Rhine Waal University of Applied Science
Faculty of Communication and Environment
Supervisor: Prof Volker Strumpen

Bachelor Thesis

to obtain the academic degree
„Bachelor of Science“
in the Winter Semester 2021

Design and Implementation of a CAN Bus Simulation Embedded System including Fault Injection using Arduino Portenta H7

—

Submitted by:

Nguyen Dang Tien Loi
Kieler Straße 391a, 22505 Hamburg
Tel.: +49 1578 1042024
E-Mail: tienloi92@gmail.com
Communication and Information Engineering, 7th Semester
Matriculation number: 21009

Issue date: XX.XX.20XX

Filing date: XX.XX.20XX

Contents

List of Figures

List of Tables

1 Introduction 3p

1.1 Background and Project Motivation

Jungheinrich is one of the largest intralogistic companies in the world, famous of high quality Forklifts and complete logistic Portfolio. With the business strategy of excelling in customer services while being competitive in price, After-Sales service play a crucial role in the company development. One of many projects to improve the Maintenance service is the development of a Diagnostic Tool named Truckscope since 2014 []. Although the tool has been rolled into production, the Application continuous development require continuous testing procedure. Rather than completely rely on the developer's testing, Jungheinrich would like to test Truckscope to verify developer's implementation thoroughly.

Truckscope is the Diagnostic Application of Jungheinrich to replace their old Diagnostic Tool named Judit. The development was outsourced to an external partner while Jungheinrich participate in the Development as the Product Owner. Truckscope is a C#Application developed to run on Window and communicate with the Truck via CAN interface. The CAN interface is implemented using a so-called INCADO cable which convert CAN message into Serial USB signal to be read by the computer. During the Continuous Development, Jungheinrich describe the needed features, provide the procedure to control the Truck and verify the Developer's implementation before any new feature publishing. Until now, new feature verification is done on a real truck. The advantage of such approach is its practicality while the disadvantage is the lack of flexibility. Since the Truck behave very consistent and is difficult to be configured to work out of normal, the probability of finding bug in the Application is limited. Therefore, a flexible CAN Bus Simulation tool including Fault Injection is needed for a more extensive Testing of Truckscope.

On the other hand, Arduino is the instrument of choice for studying and prototyping. Due to its minimal and developer-friendly design, Arduino Education boards are very popular to students, hobbyists and developers. In the Professional sector, Arduino introduced several families such as Edge Control, Portenta Family, Nicla Family, MKR Family and Nano Family []. Different from the Education Board, those Professional Arduino Boards offers advancing functionalities such as Embedded Machine Learning, AI or IoT. The Arduino Portenta H7 is a member of the Portenta Family, newly released on February 2020 [2]. With embedded FDCAN peripheral and powerful processing cores, Portenta H7 is a prominent candidate for realizing the project.

1.2 Requirement Specification

1.2.1 CAN Bus Simulation Features

To communicate with Truckscope, the CAN Bus Simulation tool must comply with the CAN protocol as well as Truckscope procedure. In general, Truckscope is a CAN 2.0A device which follow CAN specification version 2.0 for standard 11-bit identifier. The CAN Baud rate is fixed at 250000, the standard Baud rate of Jungheinrich Trucks. According to Bosch specification, a CAN node is consisted of 4 layers: Physical Layer, Transfer Layer, Object Layer and Application Layer.
Physical Layer refers to the Signal Level, Bit Representation and Transmission Medium. In case of Truckscope, the so-called INCADO cable is used as the Transmission Medium, which convert USB Serial signal into CAN Signal (CAN High & CAN Low) and vice versa. Therefore the Tool must be able to communicate CAN via INCADO cable.
Transfer Layer defines the communication mechanism such as: Fault Confinement, Error Detection and Signaling, Message Validation, Acknowledgement, Arbitration, Message Framing, Transfer Rate and Timing. Truckscope implemented fully CAN 2.0 and capable of all above functions. Those functions regulate the CAN communication between multiple CAN nodes. Although the CAN Bus Simulation tool communicate with Truckscope in a peer-to-peer manner, it still need to comply with those mechanisms for a smooth communication with Truckscope.
Object Layer concerns the Message Filtering, Message and Status Handling of the CAN communication, in detail it find which message to be transmitted, which message received to be used. Truckscope Object Layer is implemented base on several libraries. Regarding CAN protocol, it is SerCan library which come with the INCADO cable. This layer is valuable in term of referencing while designing the Tool.
In the Application Layer, Jungheinrich adopted CANopen architecture and further tailored it to the company need. Both Truckscope and the CAN Bus Simulation Tool must follow the same Jungheinrich CANopen specification to interpret the CAN message correctly.
Beside above layer features, the CAN Bus Simulation tool should be able to demonstrate the dynamic transition of parameters. It reflects the behavior of the real Truck in operation and is valuable in testing how Truckscope capture and process those variant parameters.

1.2.2 Fault Injection Features

Fault Injection is an added value to the CAN Bus Simulation tool. It is built to facilitate Testing during the Continuous Development of Truckscope. As an initial project, the scope of the tool's Fault Injection design would be held minimized but sufficient representative. Therefore the injection channel is solely through CAN messages and Faults are

designed for testing Truckscope's Object Layer. In term of handling message at Object Layer, three primary fault types are No-reply, Timeout and False response.
No-reply fault happens when a CAN request from Truckscope is not responded at all. Depending on type of CAN request, Truckscope should behave accordingly to the Jungheinrich specification such as notify error or re transmit the CAN request. To facilitate this type of fault, the Simulation Tool must be able to ignore specific CAN requests.
Timeout fault on the other hand happens when a CAN response is not received in the defined time frame. According to Jungheinrich specification, Truckscope have Timeout error as well as handler for different type of CAN request. Real-Time is crucial here since the time between request and response must be ensured correctly to justify the behavior of Truckscope. The tool should add arbitrary delay to selected CAN request and ensure the response time as tester's design.
False responses mean responses which are not followed the normal workflow of Truckscope. Truckscope must then recognize the incorrectness and act properly. To facilitate this kind of fault, the Simulation tool should allow tester to define the CAN response to specific CAN request.

1.2.3 User Interface

Since the Simulation tool is designed for testing engineers or developers, the user interface requirements are relative flexible. In general, the user expect to have a simple interface to setup and monitor the CAN Bus Simulation tool workflow.
In term of tool's setup, user should have an interface to set starting parameters and define the Fault Injection scenario. As an initial development, a raw interface with clear structure is sufficient for the user. On the other hand, an user interface to display workflow of the tool is also necessary. The desired workflow information is CAN message received and responded with timestamp, Fault Injection Scenario notification. It might be displayed in either user's PC monitor or an external LCD display.

1.3 Development Approach

Following the classic waterfall model, the Thesis would address the Design and Implementation in the following sections. Before diving into technical details, a logical analysis is necessary to defend the choice of development platform as well as development board. In general, the CAN Bus Simulation tool is decided to be developed as an Embedded Systems using specifically Arduino Portenta H7 as Development Board.
There are two main building blocks in the desired Embedded Systems: CAN Bus Simulation and Fault Injection Functionality. To accomplish those functions, the CAN Bus Simulation tool should be developed in a Real-Time development platform. Real-Time refers to the execution which must complete within an maximum time period [1]

Criteria	Embedded System	General-Purpose Computer System
Real-Time	Yes	No
Purpose	Perform dedicated functions	Perform a large number of tasks
Optimization	Yes	Not fully optimised
Reliability	Extremely reliable	Reliable
Size of Tool	Compact	Along with PC
Power Consumption	Low	High

Table 1: Embedded System versus General-Purpose Computer System

From the above table, it is reasonable to select embedded system as the development platform. The main advantage of using embedded system in this project is its real-time and optimization nature. Executing only designed tasks make the tool operation not only more efficient but also more reliable.

Among many vendors, Arduino is selected because of its various development boards and developer-friendly design. Since this initial project is aimed to be used in a real world project, Arduino professional boards are more appropriate to utilize its available industry standard peripherals. Among those professional development boards, Arduino Portenta H7 is the most prominent candidate. It is a very powerful board with the heart is a STM32H747xI micro controller, which include an Arm Cortex M7 core and an Arm Cortex M4 core operates at up to 480 MHz and 240 MHz respectively. The board is ideal for real-time application because of its dual core structure, this feature allow Portenta H7 to run tasks in parallel. Furthermore, Arduino Portenta H7 has a built-in Mbed Operating System with available RTOS API [].

Arduino Portenta H7 has an excessive list of peripherals, one of them is FDCAN. The FDCAN peripheral is not only fully compatible with the classical CAN protocol but also with the CAN FD protocol which is an extension of the classical CAN (CAN 2.0). A CAN communication tool based on those industry standard peripheral would be more reliable. On the other hand, the vast embedded peripherals also ensure the scalability of the project.

2 Literature Review 9p

2.1 Embedded System Design

2.1.1 Real Time Application

2.2 Controller Area Network

2.2.1 CAN bus

Different from traditional network such as USB or Ethernet, CAN does not send a large blocks of data point-to-point from node A to node B under the supervision of a central bus master. In a CAN network many short messages like temperature or RPM are broadcast to the entire network, which allows for data consistency in every nocde of the system. CAN bus is a multi-master, message broadcast system with maximum signaling rate of 1 Mbit per second developed by Bosch GmbH [3].

2.2.2 CAN protocol, ISO 11898

2.2.3 CAN Bus Simulation

2.3 Fault Injection

2.3.1 Fault Classification

2.3.2 Fault Design Framework

3 Design 9p

3.1 Residual CAN bus simulation design

3.2 Fault Injection Design

(What functionalities of Portenta H7 shall be used ?)

- Internet/Bluetooth
- CAN
- RTOS
- dual cores

4 Implementation 9p

4.1 CAN Communication using Portenta H7

4.2 Simulation implementation

4.3 Fault Injection Implementation

5 Results and Discussion 5p

5.1 Results

5.1.1 Final State of Tool

5.1.2 Compared to 1.2

5.2 Discussion

5.2.1 Self Comment on Portenta H7

5.2.2 Self Comment on Embedded Platform

6 Conclusions and Future Work 4p

6.1 Conclusions Project Succeed or Fail

6.2 Future Work

References

[1] Giorgio C Buttazzo. *Hard real-time computing systems: predictable scheduling algorithms and applications*. Vol. 24. Springer Science & Business Media, 2011.

[2] Salman Faris. "Arduino announces the powerful new Arduino Portenta family". In: *URL https://www.seeedstudio.com/blog/2020/01/13/arduino-announces-the-powerful-new-arduino-portenta-family/* ().

[3] Mohammad Farsi, Karl Ratcliff, and Manuel Barbosa. "An overview of controller area network". In: *Computing & Control Engineering Journal* 10.3 (1999), pp. 113–120.