Project informations

Block: Robot SW
Person responsible for this block:
eng. Bazan Michal

Table of Contents

Change sheet

Date	Description
02/22/2024	Template cration, basic info, glossary, requirements analysis.
02/23/2024	SW architecture.
03/18/2024	Change in interfaces, marked obsolete sections to be changed.
03/19/2024	Created table „implementation status".
03/22/2024	Updated status of implementation, glossary and block diagram.
03/25/2024	Updated status of implementation.
04/05/2024	Updated status of implementation.
04/06/2024	Updated status of implementation.
04/09/2024	Updated status of implementation.
04/11/2024	Updated status of implementation.
04/12/2024	Updated status of implementation.

Glossary

UT – unit test

SQT – software qualification test

Analysis of the technical state of art, review of knowledge analysis

Creating a riding robot with sensors involves integrating various technologies and components to achieve stable locomotion and effective sensing capabilities. Here's an analysis of the technical state of the art and a review of knowledge related to this endeavor:

a) **Hardware Components**:

- **Actuators**: Utilizing advanced actuators like brushless DC motors or servo motors can provide precise control over movement.
- **Sensors**: Incorporating a variety of sensors such as gyroscopes, accelerometers, encoders, and proximity sensors enables the robot to gather data about its surroundings and maintain balance.
- **Microcontrollers**: Advanced microcontrollers like Arduino or Raspberry Pi offer the computational power needed for sensor data processing and motor control.
- **Power Supply**: Efficient power management systems, such as lithium-ion batteries or fuel cells, are crucial for prolonged operation.

b) **Locomotion Mechanisms**:

- **Wheeled**: Wheeled locomotion is common due to its simplicity and stability. Differential drive or omnidirectional wheels can provide maneuverability.
- **Legged**: Legged locomotion offers advantages in traversing uneven terrain but requires more complex control algorithms and mechanisms.
- **Hybrid**: Some designs combine wheels with legs or tracks to balance stability and versatility.

c) **Control Systems**:

- **PID Control**: Proportional-Integral-Derivative control loops are commonly used for maintaining balance and controlling movement based on sensor feedback.
- **Machine Learning**: Advanced control strategies involving machine learning techniques like reinforcement learning or neural networks can optimize control in dynamic environments.

d) **Sensor Integration and Fusion**:

- **Sensor Fusion**: Combining data from multiple sensors enhances the robot's perception accuracy and reliability.
- **Environment Mapping**: Techniques like Simultaneous Localization and Mapping (SLAM) enable the robot to create maps of its surroundings for navigation.
- **Obstacle Detection**: Sensors like LiDAR, ultrasonic sensors, or cameras aid in detecting obstacles and planning collision-free paths.

e) **Safety and Redundancy**:

- **Fault Tolerance**: Implementing redundancy in critical components and fail-safe mechanisms ensures the robot can handle unforeseen situations.

- **Emergency Stop**: Incorporating emergency stop mechanisms based on sensor inputs or manual intervention prevents accidents and damage to the robot or its surroundings.

f) **Wireless Communication**:

- **Remote Control**: Providing wireless control interfaces allows operators to monitor the robot's status and intervene if necessary.
- **Data Transmission**: Wireless communication protocols facilitate real-time data transmission between the robot and external devices for telemetry and control purposes.

g) **Human-Machine Interaction**:

- **User Interface**: Designing intuitive user interfaces for controlling and monitoring the robot simplifies operation for end-users.
- **Feedback Mechanisms**: Incorporating feedback mechanisms like haptic feedback or visual indicators enhances user experience and situational awareness.

Requirements regarding this project

In this chapter, we will look into the software requirements regarding robot operating system. Because system requirements were not provided, this project will skip SWE.1 part of the process.

Type of Verification: D - Design Review, A - Analysis, T - Test, N/A - not applicable

ID	OBJ_TEXT	Verification method
SWE_2_010	Software should read sensors periodicaly.	D T
SWE_2_020	Software should use WiFi interface to communicate with operating station.	T
SWE_2_030	Software should provide an interface to control the motors.	D T
SWE_2_040	Software should use a simple communication protocol to exchange data with operating station.	D T
SWE_2_050	Software should provide odometry.	D T

Requirements regarding system components

While developing the software for the robot, specific components are required to meet the functional needs outlined in the project specification. The following requirements detail necessary components for the software implementation:

Type of Verification: D - Design Review, A - Analysis, T - Test, N/A - not applicable

ID	PARENT_ID	OBJECT_TEXT	Veryfication method
SWE_3_010	SWE_2_010	Software should pool IMU and distance sensors every 50ms.	T
SWE_3_020	SWE_2_020 SWE_2_040	Software should be able to exchange data with operating station.	T
SWE_3_030	SWE_2_020 SWE_2_040	Software should be able to send sensor data to operating station on demand and periodicaly.	T
SWE_3_040	SWE_2_020 SWE_2_040	Software should use MQTT communication protocol and use JSON format.	D
SWE_3_050	SWE_2_030	Software should provide interface to control the motion of the motor (speed, direction).	D T
SWE_3_060	SWE_2_010 SWE_2_050	Software should provide simple odometry block.	D
SWE_3_070	SWE_2_010 SWE_2_050	Software should provide odometry block with slip correction.	D
SWE_3_080	SWE_2_010 SWE_2_050	Software should provide possibility to change active odometry block.	D
SWE_3_090	SWE_2_010 SWE_2_050	Software should use simple odometry as default block.	D T
SWE_3_100	SWE_2_010 SWE_2_050	Software should reset odometry after switching the active block.	T

System architecture

Based on requirements, this chapter focuses on software architecture of the system.
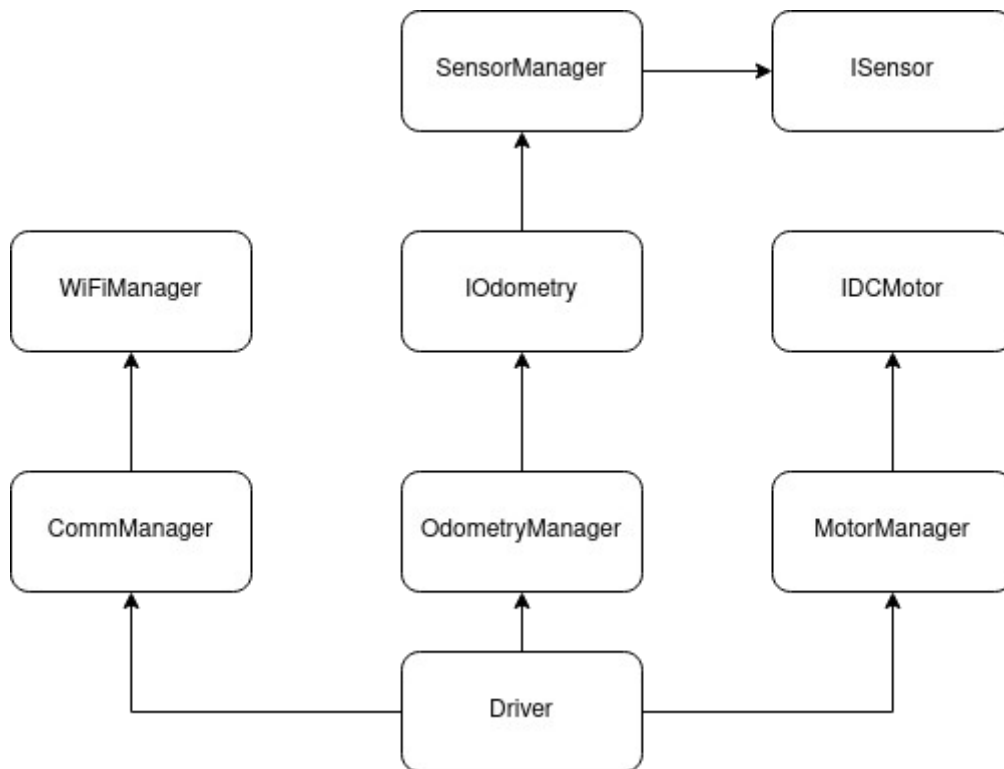

Figure 1: Software block diagram

Software detailed design

This chapter presents implementation status of the software and main components.

Implementation status

All UT can be found under robot/SW/robot_software/test directory.

Component	Brief	Implementation status	UT coverage status
ISensor	Interface for sensors.	Implemented	N/A
IDCMotor	Interface for motors.	Implemented	N/A
IOdometry	Odometry interface.	Implemented	N/A
OdometryManager	Component responsible for odometry determination.	Implemented	Covered
SimpleOdometry	Component responsible for simple odometry determination (based only on Encoders).	WIP poolOdometry to be implemented	Covered
AdvancedOdometry	Component responsible for advanced odometry determination (based on IMU readings and encoders).	WIP poolOdometry to be implemented	Covered
CommManager	Component responsible for WiFi connection and communications.	Implemented	Covered
DCMotor	Component responsible for driving a single motor.	Implemented	Covered
DistanceSensor	Component responsible for reading distance with a single sensor.	Implemented	Covered
driver	Kernel of the system.	WIP	N/A
Encoder	Component responsible for handling single encoder.	Implemented	Covered
IMU	Component responsible for handling IMU.	WIP init and pool to be implemented	Covered
MotorManager	Component responsible for driving motors attached to the system.	Implemented	Covered
SensorManager	Owner of sensors connected to the system, responsible for handling.	Implemented	Covered
WiFiManager	Component responsible for managing WiFi connection and communications	Implemented	N/A
custom::array	Custom array class to avoid STL containers in production env.	Implemented	Covered
custom::stack	Custom stack class based on custom::array.	Implemented	Covered

Requirements veryfication (SQT)

References