R&D Report

Project: Grabby

Hendrikse Jeremy & Ortstadt Julius

Project Presentation

Grabby the Warehouse Robot is an autonomous robot designed to enhance productivity while reducing costs and errors in warehouses of all sizes.

The robot consists of two main parts:

1. **The Mobile Base**, which houses the battery, logic processor, and environment sensors (LIDAR).
2. **The Grabbing System**, mounted on a lifting platform, equipped with a stereo camera.

Grabby is built to perform two primary tasks: efficiently placing boxes in storage and retrieving them when needed. A more detailed explanation of the robot is available on our GitHub.

Current project state

Development on Grabby mainly focuses on software development as the first prototype hardware was already developed during the last year.

The mobile base

During this year we want to make the robot operational. Meaning that it can autonomously navigate and perform the desired tasks. We had some functions working independently like reading the LiDAR data or moving the robot, but we still needed to combine all this under a single process.

We first focused our work on the motor encoders. These are crucial since they will provide the odometry data necessary for accurate mapping and navigation. We managed to read the data from the encoders and convert it to usable units like RPM, distance (in meters) and speed (in meters per second). To this, we coupled a Bluetooth transmitter to visualize the data on our computer independently from the NVIDIA Jetson Nano board.

Now that we had some good odometry data from the motors and the wheels, we moved on to another sensor: the Inertial Measurement Unit (IMU). We used the MPU-6050, a 6-axis IMU having an accelerometer (3 axis) and a gyrometer (3 axis). With this component we can precisely measure the linear and angular accelerations of the robot. We tried using a 9-axies IMU (MPU-9250) which additionally has a magnetometer for yaw angle estimation. However, since our robot will work indoors, this approach won't work as there is too much interference. We needed to estimate the yaw angle in order to estimate the heading of the robot so that we can keep it moving straight. Since we couldn't do it the old-fashioned way, we decided to combine the data from the two motors (which make up a differential drive) and the MPU-6050 data to estimate the yaw angle. This combined with a complementary filter worked acceptably. This is something that will need to be revised at a later date, but it is sufficient for now.

After all the work done on the IMU combined with the Arduino, we connected the sensor to the Jetson via I2C. It is better to read this sensor data directly with the Jetson as ROS already has an existing package to control this sensor. The interfacing with ROS worked and we were able to see a simulated representation of the linear and angular positions of the sensor in RVIZ.

We also tried using ROS2 on the Jetson (ROS2 Dashing as it is the only supported ROS2 version for Ubuntu 18.04 LTS which runs on the Jetson). However, after a lot of problem solving and careful consideration, we decided to keep working with ROS1 Melodic as this distribution offers a lot of support for the LiDAR and the mapping / SLAM algorithms we intend to use.

Last year we managed to successfully connect the LiDAR and the Arduino to the Jetson and to perform a basic mapping test. The resulting map was of poor quality and so this was the problem to address next. The best choice was to teleoperate the robot while it performs the mapping. If this manual approach doesn't produce an acceptable map, we can always ***bag*** the odometry data from this mapping and complement the map with this data. Also, the choice of the mapping algorithm will influence the result (HECTOR-SLAM or GMapping). This is still something that we will need to test but is likely to be validated in the coming weeks.

The grabbing system

To improve greatly the software capabilities and connect to the Jetson Nano we had to start using ROS even on the Arduino. Until now we had already started ROS on a virtual machine having the same characteristics as the Jetson Nano board. This enabled us to partially work at home on the ROS architecture and program while the robot was used for other testing tasks.

But there were a lot of issues, after having a difficult start and eventually making it work, we discovered a problem. When creating a custom service or message we need to rebuild the ***ros_lib*** library on the Arduino for it to recognize and use the custom service and message types. Except the script that rebuilds the library fails to build our services and messages making it impossible to use them. This seems like a pretty simple error but after a lot of searching and testing the errors remain. But when testing the same steps and code on the actual board and not a virtual machine, the problems disappear.

This partially solved problem induced a new one: working on the board simultaneously was not possible. We therefore borrowed a board from another group. The next step is to reset this board and clone the existing image onto this board. We then pull the GIT repo and continue developing the architecture of the robot.

Meanwhile even if we could not work and test directly ROS, we made the code structure and defined everything that would be useful. Which means that when the second card is operational, the development will be much faster for this part of the project.

Remaining Work

The mobile base

There is a lot of work that remains to be done on the mobile base for autonomous navigation. However, since a lot of small bits have already been done, we will only need to put them together and finetune everything so that all the systems work together.

The mapping and SLAM navigation is the most important one as for this to work we need to combine the odometry data from the wheels and the IMU with the data from the LiDAR and the ultrasonic sensor. To do this we need to create different ROS packages to manage all the different nodes, for example for the teleoperation, for velocity commands to drive the wheels etc. These readings and data conversion need to be very precise which is also a major step in development to be taken.

The grabbing system

There is still a lot to be done for the robot to identify and grab the parcels. First define the low-level algorithms and commands to perform basic and precise movements and actions. Then analyze the environment with the stereo-camera and QR-code reader to detect the shelves and the parcels. Finally, the robot needs to perform actions in relation to the environment. All this means that further software development is needed to implement the camera and the QR -code reader. This will then eventually be merged into the code for autonomous navigation which is still a major and complex step.