HOCH
SCHULE
OFFEN
BURG

Specifications

Line-Follower software for Pololu Zumo32U4

Doc.-Number:	Specifications_V1
Doc.-Version:	V1
Customer:	Martin Dembinsky, MSc Niclas Heitz, MSc Labor Software Engineering Hochschule Offenburg
Author/ Group:	Die Fantastischen Vier
Abgabedatum:	02.04.2024

1 Change History

Doc.-Version	Description of Modification	Date
V1	Initial revision	02.04.2024

2 Release

	Name	Responsibility	Date	Signature
Creation	Die Fantastischen Vier		28.03.2024	Fanta4Group
Verification				
Approval				
Release				

Table of Contents

List of Tables

List of Figures

3 General

3.1 Scope of Document

This document serves as a specification sheet for the customer order. It contains functional and non-functional requirements for the system. It also contains two use case diagrams of the robot and software system with detailed descriptions. This document comprises a total of 27 pages.

3.2 Abbreviations

Abbreviation	Description
OLED	Organic light-emitting diode. In this document OLED refers as a short form for the OLED-display.

tab. 1: Abbreviations

3.3 Terminology

Term	Description
Start line	The start of the *track* is crossed with a 90° crossbar
Stop line	The end of the *track* is crossed with a 90° crossbar. The system distinguish between Start- and Stop lines depending on the program state. If the race has started it shall take a Start- line as an Stop-line
track	The *track* is a black line which the robot is going to follow.
Standard *track*	Is a pre-defined *track*, given by the customer.
alarm tone	A short sound to notify in an *error state*. The sound shall follow the following schematic: 1/3 s sound 1/3 s silence 1/3 s sound at a frequency of 440 Hz. The sound shall be played with the maximum amplitude.
notification sound	A short sound with a duration of 1 s at a frequency of 440 Hz. The sound shall be played with the maximum amplitude.
booting cycle	After the robot is turned on or the reset button is pressed, the system starts the software.
error state	The status the system jumps, after an error occurs. The system follows a strict procedure which is explained in the requirements.
race modes	A selectable status of the system to change the behavior of the robot on the *track*.

safety mode	The System drives slower than in the other modes to ensure the system loses the *track* less likely.
highspeed mode	The system drives faster than in the other modes.
balanced mode	This mode is compromise between the safety and *highspeed mode*.

tab. 2: Terminology

3.4 Referenced Documents

Reference	Document-Identification	Description
[1]	11001_0099_0088_RD	Product specification for the Line-Follower Software for Pololu Zumo32U4

tab. 3: Referenced Documents

3.5 Applicable Standards

Reference	Document-Identification	Description
[1]	N/A	N/A

tab. 4: Applicable Standards

4 Introduction

4.1 System Overview

The overall system of the Pololu Zumo32u4 has an IR proximity sensor system, a line sensor array, a dual motor drive system with encider as well as an OLED display and buz-

zer also onboard a three-axis accelerometer, compass, and gyro.

4.2 Interface Overview

The board of the Arduino compatible ATmega32u4 microcontroller has a USB programming interface.

4.3 Scenarios

The Line-Follower software runs on the Zum32U4 robot platform from Pololu. The robot drives a defined *track* as fast as possible from the start area to the finish area. The *track* is marked with a black line on white background. The team with the robot that drove the fastest complete round wins.

5 Requirements

5.1 Functional Requirements

1. If the system is turned on, the system shall display the teams name for 2 s.
2. If the user hasn't started the calibration process with the push button "C", the system shall inform the user to calibrate the sensor.
3. The system shall have different sets of pre-defined parameters to change between different the following *race mode*s:
   - *safety mode*
   - *highspeed mode*
   - *balanced mode*
4. If the push button "B" on the robot is pressed, the system shall change between the different *race mode*s.
5. If the user hasn't selected any *race mode* the system shall use the *balanced mode*.

6. If the push button “C” on the robot is pressed, the system shall start the calibration process of the line sensor.
7. If the calibration process has started the system shall define the color underneath the line sensor as the color black.
8. If the calibration process is running, the system shall disable all user inputs.
9. The system shall use his line sensors to detect a black line.
10. If the push button “A” on the system is pressed, the system shall start driving after 3 s.
11. f the Line sensor isn’t calibrated, the system shall not allow the user to start the race with the push button “A”.
12. The system shall follow the detected *track*.
13. The system shall drives autonomously.
14. The system must begin its run on the *track* and at least 1 cm from the *start line*.
15. If the *track* in front of the robot ends, the system shall stop all movements and execute all of the following actions until one of them is true.
    - The system shall turn 90° right and re-detect the *track* every 10°.
    - The system shall turn 180° and re-detect the *track* every 10°.
    - The system shall turn to 90° right and drive 10 cm straight forward while re-detecting the *track*.
    - The system shall jump in an *error state*.
16. If the *start line* is detected, the system shall execute all of the following actions in the given order:
    - start the time measurement for the lap.
    - notify the user with an *notification sound*.
17. If the system detects an error, the system shall jump in an *error state* and execute all of the following actions in the given order:
    - Stop all movements,
    - Notify with an *alarm tone*.
    - Display the error reason on the OLED.
18. If the system detects the *stop line*, the system shall execute all of the following actions in the given order:
    - Stop all movements
    - Stop time measurement
    - Notify with a *notification sound*.

- Display the measured time on the OLED

19. The system shall jump in an *error state*, if either one of the following instances applies:
    - The system can't re-detect the *track* within 5 s.
    - The system can't complete one round on the standard *track* in maximum 20 s.
    - The system couldn't load the selected parameters correctly.
    - The system couldn't finish the calibration process correctly.

5.2 Non-Functional Requirements

5.2.1 Safety & Security

none

5.2.2 Data

none

5.2.3 Environmental Conditions

1. The software shall work with normal daylight & workplace light conditions.
2. The system shall work on a playfield with a white background.
3. The *track* shall not cross itself
4. The *track* shall not contain any gaps larger than 10 cm.
5. Every gap shall continue with an opening angle of 30°.
6. The start and *stop line* are marked with 90° crossbeam.
7. The crossbeam are in a distance of 3.5 cm to the center line.
8. The crossbeam shall have a length of 5 cm.
9. The start and the stop are minimum 30 cm away from any gap in the *track*.
10. The minimum distance between playfield border and the *track* is 15 cm.
11. The minimum curve radius on a *track* is 10 cm.

5.2.4 Quality

1. The software shall work independently of the charge status of the batteries, unless the charge status is below 50%.
2. The flash memory usage shall never exceed equal to 80%.

5.2.5 Computer Resources

1. The software shall run on the Zum32U4 robot platform from Pololu.
2. The software shall work 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.
3. The teams shall use the programming language C.

5.2.6 Design Constraints

none

5.2.7 Product Documentation

1. The teams shall write their documentations in English.
2. The teams shall use English for commit messages of version control system interactions..
3. The teams shall write the commit messages of version control system interactions descriptive.

5.2.8 Production

none

5.2.9 Logistic

none

5.2.10 Commercial Requirements

none

5.2.11 Further Requirements

1. The team shall not make any modifications on the hardware during executing the competition, except changing the batteries or small repairs.
2. Each team is able to test the robot on the *track* before the competition starts. During this phase the teams are allowed to change the software as well.
3. Each team shall have 3 attempts on the *track*. It counts the fastest complete round.
4. The system shall finish the race as fast as possible.

6 Interface

6.1 External Interface

not further specified

7 Document Management

7.1 Document Creation

All documents must be written in English.

8 Appendix

8.1 Use Case Diagrams

8.1.1 System Robot

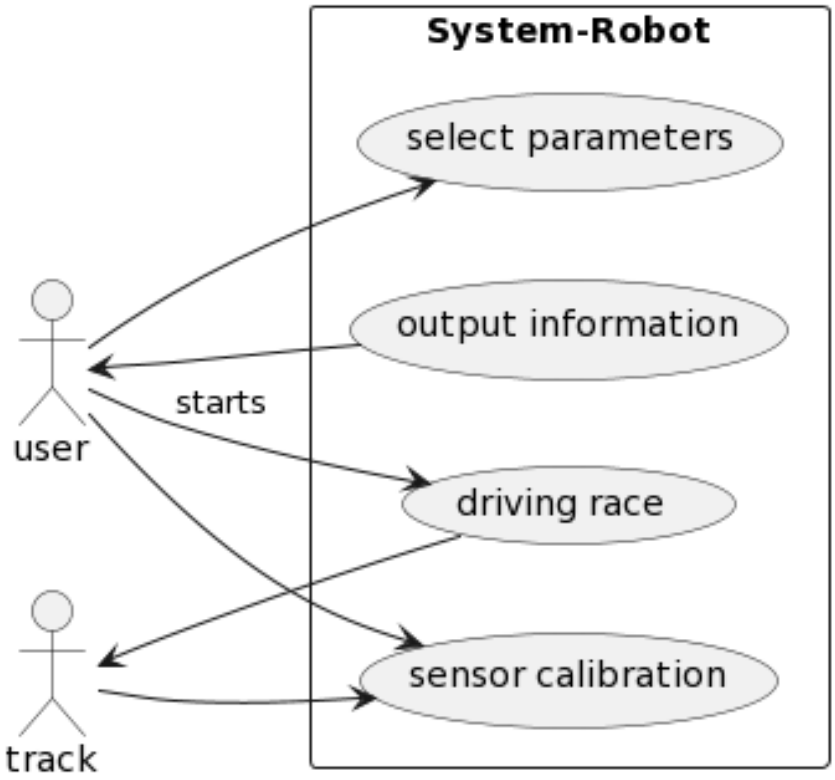

fig. 1: Use case diagram System Robot

Identifier	**R01**
Name	**select parameters**
Brief description	The users selects pre-defined parameters sets to change the behavior of the robot on the *track*.
Preconditions	The system is not in the use case “driving race“.
Postconditions	The desired *driving mode* is set.
Failure scenarios	The parameters couldn’t be loaded correctly.
Actors	User
Trigger	User
Standard workflow	1. User pushes the button B. 2. The *driving mode* is switched between *highspeed mode*, *safety mode* or *balanced mode*.
Alternative workflow	2a. The robot notifies about an error.

Identifier	**R02**
Name	**output information**
Brief description	The robot supplies the user with information through the display and the buzzer.
Preconditions	The system is not in the use case “driving race“ R03.
Postconditions	The OLED displays the refreshed parameters or the buzzer beeps.
Failure scenarios	The user couldn’t be provided with all information.
Actors	User
Trigger	The robot was powered on./ The robot has passed the *stop line*.
Standard workflow	1. The robot is powered on. 2. The display shows the team name and the selected *driving mode*. 3. The robot drives the race. 4. The robot finishes the race. 5. The display shows the race time.
Alternative workflow	3a. The robot gets into an error case. 4a. The error case is displayed and the buzzer beeps.

Identifier	**R03**
Name	**driving race**
Brief description	The user gives the command to start the race. The robot starts driving and follows the line.
Preconditions	Line sensor is calibrated, parameters are selected
Postconditions	Robot has passed the *stop line*.
Failure scenarios	The robot loses the *track*
Actors	User, *track*
Trigger	User
Standard workflow	1. User push the button A 2. Robot checks for *start line* 3. Robot detects the *start line* 4. Robot detects the *track* 5. Robot follows the *track* 6. Robot checks for end line 7. Robot stops driving
Alternative workflow	5a. The robot loses the *track*. 6a. The robot redetects the *track*. 7a. back to 6. 6b. The robot can't find the *track* within 5s. 7b. The robot stops driving. 8b. The robot notifies the user.

Identifier	**R04**
Name	**sensor calibration**
Brief description	The users pushes the button C to start the calibration process.
Preconditions	The system is not in the use case “driving race“.
Postconditions	The line sensor is calibrated.
Failure scenarios	The calibration wasn’t successful.
Actors	User, *track*
Trigger	User
Standard workflow	1. User lays the robot on the *track*. 2. User pushes the button C. 3. The robot starts the calibration process. 4. Robot notifies after the process is done.
Alternative workflow	4a. The robot notifies that the calibration wasn’t successful. 5a. The user restarts the calibration process.

8.1.2 System Software

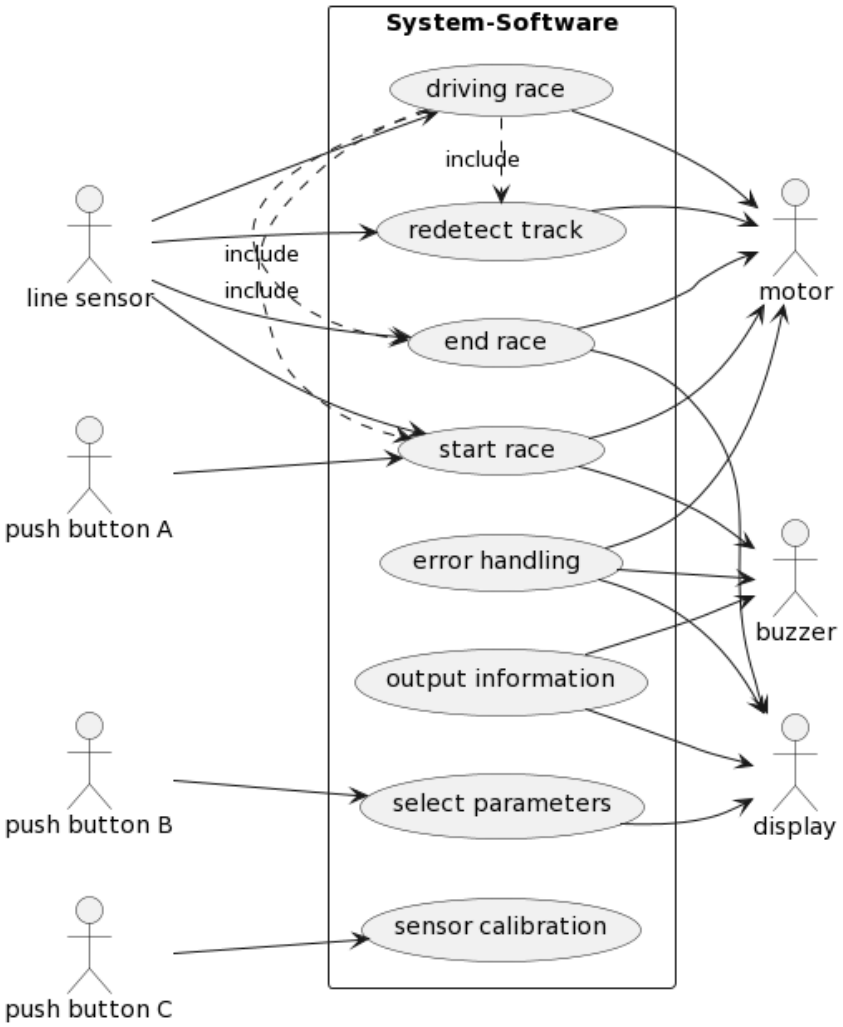

fig. 2: Use case diagram System Software

Identifier	**S01**
Name	**sensor calibration**
Brief description	By pushing the button C the calibration process is started.
Preconditions	The System is powered on.
Postconditions	The line sensor is calibrated to be able to detect the *track*.
Failure scenarios	The sensor cannot be calibrated.
Actors	push button C
Trigger	The push button C is pressed.
Standard workflow	1. The push button C is pressed. 2. The line sensor is calibrated.
Alternative workflow	2a. The calibration process fails.

Identifier	**S02**
Name	**select parameters**
Brief description	With push button B the software parameters can be switched between the parameter sets *highspeed mode, safety mode* or *balanced mode*.
Preconditions	The Software isn't in the use case "driving race".
Postconditions	The desired parameter set is selected.
Failure scenarios	There is no possibility to switch between the parameter sets.
Actors	Push button B, display
Trigger	Push button B is pressed.
Standard workflow	1. Push button B is pressed. 2. The system loads parameters. 3. The *race mode* is showed on the OLED.
Alternative workflow	2a. The system fails loading parameters. 3a. The *race mode* isn´t showed on the OLED.

Identifier	**S03**
Name	**output information**
Brief description	The software outputs information through the display and the buzzer.
Preconditions	The software isn't in the use case "driving race".
Postconditions	The software outputs information through the display or the buzzer.
Failure scenarios	The display doesn't show information or the buzzer doesn't output a signal.
Actors	display, buzzer
Trigger	The system is powered on. The system is in an error case. The system finished use case "driving race" S04.
Standard workflow	1. The software is powered on. 2. The display shows the team name and the selected *driving mode*. 3. The software is in use case "driving race" S04. 4. The software is in use case "end race" S08. 5. The display shows the race time.
Alternative workflow	2a. The display doesn't show the team name and the selected *driving mode* correctly. 5a. The display fails showing the race time.

Identifier	**S04**
Name	**driving race**
Brief description	The system controls the robot to follow the line in the selected *driving mode*.
Preconditions	use case “start race“ S07 was ended successfully.
Postconditions	The system executes use case “end race“ S08.
Failure scenarios	The software fails controlling the robot following the *track*, so the robot leaves the *track*.
Actors	line sensor, motor
Trigger	“start race“ S07 is finished
Standard workflow	1. “start race“ S07 is finished 2. The line sensor detects the *track*. 3. The software controls the motor. 4. The software is in use case “end race“ S08.
Alternative workflow	2a. The line sensor doesn’t detect the *track*. 3a. The software is in use case “error handling“ S05.

Identifier	**S05**
Name	**error handling**
Brief description	If an error occurs the system must handle it.
Preconditions	The system must detect an error.
Postconditions	The system stops the motor. The system outputs that an error occured through the display and the buzzer.
Failure scenarios	The system fails detecting the error.
Actors	motor, buzzer, display
Trigger	The system detects an error.
Standard workflow	1. The system detects an error. 2. The system stops the motor. 3. The system outputs that an error occured. 4. The system will beep for 0,33s, pause for 0,33s, then beep again for 0,33s. 5. The system will be restarted.
Alternative workflow	2a. The system fails stopping the motor. 3a. The system fails displaying that an error occured. 4a. The system fails putting out the beep tone.

Identifier	**S06**
Name	**redetect *track***
Brief description	In case there is a gap in the line or the line isn't straight forward, the system must redetect the *track*.
Preconditions	The line sensor can't detect the *track*.
Postconditions	The line sensor redetects the *track*. The software is in the use case "driving race" S04 again.
Failure scenarios	The line sensor cannot redetect the *track*. The software is in the use case "error handling"
Actors	line sensor, motor
Trigger	The line sensor can't detect the *track*.
Standard workflow	1. The line sensor doesn't detect the *track*. 2. The software controls the motor to turn the robot left and right to redetect the line. 3. The software redetects the line. Back to "driving race" S04.
Alternative workflow	3a. The software cannot redetect the line within 5s. 4a. The software stops the motor. 5a. The software is in use case "error handling" S05.

Identifier	**S07**
Name	**start race**
Brief description	The system controls the robot to start the race.
Preconditions	The system is powered on. The line sensor is calibrated. The *driving mode* is selected.
Postconditions	The system executes use case “driving race“ S04.
Failure scenarios	The software fails detecting the *start line*.
Actors	push button A, line sensor, motor
Trigger	push button A
Standard workflow	1. The push button A is pressed. 2. The system waits for 3s. 3. The system controls the motor to drive forward. 4. The system detects the *start line*. 5. The system outputs a short beep through the buzzer for 1s. 6. The system starts measuring the time. 7. The system executes use case “driving race“ S04.
Alternative workflow	2a. The system fails the time measurement. 4a. The system executes use case “error handling“ S05. 4b. The system fails detecting the *start line*. 5b. The system executes use case “error handling“ S05. 5c. The system fails putting out the beep tone.

Identifier	**S08**
Name	**end race**
Brief description	The system controls the robot to end the race.
Preconditions	The system is powered on. The system executes use case “driving race“ S04.
Postconditions	The system displays the measured time.
Failure scenarios	The software fails detecting the *stop line*.
Actors	line sensor, motor, display
Trigger	The *stop line* is detected.
Standard workflow	1. The system detects the *stop line*. 2. The system controls the motor to stop. 3. The system stops measuring the time. 4. The system outputs the measured time through the display. 5. The system outputs a short beep through the buzzer for 1s.
Alternative workflow	1a. The system fails detecting the *stop line*. 2a. The system fails controlling the motor to stop. 3a. The system fails stopping measuring the time. 4a. The system fails putting out the measured time through the display. 5a. The system fails putting out a short beep through the buzzer.