OP Coders ;

Tyre Explosion Avoiding System

Supervision: **Dr. Roaa Mubark**

Assistant Professor at Faculty of Engineering

Helwan Universty

Mentor Supervision: **Eng. Ahmed Assaf**

Team Leader at Valeo Egypt

Prepared by

Abdelrahman Hossam Sobhy Ibrahim

Seif Al-dein Ahmed Ahmed Hashad

Abdullah Mustafa Abdullah Ahmed

Abdelrhman Osama Talaat Ali

Ali Ali Sayed Badawy

OP Coders;

Beyound limits

Tyre Explosion Avoiding System

Supervision: **Dr. Roaa Mubark**
Assistant Professor at Faculty of Engineering
Helwan Universty
Mentor Supervision: **Eng. Ahmed Assaf**
Team Leader at Valeo Egypt

Prepared by

Abdelrahman Hossam Sobhy Ibrahim

Seif Al-dein Ahmed Ahmed Hashad

Abdullah Mustafa Abdullah Ahmed

Abdelrhman Osama Talaat Ali

Ali Ali Sayed Badawy

OP Coders;

“There’s nothing wrong with the car except that it’s on fire.”

– Murray Walker

“Any fool can write code that a computer can understand. Good programmers write code that humans can understand.”

– Martin Fowler

OP Coders;

Acknowledgment

First and foremost, we are grateful to our great creator Almighty Allah for the good health and wellbeing that were necessary to complete this project.

This venture would not have been possible without the generosity of many people who contributed their time and talents to the various phases of this work.

It’s our proud privilege to release the feelings of our gratitude for the immense contribution of these people who directly or indirectly made this project a success.

We would like to express our hearts full indebtedness and owe a deep sense of gratitude to our graduation project supervisor **Dr. Roaa Mubark** for her advice and knowledge and the insightful discussions and suggestions throughout the whole venture, and for patiently guiding us through its processes. You made us believe that we have what it takes and we have the potential to reach greater heights in our graduation project.

We would like to thank **Eng. Ahmed Assaf** for his dedication, his efforts and his great support to our team.

Finally, we would like to thank **Valeo Egypt** for all the efforts and help to make sure that our graduation project meets how they work and meets the qualifications of modern technologies.

Abstract

Driving Safely is a right for every person, also having fast medical care in case of accidents is a right, wherever you are and whenever. So “as Engineers” we dedicated our efforts to achieve that.

We tried to create the perfect project from our humble perspectives, this project should be able to prevent tire explosion accidents, and in case of any accidents not only tire explosion accidents, but it could also call immediately for help to save each possible life.

The idea comes to our heads due to the latest statistics reported from the National Highway Traffic Safety Administration, Tire blowouts are estimated to cause more than 700 deaths and 7800 crashes each year. Tire blowout leads the driver of the vehicle to lose control and maybe it leads the car to turn over its sides. This problem may lead other cars to crash or to stop suddenly and that also leads to other accidents. A blowout is the sudden action of loss in air pressure in any inflatable tire. Sometimes accompanied by the sound of an explosion. They’re caused by too little air pressure, extreme heat, impact damage, overloading, under inflation, or a combination thereof, the recipe, in any case, is always an excessive strain on a tire’s internal structure.

Tire Explosion Avoiding system (TEAS) seeks to make the client much safer. This is done by the contribution of various systems that will be implemented together. TEAS aims to reduce the number of those accidents by monitoring data of the tires and depending on these reading (TEAS) will decide if it’s a must to warn the driver early whether there’s a problem with his tires or not.

For example, if the tires pressure is lower or higher than the expected level, (TEAS) will sent him such a helpful instructions to stop his car and change the tires immediately. In case of a tire blowout, (TEAS) will collect various data about car movement like the steering angle, rotational rate, speed, tire pressure and gyroscope and sent them to a control unit (CU) to make its data analysis to put our hands on the cause of that tragedy. In case of an autonomous car the control unit will stop the car immediately, in case of manual driving it will help the driver to stop safely without losing control.
After data monitoring stage, (TEAS) uses this data to send warnings to the adjacent cars using vehicle to vehicle communication (V2V) to avoid subsequent accidents. It also sends them to Original Equipment Manufacturer (OEM) where their data analysis team can make use of it to generate a better algorithm to avoid occurring these problems again. In case of an accident has happened, (TEAS) sends car’s ID, type and it’s location to the authorities to tell them that there’s an accident happened, The authorities will take the needed procedures to help the injured people by calling ambulance and helping others by broadcasting a breaking news with the road that accident has happened on it to take alternative paths and preventing subsequent accidents.

In traditional way if there is a bug in the software of the system, the client must go to the nearest service station to be able to get rid of it. (TEAS) solves this problem, This is done by updating the software remotely using a Firmware Over The Air (FOTA) system. (FOTA) solves the problem of reliability of each updated that used to be present earlier. That helps the Component Supplier (tier 2 supplier) so much, The dependency of their service stations carrying the vehicle to the station, the cost of updating the software becomes very low. Also eases the role of the client, No much hassle to the customer, One click will allow him to update the software of his car immediately.

Contents

Contents

List of Tables

Contact

Abdelrahman Hossam Sobhy Ibrahim

Abdelrahmanhossam797@gmail.com

+20 120 542 7791

Seif Al-dein Ahmed Ahmed Hashad

Seifherooz@gmail.com

+20 103 388 8493

Abdullah Mustafa Abdullah Ahmed

Abdallah.moustafa333@gmail.com

+20 106 805 3019

Abdelrhman Osama Talaat Ali

Abdelrhmanosama1110@gmail.com

+20 128 156 8618

Ali Ali Sayed Badawy

Aalimahrez@gmail.com

+20 127 005 2750

List of Figures

Table of Acronyms and Definitions

Abbreviation	Meaning
FOTA	Firmware Over The Air
V2C	Vehicle To Cloud
V2V	Vehicle To Vehicle
GUI	Graphical User Interface
OTA	Over The Air
OEM	Original Equipment Manufacture
C2(Authorties)	Cloud To Authorties
MCAL	Micro-Controller Abstraction Layer
HAL	Hardware Abstraction Layer
CAN	Controller Area Network
UART	Universal Asynchronous Receiver-Transmitter
SPI	Serial Peripheral Interface
ECU	Electronic Control Unit
I2C	Inter-Integrated Circuit
OS	Operating System

Chapter 1
Introduction

1.1 Problem Definition

According to the latest statistics reported from the National Highway Traffic Safety Administration, Tire blowouts are estimated to cause more than 700 deaths and 7800 crashes each year. Tire blowout leads the driver of the vehicle to lose control and maybe it leads the car to turn over its sides. This problem may lead other cars to crash or to stop suddenly and that also leads to other accidents. A blowout is the sudden action of loss in air pressure in any inflatable tire. Sometimes accompanied by the sound of an explosion. They're caused by too little air pressure, extreme heat, impact damage, overloading, underinflation, or a combination thereof, the recipe, in any case, is always an excessive strain on a tire's internal structure.

There is a slew of issues that can cause a truck tire to explode. Oftentimes, there are several contributing factors to a tire blowout. Some of these factors include:

Improper tire inflation – Tires that are underinflated or overinflated are at risk for explosion. Too much pressure on the tire can cause overinflated tires to blow out.

Design defect – Defects in a tire can cause cracks to form on its exterior, leading to significant air loss and explosion.

Overheating – When the temperature inside a tire increases, so does the pressure. If the temperature of a tire rises to a dangerous level due to the environment or other factors, the tire is at risk for a blowout.

Poor condition – Tires that are old and need to be replaced sometimes have structural weaknesses. This can cause dangerous changes in pressure or cracks in the body of the tire.

Hazards on the road – Debris, potholes, uneven roads, and blockades on the road can cause a tire to explode depending on the location of impact.

1.2 Problem Importance

Tire Explosion Accidents cause 700 death every year but some injuries are deaths, some people after an accident like this won't move again, some won't be able to talk or see.

1.3 Current Solution

From November 1, 2014, all new passenger cars sold in the European Union must be equipped with a TPMS. (Tire-pressure monitoring system).

1.4 Objectives

1- Enhanced Tire monitoring system
2- Firmware over the air updates
3- OEM Data Collection System
4- Vehicle to Vehicle Communication
5- Authorities Informing

1.5 Project Idea

Tire Explosion Avoiding system (TEAS) seeks to make the client much safer. This is done by the contribution of various systems that will be implemented together. TEAS aims to reduce the number of those accidents by monitoring data of the tires and depending on these reading (TEAS) will decide if it's a must to warn the driver early whether there's a problem with his tires or not. For example, if the tires pressure is lower or higher than the expected level, (TEAS) will sent him such a helpful instructions to stop his car and change the tires immediately.

In case of a tire blowout, (TEAS) will collect various data about car movement like the steering angle, rotational rate, speed, tire pressure and gyroscope and sent them to a control unit (CU) to make its data analysis to put our hands on the cause of that tragedy. In case of an autonomous car the control unit will stop the car immediately, in case of manual driving it will help the driver to stop safely without losing control.
After data monitoring stage, (TEAS) uses this data to send warnings to the adjacent cars using vehicle to vehicle communication (V2V) to avoid subsequent accidents. It also sends them to Original Equipment Manufacturer (OEM) where their data analysis team can make use of it to generate a better algorithm to avoid occurring these problems again. In case of an accident has happened, (TEAS) sends car's ID, type and it's location to the authorities to tell them that there's an accident happened, The authorities will take the needed procedures to help the injured people by calling ambulance and helping others by broadcasting a breaking news with the road that accident has happened on it to take alternative paths and preventing subsequent accidents.

In traditional way if there is a bug in the software of the system, the client must go to the nearest service station to be able to get rid of it. (TEAS) solves this problem, This is done by updating the software remotely using a Firmware Over The Air (FOTA) system. (FOTA) solves the problem of reliability of each updated that used to be present earlier. That helps the Component Supplier (tier 2 supplier) so much, The dependency of their service stations carrying the vehicle to the station, the cost of updating the software becomes very low. Also eases the role of the client, No much hassle to the customer, One click will allow him to update the software of his car immediately.

1.6 Motivation

Our motivation is to help as many as possible persons in their lives, and we believe that if You have a chance to save lives If you don't take it, you may regret it. We should never forget that God granted us the power to reason so that we would do work here on Earth - so that we would use science to save lives, and give hope.

1.7 Educational Value

Working on this project has added new values not only to our technical overview, but also to our non-technical overviews that has helped us as a team and as individuals.

1.8 Technical Value

Embedded Systems (ARM based Microcontrollers)
System Architecture and design
Bootloader Concepts and design
V2C Technologies
Cloud Computing
Database
GUI Desktop Applications
Web Applications
Car Simulation

1.9 Non-Technical Value

Idea generation techniques
1- Writing a business plan
2- Work division and integration
3- Time Management
4- Documenting and presenting a large-scale project

1.10 Domain of Application

Our system needs co-operation between OEM's and local authorities for each country to have a standard global system, and with all collected information, system improvement and algorithms enhancement will be easier and faster.

Chapter 2

Background

This project needed a lot of technical background including theoretical knowledge, practical knowledge, and we had to use a lot of different programming languages and tools.

2.1 Pogramming Languages

1- **C programming Language**

C is a general-purpose, procedural computer programming language supporting structured programming, lexical variable scope, and recursion, with a static type system. By design, C provides constructs that map efficiently to typical machine instructions.

Figure 2.1 C

2- **C# programming Language**

C# is a general-purpose, multi-paradigm programming language encompassing static typing, strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented, and component-oriented programming disciplines.

Figure 2.2 C#

3- **Python programming Language**

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation.

Figure 2.3 Python

4- **Java programming Language**

Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible.

Figure 2.4 JAVA

5- **PHP**

PHP is a general-purpose scripting language geared towards web development. It was originally created by Danish-Canadian programmer Rasmus Lerdorf in 1994. The PHP reference implementation is now produced by The PHP Group.

Figure 2.5 PHP

6- **SQL**

SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system, or for stream processing in a relational data stream management system.

Figure 2.6 SQL

2.2 Software Tools

1- **Eclipse IDE**

Eclipse is an integrated development environment used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment.

Figure 2.7
Eclipse

2- **Stm32 Cube IDE**

STM32CubeIDE is an all-in-one multi-OS development tool,which is part of the STM32Cube software ecosystem. STM32CubeIde Board PhotoSTM32CubeIDE is an advanced C/C++ development platform with peripheral configuration, code generation, code compilation, and debug features for STM32 microcontrollers and microprocessors.

Figure 2.8
STM32 Cube IDE

3- **Visual Studio IDE**

Microsoft Visual Studio is an integrated development environment from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps.

Figure 2.9
Visual Studion

4- **PyCharm IDE**

PyCharm is an integrated development environment used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains.

Figure 2.10
PyCharm

5- **Visual Studio Code**

Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

Figure 2.11
VS Code

6- **XCode**

Xcode is Apple's integrated development environment for macOS, used to develop software for macOS, iOS, iPadOS, watchOS, and tvOS. It was first released in 2003;

Figure 2.12
X-Code

7- **Apache NetBeans IDE**

NetBeans is an integrated development environment for Java. NetBeans allows applications to be developed from a set of modular software components called modules. NetBeans runs on Windows, macOS, Linux and Solaris.

Figure 2.13
Apache Netbeans

8- **Apache HTTP Server**

The Apache HTTP Server, colloquially called Apache, is a free and open-source cross-platform web server software, released under the terms of Apache License 2.0. Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation.

Figure 2.14 Apache HTTP Server

9- **Matlab**

MATLAB is a proprietary multi-paradigm programming language and numeric computing environment developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages.pices of the Apache Software Foundation.

Figure 2.15
Matlab

10- **CARLA Simulator**

CARLA is an open-source simulator for autonomous driving research. CARLA has been developed from the ground up to support development, training, and validation of autonomous driving systems. In addition to open-source code and protocols, CARLA provides open digital assets (urban layouts, buildings, vehicles) that were created for this purpose and can be used freely. The simulation platform supports flexible specification of sensor suites and environmental conditions.

Figure 2.16
Carla Simulator

10- **ST Link Utility**

STM32 ST-LINK Utility (STSW-LINK004) is a full-featured software interface for programming STM32 microcontrollers. STM32 ST-LINK Utility photoIt provides an easy-to-use and efficient environment for reading, writing and verifying a memory device.

11- **Microsoft Azure Cloud Service**

Microsoft Azure, commonly referred to as Azure, is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through Microsoft-managed data centers.

Figure 2.17
Microsoft Azure

12- **Proteus**

The Proteus Design Suite is a proprietary software tool suite used primarily for electronic design automation. The software is used mainly by electronic design engineers and technicians to create schematics and electronic prints for manufacturing printed circuit boards.


Figure 2.18
Proteus

13- **Eagle**

EAGLE is electronic design automation (EDA) software that lets printed circuit board (PCB) designers seamlessly connect schematic diagrams, component placement, PCB routing, and comprehensive library content.

Figure 2.19
Eagle Cad

2.3 Hardware Tools

In our project, we used:

- various ARM-based microcontrollers that control our system such as, STM32F103C8T6, STM32F446RE, and Raspberry Pi 3 Model B.
- STLINK V2.
- ESP8266 Wi-Fi microchip.
- CAN Transceivers so that these microcontrollers can communicate with each other over the CAN communication protocol.

Those Choices was based on a study of the hardware criteria to make sure each unit of hardawre can do what is meant to be done.

Hardware Criteria

- Power efficiency
- Security
- Processing power
- Hardware interface
- Cost
- Temperature tolerance
- Hardware architecture
- Memory
- Software architecture

STM32F103C8T6 Specs

STM32F103C8T6 (known as Blue Pill Board) incorporates the high-performance ARM® Cortex®-M3 32-bit RISC core

- Operating at a 72 MHz frequency
- High-speed embedded memories (Flash memory up to 128 Kbytes and SRAM up to 20 Kbytes)
- Extensive range of enhanced I/Os and peripherals connected to two APB buses.
- It offers two 12-bit ADCs
- 3 general-purpose 16-bit timers plus one PWM timer
- Standard and advanced communication interfaces: up to two I2Cs and SPIs, three USARTs, a USB, and a CAN.


Figure 2.20
STM32F103C8T6

STM32F446RE Specs

STM32F446RE (known as Nucleo Board) is based on the high-performance Arm® Cortex®-M4 32-bit RISC core

- Operating at a frequency of up to 180 MHz.
- The Cortex-M4 core features a floating-point unit (FPU) single precision supporting all Arm® single-precision data-processing instructions and data types.
- It also implements a full set of DSP instructions and a memory protection unit (MPU) that enhances application security.
- It also incorporates high-speed embedded memories (Flash memory up to 512 Kbytes, up to 128 Kbytes of SRAM), up to 4 Kbytes of backup SRAM
- Extensive range of enhanced I/Os and peripherals connected to two APB buses, two AHB buses, and a 32-bit multi-AHB bus matrix.

All devices offer three 12-bit ADCs, two DACs

- low-power RTC
- 12 general-purpose 16-bit timers including two PWM timers for motor control, two general-purpose 32-bit timers.
- They also feature standard and advanced communication interfaces: up 4× I2C interfaces, up to four USARTs and SPIs, 2× CAN (2.0B Active).

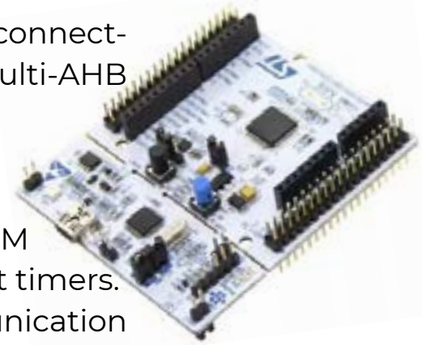


Figure 2.21
STM32F446RE

Raspberry Pi 3 Model B Specs

Raspberry Pi 3 Model B is based on the high-performance Arm® Cortex®-A53 64-bit quad core ARM Cortex-A53 processor operating at 1.2 GHz. The RPi3 also has on-board 802.11n Wi-Fi, Bluetooth, and USB boot capabilities. The RPi3 SoC is Broadcom BCM2837A0 which comes with a 1 GB SDRAM, 512 KB shared L2 cache, Broadcom VideoCore IV operates at 250MHz, 4 USB 2.0 ports, Audio output Analog via 3.5 mm phone jack; digital via HDMI, 10/100 Mbps Ethernet Port, and some low-level peripherals such as, 17x GPIO that can also used as the following: USART, I2C bus, SPI Bus with two chip selects, and I²S audio.

Figure 2.22
Raspbery PI

The Operating System (OS) That used for the RPi3 is Raspbian which is a free operating system based on Debian optimized for the Raspberry Pi hardware. An operating system is the set of basic programs and utilities that make your Raspberry Pi run. However, Raspbian provides more than a pure OS: it comes with over 35,000 packages, pre-compiled software bundled in a nice format for easy installation on your Raspberry Pi.

ESP8266 Sepcs

The ESP8266 is a Wi-Fi microchip, with a full TCP/IP stack and microcontroller capability. L106 32-bit RISC microprocessor core based on the Tensilica Xtensa Diamond Standard 106Micro running at 80 MHz with 32 KiB instruction RAM, 80 KiB user-data RAM, IEEE 802.11 b/g/n Wi-Fi, 17 GPIO pins, SPI, I²C, UART on dedicated pins, and more. However, and in our project, we used the ESP8266 as a Wi-Fi Module so that the main microcontroller can communicate with our Cloud.

Figure 2.23
ESP8266

STLINK V2

The ST-LINK/V2 is an in-circuit debugger and programmer for the STM8 and STM32 microcontrollers. The single-wire interface module (SWIM) and JTAG/serial wire debugging (SWD) interfaces are used to communicate with any STM8 or STM32 microcontroller located on an application board.

Figure 2.24
STLINK V2

CAN Transceivers

Typically, each node in a CAN system must have a device to convert the digital signals generated by a CAN controller to signals suitable for transmission over the bus cabling (differential output).

In our project, we used one MCP2551 High-Speed CAN Transceiver that serves as the interface between a CAN protocol controller and the physical bus. The MCP2551 device provides differential transmit and receive capability for the CAN protocol controller and is fully compatible with the ISO-11898 standard, including 24V requirements. It will operate at speeds of up to 1 Mb/s.

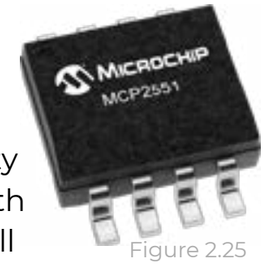


Figure 2.25
MCP2551

Also, For RPi3, we used Microchip Technology's MCP2515 stand-alone Controller Area Network (CAN) controller that implements the CAN specification, Version 2.0B. It is capable of transmitting and receiving both standard and extended data and remote frames. The MCP2515 has two acceptance masks and six acceptance filters that are used to filter out unwanted messages, thereby reducing the host MCU's overhead. The MCP2515 interfaces with the RPi3 via an industry standard Serial Peripheral Interface (SPI).

Figure 2.26
MCP2515

Bosch Steering Angle Sensor

This sensor is designed to measure rotational movement and angular speed, e.g. steering wheel angle and steering wheel speed. In order to achieve this, the sensor is using the giant magneto resistive (GMR) effect. The detection of the absolute angle is realized by means of toothed measuring gears with different ratio including small magnets. Corresponding GMR elements that change their electrical resistance according to the magnetic field direction detects the angle position of the measuring gears.
The measured voltages are A/D converted and a microcontroller performs the angle calculations. The steering angle and the steering angle speed are provided on a CAN-interface.

Figure 2.27
Bosch Steering Angle

Features

- Steering Wheel Angle: ± 780°
- Angular Speed: 0 to 1,016°/s
- 500 kbaud CAN-output

Yaw Rate Sensor YRS 3

This sensor is designed to measure the physical effects of yawing, lateral and longitudinal acceleration. In order to achieve this, the sensor features both a measuring element for yaw rate and two for acceleration, with one appropriate integrated circuit. A rotation around the third orthogonal axis, a yaw rate, creates a Coriolis force on the accelerometers, which is detected by the element. Apart from the measuring element for yaw rate, a pure surface micro machined measuring element for acceleration is utilized to measure the vehicles lateral and longitudinal acceleration. This enables a very precise application. The main feature and benefit of this sensor is its wide measuring range, the standardized 1 Mbaud/s CAN-signal output and the combination of high quality production part and robust design.

Features

- Yaw rate and acceleration measurement
- CAN output 15 Hz
- low-pass filtered
- Measurement ranges: ±4.1 g; ±160°/s

Figure 2.28 YRS3

Speed Sensor Hall-Effect HA-P

This sensor is designed for incremental measurement of rotational speed (e.g. camshaft or wheel speed). Due to the rotation of a ferromagnetic target wheel in front of the HA-P, the magnetic field is modulated at the place of the Hall probe. A Hall-effect sensor element with integrated signal conditioning circuit detects this change and generates a digital output signal. The main feature and benefit of this sensor is the combination of a high quality production part and robust design with metal housing.

Features

- Max. frequency: ≤ 10 kHz
- Air gap: 0.5 to 1.0 mm
- Bore diameter: 18 mm
- Max. vibration: 1,000 m/s² at 10 Hz to 2 kHz
- Weight w/o wire: 70 g

Figure 2.29 Speed Sensor Hall Effect

Tire Pressure Monitoring Sensor SP30

The SP30 Tire Pressure Monitoring (TPM) Sensor represents Infineon's standard pressure range TPM sensor. The SP30 offers a high level of integration by including a microcontroller, signal conditioning and LF-input stage to meet market demands for flexible, customer specific solutions and overall system cost reduction. The sensor design is based on Infineon's proprietary and patented solutions for high reliability measurements in harsh automotive environments. Its predictable and stable quality is proven in high volume applications. The SP30 measures pressures up to 900kPa, temperature, supply voltage and acceleration (optional), and by integrating these functions with an ASIC in one package, Infineon has developed the ideal product for standard pressure TPM applications.

Features

· Integrated Sensors

- Pressure
- Acceleration (optional)
- Temperature
- Voltage

· Integrated Peripherals

- Microcontroller
- On board EEPROM
- GPIOs
- ADC for signal conditioning
- 2x LF Receiver for triggering

· Measurement Ranges

- Pressure Sensor 100 to 450 kPa / 100 to 900kPa
- Temperature Sensor -40 to +125°C
- Supply Voltage Sensor 2.1 to 3.6 V
- Acceleration Sensor -12 to 115 g

Figure 2.30 SP30

SCC1300-D02 Combined Gyroscope and 3-axis Accelerometer with digital SPI interfaces

Figure 2.31 SCC1300-D02

The SCC1300-D02 is a combined high performance gyroscope and accelerometer component. The sensor is based on Murata's proven capacitive 3D-MEMS technology. The component integrates angular rate and acceleration sensing together with flexible separate digital SPI interfaces. The small robust packaging guarantees reliable operation over the

product's lifetime. The housing is suitable for SMD mounting. The component is compatible with RoHS and ELV directives. The SCC1300-D02 is designed, manufactured and tested for high stability, reliability and quality requirements. The angular rate and acceleration sensors provide highly stable output over wide ranges of temperature and mechanical noise. The angular rate sensor bias stability is in the elite of MEMS gyros. It is also exceptionally insensitive to all mechanical vibrations and shocks. The component has several advanced self diagnostics features.

Features

- ±100 °/s angular rate measurement range
- ±2 g 3-axis acceleration measurement range
- Angular rate measurement around X axis
- Angular rate sensor exceptionally insensitive to mechanical vibrations and shocks
- Superior bias stability for MEMS gyroscopes (<1°/h)
- Digital SPI interfacing
- Enhanced self diagnostics features
- Small size: 8.5 x 18.7 x 4.5 mm (w x l x h)
- RoHS compliant robust packaging suitable for leadfree soldering process and SMD mounting
- Proven capacitive 3D-MEMS technology
- Temperature range -40 °C...+125 °C

2.4 Communication Protocols

A communication protocol is a system of rules that allows two or more entities of a communications system to transmit information via any kind of variation of a physical quantity. The protocol defines the rules, syntax, semantics and synchronization of communication and possible error recovery methods. Protocols may be implemented by hardware, software, or a combination of both.

CAN(Controller Area Network)

Controller area network is an electronic communication bus defined by the ISO 11898 standards. Those standards define how communication happens, how wiring is configured and how messages are constructed, among other things. Collectively, this system is referred to as a CAN bus.

The CAN bus is a broadcast type of bus. This means that all nodes can "hear" all transmissions. There is no way to send a message to just a specific node; all nodes will invariably pick up all traffic. The CAN hardware, however, provides local filtering so that each node may react only on the interesting messages.

The CAN standard defines four different message types. The messages uses a clever scheme of bit-wise arbitration to control access to the bus, and each message is tagged with a priority.

The CAN messages

CAN uses short messages – the maximum utility load is 94 bits. There is no explicit address in the messages; instead, the messages can be said to be contents-addressed, that is, their contents implicitly determines their address.

Message Types

There are four different message types (or 'frames') on a CAN bus:

1-The Data Frame

2-The Remote Frame

3-The Error Frame

4-The Overload Frame

The Data Frame

The Data Frame is the most common message type. It comprises the following major parts (a few details are omitted for the sake of brevity):

the Arbitration Field, which determines the priority of the message when two or more nodes are contending for the bus. The Arbitration Field contains:

For CAN 2.0A, an 11-bit Identifier and one bit, the RTR bit, which is dominant for data frames.

For CAN 2.0B, a 29-bit Identifier (which also contains two recessive bits: SRR and IDE) and the RTR bit.

the Data Field, which contains zero to eight bytes of data.

the CRC Field, which contains a 15-bit checksum calculated on most parts of the message. This checksum is used for error detection.

an Acknowledgement Slot; any CAN controller that has been able to correctly receive the message sends an Acknowledgement bit at the end of each message. The transmitter checks for the presence of the Acknowledge bit and retransmits the message if no acknowledge was detected.

Note 1: It is worth noting that the presence of an Acknowledgement Bit on the bus does not mean that any of the intended addressees has received the message. The only thing we know is that one or more nodes on the bus has received it correctly.

Note 2: The Identifier in the Arbitration Field is not, despite of its name, necessarily identifying the contents of the message.

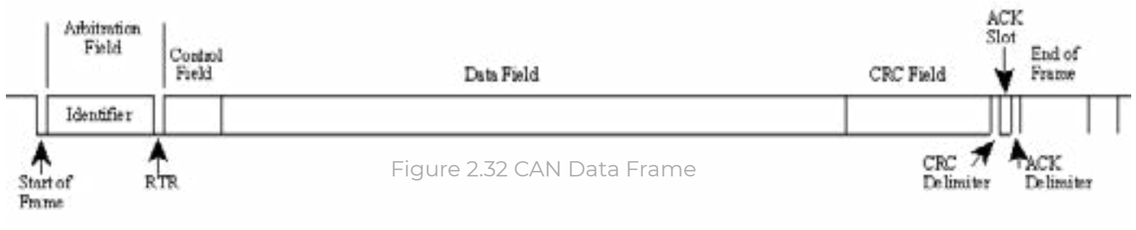


Figure 2.32 CAN Data Frame

The Remote Frame

The Remote Frame is just like the Data Frame, with two important differences: it is explicitly marked as a Remote Frame (the RTR bit in the Arbitration Field is recessive), and there is no Data Field. The intended purpose of the Remote Frame is to solicit the transmission of the corresponding Data Frame. If, say, node A transmits a Remote Frame with the Arbitration Field set to 234, then node B, if properly initialized, might respond with a Data Frame with the Arbitration Field also set to 234. Remote Frames can be used to implement a request-response type of bus traffic management. In practice, however, the Remote Frame is little used. It is also worth noting that the CAN standard does not prescribe the behaviour outlined here. Most CAN controllers can be programmed either to automatically respond to a Remote Frame, or to notify the local CPU instead. There's one catch with the Remote Frame: the Data Length Code must be set to the length of the expected response message. Otherwise the arbitration will not work. Sometimes it is claimed that the node responding to the Remote Frame is starting its transmission as soon as the identifier is recognized, thereby "filling up" the empty Remote Frame. This is not the case.

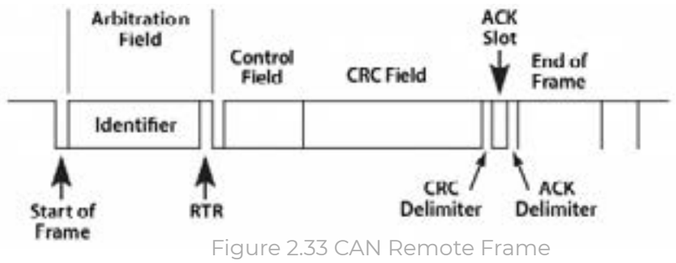


Figure 2.33 CAN Remote Frame

The Error Frame

Simply put, the Error Frame is a special message that violates the framing rules of a CAN message. It is transmitted when a node detects a fault and will cause all other nodes to detect a fault – so they will send Error Frames, too. The transmitter will then automatically try to retransmit the message. There is an elaborate scheme of error counters that ensures that a node can't destroy the bus traffic by repeatedly transmitting Error Frames.

The Error Frame consists of an Error Flag, which is 6 bits of the same value (thus violating the bit-stuffing rule) and an Error Delimiter, which is 8 recessive bits. The Error Delimiter provides some space in which the other nodes on the bus can send their Error Flags when they detect the first Error Flag.

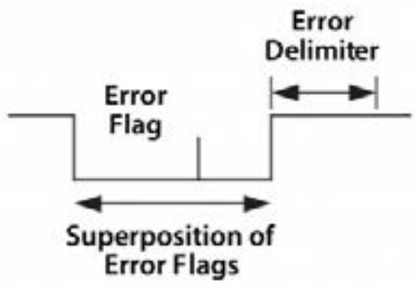


Figure 2.34 CAN Error Frame

The Overload Frame

The Overload Frame is mentioned here just for completeness. It is very similar to the Error Frame with regard to the format and it is transmitted by a node that becomes too busy. The Overload Frame is not used very often, as today's CAN controllers are clever enough not to use it. In fact, the only controller that will generate Overload Frames is the now obsolete 82526.

Bus Arbitration and Message Priority

The message arbitration (the process in which two or more CAN controllers agree on who is to use the bus) is of great importance for the actually available bandwidth for data transmission.

Any CAN controller may start a transmission when it has detected an idle bus. This may result in two or more controllers starting a message (almost) at the same time.

The conflict is resolved in the following way. The transmitting nodes monitor the bus while they are sending. If a node detects a dominant level when it is sending a recessive level itself, it will immediately quit the arbitration process and become a receiver instead. The arbitration is performed over the whole Arbitration Field and when that field has been sent, exactly one transmitter is left on the bus. This node continues the transmission as if nothing had happened. The other potential transmitters will try to retransmit their messages when the bus becomes available next time. No time is lost in the arbitration process. An important condition for this bit-wise arbitration to succeed is that no two nodes may transmit the same Arbitration Field. There is one exception to this rule: if the message contains no data, then any node may transmit that message.

Since the bus is wired-and and a Dominant bit is logically 0, it follows that the message with the numerically lowest Arbitration Field will win the arbitration.

Message Addressing and Identification

It is worth noting once again that there is no explicit address in the CAN messages. Each CAN controller will pick up all traffic on the bus, and using a combination of hardware filters and software, determine if the message is “interesting” or not.

In fact, there is no notion of message addresses in CAN. Instead, the contents of the messages is identified by an identifier which is present somewhere in the message. CAN messages are said to be “contents-addressed”.

A conventional message address would be used like “Here’s a message for node X”. A contents-addressed message is like “Here’s a message containing data labeled X”. The difference between these two concepts is small but significant.

The contents of the Arbitration Field is, per the Standard, used to determine the message’s priority on the bus. All CAN controllers will also use the whole (some will use just a part) of the Arbitration Field as a key in the hardware filtration process.

The Standard does not say that the Arbitration Field must be used as a message identifier. It’s nevertheless a very common usage.

CAN Physical Layers

The CAN Bus

The CAN bus uses Non-Return To Zero (NRZ) with bit-stuffing. There are two different signaling states: dominant (logically 0) and recessive (logically 1). These correspond to certain electrical levels which depend on the physical layer used (there are several.) The modules are connected to the bus in a wired-and fashion: if just one node is driving the bus to the dominant state, then the whole bus is in that state regardless of the number of nodes transmitting a recessive state.

Different Physical Layers

A physical layer defines the electrical levels and signaling scheme on the bus, the cable impedance and similar things.

There are several different physical layers:
The most common type is the one defined by the CAN standard, part ISO 11898-2, and it’s a two-wire balanced signaling scheme. It is also sometimes known as “high-speed CAN”.
Another part of the same ISO standard, ISO 11898-3, defines another two-wire balanced signaling scheme for lower bus speeds. It is fault tolerant, so the signaling can continue even if one bus wire is cut or shorted to ground or Vbat. It is sometimes known as “low-speed CAN”.
SAE J2411 defines a single-wire (plus ground, of course) physical layer. It’s used chiefly in cars – e.g. GM-LAN.
Several proprietary physical layers do exist.
Modifications of RS485 were used in the Old Ages when CAN drivers didn’t exist.
Different physical layers can not, as a rule, interoperate. Some combinations may work, or seem to work, under good conditions. For example, using both “high-speed” and “low-speed” transceivers on the same bus can work … sometimes.

A great many CAN transceiver chips are manufactured by NXP; alternative vendors include Bosch, Infineon, Texas Instruments and Vishay Siliconix.
A very common type is the 82C250 transceiver which implements the physical layer defined by ISO 11898. The 82C251 is an improved version.
A common transceiver for “low-speed CAN” is TJA1054 from NXP.

Maximum Bus Speed
The maximum speed of a CAN bus, according to the standard, is 1 Mbit/second. Some CAN controllers will nevertheless handle higher speeds than 1Mbit/s and may be considered for special applications.
Low-speed CAN (ISO 11898-3, see above) can go up to 125 kbit/s.
Single-wire CAN can go up to around 50 kbit/s in its standard mode and, using a special high-speed mode used e.g. for ECU programming, up to around 100 kbit/s.

Minimum Bus Speed
Be aware that some bus transceivers will not allow you to go below a certain bit rate. For example, using 82C250 or 82C251 you can go down to 10 kbit/s without problems, but if you use the TJA1050 instead you can't go below around 50 kbit/s. Check the data sheet.

Maximum Cable Length
At a speed of 1 Mbit/s, a maximum cable length of about 40 meters (130 ft.) can be used. This is because the arbitration scheme requires that the wave front of the signal be able to propagate to the most remote node and back again before the bit is sampled. In other words, the cable length is restricted by the speed of light. A proposal to increase the speed of light has been considered but was turned down because of its inter-galactic consequences.
Other maximum cable lengths are (these values are approximate):
100 meters (330 ft) at 500 kbit/s
200 meters (650 ft) at 250 kbit/s
500 meters (1600 ft) at 125 kbit/s
6 kilometers (20000 ft) at 10 kbit/s
If optocouplers are used to provide galvanic isolation, the maximum bus length is decreased accordingly. Hint: use fast optocouplers, and look at the delay through the device, not at the specified maximum bit rate.

Bus Termination
An ISO 11898 CAN bus must be terminated. This is done using a resistor of 120 Ohms in each end of the bus. The termination serves two purposes:

Remove the signal reflections at the end of the bus.
Ensure the bus gets correct DC levels.

An ISO 11898 CAN bus must always be terminated regardless of its speed. I'll repeat this: an ISO 11898 CAN bus must always be terminated regardless of its speed. For laboratory work just one terminator might be enough. If your CAN bus works even though you haven't put any terminators on it, you are just lucky.

Note that other physical layers, such as "low-speed CAN", single-wire CAN, and others, may or may not require termination. But your vanilla high-speed ISO 11898 CAN bus will always require at least one terminator.

The Cable
The ISO 11898 prescribes that the cable impedance be nominally 120 Ohms, but an impedance in the interval of [108..132] Ohms is permitted.
There are not many cables in the market today that fulfill this requirement. There is a good chance that the allowed impedance interval will be broadened in the future.
ISO 11898 is defined for a twisted pair cable, shielded or unshielded. Work is in progress on the single-wire standard SAE J2411.

CAN connectors
There is no standard at all for CAN bus connectors! Usually, each Higher Layer Protocol(!) defines one or a few preferred CAN bus connector types. Common types include
- 9-pin DSUB, proposed by CiA.
- 5-pin Mini-C and/or Micro-C, used by DeviceNet and SDS.
- 6-pin Deutch connector, proposed by CANHUG for mobile hydraulics.

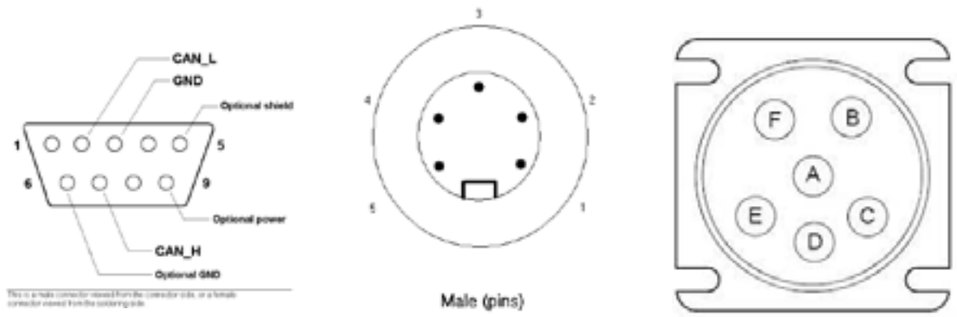


Figure 2.35 CAN Connectors

Error Detection Mechanisms

The CAN protocol defines no less than five different ways of detecting errors. Two of these works at the bit level, and the other three at the message level.

- Bit Monitoring.
- Bit Stuffing.
- Frame Check.
- Acknowledgement Check.
- Cyclic Redundancy Check.
- Bit Monitoring

Each transmitter on the CAN bus monitors (i.e. reads back) the transmitted signal level. If the bit level actually read differs from the one transmitted, a Bit Error is signaled. (No bit error is raised during the arbitration process.)

Bit Stuffing

When five consecutive bits of the same level have been transmitted by a node, it will add a sixth bit of the opposite level to the outgoing bit stream. The receivers will remove this extra bit. This is done to avoid excessive DC components on the bus, but it also gives the receivers an extra opportunity to detect errors: if more than five consecutive bits of the same level occurs on the bus, a Stuff Error is signaled.

Frame check

Some parts of the CAN message have a fixed format, i.e. the standard defines exactly what levels must occur and when. (Those parts are the CRC Delimiter, ACK Delimiter, End of Frame, and also the Intermission, but there are some extra special error checking rules for that.) If a CAN controller detects an invalid value in one of these fixed fields, a Form Error is signaled.

Acknowledgement Check

All nodes on the bus that correctly receives a message (regardless of their being “interested” of its contents or not) are expected to send a dominant level in the so-called Acknowledgement Slot in the message. The transmitter will transmit a recessive level here. If the transmitter can’t detect a dominant level in the ACK slot, an Acknowledgement Error is signaled.

Cyclic Redundancy Check

Each message features a 15-bit Cyclic Redundancy Checksum (CRC), and any node that detects a different CRC in the message than what it has calculated itself will signal an CRC Error.

Error Confinement Mechanisms

Every CAN controller along a bus will try to detect the errors outlined above within each message. If an error is found, the discovering node will transmit an Error Flag, thus destroying the bus traffic. The other nodes will detect the error caused by the Error Flag (if they haven’t already detected the original error) and take appropriate action, i.e. discard the current message.

Each node maintains two error counters: the Transmit Error Counter and the Receive Error Counter. There are several rules governing how these counters are incremented and/or decremented. In essence, a transmitter detecting a fault increments its Transmit Error Counter faster than the listening nodes will increment their Receive Error Counter. This is because there is a good chance that it is the transmitter who is at fault!

A node starts out in Error Active mode. When any one of the two Error Counters raises above 127, the node will enter a state known as Error Passive and when the Transmit Error Counter raises above 255, the node will enter the Bus Off state.

An Error Active node will transmit Active Error Flags when it detects errors.
An Error Passive node will transmit Passive Error Flags when it detects errors.
A node which is Bus Off will not transmit anything on the bus at all.
The rules for increasing and decreasing the error counters are somewhat complex, but the principle is simple: transmit errors give 8 error points, and receive errors give 1 error point. Correctly transmitted and/or received messages causes the counter(s) to decrease.

Example (slightly simplified): Let’s assume that node A on a bus has a bad day. Whenever A tries to transmit a message, it fails (for whatever reason). Each time this happens, it increases its Transmit Error Counter by 8 and transmits an Active Error Flag. Then it will attempt to retransmit the message.. and the same thing happens.

When the Transmit Error Counter raises above 127 (i.e. after 16 attempts), node A goes Error Passive. The difference is that it will now transmit Passive Error Flags on the bus. A Passive Error Flag comprises 6 recessive bits, and will not destroy other bus traffic – so the other nodes will not hear

A complaining about bus errors. However, A continues to increase its Transmit Error Counter. When it raises above 255, node A finally gives in and goes Bus Off.
What does the other nodes think about node A? – For every active error flag that A transmitted, the other nodes will increase their Receive Error Counters by 1. By the time that A goes Bus Off, the other nodes will have a count in their Receive Error Counters that is well below the limit for Error Passive, i.e. 127. This count will decrease by one for every correctly received message. However, node A will stay bus off.

Most CAN controllers will provide status bits (and corresponding interrupts) for two states:
“Error Warning” – one or both error counters are above 96
Bus Off, as described above.
Some – but not all! – controllers also provide a bit for the Error Passive state. A few controllers also provide direct access to the error counters.

The CAN controller’s habit of automatically retransmitting messages when errors have occurred can be annoying at times. There is at least one controller on the market (the SJA1000 from Philips) that allows for full manual control of the error handling.

Bus Failure Modes
The ISO 11898 standard enumerates several failure modes of the CAN bus cable:
CAN_H interrupted
CAN_L interrupted
CAN_H shorted to battery voltage
CAN_L shorted to ground
CAN_H shorted to ground
CAN_L shorted to battery voltage
CAN_L shorted to CAN_H wire
CAN_H and CAN_L interrupted at the same location
Loss of connection to termination network
For failures 1-6 and 9, it is “recommended” that the bus survives with a reduced S/N ratio, and in case of failure 8, that the resulting subsystem survives. For failure 7, it is “optional” to survive with a reduced S/N ratio.
failures 1-7, and may or may not survive failures 8-9.

2.5 Cloud Computing

Cloud Computing is the delivery of computing services such as servers, storage, databases, networking, software, analytics, intelligence, and more, over the Cloud (Internet).

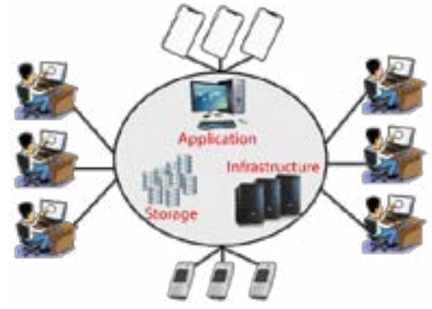


Figure 2.36 overview of cloud computing

The cloud service provider is responsible for the hardware purchase and maintenance. They also provide a wide variety of software and platform as a service. We can take any required services on rent. The cloud computing services will be charged based on usage.

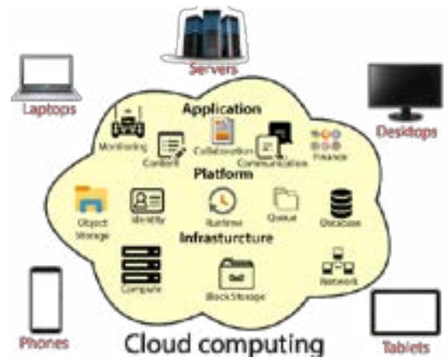


Figure 2.37 resources of cloud

The cloud environment provides an easily accessible online portal that makes handy for the user to manage the compute, storage, network, and application resources. Some cloud service providers are in the following figure.

Figure 2.38 cloud service providers

Advantages of cloud computing

1. Cost: It reduces the huge capital costs of buying hardware and software.
2. Speed: Resources can be accessed in minutes, typically within a few clicks.
3. Scalability: We can increase or decrease the requirement of resources according to the business requirements.
4. Productivity: While using cloud computing, we put less operational effort. We do not need to apply patching, as well as no need to maintain hard ware and software. So, in this way, the IT team can be more productive and focus on achieving business goals.
5. Reliability: Backup and recovery of data are less expensive and very fast for business continuity.
6. Security: Many Cloud service providers
7. offer a broad set of policies, technologies, and controls that strengthen our data security.

Types of Cloud Services

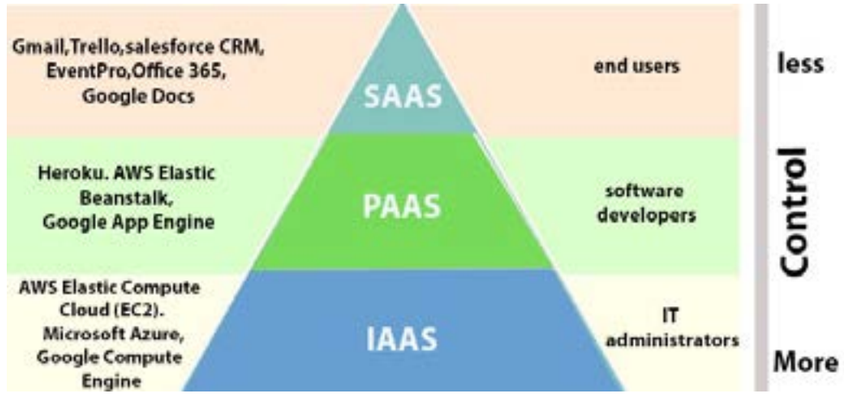


Figure 2.39 Cloud computing services

1. Infrastructure as a Service (IaaS): In IaaS, we can rent IT infrastructures like servers and virtual machines (VMs), storage, networks, operating systems from a cloud service vendor. We can create VM running Windows or Linux and install anything we want on it. Using IaaS, we don't need to care about the hardware or virtualization software, but other than that, we do have to manage everything else. Using IaaS, we get maximum flexibility, but still, we need to put more effort into maintenance.

2. Platform as a Service (PaaS): This service provides an on-demand environment for developing, testing, delivering, and managing software applications. The developer is responsible for the application, and the PaaS vendor provides the ability to deploy and run it. Using PaaS, the flexibility gets reduce, but the management of the environment is taken care of by the cloud vendors.
3. Software as a Service (SaaS): It provides a centrally hosted and managed software services to the end-users. It delivers software over the internet, on-demand, and typically on a subscription basis. E.g., Microsoft One Drive, Dropbox, WordPress, Office 365, and Amazon Kindle. SaaS is used to minimize the operational cost to the maximum extent.

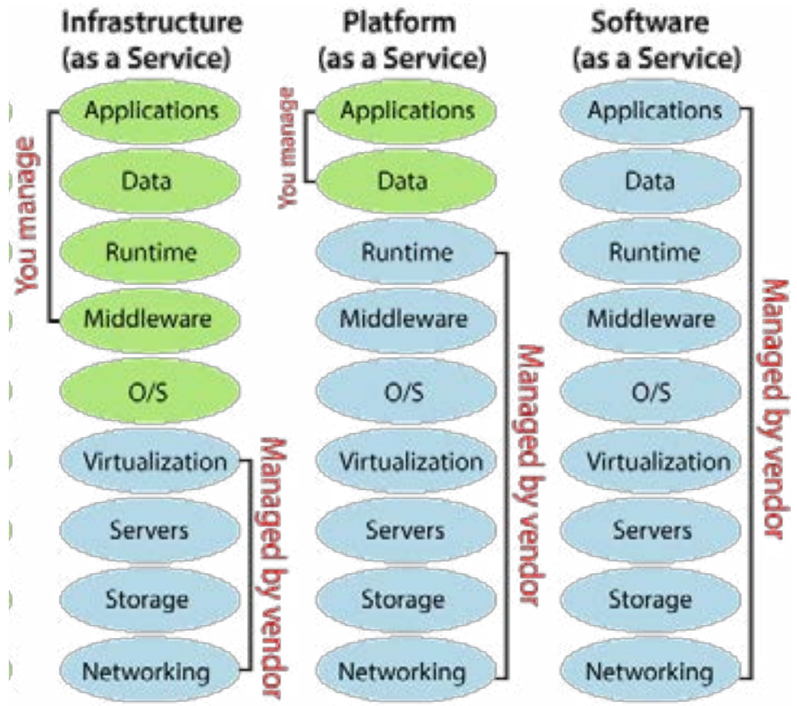


Figure 2.40 Accessibilities of services

2.6 Database

A database is an organized collection of data, generally stored and accessed electronically from a computer system. It supports the storage and of data. In other words, databases are used by an organization as a method of storing, managing, and retrieving information.

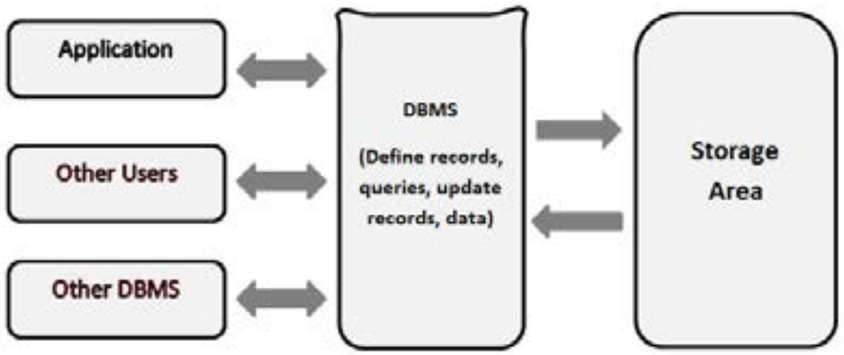


Figure 2.41 Database management system

Advantages of using Databases

• There are many advantages of databases
• Reduced data redundancy
• Reduced updating errors and increased consistency
• Greater data integrity and independence from application programs
• Improved data access to users through the use of host and query languages
• Improved data security
• Reduced data entry, storage, and retrieval costs

Disadvantages of using Databases

• There are many disadvantages of databases
• Although databases allow businesses to store and access data efficiently, they also have certain disadvantages
• Complexity
• Cost
• Security
• Compatibility
• Some examples of Databases

Some of the most popular databases are

• Oracle Database
• Sybase
• MySQL

2.7 GUI Applications

A graphical user interface is an application that has buttons, windows, and lots of other widgets that the user can use to interact with your application. A good example would be a web browser. It has buttons, tabs, and a main window where all the content loads.

Our project needed more than GUI Application, those applications were written in multiple languages like python, java, c#.

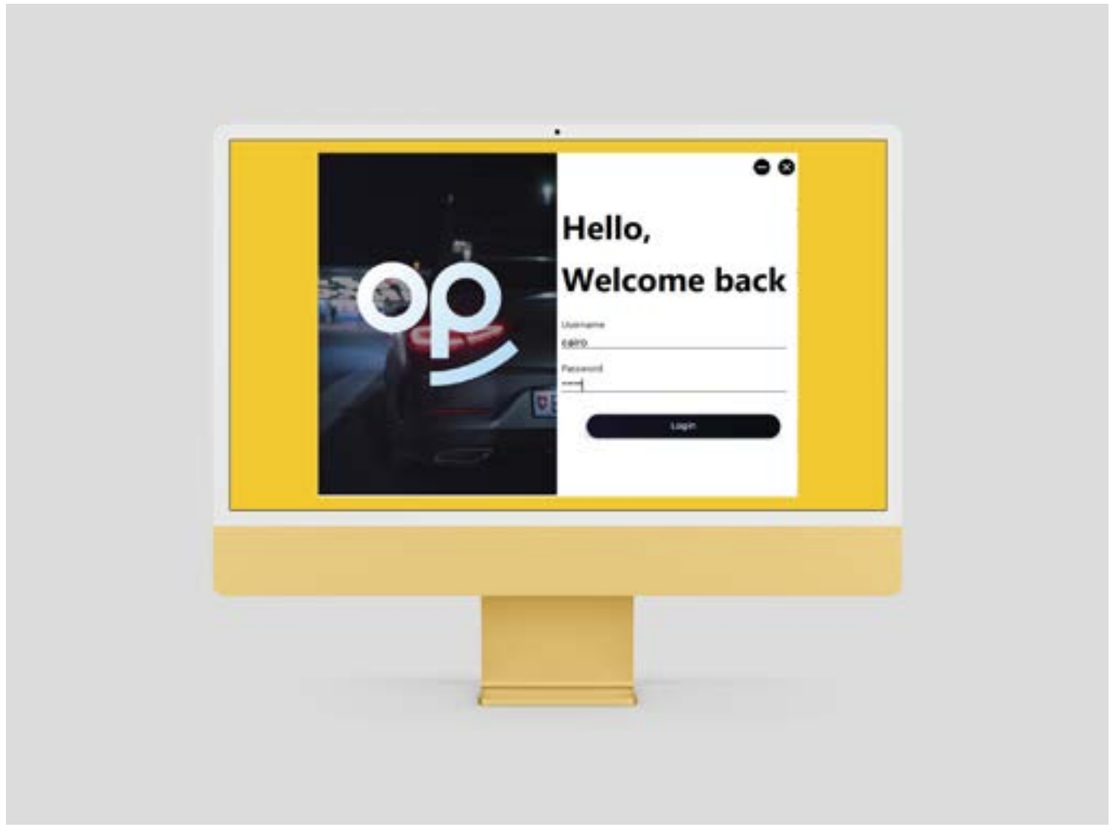


Figure 2.42 GUI Login

Chapter 3
Monitoring

3.1 Sensors

Sensors enable greater degrees of vehicle automation and futuristic designs. Sensors monitor vehicle engines, fuel consumption, and emissions, along with aiding and protecting drivers and passengers. These allow car manufacturers to launch cars that are safer, more fuel-efficient, and comfortable to drive. Sensors in our system play an important role in the control and take the right decision from the control units, and also in providing a new system based on its reading that is able to protect the car more in the future.

In this section, we discuss the different sensors that are used in the system. Modern cars make thousands of decisions based on the data provided by various sensors that are interfaced with the vehicles onboard computer systems. onboard computer systems unit (OBU) It is not in our interest now, as we explained previously that if the system works inside a car that operates with onboard computer systems We provide this data to it to make the right decision, but if it does not contain this system(OBU), we are satisfied with sending it to OEM, In both cases, we analyze this data to become information capable of warning the driver.

A car system consists of a wide range of sensor devices working together, many of these sensors operate in rough and harsh conditions that involve extreme temperatures, vibrations, and exposure to environmental contaminants. Yet, these provide vital data parameters to the electronic control unit (ECU) that governs the various engine functions effectively. Modern vehicles are built with complex electronic sensor systems. Digital computers now control engines through various sensors. Luxury cars have a multitude of sensors for controlling various features.

TARGETED SENSORS IN OUR SYSTEM

Our system is based on a set of sensors that work with a computer to improve vehicle stability by detecting and reducing the loss of traction. The most important sensor is the Yaw and Roll rate sensor, Wheel speed sensor, Steering angle sensor, and Tire-Pressure Sensor.

STEERING ANGLE SENSOR (SAS)

Steering is a system that allows the driver to guide the vehicle so the steering angle sensor is located within the steering column, the steering angle sensor always has more than one sensor packaged together in a single unit for redundancy, accuracy, and diagnostics. the main purpose of the steering angle sensor is to determine where the driver wants to steer, matching the steering wheel with the vehicle's wheels.

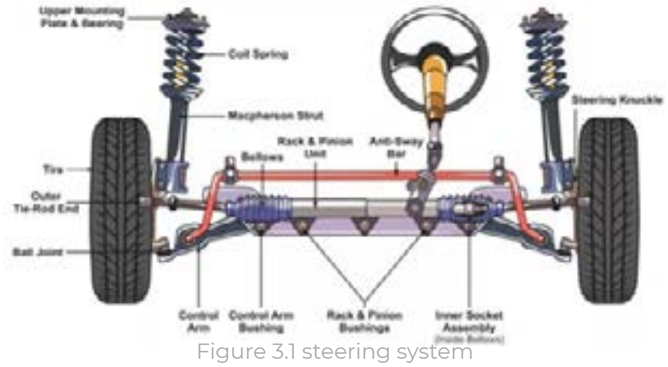


Figure 3.1 steering system

Analog SASs are wired with a 5-volt reference, chassis ground, and signal outputs, as the steering wheel is turned, the SAS produces a signal that toggles between 0 and 5 volts as the wheel is turned 360°. It is possible to observe the 0- to 5-volt signal with meters connected to the two SAS sensors. When the wheels are straight, the sensors read 2.8V and 0.4V.

Digital steering angle sensors measure the angle and turn it into information that can be shared on a serial data bus or discrete connection with a module. Instead of changing voltage, these sensors produce a signal in code that indicates the steering angle.

With the collected information, the stability control system can be used to determine the driver's intentions, how the vehicle is reacting, and what corrections can be made.

Figure 3.2 steering angle sensor

YAW AND ROLL RATE SENSOR

Before clarifying the idea of the sensor, let's imagine the car moves freely in three dimensions and imagine three lines running through the car and intersecting at right angles at the car's center of gravity. so, the rotation around the front-to-back axis is called the roll, the rotation around the side-to-side axis is called pitch and the rotation around the vertical axis is called yaw. This is where the idea of the sensor came from, so the yaw sensor measures a vehicle's yaw rate also the roll rate.

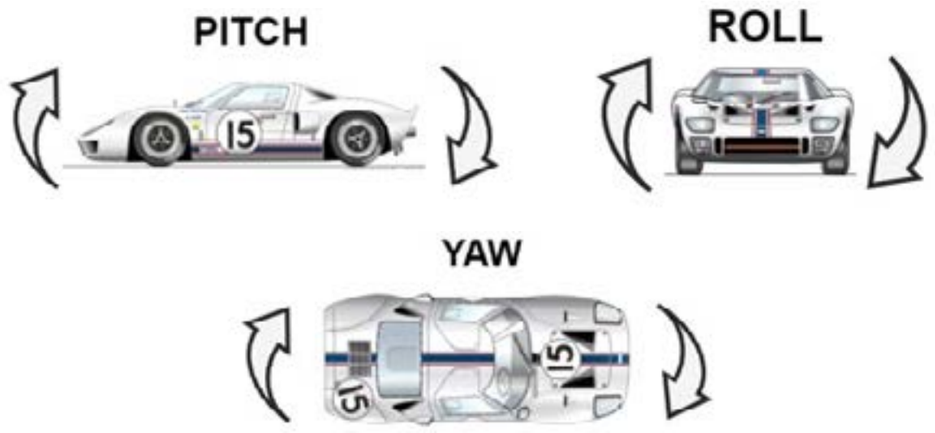


Figure 3.3 Pitch Roll Yaw Illustration

A Yaw Rate Sensor (or rotational speed sensor) measures a vehicle's angular velocity about its vertical axis in degrees or radians per second in order to determine the orientation of the vehicle as it hard-corners or threatens to roll over. In other words, the sensor determines how far off-axis a car is "tilting" in a turn. This information is then fed into ECU that compares the data with wheel speed, steering angle and if the system senses too much yaw, and send it to the UI ECU which will alert the driver and sends that information also to the Main ECU.

A Roll rate sensor also is that same idea, it's just a gyroscope which we can know the rate of change occurring on the sides of the vehicle by comparing the vehicle's actual Roll rate to the target Roll rate, we can identify to what degree the car tilted or already overturned.

Both of the previous types of sensors, the basic idea of which is based on the Gyro sensors that sense angular velocity which means the change in rotational angle per unit of time. This sensor consists of an internal vibrating element made up of crystal material in the shape of a double-T structure.

This structure comprises a stationary part in the center with ‘Sensing Arm’ attached to it and ‘Drive Arm’ on both sides. This double-T structure is symmetrical. When an alternating vibration electrical field is applied to the drive arms, continuous lateral vibrations are produced. As Drive arms are symmetrical, when one arm moves to the left the other moves to the right, thus canceling out the leaking vibrations. This keeps the stationary part at the center and the sensing arm remains static. When the external rotational force is applied to the sensor vertical vibrations are caused on Drive arms. This leads to the vibration of the Drive arms in the upward and downward directions due to which a rotational force acts on the stationary part in the center. Rotation of the stationary part leads to vertical vibrations in sensing arms. These vibrations caused in the sensing arm are measured as a change in electrical charge. This change is used to measure the external rotational force applied to the sensor as Angular rotation.

Figure 3.4 gyroscope sensor

Figure 3.5 yaw sensor

WHEEL SPEED SENSOR

For our system in vehicles, wheel speed information is essential. Therefore, we find many data from previous sensors that require processing with the data of this sensor, and here lies its importance. The wheel speed sensor is usually located at the front wheel brake disc and mounted on the hub of each wheel, but some cars are different. Some rear-wheel-drive vehicles have only an electromagnetic induction speed sensor in the main reducer or gearbox. The sensor is mounted on the main reducer housing or transmission housing, and the gear ring is mounted on the main reducer input shaft or transmission output shaft.

we can find this sensor inside cars with two types, magnetoelectric wheel speed sensors, and Hall effect wheel speed sensors.

Figure 3.6 wheel speed sensor location

Magnetoelectric speed sensors are one of the most commonly used sensor types in vehicles. The magnetoelectric wheel speed sensor is generally composed of a magnetic induction sensor head and a ring gear. The sensor head is composed of a permanent magnet, a polar shaft, an induction coil, and the like. The ring gear is a moving part, which is generally installed on the wheel hub or axle to rotate with the wheel.

The wheel speed sensor head is a stationary part, and there is a certain gap between the magnetic pole of the sensor head and the end surface of the ring gear.

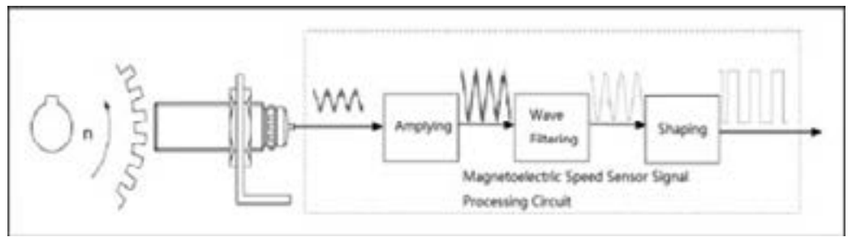


Figure 3.7 Magnetoelectric speed sensors

The magnetoelectric wheel speed sensor is composed of a permanent magnetic core and a coil. The lines of magnetic force emerge from one pole of the core, pass through the ring gear and air, and return to the other pole of the core. Since the coil of the sensor is wound on the magnetic core, these lines of magnetic force will also pass through the coil. When the wheel rotates, the ring gear (rotor) synchronized with the wheel rotates, and the teeth and gaps on the ring gear quickly pass the magnetic field of the sensor in turn. The result is that the magnetic resistance of the magnetic circuit is changed, resulting in the induction potential in the coil. Changes generate a potential pulse of a certain amplitude and frequency. The frequency of the pulse, that is, the number of pulses generated per second reflects the speed of the wheel rotation.

The Hall effect wheel speed sensor is composed of a sensing head and a ring gear. The sensor head is composed of a permanent magnet, a Hall element, and an electronic circuit. Hall effect wheel speed sensor uses the Hall effect principle, that is, the two ends of the semiconductor wafer pass to control the current, in the vertical direction of the wafer to apply the magnetic field intensity, the other two ends of the wafer will produce size and control current, magnetic induction intensity product is proportional to the potential. so, the hall element is used as the wheel speed sensor of and the magnetic induction intensity is used as the input signal.

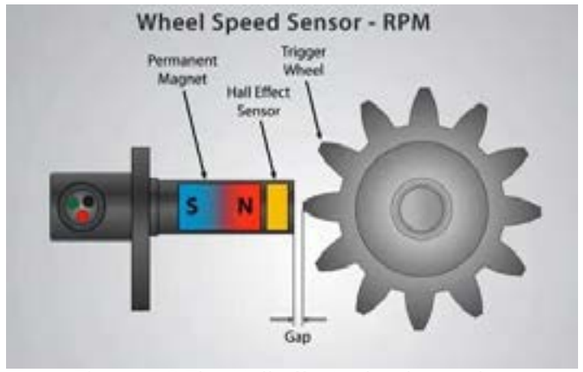


Figure 3.8 The Hall effect wheel speed sensor

The magnetic induction intensity changes with the wheel speed to generate a hall potential pulse. After amplifying, shaping, and amplifying the power amplifier in the hall IC, the pulse train is output to the outside, and its duty ratio changes with the angular velocity of the turntable. The rotation of the tooth disk alternately changes the reluctance, causing the change of magnetic induction intensity, and the hall potential pulse output by the sensor can be measured.

TIRE-PRESSURE SENSOR

A tire-pressure sensor is a small pressure sensor with a programmable electronic device integrated into one part, A tire-pressure sensor constantly measures the air pressure inside the tire. and the programmable electronic part transmits that information via low-frequency radio to the vehicle specifically to the unit responsible for collecting data from the sensors in the system, On the other hand, the sensor always comes in five pieces, Four represent the same sensor and The Fifth piece is responsible for receiving the results from the four sensors.
each sensor is powered by external batteries and there are two different ways to install them with the tire, either inside the tire located in the pressurized pocket made by a wheel and tire or outside the tire by attached to the valve stem.

Figure 3.9 pressure sensor

Any electronic pressure sensor relies on a physical reaction to applied pressure and then measuring the resulting proportional change electronically. Commonly used phenomena include changes in capacitance, or changes in ohmic resistance of a strain gauge or piezoelectric element, which are proportional to the magnitude of the deflection when pressure is applied.

In the first case, the sensor is inside the tire will work based on Piezoresistive idea, which consists of arranged in a similar bridge formation and the resistance changes according to the magnitude of the diaphragm deflection Thus, the value of the voltage applied to this resistance changes.

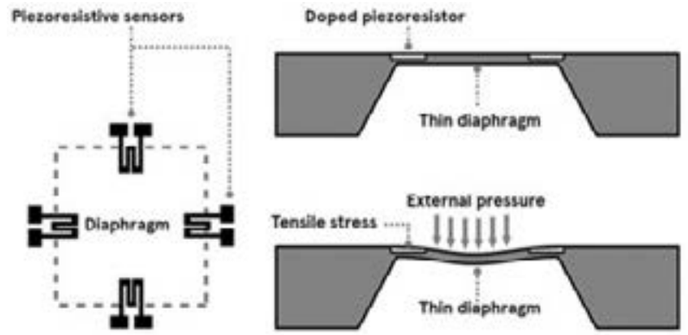


Figure 3.10 Piezoresistive pressure sensors

In the second case, the sensor is outside the tire will work based on the Capacitive idea, which contains a capacitor with one rigid plate and one flexible membrane as electrodes. The area of these electrodes being fixed, the capacitance is proportional to the distance between the electrodes. The pressure to be measured is applied to the flexible-membrane side, and the resulting deflection causes a change in capacitance that can be measured using an electrical circuit.

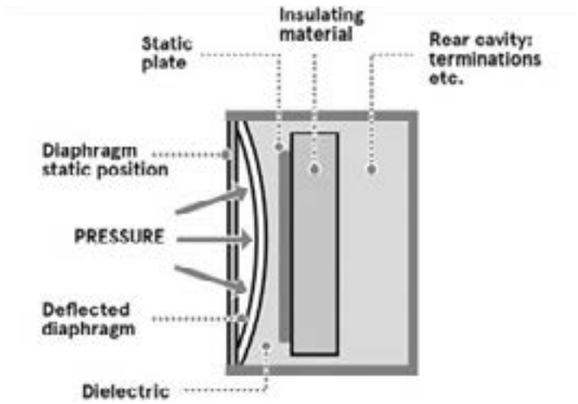


Figure 3.11 Capacitive pressure sensors

3.2 Data Collection and Simulation

In TEAS, we used a CARLA® Simulator to simulate the sensors that we talked earlier.
These sensors generate values that we are interested in such as the speed of the vehicle, the steer (wheel) angle, and the rotational angle of the vehicle which is represented in 3D vector with Pitch, Yaw, and Roll values. The following figure explain the difference between the Pitch, Yaw and Roll. Where the Pitch is the Y-axis rotation angle. Yaw is the Z-axis rotation angle. Roll is the X-axis rotation angle.

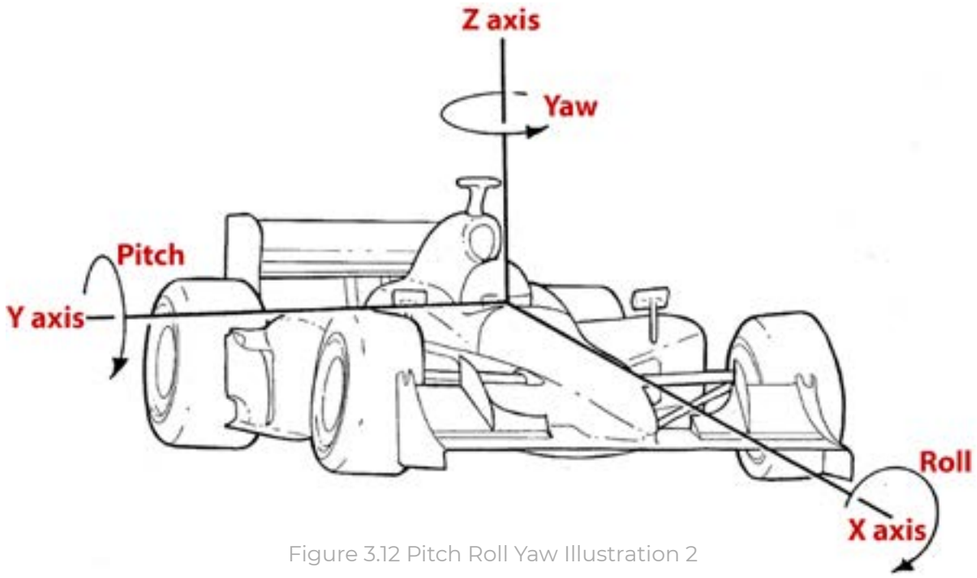


Figure 3.12 Pitch Roll Yaw Illustration 2

The following table illustrate the range of values for each sensor:

Sensor	From	To	Unit
Pressure Sensor	32	35	psi
Speed Sensor	0	220	Km/h
Rotational (Pitch)	-1.5	1.5	-
Rotational (Roll)	-3.0	3.0	-
Rotational (Yaw)	-180	180	deg
Steer (Wheel) angle sensor	-1.0	1.0	-

Table 3 Sensors range of values

Real time Simulation

Figure 3.13 Road map

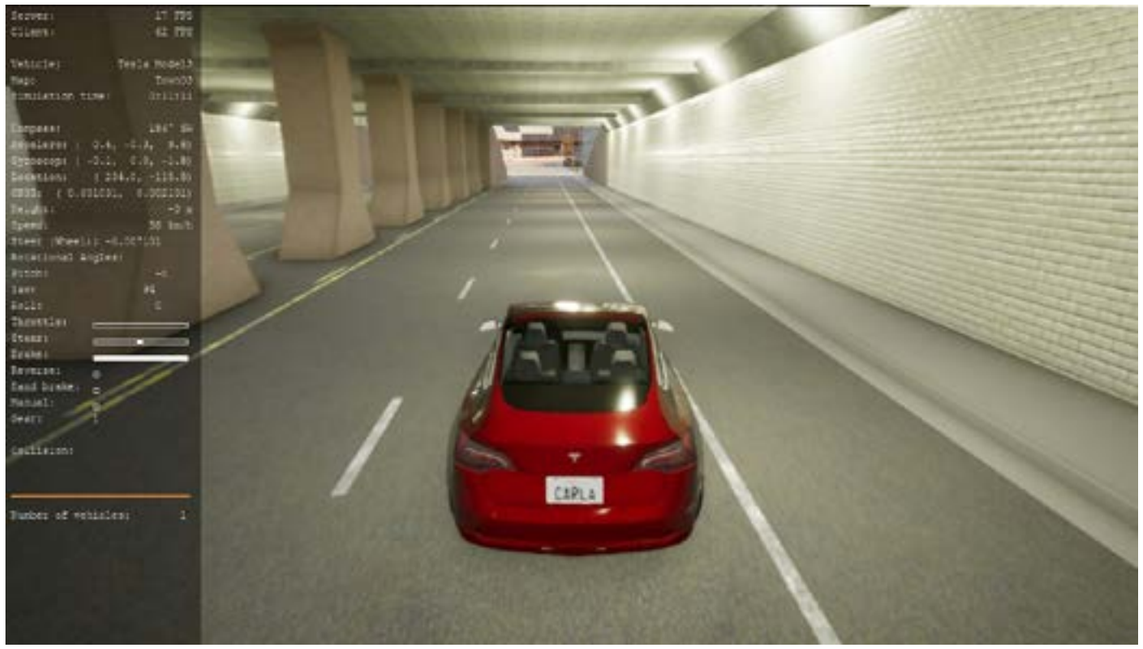


Figure 3.14 Normal Simulation

Figure 3.15 Normal Simulation 2

Figure 3.16 Crash Simulation

3.3 Data Display

The data that have been collected from the various sensors in the data monitoring system are sent through the (CAN) bus to the Visualization Unit (VU) to be displayed in front of user to ensure that all sensors' levels are al-right and the system is working well. If else then the system will send some instructions to the user to avoid accidents.

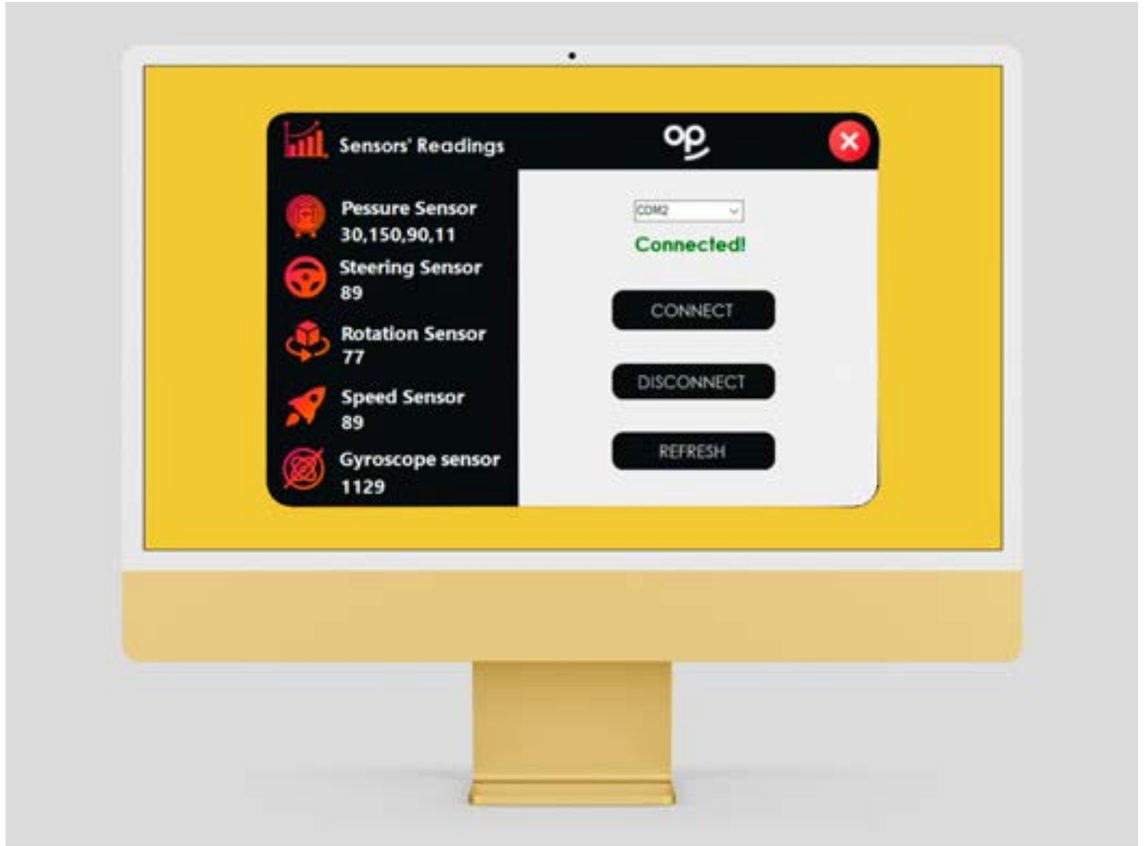


Figure 3.17 Data Display GUI

3.4 Voice Alerts

For The critical case, and in TEAS, the Visualization Unit (VU) is responsible for alerting the driver by voice so he can stop his vehicle safely. This situation is done only if one of sensors has respond with an out of range values, or with an out of control request.

From CARLA® Simulator, we recorded some of the test cases that may occur in real life. The following table shows each of these cases and in which category they belong.

Test Cases	Speed	Tire Pressure				Steering angle	Yaw Rate	Roll Rate
Optimum Value	0 : 220 Km/h	32 : 35 psi	32 : 35 psi	32 : 35 psi	32 : 35 psi	-1 : 1	-180° : 180° deg	-3 : 3
TC1	0	35	35	35	35	0	0	0
TC2	100	35	35	35	35	0	0	0
TC3	120	35	35	35	35	7.9	80°	0.9
TC4	70	35	35	35	35	-1	80°	0.9
TC5	30	29	35	35	35	1	70°	0.9
TC6	60	35	20	35	35	1	90°	0.9
TC7	90	35	35	30	35	-1	112°	0.9
TC8	170	35	35	35	31	1	120°	0.9
TC9	110	37	35	35	35	1	110°	0.9
TC10	126	35	38	35	35	-1	65°	0.9
TC11	190	35	35	50	35	-1	30°	0.9
TC12	76	35	35	35	55	1	-73°	0.9
TC13	34	60	35	35	35	1	-85°	0.9

Table 4 Test Cases

Test Cases	Test Case Status A	Test Case Status B	Action
Optimum Value	-	-	
TC1	Tires In Good Cond	Parked	No Action
TC2	Tires In Good Cond	Well Driving	No Action
TC3	Tires In Good Cond	Sensors Problem	Voice assistant
TC4	Tires In Good Cond	Well Driving	No Action
TC5	Tire 1 Underflow	Well Driving	ASK for Check
TC6	Tire 2 Underflow	Well Driving	ASK for Check
TC7	Tire 3 Underflow	Well Driving	ASK for Check
TC8	Tire 4 Underflow	Well Driving	ASK for Check
TC9	Tire 1 Overflow	Well Driving	ASK for Check
TC10	Tire 2 Overflow	Well Driving	ASK for Check
TC11	Tire 3 Explosion	Out Of Control	Voice assistant
TC12	Tire 4 Explosion	Out Of Control	Voice assistant
TC13	Tire 1 Explosion	Out Of Control	Voice assistant

Chapter 4
V2C

4.1 Introduction to V2C

As one of the exciting things about the system that we offer is not only getting to monitor and producing good and dangerous status to the driver, but it is also an attempt to create a technical community between cars and some of them capable of understanding and knowing the dangerous situation individually for each other. At another stage, the car also communicates with a cloud capable of providing rescue service for it in event of a problem. Therefore, this section starts the discussion on communication between cars and an attempt to use the best method of communication and its application.

Before we begin, let us clarify several concepts, V2V is an acronym Vehicle-to-vehicle Which can be considered part of Vehicular communication systems, this system aims to a large network make the Vehicle communicate with many nodes on roads also known as V2X, these nodes can be a vehicle, traffic lights, roads stations, and the phone's users on the road, etc. So, we can simply create a network that can connect Vehicles to each other, at that moment, we can create a network capable of serving the rest of the nodes later, this makes the V2V is important in research now from the V2X. V2V features can be providing information for Vehicles to others, such as safety warnings and traffic information. They can be effective in avoiding accidents and traffic congestion.
But for the system we are building, we try to implement the idea of vehicle cloud communication (V2C). Once to the vehicle to cloud communication, using cellular networks or WIFI the vehicle can interact with other vehicles get help from the head centers registered on the cloud as we will discuss later.

4.2 communication techniques

The process of connecting the car to the cloud can be done directly using cellular or WIFI connections, and the V2V can be done by using DSRC direct without an intermediary. But the important question remains, what made us use WIFI connections in implementing the idea?
compared between Wireless communication technologies such as the wireless LANs (WLAN IEEE 802.11a/b/g/n/p standards), and the third- and fourth-generation cellular wireless (3G or 4G) We will find a connection like cellular wireless has a significant infrastructure and has been deployed to support the dramatically increasing demand for mobile terminals to access network services anywhere and anytime.

so, it's not efficient enough to satisfy the demands of various users for reliable connection and quality of service (QoS) in all situations and problems like the Limited band and Low latency with a large number of cars available, which will increase in the future and other problems, Therefore, the implementation will be by Wi-Fi instead of cellular connections.

Dedicated short-range communications (DSRC)

Dedicated short-range communications (DSRC) wireless communication technology specifically designed for automotive use, work in one-way or two-way, and operates in the 5.9 GHz band providing direct and low latency information, it's an 802.11p-based that enables highly secure, high-speed direct communication between vehicles without involving any cellular infrastructure. The word "Dedicated" in DSRC refers to FCC has allocated 75 MHz of licensed spectrum in the 5.9 GHz band for DSRC communication and this band is divided into several channels and It is expected that V2V safety messages exchanged on Channel 172, a specific channel designated for safety. The term "Short-Range" is referring to the communication that takes place over hundreds of meters this is a shorter distance than cellular.

Advantage of DSRC

• It does not require a cellular signal or a Wi-Fi connection, but rather it communicates directly with other vehicles.
• DSRC makes vehicles on the same frequency are nearby which makes communication without lag and the signal is stronger but in cellular communication, vehicles must be sent through the network which makes the latency be higher, even if vehicles standing side by side.
• Low latency where the connection opens and closes in around 0.02 seconds.
• DSRC can provide high-speed communication even in the presence of obstructions and can even handle fast-changing environments at speeds as high as 500 km/h.
• DSRC provides a highly secure communication channel where the receiving vehicle validates the authenticity of the received messages and the messages aren't linkable to the vehicle, thus protecting the driver's privacy.
• Every vehicle broadcasts its location, heading, and speed 10 times per second in a secure and anonymous manner. All surrounding vehicles receive the message, and each estimates the risk imposed by the transmitting vehicle.

This allows a perception, detection, and assessment of dangerous situations (road obstacles and potential collisions with road users) even before they can be noticed visually.
• Despite the band is protected by FCC regulations, but short-range limitations make interference from surroundings is unlikely that make DSRC characterized by Limited interference.
• DSRC is reliable in all conditions because it is resilient to weather.

Disadvantage of DSRC

• There is no fixed frequency band to operate between countries, for example The US allocated 75 MHz of spectrum in the 5.9 GHz band, but Europe allocated 30 MHz of spectrum in the 5.9 GHz band for ITS. and DSRC systems in Europe and Japan allocated 5.8 GHz band, also in Egypt, 5.9 GHz band is allocated for another communication technique.
• until this moment, there is no hardware that developers use to work on DSRC, so we apply the idea using WIFI.

We used to implement v2c communication

• GPS module to get the location coordinates (latitude and longitude)
• Azure cloud services (infrastructure and MySQL database)

Database tables

• Car: contains description of the car, its location, its state, and the time of last update
• Accident: contains car id, the location of the accident and its time
• Car readings: contains sensors readings of the car
• User: related to the authorities

Steps of communication

1. Vehicle periodically sends the GPS coordinates and the state of the tires of the car to the cloud.
2. Then cloud updates the state, location, and the time of this of the vehicle record in the database.
3. Then the cloud responds to the vehicle if there is a near car that has problems to avoid it or if there is a near accident and sent.
4. If there is a near accident in a range of 1500 m the cloud will response with the distance between the vehicle and the accident.

&368.65,2605, Via Corona, Montebello, California, 90640, United States;

5. If there is a near vehicle in a range of 500 m the cloud will respond with the distance between the two-vehicle and the description of the vehicle to avoid it such as type, color, and plate number.

@0.00,nissan sunny,red,abc123,Repetto Street, East Los Angeles, Montebello, California, 90022-1632, United States;

States of the vehicle that are sent to the cloud

•Moving without a problem.
• Moving with tires problems.
• Tier exploded.

Handler tool

The functions of this tool can be summarized in two main points:
1. Ensure that the messages between the car and the cloud are permanently updated, and if a minute passes without an update, the tool will change the status of the message to be “expired.
2. In the event of an accident, the message will be critical, so the handler tool will put it in the accident table and allows the authorities to access this table for rescue missions and allows another car to access the accident table to avoid the roads that have a problem.
This tool has no user interface; it runs in the background , and do its tasks automatically.

Searching technique used in database

Searching in database using the distance difference equation using latitude and longitude is inefficient because it will take a much time to apply the equation to every single record in the table.

The effective way is to search using a range of latitude and longitude which is calculated by:
1. Making latitude constant and calculate the longitude difference which added or subtracted from the longitude to achieve the distance Require.
2. Making longitude constant and calculate the latitude difference which added or subtracted from the latitude to achieve the distance Require.
3. The range of latitude is from latitude-latitude difference to latitude + latitude difference
4. The range of longitude is from longitude – longitude difference to longitude + longitude difference

4.3 C2(Authorties)

In case of accidents, fast medical care is required, not only the medical care but also local authorities should be informed in a fast way. Our project guarantees to inform authorities at the moment of the accident.

After an accident happens, the car sends a request to the global server, this server will add this accident to the database including car id number, location, plate number, city and country.

But how local authorities will be informed?

We created a GUI Application, This application is connected to our database and also all local authorities car databases. Each state in a country will have an account also the country has a global account to monitor the full country accidents.
When a user logs in to our application, all the accidents will appear in tables with its current statutes (Active, InActive) and the admin can switch between the two statutes after the local authorities Rapid intervention.

Figure 4.1 GUI Monitoring Accidents

The user can also move between more than a panel to view only Active or InActive Accidents, also we have a fast search mechanism to search in our tables to make sure fast informing and intervention.

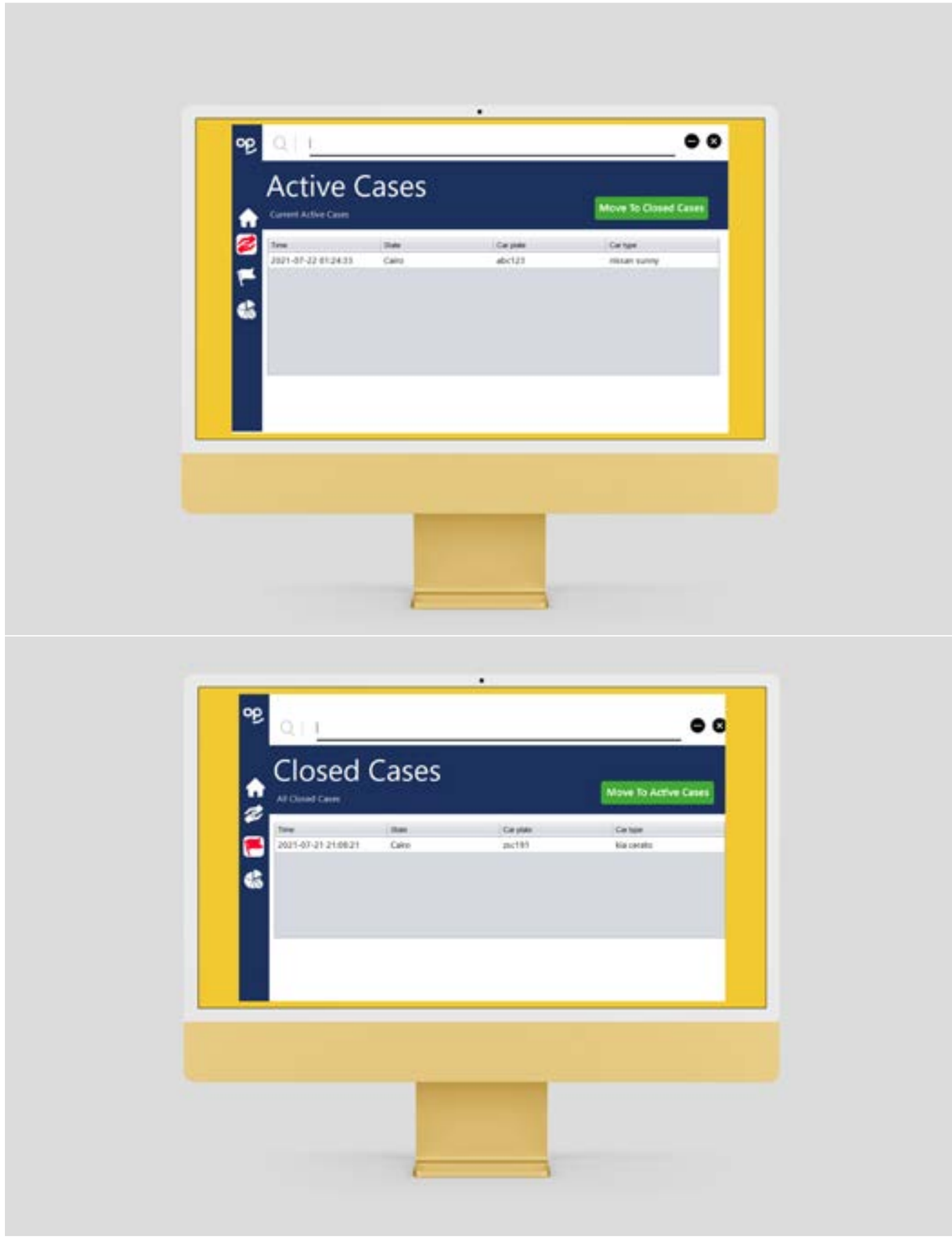


Figure 4.2 GUI Active and InActive Accidents

4.4 V2(OEM)

In case of accidents, Self Driving Cars must have an algorithm to stop the car before the situation go more wrong than current situation. Our project aims in case of accident to send all sensors readings to OEM, but why?
Those sensors readings change due to the car situation on the road, this situation changes based on Algorithms, Our target to get all readings and send them to OEM to make data analysis and use those readings to improve the current Algorithm

But How we send those data?
During the accidents all the data sent on the CAN bus will be saved on SD Card and when the car stops, the main ECU will send those readings to the OEM Cloud.

Now imagine that the whole world can provide data to enhance machine learning algorithms. that is our target to enhance the security of people in a fast and efficient way.

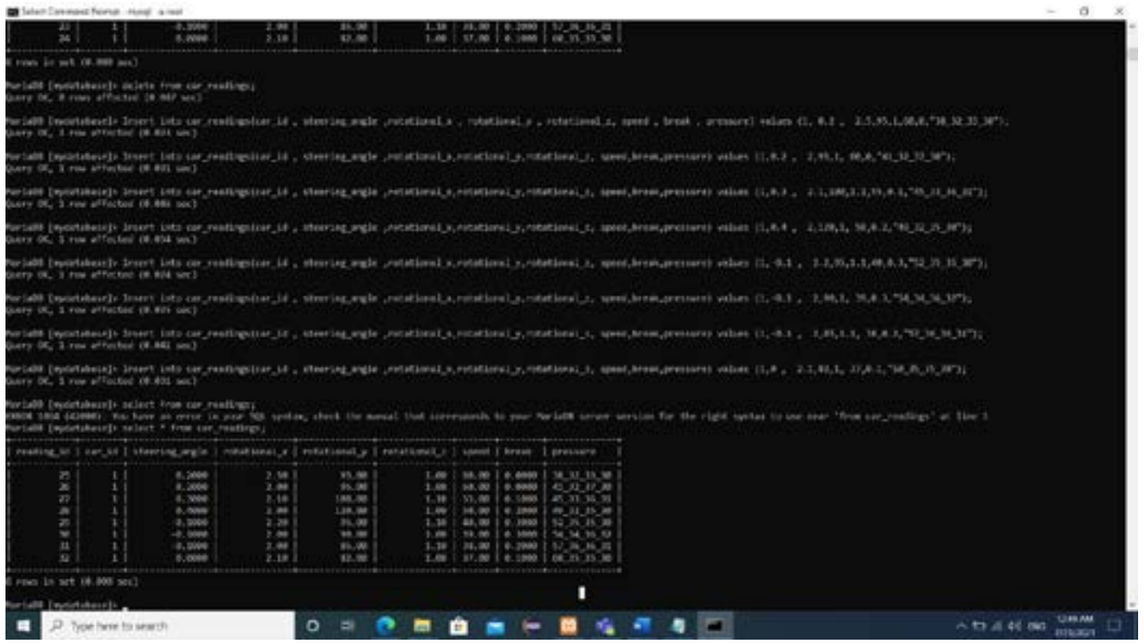


Figure 4.3 OEM Sensors reading Database

Chapter 5

FOTA

FOTA(Firmware over the air)

5.1 Flashing

Figure 5.1 Flashing STM32

The flashing process is the process of updating the content of the flash memory, and since the flash is used to store the program code, the process is also called Programming, Burning.
Recall that flash is a non-volatile memory which consists of floating gate mosfets, this kind of mosfets needs high power to change its state, this high power can't be provided by the processor, so that it is a must to use an extra hardware circuit that can provide this high power to access the flash memory, this circuit is called the flash driver.
So now the process of programming is obvious, we prepare our executable file then send it to the flash driver, the flash driver in turn will write it to the flash.
Depending on the position of the flash interface, different programming techniques are found:

1- OFF Circuit Programming

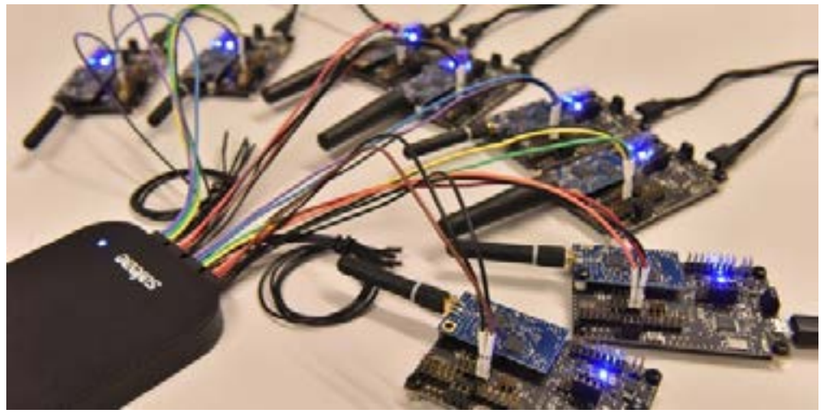


Figure 5.2 OFF Circuit Programming

This is the basic old technique, here the flash driver is external outside the microcontroller (MC) and the flash memory programming pins are connected to some (MC) pins to be programmed through. A device called burner is used, this burner is usually a hardware (HW) kit that sometimes has a socket on which the (MC) can be placed, or simply connected to the required pins with jumpers. This burner kit includes the flash driver, it also includes the communication port with the computer i.e., USB port, and uses the computer USB power.

So, the task of the burner is to get the executable file from the computer via its communication port, then it applies the required high power on the flash memory as if it uploads the file to flash memory. When the burner finishes uploading the file, the microcontroller can now be removed and connected to the application circuit.
So, to summarize this technique, to program the microcontroller, the microcontroller shall be removed from its application circuit, then connected to the burner which in turn uploads the program to the flash memory, and when it finishes, the microcontroller can now go back to its application circuit and start working, hence called off circuit programming as it requires removing the microcontroller from its application circuit. But what if the process of removing the microcontroller from its application circuit for re-programming is not that easy?! For example, in firmware update applications, where the new program shall be uploaded to a lot of microcontrollers inside their machines, this will be a big time and effort consuming task, so that a better technique has arisen. This technique is called In-Circuit Programming.

2- In Circuit Programming

It's Also called In System Programming (ISP), here the microcontroller has an internal flash driver, that can generate any necessary programming power from the system's main power supply to provide the required high power for the flash programming which is inside the (MC) also, so the flashing process is done inside the (MC) which means there is no need to remove the (MC) from the system circuit for programming, hence its name.

Figure 5.3 IN Circuit Programming

Of course this feature has provided a lot of advantages as now the (MC) can be programmed while installed in a complete system, rather than requiring the chip to be programmed prior to installing it into the system, hence it allows firmware updates to be delivered to the on-chip flash memory of microcontrollers without requiring specialist programming circuitry on the circuit board, which simplifies design work, also it allows manufacturers of electronic devices to integrate programming and testing into a single production phase, and save money.

But how to communicate with this on-chip flash driver I.e., how to send it the required executable file? Firstly, the flash driver is needed to communicate with the external world to get the required application code, this is done by serial communication, so the flash driver manufacturer defines one or more communication protocols through which the flash driver can interface, for our stm32f103 there are 2 protocols that are provided to communicate with the in-circuit flash programmer: JTAG and SWD (Serial Wire Debug).

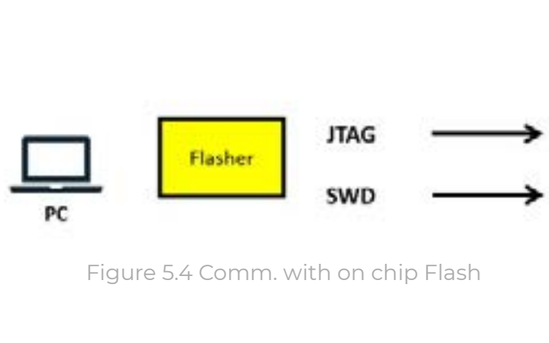


Figure 5.4 Comm. with on chip Flash

Figure 5.5 ST-LINK V2 programmer and debugger

In our system the executable file will be sent to the in-circuit flash driver through one of these protocols then the flash driver will apply the required power to program the flash memory, but how to send the executable file from the computer using the USB protocol to the flash driver by the SWD? Again, the answer is that there must be a converter that translates the USB protocol to the SWD protocol, and this will be the only function of the external device that will be used during the programming process. This is the technique we use; we used the ST-Link programmer and debugger as shown in the last figure.
One end connects into the computer. This allows us to transfer the executable file from the computer to the USBASP. The other end of the ST-LINK normally gets connected to 4 wires, which can then get hooked up easily to the programming pins on the KIT. Note that until this moment there is an extra device is used to implement the flashing process, as we needed a translator to send the executable file from the computer to the in-circuit flash driver, and this device is target-specific which means that changing the microcontroller to another one with different flash driver communication protocol will result in changing the translator device, this will be a big headache if it is required to reprogram a number of different microcontrollers in the same system, i.e., a car with 100 ECUS (electronic control units), this is one of the main reasons why the bootloader is used.

2- In App Programming(IAP)

(IAP) allows the user to re-program the flash memory while the application is running. Nevertheless, part of the application has to be been programmed in the flash memory using (ICP) previously.

Steps:

• receive the file.

• Flash it (page by page).

There is two Applications APP1 and File.elf, We need a flash driver then the APP1 must be flashed ICP firstly, now we can use IAP.

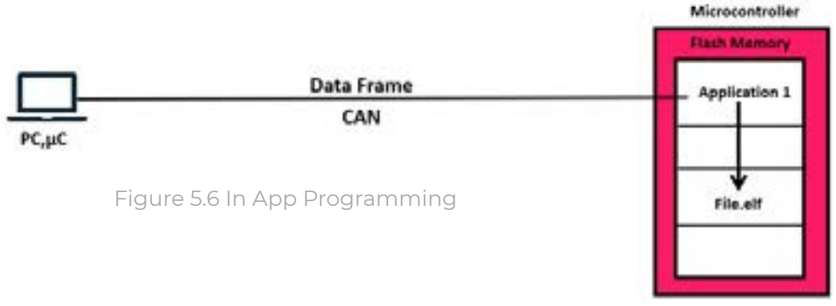


Figure 5.6 In App Programming

Note:

• Application that can be written on the flash during Run time is called Bootloader.

5.2 Bootloader

Booting (also known as booting up) is the initial set of operations that a computer system performs when electrical power is switched on. Bootloader Anyone who has turned on a computer might be familiar with the boot-up sequence as the computer flashes lines of text on screen before the Windows logo appears, what you are seeing is a bootloader in action, loading essential software to get the minimum running on the processor chip before higher-level software can run. As its name implies it can both “boot” the operating system or whatever application is to be run, and also as a second function provide a “loader” capability for developers.

Bootloader is a small OS, or application, designed to download firmware in MCU’s internal or external Flash Memory).

Why do we need a Bootloader?

• To fix Bugs.

• Updating system with new features.

Where are they stored?

They reside In ROM, or in the Flash Memory of MCU. (write protected). So, you can’t destroy them.

Bootloader Requirements:

• Needs Flash Driver (FPEC). “Responsible for program/erase flash”.

• Selects a communication Protocol (CAN). “to download programming data into memory”.

• Good design.

In embedded systems, This task can be simplified as the following:

1. The processor initializes a communication port in the microcontroller to make it ready to receive data.

2. A computer starts sending the application code through this communication port to our microcontroller.

3. the processor reads the data, then sends it to the in-circuit flash driver which in turn writes this data in the flash, eventually the target controller is updated with the code. (Flash the file page by page).

These procedures may have some problems. Let’s see how to solve them. The first problem is obvious when we take a closer look inside the flash, the bootloader is a piece of code so it resides in the flash and the processor keeps reading it during its execution phase, at the same time the flash interface writes the application program sent by the processor into the flash, but how could it be done? To make the problem clear we can say that 2 devices are accessing the flash at the same time, one device is reading (the processor), and the other device is writing (the flash driver), which can’t be done if it was the same flash. So, one of the simplest solutions which is implemented by ST company (our microcontroller’s manufacturer) is making more than one memory, each one is separated, one to store the bootloader code which is the system memory and one for the application code which is the flash memory so that you can write into one while reading from the other.

The second problem is that the computer sends the executable file via its USB but the processor can receive it via a different communication port, in our system via the (CAN), this is handled by the bootloader code and the processor.

after the processor reads data from (CAN) it then sends it to the flash driver peripheral into its registers, so in this case the processor is working as if it is a translator during the program burning, this approach can be said in other words that the microcontroller programs its own memory so it is called self-programming flash, this design method can be used for (IAP) requirements where the firmware can be updated during the running of the application, with no need to stop or reset before reprogramming.

Bootloaders are responsible for:

• Flash new application.
• Run Application.
• Flash new Bootloader.
• Select Application (if we have many applications to select).

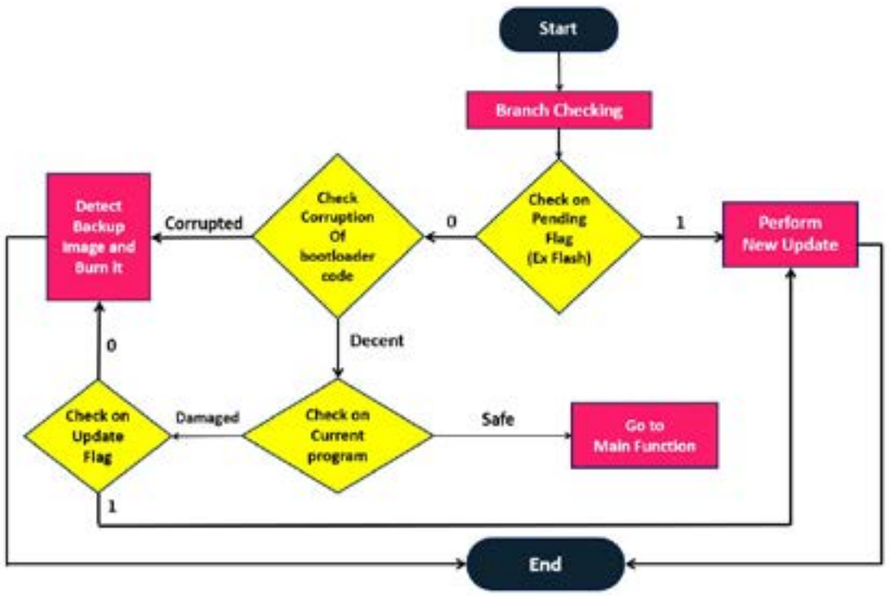


Figure 5.7 Bootloader Design

Bootloader Design

Branch checking: It's also known as Initialization phase and can do that on two Stages:

• **1st Stage**

- Initialize hardware components.
- Prepare memory space for loading 2nd Stage program.
- Copy 2nd Stage program into memory space.
- Set up stack pointer (SP).
- Jump to entry point of 2nd Stage program.

• **2nd Stage**

- Initialize hardware components used in this stage:
o Mask all interrupts.
o Set CPU speed and clock rate (RCC).
o Initialize RAM.
o Initialize GPIO.
o Disable CPU internal Instruction/Data Cache.
- Check memory map.
- Copy kernel and root file system images into memory.
- Set parameters.
- Boot kernel.

After initialization phase, The sequence is to check on a pending flag to indicate whether there's a new update to be downloaded or not. This pending flag resides on the Flash Memory. If it's value equal to one then there's a new update, then go perform this update by flashing it. if not there's no update and then you must check the validity of the bootloader code.
If corrupted, then you must check if there's a backup image of the application in the flash or not, if there's a one then burn it. If the bootloader code is decent then you will check on the current program, if safe then go to main function and run it if it's damaged then you must check on update flag to check if there's a new update to burn it, if not then detect a backup image and burn it.
All the files mentioned in the previous sequence is in HEX format.

Hex Parser

HEX file that the processor receives from (CAN) is stored as ASCII characters, which can be read by an editor. Hex is commonly used for programming microcontrollers, EPROMS, and other types of programmable logic devices. In a typical application, a compiler or assembler converts a program's source code (such machine code and outputs it into a HEX file, Common file extensions used for the resulting files are (.HEX). The HEX file is then read by a programmer to be transferred to the target system for loading in C language) to and execution.

HEX file format

HEX file consists of lines of ASCII text that are separated by line feed or carriage return characters or both. Each text line contains hexadecimal characters that encode multiple binary numbers. The binary numbers may represent data, memory addresses, or other values, depending on their position in the line and the type and length of the line. Each text line is called a record. Each text line is called a record.

A record (line of text) consists of six fields (parts) that appear in order from left to right:

1. Start code: one character, an ASCII colon :
2. Byte count: two hex digits (one hex digit pair), indicating the number of bytes (hex digit pairs) in the data field. The maximum byte count is 255 (0XFF). 16 (0x10) and 32 (0x20) are commonly used byte counts.
3. Address: four hex digits, representing the 16-bit beginning memory address offset of the data. The physical address of the data is computed by adding this offset to a previously established base address, thus allowing memory addressing beyond the 64-kilobyte limit of 16-bit addresses. The base address, which defaults to zero, can be changed by various types of records. Base addresses and address offsets are always expressed as big-endian values.
4. Record type: two hex digits, 00 to 05, defining the meaning of the data field.
5. Data: a sequence of n bytes of data, represented by 2n hex digits. Some records omit this field (n equals zero). The meaning and interpretation of data bytes depends on the application.
6. Checksum: two hex digits, a computed value that can be used to verify the record has no errors.

Figure 5.8 Hex File Example

Record types

HEX files have six standard record types, Here we will focus on 3 of them:

1. Data record

• Hex code: 00

• Example: : OB0010 00 6164647265737320676170A7

• Description: Contains data and a 16-bit starting address for the data. The byte count specifies the number of data bytes in the record. The example shown next has 0B (eleven) data bytes (61, 64, 64, 72, 65, 73, 73, 20, 67, 61, 70) located at consecutive addresses beginning at address 0010.

2. End Of File record

• Hex code: 01

• Example: 00000001FF

• Description: Must occur exactly once per file in the last line of the file. The data field is empty (thus byte count is 00) and the address field is typically 0000.

3. Extend Segment Address record

• Hex code: 02

• Example: 020000021200EA

• Description: The data field contains a 16-bit segment base address (thus byte count is always 02) compatible with 80x86 real mode addressing. The address field (typically 0000) is ignored. The segment address from the most recent 02 record is multiplied by 16 and added to each subsequent data record address to form the physical starting address for the data. This allows addressing up to one megabyte of address space.

Figure 5.9 Upload Update To Server

5.3 FOTA Limitations

Due to sending & receiving updates over the air so there's some limitations that we must take care of. The most 2 important obstacles are the validation of the data and the security of transmitting it because attackers will try to attack on data integrity to ensure that the sending data isn't the same as the delivered one by modifying it, They also attack its security to get the access to see the content of the transmitting messages and take another dangerous steps.

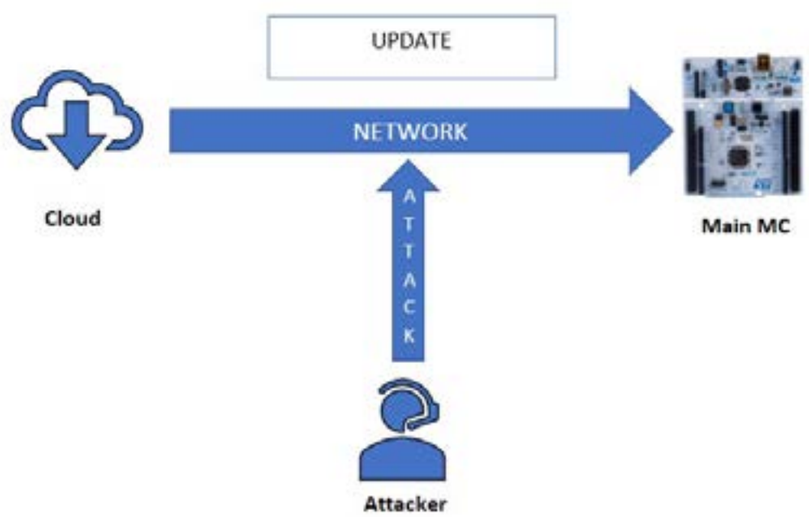


Figure 5.10 FOTA Limitations Block Diagram

Validation

(TEAS) uses HASH functions to ensure the data integrity. Hash function can be used to map data of a variable size into a fixed size data. The data that'll be returned from hash function is called digests, hash codes or simply hashes.

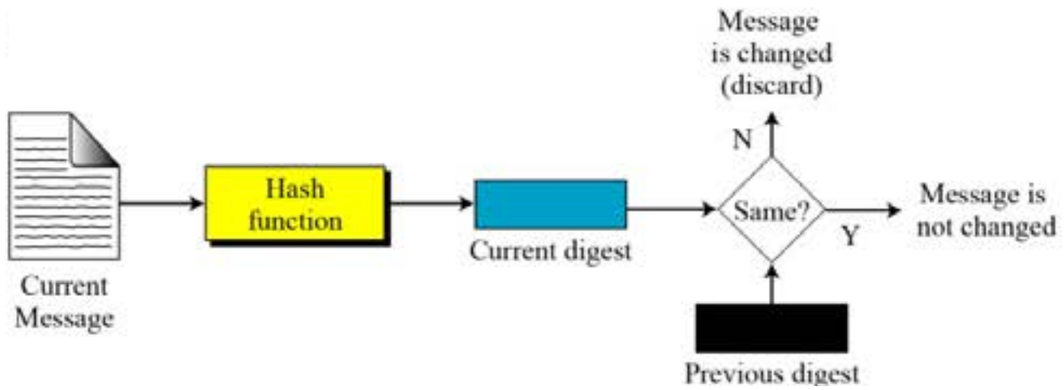


Figure 5.11 Data Integrity using Hash Function

SHA-2

Figure 5.12 Hashing Algorithm Overview

The SHA-2 family consists of six hash functions with digests (hash values) that are 224, 256, 384 or 512 bits: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256. SHA-256 and SHA-512 are novel hash functions computed with eight 32-bit and 64-bit words, respectively. They use different shift amounts and additive constants, but their structures are otherwise virtually identical, differing only in the number of rounds. We have chosen to use SHA-256 because it's:

1- Secure and trusted industry standard

SHA-2 (Secure Hash Algorithm 2) is a set of cryptographic hash functions designed by the United States National Security Agency (NSA) and first published in 2001.

2- Collisions are incredibly unlikely

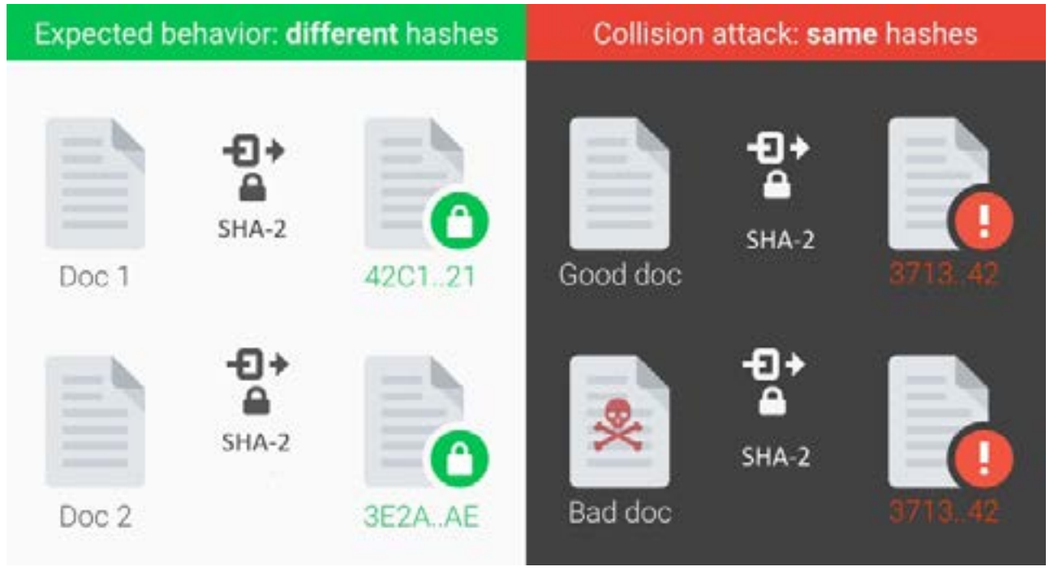


Figure 5.13 Sha-2 Collision Behavior

A collision occurs when two distinct pieces of data—a document, a binary, or a website's certificate—hash to the same digest as shown above. In practice, collisions should never occur for secure hash functions. However, if the hash algorithm has some flaws, as SHA-1 does, a well-funded attacker can craft a collision. The attacker could then use this collision to deceive systems that rely on hashes into accepting a malicious file in place of its benign counterpart. For example, two insurance contracts with drastically different terms. With older hash technologies, like MD5 which generates 128-bit hashes, this was almost a reasonable concern. With the modern SHA-2 family of hashes, data corruption due to hash collision is, for all intents and purposes, impossible.

For SHA-512, a 512-bit hash, that number of documents is just about 2^256. And 2^256 is truly astronomical number. How big is 2^256? It's about 116 trillion trillion trillion trillion trillion trillion. A bit larger than 1 followed by 77 zeros. To put a scale on it, it's significantly more than the number of atoms in our galaxy (around 2^226). Statically, that means a collision is incredibly unlikely. More unlikely that you and 12 of your colleagues being independently struck by lightning. More unlikely than your house being struck by a meteor... five times. To reach 2^256, you would have to write a document every millisecond for 2^221 years. That's 4 million trillion trillion trillion trillion trillion trillion years. More unlikely than just about anything you can imagine. Should we panic? Of course not. The problem with focusing on hash collisions as this horrible possibility for data corruption ignores all of the other, far more likely, ways in which data could be corrupted. It's like buying a carbon fiber bicycle because you're worried about getting hit by lightning, and then riding without a helmet through downtown Boston traffic. Humans are really bad at gauging risk, and so sometimes we get worried about the wrong things, or things we shouldn't be worried about at all.

3- Able to accept input of any length and output a fixed-length result (256 bit).

Figure 5.14 outputs result with fixed length (256 bit)

4- One way encryption

It means it's irreversible. If you have the hash value you can't return to the original message sent. The next Figure shows how the message authentication works overall. The one-way function, Hash-Based Message Authentication Code with Secure Hashing Algorithm 2 (SHA-2), is run over the header and payload with a secret key. The sender writes the SHA-2 hash into the authentication tag, and the receiver runs the same computation and checks its result against the tag. If the two do not match, the message authentication is said to fail and the packet is discarded.

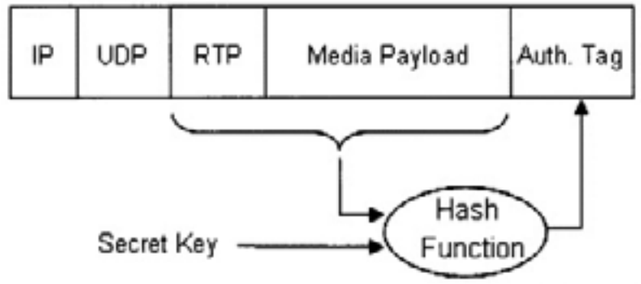


Figure 5.15 message authentication Procedure

5- The avalanche effect

- The same input will always produce the same hash.
- Slightly different inputs will produce different hashes.
- Trial-and-error is the only known way to find an input that gives a certain hash.

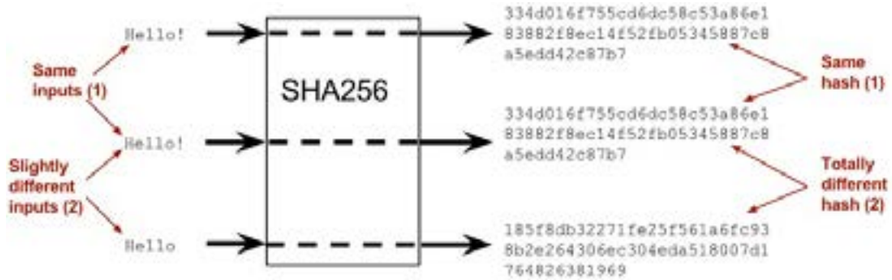


Figure 5.16 The Avalanche Effect

TEAS SHA-256 Deliverables

Figure 5.17 SHA-256 App Output

We tried the input “abc” text and get this output from our program Console. We guarantee that our output is correct by using an online HASH tool.

Security

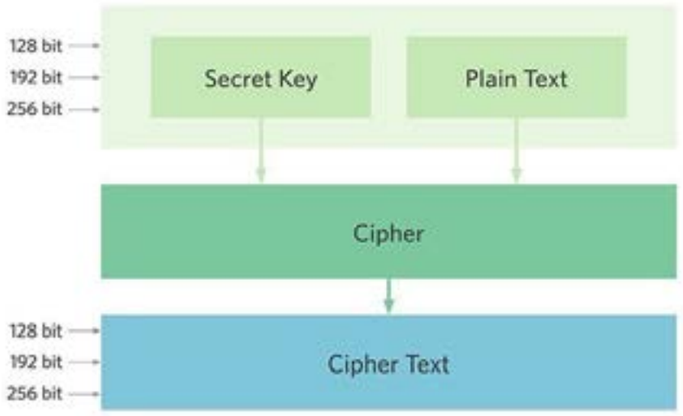


Figure 5.18 AES Design

(TEAS) uses Advanced Encryption Standard (AES) which is a FIPS-approved cryptographic algorithm that used to protect data. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. Encryption converts data to an unintelligible form called ciphertext; decrypting the ciphertext converts the data back into its original form, called plaintext. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits.

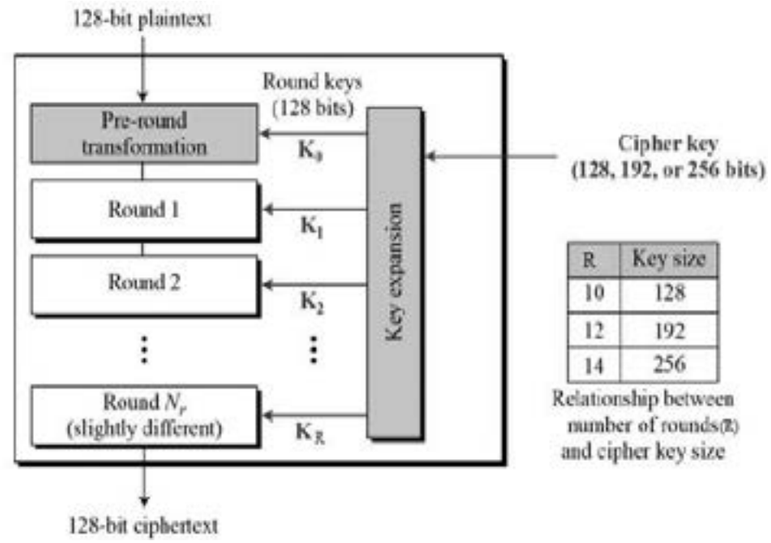
AES is The most popular and widely adopted symmetric encryption algorithm likely to be encountered nowadays. It is found at least six time faster than triple DES. A replacement for DES was needed as its key size was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attack. Triple DES was designed to overcome this drawback but it was found slow.

AES features

• Symmetric key symmetric block cipher.
• 128-bit data, 128/192/256-bit keys.
• Stronger and faster than Triple-DES.
• Provide full specification and design details.
• Software implementable in C.

Operation of AES

AES is an iterative rather than Feistel cipher. It is based on ‘substitution–permutation network’. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations). Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix –
Unlike DES, the number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key. The schematic of AES structure is given in the following figure



R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

Figure 5.19 AES structure

Encryption Process

Here, we restrict to description of a typical round of AES encryption. Each round comprise of four sub-processes. The first-round process is depicted below

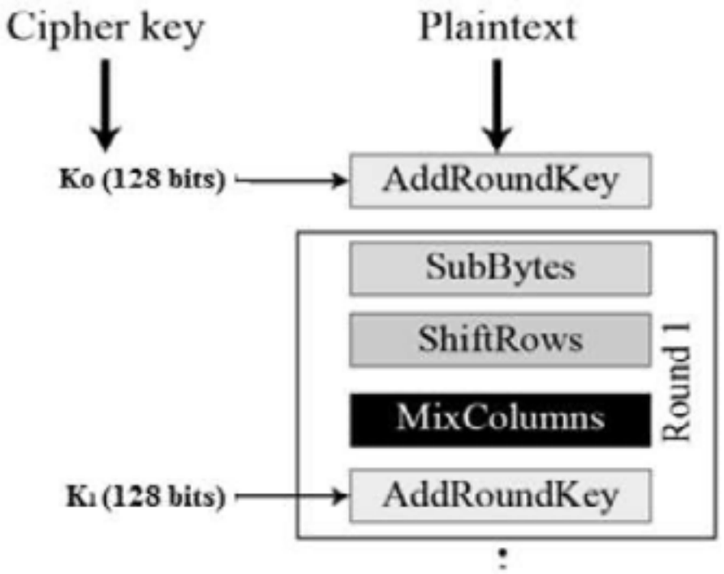


Figure 5.20 First Round Process

Byte Substitution (Sub-Bytes)

The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.

Shift-Rows

Each of the four rows of the matrix is shifted to the left. Any entries that 'fall off' are re-inserted on the right side of row. Shift is carried out as follows –

- First row is not shifted.
- Second row is shifted one (byte) position to the left.
- Third row is shifted two positions to the left.
- Fourth row is shifted three positions to the left.
- The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.

Mix-Columns

Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.

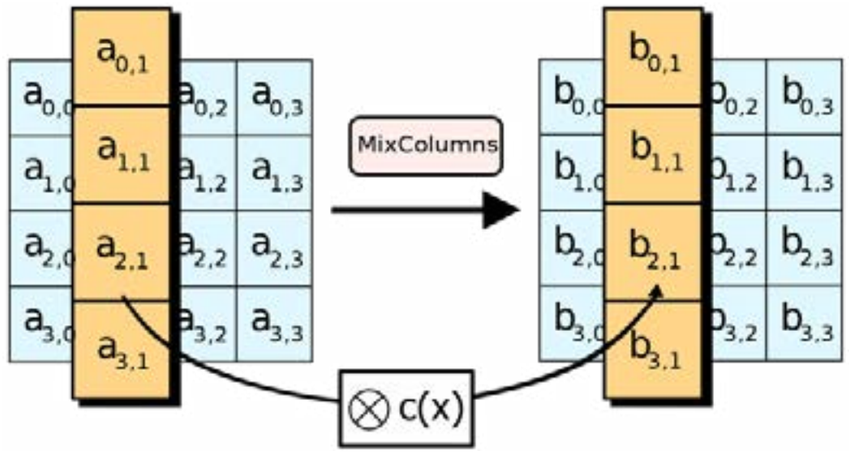


Figure 5.21 AES Mix-Columns

Add Round-key

The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

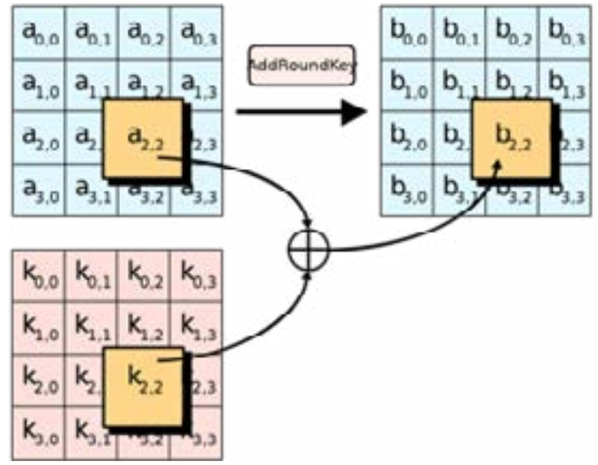


Figure 5.22 AES Add Round Key

Decryption Process

The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse order

- Add round key
- Mix columns
- Shift rows
- Byte substitution

Since sub-processes in each round are in reverse manner, unlike for a Feistel Cipher, the encryption and decryption algorithms needs to be separately implemented, although they are very closely related.

AES Analysis

In present day cryptography, AES is widely adopted and supported in both hardware and software. Till date, no practical cryptanalytic attacks against AES has been discovered. Additionally, AES has built-in flexibility of key length, which allows a degree of 'future-proofing' against progress in the ability to perform exhaustive key searches. However, just as for DES, the AES security is assured only if it is correctly implemented and good key management is employed.

TEAS AES-128 Deliverables

Figure 5.23 Receiving hex data from the server to the µ-controller.

Figure 5.24 TEAS Encryption & Decryption Outputs

To guarantee that (TEAS) outputs are alright we checked the same input data on an AES Online Tool and here's the result.

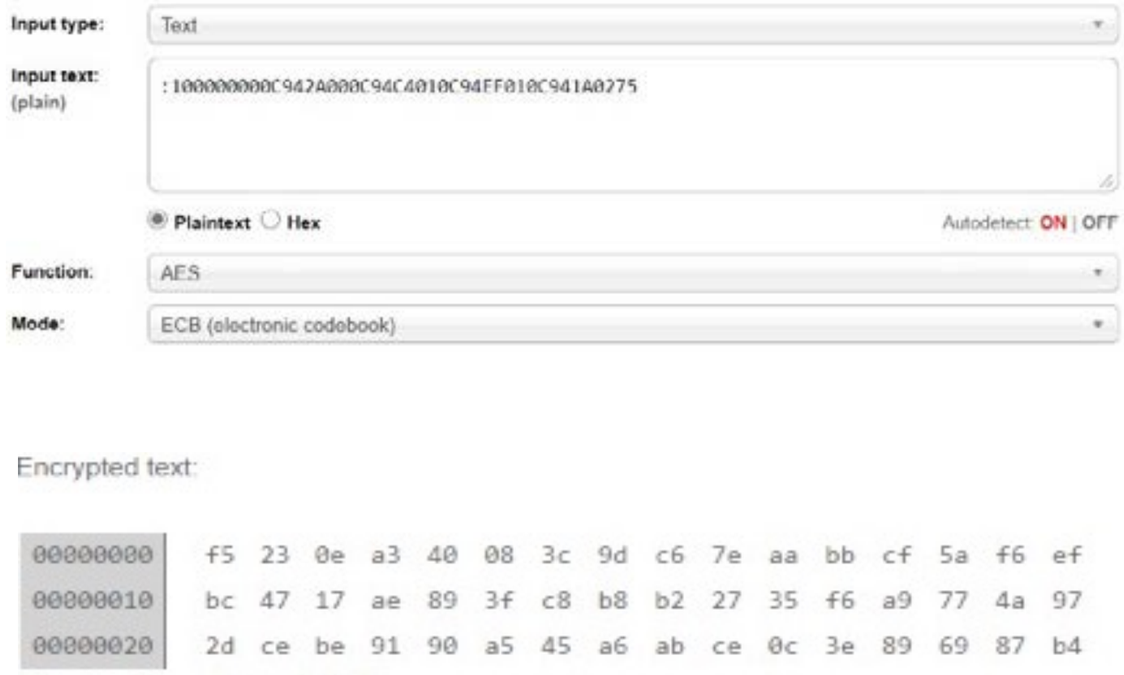


Figure 5.25 The AES Encryption & Decryption Online Tool Outputs.

Chapter 6

Software Architecture

Software Architecture

In our project, we followed the static layered architecture to design our system from the perspective of software.

The static layered architecture consists of four main layers:

- Microcontroller Abstraction Layer (MCAL).
- Hardware Abstraction Layer (HAL).
- Services Layer (Service).
- Application Layer (App).
- Library Layer (Library).

MCAL Layer

The MCAL layer contains the software driver module for all on-chip MCU peripheral modules and external devices that are mapped to memory which makes the upper software layer independent of the MCU, such as, RCC Driver, GPIO Driver, USART Driver, I2C Driver, SPI Driver, ADC Driver, AFIO Driver, CAN Driver, NVIC Driver, SYSTICK Driver, FBEC Driver, EXTI Driver, DMA Driver, and Timers Driver.

Figure 6.1 MCAL Layer

HAL Layer

The HAL implements a reusable hardware interface in software which is a way to provide an interface between hardware and software so applications can be device-independent, i.e., The HAL encapsulates the external devices of a microcontroller, such as, SD Card, Screen, Audio, DAC, Speed sensor, Pressure sensor, steering sensor, and rotational sensor, GPS and Wi-Fi.

Figure 6.2 HAL Layer

Service Layer

The Service layer contains such as operating systems that help to interface with the lower layers, in our project, we used the FreeRTOS operating system that has the features that suffice the needs of our project.

Figure 6.3 Service Layer

App Layer

The layer has the user application which has APIs usage of all lower applications.

Figure 6.4 Application Layer

Library Layer

That has the most hand-made libraries used our software such as, BIT Math Library which helps to Read/Write a specific register. STD TYPES which has the standard types that are used in our software. LIB has the definition of the most common keywords that are used in our project.

Figure 6.5 Library

The whole architecture is shown in the following figure.

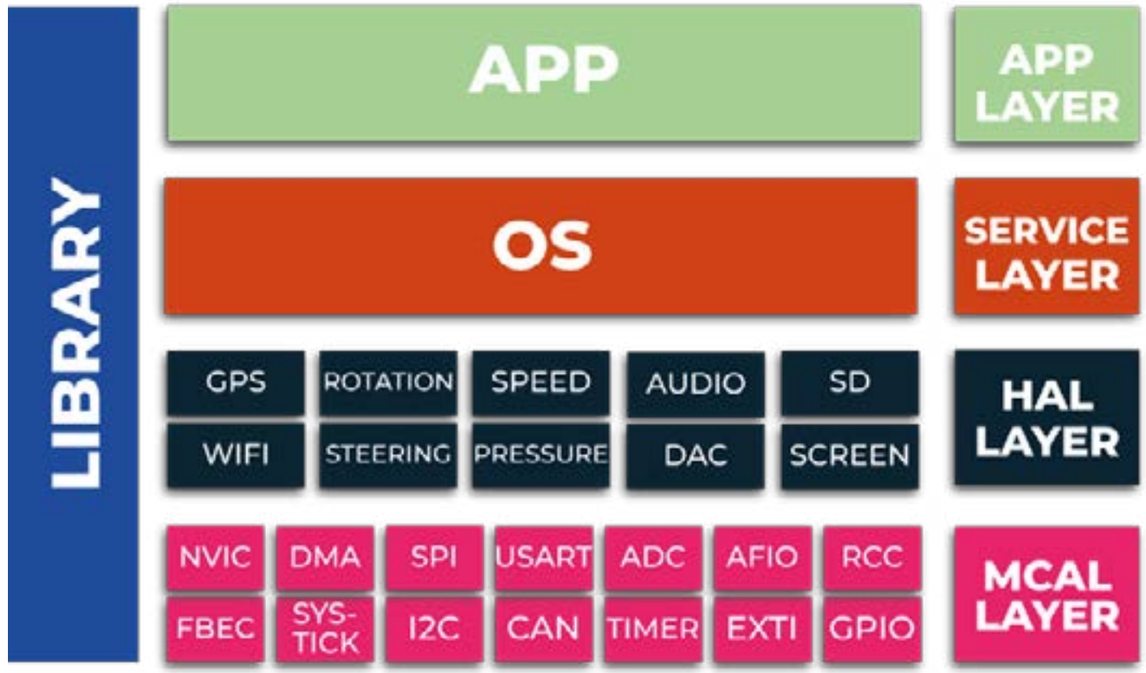


Figure 6.6 SW Architecture

Chapter 7

Hardware Architecture

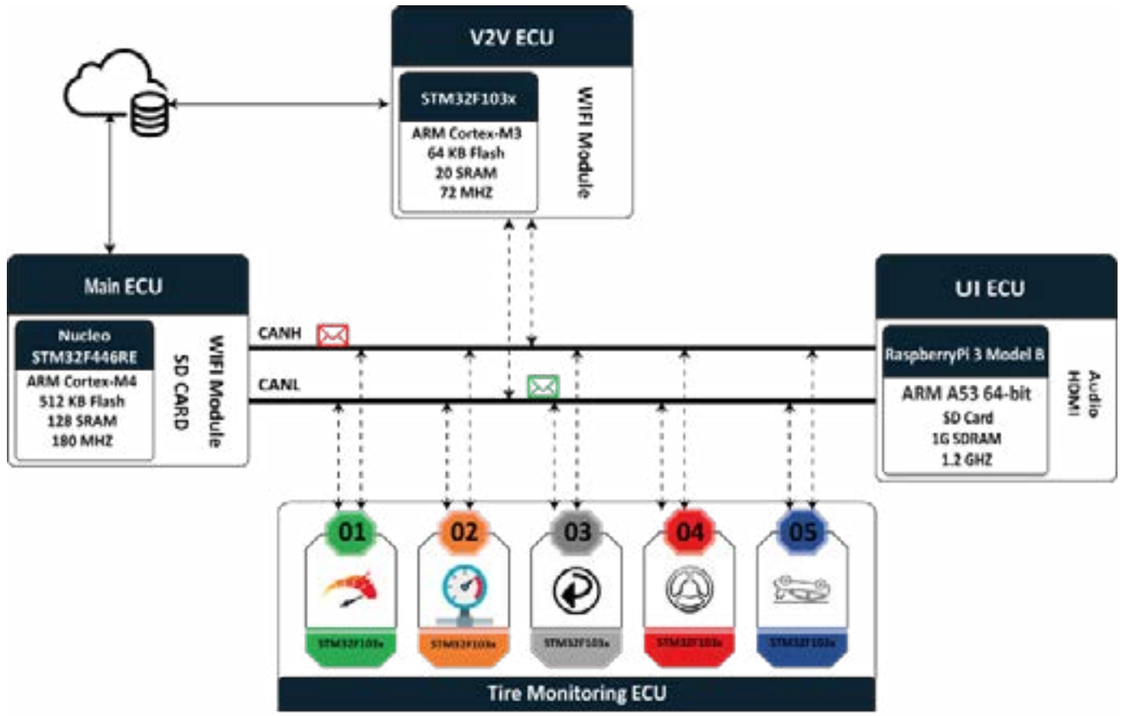


Figure 7.1 HW Architecture

The Hardware Architecture consists of four main ECUs each has a specific task (function). All of these ECUs are connected on the CAN Bus so they can communicate with each other.

The Tire Monitoring ECU:
This ECU contains four microcontrollers, one for the speed of the vehicle, one for the rotational angle, one for the Wheel angle, one for the orientation of the vehicle, and the last for the tire pressure.
Each of these microcontrollers is responsible for measures its analog quantity periodically and sending it over the CAN Bus.

The UI ECU:
consists of an RPi3, which is responsible for reporting the driver with his vehicle Tyer's pressure, its speed, the rotational angle, and the wheel angle through the attached screen. For the critical scenario, the ECU is also responsible to alert the driver through the speaker about what should do to stop the car safely.

The V2V ECU

Responsible for alerting the beside drivers to be notified that there is a problem in this area.

The Main ECU

This ECU has 3 functions:

First, it is responsible for check if the cloud has a new firmware version that needs to be downloaded or not. If there is, it downloads and saves the new firmware on the attached SD Card with its version and sending it over the CAN bus so the owner ECU can reach this update.

Second, it is also responsible for decrypting the new firmware that is got from the cloud before sending it over the CAN Bus.

Third, For the critical scenario, it collects the measured data from the CAN Bus and sending it to the cloud so that it can help to know what the reason for this accident/problem is.

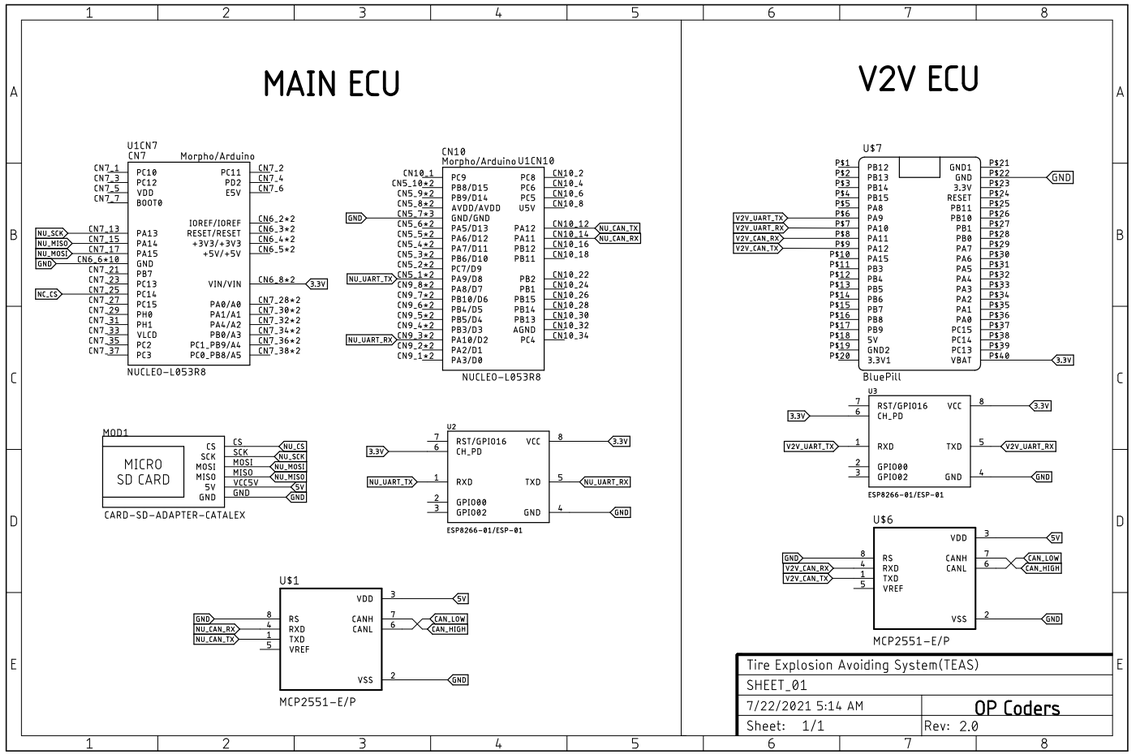


Figure 7.2 Scheme 1

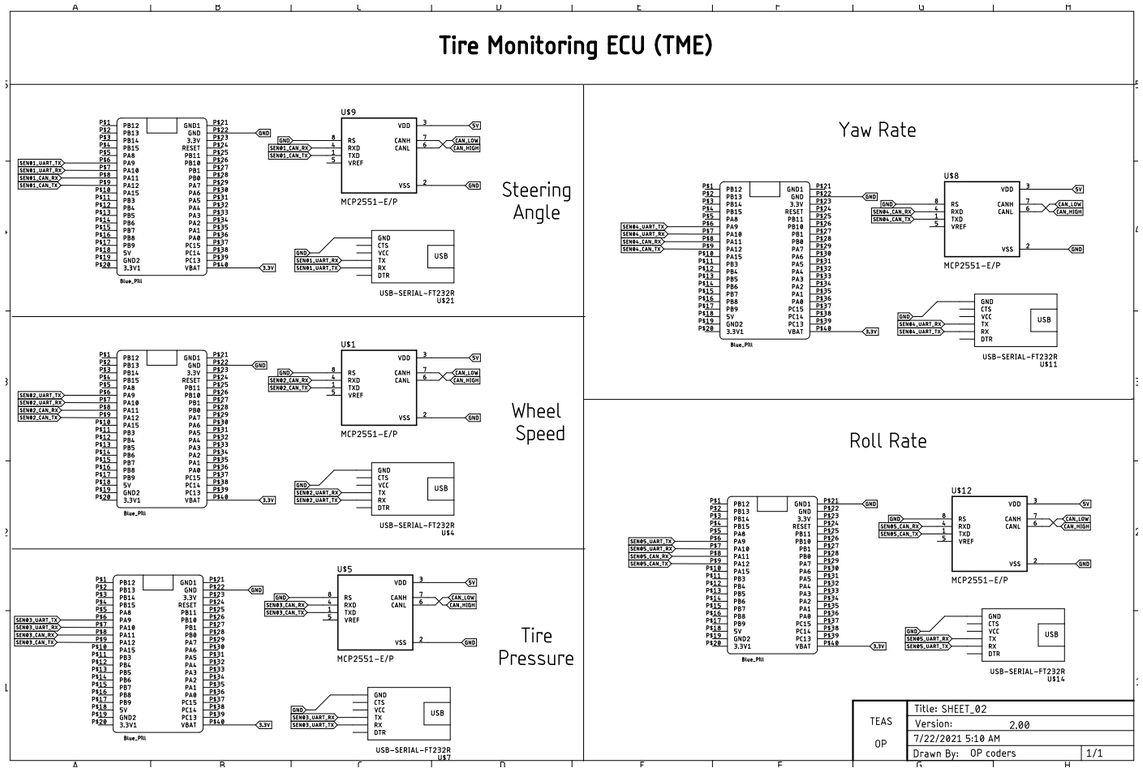


Figure 7.3 Scheme 2

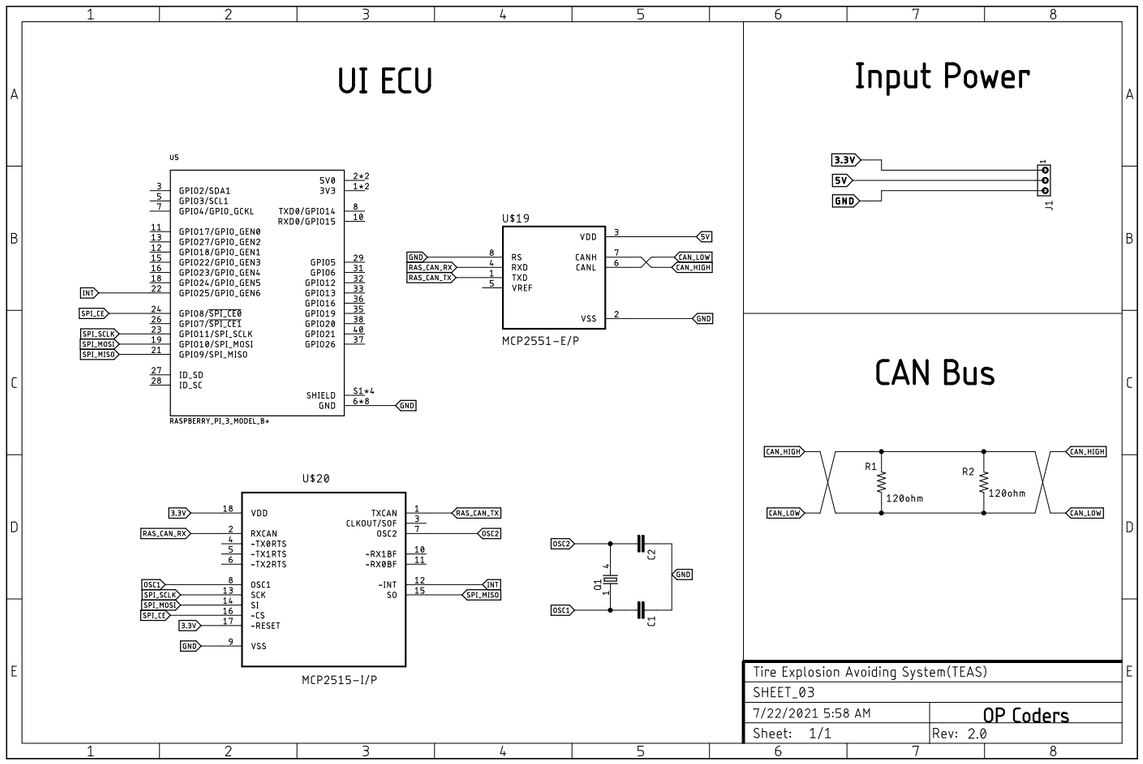
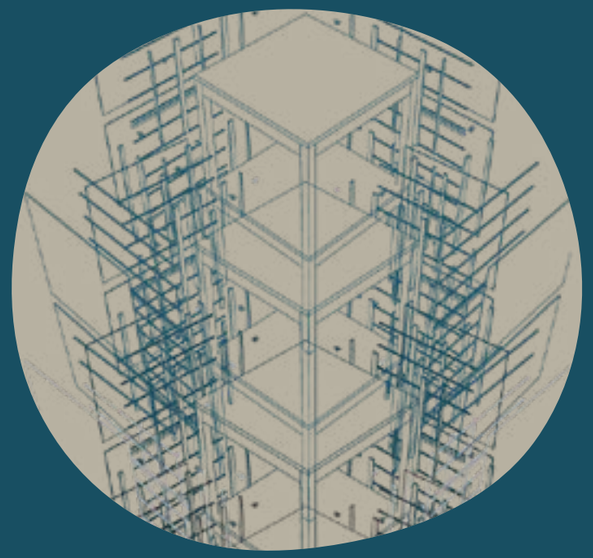


Figure 7.4 Scheme 3



Chapter 8

Architectural Design

8.1 USE Case Diagram

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

- It describes the BEHAVIOR of the system.
- The purposes of use case diagrams can be as follows:
    1- Used to gather Functional Requirements of a system.
    2- Show the interaction between the Requirements and Actors.

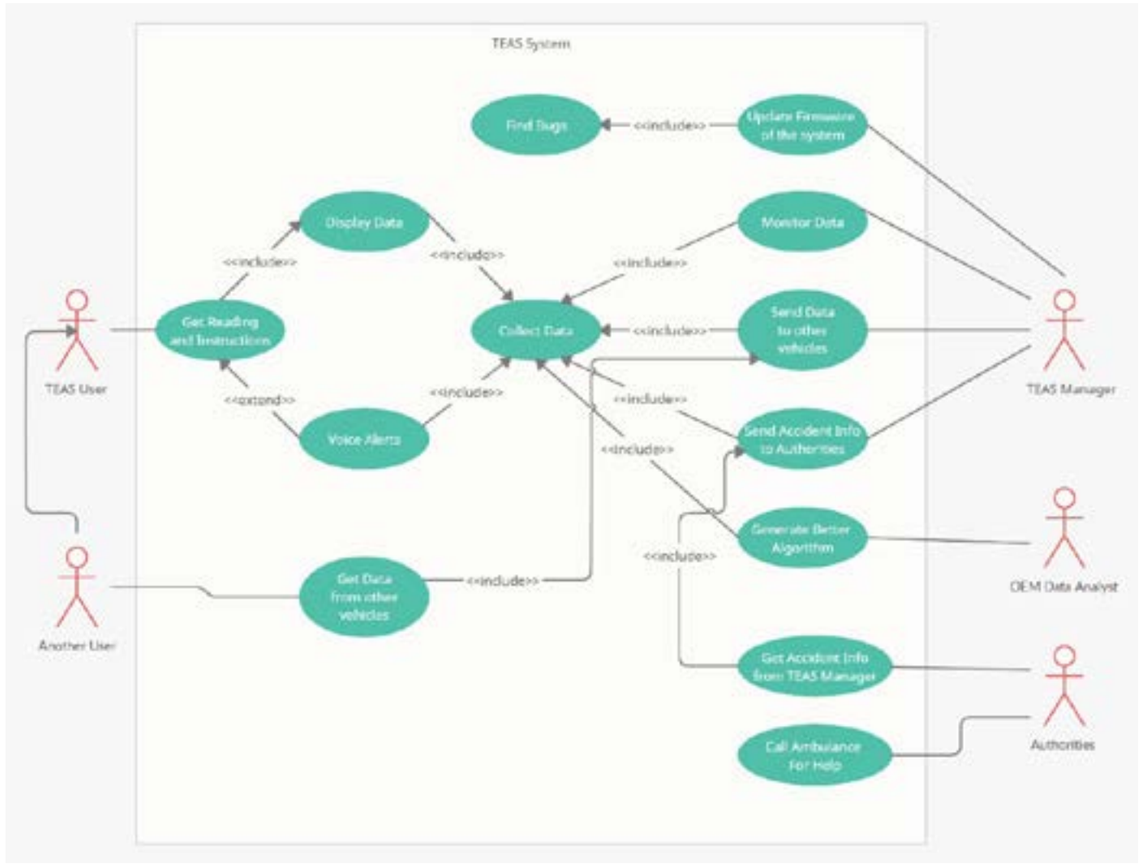
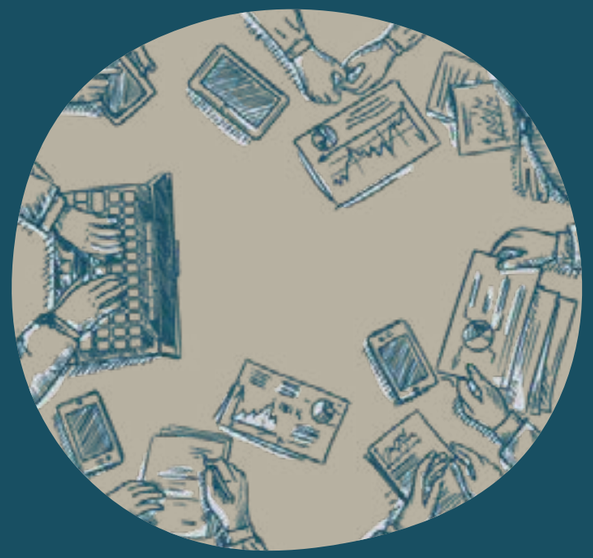


Figure 8.1 Use Case Diagram



Chapter 9

Future Work

9.1 Project Phases

Phase 1 – Getting ready for the project by searching for the best idea to help as many people as possible.

Phase 2 – After finding the idea, we started searching to study the basic knowledge we need to learn.

Phase 3 – After Studying, we started dividing the whole team into 3 sub-teams to work on the main features in parallel.

Phase 4- Each team starts to search for the libraries and frameworks that will achieve the target of each feature.

Phase 5 – For a long time of working each team build the feature their separately.

Phase 6 – All the features were integrated to be working together with the core feature, the voice commands.

9.2 Future Work

Our goal is to add more features to this project and enhance it by:

1- Add more Languages to our system and voice commands.
2- Implement our system on a real car and test it.
3- Held an agreement with big companies to get real sensors.
4- Held agreements with countries local authorities to integrate our system with countries systems.
5- Re-Integrate the system using **Autosar**.

Chapter 10
Conclusion

Conclusion

In this book, through nine chapters, we were able to clarify the basic idea of the project and with the technical aspects that we were exposed to throughout the work time, and also the theoretical aspects of the scientific parts of the project. The first step of the project started when we run into a tire blowouts problem and its negative effects on the drivers of the vehicles, to start with brainstorming to create a list of possible solutions and decide a solution through our engineering field specifically the “automotive embedded system”, which helped us create planning and start executing it. This sequence is illustrated by the seven chapters respectively.

In the first chapter, We have identified the problem we are working on, The Tire Explosion Accidents and In light of our field, we searched for the Current Solution Which ensures the mandatory use of tire pressure sensors in cars, and set a five main goals to start creating a system that helps solve the problem, Enhanced Tire monitoring system, Firmware over the air updates, OEM Data Collection System, Vehicle to Vehicle Communication and Authorities Informing. To start with these objectives the idea of our project, which can be viewed horizontally as Tire Explosion Avoiding system (TEAS) which aims to reduce the number of those accidents by monitoring data of the tires and depending on these reading, which will decide if it’s a must to warn the driver early whether there’s a problem with his tires or not. In case of a tire blowout, (TEAS) will collect various data about car movement to send them to a control unit for making its data analysis to put our hands on the cause of that tragedy and if the car is autonomous, the control unit will stop the car immediately, in the case of manual driving it will help the driver to stop safely without losing control.

in the second chapter, we summarize the technical background, theoretical knowledge, practical knowledge, and different programming languages and tools used within the project.
In the third chapter, we started analyzing and clarifying the monitoring system, start with the sensors to be used in the system that works with a computer to improve vehicle stability by detecting and reducing the loss of traction. The most important sensor is the Yaw and Roll rate sensor, Wheel speed sensor, Steering angle sensor, and Tire-Pressure Sensor, and through used a CARLA® Simulator, we can simulate these sensors.

Through this simulator, we were able to simulate the movement of the car in several different conditions and obtain from it many sensor readings that were difficult to obtain in the real world as a project experiment.

In the fourth chapter, we discussed the communication among cars and an attempt to use the best method of communication and its application, and clarify several concepts, like V2V is an acronym Vehicle-to-vehicle and V2C which acronym for vehicle-to-cloud communication. We have clarified the purpose of using the V2C system which Through it we will be able to apply the V2V idea, C2(Authorities), and V2(OEM).in the V2V system after the data monitoring stage, (TEAS) uses this data to send warnings to the adjacent cars to avoid subsequent accidents. In V2(OEM) is sending this data to the Original Equipment Manufacturer (OEM) where their data analysis team can make use of it to generate a better algorithm to avoid occurring these problems again. In C2(Authorities) will be able to communicate with authorities in case of an accident has happened, to sends car's ID, type and it's the location to the authorities to tell them that there's an accident happened, The authorities will take the needed procedures to help the injured people by calling an ambulance and helping others by broadcasting breaking news with the road that accident has happened on it to take alternative paths and preventing subsequent accidents.

In the fifth chapter, we discussed the Firmware Over the Air system (FOTA) for updating the software remotely and its features like solve the problem of reliability of each update that used to be present earlier, and helps the Component Supplier (tier 2 supplier) so much, the dependency of their service stations carrying the vehicle to the station, the cost of updating the software becomes very low. Also eases the role of the client, no much hassle to the customer, One-click will allow him to update the software of his car immediately. We started with the flashing process to updating the content of the flash memory and clarifying the different programming techniques for this process. Then we clarifying the Booting idea which is the initial set of operations that a computer system performs when electrical power is switched on and how to match the bootloader is important to Updating the system with new features.

In the sixth chapter, We have clarified the software structure that we follow in the work specifically the static layered architecture which consists of four main layers, Microcontroller Abstraction Layer (MCAL) which contains the software driver module for all on-chip MCU peripheral modules and external devices that are mapped to memory, Hardware Abstraction Layer (HAL) which provides an interface between hardware and software and contains encapsulates the external devices of a microcontroller, Services Layer (Service) which contains such as operating systems that help to interface with the lower layers, Application Layer (App) which provides the user application APIs, and last, Library Layer (Library) which contains many ready-made macros and APIs that can be used repeatedly with other layers.
In the seventh chapter, Includes hardware infrastructure first is the block diagram for the Electronic Control Units ECUs in the system (Main ECU, V2V ECU, Monitoring ECU, UI ECU), and the schematic diagram for each Electronic Control Units designed by us.

Chapter 11
References

References

[1] SAE. Class A Application Definition. J2507-1, 1997. http://standards.sae.org/j2057/1200609/.C. A.Lupini. In-vehicle networking technology for 2010 and beyond. s.l. : SAE International, 2010.

[2] D. Hristu-Varsakelis and W. Levine. Handbook of Networked and Embedded Control Systems. Birkhauser, Basel, Switzerland, 2005.

[3] J. Maslouh, A. Errami, and M. Khaldoun. Resolving the access conflict for shared Ethernet communication channel. IEEE International Conference on Next Generation Networks and Services, University of Oxford, 80–87, 2014.

[4] Vehicle Safety Communications ProjectVFinal Report, U.S. Dept. Trans., Nat. Highway Traffic Safety Admin., Rep. DOT HS 810 591, Apr. 2006.

[5] IEEE Standard for Information TechnologyVTelecommunications and Information Exchange Between SystemsVLocal and Metropolitan Area NetworksVSpecific Requirements; Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications; Amendment 6: Wireless Access in Vehicular Environments, IEEE Std. 802.11p, Jul. 2010.

[6] Draft DSRC Message Communication Minimum Performance RequirementsVBasic Safety Message for Vehicle Safety Applications, SAE Draft Std. J2945.1 Revision 2.2, SAE Int., DSRC Committee, Apr. 2011.

[7] CARLA Documentation Website, https://carla.readthedocs.io/en/latest/

[8] MPU-6000 and MPU-6050 Product Specification Revision 3.4. J2507-1, PS-MPU-6000A-00 Revision: 3.4 Release Date: 08/19/2013.

[9] PM0214 Programming manual STM32 Cortex®-M4 MCUs and MPUs programming manual, March 2020 PM0214 Rev 10.

[10] CAN-Bus Troubleshooting Guide. esd electronic system design gmbh Vahrenwalder Str. 207,30165 Hannover, Germany.

[11] STM32 advanced Arm®-based 32-bit MCUs RM0008 Reference manual. December 2018.

[12] MCP2551 High-Speed CAN Transceiver, Microchip Technology Inc., 2007.

[13] MCP2515 Stand-Alone CAN Controller with SPI Interface, Microchip Technology Inc., 2007-2019.

[14] Steering Wheel Angle Sensor LWS, © Bosch Engineering GmbH 2021 | Data subject to change without notice, V, 11. Feb 2021.

[15] Yaw Rate Sensor YRS 3, © Bosch Engineering GmbH 2021 | Data subject to change without notice, V, 11. Feb 2021.

[16] Speed Sensor Hall-Effect HA-P, © Bosch Engineering GmbH 2021 | Data subject to change without notice, V, 11. Feb 2021.

[17] Tire Pressure Monitoring Sensor SP30 media.digikey.com/Infineon, 2014.

[18] SCC1300-D02 Combined Gyroscope and 3-axis Accelerometer with digital SPI interfaces, Murata Electronics Oy, Doc.Nr. 82113000, 2013.

[19] Raspberry Pi 3 Model B Datasheet.

[20] Off-line vs In-System Programming. by Scott Bronstad | 2021. https://bpmmicro.com/off-line-vs-in-system-programming/

[21] What do Hash Collisions Really Mean? by jered floyd | 2005. https://permabit.wordpress.com/2008/07/18/what-do-hash-collisions-really-mean/

[22] One Way Hash Function, by Jeff Gilchrist, in Encyclopedia of Information Systems, 2003. https://www.sciencedirect.com/topics/computer-science/one-way-hash-function

[23] 12 Remarkable Difference between MD5 and SHA-1, by Core Differences admin, 2021. 12 Remarkable Difference between MD5 and SHA-1 with Table - Core Differences

[24] Announcing the first SHA1 collision, by Marc Stevens (CWI Amsterdam), Elie Bursztein (Google), Pierre Karpman (CWI Amsterdam), Ange Albertini (Google), Yarik Markov (Google), Alex Petit Bianco (Google), Clement Baisse (Google) , 2017. Google Online Security Blog: Announcing the first SHA1 collision (googleblog.com)

[25] Advanced Encryption Standard (AES), by Morris J. Dworkin, Elaine B. Barker, James R. Nechvatal, James Foti, Lawrence E. Bassham, E. Roback, James F. Dray Jr, 2001. Advanced Encryption Standard (AES) | NIST