Colorado School of Mines

Memo

To: Instructor and TA

From: Christian Prather

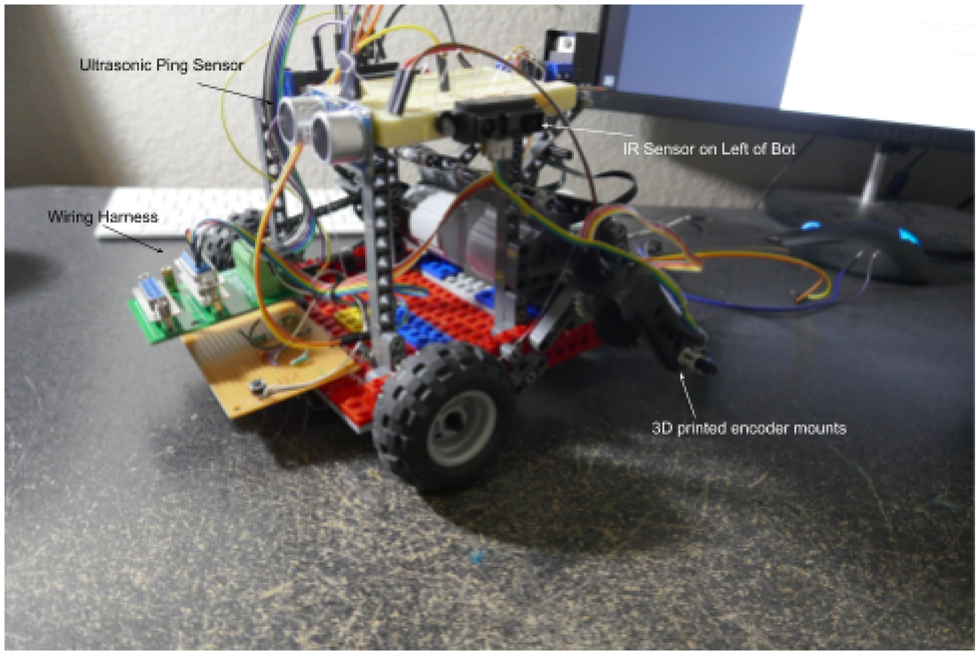
Team #: M420

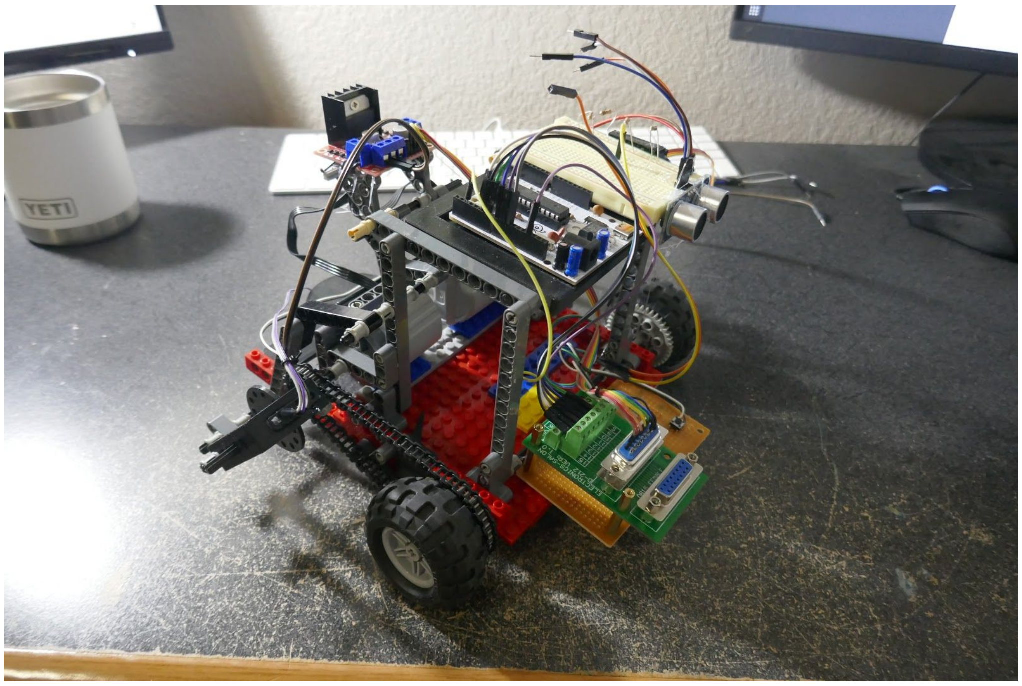
Date: 11-12-20

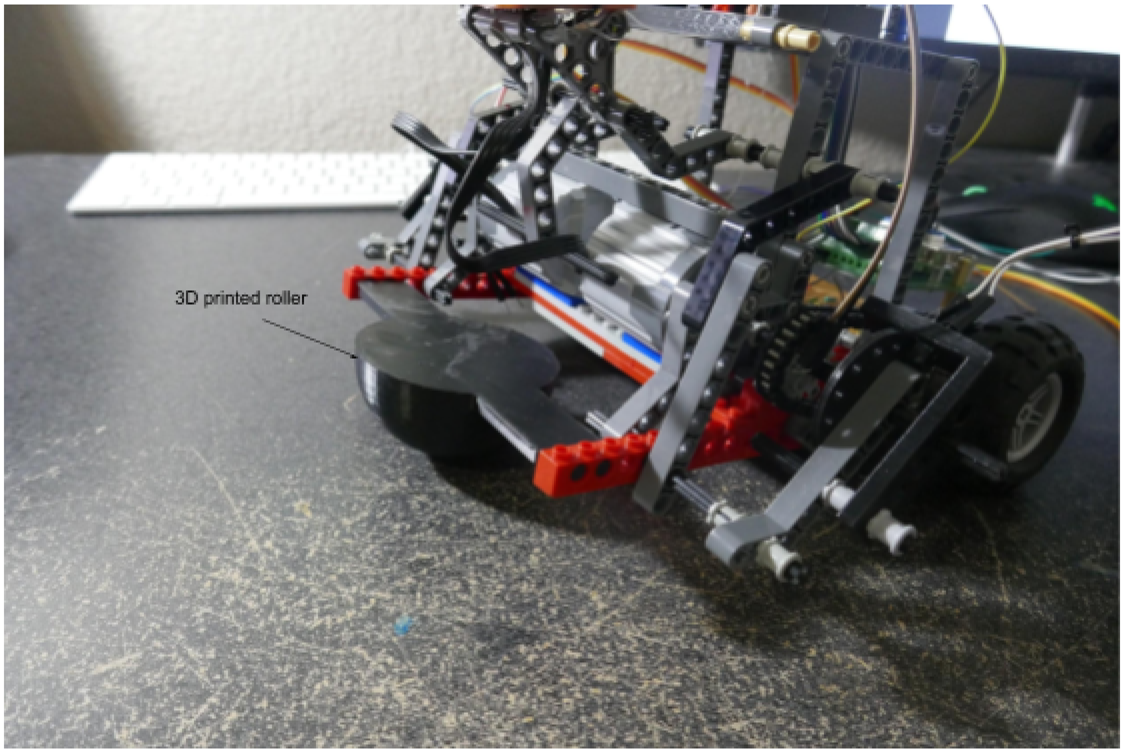
Re: Lab 4: Solving the Maze

Problem Statement:

Up until now our robotic platform has only been able to handle the navigation of a predefined maze, that is we knew what the maze looked like before we ever set the robot loose on it. This is a very rare edge case however and it is far more likely that we will have to operate in an unknown environment. This lab incorporates sensors, specifically IR and Ultrasonic to provide the robot with feedback. The robot will explore its environment until it can establish its destination. This exploration is not guaranteed to have been an optimal path so at this point the robot will recalculate an optimized approach and re-run the maze.






3D printed roller

Methods:

The approach to this problem can be broken down into three primary sections, mechanical configuration, exploration, optimization.

Mechanical:

Mechanically in order to allow a robot to explore an unknown environment it needs to have some method of understanding what's around it. This was accomplished through two sensors. An ultrasonic and an IR distance sensor were used as they offered simple information on distance from robot to object. Both sensors had to go through a level of calibration as the IR needed to have a polynomial function established to convert 0-1024 analog output to a distance in cm. This calibration was done with a separate program(ir_optimization.ino) and a simple excel file provided. The ultrasonic needed simple range limits established which was done with a trial and error check. The placement of these sensors was deliberate as well, knowing I would take a simple “walk the wall” approach to exploring/solving the maze I knew I would need to know when I was able to turn left or drive forward, this ment I would place the IR sensor on the left of the bot and the ultrasonic sensor on the front.

Exploration:

The exploration algorithm consisted of a simple decision tree based on robot state, to begin the robot is set in the center of the starting square, it follows a simple left hand rule saying if it's allowed to turn left do that otherwise drive forward. The moves taken are then recorded as well as the distance/ angle traveled. Travel was done by breaking moves this into small sections at a time, I chose to have a linear travel distance less than that of the min distance detected by the sensor as this ensures that while I am not using interrupts for the object detection I can safely drive forward without hitting an obstacle. Encoders are used to enforce a linear travel of a precise distance as discussed in the last lab. As each move is taken it is registered into an array storing the three possible moves (LEFT, RIGHT, FORWARD). The robot attempts to drive left, if that is not available then it will drive forward, and if that is not available it will turn right 90 degrees. This simulates an individual following the maze with their left hand on the wall. Once at the final destination section of the maze the robot can be notified it completed the maze with the push of a button.

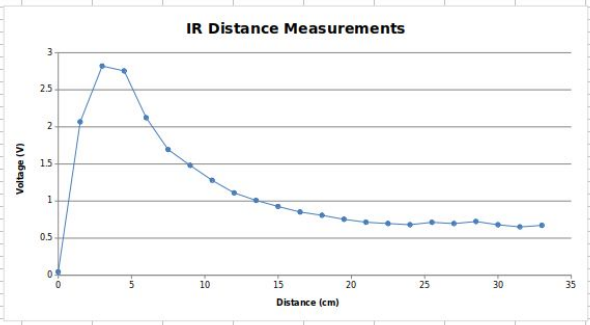
Optimization:

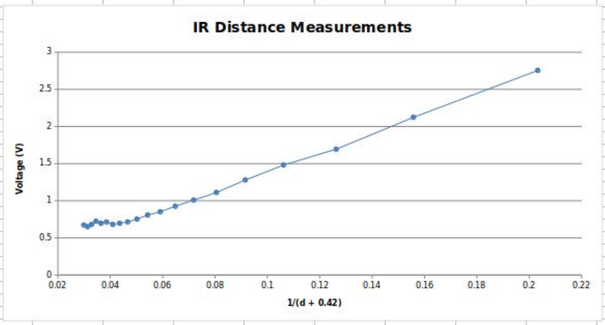
The approach algorithm explained above while simple to implement is highly inefficient in traversing the maze in an efficient way. So a post processing algorithm is run on the recorded moves to optimize the final path. The historic moves are iterated over in an attempt to locate local patterns in the data that we know can be mapped to more optimized versions which provide the same translation from point A to B. These historic patterns to optimized mappings were stored in code and multiple arrays were used to search the historic list. (this is explained in much more detail in the code). The outcome of the optimization algorithm is a simple array of optimized moves with the concatenation of distance/ angle traveled, for example two right 90 degrees are converted to a single right 180 or multiple forwards are converted to a single forward of the summed distances.

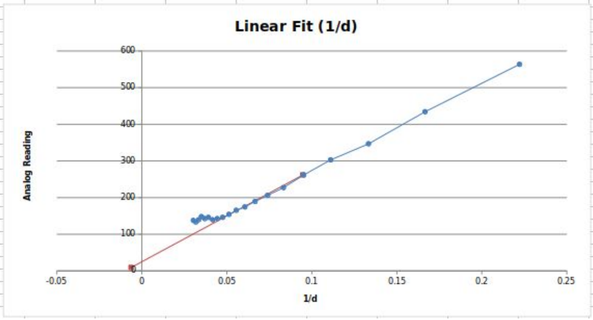
Results:

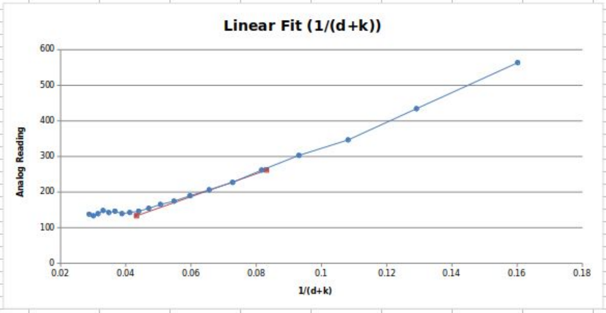
(Results all have an * by themas lab was finished at home with a simple maze construction acting as maze walls)

	Exploration time (mm:ss)	Run time	Solved
Run 1	10:36	6:23	No
Run 2	9:50	7:01	No
Run 3	10:26	7:43	No









Point 1	-0.00632	9.2
Point 2	0.09472	261.8

Point 1	0.043292	133.2
Point 2	0.083158	261.8

Conclusions:

The robot performed fairly well, its primary downfall continues to be the mechanical design and structure of it. I was able to improve upon my prior design through the inclusion of a 3D printed ball pivot in back. This drastically reduces the chances of it being caught on something and being thrown too off track for the encoders to account for. It also did well at navigating from spot to spot and while I finished the lab at home with sudo walls I feel confident it could've done very well in its detection of the boundaries of the lab maze. I was most surprised as to the complexity involved in the optimization, while there are options of existing approaches I wanted to try and implement my own. To do this I wrote a simple C++ program (attached below main.cpp) to quickly test through ideas. My resulting algorithm is ugly to say the least and in no way would pass review of another programmer but was sufficient to get the job done. I felt I did a good job of understanding the architecture I wanted with the software and how each piece should go together, this helped me to write a program that overall I am proud of. I did not do well on the optimization algorithm as stated prior and as stated in all previous labs I am unhappy with my platform's mechanical design, though I am happy with the parts I have designed with Solidworks and printed. If I had to redo the lab I would focus my attention on two areas to adjust, one would be my mechanical structure, I had some issues with my encoder mourning causing inconsistent readings and I had multiple times when parts would come apart as certain sections are under tension due to forced fits. I would also spend more time on my optimization algorithm as it is about as efficient as a potato. This would require some better architecture in pattern matching primarily.

References:

Lecture slides and the Arduino standar examples library were used as reference for this project

Appendices:

See
https://github.com/Christian-Prather/Mines-Robotics/blob/main/Lab4/custom_lab_4/docs/latex/refman.pdf for full code documentation