TI RSLK Maze Competition

Final Report

Submitted on

November 22nd, 2019

Just Some Engineers

Group ID: PA43G01

Cristian Gutierrez - Secretary, Mediator

Samuel Hewitt - Project Manager

Cliff Porter - Programmer

Trevor Babb - Research Lead

Jacob Butcher - Correspondent

**Contents**

Abstract………………………………………………………………………… 3

List of Figures………………………………………………………………….. 4

List of Tables…………………………………………………………………… 4

Summary……………………………………………………………………….. 5

Introduction…………………………………………………………………….. 5

Methods, Assumptions and Procedures………………………………………... 5

Results and Discussions………………………………………………………... 7

Conclusions…………………………………………………………………….. 8

Recommendations……………………………………………………………… 8

Works Cited……………………………………………………………………. 9

**Abstract**

This experiment was designed to test the directional capabilities of the robot contained within the Texas Instruments Robotics Learning System Kit, and the general coding abilities of the UCF Engineering students involved. These abilities were to be tested by coding a TI-RLSK to run through a simple maze, point A to point B. The objective of the competition itself was to code your robot to run the fastest time through the maze. The scope of the coding involved included a modular code which could be tweaked to suit the individual maze. Much of the coding was done by team members Cliff Porter and Trevor Babb, who created a code from scratch to meet the scope . After a few weeks of coding, the robot was found to successfully run through the test maze. During the competition the robot ran an impressive final time of 12.608 seconds, which ended up being the time to beat. We considered this a huge success, considering our expectations were for the robot to run an average time between 30 seconds and one minute.

**List of Figures**

Figure 1: “Move Forward” block 2

Figure 2: Flow chart of the pseudo code 3

**List of Tables**

Table 1: Test Data for Time of Maze Run 8

**Summary**

The key objective under investigation during this project was to program a robot that would successfully run through a maze whilst maintaining the fastest time among our peers, which required the use of flexible code to adhere to those parameters. Lastly, the scope of the project simply included entry level coding and data recording.

**Introduction**

The TI RSLK Maze Competition is a robotics competition composed of incoming engineering students at the University of Central Florida. Throughout the semester, the required methods and procedures for this competition include creating a team, assigning roles to members, building the robot, running diagnostics, writing code, and keeping reports of any information regarding our progress. At the end of the semester, students are challenged to use their final iteration of code to run their robot through a randomly selected and customized wood maze. The goal in this maze is to commence in one of the possible starting points, navigate through the various passageways, and ultimately arrive within the end zone. The purpose of this project is to develop team building practices and skills, to familiarize oneself with basic mechanical and computer engineering components, and to apply one’s skill set in order to complete a variety of tasks.

**Methods, Assumptions and Procedures**

**Methods:**

As for the roles, the criteria was adhering to the strengths of the group members. Thus, the lead programmer position was given to Cliff due to his extensive coding background, and Trevor was instated as the research lead due to similar reasons. In this sense, they were encouraged to work off of each other when coding. Cristian would be assigned both secretary and mediator so that he could focus one what assignments were due and relaying that information to the team as a whole. Jacob was given the job of correspondent due to his strong writing and revision skills, and finally, Samuel was assigned as project manager due to his schedule flexibility and his ability to mediate between our core coding group and the documentation group.

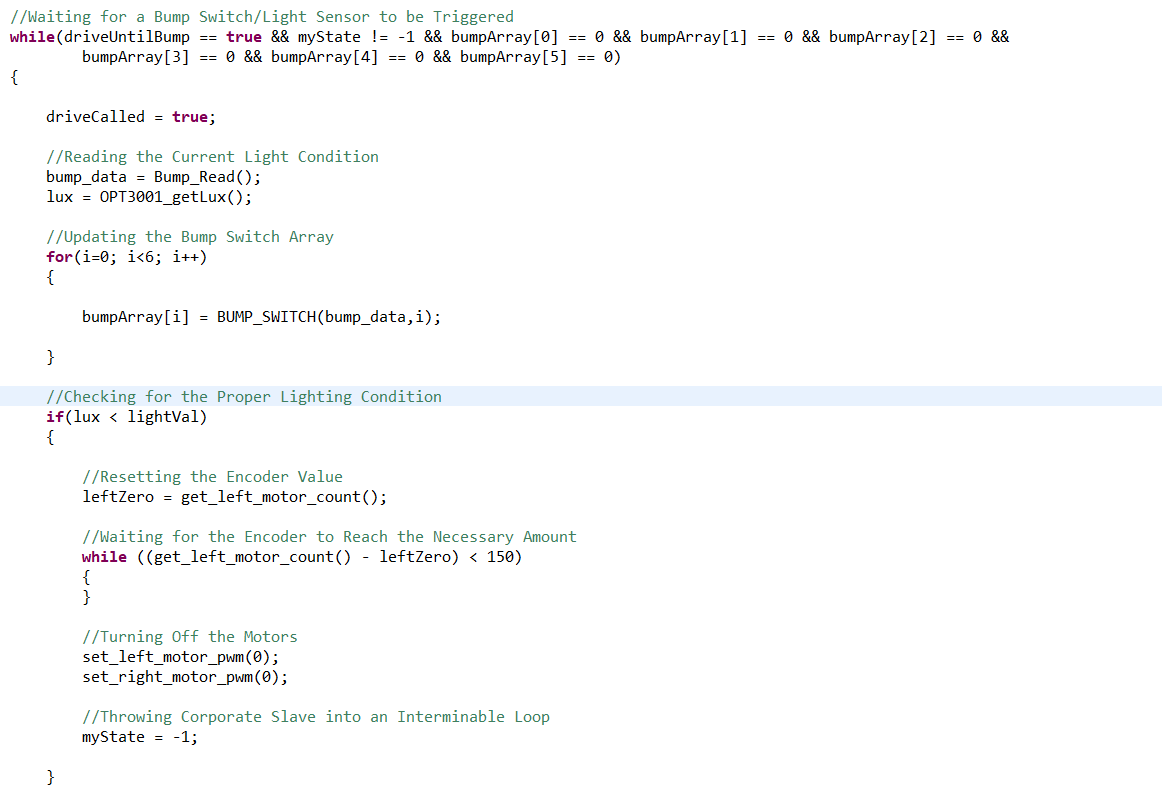
Concerning the code itself, it was decided that the best course of action would be to use modular and adjustable code, meaning that the code needed to be designed specifically for the ability to plug in information or different small lines of code. Modular code essentially allows the group to make small changes in an easier fashion because of its general format. This meant that modular code was very flexible to changes such as speed and distance units, proving to be useful as rewriting code would take longer.

**Assumptions:**

We assumed, or rather accounted for, the fact that the course was either going to be a closed loop or an open loop. A closed loop would entail that eventually, if a drive forward and turn action were repeated, then you would eventually end up at the ending. An open loop, in contrast, entails that even if you were to program a repeating action, that your robot would be stuck within a cycle without ever reaching the end goal. Based upon that assumption, action was taken to ensure that the code could be quickly adjusted to adhere to both scenarios.

**Procedure:**

Initially, it was determined by our lead programmer as well as by our research lead that the code provided to us initially had to be scrapped, and that a new code would have to be made from the ground up. The reasoning behind this choice was that the original code lacked optimization, and the way in which commands and objectives were executed tended to not follow any sound logic. Therefore, a new project file was started in CCS and new code was typed up. The first block consisted of evaluating the values of the bump switches, setting encoders to zero, and finally running the motors at a set amount of clicks. Below is the section of code that runs while the bump switches are not pressed.



**Figure 1: “Move Forward” block**

Shown here is the while loop that instructs the robot to drive forward until a sensor is triggered.

Also, the moment the code is started, the light sensor is always being evaluated to see if the robot is within the finish zone. Once it reaches the zone, all functions will cease and the program will end. However, if that is not the case, then the code will check the distance the robot has travelled. If the robot has travelled further than 2.5 squares, which was tested to be 3 feet, then it will turn 180. If not, it will execute the selected turn, which is dependent upon what the maze is and what we find appropriate to plug in to solve said maze. Code is also written so that the robot can make minute adjustments when the robot collides with a wall from the side, as well as jump to the selected turn part when both bump switches directly at the front are pressed. A flow chart of the pseudo code can be seen below for reference.



**Figure 2: Flow chart of the pseudo code**

**Results and Discussions**

The initial testing results of the TI-RSLK unit are listed below. The robot ran through the mazes in 30, 25, 40, and 20 seconds, coming out to an average of 29 seconds, which allowed us to reach the conclusion that it would most likely run through the competition maze in approximately 29 seconds. However, we found the robot to run through the maze during the competition in a mere 12.608 seconds, which remained the time to beat at the end of the competition during our lab section.

**Table 1: Test Run Completion Times of Soundwave layout**

|  |  |
| --- | --- |
| Test # | Time |
| 1 | DNF |
| 2 | 30 seconds |
| 3 | 25 seconds |
| 4 | 40 seconds |
| 5 | 20 seconds |
| **Average** | **29 seconds** |

**Conclusions**

At the time of the final tests, the group found that the robot was successful in all of the mazes, and ran at consistent times.Therefore, it was generally expected that our robot would make it to the end of the maze, regardless of the time. As a result of our testing times and stable algorithms, we also expected our time to average at thirty to fifty seconds and overall the expected results were optimistic and under one minute. However, after the competition, the robot was found to have completed the maze in only twelve seconds and six-hundred and eight milliseconds. This came as a genuine surprise as a result similar to that only happened once or twice in our testing.

**Recommendations**

Along with experimentation of the robot came many unnecessary issues and dilemmas. Many of which was the stock code. An important recommendation when participating in the competition is to **write your own code**. This is because (if you are familiar with C) the coding structure and formatting might appear more difficult to understand.Therefore, it is important to always write your own code in your own formatting to better understand what is being typed, but if the stock code is to your liking, then that should be your primary code formatting. Another recommendation is to **familiarize yourself with the capabilities of CCS**, which can be done through the instruction videos found on youtube. This way, you will know how to execute pivotal functions in your testing, such as running and debugging the code. Additionally, the importance of knowing the capabilities of the robot with allow you to realize what is efficient and what is not efficient, or what can be used at certain times.This includes the speed at which your robot should run and its effect on the battery, or the light sensor feature which plays a key role in establishing the moment at which the robot should stop within the maze.

**Works Cited**

Postell, Auree, director. *GitHub and CCS*. *GitHub and CCS*, Youtube, 17 Sept. 2019, <https://www.youtube.com/watch?v=8ohshr3gUz8&feature=youtu.be>.

Postell, Auree director. *TI RSLK Light Sensor,* Youtube, 4 Nov. 2019, <https://www.youtube.com/watch?v=3RrChydBVok&feature=youtu.be>

Postell, Auree director. *Random Mouse Algorithm*, Youtube, 28 Oct. 2019, <https://www.youtube.com/watch?v=g2QW5Qk_8oE&feature=youtu.be>

Postell, Auree director. *Lab 7 Part 6,* Youtube, 21 Oct. 2019, <https://www.youtube.com/watch?v=jyUZjTQEUT4&feature=youtu.be>