Introduction to Robotics Lab2 – Mobile robots

Authors: Louis Graz / Gauthier Bassereau

1. Introduction

This lab report details the exploration of mobile robot control and sensor data retrieval using an mBot robot [1]. The primary objectives were to:

Gain proficiency in controlling a mobile robot using C code.

Retrieve data from various onboard sensors, including an ultrasonic sensor for distance measurement, an Inertial Measurement Unit (IMU) for motion detection, and a line follower sensor for path tracking.

Practice fundamental software development practices using Git version control for code management [2].

The mBot mobile robot served as the platform for experimentation. This educational robot is equipped with an embedded microcontroller (mCore board) to which various sensors and actuators can be easily connected using RJ25 connectors [1]. The mBot is classified as a differential drive robot, meaning it has two independently controlled wheels. Due to this configuration, it can be considered quasi-holonomic, meaning it can theoretically move and rotate in any direction without changing its orientation [3].

2. Methods&Results

During this lab, our experiments were incrementally more complex in order to fully understand the how we control a robot, making it act on different types of censors.

Our first experiment was on creating a code to control the robot's movement (forward, backward, turn left, turn right) using motor speed commands. The file MeMCore.h permitted us to create variables of type MeDCMotor which has a method called run that is assigned to a certain motor and give it the variable speed (see figure 1).

```
int speed = 100;
MeDCMotor _leftMotor(9);
MeDCMotor _rightMotor(10);

void setup() {
  // put your setup code here, to run once:
  _leftMotor.run(0);
  _rightMotor.run(0);
}
```

Figure 1

Our next step was about retrieving the distance using the ultrasonic sensor and printing it on the serial monitor. Using the same library as before, we were able to create a variable of type MeUltrasonicSensor this time which had a method called distanceCM(), returning the a float value in centimetres (see figure 2).

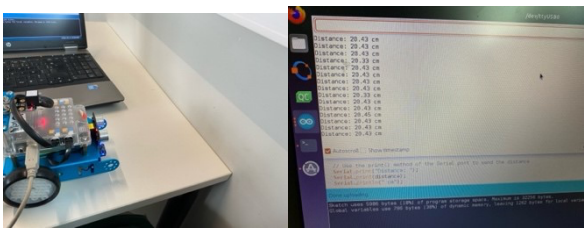

Figure 2

By combining the two previous experiments, we aimed at replicating the behaviour of a Roomba robot, which consists in going straight forward and when facing an object, pivoting by a certain number of degrees and doing the same process again and again. Our own integration of this type of robot in our code followed the steps inside an infinite loop (see figure 3)

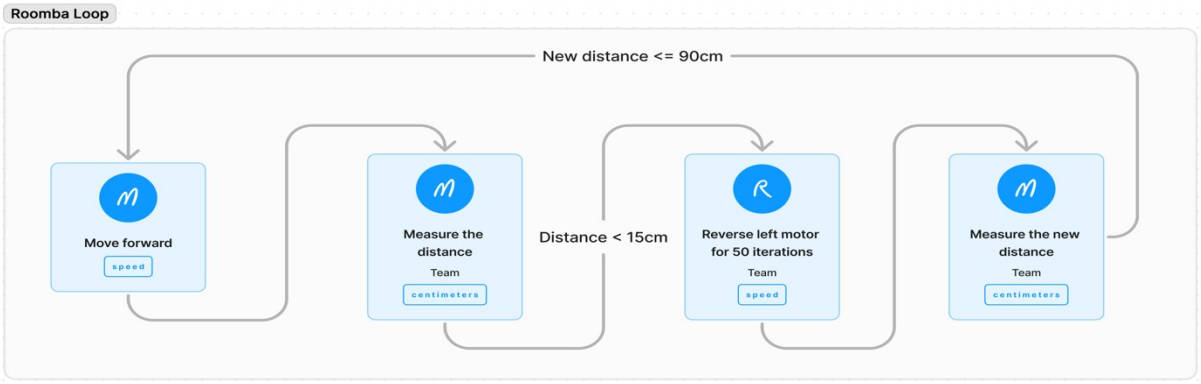

Figure 3

The double check of distance in each loop is actually an upgrade we decided to make since we observed that, without it, the robot will not pivot enough in function of the direction it hits the wall. That is why we decided to make it correct itself, it will turn then check the distance to know if he turned enough. This upgrade dramatically improved the number of failure when facing an object.

In order to properly control the robot, we need to have access to an accelerometer and gyroscope to be able to deduce orientation and so its movement in space. This data is given by the IMU sensor [6], using the method getAngleX, getAngleY and getAngleZ, we were able to get the degree of orientation around the three axis giving us valuable information for turning in case of facing an obstacle, we were able to adjust the turning to a precision orientation. Another method was the getGyroX and getGyroY which is the acceleration around the two axis which is also valuable information to give us more control on the speed of turning. In order to have access to more data we can use readData() method.

Ongoing to the application of the previous test, this section presented a challenge to develop a path control program for the robot using IMU data. The goal was to make the robot travel to a desired position by first rotating towards it and then moving straight for a calculated distance. The section introduced the concept of proportional control for rotation movements and open-loop control for translations. The robot is primarly at a positon of x=0 and y=0, based on a given target point X and Y it will calculate the wanted orientation around Z axis using the getAngleZ function and then counting the time it should take to go to the point based on its speed. The biggest challenge is probably to calculate the speed of each motor chen wanting to go to a certain angle (see figure 4).

```
void rotateToTargetAngle() {
  _errorAngle = _targetAngle - _thetaCurrent;

  while (abs(_errorAngle) > 5) {
    double w = _Kp_angle * _errorAngle;
    double leftWheelSpeed  = -_distanceBetweenWheels * w / _wheelDiameter;
    double rightWheelSpeed  = _distanceBetweenWheels * w / _wheelDiameter;
```

Figure 4

Other details of the code are all in the file Control.ino. However, we have not yet been able to test in real condition the code.

3. Discussion

In the exploration of mobile robot control and sensor data retrieval with the mBot platform, several constraints and considerations emerged. One critical aspect is the reliability and accuracy of sensor data. While the ultrasonic sensor provided distance measurements, we observed limitations in precision and accuracy, particularly in real-world scenarios where environmental factors could affect readings. For instance, variations in material composition and surface texture might impact the sensor's ability to accurately detect obstacles, thus affecting the robot's navigation. Additionally, the line follower sensor, while effective under controlled conditions, might struggle with complex line patterns or variations in ambient lighting, highlighting the importance of robust sensor calibration and environmental adaptation algorithms.

Another constraint relates to the limitations of the control algorithms implemented. In the context of the Roomba behaviour emulation, we encountered challenges in optimizing the robot's response to obstacles. While the simple pivot-and-move strategy exhibited improved performance with iterative refinement, it remains susceptible to situations where obstacles are encountered at acute angles or in confined spaces. Developing more sophisticated obstacle avoidance strategies, possibly incorporating machine learning algorithms or probabilistic reasoning, could enhance the robot's adaptability in diverse environments.

Regarding the control part, even without testing it we can see the limits of such a strategy. First the angle has an error of +/- 5 degrees which is quite a lot, but the motors and sensors don’t seem capable to modify and identify

such little margin. Moreover, the method used to get to the distance is a bit blind, if there were some kind of obstacle the robot wouldn't be aware about it and, after a certain time would think he arrived at wanted destination. To improve this, we could definitely make it do more calculations during its forward phase to review, first its orientation and then its distance, so that it can adjust progressively its commands.

4. Conclusion

In conclusion, this lab allowed us to dive into the world of mobile robot control and sensor data retrieval using the mBot platform. Throughout the experiments, we gained hands-on experience in coding, integrating sensors, and designing control algorithms. We managed to control the robot's movement, obtain data from various onboard sensors, and even emulate a Roomba robot's behavior. However, we faced some challenges related to sensor limitations and algorithmic complexities, emphasizing the need for better sensor calibration, environmental adaptation, and advanced obstacle avoidance techniques. Despite these obstacles, the iterative nature of our experiments enabled us to continuously learn and improve. To advance in the development of more sophisticated and adaptive robotic systems capable of navigating complex real-world environments, it's essential to address these constraints and keep learning from our experiences.

5. References

[1] Makeblock. mBot Robot Kit. [Online]. Available: [invalid URL removed] [Accessed: March 22, 2024]

[2] Git SCM. Git Basics. [Online]. Available: [invalid URL removed] [Accessed: March 22, 2024]

[3] Siegwart, Roland & Nourbakhsh, Illah R. (2004). Introduction to Autonomous Mobile Robots. MIT press.

[5] Makeblock. Me Ultrasonic Sensor. [Online]. Available: [invalid URL removed] [Accessed: March 22, 2024]

[6] Makeblock. Me 3-Axis Accelerometer and Gyro Sensor. [Online]. Available: [invalid URL removed] [Accessed: March