Block Avoidance

Team Members: Abdullah Al Mukaddim, Athirah Afwina, Lily Andruchak, Yixiao Yuan

Abstract

The team's project was designed to test the Thymio robot's ability to navigate a simulated obstacle course. They worked to create an aseba program that would enable to robot to follow a course and successfully maneuver around blocks. (include summary of results later)

Introduction

Robotics is a branch of Computer Science and Engineering that creates and programs robots to perform set tasks. The team's project was to write a program in the Aseba language for the robot to use in a robotics competition.

The robotics competition occurs in the course lab against other classmates; it consists of one event, the obstacle course. To successfully complete it, one must program the Thymio to follow a black line of tape with speed and accuracy. If it comes across a barrier, it must be able to identify it, drive around it without coming into contact with it, and consistently find the black line again after.

The team's biggest obstacle to overcome was sharpening the barrier dodge. The robot could easily sense and avoid the block but after it did, the turn back to the line was too wide and because of that it could miss the line entirely. They experimented with the turn method (pivoting on one wheel vs. driving one wheel forward and one wheel back) and turn style (curved turns vs. square turns vs. curve turns with a 90 degree angle to return to the line vs. triangular turns) to ensure that the way they were approaching the problem was the optimal solution. In the end we concluded that pivot turns were more effective because it didn't change the robot's position; curved turns were best for speed.

Background

A Thymio robot is a 2x2 flat white robot roughly the size of a grapefruit. It has 5 front proximity sensors 2 back proximity sensors, 2 ground sensors, a 3-axis accelerometer, and 5 touch buttons that can be programmed to initiate events. Because it only has two wheels, it can't go very fast but one can program each action to be as accurate as possible and therefore efficient. Aseba is an event-based programming language designed specifically for the Thymio. To learn it, the team used the tutorials created for the course that explained the steps to make the robot perform certain tasks. They also used the Aseba Thymio website to answer any questions that arose from their work.

Program description

The program begins with a few standard conventions: we initialize all our variables (max, min, mean and cw [clockwise]) and reset all motors, LED's and timers on start, as well as putting it in the STOPPED state. The same occurs on the pressing of the back button, effectively ending the program. On the pressing of the forward button, the program begins. The program decides to either go into FORWARD, LEFT AND RIGHT state. If both sensors have similar readings, it goes into forward. If only the left sensor senses a line, it goes left. If on the right sensor senses a line, it goes right. While in these states, the robot is sensing whether to move into either of the other states. If we are going LEFT or RIGHT, it checks to see if both sensors are similar, and thus to transition of FORWARD state. It switches from LEFT to RIGHT if the conditions for FORWARD are not met and the right sensor reading is lower, and thus darker. This works the same for transitioning form RIGHT to LEFT. If we are in the FORWARD state, we constantly check if either sensor is below a threshold (senses a line) and transitions to LEFT or RIGHT accordingly.

This moving between, LEFT, RIGHT and FORWARD state comprises most of our line following program. The LOST state ties into this. Whenever the robot moves into one of these states, a timer is set for 1 second. If this second runs out before a state change occurs (which resets the timer to 1 second) then we go into a timer event. The program will go into LOST state if both sensors values are high (meaning neither sees a line at all), which makes it back up and search for a line. Once it finds a line, it transitions back to LEFT or RIGHT state.

The object avoidance program begins when a blockage is sensed when the robot is either in LEFT, RIGHT or FORWARD state. This blockage is sensed when any of the horizontal sensors have a value above zero, changing int BLOCKED state. The robot then turns either right, if the blockage is closer to the left side, and left, as a default or if the block is closer to the right side. The variable cw is set here to either 1 or 0, which will later determine which way the robot must turn to realign itself once it re-find the line. It goes forward for a fraction of a second, which makes sure it will not hit the block, and then begins to dodge: a semicircle maneuver with one motor at full speed and the other at 200. As it dodges, it is in the FIND_LINE state, continuously searching for the line, as well as having a mechanism where if it senses the block it is dodging, it will pivot away from it before continuing its dodge. As said before, once it senses a line, it turns either left or right to realign itself with the line. This occurs in a state called ALIGN, where the robot cannot transition into LEFT, RIGHT or FORWARD; it must turn for a period of time. Then it goes into LEFT or RIGHT, depending on the value of the cw variable, which was determined when the robot first turned after sensing a block.

Together these two main programs of line following and block dodging makes up the control system of our robot. The fast correction using our LOST state also creates robustness in the program. The use of mainly sensors, specifically in turning away from the block, makes the

program flexible, instead on using strict timers that have cases where they fail. The program, while slow in line following, is reliable and works almost 100% of the time. There are extra steps in block avoidance as well, such as going forward a bit after truning away from the block, which may not always be needed and thus made the robot slower, but it aids the robot be effective even in worst case scenarios. The program, in conclusion, works best for reliability while making sacrifices in speed.

Results

For our first trial, we decided not to take any risks with speed to ensure the robot's reliability and accuracy with dodging the blocks. This way, the first trial would at least be successful and then we could experiment with the later trials. Doing this was successful. The robot dodged every block and never got lost, but the finish time of 59 seconds was a little on the slow side. However, from observing the first trials of other teams who had faster times, their robots were a lot less successful in avoiding the blocks.

During the second trial, we decided to change the THRESHOLD variable to ensure the robot would go to the forward state more often. The robot was already at maximum speed, so by making the robot go into the forward state more, it will have short bursts of forward acceleration in between wiggling to find the line. It did, however, slightly touch the block and at one point it did get lost. The LOST protocol worked perfectly; the robot recognized it was lost immediately, turned blue, and backed up until it saw the line. Even though it got lost, this trial was finished faster than the previous trial with a time of 57 seconds.

The last and final trial was by far our most successful. The sensor values were updated so that they didn't have to be so precise for the robot to move forward; before, they couldn't have a difference of greater than 200 but now the difference allowed up to 500. Because of this, the robot moved forward way more efficiently than the previous time without compromising any of its accuracy. Our final trial was completed within 50 seconds, an amazing feat considering the fastest completion was 42 seconds and not many robots were faster than ours while still maintaining accuracy.

Conclusion and Future work

Overall, we are very pleased with our findings and the results of the trial. We were able to program an accurate maneuvering technique for the robot that was consistent and reliable, and do so in a timely fashion. In the future we would've liked to speed up the robot even more by perhaps trying new turn methods or refining our forward state.

References

Trappenberg, T. (n.d.). *Tutorials_part1*[Pdf].

Thymio & Aseba. (n.d.). Retrieved October 18, 2018, from https://www.thymio.org/en:thymio