Synchronizing Real-Time Control In Centralized Automotive E/E Architectures

Synchronizing Real-Time Control In Centralized Automotive E/E Architectures

By

Mostafa Ayesh, B.Eng.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements for the Degree

Master of Applied Science

McMaster University

Master of Applied Science (2024) McMaster University
(Software Engineering) Hamilton, Ontario

Title:	Synchronizing Real-Time Control In Centralized Automotive E/E Architectures
Author:	Mostafa Ayesh, B.Eng. (McMaster University)
Supervisors:	Dr. Alan Wassyng & Dr. Richard Paige
Number of Pages:	vii, 45

Abstract

Automotive technologies have been rapidly evolving with the introduction of electric powertrains, Advanced Driver-Assistance System (ADAS) and Over The Air (OTA) upgradability. Existing decentralized architectures are not an optimal choice for these applications due to significant increases in cost and complexity. The transition to centralized architectures enables heavy computation to be delegated to a limited number of powerful Electronic Control Units (ECUs) called domain or zone controllers. The remaining ECUs, known as smart actuators, will perform well-defined and specific tasks, receiving new parameters from the dedicated domain/zone controller over a network. Network bandwidth and time synchronization are the two main challenges in this transition. New automotive standards have been developed to address these challenges. Automotive Ethernet and Time Sensitive Networking (TSN) are two standards that are well suited for centralized architectures.

This thesis presents a synchronization mechanism that leverages these two standards, to establish precise control synchronization between the centralized ECUs and the smart actuators. Additionally, a testing methodology is introduced to evaluate the synchronization performance between the centralized ECU and the smart actuator within a TSN network.

Acknowledgments

I want to express my gratitude to Dr. Alan Wassyng for the opportunity to pursue my Master's degree and his continued support throughout the years. His mentorship has been critical in shaping my academic journey and guiding me through its challenges.

I also extend my thanks Dr. Victor Bandur and Dr. Vera Pantelic for their assistance with my research. Their expertise and invaluable feedback were crucial to this thesis.

Finally, I would like to express my deepest appreciation to my family, especially my parents. Their endless encouragement and unwavering belief in me have been my greatest source of motivation throughout this journey. I truly couldn't have reached this milestone without their support, and for that, I am eternally grateful.

Contents

List of Figures

List of Acronyms

OTA Over The Air

ADAS Advanced Driver-Assistance System

ECU Electronic Control Unit

CAN Controller Area Network

TSN Time Sensitive Networking

MII Media-Independent Interface

PWM Pulse Width Modulation

E/E Electrical and Electronic

IEEE Institute of Electrical and Electronics Engineers

802.1AS Timing and Synchronization for Time-Sensitive Applications

PPS Pulse Per Second

gPTP generalized Precision Time Protocol

GM Grand Master

PHY Physical layer

EMI Electromagnetic Interference

RFI Radio Frequency Interference

FOC Field Oriented Control

MAC Medium Access Control

CPU Central Processing Unit

ADC Analog Digital Converter

SOA Service Oriented Architecture

RAM Random Access Memory

Chapter 1

Introduction

1.1 Motivation

The automotive industry is experiencing a significant architectural shift driven by the rapid evolution of vehicle technologies towards advanced safety systems, electrical powertrains, and connected vehicles. Vehicle features such as Advanced Driver-Assistance System (ADAS) (Advanced Driver Assistance System), electric motor control, battery management systems are designed as real-time control systems that require high computing power and strict timing. Supporting these systems with existing decentralized Electrical and Electronic (E/E) architectures significantly increases cost and complexity [11].

Decentralized E/E architectures have traditionally relied on Controller Area Network (CAN) as their communication network. While CAN is a robust network, its limited bandwidth is insufficient for high-bandwidth applications. Furthermore, CAN is a bus based network, where Electronic Control Units (ECUs) are connected to a single communication channel and only one ECU is allowed to transmit at any given moment.

In contrast, Centralized E/E architectures [12] are designed to consolidate computationally intensive tasks within high performance ECUs while low level hardware control is delegated to less powerful and cost-efficient ECUs. The increased volume of data transmission needed by centralized E/E architectures cannot be satisfied with CAN.

Automotive Ethernet is a high-speed communication network designed to meet the demands of the automotive environment. It offers a significant increase in data rates compared to older networks such as CAN. The enhanced data rates make it a great fit for centralized E/E architectures. To meet the requirements of real-time control systems, Time Sensitive Networking (TSN) [15] is used in conjunction with Automotive Ethernet. TSN is a set of Institute of Electrical and Electronics Engineers (IEEE) standards enabling low-latency and deterministic communication over automotive Ethernet. It allows critical data packets to be transmitted at predetermined timing, making it ideal for high-frequency, high-precision real-time control applications.

Adopting centralized E/E architectures, Automotive Ethernet and TSN, enables the development of advanced automotive systems. The process of migrating from decentralized E/E architectures to centralized E/E architectures introduces many challenges. The most relevant challenge to this work is ensuring synchronization between various ECUs across the vehicle, in order to maintain its performance and safety.

The phenomenon central to this thesis is *clock drift*, which refers to variations in the time measured by each ECU in a vehicle. Clock drift results from slight differences in the electrical and physical clock circuitry of ECUs and can degrade the performance of real-time control systems, potentially leading to failure.

This thesis presents an approach to address this problem using automotive Ethernet in conjunction with the TSN standards. By leveraging TSN compliant hardware to generate synchronization signals, the effects of clock drift can be mitigated, ensuring synchronized real-time control.

Electric motor control was selected for this work, with the goal of achieving synchronized control between a domain controller that executes the algorithms and a smart actuator generating the Pulse Width Modulation (PWM) signals delivered to the motor. Achieving precise PWM control is crucial for maintaining the performance and efficiency of electric motors. Clock drift between smart actuators and the domain controller, as well as between the smart actuators themselves, can affect the timing on of the PWM pulses, degrading the control system's performance.

In this thesis, a method utilizing TSN compliant hardware and software will be proposed to ensure that the timing of the PWM pulses generated by each smart actuator are synchronized to the domain controller.

Additionally, an evaluation method will be introduced and used to assess the performance of the proposed approach. This method involves measuring the drift between the output PWM signals and a reference signal generated from a network synchronized clock. The measured drift will be used to compare the synchronization performance between different configurations.

1.2 Related Work

The need to develop centralized Electrical and Electronic (E/E) architectures to meet the increasing demands of modern automotive technologies is evident in the multitude of white papers written by the automotive industry. More-

over, there is a growing academic literature studying centralized automotive architectures and evaluating enabling technologies and standards [2][5][6][16]. However, despite extensive research focusing on centralized architectures, there is a notable gap in the literature that addresses the challenges in synchronizing real-time automotive control systems.

1.2.1 Centralized Architectures

This thesis proposes a solution utilizing a centralized E/E architecture. Various literature highlight the benefits centralized architectures have over traditional decentralized architectures.

The work by Bandur et al. [5] studies the motivation behind centralized architectures and the supporting technologies that address some of the challenges introduced by centralized architectures. The main supporting technologies discussed in their work related to the work of this thesis are Automotive Ethernet and Time Sensitive Networking (TSN).

Kanajan et al. [10] proposes criteria for evaluating the performance of architectural models for centralized architectures. Control latency, which they define as the end-to-end latency from sensor to actuator, is one of these criteria that is important to this thesis. One of the goals of the proposed synchronization method is to minimize the control latency and improve the performance of the system.

1.2.2 Real-Time Control Synchronization

This section reviews the literature related to real-time control synchronization. The available literature focuses on real-time motor control synchronization in

industrial applications [7][14].

Sorensen, O'Sullivan, and Aaen [14] present a solution for synchronizing multi-axis motion control over real-time networks, although targeting an industrial application. The implementation involves the use of additional external hardware to generate the synchronization signals.

Figure 1.1 illustrates the proposed solution, which includes an external network controller and an I\O event scheduler for each motor. The additional hardware is used to synchronize the algorithms and the hardware signals generated to the motors.

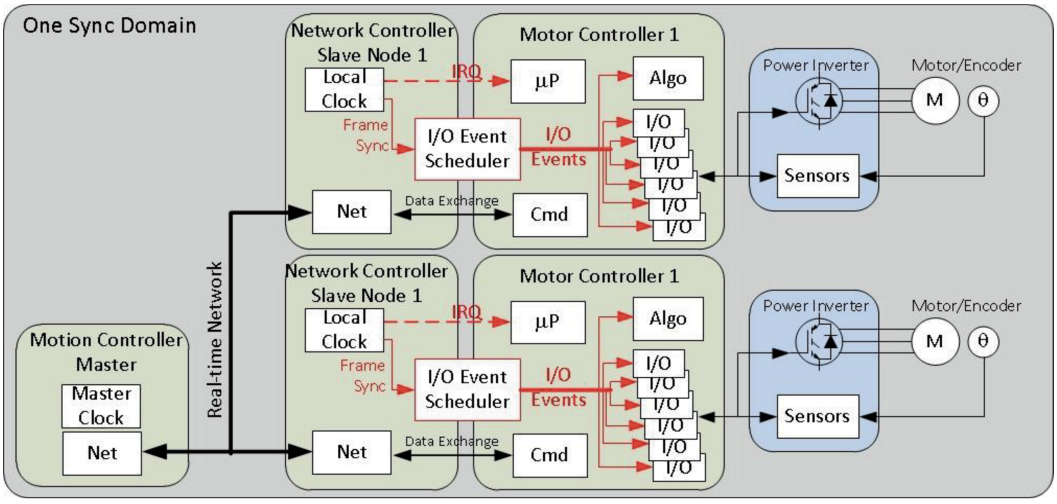


Figure 1.1: I\O event scheduler. From Sorensen, O'Sullivan, and Aaen [14]

Additionally, Dan Burlacu, Mathe, and Teodorescu [7] introduce a solution for synchronizing PWM signals, utilizing EtherCAT's Distributed Clock mechanism. Similar to Sorensen, O'Sullivan, and Aaen [14], this approach requires external hardware for generating synchronization signals.

Unlike the approaches presented by Sorensen, O'Sullivan, and Aaen [14] and Dan Burlacu, Mathe, and Teodorescu [7], which rely on external hardware for generating synchronization signals, the implementation in this thesis does

not require additional hardware. The proposed implementation leverages TSN compliant hardware within the smart actuator to generate the synchronization signals. This approach eliminates the need for external hardware synchronization signals, reducing both the cost and the complexity of the system.

1.3 Contributions

This thesis contributes the following:

- Analyzing existing and emerging hardware and software technologies designed to address the identified challenges including Automotive Ethernet and Time Sensitive Networking (TSN), to identify suitable hardware for synchronized real-time control in centralized Electrical and Electronic (E/E) architectures
- Identifying the challenges and requirements associated with migrating real-time control applications from decentralized E/E architectures to centralized E/E architectures.
- Proposing an approach for synchronizing real-time control applications within centralized E/E architectures. This approach leverages Institute of Electrical and Electronics Engineers (IEEE) 802.1 TSN standards as well as Ethernet hardware generated Pulse Per Second (PPS) signals to synchronize real-time control across a vehicle's network.
- Introducing a methodology for evaluating the performance of real-time control synchronization in centralized E/E architectures. The methodology uses clock drift as a metric to measure synchronization performance.

- Demonstrating the synchronization approach and evaluating its performance using the evaluation methodology. The demonstration includes an implementation using specialized hardware and software platforms.

1.4 Outline

The thesis is structured as follows:

- **Chapter 2** contains the background information relevant to this work. This includes the architectural changes that enable advanced automotive applications, as well as technologies that support these architectures such as Automotive Ethernet and Time Sensitive Networking (TSN). It further details
- **Chapter 3** proposes a method for synchronizing Pulse Width Modulation (PWM) signals generated by smart actuators.
- **Chapter 4** is dedicated to evaluating the proposed method.
- **Chapter 5** contains conclusions and directions for future work.

Chapter 2

Background

2.1 E/E architectures

2.1.1 Electronic Control Unit (ECU)

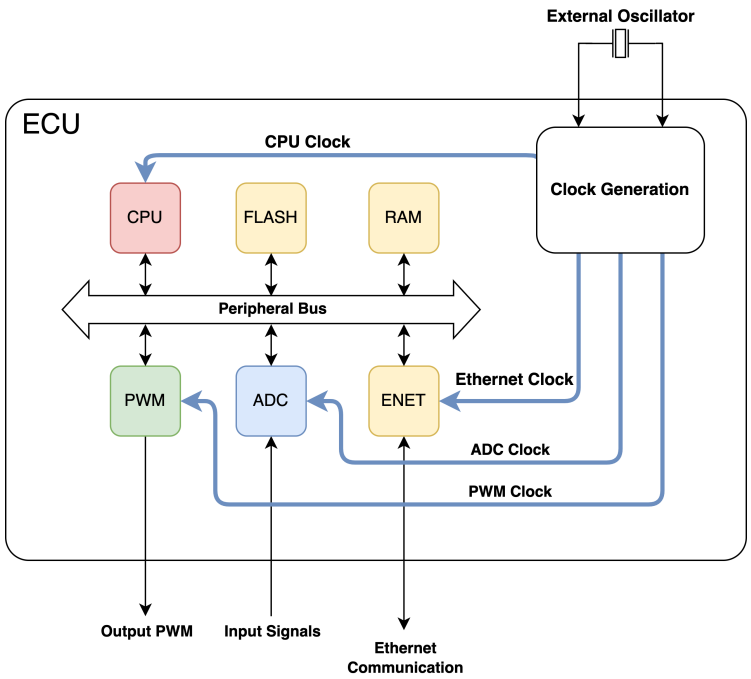


Figure 2.1: Electronic Control Unit (ECU)

An ECU is an embedded system that is responsible for monitoring and controlling different vehicle components such as headlights and electric motors. The main component of an ECU is the Central Processing Unit (CPU), which handles all the calculations and decision making within the ECU. The CPU communicates with the other components of the ECU over a peripheral bus.

In addition to the CPU, an ECU also includes Flash for storing the software as well as Random Access Memory (RAM) for temporary data storage during runtime.

An ECU also contains various peripherals designed to interact with specific vehicle systems. These may include:

- **Pulse Width Modulation (PWM)** which generates precisely timed electrical pulses configurable by the CPU
- **Analog Digital Converter (ADC)** responsible for sampling analog signals and converting them to digital data for processing by the CPU
- **Ethernet**, enabling communication between various ECUs within the vehicle

Another important component of the ECU is clock generation. Clock signals with various frequencies are required to drive each component of the ECU. An oscillator is used by the clock generation component to generate different clocks throughout the ECU. Due to sharing the same oscillator, there is an inherent synchronization between the different components of the ECU.

Figure 2.1 represents a simplified ECU, showing various components as well as how clock signals are connected within the ECU.

2.1.2 Decentralized Architectures

Most vehicles on the road today rely on decentralized Electrical and Electronic (E/E) architectures. These architectures consist of over 100 ECUs [3] each dedicated to specific functions. This approach reduces the complexity of both software & hardware within each ECU but imposes limitations on the processing capabilities [4]

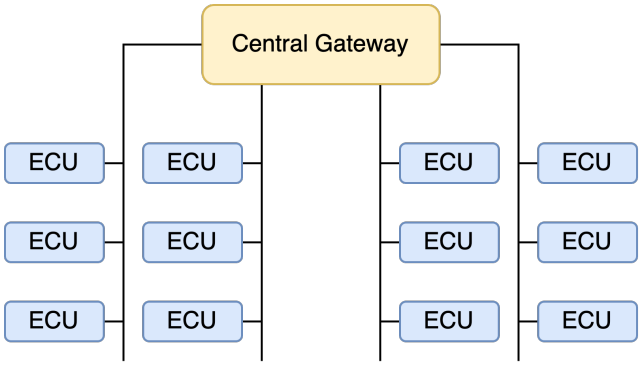


Figure 2.2: Decentralized Architecture

A simplified decentralized architecture diagram can be seen in Figure 2.2. ECUs communicate with each other over a distributed network such as Controller Area Network (CAN). A Central Gateway is used to bridge vehicle's internal networks, providing secure communication between ECUs as well as routing traffic between different networks such as CAN or FlexRay. For high-bandwidth applications, the network used and the central gateway may introduce bottlenecks.

2.1.3 Centralized Architecture

A centralized architecture allows the vehicle to have a small number of a few powerful ECUs capable of more advanced processing than traditional ECUs.

These processing tasks could include advanced arithmetic, hardware decoding/encoding, machine learning inference, etc.

There are different types of centralized architectures. The two main types of centralized architectures are domain architectures and zonal architectures.

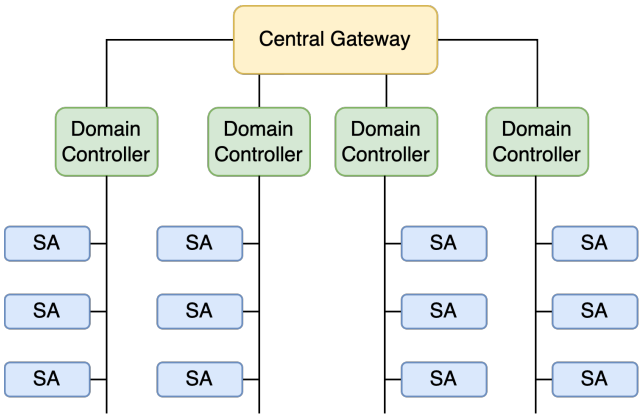


Figure 2.3: Domain Centralized Architecture

Figure 2.3 demonstrates an example Domain Centralized Architecture. It organizes a vehicle's architecture into functional domains such as Advanced Driver-Assistance System (ADAS), Infotainment, etc. Each domain consists of a domain controller and one or more smart actuators (SA).

In a Zone-oriented centralized architecture, shown in Figure 2.4, a vehicle is divided into zones by the physical location of the ECUs. It consists of zone controllers, each connected to a group of smart actuators (SA). There is also a high performance central vehicle compute that can perform complex computations for all the zones in the vehicle. A central gateway routes and manages the traffic between the Zone controllers and the Central Gateway. The main benefit is a simpler architecture due to the reduced number of ECUs and shorter cable length for networking [1].

A domain centralized architecture is used as a basis for this thesis, detailed

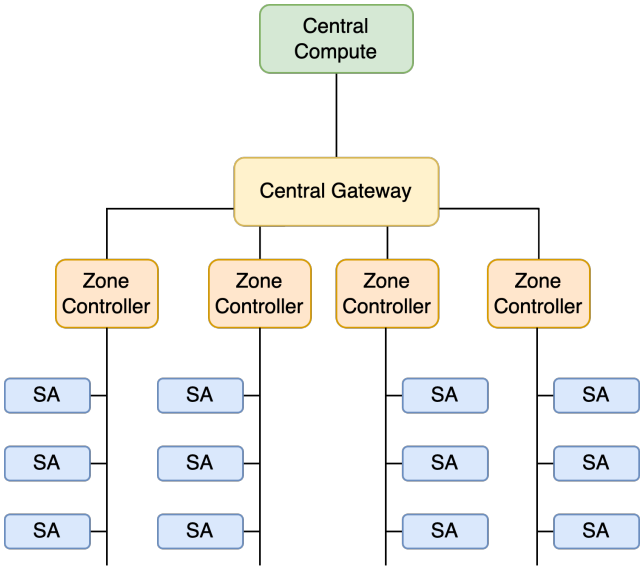


Figure 2.4: Zone Architecture

information about such architecture will be discussed in 2.2.1.

The centralization of processing performed by the powerful ECUs requires a high-bandwidth, low-latency communication network to ensure adequate data flow throughout the vehicle's network.

2.1.4 Inter-ECU Communication

Communication between ECUs is vital for the overall E/E system. The choice of communication network is influenced by the architecture. Traditional automotive network technologies including CAN and FlexRay do not satisfy the bandwidth or latency required for a centralized architecture. Automotive Ethernet is being adopted to address some of those challenges when migrating to centralized architectures.

2.1.4.1 Automotive Ethernet

Ethernet enables high bandwidth and low latency networks [3] used for various applications. However, traditional Ethernet has physical limitations that make it unsuitable for use in automotive applications.

Automotive Ethernet is designed to address these physical limitations [13] in a number of ways:

- Meets Electromagnetic Interference (EMI) and Radio Frequency Interference (RFI) requirements for automotive applications
- Reduces the weight of the harness by using single twisted pair as opposed to two twisted pairs used by traditional Ethernet

Furthermore, the greatest advantage of Automotive Ethernet over existing automotive networks (such as CAN, CAN-FD, FlexRay) is the significant increase in the data transmission rate. The result is increased bandwidth and reduced latency that allows real-time control of the network.

Technology	**Maximum Data Transmission Rate**
Auto Ethernet (100BASE-T1)	100 Mbps
Auto Ethernet (1000BASE-T1)	1000 Mbps
CAN	1 Mbps
CAN-FD	8 Mbps
FlexRay	10 Mbps

In addition to the improvements in data transmission rates, Automotive Ethernet enables more complex network topologies. Existing CAN, CAN-FD and FlexRay networks are limited to bus topologies. A bus topology is unsuitable for centralized architectures, because it allows only one device

to transmit at a time, reducing overall bandwidth and offering limited redundancy if a link becomes non-functional. In contrast, Automotive Ethernet uses network switches, facilitating advanced traffic control that allows concurrent transmission from multiple ECUs and supporting Service Oriented Architectures (SOAs) as well as Over The Air (OTA) capabilities.

A modern vehicle consists of multiple real-time control systems, performing various functionalities including but not limited to propulsion control, stability control, safety, and driver assistance. These systems require low latency and precise timing to ensure functionality.

Traditional Ethernet's traffic model is best-effort, meaning data is delivered without any timing or order guarantees. This model is unsuitable for real-time automotive systems. Time Sensitive Networking (TSN) standards address these challenges by enabling precise time synchronization and deterministic traffic scheduling [16].

2.1.4.2 Time Sensitive Networking (TSN)

TSN [15] is a set of standards developed by the IEEE 802.1 Working Group. These standards aim to provide deterministic and precise time synchronization and traffic scheduling capabilities in Ethernet networks.

Time synchronization is a key aspect of TSN, as it ensures that all devices in the network have a common and accurate view of time. This is essential for real-time applications that rely on precise timing, such as industrial automation and control systems. When devices in the network are synchronized, their events can be coordinated in a deterministic matter.

Traffic scheduling is another important aspect of TSN. It allows traffic prioritization and bandwidth allocation to specific traffic flows based on real-

time requirements. This ensures that critical traffic is delivered on time.

2.1.4.3 Time Synchronization

802.1AS [8], also known as generalized Precision Time Protocol (gPTP), is a protocol for synchronizing the clocks on networked devices. It allows devices on a network to communicate with high time precision, making it ideal for applications where accurate timing is critical, such as real-time control.

gPTP uses a master-slave model, defined as follows:

- **Grand Master (GM)**: Serves as the reference clock for the network time, distributed to the other devices on the network. The Grand Master (GM) is selected based on its precise physical clock, ensuring synchronization across the network.
- **Slave**: Receives timing information from the GM and adjusts its clock to maintain synchronization. The synchronization is performed using special gPTP packets carrying time information.

2.1.4.4 Traffic Scheduling

802.1Qbv [9] is a standard that defines the timing behavior of traffic in a network. It allows devices to prioritize time-sensitive traffic and ensure that it is delivered on time, making it suitable for applications that require predictable and deterministic behavior, such as industrial automation and control systems.

802.1Qbv defines a mechanism for scheduling traffic on a network, allowing a network schedule to specify the precise time at which a frame should be transmitted and to assign different priorities to different frames. This allows

time-sensitive traffic to be given higher priority and to be delivered on time, even in the presence of other traffic on the network.

2.2 Problem Definition

2.2.1 Domain Centralized Architecture

The processing of tasks for each vehicle function in a decentralized architecture is performed by a dedicated ECU specifically built for that function. Recent advancements in automotive technologies required the introduction of complex processing tasks. Such tasks require a significant increase in processing power and are more suitable to be carried out by an Electronic Control Unit (ECU) much more powerful than a traditional one. Additionally, these tasks require low latency and precise timing to ensure optimal performance and reliability.

A domain centralized architecture divides a vehicle into multiple logical domains, each encompassing an aspect of vehicle functionality (eg. powertrain, chassis). A domain consists of a **domain controller** and multiple **smart actuators** as seen in figure 2.3.

A **domain controller** in a centralized architecture is a powerful ECU doing most of the heavy processing tasks. By delegating these tasks to a domain controller, the compute requirements for the other ECUs in the domain which are known as **smart actuators** are reduced.

A **smart actuator** is an ECU responsible for the signal-level interactions with a vehicle's sensors (eg. radar, ultrasound, etc.) and/or actuators (eg. motor, headlights, etc.). It may collect measurements and send the results to the domain controller for processing. It may also receive commands from the

domain controller and alter its output signals accordingly.

2.2.2 Real-Time Control

Real-time hardware control applications consist of three major components: signal acquisition, processing and signal generation.

The control process starts with signal acquisition, typically sampling input analog and/or digital hardware signals. The data collected is used to calculate new control parameters. Finally, hardware output is reconfigured with the new control parameters. There are timing requirements for each step of the control process. In traditional automotive architectures, the control process is integrated into a single ECU and timing is enforced locally.

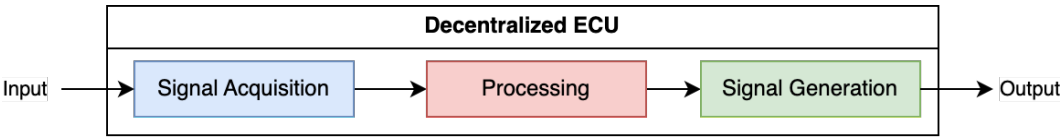


Figure 2.5: Decentralized Architecture ECU

Figure 2.5 shows an ECU in a decentralized Electrical and Electronic (E/E) architecture. All steps of the control process are carried out by the ECU, starting with the input signal acquired and ending with the output signal generated. The hardware and software components of the ECU are responsible for enforcing the timing of all the steps of the control process.

The control process in a centralized architecture may be divided between a domain controller and one or more smart actuators.

Figure 2.6 shows three ECUs in a domain centralized E/E architecture. The two smart actuators perform the input signal acquisition task. The acquired signals are communicated to a domain controller which performs the required

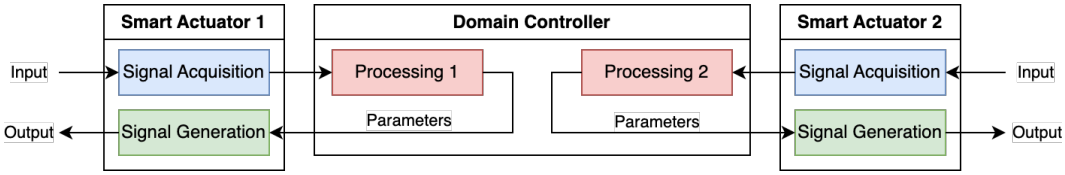


Figure 2.6: Domain Centralized Architecture

processing. The parameters calculated from processing are communicated back to the smart actuators. The smart actuators generate output signals using the calculated parameters.

The domain controller and the smart actuator must be time synchronized to enforce the timing requirements of the application.

2.2.3 Clock Drift

Clock drift is a phenomena where two clocks become de-synchronized over a period of time. This is due to one of the clocks running slightly faster or slower than the other clock.

Clocks are generated from the output of an oscillator. An oscillator output frequency may also deviate from the intended value over time caused by a number of factors, including temperature changes, aging of the components, and variations in the power supply. These deviations result in clock drift.

In addition to that, each ECU uses its own oscillator, and they may vary slightly in frequency.

Figure 2.7 shows an example of two oscillators running at slightly different frequencies. Although the variance may look minimal, it can cause instability issues in high precision control applications such as electric motor control.

Figure 2.8 illustrates the phenomenon where two clocks configured to run at identical frequency, drift apart. Small variations in the frequencies of the

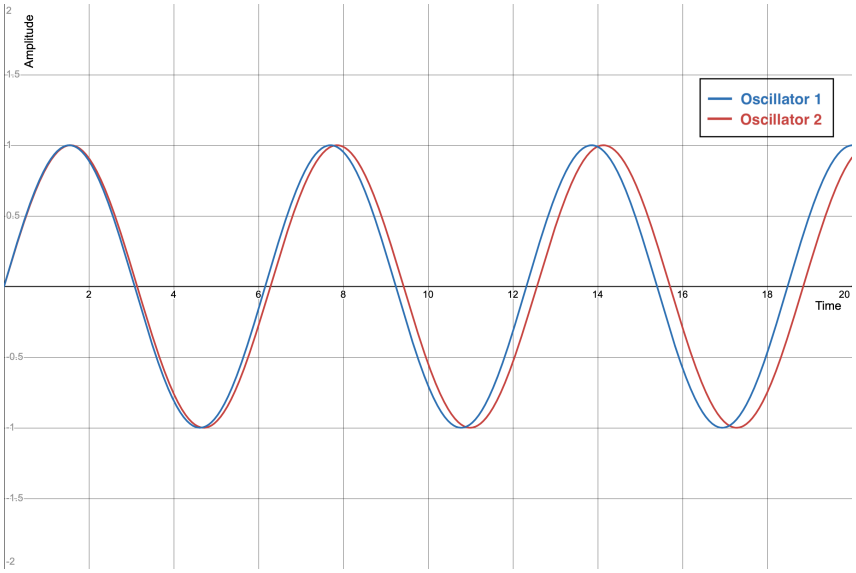


Figure 2.7: Oscillator frequency deviations

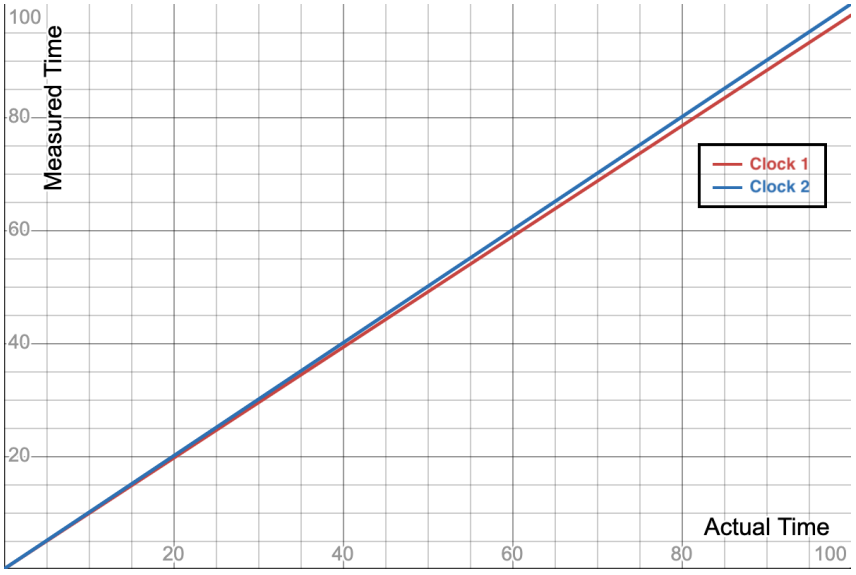


Figure 2.8: Clock Drift

two clocks can cause one clock to run faster or slower than the other, resulting in the time measured by one clock to be shorter or longer than the other clock. Peripherals rely on these clocks to generate interrupts, control hardware signals, etc. If clocks become de-synchronized, the events generated by peripherals become de-synchronized. Therefore, the clocks have to be periodically

synchronized to minimize the amount of drift, reducing jitter and ensuring synchronized control.

2.2.4 Peripheral Event Generation

Peripherals generate periodic events using a set of counters and configurable compare values. These events can be interrupts, hardware control signals, etc. The counters are periodically incremented using a signal generated by a clock source. A counter is continuously checked against one or more configurable compare values. When the counter exceeds a configurable compare value an event is generated and the counter is reset. A compare value is configured to store the time period of an event. Figure 2.9 shows how peripheral events are generated.

Clock drift causes inaccuracies in the time measured by these counters leading to de-synchronized events. Correcting clock drift is critical to ensuring these events are synchronized and are generated at a known network time. Time Sensitive Networking (TSN) can be utilized to address this issue maintaining synchronized control across the network.

2.3 TSN Hardware

Special hardware is required to support Time Sensitive Networking (TSN) standards including time synchronization and traffic scheduling. Figure 2.10 shows an example of a TSN-compliant Ethernet setup. Data is received over Ethernet by the Physical layer (PHY) and transmitted over a standard interface called Media-Independent Interface (MII) to the Medium Access Control (MAC).

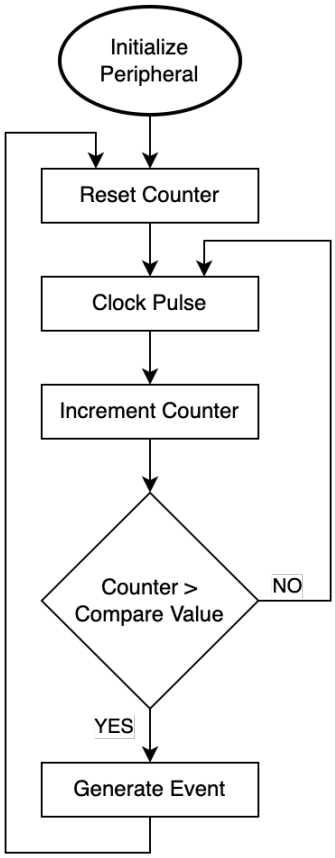


Figure 2.9: Peripheral Event Generation

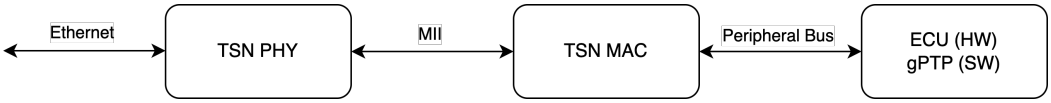


Figure 2.10: TSN Hardware Setup

To enable time synchronization, a TSN-enabled MAC and/or PHY hardware is required. When a frame is received over Ethernet, the reception time is recorded by a TSN-enabled PHY or MAC. This timestamp is added to the incoming frame, ensuring the timing information is available throughout the communication process. The generalized Precision Time Protocol (gPTP) software stack extracts the timestamp from each incoming frame and updates the necessary hardware to achieve time synchronization.

2.4 TSN Time Synchronization

Maintaining a shared timestamp between all Electronic Control Units (ECUs) in a vehicle enables synchronized real-time control. In addition to TSN-compliant hardware, a generalized Precision Time Protocol (gPTP) implementation is required to achieve time synchronization.

- Sharing a timestamp between all the devices on the network. The gPTP Grand Master (GM) communicates its current timetamp to all gPTP slaves over Ethernet.
- Minimizing drift between the clocks of all the devices on the network.

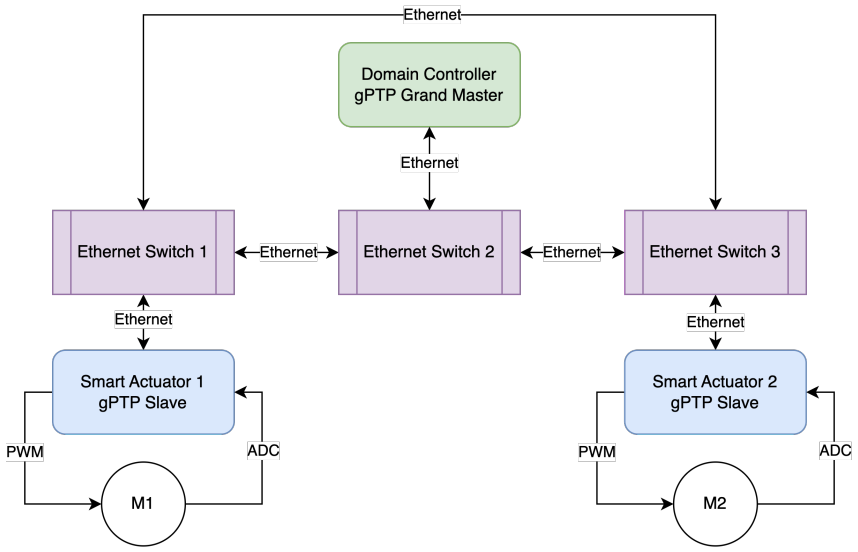


Figure 2.11: gPTP setup

Figure 2.11 demonstrates an example network with one GM and two slaves. In this network, the domain controller was selected as the gPTP GM and the smart actuators are gPTP slaves. The GM uses a good accuracy oscillator to generate a precise clock and shares the timestamp information with the rest

of the network. Every device is connected to the network through an Ethernet connection and all the traffic in the network is routed through Ethernet switches. The Ethernet switches also need to support gPTP in order to achieve time synchronization between gPTP slaves the and the GM.

2.4.1 Reference Time

The gPTP GM broadcasts synchronization messages across the network. These messages contain the information required for determining the current timestamp on the GM. The information is extracted from the messages and stored in the internal timestamp counters on each gPTP slave. When a gPTP slave is sending data over Ethernet, it attaches the timestamp from these counters to the outgoing Ethernet frame. In addition to that, the application running on the gPTP slave can access these counters to determine the current GM timestamp. By knowing the current GM time, the application can schedule future events to occur at a precise time known to other devices on the network.

2.4.2 Clock Drift Correction

The second mechanism calculates the ratio between the rate of the slave's clock and the rate of the GM clock. The calculated ratio is used to adjust the increment rate of the slave's internal counters to be closer to the increment rate on the GM. These adjustments correct the clock drift between the slave and the GM.

Chapter 3

Peripheral Synchronization using TSN

3.1 Introduction

Peripheral synchronization is crucial for enforcing timing and performance requirements on real-time control systems in centralized Electrical and Electronic (E/E) architectures. It ensures that any signals generated by the peripheral occur at a predetermined time, which is known to all the Electronic Control Units (ECUs) involved in the control process. Electric motor control was selected as a driver for the methodology due to its strict real-time requirements.

3.2 Electric Motor Control

Electric motor control is a real-time closed-loop system with strict latency requirements. Time determinism is required in each step of the control process.

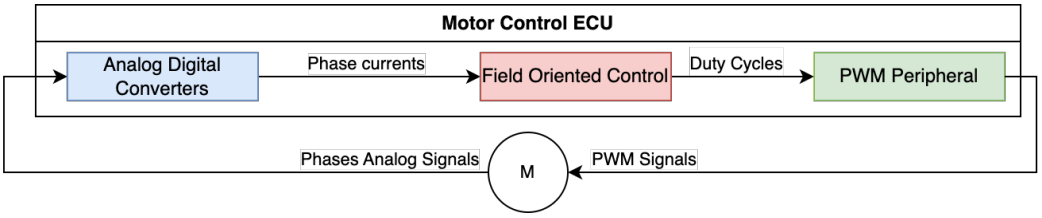


Figure 3.1: Motor Control Electronic Control Unit (ECU)

Electric motor control is handled by a dedicated ECU for each electric motor in the vehicle. An example setup can be seen in Figure 3.1.

Electric motors are operated by sending Pulse Width Modulation (PWM) signals to their internal coils, one signal for each phase of the motor. The duty cycle of each PWM signal controls the output torque generated by the motor. These duty cycles are calculated using Field Oriented Control (FOC).

FOC involves a set of algorithms used to calculate the output duty cycles for each phase of the motor. The inputs to these algorithms are the current motor parameters, including the motor speed, motor angle, and the amount of electrical current of each phase.

3.2.1 Timing Requirements

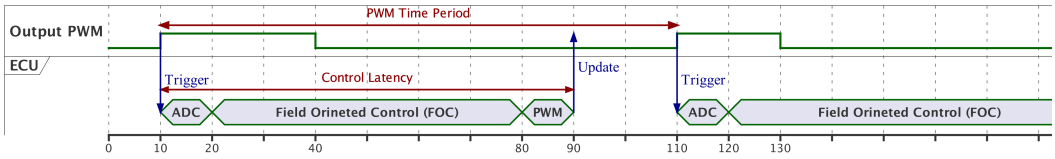


Figure 3.2: Decentralized Motor Control

Figure 3.2 shows an example timing diagram for the motor control process. The process begins with at the start of the PWM time period. A trigger is sent to the Analog Digital Converter (ADC) to start the phase measurements. Once the measurements are ready, they are handed to the FOC algorithms.

The FOC algorithms calculate a new set of duty cycles the PWM signals to the motor's phases. Finally, the PWM output is updated with the new duty cycles. The control process is repeated at the start of the next time period. In order to maintain functional motor control, the sequence needs to be completed before the start of the following PWM time period.

3.2.2 Domain Centralized Motor Control

The FOC algorithms require a considerable amount of computation. These processing requirements drive up the cost of each ECU responsible for motor control. Migrating the execution of the FOC algorithms to a centralized domain controller simplifies the hardware requirements and reduces the cost of the motor control ECUs. It also enables more advanced control algorithms especially for multi motor vehicles.

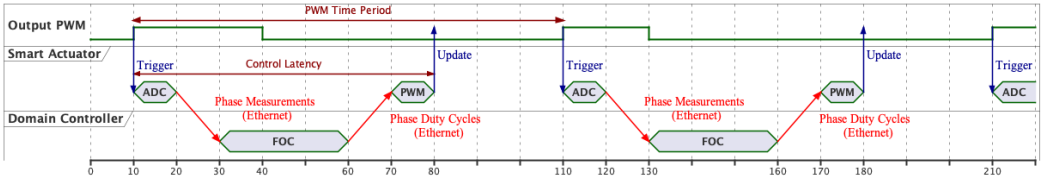


Figure 3.3: Domain Centralized Motor Control

A domain centralized motor control timing diagram can be seen in figure 3.3. When phase measurements are ready, they are transmitted over Ethernet from the smart actuator to the domain controller that performs the FOC algorithms. The calculated duty cycles are transmitted back to the smart actuator that is used to update the output PWM

Timing requirements are enforced locally in decentralized motor control using timers and interrupts. However, in centralized motor control, additional hardware and software are required to meet the same requirements.

3.3 Hardware Requirements

Specialized hardware components and features are required to support the proposed peripheral synchronization method. A Pulse Per Second (PPS) hardware component is required to generate an internal synchronization signal as input to the Pulse Width Modulation (PWM) peripheral. In addition to that, the target PWM peripheral has to support external synchronization.

3.3.1 Pulse Per Second (PPS)

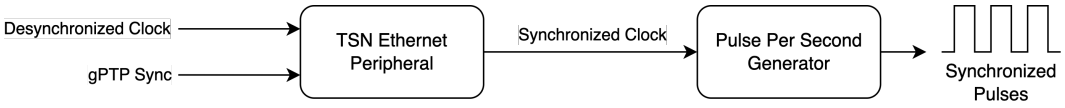


Figure 3.4: Pulse Per Second Clock Generation

PPS is a configurable output signal generated using the Time Sensitive Networking (TSN) enabled hardware's counters. The implementation of generalized Precision Time Protocol (gPTP) ensures that the hardware counters store the Grand Master (GM)'s timestamp by capturing and processing the synchronization broadcast messages received from the GM. By using the synchronized counters, the PPS is synchronized to the GM's clock. Despite its name, the time period of the generated PPS signal can be configured to be greater than or less than one second, enabling high precision time synchronization. Therefore, the PPS output signal can be used as a synchronization source for other peripherals.

The hardware capability to generate output signals or interrupts derived from the synchronized gPTP clock is required to achieve synchronization. The PPS should have a configurable frequency up to the PWM peripheral's maxi-

mum required frequency for the application and cannot be greater, otherwise it will lead to generating signals with shorter than required time period.

$$f_{PPS} \leq f_{PWM}$$

3.3.2 PWM Peripheral Synchronization

The PWM peripheral should support external synchronization. External synchronization can be achieved using external pulses, each pulse resetting the internal time period counters to their initial values. This ensures that the beginning of the PWM period is aligned with a specific timestamp which is shared with other devices on the network. The PWM peripheral must provide a signal input path for these external pulses.

3.4 Control Synchronization

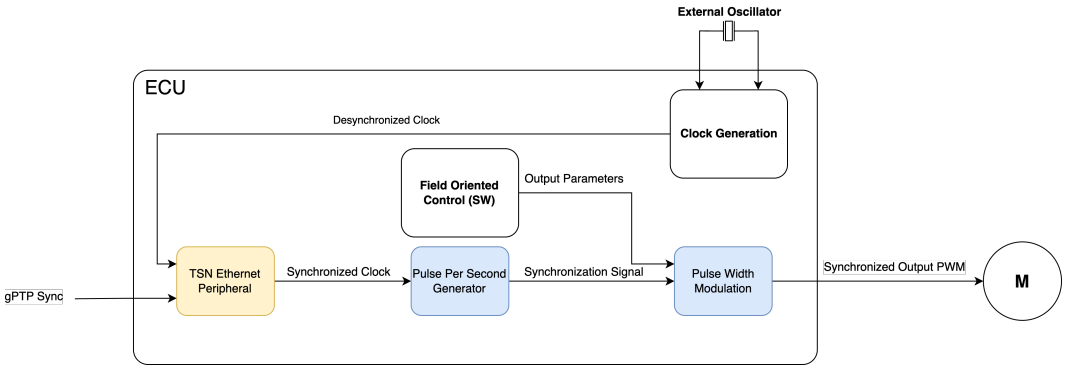


Figure 3.5: Synchronization Setup

The primary goal of the synchronization is to deterministically apply the Pulse Width Modulation (PWM) parameters calculated by the Field Oriented Control (FOC) algorithms. This can be achieved by ensuring the PWM output pulses are synchronized to the shared Grand Master (GM) timestamp.

Figure 3.5 demonstrates the synchronization of motor control using Time Sensitive Networking (TSN) hardware and software and is an expanded version of Figure 3.4. The synchronized signal from the Pulse Per Second (PPS) module is used to synchronize the output of PWM module to the electric motor, resulting in synchronized motor control.

The synchronization method and the hardware requirements were developed to support the hardware platform available for evaluation. There was only one component within the hardware that supports generating signals synchronized to the GM and that is the PPS module. Therefore, the PPS module was selected to generate the signal used to synchronize the PWM output.

Chapter 4 will introduce a hardware configuration used for synchronizing PWM signals as well as a methodology to evaluate the performance of synchronization.

Chapter 4

Evaluation

4.1 Methodology

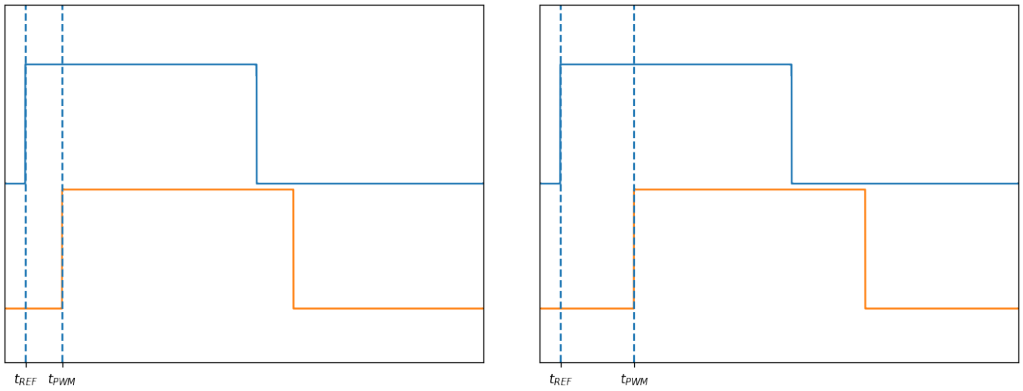
4.1.1 Measuring Drift

The drift between the Pulse Width Modulation (PWM) signal and the synchronized generalized Precision Time Protocol (gPTP) clock can be used as a criterion to evaluate the performance of peripheral synchronization. The lower the amount of drift, the better the synchronization.

To measure the amount of drift between the PWM signal and the network Grand Master (GM) clock, a reference signal is generated. The reference signal is synchronized to the GM time using gPTP. The frequency of the reference signal is equal to the frequency and duty cycle of the PWM signal.

At every pulse of the synchronization signal, the PWM output is synchronized to the network. In other words, the time difference between the edgePWM pulse and the edge of the reference signal is at its minimum value. During the time between two pulses of the synchronization signal, the PWM

signal drifts from the reference signal. The amount of drift can be calculated from the change in the time difference between the edge of the PWM signal t_{PWM} and the edge of the reference signal t_{REF}. Figure 4.1 has two plots showing the minimum and maximum time difference between the edges.



(a) $min(t_{PWM} - t_{REF})$ (b) $max(t_{PWM} - t_{REF})$

Figure 4.1: Clock Drift

The following formula can be used to calculate the maximum amount of drift between the PWM signal and the reference signal:

$$Drift = max(t_{PWM} - t_{REF}) - min(t_{PWM} - t_{REF})$$

The higher the frequency of the synchronization signal, the more frequently the PWM signal is synchronized and the lower the amount of drift. Therefore, to achieve the best synchronization performance while maintaining the desired PWM period, the synchronization frequency should be equal to the PWM frequency

$$f_{SYNC} = f_{PWM}$$

4.2 Testing

4.2.1 Hardware Configuration

The development hardware selected for the test setup was the NXP S32K39 platform for the following reasons:

- Time Sensitive Networking (TSN) enabled Ethernet Medium Access Control (MAC) and Physical layer (PHY) supporting time synchronization and hardware time-stamping
- Pulse Per Second (PPS) capable hardware for generating signals synchronized to the network time
- Pulse Width Modulation (PWM) peripheral supporting up to 20KHz output frequency (required for motor control) as well as external trigger support to re-synchronize the PWM signals

4.2.2 Software Configuration

In addition to hardware support, specific software including generalized Precision Time Protocol (gPTP) stack as well as peripheral drivers is needed. For this evaluation, a more focused configuration was created using only one Electronic Control Unit (ECU) with the following assumptions:

- The time registers in the Ethernet peripheral are synchronized to the network time. This is handled by gPTP in a fully functional network.
- The PPS generates signals at a predetermined time that is known to the other ECUs on the network.

4.2.3 Test Setup

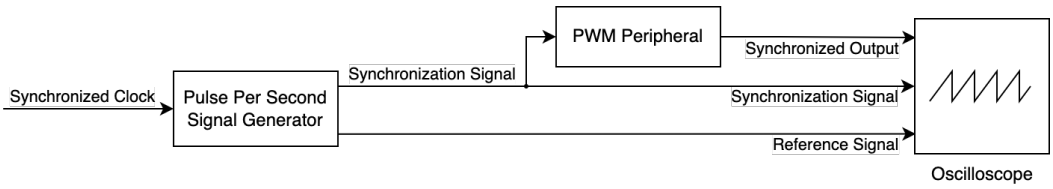


Figure 4.2: Test Setup

Figure 4.2 shows the hardware setup used for this evaluation. A clock synchronized to the gPTP Grand Master (GM) is an input to the PPS module and is used to generate two signals, a synchronization signal and a reference signal. This results in both signals being synchronized to the gPTP GM. The synchronization signal generated by the PPS is used to synchronize the output of the PWM module.

4.2.3.1 Pulse Width Modulation (PWM)

The PWM peripheral was configured as follows:

- External synchronization signal is enabled
- The output duty cycle is set to 50%
- The output frequency is set for each test case

4.2.3.2 Pulse Per Second (PPS)

Two PPS output signals were configured as follows:

- A **synchronization** signal with a duty cycle of 50% and a set frequency f_{SYNC} for each test case
  - $f_{SYNC} \times n = f_{PWM}$, where n is an integer

  - $f_{SYNC} \leq f_{PWM}$, otherwise PWM time period is shortened, and control is affected

- A **reference** signal with an output frequency f_{REF} and duty cycle equal to the PWM output

4.3 Results

4.3.1 Measurements

Signal measurements were collected using an oscilloscope. The oscilloscope channels were setup as follows:

Channel	Signal	Color
1	Pulse Width Modulation (PWM) Output (Phase A)	Yellow
2	Pulse Per Second (PPS) Generated Synchronization Signal	Magenta
3	PPS Generated Reference Signal	Cyan

Table 4.1: Oscilloscope Setup

Figure 4.3 shows an example capture from the oscilloscope of the evaluation setup recorded over 1000μs. In this capture the PWM output and the PPS generated reference signal have a frequency of 8 KHz and a duty cycle of 50%. The PPS generated synchronization signal has a frequency of 4 KHz and a duty cycle of 50%. The PWM output is synchronized at the leading edge of the PPS generated synchronization signal.

In addition to capturing the signals, the oscilloscope's statistics feature was used to measure the drift between the PWM output signal and the PPS generated reference signal.

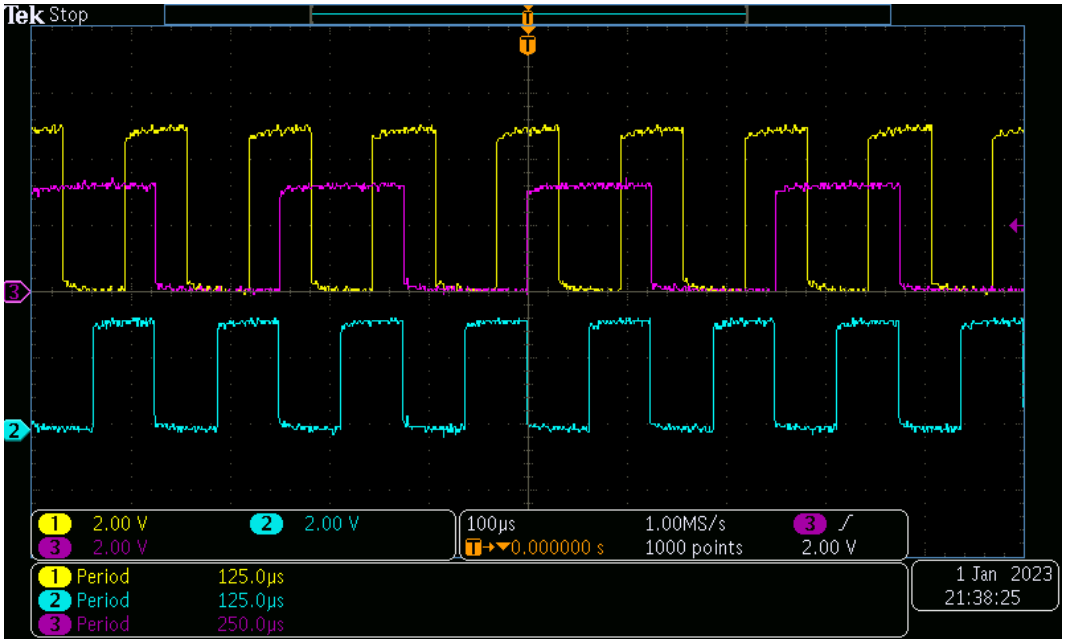


Figure 4.3: Oscilloscope Sample Capture

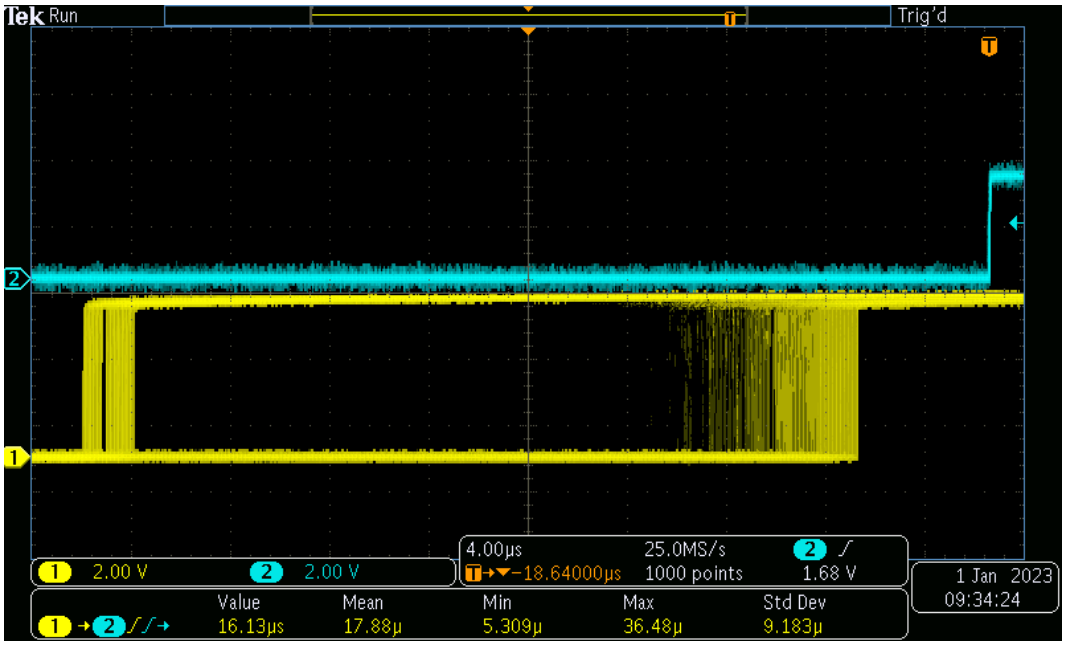


Figure 4.4: Drift Capture Example

Figure 4.4 shows an example of the statistics measured by the oscilloscope. Compared to Figure 4.3, this is a 40μs capture displaying only the PWM

output in yellow and the PPS reference signal in cyan. A sufficient number of samples are collected with the oscilloscope to make the statistics more accurate, providing a reliable measurement of the synchronization performance. The oscilloscope trigger is configured on the leading edge of the PPS generated reference signal (channel 2). We can observe the leading edge of the PWM output drifting from the reference signal. The statistics of the most importance for the evaluation are the minimum and maximum time difference between the leading edges of the PWM signal (in yellow) and the PPS generated reference signal (in cyan). They can be seen at the bottom of figure 4.4 where the minimum is 5.309μ and the maximum is 36.48μ.

In figure 4.4 the measured data is as follows:

PWM Frequency	5 KHz
Synchronization Frequency	1 Hz
$min(t_{PWM} - t_{REF})$	$5.309\mu s$
$max(t_{PWM} - t_{REF})$	$36.48\mu s$

Using the formula in the measuring drift section above we can calculate the maximum drift as:

$$\text{Drift} = 36.48\mu s - 5.309\mu s = 31.171\mu s$$

The calculated value is the worst-case drift of the PWM signal from the PPS generated reference signal and therefore from the Grand Master (GM) clock in a Time Sensitive Networking (TSN) network.

Since the motor control algorithms are synchronized to the PWM time period, calculating the drift as a percentage of the PWM time period will allow the comparison of the performance of the synchronization across different PWM frequencies.

A PWM signal with a frequency of 5 KHz has a time period of $200\mu s$. The drift calculated as a percentage is:

$$Drift_{\%} = \frac{31.171}{200} = 15.559\%$$

Using a 1 Hz synchronization signal for a 5 KHz PWM, we can limit our worst-case drift to 15.56% of the time period.

We can further improve the synchronization by increasing the frequency of the synchronization signal. Figure 4.5 shows a reduction in the maximum amount of drift due to the increased frequency of the synchronization signal.

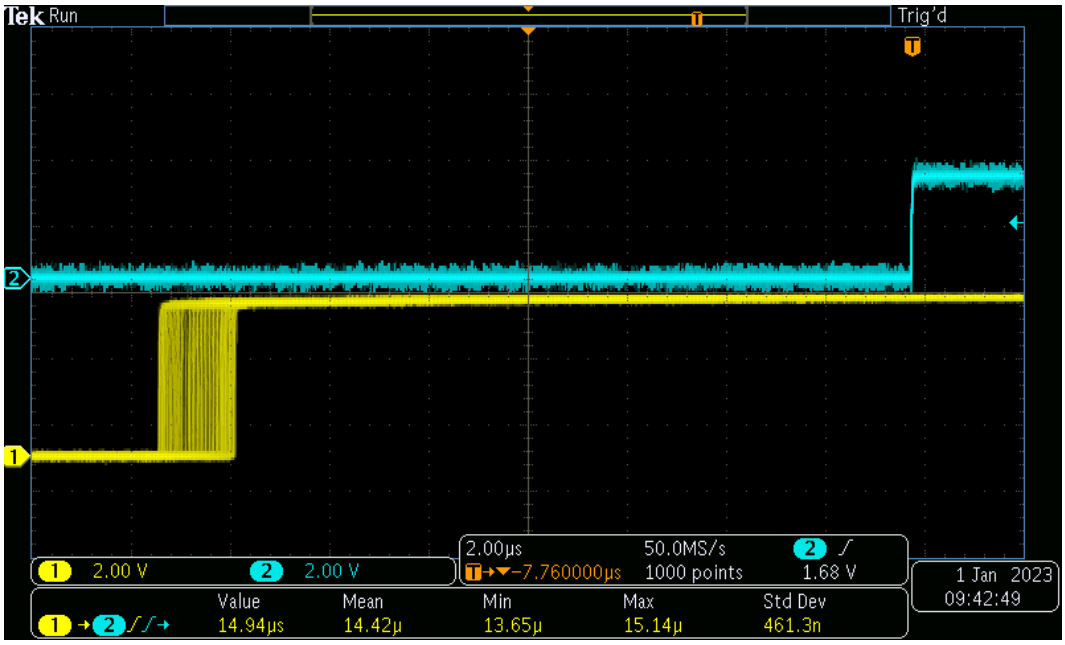


Figure 4.5: Drift (5KHz PWM, 20 Hz Sync)

$$Drift = 15.14\mu s - 13.65\mu s = 1.49\mu s$$

$$Drift_{\%} = \frac{1.49}{200} = 0.745\%$$

A drift of less than 1% of the PWM time period is achievable with a 20 Hz synchronization frequency. However, if we maintain the same synchronization frequency while increasing the PWM frequency, the amount of drift increases.

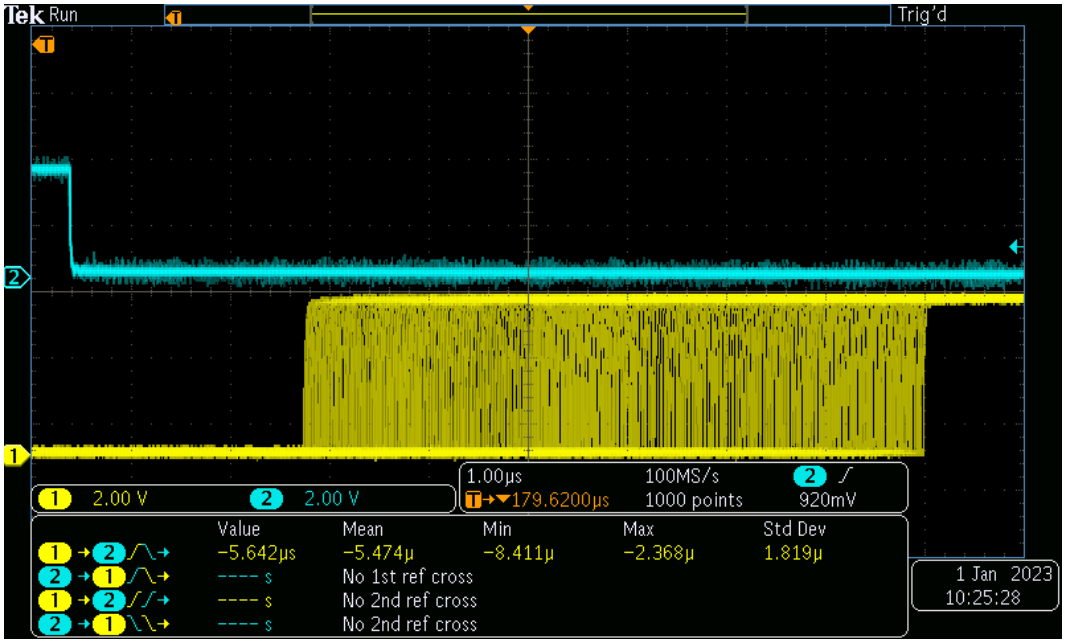


Figure 4.6: Drift (20KHz PWM, 20 Hz Sync)

In Figure 4.6 the setup is using a 20 KHz PWM and a 20 Hz synchronization signal. By calculating the drift, we observe a significantly higher drift of 12.086% compared to the 0.745% drift at 5 KHz PWM frequency.

$$Drift = -2.368\mu s - (-8.411)\mu s = 6.043\mu s$$

$$Drift_{\%} = \frac{6.043}{50} = 12.086\%$$

f_{SYNC} \ f_{PWM}	20,000	10,000	8,000	5,000	4,000
1	—	—	39.44	15.586	9.749
2	—	—	19.12	7.75	3.932
4	—	14.76	9.68	3.86	2.46
8	31.048	7.521	4.88	1.77	1.211
10	24.016	6.047	3.92	1.504	0.96
20	12.086	3.069	1.926	0.745	0.464
100	2.3896	0.613	0.386	0.145	0.097
1,000	0.238	0.057	0.038	0.15	0.008
4,000	0.054	—	0.016	—	0.003
5,000	0.04	0.008	—	0.005	—
8,000	—	—	0.002	—	—
10,000	0.018	0.004	—	—	—
20,000	0.008	—	—	—	—

Table 4.2: Test results

Table 4.2 contains the measurement of the tested PWM output and synchronization signal configurations. The values in the table are the maximum drift represented as a percentage of the PWM time period. At a given PWM frequency f_{PWM}, increasing the synchronization frequency f_{sync} reduces the maximum amount of drift before the PWM signal is resynchronized.

As mentioned in 4.1.1, to achieve the best synchronization performance, the frequency of the synchronization signal should be equal to the frequency of the PWM signal. A worst-case drift of less than 0.01% of PWM time period is achievable using the described synchronization method for the frequencies tested when $f_{PWM} = f_{SYNC}$ is highlighted in the table. To put the results in perspective, the precision allowable by the 16-bit PWM peripheral used in this evaluation is $\frac{1}{2^{16}} = 0.0015\%$.

Chapter 5

Conclusion & Future Work

5.1 Conclusion

Real-time control in centralized Electrical and Electronic (E/E) architectures requires synchronization to meet the timing and performance requirements. Although Time Sensitive Networking (TSN) standards offers time synchronization between Electronic Control Units (ECUs) in a vehicle, they are limited to guaranteeing a shared timestamp across all network devices. Achieving real-time control synchronization requires additional implementation.

This thesis proposes a method utilizing Pulse Per Second (PPS) signals generated from a generalized Precision Time Protocol (gPTP) synchronized clock and evaluates its performance. The evaluation results demonstrate that the proposed synchronization method effectively synchronizes the Pulse Width Modulation (PWM) generated signals with the network time, ensuring accurate synchronization between the smart actuator and the domain controller.

5.2 Future Work

The synchronization and evaluation methods proposed in this thesis focus on synchronizing the Pulse Width Modulation (PWM) peripheral's output and make assumptions about the rest of the setup. Additionally, although the motor control application used demonstrated good performance, it may not be a practical use case due to its sensitivity to timing inaccuracies.

The following can be explored to expand upon this work:

- Centralized zonal architectures are becoming the preferred solution over domain centralized architectures, the work can be adapted to such architectures with minimal changes.

- The performance of synchronization was evaluated by assuming ideal Time Sensitive Networking (TSN) synchronization performance. Assessing the impact of TSN synchronization performance on the application is necessary for a complete evaluation of the performance of the system.

- Applying the synchronization method to less strict applications might be more feasible. One of these applications could be Advanced Driver-Assistance System (ADAS) camera frame synchronization, which requires a lower frequency compared to the motor control application (5-60 Hz vs 2-20KHz).

Bibliography

[1] Peter Aberl, Stefan Haas, and Arun Vemuri. "How a Zone Architecture Paves the Way to a Fully Software-Defined Vehicle". en. In: (2023).

[2] Alan Amici. *Advanced ethernet could safeguard AVs against network delays*. en. May 2019. URL: https://www.axios.com/2019/05/15/advanced-ethernet-could-safeguard-avs-against-network-delays (visited on 11/20/2023).

[3] *Automotive Ethernet: An Overview*. en. URL: https://support.ixiacom.com/resources/automotive-ethernet-overview (visited on 06/06/2024).

[4] Victor Bandur et al. "A Domain-Centralized Automotive Powertrain E/E Architecture". In: Apr. 2021. DOI: 10.4271/2021-01-0786.

[5] Victor Bandur et al. "Making the Case for Centralized Automotive E/E Architectures". In: *IEEE Transactions on Vehicular Technology* 70.2 (Feb. 2021). Conference Name: IEEE Transactions on Vehicular Technology, pp. 1230–1245. ISSN: 1939-9359. DOI: 10.1109/TVT.2021.3054934. URL: https://ieeexplore.ieee.org/document/9337216 (visited on 11/20/2023).

[6] BOSCH. *230831-bosch-xc-whitepaper-ee-architektur-en.pdf*. URL: https://www.bosch-mobility.com/en/mobility-topics/ee-architecture/ (visited on 11/20/2023).

[7] Paul Dan Burlacu, Laszlo Mathe, and Remus Teodorescu. "Synchronization of the distributed PWM carrier waves for modular multilevel converters". In: *2014 International Conference on Optimization of Electrical and Electronic Equipment (OPTIM)*. Bran, Romania: IEEE, May 2014, pp. 553–559. ISBN: 978-1-4799-5183-3. DOI: 10.1109/OPTIM.2014.6851001. URL: http://ieeexplore.ieee.org/document/6851001/ (visited on 04/08/2024).

[8] *IEEE 802.1AS-2020 Timing and Synchronization for Time-Sensitive Applications*. en. URL: https://standards.ieee.org/ieee/802.1AS/7121/ (visited on 06/06/2024).

[9] *IEEE 802.1Qbv-2015 IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks*. en. URL: https://standards.ieee.org/ieee/802.1Qbv/6068/ (visited on 09/04/2024).

[10] S. Kanajan et al. "Exploring Trade-off's Between Centralized versus Decentralized Automotive Architectures Using a Virtual Integration Environment". In: *Proceedings of the Design Automation & Test in Europe Conference*. Vol. 1. ISSN: 1558-1101. Mar. 2006, pp. 1–6. DOI: 10.1109/DATE.2006.243895. URL: https://ieeexplore.ieee.org/document/1656943 (visited on 03/22/2024).

[11] Thomas Liebetrau. "E/E Architecture Transformation How it impacts value chain and networking technologies". In: *AmE 2022 - Automotive meets Electronics; 13. GMM-Symposium*. Sept. 2022, pp. 1–7. URL:

https://ieeexplore.ieee.org/abstract/document/10025908 (visited on 08/06/2024).

[12] Varun M. Navale et al. "(R)evolution of E/E Architectures". English. In: *SAE International Journal of Passenger Cars - Electronic and Electrical Systems* 8.2 (Apr. 2015). Number: 2015-01-0196, pp. 282–288. ISSN: 1946-4614, 1946-4622. DOI: 10.4271/2015-01-0196. URL: https://www.sae.org/publications/technical-papers/content/2015-01-0196/ (visited on 09/05/2024).

[13] Emanuel Panholzer et al. "Method for prediction of EMI Emissions from Automotive Ethernet to Vehicle Antennas". English. In: *Advances in Radio Science*. Vol. 19. ISSN: 1684-9965 Issue: E. Copernicus GmbH, Dec. 2021, pp. 139–146. DOI: 10.5194/ars-19-139-2021. URL: https://ars.copernicus.org/articles/19/139/2021/ (visited on 06/06/2024).

[14] Jens Sorensen, Dara O'Sullivan, and Christian Aaen. "Synchronization of Multi-Axis Motion Control Over Real-Time Networks". In: *PCIM Europe 2018; International Exhibition and Conference for Power Electronics, Intelligent Motion, Renewable Energy and Energy Management*. June 2018, pp. 1–7. URL: https://ieeexplore.ieee.org/document/8402996 (visited on 03/25/2024).

[15] *Time-Sensitive Networking (TSN) Task Group*. URL: https://1.ieee802.org/tsn/ (visited on 09/04/2024).

[16] Yanli Xu and Jinhui Huang. "A Survey on Time-Sensitive Networking Standards and Applications for Intelligent Driving". en. In: *Processes* 11.7 (July 2023). Number: 7 Publisher: Multidisciplinary Digital Publishing Institute, p. 2211. ISSN: 2227-9717. DOI: 10.3390/pr11072211.

URL: https://www.mdpi.com/2227-9717/11/7/2211 (visited on 11/20/2023).