU) and I/O peripherals. The memory inside the controller is divided into two: program memory and data memory. The program memory stores the instructions to be executed and the data memory holds the temporary data while the instructions are executing. I/O peripherals are the interface to the controller to the input/output ports.

Micro-controllers also have other peripherals that are used to interface them to the user. Some examples of peripherals are timers, analog to digital converter, digital to analog converter, serial communication (SPI, I2C, and UART). The peripherals used and configured in this thesis work are discussed below.

2.5.1 Generic timer module

The generic timer module (GTM) is an intellectual protocol (IP) block developed by Bosch and different semiconductor companies [23]. It handles the different timing operations in the microcontroller and is made up of a generic timer platform. The main task of the GTM is to offload the CPU by performing timing operations in hardware rather than in software. This can be done by various programmable modules. The GTM continuously records changes in digital input signals and these are combined with digitized analog signal for calculation. To transfer data between sub-modules GTM has an advanced router unit (ARU) integrated into it. GTM supports a wide range of timer applications, including dynamic digital pulse-width modulation (PWM) output

The GTM has many configurations and based on the configuration, it can be classified into sub-modules such as ATOM, BRC, MCS, PSM, SPE, TIM, TOM. The module used in this thesis work is the timer input module (TIM).

- **TIM:** The timer input module handles filtering and capturing of input signals to the GTM. Using TIM modules, different properties of the input signal can be measured, such as the number of rising or falling edges and the PWM signal period in parallel and sending data to the ARU [24]. It also has a filtering module which performs operations such as filtering glitches and selected mode operation. The different operations of the TIM module are capturing input time stamp, input edge counting, periodic sampling and so on.

2.5.2 Interrupt router module

An interrupt is a notification sent to the processor either from hardware or software to perform a certain action. When an interrupt is issued, the current task execution is completed by the micro-controller and the interrupt indicates what action is to be executed next. The interrupt router (IR) module includes the service request nodes (SRNs) and the interrupt control units (ICUs), which tells the processor or controller what to do when the interrupt occurs. A hardware interrupt is an electronic alerting signal sent to the processor from an external device and a software interrupt is caused either by an instruction or condition in the software function.

2.5.3 Direct memory access

Direct memory access (DMA) is used by a hardware sub-system to reduce the load on the processor by transfer data to and from input output devices. The transfer is done using a DMA controller which accesses the memory independently without involving the processor.

2.5.4 Serial peripheral interface

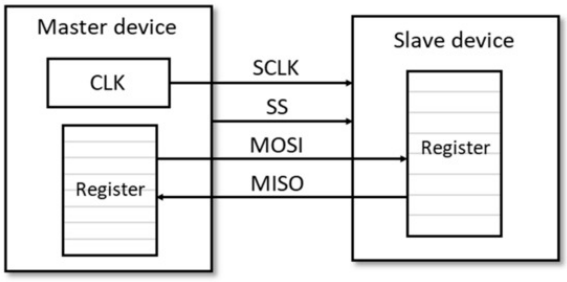

Figure 2.11: Traditional 4-wire serial peripheral interface (SPI) protocol block diagram, Source: [25]

Serial peripheral interface (SPI) is a widely used bi-directional communication interface used for synchronous serial communication between micro-controllers and peripherals. SPI consists of four signals as shown in Figure 2.11: Serial Clock, Chip Select, master-out/slave-in (MOSI) and master-in/slave-out (MISO). Serial clock is the signal which synchronizes the data. The MOSI and MISO enables SPI to operate in full duplex mode. SPI enables high speed communication at low cost and also allows multiple slave nodes.

2.5.5 Queued synchronous peripheral interface

Queued synchronous peripheral interface (QSPI) is an SPI controller that provides serial communication using a data queue to communicate with external peripherals in a micro-controller. It supports full-duplex, half-duplex and simplex modes. It also allows communication between two devices using the four signals: clock, data-in, data-out and slave select.

2.6 Filters

Filters are commonly used in electronic circuits. They can be used to alter signal characteristics with respect to frequency and phase. A filter is used to extract the important frequencies in a signal by attenuating unwanted frequencies. There are different types of filters depending on the frequency domain response such as low pass filter, high pass filter, band pass filter and band stop filter.

2.6.1 Low pass filter

A low pass filter (LPF) is designed to pass signals with lower frequencies while attenuating the higher frequencies above the cutoff frequency, often used to filter out high frequency noise in a circuit. An ideal low pass filter removes all the frequencies above the cutoff frequency and hence has a rectangular frequency response without a transition region, whereas a real low pass filter has a transition region and hence causes in a delay in the signals. The delay of the low-pass filter increases as the filter bandwidth decreases. The order of the filter is chosen based on the stop band roll-off rate. A first order low pass filter has a roll-off rate of 20 dB/decade whereas the second order low pass filter has double the roll-off rate of 40 dB/decade. Hence the higher the filter order, the higher the improvement in the stop band roll-off rate. However, an increase in the order of the filter causes an increase in the delay of the signals. Also, low pass filters are used in applications such as speech processing, reconstruction and many more [26]. It is also used to prevent the aliasing effect before sampling [27]. Figure 2.12 represents the general filter response of a low pass filter.

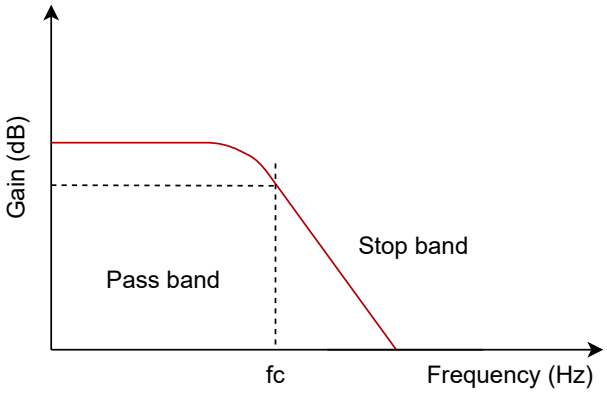

Figure 2.12: Low pass filter response

2.6.2 Moving average filter

A moving average filter is a finite impulse response (FIR) filter that helps to smoothen the signal by reducing the random noise in the signal and any sharp step response in the signal is retained [28]. This filter consists of longer time series data and shorter filter window, as the filter length increases the output becomes more smoother by

making the sharp transition unsharpened. Figure 2.13 represents the general filter response of a moving average filter.

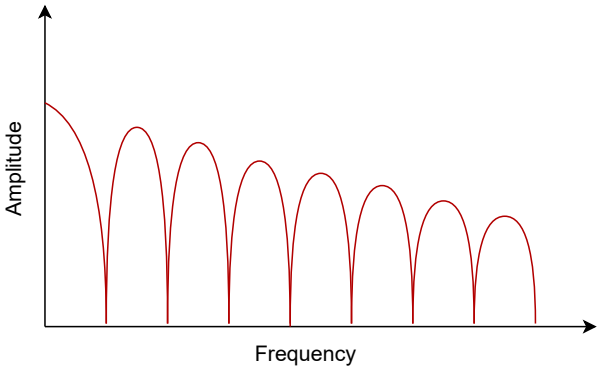

Figure 2.13: Moving average filter response

3

Design and Implementation

This chapter gives the details of the various hardware and software tools used in this thesis work and explains the reference software and the implemented prototype.

3.1 Hardware tools

The following hardware tools were used for implementing the solution using a test rig and for the truck testing.

- Engine electronic control unit
- Transmission electronic control unit
- JTAG Lauterbach debugger
- Engine speed simulator
- CAN/LIN Interface

3.1.1 JTAG Lauterbach power-debug interface

Figure 3.1: Lauterbach TRACE32 debugger interface

Lauterbach Power-Debug interface is a microprocessor development tool from Lauterbach GmbH which allows us to upload the software functions to the ECUs from a computer. It provides a Universal Serial Bus (USB) interface to the computer and allows flash programming [29]. It also supports high level language (HLL) debugging. The ECUs which are used for testing have a port available for the debugger in order to download software as well as debug the signals to and from the ECUs. Figure 3.1 shows the picture of a Lauterbach TRACE32 debugger interface.

3.1.2 Engine speed simulator

The engine speed simulator is a type of a sensor stimuli device which is used to emulate the cam and crank signals. It is controlled by the engine speed box controller GUI and it has a basic speed control functionality. Currently, only the cam and crank output signals are supported.

3.1.3 VN1640A - CAN/LIN interface

The VN1640A - CAN/LIN Interface is known as a CAN case, which is a hardware interface manufactured by Vector Informatik GmbH and is used to connect to the CAN bus as well as to a computer. It includes four channels to connect the CAN bus and a USB to connect to the computer. It allows us to observe the communication on the CAN bus.

Figure 3.2: VN1640A - CAN/LIN Interface

3.2 Software tools

3.2.1 Trace32 debugger

Trace32 is a software tool for debugging embedded software and hardware using the Lauterbach JTAG hardware interface. In the automotive industry, Trace32 is used for development of ECU software by monitoring real-time signals and debugging the signals. It also allows the programmer to debug the source code by setting up breakpoints and by executing the code line by line.

The different purposes for which the debugger tool is used in this thesis work are as follows:

- Load software - Load software to the EECU and TECU.
- Breakpoint(s) - Creating multiple breakpoints at particular parts of the code to check the flow of the code.
- Step/Step over call - Used to execute the code line-by-line and see the variables changing after code execution stops at breakpoints.

- Watch window - To watch the input-output parameters, variables and signals changing when the code is executing.
- Variable manipulation - Changing the variable values for debugging purpose.

3.2.2 ATI VISION

ATI VISION is a data acquisition, calibration, and post-analysis tool that is used in this thesis work to view and analyze the signals from the ECUs. The tool is configured to get the values from the ECUs to the computer. The ECUs are added to the device setup as CAN Calibration Protocol (CCP) based devices as shown in Figure 3.3. Once the devices are added to the configuration, the communication with the devices connected are enabled and the relevant signals from the ECUs can be viewed.

At Volvo GTT, VISION is mainly used for monitoring different signals from various ECUs in the truck during the truck testing. For this thesis work, various signals from both the TECU and EECU are selected in VISION and the live changes are monitored while the truck is moving. These logs are then saved for further analysis of the signals.

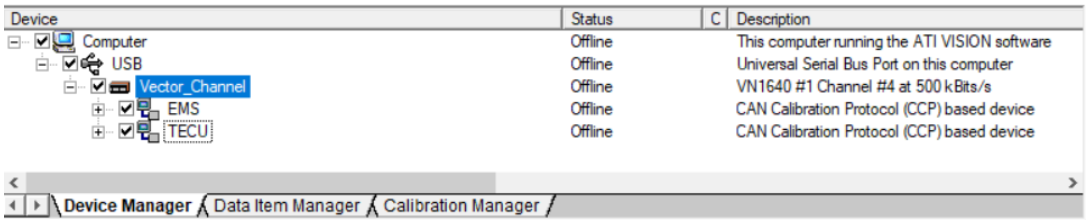

Figure 3.3: VISION device configuration

3.2.3 CANalyzer

CANalyzer is a software tool developed by Vector Informatik GmbH which is widely used in the automotive industry for the analysis of CAN, LIN, MOST and FlexRay based communications. It is an analysis tool for ECU networks and distributed systems. The CANalyzer has various features to analyze, simulate and test the signal communication. With CANalyzer it is possible to send messages that can influence the ECUs and control the hardware. Figure 3.4 shows the main configurations in the CANalyzer application.

3.2.4 RTA-OS

Real-time AUTOSAR operating system (RTA-OS) is a real-time operating system that can be configured on a computer and the built operating system (OS) files can be downloaded to a target hardware device. It is used to make configuration changes for the operating system of the target hardware device. The OS configuration is done using an executable file called rtaoscfg which is a graphical configuration editor.

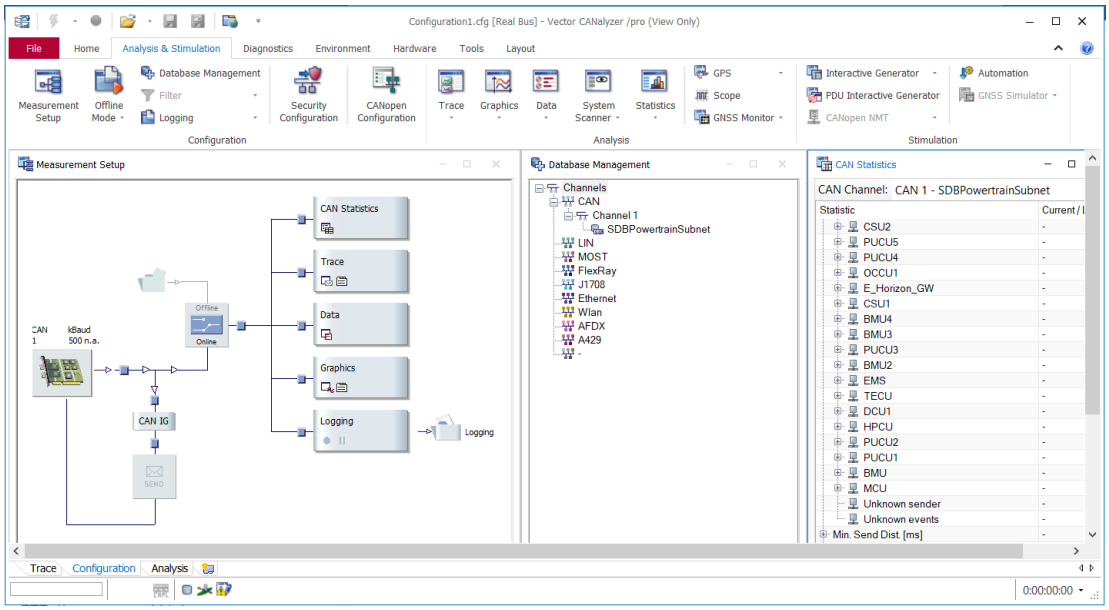

Figure 3.4: CANalyzer configuration

3.3 The reference system

The latest software solution contained by a conventional Volvo truck is taken as the reference software for this thesis work. The inter-ECU communication setup and the sensor filtering software used in the reference system are described in the following sections.

3.3.1 Inter-ECU communication

Figure 3.5 shows the software solution of the reference system for inter-ECU communication using CAN. The CAN communication system at Volvo GTT includes several components responsible for performing different tasks. The program used for the software implementation is in C programming language. Each component consumes or produces one or more signals as shown in Figure 3.5. The components contain the software functions that are scheduled and executed periodically by the operating system, and communicate with each other using signal objects. When a signal object is created, it needs to be declared as a producer and/or a consumer and for each signal and the signal behavior also needs to be updated depending on whether it is a periodic signal or aperiodic signal.

The CAN component helps in performing the translation of signal objects to CAN signal via signal mapping. In the signal mapping process, all CAN signals are connected to signal objects and these CAN signal names are updated in a communication database file known as the DBC (database CAN) file. It contains all information describing the network and can be read by numerous applications. The DBC file is made up of signals, messages and nodes or ECUs connected through CAN. The CAN signals required to be sent or received are added to the signals of

the DBC file where various signal parameters such as datatype, bit size, and units can be configured. Each CAN message frame is configured to contain one or more signals along with the transmitting and receiving nodes for each message frame. In developing a distributed ECU network based on CAN, a key component is the communication description in the form of DBC files. From the C code, the signal objects created are read and written with their Get and Set functions.

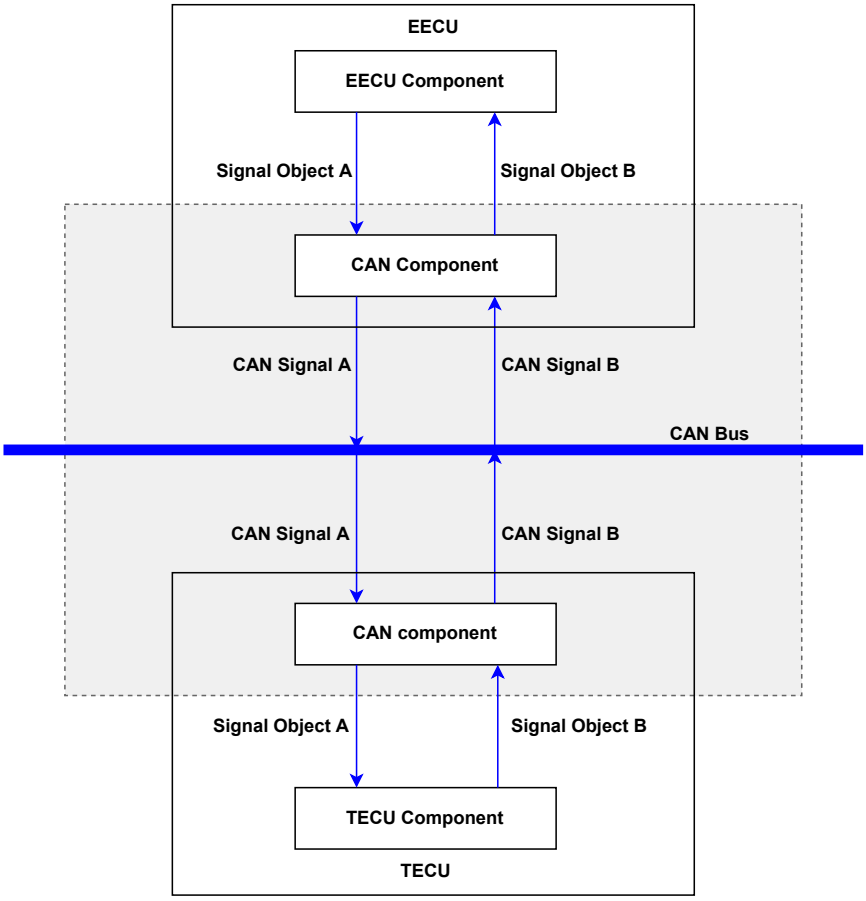

Figure 3.5: Reference software design for inter-ECU communication using CAN

3.3.2 Filtering software in TECU

In the current solution implemented in the reference system, the raw data from the inclination sensor with noisy measurements are stored in an array of five values which is updated every 10 ms. The filtering software function runs every 10 ms and converts the raw data taken from the array to the acceleration data in unit G. This data is then sent to a first order low-pass filter which has a cut-off frequency of 5 Hz and produces the final inclination value. The time and frequency domain plot for the reference system is shown in Figure 3.6. The time domain plot was done for three different engine speeds of 600, 1300 and 2000 rpm. The frequency domain plot was generated for the speed of 600 rpm. From the frequency domain plot, it can be seen that the engine vibration noise peaks are not removed by the low pass filter. The cut off frequency was chosen as 5 Hz and not a lower value, in order to

filter as much as possible the engine induced oscillations, but without introducing more delay on the same order of what is already there later in the filtering process.

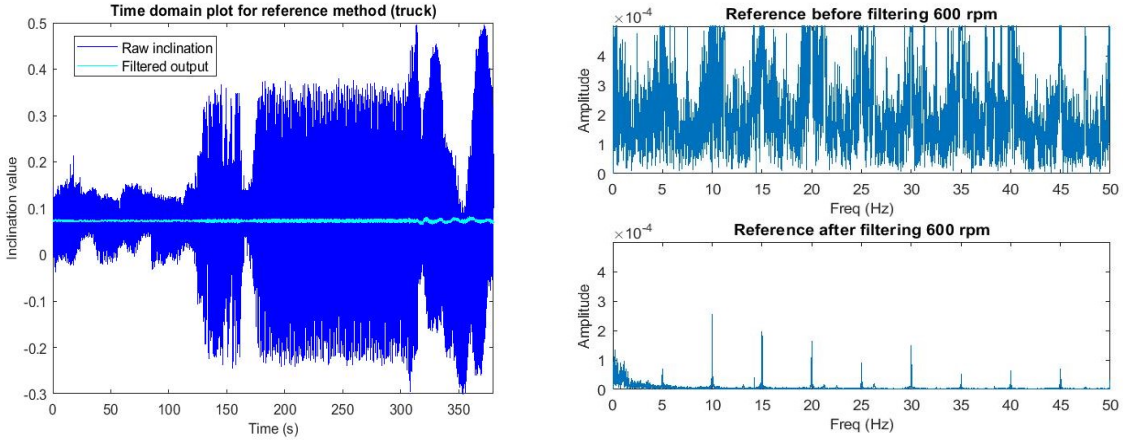

Figure 3.6: Time domain plot for reference method for 600 rpm, 1300 rpm and 2000 rpm (left) and frequency domain plot for 600 rpm for reference method (right)

3.4 Implementation

In this thesis, we used two methods to communicate between the two ECUs. These methods are explained below. Also, we discuss the implementation of the scheduling of software functions for the filtering software in the TECU.

3.4.1 Dedicated direct-line between ECUs

One of the methods identified for communication between the ECUs is a direct dedicated line connection. A direct dedicated connection was made between the output pin of one of the unused ports of the EECU and the input pin of an unused port of the TECU.

3.4.2 Test rig hardware setup

Figure 3.7 shows the test set up for the dedicated direct-line method. Figure 3.8 shows a picture of the test rig. The cam signal and crank signal wires from the EECU are connected to the engine speed simulator to emulate different engine speeds for the EECU. The communication between the EECU and TECU is done via a dedicated direct-line setup and the TECU uses the information received from the direct-line in the filtering software to reduce the noisy measurements. The inclination data is read by the TECU from the inclination sensor, which is embedded into the TECU. A debugger is then used to load software into both the EECU and the TECU and to monitor how the filtered output signal value changes when different speed values were set using the engine speed simulator.

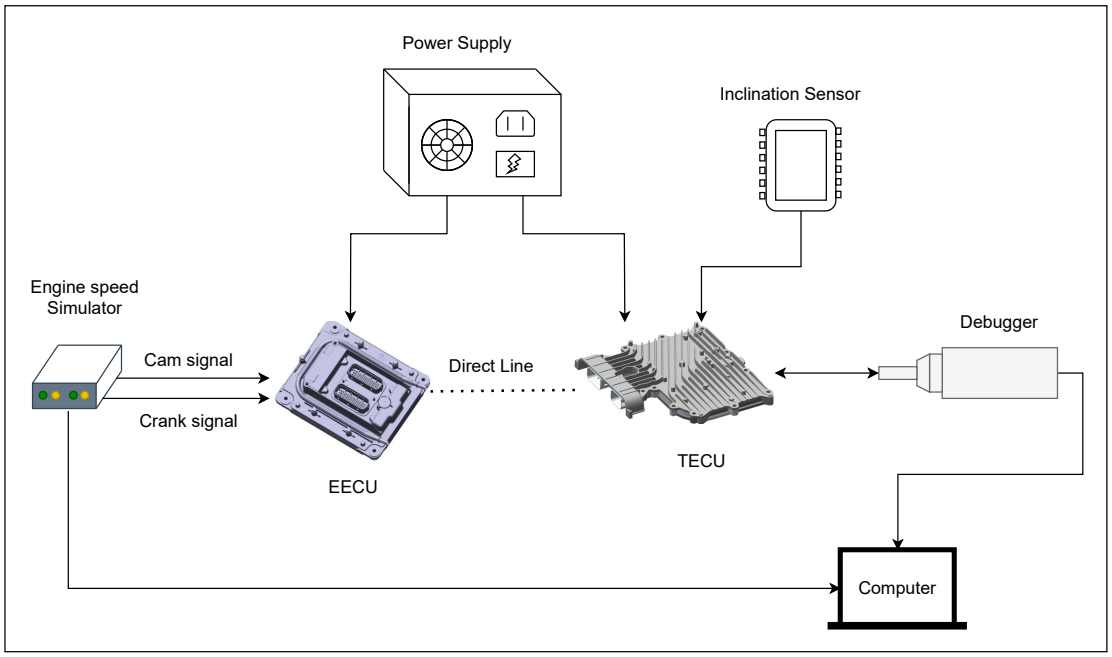

Figure 3.7: Test bench schematic for the dedicated direct-line method

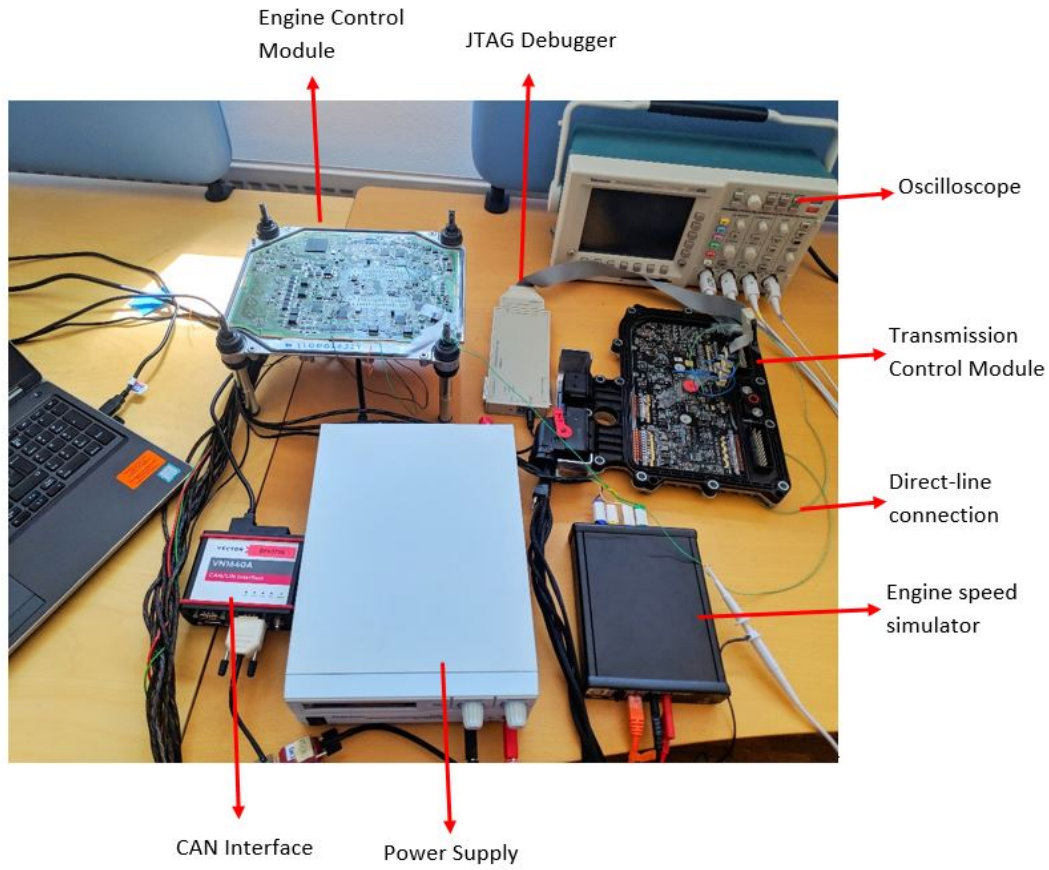

Figure 3.8: Test rig hardware set up for direct-line communication

3.4.2.1 Updating the output port configuration in EECU

The port 15 was identified to be unused and the output pin 12 was configured in the GTM peripheral in the TIM module of the micro-controller to send out high and low pulse signals from the EECU whenever the Cylinder ID signal value changes, and one end of the direct-line wire was soldered to it.

3.4.2.2 Updating the input port configuration in TECU

For this thesis work, port 32 was identified to be unused and input pin 4 was configured in the GTM peripheral in the TIM module of the micro-controller to receive the high and low pulse signals. The other end of the direct-line wire was soldered to this pin. When any changes in the received signal occurs, it triggers the interrupt added to a new engine-synchronous component added in TECU, which triggers the inclination sensor filtering software function. To configure the interrupt, the TIM channel 0.5 corresponding to the port was identified and the GTM configuration file in TECU was updated to trigger the interrupt whenever the high and low pulses are received through the direct-line connection.

3.4.2.3 OS configuration

The OS configuration was done in the following steps using the RTA-OS tool:

- A new task was added to run in the same processor core on which the inclination sensor software is configured.
- The newly added task was scheduled to run engine-synchronously by adding an interrupt to trigger the task whenever a change in the output port was detected.

Once the OS configuration was updated, the build files were generated for the TECU which were then loaded to the TECU through the Lauterbach debugger.

3.4.3 CAN communication setup between ECUs

Another communication method that was investigated was CAN. The software set up for the CAN communication involves creating the new signal objects for the signals to be sent, setting up the signal mapping for the new signal objects, database update and building the project. These steps are discussed in the subsections below. The hardware setup for the CAN communication method was done as shown in Figure 3.9.

The Engine control module and the transmission control module were connected via the CAN bus and a CAN interface was used to connect the CAN bus to the PC. The speed simulator was used to simulate the engine speed and a debugger was connected to the TECU to view the received signal values. The debugger was also used to load the software in both EECU and TECU.

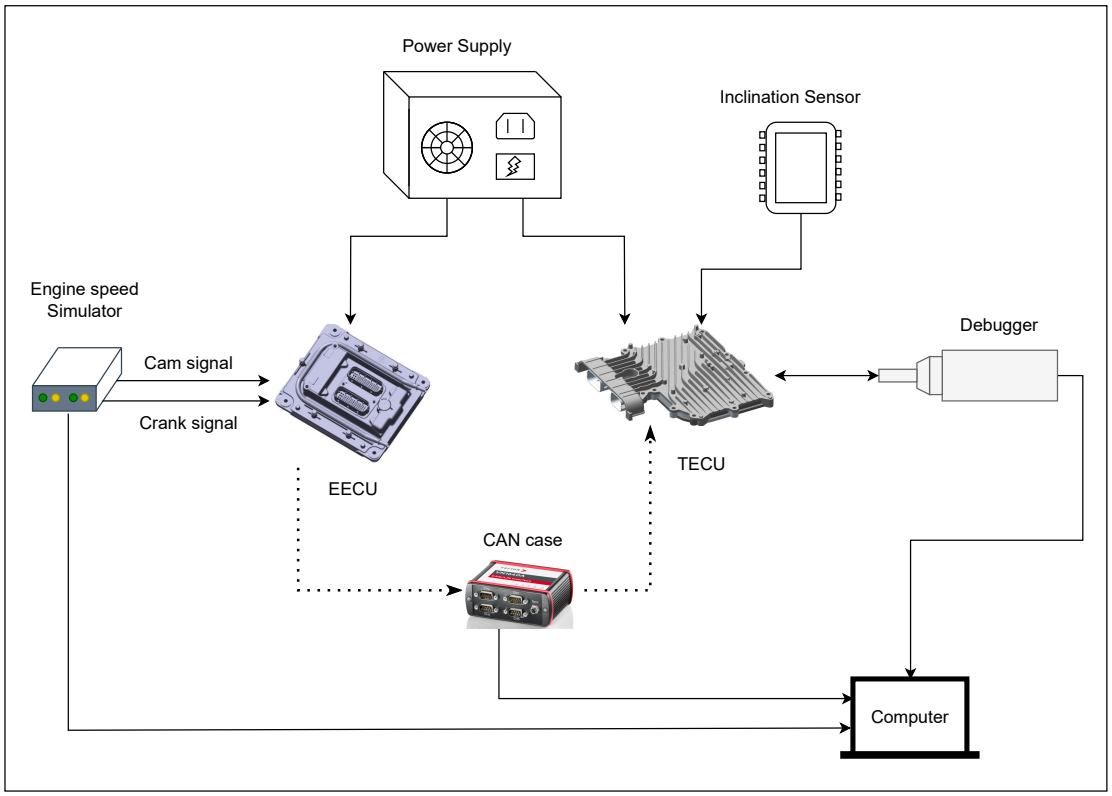

Figure 3.9: Test bench schematic for the CAN communication method

3.4.3.1 Sending signals from EECU

The Cylinder ID signal was chosen to identify the change in engine rotation position. This was calculated using the tooth times recorded using the VR sensor and this signal was available in the engine ECU component. In order to send the Cylinder ID signal via CAN to the transmission ECU, a CAN signal object was created in both the EECU and TECU components, where the software functions were scheduled. The new signal object created was then mapped to the engine ECU signal in the signal mapping file. Since the signal was to be sent when the Cylinder ID changes, it was configured to be sent aperiodically. As a result, the signal description for an aperiodic signal is given in the signal mapping.

The signal was configured to be sent aperiodically by also updating the signal send types in the DBC file. After creating the signal, a new message in the DBC file was added for sending it, and the signal was added to the same message with the message send type for sending aperiodic signals. Finally, the transmitter and receiver for the message and signal were configured. As the communication is done from EECU, the transmitter was set to EECU. Once the DBC is updated, a series of scripts were run to generate the ECU specific build files, which were then loaded to the ECUs through the debugger.

3.4.3.2 Receiving signals in TECU

The DBC file was updated for receiving the signal in the TECU by setting the receiver to TECU. For accessing the transmitted signal in the C code, a Set function was used to write the signal object in the TECU side. Finally, a sequence of scripts were run in the TECU software to generate build files, which were then loaded into the TECU hardware using the Lauterbach debugger.

3.4.3.3 Analyzing the received signal

We were able to send the engine-synchronous aperiodic signal from engine control module to transmission control module. The signal received in the TECU were monitored by connecting the Lauterbach debugger. The Engine RPM was modified using a speed box simulator and the change in the Cylinder ID and the corresponding timestamp in the TECU side were verified. Using the CANalyzer, the signals were viewed and a log of the received data was taken for further analysis. The Cylinder ID signal received in CAN for different engine speeds was viewed in the CANalyzer as shown in Figure 3.10. It can be seen that the time between cylinder firing decreases as the engine speed increases.

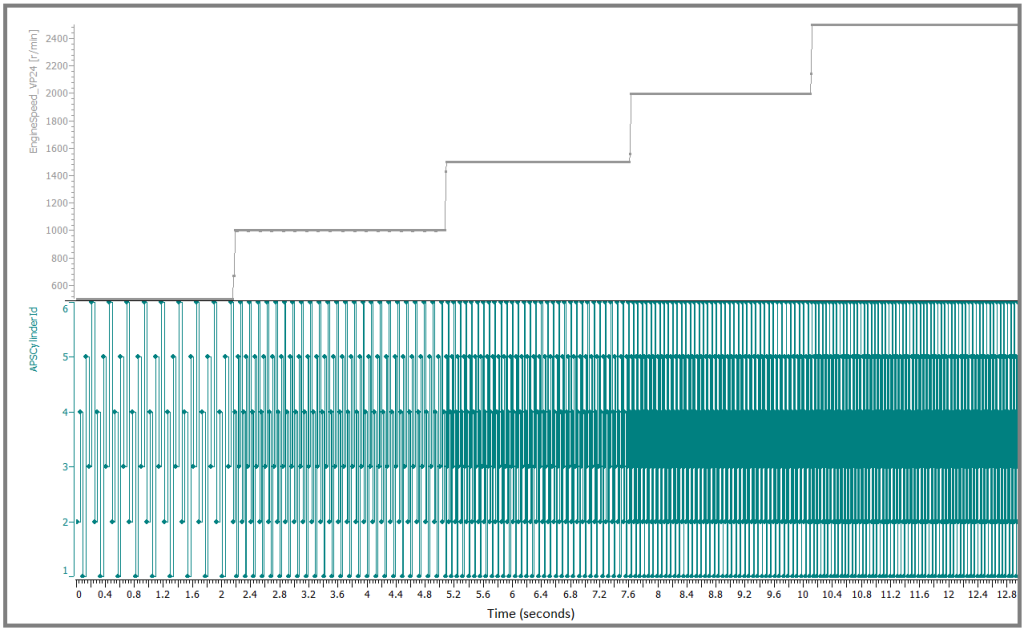

Figure 3.10: Cylinder ID and Engine speed signals as observed in CANalyzer

The time difference between cylinder firing was calculated for different engine speeds using the Equation 3.1 as shown in Table 3.1. The latency of the received signal was analyzed by comparing with time taken for transmission of the signal and the calculated expected time between cylinder firing.

$$\text{Time difference} = \frac{\text{No of seconds}}{\text{Engine speed (in rpm)} * \text{No of Cylinders per revolution}} \tag{3.1}$$

Table 3.1: The time difference between cylinder firing for different engine speeds

Engine speed (rpm)	**Time difference between cylinder firing (seconds)**
500	0.04
1000	0.02
1500	0.0133
2000	0.01
2500	0.008

The logs for different engine speeds were generated in the CANAlyzer for one minute. The time when the signal is received in the CAN for each cylinder ID is noted with different engine speeds and the signals were plotted using the MATLAB tool as shown in Figure 3.11. Thousand values were taken for each rpm, and a histogram was plotted with the number of occurrences of the signal in the Y axis and the corresponding time taken for signal reception from EECU to CAN in the X axis. The time taken for signal reception from EMS to CAN in X axis is calculated as the time difference between the time of reception of the current signal and the time of reception of the previous signal.

3.4.4 Scheduling in TECU

The process of setting up the scheduling of the filtering software in TECU differs for the two communication systems used.

- For direct-line communication, the scheduling was done by triggering an interrupt in the TECU software whenever a change in Cylinder ID is encountered.
- For CAN communication, the scheduling was done by creating a callback function to trigger the software function in TECU whenever the Cylinder ID signal was received from the EECU.

A histogram of the arrival of the Cylinder ID signal in CAN was plotted as shown in Figure 3.11 and a timing analysis was done. It was noticed that there are two peaks at higher engine rpm of 1500 and 2500. This was due to the limitation with the CAN transmission function triggered in the 1.25ms task at Volvo which causes the CAN signals to be received periodically every 1.25ms even when the signal was generated earlier from the EMS. Apart from this delay, there was also a delay caused due to other factors such as high priority tasks scheduled in CAN and other interfering processes. This delay from other processes can be ignored as the CAN message we configured was a low priority message which causes this deviation. Also, the frequency of occurrence of this was noted to be much lower as seen in the histogram plot when compared to the frequency of occurrence of delay due to the 1.25ms CAN task.

We decided to proceed with scheduling the filtering software using the direct-line method. This was due to the latency observed in the signal received through CAN due to the scheduled internal 1.25ms transmission process and the direct-line method

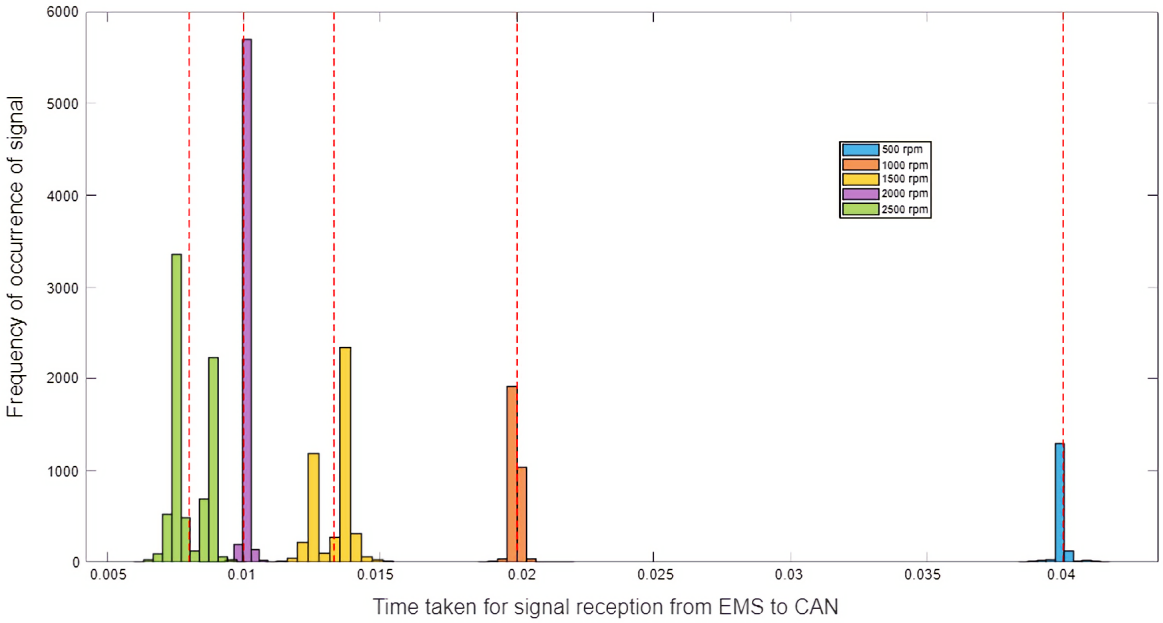

Figure 3.11: Histogram of arrival of the Cylinder ID signal in CAN

was chosen in order to observe and verify the performance improvement without this latency.

3.4.5 Filtering software in TECU

The main focus of this thesis work was to trigger an engine-synchronous software function in the TECU and there was not much focus on developing the most efficient filter for filtering out the noisy sensor measurements. Hence, we have used a simple moving average filter as it helps to filter out the noise from the raw inclination sensor data whenever the cylinder ID changes. The filtering software function was written using a C program, in which a circular buffer of size of six was created, and the new inclination value corresponding to the time of change of the cylinder ID was added to it. Since we have 6 cylinders, we set the buffer size as six for storing data corresponding to each cylinder firing. The circular buffer acts as a FIFO, where the first value inserted to the buffer is deleted whenever the new value is populated and the new value is added to the beginning of the buffer by shifting the values to the right. Each time when the new inclination value is generated, the updated buffer is given as the input to the filtering software function. We have used the moving average filter of length six (representing the six cylinders corresponding to the inclination values at each cylinder firing) to eliminate the noisy measurements in the inclination sensor due the engine vibration.

3.5 Truck testing

3.5.1 Configuration in truck

Since the dedicated direct-line method was implemented in this thesis work, the truck testing was more complex than using the CAN method. The challenging part was changing the ECUs in the truck and installing the ECUs with the direct wire connection. After the successful installation of the ECUs, the air pressure release during the removal of the TECU was restored. Finally the truck was calibrated with the new software uploaded to the EECU and TECU through the existing CAN communication using the tools VISION and Engineering ToolBox.

3.5.2 Data monitoring using VISION

For truck testing, ATI VISION was used to monitor and log the signals when the truck is running. The different signals used for analyzing the inclination sensor data and the output of the filter from EECU and TECU were added and monitored in VISION. The signals used are shown in the Table 3.2.

Table 3.2: Signals monitored in VISION

Signal name	Description
Inclination X raw data	Inclination sensor data from TECU before filtering for the X axis
Inclination Y raw data	Inclination sensor data from TECU before filtering for the Y axis
Inclination Z raw data	Inclination sensor data from TECU before filtering for the Z axis
Inclination X output data	Inclination data for the X axis after applying a moving average filter
Inclination Y output data	Inclination data for the Y axis after applying a moving average filter
Inclination Z output data	Inclination data for the Z axis after applying a moving average filter
Cylinder ID	Cylinder ID (values between 1 to 6) used to determine engine rotation position change

The test data was logged for different inclination and engine rpm values. The resolution for recording the log data in VISION was set to 5 ms which enabled in an analysis of the data.

4

Results

The solution with the direct-line communication method was implemented in a test rig as well as in a running Volvo truck and tested with various engine speeds on different road inclinations. In this chapter, the results obtained from this thesis project are presented and explained.

4.1 Test results from the test rig

The test rig setup was done for the direct-line case and a simulation was done with different speeds and the output signals obtained were analysed. In this section, the results from testing with the test rig are presented.

4.1.1 Direct-line timing analysis

The cylinder ID and the raw inclination data for an engine speed of 600 rpm is taken for both the reference and direct line method to analyse the software behaviour. The time domain data for the cylinder ID change and the raw inclination were plotted for both the reference and direct-line methods as shown in Figures 4.1 and 4.2 and compared. The goal of the thesis is to make the TECU engine-synchronous, this can be verified by checking the raw inclination data change whenever the engine-synchronous cylinder ID signal changes. It can be seen that the raw inclination data was generated engine-synchronously based on cylinder ID for the direct-line setup, whereas the raw inclination data was generated every 10ms irrespective of the engine rpm for the current setup.

The software function was scheduled in the transmission ECU and a timing analysis of the raw inclination value generation was done by recording the timestamps as shown in Figure A.2. A histogram was plotted with the time difference between the raw inclination value generation for both the direct-line communication system as well as using the reference method. The histogram was generated with the test rig setup for three different speeds of 600, 1300 and 2000 rpm which corresponds to a time of 33.33, 15.4 and 10 ms respectively for raw inclination value generation as shown in Figure 4.3. It can be noted from the graph that percentage of the frequency of occurrence mostly falls in the expected time. There is a deviation noted in the timings with the maximum and minimum deviation shown in Table 4.1.1. The maximum deviation was noted to be 265 μs which is noted to be negligible when compared to the maximum deviation observed using the CAN method.

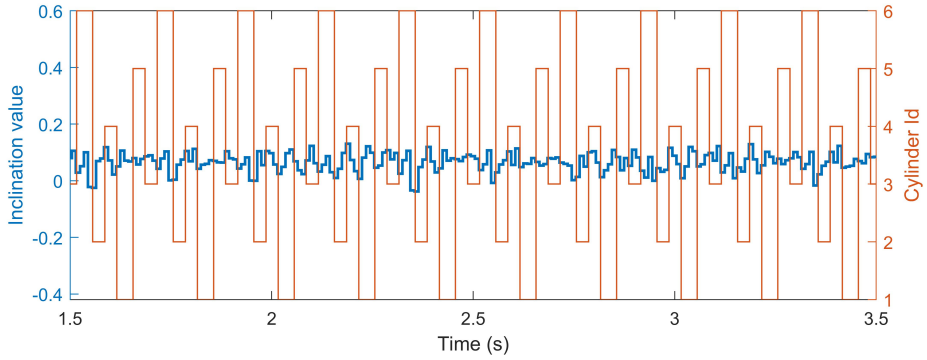

Figure 4.1: Raw inclination value generated for 600 rpm at a time interval of 10 ms for reference setup

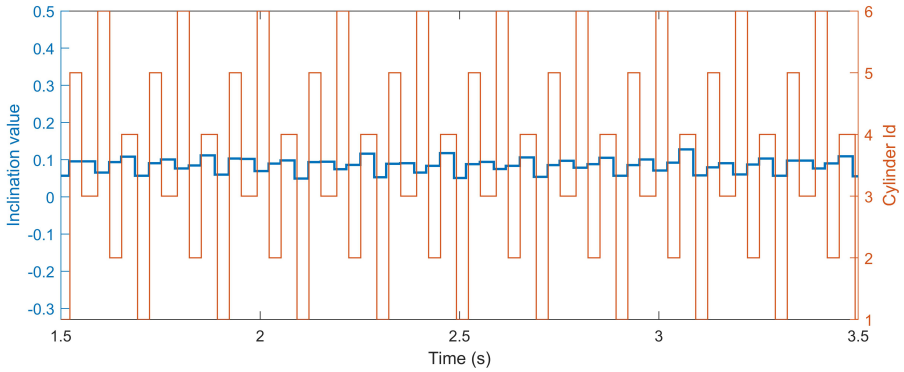

Figure 4.2: Raw inclination value generated for 600 rpm engine-synchronously for directline setup

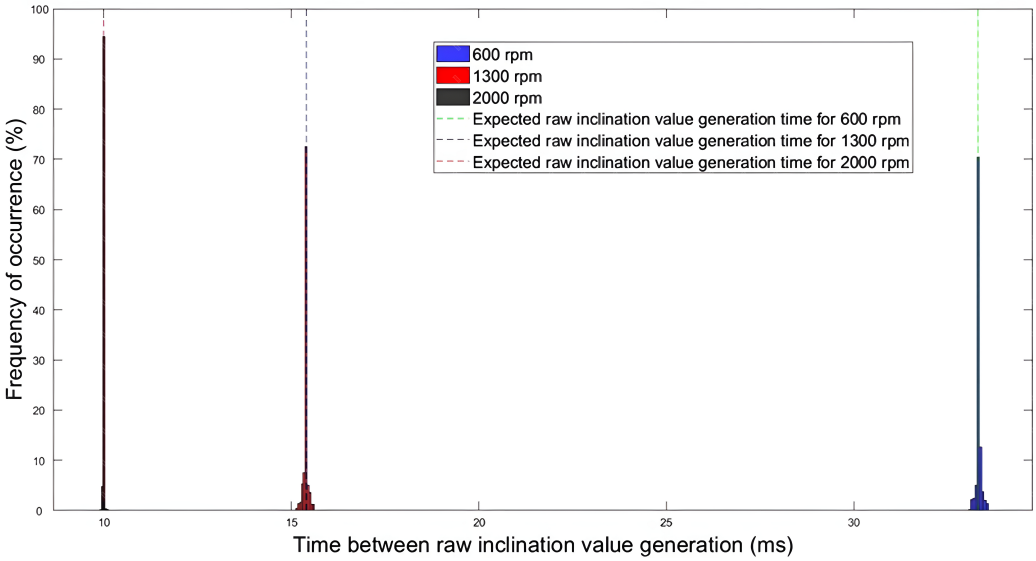

Figure 4.3: Histogram of raw inclination value generation using the direct-line method

Table 4.1: Maximum deviation from expected time of generation of raw inclination values (ms)

Engine speed (rpm)	Expected time of generation of raw inclination values	Maximum deviation
600	33.33	0.258
1300	15.4	0.265
2000	10	0.241

4.1.2 Filtering software

The time domain and frequency domain data of the raw and filtered inclination data was plotted for 600 rpm as shown in Figure 4.4. The filtering of the sensor vibrations would not be verified in the test rig as only the crank and cam rotation could be simulated, but it was not possible to simulate the vibration caused due to the fuel combustion in the cylinders. This is evident by looking at the frequency domain plot which does not show any engine-synchronous noise peaks in the raw inclination data.

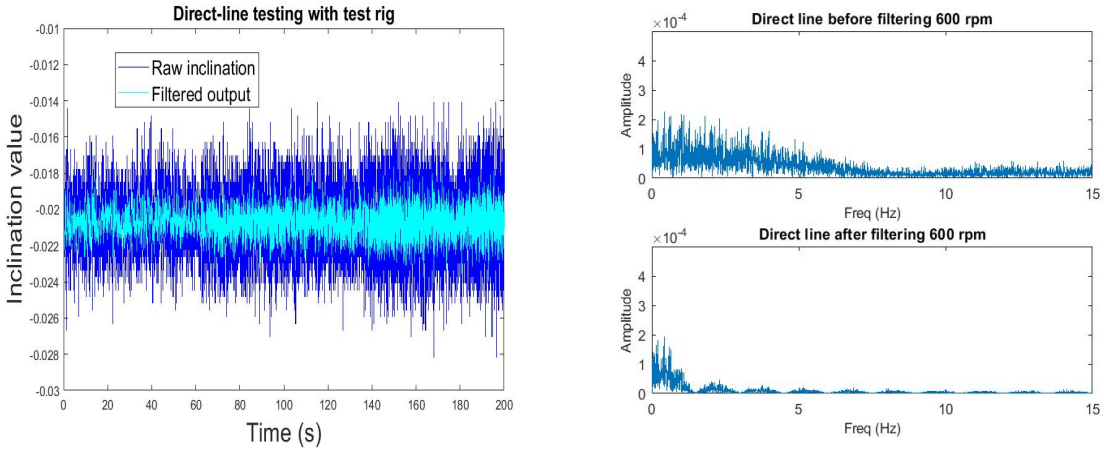

Figure 4.4: Time domain plot at 600, 1300 and 2000 rpm (left) and frequency domain plot at 600 rpm (right) for direct-line method using test rig

4.2 Test results from truck testing

In order to test the engine-synchronous software function with the actual vibrations, the ECUs used for the test rig were mounted on a truck and tested with same three speeds of 600, 1300 and 2000 rpm as done in the test rig.

For analysing the filtering software, we chose the direct-line method as it was able to schedule the software engine-synchronously and has a very low deviation from the expected scheduling time when compared to CAN.

The software was then implemented in the truck, the logs were obtained and the time-domain plot and the frequency domain plot for direct-line were generated as

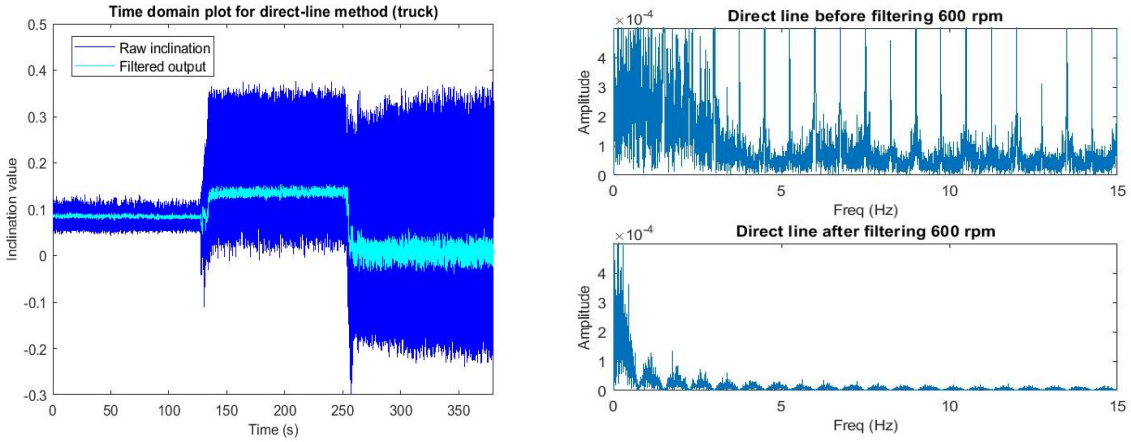

Figure 4.5: Time domain plot at 600, 1300 and 2000 rpm (left) and frequency domain plot at 600 rpm (right) for direct-line method using the truck

shown in Figure 4.5. The noise due to the vibration could be clearly seen in the raw data generated in the time and frequency domain plots of the truck when compared to the time and frequency domain plots of the test rig. Using the direct-line method, it could be seen that the raw inclination value generation as well as the filtering are done engine-synchronously.

It can be seen from the time domain plot that the raw inclination data generated as well as the filtered output has a shift in the amplitude of the signal according to different rpm. This shift is not seen in the time domain plot for the test rig. This issue was not investigated further as the truck testing happened during the end of thesis project and also as this was not in the scope of the project.

The direct-line method uses a moving average filter which takes the last six raw inclination values for the time at which the cylinder ID changes. This in turn results in the effective removal of the vibration noise peaks as seen in the frequency domain plot.

5

Conclusion

Overall, in this thesis project, we were able to work with developing the software as well as doing the hardware implementation for the proof-of-concept solution. Throughout this thesis work, we focused mainly on the communication between the two ECUs and scheduling the software functions in the ECUs.

5.1 Discussion

The communication systems were implemented in test-rig setups using CAN and dedicated direct-line methods and these methods were compared. Comparing the latency of the signal, the direct-line communication had a lower latency which resulted in signal reception closer to real time. The CAN communication setup resulted in a higher latency, due the 1.25 ms transmission task scheduled, as well as due to other interfering high-priority processes scheduled. We also identified another method in CAN where we could achieve more accurate results using the timestamps of cylinder firing, but due to lack of time, we decided to not proceed with it. We decided to proceed with the direct-line method due to the near real-time transfer of data observed. Comparing the robustness of the signals, the CAN set-up is more robust and has the additional function of data recovery and re-transmission in case of data loss. Although there is no option of data recovery and re-transmission using the direct-line implementation, it was not a concern as we could verify that there was no data loss using the direct-line method by logging the timestamps at which the raw data was generated.

The results obtained from the filtering software using the direct-line connection cannot be compared to the reference method, as the reference method is scheduled by a periodic 10 ms task and in direct-line, the scheduling of the filtering software is done engine-synchronously. The moving average filter implemented is not the most optimal solution and there is still scope for improvement in the filtering software used, but other types of filters were not explored in this thesis project, as this was not the main scope.

5.1.1 Critical review of design choices

Though we were able to achieve the expected results using the direct-line method, its main disadvantage was the additional wiring harness in the truck. This method might not work well as a stand-alone communication system, but can be used in

parallel with other existing communication methods such as CAN and Ethernet, as there is only one additional wire required. Hence, the feasibility of implementing the direct-line communication in a conventional truck needs to be verified by analysing the benefits of this solution in terms of drivability, fuel efficiency and other factors.

5.1.2 Ethical considerations

In this thesis project, full consent was obtained from Volvo GTT for the content included in this report, while ensuring adequate level of confidentiality of the data, and making sure that the primary data collection used for the analysis of the results were accurate. Also, for the test onboard a moving truck, the testing was carried out in a safe environment with prior consent from the participants.

5.2 Conclusion

Overall, we implemented the dedicated direct-line methodology to evaluate the software functionality as discussed above. We were able to verify that the software function in the TECU was triggered engine-synchronously. To verify and analyze the outcome of our approach, we implemented and tested it in a running Volvo truck. We have concluded that it is possible to develop a communication system to schedule software functions in the transmission control unit engine-synchronously. This conclusion is based on the result obtained from the previous section, which indicates that reading inclination sensor in the transmission control unit can be done engine-synchronously. Also, we were able to generate the inclination data by eliminating the vibration noise from the inclination sensor data engine-synchronously.

5.3 Future scope

The below steps identified could be undertaken at Volvo GTT or by researchers and students as prospective future work.

- We identified another method that could be used using the CAN protocol to schedule software functions engine-synchronously in the TECU. Instead of sending the Cylinder ID signal through the CAN bus, the timestamp of Cylinder firing could be saved in the software function in the engine ECU and then transmitted via CAN to the TECU and use the difference in the timestamps in the filtering software to get the correct raw inclination samples and filter them. However, due to lack of time, we were not able to verify this method and this can hence be done as a prospective future work.
- Another option that could be explored as a future work is by using other communication methods such as Ethernet for engine-synchronous communication between ECUs.
- Another improvement to this project that could be taken up as future work, is to use other filtering options to verify the most optimal filtering solution to filter the vibration noise engine-synchronously in the TECU.

Bibliography

[1] M. Kaiser, *Electronic control unit (ECU)*. Wiesbaden: Springer Fachmedien Wiesbaden, 2015, pp. 254–259. [Online]. Available: https://doi.org/10.1007/978-3-658-03964-6_16

[2] T. Michalsky and M. Büdenbender, "Testing transmission ecus with integrated sensors," *AutoTechnology*, vol. 1, pp. 62–65, May. 2001.

[3] J. Chen and G. Tan, "Study of anti-lock brake system control strategy in automobile," in *Advances in Computer Science, Environment, Ecoinformatics, and Education*, S. Lin and X. Huang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 458–465.

[4] M. Hilgers and W. Achenbach, *The Fuel System and Fuel Injection*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2021, pp. 25–29. [Online]. Available: https://doi.org/10.1007/978-3-662-60857-9_5

[5] E. Münch, H. Bestmann, C. Lovell, S. Pollmeyer, M. S. Khaniki, and D. Schulte, "Vehicle motion control layer – a modular abstraction layer to decouple ADAS from chassis actuators," in *9th International Munich Chassis Symposium 2018*, P. Pfeffer, Ed. Wiesbaden: Springer Fachmedien Wiesbaden, 2019, pp. 177–190.

[6] C. Chenet, A. Savino, and S. Di Carlo, "Using analog scrambling circuits for automotive sensor integrity and authenticity," Feb. 2022.

[7] S.-W. Lee, W.-K. Choi, and D.-J. Park, "Fast: An efficient flash translation layer for flash memory," in *Emerging Directions in Embedded and Ubiquitous Computing*, X. Zhou, O. Sokolsky, L. Yan, E.-S. Jung, Z. Shao, Y. Mu, D. C. Lee, D. Y. Kim, Y.-S. Jeong, and C.-Z. Xu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 879–887.

[8] *SRAM Cell Stability: Definition, Modeling and Testing*. Dordrecht: Springer Netherlands, 2008, pp. 39–77. [Online]. Available: https://doi.org/10.1007/978-1-4020-8363-1_3

[9] *Using the Internal EEPROM Memory*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 107–113. [Online]. Available: https://doi.org/10.1007/3-540-28308-0_8

[10] T. T. Tran, *Printed Circuit Board (PCB) Layout*. Boston, MA: Springer US, 2010, pp. 187–194. [Online]. Available: https://doi.org/10.1007/978-1-4419-6309-3_10

[11] G. O'Regan, *Microprocessor*. Cham: Springer International Publishing, 2018, pp. 183–186. [Online]. Available: https://doi.org/10.1007/978-3-030-02619-6_38

[12] F. Nouvel, W. Gouret, P. Mazério, and G. Zein, *Automotive Network Architecture for ECUs Communications*, Apr. 2009, pp. 69–90.

[13] S. Tuohy, M. Glavin, C. Eising, E. Jones, M. Trivedi, and L. Kilmartin, "Intra-vehicle networks: A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, pp. 1–12, Jan. 2014.

[14] K. H. Johansson, M. Törngren, and L. Nielsen, *Vehicle Applications of Controller Area Network. In: Hristu-Varsakelis D., Levine W.S. (eds) Handbook of Networked and Embedded Control Systems. Control Engineering. Birkhäuser Boston*, 2005, pp. 741–765. [Online]. Available: https://doi.org/10.1007/0-8176-4404-0_32

[15] Ixia, "Automotive ethernet: Enabling the future of autonomous driving - white paper www.ixiacom.com," Mar. 2020.

[16] R. Shaw and B. Jackman, "An introduction to flexray as an industrial network," in *2008 IEEE International Symposium on Industrial Electronics*, Aug. 2008, pp. 1849–1854.

[17] A. Grzemba, *MOST: The Automotive Multimedia Network*. Franzis, 2011.

[18] M. Ruff, "Evolution of local interconnect network (LIN) solutions," in *2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No.03CH37484)*, vol. 5, Nov. 2003, pp. 3382–3389 Vol.5.

[19] Reshimabai and H. Phaneendra, "Automotive inter ecu communication," *International Journal for Scientific Research and Development 3.3*, pp. 3248–3249, 2015.

[20] M. Törngren, "A perspective to the design of distributed real-time control applications based on can," *Proceedings 2nd International CAN in Automation Conference*, 1995.

[21] S. Halder, M. Conti, and S. K. Das, "Coids: A clock offset based intrusion detection system for controller area networks," in *Proceedings of the 21st International Conference on Distributed Computing and Networking*, ser. ICDCN 2020. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3369740.3369787

[22] J. Guerrero-Ibañez, S. Zeadally, and J. Contreras Castillo, "Sensor technologies for intelligent transportation systems," *Sensors*, vol. 18, p. 1212, Apr. 2018.

[23] (2015) Infineon Technologies AG TC29x B-step User Manual. [Online]. Available: https://www.infineon.com/cms/en/product/microcontroller/32-bit-tricore-microcontroller/32-bit-tricore-aurix-tc2xx/aurix-family-tc29xtx

[24] J. Sun, *Pulse-Width Modulation*. London: Springer London, 2012, pp. 25–61. [Online]. Available: https://doi.org/10.1007/978-1-4471-2885-4_2

[25] S.-L. Chen, T.-K. Chi, M.-C. Tuan, C.-A. Chen, L.-H. Wang, W.-Y. Chiang, M.-Y. Lin, and P. A. R. Abu, "A novel low-power synchronous preamble data

line chip design for oscillator control interface," *Electronics*, vol. 9, no. 9, p. 1509, Sep. 2020, doi: 10.3390/electronics9091509.

[26] M. E. Frerking, *Speech Processing*. Boston, MA: Springer US, 1994, pp. 490–547. [Online]. Available: https://doi.org/10.1007/978-1-4757-4990-8_9

[27] G. S. Moschytz, *The Sampling Theorem and Aliasing*. Cham: Springer International Publishing, 2019, pp. 381–397. [Online]. Available: https://doi.org/10.1007/978-3-030-00096-7_15

[28] K. S. Thyagarajan, *FIR Digital Filters*. Cham: Springer International Publishing, 2019, pp. 245–312. [Online]. Available: https://doi.org/10.1007/978-3-319-76029-2_7

[29] K. Matsuzawa and T. Ishihara, "Simulation of flash memory programming characteristics," in *Simulation of Semiconductor Processes and Devices 2001*, D. Tsoukalas and C. Tsamis, Eds. Vienna: Springer Vienna, 2001, pp. 266–269.

A

Appendix

A.1 Hardware test rig setup - CAN communication method

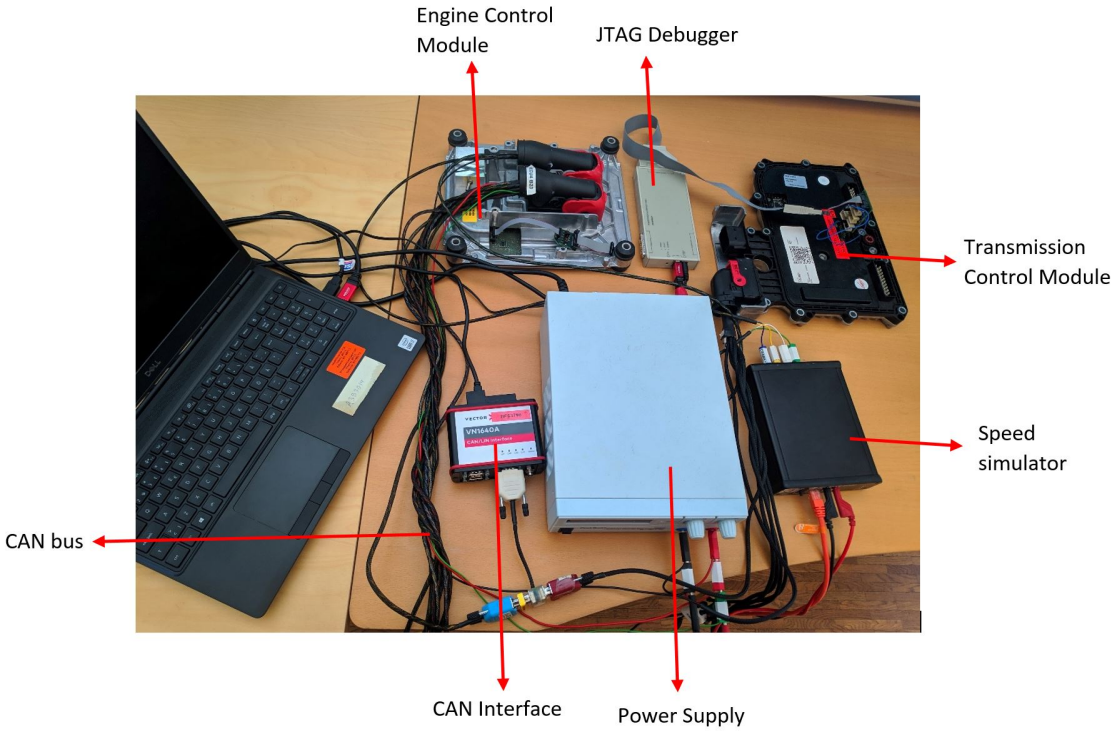

Figure A.1: Hardware test rig setup for CAN communication method

A.2 Using VISION to log the timestamps for timing analysis of the direct-line method

Figure A.2 shows the time domain plot of the direct-line method in test rig using VISION. It shows the three different speeds used for the testing and the timestamp at which the raw data is generated logged and plotted in separate graphs. The timestamps were then used to do the timing analysis for the direct-line method.

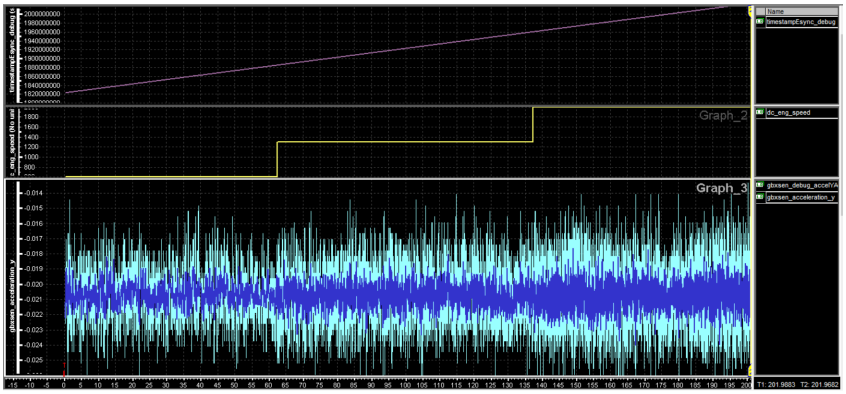

Figure A.2: Timing analysis of direct-line method in test rig using VISION

A.3 Some challenges faced during the thesis work

Some of the challenges we faced during the thesis work was the availability of the truck for testing and mounting and dismounting the ECUs from and on to the truck for every round of testing. We had two rounds of testing in the truck as the logs taken the first time could not be analysed as the frequency setting in VISION was not done correctly. This took a considerable amount of time towards the end of the project as the truck had to be booked in advance.