LAZARUS PROJECT REPORT

Ostfalia
Hochschule für angewandte Wissenschaften

UPSSITECH
ÉCOLE D'INGÉNIEURS

Group 3 Report

2021

UPSSITECH
ÉCOLE D'INGÉNIEURS

Ostfalia
Hochschule für angewandte Wissenschaften

Sommaire

I Robot.....P1

A° Sensors and Actuators.....P1

B° Communication.....P2

C° Automatic mode.....P2

II Decision.....P3

A° Connection with the robot.....P3

B° Mapping of obstacles.....P3

III HMI.....P5

A° Interface.....P5

B° Control.....P5

C° Other Feature.....P6

IV Final Assessment P6

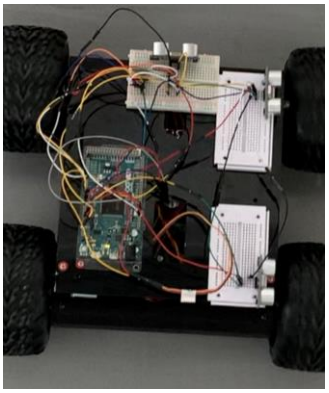
Abstract

The goal of this project was to design a robot from scratch. We had to build it, connect sensors, deal with the bluetooth connection and design the HMI. The point was to create an autonomous mode, and create an automatic mapping of the robot's trajectory and its environment. We managed to form three different teams, with every member of each team working on their favourite domain of competences. All three teams were always in touch thanks to means of communication and reunions. Due to all this work, we are able to present the result of our final work.

I Robot

Aº Sensors and Actuators

As every robot, the lazarus robot follows the perception - decision - action loop.
To achieve the mapping functionalities we need captors which are able to detect walls, and deduce a distance from them. One possibility is to use ultrasonic sensors. It's not the most precise technology but it's cheap and that's enough to do simple mapping.
We use three ultrasonic sensors (SR-04) which are placed as shows this picture :



We decided to put one sensor on the front because it's useful to detect corners or obstacles.
Then two other additional sensors on the right side as the robot needs to follow the wall and go around the room, the right side is always in front of the wall. With these sensors, we can measure the distance from the wall in front of the robot and on the right. They will be used to map the room but also to navigate in the automatic mode.
Moreover, we added 2 encoders (QME-01) on back motors to calculate the speed of the robot and deduce the position of the robot.
With these captors we can now estimate more or less precisely the environment of the robot.
To map a room, we need to move around. To do that, we use four motors. The motors controller provided can control two motors, so we plugged right motors on the first output and left motors on the second output.
Motors can be controlled through a manual mode with the HMI or through the automatic mode. You will read the details of these functionalities later on the report.

With all these sensors and actuators we can now navigate in the room and detect walls. We can also calculate the speed, coordinates and rotation angle of the robot.

Bº Communication

As we said in the previous part, we have various data to map the room.
We decided to send these data through bluetooth technologies because it's a common and reliable technology.
The coordinates are calculated according to the start position.

We send all the following data with a basic protocol through bluetooth :

- Coordinate (x and y)
- Rotation angle
- Front distance
- Front right distance
- Back right distance

We send these in a string, which start with the letter A, and end with the letter Z.
It allows the software on the computer to split the data and process it.

Cº Automatic mode

Regarding the automatic mode, we decided to set up a state machine that would take all the possible cases we took into account so as to act accordingly. Then, we have implemented the state machine directly on the arduino code. This implementation permits the robot to find the first wall, turn 90 degrees on the left and parallelize it if necessary. Then he follows the wall while he is parallel. If he detects whether he is too close or too far away from the wall he will adjust his trajectory to get away or get closer. If it is no longer parallel to the wall, he will correct his trajectory too.

Finally, if it loses the detection of the sensor front-right, he will conclude that it has to avoid an obstacle and will go forward and then turn 90 degrees on the right. Else, if the robot detects a wall in front of him, it will turn 90 to the left and restart the scenario. To this day, the robot can only make a complete turn in an empty room or a room with huges obstacles along the wall.

II Decision

Aº Connection with the robot

Concerning the connection between the robot and the computer, we opted for the bluetooth solution. Indeed, the robot communicates with the computer through serial two-way communication. Since the software part is coded in java, we used the jssc library to connect to the robot serial port .

To open the communication, the computer connects to the specified serial port and waits for a first reception (of any character) from the robot. As soon as it receives it, the software begins to process the raw data sent by the robot, detecting both start and end flags. We implemented a control function that ensures to get the data since the "A" flag, especially for the first point. Then the data is treated and transformed into a Status message. The latter is then sent to the Mapping function following a pattern observer model. If ever the connection is lost, the software will manage to connect back to the robot during the determined time. Afterwards, a serial port must be selected again.

When the operator is done with the robot, the latter is disconnected.

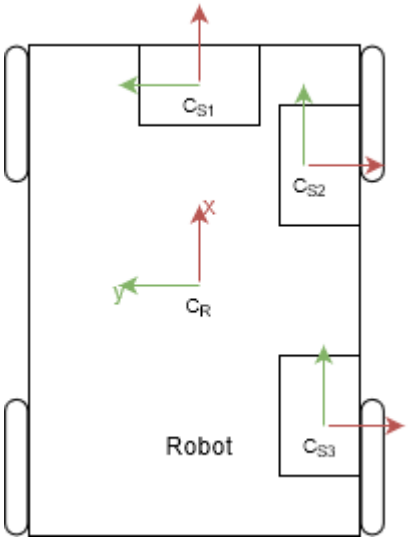
Bº Mapping of Obstacles

The problem of mapping consists of taking the positional data of the robot and its sensor data and storing the resulting positions of obstacles in a scalable, easily readable way.

The chosen solution uses a grayscale Image, which is initially completely grey to represent unknown space. The scale of the image is constant, meaning a pixel always represents a fixed distance in the real world. If the robot gets beyond the edge of the area that is given by the image's resolution, the size of the picture is doubled and the current map is drawn in the middle.

When a status message from the robot is received, the positions of the detected obstacles are computed from the robot's position and the distances that the sensors report. The corresponding spots on the image are then colored black (to represent an obstacle) and the traces from the sensor's origin to their respective end points are colored white (to represent free space). If the reported distance is near the maximum range of the sensors, it is assumed that no obstacle is detected and the end point of the trace is not colored black.

The way the start and end points of the sensor's traces (i.e. the sensor itself and an obstacle) are computed is based on transformation matrices. A transformation Matrix from world coordinates (C_0) to robot reference frame (C_R) is created with every status message from the robot.



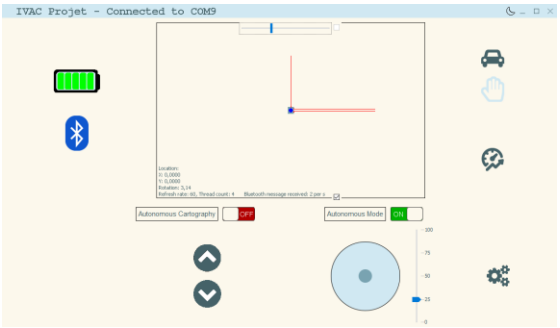
A second, static transformation matrix from robot reference frame to sensor reference frames (C_{S1}, C_{S2}, C_{S3}) are given. The product of these two gives us a transformation from world coordinates to sensor-specific reference frames. The points (0, 0) and (distance, 0) in the sensor reference frame are the start and end points of the given sensor.

New traces from newer status messages are simply drawn over the current picture. This means the picture always has the newest information, but doesn't highlight spots of "moving" objects, which can be identified by occupied->free (black->white) and free->occupied (white->black) transitions.

The resulting picture can then be simply given to the HMI, since it is already visualizable.

III HMI

A° Interface



First of all, our application is presented with a window that represents a map. This will allow us to see the trajectory that the robot follows if the autonomous cartography mode is activated. Then, we added a slider to control the zoom on the robot. Thanks to that, if the robot goes out from the window, it will automatically zoom out. On the contrary, if the robot goes back to its initial position, then it will automatically zoom in. It's also possible for the user to lock the map on the robot, so we can follow it through the world. So, if the robot goes out from the window, it will follow it in this trajectory.

Next, we implement some interactive lights. The first one simulates the robot's battery information, because it's not possible for us yet to have this information, but it's the kind of feature that it will be nice to implement if we have the opportunity to do so. The second one gives feedback to the user, so he knows if the bluetooth connection is up. The last two ones allow the user to be informed on the current state of the robot: running or stopped.

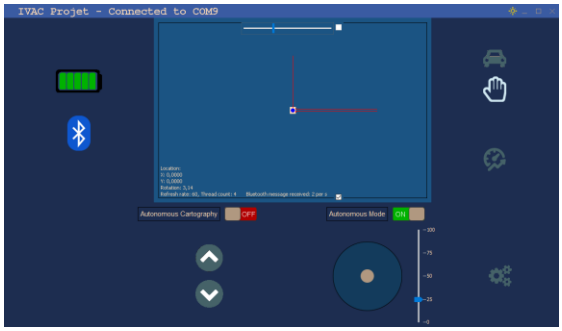
B° Control

In order to control the direction of our robot, we decided to design a joystick. This joystick allows us to control the speed of the robot. This speed will be determined regarding how much you push on this joystick. We also add a full forward and a full backwards button, that go straight at the maximum speed. This maximum speed is determined by a slider that the user can move anywhere and anytime.

Then, we thought it was a cool idea to implement a performance mode. It will allow the user to add 50% of speed at its maximum speed. Thanks to this feature, we improve fun, but we have to keep in mind safety. In a way to achieve that, a popup will drop if this mode is activated in order to warn the user about the potential danger of this feature.

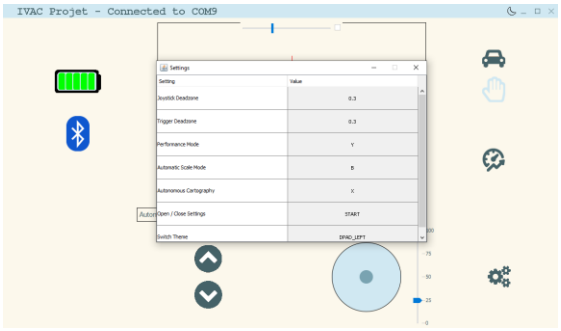
Next, we added two switch buttons. The first one is to control the autonomous cartography, and the second one is to switch between the manual mode and the autonomous mode.

C° Other Features



The first “bonus” feature we wanted to add is to be able to control the robot with a video gamepad. So, all the action possible with any button on the application will be practicable thanks to a key on the controller. Plus, when a controller is connected, the information is given to the user on the bottom left of the map. Then, we added a dark mode to the application, to improve the user experience. The user can switch between the two at any time by pressing the top right button in the title bar.

We also added a “user friendly” feature. When you keep the key “alt” pressed and you overview any key with your mouse, it provides to the user a short description of the purpose of any elements present on the application.



Then we designed a settings menu. Thanks to this menu it's possible to reassign any possible action to a different key on the controller. Next, at the launching of the application a window requests the user to select a COM port.

IV Final Assessment

This user guide will describe step by step the procedure to turn on the robot and being able to control it.

1° Turn on the switch button on the front of the robot

2° Launch your Java app

3° Select your COM port

4° Now you can connect a controller (only for Windows), or use the mouse

Final result

It's time for a little retrospective on this project, and see what we have done, what we didn't manage to realise. First the HMI, fully functional with all the features required at the beginning of the project, and even more. We tried to push it the best we could, and we are pretty proud of the result. The connection is sometimes unstable due to some components we use. But when it works well, the manual mode is functional. The autonomous mode is also working well, except that it is not able yet to avoid small obstacles like chairs for instance . However, we are still struggling to be very accurate on the calculation of the robot's position and its angle. These two elements impact the quality of our mapping, which is not fully efficient.

Regarding the exploration of the room, a more sophisticated technique that would also explore the inner area of the room and not just the outer walls was not found or implemented.

Appreciations

Concerning the Upssitech crew, this project has been a really cool opportunity to continue exploring what we have done this year. Pooling everything we have learned to design this project has been very interesting. Also, it's now the second project that this team does together, and a real alchemy has been created. We are more and more efficient, and the working atmosphere is always very nice. Plus, working with the german team has been enriching in terms of communication and teamwork. So in the end we would like to thank the Ostfalia university and the Upssitech and all the people involved in this project. We would also like to deliver a massive thank you to the AIP platform for their welcome during this project.

Regarding the Ostfalia members, this has been our first international work and it was an exciting experience. Unfortunately our best area was C and Microcontrollers, which wasn't ideal. One of us worked on the robot which proved to be annoying due to the distance, while the other member worked on higher-level java code, which took some time to accommodate. Regardless, it was great to work together.