Intelligente software voor robotstofzuiger

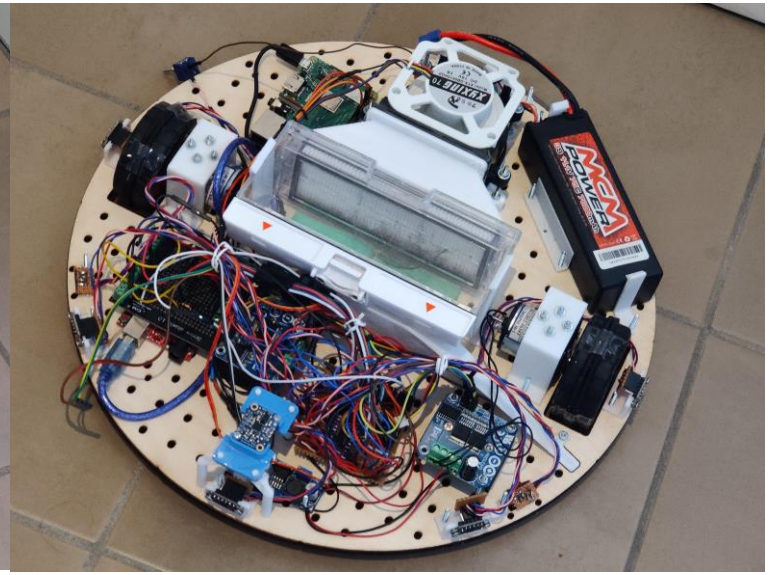
Marco Van Cleemput heeft op basis van de Puck collect robot een werkende robotstofzuiger gemaakt. Deze gebruikt zowel een arduino als een rasberry pi voor de aansturing. De Robot gebruikt Time of Flight afstand sensoren en een IMU met compas in een simpel lokalisatie algoritme. Voor deze masterproef is het de bedoeling om een geavanceerd SLAM (simulatious localisation and mapping) algoritme te gebruiken om een ongekende ruimte efficiënt te stofzuigen. Je zal moeten onderzoek doen naar hoe een map (kaart van omgeving) gemaakt kan worden. Uit experimenten blijkt dat er nog best nog sensoren bijkomen. De industriële standaard voor robot software is ROS (robot operating system), het integreren hiervan kan heel wat voordelen zorgen voor de robot.

Voor ELO/ICT-studenten

Het werk omvat

- Literatuurstudie
- Studie van de software en algoritmen
- Integreren van een SLAM algoritme
- Programmeren, debuggen en testen
- Experimenten en meten van doeltreffendheid
- Testen van sensoren

Begeleiding: Jan Lemeire & Thibault Thetier



Curiosity-driven exploration for self-learning robots

We want to achieve self-learning robots, able to learn about both their environment and their own capacities. The goal of this project is to find strategies to make a robot gather and process efficiently information, both about the environment and itself, in order to find insights that might be useful later in its life. We believe curiosity can serve as an intrinsic reward signal that can push robots to explore in the search for more knowledge.

In this project, the student will need to come up with exploration strategies to gather data (sensed data, robot state, environment configuration, accuracy/validity of sensors, etc.), evaluate how much information it contains, and make the robot learn from it. Such strategies could be used in many different contexts such as calibration, map exploitation, fault diagnosis, etc.

The student will have the opportunity to work in our new robotics lab which has a testing environment, robot prototypes, various sensors and software to analyze the sensed data and implement the algorithms.

Turtlebot

ROS
Robot Operating System

<u>The work includes</u>

- Literature study
- Getting acquainted with the robot and ROS (python or c++) systems
- Implementing possible existing exploration, evaluation and learning algorithms
- Setting up one or multiple appropriate test environments
- Testing in our robotics lab (Building K, 5th floor)

Supervision - Jan Lemeire, Thibault Thétier

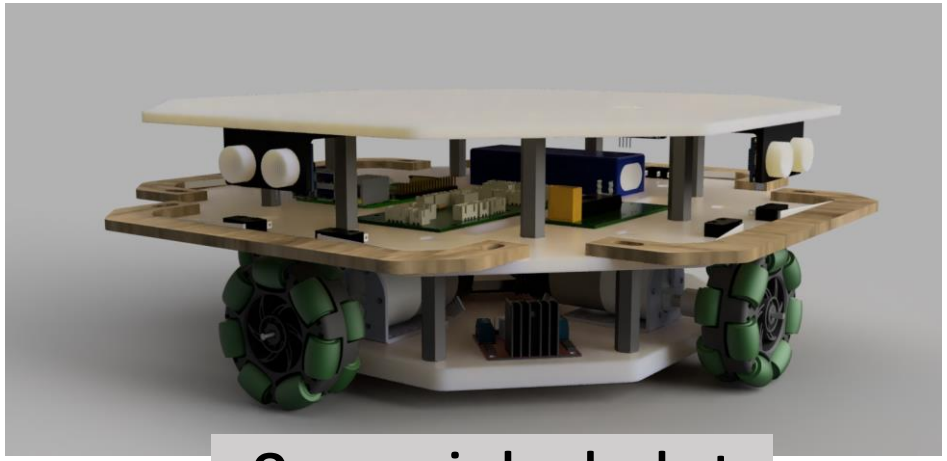
Improve the independence of mobile robots

For mobile robots to operate autonomously and reliably in uncontrolled environments, they need to be able to detect and recover from *expected and unexpected problems* that might arise while performing their tasks. Possible problems are obstacles, wrong estimation of position, broken sensor, wrong sensor values, etcetera.

To solve this problem, we propose a ‘smart’ Fault Detection, Isolation and Recovery (FDIR) architecture: the robot continuously monitors its own behavior to detect errors. Then the robot tries to find out what the problem is (called active diagnosis) and tries out several recovery strategies.

To develop algorithms, we will *emulate faults* and create problematic situations which the robot has to solve.

The student will have the opportunity to work in our new robotics lab which has a testing environment, robot prototypes, various sensors and software to analyze the sensed data and implement the algorithms.



Our omniwheel robot

<u>The work includes</u>

- Literature study
- Getting acquainted with the robot and current framework (python)
- Implementing a real-time fault-handling process
- Setting up case studies
- Testing in our robotics lab (Building K, 5th floor)

Supervision - Jan Lemeire, Thibault Thétier