VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
INFORMATION TECHNOLOGIES STUDY PROGRAM

Problem-Based Project

Technical specification

Done by:

Gabrielius Drungilas

Aistė Grigaliūnaitė

Nedas Janušauskas

Adomas Jonavičius

Supervisor:
dr. Linas Bukauskas

Vilnius
2022

Contents

1 Overview

1.1 Project overview

The goal of the project is to design an autonomous robot, which would be able to detect, avoid and follow objects in an indoor environment. The rover will be able to roam around the area freely and remember the layout. Ultrasound sensors will be used to detect objects and map the environment. A micro-controller is used to control the motors and receive input from the sensors.

1.2 Hardware diagram

The robot's hardware is represented in the following diagram:

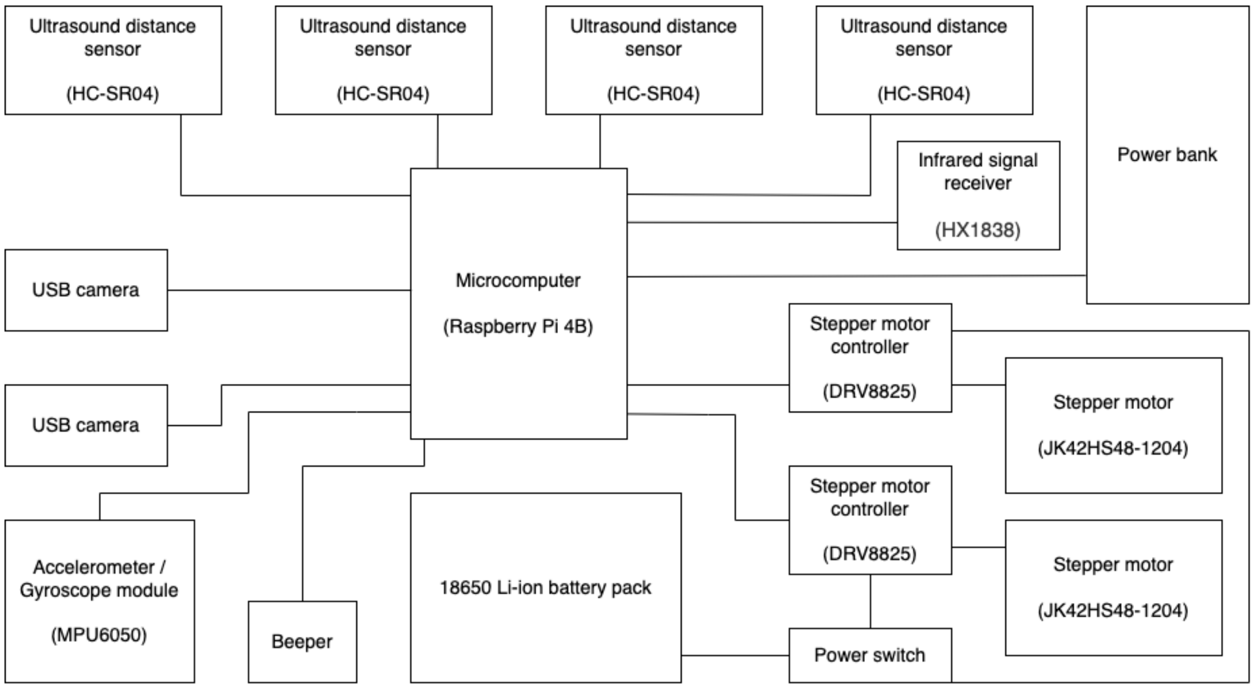
Ultrasound distance sensor (HC-SR04)
Ultrasound distance sensor (HC-SR04)
Ultrasound distance sensor (HC-SR04)
Ultrasound distance sensor (HC-SR04)
Infrared signal receiver (HX1838)
Power bank
USB camera
Microcomputer (Raspberry Pi 4B)
USB camera
Stepper motor controller (DRV8825)
Stepper motor (JK42HS48-1204)
Stepper motor controller (DRV8825)
Stepper motor (JK42HS48-1204)
Accelerometer / Gyroscope module (MPU6050)
Beeper
18650 Li-ion battery pack
Power switch

Figure 1. Hardware diagram

1.3 Context diagram

The robot’s context diagram is represented in the following:

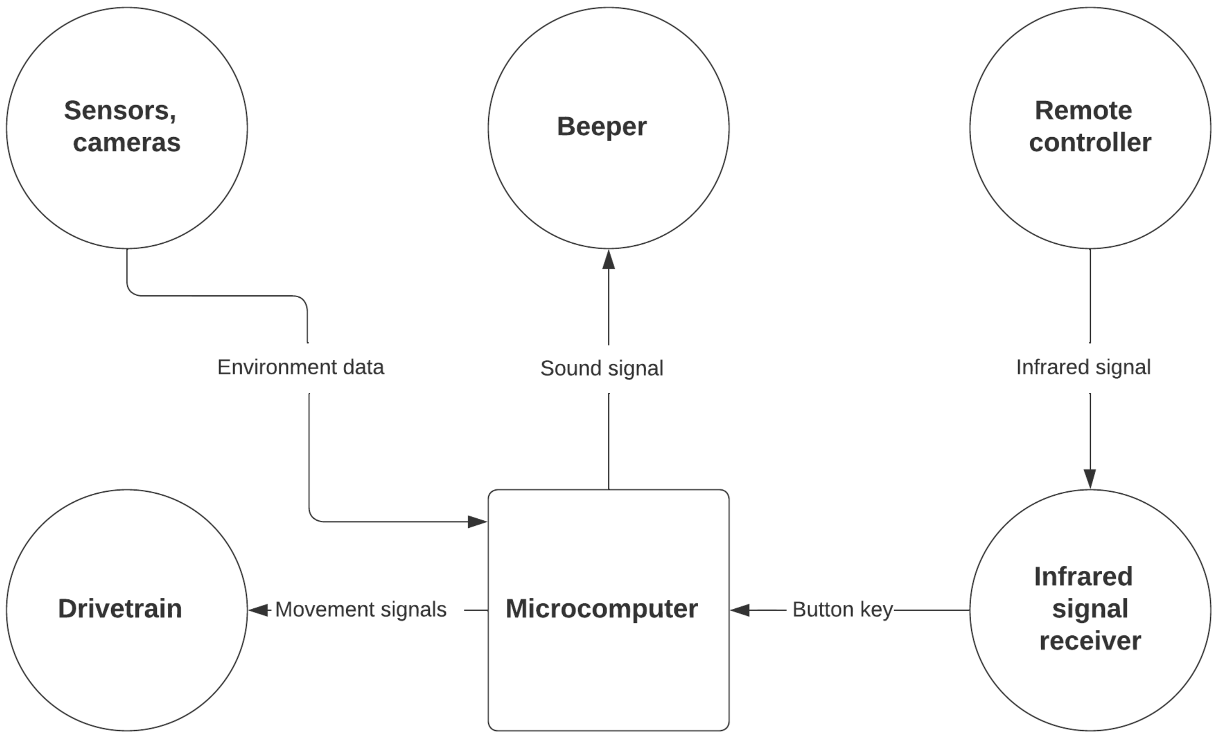

Figure 2. Context diagram

1.4 UML deployment diagram

The robot's UML deployment diagram is represented in the following:

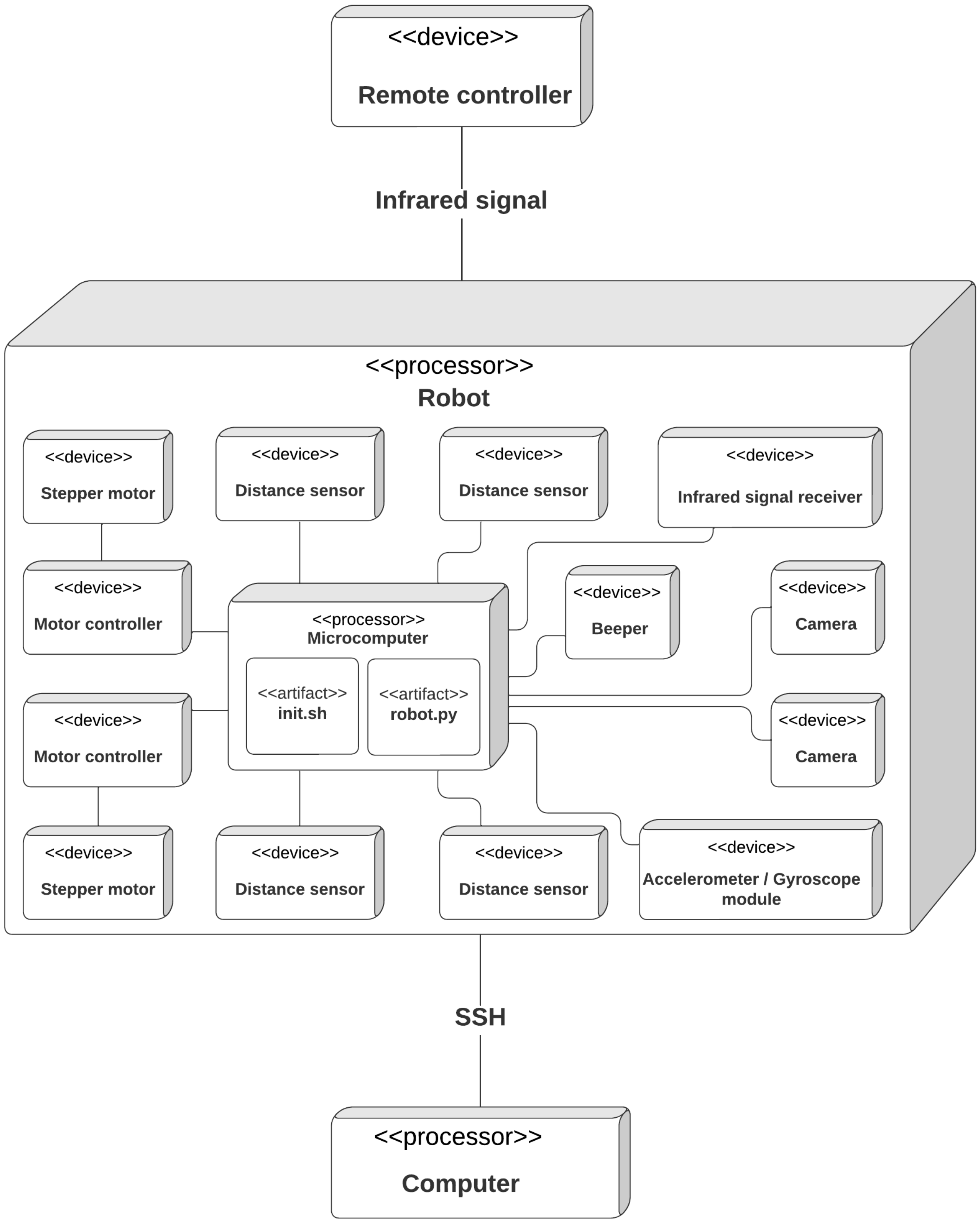

Figure 3. Deployment diagram

2 Deliverable internals

2.1 Structural aspects

The main components of the robot are the Raspberry Pi, cameras and sensors, motors, infrared signal sensor and the beeper. The cameras and sensors will be used to detect the robot's surroundings, the motors will be used to move the robot. The infrared signal sensor will be used to read the signals from the remote. The beeper will be used to warn the user of errors and events, and the Raspberry Pi will be used to execute the required code.

The Raspberry Pi will run on Linux OS, and BASH will be used to run any required scripts on startup. The code for the robot will be written using python, OpenCV libraries will be used for computer vision.

2.2 Dynamic aspects

In order to accomplish the desired functionality, the robot's components will send data to the Raspberry Pi. The infrared signal receiver, when it receives a signal, will add a task to the robot's task queue. The stepper motors will make it possible for the robot to keep track of its location in the room, as well as move a specified distance.

2.2.1 Basic robot control

A remote controller will send an infrared signal to the infrared signal receiver. This will be the main way that user is going to interact with robot. The user will be able to send signals to start different tasks like roaming freely, finding a human, returning to starting position, stopping, using beeper, and similar tasks. Tasks received by remote controller can be added to the task list and will be executed sequentially.

2.2.2 Internal logic of USB cameras

A "camera" class is be created as an extension of the thread class to be able to keep cameras working together at the same time. A separate thread is be created that will compare the view of the different cameras and calculate the disparity between different areas of the picture. Then, the output of the disparity calculation is taken, the path ahead is analyzed, it is decided whether it is clear, and the result is used to update the 2D array representing the known map of the surrounding area. Then, the next move the robot should make to get to the goal is calculated.

The known map will hold all data about the surrounding area: Each element in the array will represent the state associated with a particular area. The element will be an object with various state flags and additional information. It can hold states like: unknown, current rover position, obstacle, human, destination, starting position, and any additional useful information that could impact the decision-making of the rover. These elements will be held in a dictionary and new elements will be added if needed to represent new location.

2.2.3 Path finding

The path will be calculated using lightning algorithm, to find the shortest viable path. If such path does not exist, the bot will analyze if a path is still possible, possibly through unexplored area, then test it.

2.2.4 Threading

There will be 4 threads: 2 for camera input, 1 waiting for signals from infrared signal receiver, 1 main thread to analyze information provided by other treads, calculate appropriate response and execute it.

3 Testing

The robot will be tested using two methods:

3.1 Manually in an indoor environment

This is a reliable way to test the functionality of the code, especially when testing features that cannot be tested using a simulation, such as computer vision. Manual testing will be our preferred method.

3.2 Using simulation software "Webots"

This makes testing more accessible by sharing the code in a simulated environment. With these simulations we will test small features like movement or path finding.

4 Technologies and Tools

The following technologies will be used to build and test the robot, as well as maintain the required code.

4.0.1 Software used on the Raspberry Pi

- Linux OS - OS that will be run on the Raspberry Pi.
- Python - code for the robot's functionality.
- C++ - code for hardware tests.
- OpenCV - libraries used for computer vision.
- Bash - used to execute any scripts required on startup.

4.0.2 Software used for development

- Git and GitLab - version control.
- Webots - robot simulation environment.
- VSCode - development environment.

4.0.3 Miscellaneous software

- AutoCAD - 3D modeling.
- 3D printer - manufacturing of robot body parts.