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.
To realize such a tool, Jungheinrich find Embedded System suitable for its real-time features, reliable behavior and specialized design. On the other hand, Arduino is the classic instrument of Embedded System 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 Section, 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, coming with two asymmetric cores with included Floating Point Unit and is considered as one of the most powerful boards in the market nowadays. With an extensive embedded peripherals such as FDCAN, Ethernet, Bluetooth and more, Arduino Portenta H7 is not only fulfill all desired functionalities of the CAN Bus Simulation tool but also ensure scalability of the project in the future. Seeing

those potentials from Arduino Portenta H7, this Thesis employ this board to design and implement a CAN Bus Simulation tool including Fault Injection.

1.2 Requirement Specification

1.2.1 CAN Bus Simulation Features

To communicate with Truckscope, the CAN Bus Simulation tool must comply to the corresponding 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, same as the general Baud rate of Jungheinrich Trucks. Following Bosch definition, a CAN node is divided into 4 Layers: Application Layer, Object Layer, Transfer Layer and Physical 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 (high and low) and vice versa. Therefore the Tool must be able to read and write CAN Signal via INCADO cable.
Transfer Layer refers to 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 should still comply to those regulations to be able to communicate even in a more complex network.
Object Layer concern 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 only valuable in term of referencing rather than restrict the Tool Object Layer design.
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 understand each other.
Beside the above mentioned features, the CAN Bus Simulation tool should be able to demonstrate the parameter changes. 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 the Continuous Testing Procedure in Truckscope development. As an initial project, the

scope of the tool's Fault Injection design would be minimized but still enough representative. Therefore the injection channel is solely through CAN bus and aims to support Acceptance Test of Truckscope. Three primary fault types are No reply, Timeout and Incorrect response.
No-reply fault happens when a CAN request from Truckscope is not responded at all. According to different CAN request, Truckscope should behave accordingly to the Jungheinrich specification such as pop up error or retransmit 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 sent in the defined time frame. According to different scenario, Truckscope might ignore, retransmit request or pop up error. Real-Time is crucial here since the time between request and response must be ensured to justify the behavior of Truckscope. The tool should add arbitrary delay to selected CAN request and ensure the response time as expectation.
Incorrect responses mean responses which are not followed the normal workflow of Truckscope. Truckscope must then recognize the incorrectness and response properly. This type of fault require the Simulation tool to response as predefined 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 monitor the tool workflow, CAN request from Truckscope, CAN response from the CAN Bus Simulation tool and Fault Injection Notification.
The concern is put on the precision of the Interface in term of timestamp, CAN message as well as Fault Injection message. The interface is either on a PC monitor or an external display.

1.3 Methodology

There are two building blocks of the desired Embedded Systems: CAN Bus Simulation and Fault Injection Functionality. From the above requirements, several functionalities should be available on the development board: Real-Time operation and CAN Communication. Arduino Portenta H7 offers all of those functionalities and even more for further development.
Arduino Portenta H7 is able to perform real time tasks. The board is equipped with a Real Time Counter (RTC) peripheral and mbed Operating System with supported Real Time Operating System (RTOS) API. The heart of the Portenta H7 is a STM32H747xI micro controller, which include an Arm Cortex M7 core with frequency up to 480 MHz and an Arm Cortex M4 core with frequency up to 240 MHz. The dual core enable dual

processing ability of the Arduino Portenta H7 which make it even more suitable for the Real Time application.
FDCAN is an available peripheral on Arduino Portenta H7. The FDCAN peripheral is not only fully compatible to the classical CAN protocol but also enable to communicate in CAN FD protocol which is an extension of the classical CAN (CAN 2.0).
Although there are different ways to program Arduino Portenta H7, the desired Embedded System shall be developed using official Arduino IDE. The API is mainly used is the Hardware Abstraction Layer (HAL) API which offer deep and extensive control of the development board.

2 Literature Review 9p

2.1 Rest bus Simulation in the Context of Model-Based Testing

2.2 Fault Injection Tool

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