Table of Contents

Contents

1 Progress Report

1st Iteration

Should be achieved by:

2 Mechanical Design of MazeBot

2.1 First Iteration

Goals:

- Accurate movement

Mechanical Drawing
XOURNAL LOL

Observations & Measurements

Structural Integrity of the Drive System
In order for more accurate movement, we added a high gear ratio. However, since we added the high gear ratio, we are unable to find space to properly secure the left and right drive wheels. When testing, we found one wheel to slipped forward and the other to slipped back when turning which defeats the accuracy of the encoder. Because of this, we are unable to meet our goal of accurate movement.

Robot is too large.
Since the brick is upright, it is top heavy. We needed two rods in the back and one metal ball in the front in order to balance the robot. The additions of the two rods and one metal ball remove the spacial advantage of having the robot's brick be upright. Even though the dimensions of the robot are within the size of one square, it leaves very little room for error. As such, the robot begins to run into walls after the 3^{rd} turn.

Conclusion
In conclusion, we have decided for our next iteration to have the robot's brick be flat on the ground in order to ensure that we have enough room to properly secure our drive system. By properly securing the drive system we hope to not have to readjust every time we enter a cell.

3 Software Design of MazeBot

Our main goal with the software of the mazebot was to create program solved the problem simply and was easy to build upon. Furthermore, we wanted our software to have very few constants that we would need to tested for. For example, in order to move forward one cell, we would need to give the following function the degrees to move each of our drive motors:

```
setMotorTarget(leftMotor, degrees, 75);
```

The degrees needed to move one cell forward could be achieved by constantly testing different values of degrees to achieve the movement to the new cell. However, we chose to calculate the exact degrees that the robot's drive motors would need to move in order to move exactly one cell forward. This approach in contrast to the former has two advantages:

1. It allows us to isolate any problems with moving accurately to a mechanical problem.
2. We would not have an accumulation of error because of us testing incorrectly.

Therefore, we chose to mathematically calculate the degrees that we needed to move the motors rather than testing.

A sketch of the derivation of how many degrees to move forward is shown below:
Therefore:

```
degrees = (SIZE_OF_ONE_CELL / CIRCUMFERENCE_OF_WHEEL) * DRIVE_GEAR_RATIO * ONE_ROTATION
```

A similar derivation exists for turning the robot 90 degrees:

3.1 void goFwdCell()

your code example
Briefly explain the algorithm of the maze bot

3.2 Function Description

Explain what each function does

3.3 Flow Chart

Flow chart of the algorithms (I think there is a online thing where u can make flow chart)

3.4 Function Parameters

Yea

4 Appendix

Source of Code with brief comments