|  |
| --- |
| \_\_\_\_  ***Spezification***  *Pololu Zumo32U4*  \_\_\_\_  Doc.-Number:  SpezificationPololuZumo32U4  Doc.-Version: A2 |
| Customer:  NewTec  Buchenweg 3 89284 Pfaffenhofen a. d. Roth  Deutschland |
|  |
| Author:  Inov8tive Inc. |

Change History

|  |  |  |
| --- | --- | --- |
| Doc.-Version | Description of Modification | Date |
| A1 | Initial creation and completion of the use cases and the requirements | 01.04.24 |
| A2 | validation and approval of document | 02.04.24 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Release

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Name | Responsibility | Date | Signature |
| Creation | Maximilian Kraemer, Fabian Knapp-Rodriguez, Jan Gross |  | 01.02.24 | i.A. |
| Verification | Fabian Knapp-Rodriguez |  | 02.02.24 | i.A. |
| Approval | Jan Gross |  | 02.02.24 | i.A. |
| Release | Fabian Knapp Rodriguez |  | 02.04.24 | i.A. |

Table of Contents

Change History 2

Release 2

Table of Contents 3

List of Tables 4

List of Figures 4

1 General 5

1.1 Abbreviations 5

1.2 Terminology 5

1.3 Referenced Documents 5

2 Introduction 6

2.1 System Overview 6

2.2 Use Cases 9

3 Requirements 12

3.1 Functional Requirements 12

3.2 Non-Functional Requirements 13

3.2.1 Environmental Conditions 13

3.2.2 Quality 13

3.2.3 Computer Resources 14

List of Tables

[Table 1: Abbreviations 5](#_Toc162989139)

[Table 2: Terminology 5](#_Toc162989140)

[Table 3: Referenced Document 5](#_Toc162989141)

List of Figures

[Figure 1: Schematic of TackLine and StartLine [1] 6](#_Toc162989142)

[Figure 2: ZumoRobot and periphery [1] 7](#_Toc162989143)

[Figure 3: Use Case Diagram System Robot 7](#_Toc162989144)

[Figure 4: Use Case Diagram System Software 8](#_Toc162989145)

# General

## Abbreviations

|  |  |
| --- | --- |
| Abbreviation | Description |
| OLED | Organic Light Emitting Diode |

Table 1: Abbreviations

## Terminology

|  |  |
| --- | --- |
| Term | Description |
| *Robot* | As shown in fig. 2 Robot containing the programmable microcontroller and the peripherals |
| *ButtonA* | Pushbutton on the Robot as shown in fig. 2 |
| *ButtonB* |  |
| *ButtonC* |  |
| *PowerSwitch* |  |
| *Buzzer* | Buzzer on the Robot as shown in fig. 2, audio emitting device at constant volume but variable frequency |
| *OLED* |  |
| *LineSensorArray* | As shown in fig. 2, five line-sensors, used for distinguishing between dark and light surfaces, by measuring the reflection of the light emitted by the corresponding LED. |
| *Trackline* | As shown in fig. 1, a 1.5cm wide black line on a white background which functions as the track |
| *Startline* | As shown in fig. 1, two 5cm long 1.5cm wide black lines on a white background, orthogonal to the track line with a distance of 3.5cm to the *Trackline* |
| *Alarm* | 333ms beep, 333ms pause,333ms beep at a frequency of about 880Hz |
| *Beep* | 1s beep at a frequency of about 440Hz |

Table 2: Terminology

## Referenced Documents

|  |  |  |
| --- | --- | --- |
| Reference | Document-Identification | Description |
| [1] | 11001\_0099\_0088\_RD-Product-Specification.pdf | General specifications and description of application and the system-environment |
|  | Robot doku |  |

Table 3: Referenced Document

# Introduction

## System Overview

The Line-Follower Software for the Pololu Zumo32U4 has been developed to support line-follower competitions by enabling robots to autonomously navigate a predefined track. This detailed system overview provides a comprehensive insight into the functionalities, requirements, and applications of the software.

Figure 1 illustrates a schematic representation of the start line and the end line, both of which appear identical. Between them, the TrackLine can traverse various routes.

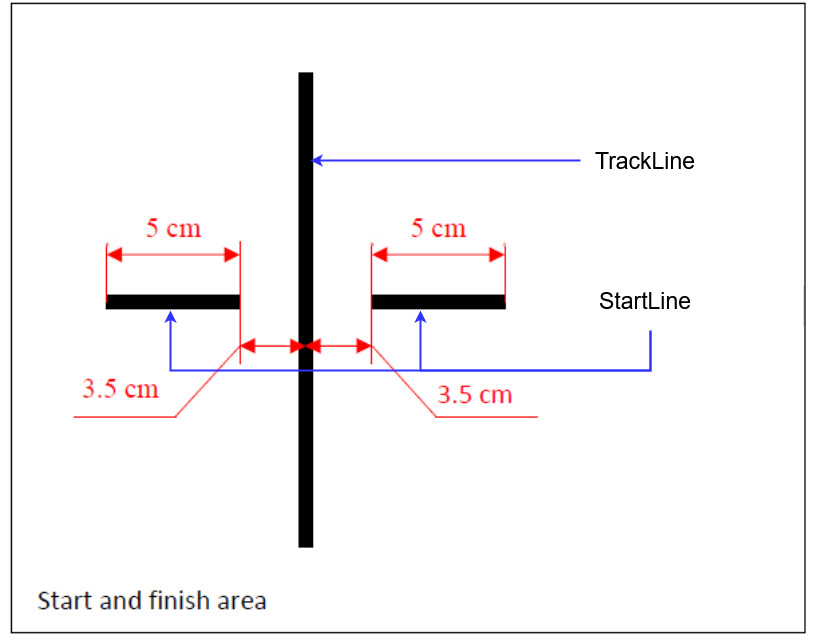


Figure 1: Schematic of TackLine and StartLine [1]

Figure 2 depicts the Pololu Zumo robot in a top-down view. It features a chain drive, an IR sensor system within the sensor array, a dual-motor drive system with encoders, an OLED display, a buzzer, an Arduino ATmega32U4 microcontroller, a USB programming interface, Pushbutton A, and a three-axis accelerometer.

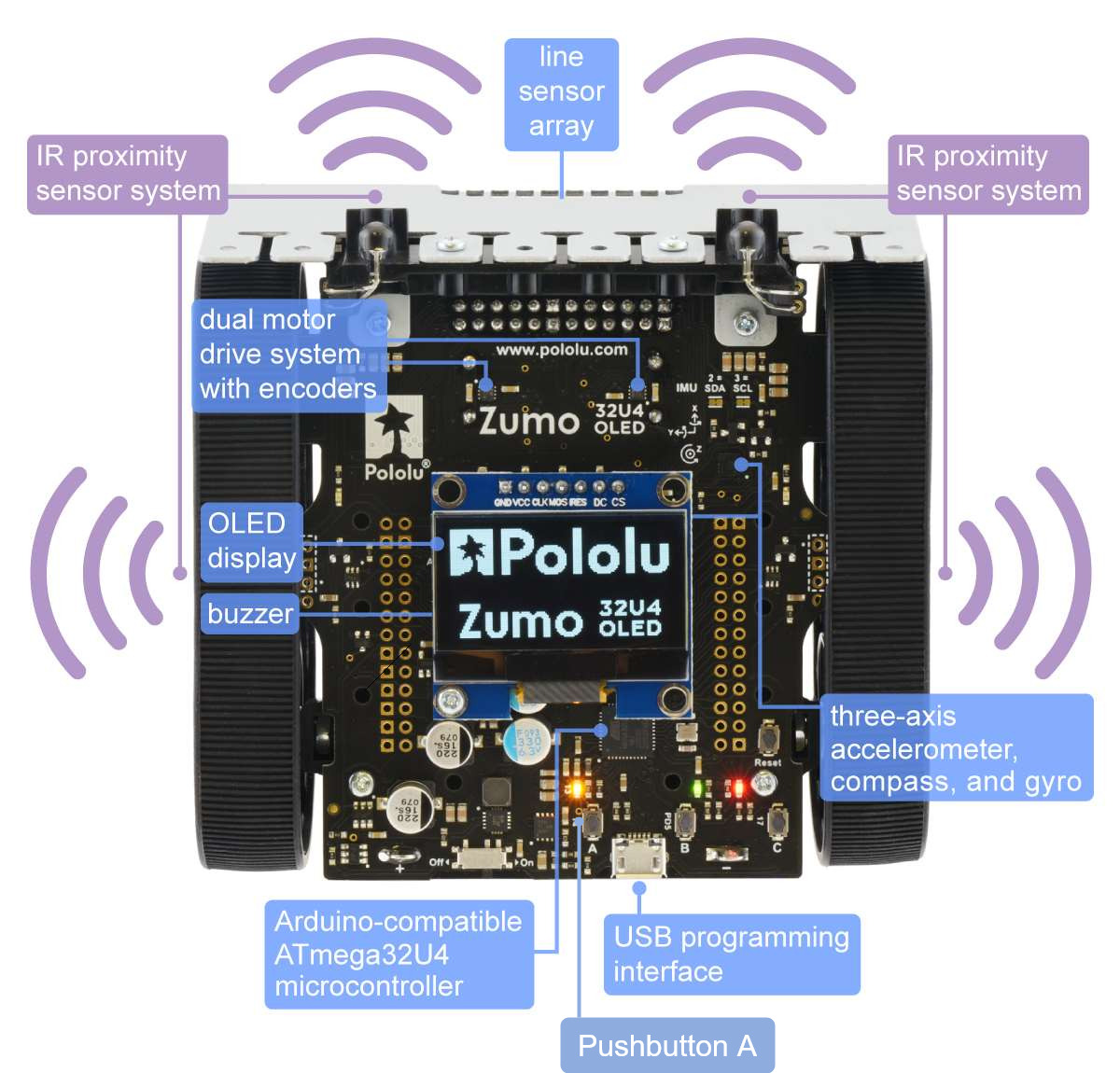


Figure 2: ZumoRobot and periphery [1]

Figure 3 showcases the Use Case Diagram of the system, where four external actors are identified: Start/Endline, Application, User, and Power Supply.

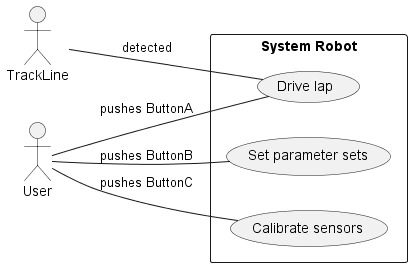


Figure 3: Use Case Diagram System Robot

Figure 4 illustrates the Use Case Diagram of the system software. On the left side, there are three inputs: Line Sensor Array, Power Supply, and Pushbutton. On the right side, three outputs are displayed: Buzzer, OLED Display, and Dual Motor Drive System.

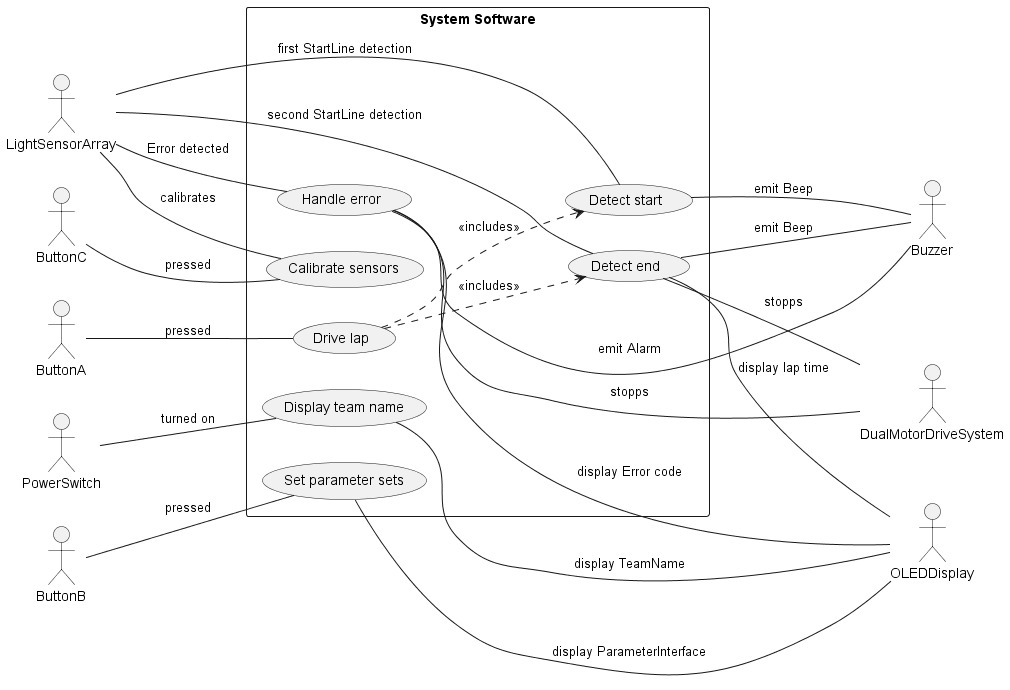


Figure 4: Use Case Diagram System Software

## Use Cases

### *Use Cases – System Robot*

|  |  |
| --- | --- |
| Identifier | R\_L1 |
| Name | Start lap |
| Brief description | 3s after the *ButtonA* is pressed by the user, the *Robot* starts driving. |
| Preconditions | The *Robot’s* sensors are calibrated and a parameter set selected. |
| Postconditions | The *Robot* is driving. |
| Failure scenarios | - |
| Actors | User |
| Trigger | User interaction |
| Standard workflow | 1. The user presses *ButtonA*. 2. Wait 3s. 3. The *Robot* starts driving. |
| Alternative workflow | - |

|  |  |
| --- | --- |
| Identifier | R\_L2 |
| Name | Drive lap |
| Brief description | After the *Robot* starts driving, it follows the detected *Trackline.* |
| Preconditions | The *Robot* is driving. |
| Postconditions | The *Robot* detected *Trackline*? |
| Failure scenarios | The *Trackline* not detected |
| Actors | *Trackline* |
| Trigger | ? |
| Standard workflow | 1. The *Robot* detects the *Trackline.* 2. The *Robot* followsthe *Trackline*. |
| Alternative workflow | 1. The *Robot* does not detect *Trackline.* 2. The *Robot* detects an error.? |

|  |  |
| --- | --- |
| Identifier | R\_C1 |
| Name | Calibrate sensors |
| Brief description | After the *Robot* is powered on or the *ButtonC* is pressed by the user, the *Robot* calibrates its sensors. |
| Preconditions | *PowerSwitch* is on. |
| Postconditions | Sensors are calibrated. |
| Failure scenarios | - |
| Actors | User |
| Trigger | User interaction |
| Standard workflow | 1. User turns *PowerSwitch* on or presses *ButtonC*. 2. Conditions for calibration are fulfilled. 3. The *Robot* calibrates its sensors. |
| Alternative workflow | - |

|  |  |
| --- | --- |
| Identifier | R\_P1 |
| Name | Set parameter sets |
| Brief description | After *ButtonB* is pressed by the user, the user can select a parameter set. |
| Preconditions | *PowerSwitch* is on. |
| Postconditions | Parameter set selected. |
| Failure scenarios | - |
| Actors | User |
| Trigger | User interaction |
| Standard workflow | 1. *ButtonB* is pressed by the user. 2. The User selects a parameter set. |
| Alternative workflow | - |

### *Use Cases – System Software*

|  |  |
| --- | --- |
| Identifier | S\_L1 |
| Name | Detect start |
| Brief description | If the *LineSensorArray* detects the *Startline* for the first time, the time measurement starts and a *Beep* is emitted. |
| Preconditions | The *Robot* is driving. |
| Postconditions | The time measurement is running. |
| Failure scenarios | *Startline* not detected. |
| Actors | *LineSensorArray, Buzzer* |
| Trigger | *Startline* |
| Standard workflow | 1. The *LineSensorArray* detects the *Startline* for the first time. 2. The time measurement starts. 3. The *Buzzer* emits a *Beep*. |
| Alternative workflow | 1. The *LineSensorArray* cannot detect the *Startline* for more than 8s. 2. The *Robot* detects an error. |

|  |  |
| --- | --- |
| Identifier | S\_L2 |
| Name | Detect end |
| Brief description | If the *LineSensorArray* detects the *Startline* for the second time, the time measurement stops, a *Beep* is emitted and the *Robot* stops. |
| Preconditions | The *Robot* is driving. |
| Postconditions | The *Robot* is not driving. |
| Failure scenarios | - |
| Actors | *LineSensorArray, Buzzer, OLED* |
| Trigger | *Startline* |
| Standard workflow | 1. The *LineSensorArray* detects the *Startline* for the second time. 2. The time measurement stops. 3. The *Robot* stops. 4. The *Buzzer* emits a *Beep*. 5. The time measurement’s time is displayed on the OLED. |
| Alternative workflow | - |

|  |  |
| --- | --- |
| Identifier | R\_L3 |
| Name | Start lap |
| Brief description | 3s after the *ButtonA* is pressed by the user, the *Robot* starts driving. |
| Preconditions | The *Robot’s* sensors are calibrated and a parameter set selected. |
| Postconditions | The *Robot* is driving. |
| Failure scenarios | - |
| Actors | *ButtonA* |
| Trigger | User interaction |
| Standard workflow | 1. The user presses *ButtonA*. 2. The *Robot’s* sensors are calibrated and a parameter set selected. 3. Wait 3s. 4. The *Robot* starts driving. |
| Alternative workflow | 1. - |

|  |  |
| --- | --- |
| Identifier | R\_L4 |
| Name | Drive lap |
| Brief description | After the *Robot* starts driving, it follows the detected *Trackline.* |
| Preconditions | The *Robot* is driving. |
| Postconditions | The *Robot* detected *Trackline*? |
| Failure scenarios | The *Trackline* not detected |
| Actors | *LineSensorArray* |
| Trigger | ? |
| Standard workflow | 1. The *Robot* detects the *Trackline.* 2. The *Robot* follows the *Trackline*. |
| Alternative workflow | 1. The *Robot* does not detect *Trackline.* 2. The *Robot* detects an error.? |

|  |  |
| --- | --- |
| Identifier | S\_E1 |
| Name | Handle error |
| Brief description | If the robot detects an error, it will notify with an *Alarm* and display the error reason on the *OLED*. Furthermore, the *Robot* will stop. |
| Preconditions | Robot detects error. |
| Postconditions | The Robot is not driving. |
| Failure scenarios | - |
| Actors | *LineSensorArray, Buzzer, OLED* |
| Trigger | Error |
| Standard workflow | 1. The *Robot* or the *LineSensorArray* detect an error. 2. The *Robot* stops driving. 3. The *Buzzer* emits an alarm. 4. The error reason is displayed on the *OLED*. |
| Alternative workflow | - |

|  |  |
| --- | --- |
| Identifier | S\_C1 |
| Name | Calibrate sensors |
| Brief description | After the *Robot* is powered on or the *ButtonC* is pressed by the user, the *Robot* calibrates its sensors. |
| Preconditions | *PowerSwitch* is on. |
| Postconditions | Sensors are calibrated. |
| Failure scenarios | - |
| Actors | *ButtonC*, *PowerSwitch*, *OLED* |
| Trigger | User interaction |
| Standard workflow | 1. User turns *PowerSwitch* on or presses *ButtonC*. 2. Conditions for calibration are fulfilled. 3. The *Robot* calibrates its sensors. 4. Success message displayed on the *OLED*. |
| Alternative workflow | - |

|  |  |
| --- | --- |
| Identifier | S\_P1 |
| Name | Set parameter sets |
| Brief description | After *ButtonB* is pressed by the user, the user can select a parameter set. |
| Preconditions | *PowerSwitch* is on. |
| Postconditions | Parameter set selected. |
| Failure scenarios | - |
| Actors | *ButtonB, OLED* |
| Trigger | User interaction |
| Standard workflow | 1. *ButtonB* is pressed by the user. 2. The set interface is displayed on the *OLED*. 3. The User selects a parameter set. |
| Alternative workflow | - |

|  |  |
| --- | --- |
| Identifier | S\_D1 |
| Name | Display team name |
| Brief description | After the *PowerSwitch* is turned on by a user, the team’s name is displayed on the *OLED*. |
| Preconditions | *PowerSwitch* is on. |
| Postconditions | ? |
| Failure scenarios | - |
| Actors | *PowerSwitch, OLED* |
| Trigger | User interaction |
| Standard workflow | 1. User turns *PowerSwitch* on. 2. The team’s name is displayed for 3s on the *OLED*. |
| Alternative workflow | - |

# Requirements

## Functional Requirements

|  |  |
| --- | --- |
| PO\_0 | Power on |
|  | If *PowerSwitch* is turned on by a user, the *Robot* shall execute all of the following actions in the given order:  • display the teams name one the *OLED*,  • the *Robot* enters calibration state and calibrates its sensors. |

|  |  |
| --- | --- |
| ST\_0 | Start drive mode |
|  | *If ButtonA* is pressed by a user, the *Robot* shall start driving after 3s. |

|  |  |
| --- | --- |
| C\_0 | Calibration mode |
|  | If *ButtonC* is pressed by a user, the *Robot* shall enter calibration state and calibrate its sensors. |

|  |  |
| --- | --- |
| C\_1 | Parameter set mode |
|  | If *ButtonB* is pressed by a user, user, the *Robot* shall execute all of the following actions in the given order:  • display the set interface on the *OLED*,  • apply the set selected by the user. |

|  |  |
| --- | --- |
| L\_0 | Start line detection |
|  | If the *Robot* detects the *Startline* for the first time, the *Robot* shall execute all of the following actions in the given order:  • the time measurement for the lap starts,  • the *Buzzer* plays a *Beep.* |

|  |  |
| --- | --- |
| L\_1 | Track line detection |
|  | If the *Robot* is driving, it shall follow the *Trackline.* |

|  |  |
| --- | --- |
| L\_2 | Track line redetection |
|  | If the *Robot* cannot detect the *Trackline*, it shall try to redetect *Trackline.* |

|  |  |
| --- | --- |
| L\_3 | End line detection |
|  | If the *Robot* detects the *Startline* for the second time, the *Robot* shall execute all of the following actions in the given order:  • the time measurement for the lap stops,  • the *Robot* stops driving,  • the *Buzzer* plays a *Beep,*  • display the measured lap time on the *OLED*. |

|  |  |
| --- | --- |
| ERR\_0 | Error handle |
|  | If the *Robot* detects an error, the *Robot* shall execute all of the following actions in the given order:  • the *Robot* stops driving,  • play an *Alarm*,  • display the error reason on the *OLED*. |

|  |  |
| --- | --- |
| ERR\_1 | Time out error |
|  | If the time measurement exceeds 20s, the *Robot* shall detect an error. |

|  |  |
| --- | --- |
| ERR\_2 | Track lost error |
|  | If the *Robot* cannot detect the *Trackline* for more than 5s, the *Robot* shall detect an error. |

|  |  |
| --- | --- |
| ERR\_3 | No start line error |
|  | If the *Robot* cannot detect the *Startline* for more than 8s after initially starting drive mode, the *Robot* shall detect an error. |

## Non-Functional Requirements

### Environmental Conditions

|  |  |
| --- | --- |
| NF\_1 | Light conditions |
|  | The software must work with normal daylight & workplace light conditions, which is equivalent to 20 to 700 lux. |

### Quality

|  |  |
| --- | --- |
| NF\_2 | Automation requirement |
|  | The *Robot* shall drive autonomously and it is not allowed to control it remotely  via cable or wireless. |

|  |  |
| --- | --- |
| NF\_3 | Battery requirements |
|  | The software shall work independently of the charge status of the batteries, unless the  charge status is below 50%. |

|  |  |
| --- | --- |
| NF\_4 | Modularity requirements |
|  | The software must be designed in a way that allows to replace the hardware by e.g. a  different robot or a hardware simulation. Unavailability of hardware features does not need  to be considered. |

### Computer Resources

|  |  |
| --- | --- |
| NF\_5 | Software requirements |
|  | The software shall be implemented in C. |

|  |  |
| --- | --- |
| NF\_6 | Flash memory requirements |
|  | The flash memory usage shall be lower than or equal to 80%. |