Software Requirement Specification

Document

Dash Data Display STM32

|  |  |
| --- | --- |
| Document ID | SOFTWARE REQUIREMENT SPECIFICATION-v0.1 |
| Version Number | 0.3 |
| Issue Date | 26.04.2023 |
| Classification | Public |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author (s)** |
| 30/12/2022 | 0.1 | Draft Version | Miroslav Mitrovic |
| 14/04/2023 | 0.2 | Changed I2C to SPI | Miroslav Mitrovic |
| 16/04/2023 | 0.3 | Changed File Structure/Added Adapter classes to components | Miroslav Mitrovic |
| 25/04/2023 | 0.4 | Changed file Structure/Removed Adapter class and added HW\_Wrapper component | Miroslav Mitrovic |
| 26/04/2023 | 0.5 | Added Pressure Sensor Interfacing Circuit- HW update | Miroslav Mitrovic |

**Table of Contents**

[1 Introduction 6](#_Toc286849724)

[1.1 Purpose 6](#_Toc286849725)

[1.2 Document Conventions 6](#_Toc286849726)

[1.3 Intended Audience and Reading Suggestions 6](#_Toc286849727)

[1.4 Project Scope 6](#_Toc286849728)

[1.5 References 6](#_Toc286849729)

[1.6 Acronyms and Abbreviations 7](#_Toc286849730)

[2 Overall Description 7](#_Toc286849731)

[2.1 Product Perspective 7](#_Toc286849732)

[2.2 Product Features 7](#_Toc286849733)

[2.3 User Classes and Characteristics 7](#_Toc286849734)

[2.4 Operating Environment 8](#_Toc286849735)

[2.5 Design and Implementation Constraints 8](#_Toc286849736)

[2.6 User Documentation 8](#_Toc286849737)

[2.7 Assumptions and Dependencies 8](#_Toc286849738)

[3 System Features 8](#_Toc286849739)

[3.1 System Feature 1 9](#_Toc286849740)

[3.1.1 Description and Priority 9](#_Toc286849741)

[3.1.2 Stimulus/Response Sequences 9](#_Toc286849742)

[3.1.3 Functional Requirements 9](#_Toc286849743)

[3.2 System Feature 2 (and so on) 10](#_Toc286849744)

[4 External Interface Requirements 10](#_Toc286849745)

[4.1 User Interfaces 10](#_Toc286849746)

[4.2 Hardware Interfaces 10](#_Toc286849747)

[4.3 Software Interfaces 10](#_Toc286849748)

[4.4 Communications Interfaces 10](#_Toc286849749)

[5 Other Nonfunctional Requirements 11](#_Toc286849750)

[5.1 Performance Requirements 11](#_Toc286849751)

[5.2 Safety Requirements 11](#_Toc286849752)

[5.3 Security Requirements 11](#_Toc286849753)

[5.4 Software Quality Attributes 11](#_Toc286849754)

[6 Other Requirements 12](#_Toc286849755)

[Appendix A: Glossary 12](#_Toc286849756)

[Appendix B: Analysis Models 12](#_Toc286849757)

[Appendix C: Issues List 12](#_Toc286849758)

# Introduction

## Purpose

This document shall cover the functional and non-functional software requirements. This SRS will focus only on SW and interfaces of the Dash Data Display Application.

## Document Conventions

For higher level abstraction System Architecture shall be followed.

## Intended Audience and Reading Suggestions

This document shall be readable by team members that are working in SW development, architecture and testing. This document is organized in a multiple chapters. Each of chapters provide corresponding information. Reading sequence shall be followed as it is in the document.

## Project Scope

Dash Data Display Application has a goal to provide valuable information to the Driver from Engine Management ECU and other available sensors(Vehicle speed, Oil Pressure and Temperature). This will be further explained in Functional Requirements.

## References

SW Architecture and SW/System Architecture diagrams are available at :  
<https://github.com/MiroslavMitrovic/IgnitionController_STM32/tree/main/Design>

## Acronyms and Abbreviations

<The following acronyms and abbreviation will apply to this document.

|  |  |
| --- | --- |
| SRS  FEE | Software Requirement Specification>  Flash Eeprom Emulation |

# Overall Description

## Product Features

Dash Data Display shall have following features:

* Display Relevant Engine and Sensor data via LCD Display
* Inform Driver about Engine/Sensor Malfunctions via Warnings
* Enter a start mileage(default shall be 0, but at production this value can be entered to any value between 0-999999)
* Calculate and display correct mileage
* Driver can reset the mileage for a trip computer
* Maximum Value of kilometers for trip computer

## User Classes and Characteristics

Following Classes shall be used:

* ~~HW\_Wrapper(to provide the separation between Application and HW Abstraction Layer)~~
* Adc\_Wrapper(to provide the separation between Application and HW Abstraction Layer for ADC component)
* Can\_Wrapper(to provide the separation between Application and HW Abstraction Layer for CAN component)
* Gpio\_Wrapper(to provide the separation between Application and HW Abstraction Layer for GPIO component)
* Spi\_Wrapper(to provide the separation between Application and HW Abstraction Layer for SPI component)
* Uart\_Wrapper(to provide the separation between Application and HW Abstraction Layer for UART component)
* Flash\_Wrapper(to provide the separation between Application and HW Abstraction Layer for Flash component)
* CAN\_Main(Main Class)
* CAN\_Processor(To process the CAN data)
* CAN\_Interface(To create interfaces for CAN data to be passed to other classes)
* ~~CAN\_Adapter(To create interface between Hardware Abstraction Layer)~~
* ~~CAN\_Proxy(Proxy class for sending the data)~~
* SPI\_Main(Main Class)
* SPI\_Processor(To process the I2C data)
* SPI\_Interface(To create interfaces for I2C data to be passed to other classes)
* ~~SPI\_Adapter(To create interface between Hardware Abstraction Layer)~~
* ~~SPI\_Proxy(Proxy class for sending the data)~~
* UART\_Main(Main Class)
* UART\_Processor(To process the I2C data)
* UART\_Interface(To create interfaces for I2C data to be passed to other classes)
* ~~UART\_Adapter(To create interface between Hardware Abstraction Layer)~~
* ~~UART\_Proxy(Proxy class for sending the data)~~
* ADC\_Main(Main Class)
* ADC\_Processor(To process the ADC data)
* ADC\_Interface(To create interfaces for ADC data to be passed to other classes)
* ~~ADC\_Adapter(To create interface between Hardware Abstraction Layer)~~
* Odometer\_Main(Main Class)
* Odometer\_Processor(To process the Sensor interrupt data)
* Odometer\_Interface(To create interfaces for VSS data to be passed to other classes)
* ~~Odometer\_Adapter(To create interface between Hardware Abstraction Layer)~~
* LCD\_Main(Main Class)
* LCD\_Processor(Functions for LCD display handling)
* LCD\_Interface(To create interfaces to LCD data to be passed to other classes)
* ~~LCD\_Adapter(To create interface between Hardware Abstraction Layer)~~
* Odometer\_Main(Main Class)
* Odometer\_Processor(To process odometer data)
* Odometer\_Interface(To create interfaces for odometer data to be passed to other classes)
* ~~Odometer\_Adapter(To create interface between Hardware Abstraction Layer)~~
* FEE\_Interface(To create interfaces for odometer data to be passed to other classes)
* FEE\_Wrapper(To have a possibility to wrap different FEE stacks)
* IgnitionSig\_Main(Main Class)
* IgnitionSig\_Processor(Process the ignition Signal Availability)
* IgnitionSignal\_Interface(To create interfaces for Ignition Signal availability to be passed to other classes)
* ~~IgnitionSignal\_Adapter(To create interface between Hardware Abstraction Layer)~~
* DashDataDisp\_Main(Main class of the Application, here is defined the main function of the app)

## Operating Environment

Application shall be constructed to be hardware independent, so that can be easily ported to the other hardware platform. Change of display it should be also possible by including corresponding Display Drivers. As a starting point STM32F407VGT6 MCU, and 20x4 characters LCD 1602 Display with I2C shield is used. Dash Data Display app shall run under FREE\_RTOS OS, version 202212.00.

## Design and Implementation Constraints

MCU shall be clocked at 168MHz(Maximum speed). LCD shall output the data at 100ms rate. Data processing shall be performed at 25ms rate. CAN Bus data from Engine Management ECU is sent at 20ms rate with datarate of 500kb. CAN bus base id is 0x5e8.CAN DBC is available in design folder of project. Flash EEPROM Emulation shall be used for storing the mileage data into flash. Memory is limited with 1MB of Flash and 128k of RAM.

## User Documentation

Additional documents are available in Design/Requirements folder.

Following Documents are available:

* Megasquirt\_CAN\_Broadcast.pdf (Description about the broadcasted Data from ECU on CAN bus)
* AN3969\_EEPROM.pdf (Application Note about EEPROM implementation from STM)
* dm00031020-stm32f405415-stm32f407417-stm32f427437-and-stm32f429439-advanced-armbased-32bit-mcus-stmicroelectronics.pdf(Reference Manual for MCU)
* stm32f407vg.pdf(Datasheet for MCU)

## Assumptions and Dependencies

Dependencies are following:

* FREE\_RTOS OS shall be integrated
* STM32 HAL generated code from STM 32 Cube IDE(v.1.10.1) will be used
* 3rd party SW for Flash EEPROM Emulation will be used from STM and integrated
* STM32 Cube IDE v.1.10.1 will be used for development purposes

# System Features

This chapter shall represent the system features that Display Dash Data shall provide.

## Display Relevant Engine and Sensor Data

This feature shall provide the relevant engine and sensor data to the driver via LCD display.

### Description and Priority

Displayed data shall contain information from engine management ECU via CAN bus and external sensors via ADC interfacing. Display of this data has a Medium Priority. There are two pages available with relevant data. User between the pages by a press of a button.

### Stimulus/Response Sequences

User actions are :

* Press Page 1 button -> show Page 1 data
* Press Page 2 button -> show Page 2 data

### Functional Requirements

Below are functional requirements:

REQ-1: Default LCD display will show the Page 1 layout

REQ-2: By pressing the Page1 button switch LCD shall revert to the Page 1

REQ-3: By pressing the Page2 button switch LCD shall switch to Page 2 layout

REQ-4: Page 1 will contain following data:

|  |  |
| --- | --- |
| Source | Data |
| CAN Bus | Engine RPM |
| GPIO Interrupt | Vehicle Speed Signal |
| Derived | Mileage |
| Derived | Trip Mileage |

REQ-5: Page 2 will contain following data:

|  |  |
| --- | --- |
| Source | Data |
| CAN Bus | Manifold Air Temperature |
| CAN Bus | Manifold Air Pressure |
| CAN Bus | Battery voltage |
| ADC | Oil Temperature |
| ADC | Oil Pressure |

REQ-5: TBD

REQ-6: TBD

REQ-7: TBD

REQ-8: TBD

## Display System Malfunction Warnings

This feature shall provide Warnings to the Driver via LCD display.

### Description and Priority

Warnings will show as a message that corresponding value is below/above threshold. Values that are considered are:

* Oil Pressure
* Oil Temperature
* Battery Voltage
* Engine RPM

Display of these warnings are considered with High Priority.

### Stimulus/Response Sequences

Warnings will be shown at order mentioned above. If there is more than one warning present, display will switch between the warnings in time sequence of 3 seconds for each warning.

### Functional Requirements

Below are Functional requirements:

REQ-9: Oil Pressure threshold warning value is less than 1 bar

REQ-10: Oil Temperature threshold warning value is greater than 140 degC

REQ-11: Battery Voltage threshold warning value is less than 12 Volts.

REQ-12: TBD

## Display Mileage Data

This feature shall provide Mileage data to the Driver via LCD display.

### Description and Priority

Mileage data shall be displayed at Page 1 of LCD display. Mileage data is derived from calculation based on inputs from Vehicle Speed Sensor and Vehicle Parameters. Mileage calculation will be based on how many pulses are per one kilometer. Storing of Mileage data will be performed at Ignition OFF signal by writing the data into FEE. Display of Mileage data are considered with Low Priority.

### Stimulus/Response Sequences

Mileage value shall be updated by increment of 1, and only when value is updated, updated value will be shown on LCD display.

### Functional Requirements

Below are Functional requirements:

REQ-13: Mileage Calculation shall be performed by measuring the pulses from VSS signal.

REQ-14: Mileage value shall be updated at increments by 1.

REQ-15: Persistency of Mileage value shall be performed by storing latest value in FEE when ignition ON-->OFF sequence is performed.

REQ-16: TBD

## Display Vehicle Speed

### Description and Priority

Vehicle Speed shall be displayed at Page 1 of LCD display. Vehicle Speed data is derived from calculation based on inputs from Vehicle Speed Sensor and Vehicle Parameters. Vehicle Speed calculation will be based on how many pulses are unit of time.

### Stimulus/Response Sequences

Vehicle Speed value shall be updated be updated at each 1 second. Updated value will be shown on LCD display.

### Functional Requirements

Below are Functional requirements:

REQ-17: Vehicle Speed Calculation shall be performed by measuring the pulses from VSS signal and calculating the speed by dividing it at 1 s with corresponding time factor.

Formula for vehicle speed calculation is following:

Where n is RPM and rd is Dynamic Rolling Radius of the vehicle(vehicle parameters)

in FEE when ignition ON-->OFF sequence is performed.

REQ-18: TBD

REQ-19: TBD

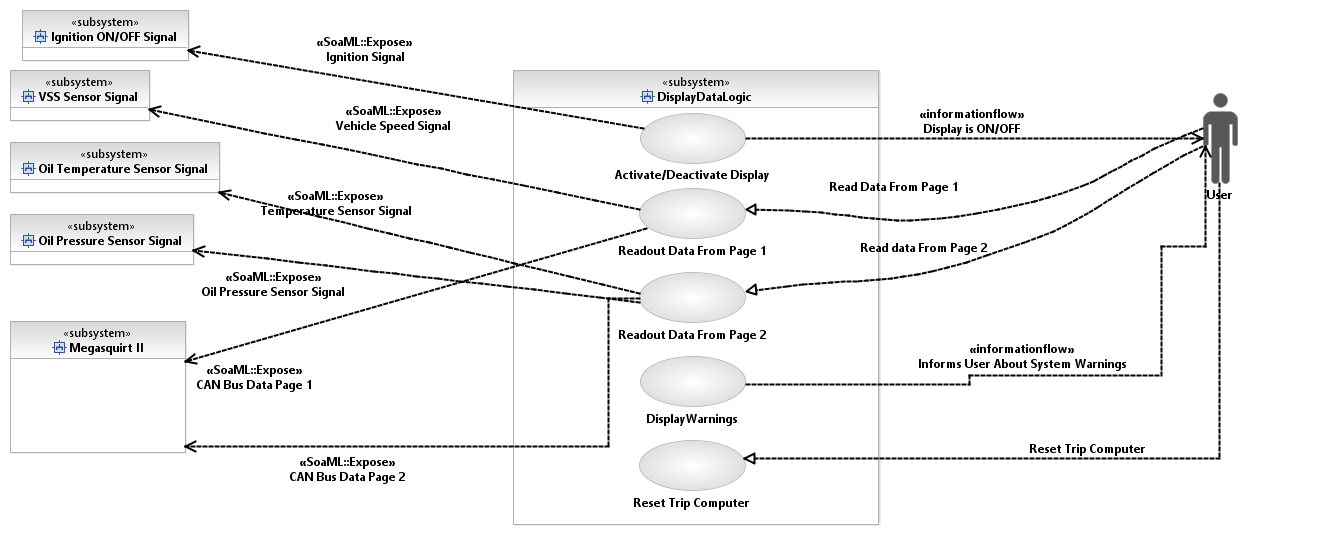
REQ-20: TBD

# External Interface Requirements

## User Interfaces

As an user interface Three Buttons will be used, one for page one selection, second one for page two selection and the third one for resetting the Trip Mileage.

Use Case diagram is below.



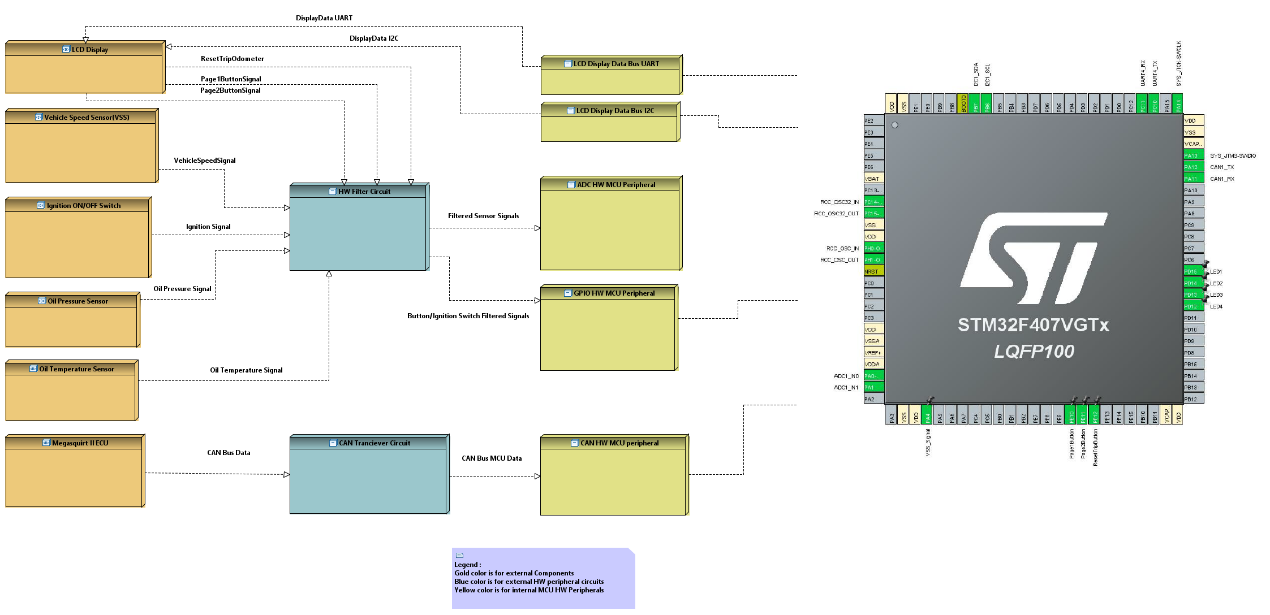
**Figure 1 Use Case Diagram**

## Hardware Interfaces

Hardware interfaces are following:

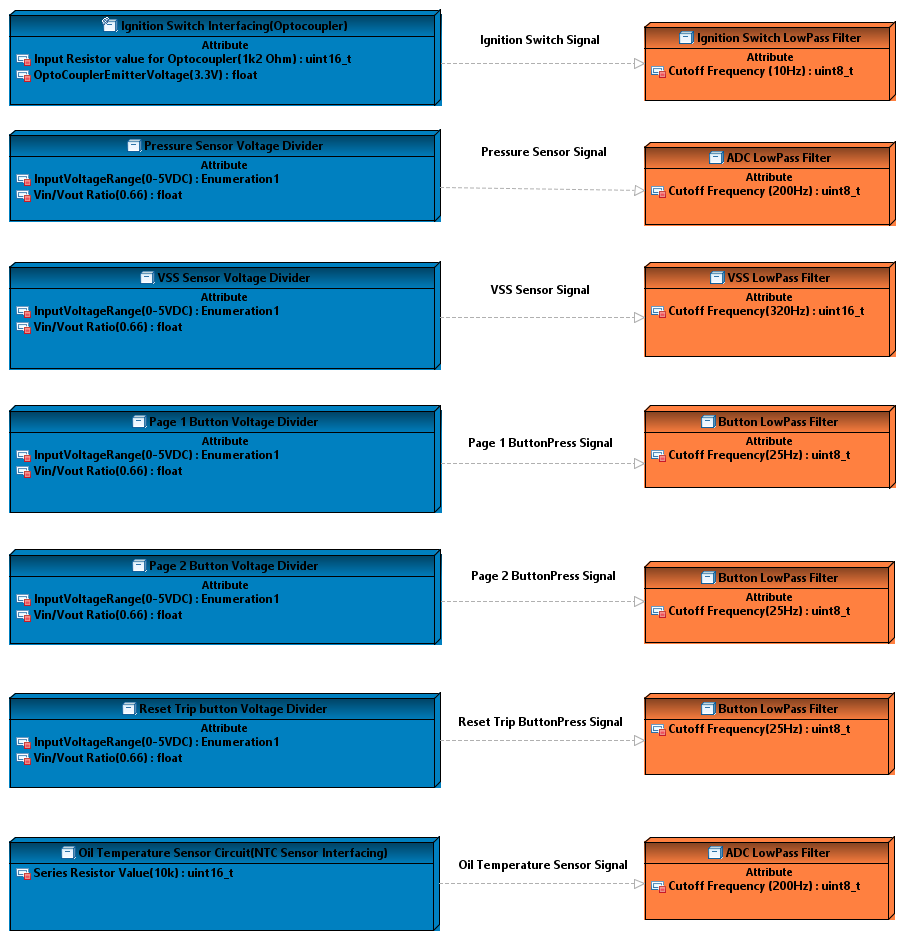
* LCD Display shall have I2C or UART interfacing
* Vehicle Speed Sensor is Hall Sensor that shall be filtered and after filtering interfaced via GPIO interrupt routine
* Ignition signal- 12+VDC signal shall be filtered and reduced to supported Voltage level for ADC HW peripheral in MCU.
* Oil Pressure sensor shall be filtered and reduced to supported Voltage level for ADC HW peripheral in MCU. Voltage signal range shall be between 0.5VDC – 4.5VDC. Please see the datasheet of chosen sensor.
* Oil Temperature sensor shall be filtered and reduced to supported Voltage level for ADC HW peripheral in MCU. Sensor is usually of NTC type, so the proper coefficients and calculation must be performed.Please see the datasheet of chosen sensor.
* Data from Megasquirt ECU shall be received via CAN bus. CAN bus signal will go to Transciever Circuit and Transciever will forward the processed data to CAN HW peripheral of MCU.
* Ignition Switch detection signal shall be interfaced via the optocoupler and filtered via low pass filter. Further it will be sent to GPIO HW peripheral of MCU.

Hardware interfaces Layout picture is below.



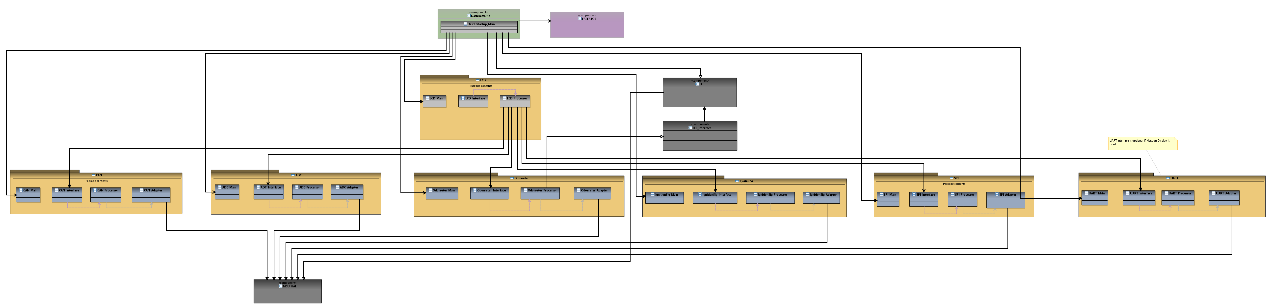
**Figure 2 HW Interfaces**

Hardware Filter Circuit Layout Diagram is below:



**Figure 3 HW filter Circuit Layout Diagram**

## Software Interfaces -TBD



**Figure 4 SW Component Layout Diagram**

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

## Communications Interfaces

Following Communication interfaces are used:

* Two UART interfaces, one for FREE RTOS realtime debugging and other one for LCD Display(optional- Nextion Display is used). Setup will be as follows : For Debugging interface(500kbaud), and for LCD (115200baud)
* CAN interface for receiving data from MS II ECU shall have following setup : using CAN standard addressing format and 500k baud rate.
* SPI interface is used for Different type of LCD controller(1602 20x4 LCD display).

# Other Nonfunctional Requirements

## Performance Requirements

REQ-21: All available data shall be processed and displayed to the driver via LCD in 100ms cyclic periods.

## Safety Requirements

REQ-22: As a safety measure simple Watchdog Triggering will be implemented. If Watchdog timer window expires, Application shall initiate a reset.

## Security Requirements

Security Requirements are not in the scope of this project.

## Software Quality Attributes

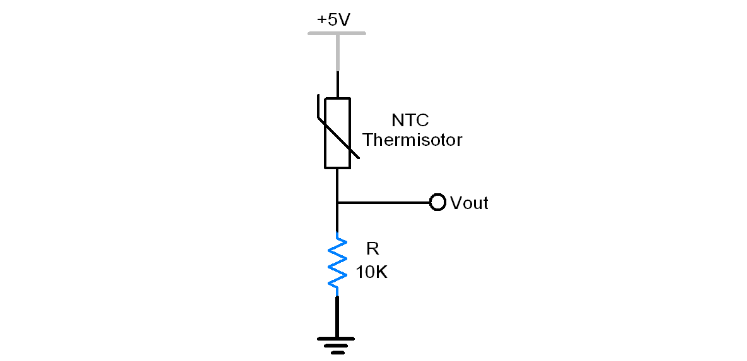
Following Software Quality Attributes shall be adhered to:

* adaptability (interfaces shall be modified easily without jeopardizing other interfaces)
* correctness (SW shall be thoroughly tested)
* portability (SW shall be easily ported to other MCUs or LCDs)

## Hardware Requirements

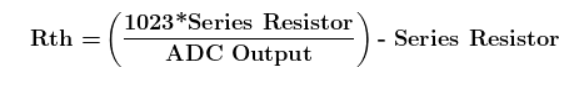
### Oil Temperature Sensor HW requirements

REQ-23: Oil Temperature Sensor shall use 10 BIT ADC conversion. Example Circuit for NTC Thermistor is below :



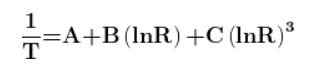
**Figure 5 NTC Thermistor Cicrcuit**

REQ-24: Oil Temperature Sensor temperature value shall be determined based on NTC sensor resistance. From equation below we can get the resistance.



**Figure 5 Determining the NTC Resistance Value**

REQ-25: Temperature will be calculated based on NTC sensor resistance using Steinhart Hart equation.



**Figure 6 Steinhart Hart equation**

### Oil Pressure Sensor HW requirements

REQ-26: Oil pressure sensor shall use 10 BIT ADC conversion.

Chosen Oil pressure sensor if of Resistor type, with measuring range from 1.0 to 10 bars. Resistance is almost linear(difference due to measuring error of measurement setup). Due to the sensor type following requirement is derived.

REQ-27: Oil pressure sensor characteristic shall be linear, or close to linear.

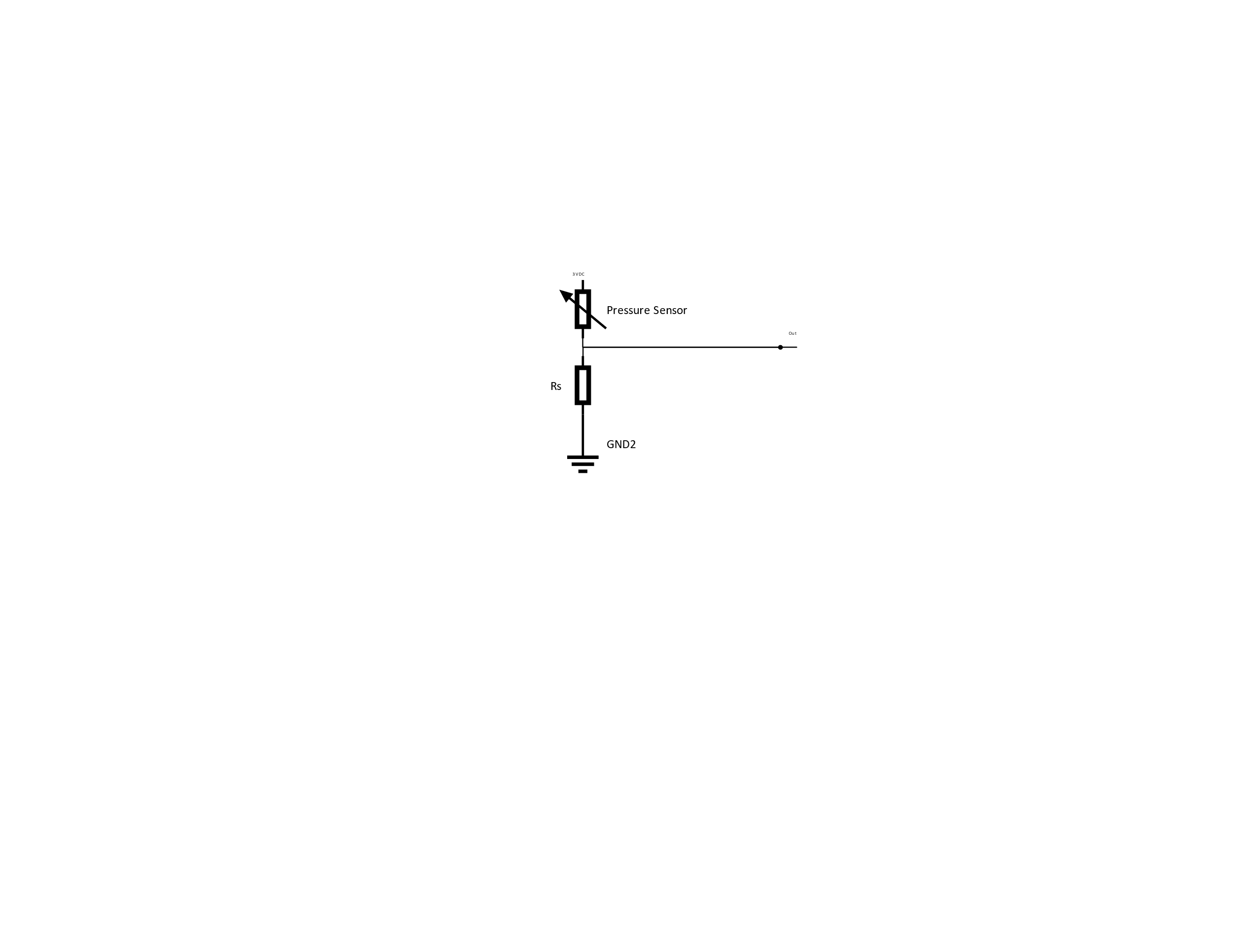
After the fitting of measured data from sensor testing, following polynomial coefficients are derived for sensor characteristic :

* P1 = 12.5737
* P2 = 19.3474

From there Pressure sensor characteristic is :

y = P1 \* x + P2 = 12.5737 \* x + 19.3474

For interfacing to the MCU ADC peripheral voltage divider circuit will be used, picture is below.



**Figure 7 Pressure Sensor Interfacing Circuit**

Voltage divider resistor Rs has the value of 270 Ohms.

# Other Requirements

TBD

# Appendix A: Glossary

TBD

# Appendix B: Analysis Models

TBD

# Appendix C: Issues List

TBD